Documentation with JSDoc & PHPDoc: Course & Best Practices

Introduction

Code documentation is essential to ensure maintainability, understanding, and to facilitate collaboration. Tools like JSDoc for JavaScript and PHPDoc for PHP allow you to write structured comments that help IDEs (like VSCode) provide hints, autocompletion, type checking, etc.

Part 1: JavaScript Documentation with JSDoc

Complete and Enhanced Example

```
"use strict";

/**
 * This function says "hello" to the given first name.
 *
 * @param {string} prenom The name to greet.
 * @returns {void} Returns nothing.
 */
function bonjour(prenom) {
    console.log("Bonjour " + prenom);
}

bonjour("Paul");
```

Documentation with Multiple Types, Return Values, and Possible Errors

```
/**
 * Logs a greeting message.
 *
 * @param {string} prenom Person's name.
 * @param {number|string} age Age, can be a number or a string.
 * @returns {void}
 */
function salutation(prenom, age) {
    console.log("Bonjour, je m'appelle " + prenom + " et j'ai " + age + " ans.");
}

salutation("Maurice", 54);
```

Function Returning a Value

```
/**
 * Calculates the sum of two numbers.
 *
 * @param {number} a First number.
 * @param {number} b Second number.
 * @returns {number} The sum of both numbers.
 */
function addition(a, b) {
    return a + b;
}

const resultat = addition(5, 3);
console.log(resultat); // 8
```

Function with Documented Error Handling

```
/**
 * Divides one number by another.
 *
 * @param {number} a Numerator.
 * @param {number} b Denominator.
 * @throws {Error} Throws an error if the denominator is zero.
 * @returns {number} The result of the division.
 */
function diviser(a, b) {
    if (b === 0) {
        throw new Error("Division by zero is not allowed.");
    }
    return a / b;
}

try {
    console.log(diviser(10, 0));
} catch (e) {
    console.error(e.message);
}
```

Deprecated Function

```
/**
 * Displays the current hour.
 *
 * @param {Date} date Date object.
 * @deprecated This function is obsolete. Use another one instead.
 * @returns {void}
 */
function heure(date) {
    console.log("It is " + date.getHours() + " o'clock");
```

```
}

heure(new Date());
```

Other Useful JSDoc Annotations

- `@async`: Indicates the function is asynchronous.
- `@callback`: Describes a callback function type.
- `@typedef`: Defines a custom type.
- `@property`: Describes an object's property.
- `@example`: Provides an example of how the function is used.
- `@see`: Refers to external documentation or another function.

Example with `@example` and `@typedef`

```
/**
 * Represents a user.
 * @typedef {Object} User
 * @property {string} name The user's name.
 * @property {number} age The user's age.
 */

/**
 * Displays user information.
 *
 * @param {User} user User object.
 * @returns {void}
 * @example
 * const u = { name: "Alice", age: 30 };
 * displayUser(u);
 */
function displayUser(user) {
    console.log(user.name + " is " + user.age + " years old.");
}
```

Part 2: PHP Documentation with PHPDoc

PHPDoc Syntax

```
<?php
/**
 * Says hello to the given name.
 *
 * @param string $name The name to greet.
 * @return void
 */
function bonjour(string $name): void {
```

```
    echo "Bonjour " . $name;
}

bonjour("Paul");
```

Common PHPDoc Annotations

Annotation	Description	Example
`@param`	Parameter type and description	`@param int $age`
`@return`	Return value type	`@return string`
`@throws`	Exceptions the method can throw	`@throws \Exception`
`@deprecated`	Marks the method as deprecated	`@deprecated`
`@var`	Type of a variable or property	`@var array<string>`

Example with Exception and Return

```php
/**
 * Performs division of two numbers.
 *
 * @param float $a
 * @param float $b
 * @return float
 * @throws \InvalidArgumentException If $b is zero.
 */
function diviser(float $a, float $b): float {
    if ($b === 0) {
        throw new \InvalidArgumentException("Division by zero is not allowed.");
    }
    return $a / $b;
}
```

Documenting Variables and Class Properties

The previous examples focused on functions, but documentation can apply to other elements as well.

In JavaScript (JSDoc)

A variable or constant:

```javascript
/**
 * Maximum number of allowed items.
 * @type {number}
 */
const MAX_ITEMS = 10;
```

A class, property, or method (including constructor):

```
/**
 * Represents a user.
 */
class User {
    /**
     * The user's name.
     * @type {string}
     */
    name;

    /**
     * The user's age.
     * @type {number}
     */
    age;

    /**
     * Creates a new user.
     * @param {string} name
     * @param {number} age
     */
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
}
```

In PHP (PHPDoc)

Same examples in PHP:

```
<?php
/**
 * Maximum number of allowed items.
 *
 * @var int
 */
const MAX_ITEMS = 10;

/**
 * User class
 */
class User
{
    /**
     * The user's name.
```

```
     *
     * @var string
     */
    public string $name;

    /**
     * The user's age.
     *
     * @var int
     */
    public int $age;

    /**
     * Constructor.
     *
     * @param string $name
     * @param int $age
     */
    public function __construct(string $name, int $age)
    {
        $this->name = $name;
        $this->age = $age;
    }
}
```

Conclusion & Best Practices

- Always document public and complex functions.
- Use clear and precise descriptions.
- Specify types, return values, possible errors, and deprecation status.
- Use automation tools to generate documentation.
- Keep documentation up to date to avoid confusion and ease maintenance.