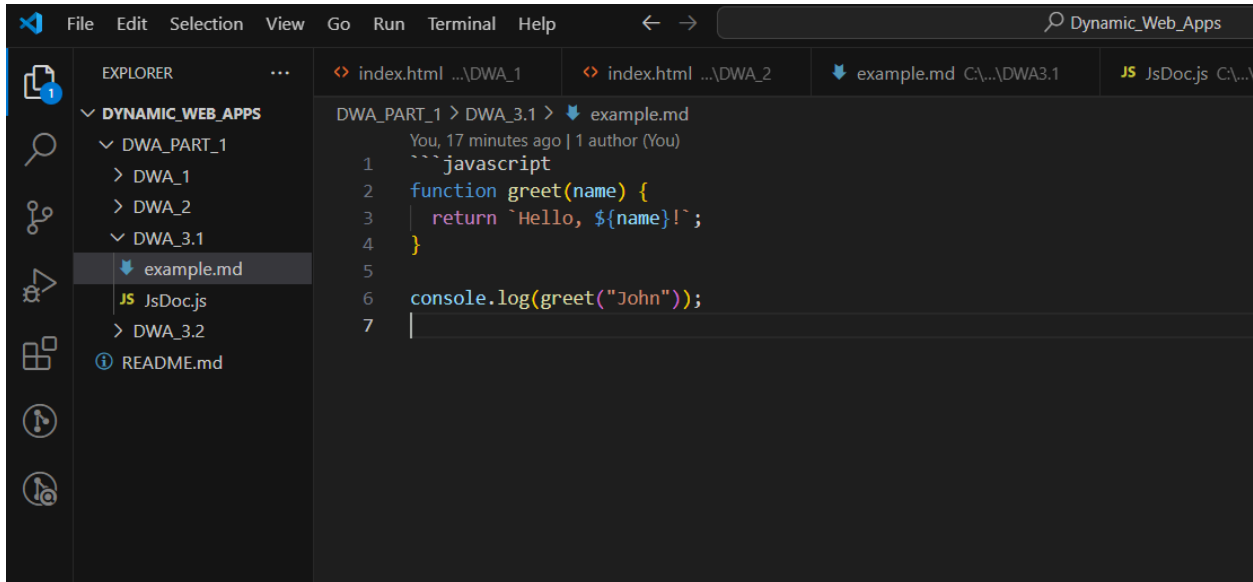


DWA_03.4 Knowledge Check_DWA3.1

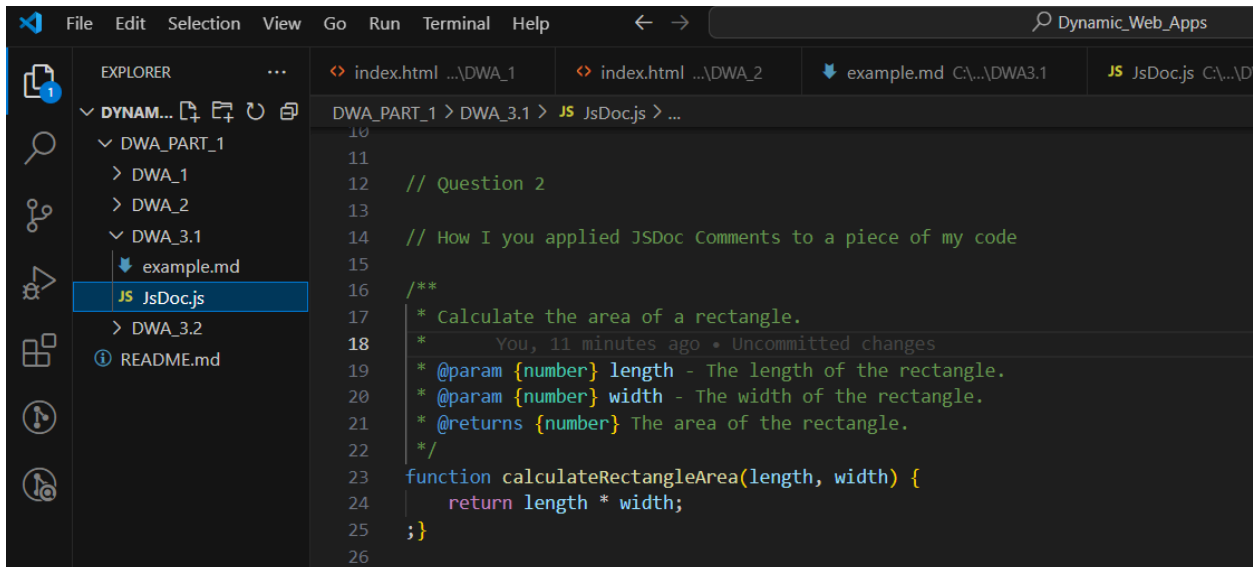
1. Please show how you applied a Markdown File to a piece of your code.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'DYNAMIC_WEB_APPS' with subfolders 'DWA_PART_1', 'DWA_2', and 'DWA_3.1'. Inside 'DWA_3.1', there is a file 'example.md' and a file 'JsDoc.js'. The main editor area shows the content of 'example.md'. It starts with a header 'DWA_PART_1 > DWA_3.1 > example.md' and a timestamp 'You, 17 minutes ago | 1 author (You)'. Below this is a JavaScript code block defined by three backticks. The code inside the block is:

```
1  ```javascript
2  function greet(name) {
3    return `Hello, ${name}!`;
4  }
5
6  console.log(greet("John"));
7  
```

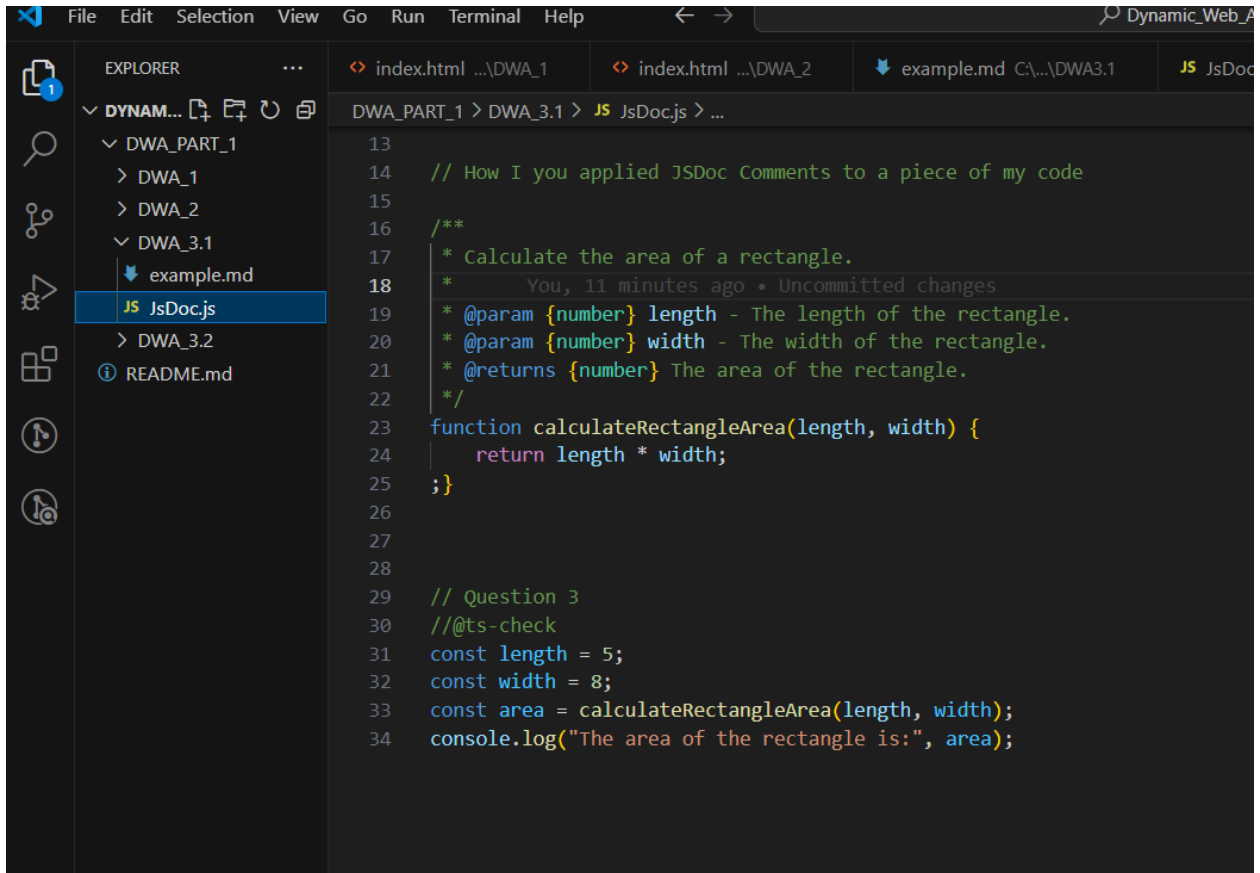
2. Please show how you applied JSDoc Comments to a piece of your code.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows the same project structure as before. The main editor area shows the content of 'JsDoc.js'. It starts with a header 'DWA_PART_1 > DWA_3.1 > JS JsDoc.js > ...' and a timestamp 'You, 11 minutes ago * Uncommitted changes'. Below this is a JSDoc comment block for a function 'calculateRectangleArea'. The comment is enclosed in three asterisks and includes a description, parameters, and a return value. The code inside the function is:

```
10
11
12  // Question 2
13
14  // How I you applied JSDoc Comments to a piece of my code
15
16  /**
17   * Calculate the area of a rectangle.
18   * You, 11 minutes ago * Uncommitted changes
19   * @param {number} length - The length of the rectangle.
20   * @param {number} width - The width of the rectangle.
21   * @returns {number} The area of the rectangle.
22   */
23  function calculateRectangleArea(length, width) {
24    return length * width;
25  };
26
```

3. Please show how you applied the @ts-check annotation to a piece of your code.



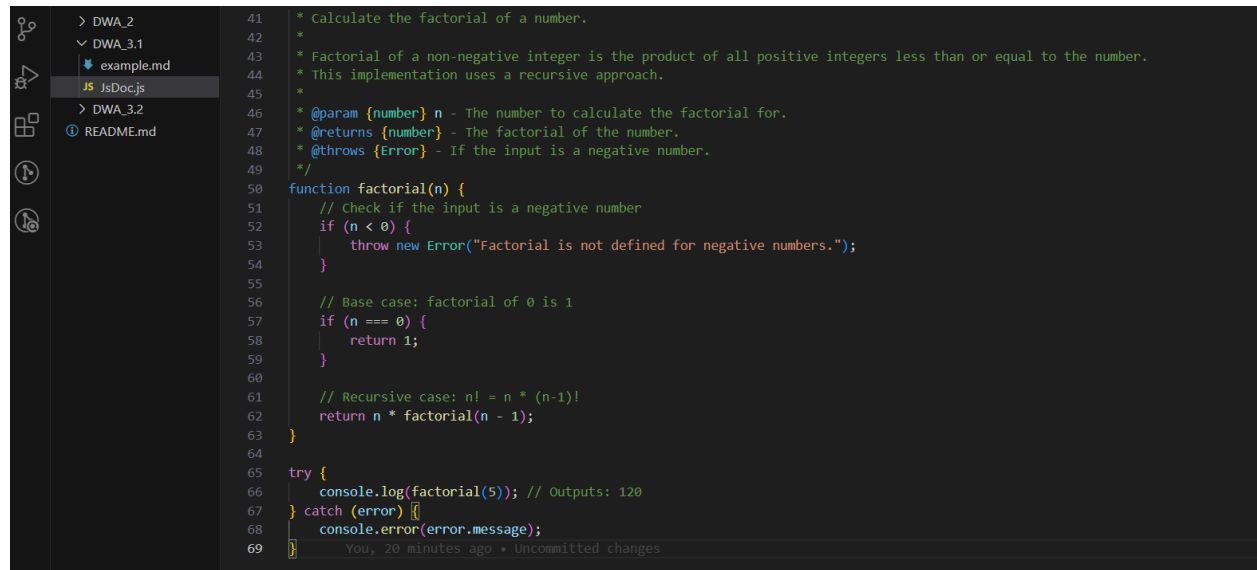
The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left displays a project structure with folders 'DWA_PART_1', 'DWA_2', and 'DWA_3.1'. Inside 'DWA_3.1', there is a file 'example.md' and a file 'JsDoc.js' which is currently selected. The main editor window shows the content of 'JsDoc.js'. The code includes JSDoc comments for a function 'calculateRectangleArea' and a variable declaration. The variable declaration is annotated with '@ts-check'.

```
13
14 // How I you applied JSDoc Comments to a piece of my code
15
16 /**
17  * Calculate the area of a rectangle.
18  * You, 11 minutes ago * Uncommitted changes
19  * @param {number} length - The length of the rectangle.
20  * @param {number} width - The width of the rectangle.
21  * @returns {number} The area of the rectangle.
22  */
23 function calculateRectangleArea(length, width) {
24     return length * width;
25 }
26
27
28
29 // Question 3
30 // @ts-check
31 const length = 5;
32 const width = 8;
33 const area = calculateRectangleArea(length, width);
34 console.log("The area of the rectangle is:", area);
```

4. As a BONUS, please show how you applied any other concept covered in the 'Documentation' module.

I will show In line Comments:

They explain specific parts of the factorial function, such as the base case and recursive case. And they also help clarify the logic and purpose of the code.



```
41  * Calculate the factorial of a number.
42  *
43  * Factorial of a non-negative integer is the product of all positive integers less than or equal to the number.
44  * This implementation uses a recursive approach.
45  *
46  * @param {number} n - The number to calculate the factorial for.
47  * @returns {number} - The factorial of the number.
48  * @throws {Error} - If the input is a negative number.
49  */
50  function factorial(n) {
51      // Check if the input is a negative number
52      if (n < 0) {
53          throw new Error("Factorial is not defined for negative numbers.");
54      }
55
56      // Base case: factorial of 0 is 1
57      if (n === 0) {
58          return 1;
59      }
60
61      // Recursive case: n! = n * (n-1)!
62      return n * factorial(n - 1);
63  }
64
65  try {
66      console.log(factorial(5)); // Outputs: 120
67  } catch (error) {
68      console.error(error.message);
69  }
```

You, 20 minutes ago • Uncommitted changes