Cascading Style Sheets

better known as CSS

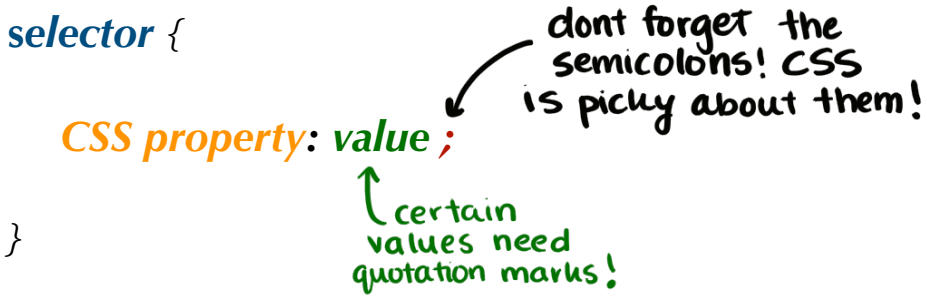
CSS is the style sheet language used to describe the look of a document written in a markup language. Which means CSS does not stand alone! It's how we edit the aesthetics and style of our HTML elements!

CSS files are referred to as style sheets. You can connect your CSS to your HTML and JS files using an external link in the head tag of your HTML element.

```
<head>
    <link rel = "stylesheet" href = "styles.css">
</head>
```

CSS is made up of "rules" that use a certain selector to style an element or set of elements. These rules use defined CSS properties to change certain aesthetic properties.

The Anatomy of an CSS Rule



The Selector:

The selector is a keyword to refer to the element or group of elements the rule will style.

The CSS Property:

The property is the keyword referring to the defined aesthetic property of the element that you are trying to change. A full reference can be found **here.**

The Value:

The chosen value for the CSS property.

CSS Selectors

Element Selectors

You can use the element tag as a selector to style ALL elements of that type the same way. For example, if I want to change the alignment of ALL the h1 headers in my document, I can use a h1 selector:

```
h1 {

    text-align: center;

}
```

Id Selectors

You can use an element's id to style a specific element. Remember each element needs a UNIQUE id, so this selector is useful to style ONE specific element at a time!

Id selectors are signified with a # before the id name.

```
<p id = "specialP" > This is a specially styled paragraph </p>

    #specialP {

        color: "red";

    }
```

Class Selectors

You can use an element's class to style a specific group of elements. Remember multiple elements can be in one class!

Id selectors are signified with a . before the class name.

```
<p class = "big" > Big Paragraph 1 of 2 </p>
<p class = "big" > Big Paragraph 2 of 2 </p>

    .big {

        font-size: 22px;
    }
```

Combining Selectors

There's a couple different ways to combine selectors to get even more specific and fancy when you're styling! For example, styling only paragraphs in a certain class, or styling elements that are in two specific classes, etc. Here's how to do some of those!

Styling A Certain Type of Element in a Specific Class

```
elementSelector.ClassSelector {

    property : value;

}
```

Example:

```
p.big {

    font-size : 22px;

}
```

Styling Elements in More Than One Class

```
ClassSelector1.ClassSelector2 {

    property : value;

}
```

Example:

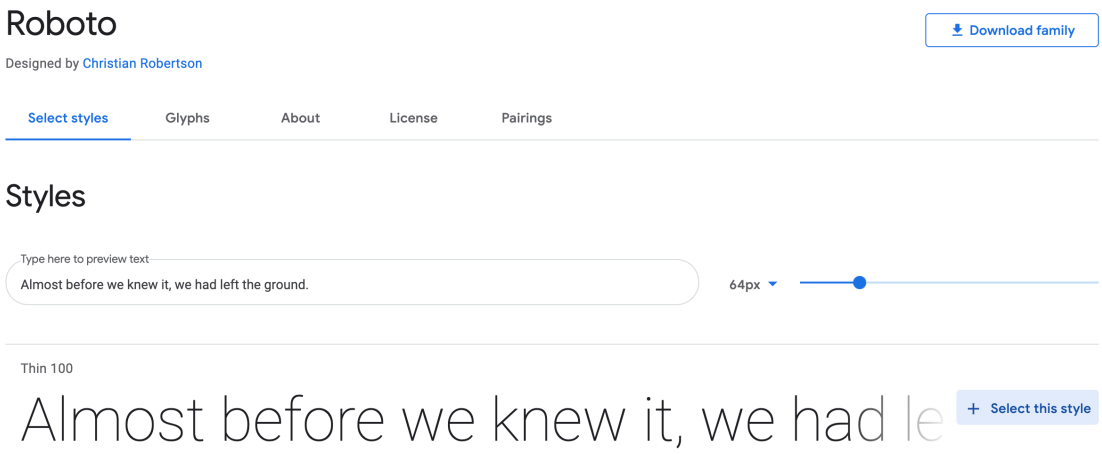
```
center.big {

    text-align: center;
    font-size : 22px;

}
```

Importing Fonts

Fonts are major! Typeface can deeply affect the aesthetics of your websites. You can change the font of the text of any element using the CSS property font-family, among others. But how can you find the fonts to choose from and import new ones?

To import a new font, you need to first find the import url. There are a lot of free font websites out there, one of them being Google Fonts. In the following tutorial, I'll go through the steps of importing a font using google fonts. Using other font sites should be fairly similar though!

Step 1: Go to https://fonts.google.com/
Step 2: Find the font you want and click on it!
Step 3: Select the style that you want by clicking the "Select This Style" button



Step 4: A menu on the right side should pop up. Click on the "import" radio button to get the import link. Copy it to your clipboard!

Step 5: At the top of your CSS file, paste the code snippet with the import link in between the <style> tags.

A Rainbow of Useful CSS Links

9 Important CSS Properties to Know

Your First Website: Landing Page Tutorial

Navigation Bars

Basic CSS Templates

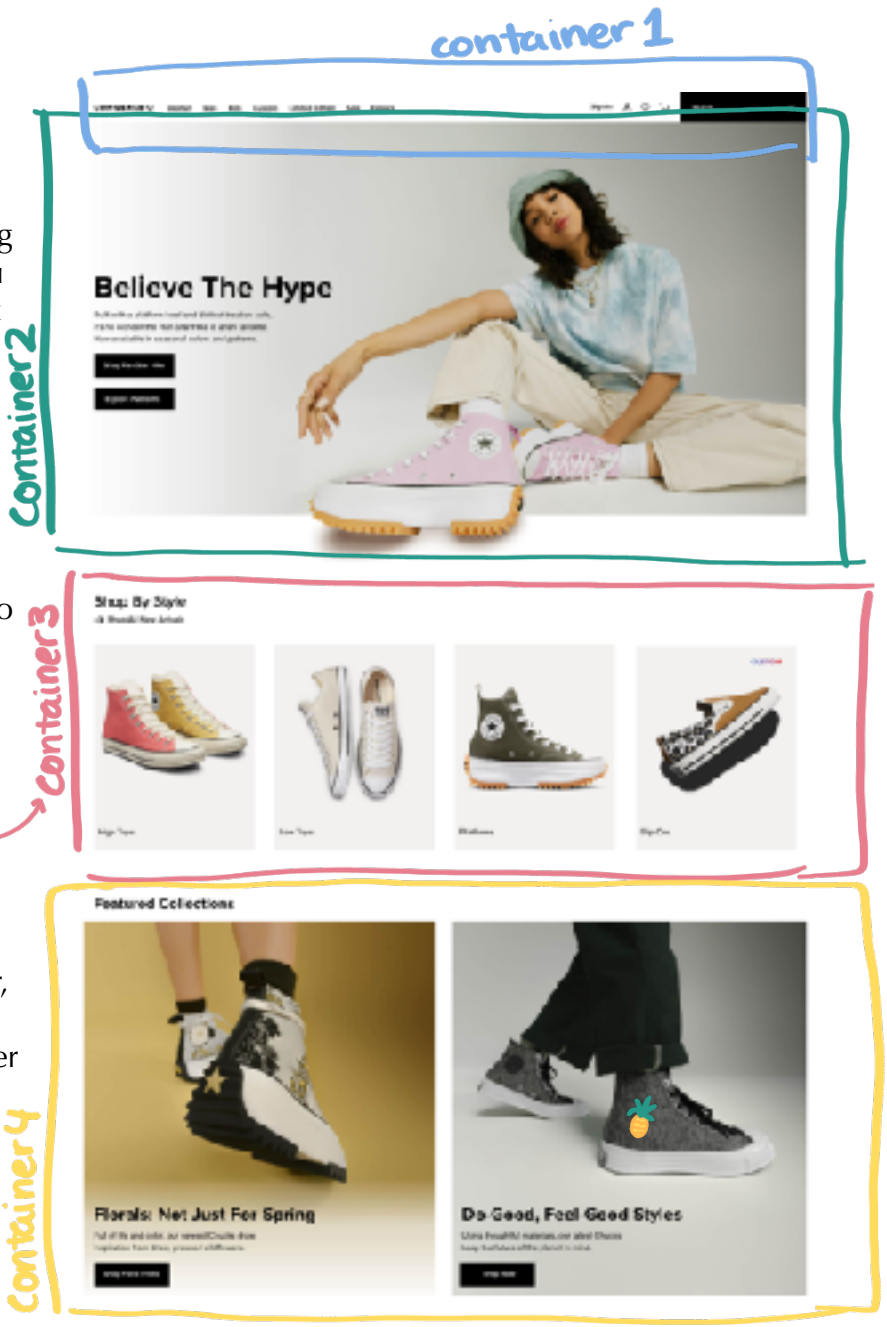
CSS Cheat Sheet

CSS Generators

Containers

One of the easiest and more aesthetic ways to organize content on a page is by creating containers of elements that you can style as one! Take a look at this website!

test yourself.!
pick another website and see if you can spot all the containers!



Here you have four visible containers: a top menu, a photo header, “Shop By Style”, and “Featured Collections”.

this container has a header, a paragraph, and four buttons that serve as links to other pages! 6 elements total!

Containers don’t have to be visually boxed in with a border, they are just collections of elements that are styled together to create a cohesive look.

How to create a container?
Easy! Put all the elements you want to group in a div and assign a class/classes to that div to style it cohesively! Certain CSS properties will help style your container.

Padding	Height/Width
Borders	Backgrounds
Margins	Display

Take a look at the example on the next page to get an idea!

Link to Full Example

```
.header-container {
  /* The image used */
  background-image: url("/imgs/banner.png");

  /* Full height */
  height: 100%;

  /* Center and scale the image nicely */
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
  margin-bottom: 50px;
}

.container {
  /* Check out the flex box ref. on the next page! */
  display: flex;

  background-color: “red”;
/* Padding refers to the space around the elements */
  padding: 5px;
  margin-bottom: 0px;
  margin-top: 0px;
  margin-left: 20%;
  margin-right: 20%;
  border-radius: 10px;

  flex-wrap: wrap;
}
```

CSS Flex Boxes

FlexBox is a CSS layout to make columns and grids that are *flexible* to the size of the page. Basically, we want to make sure your content doesn't get squished if someone has a tiny window or stretched out if they have a large window.

Let's try to build this container following the Flexbox layout.

First, you need to create your "parent" div which is the div that acts as the outer container and holds all the inner elements. Inside that parent div, you will place the individual divs of each element. The example below has a parent div, and three element divs. We've assigned the parent div the class "flex-container" to style the entire container as one!

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

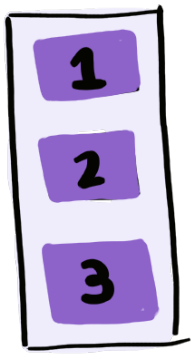
The flex container becomes flexible by setting the display property to flex.

```
.flex-container {
  display: flex;
}
```

The flex-direction property defines the direction you want to stack the items.

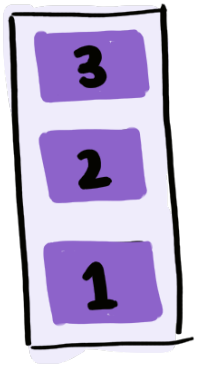
Vertically (Top to Bottom):

```
.flex-container {
  display: flex;
  flex-direction: column;
}
```



Vertically (Bottom to Top):

```
.flex-container {
  display: flex;
  flex-direction: column-reverse;
}
```



Horizontally (Right to Left):

```
.flex-container {
  display: flex;
  flex-direction: row;
}
```

Horizontally (Left to Right):

```
.flex-container {
  display: flex;
  flex-direction: row-reverse;
}
```

The flex-wrap property defines if you want the items to wrap around if needed.

Wrap Around If Needed:

```
.flex-container {
  display: flex;
  flex-wrap: wrap;
}
```

Do Not Wrap Around If Needed:

```
.flex-container {
  display: flex;
  flex-wrap: nowrap;
}
```

Wrap Around If Needed (In Reverse Order):

```
.flex-container {
  display: flex;
  flex-wrap: wrap-reverse;
}
```

And the other CSS flex properties you should know:

Property	Description
align-content	Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
align-items	Vertically aligns the flex items when the items do not use all available space on the cross-axis
display	Specifies the type of box used for an HTML element
flex-direction	Specifies the direction of the flexible items inside a flex container
flex-flow	A shorthand property for flex-direction and flex-wrap
flex-wrap	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
justify-content	Horizontally aligns the flex items when the items do not use all available space on the main-axis