# INTRODUCTION TO CSS

Cascading Style Sheets

## INTRODUCTION

We have gotten pretty decent at getting content into our pages. We can add in tags to get text, images, and even videos.

Our problem up until this point is it looks pretty lame. Really lame even. We need to spice things up! We need color, different fonts, layout changes and even animation!

This is where CSS (cascading style sheets) comes in. Once we have the content on the page with our HTML code, we add some style with CSS.

## GETTING CSS INTO HTML

There are three different ways to style your HTML pages with CSS:

- **Inline** where you use the “style” attribute of the HTML tag
- **Internal** where you use the <style></style> HTML tag
- **External** where you use the <link> tag to reference a specific CSS file

We will go through each one of these methods, but as a general rule of thumb you want to stick to external CSS as much as possible.

Using inline or internal CSS not only clutters your HTML code up, but it also actually matters when you have style rules that conflict with each other!

## INLINE CSS

Inline CSS uses the **style** attribute and can be applied to any tag in your HTML code.

For example:

```
<p style="color: red;">I am red text!</p>
```

Here we have injected our CSS rule right into the HTML tag. This is **highly discouraged** and you will get docked marks if I see this in your code.

It really makes your HTML code confusing to read for humans as the CSS can get very large.

## INTERNAL CSS

Internal CSS is when you use a **style** html tag. It is pretty simple, you create a style tag and put the CSS you want inside of it:

```
<style>
      p {
          color: red;
      }
</style>
```

It is okay if you don't understand the code in between the style tag yet, that is CSS code! We will go over that later in the slides. Again, we want to avoid this at it makes our HTML code messy.

## EXTERNAL CSS

External CSS uses totally different files to write our CSS code. This is the best method as we can re-use CSS code in different HTML pages (two different HTML pages can make use of the code written in one CSS file).

To use external CSS, we need to use the **link** HTML tag. The link tag should have two attributes:

- rel – this tells the HTML code what kind of relationship the link has to the page
- href – this is the location of the CSS file itself. This can be an internal or external link just like an image or a tag!

```
<link rel="stylesheet" href="style.css">
```

The one thing you need to do is make sure this tag goes in the **head** tag of your HTML! If you don't, the webpage can load slowly and your user will see an ugly bare HTML page and then the site will be styled.

## KNOWLEDGE CHECK

Let's see if we can play around with CSS and get some things changing.

1. Create a folder called CssIntro in your Scratch directory using either mkdir or your project starter
2. Follow the Git/GitHub steps and turn this into a repository
3. In the HTML file, add in some content. Make sure you have at least 1 p tag.
4. Add, commit and push your code.
5. Use **inline style** to change the **font-color** to **green**
6. Use **internal css** to change the **font-size** to **24px**
7. Use **external css** to change the **font-family** to **monospace**
    1. Call the new file **style.css**
8. Add, commit and push your code.

## ANATOMY OF A CSS RULE

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

The selector is used to define the HTML element we are targeting with the property, to give a value. Each property / value pairing is called a declaration.

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

## SELECTORS

CSS rules are designed to target specific HTML elements (not just tags!). We use selectors to specify which element(s) we want to target.

There are many different selectors. They can target something as simple and generic as “all p tags” or be as specific as targeting one element on the page. We will start off with the most basic in the **tag selector.**

Targeting an HTML tag:

```
p {
  ...
}
```

## SELECTORS CONT.

Next we will consider the **class selector.** This is the first selector that requires us to make some changes in our HTML. We use the **class** attribute on a tag and then use the selector:

**.class-name**

HTML:

```
<p class="red_text">Hey there!</p>
```

CSS:

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

## SELECTORS CONT.

Next we will consider the **id selector.** Again, this selector requires us to make some changes in our HTML. We use the **id** attribute on a tag and then use the selector:

**#id-name**

HTML:

```
<p id=“bold_greeting”>Hey there!</p>
```

CSS:

```
#bold_greeting {
    font-weight: bold;
}
```

## SELECTORS CONT.

We have some slides dedicated to the more complicated selectors, but for now just being introduced to classes, IDs and tag selectors is enough to get the ball rolling.

You might be wondering what to change with these selectors, but that is a large question!

Here is a list of the currently supported CSS properties: https://developer.mozilla.org/en-US/docs/Web/CSS/Reference#index

It is a lot, but just like HTML you will start to have a smaller core group of properties that you will use almost 90% of the time.

## KNOWLEDGE CHECK

Okay, let's make use of the basic selectors.

1. Add at least 6 more content tags to the knowledge check from before (add a variety of different tags).
2. Write 3 tag-based selectors in your CSS file
    1. Feel free to change anything about them that you like. Easy ones are color.
3. Give 3 tags in your HTML the **same class**
4. Write a class-based selector in your CSS and change whatever you like
    1. Another easy one would be something like font-family or background-color
5. Give 2 tags **unique** ids in your HTML
6. Write 2 id-based selectors in your CSS and change whatever you like
7. Add, commit and push.

## CSS PROPERTIES

So that's it for the most basic selectors we can use. Trust me, they get much more involved and complicated for when you want to select much more specific groups of tags.

The next idea we need to cover in CSS is the **property.** These are essentially the rules that change the look of something. CSS properties take the following general form in most cases:

***property*: *value*;**

For example:

**color: red;**

This will change the text color of whatever you have selected red. You can add as many properties to the inside of a selector as you want!

## CSS PROPERTIES

There are simply too many properties to cover. If we wanted to cover them all we would have to come to class for the next 3 months and just talk about each property one by one to cover them all.

We will talk about the very important properties, but most of your learning in CSS properties will be through experimentation.

There are lots of nuances and rules in CSS that we have not talked about. This is just to introduce us to the topic.

At least now we can start building websites that look better than the plain HTML we have been creating so far!