JS-중급 13

태그	
날짜	@2023년 6월 7일

Class

: ES6에 추가된 스펙

```
const User = function (name, age) {
  this.name = name;
  this.age = age;
  this.showName = function(){
    console.log(this.name);
  };
};

const mike = new User("Mike", 30);

-----------------------------------------
class User2{
  constructor(name, age){
    this.name = name;
    this.age = age;
  }
  showName(){
    console.log(this.name);
  }
}

const tom = new User2("Tom",19);

뉴를 통해서 호출했을때 내부에서 정의된 내용으로 객체를
생성하는것은 동일하다.

Class 키워드 사용.
내부에 constructor가 있음
constructor는 객체를 만들어주는 생성자 메소드.
new를 통해 호출하면 자동으로 실행된다.
객체를 초기화하기 위한 값이 정의된다.
이렇게 인수를 넘겨받을 수 있다.

객체의 name과 age가 생성된다.
```

```
showName처럼 class내에 정의한 메소드는 user2에
프로토타입에 저장된다.

class는 new없이 실행할 수 없다.
```

```
class : 상속

//extends

class Car {
  constructor (color){
    this.color = color;
    this.wheels = 4;
  }
  drive(){
    console.log("drive..");
  }
  stop() {
    console.log("stop!");
  }
}

class Bmw extends Car{
  park(){
    console.log("park")
  }
}

const z4 = new Bmw("blue");
```

```
Class : 메소드 오버라이딩(method overriding)

class Car {
  constructor (color){
    this.color = color;
    this.wheels = 4;
  }
  drive(){
    console.log("drive..");
  }
  stop() {
    console.log("stop!");
```

```
  }
}

class Bmw extends Car{
  park(){
    console.log("park")
  }
  stop() {
    console.log("OFF!");
  }
}

const z4 = new Bmw("blue");

z4.stop();
"OFF!"

만약 부모의 메소드를 그대로 사용하면서 확장하고 싶다면

class Car {
  constructor (color){
    this.color = color;
    this.wheels = 4;
  }
  drive(){
    console.log("drive..");
  }
  stop() {
    console.log("stop!");
  }
}

class Bmw extends Car{
  park(){
    console.log("park")
  }
  stop() {
    super.stop();
    console.log("OFF!");
  }
}

const z4 = new Bmw("blue");

// super. 키워드 사용
```

```
// 오버라이딩 overriding

class Car {
```

```
  constructor (color){
    this.color = color;
    this.wheels = 4;
  }
  drive(){
    console.log("drive..");
  }
  stop() {
    console.log("stop!");
  }
}

class Bmw extends Car{
  constructor(color){
    super(color);
    this.navigation = 1;
  }
  part(){
    console.log("park");
  }
}
```

constructor에서 this를 사용하게 전에 부모 생성자를 반드시 먼저 호출해야한다.
class의 constructor는 빈 객체로 만들어지게 되고 this로 이 객체를 가르키게 된다

반면 extends를 써서 만든 자식 클래스는 빈 객체가 만들어지고 this의 할당하는 작업을 건너뛴다.

그래서 항상 super 키워드로 부모 클래스의 constructor를 할당해야한다.