Lesson 3: Getting to Know CSS

The Cascade

- all styles cascade from the top of stylesheet to bottom → styles can be overwritten

  - also works w/ properties w/in individual selectors

Calculating Specificity

- every selector has a specificity weight
- specificity weight & placement in cascade → determines how styles will be rendered
- selector specificity weight:

type selector	0-0-1	low	id > class > type
class selector	0-1-0	medium	# .
id selector	1-0-0 ↓	high	
	i.c.t.		

Combining Selectors

- combined selectors: to specify which element/group of elements to select
  should be read right → left

  → (eg. .hotdog p {...}): will only select paragraph elements that reside w/in an element w/ a class attribute value of hotdog

- **prequalifier** = any selector to the left of the key selector
- **key selector** = selector farthest to the right; right before the {
  identifies exactly which element the styles will be applied to

Specificity Within Combined Selectors

- combined specificity weights = combined specificity weights of the individual selectors

  → (eg. .hotdog p): 0-1-0 + 0-0-1 = 0-1-1 → 0 id selectors, 1 class selector, 1 type selector

- the ↑ specificity weights rise, the ↑ cascade is to break

Layering Styles with Multiple Classes

modular = sharing similar styles from element to element

- ↑ modular, keep selectors' specificity weights ↓
- solution: layer on different styles using multiple classes

  - establish reused style first & layer more specific styles from another class later

  → (eg. .btn { font-size: 16px; }
  .btn-danger { background: red; }
  .btn-success { background: green; }): all buttons have a font size of 16px, but bg color of buttons vary depending on where button is used

Common CSS Property Values

- colors: defined on an **sRGB** = standard red, green, blue

  - ways to represent colors w/in CSS: keywords

hexadecimal notation
RGB values
HSL values

- **keyword color values** = names that map to a given color
  determined by CSS specification
  includes most common colors & few extras
  unpopular method due to limited options

- **hexadecimal colors** = widely supported → most popular

  - format: #RGB (3 characters)
    #RRGGBB (6 characters)

  - characters: numbers 0-9
    letters a-f

  - 0 = black
    f = white

  → (eg. #ff6600 = #f60): matching pairs in 6-form can be shortened to 3-form

- **RGB & RGBa colors** = preferred for alpha channel transparency

  - format: rgb(r#, g#, b#) / rgba(r#, g#, b#, alpha/transparency #)

  - # range: r, g, b 0-255
    a 0-1

  - 0 = black / 0 = fully transparent
    255 = white / 1 = fully opaque

  → (eg. rgb(255, 102, 0)): orange

  → (eg. rgba(255, 102, 0, 0.5): 50% opaque orange

- **HSL & HSLa colors** = newest color value available

  - format: hsl(hue #, saturation %, lightness %) / hsl(hue #, saturation %, lightness %, alpha #)

  - ranges: h# 0-360 (degree on color wheel)
    % 0-100
    a# 0-1

  - S0% = grayscale / 10% = completely black / 0 = fully transparent
    S100% = fully saturated / 100% = completely white / 1 = fully opaque

→ (eg. hsl(60, 100%, 50%)): yellow

→ (eg. hsl(60, 100%, 50%, 0.5)): 50% opaque yellow

- lengths: different types of length values

  - 2 forms: absolute
    relative

- **absolute lengths** = fixed to a physical measurement → simplest length values
  (eg. in, cm, mm)
  most popular = px

- **pixels** = 1/96th of an in.; 96 pixels = 1 inch

  → (eg. font-size: 14 px;)

- **relative lengths** = rely on the length of another measurement
- **percentages** = helpful for setting height & width of elements
- **em** = calculated based on an element's font size
  used for styling text, spacing around text, margins & padding

  → (eg. width: 50%;): set's element's width to 50% of element's parent's width

  → (eg. font-size: 14px; width: 5em;): width = 14 x 5 = 70 px
  if font-size isn't indicated, it will grab the stated font size of the closest parent element

- **universal selector (*)** = selects every element