Javascript

→ It is said to be one of World's most Popular Programming language.

→ It is the Programming language of the web.

→ It is used to program the behavior of web pages.

During the beginning years of the web [1990's], web pages were just static, lacking the dynamic behavior after the page was loaded in the web browser.

So, To remove this limitation, in 1995, Netscape [Browser] decided to add a scripting language to it's Browser. Then, they hired Mr. Brendan Eich to get this job done.

Brendan Eich came up with a new language & it's interpreter implementation were called livescript, When it was first shipped as part of a Navigator beta in September 1995, the name was changed to Javascript for the official release in December.

The Javascript name had created confusions, implying that it is directly related to Java which was the Newly released trending, hot language in 1995.

This was a Marketing strategy by Netscape Corporation.

As of 2022, Javascript is used by 98% of the Websites use Javascript on the client side for the Webpage behavior.

* Javascript features

→ Scripting Language - doesn't require anything to Compile it for use. Scripting is easier to create, & can be done faster. It is Just-in-Time Compiled.

The instructions written for runtime environment. It is a high level Language which interpret & execute one command at a time.

→ Light weight - it is made for data handling at the browser only. it has a limited set of libraries. Also it is only meant for client-side execution & that too for web applications. [Minimal memory, simple syntax

→ Dynamically Typed - means the type of the variable are defined based on the stored value. For example, if you declare a variable x, then you can store either a string or a number or an array or an object.

→ Object-Oriented Programming - In JS, 2 important principle with OOP are Object creation patterns (Encapsulation) & Code reuse patterns (Inheritance). But JS developers rarely use these features.

→ Functional - JS uses Functional approach also, & functions in JS can be used as objects & can be passed to other functions too.

Platform Independent - Which means JS is portable. i.e, you can simply write the script once & run it anywhere & anytime. In general, you can write your JS applications & run them on any platform or any browser without affecting the output of the script.

Interpreted - Which means the script written is executed line by line. These scripts are interpreted by Javascript interpreter which is a built-in component of the web Browser.

Client side language - JS main tasks are validating input, animation, applying styles, manipulating UI elements in the client-side.

Server-side language - With Node.js you can write Javascript code on the server-side. Which means that JS can have backend access to databases, file systems & servers.

Rich-Interfaces - you can use JS to include items such as drag & drop components & sliders to give a rich interface to website/page visitors.

Versatility - with JS you can go beyond web apps, if you want to build servers, you can do it, if you're going to develop games, you can do that too.
For Instance, if you have a web app & you want to build mobile app as well, you can do that with React Native.

Javascript Applications

→ Web/Mobile Apps
→ Real-time Networking Apps
→ Games
→ Smartwatch Apps
→ Flying Robots
→ Server Applications

Let's start with the sample JS code.

⊙ `console.log('Hello World');`

For setting up development Environment
→ Download any Editor
  • VS code [optional = Node.js]

* To place JS code in HTML you need to use <script> tags & JS Code must be inserted between these tags.

ex:-

```
<script>
  document.getElementById("demo").innerHTML = "My Javascript";
</script>
```

→ JS code/scripts can be placed in the <head>, or in the <body> section (or) in both section of an HTML page.

ex:-

```
<head>
  <script> document.write...
                //comment
  </script>
</head>
```

```
<body>
  <script>
    . . .
  </script>
</body>
```

We can also scripts in external files

file: NewScript.js

```
function myfunction() {
    document.getElementById("demo").innerHTML = "Paragraph";
}
```

→ To use an external script, put the name of the script file in the src (source) attribute of <script> tag:

```
<script src="Newscript.js"></script>
```

⊙ Datatypes in Javascript

* numbers ⇒ 10, 3.14, 25, 3e-2
  it represents an integer or a floating-point number.
  → typeof(10.2) = 'number'

* String ⇒ 'hello', "Vikram!" [Concatenation "+"]
  it represents textual data

* Boolean ⇒ Any of two values: true or false

* undefined ⇒ let a;
  a datatype whose variable is not initialized

* null ⇒ let a = null;
  denotes a null value

* object ⇒ let student = {Name: Arun};
  key-value pairs of collection of data

* Symbol ⇒
```
Var sym1 = Symbol("Hi");
Var sym2 = Symbol("Hi");
Console.log(sym1 === sym2);
```

It represents a unique identifier. Even if we create many symbols with exactly the same description, they are different values.

→ Variables

⇒ It is a container for storing data.

- Declaring a variable
  - → Var [was the only keyword to declare variable until 2015.]
  - → let
  - → Const [let & const were added in 2015]
  - → Nothing

ex:-
```
let age;
Console.log(age); ⇒ Undefined
```

- Assignment. (Assigning data values using assignment operator '=')

```
let age = 22;
Console.log(age);
Console.log("Your age is", age);
```

```
var name = "Javascript";
console.log(name);
Console.log("Your variable value is", name);
```

Var

→ Variables defined with var can be redeclared

→ var keyword have functionscope

```
Var x = 10;
{
var x = 2;
}
console.log(x); ⇒ 2
```

let

→ variables defined with let can't be Redeclared.

→ let keyword have blockscope.

```
let x = 10;
{
let x = 2;
}
Console.log(x); ⇒ 10
```

Const

→ variables defined with Const can't be redeclared.

→ Variables defined with const can't be reassigned.

→ Const also have blockscope.

ex:
```
Const x = 10;
{
const x = 2;
}
Console.log(x); x is 2
```

[X incorrect]
```
Const PI;
PI = 3.14
```

Javascript can display data in following ways:-

→ Using console.log() method in the browser, mainly for debugging purposes.

ex:-
```
<body>
    <script>
        console.log(5+6);
    </script>
</body>
```

→ using document.write()

it must be only used for testing purposes.

```
<body>
    <h1>...</h1>
    <p>...</p>
<script>
    document.write(5+6);
</script>
</body>
```

→ Using innerHTML

To access an Html element, JS can use the document.getElementById(id) method. &the "id" attribute defines the HTML element.

innerHTML defines the HTML content.

ex:-
```
<body>
    <h1> ... </h1>
    <p> ... </p>
    <p id="demo"></p>
    <script>
        document.getElementById("demo").innerHTML = 7+8;
    </script>
</body>
```

→ Using window.alert()

You can display data in the form of an alert box:

ex:-
```
<body>
  <h1> ... </h1>
  <p> ... </p>
<script>
  window.alert("Warning");
</script>
</body>
```

- Comparison operators

difference between == & ===

→ ==: Normal equality operator in which values are important while comparing but not the datatype.

→ === :- strict equality operator in which both values & datatypes are important.

ex:- Console.log(10 == "10") ⇒ True

Console.log(10 === "10") ⇒ False

Console.log(true === 0) ⇒ False

⑥ NaN - "Not-a-Number"

ex:- (0* infinity)
("hi"/5)
(%)

NaN, in JS is not a legal number.

If the result of an operation is undefined then we get NaN.

Logica operators → && [logical and]
|| [logical or]
! [logical not]

Zero value => false & Non-zero => true
empty string (" ") => false, non-empty string => True
null => undefined, Nan => false

⑥ prompt() - method displays a dialog box that prompts the user for input. This method returns the input value if the user clicks "OK", otherwise it returns null. ex:- Var num = Number(prompt("Enter"));

Var x = prompt
Var y = prompt
Var z = x+y
alert(z)

⑦ Conditional statements
→ if, else, else if

```
Var num = Number(prompt("Enter a num"))   //str to number
if(num %2 ==0);
{
  console.log("even");
}
else {
  console.log("odd")
}
```

```
<Script>
var num=Number(prompt("Enter num"))
 if(num>0)
{
 console.log("+ve num")
}
else if(num<0)
{
 console.log("-ve num")
}
else {
 console.log("zero")
}
```

⑤ Arithmetic Operators

```
let x = 20;

x = x + 1;      or   x += 1;
x = x - 1;      or   x -= 1;
x = x * 2;      or   x *= 2;
x = x / 2;      or   x /= 2;
let NewX = x % 2;
```

```
let total = 1 + 3 * (8);
Console.log(total);
let total = 1+3 *(4+5);
console.log(total);
```

operator precedance [PEMDAS]
1. parenthesis
2. exponents
3. multi & div
4. add & sub

⑥ Iterative statements

1. While loop [to execute undefined times]
2. For loop [if we know how many times the loop must be executed]

syntax:-
```
while(condition)
{
  body
}
```

ex:-
```
var count=1
while(count<=10)
{
  console.log(count)
  count++
}
```

ex2:-
```
var s="pentagon space"
var i = 0
while (i< s.length)
{
  console.log(s[i])
  i++
}
```

```
for(initialize;condition check ;increment/decrement)
{
  ...
}
```

ex:3 To print all no. div by 2 & 5.

```
var n = 5
while (n<=100)
{
  if (n%2==0 && n%5==0)
  {
    consol.log(n)
  }
  n++
}
```

eg:- Repeating loop until user gives the desired input.

```
var name=prompt("Enter ur fav subject")
while (name!="python")
{
  name = prompt("Enter your fav subject")
}
alert("Thanks for choosing python")
```

⊙ for

eg:- 10 times

```
<script>
    for(var i=0; i<=10; i++)
    {
      console.log("Javascript")
    }
</script>
```

i=0, 0<=10 → true, "Javascript"

i=10; 10<=10 → true, "Javascript"

eg:- 2, even no's from 1 to 100

```
for(var i=1; i<=100; i++)
{
  if(i%2==0)
  {
    console.log(i)
  }
}
```

eg:- print from 0th index to last index one by one.

```
var s="pentagon space"
for(var i=0; i<=s.length-1; i++)
{
  console.log(s[i])
}
```

(Secret Agent)

⊙ problem statement :- write a program that satifies the conditions:-

1. The first char of name should be 's'.
2. The last char of fav bike should be "e".
3. The lucky number should be 7.
4. The length of dish should be >=7.

```
<script>
    var name = prompt("Enter your name")
    var bike = prompt("Enter your fav bike name")
    var num = Number(prompt("Enter your lucky number"))
    var dish = prompt("Enter a dish")

    var nameflag = false
    var bikeflag = false
    var numflag = false
    var dishflag = false
```

```
if (name[0] == 's')
{
    nameflag = true
}
if (bike[bike.length - 1] == 'c')
{
    bikeflag = true
}
if (num == 7)
{
    numflag = true
}
if (dish.length >= 7)
{
    dish flag = true
}
alert ("Hello: " + name + "\n Thank you for your information")
if (nameflag && bikeflag && numflag && dishflag == true)
{
    console.log ("welcome to our group")
}
else
{
    console.log ("you are an outsider")
}
</script>
```

⊙ Functions in Javascript

Syntax:-
```
function fun_name(parameters)
{ body
 return val
}
```

ex:-
```
function wish(){
        alert("Hello good morning")
}
wish()
wish()
```

```
function wish(name)
{
  alert("Hello "+name+"good nite")
}
n = prompt("Enter a name")
wish(n)
```

Default Arguments:
```
function wish(name = "friend")
{
alert("Hi" + " "+name+" "+"good morning")
}
wish("Ramu")
wish()
```

⊙
```
function add(x,y)
{
   z = x+y
   return z
}
sum = add(100,200)
alert("sum="+sum)
```

```
function fun(s)
{
  x = s[0].toUpperCase() + s.slice(1)
  return x
}
ans = fun("pentagon")
alert(ans)
```

⊙ factorial of a number

```
function fact(n) {
      num = 1
      for(var i=2; i<=n; i++)
      {
```

```
    num = num * i
}
    Looping statement
    Statement
return num
}
console.log("Fact(4) = " + fact(4))
```

n=4, i=2; 2<=4, num = 1*2 = 2
i=3, 3<=4, num = 2*3 = 6
i=4, 4<=4, num = 6*4 = 24
i=5, 5<=4,

Javascript Scope

- Global scope - The variables that are declared outside of functions are called global variables & the space where these variables are defined is called global scope.

eg:-
```
var x=10
function f1()
{
 console.log(x)
}
function f2()
{
 console.log(x)
}
f1()
f2()
```

- Local scope - The variables that are declared inside the function are called local variables & the space where these variables are defined is local scope. They are not available outside of the function.

eg:-
```
function f1()
{
 var z=10
 console.log(z) --> 10
}
function f2()
{
 console.log(x) --> error
}
f1()
f2()
```

⑤. Higher order functions

→ we can pass one function as an argument to another function & we can return functions from inside the function, such type of functions are called higher order functions.

- setInterval() method calls a function at specified intervals (in milliseconds)

syntax:-

```
setInterval(function, milliseconds, param1, param2, ...)
```

ex:-

```
function endless(){
    console.log("hi")
    Console.log("bye")
}
```

on console,

```
setInterval(endless, 3000)
```

⑥ Anonymous functions

Functions without Names.

ex:-

```
setInterval(function(){Console.log("Anonymous fn")}, 3000)
```

```
> clearInterval(1)
```

* Given a string with a non-negative int n, return a larger string that is n copies of the original string [using function].

fun("Hello", 2) → "HelloHello", fun("Hello", 3) → "HelloHelloHello"

```
ex:- function fun(str, n){
        result = ""
        var count = 1
        while(count <= n){
            result = result + str
            count ++
        }
        return result
    }
alert(fun("Hello", 3))
alert(fun("Hi", 4))
```

- Given 3 int values a, b, c return their sum. if one of the value is 13 then it does not count towards the sum & values to it's right do not count. for ex:- if b is 13, then b & c values do not count.

ex:- sum(1,2,3) → 6
sum(1,2,13) → 3
sum(1,13,2) → 1
sum(13,2,3) → 0

⇒
```
function sum (a, b, c) {
    if (a == 13)
    {
      return 0
    }
    if (b == 13)
    {
      return a
    }
    if (c == 13)
    {
      return a+b
    }
}
alert (sum(13,1,2))
alert (sum(10,13,2))
alert (sum(10,20,13))
```

- You are driving a little too fast & police stops you to fine you. write a code to compute the result, encoded as an int value: 0 → no ticket/fine, 1 → small/fine/ticket, 2 → big ticket/fine. if the speed is 60 or less, the result is 0. if the speed is in b/w 61-80 the result is 1, if the speed is >81, the result is 2, unless it's your birthday on that day your speed can be 5 higher in all cases.

⇒
```
function speed_indicator (speed, isBday) {
    if (isBday)
    {
      speed = speed - 5
    }
```

```
    if (speed <=60)
    {
      return 0
    }
  else if (speed >=61 && speed <=80)
    {
      return 1
    }
    else
    {
      return 2
    }
}
alert ("trial 1:" + speed_indicator(60, false)) //0
alert ("trial 2:" + speed_indicator(65, false)) //1
alert ("trial 3:" + speed_indicator(65, true)) //0
```

⑥ Arrow functions

- allows us to write shorter function syntax:

Normal function,

```
let x = function(x, y){
        return x * y;
    }
```

Can be written as,

```
let x = (x, y) => x * y;
```

using arrow function.

General Arrow function Syntax

```
let function = (para1, para2 ...) => {
        statement(s)
    }
```

- function - is name of the function
- arg1, arg2, ... here are the arguments you can pass.
- statements - function body

With Arrow function:-

```
hello = () => {
    return "Hello world!";
}
```

It gets shorter, if the function has only one statement, & the statement returns a value, you can remove the curly brackets & the return keyword. [If it's a single statement then Arrow functions returns value by default]

```
hello = () => "Hello Everyone!";
```

- if you have parameters pass them inside.

```
hello = (val) => "Hello" + val;
```

Javascript Arrays

```
var sub = ["python", "java", "C", "C++"]
sub[10] → undefined
var a = ["cricket", "football", "volleyball", "khokho"]
alert(a[0])
a[0] = "Hockey"   // updating array
alert(a)
a[4] = "Tennis"   // adding new elements to array
alert(a)
```

- creating empty array

eg①
```
var a = [ ]
a.[0] = "Football"
alert(a) → [Football]
```

eg②
```
var a = new Array()
a[0] = "cric"   alert(a)
a[1] = "Fut"    alert(a)
a[2] = "Hock"
alert(a)
```

- To check length / elements of array.

→ alert(a.length)

Javascript can also hold heterogenous elements also

```
var a = ["cric", 10, football, 20]
alert(a)
```

Methods in JS Arrays.

- → push()
- → pop()
- → unshift()
- → shift()
- → indexOf()
- → slice()

→ push() - This method adds an element at the end of the array. and this method also returns the length of the array after addition of elements.

ex:
```
var a = [10,20,30,40]
X = a.push(50)
alert(x)   // length 5
alert(a)   // updated list
```

→ pop() → Removes & returns last element of the array.

```
var a = [10,20,30,40]
  x = a.pop()
  Console.log(x)   //40
  y = a.pop()
  alert(y)   //30
  alert(a) = [10,20]
```

→ unshift() → To add element at the first (0th) index

```
var a = [10,20,30,40]
 a.unshift(5)   // reverse of push()
 alert(a)
```

→ shift() → removes & returns first element

```
var a = [10,20,30,40]
x = a.shift()
alert(x)   // 10
alert(a)   // 20,30,40
```

→ indexOf() → To get the index of specified element

```
var a = [10,20,30,40]
alert(a.indexOf(30))   //2
```

⊙ if the specified element is not found, indexOf() returns -1.

→ slice() - used to (get) extract a part of an array [syntax = slice(start,end)]

slice(start,end) → returns elements from start to end-1.

slice() → it returns Complete array

- var a = [10,20,30,40,50,60,70,80]
  alert(a.slice()) // 10,20,30,40,50,60,70,80
  alert(a.slice(1,5)) // 20,30,40,50

⇒ Multidimensional Array - Array inside an array.

```
var a = [[10,20,30],[40,50],[60,70,80]]
alert [a[0]]
alert (a[1][0])
alert (a[2][2])
```

realtime ex:-

```
var books = [ ]
var input = Number(prompt("Enter 1->add, 2->display, 3--> exit,"
                          4--> remove))
while (input != 3)
{
  if(input==1)
  {
    var newBook = prompt("Enter name of the book")
      books.push(newBook)
  }
  else if(input ==2)
  {
    consalertag("Available books are", books)
  }
  else if(input==4){
    x=books.pop()
    alert("Removed books:"+x)
  }
  else
  {
    consalertag("Invalid option")
  }
  input = Number(prompt("Enter 1->add, 2->display, 3->exit;"))
                                                  4->remove"))
}
print{
consalertag("Thank you")
```

- for-of loop → A simple way to extract elements one by one from an array

```
var a = [10,20,30,40,50]
for(n of a)
{
  alert(n)
}
```

- forEach loop → syntax => arrayObj.forEach(function) [This method calls a function for each element in an array]

```
var a = [10,20,30,40,50]
function fun(x)
{
  alert(x)
}
a.forEach(fun)
```

Javascript objects

JS objects - store information in the form of key-value pairs.

```
var emp = {name:"Vikram", eid:100, place:"Blr"}
console.log(emp["name"])
console.log(emp["eid"])
```

Another way - to create object,

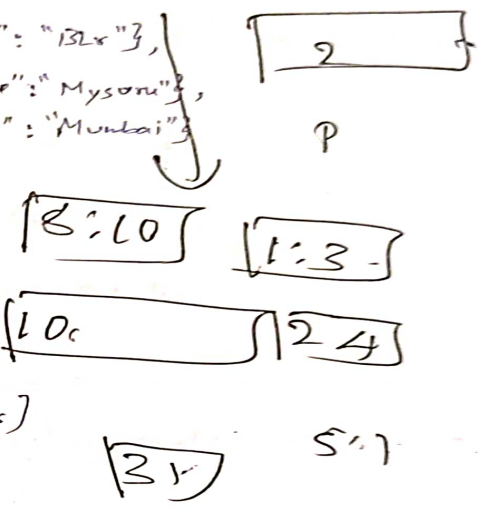
```
var emp = new Object()
emp["name"] = "Rahul"
emp["id"] = 15
emp["age"] = 25
console.log("emp")
emp["id"]= 101
console.log(emp["id"]) (or) console.log(emp.id)
```

You can use, for in loop

ex:-
```
var emp = {"name": "vikram" , "eid": "101 ;"place":"Blore"}
for (key in emp)
{
   alert(key + ":" + emp[key])
}
```

- objects inside an array

```
var emp = [ {"name": "virat", "id": 100, "place": "Blr"},
            {"name" : "vinay", "id": 101 , "place": "Mysoru"},
            {"name" : "varun" , "id":102 , "place" : "Mumbai"}
          ]
console.log (emp[0]["name"])
console.log (emp[1]["id"])
console.log (emp[2]["place"])
```



[objects can also have methods inside it]

- object methods. (functions inside the objects)

```
var emp = { name: "Virat",
            id : 55 ,
            age: 34,
            fun : function ()
             {
               alert("emp name =" + name)
             }
          }
emp.fun ()    => object method.
```

this - keyword, inside the object method, if we want to access object properties then we should use this keyword.

ex:-
```
fun : function ()
   {
   alert ("emp name =" + this.name)
   }
```

ex:-

```
fun : function()
{
    alert ("student id =" + this.id)
    alert ("emp age = " + this.age)
}
```

① Document Object Model [DOM] - is a interface between HTML, CSS & JS.

document.location
console.dir(document)

with DOM,

→ Javascript can change all HTML elements in the page.
→ JS can change all HTML attributes in the page.
→ JS can change all css styles in the page.
→ JS can also add new HTML & elements & attributes & also remove existing HTML elements & attributes.

☐ we can access HTML elements through DOM

1. document.URL → This is original URL of the website.
2. document.body → This returns everything inside body.
3. document.head → This returns head of the page.
4. document.links → This returns list of all links of the page.
5. document.getElementById() - returns element with specified id.
6. document.getElementByClassName() → returns list of elements of specific class.
7. document.getElementByTagName() → returns list of elements of specific Tags.
8. document.querySelector() → returns the first object matching css style selector.
9. document.querySelectorAll() → returns all object that matches css selector.

or:-

```
<h1> DOM example</h1>
<p>
.
.
.
</p>
<h2> fav Subs</h2>
<ul>
  <li> Java</li>
  <li> Pyth</li>
  <li> C#</li>
  <li id="How"> Javascript</li>
</ul>
<h2> Fav games</h2>
<li class="class-one"> Cricket</li>
<li> class Football</li>
<li> Tennis</li>
</ul>
<a href="..pentagon space</a>
</body>
<script>
</script>
</html>
```

```
document.URL
.
.
document.body
.
.
document.head
document.links
document.getElementById('id-one')
.
.
document.getElementsByClassName('class-one')
.
.
.
document.getElementsByTagName('h2')
.
.
document.querySelector("#id-one")
.
.
.
document.querySelectorAll('.class-one')
.
.
.
.
```

querySelector() & querySelectorAll()

→ if the specific css maps with only html element then we should use querySelector(). if multiple html elements match, then querySelector returns only first matched element.

We use querySelectorAll() - to access all matched html elements. if multiple html elements are there, then we use this method.

→ DOM based JS to change color of h1 tag.

```
var myh1 = document.querySelector('h1')
myh1.style.color = "red"
```

- Javascript HTML events

```
<head>
    <title> . . </title>
<body>
    <h2> Javascript HTML events </h2>
    <h2  onclick = "this.innerHTML = "Magic happens" style = "color: blue;"> Click
        on me </h2>
</body>
</html>
```

Onclick → event that gets executed upon clicking an element.
innerHTML → to change the content of HTML element.

e.g.:-

```
<h2  onclick = "fun(this)" style = "color:green;"> Click on Me </h2>
<script>
    function fun(id){
        id.innerHTML = "Magic Happens"
    }
</script>
```

addEventListener()

```
<h2> Event </h2>
<button id = "one"> Try it</button>
<script>
    document.getElementById("one").addEventListener("click", function(){alert(
                                                    "Hello pantagan")})
```

→ It attaches an eventhandler to an element. When using addEventListener(), JS is separated from the html for better readability.

[addEventListener(event, function)

here, function refers to action that should take place when a particular event gets generated.

To change the style of an HTML element,

Syntax:

[document.getElementById(id).style.property = new style]

```
<h1> login form </h1>
<form>
    <h2 style="color:red"> Please enter your details </h2>
    Email : <input type="text" id=1><br><br>
    Password : <input type="text" id = 2><br><br>
<button type = "button" onclick = "login_details(document.getElementById(1).value,
        document.getElementById(2).value)" > Submit </button>
</form>
<script>
    function login_details(email,pwd)
    {
      var users = [{email: 'sharan@gmail.com',
                    pwd : 12345,
                    firstname : 'sharan',
                    lastname :'bLr'
                   },
                  {email: 'shreekanth@gmail.com'
                   pwd : 123456,
                   firstname: 'shreekanth',
                   lastname : 'Mysi'
                   },
                  {email: 'sachin@gmail.com',
                   pwd : 1234567,
                   firstname : 'sachin',
                   lastname : 'Tendulkar'
                  }]
      var ispwdCorrect = false
      var isUserFound = false
      for (currentUser in users){
          if (users[currentUser]['email'] == email){
          if (users[currentUser]['pwd'] == pwd)
          {
              ispwdCorrect = true
              isUserFound = true
              break
          }
          else
          {
              isUserFound = true
              break
          }
      }
```

```
        else { isUserFound = false }
        }
        if (isUserFound = true && isPwdCorrect == false)
        {
        alert("invalid pwd")
      }
else if (userFound == true && isPwdCorrect == true)
    {
      alert("you are logged in!")
    }
    else
    {
      alert("Invalid user")
    }
  }
</script>
```