

DWA_03.4 Knowledge Check_DWA3.1

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

The screenshot shows a code editor with a file explorer on the left, a code editor in the center, and a preview window on the right. The file explorer shows a project named 'DYNAMIC_WEB_APPS' with subfolders 'DWA1', 'DWA2', and 'DWA3'. The 'DWA3' folder is selected, and the 'README.md' file is open. The code editor shows the following content:

```
1 # Dynamic Web Apps
2
3 ## DWA Part 1: Structuring JavaScript (2023)
4
5 ### Level | No Level
6
7 ### **Start Course**
8
9
10 In Part 1 of this course, Structuring JavaScript, we teach students how to write
11 clean, maintainable, and scalable code in JavaScript. The content covers various areas of
12 JavaScript development, including code style, documentation, error handling, abstraction,
13 object-oriented programming, functional programming, and state machines, teaching
14 students the skills and knowledge they need to structure their code in a way that makes it easy
15 to manage and maintain.
16
17 ### Structuring JavaScript Learning objectives:
18 - Understand strategies for managing complexity and structuring code to make it more maintainable.
19 - Learn about the origins of JavaScript and how the language is standardised and developed further.
20 - Understand the value of documentation and how different kinds of documentation are used in a project.
21 - Understand how a consistent code style makes code easier to read and maintain and discover tools for efficiently implementing these conventions.
22 - Learn how to generate errors in your JavaScript to make your code more maintainable and expandable.
23 - Develop an understanding of how abstraction highlights the crucial features of code to developers while hiding the details.
24 - Understand the principles of object-oriented programming (OOP) and how it contributes to code modularity and maintainability.
25 - Learn the principles of encapsulation and how to write more secure code.
26 - Understand the principles of polymorphism and
```

The preview window shows the rendered content of the Markdown file, which includes the following sections:

Dynamic Web Apps

DWA Part 1: Structuring JavaScript (2023)

Level | No Level

Start Course

In Part 1 of this course, **Structuring JavaScript**, we teach students how to write clean, maintainable, and scalable code in JavaScript. The content covers various areas of JavaScript development, including code style, documentation, error handling, abstraction, object-oriented programming, functional programming, and state machines, teaching students the skills and knowledge they need to structure their code in a way that makes it easy to manage and maintain.

Structuring JavaScript Learning objectives:

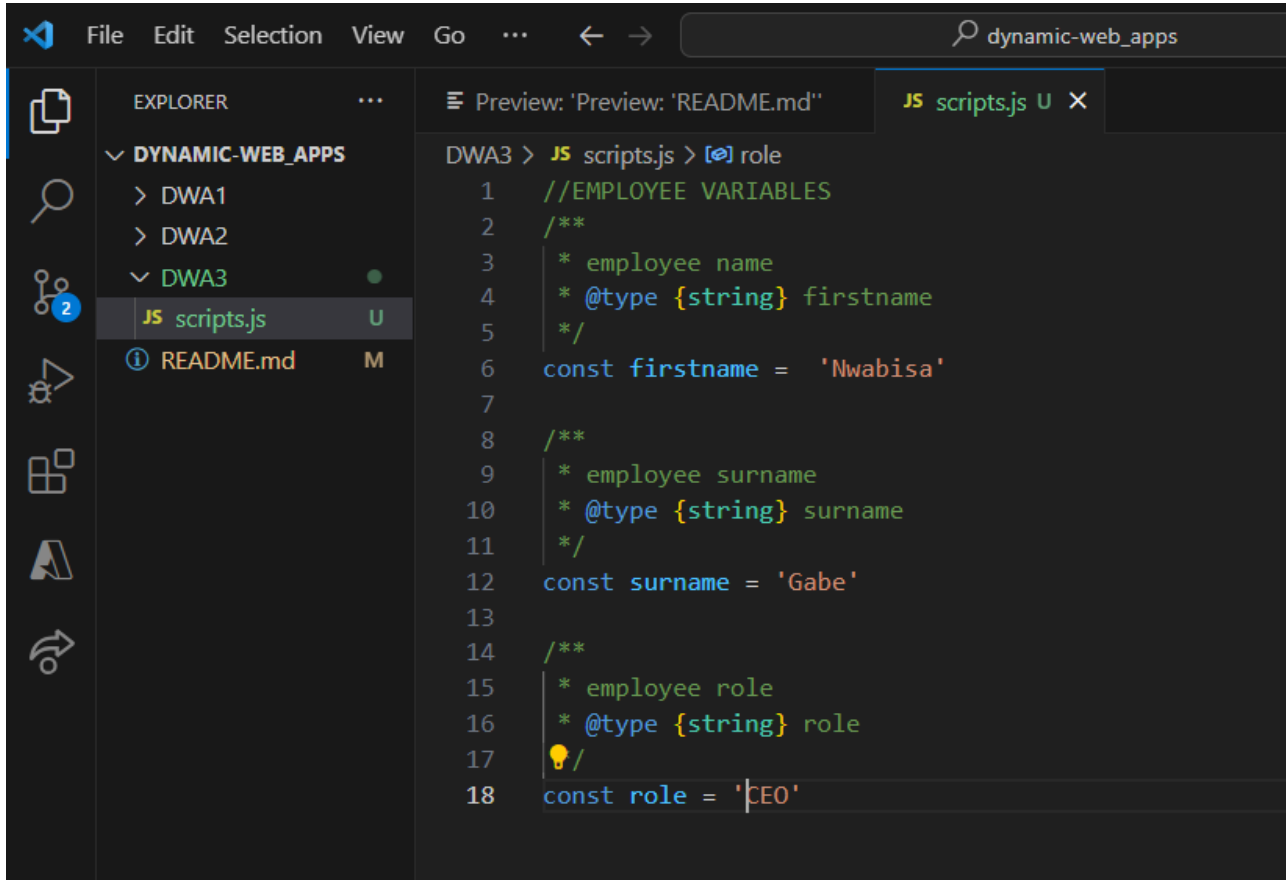
- Understand strategies for managing complexity and structuring code to make it more maintainable.
- Learn about the origins of JavaScript and how the language is standardised and developed further.
- Understand the value of documentation and how different kinds of documentation are used in a project.
- Understand how a consistent code style makes code easier to read and maintain and discover tools for efficiently implementing these conventions.
- Learn how to generate errors in your JavaScript to make your code more maintainable and expandable.
- Develop an understanding of how abstraction highlights the crucial features of code to developers while hiding the details.
- Understand the principles of object-oriented programming (OOP) and how it contributes to code modularity and maintainability.
- Learn the principles of encapsulation and how to write more secure code.
- Understand the principles of polymorphism and inheritance and how to use them to share code between related object classes.
- Develop an understanding of functional programming concepts, including purity and immutability, and how to use higher-order functions to transform data.
- Learn how to create state machines to model complex systems.
- Practice writing clean code through hands-on coding exercises and examples.

In Part 2 of this course, **JavaScript Frameworks**, we cover a range of popular JavaScript frameworks and tools for building modern web applications. The content covers everything from the most common frameworks like Angular, React, and Vue to newer, more lightweight alternatives like Alpine and Svelte. Students will learn to use these frameworks to build scalable, testable, high-performing applications.

JavaScript Frameworks Learning Objectives:

- Understand what JavaScript frameworks are and how they can help you

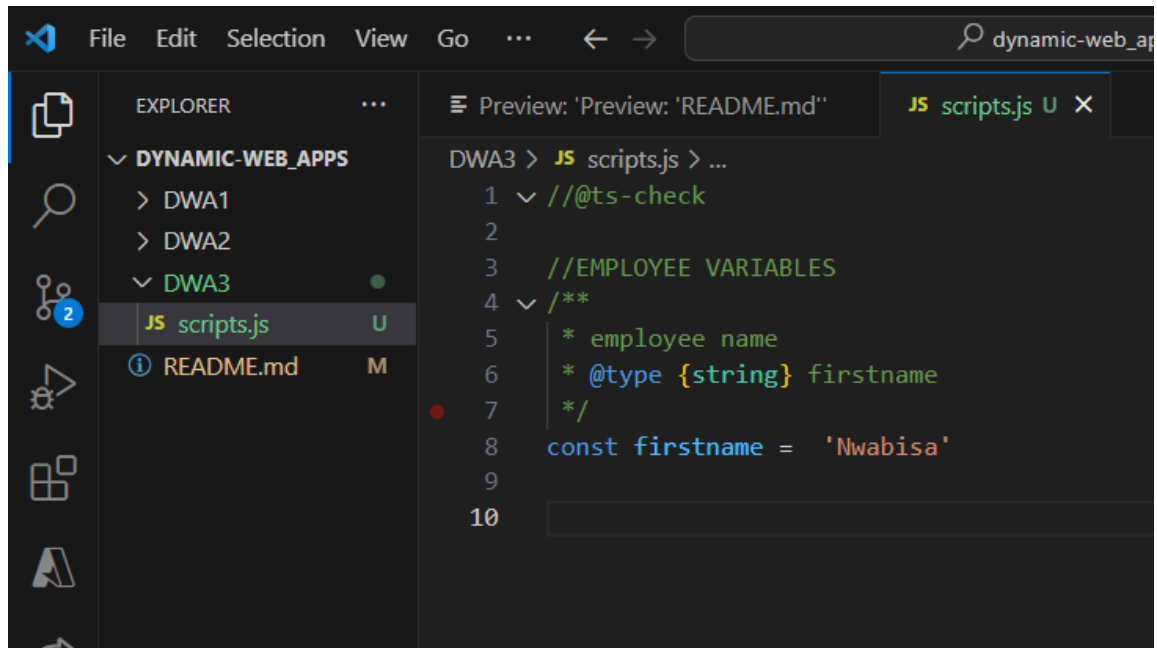
2. Please show how you applied JSDoc Comments to a piece of your code.



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The 'DYNAMIC-WEB_APPS' folder is expanded, showing subfolders 'DWA1', 'DWA2', and 'DWA3'. Inside 'DWA3', the file 'scripts.js' is selected, indicated by a blue highlight and a '2' in a circle. The 'README.md' file is also visible. The main editor area displays the content of 'scripts.js', which includes JSDoc comments for variables 'firstname', 'surname', and 'role'. The code is as follows:

```
DWA3 > JS scripts.js > [🔍] role
1  //EMPLOYEE VARIABLES
2  /**
3   * employee name
4   * @type {string} firstname
5   */
6  const firstname = 'Nwabisa'
7
8  /**
9   * employee surname
10  * @type {string} surname
11  */
12  const surname = 'Gabe'
13
14  /**
15  * employee role
16  * @type {string} role
17  *
18  const role = 'CEO'
```

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

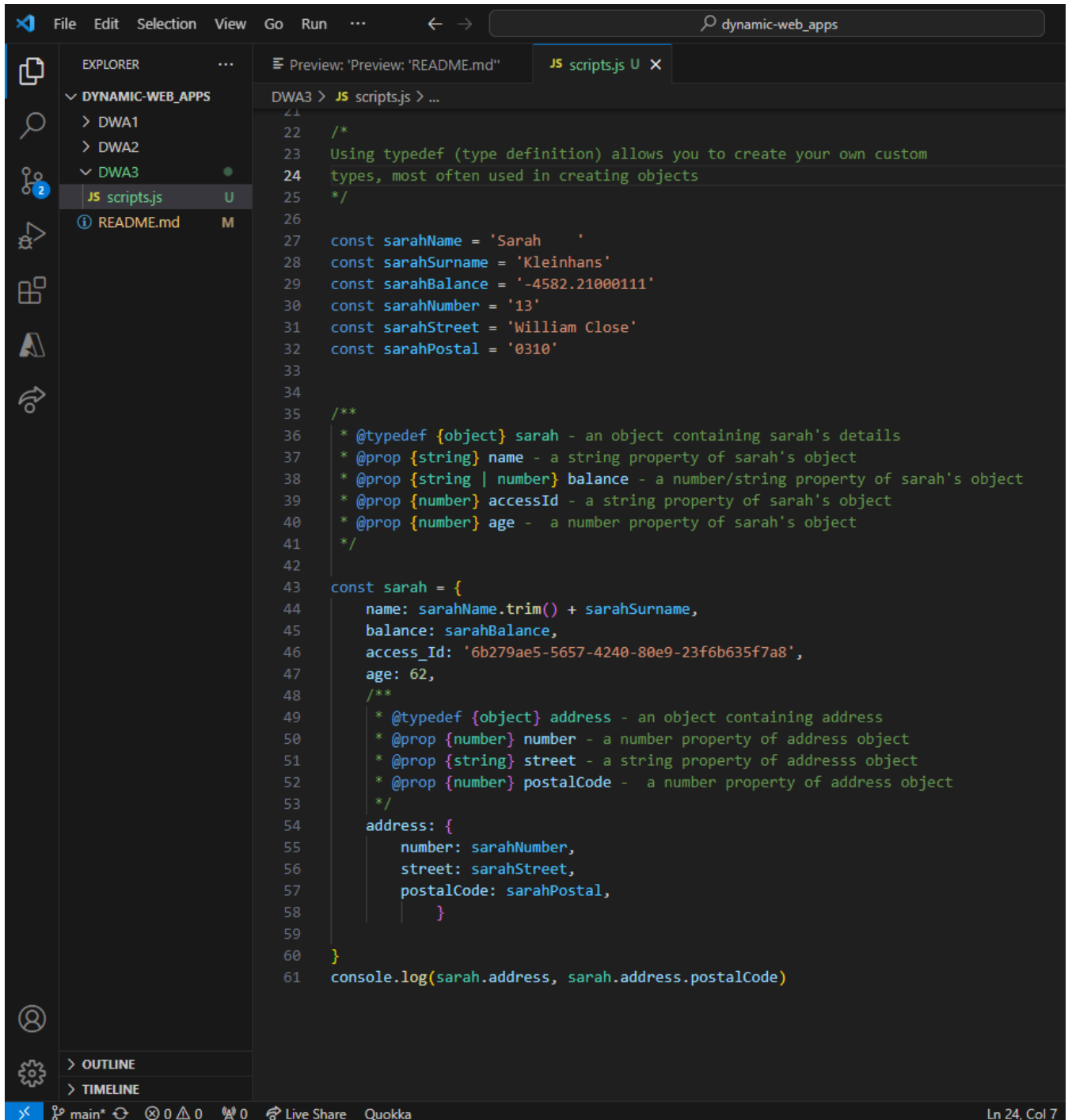


The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left and the Editor view on the right. The Explorer sidebar shows a project structure with a folder named 'DYNAMIC-WEB_APPS' containing subfolders 'DWA1', 'DWA2', and 'DWA3'. Inside 'DWA3', there is a file 'scripts.js' (highlighted with a blue selection bar and a 'U' icon) and a file 'README.md' (with an 'M' icon). The Editor view shows the content of 'scripts.js' with the following code:

```
DWA3 > JS scripts.js > ...
1  //@ts-check
2
3  //EMPLOYEE VARIABLES
4  /**
5   * employee name
6   * @type {string} firstname
7   */
8  const firstname = 'Nwabisa'
9
10
```

The code demonstrates the application of the `@ts-check` annotation at the top of the file (line 1) to enable TypeScript checking for the JavaScript code that follows. The code also includes a JSDoc comment (lines 4-7) for the `firstname` variable, which is then assigned the value `'Nwabisa'` on line 8.

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



```
File Edit Selection View Go Run ... ← → dynamic-web_apps

EXPLORER
DYNAMIC-WEB_APPS
  DWA1
  DWA2
  DWA3
    JS scripts.js
    README.md

Preview: 'Preview: README.md' JS scripts.js U X
DWA3 > JS scripts.js > ...
21
22 /*
23 Using typedef (type definition) allows you to create your own custom
24 types, most often used in creating objects
25 */
26
27 const sarahName = 'Sarah '
28 const sarahSurname = 'Kleinhans'
29 const sarahBalance = '-4582.21000111'
30 const sarahNumber = '13'
31 const sarahStreet = 'William Close'
32 const sarahPostal = '0310'
33
34
35 /**
36  * @typedef {object} sarah - an object containing sarah's details
37  * @prop {string} name - a string property of sarah's object
38  * @prop {string | number} balance - a number/string property of sarah's object
39  * @prop {number} accessId - a string property of sarah's object
40  * @prop {number} age - a number property of sarah's object
41  */
42
43 const sarah = {
44   name: sarahName.trim() + sarahSurname,
45   balance: sarahBalance,
46   access_Id: '6b279ae5-5657-4240-80e9-23f6b635f7a8',
47   age: 62,
48   /**
49    * @typedef {object} address - an object containing address
50    * @prop {number} number - a number property of address object
51    * @prop {string} street - a string property of address object
52    * @prop {number} postalCode - a number property of address object
53    */
54   address: {
55     number: sarahNumber,
56     street: sarahStreet,
57     postalCode: sarahPostal,
58   }
59 }
60
61 console.log(sarah.address, sarah.address.postalCode)
```

main* 0 0 0 Live Share Quokka Ln 24, Col 7

