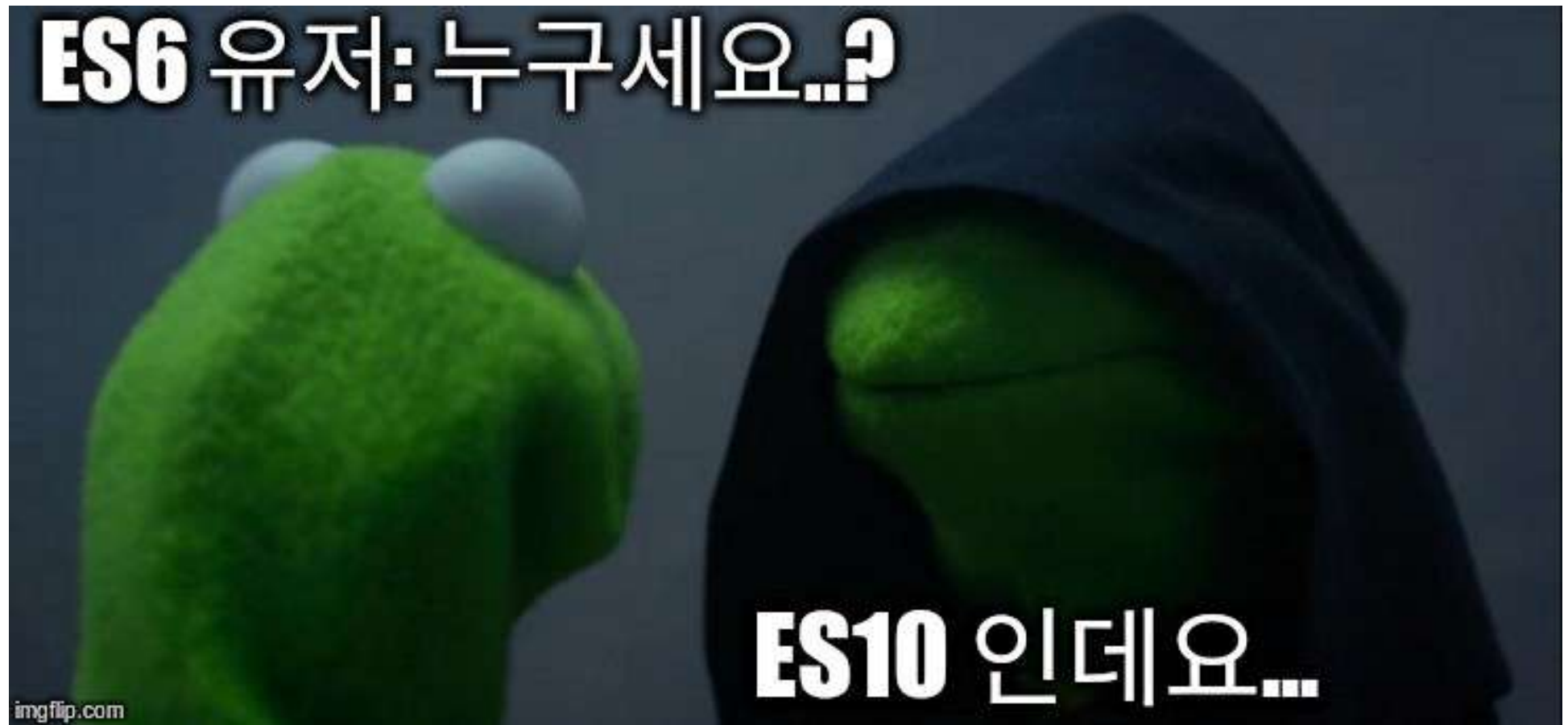


ES10 미리보기

첫 만남



ES 에 대해서

ECMAScript version

=

ES

version

=

JavaScript

실패할 수 있는 연산



```
const myList = [];  
myList.pop( );
```

그들은 어디에나 있습니다



타버린 김치볶음밥

나쁜 예

맨밥

밥 볶기

김치볶음밥

|

탄 김치볶음밥

치즈 올리기

치즈김치볶음밥

|

탄 치즈김치볶음밥

좋은 예

맨밥

밥 볶기

김치볶음밥

|

탄 김치볶음밥

냉장고에서 먹다 남은 김치볶음밥
꺼내기

치즈 올리기

치즈김치볶음밥


지금까지 이야기한 것

1. 유용하다고 할 수 있을 정도 수준인 프로그램들은 실패할 수 있는 연산을 다룹니다 .
2. 실패할 수 있는 연산은 실패한 경우에 대한 로직과 성공한 경우에 대한 로직을 요합니다 .

연산의 실패를 나타내기 위한 방법들

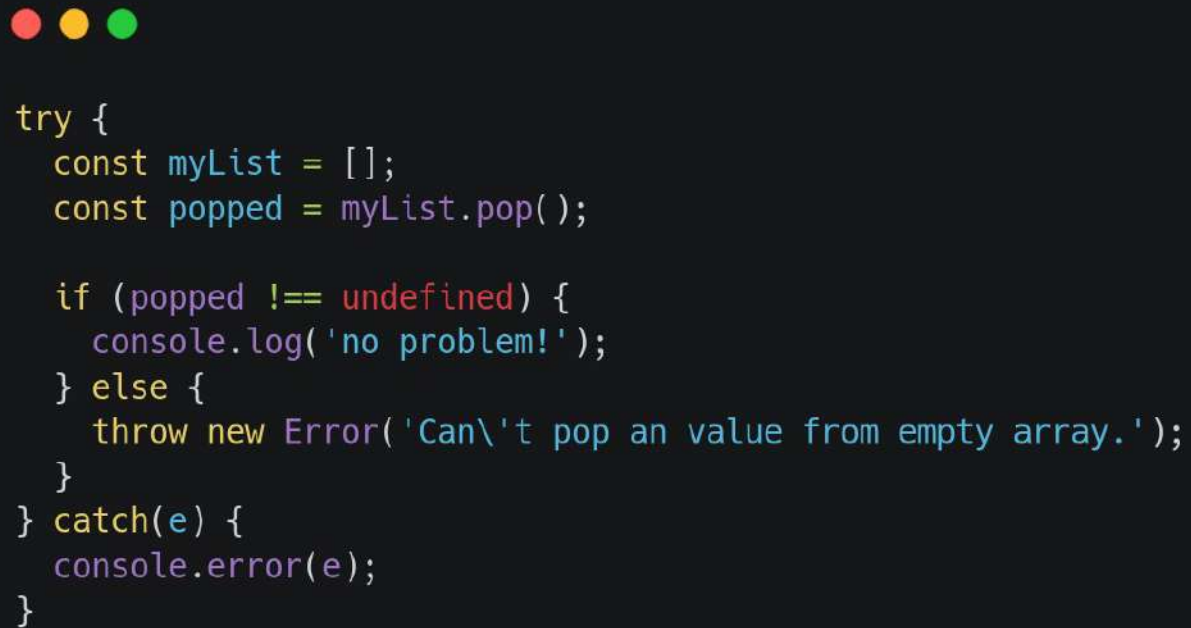
1. 연산의 실패를 나타내기 위한 값 (`undefined`, `null`) 을 반환하기
2. `throw` 문을 통해 에러를 발생시키기

Try Statement!



```
try {  
    // 실패할 수도 있는 연산  
} catch(errorBinding) {  
    // 실패한 경우  
}
```

Try



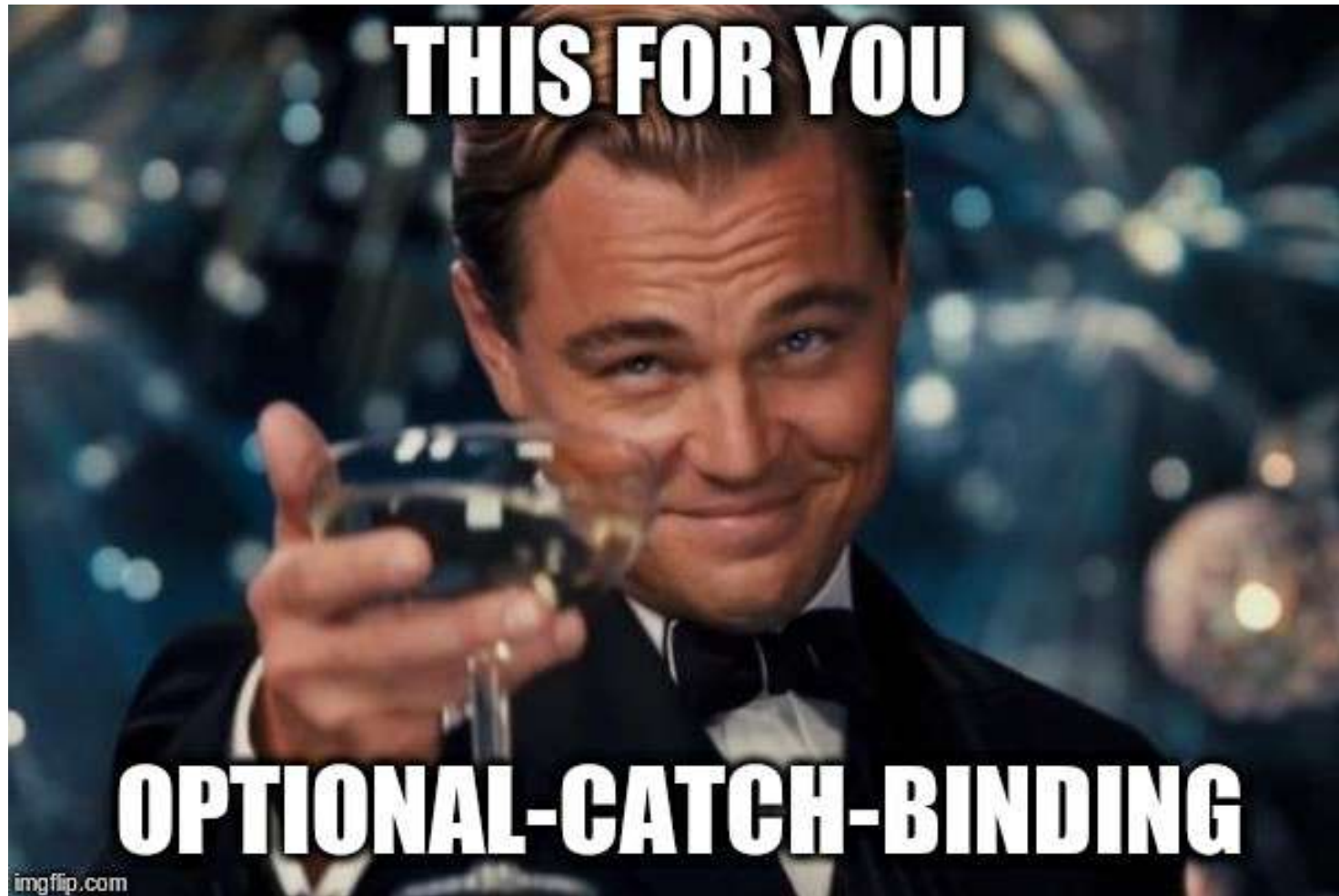
```
try {  
  const myList = [];  
  const popped = myList.pop();  
  
  if (popped !== undefined) {  
    console.log('no problem!');  
  } else {  
    throw new Error('Can\'t pop an value from empty array.');  }  
} catch(e) {  
  console.error(e);  
}
```

문제는



```
let 치즈김치볶음밥;  
  
try {  
  const 김치볶음밥 = new 김치볶음밥();  
  
  치즈김치볶음밥 = 치즈올리기(김치볶음밥);  
} catch(useless) {  
  치즈김치볶음밥 = 냉장고.전에먹던치즈김치볶음밥;  
}
```

여기 여러분들을 위한 선물이 있습니다

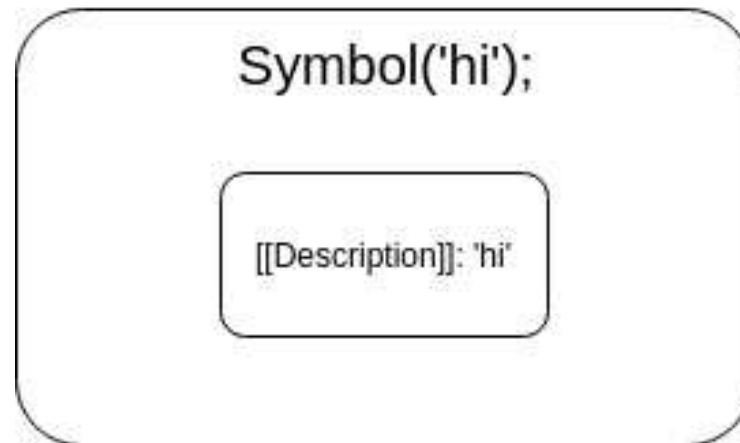
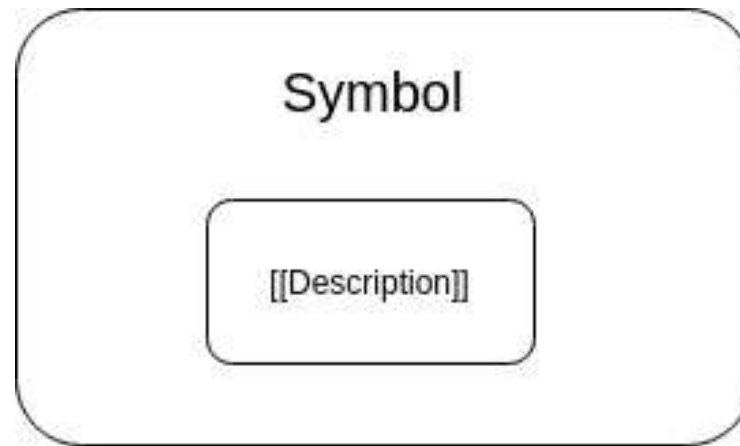


Optional Catch Binding



```
let 치즈김치볶음밥;  
  
try {  
    const 김치볶음밥 = new 김치볶음밥();  
  
    치즈김치볶음밥 = 치즈올리기(김치볶음밥);  
} catch {  
    치즈김치볶음밥 = 냉장고.전에먹던치즈김치볶음밥;  
}
```

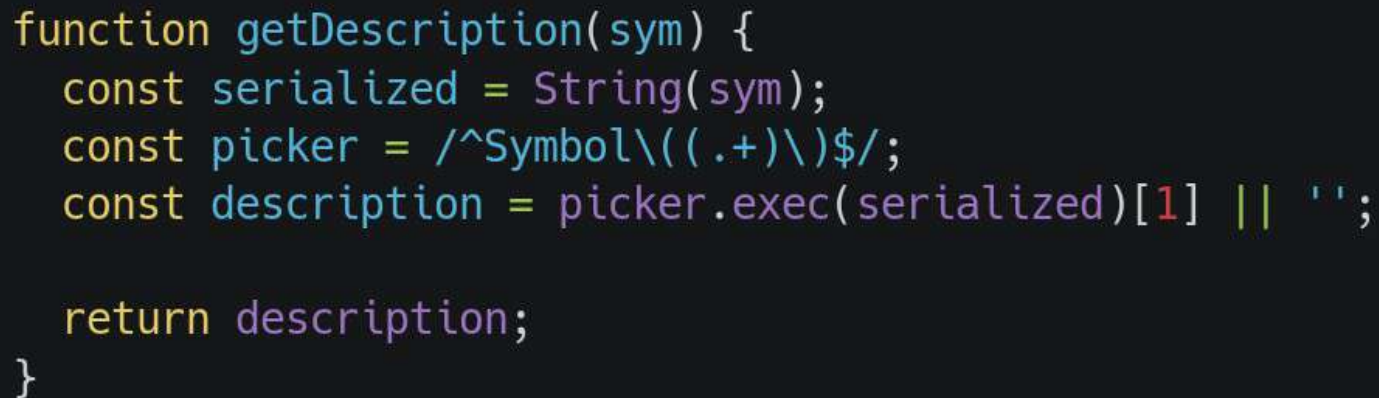
Internal of Symbol



Symbol

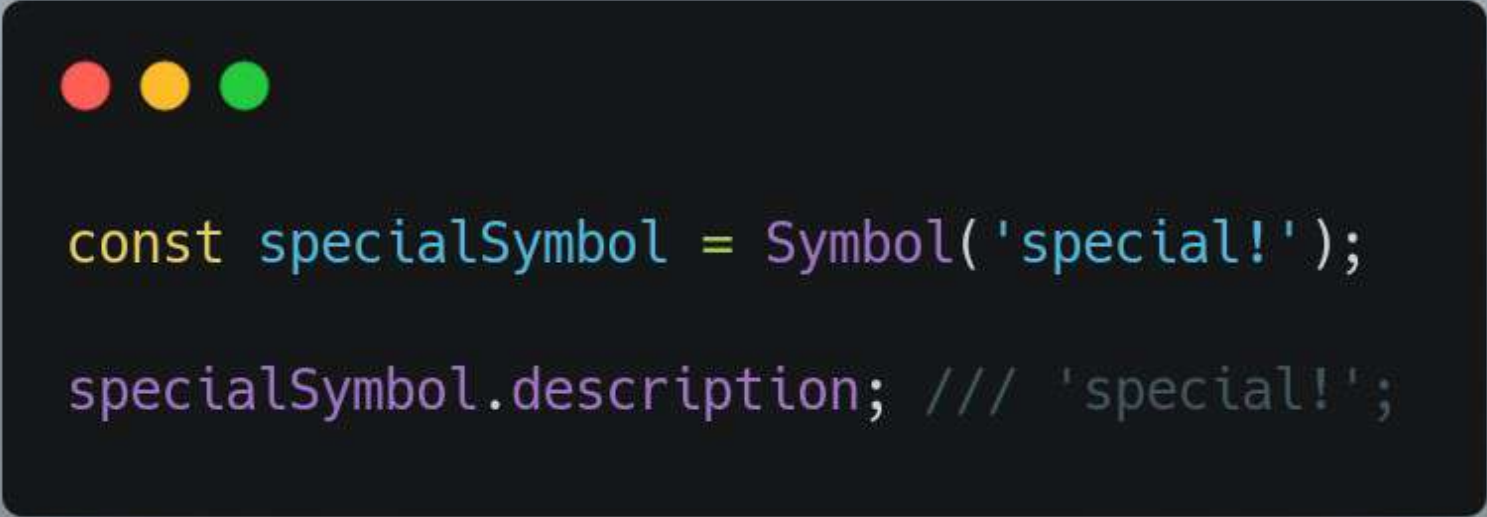
```
Symbol(/* description */);  
const mySymbol = Symbol(); // 값만 보아서는 도대체 뭘 의미하는 지 알 수 없습니다.  
const mySymbol2 = Symbol('used to debug module A'); // 이제 그 의미가 또렷하게 드러나는군요.  
  
mySymbol.toString(); // 'Symbol()'  
mySymbol2.toString(); // 'Symbol(used to debug module A)'
```

Description 은 직접적으로 접근할 수 없었습니다



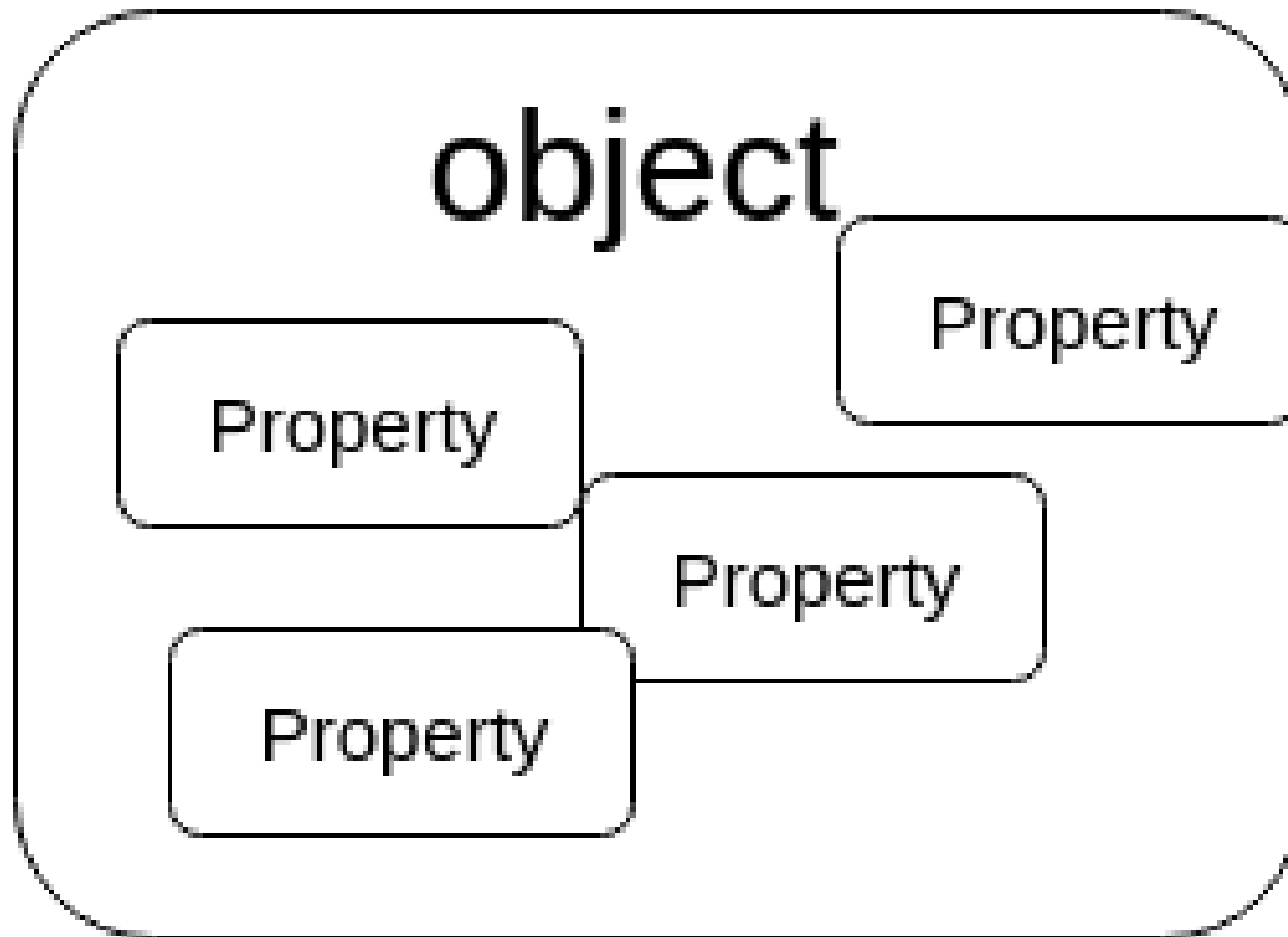
```
function getDescription(sym) {  
  const serialized = String(sym);  
  const picker = /^Symbol\((.+)\)$/;  
  const description = picker.exec(serialized)[1] || '';  
  
  return description;  
}
```


이제는 아닙니다



```
const specialSymbol = Symbol('special!');  
specialSymbol.description; /// 'special!';
```

Essential of object

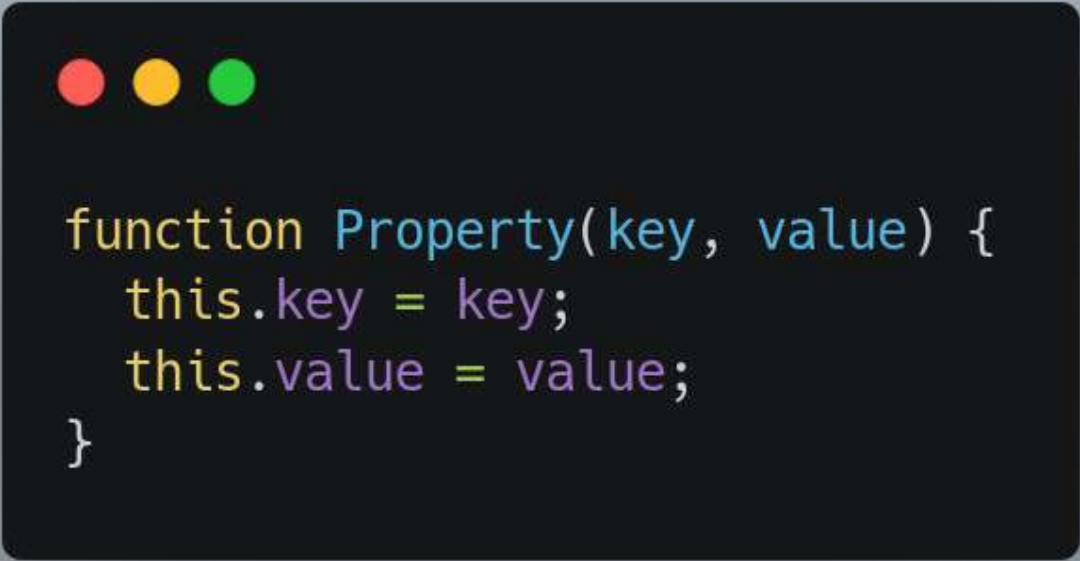


본질을 파악하기



```
const myObj = { a: 1, b: 2 };  
const myAlternativeObj = [propertyA, propertyB];
```

Property



```
function Property(key, value) {  
  this.key = key;  
  this.value = value;  
}
```

Alternative object



```
const myAlternativeObj = [new Property('a', 1), new Property('b', 2)];
```

내부를 다시 열어봅시다



```
const myAlternativeObj = [{ key: 'a', value: 1 }, { key: 'b', value: 2 }];
```

더 간단하게 !



```
const myAlternativeObj = [['a', 1], ['b', 2]];
```

Object.entries



```
const myObj = { a: 1, b: 2 };  
const myAlternativeObj = Object.entries(myObj); // [['a', 1], ['b', 2]];
```


Object.fromEntries



```
const myObj = { a: 1, b: 2 };  
const myAlternativeObj = Object.entries(myObj);  
const myObj2 = Object.fromEntries(myAlternativeObj); // { a: 1, b: 2 };
```



```
const myObj = { a: 1, b: 2 };  
const myMap = new Map(Object.entries(myObj));  
const myObj2 = Object.fromEntries(myMap); // { a: 1, b: 2 };
```

String trimming

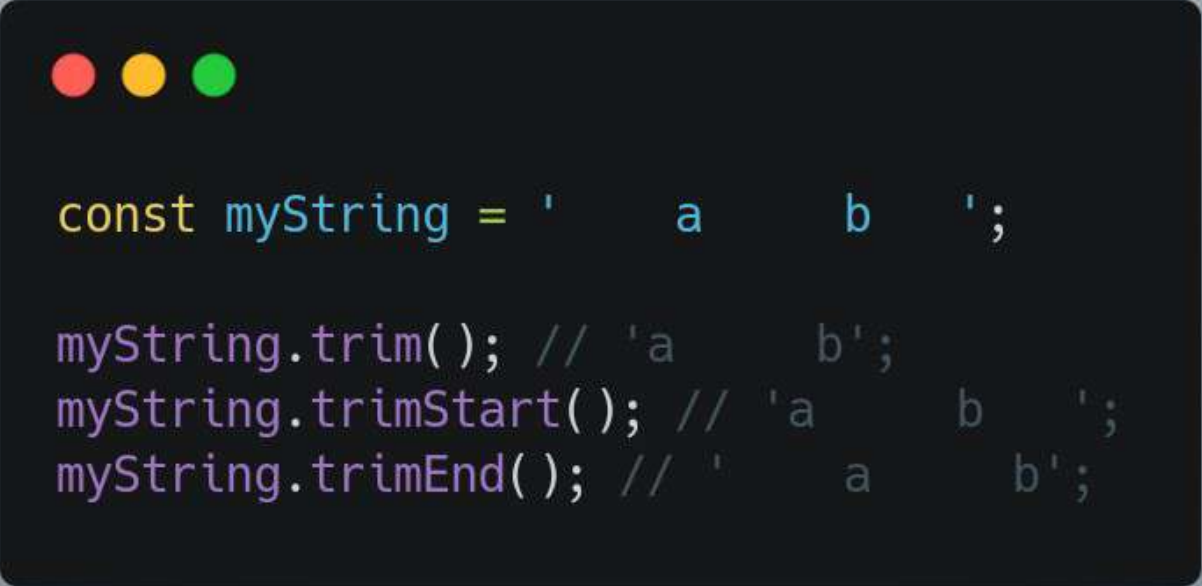


Aliases



```
String.prototype.trimLeft.name; // 'trimStart';  
String.prototype.trimRight.name; // 'trimEnd'
```

Trimming methods



```
const myString = '  a  b  ';  
  
myString.trim(); // 'a  b';  
myString.trimStart(); // 'a  b  ';  
myString.trimEnd(); // '  a  b';
```

Array.prototype.flat



```
const array1 = [[1, 2], [3, 4], [5, 6]];
```

```
array1.flat(1); // [1, 2, 3, 4, 5, 6];
```

```
const array2 = [[[1], [2]], [[3], [4]], [[5], [6]]];
```

```
array2.flat(2); // [1, 2, 3, 4, 5, 6];
```

The Smoosh gate



[Core](#) [More](#) [Blog](#) [Forge](#) [Contribute](#)

MooTools is a collection of JavaScript utilities designed for the intermediate to advanced JavaScript developer. It allows you to write powerful and flexible code with its elegant, well documented, and coherent APIs.

MooTools code is extensively documented and easy to read, enabling you to extend the functionality to match your requirements.

Open Source License

MooTools libraries are released under the Open Source MIT license which gives you the possibility to use them and modify them in every circumstance.

How to use?

MooTools Selectors

Selectors for DOM Elements

```
// get elements by class
$$('.foo'); // or even: document.getElements('.foo');

// selector with different elements
$$('div.foo, div.bar, div.bar a');

// get a single element
document.getElement('div.foo');
```

How to create new DOM elements

A simple MooTools Element example.

```
// the short way
new Element('div#bar.foo');
```

Ajax!

MooTools uses a Class called Request.

```
// create a new Class instance
var myRequest = new Request({
  url: 'getMyText.php',
  method: 'get',
  onRequest: function(){
    myElement.set('text', 'loading...');
  },
  onSuccess: function(responseText){
    myElement.set('text', responseText);
  },
  onFailure: function(){
    myElement.set('text', 'Sorry, your request failed');
  }
});

// and to send it:
myRequest.send(data);
```

Mapping



```
const myArray = [1, 2, 3, 4];  
const mappedMyArray = myArray.map(x => x + 1); // [2, 3, 4, 5];
```


Array.prototype.flatMap



```
const myArray = [1, 2, 3, 4];  
const mappedMyArray = myArray.map(x => [x]); // [[2], [3], [4], [5]];  
const flattenedMappedMyArray = mappedMyArray.flat(1); // [2, 3, 4, 5];  
  
const flatMappedMyArray = myArray.flatMap(x => [x]); // [2, 3, 4, 5];
```

Any Questions?



추가시간 !

