CSS Selectors

Selector : is the first part of CSS-rule.

→ It is pattern of element and other terms that tell the browser which HTML elements should be selected to have CSS property value

⇒ the element or elements selected by the selector are called as "subject of the selector".

CSS-Selectors: are used to target HTML elements on the web pages that we want to style.

Types of selectors are: (5) — UICEG

1. Universal selector
2. ID selector
3. Class "
4. Universal element "
5. Group "

Universal-s : (*)

- They're denoted by asterisk sign (*)
- It selects all elements on the pages/document.

ex: `<HTML> <head> <style> * { color: green; font-size: 20px; } </style> </head> <body> <h2> the heading </h2> <p> all elements are selected </p> </body> </HTML>`

ID-Selector: (#)

- They are used for a single element.
- started with the '#' symbol
- Selects ID-attribute of HTML document for specific element.
- Id's are unique

ex: `<h, H, style> #para1 { color: blue; } </style> </Head> <body> <p id="para1"> Hello </p> <p> this is not affected </p> </body> </ HTML`

Class-selector: (.)

→ can be used in multiple elements

→ or elements in specific class. attribute

→ Started with period character (.) followed by class name.

ex:

```
<HTML> <head> <style>
    .center { color: blue; }
</style> </head>
<body>
<h1 class = "center"> blue heading </h1>
<p class = "center"> blue paragraph </p>
</body> </HTML>
```

Element Selector:

→ Selects HTML element by name.

ex:

```
<HTML> <head> <style>
    p { color: blue; }
</style> </head>
<body>
  <p> will be applied on every paragrap </p>
  <p id="par"> same too </p>
  <p> this two </p>
</Body> </HTML>
```

Group selectors :

⇒ used to select all elements with same style definitions.

⇒ used to minimize the code.

⇒ commas (,) are used to separate each selector when grouping

ex:

```
<Html> <head> <style>
    h1, h2, p { color: blue; }
</style> </head> <body>
<h1> Hello </h1>
<h2> how are you </ h2>
<p> welcome sir </p>
</Style Body> </ HTML>
```

CSS- preprocessors : (*)

are tools or scripting languages that are used to extend the functionality and capabilities of BASIC-CSS.

→ By adding features like Variables, Nesting, logics, mechas if-else, functions in css file....

⇒ they have their own syntax that will be compiled into regular css

• Some of popular CSS-PP are:

1→ SASS 2- LESS 3- STYLUS

SASS : ($)

→ Syntactically awesome stylesheet

→ is one of the most widely used CSSPP.

→ It was first developed in RUBY but later on add part of NPM.

→ in this all statements & variables are declared using '$' sign

→ SASS files can be created using two o/k syntaxes → • SASS
→ • SCSS (Scssy)

-- • SASS

is an indentation based which means it use indentation to separate code blocks instead of {} curly Braces and (;) semicolons.

ex:

```
$color: red
$BgColor: blue

Body
    color: $color
    Bgcolor: Bg color
```

-- • SCSS

- this more like traditional CSS, bcz it uses {}, and ; semicolons

ex:

```
$color: red;
$Bgcolor: Blue;
Body { color: $color;
       Bgcolor: $Bgcolor; }
```

Less [leaner style sheet]

→ in this variables & statements are declared by using "@" operator
→ it is close to SASS
→ features are: variables, nesting, functions & operations etc

example:

```
@color: Red;
@Bgcolor: Blue;
Body [ color : @color;
       Bgcolor : @bgcolor;
```

Stylus:

- is less popular but very powerful CSS-pp.
→ offers alot of flexibility interms of syntax & features
→ it allows both indentation & Regular-CSS like syntax.

CSS Flexbox and CSS-GRID (*)

Flexbox: It is one dimensional layout method for arranging items in rows or columns.
→ It made easier to design flexible & responsive layout.
•) to define FB module we need 1st to define ff:

•) **flex-container**: it is parent element that holds flex items.
- defined by setting Display property to "flex" or "inline-flex"

•) **flex items**: this is the child elements of flex container.

example
```
<div class = "flex-container">
  <div> one </div>
  ....
</div>
```
→ HTML

```
.flex-container { display: flex;
                  flex-direction : row;
                }
```
→ CSS

* properties of FC:
→ flex-direction
→ align-items
→ flexwrap (nowrap, wrap etc)

* properties of FI:
→ flex-grow
→ flex-shrink
- flex-basis
- align self

CSS-GRID: its two(2) 2-Dimensional layout sys.

→ It allows you to create complex layouts using rows and columns.
→ It provides more powerful and flexible way to design web layouts one of flex box
•) **GRid container**: it is parent element that holds the grid items, display: grid.
G-ITems: child elements of grid container.

example:
```
<div class = "container">
  <div class = "item"> <item 1 .... 5 </div>
</div>
.container { display: grid;
             gap : 10px;
           }
.item { background-color: blue;
        padding : 20px;
      }
```

*JAVASCRIPT FUNCTIONS AND OBJECTS, DOM manipulation (***)

Functions: they're blocks of codes designed to perform a particular task, they're executed when they're called (invoked)
→ with this you can re-use code, or write code that can be used many times
→ u can use same code w different argument to produce diff results.

Objects: they're collection of properties where a property is an Association (has) bw an name/key and a value.
→ they can contain multiple values & functions
→ they're like variable that can contain multiple value.
• they're declared w 'const' keyword. or
• they can be declared using "New" keyword.

Ex: `const person [ fname: 'A', lname: 'B' age: 10 };`

```
const person = new Object;
person.fname, lname, age = 10;
```

You can Access object property in two ways: `[""]` or `[.]`

ex: `person.lastname;`
`person ["lastname"]`

(*) Objects are "mutable" they're addressed by reference Not Value.

ex: `const x = person;` → creates a copy file of object
`x.age = 20;` → change age in both

Document Object Model (Dom)

→ refers to the process of using JS to interact with and change the structure (Html), style and content of webpage.

•) **properties are**: ⇒ change all HTML elements in the page.
- ⇒ change all HTML attributes
- ⇒ remove existing Html attributes & elements.
- ⇒ ADD NEW " " "
- ⇒ create new Html events
- ⇒ change styles of css in the page.

Short example

```
<html> <head>
<title> Dom manipulation eg. </title> </head>
<body>
<p id="paragraph"> This is paragraph </p>
<button onclick = "changeTxt()"> changeit </button>
<script>
    Function changeTxt() {
        ..... Let parag = document.GetElementsById("paragraph");
        parag.contentText = "I changed the text";
    }
</script>
</body> </html>
```

accessing Dom element — Dom Traversal — changing content of Html element

This is paragraph → I changed the text (Changeit / Events)

ADvanced JS- concepts

Closure • prototype • Asynwait

Closure: It is a function that have access to the parent scope, even after parent function has been closed.

Why	ADV
→ for maintaining state	→ protects variables from external access.
→ data privacy	→ allows more flexible code design.

```
Function outer() {
    let name = "abdi";
    function inner() {
        return 'hello ${name}' ; }
    return inner;
}
```

Async-await

First:

Promise: is an object represents Eventual completion or failure of a an asynchronous operation and its resulting value.

Await: an operator used to wait a promise it can only be used inside a asyn function.

→ It makes JS to wait until promise settles & returns its result.

Async: it is keyword placed before function we mean that function will always return a promise

ex:
```
async function F() {
    let promise = new promise ((resolve, reject)) => {
        setTimeout ( () => resolve ("done"), 1000 )
    } );
    Let result = await promise ;   → waits until response
    alert (result); }
F();
```

prototype: is fundemental concept for building re-usable code & creating custom objects.

→ It is an object used as template to create another objects or new one.

→ You can also add new custom methods & properties to the prototype we will be inherited by all objects created from it

→ It allows easy extension of existing objects and code- re-use.

•) all JS-objects inherit properties & methods from a prototype:

ex: Date object inherits from Date.prototype.
Array " " " Array.prototype --- So... on.
↳ and they all inherit from Object.prototype which is on TOP..

Ex
```
function me (first, last) {
    this.firstName = first; = last }

me.prototype.fullname = function () {
    return this.firstname + this.lastname;
};

Let a = new me ("abdi", "omar");
a.fullname();
```

Single Page Activation (SPA's)

It is type of web applications that loads & updates contents without refreshing the entire page.

→ updates single page HTML as user interacts with the app.

ex :- React & angular are built in SPA's

→ They allow you to manage state & render components without Reloading the page

Client Side Scripting vs Server side Scripting

Client Side Scripting	Server side Scripting
• Source code is visible to the user	• Not visible because output is in HTML page.
• main function is to provide requested o/p to end user	• to manipulate & provide access to respective DB as per Request
• depends on Browser & its version	• any serverside script be used, doesn't depend on the client
• Runs user's computer	• on webserver.
• doesn't provide security of the Data	• provides more security for Data.
• It's techniq used in web develop. in wk scripts run on clients comp.	• uses scripts on webserver to produce response wk is customized for each clients response.
• HTML, CSS, JAVASCRIPT	• Python, PHP, JAVA, Ruby used
• No need to interact w server	• It's all about interacting w server.
• ADV - faster response times - more interactive apps - manage & handle DB - Send request to server	• ADV - highly customize - response req. - access rights based on user.

DB for client side

DB :
- mySQL, mongoDB, postgreSQL...

- Frameworks :
  - express (node.js)
  - Django (PY)
  - Spring (Java)

(S.S)

Frameworks
- JQuery, react, Angular. and Vue.js

Scripting :-

It refers use of programming language to add Dynamic funcly, to website, It's classified into Serverside (SS) & Client side (CS)

Serverside :-

It occurs on webserver where server processes the Script & Generates HTML, CSS & JS files that are sent to client's web-Browser.

- It is used for common tasks such as :- DB-interactions, User-authentication and email sending

Client side :-

It takesplace on client's web Browser where script is executed & interacts with the user.

- It is usually used for validation, Animation, & Dynamic updates.

CSS - SYNTAX

It consists of a selector that tells HTML element wk will be affected followed by series of property & value pairs wk are called declaration Block

Syntax :-

selector { property: value; property2 : value 2; } ← rule.

↳ declaration

↳ declaration Block

example

```
test { color : red ; }
```

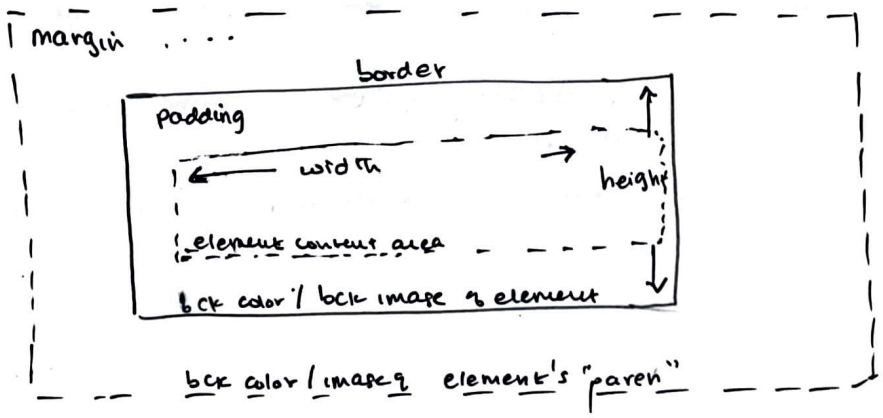
test → selector, color → property, red → value

```
p {
    font-weight : bold;
    font-family : Arial;
}
```

The Box-model in CSS (★★★)

→ It is fundamental concept wlc describes how elements are structured and rendered on a web page.

→ All HTML elements exists within Element Box



Properties of Box model are following :-

1- Background
2. Borders
3- Margins
4- padding
5- Width
6- Height

① Background :-

bck color / image of an element fills an element out to its border.

- Used to display presentational images.
- It allows to set Background values in one property.

B-properties :- - background - attachment → " repeat
- " color → " size
- " image
- " position

② Borders :-

- It provides a way to visually separate elements.
- → You can put borders all sides of elements as you want.

Border-properties :- - Border-style → " color → " image
- " width → " radius

③, ④ Margins & padding

Margin → space bln two paragraphs &

⑤, ⑥ Width, Height

Specify size of elements content area.