GUIDA ALLA DOCUMENTAZIONE DEL CODICE CON JSDODC O TSDOC E TYPEDOC

Sommario

1. INTRODUZIONE

La documentazione del codice è essenziale per mantenere il progetto chiaro, leggibile, scalabile e collaborativo. Serve a:
- Capire cosa fa il codice (anche dopo mesi)
- Facilitare il lavoro di altri sviluppatori
- Aiutare la manutenzione e l’estensione del progetto
- Migliorare l'inserimento di nuovi dev

2. QUANDO E Perché DOCUMENTARE

- Durante lo sviluppo: documenta mentre scrivi il codice.
- Durante refactor: aggiorna i commenti insieme al codice.
- Durante la revisione: la documentazione aiuta a capire lo scopo del codice.

3. JSDOC

3.1 COS'È

JSDoc è una convenzione di commenti + uno strumento che consente di documentare codice JavaScript tramite commenti speciali.

3.2 SINTASSI E TAG PRINCIPALI

Esempio base:
```js
/**
 * Calcola la somma di due numeri.
 * @param {number} a - Primo numero
 * @param {number} b - Secondo numero
 * @returns {number} Somma di a e b
```

```
 */
function sum(a, b) {
  return a + b;
}
```

Tag comuni:
- @param – documenta i parametri
- @returns – descrive il valore restituito
- @example – mostra un esempio d’uso
- @typedef, @property – per oggetti complessi
- @class, @constructor – per classi

3.3 BEST PRACTICE

- Commenta mentre sviluppi
- Aggiorna sempre i commenti insieme al codice
- Usa una struttura coerente
- Scrivi in modo chiaro e conciso
- Includi esempi reali
- Documenta anche scelte progettuali e limiti noti

3.4 ESEMPI

Funzione:
```
/**
 * Moltiplica due numeri.
 * @param {number} x
 * @param {number} y
 * @returns {number}
 */
function multiply(x, y) {
  return x * y;
}
```

Classe:
```
/**
 * Rappresenta una persona.
 */
class Person {
  /**
   * @param {string} name
   * @param {number} age
   */
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
```

```
  /**
   * Ritorna una stringa con i dettagli.
   * @returns {string}
   */
  describe() {
    return `${this.name} is ${this.age} years old.`;
  }
}
```

4. TSDOC + TYPEDOC

4.1 COS'È

- TSDoc è uno standard per scrivere commenti documentativi in progetti TypeScript.

- TypeDoc è il tool che genera la documentazione leggibile (in HTML o Markdown) da file TypeScript documentati con commenti TSDoc.

La differenza: TSDoc è lo standard di commento, TypeDoc è lo strumento che lo interpreta.

4.2 SETUP TYPEDOC

npm install --save-dev typedoc

Crea un file typedoc.json a livello di root:

```
{
    "entryPoints": [
        "src/**/*.ts",
        "src/**/*.tsx",
        "src/**/*.jsx",
        "src/**/*.js"
        ],        // Entry points for your TypeScript/JavaScript files
    "out": "docs",                  // Output folder for generated docs
    "excludePrivate": true,         // Exclude private members from the documentation
    "excludeProtected": true,       // Exclude protected members
    "includeVersion": true,         // Include project version in the docs
    "readme": "README.md",          // Use your README as the landing page
    "tsconfig": "tsconfig.json"     // Reference your TypeScript config
  }
```

Nel package.json, aggiungi:

```
"scripts": {
  "docs": "typedoc"
}
```

4.3 ESEMPI CON TSDOC

```
import { Box } from "@mui/material";
import ApplicationsList from "../ApplicationsList";
import Grid from "../Grid";
import { useDataContext } from "../../context/useDataContext";

/**
 * `ApplicationsContainer` è un componente che gestisce la visualizzazione
congiunta
 * dell'elenco delle applicazioni e della griglia delle etichette associate.
 *
 * @remarks
 * Il componente utilizza il contesto `DataContext` per ottenere:
 * - l'elenco delle applicazioni (`options`)
 * - la funzione per aggiornare la selezione (`handleChange`)
 * - l'applicazione selezionata (`selectedOption`)
 *
 * Include:
 * - `<ApplicationsList />`: lista interattiva di applicazioni
 * - `<Grid />`: griglia dinamica aggiornata in base all'app selezionata
 *
 * @returns Un contenitore `<Box>` con i due componenti affiancati.
 *
 * @component
 */
const ApplicationsContainer = () => {
  const { options, handleChange, selectedOption } = useDataContext();

  return (
    <Box sx={{ display: "flex", flexGrow: 1, overflow: "auto" }}>
      <ApplicationsList options={options} handleChange={handleChange} />
      <Grid selectedApp={selectedOption} />
    </Box>
  );
};

export default ApplicationsContainer;
```

4.4 GENERAZIONE DELLA DOCUMENTAZIONE

Esegui:

npm run docs

Verrà generata una cartella /docs con la documentazione HTML

5. BEST PRACTICE GENERALI

- Non documentare l’ovvio: se qualcosa è autoesplicativa (es. isVisible: boolean), puoi anche evitare di duplicare informazione.
- Documenta l'intenzione, non solo il comportamento.
- Sii coerente nei tag e nel linguaggio.
- Includi esempi di uso.
- Evita commenti falsi o obsoleti.

6. CONCLUSIONE

Documentare è parte integrante dello sviluppo professionale. Con JSDoc per JS e TSDoc + TypeDoc per TS puoi creare documentazione coerente e utile anche per chi non ha scritto il codice.