04

04-3. 프로토 타입

JavaScript Object Oriented Programming

프로토타입

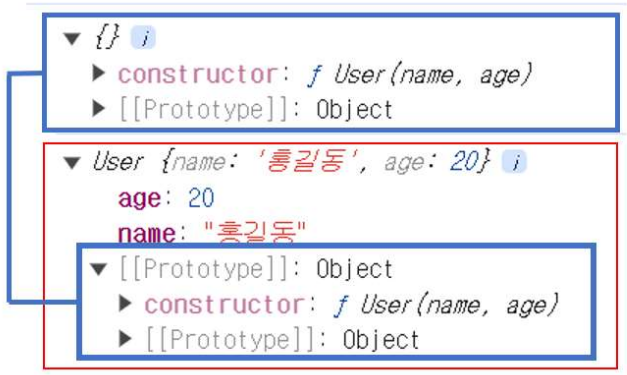
- "프로토타입"이라는 단어의 일반적인 의미는 어떤 제품을 만들기 전의 시제품이라는 의미가 있습니다.
- 자바스크립트에서 프로토타입은 객체의 시제품 정도로 이해해 주면 됩니다.

- 자바스크립트에서 함수를 선언하면 자동으로 그 함수를 위한 프로토타입 객체가 만들어 집니다.
- 여기서 중요한 점은 함수를 이용해 객체를 생성해야 만들어지는 것이 아니라 함수를 선언하는 것만으로 자동으로 프로토타입 객체가 만들어 지는 것입니다.

프로토타입

프로토타입 객체

```
1   function User(name, age){
2     this.name = name
3     this.age = age
4   }
5   console.log(User.prototype)
6   console.log(new User('홍길동', 20))
```



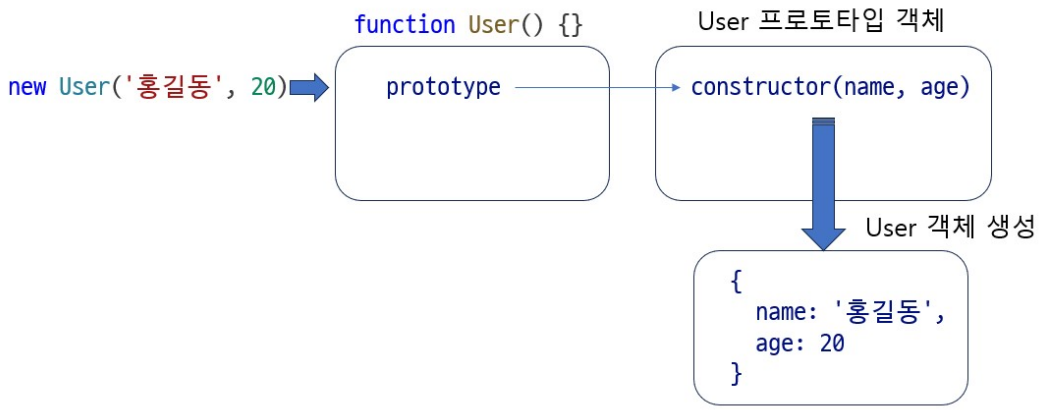
User.prototype

new User('홍길동', 20)

프로토타입

- 프로토타입 객체에 자동으로 constructor 라는 생성자가 추가되어 있습니다.
- 자바스크립트에서 함수를 이용해 객체를 생성한다는 의미는 사실 내부적으로 프로토타입의 생성자를 이용하는 것입니다.

- User 라는 이름의 함수가 있고 이 함수를 이용해 new User() 로 객체를 생성하게 되면 내부적으로는 User 함수의 프로토타입 객체내의 constructor() 라는 생성자가 이용되고 이 생성자에 의해 객체가 생성되는 것입니다.



프로토타입

- 프로토타입 객체는 객체 생성을 위한 생성자를 제공하는 것 이외에 멤버를 추가할 수도 있습니다.
- 함수의 객체는 프로토타입 이용해 생성하게 됨으로 프토토타입에 추가된 멤버는 그 함수를 이용해 생성되는 여러 객체에서 필요한 공통의 멤버이며 여러 객체에서 공유하는 멤버입니다.

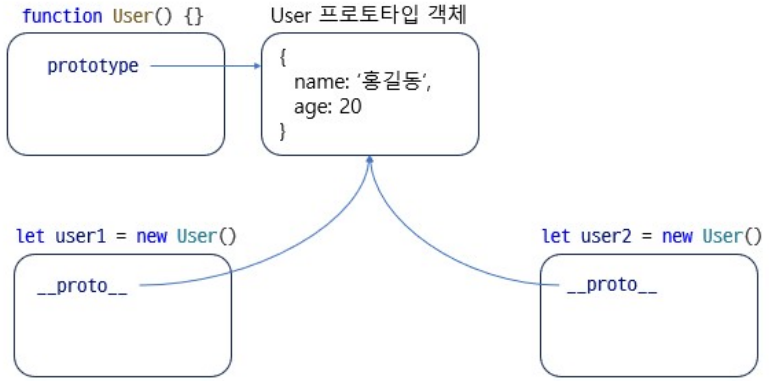
- 프로토타입에 멤버를 등록할 때는 "함수명.prototype.멤버명" 이며 이렇게 등록된 프로토타입의 멤버를 함수의 객체에서 이용할 때는 "객체명.멤버명" 입니다.

프로토타입

```
function User() {
  User.prototype.name = '홍길동'
}
User.prototype.age = 20

let user1 = new User()
let user2 = new User()
console.log(user1.name, user1.age)//홍길동 20
console.log(user2.name, user2.age)//홍길동 20

console.log(user1.__proto__.name, user1.__proto__.age)//홍길동 20
console.log(user2.__proto__.name, user2.__proto__.age)//홍길동 20
```



프로토타입과 this

- 객체를 위한 프로퍼티와 함수가 선언되어야 하는데 객체의 this 멤버로 선언할 수도 있고 프로토타입 멤버로 선언할 수도 있습니다.
- this 멤버나 프로토타입 멤버를 모두 객체에서 이용할 수는 있지만 이 둘은 차이가 있습니다.

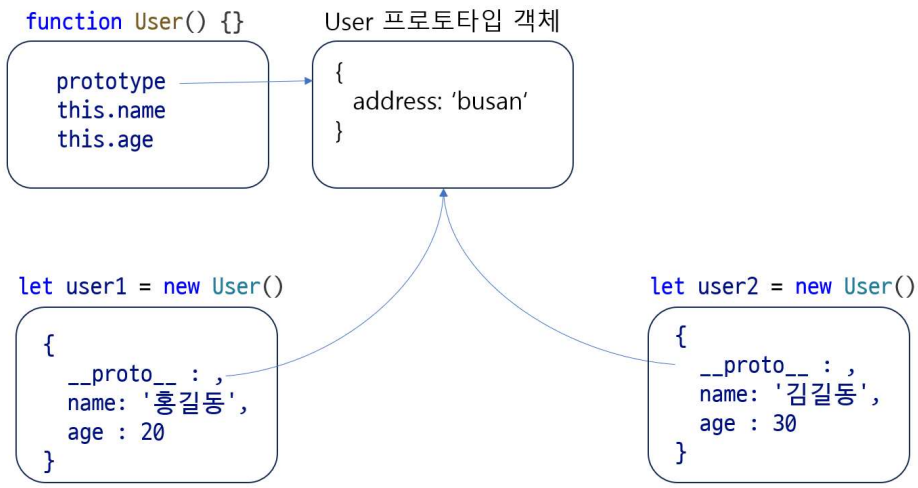
- this 의 멤버는 객체별로 다른 데이터가 유지되지만 프로토타입의 멤버는 여러 객체가 만들어져도 모두 동일 데이터를 가지게 됩니다.

- this 는 객체가 생성될 때마다 객체별로 메모리가 별도로 할당되기 때문에 객체별로 다른 값을 유지할 수 있는 것이며 프로토타입은 여러 개의 객체가 만들어진다고 하더라도 객체끼리 공유되기 때문입니다.

프로토타입과 this

this 와 프로토타입 멤버

```
1   function User(name, age, address){
2     this.name = name
3     this.age = age
4     User.prototype.address = address
5   }
6   let user1 = new User('홍길동', 20, 'seoul')
7   let user2 = new User('김길동', 30, 'busan')
8
9   console.log(user1.name, user1.age, user1.address)//홍길동 20 busan
10  console.log(user2.name, user2.age, user2.address)//김길동 30 busan
```

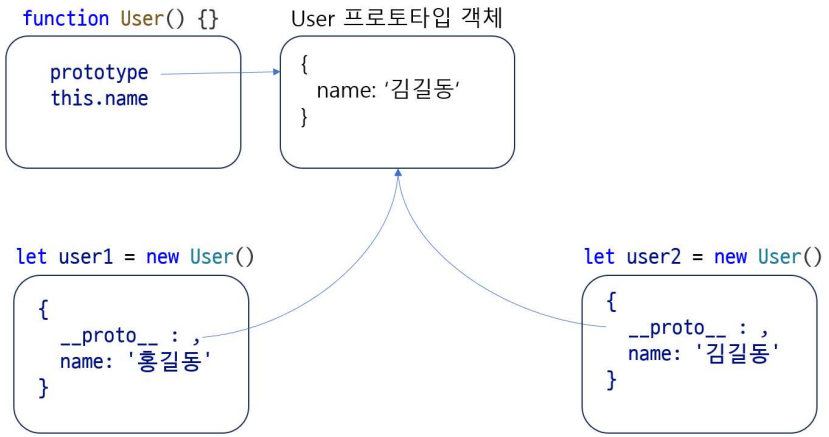


프로토타입과 this

- this 와 프로토타입에 동일 이름의 멤버 선언이 가능하며 이 경우 객체로 멤버에 접근하면 this 에 선언된 멤버가 이용되게 됩니다.
- 객체로 어떤 멤버에 접근하게 되면 자신의 메모리에서 찾게 되고 없는 경우에만 내부 참조를 이용해 프로토타입의 멤버에 접근하게 됩니다.

this 와 프로토타입 멤버 중복

```
1   function User(name){
2     this.name = name
3     User.prototype.name = name
4   }
```



프로토타입과 this

- 프로토타입과 this 에 동일한 이름의 멤버가 선언이 가능하다는 이야기인데, 이를 이용해 객체들끼리 공유해야 하는 멤버를 프로토타입으로 선언한 후에 어떤 특정 객체에서 프로토타입에 선언한 멤버를 다시 자신의 this 에 선언해서 특정 객체만 다른 값 혹은 로직을 가지도록 할 수 있습니다.

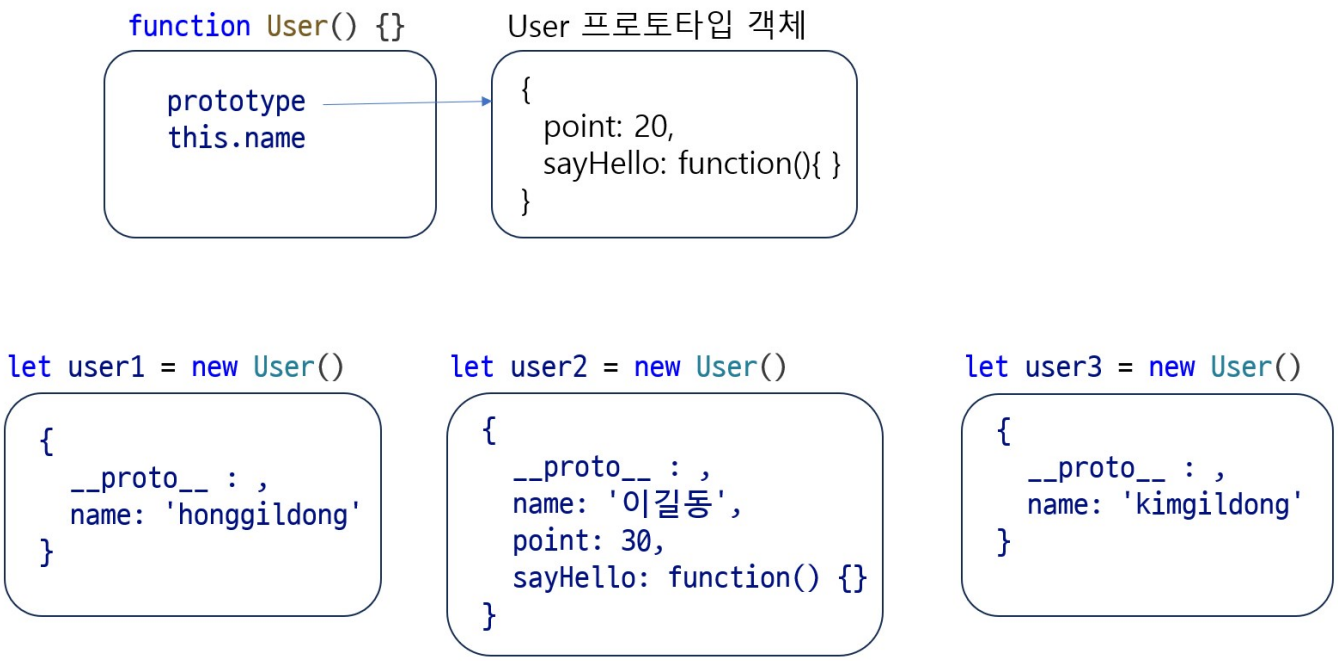
```
function User(name){
  this.name = name
  User.prototype.point = 20
  User.prototype.sayHello = function(){
    console.log(`Hello ${this.name}, point : ${this.point}`)
  }
}

let user1 = new User('honggildong')
user1.sayHello()//Hello honggildong, point : 20

let user2 = new User('이길동')
user2.point = 30
user2.sayHello = function() {
  console.log(`안녕하세요 ${this.name}, 포인트 : ${this.point}`)
}
user2.sayHello()//안녕하세요 이길동, 포인트 : 30

let user3 = new User('kimgildong')
user3.sayHello()//Hello kimgildong, point : 20
```

프로토타입과 this


function User() {}
prototype
this.name
User 프로토타입 객체
{
point: 20,
sayHello: function(){ }
}
let user1 = new User()
{
__proto__ : ,
name: 'honggildong'
}
let user2 = new User()
{
__proto__ : ,
name: '이길동',
point: 30,
sayHello: function() {}
}
let user3 = new User()
{
__proto__ : ,
name: 'kimgildong'
}

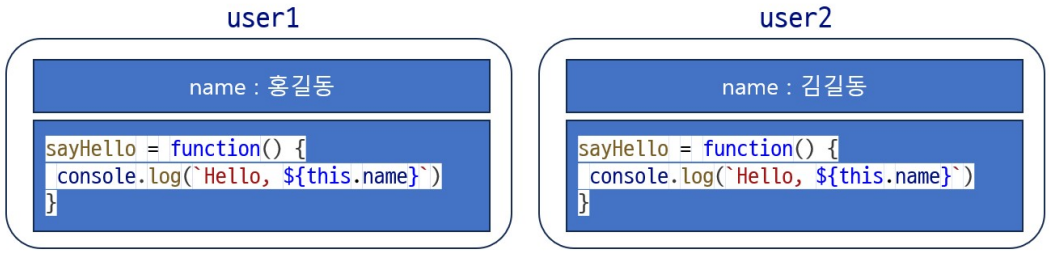
프로토타입 – 메모리 효율성

- 프로토타입을 이용하게 되면 메모리 효율성 측면에서도 이점을 얻을 수 있습니다.
- 주로 이 경우는 프로퍼티보다는 함수를 선언하는 경우입니다.
- 동일 타입의 객체가 여러 개 생성되는 경우 각 객체에 등록되는 함수는 거의 대부분 로직이 동일한 것이 일반적입니다.
- 그런데 이 함수를 this 에 선언하게 되면 동일한 동작을 하는 함수가 객체별로 매번 메모리에 할당되어 메모리가 불필요하게 점유되는 상황이 발생할 수도 있습니다.

프로토타입 – 메모리 효율성

객체 메모리

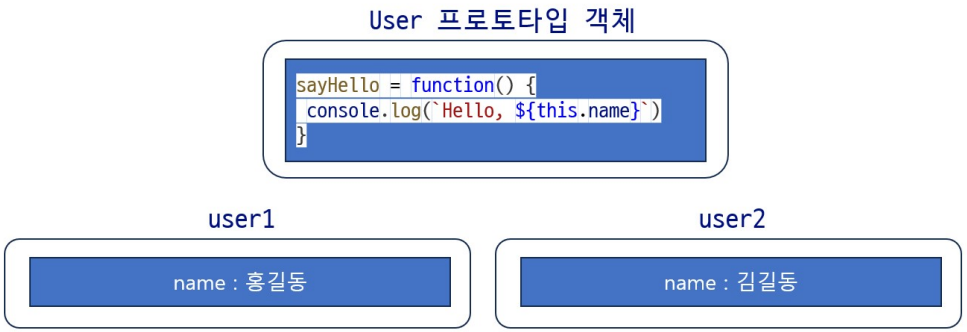
```
1   function User(name){
2     this.name = name
3     this.sayHello = function() {
4       console.log(`Hello, ${this.name}`)
5     }
6   }
7   let user1 = new User('홍길동')
8   let user2 = new User('김길동')
9   user1.sayHello()//Hello, 홍길동
10  user2.sayHello()//Hello, 김길동
11
12  console.log(user1.sayHello == user2.sayHello)//false
```



프로토타입 – 메모리 효율성

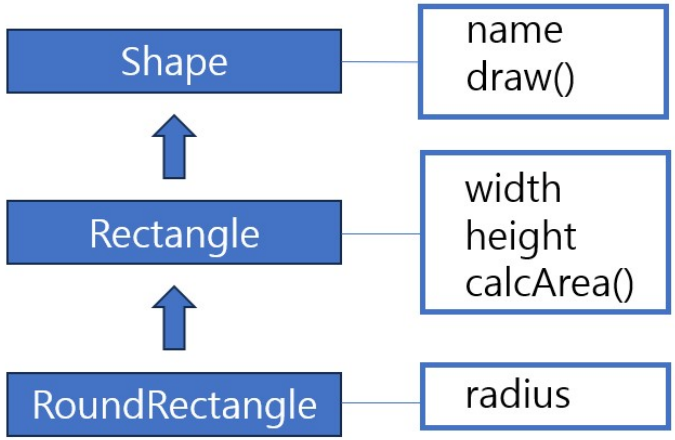
프로토타입으로 함수 공유

```
1   function User(name){
2     this.name = name
3     User.prototype.sayHello = function() {
4       console.log(`Hello, ${this.name}`)
5     }
6   }
7   let user1 = new User('홍길동')
8   let user2 = new User('김길동')
9   user1.sayHello()//Hello, 홍길동
10  user2.sayHello()//Hello, 김길동
11
12  console.log(user1.sayHello == user2.sayHello)//true
```



프로토타입 – 상속 구현

- 객체지향 프로그래밍에서 상속이란 어떤 객체에 선언된 멤버를 그대로 이어받아 새로운 객체를 선언하는 방법을 의미합니다.
- 상속이 주는 이점은 여러가지가 있지만 코드의 재사용 측면에서 유용한 기법이며 객체지향 프로그래밍의 핵심 기법 중 하나입니다.
- 자바스크립트에서 상속을 이용한 코드의 재사용도 프로토타입을 이용해 구현할 수 있습니다.



프로토타입 – 상속 구현

상위 객체를 하위 프로토타입으로 지정

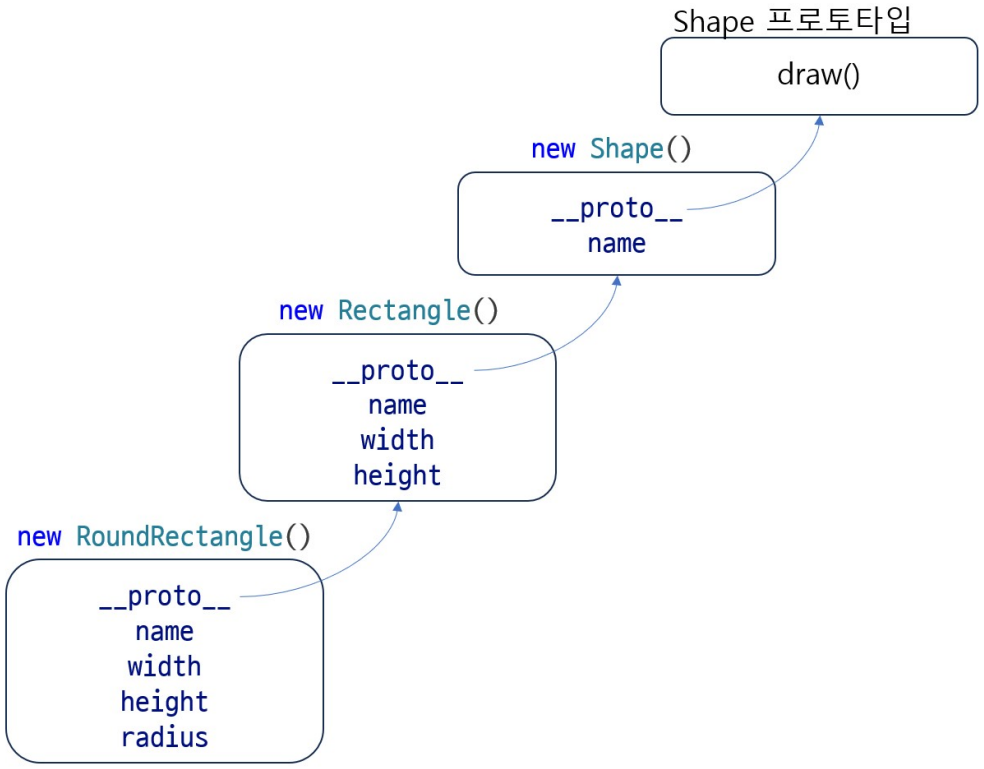
- 프로토타입도 객체입니다.
- 그럼으로 상위 함수의 객체를 하위 함수의 프로토타입으로 지정해서 상위에 선언된 것을 하위에서 이용하게 할 수 있습니다.

```
function Shape(name){
  this.name = name
}
Shape.prototype.draw = function(){
  console.log(`${this.name} 도형을 그립니다.`)
}

function Rectangle(name, width, height){
  //부모의 this 를 자식의 this 에 바인딩(복재)
  Shape.apply(this, [name])
  this.width = width
  this.height = height
}
Rectangle.prototype = new Shape()
```

```
function RoundRectangle(name, width, height, radius){
  Rectangle.apply(this, [name, width, height])
  this.radius = radius
}
RoundRectangle.prototype = new Rectangle()
```

프로토타입 – 상속 구현


Shape 프로토타입
draw()
new Shape()
__proto__
name
new Rectangle()
__proto__
name
width
height
new RoundRectangle()
__proto__
name
width
height
radius

프로토타입 – 상속 구현

상위의 프로토타입을 하위 프로토타입으로 지정

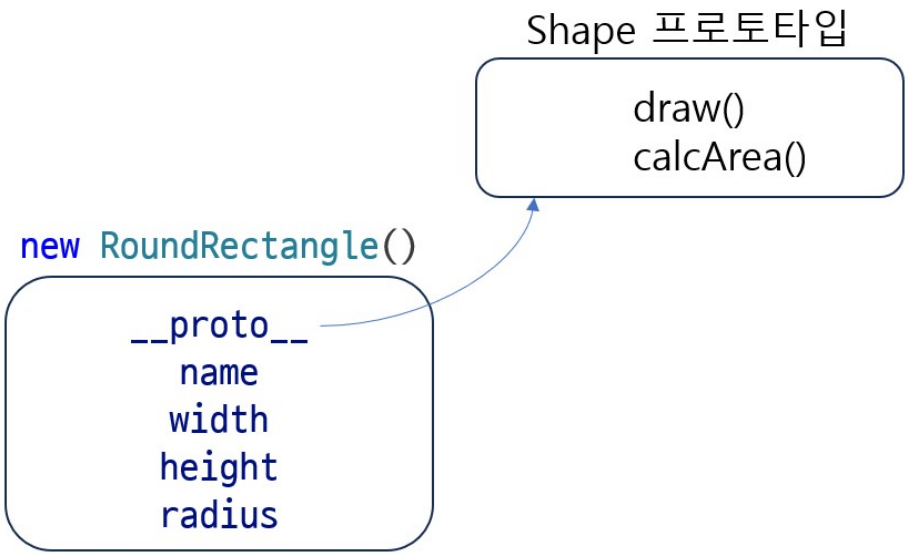
- 상위 객체를 프로토타입으로 이용하는 방법은 상위 객체가 생성되어야 합니다.
- 상위 객체가 생성되어 이용되어야 하거나 아니면 함수의 프로토타입들이 별도로 유지되어야 하는 경우는 이 방법을 이용할 수도 있지만 불 필요하게 객체가 생성될 수도 있는 방법입니다.

- 상위의 프로토타입을 그대로 하위에서 프로토타입으로 이용하게 할 수도 있습니다.

```
function Rectangle(name, width, height){
  Shape.apply(this, [name])
  this.width = width
  this.height = height
}
Rectangle.prototype = Shape.prototype
```

```
function RoundRectangle(name, width, height, radius){
  Rectangle.apply(this, [name, width, height])
  this.radius = radius
}
RoundRectangle.prototype = Rectangle.prototype
```

프로토타입 – 상속 구현


Shape 프로토타입
draw()
calcArea()
new RoundRectangle()
__proto__
name
width
height
radius

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare