Beginning HTML and CSS

Class 2

Website Anatomy Review

Anatomy of a website

Your Content
+ HTML: Structure
+ CSS: Presentation
=Your Website

A website is a way to present your content to the world, using HTML and CSS to present that content & make it look good.

Anatomy of a website

Concrete example

- A paragraph of text is your **content**
- Putting your content into an HTML tag to mark it up as a paragraph is **structure**

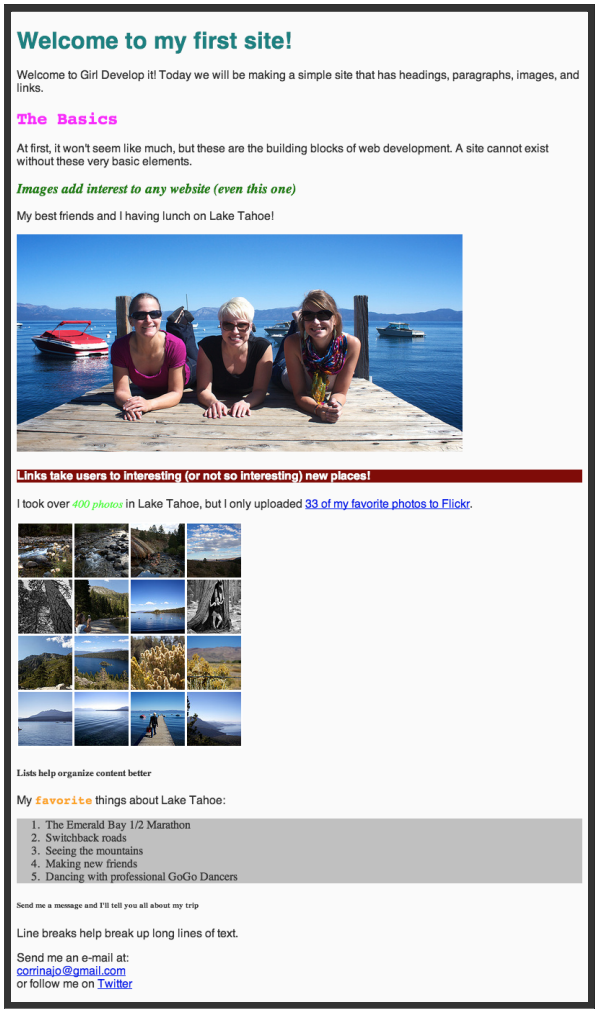
```
<p>A paragraph of text is your content</p>
```

- Making your paragraph text green and 48px is **presentation**

A paragraph of text is your content

What We'll Build Today

Today we'll learn how to add style to our HTML page from last class.



Welcome to my first site!

Welcome to Girl Develop it! Today we will be making a simple site that has headings, paragraphs, images, and links.

The Basics

At first, it won't seem like much, but these are the building blocks of web development. A site cannot exist without these very basic elements.

Images add interest to any website (even this one)

My best friends and I having lunch on Lake Tahoe!

Links take users to interesting (or not so interesting) new places!

I took over *400 photos* in Lake Tahoe, but I only uploaded 33 of my favorite photos to Flickr.

Lists help organize content better

My favorite things about Lake Tahoe:

1. The Emerald Bay 1/2 Marathon
2. Switchback roads
3. Seeing the mountains
4. Making new friends
5. Dancing with professional GoGo Dancers

Send me a message and I'll tell you all about my trip

Line breaks help break up long lines of text.

Send me an e-mail at:
corrinajo@gmail.com
or follow me on Twitter

Intro to CSS

CSS: What can it do?

Colors, positioning, sizing, and so much more.

CSS: What does it look like?

```
}
body {
    font: 100% "Gotham", sans-serif;
    width: 100%;
    padding: 0;
    margin: 0;
}
header{
    clear:both;
    width: 100%;
}
h3{
    font-size: 110%;
    font-weight: normal;
    letter-spacing: 1px;
    margin-bottom: 12px;
    margin-top: 10px;
    text-transform: uppercase;
}
a{
    color: #01A9B4;
    text-decoration: none;
    background-color: transparent;
}
a:hover
{
    color: #222;
}
#page, #site_bar_content, .header, .footer {
  max-width: 1200px;
  min-width: 480px;
  margin: 0px auto;
}
.header{
    padding: 10px 15px;
}
```

CSS: What is it?

CSS stands for Cascading Style Sheets

CSS is a "style sheet language" that lets you style the elements on your page.

CSS works in conjunction with HTML, but is not HTML itself.

CSS: History

- First Working Draft: December 1996
- Wasn't fully usable in *any* browser until Internet Explorer 5 for Mac in March 2000
- CSS 2.1 bounced between Draft status and Recommendation status between 2004-2010
- CSS 3 work is ongoing, but most modern browsers have good support

CSS Rule Basics

The CSS Rule

The CSS Rule: Terms

```
selector {
  property: value;
}
```

A block of CSS code is a rule.

The rule starts with a selector.

It has sets of properties and values.

A property-value pair is a declaration.

CSS Syntax

Properties and values are separated by a colon, creating a declaration.

Declarations end with a semicolon.

Declaration groups are surrounded by curly brackets, creating a rule.

```
selector {
  property: value;
  property: value;
  property: value;
}
```

A Basic Selector: Element

```
p {
  property: value;
}
```

Selects all paragraph elements.

```
img {
  property: value;
}
```

Selects all image elements.

A Basic Property: Font-size

The **`font-size`** property specifies the size of the text.

```
p {
  font-size: 12px;
  font-size: 1.5em;
  font-size: 100%;
}
```

Font-sizes can have different units, including pixels (px), ems (em), and percentage (%)

But wait, where does it all go?

Connecting CSS to HTML

3 ways

"Inline"

"Embedded"

"External"

Connecting CSS to HTML: Inline

```
<p style="color:red">Some text.</p>
```

- Uses the HTML attribute `style`
- Only applies to that element
- Difficult to manage in large projects
- **Highly discouraged**

Connecting CSS to HTML: Embedded

```
<head>
  <style type="text/css">
    p {
      color: blue;
      font-size: 12px;
    }
  </style>
</head>
```

- Inside the **head** element
- Uses **<style>** tag
- Only applies to that HTML file

Connecting CSS to HTML: Linked

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

- Inside the **`head`** element
- Uses **`<link>`** tag
- Shared resource for several pages
- Reduced file size & bandwidth
- Easy to maintain in larger projects

Preferred by developers everywhere!

More Selectors

Selector: ID

```
#footer {
  property: value;
}
```

Selects all elements with an id of "footer".

```
<p id="footer">Copyright 2011</p>
```

The associated HTML.

Selector: Class

```
.warning {
  color: red;
}
```

Selects all elements with a class of "warning".

```
<p class="warning">Run away!</p>
```

The associated HTML.

IDs vs. Classes

ID: only one per page.

The "#" is how you tell CSS "this is an id."

Example: A webpage only has one page footer.

vs.

Class: many similar elements.

The "." is how you tell CSS "this is a class name."

Example: Many paragraphs can be warnings.

Selector: Descendant

```
p em {
  color: yellow;
}
```

Selects all em elements that are within a paragraph

```
<p>This is <em>important.</em></p>
```

The associated HTML.

Selector: Chaining

You can even mix selector types

```
p em {
  color: yellow;
}
p.warning em {
  color: maroon;
  font-size: larger;
}
```

```
<p>This is <em>important.</em></p>
<p class="warning">This is a warning of <em>importance.</em></p>
<div>This is not very <em>important.</em></div>
```

This is *important.*

This is a warning of *importance.*

This is not very *important.*

More Properties

Property: Color

The **`color`** property changes the color of the text.

```
p {
  color: red;
  color: #ff0000;
  color: rgb(255, 0, 0);
}
```

There are multiple color value types. Shown above: color name, hexadecimal value, and RGB value.

The 17 standard named colors are: aqua, black, blue, fuchsia, gray, grey, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow.

A really cool color tool: Color Hexa

Property: Background-color

The **`background-color`** property changes the color of the element background.

```
p {
  background-color: black;
  background-color: #000000;
  background-color: rgb(0,0,0);
}
em {
  background-color: lime;
}
```

Don't forget to change the *text color* or you won't be able to read your content.

Property: Font-family

The **`font-family`** property defines which font is used.

```
p {
  font-family: "Times New Roman";
  font-family: serif;
  font-family: "Arial", sans-serif;
}
```

Fonts can be specified by a specific font name, generic stule name, or a comma-separated list of both (called a "font stack").

Property: Font

```
p {
  font-style: italic;
  font-weight: bold;
  font-size: 28px;
  font-family: serif;
}
```

can be simplified to

```
p {
  font: italic bold 28px serif;
}
```

Shorthand properties like **`font`** are awesome because

developers hate typing.

Property Values

Different properties require different value types.

Some always take just one value:

```
p { color: white; }
```

Some allow a comma-separated list of values:

```
p { font-family: Arial, "Helvetica Neue", sans-serif; }
```

Shorthand properties take a space-separated list:

```
p { border: 1px solid fuchia; }
```

If you're having trouble remembering, references are always handy.

CSS Properties

Many CSS properties have self-explanatory names:

- background-color
- font-family
- font-size
- color
- width
- height

Comprehensive list of all CSS properties

The Cascade

Cascading

Styles "cascade" down until changed

```
p{
  color: blue;
  font-family: 'Times New Roman';
}
.red{
  color: red;
}
#special{
  font-family: Arial;
}

<p>Paragraph</p>
<p class="red">Paragraph</p>
<p class="red" id="special">Paragraph</p>
```

Paragraph

Paragraph

Paragraph

Inheritance

Properties like colors and fonts are "inherited" from their parents. This is why you don't have to set them for every element on the page.

```
ul{
  color: red;
}
.special {
  color: blue;
}

<ul>
  <li>This is content that is <em>important</em></li>
  <li class="special">This is content that is also <em>important</em></li>
</ul>
```

- This is content that is *important*
- This is content that is also *important*

Specificity

Selectors have "importance" that can interrupt the cascade.

```
#special{ font-family: Arial; }
.red{ color: red; }
p{
  color: blue;
  font-family: 'Times New Roman';
}

<p>Paragraph</p>
<p class="red">Paragraph</p>
<p class="red" id="special">Paragraph</p>
```

Paragraph

Paragraph

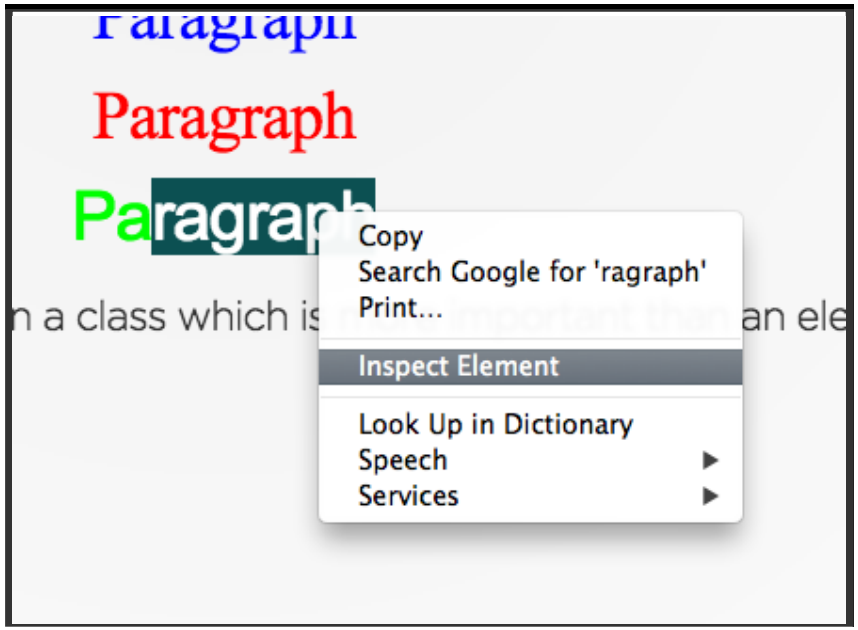
Paragraph

An id is more important than a class which is more important than an element.

Troubleshooting

Even experienced developers run into issues with the cascade and specificity.

This is where Developer Tools are useful.




```
<div class="class2-specificity-example">
  <p>Paragraph</p>
  <p class="red">Paragraph</p>
  <p class="red" id="special" style="color:lime;">Paragraph</p>
</div>
```

```
element.style {
  color: #0F0;
}
.class2-specificity-example #special  class2.html:607
{
  font-family: Arial, sans-serif;
}
.class2-specificity-example .red {    class2.html:610
  color: #F00;
}
.class2-specificity-example p {       class2.html:613
  color: #00F;
  font-family: 'Times New Roman', serif;
}
```

This is why we don't like inline styles. They have a lot of importance, which complicates things.

Questions?