#45 Javascript – Introduction.

Date

→ What is javascript ?

* Javascript is a client side scripting language.
* It is used to make webpages alive.
* It is used to programmatically performs actions within the page.
* When js was created, it was initially called "Livescript".
* But java was very popular language at that time, so it was decided that positioning a language as a "younger brother" of java would help.

→ What Javascript can do ?

1. Javascript can execute not only in the browser, but also on the server.
2. We will use Javascript as a client as well as server side language.
3. Javascript has evolved greatly as a language and is now used to perform wide variety of tasks.

Date

→ What can in-browser javascript do?

Javascript → safe. Low level GPU.

(1) Add new HTML and change existing HTML from DOM.

(2) React to events → response from server
→ key press
→ Mouse movement

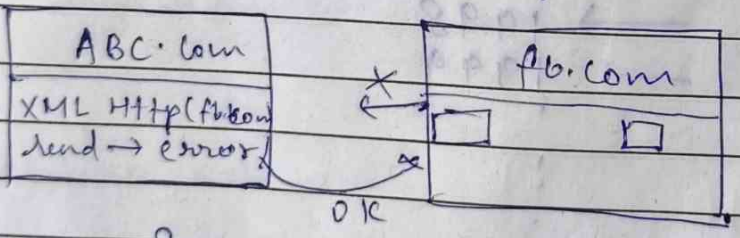
(3) Ajax requests

(4) Get and set cookies and use local storage.

→ What can't in-browser javascript do?

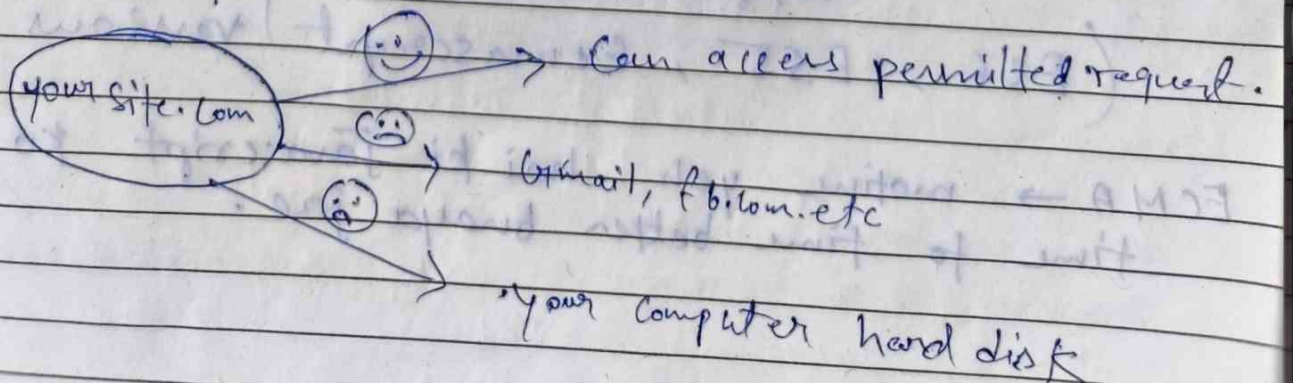
(1) Read/write to and from computer harddisk.

(2)



Same origin policy.

(3) Summary →



Date

→ What makes Javascript unique :-

(1) Support of HTML and CSS (integration).

(2) Simple things → Simple. API's.

(3) Major modern browsers. Enabled by default

→ Javascript versions :-

(1) Maintained by Community

(2) Js created in 1995

(3) ECMA (update Js frequently) → 1997.

1st version → 1997
2nd " → 1998
3rd " → 1999
:
:
ECMA → 2015
2016
2017
2018

(ES6, ES5, Ecmascript) versions of J

ECMA → motive yeh hai ki Javascript ko time to time better banaya jae.

Spiral

#46 Javascript write in developer console.

Date

- <script> </script> tag is used to write JS in HTML.
- JS can be write in
- <Head> your JS here </Head>

OR

- <Body> ... HTML content ... your JS here </Body>

OR

- <script src = "path of JS file"></script>

→ Better option. (Just </Body> se pehle).

For more visit : JS dev console 45.html

—x—

#47 Variables, datatypes and operators in JS :-

- Variables and datatypes-

```
var a = 78;  —— int
var b = "Hellow world";  —— String
Console.log(a);  —— return 78
Console.log(b);  —— return "Hellow world"
Console.log(c);  → Error.
```

★ We don't define the data type it is understanding automatically as it is using dynamic typing.

Spiral

Date

→ Operators in JS :-

In 2 + 3 '+' is operator and 2, and 3 are operands.

Operands : entities on which operators operate.

→ Types of operator :-

1. Unary operator :- It has single operand (x = -x).

Ex - C = -C;
Console.log(C);

2. Binary operator :- It has two operand (x = x + 3).

Ex - C = C + 3
Console.log(c);

Note :- we are skipping bitwise operator in web d

- Basic Arithmetic operator :- num1 and num2 are
  - num1 + num2 = add.
  - num1 - num2 = sub
  - num1 * num2 = Mult
  - num1 / num2 = divide
  - num1 % num2 = modulus
  - num1 ** num2 = exponential.
  - num1++ = num1 printing then increase by
  - ++num1 = num1 is increasing then num1 pr
  - Sly, num1-- and --num1.

For more visit : Var dt op 46.html.

Spiral

#48. String in Javascript

Date

- var string = "this.";
  >>> this.
- var string = 'thi"s.';
  >>> thi"s.

- var name = "Hrithik";
  var message = "Hiii";
  var Roll = "05";
  console.log(name + message + Roll);
  >>> HrithikHiii05

- Finding length of a string :-
  var len = length.name;
  console.log(length of name is ${len}.);
  >>> length of name is 7.
  ① → backtick (below esc).

- Escape sequences :-
  1. \\ : for printing backlash (\n = \n)
  2. \n : new line
  3. \t : Tab space.

- String using construction :-
  var y = new String("this");
  console.log(y);
  >>> String {'this'}

- Inserting a string in DOM using JS :-
  document.getElementById('content').innerHTML = '<H3> this is a h3 heading. </h3>'.
  >>> this is a h3 heading (will displayed in DOM).

For more visit : String js 47.html

Spiral

#49. String functions in Javascript.

Date

→ First occurance of a substring:
var position = str.indexOf('is');
console.log(position);

→ For last occurance of a substring:
lastIndexOf()

→ Substring from a string by slicing:
var substr = str.slice(1,5);
s, e+1*

Slice(m,n) :- m = starting index, n = ending index
It will slice the string from 'm' to 'n-1'.

→ Substring from a string by Substr:
var substr = str.Substr(1,5);
s, l*

Substr(m,n) :- m = starting index, n = no. of character
It will slice the string from m to till the n-value.

→ To replace a string characters by other characters or string:
var replaced = str.replace('a string', 'new string');
It will not permanently replaced.*

→ To convert strings to lowercase and UPPERCASE:
console.log(str.toUpperCase());
console.log(str.toLowerCase());

→ To Concate two or more string:
var newstr = str.Concat('New string');] choice
var newstr = str + 'New string';
console.log(newstr);

Spiral

Date

→ To trim the spaces from starting and ending of the string.
Console.log (str_with_whitespaces.trim());

→ Extracting characters from a string.

var char3 = str.charAt(n);
where 'n' = the index of that character.

→ To print character code :-
var char3 = str.charCodeAt(n);
At will return the the charactercode.

→ Slicing shortcut for one character:
Console.log (str[n]);
Here 'n' is the index of that character.

For more visit: String functions 4 8.html

Spiral

#5° Scope and Conditionals statements.

Date

→ Scope of a variable :-

- Var variable name :- Var is used to declare a variable globally. If you put {var - -} inside a block still its scope will be global.

Ex - * var string = "Hii";
var string = "Hellow";] will work

It will print Hellow we can use same variable name using var and it will update the value as many time we use it.

- let variable name :- let can be used as globally but if you put {let - -} inside a block it will have scope only in the block.

Ex - * let string = "Hii"
let string = "Hellow";] will not work give error.

- Const x = "This cannot be changed";
Const x = "This can be change";] will not work give error

Reason - 'Const' means constant its value cannot be changed.

→ Where to use, var, let and Const :-

- var :- When you have to update or change the variable periodically as per the conditions.
- let :- When you have to use a variable specially in a block. and let is mainly used in the program.

Spread

Date

• Const :- When you have to make a variable and you don't want to change the value you want to keep the value unique.

→ Conditional statements:

• if (condition) {
  statement
}

• else (condition)
{
  statement
}

• else if (condition)
{
  statement
}

★ How to use ?
We use these statements according to our needs :-
• When we have to check one or two condition then we (if - else) conditon.
• When we have to check more than two conditions then we (if-else-ladder).

→ Switch Case :- Conditional statements :-

```
Const variable_name = value;
switch (variable_name) {
  Case value :
  console.log ("The value matches");
  Break;

  Case wrongvalue :
  console.log ("value not matches");
  :
  default :
  Console.log ("default value if none matches");
  Break ;
```

★ if we dont use break; it will print all the values after the target value including the default.

For more visit : Scope and Cond 49.html

Spiral

#51. Arrays and Objects in Javascript

Date

- Object :- Object is a non-primitive data-type that allows you to store multiple collections of data.

→ Creating an Object :-

```
let employee = {
    name : "hritik",
    Salary : 500000,
    education : "btech",
    "education2" : "CSE",
    Specilization : "CS".
}
Console.log(employee);
```

```
let object_name {
key : value,
key : value,
.
.
.
}
```

→ ~~Way~~ Array :- Array is used to store multiple values of a single variable.

- ways to create an array :-

★ 1st way : let names = [1, 2, 4, "Hritick, undefined];

★ 2nd way : let names = new Array (1, 2, 4, "Hritick", undefi

→ Creating an ~~array~~ empty array of length n :-

★ let names = new Array (n);

→ length of array :

Console.log(names.length);

→ Sorting of array :-

★ names = names.sort();

→ adding element to array :- (~~to~~ to the last)

names.push("this is pushed");

→ Deleting/removing an element from the last of array :

names.pop();

#52. Functions in Javascript.

Date

• Function :- a set of statements that performs a task or calculates a value.

Creating a function with example :-

```
function greet(name, greettext = "greeting from IS");
{
  let name1 = "name1";
  console.log(greettext + " " + name);
}
```

(x) (y) (*) → default value

* → Block level variable (local) does not execute.

```
let name = "Hrithik";
let name1 = "shubham";
let name2 = "Saurabh";
let name3 = "Rohan";
let name4 = "Shiven";
```

variables of name (x)

```
let greetTexts = "Good Morning";
```

greettext variable (y)

Calling the functions:

```
greet(name, greetTexts);
greet(name1, greetTexts);
greet(name2, greetTexts);
greet(name3, greetTexts);
greet(name4);
```

default value of y will show here.

if we create function and forget to call it, it will not be executed. 😐

But in this above function ~~no the value~~ function did not returning anything.

Date

→ A function that returns :-

```
function sum (a, b, c){
    let = a + b + c;
    return d;
```

* Console.log ("function is returned");
    ↳ * This statement will not be execute because in returning function no of line of statement will be executed after the 'return' statement.

}

⇒ Calling this function :

```
let returnVal = sum (1, 2, 4)
Console.log (returnVal);
```

↳ This will return the sum of 1 + 2 + 4 as d which is equal to '7'.

For more visit :- functions 51.JS , functions in JS 51.HTML

#53. Alert, Prompt, and Confirm Interaction in Js.

Date

• ALERT :- Alert in in-browser Js is a method to show a pop-up ad. like dialog box on your website.
It does not return anything (value - undefined).

>>> alert("This is a message");

• PROMPT :- prompt is a method to take user input in that alert like box.

~~>>> prompt (console~~
~~let prompt~~

>>> let name = prompt("What is your name", "Guest");
(user input, default input)

• Confirm :- Confirm is used to double-check an option that you have selected like a dialog box will double check you.

>>> let deletePost = Confirm("Do you want to delete");

```
if (deletepost) {
  console.log("Your post is deleted"); }
else {
    console.log("post not deleted");
    }.
```

for more visit : Alert-52.html

Spiral

New - JS

Date

→ Ecmascript → different docs of js are talking in same language (standard of js).

→ Execute js → ① Browser, ② insert in HTML, ③ Nodejs.

Ch-1

- variables: var a = 7 (a: identifier, =: assignment operator, 7: literal)
- Rules of Naming var.

Var → global, let, const → local.

Chap-2

- Operator's Arithmetic, assignment, Comparison, logical, ternary → Condition ? exp1 : exp2.
- Condition: if, if-else, if-else-if.

Ch-3

- For loop → for in (obj), for of (array).
- while loop.
- Do while loop.
- Function :-

```
Function myfunc(p1, p2){
    Code —
}
myfunc(1, 2).
```

```
Const sum = (a, b) => {
    let c = a+b
    return c
}
let y = sum(1, 2)
Console.log(y).
```

Date

ch-4

- String.
- Template literal - `Hey "Hrithik" 'nice' it.`
  `TW's is ${var}`
- Escape sequence - "Adam D\'Angelo"
  \n, \t, \r → carriage return.
- String properties & methods.

ch-5

- Array → stores only one type of value.
- Addressing = num[0] >>> 1
- Finding length = num.length >>> 4
- Changing the values = num[2] = 8.
- type of (num) = object.
- Array methods. — (12)
- Looping through arrays:-

1. For each loop:- Calls a function once for each element.

```
Const a = [1, 2, 3]
a.foreach ( (value, index, array) => {
        //function logic
    });
```

2. Map() → Creates a new array by performing some operations on each array

Date

```
Const a = [1, 2, 3]
a.map((value, index, array) => {
        return value * value;
});
```

3. Filter() → filter an array with values that passes a test. Creates a new array.

```
Const a = [1, 2, 3, 4, 5, 7]
a.filter(greater than 5)
```

4. Reduce method. → reduces an array to a single value.

```
Const n = [1, 8, 7, 11]
let sum = numbers.reduce(add);
          ↓
        1+8+7+11
```

function

5. Array from → used to create an array from any other object.

```
Array from("Mrittik")
```

6. for of → get values from an array.

7. for in → get keys from an array.

Date

Ch-6

Window (Controls window | redirect, open ...)

- DOM (whole HTML as per JS obj).
- BOM (alert, prompt, confirm)
- JS Core

Ch-7

- Text node, Element node, Comment node } DOM Tree
- Autocorrection. - (auto move inside body)
- Child nodes - direct children.
- Descendant nodes :- Children and their child nodes
- element first-child — first element child
- element lastchild — last element
- element childnodes — All child nodes.

- elem.childNodes[0] === elem.firstChild
- elem.childNodes[elem.childNodes.length-1] === elem.lastChild.
- elem.childNodes() - check weather array or not
  → type — Collection not array.

- left and previous sibling are same.
- right and next siblings are same.
- Element only navigation — no text or comment node.
- Table links.

→ Searching the DOM :-

- document.getElementById.
- document.querySelectorAll
- " . querySelector
- " . getElementsByTagName
- " . getElementsByClassName

- matches()
- closest()
- and
- contains() method.

Ch-8

Date

- console.dir function.
- tagName / NodeName
- InnerHTML ⇒ HTML inside the Element node.
- OuterHTML ⇒ HTML + Element itself.
- textContent ⇒ Text — all tags inside an elemt
- Hidden property ⇒ visible or not.
- Attribute method ⇒
  - elem.hasAttribute(name)
  - elem.get
  - elem.set — (name, val) - set value of an elemt
  - elem.remove
  - elem.attribute → get collection of all attr.
- data — attributes — data-one → elem.dataset.one

→ Creating a div using js and appending to body.

- node.append(e) — at end
- node.prepend(e) — at begn
- —— before(e) — before node
- —— aftr(e) — aftr node
- node.replaceWith(e) — e is a node replace frm

- Insert Adjacent HTML / Text / Elemt.
  → Before begn, aftrbegn, befrend, aftrend,

→ Node removal: node.remove()

```
let id1 = document.getElementById("id1")
id1.remove().
```

- ClassName & ClassList.

- elem.classList.add/remove("class") - Add/remove to cls
- elem.classList.toggle("class") - add if not there otherwise remove
- elem.classList.contains("class") - check and return true or false

Date

- SetTimeout ⇒ let timerId = setTimeout ( ) ⇒

alert ("name"), 1000) ;

Clear Timeout (timerId) — cancel the timer

- setInterval :⇒ Same syntax, but runs continuously we can stop by = clearInterval (timerId).

- Browser Events ⇒

→ Mouse Events — click, rightclick (contextmenu) etc.

→ Keybord Events — keydown, keyup etc

→ form element events — submit, focus etc

→ Document events — DOM ContentLoaded.

- Event Handling — Can be Handled like onclick =

- Add/remove Event listner — assign multiple handlers to an event.

elem. Add/remove EventListen (event, handler).

- The Event Object. event happens — browser — creates event object — put details — pass as an argument — handler.

Ch-9

Date

- Callback function
- Handling errors. - Handle callback errors by sending error around.

```
function loadScript (src, callback) {
    let script = document.createElement('script');
    script.src = src;
    script.onload = () => callback(null, script);
    script.onerror = () => callback(new error("failed"));
    document.head.append(script);
}
```

→ callback hell / pyramid of doom —

↳ Solution ⇒ promises. { either resolve or failure } both cases are handled.

```
let promise = new promise (function (resolve, reject) {
    // executor
});
```

→ Consumers: then, catch.

```
promise.then (function (result)
    fn : (error)
);
```

- promise.then (alert); — only success
- promise.catch (alert); — only error.
- promise.finally(() => {}) — good cleanups.

→ Attaching multiple handlers ⇒ one promise, multiple handlers, pass independently.
result ...

Spiral

Date

→ Promise API :-
- Promise.all(promises).
- Promise.allSettled(—)
- Promise.race(—)
- Promise.any(—)
- Promise.resolve(value) →
- Promise.reject(value).

- Async / await. (wait until promise settles and return its value)
  - async → returns promise; wraps otherwise in a promise
  - await → used inside async function

- Error Handling

```
try {
    try code
} catch (err) {
    error Handling
}.
```

Synchronously — agar try kamandar setTimeout() me lai to kaam nhi karta catch.

→ The error object.

if (x) { throw new Error("......."); }

or

let error = new SyntaxError(msg) or new ReferenceError()

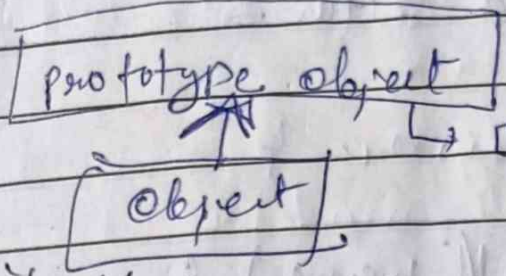
- The finally clause.

→ Ye chlega hi har case me.

Ch-10

Date

→ · Prototype



- prototypal Inheritance :
- setting prototype — — proto — —
- Classes and object —
  template → create

```
Class My-Class {
    // Class methods
    Constructor() {---}
    Method1() {---}
    method2() {---}
}
```

↳ new MyClass () (create object with new proto)

- Inheritance :
- extend keyword ⇒ · Class Monkey extends Animal
- Method overriding —
- Super keyword — Super(a, b) call parent constructor

- Overriding constructor · — Happens when we don't write our own Const —

↓

→ must call super()

→ super.method() in child → get parent method

⇒ Static method → belongs to Class but not Spiral object.

Date

- Getter and Setter –

```
Class person {
    ____
    get(name) {
        return this.name;
    }
}

set name(newName) {
    this.name = newName;
}
```

- Instance of operator. – if a object Enters belongs to certain class.

```
obj <obj>. instanceof <class>
>>> true.
```

Ch - 41

Date

- IIFE Syntax: (function () {
  ...
  ...
  }) ();   IIFE Syntax.

avoid polluting the global namespace.

- Destructuring:-
  [10, x, ...rest] = [10, 80, 7, 11, 21, ...]
  (80 → x; 7, 11, 21, ... → rest)

  more example visit notes (Chodu).

- Spread syntax:-
  ① Const arr = [1, 7, 11]
  Const obj = {...arr}; // {0:1, 1:7, 2:11}

  ② Const nums = [1, 2, 7]
  Console.log(Sum(...nums)) // 10.

→ local & global scope.
- Block, function, global.
  (Block, function → local (let, const); global → var)

→ Hoisting: JS me aisa chalta hai variable use pehle kr lo declare baad me kro kyunki execution se pehle woo top of the code pe chala jata.
(only hoist declaration ✓ not initialization ✗ (undefined))

ex:
Console.log(num) — let, const = error
let num = 6; — var = undefined.