Introduce the idea of accessibility (commonly shortened to a11y, with the "11" representing the eleven characters between "a" and "y"). Accessibility has to do with making our web pages usable by different kinds of users (for instance, both sighted and visually impaired users)

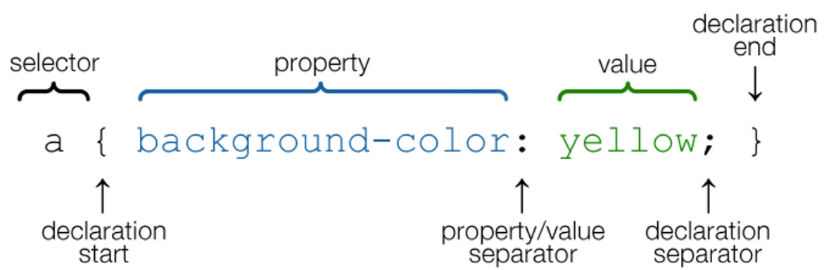
Note that default styling varies between browsers. If you create an HTML page with no CSS and view it in Chrome and Firefox, it probably won't look exactly the same. In order to cancel out style differences between browsers, frontend developers often use what is called a CSS reset to zero out default styles and ensure a consistent user experience across browsers. We'll learn more about CSS and CSS resets later in this unit.

**Visual Studio Code Quick Way For Boilerplate** → ! + Tab

**Three Ways to Add CSS to HTML**

1. Inline style via *<h1 style=“color: red”* will make the h1 element red
2. Internal style sheet by declaring *<style>* in the <head> element
3. External style sheet by linking to an css file via *<link rel= “\_\_” type= “\_\_” href= “\_\_”>*

**CSS Selectors:**



**Colors in CSS:**

* Color names (aka red, blue, yellow)
* HTML5 Color names
* Hexadecimal
  + Specified with #RRGGBB where RR, GG and BB are hexadecimal integers between 00 and FF specifying the intensity of the color
  + #0000FF is the *bluest blue*
* RGB
  + Each parameter defines the intensity of the color as integer between 0 and 255
  + rgb(0, 0, 255) is the *bluest blue*

*display: inline;* to only extend the css styling to the length of the content inside

* Need to apply to <h1> for example, because it is a block element

**CSS Class and ID**

**ID** is styled with *#name* and is **unique to one** element | **Class** is styled with *.name*

The difference between an ID and a class is that an ID can be used to identify one element, whereas a class can be used to identify more than one.  
**ID > Class > Tag** when conflict between CSS attributes

Nested elements with classes are more specific than classes on the outside element

**To apply more than one class to an element just write it with a space in the class attribute**

**Explanation of the CSS Box:**

* Content - The content of the box, where text and images appear
* Padding - Clears an area around the content. The padding is transparent
* Border - A border that goes around the padding and content
* Margin - Clears an area outside the border. The margin is transparent

**Crash Course is not covering the following advanced topics (look on own time):**

1. Flexbox (Traversy Media has video on flexbox)
2. Animation
3. Transitions
4. Pseudo Elements

Stands for **Cascading Stylesheets**, a stylesheet/styling language and is used for website layout and design. Can be extended with Sass/Less (to extend the functionality of CSS)

Look up what HTML **iframe** is

**font-family** will declare the font, you can specify other fonts in case the first one doesn’t load. If you want to import fonts (Google fonts), you need to import it like a style sheet

**font-size**

**font-weight** (normal, bold, italics)

**font:** font-weight, font-size, font-family

**text-decoration**

**text-transform**

**letter-spacing**

**word-spacing**

**line-height** is the height between the lines (em is relative to the font-size of the element)

**margin: auto** will set an auto margin on all the sides

Setting width to percent (80% for example) makes page responsive when screen size changes

Use height the same way as width. *Min-height* and *min-width* will scale the height,width bigger as the screen size changes and will begin with specified minimum

**Padding** - is the space in between that element and its border

**Border** - contains size of border, color and styling of border

* *border-radius*

**Margin** - is the space outside of the content

Every element in HTML has some default padding and margin and if you want to zero everything out in the beginning you use the **\* { }**

**text-align**: will align the text to the center, left, right, etc..

**list-style** to style the bullet points in the list

**list-style-image: url(‘../images/check.png’)**

**CSS Alignment and Floating**

**float:** parameter will shift the text to the argument

**box-sizing:border-box** will take the border, padding into account with the width and make the elements fit on the same block. **Flexbox can do the same more efficiently so check that out**

* When you float and you’re finished, you want to clear it or else will look messed up. In order to float it, make div with class=“clr” and add css styling clear:both

**Positioning in CSS**

* **Static** (default value) renders element in order of the document flow
* **Relative** element is positioned relative to its normal position
* **Absolute** will allow us to target whatever position we want **inside of a relative element**
* **Fixed** fixed pos. to browser window (in the same position no matter where we scroll)
* **Initial** sets property to its default value
* **Inherit** take the property of its parent element

Relative and absolute to position elements are helpful when trying to make games with CSS

Fixed position is useful for things like social media links or navbar (always want user to see it)

* *overflow* property specifies what should happen if content overflows an element’s box. Property specifies whether to clip content or to add scrollbars when an element’s content is too big to fit in a specified area
  + Visible (default value)
  + Hidden: when overflow is clipped, the rest of the content will be invisible
  + Scroll: the overflow is clipped but the scroll-bar is added to see the rest
  + Auto

**How to Use Image as Element Background**

*background-image: url(“\_\_”);* link to the path to the image that you want

*background-position* sets starting position of the bg img (can use percentages, pixels, or even center) For example: *background-position: center center* will center it vertically and horizontally

*background-repeat: no-repeat* if you don’t want the image to repeat

**CSS Pseudo Classes**

Used to target certain things

*.my-list li:first-child {}* will target the first child of li within the class my-list

*.my-list li:last-child {}* same concept as the above but for last child

*.my-list li:nth-child(x) {}* will target the argument within nth-child

* **nth-child** can take arguments even and odd

**To Make Responsive at Mobile Size**

Add a media query: only going to be in effect if the screen size is under 600px

*@media(max-width:600px) {}*

**Cursor**

*cursor: “attribute”* ranging anywhere from pointer, auto, help, progress, etc..

**Box Shadow Generator**

https://www.cssmatic.com/box-shadow

**Best CSS Selectors to Memorize:**

1. *h1 p {}* will select all of the paragraphs within h1 tags
2. *.my-form input[type=“text”]* select all of the inputs with type text within the class my-form
3. *div, p* selects all of the div elements and all the p elements
4. Buttons and anchor tags can have different states access them with the colon **:**
   1. a:visited {} will make css styling to the visited links of the anchor tags