Intermediate HTML and CSS part 1

By Lea Ann Bradford for Girl Develop It San Diego

Welcome

Girl Develop It is here to provide affordable and accessible programs to learn software through mentorship and hands-on instruction.

Some “rules”

- We are here for you
- Every question is important
- Help each other
- Have fun

Introductions

- Lea Ann Bradford
- Software Developer @ Notch8
- Former student of GDI front end series
- Website: leaannbradford.com
- Twitter: @LeaABradford
- LinkedIn: linkedin.com/in/leaannbradford

Introductions

Tell us about yourself:

- Who are you?
- What do you hope to get out of this class?
- Do you collect or have a collection of anything?

Tools We Will Use Today

Web Browser:

- Chrome - preferred
- Safari
- Firefox

Text Editor

- Sublime Text - preferred
- Atom
- Notepad++

What Are We Going to Build?

We are going to create a personal portfolio web page

- We will do a quick review, but won't be covering the concepts from the first class
- We will show you some more advanced CSS concepts for you to try
- You can use the code snippets and assets we have assembled as helpers

Let's Review Web Languages

The three languages that make up the part of the web page that the client will see:

- HTML
- CSS
- JavaScript

What Makes Up a Web Page

- HTML = Structure
- CSS = Presentation
- Your content

Basic HTML Structure

There are 2 main parts to HTML:

- Doctype Declaration
- Document Tree

Doctype Declaration

- The declaration is not an HTML tag;
- it is an instruction to the web browser about what version of HTML the page is written in
- It is always the first thing on your page

- In HTML 5, we have a simplified doctype:

```
<! DOCTYPE html >
```

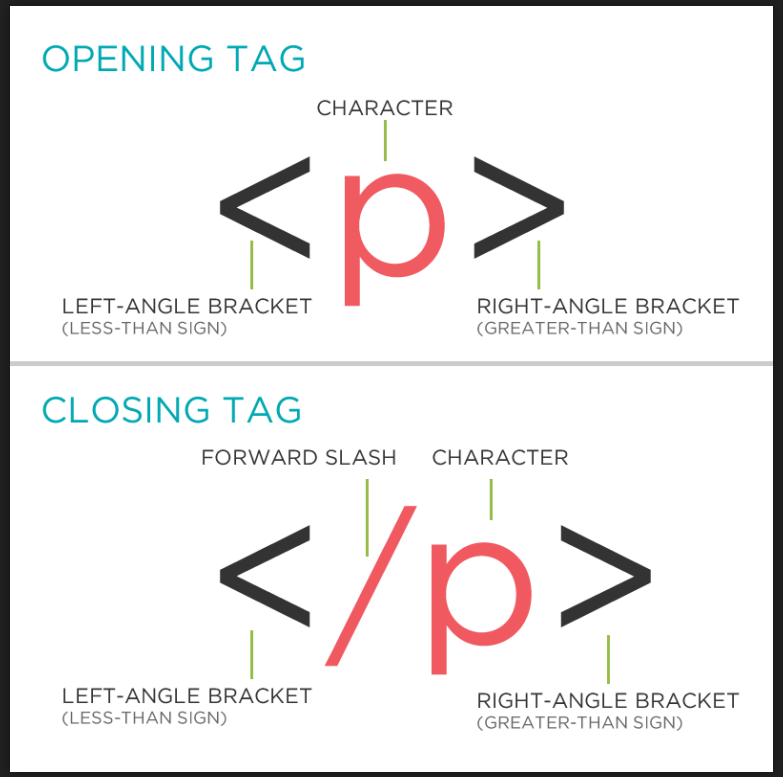
Document Tree

- A web page is considered a document tree because the markup is organized like the branches of a tree
- Notice the symmetry of the tree
- Tags have an opening and a closing
- Tags are nested and indented

```
<!doctype html>
<html>
    <head>
        <title>This is my web page</title>
    </head>
    <body>
        <h1>This is my web page</h1>
        <p>My page is a small page because
        it is just a sample</p>
    </body>
</html>
```

Anatomy of a Tag

- A tag consists of an opening tag and a closing tag
- Tags are enclosed in angle brackets
- Closing tags match opening tags and begin with a forward slash



Anatomy of an Element

- An element is an individual component of HTML
- Some examples are paragraph, heading, table, list, link, image
- An element consists of an opening tag, closing tag, and the content inside.

```
<tagname>Content inside</tagname>
```

```
<p>This is a paragraph</p>
```

Anatomy of an Element

- Most elements are container elements, which means the need a opening and closing tag
- Some elements are stand alone elements, which means they are self-closing and do not have content inside

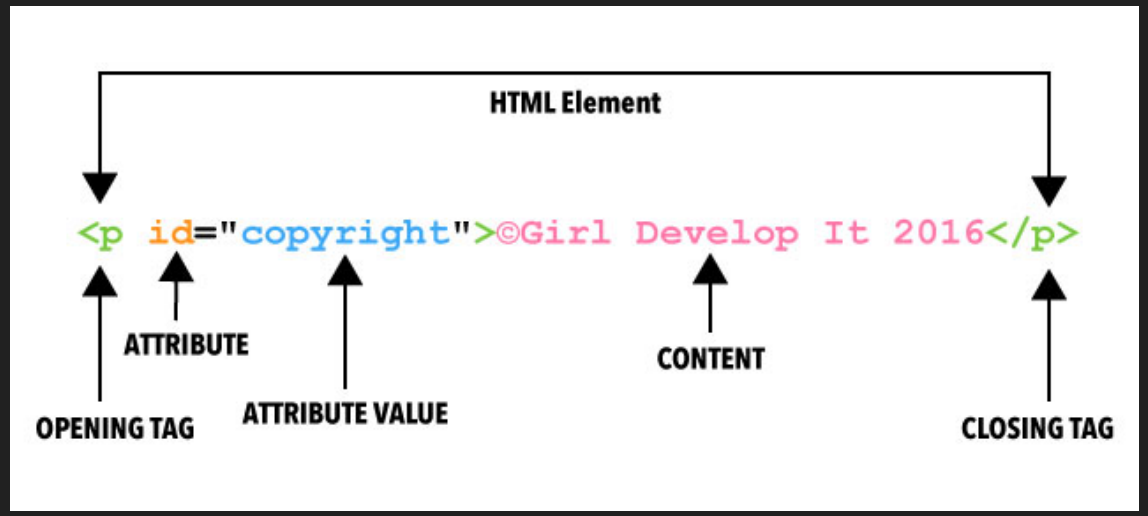
```
<p>A paragraph tag is a container element and has content inside</p>

<hr />

<img src="my-image.jpg" />
```

Attributes and Values

- html tags sometimes need additional information
- This information is provided in the form of attributes
- Attributes live inside the <angle brackets>
- Values are assigned to a given attribute
- Values are placed inside “quote marks”
- Some attributes are class, id, style, source



Nesting

- HTML elements "nest" inside one another
- Your web page content is inside your <body> tags, or "nested"
- Elements that open first will close last

```
<body>
    <p>This is my content. My paragraph
    is "nested inside the body tag</p>
    <p>The body tag opens first and
    closes last</p>
    <p>Nested elements are indented from
    their parent elements</p>
</body>
```

Semantic Markup – Section Tags

- Section tags make it easy to group items so you can format them with CSS
- You can apply IDs or Classes to these elements so you can control their styles with CSS
- This is a list of section tags you will find yourself using on almost every page you create

```
<body>
<header>
<nav>
<section>
<article>
<aside>
<address>
<footer>
```

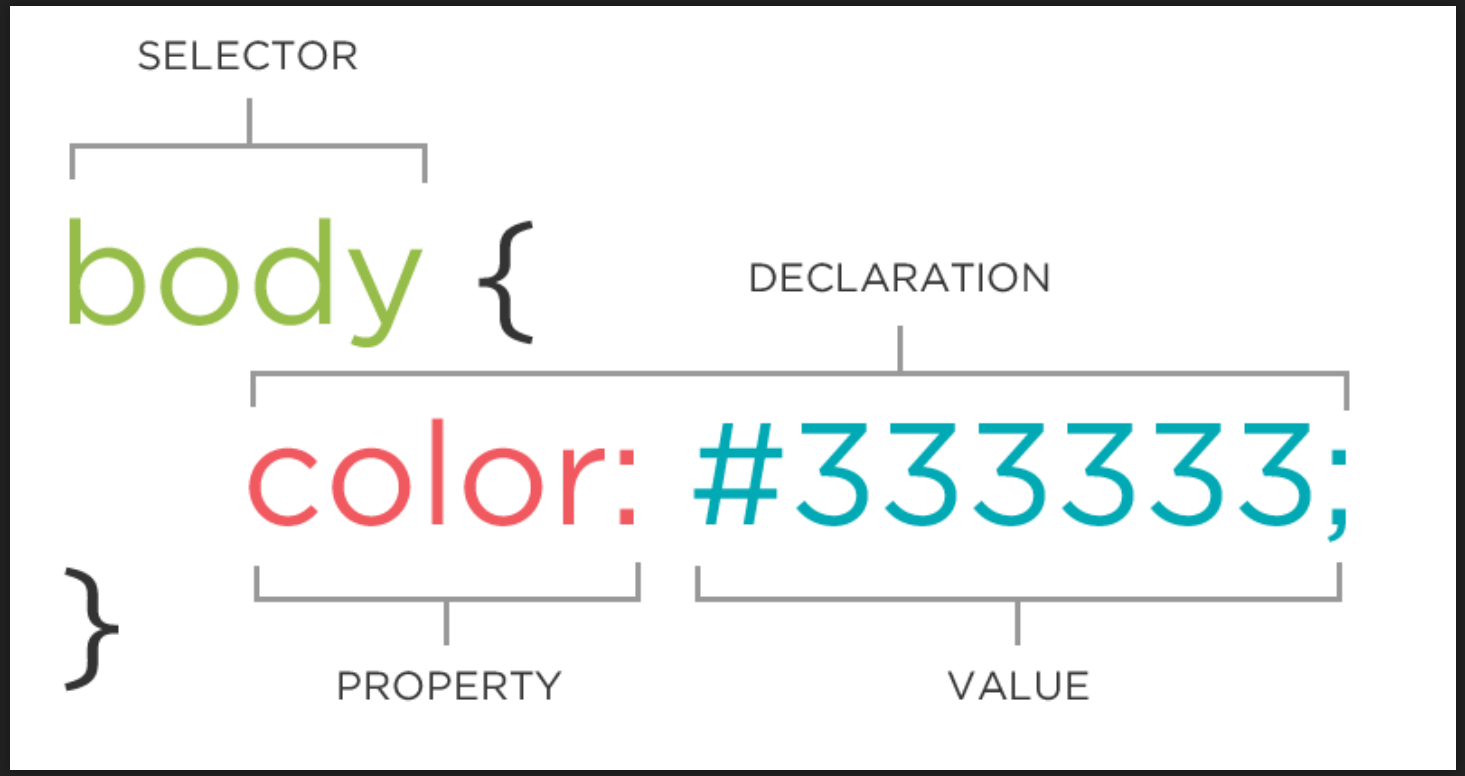
What Can CSS Do?

- Add colored text
- Position elements
- Size elements

How Does CSS Work

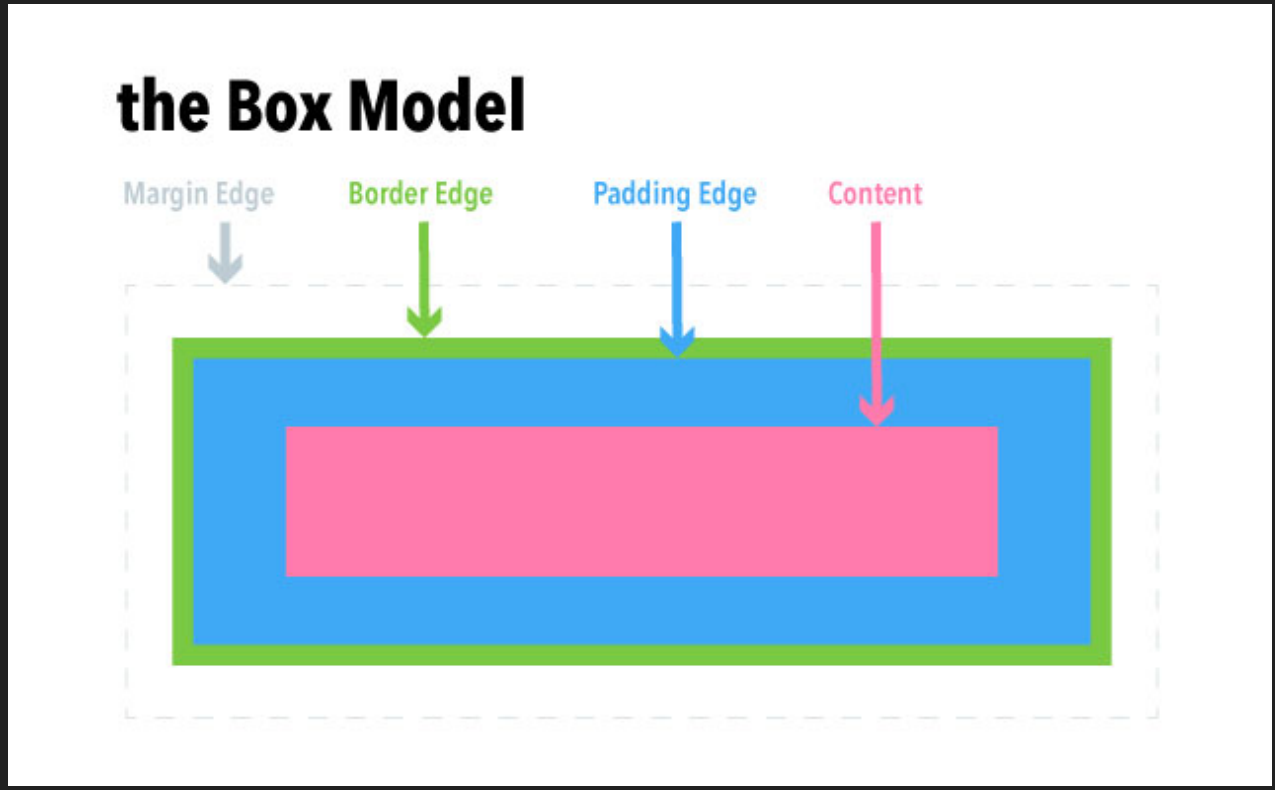
- CSS works by outlining style rules
- Style rules contain properties and property values that define how an element will look
- Style rules are applied to elements on your page

Anatomy of a CSS Rule



CSS Box Model

- All HTML elements can be considered as boxes.
- In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element.
- It consists of: margins, borders, padding, and the actual content.



Linking Your Style Sheet

- A link to your CSS file is placed in the <head> tag

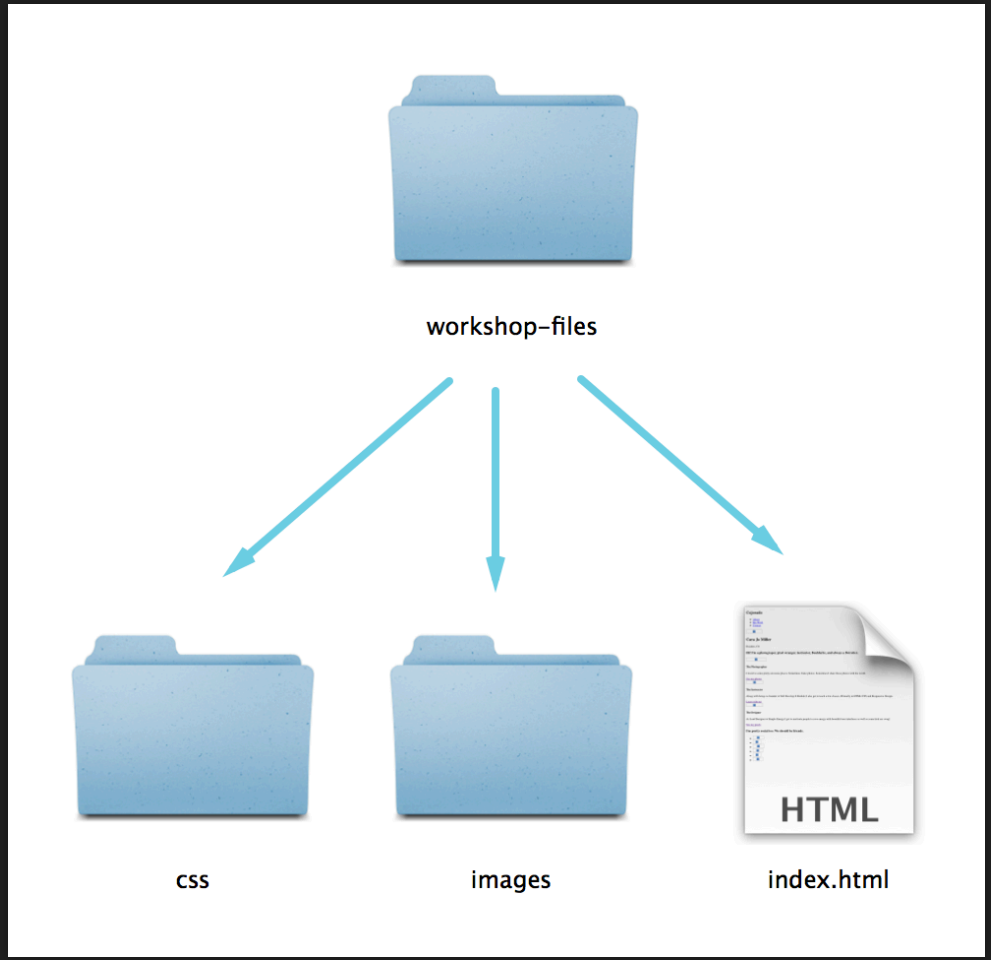
```
<head>
   <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

Starter Files

- Download at www.leaannbradford.com
- Includes starter HTML and CSS files
- We also included some image files and code snippets to help you

Folder Structure

- Our folder structure is set up in a commonly used format



Some Standard Practices – CSS Reset

- Even though HTML is the structure and CSS is the design, some HTML elements have default styles
- Different browsers display these things differently. A reset gets rid of these inconsistencies
- We can remove these default styles by adding a CSS reset

- **Examples include:**
  - Bulleted lists have standard bullets
  - Paragraphs & headings have default padding
  - Links are blue and underlined

Some Standard Practices – CSS Reset

Most Elements

```
margin: 0;
padding: 0;
border: 0;
font-size: 100%;
font: inherit;
vertical-align: baseline;
```

Lists

```
list-style: none;
```

Some Standard Practices – CSS Reset

- We have included a CSS reset file for you so you won't need to type this out.

Some Standard Practices – Widths & Sizes

- Screen sizes vary from computer to computer.
- Standardize your site on different screen sizes by defining specific widths.
- Wrap your content in containers to control the max width it can span across a screen.
- Keep in mind screen sizes also mean different font size display.
- Retina screens have a higher pixel density and a larger resolution, so fonts appear smaller.

Some Standard Practices – Fixed Width

- Fixed websites have a set width for the wrapper, usually 960px to 1024px.
- The elements inside the wrapper have a set width, or percentage of the wrapper width.
- No matter the screensize, the user always sees the same size site.

Some Standard Practices – Fluid

- Also referred to as a liquid layout.
- Majority of the components, including the wrapper, have percentages for their widths.
- The layout adjusts for the width of the user's screen resolution.
- Sounds like a responsive site right!? More on that later.

Some Standard Practices – Containers

- Wrappers are a good way to center content if the screen width is wider than your content.
- The container will take up 100% of the screen if the width of the viewport is less than 1440px.
- If the viewport is wider than 1440px, it will reach it's max width, and become centered in the viewport.

```
.container {
    width: 100%;
    max-width: 1440px;
    margin: 0 auto;
}
```

Let's Develop It

- Let's start your portfolio page
- Change the <title> of your page in the <head> of your HTML file
- Add semantic html section elements: nav, header, section, and footer
- Make sure your style sheet is working by adding a CSS style rule

Fixed Navigation

- In the first part of this series, we create a simple crude navigation
- We are going to show you two popular types of navigation styles
  - Top fixed
  - Side fixed

- Fixed navigation allow users to have access to navigational elements at all times
- Be careful - screen sizes vary, and it reduces the amount of content visible on smaller screens

Navigation – HTML

- Let's include it in a header, since we know we'll be grouping related content here.

```
<header>
    <nav>
        <ul>
            <li><a href="#home">About Me</a></li>
            <li><a href="#about">My Skills</a></li>
            <li><a href="#skills">Contact Me</a></li>
        </ul>
    </nav>
</header>
```

Top Navigation – CSS

- Remember, using fixed position is like using absolute position, except that it's fixed to the viewport, not the containing element.
- We also have to define a width for it, and its location.

```
nav {
    position: fixed;
    top: 0;
    left: 0;
    background: #fafafa;
    border-bottom: 2px solid #ccc;
    height: 70px;
    width: 100%;
}
```

Top Navigation – CSS

- Because we've fixed the nav to the top of the viewport, we need to bump the content of the body down to be visible to the user.
- This should be the same, or more than, the height of the navigation bar.

```
body {
    padding-top: 70px;
}
nav {
    position: fixed;
    top: 0;
    left: 0;
    background: #fafafa;
    border-bottom: 2px solid #ccc;
    height: 70px;
    width: 100%;
}
```

Top Navigation – CSS

- Now we need to get those list items next to each other instead of stacked.
- Let's float them to the left and add some padding to the links so they have a large clickable area.

```
nav li {
    float: left;
    width: auto;
}
nav li a {
    padding: 25px 10px;
    display: block;
}
```

Side Navigation – CSS

- Remember, using fixed position is like using absolute position, except that it's fixed to the viewport, not the containing element.
- We also have to define a width for it, and its location.

```
nav {
    position: fixed;
    top: 0;
    left: 0;
    background: #fafafa;
    border-right: 2px solid #ccc;
    height: 100%;
    width: 20%;
}
```

Side Navigation – CSS

- Because we've fixed the nav to the top of the viewport, we need to bump the content of the body down to be visible to the user.
- This should be the same, or more than, the height of the navigation bar.

```
body {
    padding-left: 25%;
}
nav {
    position: fixed;
    top: 0;
    left: 0;
    background: #fafafa;
    border-right: 2px solid #ccc;
    height: 100%;
    width: 20%;
}
```

Side Navigation – CSS

- For side nav, we don't need to get those list items next to each other, we want them to remain stacked.
- But we do want to add some padding to the links so they have a large clickable area.

```
nav li {
    width: auto;
}
nav li a {
    padding: 25px 10px;
    display: block;
}
```

Fixed Navigation – Container

- Notice how the edge of the nav bumps up against the edge of the browser? Let's fix that by adding a container around it.

```
<header>
    <div class="container">
        <nav>
            <ul>
                <li><a href="#home">About Me</a></li>
                <li><a href="#about">My Skills</a></li>
                <li><a href="#skills">Contact Me</a></li>
            </ul>
        </div>
    </nav>
</header>
```

Fixed Navigation – Container

- Let's give the container a fixed width and see what happens.
- Now wherever we use .container it will be 1024px wide and centered

```
.container {
    width: 100%
    max-width: 1024px;
    margin: 0 auto;
}
```

Let's Develop It

- Let's work on our navigation
- Remove the underlines on the links with text-decoration
- Change the color of the links
- Try adding a background color on hover

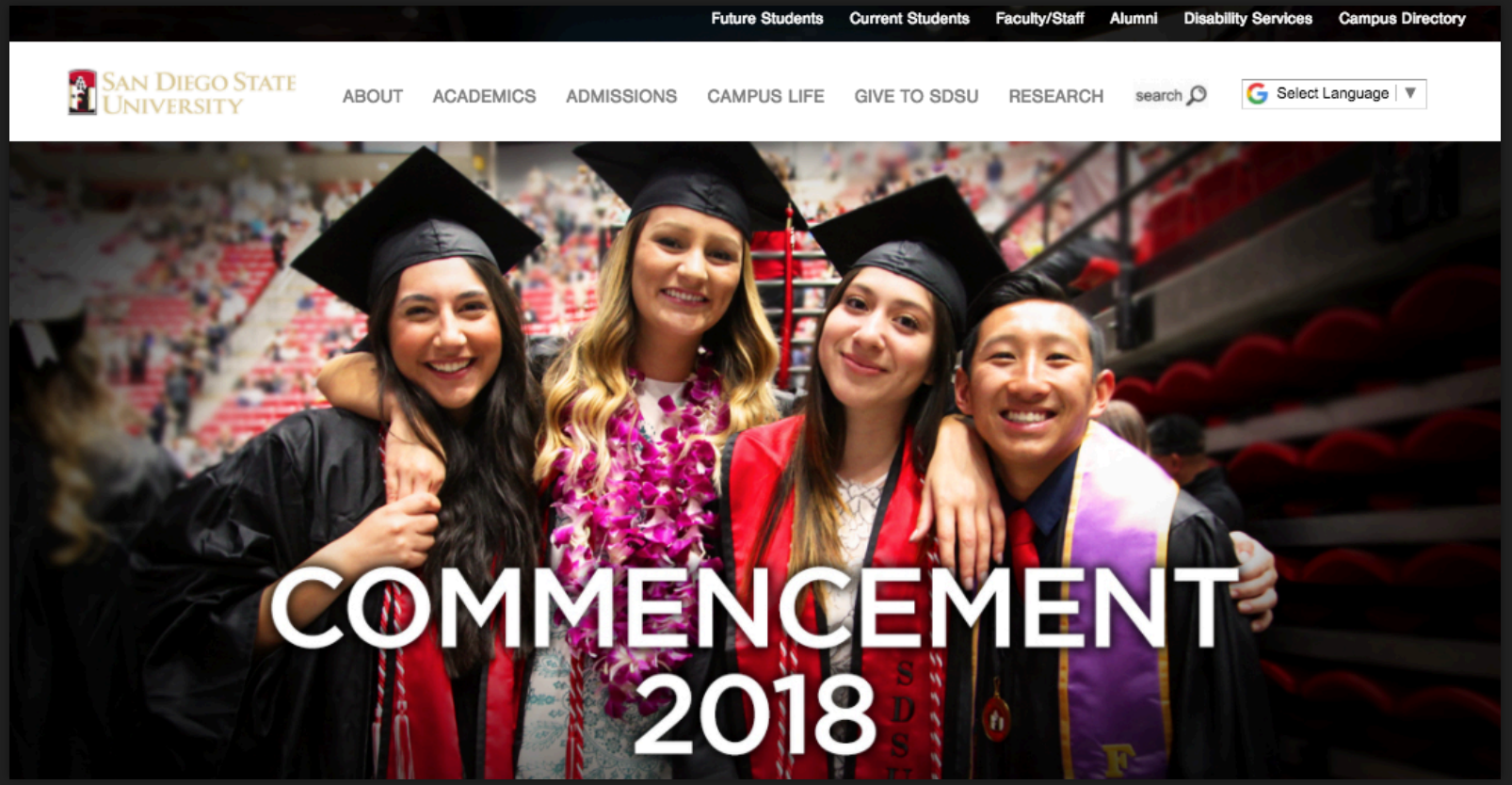
Hero Section

- Large banner image, prominently placed on a web page, generally in the front and center
- First visual a visitor encounters on the site and its purpose is to present an overview of the site's most important content
- Often consists of image and text, can be static or dynamic

Hero Examples

Hero Examples

Hero Examples



Hero Section – HTML

- Let's get started on our hero section. It should look something like this.

```
<section id="hero">
    <img src="images/profile.jpg" alt=”Lea Ann" />
    <h2>Lea Ann Bradford</h2>
    <span class=”tagline">Front End Developer</span>
</section>
```

Hero Section – CSS

- Let's use CSS to add a background image

```
#hero {
    background: url(../images/hero.jpg) no-repeat top left;
    color: #fafafa;
    text-align: center;
}
```

Hero Section – CSS

- Let's add some height and padding as well
- Remember the box model? Our height now is equal to the height + padding

```
#hero {
    background: url(../images/hero.jpg) no-repeat top left;
    color: #fafafa;
    text-align: center;
    height: 350px;
    padding: 25px 0;
}
```

Hero Section – CSS

- Let's add one more property to our background
- cover scales the image to the largest size such that both its width and its height can fit inside the content area. round image

```
#hero {
    background: url(../images/hero.jpg) no-repeat top left;
    color: #fafafa;
    text-align: center;
    height: 350px;
    padding: 25px 0;
    background-size: cover;
}
```

Hero Section – CSS

- We need to fix our headshot and headline now – let's use a descendent selector

```
#hero img {
    width: 150px;
}
```

Hero Section – CSS

- Let's give our image a border radius
- We can add rounded corners on all sides, some sides or even make a circle

```
#hero img {
    border-radius: 20px;
}
#hero img {
    border-radius: 10px 40px;
}
#hero img {
    border-radius: 10px 20px 40px 80px;
}
#hero img {
    border-radius: 50%;
}
```

Let's Develop It

- Let's work on our hero section
- Adjust the font size of the header
- Add a border to the span