



[View, add and edit your notes in the app](#)

ES6 full course in Hindi | ECMA script 6 full tutorial

Generated on November 8, 2023

Summary

Notes

Screenshots

Bookmarks

30

126

0

What is ECMA Script

- European Computer Manufacturers Association (ECMA) Script.
- Provide Standard to javascript for different Browsers.
- Also Used with Node js.
- This is not framework or Library.

We use var for variables
var a=10
Now We can use let also
let a=10

Anil Sidhu

11:58 PM 12/18/2020

2:35

Edits deck

slides.com/anilsidhu/deck-626da0/edit

History & Version Script

- First appeared 1997; 23 years ago.
- First Editions in June 1997.
- ES6 (6th edition) in 2015 also called ECMA Script 2015.
- ES11 (11th edition) in June 2020.

Anil Sidhu

< >

Type here to search

11:59 PM 12/18/2020

▶ 3:52

Edits deck

slides.com/anilsidhu/deck-626da0/edit

What we will cover

- Let, const
- Arrow Function
- Class
- Promise
- Rest Parameters
- Spread Operator
- ** Operator
- Much More

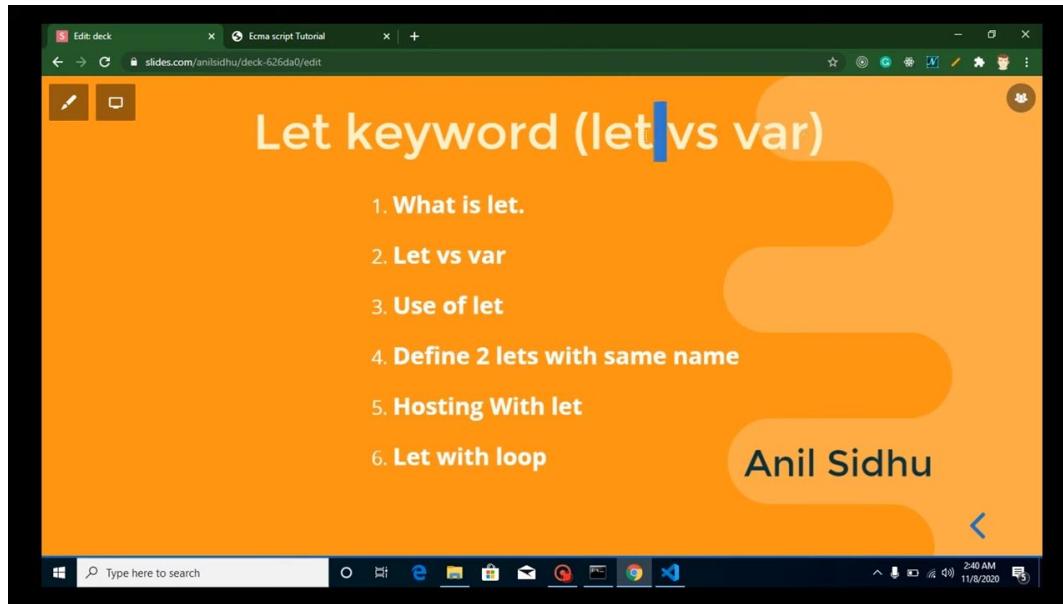
Anil Sidhu

< >

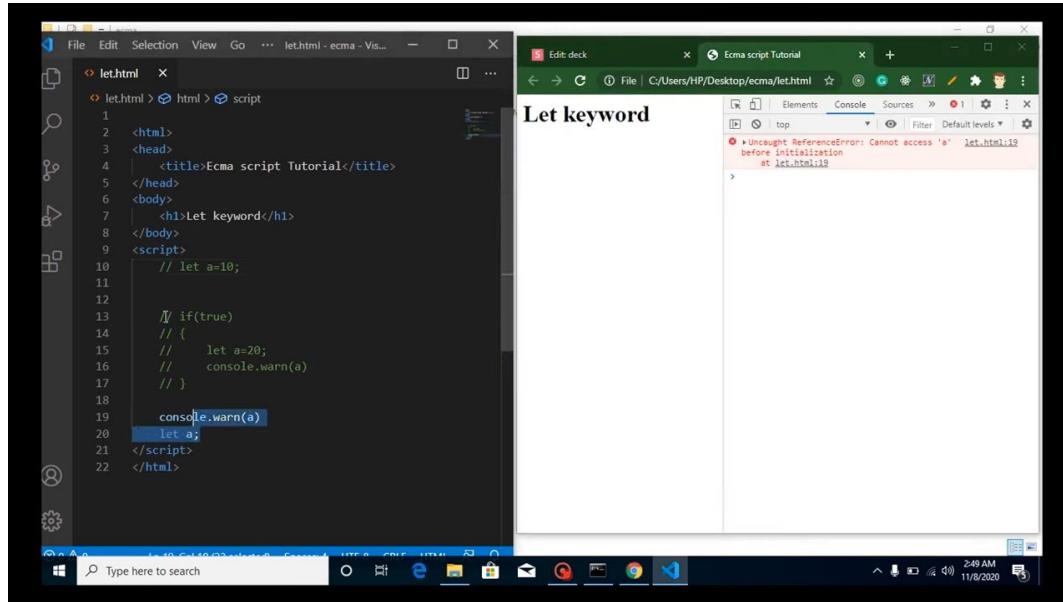
Type here to search

12:00 AM 12/19/2020

▶ 4:00



▶ 4:42



▶ 12:26

```
File Edit Selection View Go ... let.html - ecmascript - Vis... □ ...
```

```
let.html x
```

```
let.html > html > script
```

```
1 <html>
2 <head>
3 <title>Ecma script Tutorial</title>
4 <body>
5 <h1>Let keyword</h1>
6 </body>
7 <script>
8   for(var i=0;i<10;i++)
9   {
10     setTimeOut(()=>{
11       console.warn(i)
12     },1000)
13   }
14 </script>
15 </html>
```

```
File Edit deck x Ecma script Tutorial x +
```

```
Let keyword
```

```
File C:/Users/HP/Desktop/ecma/let.html
```

```
Elements Console Sources
```

```
top
```

```
let.html:13
```

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

▶ 13:44

```
File Edit Selection View Go ... let.html - ecmascript - Vis... □ ...
```

```
let.html x
```

```
let.html > html > script
```

```
1 <html>
2 <head>
3 <title>Ecma script Tutorial</title>
4 </head>
5 <body>
6 <h1>Let keyword</h1>
7 </body>
8 <script>
9   for(let i=0;i<10;i++)
10   {
11     setTimeOut(()=>{
12       console.warn(i)
13     },1000)
14   }
15 </script>
16 </html>
```

```
File Edit deck x Ecma script Tutorial x +
```

```
Let keyword
```

```
File C:/Users/HP/Desktop/ecma/let.html
```

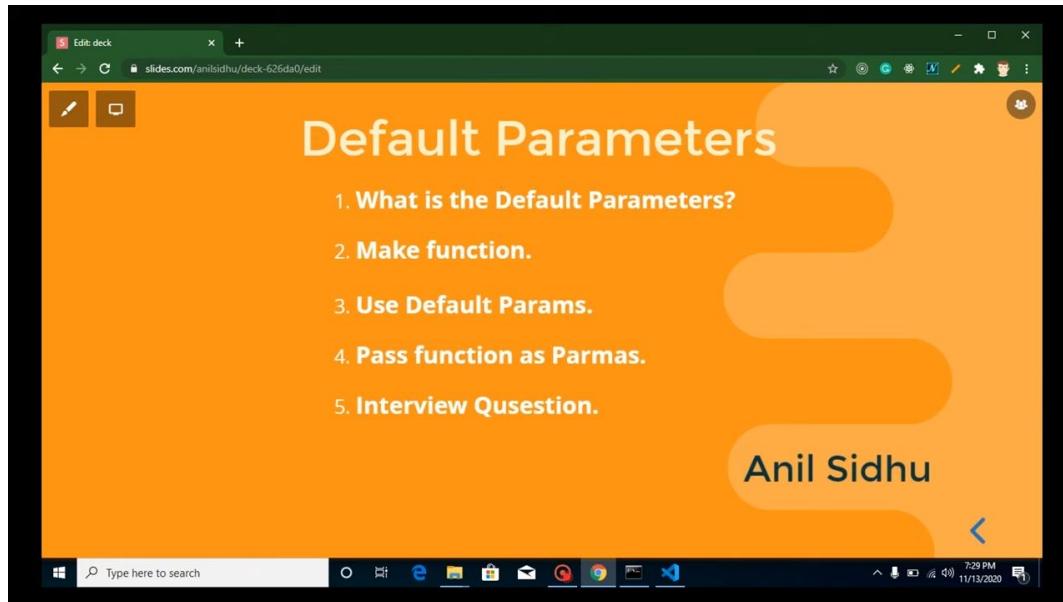
```
Elements Console Sources
```

```
top
```

```
let.html:13
```

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

▶ 14:46

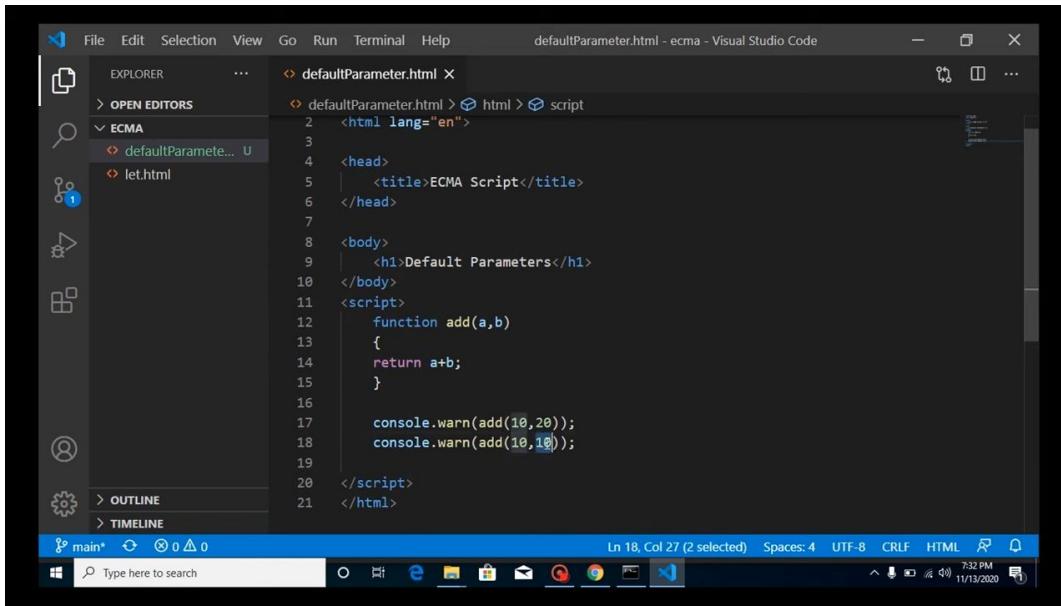


▶ 15:23

A screenshot of Visual Studio Code showing an HTML file named "defaultParameter.html". The code in the editor is:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>ECMA Script</title>
</head>
<body>
    <h1>Default Parameters</h1>
</body>
</html>
```

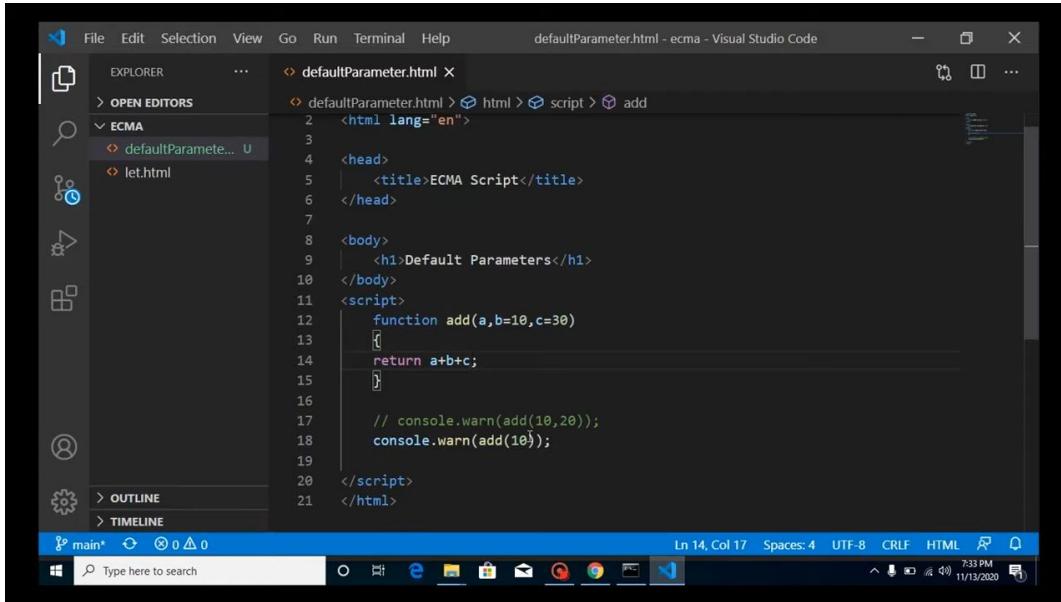
▶ 16:10



```
<html lang="en">
<head>
    <title>ECMA Script</title>
</head>
<body>
    <h1>Default Parameters</h1>
</body>
<script>
    function add(a,b)
    {
        return a+b;
    }

    console.warn(add(10,20));
    console.warn(add(10,10));
</script>
</html>
```

▶ 17:57



```
<html lang="en">
<head>
    <title>ECMA Script</title>
</head>
<body>
    <h1>Default Parameters</h1>
</body>
<script>
    function add(a,b=10,c=30)
    [
    ]
    return a+b+c;
    [
    ]

    // console.warn(add(10,20));
    console.warn(add(10));
</script>
</html>
```

▶ 18:49

File Edit Selection View Go Run Terminal Help defaultParameter.html - ecma - Visual Studio Code

EXPLORER OPEN EDITORS ECMAScript defaultParameter.html let.html

```
4 <head>
5 | <title>ECMA Script</title>
6 </head>
7
8 <body>
9 | <h1>Default Parameters</h1>
10 </body>
11 <script>
12   function someVal()
13   {
14     return 100;
15   }
16   function add(a = 10, b = someVal(), c)
17   {
18     (parameter) => number
19     return a + b + c;
20   }
21
22 // console.warn(add(10,20));
23 // console.warn(add(10));
24
```

Ln 23, Col 13 Spaces: 4 UTF-8 CRLF HTML

▶ 19:47

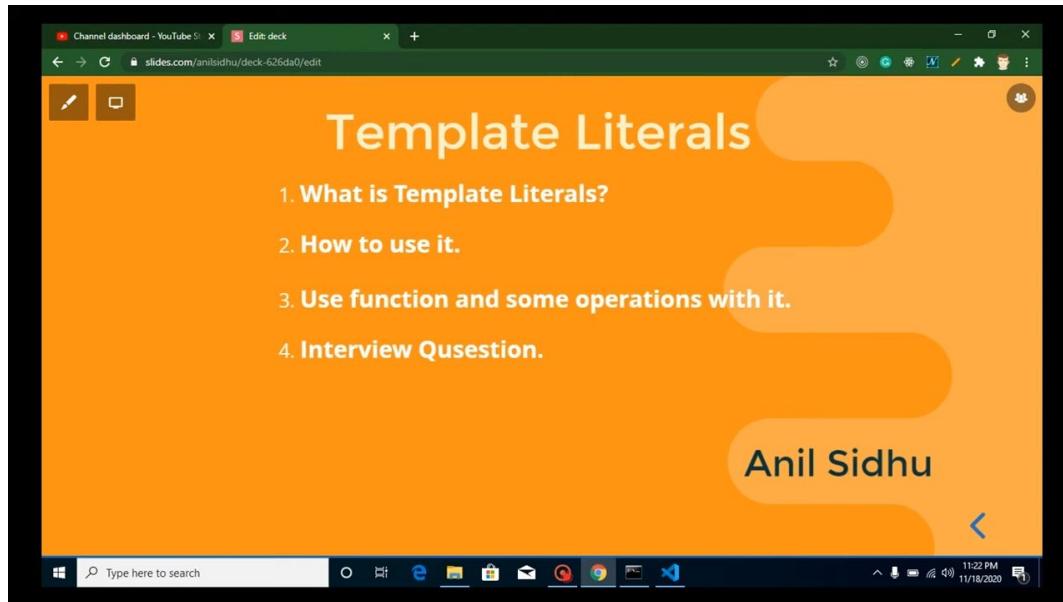
File Edit Selection View Go Run Terminal Help defaultParameter.html - ecma - Visual Studio Code

EXPLORER OPEN EDITORS ECMAScript defaultParameter.html let.html

```
10 </body>
11 <script>
12   // function someVal()
13   // {
14   //   return "last";
15   // }
16   // function add(a,b="middle",c=someVal())
17   // {
18   //   return a + " " + b + " " + c;
19   // }
20
21 // // console.warn(add(10,20));
22 // // console.warn(add("first"));
23 function main(a: any, b?: any)
24 function main(a, b = main())
25 {
26   return a + b;
27 }
28 main();
29 </script>
30 </html>
```

Ln 26, Col 6 Spaces: 4 UTF-8 CRLF HTML

▶ 21:56



▶ 22:14

```
<html>
<head>
<title>Ecma script Tutorial</title>
</head>
<body>
<h1>Template Literals</h1>
</body>
<script>
let quote="either you run the code or code runs you";
let by = 'Anil Sidhu';
let user="peter"
x=10;
y=20;
function fullName()
{
    return "Anil Kumar Sidhu";
}
console.warn("either you run the code or code runs" +
"you" + "Hello "+user)
console.warn(`either you run the code or code runs you
${by} Hello ${x*y}`);
</script>
</html>
```

▶ 29:32

A screenshot of a presentation slide titled "find and findIndex". The slide has a yellow background with white text. It lists six points: 1. Why we Use find?, 2. Why we use findIndex?, 3. Example with array, 4. Example with array of object, 5. Example with custom function, and 6. Interview Qusestion. Below the list is a signature "Anil Sidhu". The browser address bar shows "slides.com/anilsidhu/deck-626da0/edit". The system tray at the bottom right indicates the date as 11/23/2020 and the time as 11:24 PM.

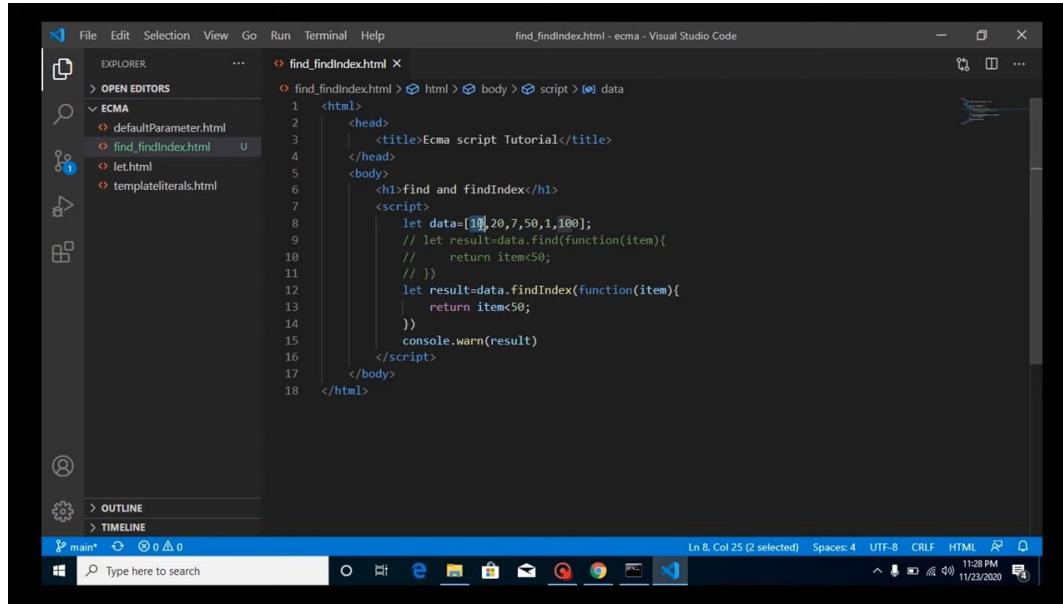
▶ 31:03

A screenshot of Visual Studio Code showing the file "find_findIndex.html". The code editor displays the following HTML and JavaScript:

```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>find and findIndex</h1>
    <script>
      let data=[10,20,7650,1,100];
      let result=data.findIndex(function(item){
        return item<10;
      })
      warn(...data: any[]);
      console.warn(result)
    </script>
  </body>
</html>
```

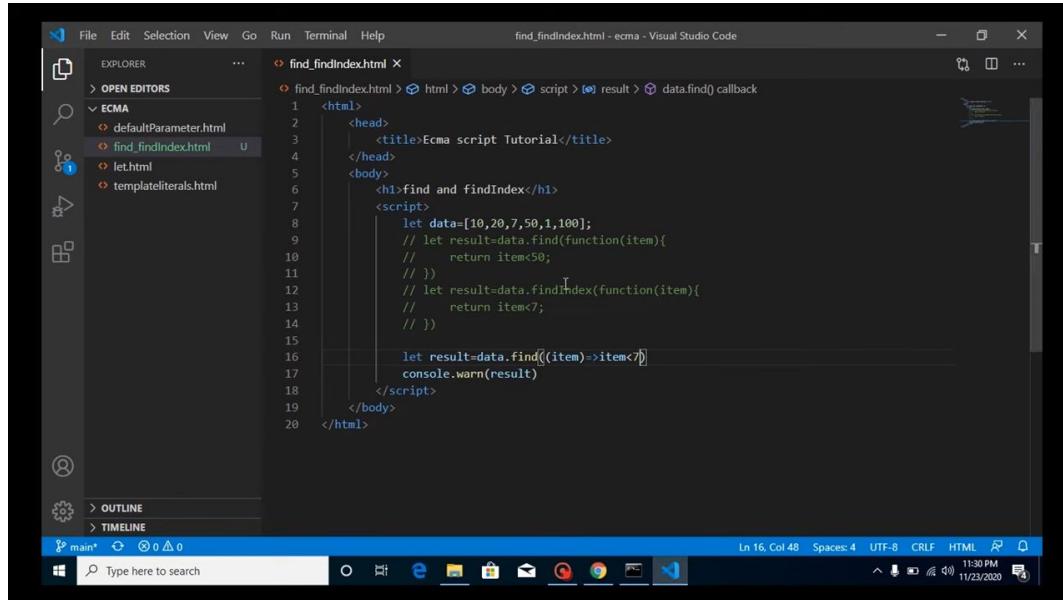
The Explorer sidebar shows other files like "defaultParameter.html", "let.html", and "templateiterals.html". The status bar at the bottom right shows the date as 11/23/2020 and the time as 11:28 PM.

▶ 34:53



```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>find and findIndex</h1>
    <script>
      let data=[10,20,7,50,1,100];
      // let result=data.find(function(item){
      //   return item<50;
      // })
      let result=data.findIndex(function(item){
        return item<50;
      })
      console.warn(result)
    </script>
  </body>
</html>
```

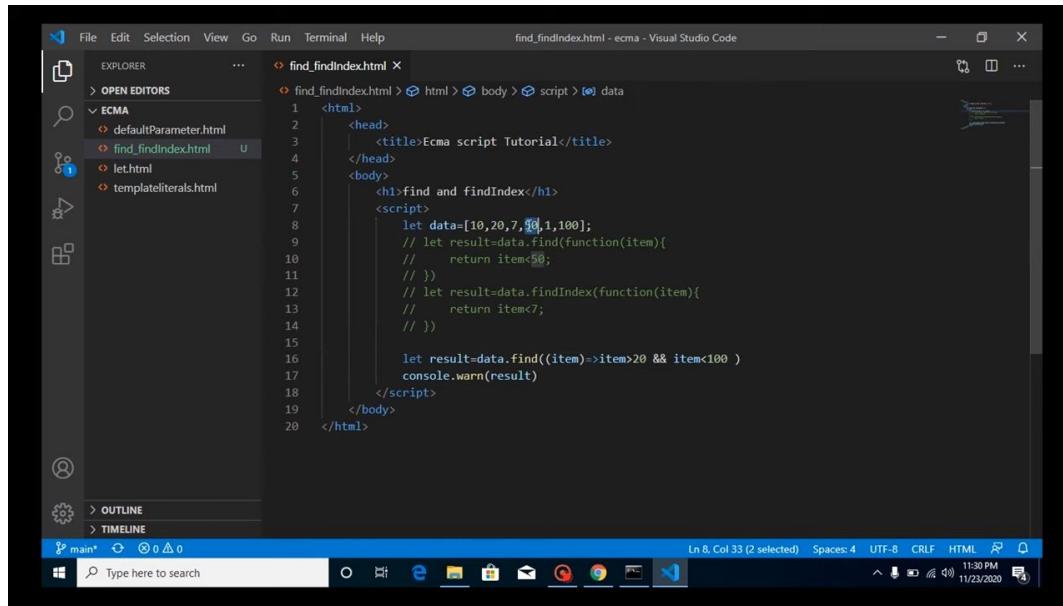
▶ 35:23



```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>find and findIndex</h1>
    <script>
      let data=[10,20,7,50,1,100];
      // let result=data.find(function(item){
      //   return item<50;
      // })
      // let result=data.findIndex(function(item){
      //   return item<70;
      // })

      let result=data.findIndex(item=>item<70)
      console.warn(result)
    </script>
  </body>
</html>
```

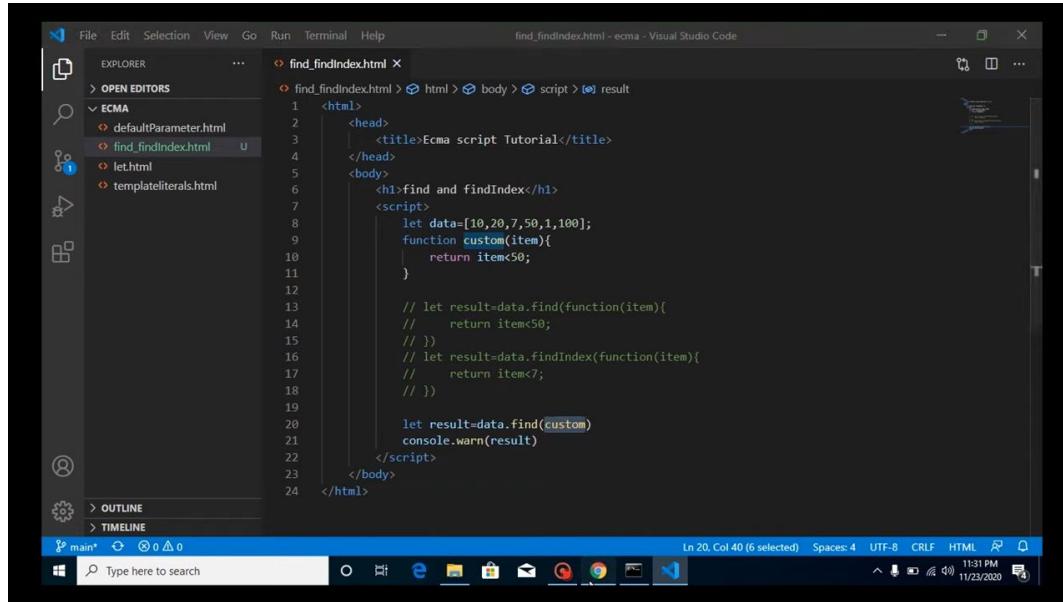
▶ 36:54



```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>find and findIndex</h1>
    <script>
      let data=[10,20,7,50,1,100];
      // let result=data.find(function(item){
      //   return item<50;
      // })
      // let result=data.findIndex(function(item){
      //   return item<7;
      // })

      let result=data.find((item)=>item>20 && item<100)
      console.warn(result)
    </script>
  </body>
</html>
```

▶ 37:48

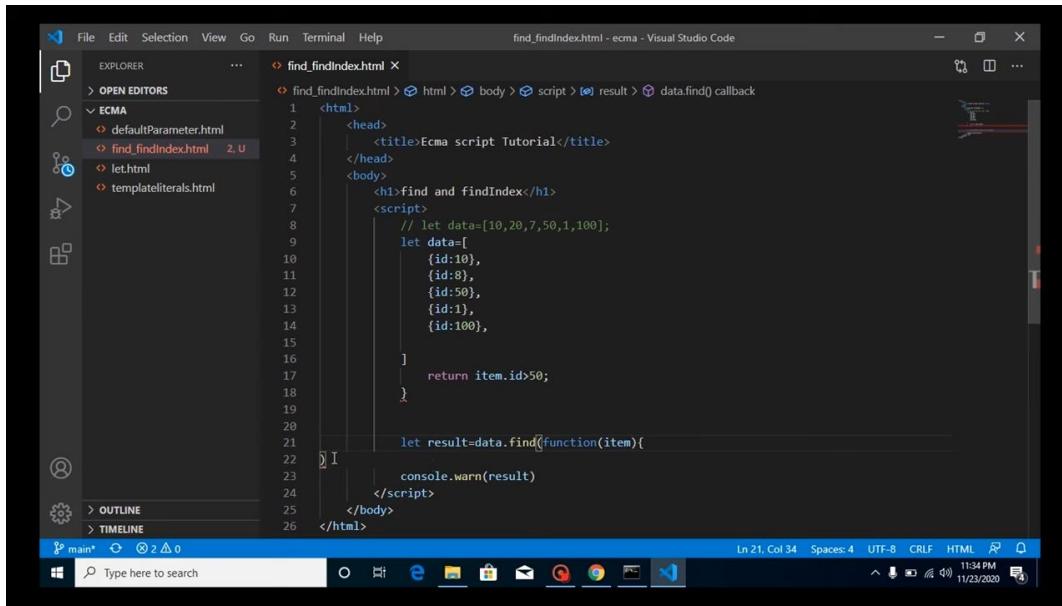


```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>find and findIndex</h1>
    <script>
      let data=[10,20,7,50,1,100];
      function custom(item){
        return item<50;
      }

      // let result=data.find(function(item){
      //   return item<50;
      // })
      // let result=data.findIndex(function(item){
      //   return item<7;
      // })

      let result=data.find(custom)
      console.warn(result)
    </script>
  </body>
</html>
```

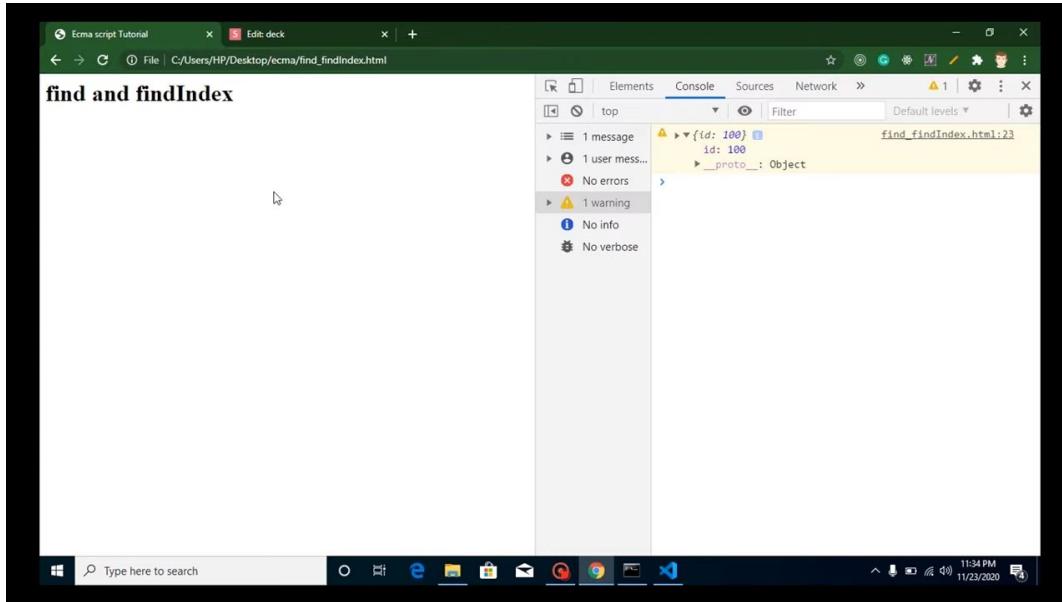
▶ 38:52



```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>find and findIndex</h1>
    <script>
      // let data=[10,20,7,50,1,100];
      let data=[
        {id:10},
        {id:8},
        {id:50},
        {id:1},
        {id:100},
      ]
      return item.id>50;
    </script>
  </body>
</html>
```

Ln 21, Col 34 Spaces: 4 UTF-8 CRLF HTML

▶ 40:59



Ecma script Tutorial

find and findIndex

Console

```
! 1 message
> !> {id: 100} 11:34 PM 11/23/2020
  id: 100
  > __proto__: Object
```

1 user mess...

No errors

1 warning

No info

No verbose

▶ 41:13

A screenshot of Visual Studio Code showing the file 'find_findIndex.html'. The code uses the 'find' method on an array to find items where the id is greater than 50. The array contains objects with id and name properties.

```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>find and findIndex</h1>
    <script>
      // let data=[10,20,7,50,1,100];
      let data=[
        {id:10,name:"a"}, 
        {id:8}, 
        {id:50}, 
        {id:1}, 
        {id:100}, 
      ]
      
      let result=data.find(function(item){
        return item.id>50;
      })
      console.warn(result)
    </script>
  </body>
</html>
```

▶ 41:23

A screenshot of a presentation slide titled 'Arrow Function in Detail'. The slide lists seven points: 1. How to use arrow function? 2. Why use arrow function? 3. Arrow function Example 4. Complex example 5. This in Arrow Function 6. Arrow vs normal function 7. Interview Questions. The slide is attributed to 'Anil Sidhu'.

- 1. How to use arrow function?
- 2. Why use arrow function?
- 3. Arrow function Example
- 4. Complex example
- 5. *This* in Arrow Function
- 6. Arrow vs normal function
- 7. Interview Questions.

Anil Sidhu

▶ 42:54

A screenshot of Visual Studio Code showing an HTML file named 'arrow.html'. The code defines an arrow function 'test3' that returns 1. The code editor interface is visible with various icons and a status bar at the bottom.

```
2 <head>
3   <title>Ecma script Tutorial</title>
4 </head>
5 <body>
6   <h1>Arrow function</h1>
7   <script>
8     let data = [10, 20, 30, 40, 50];
9
10    function test()
11    {
12      return 1;
13    }
14
15    let test2=function()
16    {
17      return 1
18    }
19
20    let test3=()=>{
21      return 1
22    }
23
24  </script>
25 </body>
26
27 </html>
```

▷ 44:53

we use the arrow function because they fundamentalistic and simple to use

▷ 45:36

A screenshot of Visual Studio Code showing the same 'arrow.html' file. This time, the code uses an arrow function within a map operation to double each item in the 'data' array. A warning message 'warn(...data: any[])': void is shown in the code editor.

```
2 <head>
3   <title>Ecma script Tutorial</title>
4 </head>
5 <body>
6   <h1>Arrow function</h1>
7   <script>
8     let data = [10, 20, 30, 40, 50];
9
10    let newData= data.map(function(item){
11      return item*2
12    })
13
14    warn(...data: any[])
15    console.warn(newData)
16
17  </script>
18 </body>
19 </html>
```

▷ 47:20

▶ 47:56

arrow function ka this hota he nhi hai wo parent ko lauy kar chalta hai

▶ 48:32

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure with files like `arrow.html`, `defaultParameter.html`, `find_index.html`, `let.html`, and `template_literals.html`. The `arrow.html` file is currently open.
- Code Editor (Center):** Displays the content of `arrow.html`:

```
2 <head>
3   <title>Ecma script Tutorial</title>
4 </head>
5 <body>
6   <h1>Arrow function</h1>
7   <script>
8     let data = [10, 20, 30, 40, 50];
9
10 // let newData= data.map(function(item){
11 //   return item*2
12 // })
13 let newData= data.map((item)=>item*2)
14
15 console.warn(newData)
16
17 class App extends cmmponent{
18   <button onClick=(<-->this.test())>Test</button>
19 }
20
21 </script>
22 </body>
```
- Bottom Status Bar:** Shows the file name (`arrow.html - ecma - Visual Studio Code`), line number (Ln 23), column number (Col 5), spaces (Spaces: 4), encoding (UTF-8), file type (CR/LF), and date/time (11:38 PM 11/24/2020).
- Bottom Taskbar:** Shows icons for File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar.

▶ 49:18

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Editor Bar:** arrow.html - ecmascript - Visual Studio Code.
- Explorer:** OPEN EDITORS (1 UNSAVED), ECMA (arrow.html, defaultParameter.html, find_findIndex.html, let.html, template_literals.html).
- Code:** Content of arrow.html:

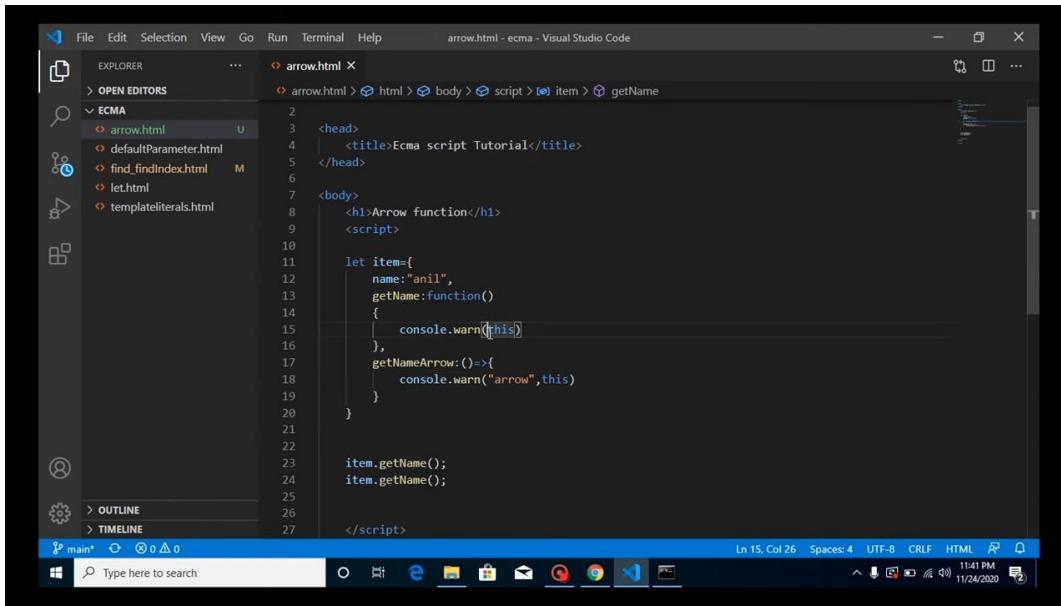
```
2 <head>
3   <title>Ecma script Tutorial</title>
4 </head>
5 <body>
6   <h1>Arrow function</h1>
7   <script>
8     let data = [10, 20, 30, 40, 50];
9
10 // let newData= data.map(function(item){
11 //   return item*2
12 // })
13 let newData= data.map((item)=>item*2)
14
15 console.warn(newData)
16
17 class App extends component{
18   <button onClick={()=>this.test()}>Test</button>
19   <button onClick={this.test.bind(this)}>Test</button>
20 }
21 </script>
```
- Bottom Status Bar:** In 21, Col 28, Spaces: 4, UTF-8, CRLF, HTML, 11:38 PM, 11/24/2020.
- Search Bar:** Type here to search.

▶ 49:34

The screenshot shows a browser developer tools window with the following details:

- Title Bar:** Ecma script Tutorial, Edit deck.
- Console Tab:** Elements, Sources, Default levels.
- Console Output:** 1 message, 1 user message, No errors, 1 warning.
 - Warning: arrow.html:15 {name: "anil", getName: f} arrow.html:15 {name: "anil", getName: f} name: "anil" __proto__: Object
- Bottom Status Bar:** 11:40 PM, 11/24/2020.
- Search Bar:** Type here to search.

▶ 51:36



```
File Edit Selection View Go Run Terminal Help arrow.html - ecma - Visual Studio Code

EXPLORER arrow.html ...
OPEN EDITORS ECMA
arrow.html defaultParameter.html find_findIndexhtml let.html templateiterals.html

<head>
  <title>Ecma script Tutorial</title>
</head>
<body>
  <h1>Arrow function</h1>
  <script>

    let item={
      name:"anil",
      getName:function()
      {
        console.warn(this)
      },
      getNameArrow:()=>{
        console.warn("arrow",this)
      }
    }

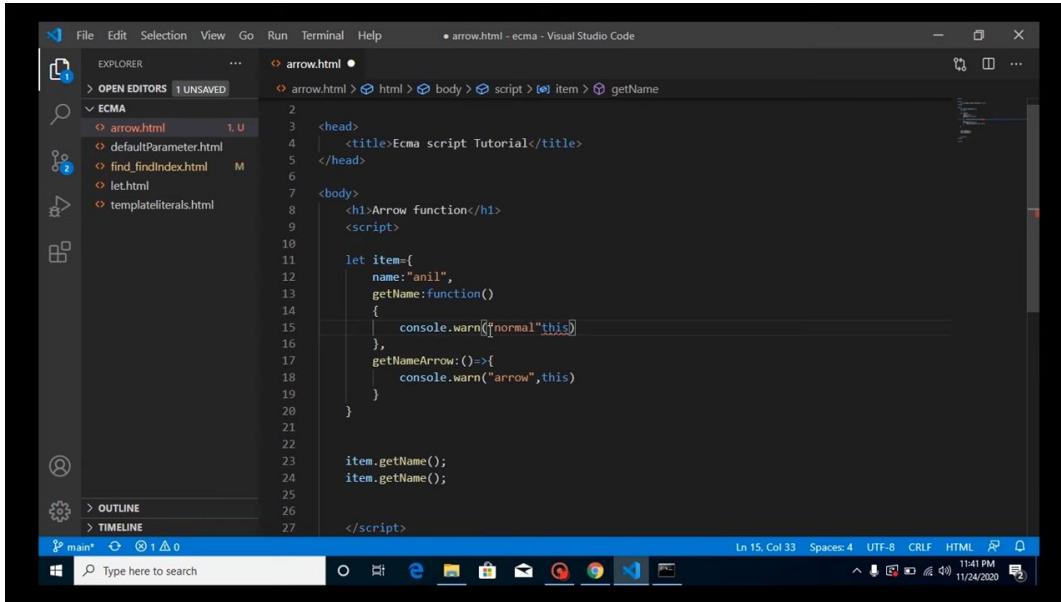
    item.getName();
    item.getName();

  </script>

```

main* 0 0 0 Type here to search 11:41 PM 11/24/2020

▶ 52:28



```
File Edit Selection View Go Run Terminal Help arrow.html - ecma - Visual Studio Code

EXPLORER arrow.html • 1 UNSAVED
OPEN EDITORS ECMA
arrow.html defaultParameter.html find_findIndexhtml let.html templateiterals.html

<head>
  <title>Ecma script Tutorial</title>
</head>
<body>
  <h1>Arrow function</h1>
  <script>

    let item={
      name:"anil",
      getName:function()
      {
        console.warn("normal",this)
      },
      getNameArrow:()=>{
        console.warn("arrow",this)
      }
    }

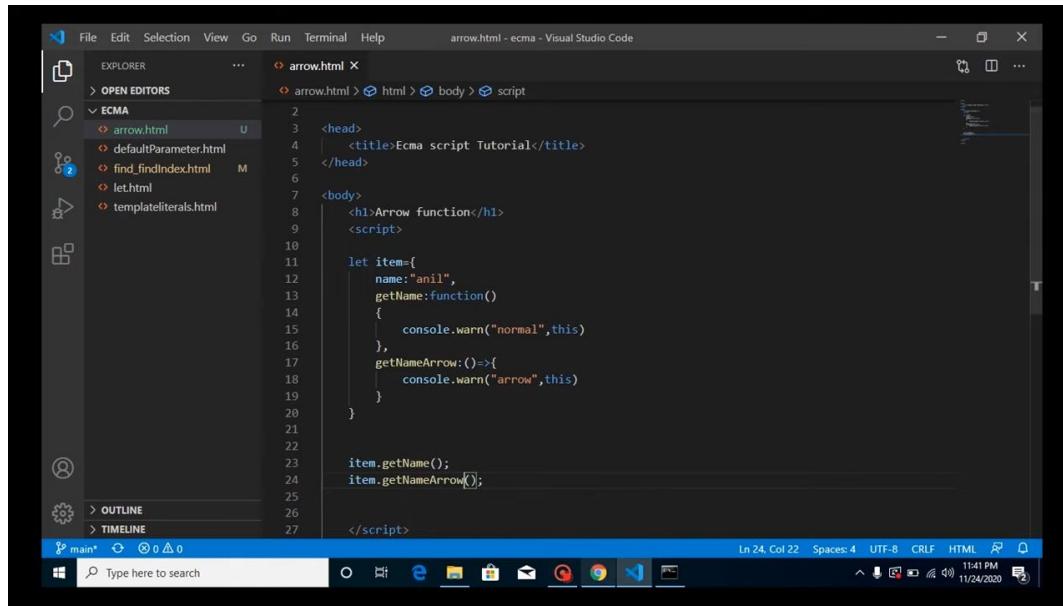
    item.getName();
    item.getName();

  </script>

```

main* 0 1 0 Type here to search 11:41 PM 11/24/2020

▶ 52:32

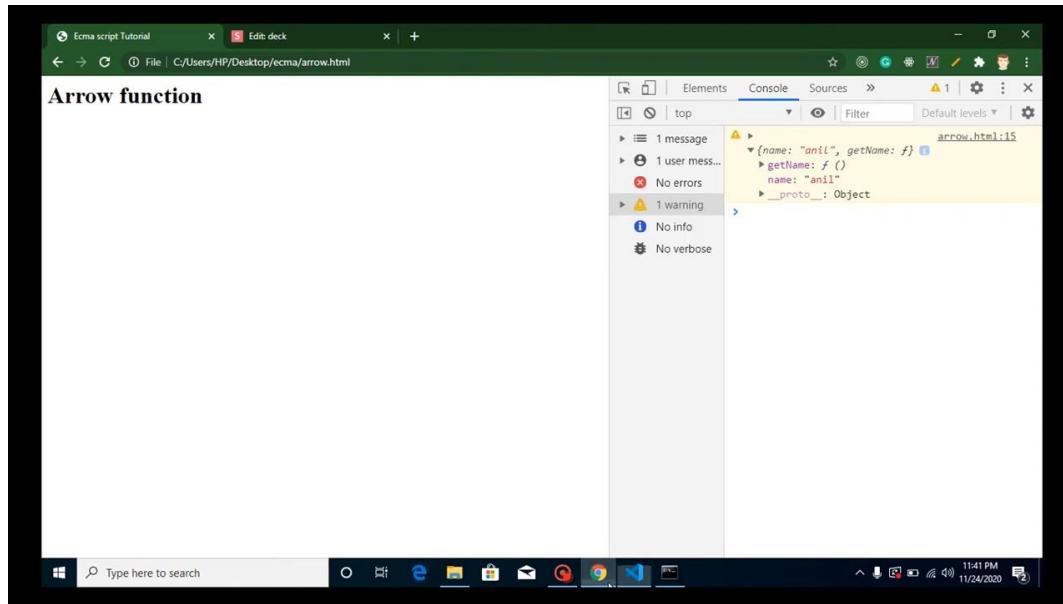


```
File Edit Selection View Go Run Terminal Help
arrow.html - ecma - Visual Studio Code

EXPLORER arrow.html ...
OPEN EDITORS ECMA
arrow.html defaultParameter.html find_findIndex.html let.html template_literals.html
> OUTLINE > TIMELINE
main* 0 0 0 Type here to search
In 24, Col 22 Spaces: 4 UTF-8 CRLF HTML
11:41 PM 11/24/2020
```

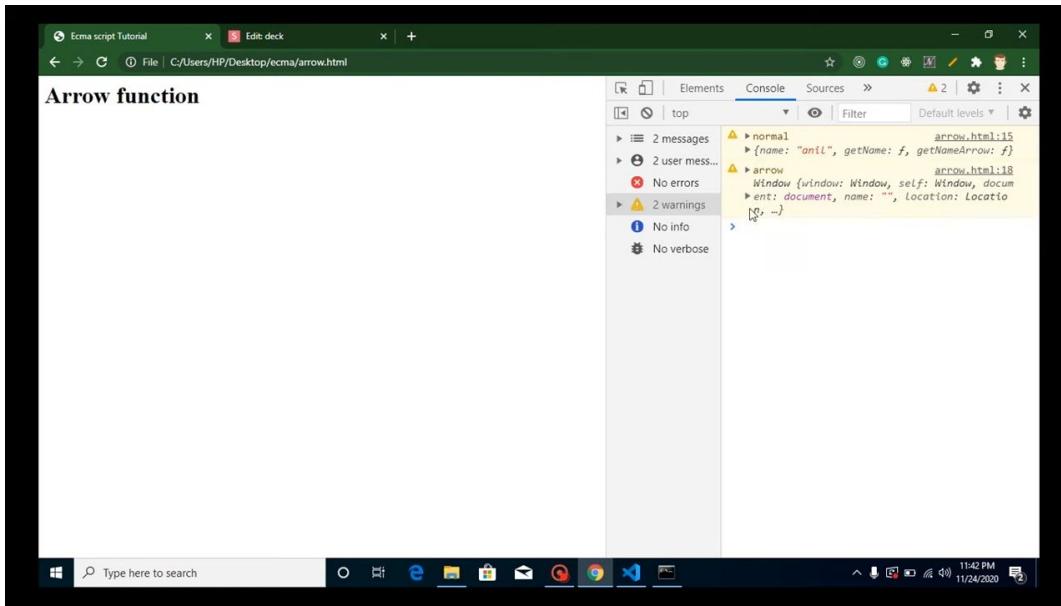
```
2 <head>
3   <title>Ecma script Tutorial</title>
4 </head>
5 <body>
6   <h1>Arrow function</h1>
7   <script>
8
9     let item={
10       name:"anil",
11       getName:function()
12       {
13         console.warn("normal",this)
14       },
15       getNameArrow:()=>{
16         console.warn("arrow",this)
17       }
18     }
19
20     item.getName();
21     item.getNameArrow();
22
23   </script>
24
25
26
27 </body>
28 </html>
```

▶ 52:38



```
Ecma script Tutorial x Edit deck x +
File | C:/Users/HP/Desktop/ecma/arrow.html
Elements Console Sources 1 | Default levels
top
1 message
1 user mess...
No errors
1 warning
No info
No verbose
arrow.html:15
{name: "anil", getName: f}
  > getName: f()
    name: "anil"
  > __proto__: Object
```

▶ 52:39



▶ 52:43

A screenshot of a presentation slide titled "Classes". The slide has a yellow background with a large orange swoosh graphic. The title "Classes" is centered at the top. Below it is a list of 5 points:

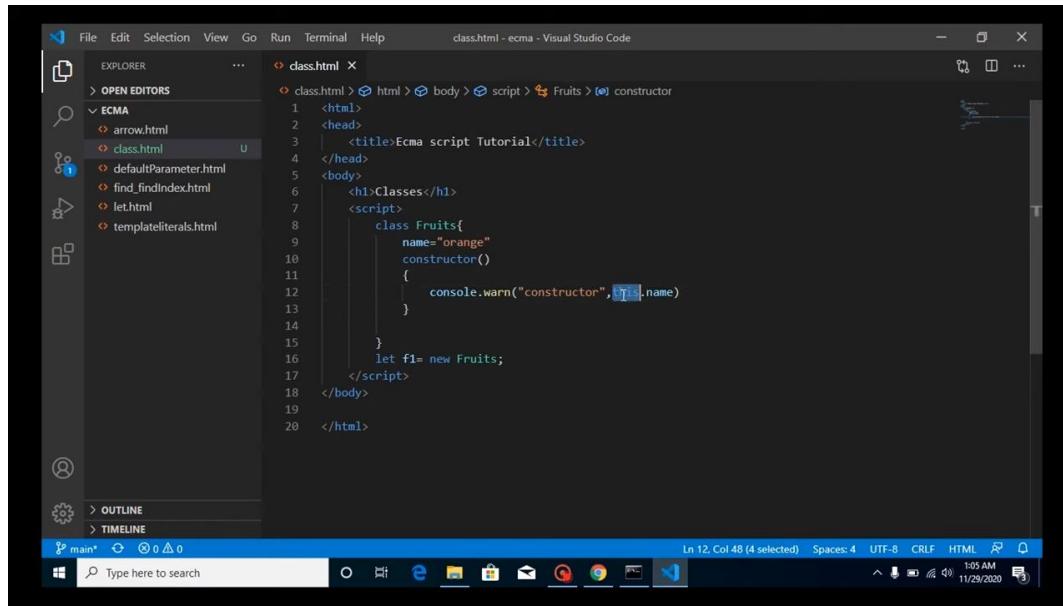
1. What is classes?
2. Make class and Call it?
3. Define constructor, Property and function?
4. Make class instance?
5. Interview Questions.

In the bottom right corner, there is a signature that reads "Anil Sidhu". The browser's taskbar at the bottom shows various pinned icons and the date/time as 11/29/2020 1:02 AM.

▶ 54:18

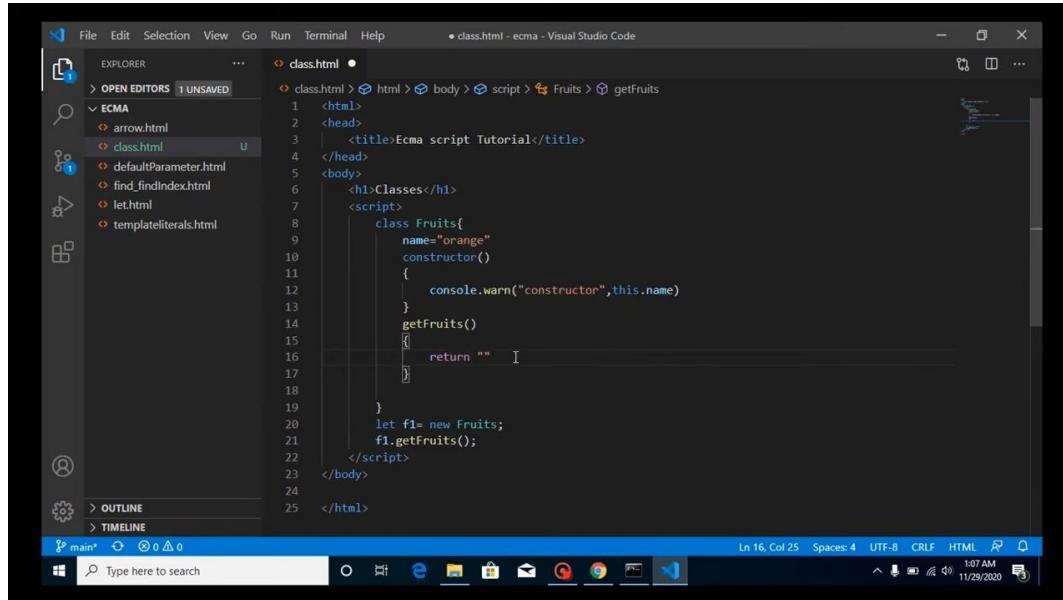
ya es6 main aey thi

▶ 54:29



```
class.html
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>Classes</h1>
    <script>
      class Fruits{
        name="orange"
        constructor()
        {
          console.warn("constructor",this.name)
        }
        let f1= new Fruits();
        getFruits()
        {
          return ""
        }
      }
    </script>
  </body>
</html>
```

▶ 58:10



```
class.html
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>Classes</h1>
    <script>
      class Fruits{
        name="orange"
        constructor()
        {
          console.warn("constructor",this.name)
        }
        getFruits()
        {
          return ""
        }
      }
      let f1= new Fruits();
      f1.getFruits();
    </script>
  </body>
</html>
```

▶ 59:24

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** class.html - ecma - Visual Studio Code.
- Explorer:** Shows the file structure under ECMA, including arrow.html, defaultParameter.html, find_findIndex.html, let.html, and template_literals.html. The class.html file is currently selected.
- Editor:** Displays the code for class.html:

```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>Classes</h1>
    <script>
      class Fruits{
        name="orange"
        constructor()
        {
          console.warn("constructor",this.name)
        }
        getFruits()
        {
          return "apple"
        }
      }
      let f1= new Fruits();
      console.warn(f1.getFruits())
    </script>
  </body>
</html>
```
- Bottom Status Bar:** Ln 21, Col 35, Spaces: 4, UTF-8, CRLF, HTML, 107 AM, 11/29/2020.
- Taskbar:** Shows icons for various applications like File Explorer, Task Manager, and File History.

▶ 59:43

The screenshot shows the Microsoft Edge DevTools interface with the following details:

- Title Bar:** Edit deck - Ecma script Tutorial.
- Console Tab:** Shows the following log output:

```
> constructor orange class.html:12
> apple class.html:21
```
- Left Sidebar:** Shows a tree view with categories like 2 messages, 2 user mess..., 2 warnings, etc.
- Bottom Status Bar:** 107 AM, 11/29/2020.
- Taskbar:** Shows icons for various applications like File Explorer, Task Manager, and File History.

▶ 59:59

Inheritance

1. What is Inheritance?
2. Make parent and child class?
3. Inheritance parent class?
4. Complete Example ?
5. Interview Questions.

Anil Sidhu

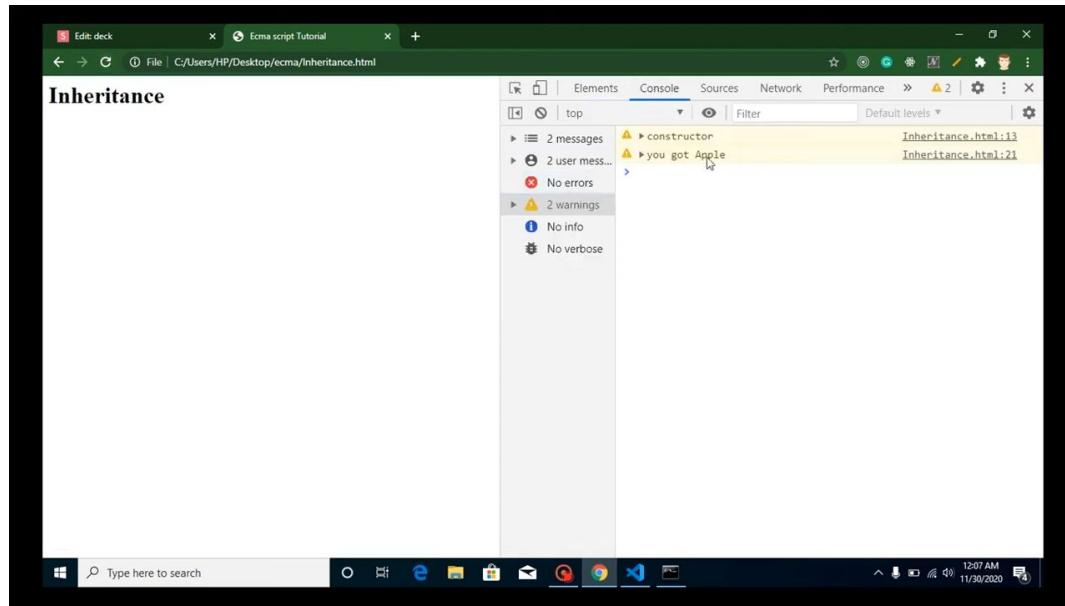
▶ 1:01:43

prents ki properties child main ana like eyes father k jasi hair
mother k jasa

▶ 1:02:50

```
File Edit Selection View Go Run Terminal Help • Inheritance.html - ecmascript - Visual Studio Code
SOUR... E ✓ ... Inheritance.html ●
Changes Inheritance.html > html > body > script
Message (Ctrl+Enter to co...
1 <html>
2   <head>
3     <title>Ecma script Tutorial</title>
4   </head>
5 
6   <body>
7     <h1>Inheritance</h1>
8     <script>
9       class Fruits{
10         constructor()
11         {
12           console.warn("constructor")
13         }
14         getFruit()
15         {
16           return "you got Apple";
17         }
18       }
19       let f1= new Fruits;      getFruit(): string
20       console.warn(f1.getFruit())
21     </script>
22   </body>
23 
24 </html>
In 21, Col 34  Spaces: 4  UTF-8  CRLF  HTML  ⚡
main*  Type here to search  0  12:07 AM  11/30/2020
```

▶ 1:04:44

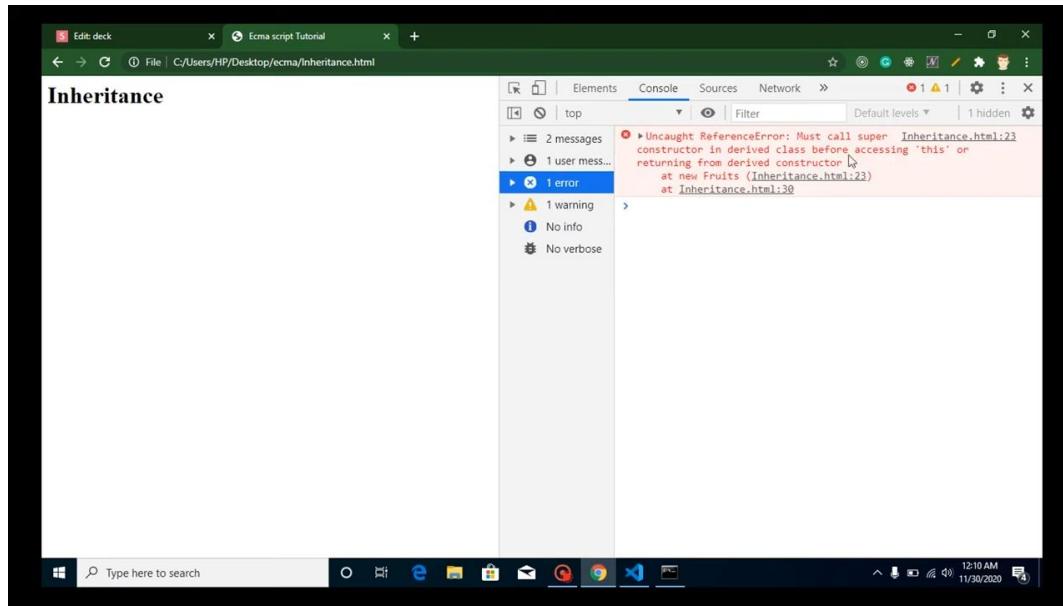


▶ 1:04:55

A screenshot of the Visual Studio Code editor. The file being edited is `Inheritance.html`. The code defines two classes: `Category` and `Fruits`. The `Category` class has methods `dryFruit()` and `pomeFruit()`. The `pomeFruit()` method returns "apple is pome fruit". The `Fruits` class has a constructor that logs "constructor" to the console and a `getFruit()` method. The code is annotated with comments explaining the inheritance structure.

```
<html>
<head>
<title>Ecma script Tutorial</title>
</head>
<body>
<h1>Inheritance</h1>
<script>
class Category{
    dryFruit()
    {
        return "Almond is dry fruit"
    }
    pomeFruit()
    [
        return "apple is pome fruit"
    ]
}
class Fruits{
    constructor()
    {
        console.warn("constructor")
    }
    getFruit()
    {
    }
}
```

▶ 1:05:51



▶ 1:07:40

A screenshot of Visual Studio Code showing the file Inheritance.html. The code defines two classes: Category and Fruits. Category has methods dryFruit() and pomeFruit(). Fruits extends Category and overrides pomeFruit() to return "apple is pome fruit". It also has a constructor that logs "constructor" and a getFruit() method that returns "you got Apple".

```
<html>
<head>
<title>Inheritance</title>
</head>
<body>
<script>
class Category{
    dryFruit()
    {
        return "Almond is dry fruit"
    }
    pomeFruit()
    {
        return "apple is pome fruit"
    }
}
class Fruits extends Category{
    constructor()
    {
        console.warn("constructor")
    }
    getFruit()
    {
        return "you got Apple";
    }
}
let f1= new Fruits;
let c1= new Category();
console.warn(f1.pomeFruit())
console.warn(c1.dryFruit())
</script>
</body>
</html>
```

▶ 1:07:52

```
<h1>Inheritance</h1>
<script>
    class Category{
        dryFruit()
        {
            return "Almond is dry fruit"
        }
        pomeFruit()
        {
            return "apple is pome fruit"
        }
    }
    class Fruits extends Category{
        constructor()
        {
            super();
            console.warn("constructor")
        }
        getFruit()
        {
            return "you got Apple";
        }
    }
    let f1= new Fruits;
    let c1= new Category;
    console.warn(f1.pomeFruit())

```

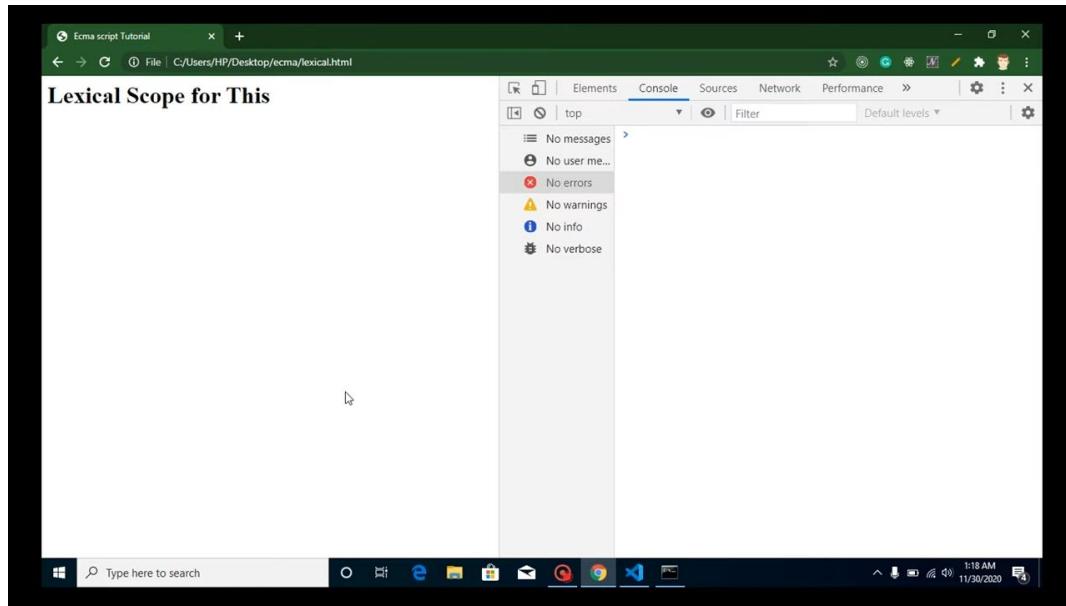
▶ 1:08:13

super key words parent ki calss k key word ko call kar ny k leya
use kar tay hai

▶ 1:08:55

```
Inheritance
Uncaught TypeError: c1.getFruit is not a function
at Inheritance.html:33
```

▶ 1:09:31



▶ 1:09:56

kisi chez ki kitni range hai us ko lexical scope boltay hai

▶ 1:10:21

object k under us he object ki dosri property use kar nay ho tu
this use karty hai

▶ 1:10:47

lexical scope ki problem arrow function say resolve hoti hai

▶ 1:11:44

The screenshot shows the Visual Studio Code interface with the following details:

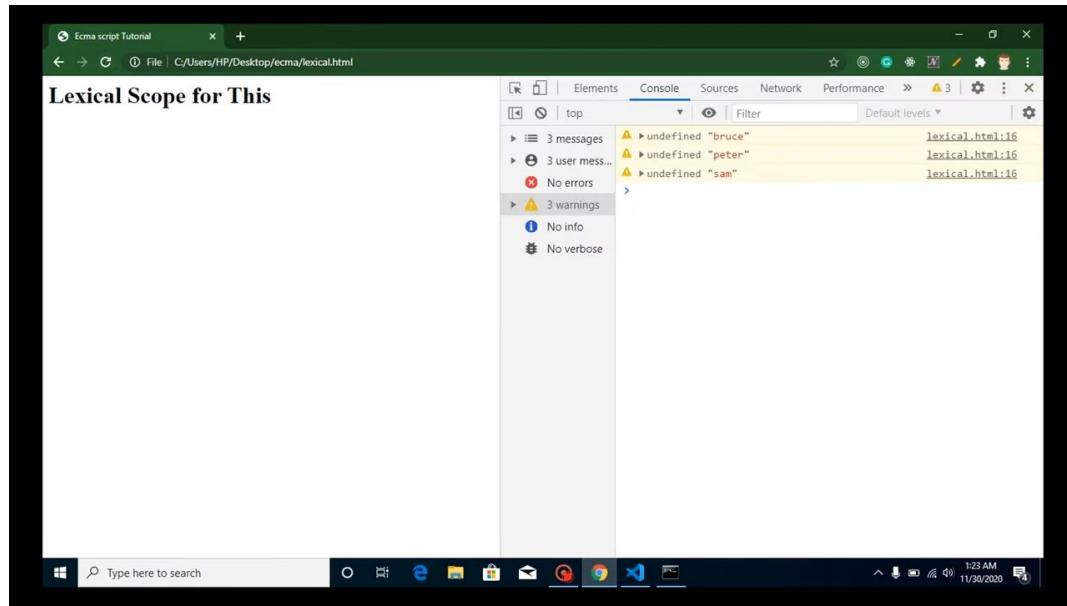
- File Explorer:** Shows a tree view of files under the 'ECMA' folder, including 'arrow.html', 'class.html', 'defaultParameter.html', 'find_index.html', 'Inheritance.html', 'let.html', 'lexical.html' (which is selected), and 'template_literals.html'.
- Code Editor:** Displays the content of 'lexical.html'. The code includes an HTML structure with a title and a script block. Inside the script, a variable 'data' is defined with properties 'list' and 'names', and a method 'getFriends'. A call to 'console.warn' is made with the argument 'this.list'.
- Status Bar:** Shows the line number (Ln 15), column (Col 34), spaces (Spaces: 4), encoding (UTF-8), and file type (CRLF HTML).

▶ 1:13:24

The screenshot shows a browser developer tools window with the following details:

- Title Bar:** Shows the tab name 'Ecma script Tutorial' and the file path 'C:/Users/HP/Desktop/ecma/lexical.html'.
- Console Tab:** Active tab, showing the following output:
 - 1 message
 - 1 user mess...
 - No errors
 - 1 warning
 - No info
 - No verbose
- Status Bar:** Shows the time '11:21 AM' and date '11/30/2020'.

▶ 1:13:29



▶ 1:14:55

```
<html>
<head>
<title>Ecma script Tutorial</title>
</head>
<body>
<h1>Lexical Scope for This</h1>
<script>
let data={
  list:'friend',
  names:['bruce','peter','sam'],
  getFriends:function()
  {
    this.names.map((item)=>{
      console.warn(this.list,item)
    })
  }
}
data.getFriends()
</script>
</body>
</html>
```

▶ 1:16:28

Rest Operator

1. What is Rest Operator?
2. Make a function?
3. Use Rest operator?
4. Complete Example?
5. Interview Questions.

Anil Sidhu

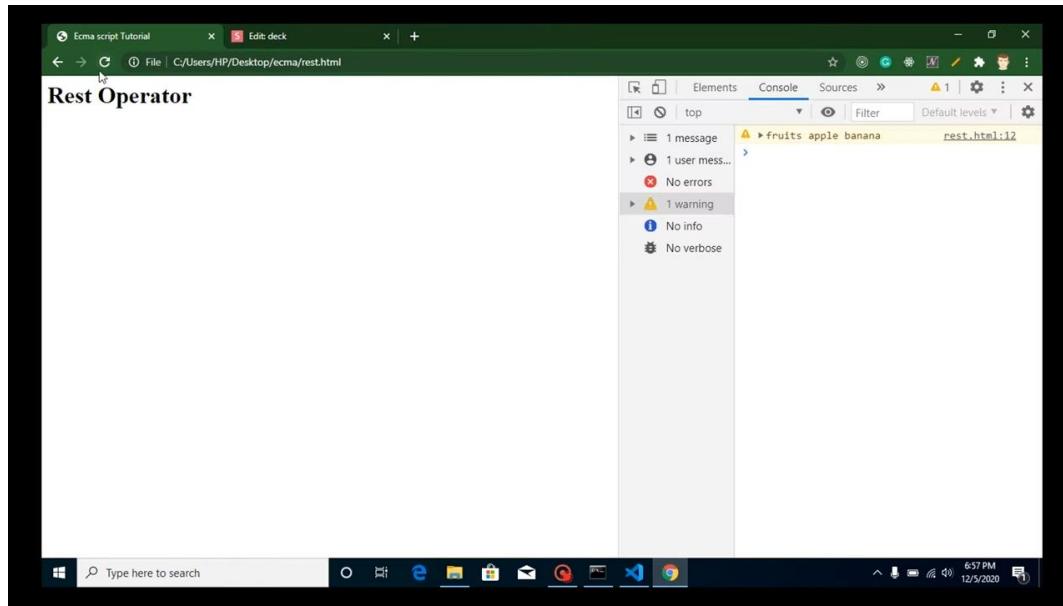
▶ 1:17:00

baqi ki bachi ho chezy rest operator hai

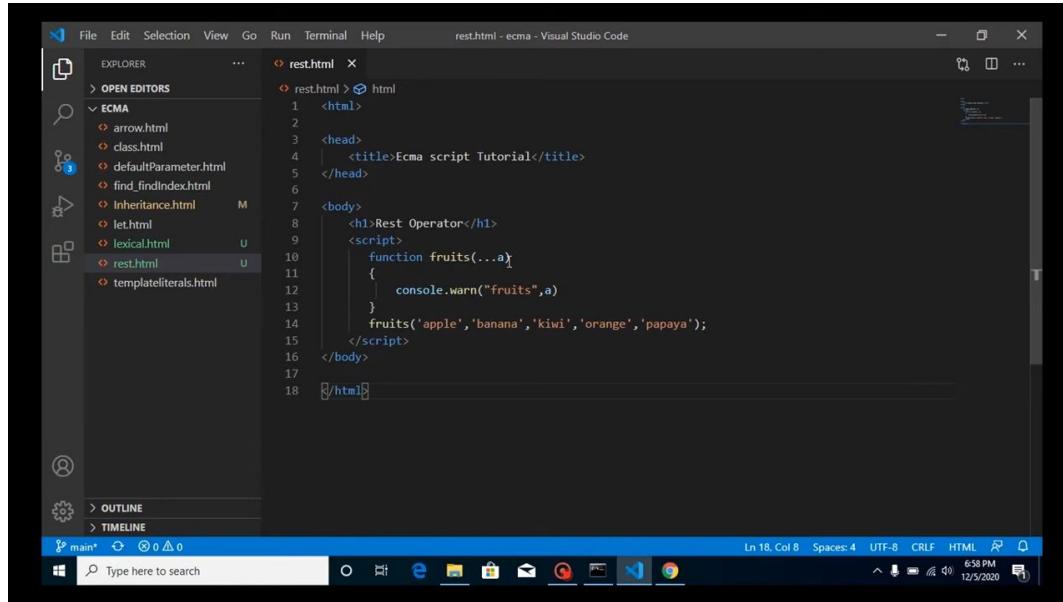
▶ 1:17:37

```
<html>
<head>
<title>Ecma script Tutorial</title>
</head>
<body>
<h1>Rest Operator</h1>
<script>
function fruits(a,b)
{
    console.warn("fruits",a,b)
}
fruits('apple','banana','kiwi','orange','paya');
</script>
</body>
</html>
```

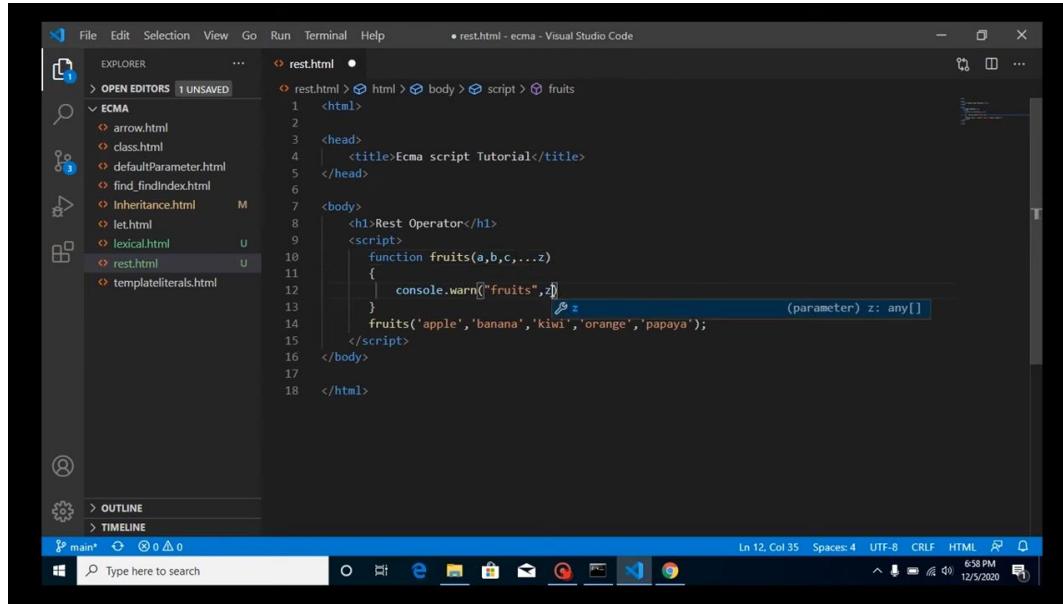
▶ 1:19:48



▶ 1:20:03

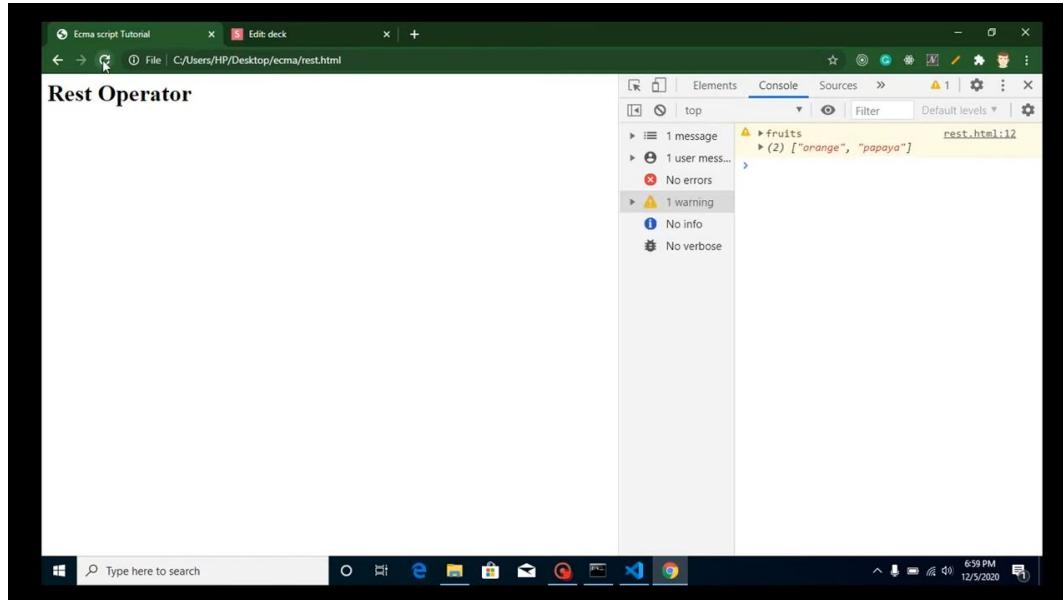


▶ 1:20:56



```
1 <html>
2 <head>
3   <title>Ecma script Tutorial</title>
4 </head>
5 <body>
6   <h1>Rest Operator</h1>
7   <script>
8     function fruits(a,b,c,...z)
9     {
10       console.warn(`fruits`, z)
11     }
12     fruits('apple','banana','kiwi','orange','papaya');
13   </script>
14 </body>
15 </html>
```

▶ 1:21:17



```
fruits
(2) ["orange", "papaya"]
```

▶ 1:21:22

The screenshot shows the Visual Studio Code interface with the file 'rest.html' open. The code defines a function 'fruits' that logs its arguments to the console. A 'test()' function is also defined, which calls 'fruits' with several arguments. The code is as follows:

```
1 <html>
2 <head>
3   <title>Ecma script Tutorial</title>
4 </head>
5 <body>
6   <h1>Rest Operator</h1>
7   <script>
8     function fruits(a,...z)
9     {
10       console.warn("fruits",z)
11     }
12     function test()
13     {
14     }
15     fruits('apple','banana','kiwi','orange','papaya');
16   </script>
17 </body>
18 </html>
```

▶ 1:23:17

The screenshot shows the Microsoft Edge DevTools interface with the 'Console' tab selected. The output window displays the following message:

```
fruits
(5) ["banana", "kiwi", "orange", "papaya"]
  0: "banana"
  1: "kiwi"
  2: "orange"
  3: "papaya"
  4: f: test()
length: 5
__proto__: Array(0)
```

▶ 1:23:23

The screenshot shows a presentation slide titled "Promise in ES6". The slide has a yellow background with white text. It lists four main points: 1. Why use Promise, 2. Understand Promise need with example?, 3. Complete Example., and 4. Interview Questions. Below the list, the name "Anil Sidhu" is displayed in a white box. The slide is titled "Promise in ES6".

▶ 1:23:39

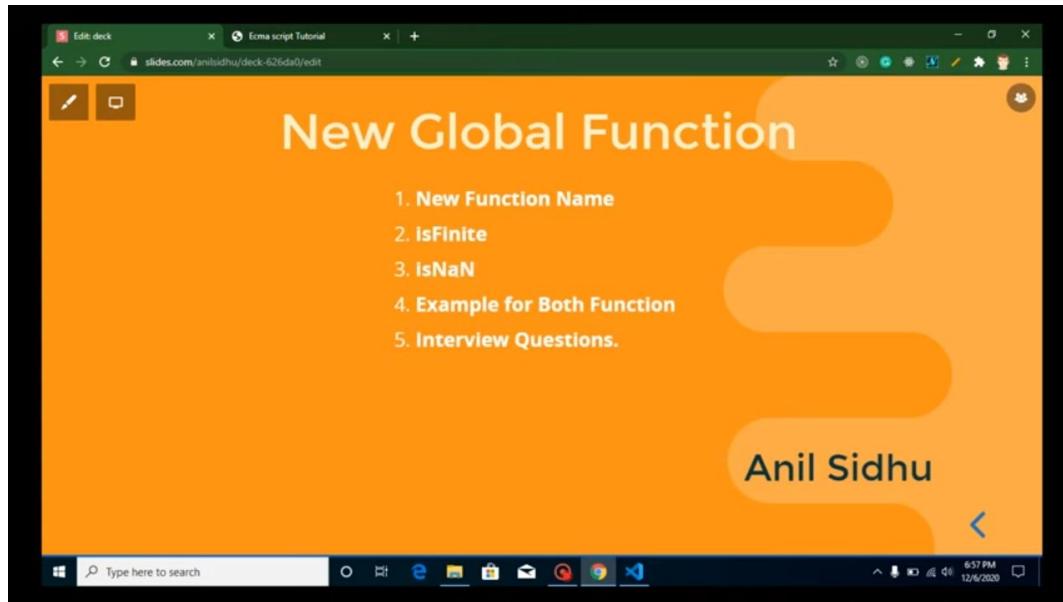
promise mian jab ek plock of code exciute ho jay tab he out putt milta hai us main with the passaeg of time

▶ 1:27:10

The screenshot shows a Visual Studio Code interface with a dark theme. The left sidebar shows an "EXPLORER" view with several files listed under "OPEN EDITORS" and "ECMA". The "promise.html" file is currently open in the editor. The code in the file is as follows:

```
1 <html>
2
3   <head>
4     <title>Ecma script Tutorial</title>
5   </head>
6
7   <body>
8     <h1>Rest Operator</h1>
9     <script>
10       let a=undefined;
11
12
13       let promiseData= new Promise((resolved,reject)=>{
14         setTimeout(() => {
15           a="Hello ECMA"
16           resolved("done")
17         }, 3000);
18       })
19
20       promiseData.then(()=>{
21         alert(a);
22       })
23
24     </script>
25
26
```

▶ 1:29:16



▶ 1:31:51

The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar shows a file tree with several files under the "ECMA" folder, including "arrow.html", "class.html", "defaultParameter.html", "find_index.html", "global.html", "Inheritance.html", "let.html", "lexical.html", "promise.html", "rest.html", and "template_literals.html". The "global.html" file is currently open in the editor. The code in the editor is:

```
1 <html>
2   <head>
3     <title>Ecma script Tutorial</title>
4   </head>
5
6   <body>
7     <h1>Global Function</h1>
8     <script>
9       |   console.warn(isFinite(10/0))
10      | </script>
11   </body>
12 </html>
```

The status bar at the bottom indicates the file is "global.html - ecma - Visual Studio Code". The bottom taskbar shows the date/time as 6:59 PM, 12/6/2020.

▶ 1:33:17

The screenshot shows a presentation slide titled "Destructuring Array". The slide has a yellow background with a large orange swoosh graphic on the right. The title is at the top, followed by a list of five points: 1. What Is Destructuring, 2. Example of Destructuring, 3. Destructuring with rest operator, 4. Default Value and skip value, and 5. Interview Questions. Below the list is a signature bubble containing "Anil Sidhu". At the bottom left is a search bar and taskbar, and at the bottom right is a system tray showing the date and time.

▶ 1:36:05

array ko unpack kar na variables k under destructurin hota hai

▶ 1:36:45

The screenshot shows a Visual Studio Code interface. The Explorer sidebar on the left lists several files under the "ECMA" folder, including "destructuring.html" which is currently open. The main editor window displays the following HTML code:

```
<html>
<head>
    <title>Ecma script Tutorial</title>
</head>
<body>
    <h1>Destructuring Array</h1>
    <script>
        | const [count, setCount]=useState()
    </script>
</body>
</html>
```

The status bar at the bottom shows the file path "destructuring.html - ecma - Visual Studio Code", line 10, column 30, and other details like encoding and file type.

▶ 1:37:22

ya destructing of array hai

▶ 1:37:31

The screenshot shows the Visual Studio Code interface with the file `destructuring.html` open. The code demonstrates array destructuring:

```
1 <html>
2   <head>
3     <title>Ecma script Tutorial</title>
4   </head>
5
6   <body>
7     <h1>Destructuring Array</h1>
8     <script>
9       let fruits=['apple','mango','kiwi'];
10      let [fruit1,fruit2,fruit3]=fruits;
11      console.warn(fruit1)
12      console.warn(fruit2)
13      console.warn(fruit3)
14
15
16    </script>
17  </body>
18 </html>
```

▶ 1:40:17

The screenshot shows the browser developer tools' Console tab. The output window displays the results of the `console.warn` statements from the code above:

```
apple
mango
kiwi
```

▶ 1:40:22

The screenshot shows the Visual Studio Code interface with the file 'destructuring.html' open. The code in the editor is:

```
1 <html>
2   <head>
3     <title>Ecma script Tutorial</title>
4   </head>
5
6   <body>
7     <h1>Destructuring Array</h1>
8     <script>
9       let fruits=['apple','mango','kiwi'];
10      let [fruit1,,fruit3]=fruits;
11      console.warn(fruit1)
12      // console.warn(fruit2)
13      console.warn(fruit3)
14
15
16    </script>
17  </body>
18
19 </html>
```

▶ 1:40:51

The screenshot shows a browser developer tools window with the 'Console' tab selected. The output in the console is:

```
apple
kiwi
```

▶ 1:40:56

array main elemenets ho orth nekalo tu

▶ 1:41:27

The screenshot shows the Visual Studio Code interface with the file 'destructuring.html' open. The code editor displays the following HTML and JavaScript:

```
<html>
<head>
    <title>Ecma script Tutorial</title>
</head>
<body>
    <h1>Destructuring Array</h1>
    <script>
        let fruits=['apple','mango','kiwi'];
        let [fruit1,,fruit3,fruit4]=fruits;
        // console.warn(fruit1)
        // console.warn(fruit2)
        // console.warn(fruit3)
        console.warn(fruit4)
    </script>
</body>
</html>
```

The 'EXPLORER' sidebar on the left lists various files under the 'ECMA' category, including 'arrow.html', 'class.html', 'defaultParameter.html', 'destructuring.html', 'find_index.html', 'global.html', 'Inheritance.html', 'let.html', 'lexical.html', 'promise.html', 'rest.html', and 'template_literals.html'. The 'OUTLINE' and 'TIMELINE' tabs are also visible.

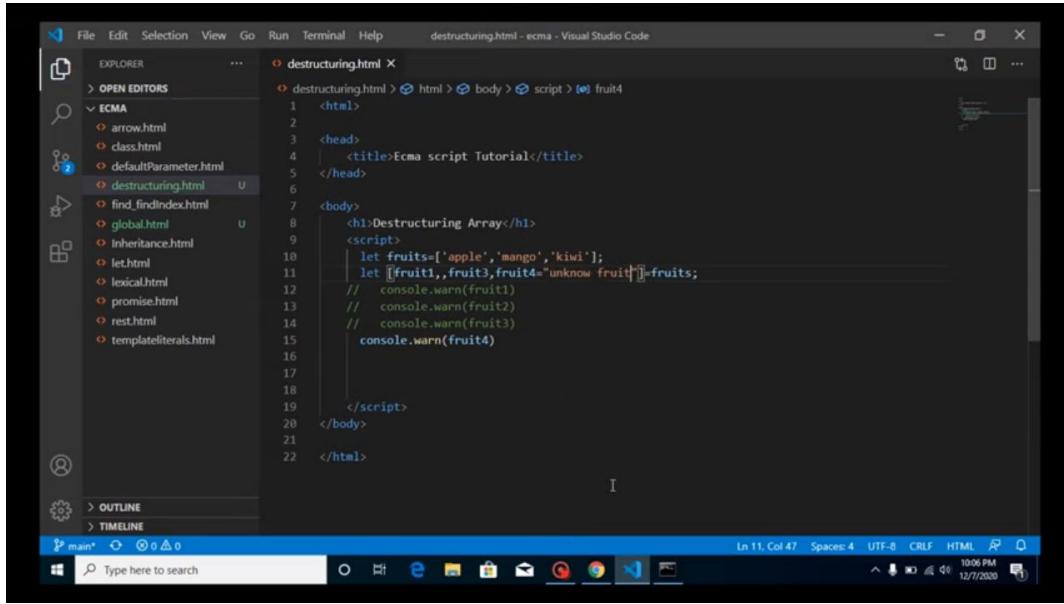
▶ 1:41:46

The screenshot shows the Microsoft Edge DevTools interface with the 'Console' tab selected. The console output shows the following message:

```
destructuring.html:15 undefined
```

The 'Elements' tab is also visible in the top navigation bar. The status bar at the bottom indicates the time as 10:06 PM and the date as 12/7/2020.

▶ 1:41:55

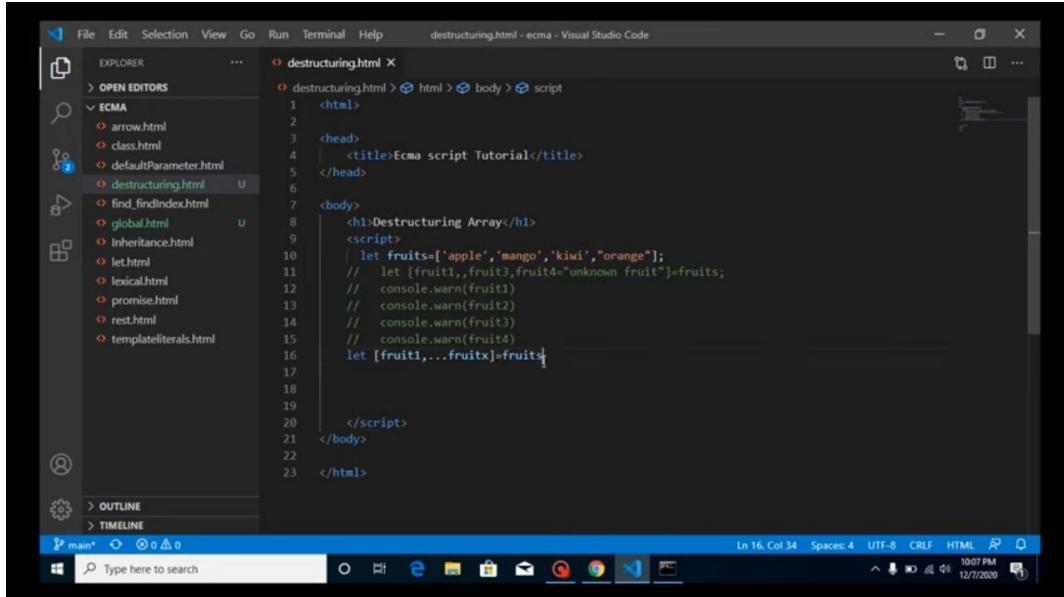


```
1 <html>
2   <head>
3     <title>Ecma script Tutorial</title>
4   </head>
5
6   <body>
7     <h1>Destructuring Array</h1>
8     <script>
9       let fruits=['apple','mango','kiwi'];
10      let [fruit1,,fruit3,fruit4='unknown fruit']=fruits;
11      // console.warn(fruit1)
12      // console.warn(fruit2)
13      // console.warn(fruit3)
14      console.warn(fruit4)
15
16
17
18
19     </script>
20   </body>
21
22 </html>
```

▶ 1:42:18

default value hai ya : per

▶ 1:42:59



```
1 <html>
2   <head>
3     <title>Ecma script Tutorial</title>
4   </head>
5
6   <body>
7     <h1>Destructuring Array</h1>
8     <script>
9       let fruits=['apple','mango','kiwi','orange'];
10      let [fruit1,fruit3,fruit4='unknown fruit']=fruits;
11      // console.warn(fruit1)
12      // console.warn(fruit2)
13      // console.warn(fruit3)
14      // console.warn(fruit4)
15      let [fruit1,...fruitx]=fruits;
16
17
18
19
20     </script>
21   </body>
22
23 </html>
```

▶ 1:43:42

jab pata na ho array main kitnay items hai t rest operator use
ary gay

▶ 1:43:54

The screenshot shows a presentation slide titled "Destructuring Object". The slide has a yellow background with orange wavy shapes. At the top, there is a navigation bar with icons for edit, back, forward, and search. Below the title, there is a list of five points:

1. What Is Destructuring
2. Example of Destructuring
3. Destructuring with rest operator
4. Default Value
5. Interview Questions.

In the center of the slide, the name "Anil Sidhu" is displayed in a white box. The slide is presented in a browser window titled "Edit deck" with the URL "slides.com/anilSidhu/deck-626d50/edit". The browser interface includes a search bar, a toolbar with various icons, and a status bar at the bottom showing the time as 11:33 PM and the date as 12/9/2020.

▶ 1:45:04

array main destruturing hoti hai index ka hessab say or object
main destruturing hoti hai keys k hessab say

▶ 1:45:29

ek object ko diffrent diffrent variabes main assign kar dena ya
break kar dena destruturing of object o ta ha

▶ 1:45:48

difference between object and array destruturing array k under
index say jab k object k under keys say values ko destrutured
kar ta hai

▶ 1:45:58

The screenshot shows the Visual Studio Code interface. The left sidebar has a tree view under 'OPEN EDITORS' with several files listed under 'ECMA'. The main editor area displays the code for 'destructuring_object.html'. The code includes an HTML structure with a title and a script section that logs 'anil@test.com' to the console.

```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>Destructuring Object</h1>
    <script>
      let user={name:'anil sidhu',email:'anil@test.com',mobile:9999}
      let {email}=user
      console.warn(email)
    </script>
  </body>
</html>
```

▶ 1:48:06

The screenshot shows a browser developer tools window with the 'Console' tab selected. It displays a single log message: 'anil@test.com' from the file 'destructuring_object.html' at line 12. The browser's address bar shows the file path 'C:/Users/Hp/Desktop/ecma/destructuring_object.html'.

Console output:

```
anil@test.com
```

▶ 1:48:20

A screenshot of Visual Studio Code showing an open file named `destructuring_object.html`. The code demonstrates object destructuring:

```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>Destructuring Object</h1>
    <script>
      let user={name:'anil sidhu',email:'anil@test.com',mobile:9999}
      let {email,name}=user
      console.warn(email)
      console.warn(name)
    </script>
  </body>
</html>
```

The code defines an object `user` with properties `name`, `email`, and `mobile`. It then destructures `user` into `email` and `name` using a brace operator, and logs them to the console.

▶ 1:48:54

A screenshot of a browser's developer tools console tab, showing the output of the code run in the previous screenshot. The console shows one warning message:

```
anil@test.com:12
```

The warning message is: `Warning: console.warn is deprecated. Please use console.log instead.`

▶ 1:48:57

A screenshot of Visual Studio Code showing an editor window for 'destructuring_object.html'. The code demonstrates object destructuring:

```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>Destructuring Object</h1>
    <script>
      let user={name:'anil sidhu',email:'anil@test.com',mobile:9999}
      let [..name]=user
      // console.warn(email)
      // console.warn(name)
      console.warn(mobile)
    </script>
  </body>
</html>
```

The code defines an object 'user' with properties 'name', 'email', and 'mobile'. It then destructures 'user' into an array where the first element is 'name'. It also includes comments and a 'console.warn' call for 'email' and 'name', and a final 'console.warn' for 'mobile'.

▶ 1:50:26

A screenshot of a browser's developer tools showing the 'Console' tab. The output shows the result of running the previous code:

```
destructuring_object.html:14
> {name: "anil sidhu", email: "anil@test.com", mobile: 9999}
```

The browser's status bar at the bottom indicates the time is 11:39 PM on 12/9/2020.

▶ 1:50:51

rest operator object k case main bachi hoay he items otha kar output detta hai

▶ 1:51:43

Spread Operator

1. What Is Spread Operator
2. Send Params in Function
3. Copy Array
4. Merge Array
5. Break Reference
6. Interview Questions.

Anil Sidhu

▶ 1:52:02

kaffe had tak rest operator ki terah he dekh ta hai ya tu exsiti

▶ 1:52:12

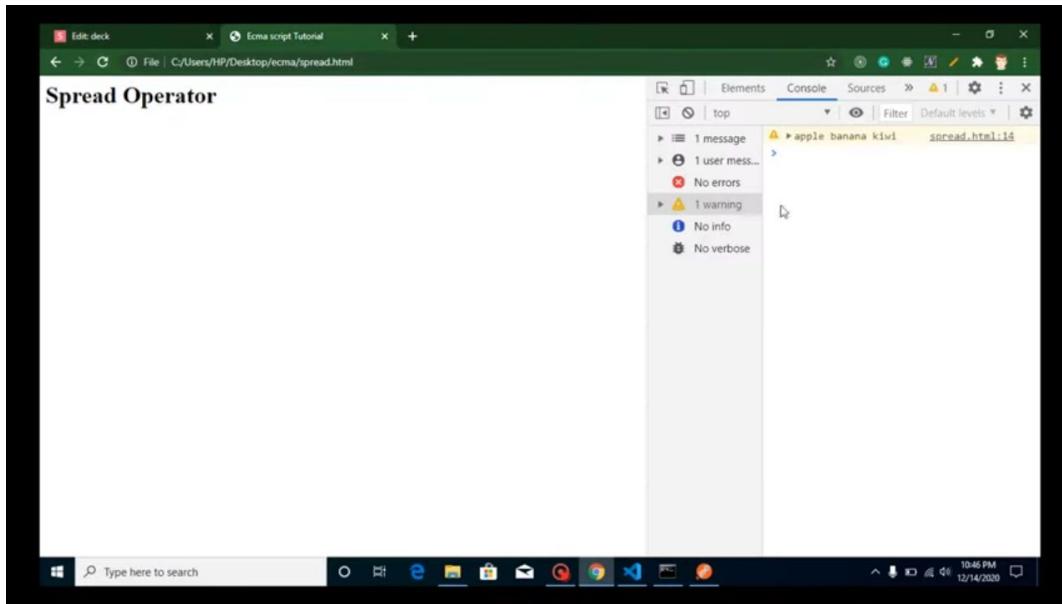
ek array ko dosry array k bech main ghosana ho ya parameter
main dalna ho tu wo existing parameter ya array main ko speal
kar k apni jaga bana de ga

▶ 1:52:25

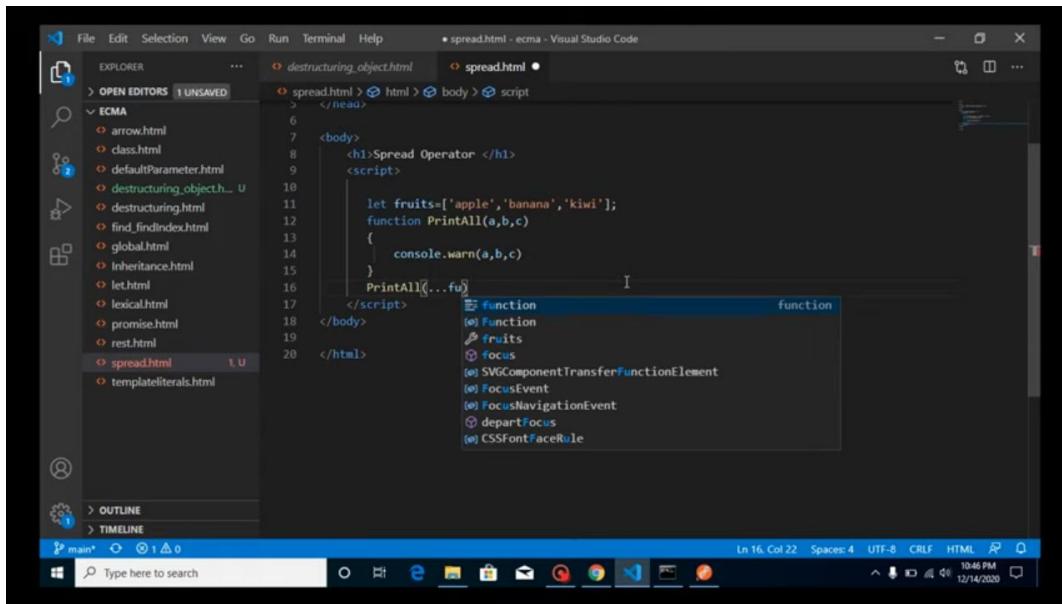
```
File Edit Selection View Go Run Terminal Help • spread.html - ecma - Visual Studio Code
EXPLORER OPEN EDITORS 1 UNSAVED
ECMA
arrow.html
class.html
defaultParameter.html
destructuring_object.html
destructuring.html
find_index.html
global.html
Inheritance.html
let.html
lexical.html
promise.html
rest.html
spread.html + U
template_literals.html

<head>
</head>
<body>
<h1>Spread Operator </h1>
<script>
let fruits=['apple','banana','kiwi'];
function PrintAll(a,b,c)
{
    console.warn(a,b,c)
}
PrintAll(...fruits)
</script>
</body>
</html>
```

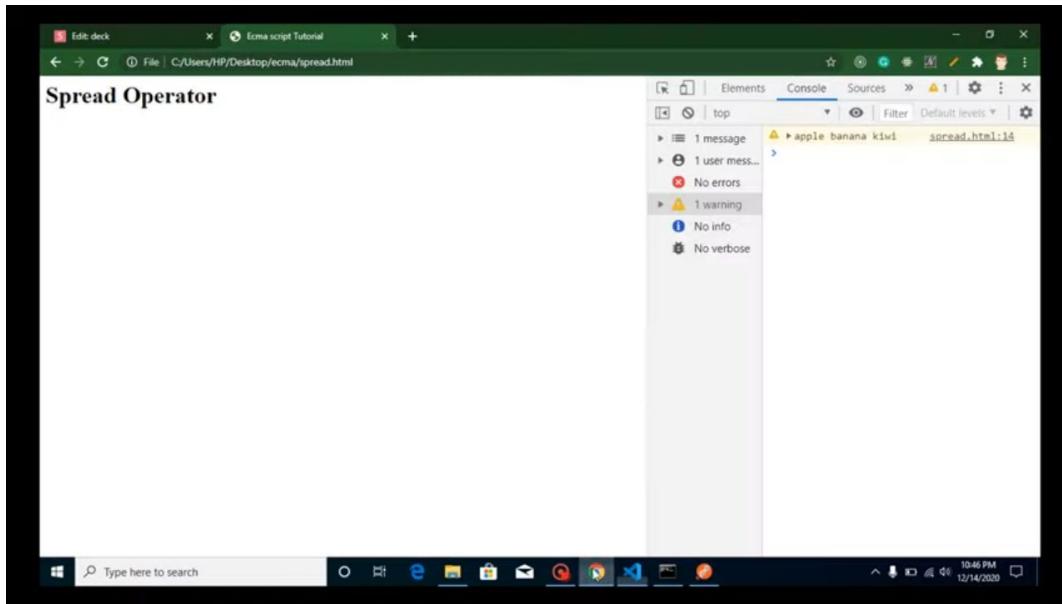
▶ 1:54:33



▶ 1:54:39



▶ 1:54:53



▶ 1:54:56

upper wala rest hai jab k nechay wala spred operator hai

▶ 1:55:45

```
File Edit Selection View Go Run Terminal Help spread.html - ecma - Visual Studio Code

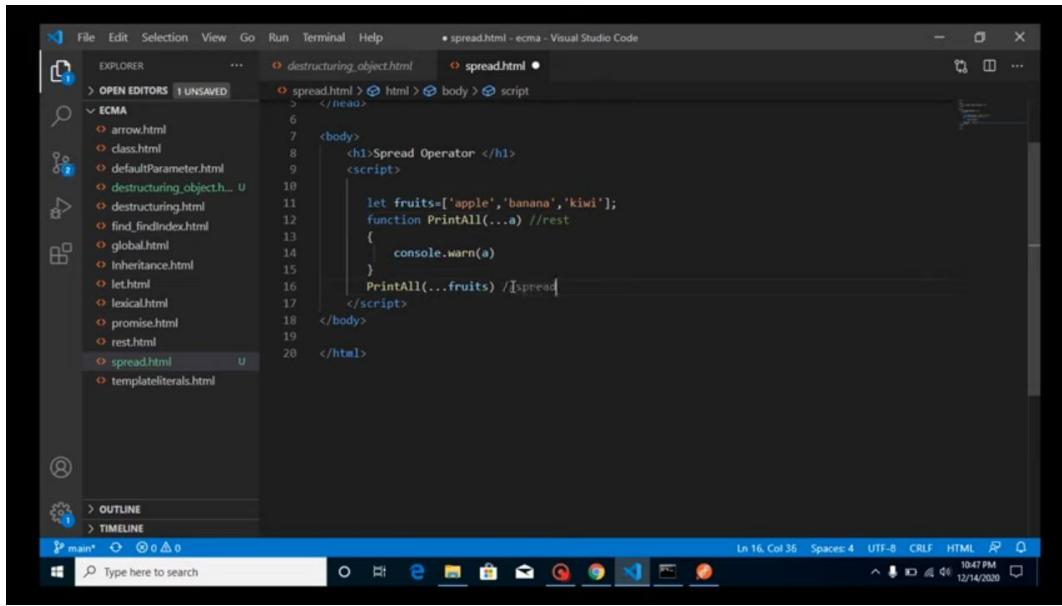
OPEN EDITORS
ECMA
arrow.html
class.html
defaultParameter.html
destructuring_object.html
destructuring_object.html
find_index.html
global.html
Inheritance.html
let.html
lexical.html
promise.html
rest.html
spread.html
template_literals.html

spread.html
<head>
</head>
<body>
<h1>Spread Operator </h1>
<script>
let fruits=['apple', (parameter) a: any[]]
function PrintAll(...a)
{
    console.warn(a)
}
PrintAll(...fruits)
</script>
</body>
</html>
```

▶ 1:55:45

rest operator sub ko ekhata kar detta hai

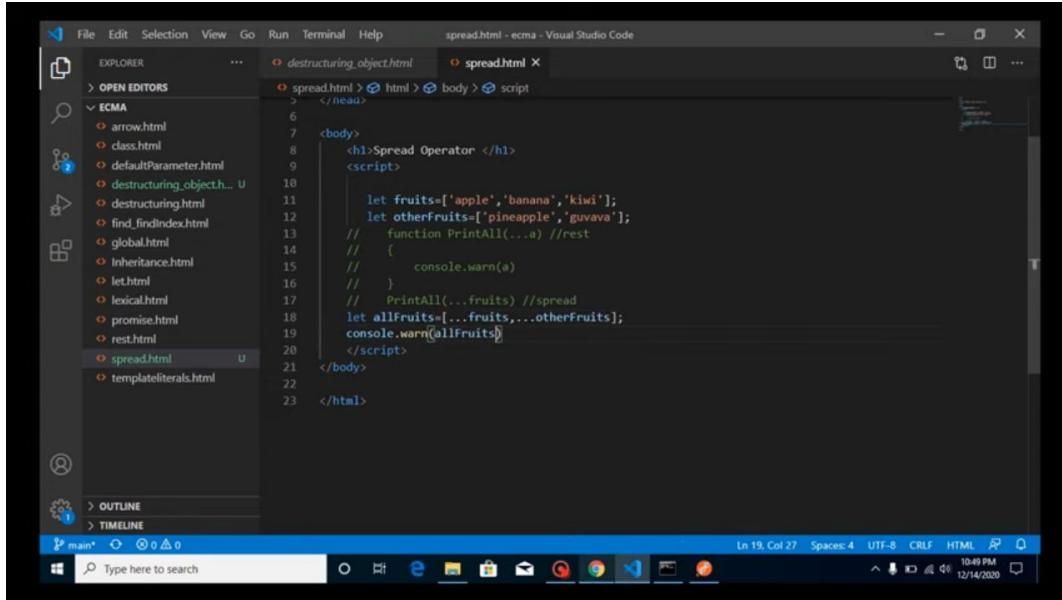
▶ 1:55:47



```
File Edit Selection View Go Run Terminal Help
OPEN EDITORS ... spread.html - ecmascript - Visual Studio Code
ECMA
arrow.html
class.html
defaultParameter.html
destructuring_object.html
destructuring.html
find_index.html
global.html
Inheritance.html
let.html
lexical.html
promise.html
rest.html
spread.html
template_literals.html
spread.html
</head>
<body>
  <h1>Spread Operator </h1>
  <script>
    let fruits=['apple','banana','kiwi'];
    function PrintAll(...a) //rest
    {
      console.warn(a)
    }
    PrintAll(...fruits) //spread
  </script>
</body>
</html>
```

Ln 16, Col 36 Spaces: 4 UTF-8 CRLF HTML ⚡ 10:47 PM 12/14/2020

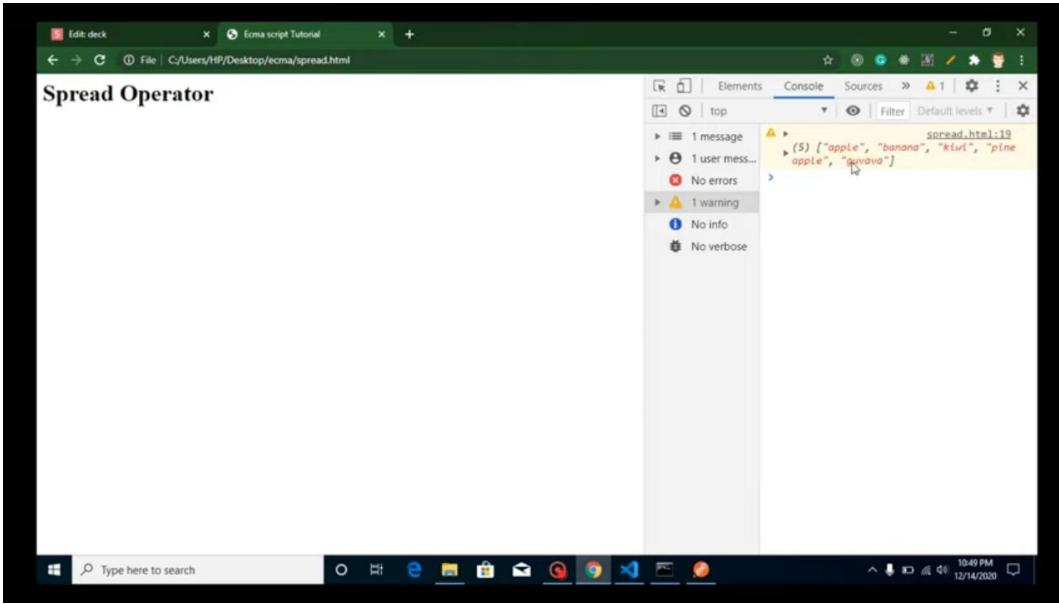
▶ 1:56:13



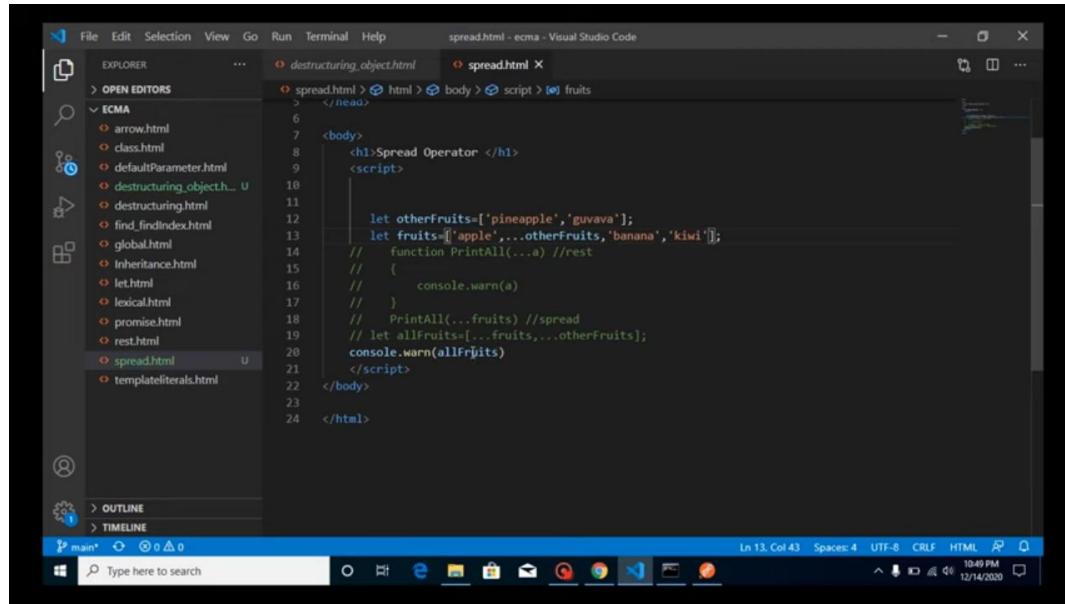
```
File Edit Selection View Go Run Terminal Help
OPEN EDITORS ... spread.html - ecmascript - Visual Studio Code
ECMA
arrow.html
class.html
defaultParameter.html
destructuring_object.html
destructuring.html
find_index.html
global.html
Inheritance.html
let.html
lexical.html
promise.html
rest.html
spread.html
template_literals.html
spread.html
</head>
<body>
  <h1>Spread Operator </h1>
  <script>
    let fruits=['apple','banana','kiwi'];
    let otherFruits=['pineapple','guava'];
    // function PrintAll(...a) //rest
    // {
    //   console.warn(a)
    // }
    // PrintAll(...fruits) //spread
    let allFruits=[...fruits,...otherFruits];
    console.warn(allFruits)
  </script>
</body>
</html>
```

Ln 19, Col 27 Spaces: 4 UTF-8 CRLF HTML ⚡ 10:49 PM 12/14/2020

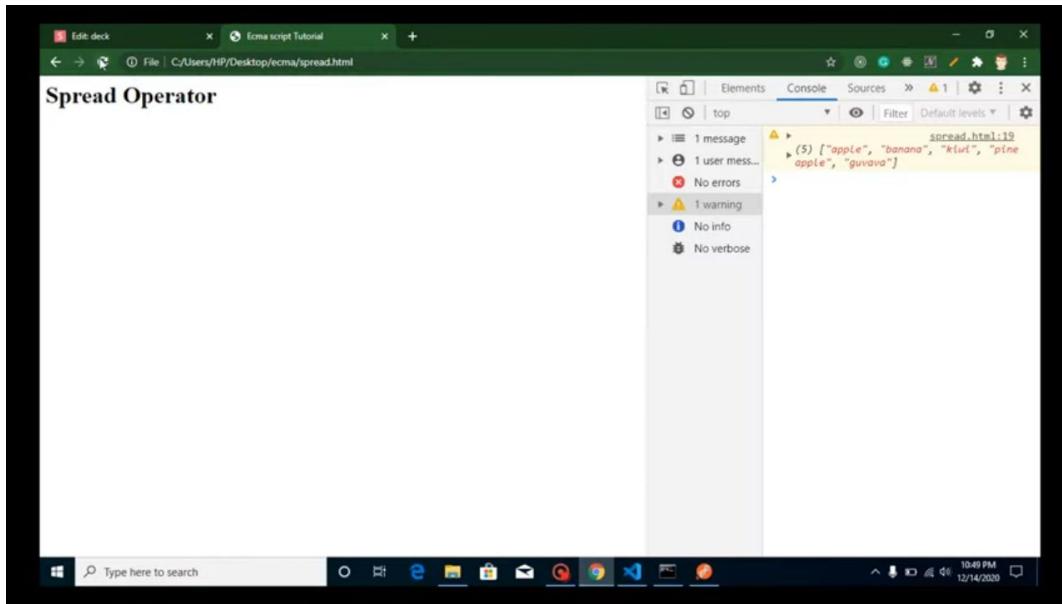
▶ 1:57:38



▶ 1:57:42



▶ 1:58:14

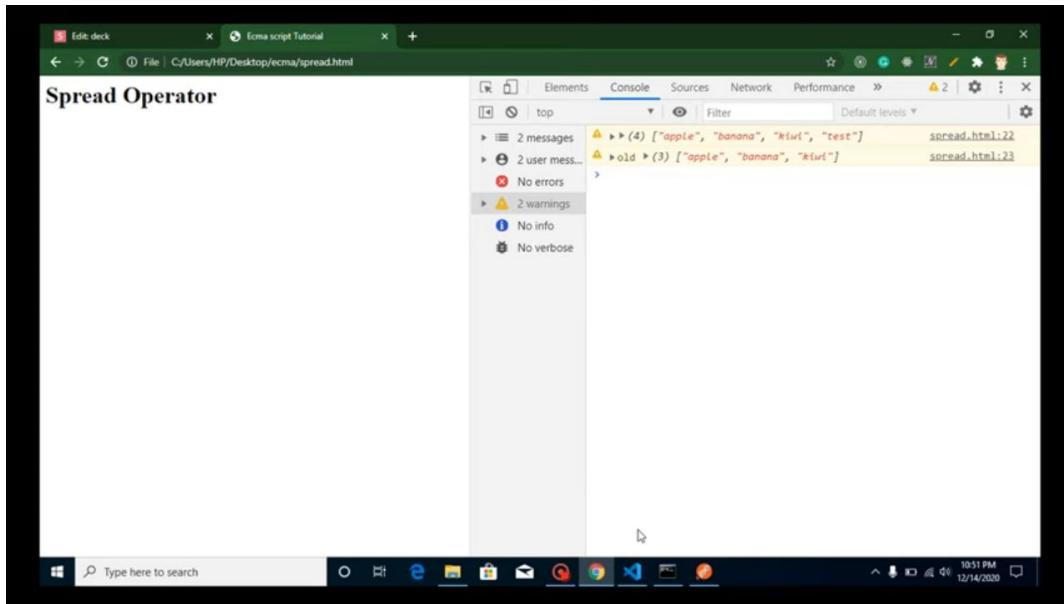


▶ 1:58:20

A screenshot of Visual Studio Code. The left sidebar shows the file structure under "OPEN EDITORS" and "ECMA" categories. The "spread.html" file is open in the editor. The code is as follows:

```
<head>
<body>
<h1>Spread Operator </h1>
<script>
// let otherFruits=['pineapple','guava'];
let fruits=['apple','banana','kiwi'];
// function PrintAll(...a) //rest
// (
//   console.warn(a)
// )
// PrintAll(...fruits) //spread
// let allFruits=[...fruits,...otherFruits];
let newFruits=[...fruits]
newFruits.push("test")
console.warn(newFruits)
console.warn("old",fruits)
</script>
</body>
</html>
```

▶ 2:00:17



▶ 2:00:25

reference ko break kar nay k leua [...fruits] lagaya hai

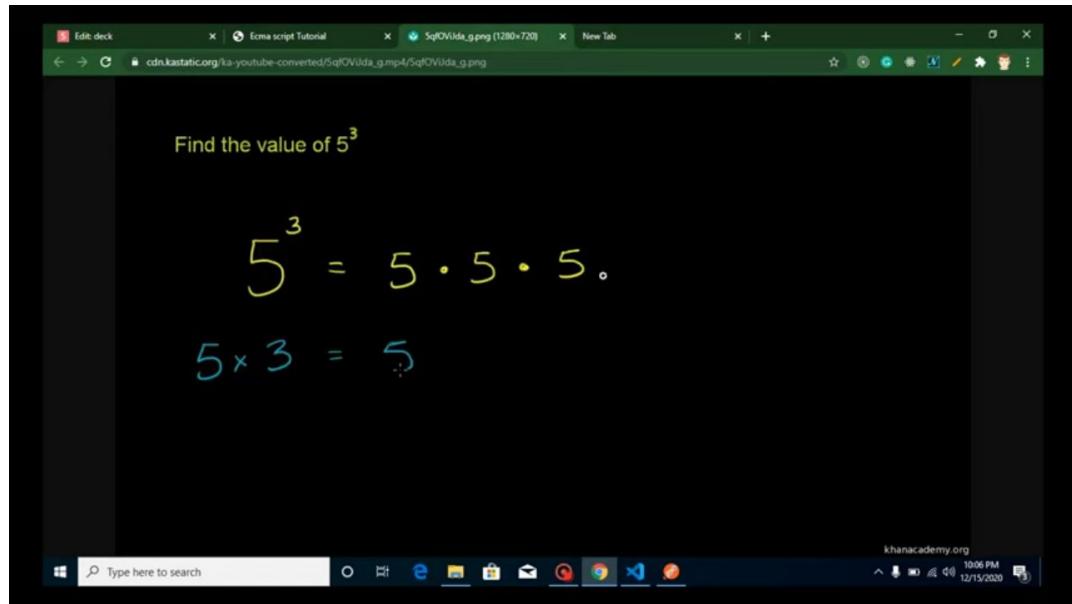
▶ 2:01:10

A screenshot of a presentation slide titled "Exponentiation Operator" from slides.com. The slide has a yellow background with white text. It lists five points:

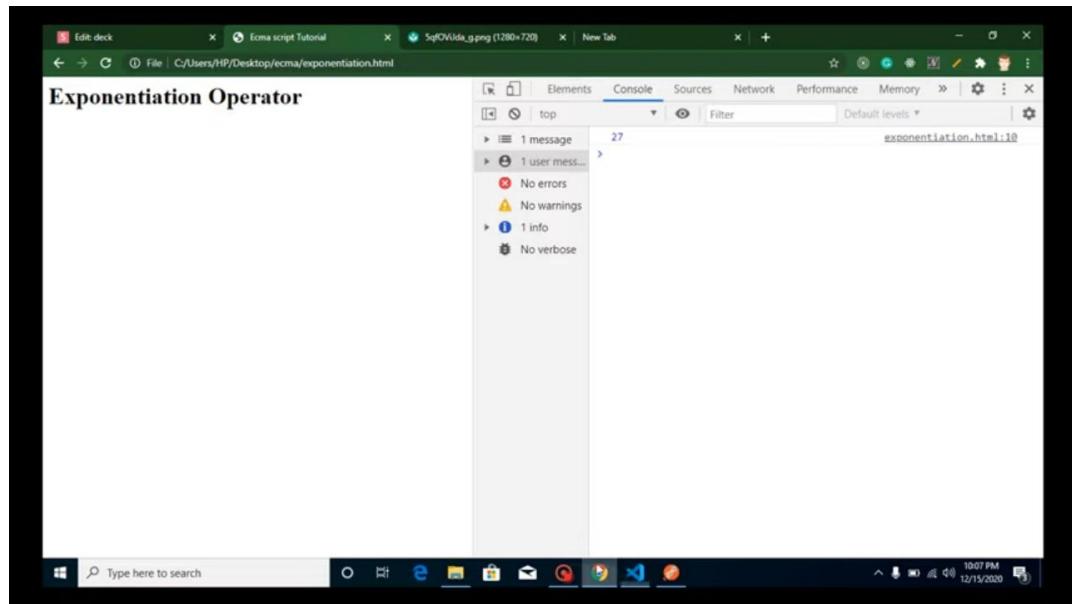
1. Exponentiation operator (**)
2. Alternative of Math.pow(2, 3)
3. How to use it.
4. Some Example.
5. Interview Questions.

A circular callout bubble in the bottom right corner contains the name "Anil Sidhu". The browser's toolbar and status bar are visible at the top and bottom respectively.

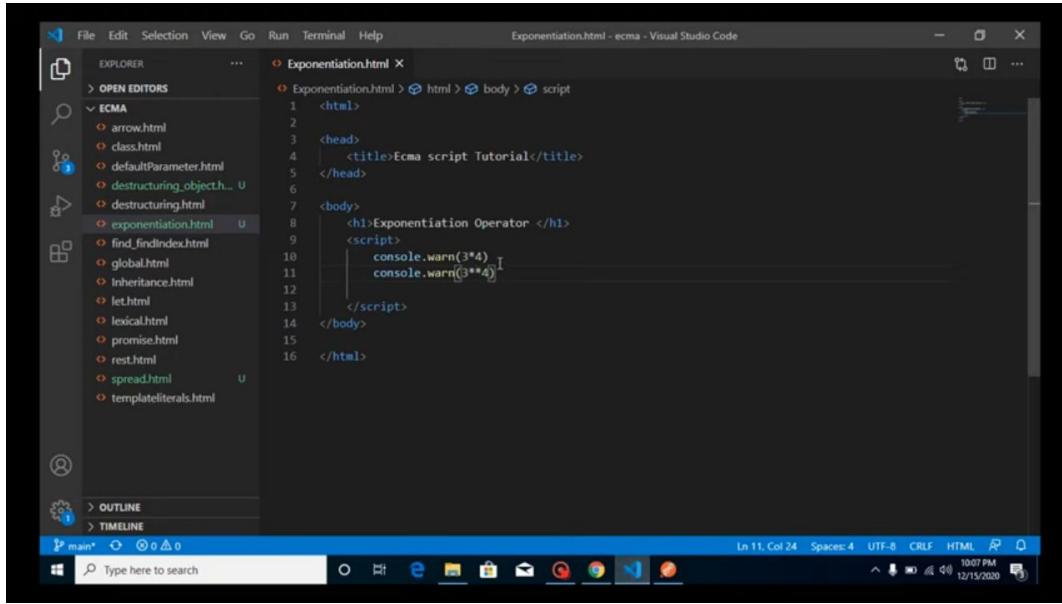
▶ 2:01:16



▶ 2:02:29

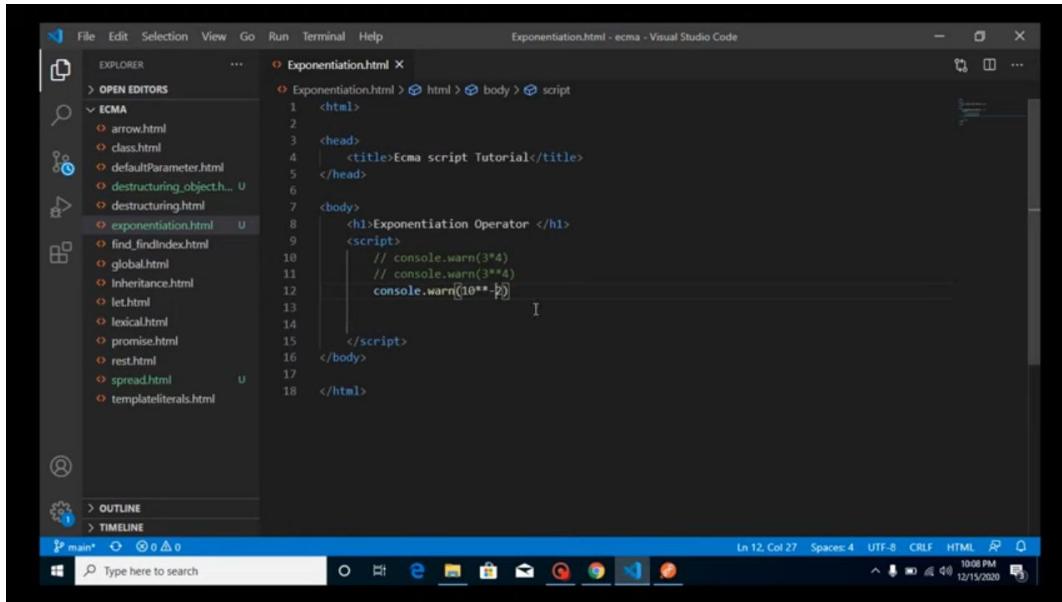


▶ 2:03:44



```
1 <html>
2
3 <head>
4   <title>Ecma script Tutorial</title>
5 </head>
6
7 <body>
8   <h1>Exponentiation Operator </h1>
9   <script>
10    console.warn(3**4)
11    console.warn(3***4)
12  </script>
13 </body>
14
15 </html>
```

▶ 2:03:52



```
1 <html>
2
3 <head>
4   <title>Ecma script Tutorial</title>
5 </head>
6
7 <body>
8   <h1>Exponentiation Operator </h1>
9   <script>
10    // console.warn(3**4)
11    // console.warn(3***4)
12    console.warn(10***4)
13  </script>
14 </body>
15
16 </html>
```

▶ 2:04:44

0.01 dee ga ya

▶ 2:05:24

The screenshot shows the Visual Studio Code interface with the file 'Exponentiation.html' open. The code in the editor is:

```
1 <html>
2   <head>
3     <title>Ecma script Tutorial</title>
4   </head>
5   <body>
6     <h1>Exponentiation Operator </h1>
7     <script>
8       // console.warn(3**4)
9       // console.warn(3***4)
10      // console.warn(10**-2)
11      // console.warn(10**NaN)
12      console.warn(8**3***2)
13
14     </script>
15   </body>
16
17 </html>
```

The status bar at the bottom indicates 'Ln 14, Col 29'. The taskbar at the bottom shows icons for various applications.

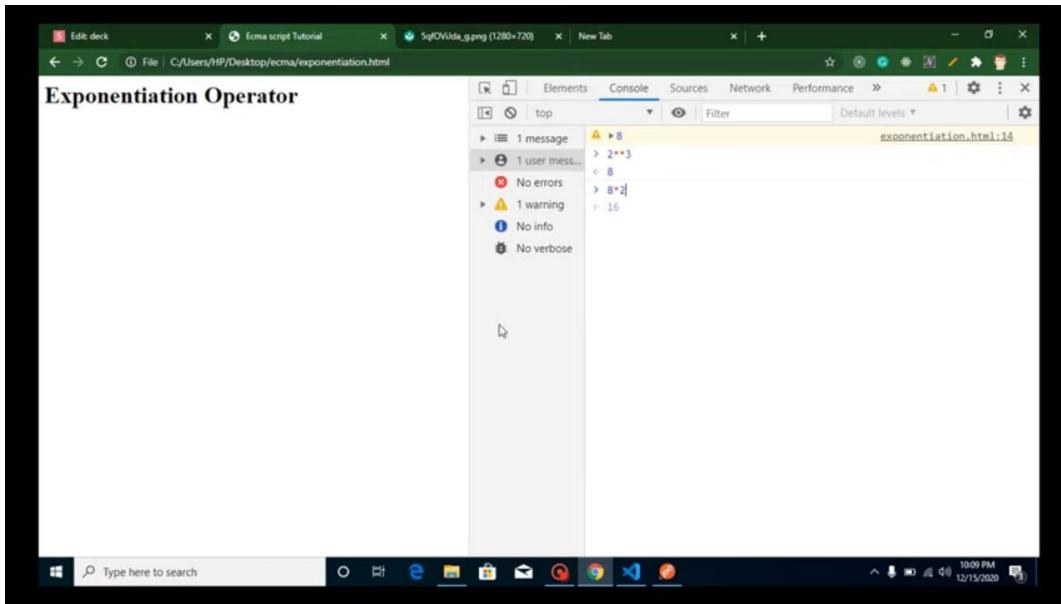
▶ 2:05:45

The screenshot shows the browser developer tools open to the 'Console' tab. The output window displays the following messages:

- > 8
- > 2**3
- < 8
- > 8**8
- < 16777216

The status bar at the bottom indicates 'Ln 14, Col 29'. The taskbar at the bottom shows icons for various applications.

▶ 2:06:05

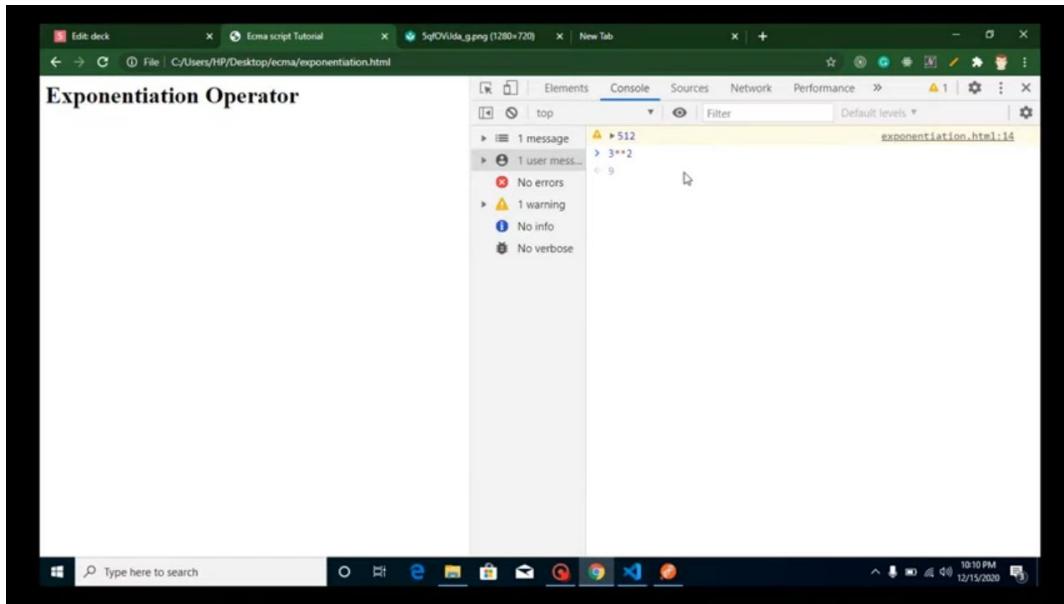


▶ 2:06:09

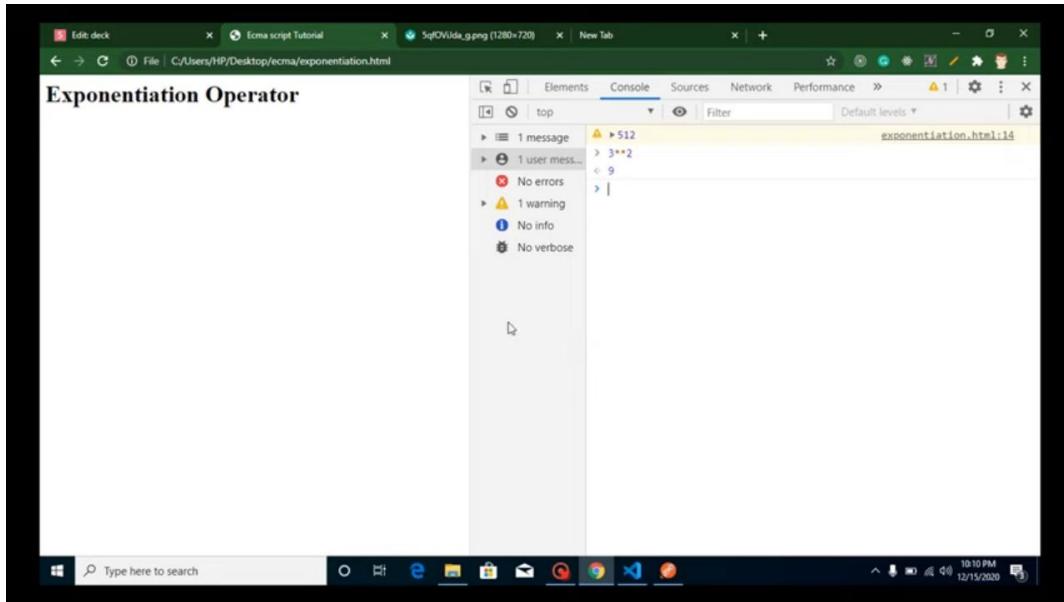
A screenshot of Visual Studio Code showing the code for "Exponentiation.html". The code is as follows:

```
1 <html>
2   <head>
3     <title>Ecma script Tutorial</title>
4   </head>
5
6   <body>
7     <h1>Exponentiation Operator </h1>
8     <script>
9       // console.warn(3*4)
10      // console.warn(3**4)
11      // console.warn(10**2)
12      // console.warn(10**NaN)
13      // console.warn(2**3**2)
14
15    </script>
16
17  </body>
18
19 </html>
```

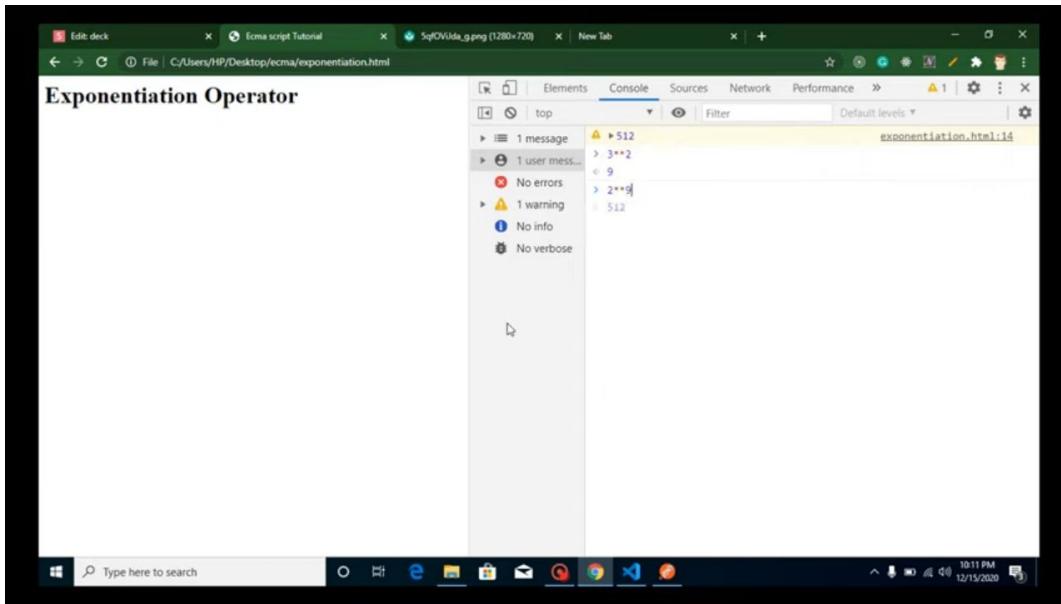
▶ 2:06:43



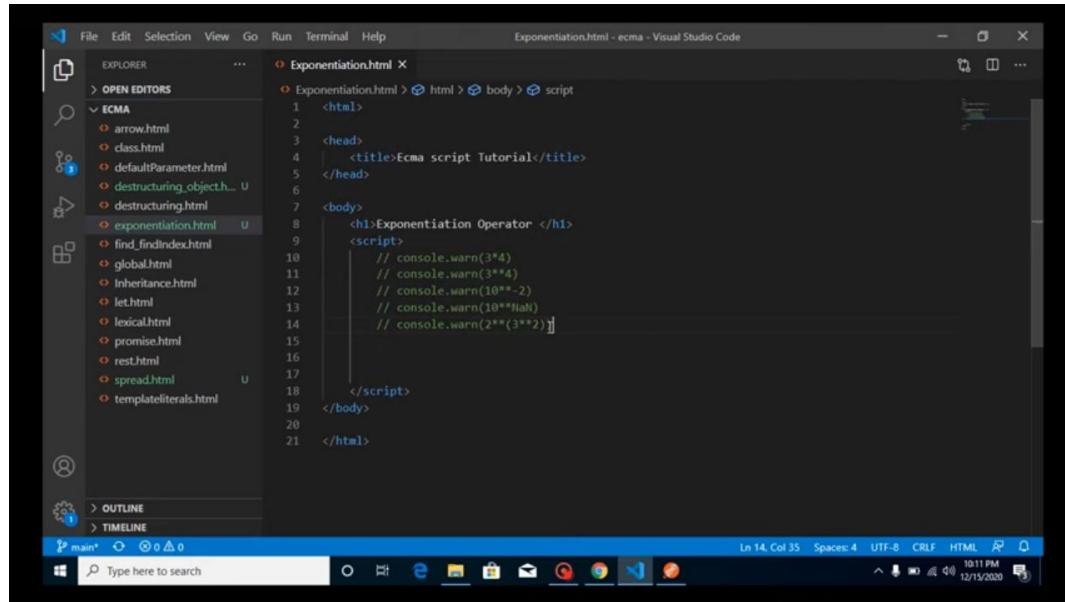
▶ 2:06:48



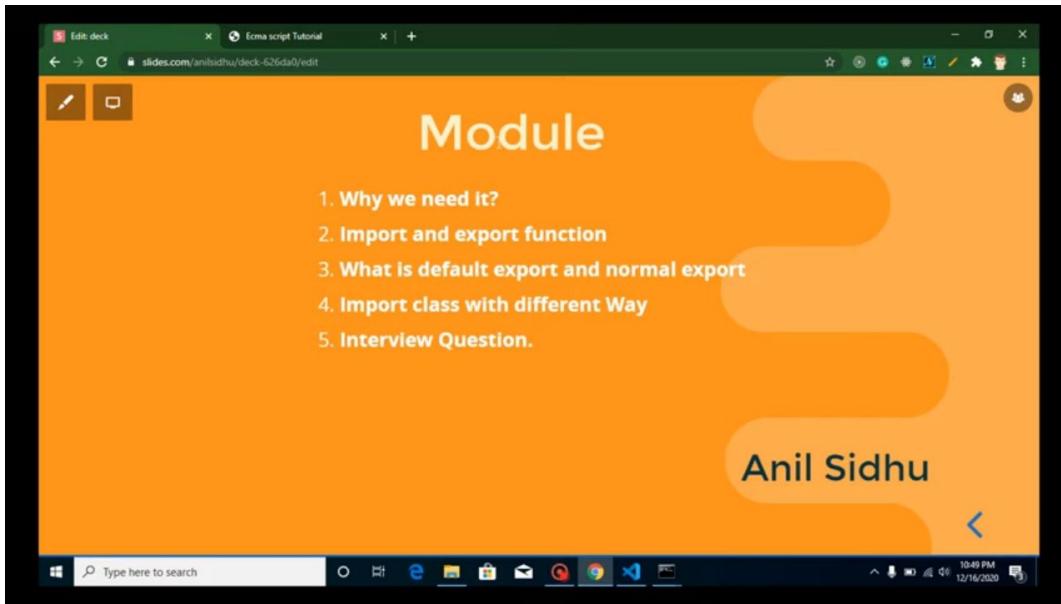
▶ 2:07:00



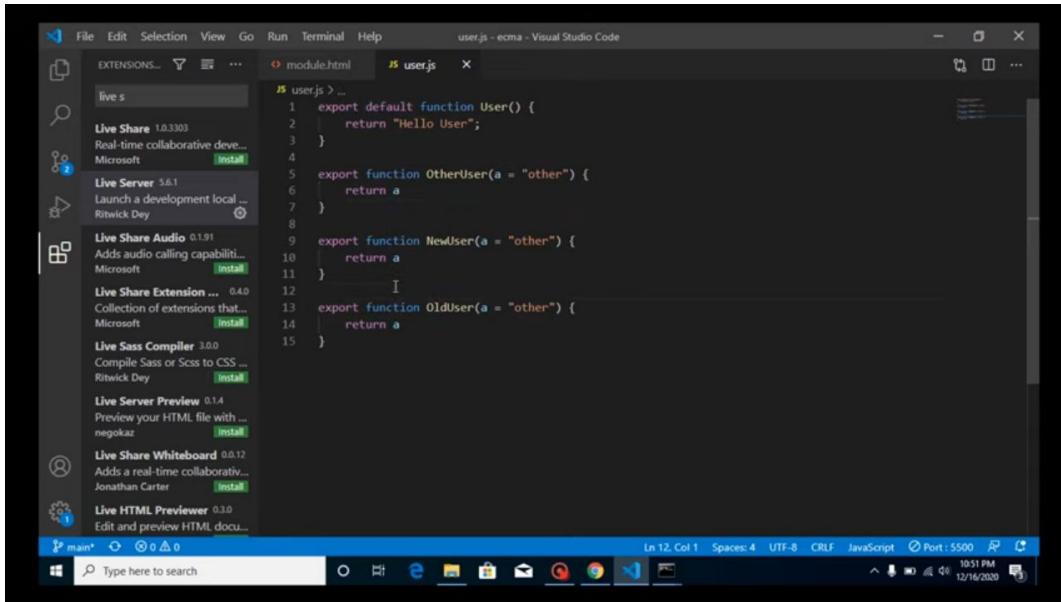
▶ 2:07:03



▶ 2:07:26



▶ 2:09:04



▶ 2:10:37

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows extensions installed: Live Share (1.0.3303), Live Server (5.6.1), Live Share Audio (0.1.91), Live Share Extension (0.4.0), Live Sass Compiler (3.0.0), Live Server Preview (0.1.4), Live Share Whiteboard (0.0.12), and Live HTML Previewer (0.3.0).
- Code Editor:** Two files are open: `module.html` and `user.js`.
 - `module.html` contains the following code:

```
<html>
<head>
    <title>Ecma script Tutorial</title>
</head>
<body>
    <h1>Module</h1>
    <script type="module" defer>
        import User './user.js';
    </script>
</body>
</html>
```
 - `user.js` contains the following code:

```
export default function User() {
    return "Hello User";
}

export function OtherUser(a = "other") {
    return a;
}

export function NewUser(a = "other") {
    return a;
}

export function OldUser(a = "other") {
    return a;
}
```
- Terminal:** Shows the command `user.js` was run.
- Status Bar:** Shows the file is ECMAScript, has 13 lines and 4 columns, uses UTF-8 encoding, and is running on port 5500.

▷ 2:10:45

module allow karta hai ek file k koch code ko dosri file main use kar nay k leya

▷ 2:11:14

ek file export karti hai or dosri file import karti hai

▷ 2:12:52

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows extensions installed: Live Share (1.0.3303), Live Server (5.6.1), Live Share Audio (0.1.91), Live Share Extension (0.4.0), Live Sass Compiler (3.0.0), Live HTML Previewer (0.3.0), and Live Server Preview (0.1.4). Additionally, Emmet Live (1.0.0) is listed.
- Code Editor:** Two files are open: `module.html` and `user.js`.
 - `module.html` contains the following code:

```
<html>
<head>
    <title>Ecma script Tutorial</title>
</head>
<body>
    <h1>Module</h1>
    <script type="module" defer>
        import User './user.js';
    </script>
</body>
</html>
```
 - `user.js` contains the same code as the previous screenshot:

```
export default function User() {
    return "Hello User";
}

export function OtherUser(a = "other") {
    return a;
}

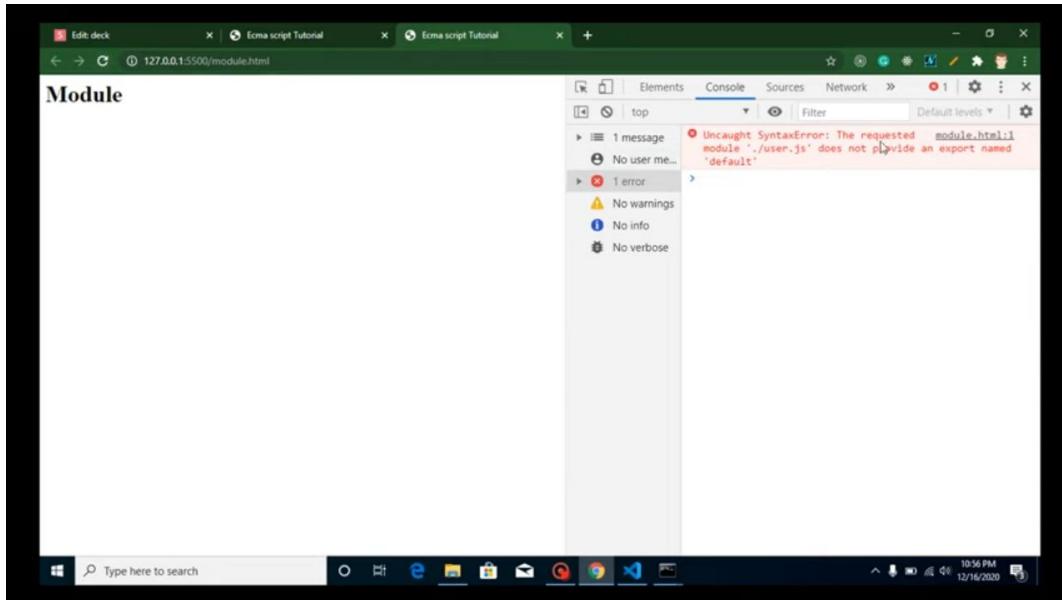
export function NewUser(a = "other") {
    return a;
}

export function OldUser(a = "other") {
    return a;
}
```
- Terminal:** Shows the command `user.js` was run.
- Status Bar:** Shows the file is ECMAScript, has 13 lines and 5 columns, uses UTF-8 encoding, and is running on port 5500.

▷ 2:15:34

```
module.html X user.js
live
Live Share 1.0.3303
Real-time collaborative deve...
Microsoft
Live Server 5.6.1
Launch a development local ...
Ritwick Dey
Live Share Audio 0.1.91
Adds audio calling capabilit...
Microsoft
Live Share Extension ... 0.4.0
Collection of extensions that...
Microsoft
Live Sass Compiler 3.0.0
Compile Sass or Scss to CSS ...
Ritwick Dey
Live HTML Previewer 0.3.0
Edit and preview HTML docu...
Harshdeep Gupta
Live Server Preview 0.1.4
Preview your HTML file with ...
negokaz
Emmet Live 1.0.0
Expand your Emmet abbrevi...
Ln 10, Col 38 Spaces: 4 UTF-8 CRLF HTML Port: 5500
main* 0 0 0 0
Type here to search
```

▶ 2:15:46



▶ 2:15:59

A screenshot of Visual Studio Code showing a workspace with two files open: `module.html` and `user.js`. The `user.js` file contains the following code:user.js
1 export default function User()
2 {
3 return "Hello User";
4 }The `module.html` file contains the following code:module.html
1 <html>
2 <head>
3 <title>Ecma script Tutorial</title>
4 </head>
5 <body>
6 <h1>Module</h1>
7 <script type="module" defer>
8 import User,(UserOther) from './user.js';
9 function UserOther()
10 {
11 console.warn(`in current \${[]}`);
12 }
13 User();
14 UserOther();
15 </script>
16 </body>
17 </html>The status bar at the bottom shows the file is in JavaScript mode (JavaScript), port 5500 is open, and the current time is 10:56 PM on 12/16/2020.

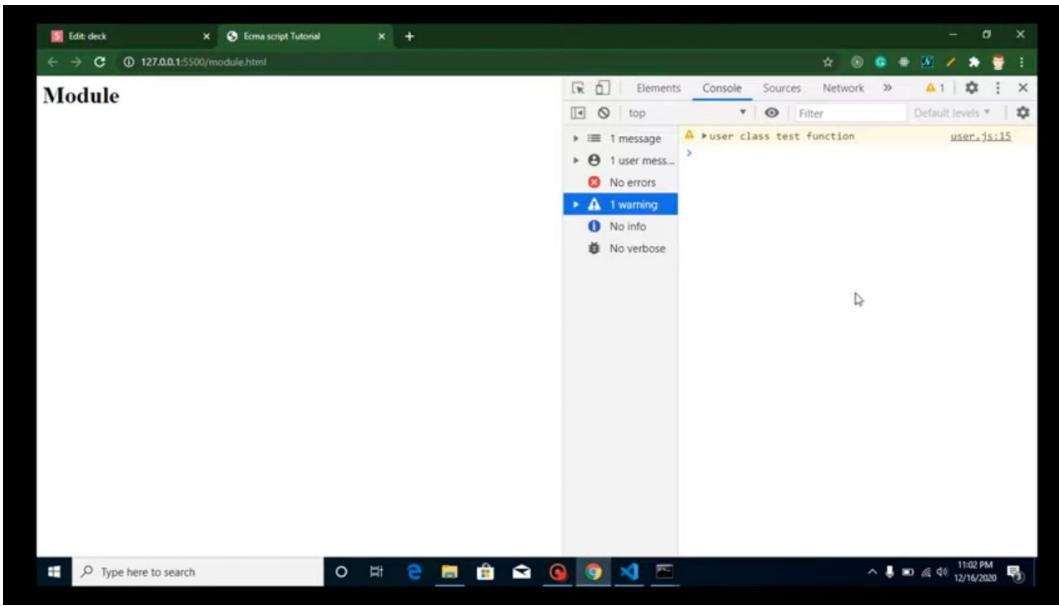
▶ 2:16:15

ek file main he he export defult ho ga baqi nhi hon gay

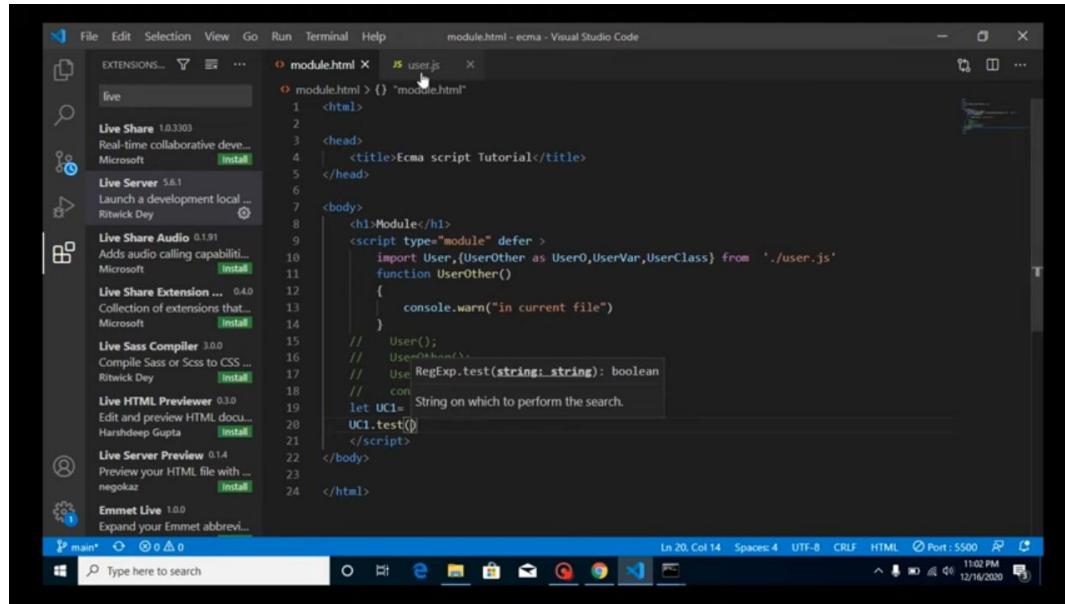
▶ 2:16:52

A screenshot of Visual Studio Code showing the execution result of the module-based code. The browser tab shows the output of `module.html`:<!DOCTYPE html>
<html>
<head>
<title>Ecma script Tutorial</title>
</head>
<body>
<h1>Module</h1>
<script type="module" defer>
import User,(UserOther) from './user.js';
function UserOther()
{
 console.warn(`in current \${[]}`);
}
User();
UserOther();
</script>
</body>
</html>The browser window displays the rendered HTML with the heading "Module" and the warning message "in current []". The status bar at the bottom shows the file is in HTML mode (HTML), port 5500 is open, and the current time is 10:59 PM on 12/16/2020.

▶ 2:19:43



▶ 2:22:44



▶ 2:22:48

```
user.js 1 user.js
1  export default function User()
2  {
3      console.warn("hello user")
4  }
5
6  export function UserOther()
7  {
8      console.warn("hello other user")
9  }
10 export let UserVar="hello user var";
11
12 export class UserClass{
13     test()
14     {
15         console.warn("user class test function")
16     }
17 }
18
19 class UserClass{
20     test()
21     {
22         console.warn("user class test function")
23     }
24 }
```

Live Share 1.0.3303
Real-time collaborative deve...
Microsoft [Install](#)

Live Server 5.6.1
Launch a development local ...
Ritwick Dey

Live Share Audio 0.1.91
Adds audio calling capabilit...
Microsoft [Install](#)

Live Share Extension ... 0.4.0
Collection of extensions that...
Microsoft [Install](#)

Live Sass Compiler 3.0.0
Compile Sass or Scss to CSS ...
Ritwick Dey [Install](#)

Live HTML Previewer 0.3.0
Edit and preview HTML docu...
Harshdeep Gupta [Install](#)

Live Server Preview 0.1.4
Preview your HTML file with ...
negokaz [Install](#)

Emmet Live 1.0.0
Expand your Emmet abbrevi...

main* 0 0 0 0 Type here to search

Ln 19, Col 1 Spaces:4 UTF-8 CRLF Port: 5500

11:03 PM 12/16/2020

▶ 2:22:58

Generators

1. What Is Generators
2. How to make Generator function
3. what Is yield In Generators
4. How to use it
5. Interview Question.

Anil Sidhu

▶ 2:24:43

function hotay hai in ko poss or resume kar saktay hai

▶ 2:25:14

A screenshot of Visual Studio Code showing the file `generators.html`. The code defines a generator function `steps` that logs the sum of `a` and `b` to the console:

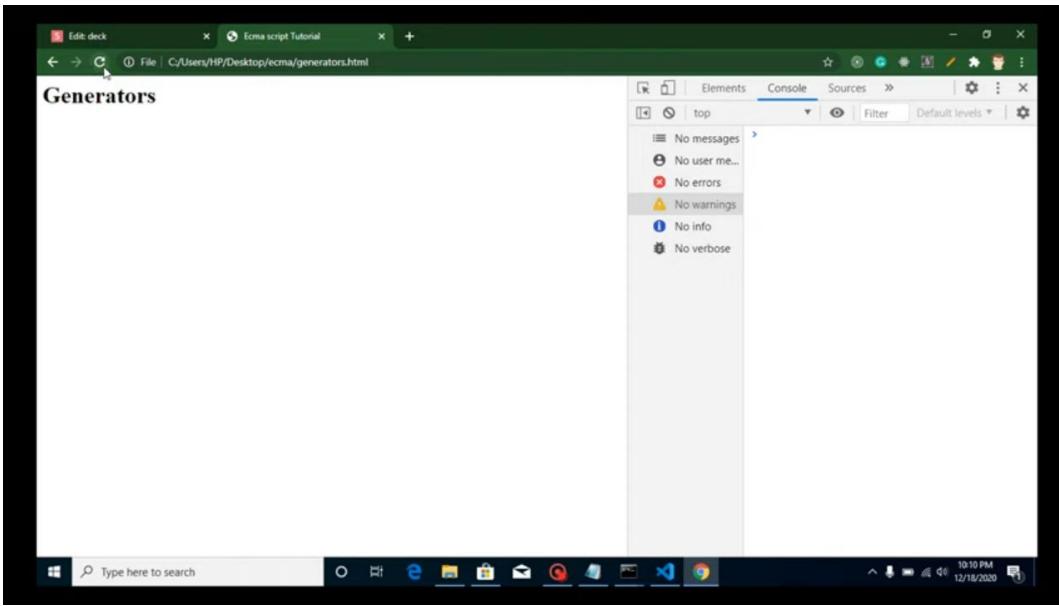
```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>Generators</h1>
    <script>
      function* steps()
      {
        let a=10;
        let b=20;
        console.warn([10+20])
      }
    </script>
  </body>
</html>
```

▶ 2:26:59

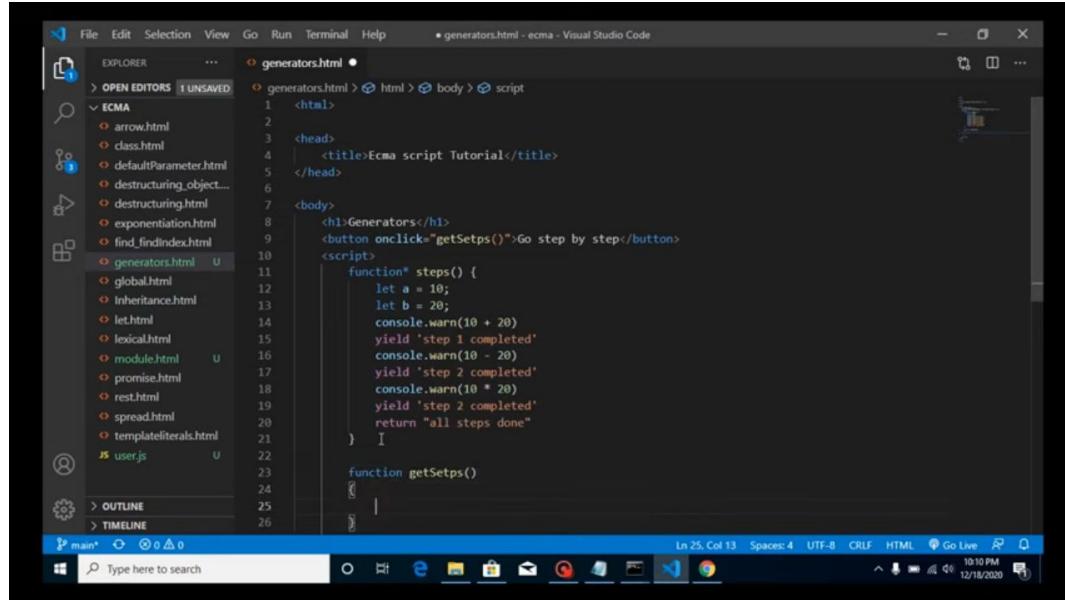
A screenshot of Visual Studio Code showing the file `generators.html`. The code now includes `yield` statements to return values from the generator function `steps`:

```
<html>
  <head>
    <title>Ecma script Tutorial</title>
  </head>
  <body>
    <h1>Generators</h1>
    <script>
      function* steps()
      {
        let a=10;
        let b=20;
        console.warn('step 1 completed')
        yield 'step 2 completed'
        console.warn('step 2 completed')
        console.warn([10+20])
        yield 'step 1 completed'
      }
    </script>
  </body>
</html>
```

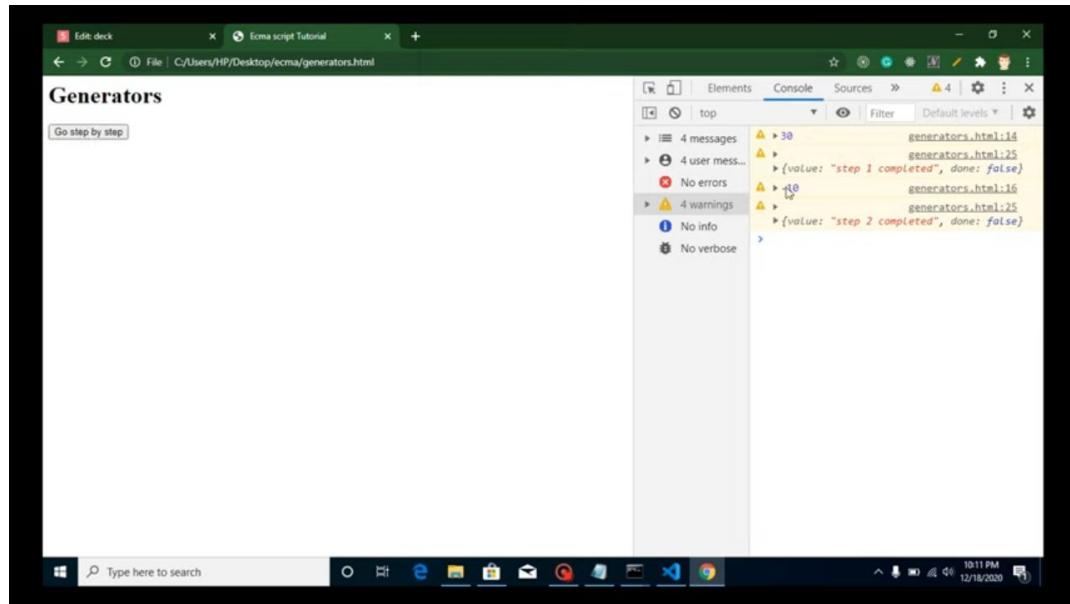
▶ 2:27:40



▶ 2:28:16



▶ 2:29:04



▶ 2:30:01

yield steps banany k kam aey tay hai

▶ 2:31:00