Documentation Tool: JSDoc

Introduction

1. What is JSDoc?

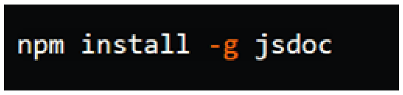
JSDoc is an open-source documentation generation tool that simplifies the process of creating and maintaining documentation for JavaScript projects, libraries, APIs, or any other JavaScript-based documentation needs. It's widely favored for its flexibility, extensibility, and compatibility with modern JavaScript features including ES6+ syntax.

For more info CLICK HERE.

Installation & Usage

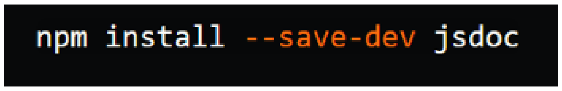
1. How to install JSDoc?

Install JSDoc globally using this command:



```
npm install -g jsdoc
```

Or use the following command to install it for a single project:



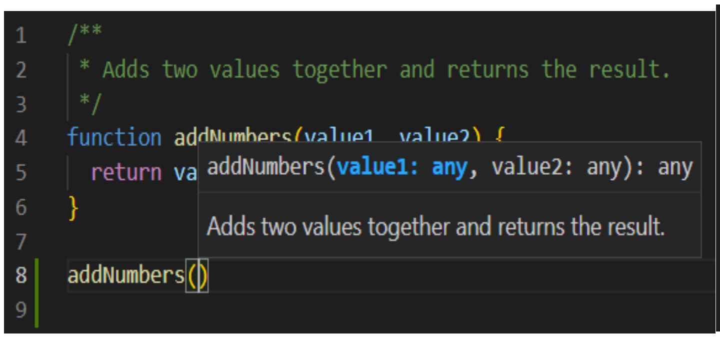
```
npm install --save-dev jsdoc
```

2. How to make JSDoc documentation?

To start documenting code, a comment has to be added starting with /** over each block of code the user wants to document: Modules, methods, classes, functions, etc.

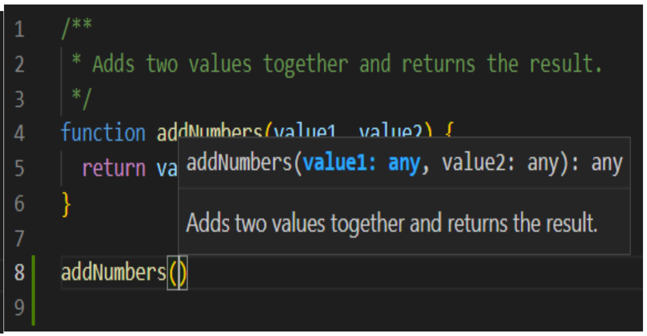
For more info CLICK HERE.

Here are some sample code:



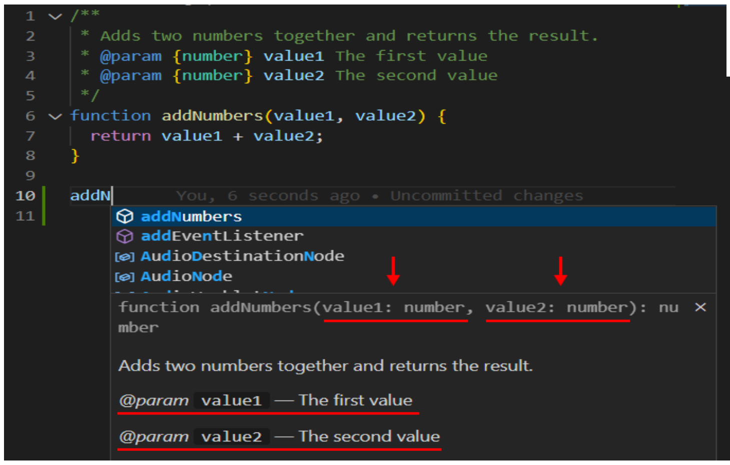
```
/**
 * Adds two values together and returns the result.
 */
function addNumbers(value1, value2) {
  return va
}

addNumbers()
```

```
addNumbers(value1: any, value2: any): any

Adds two values together and returns the result.
```



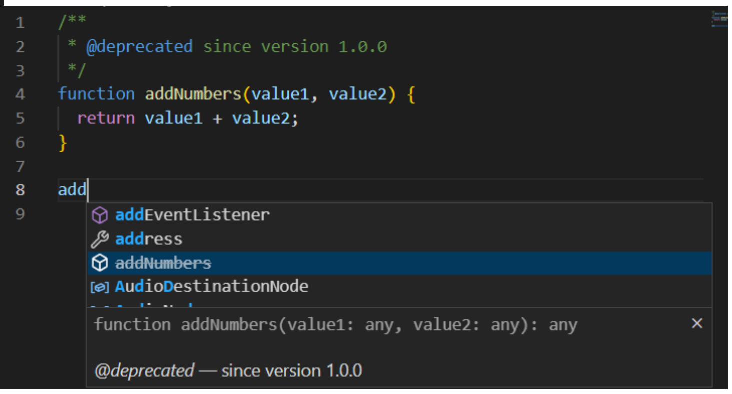
```
/**
 * Adds two values together and returns the result.
 */
function addNumbers(value1, value2) {
  return va
}

addNumbers()
```

```
addNumbers(value1: any, value2: any): any

Adds two values together and returns the result.
```



```
/**
 * Adds two numbers together and returns the result.
 * @param {number} value1 The first value
 * @param {number} value2 The second value
 */
function addNumbers(value1, value2) {
  return value1 + value2;
}

addN      You, 6 seconds ago • Uncommitted changes
```

```
function addNumbers(value1: number, value2: number): nu
mber

Adds two numbers together and returns the result.

@param value1 — The first value

@param value2 — The second value
```



```
/**
 * @deprecated since version 1.0.0
 */
function addNumbers(value1, value2) {
  return value1 + value2;
}

add
```

```
function addNumbers(value1: any, value2: any): any

@deprecated — since version 1.0.0
```

Remember, the more info you add to your comments, the more detailed your API documentation will be. But also, find the amount of detail that feels right to you. It's better to have all your code documented with only a few tags than to have only a few methods fully documented using all the tags because it was too much and you couldn't keep up.

Export

After adding the comments all that's left to do is generate your documentation website:

Export files or folders

Simply call jsdoc and add the path to the file or folder.

```
jsdoc path/to/my/file.js
```

Tags and Types

JSDoc supports a variety of tags to document different aspects of your code. Some common tags include:

- @param: Documents a function parameter.
- @returns: Documents the return value of a function.
- @type: Specifies the data type of a variable.
- @description: Provides a detailed description of the code.
- @example: Includes example code.
- @see: Adds references to other resources or related documentation.

Advantages & Disadvantages of JSDoc

JSDoc offers advantages like adding type information and documentation directly into JavaScript code via comments, improving readability and tooling without a separate build step, but its disadvantages include being more verbose than TypeScript, the possibility of type inaccuracies, and potential performance overhead from runtime checks.

Advantages

- **Enhanced Readability and Maintainability:** JSDoc comments provide clear documentation within the code itself, making it easier to understand and maintain.
- **Improved Tooling and Editor Support:** Because type information is embedded in comments, IDEs and other development tools can offer better code completion, type checking, and error detection.
- **No Additional Build Step Required:** Unlike TypeScript or other languages that compile to JavaScript, JSDoc works directly with JavaScript comments, meaning no extra build process is necessary.
- **Type Safety and Error Prevention:** JSDoc enables static typing, which helps catch potential errors during development rather than at runtime, leading to more robust code.

- **Gradual Adoption:** Developers can incrementally adopt JSDoc into existing JavaScript projects, adding types where needed without a complete overhaul.

Disadvantages

- **Verbosity:** Adding detailed type information via comments can make the code more verbose and clutter the code for some developers.
- **Potential for Inaccuracy:** Type information in JSDoc is only as accurate as the comments themselves; if comments are not updated, they can become misleading or incorrect.
- **No True Static Typing:** JSDoc provides type information for tools but doesn't offer the same level of true static analysis and compile-time error catching as a compiled language like TypeScript.
- **Performance Overhead:** While JSDoc itself doesn't add a build step, any runtime validation or type enforcement performed based on the JSDoc comments can introduce performance overhead.
- **Syntax Differences:** Some developers may find the syntax for adding types in JSDoc less intuitive or more cumbersome compared to languages designed with types from the ground up.

Despite these limitations, JSDoc remains a powerful and widely used documentation tool in the software development community.