

22장 this



모던 자바스크립트 Deep Dive

this 키워드

객체: 상태를 나타내는 프로퍼티와 동작을 나타내는 메서드를 하나의 논리적인 단위로 묶은 복합적 자료구조

메서드가 자신이 속한 객체의 프로퍼티를 참조하려면
자신이 속한 객체를 가리키는 식별자를 참조할 수 있어야 함

this 키워드

객체 리터럴 방식 -> getDiameter 메서드 내에서 메서드 자신이 속한 객체를 가리키는 식별자 **circle**을 참조하고 있음

```
const circle = {  
  radius: 5, → 프로퍼티  
  getDiameter() {  
    return 2 * circle.radius; → 메서드  
  },  
};  
console.log(circle.getDiameter());
```

this 키워드

생성자 함수 방식 -> 함수를 정의하는 시점에는 생성자 함수가 생성할
인스턴스를 가리키는 식별자를 알 수 x

```
function Circle(radius) {  
  ? .radius = radius;  
}  
const circle = new Circle(5);
```

this 키워드

- **this:** 자신이 속한 객체 또는 자신이 생성할 인스턴스를 가리키는 자기 참조 변수
- 자바스크립트 엔진에 의해 암묵적으로 생성되며, 코드 어디서든 참조 가능
- this 바인딩은 함수 호출 방식에 의해 동적으로 결정됨

함수 호출 방식과 this 바인딩

1. 일반 함수 호출
2. 메서드 호출
3. 생성자 함수 호출
4. `Function.prototype.apply` / `call` / `bind` 메서드에 의한 간접 호출

함수 호출 방식과 this 바인딩 - 1. 일반 함수 호출

일반 함수로 호출된 모든 함수 내부 this에는 전역 객체가 바인딩됨

```
function foo() {  
    console.log(this); // window  
    function bar() {  
        console.log(this); // window  
    }  
    bar();  
}  
foo();
```

함수 호출 방식과 this 바인딩 - 1. 일반 함수 호출

Strict mode가 적용된 일반 함수 내부 this에는 undefined가 바인딩됨

```
function foo() {  
  'use strict';  
  console.log(this); // undefined  
  function bar() {  
    console.log(this); // undefined  
  }  
  bar();  
}  
foo();
```


함수 호출 방식과 this 바인딩 - 1. 일반 함수 호출

메서드 내부의 중첩 함수나 콜백 함수의 this 바인딩을 메서드의 this 바인딩과 일치시키기 위한 방법

```
const obj = {  
  value: 100,  
  foo() {  
    const that = this;  
    setTimeout(function () {  
      console.log(that.value); // 100  
    }, 100);  
  },  
};
```

→ this 바인딩 (obj)을 변수 that에 할당

→ 콜백 함수 내부에서 this 대신 that 참조

함수 호출 방식과 this 바인딩 - 1. 일반 함수 호출

메서드 내부의 중첩 함수나 콜백 함수의 this 바인딩을 메서드의 this 바인딩과 일치시키기 위한 방법

```
const obj = {  
  value: 100,  
  foo() {  
    setTimeout(() => console.log(this.value), 100);  
    //100  
  },  
};
```

함수 호출 방식과 this 바인딩 - 2. 메서드 호출

메서드 내부의 this에는 메서드를 호출한 객체가 바인딩됨

```
const person = {  
  name: 'Lee',  
  getName() {  
    return this.name;  
  },  
};  
console.log(person.getName());
```

함수 호출 방식과 this 바인딩 - 2. 메서드 호출

메서드 내부의 this에는 메서드를 호출한 객체가 바인딩됨

```
const anotherPerson = {  
  name: 'Kim',  
};
```

```
anotherPerson.getName = person.getName;  
console.log(anotherPerson.getName());
```

→ getName 메서드를
호출한 객체는
anotherPerson

```
const getName = person.getName;
```

```
console.log(getName());
```

→ getName 메서드를 일반 함수로 호출

함수 호출 방식과 this 바인딩 - 3. 생성자 함수 호출

생성자 함수 내부의 this에는 생성자 함수가 생성할 인스턴스가 바인딩됨

```
function Circle(radius) {  
  this.radius = radius;  
  this.getDiameter = function () {  
    return 2 * this.radius;  
  };  
}
```

```
const circle1 = new Circle(5);  
console.log(circle1.getDiameter());
```

반지름 5인 Circle 객체 생성

함수 호출 방식과 this 바인딩 - 4. apply/call/bind 메서드에 의한 간접 호출

- `Function.prototype.apply`
- `Function.prototype.call`
- `Function.prototype.bind`

함수 호출 방식과 this 바인딩 - 4. apply/call/bind 메서드에 의한 간접 호출

apply() 메서드는 주어진 this 바인딩과 인수 리스트 배열을 사용해 함수를 호출함

```
Function.prototype.apply(thisArg[, argsArray])
```

- thisArg -> this로 사용될 객체
- argsArray -> 함수에게 전달할 인수 리스트의 배열 또는 유사 배열 객체
- returns -> 호출된 함수의 반환값

함수 호출 방식과 this 바인딩 - 4. apply/call/bind 메서드에 의한 간접 호출

`call()` 메서드는 주어진 `this` 바인딩과 ,로 구분된 인수 리스트를 사용하여 함수를 호출함

```
Function.prototype.call(thisArg[, arg1[, arg1[, ...]]])
```

- `thisArg` -> `this`로 사용될 객체
- `arg1, arg2 ...` -> 함수에게 전달할 인수 리스트
- `returns` -> 호출된 함수의 반환값

함수 호출 방식과 this 바인딩 - 4. apply/call/bind 메서드에 의한 간접 호출

```
function getThisBinding() {  
    console.log(arguments);  
    return this;  
}
```

```
const thisArg = { a: 1 };
```



this로 사용할 객체

```
console.log(getThisBinding.apply(thisArg, [1, 2, 3]));  
console.log(getThisBinding.call(thisArg, 1, 2, 3));  
// { '0': 1, '1': 2, '2': 3 }  
// { a: 1 }
```

함수 호출 방식과 this 바인딩 - 4. apply/call/bind 메서드에 의한 간접 호출



```
function getThisBinding() {  
    return this;  
}
```

`const thisArg = { a: 1 };` → this로 사용할 객체

```
console.log(getThisBinding.bind(thisArg));  
// getThisBinding  
console.log(getThisBinding.bind(thisArg)());  
// {a: 1}
```

함수 호출 방식과 this 바인딩 - 4. apply/call/bind 메서드에 의한 간접 호출

bind 메서드로 메서드의 this와 메서드 내부 함수의 this 불일치 문제 해결

```
const person = {  
  name: 'Lee',  
  foo(callback) {  
    1  
    setTimeout(callback.bind(this), 100);  
  },  
};  
  
person.foo(function () {  
  console.log(this.name); // Lee 2  
});
```