**Summit JS+ Jansir Improvisation**

Introduction:

* We are going to follow thoroughly W3school.
* Rigorously phase by phase which has been instructed in w3school.

How to follow:

* Don’t skip any video.
* Follow the order.
* Attend the quiz
* Get in the group of summit facebook group.

#01 Introduction to JavaScript

The mother language of web. Without js wev dev is impossible. Js is a INTERACTIVE language. Means, if we as a client or use do something for ex: clicking something, that is what called interactive. Js is one of the 3 must learn language about web. Html, css and JS. Html for content, css for layout, js for behavior.

Java and js is completely diff both in concept and design. Invented by Brenda eich in 1995. Became ecma standard in 1997. ECMA-262 is the official name of the standard. Ecmascript is the official name of the language. Latest version of Ecmascript is ES2021.

Rem[js works as camel case. always]

C: document.getElementById(“Demo”).innerHTML= “hello javascript”;

here document is system. getlementbyid is a method and innerhtml is the element that has been changed to hello script. Understand, it has been assigned on the spot between the tag start and end. For ex: <p id=”demo”>Hello JavaScript</p>.

C: document.getElementById(“demo”).style.fontsize = “35px”

Here we are changing the value of css file with js command. Similliar like previous code which change the html element. Js can do both.

Rem [we cant use hifen after dot. Its system.]

C: document.getElementById(“demo”).style.display=”block”;

Above code is editing the display of a css and changing it to block.

WEHERE TO:

Js is inserted into <script></script> tag. And we can add it to any html page Or code. It can be written in <head> or <body> tag. We can add multiple script tag in one html file.

Rem [place your script tag or js code just before the ending tag of body. This is good practice. If we had thousand or millions line to executed, then it becomes slow if we don’t do this.]

We can also add JS external file. File extension is dot js. Ex: .js

C: <script src=”firstcode.js”></script>

We can also extract the file from local drive with path. Ex: “summitjs/js/text.js”

Advantage of external to speed up the html code. Readable. And cached js file can speed up the page loads.

#02 JavaScript Output

Js can display data in different ways

* innerHTML
* document.write()
* window.alert()
* console.log();

Rem [the most important display output command is innerHTML and console.log() ]

Innerhtml:

innerhtml can just push any value or assign the value as code, in between the starting and ending tag. For ex:

Previous,

<p id=”demo”></p>

Next,

<script>

Document.getElementById(“demo”).innerHTML=5+6;

</script>

Now,

The browser will show 11. Why ? cause we added and put it in between the tag of p as script code.

<p id=”demo”>11</p>.

Document.write():

Simply,

<script>

document.write(5+6);

</script>

This will simply print 11 in the p tag.

Rem [if we use document.write. then the browser delete everything and only will be there document instruction and its demanded output. Nothing else. Browser will execute all the element of html and css and even document.write code too. But we got to console and write this command then all the page will be gone and only output will be there which is given on the console command. Oke ? Now understand one thing. We understand that we can interact with js even after a loaded page. So another thing to understand is the JS is client side. Why cause can interact even after the loaded and we can change that. But not server. So it’s a client side prog lang.

Rem [document.write uses for TESTING]

Windot.alert():

This will simply put this command and its out first as a pop up box. And then will load the page.

Simple. Even we can skip the the ‘window’ keyword. We can just put alert(5+6); and the code will be executed.

Console.log():

This is uses in console. To print the output of the demand on the CONSOLE. Watch carefully. Not in a browser or box or page. On the console the output will be. This is for the coder to put some evidence to put info out.

Print():

This command is to interact with the device printing method. Js does not have power over machine of the user. Except this line to interact with printing ability.

#03 JavaScript Statement.

Statement are the instruction to the web browser. It has, values, operator, expressions, keywords etc. statement are executed 1 by 1 , line by line. Semicolon are very good practice. And we can add it even in one line of code and separated by semicolon. Java script ignore multiple space.

A good practice is to use space around operator.

Bad: let x=a+b;

Good: let x = a + b;

Both will work but good practice.

If code line is so big and takes 2nd line then pls separate the line after and operator.

#04 JavaScript Syntax

Syntax is the set of rules how js programs are constructed.

Js syntax defines 2 types of values.

* Fixed values also called LITERALS
* Variable values also called Variables.

Fixed values:

2 important rules for fixed values are

* Numbers

10.50,

10001

Numbers are written with or without decimals.

* Strings

Strings are written with double or single quotes.

“jhon doe”

‘jhon doe’

Variable is a place holder. Use to store data.

Js uses keywords var, let and const to declare variable.

Expression:

Js expression is important. Statement and expressions is not the same. And expression is a combination of values, variables and operator which computes to a SINGLE value. A statement may have many expression but expression is only 1.

Ex: 5 \* 10 [this is an expressin]

So To be a expression we have value 5, operator \* and 10 value and the evaluated value is 50.

Expression may also have variable values

Ex: x=50;

Values can be various types. Number and string for ex.

Identifiers:

Are just the names. It uses to to name variable keyword functions labels.

1st character can be LETTER, Underscore, Dollar sign

Subsequent characters may be letters, digits, underscores or dollar sign.

Numbers are not allowed as a first character.

Identifiers are CASE sensitive.

Came case is very useful in js.

Hello\_world,

helloWorld

05# JavaScript Comments

Single line: //

Multiline: /\*this is a comment \*/

06# JavaScript Variable:

var, let, const

var x = 5;

or we can write in 2 line

var x;

x=5;

#07 JavaScript LET keyword

* Let CANNOT be re declared but CAN reassign.
* Let must be declared before use
* Let have block scope.

So,

Let x= “jhon doe” [declared]

Let x = 0; [syntax error x has already been declared]

But with var we can do that

Var x = “jhon doe” ;

Var x = 5;

X will be replaces by 5.

ScopeS:

Before 2015 js has only 2 scopes global and function

After that they had introduced let and const keyword

These 2 keywords provide BLOCK scope in js

Any let declared inside a scope CANNOT be accessed from outside

Any var declared inside a scope CAN be used from outside. That’s is why var keyword can not have block scope.

Redeclaring a variable with var will be replace by the new value.

So,

Var x= 50

{

Var x = 2;

}

Console.log(x);

[above code will print x= 2]

So this problem been solved by let, which help us to redeclare a variable inside a block and will not change the outside. But REMEMBER,

Re declaring variable with let keyword in SAME block is not allowed.

Let Hoisiting:

Suppose,

carName= “volvo” [assigning Volvo as a carname]

var carName; [in 2nd line we declaring the variable]

conole.log(carName);

[above code will print volvo inspite of shuffling the line. Cause js understand and go through the whole code and Declaration will taken to UP. That the smartness of js. This is called hoisiting.]

Now there is a twist.

Var and let both do the hoisiting. But var works fine, let DOES NOT. Let just got the variable but do not put or assign any value in it. Let keeps it dead zone. It becomes temporal dead zone. Cause we know when we declare a variable in js, variable stays undefined.so in var type it stays undefined but in let hoisiing it will be dead zone.

#08 JavaScript CONST

CAN NOT

* Reassign a constant value
* Reassign a constant array
* Reassging a constant object

CAN

* Change the element of the constant array
* Change the properties of the constant object

Variable declared with const keyword in a block is not the same as outside block, same as let.

BUT, re declaring a const variable of a same name in DIFFERENT scope is allower. On the other hand same name of var, let, consta is not allowed.

Const hoisiting is just as same as LET.

#09 JavaScript Operator

Arithmetic operator:

+ is an addition operator

‘\*’ is multiplication

But, ‘\*\*’ double star is exponentiation.

Assignment operator:

X +=y [x=x+y]

X -= y [x=x-y]

X\*\*=y [x=x\*\*y]

String operator

+ operator can also used to add strings [concatenate]

Ex:

Let text1= “jhon”

Let text2 = “doe”;

Let tex3= text1 + “ ”+text2;

Result: jhon doe

[this is called concatenation and additional space in text3 will also be there. Remember that is not an empty space. That’s a intentional space.]

Ex:

Let text1= “what a nice”

Text1 += “day”

Now, some js special case.

Adding 2 numbers will return the sum

But adding a STRING + NUMBER then it will return a string

Ex:

Let x = 5+5; = 10

Let y = “5” + 6; = 56

Let z = “hello” + 5; = hello5

Comparison operator:

== equal to

===equal value and equal type

!= not equal to

!== not equal value and not equal type.

? ternary operator

Logical operator

&& logical and

|| logical or

! logical not

Type operator

typeof [returns the type of a variable]

instanceof [return true if an object is an instance of an object]

#10 JavaScript Assignment

Let x= 10;

X +=5;

Result: 15

Let x = 10

Let %=5;

Result: 0

Code: document.getElementById(“demo”).innerHTML= x;

#11 JavaScript DataType

Number,string, object and more.

Without data types a computer cannot safely solve this.

For ex:

Let x = 16 + “volvo”;

[we know js will print 16volvo but how ? how does it know that we have to add a number with string and convert it to a string. Cause JS smartly interpret the 16 into a string and add both string and print it. ]

Now ex:

Let x = 16+16+”volvo”;

[now what will be the result ? 1616volvo ? NOOOOOOO. Why ? cause js evaluates expression from left to right. So 1st 16 is a number and 2nd  16 is another number and add it. Then it finds a another string then again as previous JS interpret number as a string and add it as a string so result will be = 32volvo]

Again ex:

Let x = “volvo”+ 16+16

Result: volvo1616 [cause first left to right it gets string and add it with 16 and then again 16. All of them turned out to be string. As js prioritize string over number]

Types of Dynamic:

Js is dynamic language why cause a single variable can hold a different data types.

Let x =5;

X= 10;

X= “jansir”

You CANNOT add a double string in a string. Common sense.

For ex:

Let x =”it’s alright”; [we are using single quote in this ex]

Let x = ‘how are you “jhonny”’; [we are using double quote in ex]

Numbers types:

Extra large number can be written in scientific notation

Let y = 123e5;

Let x= 123e-5;

Boolean types:

True and false. Boolean are often used to conditional testing.

In html file for ex:

<script>

Let x = 5;

Let y = 5;

Let z = 10;

Document.getElementById(“demo”).innerHTML= (x==y) + <br> + (x==z);

</script>

[above code, in html file we are just asking if its equal or not after the = operator beside innerHTML and it will print just true and failse in 2 line cause we have also added a br tag there.]

In js file for ex; wil be as normal js code and if else check.

Array types:

Array are written with square brackest, Separated by comma, indexes are zero based.

Ex: const cars = [“saab”, “volvo”, “BMW”];

Object types:

Object are written with curly braces. Properties and its values are written in colon pairs and other properties and their values are separated by coma.

Ex: const person = {firstName:”Mehrab”, lastName:”hossian”};

Person is object , firstName/lastName is property and Mehrab/Hossain in value.

Undefined:

Let car; [value is undefined, type is undefined]

Even we can set up the value by giving it as undefined

car = undefined; [value is undefined, type is undefined] [job interview]

Empty values:

Empty values are nothing to do with UNDEFINED

So empty string has both legal value and type

Ex:

Let car = “”; [value is “”, the typeof is string]

LASTLY:

Array string number undefined is Data types

Typeof is an OPERATOR. NOT DATA TYPE. Remember that.

Js function is a block of code design to perform a particular task. To do a work many times with same code with different arugment and parameter,

Ex:

function myFuntion(p1, p2)

{

return p1 \* p2; // returns the product of p1 and p2.

}

[function defined with function keyword and then name of the function and follower by () where there will be parameter, separated by coma. And the code to be executed if the function is called is INSIDE the curly braces. ]

Ex:

Function sleep()

{

Console.log(“he is sleeping”);

}

Console.log(“he is walking”);

Sleep();

Result: 1st he is walking then 2nd he is sleeping. As we are calling the function after the 1st command of out. REMEMBER, if we don’t give first brace while we are calling the function it will not call anything. So, sleep() is the right way not sleep;

Another ex: passing parameter:

function walking(names)

{

  console.log(names + "he is walking");

}

console.log("jansir");

console.log("Sharry");

console.log("Rehan");

result will be as simple but with name different

Multiple parameter ex:

function walking(names, totalkilo)

{

  console.log(names + "he is walking" + totalkilo);

}

console.log("jansir", "10 kilo");

console.log("Sharry", "20 kilo");

console.log("Rehan", "30 kilo");

Rem= [name, totalkilo these are parameter and “jansir” “10kilo” these are argument] [job interview]

Function INVOCATION:

* When an event occurs (when user clicks)
* When it is invoked by js code by cliend
* Automatically (self invoked)

Function return:

After typing return no statement of command or code works. So return command is the last thing to write in a function

Ex:

let x = productFunction(4,10);

function productFunction(x,y)

{

  return x\*y;

}

console.log(x);

result: 40.

Another return ex:

let x = productFunction(5,10);

function productFunction(x,y)

{

  let/var z=x\*y

  return z;

}

console.log(x);

let or var in a function saving the product and returning correctly.

Another ex:

let x = productFunction(5,10);

function productFunction(x,y)

{

  console.log("hello");

}

console.log(x);

result: hello and undefined

why ? cause we are not returning the value inspite of being called. So JS is smart and pass a return value with undefined value in it.

Another ex in html script

<script>

function toCelsius(f)

{

return (5/9) \* (f-32);

}

document.getElementById("demo").innerHTML = toCelsius(77);

</script>

Another thing.

We we just call the function without braces beside it then the whole body will be returned

For ex:

function productFunction(x,y)

{

  console.log("hello");

}

let x = productFunction;

console.log(x);

result will be :

ƒ productFunction(x,y)

{

console.log("hello");

}

Why ? cause we are storing the whole function into x var, and printing it. Remember the whole function is now working as an OBJECT. Cause remember we can all other datatype in object. Slowyly we will learn

Rem = [giving braces is important now, as you have got to know]

Another example as we can call and use function output directly.

Ex:

function productmath(x,y)

{

  return x\*y;

}

console.log("the product of 2 numbers 5 and 10 is " +productmath(5,10)+ " = fifty");

let x=("the product of 2 numbers 5 and 10 is " +productmath(5,10)+ " = fifty");

console.log(x);

result will be both same of 2 conosle.log

another thing to rem = if we declare any variable with let/var in the function scope. We CAN NOT use it outside of it.

So that solves another problem, which is. We can name same variable in a different function. Simple.

#13 JavaScript Object

Ex: a car is a object, name/model/color is PROPERTIES and drive/start/break is METHOD/functionality.

Remember method and function are same in js. Function that is in the object are called method. Simple

When we declare the property and value directly in an object then the syntax is called literal syntax.

For ex:

const car = {

  model: 'Benz',

  weight: '500kg',

  color: "white",

}

We can access the proerpty of the object as this way

const car = {

  model: 'Benz',

  weight: '500kg',

  color: "white",

  drive: function(){

    console.log("the car is starting");

  } ,

  breaked: function(){

    console.log("the is car has stoped");

  }

}

console.log(car.model); // first way

console.log(car["color"]); // second way

result:

benz and white

now if we want to call the method of the object.

We just add another line

car.drive(); [will print ”the car is starting”]

THIS:

If we say this.color, will print the color of the object.

We can use this.functionName to call a another function of the same object.

const car = {

  model: 'Benz',

  weight: '500kg',

  color: "white",

  drive: function(){

    this.breaked();

    console.log("the car is starting");

  } ,

  breaked: function(){

    console.log("the is car has stoped");

  }

}

console.log(car.model); // first way

console.log(car["color"]); // second way

car.drive();

we will understand more about THIS keyword in future.

Another thing about ‘new’ keyword

DO NOT declare any variable as new , as it will count it as a object.

Let x = 5;

Let x = new 5;

Another thing,

We cannot print a variable without its initialization

function sayHi()

{

  console.log(name);

  console.log(age);

  let name = "jhon";

  var age= 50;

}

sayHi();

result: error.

So first initialaize and then it will work.

Another thing,

We can store an object in another object property.

How ?

const mouse = {

  nam: "micky",

  color: "white"

}

const cat=

{

  nam: "puchi",

  friend: mouse

}

console.log(mouse.nam); // calling obj mouse and nam property

console.log(mouse["nam"]); // same as previous just diff style

console.log(cat.friend.nam); //calling cat obj friend prop and nam prop

console.log(cat["friend"]["nam"]); // same as above line diff style

console.log(cat.friend.color); //same.

The code speaks with comment. Read it.

#14 JavaScript EVENT

HTML event are “things” that happen to html element.

When js is used in html pages, js can ‘react’ on these events.

JavaScript Event Handlers

Event handlers can be used to handle and verify user input, user actions, and browser actions:

* Things that should be done every time a page loads
* Things that should be done when the page is closed
* Action that should be performed when a user clicks a button
* Content that should be verified when a user inputs data
* And more ...

Many different methods can be used to let JavaScript work with events:

* HTML event attributes can execute JavaScript code directly
* HTML event attributes can call JavaScript functions
* You can assign your own event handler functions to HTML elements
* You can prevent events from being sent or being handled
* And more ...

#15 JavaScript STRING

Let text= “asdasdoihasdoihasiod”;

Let length= text.length;

Escape Character:

Escape character helps to print a word which is a under double or single quotation in a string

For ex:

Let text = “he is a “good” boy of dhaka”;

Here, js will give error, following code we do it.

//now supoose we need double or single quoataion inside a

//string or even a / sign how shall we do.

let text2 = "this is \"sparta\" okeay!!!!";

console.log(text2);

let text3 = "this is \'viking\' okay";

console.log(text3);

let text4= "this is \\Backslash\\ okay ";

console.log(text4);

so, \’ or \” or \\ is the answer for the problem.

BREAK a LINE

Break a line with a backslah or make a readable new line. Bur remember it’s not a new line, just a easy term to read.

Let text =”helloe \jansir” ;

Its not a preffered way. Good practice is to put a + and add the rest of the string code. As its does the same.

STRING Object:

Let x = “jansir” [this is called literal syntax]

Let x = new String(“hossain”); [this is not a literals syntax]

let x = "John";

let y = new String("John");

console.log(x==y);

== double equal FUN (only values)

fun is, even though these two values are same but type are diff, js will compare with this compare operator == and return true. But will give false if the values are diff by any character.

=== triple equal fun (values + Type)

Again above code will give this time the result FALSE. Why cause value may be same but type is not same.

REMEMBER = 1 string and 1 object can be compare and we could get true or false. but 2 object CANNOT be compared , this will always give FALSE. So object cannot be compared with any, whether its double or triple equal.

#16 JS STRING METHOD

Primitive values like “jhon doe” cannot have proerpties and method. But JS gives that advantage. And method+ prorperties are available in primitive values.cause JS treat primitive values as OBJECT when executing method and properties.

Extracting STRING PART 3 ways

* slice(start,end)
* substring(start, end)
* substr(start, length)
* // Now String Extract in 3 ways.
* // Slice method
* let text1= "Apple, orange, Mango";
* console.log(text1.slice(7,10));

this is print ora.

REM= in JS string also start like array and count from zero.

So the 7th position is O and 10th A . result= ora

Above code is cutting from 7th to 10th position and saving it in text1 variable. And this is 1 of the function of string extraction.

So text1.slice(7,10) is RETURNING a value.

// we can do it passing negetive value also.

console.log(text1.slice(-10, -6));

but remember= in this situation the count down start from the end and come to the 10th position and then goes to the 6th position. Again from the end of the string.

Now another thing to remember is that= the slice parameter always have to smaller and then bigger. Means (7,10) or (-16, -8)

But, not bigger and smaller . why cause JS start taking action from left tor right. So if we put big one first and then smaller it will give nothing just blank.

Another thing.

//we can skip the the 2nd paramiter of slice it will

//give rest of string from 1st postion

console.log(text1.slice(7));

console.log(text1.slice(-10));

Now, Substring extraction system, 2nd

Same as slice just negative parameter is NOT accepted. And rest is same

Now, substr extraction system .3rd

Is also smiliar to slice but difference is the 2nd parameter actually specifies the length of the extracted part.

let text2= "01234567891011121314";

console.log(text2.substr(7,5));

this is print 78910. And negative system works as slice as same.

Replace String Content:

let text3= "Bithi Tajrin Manna";

let text4= text3.replace("Tajrin","Jerin");

console.log(text4);

just call replace function and put 1st parameter which one you want to change and 2nd parameter which will be put on. Simple.

Bur REMEMBER= text4 is a new new string not the text3. So if we call text3 it will remain as it was. Just text4 has slight change been done by replace and it’s a new string.

Another thing to remember which is, when 1st parameter is there and if it even matches with several word in the string. JS will only replace the 1st word from the LEFT. And rest will be as it was.

So,

* replace method return new string
* replace method does not change the original string
* replace method only replace the 1st match.
* Replace method is case sensitive.

So solve this case sensitive and multiple matches prob, js has

* Text3.replace(/bithi/i, “jerin”); [so don’t put quote and forward slash with i ]
* Text3.replace(/bithi/g, “jerin”) [same as above with slight diff is g. to find multiple matches and replace.]

Case change:

//converting upper and lower case

let text5= "hewWhwheWelke";

let text6= text5.toUpperCase();

console.log(text6);

let text7= text5.toLowerCase();

console.log(text7);

Concat Method:

text = "Hello" + " " + "World!";  
text = "Hello".concat(" ", "World!");

2nd line code of concat, is like same a 1st line

Both print same.

FORMALY we learn that, STRING are Immutable. String cannot be changed, only replaced.

Immutable means cannot be muted. So string is immutable.

String trim():

let text1 = "      Hello World!      ";  
let text2 = text1.trim();

just removing the white space or unnecessary space.

charAt() method:

let text = "HELLO WORLD";  
let char = text.charAt(0);

will return H;

charCodeAt() method:

let text = "HELLO WORLD";  
let char = text.charCodeAt(0);

this is return is UNI code value of 0 position character.

Accessing property of string.

let text = "HELLO WORLD";  
let char = text[0];

this is print H.

property access look like an array but NOT array

if no character found in that position return undefined. Where, charAt() return empty string.

Converting a string to Array:

A string can be converted into an array with split() method.

3 system:

* Text.split(“,”) //split on commas
* Text.split(“ ”) //split on spaces
* Text.split(“|”) //split on pipe
* Text.split(“”) //split all string as single character
* //Converting String to array
* let text = "Hello Jansir Hello";
* console.log(text.split(""));
* console.log(text.split(" "));

#17 JavaScript STRING SEARCH

* string indexOf();

let str = "Please locate where 'locate' occurs!";  
str.indexOf("locate");

result= 7;

if nothing found then it will return -1.

* str.lastIndexOf(“please”) ; result= 7 cause it will return the last match index position.
* Str.indexOf(“please”,15): it will start searching from postion 15 and seek.
* Str.lastIndexOf(“please”,15): it will start searching form end and keep counting to 15 postion and then search. Just the opposite.

Search method

* Str.search(“please”): [same as indexOf resutl]

But ,

As both search and indexof are same result but they are not same method.

Search cannot take second start postion

Index cannot take regular expression.

String Match():

Just take as normal match just result will be the total amount of match

Let x= “hello ello kello elloeeing felloe”;

x.match(/ell/g);

simple.

#18 JavaScript String Template

Back-tics syntax

let text = `Hello World!`;

quoates Inside string

let text = `He's often called "Johnny"`;

multiline string

let text =  
`The quick  
brown fox  
jumps over  
the lazy dog`;

Variable substitution

let firstName = "John";  
let lastName = "Doe";  
let text = `Welcome ${firstName}, ${lastName}!`;

Expression substitution

let price = 10;  
let VAT = 0.25;  
  
let total = `Total: ${(price \* (1 + VAT)).toFixed(2)}`;

#19 JavaScript Number:

* Js are all floating point number. 64 fit

[52 bits value, 11 bit exponent and 1 bit sign]

* Precision

Interger number up 15 digits.

Maximum number of decimal is 17.

But, floating point arithmetic is NOT 100 accurate.

Ex:

0.2+0.1 = 0.30000000004 ;)

To solve the problem.

Multiply both number with 10 and devide it with 10. You get the actual result.

* Numeric string

Let x= 100;

Let y = “100”;

Now,

Suppose

We take.

Let x = “100”;

Let y = “10”;

Let z= x/y;

Js will result to 10. Why cause js convert string to numeric.

Same for – minus and multiplicaition \* but not +.

When plus arrives it just concatenate.

* Nan not a number.

Let x = “hello”;

Let y = “10”;

Result: if we devide it or doanything accept +, result will be not a number

console.log("Hello Number");

// let x = "10";

// let y = 10;

// let z = x/y;

// console.log(z); result 1

// let x = "10";

// let y = 10;

// let z = x\*y;

// console.log(z);  result 10

// let x = "10";

// let y = 10;

// let z = x-y;

// console.log(z);  result 0

// let x = "10";

// let y = 10;

// let z = x+y;

// console.log(z);  result 1010

// let x = "hello";

// let y = 10;

// let z = x/y;

// console.log(z); result : Nan

// let x = "hello";

// let y = 10;

// let z = x/y;

// console.log(isNaN(x));

// console.log(isNaN(y));  resul of x = true. y= false

Why cause we are asking not a number ? yes and no just like human rule.

REM= nan itself if a number. So if we ask typeof Nan. Result: number. [job interview]

* Infinity number

let myNumber = 2;

while (myNumber != Infinity) {

  myNumber = myNumber \* myNumber;

  console.log(myNumber);

}

//// Execute until Infinity

* Hexadecimal.

Js always display as base 10. Byt toString() method output number from base 2 to 36.

Hexa 16, decima 10, octal 8, binary 2. Computer works as binary 2.

For ex:

Decimal to Binary:

348. so as we know decimal is base on 10. Binary on 2.

So, we devide the number base on which one we want to convert it. So we want to convert ti binary means 2. So we devide 348 by 2. Ultill result is 0.

348>147>84>43>21>10>5>2>1>0 that is result we get if we keep deviding it by 2.

And that who we get. From reverse remainder result. 101011100.

This is just the remainder of 348 as we keep dividing it to 0. This is also called successive division.

So now if we want to do

Decimal to ocal we just divide it by 8 .same process.

Decimal to hexa same. But remember hexa is A from 10.

* Now ,
* let str= 'A' + 'M' + + 'E' + 'R' + 'I' + 'C' + 'A';
* console.log(str);

this is print AMNaNRICA

why , cause js just interpret this extra + and its character after that as NAN.

#20 JavaScript Number Method.

* toString() method

return a number as a string. Simple

* toExponential() mehod

let x = 9.655

x.toExponential(2); return: 9.66e+0

x.toExponential(4); return: 9.6565e+0

x.toExponential(6); return: 9.565454e+0.

Simple. So just round up with the specific amount you want.

* toFixed() method

let x= 9.655

x.toFixed(0): return: 10:

x.toFixed(2): return: 9.65:

x.toFixed(4): return: 9.655:

so just fixed the float.

* toPrecision() method

let x= 9.655

x.toFixed(3): return: 9.65:

just the exact amount of number u want to display

Converting VARIABLE to NUMBERS

* Number() method =

Number(true);  
Number(false);  
Number("10");  
Number("  10");  
Number("10  ");  
Number(" 10  ");  
Number("10.33");  
Number("10,33");  
Number("10 33");  
Number("John");

Result:

1  
0  
10  
10  
10  
10  
10.33  
NaN  
NaN  
NaN

* parseInt() method

parseInt("-10");  
parseInt("-10.33");  
parseInt("10");  
parseInt("10.33");  
parseInt("10 20 30");  
parseInt("10 years");  
parseInt("years 10");

result:

-10  
-10  
10  
10  
10  
10  
NaN

* parseFloat() method

parseFloat("10");  
parseFloat("10.33");  
parseFloat("10 20 30");  
parseFloat("10 years");  
parseFloat("years 10");

result:

10  
10.33  
10  
10  
NaN

Number Properties:

* MAX\_VALUE
* MIN\_VALUE
* POSITIVE\_INFINITY
* NEGATIVE\_INFINITY
* NaN

REMEMBER= number properties CANNOT be used on variables.

Cause Number proerpties belong to js number object wrapper call NUMBER

So it can only be accessed as Number.MAX\_VALUE.

But if we give like,

Let x = 6;

x.MAX\_VALUE;

result will be undefined.

So what we can do.

Let x= Number.MAX\_VALUE;

Calling the number max value method and saving it to another variable.

#21 JavaScript Array

Array is special variable which can hole more than 1 value. An array can hold up to 1 to thousand of data and access them referring an index number.

Some ruel:

* Declare array with CONST keyword
* Space and line breaks are not important declaration can have multiple line
* Array are special type of object
* Array uses number to access element and object use names to access
* You can have (OBJECT /Fucntion/ ARRAY) in an array

//Creating an ARRAY;

//common PRACTICE array declaring by const.

// const carArray= ["corolla","Nissan","Alien","Maruti","Tyota"];

// console.log(carArray);

//declaration can have multiple line

// const carArray= ["corolla",

// "Nissan",

// "Alien","Maruti",

// "Tyota"];

// console.log(carArray);

//you can provide element after creating an array

// const carArray=[];

// carArray[0]="corolla";

// carArray[1]="Nissan",

// carArray[2]="Alien";

//you can also crate an array using NEW keyword which is object

//Rem= element are in 2ndbraces not 3rdbraces as array. ;)

// const carArray= new Array("Maruti","Suzuki","toyota");

// console.log(typeof carArray);

//Changing an element into an array

// const carArray=["corolla","Nissan","Alien","Maruti","Tyota"];

// console.log(carArray);

// carArray[1]= "BMW";

// console.log(carArray);

//Accessing the full array in HTML file

// document.getElementById("tag\_name").innerHTML=carArray;

//Array may have Object/function/anotherarray in itself.

//Rem= array is a object. if we type typeOf it return object.

//Accessing the first and last element

// const carArray=["corolla","Nissan","Alien","Maruti","Tyota"];

// let x = carArray[0];

// let y = carArray[carArray.length-1];

// console.log(x+" and "+y);

//Rem= its not java that you just type: carArray(length-1).

// you have to put another time as the object is calling.

// so the line is carArra[carArray.length-1];

//looping through an array

// const carArray=["corolla","Nissan","Alien","Maruti","Tyota"];

// let len= carArray.length;

// for(let i=0; i<len;i++)

// {

//   let val=carArray[i];

//   console.log(val);

// }

//Adding an element into an Array

// const carArray=["corolla","Nissan","Alien","Maruti","Tyota"];

// carArray.push("RangeRover");

// console.log(carArray);

//element will be added in LAST.

//you can also add using length

// carArray[carArray.length]="LandRover";

// console.log(carArray);

//again remember while adding length as you have to type

//carArray and carArray as array and obejct twice.

//adding an element in high indexes create undefined hole

//so we call an index if there is no value then return undefined

//if we use name indexes JS will redefine the array as object

//creating array in 2 style

// const arrayfirst= new Array(40);

// const arraysecond= [40];

//these 2 above line is not the same, as 1st one declaraing

// new object type array of 40 indexes. and 2nd line array is

// taking element '40' as only 1 element in it.

//now if we write typeof array it will always return OBEJCT

//to solve this prob to recognize an array we use

// instanceOf operator and Array.isArray method

// const carArray= ["corolla","Nissan","Alien","Maruti","Tyota"];

// console.log(Array.isArray(carArray));

// console.log(carArray instanceof Array);

#22 JavaScript Array Method

console.log("Array Method");

//convert array to string

// const fruit= ["Mango","Orange","Banana","Apple","Grape"];

// console.log(fruit.toString());

//but if we want to join the array with something

// const fruits= ["Mango","Orange","Banana","Apple","Grape"];

// console.log(fruits.join(','));

//if we use POP then it will just pop the last elemet

// const fruits= ["Mango","Orange","Banana","Apple","Grape"];

// fruits.pop();

// console.log(fruits);

//as you can see it changed the total array by poping last element.

//as you know that push adds element at the last.

//we can remove element from the starting as well using shfit.

// const fruits= ["Mango","Orange","Banana","Apple","Grape"];

// fruits.shift();

// console.log(fruits);

//mango will be gone and return what has been removed

//POP remove form the LAST , SHIFT remove from the FIRST.

//Push add from the LAST, UNSHIFT ADD from the FIRST.

// const fruits= ["Mango","Orange","Banana","Apple","Grape"];

// fruits.unshift("Pineapple");

// console.log(fruits);

//Changing Element

// const fruits= ["Mango","Orange","Banana","Apple","Grape"];

// fruits[0]="pinaapple";

// console.log(fruits);

//Deleting elements

// const fruits= ["Mango","Orange","Banana","Apple","Grape"];

// delete fruits[0];

// console.log(fruits);

//see the diff, just deleted the first element not shifted.

//just undefined hole in there and even length been the same.

//so clearly understand the diff bettween delete. but its not

//encouraged.

//NO splice() method

// const fruits= ["Mango","Orange","Banana","Apple",

// "Grape","Watermelon"];

// fruits.splice(3,1,"Kola","kathal","Jambura");

// console.log(fruits);

//so, 1st para is the position,2nd how many u want to delete from

//that positin and rest of the para is the element you want to

//push from the give index in 1st prositon

//SIMPLE.

//even following command will keep only 2 element

//fruits.splice(2);

//following command will delete 2 element from given

//fruirts.splice(2,2);

//even to remove an element and reduce the size we use splice

//that how we ignore undefined holes.

// fruits.splice(0,1);

// console.log(fruits);

//so, splice reduce the main array and return the element its

//remoing. nothing removine ? then null will be returned.

//Concat method. just add it after the array we want to add

// const myGirls = ["Cecilie", "Lone"];

// const myBoys = ["Emil", "Tobias", "Linus"];

// const myChild= myGirls.concat(myBoys);

// console.log(myChild);

// so we adding another array into the array and we are saving

// it to another array to print

// so thats the DIFF between SPLICE and CONCAT, splice change the

//main array where concat does not.

//we can concaat 3 array or we can contat elements too.

// const myGirls = ["Cecilie", "Lone"];

// const myBoys = ["Emil", "Tobias", "Linus"];

// const myChild = myGirls.concat(myBoys,"Sandip",5);

// console.log(myChild);

//SLICE method

// const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];

// console.log(fruits);

// const citrus = fruits.slice(1,4);

// console.log(citrus);

//so slice is,1st para is the starting postion for the slice and

//2nd para is until. rem= not the postions. so above code. will

//give orange lemon and apple not mango. so not untill that.

//Slice method create a new array Does not change the source array

//like SPLICE. oke ?

#23 JavaScript Array SORT

If my Numbers are sorted as string, “25” is bigger than “100”, because 2 is bigger than 1. So the result will be incorrect

So we can FIX this solution by COMPARE FUNCTION.

const points = [40, 100, 1, 5, 25, 10];

points.sort(function(a, b){

  return a - b

}

);

console.log(points);

sort function returns negative/zero/prositive value, depending on the argument

if result is negative , a is sorted before b

if result Is positive, b is sorted before a

if the result is 0 no changes are done with the sort order of the value.

So 40 goes with 100 then 40 goes with 1 and then 40 goes with 5 like this and sorte

REM =compare function compares 2 values at a time

For ascending a-b

For desceneding b-1

Math.random

* Return 0.5-Math.random();

Highest and lower value:

There are no built in function for that. So what we do now, sort from ascending order and chose the first and last index. ;)

//Find Highest and Lower Value.

const points = [40, 100, 1, 5, 25, 10];

points.sort(function(a, b){

  return a - b

}

);

let firstelement=points[0];

let lastelement=points[points.length-1];

console.log(firstelement);

console.log(lastelement);

max and min value function:

//max value

const points = [40, 100, 1, 5, 25, 10];

function myFunction(arr){

  return Math.max.apply(null,arr);

};

let x=myFunction(points);

console.log(x);

//minvalu

const points = [40, 100, 1, 5, 25, 10];

function myFunction(arr){

  return Math.min.apply(null,arr);

};

let x=myFunction(points);

console.log(x);

JS array can contain OBJECT too.

And we can use sort function to sort them too

//we can also sort object in array using compare func

const cars = [

  {tap:"Volvo", year:2016},

  {tap:"Saab", year:2001},

  {tap:"BMW", year:2010}

];

cars.sort(function(a,b){

  return a.year- b.year;

});

console.log(cars);

#24 JavaScript Array Iteration.

//ForEach() method

const numbers= [45,4,9,16,25];

function myFunction(value,index,array){

  console.log(value);

  console.log(index);

  console.log(array);

  console.log("----")

}

numbers.forEach(myFunction);

//above code foreach passing 3 parameter.

Map method:

* Creates new ARRAY DOES NOT change the original array
* DOES NOT execute the function without array elements or values
* //Array.map() method
* const numbers= [45,4,9,16,25];
* function myFunction(value){
* return value\*2;
* }
* const newNumbers=numbers.map(myFunction);
* console.log(newNumbers);
* //above code we ara calling map method as same as foreach function
* //we are multiplying each element by 2. and we are saving the
* //total array in a new array and printing it.

Filter method:

//Filter() method.

//suppose we need all element greater than 20 how do

//Array.map() method

const numbers= [45,4,9,16,25];

function myFunction(value){

  return value>10;

}

const newNumbers=numbers.filter(myFunction);

console.log(newNumbers);

//so above code, we are returning all those values which are

//greater than 10. and array will save the new elemenets in

//new array. but we want to save the true or malse just use.

//map word insted of filter. that sit.

Reduce method()

//Reduce() Method

const numbers= [45,4,9,16,25];

function myFunction(total,value,index,array){

  console.log(total);

  return total+value;

}

const newNumbers=numbers.reduce(myFunction);

console.log(newNumbers);

//reduce does not create another array rather it reduces

//to a 1 value. 1st total is zero and adding the value which

//is 45 and returning. again adding value 4 with 45. so so..

Changing the initial as 10.

//now, we can pass initial value

const numbers= [45,4,9,16,25];

function myFunction(total,value,index,array){

  console.log(total);

  return total+value;

}

const newNumbers=numbers.reduce(myFunction,10);

console.log(newNumbers);

//so above code we are setting initial vlaue as 10

//and adding first element 45 and returnting 45+10

//=55 and +4=59 then so so...

#25 JavaScript Array const

Array should be always declare as const but we know that const cant be reassign.

And that rule also goes with array too.

Why ? cause array itself is not a const. so don’t get confuse here. We are declareing an array and reffering a const variable in it. So we can re assign array but we can change the element.

But,

Array declaring with var variable we can do that. Redaclare /using in block withtout block. As u wish

Just remem= same scope is not allowed to redeclare.

JUST READ THE Array Const. its conceptual.

#26 JavaScript DATES

//creating date objects, but it created with new Date() constructor

//

const d = new Date();

console.log(d);

there are 4 ways to create new date object

* New Date();
* New Date(year,month,hours,minutes,seconds,miliseconds)
* New Date(miliseconds);
* New Date(date string);

//now lets call them by year months hour seconds milisecond

// const d = new Date(2020,11,24,54,74);

// console.log(d);

//Reme= december is 12 by normal, but in code js its 11

//cause we know everything start from zero.

//6 number specify year months day hour minute seconds

//5 number specify year months day hour minute

//4 number specify year months day hour

//... so so.

//but, you CANNOT ommit months. if we give only 1 parameter

//it will take it as a milisecond.

#27 JavaScript Date Format.

ISO is most internation standard JS system

Just read the w3school Date Format section

#28 JavaScript Get Date method

#29 JavaScript Set Date method

For above 2 hashtag just read the W3school.

#30 JavaScript Math.

* Math.PI; result: 3.141613215154;

Unline other object, math has no constructor. Kinda built in object. All the method and properties of math object can be used.

8 mathmatecial constant

Math.E        // returns Euler's number  
Math.PI       // returns PI  
Math.SQRT2    // returns the square root of 2  
Math.SQRT1\_2  // returns the square root of 1/2  
Math.LN2      // returns the natural logarithm of 2  
Math.LN10     // returns the natural logarithm of 10  
Math.LOG2E    // returns base 2 logarithm of E  
Math.LOG10E   // returns base 10 logarithm of E

4 methods to round an integer number

|  |  |
| --- | --- |
| Math.round(x) | Returns x rounded to its nearest integer |
|  |  |
| Math.ceil(x) | Returns x rounded up to its nearest integer |
|  |  |
| Math.floor(x) | Returns x rounded down to its nearest integer |
|  |  |
| Math.trunc(x) | Returns the integer part of x ([new in ES6](https://www.w3schools.com/js/js_es6.asp)) |

#31 JavaScript Problem 1

Check the code file in vs code summit.

#32 JavaScript Boolean

Read the w3school. Simple.

* 10>9
* Boolean(10>9)

Both returns same result true.

== equal to , > greater than, < less than.

Comparaison and Condition is the base of JS Boolean character

Eveything with value in JS is true

forex:

100

3.14

-15

These all are true. Why ? js thinks this are value and its there that’s why true.

But in which base? Condition and compare based.

* If no value then its false.

eX;

let x =0;

console.log(Boolean(x)); result false

* empty string also false
* undefined also false
* null also false
* and false value itself a false
* Nan is also false

So above all types will return false if Boolean function find it.

Interview asks. What are the falsy value of js. Answer is above types.

Boolean can be Object

Let x = new Boolean(false);

But we are gonna ignore it.

Always REMEMBER (==) double equal is always everywhere is comparaison operator which checks the value. And (===) is also comparator operator check value and type

So for ex:

Lex x =false;

Let y = new Boolean(false);

(x==y) // answer true as value is same

(x===y) // answer false as value same but type diff

Even, we declare both variable new object and check double equal or triple equal both will be giving false. CUASE WE KNOW OBJECT CANNOT BE COMPARED>

#33 JavaScript Comparison

Lets set , x = 5

* For == double equal
* X ==8 is false
* X == 5 is ture
* X == “5” is TRUE why cause value is same but type is diff.
* X ===5 true
* X ===5 false as value same but type diff
* X !=8 true
* X !==5 false
* X !==”5” true cause type diff and that trufy the condition.
* X !==”8” True. Same as above.

If we compare “2” with “12”

Boolean (“2”>”12”);

True cause alphabetacillay 2 is greater than 1.

Ex: Abba abbu which will be first ?

Abba cause first 3 character both string matches but 4th is diff in both

So ’a’ will be greater then ‘u’. simple.

#34 JavaScript Conditions

Read it. Simple.

#35 JavaScript LOOP

Understand 1 thing first.

* 1+3 //this is an expression
* Print X = 1+3 // this is statement

If(statement 1, statement 2, statement 3)

REMEMBER= In js we can initialize many value in statement 1

Ex:

For(let i= 0 , len = cars.length, text=”” ; i<len ; i++)

We can even totally omit the 1st statement, its optionsl

But remember to put the semicolon of the 1st stement

let i = 2;  
let len = cars.length;  
let text = "";  
for (; i < len; i++) {  
  text += cars[i] + "<br>";  
}

look carefully semicolon is there if not then error will be sent.

We can totally omit 2nd statement too…

As we are using statement 2 to define the codititon to run the loop. So if we don’t give it. Make sure you put break inside the loop or browser will be crash. Why breaks cause break break out from the loop.

We can even totally omit 3rd statement

If we does increment or decreament inside the loop then its not necessary to give it on for loopp braces.

#36 JavaScript For In loop.

For in array is to iterate over the property of the object

For(let x in Cars)

{

Console.log(cars[x]); // remember don’t put cars.x cause there is not property called x.

// we are accessing the value of key. So key here is x and car[x] is the

// right way to access of the value of the property.

}

REMEMBER DO NOT use FOR IN over an array. Why ?

Cause index is important when it comes to array. Changing the index will changing the value. But in object. A property is name:value system. So the name of properties order could be anything. Result will always be the same if we call it.

So use for loop, for of loop , foreach method in term of ARRAY element.

#37 JavaScript For Of LOOP

Iterable objects are > array, string,map, nodelist, and more.

#38 JavaScript While lopp

Read the w3school simple.

do while will always execute the loop once before its checks the conition and then it execute the do while again if the while loop condition is fulfilled.

#39 JavaScript Break and continue.

Break is as we know simple. Just break out from the loop.

But continue is just continue the iteration from the next, so if we print from 1-10 and in number 3 iteration time, it finds continue. So 3 iteration will be skip. So output will be all the number from 1 to 10 except 3.

Another thing is label break

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**.

With a label reference, the break statement can be used to **jump out of any code block**:

So continue skips onely 1 iteration , break and label break has diff which is if we use break it jusmp out of every loop of loop or nested lopp but label break just jump out of BLOCK. ☺ simple.

#40 JavaScript Iterable

W3 school read. Simple.

#41 JavaScript Set.

* New Set() // create new set
* Add() // add a new element to the set
* Delete() // removes an element from a set
* Has() // return true if a value exists in the set
* forEach() // invokes a callback for each element in the set
* values() //returns an iterator with all the values

pass an array to the new Set() constructor

eX: const letters= new Set([‘aa’,’bb’,’cc’]);

* adding values:

letters.add(“a”);

letters.add(“b”);

letters.add(“c”);

REMEMBER: if you add equal element only the first value will be added.

So that’s one speciality of set.

* For each method

Set has exactly foreach method as array.

REST of the part just read the w3school.

#42 JavaScript Maps.

Map holds key-value pairs where the keys can be any DATATYPE

Map remembers original insertion order of the keys

* Creating a map

=passing an Array to newMap();

=create a Map and use Map.set();

Ex:

const fruit = new Map([[“apples”,500],[“banana”,300],[“orange”,200]]);

another way,

// Create a Map  
const fruits = new Map();  
  
// Set Map Values  
fruits.set("apples", 500);  
fruits.set("bananas", 300);  
fruits.set("oranges", 200);

Changing the exisiting value.

fruits.set("apples", 200);

to get a value

fruits.get("apples");    // Returns 500

to delete a value

fruits.delete("apples");

has or has not just check if the key exist in the

fruits.has("apples"); // returns true or false.

FOR EACH METHOD

// List all entries  
let text = "";  
fruits.forEach (function(value, key) {  
  text += key + ' = ' + value;  
})

//Remember other foreach method just value needed. But for map value and key both needed.

#43 JavaScript TypeOF

Read that properly. Cause its important

#44 JavaScript Type Conversion

Number() can convert String to Number.

* Number(“3.14”) //return 3.14
* Number(“ ”) // return 0
* Number(“”) // return 0
* Number(“99 00”) //return Nan

Number Method

* Number() = return number
* parseFloat()= parse a string and return a floating number
* parseInt()= parse a string and return a int number

Unary +operator

Let y = “5” // y is string

Let x = +y // x is a number

String() can convert number to > String

Ex:

String(x) // returns string from a number variable

String(123) // return string from a number literal 123

String(100+23) //return string from a expression

Rememerb? toString() does the same above ;)

Converting Date to string

String(Date())

Same as, Date().toString();

Converting Boolean to number

Number(False) // return 0

Number(True) //return 1

Converting Boolean to string

String(false) // return false

String(true) // retrun true

Automatic type converstion

Js smartly sometimes change the wrong datatype into right datatype

* 5+null // return 5 because null is converted to zero
* “5”+null // return 5null.
* “5”+2 // 52
* “5”-2 // 3
* “5” \* “2” //10

Automatic String Conversion

Read w3school

#45 JavaScript Bitwise

Read w3

#46 JavaScript Regular Expression

Read w3

#47 JavaScript ERROR

* Try = statement lets you test a block of code for ERRORS
* Catch = statement lets you handle the error
* Throw= statement lets you create custom errors
* Finally= statement lets you execute code, after try and catch, regardless of the result

Silly mistake/logical error/ hacker attack dangerous error

So,

We write the code in try block and if any error founds then js sends it to catch(err). So code in try block. Solve in catch block

Simple.

jS THROWS error

When an error get find by JS. JS will stop and generate an error message.

So we call it technically, JS will throw an exception(throw an error).

Basically what happens is js create an ERROR OBJECT with 2 properties.

Which are ‘name’ and ‘message’. Name is the built In name (ex: reference error, syntax) of the errors. And message is generated by jS what the problem there.

The Throw Statement:

Throw statement allows you to create a custom error. Can be js String, number Boolean object.

Ex:

throw “too big” //return text

throw 500 // throw a number

if you throw together with try and catch, you can control program flow and generate custom error message.

function myFunction() {  
  const message = document.getElementById("p01");  
  message.innerHTML = "";  
  let x = document.getElementById("demo").value;  
  try {  
    if(x == "") throw "empty";  
    if(isNaN(x)) throw "not a number";  
    x = Number(x);  
    if(x < 5) throw "too low";  
    if(x > 10) throw "too high";  
  }  
  catch(err) {  
    message.innerHTML = "Input is " + err;  
  }  
}

#48 JavaScript Scope

3 types

* Block Scope
* Function scope
* Global Scope
* VAR + LET + Const = accessed anywhere if its GLOBAL
* VAR + LET + CONST = NOT accessed anywhere if its in FUNCTION SCOPE
* VAR = Accessed anywhere except if its declared in Function scope

LET and CONST provide BLOCK SCOPE and variable declared block scope cannot be accessed outside the block. Simple.

BUT,

Variable declared as VAR in block or anywhere, can be accessed from anywhere. But if it’s a function scope and var is declared inside it.. then it wont work.

So in clean slate… var declared any scope except function scope can be accessed from anywhere.

Function scope are also called as LOCAL scope.

So another clean slate is.

Variable declare in functionscope CANNOT be accessed anywhere except that function. Same name variable may have in the program and its work could be different. Simple.

#50 JavaScript HOISTING.

Hoisting js default behavior of mobbing declarations to the top.

Hoisting is JavaScript's default behavior of moving all declarations to the top of the current scope (to the top of the current script or the current function).

IF var is declared later it will work

But, if let and const is declared later then reference error and syntax error.

Job interview= 3 of them are hoisted but 2 of them will give reference+syntax error. Remember that. ☺

Just declared variable on the top. Cuase js interpret a lot. So its better not to let it. ;)

One thing remember, initialization and declaration are not same.

So if we don’t initialize a variable with value. It will be undefined. So initialization is MUST. Then comes the topic that the variable will be hoisted or not. ☺

#50 JavaScript STRCIT MODE.

“use strict”; // simple

If u put it on global then it will be working all over

But if u declare it in function scope then just that scope.

So use string works one which data types ?

Number + object both

Delete and variable is not allowed if user strct

Delte a a parameter is not allowed

READ w3 for the rest of allowance.

REM= Declare ‘use strict’ in the beginning of global scope or in the beggintion of function scope if u use it.

#51 JavaScript THIS keyword.

const person = {

  firstName: "John",

  lastName : "Doe",

  id       : 5566,

  fullName : function() {

    return this.firstName + " Hello 1 " + this.lastName;

  },

  getId: function(){

    return person.fullName()+"Hello 2";

  },

  getFullname: function(){

    return this.getId()+"Hello 3"+this.fullName();

  }

};

console.log(person.fullName());

console.log(person.getId());

console.log(person.getFullname());

//above code is implicit binding

//no we are gonna learn explicit binding.

const person1= {

  fullName: function(){

    return this.firstName+" "+this.lastName+" Works in here";

  }

}

const person2={

  firstName: "Jansir",

  lastName : "Hossain",

}

console.log(person1.fullName.call(person2));

//above code is using call function and passing another object

//person2 as a parameter in another object call. that the

//speciality of call fucntion and its alled explicit binding.

|  |  |
| --- | --- |
| Precedence | Object |
| 1 | bind() |
| 2 | apply() and call() |
| 3 | Object method |
| 4 | Global scope |

#52 JavaScript Arrow Function

Read w3

#53 JavaScript Class

Js classes are templates for JS objects.just like java. We need to make an object for class. Each object can pass value by paremeter. Classes must have constructor and properties in it. Class can have different method.

Below code show how to create class construcot method and how to create object and call it.

class playerName{

  constructor(name,age){

    this.name=name;

    this.age=age;

  }

  play(year)

  {

    return this.name+` has ${year-this.age} years left`;

  }

}

let pl1= new playerName("Shakib",34);

let pl2= new playerName("Mashrafe",36);

console.log(pl1.play(40));

console.log(pl2.play(40))

#54-60 just read