

JAVASCRIPT Basics

What is JavaScript?

- Scripting language, low level programming language.
- Dynamically typed language.
- Scripts are basically some functions which tells how to do and what to do?
- Client-side language, which can be run in browser.
- Developed by Netscape (in just 10 days, in 1995), initially Mocha was named.
- Some developer adds this into C++ and it became Node.js so that we can use it outside the browser server – side.

What can we do with JS?

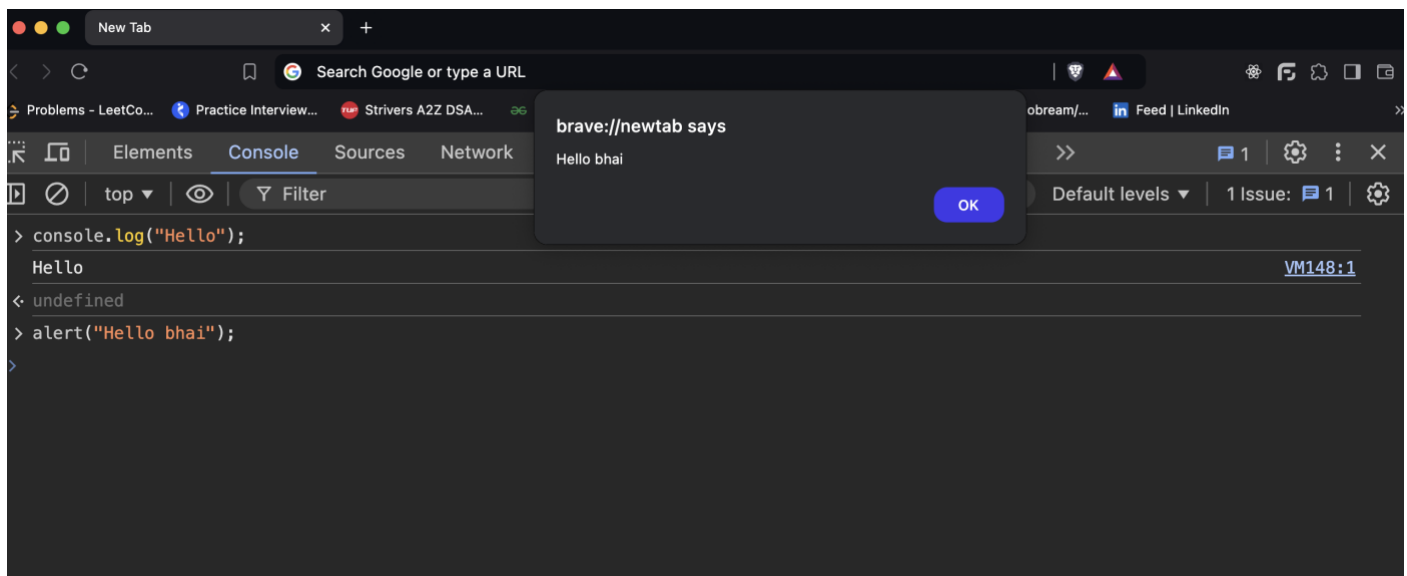
- Webapp / mobile app / network apps
- CLI tools
- Games

Do we need compiler?

- We need only JS Engine.
- Like in firefox Spider Monkey, Google V8.

Simple way to run a JS Code in Browser.

- Open any browser.
- Right click on anywhere on the screen.
- Click on inspect
- Find the console window.

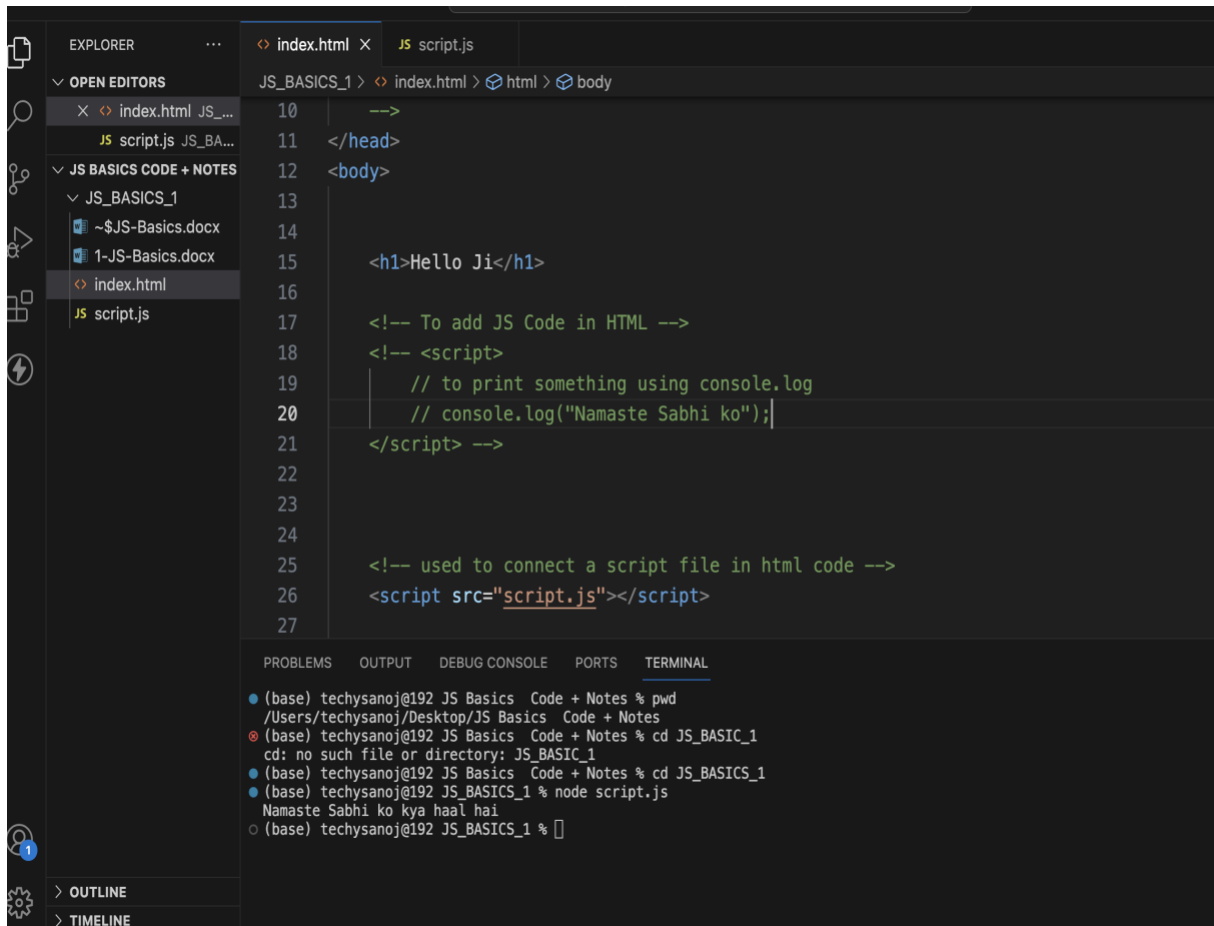


JS Setup?

- VS Code
- Node js

Adding JS Code in HTML + CSS + File

- Check file in JS_BASIC_1
- String – sequence of characters
- console.log() – use to print something in the console.
- ; - is used to end the line.
- // - is used to add comment in the JS Code.
- Create a separate file for the JS code.
- Lower ss is telling about how we can run code in terminal



```
index.html
10  -->
11  </head>
12  <body>
13
14
15  <h1>Hello Ji</h1>
16
17  <!-- To add JS Code in HTML -->
18  <!-- <script>
19  |   // to print something using console.log
20  |   // console.log("Namaste Sabhi ko");
21  </script> -->
22
23
24
25  <!-- used to connect a script file in html code -->
26  <script src="script.js"></script>
27
```

```
terminal
(base) techysanoj@192 JS Basics Code + Notes % pwd
/Users/techysanoj/Desktop/JS Basics Code + Notes
(base) techysanoj@192 JS Basics Code + Notes % cd JS_BASIC_1
cd: no such file or directory: JS_BASIC_1
(base) techysanoj@192 JS Basics Code + Notes % cd JS_BASIC_1
(base) techysanoj@192 JS_BASIC_1 % node script.js
Namaste Sabhi ko kya haal hai
(base) techysanoj@192 JS_BASIC_1 %
```

Variable

- Named memory location is called as variable
- Can be created using “let” and “var” keyword. Some of the example
 - o let a = 5;
 - o let name = “Sanoj”;
 - o let status = true;
 - o var check = false;
 - o var name = “Sanoj”;
- Difference between var and let are simple scoping of the variables.
 - o let are basically scoped in the closed curly bracket, var are basically global variable.
 - o Example:
 - {
let a = 5; // here ‘a’ is only scoped in these curly bracket
var b = 5; // but ‘b’ can be accessed outside these bracket
}
 - o let cannot be redeclared but var can be redeclared.



Here's a table outlining the differences between `let` and `var` in JavaScript:

Feature	let	var
Scope	Block-scoped (limited to the block where it is defined).	Function-scoped (or globally scoped if not inside a function).
Hoisting	Hoisted but not initialized (Temporal Dead Zone until declaration is encountered).	Hoisted and initialized with <code>undefined</code> .
Re-declaration	Cannot be re-declared in the same scope.	Can be re-declared in the same scope.
Initialization	Optional at the time of declaration (can be declared without value).	Optional at the time of declaration (can be declared without value).
Global Object Property	Does not create a property on the global object when declared globally.	Creates a property on the global object when declared globally.
Re-assignment	Can be re-assigned.	Can be re-assigned.
Temporal Dead Zone (TDZ)	Exists in the TDZ, so it cannot be accessed before initialization.	No TDZ; accessible even before declaration (but holds <code>undefined</code>).

Constant

- fixed value which cannot be changed further
- using `const` keyword

Variable Naming Convention

- Cannot be a reserved keyword like we cannot use `let` as a variable name, if as a variable name etc.
- Name should be meaning full
- Cannot start with a number.
- Cannot contain space or `'-'`.
- Naming should be done in CamelCase. Example SanojKumar etc.
- Declaring multiple variables let a, b, c; but best practice is used to create all these variables in different lines.

Primitive Types

- String: Sequence of characters example "sanoj" etc.
- Number: any number 1, 2, 3, 1.4, 1200, etc.
- Boolean: True, False.
- Undefined: Which is declared but not defined. Example let a; here a is undefined.
- Null: Value is declared and defined but the value is NULL means no value is present.

Dynamic Typing

- Means at the runtime we can change the type of the variable.
- Increases the flexibility.

```
const num = 10;
// num = 13; it will show the error assignment to constant variable

let Name = "Babbar";
console.log(Name); // prints Babbar

Name = 10;
console.log(Name); // prints Babbar
```

Reference Data Types

- Objects: Any element having properties and behavior. Like Person having name, age as properties
- Key – Value pairs are basically stored in object.
 - o `let person = {
 firstName: "Sanoj",
 age: 24
 };`
 - o Access using dot notation, `person.age`; etc.
 - o Access using bracket notation, `person['age']`;
 - o There is difference of using these two notations.
- Arrays: A list of element (items). Can be same kind of items and can be different.
 - o `let arr = [1, 2, true, 'Sanoj']`;
 - o `let arr = ["sanoj", "kumar"]`;
 - o Accessing element using Indexes. Starting with 0 index till `len - 1`;
 - o What if we try to access index which is not present it will give me undefined value.
- Functions: Later we will use it.

Operators

- Arithmetic: `+`, `-`, `/`, `%`, `**` (exponential/power)
- Assignment: `=` (equal to operator), `+=`, `-=`, `*=`, `/=`.
- Comparison: `<`, `>`, `<=`, `>=`, `==` (to check equality – strict equality), `!==` (to check not equality – strict not equality)
 - o Loose equality (`==`): Only value is checked.
 - o Strict Equality (`===`): Value + Type is checked.
- Bitwise:
 - o Bitwise AND (`&`): works on the bits of the number.
 - o Bitwise OR (`|`): works on the bits of the number.
 - o Example: `0010 & 0011` = it will give me `0010`
 - o Example: `0110 | 0011` = it will give me `0111`
- Logical: AND, OR, NOT
 - o When we are having multiple condition and we want to process them using some condition there we use Logical Operator.
 - o AND (`&&`): All condition must be true
 - o OR (`||`): Any condition is true. Short circuiting (means it will execute its execution when it will find the first truthy value).
 - o NOT (`!`): Change the Boolean value.
 - o It can also be used with non-Boolean value.
 - o Check using the code. There is something which are known as Falsie and Truthy Value.
 - o Falsie: `undefined`, `null`, `0`, `false`, `'`, `NaN`
 - o Truthy: Anything that is not falsie.
- Pre / Post Increment / Decrement operators:
 - o Pre (`++ x`) first increment then use
 - o Post (`x ++`) first use then increment
 - o Similar in Decrement
- Ternary Operator
 - o `Condition? val1 : val2`
 - o If condition is true then it will be assigned `val1` else `val2`
 - o Like `let status = (age >= 18) ? 'I can vote' : 'I cannot Vote'`;

Operator Precedence

- Using brackets
- Example: `let c = a + b * d / c`;
- Use brackets: `let c = a + ((b * d) / c)`;

```

console.log(3 + 5); // 8
console.log(5 - 10); // -5
console.log(5 ** 2); // 25

console.log(2 > 3); // false
console.log(3 > 2); // true
console.log(3 === 3); // true
console.log(1 == '1'); // true
console.log(1 === '1'); // false

console.log(Name++); // 10
console.log(++Name); // 12

let age = 17;
let sta = (age >= 18) ? 'Yes' : 'No';
console.log(sta); // No

console.log(true || false); // true
console.log(false || 'sanoj'); // sanoj
console.log(true && 'sanoj'); // sanoj;

```

Control Statements

- if – else:
 - if (condition) {
 - }
 - Elseif (condition) {
 - }
 - else {
 - }

```

let marks = 98;
if(marks > 90) {
  console.log('A');
}
else if(marks >= 80) {
  console.log('B');
}
else if(marks >= 70) {
  console.log('C');
}
else {
  console.log('D');
}

// Output = A

```

- Switch:
 - switch(expression) {
 - case 1: ---
 - case 2: --
 - case 3: --
 - default: --

```

let check = 2;
switch(check) {
  case 1: console.log('A');
    break;
  case 2: console.log('B');
    break;
  case 3: console.log('C');
    break;
  default: console.log('Enter a valid number');
}

// B

```

Loops

- For Loop:

- When we want to do repetition of task.
- `for (let i = 0; i <= 5; i++) { - - }`
- first part is initialization, then condition, then increment or decrement or updating.

```
// For Loop
for(let i = 0; i<5; i++) {
  console.log(i);
}
// 0, 1, 2, 3, 4
```

- While Loop:

- Only condition is checking and updation is done inside the block and initialization is done outside the while loop
- `While(condition) { - - , updation }`

```
// While Loop
let i = 0;
while(i < 5) {
  console.log(i);
  i++;
}
//0, 1, 2, 3, 4
```

- Do While Loop:

- Same as while loop but it will run 1 time for sure.
- `Do {`
`} while(condition)`

```
// do while loop
let c = 0;
do {
  confirm.log(c);
}while(c <=4);
// 0, 1, 2, 3, 4
```