DWA_03.4 Knowledge Check_DWA3.1

---

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

With the flexible markup language known as Markdown, you may format text, add photos, make tables, and more. It is frequently used for online forums, README files, and documentation.

```
// Markdown Example

// @ts-check

/**
 * Generates a random number between a specified minimum and maximum value.
 * @param {number} min - The minimum value for the random number (inclusive).
 * @param {number} max - The maximum value for the random number (inclusive).
 * @returns {number} The generated random number.
 */
function generateRandomNumber(min, max) {
  // Generate a random number between min and max
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

// Usage example
const randomNumber = generateRandomNumber(1, 10);
console.log(randomNumber); // Output: Random number between 1 and 10
```

---

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

JSDoc comments are used to document the `generateRandomNumber` function. Below is an explanation of how JSDoc is applied:

1. The opening `/**` indicates the start of a JSDoc comment block.
2. `* Generates a random number between a specified minimum and maximum value.` briefly describes the function's purpose.
3. `* @param {number} min - The minimum value for the random number (inclusive).` describes the `min` parameter of the function. The `{number}` notation specifies the parameter type.
4. `* @param {number} max - The maximum value for the random number (inclusive).` describes the `max` parameter of the function, following the same pattern as the previous parameter.
5. `* @returns {number} The generated random number.` documents the return value of the function. The `{number}` notation indicates the return type.
6. The closing `*/` denotes the end of the JSDoc comment block.

Therefore, JSDoc comments are done in this manner, the code provides clear documentation about the function's purpose, parameters, and return values. This documentation can be used by developers, IDEs, and tools to understand the code's functionality and assist in code analysis and development.

---

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

In this code, the @ts-check annotation is introduced at the beginning of the file. The JavaScript code will undergo a static type check as a result of this annotation. During development, TypeScript evaluates the code for type problems and offers feedback when @ts-check is enabled. It offers improved tools support and aids in the detection of potential type-related problems in JavaScript development. The @ts-check annotation enables you to use TypeScript's static type-checking features even in JavaScript files, enhancing code quality and spotting mistakes early in the development cycle.

---

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

```
// @ts-check

/**
 * Generates a random number between a specified minimum and maximum value.
 * @param {number} min - The minimum value for the random number (inclusive).
 * @param {number} max - The maximum value for the random number (inclusive).
 * @returns {number} The generated random number.
 */
function generateRandomNumber(min, max) {
  // Generate a random number between min and max
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

// Usage example
const randomNumber = generateRandomNumber(1, 10);

// Output the generated random number
console.log(randomNumber);

// Explanation of the output
console.log("An above number is a random number between 1 and 10 (inclusive).");
```

We have added inline comments to this revised code to explain certain sections and provide more explanations:

- The output is explained in the note that appears above the console.log(randomNumber) command, which states that it shows the generated random number.
- The aim of the output statement is further explained by the comment that comes after the console.log("An above number is a random number between 1 and 10 (inclusive)") statement.

We may highlight crucial information, provide more thorough explanations, and enhance the readability of the code by strategically using comments. Future developers who read or work with the code can benefit from these comments as they will assist them understand the functionality and behavior of the code.
