# Cascading Style Sheets (CSS)

## Ways to Add CSS to HTML:

1. **Inline** - Targeted to a single element

<h1 style="color: blue;">Style Me in Blue!</h1>

1. **Internal** - Targeted to a single web page, the below code written within the <head > section of a HTML document.

<style>

h1 {

color: red;

}

</style>

1. **External** - Targeted to multi-page websites, the below link tag is

<link rel="stylesheet" href="./style.css" />

style.css

html {

color: blue;

}

**CSS Selectors**

1. **Element Selector** - Targets elements based on their HTML tag elements

p {

color: blue;

}

1. **Class Selector** - Targets elements based on the value of the class attribute

.highlight {

background-color: yellow;

}

1. **ID Selector** - Targets elements based on the value of the id attribute

#header {

font-size: 24px;

}

1. **Attribute Selector** - Targets elements based on their attributes and values

input[type="text"] {

border: 1px solid black;

}

1. **Universal Selector** - Targets all elements

\* {

/\* properties here \*/

}

## Font Properties

1. **Font Size**:
   * 1px = 1/96th of an inch
   * 1pt = 1/72nd of an inch
   * 1em = 100% of parent (head element)
   * 1rem = 100% of root parent (html element)
2. **Font Weight**:
   * normal
   * bold
   * lighter
   * bolder
   * Any number (100 - 900)
3. **Font Family**:
   * General font family used is sans-serif or serif preceding with Helvetica
4. **Text Align**:
   * center
   * right
   * left

## Inspecting CSS Files in Browser

* **Command to directly open CSS code**: Ctrl + Shift + I
* Click on the three dots located at the right corner.
* Select **More tools** > **Developer Tools**
* Alternatively, right-click and select **Inspect**

## The CSS Box Model

## 

1. **Border**:

border: size style color;

border-left, border-right, border-top, border-bottom;

1. **Margin**:

margin: size;

1. **Padding**:

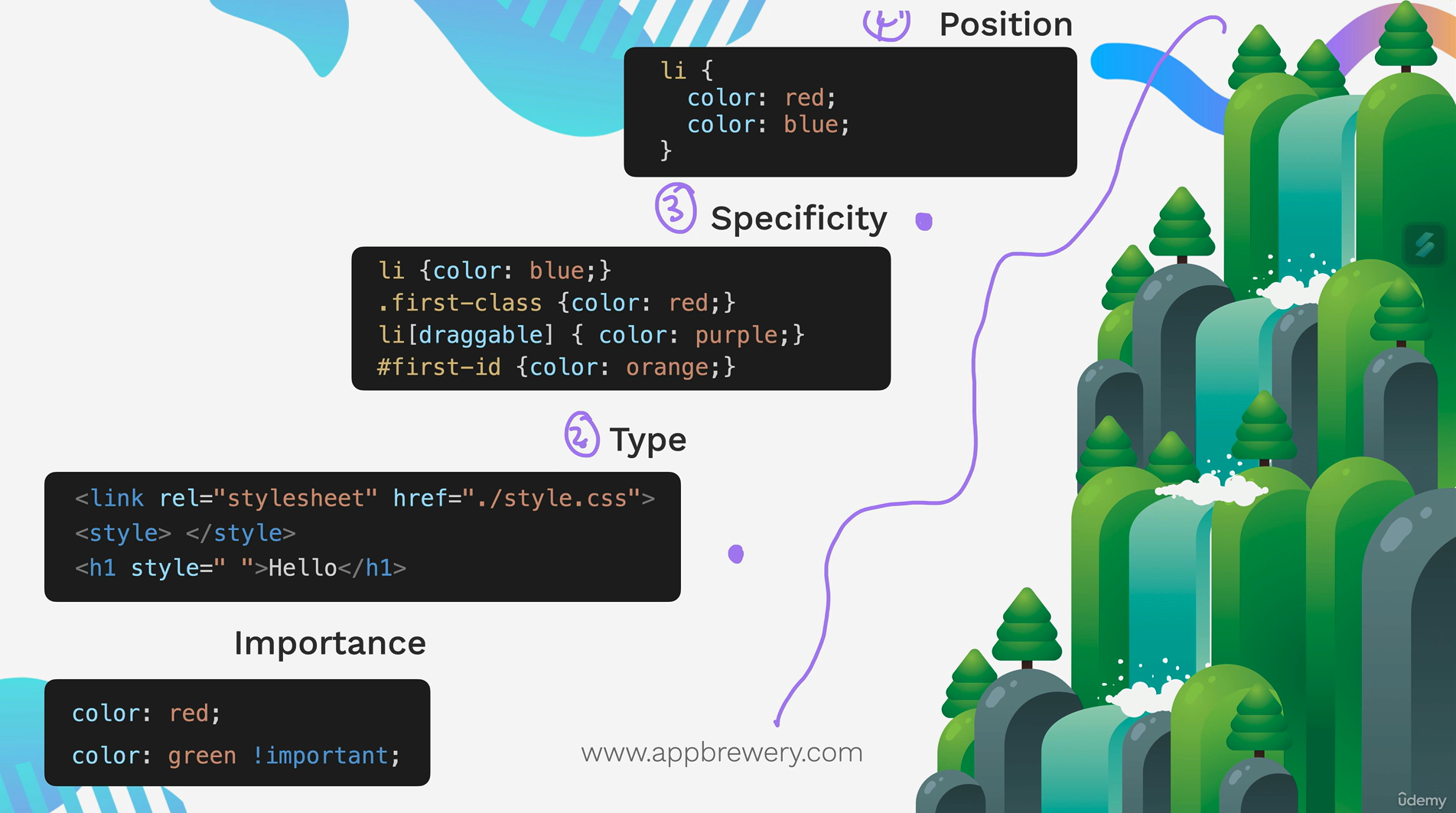
padding: size;

Note: We can also write like this for all box models

border: size-top size-right size-bottom size-left;

## CSS Cascade

The cascade determines which property is applied when multiple styles are assigned:



**Position**:

li {

color: blue;

color: red;🡨 Applied

}

**Specificity**:

Element Selector

Class Selector

Attribute Selector

ID Selector

**Type**:

* + - * 1. External
        2. Internal
        3. Inline

**Importance**:

color: red;

color: blue !important; This line is considered irrespective of other properties.

## Combine CSS Selectors

1. **Group**:

selector, selector; /\* Selects a group of classes, IDs, or element selectors at a time \*/

1. **Child**:

parent > child; /\* Targets direct children of a specified parent element \*/

1. **Descendant**:

parent descendant; /\* Targets descendants(regardless of their depth properties will be applied)within a parent element \*/

1. **Chaining**:

selector selector; /\* To select more accurate constraints \*/

1. **Combining Combiners**:

selector selector selector; /\* Hybrid selector \*/

## CSS Positioning

1. **Static**: Default position
2. **Relative**: Displaced from the static position
3. **Absolute**: Relative to the nearest positioned ancestor
4. **Fixed**: Relative to the top-left corner of the browser window
   * **Z-index**: Controls the depth of the containers

* **We generally use left, right, top and bottom properties to make positioning of the containers make sure to add position to some type example- *position: relative; .***

<https://appbrewery.github.io/css-positioning/>

## CSS Display

Three main values for the display property:

1. **Block**: Takes up the full line of the web page; height and width can be specified.
2. **Inline**: Takes up as much space as necessary based on the content.
3. **Inline-block**: Combines block and inline; does not start on a new line.

## CSS Float

* **Float**: Makes an element stick to the right or left side of the web page.

**Float**: left/right;

* **Clear**: Clears all the floated contents based on the specified side.

## Creating Responsive Websites

## To make web pages responsive the most common and effective techniques:

*Responsiveness*: A website which adapts to various screen sizes like (tablet, laptop, mobile phone) using:

1. Fluid Layouts (Percentage-based Widths)
2. Media Queries
3. CSS Grid
4. CSS Flexbox
5. External Frameworks (e.g., Bootstrap)
6. Use of *rem* and *em* units

## Media Queries (@media)

## 

Syntax:

@media (max-width: 600px) and (min-width: 200px) {

/\* Styles for screens will apply within the specified width \*/

}

* Multiple media queries can be used in one web page.
* object-fit: cover; ensures images or videos fully occupy a container, with excess parts cropped.

## Flexbox

* **Flex Display**: Use display: flex to arrange items in a row or column.
* **Flex Direction**: Default is row; set to column for top-to-bottom arrangement.
* **Flex Basis**: Defines the width (for rows) or height (for columns) of flex items.
* **Order**: Controls the position of flex items.
* **Flex Wrap**: Wraps flex items if they exceed container width.
* **Justify-content**: Aligns items along the main axis.
  + flex-start
  + center
  + flex-end
  + space-between
  + space-around
  + space-evenly
* **Align-items**: Aligns items along the cross axis.
  + flex-start
  + flex-end
  + baseline
  + center
  + stretch
* **Align-self**: Aligns individual items.
* **Align-content**: Applies to wrapped items.

### Flex Sizing

* **Priority List**: [content-width < width < flex-basis < (min-width/max-width)]
* **flex-grow**: Defines how an item grows relative to others.
* **flex-shrink**: Defines how an item shrinks relative to others.
* **Short Form**: flex: 1 1 0;

## Grid

* **Display Grid**: Use display: grid to create a grid layout.
* **Grid Template Columns/Rows**:

grid-template-columns: 1fr 1fr;

grid-template-rows: 1fr 1fr;

gap: 15px;

* **Fixed Sizes**:

grid-template-columns: 100px 200px;

grid-template-rows: 200px 200px;

* **Grid Placement**:

grid-column: span 2;

grid-row: span 2;

grid-column-start: 1;

grid-column-end: 4;

grid-row-start: 3;

grid-row-end: 4;

* **Negative Numbers**: Indexing from the right side of the grid.
* **Grid Area**:

grid-area: 3 / 1 / 4 / 2;

* **Exercises**:
  + [Grid Garden](https://appbrewery.github.io/gridgarden/)
  + Search for "Mondrian Compositions" for practice.

## Principles of Web Design

1. Color Theory
2. Typography
3. User Interface
4. User Experience