**Frontend Development:**

the part of an application or website that users directly see and interact with.

**Key aspects:**

* User interface (UI): the design and layout of the application/website
* User experience (UE): how and what users interact with the application.
* Core technologies: HTML, CSS, JavaScript
* Client-side development: FE code runs in the user’s browsers, making it accessible directly to them

**Core Technologies:**

HTML: hypertext markup language

to create structure of the webpage.

CSS: cascading style sheets

To style and layout web page.

JS: java script

Adds interactivity and dynamic behaviour to the web application.

**HTML**

Hyper Text Markup Language

To create the structure of a webpage

Webpage: a page that loads data from the server to client in a web browser

Different resolutions: 1280, 1338, 1536, 1640, 1920

HMTL has elements and attributes.

Boiler plate code – common code for almost any page. Can be obtained by ! + enter key. This is called emmet abbreviation, a tool kit to get the boiler plate code without typing everything.

<!DOCTYPE html> tells browser the html version we are using

<html lang="en"> html tag, root of a html doc, the biggest element of a webpage containing many other elements

<head> container for metadata, data that doesn’t get displayed on the browser

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0"> viewport is for responsive responsiveness (aspect ratio- size of device)

    <title>Document</title> contains the title of the page

</head> contains the data that the browser renders

<body>

</body>

</html>

We can inspect the code in view page source or in inspect. And can also make any changes in inspect.

Comments: <!—single line comment-->

tags - are names used to represent a particular format of content

tags + content => element

1. Block: group of elements

<elementNm></elementNm>

Eg: <p></p>, <div></div>

1. Inline: an element which opens & closes on itself.

<element/>

Eg: <input/>

Attributes – used to add additional characteristics/ properties to the elements

<elementNm attributeName= “attribute” />

<elementNm attributeName= “attribute” ><elementNm/>

Topics:

formatting elements - <strong> bold chars, <br/> break the lines, <u> underline

anchor elements – links <a> <a/>

lists – ordered lists <ol>, unordered lists <ul>, list items <li>, definition list of items <dl>, def term <dt>, def data <dd>

tables - <table>, table row <tr>, table head <th>, table data <td>, <caption> to mention heading of the table, table has rows and rows has columns.

Forms - <input type= “text/number/email” checked required />

checked – sets default value, required – can’t leave it empty, definitely should give a value

field set - <fieldset> <legend> heading for fieldset, <label> for titles, <input />

media objects - <video controls width = “150px” src = “path” />

<audio controls src = “path”/>

Controls – full screen option, play pause options.

semantic elements - <main> <header> <nav> <section> <article> <footer>

**CSS**

Cascading style sheets

Purpose: to design or style a webpage

Types/ methods:

Inline – used to style a sp element

Internal - <style> in head section of html

External – links an external style file in the head section

Selectors:

class (.) – selects html ele with sp class attribute

id (#) – selects html ele with sp id attribute

tags – selects html tag names

universal (\*) – selects entire page.

Color systems:

Rgb – red, green, blue, range: 0-255

Color: rgb(0,255,0) – green, Rgb(255,0,0)- red, Rgb(0,0,255)- blue

Rgb(255,255,0)- yellow

Hexadecimal (hex) hex= 6, decimal-10 🡪 total 16 digits

contains 0-9 (decimal num) + a,b,c,d,e,f (6 alphabets)

color: #ff0000 – red, #00ff00 – green, #0000ff- blue

comments: /\* comments \*/

text-properties:

text-align, text-decoration, font-weight, font-family, font-styles, line-height, text-transformation

font-family: arial, roboto, geneva

-if arial is not supported roboto will be applied, if roboto is not supported Geneva will be applied. This is called fall back mechanism

Box-model:

Margin - gap btw the main element and console or other elements,

Border – boundary,

padding – space around the content,

height & width of content

display properties:

inline – takes only the space required by the element (without any margin/padding)

block – takes full width space

inline-block – similar to inline but we can set margin and padding

none – to remove element from document flow

alpha channel – opacity (0-1)

rgba(212,54,541,0.5), opacity – 0 => invisible, a => 1 completely visible, a=>0.1-0.9 – visible with diff opacity levels

units in css: absolute(px, cm,inches), relative units (%, em, rem)

% - size is in relative to the elements parent object

Em – font size of the element relative to the font size of the parent element or if em is used for width, it will be in comparative to the font size of the element.

Par ele font size: 10px -> elemenet font size: 2em => 10\*2=> 20px, width: 4em = > 4\*20px => 80px

Rem – root em – font size of the root element

Font size of root ele- 12px, width of inner ele – 5rem => 5\*12 = > 60px

Vh- relative to 1% viewport(browser) height

Vw – relative to 1% viewport (browser) width

Positions:– how an element is positioned in a document – static (default) / relative/ absolute/ fixed

Static- default position (top, right, bottom, left, z-index properties have no effect)

Relative – ele is relative to itself (the above properties will work)

Absolute – positioned relative to its closest non-static positioned ancestor

Fixed – positioned relative to browser (removed from flow)

Sticky – positioned based on users scroll position

z-index:

stack level of elements

elements with larger z-index overlap the elements with smaller z-index

def value – 0

+ve values come on to the surface and -ve values go underneath the other elements with higher z-index values

Background images:

Background-image – setting img as bg.

Background-size – cover- covers the complete image/contain – fits the size of element and extra image will be discarded or repeated to fill the space/auto – def

Background-repeat: no-repeat – to avoid repetition of the image.

Flex box:

Flexible box Layout, 1-D layout method for arranging items in rows n columns

Display: flex

Flexbox properties-

Flexbox direction – sets how items are placed in the flex container, along which axis and direction

Flexbox-direction: row(def) (main axis- left to right)/ row-reverse (main-axis- right-left)/ column (main axis- top- bottom)/ column-reverse (main axis- bottom-top)

Flex items fit in the container by adjusting its own size

Justify-content – alignment along the main axis – start/end/centre/space-around/space-between/space-even

Align-content – alignment of space btw n around the content along cross-axis

Adjusts the items along the cross axis, center/flex-start/flex-end

Align-items – align items along the cross axis

Flex-wrap – nowrap/ wrap/ wrap-reverse

Flex for items –

Align-self: alignment of individual item along the cross axis

Flex-grow: how much flex item will grow relative to the rest of the flex items if space is available

Flex-shrink: how much flex item will shrink relative to the rest of the flex items if space is not available

Media queries:

Creates a responsive website

@media (width/min-width/max-width: \_\_px) {

Selector {

Properties: values

}

}

Transitions:

Enables to define the transition btw 2 states of an element

Transition-property: property to transit (font-size, width etc)

Transition-duration: duration of transition (2s, 4ms)

Transition-timing-function: ease-in/ease-out/ linear/ steps..

Transition-delay: 2s/4ms..

State of an item is defined by pseudo classes

Transform:

Used to apply 2D n 3D transformations to an element

Transform: rotate(xdeg) rotate/rotateX/Y/Z

Scale(x,y) – change in size along x n y axis

Translate(x,y) – move along axis in given dimensions

Skew(xdeg)

Animations: to animate css

@keyframe animeName {

From {property: value} – starting state

To {property: value} – end state

}

Animation properties: animation-name/duration/timing-function/delay /iteration-count/ direction(normal/reverse/alternative/alternate-reverse

**JS**

A programming lang to give instructions to the computer.

The code is written in browser. (inspect -> console) – a temporary way. Generally written in code editor(VS code)

Alert(“-----"); -> gives a popup msg on the console screen

Ctrl + l (windows) - > to clear the console

Console.log(“---“); -> to log(print) the output on console

; -> terminates the line of code

Js is connected to browsers via html as <script> just after the </body>

Run the html file and check in console in inspect.

Variables: - Containers for data storage

It can be changed. Ex: r = 10; r here is a variable and 10 is the data stored in the var

Text data is stored in either “” or ‘’ – they are referred as strings

“” / ‘’ is not used in console to print if it is a var

Js is a dynamically types lang – type of var need not be specified 🡪 js interpreter itself checks the type of a var during it is run time.

= -> assigning operator

== -> checking if the values are same

=== -> checking if the values are same along with the data types

Var rules:

1. Case sensitive
2. Only letters, digits, \_ and $ is allowed, even spaces are not allowed
3. Only a letter, \_ or $ can be used as first char of a var but not numbers
4. Resevered or key words cant be used as vars
5. Generally camel case is used for assigning a var (camelCase)

A kw should be used while assigning any var. kw available for purpose are var, let and const

And let and const are more considerable

Var – can be re-declared and updated. Global scope

Let- cant be redeclared but can be updated. – block scope, can be declared and value can be assigned/ initialized later at any time

Const – cant be re-declared nor updated. Block scope, const has to be initialized while declaring itself

Data types: can be checked using typeof(var) -> gives the dt of the var

Primitive data types: 7 -> number, string, null (absence of data), undefined, bigint, symol

Non-primitive data types: objects (arrays, funcs) -> collection of values, ogj had key: value pair in {}

Accessing value from an obj – key is used -> var.key or var[‘key’]

Assigning/ updating value of a key in an object => var[“key”]= value

Values of keys in obj can be changed as the address of obj is not being changed unlike the nrml var where when value is changed the address changes.

Comments:

//single line cmt

/\* multi

Line cmt \*/

Operators: - used to perform some operations on data

A + b -> expression, a, b are operands, + is operand

1. Arithmetic operations- + - \* /, modulus (% - remainder), exponential (\*\*- power), increment (++) and decrement(--) (unary operators)

A++ -> post increment – value changes from next step of operation

++a -> pre increment - values changes first and then operations will work.

a-- -> post decrement – value changes from next line of code

--a -> pre decrement -> value changes first and then the operations will take place.

b. assignment operators: = (assigning a value) += (a += 1 🡺 a= a+1) -= \*= %= \*\*=

c. comparison operators: == (equal to, != (not equal to), === (equal and same dt), !== (not equal and not same type), >, >=, <, <=

d. logical operators- &&, ||, !

cond 1 && cond 2 – true only if 2 conds are true

cond 1 || cond 2 – true if any one cond is true

!cond 1– true if cond is false

Conditional statements: - to implement conditions in the code

If stmt: Executes a cond if exp is true.

If-else stmt: executes either of the stmts.

Else-if stmt: executes the true cond among many conds given

e. ternary operator: cond? True output : false output, only operator that takes 3 operands

f. switch-case-break – checks a conds and breaks the later code from execution

loops: used to execute same piece of code repeatedly (again n again)

1. For loop – for (let i=1; i<=count; i++) {stmts}, let i= 1 -> initialization, i<=count- condition, updation
2. While loops – initialization; while (stoppingcondition) {stmts, updation}
3. Do-while loops – do{stmts} while (stopping cond)
4. For-of loops – for (let var of strVar) {stmts}
5. For-in loops – for (let key in objVar) {strmts}

Strings: a sequence of chars used to represent text

Creating a string – let strVar = “----” or ‘----’

String length – strVar.length

String indices – strVar[0], strVar[1], 0,1 here are index numbers.

template literals - a way to have embedded expressions in strings, `the cost of ${obj.key} is ${obj.val}`

string interpolation: to create string by substituting placeholders

escape characters: \n – new line

string methods - builtin funcs to manipulate a string

str.toUpperCase(), str.toLowerCase(), str.trim() – to remove white spaces, str1.concate(str2), str.replace(oldVal, newVal), str.replaceAll(oldVal, newValue) str.slice(startIndex, endIndex), str.charAt(index)

arrays: collection of items, a linear way of storing data, array doesn’t hv key, It has index.

Arrays are mutable.

Arr.length

Array is a type of object with key as its index

Array indices – arr[0], arr[2], 0,2 here are indices.

array methods:

push() - add to end,

pop() - delete from end and return,

toString() - converts array to string

concat() - joins multiple arrays and returns result

unshift() - add to start

shift() - delete from start and return

functions - block of code that performs specific task, can be invoked(called) whenever needed

2 steps – func def, func calling.

1. func def - Function funcName() {----}
2. func calling - funcName();

reduces the redundancy (repeatition) to minimize the code

forEach loop in arrays

arr.forEach(callBackFunc) , callBack is a func passed as an argument to another func,

callBAckFunc is a func that executes for each ele in an array

func attached to any object or data structure is refered as method

higher order functions - either takes a func as a parameter or returns a func

map - creates a new array from the values returned by the call back func

DOM – Document Object Model

Window -> document -> html -> head, body, every node in this tree is an object node

We can access html tags in js as an obj

any styles based on user input can be changed dynamically without making any changes/manipulation to the html or css an be done in js thru dom

DOM Manipulation

Script tag is written inside body tag so that the elements in html can be accessible to dom in js

accessing elements –

1. selecting via id – document.getElementById(“—“) – returns id elements
2. selecting via class name – document.getElementsByClassName(“---“) – returns html collection as array
3. selecting via tag name – document.getElementsByTagName(“----“) – returns elements with tag mentioned.
4. Query selector – document.querySelector(‘#id/.className/tag’) – returns first element with mentioned element
5. To select all elements using query selectors – document.querySelectorAll(‘#id/.className/tag’)

Properties: to access and change their values

1. .tagName – return tag name for element nodes
2. .innerText – returns the text content of the element and all its children
3. .innerHTML – returns the html contents in the elements
4. .textContent – returns textual content even for hidden elements

Nodes: text node, cmt node, element node

Accessing child nodes from the parent nodes - .children[0], .firstChild, .LastChild

Attributes:

.getAttribute(attr) – to get the attribute value

.setAttribute(attr, value) – to set the attribute value

Style – node.style.styleAttribute = “---“;

Create element: let ele = document.createElement(‘tagName’);

Insert elements:

Node.append(ele) – adds at the end of the node (inside the element)

Node.prepend(ele) – adds at the start of the node (inside the element)

Node.before(ele) – adds before the node (outside the element)

Node.after(ele) – adds after the node (outside the element)

Delete element – node.remove()

To add or remove classes from elements –

Node.classList.add(“newClsName”);

Node.classList.remove(“clsName”);

Events: the change in the state of an obj is known as event.

Inline (increases the html code), js (can add only 1 event at a time), so event listerners is the best choice

Used to trigger any action, can arise from event intereaction/ evnt. (like battery pop up) and many more

Types – mouse events(click, double tap…. ), key board e (key press, key down, key up, ker release)

Form e (submit etc), print etc)

Onclick, ondblclick, onmouseover – inline events – handling events inside a html

Node.event = () => { handler };

Event obj – has details about the event

All event handlers have the access to the event obj’s properties and methods

Node.event = (e) => { handler};

e.clientX – position along x-axis, e.clientY – position along y-axis, e.target – gives the ele , e.type – type of event

event listeners:

node.addEventListerners(e, callback func /handlers)

OOPS – object-oriented programming

Objects: an entity having state and behaviour (properties (vars) n methods (funcs))

Js obj has sp property called prototype, it can be set using \_\_proto\_\_

simply prototype is a reference of an obj

when obj and prototype has same methods, the method in obj will be applied.

Inheritance - Class Child extends ParentCls { ----- } – child cls has all methods n props of parent cls

super kw - used to call the constructor of its parent cls to access the parents props and methods - super(args), super.parentMethod(args)

whenever a child(derived) cls has to use this.arg in it, it should first invoke the constructor from its parents cls. this is done using super()

it also conveys any info from the child to parent cls as an argument for the super method.

// error handling - try-catch - to handle unexpected error in the code to avoid the code to stop running when any error occurs

let a = 10;

let b = 20;

console.log("a+b: ", a+b);

// console.log("a+b: ", a+c); // there is an error. so for such unexpected errors try-catch is used to avoid errors during compilation.

try {

    console.log("a+b: ", a+c);

} catch (err) {

    console.log(err); //the error will be logged as a text msg instead of an error

}

Async await >> promises >> callback

Synchronous – code runs in a particular sequence of instructions given in the program. Each instruction waits for the previous instruction to complete its execution. – may cause delay in the UI.

Asynchronous – allows to execute next instructions immediately and doesn’t block the flow.

APIs, setTimeOut()

Callbacks: a func passed as an argument to another func

callback hell - nested callbacks stacked below one another forming a pyramid structure

Promises - to resolve the above callback hell problem

it is a js obj that eventually completes a task, it has 2 handlers - resolve, reject - these are the callbacks provided by js

3 states of a promise – pending, fulfilled, rejected

an api returns new promise that a programmer works with.

Promises are used using .then() and .catch()

Promise chain: a prom returning a prom

promFunc().then((res)=>{return promFunc2()} .then((res)=>{return promFunc3()} .then((res)=>{concole.log(res)};

async-await - always returns a promise - async kw used before a func, await pauses the execution of its surroundings async func until the promise is settled.

iife - immediately invoked functions - called immediately as soon as it is defined. - doesnt hv to all it unlike async-await - but cant be used elsewhere. so it is used only when it is required to execute only once.

method 1- (function () {

    ...

}) ();

method 2 - (() => {

    ...

}) ();

method 3 - (Async () {

    ...

}) ();