JSDoc & PHPDocによるコードドキュメント：講座とベストプラクティス

はじめに

コードのドキュメントは、保守性、理解のしやすさ、そしてチームでのコラボレーションを可能にするために不可欠です。
JavaScriptのJSDocやPHPのPHPDocのようなツールを使うことで、構造化されたコメントを記述でき、VSCodeなどのIDEで補完、型チェック、ヒント表示などが可能になります。

第1部：JSDocによるJavaScriptのドキュメント化

完全な例と強化されたコメント

```
"use strict";

/**
 * 与えられた名前に挨拶をする関数。
 *
 * @param {string} prenom 挨拶する名前
 * @returns {void} 何も返しません
 */
function bonjour(prenom) {
    console.log("Bonjour " + prenom);
}

bonjour("Paul");
```

複数型の引数、戻り値、エラー処理の例

```
/**
 * 挨拶メッセージを出力する関数。
 *
 * @param {string} prenom 名前
 * @param {number|string} age 年齢（数値または文字列）
 * @returns {void}
 */
function salutation(prenom, age) {
    console.log("Bonjour, je m'appelle " + prenom + " et j'ai " + age + " ans.");
}

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

値を返す関数

```
/**
 * 2つの数値を加算する関数。
 *
 * @param {number} a 1つ目の数値
 * @param {number} b 2つ目の数値
 * @returns {number} 合計値
 */
function addition(a, b) {
    return a + b;
}

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

エラーをドキュメント化した関数

```
/**
 * 割り算を行う関数。
 *
 * @param {number} a 分子
 * @param {number} b 分母
 * @throws {Error} 分母が0の場合にエラーを投げる
 * @returns {number} 結果
 */
function diviser(a, b) {
    if (b === 0) {
        throw new Error("0での除算はできません。");
    }
    return a / b;
}

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

非推奨の関数

```
/**
 * 現在の時刻を表示する（非推奨）。
 *
 * @param {Date} date Dateオブジェクト
 * @deprecated この関数は非推奨です。別の関数を使用してください。
 * @returns {void}
 */
function heure(date) {
    console.log("Il est " + date.getHours() + " heures");
```

```
}

heure(new Date());
```

その他の便利なJSDocタグ

- `@async`：非同期関数であることを示す
- `@callback`：コールバック関数の型を記述
- `@typedef`：カスタム型を定義
- `@property`：オブジェクトのプロパティを記述
- `@example`：使用例を提供
- `@see`：他の関数や外部リファレンスを参照

`@typedef` と `@example` の例

```
/**
 * ユーザーを表すオブジェクト型
 * @typedef {Object} User
 * @property {string} name ユーザーの名前
 * @property {number} age ユーザーの年齢
 */

/**
 * ユーザー情報を表示する関数。
 *
 * @param {User} user ユーザーオブジェクト
 * @returns {void}
 * @example
 * const u = { name: "Alice", age: 30 };
 * displayUser(u);
 */
function displayUser(user) {
    console.log(user.name + "は" + user.age + "歳です。");
}
```

第2部：PHPDocによるPHPのドキュメント化

PHPDocの基本構文

```
<?php
/**
 * 名前に挨拶する関数
 *
 * @param string $name 挨拶する名前
 * @return void
 */
function bonjour(string $name): void {
```

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

bonjour("Paul");
```

よく使われるPHPDocアノテーション

タグ	説明	例
`@param`	引数の型と説明	`@param int $age`
`@return`	戻り値の型	`@return string`
`@throws`	例外の種類	`@throws \Exception`
`@deprecated`	非推奨であることを示す	`@deprecated`
`@var`	変数やプロパティの型	`@var array<string>`

例外と戻り値付きの関数

```php
/**
 * 割り算を行う関数。
 *
 * @param float $a
 * @param float $b
 * @return float
 * @throws \InvalidArgumentException $bが0の場合にスローされる
 */
function diviser(float $a, float $b): float {
    if ($b === 0) {
        throw new \InvalidArgumentException("0で割ることはできません。");
    }
    return $a / $b;
}
```

変数・プロパティ・クラスのドキュメント

JavaScript (JSDoc)

```javascript
/**
 * 許容される最大アイテム数
 * @type {number}
 */
const MAX_ITEMS = 10;

/**
 * ユーザークラス
 */
```

```
class User {
    /**
     * 名前
     * @type {string}
     */
    name;

    /**
     * 年齢
     * @type {number}
     */
    age;

    /**
     * コンストラクタ
     * @param {string} name
     * @param {number} age
     */
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
}
```

PHP (PHPDoc)

```
<?php
/**
 * 最大アイテム数
 *
 * @var int
 */
const MAX_ITEMS = 10;

/**
 * ユーザークラス
 */
class User
{
    /**
     * 名前
     *
     * @var string
     */
    public string $name;

    /**
     * 年齢
     *
     * @var int
     */
```

```
    public int $age;

    /**
     * コンストラクタ
     *
     * @param string $name
     * @param int $age
     */
    public function __construct(string $name, int $age)
    {
        $this->name = $name;
        $this->age = $age;
    }
}
```

結論とベストプラクティス

- 公開関数や複雑な処理は必ずドキュメント化する
- 説明は明確かつ簡潔に
- 型、戻り値、例外、非推奨情報を記載する
- 自動生成ツール（JSDocやphpDocumentorなど）を活用する
- ドキュメントは常に最新に保つ