Full setup and workflow for adding JSDoc/TSDoc and actually seeing the benefits in VS Code. We'll cover both in-editor hover docs and optional generated docs.

Step 1. Add JSDoc/TSDoc Comments

Add JSDOC or TSDOC comments to your code:

```
import { create } from 'zustand';

interface AuthState {
  /**
   * Whether the user is currently logged in.
   */
  isLoggedIn: boolean;

  /**
   * Marks the user as logged in.
   *
   * Side effect:
   * - Stores `"true"` in `localStorage` under the key
`"isLoggedIn"`.
   */
  login: () => void;

  /**
   * Marks the user as logged out.
   *
   * Side effect:
   * - Removes the `"isLoggedIn"` item from `localStorage`.
   */
  logout: () => void;
}

/**
 * Zustand store for authentication state.
 *
 * - Initializes `isLoggedIn` from `localStorage`.
 * - Provides `login` and `logout` methods to update both
 *   the store state and `localStorage`.
 *
 * @example
 * ```tsx
 * const { isLoggedIn, login, logout } = useAuthStore();
 *
 * if (!isLoggedIn) {
 *   login();
 * }
 * ```
 */
export const useAuthStore = create<AuthState>((set) => ({
  isLoggedIn: localStorage.getItem('isLoggedIn') ===
'true',
  login: () => {
    localStorage.setItem('isLoggedIn', 'true');
    set({ isLoggedIn: true });
  },
  logout: () => {
    localStorage.removeItem('isLoggedIn');
    set({ isLoggedIn: false });
  },
}));
```

Step 2. Make Sure VS Code is Configured

1. **Open your project in VS Code**.
2. Ensure you have the **TypeScript extension** (built-in).

If using plain JS, enable `"checkJs": true` in your `tsconfig.json` or `jsconfig.json`.

```
{

  "compilerOptions": {
    "checkJs": true
  }
}
```

This makes VS Code parse JSDoc in JS files.

3. For TypeScript (`.ts`/`.tsx`), nothing extra is needed — TSDoc is supported by default.

Step 3. See the Docs in Action

- Hover your mouse over `useAuthStore` → You'll see the block comment appear in a tooltip.
- Hover over `login` or `logout` → You'll see their descriptions.
- Start typing `useAuthStore().` → IntelliSense autocompletion will show each method with your description.

Step 4. Generate Documentation (Optional)

If you want full **HTML/Markdown documentation**:

For TypeScript (TSDoc → TypeDoc)

```
npm install --save-dev typedoc
npx typedoc src/index.ts
```

- Output will be in a `docs/` folder (by default).
- Open `docs/index.html` in a browser to view it.

For JavaScript (JSDoc)

```
npm install --save-dev jsdoc
npx jsdoc -c jsdoc.json
```

(`jsdoc.json` config defines source files & output folder).

Step 5. Extra Tags You Can Use

- `@example` → show usage
  - JSDoc/TSDoc comments always attach to the very next declaration such as useAuthStore - (function, class, variable, etc.).
- `@remarks` → extra notes
- `@see` → cross-reference another function/class

- `@deprecated` → adds a warning in VS Code

Example:

```
/**
 * @deprecated Use `newLogin` instead.
 */
login: () => void;
```

Result:

- Hovering in VS Code → shows your descriptions.
- Autocomplete → shows contextual info.
- Optional → run TypeDoc/JSDoc to get a full docs website.

Run TypeDoc

1. Running TypeDoc (for TypeScript)

Install:

```
npm install --save-dev typedoc
```

Run:

```
npx typedoc src/index.ts
```

This will generate docs into a `docs/` folder (by default). Open `docs/index.html` in a browser.

If you want Markdown output instead of HTML:

```
npx typedoc --plugin typedoc-plugin-markdown src/index.ts
```

2. Running JSDoc (for JavaScript)

Install:

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

Run:

```
npx jsdoc src --destination docs
```

This scans your `.js` files in `src/` and generates HTML docs inside a `docs/` folder.

Open `docs/index.html` to view them.

If you want more control, create a `jsdoc.json` config:

```
{
  "source": {
    "include": ["src"]
  },
  "opts": {
    "destination": "docs"
  }
}
```

Then run:

```
npx jsdoc -c jsdoc.json
```

3. Demo of IntelliSense / Autocomplete in VS Code

Imagine you have this code in your project:

```
const { isLoggedIn, login, logout } = useAuthStore();

login();
```

When you type `useAuthStore().` → autocomplete popup:

Before docs:

```
login(): void
logout(): void
isLoggedIn: boolean
```

After adding TSDoc/JSDoc:

```
login(): void
```

```
   Marks the user as logged in.
   Side effect: Stores "true" in localStorage.

logout(): void
   Marks the user as logged out.
   Side effect: Removes "isLoggedIn" from localStorage.

isLoggedIn: boolean
   Whether the user is currently logged in.
```

4. Demo of Hover Docs

When you hover over `login`, you'll see:

Before docs:

```
(property) login: () => void
```

After docs:

```
(property) login: () => void

Marks the user as logged in.

Side effect:
- Stores "true" in localStorage under the key "isLoggedIn".
```

So the workflow is:

1. Add JSDoc/TSDoc comments.
2. Run `typedoc` (TS) or `jsdoc` (JS) to generate static docs (optional).
3. In VS Code, hover/autocomplete will show the docs automatically.