1. **CSS Types**

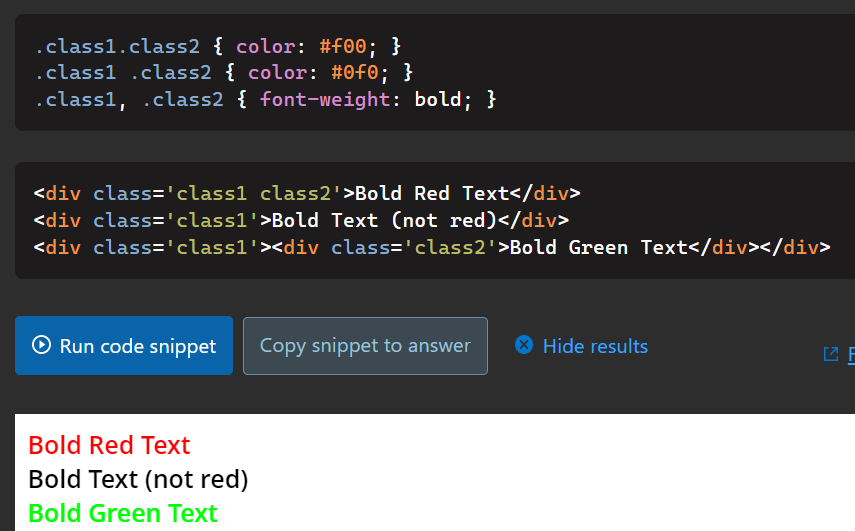
There are three kinds of CSS ways.

**Inline** – when you write within the HTML tag itself. It is a very bad practise and not recommended.  
**Internal** – When you create a Style section and write within the HTML file but separately from the HTML code.  
**External** – The one which is used at most places and is the professional way.

Hierarchy of CSS: Inline>Internal>External. This means any style specified with Internal CSS will override Internal and External style.

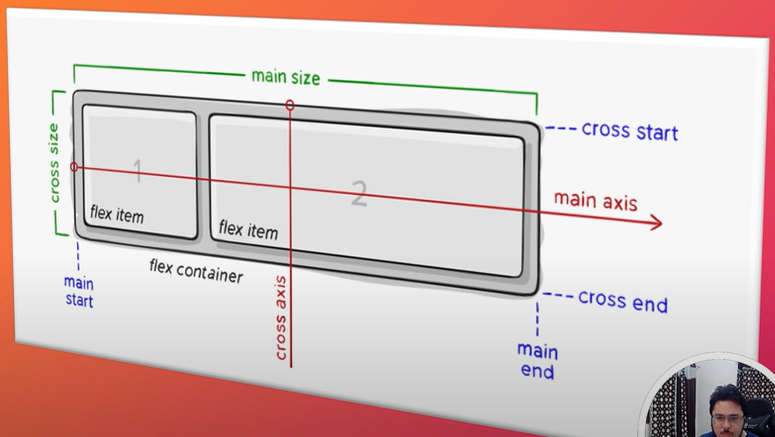
1. **CSS Basic Syntax**
2. [**Selector**](https://webplatform.github.io/docs/css/selectors/) **{property: value;}**
3. Selector tells what tag or class or id to select. Selectors are of three types: **tags, classes and ids**.   
   Property is the parameter you want to change. Value is the specific we want to give to the property.
4. [**CSS Combinators**](https://www.freecodecamp.org/news/css-combinators-to-select-elements/#:~:text=The%20descendant%20combinator%20matches%20a,direct%20children%20of%20another%20element.)**: or** [**this**](https://www.w3schools.com/css/css_combinators.asp) **or** [**(video)**](https://youtu.be/ZKRRUUPl8SA)

|  |  |  |
| --- | --- | --- |
| SN | Combinator | Function |
| 1 | Comma | **Comma** groups the classes (applies the same style to them all). It is about applying the same style to multiple selectors at once. |
| 2 | Empty space **descendant Combinator** | It tells that the following selector must be within the first selector and it is applied to the **right-most one.** It will select any tag/class inside the one specified to it’s left. |
| 3 | Dot combinator(Both or AND) | NO GAP when we apply a style to an element which has both the classes. |
| 4. | **Child Combinator** | It will apply the style to a direct child only, not if the right most is a grand children of the tag to it’s left. MUST be a direct children. |
| 5. | General sibling selector | The sibling combinator used between selectors matches elements that are siblings of another element, that is, the elements mentioned which are at the same hierarchical level and appear after the one mentioned at left. |
| 6. | Adjacent Sibling selector | The adjacent combinator matches only the IMMEDIATE sibling that comes after an element. |
|  |  |  |

  
  
So here we have 2 classes: Class 1 and class 2.  
1. **The first style statement is the**  CASE 3 here. So it applies to div 1 which has BOTH CLASS1 AND CLASS2. Just like **and** operator if a.b true then only it works.  
2. **The second style statement is** applied to the right most class ie class 2 which is inside all other classes mentioned at the left of this. HEIRARCHIAL/**NESTING** of classes. Hence we see the **Third div** being changed to green.  
3. **The third style statement is** applied to **all the divs which have either class one or class 2 or both obv.**

1. A class name can be used by multiple HTML elements, while an id name must only be used by one HTML element within the page.
2. [Inheritance](http://web.simmons.edu/~grabiner/comm244/weekfour/css-concepts.html)
3. **The box model of Website styling** tells us that we can treat each element as a rectangular box, which takes up some space on the screen and we can apply all the properties to this box. This CSS box wraps around the HTML elements to provide style. **It has – content edge, padding edge, border edge and the margin edge.** It is the box shown when we do inspect element in a browser.
4. CSS display property has two things – Flex and grid. You must learn both. (Leaving it’s notes for a while.)
5. **Positioning**
   1. **Static:** By default, position an element based on its current position in the flow. The top, right, bottom, left and z-index properties do not apply.
   2. **Relative:** We change the position of the element with respect to the position it is supposed to have if not explicitly set without changing layout.
   3. **Absolute:** Position an element based on its closest positioned ancestor position ie with respect to the parent.
6. **Centering elements:** The **two** mostly used methods are – either provide margin as auto from the two ends(move in clockwise from the top.) or use text-align:center; to center the elements. Other ways include Grids, Flexbox, Bootstrap etc.
7. **Google Fonts API:** We can use [Google fonts](https://fonts.google.com/) to resource the fonts which might not be present at the user end.
8. [**CSS Sizing**](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Sizing_items_in_CSS)**:** We can use [different units](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units) to size our text or even images. Some of them are – em, rem, px, (vh/vw – viewport height and width) and percentages.
9. [**Clear and Float**](https://www.w3schools.com/css/css_float.asp)**:** It is like using text wrap in html. Float decides what happens to the images, instead of being inline elements they go to side where we ask it to float, and then we can use clear property to adjust the text flow around it. Use the link to read more. I am writing here just to mention and recall that something like this is also there, incase you run into a problem and refer back here.

**Flexbox: Axes/direction and align items or justify:**

****

The main axis is defined by flex direction – row, column and row-reverse, column-reverse. Row is for horizontal or inline direction placement of objects, columns do it in vertical direction or the block direction.

**Flex-direction: column;**

It means to make vertical the primary axes, and horizontal axes as secondary axes and vice versa for row.

The following works along the inline axes of the container.

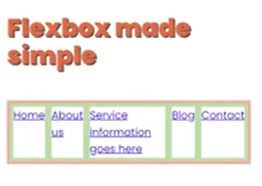
**Justify-content: start || center || space-around || space-evenly || space=between**

**Align-items: center || stretch || start || end 🡨** These can work along the column direction as well. Flex-direction can be column. We do not need to set width or height explicitly.

**Philosophy of flexbox- how to think like flexbox([Kevin Powell](https://youtu.be/u044iM9xsWU)):**

***Flex items behave in a different way, they want to be as small as it can possibly get to without wrap unless necessary while maintaining everything in one line, in accordance to the properties.***

* Initially the li-s are block items which stretch to fill entire row, but when we turn on display:flex, they turn into FLEX-ITEMS.
* So now they are going to behave like flex items, and display: block for the li-s of the ul or ol is not going to work.

* If we set for li – width:max-content; without display:flex which means the flex items are normal elements now, it will show the li as block elements but their personal boxes would be small.   
  So what happens with display:flex? It makes them go in one line with max-content thing to ensure small size without wrap.
* And now if it runs out of room if we squish the website, it tries to wrap around as the min-content as the smallest possible if there is not enough room in the parent.  
    
     
  -------------------------------------------
* Flex shrink prevents overflow outside parent container. Flex grow allows to utilise entire space within container in some ratio as specified for each element. By default it is 1.
* Flex-wrap: wrap; this makes them flow to other rows with max content initially but if we go even further squishing, it shrinks same content to multiple rows.
* As long as it can be within one line without wrap.
* **Flex:1 -> It makes flex-grow, flex-shrink as 1 (default) and flex-basis as 0(changed).  
  *The flex-grow* makes each of the boxes evenly distribute the available container space.  
  *The flex shrink* makes them break into lines if webpage is squished too much or a different device is used.  
  *Flex-basis* is very similar to width of item ie flex-basis as 100% makes each element want to be 100% their parent size,and it overflows the container by 200% without flex-shrink turned on.**
* Flex basis is dealt more with main and cross axes topics we will see that later.
* [**BEM Naming convention**](https://youtu.be/xaXmoVZ3koo)
* [**Custom Properties in CSS**](https://youtu.be/PHO6TBq_auI)