

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 used to format text easily. It allows for defining the appearance of text, such as headers, styles, links, and lists, without complex codes. Markdown is commonly used in documentation, articles, and notes, enabling writers to focus on content rather than visual presentation. It is also utilized on platforms like GitHub, where a "readme.md" file is often found in repositories to provide a description or introduction. Markdown can also be used to build web pages, separating writing from presentation and allowing for collaboration. It can customize GitHub profile pages with formatting, links, images, and more. Overall, Markdown simplifies formatting and styling text in various contexts, making it popular among writers and developers.

```
① README.md > 📁 # Tally Count App > 📄 ## Short Description
1  <!-- omit in toc -->
2  # Tally Count App
3
4  ## Short Description
5  Introducing the Tally Counter App - a simple and user-friendly tool designed for efficient tally counting.
6  This app embraces the principle of minimalism, offering only the minimal functionality required for counting,
7  making it an ideal choice for those who value simplicity and ease of use. With its intuitive interface, users can
8  effortlessly keep track of counts with just a few taps. The app provides all the basic features needed in a
9  tally count app, including a large and clear counter display, a reset button for starting fresh, and a convenient
10 count increment button for quick and accurate tallies. Say goodbye to unnecessary complexities and hello to a
11 straightforward tally counting experience with the Tally Counter App.
12
13 Table of Contents (up to date)
14 - [Short Description](#short-description)
15 - [Features](#features)
16 - [Requirements](#requirements)
17
18 ## Features
19 - Simple and user-friendly
20 - Offering the minimal functionality
21 - Provides basic features for counting
22
23 ## Requirements
24
25 - An IDE like VS Code
26 - Basic HTML, CSS and JavaScript skills
27 - A browser similar to Chrome
28 - A very good mouse to count frequently
```

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

JSDoc is a powerful markup language employed to annotate JavaScript source code files. By incorporating comments enriched with JSDoc syntax, programmers can seamlessly augment their codebases with comprehensive documentation that precisely describes the application programming interface (API) of the software they are developing.

```
/**
 * A handler that fires when a user starts dragging an order.
 * It sets the active dragging column in the state object (data.js)
 * to the column over which the order was originally located.
 * @param {Event} event
 */
const handleDragStart = (event) => {
  dragged = event.target.closest('.order')
  state.dragging.source = state.dragging.over
  id = dragged.dataset.id
}
```

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

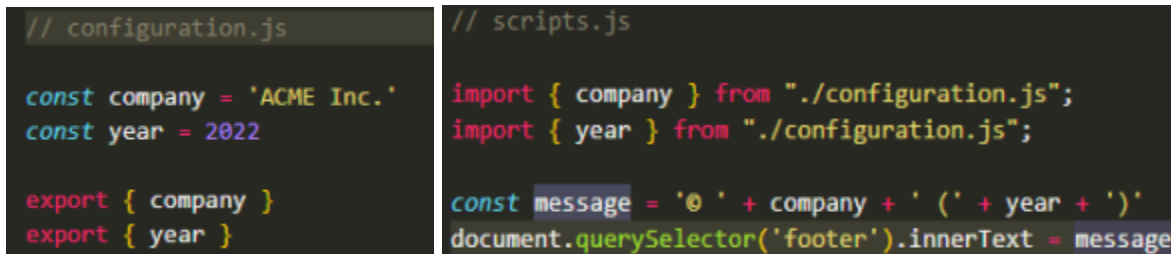
By including the "**@ts-check**" directive in your JavaScript files, you activate TypeScript's static type checking capabilities within your codebase. This feature empowers TypeScript to meticulously analyze and verify the types of variables, function parameters, and return values, ensuring type safety and catching potential errors during the development process. When encountering any type mismatches or inconsistencies, TypeScript promptly raises error notifications, providing developers with valuable insights and enabling them to rectify these issues proactively. The integration of "**@ts-check**" significantly enhances the reliability and robustness of JavaScript codebases, promoting a more professional and rigorous approach to software development.



```
JS Challenge1Script.js X JS Challenge2Script.js
Challenge1 > JS Challenge1Script.js > ...
1 //@ts-check
2 const firstName = 'John';
3 const age = 35;
4 const hobby = 'Coding';
5
6 const logTwice = () => {
7   console.log('parameter')
8   console.log('parameter')
9 }
10
11 function hello () {
12   console.log(`Hello, ${firstName} (${age}). I love ${hobby}!`)
13 }
14
15 hello()
16 hello()
```

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

I have chosen the **Import** and **Export** example of my IWA3 Challenges.



```
// configuration.js
const company = 'ACME Inc.'
const year = 2022

export { company }
export { year }

// scripts.js
import { company } from "./configuration.js";
import { year } from "./configuration.js";

const message = `@ ` + company + ` (' + year + `)'
document.querySelector('footer').innerText = message
```

