JavaScript Functions and Objects:
(자바스트립트 함수와 객체)

JavaScript Functions

<실습1> checkGuess() 함수 기본

프로그램을 완성하여 보자

guess1.html

```
<script>
var guess =  "green";
var answer = checkGuess();
alert(answer);
  function checkGuess(){
    var answers=["red","green","blue"];
    var index = Math.floor(Math.random()*answers.length);
    if(guess == answers[index]){
        answer = "You're right! I was thinking of " +answers[index];
    } else{
        answer="sorry, I was thinking of "+ answers[index];
    }
    return answer;
  }

</script>
```

Functions 선언 및 분석 – 인수전달

```
function addScore(level, score) {
    var bonus = level * score * .1;
    return score + bonus;
}
```

Parameters와 arguments의 차이

매개변수로 함수를 선언
parameters

```
function cook(degrees, mode, duration) {
    // your code here
}
```

인자로 함수를 호출
arguments

```
cook(425.0, "bake", 45);
```

```
cook(350.0, "broil", 10);
```

순서 중요!!

매개변수는 단 한번만 정의 할 수 있지만 여러 번 인자를 넘겨받아 함수를 호출 할 수 있다.

함수 작동 방식 1

새로운 bark() 함수를 정의해 보자:

- 두 개의 파라메타를 가진다: dogName and dogWeight
- Dog의 몸무게에 따라 Dog의 짖는 소리를 리턴 한다

```
function bark(dogName, dogWeight) {
    if (dogWeight <= 10) {
      return dogName + " says Yip";
    } else {
      return dogName + " says Woof";
    }
}
```

함수 작동 방식 2

함수를 **호출**해 보자!

- 함수 이름(name)을 사용하고 요구되는 인자, 아규먼트(arguments) 제공

인자, 아규먼트(arguments)

```
bark("Fido", 50);
```


```
function bark(dogName, dogWeight) {
    if (dogWeight <= 10) {
      return dogName + " says Yip";
    } else {
      return dogName + " says Woof";
    }
}
```

함수 작동 방식 3

함수의 바디(body)를 동작시켜 보자!

- 바디의 문장(statements)은 탑다운(from top to bottom) 방식으로
- 함수파라메타 dogName과 dogWeight에 함수호출에 의해 전달된 아규먼트 할당

인자, 아규먼트(arguments) 전달

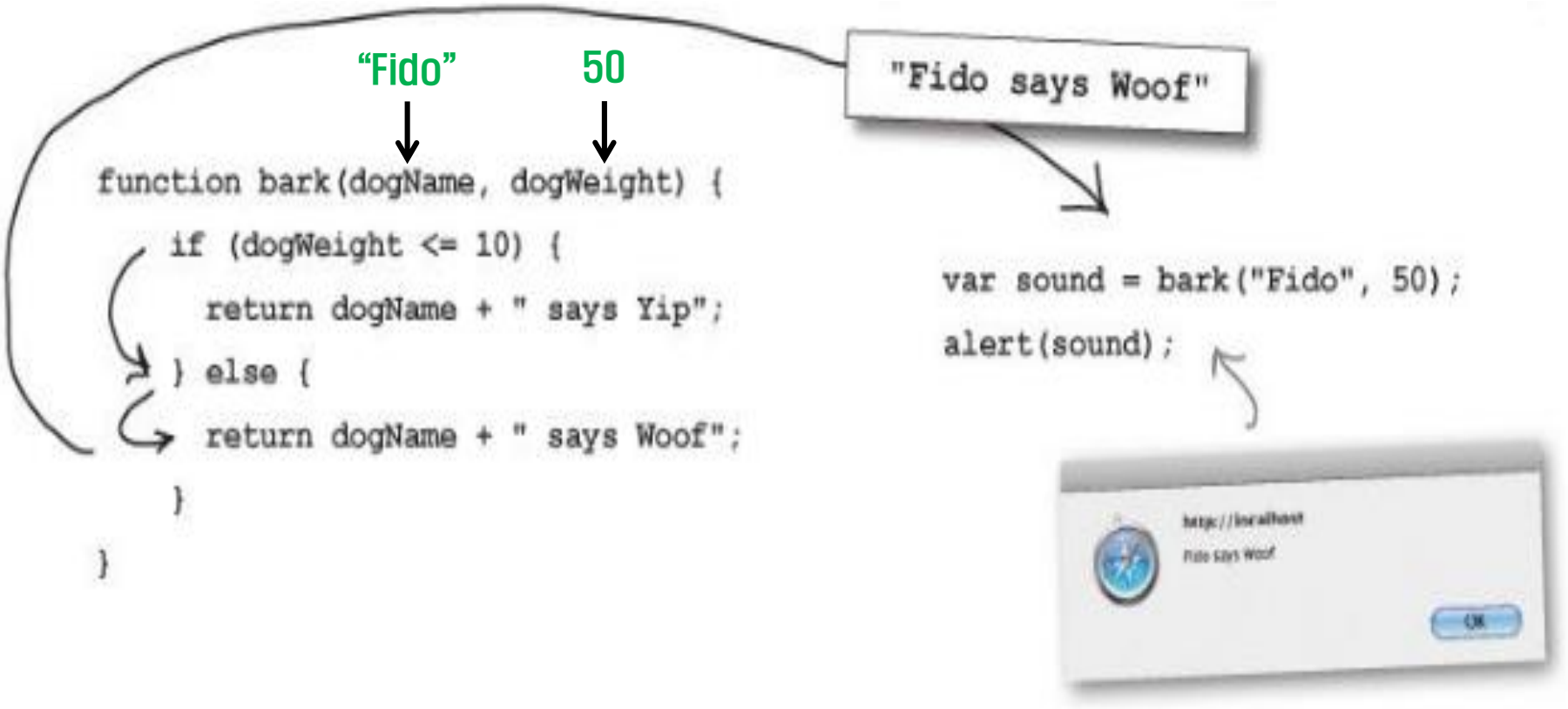
"Fido" 50

```
function bark(dogName, dogWeight) {
    if (dogWeight <= 10) {
      return dogName + " says Yip";
    } else {
      return dogName + " says Woof";
    }
}
```

함수 작동 방식 4

함수 바디에 return 문장을 가질 수 있다

- 호출을 수행한 코드에 값(value)을 리턴해 준다



지역변수(local)와 전역변수(global)

- var 키워드를 이용하여 변수(variable)를 선언할 수 있다
- 함수 내부(inside)에 변수를 선언할 수 있다:
  - ✓ 변수가 함수 바깥부분(outside)에 선언되면: GLOBAL 변수
  - ✓ 변수가 함수 내부(inside)에 선언되면: LOCAL 변수

변수의 짧은 인생 !

변수의 고단한 일생은 매우 짧을 수 있다. 즉 변수가 글로벌(global)이 아니라면, 심지어 글로벌 변수라 해도 일생이 제한을 갖는다.
무엇이 변수의 일생을 결정하는가? 다음처럼 한번 생각해 보자:

- 글로벌 변수는 브라우저에 페이지가 존재하는 한 살아있다
  - ✓ 글로벌 변수는 자바스크립트 코드가 페이지에 로드될 때 일생을 시작한다. 그러나 페이지가 사라지면 글로벌 변수의 일생도 종료된다
  - ✓ 같은 페이지가 다시 로드된다 해도 모든 글로벌 변수는 소멸되었기 때문에 새로 로드된 페이지에서 다시 생성된다
- 지역변수는 함수가 종료될 때 사라진다
  - ✓ 지역변수는 함수가 최초 호출될 때 생성되어 함수가 값을 리턴할 때까지 살아있다

지역변수와 전역변수의 이름이 같다면?- *You "shadow" your global*

- 전역변수 beanCounter와 아래와 같은 함수가 정의되었다고 하자:

```
var beanCounter = 10;

function getNumberOfItems(ordertype) {
    var beanCounter = 0;
    if (ordertype == "order") {
        // do some stuff with beanCounter
    }
    return beanCounter;
}
```

- 함수 안에서 beanCounter에 대한 모든 레퍼런스는 지역변수를 참조하는 것으로 간주된다

- 따라서 전역변수를 지역변수의 그림자 영역(shadow) 안에 있다고 말할 수 있다
- 지역변수의 그림자에 전역변수가 가려져 있어 전역변수를 볼 수 없습니다.
- 지역변수와 전역변수는 서로에게 아무 영향을 주지 않는다.

지역변수와 전역변수 의 스코프(Scope)

함수의 변수는 지역적인 스코프(*locally scoped*)를 갖는다

변수를 정의하는 위치가 변수의 Scope를 결정한다.

함수 바깥쪽에서 정의한 변수는 전역적인 스코프(*globally scoped*)를 갖는다:

```
var avatar = "generic";
var skill = 1.0;
var pointsPerLevel = 1000;
var userPoints = 2008;

function getAvatar(points) {
    var level = points / pointsPerLevel;

    if (level == 0) {
       return "Teddy bear";
    } else if (level == 1) {
       return "Cat";
    } else if (level >= 2) {
       return "Gorilla";
    }
}

function updatePoints(bonus, newPoints) {
    for (var i = 0; i < bonus; i++) {
        newPoints += skill * bonus;
    }
    return newPoints + userPoints;
}

userPoints = updatePoints(2, 100);
avatar = getAvatar(2112);
```

함수에 꼭 이름을 부여하지 않아도 된다

- 함수는 익명(anonymous)이 될 수 있다:
  - ✓ 왜 이런 함수를 필요로 할까?
  - ✓ 이름없는 함수(function without a name)를 정의해 보자:

```
function(num) {
    return num + 1;
}
```

이름없는 함수 생성

변수에 함수 할당

```
var f = function(num) {
    return num + 1;
}
var result = f(1);
alert(result);
```

그 변수를 사용하여 함수 호출

결과는?

변수에 함수를 할당 할 수 있다

- 변수를 사용하여 numbers, boolean values, strings, arrays 등을 저장(store) 할 수 있다
- 또한 변수에 함수를 할당(assign)할 수 있다!

```
function addOne(num){
return num+1;
}
//전달된 인자에 1을 더하는 간단한 함수를 정의

var plusOne=addOne;
//addOne()이라고 함수를 호출한 것이 아니라
//addOne 함수 이름만 사용하여   plusOne이란 변수에 함수 할당

var result =plusOne(1);
//plusOne에 함수가 할당됐으므로 1이라는 정수 인자와 함께 호출하
여 //return 받음
//결과는?
```

<실습2> 변수에 함수를 할당 예

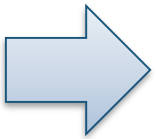
functionex.html

```
<!doctype html>
<head>
<title>function할당</title>
<meta charset="utf-8">
</head>
<body>
<script>
//그러나 변수에 함수를 할당(assign)할 수 있을까?

function addOne(num){
return num+1;
};
//전달된 인자에 1을 더하는 간단한 함수를 정의

var plusOne=addOne;
//addOne()이라고 함수를 호출한 것이 아니라
//addOne 함수 이름만 사용하여 plusOne이란 변수에 함수 할당

var result =plusOne(1);
alert(result);
//plusOne에 함수가 할당됐으므로 1이라는 정수 인자와 함께 호출하여
//return 받음
</script>
</body>
</html>
```

값의 일환인 함수로 무엇을 할 수 있을까요?

- 왜 이게 유용할까?
  - ✓ 여기서 중요한 점은 **변수에 함수를 할당**할 수 있다는 것이 아니라 **함수**가 실제로는 **값**(value)이라는 사실이다
  - ✓ 다양한 표현

```
function init() {
    alert("you rule!");
}
window.onload = init;
```

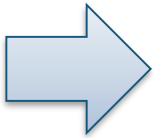


좀더 깔끔하게 표현하면

함수의 값을 직접 할당!!

```
window.onload = function() {
    alert("you rule!");
}
```

```
window.onload = outputResults;

function outputResults(){
    alert("you rule!");
}
```

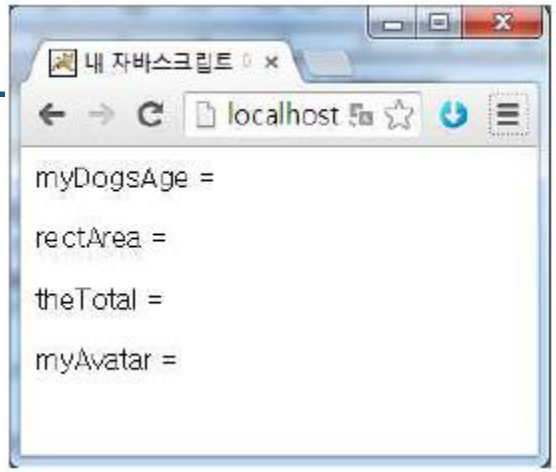


```
window.onload =function (){
    alert("you rule!");
}
```

<실습3> 함수들을 입력하여 결과를 확인 – 교재 참조

```
<!doctype html><html lang="en"><head>
<title>내 자바스크립트 예제</title>
<meta charset="utf-8">
</head>
<body>
<script>
  function dogsAge(age){
    return age*7;
  }
  var myDogsAge = dogsAge(4);
  function rectangleArea(width, height){
    var area = width * height
    return area;
  }
  var rectArea =rectangleArea(3,4);

  function addUp(numArray){
    var total =0;
    for(var i=0; i< numArray.length;i++){
    total += numArray[i];
    }
    return total;
  }
  var theTotal=addUp([1,3,5,9]);

  function getAvatar(points){
    var avatar;
    if(points <100){
        avatar = "쥐";
    }else if (points >100 && points < 1000){
        avatar="고양이";
    }else {
        avatar="원숭이";
    }
    return avatar;
  }
  var myAvatar = getAvatar(335);

  document.write("myDogsAge = " + myDogsAge + "<p>");
  document.write("rectArea = " + rectArea + "<p>");
  document.write("theTotal = " + theTotal + "<p>");
  document.write("myAvatar = " + myAvatar + "<p>");

  </script>
  </html>
```

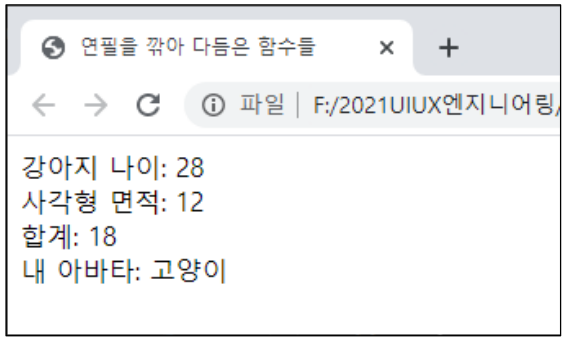
functionvalue.html



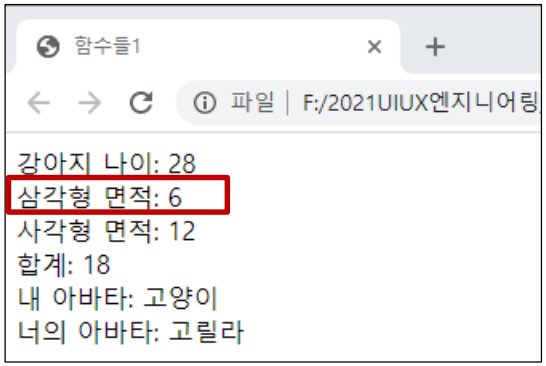
<실습4> 여러 functions 응용 – 교재 참조

- functionSharpen.html – 변수에 함수 할당
- functionSharpenup.html – 함수 추가(삼각형면적(triangleArea) , Avatar1(교재), 출력
- functionSharpen_upgrade.html – 전역변수 사용, 함수, 제어문 활용

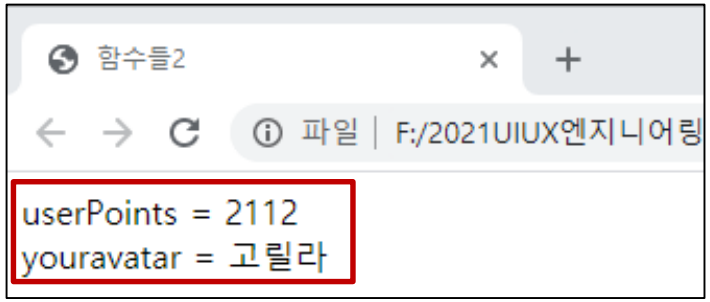
functionSharpen.html



functionSharpenup.html



functionSharpen_upgrade.html



<활용>

```
window.onload = outputResults;
function outputResults() {
  var output = document.getElementById("output");
```

```
<div id="output">
```

JavaScript Objects

“Objects” 가 무엇인가?

객체(Objects)는 자바스크립트 프로그래밍 기술을 한 수준 높여준다.

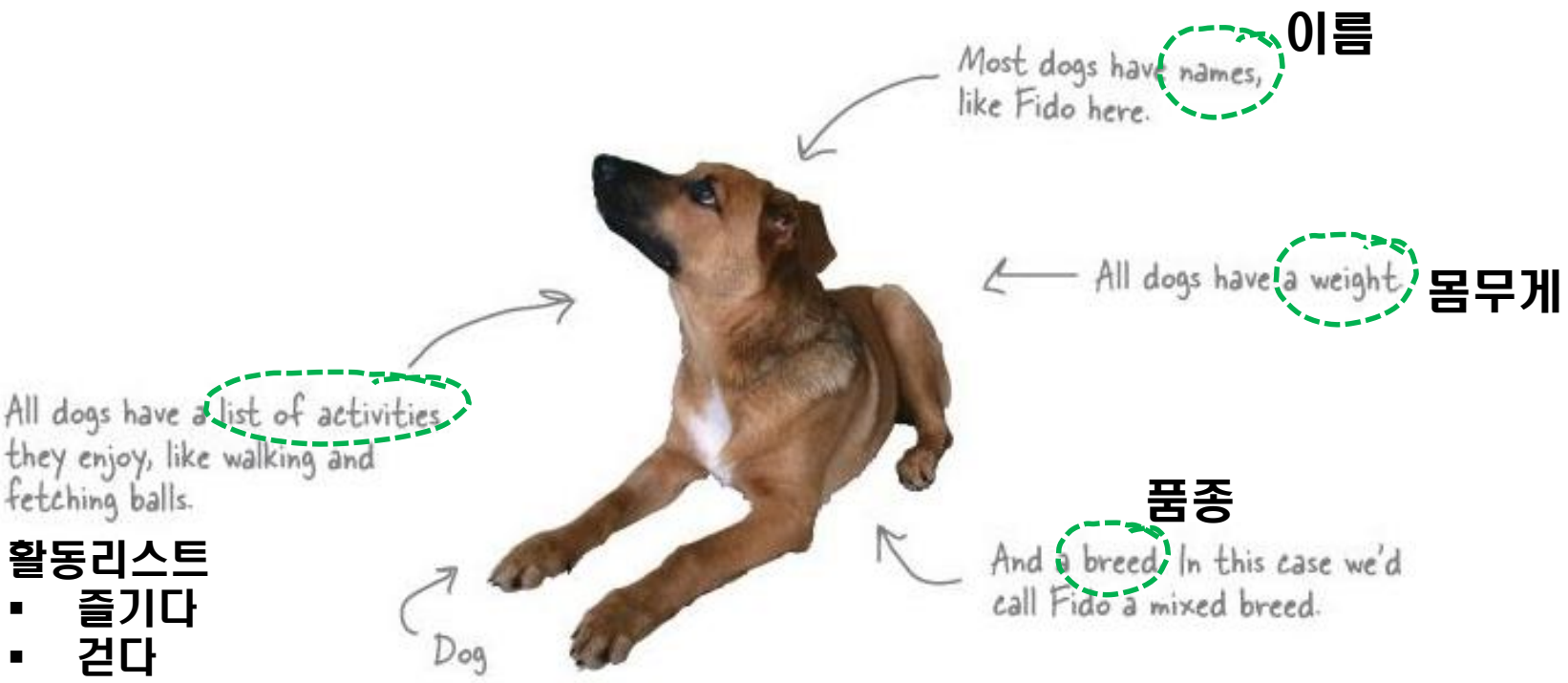
Objects are going to take your JavaScript programming skills to the next level

- to managing complex code,
- to understanding the DOM,
- to organizing your data,
- HTML5 JavaScript APIs 를 package하여 사용할 수 있는 기본 방법

Did someone say "Objects"?!

JavaScript objects : 속성의 결합체

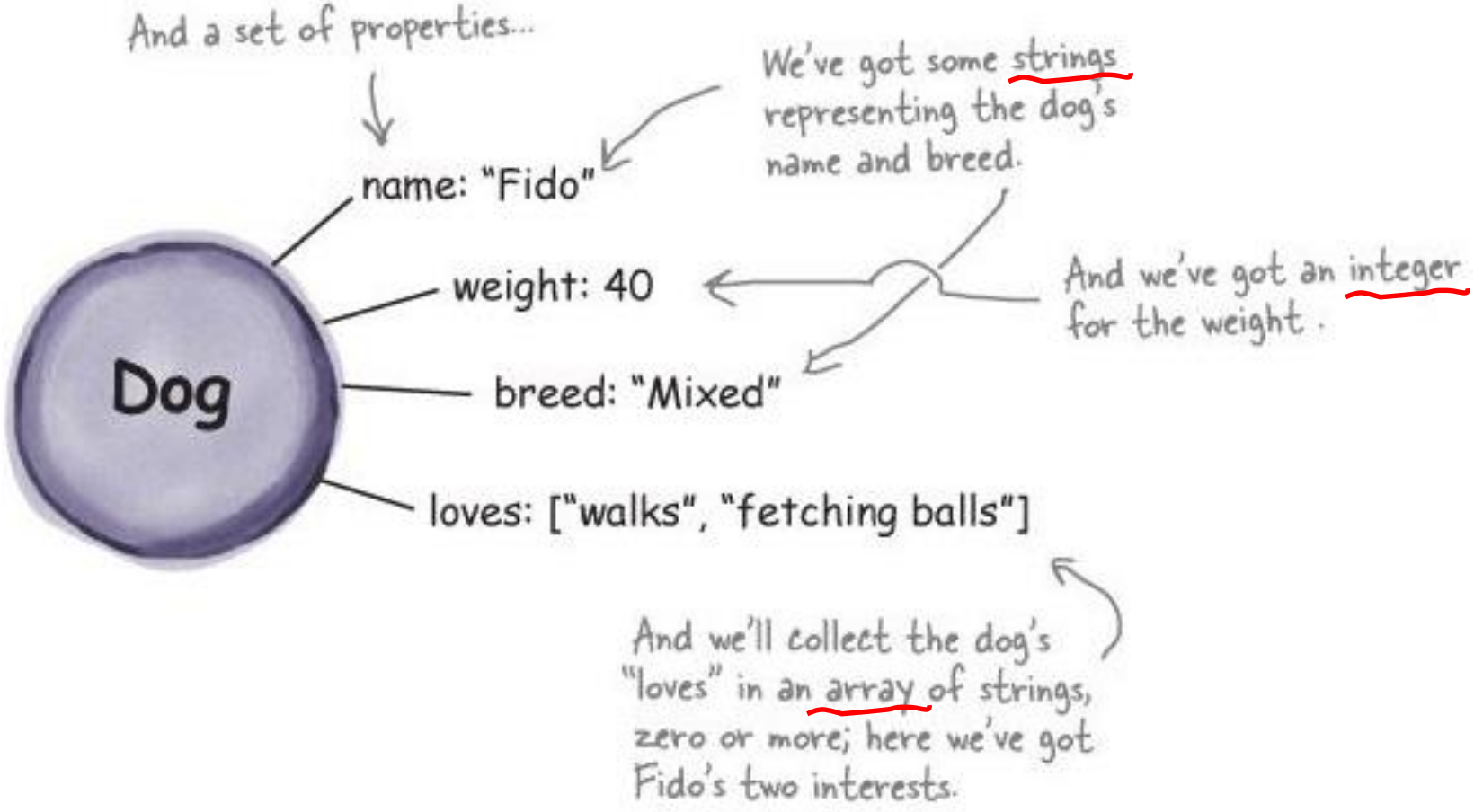
Let's take an example, say, a dog. A dog's got properties:



활동리스트
- 즐기다
- 걷다
- 볼을 가져오다

속성을 생각하여 봅시다

이런 속성들은 자바스크립트 데이터 유형과 연관 지어 생각하여 봅시다.



자바스크립트에서 객체를 만드는 방법

다음과 같이 객체를 만들어 보자

```
var fido = {
    name: "Fido",
    weight: 40,
    breed: "Mixed",
    loves: ["walks", "fetching balls"]
};
```

점 연산자(.)를 사용해서 객체의 속성에 접근

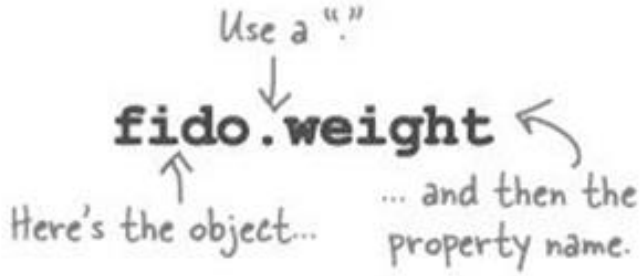
점 연산자(.)를 사용해서 객체의 속성에 접근 할 수 있습니다.
일반적으로 [“문자열”]표기법보다
더 읽기 쉬운 방법입니다.

- fido.weight is fido의 몸무게
- fido.breed is fido의 품종
- fido.name is fido의 이름
- fido.loves is fido가 좋아하는 것을 담고 있는 array

객체로 할 수 있는 것들 – 1

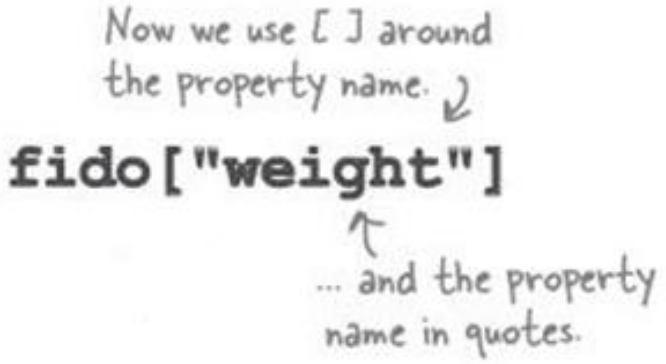
1. “dot” notation: 점 연산자로 객체 속성에 접근하기

```
if (fido.weight > 25) {
    alert("WOOF");
} else {
    alert("yip");
}
```



2. string 과 [ ] notation: [] 와 문자열로 속성에 접근하기

```
var breed = fido["breed"];
if (breed == "mixed") {
    alert("Best in show");
}
```



객체로 할 수 있는 것들 – 2

3. property's value: 속성값 변경 하기

```
fido.weight = 27;
fido.breed = "Chawalla/Great Dane mix";
fido.loves.push("chewing bones");
```

We're changing Fido's weight...

... his breed...

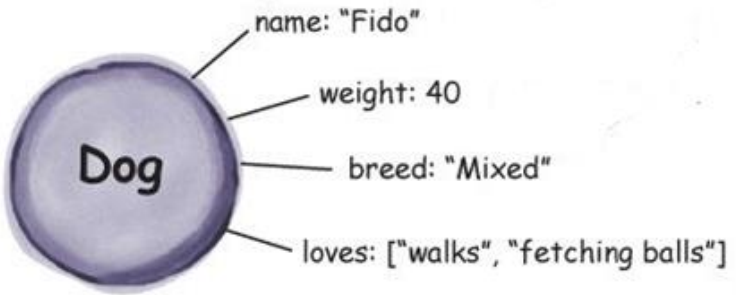
... and adding a new item to his loves array.

4. object's properties: 객체의 모든 속성 나열 하기

```
var prop;
for (prop in fido) {
    alert("Fido has a " + prop + " property ")
    if (prop == "name") {
        alert("This is " + fido[prop]);
    }
}
```

To enumerate the properties we use a for-in loop.

Name, weight, breed



=> dog4.html 테스트

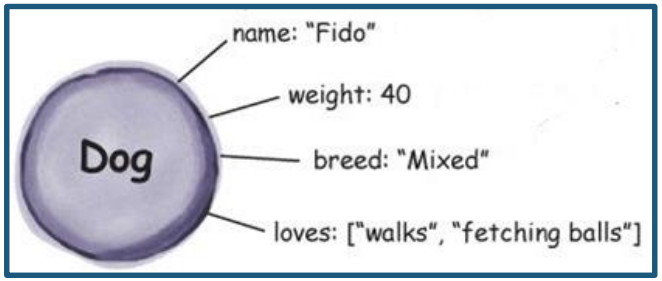
객체로 할 수 있는 것들 – 3

5. object's array: 객체의 배열 조작하기

```
var likes = fido.loves;
var likesString = "Fido likes";

for (var i = 0; i < likes.length; i++) {
    likesString += " " + likes[i];
}
alert(likesString);
```

=> dog5.html 배열로 만들어 테스트

fido likes 걷기1 공물어오기1 공을 물어뜯기1



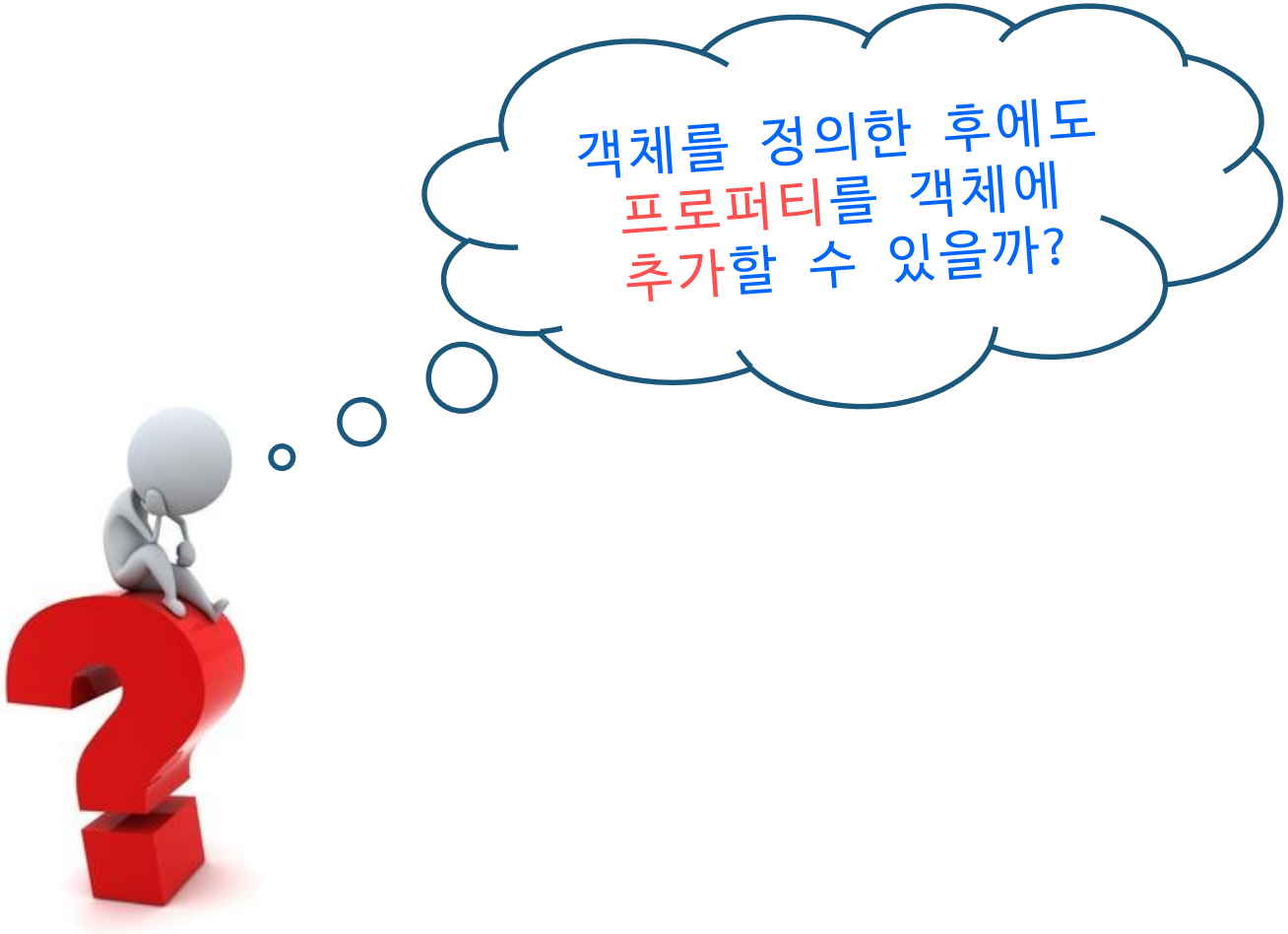

```
var fido = {
    name: "Fido",
    weight: 40,
    breed: "Mixed",
    loves: ["walks", "fetching balls"]
};
```

6. Pass an object to a function: 객체를 함수로 전달하기

```
function bark(dog) {
    if (dog.weight > 25) {
        alert("WOOF");
    } else {
        alert("yip");
    }
}
```

=> next 생성자

언제든지 add or delete properties ?



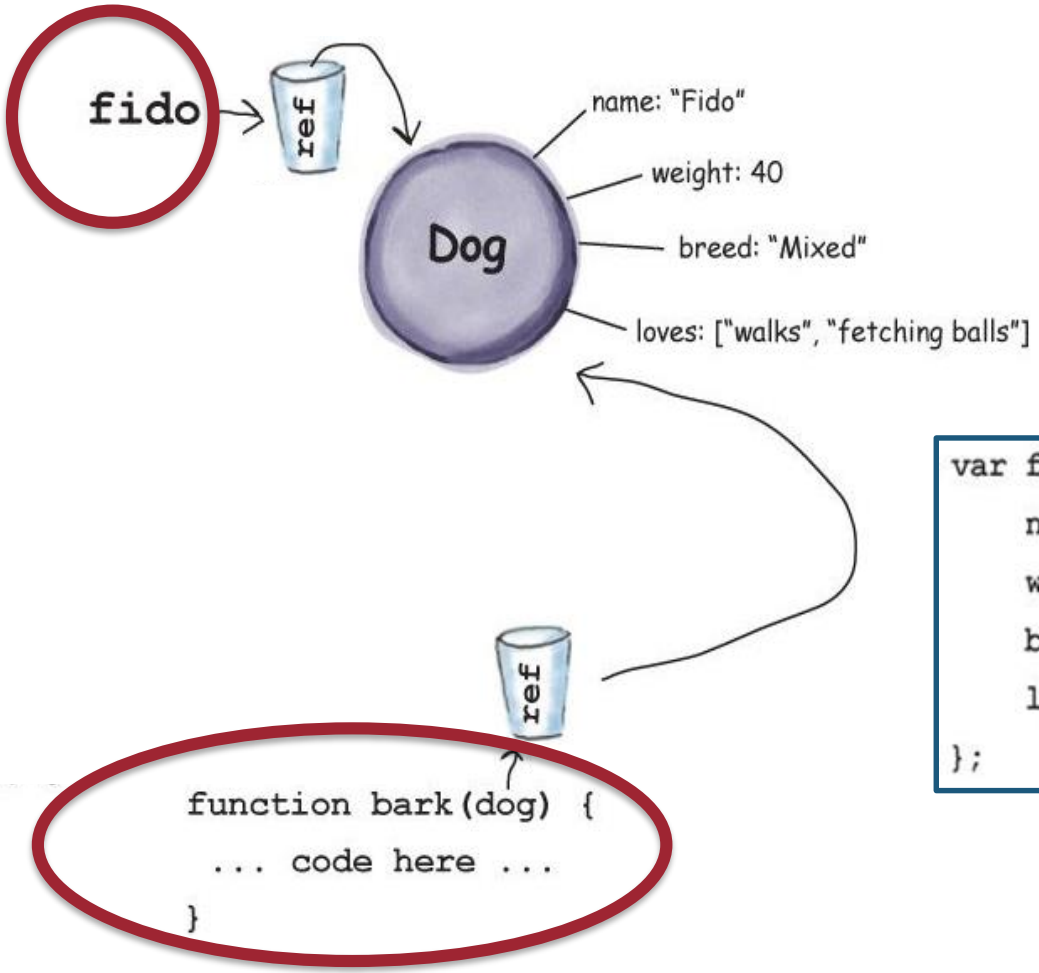
언제든지 add or delete properties !

- 객체에 프로퍼티를 추가하는 방법:
  - ✓ 단순히 새로운 프로퍼티에 값을 할당한다:
  - ✓ fido.age = 5;
  - ✓ 이 시점부터 객체 fido는 새로운 프로퍼티 age를 가지게 된다.

- 마찬가지로 delete 키워드를 이용하여 프로퍼티를 삭제할 수 있다:
  - ✓ delete fido.age;
  - ✓ 이것은 프로퍼티 값만 삭제하는 것이 아니라 프로퍼티 자체를 삭제하는 것이다.

objects 를 functions 로 전달하는 문제

- 객체가 변수에 할당될 때 변수는 객체 자체가 아니라 객체에 대한 레퍼런스를 저장, 레퍼런스를 객체에 대한 포인터로 간주하라.




```
var fido = {
    name: "Fido",
    weight: 40,
    breed: "Mixed",
    loves: ["walks", "fetching balls"]
};
```

중복 코드가 발생하는 문제는? – 생성자 사용

- 객체에 복사할 때마다 중복 코드가 발생한다.
- 생성자란 무엇인가?
  - 객체를 생성할 수 있고 그것들을 모두 똑같이 만들 수 있는 특수한 함수이다.
- 생성자 함수 생성

```
function Dog(name, breed, weight){
        this.name = name;
        this.breed= breed;
        this.weight = weight;
        this.bark = function(){ .....};
}
```

- "객체 지향" 프로그래밍의 목적 중의 하나가 바로 코드 재사용을 극대화 하는 것

```
var fido = new Dog("파이도", "잡종", 38);
var tiny = new Dog("타이니", "치와와", 8);
var clifford = new Dog("클리포드", "푸들", 65);
```

dog4_5up.html => 중복되는 문제 => 생성자 필요

this 키워드 사용 - 1

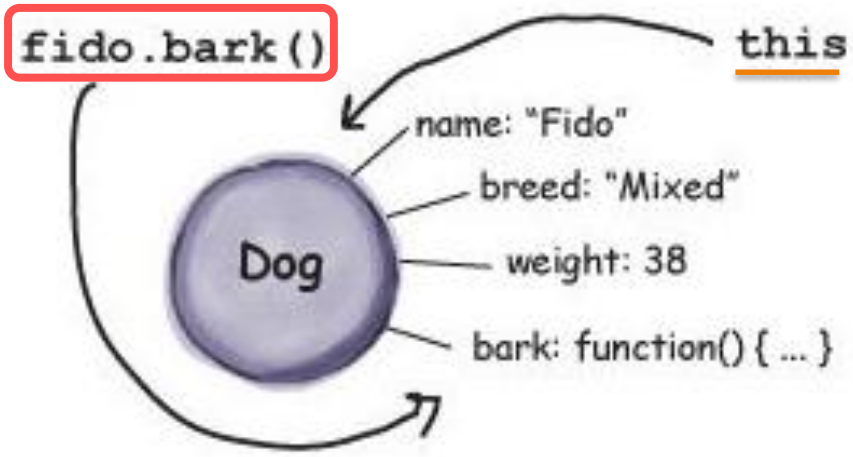
1. fido에 할당된 dog 객체를 얻었다고 하자

ex) var fido = new Dog("파이도", "잡종", 38);

this 키워드 사용 – 2

2. fido 객체의 bark()를 호출한다:

var fido = new Dog("파이도", "잡종", 38);




```
//생성자 코드
function Dog(name, breed, weight){
  this.name = name;
  this.breed= breed;
  this.weight = weight;
  this.bark = function(){
      if (this.weight >25){
      alert(this.name + "가 멍멍 짓습니다!");
      }else{
      alert(this.name + "깽깽 거립니다!");
      }
  };
}
```

- 생성자의 코드에 있는 모든 this는 생성자(메소드)를 호출한 객체의 레퍼런스로 해석된다.
- 따라서 fido.bark를 호출한다면 this는 fido를 참조하게 된다.

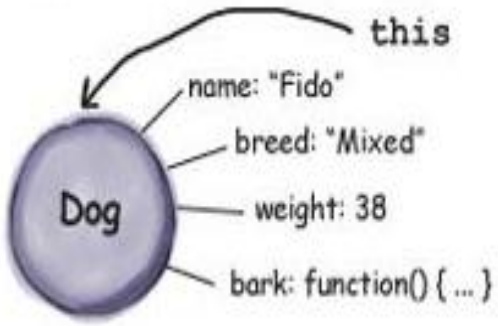
this 키워드 사용 – 3

3. "this"는 항상 메소드가 호출된 객체를 참조한다:

```
function Dog(name, breed, weight){
                this.name = name;
                this.breed= breed;
                this.weight = weight;
                this.bark = function(){ ……};
}
```

```
fido=new Dog("Fido","Mixed",38);
```

```
fido.bark()
```




```
tiny=new Dog("Fido","Mixed",38);
```

```
tiny.bark()
```


```
clifford=new Dog("Fido","Mixed",38);
```

```
clifford.bark()
```

- fido.bark를 호출한다면 this는 fido를 참조하게 된다
- tiny.bark를 호출한다면 this는 tiny를 참조하게 된다
- clifford.bark를 호출한다면 this는 clifford를 참조하게 된다

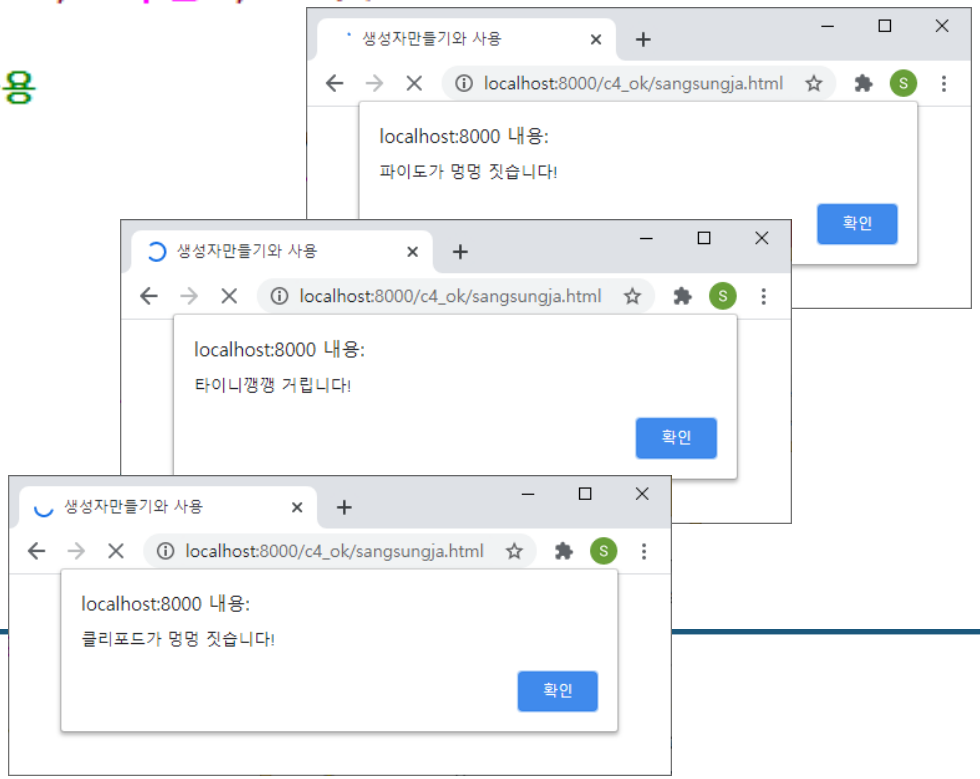
<실습5> Dog 생성자 함수 만들기 – 교재 참조(1/2)

sangsungja.html

```
<!doctype html>
<html lang="en">
<head>
<title>생성자만들기와 사용</title>
<meta charset="utf-8">
<script>
//1.개에 대한 생성자 함수 만들기

function Dog(name, breed, weight){
    this.name = name;
    this.breed= breed;
    this.weight = weight;
    this.bark = function(){
    //함수값을 bark 속성에 할당해서 초기화함으로써 메소드를 추가
        if (this.weight >25){
        alert(this.name + "가 멍멍 짓습니다!");
        }else{
        alert(this.name + "깽깽 거립니다!");
        }
    };//객체를 정의하는 것과 구문이 다르다는 점
      //문장이므로 일반적인 함수에서 하듯이 맨끝에 ; 를 붙인다.
}
```

<실습5> 생성자 사용 및 실행화면(2/2)

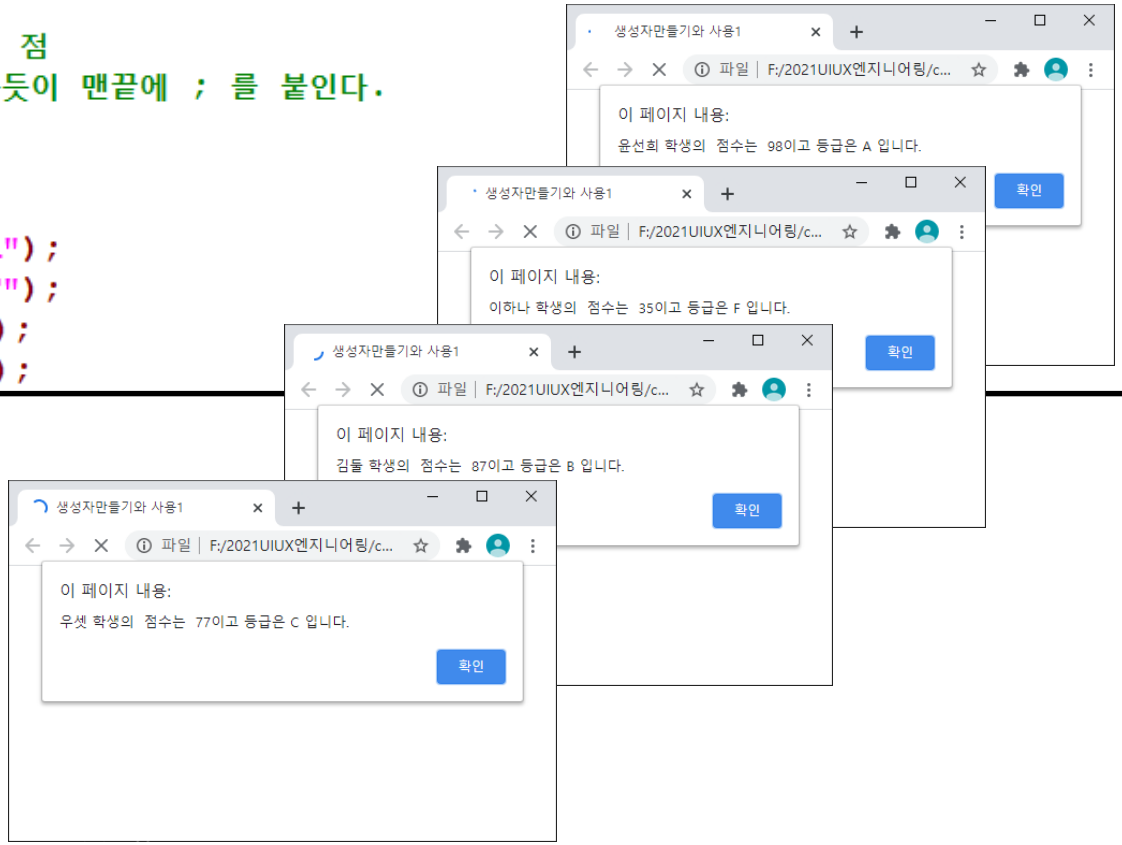
```
23 //2.생성자 사용
24 //dog생성
25 var fido = new Dog("파이도", "잡종", 38);
26 var tiny = new Dog("타이니", "치와와", 8);
27 var clifford = new Dog("클리포드", "푸들", 65);
28
29 //bark 메서드를 호출해서 작동해서 사용
30 fido.bark();
31 tiny.bark();
32 clifford.bark();
33
34 </script>
35 </head>
36 <body>
37 document.write("생성자 test");
38 </body>
39 </html>
```



<실습6> 학생에 대한 생성자 만들기 – 만들어 보자

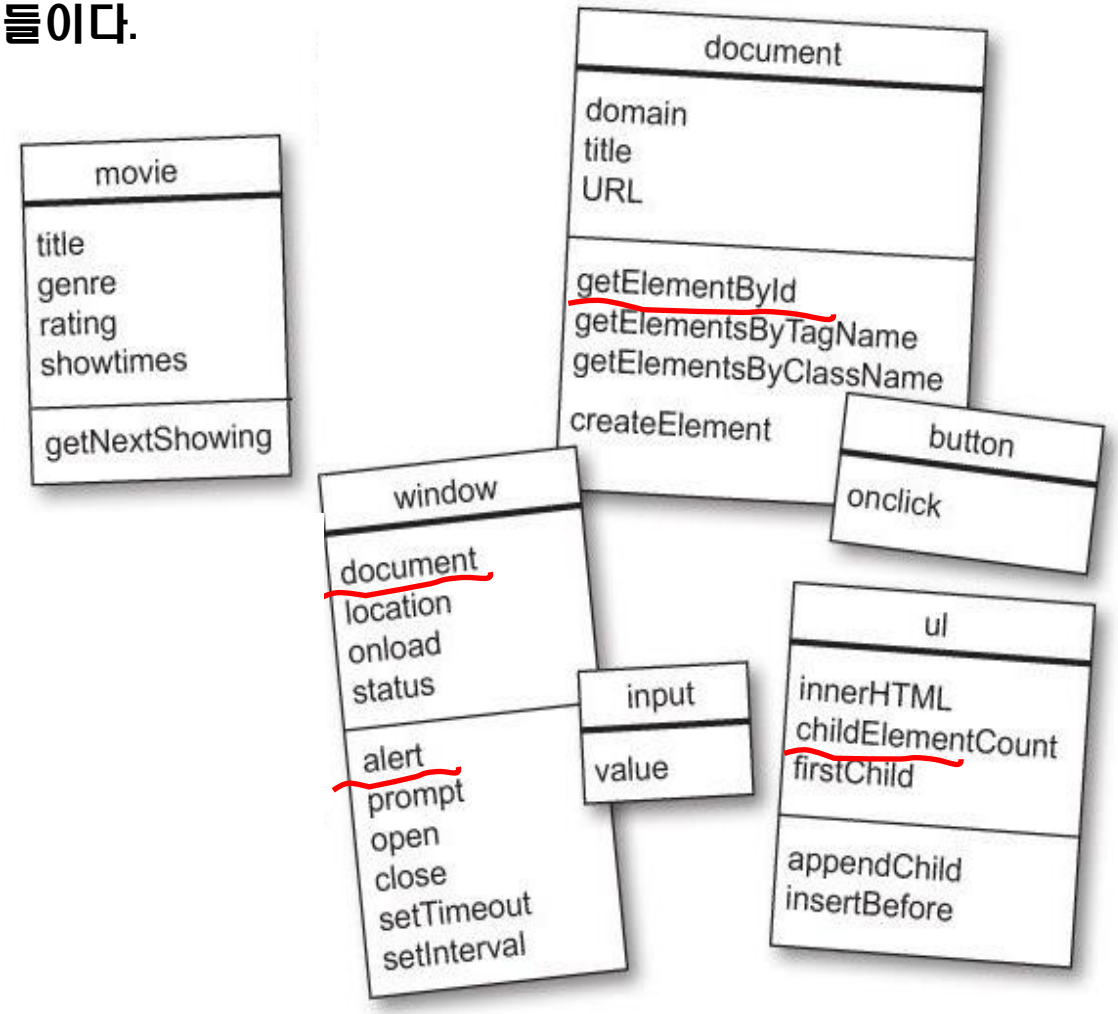
sangsungjaup.html

```
//1.학생에 대한 생성자 함수 만들기

function stud(name, score, grade){
    this.name = name;
    this.score= score;
    this.grade = grade;
    this.comment = function(){ //함수값을 comment 속성에 할당해서 초기화함으로써 메소드를 추가

        alert(this.name+" 학생의  점수는  " + this.score +"이고 등급은 "+ this.grade + " 입니다.");

    };//객체를 정의하는 것과 구문이 다르다는 점
        //문장이므로 일반적인 함수에서 하듯이 맨끝에 ; 를 붙인다.
}
//2.생성자 사용
//stud생성
var stud1 = new stud("윤선희", 98, "A");
var stud2 = new stud("이하나", 35, "F");
var stud3 = new stud("김둘", 87, "B");
var stud4 = new stud("우셋", 77, "C");
```



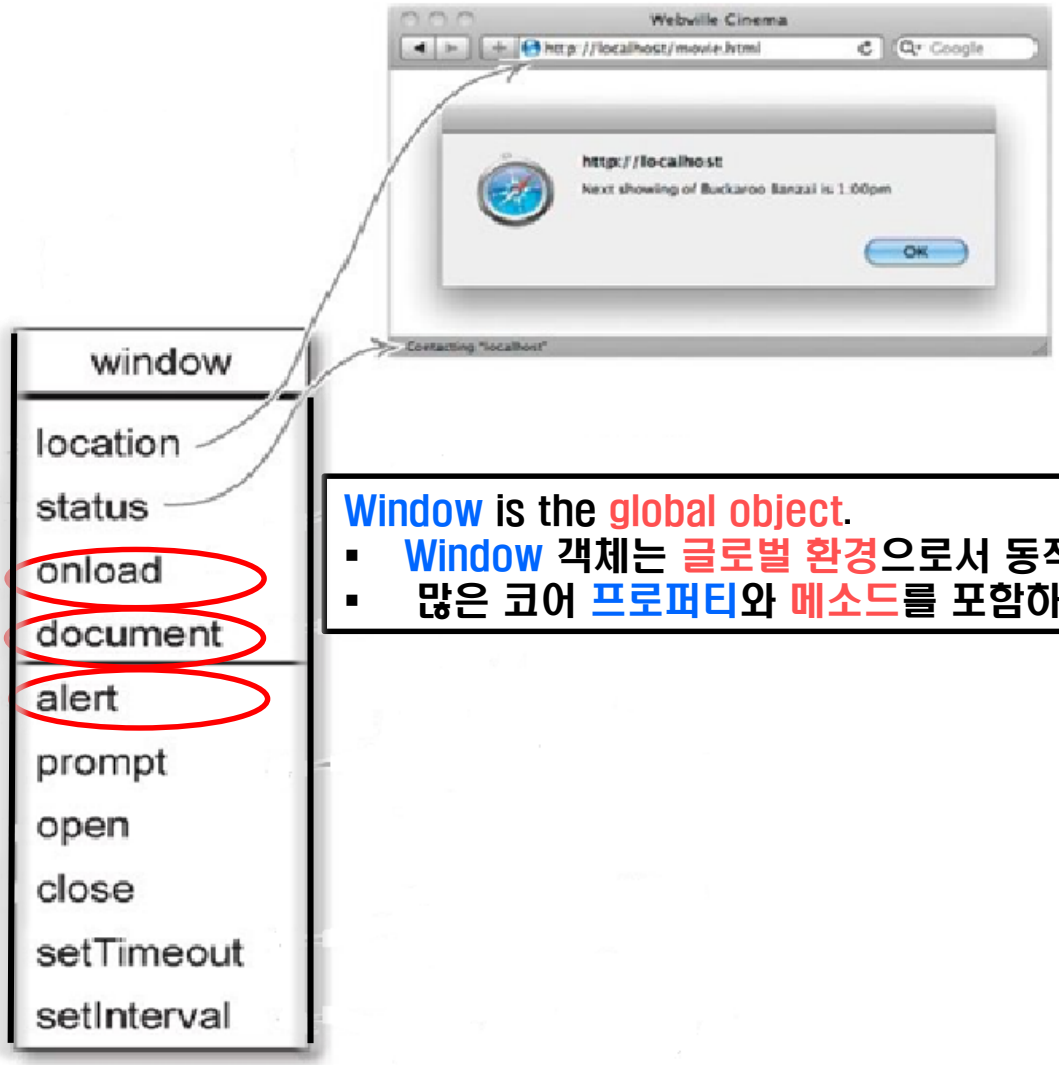
그 외 현장에서 사용되고 있는 자바스크립트 객체 는?

... 객체들이 주변에 널려있다.
예를 들어, document.getElementById로 부터 돌려 받은
엘리먼트들과 마찬가지로
document와 window도 객체들이다.



도데체 window 객체의 정체는 무엇일까요?

- window 객체는 자바스크립트 프로그램을 위한 글로벌 환경(global environment)과 애플리케이션의 메인 윈도우(main window)를 제공해준다.



Window is the global object.

- Window 객체는 글로벌 환경으로서 동작한다.
- 많은 코어 프로퍼티와 메소드를 포함하고 있다.

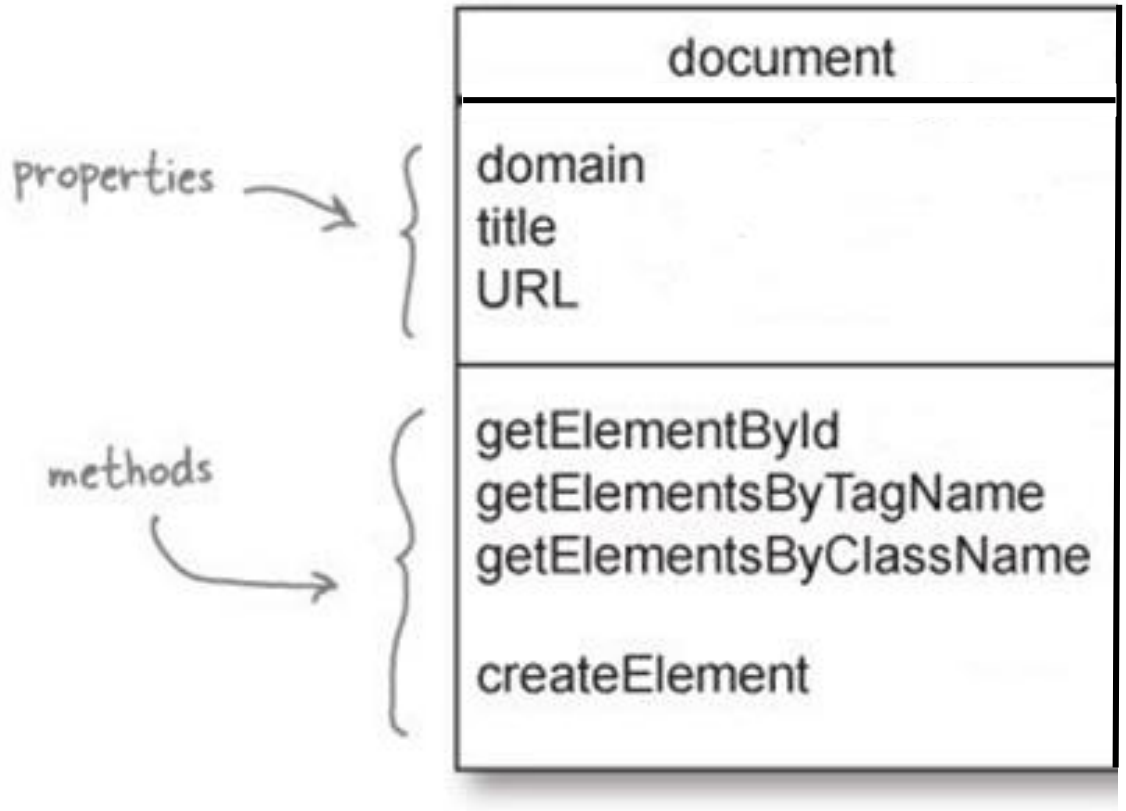
window.onload 자세히 살펴보기

- window.onload event handler
- Window.onload 속성에 함수를 할당함으로써 페이지가 로드 되고 DOM 설정이 완전히 끝날 때 까지 코드가 실행되지 않도록 한다.

```
window.onload = function() {
    // code here
};
```

document 객체 자세히 살펴보기

- document 객체는 DOM을 접근(access)하는데 사용된다
- 앞에서 보았듯이 사실 document 객체는 window 객체의 프로퍼티이다
- 물론 window.document처럼 사용하지는 않는다 (보통생략)



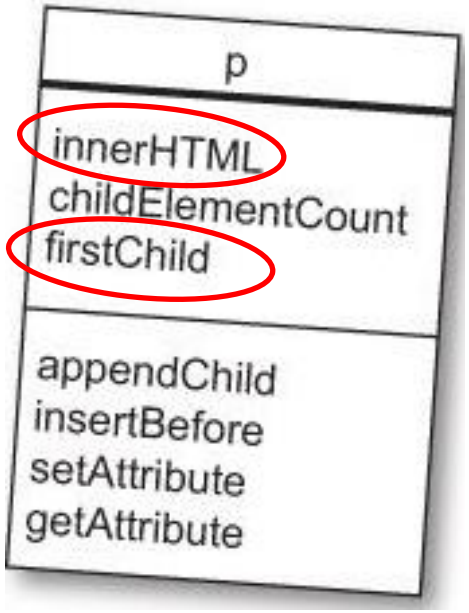
document.getElementById 자세히 살펴보기

- Document는 document객체로, DOM에 접근할 수 있게 하는 빌트인 자바스트립트 객체입니다.
- getElementById 메서드는 id를 이용해서 요소를 가져온다.

```
var div = document.getElementById("myDiv");
```

element 객체 자세히 살펴보기

- getElementById 에 의해 반환되는 엘리먼트 또한 객체이다.
- 이미 앞에서 innerHTML과 같은 엘리먼트 프로퍼티를 본 적이 있다.




```
window.onload = outputResults;
function outputResults() {
    var output = document.getElementById("output");
    output.innerHTML = "강아지 나이: " + myDogsAge
    + "<br>사각형 면적: " + rectArea
    + "<br>합계: " + theTotal + "<br>내 아바타: " + myAvatar;
}
</script>
</head>
<body>
<div id="output">
</div>
```

window객체와 document객체의 특징

-window 객체는 global object 이다.
-window 객체는 자바스크립트 프로그램과 애플리케이션 메인 원도우의 global environment 객체이며 main window를 제공하며 많은 코어 프로퍼티와 메소드를 포함하고 있다.
-window는 생략한다.

- document 객체는 DOM을 접근(access)하는데 사용된다.
- document 객체는 window 객체의 프로퍼티이다.
- getElementById 에 의해 반환되는 엘리먼트 또한 객체이다.

function & object의 다양한 예제 다루기

- 자바스크립트의 변수 타입
- 자바스크립트의 객체 타입
- 자바스크립트의 함수
- 익명 함수
- 객체의 속성으로 함수 할당1
- 객체의 속성으로 함수 할당2
- 객체 선언 시 속성 정의

자바와 자바스크립트의 변수 타입 비교

- 자바는 자료형(타입)을 명시하는 언어
- 자바스크립트는 자료형을 명시하지 않는 언어
- 내부에서는 자료형에 따라 변수 상자의 크기가 달라짐

자바스크립트의 자료형

- boolean, number, string 이 있으며, 그 외에 undefined, null, Object 자료형이 있음

자료형	설명
Boolean	[기본 자료형] true와 false의 두 가지 값을 가지는 자료형
Number	[기본 자료형] 64비트 형식의 IEEE 754 값이며 정수나 부동소수 값을 가지는 자료형 몇 가지 상징적인 값을 가질 수 있음: NaN(숫자가 아님), +무한대(Number.MAX_VALUE로 확인), -무한대(Number.MIN_VALUE로 확인)
String	[기본 자료형] 문자열 값을 가지는 자료형
undefined	값을 할당하지 않은 변수의 값
null	존재하지 않는 값을 가리키는 값
Object	객체를 값으로 가지는 자료형 객체는 속성들을 담고 있는 가방(Collection)으로 볼 수 있으며, 대표적인 객체로 Array나 Date를 들 수 있음

변수 만들기

- 변수 앞에는 var 키워드를 붙임

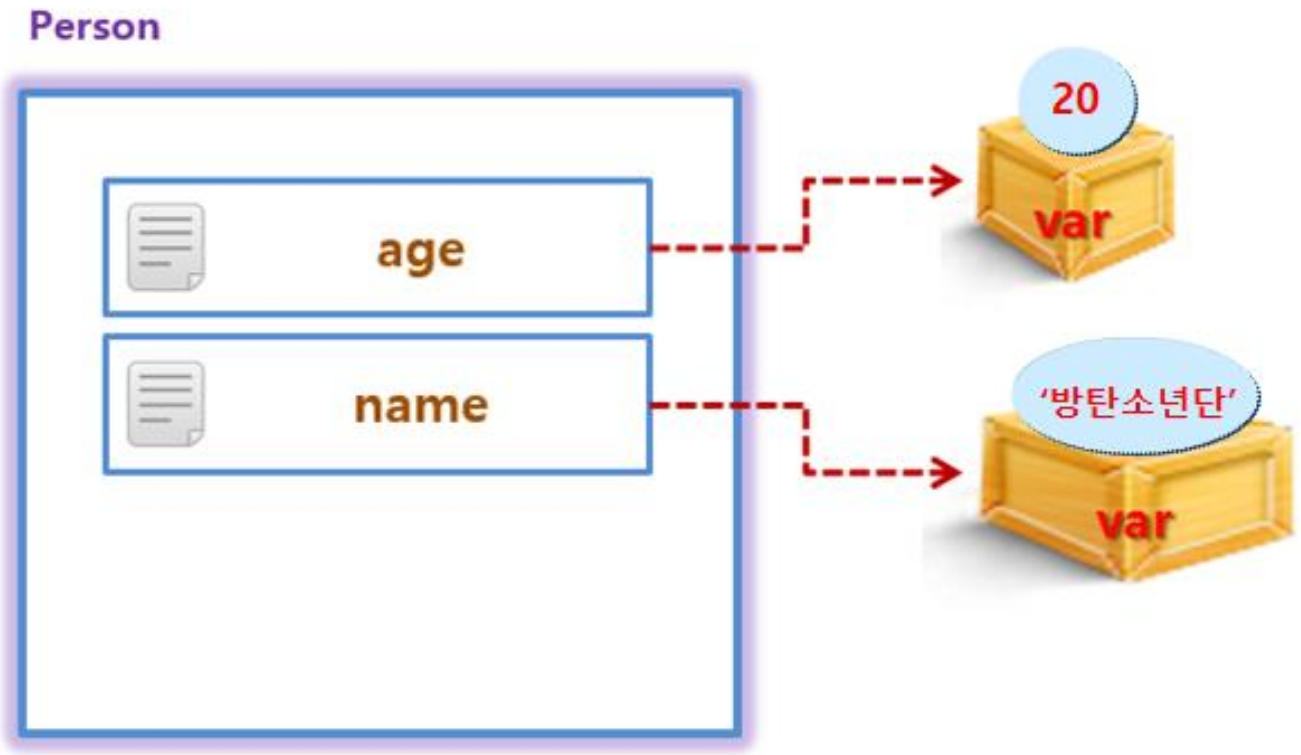
```
var age = 20;
console.log('나이 : %d', age);
var name = '방탄소년단';
console.log('이름 : %s', name);
```

test1.html 자바스크립트의 변수 타입

```
var name;
console.log('name :' + name);

var age = 20;
console.log('나이 : %d', age);

var name = '방탄소년단';
console.log('이름 : %s', name);
```

자바스크립트의 객체

- 속성들이 이름 – 값 의 형태로 들어가 있음



객체 만들기

- 객체는 { } 기호를 이용해 만듬
- 객체 안의 속성은 . 연산자를 이용해 접근하거나 객체이름 뒤에 [ ] 를 붙이고 그 안에 속성 이름을 문자열로 넣어 접근할 수 있음

```
var Person = {};
Person["age"] = 20;
Person["name"] = '방탄소년단';
Person.mobile = '010-1000-1000';
console.log('나이 : %d', Person.age);
console.log('이름 : %s', Person.name);
console.log('전화 : %s', Person["mobile"]);
```

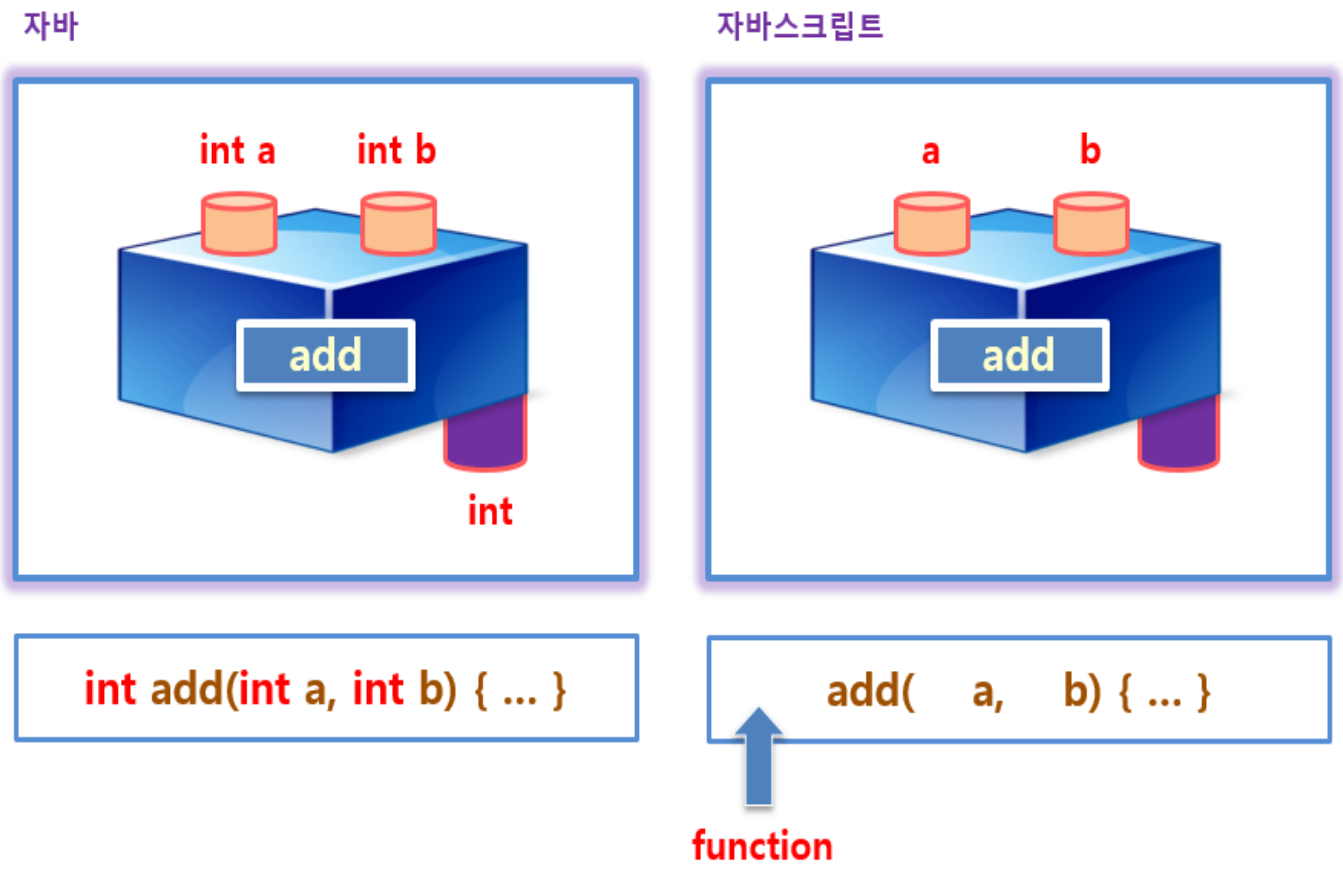
test2.html 자바스크립트의 객체 타입

```
var Person = {};

Person['age'] = 20;
Person['name'] = '방탄소년단';
Person.mobile = '010-1000-1000';

console.log('나이 : %d', Person.age);
console.log('나이 : %d', Person['age']);
console.log('이름 : %s', Person.name);
console.log('이름 : %s', Person['name']);
console.log('전화 : %s', Person.mobile);
console.log('전화 : %s', Person['mobile']);
```

- 객체는 중괄호를 이용해 만들어지며 그 안에 속성을 추가할 수 있음

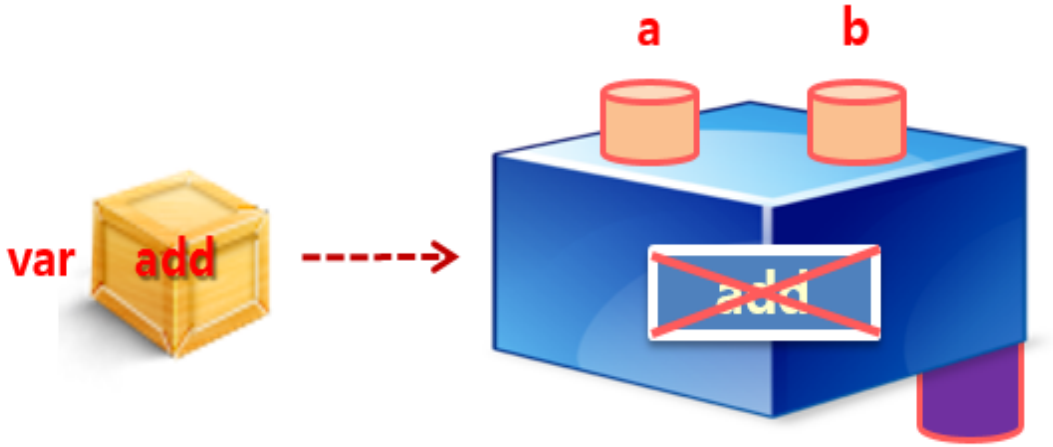
자바와 자바스크립트의 함수 비교

- 파라미터의 타입과 반환되는 값의 타입을 명시하지 않음
- 함수 앞에는 function 키워드를 붙임



함수를 변수에 할당하기

- 자바스크립트에서는 함수를 일급 객체(First class object)로 다룸
- 따라서, 함수가 변수에 할당될 수 있음
- 변수에 할당될 경우 두 가지 이름으로 함수를 호출할 수 있으므로 원래의 함수 이름을 생략하고 익명함수(Anonymous Function)라고 부름




```
var add = function (    a,    b) { ... };
```

함수 만들어 실행하기

- 함수를 만들고 실행할 수 있음
- 선언문 (Declaration)

```
function add(a, b) {
    return a + b;
}
var result = add(10, 10);
console.log('더하기 (10, 10) : %d', result);
```

test3.html 자바스크립트의 함수

```
function add(a, b) {
        return a + b;
}

var result = add(10, 10);

console.log('더하기 (10, 10) : %d', result);
```

- 자바와 같은 타입 기반 언어의 함수에서 타입만 제외한 형태

함수 만들어 변수에 할당하기 – 익명함수

- 변수 이름으로 호출 가능

```
var add = function (a, b) {
    return a + b;
};
var result = add(10, 10);
console.log('더하기 (10, 10) : %d', result);
```

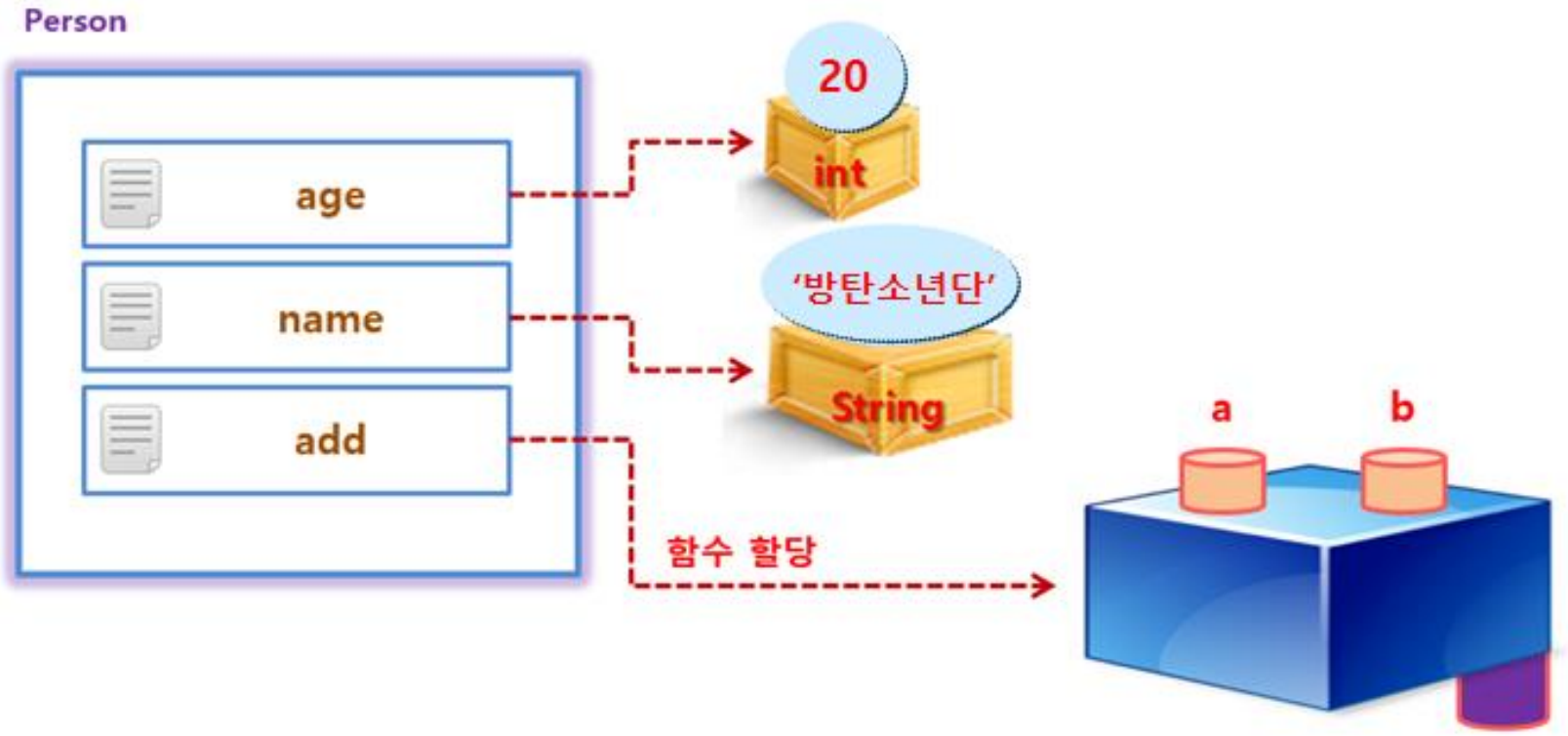
test4.html 익명 함수

```
var add = function (a, b) {
  return a + b;
};

var result = add(10, 10);

console.log('더하기 (10, 10) : %d', result);
```

- 변수의 값으로 함수가 할당될 수 있음
- 변수의 값으로 함수를 구분하므로 함수의 이름을 넣을 필요가 없음
- 함수 선언문이 아니라 함수 표현식으로 추가함

객체의 속성으로 함수 할당하기

- 객체의 속성도 변수처럼 처리되므로 함수 할당 가능



객체의 속성에 함수 할당

```
var Person = {};
Person["age"] = 20;
Person["name"] = '방탄소년단';
Person.add = function(a, b) {
    return a + b;
};
console.log('더하기 : %d', Person.add(10, 10));
```

test5.html 객체의 속성으로 함수 할당

```
var Person = {};

Person['age'] = 20;
Person['name'] = '방탄소년단';
Person.add = function(a, b) {
   return a + b;
};
console.log('더하기 : %d', Person.add(10,10));
```

- 객체의 속성으로 함수 할당
- 함수는 변수에 할당할 수 있으므로 객체의 속성으로도 함수를 할당 할 수 있음

test6.html 객체의 속성으로 함수 할당

```
var Person = {};

Person['age'] = 20;
Person['name'] = '방탄소년단';

var oper = function(a, b) {
   return a + b;
};

Person['add'] = oper;

console.log('더하기 : %d', Person.add(10,10));
```

- 함수를 변수에 할당한 후 객체의 속성으로 함수를 할당할 수 있음

객체를 만들 때 속성 할당하기

```
var Person = {
    age: 20,
    name: '방탄소년단',
    add: function(a, b) {
        return a + b;
    }
};
console.log('더하기 : %d', Person.add(10, 10));
```

test7.html 객체 선언 시 속성 정의

```
var Person = {
    age: 20,
    name: '방탄소년단',
    add: function(a, b) {
        return a + b;
    }
};
console.log('더하기 : %d', Person.add(10,10));
```

객체 활용 – Webville Cinema를 소개 합니다.

2개의 간단한 movie 객체를 설계해 보자:

- 각 객체는 a title, a genre, a movie rating (1–5 stars) and a set of showtimes를 포함한다

Movie 객체를 어떻게 만들었나요?

movie 객체를 어떻게 생성할 수 있을까?

```
var banzaiMovie = new Movie("카우보이 밴자이", "고전 컬트", 5,
                                        ["1:00pm","5:00pm","7:00pm","11:00pm"]);


var plan9Movie = new Movie("외계로부터의 9호 계획", "고전 컬트", 2,
                                        ["3:00pm", "7:00pm", "11:00pm"]);
```

다음 편 영화는…

- 우리는 객체와 함수를 혼합하여 사용하는 것을 이미 살펴보았다.
- 이제 한 단계 더 나아가서 영화의 다음 상영시간(showtime)을 알려주는 자바스크립트 코드를 작성해 보자.
- 작성하게 될 함수는 movie 객체를 아규먼트로 받아 들여 현재시간을 기준으로 다음 상영시간을 알려주는 문자열을 리턴 하도록 한다.

준비된 코드(baked code)

```
function getTimeFromString(str) {
    var theTime = new Date();
    var time = str.match(/(\d+)(?::(\d\d))?\s*(p?)/);
    theTime.setHours( parseInt(time[1]) + (time[3] ? 12 : 0) );
    theTime.setMinutes( parseInt(time[2]) || 0 );
    return theTime.getTime();
}
```

- 1am, 3am **형식의 문자열을 받아서 밀리세컨드 단위로 변환하는 미리 준비한 코드**

<실습7> 영화상영 안내 – 객체 만들기(1/3)

cinema.html

```
<!doctype html>
<html lang="en">
<head>
<title>웹마을 시네마</title>
<meta charset="utf-8">
<script>
window.onload = function() {

    var banzaiMovie = new Movie("카우보이 밴자이",
                                "고전 컬트",
                                5,
                                ["1:00pm", "5:00pm", "7:00pm", "11:00pm"]);
    alert(banzaiMovie.getNextShowing());

    var plan9Movie = new Movie("외계로부터의 9호 계획",
                               "고전 컬트",
                               2,
                               ["3:00pm", "7:00pm", "11:00pm"]);
    alert(plan9Movie.getNextShowing());

    var forbiddenPlanetMovie = new Movie("금지된 세계",
                                         "고전 SF",
                                         5,
                                         ["5:00pm", "9:00pm"]);
    alert(forbiddenPlanetMovie.getNextShowing());

}
```

<실습7> 영화상영 안내 – Movie 함수 만들기 – 교재 참조(2/3)

cinema.html

```
28
29 function Movie(title, genre, rating, showtimes) {
30     this.title = title;
31     this.genre = genre;
32     this.rating = rating;
33     this.showtimes = showtimes;
34     this.getNextShowing = function() {
35         var now = new Date().getTime();
36         for (var i = 0; i < this.showtimes.length; i++) {
37             var showtime = getTimeFromString(this.showtimes[i]);
38             if ((showtime - now) > 0) {
39                 return this.title + "의 다음 상영시간은 " + this.showtimes[i] + "입니다";
40             }
41         }
42         return null;
43     };
44 }
45
46 function getTimeFromString(str) {
47 //준비된 코드
48 }
49
50 </script>
51 </head>
52 <body>
53 </body>
54 </html>
```

- 자바에서처럼 클래스로 안 만들고 함수를 그대로 틀로 사용

<실습7> 실행결과(3/3)

cinema.html

현재시간 : 오후 4시 경우

<실습8> cinema_up.html 만들기

다음과 같은 영화 객체를 추가 후 프로그램을 실행하여 보자 :

```
var mirimMovie = new Movie("미림 영상",
                "현대 SF",
                 3,
                 ["3:00pm", "4:00pm", "9:00pm", "10:00pm"]);
 alert(mirimMovie.getNextShowing());
```

<실습9> cinema_trace.html – 교재 참조

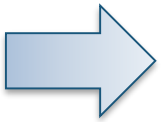
document.write 를 추가하고 프로그램 실행과정을 추적하여 보자 :

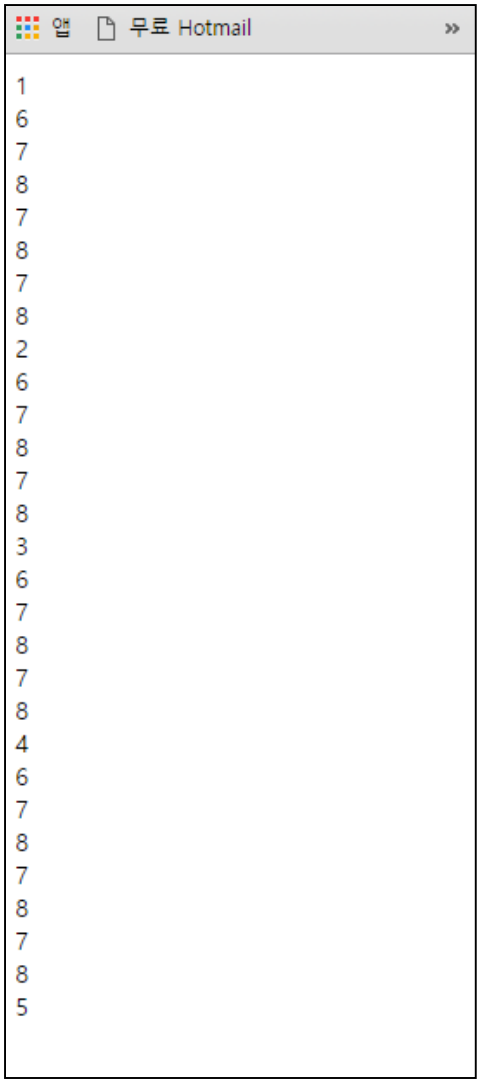
```
document.write("1"+"<br>");
~ document.write("8"+"<br>");
```

```
document.write("5"+"<br>");
document.write("3300_윤선희"+"<br>");
```

```
var mirimMovie = new Movie("미림 영상",
                "현대 SF",
                 3,
                 ["3:00pm", "4:00pm", "9:00pm", "10:00pm"]);
 alert(mirimMovie.getNextShowing());
```

1. 영화시간과
   실행하는 시간 따라 결과가 다르게 나옴
   (컴퓨터 시간)

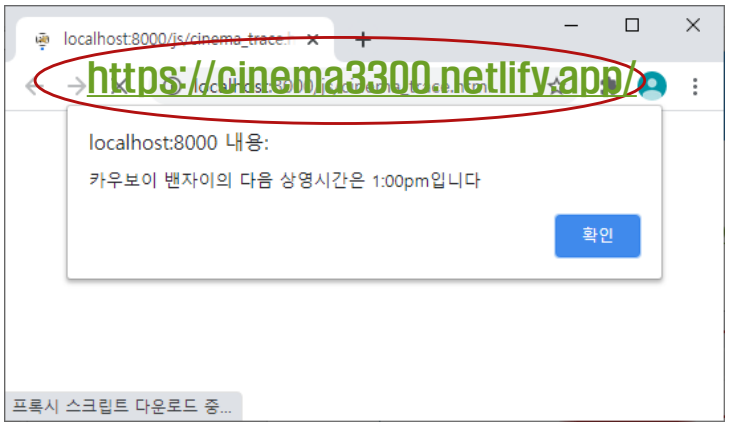




<실습> 곡 제시 프로그램 – 비동기 방식 실행과정 분석하기

1. 실행 첫 화면

https://cinema3300.netlify.app/



2. 최종 화면

<조건>

1. document 1~ 8까지와 학번이름까지 넣기
2. 영상 상영시간 모두 다르게 수정(자유롭게)
3. 현재 실행 시간 : 학생 컴퓨터 시간
4. 하이퍼링크 넣고 클릭하면 실행하기
5. 제출 프로그램 index.html(cinema_trace.html)

<nettify 에 올릴때 주의사항>

폴더를 만들고 폴더 안에 index.html로 이름 바꾸기

그외 JavaScript String 객체 활용

JavaScript String indexOf() Method

▶ indexOf() Method

The indexOf() method returns the position of the first occurrence of a specified value in a string.
This method returns -1 if the value to search for never occurs.

▶ http://www.w3schools.com/jsref/jsref_indexof.asp 참조

<실습10> String indexOf() Method – 교재 참조

indexof.html

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button to locate where in the string a specifed value occurs.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
    var str = "Hello world, welcome to the universe.";
    var n = str.indexOf("welcome");
    document.getElementById("demo").innerHTML = n;
}
</script>
</body>
</html>
```

<실습11> 곡 제시 함수 만들기 – 교재 참조

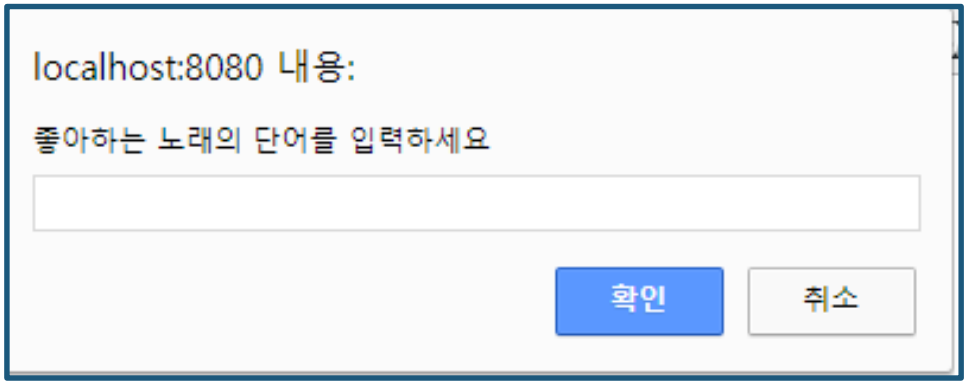
songs.html

```
14  function suggestSong() {
15      var index = Math.floor(Math.random() * songs.length);
16      var suggestion = songs[index];
17      return suggestion;
18  }
19  function searchForSong(word) {
20      var song;
21      for (var i = 0; i < songs.length; i++) {
22          song = songs[i];
23          if (songs[i].indexOf(word) >= 0) {
24              return song;
25          }
26      }
27      return "없는 곡입니다";
28  }
```

<실습12> 찾고자 하는 노래를 신청 받기 – 교재 참조

파일명은 songsup.html 으로 다음과 같이 신청곡이 있는가 검색할 수 있고 추천곡을 받는 프로그램을 만들어 보자.

songsup.html



총5 곡이 준비되어 있습니다.

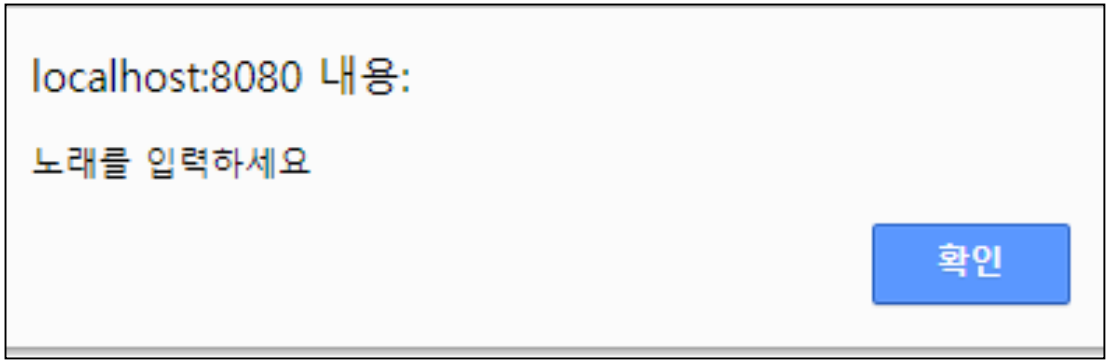
5번째곡 오늘의 추천곡은 수업시간이 신나요입니다.

봄이 왔어요곡이 있습니다

<실습13> 노래가 입력이 안 되는 경우 체크 추가

songsupgrade.html

```
5  if (num == "")
6  {
7          alert("노래를 입력하세요");
8          var num = prompt("좋아하는 노래의 단어를 입력하세요","봄이 왔어요");
9  }
```



Q & A