Digital Art Club

Workshop 2: Webdesign
Expressing Yourself Online
February 27th, 2012

Start from the basics, then learn to program a complete page at “Putting it Together”. See further resources to see more.

Table of Contents

THE BASIC TOOLS: HTML, CSS, AND JAVASCRIPT ................................................................. 3

A Short Note on Conventions, Browser Issues, and Other Specifications .................................. 3

A SHORT WORD ON USING PROGRAMS .................................................................................. 3

TAGS: HTML ELEMENTS ........................................................................................................ 4

Head ........................................................................................................................................ 4

<meta> ................................................................................................................................. 4
<title> .................................................................................................................................... 4
<link> .................................................................................................................................... 5
<script> ............................................................................................................................... 5

BODY ....................................................................................................................................... 5

Headers .................................................................................................................................. 5
Paragraph .............................................................................................................................. 5
Linebreak ............................................................................................................................... 5
Styling Font: Bold, Italic—Strong, Emphasized ........................................................................ 6
Making Lists: <ul> and <ol> ................................................................................................. 6
Images ..................................................................................................................................... 6
Links ....................................................................................................................................... 6

CSS: MAKING YOUR TEXT BEAUTIFUL ............................................................................... 7

Inline-styling vs. External Style Sheets ..................................................................................... 7
Using Classes and IDs to target styling ................................................................................... 7
THE BASIC TOOLS: HTML, CSS, AND JAVASCRIPT

So you want to learn how to make a website. Here are your tools: HTML, CSS, and JavaScript. In a more advanced tutorial we’ll go over the latter, but in this one we’ll just touch on the basics.

HTML stands for Hyper Text Markup Language, which is a fancy way of saying a special language that gives each set of words a special property. HTML is not programming, as you will see. It consists of a set of tags <> wrapped around the content you want to display.

CSS stands for Cascading Style Sheets, which adds properties to each tag <>, like color, bold, height, width, etc. It is even less of a programming language than HTML.

JavaScript is a programming language, but we won’t touch on it in this tutorial.

A Short Note on Conventions, Browser Issues, and Other Specifications

In this tutorial, I’ll be teaching you HTML4. There is a nice new shiny version called HTML5; unfortunately it is not completely supported—lovely old Internet Explorer which hogs 50% of the browser share does not like it. The chart from here illustrates this:

As you can see, IE only supports HTML5 at version 9. NINE! So when developing, you can pretend not to care about IE and just go straight to HTML5, or worry about the 50% still on IE. There is a very simple hack around this, using JavaScript, but we’ll leave JavaScript away. If you are interested in HTML5, Dive into HTML5 is a fantastic free book that explains everything you need to know, but does assume you know HTML4 already.

I will, however, be using CSS3. CSS3 adds some nice fancy effects, which means the website will just look a little uglier in IE.

A SHORT WORD ON USING PROGRAMS

I will be using Komodo Edit (free program) on my PC. Textwrangler is a popular Mac program, and Notepad++ is the other popular PC program.

Dreamweaver and other such programs are evil if you are looking to learn to make websites.
TAGS: HTML ELEMENTS

Here’s a basic structure for a web document:

```html
<!DOCTYPE html>
<html>
<head>
  Information for browser goes here, is not visible
</head>
<body>
  Content of your webpage, visible to the viewer
</body>
</html>
```

The `<!DOCTYPE html>` declares this is an HTML document, something left over from the vestiges of old browser wars. `<html>` begins the webpage, and it closed by `</html>`.

**Head**

The head presents information to the browser, but is not displayed to the viewer. Commonly in the header:

```html
<head>
  <meta charset="utf-8"/>
  <meta content="digital art, digital art club, harvard" name="keywords" />
  <meta content="Harvard’s only Digital Art Club" name="description" />

  <title>Digital Art Club</title>

  <link rel="stylesheet" href="style.css" type="text/css" media="screen, projection"/>
  <link rel='shortcut icon' href='images/favicon.ico' type='image/x-icon' />  
  <script src="js/jquery.min.js" type="text/javascript"></script>
</head>
```

**<meta>**

The meta tags are for special use.

```html
<meta charset="utf-8"/>
<meta content="digital art, digital art club, harvard" name="keywords" />
<meta content="Harvard’s only Digital Art Club" name="description" />
```

UTF-8 is the encoding of the document; this is enough to include all English characters. The others provide descriptions for search engines.

**<title>**

The title of the website is between the `<title>` tags.

```html
<title>Digital Art Club</title>
```
The stylesheet, style.css file, is linked by <link>. Also linked is a shortcut icon—a picture you see in your tab when visiting the site. Also mobile icons will use <link> to do so.

Any scripts are added by <script> tag.

**BODY**

The body contains all the content of your webpage. Some common elements:

```html
<body>
  <h1>Hello, World!</h1>
  <p>Welcome to my first webpage.</p>
</body>
```

Save your file as index.html. Open it in a browser, and you should see a welcome message!

**Headers**

The several header tags make nice distinctions from boring paragraphs. They are `<h1>`, `<h2>`, `<h3>`... etc, up to `<h6>`.

**I'm in h1**

**I'm in h2**

**I'm in h3**

**I'm the wee little one in h7**

**Paragraph**

The paragraphs are closed in `<p>`. These are automatically styled to have spaces between each one.

**This is a paragraph**

**This is another paragraph**

This gets annoying if you want two things right underneath each other, though. What if you wanted the two lines of text right below each other?

**Linebreak**

So that’s where line breaks come in handy! Here they are called `<br/>` (note the slash, it means this tag closes on itself) and the code will produce image shown in the side.
This is a paragraph
And I'm a separate line that won't be spaced weird

**Styling Font: Bold, Italic—Strong, Emphasized**
If you remember old-school HTML, `<b>` would bold font, `<i>` would italicize it, and `<u>` would underline it. `<u>` stays the same, but `<b>` and `<i>` are old-hat and won’t work anymore. Instead, use `<strong>` to bold, and `<em>` to italicize.

*I'm bold...er...strong!* *I'm italicized...er...emphasized!* *I'm underlined.*
There’s some other ones you can play with as well. See more at the wonderful [WC3 schools](http://www.w3schools.com).

*I'm striked text!* I'm some code! I'm a subscript!

**Making Lists: `<ul>` and `<ol>`**
Lists are made by calling a list tag, then wrapping each list element in a `<li>` tag. `<ul>` are *unordered* with bullets for default, `<ol>` are *ordered* with numbers for default.

```html
<ul>
  <li>one</li>
  <li>two</li>
  <li>three</li>
</ul>
<ol>
  <li>one</li>
  <li>two</li>
  <li>three</li>
</ol>
```

<table>
<thead>
<tr>
<th>Unordered list:</th>
<th>Ordered list:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• one</td>
<td>1. one</td>
</tr>
<tr>
<td>• two</td>
<td>2. two</td>
</tr>
<tr>
<td>• three</td>
<td>3. three</td>
</tr>
</tbody>
</table>

**Images**
Images are added with `<img>` tag.

```html
<img src="http://ninja.jpg.to/" alt="Ninja"/>
```

The address to the image is in the src. To stay compliant with web standards, you must provide an “alt” text, which will appear if your image doesn’t load. This line of HTML will give you an image of a ninja. Note the img tag is a self-closing one.

**Links**
Linking to other pages is done by the anchor `<a>` tag.
<a href="http://google.com" target="_blank">Click me to go to Google!</a>

The link address is within href. The target part is optional, but target="_blank" means the page should open in a new window. Between the anchor tags is what the viewer sees to click.

You can have the user click on an image replacing the text with <img>.

You can jump to another part of the page with #, given that you gave an element an id. More info on this later.

There’s also more stuff, such as making tables, put I suggest you to go to w3schools for more information on those!

**CSS: MAKING YOUR TEXT BEAUTIFUL**

CSS makes your text beautiful. It controls properties like background color, text color, fonts, borders, and position. The easiest way to learn CSS is by actually using it, so I won’t go into too much detail into each possible CSS properties, except when we need them. But a little about CSS conventions...

**Inline-styling vs. External Style Sheets**

Let’s see the effects of CSS with some inline styling.

<p style="text-align:center; color:red;">This text is centered and red!</p>

This text is centered and red!

You can add style to any tag, whether <h1> or <p> or <li>. This is useful when you just want to target one specific thing. This, however, is considered highly bad style. You should separate out your styles into one place, so that you leave the markup and the styles separate. You can add CSS to your <head>, like so.

```
<style>
  p{
    color:red;
    text-align:center;
  }
</style>
```

Now you’ve taken all the styling away from the <body>. This says that on any tag <p>, you give it a color red, and text-align it centered. This is better, but still you have CSS cluttered with your HTML. Take it out completely with a <link>.

```
<link rel="stylesheet" href="style.css" type="text/css"/>
```

Now put all your styles in the style.css file. Refresh your page, and your changes will appear.

**Using Classes and IDs to target styling**

The previous examples were annoying though, because they targeted EVERY <p>. What if you want to do only one?
Use id= to give a particular <p> a special name, and style on that.

```html
<p id="red">This is red and centered.</p>

#red{
  color: red;
  text-align: center;
}
```

Only the <p> with id="red" should be red.

Use a class when you want to target multiple sets of a certain tag. For example, let's say I want two paragraphs blue.

```html
<p class="blue">This is blue</p>
<p>This won't be blue.</p>
<p class="blue">This also will be blue.</p>

.blue{
  color: blue;
}
```

This is blue

This won't be blue.

This also will be blue.

Note ids are called with #, classes are called with a period. Use ids when you have a unique style for one element. This is essential for JavaScript later! Use classes when you want to style multiple items the same way.

**On Colors: Names, HEX, RGB**

I've been lazy and using color names for CSS. There are some default supported ones, like white, red, blue, smokewhite... see them all [here](#).

What you should use to get more unique ones are HEX codes. They consist of six characters. #FFFFFF is white, #000000 is black, #FF0000 is red...read more about it [on Wikipedia](#). RGB is another formatting, but instead of using the hexadecimal system, it runs from 0-255.

```css
#white{
  color: #ffffff;
  color: rgb(255, 255, 255);
}

#black{
  color: #000000;
  color: rgb(0, 0, 0);
}
```

You can find HEX codes and RGB values in Photoshop when you click on the color.
Some Sample Styling

Let’s style some links, which by default are ugly, blue, and underlined.

Let’s also be the pride in our web design.

<a> I’m an unstyled link.</a>

<a> has some special states. Test them out!</a>

```css
a:link{
    background: #800;
    color: #ffffff;
    font-family: arial;
    text-decoration: none;
    display: block;
    padding: 10px;
}

a:hover{
    background: #000;
    color: #800;
}
```

Now I’m in style!

Display, Position, Box Model, Floats—All Important, but complicated to explain

The sample link styling above touched upon some complicated parts of CSS—display and padding. Another confusing CSS property is position and float. Know they exist; read more about positioning, display, the box model, and floats on w3schools. The following example will try to help explain these theories.
PUTTING IT TOGETHER


Lots of websites have free templates. This is a relatively simple one that we can use to put together everything we learned. Prepare a document called *index.html* which is the webpage, and a *style.css* file that will have all the styles. Let’s start with making a header!
Grabbing the Image Files

We should save all the image files from the psd file to use later in our template. Make a new images folder for all the graphics we’ll be using in the webpage.

A Quick Rundown on Image File Formats

Note that if you crop anything and save it as a JPEG, you will get it with the gray background of the entire document. Let’s say you decided to change the background of your website to be black; your logo will be useless because you have a gray background that doesn’t fit in with the rest!

We need to use an image format that supports transparency. Here is where PNG comes in handy. First, in the Creative Studio psd file hide all the layers that include a background until you see the gray grids. This template grouped the layers beautifully, so all you have to do is hide the background layer and the background folder.

Now crop out the logo, and save as logo.png. Your logo is an image with a transparent background, and will now play nicely with any sort of backdrop you want to give it, even patterns.

Slices: A Fast Way to Save Multiple Images from Photoshop

Besides the logo, however, we need a bunch more images from this Photoshop file. Slice tool can help us here.
I sliced out all the images I’ll need from the document, including the logo, the centerpiece picture, the arrows next to the headers, the checkboxes, the portfolio thumbnails, the Connect With Us icons, and the smaller logo on the bottom. I made all the backgrounds transparent so I can save them as PNG.
Go to File -> Save for Web and Devices. A huge dialog will popup with lots of options. Note the preview screen has a white background. Select the slices you want to save, or hit ctrl+a to select everything. Change preset to PNG-24, hit save.

Now choose the location you wish to save all your files. Photoshop will automatically create an images folder if you do not have one, so if you already made an images folder, be on the level outside of it. See screenshot below:

Change **Save as type** to **Images Only**, and **Slices** to **All User Slices**. Click **Save**.

Ta-da! All your slices are now in the images folder. You might have some extra ones if your slices were funky, and you’ll want to change the names to be more descriptive.
After cleaning up and changing to memorable filenames:

Now to code!

The Header
Simple, hopefully self-explanatory now:

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>Creative Studio</title>
  <link rel="stylesheet" href="style.css" type="text/css">
</head>

Let's add the body at the end.

<body>
</body>
</html>
```

Basic Styling: Browser Reset
The first thing you should add to your CSS file is a browser reset. Each browser—Firefox, Chrome, IE (please tell me you’re not using IE) has different defaults for spacing out elements. To make sure your design stays consistent across different browsers, do a small reset of values that you can add in later.

```css
body, a, p, h1, h2, h3, h4, img, ul, li{
  padding: 0px;
  margin: 0px;
}
```
Selecting Background Color and Fonts

Looking at the Photoshop document, you can see the background is an off-white. Check what the HEX code is by using the color picker. The HEX can be found at the bottom—#f4f4f4. Next, check the fonts used by selecting the text tool and clicking on a text box. This document is all written in Arial.

Our style sheet is then:

```css
body{
  color: #f4f4f4;
  font-family: arial, sans-serif;
}
```

The list of multiple fonts I have is called *font-stacking*. Let’s say you wished to use a fancy font, Tw Century MT, but other computers may not have it. You can set your font stack to “Tw Century MT”, arial, sans-serif so that computers will load the first font if they have it; then move to the second; then finally use a default sans-serif font if all else fails.

There are ways of embedding fonts so that this wouldn’t matter, but we’ll leave that to another tutorial.

Another piece of general styling I like to get out of the way is links (none of that underlined ugly blue!) Get the HEX for the orange using Photoshop color picker.

```css
a{
  color: #fe9b00;
  text-decoration: none;
}
```

```css
a:hover{
  color: #000000;
}
```
The Body
I am going to chop the body up into the header, centerpiece image, main, and footer.

```html
<body>
  <div id="header">
    Has the logo, the navigation
  </div>
  <div id="centerpiece">
    Has the image
  </div>
  <div id="main">
    Has all the textboxes, portfolio, and updates
  </div>
  <div id="footer">
    Tiny info on the bottom
  </div>
</body>
```

Each section is in its own `<div>` group, and gave them ids. This means everything inside `<div id="header">` `<div>` such as the logo, the navigation, will all be in the horizontal row up top.

Header
The header is made of the logo and the menu.

```html
<div id="header">
  <div class="container">
    <h1>Creative Studio</h1>
    <a href="index.html"><img src="logo.png" alt="Logo" /></a>
    <div id="nav">
      <ul>
        <li><a href="">Home</a></li>
        <li><a href="">About Us</a></li>
        <li><a href="">Services</a></li>
        <li><a href="">Blog</a></li>
        <li><a href="">Contact Us</a></li>
      </ul>
    </div>
  </div>
</div>
```

Quite a bit for a short header! Let's grow through it bit by bit.

The `.container` class is yet another `<div>` group, but for styling purpose we'll later see.

Between the `<h1>` tag is the title of the site, which we'll later hide. Search engines use the `<h1>` tag to find your website, but this website has an image for its title, so we'll have to cheat a bit to work around this.

 `<img>` links to the logo. Common practice lets it so that when I click on the logo image, it will lead back to the home page.

Standard convention puts navigation bars in an unordered list. I also wrapped it in its own `<div>` called `#nav`. 
Styling the Header

Let’s style the header. Open style.css, the external file you’ll be using to keep your CSS. We need to move the logo to the left, the navigation to the right, and somehow center everything on the page to a certain width.

Centering

This is easy. The .container class we made earlier? CSS trick will help you here! First, let’s check the width of the document using the ruler tool, measuring from one side of the centerpiece image to the other. (Measure tool is hidden under the eyedrop tool.) The width appears at the top—it is 960px.

Add this to your style sheet:

```css
.container{  
    width: 960px;
    margin: 0 auto;
}
```

This gives your header a width of 960px and centers it on the page. Margin 0 auto does all the centering work for you.

Styling the Navigation

Next is the navigation bar. We put it in a list, so it’s kind of odd-looking right now...

Easy stuff first—let’s get rid of the bullets, and make all the text uppercase.

```css
#nav ul{
    list-style: none;
    text-transform: uppercase;
}
```
The #nav ul part says for the unorderlist list in the #nav element, change everything to be like so.

Next, let’s make the list come out as a row, rather than stacked. Select the li element this time, in #nav li:

```html
#nav li{
    display: inline;
    padding: 30px 20px;
}
```

I also added some space between the items using `padding`. This adds extra space between the elements—I used a shortcut method that says *add 30px padding to the top and bottom, then a 20px between to the left and right of each list element*. This is the same as stating:

```html
#nav li{
    display: inline;
    padding-top: 30px;
    padding-bottom: 30px;
    padding-left: 20px;
    padding-right: 20px;
}
```

But much less verbose.
**Floats**

Here is the tricky bit—we need to move this navigation to the right of the page. Selecting the #nav element, apply a style like so:

```css
#nav{
    float: right;
}
```

Now it’s up! Floats are tricky because they take items out of the normal flow. Read a better explanation [at W3schools](https://www.w3schools.com). For an exploration of why to use them with caution, see what happens if you apply the float to the #nav ul element instead:

Or the #nav li element:

Fun activity for you at home exercise: Figure out how floats work! :D But for now, back to float: right; to the #nav div.

But it’s too close to the top. So I’m going to add a margin-top to the #nav element to push it down a bit. For a discussion of why I used margin instead of padding, read more about the [CSS box model](https://www.w3schools.com/css/css_boxmodel.asp).

Yay!
**Making Dividers**

Back to the Photoshop template – there’s a gray line in between the header and the centerpiece. There are multiple ways to achieve this effect—I am going to use border bottom.

Imagine the entire header as a box. I want to give the bottom of that box a gray border. Our box is `<div>` with id `header`.

```css
#header{
    border-bottom: 3px solid #ccc;
}
```

If you open up the webpage though, that doesn’t look right…it’s stretched all the way across the entire browser window. Why?

If you recall, the logo and the nav are all nicely centered thanks to a margin: 0 auto trick applied to the `.container` class. The #header is outside of that container, so the border of #header naturally is not confined to the width and centering of `.container` class.

One fix is to apply the border-bottom to the `.container`:

```css
#header .container{
    border-bottom: 3px solid #ccc;
}
```

Now it’s not stretched over the entire width.

However, when I look at the entire template page, there’s TONS of these grey dividing lines. I don’t want to have to apply a border-bottom to each element. Luckily, HTML has a horizontal rule element we can use. Put `<hr />` in your header, inside `.container` class. Style it as so:

```css
hr{
    border: none;
    background-color: #ccc;
    height: 3px;
    margin: 20px 0px;
}
```

I gave it a margin of 20px on top and on bottom so that things would be spaced out nicer. **Fun exercise:** See what happens when you change `margin` to `padding`. [CSS Box Model](https://developer.mozilla.org/en-US/docs/Web/CSS/box_model) at work!
Finally, just to fix up more spacing issues, I added a margin-top to #header, so that everything is pushed down a little from the top of the page. I also decided the skinny font of the navigation menu looks bad, so I made it bold.

```css
#header{
  margin-top:20px;
}
#nav li{
  display: inline;
  padding: 30px 20px;
  font-weight: bold;
}
```

**Centerpiece**
That was a LOT of typing for the header. Next part is easy! Add a .container class to #centerpiece, and place the image in.

```html
<div id="centerpiece">
  <div class="container">
    <a href="#portfolio"><img src="images/center.png" alt="image" /></a>
  </div>
</div>
```

The href to #portfolio means clicking on this link will take you to the element called portfolio. We don’t have an element named #portfolio yet, so clicking this will go nowhere for now.

You should need no styling on this part to make it look right.
Main

Next we go into the #main. Again, add a .container to center everything.

Using Floats

These three boxes are in one row, so I am going to put each group in its own div and give them a similar class. Because they take up a third of a page, I will call them .third.

```html
<div id="main">
    <div class="container">
        <div class="third">
            <h2>About Us</h2>
            <p>Some things about us blah blah blah blah blah blah blah blah blah blah</p>
            <a href="#">Read</a>
        </div>
        <div class="third">
            <h2>Benefits</h2>
            <ul>
                <li>Here are some benefits</li>
                <li>Another benefit</li>
                <li>And another one</li>
            </ul>
        </div>
        <div class="third">
            <h2>Testimonials</h2>
            <p>A great testimonial blah blah blah blah blah blah blah blah blah</p>
            <p>A Great Person</p>
        </div>
    </div>
</div>

.third{
    width: 300px;
    float: left;
    padding-right: 20px;
}
```

Note 3 boxes: width of 300px * 3 + padding-right of 20px * 3 = 960px, the width of the container. Try increasing the width of anything, and see what happens...
Styling with Background Images

The h2 has a nice arrow to its left. We can achieve that with background-image property of CSS, along with some padding to move the text over.

```css
.third h2{
  text-transform: uppercase;
  border-bottom: 5px solid #000;
  margin-top: 30px;
  margin-bottom: 15px;
  font-size: 16px;
  padding-bottom: 5px;
  padding-left: 20px;
  background: url('images/arrow.png') no-repeat;
}
```

Unfortunately, the bullets are oddly spaced. This is because we reset ul up in the body reset. Add some padding to the ul element to fix this.

```css
.third ul{
  padding-left: 20px;
}
```

Custom List Images

Same thing with lists, you can use your own image instead of the default bullet. Add a little padding to increase space between each list item.

```css
.third li{
  list-style-image: url('images/check.png');
  padding-bottom: 10px;
}
```

Exercise: Style the “Read” and the “Awesome Person”

Try it! Give them a class or id and add some styling.
Fixing Floats: Using the clear: both;

Finally, let’s add the divider below the three text boxes. Add the `<hr />` again.

```html
<div class="third">
  <h2>Testimonials</h2>
  <p>A great testimonial blah blah blah blah blah blah blah blah blah blah blah</p>
  <p class="name">A Great Person</p>
</div>

<hr />
```

However, if you refresh your page...it’s not there.

If you are using Chrome or Safari to view your page, you can do a little debugging by right-click -> Inspect Element. A dialog appears on the bottom, with the HTML on one side, and CSS properties on the right. Hovering over each HTML line gives you a ton of information about your code, including the location, height, width, and box model for each of your elements. Use this with abandon, it’s a great tool to help you.

Let’s try finding out what went wrong with the `<hr>` and why it’s not appearing. Find it on the HTML side by expanding the arrows, and hover over it.

The debugger claims that `<hr>` is right there, next to testimonials. But invisible. Why?
Remember the float: left; property we added to the .third class for the boxes? This float is still in effect; which means anything that comes after it will be positioned strangely. Try adding some text after the <hr> you made. It should pop up in a weird place.

To fix this, you have to clear the floats; this will fix your document and allow it to flow normally again. You apply the clear property after a set of elements you floated. In this case, it would be convenient to add it to HR.

```css
hr{
    border: none;
    background-color: #ccc;
    height: 3px;
    margin: 20px 0px;
    clear: both;
}
```

Now it’s right.

### Combining CSS styles

The <h2> wrapping around OUR PORTFOLIO has a lot of the same styles that we just added to .third h2. So does the smaller headlines, BLOG UPDATES and CONNECT WITH US, which will be wrapped in <h3>. CSS allows you to share styles, so I am going to make the styles apply to ALL <h2> and <h3>. Use a comma.

```css
h2, h3{
    text-transform: uppercase;
    font-size: 16px;
    margin-top:30px;
    margin-bottom: 15px;
    border-bottom: 5px solid #000;
    padding-bottom: 5px;
    padding-left: 20px;
    background: url('images/arrow.png') no-repeat;
}
```
However, only the .third h2 has a black border, so I take it out of the h2 statement and add it only to .third h2.

```css
h2, h3{
  text-transform: uppercase;
  font-size: 16px;
  margin-top: 30px;
  margin-bottom: 15px;
  padding-bottom: 5px;
  padding-left: 20px;
  background: url('images/arrow.png') no-repeat;
}
.third h2{
  border-bottom: 5px solid #000;
}
```

**Linking to Parts on the Same Page**

I gave OUR PORTFOLIO an id #portfolio. Scroll back up to the center image. Remember how you had linked the image to `href="#portfolio"`?

Click on the image, and you’ll be taken down to the portfolio `<h2>`.

**Practice: Portfolio, Blog Updates, Connect With Us, and Footer**

These I will leave to yourself to do! Use floats. Clear floats. Use classes. Use ids. There’s many ways to implement the bottom portion, I leave it to your discretion. I will, however, go over a small part of the footer that needs to be remarked on. Also, there is one black bar at the very top that you will also need to add. I just gave up on the contact email at the top right. :(

**VALIDATE**

Now to validate our code. Go to W3Schools HTML Validator at [http://validator.w3.org/](http://validator.w3.org/). Choose validate by direct input, and copy and paste all your code in.

All my warnings were forgetting to add the alt attribute to image tags; easily fixed.

This is really fun to do with professional websites – for example, take a lot at the results of [harvard.edu](http://harvard.edu)! Tons of easily fixed mistakes. Tsk tsk.
FURTHER RESOURCES

W3schools is the to-go place to learn basics, and explains concepts much better than I. They also have the full references for every property, so you can play with all the attributes to your content.

Smashing Magazine has an excellent collection of tutorials and design inspiration. Their hand-picked CSS collections has lots of great tips to take you beyond beginning level webdesigner.

CSS-tricks has a collection of awesome code snippets to use, and good tutorials to browse.

Thanks for reading! Let me know if you have any questions at yzhou@college.harvard.edu.

Source files for the workshop can be found here.