

객체

객체 선언 / 호출

- 객체는 키값을 사용하여 속성(멤버변수) 식별
- 중괄호를 사용하여 객체생성
- '[]'(대괄호) 또는 ' . '(점)으로 요소의 값에 접근
- 속성에 모든 자료형이 올 수 있음, 그 중 함수 자료형인 요소 : 메소드
- 객체 내에서 자신의 속성을 호출할때 반드시 this키워드를 사용
- 객체의 모든 속성을 출력하려면 for in문을 사용해야 함.
 - ☞ 단순 for문이나 while문으로 출력 불가.

☞ 식별자로 사용할 수 없는 문자(띄어쓰기, 특수문자)를 속성 으로 사용할 경우 ' '로 묶어서 선언하고, 접근시에는 []만 가능

선언하는 방법

```
var 변수명(객체명) = {  
    속성(키값) : 값 ,  
    속성(키값) : 값 ,  
    속성(키값) : 값  
};
```

속성값 접근

```
변수명(객체명)['요소명(키값)'];  
변수명(객체명)['요소명(키값)'];  
또는  
변수명(객체명).요소명(키값);  
변수명(객체명).요소명(키값);
```

in / with 키워드

- in : 객체 내부에 해당 속성이 있는지 확인하는 키워드
- with : 코드를 줄여주는 키워드, 호출시 객체명 생략 가능

in 키워드

속성명 in 변수명(객체명) // 있으면 true, 없으면 false

with 키워드

with(변수명(객체명)){ 속성명; 속성명; }	with 미사용시 변수명(객체명).속성명; 변수명(객체명).속성명;
--------------------------------------	---

객체 속성 추가 및 삭제

- 속성 및 매소드를 동적으로 추가 및 삭제 가능

추가

```
변수명(객체명).속성명='값';  
변수명(객체명).속성명='값';  
변수명(객체명).속성명=function (){  
    메소드 로직;  
    [return [리턴값]];  
};
```

삭제

```
delete(변수명(객체명).속성명);
```

객체배열 활용

- 생성된 객체를 배열에 넣어 활용 가능

var 변수명 = [];//배열 생성

데이터 대입

변수명.push({ 속성명:'값',속성명:'값',속성명:'값' ... });

변수명.push({ 속성명:'값',속성명:'값',속성명:'값' ... });

변수명.push({ 속성명:'값',속성명:'값',속성명:'값' ... });

함수활용 객체생성

- 함수의 매개변수에 필요한 속성값을 다 받아서 객체를 생성 후 리턴

함수활용 객체생성

```
function 함수명(value1, value2, value3, ...){  
    var 변수명(객체명) = {  
        속성명:'value1',  
        속성명:'value2',  
        ....  
    }  
    return 변수명(객체명);  
}
```

생성자 함수

- this키워드를 사용하여 속성을 생성하는 함수
- new라는 키워드를 사용하여 객체 생성
- 생성자명의 첫 글자는 대문자로 시작
- instanceof로 어떤 생성자로 생성된지 확인가능

```
function 생성자명(value1, value2, value3, ....){  
    this.속성명='value1',  
    this.속성명='value2',  
    ....  
}  
}
```


함수활용 객체생성 VS 생성자

- 중복 매소드를 저장하는 방식차이
 - 함수활용 : 중복되는 메소드를 객체별로 만들어서 저장
 - 생성자 : prototype이라는 내부 객체를 이용하여 저장 가능
 - ☞ 하나의 메소드를 이용해서 전체 객체가 다 활용(중복저장 X)

prototype 메소드 생성

변수명(객체명).prototype.메소드명=function(){ };

캡슐화

- 생성자 함수에서 속성 선언시 this키워드를 사용하지 않고 지역변수로 선언
- this키워드 사용 setter/getter 메소드 작성
 - ☞ 클로저 활용(지역변수를 지우지 않고 사용하는 기능)

```
function 생성자명(value1, value2, value3, ....){  
    var 속성명='value1';  
    var 속성명='value2';  
    this.set속성명= function( ) { };  
    this.get속성명= function( ) { };  
}  
}
```

상속

- 다른 객체를 상속받아 사용 가능
- 속성으로 객체를 추가하는 방법 / call메소드 이용하는방법

```
function 생성자명(value1, value2, value3, ....){  
    this.속성명1=상속받을 객체명;  
    this.속성명1(value1, value2);//생성자 호출  
    //////////// call메소드이용/////////  
    상속받을 객체명.call(this,value1,value2);  
}  
}
```