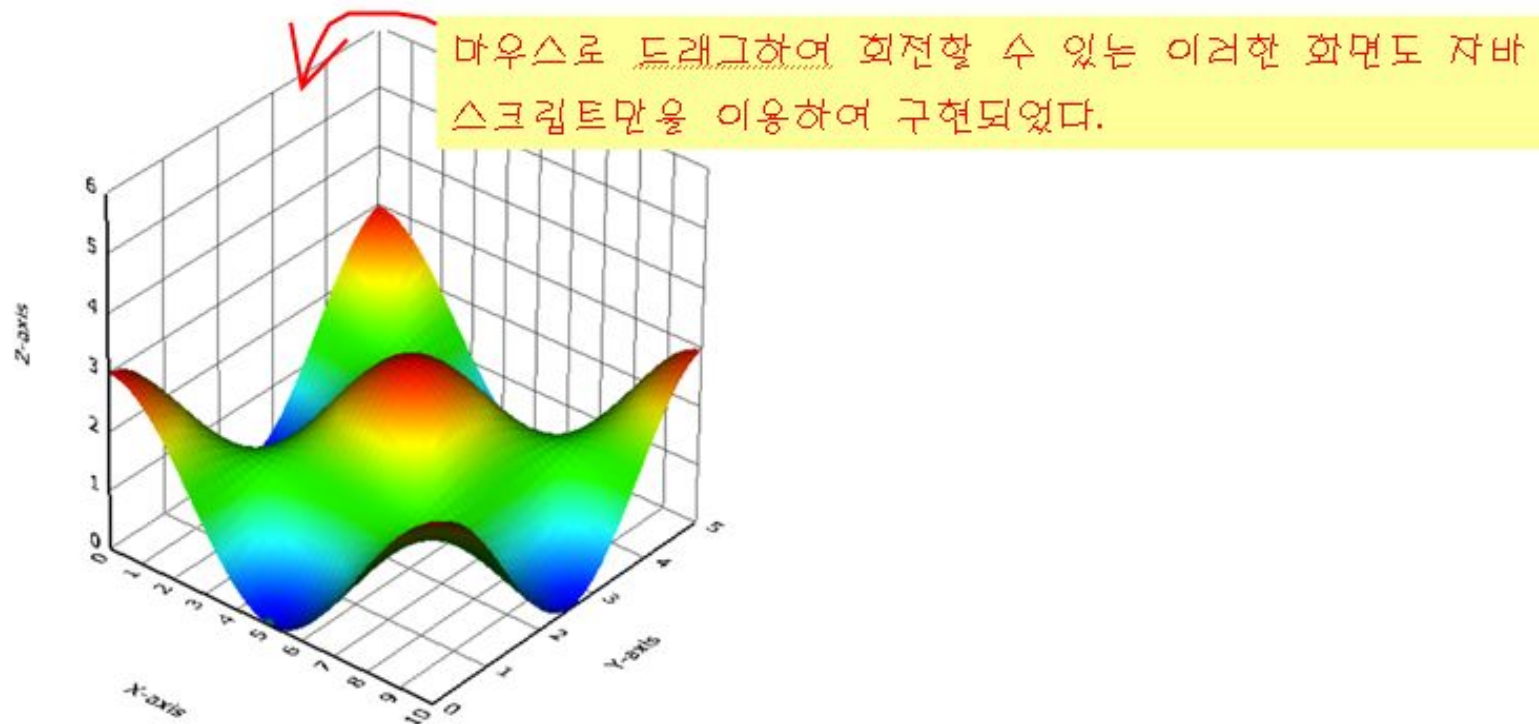


## 08 자바스크립트 기초

# 자바스크립트 소개

- 자바스크립트(javascript): 동적인 웹 페이지를 작성하기 위하여 사용되는 언어
- 웹의 표준 프로그래밍 언어
- 모든 웹브라우저들은 자바스크립트를 지원



# HTML5 기술의 핵심



# 자바스크립트 역사

- 넷스케이프의 브렌던 아이크(Brendan Eich)가 개발
- 처음에는 라이브스크립트(LiveScript)
- 최신 버전은 자바스크립트 1.8.5
- ECMA(European Computer Manufacturer's Association)이 ECMAScript라는 이름으로 표준을 제정-> ECMA-262

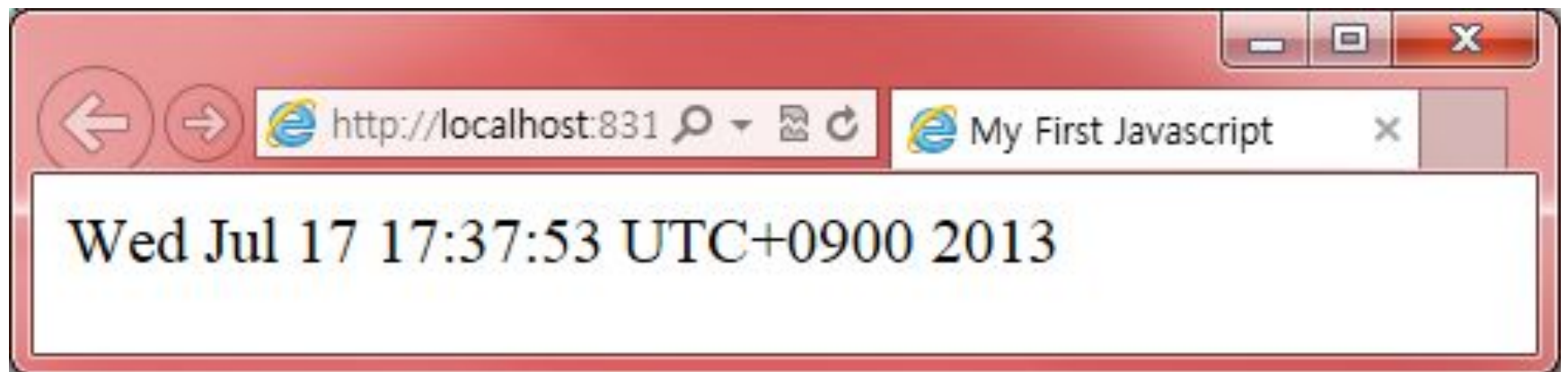


# 자바스크립트 특징

- 인터프리트 언어 - 컴파일 과정을 거치지 않고 바로 실행시킬 수 있는 언어
- 동적 타이핑(dynamic typing) - 변수의 자료형을 선언하지 않고도 변수를 사용할 수 있는 특징
- 구조적 프로그래밍 지원 - C언어의 구조적 프로그래밍을 지원한다. 즉 **if else, while, for** 등의 제어 구조를 완벽 지원
- 객체 기반 - 전적으로 객체지향언어이다. 자바스크립트의 객체는 연관배열(**associative arrays**)
- 함수형 프로그래밍 지원 - 자바스크립트에서 함수는 일급 객체(**first-class object**)이다. 즉 함수는 그 자체로 객체이다. 함수는 속성과 **.call()**과 같은 메서드를 가진다.
- 프로토타입-기반(**prototype-based**) - 상속을 위해 클래스 개념 대신에 프로토타입을 사용

# 첫 번째 예제

```
<!DOCTYPE HTML>
<html>
<head>
  <title> 첫 번째 Javascript </title>
</head>
<body>
  <script>
    var now = new Date();
    document.write(now);
  </script>
</body>
</html>
```



# 자바스크립트의 용도

- 이벤트에 반응하는 동작을 구현할 수 있다.
- **AJAX**를 통하여 전체 페이지를 다시 로드하지 않고서도 서버로부터 새로운 페이지 콘텐츠를 받거나 데이터를 제출할 때, 사용한다.
- **HTML** 요소들의 크기나 색상을 동적으로 변경할 수 있다.
- 게임이나 애니메이션과 같은 상호 대화적인 콘텐츠를 구현할 수 있다.
- 사용자가 입력한 값들을 검증하는 작업도 자바스크립트를 이용한다.



# 자바스크립트의 미래

- 자바스크립트는 본래 클라이언트 웹페이지를 위한 프로그래밍 언어였지만 그 용도는 점점 더 확장되고 있다.
- **Node.js** : 웹서버와 같은 애플리케이션을 작성하기 위해 설계된 서버-사이드(Server-Side) 소프트웨어 시스템



- **jQuery** : 자바스크립트 라이브러리



- **JSON** : 자바스크립트의 객체 표기법(Javascript Object Notation)은 개발 언어 독립적인 데이터 형식으로서 데이터 전송용 XML을 대체하고 있다. 심지어 문서 데이터베이스의 표준 저장 형식으로도 사용된다.



JavaScript Object Notation

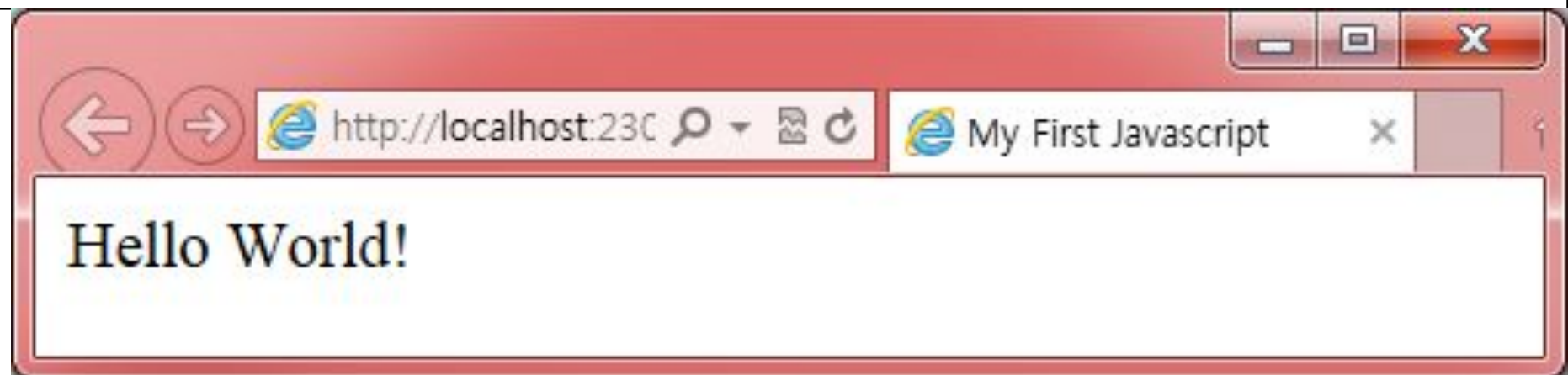


# 자바 스크립트의 위치

- 내부 자바스크립트
- 외부 자바스크립트
- 인라인 자바스크립트

# 내부 자바 스크립트

```
<!DOCTYPE HTML>
<html>
<head>
  <title>My First Javascript </title>
  <script>
    document.write("Hello World!");
  </script>
</head>
<body></body>
</html>
```

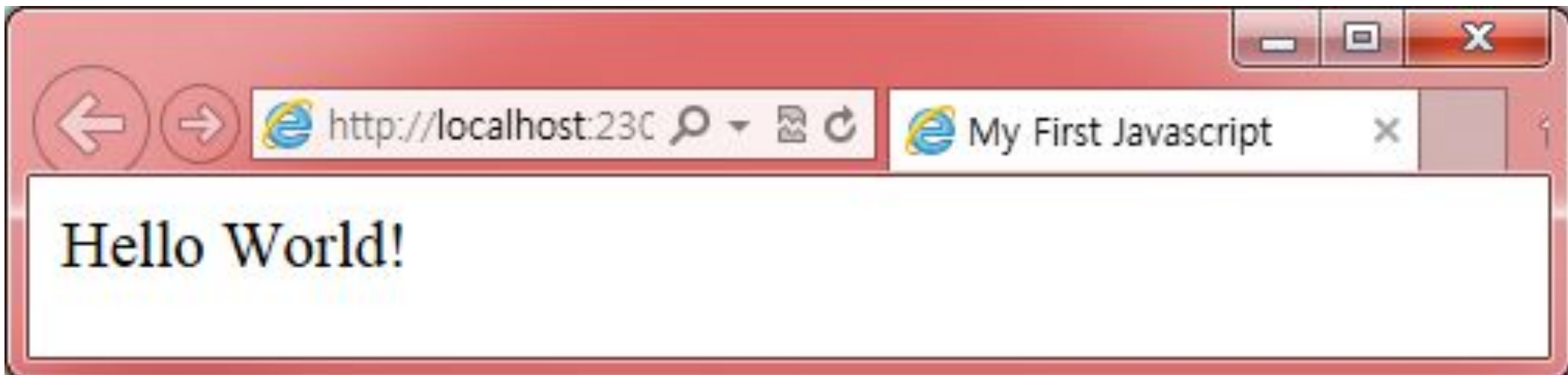


# 외부 자바 스크립트

```
<!DOCTYPE html>
<html>
<head>
  <script src="myscript.js"></script>
</head>
<body>
</body>
</html>
```

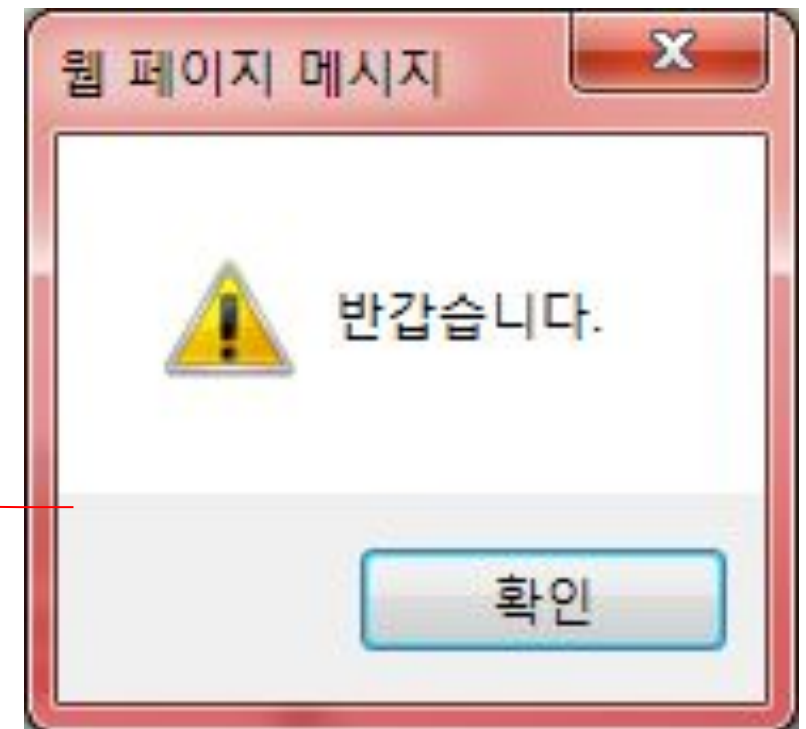
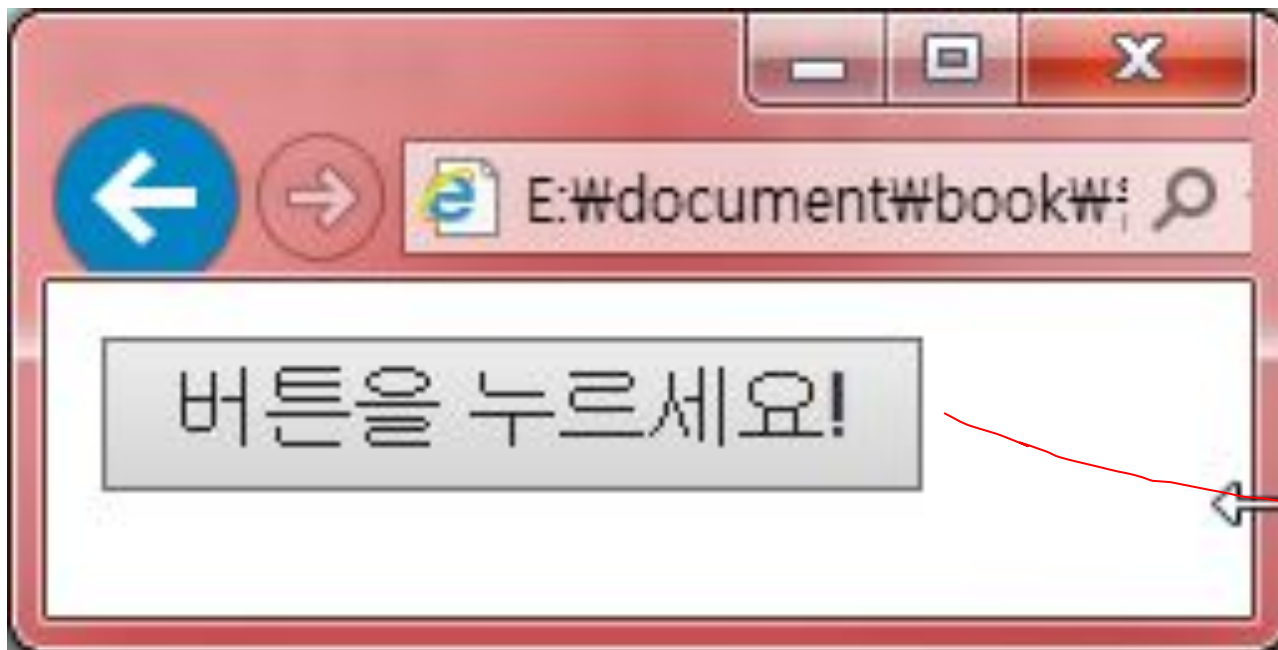
*myscript.js*

```
document.write("Hello World!");
```



# 인라인 자바 스크립트

```
<!DOCTYPE html>  
<html>  
<body>  
  <button type="button" onclick="alert('반갑습니다.')">버튼을  
  누르세요!</button>  
</body>  
</html>
```



# 문장

- 자바스크립트 문장(statement)들은 웹 브라우저에게 내리는 명령

1. 문서에 "Hello World!"를 추가하시오.
2. 화면에 경고창을 띄우시오.
3. 변수를 하나 만드시오.
4. ...



```
document.write("Hello World!");  
alert("warning!!");  
var count;  
...
```



순차적  
으로  
실행된  
다.

# 주석문

- // - 단일문장 주석

```
// id가 heading1인 헤딩요소를 찾아서 내용을 바꾼다.  
document.getElementById("heading1").innerHTML = "My HomePage";
```

- /\* \*/ - 다중 문장 주석

```
/*  
    이 코드는 웹 페이지의 헤딩의 내용을 변경한다.  
*/  
document.getElementById("heading1").innerHTML = "My HomePage";
```

# 변수

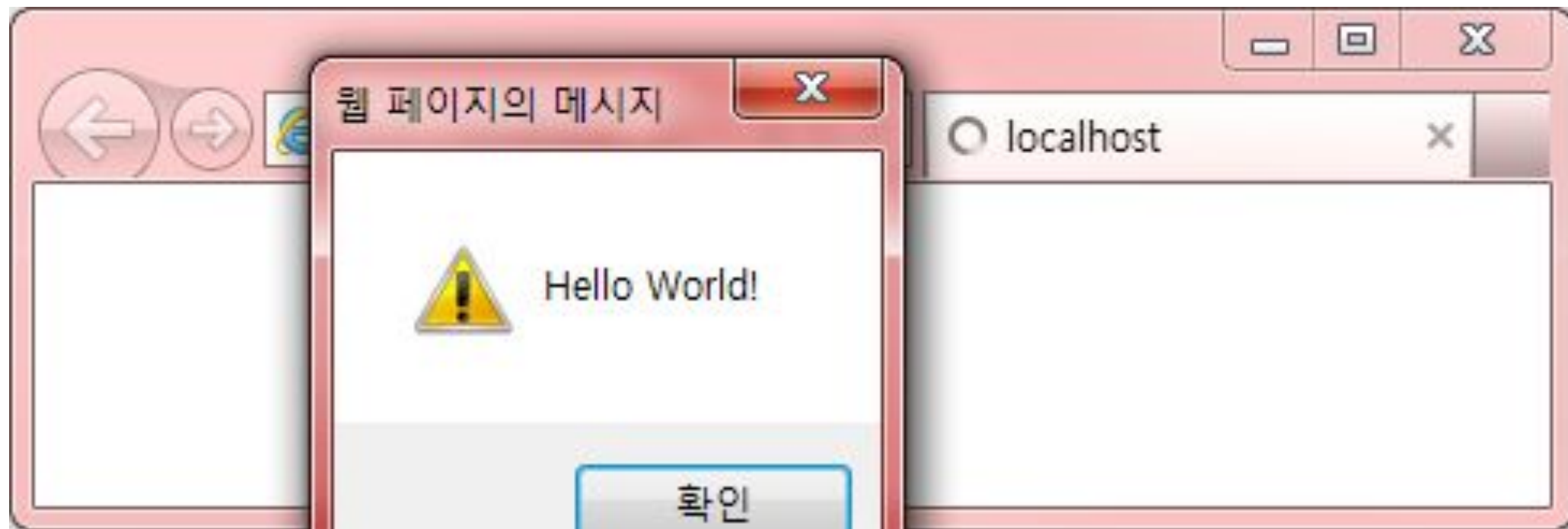
- **변수(variable)**는 데이터를 저장하는 상자
- **var** 키워드를 사용하여 선언(**declare**)한다.





# 예제

```
<script>  
  var x;  
  x = "Hello World!";  
  alert(x);  
</script>
```



# 변수 명명 규칙

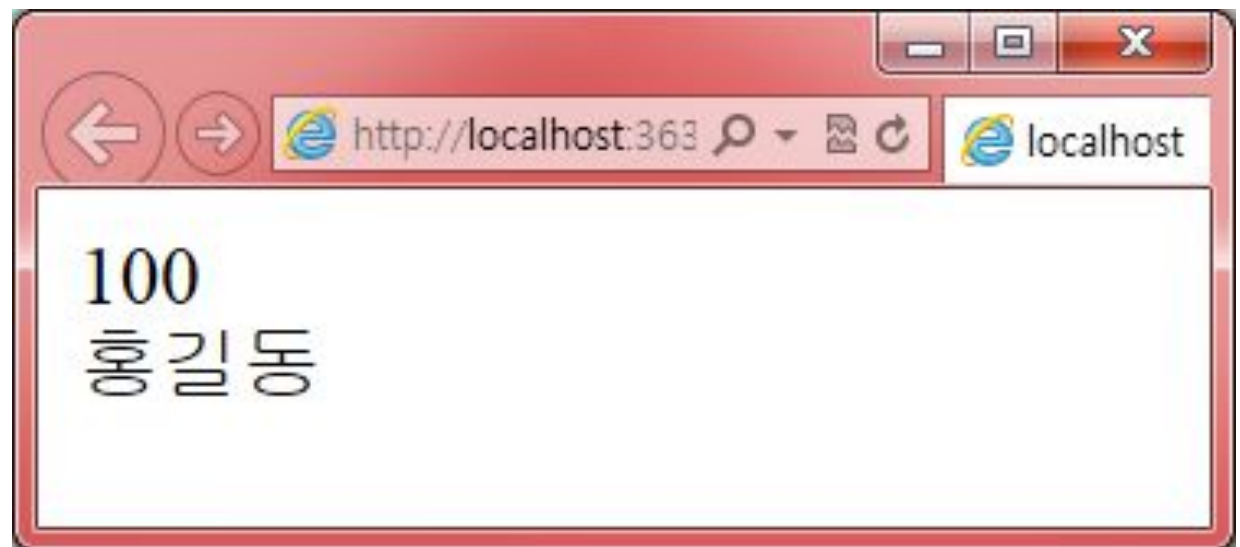
- 변수 이름은 문자로 시작해야 한다.(숫자로 시작하면 안된다.)
- 변수 이름은 \$나 \_로 시작할 수 있다.
- 변수 이름은 대소문자를 구별한다.(count와 Count는 서로 다른 변수이다.)
- 예약어는 변수명으로 사용할 수 없다.

# 자료형

- 수치형(number) - 정수나 실수
- 문자열(string) - 문자가 연결된 것, ""나 "로 표현
- 부울형(Boolean) - true 또는 false
- 객체형(object) - 객체를 나타내는 타입
- Undefined - 값이 정해지지 않은 상태

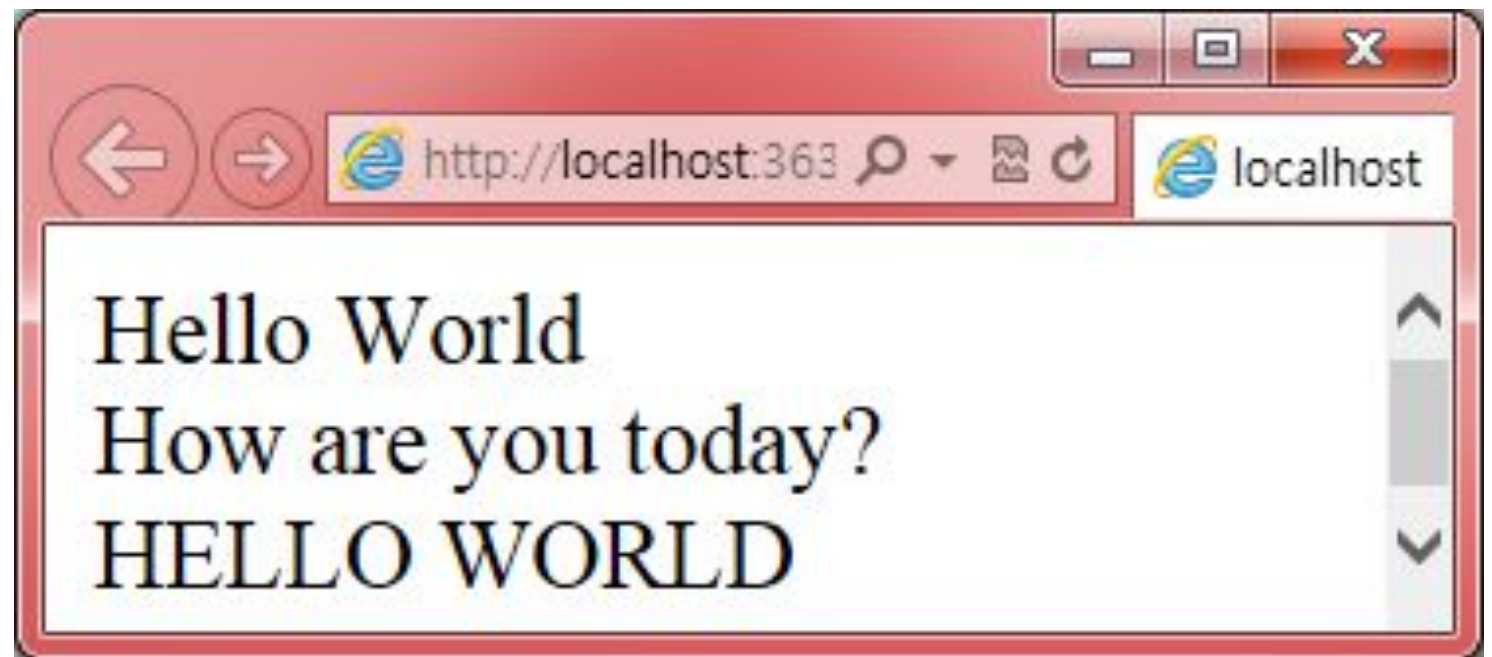
# 예제

```
<script>  
  var s;  
  
  s = 100;  
  document.write(s + "<br>");  
  
  s = "홍길동";  
  document.write(s + "<br>");  
</script>
```



# 예제

```
<script>  
  var s = "Hello World";  
  var t = "How are you" + " today?";  
  
  document.write(s + "<br>");  
  document.write(t + "<br>");  
  document.write(s.toUpperCase() + "<br>");  
</script>
```



# 객체형

- 객체(**object**)는 사물의 속성과 동작을 묶어서 표현하는 기법
- (예) 자동차는 메이커, 모델, 색상, 마력과 같은 속성도 있고 출발하기, 정지하기 등의 동작도 가지고 있다.

```
<script>
```

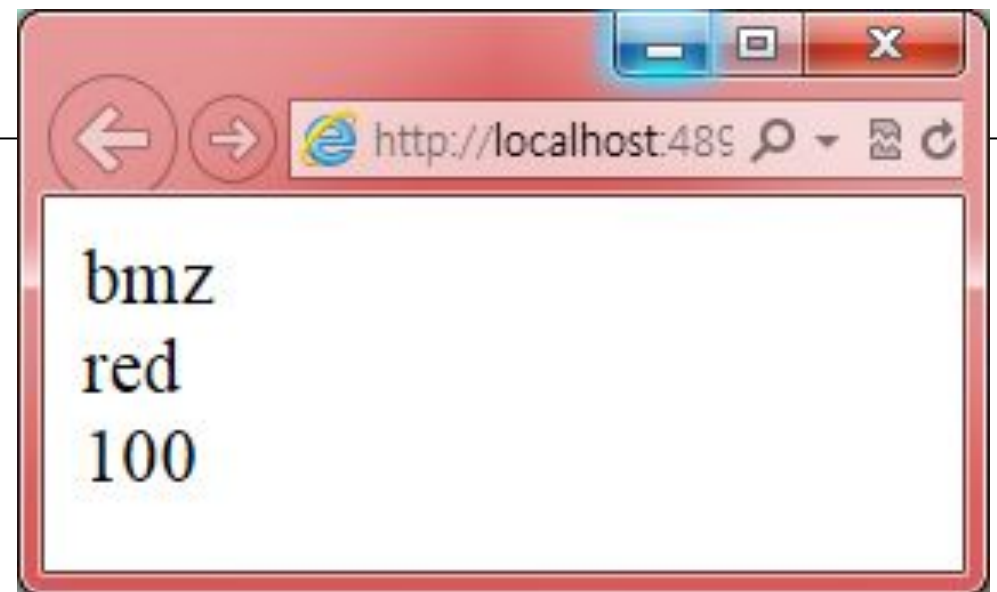
```
var myCar = {model:"bmw", color:"red", hp:100};
```

```
document.write(myCar.model + "<br>");
```

```
document.write(myCar.color + "<br>");
```

```
document.write(myCar.hp + "<br>");
```

```
</script>
```



# 산술 연산자

연산자	설명	예	수식의 값
+	덧셈	$x = 3 + 2$	
-	뺄셈	$x = 3 - 2$	
*	곱셈	$x = 3 * 2$	
/	나눗셈	$x = 3 / 2$	
%	나머지	$x = 3 \% 2$	
++	증가	$++x$	
--	감소	$--x$	



# 대입 연산자

- 변수에 값을 할당한다.
- 수식 ' $z = x + y$ '는  $x$ 값과  $y$ 값을 더한 값을  $z$ 에 대입한다는 의미이다.
- 대입연산자 "="는 산수에서의 같다라는 의미가 아니라 오른쪽에 있는 값을 왼쪽에 있는 변수에 저장하겠다는 의미를 갖는다.
- "같다"를 표현할 때는 "=="을 사용한다.

# 복합 대입 연산자

- 다음 표는  $x = 10, y = 5$ 라고 가정하고 대입연산이 어떻게 수행되는지를 설명한다.

연산자	예	동일한 수식	결과
$+=$	$x += y$	$x = x + y$	
$-=$	$x -= y$	$x = x - y$	
$*=$	$x *= y$	$x = x * y$	
$/=$	$x /= y$	$x = x / y$	
$\% =$	$x \% = y$	$x = x \% y$	

# 문자열에서의 '+' 연산자

- + 연산자는 문자열을 결합하는 용도로도 사용된다.
- 즉 + 연산자가 문자열에서 사용되면 문자열 결합의 의미가

```
s1 = "Welcom to ";  
s2 = "Javascript";  
s3 = s1 + s2;
```

- 숫자와 문자열을 + 연산자로 합하면 숫자를 문자열로

```
x = 1 + 1;  
y = "Car" + 1;  
document.write(x + "<br>");  
document.write(y + "<br>");
```

# 비교 연산자

- 논리문장에서 값들을 비교하는 용도로 사용
- 다음 표에서  $x$ 의 값은 1이라고 가정한다.

연산자	설명	예	결과
==	값이 같으면 true	$x == 1$	
		$x == 2$	
!=	값이 다르면 true	$x != 2$	
>	크면 true	$x > 2$	
<	작으면 true	$x < 2$	
>=	크거나 같으면 true	$x >= 2$	
<=	작거나 같으면 true	$x <= 2$	

# 비교 연산자

- 비교연산자는 다음과 같이 조건문에서 많이 사용된다. 아직 학습하지 않았지만 다음 문장의 의미를 추리하여 보자.

```
if (age > 18) {  
    msg = "입장하실 수 있습니다.";  
}
```

- 다음의 결과를 확인해보자.

```
<script>  
    var x = 10;  
    var y = 20;  
    document.write((x > y) + "<br>");  
    document.write((x < y) + "<br>");  
    document.write((x == y) + "<br>");  
    document.write((x != y) + "<br>");  
</script>
```

# 비교 연산자

- === 연산자와 !== 연산자

연산자	설명	예	결과
===	값과 타입이 모두 같으면 참	x === 1	
		x === "1"	
!==	값과 타입이 다르면 참	x !== 1	
		x !== "1"	

# 논리 연산자

- 여러 개의 조건을 조합하여 참인지 거짓인지를 따질 때 사용
- 예를 들어 "비가 오지 않고 휴일이면 테니스를 친다."라는 문장에는 "비가 오지 않는다" 라는 조건과 "휴일이다" 라는 조건이 동시에 만족이 되면 테니스를 친다는 의미가 포함되어 있다.

연산자	사용 예	의미
&&	x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
	x    y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
!	!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참



# 조건 연산자(삼항 연산자)

- $x > y$  가 참이면  $x$ 가 수식의 값이 된다.
- $x > y$  가 거짓이면  $y$ 가 수식의 값이 된다.

```
maxValue = (x > y) ? x : y;
```

# 연산자 우선순위

우선순위	연산자	우선순위	연산자
1	. [] new	10	&
2	()	11	^
3	++ --	12	
4	! ~ + -(부호) typeof void delete	13	&&
5	* / %	14	
6	+ -(사칙연산자)	15	?: (삼항연산자)
7	<< >> >>>	16	yield
8	< <= > >= in instanceof	17	= += -= *= /= %= <<= >>= >>>= &= ^=  =
9	== != === !==	18	,

# prompt() 함수

```
<script>
```

```
    var age = prompt("나이를 입력하세요", "만나이로 입력합니다.");
```

```
</script>
```



# 덧셈 예제1

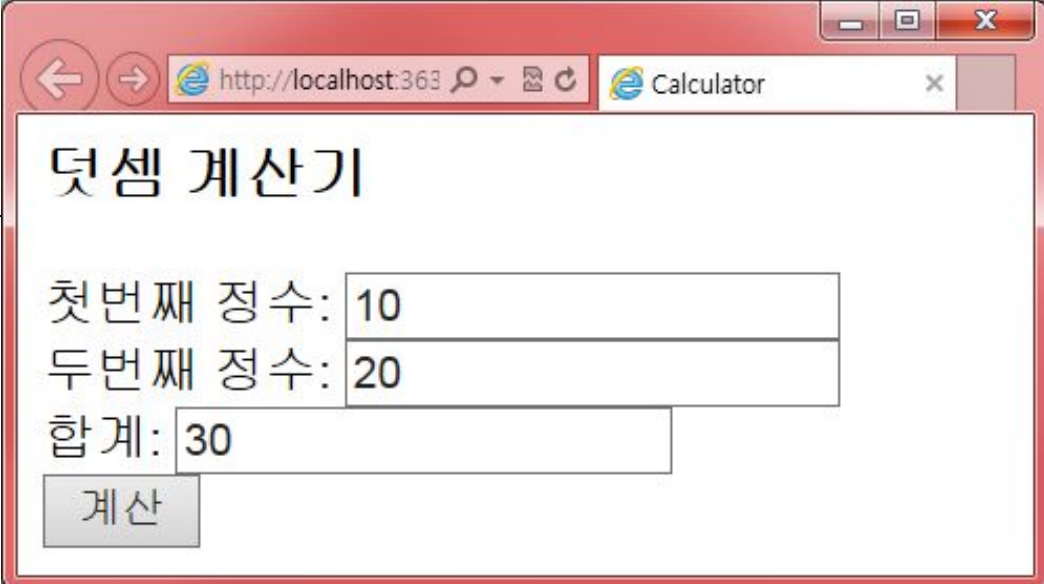
```
<script>  
  var x, y;  
  var input;  
  input = prompt("정수를 입력하십시오", "정수로");  
  x = parseInt(input);  
  input = prompt("정수를 입력하십시오", "정수로");  
  y = parseInt(input);  
  document.write(x + y + "<br>");  
</script>
```

## 덧셈 예제2

```
<html>
<head>
  <title>Calculator</title>
  <script>
    function calc() {
      var x = document.getElementById("x").value;
      var y = document.getElementById("y").value;
      var sum;
      sum = parseInt(x) + parseInt(y);
      document.getElementById("sum").value = sum;
    }
  </script>
</head>
```

## 덧셈 예제2

```
<body>
  <h3>덧셈 계산기</h3>
  <form name="myform" action="..." method="POST">
    첫번째 정수:
    <input id="x"><br>
    두번째 정수:
    <input id="y"><br>
    합계:
    <input id="sum"><br>
    <input type="button" value="계산" onclick="calc();">
  </form>
</body>
</html>
```



덧셈 계산기

첫번째 정수: 10

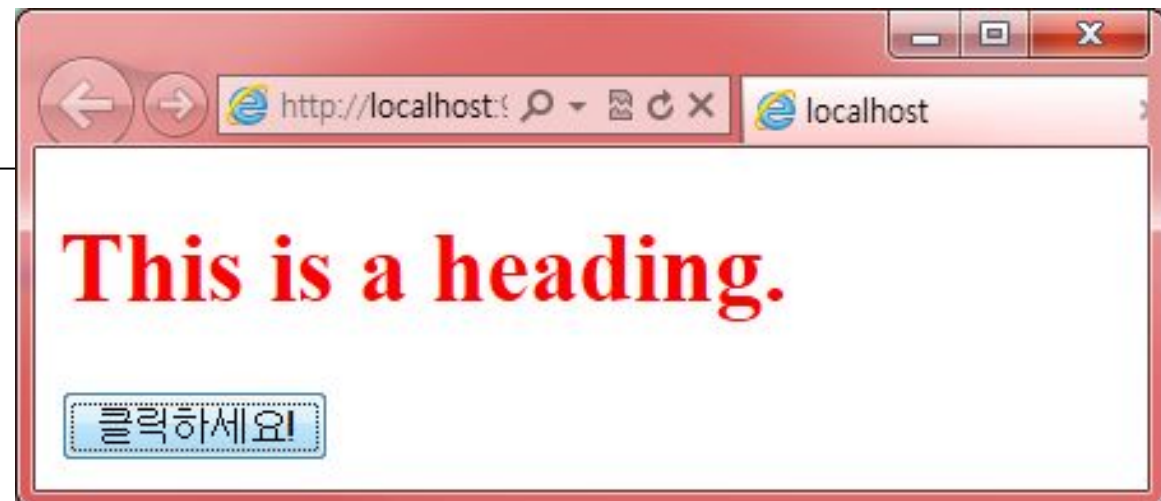
두번째 정수: 20

합계: 30

계산

# HTML 요소에 접근하기

```
<!DOCTYPE html>
<html>
<body>
  <h1 id="test">This is a heading.</h1>
  <script>
    function func() {
      e = document.getElementById("test");
      e.style.color = "red";
    }
  </script>
  <button type="button" onclick="func()">클릭하세요!</button>
</body>
</html>
```





# 제어문



제어문



조건문



반복문

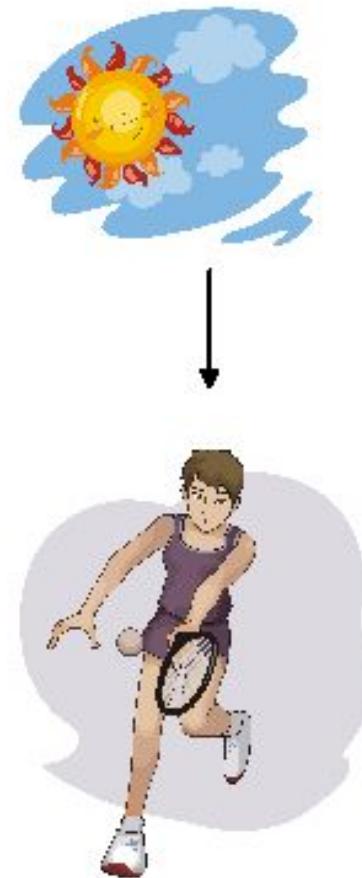
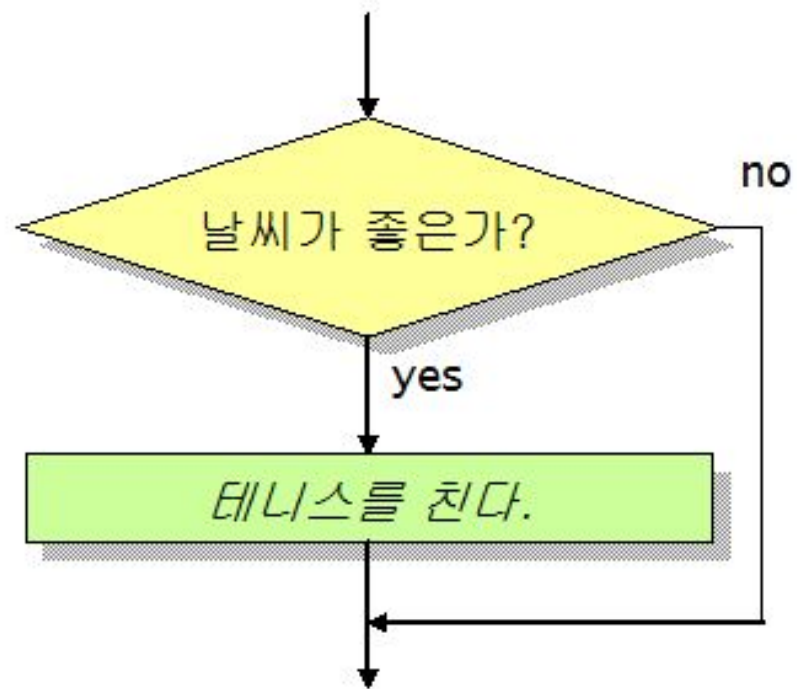
```
if( 연봉 > 2500 )  
    취업;  
else  
    고시 준비;
```

```
while(토플성적 < 800)  
    영어공부;
```

# 조건문의 종류

- if 문   if(조건문) 실행문;
- if else 문
- switch 문

# if 문



```
if (time < 12) {  
    greeting = "Good Morning!";  
}
```

# if-else 문

형식	<pre><b>if</b> (조건식) {     문장 1; } <b>else</b> {     문장 2; }</pre>
설명	만약 조건식이 참이면 문장1이 실행된다. 그렇지 않으면 문장2가 실행된다.

```
if (time < 12) {  
    msg = "Good Morning!";  
} else {  
    msg = "Good Afternoon!";  
}
```

# 연속적인 if 문

```
<script>
  var msg = "";
  var time = new Date().getHours();
  if (time < 12) {      // 12시 이전이면
    msg = "Good Morning";
  } else if (time < 18) { // 오후 6시 이전이면
    msg = "Good Afternoon";
  } else {             // 그렇지 않으면(오후 6시 이후이면)
    msg = "Good evening";
  }
  alert(msg);
</script>
```



# if 문제

- 숫자 2개와 연산자 1개를 입력 받아 연산자에 맞는 계산결과를 출력하는 프로그램을 작성하시오.

Explorer 사용자 프롬프트

스크립트 프롬프트:  
첫번째 숫자를 입력하세요

확인 취소

25

Explorer 사용자 프롬프트

스크립트 프롬프트: 결과  
연산자를 입력하세요  
(+, -, \*, /, % 중 하나)

확인 취소

+

Explorer 사용자 프롬프트

스크립트 프롬프트:  
두번째 숫자를 입력하세요

확인 취소

63

← → D:\html\조건문 문제 D:\html\조건문 문제

25+63 = 88

# switch 문

- if문과 비슷하게 조건에 따라 프로그램의 흐름을 분기시키기 위해 사용된다.
- if문의 경우 조건식이 참이냐 거짓이냐에 따라서 실행할 문장이 둘 중의 하나로 결정되기 때문에 연속적인 if문을 쓸 경우에는 **switch**문을 사용하는 것이 좋다.
- **switch**문은 제어식의 값에 따라 다음에 실행할 문장을 결정하게 된다.

형식

```
switch(제어식) {  
    case c1:  
        문장1;  
        break;  
    case c2:  
        문장2;  
        break;  
    default:  
        문장d;  
        break;  
}
```

# switch 문

```
<script>
```

```
var grade = prompt("성적을 입력하시오:", "A-F사이의 문자로");
```

```
switch (grade) {
```

```
  case 'A': alert("잘했어요!");
```

```
    break;
```

```
  case 'B': alert("좋은 점수군요");
```

```
    break;
```

```
  case 'C': alert("괜찮은 점수군요");
```

```
    break;
```

```
  case 'D': alert("좀더 노력하세요");
```

```
    break;
```

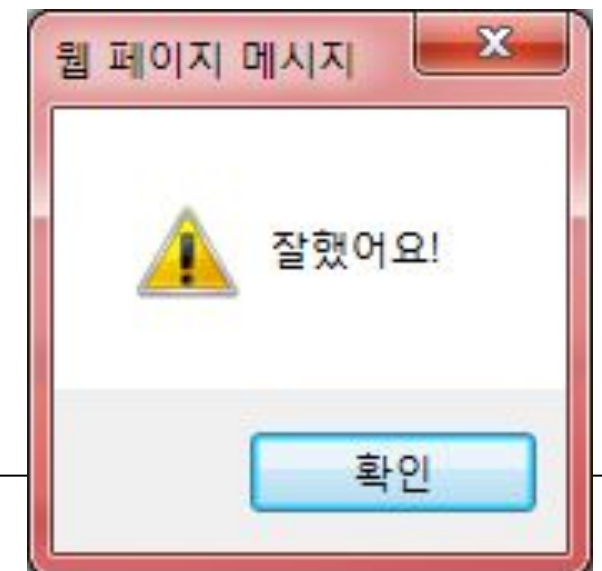
```
  case 'F': alert("다음학기 수강하세요");
```

```
    break;
```

```
  default: alert("알수없는 학점입니다.")
```

```
}
```

```
</script>
```





# switch 문제

- 점수를 입력받아 학점을 출력하시오.(switch문을 이용)
  - 점수가 90 ~ 100이면 'A'
  - 점수가 80 ~ 89이면 'B'
  - 점수가 70 ~ 79이면 'C'
  - 점수가 60 ~ 69이면 'D'
  - 점수가 0 ~ 59이면 'F'
  - 출력은 document.write()를 이용

# 문제 1

- 두 사람의 가위, 바위, 보를 입력 받아 승자를 출력하는 프로그램을 작성하시오.

The image shows a web application interface for a Rock-Paper-Scissors game. It consists of two input prompts and a result display.

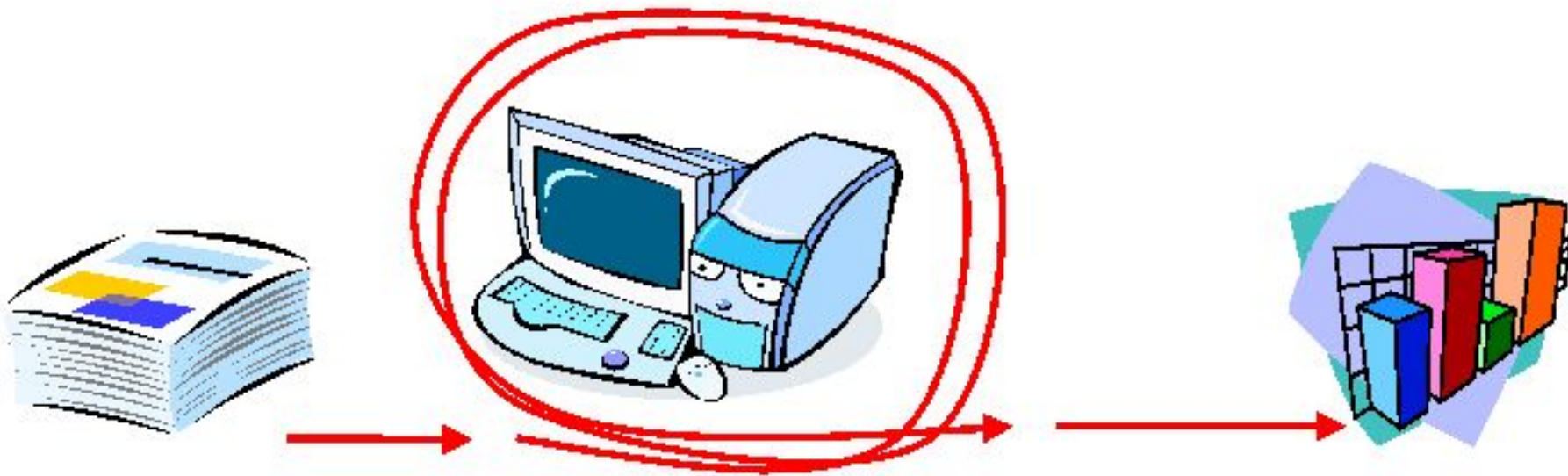
**Input Prompt 1:** Explorer 사용자 프롬프트. 스크립트 프롬프트: 첫번째 사람의 가위 바위 보를 입력하세요. Input field: 가위. Buttons: 확인, 취소.

**Input Prompt 2:** Explorer 사용자 프롬프트. 스크립트 프롬프트: 두번째 사람의 가위 바위 보를 입력하세요. Input field: 바위. Buttons: 확인, 취소.

**Result Display:** 첫번째 사람 : 가위  
두번째 사람 : 바위  
결과 : 두번째 사람 승!!!

# 반복문

- 같은 처리 과정을 여러 번 되풀이하는 것



# 반복문의 종류

- **while** – 지정된 조건이 참이면 반복 실행한다.
- **for** – 주로 정해진 횟수 동안 코드를 반복 실행한다.

# while 문

```
var i = 0;
```

```
while( i < 10 ) ← 반복 조건, 변수 i가 10보다 작으면 반복 계속
```

```
{
```

```
    document.write(i+" <br>"); ← 반복되는 문장, i를 화면에 출력한다.
```

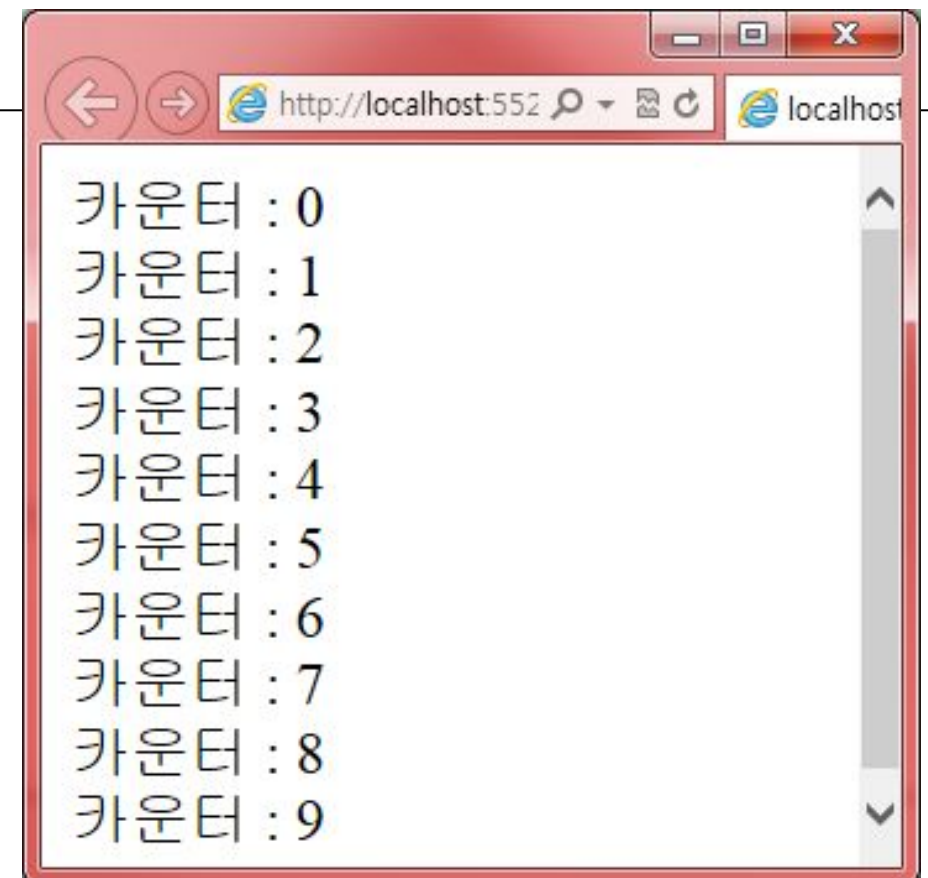
```
    i++; ← 한 번의 반복마다 i를 하나 증가시킨다.
```

```
}
```

이 문장이 없으면 무한히 반복한다.

# while 문

```
<script>
  var i = 0;
  while (i < 10) {
    document.write("카운터 : " + i + "<br />");
    i++;
  }
</script>
```

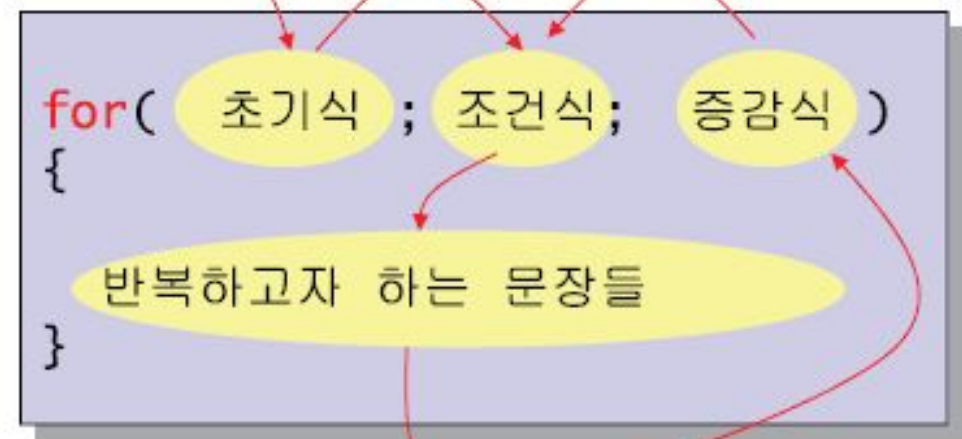


# for 문

초기식      조건식      증감식

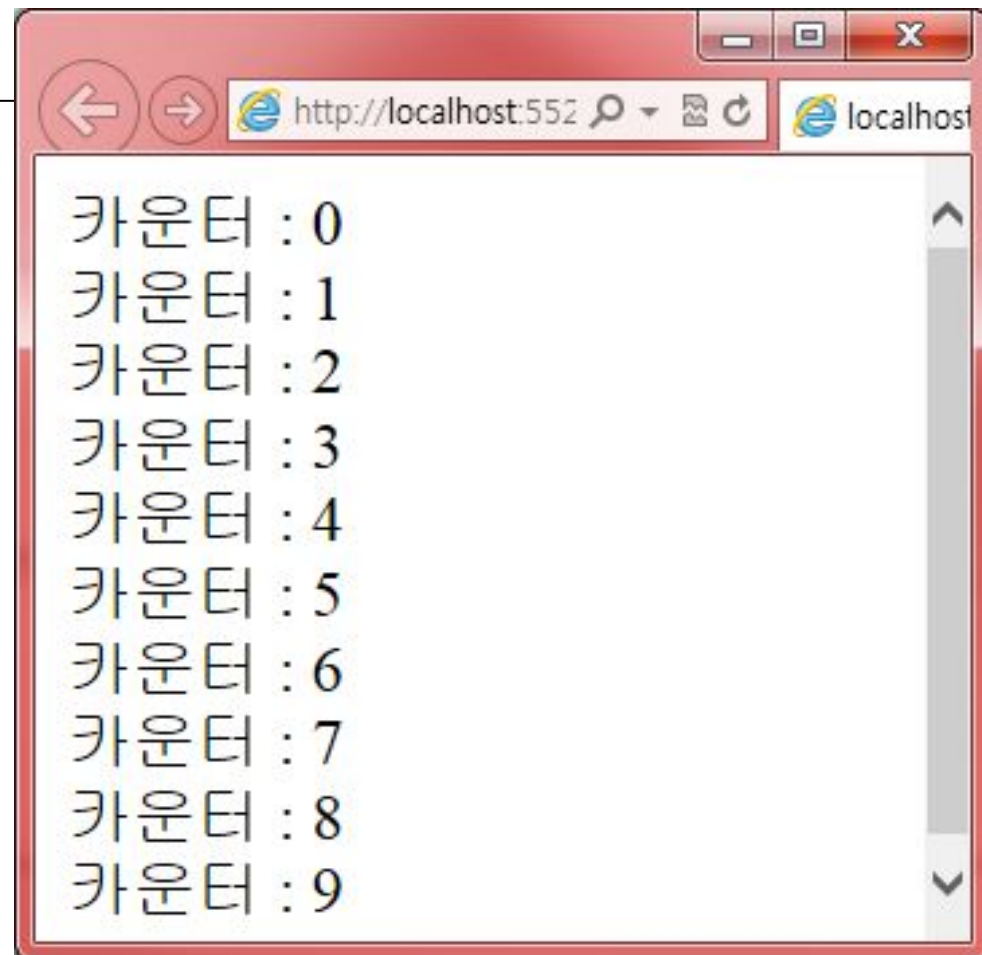
```
for( i=0 ; i<10 ; i++ )  
{  
    document.write(i+"<br />");  
}
```

반복되는 문장



# for 문

```
<script>  
  for (var i = 0; i < 10; i++) {  
    document.write("카운터 : " + i + "<br />");  
  }  
</script>
```





# 중첩 반복문

- 하나의 **for**문 안에 다른 **for**문이 내장될 수 있다.
- 반복문이 중첩될 때는 반복문 제어 변수로 서로 다른 변수를 사용해야 한다.

# 중첩 반복문 예제

```
<style>
  table, td {border:1px solid black;}
</style>
<script>
  document.write("<h1>구구단표</h1>");
  document.write("<table>");
  for (var i = 1; i <= 9; i++) {
    document.write("<tr>");
    document.write("<td>" + i + "</td>");
    for (var j = 2; j <= 9; j++) {
      document.write("<td>" + i * j + "</td>");
    }
    document.write("</tr>");
  }
  document.write("</table>");
</script>
```



1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

# do/while문

- while문과 비슷하나 반복 조건을 처음이 아니라 끝에서 검사한다는 점이 다르다.
- do/while문은 일단 문장을 한 번 실행하고 나서 조건을 검사하고 싶을 때 사용한다.

```
<script>  
  var i = 0;  
  do {  
    document.write("카운터 : " + i + "<br>");  
    i++;  
  } while (i < 10);  
</script>
```

# for/in 반복문

- 객체 안의 속성들에 대하여 어떤 처리를 반복할 수 있는 구조
- for/in 반복문을 이용하면 객체 안의 모든 속성에 대하여 어떤 연산을 실행할 수 있다.

```
<script>  
  var myCar = { make: "BMW", model: "X5", year: 2013 };  
  var txt = "";  
  for (var x in myCar) {  
    txt += myCar[x] + " ";  
  }  
  document.write(txt);  
</script>
```



# break 문장

- 반복문을 벗어나기 위해 사용
- 반복문 안에서 **break** 문이 실행되면 반복문을 빠져나오게 된다.

```
<script>
  var msg = "";
  for (var i = 0; i < 10; i++) {
    if (i == 3) {
      break;
    }
    msg += i + "<br>";
  }
  document.write(msg);
</script>
```

# continue 문장

- 현재 실행하고 있는 반복 과정의 나머지를 생략하고 다음 반복문을 시작하게 만든다.
- 예를 들어 0부터 10까지의 정수 중에서 3만 제외하고 출력하는 예제를 보면 0부터 10까지 정수를 하나씩 조사하다가 현재 정수가 3이면 **continue**를 실행해서 현재 반복을 중지하고 다음 반복을 시작한다.

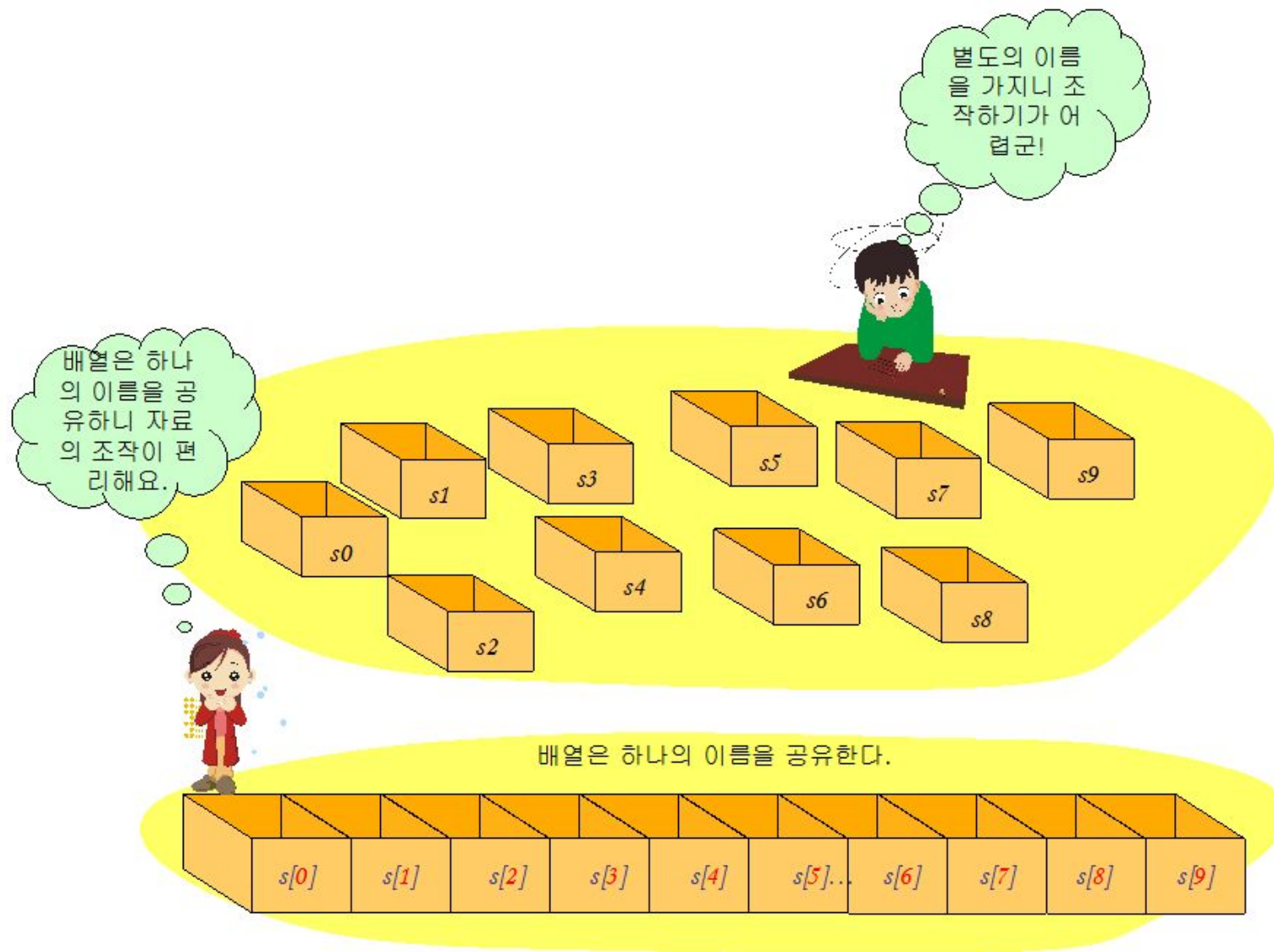
```
<script>
  var msg = "";
  for (var i = 0; i < 10; i++) {
    if (i == 3) {
      continue;
    }
    msg += i + "<br>";
  }
  document.write(msg);
</script>
```

# 문제

1. 1부터 10까지의 합을 구하는 프로그램을 작성하시오.
2. 1부터 200까지의 짝수의 합을 구하는 프로그램을 작성하시오.(continue를 이용)
3. 사용자가 입력한 값을 계속 더하고, 사용자가 0을 입력하면 그때까지 누적된 값을 출력하는 프로그램을 작성하시오.
4. 다중 for문을 이용해서 1~ 10 까지 중  
i와 k의 더한 합이 3의 배수일때만 출력 continue를 이용
5. 1~100 까지 중 2의 배수이면서 3의 배수인것만 출력
6. 두 수를 입력(prompt) 두수의 합이 100이상일때만 출력  
continue를 이용 , 두수 모두 0 이 입력되면 종료

# 배열

- 많은 값을 저장할 수 있는 공간이 필요할 때 배열을 사용한다.
- 서로 관련된 데이터를 차례로 접근하여서 처리할 수 있다.





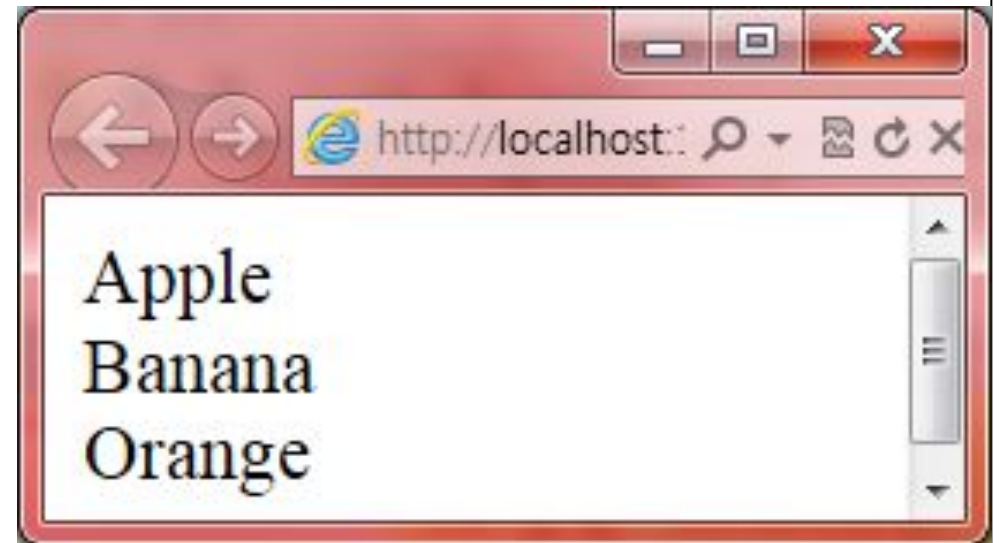
# 배열을 생성하는 2가지 방법

- 리터럴로 배열 생성
  - `var fruits = ["apple", "banana", "peach"];`
- Array 객체로 배열 생성
  - `var fruits = new Array("apple","banana","orange");`
- `var fruits = new Array();`
- `fruits[0] = "Apple";`
- `fruits[1] = "Banana";`
- `fruits[2] = "Orange";`

# 예제

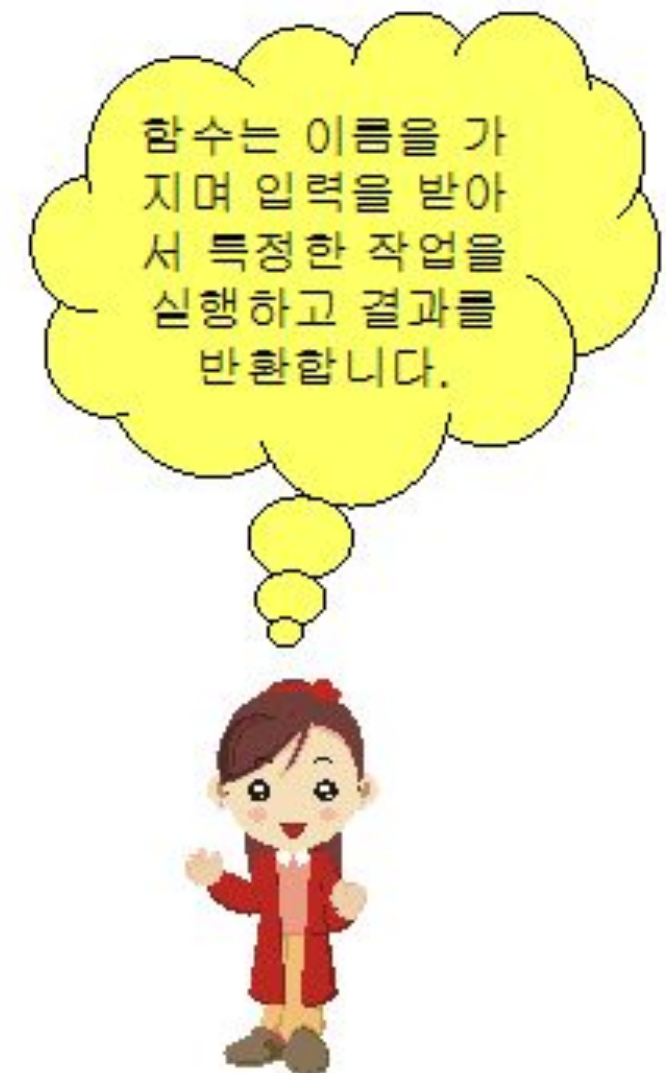
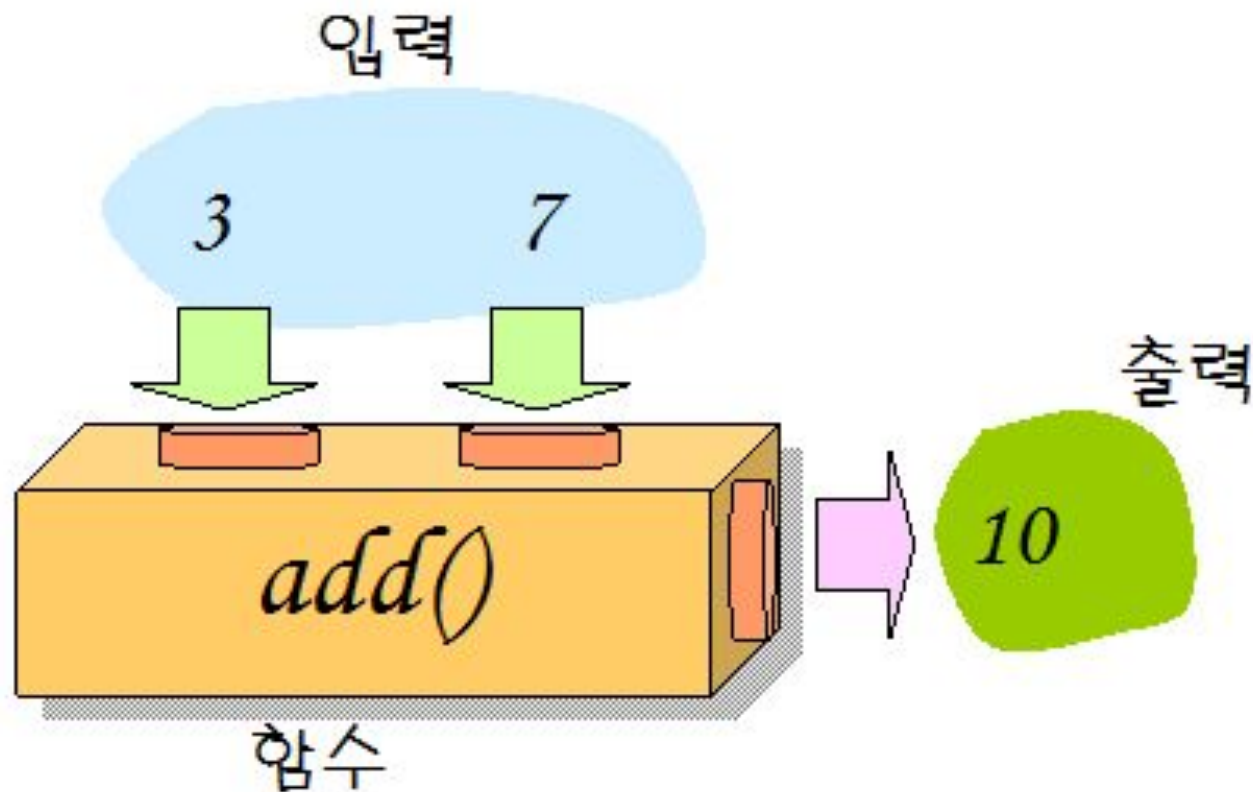
```
<!DOCTYPE html>
<html>
<body>
<script>
  var i;
  var fruits = new Array();
  fruits[0] = "Apple";
  fruits[1] = "Banana";
  fruits[2] = "Orange";

  for (i = 0; i < fruits.length; i++) {
    document.write(fruits[i] + "<br>");
  }
  for(x in fruits ){
    document.write(fruits[x] + "<br>");
  }
</script>
</body>
</html>
```



# 함수

- 함수는 입력을 받아서 특정한 작업을 수행하여서 결과를 반환하는 블랙 박스



# 함수 만들기

1. 파라미터도 있고 반환 값도 있는 함수

```
function 함수명(파라미터1, 파라미터2, ...) {  
    return 반환값;  
}
```

2. 파라미터는 있고 반환 값은 없는 함수

```
function 함수명(파라미터1, 파라미터2, ...) {  
    명령문;  
}
```

3. 파라미터는 없고 반환 값은 있는 함수

```
function 함수명() {  
    return 반환값;  
}
```

4. 파라미터도 없고 반환 값도 없는 함수

```
function 함수명() {  
    명령문;  
}
```

# 함수의 호출

- 함수는 호출에 의해서 실행

```
function showDialog(para1, para2) {  
  명령문1;  
  명령문2;  
}  
  
showDialog(arg1, arg2);
```

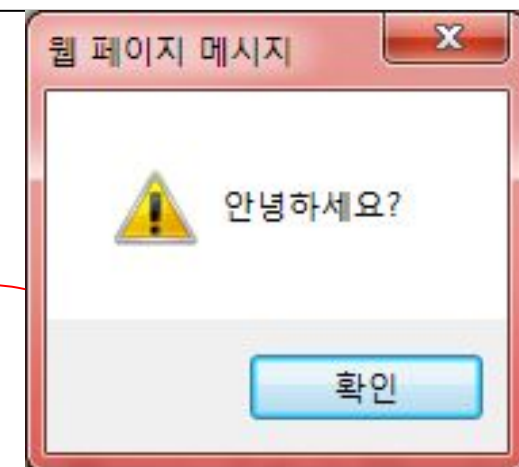
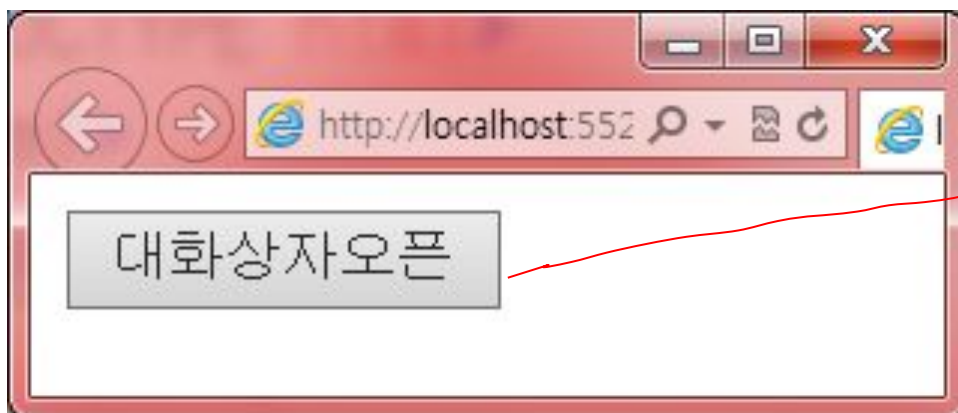
매개변수

인수

- 인수(argument) : 함수를 호출할 때는 어떤 값을 함수로 전달하는 값
- 인수는 데이터 타입이 없을 뿐만 아니라 개수에도 제약이 없다.
- 실 인수가 남으면 무시되고, 모자라는 인수는 undefined가 된다.
- 매개변수 (parameter) : 함수를 만들 때 인수로 받을 변수를 선언하는 것

# 예제

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function showDialog() {
      alert("안녕하세요?");
    }
  </script>
</head>
<body>
  <button onclick="showDialog()">대화상자오픈</button>
</body>
</html>
```



# 함수 예제

```
<body>
  <form>
    첫째 :<input type="text" id="x"><br>
    둘째 :<input type="text" id="y"><br>
    결과 :<input type="text" id="sum"><br>
    <input type="button"
      onclick="calc()" value="확인">
    <br>
    <p>첫째 값 :<span id="sp1"></span> </p>
    <p>둘째 값 :<span id="sp2"></span> </p>
    <p>결과 :<span id="sp3"></span> </p>

  </form>
</body>
```

# 함수 예제

```
<script>
  function calc(){

    //value : 입력받는 html의 <input>태그에서 값을
    //가져오거나 대입(출력)할때 사용
    var a = document.getElementById('x').value;
    var b = document.getElementById('y').value;
    var res = parseInt(a) + parseInt(b);

    document.getElementById('sum').value = res;

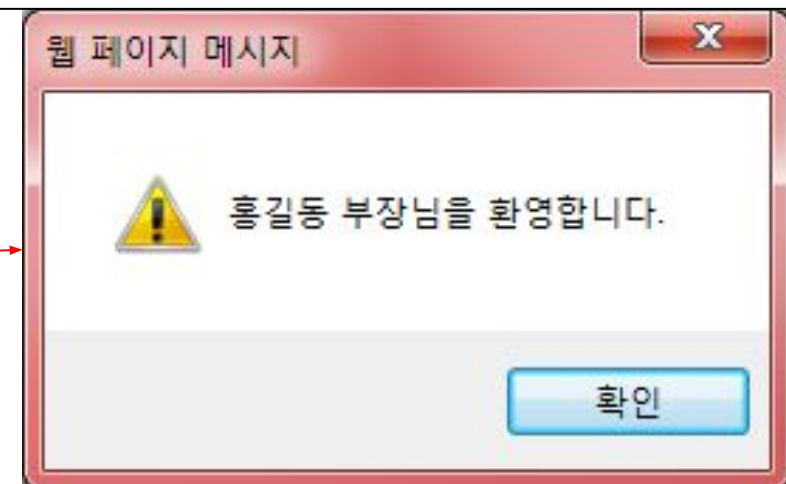
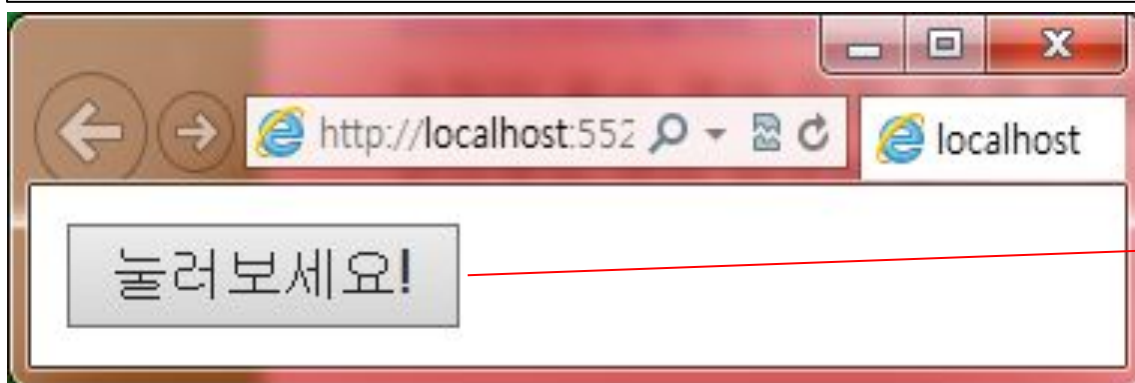
    //////////////////////////////////////
    //innerHTML - > 입력태그가 아닌 다른 태그에 출력
    document.getElementById('sp1').innerHTML = a;
    document.getElementById('sp2').innerHTML = b;
    document.getElementById('sp3').innerHTML = res;

  }
</script>
```



# 인수와 매개 변수

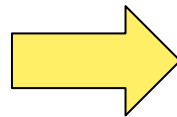
```
<!DOCTYPE html>
<html>
<head>
  <script>
    function greeting(name, position) {
      alert(name + " " + position + "님을 환영합니다.");
    }
  </script>
</head>
<body>
  <button onclick="greeting('홍길동', '부장')">눌러보세요!</button>
</body>
</html>
```



# 무명 함수

- 함수를 만들어서 한번만 사용할 때 이름을 주지 않고 한번만 사용하는 경우 무명함수(anonymous function)라고 한다.

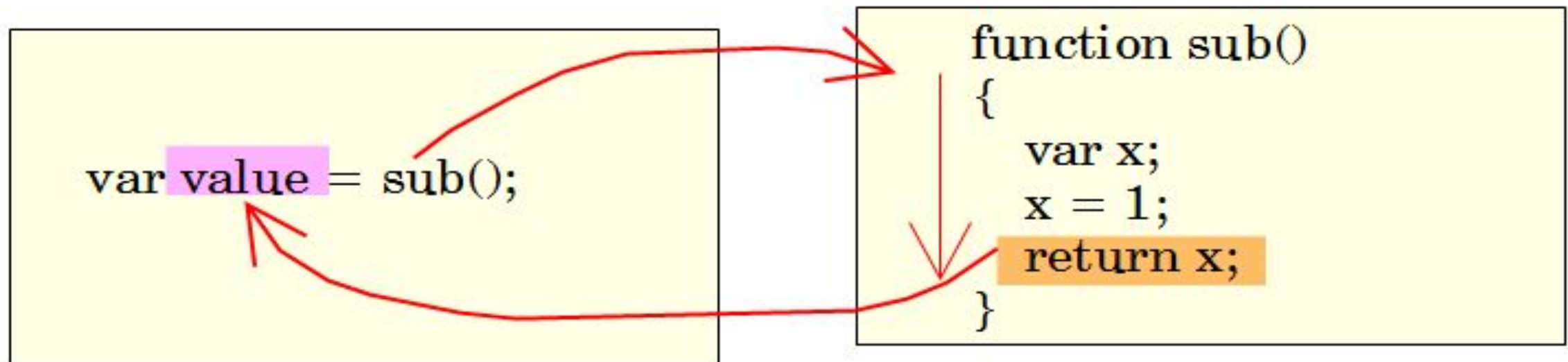
```
function showDialog(str) {  
    alert(str);  
}  
showDialog("안녕하세요.");
```



```
// 무명함수의 실행  
(function (str) {  
    alert(str);  
})("안녕하세요.");
```

# 함수의 반환값

- `return` 문장을 사용하여 외부로 값을 반환



- 반환된 값을 어디에 저장하기 않고 바로 수식에 사용해도 된다.
- `window.onload = function(){`

- `document.getElementById("para1").innerHTML = sub();`

# 함수 반환값

```
<script>
function sub(a,b){
    return a+b;
}
window.onload = function(){
    //var res = sub(4,5);
    //document.getElementById("aa").innerHTML = res;
    document.getElementById("aa").innerHTML =sub(4,5);
}
</script>
<body>
    <p id="aa"></p>
</body>
```

# 예제

```
<style>
  div{
    background : yellow;
    border : 1px solid red;
    width : 300;
    height : 500;
  }
</style>
<body>
  <input type="button" value="시작"
    onclick="randProc()"> <br>
  <hr color='blue'>
  <!-- 시작 버튼 누르면 랜덤수 5개 를 출력 -->
  <h1>출력위치 </h1>
  <div id="res"></div>
</body>
```



# 함수의 반환값

- 단순히 함수를 종료하고 싶은 경우에도 사용할 수 있다.

```
function divide(a, b) {  
  if (b == 0) {  
    return;  
  }  
  return a / b;  
}
```

만약  
파면가  
0이면  
나눗셈  
에  
오류  
가  
발생  
함.  
이  
때  
이  
함  
수  
를  
종  
료  
하  
고  
되  
는  
것  
이  
다.

# 지역변수

- 함수 안에서 선언된 변수
- 함수 안에서만 사용 가능
- 다른 함수에서도 똑같은 이름으로 선언이 가능함
- 지역변수는 함수가 종료되면 자동적으로 소멸된다.

```
function add(a, b) {  
  var sum = 0;  
  
  sum = a + b;  
  return sum;  
}
```

다른 함수에서는 **sum**을 사용할 수 없다

# 지역변수

```
function add(a, b) {  
    var sum = a + b;  
}  
function sub(a,b){  
    var res = a-b;  
}  
window.onload= function() {  
    add(4,5);  
    document.write("add=" + sum); //오류 , 반환값을 이용  
    sub(10, 4);  
    document.write("sub=" + res); //오류 , 반환값을 이용  
}
```



# 전역변수

- 함수 외부에서 선언된 변수
- 웹 페이지 상의 모든 스크립트와 모든 함수는 전역변수를 사용할 수 있다.
- 전역변수는 사용자가 웹페이지를 닫으면 소멸된다.

```
var sum = 0;
```

```
function add(a, b) {
```

```
    sum = a + b;
```

```
}
```

# 전역변수

```
<script>
var sum = 0;
function add(a, b) {
    sum = a + b;
}
function sub(a,b){
    sum = a-b;
}
window.onload= function() {
    add(4,5);
    document.write("add=" + sum);
    sub(10, 4);
    document.write("sub=" + sum);
}
</script>
```

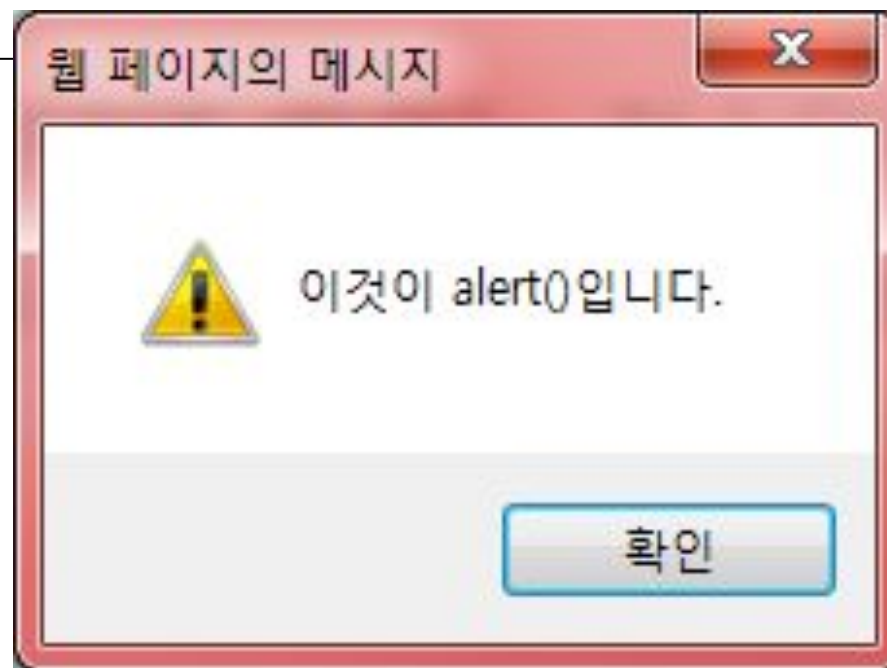
# 전역변수

- 선언되지 않은 변수에 값을 대입하면 그 변수는 자동적으로 전역변수가 된다.
- 예를 들면 다음과 같은 문장은 함수 안에서 실행되더라도 변수 **userName**을 전역변수로 선언하는 것이나 마찬가지이다.

```
function add(a, b) {  
    userName = "초파";    sum = a + b;  
    document.write(userName);  
}  
function sub(a,b){  
    userName = "나초";    sum = a- b;  
    document.write(userName);  
}  
window.onload= function() {  
    add(4,5);  
    document.write("add=" + sum);  
    sub(10, 4);  
    document.write("sub=" + sum);  
}
```

# alert() 함수

```
<script>  
    alert("이것이 alert()입니다.");  
</script>
```

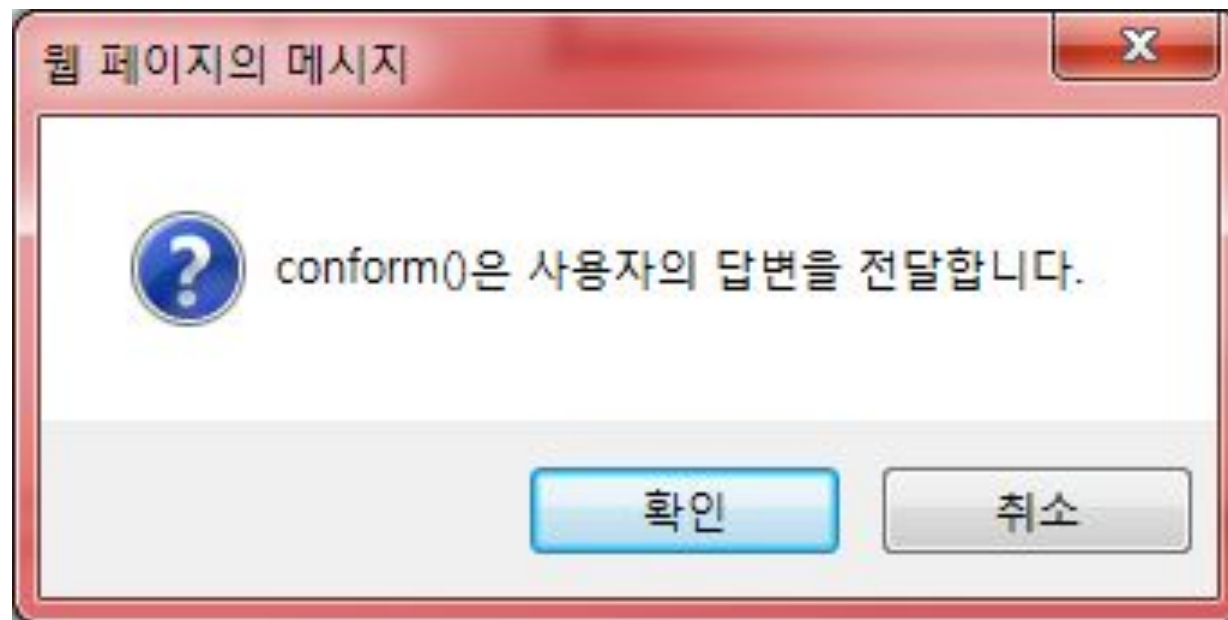


# confirm() 함수

```
<script>
```

```
    var user = confirm("confirm()은 사용자의 답변을 전달합니다.");
```

```
</script>
```



# prompt() 함수

```
<script>
```

```
    var age = prompt("나이를 입력하세요", "만나이로 입력합니다.");
```

```
</script>
```

