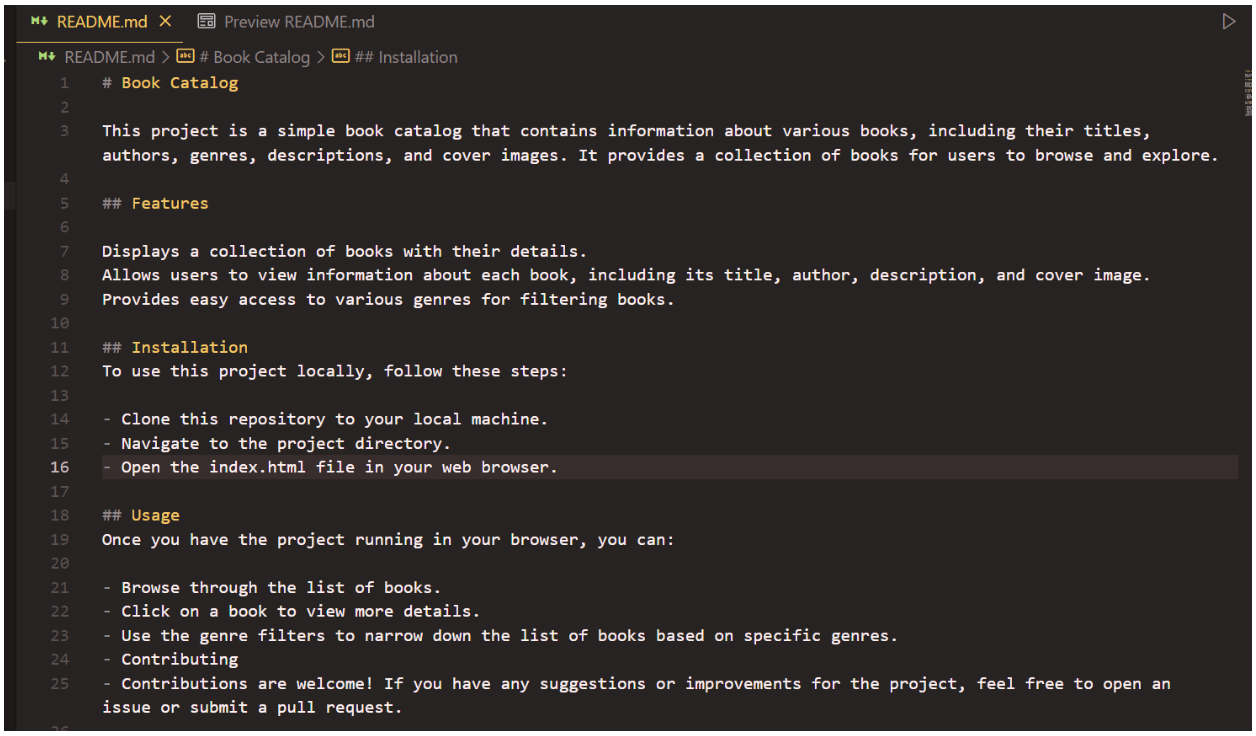
# DWA_03.4 Knowledge Check_DWA3.1

---

1. Please show how you applied a Markdown File to a piece of your code.
Book App from Interactive Web Apps:

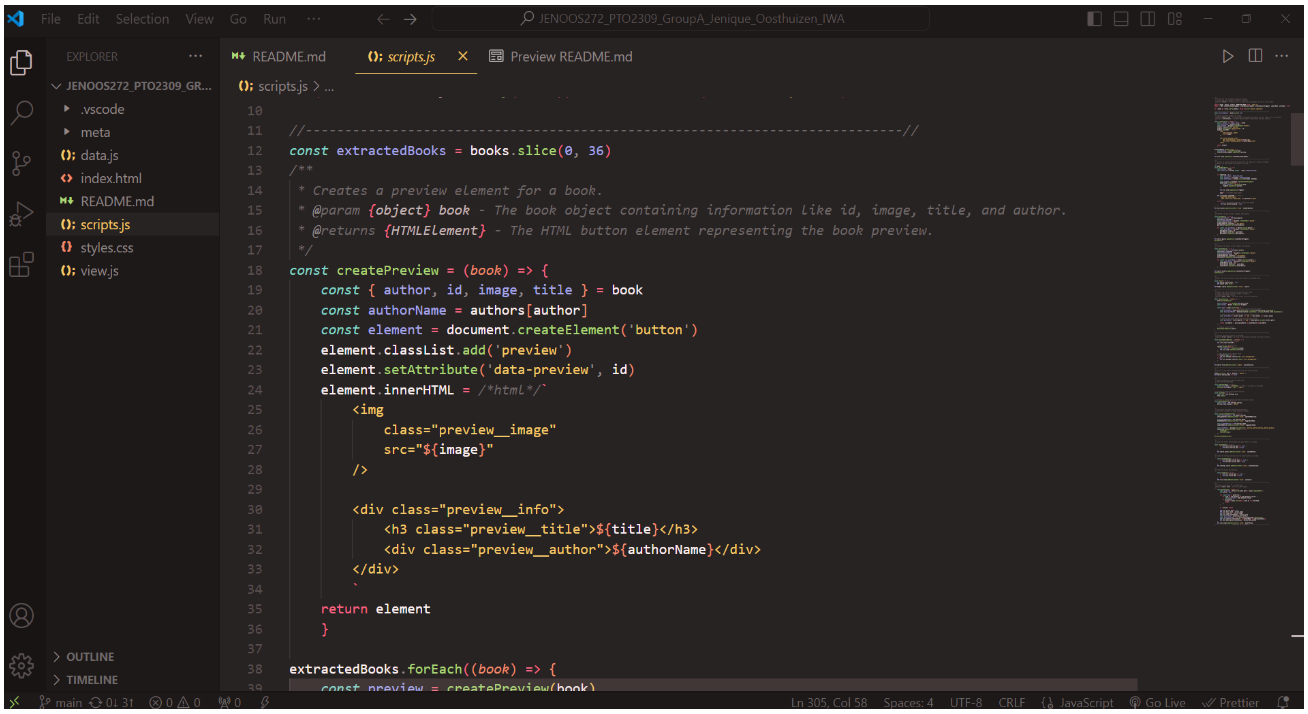


```markdown
# Book Catalog

This project is a simple book catalog that contains information about various books, including their titles, authors, genres, descriptions, and cover images. It provides a collection of books for users to browse and explore.

## Features

Displays a collection of books with their details.
Allows users to view information about each book, including its title, author, description, and cover image.
Provides easy access to various genres for filtering books.

## Installation
To use this project locally, follow these steps:

- Clone this repository to your local machine.
- Navigate to the project directory.
- Open the index.html file in your web browser.

## Usage
Once you have the project running in your browser, you can:

- Browse through the list of books.
- Click on a book to view more details.
- Use the genre filters to narrow down the list of books based on specific genres.
- Contributing
- Contributions are welcome! If you have any suggestions or improvements for the project, feel free to open an issue or submit a pull request.
```

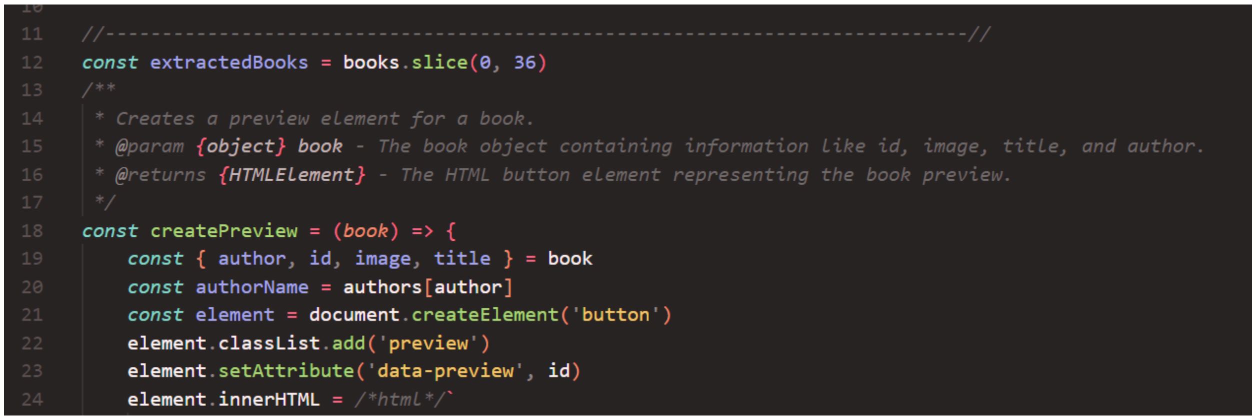
Markdown was created by John Gruber in 2004. Markdown is a lightweight markup language that you can use to add formatting elements to plaintext documents, it is also now one of the world's most popular markup languages.

---

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



```javascript
//--------------------------------------------------------------------------//
const extractedBooks = books.slice(0, 36)
/**
 * Creates a preview element for a book.
 * @param {object} book - The book object containing information like id, image, title, and author.
 * @returns {HTMLElement} - The HTML button element representing the book preview.
 */
const createPreview = (book) => {
    const { author, id, image, title } = book
    const authorName = authors[author]
    const element = document.createElement('button')
    element.classList.add('preview')
    element.setAttribute('data-preview', id)
    element.innerHTML = /*html*/`
        <img
            class="preview__image"
            src="${image}"
        />

        <div class="preview__info">
            <h3 class="preview__title">${title}</h3>
            <div class="preview__author">${authorName}</div>
        </div>
    `

    return element
    }

extractedBooks.forEach((book) => {
    const preview = createPreview(book)
```



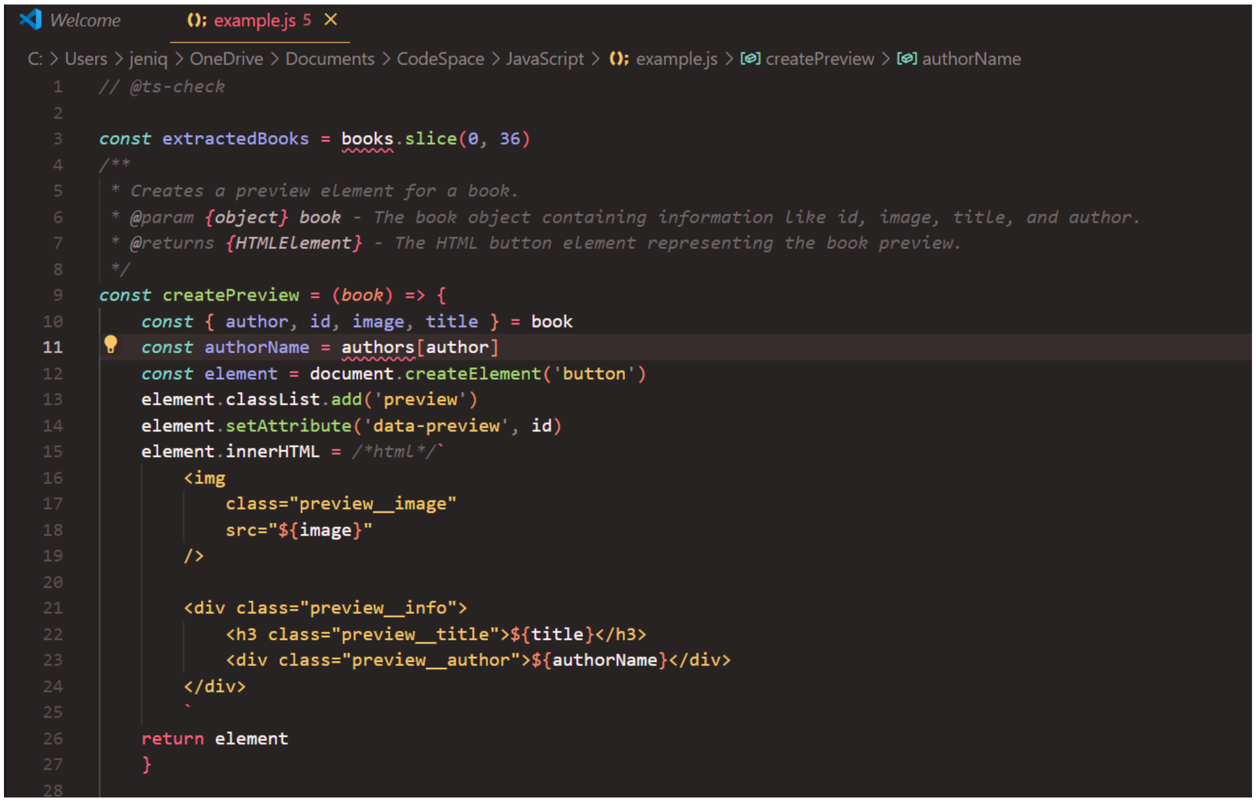
```javascript
//--------------------------------------------------------------------------//
const extractedBooks = books.slice(0, 36)
/**
 * Creates a preview element for a book.
 * @param {object} book - The book object containing information like id, image, title, and author.
 * @returns {HTMLElement} - The HTML button element representing the book preview.
 */
const createPreview = (book) => {
    const { author, id, image, title } = book
    const authorName = authors[author]
    const element = document.createElement('button')
    element.classList.add('preview')
    element.setAttribute('data-preview', id)
    element.innerHTML = /*html*/`
```

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.

[https://jsdoc.app/about-getting-started#:~:text=JSDoc%20comments%20should%20generally%20be,3%20stars%20will%20be%20ignored](https://jsdoc.app/about-getting-started#:~:text=JSDoc%20comments%20should%20generally%20be,3%20stars%20will%20be%20ignored).

3. Please show how you applied the @ts-check annotation to a piece of your code. TSCheck is a tool for finding bugs in hand-written TypeScript type definitions (.d.ts files). It works by comparing the type definitions to the actual JavaScript library implementation.



Welcome   example.js 5 ×

C: > Users > jeniq > OneDrive > Documents > CodeSpace > JavaScript > example.js > createPreview > authorName

```
// @ts-check

const extractedBooks = books.slice(0, 36)
/**
 * Creates a preview element for a book.
 * @param {object} book - The book object containing information like id, image, title, and author.
 * @returns {HTMLElement} - The HTML button element representing the book preview.
 */
const createPreview = (book) => {
    const { author, id, image, title } = book
    const authorName = authors[author]
    const element = document.createElement('button')
    element.classList.add('preview')
    element.setAttribute('data-preview', id)
    element.innerHTML = /*html*/`
        <img
            class="preview__image"
            src="${image}"
        />

        <div class="preview__info">
            <h3 class="preview__title">${title}</h3>
            <div class="preview__author">${authorName}</div>
        </div>
        `

    return element
    }

```

https://github.com/asgerf/tscheck

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