JSDoc Documentation

Author: Akila Nipo

Introduction

JSDoc is a documentation generator for JavaScript code.It provides a structured way to document code and generate API documentation. It helps developers understand how to use and integrate various parts of the codebase through descriptive comments and examples. JSDoc can be used for documenting functions, parameters, return values, components, and custom types, improving code readability and maintainability.

Why to choose JSDOC?

JSDoc vs. Docusaurus

Aspect	JSDoc	Docusaurus
Documentation Type	Automated - Dynamic	Manual - Static
Ideal Use Case	API and code documentation	Project guides, tutorials, user manuals
Updating	Automatically updates with code	Requires manual updates

Summary: Choose **JSDoc for code-level** documentation and Docusaurus for project-level documentation.

JSDoc vs. ESDoc

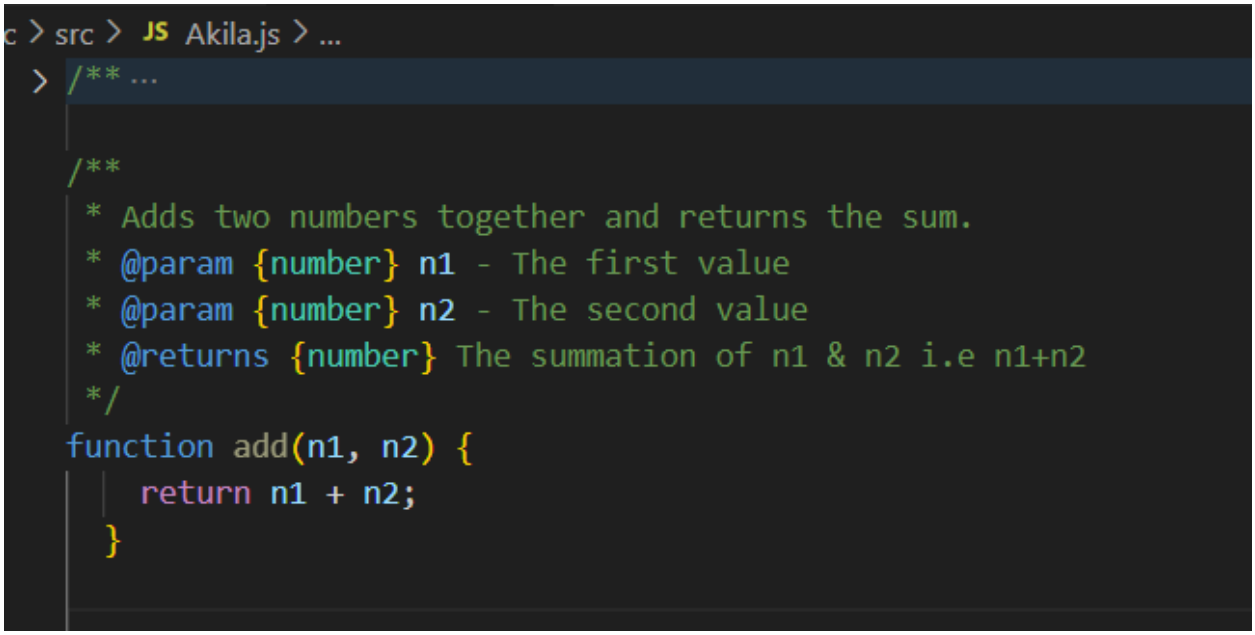
Aspect	JSDoc	ESDoc
Community Support	Strong, frequent updates	Smaller community, less frequent updates
Plugins & Extensibility	Highly extensible with a large ecosystem	Limited plugin support, less customizable

Summary: **JSDoc has a larger community and greater extensibility**, making it a more versatile choice than ESDoc.

Basic Syntax

JSDoc comments are written above functions, methods, classes, or any other code blocks you want to document. They start with `/**` and end with `*/`.

- `Documenting a Function`
  JSDoc allows you to document functions, including their parameters, return types, and a description of what the function does. Here's an example:



- `Fetch Data from an API`
  The example shows how to use the fetchData function to fetch data from an `API`

and handle both successful responses and errors.

- **See More: Click Here**

Tags and Types

Tag	Description
`@param`	Describes a function parameter, including its name and data type, and provides a brief description.
`@returns`	Describes the return value of a function, including its data type and a brief description of what it returns.
`@example`	Provides a code example demonstrating how to use the function or feature being documented.

`@deprecated`	Indicates that a function or feature is deprecated, typically including information about what to use instead.
`@typedef`	Defines a custom type or object schema, documenting its properties and their types.

Installation

- `Prerequsite`
  Before installing JSDoc, make sure Node.js is installed on your machine.
  Installation Link: `Click Here`

To install JSDoc, one can choose between global and local installation:

- `Global Installation`
  To install JSDoc globally on local machine, use the following command. This allows you to run JSDoc from any directory.

```
npm install -g jsdoc
```

- `Local Installation`
  To install JSDoc locally within project, use this command. This ensures it's only available in the project where it's installed.

```
npm install --save-dev jsdoc
```

Generating Documentation

Once installed, you can generate documentation by running JSDocs on your source files.
For more: Click.

- `For Single File`

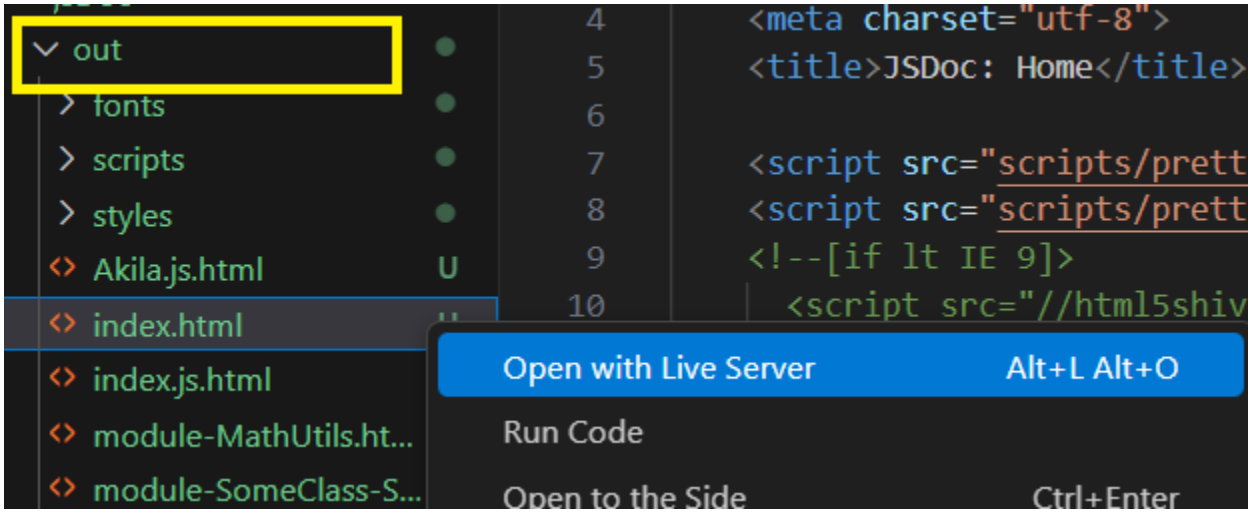
```
jsdoc index.js
```

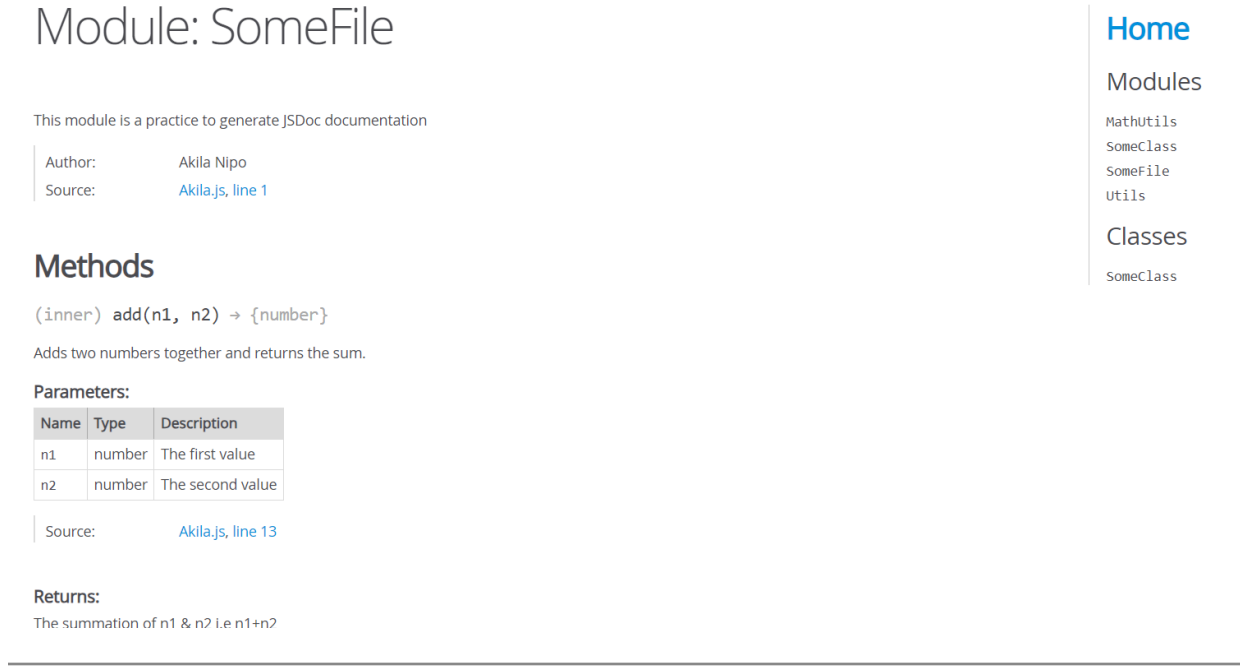
- `For All Files in a Directory`

```
jsdoc src -r
```

Generating Documentation(cont.)

- The generated documentation will be placed in an `out` folder by default.
- Open index.html inside this folder to view the documentation in your web browser._

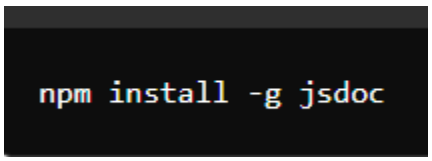




A Quick Guide to Generate Documentation

Step 1: Set Up Your Environment

- Make sure Node.js is installed on your system. You can download it from nodejs.org.
- Install `jsdoc` globally by running the following command:



Step 2: Create Your JavaScript File

- At first, create a JavaScript file named `Akila.js` and add the following code:

```
/**
 * @author Akila Nipo
 */

/**
 * Adds two numbers together and returns the sum.
 * @param {number} n1 - The first value
 * @param {number} n2 - The second value
 * @returns {number} The summation of n1 & n2 i.e n1+n2
 */
function add(n1, n2) {
    return n1 + n2;
}
```

JS Akila.js U

jsDoc > src > JS Akila.js > ...

```
/**

 * @author Akila Nipo
 */

/**
 * Adds two numbers together and returns the sum.
 * @param {number} n1 - The first value
 * @param {number} n2 - The second value
 * @returns {number} The summation of n1 & n2 i.e n1+n2
 */
function add(n1, n2) {
    return n1 + n2;
 }
```

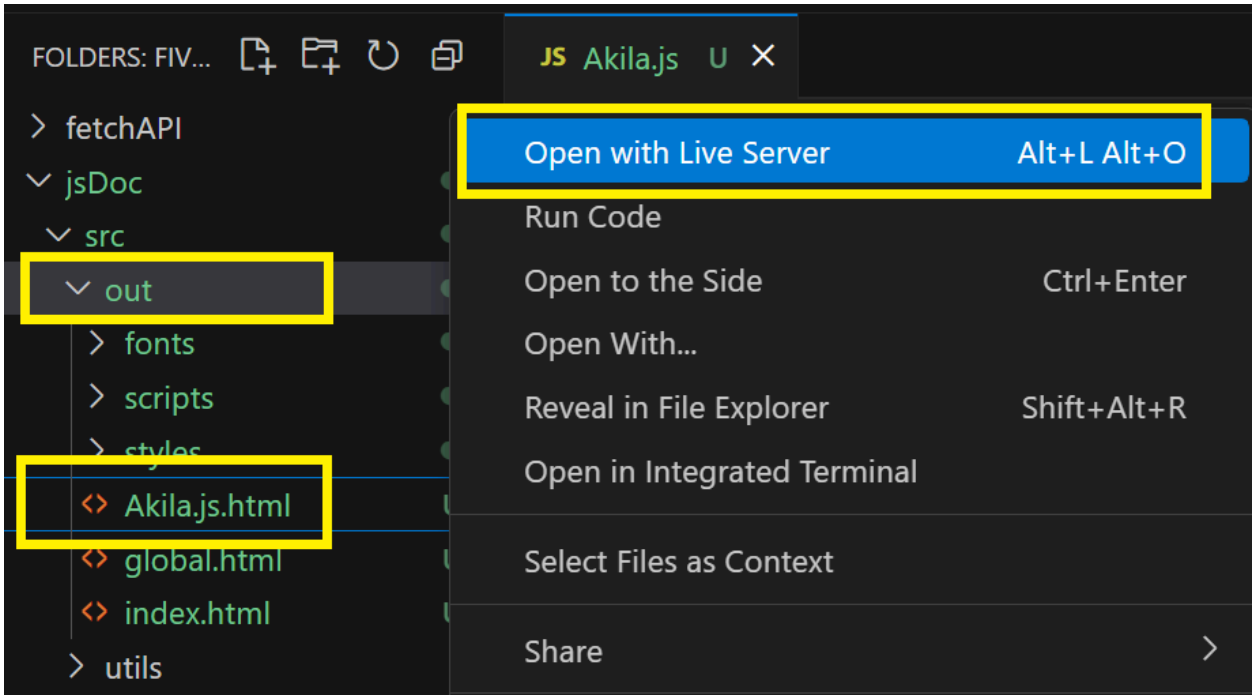
Step 3: Generate Documentation Using JSDoc

- For All Files in a Directory

```
jsdoc src -r
```

Step 4: Locate the Generated HTML File

- After running the command, an `out` folder will be created containing an HTML file named `Akila.js.html`.



Step 5: Open the HTML File with Live Server

- Open the `Akila.js.html` file using Live Server to view the documentation.

Some JSDoc Plugins

---

i) JSDoc Default Plugins:

- Markdown Support : Converts JSDoc *comments into Markdown*, making documentation easy to read and share, especially on platforms like *GitHub.* Click jsdoc/plugin-markdown
- JSDoc GitHub Action: JSDoc GitHub Action

---

ii) Community Plugins:

- JSDoc-to-Markdown: - Converts JSDoc comments into `Markdown files`, which can be used to generate `README files` or other documentation formats. Click jsdoc-to-markdown

---

iii) Templates:

JSDoc provides a range of customizable templates that enhance documentation quality and usability, improving readability and user experience for API documentation.

- JaguarJS
- DocStrap
- JSDoc3Template
- Minami
- Docdash
- TUI JSDoc Template
- Better-Docs
- More JSDoc Templates

References:Click Here