5. Class, Object

5-1. Class 작성 방법 및 구성 요소

5-2. 상속

5-3. Object

5-4. 요소 내보내기 / 가져오기

■ 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)
}
```

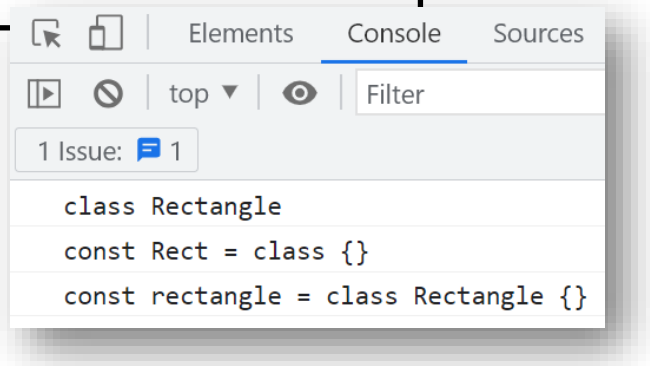
■ Class

● Class 작성방법

05-01-class-작성방법.html

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

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

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

```
const r1 = new Rectangle();
const r2 = new Rect();
const r3 = new rectangle();
```



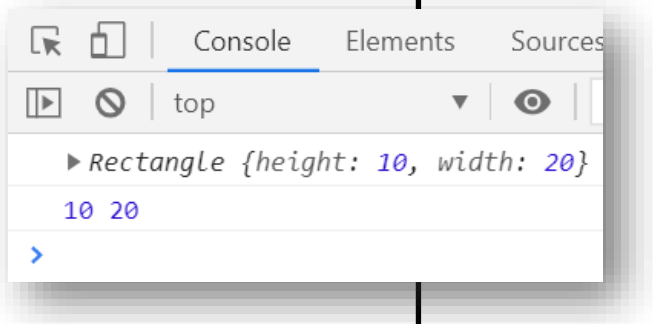
■ Class

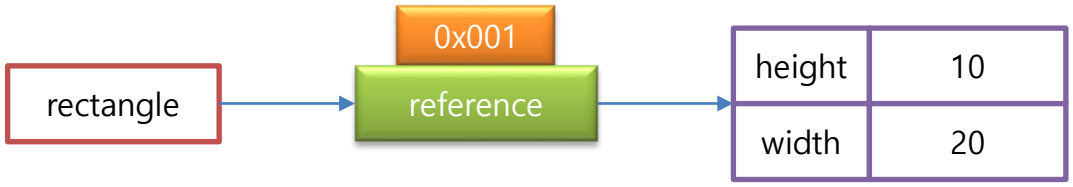
● 생성자

- 객체 생성 시 속성 생성 또는 값을 할당 (초기화)

05-02-생성자.html

```
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

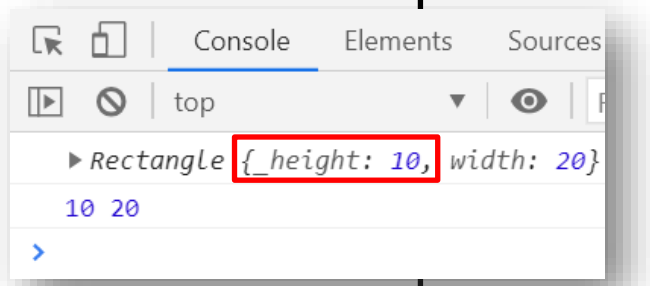
● Getter / Setter

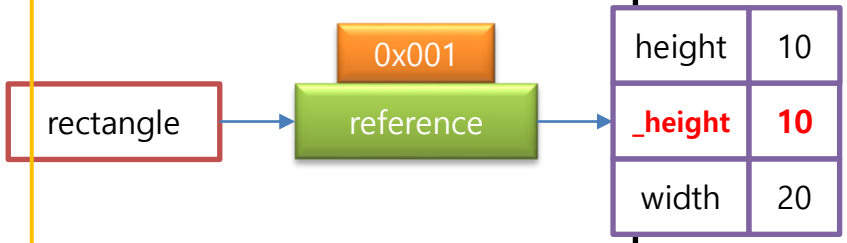
- 속성에 값이 입력되거나 조회될 때 동작하는 기능

05-03-getter-setter.html

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
    this.height = 1; ➔ set height(1)

  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);
```

_height 변수명은 get과 set 동일하게 작성





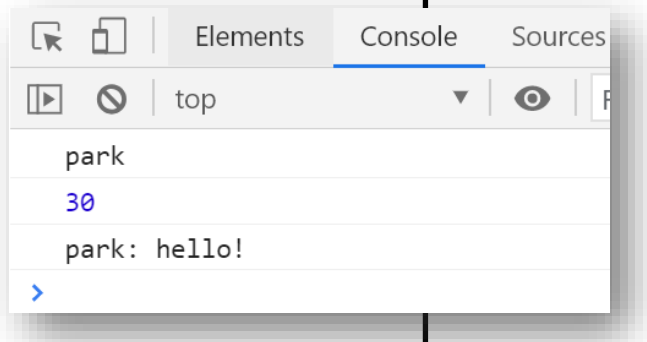
■ Class

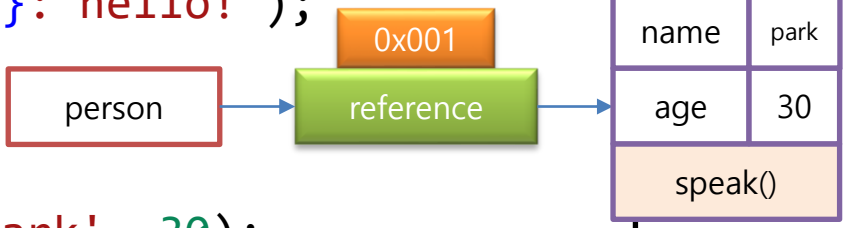
● Method

- 클래스 내의 함수

05-04-method.html

```
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();
```

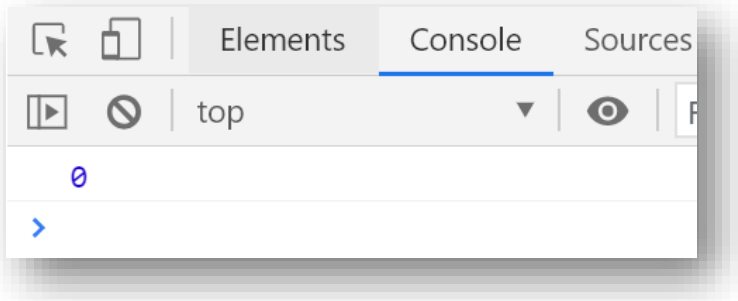




■ 연습문제 (05-연습문제1.html)

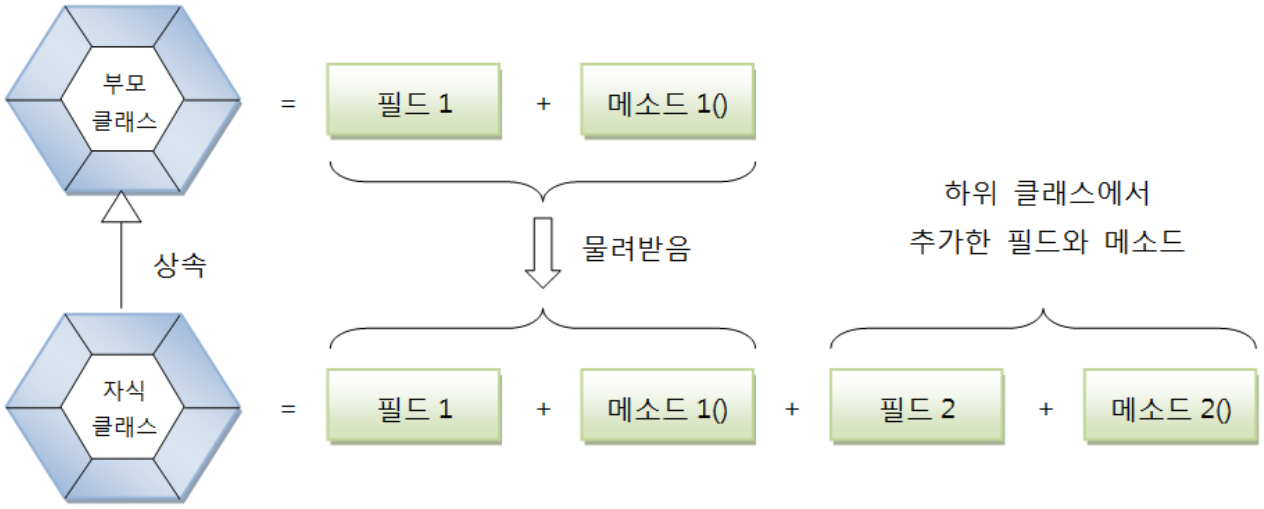
- 생성자와 Getter / Setter를 가지는 User 클래스 작성
  - 생성자 : firstName, lastName, age
  - Getter / Setter : age에 음수가 입력되면 0으로 저장

```
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)

05-05-상속.html

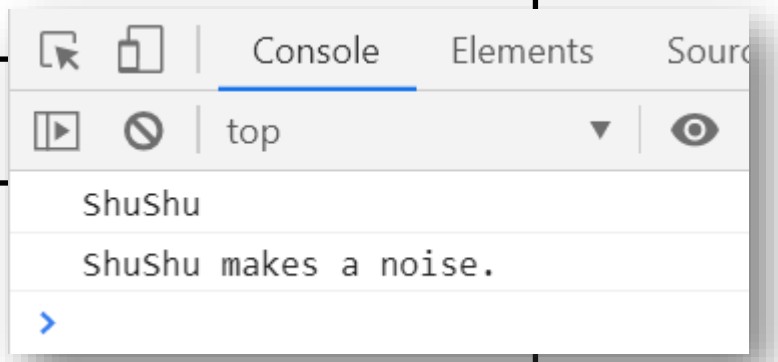
```
class Animal {
  constructor(name) {
    this.name = name;
    console.log();
  }
  speak() {
    console.log(`${this.name} makes a noise.`);
  }
}
```

● 자식 클래스 (sub class)

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

● 사용

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

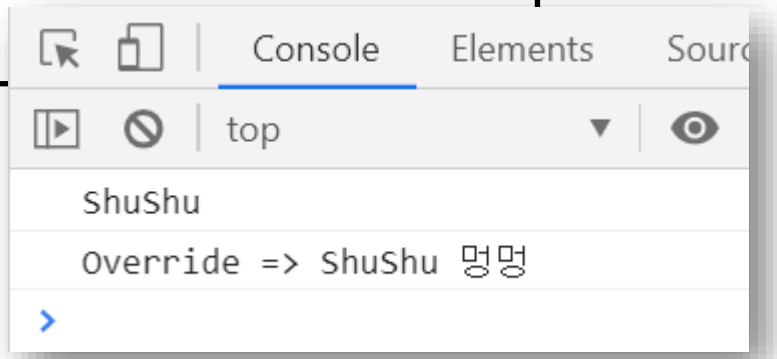


■ 상속

● Override

05-06-override.html

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

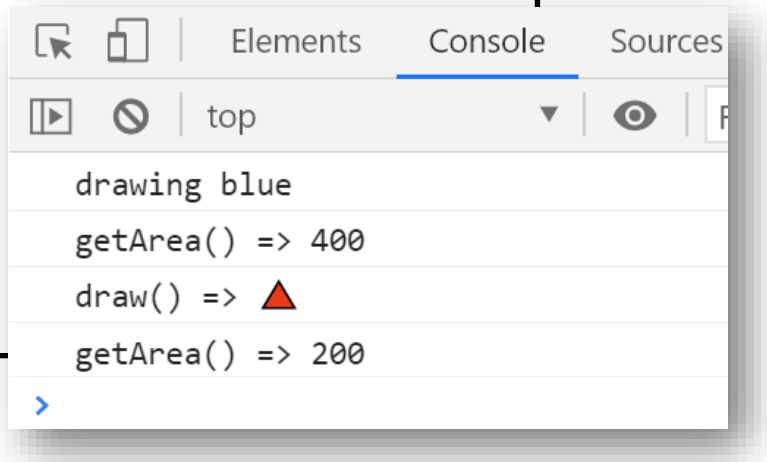


■ 연습문제 (05-연습문제2.html)

● 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(`draw`);
  }

  getArea() {
    console.log(`getArea`)
  }
}
```

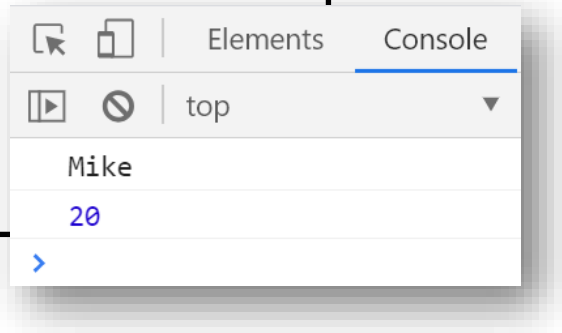


■ Object

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

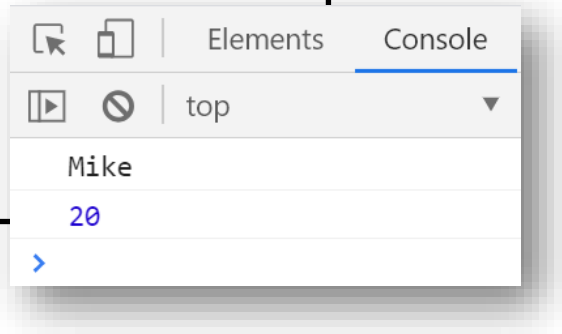
05-07-object.html

```
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);
```

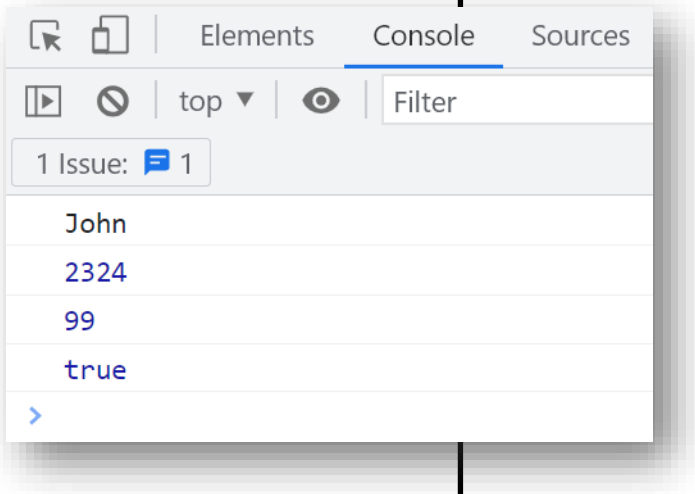


■ Object

● 변수 제어의 편리함

05-08-object.html

```
const playerName = 'John';
const playerPoint = 2324;
const playerLevel = 99;
const playerWeapon = true;

const player = {
  'name': 'John',
  'point': 2324,
  'level': 99,
  'weapon': true
}
console.log(player.name);
console.log(player.point);
console.log(player.level);
console.log(player.weapon);
```



■ Object

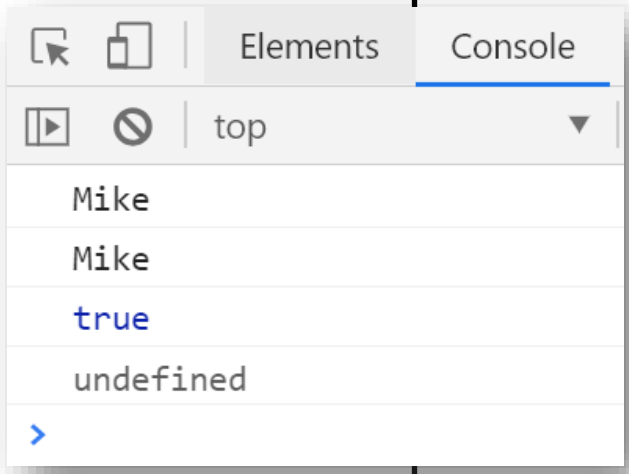
● 초기화

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

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

● 속성 제어

05-09-object-제어.html

```
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);
```

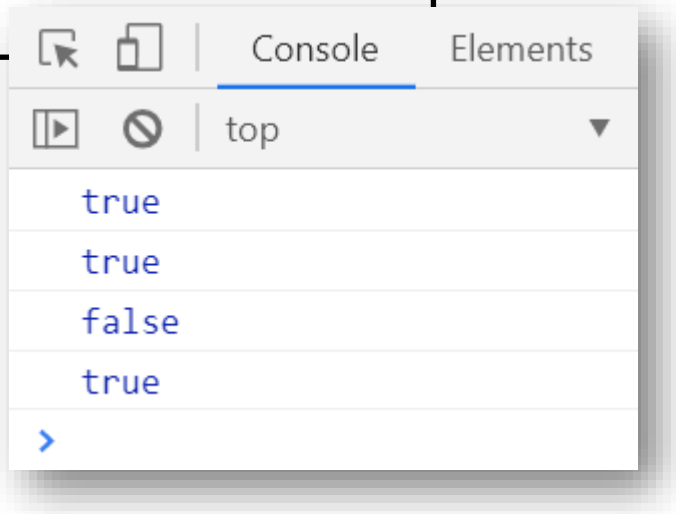


■ Object

● 속성 존재 확인

05-10-object-in.html

```
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

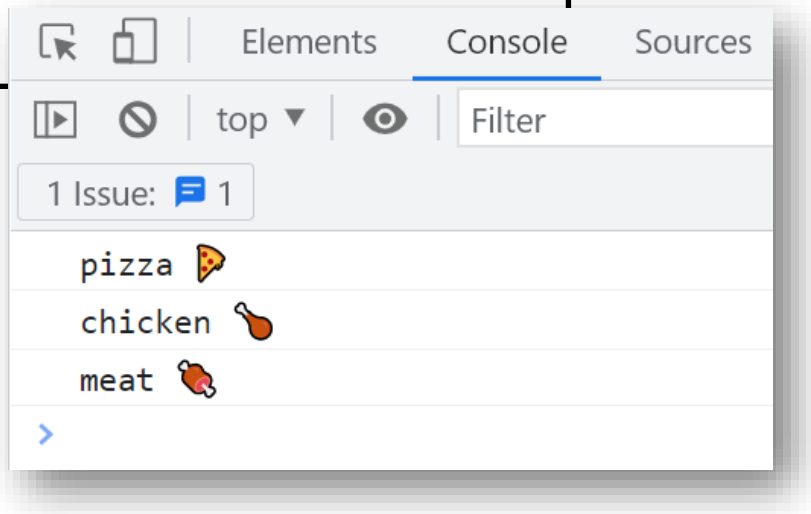
● 반복문을 이용한 속성 제어

05-11-object-loop.html

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

- key를 알아낸 후 value 확인 (for ... in)

```
for(const key in obj) {
  console.log(key, obj[key]);
}
```

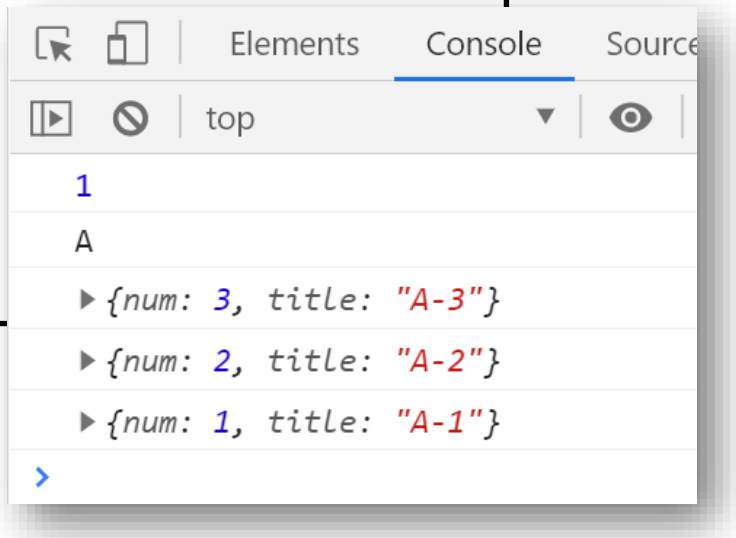


■ 연습문제 (05-연습문제3.html)

● 반복문을 사용하여 article 객체가 가지고 있는 모든 속성들의 값 출력하기

- for in / typeof 활용

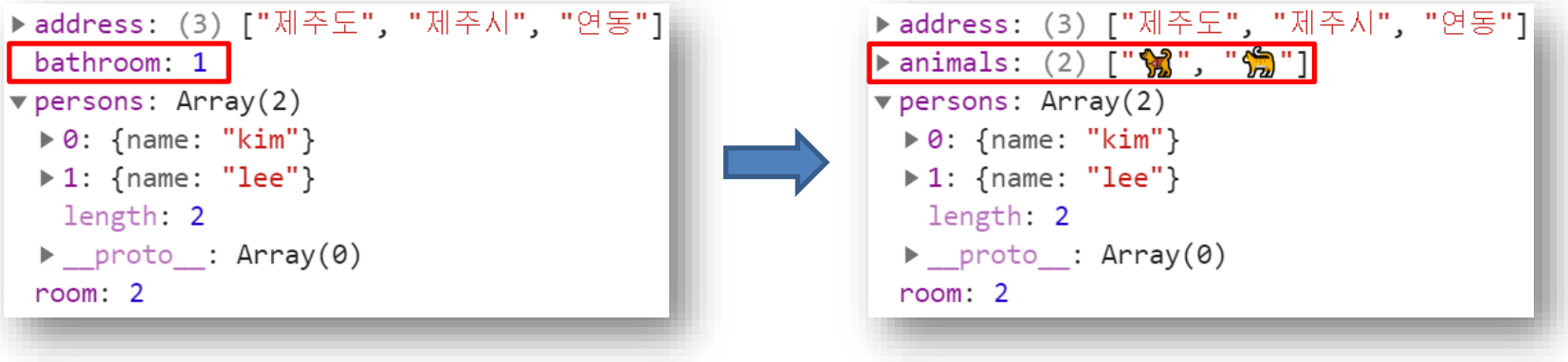
```
const article = {
  num: 1,
  title: 'A',
  reply: [
    { num: 3, title: 'A-3' },
    { num: 2, title: 'A-2' },
    { num: 1, title: 'A-1' }
  ]
};
```



■ 연습문제 (05-연습문제4.html)

● 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

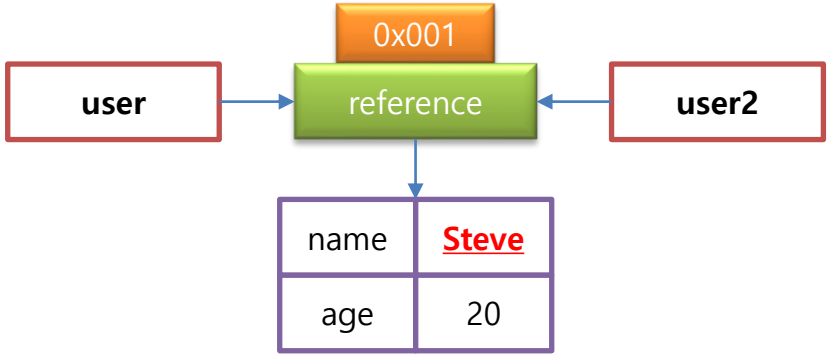
● object 복사

- 대입 연산자 사용시 문제점

05-12-object-copy.html

```
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



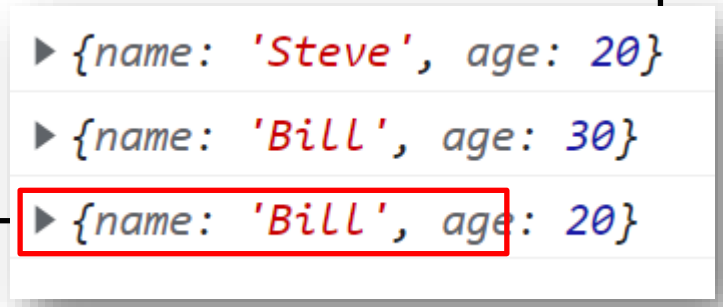
■ Object

● object 복사

- assign()

05-13-object-copy.html

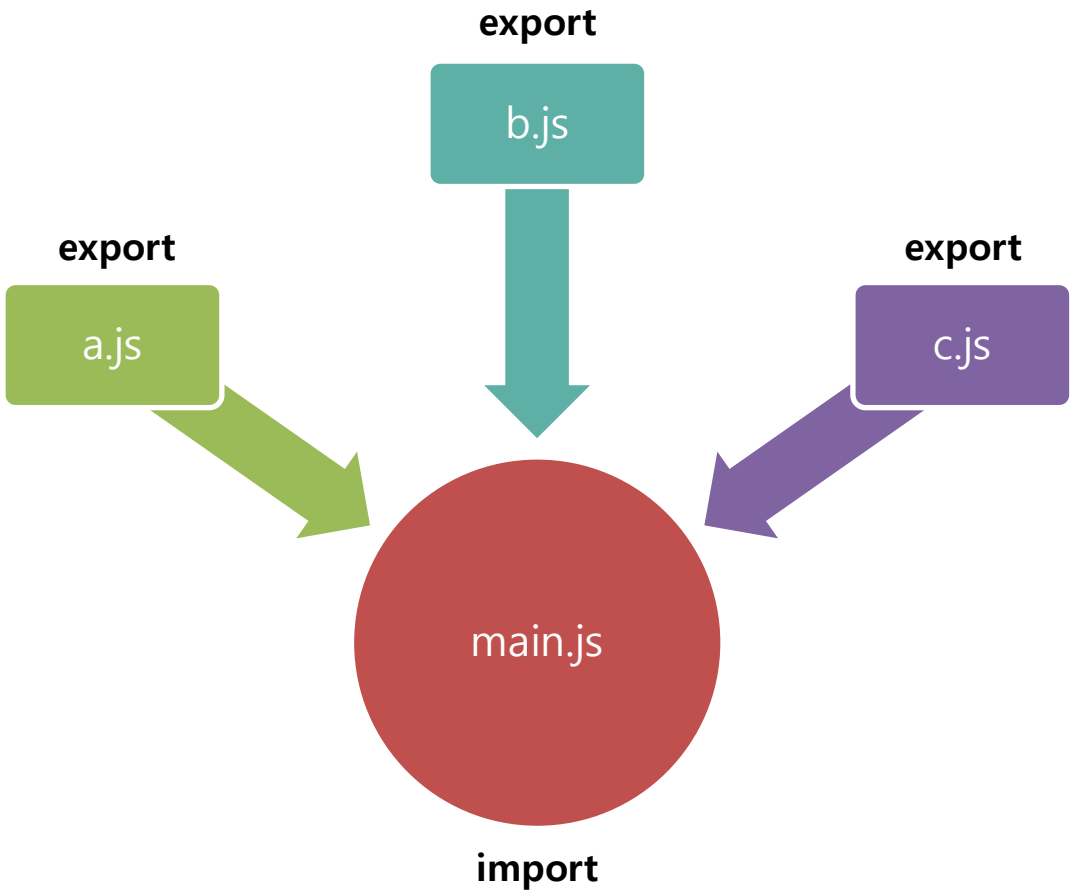
```
const user = { name: 'Bill', age: 20 };

const user2 = {};
for (const key in user) {
  user2[key] = user[key];
}
user2.name = 'Steve';
console.log(user2);

const user3 = {};
Object.assign(user3, user);
user3.age = 30;
console.log(user3);

console.log(user);
```



■ export / import

● 모듈 내보내기 / 가져오기

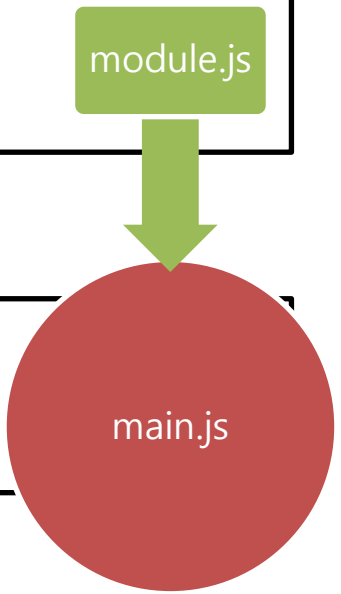
- 함수, 클래스, 변수(상수) 등



■ export / import

● export (05-14-module.js)

```
export function plus() {
  return 2 + 1;
}
```

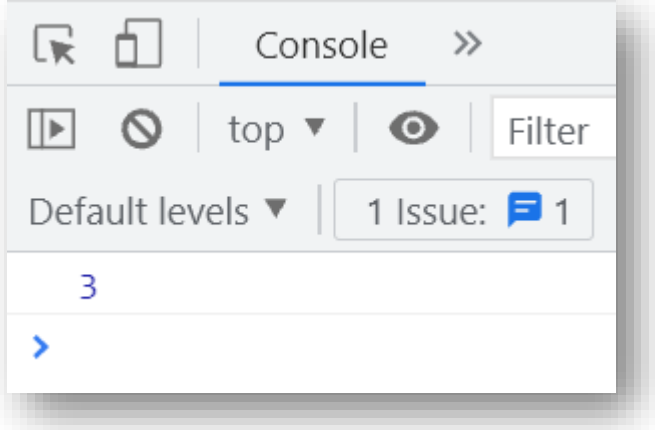


● import (05-14-main.js)

```
import { plus } from './05-14-module.js';

console.log(plus());
```

● 05-14-export-import.html

```
<script type="module" src="./05-14-main.js"></script>
```



■ export

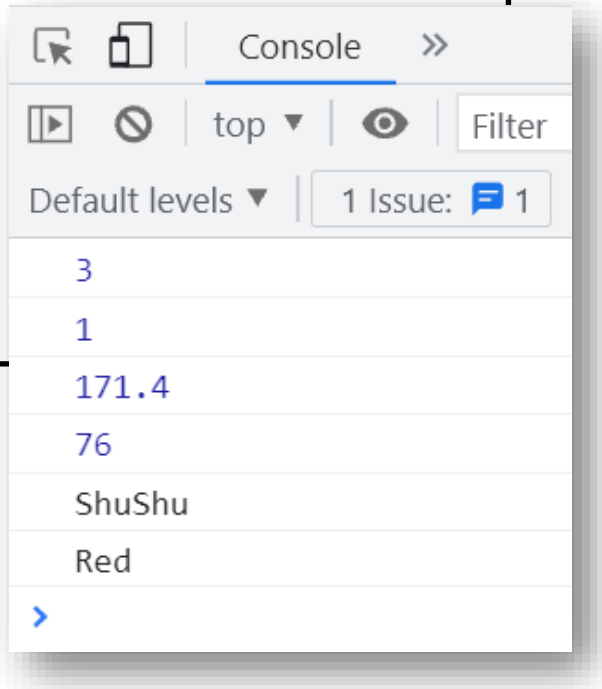
● 05-15-module.js

```
export function plus() { // 함수 export
  return 2 + 1;
}

export function minus() { // 함수 export
  return 2 - 1;
}

export const height = 171.4; // 상수 export

export let weight = 74.3; // 변수 export
weight = 76; // 값 변경

export class Animal { // 클래스 export
  constructor(name) {
    this.name = name;
  }
}
```

■ 개별 import

● 05-15-main.js

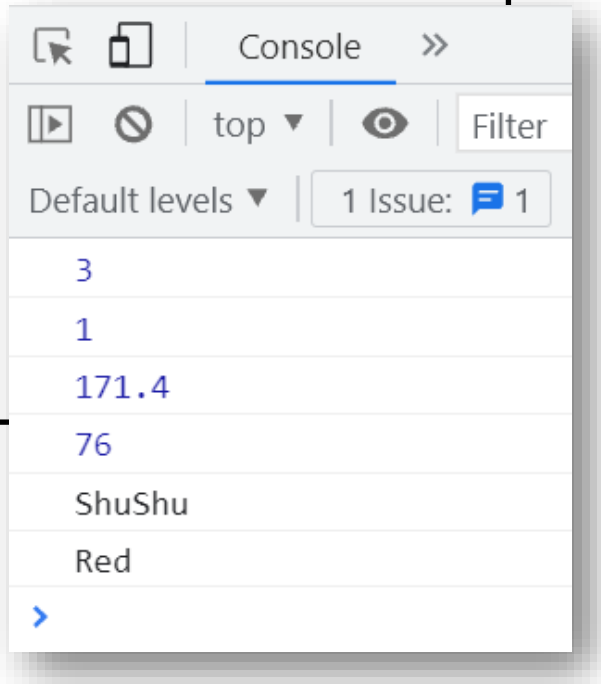
```
// import { export이름1, export이름2 } from 'OOO.js';
import { plus, minus, height, weight, Animal, color }
  from './05-15-module.js';

console.log(plus());
console.log(minus());

console.log(height);
console.log(weight);

const animal = new Animal('ShuShu');
console.log(animal.name);

console.log(color);
```



■ 전체 import

● 05-15-main.js

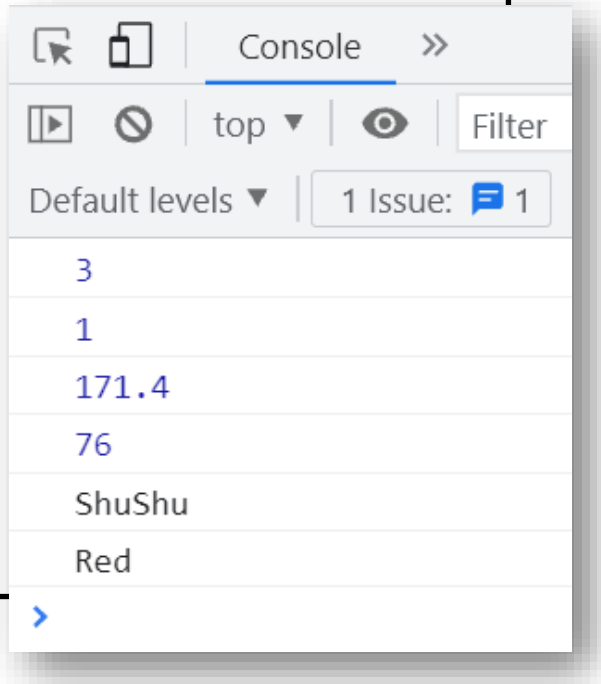
```
// import * from 'OOO.js';
import * as module from './05-15-module.js';

console.log(module.plus());
console.log(module.minus());

console.log(module.height);
console.log(module.weight);

const animal =
  new module.Animal('ShuShu');

console.log(animal.name);

console.log(module.color);
```



■ 별칭 사용 import

● 05-15-main.js

```
// import { export이름1 as 별칭1, export이름2 as 별칭2 }
// from 'OOO.js';

import {
  plus as p, minus as m, height as h, weight as w,
  Animal as A, color as c
} from './05-15-module.js';

console.log(p());
console.log(m());

console.log(h);
console.log(w);

const animal = new A('ShuShu');
console.log(animal.name);

console.log(c);
```



■ default export / import

- 05-16-module.js

```
export default function() {
  return 'Default!'
}
```

- 05-16-main.js

```
import func from './05-16-module.js'
console.log(func());

import { default as func2 } from './05-16-module.js'
console.log(func2());
```