Documentation Tool: JSDoc

1. Introduction

Documentation is an essential component in software development, particularly for React Native apps where components, functions, and hooks interact across multiple screens. JSDoc is a widely-used tool for generating documentation directly from JavaScript or TypeScript source code.

By using JSDoc, we can maintain clear, structured, and up-to-date documentation that helps:

- Onboard new developers quickly
- Maintain consistency across codebase
- Generate interactive documentation for APIs and modules

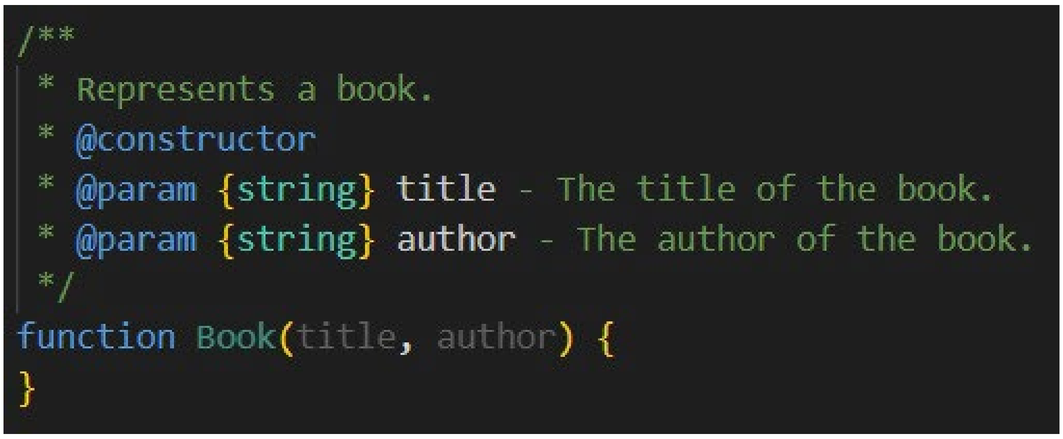
2. What is JSDoc

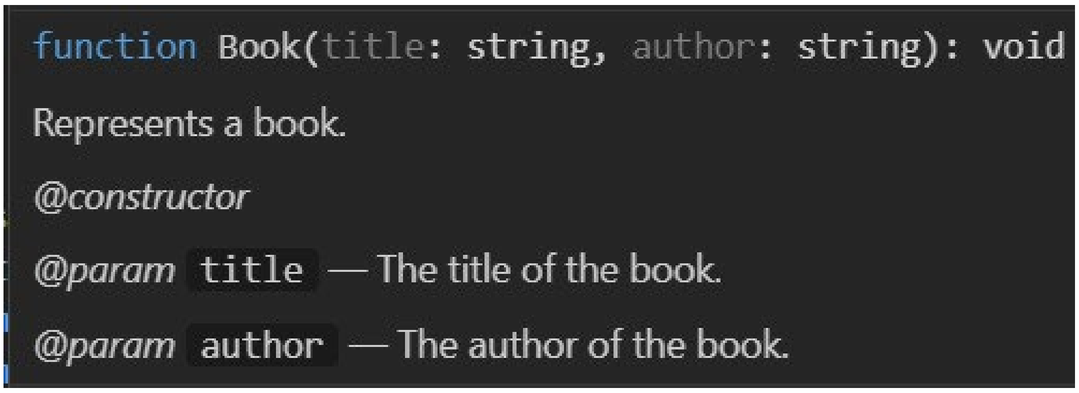
JSDoc is an API documentation generator for JavaScript, similar to Javadoc or phpDocumentor that parses specially formatted comments in code to produce structured documentation in HTML or Markdown.

It supports:

- Functions, methods, and parameters
- Classes and inheritance
- Modules and namespaces
- Custom types using **`@typedef`**

Example:


```
/**
 * Represents a book.
 * @constructor
 * @param {string} title - The title of the book.
 * @param {string} author - The author of the book.
 */
function Book(title, author) {
}
```


```
function Book(title: string, author: string): void
```
Represents a book.

@constructor

@param `title` — The title of the book.

@param `author` — The author of the book.

3. Installation

Step 1: Initialize Project and Install JSDoc

First, initialize Node.js project and install JSDoc as a development dependency:

```
npm init -y
```

```
npm i -D jsdoc
```

Key Points:

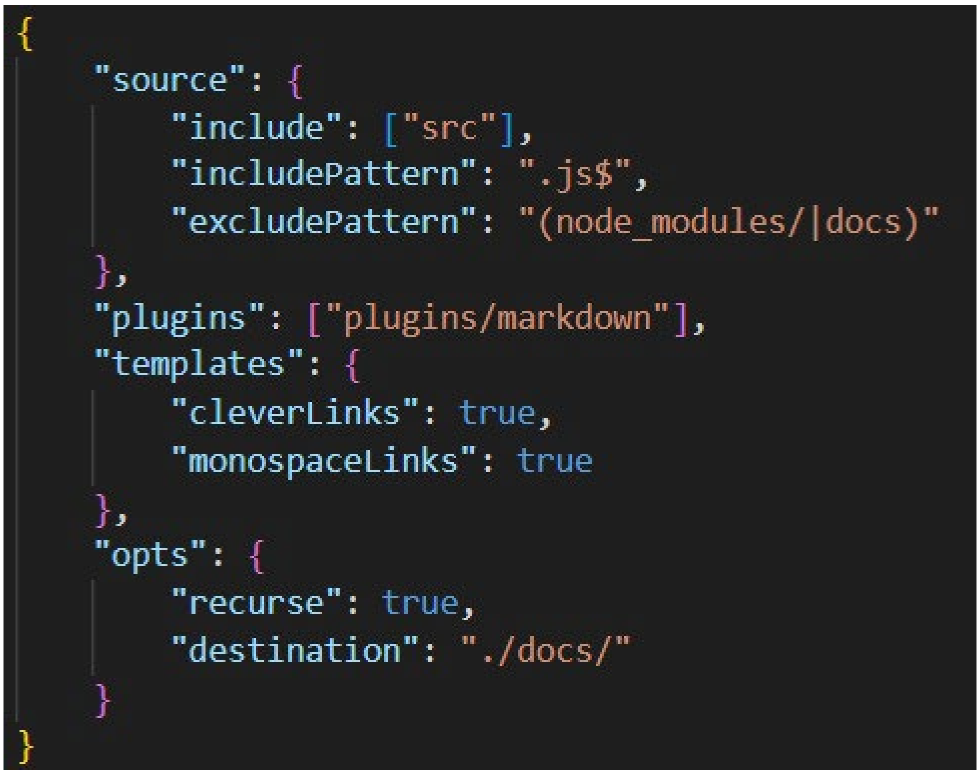
- `npm` - Node Package Manager, used to install and manage JavaScript packages.
- `i` - Short for `install`. It tells `npm` to install a package.
- `-D` - Short for `--save-dev`. It indicates that the package is a development dependency.
  - Development dependencies are tools needed only during development (like JSDoc, linters, testing frameworks) and not required in production.
  - They are stored under "devDependencies" in package.json.

This ensures JSDoc is available for generating documentation without affecting production dependencies.

Step 2: Configure JSDoc

Create a jsdoc.json configuration file in the project root. This file defines which files to include, plugins to use, template options, and the destination folder for the generated documentation.

Example jsdoc.json:



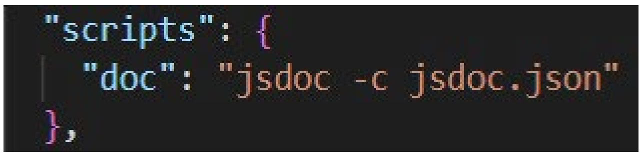
```json
{
    "source": {
        "include": ["src"],
        "includePattern": ".js$",
        "excludePattern": "(node_modules/|docs)"
    },
    "plugins": ["plugins/markdown"],
    "templates": {
        "cleverLinks": true,
        "monospaceLinks": true
    },
    "opts": {
        "recurse": true,
        "destination": "./docs/"
    }
}
```

Key Points:

- **`include`** - Specifies the folder(s) containing source files (src).
- **`includePattern`** - Filters files with .js extension.
- **`excludePattern`** - Excludes unwanted directories (node_modules and docs).
- **`plugins`** - Enables Markdown support.
- **`templates`** - Configures link styles and monospace formatting.
- **`opts.destination`** - Folder where HTML documentation will be generated.
- **`opts.recurse`** - Ensures subdirectories are included automatically.

Step 3: Update package.json Scripts

Add a doc script to simplify documentation generation:



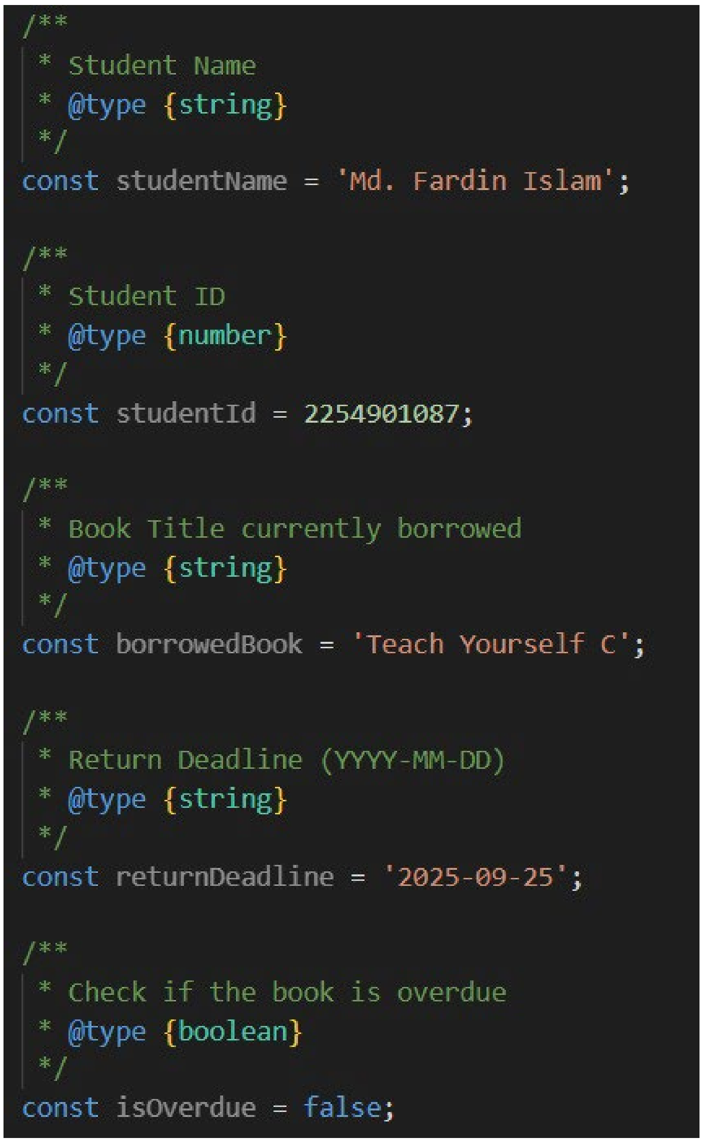
```
"scripts": {
  "doc": "jsdoc -c jsdoc.json"
},
```

This allows you to generate documentation with a single command:

```
npm run doc
```

Step 4: Prepare Source Files

Ensure project source files are located in the src folder and include JSDoc comments. For example, create index.js inside src:



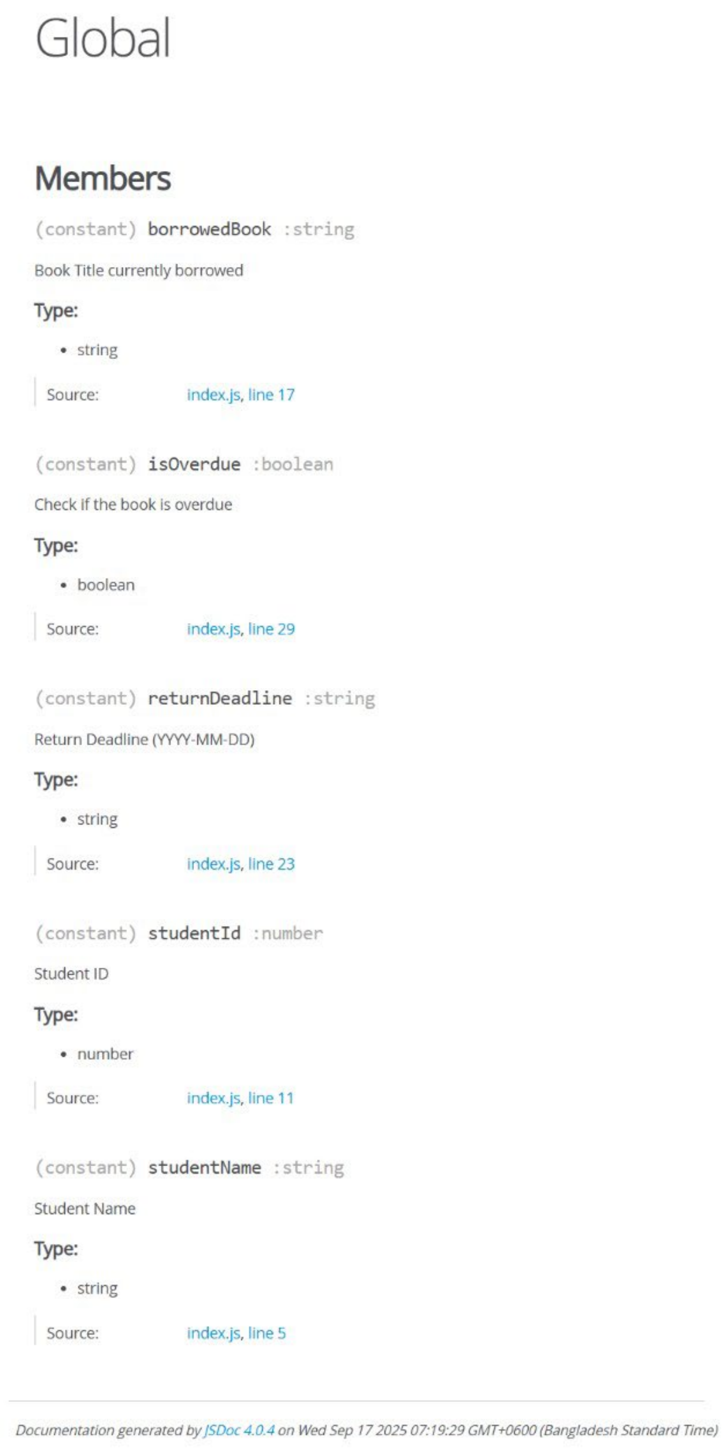
```
/**
 * Student Name
 * @type {string}
 */
const studentName = 'Md. Fardin Islam';

/**
 * Student ID
 * @type {number}
 */
const studentId = 2254901087;

/**
 * Book Title currently borrowed
 * @type {string}
 */
const borrowedBook = 'Teach Yourself C';

/**
 * Return Deadline (YYYY-MM-DD)
 * @type {string}
 */
const returnDeadline = '2025-09-25';

/**
 * Check if the book is overdue
 * @type {boolean}
 */
const isOverdue = false;
```

Step 5: Generate Documentation

Run the doc script to create HTML documentation:

```
npm run doc
```

The documentation will be generated in the docs/ folder as specified in jsdoc.json. docs/index.html can be opened in a browser to view the formatted documentation.



Global

Members

(constant) borrowedBook :string

Book Title currently borrowed

Type:

- string

Source: index.js, line 17

(constant) isOverdue :boolean

Check if the book is overdue

Type:

- boolean

Source: index.js, line 29

(constant) returnDeadline :string

Return Deadline (YYYY-MM-DD)

Type:

- string

Source: index.js, line 23

(constant) studentId :number

Student ID

Type:

- number

Source: index.js, line 11

(constant) studentName :string

Student Name

Type:

- string

Source: index.js, line 5

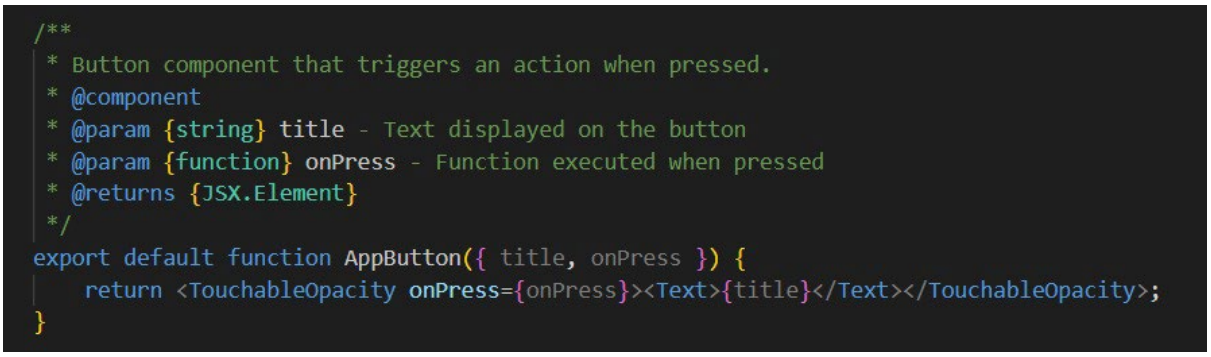
Documentation generated by JSDoc 4.0.4 on Wed Sep 17 2025 07:19:29 GMT+0600 (Bangladesh Standard Time)

4. Usage Guidelines

4.1 Comment Syntax

- Use block comments starting with `/**`
- Begin each line with `*`
- Include a description and appropriate tags

React Native Component Example:



```
/**
 * Button component that triggers an action when pressed.
 * @component
 * @param {string} title - Text displayed on the button
 * @param {function} onPress - Function executed when pressed
 * @returns {JSX.Element}
 */
export default function AppButton({ title, onPress }) {
    return <TouchableOpacity onPress={onPress}><Text>{title}</Text></TouchableOpacity>;
}
```

4.2 Common Tags

Tag	**Description**
`@param`	Describes function parameters or props
`@returns`	Describes return value or JSX element
`@component`	Marks React components
`@example`	Provides a usage example
`@typedef`	Defines custom types
`@deprecated`	Marks outdated code
`@see`	Links to related functions or modules

5. Advantages

- **Maintains documentation close to the code:** Embedding documentation directly in source files ensures it remains consistent with code changes and improves readability.
- **Automatically generates HTML/Markdown documentation:** The JSDoc CLI can produce structured, web-friendly documentation, saving time and effort.
- **Improves onboarding and collaboration:** Clear documentation helps new developers understand project structure, functions, and components faster.

- **Provides type hints and validation for JavaScript:** JSDoc allows specifying types using tags like `@type` without requiring TypeScript, providing “TypeScript-lite” benefits for editors and IDEs.
- **Integrates with IDEs and IntelliSense:** Modern IDEs (e.g., VS Code) use JSDoc comments to provide autocompletion, inline documentation, and tooltips.
- **Supports templates and plugins for customization:** Documentation styling and formatting can be enhanced using templates (like Minami) and plugins.
- **Easy to understand and implement:** JSDoc relies only on comments and intuitive tags, making it simple for developers to add or update documentation without extra tools or compilers.
- **Type safety without TypeScript:** Provides optional type hints in plain JavaScript, helping catch potential errors early and improving editor suggestions.

6. Limitations

- **Manual maintenance required:** Comments must be updated whenever code changes to ensure the generated documentation is accurate.
- **Dynamic or runtime-generated code may be missed:** JSDoc cannot fully document code that is created or modified at runtime.
- **Learning curve for new developers:** Developers unfamiliar with JSDoc syntax may need guidance to use it effectively.
- **Default templates may be basic:** HTML output may look simple unless customized with third-party templates or plugins.
- **Cannot enforce strict type safety like TypeScript:** While JSDoc provides type hints, it does not perform compile-time checks.
- **Potential for redundancy:** Some comments may duplicate information already evident from code, requiring balance between conciseness and completeness.

7. Best Practices

- Document all components, screens, hooks, and utility functions
- Consistently use JSDoc tags (`@param`, `@returns`, `@component`)
- Keep jsdoc.json at project root for uniform configuration
- Use modern templates for visually appealing HTML docs
- Include examples with `@example` for complex components
- Generate documentation regularly, e.g., during CI/CD or before releases

8. Conclusion

JSDoc is a reliable and efficient tool for documenting JavaScript code, particularly in React Native projects. By using JSDoc:

- Documentation remains consistent with the code
- Developer onboarding is simplified
- Maintenance and scaling of the project are easier
- Code readability and collaboration are enhanced

Proper adherence to best practices ensures that your documentation is clear, up-to-date, and useful for the entire development team.

9. References

- JSDoc Official Website: https://jsdoc.app
- JSDoc GitHub Repository: https://github.com/jsdoc/jsdoc
- JSDoc npm Package: https://www.npmjs.com/package/jsdoc
- Traversy Media, JSDoc Tutorial for Beginners, YouTube.