+ JS Intro →

The most popular programming lang
It is the programming lang of the web

what can you do with js →
- To build Interactive webpages
- Realtime Networking apps
- Mobile Applications
- command line tools
- games

where does js code runs?
- only in browser

A browser is known to be a javascript engine that can execute js code

Fire-fox → Spider Monkey

JS engine name

chrome → V8 small

what is Node then?
Extract the open source js attach to a C++ code and this is how Node is made (having google's js engine)

Most often JS ASKED Questions

what we can build with JS?
with js we can build the backend of our website.

what is the difference

ECMA Script	Javascript
just a specification	a prog lang
It is Responsible for "Defining standards"	It confirms this specification
ES6 being recent	

Google
↳ right click
↳ Inspect
↳ Console → js console

```
> console.log('Hello world');
  Hello world
> 2+2
  4
> alert('yo')
```

Where to Start

(1) use code editor?
- VScode — • code.visualstudio.com • cross platform
- sublime text — powerful editor
- Atom

(2) use Node
- nodejs.org download
- we can also use chrome v8 engine too

(3) use live server in VS
- download live server extension
- open with live server for an html file.

(4) create a folder
- on desktop
- js practise folder

(5) create an html file
- html file in folder

(6) open with live server

- After download extension - restart VS code after download

(1) - use code editor - VScode
(2) - use Node - nodejs.org
(3) - use live server extension to run html
(4) - create a folder in desktop
(5) - create an html file in that folder
(6) - run by right clicking and opening it with live server.
(7) - use script tag

```
<body>
  .
  .
  .
  <script> - - - - <script>
</body>
```

(7) where goes js code in html
It goes under <script> tag
the script tag comes under

<Head> or	<body>
<script>	<script>

best practice is to add it at the ~~end~~ bottom of body ~~script~~ section

#seperation of concerns

keeping things seperated → clothes kept not on the bed but in a seperate cupboard

* we seperate html which is all about content from javascript which is all about behaviour

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" ~~content=~~
    content="width=device-width,
     initial-scale=1.0">
  <meta http-equiv="X-UA-compatible"
    content="ie=edge">
  <title> Document </title>
</head>
<body>
  <h1> Hello world </h1>
  <script src="index.js"> </script>
</body>
</html>
```

index.js

```
console.log ("Hello world");
```

a seperate file for ~~es~~ js

mention here

JS in node

→ downloading node from nodejs.org

→ open up command prompt → head over to the folder created js-practice

→ now run command node index.js

Variables

→ we use a variable to store data temporarily

declaration var a ~~=~~ → declaring with var
~~can~~ brings up issue
so
→ $\boxed{\text{let name}}$; by default variables defined there value is undefined

→ ~~&~~ strings can be both single or double quotes.

① → the name can't be a reserved keyword
② → the name should be meaningful (give on purpose of the var)
③ → they cannot start with the no.
④ → they cannot contain a space or a hyphen (-)
⑤ → can use camel Notation
⑥ → they are case sensitive
⑦ → to declare multiple variables

```
let firstName='abc', lastName='cde';
or
singe line   ~~fir~~ let firstName='abc';
single line  let lastName='cde';
```

Rules for Variable Name Selection

Page No.
Date : / /

Constants

- To ~~change~~ not change the value of variable we use constants → when you don't want to reassign values.

```
const interestRate = 0.3;
interestRate = 1;
```
→ error will be seen on console

Primitive types → there are two types → primitive types / Reference types

→ They are also known as value types too. the members are ↵

String   Number   Boolean   Undefined

```
# let name = 'Mosh';        String literal
# let age = 30;             Number literal
# let isApproved = true;    Boolean literal.
# let firstName = ;         Undefined
      or
  let lastname = null;  → clear out the value.
```

Dynamic Typing

Javascript is a dynamic language

static language → statically typed	Dynamic → dynamically typed
when we declare a variable it's ~~value~~ type can't be changed in the future	when we declare a variable, it's type can be changed at runtime

typeof to check the type of variable. → typeof name
→ "string"

* undefined is a type and also a value
* a value as null assigned to a var. It's typeof gives the type object
  ie let selectedColor = null; typeof selectedColor "object"

·objects·

$$\begin{tikzcd}
 & & \text{String} \\
 & & \text{Number} \\
\text{As mentioned before, there are two types of datatypes} \arrow[r] \arrow[rd] & \text{Primitive} \arrow[ruu] \arrow[ru] \arrow[r] \arrow[rd] \arrow[rdd] & \text{boolean} \\
 & \text{Referenced} \arrow[ld] \arrow[d] \arrow[rd] & \text{undefined} \\
\text{object} & \text{Array} & \text{function} \\
 & & \text{null}
\end{tikzcd}$$

what is an object → An object in js and other languages too is like an object in real life. Eg

A person → name
→ age   these are the properties of
→ address the object person

when dealing with multiple referenced & related variables. we can put these variables inside of an object.

For eg we have two var name and age → let name = 'Mosh';
let age = 30;

So here rather than referencing these two var separately. Making a single object person and putting these two variable inside of it. And referencing that instead

It makes our code cleaner →

```
let person = {
    name: 'Mosh',
    age: 30,
};
console.log(person);
```

① By object literal

② Key: value here are the two properties

Change the property name
// Dot Notation
```
person.name = 'John';
console.log(person.name);
```
// bracket Notation

Page No.
Date: / /

person['name'] = 'Mary';
console.log (person.name);

which Approach is better - Dot or bracket Notation.
- dot Notation is a bit more concise that should be a default choice.
- bracket Notation is helpful when we dont know value at the moment
  eg let selection = 'name';
  person [selection] = 'Mary';
  console.log (person.name);

Arrays

· List of products in a cart
· List of colors a user has selected

① when dealing with list of objects we use Array - Array is needed to store that list

② let selectedcolors = [→ square brackets denotes Array literal
'red', 'blue', 'black'];

③ Every element has an index in the Array   console.log (selected colors [0]);

④ As JS is a dynamic language. so the type of variable can change at runtime The same principle applies to arrays. ~~so the type of variab~~ so the length of the array as well as the type of object in array are dynamic as in they can change eg selected colors [2] = 'green'; → the array length will now change
selected colors [3] = 1;

⑤ Type of an array is object

functions

one of the fundamental building blocks in js. A set of statements that performs a task or calculates a value

For eg function greet() {
```
        console.log("Hello");
    }
```
// we do not need a semicolon at the end of the function as we are not declaring it as a variable.

```
//calling a function
greet();
```

Q1= write a function that returns the square of a number
```
function findsquare (n) {
    return n*n;
}
console.log(findsquare(2));
```

Q2= write a function to convert celsius to Fahrenheit
```
function calFahrenheit (cel) {
    return ((cel *(9/5))+32 ;
}
console.log (calFahrenheit(20)) //68
```

Q= write a function to find area of rectangle
```
function area (l,b) {
  return ("The area of rectangle is ${a*b}");
}
console.log(area(10,20));
```

Q write a func to cal area and peri of circle
```
function circlevalue(rad){
    return `Peri : ${2 * Math.PI * rad },
     Area : ${ Math.PI * rad * rad }`;
}
```

Page No.
Date :

Q write a function to reverse a Number

```
function reverseNum (num) {
    var reverse = 0;
    while (num != 0)
    {
        reverse = reverse * 10;
        reverse = reverse + num % 10;
        num = Math.trunc(num/10); → remove
                                    decimal
                                      no.
    }
    return reverse;
}
console.log (reverseNum (123))
```

Q count no. of vowels in a string

```
function countvowel (str) {
    let count = 0;
    str = str.toLowercase();
    for (var i=0 ; i< str.length ; i++) {
        if (str.charAt (i) == "a") ||
            str.charAt (i) == "e") ||
            str.charAt (i) == "i") ||
            str.charAt (i) == "o") ||
            str.charAt (i) == "u") {

            count++;
        }
    }
    return count;
}
console.log ((countvowel("Hello"));
```

Q Flatten array of arrays using JS reduce
flatten a 2D array to 1D array by using JS redux

```
function flattenArray (arr) {
    return arr.reduce((result, array) => result.concat(array));
}
    console.log (flattenArr([[1,2,3],[4,5,6],[7,8,9]]))
        // [1,2,3,4,5,6,7,8,9]
```

Q write a program to check it entered no. is a palindrome

0 1 2
LoL

3-0-1 = 2

1
0

3-1-1
1

```
function checkPalindrome (str) {
 for (var i=0; i<str.length; i++) {
   if (str.charAt(i) != str. charAt(str.length-i-1)) {
       return false;
   }
 }
 return true;
}
```

Q calculate simple Interest based on principal Amount

```
function simpleInt~~erest~~ (principle, rate, time) {
   var finalAmt = principle + principle * time * rate;
   return finalAmt;
 }
 console.log (simpleInt (20000, 5, 2)) //220000
```

Q cal compount interest based on principle amount

```
 function compoundInt (princi, rate, time) {
    var interest = princi * (Math.pow((1+ (rate/n)), (n * time)));
    return principle + interest;
 }
 console.log (compountInt (20000, 5, 4, 2)) //1304250
```