DWA_03.4 Knowledge Check_DWA3.1

---

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

Markdown is a lightweight markup language that you can use to add formatting elements to plaintext text documents.

For example, to denote a heading, you add a number sign before it (e.g., `# Heading One`). Or to make a phrase bold, you add two asterisks before and after it (e.g., `**this text is bold**`). It may take a while to get used to seeing Markdown syntax in your text, especially if you're accustomed to WYSIWYG applications

---

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

JSDoc's purpose is to document the API of your JavaScript application or library. It is assumed that you will want to document things like modules, namespaces, classes, methods, method parameters, and so on.

JSDoc comments should generally be placed immediately before the code being documented. Each comment must start with a `/**` sequence in order to be recognized by the JSDoc parser. Comments beginning with `/*`, `/***`, or more than 3 stars will be ignored. This is a feature to allow you to suppress parsing of comment blocks.

---

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

The last line of the previous code sample would raise an error in TypeScript, but it doesn't by default in a JS project. To enable errors in your JavaScript files add: `// @ts-check` to the first line in your `.js` files to have TypeScript raise it as an error.

Unset

```
// @ts-check
/** @type {number} */
var x;

x = 0; // OK
x = false; // Not OK
```

---

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

The `throw` statement allows you to create a custom error.

```
throw "Too big";      // throw a text
throw 500;            // throw a number
```
