코어 JS - 6단원 답변

```
// 박범준
1. static method는 무엇이며, 왜 사용하나요?

- static 키워드를 사용하여 메소드를 정의하면
    new "클래스이름" 으로  인스턴스 객체를 생성하지 않아도 해당 메소드를

class A{
    static a(){
        console.log("a");
    }
}

A.a(); // "a"

- 정적 메서드를 사용하는 이유
    1. 정적 메소드는 오버라이드가 불가능하며 절차지향적인 성향이 강함 (무분
    2. 메모리 측면에서 효율적, 객체를 생성하지 않고 사용가능 하기 때문에 ㅎ

class MathUtil {
    static add(a, b) {
        return a + b;
    }
}

console.log(MathUtil.add(5, 10)); // 15

3. 객체 전용 메소드의 예외사항은 무엇인가요?

- 1. 정적 메서드와의 혼동
정적 메서드는 클래스 자체에 속하고, 객체 전용 메서드는 객체(인스턴스)에 속함
정적 메서드는 객체의 데이터를 다룰 수 X
->인스턴스 데이터(this)를 사용하려고 하면 오류가 날 수 있음
class MyClass {
    static staticMethod() {
```

```
        console.log(this.value); // 오류: 정적 메서드는 객체 데이터
    }
}

- 2. this가 예상과 다르게 동작할 때
객체 전용 메서드 안에서 다른 함수(특히 일반 함수)를 호출하면,
그 함수 안에서 this가 객체를 가리키지 않고, 전역 객체(브라우저에서는 wind
class MyClass {
    instanceMethod() {
        function innerFunction() {
            console.log(this.value); // 여기서 `this`는 전역 객체
        }
        innerFunction(); // innerFunction이 호출되면 'this'는 전
    }
}

const obj = new MyClass(42);
obj.instanceMethod(); // undefined

=> 화살표 함수를 사용하거나 this를 다른 변수에 저장하면 해결 가능

- 3. 프로토타입 체인에서의 문제
객체 전용 메서드가 다른 객체에서도 호출될 수 있음
const obj = {
    value: 10,
    showValue: function() {
        console.log(this.value);
    }
};

const anotherObj = { value: 20 };
anotherObj.showValue = obj.showValue;
anotherObj.showValue(); // 20 (기대하지 않은 결과)
```

```
// 손원륜
2. Object.create() 메서드는 무엇이며, 어떻게 Prototype을 설정하는가?
- 객체를 생성하는 메서드로, 지정된 프로토타입 객체와 속성을 가지는 새 객체를
```

```
Object.create(proto[, propertiesObject])
매개변수 proto:새로 만든 객체의 프로토타입이어야 할 객체

const proto = {
    greet() {
        console.log('Hello!');
    }
};

const obj = Object.create(proto);
obj.greet(); // 'Hello!'

-----------------------------------------------
const grandparent = {
    grandparentMethod() {
        console.log('Hello from grandparent');
    }
};

const parent = Object.create(grandparent);
parent.parentMethod = function() {
    console.log('Hello from parent');
};

const child = Object.create(parent);
child.childMethod = function() {
    console.log('Hello from child');
};

child.childMethod(); // 'Hello from child'
child.parentMethod(); // 'Hello from parent'
child.grandparentMethod(); // 'Hello from grandparent'
```

```
// 윤대영
Q3. static method, static property 선언 방법과 사용 이유에 대해 알려

static method
```

선언 방법
- 클래스 내에서 static 키워드를 사용하여 선언됨
- 정적 메서드는 클래스 이름을 통해 직접 호출
- 클래스 자체에 관련된 기능을 구현하는 데 사용됨

```
class MyClass {
    static staticMethod() {
        return 'This is a static method';
    }
}

console.log(MyClass.staticMethod()); // 'This is a static met
```

사용 이유
1. 공유 메서드
- 정적 메서드는 클래스의 모든 인스턴스에서 동일하게 사용될 수 있는 기능을 구
- ex) 유틸리티 함수, 팩토리 메서드, 데이터 변환 메서드 등

2. 인스턴스와 무관한 기능
- 클래스 자체에 관련된 동작을 수행하거나 인스턴스와 상관없는 기능을 제공할
- 인스턴스의 상태를 변경하거나 참조할 필요가 없는 기능을 정적 메서드로 정의

static property
선언 방법
- 클래스 내에서 static 키워드를 사용하여 선언됨
- 클래스 이름을 통해 직접 호출
- 클래스와 관련된 데이터를 저장(수값, 설정값)

사용 이유
1. 공유 상태
- 정적 속성은 클래스의 모든 인스턴스가 공유해야 하는 데이터를 저장할 때 유용
- ex) 특정 설정값이나 카운터와 같이 클래스 전체에서 공통으로 사용해야 하는

2. 상수 정의
- 클래스와 관련된 상수를 정적 속성으로 정의할 수 있음
-> 코드의 가독성을 높이고, 유지보수를 쉽게 만듦

```
class Calculator {
```

```
    static PI = 3.14159;

    static calculateCircleArea(radius) {
        return Calculator.PI * radius * radius;
    }
}

console.log(Calculator.PI); // 3.14159
console.log(Calculator.calculateCircleArea(5)); // 78.53975
```

정적 메서드는 함수(메서드)
- 어떤 작업을 수행하고, 인자를 받아 처리할 수 있음

정적 속성은 데이터
- 클래스와 관련된 값을 저장하고, 필요할 때 읽거나 쓸 수 있음