JavaScript Beginning to Mastery Syllabus

JavaScript vs EcmaScript

Install vscode and run first program

Basics of Programming in JavaScript

Hello world program

Ecma Script( is js standard to ensure JS to run on different browser)

1. Es6 2015

Declare variable using var

More about variable

Let

Const

String Indexing

**Useful string methods**

**1. trim(),toUpperCase(), toLowerCase, slice(starting index, ending index-1)**

Template Strings

Null- type of null is object(which is a bug)

, undefined- which is not defined(type of undefined is undefined)

, BigInt for larger number(eg 12344567890000000236464)

, typeof

Booleans and Comparison Operator

Truthy- when variable is defined

and Falsy Values- null, undefined, 0

If else statement

Ternary Operator

&& || operator

Nested if else

If elseif else

Switch statement

While loop

While loop examples

For loop

For loop examples

Break and continue keyword

Do while loop

Arrays in JavaScript

Intro to arrays -(type of object)

Push-add

pop-delete

shift-deletefirst

unshift-addfirst

Primitive vs reference data types

Primitive- stores mostly data type in stack

Reference - stores the data in heap

And the pointer address stores in the stack

Clone array & spread operator-[...arrayName]

For loop

use const for creating arrays

While loop in array

For of loop-for(fruit of fruits){console.log(fruit)}

For in loop-for index

Array destructuring

**letarr=["apple","mango"];**

**let[fruit1,fruit2]=arr;**

**console.log(fruit1,fruit2);**

Objects in JavaScript

Intro to objects

Dot vs Bracket Notation

Iterate objects

Computed properties

Spread operator in objects

Object Destructuring

Objects inside Array

Nested Destructuring

Functions in JavaScript

Function declaration- direct fn

Function Expression- storing function in const or let

Arrow Functions-

Function declarations are hoisted (covered in great detail , later in this course)

Function declration are hoiseted-

But function expresiion and arrow function are not hoisted they will give refrenve error.

Function inside function-

functionmyFun2(){

console.log("function inside function");

}

functionmyFun(a){

a();

}

myFun(myFun2);

Lexical Scope-it will try to find the variable first inside its lexial scope then it goes to parent and then parent.

letmyfullName="chotu";

functionmyFun(){

letmyfullName="Ashwani";

constinsideFun=()=>{

letmyfullName="Ashwani Kumar Singh"

console.log(myfullName);

}

insideFun();

}

console.log(myFun());

Block Scope Vs Function Scope

Var is functions scope-

if(true){

varName="ashwani";

}

if(true){

console.log(Name);

}

Let and const are block scope

Default Parameters- variable default value is paased in the parameter f arguments is missing.

functionmyFun(a,b=0){

returna+b;

}

console.log(myFun(2));

Rest Parameters-inside the paramter values are passed with the help of....(3dots)

functionaddFun(...numbers){

letsum=0;

for(leti=0;i<numbers.length;i++){

sum+=numbers[i];

}

returnsum;

}

console.log(addFun(2,3,4,5));

Parameter Destructuring-Inside the parameter obj key are wriiten in braces

constobj={

name:"ashwani",

age:25

}

functionmyFun({name,age}){

console.log(name,age)

}

myFun(obj);

Very brief intro to callback functions(covered in great detail , later in the course)-

constcallBack=()=>console.log("callBack function");

functionmyFun(callBack){

callBack();

console.log("hey there..")

}

myFun(callBack);

Functions returning Functions

Very Important Array Methods

Foreach method

Map method

Filter

leta=[2,5,1,-3];

constans=a.filter((number)=>{

returnnumber>5;

})

console.log(ans);

Reduce-

leta=[2,5,1];

constans=a.reduce((acc,cSum)=>{

returnacc+cSum;

},0)

console.log(ans);

Sort

Find

constusers=

[

{name:"ash", userId:1, age:24},

{name:"aj", userId:2, age:23},

{name:"raj", userId:3, age:25}

]

constans=users.find((user)=>{

returnuser.userId==1;

})

console.log(ans);

Every-check each value of array , if all holds good then return true

Some- check any value is true , if hlods good then return true

Fill method

Splice method-

constarr=["item1", "item2", "item3","item4"];

arr.splice(2,0,"newItem");

console.log(arr);

More useful things

Iterables- array and string are iterable which has length property

Sets- its kisnd of hashet- it has unique elements, has no order

Maps- map stores key value pairs- -- key can be array , string, number and obj

Object.assign-

Optional chaining- optional chaining is used in geeting obj .value

If obj.value is not presnet we will use? to get undefined value without getting the error

Eg obj?.name

Obj?.age

Object Oriented JavaScript / Prototypal Inheritance

Methods

This keyword, Window object- this is the parent object , whenever we call this

It will call the parent obj

In Case of there is no obj it will window obj

Call , apply and bind method

functionabout(religion){

console.log(`${this.fullName}, ${this.age},${this.designation}${religion}`)}

constuser1={

fullName:'Ashwani Kumar Singh',

age:25,

designation:"SE"

}

constuser2={

fullName:'raj Kumar Singh',

age:25,

designation:"SE"

}

about.call(user1,"hindu");

Call method is used to bind the current obj

Apply method is used to pass the arguments in the array which is passed in the fundtion argumnets .

In Bind method we store the value with the help of bind

And we can call the method in future whenver we want

Some warnings

This inside arrow functions

Normal This- this is used to call current obj

Arraow this: Normal This- this is used to call one level up the current obj

Short syntax for methods

Factory functions & discuss some memory related problems

First solution to that problem

Why that solution isn’t that great

What is \_\_proto\_\_ , [[prototype]]

--proto-- is basically object which is linked with the nested object

Proto is baiscally a reference to the object which has method and properties

We can access with the help of \_\_proto\_\_

If the current object does have the method it will check with the proto linked to it.

constprintInfo={

about:function(){

return`${this.name}and ${this.age}`;

},

is18:function(){

returnthis.age>18;

}

}

constuserInfo=function(name,age){

constuser=Object.create(printInfo);

user.name=name;

user.age=age;

// user.about=printInfo.about;

// user.is18=printInfo.is18;

returnuser;

}

constuser1=(userInfo("ashwani",26));

constuser2=(userInfo("ajay",25));

console.log(user1.is18());

// console.log(user2);

// user2.user.about();

InobjectCreationwhenwehavemiliionsofuserweneed

firstceratetheobjectwithproperties

andtohavemethods

weneedtocreatesepeartemethod

whileaccesingthemethodsweneedtolinktheobjectwiththerefrenceofallthemethodsclass

What is prototype

* Prototype creates free memory which can be used to add properties and methods
* We can link the funvtion with help of
* functionName.proptype.propertyName
* Or
* functionName.proptype.methodName

* We need to specify the object with current with protype using create object method

Use prototype

New keyword

New keyword helps to create **this** empty object

Return this to the object

Object.create(Object.prototype)// this chaining will auto done by the help of new key word

functioncreateUser(name,age,designation){

this.name=name;

this.age=age;

this.designation=designation;

}

constuser1=newcreateUser("ashwani",25,"SE");

createUser.prototype.printDetails=function(){

return`user name is ${this.name}and he is ${this.designation}`;

}

createUser.prototype.is18=function(){

return` ${this.age>18}`;

}

console.log(createUser.prototype);

Constructor function with new keyword

More discussion about proto and prototype

Class keyword

* Class is used to create object
* We can vreate class with the help of constructor
* We can assign all the properties isnide constrcuror
* We can assign all methods inside the class
* -We can also create sub class
* - using extnd key word
* We can create new propertied and existing properties can be drived from parent class
* - all existiong propertie need to mention in super key word
* We can also create other method in child class

classAnimal{

constructor(name,age,food){

this.name=name,

this.age=age,

this.food=food

}

eat(){

return`${this.name}is eating ${this.food}`

}

}

classDogextendsAnimal{

constructor(name,age,speed,food){

super(name,age,food);

this.speed=speed;

}

run(){

return`${this.name}can run at ${this.speed}kmph`

}

}

constdog1= newDog("tommy",12,34,"chicken");

console.log(dog1.run());

Example using class keyword

Super keyword

Method overriding

Getters and setters

Static methods and properties

How JavaScript Works

Global Execution context

1. Compiling Phase

2. Code Execution phase

1. Global execution context

In (GEC) al the vaiable and function which is created in lexial scope will be set under global memory inside stack

And the vaiable inistialized with var keyword will be set to undefined

This and window in global execution context

Hoisting

1.Var is hoisted and it will undefined

2. Let and const is also hoiseted but they are not initialized

They will throw uninialized error(REFRENCE ERROR)

3. TDZ is the area whree the vaiable is accessed before initaliaztion

1. In case of var it will throw Refrence error not initialized
2. In case of not declaring it will throw Refrence eoor not declred

Are let and const are hoisted ? What is a reference Error ?

Function execution context

It is same as global execution context.

Scope chain and lexical environment

Intro to closures

Closure example 1

Closure Example 2

Closure Example 3

DOM Tutorial

HTML and CSS Crash course ( Around 30-40 minutes)

Async vs defer

1.before head - it will throw error as

- first it will load js

- then it will execute js

- but at the time of js exceution it will throw error

- as the html is not parsed yet

2. Before body

- first it will parse html

- then it will load js

- but it will execute js

- (it will take much time hand hence it is not efficient)

3.before head using async

- it wil parse and load js-simultaneously

- but later when loading of js finishes

- the it will stop html parsing and execute js

- so it can alos have chance to throw error

4.before head using defer

- it wil parse and load js-simultaneously

- but later when loading of js finishes

- the it will continue html parsing

And at last execute js

Using defer is the best way to load attach js file

Select elements using id- using Id(unique)

QuerySelector-node list same as css(query selctor all

textContent & innerText- text content is all hideen +visble

Inner text- only vissible

Change the styles of elements using js-style-color

Get and set attributes-place holder, href

Select multiple elements and loop through them

1. Html collections- cant use forEach loop- array like objects

2. Node list use- simplefor. For of ,for Each- array like objects

InnerHTML-the whole html inside tag

Deeply understand dom tree, root node , element nodes, text nodes

constrtNode=document.getRootNode();

constrtChild=rtNode.childNodes;

consthtml=rtChild[1];

consthtmlchild=html.childNodes;

constbody=htmlchild[2];

console.log(body.children);

ClassList-all classes of the class

Addd, remove,toggle

Add new elements to page

Create elements

Insert adjacent elements

Clone nodes

More methods to add elements on page

How to get the dimensions of the element

Intro to events

This keyword inside eventListener callback

Add events on multiple elements

Event object

How event listener works

Practice with events

Create demo project

More events

Event bubbling

Event Capturing

Event delegation

Create Project using event delegation

Asynchronous JavaScript

Is Javascript a synchronous or asynchronous programming language ?

SetTimeout()

SetTimeout() with 0 millisecond

Callback Queue

SetInterval and create little project with setInterval

Understand callbacks in general

Callbacks in asynchronous programming

Callback Hell and Pyramid of doom

Intro to promises

Microtask Queue

Function that returns promise

Promise and settimeout

Promise.resolve and more about then method

Convert nested Callbacks to flat code using promises

Intro to Ajax, HTTP Request

XHR requests

Error handling in XHR requests

XHR request Chaining

Promisifying XHR requests and chaining using then method

Fetch API

Error Handling in Fetch API

Consume Promises with async and Await

Split code into multiple files using ES6 modules.

Congratulations

Now you know javascript in Great Details

What next ?