객체

▶ 객체 선언 및 호출

객체는 키 값을 사용하여 속성(멤버 변수) 식별

중괄호를 사용하여 객체를 생성하고 [ ] 또는 .으로 요소 값에 접근

속성에 모든 자료형이 올 수 있으며 **그 중 함수 자료형인 요소 : 메소드**

객체 내에서 자신의 속성을 호출할 때 반드시 this키워드를 사용하고

객체의 모든 속성을 출력하려면 단순 for문이나 while문으로는 출력이 불가하니 for in문을 사용해야 함

* 식별자로 사용할 수 없는 문자(띄어쓰기, 특수문자)를 속성으로 사용할 경우 ' '로 묶어서 선언하고 접근 시에는 [ ]만 가능

▶ 객체 선언 및 호출

✓ 선언 방법

```
var 변수 명(객체 명) = {
        속성(키 값) : 값,
        속성(키 값) : 값,
        속성(키 값) : 값
};
```

✓ 속성 값 접근

```
변수 명(객체 명)['요소 명(키 값)'];
또는
변수 명(객체 명).요소 명(키 값);
```

▶ in/with 키워드

in : 객체 내부에 해당 속성이 있는지 확인하는 키워드

with : 코드를 줄여주는 키워드, 호출 시 객체 명 생략 가능

in 키워드

속성 명 in 변수 명(객체 명) // 있으면 true, 없으면 false

with 키워드

```
with(변수 명(객체 명)){
    속성 명;
}
```

* with 미사용 시 : 변수 명(객체 명).속성 명;

▶ 객체 속성 추가 및 삭제

속성 및 메소드를 동적으로 추가/삭제 가능

추가

```
변수 명(객체 명).속성 명 = '값';
변수 명(객체 명).속성 명 = "값";
변수 명(객체 명).속성 명 = function(){
        메소드 로직;
        [return [리턴 값];]
};
```

삭제

```
delete(변수 명(객체 명).속성 명);
```

▶ 객체 배열 활용

생성된 객체를 배열에 넣어 활용 가능

var 변수 명 = [ ]; // 배열 생성

데이터 대입 : 변수 명.push({속성 명: “값”, 속성 명: “값”, ...});

▶ 함수 활용 객체 생성

함수의 매개변수에 필요한 속성 값을 다 받아서 객체 생성 후 리턴

```
function 함수 명(value1, value2, value3, ...){
        var 변수 명(객체 명) = {
                속성 명: 'value1';
                속성 명: "value2";
                ...
        }
        return 변수 명(객체 명);
}
```

▶ 생성자 함수

this키워드를 사용하여 속성을 생성하는 함수로 new라는 키워드를 통해 객체 생성, 생성자 명의 첫 글자는 대문자로 시작하고 instanceof로 어떤 생성자가 생성 됐는지 확인 가능

```
function 생성자 명(value1, value2, value3, ...){
        this.속성 명 = 'value1',
        this.속성 명 = 'value2',
        ...
}
```

▶ 함수 활용 객체 생성과 생성자

함수 활용한 객체 생성과 생성자의 차이는 중복 메소드 저장 방식에 있음

- 함수 활용 : 중복되는 메소드를 객체 별로 만들어 저장
- 생성자 : prototype이라는 내부 객체를 이용하여 저장 가능

* 하나의 메소드를 이용해서 전체 객체가 다 활용(중복 저장X)

prototype 메소드 생성

변수 명(객체 명).prototype.메소드 명 = function(){};

▶ 캡슐화

생성자 함수에서 속성 선언 시 this키워드 사용하지 않고 지역변수로 선언

this키워드 사용하여 setter/getter 메소드 작성

* 클로저 활용(지역 변수를 지우지 않고 사용하는 기능)

```
function 생성자 명(value1, value2, value3, ...){
        var 속성 명 = 'value1';
        var 속성 명 = 'value2';
        this.set속성 명 = function(){};
        this.get속성 명 = function(){};
}
```

▶ 상속

다른 객체를 상속 받아 사용 가능

속성으로 객체를 추가하는 방법과 call 메소드를 이용하는 방법이 있음

```
function 생성자 명(value1, value2, value3, ...){
        this.속성 명 = 상속 받을 객체 명;
        this.속성 명(value1, value2); // 생성자 호출
        상속 받을 객체 명.call(this, value1, value2); // call메소드 이용
}
```