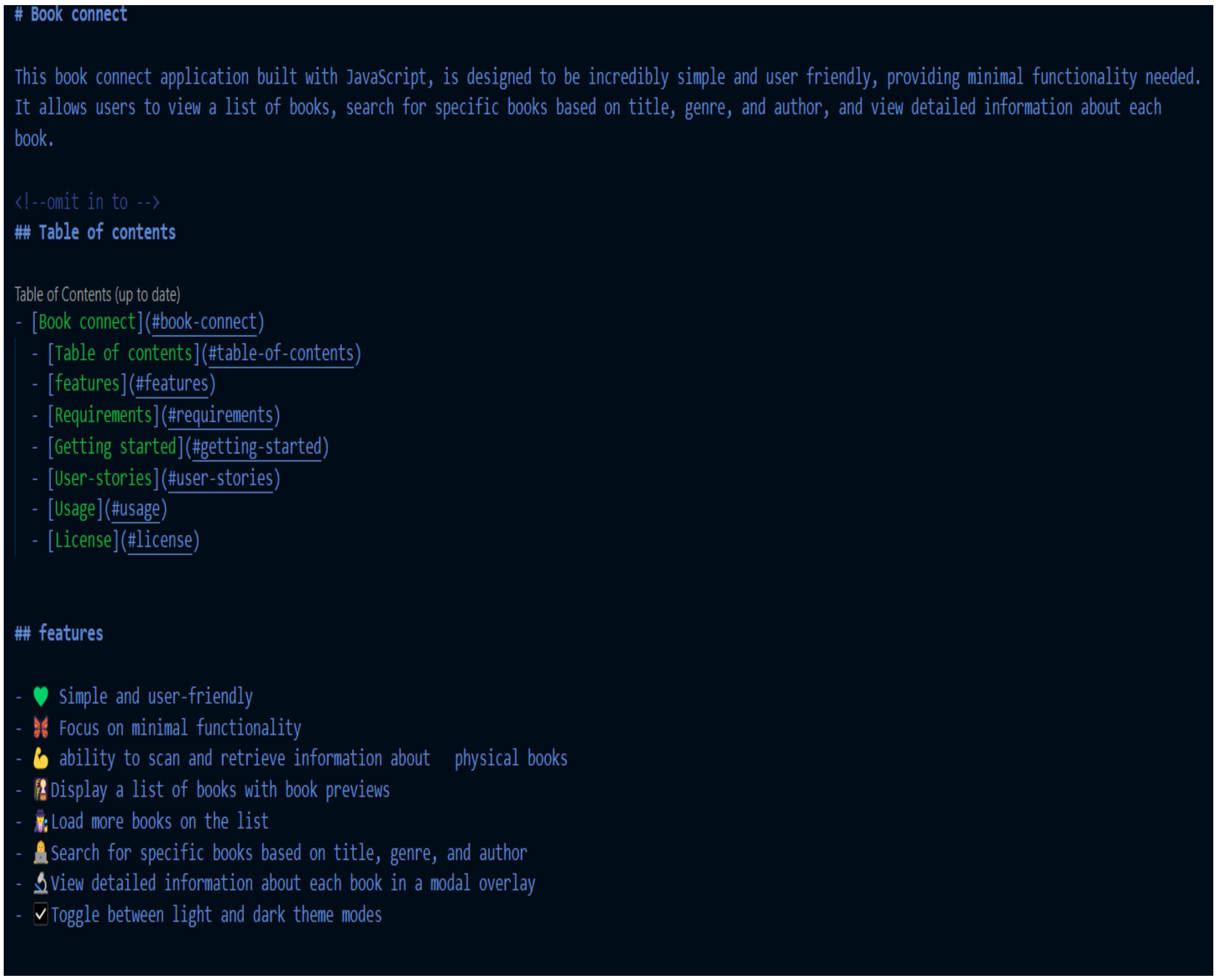
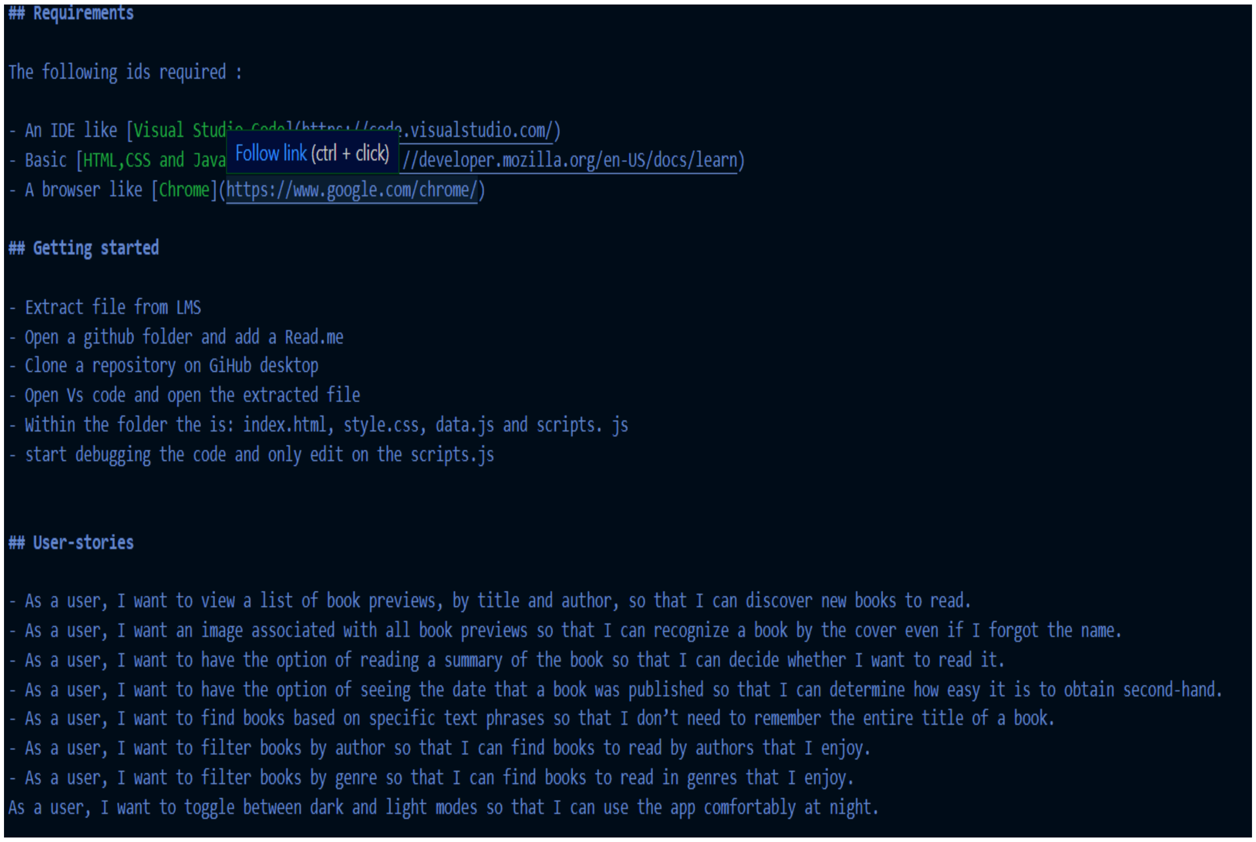
DWA_03.4 Knowledge Check_DWA3.1

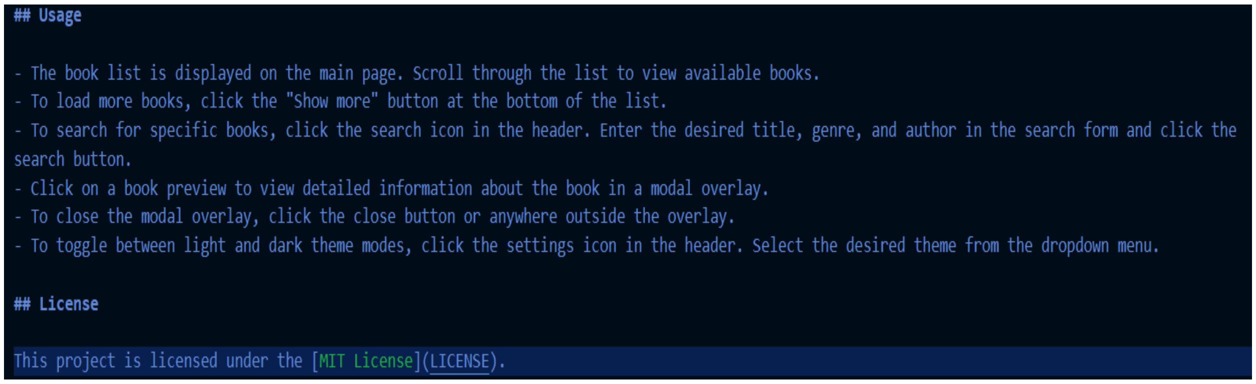
1. Please show how you applied a Markdown File to a piece of your code.



```
# Book connect

This book connect application built with JavaScript, is designed to be incredibly simple and user friendly, providing minimal functionality needed.
It allows users to view a list of books, search for specific books based on title, genre, and author, and view detailed information about each
book.

<!--omit in to -->
## Table of contents

Table of Contents (up to date)
- [Book connect](#book-connect)
  - [Table of contents](#table-of-contents)
  - [features](#features)
  - [Requirements](#requirements)
  - [Getting started](#getting-started)
  - [User-stories](#user-stories)
  - [Usage](#usage)
  - [License](#license)


## features

- 💚 Simple and user-friendly
- 🦋 Focus on minimal functionality
- 💪 ability to scan and retrieve information about   physical books
- 👩‍🌾Display a list of books with book previews
- 🕵️‍♀️Load more books on the list
- 👩‍💻Search for specific books based on title, genre, and author
- 🔬View detailed information about each book in a modal overlay
- ☑️Toggle between light and dark theme modes
```



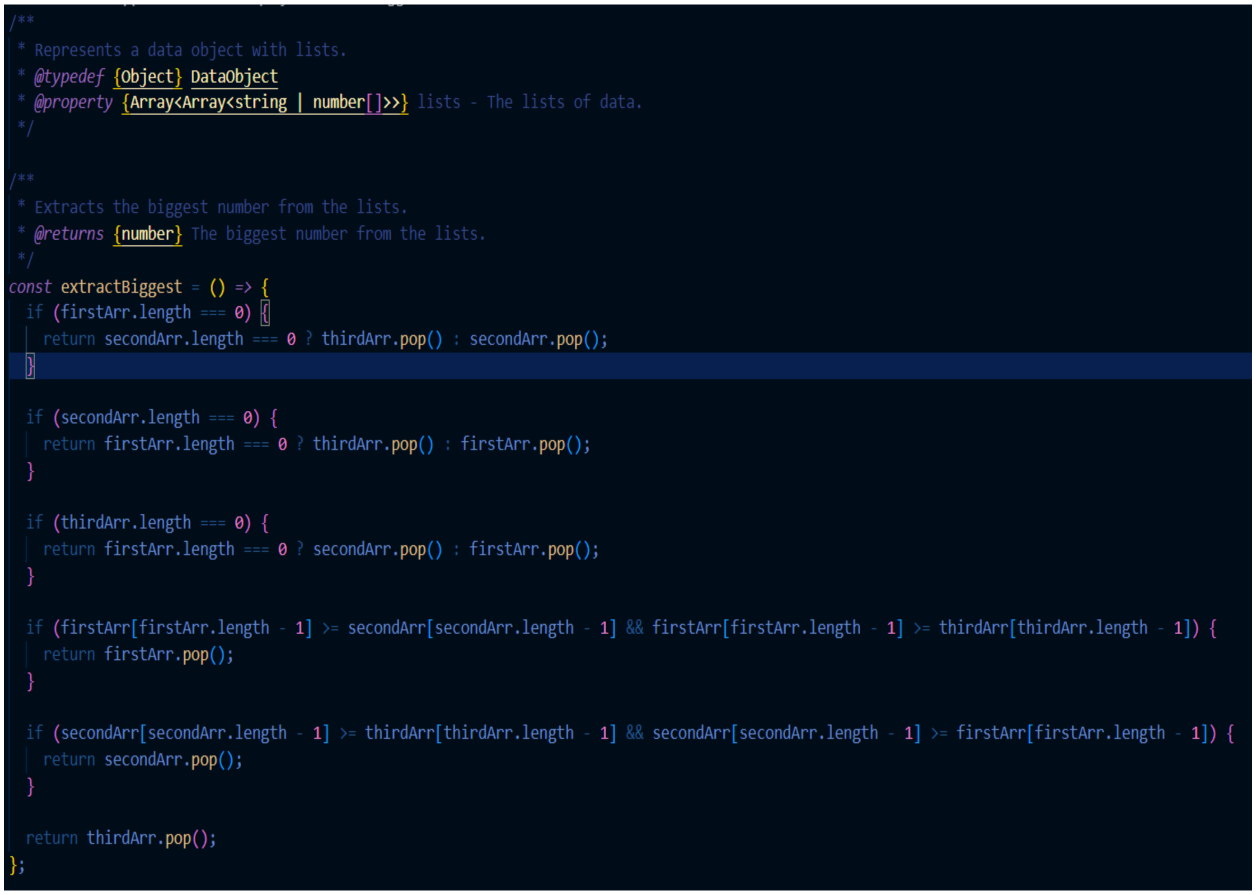
```
## Requirements

The following ids required :

- An IDE like [Visual Studio Code](https://code.visualstudio.com/)
- Basic [HTML,CSS and Java //developer.mozilla.org/en-US/docs/learn)
- A browser like [Chrome](https://www.google.com/chrome/)

## Getting started

- Extract file from LMS
- Open a github folder and add a Read.me
- Clone a repository on GiHub desktop
- Open Vs code and open the extracted file
- Within the folder the is: index.html, style.css, data.js and scripts. js
- start debugging the code and only edit on the scripts.js


## User-stories

- As a user, I want to view a list of book previews, by title and author, so that I can discover new books to read.
- As a user, I want an image associated with all book previews so that I can recognize a book by the cover even if I forgot the name.
- As a user, I want to have the option of reading a summary of the book so that I can decide whether I want to read it.
- As a user, I want to have the option of seeing the date that a book was published so that I can determine how easy it is to obtain second-hand.
- As a user, I want to find books based on specific text phrases so that I don’t need to remember the entire title of a book.
- As a user, I want to filter books by author so that I can find books to read by authors that I enjoy.
- As a user, I want to filter books by genre so that I can find books to read in genres that I enjoy.
As a user, I want to toggle between dark and light modes so that I can use the app comfortably at night.
```



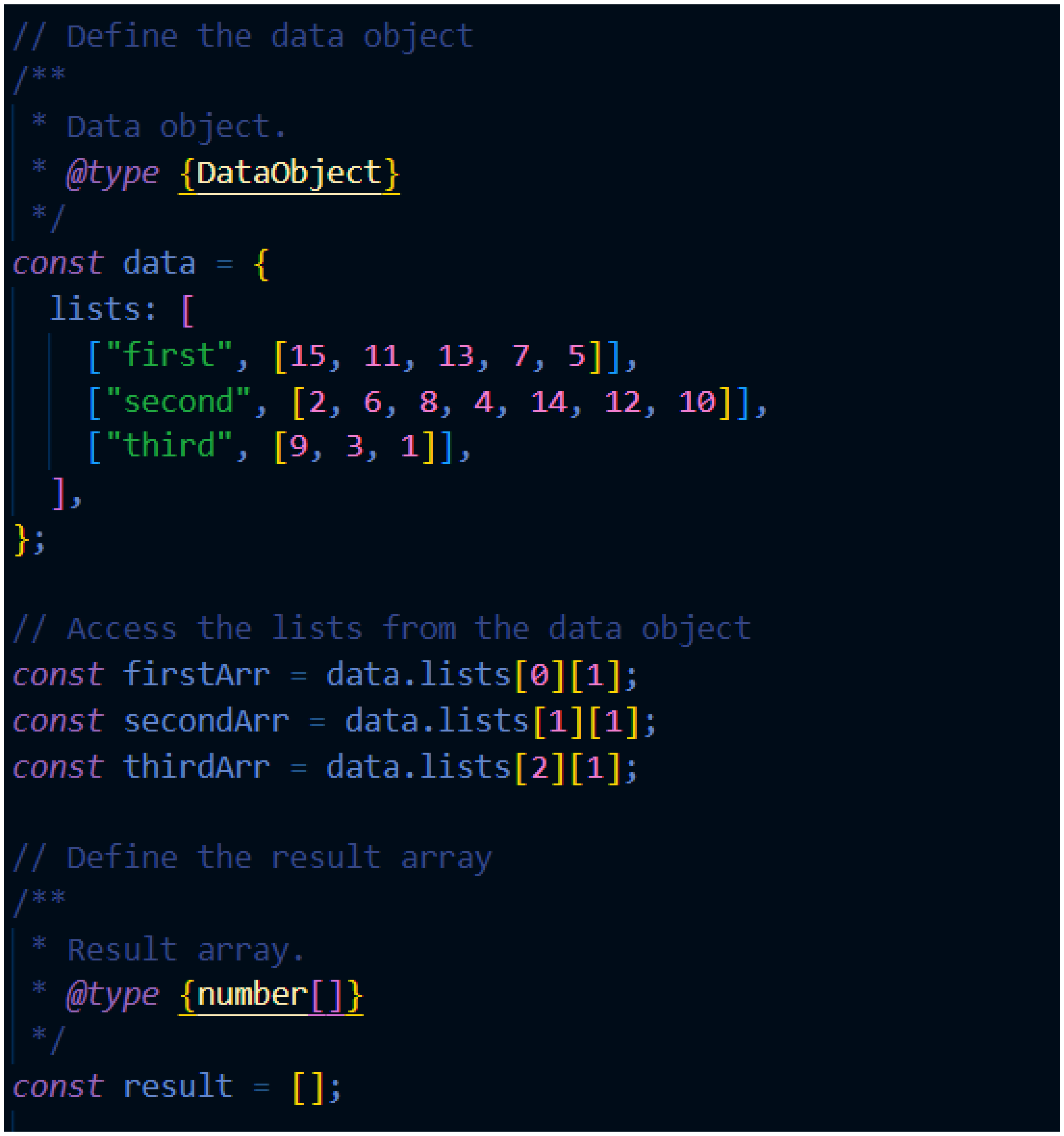
```
## Usage

- The book list is displayed on the main page. Scroll through the list to view available books.
- To load more books, click the "Show more" button at the bottom of the list.
- To search for specific books, click the search icon in the header. Enter the desired title, genre, and author in the search form and click the
search button.
- Click on a book preview to view detailed information about the book in a modal overlay.
- To close the modal overlay, click the close button or anywhere outside the overlay.
- To toggle between light and dark theme modes, click the settings icon in the header. Select the desired theme from the dropdown menu.

## License

This project is licensed under the [MIT License](LICENSE).
```

---

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



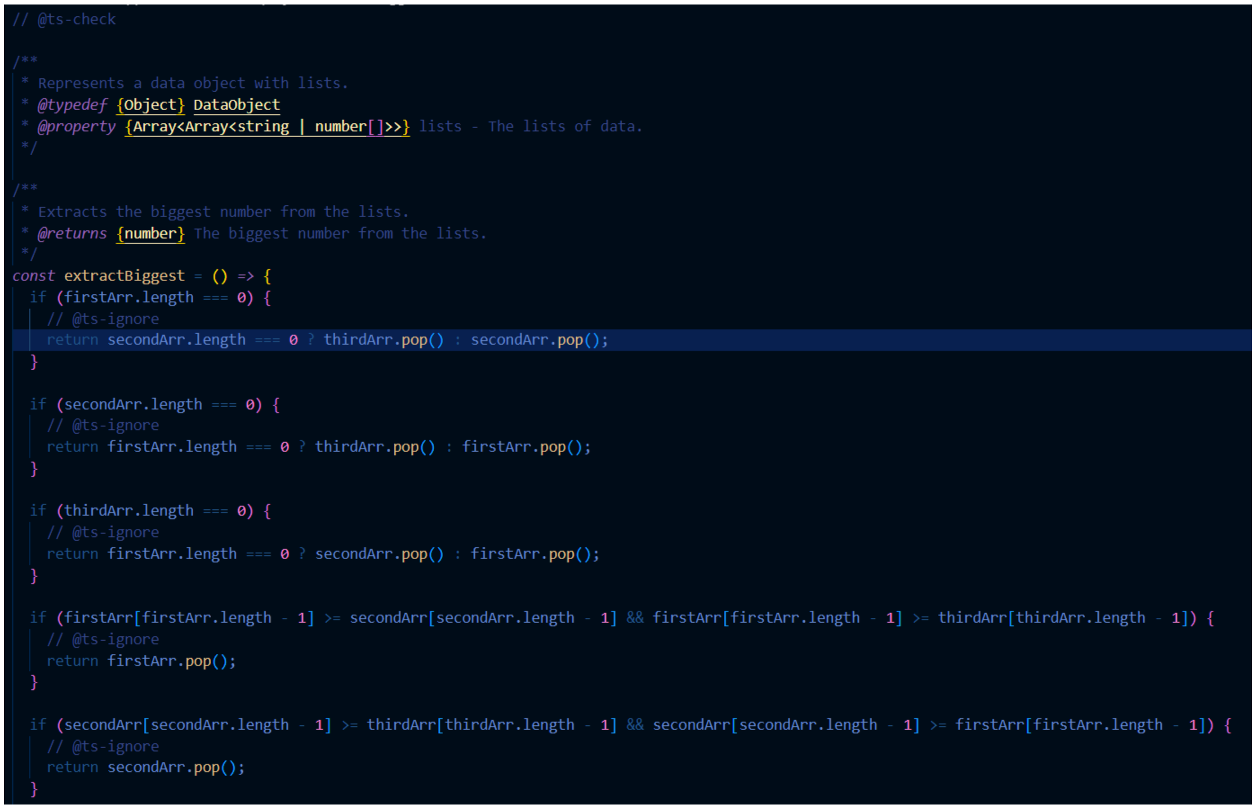
```javascript
/**
 * Represents a data object with lists.
 * @typedef {Object} DataObject
 * @property {Array<Array<string | number[]>>} lists - The lists of data.
 */

/**
 * Extracts the biggest number from the lists.
 * @returns {number} The biggest number from the lists.
 */
const extractBiggest = () => {
  if (firstArr.length === 0) {
    return secondArr.length === 0 ? thirdArr.pop() : secondArr.pop();
  }

  if (secondArr.length === 0) {
    return firstArr.length === 0 ? thirdArr.pop() : firstArr.pop();
  }

  if (thirdArr.length === 0) {
    return firstArr.length === 0 ? secondArr.pop() : firstArr.pop();
  }

  if (firstArr[firstArr.length - 1] >= secondArr[secondArr.length - 1] && firstArr[firstArr.length - 1] >= thirdArr[thirdArr.length - 1]) {
    return firstArr.pop();
  }

  if (secondArr[secondArr.length - 1] >= thirdArr[thirdArr.length - 1] && secondArr[secondArr.length - 1] >= firstArr[firstArr.length - 1]) {
    return secondArr.pop();
  }

  return thirdArr.pop();
};
```



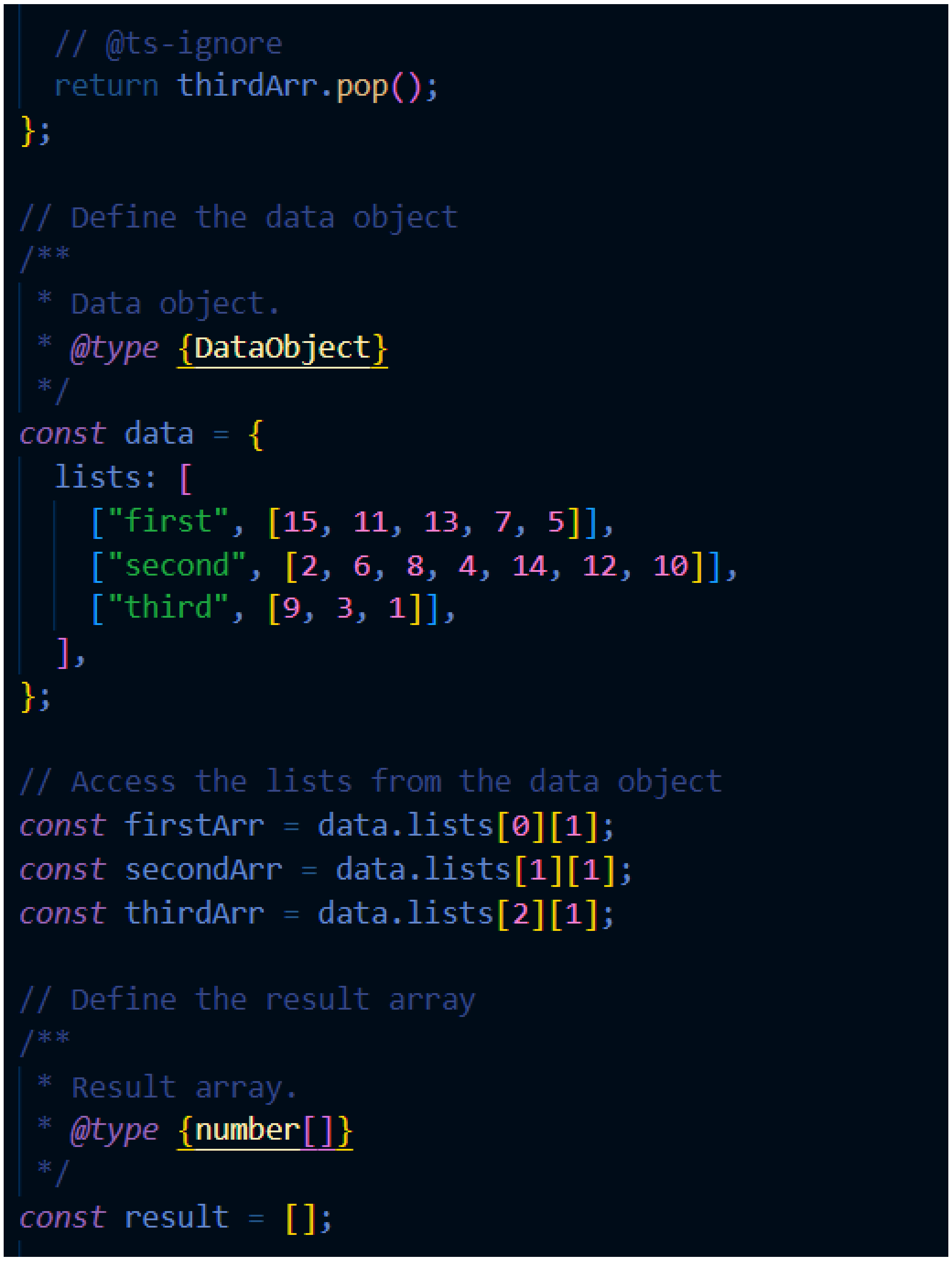
```
// Define the data object
/**
 * Data object.
 * @type {DataObject}
 */
const data = {
  lists: [
    ["first", [15, 11, 13, 7, 5]],
    ["second", [2, 6, 8, 4, 14, 12, 10]],
    ["third", [9, 3, 1]],
  ],
};

// Access the lists from the data object
const firstArr = data.lists[0][1];
const secondArr = data.lists[1][1];
const thirdArr = data.lists[2][1];

// Define the result array
/**
 * Result array.
 * @type {number[]}
 */
const result = [];
```

---

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



```
// @ts-check

/**
 * Represents a data object with lists.
 * @typedef {Object} DataObject
 * @property {Array<Array<string | number[]>>} lists - The lists of data.
 */

/**
 * Extracts the biggest number from the lists.
 * @returns {number} The biggest number from the lists.
 */
const extractBiggest = () => {
  if (firstArr.length === 0) {
    // @ts-ignore
    return secondArr.length === 0 ? thirdArr.pop() : secondArr.pop();
  }

  if (secondArr.length === 0) {
    // @ts-ignore
    return firstArr.length === 0 ? thirdArr.pop() : firstArr.pop();
  }

  if (thirdArr.length === 0) {
    // @ts-ignore
    return firstArr.length === 0 ? secondArr.pop() : firstArr.pop();
  }

  if (firstArr[firstArr.length - 1] >= secondArr[secondArr.length - 1] && firstArr[firstArr.length - 1] >= thirdArr[thirdArr.length - 1]) {
    // @ts-ignore
    return firstArr.pop();
  }

  if (secondArr[secondArr.length - 1] >= thirdArr[thirdArr.length - 1] && secondArr[secondArr.length - 1] >= firstArr[firstArr.length - 1]) {
    // @ts-ignore
    return secondArr.pop();
  }
```

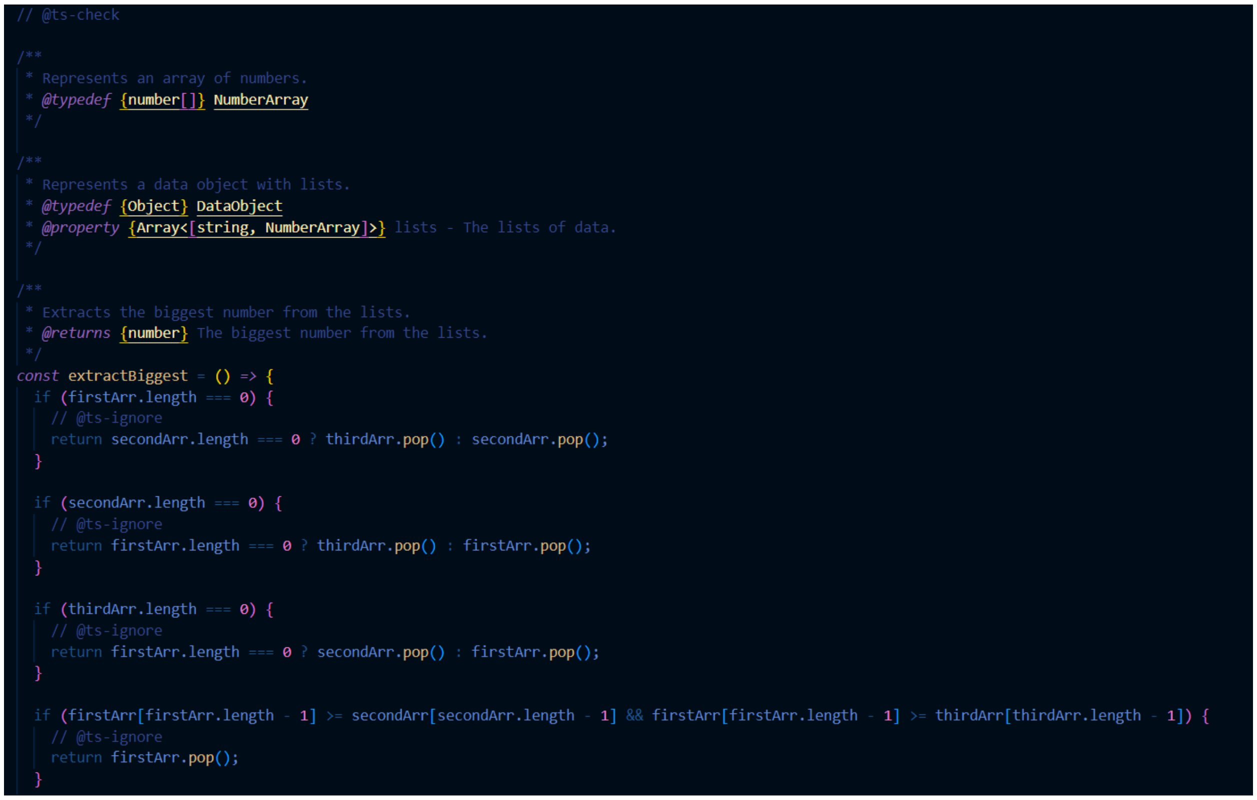


```
  // @ts-ignore
  return thirdArr.pop();
};

// Define the data object
/**
 * Data object.
 * @type {DataObject}
 */
const data = {
  lists: [
    ["first", [15, 11, 13, 7, 5]],
    ["second", [2, 6, 8, 4, 14, 12, 10]],
    ["third", [9, 3, 1]],
  ],
};

// Access the lists from the data object
const firstArr = data.lists[0][1];
const secondArr = data.lists[1][1];
const thirdArr = data.lists[2][1];

// Define the result array
/**
 * Result array.
 * @type {number[]}
 */
const result = [];
```

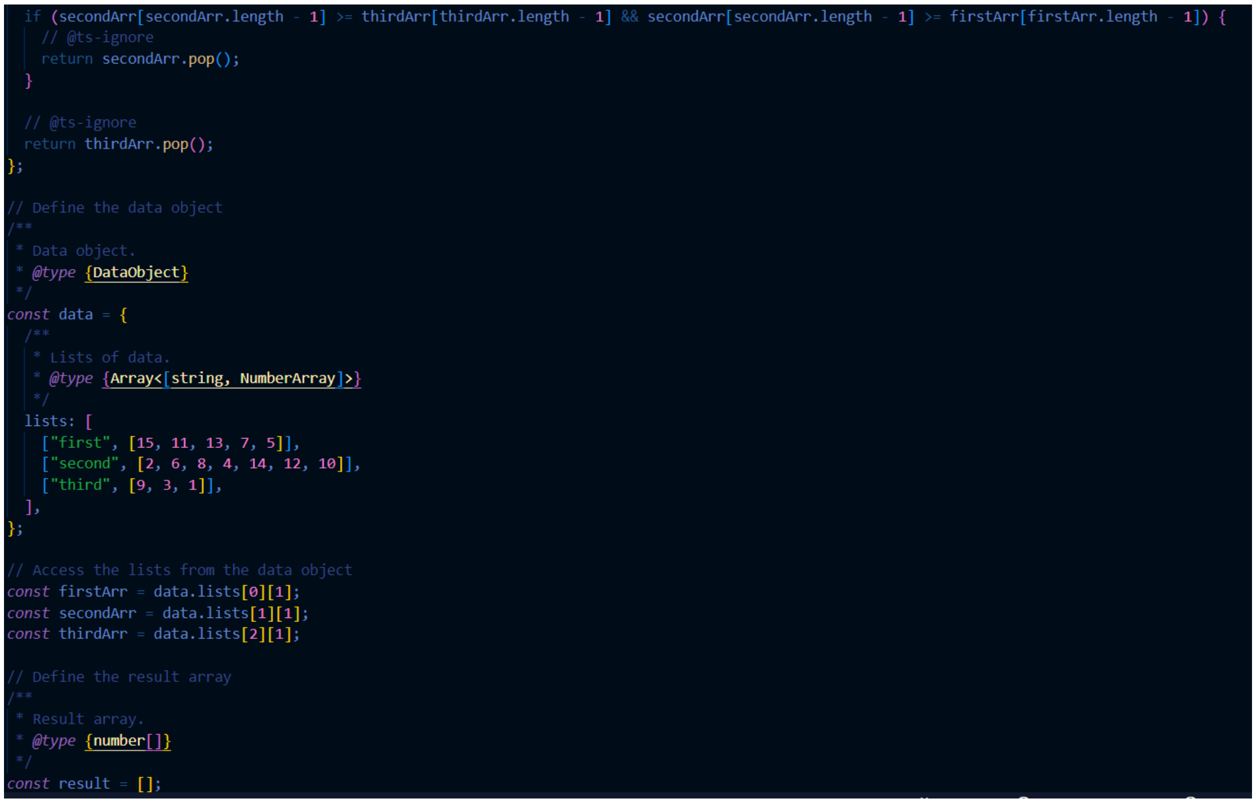
_

---

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



```
// @ts-check

/**
 * Represents an array of numbers.
 * @typedef {number[]} NumberArray
 */

/**
 * Represents a data object with lists.
 * @typedef {Object} DataObject
 * @property {Array<[string, NumberArray]>} lists - The lists of data.
 */

/**
 * Extracts the biggest number from the lists.
 * @returns {number} The biggest number from the lists.
 */
const extractBiggest = () => {
  if (firstArr.length === 0) {
    // @ts-ignore
    return secondArr.length === 0 ? thirdArr.pop() : secondArr.pop();
  }

  if (secondArr.length === 0) {
    // @ts-ignore
    return firstArr.length === 0 ? thirdArr.pop() : firstArr.pop();
  }

  if (thirdArr.length === 0) {
    // @ts-ignore
    return firstArr.length === 0 ? secondArr.pop() : firstArr.pop();
  }

  if (firstArr[firstArr.length - 1] >= secondArr[secondArr.length - 1] && firstArr[firstArr.length - 1] >= thirdArr[thirdArr.length - 1]) {
    // @ts-ignore
    return firstArr.pop();
  }
```



```
  if (secondArr[secondArr.length - 1] >= thirdArr[thirdArr.length - 1] && secondArr[secondArr.length - 1] >= firstArr[firstArr.length - 1]) {
    // @ts-ignore
    return secondArr.pop();
  }

  // @ts-ignore
  return thirdArr.pop();
};

// Define the data object
/**
 * Data object.
 * @type {DataObject}
 */
const data = {
  /**
   * Lists of data.
   * @type {Array<[string, NumberArray]>}
   */
  lists: [
    ["first", [15, 11, 13, 7, 5]],
    ["second", [2, 6, 8, 4, 14, 12, 10]],
    ["third", [9, 3, 1]],
  ],
};

// Access the lists from the data object
const firstArr = data.lists[0][1];
const secondArr = data.lists[1][1];
const thirdArr = data.lists[2][1];

// Define the result array
/**
 * Result array.
 * @type {number[]}
 */
const result = [];
```