**JavaScript Notes**

**History of JS :**

JavaScript was first known as Live Script, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name Live Script. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

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

**What is JavaScript :**

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

The ECMA-262 Specification defined a standard version of the core JavaScript language.

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

**How to Add JS codes into HTML Files :**

Script in <head>...</head> and <body>...</body> section.

Script in an external file and then include in <head>...</head> section.

<html>

<head>

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

.... codes....

</script>

</head>

<body>

<script>

.... codes....

</script>

</body>

</html>

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

**Different Ways to Print Output Data:**

1. document.write("Hello World");
2. window.alert("This is Alert Box");
3. console.log("Hi Coders..");

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

JavaScript Reserved Words :

A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| abstract | boolean | break | byte | case | catch |
| char | class | const | continue | debugger | default |
| delete | do | double | else | enum | export |
| extends | false | final | finally | float | for |
| function | goto | if | implements | import | in |
| Instanceof | int | interface | long | native | new |
| null | package | private | protected | public | return |
| short | static | super | switch | synchronized | this |
| throw | throws | transient | true | try | typeof |
| var | void | volatile | while | with |  |

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

**Variables are Containers for Storing Data :**

1. The var keyword was used in all JavaScript code from 1995 to 2015
2. The let and const keywords were added to JavaScript in 2015.
3. The var keyword should only be used in code written for older browsers.

**JavaScript Variables can be declared in 3 ways:**

Using var Ex : var name = 'JS';

Using let Ex : let age = 25 ;

Using const Ex : const city = "Chennai";

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

**Data Types :**

**There are Two types available in JS**

**Primitive Types :**

1.String Ex : var name = 'imran'

2.Integer Ex : let x = 20

3.Boolean Ex : isAlive = true / false

4.Undefined Ex : var name = undefined;

5.Null Ex : let age = null ;

**Reference Types :**

1.Object Ex : let myObj = {}

2.Array Ex : let myArr = []

3.Function Ex : () => {}

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

**Basic String Methods :**

Javascript strings are primitive and immutable

All string methods produces a new string without altering the original string.

String length String charAt() String charCodeAt()

String at() String [ ] String slice()

String substring() String substr() String split()

String toLowerCase() String concat() String trim()

String trimStart() String trimEnd() String padStart()

String padEnd() String repeat() String replace()

String replaceAll() String toUpperCase()

**Search Methods :**

String indexOf() String lastIndexOf() String search()

String match() String matchAll() String includes()

String startsWith() String endsWith()

**Template Strings / Template Literals :**

use back-ticks (``) rather than the quotes ("") to define a string:

**Example :**

let firstname = "Java";

let lastname = "Script";

let fullname = `YOUR FULL NAME : ${firstname + lastname}`;

console.log(fullname);

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

**JavaScript Array Methods :**

Array length Array toString() Array at()

Array join() Array pop() Array push()

Array shift() Array unshift() Array delete()

Array concat() Array copyWithin() Array flat()

Array splice() Array toSpliced() Array slice()

**Search Methods :**

Array indexOf() Array lastIndexOf() Array includes()

Array find() Array findIndex() Array findLast()

Array findLastIndex()

**Sort Methods :**

Array sort() Array reverse() Array toSorted()

Array toReversed()

**Iteration Methods :**

Array forEach Array map() Array flatMap()

Array filter() Array reduce() Array every()

Array some() Array from() Array keys()

Array entries() Array with()

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

**What is an Operator?**

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and ‘+’ is called the operator.

**JavaScript supports the following types of operators.**

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Assignment Operators
* Conditional (or ternary) Operators
* Type Operators

**1.Arithmetic Operators :**

Assume variable A holds 10 and variable B holds 20, then:

1. + (Addition) Adds two operands Ex: A + B will give 30
2. - (Subtraction) Subtracts the second operand from the first Ex: A - B will give -10
3. \* (Multiplication) Multiply both operands Ex: A \* B will give 200
4. / (Division) Divide the numerator by the denominator Ex: B / A will give 2
5. % (Modulus) Outputs the remainder of an integer division Ex: B % A will give 0
6. ++ (Increment) Increases an integer value by one Ex: A++ will give 11

7. -- (Decrement) Decreases an integer value by one Ex: A-- will give 9

**2.Comparison Operators :**

Assume variable A holds 20 and variable B holds 30, then:

1. == (Equal)

Checks if the value of two operands are equal or not, if yes, then the condition becomes true.

Ex: (A == B) is not true.

2. != (Not Equal)

Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true.

Ex: (A != B) is true.

3. > (Greater than)

Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true.

Ex: (A > B) is not true.

4. < (Less than)

Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true.

Ex: (A < B) is true.

5. >= (Greater than or Equal to)

Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true.

Ex: (A >= B) is not true.

6. <= (Less than or Equal to)

Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true.

Ex: (A <= B) is true.

**3.Logical Operators :**

Assume variable A holds 10 and variable B holds 20, then:

1. && (Logical AND)

If both the operands are non-zero, then the condition becomes true.

Ex: (A && B) is true.

2. || (Logical OR)

If any of the two operands are non-zero, then the condition becomes true.

Ex: (A || B) is true.

3. ! (Logical NOT)

Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false.

Ex: ! (A && B) is false.

**4.Bitwise Operators :**

Assume variable A holds 2 and variable B holds 3, then:

1. & (Bitwise AND)

It performs a Boolean AND operation on each bit of its integer arguments.

Ex: (A & B) is 2.

2. | (BitWise OR)

It performs a Boolean OR operation on each bit of its integer arguments.

Ex: (A | B) is 3.

3. ^ (Bitwise XOR)

It performs a Boolean exclusive OR operation on each bit of its integer arguments.

Exclusive OR means that either operand one is true or operand two is true, but not both.

Ex: (A ^ B) is 1.

4. ~ (Bitwise Not)

It is a unary operator and operates by reversing all the bits in the operand.

Ex: (~B) is -4.

5. << (Left Shift) It moves all the bits in its first operand to the left by the number of places specified in the second operand.

New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying it by 2,

shifting two positions is equivalent to multiplying by 4, and so on.

Ex: (A << 1) is 4.

6. >> (Right Shift) Binary Right Shift Operator. The left operand’s value is moved right by the number of bits specified by the right operand.

Ex: (A >> 1) is 1.

7. >>> (Right shift with Zero) This operator is just like the >> operator, except that the bits shifted in on the left are always zero.

Ex: (A >>> 1) is 1.

**5.Assignment Operators :**

1. **=** (Simple Assignment )

Assigns values from the right side operand to the left side operand

Ex: C = A + B will assign the value of A + B into C

2. **+=** (Add and Assignment)

It adds the right operand to the left operand and assigns the result to the left operand.

Ex: C += A is equivalent to C = C + A

3. **-=** (Subtract and Assignment)

It subtracts the right operand from the left operand and assigns the result to the left operand.

Ex: C -= A is equivalent to C = C - A

4. **\*=** (Multiply and Assignment)

It multiplies the right operand with the left operand and assigns the result to the left operand.

Ex: C \*= A is equivalent to C = C \* A

5. **/=** (Divide and Assignment)

It divides the left operand with the right operand and assigns the result to the left operand.

Ex: C /= A is equivalent to C = C / A

6. **%=** (Modules and Assignment)

It takes modulus using two operands and assigns the result to the left operand.

Ex: C %= A is equivalent to C = C % A

**6.Conditional Operator :**

1. **? :** (Conditional )

If Condition is true? Then value X : Otherwise value Y

Ex :

let age = 20;

(age == 20) ? 'Age is 20' : 'Age is Not Equal to 20 ' ;

**7.Type Operator :**

1. ( typeof ) It returns the type of a variables

2. ( Instanceof ) It returns true if an object is an instance of a specified object type

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

**JavaScript Math Methods :**

1. abs(x) Returns the absolute value of x
2. acos(x) Returns the arccosine of x, in radians
3. acosh(x) Returns the hyperbolic arccosine of x
4. asin(x) Returns the arcsine of x, in radians
5. asinh(x) Returns the hyperbolic arcsine of x
6. atan(x) Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
7. atan2(y, x) Returns the arctangent of the quotient of its arguments
8. atanh(x) Returns the hyperbolic arctangent of x
9. cbrt(x) Returns the cubic root of x
10. ceil(x) Returns x, rounded upwards to the nearest integer
11. cos(x) Returns the cosine of x (x is in radians)
12. cosh(x) Returns the hyperbolic cosine of x
13. exp(x) Returns the value of Ex
14. floor(x) Returns x, rounded downwards to the nearest integer
15. log(x) Returns the natural logarithm (base E) of x
16. max(x, y, z, ..., n) Returns the number with the highest value
17. min(x, y, z, ..., n) Returns the number with the lowest value
18. pow(x, y) Returns the value of x to the power of y
19. random() Returns a random number between 0 and 1
20. round(x) Rounds x to the nearest integer
21. sign(x) Returns if x is negative, null or positive (-1, 0, 1)
22. sin(x) Returns the sine of x (x is in radians)
23. sinh(x) Returns the hyperbolic sine of x
24. sqrt(x) Returns the square root of x
25. tan(x) Returns the tangent of an angle
26. tanh(x) Returns the hyperbolic tangent of a number
27. trunc(x) Returns the integer part of a number (x)

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

**JavaScript if, else, and else if :**

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

**if** (condition) {

// block of code to be executed if the condition is true}

Use the else statement to specify a block of code to be executed if the condition is false.

**if** (condition) {

// block of code to be executed if the condition is true}

**else** {

// block of code to be executed if the condition is false}

Use the else if statement to specify a new condition if the first condition is false.

**if** (condition1) {

// block of code to be executed if condition1 is true}

**else if** (condition2) {

// block of code to be executed if the condition1 is false and condition2 is true}

**else** {

// block of code to be executed if the condition1 is false and condition2 is false}

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

**JavaScript Switch Statement**

Use the switch statement to select one of many code blocks to be executed.The break Keyword:

When JavaScript reaches a break keyword, it breaks out of the switch block.

This will stop the execution inside the switch block.

It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.-The default keyword specifies the code to run if there is no case match:Syntax :

<p id="demo"></p>

<script>

let text;

switch (new Date().getDay()) {

case 6:

text = "Today is Saturday";

break;

case 0:

text = "Today is Sunday";

break;

default:

text = "Looking forward to the Weekend";

}

document.getElementById("demo").innerHTML = text;

</script>

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

**JavaScript Loops**

**Different Kinds of Loops :**

1. for - loops through a block of code a number of times
2. for/in - loops through the properties of an object
3. for/of - loops through the values of an iterable object
4. while - loops through a block of code while a specified condition is true
5. do/while - also loops through a block of code while a specified condition is true

**The For Loop :**

**The for statement creates a loop with 3 optional expressions:**

for (expression 1; expression 2; expression 3) {

// code block to be executed

}

Expression 1 is executed (one time) before the execution of the code block.

Expression 2 defines the condition for executing the code block.

Expression 3 is executed (every time) after the code block has been executed.

<script>

let text = "";

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

text += "The number is " + i + "<br>";

}

console.log(text);

</script>

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

**The For In Loop**

The JavaScript for in statement loops through the properties of an Object:

<p id="demo"></p>

<script>

const person = {fname:"John", lname:"Doe", age:25};

let txt = "";

for (let x in person) {

txt += person[x] + " ";

}

document.getElementById("demo").innerHTML = txt;

</script>

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

**JavaScript For Of Loop**

The for of statement loops through the values of any iterable object:

<p id="demo"></p>

<script>

const cars = ["BMW", "Volvo", "Mini"];

let text = "";

for (let x of cars) {

text += x + "<br>";

}

document.getElementById("demo").innerHTML = text;

</script>

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

**JavaScript While Loop**

<p id="demo"></p>

<script>

let text = "";

let i = 0;

while (i < 10) {

text += "<br>The number is " + i;

i++;

}

document.getElementById("demo").innerHTML = text;

</script>

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

**JavaScript Do While Loop**

<p id="demo"></p>

<script>

let text = ""

let i = 0;

do {

text += "<br>The number is " + i;

i++;

}

while (i < 10);

document.getElementById("demo").innerHTML = text;

</script>

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

The **continue** statement (with or without a label reference) can only be used to skip one loop iteration.

The **break** statement, without a label reference, can only be used to jump out of a loop or a switch.

for (let i = 0; i < 10; i++) {

if (i === 3) { **break**; / **continue**; }

text += "The number is " + i + "<br>";}

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

**JavaScript Functions**

JavaScript functions are defined with the function keyword.You can use a function declaration or a function expression.A JavaScript function is a block of code designed to perform a particular task.A JavaScript function is executed when "something" invokes it (calls it).

**Syntax: General function**

function myFun(a,b){

return a\*b}let result = myFun(10,20);console.log(result)

**Syntax : Arrow Function / Function Expression**

const myfunc = (a,b) => {

return a+b}let res = myfunc(10,20);console.log(res)

**Syntax : Anonymous Function**

const addNumbers = function(a,b){

return a+b;}let result = addNumbers(100,150)console.log(result);

**Syntax : Immediatly Invoked Function**

(

function(name){

document.write('Hi my Name is ',name);

}

)('Javascript')

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

**JavaScript Class :**

JavaScript Classes are templates for JavaScript Objects.Use the keyword 'class' to create a class.Always add a method named constructor():

<script>

**//Declare class**

class employee{

**//initializing an object**

constructor(id,name){

this.id = id;

this.name = name;

}

**//Declaring Method**

detailfunc(){

document.write(this.id +" - "+ this.name+".<br>");

}

}

**//Passing object to a variable**

let e1 = new employee(100,'Javascript');

let e2 = new employee(101,'JS Classes');

e1.detailfunc();

e2.detailfunc();

</script>

**JavaScript Class Inheritance**

To create a class inheritance, use the extends keyword.A class created with a class inheritance inherits all the methods from another class:

<script>

class Car {

constructor(brand) {

this.carname = brand;

}

present() {

return 'I have a ' + this.carname;

}

}

class Model extends Car {

constructor(brand, mod) {

super(brand);

this.model = mod;

}

show() {

return this.present() + ', it is a ' + this.model;

}

}

let myCar = new Model("Ford", "Mustang");

document.getElementById("demo").innerHTML = myCar.show();</script>

The **super**() method refers to the parent class.By calling the super() method in the constructor method, we call the parent's constructor method and gets access to the parent's properties and methods.

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

**JavaScript Asynchronous Programs**

**JavaScript Callbacks :**

A callback is a function passed as an argument to another function,This technique allows a function to call another function,A callback function can run after another function has finished.

<p id="demo"></p>

<script>

function myDisplayer(some) {

document.getElementById("demo").innerHTML = some;

}

function myFirst() {

myDisplayer("Hello");

}

function mySecond() {

myDisplayer("Goodbye");

}

myFirst();

mySecond();

</script>

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

**JavaScript Promises :**

Promises are used for asynchronous programming in JavaScript. They represent the eventual completion (or failure) of an asynchronous operation and allow chaining of operations using .then() and .catch().

A JavaScript Promise object can be:

1. Pending
2. Fulfilled

3.Rejected

<p id="demo"></p>

<script>

function myDisplayer(some) {

document.getElementById("demo").innerHTML = some;

}

let myPromise = new Promise(function(myResolve, myReject) {

let x = 0;

// some code (try to change x to 5)

if (x == 0) {

myResolve("OK");

} else {

myReject("Error");

}});

myPromise.then(

function(value) {myDisplayer(value);},

function(error) {myDisplayer(error);});

</script>

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

**JavaScript Async :**

The async/await syntax is a modern approach to handle asynchronous operations. It allows writing asynchronous code in a more synchronous-like manner, making it easier to read and maintain.

<h2 id="demo"></h2>

<script>

async function myDisplay() {

let myPromise = new Promise(function(resolve, reject) {

resolve("I love You !!");

});

document.getElementById("demo").innerHTML = await myPromise;

}

myDisplay();

</script>

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