■ Class

● 자료와 기능을 담는 템플릿

- 연관 있는 데이터를 한 곳에 묶어 놓는 컨테이너
- 전체적인 모습은 같지만 내부 구성요소의 값은 다를 수 있음

```
class Person {
  name;       // 속성 (field)
  age;        // 속성 (field)
  speak() {} // 행동 (method)
}
```

■ Class

● Class 작성

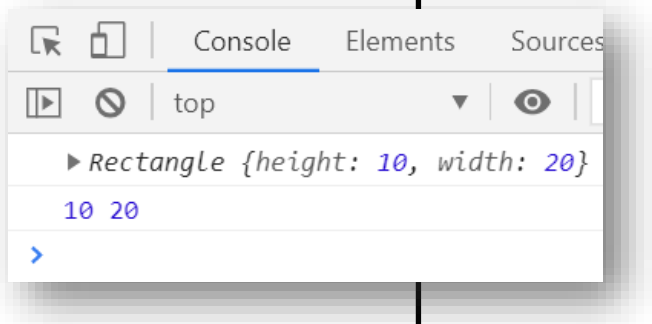
```
// 클래스 선언
class Rectangle {
  // ...
}

// 클래스 표현식 (익명)
const rectange = class {
  // ...
};

// 클래스 표현식 (이름 지정)
const rectange = class Rectangle {
  // ...
};
```

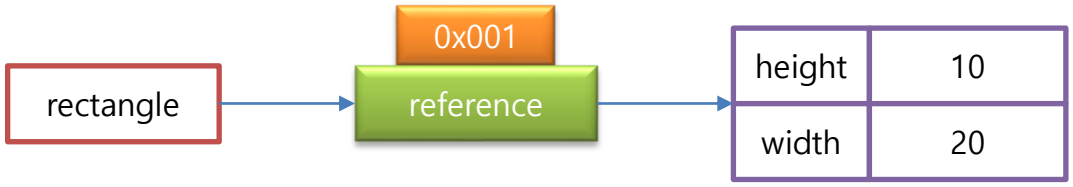
■ Class

● 생성자

- 객체(instance) 생성 시 속성 생성 또는 값을 할당하는 초기화 행위

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
}
const rectangle = new Rectangle(10, 20);
console.log(rectangle);
console.log(rectangle.height, rectangle.width);
```



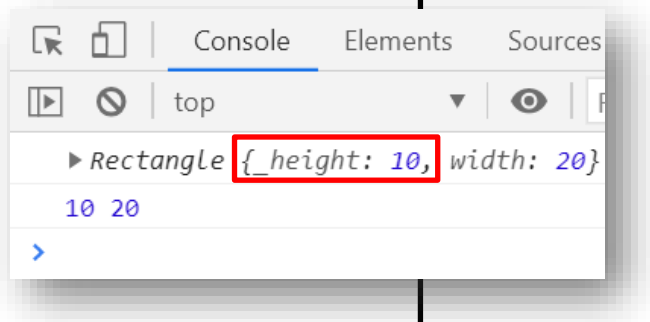


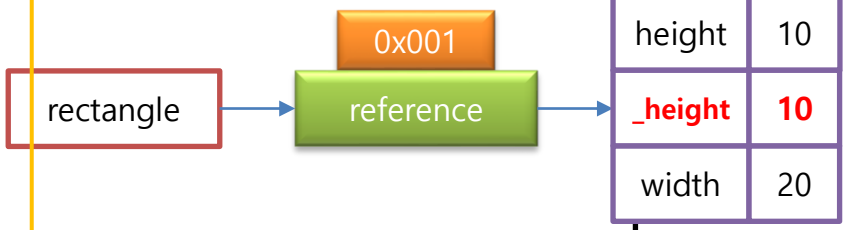
■ Class

● Getter / Setter

- 속성에 값이 입력되거나 조회될 때 동작하는 기능

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }

  get height() {
    return this._height;
  }

  set height(value) {
    this._height = value;
  }
}
const rectangle = new Rectangle(10, 20);
console.log(rectangle);
console.log(rectangle.height, rectangle.width);
```



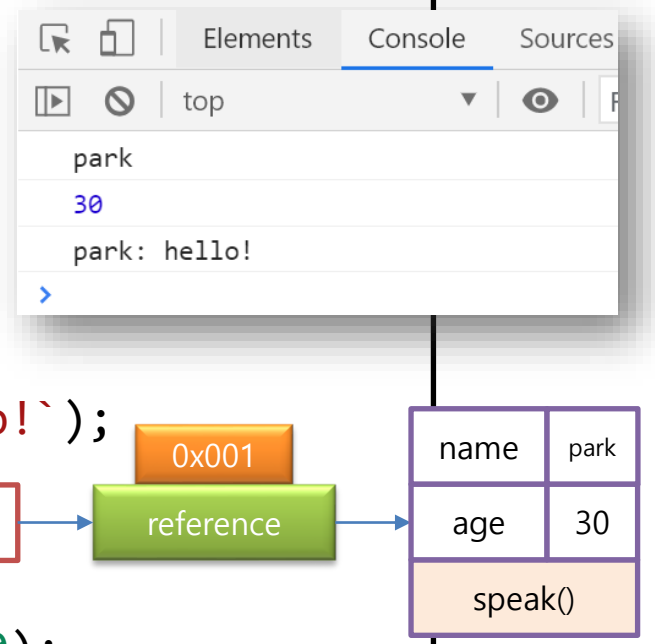


■ Class

● Method

- 클래스 내의 함수

```
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  speak() {
    console.log(`${this.name}: hello!`);
  }
}

const person = new Person('park', 30);
console.log(person.name);
console.log(person.age);
person.speak();
```

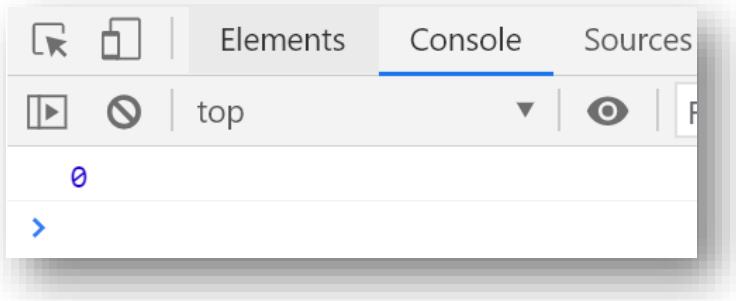


■ 연습문제

● 생성자와 Getter / Setter를 가지는 User 클래스 작성

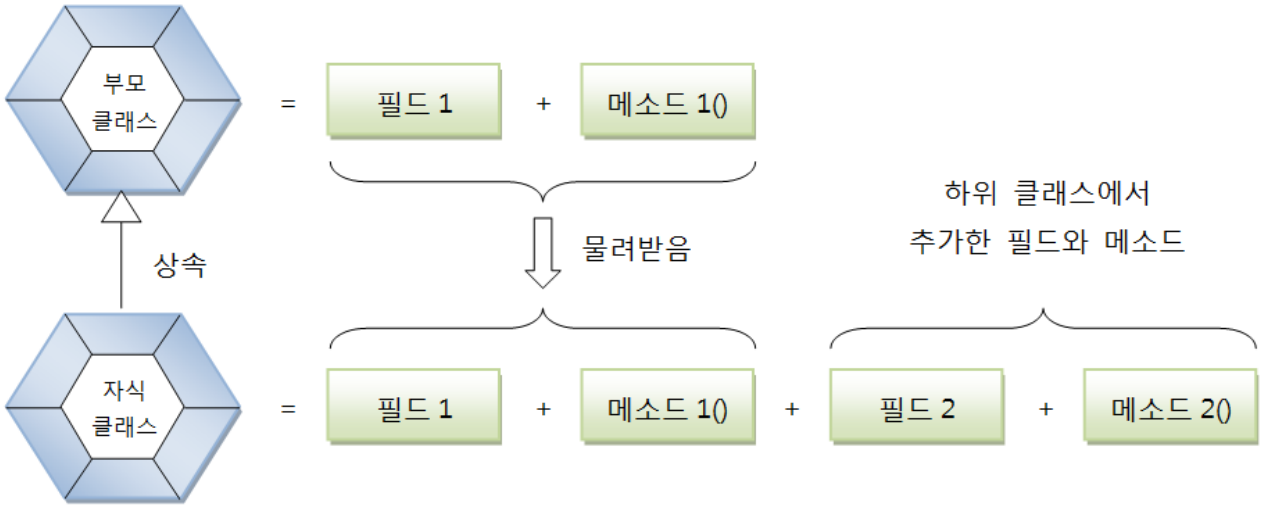
- 생성자 : firstName, lastName, age
- Setter : age에 음수가 입력되면 0으로 저장

```
const user = new User('Steve', 'Job', -1);
console.log(user.age);
```



■ 상속

- 부모가 가진 것을 자식에서 물려주는 행위
  - 부모 클래스 : super class
  - 자식 클래스 : sub class
- 부모 클래스의 생성자, 변수, 메소드를 그대로 사용 가능
  - 필요한 경우 Override 를 통해 변경 가능



■ 상속

● 부모 클래스 (super class)

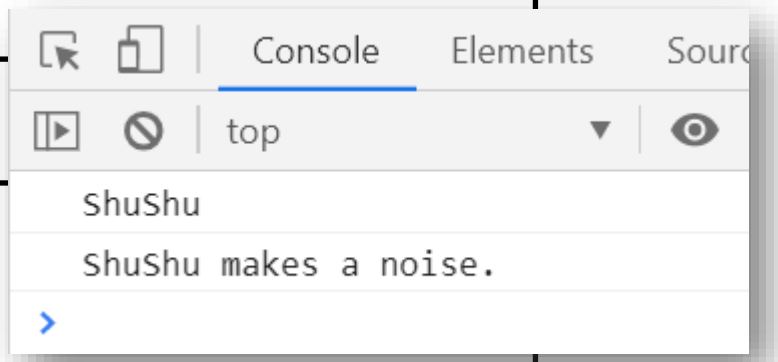
```
class Animal {
  constructor(name) {
    this.name = name;
    console.log();
  }
  speak() {
    console.log(`${this.name} makes a noise.`);
  }
}
```

● 자식 클래스 (sub class)

```
class Dog extends Animal {
  // something
}
```

● 사용

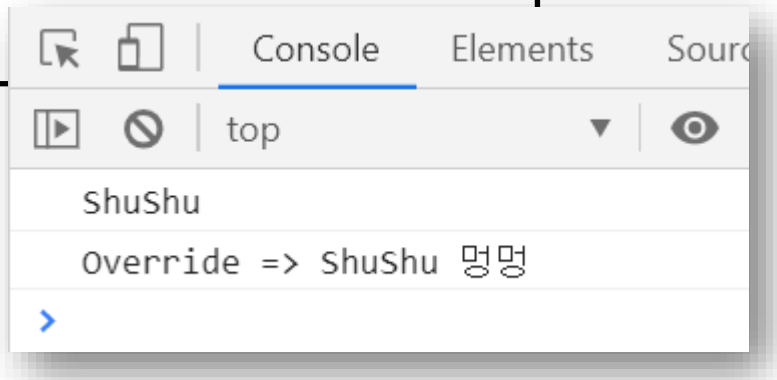
```
const dog = new Dog('ShuShu');
console.log(dog.name);
dog.speak();
```



■ 상속

● Override

```
class Dog extends Animal {
  constructor(name) {
    super(name)
  }
  speak() {
    console.log(`Override => ${this.name} 멍멍`);
  }
}
```

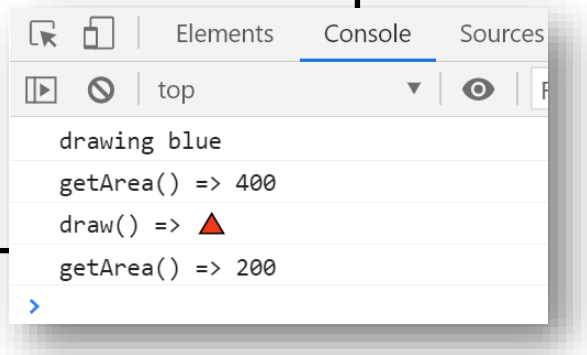


■ 연습문제

● Shape 클래스를 상속받는 Rectangle, Triangle 클래스 작성

- draw(), getArea() 메소드 오버라이딩

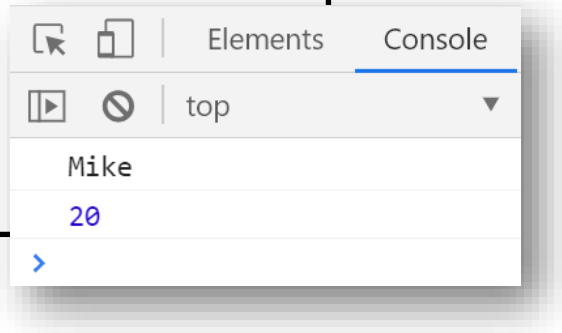
```
class Shape {
  constructor(width, height, color) {
    this.width = width;
    this.height = height;
    this.color = color;
  }

  draw() {
    console.log(`drawing ${this.color}`);
  }

  getArea() {
    return this.width * this.height;
  }
}
```



■ Object

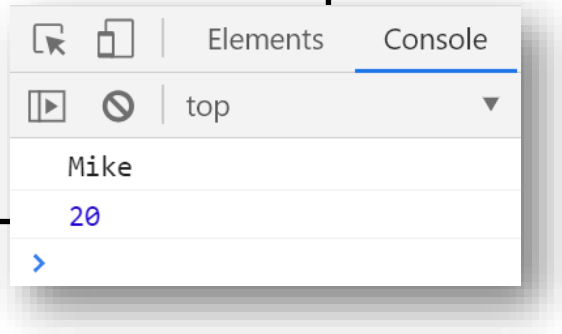
- 주로 데이터 관리를 위해서 사용 (**{ "key" : "value" }**)
- Object 미사용

```
const name = 'Mike';
const age = 20;
function print(name, age) {
  console.log(name);
  console.log(age);
}
print(name, age);
```



- Object 사용

```
const obj = { name: 'Mike', age: 20 };
function print(person) {
  console.log(person.name);
  console.log(person.age);
}
print(obj);
```



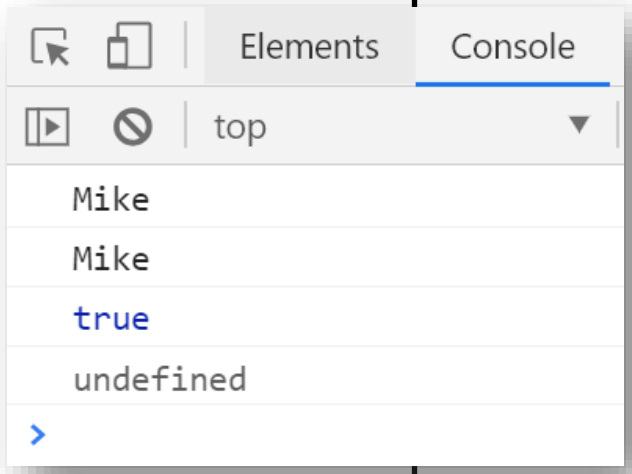
■ Object

● 초기화

```
// 'object literal' syntax
const obj = {};

// 'object constructor' syntax
const obj = new Object();
```

● 속성 제어

```
const obj = { name: 'Mike', age: 20 };

// 속성 조회
console.log(obj.name); // Dot 연산자 사용
console.log(obj['name']); // 대괄호 사용

// 속성 추가 또는 변경
obj.hasJob = false;
obj['hasJob'] = true;
console.log(obj.hasJob);

// 속성 삭제
delete obj.hasJob;
console.log(obj.hasJob);
```



■ Object

● 함수에서 object 반환

```
function makePerson(name, age) {
  return {
    name: name,
    age: age
  }
}
```

=

```
function makePerson(name, age) {
  return {
    name,
    age
  }
}
```

Key와 value에 입력되는 변수명이 같은 경우 생략 가능

● 생성자 함수

```
function Person(name, age) {
  return {
    name, age
  }
}
const p = Person('kim', 30);
console.log(p);
```

≈

```
function Person(name, age) {
  this.name = name;
  this.age = age;
}

const p = new Person('kim', 30);
console.log(p);
```

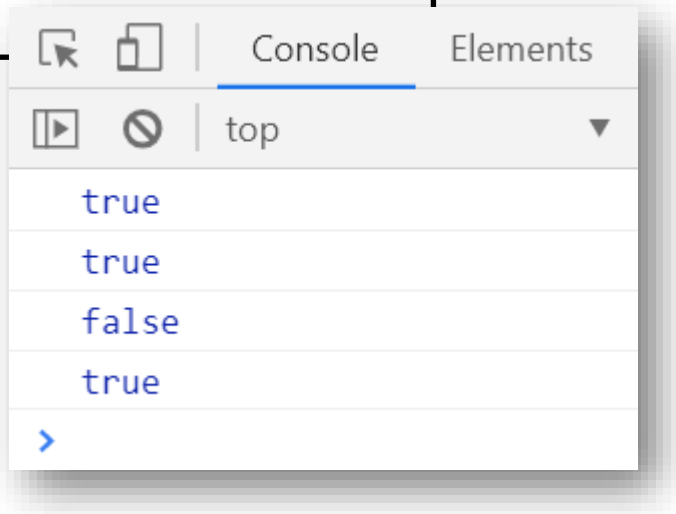
```
▼ {name: "kim", age: 30}
    age: 30
    name: "kim"
  ▶ __proto__: Object
```

```
▼ Person {name: "kim", age: 30}
    age: 30
    name: "kim"
  ▶ __proto__: Object
```

■ Object

● 속성 존재 확인

```
const user = {
  name: 'Bill',
  age: 20,
  hasCar: true
}

console.log('name' in user); // true
console.log('age' in user); // true
console.log('random' in user); // false
console.log(user.hasCar); // true
```



■ Object

● 반복문을 이용한 속성 제어

```
const obj = {
  'pizza': '🍕',
  'chicken': '🍗',
  'meat': '🍖'
};
```

- key를 알아낸 후 value 확인 (for ... in)

```
for(key in obj) {
  console.log(key);
}
```

- 모든 value를 순서대로 하나씩 꺼내어 확인 (for ... of)

```
for(value of array) {
  console.log(value);
}
```

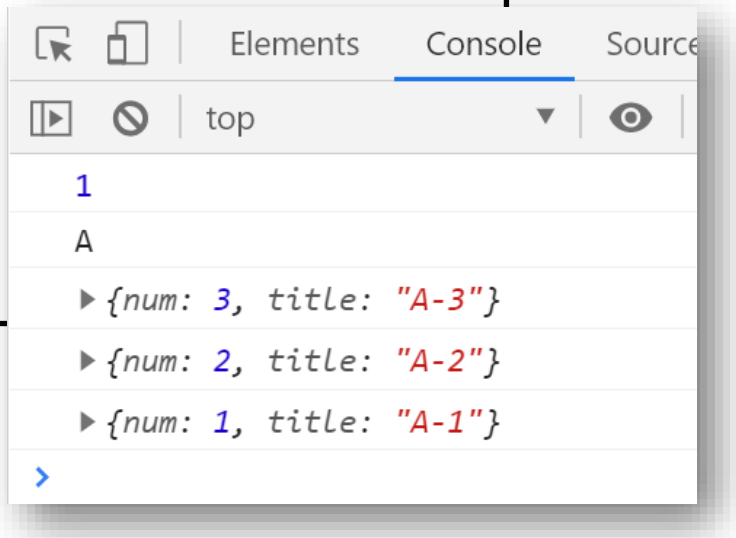
iterable 자료형 일때 적용 가능

■ 연습문제

● 반복문을 사용하여 article 객체가 가지고 있는 모든 속성들의 값 출력하기

- for in

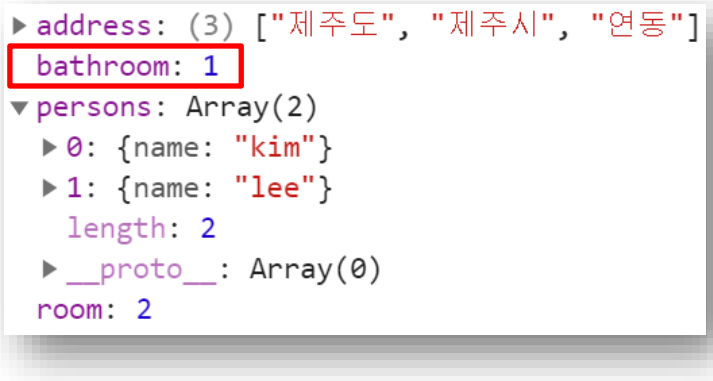
```
const article = {
  num: 1,
  title: 'A',
  reply: [
    { num: 3, title: 'A-3' },
    { num: 2, title: 'A-2' },
    { num: 1, title: 'A-1' }
  ]
};
```

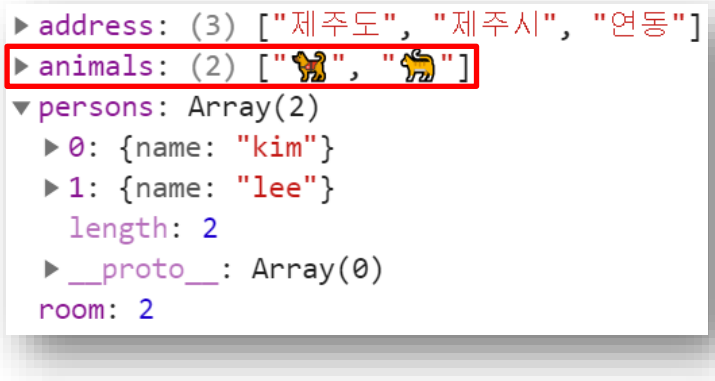


■ 연습문제

● 속성 중 bathroom을 제거하고 animals 배열 ['개', '고양이'] 속성 추가하기

```
const house = {
  bathroom: 1,
  room: 2,
  address: ['제주도', '제주시', '연동'],
  persons: [
    { name: 'kim' },
    { name: 'lee' }
  ]
};
```





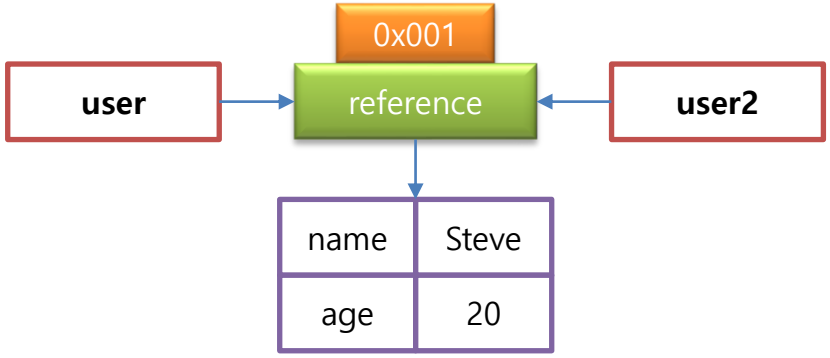
■ Object

● object 복사

- 대입 연산자 사용시 문제점

```
const user = { name: 'Bill', age: '20' };
const user2 = user;
user2.name = 'Steve';
console.log(user.name); // Steve
```

user2 속성의 값을 변경하면 user 속성의 값도 같이 변경



■ Object

● object 복사

- assign()

```
const user3 = {};
for(key in user) {
  user3[key] = user[key];
}
console.log(user3);
                                        // const user4 = {}
                                        // Object.assign(user4, user)
const user4 = Object.assign(user);
console.log(user4);
```

- mix

```
const fruit1 = { color: 'red', price: 'high' };
const fruit2 = { color: 'blue', size: 'big' };
const mixed = Object.assign({}, fruit1, fruit2);
console.log(mixed);
```