



Build a web site from scratch - Version 3



# Web Principals



Mark Webster - [www.websterart.com](http://www.websterart.com)

## A note about this book.

My name is Mark Webster. I am a teacher at Clover Park Technical College in Lakewood Washington: [www.cptc.edu](http://www.cptc.edu).

No web designer lives in a vacuum. I have studied and learned from many books about the internet, beginning in 1997 when I was dragged kicking and screaming into the computer age. This new curriculum, book if you will, was begun in 2002 when I was laid off from my one year, full time job as a web designer in downtown Tacoma. Prior to that I spent 28 years running Heidelberg multicolor printing presses around Tacoma and Seattle. I was a journeyman printer, and I still miss the smell of printing ink.

© 2019 Mark Webster. If you are taking my class, you have my permission to copy this pdf to your computer for personal use. Do not post it on the internet, or share it with your friends. Doing so will lead to bad karma and is possibly illegal. This book is not meant to be an online class. I expand and explain many of the key concepts during my lectures, this book alone will not make you a web developer

## What is the Internet

### What is the Internet?

The Internet is a vast network of millions of computers wired together all over the world via broadband networks and phone lines.

### Where does a website exist?

Each website that you visit is actually a folder on someone's computer. This folder is a special restricted folder on a specially set up computer that is left connected to the internet and turned on 24/7 so you can visit it (the web site) whenever you wish.

### What is a Server and how does it work?

The folders on these 24/7 Internet computers, called



servers, have a front door and a back door. The front door is for you, the web surfer. The back door is for the web designer who built the sit

To use an analogy, picture the window at Nordstroms. When you walk up to the window and look inside at the pretty displays, you are “surfing” the web.

You can look, but you can't touch or move anything around.

As you are admiring the Nordstrom display through the glass, you notice a little door in the display area open and a Nordstrom employee steps up into the display window and begins moving things around.

The person inside the window is like a Web Designer. She creates pretty things in the display for you to look at in the hopes that you will like them and enter the store to make a purchase.

When you look through the glass, you are looking at the display (web page) as a visitor or “web surfer” and are usually viewing an <http://www.somewebpage.com> address.

The web designer visits the same page, but comes in through a special door (or internet address), usually an <ftp://www.somewebpage.com> address. She has to enter passwords and use special software that allows her not only to look at the files on the web site as you do, but to change and update the web site.

In web terms, the “special door” is merely a couple passwords that are given to you by the company that hosts your website.



## What is the Internet (continued)

### What does a Web Designer do, in general?

A Web Designer's first task is to get a client who wants to sell their product or service on the Internet but doesn't have the technical skills. Once the client answers a number of difficult questions, the Web Designer will begin to gather the content required for the Web Site.

This can be a long, drawn out process. For many businesses, creating a Web Site can be like writing a book about their company. Most business professionals are not skilled writers and you may need to sub contract the text content out to a professional copywriter. If you are not skilled with a digital camera, you may also need to sub contract out the Photography.

The web designer will build the page on her own computer in a folder on her hard drive.

Rather like rehearsing a play in a rehearsal hall, she experiments with different ideas on her local hard drive where no one else can see it. Once she has made it as pretty as she can, she gets approval from her customer (this can be a long process with many revisions).

She will check it for bugs, test it in different browsers and operating systems, including Smartphones (iPhone, Android) then, when everything is perfect, she will use her software and passwords to update the website, (upload the new files to the server)

To return to the analogy of Nordstroms: she will open the door in the back of the display window and carry her new display into the window room where you can see it.

In technical web terms, she will upload her files from her local computer to the ISP who is hosting her site.

All web sites work this way. They exist both on the Internet where you can view them, and also on the web designer's computer where they were originally created.

### What is a Web Page and what is a Browser?

All the billions of web pages on the Internet share one thing in common, they are all pages of text with coded messages telling a browser what to do. Every web page can be opened in a word processor, and many pages are built in a simple word processing program like Notepad.

All web pages (with a few notable exceptions) end in the extension: **.html**

Whenever you click on a web page on the Internet, or even on your local hard drive, Windows XP reads the extension on the file, sees that it is **.html** and remembers that it is supposed to open anything ending in **.html** with the default Internet browser, usually Internet Explorer.

Your browser reads the code the web designer has written, finds the image files described in the code, reads the measurements in the code that tell where the various images and text should go on the page, then puts it all together in living color.

### Do I have to be a programmer to be a Web Designer?

No. Programming alone will not build a marketable web page. Web Design requires an interesting mix of salesmanship, creative thinking, Photoshop skills and low level programming. Most Web Designers end up getting pretty good at programming in the end, but high level programming skills are not an absolute requirement.

### What exactly is programming code?

## What is the Internet (concluded)

Code is what humans write when they need to speak to machines (computers). A web designer creates files in a text editor, such as Notepad, that end in the extension **.html**

The text in these html files is specially written so the browser software can “read” it (technical term is: parse). The Web Designer is, in effect, speaking to the machine and the machine (the browser software running on the computer) requires very specific letters and symbols written in a specific order. Writing code like this that a machine can understand is called programming html.

### Are there software programs that will write the code for me?

Yes. Dreamweaver is currently the most popular, but it is not the perfect answer.

As the web matured around 1997 and the pages became prettier, the code became harder and harder to write in Notepad.

When Graphic Designers began to enter the Web Design field to help the programmers make “pretty” pages, they demanded software that would allow them

to create web pages without learning a foreign language (html code).

Software programs were developed that could write reasonably decent html code and allowed Graphic Designers to do Web Design in an environment that felt a lot like Microsoft Word. They could use menus to insert pictures and tables without knowing any html code.

Unfortunately, these programs, collectively called “wysiwyg’s” (What You See Is What You Get) had their limits. Web Designers often find that a problem impossible to fix using Dreamweaver’s menus is easy to fix in Notepad.

As a result, most experienced Web Designers learn to build basic web pages in Notepad first. After they get comfortable with the logic of programming, most upgrade to a code editor that offers syntax highlighting and code hints. Dreamweaver, Brackets, Sublime and Textedit are some of the popular editors.

When it comes to html, the human mind is usually smarter than a software program.

~ ~ ~

## Lesson #1: tell a story with pictures and words

- ▶ Build a webpage in Notepad
- ▶ Use headers and paragraphs
- ▶ Write a couple paragraphs about yourself: family, work, hobbies, sports, previous schools...tell us who you are.
- ▶ Put a few photos of yourself or your hobbies on your webpage. You can save them down from Facebook, or from a phone or digital camera.
- ▶ Present your webpage to the class (if you are taking my class in college)

## Writing your first HTML 5 code

Dreamweaver, Sublime, Atom, Brackets and a couple others are programming text editors used by professional Web Developers. They are great programs, but for beginners, they can be confusing. Fortunately, long before Dreamweaver and the others existed (mid-nineties), Web Designers were building web pages in a simple little utility that has come with operating systems since before Windows 95. It's called Notepad.

NOTE: if you have a Mac, it's called textEdit or TextWrangler (free download)

Dreamweaver tries to manage our html for us, and usually does a great job, but it is just a software program.

Your mind, armed with some basic training, is a far better tool than any software program. There are many times when a little knowledge of code logic in a simple text editor like Notepad, or TextWrangler can easily fix a problem that is next to impossible to fix with Dreamweaver.

**STEP ONE:** To open Notepad on your computer:

click the **start button** > **all programs** > **accessories**. You can also simply type notepad in the search box right above the start button.

**STEP TWO:** In the blank Notepad window, type these three lines.

**NOTE:** The first line **<!doctype html>** tells the browser (Internet Explorer, Firefox, etc.) that the code it is about to read (parse) in the Notepad file is written in the language called HTML5.

```

<!doctype html>
<html lang="en">

</html>

```

The "less than" and "greater than" characters **< >** tell the browser that the words enclosed therein are strictly for programming the browser, they are not meant to be seen on the web page.

Words inside enclosing "less than" and "greater than" characters **<!doctype html>** are like the engine under the hood of your car. The engine has to be there for the car to work, but the engine should not be seen by the people driving the car.

Because "less than" and "greater than" are tedious to say, and to type, I will call them "brackets". As in: "put starting and stopping brackets around that word", like this: **<!doctype html>**

The second line **<html lang="en">** tells the browser that the language coming up is HTML. And it is the English version of HTML "lang="en"", as opposed to Spanish, or Russian.

The last line **</html>** tells the browser that you are done writing programming code, and it should consider the page finished...no more code will be written. The browser can begin displaying the webpage, based on the instructions you programmed in the HTML.

(continued on next page)

## Saving your Notepad file with .html extension

There are many different flavors of HTML, dating back to the early 1990's. We will be writing primarily in HTML5, but I will expose you to the older languages as well.

There are many legacy pages out on the Internet, and you might be asked to work on one. It can be faster to simply maintain a web page in it's current (old) language, rather than converting it to HTML 5.

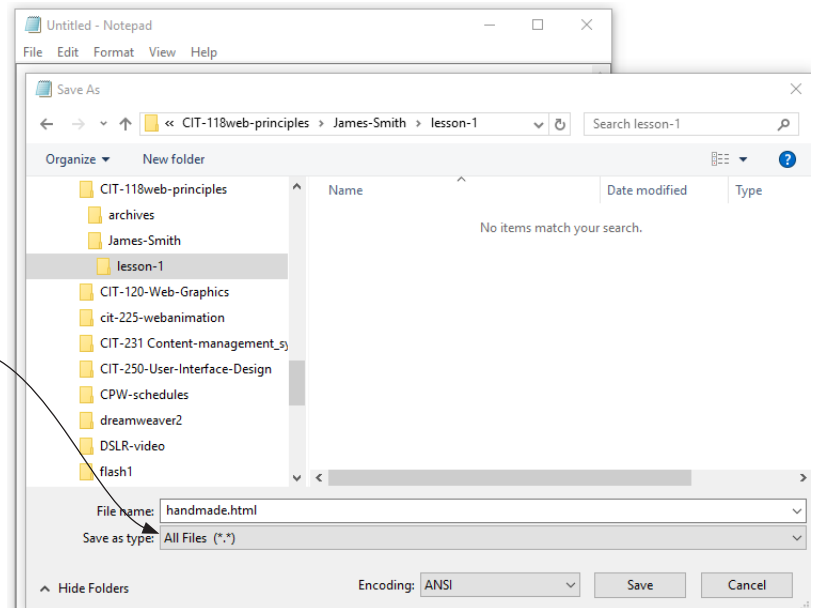
As an example: think of a house with electrical wiring from the 1940s. Perhaps an electrical outlet has gone bad. The house really needs to be rewired up to code, but that is very expensive. It's simpler and faster to just fix the bad outlet rather than rewiring the whole house.

Older versions of HTML code work fine. Browsers are very forgiving. But if you are creating a new web page it's best to write in the latest language: currently HTML 5 and CSS 3. The newest language allows you to take advantage of the new technologies and features that are constantly being developed and deployed on the Internet.

**STEP ONE:** In Notepad (or TextWranger - Mac), click: File > Save, but **be careful**, there are several things you have to do before saving it.

**STEP TWO:** Below the file name box there is a drop list that says: **Save as type:**  
Drop that list and choose: **All Files (\*.\*)** This allows you to type your own extension of **.html**, instead of the default Notepad **.txt** extension. A **.txt** extension will **not** work.

**STEP THREE:** When it asks you where to save it, create a new folder named **lesson-1** and save the file in there with a name of **handmade.html**.



**NOTE:** if you are taking my college class, you will need to create your personal folder (James-Smith) on the campus network, and create a **lesson-1** folder inside that:

\\ms1729\shared-storage\.....\room107storage\mwebster\cit-118\James-Smith\lesson-1\



## Adding in the head, title and body

The reason you put the extension “html” on your “handmade.html” page is that your operating system needs to know what software to use to open the files you save to your hard drive. When you double click on a file ending in “.html”, your operating system looks at the extension on the file, sees the “.html” and knows it needs to open the file with your default Internet Browser, usually Internet Explorer, Chrome or Firefox.

A Browser is a software program that has been programmed to respond to simple “tags” that are written (or coded) by web designers in the html files.

Back in notepad, notice how the words we’ve typed are enclosed in brackets.

And notice how the last word “</html>” has a forward slash after the first bracket.

The first word “<html lang=“en”>” is the starting “tag”, and the second word “</html>” is the stopping “tag”.

When we type these html “tags”, we are communicating with a machine (the browser software which runs on your computer). The starting and stopping tags are like the on and off switch on a machine. If you tell a machine to start, you also have to tell it when to stop.

**STEP ONE:** Modify the code in your Notepad window until it looks like this.

**NOTE:** study this new code and notice that each time I add a new tag like <head>, <title> or <body>, there is a corresponding matching stopping tag </head>, </title> or </body>. There may be words in between the starting and stopping tags, or even other tags nested inside parent tags, like this: <head> <title></title></head>, but there can be seen clear order of starting and stopping tags. Without this clear order of starting and stopping HTML tags, your programming can fail. The browser won't understand what you are commanding it to do.

**STEP TWO:** Save your Notepad file again.

**STEP THREE:** For any web page to display correctly, these tags in this order are required. Your page may work here at school with some of these html tags missing or out of order, but this is only because we have the latest browsers. Newer browsers are very forgiving of code errors. However, many people use older browsers because they are afraid of installing new software. You probably know someone who has a 5 year old computer. They refuse to buy a new one because “it’s working fine, why would I want to replace it?”. As Web Designers, we need our code to be as clean and logical as possible.

```
<!doctype html>
<html lang="en">

<head>

    <title>

    My page title goes here!

    </title>

</head>

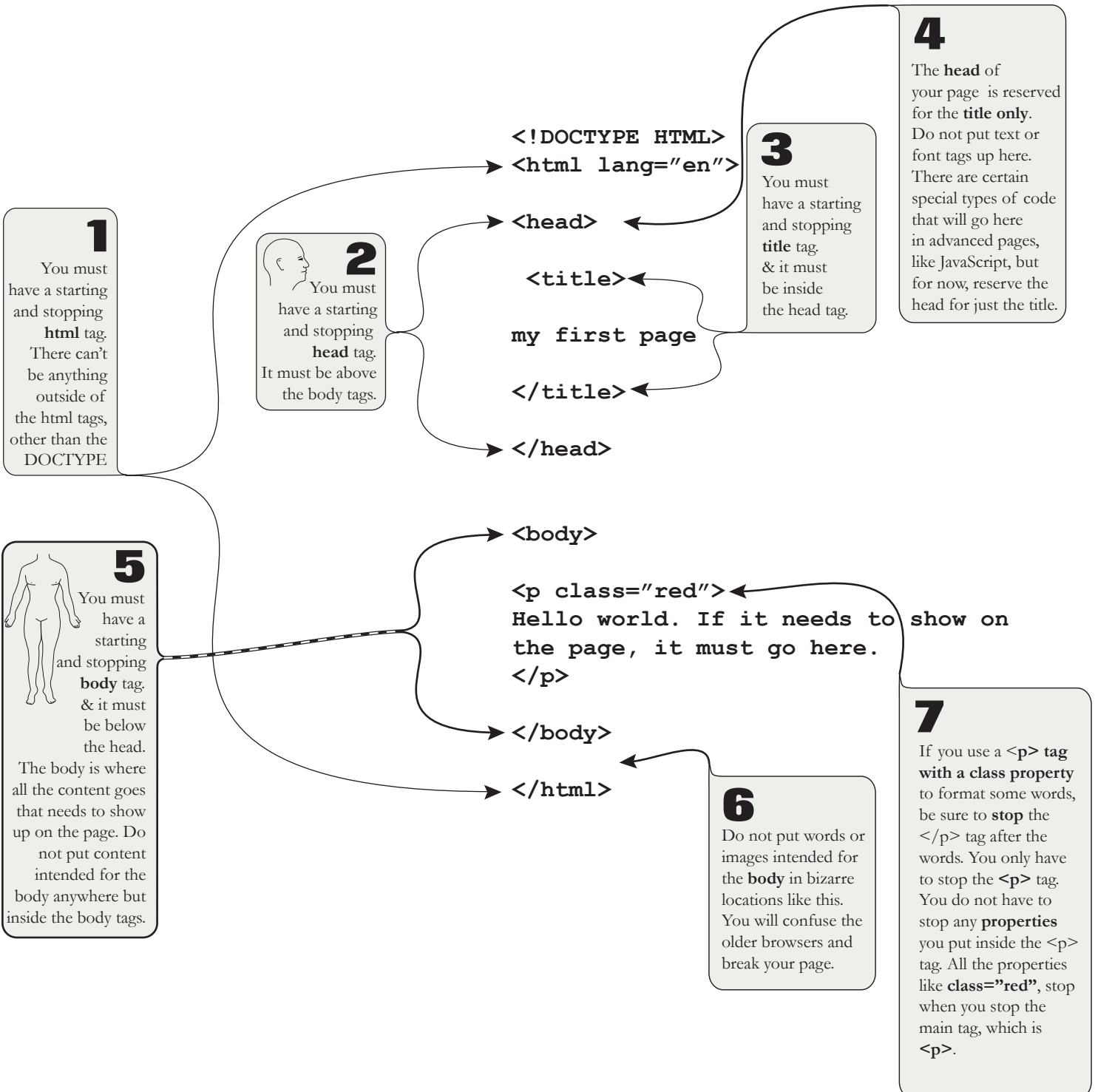
<body>

    Hello World. This is the body of
    my webpage where I can type
    anything I want!

</body>

</html>
```

## Checklist for common HTML code errors



## Viewing your web page in the browser

**STEP ONE:** Make your Notepad window very small on your desktop. A good width is about 3 inches wide and as tall as needed to display all the code.

**STEP TWO:** Use Windows Explorer (or Finder on Mac) to find the containing folder for your **handmade.html** notepad file.

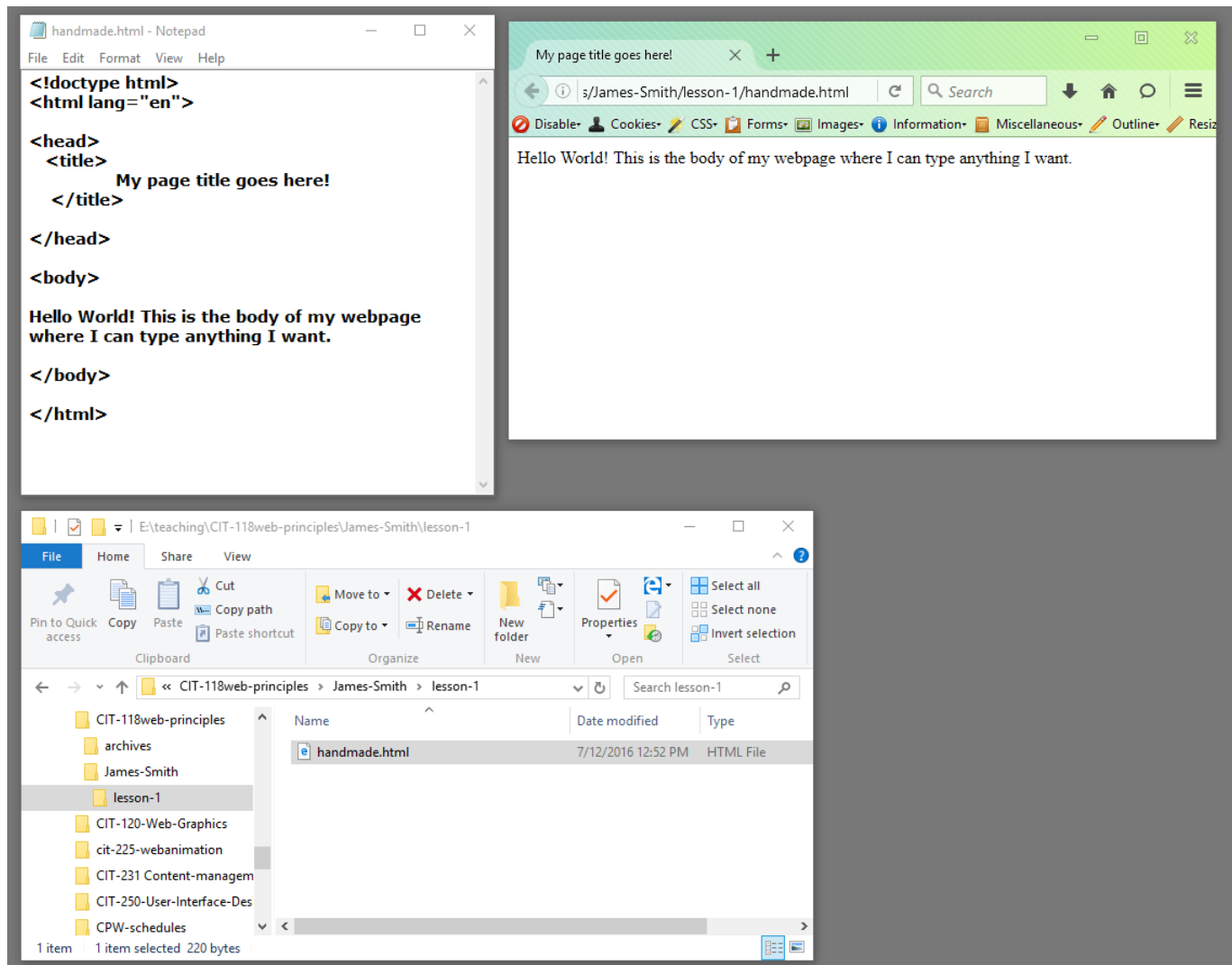
**STEP THREE:** Make the folder very small on your desktop so that you can see the handmade.html file, sitting in the containing parent folder where it lives. Keep this folder open at all times.

**STEP FOUR:** Double click on the handmade.html file. It should open in whichever browser has been told to be the default browser for your computer.

**STEP FIVE:** Make the browser window very small on your desktop. You should now have 3 windows on your desktop, arranged and separated so you can see the entire contents of all 3 windows, as shown below.

**STEP SIX:** If your page didn't open in the browser when you double clicked it, or if it opened blank, review the **checklist of common code errors** on the previous page. Also, check to see you have the correct extension on the file name. See next page for instructions on how to **turn on extensions**, or google it.

**NOTE:** each time you make a change to the Notepad file, click save and then go to the browser and hit refresh. The browser doesn't automatically refresh. It has no idea you are making changes in Notepad. When you hit the browser refresh button, the browser reads all of your HTML code fresh, and displays any changes you have programmed.



## Un-hiding extensions in Windows 10

If your handmade.html file didn't open in the browser, but instead opened up in Notepad, it's possible you missed a step when you saved the file. To fix the problem, follow these steps:

**STEP ONE:** click: View > Options > Change folder and search options

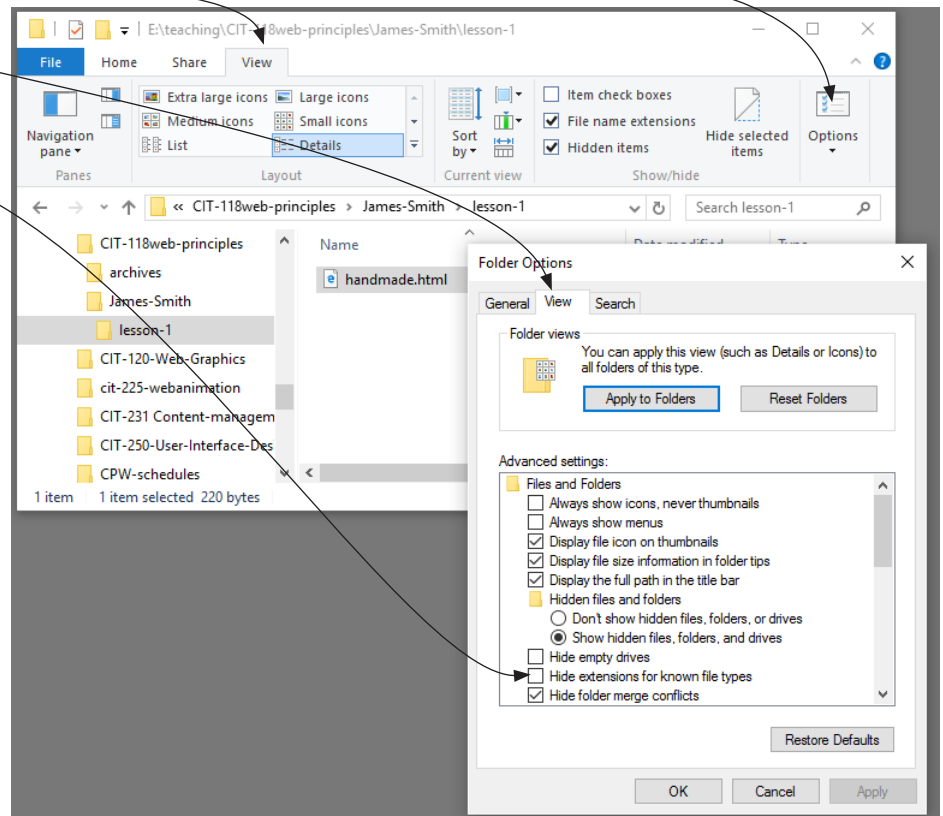
**STEP TWO:** In Folder Options, click the View tab

**STEP THREE:** Under Advanced Settings, scroll down and UNCHECK "Hide extensions for known file types".

**NOTE:** The default setting in Windows is to hide extensions. A file's extension determines which software program will open it, if it is double clicked. Problems occur when you assume a file extension is one thing, when it is actually something else. So be sure that Windows is NOT hiding extensions from you

**STEP FOUR:** After these changes, you should be able to see the entire file name, including extension. It should be **handmade.html**

If it isn't, rename the file. **NOTE:** you will need to close the Notepad file before you can rename it.





## Styling your text for SEO

If you are going to put a web page on the Internet, you want people to find it. Getting found on the internet is a huge can of worms. There are people who do nothing but modify web pages to be easily found. The people who do this are called **Search Engine Optimization** experts, or **SEO** for short.

But a few basic rules will get you started. If you build your page right the first time, it will be naturally optimized for Search Engine Optimization, henceforth called SEO. Starting inside the starting body tag, there should be a header. Under that you can put a sub header, and then you can start placing paragraphs.

**STEP ONE:** In your Notepad file, add a starting and stopping H1 tag like this `<h1></h1>`

**STEP TWO:** Type your name inside the starting and stopping `<h1>` John Smith `</h1>` tags.

**STEP THREE:** Add a subheader using `<h2></h2>` tags. `<h2>` tags are a smaller font size than `<h1>` tags, and are considered less important by the search engines, since they are smaller.

**STEP FOUR:** Add a few starting and stopping `<p></p>` tags. Inside the `<p></p>` tags, type a few sentences describing yourself. Talk about your hobbies, your previous schools, any sports you might have done, clubs you belong to, online games you like to play, basically tell your story in a few paragraphs.

**NOTE:** keep your homework PG rated, this is a college environment. We expect everyone to be civilized: no swear words, no offensive images. A good rule of thumb is to treat your classmates as you would like them to treat you: with respect and courtesy.

**STEP FIVE:** When you are done, save your page. Go to your browser and hit the refresh button. It should look like this, but with your words instead of mine.

```
<!doctype html>
<html lang="en">

<head>

  <title>
  My page title goes here!
  </title>

</head>

<body>

  <h1>
  My name is John Smith
  </h1>

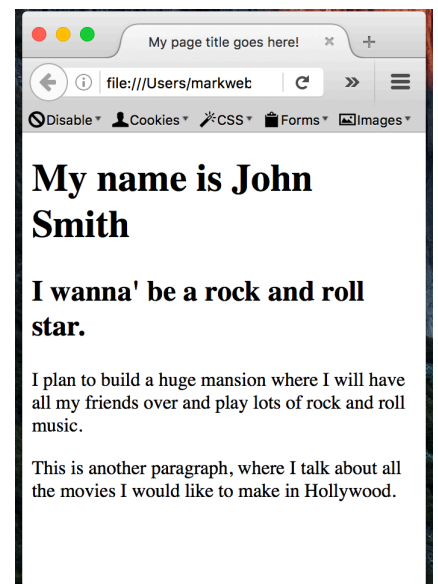
  <h2>
  I wanna' be a rock and roll star.
  </h2>

  <p>
  I plan to build a huge mansion where I will have
  all my friends over and play lots of rock and roll
  music.
  </p>

  <p>
  This is another paragraph, where I talk about all
  the movies I would like to make in Hollywood.
  </p>

</body>

</html>
```



## Placing your image on your webpage

Imagine that you are a playwright for a theatrical company. You write the script for the play that will be performed by actors and actresses in the theater. If an actor is supposed to appear from "stage left" when the play starts, he had better be standing in the wings ready to go, not down the street getting a latte.

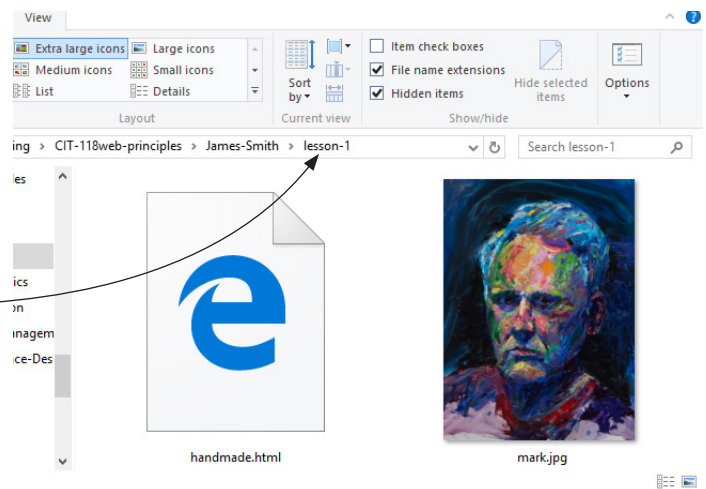
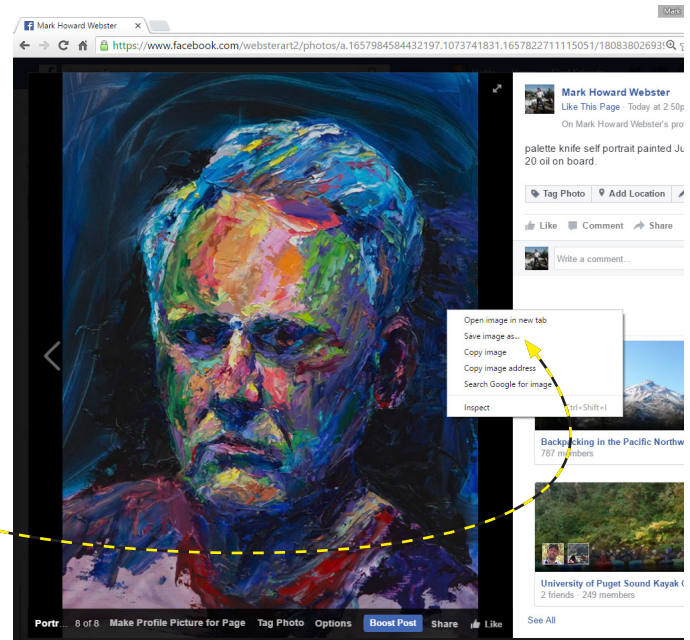
Programming HTML is similar to writing the script for a play. You list a long series of commands for what the browser should display. If you want an image to appear on your webpage, you tell the browser where the image should come from (stage left, ie: the images folder) and what its name is. By default, the image will appear on the webpage after any other HTML elements you have in the code above the image tag.

Every element on the webpage is by default floating up and left, though we can override that with code.

You are going to need an image of yourself, or your hobby. We need a jpg. I will have a digital camera, but it's easier if you get one from your Facebook, Instragram, or other social media account.

**STEP ONE:** to get your picture out of Facebook, click on a picture in a Facebook gallery to make it as big as possible, not a thumbnail. Right click on the picture and choose **save image as**.

**Change the name of the file to something simple,** like **mark.jpg**. Do not use capital letters, numbers, or spaces in the file name. The save image menu is slightly different in the different browsers, you can always ask google if you can't figure out how to save an image from a website. You do have to get the image from wherever you found it online into your **lesson-1** folder. An easy way is to save it to your desktop. Then minimize Facebook, and use Windows Explorer to copy it into your **lesson-1** folder...which you should keep open at all times. Paste the jpg into the folder alongside the **handmade.html** file. They need to be side by side files in their parent folder **lesson-1**, like two peas in a pod.



## image code

**STEP ONE:** To make an image appear, we have to write the code for image, which is `<img>`

**STEP TWO:** `<img>` by itself won't make the image appear. The browser needs more information. You have to tell it where the image lives on your computer. This is called a source property, and you write it like this:

```

```

It will not appear on your webpage unless you call it by its **exact name, including extension**. **src** stands for source. What is the "source" of this image, where does it live in relation to the handmade.html file?

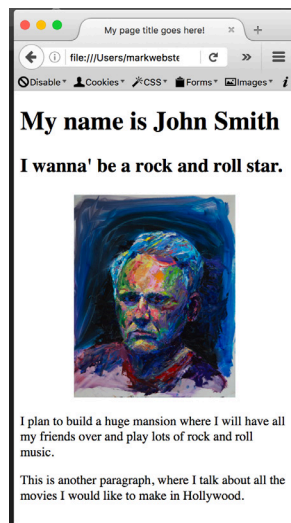
If it is in the same containing parent folder (**lesson-1**), all you have to do is put the name of the jpg including extension, inside the double quotes, like this.

The image **MUST** be called from the same folder as the html file. Websites are built in folders, you can not tell things to appear on your webpage from other locations around your hard drive, or, at least, not if you expect them to work when this web page is lifted up onto the internet.

Every thing that appears on your website must be contained within the **lesson-1** folder. This includes images, other linked html files, special fonts, pdf's, javascript files, etc.

**NOTE:** Depending on where you found your image, it may cause problems because it is too large. We will fix this later in Photoshop, but a quick fix now for an oversize photo is to add a width property to the image tag.

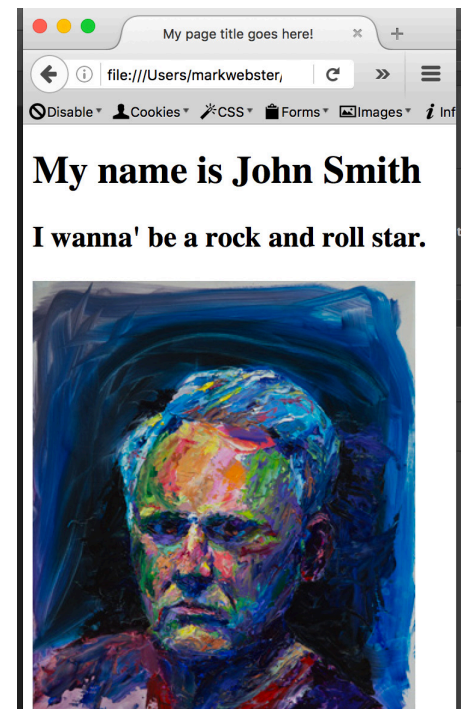
**STEP THREE:** (optional) Add a width property to the `<img>` tag that tells the image to display smaller than it's normal or native pixel width. If you'd like to center your image, you can enclose it in `<p>` tags, and add an `align="center"` property to the starting `<p>` tag as shown. There is a better way to center images using CSS style sheet rules, but we will do that later.



```
<h2>
I wanna' be a rock and roll star.
</h2>



<p>
I plan to build a huge mansion...
</p>
```



```
<h2>
I wanna' be a rock and roll star.
</h2>

<p align="center">

</p>

<p>
I plan to build a huge mansion...
</p>
```

## Text and image together

Here is the how my desktop looks right now. Note that you can see all three windows:

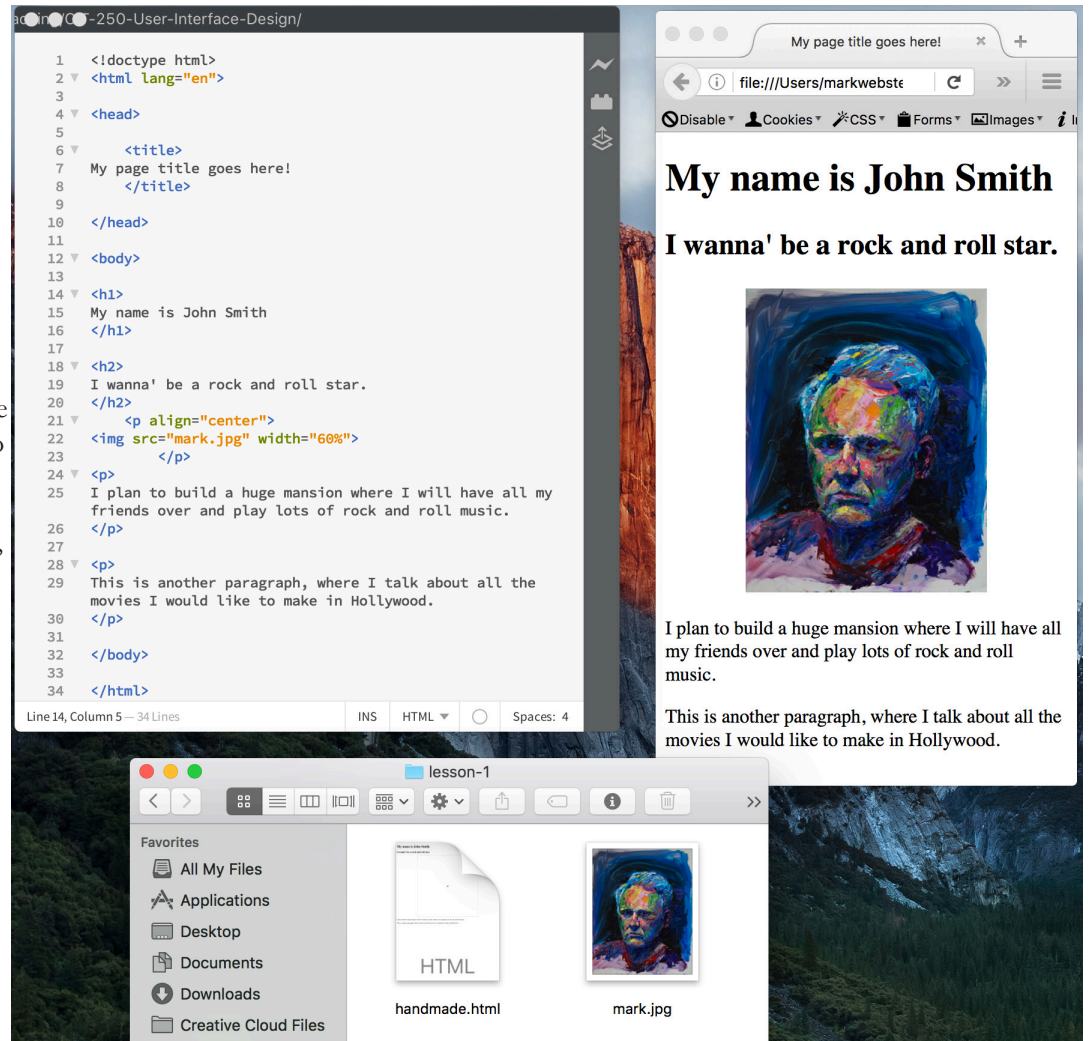
1. the notepad code
2. the parent folder: lesson-1
3. the firefox browser

If you want to have two pictures side by side, there are some advanced techniques you can try on this website:

<https://www.ghostforbeginners.com/place-images-side-by-side/>

Note that this side by side technique has problems if your images are too large, like right out of your camera.

To shrink your image in Photoshop, see page 48.



```
<div class="container">
  <div style="float:left;width:49%">
    
  </div>
  <div style="float:right;width:49%">
    
  </div>
</div>
```



# Cascading Style Sheets: CSS

We can make this page a lot prettier if we could do something with the text. To change our text color, we have to delve into CSS, or, style sheets. Lets tell the text that is surrounded by starting and stopping `<h1></h1>` tags to be red, here is how:

**STEP ONE:** In your code, between the stopping title tag `</title>` and the stopping head tag `</head>`

add this code:

Note there are two starting and stopping style tags `<style></style>`

Those style tags define the beginning and end of the style rules that will apply to any content (markup) found down in the body area of the web page. This is one CSS rule:

`h1 {color: red;}`

Selector      Declaration block

The h1 is called a selector. We are telling the browser that if it finds any h1 tags down in the body, follow the declaration inside the starting and stopping curly braces `{ color: red;}` `color: red;` is called a declaration, and it consists of a property (color) and a value (red). The colon `:` between color and red is sort of like an equals sign. You can think of the colon as the words "shall be". The semi-colon `;` after red is required. The semi-colon tells the browser that declaration is complete. There MUST be 2 curly braces in each rule!

You can also use hexadecimal colors: `color: #ff0000;`

```
<!doctype html>
<html lang="en">

<head>
  <title>
    My page title goes here!
  </title>

  <style type="text/css">
    h1 {
      color: red;
    }
  </style>

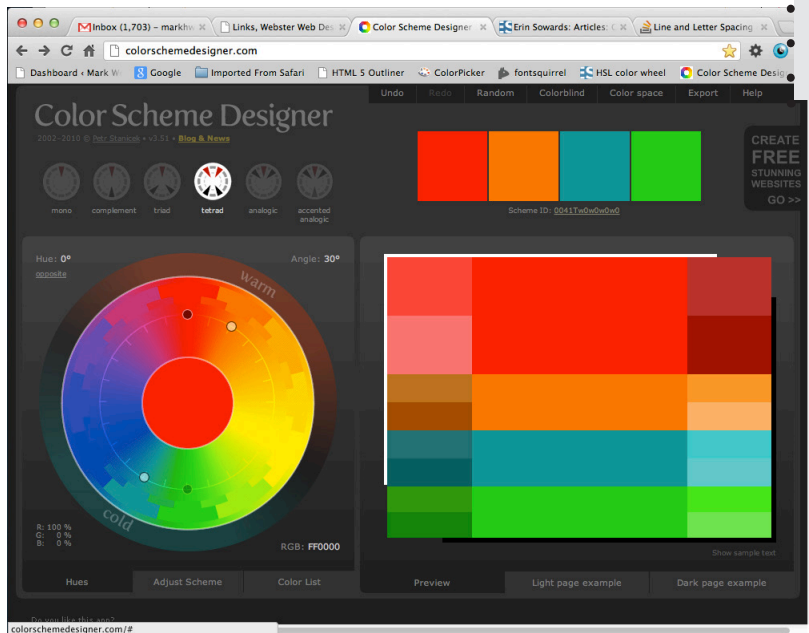
</head>

<body>

  <h1>
    My name is John Smith
  </h1>

  <h2>
    I wanna' be a rock and roll star.
  </h2>
  
  <p>
    I plan to build a huge mansion, where I will have all
    my friends over and play lots of rock and roll music.
  </p>

</p>
```



A great resource for picking cool color schemes is here:

<http://colorschemedesigner.com/>

<http://www.colourlovers.com/palettes>

You can find a very nice color picker here:

<http://www.colorpicker.com/>

## Expanding on the header CSS

Try adding these additional declarations within the h1 curly braces:

```
h1 {  
    color: #a64b00;  
    font: 2.3em Verdana, Helvetica, sans-serif;  
    letter-spacing: 0.1em;  
    border: 2px dotted #000;  
    background-color: #ffb273;  
    padding: 5px;  
    text-align: center;  
}
```

### NOTE:

I'm using hexadecimal HTML colors. You can find them online, or in the Photoshop color picker. \*\*\*<http://www.colorpicker.com/>\*\*\*

- **color** means font color
- In the font declaration, I'm combining the font-size, and my top three preferences for fonts, starting with Verdana. I'm saying: I'd like Verdana, but if you (the browser) can't find it, use Helvetica instead. If that isn't available, please at least use sans-serif.
- **letter-spacing** is just like it sounds, space between letters, AKA: tracking.
- **border** refers to the border around the header, it makes a box. It can be dotted, solid, double, etc. If all the hex numbers are the same, you only need 3 (#000)
- **background-color** is the color of the area inside the border
- **padding** is the space between the header letters and the border
- **text-align: center;** tells the text to center horizontally within the border.
- Every new declaration **must** have a colon and a semi-colon. **A missing ending semi-colon at the end is the most common CSS error you will make.** The browser gives up if it doesn't see the ending semi-colon on a CSS declaration.
- FYI: The browser will already make h1 text big, but by specifying the size in em, you get more control, and you override the default browser style sheet. An em measures about 14 pixels.

A great reference resource for all the Cascading Style Sheet (CSS) properties is here:

<http://www.w3schools.com/cssref/default.asp>

# Forcing Smartphones not to scale content

**STEP ONE:** Add these two lines of code to your Notepad file. Place them just below the starting `<head>` tag:

```
<meta charset="utf-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

NOTE: These meta tags are new in HTML 5, and they force smartphones to NOT

automatically scale the page to fit the screen (viewport). Pictured below I have a screen capture from my IOS simulator, running as part of the free Xcode package, available on the Mac App Store.

There is also a free Android simulator available for both Mac and PC at <http://developer.android.com/sdk>. It's complicated to install, but there are tutorials at [www.lynda.com](http://www.lynda.com).

Here is the same page in Firefox to right.

```
<!doctype html>
<html lang="en">

<head>
```

```
<meta charset="utf-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>
```

```
My page title goes here!
```

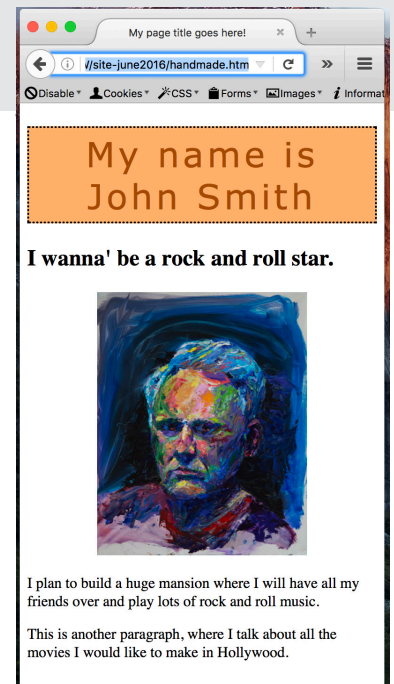
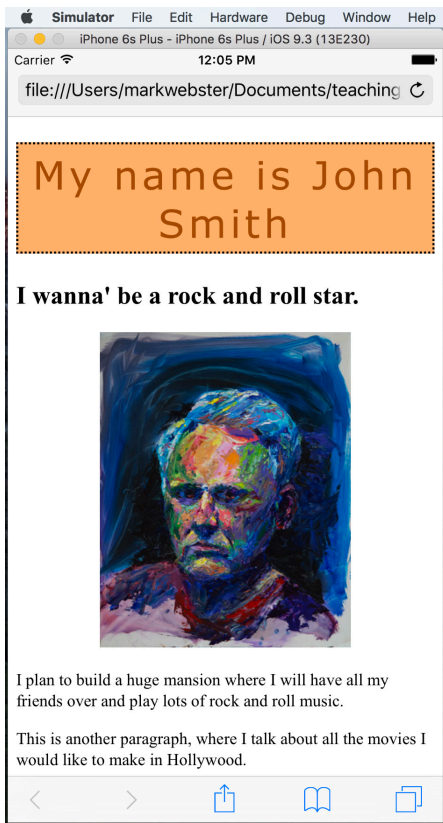
```
</title>
```

```
<style type="text/css">
```

```
h1 {
```

```
color: #a64b00;
font: 2.3em Verdana, Helvetica, sans-serif;
letter-spacing: 0.1em;
border: 2px dotted #000;
background-color: #ffb273;
padding: 5px;
text-align: center;
```

```
}
```



## CSS Box Model

Cascading Style Sheets (CSS) started out as a way to manage the appearance of all the fonts on the pages of a large website. Before CSS (1997-ish), Web Designers used font tags directly around the words they wanted to change:

```
<font face="Arial" color="red"> red words</font>
```

This led to huge maintenance issues when updates were needed, or the client wanted green words instead of red.

CSS was initially used to move the font formatting style information up into the head area of the webpage. If the website was larger than one page, the font style information was moved into an external file. This external file would have nothing but style sheet rules, and could be imported into each of the webpages, via a link that essentially says: go get these styles, and put them here in the head, even though they aren't really here. We will do that later...

Designers found that CSS could do much more than make fonts pretty, it could provide structure to the web page. It could make columns and navigation menus. Starting around 2003, CSS replaced HTML tables as a way to structure the web page.

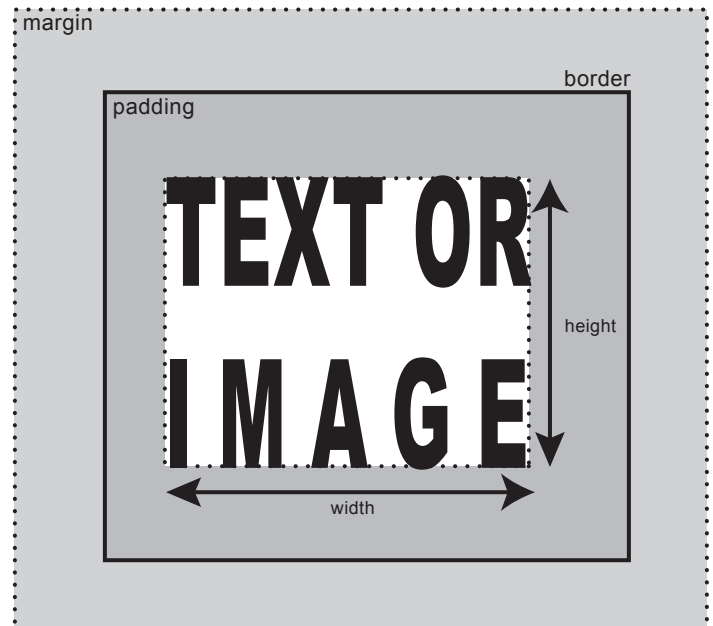
An added benefit was that a CSS structured web page could be more effectively optimized for Search Engines, and visually impaired people could more easily "read" the content using auditory screen readers.

CSS hinges on the Box Model. In the Illustration above, the words "TEXT OR IMAGE" are considered to be contained in a virtual box (the innermost dotted line). You can tell that box to have a certain width and height, or you can let the box find its dimensions naturally.

Around that first box is padding. At the outer edge of the padding there can be a border. Beyond the border you can specify how much margin there is outside of the border. The margin controls how close other objects on the web page are allowed to approach.

From the border inward you can have a background-color, or a background-image. New in HTML 5 and CSS 3 are drop shadows and alpha transparencies, which can be added to the box model mix.

It's best to learn the Box Model by doing. We could talk theory all day, but it's more fun to dive in start tossing code around, see what sticks to the wall.





## Wrapping content in a box

**STEP ONE:** Open your handmade.html file in Notepad

**STEP TWO:** Within the starting and stopping body tags `<body></body>`, add a starting and stopping `<div></div>`

**STEP THREE:** In the starting `<div>` tag, add a property so it looks like this `<div id="wrapper">`

to the right of the stopping `</div>` tag, add an HTML comment so it looks like this:

`</div> <!-- end wrapper div -->`

Here are the two new div tags in place.

A `<div>` tag is a bit like a paragraph. It forces a new line, but unlike a paragraph, you can add CSS formatting to a div tag and use it for structure.

We added the `id="wrapper"` property to the starting div tag to distinguish it from other div tags that will be on the page. We will have numerous div tags on our web page. When we need to speak to an individual div tag, we give it an `id` property.

The property (`id="wrapper"`) stops when you stop the div tag.

Because we may have divs within divs, sort of like rooms within a house, I added an invisible (to the browser) HTML comment tag to help me remember which div is stopping. This comment tag `<!-- end wrapper div -->` is only for me, and you, the programmers. Web page code gets big and long. Commenting your codes helps you to remember things. The browser is blind to comments.

An HTML comment it required to have this: `<!-- -->`

You can put anything you want between the 4 dashes, there are no restrictions.

The browser also does not see more than one horizontal space in a line, and it does not see tabs or hard and soft returns. The browser is basically blind to white space...with a few exceptions.

```
<!doctype html>
<html lang="en">

<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-
width, initial-scale=1.0">

<title>
My page title goes here!
</title>
<style type="text/css">

h1 {
    color: #a64b00;
    font: 2.3em Verdana, Helvetica, sans-serif;
    letter-spacing: 0.1em; /*comment goes here*/
    border: 2px dotted #000000;
    background-color: #ffb273;
    padding: 5px;
    text-align: center;
}

</style>
</head>
<body>
<div id="wrapper">
<h1>
My name is John Smith
</h1>
<h2>
I wanna' be a rock and roll star.
</h2>

<p>
I plan to build a huge mansion, where I will have all
my friends over and play lots of rock and roll music.
</p>
<p>
This is another paragraph, where I talk about all the
movies I would like to make in Hollywood.
</p>
</div> <!-- end wrapper div -->
</body>
</html>
```

## Controlling margins and padding

Now that we have a div with an id of wrapper, lets use that to add some style to our page structure.

**STEP ONE:** Up in the head of your webpage, find the closing curly brace at the end of the h1 style sheet rule. Click to the right of it and press the enter key several times to make some blank lines.

**STEP TWO:** Browsers by default add margins and padding around your web page content. To get rid of it, add this style sheet rule for the `<body>` tag that zeros out the margins and padding:

```
body {
  margin: 0;
  padding: 0;
  background-color: #666;
}
```

**STEP THREE:** After the closing curly brace for the new body rule, add some more white space (enter, enter, enter) and type this rule that speaks to the new wrapper div:

```
#wrapper {
  margin-right: auto;
  margin-left: auto;
  padding: 10px;
  width: 80%;
  max-width: 1400px;
  background-color: #fff;
}
```

The `#wrapper` syntax means that we are speaking to a tag that has an `id="wrapper"` property. Put another way:

`#wrapper` in the style sheet will speak to `<div id="wrapper">` in the body of our web page.

```

<!doctype html>
<html lang="en">

<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-
width, initial-scale=1.0">

<title>
My page title goes here!
</title>
<style type="text/css">

h1 {
  color: #a64b00;
  font: 2.3em Verdana, Helvetica, sans-serif;
  letter-spacing: 0.1em; /*comment goes here*/
  border: 2px dotted #000000;
  background-color: #ffb273;
  padding: 5px;
  text-align: center;
}

body {
  margin: 0;
  padding: 0;
  background-color: #666;
}

#wrapper {
  margin-right: auto;
  margin-left: auto;
  padding: 10px;
  width: 80%;
  max-width: 1400px;
  background-color: #fff;
}

</style>

</head>
<body>
<div id="wrapper">
<h1>
My name is John Smith
</h1>
<h2>

```

## #wrapper explanations

**margin-right: auto;**  
**margin-left: auto;** has the effect of centering our wrapper div (it's a box, remember?) within the containing, or parent tag, which is the `<body></body>`

**padding: 10px;** puts some white padding all the way around the inside of the wrapper div box. I may have forgot to mention that I told the body to have a grey background-color: #666;

**width: 80%;** tells the wrapper to act like an 80% spring. As the browser window is resized, it figures out what 80% is, and sizes itself to that, automatically centered on left and right.

**max-width: 1400px;** is a warning to the browser. Don't make the wrapper any bigger than 1400pixels. That is our maximum width. Any wider doesn't make sense. Studies have shown that wide web pages are very hard to read. Think of newspaper columns.

```

<!doctype html>
<html lang="en">

<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>
My page title goes here!
</title>
<style type="text/css">

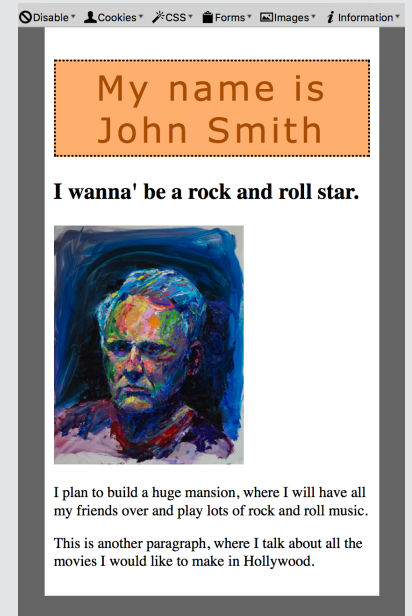
h1 {
    color: #a64b00;
    font: 2.3em Verdana, Helvetica, sans-serif;
    letter-spacing: 0.1em; /*comment goes here*/
    border: 2px dotted #000000;
    background-color: #ffb273;
    padding: 5px;
    text-align: center;
}

body {
    margin: 0;
    padding: 0;
    background-color: #666;
}

#wrapper {
    margin-right: auto;
    margin-left: auto;
    padding: 10px;
    width: 80%;
    max-width: 1400px;
    background-color: #fff;
}

</style>
</head>
<body>
<div id="wrapper">
<h1>
My name is John Smith
</h1>
<h2>
I wanna' be a rock and roll star.
</h2>

```



## Centering with a .class style sheet rule

You may have noticed that you can't easily center your image on your web page horizontally. A hack would be to put starting and stopping center tags around the image tag: `<center> <img> </center>`. But that is frowned on in HTML 5. A better solution is to create a custom style sheet rule for centering objects on our webpage.

Up until now we've added style sheet rules that affected tags (h1, h2, p, #wrapper) that were already in the body area of our code.

You can also write a style sheet rule that is more universal. It's called a style sheet "class", and here is how to use it.

In the head of your web page, within the starting and stopping `<style></style>` tags, add this new style sheet rule after all of the other style sheet rules. Remember, style sheet rules go up in the head of your webpage, not down in the body of your page:

```
.center {
  display: block;
  margin-right: auto;
  margin-left: auto;
}
```

### EXPLANATION:

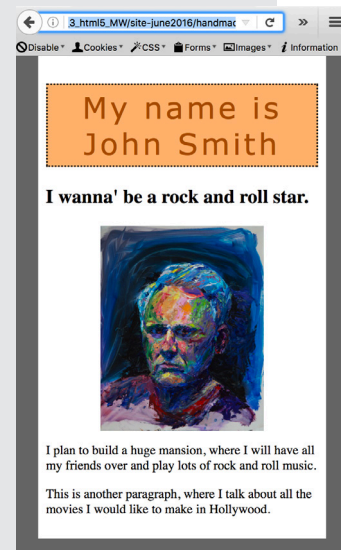
**display: block;** forces a behavior like a paragraph tag  
**margin-right: auto;**  
**margin-left: auto;** has the effect of centering whatever it is applied to within the parent element, which in this case is the `<div id="wrapper">` element

**STEP ONE:** Inside your image tag, add a space before the closing bracket (`>`) and add this property:  
**class="center"**

**STEP THREE:** Save your code and refresh your browser window, your image should now be centered horizontally.

```
<style type="text/css">
h1 {
  color: #a64b00;
  font: 2.3em Verdana, Helvetica, sans-serif;
  letter-spacing: 0.1em; /*comment goes here*/
  border: 2px dotted #000000;
  background-color: #ffb273;
  padding: 5px;
  text-align: center;
}
body {
  margin: 0;
  padding: 0;
  background-color: #666;
}
#wrapper {
  margin-right: auto;
  margin-left: auto;
  padding: 10px;
  width: 80%;
  max-width: 1400px;
  background-color: #fff;
}
.center {
  display: block;
  margin-right: auto;
  margin-left: auto;
}
</style>
</head>
<body>
<div id="wrapper">
<h1>
My name is Elvis Presley
</h1>
<h2>
I wanna' be a rock and roll star.
</h2>

<p>
I plan to build a huge mansion called Graceland, where
I will have all my friends over and play lots of rock and
roll music.
</p>
<p>
This is another paragraph, where I talk about all the
movies I would like to make in Hollywood.
</p>
</div> <!-- end wrapper div -->
```

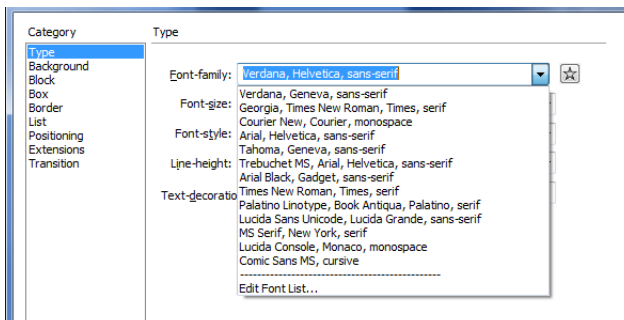


# Serif headers, sans-serif body

When the Internet became popular, testing was done to find the most readable font. Researchers discovered that if you have a lot of body copy, like a long story, or blog post, it should be displayed in a **sans-serif** font like Verdana, Arial, or Tahoma, among others.

**Serif** fonts like Times Roman have been used for centuries in Print, primarily because they last longer on the printing press. But those very serifs (that make them so durable on long printing press production runs) look muddy on computer screens.

Another problem with using fonts on web pages is that you never know whether your end user will have the same fonts as you. You can get any font to display on your web page, if it is installed in the fonts folder of the machine viewing the web page. Again, research was done, and these fonts are safe to use, as long as you include the listed second, third and fourth choices. (This list is from Dreamweaver)



## Rule of Thumb:

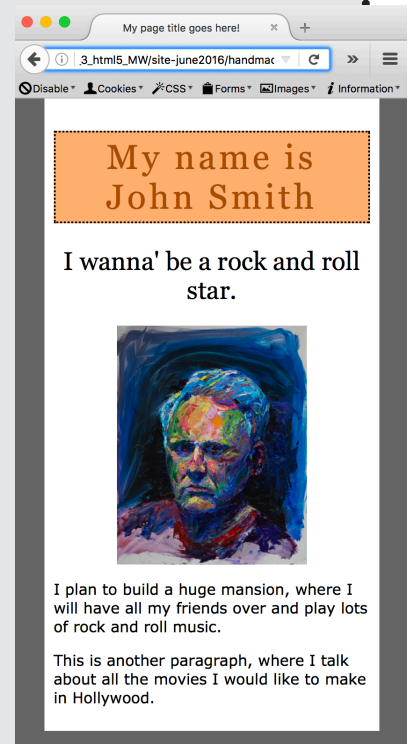
- Use serif fonts for headers
- Use sans-serif fonts for body copy

In HTML 5 & CSS 3, it is possible use something called Web Fonts, which allows many more fonts, but it is something we will do later on, after we master more basic concepts, like columns. But here is a teaser: <https://www.google.com/fonts>

**STEP ONE:** With these rules in mind, make some changes to the h1, h2, and body style sheet rules. They are shown in **bold brown** in the code to the right.

```
<style type="text/css">
h1 {
    color: #a64b00;
    font: 2.3em Georgia, Times, serif;
    letter-spacing: 0.1em; /*comment goes here*/
    border: 2px dotted #000000;
    background-color: #ffb273;
    padding: 5px;
    text-align: center;
}
h2 {
    font: 1.7em Georgia, Times, serif;
    text-align: center;
}
body {
    margin: 0;
    padding: 0;
    background-color: #666;
    font: 1em Verdana, Helvetica, sans-serif;
}
#wrapper {
    margin-right: auto;
    margin-left: auto;
    padding: 10px;
    width: 80%;
    max-width: 1400px;
    background-color: #fff;
}
.center {
    display: block;
    margin-right: auto;
    margin-left: auto;
}
</style>
</head>
<body>
<div id="wrapper">
<h1>
My name is Elvis Presley
</h1>
<h2>
I wanna' be a rock and roll star.
</h2>

<p>
I plan to build a huge mansion, where I
will have all my friends over and play lots
of rock and roll music.
This is another paragraph, where I talk
about all the movies I would like to make
in Hollywood.
</p>
I plan to build a huge mansion called Graceland, where I
will have all my friends over and play lots of rock and roll
music.
</div>
</body>
</html>
```





## Parent Child relationship in programming

Our web page is looking fine, but it's quite simple: nothing is clickable, and there is very little room for expansion, other than vertically. Web pages (and web sites) are meant to be interactive, and they normally have multiple pages such as home, gallery, resume, contact, etc.

### CSS Box Model

Before we start turning this single web page into a multiple page website we need to design a structure on our web page that will make it adaptable to the many different types of content we may display. We also want to make sure that our webpage looks good on all devices, from smartphones (Android and iPhone) to 20 inch computer monitors.

In our current design, our only "structure" is our starting and stopping wrapper div tag.

```
<div id="wrapper">
content goes here
</div>
```

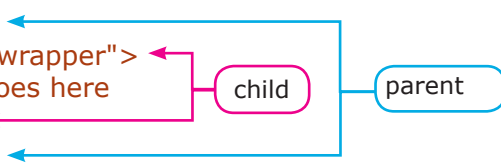
This div tag, with an id property of "wrapper" can be thought of as a virtual box. The browser considers it to be a box, hence the term:

**CSS Box Model.** We have told this wrapper box, in the CSS styles to be

**width: 80%;**

When you command a box to have a width, the browser looks at the code within the body of the webpage to see where the box lives. It determines the correct display width of the box based on the available space within the parent element. On our web page, the parent element, (HTML tag) is the `<body></body>` :

```
<body>
<div id="wrapper">
content goes here
</div>
</body>
```

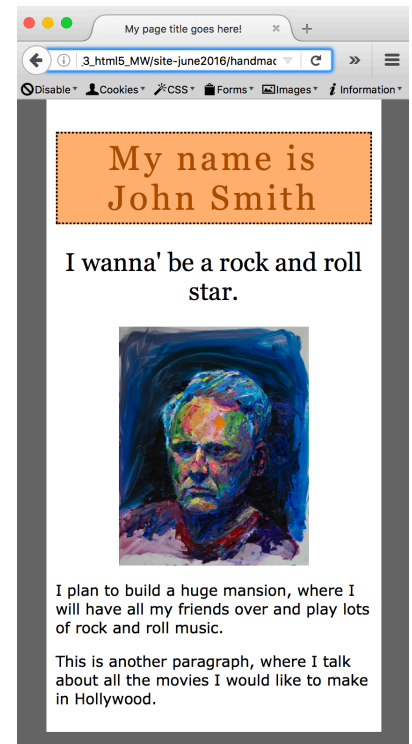
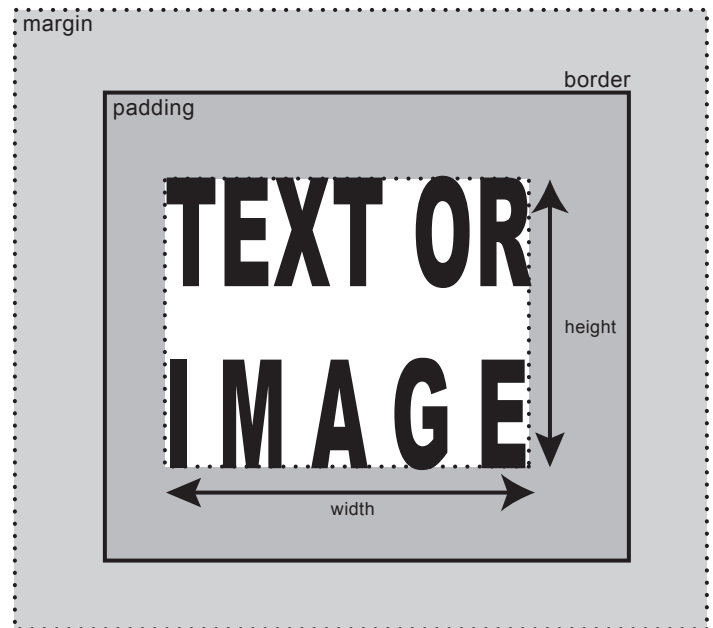


This (parent > child) relationship is an essential concept, remember it, and don't forget to call your mother 😊

**STEP ONE:** Close all the software on your desktop, including any notepad (textwrangler), or browser windows (Firefox, Safari, IE, etc).

**STEP TWO:** Navigate into your **James-Smith** folder in the campus network.

**STEP THREE:** Right click within your James-Smith folder and choose New > folder. Name the new folder: **lesson-2**.



## camelCase and Starting fresh

**STEP ONE:** using Windows (or Finder on a Mac) navigate into your first homework folder and copy the **handmade.html** file. The keyboard shortcut to copy a selected file is CTRL + C (on Mac it's: CMD + C)

**STEP TWO:** Navigate back into your **James-Smith > lesson-2** folder and choose **Paste**. Keyboard shortcut for paste: CTRL + V

**STEP THREE:** Change the name of your **handmade.html** file to: **index.html**

NOTE: \*do not\* use capital letters or spaces in file names in this class. In the world of programming, we live by the KISS principal. Keep It Simple Stupid.

**Capital letters or spaces in your web site file names will cause your web site to break!**

If you must have a long file name made from multiple words, you have a couple options. You can use camelCase:

myDogIsBrownAndBlack.jpg

camelCase refers to the hump in the filename. Each new word is capitalized, but there are \*no spaces\* in the file name. NOTE: I am breaking my own rule about not using capital letters here. Capital letters are fine, and expected when you are talking to other humans. But when you are programming code avoid capitals if you can.

When you put your website online, your webhost requires you to name the main file: **index.html**, NOT: **Index.html**.

Using a file name of **Index.html** will break your website. Also, if you keep your habit of capitalizing first letters in website filenames, it will complicate your HTML programming, because you will have to remember things like: "Did I or did I not capitalize that file name...I can't remember!"



It is also OK to use dashes in a long file name:

my-dog-is-brown-and-black.jpg

Both camelCase and dashes allow long file names, without using the illegal \*space\* character. But if you can, name your website files with lowercase short single words.

I have noticed many of you are writing code with a lot of errors. This is normal for new web designers. We could go back and fix your bad code...but that would take too long.

Instead, lets start from scratch.

**STEP FOUR:** Right click on your index.html file in your lesson-2 folder and choose > open with Notepad. If you don't get the right click option, open Notepad directly, and drag the file into Notepad.

**STEP FIVE:** delete all the code, every bit of it, so that you have a blank Notepad window. Remember, you have this code preserved in the handmade.html file back in the first lesson folder, so it's not gone forever.

## Creating an external style sheet file

**STEP ONE:** type the code you see to the right, and **save**.

Be careful! Spelling details matter.

Take note of the parent > child relationships:

The starting and stopping **<html>** **</html>** tags are the top level parent tags.

**<head>** and **<body>** are both children of the parent **<html>** tag.

The **<title>** is nested within, or "is a child of" the **<head>** tag

**STEP TWO:** Close your Notepad code window.

**STEP THREE:** Using Windows (or Finder on a Mac) drill down to the inside of this new lesson-2 folder.

```

index.html
<!doctype html>
<html lang="en">

<head>

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>
  John Smith - home
  </title>

</head>

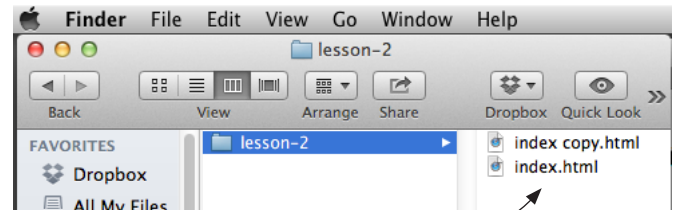
<body>

</body>

</html>

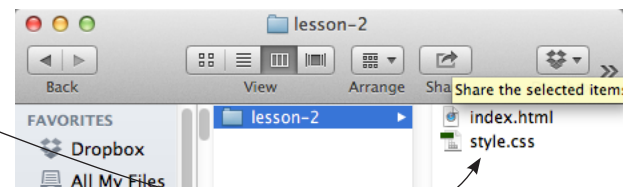
```

**STEP FOUR:** Touch the **index.html** file. Copy it, and then right click in a blank area of the folder and choose paste. It should come in with a file name of: **index copy.html**, or something similar.



**STEP FIVE:** change the name of the copied file to **style.css**

**STEP SIX:** Right click on the style.css file and choose: open with > Notepad.



**STEP SEVEN:** delete all the code in the style.css Notepad window so it is completely empty.

## Dimmer switch and style sheets

**STEP ONE:** type the code to the right inside the **style.css** file and click **save**

NOTE: do not copy and paste from previous homework. You need to practice your typing skills. As you type, think about the logic of what you are typing, picture what each line of code will do on the web page.

Because we are typing our style sheet rules in this new file named **style.css**, we have to find a way to get our main page, the **index.html** webpage to read this code on style.css.

Here is an analogy for what we are about to do.

Imagine the lights on your cars dashboard. When you turn that little knob on the dashboard, all of your cars instrument lights get brighter, or darker, instantly. That dashboard dimmer knob on your car is like the external style sheet on a website. If we change the color of a style in our external **style.css** file, all of our web pages in our web site should change their colors, instantly.

What we need is a "wire" to connect the **index.html** file to the **style.css** file. In fact, we need a wire to connect **style.css** to all the web page html files we will eventually have in our website.

**STEP TWO:** After you have saved all your changes to the **style.css** file, minimize it to your taskbar.

**STEP THREE:** Use Windows (or Finder) to open the parent folder of these files: Elvis-Presley/lesson-2/

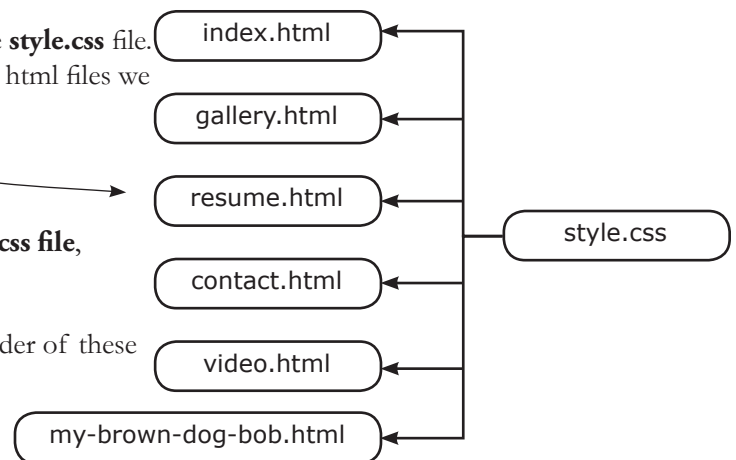
```

body {
    margin: 0;
    padding: 0;
    background-color: #666;
    font: 1em Verdana, Helvetica, sans-serif;
}

#wrapper {
    margin-right: auto;
    margin-left: auto;
    padding: 10px;
    width: 80%;
    max-width: 1400px;
    background-color: #fff;
}

.center {
    display: block;
    margin-right: auto;
    margin-left: auto;
}

```



## Linking to external style sheet file

On our first web page (handmade.html) you may recall that we put the style sheet rules up in the head of our web page, where they worked fine. But imagine a 20 page website, where each page had a style sheet rule that told the h1 tag to be blue. When you decide that you'd much prefer orange for your h1 headers, you would have to open 20 pages and make that single change before your entire website would have the new orange header.

A much better plan is to use an external style sheet, and have all the web pages link to it. You can make a change to the h1 style sheet rule in the external **style.css** file, and all of your website pages update instantly. They are all connected via the linked style sheet file, just like the dashboard lights on your car are all connected (wired) to the dimmer control switch.

**STEP ONE:** up in the head of your **index.html** code, which you should have open in Notepad, find the closing head tag **</head>** and click to it's left to get a blinking cursor. Press the enter key several times to get some blank white lines.

**STEP TWO:** type this line of code, it is the "wire" that will connect all of your html pages to a central style sheet file:

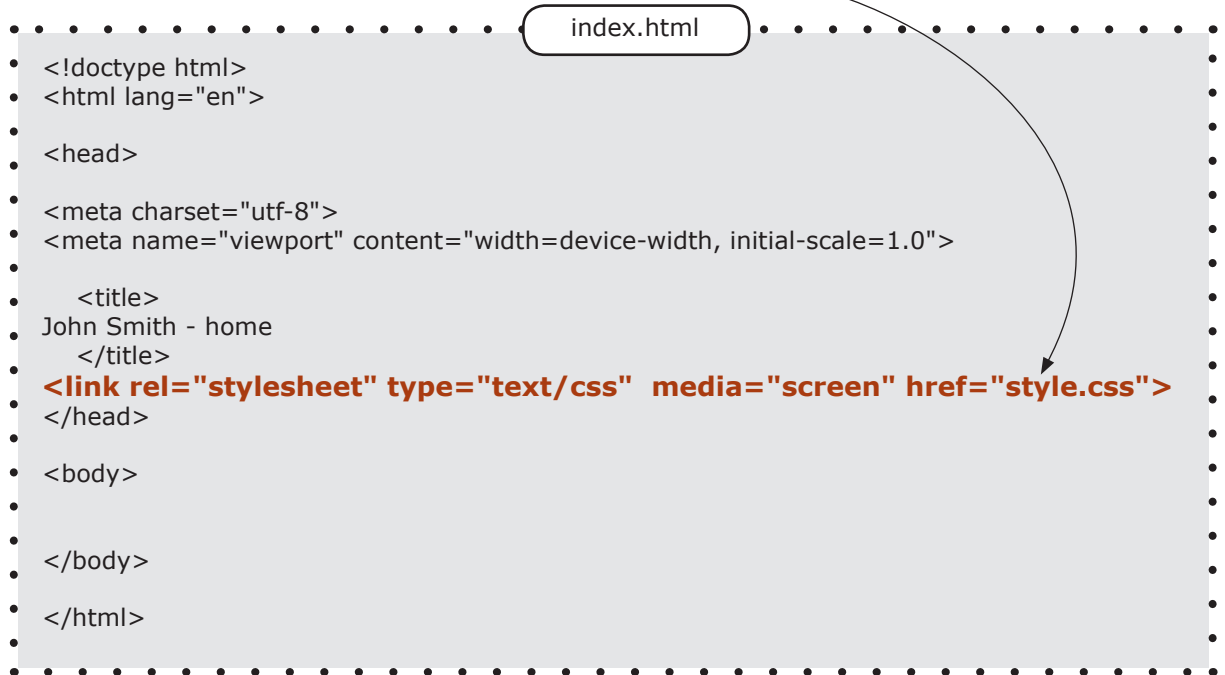
**<link rel="stylesheet" type="text/css" media="screen" href="style.css">**

The most important line of code here is:

**href="style.css"**

This tells the browser to go get the style sheet rules we wrote in our **style.css** file and "link" to them. ("link" in this context means: "import")

It is as if we wrote them directly in the head of our html file, like we did on **handmade.html**, but instead they are getting imported from the external file.





## Testing your linked style sheet

**STEP ONE:** Inside the starting and stopping body tags, add in a starting and stopping div tag, as we did before.

**STEP TWO:** Within the div tag, add a header tag. All the new code is shown here in **bold brown** text. Save the file.

**STEP THREE:** Use Windows (or Finder) to navigate into the parent folder: lesson-2

**STEP FOUR:** right click on the **index.html** file and choose: **open with > Firefox** (or IE, Safari, etc)

**STEP FIVE:** If you see the white box **<div id="wrapper">** displaying at 80% of the browser window **<body>**, you have successfully imported your external style sheet. I will walk around and assist during the lecture. This is an easy step to break.

```

index.html
<!doctype html>
<html lang="en">

<head>

<meta charset="utf-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">

<title>
John Smith - home
</title>
<link rel="stylesheet" type="text/css" media="screen"
href="style.css">
</head>

<body>

<div id="wrapper">

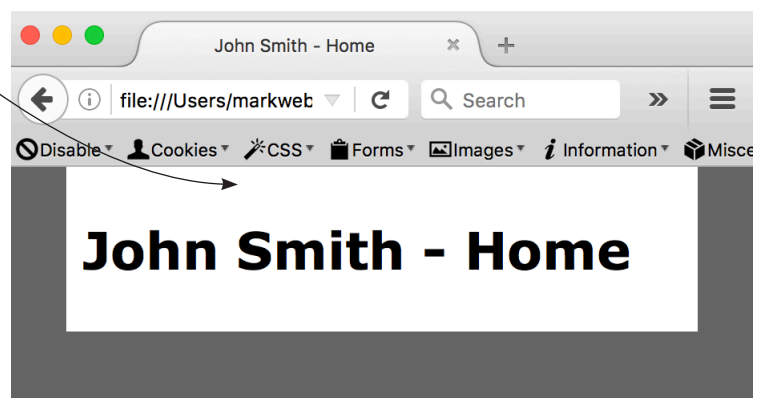
<h1>
John Smith - Home
</h1>

</div> <!-- end wrapper div -->

</body>

</html>

```



## Building an Interface structure

We need an interface structure in our web page. This structure needs to be built with a number of parameters in mind:

- It needs to be search engine friendly.
- It needs to be accessible to people with screen readers (blind people).
- It needs to work across all the devices (viewports) that might access it: computers, iPads, iPhones, Android phones, dumb phones, TV screens.
- It needs to work in all modern browsers, meaning browsers less than 4 years old.

We are going to build a series of nested html elements. Remember: an html element is like a box. A `<div id="wrapper">` tag is a classic example of the CSS box model.

Refer back to page 24 for an illustration of the CSS box model.

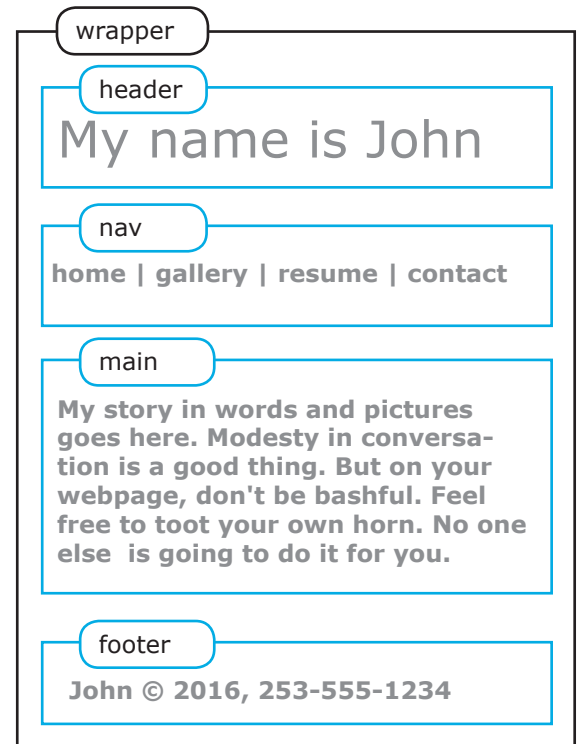
We can put boxes within boxes, and then boxes inside those boxes. We can tell the boxes to behave like columns. We can even tell certain boxes to show or hide, based on mouse movement. We can tell boxes to float and reposition themselves like a bowl filled with ping pong balls.

But first we will start with a simple set of nested boxes as shown to the right.

**STEP ONE:** Delete your h1 tag from inside the wrapper div

**STEP TWO:** Add the following code, inside the starting and stopping wrapper div tags. The new code you need to type is highlighted in bold brown.

NOTE: I have used the tab key to add horizontal white space in my code. These tabs can help you to see the parent > child relationship between these nested div boxes. Tabs can also help you find bugs in



```

    . . . . . index.html . . . . .
    <body>
    <div id="wrapper">
      <header class="masthead">
      </header>
      <nav class="main-menu">
      </nav>
      <main class="main-area">
      </main>
      <footer class="footer-area">
      </footer>
    </div> <!-- end wrapper div -->
  </body>
</html>
  
```

## Add content in your HTML elements

These new elements (header, nav, main & footer) may look strange if you are used to older versions of HTML. The new elements were created by the world wide web community, officially known as the:  
<https://www.w3.org/>

These new elements bring in some "semantic" meaning to the elements that is widely understood by all the modern browsers, and many hardware devices such as tablets, smartphones and screen reader applications for blind people.

For example, the old way was to type `<div id="header">`, but that tag has no logical meaning attached to it, at least in the mind of the browser.

Now with HTML5 we can use the new `<header>` element, which is understood to mean a place where you put the top identifying information for the web page.

`<header>` has semantic meaning, whereas

`<div id="header">` does not.

For further reading go to:

<http://blog.teamtreehouse.com/use-html5-sectioning-elements>

**STEP ONE:** Add some h1 and paragraph tags inside each of your elements as shown, new code in bold brown.

**NOTE:** in this element `<header class="masthead">` I am saying I want the browser to style the **header** element using the class of **masthead**.

index.html

```
<body>
<div id="wrapper">
  <header class="masthead">

    <h1> John Smith - Home </h1>

  </header>

  <nav class="main-menu">

    <p>home | gallery | resume | contact</p>

  </nav>

  <main class="main-area">

    <p>lots of words go here...</p>

  </main>

  <footer class="footer-area">

    <p>Copyright John Smith</p>

  </footer>

</div> <!-- end wrapper div -->
</body>
</html>
```

## Add the new styles

We can speak to the header element with a class="masthead" by writing a style sheet rule like this:

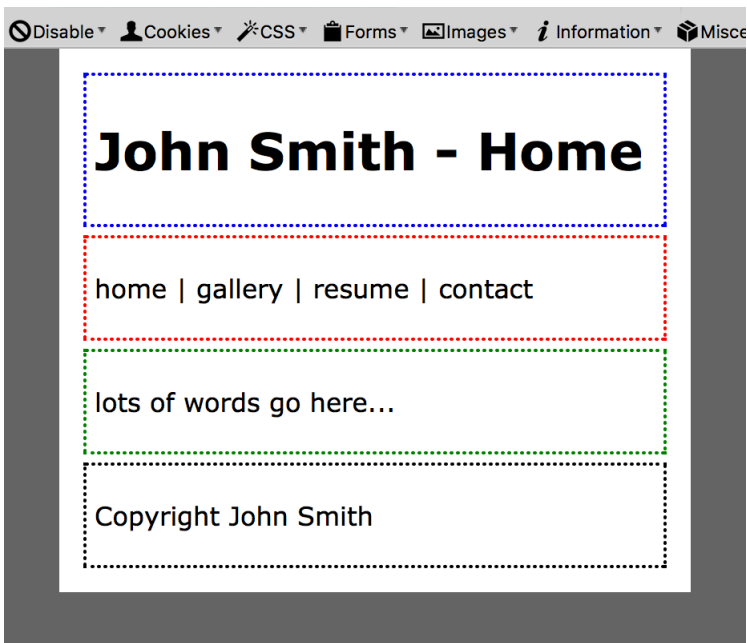
**header.masthead {declarations here}**

The dot (.) between header and masthead means: if there is a header element that has a class of masthead, apply the following rules.

**STEP ONE:** open your **style.css** file in Notepad

**STEP TWO:** add these new style sheet rules below the last pre-existing style sheet rule

And here is how it should look after you **save** all your changes to both **styles.css** and **index.html**



```

style.css

body {
    margin: 0;
    padding: 0;
    background-color: #666;
    font: 1em Verdana, Helvetica, sans-serif;
}

#wrapper {
    margin-right: auto;
    margin-left: auto;
    padding: 10px;
    width: 80%;
    max-width: 1400px;
    background-color: #fff;
}

.center {
    display: block;
    margin-right: auto;
    margin-left: auto;
}

header.masthead {
    border: 2px dotted blue;
    margin: 5px;
    padding: 5px;
}

nav.main-menu {
    border: 2px dotted red;
    margin: 5px;
    padding: 5px;
}

main.main-area {
    border: 2px dotted green;
    margin: 5px;
    padding: 5px;
}

footer.footer-area {
    border: 2px dotted black;
    margin: 5px;
    padding: 5px;
}

```

## Unordered lists for navigation

We need some good looking navigation on our website. We could use the simple words we typed on the last couple pages:

home | gallery | resume | contact

They could be made into clickable links, but they aren't pretty. By the way, that weird vertical line between the words is the pipe key. To get it, hold down shift, and press the backslash key (\). A better and more versatile way to add links to your webpage is to start with list items, and modify the list items:

**STEP ONE:** In your index.html page, delete the entire paragraph inside the `<nav>` element. This will leave you with an empty tag, like this:

```
<nav class="main-menu">

</nav>
```

**STEP TWO:** Inside your nav box `<nav>` add a starting and stopping **unordered list** tag `<ul>`. Inside that, start and stop a **list item** `<li>`, and inside the list item, type **home**. Maintain the indentation I am showing here by pressing the tab key each time you put a new "child" inside a "parent". (`<li>` is a child of `<ul>`)

**STEP THREE:** Around the word "home", place starting and stopping `<a>` tags. In the starting `<a>` tag, add the property `href="index.html"`. Code is shown to right.

index.html

```
<body>
<div id="wrapper">
  <header class="masthead">

    <h1> John Smith - Home </h1>

  </header>

  <nav class="main-menu">

    <ul>

      <li>home</li>

    </ul>

  </nav>
```

index.html

```
<nav class="main-menu">

  <ul>

    <li><a href="index.html">home</a></li>

  </ul>

</nav>
```

The `<a>` tag stands for anchor. When you surround a word with starting and stopping `<a>`/`</a>` tags, you are telling the browser that the word is an "anchor" that will lead your webpage visitor to another page...should they click the word. Just the `<a>` tag alone won't work, it must have the `href="?"` property. Inside those double quotes you tell the browser where it should navigate to when they click the word. href stands for hypertext reference. By writing `<a href="index.html">home</a>`, we are telling the browser this: If someone clicks on the word home, the browser should navigate to, and open the file called: **index.html**



## Duplicating the list items <li>

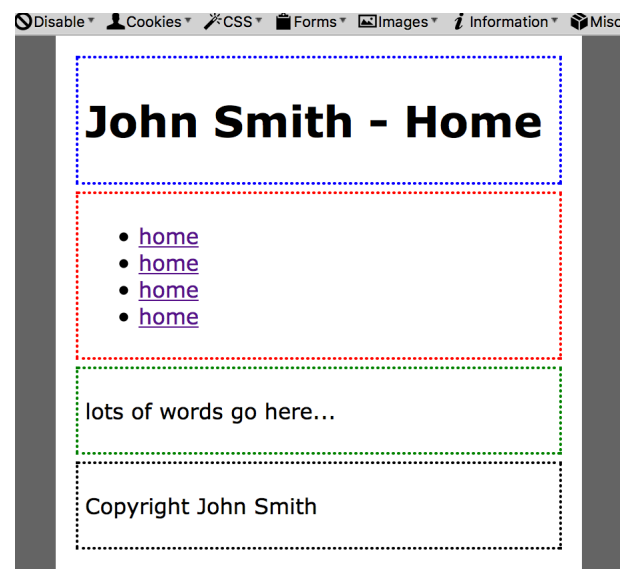
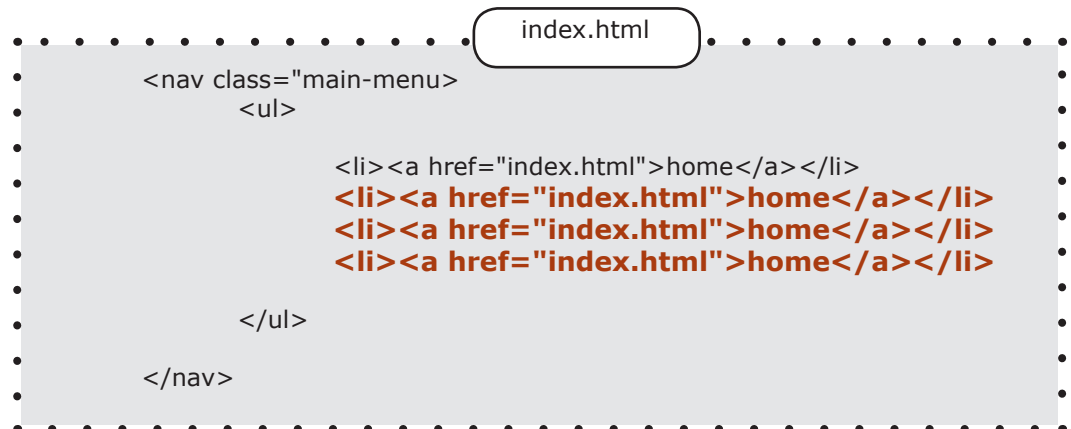
This is how your unordered list should look now.



**STEP ONE:** Copy the entire line that contains the list item, including the tab spacing at the left of the <li>, and paste it in on a new line immediately below.

Repeat two more times until your code looks like this:

**STEP TWO:** Save your page and make sure it looks like this screenshot before you go any further. If it doesn't, check your code for spelling errors.



## Preparing your anchor links for navigation

**STEP ONE:** In the second list item, replace the word index with gallery, and replace the word home with gallery.

**STEP TWO:** In the third list item, type resume, and resume. In the fourth line, type contact and contact. See the code to right.

Here is how it should look in the browser after you save and refresh.

NOTE: if you click any of the links except home, you will get a broken page error because those webpages (gallery, resume, contact.html) don't yet exist. Home does exist, it is called index.html, and it is the file we are working on. So you can click the link to home, the browser navigates to it, but you are already there, so it is like hitting refresh.

This navigation style will work fine, and it is very functional as far as search engines and blind screen readers...but let's make it prettier....that's why they call it Web Design!

```

<nav class="main-menu">
  <ul>
    <li><a href="index.html">home</a></li>
    <li><a href="gallery.html">gallery</a></li>
    <li><a href="resume.html">resume</a></li>
    <li><a href="contact.html">contact</a></li>
  </ul>

```

### John Smith - Home

- [home](#)
- [gallery](#)
- [resume](#)
- [contact](#)

lots of words go here...

Copyright John Smith

**STEP THREE:** Open your style.css file in Notepad. Add the code shown in bold brown.

**EXPLANATION:** Notice the odd selector syntax. This is called specificity. In plain English, this selector is saying, if there is a **nav** element in the body section of the **index.html** page that has a class of **main-menu** AND that element (**<nav class="main-menu">**) has a child **<ul>** tag AND that **<ul>** tag has a child **<li>** tag, then the following declarations shall apply. The space between **.main-menu** and **ul** has the effect of saying: "with a child of". By using spaces between selectors, we are being very specific. The only **<li>** tags that are affected by this rule are the ones that satisfy all the specifics in the long selector.

**display: inline-block;** tells the words inside the list item to display side by side (inline), as opposed to one on top of the other. The block part of the property allows the word to be clickable all the way out to the border of the **<li>** box model, including the padding and background color.

We will add padding, border and background color later.

```

header.masthead {
  border: 2px dotted blue;
  margin: 5px;
  padding: 5px;
}

nav.main-menu {
  border: 2px dotted red;
  margin: 5px;
  padding: 5px;
}

nav.main-menu ul { list-style: none; }
nav.main-menu ul li {
  display: inline-block;
}

main.main-area {
  border: 2px dotted green;
  margin: 5px;
  padding: 5px;
}

footer.footer-area{

```

## Navigation links as Tabs

This is how your navigation links should look after completing the code on the previous page. The unordered links are displaying side by side (inline).

NOTE: Do not test your page in IE yet. Use Firefox or Chrome.

**STEP ONE:** add to, and or modify your style sheet code until it looks like the code you see to the right.  
Be careful! Spelling details matter.

After you type this new code,  
this is how your browser should look on hover.



**Limited Explanations:** I will explain more in class...

I zeroed padding on the nav, and zeroed out padding and margin on the unordered list tag.

**text-decoration: none;** means to remove underlines from links `<a>`

The padding values on `ul li a`, controls how much red there is around the text, which in turn controls the size of the buttons. I used a small **font-size: 0.75em;** to keep the buttons from wrapping on a smartphone screen.

`nav.main-menu ul li a:hover` is saying that if there is a `nav` with a class of `main-menu`, which has a child of `ul`, which has a child: `li`, which has a child `a` AND the `<a>` tag is being hovered over by the mouse (`a:hover`), the following rules inside the curly braces shall apply. In other words, that rule enables rollover effects.

**STEP TWO:** zzz



## The complete style.css file - so far

On previous pages I've been showing just the changes to the style.css code. On this page, I have all the code we have written so far in the style.css file. Take a moment to check it against your code.

### style.css

```
body {
    margin: 0;
    padding: 0;
    background-color: #666;
    font: 1em Verdana, Helvetica, sans-serif;
}

#wrapper {
    margin-right: auto;
    margin-left: auto;
    padding: 10px;
    width: 80%;
    max-width: 1400px;
    background-color: #fff;
}

.center {
    display: block;
    margin-right: auto;
    margin-left: auto;
}

header.masthead {
    border: 2px dotted blue;
    margin: 5px;
    padding: 5px;
}

nav.main-menu {
    margin: 0;
    padding: 0;
    text-align: center;
    border-bottom: 2px solid #db3737;
}

nav.main-menu ul {
    list-style: none;
    margin: 0;
    padding: 0;
}
```

### style.css (continued)

```
nav.main-menu ul li {
    display: inline-block;
}

nav.main-menu ul li a {
    display: block;
    text-decoration: none;
    padding: 0.3em;
    background-color: #c2c2c2;
    font-size: 0.75em;
    color: #2c2e5c;
}

nav.main-menu ul li a:hover {
    color: #fff;
    background-color: #db3737;
}

main.main-area {
    border: 2px dotted green;
    margin: 5px;
    padding: 5px;
}

footer.footer-area {
    border: 2px dotted black;
    margin: 5px;
    padding: 5px;
}
```

## The complete index.html code - so far

Here is all the code in our index.html file, so far.

**STEP ONE:** add this line of code to make the IE browser happy when viewing the page over the local network.

Pictured below is how it looks in the browser when you hover your mouse over the link to **gallery**.

The browser feels where your mouse is, and when it senses your mouse is hovering over an anchor tag `<a></a>` that has a style sheet rule for **a:hover**, it applies those styles (the red) to the anchor tag.. as long as your mouse is still in the hover state.

Later we will write a style sheet rule that tells the home page link to stay permanently in the "red" state, so our users get confirmation that they are on the home (index.html) page. This is called the "you are here flag".

index.html

```
<!doctype html>
<html lang="en">

<head>

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="IE=edge" >
<title>
John Smith - home
</title>
<link rel="stylesheet" type="text/css" media="screen" href="style.css">
</head>

<body>

<div id="wrapper">

    <header class="masthead">
        <h1>John Smith - Home</h1>
    </header>

    <nav class="main-menu">
        <ul>
            <li><a href="index.html">home</a></li>
            <li><a href="gallery.html">gallery</a></li>
            <li><a href="resume.html">resume</a></li>
            <li><a href="contact.html">contact</a></li>
        </ul>
    </nav>

    <main class="main-area">
        <p>lots of words go here...</p>
    </main>

    <footer class="footer-area">
        <p>Copyright 2016 John Smith</p>
    </footer>

</div> <!-- end wrapper div -->

</body>

</html>
```

Disable Cookies CSS Forms Images Information

# John Smith - Home

home gallery resume contact

lots of words go here...

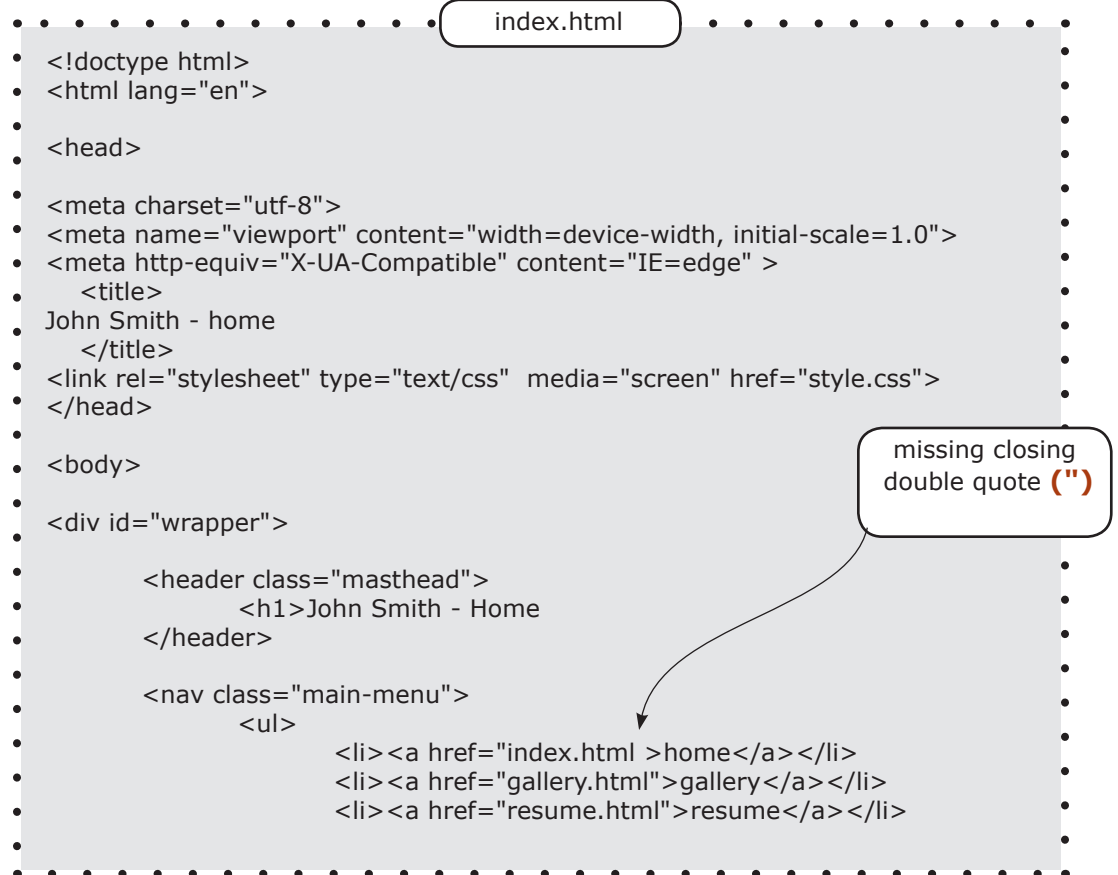
Copyright John Smith



## Find errors with free Validation services

You will have noticed by now that it is hard to find coding errors. You will see the broken web page in the browser, but there is no easy way to find the code that is breaking it. There are websites you can go to that will examine your code and show you your errors. One of the reasons you have a doctype in your index.html is to help with the free online code validation services. Our `<!doctype html>` tells the validator that we are writing HTML 5, and it should examine our code for errors based on the rules of HTML 5, not some other language, like HTML 4.

Let's test our code in a validator. Before we test it, let's intentionally break something, so the validator has something to find.



**STEP ONE:** On your index.html page, find the starting and stopping `<a>home</a>` tags

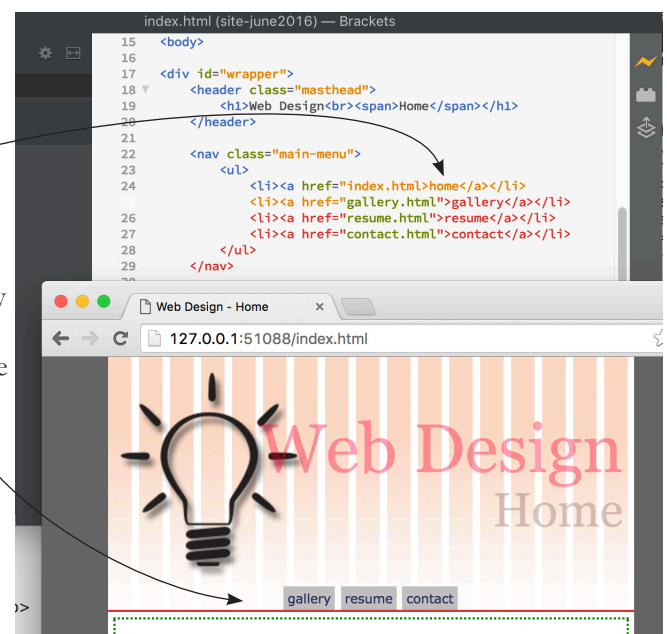
**STEP TWO:** Even though it is wrong, delete the closing double quote character from `<a href="index.html">` as shown to the right.

**STEP THREE:** Save the file, and open it in Firefox or Chrome.

**NOTE:** By deleting just the one closing double quote, my entire home button has vanished. Note also that by using a fancier code editor (Brackets), I get colored code. When you learn to read code coloring, you will recognize that this code is shouting at us, and pointing toward the error. Brackets knows that each equals sign should be followed by two double quotes. It highlights the property after the equals sign in yellow. In this case the yellow coloring extends much further than it should, because of the missing double quote.

**STEP FOUR:** Select and **copy** all the code in your index.html file.

**STEP FIVE:** Go to this webpage: <https://validator.w3.org/> Click the tab for **validate by direct input**. Paste in your code, and press the **check** button.



## W3C Validator - Errors found!

This is how the Validator looks after it has examined your code. It tells you how many errors it found. Farther down the page, it tells you exactly what line number they are on. It basically points right at the problem, just as the code syntax coloring did in the Brackets code editor.

**STEP ONE: Close Notepad.** And you may also want to close your fancy code editor, if you are using one. Never use two editors at once on the same file, they will fight over ownership. Start Dreamweaver, we will take a brief tour of it's code editing capabilities.

**STEP TWO:** Make Dreamweaver a small window. Position your lesson-2 homework folder next to Dreamweaver in another small window.

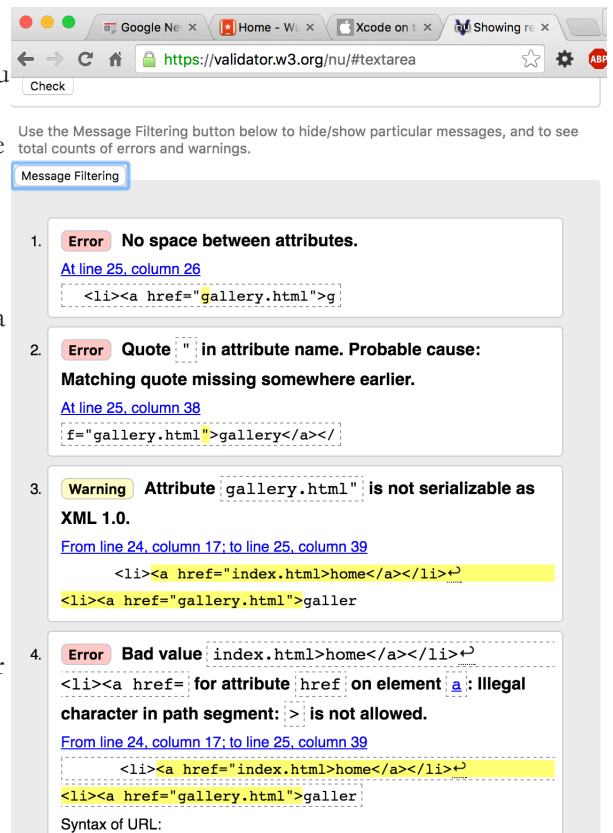
**STEP THREE:** Drag your index.html file directly into Dreamweaver, which should open it, or right click: open with > Dreamweaver

**STEP FOUR:** In Dreamweaver, click the top left button that says Code. Scroll down in Dreamweaver to whichever code line number the validator said you had a missing closing double quote.

If you don't see the line numbers in Dreamweaver, turn them on by choosing:

View > Code View Options > Line Numbers

**STEP FIVE:** Put the ( " ) back in, save, and re-validate. You should have a clean Validation.

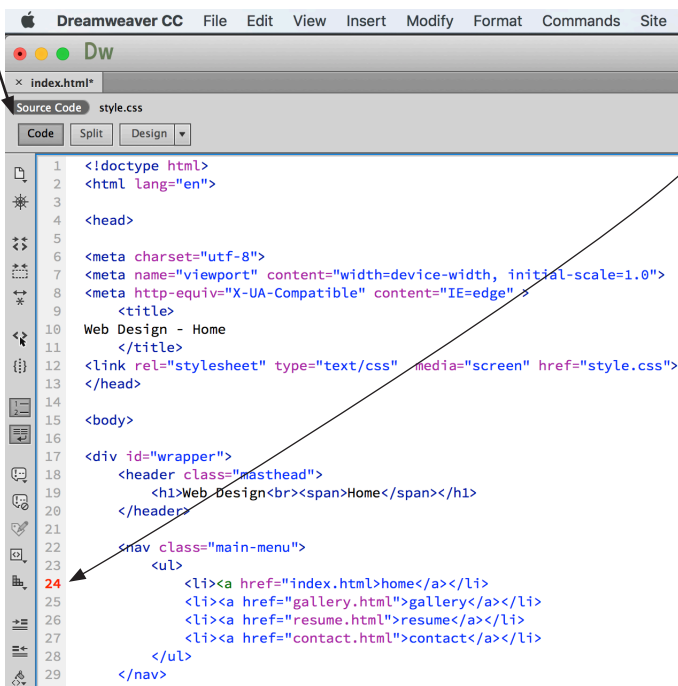


**NOTE:** Dreamweaver also colors your code, and it even puts a red line number where it sees an error, which can be very helpful.

But for beginners, Dreamweaver can be very confusing. We will get deeper into Dreamweaver later. We broke a number of rules using Dreamweaver this way.

While it may be tempting to work in Dreamweaver instead of Notepad, I recommend closing Dreamweaver, and returning to Notepad. You will learn more in Notepad, because you have to think harder.

If you really, really want to have code syntax coloring, try either Backets or Sublime. They are easier to figure out than Dreameaver.



# Validate your styles.css

**STEP ONE:** after you get a clean validation on your HTML, use the same website to validate your style.css code.

**NOTE:** this one is less friendly. For example, if you delete the closing curly brace after your nav {} style sheet rule, the validator will throw an error about one of the adjacent stylesheet rules. It will completely hide the nav { style sheet rule in the results area of errors. But this in itself can be helpful, as it gives you clues.

**STEP TWO:** Spend some time making sure that both index.html and styles.css validate, **it is important.**

## Designing our header

We have a decent looking page, but we need to jazz it up. Images are the only way to make this thing pretty. Because we don't want to make a huge header image that won't fit on a smart phone screen, we are going to make a logo, and set that on top of a tiling background pattern. These graphics will be made in Illustrator and Photoshop. I know that many of you are just starting out in Photoshop, and have never seen Illustrator, while others in this class have had several classes in both. I will give everyone the images, since making them from scratch is beyond the scope of this class. However, I will provide the original Illustrator and Photoshop files to you, in case you want to tinker with my design ideas.

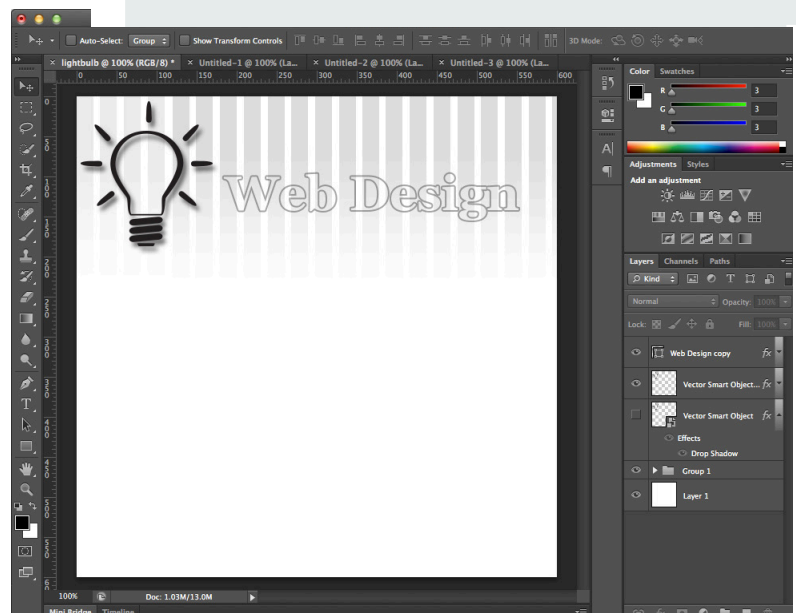
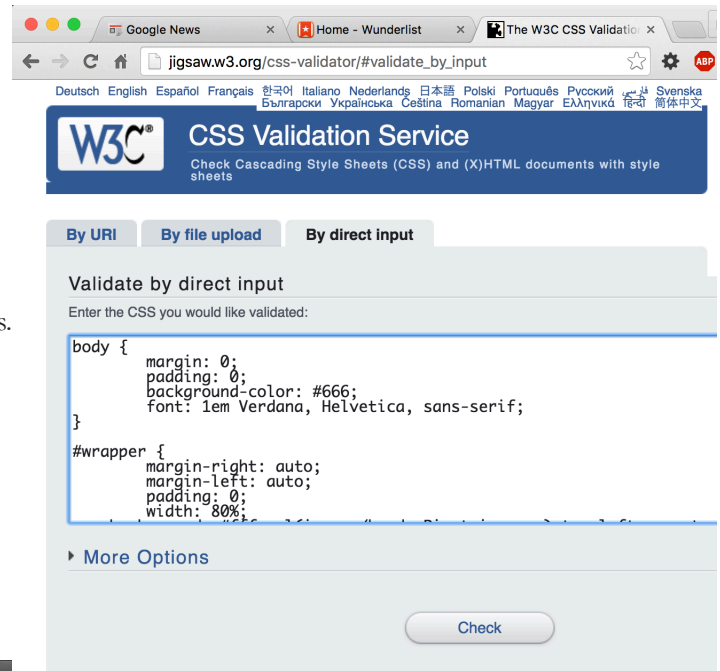
**If you know Photoshop and Illustrator** read on, otherwise skip to page:43

Look for the two lightbulb files: .psd and .ai, in the resources folder.

My basic principle was to design in Illustrator and Photoshop, using the best tools of each to come up with some images that I could then place in my webpage.

I drew the lightbulb logo using the pen tools. To get ideas for the logo, I went to istockphoto and looked at lightbulb illustrations. I tried to make one that was a composite of everything I saw there, but not an outright copy (rip off) of any one design. I dragged the logo from Illustrator to Photoshop.

For the vertical bars, I made a new layer in Pshop made a selection and poured a gray to transparent gradient down through the selection. I had to lower the layer opacity overall, also. I made duplicate layers, and dragged them out to the right, lining them up with space in between.



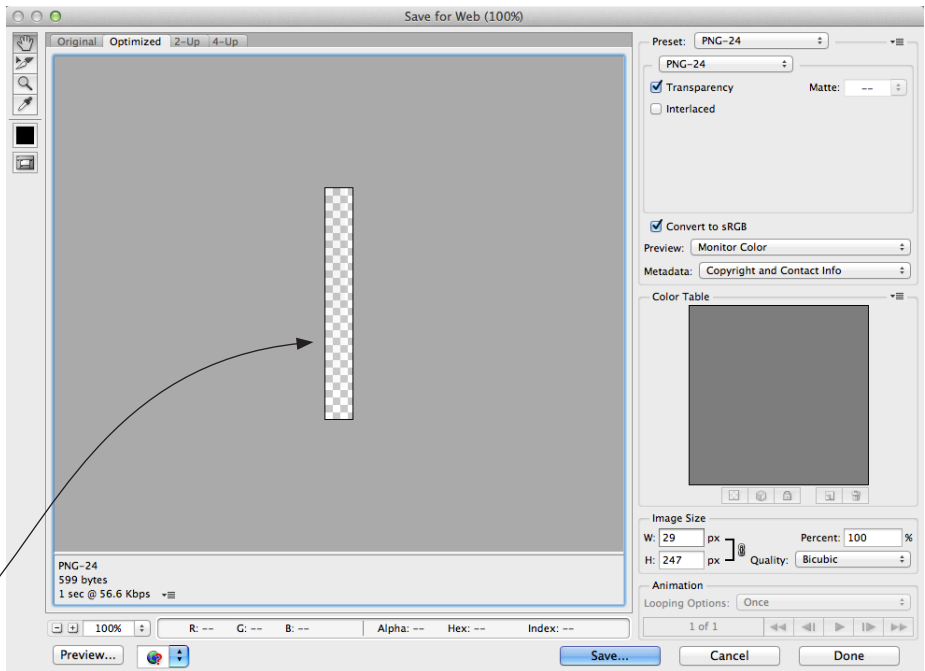
## Artwork tips, if you know Photoshop

NOTE: these are not required as I know many of you don't know Photoshop and Illustrator yet. Skip to page 43 if that's you.

After I got the mock up looking good in Photoshop, I turned off the white page background layer so I had transparency indicators. I made a selection (29x247px) around one of the vertical bars, and the white space next to it, and pressed Ctrl + shift + C to copy all the pixels on all the layers.

I made a new Photoshop document with a transparent background, pressed Ctrl + V to paste, and chose **save for web**.

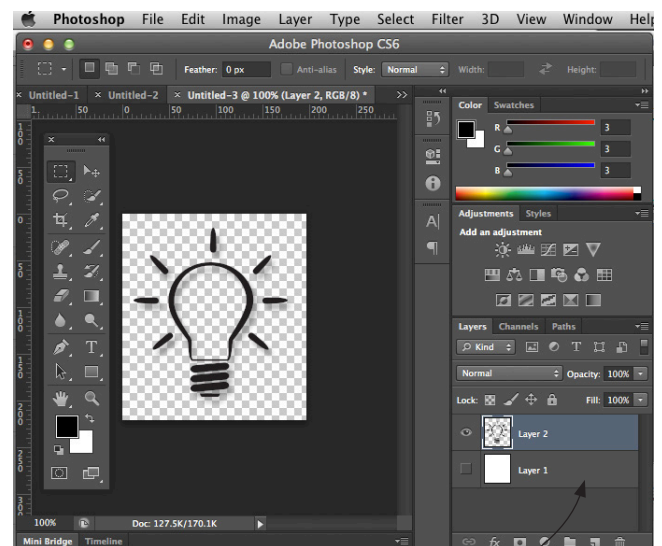
I chose **PNG 24**, with **transparency checked**.



NOTE: the gray bar doesn't even show up because it is so transparent. The only way to see if it is there is to temporarily put a white background layer under it, before you save for web. Just be sure to turn off the white background layer.

I did the exact same thing to the logo layer. It has a drop shadow that I wanted to be transparent (in the html), so it could show the vertical columns image underneath. Here I have copied it out of the main lightbulb interface mockup psd. Note that you can see the drop shadow over the transparency indicators. I exported from Photoshop using file save for web, as above.

NOTE: I did add in a temporary white background layer underneath the lightbulb layer, so I could check to see if my dropshadow had come across correctly in the copy. But I have it turned off preparatory to exporting as a transparent png.



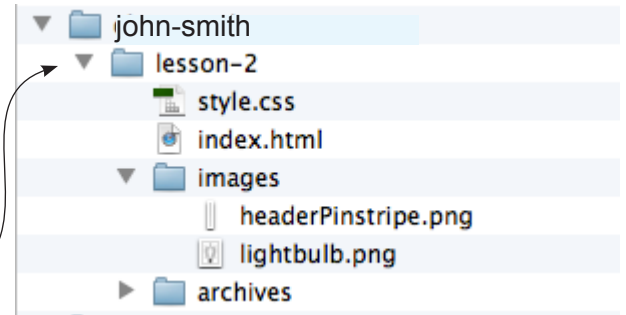
## Image in background of #wrapper

**STEP ONE:** Copy the images folder from the resources folder.

NOTE: a well organized website has all the images inside an images folder. If the site gets big enough, you can put all the html pages inside an html folder, and leave just the index.html file out at the root.

**index.html** is required to be at the root by the server (your web host provider: hostgator, godaddy, etc.)

**STEP TWO:** Here is how your folder structure needs to look. lesson-2 is the "root" folder. Inside that we have the index.html and style.css files. Also at the "root" level, we have an images folder, inside of which are the two images we will use in the upcoming code. They were both exported from Photoshop as transparent PNG's



**STEP THREE:** Open your style.css file in Notepad

**STEP FOUR:** find the style sheet rule for #wrapper

**STEP FIVE:**

make changes to the padding and background properties as shown.

EXPLANATION:

- I set the padding to zero so the red border under the nav element could get all the way across it's parent (the wrapper div)
- In the long background declaration, I start by telling the background to have a #fff (white) background, if there is space left over after I tell the image to appear.
- Next I tell the image to appear from it's parent folder (images).And I put it inside an url property. URL stands for Uniform Resource Locator.
- There is a required space after the closing parenthesis, then the words: top left. That means position the image in the top left corner of the parent div (#wrapper). I tell the image (headerPinstripe.png) to repeat in the x direction. That means, tile the image sideways for as long as there is room in the div.

style.css

```
body {
    margin: 0;
    padding: 0;
    background-color: #666;
    font: 1em Verdana, Helvetica, sans-serif;
}

#wrapper {
    margin-right: auto;
    margin-left: auto;
    padding: 0;
    width: 80%;
    background: #fff url(images/headerPinstripe.png) top left repeat-x;
    max-width: 1400px;
}
```



## Image in background of header element

This is how the headerPinstripe.png looks as it tiles (repeat-x) horizontally across the background of the wrapper div. Where it stops, the background white color takes over down to the bottom of the wrapper.

NOTE: my .png image may a different color than yours.



**STEP ONE:** Next up we have to zero out the margins and padding on the header element, give it a height that is tall enough to make room for our logo the lightbulb.png, and tell the png to appear with an url() call. Code shown to right in bold brown.



You may or may not see a slight alignment issue along the bottom of your buttons where they meet the red border-bottom line. The code hack above for **nav.main-menu** should fix it. In case I've not mentioned it earlier, 1 em is the size of a capital letter "M" in the default browser font family and size. So 1em = 13pixels.



## Styling the h1 header

That looks fine, but we need to get control of that h1 header. We need it there for the search engines, but there is no reason we can't make it prettier and prevent it from interfering with our logo.

**STEP ONE:** type the new header h1 style sheet rule listed to the right.

### EXPLANATION:

I made the selector explicit: **header.masthead h1** so that any other h1 tags on the page would not be affected because they wouldn't have the parent element of **header.masthead**.

**position: relative;** is used to give us the ability to position the h1 "relative" to where it would normally have fallen in the flow of the documents structure.

**top: 60px;** tells it to move down 60 px relative to where it was naturally.

**right: 10px;** means move the h1 all the way over to the right inside it's parent, which is header...but then move it leftward by 10px.

I made the **font: 4em**, which is really big. Go big, or go home!

**color: #4554af;** is the font color older browsers will use, we call this a fall back declaration. An older browser (5 years old?) will understand and use that color, because the next declaration will not make sense. Newer browsers will get that line, but they will continue on to the next line, and understand it as well. The last line written, and understood, wins the argument.

**color: hsla(0, 0%, 38%, 0.3);** is new CSS 3 syntax. This uses a new color model called HSLA, which stands for Hue, Saturation, Lightness, Alpha. The coolest part of all that is the 0.3 value, which stands for Alpha. 0.3 is the same as 30% opacity. In other words we now have an h1 tag which is 30% transparent, so the graphics show through underneath. As far as how to choose values for the other things like, Hue, Saturation, Lightness, you can try the free Brackets code editor. It has an HSLA color picker built in. You can also find one online by searching for **HSLA color picker**.

style.css

```
header.masthead {
  margin: 0;
  padding: 0;
  height: 215px; /* this makes room for height of lightbulb.png */
  background: url(images/lightbulb.png) top left no-repeat;
}

header.masthead h1 {
  position: relative;
  top: 60px;
  right: 10px;
  padding: 0;
  margin: 0;
  font: 4em Georgia, Times, serif;
  text-align: right;
  color: #4554af; /**this is fall back color for old browsers**/
  color: hsla(0, 0%, 38%, 0.3);
  /**hsla = Hue, Saturation, Lightness, Alpha, 0.3 = 30% opacity**/
}
```

## Finished header

**STEP ONE:** Open your index.html file in Notepad. Make this small change to the `<h1>` tag. Add a `<br>` tag between the second and third word...if you have 3 words as I do here. `<br>` forces a line break, also known as a "soft return". It's like pressing Shift+Enter in Microsoft Word. This should give you the result shown to the right. Our header now

looks pretty nice. It's not terribly original, but it looks a lot better than it did, and once you learn Photoshop better I'm sure you can come up with something more creative.

NOTE: I did tell my style sheet rule for `nav.main-menu ul li a` to have a `font-size: 1em`; which is bigger than it was before.

**STEP TWO:** In your `<h1></h1>` tag, surround the third word with starting and stopping `<span></span>` tags. `<span>` is a tag that does nothing by itself. It is called an "inline" tag because it is meant to style words that are inline in a sentence. By hanging a `<span>` tag around the third word, we can write a style sheet rule that tells `<span>` words to be a bit smaller, and colored differently.

**STEP THREE:** Open style.css and add the code shown to the right in bold brown. I changed the saturation value to 20%. `4em/0.8em` means: font-size/leading. I'm not sure why the span font-size had to be 0.7em to work. Might be some inheritance going on there between the parent/child relationship of

`<h1><span>`

```

index.html
<body>
  <div id="wrapper">
    <header class="masthead">
      <h1> Web Design&ltbr>Home </h1>
    </header>
  </div>
</body>

```



```

index.html
<body>
  <div id="wrapper">
    <header class="masthead">
      <h1> Web Design&ltbr><span>Home</span> </h1>
    </header>
  </div>
</body>

```

```

style.css
header.masthead h1 {
  position: relative;
  top: 60px;
  right: 10px;
  padding: 0;
  margin: 0;
  font: 4em/0.8em Georgia, Times, serif;
  text-align: right;
  color: #4554af; /**this is fall back color for old browsers, listed first**/
  color: hsla(0, 0%, 38%, 0.3);
  /**hsla = Hue, Saturation, Lightness, Alpha, 0.3 = 30% opacity (alpha)***/
}

header.masthead h1 span {
  font-size: 0.7em;
  color: hsla(0, 20%, 38%, 0.3);
}

```

## Tweaking the nav position

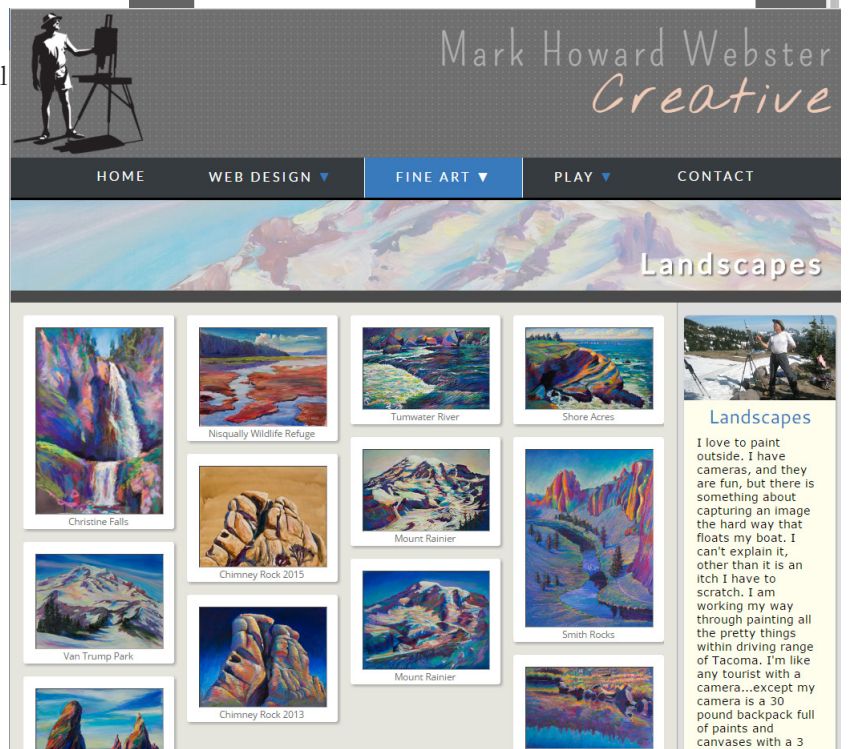
Here it is in my iPhone emulator showing the different color, and smaller font size of the `<span>Home</span>`. We have some issues on the tiny browsers of smartphones. This is something we will fix later with some new CSS 3 tricks called Media Queries, but let's focus on more important things for now.

## Thumbnail Gallery

This web page needs to be able to show some of our pictures. A great use of your gallery web page is to display digital artwork you have done in Photoshop, Illustrator, InDesign, or captured with your camera, whether film or digital. Because I knew you may not have anything with you, I've placed some large files in the web principles folder for you to practice on. Their file names look like this IMG\_0996.JPG

These files are 7 megabytes in size. They need to be shrunk down to around 200k for the full size image, and under 20k for the thumbnail.

NOTE: we need two different images for a gallery webpage to work. The thumbnail image, and the full size version. You can see the thumbnail principal in action on facebook. Look at any picture gallery and you will see that the images are not shown full size, unless you click on them, and facebook goes into "slide-show" mode where it shows them full size. But full size on the internet simply means that the image fits nicely in the browser window. Full size in Digital Photography means the 4000 pixel wide image that came out of the camera. We will cover all this and more in the next section.



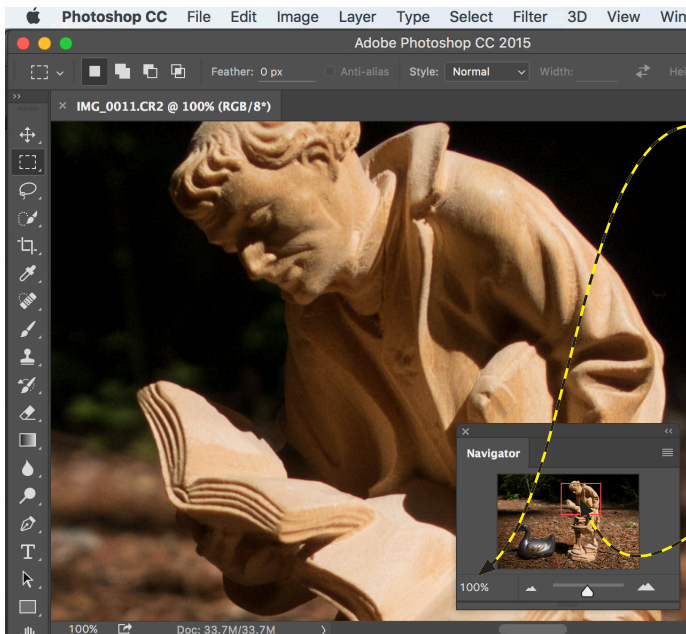
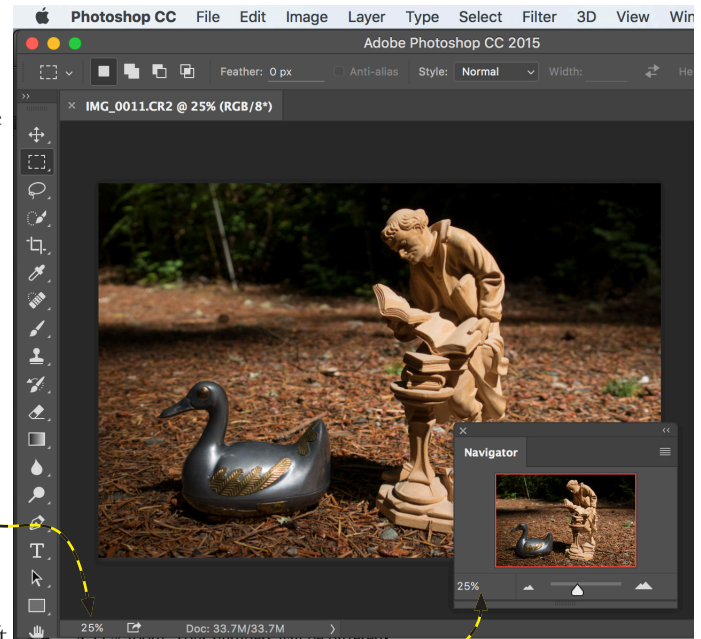
## Preparing images in Photoshop

**STEP ONE:** copy 4 of the six megabyte images from the resources folder. Use your own images if you have access to the full size originals. Modern cameras and smart phones shoot images with file sizes of 5 to 25 megabytes. Navigate to your john-smith folder and paste the full size images in at the root of your name.

**STEP TWO:** Make Photoshop a small window on your desktop. Bring up your Windows documents folder in a small window next to Photoshop. Drag one of the provided images directly into the top head area of Photoshop, which should open the image.

**STEP THREE:** When Photoshop opens an image that is too big to fit, it zooms out on the photo until it can fit it all on screen. In this example shown here, Photoshop has zoomed out to 25% zoom. Your numbers will be different.

**STEP FOUR:** In Photoshop, click Window > Navigator. When the Navigator palette opens, click on the word Navigator at the top left corner and tear it away from wherever it is docked. Drop it in the middle of Photoshop. Then gradually drag it back towards the right until you see some blue lines flashing. This is the docking indicator getting excited. Don't let it dock, keep Navigator palette undocked for now.



**STEP FIVE:** Grab the zoom slider on Navigator and slide it up to 100%. NOTE: you can also click the big mountain button to right of the zoom slider.

**STEP SIX:** This is the actual size of the image at 100% zoom. At 100% zoom, called "Actual Size" in Photoshop, you are seeing 72 pixels in every inch of glass on your computer monitor. Note, this can vary by monitor and computer. For example, a 2016 Macbook with Retina display can display 226 pixels per inch (PPI). Notice that you have to scroll to see the entire image, it doesn't fit at 100%. You can also grab the red box in Navigator to scroll around the image. That red Navigator box represents the part of the image that can currently fit in Photoshop's window. If you were to put this image on your webpage, two things would happen. It would load very, very slowly (because it is huge) and

the browser wouldn't be able to fit it on the webpage. You could write code to force the image to fit, but it is much better to manually toss out all the extra pixels so it loads faster.



## Shrinking your image

You may not have noticed, but when you upload images to facebook, straight from your digital camera, facebook automatically shrinks them. Did you ever wonder why they are so much smaller on facebook? The web developers at facebook wrote php scripts that automatically do what we are about to manually do in Photoshop. They know that your facebook page will load much faster if it only has to load little thumbnail images, instead of full size webpage images

**STEP ONE:** In Photoshop, click Image > Image Size

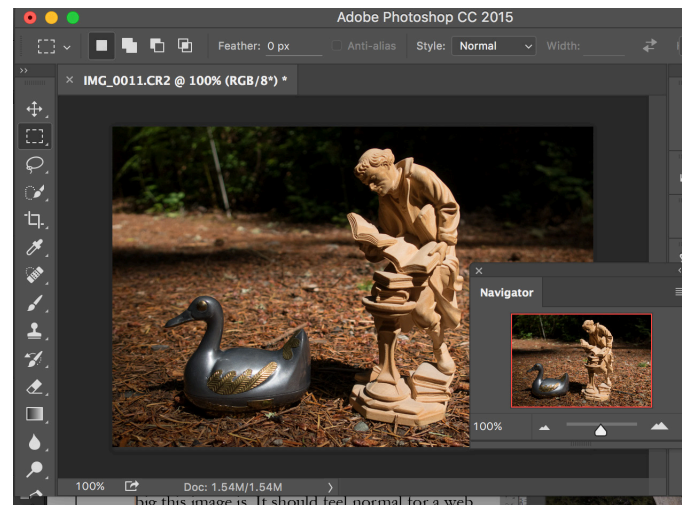
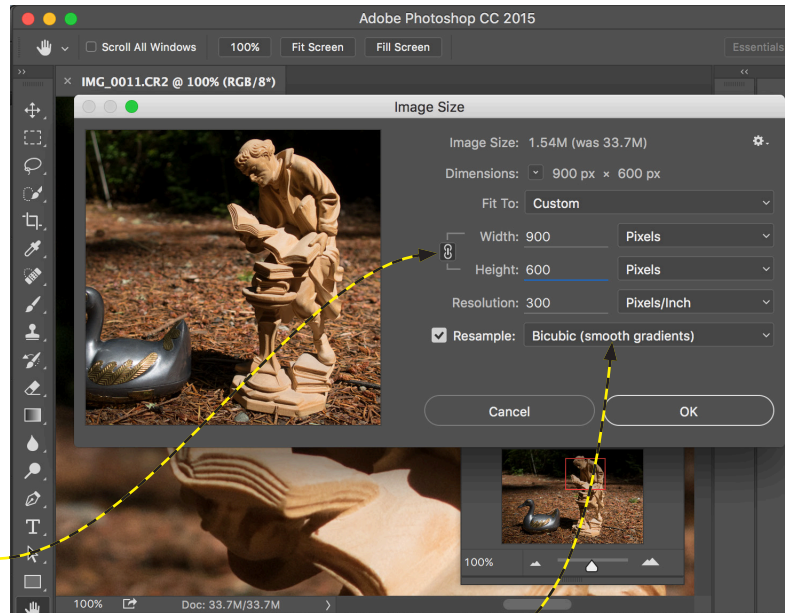
**STEP TWO:** Check the box for **resample** and ensure that width and height are linked via the depressed **chain** button.

**STEP THREE:** Choose: **Bicubic (smooth gradients)**. This controls how Photoshop manages the pixels as it resamples (tosses) out the extra pixels. This choice seems the best, but you can do your own testing if you wish.

**STEP FOUR:** Enter 600 for the height and click OK.

**NOTE:** the width doesn't matter nor does it matter if the image is landscape or portrait (wide or tall). The resolution setting is also irrelevant. All we care about is the pixel height value. 600 is a nice height. It allows the image to fit comfortably on most monitors and devices. If you knew for sure that most of your visitors would be using smartphones, you could go smaller. The percentage of smartphone users versus computer users varies by target market. You can find the stats by googling: browser stats for the current year, 2016 as of this writing.

**STEP FIVE:** Make Photoshop wide enough on your monitor so you can see the gray artboard outside the edge of the photo. Think about how big this image is. It should feel normal for a web page sized image. If it seems too big, back up in history (Ctrl + alt + z) until the image is full size again, and then repeat the **image > image size** process using smaller height numbers. You get to choose how big. Remember that not everyone has huge monitors. We designers tend to buy large monitors to handle Adobe



## Exporting jpgs

**STEP ONE:** Choose file > export> save for web legacy

**STEP TWO:** In the top right hand corner, choose Preset > JPEG High.

**STEP THREE:** Choose two up from the menu at the top left hand corner. Photoshop will show you the image before and after making it into a jpg.

**STEP FOUR:** Drag the quality slider left and right from zero to 100 percent quality. Observe the relationship between quality, and file size: Lower file size, equals lower quality.

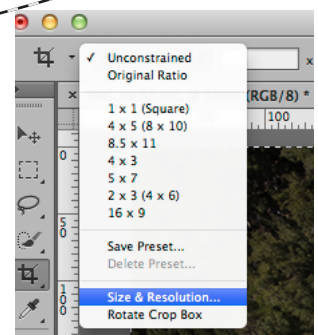
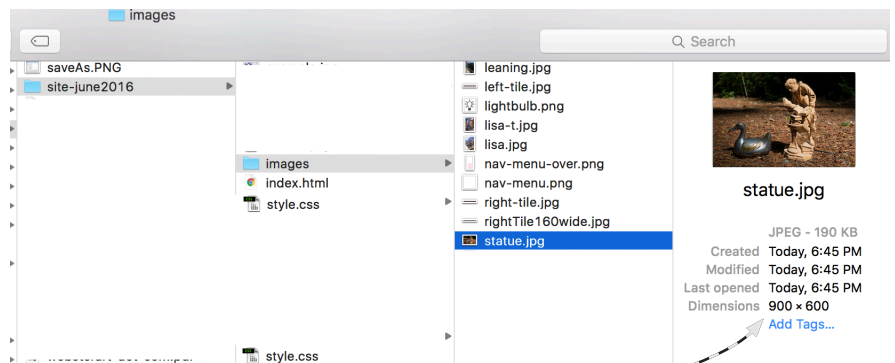
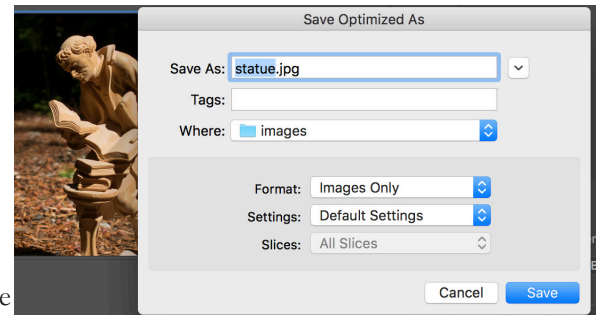
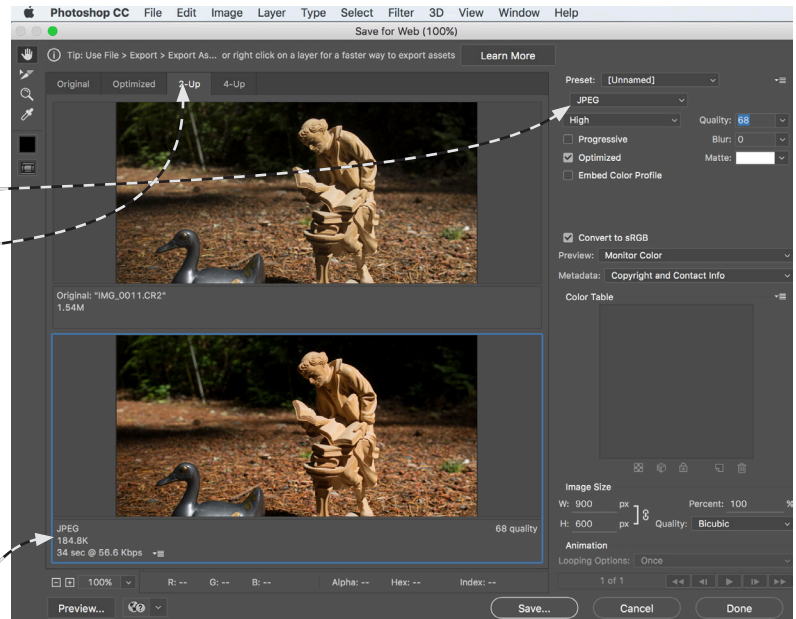
**STEP FIVE:** You get to pick the quality number here. It's nice to keep the file size under 200k, and the quality in the 60's, but I've been know to use quality of 85, with file sizes near 300k...it's really personal choice. In our web shop (last century), we used to use quality numbers in the 40's, but that was before broadband.

**STEP SIX:** Click the **save button** and navigate down into your website images folder: **john-smith/lesson xx/images**. Give the file a simple one word, lowercase name. On the file pictured here, I named it **statue.jpg**. If you don't put the .jpg extension on the file name, Photoshop will do it for you. It knows you want to make a jpg. Never use capital letters in a file name! **Click Save.**

**STEP SEVEN:** Minimize Photoshop and navigate down through the network storage to the images folder in your website folder. This screenshot is from Finder on a Mac, but Windows has a very similar folder view. Your operating system doesn't matter, what does matter is that you have the new jpg in the images folder, and you know what size it is (600 pixels tall), and you know it's name, including extension, which should be .jpg. Check and see!

**STEP EIGHT:** Go back to Photoshop where the image should still be sitting at 100% zoom.

**FILE MANAGEMENT:** Building a website is like managing a play. You have to know the address of the rehearsal hall and the live theater. You have to manage all of the actors, supporting personel and stage props. You have to make sure that they are all in the right location, at the right time in the script. If an actor has to appear halfway into the play from stage right, wearing a suit and tie, you need to give him those instructions. In web design, you need to be constantly aware of where your practice website lives, and where any related assets are located, such as images.





## Cropping the thumbnail jpg

**STEP ONE:** in the Image Size dialog box, enter a width of 200px and click OK. Photoshop will automatically pick the correct height.

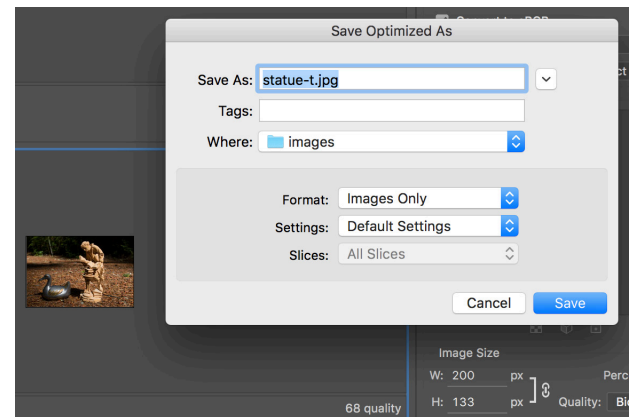
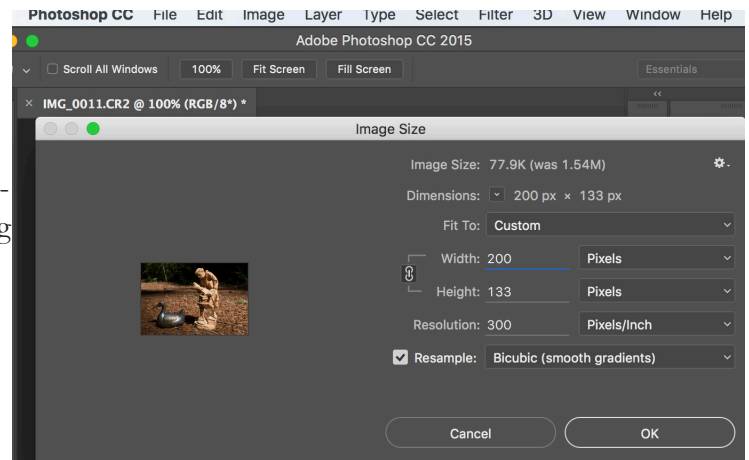
**NOTE:** I chose 200px as the width because we are going to put our thumbnail images in columns that measure approximately 200px wide. Also, 200 pixels is going to make the jpg very small. With small, fast loading thumbnail images we can load them into the browser very quickly, even on a bad data connection.

**STEP TWO:** Choose: file > save for web. Photoshop will remember all your settings from before as far as jpg quality and save to location.

Click the **Save** button. Note that the file size is much, much smaller, typically under 10 Kilobytes.

**STEP FIVE:** Give the file a similar name, but add a "-t" in front of the ".jpg" and click **Save**. So the big one (600px high) will be named **statue.jpg**, and the small one (200px wide) will be named **statue-t.jpg**

**NOTE:** "-t" (dash t) is a common way to indicate the file is a thumbnail, as opposed to the larger, 600 pixel tall jpg.



## figure and figcaption

We used to wrap images in div tags, and then style the div tags to look like little matted picture frames. That still works fine, but there is a way to present the images in code that has more semantic meaning to both search engines and browsers. HTML5 has a new tag that is specifically designed to make pictures frames for images. It's called the `<figure>` tag.

**STEP ONE:** Open index.html in your code editor.

**STEP TWO:** Locate the starting `<main>` tag. Make some blank lines below it and type the code you see to the right in bold brown.

**Explanation:** I added a class of thumb to the starting figure tag so we can style it up to make the picture frame around our thumbnail image. Also, the `figure` tag has a child element called `figcaption`. Any words you add inside the `figcaption` element are understood by the browser, search engines and screen readers. They know that those words describe the image.

**STEP THREE:** Locate the starting `<body>` tag and add an `id="home"` property as shown to the right in bold brown. We will use these id properties in our body tags to give us a hook on which to hang styles that only apply to one specific page, in this case the home page (index.html)

**STEP FOUR:** Open style.css and add this new `body#home` rule at the bottom of the style sheet. There **is** a space between the "e" and the period. That space means "that has a child of". This new style selector says: if there is a body tag with an id of home, and it has a child element with a class of main-area; the following rules apply. But only if all of that is true.

Bottom line, this new rule will give our main.main-area element a light gray background color, but only on the index.html page. The `display: block;` helps IE understand the correct size of the `<main>` element. Later when we have more pages in our website, such as gallery, resume and contact, we will style their respective body tags with unique id properties as we did this one.

**STEP FIVE:** Add a new `figure.thumb` rule. This should give us a white picture frame on a light gray background.

index.html

```
<main class="main-area">

  <figure class="thumb">

    <figcaption>
      Wooden statue
    </figcaption>

  </figure>

  <p>
    My name is John Smith. I am studying web design and
    development. This website is a vehicle for me to explore HTML5
    & CSS3. I will be showing images that demonstrate my
    skills with a variety of software programs used in developing
    websites. I will be graduating in the next year and plan to look
    for work in the greater Seattle area. If you like what you see,
    navigate to my contact page and let's get in touch!
  </p>

</main>
```

index.html

```
</head>

<body id="home">

  <div id="wrapper">
```

style.css

```
/** there is code above this line **/

body#home .main-area {
  background-color: #e0e0e0;
  display: block; /*hack for IE*/
}

figure.thumb{
  background-color: #fff;
  padding: 15px;
  margin: 5px;
}
```

## Picture framing

We have our picture frame, but the browser isn't sure how wide to make it, so it guesses at 100% of it's parent. I have a fix for that, but before we do that let's turn off the dotted borders and margins on the main and footer elements. Those were handy for setting up and aligning the initial interface elements, but now they are not needed.

**STEP ONE:** remove or comment out the border and margin rules on the two rules shown to the right. That should look like this:

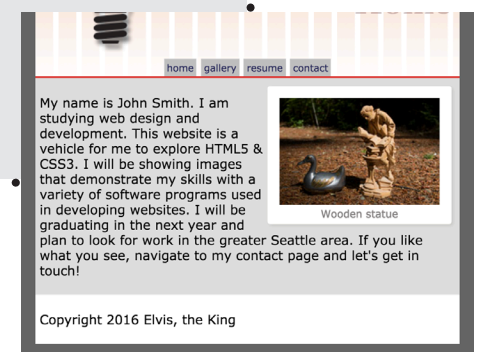
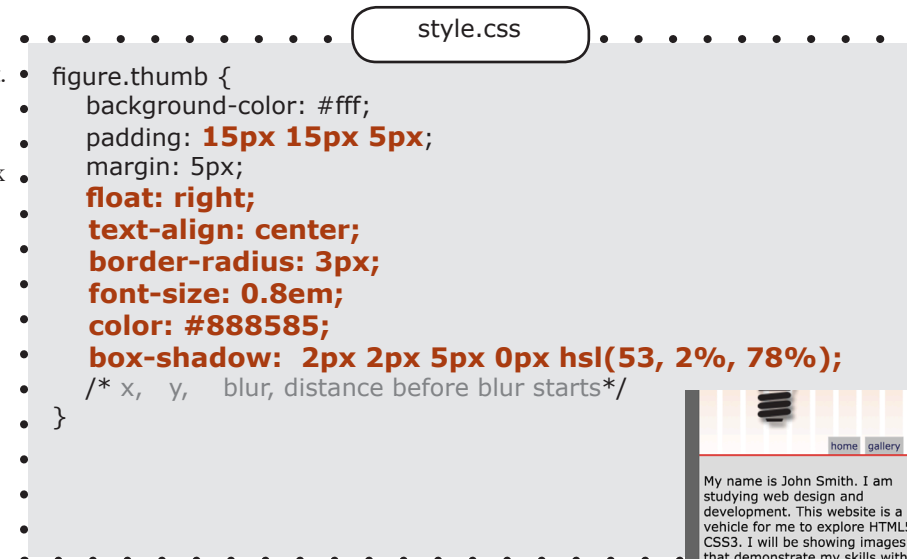
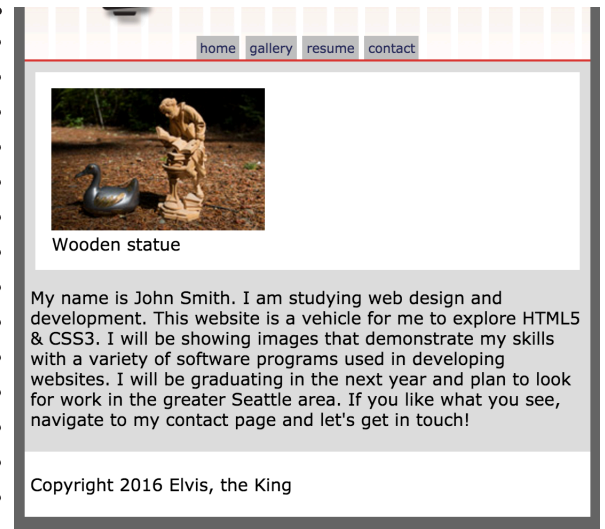
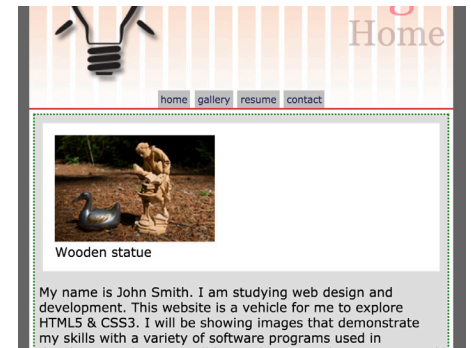
**STEP TWO:** modify the figure.thumb rule as shown in bold brown. This should give you a nicely framed thumbnail.

**EXPLANATION:**

**float: right;** this means that any `<figure>` tags this is applied to will literally float to the right. Any text that is nearby will flow around it to the left. Think of a rock in a river. The rock has been told to float: right; so the water flows around it on the left.

**padding: 15px 15px 5px;** There are 4 sides to the CSS box model. When you only specify 3 values, the first and last are the top and bottom padding, while the middle value can be thought of as the waist (left and right). I added round corners (radius) and a drop shadow. To get drop shadow code, google: "CSS3 box-shadow generator"

**STEP THREE:** If your picture breaks though the bottom of the footer, there is a fix coming up soon. Search the pdf for: `<br class="clear-float">`



## Style your footer

They say every story should have a beginning, middle and end. We have a nice looking header, our nav and content are looking good, it's time to style the footer. Since we end the nav with a red border, let's put the same red border on the top of the footer, and give it a nice background color that matches our gray color scheme. The font in a footer is normally smaller to indicate its lower importance as compared to the main content area and navigation menus. Footers are typically where you put your copyright info, phone numbers, business address, and alternate navigation.

**STEP ONE:** In your style.css file, find the style sheet rule for footer. Edit the footer style sheet rule so it looks like the brown code pictured to the right.

**STEP TWO:** Your webpage should look like the screenshot pictured below. If you have problems, compare your code with the code pictured on the next two pages.

A cool trick for debugging and comparing code is to set your notepad window side by side on your screen with my pdf file.

Put your left index finger on your first line of code. Put your right index finger on my first line of code. Drag your fingers along the lines of code in the two windows, pointing at each word on each side, comparing and looking for typing errors. Don't forget to clean your screen when you are done :-)

NOTE:

You should always work with WordWrap turned off when you are working on code. In Notepad choose: format > Word Wrap.

**STEP THREE:** After the last rule, add this **.clear-float** style sheet rule. It will fix a problem you may encounter on index.html where the floating thumbnail breaks through the footer.

Explanation:

**clear: both;** We want a **<br class="clear-float">** tag to be like a line in the sand. No thumbnail images should be allowed past it. Because the thumb is **floating**, we say **clear** all floats on **both** sides.

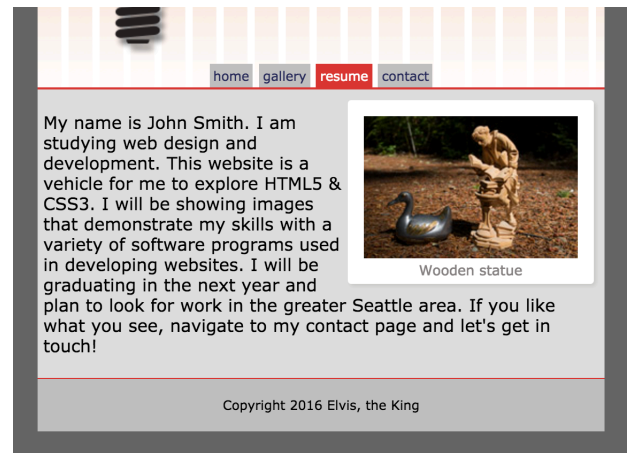
**height: 0;** We want the **<br>** tag to function, but not to have any height.

**font-size: 0.2em;** same idea. This is a bit of a hack, but it works.

**STEP EIGHT:** To use this tag, open index.html, locate the closing **</main>** tag and place a

style.css

```
/** there is code above this line **/
footer.footer-area {
    border-top: 1px solid #db3737;
    margin: 0;
    padding: 5px;
    background-color: #c2c2c2;
    font-size: 0.75em;
    text-align: center;
}
/** there is code below this line **/
```



index.html

```
<main class="main-area">
  <figure class="thumb"><!-- this floats right -->
    <a href="images/statue.jpg">
      
    </a>
    <figcaption>
      Wooden statue
    </figcaption>
  </figure>
  <p>My name is John Smith. Lots of words go here...
</p>
  <br class="clear-float">
</main>
<footer class="footer-area">
  <p>Copyright 2016 John Smith</p>
</footer>
</div> <!-- end wrapper div -->
```

style.css

```
.clear-float {
    clear: both;
    height: 0;
    font-size: 0.2;
}
```

## Complete style.css (so far)

style.css

```
body {
  margin: 0;
  padding: 0;
  background-color: #666;
  font: 1em Verdana, Helvetica, sans-serif;
}
#wrapper {
  margin-right: auto;
  margin-left: auto;
  padding: 0;
  width: 80%;
  background: #fff url(images/headerPinstripe.
png) top left repeat-x;
  max-width: 1400px;
  background-color: #fff;
}
.center {
  display: block;
  margin-right: auto;
  margin-left: auto;
}
header.masthead {
  margin: 0;
  padding: 0;
  height: 215px;
  /* this makes room for height of lightbulb.png */
  background: url(images/lightbulb.png) top left
no-repeat;
}
header.masthead h1 {
  position: relative;
  top: 60px;
  right: 10px;
  padding: 0;
  margin: 0;
  font: 4em/0.8em Georgia, Times, serif;
  text-align: right;
  color: #4554af;
  /**this is fall back color for old browsers**/
  color: hsla(348, 100%, 50%, 0.36);
  /***hsla = Hue, Saturation, Lightness, Alpha, 0.3
= 30% opacity***/
}
header.masthead h1 span {
  font-size: 0.7em;
  color: hsla(0, 20%, 38%, 0.3);
}
nav.main-menu {
  margin: 0;
  padding: 0;
  text-align: center;
  border-bottom: 2px solid #db3737;
  padding-bottom: 0.01em;
}
```

These short  
arrows mean code  
should be all on  
one line

style.css - continued

```
nav.main-menu ul {
  list-style: none;
  margin: 0;
  padding: 0;
}
nav.main-menu ul li {
  display: inline-block;
}
nav.main-menu ul li a {
  display: block;
  text-decoration: none;
  padding: 0.3em;
  background-color: #c2c2c2;
  font-size: 0.75em;
  color: #2c2e5c;
  /* margin-bottom: 0.015em; */
}
nav.main-menu ul li a:hover {
  color: #fff;
  background-color: #db3737;
}
main.main-area {
  padding: 5px;
}
footer.footer-area {
  border-top: 1px solid #db3737;
  margin: 0;
  padding: 5px;
  background-color: #c2c2c2;
  font-size: 0.75em;
  text-align: center;
}
body#home .main-area {
  background-color: #e0e0e0;
  display: block; /*hack for IE*/
}
figure.thumb {
  background-color: #fff;
  padding: 15px 15px 5px;
  margin: 5px;
  float: right;
  text-align: center;
  border-radius: 3px;
  font-size: 0.8em;
  color: #888585;
  box-shadow: 2px 2px 5px 0px hsl(53, 2%, 78%);
  /* x, y, blur, distance before blur starts */
}
```

## Complete index.html [so far]

index.html

```

<!doctype html>
<html lang="en">

<head>

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="IE=edge" >
<title>
Web Design - Home
</title>
<link rel="stylesheet" type="text/css" media="screen" href="style.css">
</head>

<body id="home">

<div id="wrapper">
  <header class="masthead">
    <h1>Web Design<br><span>Home</span></h1>
  </header>

  <nav class="main-menu">
    <ul>
      <li><a href="index.html">home</a></li>
      <li><a href="gallery.html">gallery</a></li>
      <li><a href="resume.html">resume</a></li>
      <li><a href="contact.html">contact</a></li>
    </ul>
  </nav>

  <main class="main-area">
    <figure class="thumb">
      
      <figcaption>
        Wooden statue
      </figcaption>
    </figure>
    <p>My name is John Smith. I am studying web design and development. This website is a vehicle for me to explore HTML5 & CSS3. I will be showing images that demonstrate my skills with a variety of software programs used in developing websites. I will be graduating in the next year and plan to look for work in the greater Seattle area. If you like what you see, navigate to my contact page and let's get in touch! </p>
  </main>

  <footer class="footer-area">
    <p>Copyright 2016 Elvis, the King</p>
  </footer>
</div> <!-- end wrapper div -->

```

**STEP ONE:** Add an alt="wooden statue" to your image tag, as shown above. The alt property stands for alternate image description.(for blind people). If you try to validate your code, you will get an error without the alt property. The alt property also gives you a tool tip on mouse hover over the image. To get a hover in IE \*and\* Firefox, you also need a title property. Both together are written like this:

```

```



## Clickable Thumbnails

The point of using thumbnails on a webpage is to speed up the display of the page. Imagine opening a library book, and the pictures took 30 seconds to appear. You don't want to do that to your web page visitors. People have very short attention spans on the internet. It's ok to have a webpage with 20 images, but you should never show the big ones until you are sure your visitor really wants to see them. For that we use thumbnails. If they see a thumbnail that looks cool, they will click on it, which indicates they are willing to wait for it to download. We can program that function with the following steps. We will make just one clickable thumbnail, and return to fill the rest out later. Before we start adding more images, we need to get this website working on the internet. And that requires file management and spelling skills, all of which will become apparent in the next few pages.

**STEP ONE:** to make your thumbnail link to the bigger picture that we made, you have to surround the image tag with starting and stopping anchor tags. The href value in the `<a>` tag tells the browser where it should go (a folder named **images**) and what file it should open

**statue.jpg :**

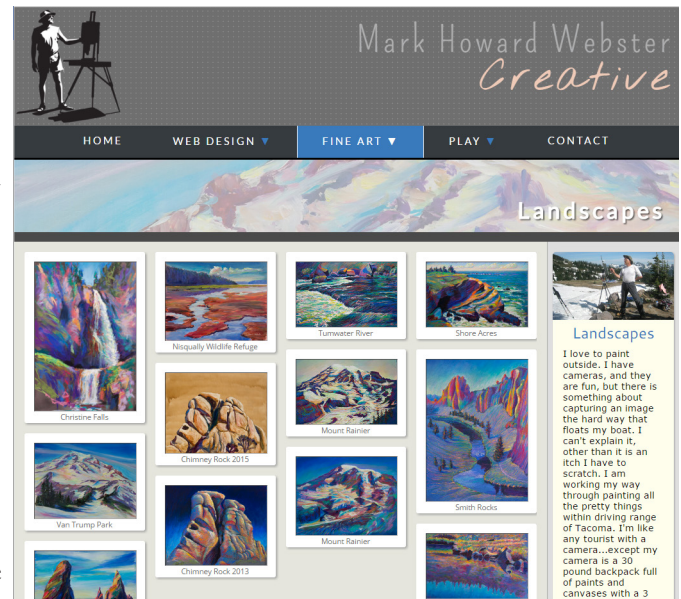
```
<figure class="thumb">
  <a href="images/statue.jpg">
    
  </a>
  <figcaption>
    Wooden statue
  </figcaption>
</figure>
```

When you save your changes and view it in the browser you will see that you can now click the thumbnail. For this code to work, you have to have the statue-t.jpg, and the statue.jpg in the images folder. Your thumbnail may be outlined in blue on the webpage. This blue border is a result of the browser trying to be helpful. The browser has been programmed to "underline" every text link. Text should be underlined, if it is click-able, but a thumbnail is expected (by internet savvy people) to be click-able, so there is no reason for it to be "underlined" in blue.

**STEP TWO:** to disable the "underlining" blue link border on the thumbnail, open your style.css file in Notepad and add this new rule to the bottom of the style sheet:

```
img{border: none;}
```

**STEP THREE:** If you would like to show artwork you created in Illustrator or InDesign in your gallery, here are a couple tricks to get the artwork into Photoshop with minimal stress. Organize the Illustrator or InDesign file so that you can see it to best advantage (zoom in or out, focus on best part, etc), then press tab to hide the palettes. On the PC, Windows 7+, open the Snipping Tool, drag a marquee around the artwork you want to capture. Go to Photoshop, open a new document and choose edit paste to bring the image from your clipboard to Photoshop. On the mac, press command+control+shift+4 and drag over the area you want to place on the mac clipboard, then paste into a new photoshop file. To export out the screenshot from Photoshop, choose file > save for web as we did for the previous images. Don't forget to make a thumbnail.



## Add a gallery page

We've got a good looking webpage. It has a clean interface structure that will look good anywhere, on any device. Thanks to our careful work on the organization of the element boxes, it is built to be friendly to search engines and screen readers (blind people). We can continue to refine the look and feel via the external style.css file. But we need to start expanding this webpage into a real web site, with multiple pages. There is no sense in reinventing the wheel here, so let's use our index.html page as a template to make more pages.

**STEP ONE:** Close any code or browser windows you have open. Navigate down into your current working folder, it may be named "lesson-3". Copy the index.html file, and then paste it right back into the same folder. It should come into the folder with a name of index copy.html, or something similar depending on if you are mac or pc.

**STEP TWO:** change the name of the copied file to **gallery.html**

NOTE: do not use capital letters, do not call it myGallery.html.

Keep it simple: **gallery.html**

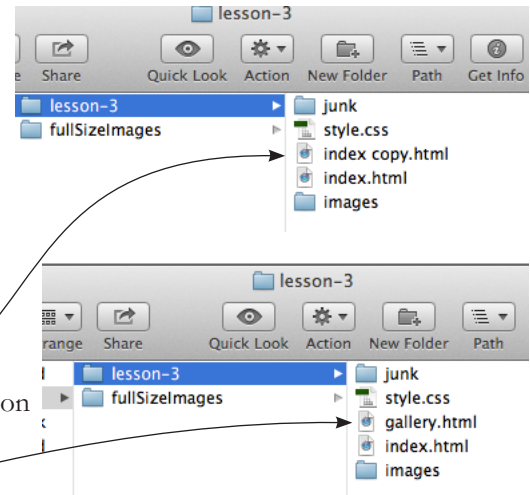
**STEP THREE:** right click on the gallery.html file and choose: open with > Notepad.

**STEP FOUR:** In your <title> tag, change the word Home to Gallery

**STEP FIVE:** In your <h1> tag, change the word Home to Gallery.

**STEP SIX:** In your body tag, change the id to **gallery**.

**STEP SEVEN:** Down in your <nav>, examine your href properties. Notice that there is an href="index.html", and an href="gallery.html". If your visitor clicks those words in the browser, the browser will navigate to, and open the files named in the href property. Be very careful about changing either the href property, or the name of the webpages in your website. If you have an href property of href="gallery.html", but you name your file **Gallery.html**, or **myGallery.html**, the browser will not be able to find the page, especially once it is up on a live server. Capital letters are considered different characters than lowercase letters, on the internet.



gallery.html

```
<!doctype html>
<html lang="en">

<head>

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="IE=edge" >
<title>
Web Design - Gallery
</title>
<link rel="stylesheet" type="text/css" media="screen" href="style.css">
</head>

<body id="gallery">

<div id="wrapper">
  <header class="masthead">
    <h1>Web Design<br><span>Gallery</span></h1>
  </header>

  <nav class="main-menu">
    <ul>
      <li><a href="index.html">home</a></li>
      <li><a href="gallery.html">gallery</a></li>
      <li><a href="resume.html">resume</a></li>
      <li><a href="contact.html">contact</a></li>
    </ul>
  </nav>
```

## Navigating between gallery and home

**STEP ONE:** Navigate into your current homework folder, right click on the **index.html** file, choose open with Firefox or Chrome. Click the **gallery button**. You should see the new **gallery.html** page open. Check to see that your title says Gallery, and that your subheader says Gallery.

If your browser window is wide enough to see the address bar, you can see the file names change with each click.

**STEP TWO:** click the link back to home, the browser should navigate back to the home page (index.html). This is called navigating via hyperlinks, and it is what made the internet. Instead of the tedious process involved in opening files from folders by drilling down through nested, and often buried folders using Windows Explorer, or Mac Finder, we set up navigation in HTML code using

`<a href="">` `</a>` tags. HREF stands for Hypertext REFerence. `<a>` stands for anchor.



## Uploading your website to the Internet

We need a software program that can upload files to the internet. You have done something similar if you uploaded photos to facebook. Ask Uncle Google for "free ftp upload software". FTP stands for File Transfer Protocol. We will use Dreamweaver, it has a built in ftp uploader.

Before we start using Dreamweaver, we need to configure it so it can work correctly.

Dreamweaver is very capable of completely destroying your website if you ignore some basic rules.

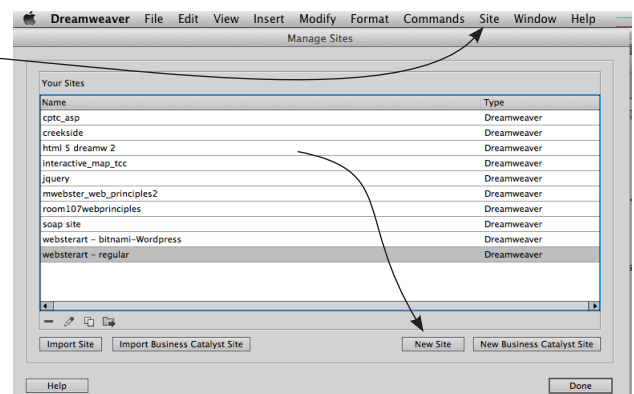
First, Dreamweaver needs to know where your website lives locally. Is it on your flash drive, your hard drive, or on the local network?

Second, Dreamweaver needs to know where your website will be hosted on the internet. Is it at your domain name (www.mywebpage.com) or is it on some free web hosting site? Dreamweaver has to know where it will carry files from, and where it will carry them to. Dreamweaver's ftp client is a lot like a taxi in that it needs to know where it is picking you up, and where it is dropping you off.

**STEP THREE:** Open Dreamweaver. Once it starts, click: **Site > Manage Sites**

**STEP FOUR:** Click **New Site**

**NOTE:** You won't have this long list of pre-existing sites. You may have a few in there, if other people use your computer. I will talk more about this during the lecture.



## Configure Dreamweaver for uploading

**STEP ONE:** type your name in the Site Name box.

**STEP TWO:** In the Local Site Folder box, Dreamweaver is asking you where you store your website locally. We store our websites in the campus network called Room 107 storage. You will need to click the folder button to the right of the box, and drill down to your parent homework folder for the website files. In my classes, this is typically called **Lesson-3**, or due-next-class-day as in (due10-29). Whichever it is, we need to tell Dreamweaver exactly where to find the folder that contains your **index.html** page. **Do not** select your John Smith folder, you need to drill inside that and choose the parent folder for the **index.html** file.

**STEP THREE:** Click the **choose** button when you can see your **index.html** file, and your images folder.

**STEP FOUR:** Examine the path in the Local Site Folder box. If you are at this college, it will look similar to this:

\\ms1729\shared storage\room107storage\mwebster\web principles\john-smith\lesson-3\

This long path is basically a DOS path, telling Dreamweaver exactly where your files live on the campus network. If you try this at home, make sure you put the parent folder somewhere on your hard drive where you can find it, and guide Dreamweaver to that location. NOTE: I am writing this at home, so my address is different.

**STEP FIVE:** Click the button for **servers**. This is where we tell Dreamweaver where the website will be hosted on the Internet.

**STEP SIX:** Click the "plus" sign to add a new server.

**STEP SEVEN:** Choose: Connect using: **SFTP**

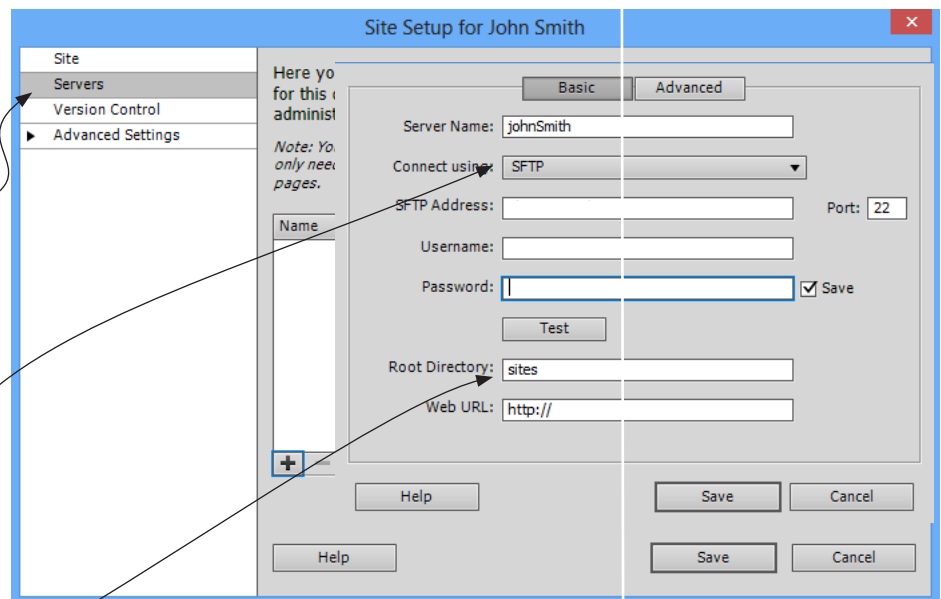
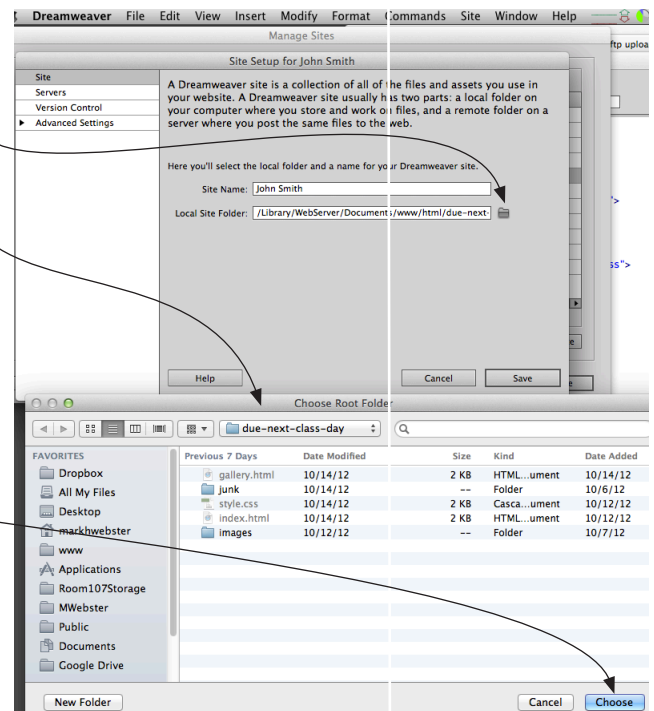
**STEP EIGHT:** SFTP Address: will give in class

**STEP NINE:** Enter the username and password, which I will give you in class. If you bought your domain name, they will come in the email from your provider.

**STEP TEN:** Root Directory: **sites**

**STEP ELEVEN:** Click the Test button. If there is any pause at all, it didn't work, check your settings.

**NOTE:** The port is automatically set to 22, the server name box & Web URL boxes don't matter.

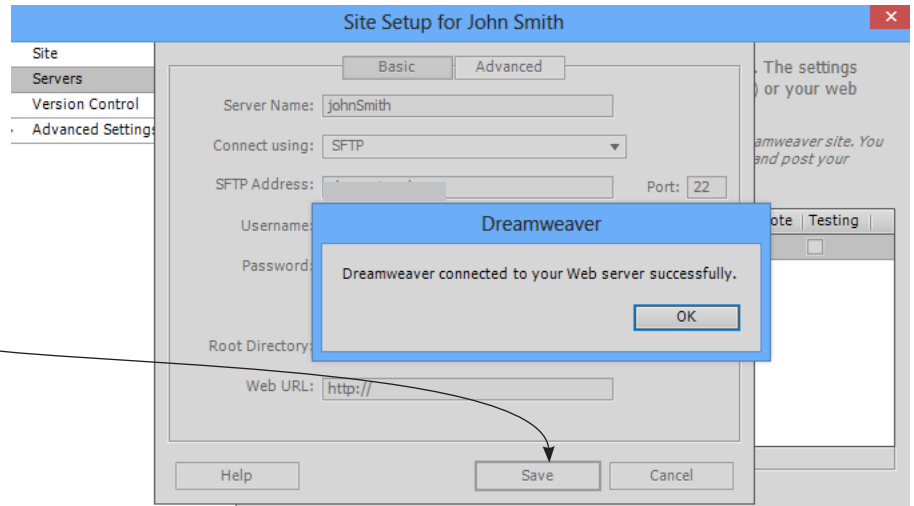


## Connecting to remote server

**STEP ONE:** If you get this successful connection message, click OK.

**NOTE:** this can be tricky...I will assist in class.

**STEP TWO:** After you've connected, click the Save button. This preserves all the server settings and passwords within the Dreamweaver software installed on your local hard drive.

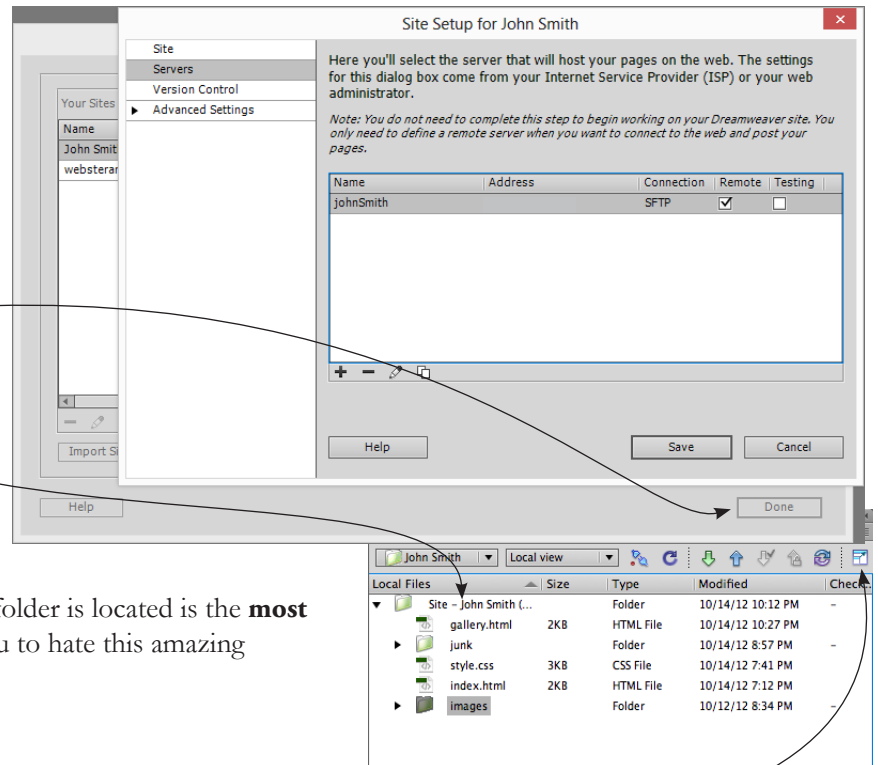


**STEP THREE:** Click the **Save** button again on the next dialog box, and then click the **Done** button.

**STEP FOUR:** You should now see the files panel. If you don't, choose: window > files

**STEP FIVE:** Hover your mouse over the top folder, it should popup a tooltip menu giving you the directory path of where the parent folder of your website is located. Make a habit of checking this! Loosing track of where your website parent folder is located is the **most common** Dreamweaver error, and it will cause you to hate this amazing piece of software.

**STEP SIX:** Click the expand/collapse button.

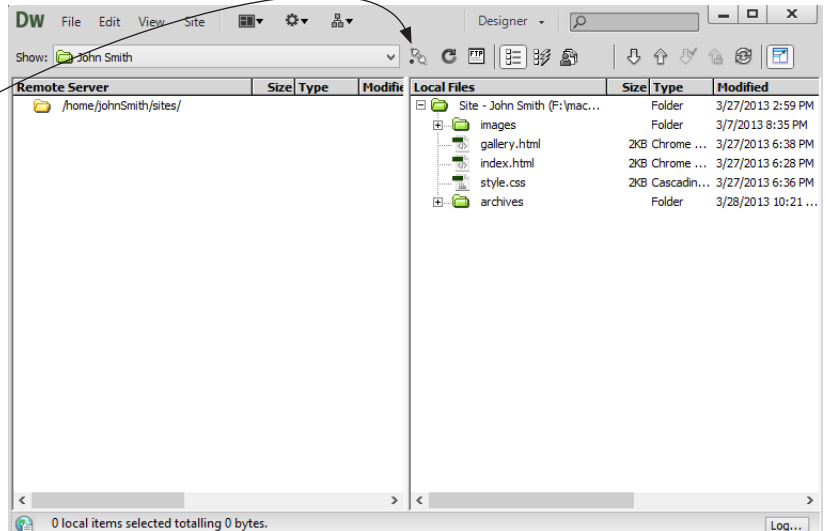




## Connect

**STEP ONE:** Note that the right side of the Site panel says “local files”, while the left side says “Remote Server”

**STEP TWO:** Click the “connect” button and Dreamweaver should connect to the server on the internet and show you the folders it sees up there.



## WARNING

**Do not post photos of famous people on the website you build for this class!**

Do not grab photos of sports stars, movie stars, or any other famous person, and post them on your web page. Famous people have agents and lawyers who spend their days looking for people like you who "borrow" their clients images for personal gain.

When a famous person finds their image on your website, they will call the president of this college, who will call my dean, who will call me. I don't want to get that call...again.

Be careful what you put on your website. It must all be PG rated, and you must not use images that you don't own. Anyone can see these pages, it's not like Facebook where you can restrict access.

Your access to this server will last as long as you are a student here and taking web classes. If you need it to last longer, talk to me in private.

If you need a permanent place to host your website, with your own domain name, search google for "web hosting reviews"

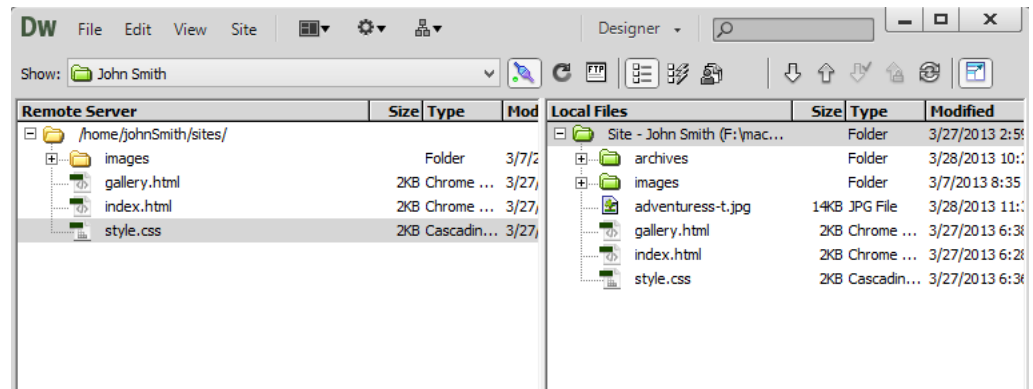
<http://www.top10bestwebsitehosting.com/>

Hostgator has been in the top ten for several years. Cheaper is not necessarily better. Ask me during lab for more information.



# Uploading your website

**STEP ONE:** Highlight your files on the right side of the window (local files).



**STEP TWO:** Drag and drop from right side to left side.

NOTE: you can't move the very top (root) file on the right side. Also, when you get over to the left side, it helps to drop your files on the very top, root folder, named **/home/johnSmith/sites/** if you are John. You will see a progress bar as the files are transferred from your lesson-3 homework folder up to the remote server on the Internet.

**STEP THREE:** Open up any browser (including your smartphone) and go to this internet address:

<http://elmo.cptc.edu/johnSmith/>

If your name is James Smith your address would be:  
<http://elmo.cptc.edu/jamesSmith/>

NOTE: careful capitalization is iMpOrTaNt!

**STEP FOUR:** Click the button for home and gallery. The browser should easily navigate back and forth between pages. If any images are missing, open your page in Notepad and examine the names of the image files. For example, if you have some code that says ``, and it has worked up until now when you put it on the internet...chances are you named the file in your images folder **Dog.jpg**

Neither your home computer, nor the campus network storage folder cares about capitalization conflicts between file names and code...but the server **does** care, it considers a capital letter to be different character than a lowercase letter.



## You are here flag

We need an indicator that tells our web site visitor which page they are visiting. Typically this is done in the navigation menu. The current page's button is lit up to differentiate it from the other buttons. Like an elevator's floor list, it lights up the number for the current floor. It says: You are here!

Web Designers call this the You Are Here flag, or, for short, urhere. Let's make that happen in our website.

**STEP ONE:** Open your style.css file in Notepad.

**STEP TWO:** locate and copy all 5 lines of this style sheet rule:  
**nav.main-menu ul li a:hover**

**STEP THREE:** Make some blank lines below that style sheet rule, and then paste it in as a duplicate style sheet rule.

**STEP FOUR:** Edit the selector so it reads like this:

This is called a multiple selector style sheet rule. We are telling the browser that if it finds a match for any of those four selectors, use the declarations in the curly braces. Note the last selector does not have a comma.

**STEP FIVE:** Open your index.html file in Notepad.

**STEP SIX:** Locate the starting anchor tag `<a>` for each of the navigation buttons. Click to the left of the closing greater than (`>`) sign. Add a space, and then type `class="home"`, or `class="gallery"`, as shown.

**EXPLANATION:** When the browser opens either index.html, or gallery.html, it will look for an element with an **id of home**. If it finds that, and a child anchor tag with a **class of home** it will apply the declarations which make the button red. This will also work on the resume and contact pages, when we make those later on.

style.css

```

/** there is code above this line */
nav.main-menu ul li a {
    display: block;
    text-decoration: none;
    padding: 0.3em;
    background-color: #c2c2c2;
    font-size: 0.75em;
    color: #2c2e5c;
}

nav.main-menu ul li a:hover {
    color: #fff;
    background-color: #db3737;
    border-bottom-color: #db3737;
}

#home nav.main-menu ul li a.home,
#gallery nav.main-menu ul li a.gallery,
#resume nav.main-menu ul li a.resume,
#contact nav.main-menu ul li a.contact
{
    color: #fff;
    background-color: #db3737;
    border-bottom-color: #db3737;
}

```

index.html

```

<body id="home">
<div id="wrapper">
<!--
more code goes here, not shown
-->

<nav class="main-menu">
  <ul>
    <li><a href="index.html" class="home">home</a></li>
    <li><a href="gallery.html" class="gallery">gallery</a></li>
    <li><a href="resume.html" class="resume">resume</a></li>
    <li><a href="contact.html" class="contact">contact</a></li>
  </ul>
</nav>

<main class="main-area">
  <p>lots of words go here...</p>
</main>

```

## You are here flag concluded

**STEP ONE:** Close the index.html file and save the changes.

**STEP TWO:** Open the gallery.html file in Notepad.

**STEP THREE:** Repeat this process on the **gallery.html** page.

NOTE: these properties we are adding to the anchor tag can be thought of as adjectives. For example, if we had a car tag, it would look like this. You **must** put spaces between each property.

**<car** make="ford" doors="4" color="red" year="2016" model="mustang" transmission="automatic" horsepower="250">

```

    . . . . .
    . . . . . gallery.html . . . . .
    . . . . .
    <body id="gallery">
    <div id="wrapper">

    <!--
    more code goes here, not shown
    -->

    <nav class="main-menu">
      <ul>
        <li><a href="index.html" class="home">home</a></li>
        <li><a href="gallery.html" class="gallery">gallery</a></li>
        <li><a href="resume.html" class="resume">resume</a></li>
        <li><a href="contact.html" class="contact">contact</a></li>
      </ul>
    </nav>
    . . . . .
  
```

**STEP FOUR:** Open **index.html** in Firefox. Note the home button. It is permanently red. This is called the **You are here flag**. Like the light on the elevator, it tells you what floor (page) you're on. Click the link to gallery. The gallery button should be lit up permanently.

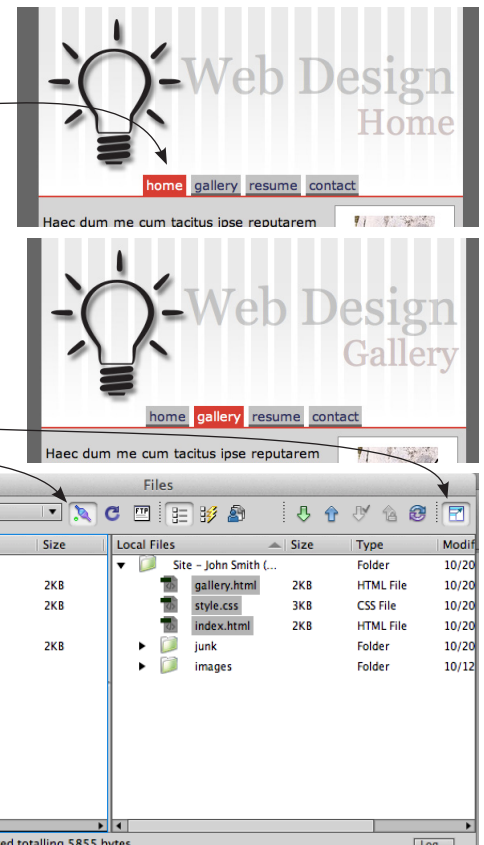
**STEP FIVE:** Now that you have made some cool changes, let's upload the changes to the internet. If you closed Dreamweaver, re-open it. Click **window > files**.

**STEP SIX:** Click the expand button so Dreamweaver shows you the remote server side.

**STEP SEVEN:** Click the connect button

**STEP EIGHT:** On the right side, shift select the 3 changed files: gallery, style, index

**STEP NINE:** Drag them from right to left, and then go out to the internet and check them at <http://elmo.cptc.edu/johnSmith/>



## Dreamweaver upload errors

Here on the campus computers, you may get an error when you upload files. The message will be something like this: *dwsync is unable to be updated.* This is Dreamweaver trying to memorize the times stamps on all the files in the site, both local and remote. For it to do this it has to change files deep in the hard drive of your computer. Our IT department has locked down the hard drives of these classroom computers to protect them from viruses, and this dwsync thing is intercepted as a virus.

When you see this error message it is usually right after uploading. Simply click ok several times until it goes away. Your files uploaded just fine, but Dreamweaver was not able to track the time stamps and store them in the internal sync file.

You will never get this message at home because you have administrator privileges there, and Dreamweaver can do it's thing unhindered. Here on campus, these are the steps to follow to stop seeing the error message.

**STEP ONE:** Choose site > manage sites.

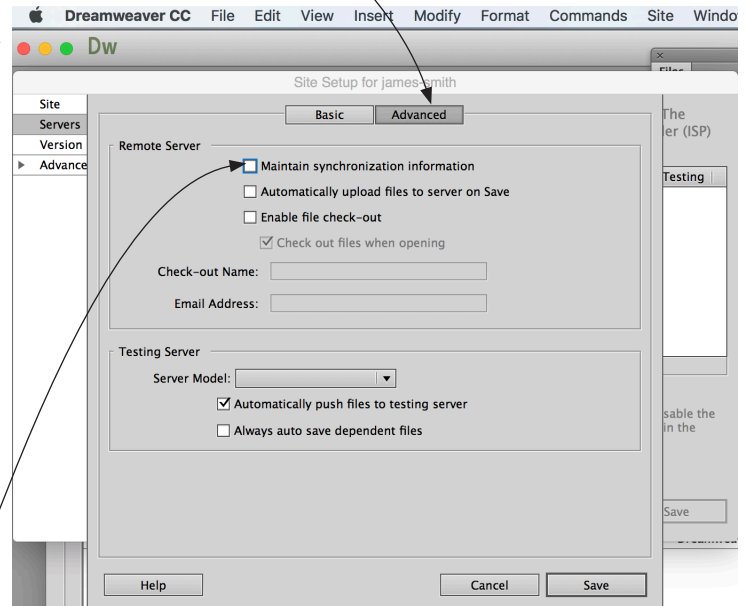
Double click your named site.

Go to servers.

Double click your defined server (elmo)

Click the Advanced tab

**Uncheck:** Maintain synchronization information



# Uploading without Dreamweaver

Dreamweaver is not the only game in town for uploading. You can also use a free FTP client called filezilla. It's dialog boxes are slightly different than Dreamweaver, but the concept is the same, and it's free open source software. If you are in my lab at college, I can show you how to use it. One tricky thing is that you have to manually enter port 22, which Dreamweaver gets automatically when you choose SFTP. And filezilla isn't as specific about "defining a site". If you are in my college class, and you use filezilla, be sure you only upload into the "sites" directory. Otherwise you can corrupt your account.

If you are using a commercial account like godaddy or hostgator, read their instructions carefully. There is usually a specific subdirectory into which you must upload your files. All of that information will be in the email you receive from them when you set up and pay for your account.

To find a good place to host your website, do a search for: "website hosting reviews 2016". Web hosting companies, (like laptop makers) change over the years, and a good company 4 years ago may have been bought out by someone new who does not put as much emphasis on customer service. Be sure and do your research before you plunk down your money. Most web hosts will combine domain name registry with hosting space in one package. It is possible to do it separately. Do a web search for "domain name registry".

**FileZilla** The free FTP solution

Home  
**FileZilla**  
 Features  
 Screenshots  
 Download  
 Documentation  
**FileZilla Server**  
 Download  
**Community**  
 Forum  
 Project page  
 Wiki  
**General**  
 FAQ  
 Contact  
 License  
 Privacy Policy  
**Development**  
 Source code  
 Nightly builds  
 Translations  
 Version history  
 Changelog  
 Issue tracker  
**Other projects**  
 libfilezilla  
 Octochess  
 PayPal donate  
 Sponsors: NOD

**Overview**

Welcome to the homepage of FileZilla, the free FTP solution. Both a client and a server are available. FileZilla is the terms of the GNU General Public License

Support is available through our [forums](#), the [wiki](#) and the [bug and feature request trackers](#).

In addition, you will find documentation on how to compile FileZilla and nightly builds for multiple platforms in

**Quick download links**

**Download FileZilla Client**  
All platforms

**Download FileZilla Server**  
Windows only

Pick the client if you want to transfer files. Get the server if you want to make files available for others.

**News**

**2016-08-11 - FileZilla Server 0.9.58 released**

**New features:**

- TCP send buffer auto-tuning
- Performance improvements to reduce CPU usage under high load
- Disabled IDEA and SEED ciphers for FTP over TLS

**Bugfixes and minor changes:**

- Fixed potential crash if closing connections with pending socket messages
- A missing home directory is no longer treated like an empty directory

**2016-08-03 - FileZilla Client 3.20.1 released**

**Bugfixes and minor changes:**

- Fixed rename file exists action on downloads

TopTenReviews Compare the Best, Buy the Best for You

Our editorial staff evaluates products and services independently, but Top Ten Reviews may earn money when you click on links. [Learn More](#)

**2016 BEST** Web Hosting Reviews  
 SERVICES / Web Hosting / Web Hosting Review

**inmotion hosting** Start now for **\$2.95** [Visit Site](#)

| Rank | Provider         | Price  | Overall Rating |
|------|------------------|--------|----------------|
| 1    | inmotion hosting | \$2.95 | 9.73 / 10      |
| 2    | arvix            | \$3.20 | 9.65 / 10      |
| 3    | just host        | \$2.50 | 8.95 / 10      |
| 4    | bluehost         | \$3.49 | 8.85 / 10      |
| 5    | GreenGeeks       | \$3.96 | 8.63 / 10      |
| 6    | Webhostingpad    | \$1.99 | 8.63 / 10      |
| 7    | HostGator        | \$3.45 | 8.53 / 10      |
| 8    | DreamHost        | \$7.95 | 8.35 / 10      |
| 9    | GoDaddy          | \$5.99 | 8.30 / 10      |
| 10   | HostMonster      | \$3.49 | 8.28 / 10      |

Overall Rating: [Overall Rating](#) [Hosting Package](#) [Help & Support](#) [Control Panel Options](#) [Security](#)



## Filling out your gallery page

**STEP ONE:** Prepare at least 6 more images for your gallery page. You will need thumbnails and large images. Refer back to page 48 for instructions.

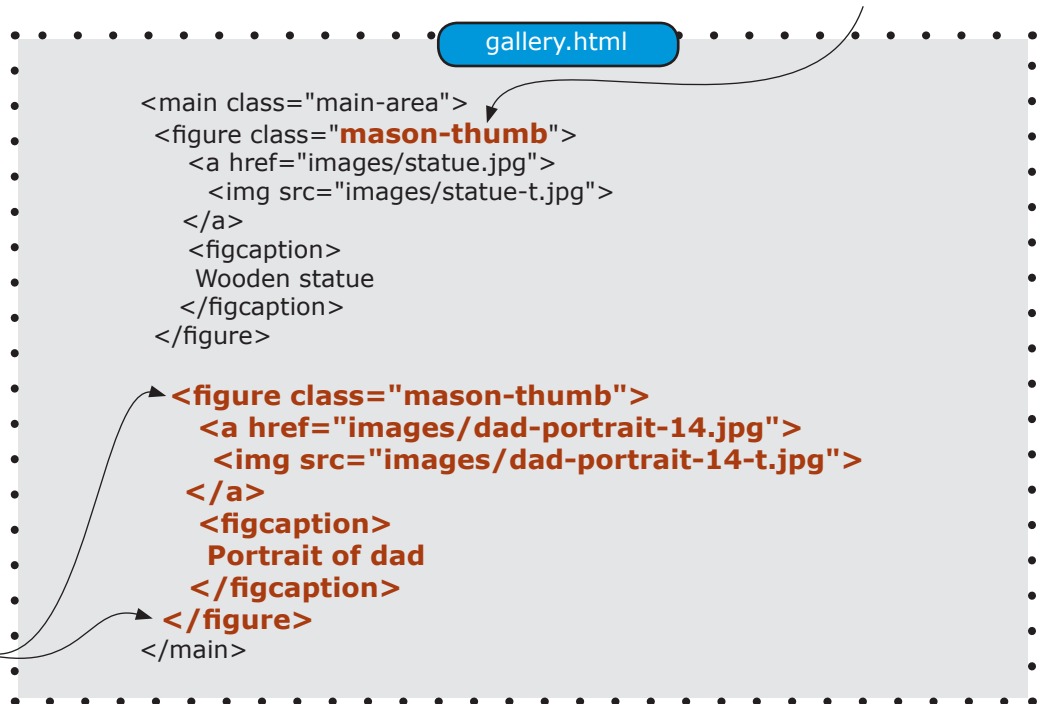
**STEP TWO:** Open gallery.html in Notepad. Change the class property on the figure tag to: **mason-thumb**.

NOTE: changing the class will allow us to write a special style sheet rule that will only apply to thumbs on the gallery page, and will not affect our thumb on index.html

**STEP THREE:** Copy the entire figure tag from figure start to figure stop: **<figure></figure>**

**STEP FOUR:** Make some blank lines below it and paste it back in as a duplicate figure tag.

**STEP FIVE:** Edit the references to images and the figcaption to reflect the new thumbnails you would like to have on your gallery page. You can have as many as you like, but put at least 4 there. Make sure that for each image, you have the big and the small images in the images folder. Then check each small image (dog-t.jpg) to make sure it is exactly 200 pixels wide.



## New figure style rule

**STEP ONE:** Open style.css

**STEP TWO:** copy the entire `figure.thumb` style sheet rule, then paste it in below on a new line

**STEP THREE:** edit the selector to read: `figure.mason-thumb`.

**STEP FOUR:** Comment out the `float: right;` declaration. Floating elements is a useful tool to have in your toolbox, however, HTML5 has some positioning tricks that are far superior to old school floating.

**STEP FIVE:** edit the `body#home .main-area` rule as shown. Note the coma after the first selector. I am saying that for either the gallery or index pages, make the background of main be a light gray color.

style.css

```
figure.thumb {
    background-color: #fff;
    padding: 15px 15px 5px;
    margin: 5px;
    float: right;
    text-align: center;
    border-radius: 3px;
    font-size: 0.8em;
    color: #888585;
    box-shadow: 2px 2px 5px 0px hsl(53, 2%, 78%);
    /* x, y, blur, distance before blur starts*/
}
figure.mason-thumb {
    background-color: #fff;
    padding: 15px 15px 5px;
    margin: 5px;
    /*float: right;*/
    text-align: center;
    border-radius: 3px;
    font-size: 0.8em;
    color: #888585;
    box-shadow: 2px 2px 5px 0px hsl(53, 2%, 78%);
}
body#home .main-area,
body#gallery .main-area {
    background-color: #e0e0e0;
}
```

## Masonry Gallery

By taking away the float, our figure elements stack up one on top of the other, like paragraph tags. The figure element is by default a block level element. Block means it goes all the way across it's parent, which is `<main>`, before another element is allowed to show below it. We can use this block level propensity to our advantage as we build out our gallery.

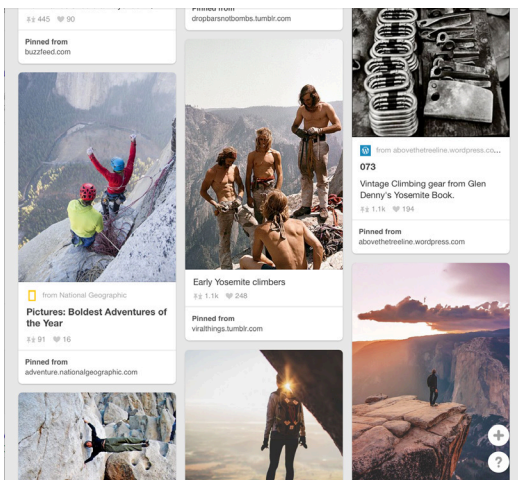
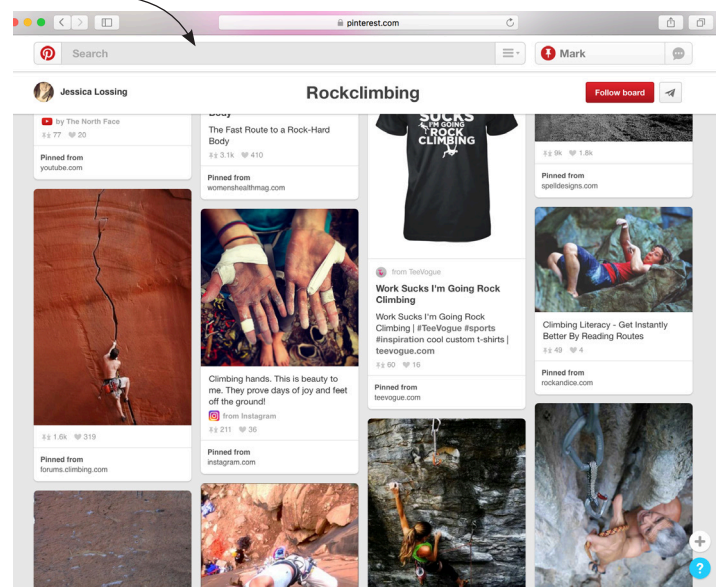
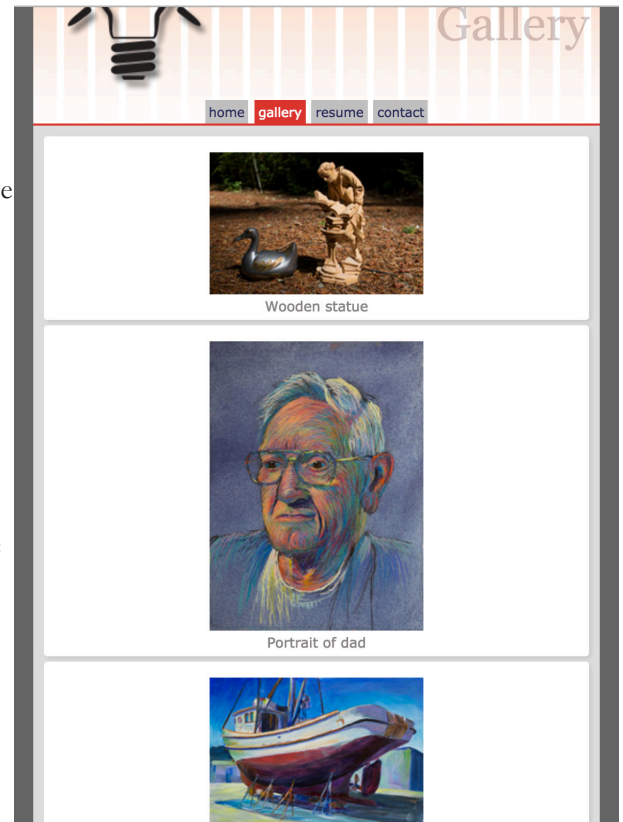
## Pinterest

Pinterest was one of the first large websites to feature a masonry style layout. Masonry refers to the lost art of building rock walls with random sized stones. Masonry as it applies to web design means arranging a page full of similar width "cards" in columns.

Notice how this pinterest page is made from a series of white round corner boxes on a light gray background. Each box has a subtle dropshadow that makes it appear to lift up off the page, like a thick card. Each box has a picture in the top and a caption in the bottom, and each box is clickable. In web design, this is called an "action box". If I shrink the browser window pinterest changes it from 4 columns of boxes to three columns. If I go there with a smartphone in portrait mode, I only see two columns. This is called responsive design. The webpage "responds" to the width of the browser window (viewport).

Have you ever noticed how newspaper stories have narrow columns of text arranged in columns? They do that to make

it easy for the eye muscles which have to flick back and forth from the end of the first line to the beginning of the next line. When you get to the end of the column, you look up to the top of the next column and start reading downward again. We can replicate that newspaper column user experience on our gallery.html page.



## Road map

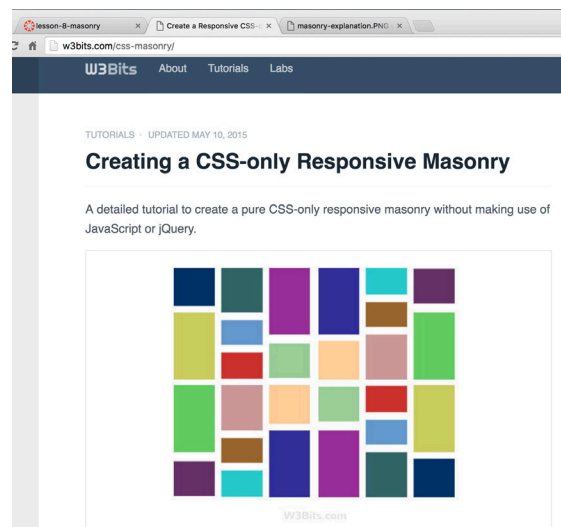
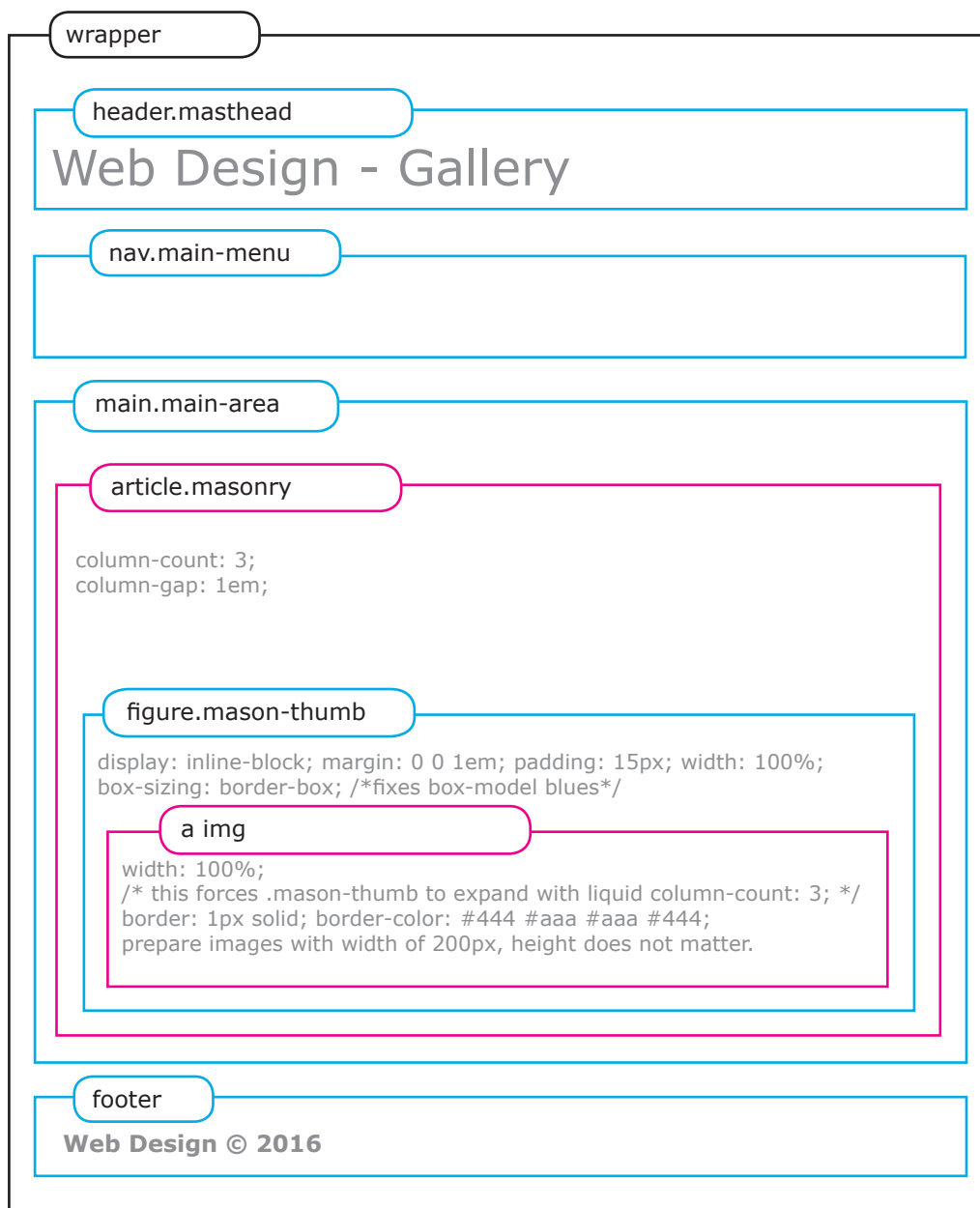
HTML5 has a new css property called: column-count. Column-count allows you to wrap a long story across 2 or more columns. But even better, we can wrap our figure thumbs through the columns.. We can have the columns and the figure elements wrapping in the columns be fully responsive to the viewport width (the browser width).

Having the masonry be responsive means we have to carefully calibrate the pixel width of the columns to avoid having the thumbnails be either too big or too small. We can do that using some dynamic code that allows the browser to change the number of columns based on the viewport width.

The illustration to the right shows the CSS box model structure we will use to get our masonry to function. We will build this up slowly, solving problems as they occur.

As a part of the web design community, I like to give credit where credit is due. I learned much of this masonry technique from this awesome website:

<http://w3bits.com/css-masonry/>



## Column count

**STEP ONE:** On the `gallery.html` page, wrap your existing figure elements in a starting and stopping `article` tag. Give the article tag a class of `masonry`. An article element is used to describe or delineate an area of a webpage that is potentially a stand alone object as far as the information it contains. Meaning it could be pulled out into an RSS feed and it would make sense on it's own. This article tag will later be told to be the main column in a two column layout. We will make two and three column layouts further on in this book.

**STEP TWO:** edit the `figure.mason-thumb` rule as shown.

**STEP THREE:** Add a new rule below it that speaks to the `<article class="masonry">` element. Remember that wraps around all our `figure` tags.

Save your page and check it in the browser. Note that we now have columns that cause the figure elements to wrap from bottom to top. But we need to get control of that image inside the figure element. Because it has not been told how wide to be, it is misbehaving.

**NOTE:** in some browsers (Firefox) you may have problems with `column-count`. Because it is a relatively new property you may need to add **Vendor Prefixes** that speak specifically to those older browsers. Think of Vendor Prefixes as the instruction manual on a new device. The instructions are in English, but they are also in French and Spanish for users who only know those languages.

**STEP FOUR:** To get vendor prefixes, google: "css3 vendor prefixes", google: "css3 vendor prefixes" or try this website: <http://pleeease.io/play>

You type in the top, and it adds prefixes to in the bottom window.

gallery.html

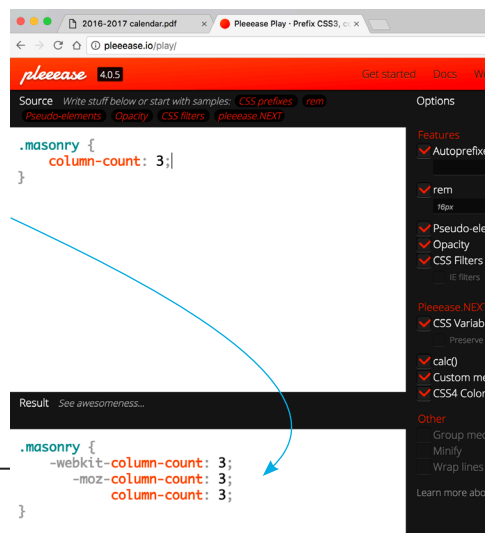
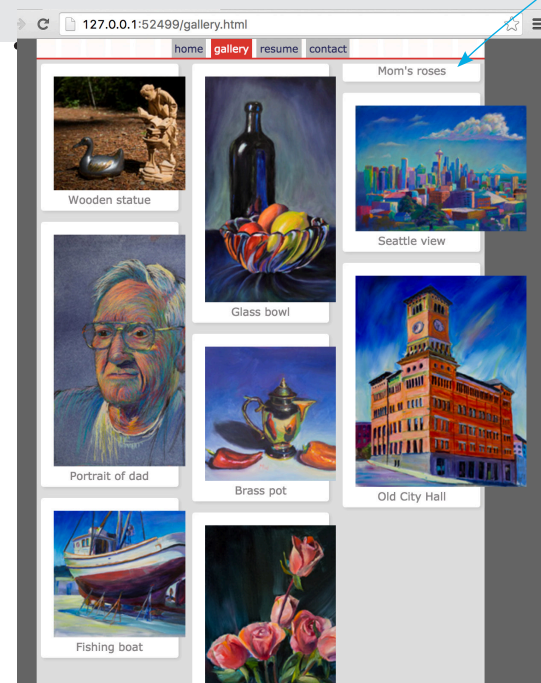
```
<main class="main-area">
<article class="masonry">
  <figure class="mason-thumb">...Do not type these figure
  elments, they are already done...</figure>
  <figure class="mason-thumb">....</figure>
  <figure class="mason-thumb">....</figure>
  <figure class="mason-thumb">....</figure>
</article>
</main>
```

style.css

```
figure.mason-thumb {
  background-color: #fff;
  padding: 15px 15px 5px;
  margin: 6px 0 1em; /*old: margin: 0 0 1em*/
  width: 100%;
  display: inline-block;
  box-sizing: border-box; /*box model fix*/
  text-align: center;
  border-radius: 3px;
  font-size: 0.8em;
  color: #888585;
  box-shadow: 2px 2px 5px 0px hsl(53, 2%, 78%);
}
```

```
.masonry {
  -webkit-column-count: 3;
  -moz-column-count: 3;
  column-count: 3;
}
```

Wrapping  
figcaption, fix is  
on next page





## Masonry concluded

**STEP ONE:** Add a new rule that tells the images inside the figure tag to be 100% in width. Rather than making them display at their native 200 pixel width, 100% makes them fill up all the available space within their parent. And their parent is the figure element, which is responding to the column count command.

Check it in your browser. You should have a good looking masonry effect. The figure elements wrap from the bottom of the first column to the top of the next column.

I also have it pictured here in my iPhone simulator. This is a free web developer tool that is built into the Mac operating system and is part of the Xcode app. Both Mac and Windows computers have something similar for Android testing.

Resize the browser window and notice how the smaller viewport shrinks the figure tags too far. This is not ideal. If you remember, our thumbnail images have a native pixel width of 200 pixels. It is best to stay within about 30% of the native width of the thumbnails. If you let the browser shrink or expand them more than that, you will lose quality.

We need to write some code that dynamically adjusts the number of columns based on the width of the viewport. This new code is called Media Queries.

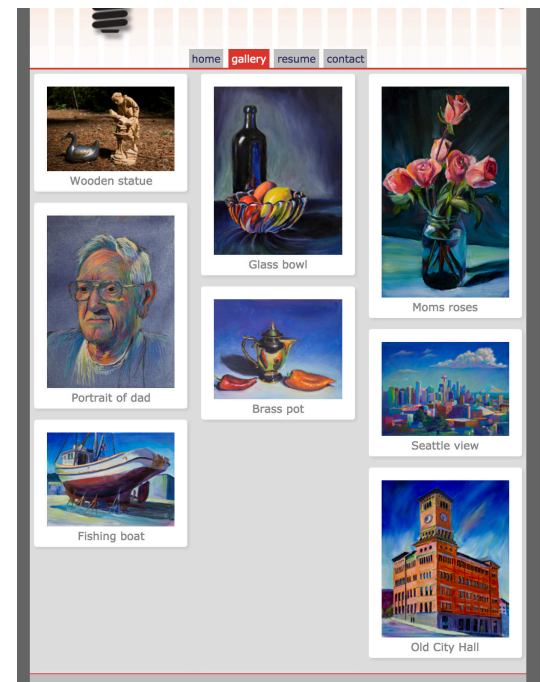
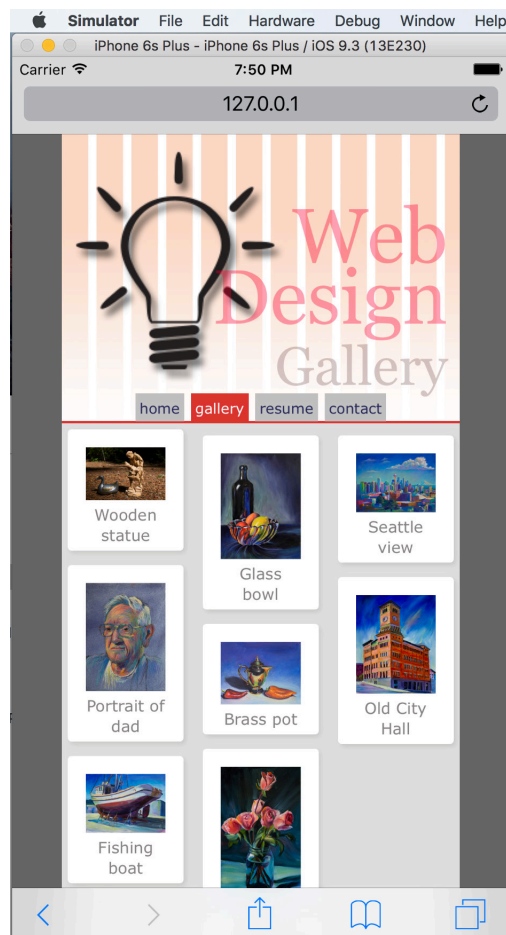
If you have trouble with a **figcaption** detaching itself from the bottom of a picture and wrapping to a new

style.css

```
.masonry {
  -webkit-column-count: 3;
  -moz-column-count: 3;
  column-count: 3;
}

figure.mason-thumb a img {
  width: 100%;
}

body#home .main-area,
body#gallery .main-area {
  background-color: #e0e0e0;
}
```

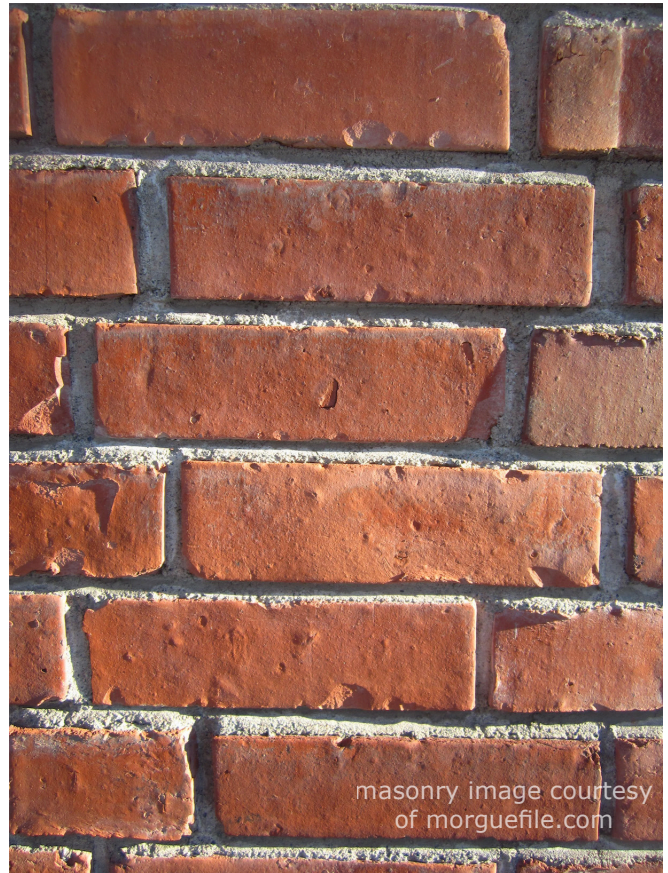


column, check your **.mason-thumb** rule to make sure everything is correct, especially the spelling on **display: inline-block**; Also check the spelling of everything in the **<head></head>** area of your page. Compare it to page 38 in this book. If none of that fixes the figcaption wrap problem, copy the files to your desktop or the remote elmo server and view the webpage there. Viewing a page that relies on inline-block over a local network, such as the campus room107 network can be enough to cause the problem. It does not make sense, but we've seen it happen.

## Media Queries for Masonry

We need a way to measure the browser window. On a Windows machine, there is a free utility called screen ruler. You can google it, or find it in the resources folder for this class. Measure the screen and determine the range of widths at which your thumbnails look good in 3 columns. Do the same measuring and testing with the column count set at 2 columns. I'm going to give you the numbers, but FYI, I used a screen ruler to come up with the right numbers. Those numbers can change if you choose to make your thumbnails larger or smaller in Photoshop. For example, at websterart.com, I chose to make my thumbnails in Photoshop a little larger at 250pixels wide, so my media query column count numbers look different.

Media Queries are a way for the browser to apply different style sheet rules based on how wide the browser (viewport) is at any given time. It doesn't matter if your end user resizes the browser window, or if they simply arrive at the website with the browser window maximized. On smart phones and tablets, the browser window is always maximized. However, even on those devices, the viewport width changes when they rotate the phone from portrait to landscape orientation. But media queries are always watching and ready to make changes to the styles if the viewport width causes problems.



Because media queries need to be able to overwrite style sheet rules that come earlier, they must always be at the very bottom of the style sheet rules.

**STEP TWO:** Comment out or delete the `.masonry {column-count: 3;}` rule. We will bring that back in our calibrated media queries.



## Adjustable column count

**STEP ONE:** At the very bottom of your style.css file, type the new rules shown to the right in brown.

Note how each one starts with an @ sign. Also notice how you have a style sheet rule complete with starting and stopping curly braces inside another style sheet rule, which has its own braces. **You need to add Vendor Prefixes**, I did not show them here in order to keep the code shorter.

This is the nature of media queries. They are conditional statements that only function if their condition is met.

Picture this conditional situation in real life:

```
if (it's raining)
{we watch tv}
else if (it's snowing)
{we go sledding}
else if (sun is out)
{we go to beach}
```

In these media query statements, we are telling the browser how many columns to use based on **if** the viewport is wide, medium, or small.

Pictured is the free:

<http://www.arulerforwindows.com/>



style.css

```
figure.mason-thumb a img {
    width: 100%;
}
body#home .main-area,
body#gallery .main-area {
    background-color: #e0e0e0;
}
@media only screen and (min-width: 1021px) and (max-width: 2500px){
    .masonry {
        column-count: 4;
        -webkit-column-count: 4;
        -moz-column-count: 4;
    }
}
@media only screen and (min-width: 730px) and (max-width: 1020px){
    .masonry {
        column-count: 3;
    }
}
@media only screen and (min-width: 390px) and (max-width: 729px){
    .masonry {
        column-count: 2;
    }
}
```

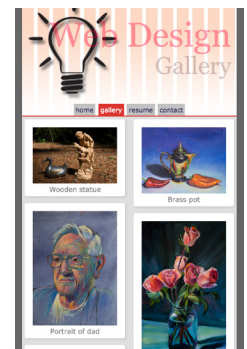
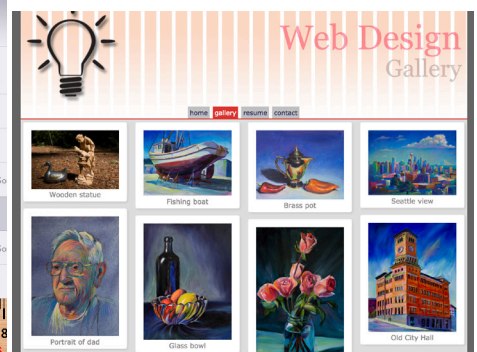
[index.html](#)

**STEP TWO:** Check your webpage at various viewport widths. Observe how your pictures behave. Feel free to tinker with my min-width and max-width numbers.

If you have trouble getting column-count to work in certain browsers, but not others, you may need to add vendor prefixes, see page 72 or: <http://pleeease.io/play>

In the 3 screenshots shown here you can see the the media queries changing the columns from 4 down to 3, and then to 2 on the most narrow viewport.

**STEP THREE:** If you think this page might be viewed on monitors larger than 2500px, you will need another @media rule defining the column count for that viewport width.





# Animated Gallery

## Animate your webpage using JQuery and Slimbox

Animated image transitions are the latest trend on gallery webpages. A typical example is a webpage that opens with a large **hero image** in the top of the webpage. If it is animated, it will fade or slide into another image. There may be captions and numbers or dots telling you which image you are viewing in the series. They can be quite complicated, especially if you want to start with thumbnails, and then go to the big images after a thumb is clicked. Later you may learn how to do all that from scratch in Javascript, but for now...

### Slimbox to the rescue!

Slimbox was developed for people who want to have an animated slideshow on their gallery page, but don't want to learn JavaScript or Flash. It allows you to click on a thumbnail and have your large images popup in a black animated box. The box has buttons for moving forward and backward between large images. It works similar to the facebook gallery function, but better.

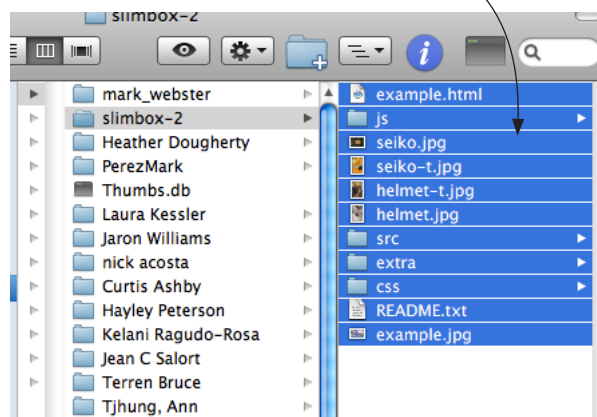
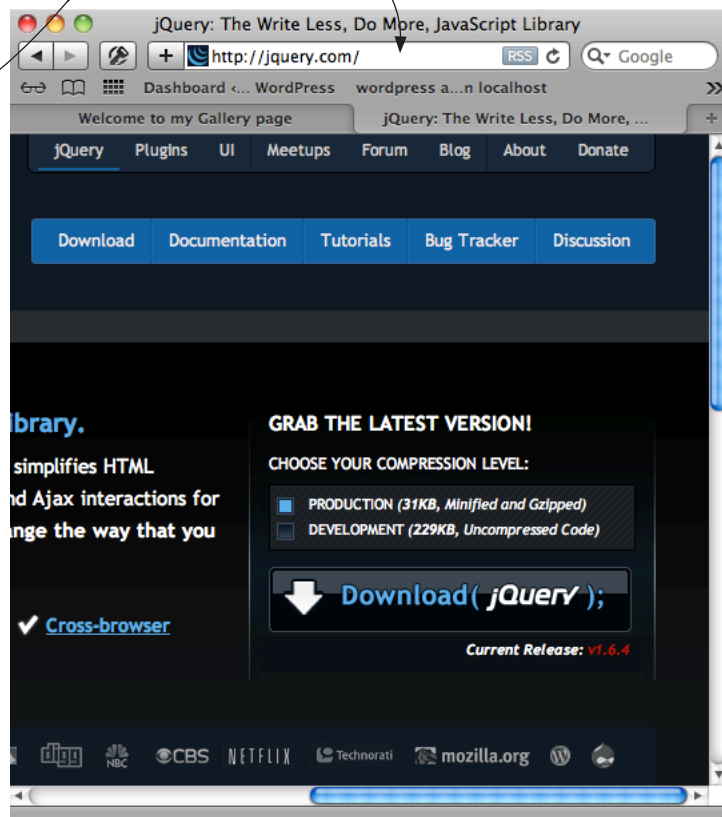
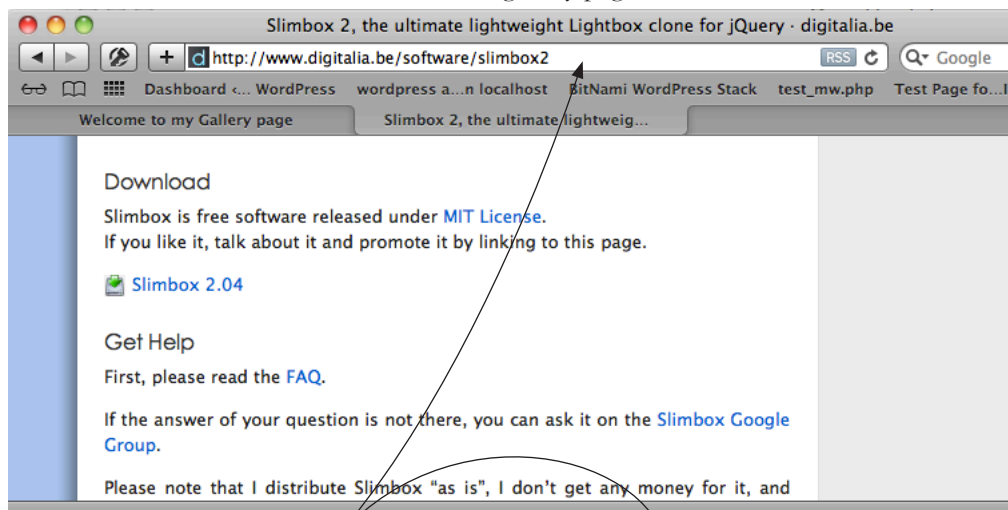
I've provided, and prepared all the files for you on the school server.

**NOTE:** if you'd like to download all the files from scratch, here are the two websites where you can find the slimbox files. If you are getting them from scratch, you need to download the latest Slimbox zip file, and the latest JQuery file from these two websites:

<http://www.digitalia.be/software/slimbox2>

<http://jquery.com/>

**STEP ONE:** To get my prepared slimbox files, go to the resources folder and copy the contents of the "**slimbox**" folder.



## Paste the files

**STEP ONE:** Navigate to your current working folder and paste in the files you copied from the slimbox folder

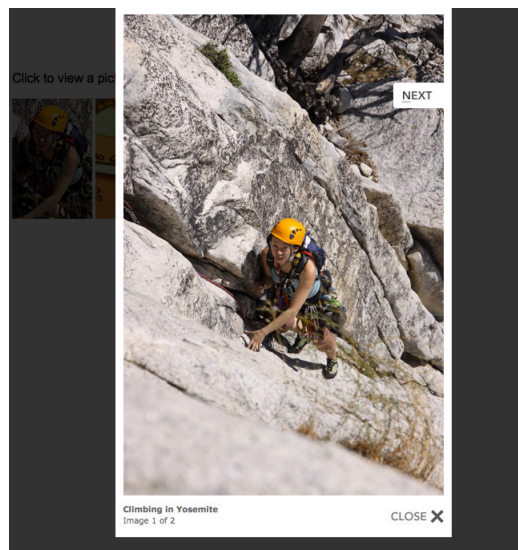
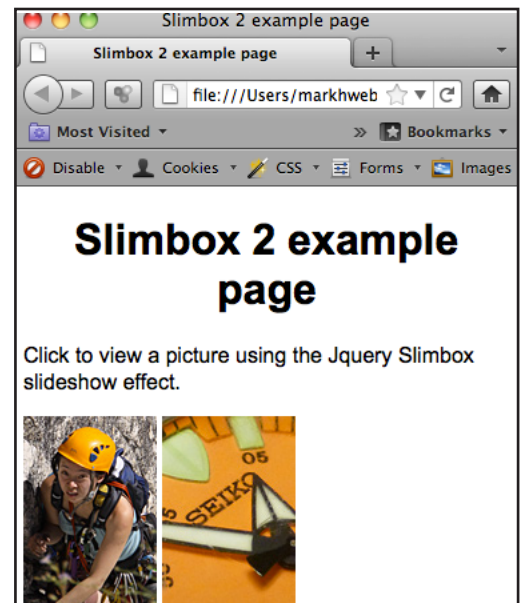
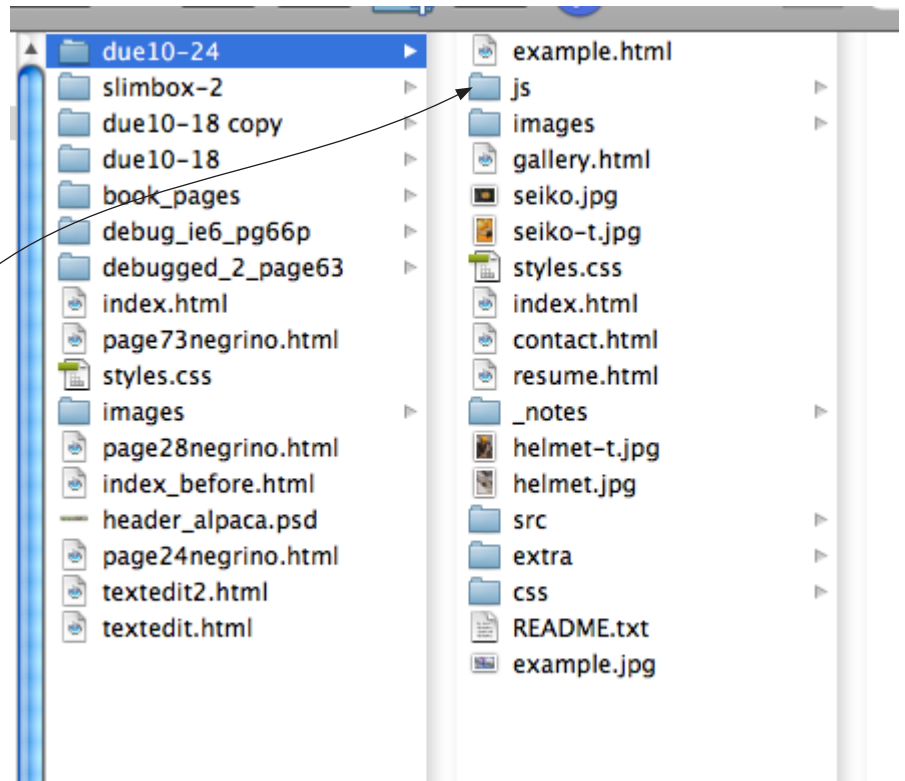
**STEP TWO:** Here I have pasted in the slimbox-2 files into my current homework folder. Note that the slimbox folders (js, src, extra, css) are at the same level as our **images** folder, and our **index.html** page. This is called the "root" of our website. Your "lesson-xx" folder is the "root" folder on your local website. Up on the remote server, your jsmith (first initial last name) folder is considered to be your "root" folder.

**STEP THREE:** From within your "root" folder, open the **example.html** file in your web browser of choice.

In the browser, you should see two thumbnail images. Click either one and you will see the thumbnail blossom up to the full size image.

The background will turn transparent black. There will be a caption at the bottom describing the image, and there will be either a back or next button on the left or right side of the image. All these special effects are created using a pre-built library of javascript called JQuery. Many websites use this new gallery slideshow effect, including Facebook.

**STEP FOUR:** From within your "root" folder, open the **example.html** file in notepad. You may also use Dreamweaver, in the all code view, but if you use Dreamweaver, make sure you have defined the site (site>manage sites) to point at the current "root" folder.





## How it works:

**STEP ONE:** This is the code from the `example.html` file. Notice the two lines of code under the title tag. Both lines feature a starting and stopping script tag. `<script></script>`. Note the `src=""` properties in both lines of code. For example, in the top line the `src` property is `"js/jquery.js"`. This line of code tells the browser to go into the `"js"` folder and import the javascript file named `"jquery.js"`.

You have to maintain the existing folder structure in your current `"root"` folder. The browser needs all those folders to be exactly where they currently are for the slimbox jquery slideshow effects to function. When you push your website up to the remote server, simply grab everything in your current root folder as you always do, and the slideshow animation will work on the remote `http://168` server just as it does now locally.

The third line of code:

`<link rel="stylesheet" href="css/slimbox2.css" type="text/css" media="screen">`

imports a special stylesheet that helps the slimbox slideshow function, and controls its appearance. Use caution editing this external stylesheet. Keep a backup and make small changes, testing in all the browsers. Personally, I never touch it.

**NOTE:** all three of these lines of code in the head of this `example.html` page:

```
<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/slimbox2.js"></script>
<link rel="stylesheet" href="css/slimbox2.css" type="text/css" media="screen">
```

have to be included in any webpage where you would like to have an animated slideshow.

**STEP TWO:** Examine the thumbnail image code. Note that there is a starting and stopping anchor tag around each image `src` tag:

```
<a href=""><img src=""></a>
```

The `href` property tells the browser to navigate to the larger version of the image if they click the thumbnail image. You have already done that in your gallery page, a thumbnail click simply opens the larger image, and you have to click the back button on the browser toolbar to get back to the thumbnails. This is not convenient for your users, though, the slimbox effect is much more intuitive, and fun.

**STEP THREE:** Note the two additional properties in the starting `<a>` tag:  
`rel="lightbox[recent]" title="Climbing in Yosemite"`

The value you enter into the **title** property (Climbing in Yosemite) becomes the caption under the image.

This property: `rel="lightbox[recent]"` is the code that connects each thumbnail with the hidden, imported jquery javascript. Each thumbnail image **must** have that `rel="lightbox[recent]"` property for it to be part of the animated slimbox slideshow.

```
<!DOCTYPE html>
<html>
<head>
  <title>Slimbox 2 example page</title>
  <script type="text/javascript" src="js/jquery.js"></script>
  <script type="text/javascript" src="js/slimbox2.js"></script>
  <link rel="stylesheet" href="css/slimbox2.css" type="text/css" media="screen" />
  <style type="text/css">
    body {
      background-color: #fff;
      font-family: arial, helvetica, sans-serif;
      color: #000;
    }
    h1 {
      text-align: center;
    }
    a {
      font-weight: bold;
      color: #f00;
    }
    img {
      border: none;
    }
  </style>
</head>
<body>
  <h1>Slimbox 2 example page</h1>
  <p>Click to view a picture using the JQuery Slimbox slideshow effect.</p>

  <a href="helmet.jpg" rel="lightbox[recent]" title="Climbing in Yosemite">
    
  </a>

  <a href="seiko.jpg" rel="lightbox[recent]" title="Seiko self winding watch">
    
  </a>
</body>
</html>
```

# Animating your gallery.html page with JQuery

**STEP ONE:** copy the three lines of code from the top of the example.html page. These are the 3 lines you need to copy:

```
<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/slimbox2.js"></script>
<link rel="stylesheet" href="css/slimbox2.css" type="text/css" media="screen" />
```

**STEP TWO:** Open your gallery.html page and paste the 3 lines of code on a new line below the stopping title tag like this:

```
<title> gallery </title>
```

```
<script type="text/javascript" src="js/jquery.js"></script>
<script type="text/javascript" src="js/slimbox2.js"></script>
<link rel="stylesheet" href="css/slimbox2.css" type="text/css" media="screen">
```

**STEP THREE:** Save the changes! NOTE: the space - forward slash at the end of some code lines ( /> ) is not needed. It is part of the old XHTML language, which is deprecated.

**STEP FOUR:** Return to the **example.html** page. Locate the starting anchor tag code for one of the thumbnail images.

**STEP FIVE:** copy these two properties from the starting anchor tag:

```
rel="lightbox[recent]" title="Climbing in Yosemite"
```

**STEP SIX:** switch to your **gallery.html** page and locate your first anchor tag that surrounds a thumbnail image. Here is the code you are looking for:

```
<main class="main-area">
  <figure class="mason-thumb">
    <a href="images/statue.jpg">
      
    </a>
    <figcaption>
      Wooden statue
    </figcaption>
  </figure>
```

Note that the starting anchor <a> tag has only the one property: **href**

**STEP SEVEN:** Click with your cursor just to the left of the greater than sign at the end of the starting anchor tag. Press the spacebar once and then paste. Here is how it should look:

```
<figure class="mason-thumb">
  <a href="images/statue.jpg" rel="lightbox[recent]" title="Wooden statue carved in Germany">
    
  </a>
  <figcaption>

    Wooden statue
  </figcaption>
</figure>
```

**STEP EIGHT:** Paste that code into every starting anchor tag on your thumbnails. You will of course need to edit the title value to describe each image individually.

# Testing and trouble shooting your slimbox slideshow

**STEP ONE:** Test your gallery.html page in the browser. There should now be animations each time you click a thumbnail. Once you do click one, the slimbox animation effect should allow you to navigate forward or backward through the large versions of the thumbnails. There should be captions under each large image. If you want to escape back to the thumbnails, you can click the black color outside the large image, or click the close button. NOTE: to make the pictures animate in correct order,

```
.thumb {float: left;}
```

**TROUBLE SHOOTING:** if it doesn't work, review these instructions, and check your spelling. Typical mistakes are not maintaining the critical folder structure that lets the browser find the imported javascript and CSS files required for the animation effects.

Other errors are not putting your images in the correct location specified in the href property. We keep our images in an images folder, not at the root of the defined website.

Another common error is to make your images too large. The larger version of the images should be no higher than 630 pixels, give or take. If you make them bigger than that, slimbox will work, but all the controls and the edges of the images are off screen.

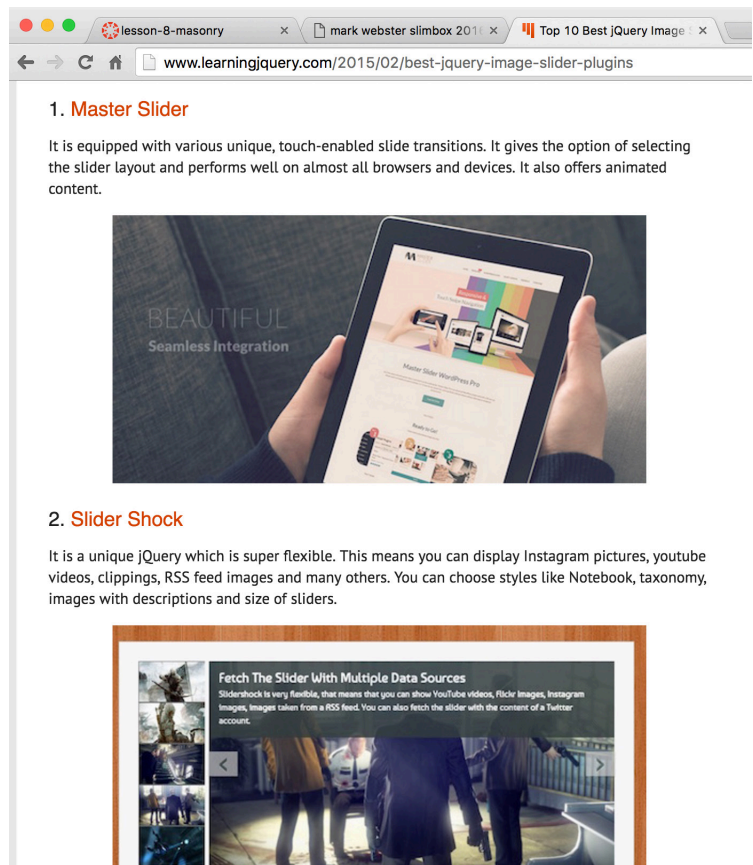
Review page 48 where I explain preparing images for the web in Photoshop.

## Further embellishments:

There are many other free Javascript sliders out there. Do a google search for:

"best free jquery sliders of 2016". They all function somewhat similarly and most of the good ones come with instructions and help pages.

NOTE: While this slimbox slideshow is the simplest one to start with, it does not animate on smartphones. It still shows the big image, but it has a line of code that detects your smartphone and disables the javascript. We call this "degrading gracefully". Later in this book we will do two other variations on gallery sliders. If you'd like to look ahead, search this pdf for either photoswipe, or flexslider. Be aware though that they are much more complicated and are best left until you have more programming experience.

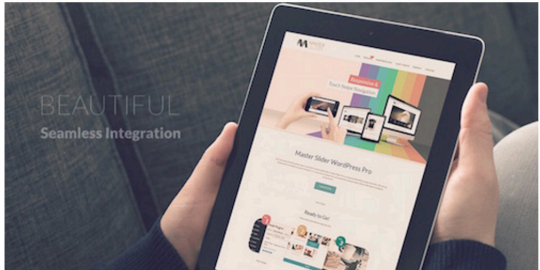


lesson-8-masonry x mark webster slimbox 201 x Top 10 Best jQuery Image x

www.learningjquery.com/2015/02/best-jquery-image-slider-plugins


### 1. Master Slider

It is equipped with various unique, touch-enabled slide transitions. It gives the option of selecting the slider layout and performs well on almost all browsers and devices. It also offers animated content.



### 2. Slider Shock

It is a unique jQuery which is super flexible. This means you can display Instagram pictures, youtube videos, clippings, RSS feed images and many others. You can choose styles like Notebook, taxonomy, images with descriptions and size of sliders.



## Create a contact.html page

**STEP ONE:** Open your **index.html** page in Notepad.

Choose **file > save as**. Change the name to **contact.html**

Drop the **Save As Type** drop list, and choose **All Files**.

This allows you to force the extension to be html

**STEP TWO:** On your newly created **contact.html**, change the title, body id and h1 tags to read **contact**.



**STEP THREE:** on body **id="contact"**, it must be a lowercase "c". The other contact words are not case sensitive.

**STEP FOUR:** clear out the interior of your **<main>** element, but leave a simple paragraph with placeholder text as shown. We will start from scratch on this one.



```

    . . . . .
    . . . . . contact.html . . . . .
    . . . . .
    <!DOCTYPE HTML>
    <!doctype html>
    <html lang="en">
    .
    <head>
    .
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="IE=edge" >
    <title>
    . Web Design - Contact
    </title>
    <link rel="stylesheet" type="text/css" media="screen" href="style.css">
    </head>
    .
    <body id="contact">
    .
    <div id="wrapper">
    .   <header class="masthead">
    .   <h1>Web Design<br><span>Contact</span></h1>
    .   </header>
    .
    .   <nav class="main-menu">
    .   <ul>
    .   <li><a href="index.html" class="home">home</a></li>
    .   <li><a href="gallery.html" class="gallery">gallery</a></li>
    .   <li><a href="resume.html" class="resume">resume</a></li>
    .   <li><a href="contact.html" class="contact">contact</a></li>
    .   </ul>
    . </nav>
    .
    <main class="main-area">
    .
    .   <p>
    .   . A few placeholder words go here.
    .   </p>
    .
    </main>
    .
    <footer class="footer-area">
    .   <p>Copyright 2016 John Smith</p>
    . </footer>
    </div> <!-- end wrapper div -->
    .
    </body>
    .
    </html>
    . . . . .
  
```

## Contact info, and spam bots

Contact pages are tricky. If you put your info out there on the internet it's too easy to get calls and spam email from people you don't want to talk to. I'll show you the dangerous way first, then a few tricks to intercept the spammers.

**STEP ONE:** If you have a cute image, drop that into the main element for decoration. I bought this tincan image at a stockphoto agency for \$2.00:  
<http://www.dreamstime.com>

I used our `class="center"` style sheet rule to center it on the page.

**STEP TWO:** I added a set of paragraph tags `<p></p>`, and added an inline style sheet rule that told the text inside the paragraph to center.

`<p style="text-align: center;">`

**STEP THREE:** Type your email address, and then surround it with starting and stopping anchor tags:

`<a>john.smith@yahoo.com</a>`

**STEP FOUR:** In the starting `<a>` tag, add this property:  
`href="mailto:john.smith@yahoo.com"`

**EXPLANATION:** the `mailto:` is the key here. Normally an anchor tag `<a>` tells the browser to navigate to a webpage. By using `href="mailto:john.smith@yahoo.com"` you are telling the browser to launch the resident email software on whatever device is accessing the webpage.

**CAUTION:** you are also handing your email address to the spam harvester

contact.html

```
<main class="main-area">



<p style="text-align: center;">

    Email me at
    <a href="mailto:mark.webster@cptc.edu">
        mark.webster@cptc.edu
    </a>

    or call my Cell: 123 456 7890

</p>

</main>
```





## Use caution with contact info...

**STEP ONE:** You can leave your code as is. Many professionals have no other choice. However, here is a hack that will derail the spam bots. Edit your starting anchor tag to read like this:

```
<a href="mailto:*REMOVE*john.smith@yahoo.com">
```

Now when someone clicks the words in between your anchor tags, it will trigger their resident email software (Outlook Express). It will open with the corrupted email address you programmed above. The person trying to email you will need to be smart enough to realize they need to remove the corrupted part of the email address. If they aren't smart enough to figure that out...perhaps you don't want to hear from them anyway?

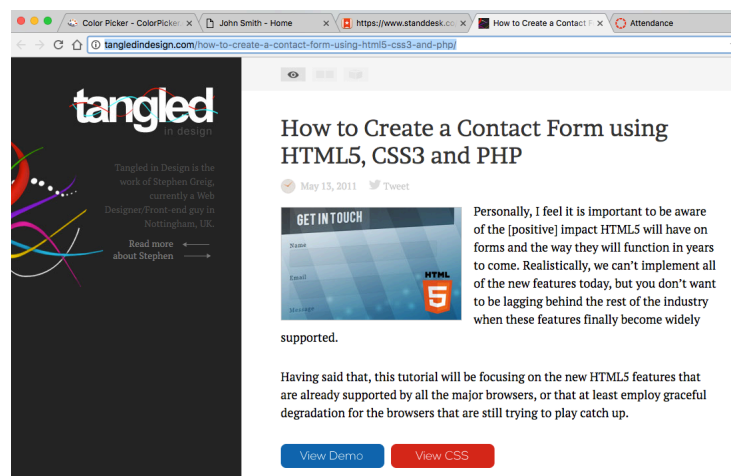
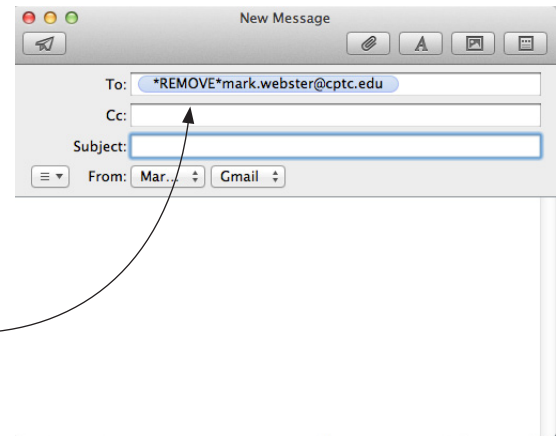
A better fix is to use a php form. Google: "php contact form tutorial". Even though your email address is there in plain sight, the spam bots look for the **mailto:** code, not the actual words in between anchor tags, so you may be safe. To be extra sure, you could make your email address an image in Photoshop, and put that between the anchor tags. You can do the same thing with your phone number, which can also get spammed. Don't put your real phone number on your website today. We are putting these up live on the internet, and unless you are looking for work today, and need it up there...caution is wise.

DO NOT put your home address on the internet! No one, not even your future boss needs to see that.

**STEP TWO:** Upload all these new change to the live server, and test everything up there, click every thumbnail and link. Watch for broken things. We will come back to the resume page later.

**NOTE:** There are some great tutorials online explaining how to make a contact form using PHP. With a PHP contact form, your email address is completely hidden from the browser. It is stored up on your remote server. The only way someone could hack your email address is if they had your login credentials for the ftp server. One of my recent college students found and implemented this tutorial. Thanks for finding this one Jessica!

<http://tangledindesign.com/how-to-create-a-contact-form-using-html5-css3-and-php/>



## Resumes as tabular data

Resumes are usually organized in columns and rows. The standard presentation is: personal information on top, followed by the "Position Desired". Next are columns with dates on the left, and details on the right. This type of presentation can be referred to as: tabular data. This means it can be "tabbed through", with the tab key, especially if you did it in Word, or Excel. If you have a lot of tabular data, it can be easier to simply insert a table for a resume in HTML using wysiwyg software like Dreamweaver.

Using tables to present our resume would be easier, but the words in your resume describing your skills are widely scattered and separated by the code that makes an HTML table work. Google can't easily see and understand the relationships between words in table data cells in different rows and columns.

Fortunately there is a way to present your information in columns and rows, without using a table. We will use something called a Definition List. A definition list is a bit like an unordered list, in that it can have an unlimited number of children. However, unlike an unordered list:

```
<ul> <li>one</li> <li>two</li> </ul>
```

each child of a definition list (dl) is broken down into two parts:

definition term (dt), and definition description (dd).

Think of a dictionary. You have your word (Definition Term: dt) and your description (Definition Description: dd).

Here is a sample of how a Definition List looks in HTML:

```
<dl>

  <dt>bird</dt>

  <dd>A winged creature that flies through the air</dd>

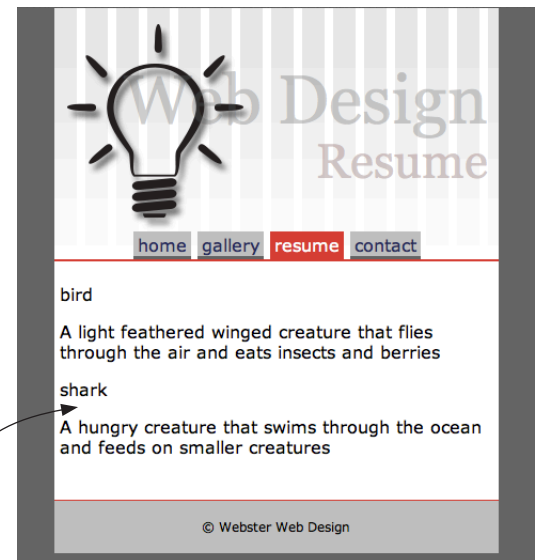
  <dt>shark</dt>

  <dd>A hungry creature that swims through the ocean...</dd>

</dl>
```

Here is how it renders on the web page.

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | processes in ordinary engines.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>EXPERIENCE:</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 1/02 to present    | Technical College Adjunct Faculty / Freelance Web Designer                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|                    | <p><b>Clover Park Technical College</b><br/>Tacoma, Washington</p> <p><b>Teaching</b><br/>Teach both beginning and intermediate levels of Flash, Photoshop, Illustrator, Dreamweaver and traditional Drawing to students in the 2 year Media Design Degree program.</p> <p>Created curriculum for classes, many of which had not been taught before (Intermediate Flash, Intermediate Illustrator, Intro to Dreamweaver, Character Animation and Basic Drawing). Motivated students of all ages to pursue excellence. Lectured on real life uses of software, going beyond the "gee whiz" effects into practical application. Students frequently remark that his <a href="#">tutorial handouts</a> are written far better than the classroom textbooks. Coordinated with other instructors to assure continuity between classes.</p> <p><b>Freelance Web Design</b><br/>Design, develop and maintain web sites for clients. Work includes making cold calls, writing contract bids, interface design, XHTML, JavaScript, tutoring and consulting, recording client interviews, transcribing and copy writing. Responsible for recruiting and supervising teams of paid student interns.</p> <p><i>Client list:</i></p> <ul style="list-style-type: none"> <li>• <a href="#">Rainier Communications Commission</a> - designed, developed, trained client in site maintenance</li> <li>• <a href="#">Creekside Chiropractic &amp; Massage</a> - designed, developed, maintenance</li> <li>• <a href="#">cptc.edu</a> - designed, developed, maintenance 9-02 to 8-04 (220 pages)</li> <li>• <a href="#">Dr. Gregg Fisher DDS</a> - designed, developed, maintenance</li> <li>• <a href="#">Smith &amp; Margol</a> - partial design, maintenance (all Flash)</li> </ul> |
| 12/00 to 1/02      | Web Designer                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|                    | <p><b>Artifex Business Identity Development</b><br/>Tacoma, Washington</p> <p>Designed interfaces in Photoshop, Imageready and Flash. Brought images into Dreamweaver and wrote rock solid HTML, CSS and Javascript that enabled correct display in all browsers, versions 4.7 or higher and all operating systems including Macintosh.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |



## New resume page

**STEP ONE:** Open your **index.html** page again in Notepad. Choose **file > save as**. Change the name to **resume.html**. Drop the **Save As Type** drop list, and choose **All Files**. This allows you to force the extension to be html

**STEP TWO:** On your newly created **resume.html**, change the title, body id and h1 tags to read **resume**.

**STEP THREE:** on body **id="resume"**, it must be a lowercase "r". The other resume words are not case sensitive.

**STEP FOUR:** clear out the interior of your **<main>** element. We will start from scratch on one.

**STEP FIVE:** If you'd like to have a copyright symbol by your name, use **&copy;**;

You can find more symbols by googling: **ascii characters html**

```

resume.html

<!DOCTYPE HTML>
<!doctype html>
<html lang="en">

  <head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="IE=edge" >
    <title>
      Web Design - Resume
    </title>
    <link rel="stylesheet" type="text/css" media="screen" href="style.css">
  </head>

  <body id="resume">

    <div id="wrapper">
      <header class="masthead">
        <h1>Web Design<br><span>Resume</span></h1>
      </header>

      <nav class="main-menu">
        <ul>
          <li><a href="index.html" class="home">home</a></li>
          <li><a href="gallery.html" class="gallery">gallery</a></li>
          <li><a href="resume.html" class="resume">resume</a></li>
          <li><a href="contact.html" class="contact">contact</a></li>
        </ul>
      </nav>

      <main class="main-area">

      </main>

      <footer class="footer-area">
        <p>&copy; Webster Web Design</p>
      </footer>
    </div> <!-- end wrapper div -->

  </body>

</html>

```

## Definition List

**STEP ONE:** type the code shown to the right in bold brown.

**STEP TWO:** Observe how the Definition List looks in the browser. Note how the `<dd>` numbers (2 & 4) are stepped over to the right. This is the default browser display of a Definition List. We can modify that with a style sheet.

**STEP THREE:** Add the code shown in bold brown below. Note that I've added descriptive classes to each starting `<dl><dt><dd>` tag. I named the classes: `defList`, `defTerm` and `defDescription`.

resume.html

```
<main class="main-area">

<dl>

    <dt> 1 </dt>

    <dd> 2 </dd>

</dl>

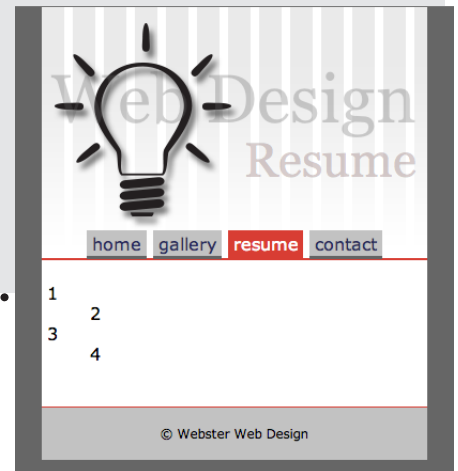
<dl>

    <dt> 3 </dt>

    <dd> 4 </dd>

</dl>

</main>
```



resume.html

```
<main class="main-area">

<dl class="defList">

    <dt class="defTerm">          1 </dt>

    <dd class="defDescription">   2 </dd>

</dl>

<dl class="defList">

    <dt class="defTerm">          3 </dt>

    <dd class="defDescription">   4 </dd>

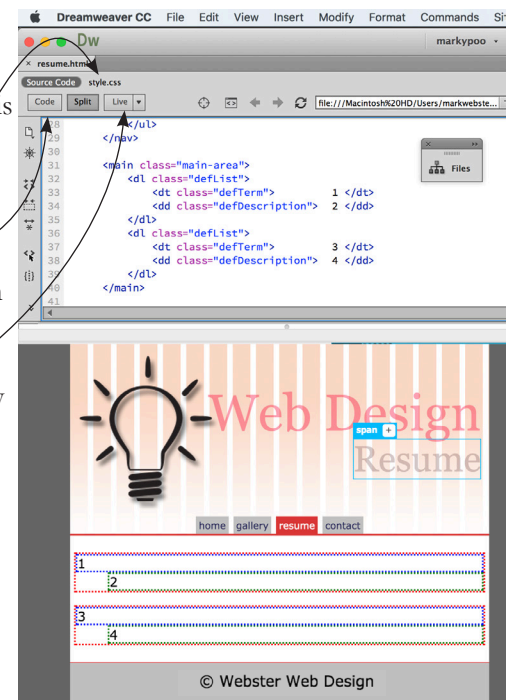
</dl>
```

## Definition List styles

The next few screenshots are in Dreamweaver. It has some useful features, but it is by no means the only code editor you should practice with.

**STEP ONE:** Open your **style.css**. There is a short cut button at the top of Dreamweaver if you have the resume.html page open.

**STEP TWO:** Anytime you are working directly on a CSS file, click the **Code** button so the split view disappears and you can see all of the CSS code. Also, when you are editing html as is shown in this screenshot you can switch between **live** and **design** views to get closer to how the real browser will display your code. I usually leave it in **design** view.



**STEP THREE:** add the following code (shown in bold brown) **above** your first media query rule.

**STEP FOUR:** It should look like this in the browser:

In an ideal world we could simply move the green boxes over to the right, and up, and we'd have rows and columns. Because of the differences in how browsers interpret html tags, we will have to get very specific in our code to recreate a table, using Definition List.



```

style.css
figure.mason-thumb a img {
    width: 100%;
}
/*
.masonry {
    column-count: 3;
}
*/
body#home .main-area,
body#gallery .main-area {
    background-color: #e0e0e0;
}
.clear-float {
    clear: both;
    height: 0;
    font-size: 0.2;
}
.defList {
    border: 2px dotted red;
}
.defTerm {
    border: 2px dotted blue;
}
.defDescription {
    border: 2px dotted green;
}
@media only screen and (min-width: 1021px) and
(max-width: 1500px){
    .masonry {
        column-count: 4;
    }
}

```



## Hanging Classes on HTML tags

**STEP ONE:** In Dreamweaver, click the short cut link back to Source Code. If this confuses you, remember that there are two files available to work on when Dreamweaver opens resume.html:

- (1.) resume.html
- (2.) style.css

You can switch back and forth between the two files by clicking the shortcut links immediately above the three buttons called Code, Split & Design

**STEP TWO:** working in code view for resume.html, copy the code for the second definition list, the one that has 3 and 4. Copy from <dl> start to </dl> stop.

**STEP THREE:** paste it in below the closing </dl> tag and change the numbers to 5 & 6.

**STEP FOUR:** In the class property add an additional class of **defListLast**. There has to be a space between the two classes: **defList defListLast**

See the code above in bold brown.

When you list two style sheet rules in one class like this, the browser uses the properties of both classes. I had to make the last Definition List behave differently, as you will see on the next page.

**STEP FIVE:** Add HTML comment tags on each of your <dt> and <dd>tags. If it is a <dt></dt>, it will be a left column.

If it is a <dd></dd>, it will be a right column. This will help you stay organized as you begin to add content to your resume.

resume.html

```

<main class="main-area">
<dl class="defList">
    <dt class="defTerm">          1 </dt> <!--***left column***-->
    <dd class="defDescription">  2 </dd> <!--***right column***-->
</dl>
<dl class="defList">
    <dt class="defTerm">          3 </dt> <!--***left column***-->
    <dd class="defDescription">  4 </dd> <!--***right column***-->
</dl>
<dl class="defList defListLast">
    <dt class="defTerm">          5</dt><!--**left column*-->
    <dd class="defDescription">  6</dd><!--*right col*-->
</dl>

```

## Fine tuning the styles to make a faux table

**STEP ONE:** type the code you see to the right in bold brown. Note, I have completely rewritten the existing defList style sheet rules. Feel free to copy and paste. There is a lot of code here, and even I don't understand it all. I discovered this code through painful trial and error experimenting. Note that there is a border-left property on .defDescription. That is building the center column divider line.

**STEP TWO:** Find your style sheet rule for **main.main-area**, and change the padding value to **15px**

**main.main-area** {padding: 15px;}

**STEP THREE:** Save all your code, for both **style.css** and **resume.html**, then preview it in the browser. You should have this. Note that there is a missing bottom border on the "fake table"

**STEP FOUR:** Add this last little style sheet rule shown in bold brown below and check your page. The "table" should be ready to use.



style.css

```

/** there is code above this line **/

.defList {
    border-top: 1px solid #646464;
    border-left: 1px solid #646464;
    border-right: 1px solid #646464;
    /** height: 100%;**hack for ie-6-7-8 ?? **/
    margin: 0;
    padding: 0;
    min-height: 50px;
}

.defTerm {
    height: 50px;
    width: 130px;
    display: block;
    padding: 0;
    margin: 0;
    padding-left: 5px;
}

/**for taller left side, change all values of 50 to 80**/

.defDescription {
    border-left: 1px solid #646464;
    margin: -50px 0 0 130px;
    min-height: 50px;
    display: block;
    /** height: 100%;**hack for ie-6-7-8 ?? **/
    padding: 0 0 0 5px;
}

```

style.css

```

/** there is code above this line **/
.defDescription {
    border-left: 1px solid #646464;
    margin: -50px 0 0 130px;
    min-height: 50px;
    display: block;
    /**height: 100%;**hack for ie-6-7-8 ??**/
    padding: 0 0 0 5px;
}

.defListLast {
    border-bottom: 1px solid #646464;
}

```

## Adding content to cells

Now that we have a working "fake table", let's build a resume.

**STEP ONE:** View the source code for resume.html. Put the page in split view, so you can see code and design. Working down in the design view, replace the number 1 with your first and last name.

**STEP TWO:** Highlight the number 2, and replace it with your telephone number, and email address. DO NOT put your street address, never, ever put your street address on the internet. Refer back to the contact page for instructions on the email address.

**STEP THREE:** To get right alignment in the Telephone cell (2), add an inline style to the <dd> tag like this:

```
<dd class="defDescription" style="text-align: right;">
```

**STEP FOUR:** Replace cell number 3 and 4 with OBJECTIVE: & Job Title. Keep your Job Title short. It does not need to be a shopping list.

**STEP FIVE:** enclose the word objective in strong, or bold <b> tags. Search Engines understand <strong> better than <b>:

```
<strong>OBJECTIVE:</strong>
```

**STEP SIX:** Before you edit 5 and 6, copy that definition list from <dl> to </dl>. Paste it back in a couple times until you have 7, 8, 9, 10. Add more if you think you need them.

**STEP SEVEN:** If you get a double thick line in the "table", remember that only the **last definition list** should have the extra class of **defListlast**. In this case, your 9,10 row starting <dl> tag should read like this:

```
<dl class=" defList defListLast">
  <dt class="defTerm"> 9 </dt>
  <dd class="defDescription"> 10</dd>
</dl>
```

NOTE: the reason for that extra class on the last definition list is this: to avoid double borders we want our "rows" to have top borders only. We manually give the last "row" a bottom-border property with the **defListLast** style sheet rule.



## Unordered list of skills

**STEP ONE:** In cell 5, type the word **SKILLS:** and make it bold.

**STEP TWO:** In Cell 6, working entirely in the code view, add a **5 item unordered list**. All the definition lists so far are shown here, with the new unordered list in bold brown.

resume.html

```
<main class="main-area">

  <dl class="defList">
    <dt class="defTerm">Mark Webster</dt>
    <dd class="defDescription" style="text-align: right;"> Telephone: 253-123-4567<br>
      email: <a href="mailto:*REMOVE*mark.webster@cptc.edu">mark.webster@cptc.edu</a>
    </dd>
  </dl>
  <dl class="defList">
    <dt class="defTerm"><strong>OBJECTIVE:</strong> </dt>
    <dd class="defDescription"> Web Designer - Full Time or Contract</dd>
  </dl>
  <dl class="defList">
    <dt class="defTerm"><strong>SKILLS:</strong></dt>
    <dd class="defDescription">
      <!--***** cell 6 *****-->
      <ul>
        <li> a </li>
        <li> b </li>
        <li> c </li>
        <li> d </li>
        <li> e </li>
      </ul>
    </dd>
  </dl>

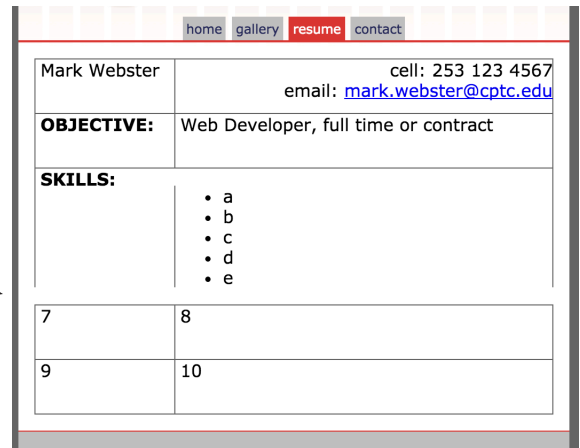
  <dl class="defList">
    <dt class="defTerm"> 7 </dt>
    <dd class="defDescription"> 8</dd>
  </dl>

  <dl class=" defList defListLast">
    <dt class="defTerm"> 9 </dt>
    <dd class="defDescription"> 10</dd>
  </dl>

</main>
```

## Fix broken resume borders

This Faux table structure is very sensitive to margins placed inside it. For example, you may find, in certain browsers, that your unordered list items blow up the table borders. There may be gaps in the lines, as shown at right. To fix this, we need to add a style sheet command that removes the default margins that come in with any unordered list: `<ul><li></li></ul>`



|                   |                                                                                                           |
|-------------------|-----------------------------------------------------------------------------------------------------------|
| Mark Webster      | cell: 253 123 4567<br>email: <a href="mailto:mark.webster@cptc.edu">mark.webster@cptc.edu</a>             |
| <b>OBJECTIVE:</b> | Web Developer, full time or contract                                                                      |
| <b>SKILLS:</b>    | <ul style="list-style-type: none"> <li>• a</li> <li>• b</li> <li>• c</li> <li>• d</li> <li>• e</li> </ul> |
| 7                 | 8                                                                                                         |
| 9                 | 10                                                                                                        |

**STEP ONE:** In style.css, type the code you see to the right in bold brown.

NOTE: This code says: if there is a `<dd>` tag with a child of `<ul>`, set the margins to zero on the top, right and bottom \*and\* make the margin 25px on the left side of the `<ul>`. I added 5px of padding to replace the default margins I stripped out.

I added padding-bottom to the list items to give them breathing room.

If these values don't work, feel free to experiment. If you change these numbers, remember to test in all the common browsers, (and browser versions) on both the Mac and PC platforms: Firefox, Chrome, Safari and IE. Also check the page on both smartphone platforms: IOS and Android

```

style.css
/** there is code above this line **/
.defListLast {
    border-bottom: 1px solid #646464;
}
dd ul {
    margin: 0 0 0 25px;
    padding: 5px;
}
dd ul li {
    padding-bottom: 8px;
}
/** there is code below this line **/
  
```



## Don't be afraid to toot your own horn

**STEP ONE:** Working directly in the code, replace the alphabetical letters with your skills. Feel free to borrow mine, they can be found here:  
<http://www.websterart.com/html/resume.php>

**STEP TWO:** Add your work experience in the rows below the Skills section. The screen shot is my old resume, feel free to borrow any of my ideas or buzz words.

NOTE: If your borders blow up, see next page for debugging tips.

|                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">home</a> <a href="#">gallery</a> <a href="#">resume</a> <a href="#">contact</a> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Mark Webster                                                                                | cell: 253 123 4567<br>email: <a href="mailto:mark.webster@cptc.edu">mark.webster@cptc.edu</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>OBJECTIVE:</b>                                                                           | Web Developer, full time or contract                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>SKILLS:</b>                                                                              | <ul style="list-style-type: none"> <li>• Excellent communication skills with an upbeat, can-do outlook</li> <li>• Excels in a team environment and is also a motivated self starter</li> <li>• Speaks fluent HTML5, CSS3, JQuery, Photoshop, Flash, Illustrator, and InDesign</li> <li>• Has created several short HD films. Self taught skills include: scriptwriting, set lighting, filming with Canon DSLR's, double system sound, and post production editing in Adobe Premier. Final project was uploaded to Vimeo.</li> <li>• Has An Outstanding eye for detail and design.</li> </ul> |
| <b>EXPERIENCE:</b>                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 1/02 current                                                                                | <a href="http://www.cptc.edu">www.cptc.edu</a> College Faculty / Freelance Web Designer                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| © Webster Web Design                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

|                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             | process or changing drop down layer based navigation links site wide using Dreamweaver's Library function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|                             | Was referred to affectionately as the "testing machine" for a meticulous eye, and was called on repeatedly to test and maintain a bug database on several applications under development. Responsible for reading "Scope of Work" documents and comparing progress of the applications (JSP and Shockwave) to specifications. Updated bug databases and created Word documents with screen shots illustrating functionality issues. Worked closely in a team environment with several JSP programmers, editing and creating the graphic user interface and html elements of a JSP application that allows time management tracking over the Internet. |
| 5-00 to 7-00                | <b>Web Design Intern</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|                             | <a href="#">Dynamics in Design</a><br>Tacoma, Washington<br>Given customer approved artist mockups of web sites. Built interfaces and wrote the HTML to turn the conceptual artwork into working web pages. Primary focus was on utilizing complex tables and Cascading Style Sheets to control and display content over multiple linked pages.                                                                                                                                                                                                                                                                                                       |
| 1975 to 12-00               | <b>Journeyman Multicolor Press Operator</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|                             | Various shops in Puget Sound, most recently <a href="#">J &amp; D Printing</a> in Tacoma. Has learned to consistently coax high quality printing from multicolor presses regardless of age, make or dampening style. Has an intuitive knowledge of a good dot, a smooth solid and a trouble free color bar. He is known for his ability to run a job right the first time, every time. For reference purposes, the Press Operator resume is <a href="#">here</a> .                                                                                                                                                                                    |
| <b>EDUCATION:</b>           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 9/99 to 9/00                | <a href="#">Web Design and Development</a><br>1 year Technical Certificate<br><a href="#">Clover Park Technical College</a><br>Tacoma, Washington<br>Grade Point Average <b>3.97</b> (while working swing!)<br>Program included a thorough grounding in the concepts, software and hardware involved in the creation of sophisticated, reliable web sites. Software studied included: Photoshop, Dreamweaver, Flash, Freehand, Powerpoint and Publisher. Other areas of study included Photography, Public Speaking, Psychology in the Workplace and Technical Math.                                                                                  |
| 1975                        | <a href="#">Offset Reprographics</a><br>1 year Technical Certificate<br>Clover Park Technical College<br>Learned the skills to enter printing industry.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <a href="#">back to top</a> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

## Do not use Design View on Resume page

If you see your resume borders blow up while you are previewing it in the browser, it is most likely because you tried to use Dreamweavers design view for editing words. Because of this definition list structure, Dreamweaver can't understand what you want when you are working in the design view and press the enter key. Try to **work exclusively in the code view**. Pay close attention to the starting and stopping `<dt> </dt>` and `<dd></dd>` tags. There can only be one set of each in each set of starting and stopping `<dl></dl>` tags. If your faux table borders blow up, look in the code view near the problem. Highlight in design view, Dreamweaver will jump to that line of code in split view.

**STEP TWO:** if needed, delete the extra sets of `<dt>&nbsp;</dt>` tags. The problem tags are shown in **bold blue** below. **Delete** them if they appear. It is just Dreamweaver being stupid.

|                   |                                                                                                           |
|-------------------|-----------------------------------------------------------------------------------------------------------|
| Mark Webster      | Telephone: 253-123-4567<br>email:                                                                         |
| <b>OBJECTIVE:</b> | Web Designer - Full Time or Contract                                                                      |
| <b>SKILLS:</b>    | <ul style="list-style-type: none"> <li>• a</li> <li>• b</li> <li>• c</li> <li>• d</li> <li>• e</li> </ul> |
| 7                 | 8                                                                                                         |
| 9                 | 10                                                                                                        |

resume.html

```

<!-- **there is code above **-->
<dl class="defList">
  <dt class="defTerm"> 7 </dt> <!--***left column***-->
  <dd class="defDescription"> 8</dd> <!--***right column***-->
  <dt>&nbsp;</dt>
  <dt>&nbsp;</dt>
  <dt>&nbsp;</dt>
</dl>

<dl class=" defList defListLast">
  <dt class="defTerm"> 9 </dt> <!--***left column***-->
  <dd class="defDescription"> 10</dd> <!--***right column***-->
</dl>

</main>

<!-- **there is code below **-->

```

## Round corner buttons

As part of the new CSS3 language, most browsers manufacturers agreed that round corners are a good thing. They (Firefox, Chrome, Safari, IE and Opera) have not yet agreed on how the code should be written. To get our code to work in all the browsers, we have to write specific lines of code that cater to each browsers appetite. Think of a table full of people at a restaurant. They all want a meal, but it's not one size fits all. No one can agree on what constitutes a proper meal. Everyone has to have different plate of food before they are happy. And some of the older browsers (IE9) may never be happy, and this code won't work at all.

In web design, we try to build pages that "degrade gracefully". In other words, we take advantage of the newer eye candy effects like round corners, but we make sure that if a browser doesn't support it, our page is still useable. The special eye candy effect may break, but it breaks gracefully, the page still works.

Our navigation works fine, but it would be prettier if it had round corners. You can find all the specs for the proper language to make round corners work in numerous books and web sites. However recently a well known author put up a website that generates them automatically based on interactive sliders.

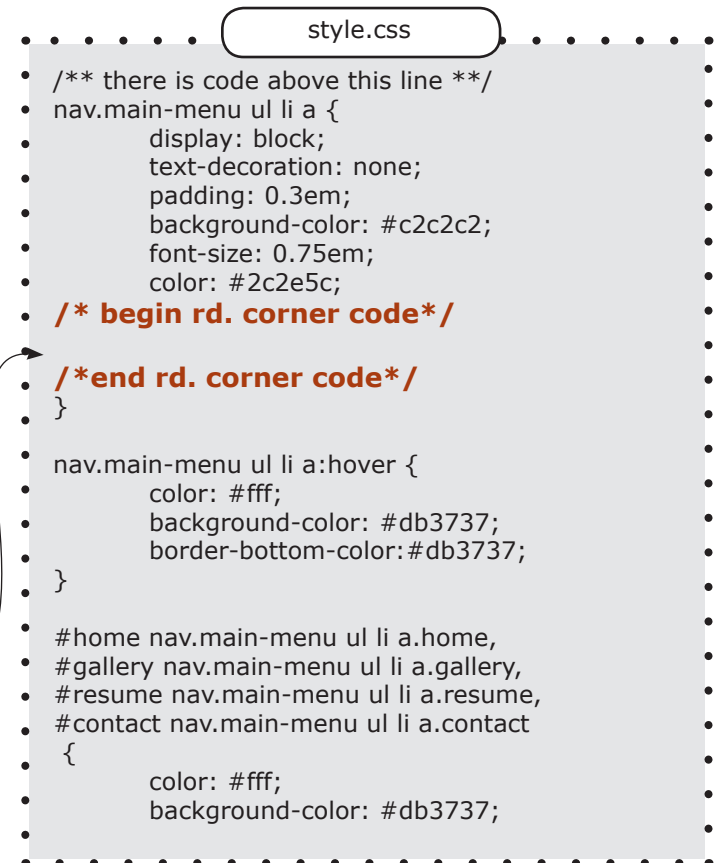
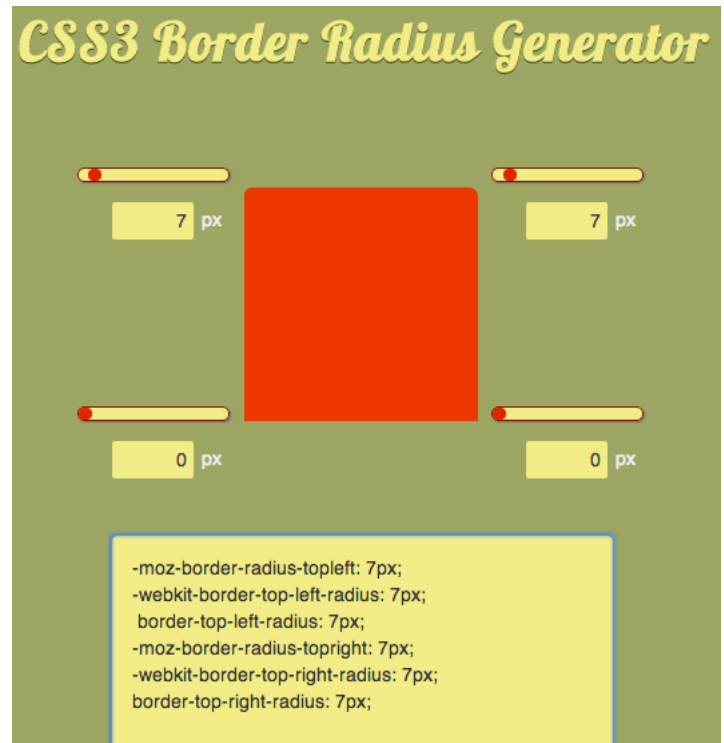
**STEP ONE:** Go to this address:  
<http://css3gen.com/border-radius/>

**STEP TWO:** Drag the top two sliders to a smallish number. Remember our tabs are small, so we don't want the round corner to encroach on the text. I recommend starting with 7px. Drag the two bottom sliders to zero (0). Copy the generated code.

**STEP THREE:** Click the shortcut link in Dreamweaver to open your style.css file. NOTE: if you don't have Dreamweaver, you can do all this in Notepad, or TextWrangler on a Mac.

**STEP FOUR:** Find the style sheet rule for:  
`nav.main-menu ul li a {}`

**STEP FIVE:** Add two starting and stopping CSS comment tags, with a blank line in between. This will allow us to easily isolate the new round corner code, in case it goes bad, or has to be edited.



## Round corner code

**STEP ONE:** Click in between your CSS comments and paste in the round corner code you copied from the website.

NOTE: there are 3 lines of code for each of the two corners we are rounding. Each line is for a different browser.

**-moz-** is for Mozilla Firefox

**-webkit-** is Android, iOS, Safari and Chrome

the third one is the ideal wording, if and when the browser manufacturers can ever agree on a common language. Modern browsers use the last line they understand, and ignore the rest. As of 2016, **border-radius** is so widely accepted you can skip the browser prefixes if you want to clean up your code. Be sure to test in all the browsers and smartphones before you commit to it. You don't want to get a phone call from a client who loves Windows XP and can't see your round corners.

**STEP TWO:** Check your page in all the browsers you have, including your smart phone, if you have uploaded your latest changes.

NOTE: Refer back to **page 59** for upload instructions

style.css

```

/**** more code above this ****/

nav.main-menu ul li a {
    display: block;
    text-decoration: none;
    padding: 0.3em;
    background-color: #c2c2c2;
    font-size: 0.75em;
    color: #2c2e5c;
    /****start round corner code****/
    -moz-border-radius-topleft: 7px;
    -webkit-border-top-left-radius: 7px;
    border-top-left-radius: 7px;
    -moz-border-radius-topright: 7px;
    -webkit-border-top-right-radius: 7px;
    border-top-right-radius: 7px;
    /****stop round corner code****/
}

nav.main-menu ul li a:hover {
    color: #fff;
    background-color: #db3737;
    border-bottom-color: #db3737;
}

#home nav.main-menu ul li a.home,
#gallery nav.main-menu ul li a.gallery,
#resume nav.main-menu ul li a.resume,
#contact nav.main-menu ul li a.contact
{
    color: #fff;
    background-color: #db3737;
    border-bottom-color: #db3737;
}

/**** more code below this****/

```

## CSS 3 eye candy

**STEP ONE:** If you would like to pursue this eye candy round corner theme, you can also put round corners on your footer. You have to paste the round corner code into two different style sheet rules before it will work:

footer

wrapper

**STEP TWO:** If you want the **drop shadow** as I have here, click the **Box Shadow** link on the CSS3Gen site and drag the sliders until it is pretty.

**STEP THREE:** Copy the code, and paste it into your style sheet rule for **#wrapper**

**STEP FOUR:** We are overdue for adding alternate navigation links down in our footer. If you are still on resume.html, click to source code, and put Dreamweaver in split view so you can see design and code.

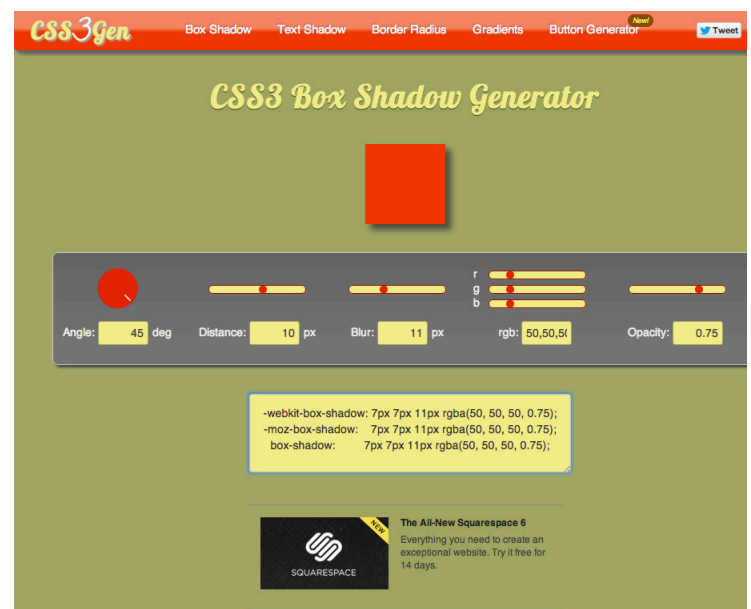
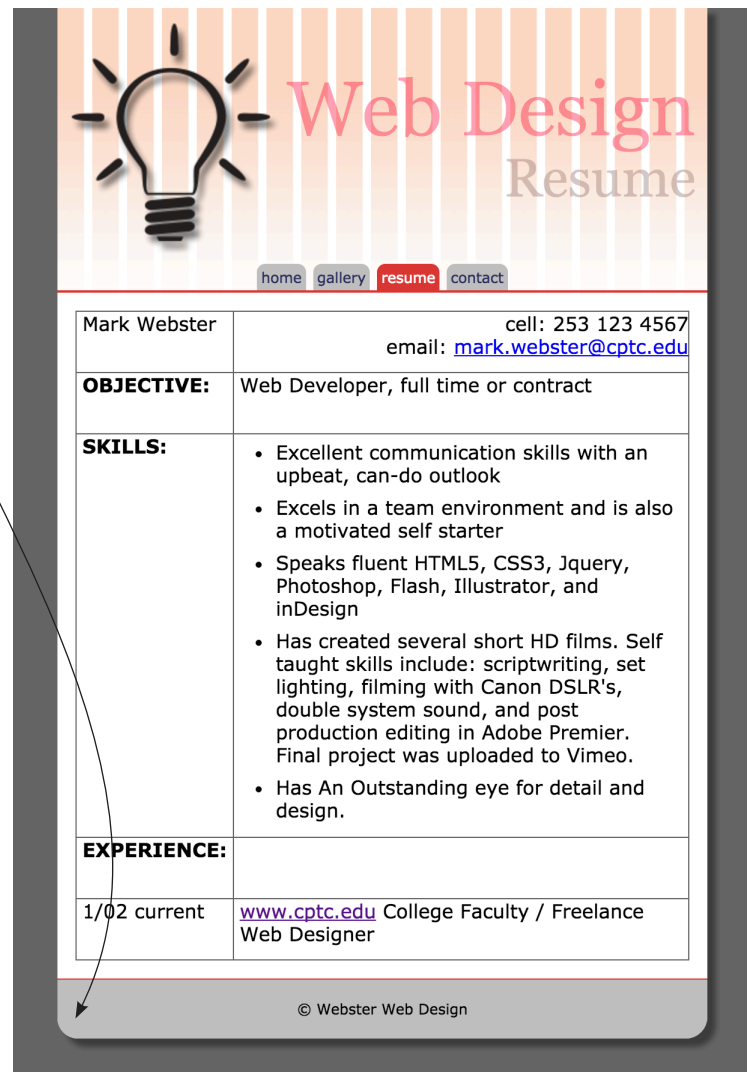
**STEP FIVE:** Highlight the text down in the footer, Dreamweaver will jump to that line of code. Go to the all code view.

**STEP SIX:** type this line of words above the paragraph that contains your copyright info

```
<p>home | gallery | resume | contact</p>
```

The vertical line is called the pipe character, and you get it by holding shift, and pressing your backslash key (\)

```
64:
65: </main>
66:
67: <footer class="footer-area">
68:   <p>home | gallery | resume | contact</p>
69:   <p>&copy; Webster Web Design</p>
70: </footer>
71: </div> <!-- end wrapper div -->
72:
73: </body>
74:
75: </html>
```





## Alternate navigation

**STEP ONE:** Highlight the word home. If our visitor clicks the word home the browser should navigate to the index.html page. Click in the link box of properties and type the word index.html, or, click the folder button to the right of the link box and select the index.html page. Make sure Dreamweaver is looking in the correct defined site folder. You can of course also type all this code from scratch in Notepad. The finished code for the alternate navigation is shown below. Note that I have put each anchor link on it's own line, for visual clarity.

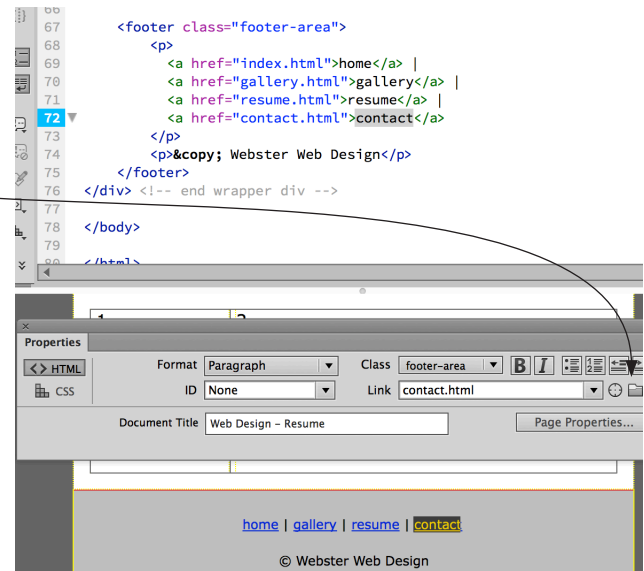
**STEP TWO:** Copy that entire paragraph that contains the alternate navigation links from `<p>....</p>`

**STEP THREE:** Open each of your web pages that do not yet have it: index.html, gallery.html & contact.html

**STEP FOUR:** Paste the new paragraph of links into each page, down in the footer as you did on resume.html

**STEP FIVE:** save all your pages, and then navigate the website with the new alternate navigation links, making sure nothing breaks.

**STEP SIX:** Post it to the internet and check it there in all the browsers at your disposal, including your smartphone if you have one, or one you can borrow.



```

    . . . . . resume.html . . . . .
    </main>
    <footer class="footer-area">
    <p>
    <a href="index.html">home</a> |
    <a href="gallery.html">gallery</a> |
    <a href="resume.html">resume</a> |
    <a href="contact.html">contact</a>
    </p>
    <p>&copy; Webster Web Design</p>
    </footer>
    </div> <!-- end wrapper div -->
    </body>
    </html>
  
```

## Navigation styles

To get control of our link colors, we have to be careful. We already have some tricked out links in our navigation. We don't want to break those when we are changing the colors of our alternate navigation links. If you know Dreamweaver, you know that you can choose >Modify>page properties, click the Links (CSS) menu and pick pretty colors.

But that will get you in trouble because it will override, and destroy your main navigation buttons, and it will do it with a local style sheet up in the head of the html page, instead of in the style.css file. Your best bet is to write the code your self in your **style.css** file.

When you write the style sheet rules, specify that they should only apply if they are within the **footer** element. This is called specificity. That space in the style sheet rule name (called: selector) between **footer** and **a** means the rule only applies if the anchor tag link (a) is contained within the **footer** element.

**STEP ONE:** Add the following code just above the first media query rule in your **style.css** file. Save your file, upload it to the server and check it in all available browsers. If you don't like the colors, you can edit them using [www.colorpicker.com](http://www.colorpicker.com) or Photoshop to pick new hexadecimal colors. If you have Dreamweaver, you can find those new style sheet rules in the CSS panel, where you should see an interactive color picker button. The Brackets code editor has a similar color picker function via the CTL+e function

```

style.css

/**** more code above this ****/

/** begin alternate navigation link colors**/

footer a:link {
    color: #933;
    text-decoration: underline;
}

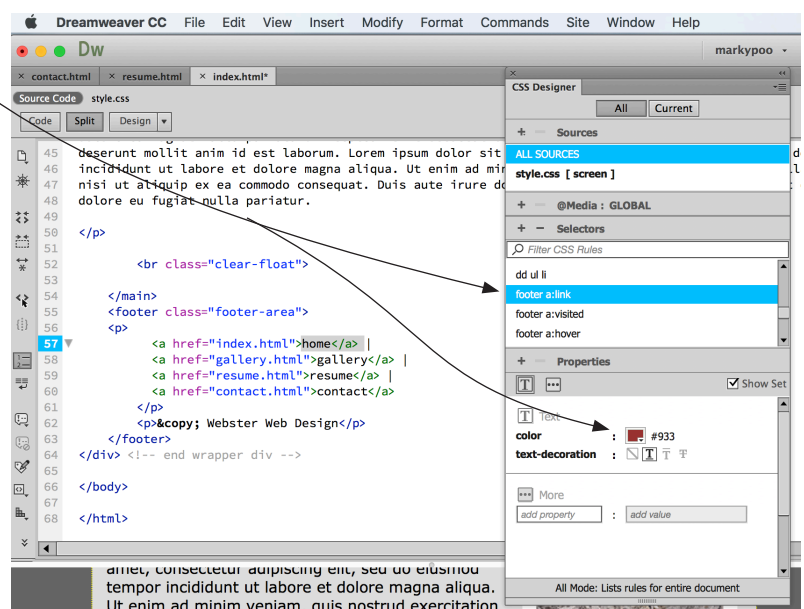
footer a:visited {
    text-decoration: underline;
    color: #993332;
}

footer a:hover {
    text-decoration: none;
    color: #993332;
}

footer a:active {
    text-decoration: underline;
    color: #993332;
}

/**** end altnav link colors****/

```



## Responsive menu

We already have media queries modifying the column count on our gallery page

Now we will add another media query that improves our buttons for viewing on small viewports. Add this new rule at the very bottom of the existing style sheet.

**STEP ONE:** Open your style.css file, add some new blank lines below all the other style sheet rules and type the code you see to the right in bold brown.

NOTE: This code means that if the webpage is being viewed on a screen, (as opposed to being printed) \*and\* it is less than 600px wide, the following rules shall apply:

Make the wrapper div have zero margin, and be 100% wide.

Make the navigation list item buttons act like paragraphs (block) one above the other, instead of inline.

**STEP TWO:** Save your page and view it in the browser window. Gradually shrink the browser window down to about 600px and watch the gray margins around the wrapper disappear.

More of our content is onscreen now, and the buttons are bigger. Those round corners are looking strange, and I'd like the buttons to be taller so they are easier to press. It would also be good to make the h1 tag smaller, and the altnav links larger.

style.css

```

/** there is code above this line **/

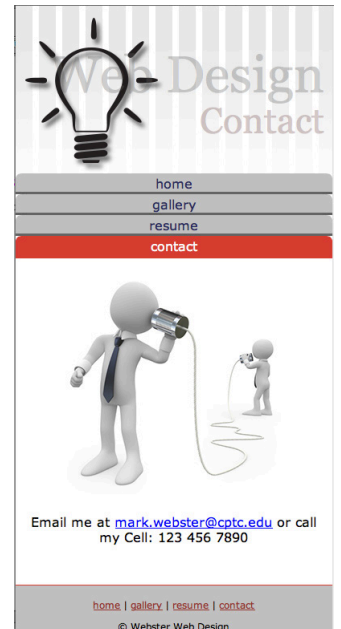
***begin media query***

@media screen and (max-width: 600px)
{
    #wrapper {
        margin: 0;
        width: 100%;
    }

    nav.main-menu ul li {
        display: block;
    }

} *** end media query***

```



## Bigger alt nav

**STEP ONE:** type the code you see to the right in bold brown. Save your page and view it in the browser. The header will get smaller (to fit better), and the alternate navigation will get larger (for fat fingers).

**STEP TWO:** Look up towards the **top** of your style sheet rules for **nav.main-menu ul li a {}**. Copy the entire style sheet rule from the # sign to the closing curly brace. If you applied the round corner code, it will be about 18 lines of code.

**STEP THREE:** Paste it into your style sheet as shown in bold brown below.

style.css

```
/** there is code above this line **/
```

```
@media screen and (max-width: 600px) {
  #wrapper {
    margin: 0;
    width: 100%;
  }
```

```
nav.main-menu ul li {
  display: block;
}
```

```
header.masthead h1 {
  font: 3em/0.8em Georgia, Times, serif;
}
```

```
footer a {
  font-size: 1.5em;
}
```

```
}/*** end media query****/
```

style.css

```
@media screen and (max-width: 600px) {
```

```
  #wrapper {
    margin: 0;
    width: 100%;
  }
```

```
nav.main-menu ul li {
  display: block;
}
```

```
header.masthead h1 {
  font: 3em/0.8em Georgia, Times, serif;
}
```

```
footer a {
  font-size: 1.5em;
}
```

```
nav.main-menu ul li a {
  display: block;
  text-decoration: none;
  padding: 0.3em;
  background-color: #c2c2c2;
  font-size: 1em;
  color: #2c2e5c;
  /***start round corner code****/
  -moz-border-radius-topleft: 7px;
  -webkit-border-top-left-radius: 7px;
  border-top-left-radius: 7px;
  -moz-border-radius-topright: 7px;
  -webkit-border-top-right-radius: 7px;
  border-top-right-radius: 7px;
  /***stop round corner code****/
}
```

```
}/*** end @media screen****/
```

## Media Query: undo the round corners

**STEP ONE:** Make the media query changes in bold brown shown to the right. They are:

Change all your radius values to zero.

Increase the `nav` font-size to 1.5em

Add a `margin-top: 3px;` declaration. This will separate our new improved smartphone buttons.

Tell the lightbulb logo to shrink. It is the background image in `header.masthead`

**STEP TWO:** Post your page live to the internet. Fill out your content as much as possible. Take ownership of your website. Make it as pretty as you can. Make sure it exists in both locations (local and remote server), and back it up to at least one external harddrive.

If you like what you have built, consider purchasing your own domain name. Your own name is the best way to go long term, and it's under \$80 a year for name and hosting.

Search google to find the best web host: "best web hosting 2016"

You will find review sites like this that give you the current contenders:

<http://www.reviews.com/web-hosting/>

style.css

```
@media screen and (max-width: 600px) {
  #wrapper {
    margin: 0;
    width: 100%;
  }
  nav.main-menu ul li {
    display: block;
  }
  header h1 {
    font: 3em/0.8em Georgia, Times, serif;
  }
  footer a {
    font-size: 1.5em;
  }
  nav.main-menu ul li a {
    display: block;
    text-decoration: none;
    padding: 0.3em;
    margin-top: 3px;
    background-color: #c2c2c2;
    font-size: 1.5em;
    color: #2c2e5c;
    /****start round corner undo code****/
    -moz-border-radius-topleft: 0;
    -webkit-border-top-left-radius: 0;
    border-top-left-radius: 0;
    -moz-border-radius-topright: 0;
    -webkit-border-top-right-radius: 0;
    border-top-right-radius: 0;
    /****stop round corner undo code****/
  }
  header.masthead {
    background-size: 25%;
    /*shrinks the lightbulb logo*/
  }
}

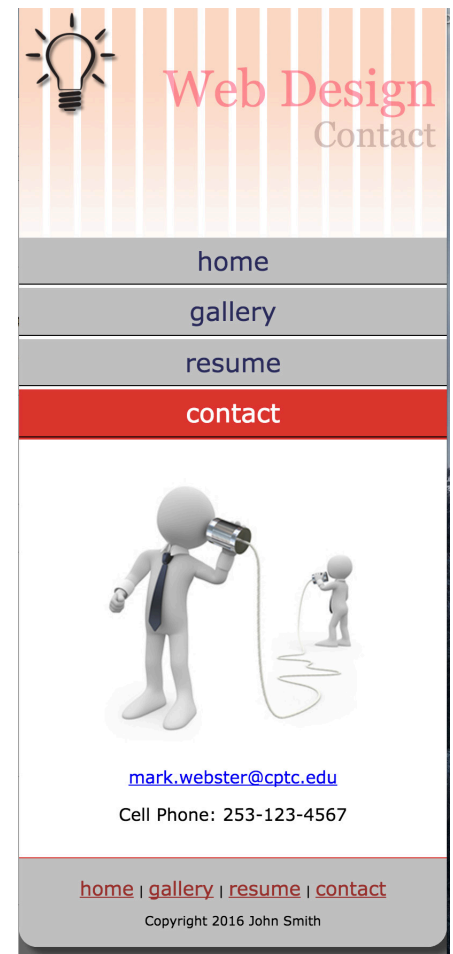
/**** end @media screen****/
```



## Media Query in action

View port smaller than 600px, media query style sheet rule active

Viewport bigger than 600px, media query style sheet rule not active.



Depending on the size of your contact image you may need to add a **media query** style sheet rule to prevent it from blowing through the wrapper on smart phones:

```
body#contact .main-area img {
    width: 70%;
}
```

## Add an Animation page to your website

Most commercial websites have advertising banners. Whether you love them or hate them, websites depend on advertising banners to generate income.

Next we will add a new page to our website called animation, and build in a column on the right hand side. Once we get the column working, we can drop in a mobile friendly animated sidebar banner ad as shown to the right. On this website: \* <http://websterart.com/html/animation-v4.php> \* I have both a horizontal and a vertical animated banner ad.

**STEP ONE:** Copy the **final** website folder from your CPW-118 Web Principals class. If you weren't in that class, I have placed a website you can borrow in the resources folder for this book/class.

It will be named:

**finished-site-cpw-118**

**STEP TWO:** Navigate into the folder on the network or on your harddrive where you are building this website and make a new folder named JohnSmith, if that's your name.

**STEP THREE:** Make a new folder inside your JohnSmith folder named: **Lesson1**. Inside that folder paste the files and folders you copied from your final Web Principals website folder, either yours, or the ones you borrowed from me listed above.

**STEP FOUR:** Start Dreamweaver and create a new site. Define the site to point at the new Lesson1 folder you created in the step above.

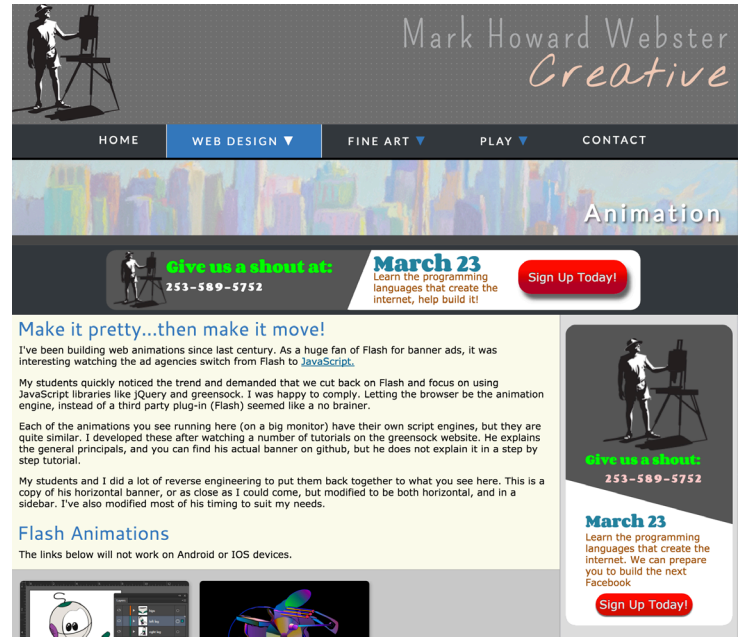
**NOTE:** Instructions for defining a site in Dreamweaver are on Page 59 of this PDF. You can also "define a site" in Brackets or Sublime. Simply drag the folder into the software.

**STEP FIVE:** In Dreamweavers file tab (window>files) double click the **index.html** file to open it.

**STEP SIX:** scroll down to the nav section and add another list item as shown in bold brown here. This will add a fifth button.

index.html

```
<nav class="main-menu">
  <ul>
    <li><a href="index.html" class="home">home</a></li>
    <li><a href="gallery.html" class="gallery">gallery</a></li>
    <li><a href="animation.html" class="animation">animation</a></li>
    <li><a href="resume.html" class="resume">resume</a></li>
    <li><a href="contact.html" class="contact">contact</a></li>
  </ul>
</nav>
<main class="main-area">
```



## Adding links to new Animation page

**STEP ONE:** Scroll down to the footer element and add a link for the new animation page to the alternate navigation area as shown here in bold brown.

**STEP TWO:** Dreamweaver has a cool new function that is worth experimenting with. Click the button for **Design**, and then click the **Live** button. Dreamweaver comes very close to how a real browser would work. If you don't like that, choose **file>preview in browser**, or press **F12**.

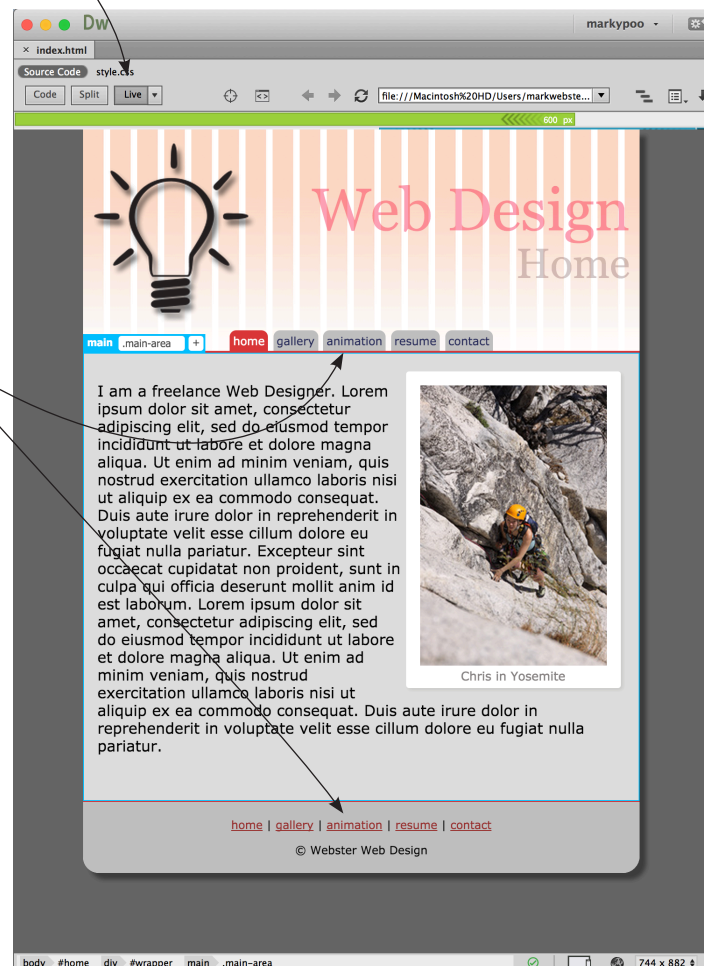
```

    . . . . . index.html . . . . .
    <footer class="footer-area">
        <p>
            <a href="index.html">home</a> |
            <a href="gallery.html">gallery</a> |
            <a href="animation.html">animation</a> |
            <a href="resume.html">resume</a> |
            <a href="contact.html">contact</a>
        </p>
        <p>&copy; Webster Web Design</p>
    </footer>
</div> <!-- end wrapper div -->
</body>
</html>
  
```

**STEP THREE:** Check to make sure you have the new navigation links in place to the 'yet to be created' animation.html page.

**STEP FOUR:** Open **all** the other html pages on your website and make these same additions to the top and bottom navigation.

**NOTE:** If you know, or have done PHP includes, they are a much more efficient way to manage common repeating elements on a multipage website, like these navigation elements. Alternatively, you can use the Dreamweaver "library item" function. Search this pdf for "library item", I explain it later on.



## Customize the Animation page

### STEP ONE: In

Dreamweaver, on the index.html page, choose file save as and change the name to **animation.html**. NOTE: make certain that your software does the "save as" to the right folder location. Some software programs (Dreamweaver) will save to the last saved location which may be off on the planet of Mars somewhere.

**STEP TWO:** View the source code and edit the title to read Animation

**STEP THREE:** Edit the <h1> tag to read Animation.

**STEP FOUR:** Edit the body id to read **animation**. New code is shown in bold brown.

**STEP FIVE:** in **style.css**, add the new selector to the existing chained selectors as shown. In case you've forgotten, this rule tells the browser that if it finds any of those chained selectors to be true, it should turn the button red, giving us our "urhere" effect.

**NOTE:** There are commas on each chained selector except the last one.



animation.html

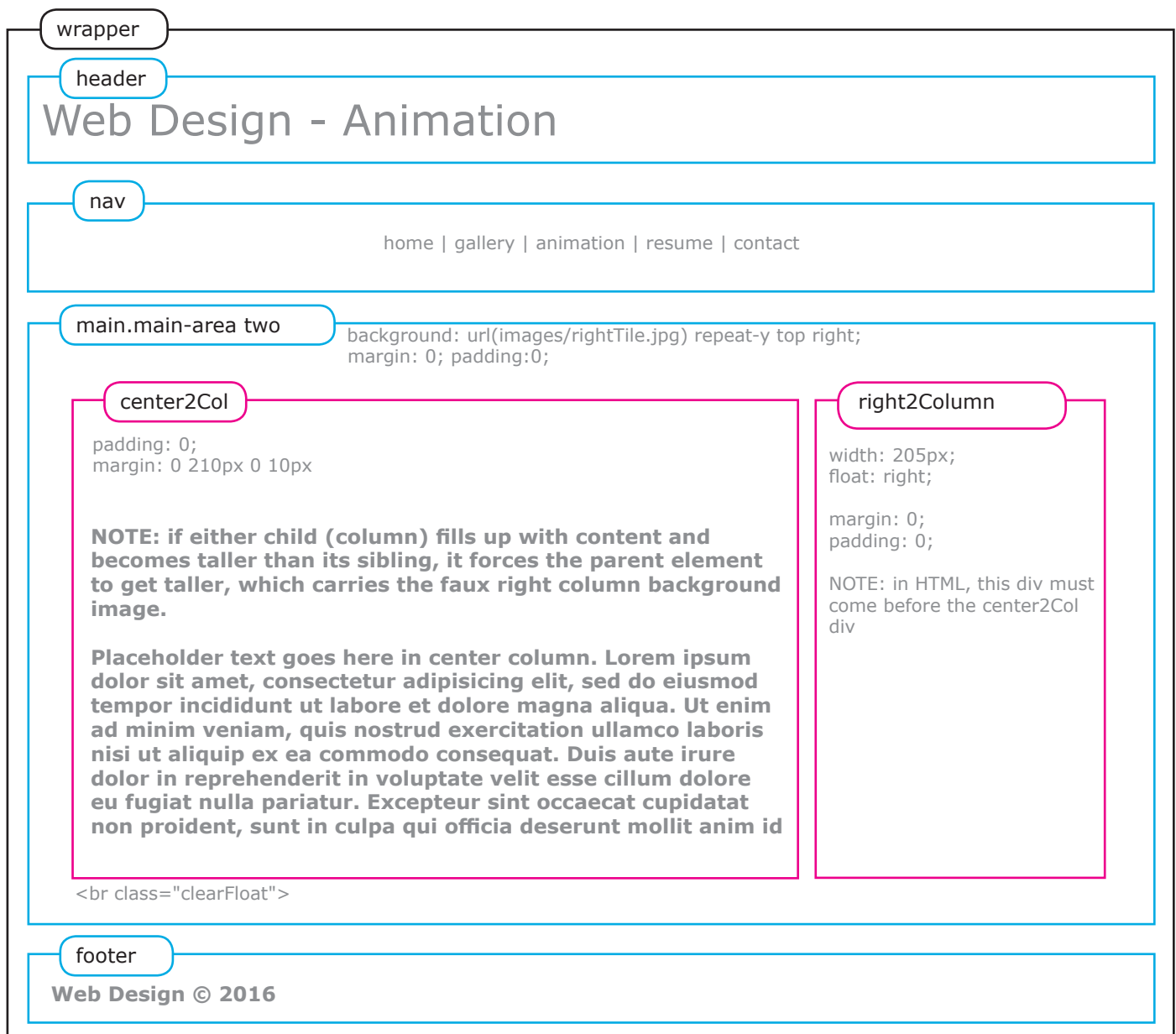
```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>
John Smith - Animation
</title>
<link rel="stylesheet" type="text/css" media="screen" href="style.css">
</head>
<body id="animation">
<div id="wrapper">
<header class="masthead">
<h1>Web Design<br><span>Animation</span></h1>
</header>
<nav class="main-menu">
<ul>
<li><a href="index.html" class="home">home</a></li>
<li><a href="gallery.html" class="gallery">gallery</a></li>
<li><a href="animation.html" class="animation">animation</a></li>
<li><a href="resume.html" class="resume">resume</a></li>
<li><a href="contact.html" class="contact">contact</a></li>
</ul>
</nav>
```

style.css

```
nav.main-menu ul li a:hover {
    color: #fff;
    background-color: #db3737;
}
#home nav.main-menu ul li a.home,
#gallery nav.main-menu ul li a.gallery,
#animation nav.main-menu ul li a.animation,
#resume nav.main-menu ul li a.resume,
#contact nav.main-menu ul li a.contact {
    color: #fff;
    background-color: #db3737;
}
main.main-area {
    padding: 15px;
}
```

## From one column to two columns

The graphic below illustrates the structure we will use for converting our one column interface into two columns. My element naming conventions are based on the assumption that we will, at a later point in time, move onto a three column model. But we will start with two columns. Study the graphic below. I have added in some of the key CSS declarations that make the two column structure function. There is an image in the background of the `<main class="main-area two">` element. This image will be made in Photoshop using colors that match the background of the animation side banner we will drop into the column. The animation itself will have a fixed size. This two column structure is designed to accomodate variable amounts of text in either left or right column. In other words, if either column expands vertically, it will force the parent element `main` to expand. Because there is a background image in `main`, that background image will tile downward, presenting the appearance of a "column", even though there may not be content filling the right column element.





## Animation page HTML markup

Before you start this process of building the animation page, **make a copy of your entire website** and paste it into both your flash drive, and your desktop>my documents. If this blows up on us, we will need a way to step back in time, and these backups will do that for us.

**STEP ONE:** On **animation.html**, clear out your **<main>** element so that the only thing left is one short paragraph followed by the **<br class="clear-float">**

**STEP TWO:** Find this tag:  
**<main class="main-area">**

Change it to read:

**<main class="main-area two">**

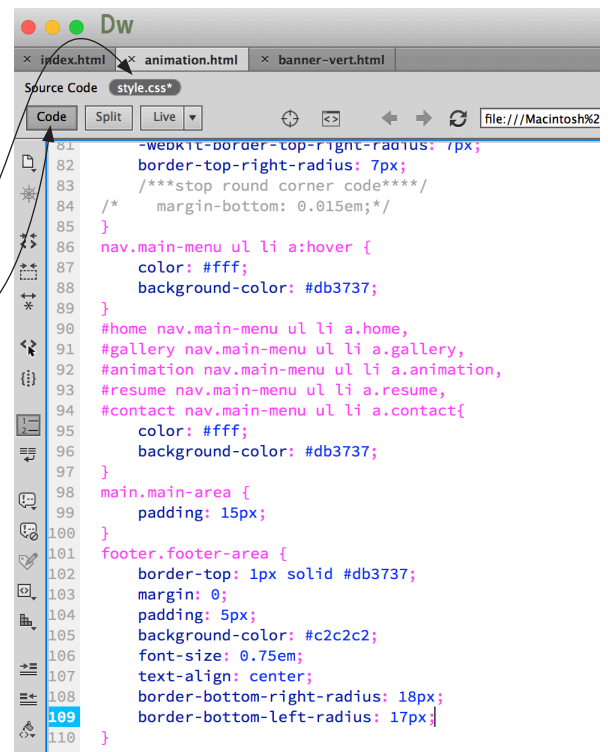
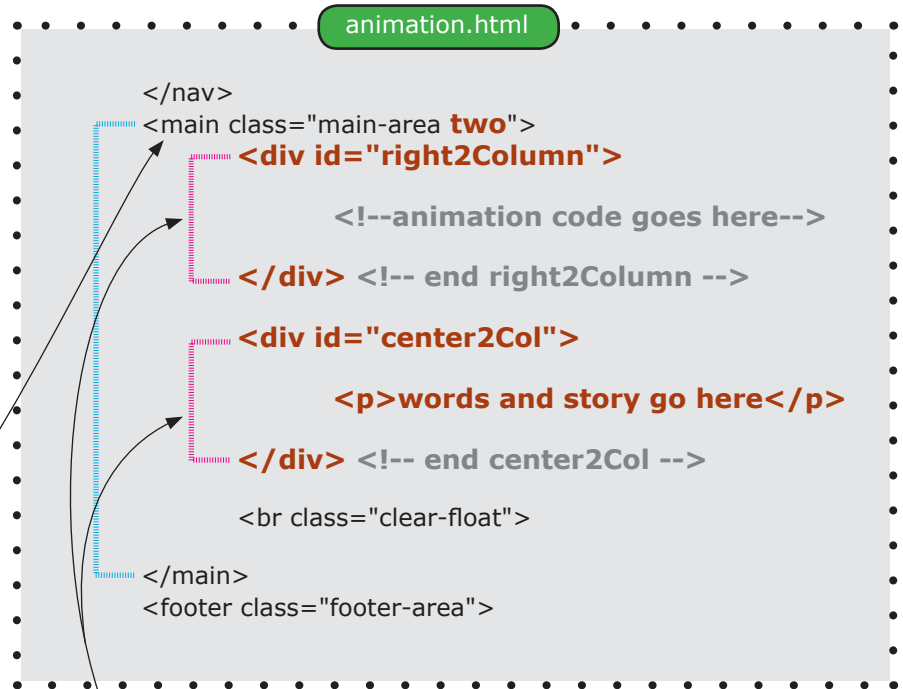
**NOTE:** We had to create a new style sheet rule named **two** for this animation page so we can add special properties to this **main** element that will enable two columns.

**STEP THREE:** Inside the **<main class="main-area two">** add two new starting and stopping divs named: **right2Column** and **center2Col** as shown. Put a short sentence in paragraph tags in the second div.

**STEP FOUR:** Make sure the **<br class="clear-float">** is between the end of the **center2Col** div and the ending **main** element. The **<br>** tag is needed because the right column will be told to float and could bust through the bottom of it's parent.

**STEP FIVE:** Examine the bare bones structure above, as well as the illustration on the previous page, make sure everything is named and organized correctly.

**STEP SIX:** In Dreamweaver, click the shortcut button to the linked style.css file. Also click the Code button and scroll down to the style sheet rule for **main.main-area**.



## Animation page style changes

**STEP ONE:** Add a new style sheet rule named **main.main-area.two** immediately below the **main.main-area** style sheet rule as shown in bold brown. There are no spaces in the selector.

**NOTE:** this rule is saying to the browser: If there is an element named **main** that has a class of **main-area** **AND** a class of **two**, apply the following declarations in the curly braces. The element will inherit the **main.main-area** declarations, but will override them if there is a conflict

**STEP TWO:** Add a new rule called **#right2Column**.

**STEP THREE:** add a new rule below that called **#center2Col**

**STEP FOUR:** Edit the new style sheet rules as shown in bold brown below.

### EXPLANATION:

The rightTile (not made yet) gives us our faux column when the content in either center or right gets taller than the banner ad.

**#center2Col** div has not told to float, or to have a width, it figures it out. We give it 210px margin on the right to provide space for the **right2Column** to float into.

In the html markup, **right2Column** must come first within the parent **main** element. The border-top property on the main element is a hack to force a tight fit between nav and main.

**NOTE:** after I inserted some words the right column shows no indication that it extends to the footer. Even if you gave it a background color, it would only be as tall as it's text. It gets it's height from the words within it. Our solution is to put a tiling image in it's **main** parent. Just like a bike tire gets bigger when you inflate it's inner tube, our **main** will get taller if either of it's children get taller.

```

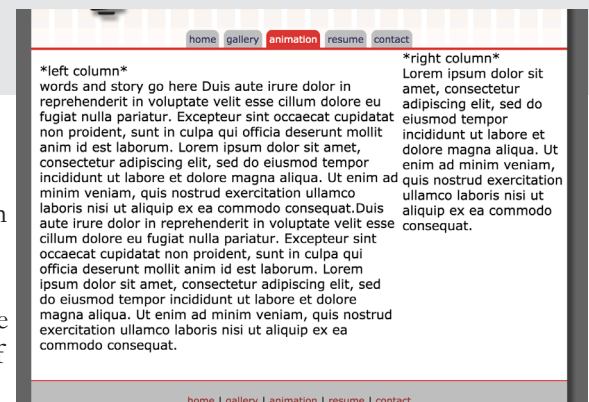
style.css
/** there is code above this line */
}
main.main-area {
    padding: 15px;
}
main.main-area.two{
}
#right2Column {
}
#center2Col{
}
/** there is code below this line */

```

```

style.css
main.main-area.two {
    margin: 0;
    padding: 0;
    background: #f8f2ec; /*color is optional*/
    background: url(images/rightTile.jpg) repeat-y top right;
    border-top: 1px solid #db3737;
}
#right2Column { /*this div must come before center2Col div in html markup */
    width: 205px;
    float: right;
    margin: 0;
    padding: 0; /*never put padding and width on same element*/
}
#center2Col {
    padding: 0;
    margin: 0 210px 0 10px;
}

```

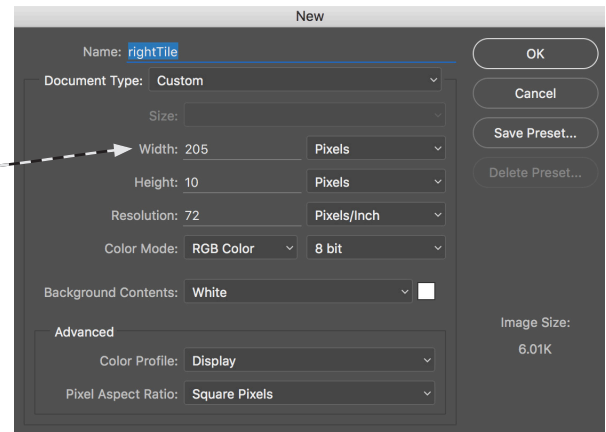


## Right column hack

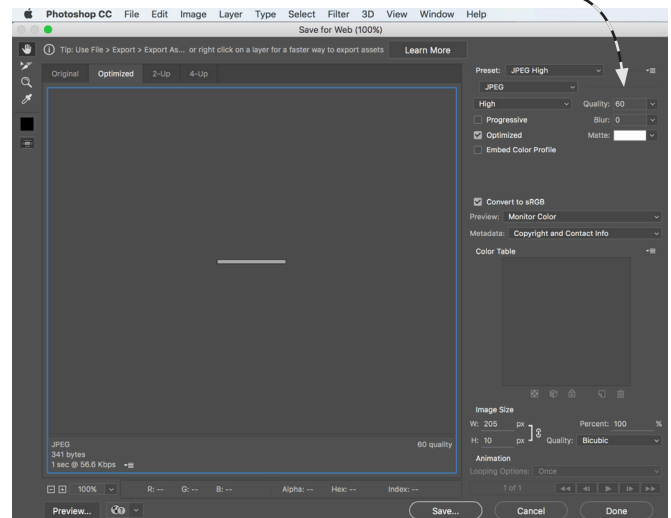
**STEP ONE:** In Photoshop, choose file>new and tell it to be **205px** wide by **10px** high, resolution: **72**. Give it a name: **rightTile**

**STEP ONE:** Choose #aaaaaa as your foreground color, though you can change it later.

**STEP TWO:** Press the alt and backspace key together. That should pour your gray foreground color into the layer in Photoshop.

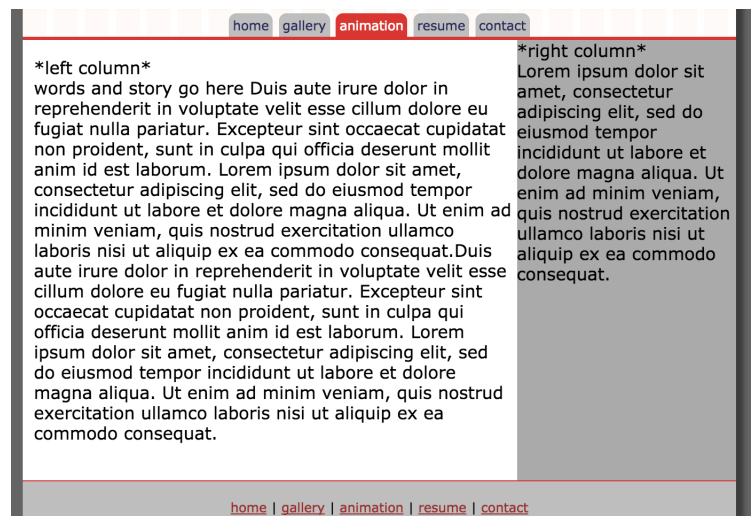


**STEP THREE:** Choose file>export>save for web(legacy). Choose high quality jpg and click OK. Save the file down into your images folder. If you are new to our network, click the desktop button on the left, and look for the shortcut to "room107storage". Name it **rightTile.jpg**



\\ms1729\shared storage\room107 storage\mwebster\CPW-250\johnSmith\Lesson1\images\

**STEP FOUR:** Refresh your page in the browser, you should now see the tile image filling out your right column. It is actually tiling through the parent element **main.main-area.two** but it looks like it is coming from the **#right2Column** div, which is currently simply filled with a few sentences of optional latin text



# Grabbing the animation elements

**STEP ONE:** Go to the resources folder for this book/class and copy the folder named **vertical-banner-ad**. Paste it into your current site folder at the root.

**STEP TWO:** Go into the **vertical-banner-ad/js** folder and copy all the \*.js files, there are 4 of them. Paste those 4 js files into the site root js folder.

**john-smith/lesson1/js** is your current site root js folder.

**STEP THREE:** From the **vertical-banner-ad** folder copy everything but the js folder. Paste the 3 files into the root of your current site. They should live at the same level as **style.css**

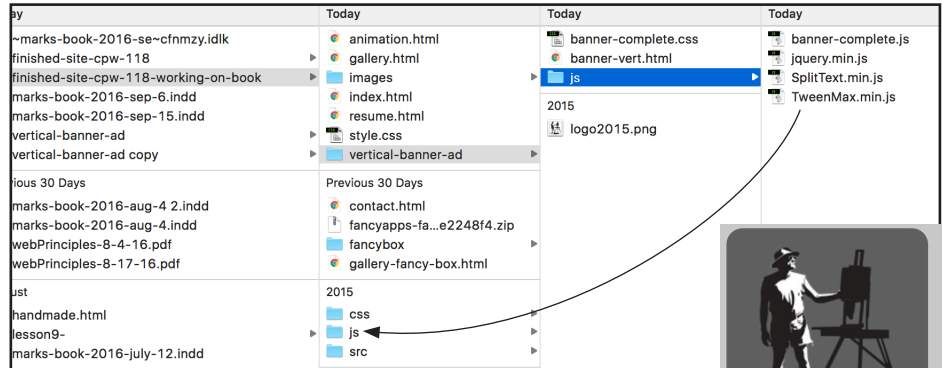
**STEP FOUR:** Now that you have moved all the parts of the animation into place, double click on **banner-vert.html** at the root of your site, it should animate and look like this.

NOTE: if you took the CPW-225 class, you can use your own vertical banner ad instead of mine.

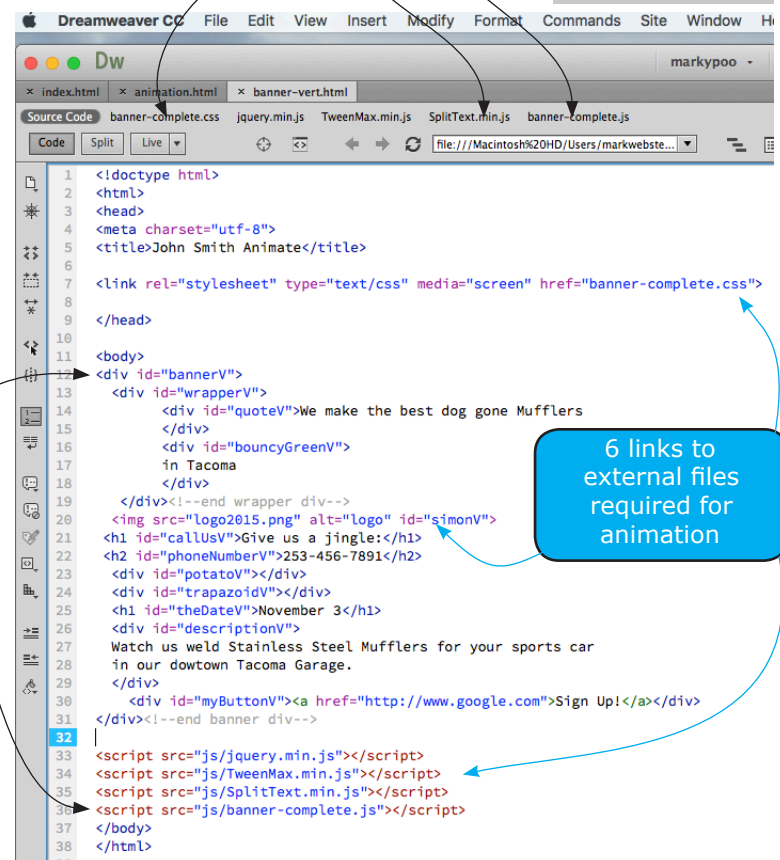
**STEP FIVE:** Next open the same **banner-vert.html** in Dreamweaver. You can use any code editor, but Dreamweaver has some distinct advantages when working with web pages that depend on a lot of imported files. It all comes down to file management and logic, but DW just makes it easier.

**STEP SIX:** Examine the links to external files. There 6 of them if you include the logo image. Click each of the shortcut links and make sure Dreamweaver can find them.

**STEP SEVEN:** Copy all the code between the starting and stopping body tags.



5 shortcut buttons to imported files, test them!



## Bringing in the animation code

**STEP ONE:** Open `animation.html`, clear out the contents of the right column div so you just have the starting and stopping div elements like so:

```
<div id="right2Column">

</div> <!-- end right2Column -->
```

**STEP TWO:** paste the animation code inside those two divs as shown in brown code to right.

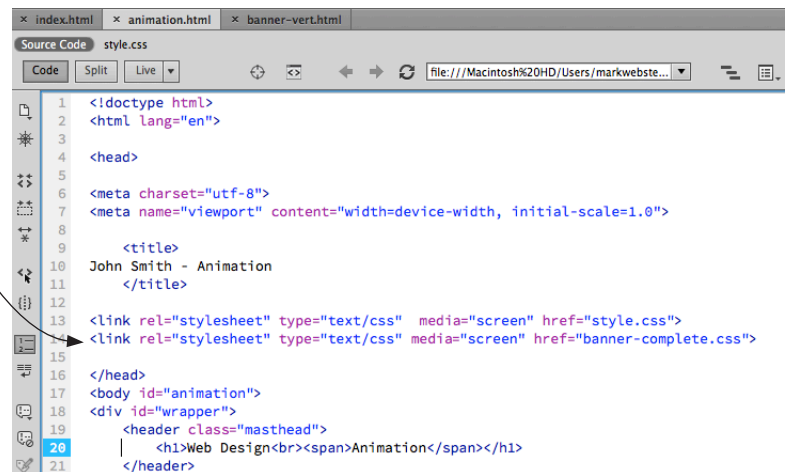
**STEP THREE:** go back to `banner-vert.html` and copy the link to the external style sheet named: `banner-complete.css`

**STEP FOUR:** paste the link into `animation.html` right below the existing link to `style.css` as shown in the screenshot.

animation.html

```
<main class="main-area two">
  <div id="right2Column">
    <!--animation code starts here-->
    <div id="bannerV">
      <div id="wrapperV">
        <div id="quoteV">We make the best dog gone Mufflers
        </div>
        <div id="bouncyGreenV">
          in Tacoma
        </div>
      </div><!--end wrapperV div-->
      
      <h1 id="callUsV">Give us a jingle:</h1>
      <h2 id="phoneNumberV">253-456-7891</h2>
      <div id="potatoV"></div>
      <div id="trapazoidV"></div>
      <h1 id="theDateV">November 3</h1>
      <div id="descriptionV">
        Watch us weld Stainless Steel Mufflers for your sports car
        in our downtown Tacoma Garage.
      </div>
      <div id="myButtonV">
        <a href="http://www.google.com">Sign Up!</a></div>
      </div><!--end banner div-->

      <script src="js/jquery.min.js"></script>
      <script src="js/TweenMax.min.js"></script>
      <script src="js/SplitText.min.js"></script>
      <script src="js/banner-complete.js"></script>
    <!--animation code stops here-->
  </div> <!-- end right2Column -->
```





## Tweaks to the animation

**STEP ONE:** Once the animation is running, you may want to tweak the margins around the banner.

**STEP TWO:** Go into the `banner-complete.css` file and edit the `#bannerV` margin declaration to look like this:  
`margin: 5px auto;`

**STEP THREE:** Save your changes and refresh your browser. It should look like this.

**Troubleshooting:** If it doesn't work, use the shortcut buttons in Dreamweaver to make sure those imported files can really be found. When you write a line of code that links to an external file, like `jquery.js` for example, Dreamweaver will put a shortcut button on the file. But it won't tell you your link path is wrong unless you click the button. Same thing with the logo image. If it doesn't show up hover your mouse over the `<img>` tag. Dreamweaver, Sublime and Brackets will all give you a preview thumbnail if the path to the image is correct. If not you have to do some file management thinking.

**Improvements:** If you'd like to use your own images or text, or text colors and font families, have at it. It's just html and css you will be changing so you can't hurt anything as long as you keep your new image and font sizes and amounts about the same. The javascript is safe in external files. Editing the javascript is risky unless you like to live dangerously. The animation all happens in `banner-complete.js`. Just make sure you backup before going too crazy. If you like the animation, I do teach a class called CPW-225, or you can simply go to [www.greensock.com](http://www.greensock.com) and learn it yourself, he has tons of tutorial videos there and on youtube. Plus he has now written an excellent book. I highly recommend it:

<https://www.nobledesktop.com/books/gsap>



## CSS Column theory

Making columns work in CSS is tricky because the you never know how much content will be placed in any one column.

If the left column `center2col` becomes taller than the animation because of a longer story, the background image we made in Photoshop (`rightTile.jpg`) will begin to appear (tiling downward) on the right. The `right2Column` will not get any taller, but it looks like it is getting taller because of the background image in it's parent. Try it and see: paste a bunch of latin placeholder text into the two columns, one at a time and watch how they behave.

You can get free latin placeholder text here:

<http://www.lipsum.com/>

You can also get hipster text, if you prefer humor:

<http://hipsum.co/>

Now that we have a right column, lets add a left column. Perhaps our client wants to have his navigation over there, or some round corner boxes. Round corner boxes also known as: 'action boxes' can offer 'click throughs', such as small advertisements, or a thumbnail picture that leads the viewer deeper into the website. In this screen grab from the Multicare website, the navigation and advertisements are stacked up in the right and left hand columns, offering interior navigation to the site. Incidentally 3 of our web program graduates work for the web design firm that built the Multicare site.



**STEP ONE:** Save all your changes and close down Dreamweaver.

**STEP TWO:** Copy your Lesson1 folder. Paste it right back in on top of itself. Windows will name it **Copy of Lesson1**. Change the name of the folder to **Lesson2**

**NOTE:** We did this for several reasons. It gives us a back up as we move forward. For those of you in my college classes, it gives me a stopping point where I can grade, and it will give you practice redefining your website in Dreamweaver to point at the new Lesson2 folder.

**STEP THREE:** Redefine your website in Dreamweaver to point at the new **Lesson2** folder.

**NOTE:** Instructions for **defining a site** in Dreamweaver are on Page 59 of this PDF.

# Flexbox

## A little history

15 years ago we built our web interface structures with tables. Tables were reliable because content inside a table cell would never expand beyond the wall of the cell. You could tell the table, and / or the table data cell children how wide to be using pixels or percents, and it rarely backfired. You could nest tables inside tables to get even more granular control. In this example, (which you can still easily do in Dreamweavers Design view) I first built a one column, two row table. Inside each row I inserted a one row 3 column nested table. Using this structure, you can easily drop logo and banner images into cells 1 and 3, along with an expander pattern image from Photoshop in cell 2. Then you can stack buttons in cell 4, content in cell 5, some action boxes in cell 6 and you have state of the art web design from last century.

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |

seemed like too much to ask. We had that control with tables last century. Surely Al Gore, who invented the internet, could improve on it.

In this book so far we've built an interface structure that relies on floats and positioning. It is very reliable once all the bugs are worked out, and it performs well in all browsers going back several years as of 2016.

## Enter flexbox

Think of flexbox as a flexible 3 car garage. By giving the garage a property of **display: flex;** all 3 of the bays in the garage are exactly the same size: height and width. If anything gets bigger, everything maintains the same relationship. For example, if the viewport (browser) gets bigger, the garage gets bigger, and the bays inside the garage maintain their relationship.

If we put a bigger car in one of the garage bays, that bay will get taller to accomodate the content, and it's sibling bays will get taller too, to maintain the equal relationship between the 3 garage bays.

If you remember the dark days of tables, you are probably thinking that this was just how tables behaved, and you are right. But even better, with flexbox, we can wrap the garage bays...that was never possible with table data cells. We can also easily control alignment both inside and outside the bays in both x and y directions, including justify. Plus we can even re-order the bays, all without touching the html markup.

Then search engines got smarter and it was found that content inside tables was not easily searchable. Establishing relationships between images and text in adjacent table rows or columns required a lot of extra programming in the `<table>`, `<tr>` and `<td>` tags.

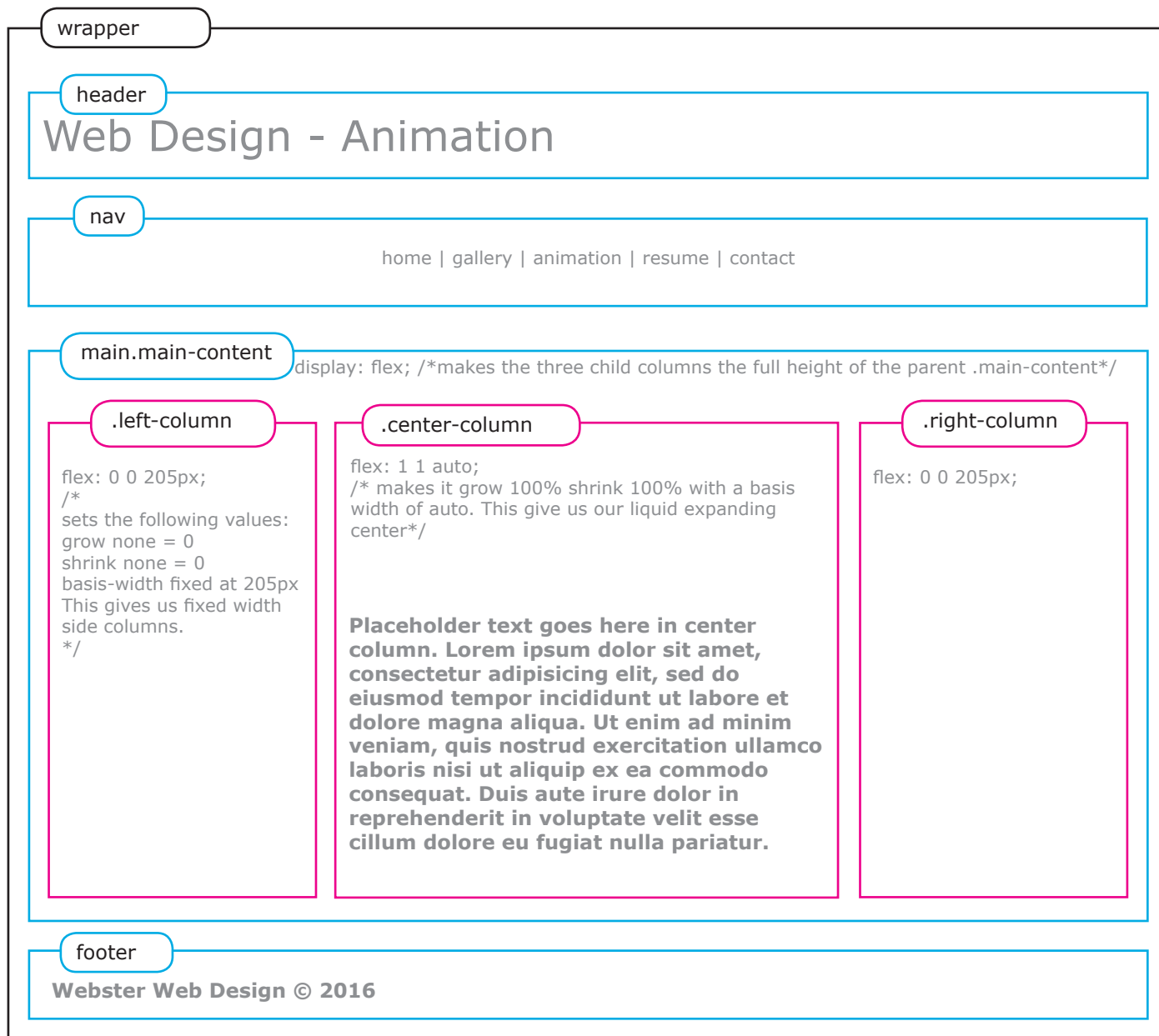
Something better was required and it was discovered that unordered lists were friendlier to search engines because of the implied relationships between content inside the `<ul>` `<li>` tags.

By using a complex matrix of `<ul>` `<li>` and `<div>` tags, combined with floats and positioning (position: absolute, relative) the web gradually moved towards an interface design environment that as almost as effective as tables used to be.

We just finished out our two column interface using that very structure.

Behind the scenes the web community and browser developers were always searching for something easier and more reliable. Everyone wanted a structure that allowed for simple alignment capabilities in both the x and y direction. And despite the fluid nature of the viewport (browser width), we needed a way to accurately control column and row width and height. It never

## Liquid three column structure, illustrated



In the graphic above, I've anoted the tags with the key style sheet declarations that make 3 columns work reliably. This is not to say this is the only way to do it. Search google for **3 column CSS tutorial** and you will find many variations on this theme.

The beauty of this column structure is that you can put as much as you want in any column and the other two columns match height. CAUTION: nothing in a right or left column can be wider than their specified width.

NOTE: This 3 column structure is not intended to be responsive to smartphones yet. It can be, but we will address that later.

## 3 column bones

We need a blank slate to start this new 3 column structure. Our index page could use some jazzing up. Let's convert that to a three column. If you have text or images there you want to keep, copy the content area code to a notepad window for safe storage. You can still use the index page content, but you need to move it out of the way while we change that one column content area into 3 columns.

**STEP ONE:** Note that here I have cleared out everything except the starting and stopping main tags.

**STEP TWO:** in the starting `<main>` tag, change the class to read: `class="main-content"`. We still need a main element, but we need one with a unique class name so we can enable 3 cols and not worry about breaking things on the 2 column animation page.

**NOTE:** HTML5 introduced a series of new tags called **Semantic** tags including, but not limited to: header, hgroup, main, article, section, aside and footer. For example, instead of using a meaningless tag like

`<div id="content">`, we can use the `<main>` tag. Modern browsers, news feeds and screen readers (for blind people) recognize a `<main>` tag as being the important part of your webpage. Meaning, it has actual unique and important human readable content, as opposed to the header, nav, or footer, which are more structural. Also, by using the `role="main"` property, we can speak to older browsers and devices that may not understand the semantic meaning of `<main>`.

More information can be found here:

[http://www.w3.org/WAI/GL/wiki/Using\\_HTML5\\_section\\_elements](http://www.w3.org/WAI/GL/wiki/Using_HTML5_section_elements)

and here:

<http://www.html5-tutorials.org/new-section-tags/introduction/>

**STEP THREE:** Add the code shown in bold brown. There are **Semantic** reasons for why I am using these tags. Read the linked webpages above for a deeper understanding, but the `<article>` tag in particular is meant to be a stand alone block of text. Meaning, if it was pulled into someones news feed, or if a blind persons software read just one thing on the page...it would be the text in the `<article>` tag.

```

index.html
</nav>
<main class="main-area">
</main>
<footer class="footer-area">

```

```

index.html
</nav>
<main class="main-content" role="main">
</main><!--end main.main-content-->
<footer class="footer-area">

```

```

index.html
<main class="main-content" role="main">
  <aside class="left-column">
  </aside> <!--end aside.left-column-->

  <article class="center-column">
  </article><!--end article.center-column-->

  <aside class="right-column">
  </aside><!--end aside.right-column-->

</main><!--end main.main-content-->

```



## Placeholder text in the columns

Now that we have the bones of the columns, we need to fill them out with some paragraphs of text.

**STEP ONE:** Type a starting and stopping paragraph tag `<p></p>` inside each of the 3 column elements. Go out to <http://lipsum.com/> and copy about 150 words of latin placeholder text. Paste the latin inside the paragraph tags. The new code is shown in bold brown to the right.

**NOTE:** For brevity's sake, I do not show all the latin text in this example, but you need to have it in your code.

**STEP TWO:** Put about 75 extra latin words in the center column. These are not precise numbers. Do rough guesses by counting 25 words and estimating. Because the center column will be wider than the outer columns, it needs more placeholder words.

**STEP THREE:** Save your page.

```

index.html

<main class="main-content" role="main">
  <aside class="left-column">
    <p>
      *left-column* Lorem ipsum
      dolor sit amet, consectetur adipisicing elit, sed do eiusmod
      tempor...
      (about 150 words)
    </p>
  </aside> <!--end aside.left-column-->
  <article class="center-column">
    <p>
      *center-column* Lorem ipsum
      dolor sit amet, consectetur adipisicing elit, sed do eiusmod
      tempor...
      (about 210 words)
    </p>
  </article> <!--end article.center-column-->
  <aside class="right-column">
    <p>
      *right-column* Lorem ipsum
      dolor sit amet, consectetur adipisicing elit, sed do eiusmod
      tempor...
      (about 150 words)
    </p>
  </aside> <!--end aside.right-column-->
</main> <!--end main.main-content-->

<footer class="footer-area">
  <p>&copy; 2016 Webster Web Design</p>
</footer>

</div> <!-- end wrapper div -->

</body>
</html>

```

# In pursuit of the Holy Grail

**STEP ONE:** View your page in the browser (Chrome, Firefox, Safari), do not use IE, it does not like the CSS media queries (which form the buttons) running on the Room 107 campus network. Oddly IE does allow the buttons to appear correctly on the live <http://elmo server>.

Observe how we have no control whatsoever of our columns. The browser sees the tags and ignores them, or at most treats them like paragraphs.

Getting browsers to allow two fixed width columns on the outside, with an expanding liquid center column is so difficult in CSS that it has been referred to as the **Holy Grail** of web design.

Honest, I'm not making this stuff up. Read it for yourself here:

<http://alistapart.com/article/holygrail>

[http://en.wikipedia.org/wiki/Holy\\_Grail\\_\(web\\_design\)](http://en.wikipedia.org/wiki/Holy_Grail_(web_design))

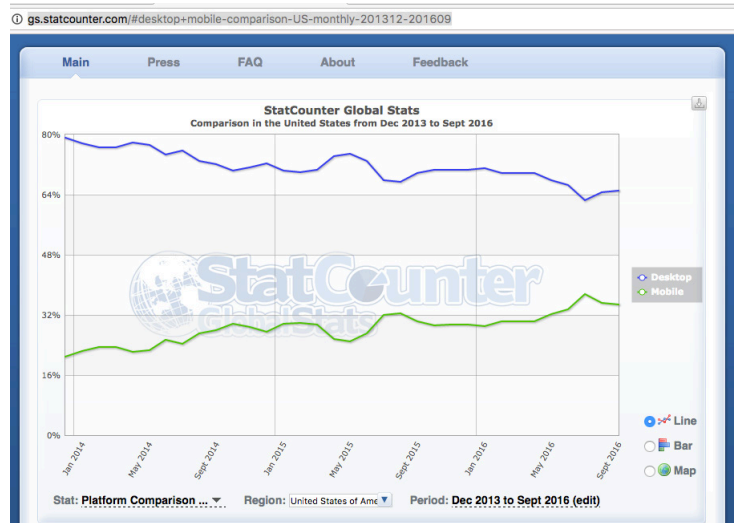
An expanding center column that resizes when the viewport expands (big monitor), framed on either side by fixed width columns holding sidebar elements (navigation, advertisements, action boxes, etc) is the ideal presentation for a website viewed on a full size computer. This can also be adapted down to mobile devices with responsive design, which we will do later.

## Keep your target audience in mind.

In North America, the percentage of users of smartphones to users of computers changes rapidly.

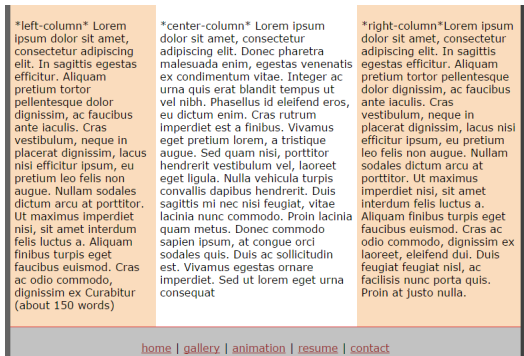
View the stats at:

<http://gs.statcounter.com/#desktop+mobile-comparison-US-monthly-201312-201609>



## Styles for the columns

**STEP ONE:** type the code shown here in bold brown. Note how the display flex property automatically figures out that the columns should be the same height, and guesses at what we intended for the widths.



**STEP TWO:** Add this new code. This time we are speaking to the children of the `.main-content` parent. The first number after flex: (0) is how much growth we want. The second value (0) is shrink. We want fixed width outer columns, no shrinking, no growing. The third value is basis-width, and in this case we set it to pixels. And with that we easily achieve the fixed width side columns, with a liquid center.

**NOTE:** You can drop custom made Photoshop pattern images in the background of the columns, and tile them in the y direction (vertically). If you do this you will need to make them the same width as the columns. Then cut the padding declarations from the columns themselves and move them to the paragraph tags inside the columns. For example:

```
.left-column p {  
    padding: 0.4em;  
}
```

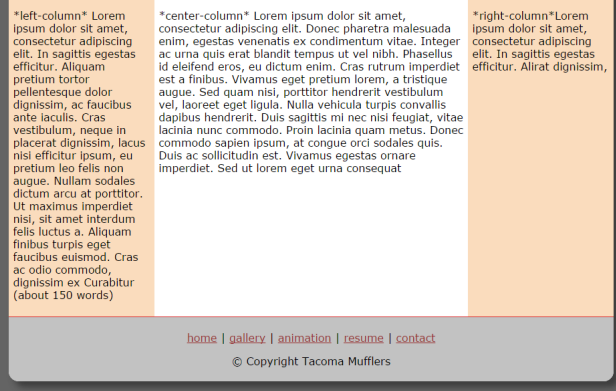
This will resolve any conflict between how browsers display an element with both padding and width properties. Don't put both on the same element! Google: "box model blues" for a detailed explanation.

style.css

```
.main-content {  
    display: flex;  
}  
.left-column {  
    background: hsl(30, 86%, 86%);  
    padding: 0.4em;  
}  
.center-column {  
    padding: 0.4em;  
}  
.right-column {  
    background: hsl(30, 86%, 86%);  
    padding: 0.4em;  
}
```

style.css

```
.left-column {  
    background: hsl(30, 86%, 86%);  
    padding: 0.4em;  
    flex: 0 0 205px;  
}  
.center-column {  
    padding: 0.4em;  
    flex: 1 1 auto;  
}  
.right-column {  
    background: hsl(30, 86%, 86%);  
    padding: 0.4em;  
    flex: 0 0 205px;  
}
```

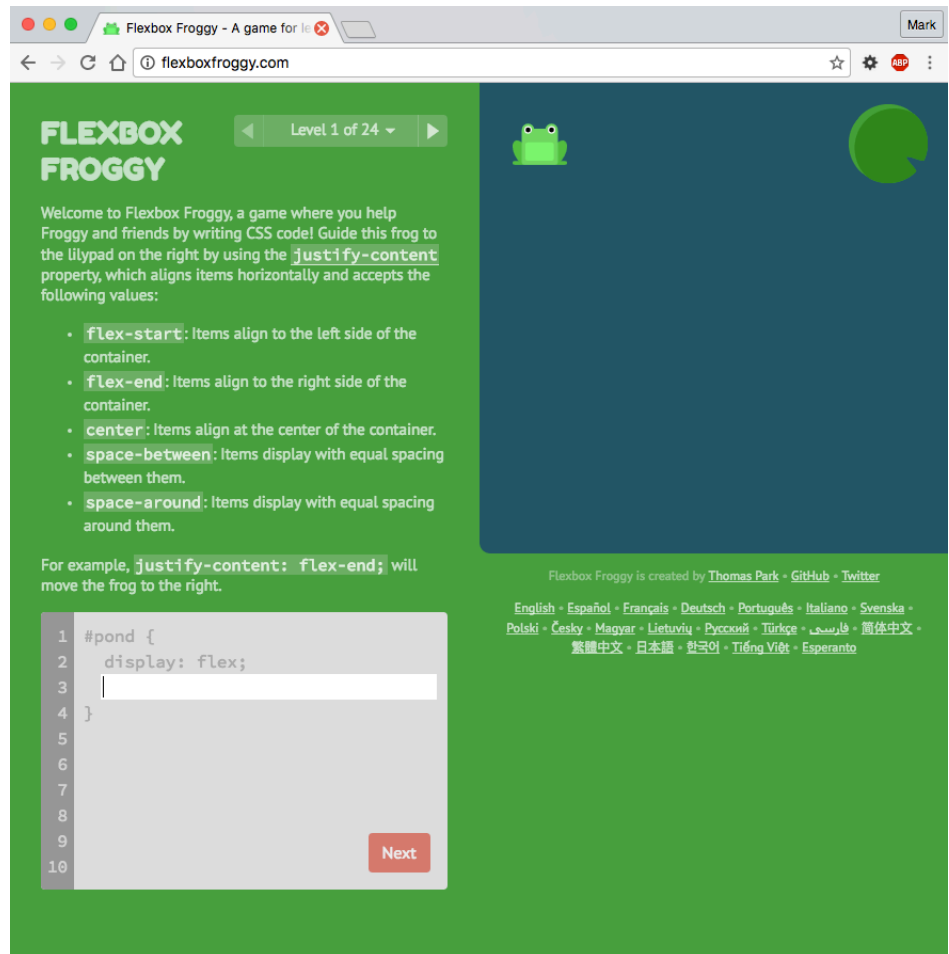


## Flexbox exercises

We will come back to Flexbox later in this book, but if you'd like to explore it in more depth, google: flexbox tutorials. One of the best is this website:

<http://flexboxfroggy.com/>

It is an interactive game written in javascript that allows you to learn all the properties of flexbox.



## Photoshop as a web design tool

**STEP ONE:** Start the **Windows Snipping Tool**. Take a 'picture' with the windows snipping tool of the top of your webpage in the browser as shown to the right. If you are on a Mac...I'm sorry :-) press Command + Control + Shift+4 to take a picture of your browser. Once you 'take a picture' your computers clipboard will hold onto that 'picture' until you copy something else.

**STEP TWO:** Start Photoshop. Choose file>new. Note that Photoshop examines the clipboard everytime you start a new document. If there is a raster image on the clipboard, it uses the pixel measurements of that raster image as the width and height, and tells you it is basing that on the clipboard. Smart software!



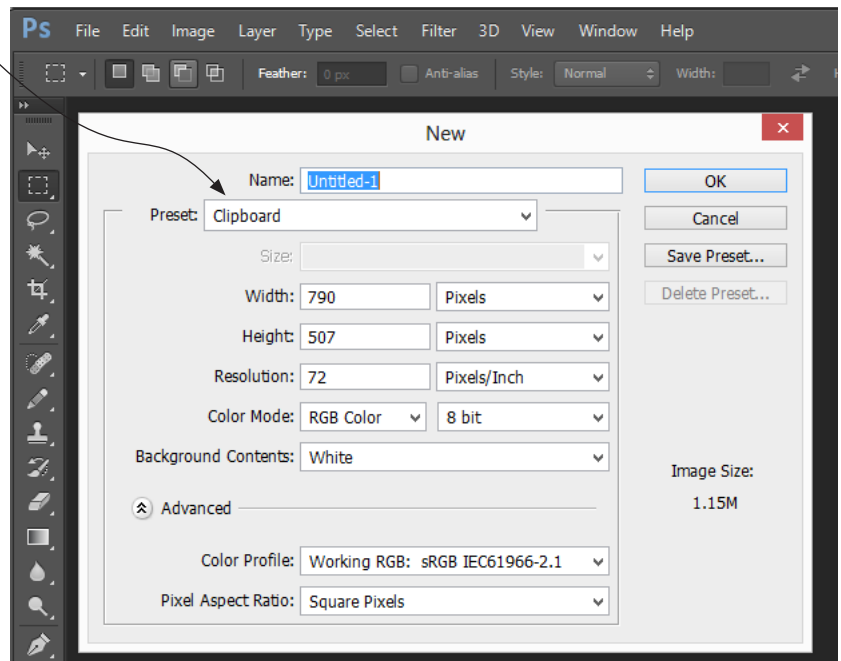
**STEP THREE:** Click ok in Photoshop and then choose edit>paste. You should see your webpage in Photoshop

**STEP FOUR:** Make a new layer at the top of the layers palette named 'content white fill'

**STEP FIVE:** Choose the rectangular marquee tool. Drag out a rectangular selection that covers up the 3 columns of text.

**STEP SIX:** Make white your foreground color and pour white into the selection to cover up the text.

Keyboard shortcut for "pour with foreground color" is Alt+Backspace



## Why are we doing all this?

We are mocking up this interface so we can **visually** experiment with some appearance options for the columns in our content area. By using Photoshop to slice background tiles, we can get total control of things like double strokes, that aren't yet possible in CSS3. With a little slicing work we can get our web page columns to look exactly like they do here in Photoshop. And you can pick any color combination you want. Just remember there will be text on top of the columns.



## Mocking up columns

After pouring your white paint over the column text Photoshop should look like this.

I **apologize** for the different colored backgrounds in these screenshots. This book has been under development for 4 years, and some of the screenshots were from before when I favored gray, while on the newer ones I favored browns.

**STEP ONE:** Choose the rectangular marquee tool. Set the style to Fixed Size and enter a width of 205px and a height of 300px.

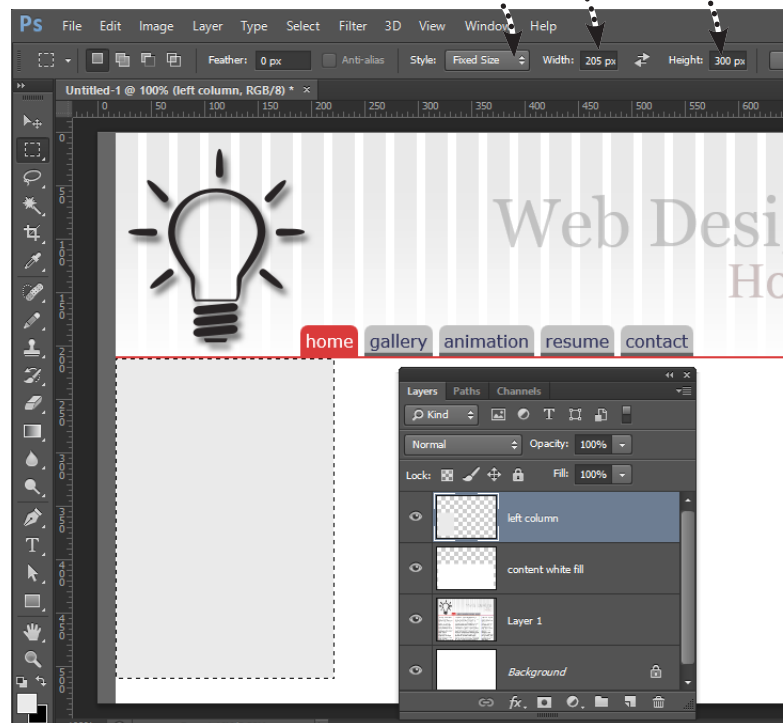
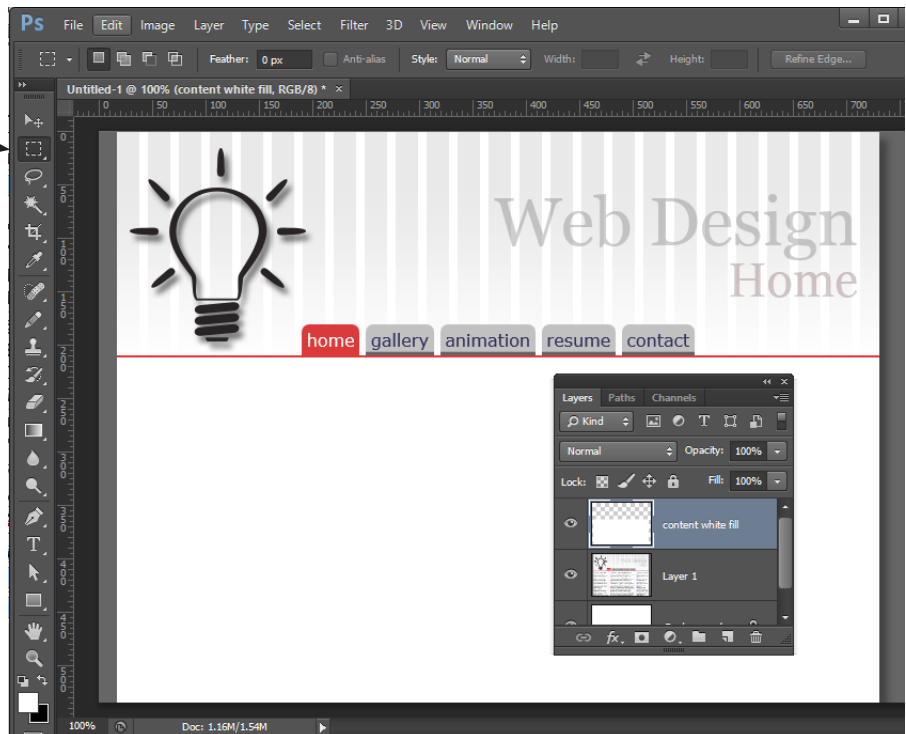
**STEP TWO:** Add a new layer named **left column**.

**STEP THREE:** Click with the marquee tool in the top left corner of the content area, right below the red line. Photoshop should draw a marching ants selection box 205 by 300. If the box doesn't quite line up, put your mouse over the middle of the box and nudge it into position with the arrow keys on your keyboard.

**STEP FOUR:** Pick a pretty color in the foreground color picker and pour that color into the selection on the left column layer by pressing **alt+backspace**. Choose **select>deselect** to get rid of the marching ant selection.

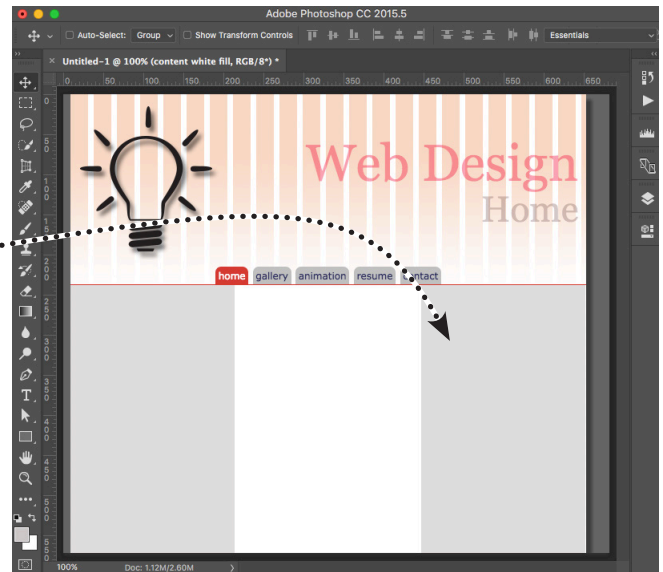
**NOTE:** I picked a color from the bars up by the lightbulb, but you can use any color you like, as long as it is very pale. This will become the background color for our column, and it may have black text over it, so light pastel colors are best. If you choose to use dark colors, you will have to tell your column text color to be white. Remember that old usability rule:

There are no books in the library with white ink on black paper.



## Eye candy columns

**STEP ONE:** Repeat the last series of steps to create a right column: same width: 205px wide. As you did before, **make a new layer** named **right column**, align the selection box to the top right hand corner of the content area and pour in the same color you used on the left side. Deselect!



**STEP TWO:** Make a new layer named **left column pinline**.

**STEP THREE:** Take the pencil tool. Choose one pixel as the thickness of the brush tip on the pencil tool.

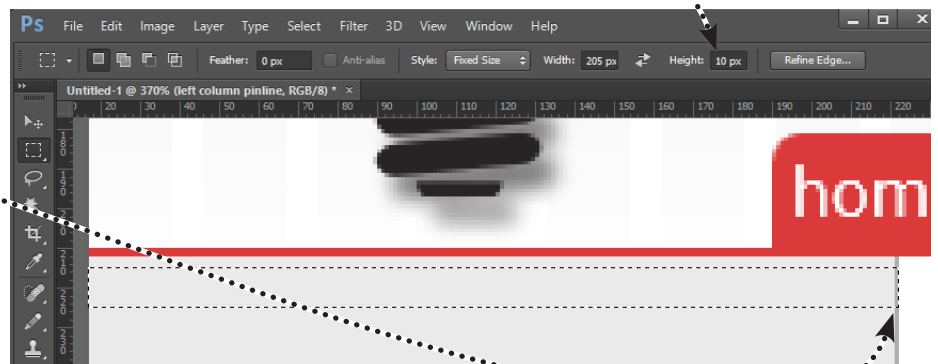
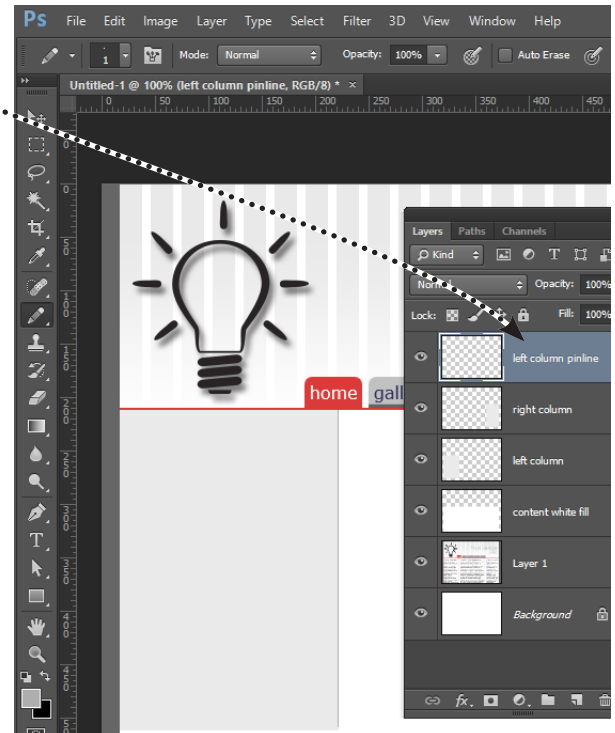
**STEP FOUR:** Pick a slightly darker color than you chose for the pastel columns.

**STEP FIVE:** Position the pencil tool over the top right hand edge of the left column. Press the shift key, then left click and drag the mouse down to the bottom of the column. It should draw a perfectly straight line. If it isn't clean, press Control+z and repeat until you get it. If it isn't in the right place, choose the move tool and move it up and down.

**STEP SIX:** Save the Photoshop file down into your homework folder with a name of interface-mockup-john-smith.psd, if that's you.

**STEP SEVEN:** Choose the rectangular marquee tool. Set the style to Fixed Size and enter a width of **205px** and a height of **10px**.

**STEP EIGHT:** Click at the top left of the left column. Then hover your mouse over the selection and nudge it around until it just fits, you may have to zoom in to get this just right. It's better if you nudge the selection down away from the red. Make sure you get the entire width of the



## Export left-tile.jpg

**STEP ONE:** Go to Photoshop's menu and choose **image>crop**.

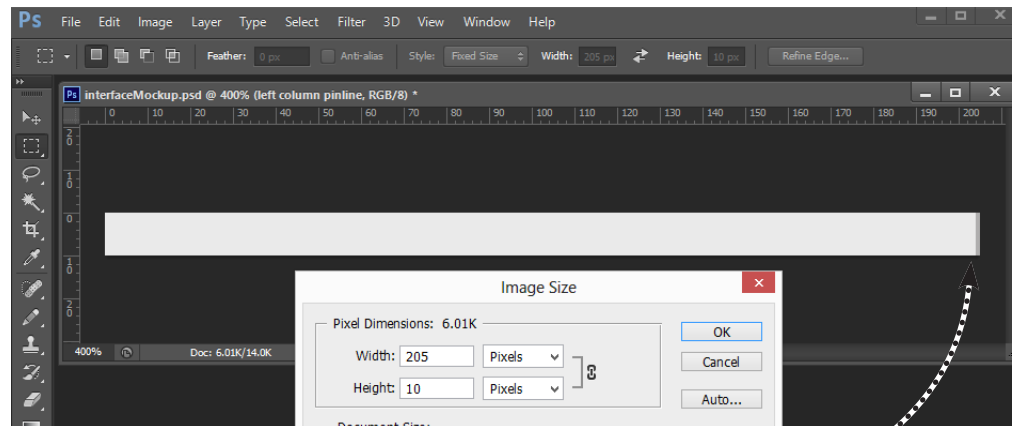
**NOTE:** this tells photoshop to crop, or 'cut away' everything that is outside the selection. There are fancier ways of doing this using the slice tool, but they don't really work any better, and take a lot longer to explain.

**STEP TWO:** Choose:

**Select>Deselect**

Choose:

**image>image size**. The image should measure 205px wide by 10px high. **Click OK**



**STEP THREE:** Check to make sure that you cropped the left column background color with the pinline on the right side. Depending on the pinline color it may be hard to see. It should be subtle, but not so subtle it disappears.

**STEP FOUR:** Choose **file>save for web**. In the Save for Web dialog box, choose a preset of **JPEG High** and set the quality to **85**.

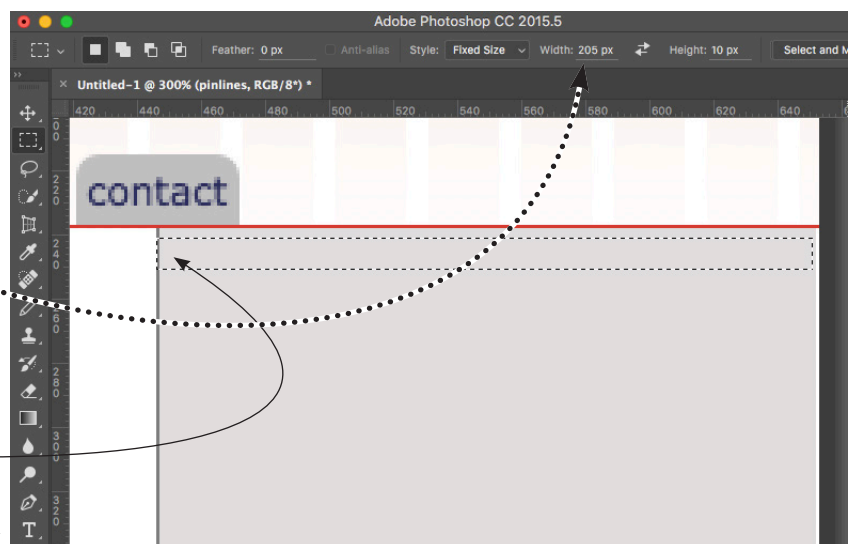
**STEP FIVE:** Click the save button and save the jpg down into your **images folder** on your current website. Give it a name of: **left-tile.jpg**

**STEP SIX:** Back in Photoshop, choose **edit>step backward** repeatedly until you are back to the full webpage mockup.

**STEP SEVEN:** Add a new layer named **right pinline** and draw a pinline down the **left** side of the **right** column. See previous page for instructions.

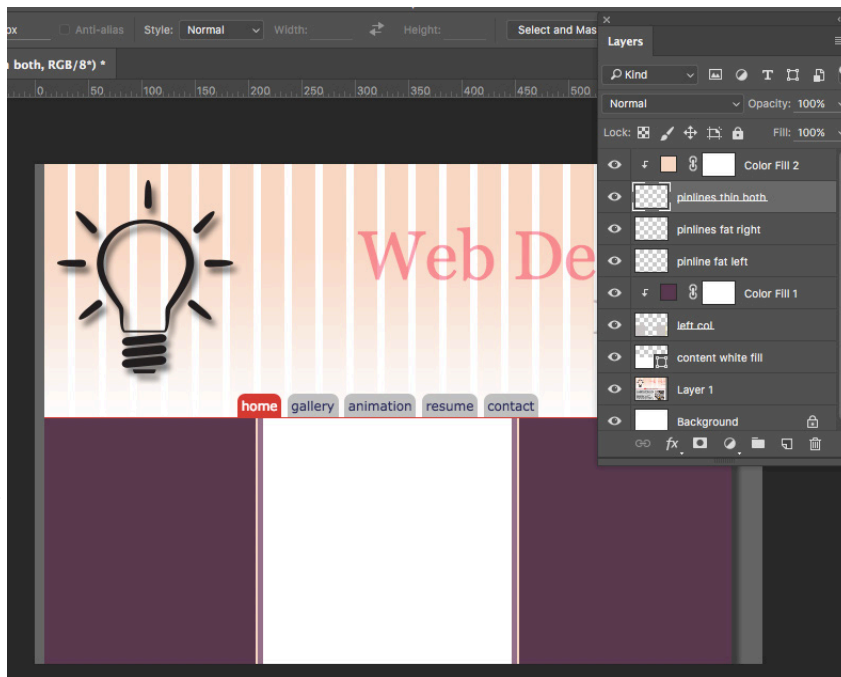
**STEP EIGHT:** Choose the rectangular marquee tool. Set the style to Fixed Size and enter a width of **205px** and a height of **10px**

**STEP NINE:** Click in the right column of the mockup. Photoshop will make the selection. Keep your mouse over the selection and nudge it into position with your keyboard arrow keys. The 205 x 10px selection needs to **exactly** fit the column, or at least the left side of the right column, including the pinline on the left. Zoom in to get more precision.



## Interface variations

If you are artsy, there are a lot of variations you can try with your column colors. I did these double pinlined columns by changing the selection tool back to normal mode. I drew out 5 pixel wide boxes that stretched to the bottom. I picked a pretty color and poured it in on new layer. I did that four times to get what you see here. You don't have to use the pencil tool to draw these, in fact it doesn't work well at anything over 1 pixel. Because these images will tile downward, you can use any colors or pinline thicknesses that you like. You could even try replicating the pattern in a nice dress shirt, as long as it tiles straight down.

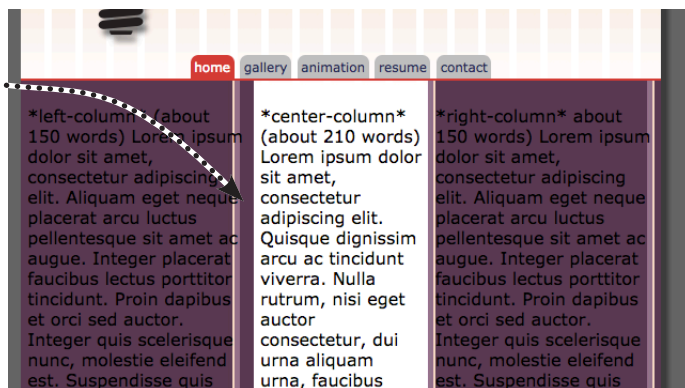


**STEP ONE:** Now that you have exported the two pattern tiles, open style.css and tell them to replace the background color we had in the two columns.

style.css

```
.main-content {
  display: flex;
}
.left-column {
  background: url(images/left-tile.jpg);
  padding: 0.4em;
  flex: 0 0 205px;
}
.center-column {
  padding: 0.4em;
  flex: 1 1 auto;
}
.right-column {
  background: url(images/right-tile.jpg);
  padding: 0.4em;
  flex: 0 0 205px;
}
```

**STEP TWO:** Check it in the browser. Note how the tile images are repeating both vertically and horizontally. The reason for this is that the columns have padding properties that are getting added to the width. The width of each column is actually 205px plus the padding value of 0.4em times 2. This relates back to the old box model blues problem and is another reason why you should **never** put a width and padding property on the same element.



## Export right-tile.jpg, implement in code

**STEP ONE:** The fix is straightforward. Comment out the padding properties in both columns.

**STEP TWO:** Add a new rule that speaks to the paragraph tags inside the columns. Add margin around the `<p>` tags. If needed, change the font color and or size to look pretty over your background image tiles.

Note that I chained the two rules into one style sheet rule using the comma between selectors.

```

style.css

.left-column {
  background: url(images/left-tile.jpg);
  /*padding: 0.4em;*/
  flex: 0 0 205px;
}
.center-column {
  padding: 0.4em;
  flex: 1 1 auto;
}
.right-column {
  background: url(images/right-tile.jpg);
  /*padding: 0.4em;*/
  flex: 0 0 205px;
}

.left-column p,
.right-column p {
  margin: 0.8em;
  color: white; /*color and font size are optional*/
  font-size: 0.9em;
}
  
```

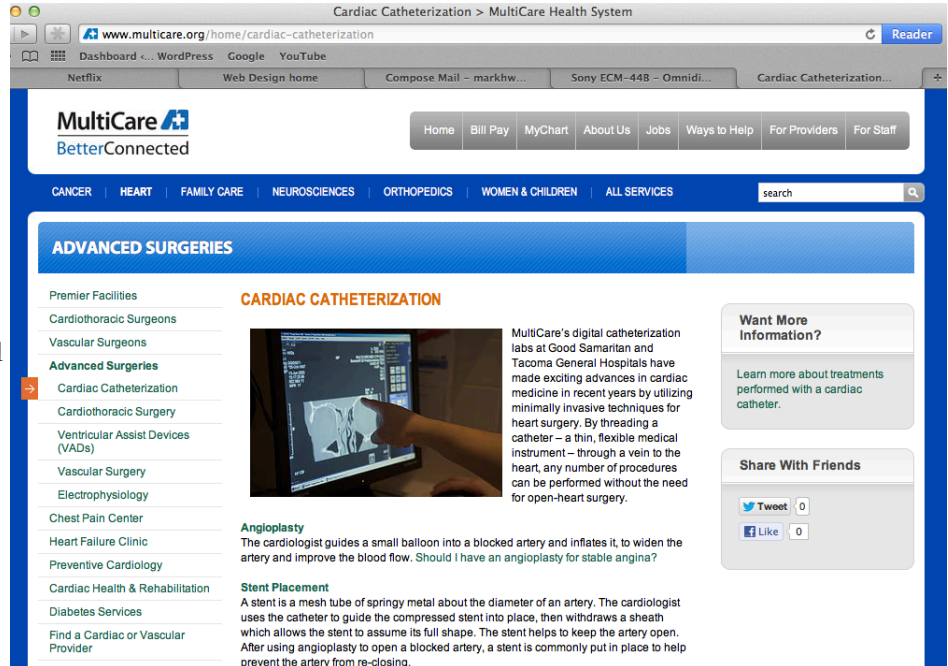
**STEP THREE:** Check it in the browser, it should look like this:





## Round corner boxes

Round corner boxes are a very useful web page interface element. They allow us to place attractively designed advertisements (action boxes) alongside our main content area. By isolating them from the rest of the page content, via the round corner container, we can move those links out of the main navigation links, and if done right, they add pleasing white space to the page, as shown here on the old MultiCare hospital web site. (The site has been redesigned since I took this screenshot) They are usually built to be multipurpose, meaning they are basically a miniature version of a full blown web page interface, meant to expand based on the content that is placed inside. They can hold text, image, headers, anything within reason. Typically they hold at least a header, and some text with a small thumbnail image.



We will build these boxes using the current standards for CSS 3. But we will design them so that if they break in an older browser, our code will degrade gracefully, so the end user can still see, and interact with them.

**STEP ONE:** Clear most of the latin text from the **.right-column**. Above it's first paragraph tag, add a new starting and stopping div tag. Give the div tag a class of **boxRight** as shown in bold brown.

```

index.html

<main class="main-content" role="main">
  <aside class="left-column">
    <p>
      *leftCol* Lorem ipsum dolor sit amet, consectetur
      adipiscing elit, sed do eiusmod tempor...
      (about 150 words)
    </p>
  </aside> <!--end aside.left-column-->
  <article class="center-column">
    <p>
      *centerCol* Lorem ipsum dolor sit amet,
      consectetur adipiscing elit, sed do eiusmod tempor...
      (about *210 words)
    </p>
  </article> <!--end article.center-column -->
  <aside class="right-column">
    <div class="boxRight">
      </div> <!--end boxRight-->
    <p>
      *rightCol* Lorem ipsum dolor sit amet,
      consectetur adipiscing elit, irure dolor in reprehenderit in voluptate velit
      esse cillum dolore eu
      (about 20 words)
    </p>
  </aside> <!--end aside.right-column-->

```



## Add content to the box, initial styling

**STEP ONE:** Inside the new `boxRight` div, add a `h3` header, and a paragraph with two sentences as shown in bold brown.

```

index.html
</article> <!--end end article center-column-->
<aside class="right-column">
  <div class="boxRight">
    <h3>Box Header</h3>
    <p>
      This is a paragraph within the round corner box.
      This text should wrap downward within the box.
    </p>
  </div><!--end box right-->
<p>
  *rightCol* Lorem ipsum dolor sit amet,
  consectetur adipisicing elit,irure dolor in reprehenderit in voluptate velit
  esse cillum dolore eu
  (about 20 words)
</p>
</aside><!--end aside.right-column-->

```

**STEP TWO:** add the two new style sheet rules shown in bold brown:

**.boxRight**

**.boxRight h3**

NOTE: I picked some of these colors in Photoshop. It's easy to mock this up in Photoshop on a couple new layers to experiment with color combinations. Depending on your color scheme, it can take a while to get the colors tweaked just right. Once you get it pretty, copy the hexadecimal colors out of Photoshop and paste them into your code.

**STEP FOUR:** Azzza

```

style.css
.right-column {
  background: url(images/right-tile.jpg);
  /*padding: 0.4em;*/
  flex: 0 0 205px;
}
.left-column p,
.right-column p {
  margin: 0.8em;
  color: white; /*color and font size are optional*/
  font-size: 0.9em;
}
.boxRight {
  width: 175px;
  background-color: #dfdede;
  border: 1px solid #adadad;
  margin-top: 20px;
  margin-left: 7px;
}
.boxRight h3 {
  text-align: center;
  font: 1.3em Georgia, Times, serif;
  color: #fff;
  padding: 3px 0 3px; /**3px top (0 rt & left) 3px bott.**/
  margin: 0;
  background-color: #666; /**666 is short for 666666**/
}

```

## Code that degrades gracefully

**STEP ONE:** Save the page and view it in your browser. This is how the box will look in older browsers that don't allow round corners in CSS3. It will still function...it **degrades gracefully**. Now let's do the fancy stuff.

**STEP TWO:** One last thing, the paragraph text is too close to the walls of the box. Add this little style sheet rule right below the last one:

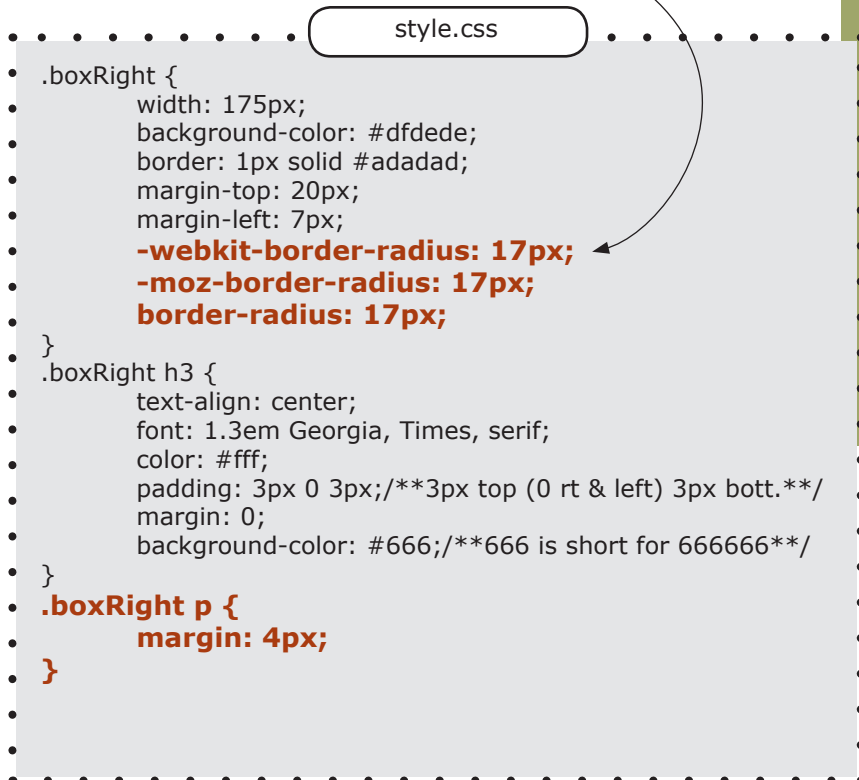
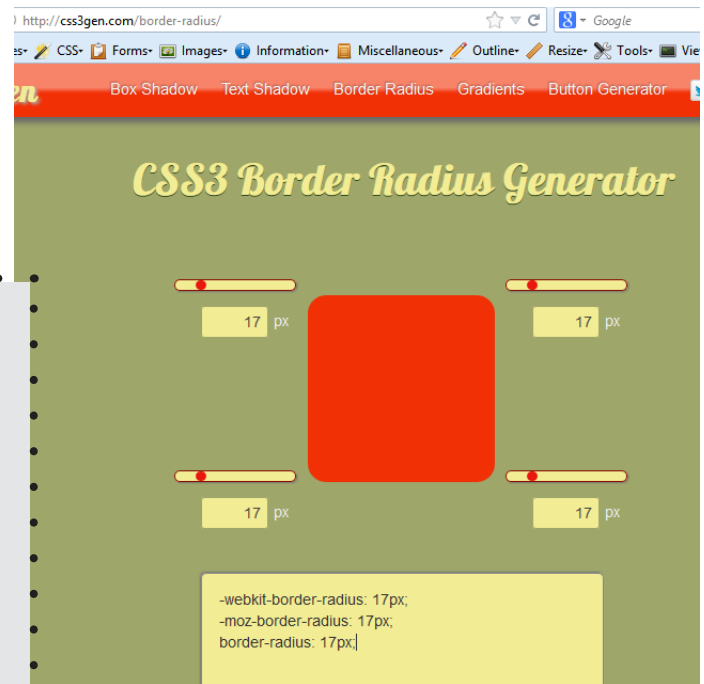
```
.boxRight p {  
    margin: 4px;  
}
```

**NOTE:** in case you are rusty at your CSS language, that selector means that if there is an element on the page that has a class of **boxRight**, and there is a paragraph tag inside the **boxRight**, give the words inside the **<p>** tag 4 pixels of margin on all 4 sides.

**STEP THREE:** Go out to this website: <http://css3gen.com/border-radius/>

**STEP FOUR:** Set all 4 sliders at 17 and copy the 3 lines of code.

**STEP FIVE:** Paste the code into the bottom of the **.boxRight** rule as shown in bold brown



## Round em up!

If you check this in your browser now you will see that the boxRight div has round corners, but it's child the <h3> header does not. So much for inheritance!

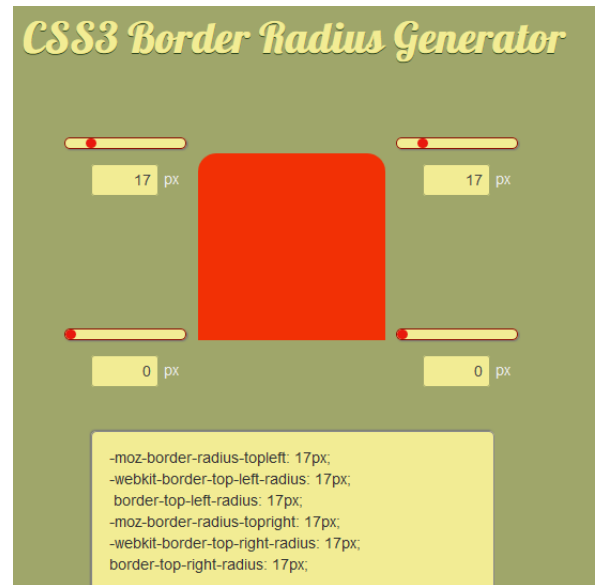
The fix is easy.

**STEP ONE:** Go back to the web page and tell it to have 17px round corners on top, but no round corners (0) on the bottom. Copy the code.

**STEP TWO:** Locate your .boxRight h3 rule. Paste the new code into the bottom as shown in **bold brown**.

NOTE: when you bring in this much weird code at once, it's wise to put it inside starting and stopping CSS comment tags...again, as shown.

Also, note that when you only want the top corners round, it takes more code. Because instead of saying border-radius, which makes all 4 corners round, you have to say border-radius-topleft. Study the code, there is logic there if you stare at it for a while.



style.css

```
.boxRight {
    width: 175px;
    background-color: #dfdede;
    border: 1px solid #adadad;
    margin-top: 20px;
    margin-left: 7px;
    -webkit-border-radius: 17px;
    -moz-border-radius: 17px;
    border-radius: 17px;
}

.boxRight h3 {
    text-align: center;
    font: 1.3em Georgia, Times, serif;
    color: #fff;
    padding: 3px 0 3px; /*3px top (0 rt & left) 3px bott.**/
    margin: 0;
    background-color: #666; /*666 is short for 666666**/
    /*begin round corner code*/
    -moz-border-radius-topleft: 17px;
    -webkit-border-top-left-radius: 17px;
    border-top-left-radius: 17px;
    -moz-border-radius-topright: 17px;
    -webkit-border-top-right-radius: 17px;
    border-top-right-radius: 17px;
    /*end round corner code*/
}

.boxRight p {
    margin: 4px;
}
```

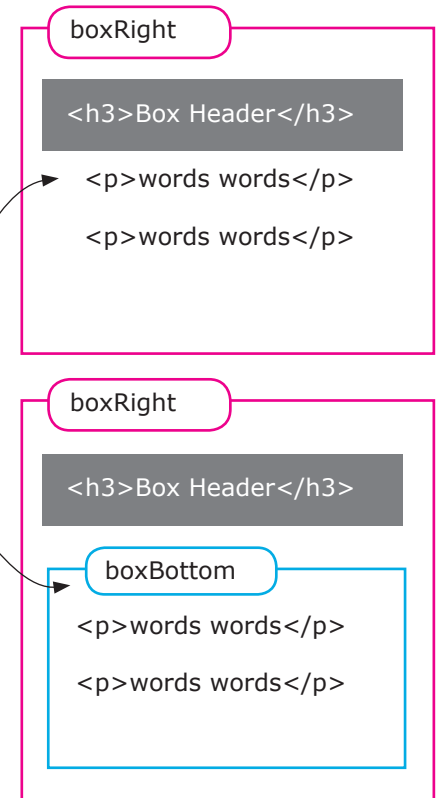


## Nesting colored boxes

The round corner box is designed to be modular. This means that it will be used multiple times on the website with varying amounts of text in both the top and bottom of the box. Both top and bottom of the box will stretch to accommodate any amount of content. Our box will do that right now. Try it and see.

However, if we want to make it pretty with unique vertical linear gradients on both the top and bottom of the box (as on the Multicare website), we need to have separate box model elements for top and bottom. We can use the `<h3>` tag as the top of the box, but the bottom of the box is not defined. The bottom is simply what remains of the `<div class="boxRight">` after the `<h3></h3>` displays.

To get control of the colors (gradients) in the bottom half of the box, we need to nest in a new div called **boxBottom**. It needs to enclose the existing paragraph tags.



**STEP ONE:** On your `index.html` page, find your `<div class="boxRight">` tag. Immediately under the `<h3>Box Header</h3>` tag, start a new div:

```
<div class="boxBottom">
```

**STEP TWO:** Stop the div tag **between** the last paragraph tag and the ending div tag for `boxRight`. Code is shown in bold brown to the right.

This new `boxBottom` div will give us a place to hang our eye candy for the background of the bottom of the box.

index.html

```

<div class="boxRight">
  <h3>Box Header</h3>
  <div class="boxBottom">
  <p>
    This is a paragraph within the round corner box. This text should
    wrap downward within the box.
  </p>
  </div><!--end boxBottom-->
</div><!--end boxRight-->
<p>
  *rightCol* Lorem ipsum dolor sit amet,
  consectetur adipisicing elit, irure dolor in reprehenderit in voluptate velit esse
  cillum dolore eu
  (about 20 words)
</p>

```

## Eye candy gradient boxes

We will add the gradients to the top of the box first, which means we will apply them to the `<h3>` tag as background colors.

This is the website we will use to generate our gradients:

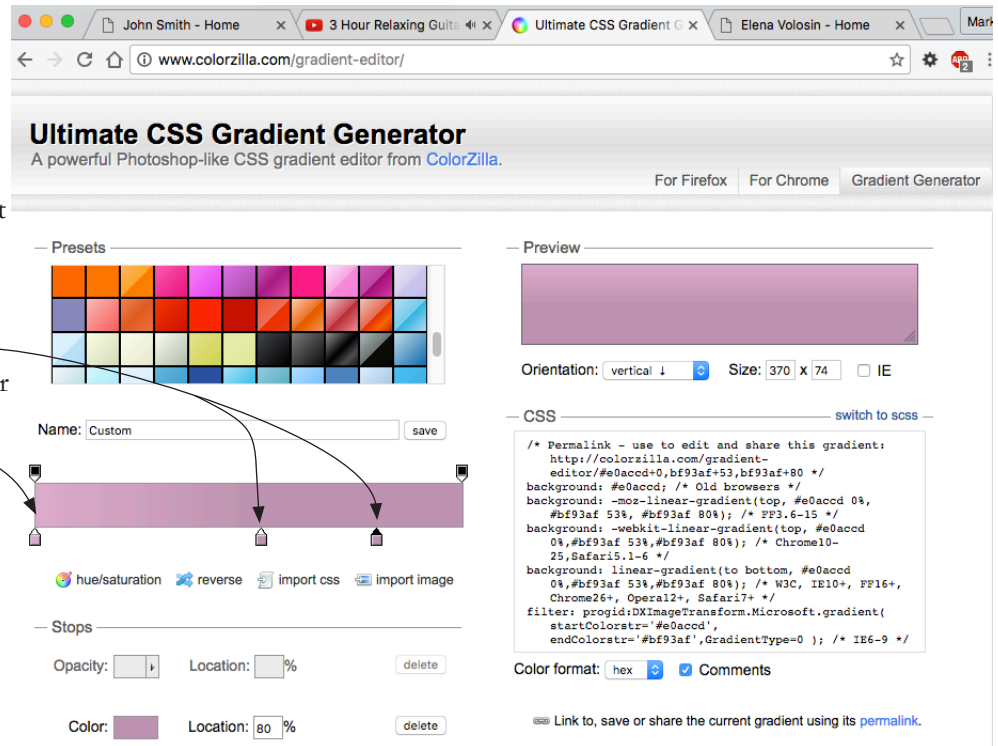
<http://www.colorzilla.com/gradient-editor/>

**STEP ONE:** I'd like the top of the box to have a slightly rounded appearance, as if light was shining down on it from above. I'd also like the top of the box to be expandable, capable of holding two lines of text...you can never predict what your client will want to stick in there.

**STEP TWO:** For the two tabs on the right, pick the exact same color. But for the tab on the left, pick a lighter color.

NOTE: to remove a tab, pull straight down. To add a tab, click where you want it. To edit the color on a tab, double click it. When you get a color you like, hover your mouse over the lower right corner of the CSS window and press the **copy** button.

**STEP THREE:** Switch to style.css. Find the `.boxRight h3` rule and paste in the new code from [www.colorzilla.com](http://www.colorzilla.com)

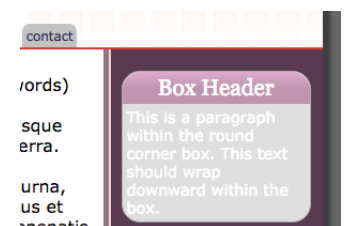


```

152 .boxRight h3 {
153     text-align: center;
154     font: 1.3em Georgia, Times, serif;
155     color: #fff;
156     padding: 3px 0 3px; /**3px top (0 rt & left) 3px bott.**/
157     margin: 0;
158     background-color: #666; /**666 is short for 666666**/
159     /* Permalink - use to edit and share this gradient: http://colorzilla.com/grad
160     background: #e0accd; /* Old browsers */
161     background: -moz-linear-gradient(top, #e0accd 0%, #bf93af 53%, #bf93af 80%);
162     background: -webkit-linear-gradient(top, #e0accd 0%, #bf93af 53%, #bf93af 80%);
163     background: linear-gradient(to bottom, #e0accd 0%, #bf93af 53%, #bf93af 80%);
164     filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#e0accd', en
165     /*begin round corner code*/
166     -moz-border-radius-topleft: 17px;
167     -webkit-border-top-left-radius: 17px;
168     border-top-left-radius: 17px;
169     -moz-border-radius-topright: 17px;
170     -webkit-border-top-right-radius: 17px;
171     border-top-right-radius: 17px;
172     /*end round corner code*/
173 }
174 .boxRight p {
175     margin: 4px;
176 }

```

**STEP FOUR:** Refresh and you should get this. Please use different colors than I am using here. Mine isn't pretty right now because I'm focused on writing, and not making it mona lisa pretty.



## Bottom of the box

**STEP ONE:** Add the two new rules shown to the right in brown.

NOTE: The border top and bottom properties are both hacks. Without them the `.boxBottom` has margins above and below. I didn't want a border on the bottom, but it was needed for fit, so I made it invisible. <http://hslpicker.com/#fff,0>

The border radius settings don't have browser prefixes because quite frankly they aren't needed anymore in 2016.

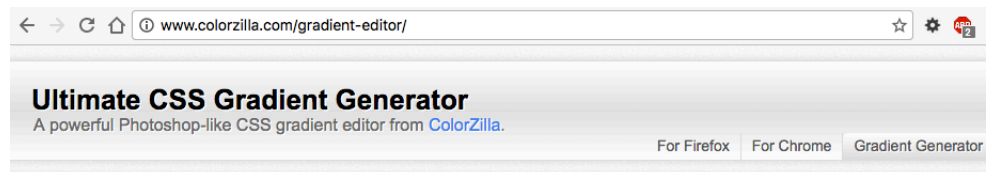
The long selector on the `.boxBottom p` is needed because I had some conflicts in previous style sheets and wanted to be very specific to make sure it answered.

```
style.css

.boxBottom {
  border-top: 1px solid black;
  border-bottom: 1px solid hsla(0, 0%, 100%, 0);
  border-bottom-left-radius: 17px;
  border-bottom-right-radius: 17px;
}

.right-column .boxRight .boxBottom p {
  color: black;
}

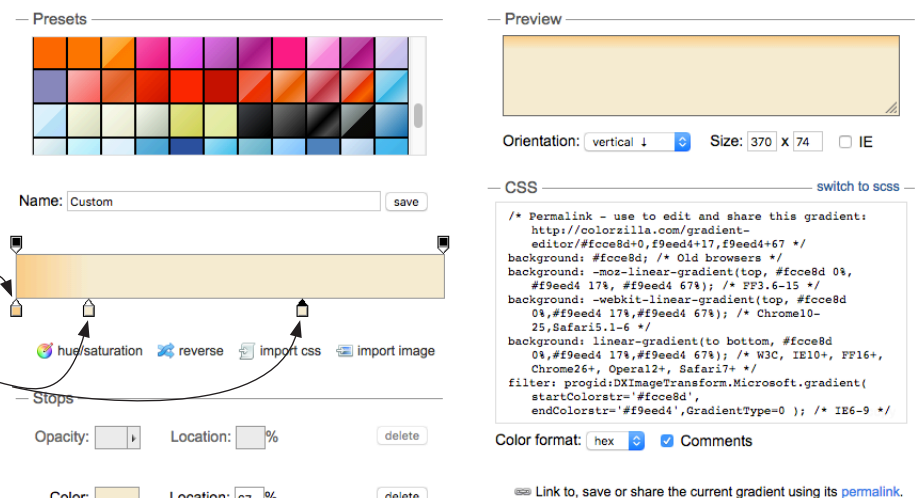
.boxRight p {
  margin: 4px;
}
```



**STEP TWO:** Return to colorzilla and generate the gradient you'd like to have on the bottom of the box. My concept this time was that the top of the box cast a shadow over the bottom, which quickly dissipated.

**STEP THREE:** This tab is the darker shadow color.

**STEP FOUR:** These two tabs need to be lighter, but the same (lighter) color, and the position of the tabs does matter. I want to be able to pour words into this box and have the cast shadow stay relatively consistent as the gradient gets taller.



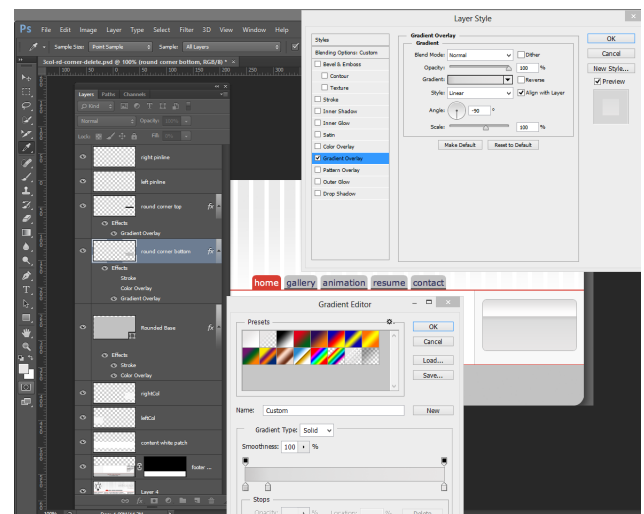
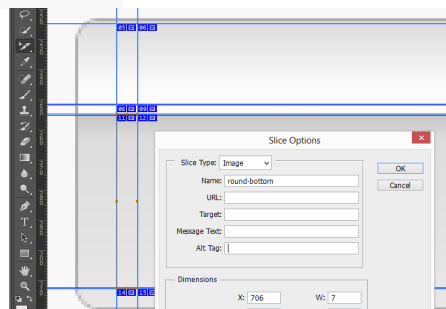


```

174 .boxBottom {
175     border-top: 1px solid black;
176     border-bottom: 1px solid hsla(0, 0%, 100%, 0);
177     /* Permalink - use to edit and share this gradient: http://colorzilla.com/gradient-editor/#f99e4d+0%2cf99e4d+100% */
178     background: #f99e4d; /* Old browsers */
179     background: -moz-linear-gradient(top, #f99e4d 0%, #f99e4d 17%, #f99e4d 67%);
180     background: -webkit-linear-gradient(top, #f99e4d 0%,#f99e4d 17%,#f99e4d 67%);
181     background: linear-gradient(to bottom, #f99e4d 0%,#f99e4d 17%,#f99e4d 67%); /*
182     filter: progid:DXImageTransform.Microsoft.Gradient( startColorstr='#f99e4d', endColorstr=
183     border-bottom-left-radius: 17px;
184     border-bottom-right-radius: 17px;
185 }
186 .right-column .boxRight .boxBottom p {
187     color: black;
188 }
189 .boxRight p {
190     margin: 4px;
191 }

```

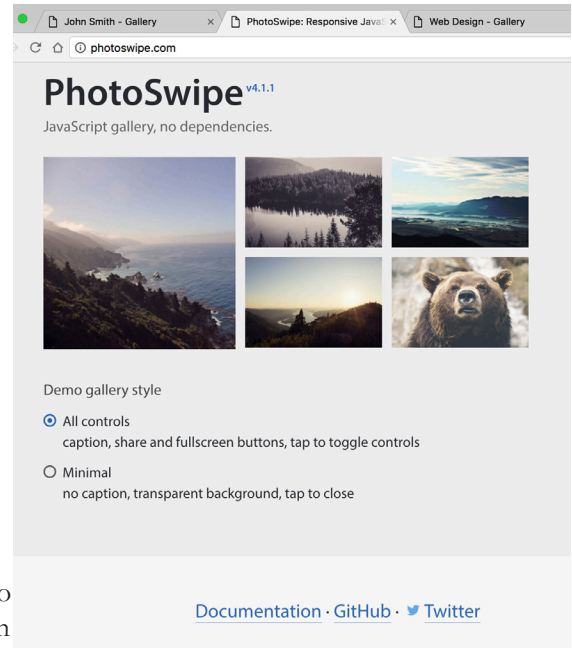
NOTE: you do not have to make your gradients in Photoshop, but you should be aware that it is still very workable, and you may run across them while working on legacy websites.



## Responsive Gallery

Now that you've learned how to make columns you may notice that they do not respond well on smartphones. I have a fix for that coming up, but perhaps a more pressing problem is the gallery we built with slimbox back on pages 76 through 80. Our slimbox gallery animates beautifully on computers, but does not animate at all on smartphones. Over the next pages we will implement a smartphone mobile friendly gallery function that works so well it replicates the native gallery app on your smartphone, complete with swipe gestures. It also allows the same full screen slideshow experience on your computer. It was developed over the last few years by a series of programmers and is now freely available to the public. You can also buy it as a plugin if you have a Wordpress site, but we will be using the free version.

Because it was developed by programmers, and given out freely, there are very few fully explained tutorials on the internet. Those guys always assume you are a hard core programmer and can fill in the missing steps and logic. So without further ado, here is my explanation of how to use photoswipe, which you can view here on both your computer and your smartphone:



<http://photoswipe.com/>

The first thing we need to do is prepare the images in Photoshop. Because this can be a repetitive process, (dare I say boring?) we are going to automate the image preparation using something called Photoshop Actions. As before, we will need a thumbnail and a big image for each picture in the gallery. But because this is a much better slideshow function, we will make the pictures bigger, as in 1080 pixels tall. We don't need to worry about whether they will fit on the screen of the various devices that view the pictures because the javascript will shrink them as needed. The caption will be overlaid on the pictures. This means we don't need to shorten the images to make room for the caption as we did on slimbox. If you haven't already checked out photoswipe.com, do so now. You will notice that the caption disappears if you don't move your mouse for 3 seconds, or if you tap the picture, as on a smartphone.

As before, for this process to make sense, **your images will need to be large**, preferably right out of your camera. Because we will be shrinking them down to 1080 pixels high, they need to have a height larger than that. This automated action that we will be setting up is going to automatically size them down to 1080 high. If you choose the wrong images, as in something like 600 pixels tall, Photoshop will upsize them to 1080px, and that will make them very fuzzy. Photoshop can shrink images without a loss in quality, but it cannot make them larger than they are. Think of the way we handle money: you can shrink a dollar down to fifty cents, but you can't stretch a dollar to make a dollar fifty. Pixels are the same way. You can shrink them, but you can't stretch them.

**STEP ONE:** Grab some large (over 5 megabyte) images. You can use the same ones you used before, but you might want to use different images to keep the confusion down first time through. I have provided some of my high res pictures for you practice with. They are in the **resources/big-images** folder provided with this book

# Preparing photoswipe images

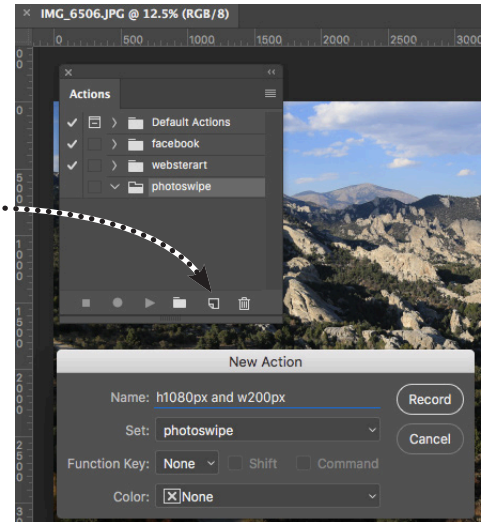
**STEP ONE:** Open Photoshop, open a large image, 5 megabytes or larger.



**STEP TWO:** Open the **Actions** panel and click the **new set** button at the bottom of the panel. Name the set: **photoswipe** and click OK.

**STEP THREE:** Minimize Photoshop. Use Windows (or the Mac Finder) to navigate down into your current working website's **images** folder. Make a new folder inside your images folder named: **h1080**

NOTE: We had to make this folder beforehand because this automated action we are setting up is going to need to save into that folder, so it must exist before we record the action of saving into it.



**STEP FOUR:** Back in Photoshop, click the **Create new action** button. Name the action **h1080px and w200px** then click **Record**.

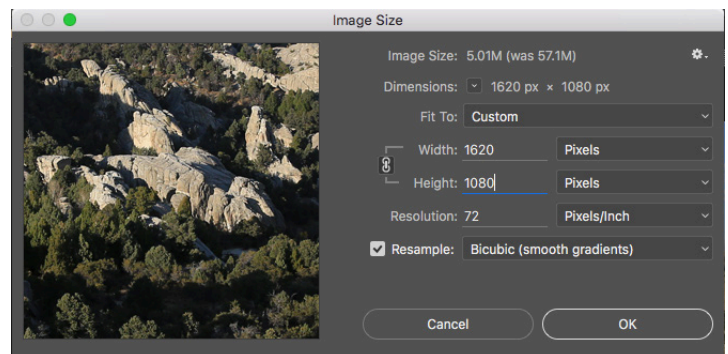
**NOTE:** Photoshop is now recording everything you do. You will see a little red light at the bottom of the actions panel indicating it is recording.

**STEP FIVE:** Choose Image > Image Size

**STEP SIX:** Enter 1080 Pixels for the height.

**STEP SEVEN:** Ensure that Resample is checked, and that the type of resampling is Bicubic (smooth gradients). This preserves quality. Resolution doesn't matter in web design, so ignore that.

**STEP EIGHT:** Click OK





## Record an Action

**STEP ONE:** zoom into 100% magnification (Ctrl/Command + 1). Your picture should still be nice and sharp. Observe what is happening in the Actions panel. Photoshop recorded your trip through the Image > Image Size process.

Imagine if you had a robot at home that could record your housework. You could press the Record button on the robot. It would follow around all evening. It would record you dumping the garbage, sweeping the floor, cooking dinner, washing dishes and walking the dog. Once you told the robot to stop recording, it would then have that entire process memorized.

Next time you walk in the door after work, you simply select the recorded action and press play. It would scurry around doing all of your chores for you, but at lighting speed, because robots are smarter and faster than we are. You could relax while the robot did all the work, secure in the knowledge that it would finish everything perfectly.

Now, you might get tired of having the exact same meal everynight, and if the robot couldn't find the same steak in the fridge that you used, it would not be able to make dinner, and would likely blow a fuse.

So enough with the analogies, let's get back to work recording.

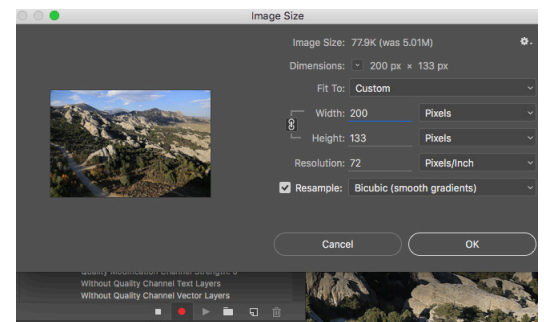
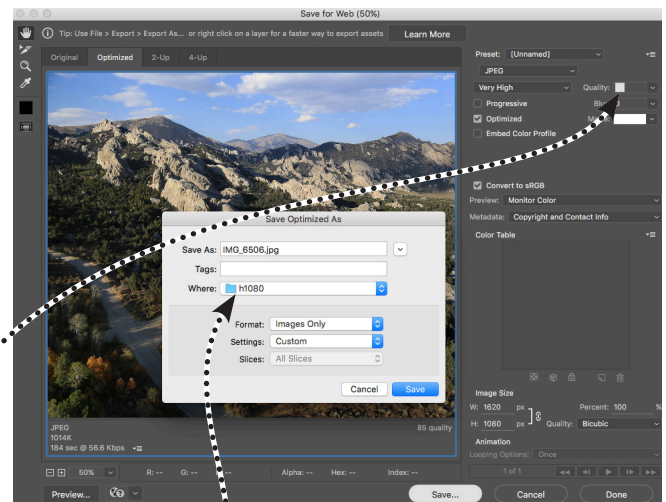
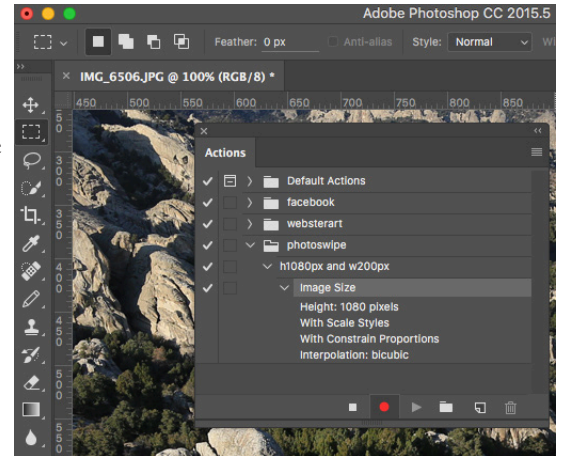
**STEP TWO:** Choose **file > export > save for web legacy**

**STEP THREE:** Choose a Preset value of **JPEG High**, with a quality setting of at least 85. NOTE, if you have a ton of images, lowering the quality down into the 60's makes your page load faster. Make sure the height is at **1080**, it should be because you set it there in the previous step.

**STEP FOUR:** Click the **save** button and navigate down into your **images/h1080** folder. **DO NOT change the name!** If you accidentally change the name, that will get recorded into the action and you will have to discard the recorded Action and start again. Note that Photoshop recorded the export process, including your quality settings and the path to the h1080 folder on your computer. It's recording everything you do. It doesn't record anything outside the Photoshop application, like when you go check your email, but it watches like a hawk when you do things inside Photoshop.

**STEP FIVE:** Choose image > image size again. Set the width to 200 pixels and click OK. This latest change has created the thumbnail.

**STEP SIX:** Choose **file > export > save for web legacy**. Use all the same settings (don't change the name!) but this time save it to a different directory. Save it to the root of your **images** folder.



## Actions concluded

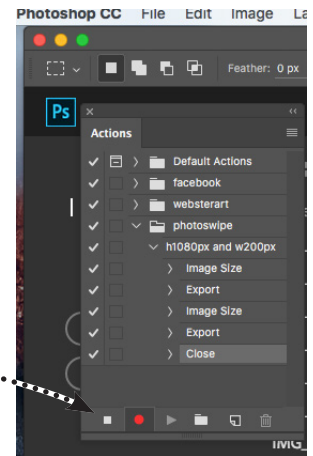
**STEP ONE:** Close the file and don't save the changes. You may need to choose **window > action** to turn the actions panel back on, but it should still be recording your every move.

**STEP TWO:** Press the **stop** button on the actions panel to stop the recording.

**NOTE:** If you have the five action steps I show here, you have a perfect recorded action. They are: Image size; Export; Image Size; Export; Close

If it didn't work, touch the action (the line with h1080...), press the garbage can icon. Make sure you have the image open in Photoshop before you start recording again. It is common to make recording errors, keep trying until you get a clean one.

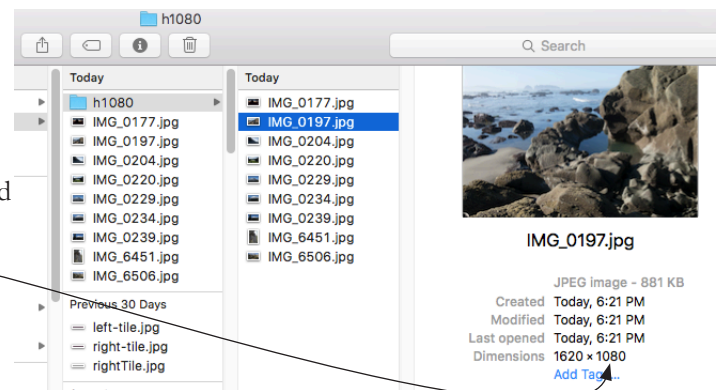
**STEP THREE:** To use your shiny new Action, fold it up until all you see is the line with **h1080px and w200px**. Click on that line, it will turn gray indicating it is the selected action.



**STEP FOUR:** Open another 5 megabyte image. Press the **play** button on the Actions panel. Did you see it happen? Photoshop runs through the 5 steps in the blink of an eye. Look in your **images** folders, including the subfolder called **h1080**. You should see the files in there. Touch them and check their measurements using either Windows or the Finder on the Mac. They should measure 1080px **tall** in the h1080 folder, and 200px **wide** in the images folder.

**STEP FIVE:** Run the action on at least 6 big images. It will work on RAW files too, but you first have to get them open in Photoshop, meaning, past the RAW plugin.

**NOTE:** It is possible to record the settings you choose in the RAW plugin as part of the Action, but this complicates the process considerably. For instructions, do a google search for: **batch convert raw files to jpegs using actions**, or, if you are in my college class, ask me in lab.



## Programming photoswipe

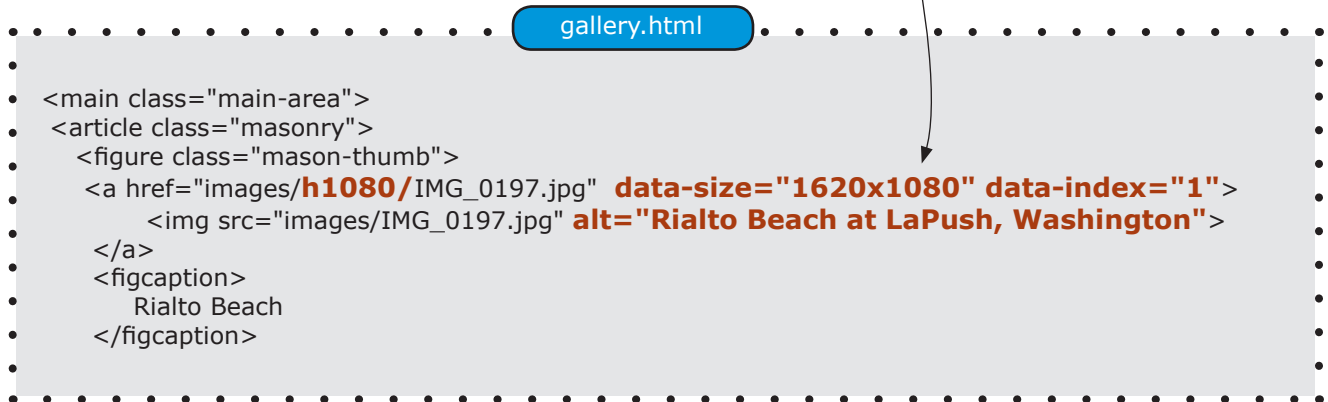
**STEP ONE:** Back up your entire site to your flash drive before going to the next step. We will be making big changes to the gallery.html page. There is a small chance you won't like them, and a bigger chance you will break the page so badly it would be easier to start fresh from your backup.

**STEP TWO:** open **gallery.html** in your code editor of choice. Once again, I recommend Dreamweaver because it will give you shortcut buttons to the imported javascript files (libraries). If you choose to use one of the other editors, be sure to preview the page in Firefox. Then, if/when it doesn't work, view the source in Firefox and click the blue underlined links to the imported javascript files. If they can't be found, you have a simple file management problem.

**STEP THREE:** Find your first `<figure></figure>` element and copy it. Then paste it in right below the starting `<article class="masonry">` tag.

**STEP FOUR:** Remove the `rel=""` and `title=""` properties from the starting anchor `<a>` tag.

**STEP FIVE:** Replace them with `data-size=""` and `data-index=""` properties as shown below. You will have to manually get the width and height properties of the image by examining it in Windows explorer, or Finder on a Mac. Note also that I am specifically going into the `images/h1080` folder to get the bigger image. The `data-index` property simply puts numbers (pic 1, pic 2, etc) in the caption of the gallery slider. This is the first image, so it gets number 1. You must have an `alt=""` property in the `<img>` tag. In one of the external \*.js files, I have written a custom javascript function that converts the words in the alt property into the picture caption on the slider.



**STEP SIX:** Once you have edited this figure tag, and triple checked it for errors, copy it and paste it in below this existing figure tag. Then edit it again for the next new image you'd like to use. Don't forget to change the `data-size` and `data-index`. I recommend using at least 6 new images first time through to cut down on confusion. It would be a good idea to cut out all your old slimbox figure tags to a notepad file, and save them there for later. Once you get this running (animating) with new images, you will be better prepared to get your old images working. For those of you who are thinking ahead, note that I did not change the size or location of the thumbnails, I simply moved the big images into a subfolder called `h1080`. This allowed me to keep the file name the same in the Photoshop Action



# Photoswipe

**STEP ONE:** Examine my screenshot. This shows 6 figure tags ready to be animated. Note that the **data-index** properties start at 1 and increment up to 6. It's ok if the width and height of the of the big pictures varies from what you see here. For example, you could choose to use pictures with a width of 600 and a height of 800 to make them load faster. Just make sure you enter the exact values in the **data-size=""** property. You can even mix landscape and portrait images. The script doesn't care, just give it the proper dimensions of the large file. The thumbnails must remain exactly 200 pixels wide or you could run into trouble with the masonry effect.

**STEP TWO:** As we did with the javascript animation banner, we are going to need to copy and paste a bunch of files and code. Go to the resources folder for this book and copy the folder named **dist**

**STEP THREE:** Paste that **dist** folder at the **root** of your current site. It should be at the same level as the html pages.

**STEP FOUR:** Go copy the gallery-photoswipe.html file from the resources folder into the root of your current site.

**STEP FIVE:** Open gallery-photoswipe.html in your code editor.

**STEP SIX:** Copy the two links to external style sheets from the head over to the head of your gallery.html file. And remove the link to the jquery.js file, as well as the two links to slimbox files, both css and js.

**STEP SEVEN:** Check that your head looks exactly like this:

```

33
34 <main class="main-area">
35 <article class="masonry">
36 <figure class="mason-thumb">
37 <a href="images/h1080/IMG_0197.jpg" data-size="1620x1080" data-index="1">
38 
39 </a>
40 <figcaption>
41 Low Tide
42 </figcaption>
43 </figure>
44
45 <figure class="mason-thumb">
46 <a href="images/h1080/IMG_0177.jpg" data-size="1620x1080" data-index="2">
47 
48 </a>
49 <figcaption>
50 Sunset at Rialto
51 </figcaption>
52 </figure>
53
54 <figure class="mason-thumb">
55 <a href="images/h1080/IMG_0204.jpg" data-size="1620x1080" data-index="3">
56 
57 </a>
58 <figcaption>
59 Beach sunrise
60 </figcaption>
61 </figure>
62
63 <figure class="mason-thumb">
64 <a href="images/h1080/IMG_0220.jpg" data-size="1620x1080" data-index="4">
65 
66 </a>
67 <figcaption>
68 Tipsoo Lake
69 </figcaption>
70 </figure>
71
72 <figure class="mason-thumb">
73 <a href="images/h1080/IMG_0229.jpg" data-size="1620x1080" data-index="5">
74 
75 </a>
76 <figcaption>
77 Cloud cap
78 </figcaption>
79 </figure>
80
81 <figure class="mason-thumb">
82 <a href="images/h1080/IMG_6506.jpg" data-size="1620x1080" data-index="6">
83 
84 </a>
85 <figcaption>
86 City of Rocks
87 </figcaption>
88 </figure>
89 </article>
90 </main>

```

gallery.html

```

<!doctype html>
<html lang="en">

<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title>
John Smith - Gallery
</title>

<link href="dist/photoswipe.css" rel="stylesheet">
<link href="dist/default-skin/default-skin.css" rel="stylesheet">

<link rel="stylesheet" type="text/css" media="screen" href="style.css">
</head>

<body id="gallery">

```

# Finishing photostripe

**STEP ONE:** Return to **gallery-photoswipe.html** and copy all the code commented as  
`<!--photoswipe html interface code-->`

Basically it is all the code between the ending wrapper div and ending body tag.

This block of code includes links to 4 external js files down at the bottom. But equally important is the series of nested div tags. These tags form the interface and navigational elements of the photostripe application when it is active, meaning: you clicked on a thumbnail and it animated in the big picture. All of these div elements are hidden until then.

**STEP TWO:** Paste that big block of code into your **gallery.html** page. Put it in the exact same location, ie: right above the closing `</body>` tag.

**STEP THREE:** Make sure that you have the main jquery file referenced here. We used it earlier on the banner ad animation, but it may have a different name. You can always download the latest version of the main jquery library here: <https://jquery.com/>  
 Just make sure you know where it is, and what its name is before you link to it.

**STEP FOUR:** edit this line of code in your **gallery.html** page, as shown in bold brown.

You are adding a class of **picture** to the `<article>` element. This is what tells the browser and the scripts where to look for the images that will be animated.

**STEP FIVE:** Save your page and test it in the browser, it should work.

```
<footer class="footer-area">
  <p>Copyright 2016 Elvis, the King</p>
</footer>
</div> <!-- end wrapper div -->

<!--*****begin photoswipe html interface code-->
<div class="pswp" tabindex="-1" role="dialog" aria-hidden="true">
  <div class="pswp__bg"></div>
  <div class="pswp__scroll-wrap">

    <div class="pswp__container">
      <div class="pswp__item"></div>
      <div class="pswp__item"></div>
      <div class="pswp__item"></div>
    </div>

    <div class="pswp__ui pswp__ui--hidden">
      <div class="pswp__top-bar">
        <div class="pswp__counter"></div>
        <button class="pswp__button pswp__button--close" title="Close (Esc)"></button>
        <button class="pswp__button pswp__button--share" title="Share"></button>
        <button class="pswp__button pswp__button--fs" title="Toggle fullscreen"></button>
        <button class="pswp__button pswp__button--zoom" title="Zoom in/out"></button>
        <div class="pswp__preloader">
          <div class="pswp__preloader__icn">
            <div class="pswp__preloader__cut">
              <div class="pswp__preloader__donut"></div>
            </div>
          </div>
        </div>
      </div>
      <div class="pswp__share-modal pswp__share-modal--hidden pswp__single-tap">
        <div class="pswp__share-tooltip"></div>
      </div>
      <button class="pswp__button pswp__button--arrow--left" title="Previous (arrow left)">
      </button>
      <button class="pswp__button pswp__button--arrow--right" title="Next (arrow right)">
      </button>
      <div class="pswp__caption">
        <div class="pswp__caption__center"></div>
      </div>
    </div>
  </div>
  </div>
  <script src="js/jquery-2.1.4.min.js"></script>
  <script src="dist/photoswipe.min.js"></script>
  <script src="dist/photoswipe-ui-default.min.js"></script>
  <script src="dist/photoswipe-mwebster.js"></script>
<!--***** end photoswipe html interface code-->
</body>

</html>
```

gallery.html

```
<main class="main-area">
  <article class="picture masonry">
    <figure class="mason-thumb">
      <a href="images/h1080/IMG_0197.jpg" data-size="1620x1080" data-index="1">
        
      </a>
      <figcaption>
        Rialto Beach
      </figcaption>
    </figure>
  </article>
</main>
```

**Debugging:** Test every shortcut button in Dreamweaver to see if those imported files can really be found. Or, without Dreamweaver, view the source in Firefox and click the links to external files there. Those imported files are necessary for the animation. Make sure that all images can be found by the browser. Make sure that you programmed the correct dimensions `data-size=""` widths for each image. If it doesn't make sense, run the html through the online html validator, it needs to be clean code.

## Running photoswipe

Once you click a thumbnail, photoswipe should animate in the full size image. You can click the arrows to change pictures, or swipe the image with the mouse. Note how intuitive it is. If your mouse stops moving for a while, the script senses that you are enjoying the image, and have read the caption, so it hides the arrows and caption. Moving your mouse brings those back, and either clicking the black, or pressing escape returns the thumbnails. Rolling the mousewheel and pressing right arrow also does something.

To see the full power of photoswipe you need to view it on your smartphone. It works great on either android or ios.

**STEP ONE:** Connect to your remote server and upload all the new images, including the subfolder named **h1080**.

**STEP TWO:** Upload the **dist** folder, plus the **gallery.html** file.

**STEP THREE:** Check that you have uploaded the correct jquery file referenced in the link:

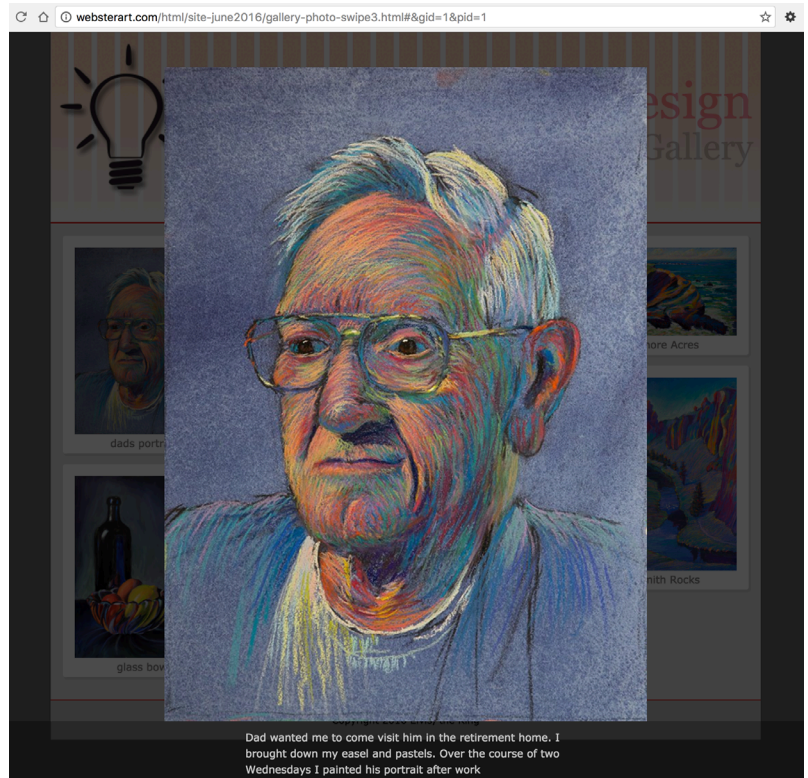
```
<script src="js/jquery-2.1.4.min.js"></script>
```

I think it is the same jquery file we used on the banner animation.

Speaking of which, upload the files that make the banner ad work. You will need to refer back to page 111 to get a list of those files.

And don't forget that you made a few tile images in photoshop for the columns, and the changes to your **style.css** file. Upload everything and take a tour on your smartphone if you own, or can borrow one. Test every page and animation.

The most common error on Photoswipe is forgetting to edit the **data-size=""** widths on each anchor tag. Each picture must have it's exact measurements in the data-size property. If you get them wrong, the picture will be distorted. Another common error is to misspell the file name on the image. It can work locally when you are testing it on your computer, but when you upload it to the server, images break if the code calls a **dog.jpg** to appear, but the file name is actually **Dog.jpg**.

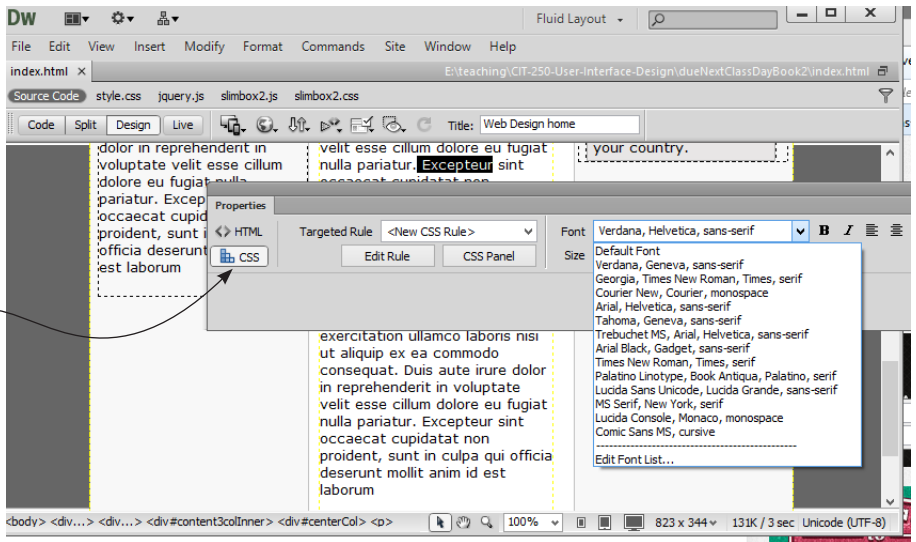


Dad wanted me to come visit him in the retirement home. I brought down my easel and pastels. Over the course of two Wednesdays I painted his portrait after work

## Web Fonts and typography

Picking fonts for your web page used to be easy. You were limited to a list of about 20 fonts that all computers were likely to have, no matter where your webpage was viewed. It can still be that easy, the list hasn't changed.

If you'd like to see it, go to the design view in Dreamweaver, highlight some words on your webpage, click the CSS button on the left side of the properties panel and drop the font drop list. There are a dozen basic choices in there, and you can be relatively certain they will work. Each choice is broken down into first, second and third choice. Note there is a menu for Edit Font List. **Don't use that.**



If you edit the font list, Dreamweaver will allow you to pick exotic fonts from your hard drive. The problem is that those fonts may be on your hard drive, but they are unlikely to be on the hard drive of your web page visitor. If the browser can't find a font you specify, it defaults to your second choice. If the browser can't find any of your choices, it will default to Times Roman, which is the built in font in most browsers.

A long discussion of the do's and don't of typography is beyond the scope of this class. If you would like to read up on it, do a google search on

**"what font should I use?"**

This website is typical of what you will find:

<http://www.smashingmagazine.com/2010/12/14/what-font-should-i-use-five-principles-for-choosing-and-using-typefaces/>

**The Non-Designer's Design Book - Robin Williams** is a good introduction to design and typography. As a general rule of thumb, you should not have more than 2 or 3 fonts on your web page. Depending on your design, it can be nice to have something exotic in the header <h1>. Don't be afraid to be bold, but remember you need to make the type easy to read. In particular, on your content body text, where you are telling stories, or describing the customers products, san-serif fonts have been proven to be most readable on computer screens: Verdana, Arial, Tahoma. Black text on a white background is ideal, though there is always some wiggle room for designers regarding the exact hues of the dark text and the light background.

**“What Font Should I Use?: Five Principles for Choosing and Using Typefaces**

By **Dan Mayer**

December 14th, 2010    Fonts, Typography    124 Comments

For many beginners, the task of picking fonts is a mystifying process. There seem to be endless choices — from normal, conventional-looking fonts to novelty candy cane fonts and bunny fonts — with no way of understanding the options, only never-ending lists of categories and recommendations. Selecting the right typeface is a mixture of firm rules and loose intuition, and takes years of experience to develop a feeling for. Here are **five guidelines for picking and using fonts** that I’ve developed in the course of using and teaching typography.

Advertisement

**MASTER THE DOMAIN**

FULL SAIL UNIVERSITY. Online Web Design & Development Degree Program

GET INFO



# Google Fonts

Because there are copyright issues with fonts, describing the exact best source for finding free web fonts is a moving target. As of this writing, google seems to be in the top two or three.

**STEP ONE:** Go to:  
[www.google.com/fonts](http://www.google.com/fonts)

**STEP TWO:** Find a nice header font. If you find fonts painful, I understand. Without some classes in the art of typography, you are running on instinct. Remember: less can be more. As you find fonts that look interesting, click the blue:

**Add to Collection** button. Note that there are a number of filters you can use to refine your search for a particular style of font.

In this screen shot I have 6 fonts in my collection.

**STEP THREE:** Narrow your font choices down to two. I know, it's painful.

**STEP FOUR:** Click the **Use** button

**STEP FIVE:** Click the **@import** button.

NOTE: this will generate code that has to be copied into your **style.css** file

Google fonts are very easy to use, but be aware they will only work when you are connected to the internet, they won't work if you are offline. The webpage doesn't have to be uploaded, but to see the fonts you have to be connected to the internet. The browser gets the font outlines directly from google.

The image is a composite of two screenshots from the Google Fonts website. The top screenshot shows the main search interface with filters for Thickness, Slant, and Width. A collection of 6 font families is shown, including Henry Penny, Rokkitt, Irish Grover, and Federant. A red arrow points from the 'Add to Collection' button in the text to the 'Add to Collection' button in the interface. The bottom screenshot shows a browser window with the Google Fonts 'Use' button clicked, displaying the '@import' code and the CSS integration instructions. A red arrow points from the 'Use' button in the text to the 'Use' button in the browser window.

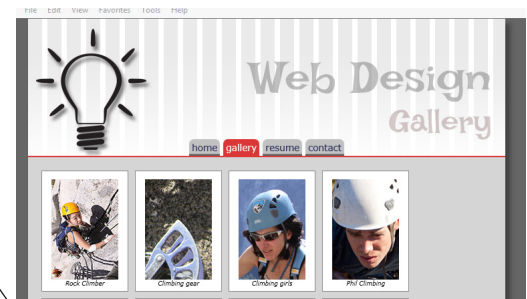


## Free web fonts in CSS 3

**STEP ONE:** Open your **style.css** file and paste the **@import** code you copied from google fonts into the very top of your style sheet.

**@import** should always come **before** any other style sheet rules.

This particular line of code brings the font outlines into the browsers cache. They are available now to be called on like any font that is in your normal font folder on your hard drive. But this font does not live in your font folder, it lives at google. If google ever goes out of business, your website will be in trouble.



style.css

```
@import url(http://fonts.googleapis.com/css?family=Averia+Sans+Libre:700|Irish+Grover);

body {
    margin: 0;
    padding: 0;
    background-color: #666;
    font: 0.85em Verdana, Helvetica, sans-serif;
}

#wrapper {
    margin-right: auto;
    margin-left: auto;
    padding: 0;
```

**STEP TWO:** Locate your **header h1** style sheet rule. Copy the font name that Google listed in its instructions and paste that into your font declaration as shown. You may need to tweak the size/leading based on the relative native size of the new font. Note the single quotes around the font name.

4. Integrate the fonts into your CSS:

The Google Fonts API will generate the necessary browser-specific CSS to use the fonts. All you need to do is add the font name to your CSS styles. For example:

```
font-family: 'Averia Sans Libre', cursive;
font-family: 'Irish Grover', cursive;
```

style.css

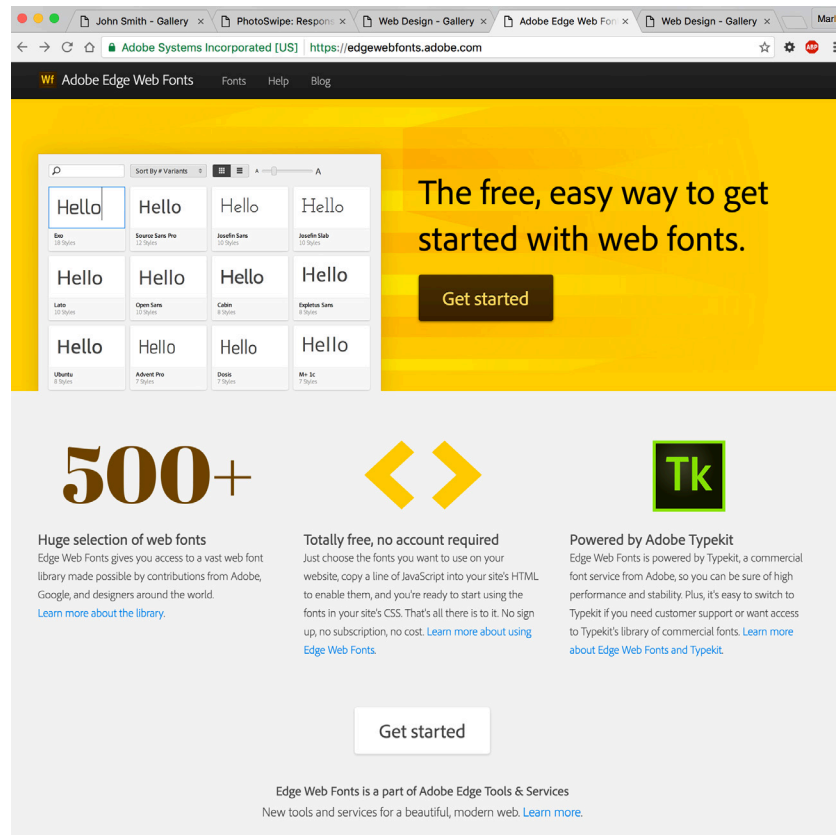
**STEP THREE:** Save the page and view it locally. If it works there, upload it and check it on all the browsers and devices you have available.

```
header h1 {
    position: relative;
    top: 60px;
    right: 10px;
    padding: 0;
    margin: 0;
    font: 5em/0.85em 'Irish Grover', cursive;
    text-align: right;
    color: #4554af;
    color: hsla(0, 0%, 38%, 0.3);
}
```

# Adobe Web Fonts

In addition to free google web fonts, Adobe has their version of free online fonts. Adobe invented theirs **after** google, so they had the benefit of observing it working at google and coming up with some improvements.

**STEP ONE:** Go to this webpage:  
<https://edgewebfonts.adobe.com/>  
 Press the **Get started** button.

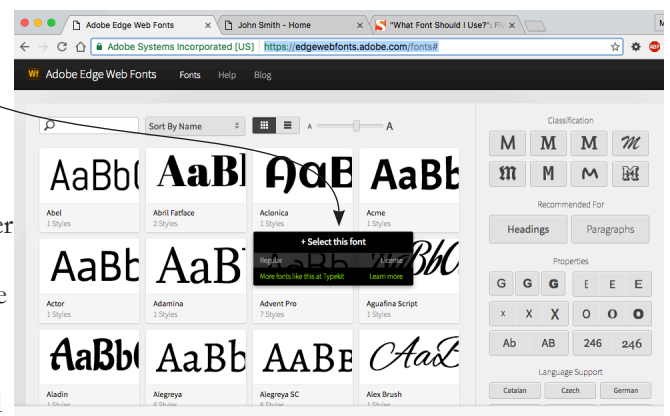


**STEP TWO:** Hover your mouse over the font you like and press the **Select this font** menu.

**STEP THREE:** Select several fonts as I show here, you get to pick the fonts.

**STEP FOUR:** Note that the script tag chains the font families together with semi-colons, but the last font-family name has a .js extension after it. So, if you decide later that you only want to use one of those font families, simply edit the script tag.

**STEP FIVE:** Note that they tell you in their websites step 2 to "Add this Javascript tag to the head". That is a **bug** on their website. It will not work. After you paste it into your `<head>`, edit it to read:  
`<script src="http://use.edgefonts.net/aclonica;aladin;aguafina-script.js"></script>`  
 That bug has been on their website from day one, perhaps it will be fixed by the time you read this.



**NOTE:** Avoid using anymore fonts than is absolutely necessary. They take time to download, and a slow loading page is not fun.



## Uploading fonts

It is possible to upload your own fonts to your webserver. The tricky part is that you must own the fonts, meaning you purchased them, and they must be legally eligible for web embedding. This website has been around for a long time and they have a page dedicated to generating webfonts:

<https://www.fontsquirrel.com/tools/webfont-generator>

Because of the free online webfont services like Adobe and Google, you may not need to use self-hosted webfonts. However you should be aware of how they work since it used to be the standard, and you will still run across them, particularly if you study web design at places like [www.lynda.com](http://www.lynda.com).

For those html tutorial websites, and for books you might buy online, such as the greensock web animation book mentioned earlier in my book, you will often download resources folders. These folders contain prebuilt webpages so you can work along with the author. She (or he) wants the webpage you are editing to match exactly with the screenshots shown in her book. Typically, because it is a professionally edited book, the author has chosen special fonts to make it look nice. And because she knows you may work on the lessons without an internet connection, you won't have access to fonts hosted at google or Adobe. So she will choose fonts for the html pages that she has purchased and for which she has generated webfonts.

Here is how the greensock (<http://www.greensock.com>) author uses his webfonts in the resources folder that I downloaded when I bought his excellent book.

Note that there is a fonts folder, with a subfolder named after a font family, and 4 font files in that folder. These are the actual font outlines, similar to what you have in your computers fonts folder, but these are designed to be uploaded and hosted on a webserver, or, in this case, downloaded and used to make a webpage look pretty while it is stored locally on your hard drive without an internet connection.

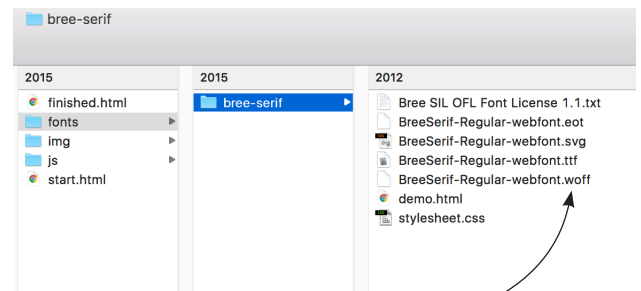
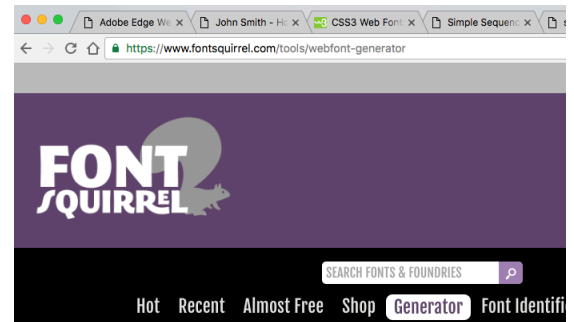
Here is a deeper explanation of these webfont file extensions:

[http://www.w3schools.com/css/css3\\_fonts.asp](http://www.w3schools.com/css/css3_fonts.asp)

Pictured is a snippet of code from the greensock files mentioned above. Note how there is an **@font-face** style sheet rule. Inside that is a **src: url('')** path that points to the font outline files that are stored down in the **fonts/bree-serif/...** folder.

After you point the browser at the source of the font outlines, you can simply call them in your style sheet rules by name. The browser figures out which one it prefers and uses that one.

```
<!DOCTYPE html>
<html>
<head>
<title>Simple Sequence</title>
<meta charset="utf-8">
<style type="text/css">
  @font-face {
    font-family: 'BreeSerifRegular';
    src: url('fonts/bree-serif/BreeSerif-Regular-webfont.eot');
    src: url('fonts/bree-serif/BreeSerif-Regular-webfont.eot?#iefix')
    format('embedded-opentype'), url('fonts/bree-serif/BreeSerif-Regular-webfont.woff')
    format('woff'), url('fonts/bree-serif/BreeSerif-Regular-webfont.ttf')
    format('truetype'), url('fonts/bree-serif/BreeSerif-Regular-webfont.svg#BreeSerifRegular')
    format('svg');
    font-weight: normal;
    font-style: normal;
  }
  body {
    background: #000 url(img/bg-body.jpg) repeat-x;
    font-family: 'BreeSerifRegular';
    color: #eee;
  }
}
```

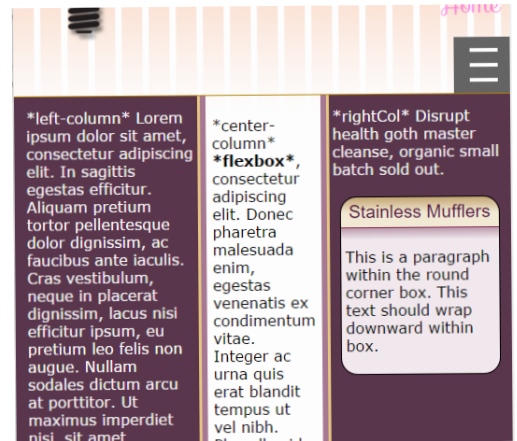


# Responsive by Design

For a while now we have had a problem with our columns. They look fine on viewports down to iPad size, but below that they fail.

**STEP ONE:** View your webpage on a smartphone, or if you don't have one, you can squish the browser window down until the center column gets excessively narrow. Here are common viewport widths:

Desktop: 1000px  
Tablet: 768px  
Mobile: 480px



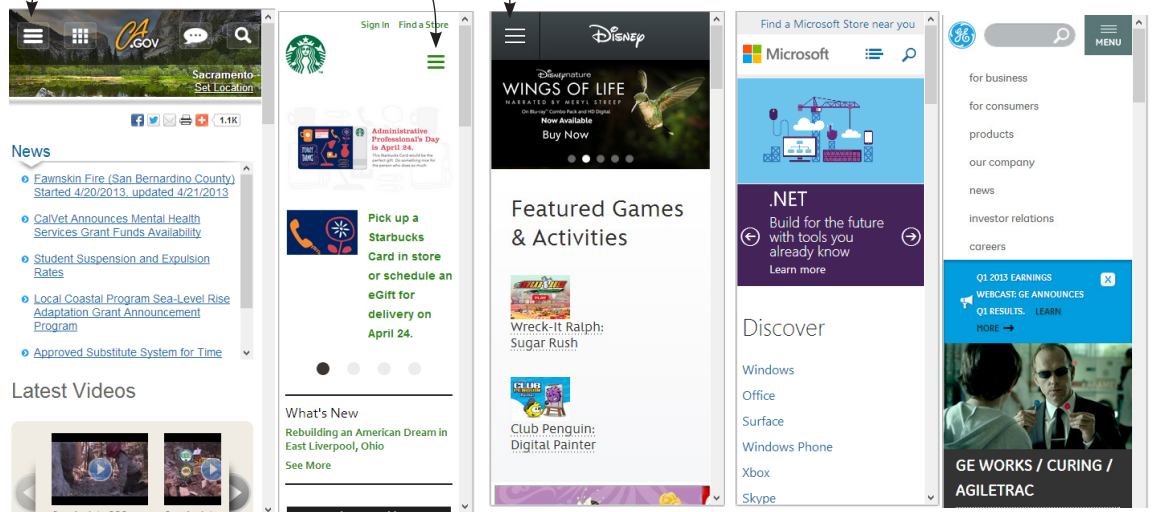
You will see our center column problem. We also need to fix our navigation menu.

**STEP TWO:** Go to this website and read the story about responsive design :  
<http://www.acquia.com/blog/5-surprisingly-great-examples-responsive-websites>

**STEP THREE:** If it is not there when you read this, go directly to these websites. View them full screen, and then resize them down to smartphone size. Watch what happens to the navigation menu.

<http://www.microsoft.com/>  
<http://www.ge.com/>  
<http://disney.com/>  
<http://www.ca.gov/>  
<http://www.starbucks.com/>

These websites are demonstrating the new responsive design theory: **Navigation should be hidden by default** on a smartphone. Content is king with limited real estate. They also convert the columns to vertically stacked block-level elements. Click the nav buttons to watch the menu magic.



## Responsive columns

When someone views a webpage on a smartphone, we need to display the most important thing first, after our branding. We will always keep our branding at the top. It's important that they can see our logo, and the name of the site `<h1>` tag.

But after the header, we need to get them to the "meat and potatoes" quickly. That is typically the content in your center column. Whether it is the thumbnails on the gallery page, or the blog posting on the index.html page, they need to see that immediately.

We know they are on a smart phone because of our media queries. In case you have forgotten, media queries allow the browser to watch the viewport width. When the browser feels that the viewport is less than the max-width property, the media query stylesheet rules go from being **hidden** to **active**. Because they are the last thing on the stylesheet, they override rules that are higher in the style sheet. In style sheets, the last thing stated wins.

**STEP ONE:** Open the **style.css** file. Scroll all the way to the bottom until you see this block of code. Change the **max-width** property to **733px**. That more accurately reflects the break point at which our media queries need to activate, based on our 3 column design.

**STEP TWO:** Edit the **header h1** as shown.

**Old:** font: 3em/0.8em Georgia, Times, serif;

**New:** font: 3em/0.8em 'Irish Grover', cursive; top: 20px;

(Or whatever google or adobe font you have chosen)

**STEP TWO:** edit **.main-content** as shown.

This changes the flex direction from the default row to column. We could have just said **display: block**; as this would have the same effect. However, it is useful to hold onto the flex properties on main-content so we can use the re-ordering capabilities built into flexbox.

style.css

```
/**begin @media queries**/

@media screen and (max-width: 733px) {
  #wrapper {
    margin: 0;
    width: 100%;
  }
  nav.main-menu ul li {
    display: block;
  }
  header.masthead h1 {
    font: 3em/0.8em 'Irish Grover', cursive;
    top: 20px;
  }
  footer.footer-area a {
    font-size: 1.5em;
  }
  nav.main-menu ul li a {
    display: block;
    text-decoration: none;
    padding: 0.2em 0.3em 0.1em 0.3em;
    margin-top: 3px;
    border-bottom: 0.2em solid #646464;
    background-color: #c2c2c2;
    font-size: 1.5em;
    color: #2c2e5c;
    /****start round corner undo code****/
    -moz-border-radius-topleft: 0;
    -webkit-border-top-left-radius: 0;
    border-top-left-radius: 0;
    -moz-border-radius-topright: 0;
    -webkit-border-top-right-radius: 0;
    border-top-right-radius: 0;
    /****stop round corner undo code****/
  }
}

.main-content {
  flex-direction: column;
}

/** end @media queries****/
```



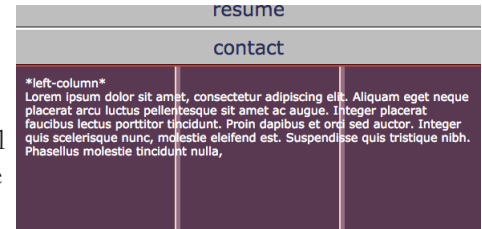
## Tweaking the media queries

**STEP ONE:** Save your change and view the page in the browser. Shrink it down to the point where the media queries activate. The default flex behaviour is **row**: 3 equal height items displaying left to right in one row. By saying **flex-direction: column**; we changed the flex items to be 3 items in one column, stacked one on top of the other in 3 rows. Some browsers get confused about the appropriate height of the rows. A fix is coming soon below.

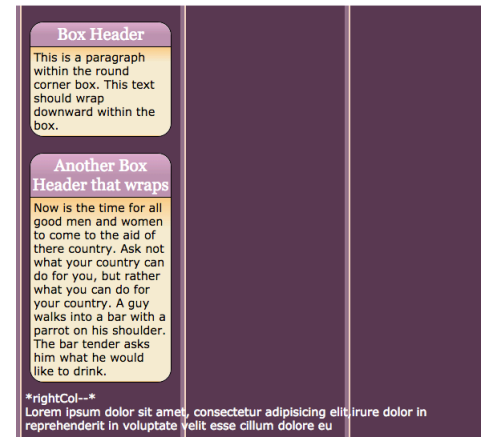
Now we need to make the center column be the first item under the navigation. We have a choice of moving the left and right column content under the more important center column content, or hiding it entirely. There are often sacrifices we have to make for the smaller screen. It really depends on your clients needs. We also need to turn off the left and right tile jpgs and center the round corner box.

**STEP TWO:** Add the code shown to the right in bold brown.

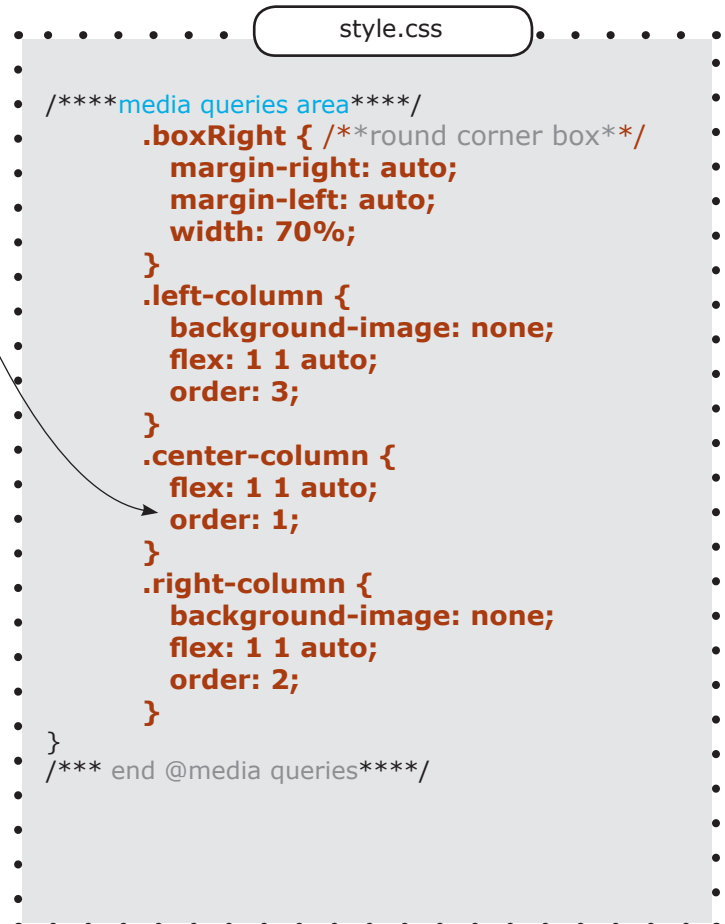
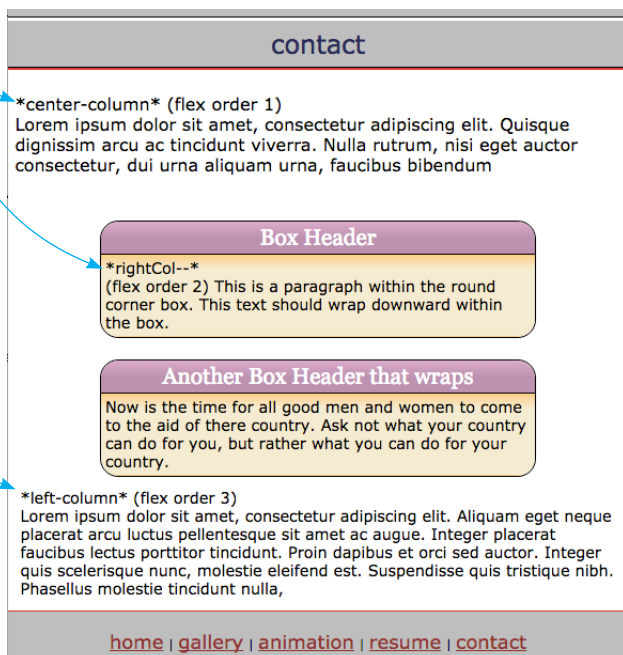
**NOTE:** Because we are in **flex-direction: column**; we can change the **order** of which flex element comes first. This new code moves the **center-column** to the top of the stack (column).



\*center-column\*  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque dignissim arcu ac tincidunt viverra. Nulla rutrum, nisi eget auctor consectetur, dui urna aliquam urna, faucibus bibendum



\*rightCol--\*  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque dignissim arcu ac tincidunt viverra. Nulla rutrum, nisi eget auctor consectetur, dui urna aliquam urna, faucibus bibendum



## Under construction

I'm sure I'll put something useful on this page, but for now, this is photo from a recent vacation I took to the high desert above Palm Springs, California.

style.css

Wake me up when it's time to go to work.



## Nav menu button

If you have a free screen ruler installed:

<http://www.arulerforwindows.com/>

<http://www.pascal.com/software/freeruler/>

You can measure the nav menu button. 60px seems to be a common size. The button has to be big enough for your finger to find it on the phone.

Ben Frain has written an excellent book called:

### Responsive Web Design with HTML5 and CSS3

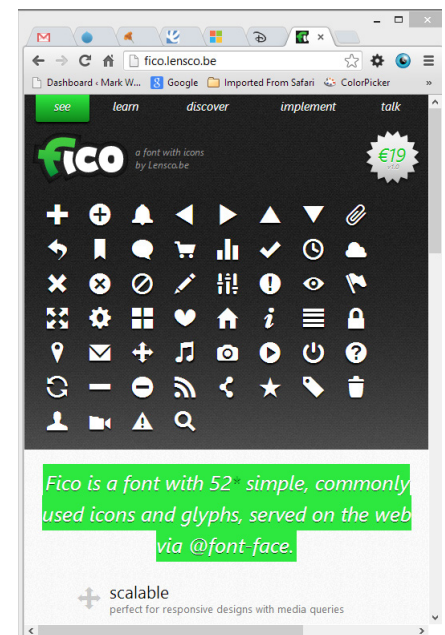
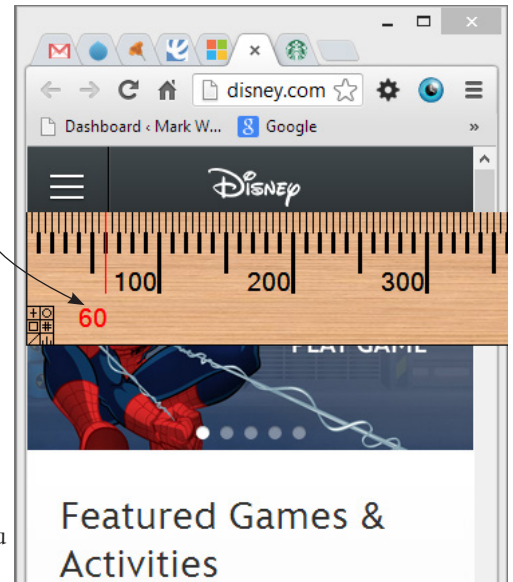
<http://www.amazon.com/Responsive-Web-Design-HTML5-CSS3/dp/1849693188/>

<http://benfrain.com/>

In his book, he recommends using web fonts for common icons like the nav menu button.

On his website he uses the fico webfont. <http://fico.lensco.be/> It will cost you \$25. If you have the money, it might be worth it. Assuming you want images for navigation, it's easier to serve up one clickable character than it is to make the graphic in Photoshop and use a custom made image for each navigation link. The font is likely to download quicker, and be much more versatile (font color, transparency) than images made in Photoshop. Do remember though that older browsers don't support webfonts. However, if this is just for smartphone navigation, you have nothing to worry about, all smartphones have new browsers :-)

To save money, we will make our nav menu button in Photoshop.



### STEP ONE: Choose file>new in Photoshop

Name: **nav-menu**

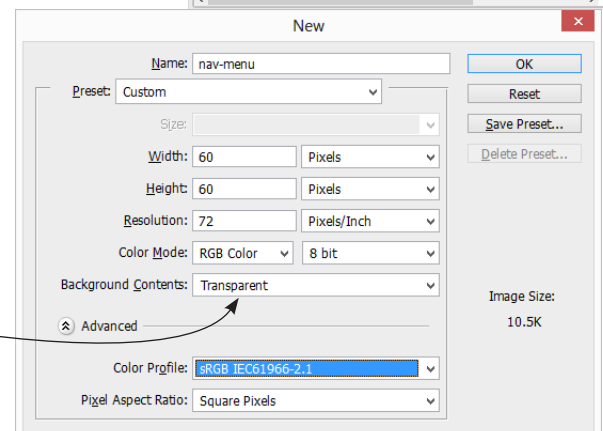
Width: **60 Pixels**

Height: **60 Pixels**

Resolution: **72**

Background Contents: **Transparent**

Click OK





## Button in Photoshop

**STEP ONE:** Make a new layer named **black fill**. Press the D for default key. This will give you a black foreground color and a white background color.

**STEP TWO:** Hold down the alt key and press the backspace key.(delete on a mac). This will pour black onto the new black fill layer.

**STEP THREE:** Make a new layer named **white dashes**

**STEP FOUR:** Zoom into 300 %

**STEP FIVE:** Turn on the info panel: window>info. Tear off (undock) your layers and info palettes so they are visible all the time.

**STEP SIX:** Choose the rectangular marquee tool.

**STEP SEVEN:** Press **d** and then **x**. This will give you white as the foreground color. (**x** swaps foreground and background)

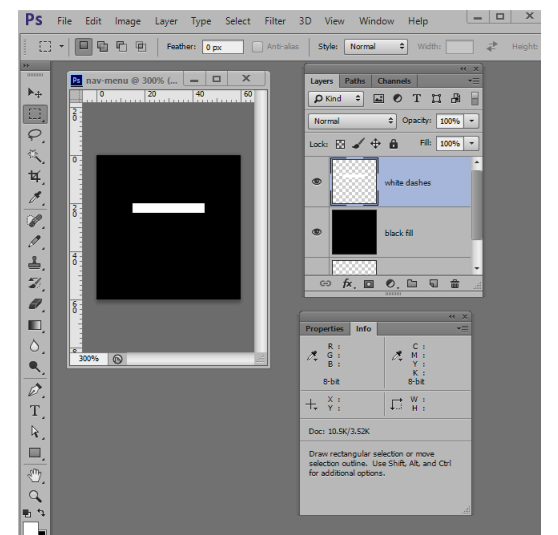
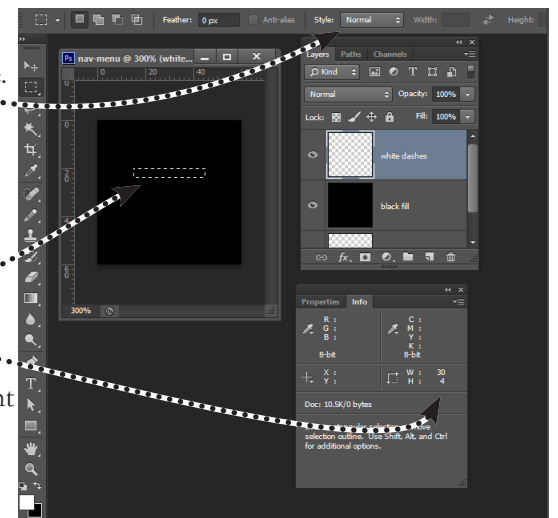
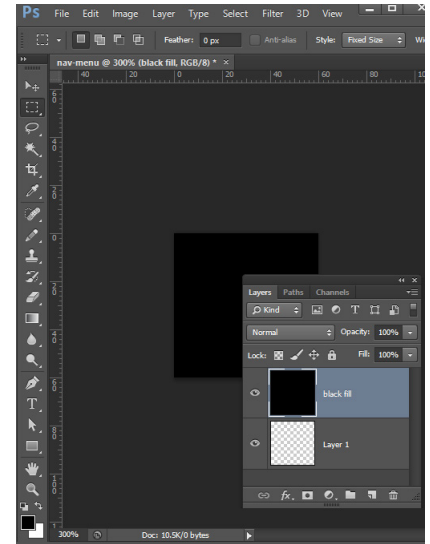
**STEP EIGHT:** Make sure the rectangular marquee tool is in style>normal mode.

**STEP NINE:** Drag out a selection that measures 30 x 4.

NOTE: if info palette is displaying inches, press control+R (view>rulers), right click on the ruler and choose pixels.

**STEP TEN:** Pour white paint into the selection: alt+backspace

**STEP ELEVEN:** Deselect (Ctrl+D)



## Button eye candy

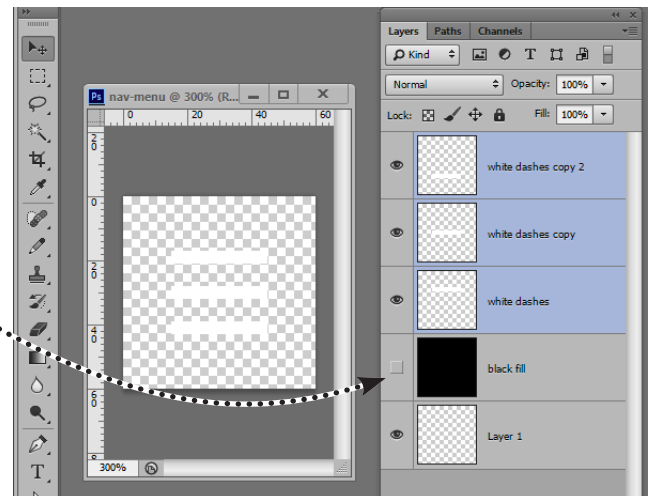
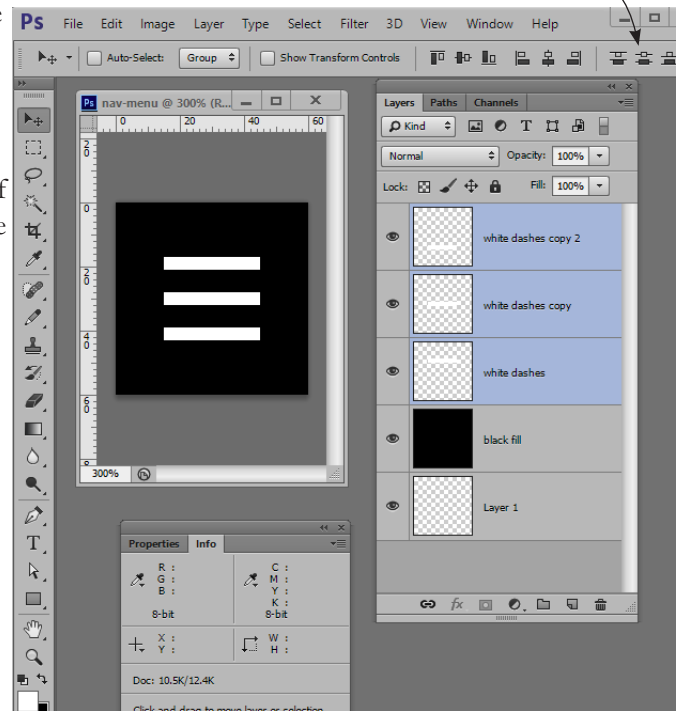
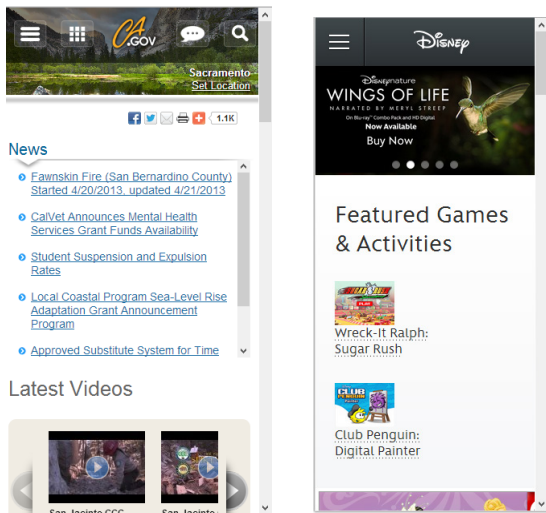
**STEP ONE:** Choose the move tool. Right click on the **white dashes** layer and choose **duplicate layer**. Click OK, the name doesn't matter.

**STEP TWO:** Make sure you are deselected and that you have the move tool. Tap the down arrow key on your keyboard. Drive the duplicate white dash down until it looks good.

**STEP THREE:** Repeat that process until you have 3 dashes as shown. You can move them as a group by shift selecting the 3 dash layers and pressing the nudge key (arrow key on keyboard)

**STEP FOUR:** When the white dash layers are shift selected together you can use the align options for the move tool to line them up, or evenly distribute their horizontal centers.

NOTE: There are many variations you can do to these dashes. If you know Photoshop, you could add a temporary white fill layer and experiment with drop shadows or outer glow layer styles. My plan is to make a transparent png image as was done on the State of California website, but feel free to be creative if you are comfortable with Photoshop.



**STEP FIVE:** Hide visibility for the black fill layer. You must see the checkerboard transparency indicators for the next step to work.



## Export out the Transparent PNG button

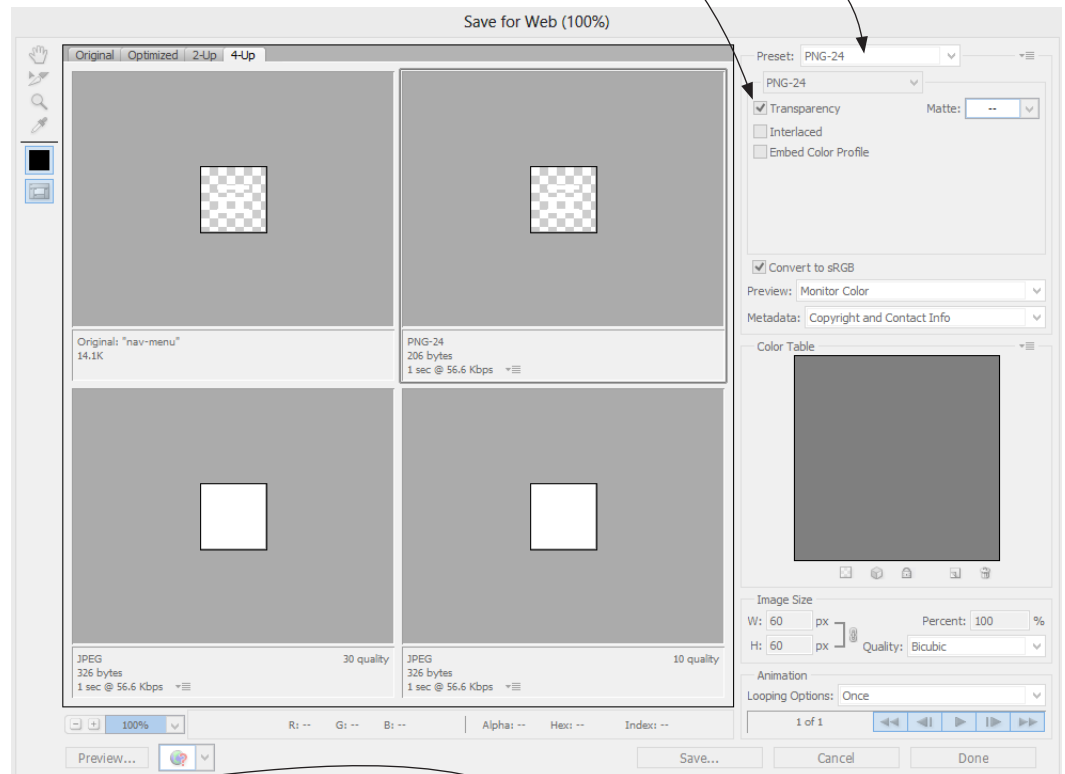
**STEP ONE:** Choose file>save for web

**STEP TWO:** Select Preset>PNG-24

**STEP THREE:** Check the box for Transparency

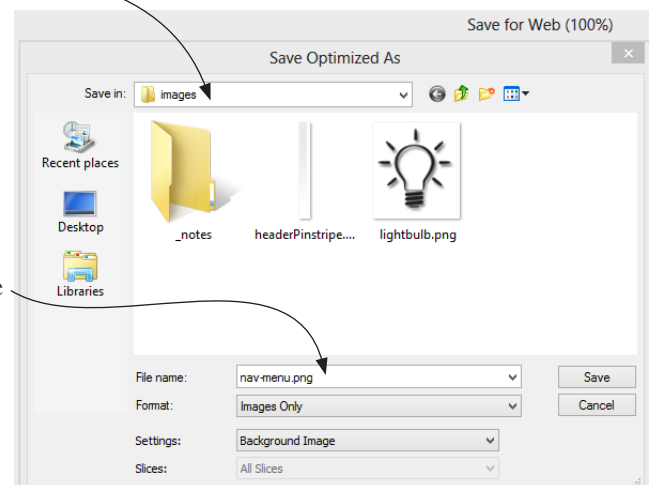
**STEP FOUR:** Click the 4-Up tab.

Note that the JPEG file format does not allow transparency, but PNG-24 does.



**STEP FIVE:** save the image down into your images folder for the currently defined site (as defined by Dreamweaver)

NOTE: Because you are saving a PNG, Photoshop only shows you the other PNG images, of which there are 2, if you are using the website we built earlier.



**STEP SIX:** Make sure the name is **nav-menu.png**. A different name is fine, but you will have to remember it for the coding.

**STEP SEVEN:** Back in Photoshop, save the Photoshop file down into your homework folder. Leave Photoshop open in case something went wrong, but minimize it.

## Menu button

Bringing the nav-menu.png button into our interface will take some clever programming. We need the button to hide when normal computers access our page. If the viewport is small, as on a mobile device, we want our menu button to appear at the same time as our navigation is hidden. If the viewport becomes normal sized again, we have to bring back our normal navigation, and hide the menu button. We will program it in stages, exploring possible options as we go.

**STEP ONE:** Open the `index.html` page and click on a new line immediately above the closing header tag:  
`</header>`

**STEP TWO:** Add the code in bold brown.

NOTE: I've put the new button inside anchor tags to maximize usability. I put it in a div tag so I could more easily speak to it with script. We will need to show it, hide it, and change it's

background color based on your color scheme. Remember, the png is white dashes on a transparent background, so you can color the div behind it.

**STEP THREE:** If you haven't done it already, put in the `<meta>` tag up at the top in the head. Thanks to former student Terren B. for the tip. That solves some problems IE 11 can experience when testing over a local network, as well as making IE9 happier

index.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="IE=edge" >

<title>
John Smith - Home
</title>
<link rel="stylesheet" type="text/css" media="screen" href="style.css">
<script src="http://use.edgefonts.net/aclonica;aladin;aguafina-script.js"></script>
</head>
<body id="home">
<div id="wrapper">
  <header class="masthead">
    <h1>Web Design<br><span>Home</span></h1>

    <div id="navPhone">
      <a href="#">
<b></b>
      </a>
    </div><!--end navPhone div-->

  </header>

  <nav class="main-menu">
```

## Positioning the menu button

**STEP ONE:** Up towards the top of the style.css file, locate the last style sheet rule for nav, there are several of them. Add the new style sheet rule shown in bold brown.

NOTE: you can make the background color anything you want to match your color scheme. The **top: 153px;** value will need to be tweaked to match your needs. Play with different numbers until it lines up with the top of your navigation links, in the narrow viewport.

```

style.css

nav.main-menu ul li a:hover {
    color: #fff;
    background-color: #db3737;
    border-bottom-color: #db3737;
}
nav.main-menu ul li a.urhere{
    color: #fff;
    background-color: #db3737;
    border-bottom-color: #db3737;
}
#navPhone {
width: 60px;
height: 60px;
background-color: #666;
position: absolute;
top: 153px;
right: 0;
}

main.main-area {
    margin: 0;
    padding: 15px;
    background: #d6d5d5;
}

```

**STEP TWO:** Up near the top in the style sheet, locate the **#wrapper** rule. Add this declaration to the existing declarations:

**position: relative;**

NOTE: The menu button was difficult to position without that last change. The button is showing up, but it needs a lot of work. If yours doesn't show, check your images folder to make sure it is there, and that you called it by the right name.

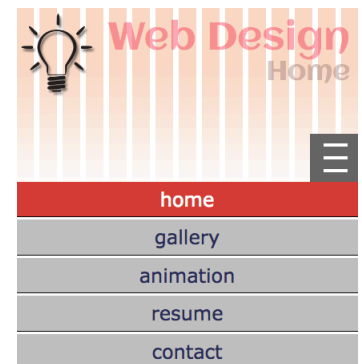
There are some great explanations out on the internet explaining the difference between position fixed, relative and absolute. They each have their uses. Google it for more details.

```

style.css

#wrapper {
    position: relative;
    margin-right: auto;
    margin-left: auto;
    padding: 0;
    width: 80%;
}

```



\*center-column\* (flex order 1)  
 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque dignissim arcu ac tincidunt viverra. Nulla rutrum, nisi eget auctor consectetur, dui urna aliquam urna, faucibus bibendum

## Show hide menu button

**STEP ONE:** Down in **media Query area**, add the code in bold brown. That should align the bottom of the menu button with the top of the home navigation button. It also tells the navPhone to show, next we will tell it to hide. (display: block means display show)

```

style.css

/****begin media query****/
@media screen and (max-width: 733px)
{
    #wrapper {
        margin: 0;
        width: 100%;
    }

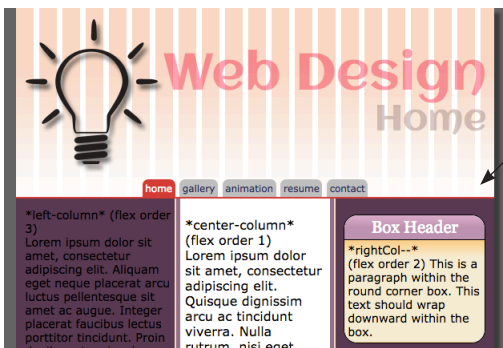
    #navPhone {
        display: block;
    }
    nav.main-menu{
        margin-top: 0;
    }
    nav.main-menu ul li {
        display: block;
    }
}

```

**STEP TWO:** Find the **#navPhone** style sheet rule up at the top of the style sheet rules, the one that is NOT in the media query area. Add a **display: none;** declaration as shown.

**STEP THREE:** Resize your window, the menu button should be showing or hiding based on viewport size.

Now you see menu button, now you don't



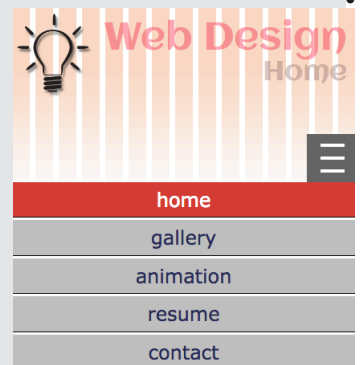
```

style.css

nav.main-menu ul li a:hover {
    color: #fff;
    background-color: #db3737;
    border-bottom-color: #db3737;
}
nav.main-menu ul li a.urhere{
    color: #fff;
    background-color: #db3737;
    border-bottom-color: #db3737;
}
#navPhone { /*this is not media queries area!*/
    display: none;
    width: 60px;
    height: 60px;
    background-color: #666;
    position: absolute;
    top: 153px;
    right: 0;
}

main.main-area {
    margin: 0;
    padding: 15px;
    background: #d6d5d5;
}

```



\*center-column\* (flex order 1)  
 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque dignissim arcu ac tincidunt viverra. Nulla rutrum, nisi eget auctor consectetur, dui urna aliquam urna, faucibus bibendum

Box Header

## Show/hide navigation links with media queries

Now that we have a menu button for controlling whether our navigation is visible or hidden, we need to give our navigation a name so we can speak to it.

**STEP ONE:** In your `index.html` page, find the `<ul>` tag that is inside `<nav class="main-menu">`

```
index.html
<nav class="main-menu">
  <ul id="navPhoneMenu">
    <li><a href="index.html" class="home">home</a></li>
    <li><a href="gallery.html" class="gallery">gallery</a></li>
    <li><a href="animation.html" class="animation">animation</a></li>
    <li><a href="resume.html" class="resume">resume</a></li>
    <li><a href="contact.html" class="contact">contact</a></li>
  </ul>
</nav>
```

**STEP TWO:** Add a new id tag to that starting `<ul>` tag as shown in bold brown.

This will allow us to speak to the unordered list with css and jquery.

**STEP THREE:** In the **media queries** section of your style.css, locate the style sheet rule for `#navPhone`. Add a new style sheet rule below that as shown in bold brown. These two rules in media queries turn the menu button on (show it) while simultaneously hiding the navigation links. Both of these things happen only when the viewport is mobile sized. (media queries are active)

```
style.css
/**begin media query***/
@media screen and (max-width: 733px)
{
  #wrapper {
    margin: 0;
    width: 100%;
  }

  #navPhone {
    display: block;
  }
  #navPhoneMenu {
    display: none;
  }
  nav.main-menu {
    margin-top: 0;
  }
}
```

**STEP FOUR:** Before you test this, scroll up to the top of your style sheet in the non-media queries area. Locate the `#navPhone` style sheet rule. Add the new rule shown in bold brown.

This rule and the one above it hide the menu button, and show the navigation links, when the viewport is at normal computer width.

Test your page, the navigation links should be visible at a wide viewport, and hidden at a narrow viewport, while the button should be showing at narrow, while hiding at wide.

```
style.css
#navPhone { /*this is not media queries area!*/
  display: none;
  width: 60px;
  height: 60px;
  background-color: #666;
  position: absolute;
  top: 153px;
  right: 0;
}
#navPhoneMenu { /*this is not media queries area!*/
  display: block;
}
main.main-area {
  margin: 0;
  padding: 15px;
  background: #d6d5d5;
}
```



## Setting up for jquery

Now our challenge is to get the navigation links to show and hide based on clicks of the menu button. There are techniques for doing it with all CSS, and it can even animate in new browsers, but for a reliable user experience, we will dip into some jquery. JQuery is a collection of javascript libraries that have been dumbed down for the common man. Supergeeks can write javascript from scratch, but many designers need something that just works, without a lot of intensive programming experience.

This is an excellent book on jquery:

**Javascript & jquery: the missing manual** - David McFarland

I learned most of what I know about jquery from his book, and from some several www.lynda.com tutorials. There are also a lot of people writing about jquery on the internet in open web forums. Ask your question to google and you will probably find the answer, though it may not be explained in a language you can understand. For true creative thinking and problem solving in jquery, you need some baseline knowledge. Both lynda.com and the book listed above are excellent resources.

I will walk you through the process of using jquery to turn our menu button into a toggle switch, similar in functionality to what is used on the disney, microsoft and starbucks websites. Before we can write jquery we have to import the master jquery library. The library has all the hard work done regarding cross browser compatibility. Fortunately we are already using jquery on our gallery photoswipe page.

**STEP ONE:** Open **gallery.html** and copy the

`<script src="js/jquery-2.1.4.min.js"></script>` tag that imports the jquery library. You will find it down towards the bottom of gallery near the closing body `</body>` tag. Paste it into `<head>` right below the link to the external style sheet. It must come before any custom jquery script we write ourselves.

**STEP TWO:** Click and add some new lines above the closing `</head>` tag.

**STEP THREE:** Start and stop a `<script></script>` tag.

**STEP FOUR:** add the `$(document).ready` code inside the script tags as shown in bold brown. This is the bare minimum required for jquery. `document.ready` tells the browser not to do anything until the entire page is **ready**, as in downloaded into the browsers cache.

```

<title>
John Smith - Home

</title>
<link rel="stylesheet" type="text/css" media="screen" href="style.css">
<script src="http://use.edgefonts.net/aclonica;aladin;aguafina-script.js"></script>
<script src="js/jquery-2.1.4.min.js"></script>
<script>
    $(document).ready(function() {
    }); //end document.ready
</script>
</head>

```

## Jquery magic

**STEP ONE:** Add the three lines of code shown in bold brown.

```

    . . . . . index.html
    . . . . .
    . . . . . <script>
    . . . . .     $(document).ready(function() {
    . . . . .
    . . . . .         $('#navPhone').click(function(){
    . . . . .             $('#navPhoneMenu').slideToggle(300);
    . . . . .         }); //end click function
    . . . . .     }); //end document.ready
    . . . . . </script>
    . . . . .
    . . . . . </head>
    . . . . .
  
```

### EXPLANATION:

the first line:

**\$('#navPhone').click(function(){**  
initializes a click event (function) on the **#navPhone** div, which is our menu button.

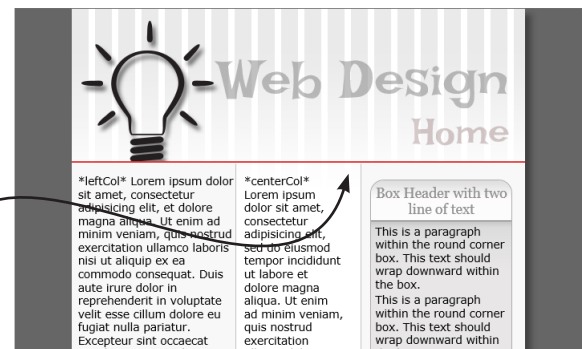
The middle line:

**\$('#navPhoneMenu').slideToggle(300);**  
is what will happen if there is a click event on the **#navPhone** menu button. Note that it specifically mentions the **#navPhoneMenu** div, which is our navigation links menu.

Following that is the **slideToggle** function. This tells the entire navigation menu to slide down over the course of 300 milliseconds, or one third of a second. **slideToggle** is a pre-defined jquery function. Think of it as the DVD drive button on your computer. Press it once and it slides into view, press it again and it hides.

**STEP TWO:** Test your webpage. Shrink your webpage down to mobile width. Click the menu button, you should see your menu slide into view. Click the menu button again and it will retract. There is much more we could do here with jquery including animating the alpha/opacity, and bounce easing, but that is a topic for another class. The information is out there on the internet if you want to pursue it.

**STEP THREE:** Fold up the menu, and then resize the window back to a normal computer sized viewport. Uh Oh. Where did the navigation menu go? Houston, we have a problem!



## Jquery resize function

Our missing menu for full size computers is related to the power of jquery. It is over-riding the **display: block;** command in our style sheet. Once you toggle those navigation links off with jquery, only jquery can bring them back.

**STEP ONE:** After the click function, add the jquery code shown in bold brown. This is basically a media query, but one written in jquery. It creates a **window resize** function that constantly watches the viewport. If you resize the browser window and it's

greater than 733px, it shows the #navPhoneMenu div, which is our navigation links. If it's smaller than 733, it hides it.

**NOTE:** Depending on the browser, CSS media queries and jquery may use different numbers due to how they measure the browser window (with or without the scroll bar). You may need to tweak the numbers to get the correct combination.

In the real world, people either come to your website with a mobile device, or they don't, so this new resize function may not be needed, other than for testing purposes. There is a chance it might be needed if you anticipate people flopping their iPad sideways, but this is something you will only know through testing.

```

index.html
<script>
    $(document).ready(function() {
        $('#navPhone').click(function(){
            $('#navPhoneMenu').slideToggle(300);
        }); //end click function

        $(window).resize(function(){
            if($(window).width() > 733){
                $('#navPhoneMenu').show();
            } else{ // window is < 733 = smartphone
                $('#navPhoneMenu').hide();
            } // end if window.width()
        }); //end window.resize function
    }); //end document.ready
</script>
</head>

```

## Corrections:

I have made some errors in logic in the structure of our interface design.

We need to go back and fix a problem with the red line that is between our navigation and the content area. We need the line there as a divider, but having the line come from the nav element is causing problems. For example, when we want the nav to hide on smartphones, the red divider line will disappear. I found, and explained a workaround by telling the unordered list to hide. That basically emptied out the nav, but left its red border. This was bad thinking on my part.

A much more logical approach is to remove the red border property from the bottom of the nav and apply it to the top of the main element. That will allow us to hide the entire nav, while keeping our red border. We can completely skip giving an id to the unordered list, which will clean up and simplify our code.

Fixing it will not be hard, but will involve some careful backtracking.



## Move red border to content div

**STEP ONE:** Find your **nav** style sheet rule and **delete** this border property out of it:

**border-bottom: 2px solid #db3737;**

Which will leave it looking like this.

NOTE: this is **not** the mediaqueries area.

style.css

```

nav.main-menu {
  margin: 0;
  margin-top: -20px;
  padding: 0;
  text-align: center;
}

/** lots of other code goes in between here, not shown**/

main.main-area {
  margin: 0;
  padding: 15px;
  background: #d6d5d5;
border-top: 2px solid #db3737;
}

main.main-content{
  display: flex;
border-top: 2px solid #db3737;
}
main.main-area.two{
  margin: 0;
  padding: 0;
  background: url(images/rightTile.jpg) top right repeat-y;
border-top: 2px solid #db3737;
}

```

**STEP TWO:** Add a **border-top** property to the following div's as shown.

- **main.main-area**
- **main.main-content**
- **main.main-area.two**

This will move the red border out of the nav, and into the content divs, which is where it should have been all along...had we known we were going to be hiding the nav.

Now that the main elements are showing our red border, we can use **nav** to show/hide the menu, instead of showing and hiding its child element: the **#navPhoneMenu** unordered list

**STEP THREE:** Down in the **media queries** area, delete or comment out the **#navPhoneMenu** rule.

**STEP FOUR:** Add a property of **display: none;** to the **nav.main-menu** style sheet rule as shown in bold brown.

style.css

```

/**begin media query**/
@media screen and (max-width: 733px)
{
  #wrapper {
    margin: 0;
    width: 100%;
  }
  #navPhone {
    display: block;
  }
  /***#navPhoneMenu {
    display: none;
  }***/
  nav.main-menu {
    margin-top: 0;
display: none;
  }
}

```

## Adjust the jquery

**STEP ONE:** Go to the head of your index.html page and edit the jquery. All three occurrences of **#navPhoneMenu** need to be replaced with **nav** as shown here in bold brown

While this may seem like a small change, it is much better logic, and will simplify our work in the future as we add dropdowns to the navigation menu.

NOTE: **e.preventDefault();** will fix a bug on smartphones where a click on the hamburger causes a 'scroll to top' event thereby hiding the menu offscreen.

**STEP TWO:** Remove the **id="navPhoneMenu"** from the **<ul>** tag. It can cause problems.

## Menu Remodel

For a while now we've had some issues with our menu, namely: it isn't very pretty. The red round corners are nice, but we can do so much better. Also, the red round corners do not lend themselves well to adding dropdowns. For example, if there is a dropdown menu, should each drop menu item also be round, or square? Our menu will be much more usable if we switch it over to the standard bar format. Onward to the Menu Remodel!

**STEP THREE:** Go to the **nav** up in the top of your style.css file. This is **\*not\*** the media queries area. Edit the **nav**, and **nav ul** style sheet rules as shown in bold brown.

```
index.html
<script>
  $(document).ready(function() {

    $('#navPhone').click(function(e){
      $('nav').slideToggle(300);
      // - fixes page jump
      e.preventDefault();
    }); //end click function

    $(window).resize(function(){
      if($(window).width() > 712){
        $('nav').show();
      } else{ // window is < 712 = smartphone
        $('nav').hide();
      } // end if window.width()
    }); //end window.resize function
  }); //end document.ready
</script>
</head>
```

```
index.html
<nav class="main-menu">
  <ul>
    <li><a href="index.html" class="home">home</a></li>
    <li><a href="gallery.html" class="gallery">gallery</a></li>
```

```
style.css
nav.main-menu {
  margin: 0;
  padding: 0;
  text-align: center;
  background: #f1eeee;
  border-top: 1px solid #aaa;
  padding-bottom: 0.01em; /*optional hack*/
}
nav.main-menu ul {
  list-style: none;
  margin: 0;
  padding: 0;
  background: #f1eeee;
  border: 1px solid #ccc;
  border-width: 0 1px; /** left and right border gets 1px **/
}
```



## Content top borders to gray

**STEP ONE:** As part of our menu remodel, I am switching our webpage over to a gray scale theme. Feel free to use any color scheme you like, but if you want to follow me and see where I go, make the changes shown to the right to the three main elements which used to have a red line at the top, now changed to gray.

That should give you this:



**STEP TWO:** edit the `nav.main-menu ul li` style sheet rule as shown in bold brown.

**STEP THREE:** edit the `ul li a` as shown, and **remove** the round corner code altogether.

**STEP FOUR:** edit the sheet rule for `ul li a:hover`

**STEP FIVE:** Edit the `urhere` rule so its colors match `a:hover`. That should give you a flat bar style navigation made from shades of gray. Explanations of the `urhere` function are back on page 64.

style.css

```
main.main-area {
    margin: 0;
    padding: 15px;
    background: #d6d5d5;
    border-top: 2px solid #aaa;
}

main.main-content{
    display: flex;
    border-top: 2px solid #aaa;
}

main.main-area.two{
    margin: 0;
    padding: 0;
    background: url(images/rightTile.jpg) top right repeat-y;
    border-top: 2px solid #aaa;
}
```



style.css

```
nav.main-menu ul li {
    display: inline-block;
    position: relative;
    margin: 0 -2px 0 -2px; /*hack to fix margins*/
    padding: 0;
    width: 8em;
    line-height: 1em;
}

nav.main-menu ul li a {
    display: block;
    text-decoration: none;
    padding: 0.4em;
    font-size: 1em;
    color: #454545;
}

nav.main-menu ul li a:hover {
    background: #454545;
    color: #fff;
}

/*urhere rule below*/
#home nav.main-menu ul li a.home,
#gallery nav.main-menu ul li a.gallery,
#animation nav.main-menu ul li a.animation,
#resume nav.main-menu ul li a.resume,
#contact nav.main-menu ul li a.contact {
    background-color: #454545;
    color: #fff;
}
```



## Drop down menu

As part of our Menu Remodel we will upgrade to a drop menu. It can be added anywhere, but I'm going to assume you need to add a drop menu to your gallery button. We will be modeling our menu on the excellent drop down menu system developed in "**More Eric Meyer on CSS**" by Eric Meyers. Eric is the master. His book is a bit old now, but still worth a look.

**STEP ONE:** In **index.html**, locate this tag, it is in the nav section

```
<li><a href="gallery.html" class="gallery">gallery</a></li>
```

**STEP TWO:** Shove the closing `</li>` tag down 4 lines, then add starting and stopping comment tags as shown.

**STEP THREE:** In the starting `<li>` tag for gallery, add a class of **dropmenu**

**STEP FOUR:** Within the two starting and stopping comment tags, write an entire starting and stopping unordered list that includes four list items:

- photography
- video
- illustrator
- inDesign

**NOTE:** we still have a link to **gallery.html**, but it has been moved down to the word **photography**.

The original gallery anchor tag now has an

`href="javascript:;"`

property. Computer users will

get a hover dropdown. But because smartphones **don't**

**have hover**, they need an

`href="javascript:;"` click element to trigger the dropdown. "javascript + colon + semicolon" is a hack to get an anchor response that triggers the menu, but goes nowhere.

**STEP FIVE:** Save and refresh. This is what you should have now. We will fix the high out of position menu shortly.

```

<nav class="main-menu">
  <ul>
    <li><a href="index.html" class="home">home</a></li>
    <li><a href="gallery.html" class="gallery">gallery</a>
      <!-- begin drop menu -->

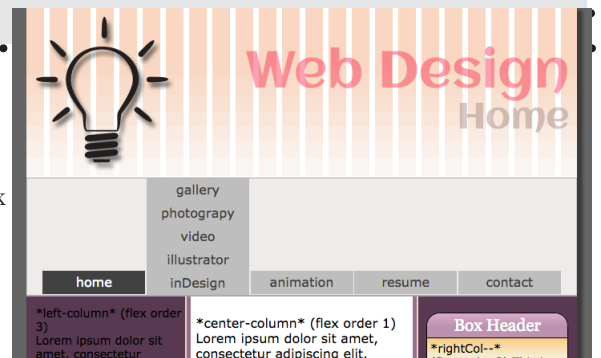
      <!-- end drop menu -->
    </li>
    <li><a href="animation.html" class="animation">animation</a></li>
    <li><a href="resume.html" class="resume">resume</a></li>
    <li><a href="contact.html" class="contact">contact</a></li>
  </ul>
</nav>

```

```

<nav class="main-menu">
  <ul>
    <li><a href="index.html" class="home">home</a></li>
    <li class="dropmenu"><a href="javascript:;"
      class="gallery">gallery</a>
      <!-- begin drop menu -->
      <ul class="leveltwo">
        <li><a href="gallery.html">photography</a></li>
        <li><a href="video.html">video</a></li>
        <li><a href="illustrator.html">illustrator</a></li>
        <li><a href="indesign.html">inDesign</a></li>
      </ul>
      <!-- end drop menu -->
    </li>
    <li><a href="animation.html" class="animation">animation</a></li>
    <li><a href="resume.html" class="resume">resume</a></li>
    <li><a href="contact.html" class="contact">contact</a></li>
  </ul>
</nav>

```



## Hang the menu and hide it

To get our drop menu `<ul>` to behave, we have to absolutely position it within its relative positioned parent `<li>gallery</li>`.

If we do that, our drop menu will look at its parent, figure out how much space it's parent needs, and hang itself immediately below it. However, because it is positioned absolutely, it is taken out of the flow of the other list items, and simply hangs there, outside of it's parent elements, exactly as is needed

**STEP ONE:** Add the three new rules to the right as shown. Save your page and preview it, it should look like this. You should have nice rollover effects.

**NOTE:** the last one is required to keep the hover color active on the gallery `<a>` while your mouse is down on the hanging menu. In essence you are getting a dual hover effect, one on `a.gallery` and one on the `<a>` tags in the menu. In case you've forgotten, this selector means: if there is an element `nav` with a class of `main-menu`, and it has a `<li>` descendent (child), with a class of `dropmenu` (`li.dropmenu`), and the dropmenu is in a `hover` state and there is a descendent `<a>` tag with a class of `gallery` use the following declarations.

```
style.css

/*urhere rule above*/
nav.main-menu ul ul {
    position: absolute;
    width: 8em;
    border: 1px solid #ccc;
}
nav.main-menu ul ul li {
    border-top: 1px solid #ccc;
}
nav.main-menu li.dropmenu:hover a.gallery {
    background: #454545;
    color: #fff;
}
```



**STEP TWO:** When you are happy with the colors of your hanging menu, tell it to hide by adding a `display: none;` property to the `nav.main-menu ul ul` rule.

Save your code and view it, the menu should be gone.

**NOTE:** we can speak to the dropmenu because it is an unordered list inside an unordered list (`ul ul`), which is inside the `nav`

```
style.css

/*urhere rule above*/
nav.main-menu ul ul {
    position: absolute;
    width: 8em;
    border: 1px solid #ccc;
    display: none;
}
nav.main-menu ul ul li {
    border-top: 1px solid #ccc;
}
```

## Add the drop arrow to gallery

We should have a downward pointing arrow by the word gallery. This would be easy enough to make in Photoshop, and position using a background url property, but to save time I will do it with a unicode font, which seems to work everywhere, even androids and iphones.

**STEP TWO:** Locate this line of code in the nav

```
<li class="dropdown"><a href="javascript:;" class="gallery">gallery</a>
```

edit it to read like this:

```
<li class="dropdown"><a href="javascript:;" class="gallery">gallery &#x25BC;</a>
```

If you would like to read more about unicode fonts, visit this webpage:

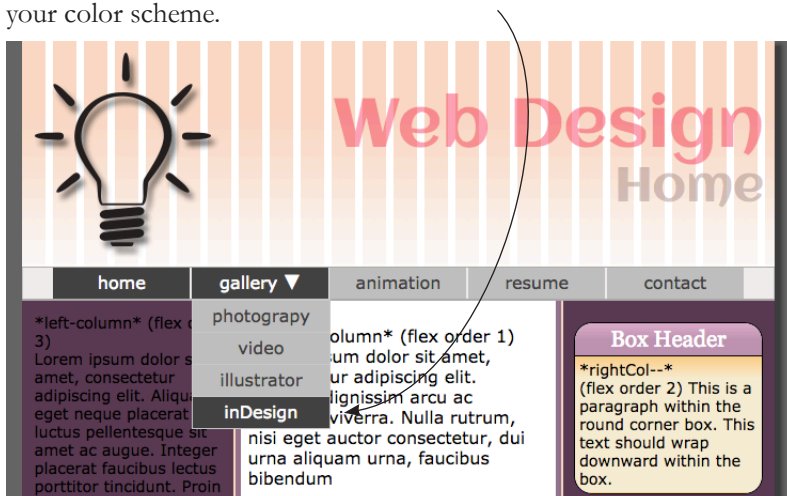
<http://www.alanwood.net/unicode/arrows.html>

**STEP THREE:** Add the long selector shown to right. Note the only thing it does is **display: block;**

**STEP FOUR:** See if you can follow the logic. As you read through the selector, when you come to spaces, like the space between **nav.main-menu** and **ul**, say the words (that has a child of). If there is a **period** between selector words, like **li.dropdown**, say the words (that has a class of). The colon means (that is in a state of).

```
style.css
nav.main-menu ul ul {
  position: absolute;
  width: 8em;
  border: 1px solid #ccc;
  display: none;
}
nav.main-menu ul ul li {
  border-top: 1px solid #ccc;
}
nav.main-menu li.dropdown:hover a.gallery {
  background: #454545;
  color: #fff;
}
nav.main-menu ul li.dropdown:hover ul.leveltwo {
  display: block;
}
```

Your menu should be working now. The colors are not perfect, see if you can fine tune them to suit your color scheme.



## Copy & paste the new nav and jquery

This menu is nice enough that we should apply it to all the pages on the website, including the new pages that don't yet exist: the ones in the drop menu.

**STEP ONE:** Copy and paste the **navPhone div** from `index.html` into all four pages on your website:

- `gallery.html`
- `resume.html`
- `animation.html`
- `contact.html`

Note that it should be placed above the closing `</header>` tag on each page.

```

index.html (finished-site-cpw-118-working-on-book) — Brackets
index.html
33 </head>
34 <body id="home">
35 <div id="wrapper">
36 <header class="masthead">
37 <h1>Web Design<br><span>Home</span></h1>
38
39 <div id="navPhone">
40 <a href="#">
41 
42 </a>
43 </div><!--end navPhone div-->
44
45 </header>
46 <nav class="main-menu">

```

**STEP TWO:** On `index.html`, highlight and copy all of the `script` tags. In this screenshot from Brackets, it's lines 12 thru 31

**STEP THREE:** Open the 4 pages that don't yet have this code and paste it into the same location, below the link to the external style sheet, which will be there already.

```

index.html (finished-site-cpw-118-working-on-book) — Brackets
index.html
6 <meta http-equiv="X-UA-Compatible" content="IE=edge">
7
8 <title>
9 John Smith - Home
10 </title>
11 <link rel="stylesheet" type="text/css" media="screen" href="style.css">
12 <script src="http://use.edgefonts.net/aclonica;aladin;aguafina-script.js"></script>
13 <script src="js/jquery-2.1.4.min.js"></script>
14 <script>
15 $(document).ready(function() {
16   $('#navPhone').click(function(e){
17     $('#nav').slideToggle(300);
18     //prevent default action - fixes page jump
19     e.preventDefault();
20   }); //end click function
21
22   $(window).resize(function(){
23     if($(window).width() > 733){
24       $('#nav').show();
25     } else { // window is < 712 = smartphone
26       $('#nav').hide();
27     } // end if window.width()
28   }); //end window.resize function
29
30 }); //end document.ready
31 </script>
32
33 </head>
34 <body id="home">

```

**STEP FOUR:** On `index.html`, copy the entire nav from the start `<nav>` tag to the stop. Paste that code over the top of the existing nav code on each page.

`<nav class="main-menu">`  
 ...  
`</nav>`

```

index.html
37 <h1>Web Design<br><span>Home</span></h1>
38
39 <div id="navPhone">
40 <a href="#">
41 
42 </a>
43 </div><!--end navPhone div-->
44
45 </header>
46 <nav class="main-menu">
47 <ul>
48 <li><a href="index.html" class="home">home</a></li>
49 <li class="dropmenu"><a href="javascript:;" class="gallery">gallery </a>
50 <!-- begin drop menu -->
51 <ul class="leveltwo">
52 <li><a href="gallery.html">photography</a></li>
53 <li><a href="video.html">video</a></li>
54 <li><a href="illustrator.html">illustrator</a></li>
55 <li><a href="indesign.html">indesign</a></li>
56 </ul>
57 <!-- end drop menu -->
58 </li>
59 <li><a href="animation.html" class="animation">animation</a></li>
60 <li><a href="resume.html" class="resume">resume</a></li>
61 <li><a href="contact.html" class="contact">contact</a></li>
62 </ul>
63 </nav>
64 <main class="main-content" role="main">
65 <aside class="left-column">

```



## Media Query Menu Makeover

You have probably noticed that our menu is not doing well on Mobile devices. In fact, it looks horrible, and there is no way to view the drop menu. Another menu makeover coming up!



**STEP ONE:** In `style.css`, scroll down to the **media queries** area. Locate the media query rule: `nav.main-menu` and make the changes shown in bold brown.

NOTE: I used `position: relative;` on `nav.main-menu` so I could absolutely position it's child element: `<ul>`

```

style.css

/****begin media query****/
@media screen and (max-width: 733px)
{
    #wrapper {
        margin: 0;
        width: 100%;
    }
    /** more rules here, but not shown**/
    nav.main-menu{
        margin: 0;
        display: none;
        position: relative;
        border: none;
    }
    nav.main-menu ul li {
        display: block;
        border-bottom: 1px solid black;
    }
}

```

## Media Query edits

**STEP ONE:** Add the `nav.main-menu ul` rule underneath the `nav.main-menu` rule you edited on the last page.

**NOTE:** This rule makes the menu's wider and moves them to the right, under the menu button.

**STEP TWO:** Modify `nav.main-menu ul li` as shown. This adds dividers between the top level list items. Divider lines aren't needed on pc's, but are needed on vertically stacked smartphone menus.

**STEP THREE:** Add the new `nav.main-menu ul ul` rule

**NOTE:** position absolute allows me to hang the dropdown menu 12em from the right side. This flies the menu out left. `top: -2px;` is a tweak.

**STEP FOUR:** Edit `nav.main-menu ul li a` as shown. I completely gutted this one. That was all garbage left over from our round corner days. I increased the font-size and padding to make fatter buttons for cell phone drivers.

Make sure you edit the existing rule, not add an extra one above the old one, it was a couple rules down in the previous version of the style sheet.

**STEP FIVE:** Add the two new rules to fine tune the menu's appearance.

These changes make the mobile width navigation look fabulous, even on iphones and androids, but the changes killed the slide down animation. Let me know if you solve that dilemma. In the meantime, go to the jquery in the top of each page and change the `slideToggle` to read `fadeToggle`.

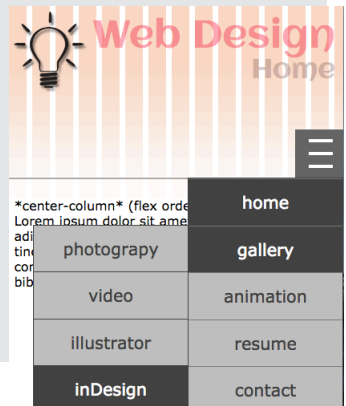
```
$('#nav').fadeToggle(300);
```

style.css

```

/**begin media query**/
@media screen and (max-width: 733px)
{
  /** more rules here, but not shown**/
  nav.main-menu {
    margin: 0;
    display: none;
    position: relative;
    border: none;
  }
  nav.main-menu ul {
    width: 12em;
    position: absolute;
    right: 0;
    top: 0;
    border: 1px solid #454545;
  }
  nav.main-menu ul li {
    display: block;
    width: 12em;
    border-top: 1px solid #7a7a7a;
    margin: 0;
  }
  nav.main-menu ul ul {
    position: absolute;
    width: 12em;
    right: 12em;
    top: -2px;
    margin-right: 0;
    border: 1px solid #454545;
  }
  nav.main-menu ul li a {
    font-size: 1.3em;
    padding: 1em;
    margin-right: 0;
  }
  nav.main-menu ul ul li {
    border-top: 1px solid #454545;
  }
  nav.main-menu ul ul li:first-child {
    border-top: none;
  }
}

```



## Flexbox for the menu

Make a backup copy of your website. The menu we just finished is quite marketable. You need to keep it pristine in case you have a need for it later. The upcoming changes will be dramatic and will destroy the menu we just finished.

### \*The last menu makeover\*

**STEP ONE:** Edit these two style sheet rule as shown in bold brown

NOTE: I have commented out everything in the `nav ul li` rule.

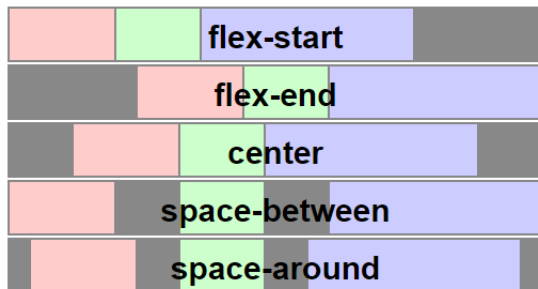
I told the parent `<ul>` tag to have a `display: flex;` property. This means that its children (the list items) are now displaying in the default **row** behavior. They are also using the default left alignment. But, because we are using flexbox, we now have a lot of tools at our disposal to make these buttons much prettier using alignment properties that have been common in word processing programs for decades.

For a full explanation of the Flexible Box Layout model, go to this address and search for **space-around**:

<http://www.w3.org/TR/css3-flexbox/#flex-property>

**NOTE:** Be sure you have removed all rules concerning:

`#navPhoneMenu`. Do a search (CTRL + F) and delete any that you find.

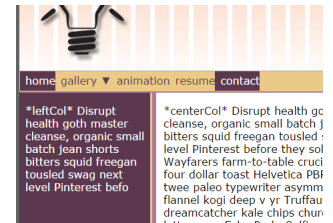


Screenshot below is from the w3.org

```
style.css

nav.main-menu ul {
  list-style: none;
  margin: 0;
  padding: 0;
  background: #ebca87;
  border: 1px solid #ccc;
  border-width: 0 1px;
  display: flex;
}

nav.main-menu ul li {
  /*display: inline-block;*/
  /*margin: 0px -2px 0px -2px;*/
  /*padding: 0;*/
  /*width: 8em;*/
  /*line-height: 1em;*/
}
```

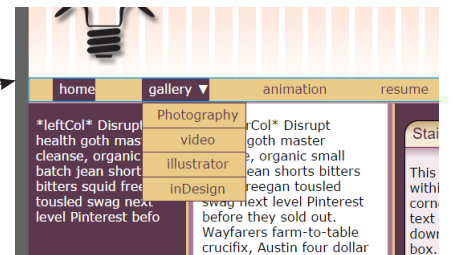


```
style.css

nav.main-menu ul {
  list-style: none;
  margin: 0;
  padding: 0;
  background: #ebca87;
  border: 1px solid #ccc;
  border-width: 0 1px;
  display: flex;
  justify-content: space-around;
}
```

**STEP TWO:** add `justify-content: space-around;` to the `#nav ul`

On our menu, `justify-content` has caused the `<li>` tags to spread out evenly across the parent `<ul>` element. The `space-around` property does some fancy math to adjust margins between list items. But there is still a problem: they get their width based on the text in the anchor tags. We can fix that by speaking directly to the child `<li>` tags.



## Flex grow, shrink, basis-width

Similar to how you can tell a `<div>` or `<td>` to have a width property in pixels, em's or percents, we can also speak to the child `<li>` flex elements.

**STEP ONE:** edit the `nav.main-menu ul li` rule as shown

This tells the `<li>` tags (children of the `<ul>` flexbox parent) to neither grow nor shrink. And to have a fixed width of 8em, which currently matches the drop menu.

But notice what happens when the viewport gets narrower than 40ems (5 `<li>` @ 8em each). The `<ul>` blows through the sides of the parent wrapper. You should play around with different combinations of numbers for grow, shrink and basis-width to see how it behaves.

**STEP TWO:** Modify `nav.main-menu ul li` as shown.

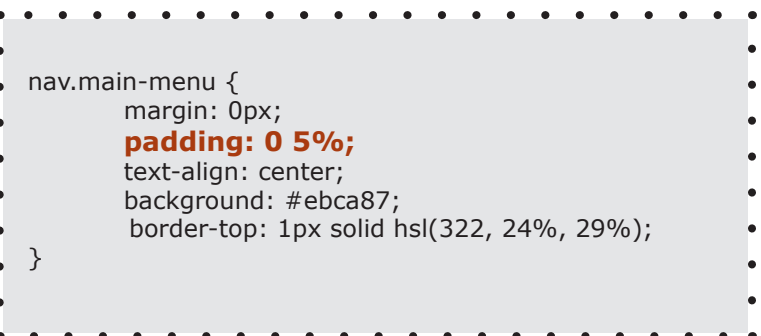
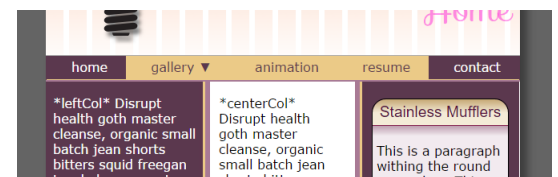
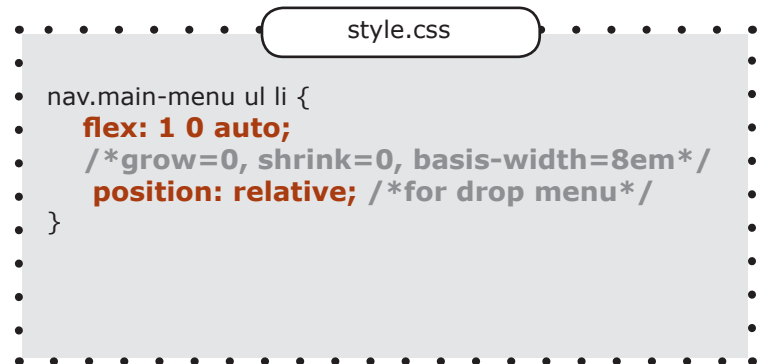
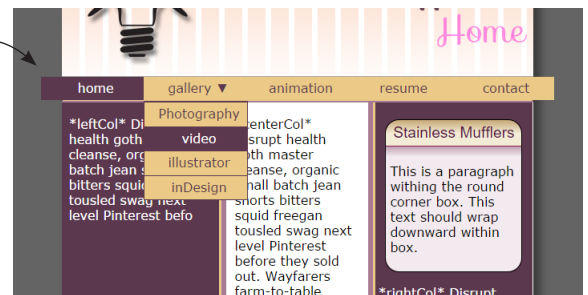
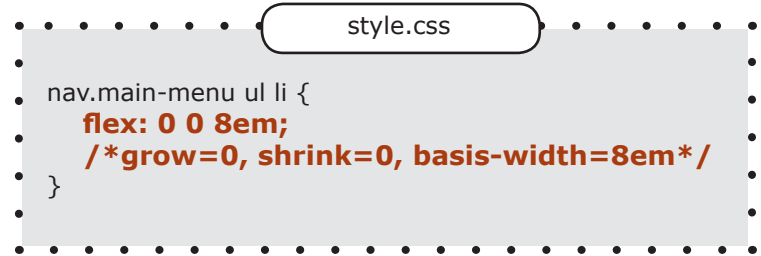
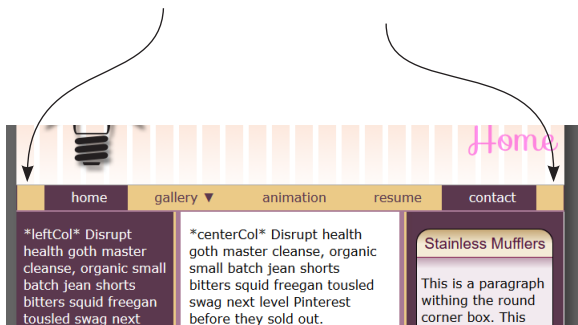
NOTE: the position relative is required for the drop menu both for proper width, and because we absolutely position the child `<ul>` dropdown.

This is perhaps our best look so far, as the `<li>` tags, are growing to fill the available space. The dropdown width doesn't match, but we can fix that later. To get that margin back on the left and right sides of the buttons, we can modify margin and padding on the parent `<ul>`

**STEP THREE:** delete the border properties on `nav.main-menu ul`

**STEP FOUR:** then modify the `nav.main-menu` property as shown to distribute 5% of padding on either side of the `<ul>` tag. If you still have that padding-bottom hack in there, remove that as well.

This gives us a reasonably pretty menu that survives larger viewport widths. (Still need to fix that dropdown width)



# Menus

To get more precise control over our rollovers we need to move the background colors and hover effects out of the `<a>` tags and onto the `<li>` tags. **NOTE:** I changed all my colors to black and white to make this easier to write up. Change the black and white back to your color scheme later.

**STEP ONE:** cut the background property out of `nav.main-menu ul li a`

**STEP TWO:** paste it into `nav.main-menu ul li`

The down side of moving our rollovers to the `<li>` elements is that a text color rule for an `<li>` tag will get overruled by a text color rule for an `<a>` tag. We will have to add a lot of specificity to get our control back.

**STEP THREE:** modify the `li.dropmenu:hover a.gallery` selector to this: `li.dropmenu:hover` and comment out the `color: white;`

**STEP THREE:** Add this new rule to fix the color of the gallery anchor text when the drop menu is hanging

**STEP FOUR:** Cut the background-color property out of `ul li a:hover` so it is just a text color change.

**STEP FIVE:** Add a new `li:hover` rule and paste in the background-color you cut from the `a:hover` rule. This should bring back your normal rollover look, but now it's happening on the `<li>` tag.

**STEP SIX:** Add two new `.leveltwo` rules to get control of the dropmenu text colors. Note, these rules might not be in this order.. If this gets confusing, check next page for all the code.

style.css

```
nav.main-menu ul li {/**
  flex: 1 0 auto;
  background-color: white;
  position: relative;*/for drop menu*/
}

nav.main-menu ul li a {
  display: block;
  text-decoration: none;
  padding: 0.5em;
  margin: 0px;
  /*background-color: white*/
  font-size: 1em;
  color: black;
}
```

style.css

```
nav.main-menu li.dropmenu:hover {
  background-color: black;
  /*color: white;*/
}

nav.main-menu li.dropmenu:hover a {
  color: white;
}
```

style.css

```
nav.main-menu ul li:hover {
  background-color: black;
}

nav.main-menu ul li a:hover {
  color: white;
}

nav.main-menu ul li.dropmenu:hover ul.leveltwo a {
  color: black;
}

nav.main-menu ul li.dropmenu:hover ul.leveltwo a:hover {
  color: white;
}
```





## nav code

style.css (partial)

```

/*****
begin nav
*****/
nav.main-menu {
    margin: 0;
    padding: 0 5%;
    text-align: center;
    background: silver;
    border-top: 1px solid gray;
    padding-bottom: 0.01em;/*optional hack*/
}
nav.main-menu ul {
    list-style: none;
    margin: 0;
    padding: 0;
    background: white;
    border: 1px solid gray;
    border-width: 0 1px; /** left and right 1px **/
    display: flex;
    justify-content: space-around;
}
nav.main-menu ul li {
    flex: 1 0 auto;
    position: relative;
    background-color: white;
}
nav.main-menu ul li: hover {
    background-color: black;
}
nav.main-menu ul li a {
    display: block;
    text-decoration: none;
    padding: 0.4em;
    font-size: 1em;
    color: black;
}
nav.main-menu ul li a: hover {
    color: white;
}
nav.main-menu ul li.dropmenu: hover ul.leveltwo a {
    color: black;
}
nav.main-menu ul li.dropmenu: hover ul.leveltwo a: hover {
    color: white;
}
nav.main-menu ul ul {
    position: absolute;
    width: 8em;
    border: 1px solid #ccc;
    display: none;
}
nav.main-menu ul ul li {
    border-top: 1px solid gray;
}

```

```

nav.main-menu li.dropmenu: hover {
    background-color: black;
}
nav.main-menu li.dropmenu: hover a {
    color: white;
}
nav.main-menu ul li.dropmenu: hover ul.leveltwo {
    display: block;
}
/*urhere rule below*/
#home nav.main-menu ul li a.home,
#gallery nav.main-menu ul li a.gallery,
#animation nav.main-menu ul li a.animation,
#resume nav.main-menu ul li a.resume,
#contact nav.main-menu ul li a.contact {
    background-color: black;
    color: #fff;
}

#navPhone {
    width: 60px;
    height: 60px;
    background-color: #666;
    position: absolute;
    top: 153px;
    right: 0;
    display: none;
}
/*****
end nav
*****/

```

## Button eyecandy

Next I would like to add pinline borders to the sides of the buttons on rollover. Because borders on rollover make the buttons (list items) wider, we need to also add the borders to the list items in their normal "off" state. We can do this by adding invisible (alpha: 0;) borders, and then simply turn them on at mouseover. They will be like Christmas lights on a tree in August...waiting for electricity.

**STEP ONE:** add invisible borders to the sides of `nav.main-menu ul li` Because I'm using black and white, I made it 2 pixels of invisible red, but with a darker color scheme such as blue and grays, 1 pixel is plenty.

<http://hslpicker.com/#f05,0>

**STEP TWO:** Modify `nav.main-menu ul li:hover` as shown. This will switch the border color from invisible to visible on rollover (hover). The lights on the tree should light up now.

NOTE: they call it Cascading Style Sheets because rules get cascaded downward. Think of a clear stream in the woods. Imagine that a pond overflowed uphill and let a bunch of muddy water flow into the clear stream. Everything downstream from where the muddy water flowed in would be colored brown. The new rules we just wrote have cascaded down into the hanging dropdown. I didn't want side borders on the hanging menu, just on the horizontal top level menu. To fix it, we need to speak directly to the `.leveltwo li` descendents. To use the stream analogy, this will pump clear water back into the stream below the muddy inflow.

**STEP THREE:** Add the `ul.leveltwo li` rule

This should give you very a very nice menu system on the large screen, we will fix the dropdown on smart phones in coming pages.

This would be a good time to pick some prettier colors than black and white.

If you are using Brackets, click in a color value, like black, and press CTL + e to access the color picker. Once you find colors you like, simply go through and paste your light and dark colors over the top of my black and whites.

```

style.css
nav.main-menu ul li {
  flex: 1 0 auto;
  background-color: white;
  position: relative;
  border: 2px solid hsla(340, 100%, 50%, 0);
  border-width: 0 2px; /*invisible red*/
}

```

```

style.css
nav.main-menu ul li:hover {
  background-color: black;
  border: 2px solid red;
  border-width: 0 2px;
}
nav.main-menu li.dropdown ul.leveltwo li {
  border-left: none;
  border-right: none;
  border-top: 1px solid gray;
}

```



## Animated button transitions

**STEP ONE:** add an animation transition to `nav.main-menu ul li`

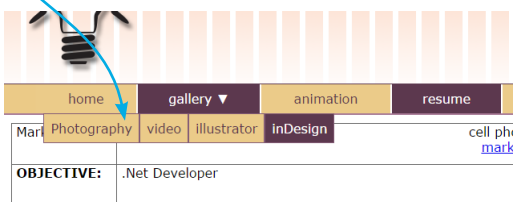
**STEP TWO:** You may have noticed a problem with your drop menu hiding under either the photoswipe thumbnails, or the Javascript banner ad. The fix is to add a z-index property to the nav list item. z-index acts like layers in Photoshop. The higher the z-index property, the closer the element is to your eyeballs, meaning it will move on top of things that are covering it up. The reason the banner ad is above the dropdown is because the banner ad is ripe with z-index, it has z-index values all the way up to 10. Hence the need to use 11 to get up above the banner ad.

**STEP THREE:** if you are using the free Brackets code editor (<http://www.brackets.io>) click in the word **ease**, and press CTRL + e. This will launch the animation editor. Yank on the bezier curve handles to modify the easing of the animation. To make it longer change the **1s** to **1.5s**. Shorter would be **0.5s**. It's measured in one thousands of a second.

It's time to fix the width on that drop menu. Because the flex property allows the parent list items to grow, the child list items need a way to grow with their parent.

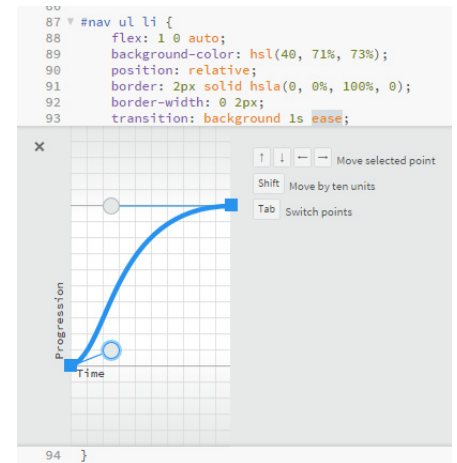
**STEP FOUR:** edit this style sheet rule by changing the **display: block;** to **display: flex;**

Not quite what we had in mind! Because they are set up to be block level list items, changing them to flex breaks things. But flex is the easiest way to get a match between the width of the triggering menu, and it's descendent dropdown .



style.css

```
nav.main-menu ul li {
  flex: 1 0 auto;
  background-color: hsl(40, 71%, 73%);
  position: relative;
  z-index: 11;
  border: 2px solid hsla(0, 0%, 100%, 0);
  border-width: 0 2px;
  transition: background 1s ease;
```



style.css

```
nav.main-menu ul li.dropdown: hover ul.leveltwo {
  display: flex;
}
```

## Drop menu width

Our problem is that the **display: flex;** property on the parent `<ul>` tag is displaying as a row. We need to enable the “wrap” property for that flex element. This allows the `<li class="dropdown">` flex item to wrap. It’s weird because it’s positioned absolutely...but in its own way it is wrapping, and wrapping the child gives us a matching width on the dropdown to the parent menu.

**STEP ONE:** Add **flex-wrap: wrap;** to the **nav.main-menu ul** style sheet rule. Note, you may or may not see this problem, not all browsers will show it.

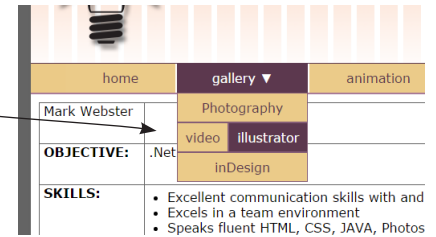
Note how the dropdown is wrapping. It matches the width of the main buttons, but the wrap property cascades down to the child list items. We can fix that by speaking to the ul ul child flex items.

**STEP TWO:** Add this style sheet rule. By setting the basis-width to 8em, we prevent it from getting small enough to wrap. It will still grow and shrink with the viewport, but 8em seems to be a happy value.

**STEP THREE:** Locate the **nav.main-menu ul ul** rule and comment out the width property. That was fine when it was a fixed width hanging **display: block;** menu. But now that it is a flexible width menu, we need to let the 8em basis width control it.

style.css

```
nav.main-menu ul {
  list-style: none;
  margin: 0;
  padding: 0;
  background: hsl(40, 71%, 73%);
  display: flex;
  flex-wrap: wrap;
  justify-content: space-around;
}
```

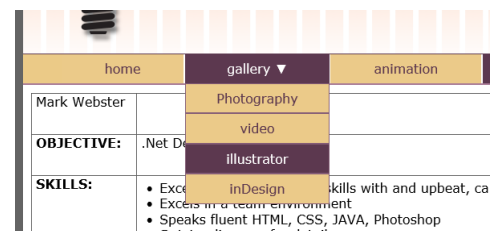
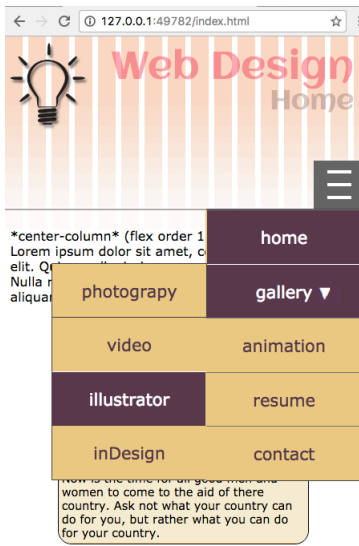


style.css

```
nav.main-menu ul li.dropdown:hover ul.leveltwo {
  display: flex;
}
nav.main-menu ul li.dropdown:hover ul.leveltwo li {
  flex: 1 1 8em;
}
```

style.css

```
nav.main-menu ul ul {
  position: absolute;
  /*width: 8em;*/
  border: 1px solid hsl(322, 24%, 29%);
  display: none;
}
```



## Find and Replace

We need to move the animation button out of the main nav and down to the drop menu under the gallery button. This will allow the navigation to fit easily on smaller viewports before media query takes over. We have to do it to all 5 pages.

**STEP ONE:** You can do it with any code editor, but Dreamweaver makes it easy. Before you do this, make sure you have correctly defined your site: Site > Manage Sites. Point it at your current working root folder. **Root folder** refers to the folder that contains (is the parent of) your **index.html** file.

**STEP TWO:** on **animation.html**, change the body id to read gallery: `<body id="gallery">` I did this so the animation page would have a lit up gallery button, since it will become part of the gallery button drop down group.

**STEP THREE:** Highlight and copy the entire nav element `<nav></nav>`.

**STEP FOUR:** Bring up the Dreamweaver Files panel, in contracted **Local Files** view, as shown. Click on the **root folder** (it's the top one) and press **Ctrl + F** (Cmd + F on a mac)

**STEP FIVE:** Change the **Find In** to read: **Entire Current Local Site**. Set **Search** to: **Source Code**

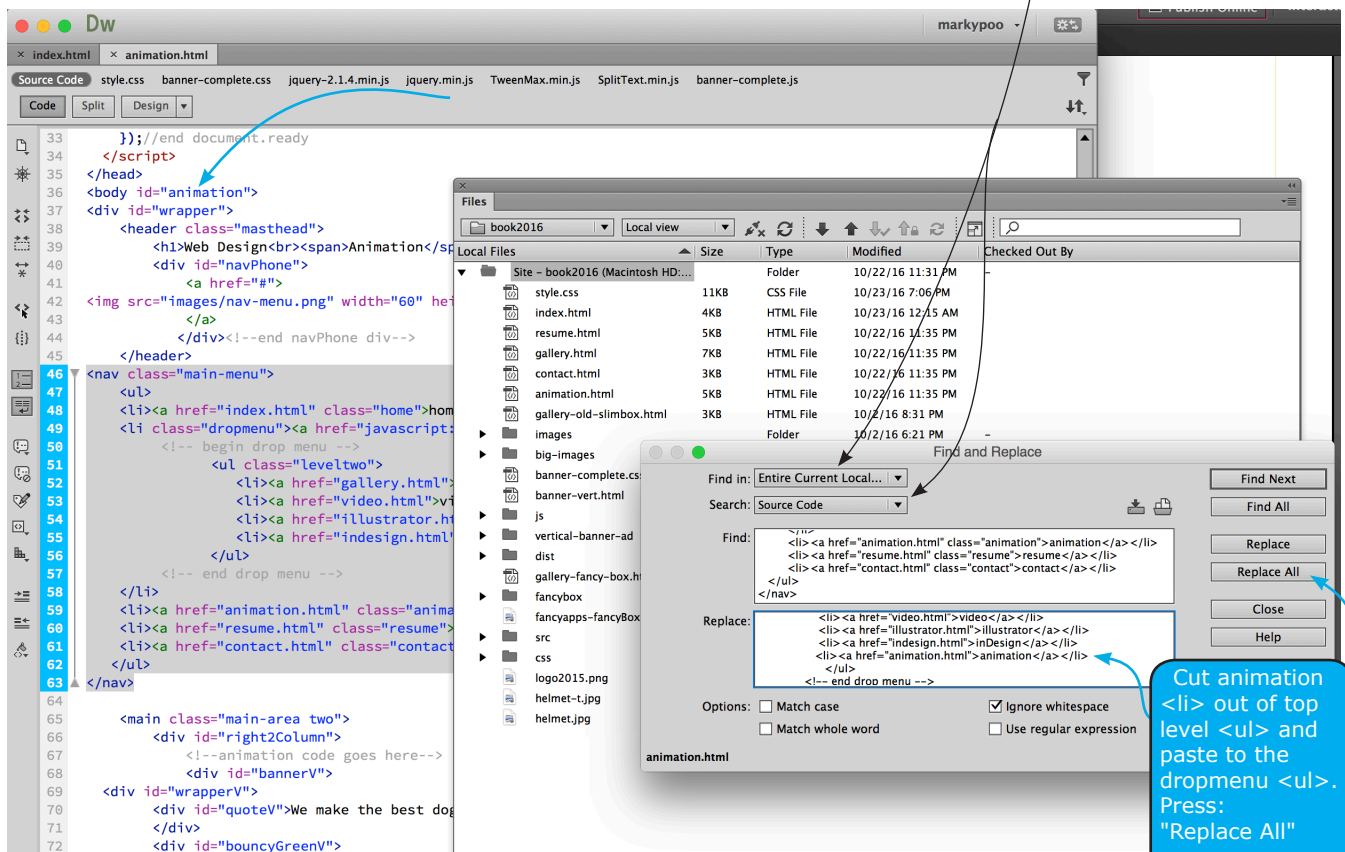
**STEP SIX:** Paste the entire **nav** into both the **Find** and **Replace** windows.

**STEP SEVEN:** In the **Replace** window, cut the animation list item out of the top level `<ul>` using Ctrl + X.

**STEP EIGHT:** Paste it back in below the inDesign list item inside the dropdown menu. Remove the **class="animation"** from the animation list item so it matches it's siblings.

**NOTE:** It can be tricky to do this inside the find and replace boxes. You can also open two side by side notepad windows. Paste the entire current nav into both notepad windows. Make the changes you want in the second notepad window, ie: moving the animation list item to the dropdown menu `<ul>`. Then manually copy the current existing nav to the Find box. Copy your edited nav to the the Replace box.

**STEP NINE:**  
Press the  
**Replace All**  
button.

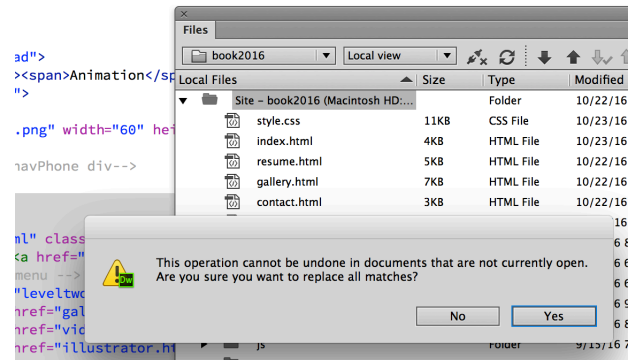




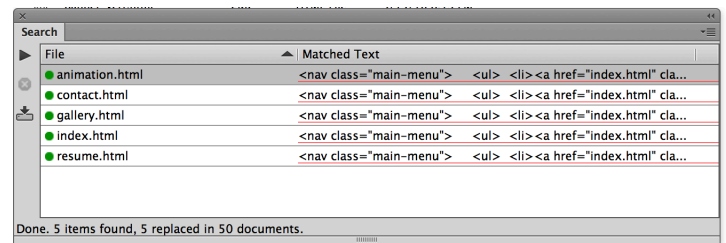
# Find and Replace

**STEP ONE:** Click **Yes** when Dreamweaver asks you if you are sure you want to replace all matches.

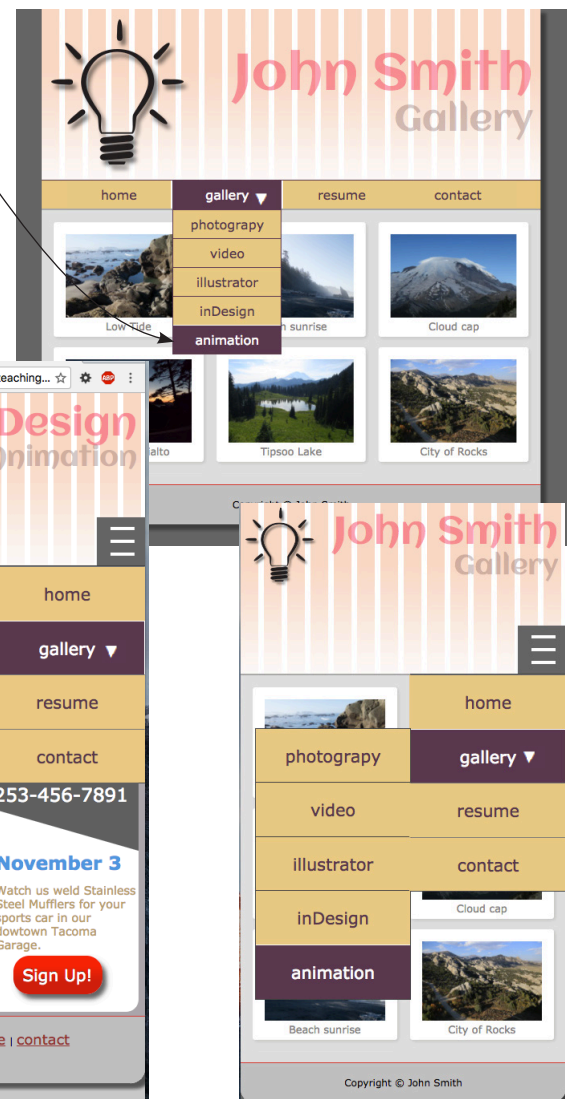
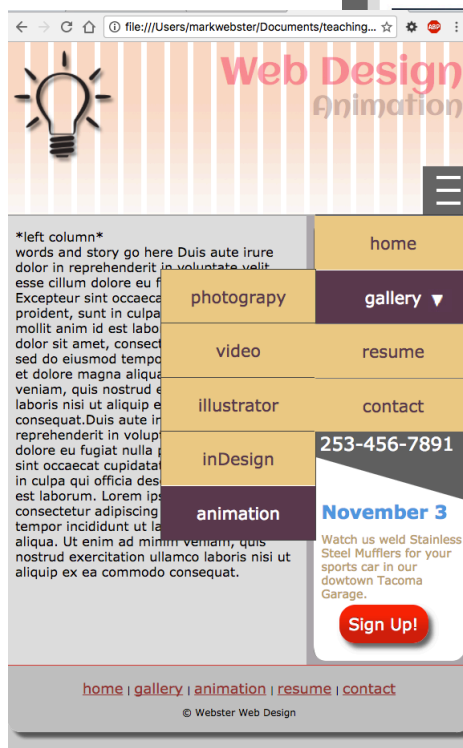
As the dialog box says, it will replace code even if the window isn't open. Before I learned about library items or php includes, I used to do Find and Replace to a 50 page website. Dreamweaver updated all the pages in about 30 seconds.



After Dreamweaver finishes it brings up a dialog box showing you the results. If you double click any of the files it will open and show you the change. Be aware that any html files that are currently open in Dreamweaver have had the changes made, but those changes were not saved. You need to click **Save All** for the changes to be permanent in open files.



These screen shots show the changed location of the animation list item, the urhere function lighting up the gallery button on the animation.html page, and the drop menu sucessfully staying above both the photoswipe thumbnails, and the Javascript banner ad, courtesy of the z-index: 11 property.



## Mobile menu

The smartphone menu needs some tweaking. It works ok for an iPad, but it's too large to fit on the screen of an old iPhone, and not aligned correctly.

**STEP ONE:** Down in media queries, edit your style sheet rules as shown. Go through them carefully one rule at a time. I've tweaked various things until it seems happy on all devices and viewports.

```

style.css

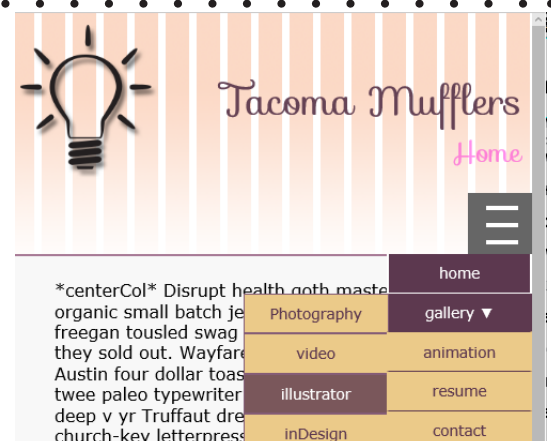
/*media queries area*/

nav.main-menu{
  margin-top: 0;
  position:relative;
  border:none;

  /*this hides nav on phone at first arrival,
  afterward jquery controls it*/
  display: none;
}
nav.main-menu ul {
  width: 8.5em;
  position:absolute;
  right:0;
  top: 0;
  border-bottom: 1px solid hsl(322, 24%, 9%);
}
nav.main-menu ul li { /*removed some declarations here*/
  border: 1px solid hsl(322, 24%, 29%);
  border-width: 1px 0 0 0;
  flex: 0 0 8.5em;
}
nav.main-menu ul li:hover{
  border: 1px solid #fff;
  border-width: 1px 0 0 0;
}
nav.main-menu ul li a {
  font-size: 0.85em;
  padding: 0.7em;
}
nav.main-menu ul ul { /*removed some declarations here*/
  position:absolute;
  right: 8.5em;
  top: 0;
  border: 1px solid hsl(322, 24%, 29%);
  border-width: 1px 0;
}
nav.main-menu ul ul li a {
  padding: 0.75em; /*alignment hack*/
}

```

Note that the flyout menu arrow is on the wrong side of the word (gallery) and it's pointing down, not left. We can fix that but it will require altering the html markup. We will have to put both arrows (down and left) in there at the same time, and tell them to show or hide depending on viewport width.



## Flyout arrows

**STEP ONE:** Locate your `<li class="dropdown">` and edit it as shown in bold brown. This will give you arrows on both sides of the word gallery.

```
index.html
<li class="dropdown">
  <a href="javascript:;" class="gallery">
    <span class="small">&#x25C0;</span>
    gallery
    <span class="big">&#x25B6;</span>
  </a>
  <!--begin drop menu-->
  <ul class="leveltwo">
    <li><a href="gallery.html">Photography</a></li>
    <li><a href="video.html">video</a></li>
    <li><a href="illustrator.html">illustrator</a></li>
    <li><a href="indesign.html">inDesign</a></li>
  </ul>
  <!--end drop menu-->
</li>
```

**STEP TWO:** copy and paste that new code into all of your html pages.

**STEP THREE:** up in the top nav area of your style sheet, add this new rule. It will hide the arrow meant for small screens.

```
style.css
.dropdown a .small {
  display: none;
}
```

**STEP FOUR:** Down in media queries, add these two new rules. They will show the left arrow on smartphones, and hide the down arrow.

**STEP FIVE:** Clean up your custom javascript. Remove any extra stuff to make it match this exactly.

```
index.html
<script>
$(document).ready(function(){
  $('#navPhone').click(function(e){
    $('#nav').fadeToggle(300);
    // cancel the default action - fixes page jump
    e.preventDefault();
  });//end navPhone.click function
$(window).resize(function(){
  if($(window).width() > 712){// must be a computer
    $('#nav').show();
  } else {// window is < 712 = smartphone
    $('#nav').hide();
  }//end if window.width
});//end window.resize
});//end document.ready
</script>
```

```
style.css
/*media queries area*/
.dropdown a .small {
  display: inline-block;
}
.dropdown a .big {
  display: none;
}
```



## Move custom jquery to an external script

**STEP ONE:** Make sure you have your website backed up before we do the next step. Get out your flash drive and copy/backup the entire website

**STEP TWO:** Examine the code at the top of your website. Note the lines of code numbered 11, 12 & 13 in the screen shot pictured here. These lines import code. Lines 12 & 13 import javascript libraries. These are also commonly known as "js" files because of the file extension \*.js, which stands for javascript.

```

1 <!doctype html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <meta http-equiv="X-UA-Compatible" content="IE=edge" >
7
8 <title>
9 John Smith - Home
10 </title>
11 <link rel="stylesheet" type="text/css" media="screen" href="style.css">
12 <script src="http://use.edgefonts.net/aclonica;aladin;aguafina-script.js"></script>
13 <script src="js/jquery-2.1.4.min.js"></script>
14 <script>
15 $(document).ready(function() {
16     $('#navPhone').click(function(e){
17         $('#nav').fadeToggle(300);
18         //prevent default action - fixes page jump
19         e.preventDefault();
20     }); //end click function
21     $(window).resize(function(){
22         if($(window).width() > 712){
23             $('#nav').show();
24         } else{ // window is < 712 = smartphone
25             $('#nav').hide();
26         } // end if window.width()
27     }); //end window.resize function
28
29 }); //end document.ready
30 </script>
31
32 </head>
33 <body id="home">
  
```

Next note the code inside the starting and stopping `<script></script>` tags. Although this is code written in the jquery language, it can be imported from an external js file. For example: rather than having 17 lines of custom jquery script, we can import it from an external js file with one line of script:

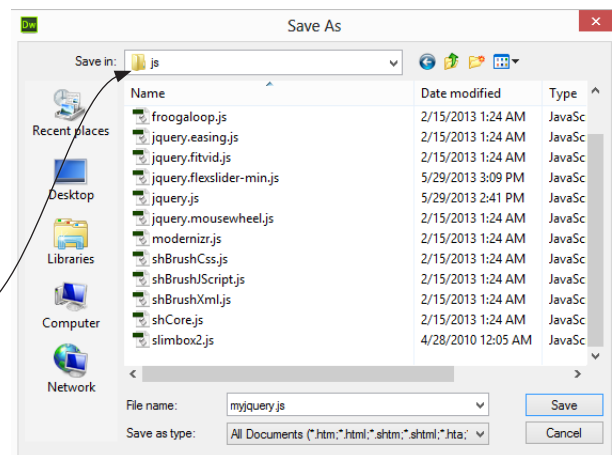
```
<script src="js/myjquery.js"></script>
```

**STEP THREE:** Highlight and **cut** all the jquery script between (but not including) the starting and stopping `<script></script>` tags. If your code looks like the screenshot above, **cut** the code from line 15 to 29. Then delete the starting and stopping `<script></script>` tags

**STEP FOUR:** In Dreamweaver (or any code editor), choose file>new document>document type>JavaScript

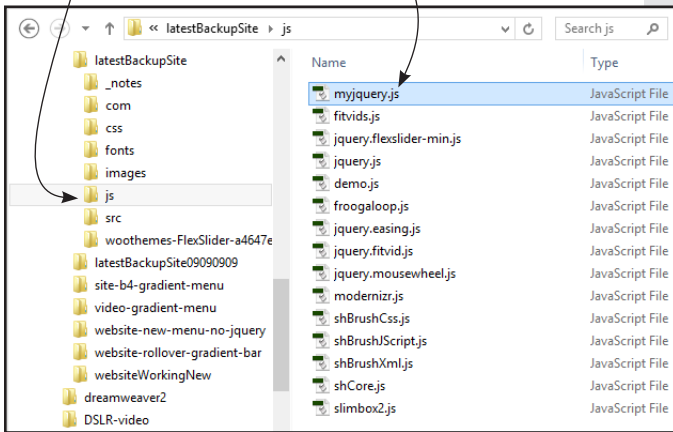
**STEP FIVE:** Choose edit>paste to bring in your cut jquery code.

**STEP SIX:** Choose file>save as. Navigate down into your "js" folder. Name the file **myjquery.js**



## Import our custom jquery

**STEP ONE:** Here is how your **myjquery.js** file should look. Make sure it is saved in the **js** folder on your website where all the other **js** files live.



```

myjquery.js
$(document).ready(function(){
  $('#navPhone').click(function(e){
    $('#nav').fadeToggle(300);
    // cancel the default action - fixes page jump
    e.preventDefault();
  }); //end navPhone.click function
$(window).resize(function(){
  if($(window).width() > 712){ // must be a computer
    $('#nav').show();
  } else { // window is < 712 = smartphone
    $('#nav').hide();
  } //end if window.width
}); //end window.resize
}); //end document.ready

```

**STEP TWO:** Add the new line of code shown below in bold brown.

NOTE: this line of code imports our custom jquery into the head of our index.html page. It's just like one of the links to the external style sheets, except this is bringing in jquery code, instead of CSS.

**STEP THREE:** Open all your webpages at once and copy/paste this line of code into the head of each page. If you have already copied the

`<script></script>` jquery code manually to those other pages, be sure to delete it before pasting in this link to the external **myjquery.js** code.

index.html

```

<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="IE=edge" >
<title>
John Smith - Home
</title>
<link rel="stylesheet" type="text/css" media="screen" href="style.css">
<script src="http://use.edgefonts.net/acronica;aladin;aguaфина-script.js"></script>
<script src="js/jquery-2.1.4.min.js"></script>
<b><script src="js/myjquery.js"></script></b>
</head>
<body id="home">
<div id="wrapper">

```



## Sliders

The absolute best sliders (support, documentation, features) will often cost you some cash.

This one is top of the line:

<http://wowslider.com/jquery-slider-bar-kenburns-demo.html>

It's free for personal use, but for commercial use it costs \$69 dollars. If you have a client who really wants a slider, and is willing to pay for it, that might be worthwhile.

There are free ones, but they are typically not as fancy:

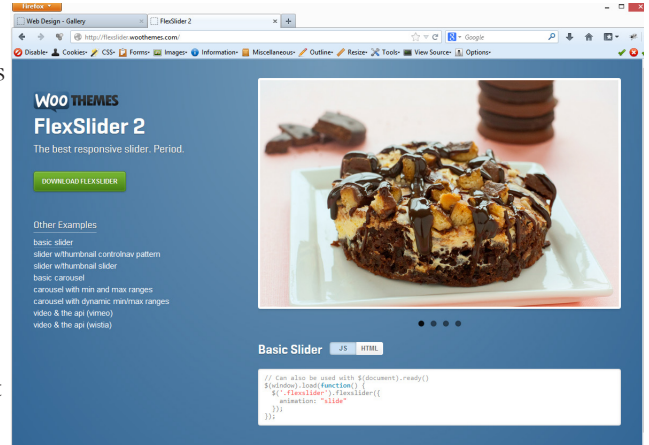
<http://www.skitter-slider.net/>



| Downloads and Licenses                                                                                                                                     |                                |                                   |                                                                |                                      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|-----------------------------------|----------------------------------------------------------------|--------------------------------------|
|                                                                                                                                                            | Non-Profit<br>Free<br>DOWNLOAD | Single Website<br>\$69<br>BUY NOW | Unlimited Websites<br>OCT. 2016 PROMO!<br>\$99 \$64<br>BUY NOW | Enterprise<br>\$299 \$149<br>BUY NOW |
| Use on commercial sites                                                                                                                                    | ✗                              | ✓                                 | ✓                                                              | ✓                                    |
| Unbranded slides<br><small>Option to remove the WOWSlider watermark and add your own logo to images</small>                                                | ✗                              | ✓                                 | ✓                                                              | ✓                                    |
| Number of websites                                                                                                                                         | Unlimited                      | 1                                 | Unlimited                                                      | Unlimited                            |
| Number of installations<br><small>Max number of WOWSlider app activation on PC or Mac computers</small>                                                    | Unlimited                      | 2                                 | 2                                                              | 10                                   |
| MegaBundle 2014<br><small>15 awesome web design apps (reg. \$1,145) with a huge discount</small>                                                           | ✗                              | ✗                                 | 92% off                                                        | 92% off                              |
| Redistribution<br><small>Enterprise License allows you to redistribute WOWSlider as a part of your larger projects, such as templates, themes, CMS</small> | ✗                              | ✗                                 | ✗                                                              | ✓                                    |
| 1 year of customer support                                                                                                                                 | ✗                              | ✓                                 | ✓                                                              | ✓                                    |
| Free updates for 1 year                                                                                                                                    | ✗                              | ✓                                 | ✓                                                              | ✓                                    |

## Responsive Slider

We've already implemented the photoswipe slider, and while it seems like the best slider to me, many people are still using the FlexSlider. It's limitations are that all the pictures have to be the exact same size. And all the big pictures load at once. There are no thumbnails. This can make the page load very slowly. On the plus side, it is fully responsive, and has those cool little buttons below it that tell you which picture is active, and how many there are in the show. Keep in mind also that using a "hero image" slider as a way to sell the objects in the photos via a click through has been proven to be very ineffective. If people click at all, they only click the first image. Nevertheless, it can still be a cool way to show off half a dozen photos. And building one is almost a "rite of passage" for new web designers.



**STEP ONE:** Do a google search for "flexslider 2". It might even be called flexslider 3 by the time you read this. Another way to find it is to search for "best jquery sliders 2016". We will use the flexslider 2 because it's free, it's relatively simple, and it's very good.

<http://flexslider.woothemes.com/>

The FlexSlider 2 jquery slider is one of the first free ones to be responsive and support smartphone swipe gestures. The documentation on version 2 is rather lean at this time, but these free sliders get better all the time, and it may be a completely different animal by the time you read these words.

All jquery sliders share some things in common. They all have some form of explanations, typically with colored code showing you exactly what to copy and paste into what part of your webpage.

They also all feature a zipped download file. When you unzip the file you will see the working web page. Typically the unzipped file will consist of some webpages that contain links to a series of folders containing images, CSS files, and JS files. The flexslider 2 comes with 5 different web pages featuring variations on the central slider. We will use the bare bones basic version of his slider. Save the more exotic sliders for later.

**STEP TWO:** Note that on this flexslider, there is a much more comprehensive set of instructions on the original flexslider (version 1), than on the new flexslider 2. Download the files for flexslider 2, but view the instructions for flexslider 1 :

<http://www.woothemes.com/flexslider/>

**STEP THREE:** Make sure your site is defined correctly, then open gallery.html. Do a save as, and change the name to illustrator.html, or whatever you already have in place in your sub navigation links.

## Modify default code

<http://www.woothemes.com/flexslider>

**STEP ONE:** Follow the instructions on the link above. Note that in the first step, they tell you get one of your scripts from <https://ajax.googleapis.com>... This is optional. If you do this, you will not be able to test your webpage unless you are online. On previous pages I showed you how to download and link to jquery libraries. Make sure that you have a link that imports a jquery library before you import the flexslider.js file.

**STEP TWO:** Note the direct link to a file called [jquery.flexslider.js](#)

This is not ideal. Look in the downloaded unzipped files for a folder called **demo**. Inside that you will see that the flexslider creator has organized everything into folders. For example, all, or most, of the javascript files (\*.js) are in a folder called **js**. We already have this folder structure in our website dating back to slimbox, photoswipe and the banner ad.

**STEP THREE:** copy his js and css files from their respective folders into our pre-existing folders. Then, in the head of **illustrator.html**, add the code show in bold brown. Note that I have edited the source paths to reflect our existing folder organization. I changed the name of **jquery.js**. Why keep the version number?

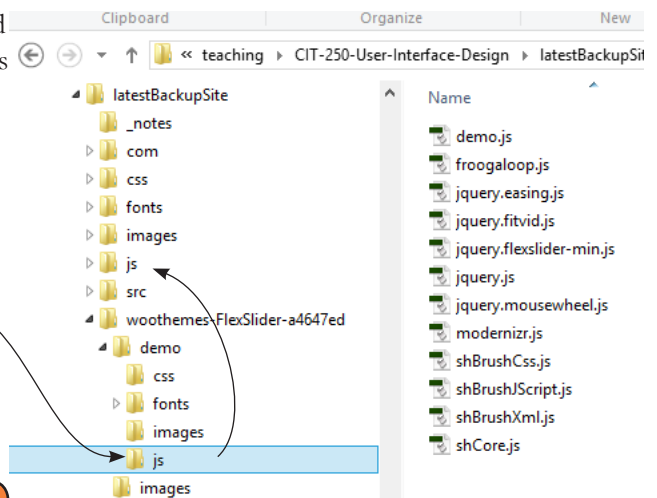
**STEP FOUR:** Azzza

## Get started with FlexSlider in 3 easy steps!

### Step 1 – Link files

Add these items to the <head> of your document. This will link jQuery and the FlexSlider core CSS/JS files into your webpage. You can also choose to host jQuery on your own server, but Google is nice enough to take care of that for us!

```
<!-- Place somewhere in the <head> of your document -->
<link rel="stylesheet" href="flexslider.css" type="text/css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery">
<script src="jquery.flexslider.js"></script>
```



illustrator.html

```
<title>Web Design - Illustrator Gallery</title>
<meta name="viewport" content="width=device-width; initial-scale=1.0; maximum-scale=1.0; user-scalable=0;">
<meta http-equiv="X-UA-Compatible" content="IE=edge" >
<script src="js/jquery.js"></script>
```

```
<script src="js/jquery.flexslider-min.js"></script>
```

```
<script type="text/javascript" src="js/myjquery.js"></script>
```

```
<link rel="stylesheet" href="css/flexslider.css" type="text/css" media="screen" >
```

```
<link rel="stylesheet" type="text/css" media="screen" href="style.css">
```

## Bring in your images

NOTE: on the previous page, my jquery libraries imports, and the meta tag are slightly different from the instructions. I am compositing his flexslider (1) instructions with the actual code he uses on his flexslider 2 demo/index.html page. I want it to be responsive, which will only happen with flexslider 2.

**STEP ONE:** Copy his code for Step 2. Note that he has simplified this code to make it easier to read.

**STEP TWO:** On your illustrator.html page, clear out the **content div** so the only thing left is the **<br>** tag. Paste in his code above the **<br>** tag.

**STEP THREE:** Modify his code as shown in bold brown.

NOTE: I have sourced the images inside our images folder. Also, I picked all **landscape images** of the **exact same size**, no portrait images. Portrait images are awkward. If you must use a portrait image, one solution is to bring it into Photoshop and add black on the sides to match the other landscape images, as shown below.

\*There are sliders that accomodate landscape/portrait images:

<http://dimsemenov.com/plugins/royal-slider/>

**STEP FOUR:** Examine (but don't copy) his code in step 3 of the website. He is assuming that you don't have any jquery on your page. We already have a bunch of custom jquery in our imported external **myjquery.js** file.

### Step 2 – Add markup

The FlexSlider markup is simple and straightforward. First, start with a **sin** element, **<div class="flexslider">** in this example. Then, create a **<ul class="slides">** important to use this class because the slider targets that class specific and anything else you desire into each **<li>** and you are ready to rock.

```
<!-- Place somewhere in the <body> of your page -->
<div class="flexslider">
  <ul class="slides">
    <li>
      
    </li>
    <li>
      
    </li>
    <li>
      
    </li>
  </ul>
</div>
```

illustrator.html

```
</nav>
<main class="main-area">

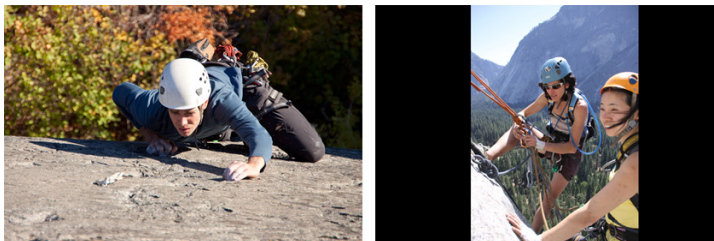
  <div class="flexslider">
    <ul class="slides">
      <li></li>
      <li></li>
      <li></li>
      <li></li>
      <li></li>
      <li></li>
      <li></li>
    </ul>
  </div><!--end flexslider div-->

  <br class="clearFloat">
</main>
```

### Step 3 – Hook up the slider

Lastly, add the following lines of Javascript into the **<head>** of your document, below the links from Step 1. The **\$(window).load()** is required to ensure the content of the page is loaded before the plugin initializes.

```
<!-- Place in the <head>, after the three links -->
<script type="text/javascript" charset="utf-8">
  $(window).load(function() {
    $('flexslider').flexslider();
  });
</script>
```



Portrait image converted to landscape

## Add the new jquery

**STEP ONE:** Open your **illustrator.html** file, add the code shown in bold brown.

NOTE: We can't add in the new **\$('.flexslider').flexslider** function to our external **myjquery.js** because it isn't needed site wide, and would cause problems on other html files. I've added it directly into the **illustrator.html** file to limit it to just this file. I've also added in two options:

**animation: "slide",**  
**pauseOnHover: true,**

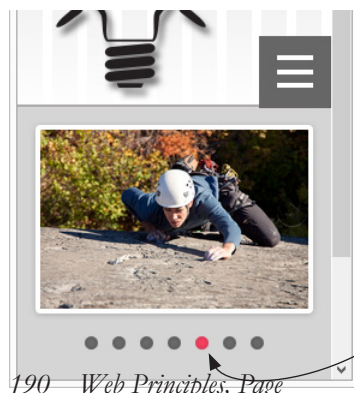
These tell the jquery to use the horizontal slide option, and to pause the slideshow when there is a mouse hover event.

**STEP TWO:** Take a look at some of the other options he has in **step 4** of the website. If you want to try them out, simply add each one below the two we have already.

Upload all the files to the live server and check it in all the browsers, including a mobile device. Test the swipe function (changing pictures by swiping on your smartphone)

You could get a lot fancier with additional options like captions, or thumbnails, but they add complexity. For many uses, this is all you need.

**STEP THREE:** If you would like to customize the colors of the little gray navigation circles under the slider, open the **flexslider.css** file. In Dreamweaver, click: **commands>apply source formatting** (to indent the css for readability). Scroll down to **line #164** and edit the style sheet rule as shown in bold brown.



NOTE: I figured out what was making the round circles by using the developer tools in Chrome. There is a cool function

```

</title>

<script type="text/javascript" src="js/jquery.js"></script>

<script src="js/jquery.flexslider-min.js"></script>
<script type="text/javascript" src="js/myjquery.js"></script>

<link rel="stylesheet" href="flexslider.css" type="text/css">

<script type="text/javascript">
  $(window).ready(function(){
    $('.flexslider').flexslider({
      animation: "slide",
      pauseOnHover: true
    });
  }); //end window.ready
</script>

<link rel="stylesheet" type="text/css" media="screen" href="style.css">
</head>
  
```

### Step 4 – Tailor to your needs

Listed below are all of the options available to customize FlexSlider to suite your needs, along with their default values. For examples on how to use these properties for advanced setups, check out the Advanced Options section.

```

namespace: "flex-",           //{NEW} String: Prefix string attached to
selector: ".slides > li",     //{NEW} Selector: Must match a simple pat
animation: "fade",            //{NEW} String: Select your animation type, "fa
easing: "swing",              //{NEW} String: Determines the easing meth
direction: "horizontal",      //{String: Select the sliding direction, "
reverse: false,                //{NEW} Boolean: Reverse the animation di
animationLoop: true,          //{Boolean: Should the animation loop? If
smoothHeight: false,          //{NEW} Boolean: Allow height of the slid
startAt: 0,                    //{Integer: The slide that the slider shou
slideshow: true,              //{Boolean: Animate slider automatically
slideshowSpeed: 7000,         //{Integer: Set the speed of the slideshow
animationSpeed: 600,          //{Integer: Set the speed of animations, i
initDelay: 0,                  //{NEW} Integer: Set an initialization de
randomize: false,             //{Boolean: Randomize slide order
  
```

### flexslider.css

```

.flex-control-paging li a.flex-active {
  background: #f00;
  background: rgba(255,28,66,0.8);
  cursor: default;
}
  
```



## Responsive Video

Videos play a big part in the modern web, and they can be made fully responsive using some tricks. You can find more on this by searching for "responsive CSS3 video"

**STEP ONE:** Go to [www.vimeo.com](http://www.vimeo.com). Click on any video, then click the "share" button. Click in the "embed" box to highlight and copy the code.

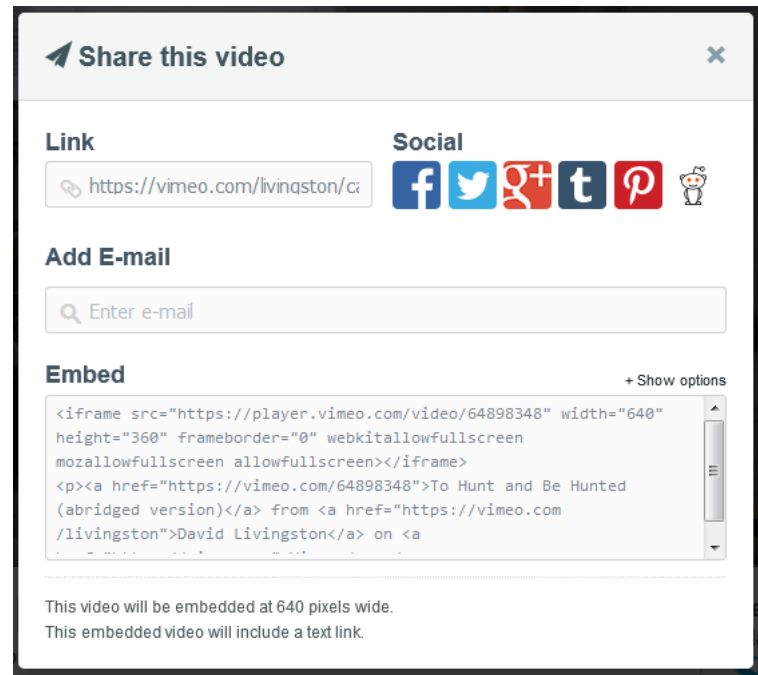
**STEP TWO:** Paste it into your content div on your video page

**STEP THREE:** Strip down the iframe code until it looks as shown below. Note that I have removed the width and height properties.

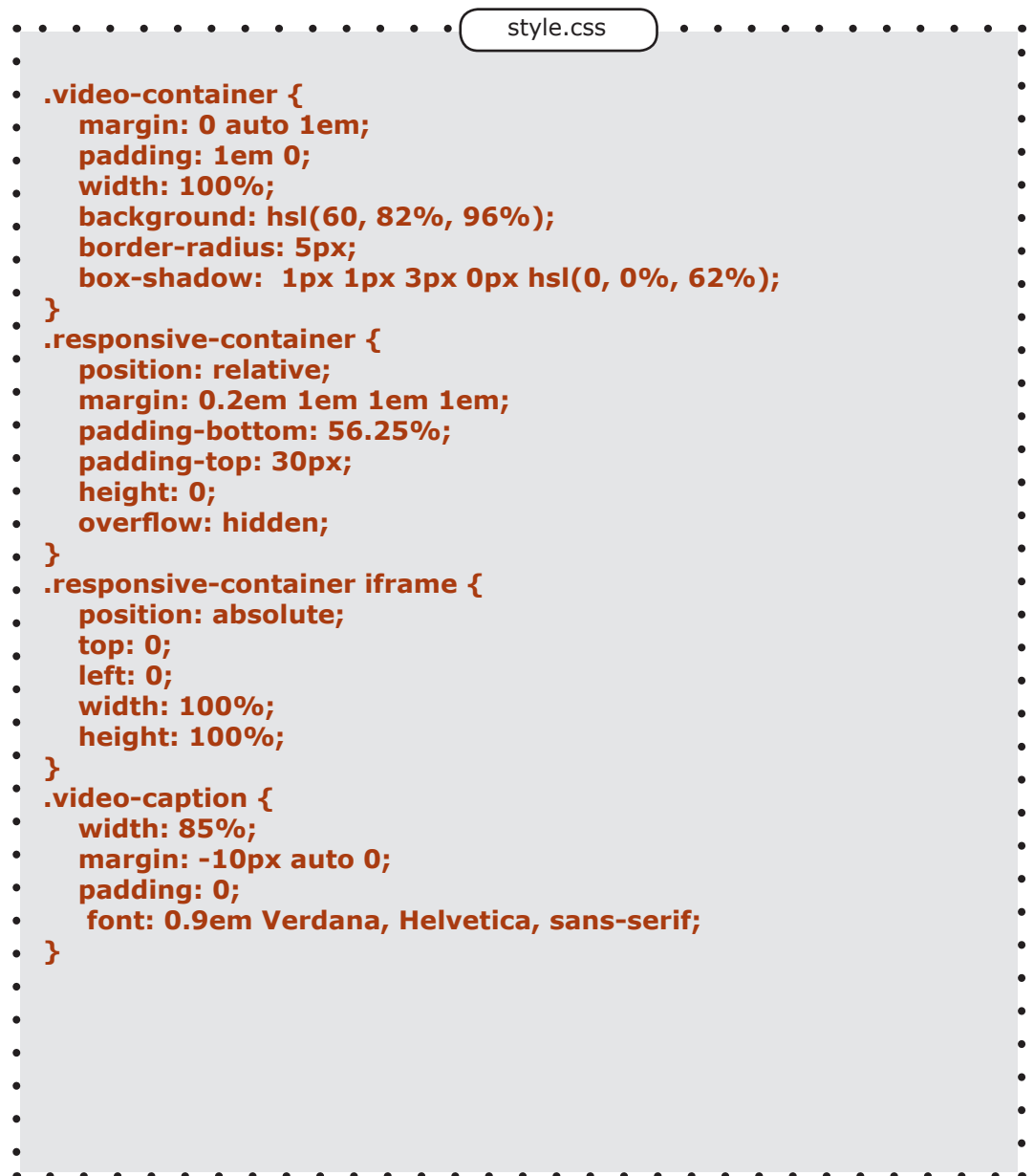
**STEP FOUR:** Add the extra `.video-container`, `figure` and `figcaption` elements above and below the iframe.

NOTE: wrapping the iframe in these tags makes the search engine happy, and allows us to force the video into being responsive.

**STEP FIVE:** zzzz



**STEP ONE:** Add the style sheet rules shown here to the bottom of your stylesheet, but above the media queries area.





## Update the footer - Library Item

The alternate navigation down in our footer is out of date. We have been adding pages to our main navigation links, but not to our footer. But before you start updating, cutting and pasting code, check out this Dreamweaver trick:

Dreamweaver was originally designed for graphic artists who didn't want to learn serious programming. Ideally the footer should be moved into a PHP include.

But if you don't know PHP, Dreamweaver has a cool function in the Assets panel called the Library.

Last century, in another life, this little trick I'm about to show you earned me a \$2.00 an hour raise. My boss didn't know about it. He was stunned when I showed him how easy this was.

**NOTE:** Backup your site before this step

**STEP ONE:** Highlight the entire footer. The code is shown in bold to right.

**STEP TWO:** Open your Dreamweaver Assets panel

**STEP THREE:** Click the bottom left Assets button, it looks like a library book.

**STEP FOUR:** With your footer code highlighted (Source Code view), press the **New Library Item** button.

**STEP FIVE:** Click Ok if it complains that it might not look the same, and click the Update button for Update links.

**STEP SIX:** In the lower window, you will see a highlighted file name. Type **altnav** and press enter.

**NOTE:** Several things just happened. Dreamweaver created a file called **altnav.lbi** and stored it in a new folder called **Library** at the root of your defined site folder. It also surrounded your footer with starting and stopping comment tags that are specifically written to trigger Dreamweaver's dynamic memory. From now on, don't ever touch this yellow highlighted code.

```

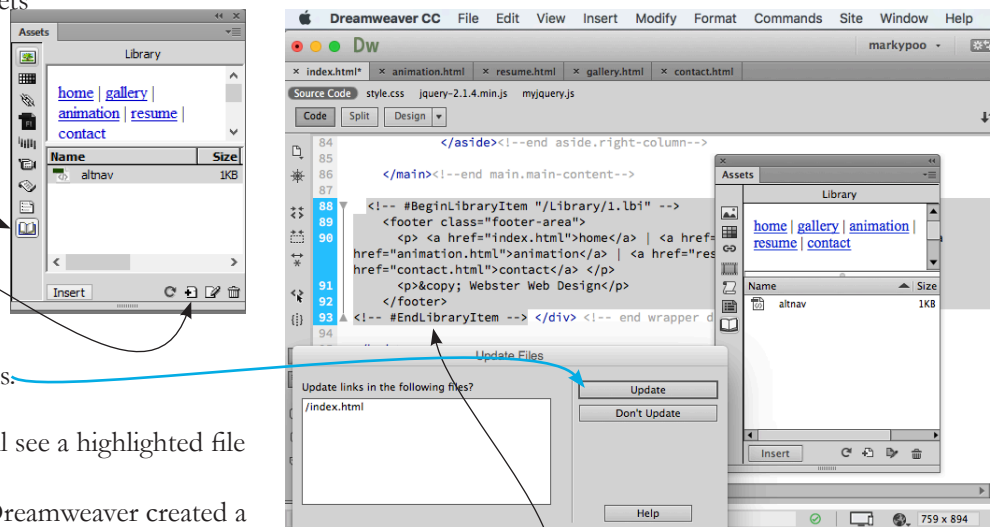
index.html

</main>

<footer class="footer-area">
  <p>
    <a href="index.html">home</a> |
    <a href="gallery.html">gallery</a> |
    <a href="animation.html">animation</a> |
    <a href="resume.html">resume</a> |
    <a href="contact.html">contact</a>
  </p>
  <p>&copy; 2013 Webster Web Design</p>
</footer>

</div><!--end wrapper div -->
</body>
</html>

```



```

</main><!--end main.main-content-->

<!-- #BeginLibraryItem "/Library/altnav.lbi" -->
<footer class="footer-area">
  <p> <a href="index.html">home</a> | <a href="
  href="animation.html">animation</a> | <a href="
  href="contact.html">contact</a> </p>
  <p>&copy; Webster Web Design</p>
</footer>
<!-- #EndLibraryItem --> </div> <!-- end wrapper
</body>

```

## Edit a Library Item

**STEP ONE:** In the Library panel, double click the **altnav.lbi** file. You can also open it from the files panel. It lives in the Library folder.

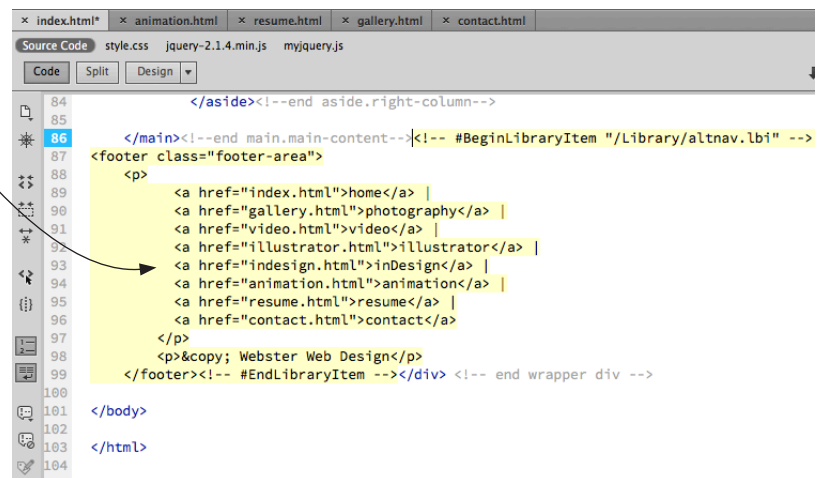
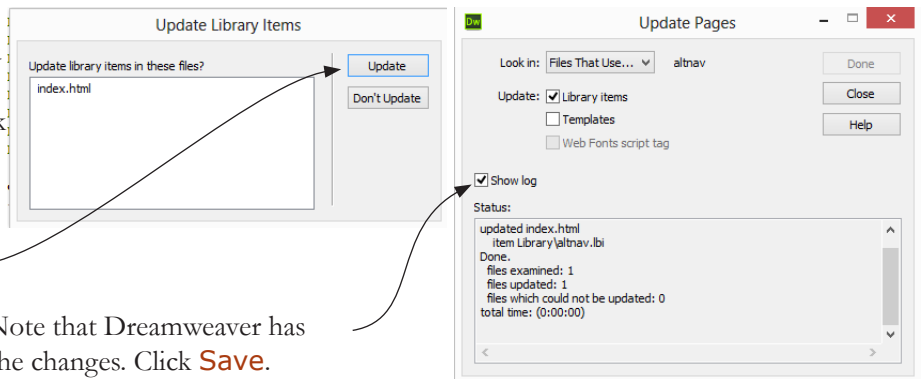
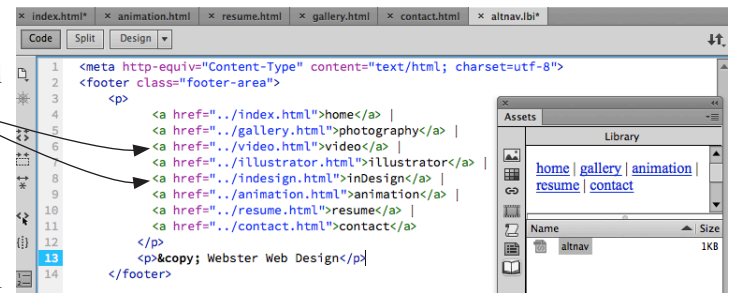
**STEP TWO:** Add in the missing links for pages that were linked in the nav.main-area but not in the footer.

Note the dot dot slash (../) in front of each file name. Dreamweaver writes it this way because it keeps track of this snippet of code (**altnav.lbi**) and stores it in the Library folder. From where this snippet of code lives, it is indeed up one level (../) in the site folder structure. Dreamweaver won't actually use this (../) syntax on your webpages. It simply stores the information this way. It will write your code correctly.

**STEP THREE:** Save your changes to the **altnav.lbi** file. Dreamweaver notices that you are making changes to a code snippet that is currently being used on the website. It will ask you a question. Click the **Update Button**.

**STEP FOUR:** In the **Update Pages** dialog box, check the box for **Show Log**.

**STEP FIVE:** Switch to your **index.html** file. Note that Dreamweaver has updated the Library item, but it hasn't saved the changes. Click **Save**.





## Apply Library items to pages

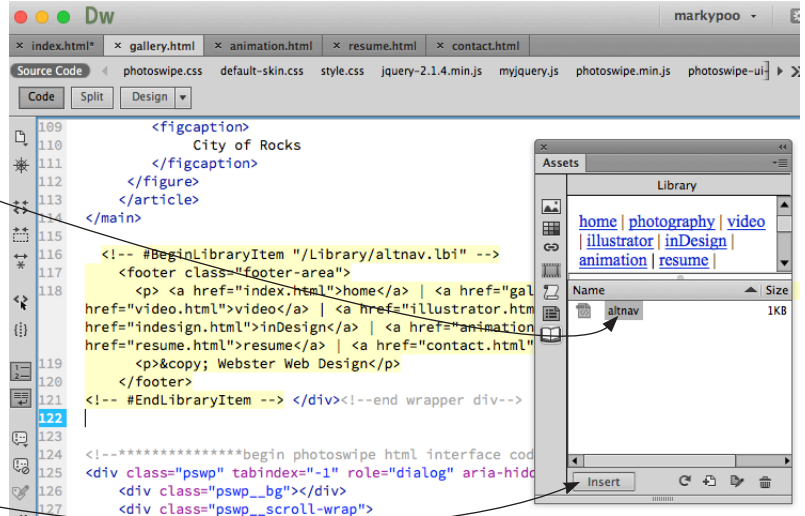
**STEP ONE:** Open all your html pages at once. Starting with **gallery.html**, scroll down to the **footer** and delete it.

**STEP TWO:** In your Assets>Library panel, touch the **altnav** item in the lower window.

**STEP THREE:** Make sure your cursor is blinking in the code view of **gallery.html** where you want the footer div to be located.

**STEP FOUR:** Click the Library>**Insert** button. Save the file.

**STEP FIVE:** Repeat steps one through four on all your html files.



**NOTE:** The next time you add a new page to your website and need to edit the footer, simply edit the library item. The pages don't even need to be open for Dreamweaver to get it right. I've done 200 pages at once, in about 30 seconds.

If Dreamweaver gives you grief with this Library function, it will be caused by one of these two issues:

**(A.)** You did not define your site correctly when you opened Dreamweaver

or

**(B.)** You are trying to do it over the campus network. Dreamweaver doesn't always allow the Library function to work over the campus network. Move the website to your desktop/flash drive. Define your site to point at the new local location and try again

If it goes completely wonky, close Dreamweaver. Open your files in Notepad and manually delete all the library items from the webpages. Delete the library folder and restart Dreamweaver. Re-define the site and try again.

**STEP SIX:** Navigate your website using the alternate navigation links. **NOTE:** there isn't currently an **indesign.html** page. I left it in there figuring you would create one using the **save as** technique. Depending on whether you want the Photoswipe or Flexslider effect:

**save as** photography (gallery.html) [photoswipe] or  
save as illustrator.html [flexslider]



## Getting fancier

In case any of you want to add a column to the right of your photoswipe thumbnails, similar to what is on [websterart.com](http://websterart.com), here is the structure I used. By now you probably have the skills to use the chrome inspector tool to find my css code that makes that all work. But this should get you started.

