Markdown File in Code:

~~~~
# Sample Markdown File

This is a Markdown file that can be used to provide documentation for your code. You can include headings, lists, code blocks, and more.

## Code Example

```javascript

function add(a, b) {

  return a + b;

}

You can reference this Markdown file in your code's documentation or README to explain concepts and provide usage instructions.
~~~~

vbnet

In your code's README or documentation, you can include the Markdown file like this:

~~~~
```markdown

# My Project Documentation

## Usage

To add two numbers in JavaScript, you can use the `add` function defined in the [Sample Markdown File](sample.md) like this:

```javascript

const result = add(3, 4);

console.log(result); // 7

JSDoc Comments in Code:

javascript

Copy code

/**

 * Adds two numbers.

 *

 * @param {number} a - The first number to add.

 * @param {number} b - The second number to add.

 * @returns {number} The sum of the two numbers.

 */
~~~~

```
function add(a, b) {
  return a + b;
}
```

JSDoc comments are used to provide type annotations and documentation for JavaScript functions, variables, and classes. They help improve code readability and provide valuable information for developers and IDEs.

@ts-check Annotation in Code:

javascript

Copy code

```
// @ts-check

/**
 * Adds two numbers.
 *
 * @param {number} a - The first number to add.
 * @param {number} b - The second number to add.
 * @returns {number} The sum of the two numbers.
 */
function add(a, b) {
  return a + b;
}

const result = add(3, "4"); // This will trigger a type-checking error
console.log(result);
```

The @ts-check annotation is used in a JavaScript file to enable TypeScript type-checking in that file. It helps catch type errors at compile-time, providing better code quality and reducing runtime errors. In the example above, passing a string instead of a number will trigger a type-checking error.