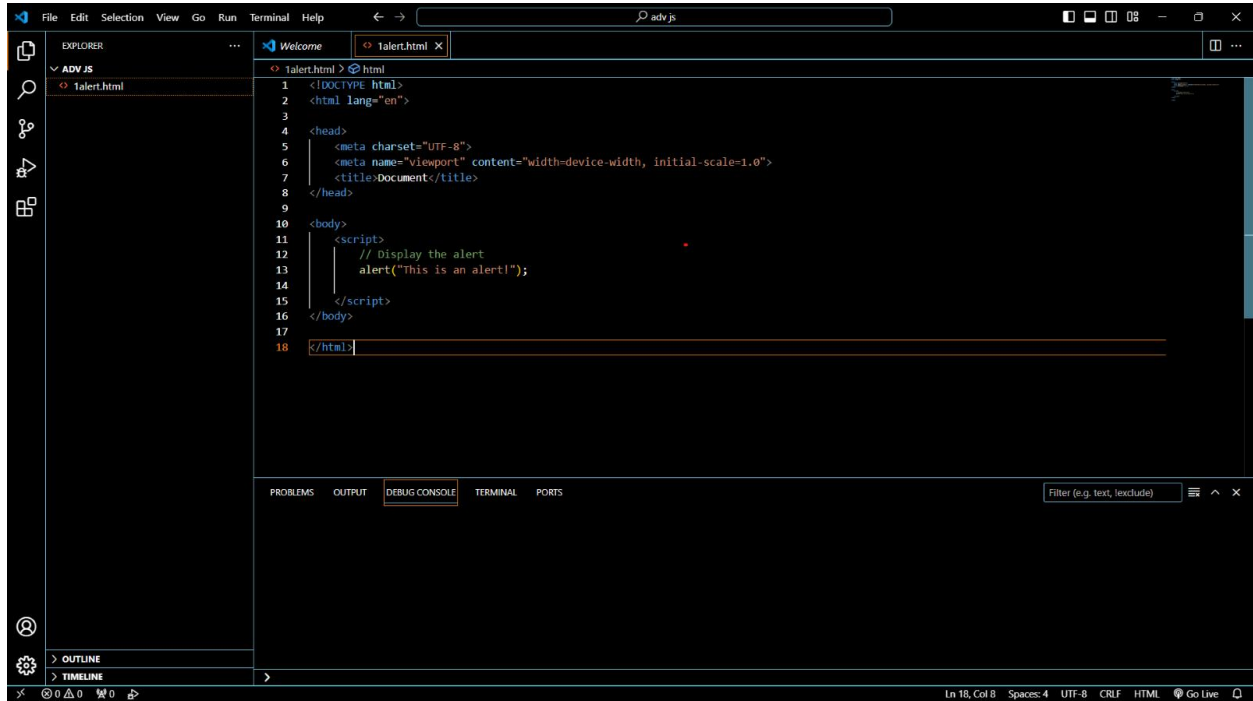


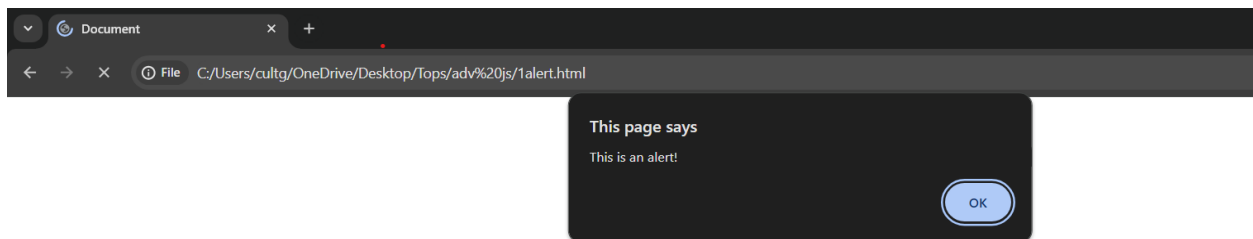
Advanced JavaScript

1. Write a program to Show an alert

-->



```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <script>
12     // Display the alert
13     alert("This is an alert!");
14   </script>
15 </body>
16 </html>
```



2. What will be the result for these expressions?

1. $5 > 4$

2. $"apple" > "pineapple"$

3. $"2" > "12"$

4. $undefined == null$

5. $undefined === null$

6. $null == "\n0\n"$

7. $null === +"\n0\n"$

-->1. true

2. false

3. true

4. true

5. false

6. false

7. false

3. Will alert be shown?

if ("0") { alert('Hello'); }

--> Yes the alert will be shown.

4. What is the code below going to output? alert(null || 2 || undefined);

--> The output will be 2.

5. The following function returns true if the parameter age is greater than 18. Otherwise it asks for a confirmation and returns its result:

function

checkAge(age)

{

```
if (age > 18) { return true; }  
else {  
// ...return confirm ('did parents allow  
you?');  
}  
}
```

--> In this function:

- If the age is greater than 18, it immediately returns true.
- If the age is not greater than 18, it prompts the user with a confirmation dialog asking whether the parents allowed it. The function then returns the result of this confirmation dialog. If the

user clicks OK, it returns true; if the user clicks Cancel, it returns false.

6. Replace Function Expressions with arrow functions in the code below:

Function

ask(question, yes, no)

{ if (confirm(question))yes();

else

no();

}

ask("Do you agree?", function()

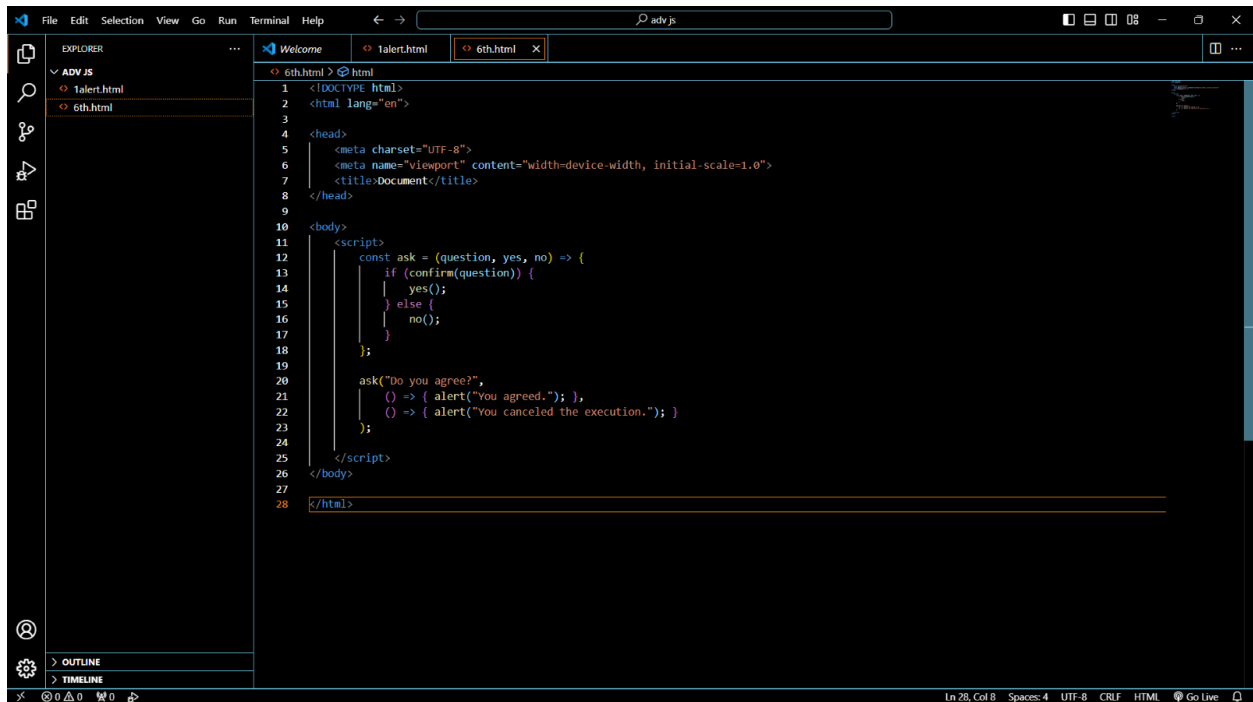
{ alert("You agreed."); },

function() {

alert("You canceled the execution.");

}

-->



```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <script>
12     const ask = (question, yes, no) => {
13       if (confirm(question)) {
14         yes();
15       } else {
16         no();
17       }
18     };
19
20     ask("Do you agree?",
21       () => { alert("You agreed."); },
22       () => { alert("You canceled the execution."); }
23     );
24   </script>
25 </body>
26
27
28 </html>
```

Data Types and Objects

7. Write the code, one line for each action:

a) Create an empty object user.

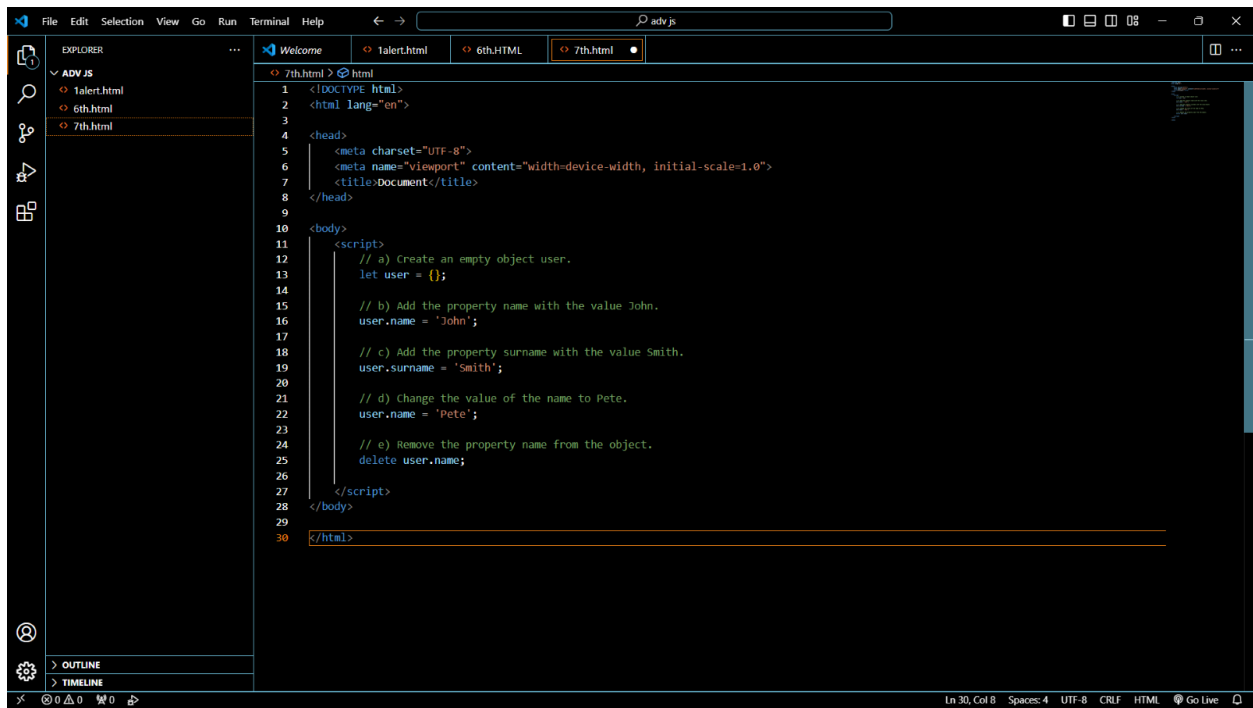
b) Add the property name with the value John.

c) Add the property surname with the value Smith.

d) Change the value of the name to Pete.

e) Remove the property name from the object.

-->



The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left shows a project named 'ADV JS' with files '1alert.html', '6th.html', and '7th.html'. The main editor area is open to '7th.html', which contains the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <script>
12     // a) Create an empty object user.
13     let user = {};
14
15     // b) Add the property name with the value John.
16     user.name = 'John';
17
18     // c) Add the property surname with the value Smith.
19     user.surname = 'Smith';
20
21     // d) Change the value of the name to Pete.
22     user.name = 'Pete';
23
24     // e) Remove the property name from the object.
25     delete user.name;
26
27   </script>
28 </body>
29
30 </html>
```

The status bar at the bottom indicates 'Ln 30, Col 8', 'Spaces: 4', 'UTF-8', 'CRLF', 'HTML', and 'Go Live'.

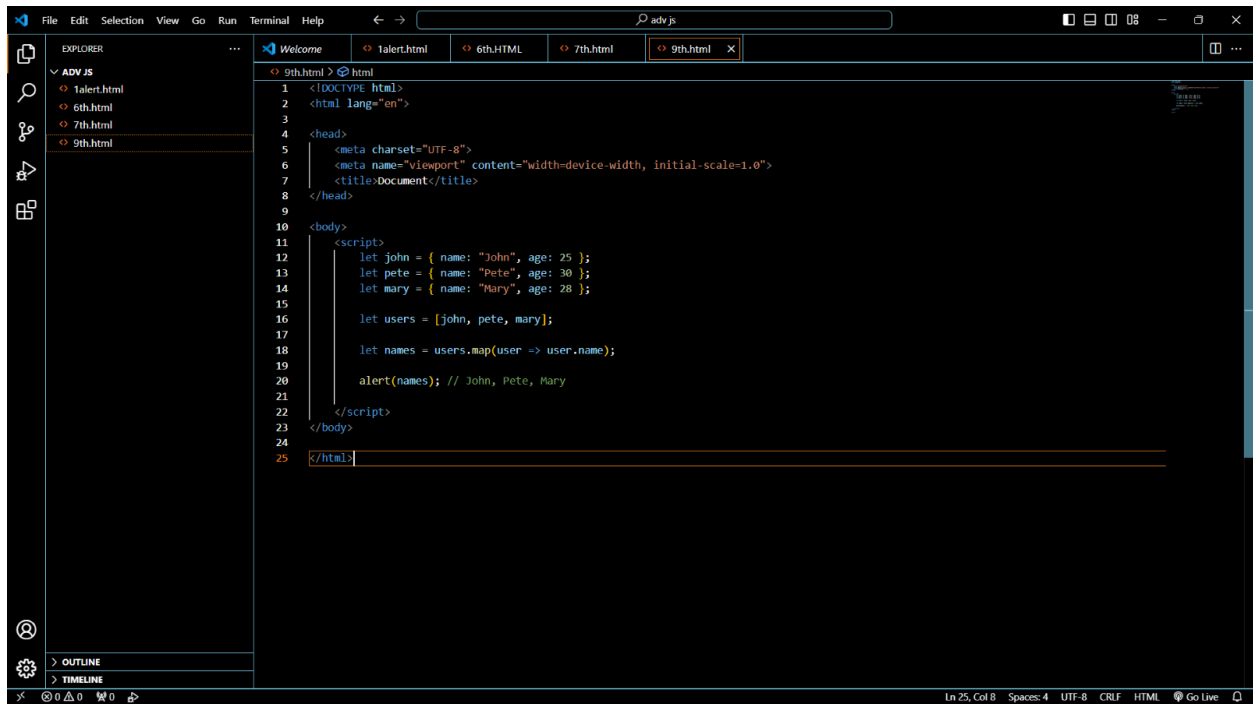
8. Is array copied?

```
let fruits = ["Apples", "Pear", "Orange"];  
// push a new value into the "copy" let  
shoppingCar=fruits;  
shoppingCart.push("Banana"); // what's  
in fruits?  
  
alert( fruits.length ); // ?
```

--> Yes this array is copied.

```
9. Map to names let john = { name:  
"John", age: 25 }; let pete = { name:  
"Pete", age: 30 }; let mary = { name:  
"Mary", age: 28 }; let users = [john, pete,  
mary ]; let names = /* ... your code */  
alert( names ); // John, Pete, Mary
```


-->



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left lists files: 1alert.html, 6th.html, 7th.html, and 9th.html. The main editor area displays the content of 9th.html, which is an HTML document. The code includes a DOCTYPE declaration, HTML and head tags with charset and viewport meta tags, and a script block. The script defines three user objects (john, pete, mary), combines them into an array (users), and uses the map function to create a new array (usersMapped) with full names. Finally, it calls alert(names) to display the names of the users.

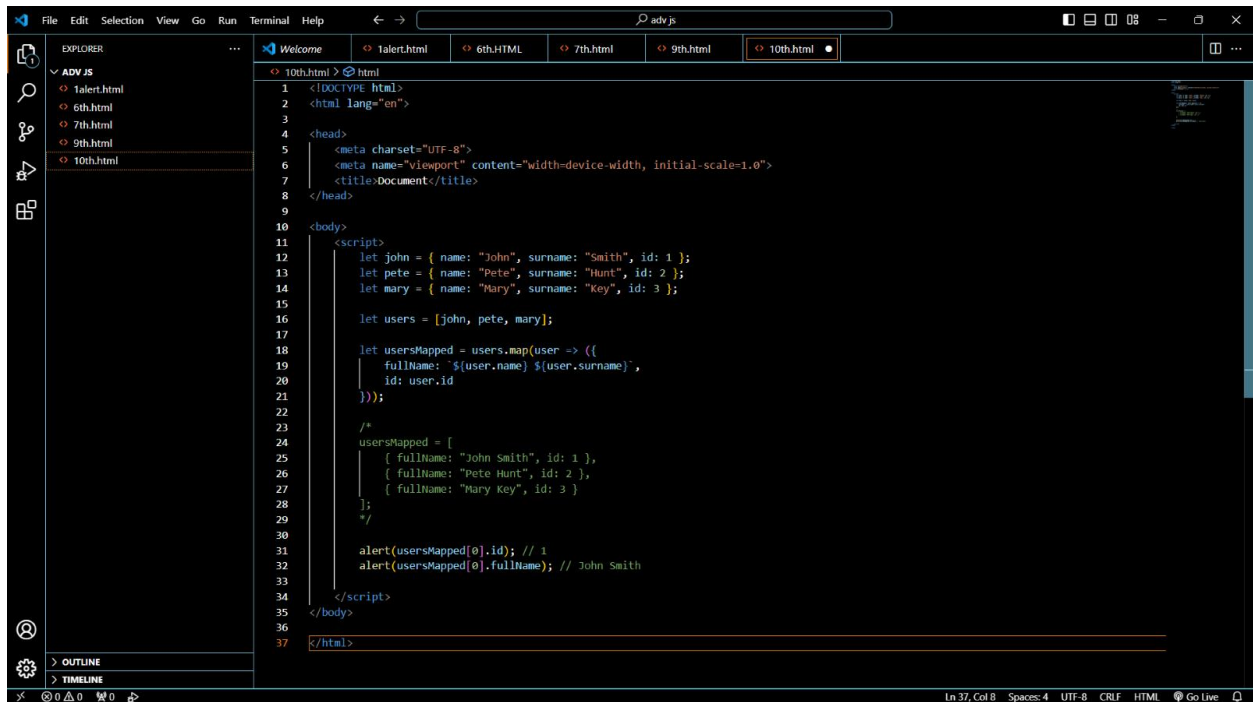
```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <script>
12     let john = { name: "John", age: 25 };
13     let pete = { name: "Pete", age: 30 };
14     let mary = { name: "Mary", age: 28 };
15
16     let users = [john, pete, mary];
17
18     let names = users.map(user => user.name);
19
20     alert(names); // John, Pete, Mary
21
22   </script>
23 </body>
24
25 </html>
```

10. Map to objects let john = { name: "John", surname: "Smith",id:1 }; let pete = { name:"Pete", surname:"Hunt", id: 2 };letmary={name:"Mary",surname:"Key",id:3};let users = [john, pete, mary]; let usersMapped = /* ... your code ... */ /* usersMapped = [{ fullName: "John Smith", id: 1 }, { fullName: "Pete Hunt",

id: 2 }, { fullName: "Mary Key", id: 3 }]

**/alert(usersMapped[0].id)//1alert(user
sMapped[0].fullName// John Smith.*

-->

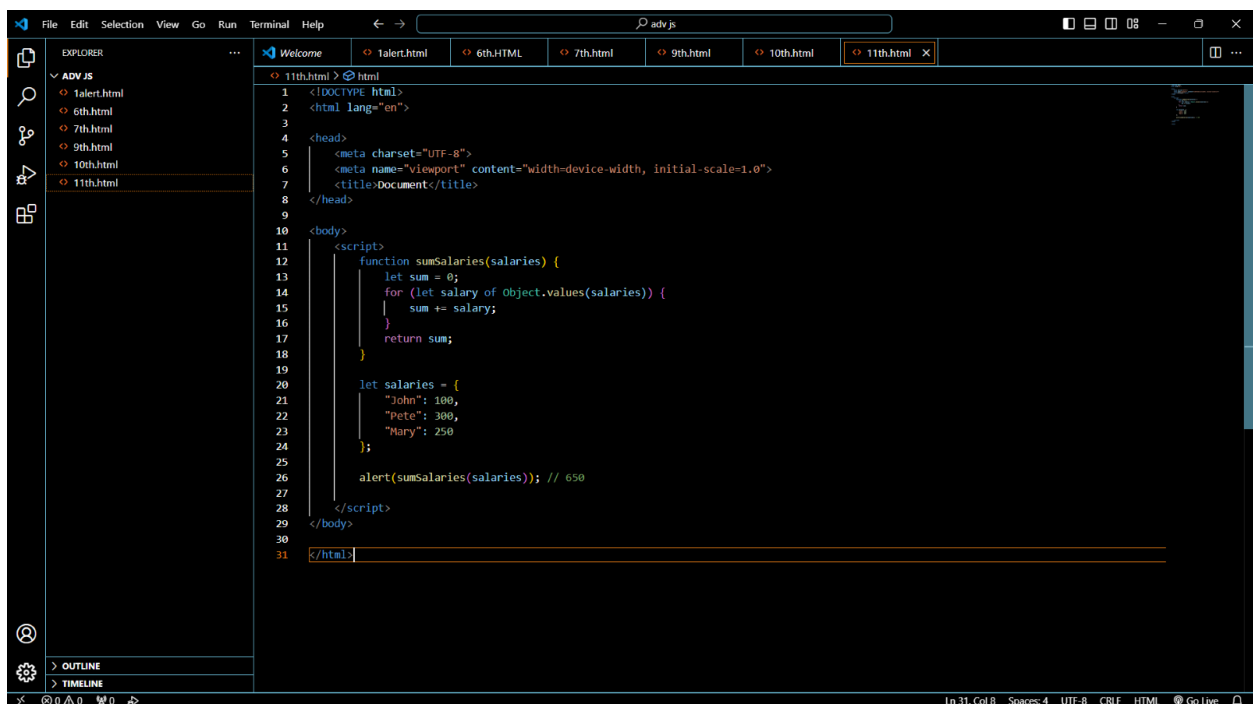


```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <script>
12     let john = { name: "John", surname: "Smith", id: 1 };
13     let pete = { name: "Pete", surname: "Hunt", id: 2 };
14     let mary = { name: "Mary", surname: "Key", id: 3 };
15
16     let users = [john, pete, mary];
17
18     let usersMapped = users.map(user => ({
19       fullName: `${user.name} ${user.surname}`,
20       id: user.id
21     }));
22
23     /*
24     usersMapped = [
25       { fullName: "John Smith", id: 1 },
26       { fullName: "Pete Hunt", id: 2 },
27       { fullName: "Mary Key", id: 3 }
28     ];
29     */
30
31     alert(usersMapped[0].id); // 1
32     alert(usersMapped[0].fullName); // John Smith
33
34   </script>
35 </body>
36
37 </html>
```

*11. Sum the properties There is a
salaries object with arbitrary number of
salaries. Write the function
sumSalaries(salaries) that returns the*

sum of all salaries using Object.values and the for..of loop.If salaries is empty, then the result must be 0. let salaries = { "John": 100, "Pete": 300, "Mary": 250 }; alert(sumSalaries(salaries)); // 650

-->

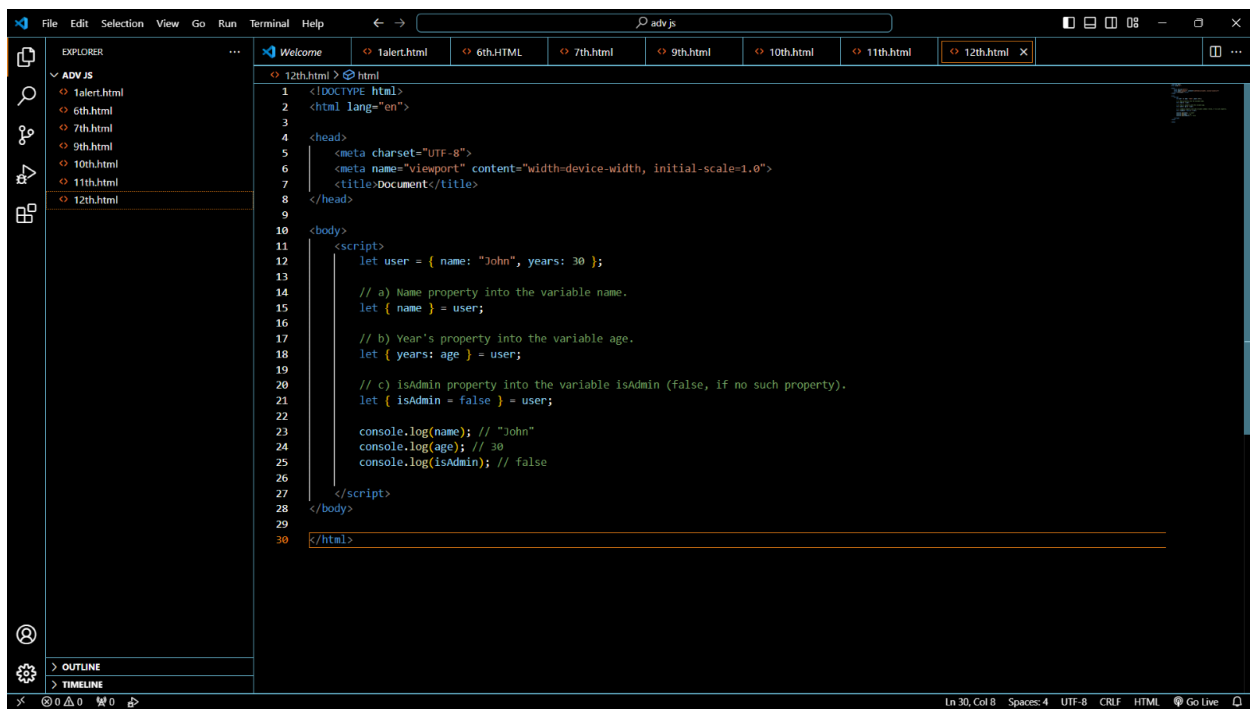


```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <script>
12     function sumSalaries(salaries) {
13       let sum = 0;
14       for (let salary of Object.values(salaries)) {
15         sum += salary;
16       }
17       return sum;
18     }
19
20     let salaries = {
21       "John": 100,
22       "Pete": 300,
23       "Mary": 250
24     };
25
26     alert(sumSalaries(salaries)); // 650
27
28   </script>
29 </body>
30
31 </html>
```

12. Destructuring assignment We have an object: Write the Destructuring

assignment that reads: a) Name property into the variable name. b) Year's property into the variable age. c) isAdmin property into the variable isAdmin (false, if no such property) d) let user = { name: "John", years: 30};

-->



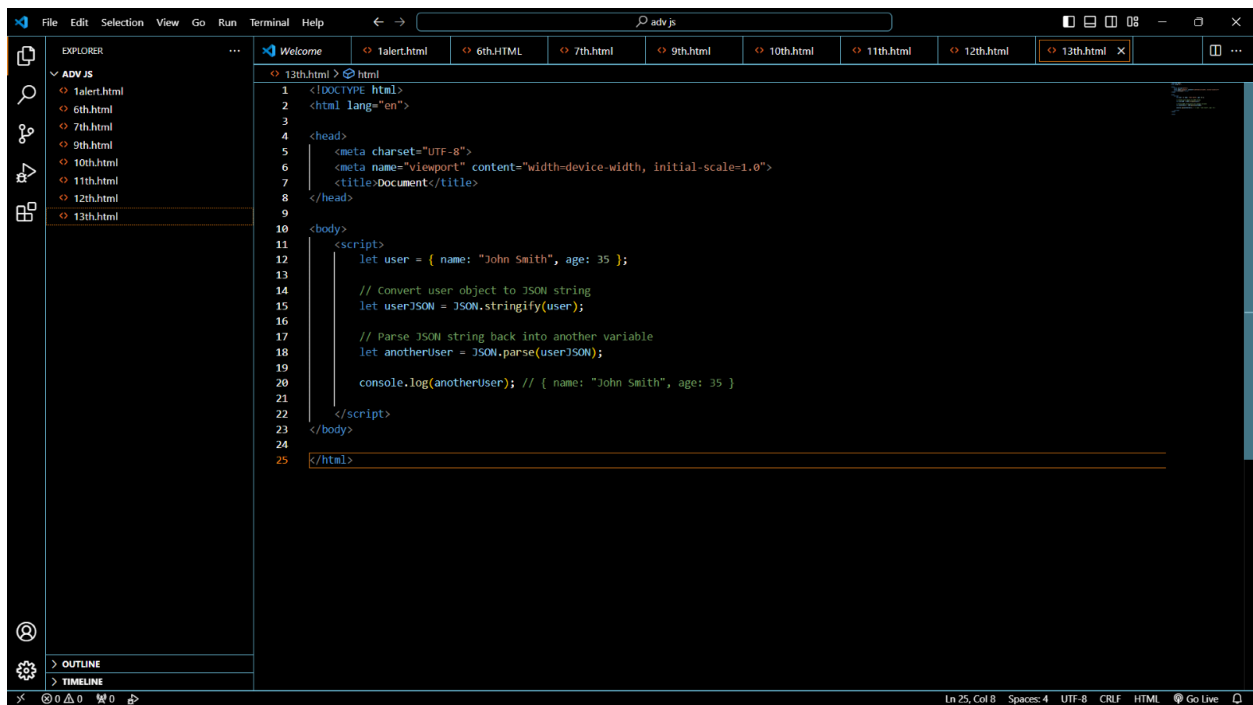
The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left displays a file tree with 'ADV JS' as the root, containing files from '1st.html' to '12th.html'. The main editor area is open to '12th.html', which contains the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <script>
12     let user = { name: "John", years: 30 };
13
14     // a) Name property into the variable name.
15     let { name } = user;
16
17     // b) Year's property into the variable age.
18     let { years: age } = user;
19
20     // c) isAdmin property into the variable isAdmin (false, if no such property).
21     let { isAdmin = false } = user;
22
23     console.log(name); // "John"
24     console.log(age); // 30
25     console.log(isAdmin); // false
26
27   </script>
28 </body>
29
30 </html>
```

The status bar at the bottom indicates 'Ln 30, Col 8', 'Spaces: 4', 'UTF-8', 'CRLF', 'HTML', and 'Go Live'.

13. Turn the object into JSON and back
Turn the user into JSON and then read it
back into another variable. user =
{ name: "John Smith", age: 35};

-->



```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <script>
12     let user = { name: "John Smith", age: 35 };
13
14     // Convert user object to JSON string
15     let userJSON = JSON.stringify(user);
16
17     // Parse JSON string back into another variable
18     let anotherUser = JSON.parse(userJSON);
19
20     console.log(anotherUser); // { name: "John Smith", age: 35 }
21
22   </script>
23 </body>
24
25 </html>
```

Document, Event and Controls

14. Create a program to hide/show the password

-->

File Edit Selection View Go Run ... adv.js

EXPLORER

- ADV.JS
 - 1alert.html
 - 6th.html
 - 7th.html
 - 9th.html
 - 10th.html
 - 11th.html
 - 12th.html
 - 13th.html
 - 14th.html
- OUTLINE
- TIMELINE

14th.html > html > body > script

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Password Field with Show/Hide Functionality</title>
8   <style>
9     .password-container {
10       width: 250px;
11     }
12
13     .password-input {
14       width: 100%;
15       padding: 10px;
16       border: 1px solid #ccc;
17       border-radius: 5px;
18       box-sizing: border-box;
19     }
20
21     .toggle-password-label {
22       display: block;
23       margin-top: 5px;
24     }
25   </style>
26 </head>
27
28 <body>
29
30   <div class="password-container">
31     <input type="password" id="password" class="password-input" placeholder="Enter your password">
32     <label for="togglePassword" class="toggle-password-label">
```

Ln 43, Col 12 Spaces: 4 UTF-8 CRLF HTML Go Live

File Edit Selection View Go Run ... adv.js

EXPLORER

- ADV.JS
 - 1alert.html
 - 6th.html
 - 7th.html
 - 9th.html
 - 10th.html
 - 11th.html
 - 12th.html
 - 13th.html
 - 14th.html
- OUTLINE
- TIMELINE

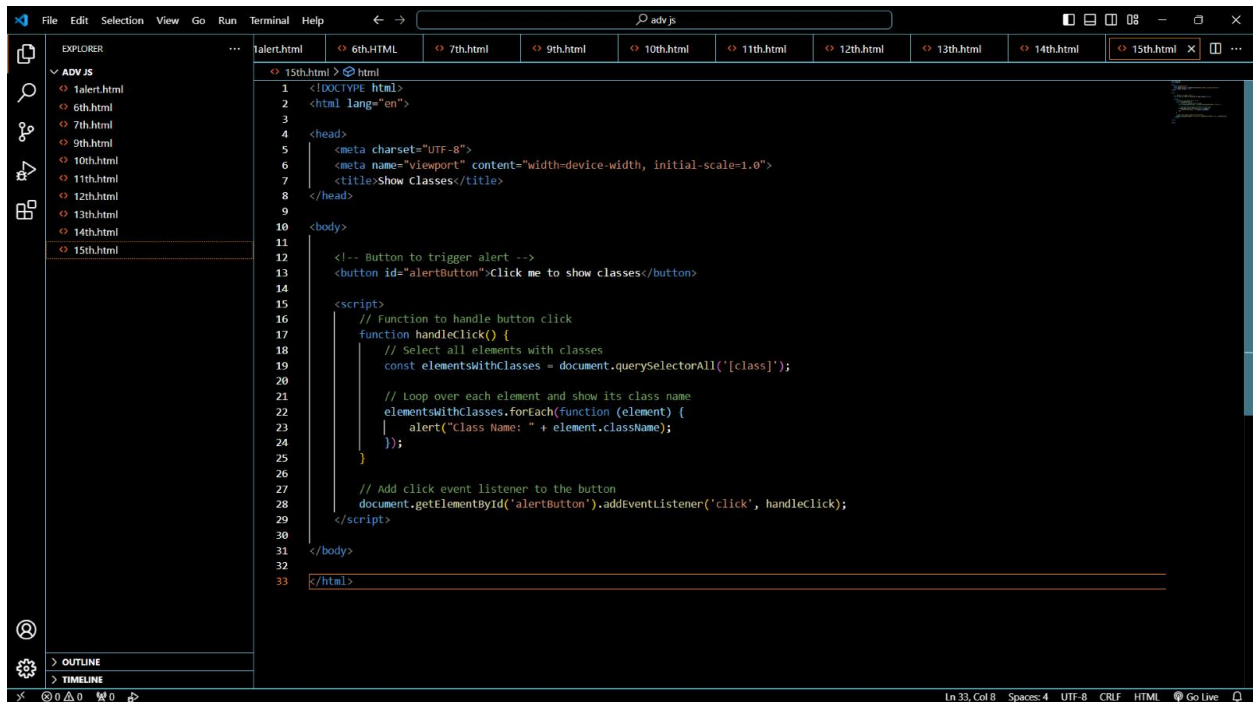
14th.html > html

```
2 <html lang="en">
4 <head>
8   <style>
21     .toggle-password-label {
23       margin-top: 5px;
24     }
25   </style>
26 </head>
27
28 <body>
29
30   <div class="password-container">
31     <input type="password" id="password" class="password-input" placeholder="Enter your password">
32     <label for="togglePassword" class="toggle-password-label">
33       <input type="checkbox" id="togglePassword"> Show Password
34     </label>
35   </div>
36
37   <script>
38     const togglePassword = document.getElementById('togglePassword');
39     const passwordInput = document.getElementById('password');
40
41     togglePassword.addEventListener('change', function () {
42       passwordInput.type = this.checked ? 'text' : 'password';
43     });
44   </script>
45
46 </body>
47
48 </html>
```

Ln 48, Col 8 Spaces: 4 UTF-8 CRLF HTML Go Live

15. Create a program that will select all the classes and loop over and whenever i click the button the alert should show

-->



```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Show Classes</title>
8 </head>
9
10 <body>
11
12   <!-- Button to trigger alert -->
13   <button id="alertButton">Click me to show classes</button>
14
15   <script>
16     // Function to handle button click
17     function handleClick() {
18       // Select all elements with classes
19       const elementsWithClasses = document.querySelectorAll('[class]');
20
21       // Loop over each element and show its class name
22       elementsWithClasses.forEach(function (element) {
23         alert("Class Name: " + element.className);
24       });
25     }
26
27     // Add click event listener to the button
28     document.getElementById('alertButton').addEventListener('click', handleClick);
29   </script>
30
31 </body>
32
33 </html>
```

16. Create a responsive header using proper JavaScript



17. Create a form and validate using JavaScript

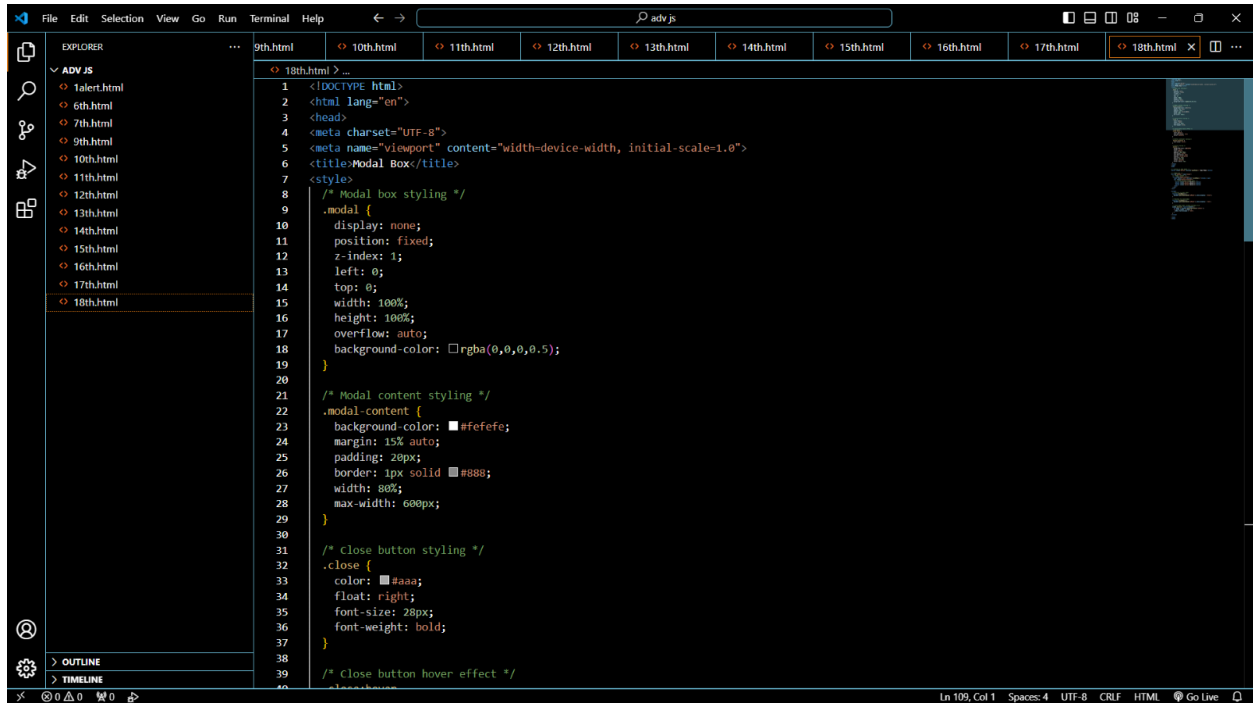


```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Form validation</title>
8   <style>
9     .error {
10       color: red;
11     }
12   </style>
13 </head>
14
15 <body>
16
17   <h2>Registration Form</h2>
18
19   <form id="registrationForm" onsubmit="return validateForm()">
20     <div>
21       <label for="username">Username:</label>
22       <input type="text" id="username" name="username">
23       <span id="usernameError" class="error"></span>
24     </div>
25     <div>
26       <label for="email">Email:</label>
27       <input type="email" id="email" name="email">
28       <span id="emailError" class="error"></span>
29     </div>
30     <div>
31       <label for="password">Password:</label>
32       <input type="password" id="password" name="password">
33       <span id="passwordError" class="error"></span>
34     </div>
35     <div>
36       <input type="submit" value="Register">
37     </div>
38   </form>
39
40 </body>
```

```
2 <html lang="en">
15 <body>
40 <script>
41   function validateForm() {
42     document.getElementById('emailError').textContent = 'Invalid email format';
43     return false;
44   }
45
46   if (password === '') {
47     document.getElementById('passwordError').textContent = 'Password is required';
48     return false;
49   } else if (password.length < 8) {
50     document.getElementById('passwordError').textContent = 'Password must be at least 8 characters long';
51     return false;
52   }
53
54   return true;
55 }
56
57 function isValidEmail(email) {
58   const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
59   return emailRegex.test(email);
60 }
61 </script>
62
63 </body>
64
65 </html>
```

18. Create a modal box using css and Js with three buttons

-->



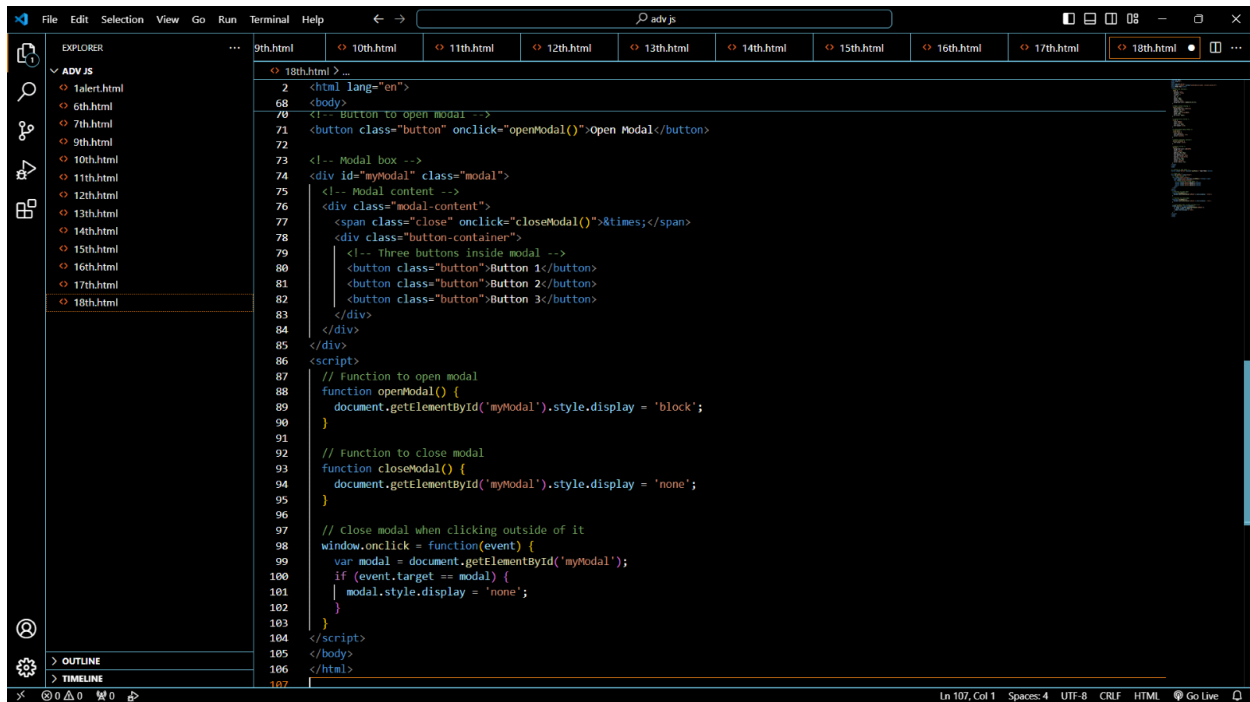
The screenshot shows a VS Code editor with the file explorer on the left displaying a directory named 'ADV JS' containing files from 1st.html to 18th.html. The main editor window is open to 18th.html, showing the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Modal Box</title>
7   <style>
8     /* Modal box styling */
9     .modal {
10       display: none;
11       position: fixed;
12       z-index: 1;
13       left: 0;
14       top: 0;
15       width: 100%;
16       height: 100%;
17       overflow: auto;
18       background-color: rgba(0,0,0,0.5);
19     }
20
21     /* Modal content styling */
22     .modal-content {
23       background-color: #fefefe;
24       margin: 15% auto;
25       padding: 20px;
26       border: 1px solid #888;
27       width: 80%;
28       max-width: 600px;
29     }
30
31     /* Close button styling */
32     .close {
33       color: #aaa;
34       float: right;
35       font-size: 28px;
36       font-weight: bold;
37     }
38
39     /* Close button hover effect */
40     .close:hover {
41       color: black;
42     }
43   </style>
44 </html>
```

The status bar at the bottom indicates the cursor is at line 109, column 1, with 4 spaces, using UTF-8 encoding and CRLF line endings. The 'Go Live' extension is also visible in the bottom right corner.



```
1: 18th.html > ...
2: <html lang='en'>
3: <head>
7: <style>
32: .close {
37: }
38:
39: /* Close button hover effect */
40: .close:hover,
41: .close:focus {
42: color: black;
43: text-decoration: none;
44: cursor: pointer;
45: }
46:
47: /* Button container styling */
48: .button-container {
49: text-align: center;
50: }
51:
52: /* Button styling */
53: .button {
54: background-color: #4CAF50;
55: border: none;
56: color: white;
57: padding: 10px 20px;
58: text-align: center;
59: text-decoration: none;
60: display: inline-block;
61: font-size: 16px;
62: margin: 4px 2px;
63: cursor: pointer;
64: border-radius: 4px;
65: }
66: </style>
67: </head>
68: <body>
69:
70: <!-- Button to open modal -->
71: <button class='button' onclick='openModal()'>Open Modal</button>
```



```
1: 18th.html > ...
2: <html lang='en'>
68: <body>
70: <!-- Button to open modal -->
71: <button class='button' onclick='openModal()'>Open Modal</button>
72:
73: <!-- Modal box -->
74: <div id='myModal' class='modal'>
75: <!-- Modal content -->
76: <div class='modal-content'>
77: <span class='close' onclick='closeModal()'>&times;</span>
78: <div class='button-container'>
79: <!-- Three buttons inside modal -->
80: <button class='button'>Button 1</button>
81: <button class='button'>Button 2</button>
82: <button class='button'>Button 3</button>
83: </div>
84: </div>
85: </div>
86: <script>
87: // Function to open modal
88: function openModal() {
89: document.getElementById('myModal').style.display = 'block';
90: }
91:
92: // Function to close modal
93: function closeModal() {
94: document.getElementById('myModal').style.display = 'none';
95: }
96:
97: // Close modal when clicking outside of it
98: window.onclick = function(event) {
99: var modal = document.getElementById('myModal');
100: if (event.target == modal) {
101: modal.style.display = 'none';
102: }
103: }
104: </script>
105: </body>
106: </html>
107:
```

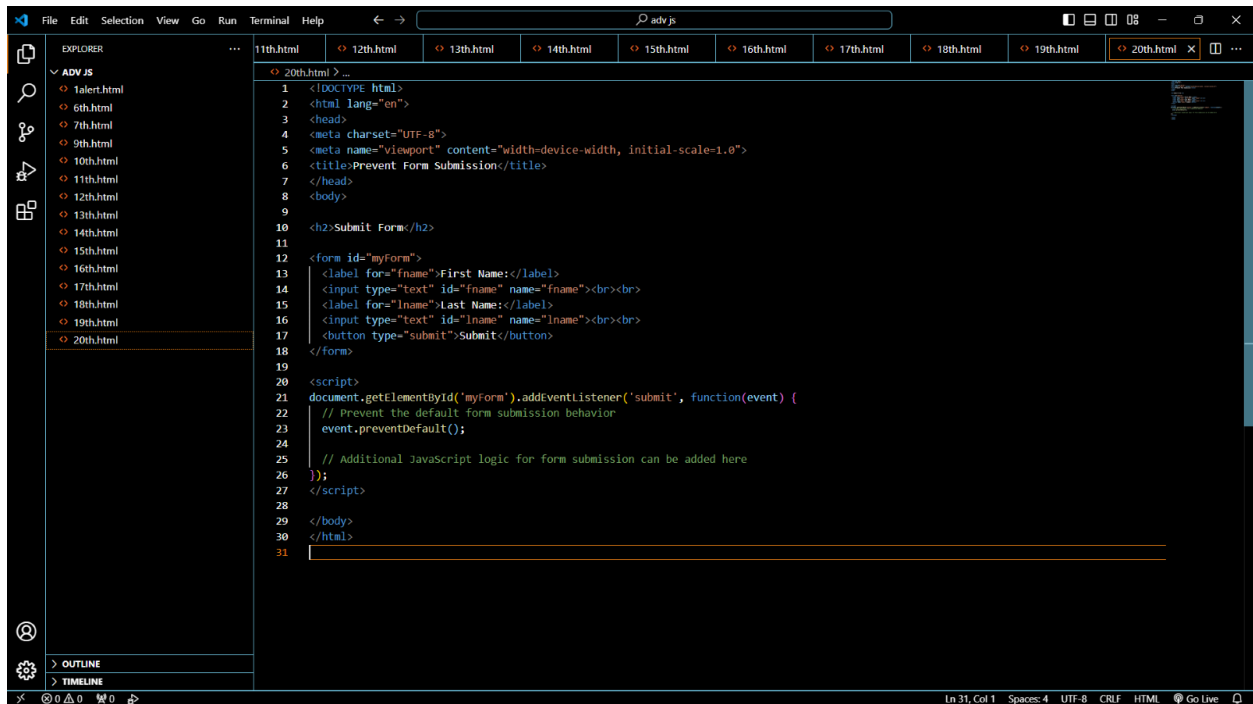
19. Use external js library to show slider



```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Slider </title>
8   <link rel="stylesheet" type="text/css"
9     href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.8.1/slick.min.css" />
10  <link rel="stylesheet" type="text/css"
11    href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.8.1/slick-theme.min.css" />
12 </head>
13
14 <body>
15
16   <div class="slider">
17     <div></div>
18     <div></div>
19     <div></div>
20   </div>
21
22   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
23   <script src="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.8.1/slick.min.js"></script>
24
25   <script>
26     $(document).ready(function () {
27       $('.slider').slick({
28         autoplay: true,
29         autoplaySpeed: 2000,
30         dots: true
31       });
32     });
33   </script>
34
35 </body>
36
37 </html>
```

20. Prevent the browser when i click the form submit button

-->



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Prevent Form Submission</title>
7 </head>
8 <body>
9
10 <h2>Submit Form</h2>
11
12 <form id="myForm">
13   <label for="fname">First Name:</label>
14   <input type="text" id="fname" name="fname"><br><br>
15   <label for="lname">Last Name:</label>
16   <input type="text" id="lname" name="lname"><br><br>
17   <button type="submit">Submit</button>
18 </form>
19
20 <script>
21 document.getElementById('myForm').addEventListener('submit', function(event) {
22   // Prevent the default form submission behavior
23   event.preventDefault();
24
25   // Additional JavaScript logic for form submission can be added here
26 });
27 </script>
28
29 </body>
30 </html>
31
```

New Request

21. What is JSON

--> JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate.

22. What is promises

--> Promises are a fundamental concept in JavaScript used for handling asynchronous operations. They represent a value that may be available now, or in the future, or never. Promises are mainly used for handling asynchronous operations like fetching data from a server, reading files, or executing long-running computations.

23. Write a program of promises and handle that promises also

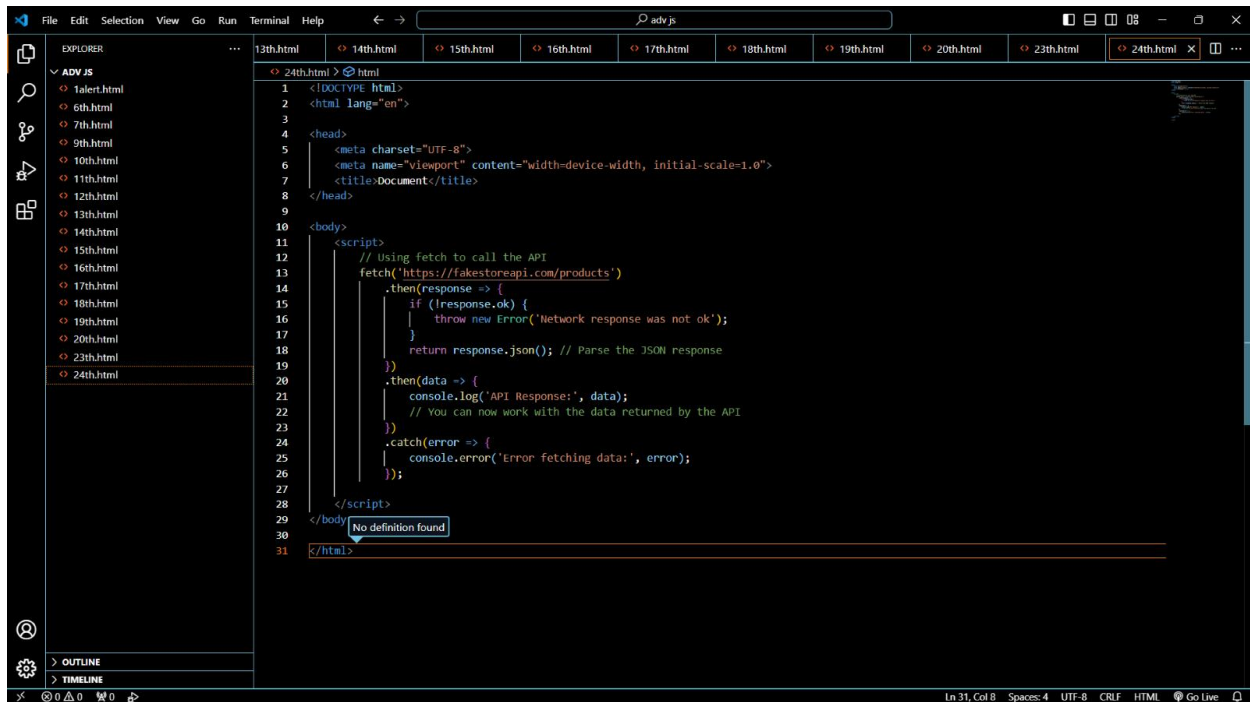


```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10
11   <script>
12     // Function that returns a promise
13     function fetchData() {
14       return new Promise((resolve, reject) => {
15         // Simulate fetching data asynchronously
16         setTimeout(() => {
17           const data = {
18             name: 'John',
19             age: 30
20           };
21           const error = false; // Set to true to simulate an error
22           if (!error) {
23             resolve(data); // Resolve with data
24           } else {
25             reject('Error: Unable to fetch data'); // Reject with error message
26           }
27         }, 1000);
28       });
29     }
30
31     // Using the promise
32     fetchData()
33       .then((data) => {
34         console.log('Data fetched successfully:', data);
35         return data;
36       })
37       .then((data) => {
38         // Perform additional processing on the fetched data
39         console.log('Additional processing:', data);
40       })
41       .catch((error) => {
42         console.error('Failed to fetch data:', error);
43       })
44       .finally(() => {
45         console.log('Promise completed'); // This will be executed regardless of the outcome
46       });
47   </script>
48 </body>
49 </html>
```

24. Use fetch method for calling an api

<https://fakestoreapi.com/products>

-->



```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9
10 <body>
11   <script>
12     // Using fetch to call the API
13     fetch('https://fakestoreapi.com/products')
14       .then(response => {
15         if (!response.ok) {
16           throw new Error('Network response was not ok');
17         }
18         return response.json(); // Parse the JSON response
19       })
20       .then(data => {
21         console.log('API Response:', data);
22         // You can now work with the data returned by the API
23       })
24       .catch(error => {
25         console.error('Error fetching data:', error);
26       });
27   </script>
28 </body>
29 </html>
30
31 </html>
```

JavaScript Essentials

25. What is JavaScript Output method?

--> JavaScript output methods are ways to display information to the user in a web browser. The main JavaScript output methods include:

1. console.log(): Used to send messages or data to the browser's console for debugging and logging purposes.


2. alert(): Displays a simple dialog box with a message to the user, commonly used for notifications or alerts.

3. document.write(): Writes HTML content or JavaScript code directly to the document, but it's rarely used due to potential side effects.

4. innerHTML property: Sets or returns the HTML content of an element on the webpage, allowing dynamic updates of content displayed to the user.

26. How to used JavaScript Output method?

--> JavaScript output methods are used to display information to users in a web browser.



The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'ADV JS' containing files from 14th.html to 26th.html. The code editor is open to 26th.html, which contains the following HTML code:

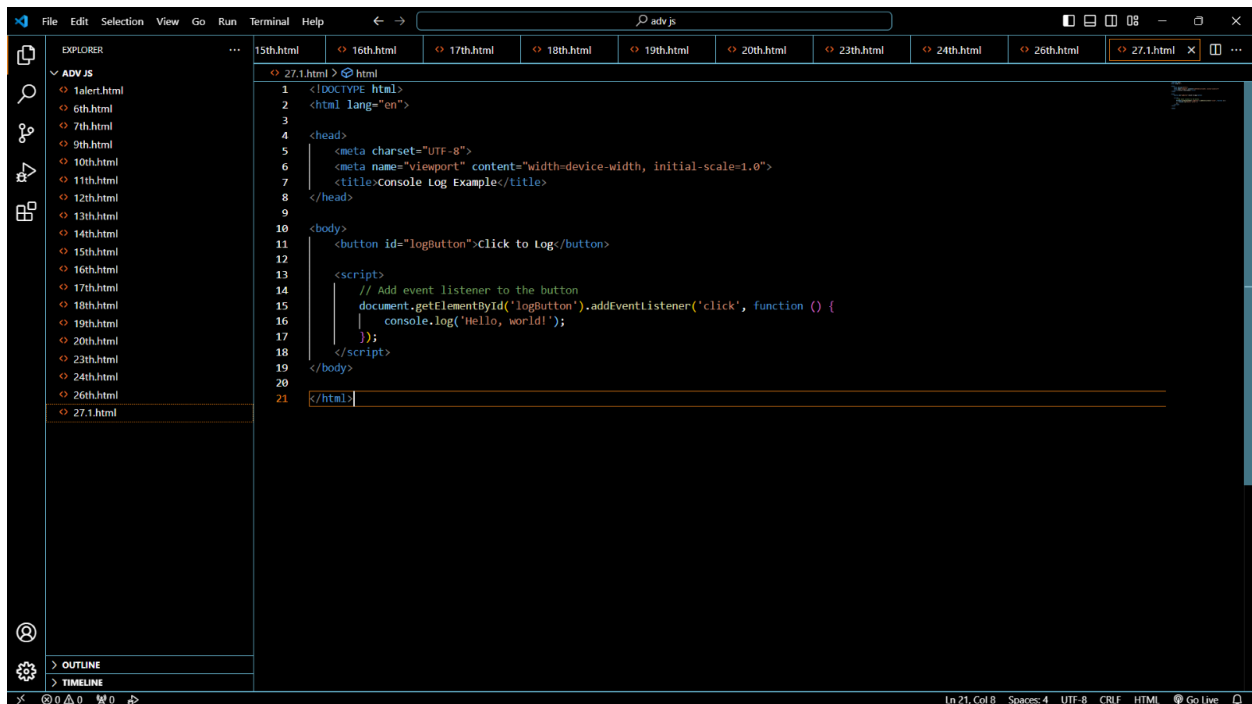
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9   <script>
10    // console.log()
11    console.log('Hello, world!');
12
13    // alert()
14    alert('Hello, world!');
15
16    // document.write()
17    document.write('Hello, world!');
18
19    // innerHTML property
20    document.getElementById('output').innerHTML = 'Hello, world!';
21
22  </script>
23 </body>
24 </html>
```

27. How to use JavaScript Events to do all examples?

-->

JavaScript events allow you to trigger actions based on user interactions or other occurrences in the browser.

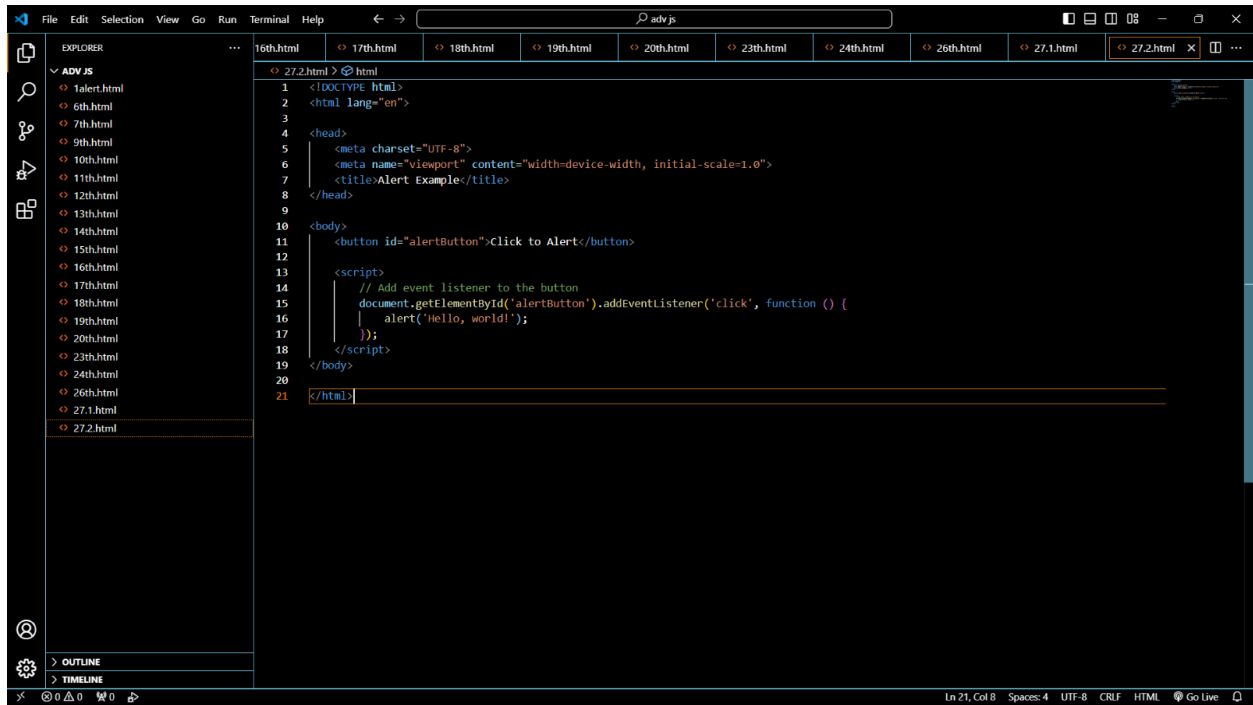
1. Console Log Using Event Listener:



The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left lists a directory named 'ADV JS' containing files from 1st.html to 27.1.html. The main editor area displays the content of 27.1.html, which is an HTML document. The code includes a basic HTML structure with a head section containing meta tags for charset and viewport, and a title 'Console Log Example'. The body section contains a button with the text 'Click to Log' and an ID of 'logButton'. A JavaScript script is embedded in the body, which uses the addEventListener method to attach a click event listener to the button. The listener function logs the message 'Hello, world!' to the console. The status bar at the bottom indicates the cursor is at line 21, column 8, with 4 spaces, in UTF-8 encoding, CRLF line endings, and HTML mode.

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Console Log Example</title>
8 </head>
9
10 <body>
11   <button id="logButton">Click to Log</button>
12
13   <script>
14     // Add event listener to the button
15     document.getElementById('logButton').addEventListener('click', function () {
16       console.log('Hello, world!');
17     });
18   </script>
19 </body>
20
21 </html>
```

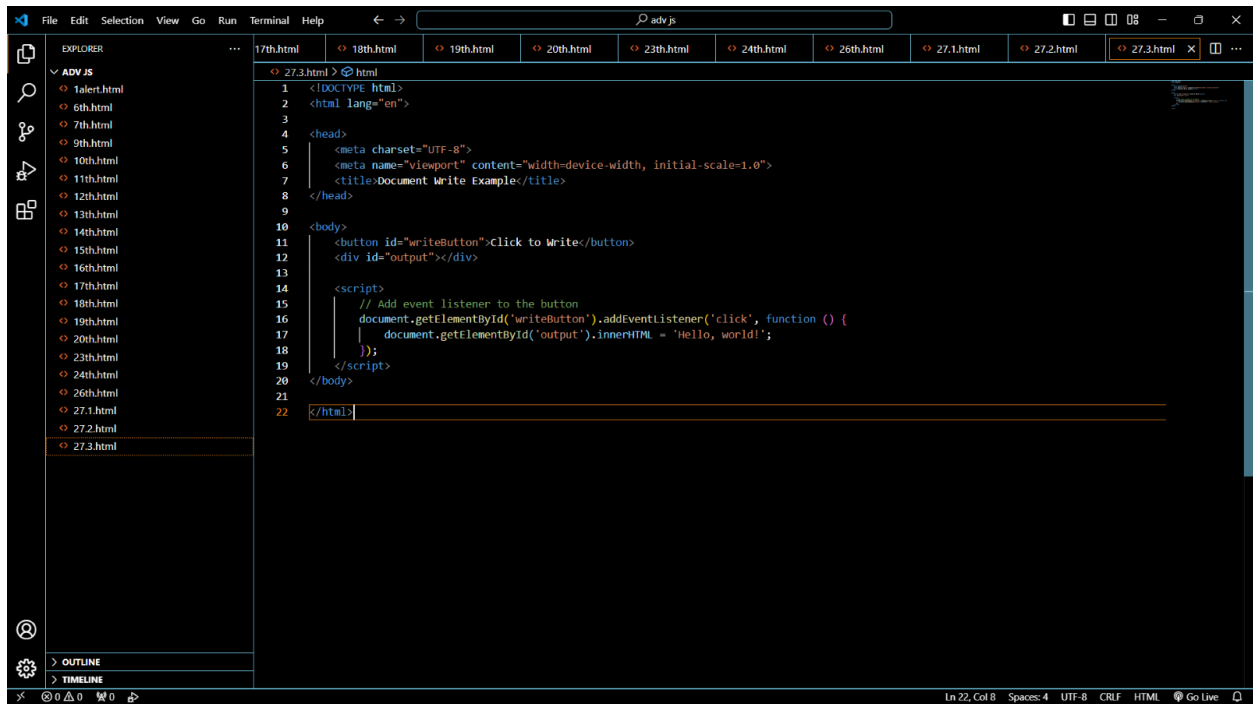
2. Alert Using Event Listener:



The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left lists files under 'ADV JS', including 16th.html through 27.2.html. The main editor area displays the content of 27.2.html, which is an HTML document. The code includes a DOCTYPE declaration, a meta charset of UTF-8, a viewport meta tag, and a title 'Alert Example'. In the body, there is a button with the text 'Click to Alert' and an ID of 'alertButton'. A JavaScript event listener is attached to this button, which calls the 'alert' function with the message 'Hello, world!' when clicked. The status bar at the bottom indicates the cursor is at line 21, column 8.

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Alert Example</title>
8 </head>
9
10 <body>
11   <button id="alertButton">Click to Alert</button>
12
13   <script>
14     // Add event listener to the button
15     document.getElementById('alertButton').addEventListener('click', function () {
16       alert('Hello, world!');
17     });
18   </script>
19 </body>
20
21 </html>
```

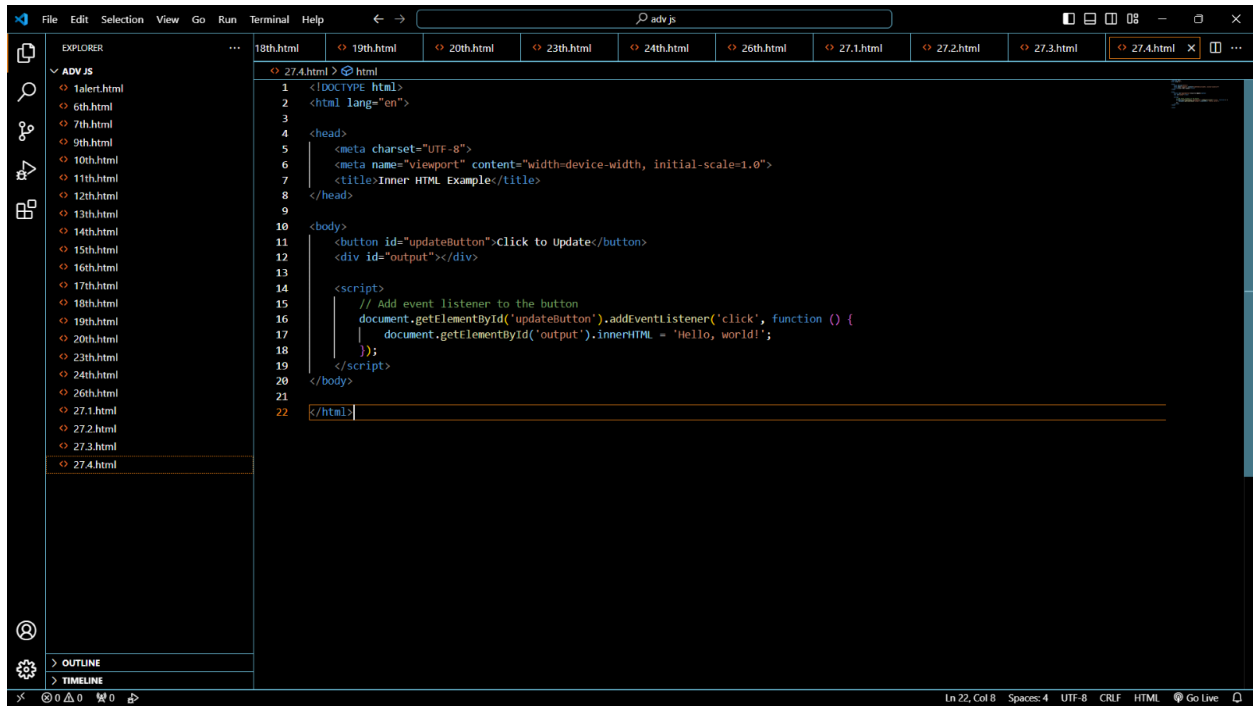
3. Document Write Using Event Listener:



The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left lists files under 'ADV JS', including 16th.html through 27.3.html. The main editor area displays the content of 27.3.html, which is an HTML document. The code includes a DOCTYPE declaration, a meta charset of UTF-8, a viewport meta tag, and a title 'Document Write Example'. In the body, there is a button with the text 'Click to Write' and an ID of 'writeButton', followed by a div with the ID 'output'. A JavaScript event listener is attached to the button, which calls the 'innerHTML' property of the 'output' div and sets it to 'Hello, world!' when clicked. The status bar at the bottom indicates the cursor is at line 22, column 8.

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document Write Example</title>
8 </head>
9
10 <body>
11   <button id="writeButton">Click to Write</button>
12   <div id="output"></div>
13
14   <script>
15     // Add event listener to the button
16     document.getElementById('writeButton').addEventListener('click', function () {
17       document.getElementById('output').innerHTML = 'Hello, world!';
18     });
19   </script>
20 </body>
21
22 </html>
```

4. Inner HTML Using Event Listener:



The screenshot shows the Visual Studio Code editor interface. The Explorer panel on the left displays a file tree with a folder named 'ADV JS' containing files from 1st.html to 27.4.html. The main editor area is open to 27.4.html, which contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Inner HTML Example</title>
8 </head>
9
10 <body>
11   <button id="updateButton">Click to Update</button>
12   <div id="output"></div>
13
14   <script>
15     // Add event listener to the button
16     document.getElementById('updateButton').addEventListener('click', function () {
17       document.getElementById('output').innerHTML = 'Hello, world!';
18     });
19   </script>
20 </body>
21
22 </html>
```

The status bar at the bottom indicates the current position is Line 22, Column 8, with 4 spaces, UTF-8 encoding, CRLF line endings, and the HTML file type.