Arrow Function of JS

Agenda

1. 선언
2. 호출
3. this
4. Arrow Function 사용하면 안되는 경우

1. 선언

화살표 함수(Arrow Function)은 `function` 키워드 대신 화살표(`=>`)를 사용해서 보다 간략한 방법으로 함수를 선언할 수 있다. 화살표 함수의 기본 문법은 아래와 같다.

```
() => { ... } // 매개변수가 없을 경우
x => { ... } // 매개변수가 한 개인 경우, 소괄호를 생략 가능하다.
(x, y) => { ... }

x => { return x * x }
x => x * x  // 함수 몸체가 한줄의 구문이라면 중괄호를 생략 가능하며 암
() => { return { a: 1}; }
() => ({ a: 1 }) // 위 표현과 동일하다. 객체 반환시 소괄호를 사용한다

() => {
    const x = 10;
    return x * x;
}
```

2. 호출

화살표 함수는 익명 함수로만 사용할 수 있다. 따라서 화살표 함수를 호출하기 위해서는 함수 표현식을 사용한다.

```
// ES5
var pow = function (x) { return x * x; };
console.log(pow(10)); // 100
```

```
// ES6
const pow = x => x * x;
console.log(pow(10)); // 100
```

또는 콜백 함수로 사용할 수도 있다. 일반적인 함수 표현식보다 표현이 간결하다.

```
// ES5
var arr = [1, 2, 3];
var pow = arr.map(function (x) {
    return x * x;
});

console.log(pow); // [1, 4, 9]
```

```
// ES6
const arr = [1, 2, 3];
const pow = arr.map(x => x * x);

console.log(pow); // [1, 4, 9]
```

3. this

일반 함수와 화살표 함수의 가장 큰 차이점

=> `this` 에 바인딩할 객체

3.1 일반함수의 `this`

- 함수 **호출** 시 **동적**으로 결정
- 생성자 함수 or 메소드(객체 멤버)의 경우
    - 상위 스코프의 `this` 를 가리킴
- 위 경우를 제외한 모든 함수의 경우
    - 전역 객체 `window` 를 가리킴

```
function Prefixer(prefix) {
    this.prefix = prefix;
};

Prefixer.prototype.prefixArray = function (arr) {
    // (A)
    return arr.map(function (x) {
        return this.prefix + ' ' + x; // (B)
    });
};

var pre = new Prefixer('Hi!');
console.log(pre.prefixArray(['onsuk', 'kwon', 'seunguk']))
```

- (A) 지점에서의 `this` 는 `pre` 이다.
- (B) 지점에서의 `this` 는 전역 객체 `window` 이다.

생성자 함수와 객체의 **메소드**를 제외한 모든 함수(*내부 함수, 콜백 함수 포함*) 내부의 `this` 는 전역 객체를 가리키기 때문이다.

콜백 함수 내부의 `this` 가 메소드를 호출한 객체(생성자 함수의 인스턴스)를 가리키게 하려면 3가지 방법이 있다.

그 중 한가지를 소개하겠다.

```
// Solution : bind(this)
function Prefixer(prefix) {
    this.prefix = prefix;
};

Prefixer.prototype.prefixArray = function (arr) {
    return arr.map(function (x) {
        return this.prefix + ' ' + x;
    }.bind(this));
};

var pre = new Prefixer('Hi!');
console.log(pre.prefixArray(['onsuk', 'kwon', 'seunguk']))
```

3.2 화살표 함수의 `this`

- 함수 **선언** 시 **정적**으로 결정
- **언제나** 상위 스코프의 `this` 를 가리킴
- 위 방법을 syntax sugaring한 것

```
function Prefixer(prefix) {
    this.prefix = prefix;
};

Prefixer.prototype.prefixArray = function (arr) {
    return arr.map(x => `${this.prefix} ${x}`);
};

const pre = new Prefixer('Hi!')
console.log(pre.prefixArray(['onsuk', 'kwon', 'seunguk']))
```

4. 화살표 함수를 사용하면 안되는 경우

화살표 함수는 Lexical this를 지원하므로 콜백 함수로 사용하기 편리하다.

하지만 화살표 함수를 사용하는 것이 오히려 혼란을 불러오는 경우도 있으므로 주의해야 한다.

4.1 메소드

화살표 함수로 메소드를 정의하는 것은 피해야 한다.

```
// Bad
const person = {
    name: 'onsuk',
    sayHi: () => console.log(`Hi ${this.name}`)
};

person.sayHi(); // Hi undefined
```

- 여기서 `this`는 `person` 객체가 아닌 전역 객체 `window`를 가리키고 있다.

이와 같은 경우는 메소드를 위한 단축 표기법인 ES6의 축약 메소드 표현을 사용하는 것이 좋다.

```
// Good
const person = {
    name: 'onsuk',
    sayHi() {
        console.log(`Hi ${this.name}`);
    }
};

person.sayHi(); // Hi onsuk
```

4.2 prototype

화살표 함수로 정의한 메소드를 `prototype` 에 할당하는 경우도 동일한 문제가 발생한다. 예시를 통해 살펴보자.

```
// Bad
const person = {
    name: 'onsuk'
};

Object.prototype.sayHi = () => console.log(`Hi ${this.nam

person.sayHi(); // Hi undefined
```

- 화살표 함수로 객체의 메소드를 정의했을 때와 동일한 문제가 발생한다.

4.3 생성자 함수

생성자 함수로 사용할 수 없다.

생성자 함수는 `prototype` 프로퍼티가 가리키는 프로토타입 객체의 `consturctor` 가 있어야 한다.

하지만 화살표 함수는 `prototype` 프로퍼티를 가지고 있지 않다. 따라서 `consturctor` 이 없기 때문에 생성자 함수로 사용할 수 없다.

```
const Foo = () => {};

console.log(Foo.hasOwnProperty('prototype')); // false

const foo = new Foo(); // Uncaught TypeError: Foo is not a
```

4.4 addEventListener 함수의 콜백 함수

`addEventListener` 함수의 콜백 함수를 화살표 함수로 정의하면 `this`가 상위 컨택스트인 전역 객체 `window`를 가리킨다.

```
// Bad
var button = document.getElementById('myButton');

button.addEventListener('click', () => {
    console.log(this === window); // true
    this.innerHTML = 'Clicked button';
});
```

- 따라서 `function` 키워드로 정의한 일반 함수를 사용하여야 한다.

```
// Good
var button = document.getElementById('myButton');

button.addEventListener('click', function() {
    console.log(this === button); // true
    this.innerHTML = 'Clicked button';
});
```

Reference

화살표 함수