DWA_03.4 Knowledge Check_DWA3.1

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

To apply a Markdown file to a piece of JavaScript code, you can use a JavaScript library or module that parses Markdown and converts it to HTML. Here's an example using the "markdown-it" library:

1. Install the "markdown-it" library by running the following command in your project's directory using a package manager like npm or yarn:

[npm install markdown-it]

2. Create a JavaScript file (e.g., markdownExample.js) and import the "markdown-it" library:

[const markdown = require('markdown-it')();]

3. Read the Markdown file using a file system module, such as the built-in ` fs ` module in Node.js:

const fs = require('fs');

const markdownFile = fs.readFileSync('path/to/your/markdownFile.md', 'utf-8');

4. Convert the Markdown content to HTML using the markdown-it library:

const htmlContent = markdown.render(markdownFile);

5. You can now use the generated HTML in your JavaScript code as needed. For example, if you're working with a web application, you can inject the HTML into a specific element on your webpage:

const targetElement = document.getElementById('targetElementId');

targetElement.innerHTML = htmlContent;

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

To apply JSDoc comments to a piece of JavaScript code, you can follow these steps:

Identify the JavaScript code or function you want to document. JSDoc comments are typically placed directly above the code or function they are documenting.

Start the JSDoc comment with /** (two asterisks) to indicate the beginning of the comment block.

Provide a description of the code or function using plain text within the comment block. This description should explain the purpose, functionality, and any important details of the code or function.

Use JSDoc tags to add structured information to your comment. Tags start with the @ symbol, followed by the name of the tag and its value (if applicable). Some commonly used JSDoc tags include:

@param to describe the parameters of a function, including their types and descriptions.

@returns to describe the return value of a function.

@throws to document the possible exceptions or errors that a function may throw.

@example to provide example usage of the code or function.

Close the JSDoc comment with */ (asterisk and forward slash) to indicate the end of the comment block.

Here's an example that demonstrates the application of JSDoc comments to a JavaScript function:

```
/**
 * Calculates the sum of two numbers.
 *
 * @param {number} a - The first number.
 * @param {number} b - The second number.
 * @returns {number} The sum of the two numbers.
 * @throws {Error} If either of the parameters is not a number.
 *
 * @example
 * // Returns 5
 * const result = calculateSum(2, 3);
 */
function calculateSum(a, b) {
  if (typeof a !== 'number' || typeof b !== 'number') {
    throw new Error('Both parameters must be numbers.');
  }
  return a + b;
}
```

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

To apply the @ts-check annotation to a JavaScript file , you can follow these steps:

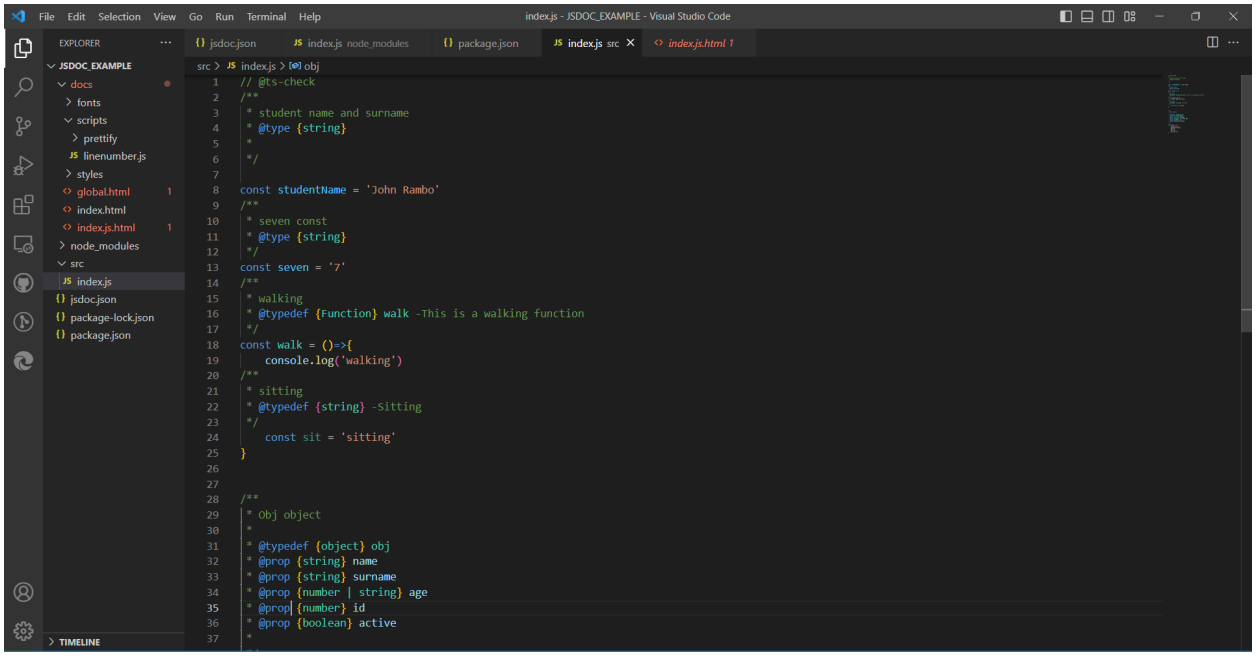
1. Open your JavaScript file in a code editor like Visual Studio Code.

2. Add the @ts-check comment at the top of your JavaScript file. This comment tells the editor to enable type-checking for the JavaScript code.

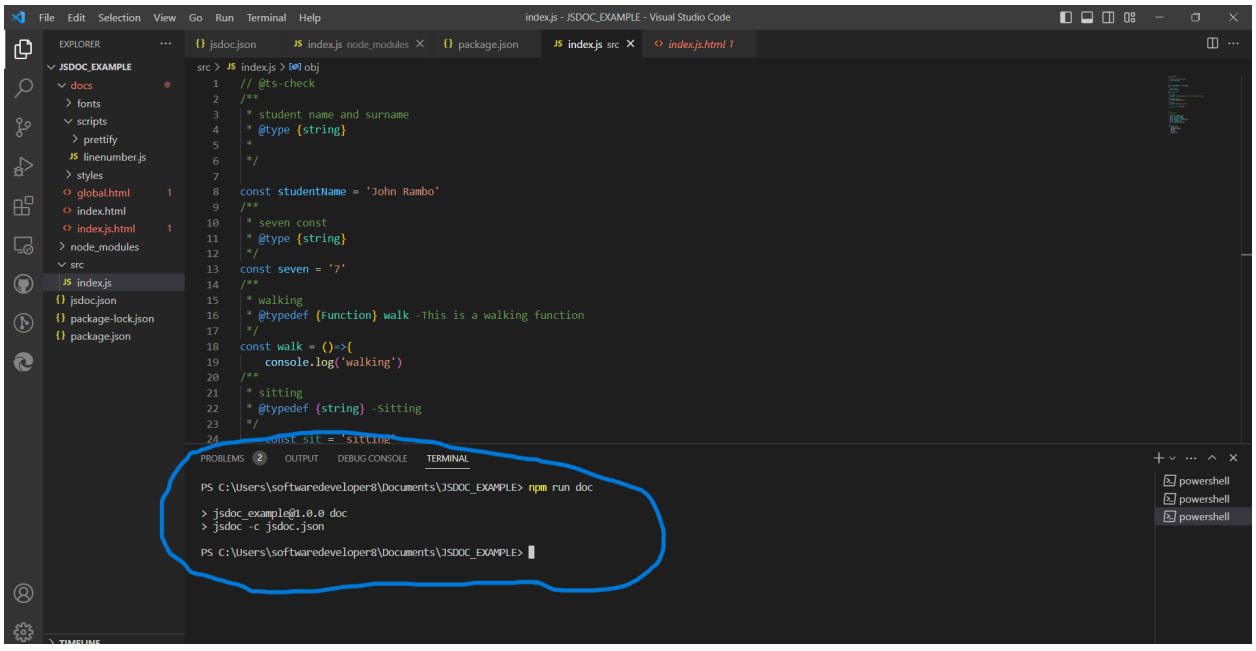
Here is an Example :

```
// @ts-check

/**
 * Adds two numbers together.
 *
 * @param {number} a - The first number.
 * @param {number} b - The second number.
 * @returns {number} The sum of the two numbers.
 */
function addNumbers(a, b) {
  return a + b;
}

// Example usage
const result = addNumbers(2, '3'); // Error: Argument of type '"3"' is not a
console.log(result);
```

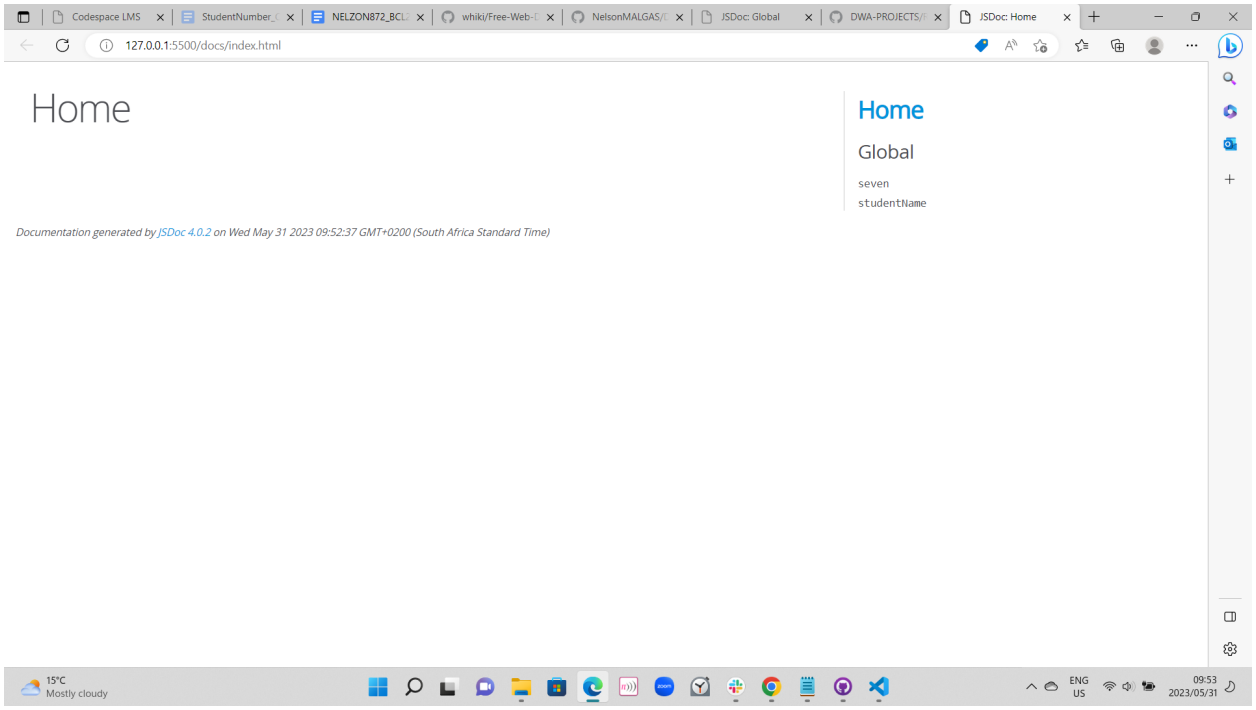
4. As a BONUS, please show how you applied any other concept covered in the 'Documentation' module.



In the above code snippet i was doing a step by step coding along with the JSDoc crash course from the LMS where i created the studentName constant and other variable



In the terminal is where i run the code (the circled section)



The above is the document after running the code snippets