Styling with CSS

Hunter Jorgensen

Background

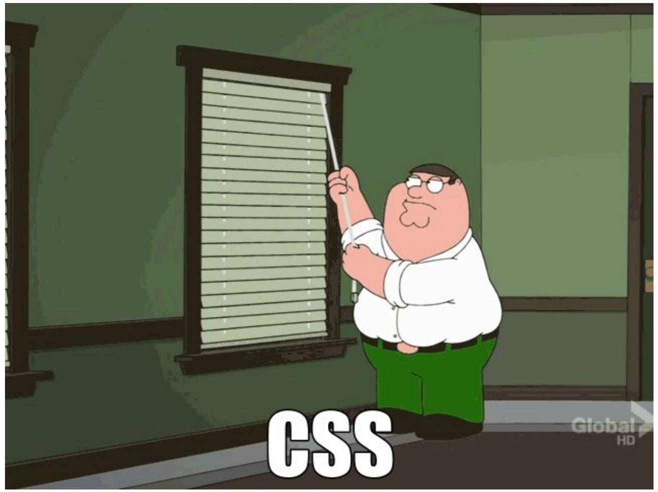
CSS has a less exciting history than HTML

Developed in 1994 by Hakon Wium Lie and Bert Bos, and then released publicly in 1996

The newest ‘version’ of CSS, CSS3, should actually be considered a set of recommendations, suggestions and requirements for browsers to support.

This approach is now modularized and new features are released at a steady pace. This can mean there is deviance between browsers, but this is less common today (i.e., for this class, just use a modern browser!)

What is CSS? It gets a bad rep



What is CSS

CSS - Cascading Style Sheet

The Cascading part refers to how style is applied to elements (it *cascades* down the hierarchy) until it is replaced by something else.

Presents rules on how data is styles, laid out, and more.

Can apply to XML, SVG, and others but most commonly associated with HTML

CSS Syntax

CSS is made up of a selector, property and value. In this example, the CSS is selecting a specific HTML tag (h1, in this case) and modifying its colors:

```
h1 {

    color: red;

    background-color: #4CAF50; /* both of these colors work! */

}

How do you think this will affect h1 tags on a page?
```

How to Use CSS

Can be linked in the head element:

```
<head>
     <link href="/some/external/style/file.css" rel="stylesheet"/>
</head>
```

Embedded in the body

```
<style>
 p { text-align: center; }
</style>
```

Defined in line:

```
 <p style="color: green">This paragraph is now green</p>
```

Unless specified otherwise or we are doing a demo, CSS should be imported via a the <link> tag.

Putting the Cascading in CSS

If an element is selected, any sub elements will also be affected by the styling (hence the ‘cascading’ bit):

```
div {
    color: purple;
}
```

```
<div>
    <h1>Hello, my Friend</h1>
    Did you know my cat is grumpy?
</div>

<h1> Goodbye, my friend </h1>
```

A note about HTML class and IDs

Nearly all HTML tags have 2 extra attributes:

id: a single string that uniquely identify an HTML tag. Should be unique on a page

class: space separate list of strings that be shared across multiple elements

```
<div id="firstDiv" class="purple-class small-font">First Div</div>

<div id="secondDiv" class="purple-class medium-font">Second
Div</div>

<div id="thirdDiv" class="purple-class small-font">Second Div</div>
```

Note about flow of style

Oftentimes, there might be conflict among different CSS selectors. So keep in mind that CSS applies based on specificity: any tag level style will be overridden by a class level style, which in turn will be overwritten by any ID level selectors.

Additionally, if a tag has several classes with conflicting style properties, the right most class will override the style.

This ONLY applies in the case of conflicting style properties.

Selecting Specifically

With HTML classes and ID’s, you can use CSS to specify exact elements to apply styling too (you still need to be wary of cascading/inheritance!)

```
#firstDiv {

    color: green;

}
```

```
.purpleClass {
    background-color: purple;
}

.small-class {
    font-size: 11em;
}
```

This is really the ONLY thing about CSS that you’re expected to memorize

Naming Classes and ID’s

In software development, naming is always a challenge. While we won’t focus on it too much in this class, it’s a good practice to name classes and ID’s based on what the content is related to on the page. So names like “login-page-username-field” is better than “field”.

Colors in CSS

When describing colors in CSS, you can use a name (`black`), and RBG value (`#000000`) or RGB function (`rgb(0,0,0)`) are all valid syntax.

When calculating RGB value if you append 2 more characters (`#000000c0`) or pass in a fourth argument `rgba(0,0,0,.25)`, this refers to opacity/transparency.

We'll talk more about colors when we revisit web design later in the course.

Rules

Some CSS rules have an optional annotations known @rules (or at-rules) that can further modify the styling:

```
@media(min-width: 600px) {
    h1 {
        color: purple;
    }
}
```

@media is the only one we'll care about since it controls what screen size a style will be visible at. But there are others that you can explore!

Demo

@media

@media is probably the most common CSS at-rule as it helps with scaling a screen.

A rule of thumb is for the following screen sizes:

Screen Size	Pixel Size
Extra Small (ipod, etc.)	< 576px
Small (cell phone)	>= 576px
Medium (table)	>= 768px
Large (computer)	>= 992px
Extra Large (large computer screen)	>= 1200px

Note: you made need to set the viewport in the <head> tag with:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

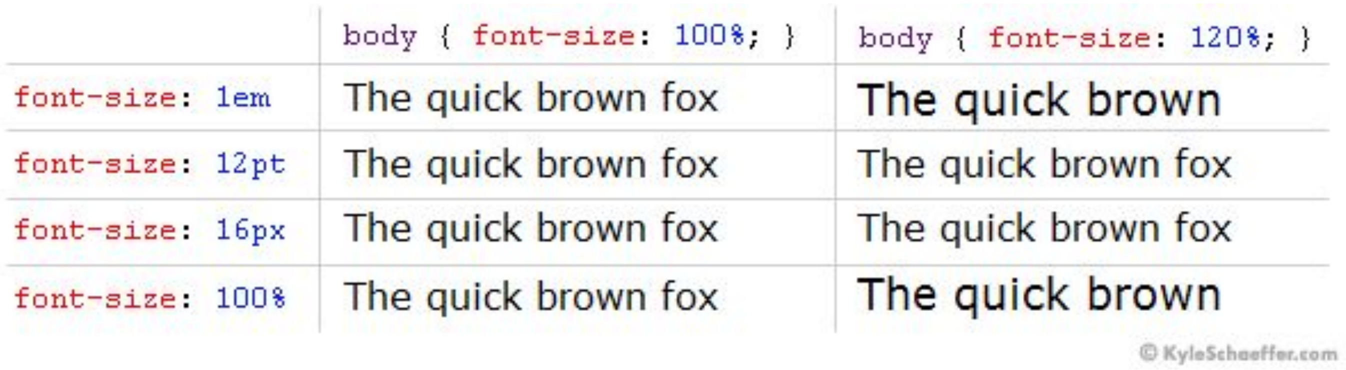
Numbers in CSS

One pecularity in CSS is that you can use a variety of increments to determine numbers. For the height property (for instance) you can use things like *em*, *px*, *%*, *in*, *rem*, etc.

Absolute Measurements (cm, mm, in, etc.) are based on the size of the screen

Relative Measure (em, ex, %, etc.) are based on the size of the parent element

[Demo]



	`body { font-size: 100%; }`	`body { font-size: 120%; }`
`font-size: 1em`	The quick brown fox	The quick brown
`font-size: 12pt`	The quick brown fox	The quick brown fox
`font-size: 16px`	The quick brown fox	The quick brown fox
`font-size: 100%`	The quick brown fox	The quick brown

© KyleSchaeffer.com

Number Chart Guide

There are a lot of number types: you don’t know need to know them all! I’ve listed out most of them, and highlighted important ones

Unit	Absolute or Relative	Equivalence
cm (centimeter)	Absolute	1cm = 96px/2.54
in (inch)	Absolute	1in = 2.54cm = 96px
pt (points)	Absolute	1pt = 1/72th of 1in
px (pixel)	Absolute	1px = 1/96th of 1in
rem	Relative	Relative to font size of root
vw/vh	Relative	% relative to screen width/height

Sizing and Positioning

You can control the size of an element with **height** and **width**. (watch out for overflow!)

You can control the positioning with **top**, **left**, **right**, or **bottom** (which are required for all positions except relative)

Position:

Relative position means that it's based off of the location of its normal location.

Absolute position means that it's based off of the location of the parent element.

Fixed position means it's based off the position of the browser.

Sticky position means that it's based off of the location of where the user scrolls.

Demo!

Z-index

The z-index determines the stacking level of the element: the higher the number, the higher it will be on the stack!

If we go to the previous demo, we can update the z-index value to experiment!

Float

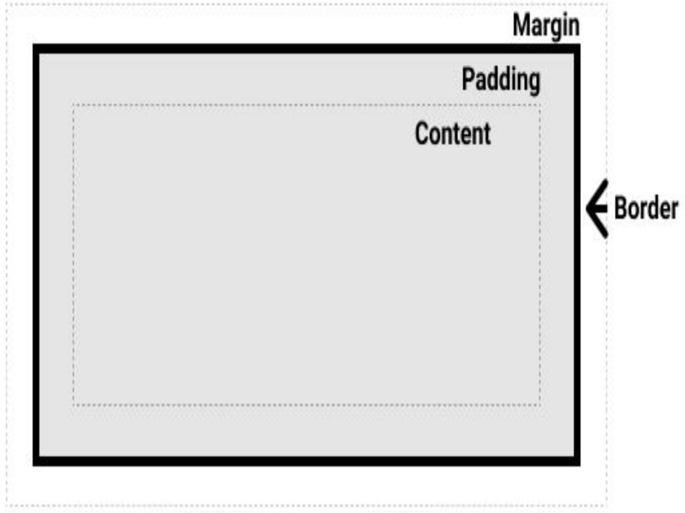
It puts an element to the left or right of it’s parent element (also known as container) and removes it from the normal flow of the page (and allows in-line elements to wrap around it). In the demo below, you’ll see that the text will “float” around the box depending on the float settings.

Demo

Box Model

One helpful way to think about CSS affecting elements is through the box model:

- You’ve seen width/height and that refers to content only
- **Margin** is the space between the next element
- **Padding** is the space between the content and the border
- **Border** is the edge of the content
- Absolute Height/Width is the combination of all of these, EXCEPT for the margin



Flex

Flex is a relatively new but incredibly important part of styling. It allows for a parent component to manage the spacing and sizing of multiple children components. Essentially, it allows us to nicely create rows, columns, etc.

A parent, or container, element needs to have the property: *display: flex;*

Then each individual element can be modified as needed using the *flex* property.

Demo

Reference

Relational Selectors

There are also more advanced selectors, that rely on the relationship between elements:

Name	Syntax	Explanation
Grouping Selector	A, B	Styling applies to A and B
Descendent Combinator	A B	Will select if B is a descendent of A of any level
Child Combinator	A > B	Select if B is a direct descent of A
General Sibling	A ~ B	Select if B and A have same parent, and B is after A in template
Adjacent Sibling	A + B	Select if B is sibling of A, and immediately follows
Attribute Selector	[attr=value]	Applies if HTML attr equals value

Pseudo Classes

Indicated by the ‘:’ - allows for selection based on information outside of document tree.

With this, you can do some clever styling and handle basic interaction. For instance, you can change the styling when a user is hovering over an element, or the styling the first/last element of a list.

Demo

Pseudo Elements

Pseudo elements, defined by the ‘::’ notation, allows you to style certain parts of an element, such as the first line of a paragraph, the content immediately before/after and so forth.

Demo

Transitions/Animations

Transitions control how a CSS property might go from one state to another. A transition takes a unit of time, and controls how long it takes for a change in state to occur. Usually, this happens when JavaScript dynamically adds or removes classes, but with pseudo classes, we can also use this property.

Demo

Notice that as we hover over the box, the we transition from one value of the property to another.

For more control, we can use the @keyframes @rule

Variables

CSS3 allows us to set simple ‘variables’ that apply from the parent to all of its children. So if we want to set common values, this is a good way to do it.

See last demo for how this works

Debugging

Your browser is an excellent tool to debug CSS!

It allows you to edit values on the fly and to see the entire box model.

!Important

One last piece: you can override the normal cascading flow by setting a value as important

```
h1 {

	border: none !important;

}
```

This will force the property to have that value for that selector.

However, this is VERY bad practice and should only be used as a last resort.

Some Inspiration...

CSS is a VERY powerful tool, with lots of functionality. I only really touch on the basics in this course.

While I don't expect this level of advanced use, there are a lot of interesting things you can do with CSS alone. For instance, a student last semester showed me this blog with some cool ideas. Please be inspired by what is possible!

In Class Lab

Create 3 squares that have at least 3 of the following properties:

- The squares are in a horizontal row, not a vertical column
- Is fixed in the center of the browser screen (use Lorem Ipsum to generate a bunch of text for your page)
- When you hover over the button, the color of square changes from red to green
- Has a purple border but when the screen is less than 600px wide, the border disappears
- The squares have 20px space in between each of them

Submit to Nandish for 2 points