"Having understanding of fundamentals is more important than being a master of specific technology"

PAGE NO. :
DATE : / /

"Rest of the things you can Achieve through team work and Assistant tools"

"Hypersonic Js cheatsheet"

Chronologically,

(by "Brendan Eich") in 10 days

"Live Script" → "Java Script" (1995) → "ECMA Script"

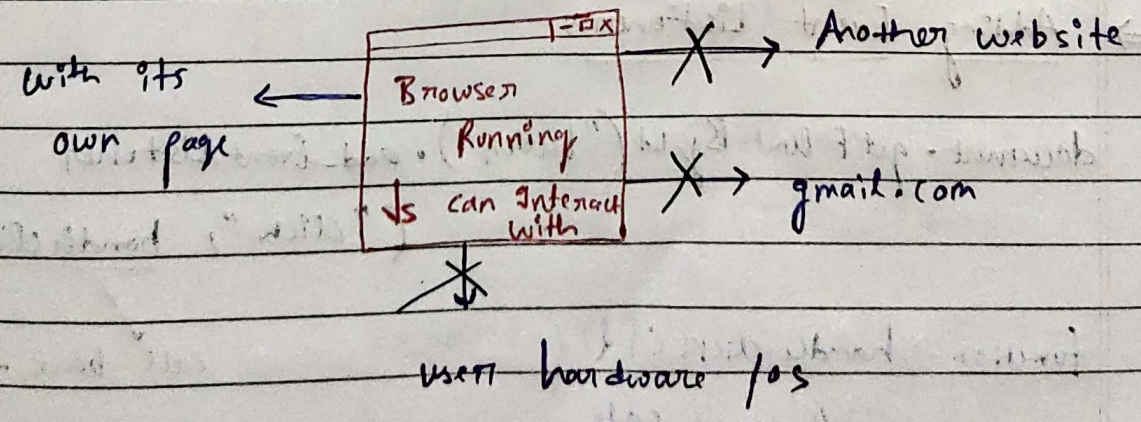
[Initially] "named only for marketing [at times when reasons" [java was popular] [when become fully independent lang.]

* Js runs on any device with "Js Engine"

Js → use to give web page a behaviour

Js Engine → V8 chrome, edge, opera | spider Monkey | Nitro in safari

* "Js is single-threaded [do one thing at a time but it uses asynchronous and event-driven features"




⇒ "98 % websites use Js for webpage behaviour"

PAGE NO.:
DATE: / /

```
alert ("hello")
var x = prompt ("enter variable x :");
console.log(x); // to print x
```

document - object - Model [ tree representation of html document ]

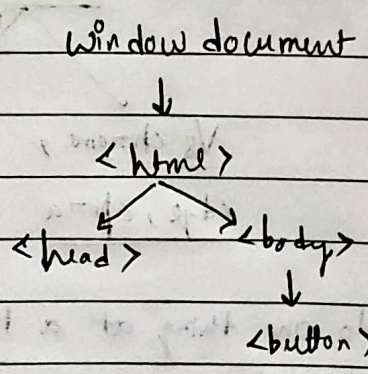
↑

"DOM - Manipulation"

↓

"Changing content of element"

```
document.getElementById ("heading").innerHTML = "Hi";
```




⇒ Adding Event Listeners

```
document.getElementById ("button").addEventListener
                                ("click", handleclick);
function handleclick () {
        // your code
}
```

callback function.

PAGE NO. :
DATE : / /

⇒ Playing Audio :

```
var audio = new Audio (" filepath");
audio.play();
```

⇒ JavaScript object

"Any real world entity with its own identity"

```
var doodhwala = {
        name : " chipa",
        #age : 21,
        lang : ["hindi", " japanese"],
        Ethnicity: " Alien"
}

alert (" hello, we have " + doodhwala.name);
```

by putting '#' it becomes private

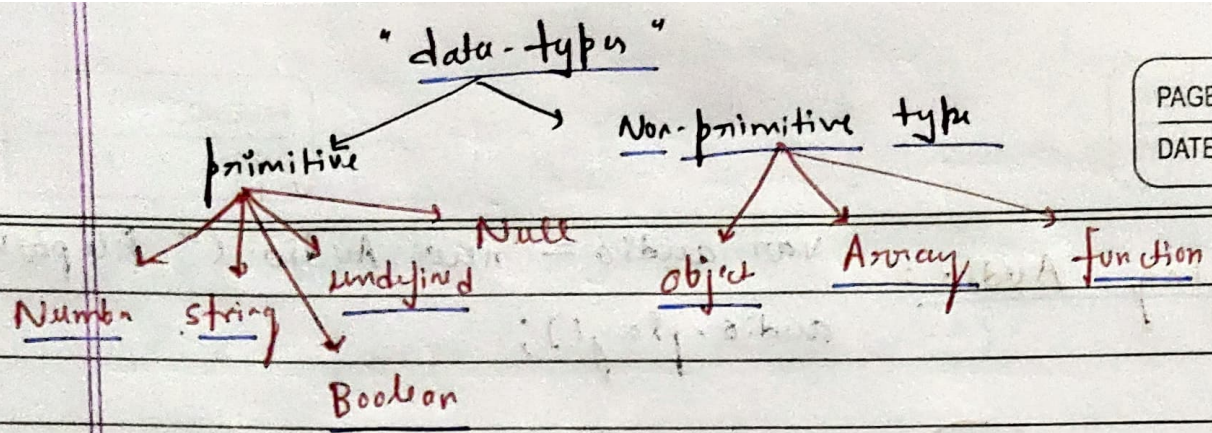
↳ array

⇒ Hoisting ⇒ using a variable or function without forward declaration.

eg → sayHelloTo (" Rajesh");

```
function sayHelloTo (name){
    console.Log (" hell sir" + name);
    console.Log (`Goodmorning ${name}`);
}
```

string formatting with back ticks ``




PAGE NO. :
DATE : / /

```
var isEqual = ("5" == 5)   // true, just compares value
var isEqual = ("5" === 5)  // false, also checks type
```

⇒ JavaScript Closures → "Allowing inner function to access its outside function scope"

```
function increment (raise-value){
    return function (salary){
        return salary + raise-value;
    };
}
```

"returns a function"

```
var bonus-function = increment (10000);
cout << "Your total income = " + bonus-function(50000);
```

output ⇒ Your total income = 60000

PAGE NO.:
DATE: / /

## ⇒ Javascript data transformation

```
var arr = [10, 20, 30]
```

1) `var divided = arr.map(element => element/3);`

or

```
arr.map(function(element) => { return element/3; });
```

or

```
arr.map((element) => { return element/3 });
```

```
console.log(divided);   // [1, 2, 3].
```

2) `var odd_elements = arr.filter(element => element % 2 !== 0);`

```
var index = 1;
```

3) `var total_sum = arr.reduce((accumulator, value) => value + accumulator, index);`

(20+30) = 50

## ⇒ Event propagation:

Clicking this will invoke that event or parent element 

 <a> click me </a> 

Suppose I added event Listener to 'p' element and Call back function for that event is alert("hello");

So if I click anchor tag <a> automatically event Listener attached to  gets invoked and vice-versa

"used in Responsive web-Applications, for fetching data from servers or performing time consuming operations without freezing user-interface"

PAGE NO.:
DATE: / /

# "Asynchronous programming in Javascript"

↓

"executing code without blocking main thread of Application"

## Asynchronous Js Concepts

- Call back function
- promises
- Async/Await
- Event handlers
  - event listeners

```
setTimeout (function(){
    robot.move();
}, 2000);
```

```
$("o").click
(function(){});
```

will execute
robot.move();
every 2000 millisecond

⇒ promises ⇒ represent a value or error that may not be available but yet it will be resolved at some point in future.

PAGE NO.:
DATE: / /

eg.

```
var server_status = "up";

function fetchData() {                          // creating a promise

    return new Promise((resolve, reject) => {

        if (server_status == "up")
            resolve("fetched successfully");
        else
            reject("Server is down");

    });
}

                                                // consuming promise
fetchData()
    .then((result) => {
        console.log(result);
    })
    .catch((error) => {
        console.log(error);
    });
```

→ output : "fetched successfully"

PAGE NO. :
DATE : / /

**Async / Await** → simplifies working with promises.
(In ES6+ JS) with more readable code

eg.

```
async function fetchData(){
    try {
        const result = await fetch("link");
        const data = await result.json();
    }
    catch (error){
        console.log(error);
    }
}
fetchData();
```

"Functions in JS"

- Named
  ```
  function add(x,y){
  }
  ```
- Anonymous
  ```
  function(x,y){
  }
  ```
- Arrow
  ```
  (x,y)=>{}
  ```
- Callback
  function passed as Argument later invoked by called function.

PAGE NO :
DATE : / /

→ eg of callback

```
function divide (x, y, Exception) {
    if (y == 0) {
        Exception ("divide by zero");
    }
}

var res = divide (20, 0, (error) => {
    console.log (error); });
```

// output : divide by zero

.....x.....

Rest Assured to God! Jay shree "RAM"!