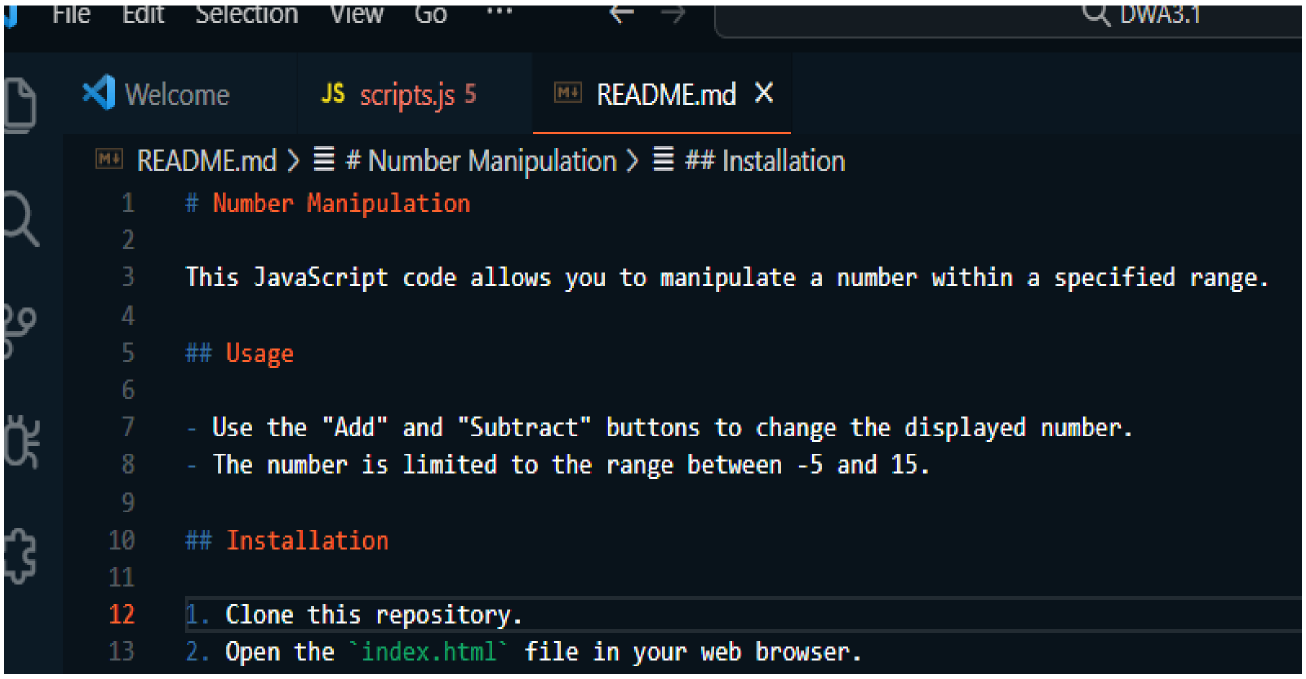
DWA_03.4 Knowledge Check_DWA3.1

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

In this project, I created a document using Markdown to explain what the code does.

- I did this because Markdown makes it easy to format text and create a user-friendly guide.
- It helps people understand how to use the code and set it up.
- By using Markdown, I made sure that users can quickly grasp the purpose of the code and how to get started with it.


File Edit Selection View Go ···
DWA3.1
Welcome
scripts.js 5
README.md
README.md > # Number Manipulation > ## Installation

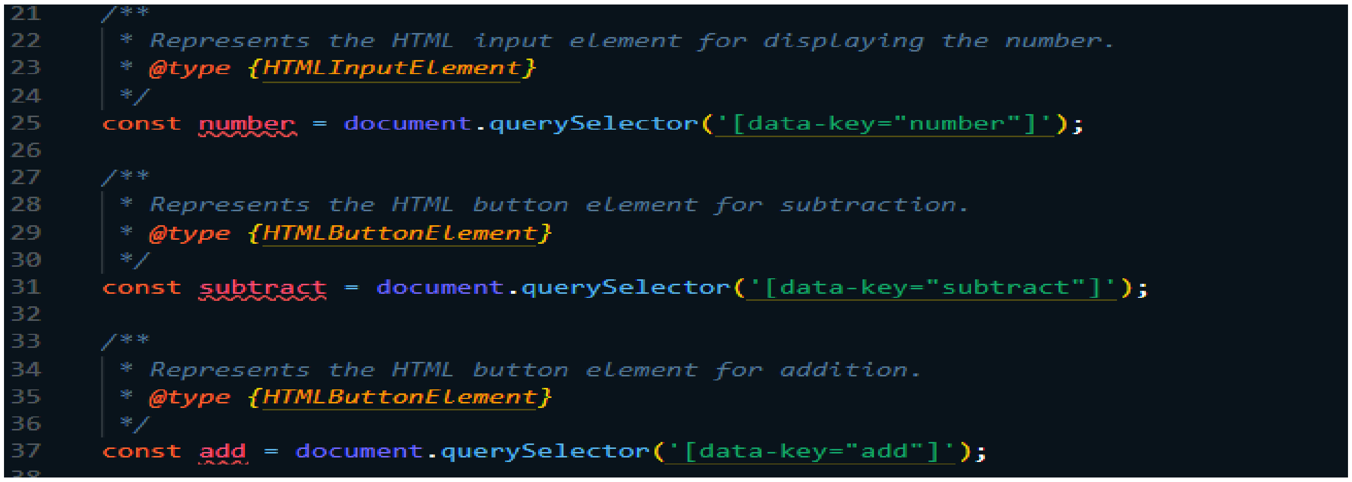
```
# Number Manipulation

This JavaScript code allows you to manipulate a number within a specified range.

## Usage

- Use the "Add" and "Subtract" buttons to change the displayed number.
- The number is limited to the range between -5 and 15.

## Installation

1. Clone this repository.
2. Open the `index.html` file in your web browser.
```

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

I added comments in the JavaScript code to explain what different parts of the code do.

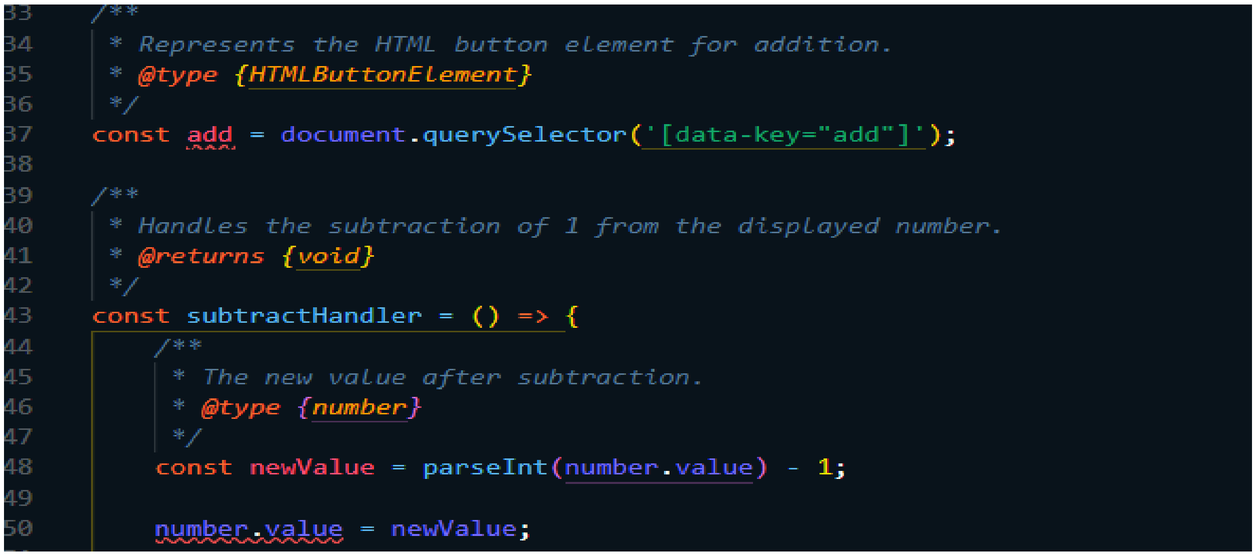
- These comments are like notes to help other developers (and myself) understand how functions and variables work.
- For example, I used comments to say whether a function returns something or not.
- These comments are essential because they make the code easier to understand and reduce the chances of mistakes when changing the code.

These JSDoc comments are used to document the purpose and type of the **MAX_NUMBER** and **MIN_NUMBER** variables. The ***@type*** symbol indicates the expected data type, which, in this case, i***s number***. These comments are important because they provide clear information about the variables' intended use and data type, helping developers avoid unintentional data type errors when working with these constants. They also serve as documentation for future developers who may encounter these variables, making it easier for them to understand their significance and constraints.



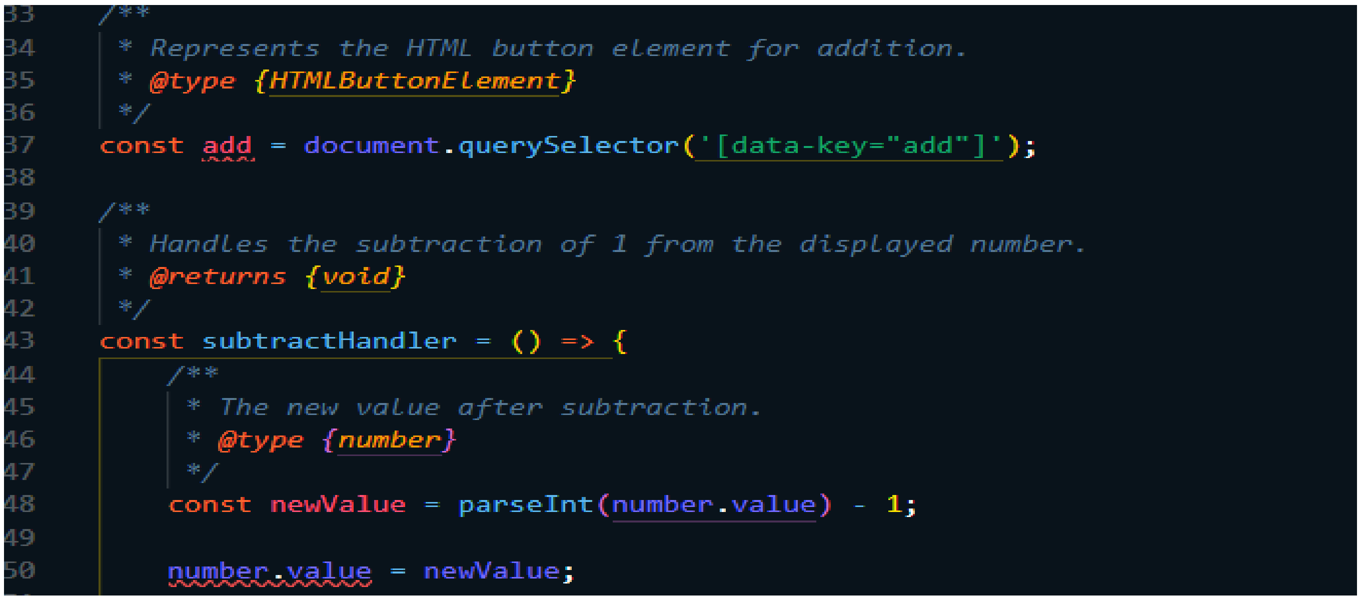
```
/**
 * Represents the HTML input element for displaying the number.
 * @type {HTMLInputElement}
 */
const number = document.querySelector('[data-key="number"]');

/**
 * Represents the HTML button element for subtraction.
 * @type {HTMLButtonElement}
 */
const subtract = document.querySelector('[data-key="subtract"]');

/**
 * Represents the HTML button element for addition.
 * @type {HTMLButtonElement}
 */
const add = document.querySelector('[data-key="add"]');
```

These JSDoc comments document the purpose of the ***number***, ***subtract***, and ***add*** variables and specify their expected ***types***, which are HTML input and button elements in this case. The **@type** symbol is used to indicate the expected data type. These comments are essential because they provide information about the variables' roles in the DOM and their expected types, making it easier for developers to work with these elements in the code. They also assist code editors and tools in providing accurate suggestions and type checking.



```
/**
 * Represents the HTML button element for addition.
 * @type {HTMLButtonElement}
 */
const add = document.querySelector('[data-key="add"]');

/**
 * Handles the subtraction of 1 from the displayed number.
 * @returns {void}
 */
const subtractHandler = () => {
    /**
     * The new value after subtraction.
     * @type {number}
     */
    const newValue = parseInt(number.value) - 1;

    number.value = newValue;
```

These JSDoc comments are used to document the purpose of the **subtractHandler** and **addHandler** functions. They specify that these functions have no return value ***(@returns {void}).*** These comments are vital because they clarify the functions' roles and behaviors in the code. They provide information about the functions' expected input and output, making it easier for developers to use and understand these functions. Additionally, code editors and tools can leverage these comments to offer better code suggestions and type checking, improving code quality and reliability.



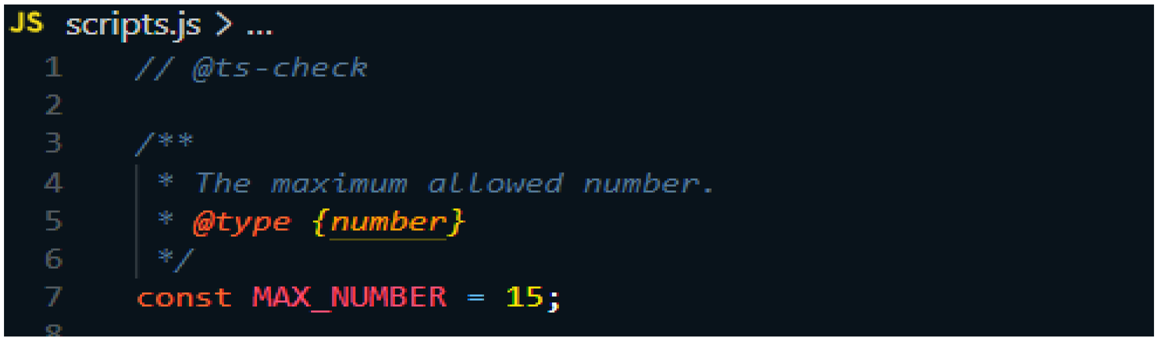
```
/**
 * Represents the HTML button element for addition.
 * @type {HTMLButtonElement}
 */
const add = document.querySelector('[data-key="add"]');

/**
 * Handles the subtraction of 1 from the displayed number.
 * @returns {void}
 */
const subtractHandler = () => {
    /**
     * The new value after subtraction.
     * @type {number}
     */
    const newValue = parseInt(number.value) - 1;

    number.value = newValue;
```

---

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

I put an annotation at the top of the JavaScript file to enable TypeScript checking.

- This makes the code stronger and less likely to have errors related to data types.
- TypeScript checking ensures that the code follows strict rules about how different data types are used.
- This is helpful when working on larger projects or with a team because it catches problems early and prevents issues when the code is running.

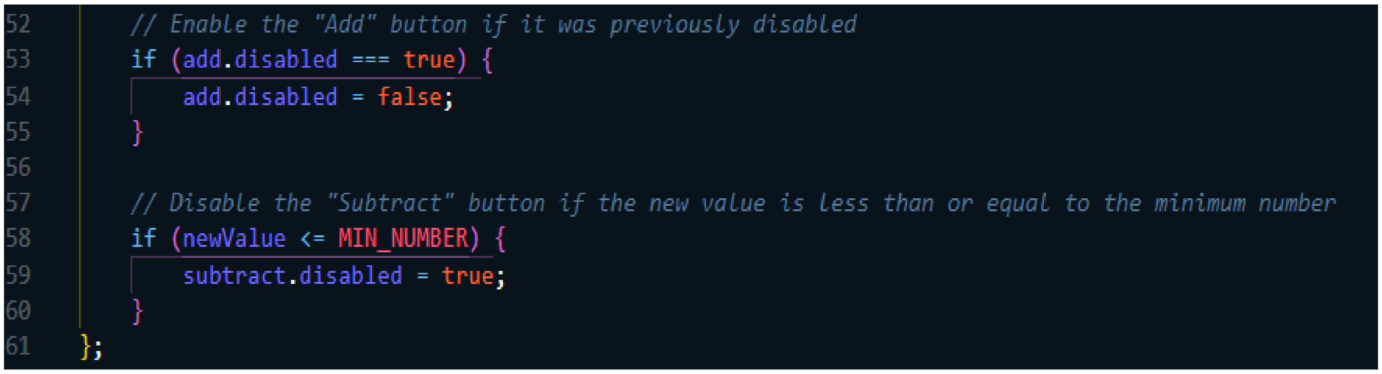

```
JS scripts.js > ...
// @ts-check

/**
 * The maximum allowed number.
 * @type {number}
 */
const MAX_NUMBER = 15;
```

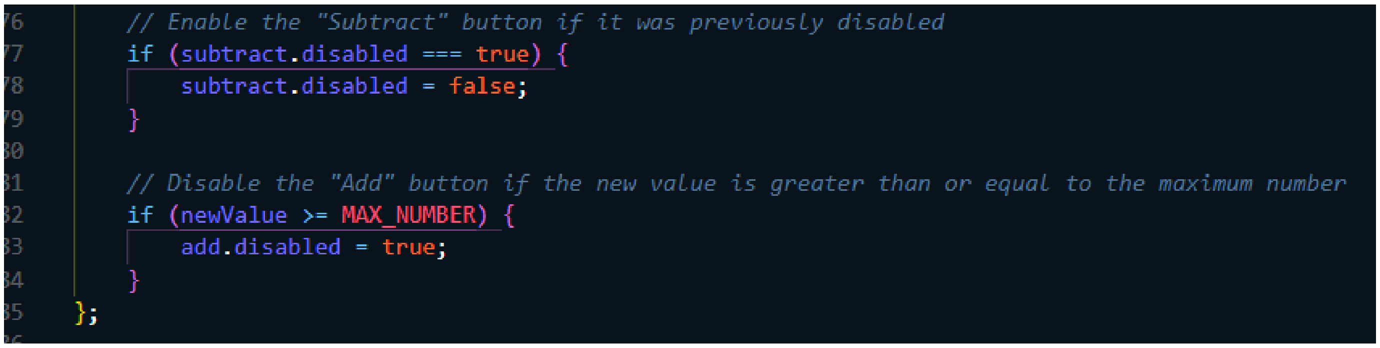
---

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

Along with JSDoc comments, I also added comments in the code to explain why certain parts of the code are written the way they are.

- For example, I explained in comments why the "Add" and "Subtract" buttons are enabled or disabled under specific conditions.
- These comments help future developers understand why the code is written in a particular way. This makes the code easier to work with, reduces confusion, and ensures that the code remains reliable and easy to modify.


```
    // Enable the "Add" button if it was previously disabled
    if (add.disabled === true) {
        add.disabled = false;
    }

    // Disable the "Subtract" button if the new value is less than or equal to the minimum number
    if (newValue <= MIN_NUMBER) {
        subtract.disabled = true;
    }
};
```


```
    // Enable the "Subtract" button if it was previously disabled
    if (subtract.disabled === true) {
        subtract.disabled = false;
    }

    // Disable the "Add" button if the new value is greater than or equal to the maximum number
    if (newValue >= MAX_NUMBER) {
        add.disabled = true;
    }
};
```