5. CSS properties

5.1 Introduction to CSS properties

- We have seen during our last class that a CSS rule consists of a selector and a declaration block. In the example below, h1 is the selector, everything within the curly braces is the declaration block, color is the property and blue is the value.
    - Example: h1 {color: blue;}
- The CSS selector determines what HTML elements to target with the CSS rule. The CSS properties specifies what to style of the targeted HTML elements.
- Be aware that any syntax error in a rule definition invalidates the entire rule. Invalid rules are ignored by the browser.
- Refer the following link to see the CSS properties available for use:
    - https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

5.2 Introduction to most common CSS properties

- CSS properties are like words in a language. You don't have to know every single word in the dictionary to form a sentence and communicate. The more words you know the better. But the key is in effectively using the ones you know.
- The most important CSS properties include display, width, height, margin, padding, color, background and font.
- Please refer this link to see the 9 important CSS properties you must know:
    - https://zellwk.com/blog/9-important-css-properties-you-must-know/

5.3 Display property

- The display CSS property determines how an element is going to be displayed on a browser. The property sets whether an element is treated as a block or inline element. By default, the block mode is assumed. Below are the most common values for display property:
    - **block** (this causes an inline element to act like block-level element. It makes an inline element start on a new line and take up take up as much horizontal space as it can
        - Example: span {display: inline} makes all spans to start on a new line and take up take up as much horizontal space as they can

- **inline** (this causes a block-level element to act like an inline element)
    - Example: p{display: inline} causes all your paragraphs to display on the same line
- **inline-block** (this causes the block-level element to act like an inline element, but inline-block allows for additional width and height properties to be applied to the element)
- **none** (this causes the element to be completely removed)
- **Flex** (a flex container expands items under it to fill available free space or shrinks them to prevent overflow. The width of flex items automatically adjusts to fit inside the container.)

5.4 Position property

- The position CSS property sets how an element is positioned in a document. Elements are then positioned using the top, bottom, left, and right properties. Note: top, bottom, left and right properties will not work unless the position property is first set. The 6 types of positioning in CSS are:
    - **static**: this is the default value, meaning, is not positioned in any special way; it is always positioned according to the normal flow of the page.
    - **sticky**: the element is positioned based on the user's scroll position. A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport, then it "sticks" in place (like position:fixed).
    - **fixed**: the element is positioned related to the to the viewport or browser window. This means that, it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
    - **relative**: the element is positioned relative to its normal position. Setting the top, right, bottom, and left properties of a relatively positioned element will cause it to be adjusted away from its normal position
    - **absolute**: positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed position)
- Please refer to this link to visualize the effects values of position property: https://www.w3schools.com/css/css_positioning.asp

5.5 Width and height

- The CSS height and width properties are used to set the height and width of an element. The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

```
Ex:  div {
        height: 200px;
        width: 50%;
      }
```

- **The height and width properties may have values in the following**:
  - **auto** - this is default. The browser calculates the height and width
  - **length** - defines the height/width in px, cm, em, rem etc.
  - **Percentage (%)** - defines the height/width in percent of the containing block
  - **initial** - Sets the height/width to its default value
  - **inherit** - The height/width will be inherited from its parent value
- **max-width vs width**
  - **max-width**: it sets the maximum width an element can have. When the width of the browser window/viewport is smaller than the specified width of an element, the browser adds a horizontal scroll bar to allow us to view the element by scrolling to the side. In such cases, it is better to use max-width to set the width of the element not to be above the width of the browser/viewport.
  - **width**: if an element has a width less than the browser's/viewport's width, the width property will be used as opposed to the max-width property.
- Please refer to this link to visualize the effects values of width and height properties: https://www.w3schools.com/css/css_dimension.asp

5.6 Margin and padding

- Margin and padding are the two most used properties for dictating space between elements. In CSS, the term "box model" is used when talking about design and layout. The CSS box model is essentially a box that wraps around every HTML element. It consists of margins, borders, padding, and the actual content (text or image content). In the middle of the box, you have the content area (let's say an image), surrounding the image, you have the padding, surrounding padding, you have the border and surrounding the border, you have the margin.
- **Margin**: a margin is the space outside something. In CSS, margin property controls the space outside an element. Margin is outside of the border
    - Margin properties take top, bottom, right and left values. Example: margin-top, margin-right, margin-bottom, margin-left
    - Tip: you can center an HTML element if you give a 0 margin at the top and bottom of an element and auto to the left and right of an element.
- **Padding**: a padding is the space inside something. In CSS, padding property controls the space inside an element. Padding is inside the border
    - Padding properties take top, bottom, right and left values. Example: padding-top, padding-right, padding-bottom, padding-left
- **Shorthand for padding and margin values:**
    - If the margin/padding property has four values (margin: 25px 50px 75px 100px):
        - The margins apply to the top (25px), right (50px), bottom (75px), and left (100px) in a clockwise order
    - If the margin/padding property has three values (margin: 25px 50px 75px):
        - It means that top margin is 25px, right and left margins are 50px and bottom margin is 75px
    - If the margin/padding property has two values (margin: 25px 50px):
        - It means that top and bottom margins are 25px, right and left margins are 50px
    - If the margin property has one value (margin: 25px):
        - It means that all four margins are 25px

5.7 Applying margin and padding properties to Puppy Lover's page project

- Watch this class demo video to see how margin and padding are applied to Puppy Lover's page: https://youtu.be/o63_SjND94Q

5.8 Borders

- The CSS border properties allow you to specify the style, width, and color of an element's border. The border property is a shorthand property for the following sub-properties:
    - border-width
        - The border-width property sets the width of an element's four borders. This property can have values like thin, medium, thick or values in pixels.
    - border-style
        - The border-style property specifies what kind of border to display. A border-style can have dotted, solid, dashed or other values. Please note that it is always required to specify the border-style
    - border-color
        - The border-color property sets the color of an element's four borders.
        - If border-color is omitted, the color applied will be the color of the text.
- Example: border: 1px solid black;

5.9 Background

- The background property is used to set the background of an HTML element. Please note that, the background of an element can be an image or color. The background property can be a shorthand property for the following sub-properties:
    - background-color (specifies the background color to be used)
    - background-image (specifies ONE or MORE background images to be used)
    - background-position (specifies the position of the background images)
    - background-size (specifies the size of the background images)
- Example on how to set different background properties in one declaration

```
body {
        background: green url("img_tree.gif") center cover;
}
```

5.10 Fonts

- The CSS font property is used to change the face of the font of a text. The font property can be a shorthand property for the following sub-properties:
  - **font-style**: the font-style CSS property sets whether a font should be styled with a normal, italic, or oblique face from its font-family
  - **font-weight**: the font-weight CSS property sets the weight (or boldness) of the font. The weights available depend on the font-family that is currently set.
  - **font-size**: the font-size CSS property sets the size of the font. Changing the font size also updates the sizes of the font size-relative <length> units, such as em, ex, and so forth.
  - **font-family**: the font-family CSS descriptor allows authors to specify the font family for the font specified in an @font-face rule.
- **The @font-face**: the CSS at-rule specifies a custom font with which to display text; the font can be loaded from either a remote server or a locally installed font on the user's own computer. See the example below:

```
@font-face {
        font-family: "Open Sans";
        src:    url("/fonts/OpenSans-Regular-webfont.woff2") format("woff2"),
                url("/fonts/OpenSans-Regular-webfont.woff") format("woff");
}
```

- **To add custom fonts locally, refer instructions from this website:**
  - https://www.pagecloud.com/blog/how-to-add-custom-fonts-to-any-website
- **Refer this link to use Google Fonts in your website**:
  - https://www.freecodecamp.org/news/how-to-use-google-fonts-in-your-next-web-design-project-e1ad48f1adfa/

5.11 Color

- The color CSS property sets the color value of an element's text and text decorations. Please note that the color we set for our text may be used as an indirect value on other properties and is the default for other color properties, such as border-color.
- In CSS, a color can be specified using different ways. Browsers understand if you use any of the ways to set color of a text. Here are the three ways:
    - Setting color using a hexadecimal value (hexadecimal representation of the RGB numbers). A hexadecimal color uses a mix of six numbers and is specified with #RRGGBB. Example: body {color: #92a8d1;}
    - Setting color using an RGB value. Example: body {color: rgb(201, 76, 76);}
    - Setting color by simply using name of the color. Example: body {color: red; }

5.12 Priority order in CSS

- CSS rules often conflict with one another. In fact, this is what we want. The trick is understanding how conflicting rules will apply. In CSS, styles sheets cascade by order of importance. If rules in different style sheets conflict with one another, the rule from the most important style sheet wins.
- Priority order in CSS means that the more specific rules override more general ones. Specificity is defined based on how many IDs, classes, and element names are involved, as well as whether the! important declaration was used. When multiple rules of the same "specificity level" exist, whichever one appears last wins. For example, values defined as important (!important) will have the highest priority. Inline CSS has a higher priority than embedded and external CSS or class definitions.
- In short, you can look at the below expression to understand which css takes priority order over the other:
    - Value defined as !important > inline CSS > embedded styles (internal style sheets)> id nesting > id > class nesting > class > tag nesting > tag.