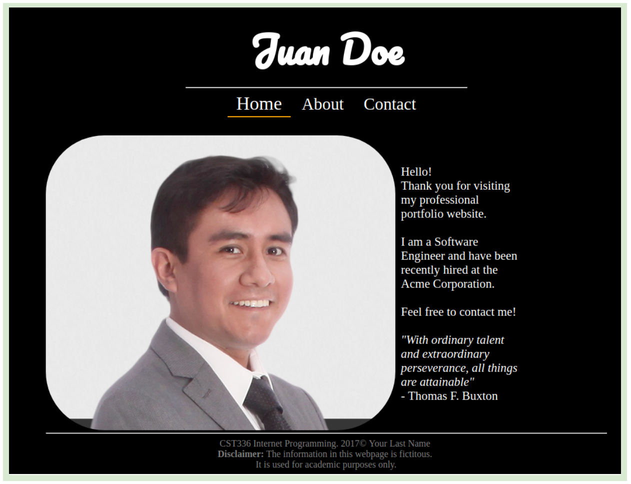
Personal Portfolio

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.

Here is a completed version of the project:
Personal Portfolio



Hello!
Thank you for visiting my professional portfolio website.

I am a Software Engineer and have been recently hired at the Acme Corporation.

Feel free to contact me!

"With ordinary talent and extraordinary perseverance, all things are attainable"
- Thomas F. Buxton

CST336 Internet Programming. 2017© Your Last Name
Disclaimer: The information in this webpage is fictitous.
It is used for academic purposes only.

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:
    a. Add paragraphs <p>
    b. Add line breaks <br>
    c. Add images <img src="example.jpg">
    d. Add Links <a
    e. href="example.com">
    f. Bold text using <strong>
    g. Add comments <!-- -->
    h. Create ordered and unordered list
    i. Create a basic table with data <table>
    j. Create <div> containers
3. Use Semantic tags to:
    a. Create header, footer, and side sections.
4. Apply inline, internal and external CSS:
    a. Use classes and id to apply style
    b. Change background and text color
    c. Create a navigation menu
    d. Center text and content
    e. Use online fonts
5. Embed a Google Form within an iframe

Planning!

Take a moment to think about our goal for this project and how we would go about achieving it.

a) 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.

b) Identify the sections and elements that you will need (images, navigation, links, footer, etc.)

c) 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!

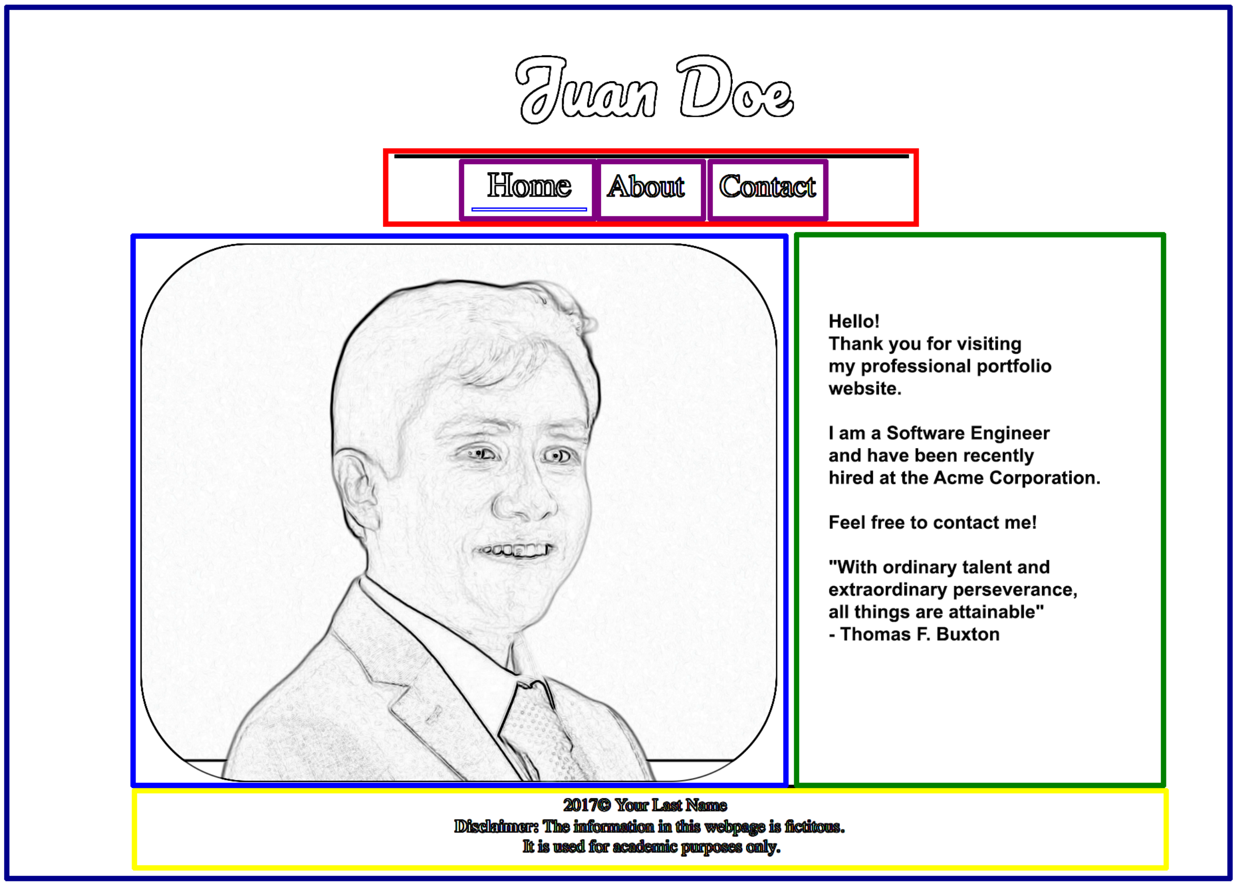
Before we start writing any code for our website, it's best to take some time to decide what you want it to contain and how to structure it.

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:



Hello!
Thank you for visiting
my professional portfolio
website.

I am a Software Engineer
and have been recently
hired at the Acme Corporation.

Feel free to contact me!

"With ordinary talent and
extraordinary perseverance,
all things are attainable"
- Thomas F. Buxton

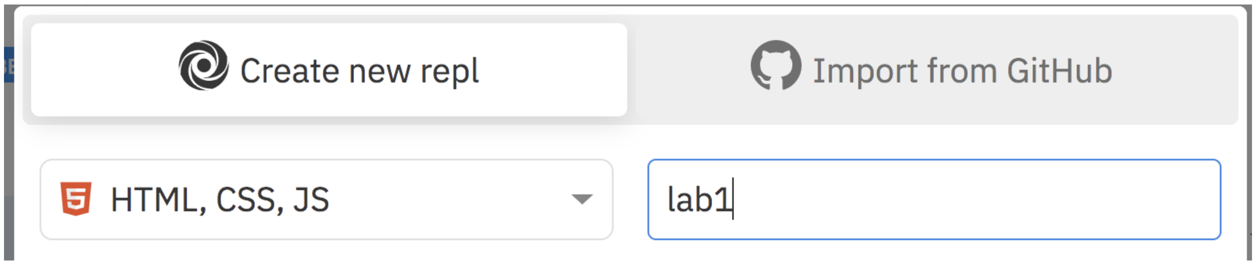
2017© Your Last Name
Disclaimer: The information in this webpage is fictitous.
It is used for academic purposes only.

It's handy to imagine web pages as a **set of rectangular containers** (or boxes) which can be nested inside of each other. In this example, we see the body (the outermost rectangle 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

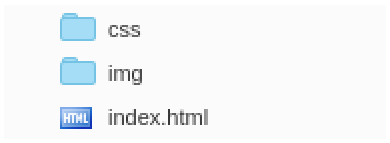
2.1 Create a new repl using “HTML, CSS, and JS” language. Name it **lab1**



2.2 Download and uncompress the zip file containing the starting point for our website here: https://cst336.herokuapp.com/projects/portfolio/portfolio.zip

2.3 Upload the project folder into your repl, replacing the index. html file.

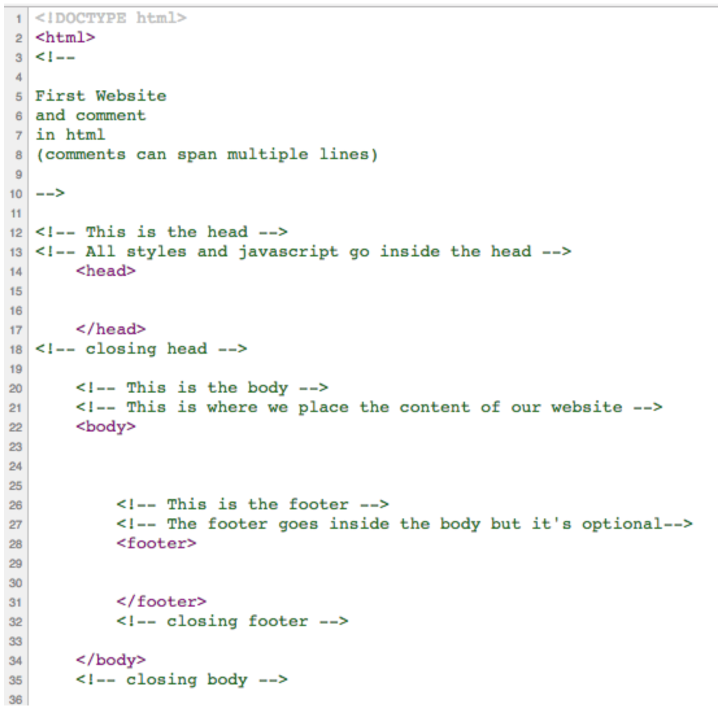
2.4 This is the folder structure you should have within your repl. You’ll need to create the “css” folder manually and move inside it the empty “style.css” file. You can remove the “scrip.js” file, since this lab will not use any JavaScript code.



Lesson 3: Scripting your first website

3.1: Open your file called **“index.html” :**
Let’s analyze its current content:

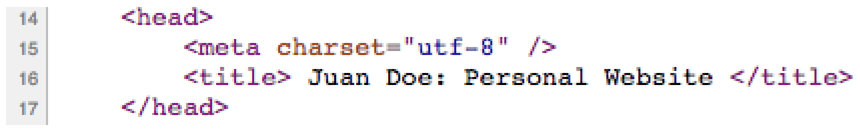
- **<!DOCTYPE html>** (line 1) is required and tells the browser to treat the file as a standard HTML 5 file,
- **<!-- -->** (lines 3 to 10) Defines a comment
- **<head>** (line 14) Defines information about the document, such as title, author, etc.
- **<body>** (line 22) Defines the document's body
- **<footer>** (line 28) Defines a footer for a document or section, which is optional.



```
1  <!DOCTYPE html>
2  <html>
3  <!--
4
5  First Website
6  and comment
7  in html
8  (comments can span multiple lines)
9
10 -->
11
12 <!-- This is the head -->
13 <!-- All styles and javascript go inside the head -->
14     <head>
15
16
17     </head>
18 <!-- closing head -->
19
20     <!-- This is the body -->
21     <!-- This is where we place the content of our website -->
22     <body>
23
24
25
26         <!-- This is the footer -->
27         <!-- The footer goes inside the body but it's optional-->
28         <footer>
29
30
31         </footer>
32         <!-- closing footer -->
33
34     </body>
35     <!-- closing body -->
36
```

3.2: Update the Head section

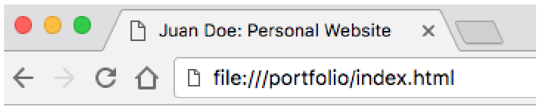
There are two HTML tags that should always be included as part of the Head section: <meta char="utf-8" /> and <title></title>. Include these lines within the Head section (use your own name instead)



```
<head>
    <meta charset="utf-8" />
    <title> Juan Doe: Personal Website </title>
</head>
```

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:

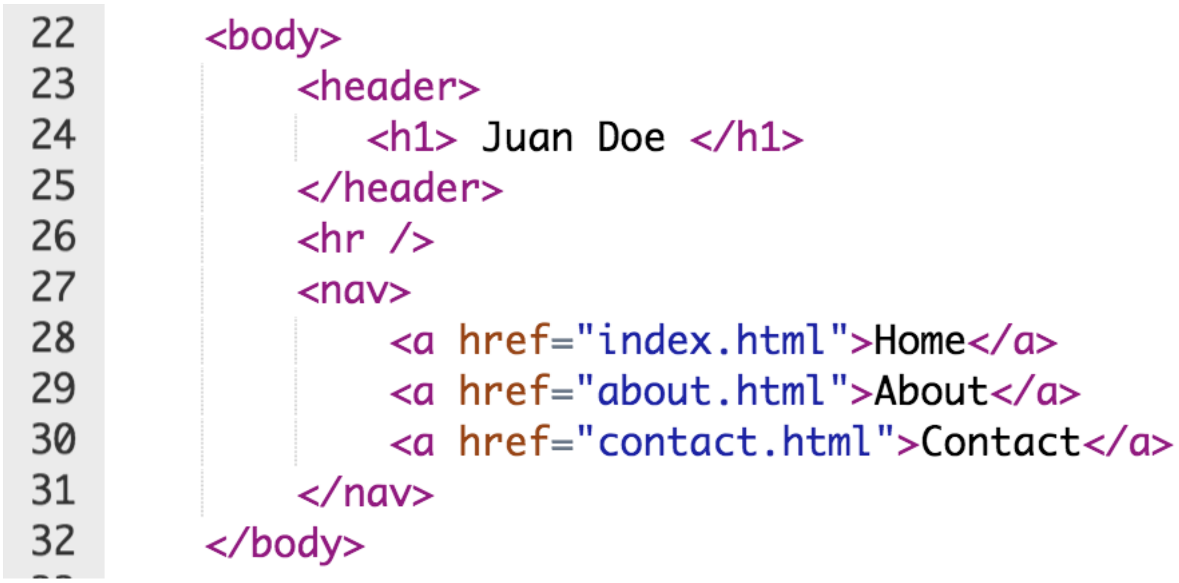
Some HTML tags expect content, so they need an opening and a closing tag, like
<title> </title> and <h1> </h1>

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" **/>**
However, ending empty tags with the slash is optional in HTML5.

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

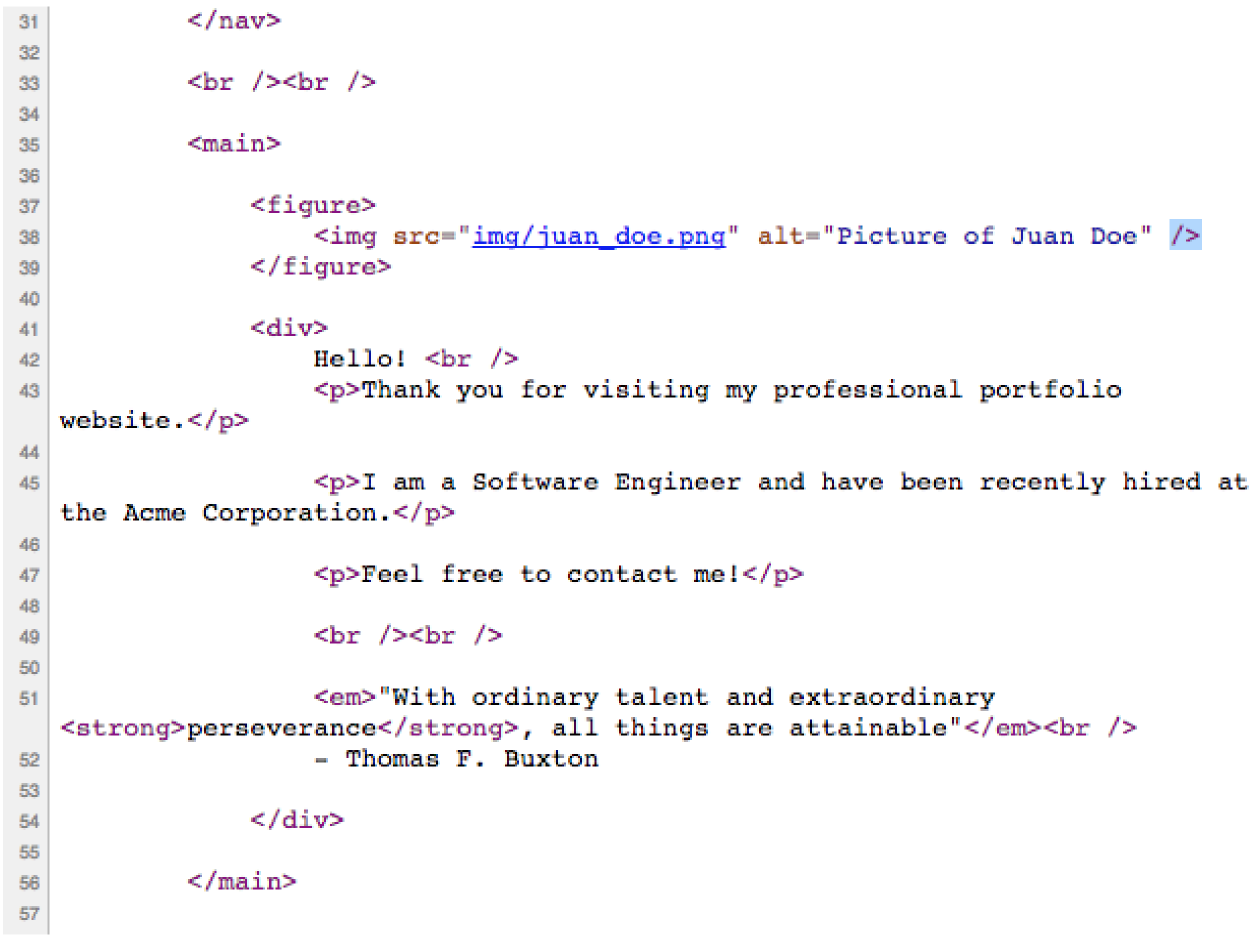
All the content that will be displayed within a web page must be included within the <body> tags. Add the following code to the web page, right below the <body> tag:



```
22      <body>
23          <header>
24              <h1> Juan Doe </h1>
25          </header>
26          <hr />
27          <nav>
28              <a href="index.html">Home</a>
29              <a href="about.html">About</a>
30              <a href="contact.html">Contact</a>
31          </nav>
32      </body>
```

- **<header>** **(**line 23**)** defines a semantic section as a header
- **<h1> - <h6>** (line 24) defines HTML headings; gets smaller as number gets higher
- **<hr />** (line 26) defines a thematic change in content, visually represented by a line or "horizontal rule". Notice that the <hr> tag is an **empty** HTML element (it has no content, therefore, there is closing </hr>). There are several other empty tags, such as <br>. The empty tags can be closed by having a forward slash: <hr /> or <br />. However, in HTML5 it's not necessary to close the empty tags.
- **<nav>:** (line 27) defines a semantic section as a navigation menu.
- **<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).

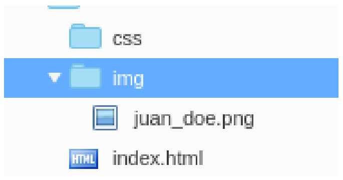
3.4: Add an image and content
Add the following code right below the closing </nav> tag added in the previous step.



```
31          </nav>
32
33          <br /><br />
34
35          <main>
36
37              <figure>
38                  <img src="img/juan_doe.png" alt="Picture of Juan Doe" />
39              </figure>
40
41              <div>
42                  Hello! <br />
43                  <p>Thank you for visiting my professional portfolio website.</p>
44
45                  <p>I am a Software Engineer and have been recently hired at the Acme Corporation.</p>
46
47                  <p>Feel free to contact me!</p>
48
49                  <br /><br />
50
51                  <em>"With ordinary talent and extraordinary <strong>perseverance</strong>, all things are attainable"</em><br />
52                  - Thomas F. Buxton
53
54              </div>
55
56          </main>
57
```

- **<br />** (line 33) creates a line break (carriage-return). It doesn't use 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 visually impaired 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**

Note: 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!

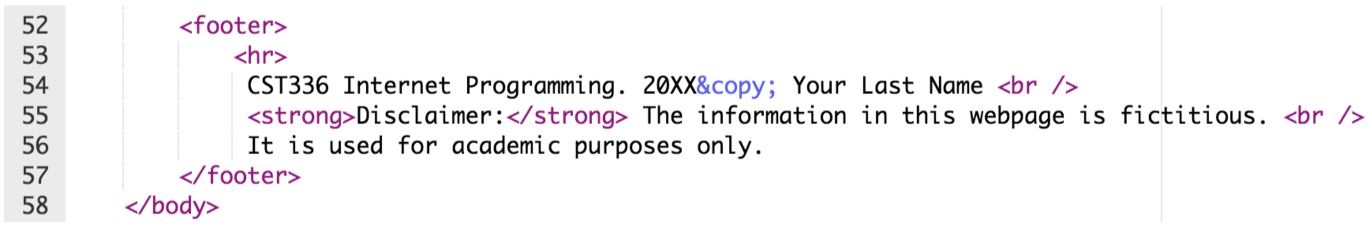
The <img> tag provides a "width" and "height" attributes that allow you to modify the display size of the image, however, these attributes do NOT change the actual image size.

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.



```
52        <footer>
53            <hr>
54             CST336 Internet Programming. 20XX&copy; Your Last Name <br />
55             <strong>Disclaimer:</strong> The information in this webpage is fictitious. <br />
56             It is used for academic purposes only.
57        </footer>
58    </body>
```

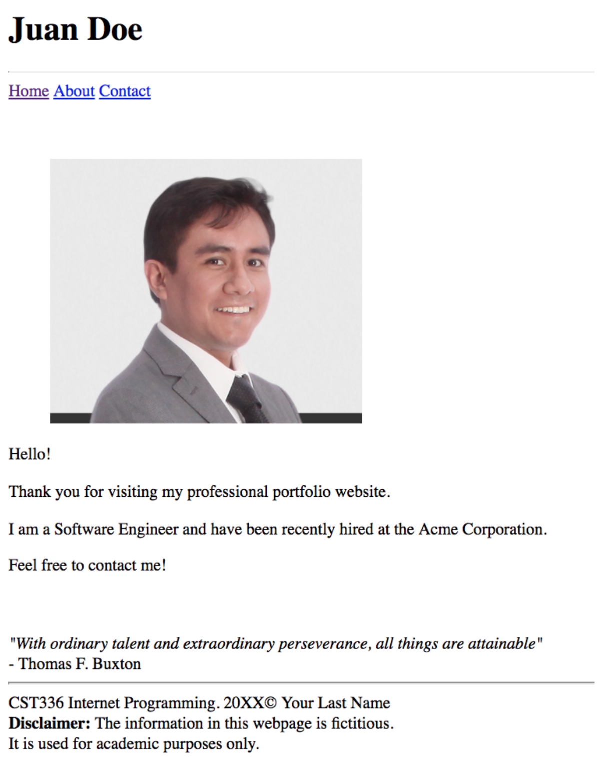
- **&copy** (line 54) creates a copyright symbol.
  (Update the year and your last name)

Let's see our page!

Since we have added just HTML code, if you're working on your local computer, you can just double click on the file name to open it in any browser of your preference.

Note: "index" is a special file name in web servers, when a file is called "index", you don't need to type it as part of the URL. However, you'd need to type it if the file name is anything else.

When opening the page in a web browser, you should be able to see something like the screenshot below. So far we have added the content and also the web page structure (the semantic tags).



Juan Doe

Home About Contact

Hello!

Thank you for visiting my professional portfolio website.

I am a Software Engineer and have been recently hired at the Acme Corporation.

Feel free to contact me!

"With ordinary talent and extraordinary perseverance, all things are attainable"
- Thomas F. Buxton

CST336 Internet Programming. 20XX© Your Last Name
Disclaimer: The information in this webpage is fictitious.
It is used for academic purposes only.

Feel free to keep customizing this page, in addition to replacing the image and name, you can also change the quote and the text.

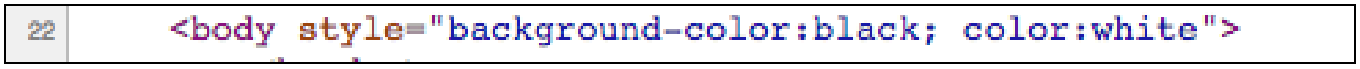
We can proceed now to style the page by adding CSS rules.

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:



```
    <body style="background-color:black; color:white">
```

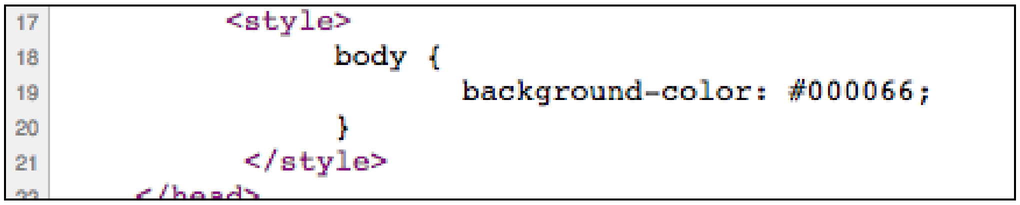
Save and open the file. You should see that it has a black background and white text now.
While you might need to use inline styles in rare occasions, its use is highly discouraged because it makes the code difficult to maintain. Don't remove the inline style just yet, we'll remove it later on.

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:



```
            <style>
                  body {
                         background-color: #000066;
                  }
             </style>
        </head>
```

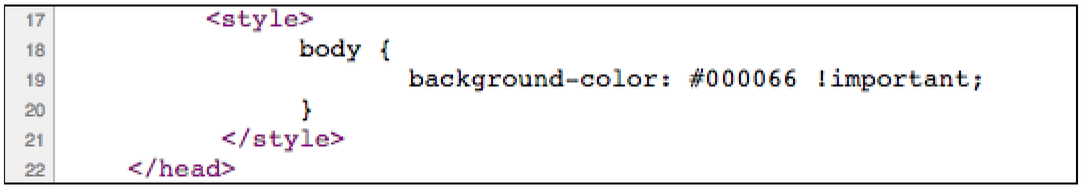
Be sure to type correctly the **<style></style>** tags.

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

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:



```
            <style>
                  body {
                         background-color: #000066 !important;
                  }
             </style>
        </head>
```

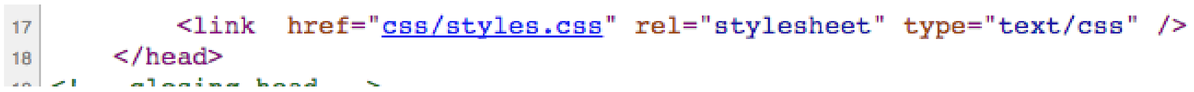
If you save and open the page, you’ll see the new background color now.

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:



```
17          <link  href="css/styles.css" rel="stylesheet" type="text/css" />
18      </head>
```

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. It can have any name but it must have .css extension. 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.

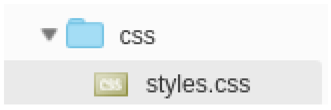
Another way is to use an @import directive within the <style> tags, like this:

```
<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.

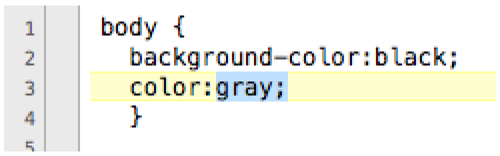
Be aware that the <link> tag must NOT be included within the <style> tags.

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:



```
body {
  background-color:black;
  color:gray;
  }
```

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:

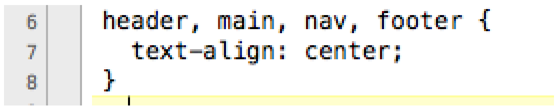
If you don't see the styles applied to your web page, take the time to debug your code. 99% of the CSS problems are caused by **typos**. Make sure you typed everything correctly, including the file names (they are case sensitive). Some potential reasons your CSS code might not work:

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.



```
header, main, nav, footer {
  text-align: center;
}
```

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, you can use CTRL + / to add comments).

```css
nav a {   /* links within "nav" container */
    padding:15px;
    padding-bottom:4px;
    font-size: 1.8em;
    text-decoration: none;   /* removes underline */
    color: white;            /* by default, links are blue */
}
```

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.

```css
nav a:hover {  /*pseudo class*/
    border-bottom:2px orange solid;
    font-size: 2em;
    transition: .9s;  /* transition effect */
}
```

4.4.5 Positioning Content Side by Side.

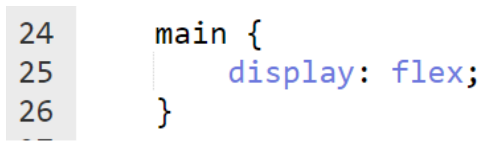
Currently, the image is showing above the welcoming blurb but we really want them to be side by side. CSS provides several ways to create a multicolumn layout. A very popular way is using **Flexbox** (flexible box).

Follow these 3 steps to apply the Flexbox method:

1) Identify the elements that need to be next to each other. 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 elements that need to be within each other are <figure> and <div>

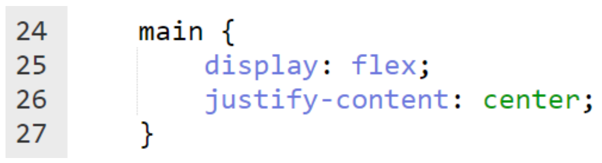
2) Surround all elements that need to be next to each other within another tag... which we're already doing! As you can observe, the **<main>** tag is surrounding these two elements. It didn't have to be "main", it could have been any other HTML element, such as a "div" container.

3) Add the "display: flex" CSS rule to the surrounding tag



```
main {
    display: flex;
}
```

Save the file and open it in a browser. The image and the welcome text should be displayed next to each other now, however, they're not centered. To center it, use the "**justify-content**" property:



```
main {
    display: flex;
    justify-content: center;
}
```

Note:

If there were more <div> containers within the <main> tag, all of them would be displayed next to each other. If you only wanted just the top two next to each other, then you'd need to surround them within a new tag and apply the "display: flex" rule to that new tag.

Refer to w3schools 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) 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.

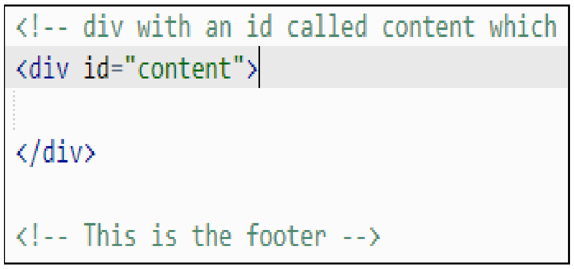
Lesson 5: About Me Page

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 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.



```
<!-- div with an id called content which
<div id="content">

</div>

<!-- This is the footer -->
```

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.

Notes:

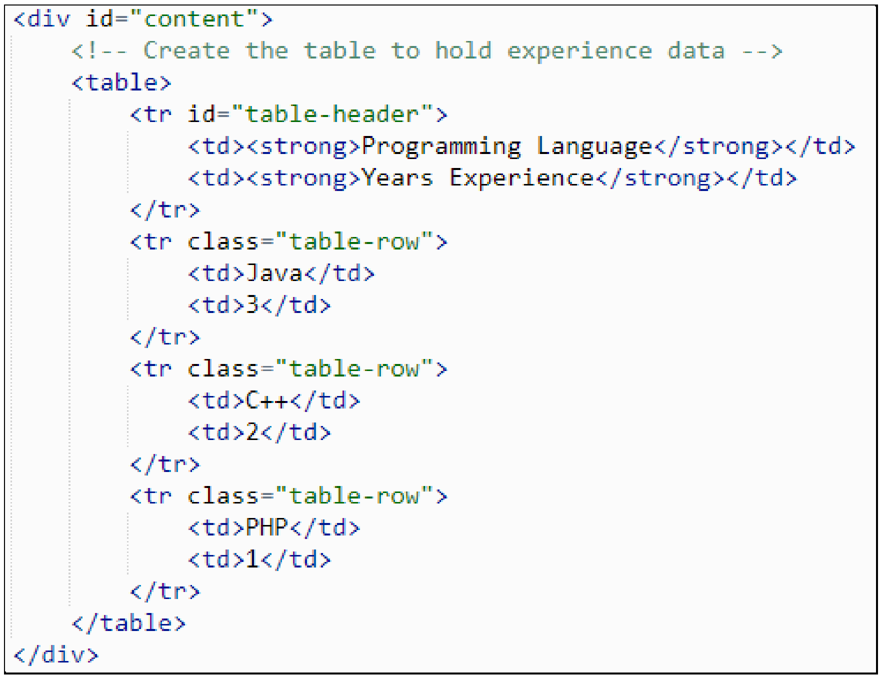
The purpose of an “id” as the attribute of an HTML element is to identify that element to access it via CSS or JavaScript.

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.

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)



```
<div id="content">
    <!-- Create the table to hold experience data -->
    <table>
        <tr id="table-header">
            <td><strong>Programming Language</strong></td>
            <td><strong>Years Experience</strong></td>
        </tr>
        <tr class="table-row">
            <td>Java</td>
            <td>3</td>
        </tr>
        <tr class="table-row">
            <td>C++</td>
            <td>2</td>
        </tr>
        <tr class="table-row">
            <td>PHP</td>
            <td>1</td>
        </tr>
    </table>
</div>
```

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 independently 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>**. Customize this list with your own interests!

```
<ul>
    <li>Video games: I own four consoles and like all kind of games but my favorite is The Legend of Zelda</li>
    <li>Soccer: I am part of a team and we play every Saturday. I'm the goalie</li>
    <li>Music: I love all kind of music but my favorite is R&B and Pop.</li>
    <li>Programming: I spend at least two hours every day using and learning programming languages </li>
</ul>
```

Save the file and open it in a browser window. You’ll notice that the content of the table and the unordered list are NOT centered. You might be thinking that they could be centered by applying the “text-align:center” CSS rule to #content, but they won’t (try it!).

Notes:

To center a block element, such as a Table or a Div container, you must assign the value of “auto” to the properties: margin-left and margin-right:

margin-left: auto;
margin-right: auto;

Optionally, you could use the shortcut version:

margin:0 auto;

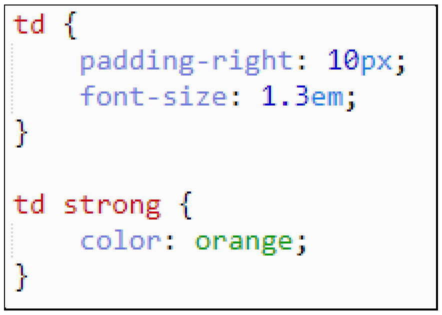
When you provide only two values to margin, the first is applied to the top and bottom, and the second is applied to the left and the right

5.5: Add the following CSS rules to the **styles.css** to center the table:.

```
table {
    margin: 0px auto;
}
```

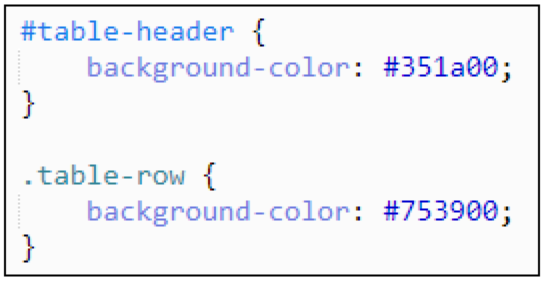
5.6: We can make it a touch more legible by adding **10px of padding** and more consistent with the homepage by increasing the **font-size to 1.3em**.

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'**.



```
td {
    padding-right: 10px;
    font-size: 1.3em;
}

td strong {
    color: orange;
}
```

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.



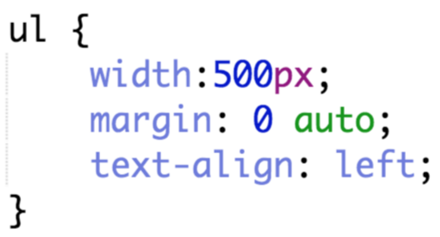
```
#table-header {
    background-color: #351a00;
}

.table-row {
    background-color: #753900;
}
```

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 list of interests needs a touch of styling as well. Use `margin:0 auto;` to center the whole block. You must also specify a width.



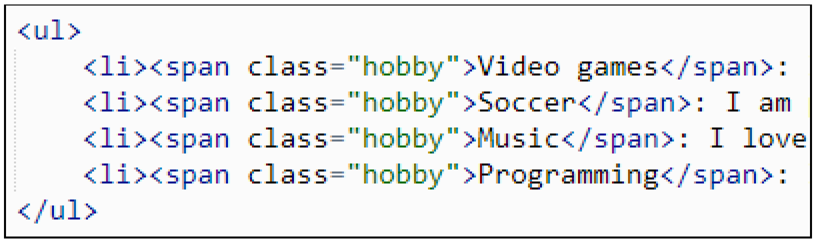
```
ul {
    width:500px;
    margin: 0 auto;
    text-align: left;
}
```

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?

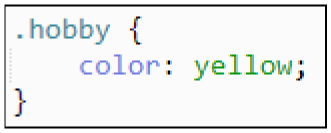
5.9: In order to style some element on a page, it needs to be contained within html tags. If you want to style just the hobby nouns on the list (video games, soccer, etc), we need to put tags around them. For this we will use the **<span>** tag with a class we can reference in the stylesheet.

about.html



```
<ul>
    <li><span class="hobby">Video games</span>:
    <li><span class="hobby">Soccer</span>: I am
    <li><span class="hobby">Music</span>: I love
    <li><span class="hobby">Programming</span>:
</ul>
```

styles.css



```
.hobby {
    color: yellow;
}
```

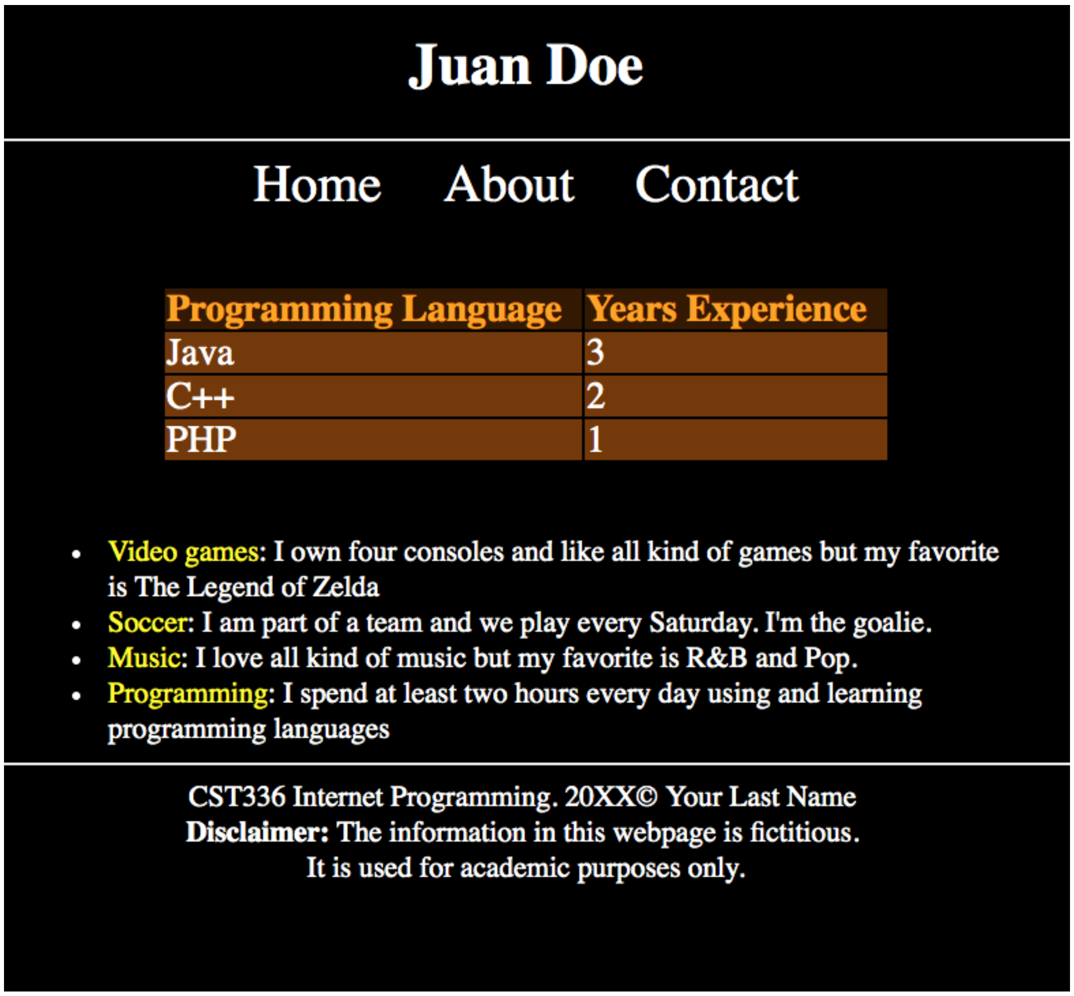
Notes:

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:



Juan Doe

Home About Contact

Programming Language	Years Experience
Java	3
C++	2
PHP	1

- Video games: I own four consoles and like all kind of games but my favorite is The Legend of Zelda
- Soccer: I am part of a team and we play every Saturday. I'm the goalie.
- Music: I love all kind of music but my favorite is R&B and Pop.
- Programming: I spend at least two hours every day using and learning programming languages

CST336 Internet Programming. 20XX© Your Last Name
Disclaimer: The information in this webpage is fictitious.
It is used for academic purposes only.

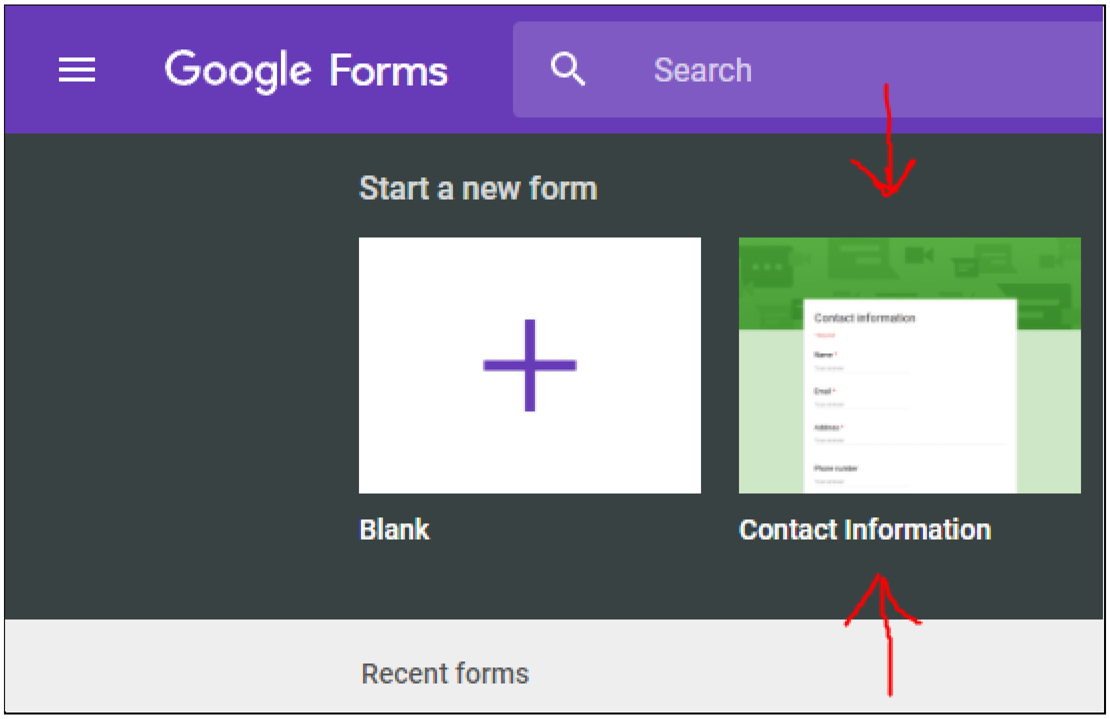
Lesson 6: Contact Page

Let's let friends contact us!

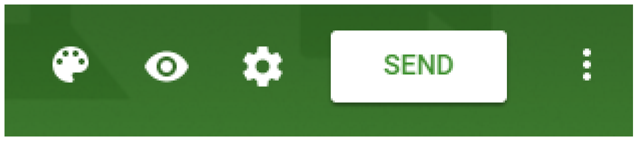
6.1: Next we can move onto the Contact page. Like the about page, we will copy over our index.html and remove everything inside **<div id="content">**.

6.2: For this page, we're going to use a **Google Form** to handle the contact feature. Later in the semester we will create our own forms, but that requires server programming. Besides, Google Forms are super slick and work great!

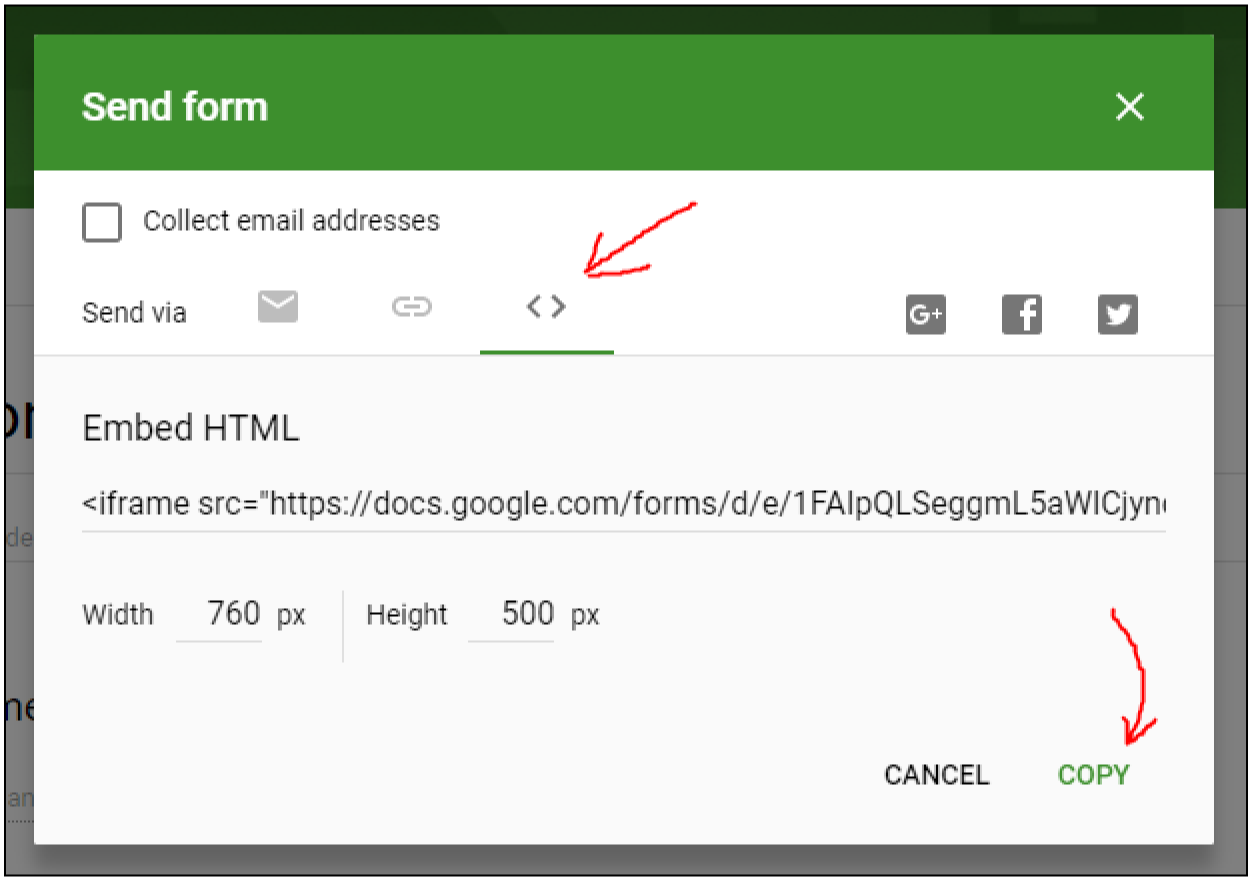
Go to https://docs.google.com/forms while logged into your Google account and select the **Contact Information** template from the gallery at the top.



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.


Send form

Collect email addresses

Send via

Embed HTML

```
<iframe src="https://docs.google.com/forms/d/e/1FAIpQLSeggmL5aWICjyn
```

Width 760 px Height 500 px

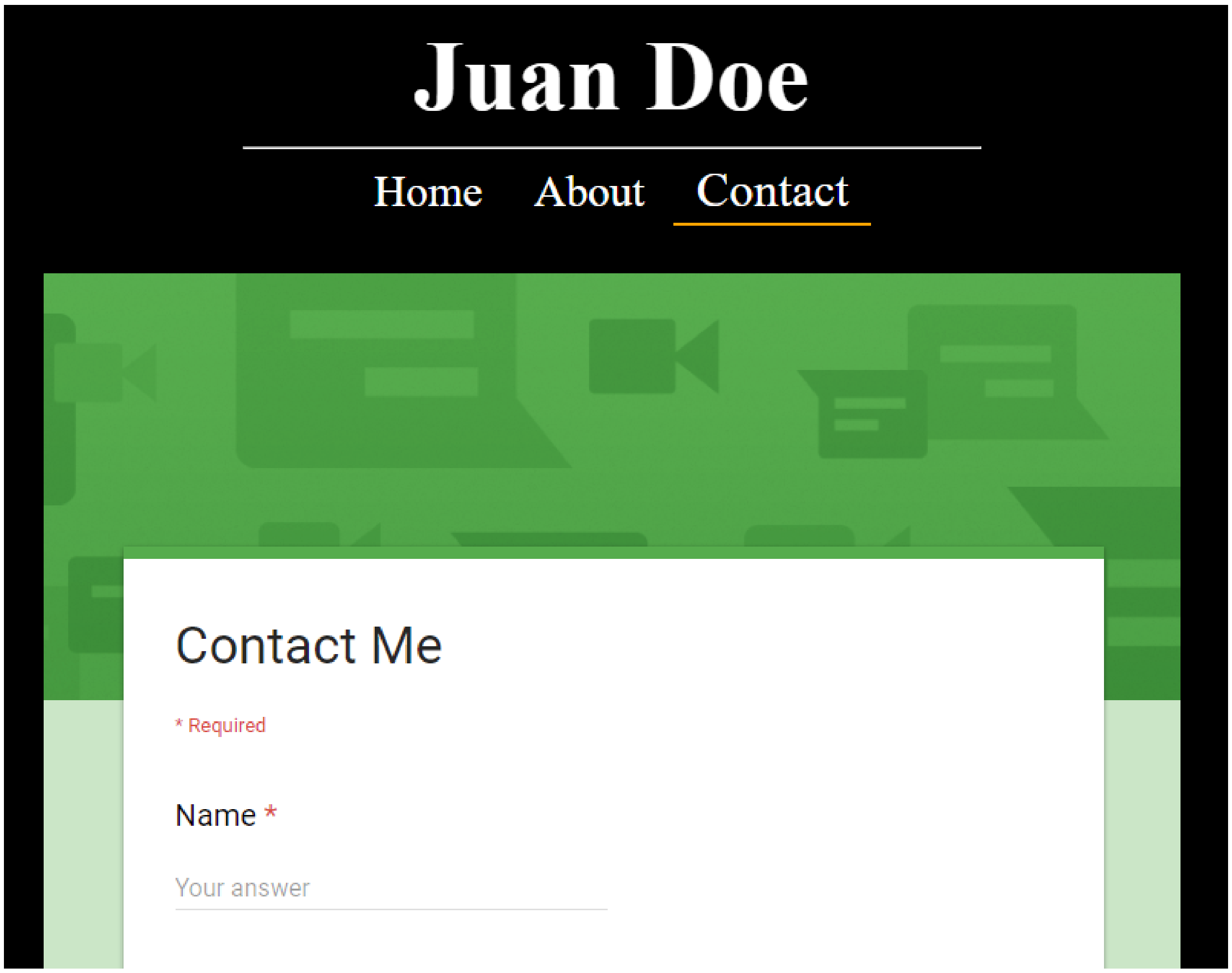
CANCEL COPY

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.

```
<!-- div with an id called content which we will re
<div id="content">
    <iframe src="https://docs.google.com/forms/d/e/1
</div>
```

The result:



Contact Me

* Required

Name *

Your answer

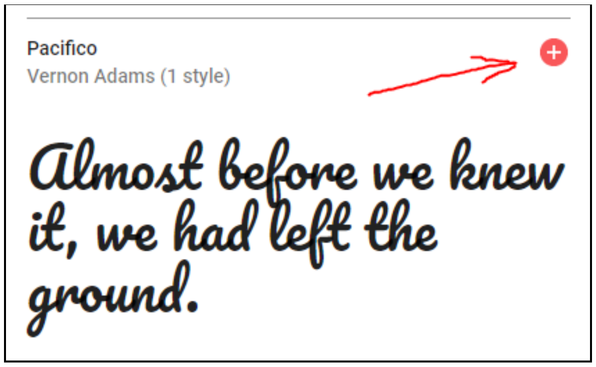
Lesson 7: Using Online Fonts

Let's get some stylish fonts!

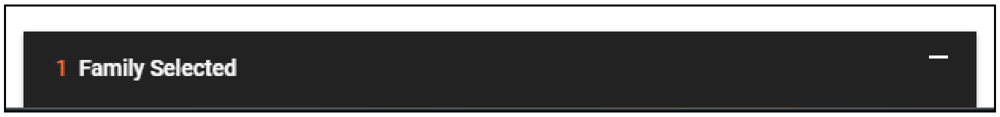
7.1: Our website is mostly complete, but the header could use something to help it stand out. One option would be to use an image, but consumes more bandwidth and doesn't allow for scaling. A better way is to use a different and nicer font than the rest of the web page.

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.

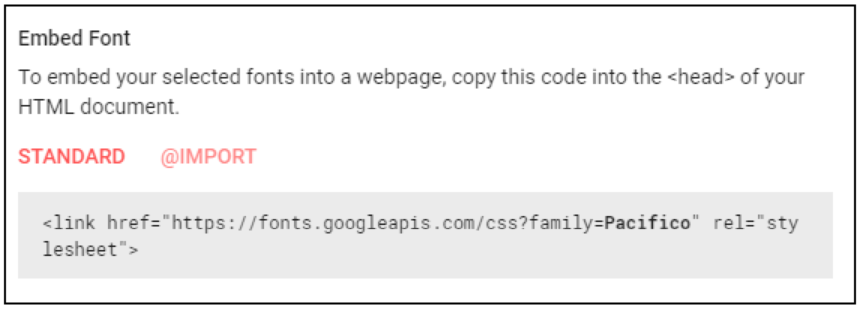
Go to fonts.google.com and search for Pacifico. Click the **+** to add the font family to your selection.



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**.



Embed Font

To embed your selected fonts into a webpage, copy this code into the <head> of your HTML document.

STANDARD @IMPORT

```
<link href="https://fonts.googleapis.com/css?family=Pacifico" rel="stylesheet">
```

```
    <link  href="css/styles.css" rel="stylesheet" type="text/css" />
    <link href="https://fonts.googleapis.com/css?family=Pacifico" rel="stylesheet">
</head>
```

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.

```
h1 {
    font-family: 'Pacifico', cursive;
    font-size: 4em;
    margin-bottom: 0px;
}
```