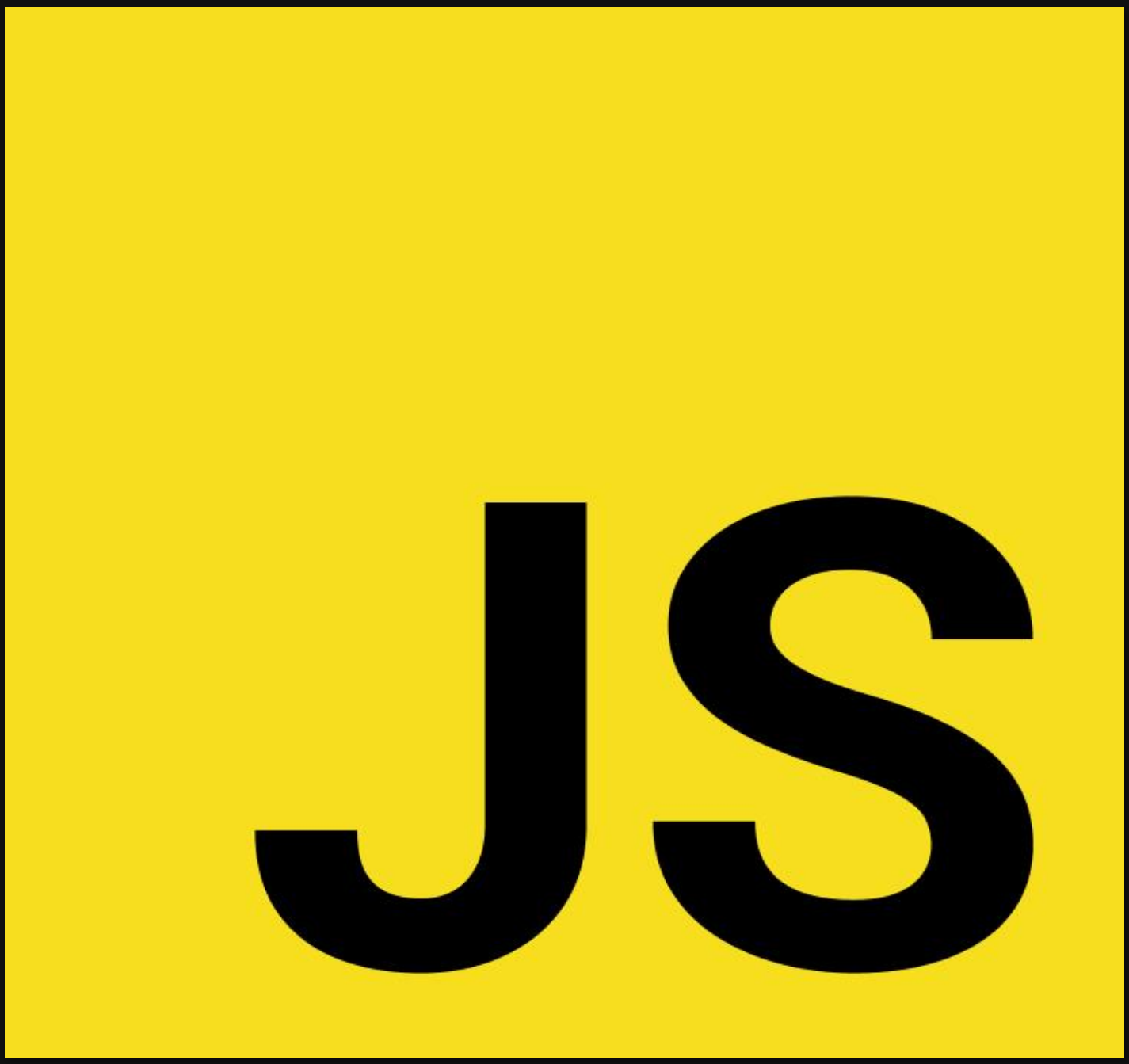


Js Note by: Nabin Acharya

Contents

What is JavaScript?

JavaScript is a high-level, dynamic, and interpreted programming language that is primarily used for building interactive and dynamic web pages. It is one of the core technologies of web development, along with HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets). JavaScript allows developers to add interactivity, manipulate the Document Object Model (DOM), and respond to user actions within a web browser.

Uses of JavaScript?

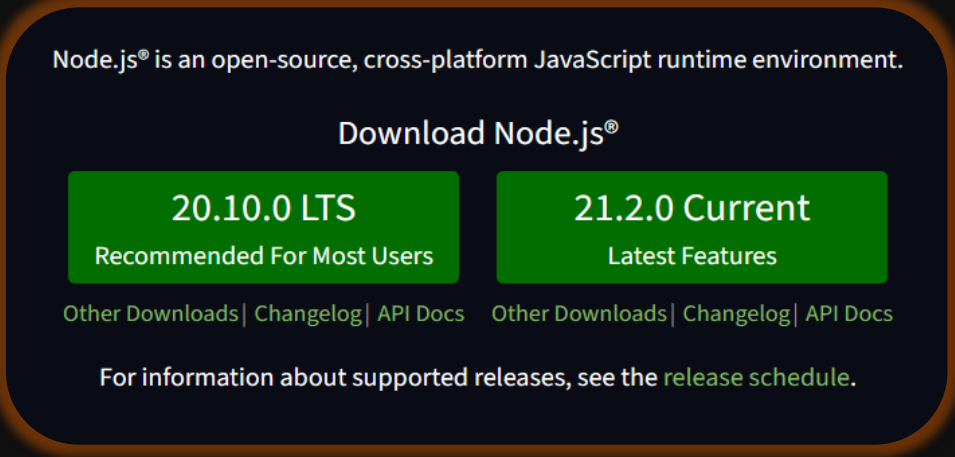
1. Clint-side Scripting.
2. Web Development.
3. Dom Manipulation.
4. Browser APIs.
5. Server-side Development(Node.js).
6. Animation and Effects.
7. Mobile App Development.
8. Desktop App Development(Electron).

Node.js JavaScript Setup?

Click the orange button to download node js.

Then you can download LTS(long-term-support) version.

Then run the .exe file and click next next next and install.installation is complete.



What is Variables?

Variables are the container to store different kind of data like(string, number, Boolean, ---etc).

How to declare variables in JavaScript?

There are three way to declare variables in javaScript:

1. Var
2. Let
3. Const

What is difference between var, let, and const?

1. var

   var is used to declare those variables which may/will change overtime during the execution of program.

   ```
   Var x;
   X=5;
   ```

2. Let

   Simillar to var let is used to declare those variables which may/will change overtime during the execution of program.

   ```
   Let y;
   Y=10;
   ```

3. Const

   const is used to declare those variables which will not/can't change overtime during the execution of program.

   Variable redecleration is not possible in const.

   ```
   Const PIE=3.14;
   ```

Different between let and var

Var	Let
Var is function scope	Let is block scope

Scope

```
Understanding of the scope
    function greet() {
        var firstname ="Ram"
        console.log(firstname)
    }
    console.log(firstname)
    Block scope
    if(true){
```

```
        const lastname ="Sita"
        console.log(lastname)

    }
    console.log(lastname)
```

What is Hoisting?

Hoisting is the default behaviour of javaScript that varise "Value decleration with var" and "Function decleration"to the top of the scop not the value.

```
=> Var is Hoisted
=> Let is not Hoisted
=> Const is not Hoisted
```

Note:

JavaScript is backward compatible language.

Data Types in JavaScript?

In javaScript, datatypes refer to what type/kind of value is/can be hold by a variable

JavaScript is dynamically typed programming language, which means the value hold by a variable can be changed into another datatype during runtime.

There are two major classification of datatypes in javaScript. they are:

Primitive Datatypes.

1. String
2. Number
3. Bigint
4. Boolean
5. Undefined
6. Null
7. Symbol

Non-premitive datatype.

a. Array
b. Object

Operators in JavaScript?

Operators are pre-Defined sign and symbols that performs some predefine actions on the operands /value.

1. Arithmetic Operators
2. Assignment Operators(=, +=, -=)
3. Comparison(Relational)Operator
4. String Operators
5. Logical Operators
6. Bitwise Operators
7. Ternary Operators
8. Type Operators

Arithematic Operators

1. Addition Operator (+)
2. Substraction Operator (-)
3. Multiplication Operator(*)
4. Division Operator(/)
5. Moduli Operator(%)
6. Exponential Operator(**)

Type Coercion

=> It is the defult behaviour of javaScript that change one datatype to another datatype fourcefully to excute operation succrssfully.

Type Conversion

=> It is the process of converting one dataType to another dataType by developer's will to execute the operation as expected

String to number conversion

ParseInt

ParseFloat

PRE ES6 METHODS

Assignment Operators

```
let x = 5; //i (=) is assignment operator
It assign the value in x
-------------------------------------
ii.Addition Assignment(+=)
x += 7; // equivalent to x=x+7.
```

```
---------------------------------------
iii.Subtraction Assignment(-=)
x-=4; //x=x-4
---------------------------------------
iv.Division Assignment(/=)
x/=4; //x=x/4
---------------------------------------
v.Multplication Assignment(*=)
x*=4; //x=x*4
---------------------------------------
vi.Modulo Assignment(%=)
x%=4; //x=x%4
---------------------------------------
vii.Exponential Assignment(**=)
x**=4; //x=x**4
```

Comparision Operators

1. Greater Than (>)
2. Less Than(<)
3. Greater than Equals to(GTE){>=}
4. Less than Equals to(LTE){>=}
5. (Not Equals to) !=
6. Equals to (Loose) (==)
7. equal to(Strict) (===)

Logical Operator

```
These are the operator which is used to combine value and manipulate
them.
i.AND Operator (&&)
/**It returens true only if all of the values are True. */
ii.or Operator (||)
/**It returens true if one of the value is true. */
iii.Not Operator (!)
/**It is one of the uniary Operator that returns false if the value is
true and vice versa */
```

Ternary Operator?

Ternary (Conditional)Operator: It is the Shorthand of IF Else Statement.since it is an operator it can be stored in the variable.

Syntax:

Condition ? IfTrue Exexuted : IfFalse Exexuted

Condition? Code to be executed if the condition is True: Code to be executed if the condition is False

```
let age = 19
age >= 18 ? console.log("You can Vote") : console.log("You cannot Vote")
```

Nullish Coelscing Operator

In this Operator if LHS value is null or undefined then only RHS will be exexuted.

```
a ?? console.log(`Thus executed because there's null or undefine value
is LHD\n`)
?? console.log("run this is nalll")
```

Unary Operator

1. NOT Operator (!)
2. typeof operator

```
console.log(typeof 42);
// Expected output: "number"

console.log(typeof 'blubber');
// Expected output: "string"

console.log(typeof true);
// Expected output: "boolean"

console.log(typeof undeclaredVariable);
// Expected output: "undefined"
```

3. Unary +
4. Unary -
5. Spread Operator (...)
6. Increment Decrement Operator(++, --)

Strings in JavaScript?

A string is a sequence of characters and can contain letters, numbers, symbols and even spaces.

Example:

Let name=”Nabin”

Const name=’Nabin Achary’

```
//Way to Wright String
/**
 * 1. ''
 * 2.""
 * 3.``
 */
```

JavaScript String Methods?

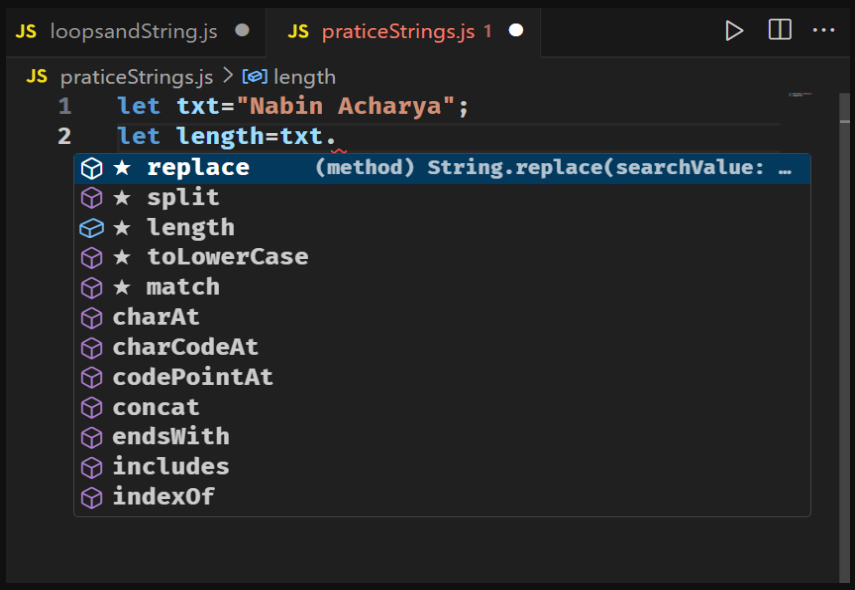
String length
String slice()
String substring()
String substr()
String replace()
String replaceAll()
String toUpperCase()
String toLowerCase()
String concat()
String trim()
String trimStart()
String trimEnd()
String padStart()
String padEnd()
String charAt()
String charCodeAt()
String split()

Example:

```
Let txt=”Nabin Acharya”;
Let length= txt.length;
let length=txt.toLocaleUpperCase();
```

Split a string into substrings using the specified separator and return them as an array.

```
let length=txt.split();
```



```
let txt="   Nabin Acharya   ";
let length=txt.trim();
console.log(length);

let txt="0123456";
let str=txt.slice(1,5);
console.log(str);

let txt="Nabin";
let txt2=" Acharya";
let str=txt.concat(txt2);
console.log(str);

let txt="Nabin";
let length=txt.replace("N","A");
console.log(length);

let txt="LearnJavaScript";
let length=txt.charAt(5);
console.log(length);
```

Templet literals in javaScript?

```
console.log("He's` nabin ")
console.log(`hdbshjsbhj" du`)

let name = "Ram Thapa"
let address = "KTM"
let age = 19

//console.log("I am " + nam + "," + "I am live in" + address + "I am" +
age + "Years Old")
console.log(`I am, ${name}, I live in ${address} and I am ${age} Years
old`)
```

Conditional Statement (if-else)

1. If

   if (*condition*) {
   // block of code to be executed if the condition is true
   }

   if the condition is true block statement will exexuted

   Example:

   Let votersAge=20

```
If(votersAge<18){
        Console.log("You are Eligible to vote"); }
```

2. If-else
   If-else has to block if and else. If the condition is true if block will executed else, else block will executed

   Example: let's take the same example

```
Let votersAge=20
If(votersAge<18){
        Console.log("You are Eligible to vote");
}else{
Console.log("You are  not Eligible to vote"); }
```

3. switch

The switch statement is used to perform different actions based on different conditions.

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

Break and Continue in javaScript

Break is used to come out of loop or any iteration

Continue is used to skip one iteration on loop

Loop in JavaScript

JavaScript Supports different kinds of loops

1. For-loop
   The for statement creates a loop with 3 optional expressions

```
for (expression 1; expression 2; expression 3) {
  // code block to be executed }
```

2. While-loop
   while (condition) {

   // code block to be executed

   }

   If the condition is true while-block will executed

   To stop loop we can also use break; in loop

3. Do-while-loop
   Do-while is similar to while but while will not if condition is false(or not satisfied)

   But in do-while it will runs atlest once.

   do {

   // code block to be executed

   }

   while (condition);

4. For-in-loop

for-in loop is used for object

```
lets learn how

let student={
    name:"Rahul kumar",
    age:25,
    cgpa:7.5,
    isPass:true
};

for (const key in student) {
    console.log("key=",key,"value=",student[key]);
}
```

5. For-of-loop

for-of loop is used for arrayand strings

```
let str="NabinAcharya";
for(let i of str)
{
    console.log(i);
}
```

Arrays in JavaScript

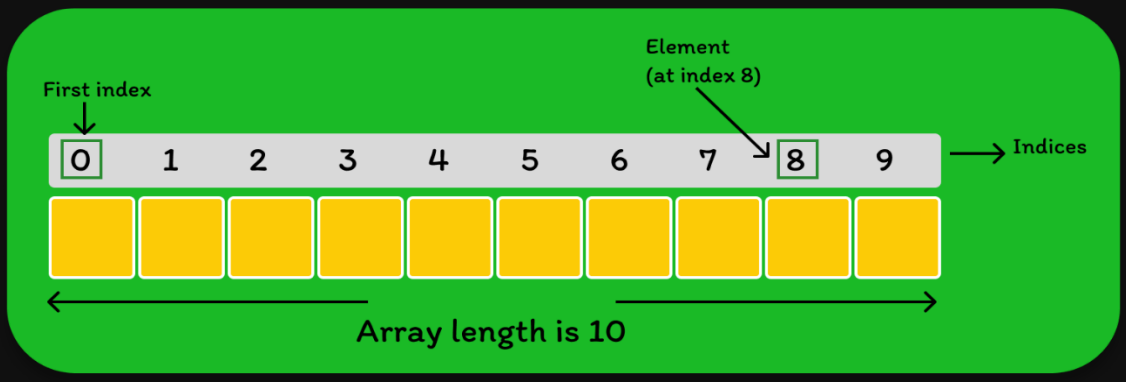
Arrays are the collection of items.

It is a special non-primitive datatype. It is used to hold similar data as collection.By using array we can hold multiple data in a single variable and manipulate them.

since javascript is a dynamically typed programming language, an array can hold multiple, value of varying datatypes.

```
let marks=[89,15,68,58,25];
```

This is array of students marks




```
let friends=["Nabin", "Sairaj", "Hemanta", "Adyatan"];
console.log(friends[0]);
console.log(friends[2]);
console.log(friends[3]);
```

Looping over an Array

Loops -> iterables(strings, objects, Arrays)

Get Element by array using loop

```
for (let index = 0; index < array.length; index++) {

}
```

Print array using for loop

```
let friends=["Nabin", "Sairaj", "Hemanta", "Adyatan"];
for (let index = 0; index < friends.length; index++) {
    console.log(friends[index]);
}
```

Print array using for-of loop

```
let friends=["Nabin", "Sairaj", "Hemanta", "Adyatan"];
```

```
for (const iterator of friends) {
    console.log(iterator);
}

//pratice questions sum 5 students marks
let students=[85,65,98,65,89];
let sum=0;
for (let iterator of students) {
    console.log(iterator);
    sum +=iterator;
}
console.log(sum);
//Pratice apply 10 percent offer in the array items
let items=[250, 645, 300, 900, 50]
for (const iterator of items) {
    let discount=iterator/10;
    console.log(`Before ${iterator} After Discount${iterator-
discount}`);
}
```

Array Methods in JavaScript?

Array length	Array join()
Array toString()	Array delete()
Array pop()	Array concat()
Array push()	Array flat()
Array shift()	Array splice()
indexof()	Includes( )
Array unshift()	Array slice()

length property

```
//In JavaScript, length is a property of an array which returns the
total number of element present in array.
 console.log(fruits.length);
```

Push( ) helps to add items in the last

```
let books=["C","c++","java", "JavaScript"]
books.push("Python", “Kotlin”)
console.log(books);
```

pop( ) helps to delete from end and return

```
let books=["C","c++","java", "JavaScript", "Python"]
console.log(books);
let deletedItem=books.pop()
console.log("Deleted",deletedItem);
console.log(books);
```

toString( ) helps to convert array into string

```
let marks=[25,26,98,21,55]
const strmarks=marks.toString();
console.log(typeof(strmarks),marks);
```

concat( ) method helps to add two arrays

```
let marvel_heros=["thor","spiderman","ironman"]
let dc_heros=["superman", "batman"]
let heros=marvel_heros.concat(dc_heros);
console.log(heros);
```

Unshift( ) add to start like push( )

Shift( ) delete from start and return like pop( )

Slice( ) returns a piece of array

Slice(startindx, endindx)

```
let books=["C","c++","java", "JavaScript", "Python"]
console.log(books);
console.log(books.slice(1,3));
```

Splice( ) Changes orginal array(add, remove, replace)

Splice(startindx, delcount, newEli)

```
let arr=[1,2,3,4,5,6,7,8]
arr.splice(2,2,101,102)
//Add Element
arr.splice(2,0,101);

//Delete Element
arr.splice(3,1);

//Replace  Elements
arr.splice(3,1,101);
console.log(arr);
```

IMP Array Methods

1. Map()
   map: ->Creates a new array with the results of some operation.The value its callback returns are used to from new array.

```
arr.map(callbackFnx(value,index,array))
let nums=[67,52,39]
let newArr=nums.map((val)=>{
    return val*2;
});
console.log(newArr);
```

2. Filter()
   filter creates a new array of elements that give true for a condition/filter.

```
let arr=[1,2,3,4,5,6,7]
let evenarray=arr.filter((val)=>{
    return val%2==0;
});
console.log(evenarray);
```

3. Reduce()
   reduce performs some operations and reduces the array to a single value.it returns that single value.

```
arr=[1,2,3,4];
const output=arr.reduce((res,curr)=>{
    return res+curr;
});

console.log(output);
lets understand
res=1 and curr=2 1+2=3
res=3 and curr=3 3+3=6
res=4 and curr=6 4+6=6
output is: 10
```

Function in JavaScript

Block of code that performs a specific task, can be invoked whenever needed.

function defination

```
function name(params) { //Syntax of function
```

```
}
```

Function Call

```
functionName(); like console.log();
```

//Create a function which prints welcome on console

```
function myFunction(){
    console.log("Welcome");
}


function printFunction(msg){
    //parameter ->input
    console.log(msg);
}
// printFunction("Hello");//argument
```

Create a function which add two numbers

```
function calculator(num1,num2){
    a=num1;b=num2;
    console.log(`The sum of${a}+${b}is: ${a+b}`);
}
calculator(10,20);

function with return keyword
function func1(a,b) {
    let sum=a+b;
    return sum;
}
func1(20,30);
```

Arrow function in Js

```
Arrow function is the compact way of writing a function
const arrowsum=(a,b)=>{
    console.log(a+b);
};

arrowsum(10,20);
multiplication arrow function
const arrowmul=(x,y)=>{
    console.log(x*y);
};
arrowmul(5,10);
```

```
// we can remove {braces} also for single line
const printHello=()=>console.log("Hello");
printHello();

//pratice questions
function vowels(a) {
    for (const i of a) {
        if(i==='a' ||i==='e' ||i==='i'||i==='o'||i==='u')
        {
            console.log(`${i} is vowels`);
        }
    }

}

// vowels("hello");

let countVowles=(str)=>{
    let count=0;
    for (const i of str) {
        if(i==='a' ||i==='e' ||i==='i'||i==='o'||i==='u')
        {
            count++;
            console.log(`${i} is ${count}vowels`);

        }
    }
    return count;
};
countVowles("aeiou")
```

Array Advance Concepts

Map method in js

map: ->Creates a new array with the results of some operation.The value its callback returns are used to from new array

```
arr.map(callbackFnx(value,index,array))
let nums=[67,52,39]

let newArr=nums.map((val)=>{
    return val*2;
});
console.log(newArr);
```

Filter method in js
filter creates a new array of elements that give true for a condition/filter.

```
1st parameters    =>elements of the array
2st parameters    =>index of the array
3st parameters    =>array itself(rarely used)
let arr=[1,2,3,4,5,6,7]
let evenarray=arr.filter((val)=>{
    return val%2==0;
});

console.log(evenarray);
```

Reduce method in js
reduce performs some operations and reduces the array to a single value.it returns that single value.

```
arr=[1,2,3,4];
const output=arr.reduce((res,curr)=>{
    return res+curr;
});

console.log(output);
// lets understand
// res=1 and curr=2 1+2=3
// res=3 and curr=3 3+3=6
// res=4 and curr=6 4+6=6
//output is: 10

// find largest number
arr=[22,58,6,9,41,25,87,69]
const output=arr.reduce((res,curr)=>{
    return res>curr?res:curr;
});

console.log(output);

// pratice
```

```
let marks=[78,95,85,93,97,25,44,65]
let scoremarks=marks.filter((val)=>{
    return val>90;
});

console.log(scoremarks);
```

for Eatch loop in Arrays

arr.forEach(callBackFunction)
callbackFunction : here, it isa function to execute for each element in the array

A callback is a function passed as an argument to another function.

```
let arr=[1,2,3,4,5,6,7]
let city=["ktm", "butwal", "chitwan", "palpa"]

arr.forEach(function printval(val){
    console.log(val);
})
generally we use arrow-Function in for-each
arr.forEach((val)=>{
    console.log(val);
})


city.forEach((val,index,array)=>{
    console.log(val.toUpperCase(),index,array);
});
```

note: for-each also called higher order function/method

```
pratice question:
arr.forEach((val)=>{
    sqr=val*val;
    console.log(`Sqr of ${val} is:${sqr}`);
})

let nums=[67,52,39];
let calcSquare=(num)=>{
    console.log(num*num);
};
nums.forEach(calcSquare)
```

DOM Manipulation

Window Object

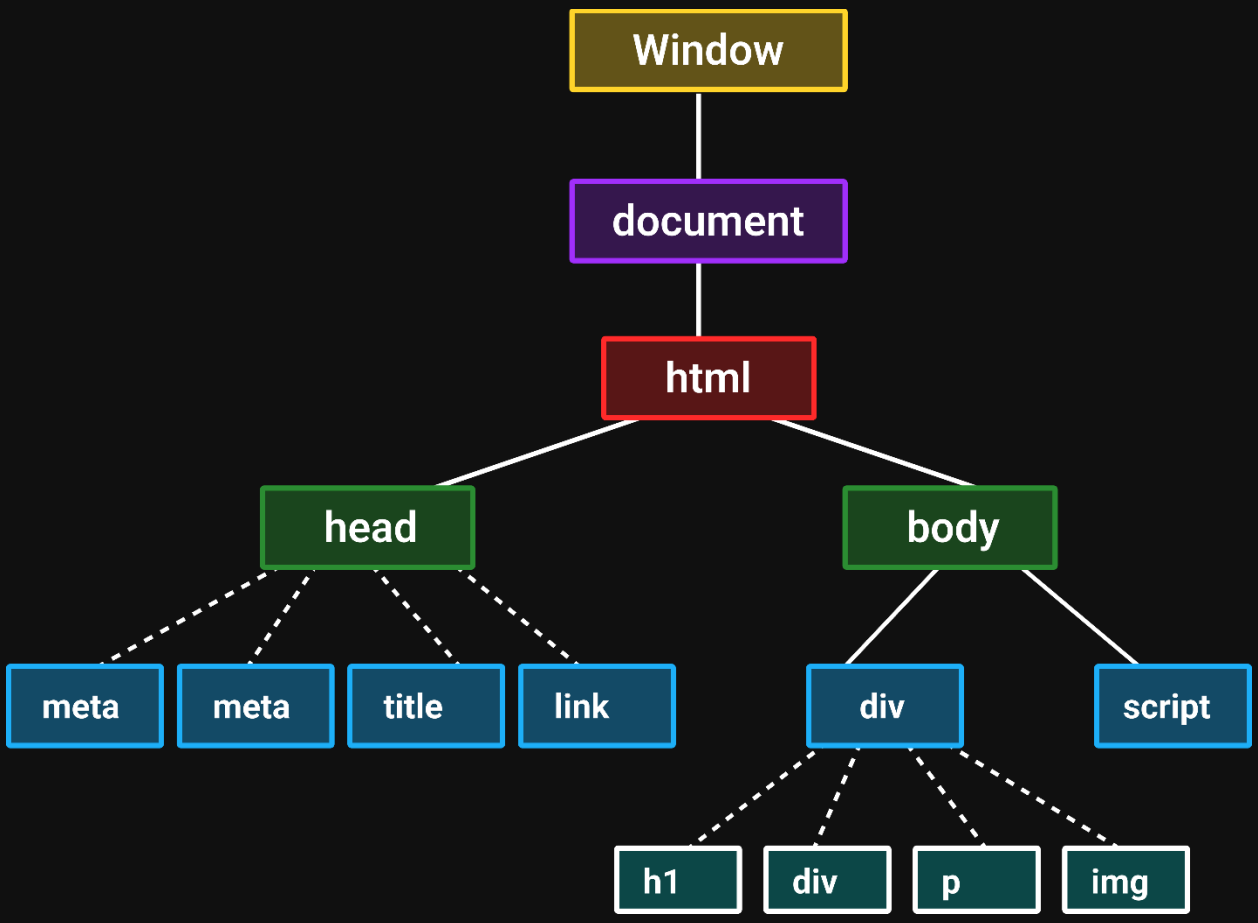
The window object represents an open window in a browser. It is browser's object (not JavaScript's)

& is automatically created by browser.

It is a global object with lots of properties and methods.

What is DOM?

When a web page is loaded, the browser creates a Document Object Model (DOM) of the page.



Select Elements using DOM

Selecting with id

Document.getElementById(“heding”);

Selecting with Class

Document.getElementByClassName(“heding2”);

Selecting with Class

Document.getElementByTagName(“h1”);

Query Selector

document.querySelector(“#myId / .myClass / tag”)

//returns first element

document.querySelectorAll(“#myId / .myClass / tag”)

//returns a NodeList

Properties

- tagName : returns tag for element nodes
- innerText : returns the text content of the element and all its children
- innerHTML : returns the plain text or HTML contents in the element
- textContent : returns textual content even for hidden elements

Some Pratice questions.

1. Create a heding h1 with text “Js Note”. Select the element using query selector and change the font color of red using javascript.

2. Create a heding h1 with text “Js Note”. Select the element using query selector and change the text “hello coders” using javascript.
3. Create 3 divs with common class name - “box”. Access them & add some unique text to each of them.

Attributes

getAttribute(attr) //to get the attribute value

setAttribute(attr, value) //to set the attribute value

Style

node.style

h1.style.color=”red”;

body.style.backgroundColor=”green”;

Insert Element

```
let el = document.createElement(“div“)
```

- node.append(el) //adds at the end of node (inside)
- node.prepend(el) //adds at the start of node (inside)
- node.before(el) //adds before the node (outside)
- node.after(el) //adds after the node (outside)

Delete Element

node.remove( ) //remove the node

practice questions
Create a new button element. Give it a text “click me”, background color of red & text color

of white.

Events in JS

The change in the state of an object is known as a event.

Events are fired to notify code of "interesting changes" that may affect code execution.

- Mouse events (click, double click etc.)
- Keyboard events (keypress, keyup, keydown)
- Form events (submit etc.)
- Print event & many more

Example:

```
Btn.onClick=()=>{
        Console.log("Btn was Clicked"); }
```

Event Object

it is a special object that has details about the event.

All event handlers have access to the Event Object's properties and methods.

It is also helps to make element move X and Y direction. Which is used to make games.

Example:

```
Node.event=( e ) =>{
        //handle here }
e.target, e.type, e.clientX, e.clientY
```

Event Listeners

node.addEventListener(event, callback)

node.removeEventListener(event, callback)

//the callback reference should be same to remove

Classes and Objects

Prototypes in JS
A javaScript object is an entity having state and behavior (properties and method).

JS objects have a special property called prototype.

We can set prototype using _ _ proto _ _

*If object & prototype have same method, object's method will be used.

Classes in JS
Class is a program-code template for creating objects.

Those objects will have some state (variables) & some behaviour (functions) inside it.

```
Class MyClass{
        Constructor( ) (….}
        myMethod( ) {….}
}
```

Let myObj=new MyClass( );

Constructor( ) method is automatically invoked by new initialized object

Inheritance in JS
inheritance is passing down properties & methods from parent class to child class.

```
Class Parent{
}
Class Child extends Parent{
}
```

*If Child & Parent have same method, child's method will be used. [Method Overriding]

Super Keyword
The super keyword is used to call the constructor of its parent class to access the parent's properties and methods.

Super(args) //calls parent's constructor

Super.parentMethod(args)

Pratice Question
Qs. You are creating a website for your college. Create a class User with 2 properties, name & email. It also has a method called viewData( ) that allows user to view website data.

Qs. Create a new class called Admin which inherits from User. Add a new method called editData to Admin that allows it to edit website data.

Error Handling in JS
Try-catch

```
Try{
    ….code
}catch ( err ){ //err is error object
    …handling error
}
```

Async Programming in JavaScript

Async await >> promise chains >> callback hell

Sync in JS

Synchronous

Synchronous means the code runs in a particular sequence of instructions given in the program. Each instruction waits for the previous instruction to complete its execution.

Asynchronous

Due to synchronous programming, sometimes imp instructions get blocked due to some previous instructions, which causes a delay in the UI. Asynchronous code execution allows to execute next instructions immediately and doesn't block the flow.

CallBacks
A callback is a function passed as an argument to another function.

Callback Hell

Callback Hell : Nested callbacks stacked below one another forming a pyramid structure. (Pyramid of Doom).

This style of programming becomes difficult to understand & manage.

Promises

Promise is for “eventual” completion of task. It is an object in JS. It is a solution to callback hell.

Let promise=new Promise ((resolve, reject) =>{ …. })

There is function with two handler resolve and reject.

*resolve & reject are callbacks provided by JS

A JavaScript Promise object can be:

Pending : the result is undefined

Resolved : the result is a value (fulfilled) resolve(result)

Rejected : the result is an error object reject(error)

*Promise has state (pending, fulfilled) & some result (result for resolve & error for reject).

.then( ) & .catch( )

Promise.then((res) =>{….})

Promise.catch((err) =>{….})

Async-Await

Async function always returns a promise.

Async function myFunc( ) {….}

await pauses the execution of its surrounding async function until the promise is settled.

IIFE : Immediately Invoked Function Expression

IIFE is a function that is called immediately as soon as it is defined.

```
(function (){
    // code
})();

(()=>{
    //code
})();

(async ()=>{
    //...
})();
```