**CHAPTER 6: CSS THE VERY BASICS**

**What is CSS?**

CSS is a language for styling documents. CSS stands for cascading style sheets.

CSS Rules:

(almost) everything we do in CSS follows this basic pattern:

selector{

    property: value

}

Make all h1 purple

h1{

    color: purple;

}

Make all image elements 100 px wide and 200 px tall

img{

    width: 100px;

    height: 200px;

}

Select every other text input and give it a red border

input[type="text"]:nth-of-type(2n){

    border: 2px solid red;

}

**Including Styles Correctly:**

We can write our styles directly in html elements called inline styles, but this is not recommended because it is not possible to share styles between html elements.

<body>

    <h1 >Hello World</h1>

    <button style = "background-color: aqua; border: black; border-style: dotted;">I AM BUTTON</button>

</body>

We can also write our styles in <style> element but we cannot share these styles across documents. So this is also not recommended.

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>CSS Intro</title>

  <!--All h2 tags will be have color burlywood-->

  <style>

    h2 {

      color: burlywood;

    }

  </style>

</head>

We can write our styles in a .css file and include it in our html document using a link in the <head> of our html document. This is recommended as css files can be shared across documents.

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>CSS Intro</title>

  <!--This is how we connect css files to html files-->

  <link rel="stylesheet" href="index.css">

<!--rel means relationship-->

</head>

The **<link>** [HTML](https://developer.mozilla.org/en-US/docs/Web/HTML) element specifies relationships between the current document and an external resource. This element is most commonly used to link to [stylesheets](https://developer.mozilla.org/en-US/docs/Glossary/CSS), but is also used to establish site icons (both "favicon" style icons and icons for the home screen and apps on mobile devices) among other things.

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link>

href means hypertext reference, generally the path of the file we want to connect to

**Color and background-color properties**

The **color** property changes the color of the text. The **background-color** changes the color of the background of the html element. Note that the **background** attribute can also be used to change the background color but we can also change images and much more.

**Color Systems: RGB and named colors**

Color codes can be in the form of rgb, hexademical etc.

Eg: rgb(255, 0, 255) = #ff00ff in hexademical

**Common Text Properties**

1. text-align (<https://developer.mozilla.org/en-US/docs/Web/CSS/text-align>): The text-align CSS property sets the **horizontal alignment** of the inline-level content inside a block element or table-cell box.
2. font-weight (Defines the boldness of a font)
3. text-decoration (The text-decoration shorthand CSS property sets the appearance of decorative lines on text like underline, overline etc)
4. line-height (Used to specify the distance between lines of text)
5. letter-spacing (Spacing between letters of text)
6. Fonts, fonts size and font family

These are some of the important text properties

**Font sizes basics with Pixels**

**Relative units**: EM, REM, VH, VW, % and more (1 vh = 1 % of viewport height. 1 vw = 1 % of viewport width)

**Absolute units**: PX, PT, CM, IN, MM and more

(Read more about these)

(NOTE: 1px does not necessarily equal the width of exactly one pixel. Not recommended for responsive websites. Pixels are by far the most commonly used absolute unit. 1 px is roughly 1/96 inch)

**The Font family property**

(Syntax: font-family)

Note that many font families are not supported by all browsers. To know which font-family is supported by most computers use [cssfontstack.com](https://www.cssfontstack.com/).

    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;

What this means is that ‘Gill Sans’ is the first choice for the font-family, if ‘Gill Sans’ is not available, then ‘Gill Sans MT’ will be used. If that font-family is also not available, then Calibri will be used and so on...

**MODULE 7: THE WORLD OF CSS SELECTORS**

**Universal and Element Selectors:**

1: Universal Selector

\* {

    color: black;

}

\*  indicates select everything. Also known as universal selector. (NOT very commonly used)

2: Element Selector (These are commonly used) : select html elements.

h2 {

    color: rgb(27, 77, 4);

    background-color: rgb(84, 136, 184);

}

Everything inside the <h2> tag will have these property.

<h2>Happy birthday to the strongest/bodyguard <p>gunner</p></h2>

As the <p> tag is inside of <h2> tag, the content inside the <p> tag will also have the properties of <h2> tag mentioned in the css file.

We can also select multiple html elements

h1, h2 {

    color: magenta;

}

**The ID Selector**

ID Selector: Select the element with id of ‘login’. Notice the # before the id

#login {

    color: burlywood;

}

**The class selector**

Class selector (also used frequently)

Select elements with class of “complete”. (Notice the . symbol)

.complete {

    color: green;

}

id should be unique but classes need not be unique.

**The Descendant Selector** (Do not confuse with direct descendant)

(also very commonly used)

Select all <a>’s that are nested inside an <li>

li a {

    color: teal;

}

Note that No <li>’s will be styled. Only <a>’s which are nested inside <li> will be styled.

span li a {

    color: teal;

}

This means the <a>’s which are nested inside <li>’s which are nested inside <span>’s will be styled.

.post li a {

    color: teal;

}

This means the <a>’s which are nested inside <li>’s which are nested inside “post” class will be styled. **Note that here descendants need not be direct descendants.** These descendants can be nested way down.

We can have multiple cursor positions by holding the alt button.

Use nav tag when you have a list of links

**The Adjacent Selector:**

Select **only** the paragraghs that are **immediately** preceded by an <h1>. Note <h1> here is **not a parent element** of p. <h1> is **adjacent** to <p>

h1 + p {

    color: red

}

  <section>

    <h1>Popular Posts</h1>

    <p>Posted by <a href="u/not\_funny">/u/not\_funny</a></p>

  </section>

Here <p> is immediately preceded by <h1>. So <p> will have a color of red.

**Direct Descendant:**

Select only the <li>’s that are **direct children** (one level down) of a <div> element

div > li {

    color: white;

}

**Attribute Selector**

Select all input elements where the type attribute is set to “text”

input[type = "text"] {

    width: 300px;

    color: yellow;

}

Select every class “post” inside section

section[class = "post"] {

    background-color: purple;

}

<!—These two are same -->

section.post{

    background-color: purple;

}

<!—Notice that there is no space between “section” and “.post”. -->

**Pseudo Classes**

Keywords added to a selector that specifies a special state of the selected elements. We usually use Pseudo classes when we want an element change its appearance when a user does something to the element like hovering, enabling the radio buttons, making a button active etc.,)

:active

:checked

:first

:first-child

:hover

:not()

:nth-child()

:nth -of-type()  (There are many other pseudo classes. Read more from MDN Docs)

Notice that they all start with a **colon.**

a:hover {

    color: orange;

}

All anchor tags will have a text color of orange when hovered.

.post a:hover {

    color: orange;

}

All anchor tags inside the “post” class will have a text color of orange when hovered.

Checked is used with mainly radio buttons and checkboxes. (Change the style when checked)

**nth-of-type:** Select every odd element of class post.

.post:nth-of-type(2n + 1) {

    background-color: cyan;

}

(Read more of these pseudo classes from MDN)

Difference between nth-child() and nth-of-type(): <https://css-tricks.com/the-difference-between-nth-child-and-nth-of-type/>

(VIDEO 75)

**Pseudo elements**

Keyword added to a selector that lets us style a particular part of selected element

**::**after

::before

::first-letter

::first-line

::selection

Select first letter of each paragraph.

p::first-letter {

    color: red;

    font-size: 130%;

}

(Read more from MDN)

**The CSS CASCADE**

The order our styles are declared in and linked to matters!

h1 {

    color: red;

}

H1 {

    color: purple;

}

In this case there are conflicted styles. So h1 will be set to red first and then set to color purple because of “cascade”.

This order also matters when we have two or more CSS files linked to a html file

Lets say we have a html file which contains this

<head>

  <meta charset="UTF-8">

  <meta http-equiv="X-UA-Compatible" content="IE=edge">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>CSS Selectors</title>

  <link rel="stylesheet" href="index.css">

  <link rel="stylesheet" href="app.css">

</head>

Index.css contains

h1, h2 {

    color: magenta;

}

App.css contains

h1, h2 {

    color: purple;

}

As link to the app.css is below the index.css file, h1 and h2 will have text-color of purple. If the app.css link was above the index.css file, h1 and h2 would have color of magenta.

**Specificity**

Specificity is used to resolve conflicting styles.

Specificity is how the browser decides which rules to apply when multiple rules could apply to the same element. It is a measure of how specific a given selector is. The more specific selector “wins”.

**(SEE VIDEO 78 AND QUIZ 79)**

Formula for specificity: ID, {classes, attributes and pseudo classes}, elements and pseudo-elements

This is going to apply to only those properties in the selectors which have conflict. Other properties which have no conflict will be applied.

In case of tied specificity, the one which comes last wins.

**TIP: Chrome Dev Tools and CSS**

We can also see the styles that are being applied and striked off styles that are not being applied because of low specificity in the styles tab of chrome dev tools. (VIDEO 79)

**Inline Styles (Styles directly written in html elements) & Important**

They are bad practice and **should not be used**. Inline styles and important have higher specificity than ids.

Eg:

p::first-letter {

    color: red !important;

    font-size: 130%;

}

When we do this, the color of the first letter will be red as important overrides ids also. (NOT recommended but may be handy in some cases)

**CSS Inheritance**

body {

    color: blue;

}

When we do this, all elements which are nested in the body element will have text-color of blue because nested elements receive the properties of parent elements. This is called CSS Inheritance and we can see the inheritance information in the style tab of chrome dev tools.

But certain elements do not inherit by default.

Lets say we have a form which consists of input and a button

.form {

    color: green;

}

When we see this in the browser, the input and button text does not inherit the color green by default.

We can make this work using “inherit” keyword as shown below

Button, input{

    color: inherit;

}

Color will be inherited from the nearest parent’s color

**MODULE 8: CSS BOX MODEL**

**Box Model: Width and height**

In CSS, Box model is just an idea that everything is treated as a box which has border, padding, margin etc.,

A picture containing graphical user interface

Description automatically generated

1: **Width and Height:** By default, the width and height CSS property sets the width and height of the **content-area (content box) (**Because by default, box-sizing is set to “content-box”).

Diagram

Description automatically generated

(Read about box size: [https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing](https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing%20))

If we set box-sizing to **content-box**, the width and height will only be limited to content only, padding, borders are going to add up and increase the width and height of the whole box. And If we set box-sizing to **border-box**, the overall width and overall height of the whole box will be the width and height.

For content-box: width-of-content-box = width, height-of-content-box = height

For border-box: width-of-content-box + (2 \* padding + 2 \* border) = width, height-of-content-box + (2 \* padding + 2 \* border) = height

Note: In the above expressions, margin is not included.

**Box Model: Border and Border radius**

The main properties of border are border-width, border-color, border-style and border-radius

**Border-width**: Controls the thickness of the border.

**Border-color:** Controls the color of the border

**Border-style**: Controls the border style- dashed, solid etc

Tip: We can use border radius to make a circle, just set the border radius to half of width or height. We can make many different shapes using border radius.

**The display Property:**

1. **Inline:** Width and height are ignored. Margin and Padding are respected horizontally but not vertically.
2. **Block:** Block elements break the flow of a document. Width, Height, Margin, Padding are respected.
3. **Inline-Block:** Behaves like an inline element except the width, height, margin and padding are respected.

body has a default margin. So a lot of people first set the body margin to zero when they start any project (VIDEO 86)

body{

  margin: 0;

}

<https://stackoverflow.com/questions/10040464/adding-margin-causes-sister-elements-to-have-the-same-margin> (See this: setting margin top on an incline-block element causing other elements to have the same margin top)

**CSS Units:**

**Relative**: EM, REM, VH, VW, % AND MORE

**Absolute**: PX, PT, CM, IN, MM

1. **PX**: PX are the most commonly used **absolute** unit. 1px does not necessarily equal the width of exactly one pixel. Pixels are **NOT** recommended for responsive websites.
2. **Percentages**: Sometimes, it’s a value from the parent and other times it’s a value from the element itself. **Width: 50% ->** half the width of the parent. **Line-height: 50%** -> half the font-size of the element itself.  (VIDEO 88)
3. **Em**: With font-size, 1em equals the font-size of the parent. 2em’s is twice the font-size of the parent, etc. With other properties, 1em is equal to the computed font-size of the element itself. (VIDEO 89)
4. **Rem**: Relative to root **html** element’s font-size. If the root font-size is 20 px, 1 rem is always 20px, 2rem is always 40px etc. (VIDEO 90)

**MODULE 9: OTHER ASSORTED USEFUL CSS PROPERTIES**

**Opacity and alpha channel:**

alpha varies from 0 to 1, 0 being completely transparent and 1 being completely opaque. Syntax: rgba(200, 135, 100, 0.5).

one (opacity) is a property; the other is the component of the *value* of a color property, such as background-color, box-shadow-color, or border-color. Most importantly, opacity affects the *entire element it is applied to,* whereas rgba affects only *one aspect.* Opacity also varies from 0 to 1, 1 being completely opaque and 0 being completely transparent

<http://thenewcode.com/239/Whats-The-Difference-Between-opacity-and-rgba>

(VIDEO 92)

**Position: (Refer:** [**https://developer.mozilla.org/en-US/docs/Web/CSS/position**](https://developer.mozilla.org/en-US/docs/Web/CSS/position)**)**

The position CSS property sets how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of positioned elements.

1) Static: The element is positioned according to the normal flow of the document. The [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top), [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom), [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left), and [z-index](https://developer.mozilla.org/en-US/docs/Web/CSS/z-index) properties have *no effect*. This is the default value.

2) Relative: The element is positioned according to the normal flow of the document, and then offset *relative to itself* based on the values of top, right, bottom, and left. The offset does not affect the position of any other elements; thus, the space given for the element in the page layout is the same as if position were static.

3) Absolute: The element is removed from the normal document flow, and no space is created for the element in the page layout. It is positioned relative to its closest positioned ancestor, if any; otherwise, it is placed relative to the initial [containing block](https://developer.mozilla.org/en-US/docs/Web/CSS/Containing_block). Its final position is determined by the values of top, right, bottom, and left.

(Closest Positioned Ancestor is the closest ancestor whose position is not static)

(The containing block in which the root element ([<html>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/html)) resides is a rectangle called the **initial containing block)**

**4) Fixed:** The element is removed from the normal document flow, and no space is created for the element in the page layout. It is positioned relative to the initial [containing block](https://developer.mozilla.org/en-US/docs/Web/CSS/Containing_block) established by the [viewport](https://developer.mozilla.org/en-US/docs/Glossary/Viewport), except when one of its ancestors has a transform, perspective, or filter property set to something other than none (see the [CSS Transforms Spec](https://www.w3.org/TR/css-transforms-1/#propdef-transform)), in which case that ancestor behaves as the containing block.

The element will not disappear when the page is scrolled, It is fixed to that position.

Position Fixed is generally used for navigation bars.

5) **Sticky**: The element is positioned according to the normal flow of the document, and then offset relative to its nearest scrolling ancestor and containing block (nearest block-level ancestor), including table-related elements, based on the values of top, right, bottom, and left. The offset does not affect the position of any other elements.

(VIDEO 93)

**CSS Transitions:**

Used for animations

transition is the shorthand property

Syntax:

transition: property-name, duration of the transition, transition function, delay for transition.

<https://developer.mozilla.org/en-US/docs/Web/CSS/transition>

(VIDEO 94)

**CSS Transforms:**

Transforms are used to rotate, scale, skew or translate an element

**NOTE: for block elements, when we set margin-left and margin-right to be auto, the elements will get centred in their containing blocks. This does not work for margin-top and margin-bottom.**

**margin: auto works for horizontal margins only and it also won’t work with floated and inline elements and by itself, it also cannot work in absolute and fixed positioned elements**

Note: Use transforms to translate an element instead of position property because transforms are better performance wise. (Not clear of the reason)

[**https://developer.mozilla.org/en-US/docs/Web/CSS/transform**](https://developer.mozilla.org/en-US/docs/Web/CSS/transform)

**(VIDEO 95)**

**More about Background Property**

<https://developer.mozilla.org/en-US/docs/Web/CSS/background>

(VIDEO 97)

**Google fonts is amazing (VIDEO 98)**

https://fonts.google.com/

We can use calc() function to make calculations in css

img{

    width: 40%;

    margin: calc(20%/4);

}

(VIDEO 99)

**NOTE:**

<img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSnBHkofUeLUgqhZzq3SThD304LAOSAbgakNA&usqp=CAU" alt="">

<img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSdHlbf7qg6y6JSs0AGD5O\_WW7U327mTr7vGA&usqp=CAU" alt="">

<img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTe95-bGY1zliejdM0FssE9jzcTpr\_zMSZjiw&usqp=CAU" alt="">

 is different from

<img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSnBHkofUeLUgqhZzq3SThD304LAOSAbgakNA&usqp=CAU" alt=""><img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSdHlbf7qg6y6JSs0AGD5O\_WW7U327mTr7vGA&usqp=CAU" alt=""><img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTe95-bGY1zliejdM0FssE9jzcTpr\_zMSZjiw&usqp=CAU" alt="">

in the first snippet, there is a whitespace character between images and you will get a white space between images. This is a well known problem for inline elements in HTML. So for Inline elements, be careful of the spaces in HTML.

(VIDEO 100)

**MODULE 10: FLEXBOX**

FlexBox is a one dimensional layout for laying out items in a row or column.

It is a recent addition to CSS.

Flexbox allows us to distribute space dynamically across elements of unknown size. hence the term “flex”.

display: flex;

When we set display to flex, then the flex properties will apply to the **direct children** of the container (Flex Container).

In flexbox, there are two axis: main axis and cross axis. The main axis direction is defined by flex-direction and cross axis is perpendicular to the main axis.

flex-direction: row;

flex-direction: row-reverse;

flex-direction: column;

flex-direction: column-reverse;

In flexbox, the contents will not overflow, even if the dimensions are greater than the container block, the contents’ dimensions would be adjusted to fit inside the container.

(VIDEO 103)

**Justify-Content**: The justify-content property aligns the items along the main-axis. These are the mostly used alignments

justify-content: flex-start;

justify-content: flex-end;

justify-content: center;

justify-content: space-around;

justify-content: space-between;

justify-content: space-evenly;

(VIDEO 104)

**Flex-wrap:**

In flex-box, when the items are too large to be displayed in a single line, the items shrink so as to fit in the flex-container. To avoid this we use flex-wrap property so that the items do not shrink and are displayed in the next line(s).

flex-wrap: nowrap; /\*Default value\*/

flex-wrap: wrap;

flex-wrap: wrap-reverse; /\*Cross axis direction becomes opposite to the earlier direction \*/

(VIDEO 105)

Justify-content: aligns the items along the main-axis;

Align-items: aligns the items along the cross-axis;

/\* Basic keywords \*/

align-items: normal;

align-items: stretch;

/\* Positional alignment \*/

/\* align-items does not take left and right values \*/

align-items: center; /\* Pack items around the center \*/

align-items: start; /\* Pack items from the start \*/

align-items: end; /\* Pack items from the end \*/

align-items: flex-start; /\* Pack flex items from the start \*/

align-items: flex-end; /\* Pack flex items from the end \*/

/\* Baseline alignment \*/

align-items: baseline;

align-items: first baseline;

align-items: last baseline; /\* Overflow alignment (for positional alignment only) \*/

align-items: safe center;

align-items: unsafe center;

/\* Global values \*/

align-items: inherit;

align-items: initial;

align-items: revert;

align-items: revert-layer;

align-items: unset;

**align-content and align-self:**

Refer: <https://stackoverflow.com/questions/27539262/whats-the-difference-between-align-content-and-align-items>

The align-items property of flexbox aligns the items inside a flex container along the cross axis just like justify-content does along the main axis. (For the default flex-direction: row the cross axis corresponds to vertical, and the main axis corresponds to horizontal. With flex-direction: column those two are interchanged respectively).

But align-content is for **multi-line flexible boxes**. It has no effect when items are in a single line. It aligns the whole structure according to its value. Here's an example for align-content: space-around;

**Flex-basis, Flex-grow, Flex-shrink**

The flex-basis CSS property sets the initial main size of a flex item. (Flex-basis sets the size of the body in main axis direction. So use flex-basis instead of width property if you plan on changing the flex-direction)

Flex-grow: Controls the amount of available space an element should take up. Accepts a unit-less number value.

Flex-shrink: If items are larger than the container, they shrink according to flex-shrink.

**Media Queries:**

Media Queries are used to build responsive styles.

/\* When viewport width is <= 280px, container width should be 400px \*/

@media (max-width: 280px){

    #container{

        width: 400px;

    }

}

@media (min-width: 1200px){

    #container{

        flex-direction: row;

    }

    .pricing-panel{

        border-bottom: none;

        border-right: 2px solid black;

    }

    .pricing-panel:last-child{

        border-right: none;

    }

}

**CSS Reset**

The goal of a reset stylesheet is to reduce browser inconsistencies in things like default line heights, margins and font sizes of headings, and so on. When starting with styling, use the below stylesheet to reset default css values.

(refer this: <https://meyerweb.com/eric/tools/css/reset/)>

**Bootstrap:**

Refer: <https://getbootstrap.com/>