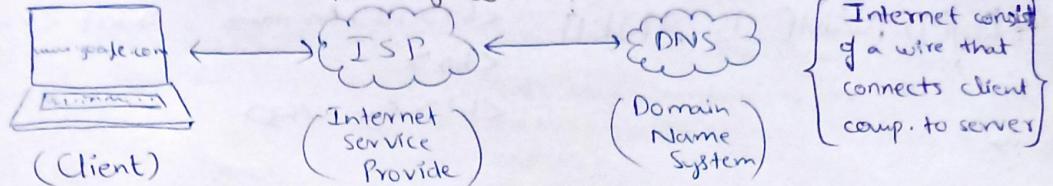


S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
		<u>Web - Dev</u>		
1)		Front-End Web Development		
2)		Introduction to HTML		
3)		Intermediate HTML		
4)		Introduction to CSS		
5)		Intermediate CSS { 2023 - Updated }		
1)		Introduction to HTML		
2)		Intermediate HTML		
3)		Multi-Page Websites		

FRONT-END WEB DEVELOPMENT

- How does the Internet Actually work?



When we type → `google.com` , then BTS (behind the scene) , my browser will send that message to your ISP .

(there are the ppl. who you pay to be able to access the internet)

ISP then relay that message to DNS server , then DNS server looks in its database to find the exact IP address of that website that you're trying to access.

- Every single comp. has IP address (kinda like postal code)

Once the DNS server finds the IP address , it will send it back to your browser through the ISP.

- How do websites work?

Now data that we receive from server usually consists of 3 types of files : HTML, CSS, JavaScript

- ex: If a website was a House:

HTML would be the actual bricks of the house . Similarly , the HTML file contains the content of your website like the text content or the images or buttons or links

CSS would help in shape of door or colour of walls . The CSS file would determine how my website will look , like background of page & shape of button , helps in styling the font

JavaScript would be like adding light bulbs that can turn on & off . It allows my website to ~~actually~~ do things & become functional

INTRODUCTION TO HTML (PRACTICE 1)

HTML = Hyper Text Markup Language

- Using HTML Tags

Go to → codepen.io → Start Coding → In HTML

`<h1> Hi </h1>` → H1

`<h2> Hi </h2>` → H2

`<h3> Hi </h3>` → H3

Suppose if we want to add a line break → then write → `
`

Cells that - h1 code ends after H1

{ < > }

↑
Self-closing tag

In website

HI | MY NAME IS ANKIT

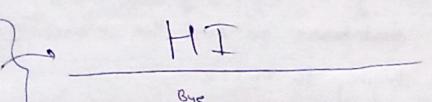
Bye

To refer more → devdocs.io

The Anatomy of an HTML Tag

Now if we want to add horizontal lines in our website, we use → `<hr>` (horizontal rule)

e.g. `<h1> HI </h1>`
`<hr>`
`<h3> Bye </h3>`



Now: there are diff. attributes for 'hr' → e.g. align, color, noshade, size;
 `<>` = (opening tag) width

e.g. `<hr size="3" noshade>`
 + space
 HTML attribute
 element

+ `<center>` will change whole text to centre
 `</center>`
 closing tag = `</>`

→ Comments: `<!-- --!>`

(Ignored by browser)

What is the HTML Boilerplate?

In VS code → `_index.html` ← (how to create a file for HTML)

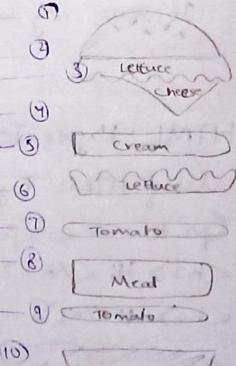
↳ Shortcut → !

↳ we will get: embed abbreviation ✓

THE HTML BOILERPLATE

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title> Website name </title>
  </head>
  <body>
    <h1> Hello World! </h1>
  </body>
</html>
```

HAMBURGER



①: `<!DOCTYPE html>` : tells the browser that we have written our code in HTML 5 (latest version)

②, ⑩: `<html lang="en">` ↗ this is the root of the document
 ↗ this attribute says: language of the text content in the element (like for blind people, they have screen readers)
 ↗ Inside `<html>` only, all the stuff will be written

③, ⑥: `<head>` : this is an area where imp. info about our website is placed that is not going to be displayed to the user.

④: `<meta charset="UTF-8">` ↗ ensures that chars. that you're using (default) on your website gets displayed correctly (e.g., +, -, emojis can be shown as per our meta charset)
 ↗ This is convention, so let it be like that

⑤: `<title> My Website </title>` : this title is displayed at top in the Tab bar

⑦, ⑨: `<body>` ↗ this is body element, where we will create & write our website

↳ this is where all of the content of website goes {texts, titles, images, links, ... etc.}

⑧: content

In VS code: ! → !

(just keeps our code compatible with Internet Explorer)
we can delete this line as we don't use IE anymore

`<meta charset="UTF-8">`
`<meta http-equiv="X-UA-Compatible" content="IE=edge">`
`<meta name="viewport" content="width=device-width, initial-scale=1.0">`

↳ tells how website should be displayed relative to the screen that it's being rendered on.

* We add attributes by giving it a name, and then after an equal sign, we give it a value:

How to Structure Text in HTML?

→ Right click on our file in VS Code → Copy path → Paste in google to (See our website)

Click on bottom → Go Live

* `<p>` ↗ paragraph tag
 `</p>`

* Writing in italics → <i> & </i>
 (only italics) &

 (italics + emphasis there)
 ↑
 (better to use)

* → bold writing

ex: <body>
 <h1> Hi </h1>
 <p> My name is Ankit Verma </p>

• HTML Lists

* To write in bullet points →

 Hi

 { Hi
 ↗ (list items)

* To write in no.: →

 —

→ If we want 'ol' to start at particular no. ⇒ <ol start="7">
 —

If using roman numerals → <ol type="i">
 —

• HTML Image Elements

HTML attribute

 ↗ if from browser ↓ if from folder
 (HTML element)

In browser
 (right click on image)
 (to get its address)
 will appear if image unable to load
 (alternative text)

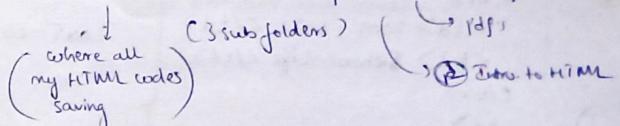
If we want to resize image →

① If interested in Search Engine Optimization & getting your website ranking for certain key words, this is something that Google looks at to try & figure out what your webpage is about

* Now to remove our reliance on web, we can also put image into our folder (where HTML documents are saved)

* If we want to crop our image in circle → crop-circle.imageonline.co

ex: My folder → Coding > Web-dev > Front End Web Development → images



In VS code, to render image, {let image name = ankit.jpg}

• HTML Links & Anchor Tags

This is the part where we learn 'HT' (hypertext) of HTML.

(HTML attribute) → (hyperlink reference)
 Hello → anchor tag
 (HTML element) This is the website which our browser will take if we click on 'Hello'
 (Link text)

* Shortcut → type 'a' in VS code, we will get!

* If we want to open more than 1 link, then press 'ctrl' button on keyboard & click on links.

* Now if we want a hyperlink which opens our part of website, then → ① Open a new file in your folder where you're saving all your html links.

② ex: → Let file name = contact.html

③ Then: Contact details

If this is inside a folder called 'Hi', then write as: "Hi/contact.html"

INTERMEDIATE HTML

HTML Tables

ex: <h3> Work Experience </h3>
 <p> 2005 - 2022 School Life </p>
 <p> 2023 College Life </p>

* Now to make it look more nice, we can create it as table!

(type - t)

```
<table>
  <tr>
    <td> 2005 - 2022 </td>
    <td> School Life </td>
  </tr>
  <tr>
    <td> 2023 </td>
    <td> College Life </td>
  </tr>
</table>
```

Work Experience

2005 - 2022	School Life
2023	College Life

* In tables we also have headers, body & footers

(by default its bold)

```
<table>
  <thead>
    <tr>
      <th> Dates </th>
      <th> Work </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td> 2005-2022 </td>
      <td> School Life </td>
    </tr>
    <tr>
      <td> 2023 </td>
      <td> College Life </td>
    </tr>
    <tr>
      <td> Statistics </td>
      <td> Now </td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td> Statistics </td>
      <td> Now </td>
    </tr>
  </tfoot>
</table>
```

Dates Work

2005-2022	School Life
2023	College Life
Statistics	Now

Now if we want borders of table also, then:

<table border="1"> (This '1' is in pixels)
 </table>

But table with this kind of border, it doesn't look good so much. Just manageable!!

Using HTML Tables for Layout

Now if we want to show our info. on right side of image, then we can use tables.

<body> in pixels

```
<table cellspacing="20">
  <tr>
    <td></td>
    <td> Ankit Verma </td>
  </tr>
</table>
```



* Win + . (a) Win + ; → to get emojis in Windows
 ↑ ↑
 (fullstop) (semicolon)

HTML Tables Code Challenge

Q) Write code → so output : Skills

A) <h3> Skills </h3>

```
<table border="1">
  <tr>
    <td>
      <table cellspacing="10">
        <tr>
          <td>iOS Development </td>
          <td> ★★★★ </td>
        </tr>
        <tr>
          <td> Web Development </td>
          <td> ★★★★ </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

iOS Development ★★★★	Photography ★
Web Development ★★★★	Painting ★

(We use nested table concept here)

```
<td>
  <table cellspacing="10">
    <tr>
      <td> Photography </td>
      <td> ★★ </td>
    </tr>
    <tr>
      <td> Painting </td>
      <td> ★ </td>
    </tr>
  </table>
</td>
</tr>
</table>
```

• HTML Forms

* We can create forms

• <form>

```

<label> Name: </label>
<input type = "text" />
<label> Password: </label>
<input type = "password" />
<input type = "submit" />
</form>

```

+ There are many kinds of input form in VS code
 e.g. date, range, email, colorpicker, etc

• Forms in Practice - Create a Contact Me Form

(Go to → contact.html)

<body>

① - <h1> My contact details </h1>

② - <p> My Fictional Address </p>

③ - <p> myemail@gmail.com </p>

④ -

⑤ - <form action = "mailto:(your@gmail)" method = "post" enctype = "text/plain">

⑥ - <label> Your Name: </label>

⑦ - <input type = "text" name = "My Name:" id = "" >

⑧ - <label> Your Email: </label>

⑨ - <input type = "email" name = "My email" id = "" >

⑩ - <label> Your Message: </label>

⑪ - <textarea name = "My message" rows = "10" cols = "30" > </textarea>

⑫ -

⑬ - <input type = "submit" name = "" >

⑭ - </form>

</body>

{meanings}

(so that O/P shows no symbol, only plain text)

(means it posts user input to you)

(performs action)

⑯, ⑰, ⑱: In btw 'label', we should write — what we want from user
 ⑲, ⑳: <input type = "text" name = "My name" id = "" >

 "My name"
 "My email"
 ↓
 (in what format we want user to type)
 ↓
 (shown in O/P)

㉑: <textarea name = "My message" rows = "10" cols = "30" > </textarea>
 creates a box like str.
 ↓
 (shown in O/P)
 border shown
 inner lines are not shown

㉒ <input type = "submit" name = "" >
 ↓
 (shows submit button)

+ After clicking on Submit, "action" starts.

e.g. (After Go Live)

My contact details

My fictional Address

myemail@gmail.com

Your Name: AV

Your Email: ankit@gmail.com

YOI

Your Message:

(we can resize)

Submit

{After clicking submit, my "Mail" opens up}

O/P: My Name = AV
 My Email = ankit@gmail.com
 My message = YOI

• Publish Your Website!

* We use GitHub!

Steps: 1) Click on '+' (near your prof. pic) → new repository

2) Repository name = Project name (write it)

• Public

• Add a ReadMe File → (Create repository)

* Now click on Add file → upload files (which are used in your code)

• Commit changes → write related to code/proj., as it will show our note (on outside)

4) Settings → Pages → Source → choose: (main) → Save

5) Our website is now published & we can access it from anywhere
ex: <https://ankit8125.github.io/Angela-Yu-CV/>

Website link
Skeleton

6) If we see a 404 error, then try after 30 min as Github servers are busy.

* Now, our main file (which is our homepage) is actually called "index.html", and it should have a lowercase i & it should be spelled exactly like this!

This is very imp. because Github is going to look for a file with this name in order to serve as your homepage

7) Now if we see <DOCTYPE html> written on top of our website, then we can edit in index.html & delete that.

INTRODUCTION TO CSS

(Cascading Style Sheets)

• Inline CSS

An example of website using CSS: <https://web.archive.org/web/20180819202235/https://seanhalpin.io/>

Now if we want to colour our website, then:

<body style="background-color: ____;">
 (attribute) (CSS code)
 we can type colour name
 We can write hexadecimal code of that colour.
 (type: colors (css mdn) {&} get) (go to: colorhunt.co)

* Types of background color ways:

⇒ writing in hexa dec. ways / color name.

⇒ rgba color model: r → red (255 corresponds to 100%)
 denoted: rgba(, , ,)
 g → green
 b → blue
 (Optional) a → alpha → no. b/w 0 & 1 → (varies transparency)
 1 = opaque

ex: <body style="background-color: rgba(120, 75, 75, 0.5);">

⇒ hsla model: h → hue → taking an angle of color circle (if no. written - then its in degree)
 s → saturation (0% → completely saturated, 100% → grey)
 l → light → (100% - white, 0% - black)
 a → alpha

→ background-color: current color; } changes whole website to black
, transparent; } changes whole website to transparent

Internal CSS

Now if we have to change colour of line, then we have to write background color -- at every line where <hr> is present -- lengthy process.

So instead of that, we write → <head> :

In btw style tag, we have to specify name of element that we have to change & then we create a set of curly braces inside which we are going to write CSS code.

<style> } the part which we wanna change in whole website at a time
</style>
</head>

Now basically a line isn't a line after all,

Line = ——, rectangle with zero height

(changes inside space as white)

Now browser has some default CSS coding

ex: <style>
 body{
 background-color: blue;
 }
 hr{
 background-color: white;
 }
</style>

for ex, (type → css default for browser) → in that → for <hr>: display: block;
 margin-top: 0.5em;
 margin-bottom: 0.5em;
 margin-left: auto;
 margin-right: auto
 border-style: inset
 border-width: 1px
(pixels)

Now there are diff. border styles.

border-top-style: none; → (no border)

hidden;

dotted:

Solid:

double:

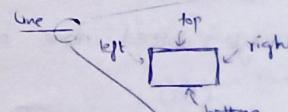
groove:

ridge:

inset:

outset:

dashed:



* If border-style: none;
(then no borders will be)

* Everything that exist on webpage are essentially just boxes. If we can make those boxes appear → by using "pesticide" extension of chrome.

* We can also resize image → <style>

```
img {  
    height: 200px;  
}
```

image will get resized.

* In <hr>:

```
<style>  
hr {  
    background-color: blue;  
    border-style: none;  
    height: 2px;  
    width: 100px;  
}
```

(we can also write `width: 30%;`; which means it will make 30% of line based on our webpage size, so if we minimize our webpage, our line will also get adjusted according to it.)

Q) Convert the line: ————— to

A) If → <hr { background-color: white; border-style: dotted; height: 0px; width: 30%; }>

Now → border-style property:

(i) 1 value specified, it applies same style to all four sides

(ii) 2 values specified, 1st one applies to top & bottom

2nd one applies to left & right

(iii) 3 values specified, 1st → top, 2nd → left & right, 3rd → bottom

(iv) 4 values specified → top, right, bottom, left → in that order

It's applied

(clockwise)

```
hr {  
    border-style: dotted none none;  
}
```

(OR)

```
hr {  
    border-style: none;  
    border-top-style: dotted;  
}
```

—————,

So answer → hr {

```
border-style: none;  
border-top-style: dotted;  
border-color: grey;  
border-width: 2px;  
width: 5%;
```

(grey)
(color)

}

- External CSS

Now suppose we want our 'hobbies' & 'contact me' page also to look like main page, so it would little length if we copy-pasted 'style' code in each file. So instead of that we create a new folder → CSS → inside, we create new file → styles.css

Now we cut-paste from index to styles.css

(cut inside of <style> → can delete)
</style>

And now wherever we want to apply; be it - index, hobbies, contact

just write this piece of code in <head>

```
<link rel="stylesheet" href="css/styles.css">  
</head>
```

* To make the text colourfull;

then inside styles.css → h1 { color: blue } } → changes all <h1> to blue colour

- How to Debug CSS Code

* To check our code in chrome → right click → inspect

→ more tools → developer tools

Now, one case of debugging is;

```
<head>  
    <link rel="stylesheet" href="css/styles.css">  
</style>  
    body { background-color: red; }  
</style>  
<head>  
    <body style="background-color: white;">  
        </body>
```

(let color pointing here = blue)

Now our website will show us white background. So to know what's happening → Click on inspect

→ In elements: select body → Now at bottom, see styles

It's showing all of the CSS that is being applied to body of our webpage

→ Now the chrome has default priority to arrange in this way.

(First 'body'; then in 'style'; then 'styles.css')
↑ ↑ ↑
(inline CSS) > (internal CSS) > (external CSS)

→ we can untick the color that we don't want to show up on our webpage

This unticking & ticking on chrome, affects how the webpage gets displayed. {Our code will not change}

So if we hit 'refresh', it goes back to how it will be displayed across all the browsers. So change is done locally.

* This means, we can apply a global CSS rule to all of our webpages, but on the individual web pages, we can apply more specific rules using internal (or) inline CSS as more or less one-off changes for that specific page or that specific element on that page

The Anatomy of CSS Syntax

Selector { property : value } ;
~~~~~|~~~~~|~~~~~|~~~~~  
(who do you want to change) (what you want to change) (how much we want to change)  
ex: h1 {color: red};

\* Now suppose in styles.css → h1 {color red};  
then vs code will show 'problems' notification at bottom, so that we can resolve it.

\* h1 {  
color: red;  
font-size: 200px;  
}  
Now the best practice is to write all the properties in alphabetical order, so that it would be easy for us to solve the debugging

\* You can refer to CSS MDN references to know all about the keywords (effectively our property)

## CSS Selectors

To comment out in CSS : /\*

\*/

→ Go to → emojijs.org → for emojis

Now suppose we want to give 2 diff. backgrounds to 2 diff. images, then:

In index.html : ``  
``

Now in styles.css :

/\* (TAG SELECTORS) \*/  
img {  
background-color: red;  
}

/\* (CLASS SELECTORS) \*/  
Hi {  
background-color: blue;  
}  
Bye {  
background-color: green;  
}

← (add a full stop & write class name)

→ So now : class selector would be given first preference to img. background & if nothing is mentioned in img src, then the tag selector would be given preference

## Classes vs Ids

(under) ex: `<h1 id="heading"> Yo </h1>`

(Now in styles)

(refers to tent colour)

#heading {  
color: blue;  
}  
/\* (ID SELECTORS) \*/  
→ This changes "Yo" to blue colour & as usual ID selectors are given more preference than tag selectors

∴ Preference order = ID > Class > Tag > Chrome default style

\* Now id="heading" can be used only once in that html page whereas with class, it doesn't really matter. So if we were to create another id, then it should have some diff. name other than "heading", whereas class can have same name.

\* Use classes when you want to apply same style to group of related items & use the id to apply a specific style to a single element on your web page.

\* Any html element can have more than one class but it can only have 1 id.

ex: (index): <body>  
    |  
      
    </body>

(styles.css):  
.Bye {  
    background-color: green;  
}  
.circular {  
    border-radius: 100%;  
}

} Then our image will become circular & have its background color as green.

But if we try to do the same with id's,

(index): <body>  
    <h1 id="heading big"> Yo </h1>

(styles.css): #heading {  
    color: blue;  
}  
# big {  
    font-size: 1000px;  
}

} Now our webpage will neither be blue in color nor font size will be 1000px.

} Since the format is incorrect it doesn't read this code. & then directly reads tag selector to check if any property is mentioned or not & makes changes accordingly

→ In CSS reference mdn → we can see few keywords having a little colon in front of them & these are called: pseudo classes

(ex: let us look at :hover)

(styles.css):  
img:hover {  
    background-color: gold;  
}  
.Hi {  
    background-color: blue;  
}  
.Bye {  
    background-color: green;  
}

} So now if we move our cursor to image in class Hi, its background color will change from blue to gold & once I remove my cursor from the image, it will change back to blue. Similarly with image in class Bye

: The 'shover' helps in changing the color when cursor points at it.

## INTERMEDIATE CSS (PRACTICE 2)

• What are Favicons?

Favicons are images shown beside our website in tag region.



{ Website }

→ We can go to "favicon.cc" to make our own favicon. After making, download favicon in the same folder where our code is present.

↳(in vc code): <head>

|  
<link rel="icon" href="favicon.ico">  
</head>

Now we can see our favicon on our website.

• HTML Divs

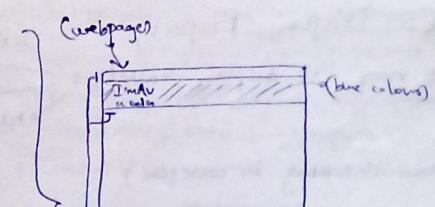
\* "Pesticide" extension for Chrome highlights all of my HTML elements & essentially all the boxes that are on screen.

(ctrl+mouse) → mouse points on a part of website → Pesticide shows what type is it (in blue colour)

\* Div stands for content division element, So it basically allows you to split up & divide your content into separate containers/boxes so that you can affect the layout of box separately

(index) → <body>  
    <div class="">  
        <h1>Im AV </h1>  
        <p>a coder</p>  
    </div>  
</body>

(styles) → div {  
    background-color: blue;  
}

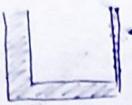


| Now to reduce this spacing

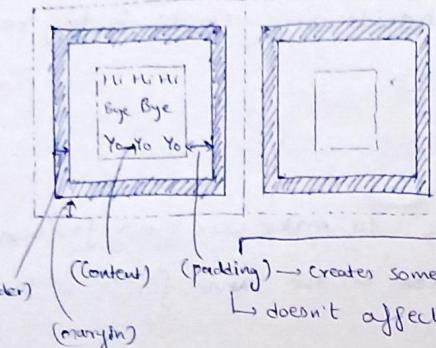
(styles):  
body {  
    margin: 0;  
}  
h1 {  
    margin-top: 0px;  
}

| (when 0, no need to specify unit)

## The Box Model of Website Styling

\* Border → 

{border-width: 0px 10px 20px 30px; }  
(go clockwise direction)



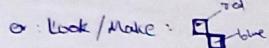
{border: 3px black solid;}

only pads the content, but not things like the background image/background color.

→ creates some space for content all-round.

→ doesn't affect content size

→ helps differentiate btw 2 boxes (if adjacent)

or: Look / Make: 

(finds)

<body>

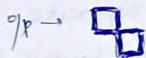
<div class="top-container">

</div>

<div class="bottom-container">

</div>

</body>

→ 

(Styles)

```
.top-container {
    background-color: red;
    width: 200px;
    height: 200px; border: solid 2px;
}
```

```
.bottom-container {
    background-color: blue;
    width: 200px;
    height: 200px; border: solid 2px;
}
```

Margin-left: 240px;

## CSS Display Property

It has 4 display values:

- Block
- Inline
- Inline-Block
- None

(Now checking in codepen)

\* Block elements are those that take up essentially the whole width of the screen on a web page (we can use pesticide & check), so effectively blocking out any other elements from sitting next to it on the left or on the right.

## Common block elements:

- Paragraphs (<p>)
- Headers (<h1> through <h6>)
- Divisions (<div>)
- List & List items (<ol>, <ul>, &c <li>)
- Forms (<form>)

ex: <p> a programmer </p> {Suppose we want to underline "pro"}

↳ <p> a <span class="pro"> pro </span> grammer. </p> } (using pesticide)  
→ styles → → .pro  
text-decoration: underline;  
↓  
→ a [programmer]  
↓  
→ pro is still in the same line

\* An Inline display element only takes up as much space as it needs to in the height & in its width so you can see that box around span is only as large as it needs to be!

## Common inline elements:

- Spans (<span>)
- Images (<img>)
- Anchors (<a>)

Ex: In practice 1 → that's why: My Hobbies Contact Me were side by side

• <p> Hello </p> { p { background-color: red; } }

→ <span> Hello </span> { span { background-color: blue; } }

→ (automatically goes to next line)

→ If span & para in same line  
→ span) para (preference)

In inline elements, we can't change the width of all the spans, but if we specify width to block elements, then it complies (But it still doesn't let any other element sit on the same line)

\* We can change the display property of any given element:

Ex: For paragraph element

↳ P { background-color: red; display: inline; }

→ [Hello]

→ (now can't change width)

- \* Inline - Block elements:
  - we can change width
  - we can make them go onto same line
- And this is kind of what image elements are like: inline elements (displayed next to each other). Ex: inline-block elements (can change it's size)
- \* None - simply removes that element from website, as if it didn't exist

Visibility - makes that element disappear but it keeps its original position. So all the other elements still flow around it as if it's still there.

```
<p>Hello</p>
<p class="second-p">World</p>
<span>Hello</span>
<span>World</span>
```

(Style) • p{  
background-color: red;  
width: 100px;  
display: inline-block;  
}

Span {  
background-color: blue;  
width: 100px;  
}

• second-p {  
display: none; } → [Hello] [Hello] [World]  
(OR)  
• second-p {  
visibility: hidden; } → [Hello] [Hello] [World]

- Learn More About Web Design
- Typeface: Name of the design in family of styles
  - Ex: Arial, Serif
- \* San-serif family looks so much friendly, more approachable, more novel and more contemporary
- (Try to make stuff using 2 fonts in general) ....
- Ex: Heading → Serif  
Body → San-serif { on vice versa } just creates more interest in my design.

### • CSS Static and Relative Positioning

Without CSS, our HTML element already has predefined rules for how it should be displayed on my webpage even if we don't have CSS

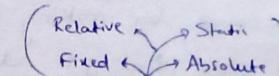
Let's take a look at these rules.

1) "Content is Everything": Block elements even though they take up 100% of width, height is still determined by the content so my content is the first thing that determines how large things get displayed & what the height & the width will be. And this is despite any CSS.

### 2) "Order Comes From Code"

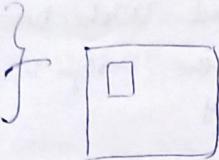
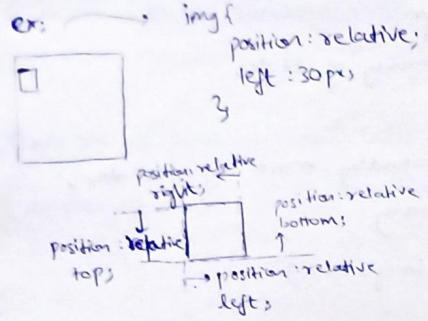
3) "Children Sit On Parents": By default our HTML elements that are children, they will sit on top of our parent element

```
<div>
  <h1>a <span>pro </span> grammar</h1>
</div>
```



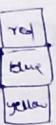
Now we can also set CSS property position property in order to position elements on screen the way that we want instead of just going along with the default layout.

- \* Static: All HTML elements are static in their position by default & static just means go along with HTML rules.
- \* Relative: Allows us to position the element that we select relative to how it would have been positioned had it been static/default.



when we move an element that has a relative position it doesn't affect the position of anything else on screen.

Now try to convert



to ~



```

→ (HTML)
<body>
  <div class="red">
    <div>
      <div class="blue">
        <div class="yellow">
      </div>
    </div>
  </div>
</body>

```

If (position: right/left) not written, then order: R B Y

Now reason why spaces btw our squares are inconsistent is because by making it an inline-block there's actually a little space that gets added in by the browser.

### Absolute positioning

```

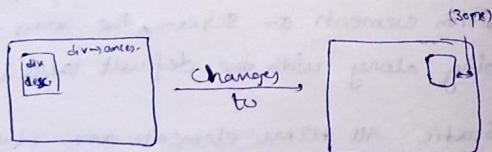
ex: div {
  position: relative;
}

```

```

  img {
    position: absolute;
    right: 30px;
  }

```



position relative to nearest positioned ancestor on top left corner of webpage.

In absolute positioning, we are positioning the element, so in this case the image element, relative to its parent, so in this case its parent is that red div.

In most cases, the parent is simply the entire body of the webpage.

\*Relative positioning means we are adding a margin relative to where the element should have been.

But absolute position means that we are adding a margin to its parent element.

ex: ①    ②    ;    when ③ not written: R  
 position: absolute;  
 left: 200px; — ④    when ③ written: B    R  
 ⑤    ⑥    ;  
 right: 100px;

By using AP (absolute positioning) it does affect the flow of our HTML. When we are using this, we are actually taking the element out of the flow of document & it's no longer considered a part of the natural flow of the document.

So that's why the blue square & yellow square have shifted up, because it's as if I've just gone & deleted that div. Basically red is now dead to blue & yellow

[ body { margin: 0px; } means now all elements will stick to the exact sides of webpage ]

Now the parent doesn't have to be the body (always). It can also be the closest parent that has a relative layer

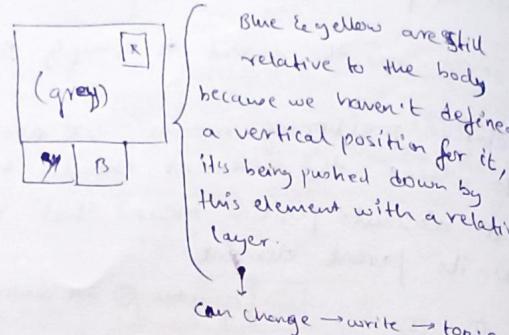
ex: ①    ;    (CSS)  
 <body>  
 <div class="container">  
 <div class="red">  
 </div>  
 ;  
 </body>

• container {  
 position: relative;  
 width: 300px;  
 height: 300px;  
 background-color: grey;  
}

② → left: 100px;    ④ { right ← deleted }    ⑤ → { position: absolute; } ⑥

Now our red square is a child of that parent container.  
And now it's defining its position relative to that grey  
container square.

If `6 {left: x;  
right: 20px;}`



\* Now if `.yellow`

`.yellow {`

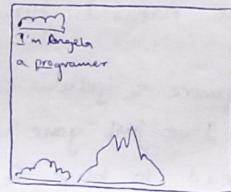
`position: fixed; /* position relative to top left */  
top: 0;`

(corner of browser window)

then if I scroll through the web page it will stay in its current position  
(useful when we have now bar on side bar)

## The Dark Art of Centering Elements with CSS

Now → my website :



To center elements → we use `:text-align` property inside our parent container

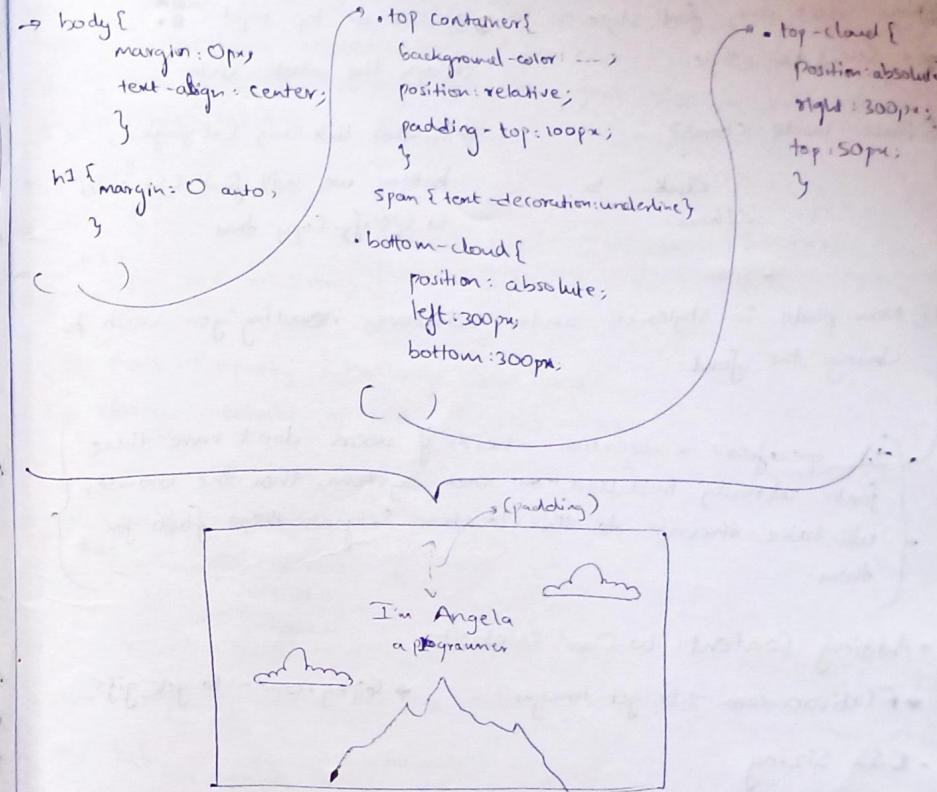
(HTML)  
`<body>`

```
<div class="top-container">  
    
  <h1>I'm Angela</h1>  
  <p>a <span class="pro">pro </span>grammer.</p>  
    
    
</div>
```

`</body>`

(style.css)

\* In order to position cloud relative to something, one of its parents has to have its position set as `relative`. And if we don't then it will be relative to the body.  
So, we want position of cloud relative to top-container.



## Font Styling in Our Personal Site

Sans-serif : **F**      Serif: **F**

For most browsers, the default serif is Times font, & the default sans serif is Arial

```
Ex: body {  
  font-family: verdana, sans-serif;  
}
```

Now if browser / OS that the user's using doesn't have this "verdana" font 'installed', then it will default to whatever is the sans serif font that is installed on their system.

\* To ensure that everybody has the same viewing experience regarding fonts on my website, we can use - font embedding.

(1) Go to: fonts.google.com

- (2) Now select any font style  
(2) + Select this style

(3) Paste inside: <head>

    <link href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;600;700&display=block" type="text/css" />  
    </head>

(4) Now paste in styles.css under whatever heading you wish to change the font.

This specifies a location where if users don't have these fonts already installed on their system, then the browser will take them to this location & grab these fonts for them.

## • Adding Content to Our Website

\* FlatIcon.com → to get images

\* Giphy.com → to get gifs

## • CSS Sizing

(styles.css)

h1 {  
    font-size: 16px;  
}  
16px = 100% = 1em  
(Static)  
means even if we change size of our webpage, our font size won't change

(Dynamic)  
adjusts itself as per webpage size

body {  
    font-size: 2em;  
}  
h1 {  
    font-size: 1em;  
}

I'm Angela } 3em

{ the parent's font-size (body) gets added to its elemental size (h1), when size is dynamic }

if → body {  
    font-size: 20px;  
}  
h1 {  
    font-size: 90px;  
}

I'm Angela } 70px → (Static size doesn't depend on any other sizes)

Now in order to avoid addition of parent size on its element, there is something called - rem (root em), so that means ignore all of the parent settings for the font size & just set it to this relative to the root.

ex: body {  
    font-size: 20px;  
}  
h1 {  
    font-size: 5.625rem;  
}

I'm Angela } 5.625em

## : CSS Font Property Challenge Solutions

To change colour of font → (style) → h1 {  
    color: blue;  
}

\* font-weight: specifies the weight / boldness of font.

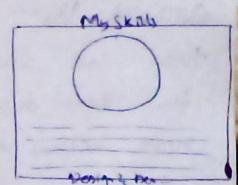
h1 {  
    font-weight: normal;  
        bold;  
        lighter;  
}

\* line-height: used to set the distance btw lines of text

ex: h1 {  
    line-height: 1; → (no units)  
        1 → (1 → default)

## • CSS Float and Clear

index: <div class="skills">  
    <h2>My Skills.</h2>  
    <div class="skill-row">  
          
        <h3>Design & Development </h3>  
        <p class="hi"> - - - - - </p>  
    </div>  
    <div class="skill-row">  
          
        <h3>Hot Wings Challenge </h3>  
        <p class="hi2"> - - - </p>  
    </div>  
</div>



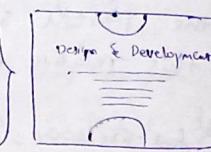
Now as we can see, the text takes 100% of space/width of webpage, so let's reduce it to 50%.

(Style) → .skill-row{  
width: 50%;  
}



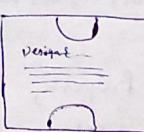
Now in order to re-arrange it to centre.

.skill-row{  
width: 50%;  
margin: auto;  
}  
{ auto helps to  
(create symmetry) }



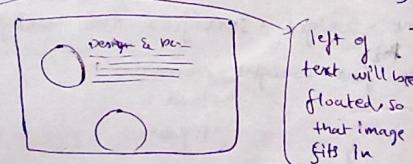
Now to make our text left-aligned.

.skill-row{  
text-align: left;  
}  
{ }



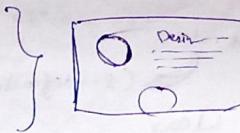
Now to wrap our text beside image, we use "float" property

.skill-row{  
float: left;  
}  
{ } ← left of  
text will be  
floated, so  
that image  
fits in



Let's give little more spacing btw image & text, to make it look more beautiful.

.skilage{  
margin-right: 30px;  
}  
{ }



Now, for example, if we wanted that title to wrap around this image & to be on the right of the image, then we can use another property to make sure that this text does not wrap & end up at the bottom over here below the image, & that property → "clear".

.hi{  
clear: left;  
}  
{ } ← clear kinda opposite  
of float

\*'padding-inline'\* means adding space on both sides

## Stylised Personal Site Solution Walkthrough

Now links present at bottom → [LinkedIn](#) [Twitter](#)

By default, all anchor tags have an underline that shows you that this is clickable

Now, we can get rid of that underline by changing text-decoration

(Style) → a{  
text-decoration: none;  
}  
{ }

a:hover{  
color: white;  
}  
{ color changes when mouse  
points on that }

\* 2023-Updated \*

## Introduction to HTML

### Self Closing Tags

→ <p> Hi </p>  
<br/>  
<p> Bye </p>

Hi \_\_\_\_\_  
Bye \_\_\_\_\_

{ (hr) horizontal  
rule element }

[<br /> on <br />]  
[<br /> on <br />]

To see World  
Heaven in Flower  
Infinity in hand  
Eternity in hour  
</p>

{ To see World  
Heaven in Flower  
Infinity in hand  
Eternity in hour }

### 'Break' element

<p>  
To see World <br />  
Heaven in Flower <br />  
Infinity in hand <br />  
Eternity in hour <br />  
</p>

→ Generally 'Break' element used for poems & stuff but if we want to go to next line → use new paragraph tag.

## Intermediate HTML

### Anchor Elements

#### HTML attributes

skeleton: <tag attribute="value" anotherattribute="value"> content </tag>

• <a draggable="true" This is Google </a> } means after selecting that text, we can also drag it.

## Image Elements

``

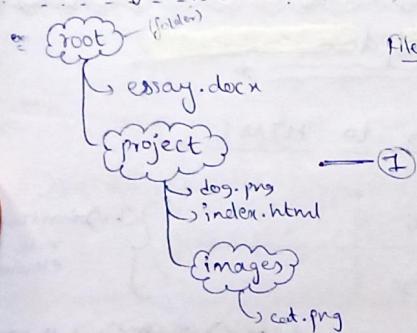
After a forward slash, we can add in the size of our image. So in this case we want a square of 200px x 200px.

## Multi-Page Websites

### Computer File Paths

A file path can direct someone to a file / folder.

#### Absolute file path:



File path: C:/Project/Images/cat.png

#### Relative file path:

If we consider "index.html" as our starting point, then relative to it how do we get "cat.png"?

images/cat.png

→ `./dog.png` → it tells to stay within the current directory & look over there  
(the folder that your file is directly located in.)  
(can also omit ./)

`../essay.docx` → it just means go up a level & then go into whichever resource/path you want to navigate

Here starting at index.html in project folder as this is where our starting point is. If we go level up using '..', then it enables us to go in root folder. Now relative to root folder, in order to navigate to essay.docx, we have to write → `../essay.docx`

## Now:

ex: ✓ 4.0 File Paths  
✓ Folder 0  
✓ Folder 3  
cat.png  
goal.png  
index.html  
rabbit.png  
solution.html  
✓ Folder 1  
✓ Folder 2  
bird.png  
fish.png  
dog.png

## Goal

All the animals  
Rabbit:  
cat:  
Dog:  
Fish:  
Bird:

**SOLUTION**  
<h2> All the Animals </h2>  
<h2> Rabbit: </h2>  
  
<h2> Cat: </h2>  
  
<h2> Dog: </h2>  
  
<h2> Fish: </h2>  
  
<h2> Bird: </h2>  


We will mostly be using relative paths.

At end of our website → we can add a footer element with our name & any copyright info. (or) other disclaimers.

ex: <footer>

<small> © Angela Yu. All rights reserved </small>

</footer>

## Introduction to CSS

### CSS Selectors

#### Attribute selector ex.

↳ `<p draggable="true"> Drag me </p>`  
`<p draggable="false"> Don't drag me </p>`

(styles.css)

↳ `p[draggable="false"] {`  
`color = red;`  
    `}`

Drag Me  
Don't drag me  
(red color)

#### Universal Selector

↳ ex: `*` `{ color: red; }`

means → select all

(start numbering from there)

<li class="note" value="2"> Hi </li>

<li class="note" id="id-selector-demo" value="3"> Yo </li>

<li class="note" value="4"> Bye </li>

</ol>

→ do change 'Yo' to blue colour

↳ (style.css)  
li[value="4"] {  
 color: blue;  
}

2. Hi  
3. Yo  
4. Bye

hybrid  
inherited

## CSS Properties

### Font Properties:

↳ Font size : 1px →  $(\frac{1}{96})$  of inch  
1pt →  $(\frac{1}{72})$  of inch

1em → 100% of parent  
1rem → 100% of root  
↳ relative to <html>  
</html>

↳ <html> → ①  
 <body>  
 <h1>Hello</h1>  
 <footer> → ②  
 <h2>Hello</h2> → ③  
 </footer>  
 </body>  
</html>

\* If ② → size = 20px  
③ → size = 2em =  $2 \times 20 = 40px$   
  
\* If ④ → size = 30px  
② → size = 20px changes to 50px  
③ → size = 2rem  
 ↴ =  $2 \times 30 = 60px$   
↳ independent of ②

Font Weight:  
normal, bold → keywords  
lighter, bolder → relative to parent  
number → 100~900

Font family → if font has many words in it, then write in inverted commas

↳ h2 {  
 font-family: "Times New Roman", serif;  
}

• Right Click → Inspect → we will get the coding for that website  
(e.g. my website)

## Text transform property:

↳ (style.css)  
h1 {  
 text-transform: uppercase;  
}

(all lowercase over full-width)

## Intermediate CSS

### The Cascade - Specificity & Inheritance

There are 4 broad categories which we look at when we're determining the overall level of importance of CSS Rule

→ Position      → Specificity      → Type      → Importance

\* POSITION: ex: li {  
 color: red;  
 color: blue;  
}  
↓  
li { color: green; }  
↳ the more lower, more preference is given  
↳ color would be shown as green.

\* SPECIFICITY: ex: <li id="first-id" class="first-class" draggable>

↳ li { color: blue; } → element ④  
.first-class { color: red; } → class ③  
li[draggable] { color: purple; } → attribute ②  
#first-id { color: orange; } → id ①

↳ Most priority given to 'id'  
↳ color shown is orange.

\* TYPE: ex: <link rel="stylesheet" href=".//styles.css" /> → external ②  
<style></style> → internal ④  
<h1 style=" " >Hello</h1> → inline ⑥

First we check any styles relevant from external stylesheet, then we check whether if there are any from internal stylesheet & finally if there are any inline.  
↳ (most priority)

\* IMPORTANCE: ex → color: red;  
color: green !important;

{ ensures that this is going to be most imp. rule relative to that element }

Important → doesn't matter how imp. this prev. rule is, it could have been inline (or it could have been in an id-selector, or it could have been at very bottom of page). This rule is going to come out as the Top Trump.

ex: `<h1 class="a-class another-class">Hello</h1>`

```
.a-class{
  color:green;
}
.another-class{
  color:blue;
}
```

## Combining CSS Selectors

(index) → `<div class="box inner-box"><p>white</p></div>`

{ (style) } .inner-box p { color:white; }

{ as long as it's closed inside the div, it counts as a descendant & we're looking for a paragraph }

Group = apply to both selector  
Ancestor → separated by comma.

↳ ex: `selector, selector { color:violet; }`

↳ (parent) ← (child) → (direct child)  
↳ selector > selector { color:firebrick; }

in order to select that particular element

→ It has to be only one level nested.

ex: `<div class="box"><p class="done">Hi</p>`

`<ul class="list"><li>Wash</li><li class="done">Read</li></ul>`

{ TO change 'li' elements to blue color.  
↳ box li { color:blue; }

Chaining = apply where all selectors are true → used to ↑ specificity

↳ selector selector {

  color:green;

  } → on: `<h1 id="title" class="big">Hello</h1>`

`h1#title.big`

{ always start with element if changing }

} color:greens

\* There is no space btw selectors.

## Combining combiners

(ancestor) (chain) selector selector {

  font-size:2rem;

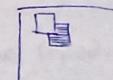
  }

## CSS Positioning

z-index → determines which one goes on top of which

↳ default = 0

↳ (in styles) z-index: -1;

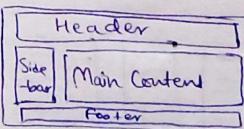


↳ border-radius = 50% (changes box to circle)

## Advanced CSS

### CSS Float

We use 'float' property on images → (in styles.css)



### How to Create Responsive Websites

Main ways to create responsive websites using CSS

#### 1) Media Queries

↳ CSS Grid

#### 2) CSS Flexbox

↳ External Frameworks... e.g. Bootstrap.

## Media Queries

Skeleton: @media (max-width: 600px) {  
h1 {  
font-size: 15px; } }

3 3  
(target: small screen)

here we define what's called break-point. It says that at this particular width, anything that is less than or equal to 600px, then should use this styling.

(it will override the general code)

ex: @media (min-width: 600px) {  
h1 {  
font-size: 15px; } }  
3 4  
600px  
(target: large screen)

Anything that is from 600px upwards, then we should have a diff. styling.

ex: @media (min-width: 600px) and (max-width: 900px) {  
/\* Styles for screens b/w 600px & 900px \*/ }  
y  
600px 900px  
ex: @media (max-width: 600px) and (min-width: 900px) {  
/\* Style for screens < 600px & > 900px \*/ }  
600px 900px

→ We write this in styles.css

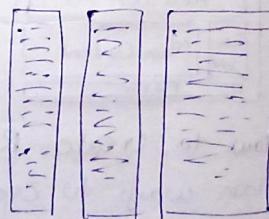
## Flexbox

### Display: Flex

```
<div class="container">  
  <div class="one"><p> - <p></div>  
  <div><p> - <p></div>  
  <div><p> - <p></div>  
</div>
```

(styles.css) → .container {

display: flex;  
gap: 10px;



<div></div>  
<span></span>  
<p></p>  
<img/>

(website)

→ when we use flexbox, we enclose all the elements that we want to display inside a flexible container, then what happens is all of these previous display values will be ignored.

And instead we're going to have everything laid out by flexbox so the width of each of the elements inside the flexbox will normally be based on the content size.  
{ doesn't matter what type of element is inside flexbox }  
(like block)

<div></div> <span></span> <p></p> <img/>

we can use 'gap' property to give space b/w them

if ~ display: inline-flex

<div></div> <span></span> <p></p> <img/>

### Flex Direction

flex-direction: column;

flex-basis: 100px;

(it's the height)

flex-direction: row;

main-axis

cross-axis

if ~ flex-direction: row;

flex-basis: 100px;

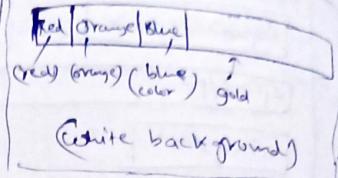
(it's the row)

100px 100px

\* flex-basis is set on child element

Ex: (body)

```
<div class="container">  
  <div class="red">Red</div>  
  <div class="orange">Orange</div>  
  <div class="blue">Blue</div>  
</div>  
</body>
```



(styles.css) → .red { background-color: red; };

.orange { " " : orange; };

.blue { " " : blue; };

.container {

color: white;

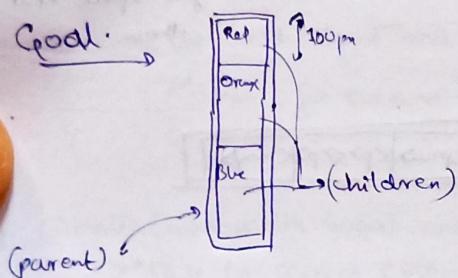
border: 5px solid gold;

display: flex;

}

②

Goal.



Soln. To change direction

→ container {

  flex-direction: column;

&

  display: inline-flex;



+ we write → flex-basis: 100px; in ②

↑  
(nothing's gonna change, as it's  
not used for parent)

Now to affect child → we use child combinator (>)

'>' placed btwn 2 CSS selectors. It matches only those elements matched by the second selector that are the direct children of elements matched by the first.

&

(\* ) CSS universal selector

~ we can select all the elements

{ ex. <div class="red">Red</div><div></div></div> }  
  { (not selected) }  
            { only children  
(not grandchild, etc.) }