■ 객체

● 객체 생성

- 기본 형식

```
<script>
    var product = {
        name : 'Google 드론',
        kind : '드론',
        price : 100000
    };
</script>
```

- 표로 나타낸 객체 내부 모습

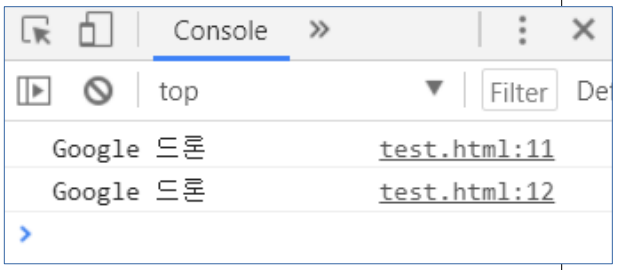
키	값
name	‘Google 드론’
kind	‘드론’
price	100000

■ 객체

● 객체 사용

- 대괄호 [] 또는 점 . 을 이용하여 내부 속성 접근 가능

```
<script>
    var product = {
        name : 'Google 드론',
        kind : '드론',
        price : 100000
    };

    var name1 = product.name;
    var name2 = product['name'];

    console.log(name1);
    console.log(name2);
</script>
```

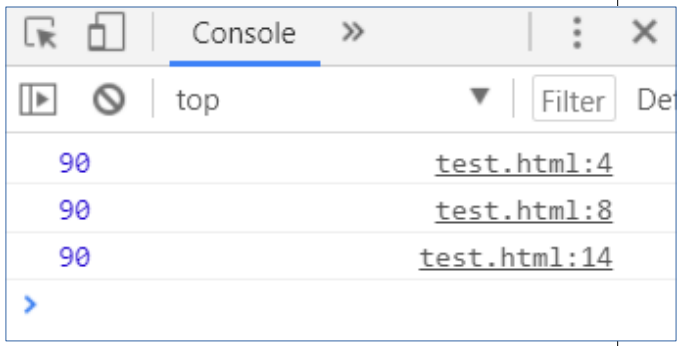


■ 객체

● 배열과 객체 비교

- 배열에서 숫자 인덱스를 사용하는 경우를 제외하고는 객체와 같음

```
<script>
    // 객체
    var scoreObj = { kor : 100, eng : 90 };
    console.log(scoreObj.eng);

    // 배열 - 키
    var scoreArr2 = [];
    scoreArr2['kor'] = 100;
    scoreArr2['eng'] = 90;
    console.log(scoreArr2.eng);

    // 배열 - 인덱스
    var scoreArr1 = [100, 90];
    console.log(scoreArr1[1]);
    console.log(scoreArr1.1); // 오류
</script>
```

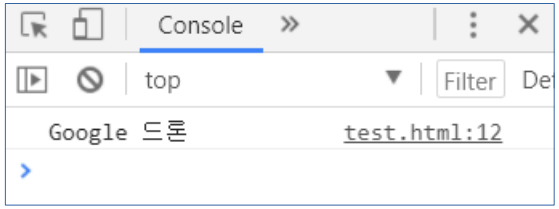


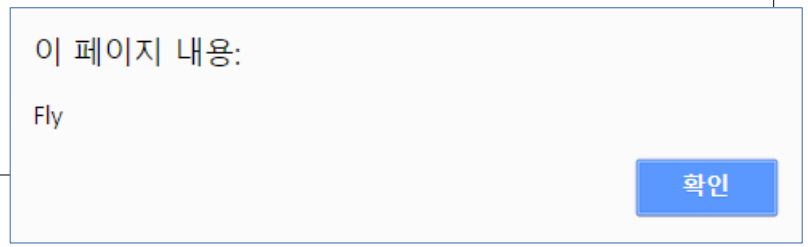
■ 객체

● 객체 사용

- 모든 자료는 객체의 내부 속성으로 지정 가능

```
<script>
    var product = {
        name : 'Google 드론',
        kind : '드론',
        price : 100000,
        play : function() {
            alert('Fly');
        }
    };

    var name = product.name;
    console.log(name);

    var play = product.play;
    play(); // 실행하려면 괄호를 붙여줌
</script>
```





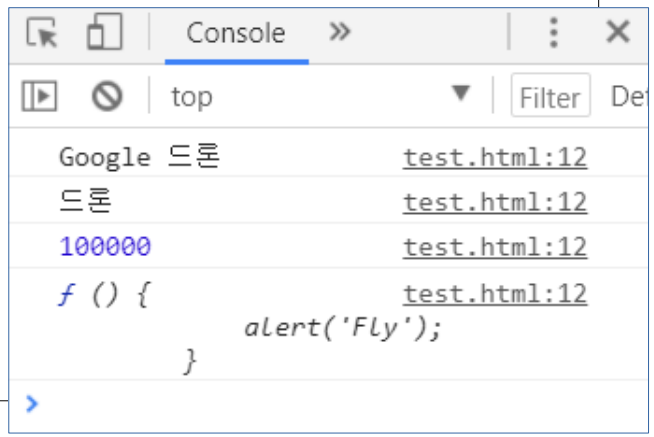
■ 객체

● 객체 사용

- 배열은 기본 for 반복문과 for-in 반복문 사용이 가능하지만

  객체는 for-in 반복문으로만 사용 가능

```
<script>
    var product = {
        name : 'Google 드론',
        kind : '드론',
        price : 100000,
        play : function() {
            alert('Fly');
        }
    };

    for(var key in product) {
        console.log(product[key]);
    }
</script>
```



■ 객체

● 객체 속성 추가

- 점 . 을 이용하여 속성 추가

```
<script>
    var person = {
        name : 'kim',
        age : 20
    };

    person.walk = function() {
        return this.name + this.address + ' 걷다';
    }

    person.address = '서울';

    document.write(person.walk());
</script>
```

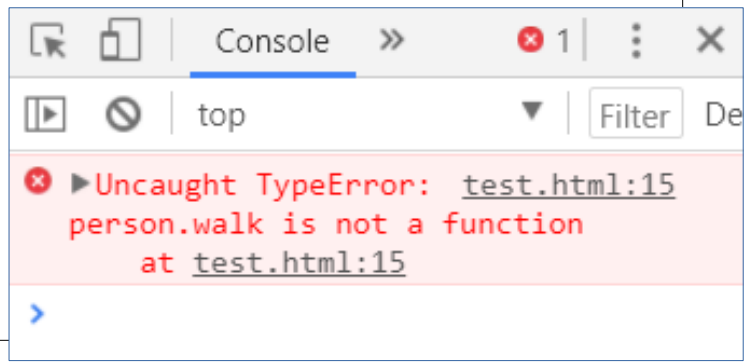
kim서울 걷다

■ 객체

● 객체 속성 삭제

- delete() 함수 사용

```
<script>
    var person = {
        name : 'kim',
        age : 20
    };

    person.walk = function() {
        return this.name + this.address + ' 걷다';
    }

    person.address = '서울';

    delete(person.walk);

    document.write(person.walk());

</script>
```

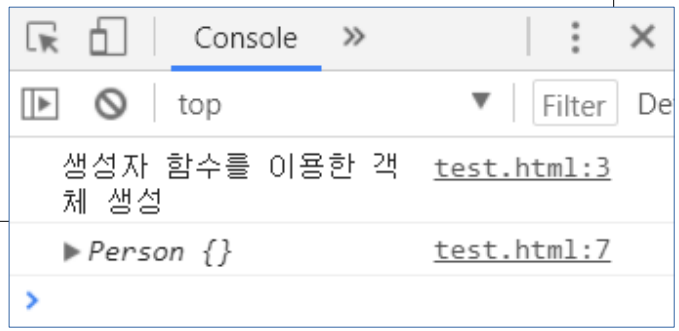


■ 생성자 함수

- 함수를 생성자의 형태로 사용 가능
- 프로토타입 적용 가능
- 캡슐화(은닉) 가능 (≒ 클로저)

- 기본 사용 형태

```
<script>
    function Person() {
        console.log('생성자 함수를 이용한 객체 생성');
    }

    var p = new Person();
    console.log(p);
</script>
```

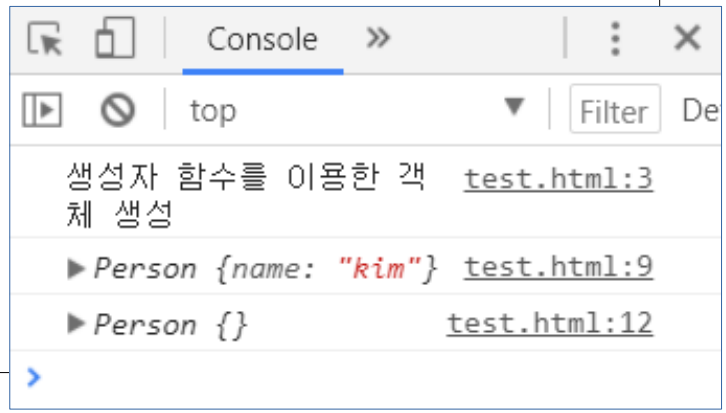


■ 생성자 함수

● 생성자 함수로 생성된 객체도 중괄호 {} 로 생성한 객체처럼 사용 가능

– 객체 내부 속성 추가 및 삭제

```
<script>
    function Person() {
        console.log('생성자 함수를 이용한 객체 생성');
    }

    var p = new Person();

    p.name = 'kim'; // 속성 추가
    console.log(p);

    delete(p.name); // 속성 삭제
    console.log(p);
</script>
```



■ 생성자 함수

● 객체에 특정 기능(함수)를 추가하려는 경우

1. 생성된 객체별 추가

```
<script>
    function Person() {  }

    var p1 = new Person();
    p1.sleep = function() {
        return 'zzz';
    };
    var p2 = new Person();
    p2.sleep = function() {
        return 'zzz';
    };

    console.log(p1.sleep());
    console.log(p2.sleep());
</script>
```

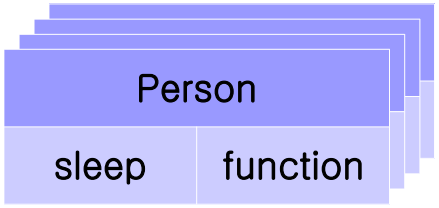
2. 생성자 함수에 추가

```
<script>

    function Person() {
        this.sleep = function() {
            return 'zzz';
        };
    }

    var p1 = new Person();
    var p2 = new Person();

    console.log(p1.sleep());
    console.log(p2.sleep());

</script>
```

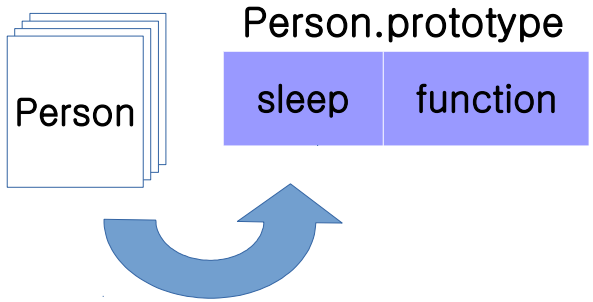
■ 생성자 함수

● 프로토타입

1. 프로토타입 적용 전



2. 프로토타입 적용 후



- 소스코드 수준의 객체 원형에 별도로 할당되는 공간
- 객체를 생성할 때마다 동일한 함수를 계속 생성하는 것은 메모리 낭비
- 개발자가 직접 만든 함수가 아닌 경우에도 기능 추가가 용이
- 객체에는 속성만 존재하도록 하고 함수는 프로토타입을 이용해서 추가

■ 생성자 함수

● 객체에 특정 기능(함수)를 추가하려는 경우

1. 생성자 함수에 추가

```
<script>

    function Person() {
        this.sleep = function() {
            return 'zzz';
        };
    }

    var p1 = new Person();
    var p2 = new Person();

    console.log(p1.sleep());
    console.log(p2.sleep());

</script>
```

2. 프로토타입을 이용하여 함수 추가

```
<script>

    function Person() {  }

    Person.prototype.sleep = function() {
        return 'zzz';
    };

    var p1 = new Person();
    var p2 = new Person();

    console.log(p1.sleep());
    console.log(p2.sleep());

</script>
```

■ 프로토타입

● Date 함수에 nowTime() 기능 추가

– 출력 형식 → hh:mm:ss

```
<script>
    var date = new Date();

    Date.prototype.nowTime = function() {
        return this.getHours() + ":" + this.getMinutes() + ":" + this.getSeconds();
    };

    console.log(date.toString());
    console.log(date.nowTime());
</script>
```