Coding Temple

GETTING TO KNOW CSS

What is CSS?

- CSS is a complex language that packs quite a bit of power
- Allows us to add layout and design to our pages, and it allows us to create rules that specify how the content of an element show appear
- For example, you can specify that the background of the page is cream, all paragraphs should appear in gray using the Arial typeface
- It's very crucial to understand exactly how styles are rendered
- The key to understanding how CSS works is to imagine that there is an invisible box around every html element

CSS Selectors

- Universal Selector
- Type Selector
- Class Selector
- ID Selector
- Child Selector
- Descendant Selector
- Adjacent Sibling Selector
- General Sibling Selector
- Attribute Selector

Universal & Type Selector

- Universal Selector
  - Applies to all elements in the document (html page)

```
* {
    color: red;
}
```

- Type Selector
  - Matches Element Names
  - Target all ***<h1>***, ***<h2>***, ***<h3>*** elements

```
h1, h2, h3 {
    color: orangered;
}
```

ID & Class Selector

- ID Selector
  - Matches an element whose *id* attribute has a value that matches the one specified after the pound or hash symbol
- Class Selector
  - Matches an element whose *class* attribute has a value that matches one specified after the period symbol
  - Examples:
    - Target any element with *class* attribute value of *note*
    - Targets only **<p>** elements whose *class* attribute has a value of *note*

```
#introduction {
    color: hotpink;
}
```

```
.note {
    font-size: 18pt;
}
```

```
p.note {
    font-size: 15pt;
}
```

Child & Descendant Selector

- Child Selector
  - Matches an element that is a direct child of another
  - Targets any **<a>** element that are children of **<li>** element, but not other **<a>** elements in the page

```
li > a {
    color: dodgerblue;
}
```

- Descendant Selector
  - Matches an element that is a descendant of another specified element (not just a direct child of that element)
  - Targets any **<a>** elements that sits inside a **<p>** element, even if there are other elements nested between them

```
p a {
    color: magenta;
}
```

Adjacent & General Sibling Selector

- Adjacent Sibling Selector
  - Matches an element that is the next sibling of another
  - Targets the first **<p>** element after any **<h1>** element (but not other **<p>** elements)

```
h1 + p {
    color: red;
}
```

- General Sibling Selector
  - Matches an element that is a sibling of another, although it does not have to be the directly preceding element
  - If you had two **<p>** elements that are siblings of an **<h1>** element, this rule would apply to both

```
h1 ~ p {
    color: red;
}
```

Attribute Selector

SELECTOR	MEANING	EXAMPLE
EXISTENCE	`[]` Matches a specific attribute (whatever its value)	`p[class]` Targets any <p> element with an attribute called `class`
EQUALITY	`[=]` Matches a specific attribute with a specific value	`p[class="dog"]` Targets any <p> element with an attribute called `class` whose value is `dog`
SPACE	`[~=]` Matches a specific attribute whose value appears in a space-separated list of words	`p[class~="dog"]` Targets any <p> element with an attribute called `class` whose value is a list of space-separated words, one of which is `dog`
PREFIX	`[^=]` Matches a specific attribute whose value begins with a specific string	`p[attr^"d"]` Targets any <p> element with an attribute whose value begins with the letter "d"
SUBSTRING	`[*=]` Matches a specific attribute whose value contains a specific substring	`p[attr*"do"]` Targets any <p> element with an attribute whose value contains the letters "do"
SUFFIX	`[$=]` Matches a specific attribute whose value ends with a specific string	`p[attr$"g"]` Targets any <p> element with an attribute whose value ends with the letter "g"

How CSS Rules Cascade

- Last Rule
  - If the two selectors are identical, the latter of the two will take precedence.
- Specificity
  - If one selector is more specific than the others the more specific rule will take precedence over more general ones
  - Examples:
    - *h1* is more specific than *
    - p b is more specific than *p*
    - *p#intro* is more specific than *p*
- Important
  - You can add *!important* after any property value to indicate that it should be considered more important than other rules that apply to the same element

```
i {
    color: green;
}

i {
    color: red;
}
```

```
* {
    font-family: Arial, Verdana, sans-serif;
}

h1 {
    font-family: 'Courier New', monospace;
}
```

```
p#intro {
    font-size: 100%;
}

p {
    font-size: 75%;
}
```

```
p b {
    color: blue !important;
}

p b {
    color: violet;
}
```

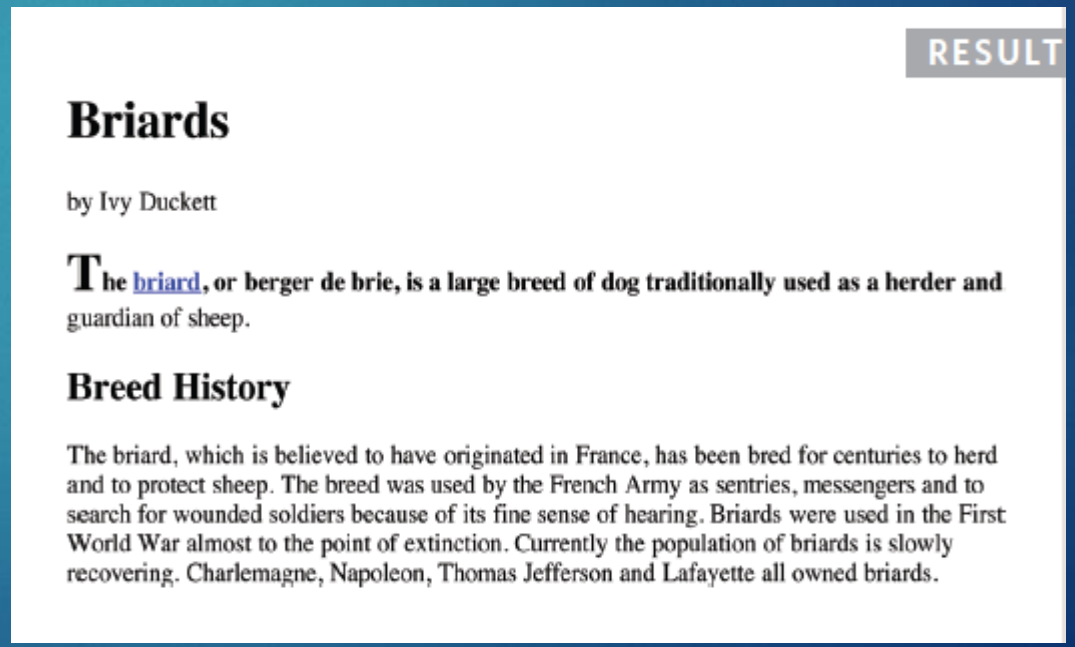
Common Length Values

- Absolute Lengths
  - Fixed physical measurements, such as inches, centimeters, or millimeters
  - Pixels
    - Is equal to 1/96th of an inch, making it 96 pixels per inch or 96ppi
    - This may slightly be different depending on high or low density viewing devices
- Relative Lengths
  - Non fixed units of measurement that rely on the length of another measurement (dynamic lengths)
  - Percentages
    - Defined in relation to the length of another object
    - If parent object has a width of **100px** and the object has a width of **50%,** than the object will have a width of **50px**
  - Em
    - Length is calculated based on the an element's font size
    - If the element has a **font-size** of **14px,** than 1 em will be equal to **14px**

What are pseudo elements?

- A **pseudo element** acts like an extra element in the code.
  - Example: If you only wanted to apply styling to the first letter of a paragraph, you can use **:first-letter** to select that element.
  - Example: To apply styling to the first line of a paragraph only, select it using **:first-line**.

```
p.intro:first-letter {
  font-size: 200%;}
p.intro:first-line {
  font-weight: bold;}
```



What are pseudo classes?

- A **pseudo class** acts like an extra value for a class attribute.
- Examples:
  - The **:visited** pseudo class allows you to select and style links that have been visited.
  - The **:hover** pseudo class allows you to style elements differently as a user hovers over them.
  - the **:link** pseudo class selects links that haven't been visited.

```
a:link {
  color: deeppink;
  text-decoration: none;}
a:visited {
  color: black;}
a:hover {
  color: deeppink;
  text-decoration: underline;}
a:active {
  color: darkcyan;}
```

Understanding Colors

RGB VALUES

Values for red, green, and blue are expressed as numbers between 0 and 255.

`rgb(102,205,170)`

This color is made up of the following values:
102 red
205 green
170 blue

HEX CODES

Hex values represent values for red, green, and blue in hexadecimal code.

`#66cdaa`

The value of the red, 102, is expressed as 66 in hexadecimal code. The 205 of the green is expressed as `cd` and the 170 of the blue equates to `aa`.

COLOR NAMES

Colors are represented by predefined names. However, they are very limited in number.

`MediumAquaMarine`

There are 147 color names supported by browsers (this color is `MediumAquaMarine`). Most consider this to be a limited color palette, and it is hard to remember the name for each of the colors so (apart from white and black) they are not commonly used.