JAVA SCRIPT

(Source - MDN Web Docs)

HTML - markup language we use to structure our webpage

CSS - CSS is language of style rules that we apply styling to our HTML content

Js - is a scripting language that enables us to create dynamically updating content, control multimedia, animate images etc.

What Js can do?
- store values in variables
- operate on pieces of text
- Run code in response to certain events such as onclick, onmouseover etc.

API - Application Programming Interface are ready made sets of code building blocks that allow a developer to implement programs that would otherwise be hard to implement

1.) Browser API's : built into our web browser.
- DOM
- Geolocation API
- Canvas & Web GL
- Audio & Video API

2.) 3rd Party API's not built in by default.
- twitter API
- GoogleMaps API
- OpenStreetMap API

Js runs top to bottom. and is a lightweight interpreted language.

Enter Js using script tag!

→ DOMContentLoaded element.

→ async attribute (only external scripts) // if scripts dont need parsing & can run independently.

→ defer // works as async but js scripts run in order they are written

→ use let (or var) to make variable

→ use const to make variable which can't change

→ use querySelector(); to select HTML elements by tag, class, id —

→ Functions

```
function funct_name(){
    // statements;
}
```

To run funct: funct_name();

→ Operators

+, −, *, /

↑ also for concating strings.

Assignment x=y, +=, -=, *=, /=, %=, **=, <<=, >>=, >>>=, &=, ^=, |=

* x+=y not same as x=x+y in a[i++] += 5 // i evaluated once.
a[i++] = a[i++] + 5 // i evaluated twice

Strict Comparision: === (both type & value) to be same.

Abstract Comparision: == (converts to same type & compares)

Operator: ==, !=, ===, !==, >, >=, <, <=

- When comparing String & Number, String is converted to number
- for Boolean. 1 is true and 0 is false
- for Object & String, Js operators attempt to convert the object to a primitive value (String or Number)

→ Events

- Events are things that happen in the browser – a button click, page loading, video playing etc. – in response to which we can run blocks of code.
- Constructs that listen out for the event happening are called event Listeners and block of code for response are

Called event handlers.

eg. submit.addEventListener('click', checkguess);

type of event we're listening for

code we want to run when the event occurs.

Some new functions

→ but = document.createElement('button');

→ document.body.appendChild(but);

→ but.parentNode.removeChild(but);

→ querySelectorAll()

→ Loops for (start value, exit condition, incrementor/decrementor){

// Statements
// to execute

}.

focus() function.

use to focus on elements so that can directly see cursor on the element I can use them instead of first clicking it itself & then using it.

eg. document.getElementById('mybut').focus();

attributes : focus({preventScroll : false});

querySelector()

This document method returns the first Element within the document that matches the specified selector, or group of selectors. If no matches are found, null is returned.

element = document.querySelector(selectors);

querySelectorAll() to get a list of all elements matching the specified selectors.

Types of Errors
- Syntax Errors
- Logical Errors.

→ Event Types for event listeners.

- → cached
- → error
- → abort
- → load
- → beforeunload
- → unload

} Resource Events

online
offline
} Network Events.

focus
Blur
} focus Event.

open
message
error
close
} WebSocket

animationstart
animationend
animationiteration
} CSS animations

transitionstart
transitionend
transitioncancel
transitionrun
} CSS Transition events

reset
Submit
} form events

auxclick
Click
dblclick
mousedown
mousecenter
mouseleave
mousemove
mouseover.
} Mouse events

and a lot more....

→ VARIABLES

Variable is a container for values whose contained values can be changed

Declare variable using var or Let.

```
Let myName;
Let myAge;

myName = 'keshav';
myAge = 20;

Let myBranch = 'CSE';
```

(var)'s design can sometimes be confusing or downright annoying.

difference b/w var & let

→ ~~var~~ In var usage, you can declare a variable after you initialise it and it will still work. (because of hoisting)

→ using var we can declare the same variable as many times as we like.

→ let allows us to declare variables that are limited in scope to the block, statement or expression unlike 'var' keyword which defines a variable globally or locally to an entire function regardless of block scope.

→ let donot create a property on global object.

```
var x = 'global';          console.log(this.x);   //global
let y = 'global';          "    "   (this.y);     // undefined
```

→ let variables are not ~~defined~~ initialized until there definition is evaluated. Accessing a variable b4 its initialization results in Reference Error. The variable is in temporal Dead Zone.

* Dont use underscores in starting of Js var names.
* " " Numbers " " " " " "

→ VARIABLE TYPES

* we don't need to declare variable types in Js.

- Numbers
- Strings
- Boolean.
- Arrays
- Objects

- Constant variables using const.

* typeof null = object

→ Dyanamic Typing: meaning that unlike other languages, we dont need to specify what data type a variable will contain

→ Basic Maths in JavaScript.

→ Number() function.

Syntax : new Number(value);

Properties : Number.EPSILON - Smallest interval b/w 2 representable numbers.

Number.MAX_SAFE_INTEGER - max safe int ($2^{53}-1$)

Number.MAX_VALE - Largest positive representable number.

Number.MIN_SAFE_INTEGER - ($-(2^{53}-1)$)

Number.Min_VALUE - Smallest positive representable num, (positive no. closest to Zero

Number.NaN

Number.NEGATIVE_INFINITY

Number.POSITIVE_INFINITY.

Number.prototype - allows addition of properties to Number Obj.

Methods : .isNaN() ; .isFinite(), .isInteger(), .isSafeInteger(),
↳ Determine value passed is. b/w $(2^{53}-1)$ & $(-(2^{53}-1))$

.parseInt(), .parseFloat(),

parseInt(string, radix)

2 to 36

for >10
A..B..C...
used.

Arithmetic + , - , * , / , ** ↳ Exponent

Operator Precedence

4 + 8/8 + 2 = 7 ✓ { because operator precedence }

and not 12/10 = 1.2 ✗