# Práctica 4: Documentaciones

## Como documentar código JavaScript usando JSDoc

### Como instalar JSDoc

Para instalar JSDoc deberemos abrir la terminal dentro de vscode y ejecutar el siguiente comando:
npm i -D jsdoc

Una vez instalado editaremos el archivo llamado package.json y donde pone script pondremos lo siguiente:

```
"scripts": {
  "doc": "jsdoc -c jsdoc.json"
},
```

Para que siempre mire dentro del archivo jsdoc.json.

Ejecutaremos npm run doc en la terminal de vs code para generar la documentación por defecto.

Luego copiaremos la carpeta default que se encuentra en:

. /node_modules/jsdoc/templates/default

Y le cambiaremos el nombre al que queramos en mi caso custom-template.

Y en el archivo jsdoc.json añadiremos la línea template con la ruta del template dentro de opts:

```
"opts": {
    "recurse": true,
    "destination": "./docs/",
    "template": "./custom-template"
}
```

Una vez hecho esto ya tendremos todo listo para empezar a documentar nuestro código.

# Como comentar en JSDoc

Para comentar una variable haremos lo siguiente:

Abriremos con /** y en la segunda línea pondremos una descripción o el nombre de la variable, luego usaremos @type para indicar el tipo de la variable, en este caso string.

```
/**
 * Nombre
 * @type {string}
 */
const nombre = 'Daniel';
```

Para comentar un array, es muy similar a una variable solo que en el @type indicaremos que es un array y también el tipo de array, en este caso el array es de tipo number.

```
/**
 * Array de temperaturas
 * @type {Array<number>}
 */
const temperatura = [16, 18.5, 20, 24, 25];
```

Para comentar un objeto

Dentro de @type pondremos otro par de {} y indicaremos los parámetros y el tipo.

```
/**
 * Objeto persona
 * @type {{id: number, nombre: string}}
 */
const persona = {
    id: 1,
    nombre: 'Daniel'
};
```

Si queremos crear un tipo de objeto con propiedades definidas, con @typedef indicaremos que estamos definiendo un objeto llamado Alumno y luego con @property iremos indicando uno por uno los parámetros del objeto, si queremos que un parámetro sea opcional lo pondremos entre [], en mi caso edad es opcional.

```
/**
 * Alumno
 * @typedef {Object} Alumno
 * @property {number} id - Id del alumno
 * @property {string} nombre - Nombre del alumno
 * @property {string} apellidos - Apellidos del alumno
 * @property {number} [edad] - Edad del alumno
 */

/**
 * @type {Alumno}
 */
const alumno = {
    id: 1,
    nombre: 'Daniel',
    apellidos: 'Ceban',
    edad: 19
};
```

Para comentar una clase primero comentaremos la clase dando una breve explicación de lo que hace, luego comentaremos el constructor usando @param y dentro los parámetros de la clase Persona usando @property indicando el tipo del parámetro el nombre y una breve descripción. Para comentar una función también usaremos @property indicando que es una función el nombre de la función y una breve descripción, luego usaremos @return para indicar que es lo que devuelve la función en mi caso no devuelve nada por eso le indicare void. Al crear una persona podemos usar @See {@link Persona} para linkearlo a la clase Persona.

```
/**
 * Clase para crear un objeto persona
 */
class Persona {
    /**
     *
     * @param {Object} informacionPersona Informacion sobre la persona
     */
    constructor(informacionPersona) {
        /**
         * @property {string} nombre - Nombre de la persona
         */
        this.nombre = informacionPersona.nombre;
        /**
         * @property {number} edad - Edad de la persona
         */
        this.edad = informacionPersona.edad;
    }

    /**
     * @property {Function} mostrarDatos - Muestra los datos de la persona
     * @returns void
     */
    mostrarDatos() {
        console.log(`Nombre: ${this.nombre} Edad: ${this.edad}`);
    }
}

/**
 * Persona 1
 * See {@link Persona}
 */
const persona1 = new Persona({
    nombre: 'Daniel',
    edad: 19
});
```

Comentar un modulo, daremos una breve descripción y usando @module indicaremos el nombre del módulo.

Aquí volvemos a ver como comentar una función, en este caso la función suma dos números y devuelve el resultado de la suma.

```
/**
 * Módulo de la calculadora
 * @module calculadora
 */

/**
 * Suma dos números
 * @param {number} num1 - Primer número
 * @param {number} num2 - Segundo número
 * @returns {number} - Resultado de la suma
 */
exports.sumar = (num1, num2) => num1 + num2;
```

Para comentar un archivo, usaremos @file indicando el nombre del archivo y una breve descripción, usando @author podemos indicar el autor y @version para indicar la versión.

```
/**
 * @file calculadora.js contiene las operaciones de la calculadora
 * @author Daniel Ceban
 * @version 1.0
 */
```

Una vez finalizada la documentación volveremos a ejecutar npm run doc en la terminal de vscode dentro de nuestro proyecto para actualizar la documentación. Abriendo el archivo index.html que se encuentra dentro de la carpeta /docs nos llevara a la documentación.