CSS DIVS, IDs, classes, and Positioning

A New Element - DIV

DIVs are empty containers

A DIV does not add or remove anything from a page

DIVs are a great to help style a page

DIVS make it significantly easier to style components

```
<div></div>
```

Lets make a Div!

```
<div style="background-color: lightblue;">

        Hello

</div>
```

IDs

The ID attribute is used to create a unique id for an element

As good practice an ID should be applied to only one element on a page

It is written as follow:

```
<div id="paragaphOne">This paragraph has a unique ID attribute</div>
```

Using an ID

In CSS to define an ID you use **#**

```
#paragraphOne {
  background-color: black;
  color: white;
  text-align: left;
}
```

Classes

The class attribute is used to create a shared “ID” across multiple elements

As good practice a class should be applied to more than one element on a page

It is written as follow:

```
<div class="paragaphClass">This paragraph has a unique class attribute</div>
```

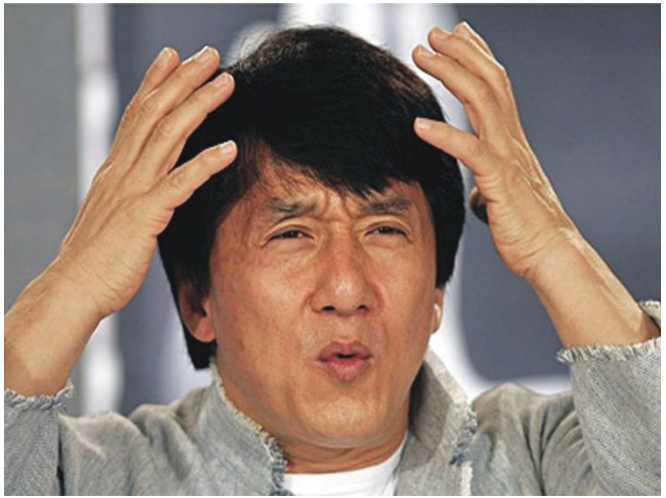
Using a class

In CSS to define a class you use .

```
.paragraphClass {
  background-color: blue;
  color: black;
  text-align: right;
}
```

What's the difference?!!?!?

- IDs are typically used when you want to write code for a specific element (IDs will become VERY important when we get into Javascript)
- Classes are used for when you want to apply code to multiple elements



Lets try it out!

Create a new web page with two elements. Apply an id to one element and a class to the other element

```
HTML - <div id="paragaphOne">This paragraph has a unique ID attribute</div>

CSS - #paragraphOne {

            background-color: black;

      }
```

```
HTML - <div class="paragaphClass">This paragraph has a unique class attribute</div>

CSS - .paragraphClass {

            background-color: black;

      }
```

CSS sizing

Elements and content can be sized differently based on the following values

- px - Defines the font size in pixels
- % - Defines the size based on the current browser windows size
- em - Relative to the font size of the element. For example 3em would make the font 3 times larger than the font around it

Font size is written as follows

```
font-size: 10px;
```

Lets try out sizing

Use the following to the 2 elements to see how the sizing attributes work

- font-size: 10px
- font-size: 2em
- font-size: 50%

Setting the Width and Height of an element

Width and Height can be used to set the vertical and horizontal size of an element

The percentage, px, em, etc measurements can be used when defining width and height

It is written as follows:

```
height: 60px;

width: 80px;
```

Min and Max Width/Height

A Minimum and Maximum Width and Height can be applied to an element

This helps you better control elements when dealing with different resolution sizes

It is written as follows:

```
min-height: 60px;

max-height: 40px;

min-width: 80px;

max-width: 30px;
```

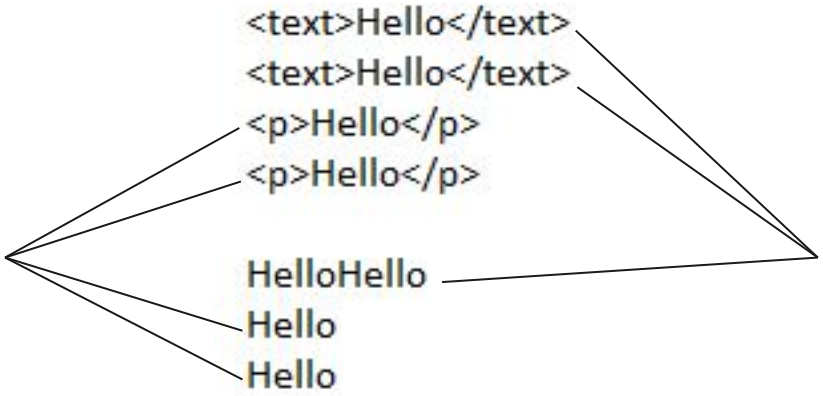
Lets try it

Try playing around with width and height

```
min-height: 60px;

max-height: 40%;

min-width: 2em;

max-width: 30px;
```

Positioning on webpages

When you add elements to your page you are creating a series of blocks that can be manipulated around your page. Elements behave different based on how they are defined. An example of this is text vs paragraph.



Paragraph does will take up the full span of a page

Text does not take up the full span of a page

Moving things around on your page

One of the most difficult things to accomplish on a web page is to get elements to display in a specific order or location. The below CSS properties were designed to help aid in this.

- Display
- Position
- Float

Position

The Position property determines where an element will be positioned on a page relative to the other elements on the page. Some examples are

- Static - ignores top, bottom, left, and right properties
- Sticky - positioned based on the user's scroll position
- Relative - positions itself relative to its normal position
- Fixed - always stays in the same place even if the page is scrolled
- Absolute - like fixed but can be overlapped on top of an element

Lets try it

```
position: static;

position: sticky;

position: fixed;
```

Display

The Display property works similar to Position, however it helps specifies the behavior of an element on the page

If your elements are “jumping” around your page try using Display

Display becomes increasingly important when you are creating a page that is both desktop and mobile friendly

The two most popular Display properties used are:

- Inline - Displays an element as an inline element. Any height and width properties will have no effect
- Block - Displays an element as a block element. It starts on a new line, and takes up the whole width

Lets try it

```
{display: inline;}

{display: block;}

{display: listitem;}

{display: none;}
```

Float

The Float CSS property will push an element to the left or right of your page and allow other elements to wrap around it.

Lets try it

```
{float: right;}

{float: left;}
```

Useful when creating layouts

box-sizing:border-box; <-includes padding and border size into size of element

```
* {

  box-sizing: border-box;

}
```

The above will apply this to all elements and is recommends to not have to keep adjusting padding and borders to get things to line up right. Put anywhere in your CSS