



웹 프로그래밍

7. JavaScript 기초 (1)

Contents



- 주석
- 변수
- 기본 자료형 (primitive values)
- 기본 연산자 (operators)
- 제어문
- 함수

주석



```
// single-line comment
```

```
/* multi-line comment */
```

- HTML: `<!-- comment -->`
- CSS/JS/PHP: `/* comment */`
- Java/JS/PHP: `// comment`

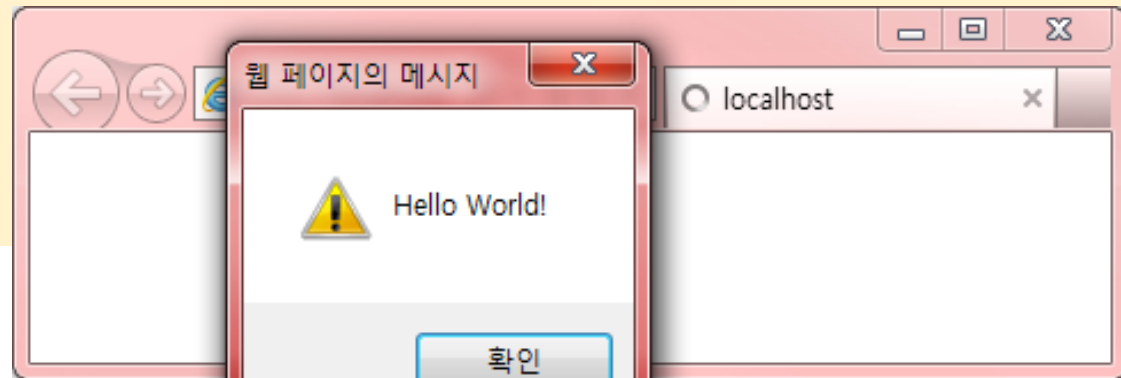
변수

- **변수(variable)**는 데이터를 저장하는 상자
 - 문자, 숫자, _, \$ 사용 가능
 - 반드시 문자로 시작
 - Unique name 사용
 - Case-sensitive
 - Reserved words 사용할 수 없음
- 변수 선언(declare): **var** 키워드로 선언



변수

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



```
var person = "John Doe", carName = "Volvo", price = 200;
```

```
var person = "John Doe",  
    carName = "Volvo",  
    price = 200;
```

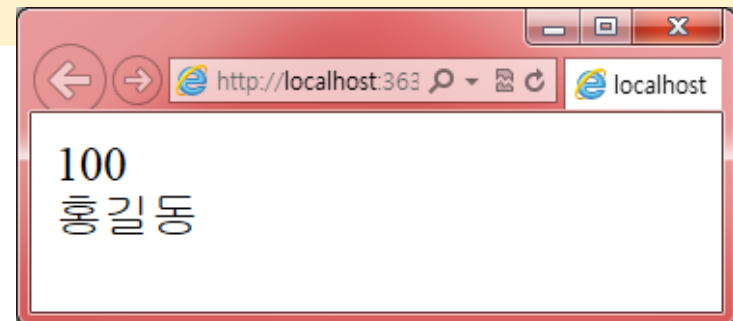
변수



- 변수의 자료형
 - 할당된 값에 따라서 자동으로 자료형 결정
- 자료형
 - 기본 자료형 (primitive values)
 - 변경 불가능한 (immutable) 상수 값
 - 수치형(number)
 - 문자열(string)
 - 부울형(boolean)
 - true / false
 - 특수 값
 - null / undefined
 - 객체형(object)
 - Math, Data, Array 등 내장 객체
 - 사용자 정의 객체

예제

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



Number

- 64비트 IEEE 754 표준 방식 하나만 존재
 - 정수와 실수의 차이 없음

```
var enrollment = 99;  
var medianGrade = 2.8;  
var credits = 5 + 4 + (2 * 3);
```

- integers and real numbers are the same type (no `int` vs. `double`)
- same operators: `+` `-` `*` `/` `%` `++` `--` `=` `+=` `-=` `*=` `/=` `%=`
- similar precedence to Java
- many operators auto-convert types: `"2" * 3` is 6

Number property & methods

● Property & Methods

- 객체 (object)에 사용되는 개념
- Primitive value 자체는 속성(property) 또는 메소드 (method)를 가질 수 없음
- JS는 **primitive values**를 **object**처럼 취급하는 것을 허용
 - 예를 들어, 숫자 상수를 JS 내장 객체인 Number로 변환(wrapper object)하여 Number 객체의 property와 method를 상속받아 사용 가능

```
var x = 1;
var y = new Number(1);
typeof x; // "number"
typeof y; // "object"
x == y // true
x === y // false
```

- 문자열에 대해서도 동일하게 적용
 - 문자열 상수도 String 객체의 속성 및 메소드 사용 가능
 - But, **number, string, boolean 상수를 객체로 선언하지 말 것!**

Number Property & Methods

Number Properties

Property	Description
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
NaN	Represents a "Not-a-Number" value
POSITIVE_INFINITY	Represents infinity (returned on overflow)

Number Methods

- `toString()` // 문자열로 반환
- `toExponential()` // 지수 표기법으로 표현된 문자열
- `toFixed()` // 명시된 자릿수 만큼 소수점 이하 표현된 문자열
- `toPrecision()` // 명시된 숫자 만큼의 길이로 표현된 문자열
- `valueOf()` // 숫자로 반환
- Global methods

Method	Description
<code>Number()</code>	Returns a number, converted from its argument.
<code>parseFloat()</code>	Parses its argument and returns a floating point number
<code>parseInt()</code>	Parses its argument and returns an integer

String

● 문자열 상수

- ‘‘ 또는 “”로 둘러싸인 문자 집합
- 16비트 부호 없는 정수 값 요소들의 집합
- 변경 불가능(Immutable)
 - 한 번 생성된 문자열 상수는 변경 불가능 (cf. 문자열 in C)
- 문자열 상수의 각 문자에 접근

```
var firstLetter = s[0];           // fails in IE
var firstLetter = s.charAt(0);    // does work in IE
var lastLetter = s.charAt(s.length - 1);
```

- Escape sequence 사용 가능 (JAVA 와 동일)
 - \', \", \&, \n, \t, \\\

```
var x = "We are the so-called \"Vikings\" from the north.";
```

```
var x = "We are the so-called \\\"Vikings\\\" from the north.";
```

String property & methods

String property

- length: 문자열 길이

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
var sln = txt.length;
```

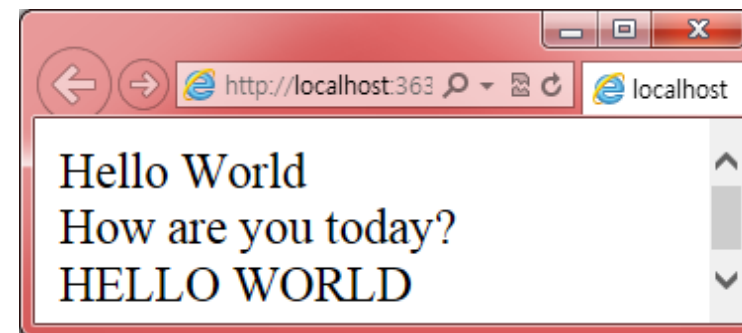
String methods

- 대소문자 변환: toUpperCase(), toLowerCase()
- 문자열 붙이기: concat()
- 특정 문자 추출: charAt(), charCodeAt()
- 문자열 내 문자열 찾기: indexOf(), lastIndexOf(), search()
- 부분 문자열 추출하기: slice(), substring(), substr()
- 문자열 대체: replace()
 - 문자열 상수 자체를 바꾸는 것이 아니라, 바뀐 새로운 문자열 반환
- 배열로 변환: split()

String

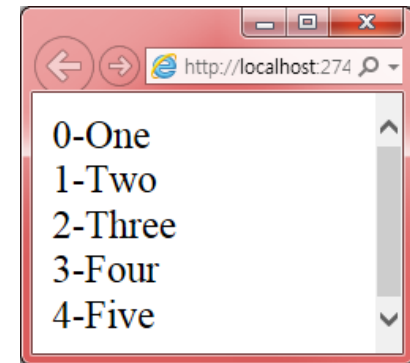
```
<script>
  var s = "Hello World";
  var t = "How are you" + " today?";

  document.write(s + "<br>");
  document.write(t + "<br>");
  document.write(s.toUpperCase() + "<br>");
</script>
```



String

```
<script>
  s = "One,Two,Three,Four,Five";
  array = s.split(',');
  for (i = 0; i < array.length; i++) {
    document.writeln(i + '-' + array[i] + '<BR>');
  }
</script>
```



Boolean

- Only two Boolean literals: **true** and **false**

```
var iLike190M = true;
var ieIsGood = "IE6" > 0;    // false
if ("web dev is great") {    /* true */ }
if (0) { /* false */ }
```

- any value can be used as a Boolean
 - "falsey" values: 0, 0.0, NaN, "", null, and undefined
 - "truthy" values: anything else
- converting a value into a Boolean explicitly:
 - `var boolValue = Boolean(otherValue);`
 - `var boolValue = !!(otherValue);`

Special values: null and undefined

```
var ned = null;  
var benson = 9;  
var caroline;  
  
// at this point in the code,  
//   ned is null  
//   benson's 9  
//   caroline is undefined
```

- `undefined` : has not been declared, does not exist
- `null` : exists, but was specifically assigned an empty or `null` value

JS operators



● 산술 연산자

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

JS operators



● 대입 연산자

Operator	Example	Same As
=	<code>x = y</code>	<code>x = y</code>
+=	<code>x += y</code>	<code>x = x + y</code>
-=	<code>x -= y</code>	<code>x = x - y</code>
*=	<code>x *= y</code>	<code>x = x * y</code>
/=	<code>x /= y</code>	<code>x = x / y</code>
%=	<code>x %= y</code>	<code>x = x % y</code>

JS operators

- 문자열 연산자: +

- String concatenation

```
txt1 = "What a very ";  
txt1 += "nice day";
```

→ What a very nice day

- Adding string and numbers

```
x = 5 + 5;  
y = "5" + 5;  
z = "Hello" + 5;
```

→ 10
55
Hello5

```
var count = 10;  
var s1 = "" + count; // "10"  
var s2 = count + " bananas, ah ah ah!"; // "10 bananas, ah ah ah!"  
var n1 = parseInt("42 is the answer"); // 42  
var n2 = parseFloat("booyah"); // NaN
```

JS operators

● 관계 연산자

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

```
5 < "7"           // true
42 == 42.0         // true
"5.0" == 5         // true
5.0 === 5          // false
"5.0" === 5        // false
```

```
var x = "John";
var y = new String("John");
var z = new String("John");

typeof x; // string
typeof y; // object
(x == y); // true because x and y have same values
(x === y); // false because x and y have different types
(y == z); // false because y and z are different objects
(y === z); // false because y and z are different objects
           // objects cannot be compared
```

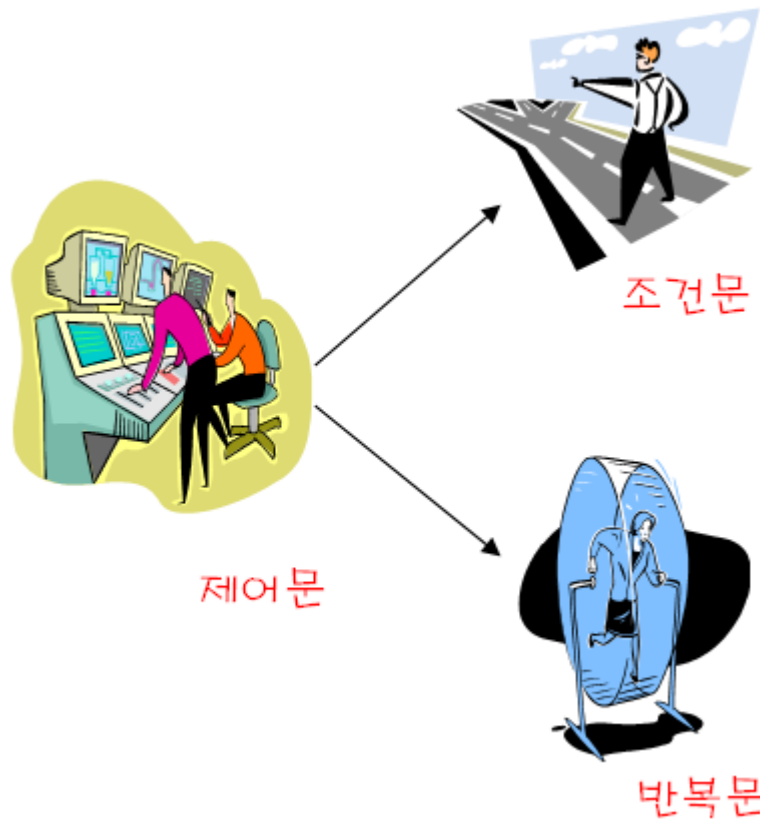
JS operators



● 논리 연산자

Operator	Description
&&	logical and
	logical or
!	logical not

제어문



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

```
while(토플성적 < 800)  
    영어공부;
```

조건문

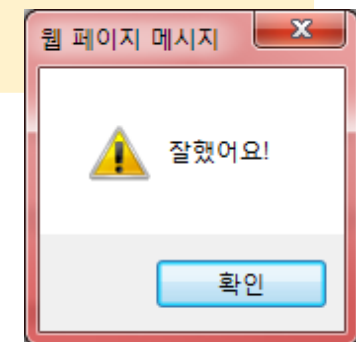
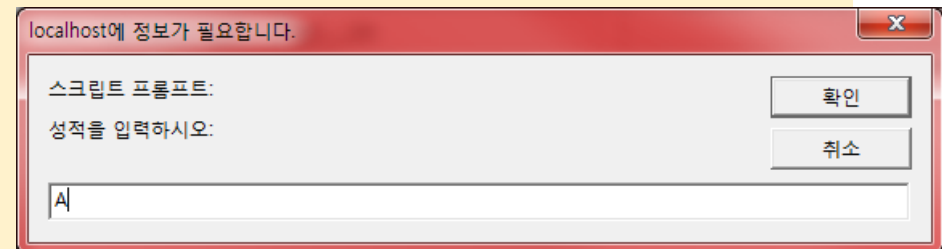


```
if (condition) {  
    statements;  
} else if (condition) {  
    statements;  
} else {  
    statements;  
}
```

- Java 조건문과 동일

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>
```



반복문



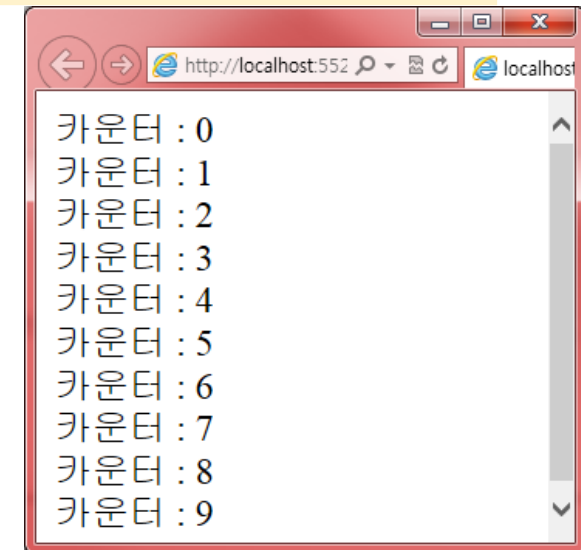
```
for (initialization; condition; update) {  
    statements;  
}
```

```
while (condition) {  
    statements;  
}
```

```
do {  
    statements;  
} while (condition);
```

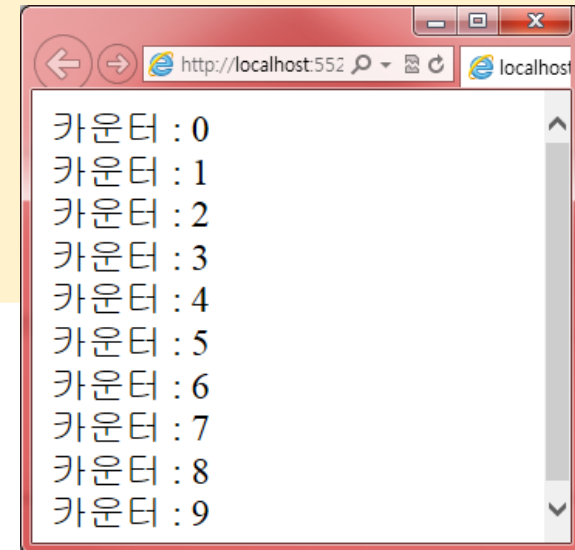
while 문

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



for 문

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

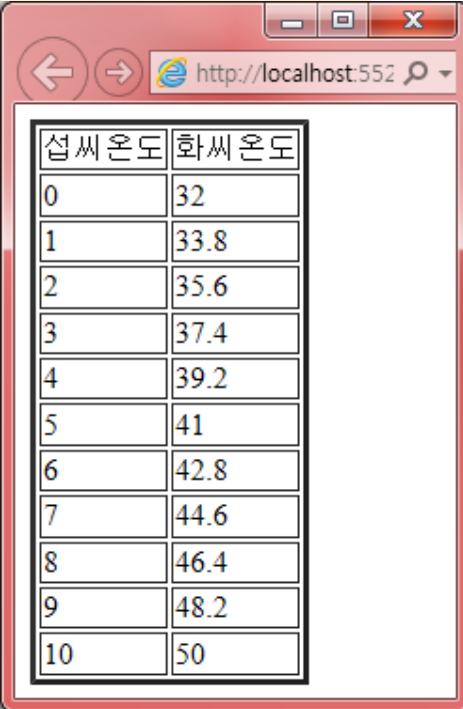


```
var sum = 0;
for (var i = 0; i < 100; i++) {
  sum = sum + i;
}
```

```
var s1 = "hello";
var s2 = "";
for (var i = 0; i < s1.length; i++) {
  s2 += s1.charAt(i) + s1.charAt(i);
}
// s2 stores "hheelllloo"
```

예제

```
<html>
<head>
  <title>온도 변환기</title>
</head>
<body>
  <table border="3">
    <tr>
      <td>섭씨 온도</td>
      <td>화씨 온도</td>
    </tr>
    <script>
      for (celsius = 0; celsius <= 10; celsius = celsius + 1) {
        document.write("<tr><td>" + celsius + "</td><td>"
          + ((celsius * 9.0 / 5) + 32) + "</td></tr>");
      }
    </script>
  </table>
</body>
</html>
```



A screenshot of a web browser window displaying a temperature conversion table. The browser's address bar shows 'http://localhost:552'. The table has two columns: '섭씨 온도' (Celsius Temperature) and '화씨 온도' (Fahrenheit Temperature). It contains 11 rows of data, with Celsius values from 0 to 10 and corresponding Fahrenheit values calculated using the formula $F = \frac{9}{5}C + 32$.

섭씨 온도	화씨 온도
0	32
1	33.8
2	35.6
3	37.4
4	39.2
5	41
6	42.8
7	44.6
8	46.4
9	48.2
10	50

함수

- 입력을 받아서 특정한 작업을 수행하여서 결과를 반환
 - 함수 호출 시, 실행됨
 - JS에서 함수의 type은 'function'이나, object 처럼 취급
 - 객체 Function의 property 및 method 사용 가능

- 함수 선언

```
function functionName(parameters) {  
  code to be executed;  
}
```

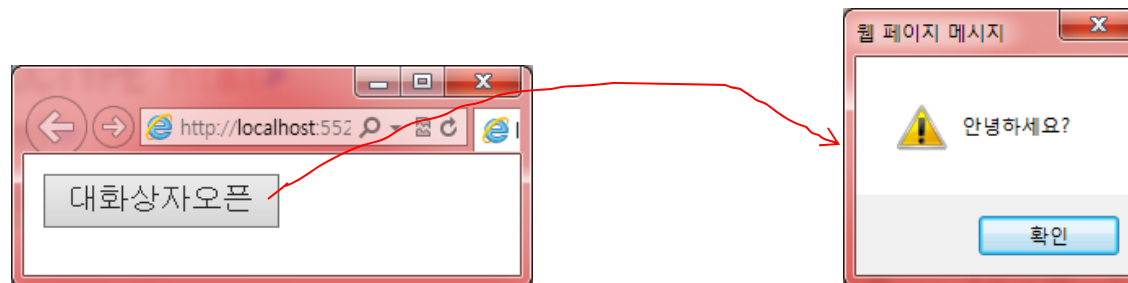
- Anonymous function

- 함수가 특정 이름 없이, 변수에 저장됨
- 저장된 변수에 의해서만 호출 가능

```
var x = function (a, b) {return a * b};  
var z = x(4, 3);
```

예제

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



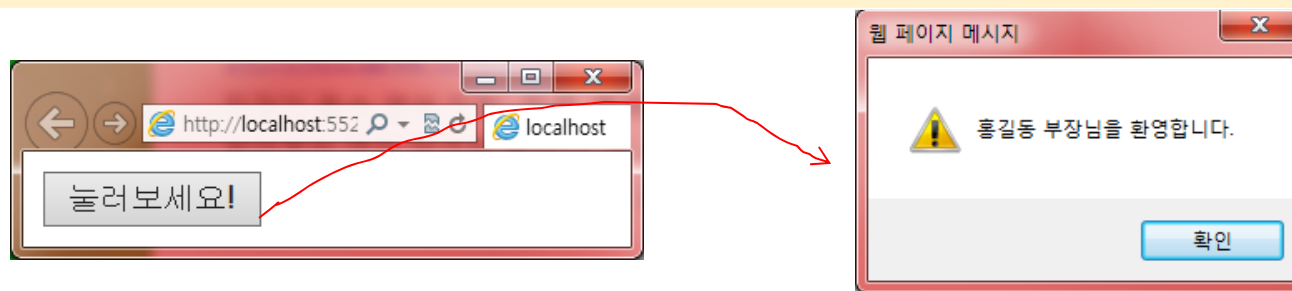
인수와 매개변수



- 매개변수(parameters)
 - 함수 정의에 사용된 변수 명
- 인수 (arguments)
 - 함수 정의에 사용된 각 변수에 할당되는 실제 값
- 기본 정책
 - 매개변수에 자료형을 명시하지 않음
 - 실인수의 자료형을 검사하지 않음
 - 실인수의 개수도 검사하지 않음
 - 매개변수에 할당되는 값이 없을 경우: `undefined`

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>
```



인수와 매개변수



● 인수 전달

● 내장 객체 `argument`

- 함수 호출 시 전달되는 모든 값들을 배열 형태로 포함

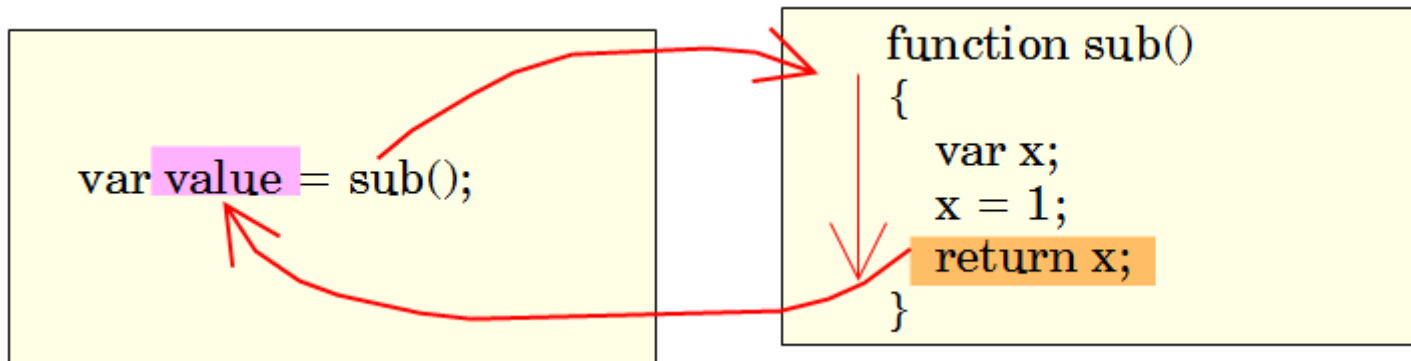
```
x = findMax(1, 123, 500, 115, 44, 88);

function findMax() {
  var i;
  var max = -Infinity;
  for (i = 0; i < arguments.length; i++) {
    if (arguments[i] > max) {
      max = arguments[i];
    }
  }
  return max;
}
```

- 실인수 전달 방식: call by value
- 인자가 Object인 경우 전달 방식: call by reference

함수의 반환값

- return 문장을 사용하여 외부로 값을 반환



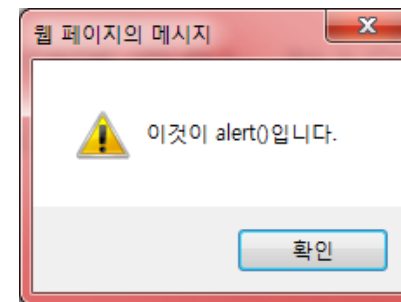
Popup Boxes



```
alert("message");    // message  
confirm("message");  // returns true or false  
prompt("message");   // returns user input string
```

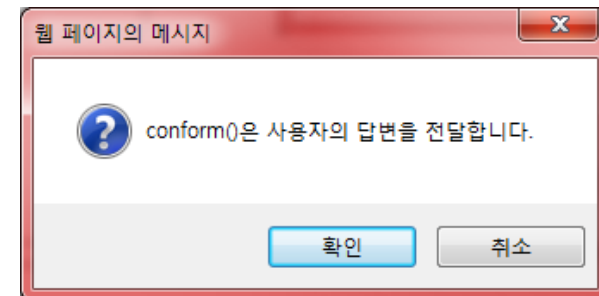
alert() 함수

```
<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>
```

