자바스크립트

객체

목차

- 객체형
- Object literal
- 기본형을 객체형으로 변환

객체형

- 객체형(Object type)의 개념
  - Primitive type을 제외한 데이터타입의 베이스 인스턴스
  - JavaScript의 거의 모든 객체는 Object의 파생 인스턴스

- 객체형의 구성
  - Object는 {}로 생성.
  - key-value 형식으로 데이터 유지. (key : value)

객체형

- 객체형의 선언 및 연산
  - 객체 내부에는 Primitive type뿐만 아니라 객체형도 들어 갈 수 있다.

```
const product =
{
    name : 'mouse',
    type : 'wireless',
    color : 'black',
    buttons : 6,
    scroll : true
}

console.log(typeof(product));
console.log(product.name);
console.log(product['type']);
console.log(product.buttons);
console.log(product.scroll);
```

'학생' 을 추상화 하여 object를 만드시오
object 내부의 값을 출력하시오

객체형

- 객체 속성(property)의 변경
  - 속성 변경

```
product.buttons = 8;
console.log(product.buttons);
```

  - 속성 추가
    - 속성 변경과 같은 방식이며 속성이 없다면 추가가 됨
  - 속성의 삭제
    - delete 객체.속성

학생객체의 속성을 변경하고 console에 출력하는 코드를 작성하시오
국어, 영어, 수학 성적이 저장되어 있는 배열을 속성으로 추가 하시오

객체형

- 메소드(Method)
  - 객체에 포함되는 함수
  - this : 객체 자신을 나타내는 레퍼런스 값
  - 화살표 함수의 경우 this를 사용할 수 없음
    - this가 해당 Object의 레퍼런스 값이 아님.
  - 속성의 형식 출력 : JSON.stringify()

product에서 color를 리턴하는 화살표함수를 추가하시오
'학생' 성적의 평균을 리턴하는 메소드를 추가하시오

```
<script>
    const product =
    {
        name : 'mouse',
        type : 'wireless',
        color : 'black',
        buttons : 6,
        scroll : true,
        getName : function() {
            return this.name;
        }

    }

    console.log(product.getName());
</script>
```

연습문제.

- 다음을 자바스크립트 객체로 만드시오(타입, 이름등은 알맞게 설정)

속성명	값
name	Macbook Pro
type	notebook
USB	3
price	$1,300

- name을 출력하는 메소드를 추가하시오
- 10개 이상 구입시엔 10%를 할인해 준다고 한다. 구매가격을 계산하는 메소드를 추가하시오
- 위 노트북은 다양한 언어를 지원한다 지원하는 언어를 속성으로 추가하시오

Object literal

- Object literal
  - Javascript에서는 객체를 만드는 여러가지 방법이 존재
  - const obj = {};와 같은 객체를 ES6에서 Object literal이라 함

- ES6에서 추가된 Object literal기능
  - 속성(property) 축약 표현
    - 속성값으로 변수를 사용하는 경우 변수이름과 속성이름이 같다면 키를 생략

```
const x = 1;
const y = 2;

const obj = {
    x : x,
    y : y
}
```


```
const x = 1;
const y = 2;

const obj = {
    x,
    y
}
```

Object literal

- ES6에서 추가된 Object literal기능
  - 메서드 축약 표현
    - 메서드의 이름과 키를 동일한 이름으로 사용할 경우 메서드를 축약하여 표현

```
var obj = {
    printHello : function()
    {
        console.log('Hello!');

    }
};

obj.printHello();
```


```
const obj = {
    printHello()
    {
        console.log('Hello!');

    }
};

obj.printHello();
```

Object literal

- ES6에서 추가된 Object literal기능
  - 계산된 속성이름(Computed property name)
    - 표현식을 사용해 Key를 동적으로 생성
    - ES5에서는 Object literal 외부에서 구현
    - ES6에서는 Object literal 내부에서도 구현 가능.

```
var prefix = 'val';
var i = 1;
var obj = {};

obj[prefix + '-' + i] = i++;
obj[prefix + '-' + i] = i++;
obj[prefix + '-' + i] = i++;

console.log(obj);
```


```
const prefix = 'val';
let i = 1;
const obj = {
    [`${prefix}-${i}`] : i++,
    [`${prefix}-${i}`] : i++,
    [`${prefix}-${i}`] : i++
};

console.log(obj);
```

Object literal

- Object의 순회
  - Object의 속성의 순서는 무순서
  - 배열처럼 순차적인 인덱스가 없음.
  - 순회하기위해서 for~in 문을 사용

```
const obj = {
    a : 1,
    b : 2,
    c : 3
};

for (const key in obj)
{
    console.log(key + ': ' + obj[key]);

}
```

기본형을 객체형으로 변환

- Primitive type을 객체 자료형으로 변환
  - 형식

```
<script>
    const f = new Number(270);
    console.log(typeof(f));
    f.test_value = 10;

    console.log(f.valueOf());
</script>
```

```
object
270
```

- new는 객체를 생성하는 키워드.
- 만약 실수로 new를 사용하지 않으면 형변환 코드가 됨

기본형을 객체형으로 변환

- 기본 자료형의 일시적 객체형 변환
  - 기본자료형은 메소드나 속성을 가질 수 없음
  - 하지만 일시적으로 이를 가질 수 있게끔 기본형을 객체형으로 변환
  - 이는 묵시적으로 변환하여 개발자에게 편의성을 제공

```
const str = '안녕하세요';
let strlen = str.length;
console.log(strlen);
```


```
const str = new String('안녕하세요');
let strlen = str.length;
console.log(strlen);
```