삼육대학교 SW융합교육원 김진호

자바스크립트 기초교육

<class>
</body>
<head>
<title>

1-3. Class
(정적 메서드와 정적 프로퍼티)

1-3-1. static method

1-3-2. static property

1. Class

1-3. static method and property (정적 메서드와 정적 프로퍼티)

1-3-1. static method (정적 메서드)

- 정적 메서드는 특정 클래스 인스턴스가 아닌 클래스 '전체'에 필요한 기능을 만들 때 사용한다.
- 클래스 선언부 안에 위치하고 앞에 static이라는 키워드를 붙인다.

1. Class

1-3. static method and property (정적 메서드와 정적 프로퍼티)

1-3-1. static method (정적 메서드)

```
class Student {

    constructor(name, height) {
        this.name = name;
        this.height = height;
    }

    // 클래스 선언부 안에 위치하고 앞에 static이라는 키워드를 붙인다.
    static compare(studentA, studentB) {
        return studentA.height - studentB.height;   // 인스턴스끼리 비교해주는 메서드
    }

}

let students = [
    new Student('유관순', 165.5),
    new Student('홍길동', 180.5),
    new Student('선덕여왕', 159.5)
];

students.sort(Student.compare); // 신장 오름차순으로 배열을 정렬한다.
console.log(students);
// Student.compare는 학생들의 신장을 비교해주는 수단으로 하나의 학생마다 필요한 메서드가 아니라
클래스의 메서드여야 한다. => 정적 메서드

Student.staticMethod = function () {
    console.log('staticMethod는 메서드를 프로퍼티 형태로 직접 할당하는 것과 동일하다.');
}

Student.staticMethod();
```

1. Class

1-3. static method and property (정적 메서드와 정적 프로퍼티)

1-3-1. static method (정적 메서드)

- 조건에 맞는 Student를 만들어야 할 때 생성자도 사용 가능하지만,
  클래스에 정적 메서드를 만들어 팩토리 메서드를 구현할 수 있다.

```
class User {

    constructor(id, registDate) {
        this.id = id;
        this.registDate = registDate;
    }

    static registUser(id) {
        return new this(id, new Date());
    }
}

let user01 = User.registUser('user01'); // User.registUser('id값')
메서드 호출을 통해 새로운 User 객체를 만든다.
console.log(user01);
```

1. Class

1-3. static method and property (정적 메서드와 정적 프로퍼티)

1-3-2. static property (정적 프로퍼티)

- 스펙에 추가된 지 얼마 안 되는 문법으로 데이터를 클래스 수준에 저장하고 싶을 때 사용한다.
- Animal 클래스 선언

```
class Animal {

    static planet = "지구";

    constructor(name, weight) {
        this.name = name;
        this.weight = weight;
    }

    eat(foodWeight) {
        this.weight += foodWeight;
        console.log(`${this.name}(은)는 ${foodWeight}kg의 식사를 하고 ${this.weight}kg이 되었습니다.`);
    }

    move(lostWeight) {
        if (this.weight > lostWeight)
            this.weight -= lostWeight;
        console.log(`${this.name}(은)는 움직임으로 인해 ${lostWeight}kg 감량되어 ${this.weight}kg이
되었습니다.`)
    }

    static compare(animalA, animalB) {
        return animalA.weight - animalB.weight;
    }

}

Animal.staticProperty = 'static을 사용한 선언은 기술적으론 클래스 자체에 직접 할당하는 것과 동일하다.';
```

1. Class

1-3. static method and property (정적 메서드와 정적 프로퍼티)

1-3-2. static property (정적 프로퍼티)

- 정적 프로퍼티와 정적 메서드는 상속이 가능하다.
- class B extends A는 클래스 B의 프로토타입이 클래스 A를 가리키게 하므로 B에서 원하는 프로퍼티나 메서드를 찾지 못하면 A로 검색이 이어진다.

1. Class

1-3. static method and property (정적 메서드와 정적 프로퍼티)

1-3-2. static property (정적 프로퍼티)

- Animal을 상속받는 Human 클래스

```
class Human extends Animal {

    develop(language) {
        console.log(`${this.name}(은)는 ${language}로 개발을 합니다. 정말 즐겁습니다^.^`);
    }

}

let humans = [
    new Human("홍길동", 70),
    new Human("선덕여왕", 50),
    new Human("신사임당", 60)
];

humans.sort(Human.compare);   // 체중 오름차순으로 정렬

humans[0].develop('JavaScript');

console.log(Human.planet);            // 정적 프로퍼티가 상속 되었다.
console.log(Human.staticProperty);    // 직접 할당한 경우도 동일하게 동작한다.

console.log(Human.__proto__ === Animal); // true - 정적 메서드 존재
console.log(Human.prototype.__proto__ === Animal.prototype); // true - 인스턴스 메서드 존재
```