**Personal Portfolio**

In this tutorial we’ll use HTML and CSS to create a basic website about yourself. By completing this tutorial you will learn how to use basic HTML tags, semantic tags, how to apply CSS from an external file, and how to use online fonts.

Throughout this class we will be using Cloud9 (C9) as our development environment. C9 combines a powerful online code editor with a full Ubuntu workspace in the cloud. 

Here is a completed version of the project:

[Personal Portfolio](http://cst336.herokuapp.com/projects/portfolio/)

By developing this project, you will be able to:

1. Describe the importance of creating a sketch before developing a website
2. Use basic HTML tags to:
   1. Add paragraphs <p>
   2. Add line breaks <br>
   3. Add images <img src=”example.jpg”>
   4. Add Links <a href=”example.com”>
   5. Bold text using <strong>
   6. Add comments <!-- -->
   7. Create ordered and unordered list
   8. Create a basic table with data <table>
   9. Create <div> containers
3. Use Semantic tags to:
   1. Create header, footer, and side sections.
4. Apply inline, internal and external CSS:
   1. Use classes and id to apply style
   2. Change background and text color
   3. Create a navigation menu
   4. Center text and content
5. Use online fonts

|  |
| --- |
| **Planning!** Take a moment to think about our goal for this project and how we would go about achieving it.     1. When designing a website or web page, it is helpful to create a mockup or wireframe. This is especially helpful when working as a part of a team, since all team members can have the same visual reference. 2. Identify the sections and elements that you will need (images, navigation, links, footer, etc.) 3. Create the folder structure that you’ll use for the website. |

**Lesson 1: Planning and Website Mockup**

Planning is the most important part, don’t skip it!

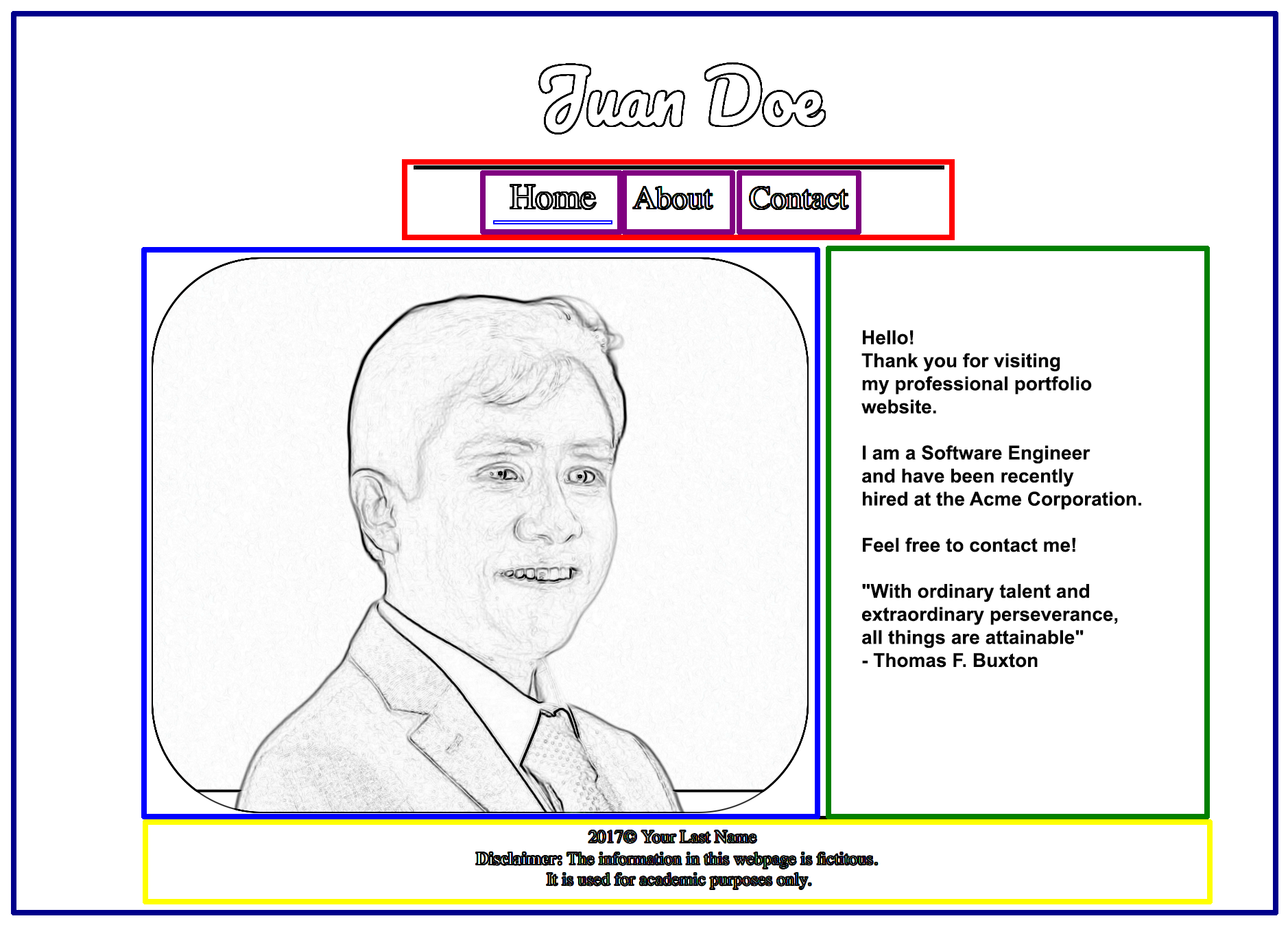
Before we start writing any code for our website, it’s best to take some time to decide what you want it to contain and how to structure it.

For our portfolio home page we want to have:

* Website title
* Navigation
* An excellent, professional portrait
* A little blurb welcoming the user
* A footer with copyright info

Next we need to think about how to layout the web page. We can create a basic mockup of the website’s structure on paper or something more sophisticated like Photoshop.

Here is the basic mockup of the Home Page we will be creating:



It’s handy to imagine web pages as a **set of containers** which can be nested inside of each other. In this example, we see the body (in dark blue) wraps the entire website. Inside the body, we have the navigation (in red), the portrait (in light blue), blurb (in green), and footer in yellow. Notice the navigation is further broken down into individual links (purple).

By placing all of these elements into their own containers, or **semantic** and **<div>** tags, we can position and style them individually. Also, by separating them out, we can easily make changes to one element without disturbing the rest of the web page.

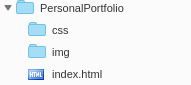
**Lesson 2: Preparing the Folder Structure**

Let's add the folder structure!

**2.1:** Download and uncompress the zip file containing the starting point for our website here: <http://cst336.herokuapp.com/projects/portfolio/portfolio.zip>

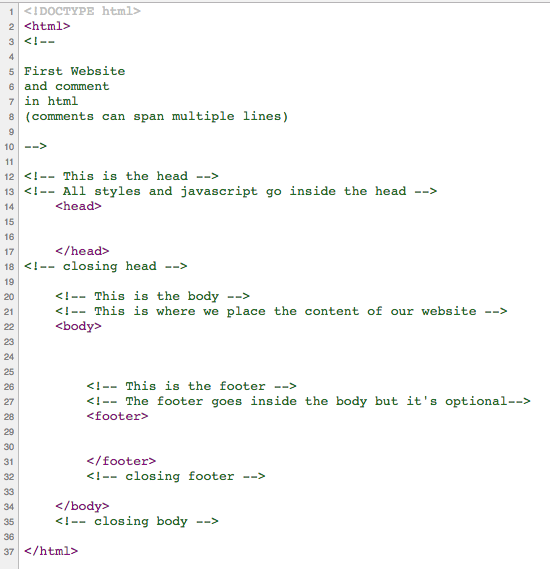
**2.2:** Upload the project folder to your Cloud9 workspace

**2.3:** This is the folder structure you should have:



**Lesson 3: Scripting your first website**

Let's create the code structure!

**3.1:** Open your file called **“index.html” :**

Let’s analyze its current content:

* **<!DOCTYPE html>** (line 1) is required and tells the browser to treat the file as a standard HTML 5 file,
* **<!-- -->** (lines 4 to 10) Defines a comment
* **<head>** (line 14) Defines information about the document, such as title, author, etc.
* **<body>** (line 23) Defines the document's body
* **<footer>** (line 30) Defines a footer for a document or section, which is optional.

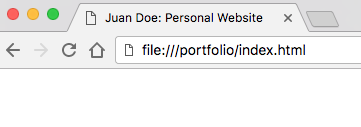
**3.2:** Update the Head section

There are two HTML tags that should be included always as part of the Head section:

<meta char=”utf-8” /> and <title></title>. Include these lines within the Head section (use your own name instead)



The “charset” line specifies the character encoding to be used by the browser. UTF-8 is Unicode, which is the most comprehensive character set currently available.

The content of the <title> tag will NOT be displayed within the web page itself. If you save the document and open it in a browser, you’ll realize that the title is shown as part of the tab of the web page. But the content of the page is blank.

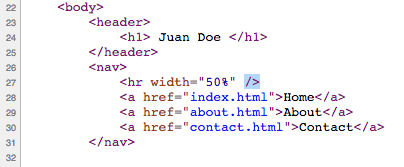
The Head section also includes the CSS and JavaScript code. We’ll add CSS later on.

|  |
| --- |
| **Notes:**    Some HTML tags expect content, so they need an opening and a closing tag, like  <title> </title> and <h1> </h1>  Other HTML tags do not expect any content, so the best practice is to end the tag by using a slash before the angle bracket, like: <meta charset=”utf-8” **/>**  HTML tags are NOT case sensitive. However, it’s best practice to stick with either upper or lower case throughout the website to keep consistency. |

**3.3:** Add the header and navigation within the Body section

All the content that will be displayed within a web page must be included within the <body> tags.

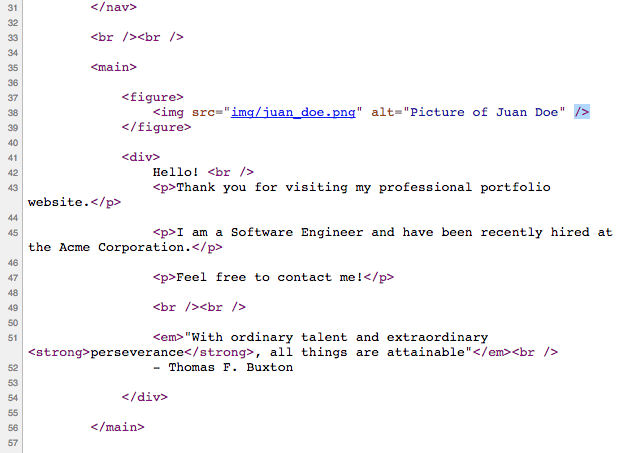
Add the following code to the web page, right below the <body> tag:

****

* **<header> (**line 23**)** defines a semantic section as a header
* **<h1> - <h6>** (line 24) defines HTML headings; gets smaller as number gets higher
* **<nav>:** (line 26) defines a semantic section as a navigation menu.
* **<hr />** (line 27) defines a thematic change in content; a “line” with a width of 50%. Notice that the <hr /> tag is not expecting any content, so it’s closing itself (there is no </hr> tag)
* **<a>** (lines 28 to 30) create a hyperlink (when clicking on “Home” it will open the file “index.html”, when clicking on “About” it will open the file “about.html” and so on).

**3.4:** Add an image and content

Add the following code right below the closing </nav> tag added in the previous step.



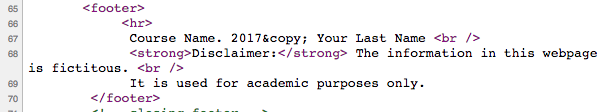
* **<br />** (line 33) creates a line break (carriage-return). It doesn’t need a closing tag.
* **<main>** (line 35) defines a semantic section in a document that is expected to include the main content
* **<figure>** (line 37) defines a semantic section that is expected to include an image
* **<img>** (line 38) displays an image. It doesn’t need a closing tag. Uses two attributes: “**src**”, which is the image source and “**alt**”, which is alternative text for blind users.
* **<div>**  (line 41) defines a container within the document
* **<p>**  (multiple lines) defines a paragraph
* **<em>**  (line 51) defines emphasized text, displayed in *italics*.
* **<strong>**  (line 51) defines strong emphasized text, displayed in **bold**

**The ‘img’ folder contains a default image of “juan\_doe”, but you should upload your own image to the folder and reference it in the <img> tag.**

|  |
| --- |
| **Bad Practice!** The <img> tag provides a “width” and “height” attributes that allow you to modify the display size of the image, however, these attributes do NOT change the actual image size.  So, if you upload a big image that is 2,000 x 2,000 pixels it will still take a lot of bandwidth and time to be downloaded, even if the values of the “width” and “height” attributes are 20.  The best practice is to change the actual image size using an image editor software. |

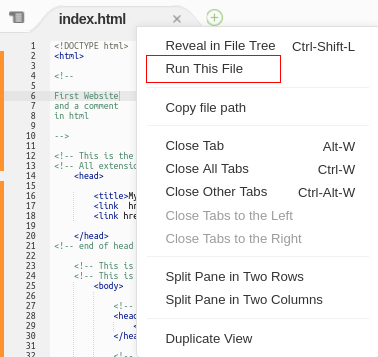
**3.5:** Create the **Footer**

The footer will allow us to display information about our website. And a disclaimer telling the user that this information is for academic purposes only.



* **&copy** (line 67) creates a copyright symbol

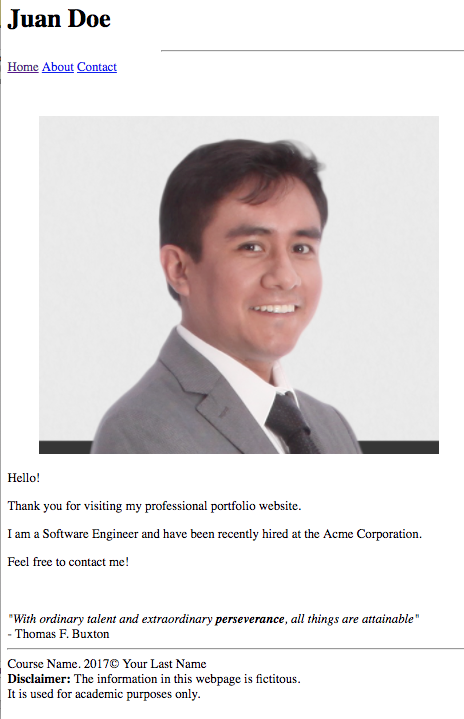
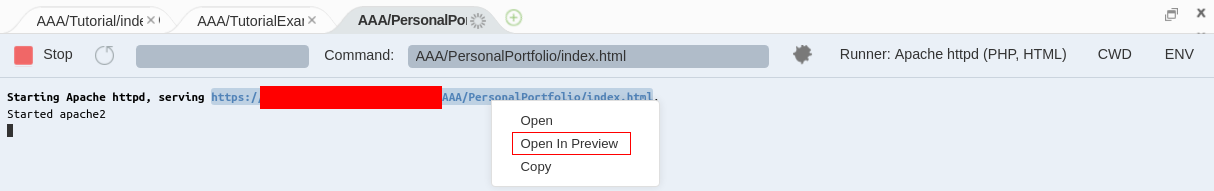
Let’s run the code!



Right click on “index.html”

Click “Run This File”

Now go to the console of index.html and right click the link and select open in preview



You should be able to see something like the screenshot to the right.

So far we have added the content and also the web page structure (the semantic tags).

We can proceed now to style the page by adding CSS rules.

Before moving on, make sure to have replaced Juan Doe’s name and picture with your own.

Lesson 4: Adding CSS Styles to Home Page

CSS (Cascading Style Sheets) is the code you use to style your webpage (i.e., change color, fonts, borders, spacing, etc.) There are three ways to apply styles: inline, internal and external.

**4.1** Applying Inline Styles.

Inline styles are included within the HTML elements themselves. Let’s change the background color of the entire web page by using an inline style. Modify the <body> tag in this way:



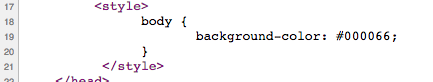
Save and open the file. You should see that it has a black background and white text now.

While you might need to use inline styles in rare occasions, its use is highly discouraged because it makes the code difficult to maintain. Don’t remove the inline style just yet, we’ll remove it later on.

|  |
| --- |
| **Bad Practice!** It is bad practice to use inline styling as it clutters code. Also, you would need to make many changes in case you decide to update some styles. A better practice is to use external styles. |

**4.2** Applying Internal Styles.

Internal styles are included within the <head> section. They are especially convenient when the styles are intended for just one web page. Add the following code within the <head> section, right below the closing </title> tag:



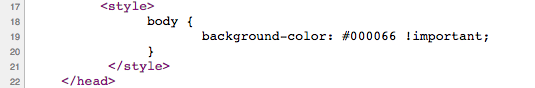
Be sure to type correctly the **<style></style>** tags.

We have just added a CSS rule. The **selector** in this rule is the “body”, that is, the whole web page. The **property** is the background-color, and the value is a dark blue color represented by its hexadecimal value #000066, For more hexadecimal values you can use [colorpicker.com](http://colorpicker.com)

However, if you save and open the document in a browser, you’ll notice that the background is still black! The reason is that the inline style takes priority.

A way to change the priority of inline style is by using the keyword: !important

Change the new style rule to read:



If you save and open the page, you’ll see the new background color now.

As you may see, internal styles are an improvement to inline styles. However, when you have multiple pages using the same styles, internal styles are not the best option because you’d need to replicate the same CSS rules as many times as web pages are. In those situations it is more convenient to use an external style file.

**4.3** Applying External Styles.

Let’s start by removing the inline style created in step 4.1 for the <body> tag. Also, remove the internal style created in step 4.2. Remove also the <style></style> tags.

Right above the closing </head> tag, add the following line:



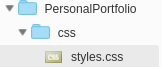
The **<link>** tag we have just added in line 17 is linking this web page with an external CSS file (which we haven’t created yet). The **href** attribute specifies the path to the CSS file: based on its value it should be within the “**css**” folder and the file name should be “**styles.css**”. Be aware that filenames ARE case sensitive in web servers.

The **rel** attribute has the value of “stylesheet”, which specifies that this external file is indeed a stylesheet. This attribute is required.

The **type** attribute used to be required but now it’s optional since stylesheets are always text.

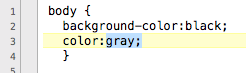
|  |
| --- |
| **Notes:**    There are actually two ways to link an external CSS file. Using the **<link>** tag is one way.  Another way is to use an @import directive within the <style> tags, like this:  <style>  **@import url("css/style.css");**  </style>  The @import method is newer and it wasn’t fully supported in all browsers when it was introduced but nowadays either method is supported, so you can use either one.  Be aware that the <link> tag must NOT be included within the <style> tags. |

**4.3.1** Add a new file within the **css** folder and called it “**styles.css**”. Note that you can use any other file name, such as “portfolio.css”, however, the name must match the name used in the **<link>** tag.



(new file created)

**4.3.2** Styling the body of the page. Add the following CSS rule to the **styles.css** file:



Save the CSS file and refresh the “Home” page in your browser. You should see that the page has a black background and the text is gray, not white.

|  |
| --- |
| **Troubleshooting:**    If you don’t see the styles applied to your web page, take the time to debug your code. 99% of the CSS problems are caused by **typos**. Make sure you typed everything correctly, including the file names (they are case sensitive). Some potential reasons your CSS code might not work:   1. You might not be linking the external CSS file at all. To ensure you are linking it, view your web page source in your browser and click on the link to your CSS file (all browsers provide a feature to view the source code). 2. You might have non accepted characters in the css file. Your CSS file must not include any HTML tag, not even <style>. Must not include HTML comments either. Any odd character will break your CSS code. 3. In odd situations, your browser might cache the CSS style and would ignore new style rules. To force the browser to take the new rules, just open the CSS file in your browser directly. Optionally, you can use these key commands to override the cache: Windows: CTRL + SHIFT + r Mac OS X: Command + Shift + r |

**4.3.3**. Centering the Content.

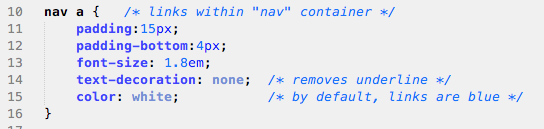
Add the following CSS rule to center the content in all the semantic containers. Notice that you can apply the same rule to multiple HTML elements if you separate them with a comma.



Save the CSS file and refresh the web page. Now all content should be centered. Try removing any of the four semantic containers listed in the CSS rule and you should see that the content for that container is aligned to the left, which is the default alignment.

**4.4.4** Styling Navigation.

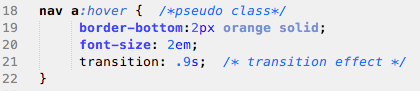
Type the code below to style the navigation menu. Pay attention to the syntax to add comments within CSS. (In most IDEs, including C9, you can use CTRL + / to add comments).



Save the file and see the changes. Take a minute to explore different values on each of the above properties.

|  |
| --- |
| Why do you think the selector of the menu links was “**nav a**” instead of just “**a**” ?  ?When you use an HTML tag as the selector of a CSS rule, the styles are applied to ALL elements with the same tag. So, if there was any other link within the page, it would have the same style as the navigation menu, which would look odd.  By using “**nav a**” as the selector, we are restricting the styles to just the links that are included within the “nav” container. Notice that there must NOT be a comma or any other character between “nav” and “a”, only a blank space. |

Now, let’s add something fancy! The following CSS rule is using the “:**hover**” pseudo class, which means that the styles are going to apply only when hovering over that specific element. You can add the :hover pseudo class to any HTML element.



**4.4.5** Positioning Content Side by Side.

Currently, the image is showing above the welcoming blurb but we really want them to be side by side. We need to use the “**float**” property to have containers next to each other. This property must be applied to all containers that will be next to each other.

Which are the two containers that need to be used as selectors for the CSS rule? If you take a look at your HTML code, the image is included within the **<figure>** container and the welcoming blurb is included within a **<div>** container.

So, the selectors could be:

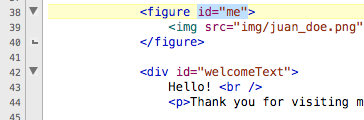
**figure, div** {

float: left;

}

However, recall that if we apply styles to HTML elements directly, all other similar HTML elements will inherit the same styles. In other words, any other image that we use in the website and the content of any other “div” container will be floating next to other elements. To avoid that, let’s add an “**id**” to these HTML containers.

Go back to edit your “**index.html**” page and add the “id” attribute to the “figure” and “div” containers, as shown below in lines 38 and 42.



|  |
| --- |
| **Notes:**    The purpose of an “id” as the attribute of an HTML element is to identify that element to access it via CSS or JavaScript.  The id **must be unique** in the entire web page. Think of it as a variable name: they are case sensitive and must not have any spaces. |

Now that we have added the id’s, we can use them as selectors… however, they MUST be preceded by a hash tag (#).

Copy the following code in your **styles.css** file.



Save the CSS file and refresh the browser. The containers should be now next to each other… however, the footer section might be in the wrong place!

When floating elements, browsers keep floating some of the following elements as well. We need to tell the browser to stop or clear the floating. Given that the following element is the **footer** container, let’s add the following CSS rule:



Now, if you refresh your browser, the footer should be below the two side-by-side sections.

Refer to [w3schools](https://www.w3schools.com/css/) for more styling options

|  |
| --- |
| It’s Your Turn!   Before moving on, add more content and styles to accomplish the following:   1. Add the CSUMB logo at the bottom of your web page, within the footer.     2) Make the footer text smaller and with different text color (your choice!)  3)The welcome text should be aligned to the left, not the center.  4) In the navigation menu, the item corresponding to the current page should stand out. In other words, “Home” should stand out in the “index.html” page. |

Lesson 5: About Me Page

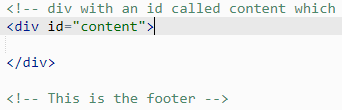
Let's expand our website beyond a single page!

**5.1:** Now that we have the content of our main page finished, let’s create an about page. Right click the “portfolio” folder in cloud9 and select “New File”. Call the file **about.html**.



**5.2:** Open up **about.html.** Instead of typing the navigation, content div, and footer again, let’s copy everything from index.html. Make sure that **the styles.css link is still in the header**. Delete everything within the <main> container, including the <main></main> tags.

For this webpage will be using a “<div>” container just to demonstrate that you can use either: a semantic tag (like <main>) or a <div> container with an id.

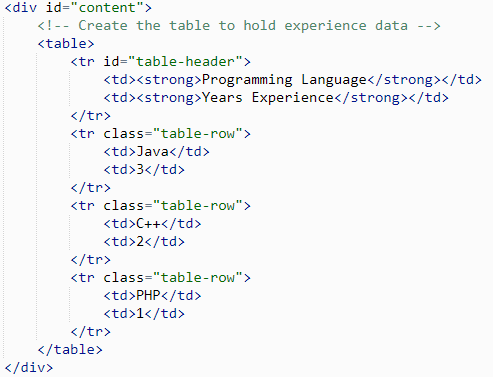


Save the file and now try the ‘About’ link in the menu. You will see we have the same menu and footer, but the contents of the page is blank and ready for us to add content.

**5.3:** On the **about.html** page we’re going to display our programming experience with different languages. A good way to represent this kind of data in html is with a **<table>**.

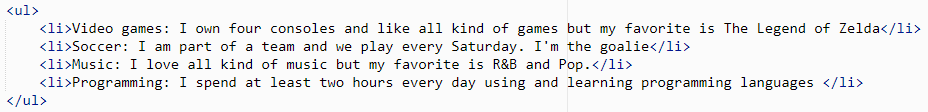
Inside each **<table>** we need to use:

* **<tr>** to create rows
* **<td>** for individual pieces of table data (cells)

****

You will notice we added the id **“table-header”** to the header **<tr>** tag and **“table-row”** to the other **<tr>** tags. This will allow us to style them independetly of each other in the stylesheet.

**5.4:** In addition to the programming experience table, let’s include a little about our interests. We can use a list for this purpose. Start by creating a new unordered list using the **<ul>**. Inside the list tags, we create new list items with **<li>**.

****

**5.5:** With table of data in place, all we need to do is add a few CSS rules to our **styles.css** to spruce it up a little. We can start by centering the table by changing to **margin** on the left and right to **auto**.

When you provide only two values to margin, the first is applied to the top and bottom, the second is applied to the left and the right.

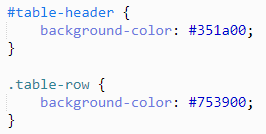
53_tblstyle.PNG

**5.6:** We can make it a touch more legible by adding **10px of padding** and more consistent with the homepage by increasing the **font-size to 1.3em**.

In order to make the header of the table stand out, we can specifically target **<strong>** tags inside of **<td>** tags and provide additional styling to them. This is done by putting the parent tag followed by the child tag. In our case: **‘td strong’**.

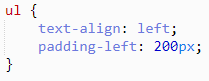


**5.7:** Using the **class** and **id** attributes we put on the rows of our table, we can change the style of them independently. The hash “**#**” character means id, the full stop or period “**.**” means class.



|  |
| --- |
| **Notes:**    To style our table, we used an **id** for the header and a **class** for the row. This is because we are only going to have a single header, whereas there are multiple rows of the table.  As a rule:   * Use **Id** to **Identify** a single element and style that. * Use **Class** to style an entire **classification** (multiple) of elements. |

**5.8:** Our table of interests needs a touch of styling as well. We should fix the center alignment inherited it inherited from the content div and add some padding to better center the bullets within the page.

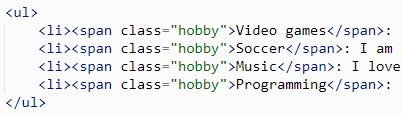


|  |
| --- |
| **How would you do it?** Let’s say you wanted to add color to each of the hobby nouns (the part before the colon), how would you do this? Given what you know about css and html, which tags would you style? |

**▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁**

**5.9:** In order to style some element on a page, it needs to be contained within html tags. If you want to style just the hobby nouns on the list (video games, soccer, etc), we need to put tags around them. For this we will use the **<span>** tag with a class we can reference in the stylesheet.

**about.html**

****

**styles.css**

**hobbycss.PNG**

|  |
| --- |
| **Notes:**    So if we need to section off a part of the page content to style, like the hobbies above, why not use a **<div>** like we did before? This is because **<div>** is a **block** element and **<span>** is an **inline** element.   * Block elements are not friendly and do not allow other elements to sit next to them. * Inline elements are very friendly and will gladly sit next to other elements.   You can change any any element to be inline or block by using the **‘display’** property in your stylesheet. |

**The result:**



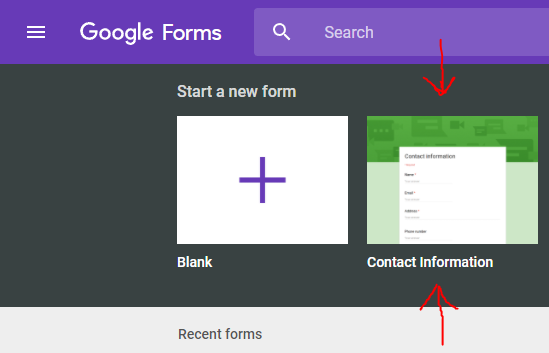
Lesson 6: Contact Page

Let's let friends contact us!

**6.1:** Next we can move onto the Contact page. Like the about page, we will copy over our index.html and remove everything inside **<div id=”content”>.**

**6.2:** For this page, we’re going to use a [**Google Form**](https://www.google.com/forms/about/) to handle the contact feature. Later in the semester we will create our own forms, but that requires PHP server programming. Besides, Google Forms are super slick and work great!

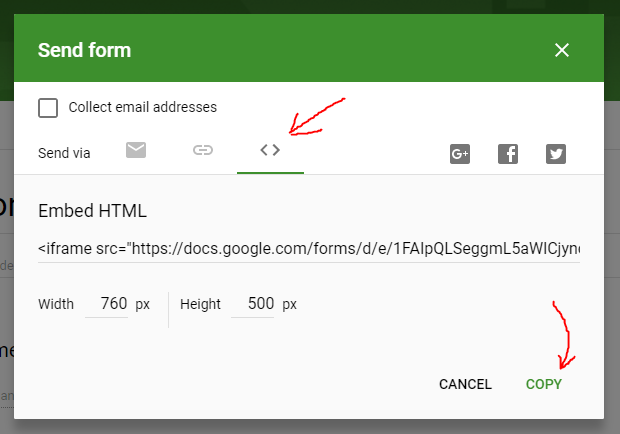
Go to <https://docs.google.com/forms> while logged into your Google account and select the **Contact Information** template from the gallery at the top.<



**6.3:** On the form edit page, make any changes you would like and when you’re done click the **Send** button in the upper right hand corner.

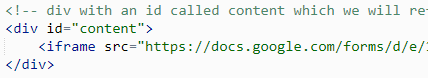
****

**6.4:** When the Send Form dialog pops up, click the menu option that looks like HTML tag brackets (**< >**)and click the copy button to add the **<iframe>** html text to the clipboard.

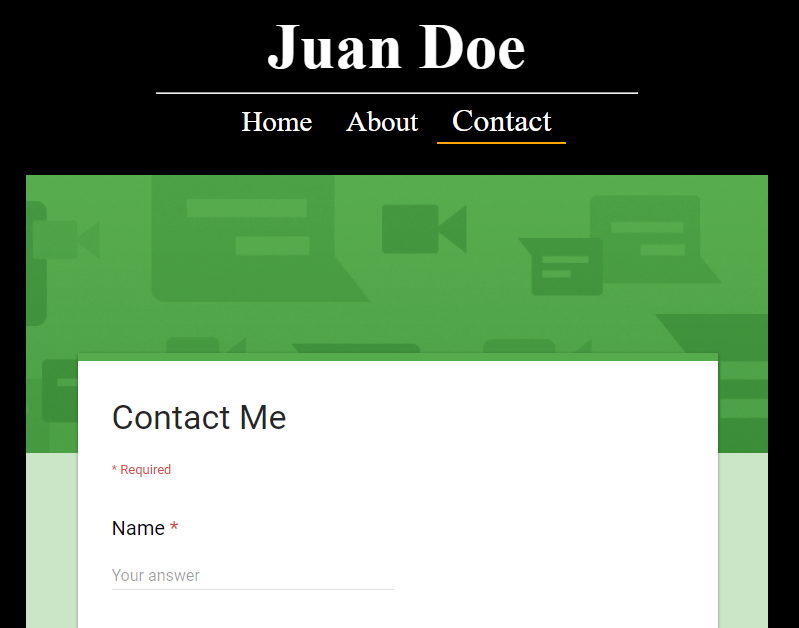


The HTML tag **<iframe>** is used to embed a “frame” or window of one website into another. The Width and Height properties allow you to select how big the window frame is.

**6.5:** Now that we have the text for our iframe form copied, let’s paste it into our contact page.



**The< result:**



**▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁**

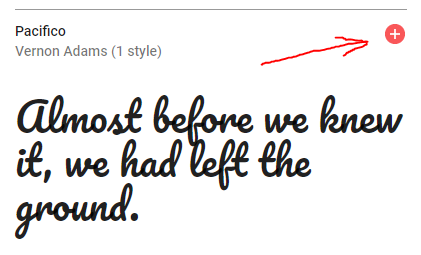
Lesson 7: Using Online Fonts

Let's get some stylish fonts!

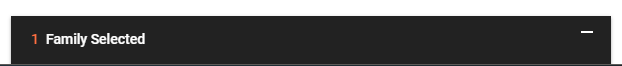
**7.1:** Our website is mostly complete, but the header could use something to help it stand out. One option would be to use an image, but consumes more bandwidth and doesn’t allow for scaling. A better way is to use a different and nicer font than the rest of the web page.

Not all of the fonts that you have in your local computer are available in all browsers, though. Only the most common fonts are available (e.g., Arial, Time News Roman, Verdana). If you want to use a fancy font that can be displayed in any web browser, a good alternative is using an external web font.

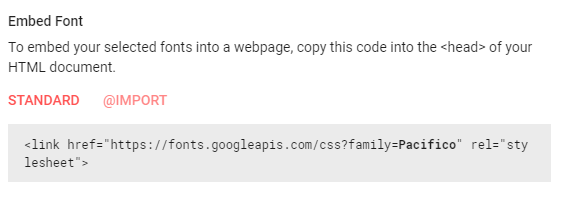
Go to [**fonts.google.com**](http://fonts.google.com) and search for Pacifico. Click the **+** to add the font family to your selection.



Click the box at the bottom to bring up your list of families.



**7.2:** Select the html **<link>** tag in the dialog and copy it into the head of **index.html**, **about.html**, and **contact.html**.





**7.3:** Now that the font is linked we can now use it by changing the **font-family** property in our CSS file. Since we want to use this font on the website header, we can add this rule to the h1 tag.

