11강_객체-심화

11-1 객체를 만드는 다양한 방법
11-2 prototype(프로토타입)
11-3 getter, setter 함수

- 2017. 09 / mculecture@gmail.com

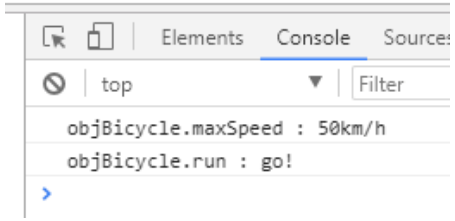
11-1 : 객체를 만드는 다양한 방법

Ex : 11_01.html

기본적인 객체 생성

```
var 객체변수명 = {
    key1 : value1,
    key2 : value2,
    key3 : value3,
    key4 : value4,
    key5 : value5,
    …
    key : value
};
```

```
//기본적인 객체 생성
var objBicycle = {
    handle : "steel",
    maxSpeed : "50km/h",
    width : "200cm",
    height: "100cm",
    run : function() {
        return "go!";
    }
}

console.log("objBicycle.maxSpeed : " + objBicycle.maxSpeed);
console.log("objBicycle.run : " + objBicycle.run());
```



11-1 : 객체를 만드는 다양한 방법

Ex : 11_01.html

함수를 이용한 객체 생성

```
function 함수명() {
        var 객체명 = {
                ...
        };

        return 객체명;
};
```

```
// 함수를 이용한 객체 생성
function createCar(name, color, speed) {
    var carObj = {
        name : name,
        color : color,
        speed : speed,
        run : function() {
            return this.speed + "km/h";
        }
    };

    return carObj;
};

var sorento = createCar("SORENTO", "GREY", 220);
console.log("sorento.name : " + sorento.name);
console.log("sorento.run : " + sorento.run());
```

```
sorento.name : SORENTO
sorento.run : 220km/h
```

11-1 : 객체를 만드는 다양한 방법

Ex : 11_01.html

생성자를 이용한 객체 생성

```
function 생성자 함수명() {
        key1 = value1;
        key2 = value2;
        ...
        key = value
};

new 생성자 함수명();
```

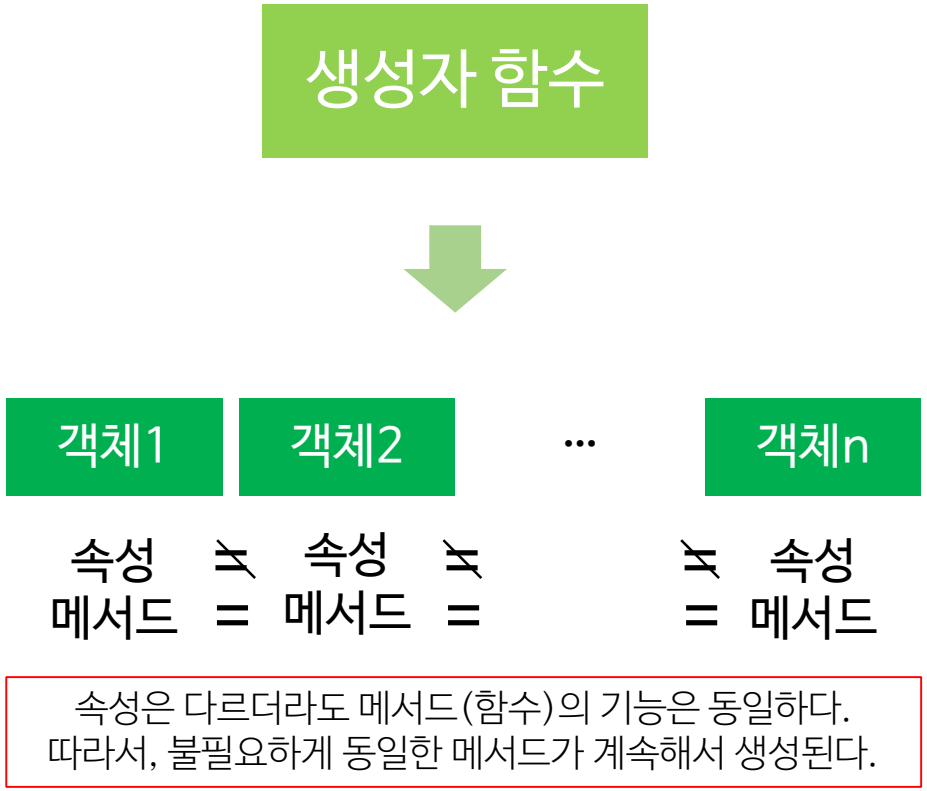
```
// 생성자를 이용한 객체 생성
function  Airplane(name, color, speed){

    this.name = name;
    this.color = color;
    this.speed = speed;
    this.fly = function() {
        return this.speed + " fly!"
    };

};

var boeing747 = new Airplane("boeing747", "white/blue", "600km/h")
console.log("boeing747.name : " + boeing747.name);
console.log("boeing747.color : " + boeing747.color);
console.log("boeing747.fly : " + boeing747.fly());
```

```
boeing747.name : boeing747
boeing747.color : white/blue
boeing747.fly : 600km/h fly!
```

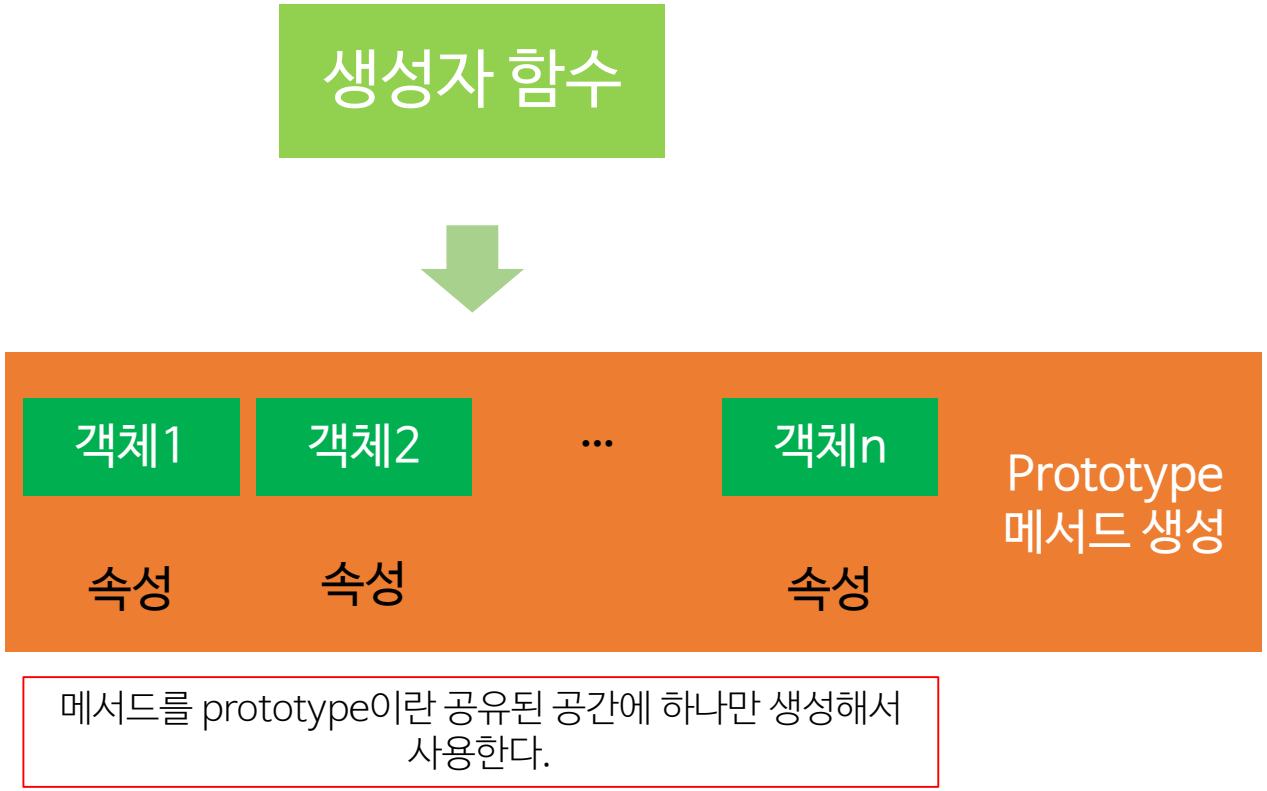
11-2 : prototype(프로토타입)

Ex : 11_02.html, 11_03.html

javascript에만 있는 특징으로 prototype이란 공유된 공간



속성은 다르더라도 메서드(함수)의 기능은 동일하다.
따라서, 불필요하게 동일한 메서드가 계속해서 생성된다.



메서드를 prototype이란 공유된 공간에 하나만 생성해서
사용한다.

11-2 : prototype(프로토타입)

Ex : 11_02.html, 11_03.html

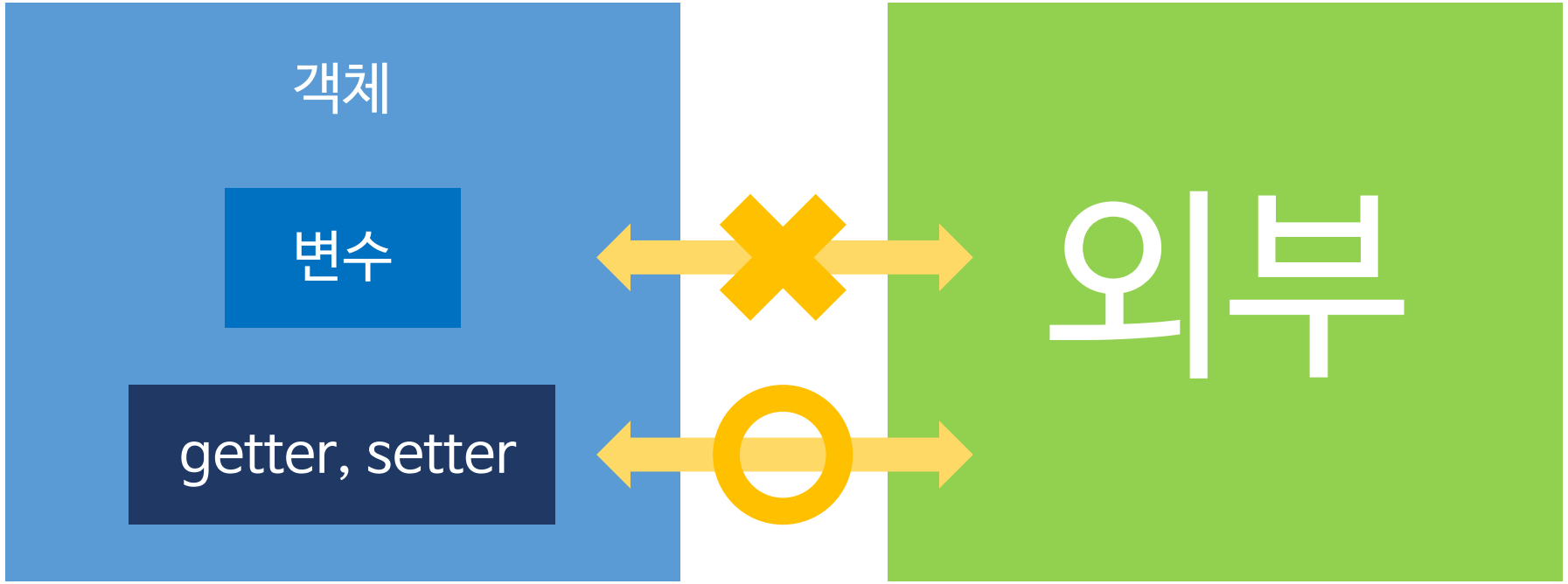
```
// 생성자를 이용한 객체 생성
function  Scoring(player, scoreFirst, scoreSecond, scoreThird){

    this.player = player;
    this.scoreFirst = scoreFirst;
    this.scoreSecond = scoreSecond;
    this.scoreThird = scoreThird;

    this.getTotal = function() {
        return this.scoreFirst + this.scoreSecond + this.scoreThird;
    };

    this.getAverage = function() {
        return (this.getTotal() / 3).toFixed(3);
        //Number.prototype.toFixed();
        //toFixed() 메서드는 고정 소수점 표현법
        //toFixed(3) : xx.123 까지 표현
    };

};
```

```
// 생성자를 이용한 객체 생성
function  Scoring(player, scoreFirst, scoreSecond, scoreThird){

    this.player = player;
    this.scoreFirst = scoreFirst;
    this.scoreSecond = scoreSecond;
    this.scoreThird = scoreThird;

};

Scoring.prototype.getTotal = function() {
    return this.scoreFirst + this.scoreSecond + this.scoreThird;
};

Scoring.prototype.getAverage = function() {
    return (this.getTotal() / 3).toFixed(3);
};
```

11-3 : getter, setter 함수

Ex : 11_04.html, 11_05.html

외부에서 객체 내부의 변수에 접근하기 위한 함수(내부함수 보호)



11-3 : getter, setter 함수

Ex : 11_04.html, 11_05.html

```
/*
    getter, setter 함수
*/

function BMICalculator (height, weight){
    this.height = height;
    this.weight = weight;
    this.bmi = function(){
        return (this.weight / (this.height * this.height)).toFixed(2);
        //Number.prototype.toFixed();
    };
}

var myBMI = new BMICalculator(1.9, 90);
console.log("myBMI.bmi : " + myBMI.bmi());
```

```
/*
    getter, setter 함수
*/

function BMICalculator (){

    var height = 0;
    var weight = 0;

    this.bmi = function(){
        return (this.weight / (this.height * this.height)).toFixed(2);
        //Number.prototype.toFixed();
    };

    this.getHeight = function() {
        return this.height;
    };

    this.setHeight = function(height) {
        if(!isNaN(height)) {
            this.height = height;
        } else {
            console.log("height is NaN(Not a Number)!");
        };
    };

    this.getWeight = function() {
        return this.weight;
    };

    this.setWeight = function(weight) {
        if(!isNaN(weight)) {
            this.weight = weight;
        } else {
            console.log("weight is NaN(Not a Number)!");
        };
    };
}

var myBMI = new BMICalculator();
myBMI.setHeight(1.9);
myBMI.setWeight(90);
console.log("myBMI.bmi : " + myBMI.bmi());
```