[FE] Javascript 문서화 방법

목차

코드 문서화의 장점

유지보수 작업 시간을 단축할 수 있습니다.

Comment(주석)

Javascript 코드 주석 방법

```
// comment-방법1

/**
* comment-방법2
*/
```

고수들? 사이에서는 코드에 설명을 위한 주석이 많아서는 안 된다고 합니다. 코드 자체만으로 코드가 무슨 일을 하는지 알 수 있는 코드가 당연히 좋은 코드이기 때문이라고 합니다.

하지만, 저 같은 초보 눈에는 코드보다는, 코드에 기능을 적어 놓은 한글 주석이 더 좋습니다.

코드를 이해하는 데에 크게 도움이 되기 때문입니다. 저는 고수가 아닙니다.

코드 자체가 무슨 일을 하는지 주석을 안 남기고 알게 하는 방법

1. 함수의 길이가 긴 경우, 함수를 분리합니다.

2. 함수나 변수 이름을 '자기 설명적인 코드(self-descriptive)'가 되도록 작성합니다. 그니까, 함수나 변수 이름 자체가 주석 역할을 할 수 있도록 이름을 정성을 들여서 짓습니다.

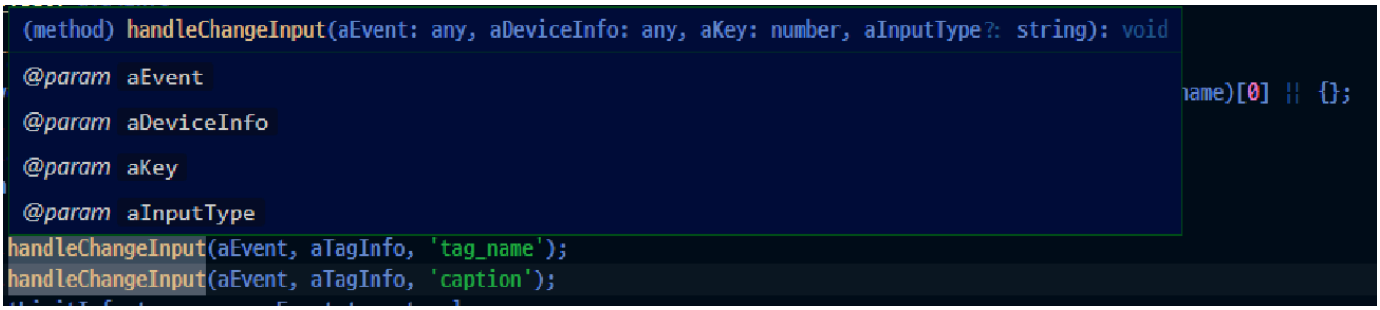
주석을 해주면 좋은 코드

- 구현 방법이 다양한 코드(시간 낭비를 하지 않게 도와줍니다.)
- 엉뚱한 기능의 코드

JsDoc

- JsDoc는 Javascript에서 사용하는 마크업 언어입니다. 단순, 주석 문법입니다.
- JsDoc는 단순 주석이므로 코드 로직에 어떠한 영향을 주지 않습니다. 타입이 틀린 매개변수가 들어와도 에러를 발생하지 않습니다.
- 주석 안에서 특정한 @(태그)을 사용해서 코드 편집기에 효과를 줄 수 있습니다.

```
/**
 * @param {object} aEvent
 * @param {object}  aDeviceInfo
 * @param {number}  aKey
 * @param {string}  aInputType
 */
handleChangeInput(aEvent, aDeviceInfo, aKey, aInputType='text') {
    aDeviceInfo[aKey] = aInputType === 'number' ? parseInt(aEvent.target.value) : aEvent.target.value;
},
```



```
(method) handleChangeInput(aEvent: any, aDeviceInfo: any, aKey: number, aInputType?: string): void
@param aEvent
@param aDeviceInfo
@param aKey
@param aInputType
handleChangeInput(aEvent, aTagInfo, 'tag_name');
handleChangeInput(aEvent, aTagInfo, 'caption');
```

JsDoc 장점

- 함수 코드에 대한 문서를 단순 주석보다 일관되게 작성할 수 있습니다. 매개변수와 함수 반환 값 정보를 주석으로 작성하는 겁니다.

```
/**
* @param {Object} employeeDetail 직원 정보 객체
* @param {string} employeeDetail.name 직원 이름
* @param {age} employeeDetail.age 직원 나이
* @return {string} 직원 정보 출력
*/
function getEmployeeDetail(employeeDetail) {
return `직원 ${employeeDetail.name}의 나이는 ${employeeDetail.age}입니다.`;
}
```

- JsDoc3을 사용해서 JsDoc 문법에 쓰여진 내용을 HTML 문서로 자동 생성할 수 있습니다.

JsDoc tags

가장 많이 사용하는 태그 순서로 작성했습니다.

@param

함수의 매개변수 인자 값의 타입과 이름을 작성할 수 있습니다.

복수의 타입을 가지는 경우, {number|string} 으로 작성하면 됩니다.

```
/**
* 함수 설명
* @param {Object} student - 학생 객체
* @param {string} student.name - 학생 이름
* @param {number} student.age - 학생 나이
*/
function 함수(student) {
//...
}
```

@return

함수가 반환하는 값을 명시합니다.

```
/**
* @return {number} 합계 출력
*/
function add(a,b) {
return a+b;
}
```

@callback

콜백으로 받은 인자 및 반환 값에 대한 정보를 명시합니다.

@author

작성자를 입력합니다. 작성자 뒤에 <>에는 이메일 주소를 입력할 수 있습니다.

```
/**
* @auther: 작성자
*/
```

@this

해당 함수 내부에 this가 참조하는 것을 표시합니다.

@description

코드에 대한 설명입니다. 그냥 주석에 작성해도 됩니다.

```
/**
* @description 함수 설명
*/
//or
/**
* 함수 설명
*/
```

@version

라이브러리 버전 정보를 입력하는 태그입니다.

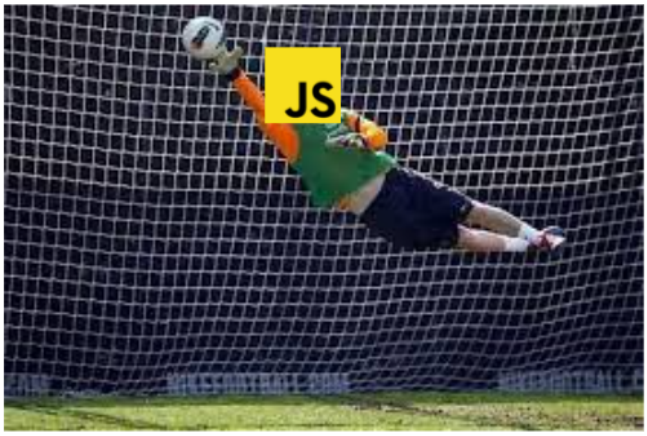
JsDoc 예시

```
/**
* 두 값을 더한 결과를 반환하는 함수
* @param {number} a
* @param {number} b
* @return {number}
*/
function add(a, b) {
return a + b;
}
```

Typescript를 사용한다면, 주석을 사용해서 문서화를 하지 않아도 Typescript의 '타입 명시' 특성으로 코드 자체가 문서화가 됩니다.

그리고, 컴파일할 때 Type 오류를 잡아주기 때문에 브라우저에서 에러를 발견하지 않아도 됩니다.

edgemaster가 안정화가 된다면, Typescript를 적극적으로 검토해서 도입해야 한다고 생각합니다.



참고

위키피디아 검색 JsDocs

JAVASCRIPT.INFO-주석

-끗-