# JavaScript

1. Make web pages interactive and Dynamic

2. Able to interact with user.

# ES5 vs ES6

ES5 -supported by most of the browsers

ES6 - Needs polyfills , transpilers

# S/W's

https://jsbin.com/?js,console

This will launch JavaScript and console

Sample Code:

Create a html page and inside body write the code to get interactivity.

**<a href="#" onclick="alert('hello')">Click Me </a>I**

Include the JavaScript in html

<script src="script.js"></script>

**script.js**

alert('hello JS');

# First JavaScript Alert and Prompts

console.log();

Outputs a message to the web console.

#### VARIABLES

1. variables are named space in the memory to store a value.

# Identifier Rules

* Cannot be a keyword
* Cannot begin with a number
* Cannot contains spaces or special characters other than \_ or $

Variables must be declared before using it. Only declare a variable once with in its scope. Otherwise you get already declared error in the code.

### **ES6 - let and const**

## Example: var

var myValue ='Hello World'

var myNumber=10;

console.log(myValue);

console.log(myNumber);

# JAVASCRIPT DATA TYPES - 7 TYPES

* Boolean
* Null
* undefined
* Number
* String
* Symbol (introduced in ES6)
* Object

## Examples

var a=true; //true or false =Boolean

var b=100; // can be written with or without decimal point =Number

var c='Hello World'; //Can be inside Single quotes or double quotes =String

var d=null;//it is explicitly nothing =Null

var e; //Has no value but it is declared =Undefined.

var f=Symbol("value"); //represent a unique identifier= Symbol

To find the variable type use typeof c ; //String

# LET AND CONST SCOPES

## LET

* New keyword for declaring variables.
* Has block level scope.

## Example : let

let name=prompt("What is your name?","None"); //Here none is default value.

console.log(name);

## CONST

* Cannot be reassigned a value
* value is immutable
* cannot be redeclared
* requires an initializer
* block level scope
* read only reference and value is fixed.

## Example :const

const pi=3.14;

console.log(pi); //cannot be changed

### **Data types Expression**

let a=""+5 //?? "5"

let b=""+6+7= "67"

let c=true+false;

## Example: let keyword

## ES6 syntax

for(let i = 0; i < 5; i++) {

console.log(i); // 0,1,2,3,4

}

console.log(i); // undefined

## ES5 syntax

for(var i = 0; i < 5; i++) {

console.log(i); // 0,1,2,3,4

}

console.log(i); // 5

**// The const Keyword**

* The new const keyword makes it possible to define constants.
* Constants are read-only, you cannot reassign new values to them.
* They are also block-scoped like let.

## Example: const

const PI = 3.14;

console.log(PI); // 3.14

PI = 10; // error

However, you can still change object properties or array elements:

## Changing object property value

const PERSON = {name: "Peter", age: 28};

console.log(PERSON.age); // 28

PERSON.age = 30;

console.log(PERSON.age); // 30

## Changing array element

const COLORS = ["red", "green", "blue"];

console.log(COLORS[0]); // red

COLORS[0] = "yellow";

console.log(COLORS[0]); // yellow

#### JAVASCRIPT OBJECTS

* An object is a collection of related data and /or functionality (Which usually consists of several
* variables and functions -Which are called properties and methods when they are inside objects).
* Objects are not primitive data types

## Sample code

var person={age:25};

person.age - Dotnotation

person['age'] - Bracket notations

## Examples

var myobject={};

console.log(myobject)

**o/p {} //nothing is stored in object**

let myObj={first:"Ram",last:"Kumar"}

myObj.first="SriRam";

console.log(myObj);

#### JAVASCRIPT ARRAYS

var shopping=['bread','milk','cheese','noodles'];

console.log(shopping);

console.log(shopping[0]);

## Array Methods

console.log(shopping.length);

#### JAVASCRIPT FUNCTIONS

* A function in JavaScript is a set of statements that performs a task or calculate value.
* Block of reusable code.
* A function definition(also called function declaration, or function statement)consists of the function keyword followed by,

1. The name of the function
2. A list of parameters to a function enclosed in paranethesis.
3. The JavaScript statements that define the function , enclosed in {}

## Functions and Scopes

**Global vs local variables**

<script>

let a="Hello";

function test(){

let b="world";

console.log(a+b);

}

test();

console.log(b);

</script>

o/p: Reference Error : b is not defined.

**default values to the function parameters.**

function sayHello(name='World') {

return `Hello ${name}!`;

}

console.log(sayHello()); // Hello World!

console.log(sayHello('John')); // Hello John!

# ARROW FUNCTIONS

* Arrow Functions are another interesting feature in ES6.
* It provides a more concise syntax for writing function expressions by opting out the function and return keywords.
* Arrow functions are defined using a new syntax, the fat arrow (=>) notation.

Let's see how it looks:

### **Function Expression**

var sum = function(a, b) {

return a + b;

}

console.log(sum(2, 3)); // 5

**// Arrow function**

var sum = (a, b) => a + b;

console.log(sum(2, 3)); // 5

## Example 2

// Single parameter, single statement

var greet = name => alert("Hi " + name + "!");

greet("Peter"); // Hi Peter!

## Multiple arguments, single statement

var multiply = (x, y) => x \* y;

alert(multiply(2, 3)); // 6

## Single parameter, multiple statements

var test = age => {

if(age > 18) {

alert("Adult");

} else {

alert("Teenager");

}

}

test(21); // Adult

## Multiple parameters, multiple statements

var divide = (x, y) => {

if(y != 0) {

return x / y;

}

}

alert(divide(10, 2)); // 5

## No parameter, single statement

var hello = () => alert('Hello World!');

hello(); // Hello World!

## Example 3

function Person(nickname, country) {

this.nickname = nickname;

this.country = country;

this.getInfo = function() {

// Outer function context (Person object)

return () => {

// Inner function context (Person object)

alert(this.constructor.name); // Person

alert(`Hi, I'm ${this.nickname} from ${this.country}`);

};

}

}

let p = new Person('Rick', 'Argentina');

let printInfo = p.getInfo();

printInfo(); // Hi, I'm Rick from Argentina

#### SPREAD OPERATOR

The spread operator, which is also denoted by (...),

// performs the exact opposite function of the rest operator.

//The spread operator spreads out (i.e. splits up) an array and passes the

// values into the specified function, as shown in the following example:

var numbers=[1,2,3,4,5];

console.log(Math.max(numbers)); //NAN Why?? Expects list of arguments not an array.

Modify the code as

var numbers=[1,2,3,4,5];

console.log(Math.max(...numbers)); //5 as result

Example to understand spread operator

var numbers=[1,2,3,4,5];

console.log(...numbers);//read the array elements and spilit in to numbers.

console.log(Math.max(...numbers));

#### FOR OF LOOP

//The new for...of loop allows us to iterate over arrays or other iterable objects

//the code inside the loop is executed for each element of the iterable object

let testResult=[1,2,3,4,5];

for(let testresult of testResult){

console.log(testresult);

}

for(let letter of letters) {

console.log(letter); // a,b,c,d,e,f

}

// Iterating over string

let greet = "Hello World!";

for(let character of greet) {

console.log(character); // H,e,l,l,o, ,W,o,r,l,d,!

}

//The for...of loop doesn't work with objects because they are not iterable.

//If you want to iterate over the properties of an object you can use the for-in loop.

--------------------------------------------------------------------------------

#### DESTRUCTURING ARRAYS

means removing some elements from array

let numbers=[1,2,3,4,5];

let [a,b]=numbers;//Take the copy of array

console.log(a);

console.log(b);

console.log(numbers);//original copy is available

let numbers=[1,2];

let [a,b,c,d]=numbers;

console.log(a);

console.log(d); //undefined becoz of the varible is not having any value

console.log(numbers);

let numbers=[1,2,3,4,5];

let [a,...b]=numbers;

console.log(b);

let numbers=[1,2,"3"];

let[a,b,c,d='Default']=numbers;

console.log(d);

swaping numbers destructure is used

let a=10;

let b=20;

[b,a]=[a,b];

console.log(a);

console.log(b);

Missing the elements by making ,

let numbers=[1,2,3];

let[a,,c]=numbers;

console.log(a);

console.log(c);

Destructuring Objects

pulling the objects data individually and printing it

let obj={name:'Ram', age:27 };

let{name,age}=obj

console.log(name);

console.log(age);

let obj={name:'Ram', age:27, greet:function(){ console.log('Hello here!!');

}

};

let{name,age,greet:hello}=obj;

console.log(name);

console.log(age);

hello();

#### TEMPLATE LITERALS

* Template literals provide an easy and clean way create multi-line strings and perform string interpolation.
* Template literals are created using back-tick (` `) (grave accent) character instead of the usual double or single quotes.
* Variables or expressions can be placed inside the string using the ${...} syntax.

**// Simple multi-line string**

let str = `The quick brown fox

jumps over the lazy dog.`;

// String with embedded variables and expression

let a = 10;

let b = 20;

let result = `The sum of ${a} and ${b} is ${a+b}.`;

console.log(result); // The sum of 10 and 20 is 30.

**REST PARAMETERS**

* ES6 introduces rest parameters that allow us to pass an arbitrary number of parameters to a function in the form of an array.
* This is particularly helpful in situations when you want to pass parameters to a function but you have no idea how many you will need.
* A rest parameter is specified by prefixing a named parameter with rest operator (...) i.e. three dots.
* Rest parameter can only be the last one in the list of parameters,

and there can only be one rest parameter. Take a look at the following example, to see how it works:

**Example 1:**

function sortNames(...names) {

return names.sort();

}

alert(sortNames("Sarah", "Harry", "Peter")); // Harry,Peter,Sarah

alert(sortNames("Tony", "Ben", "Rick", "Jos")); // John,Jos,Rick,Tony

**Example 2:**

function myFunction(a, b, ...args) {

return args;

}

alert(myFunction(1, 2, 3, 4, 5)); // 3,4,5

alert(myFunction(-7, 5, 0, -2, 4.5, 1, 3)); // 0,-2,4.5,1,3

**Example 3:**

**default values to the function parameters.**

function sayHello(name='World') {

return `Hello ${name}!`;

}

console.log(sayHello()); // Hello World!

console.log(sayHello('John')); // Hello John!