**객체생성자**

-다음과 같이 new를 사용해 새 객체를 만들 수 있음

function Animal(type, name, sound) {

this.type = type;

this.name = name;

this.sound = sound;

this.say = function() {

console.log(this.sound);

};

}

const dog = new Animal(“개”, “멍멍이”, “멍멍”);

const cat = new Animal(“고양이”, “야옹이”, “야옹”);

dog.say();

cat.say();

//결과

멍멍

야옹

**프로토타입**

* 객체 생성자 함수 아래에 “ .prototype.[원하는키] = 코드 “ 형식으로 입력을 설정할 수 있음

Animal.prototype.say = function() {

console.log(this.sound);

}

**객체생성자 상속**

* 다음처럼 .call 함수를 통해 객체 생성자를 상속받을 수 있다.
* call 함수 첫번째엔 this를 넣어주고, 다음엔 그 호출 객체 생성자 함수에서 필요한 것들을 넣어준다.
* prototype도 그냥 할당연산자로 받을 수 있음

function Dog(name, sound) {

Animal.call(this, “개”, name, sound);

}

Dog.prototype = Animal.prototype

**클래스**

1. ES6부터 추가된 class 문법

//기본 사용법

class Animal {

constructor(type, name, sound) {

this.type = name;

this.name = name;

this.sound = sound;

}

say() {

console.log(this.sound);

}

}

const Dog = new Animal(“개”, “멍멍이”, “멍멍”);

dog.say();

1. 상속

class Dog extends Animal {

constructor(name, sound) {

super(“개”, name, sound);

}

};