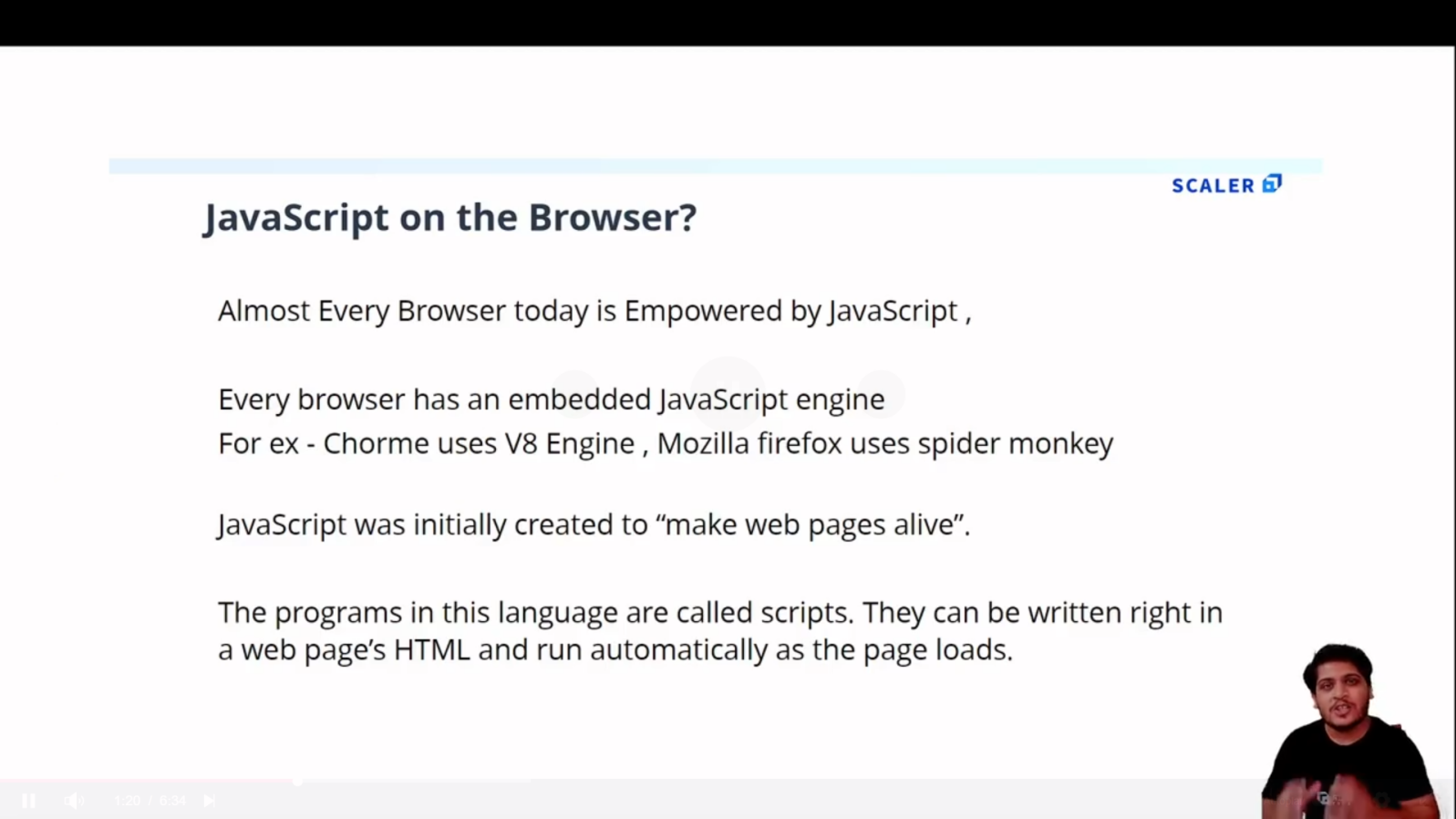
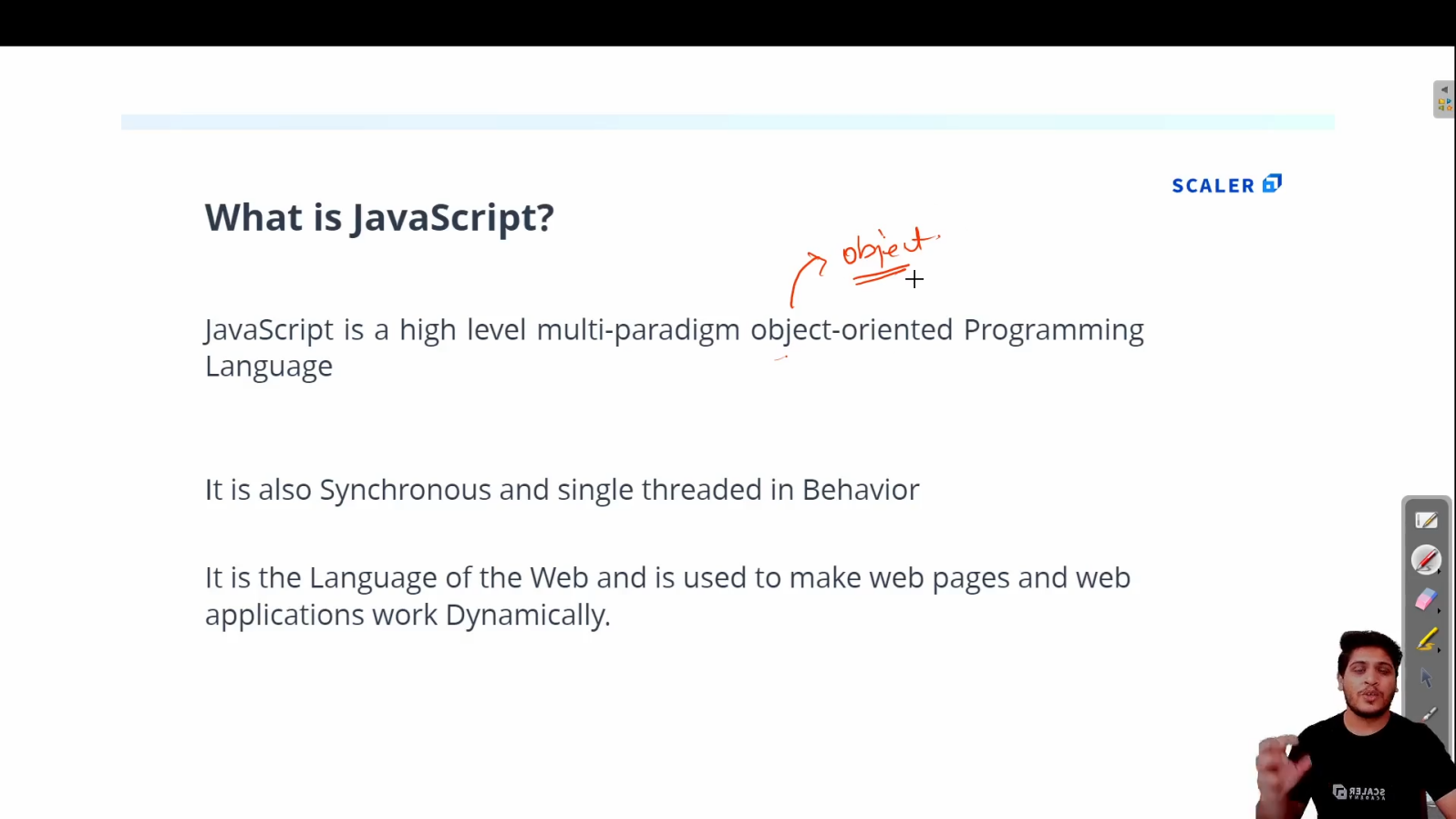
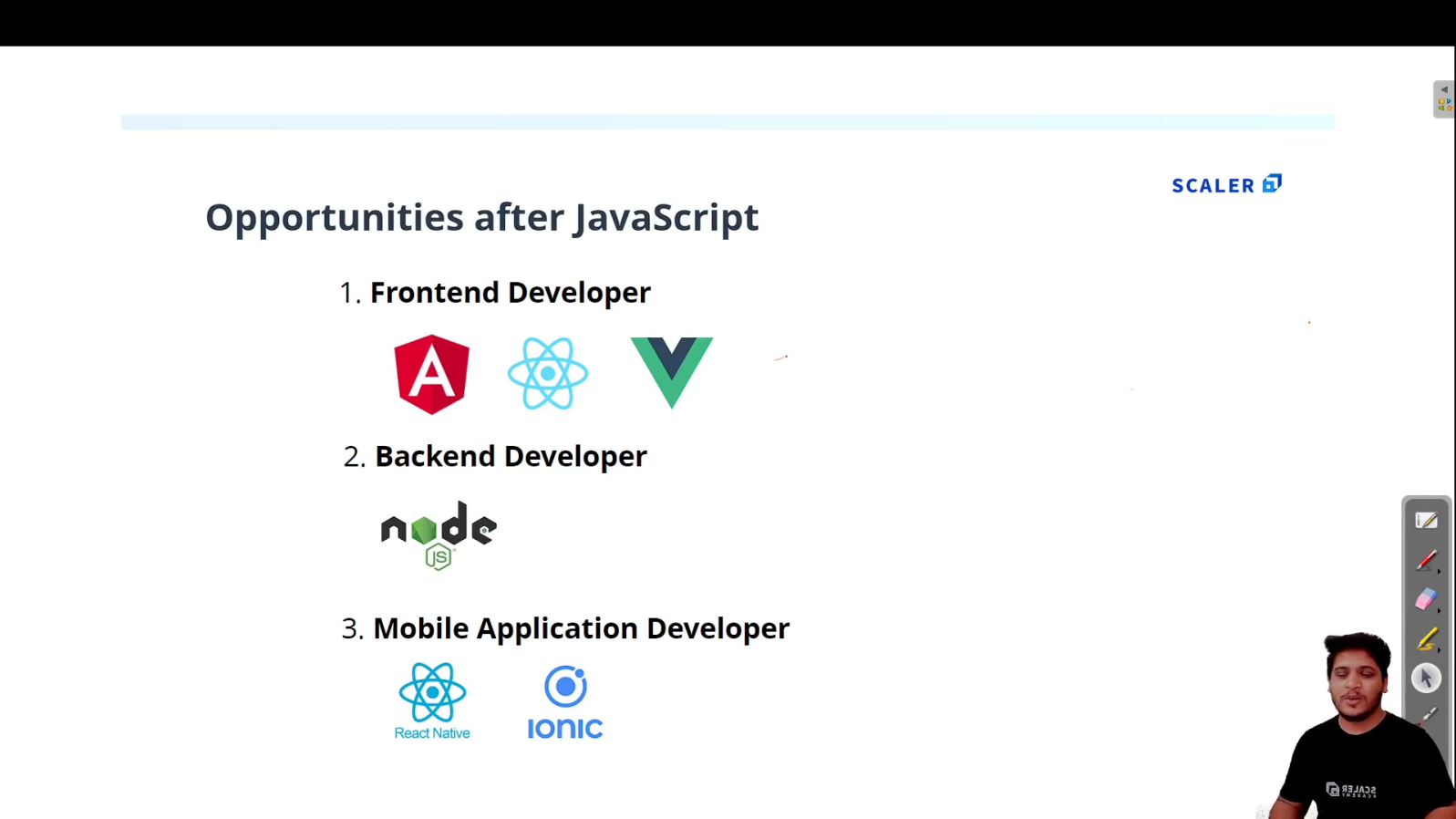
**JAVASCRIPT-Scalar**

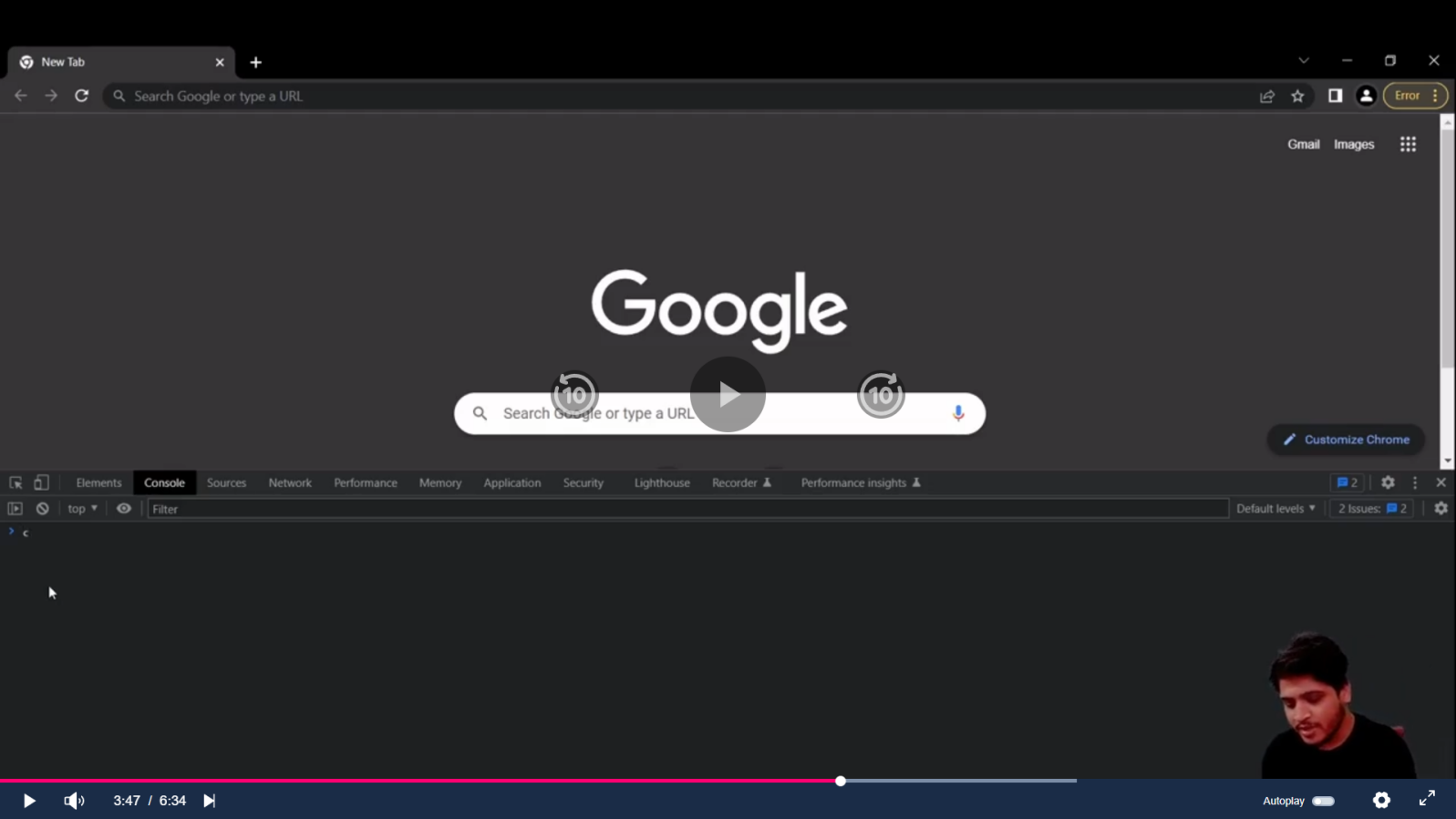
1. **Introduction:**

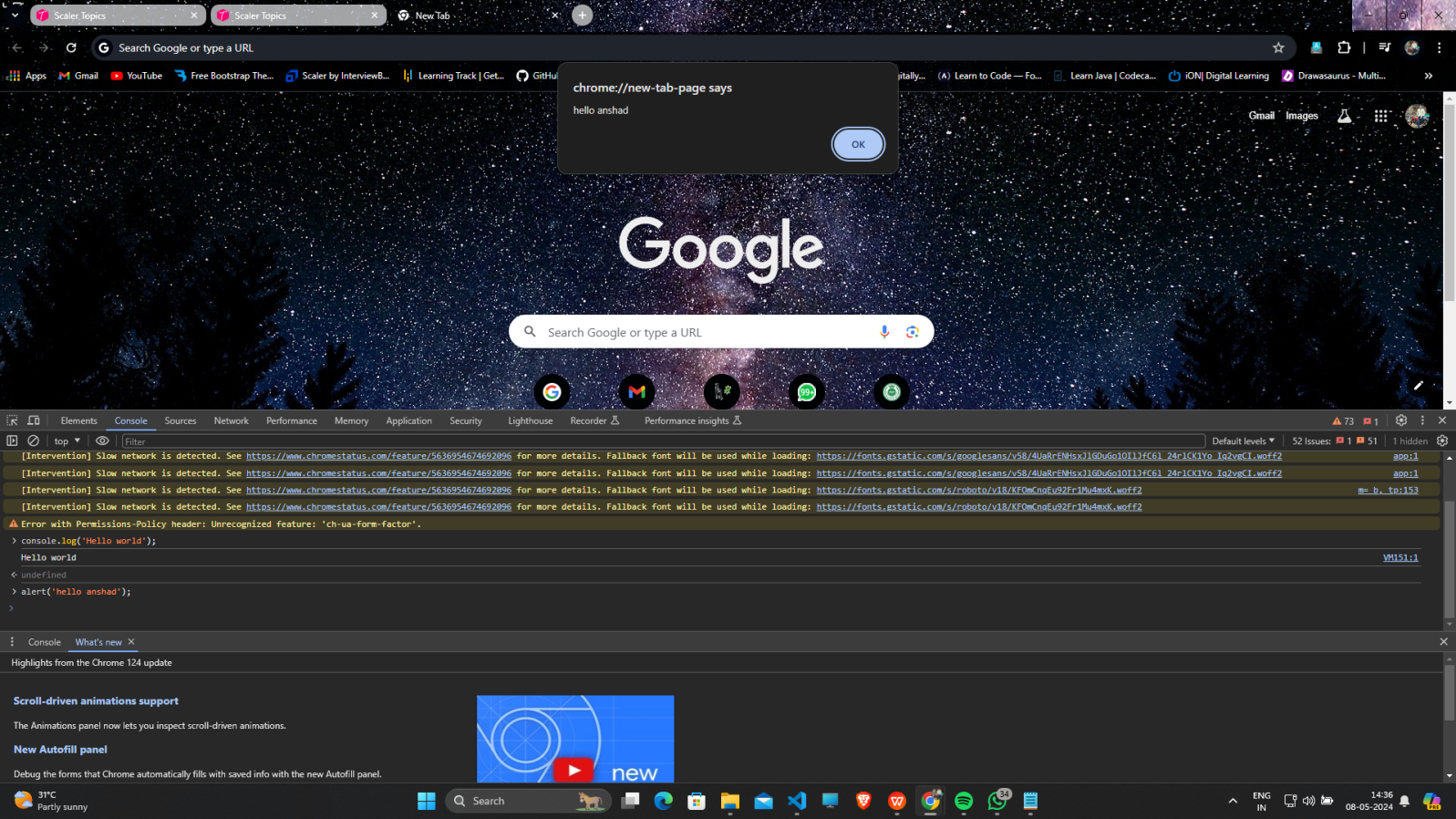
****

****

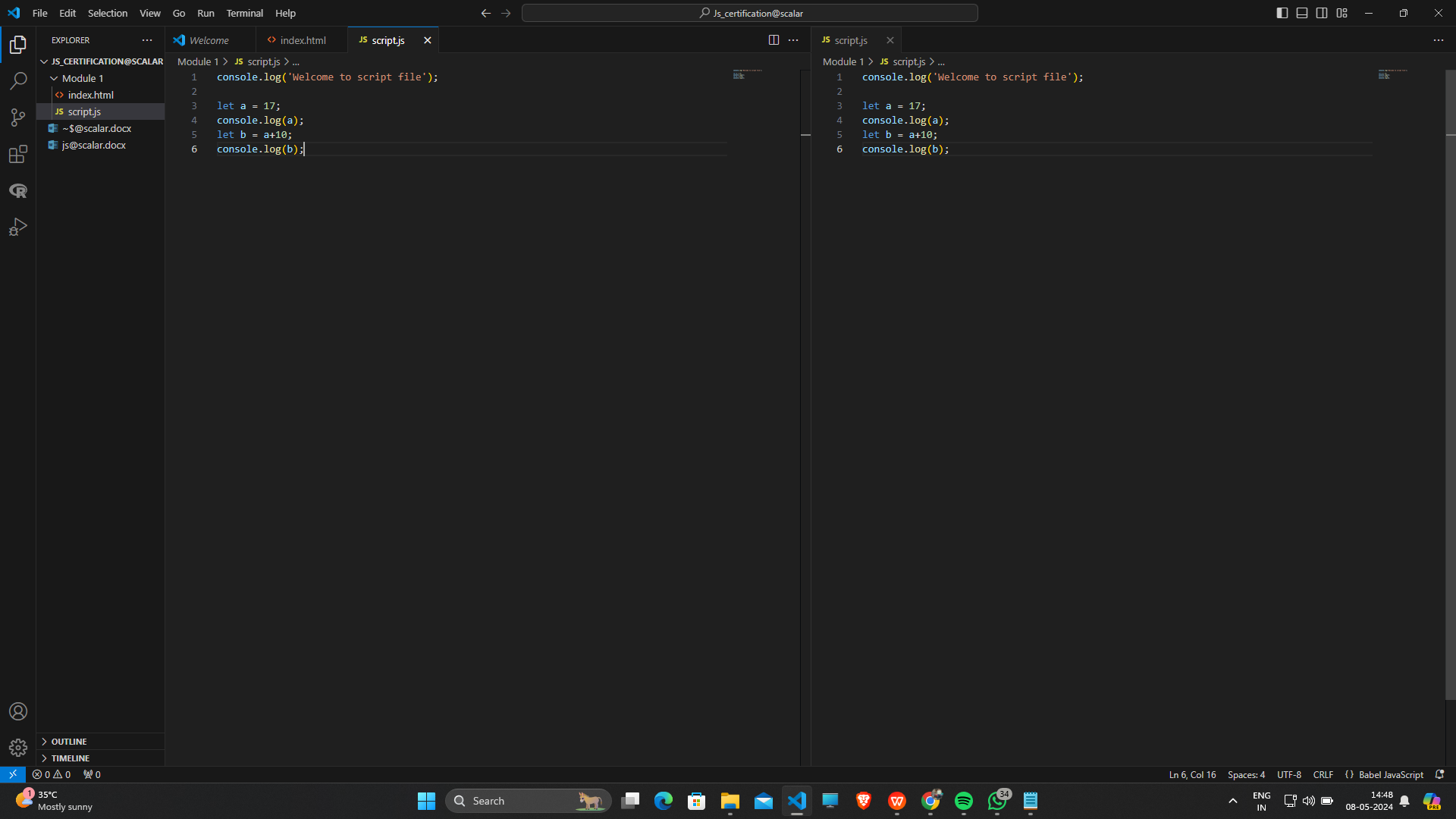
****

****

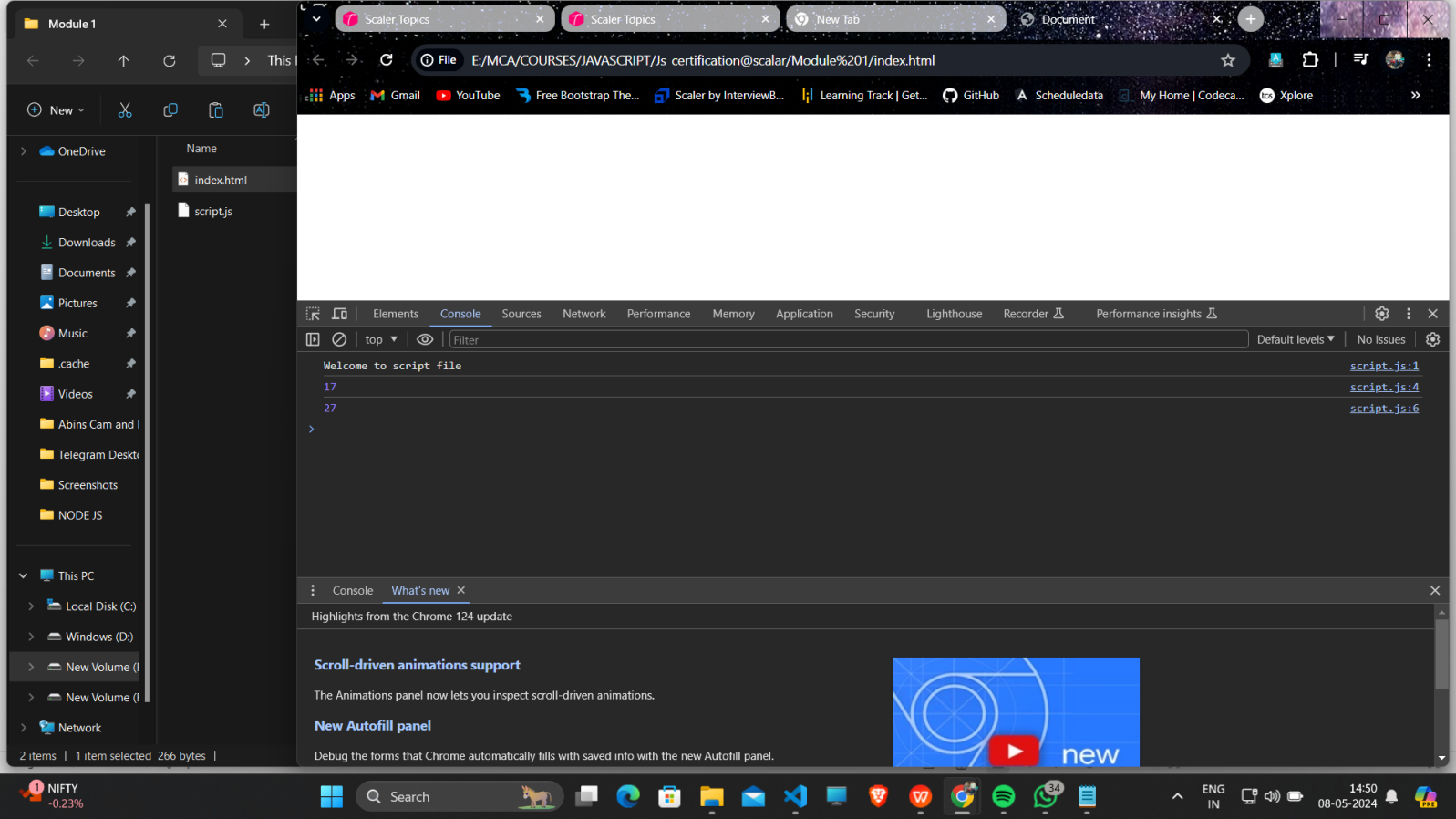
****

****

1. **First JS program :**
   1. **Create index.html and script.js file**

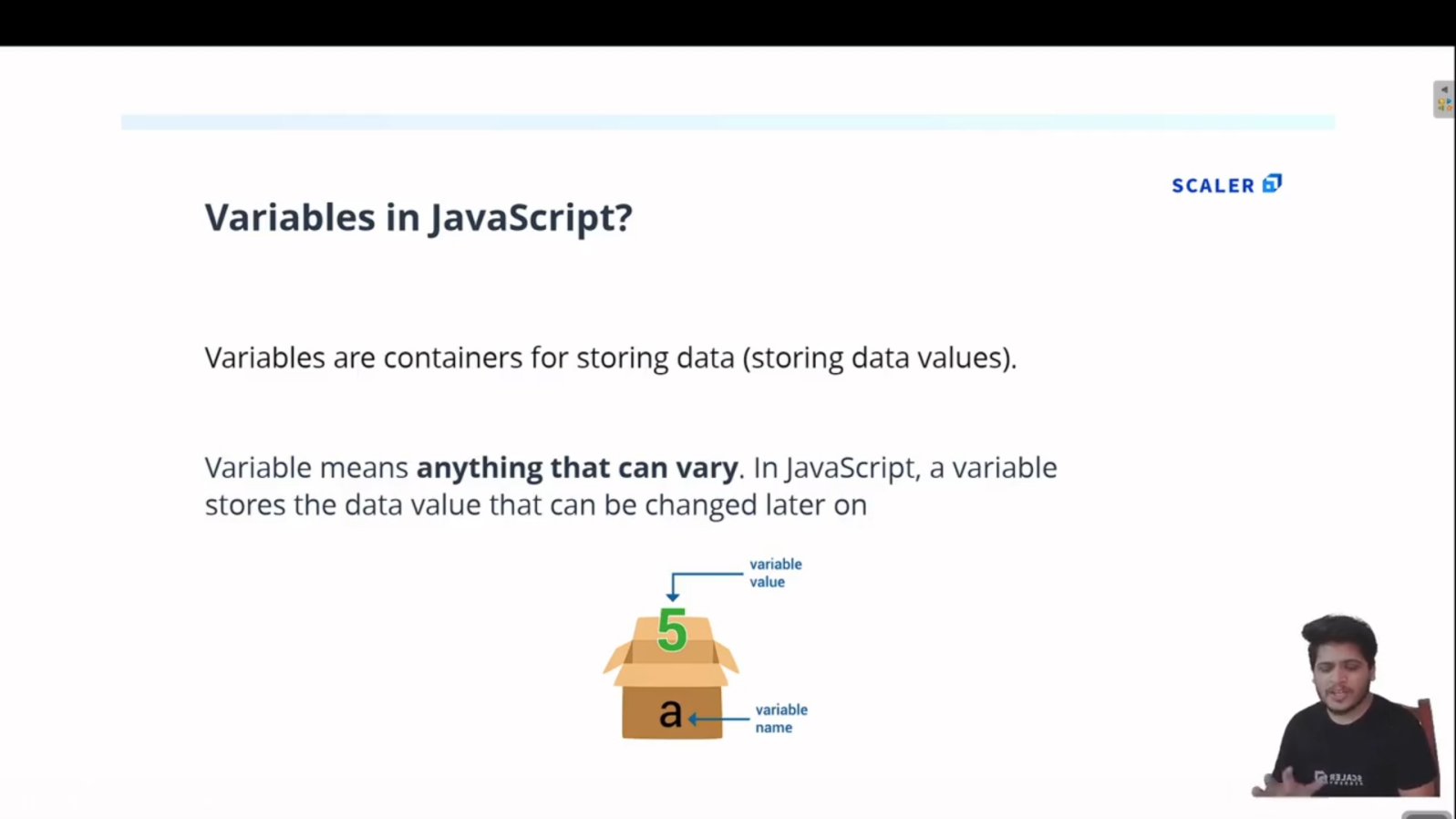
****

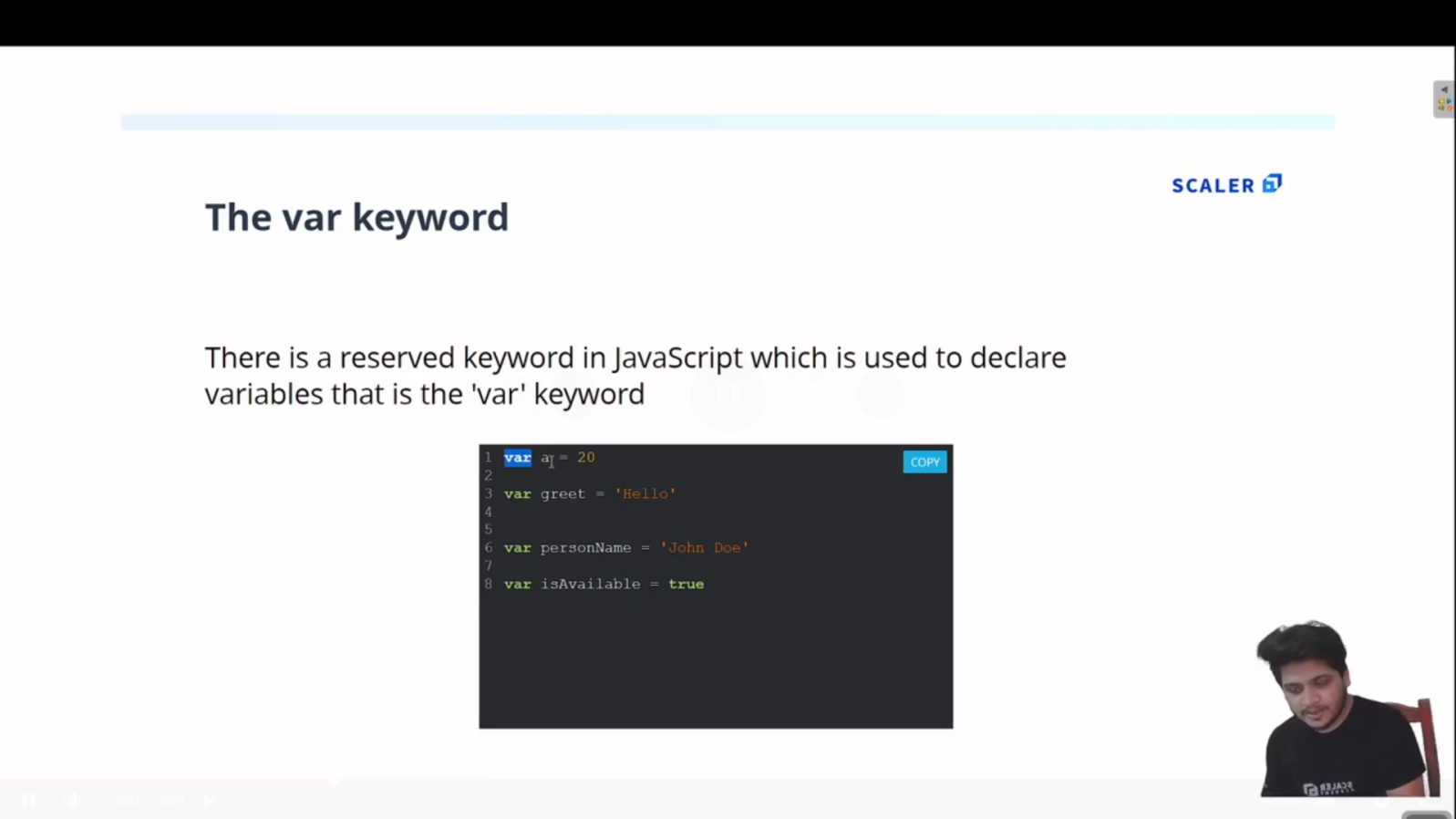
* 1. **To see Output ,open that index.html and inspect and go to console option :**

****

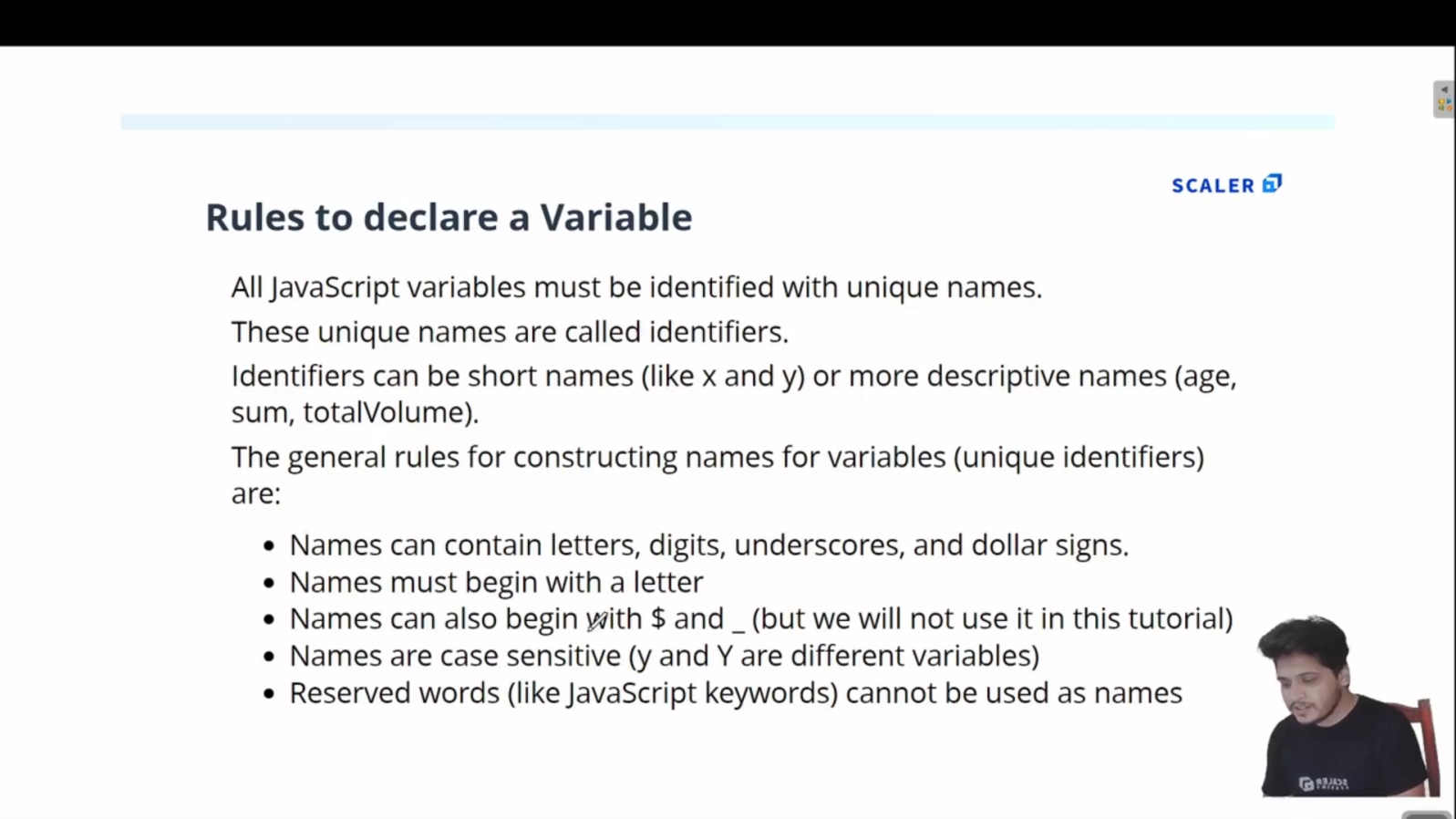
* 1. **Add an “Live Server” Extension to VScode so you can right click and select Open with Live Server ,which will directly take you to browser.**

1. **Variables in JS :**

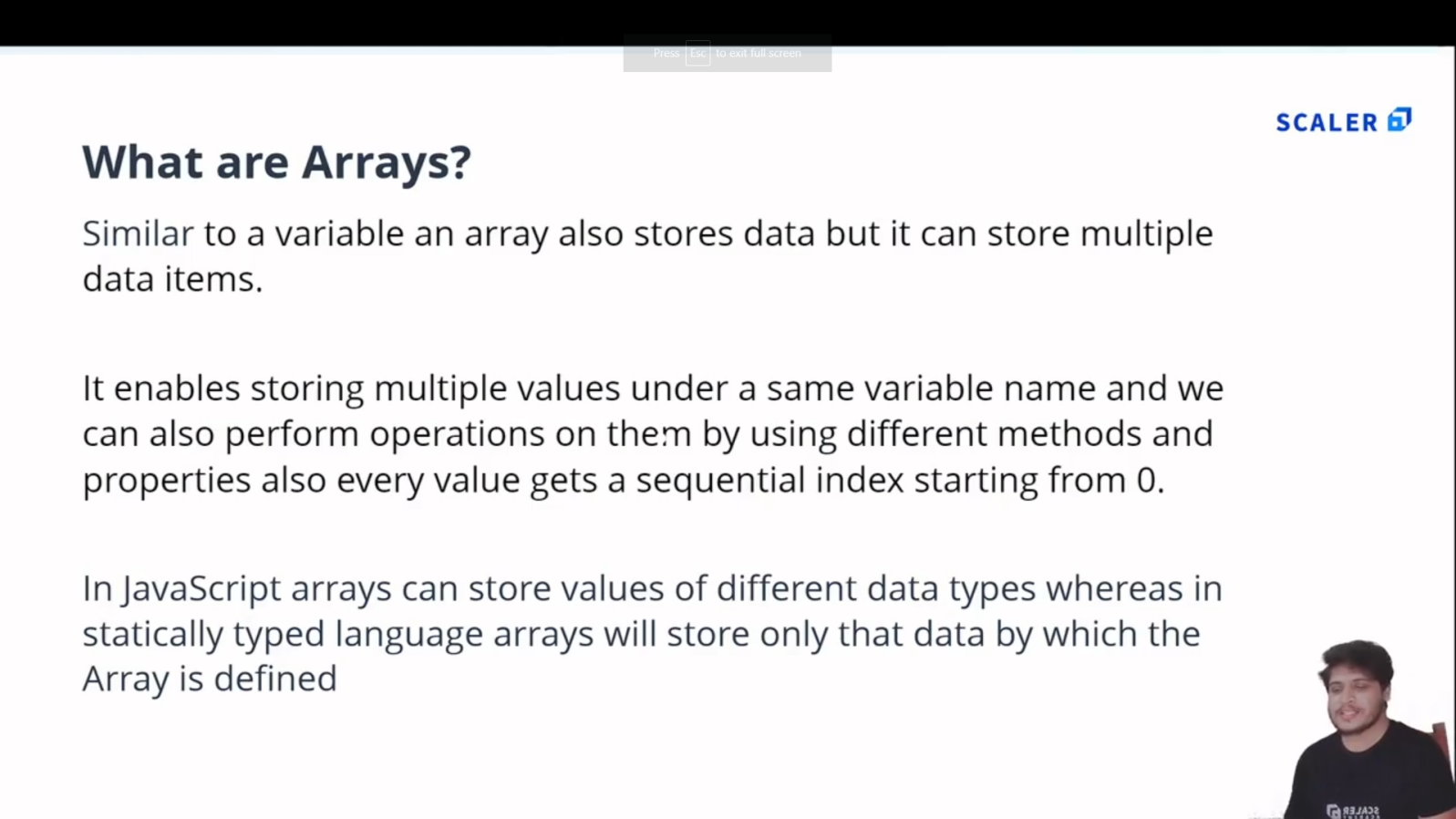
****

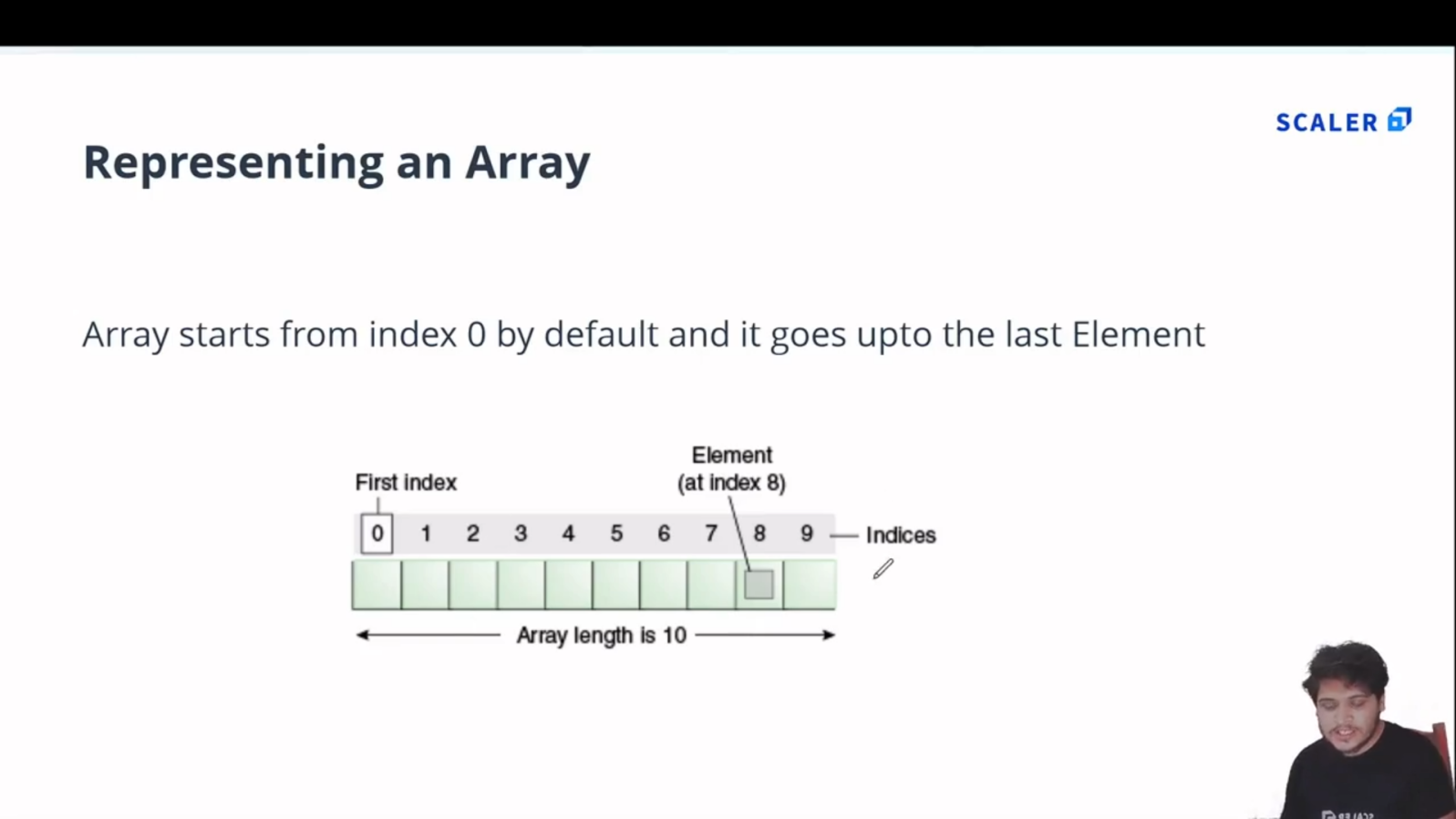
****

****

****

1. **Arrays in JS:**

****

****

* 1. **Example:**

//Array : In JS Array can store multiple values of differnt datatypes

var arr = [12, 'Ferrari' , true , 12.4];

console.log(arr);

//Accessing Elements

var a = arr[0];

console.log(a);

//replacing array elememt

arr[1] = 'Bently';

console.log(arr[1]);

//length property

console.log('Length of array is : '+arr.length);

**Output:**

[Running] node "e:\MCA\COURSES\JAVASCRIPT\Js\_certification@scalar\Module 2\array.js"

[ 12, 'Ferrari', true, 12.4 ]

12

Bently

Length of array is : 4

* 1. **Output:**

1. **Array Methods :[pop() , push() , shift() , unshift()]**

//------Inbuilt Array methods------------

//1.pop() method [REMOVES ELEMENT FROM END OF AN ARRAY]:

var arr2 = [12 , 14 , 56 , 57];

console.log('Array : '+arr2);

var c = arr2.pop();

console.log('Popped Element : '+c);

console.log('Array after popped : ', arr2);

//2.pop() method [PUSHES ELEMENT TO THE END  OF ARRAY]:

arr2.push(100);

console.log('\nArray after Push  : ', arr2);

//3.shift() method [REMOVES ELEMENT FROM START OF AN ARRAY]:

var d = arr2.shift();

console.log('\nShifted : ' + d);

console.log('Array after shift : ', arr2);

//4.unshift() method [PUSHES ELEMENT TO THE BEGINNING  OF ARRAY]:

arr2.unshift(102);

console.log('\nArray after Unshift : ', arr2);

**OUTPUT:**

[Running] node "e:\MCA\COURSES\JAVASCRIPT\Js\_certification@scalar\Module 2\tempCodeRunnerFile.js"

Array : 12,14,56,57

Popped Element : 57

Array after popped :  [ 12, 14, 56 ]

Array after Push  :  [ 12, 14, 56, 100 ]

Shifted : 12

Array after shift :  [ 14, 56, 100 ]

Array after Unshift :  [ 102, 14, 56, 100 ]

[Done] exited with code=0 in 0.11 seconds

1. **Objects in JS :**
   1. **In JS Objects are in Key-Value pair:**

|  |  |
| --- | --- |
| **Code** | **Output** |
| //In JS Objects are in Key-Value Pair  var person = {      firstName : 'Muhammad',      secondName : 'Anshad',      age : 25 ,      ownsCar : false  }  console.log(person); | {    firstName: 'Muhammad',    secondName: 'Anshad',    age: 25,    ownsCar: false  } |

* 1. **Accessing values from Object :**

|  |  |
| --- | --- |
| **Code** | **Output** |
| //-----Accessing Values from Object ------  //1.Dot notation :  console.log(person.age);  //2.Bracket notation  console.log(person['firstName']); | 25  Muhammad |

* 1. **Objects can have Arrays and Objects can be Nested:**

|  |  |
| --- | --- |
| **Code** | **Output** |
| //An Object can have an Array inside it.  //Nested Objects are possible in JS .  var cap = {      firstName : 'Steve',      lastName : 'Rogers',      age : 102 ,      friends : ['Bucky Barnes' , 'Bruce Banner' , 'Tony Stark'],      isAvenger : true ,      address : {          state : 'New York',          city : {              name : 'Brooklyn',              pincode : 123456          }      }  }  //To Access 'Bruce Banner'  :  console.log(cap.friends[1]);  //To Access 'Brooklyn' :  console.log(cap.address.city.name);  //Updating isAvenger :  cap.isAvenger = false;  console.log(cap.isAvenger);  //ADD to  object  :  cap.movies = ['age of ultron','civil war','first avenger'];  console.log(cap);  //DELETE from Object :  delete cap.age;  console.log(cap); | Bruce Banner  Brooklyn  False  {    firstName: 'Steve',    lastName: 'Rogers',    age: 102,    friends: [ 'Bucky Barnes', 'Bruce Banner', 'Tony Stark' ],    isAvenger: false,    address: { state: 'New York', city: { name: 'Brooklyn', pincode: 123456 } },    movies: [ 'age of ultron', 'civil war', 'first avenger' ]  }  {    firstName: 'Steve',    lastName: 'Rogers',    friends: [ 'Bucky Barnes', 'Bruce Banner', 'Tony Stark' ],    isAvenger: false,    address: { state: 'New York', city: { name: 'Brooklyn', pincode: 123456 } },    movies: [ 'age of ultron', 'civil war', 'first avenger' ]  } |

1. **Conditional Statements in JS (reference:codecademy):**
   1. If statement.

let sale =true;

sale=false;

if (sale) {

  console.log('Time to buy!');

}

* 1. If...Else Statement.

let sale = true;

sale = false;

if(sale) {

  console.log('Time to buy!');

}

else{

  console.log('Time to wait for a sale.');

}

* 1. Comparison Operators.(<,>,<=,>=,===,!==)

let hungerLevel = 7;

if (hungerLevel > 7){

  console.log('Time to eat!');

}

else{

  console.log('We can eat later!');

}

* 1. Logical Operators. (&&,||,!)

let mood = 'sleepy';

let tirednessLevel = 6;

if(mood === 'sleepy' && tirednessLevel > 8){

  console.log('time to sleep');

}

else{

  console.log('not bed time yet');

}

* 1. Truthy and Falsy.

let myVariable = 'I Exist!';

if (myVariable) {

   console.log(myVariable)

} else {

   console.log('The variable does not exist.')

}

* (The code block in the if statement will run because myVariable has a truthy value,even though the value of myVariable is not explicitly the value true).
* **The list of falsy values includes**:

[ 0, Empty strings like "" or '' ,null ,undefined ,NaN ].

let wordCount = 1;

if (wordCount) {

  console.log("Great! You've started your work!");

} else {

  console.log('Better get to work!');

}

let favoritePhrase = '';

if (favoritePhrase) {

  console.log("This string doesn't seem to be empty.");

} else {

  console.log('This string is definitely empty.');

}

* 1. Truthy and Falsy Assignment.
* The code below checks if username is defined and assigns a default string if it is not:

let username = '';

let defaultName;

if (username) {

  defaultName = username;

} else {

  defaultName = 'Stranger';

}

console.log(defaultName);

OR use the || operator :

let username = '';

let defaultName = username || 'Stranger';

console.log(defaultName);

* 1. Ternary Operator.

Example1:

isNightTime ? console.log('TRUE') : console.log('False section');

Example.2:

let favoritePhrase = 'Love That!';

favoritePhrase === 'Love That!'

?

  console.log('I love that!')

:

  console.log("I don't love that!");

* 1. Else If Statements.

let season = 'summer';

if (season === 'spring') {

  console.log('It\'s spring! The trees are budding!');

}

else if(season === 'winter'){

  console.log('It\'s winter! Everything is covered in snow.');

}

else if(season === 'fall'){

  console.log('It\'s fall! Leaves are falling!')

}

else if(season === 'summer'){

  console.log('It\'s sunny and warm because it\'s summer!');

}

else {

  console.log('Invalid season.');

}

* 1. The switch keyword.

let athleteFinalPosition = 'first place';

switch (athleteFinalPosition){

  case 'first place':

    console.log('You get the gold medal!');

    break;

  case 'second place':

    console.log('You get the silver medal!');

    break;

  case 'third place' :

    console.log('You get the bronze medal!');

    break;

  default :

    console.log('No medal awarded.');

    break;

}

1. **The For Loop :**

|  |  |
| --- | --- |
| **Code** | **Output** |
| //Loops are used to do some repetative tasks and used to control a flow of program  var a = 'Hello World'  //The for loop  for( var i = 0; i < 10 ; i++){      console.log(a);  }  //You have an Array and you have to square each element of that array:  var num = [2 ,3 ,4, 5 ,6 ,7,8];  var squaredArr = [] ; //empty array  for(var i = 0; i < num.length ;i++){      squaredArr.push(num[i]\*num[i])  }  console.log(squaredArr); | Hello World  Hello World  Hello World  Hello World  Hello World  Hello World  Hello World  Hello World  Hello World  Hello World  [     4,  9, 16, 25,    36, 49, 64  ] |

1. **While and Do..While Loop :**

//We have to print 1 to 10 using while loop :

//While Loop

var i = 1;

var n= 10;

while(i <= 10){

    console.log(i);

    i++;

}

//Do..While Loop -[Do while loop will always execute once and then checks condition ]

var i = 1;

var n= 10;

do{

    console.log(i);

    i++;

}while(i <= 10);

1. **For in Loop :(to loop through Objects and Arrays)**

//In JS, for-in Loop allows you to loop through the properties of an Object.

//for-in loop with Objects:

var colors = {

    primary : 'Blue',

    seccondary : 'Red',

    tertiary : 'White'

}

for(var color in colors){

    console.log(color + ' -> '+colors[color]);//gets the 'key' and 'value' of object

}

//for-in loop with Arrays :

var colorsArray = ['Yellow','Green','Orange','Pink'];

for(var color in colorsArray){

    console.log(color + ' -> '+colorsArray[color]);//gets the index and value

}

1. **For of Loop :**

|  |  |
| --- | --- |
| **Code** | **Output** |
| //The for-of statement creates a loop iterating over iterable object .  //including : built-in String ,Array, array-like objects like NodeList and also map and set.  var scores = [60 , 90 ,80 ,75]  //In for-of method,this 'score' variable have the element of Array.  //Where for-in method 'score' variable specify the index of Array  for( var score of scores){      console.log(score)  }  //method - entries()  let colors = ['Red ', 'White','Green'];  for(var [index,color] of colors.entries()){      //entries() method will carry index on first value and gives values of second value      console.log(index + '->'+color);  }  //Strings  var str = 'Scalar';  for(var c of str){      console.log(c);  } | 60  90  80  75  0->Red  1->White  2->Green  S  c  a  l  a  r |

1. FUNCTIONS in JS **:**
   1. Basic Function :

//Basic function in JS:

function greet(){

    console.log('Hello');

}

greet();//Calling function

//Parameters and Arguments :

function add(a , b){

    console.log(a+b);

}

add(2,3);

* 1. Functions as Expressions :

//Functions as Expression :

var add = function(a , b){

    console.log(a+b);

}

add(3,4);

//

var add = function(a , b){

    return a+b;

}

var sum = add ;

var result = sum(3,4);

console.log(result);

* 1. Hoisting :

//Hoisting : the process whereby the interpreter appears to move the declaration of functions,

//variables, classes, or imports to the top of their scope, prior to execution of the code.

console.log(a);//Here in JS this wont return an Error but shows 'Undefined'.

greet(); //returns hello from Anshad bcs functions can be called before or after its definition

var a = 'hi';

function greet(){

    console.log('hello from Anshad');

}

//You cannot call a Function Expression Also(below):

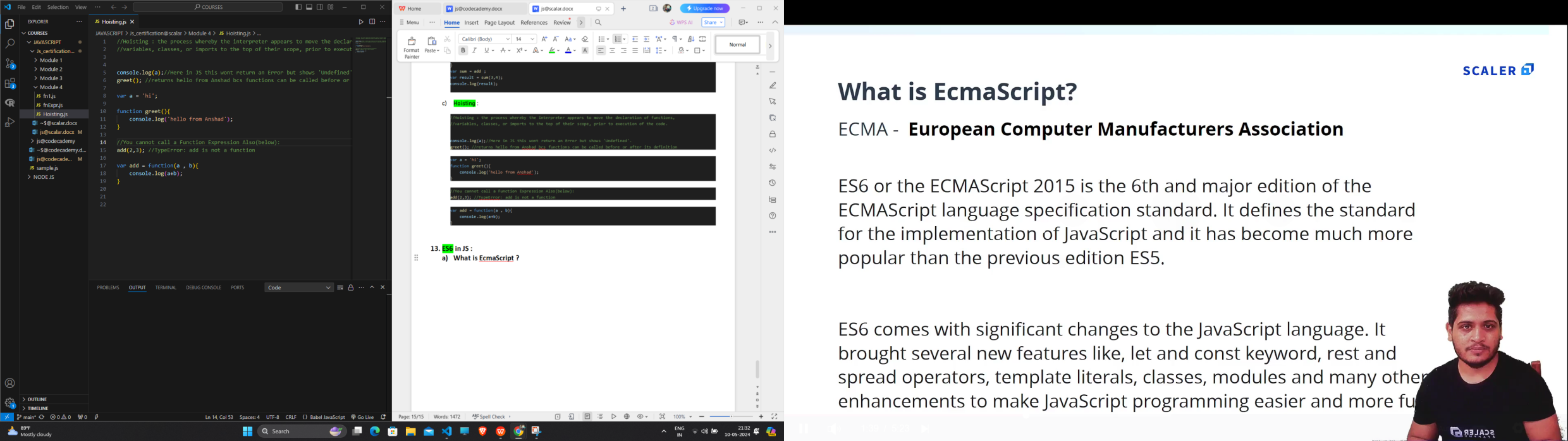
add(2,3); //TypeError: add is not a function

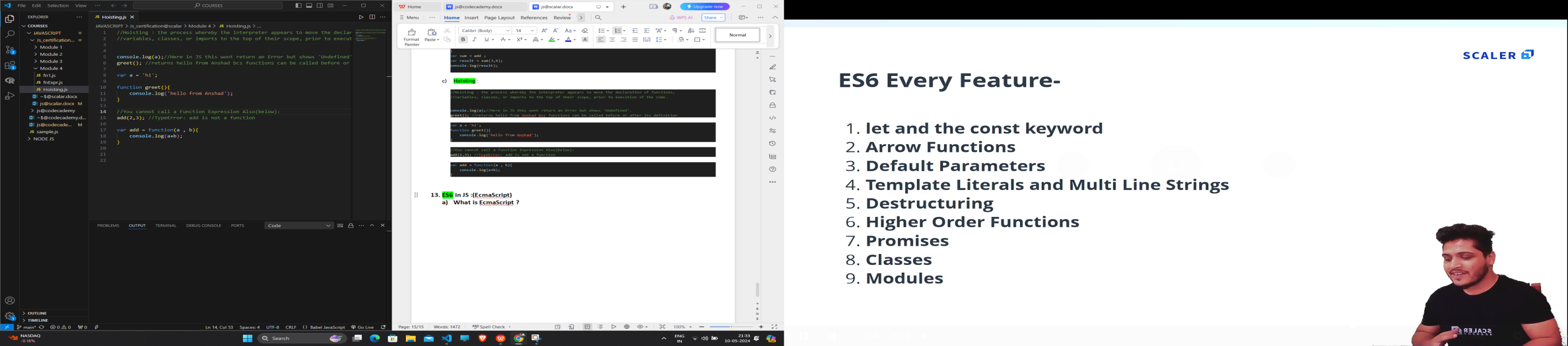
var add = function(a , b){

    console.log(a+b);

}

1. **ES6 in JS :(EcmaScript)**
   1. **What is EcmaScript ?**

****

****

* 1. **Problems with var :**

////variables declared with 'var' keyword are not blocked scoped they are function scoped.

//(that is we can access it from outside scope,which should not be allowed).

var a =35;

if(a == 35){

    var b  = 40;

    console.log(b);

}

console.log(b);//b is accesssible since variables declared using 'var' keyword are not blocked scoped.

//variables declared using var keyword are function scoped :

function test(){

    var c = 100;

    console.log(c);

}

test();

console.log(c); //ReferenceError: c is not defined

* 1. **Let and const :**

|  |  |  |
| --- | --- | --- |
| **var** | **let** | **const** |
| 1. **varibles declared using ‘var’ keyword are NOT BLOCKED SCOPED.function scoped.** | 1. **varibles declared using ‘let’ keyword are BLOCKED SCOPED.** | 1. **varibles declared using ‘const’ keyword are also BLOCKED SCOPED.** |
| 1. **Can be Re-Declared.** | 1. **Cannot be Re-Declared.** | 1. **Cannot be Re-Declared.** |
| 1. **Can be Re-Assigned.** | 1. **Can be Re-Assigned.** | 1. **Cannot be Re-Assigned.** |

//'var' has a problem that it allows us to redeclare a variable which is a PROBLEM ,which should not be Allowed.

var a =20;

var a = 35;

var b = 30;

console.log(a);

console.log(b);

//SOLUTION to "Redeclaration Problem" :  use 'let' keyword instead of 'var' :

let a =20;

let a = 35;//SyntaxError: Identifier 'a' has already been declared

let b = 30;

console.log(a);

console.log(b);

//Variable declared with 'let' keyword are Blocked SCoped.

if(true){

    let c = 40 ;

    c= 20 ; //But varible declared using 'let' keyword can be Re-Assigned.

    console.log(c);

}

console.log(c);//cannot access variable c .Since it is declared using 'let'.

//Const Keyword:(Blocked Scope,Cannot Re-Declare,Cannot Re-Assign):

const d = 100;

console.log(d);

const d = 120;//Cannot Re-Declare

d = 25 ;//Cannot Re-Assign

* 1. **What are Arrow Functions ?**

//Without Using Arrow Function :

let a = 2;

let test1 = function(){

    console.log(a+2);

}

let test2 = function(a){

    console.log(a\*2);

}

let test3 = function(a , b){

    console.log(a+b);

}

test1();

test2(4);

test3(2 , 6);

//With Using Arrow Function :

let b = 2;

let tests1 = ()=>{

    console.log(b+2);

}

//1.No need to use any parenthesis For function with one parameter.

let tests2 = b =>{

    console.log(b\*2);

}

//2.No Need Any curly braces if function is single-line function

let tests3 = (b , c) => console.log(b+c);

tests1();

tests2(4);

tests3(2 , 6);

* 1. **Template Literal and multi-line Strings (String Interpolation `` ${}):**

|  |  |
| --- | --- |
| **Code** | **Output** |
| console.log(' I am Anshad');  //If you want to make the above string to multiple lines USE BACKTICK:- [``]:  /\*  console.log(' I  am  Anshad');  \*/  console.log(`I  am  Anshad`)  //STRING INTERPOLATION using backtick `` and ${} :  let a = 24 ;  console.log('My age is '+ a);//Instead of concatenation like this .  console.log(`My age is ${a}`);//Use STRING INTERPOLATION ${} inside backtick `` . | I am Anshad  I  am  Anshad  My age is 24  My age is 24 |

* 1. **Destructuring Arrays :**

//The destructuring assignment syntax is a JavaScript expression that makes it possible to Unpack values.

//From Arrays,or properties from objects , into distinct variables.

let arr = ['Hi','I' , 'am' , 'Anshad'];

//Normal way of accessing Array elements :

let a = arr[1];

let b = arr[2];

console.log(a);

console.log(b);

//Instead we Have another Way - [DESCTRUCTING] to do this :

let [p , q ,r ,s] = arr ; //All the values of array will be assigned to respective variables.

console.log(e);

* 1. **Destructuring Objects :**

|  |  |
| --- | --- |
| **Code** | **Output** |
| //Destructing an Object :  let myObject = {      name : 'Adam',      age : 25 ,      gender : 'M'  }  let { a , b , c } = myObject;  console.log(a);//undefined  console.log(b);//undefined  console.log(c);//undefined  //We get output as undefined because we cant destructure an Object like Array.  //We have to the use the same key name used in object :  let { name , age , gender } = myObject;  console.log(name);  console.log(age);  console.log(gender);  //So Is Thare any way to give another name to this variables ?.  //YES :  let { name:n , age:ag , gender:g } = myObject;  console.log(n);  console.log(ag);  console.log(g);  //What if You have a Nested Object :  let myObject2 = {      id : 'D1',      course : 'MCA',      address : {          country : 'India',          state : 'kerala',      }  }  let {id:i , course:cr ,address : {country :cy , state : st}} = myObject2;  console.log(i);  console.log(cr);  console.log(cy);  console.log(st); | undefined  undefined  undefined  Adam  25  M  Adam  25  M  D1  MCA  India  kerala |

* 1. **Modules and Modularity :**
     1. **First create calc.js for calculator :**

//Specify the calculator functions

function add(a ,b ){

    console.log(a+b);

}

function sub(a ,b ){

    console.log(a-b);

}

function mul(a ,b ){

    console.log(a\*b);

}

function div(a ,b ){

    console.log(a/b);

}

//To export this functions to another file use 'module.exports':

module.exports = {

    addition : add ,

    subtraction : sub ,

    multiplication : mul ,

    division : div

}

* + 1. **Then import the function in main.js :**

//How to import those functions in calc ? Use : require()

const calculator = require('./calc');

calculator.addition(2 , 3);

calculator.subtraction(3 , 2);

calculator.multiplication(2 , 3);

calculator.division(6 , 3);

* + 1. **sd**
  1. **Asds :**