삼육대학교 SW융합교육원 김진호

자바스크립트 기초교육

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

1-1. Class basic syntax (클래스 기본 문법)

1-1-1. class declarations

1-1-2. class expression

1-1-3. getter setter

1-1-4. public field declarations

1. Class

1-1. class basic syntax (클래스 기본 문법)

동일한 종류의 객체를 여러 개 생성해야 하는 경우 객체 리터럴을 여러 개 생성하기 보다 클래스 문법을 통해 동일한 종류의 객체를 재생성할 수 있다.

1-1-1. class declarations (클래스 선언)

- Student 클래스를 선언하고 new Student()를 호출하면 새로운 객체가 생성되며 넘겨받은 인수 name과 함께 constructor가 자동으로 실행되어 "홍길동"이 this.name에 할당된다.
- 클래스 몸체에 정의한 메서드는 기본적으로 프로토타입 메서드가 된다.

1. Class

1-1. class basic syntax (클래스 기본 문법)

1-1-1. class declarations (클래스 선언)

```
class Student {

    // 생성자를 통해 인스턴스 생성 및 초기화
    // 생성자는 1개 이상 정의될 수 없으며 생략할 경우 암묵적으로 정의된다.
암묵적으로 this를 반환하므로 반환문은 작성하지 않는다.
    constructor(name) {
        this.group = 1;          // 고정 값으로 인스턴스 초기화
        this.name = name;        // 인수로 인스턴스 초기화
    }

    // 프로토타입 메서드
    introduce() {
        console.log(`안녕하세요 저는 ${this.group}반 학생 ${this.name} 입니다.`);
    }

    // 정적 메서드 - 03_static-method-and-property에서 다룬다.
}

let student = new Student("홍길동");     // 인수로 초기값 전달하며 객체 생성
student.introduce();                     // 메서드 호출

console.log(typeof Student);             // function - 클래스는 함수의 한 종류이다.
console.log(Student === Student.prototype.constructor);
// true - 정확하게는 생성자 메서드와 동일하다.
console.log(Student.prototype.introduce);
// [Function: introduce] - 클래스 내부에 정의한 메서드는 클래스.prototype에 저장된다.
console.log(Object.getOwnPropertyNames(Student.prototype)); // [ 'constructor', 'introduce' ]
- 현재 프로토타입에는 constructor와 introduce 두 개의 메서드가 있다.
```

1. Class

1-1. class basic syntax (클래스 기본 문법)

1-1-1. class declarations (클래스 선언)

- 정리하면 new Student()를 호출하면 Student라는 이름을 가진 함수를 만들고 함수 본문은 생성자 메서드 constructor에서 가져온다.
- 만약 생성자 메서드가 없으면 본문이 비워진 채로 함수가 만들어진다. introduce와 같이 클래스 내에서 정의한 메서드를 Student.prototype에 저장한다.
- 클래스 문법과 유사하게 기능하는 것처럼 보이는 생성자 함수를 사용할 수도 있다.

1. Class

1-1. class basic syntax (클래스 기본 문법)

1-1-1. class declarations (클래스 선언)

```
function Teacher(name) {
    this.group = 1;
    this.name = name;
}

Teacher.prototype.introduce = function () {
    console.log(`안녕하세요 저는 ${this.group}반 교사 ${this.name}입니다.`);
}

let teacher = new Teacher("유관순");
teacher.introduce();

// Student(); // 에러 발생 - TypeError: Class constructor Student cannot
be invoked without 'new'
Teacher();  // 에러 발생하지 않음

for (method in Student) {
    console.log('반복문 : ' + method);    // 출력되지 않음
}
```

1. Class

1-1. class basic syntax (클래스 기본 문법)

1-1-1. class declarations (클래스 선언)

■ 생성자 함수와 클래스의 차이점

1. 클래스 생성자를 new와 함께 호출하지 않으면 에러가 발생한다.
   함수 내부 프로퍼티 [[IsClassConstructor]] : true 가 사용된다.
2. 클래스에 정의된 메서드는 열거 불가하다. enumerable 플래그가 false이기 때문이다.
   따라서 for..in 으로 객체 순회 시 메서드 순회 대상에서 제외된다.
3. 클래스는 호이스팅이 발생하지 않는 것처럼 동작한다.
   함수 선언문 생성자 함수는 함수 호이스팅, 함수 표현식 생성자 함수는 변수 호이스팅이 발생한다.
4. 클래스는 항상 use strict 적용된다. 클래스 생성자 안 코드는 자동으로 엄격 모드가 적용된다.

■ 따라서 클래스 문법은 생성자 함수와는 다른 새로운 객체 생성 매커니즘이다.

1. Class

1-1. class basic syntax (클래스 기본 문법)

1-1-2. class expression (클래스 표현식)

익명 클래스 표현식

```
let Tutor = class {
    teach() {
        console.log('이해하셨나요~?');
    }
};

new Tutor().teach();
```

1. Class

1-1. class basic syntax (클래스 기본 문법)

1-1-2. class expression (클래스 표현식)

기명 클래스 표현식

```
let Tutee = class MyTutee {
    learn() {
        console.log('우와~ 이해했어요!');
        console.log(MyTutee); // MyTutee라는 이름은 오직 클래스 안에서만 사용할 수 있다.
    }
};

new Tutee().learn();
// console.log(MyTutee); // ReferenceError: MyTutee is not defined, MyTutee는 클래스 밖에서 사용할 수 없다.
```

1. Class

1-1. class basic syntax (클래스 기본 문법)

1-1-2. class expression (클래스 표현식)

클래스 동적 생성

```
function makeTutee(message) {
    // 클래스를 선언하고 이를 반환한다.
    return class {
        feedback() {
            console.log(message);
        };
    };
}

let SecondTutee = makeTutee("10점 만점에 10점~!");
new SecondTutee().feedback();
```

- 클래스 표현식 정의가 가능하다는 것의 의미는 함수처럼 클래스도 일급 객체이며 다른 표현식 내부에서 정의, 전달, 반환, 할당이 가능하다는 것이다.

1. Class

1-1. class basic syntax (클래스 기본 문법)

1-1-3. getter setter

- 접근자 프로퍼티는 프로토타입의 프로퍼티가 된다.
  name, price로 외부 접근하고 _name, _price로 실제 값을 담는다.
- 밑줄은 프로그래머들 사이에서 외부 접근이 불가능한 프로퍼티나 메서드를 나타낼 때 사용한다.

1. Class

1-1. class basic syntax (클래스 기본 문법)

1-1-3. getter setter

```
class Product {

    constructor(name, price) {
        // setter를 활성화한다.
        this.name = name;
        this.price = price;
    }

    // getter 함수
    get name() {
        console.log('get name 동작');
        return this._name;
    }

    get price() {
        console.log('get price 동작');
        return this._price;
    }

    // setter 함수
    set name(value) {
        console.log('set name 동작');
        this._name = value;
    }

    set price(value) {
        console.log('set price 동작');
        if (value <= 0) {
            console.log("가격은 음수일 수 없습니다.");
            this._price= 0;
            return;
        }
        this._price= value;
    }

}

let phone = new Product('전화기', 23000);
console.log(phone.name + "," + phone.price);

let computer = new Product('컴퓨터', -1500000);
console.log(computer.name + "," + computer.price);
```

1. Class

1-1. class basic syntax (클래스 기본 문법)

1-1-4. public field declarations (클래스 필드 정의 제안)

- 클래스를 정의할 때 '프로퍼티 이름 = 값'을 써주면 클래스 필드를 만들 수 있다.
- 최신 브라우저 (Chrome 72 이상) 또는 최신 Node.js(버전 12 이상)에서만 실행 가능하다.
- 참고로 클래스 필드에 함수를 할당하면 인스턴스 함수가 되므로 이는 권장되지 않는다.

```
class Book {

    name = "모던JavaScript";
    // this.price = 35000; // 문법 오류 : this.은 constructor 내부 또는
메소드 내부에서 작성해야 한다.
    price;

    introduce() {
        console.log(`${this.name}(이)가 그렇게 재밌죠~`);
    }
}

let book = new Book();
book.introduce();
console.log(book.name);
console.log(Book.prototype.name); // Book.prototype이 아닌 개별 객체에만
클래스 필드가 설정
console.log(book.price);     // undefined
```