

제6장 자바스크립트 기본

1. 자바스크립트 언어의 요소와 구조
2. 자바스크립트 코드를 웹 페이지에 삽입하는 방법
3. 자바스크립트로 브라우저에 출력하고 사용자 입력 받는 방법
4. 자바스크립트 데이터 타입과 변수
5. 자바스크립트 연산자
6. 자바스크립트 조건문
7. 자바스크립트 반복문
8. 자바스크립트 함수
9. 내장함수

자바스크립트 언어

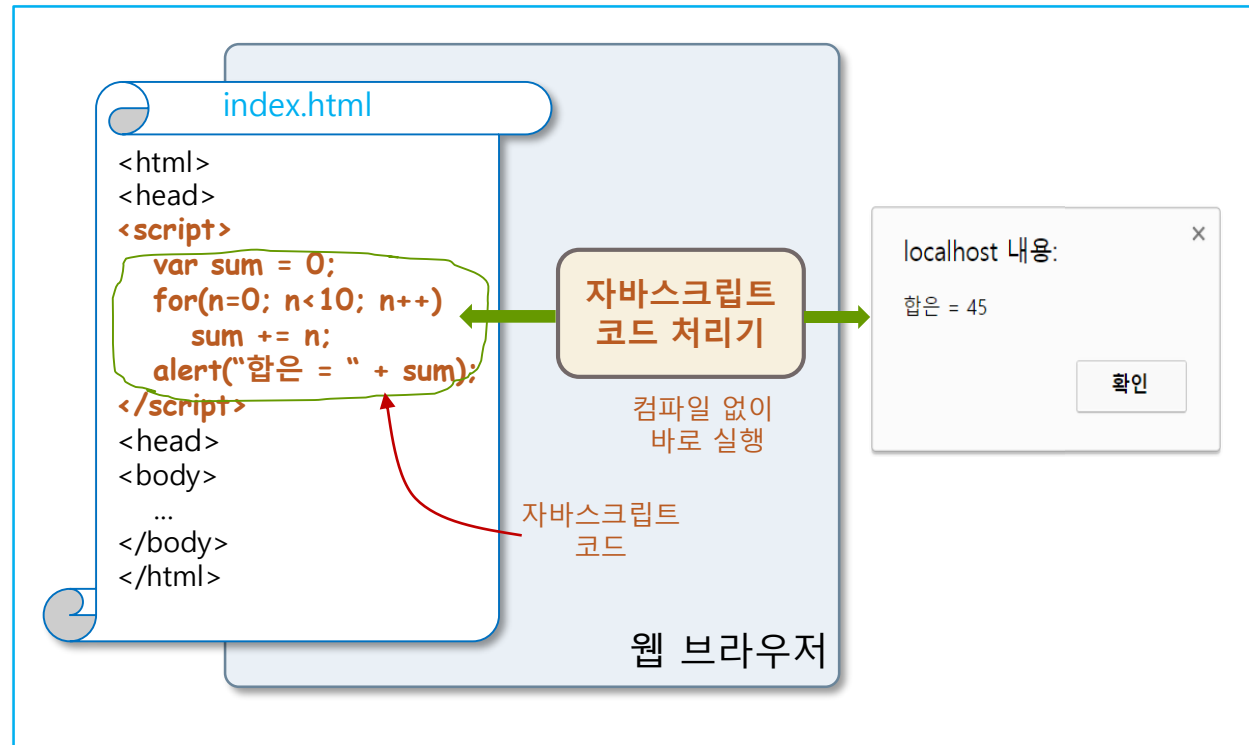
2

□ Javascript

- 1995년 넷스케이프 개발
- Netscape Navigator 2.0 브라우저에 최초 탑재
- 웹 프로그래밍 개념 창시

□ 특징

- HTML 문서에 내장
 - 조각 소스 코드
- 스크립트 언어
 - 인터프리터 실행
 - 컴파일 필요 없음
- 단순
 - C언어 구조 차용
 - 배우기 쉬움



웹 페이지에서 자바스크립트의 역할

3

- 사용자의 입력 및 계산
 - ▣ 마우스와 키보드 입력은 오직 자바스크립트로만 가능
 - ▣ 계산 기능
- 웹 페이지 내용 및 모양의 동적 제어
 - ▣ HTML 태그의 속성, 콘텐츠, CSS 프로퍼티 값 동적 변경
- 브라우저 제어
 - ▣ 브라우저 윈도우 크기와 모양 제어
 - ▣ 새 윈도우 열기/닫기
 - ▣ 다른 웹 사이트 접속
 - ▣ 히스토리 제어
- 웹 서버와의 통신
- 웹 애플리케이션 작성
 - ▣ 캔버스 그래픽, 로컬/세션 스토리지 저장, 위치정보서비스 등

자바스크립트 코드의 위치

4

□ 자바스크립트 코드 작성이 가능한 위치

1. HTML 태그의 이벤트 리스너 속성에 작성
2. <script></script> 태그에 작성
3. 자바스크립트 파일에 작성
4. URL 부분에 작성

1. HTML 태그의 이벤트 리스너에 자바스크립트 코드 작성

onclick 이벤트 리스너 속성 자바스크립트 코드 (이미지를 banana.png로 교체)

```

```

↑

HTML 태그의 이벤트 리스너 속성에 자바스크립트 코드 작성

5

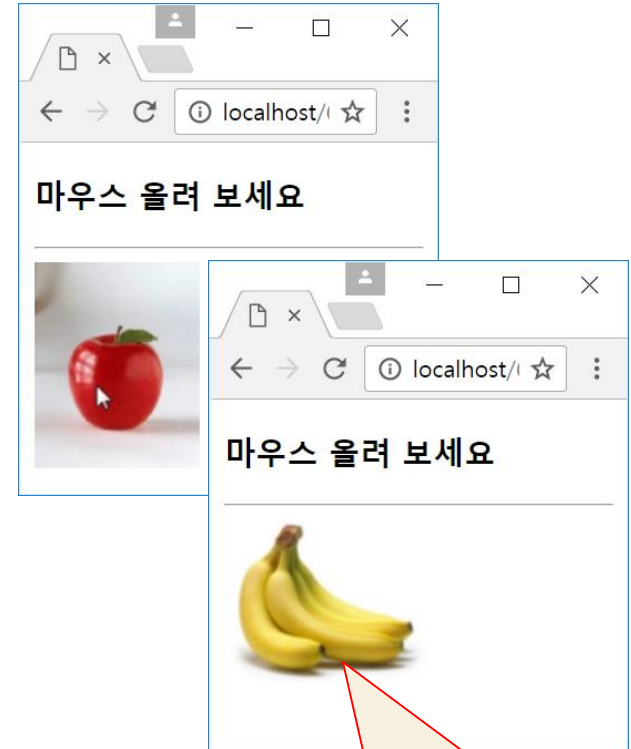
```
<!DOCTYPE html>
<html>
<head>
<title>이벤트 리스너 속성에 자바스크립트 코드</title>
</head>
<body>
<h3>마우스 올려 보세요</h3>
<hr>

</body>
</html>
```

이벤트 리스너
속성

this는 현재 img 태그를
가리키는 자바스크립트 키워드

자바스크립트
코드



이미지에 마우스를 올리면 바나나로
내리면 다시 사과로 바뀐다.

<script> </script> 태그에 자바스크립트 작성

6

□ 특징

- ▣ <head> </head>나 <body> </body> 내 어디든 가능
- ▣ 웹 페이지 내에 여러 번 삽입 가능

<script> 태그에 자바스크립트 코드 작성

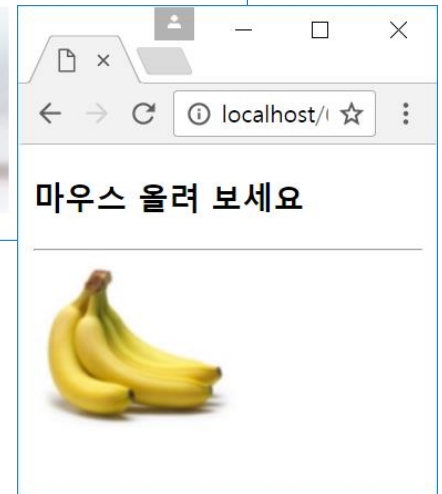
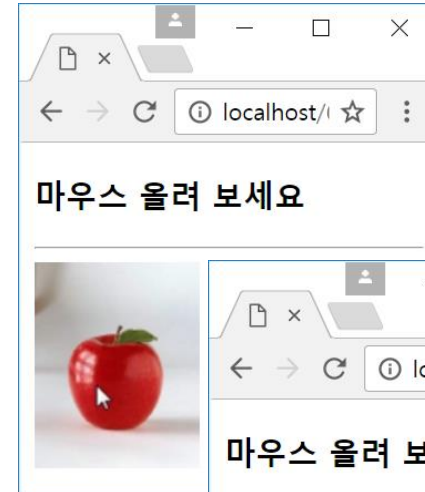
7

```
<!DOCTYPE html>
<html>
<head> <title>script 태그에 자바스크립트 작성</title>
<script>
function over(obj) {
  obj.src="media/banana.png";
}
function out(obj) {
  obj.src="media/apple.png";
}
</script>
</head>
<body>
<h3>마우스 올려 보세요</h3>
<hr>

</body>
</html>
```

obj는 전달받은
img 태그를 가리킴

this는 현재 img 태그를
가리키는 자바스크립트 키워드



자바스크립트 코드를 별도 파일에 작성

8

- 자바스크립트 코드 파일 저장
 - ▣ 확장자 .js 파일에 저장
 - ▣ <script> 태그 없이 자바스크립트 코드만 저장
- 여러 웹 페이지에서 불러 사용
 - 웹 페이지마다 자바스크립트 코드 작성 중복 불필요
 - <script> 태그의 src 속성으로 파일을 불러 사용

```
<script src="파일이름.js">  
  // HTML5부터 이곳에 자바스크립트 코드 추가 작성하면 안 됨  
</script>
```


자바스크립트 파일 작성 및 불러오기

9

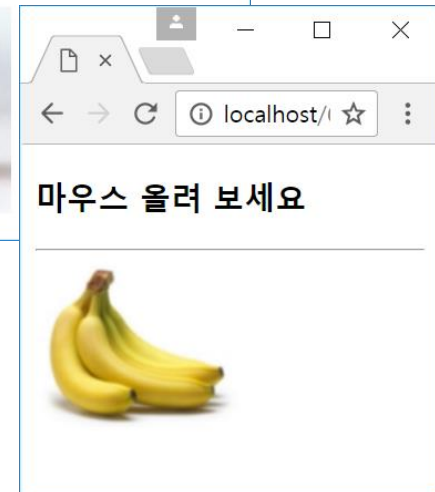
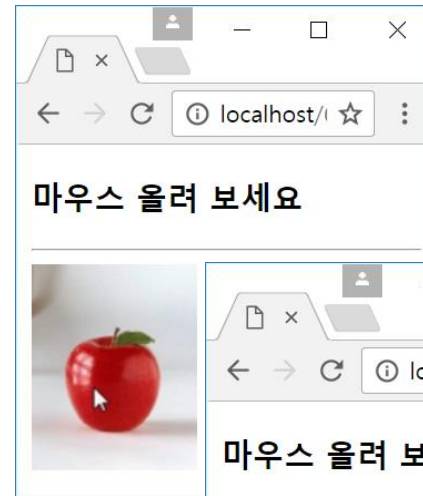
<script> 태그에 들어 있는 자바스크립트 코드를 "lib.js" 파일에 저장 후 사용

```
/* 자바스크립트 파일 lib.js */  
function over(obj) {  
    obj.src="media/banana.png";  
}  
function out(obj) {  
    obj.src="media/apple.png";  
}
```

lib.js

lib.js
불러오기

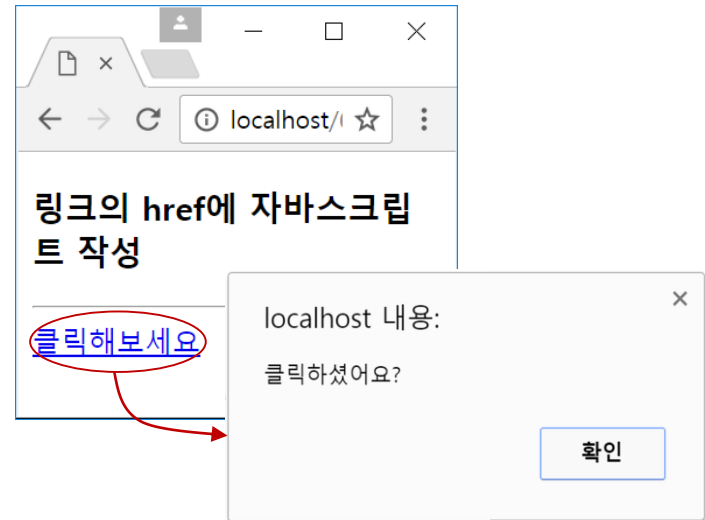
```
<!DOCTYPE html>  
<html>  
<head><title>외부 파일에 자바스크립트 작성</title>  
<script src="lib.js">  
</script>  
</head>  
<body>  
<h3>마우스 올려 보세요</h3>  
<hr>  
  
</body>  
</html>
```



링크의 href에 자바스크립트 코드 작성

10

```
<!DOCTYPE html>
<html>
<head> <title>URL에 자바스크립트 작성</title>
</head>
<body>
<h3>링크의 href에 자바스크립트 작성</h3>
<hr>
<a href="javascript:alert('클릭하셨어요?')">
  클릭해보세요</a>
</body>
</html>
```



자바스크립트로 HTML 콘텐츠 출력

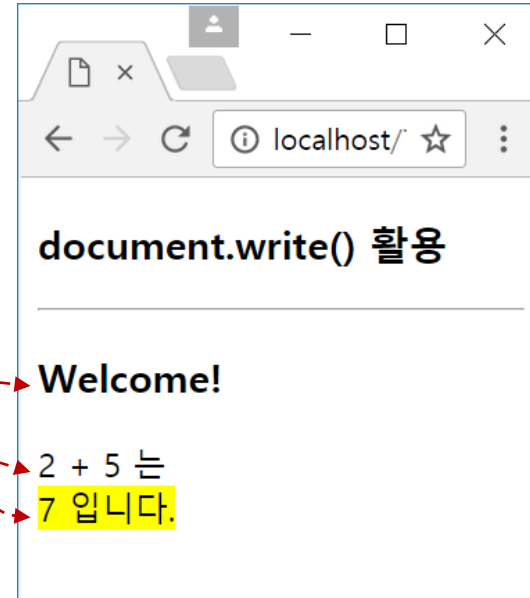
11

- 자바스크립트로 HTML 콘텐츠를 웹 페이지에 직접 삽입
 - ▣ 바로 브라우저 윈도우에 출력
 - ▣ `document.write()`
 - 예) `document.write("<h3>Welcome!</h3>");`
 - ▣ `document.writeln()`
 - `writeln()`은 텍스트에 'Wn ' 을 덧붙여 출력
 - 'Wn'을 덧붙이는 것은 고작해야 빈칸 하나 출력
 - 다음 줄로 넘어가는 것은 아님

document.write()로 웹 페이지에 HTML 콘텐츠 출력

12

```
<!DOCTYPE html>
<html>
<head> <title>document.write() 활용 </title>
</head>
<body>
<h3>document.write() 활용 </h3>
<hr>
<script>
  document.write("<h3>Welcome!</h3>");
  document.write("2 + 5 는 <br>");
  document.write("<mark>7 입니다.</mark>");
</script>
</body>
</html>
```

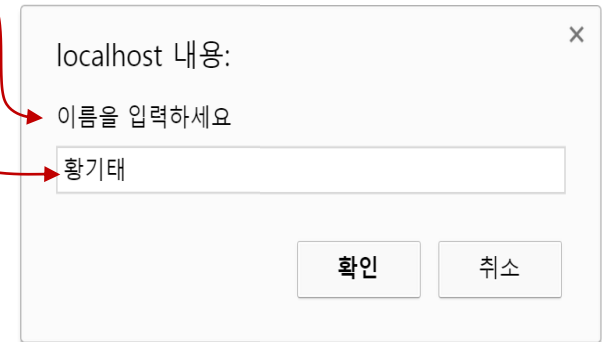


자바스크립트 다이얼로그 : 프롬프트 다이얼로그

13

- `prompt("메시지", "디폴트 입력값")` 함수
 - 사용자로부터 문자열을 입력 받아 리턴

```
var ret = prompt("이름을 입력하세요", "황기태");  
if(ret == null) {  
    // 취소 버튼이나 다이얼로그를 닫은 경우  
}  
else if(ret == "") {  
    // 문자열 입력 없이 확인 버튼 누른 경우  
}  
else {  
    // ret에는 사용자가 입력한 문자열  
}
```



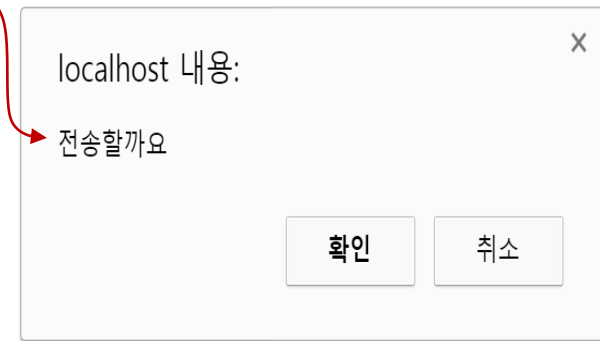
자바스크립트 다이얼로그 : 확인 다이얼로그

14

■ confirm("메시지") 함수

- “메시지”를 출력하고 ‘확인/최소(OK/CANCEL)’버튼을 가진 다이얼로그 출력
- ‘확인’ 버튼을 누르면 true, '취소' 버튼이나 강제로 다이얼로그를 닫으면 false 리턴

```
var ret = confirm("전송할까요");  
if(ret == true) {  
    // 사용자가 "확인" 버튼을 누른 경우  
}  
else {  
    // 취소 버튼이나 다이얼로그를 닫은 경우  
}
```

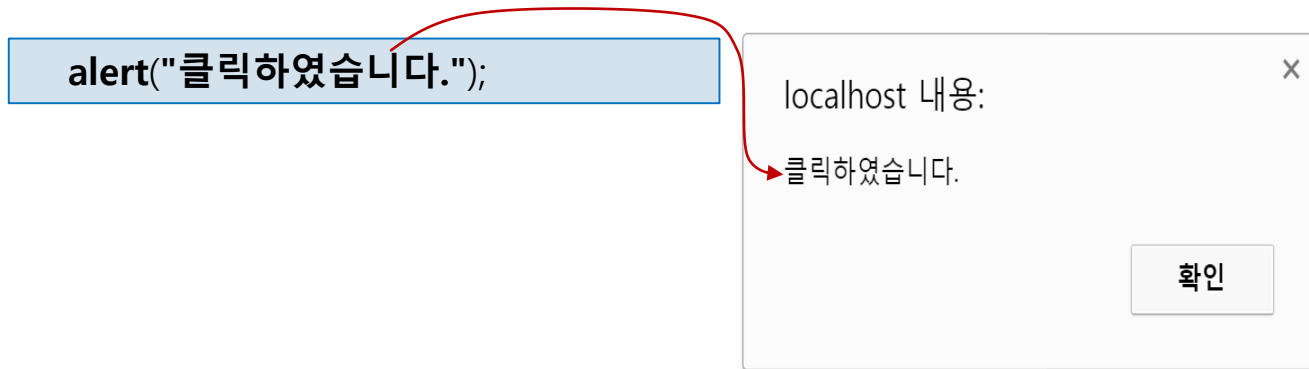


자바스크립트 다이얼로그 : 경고 다이얼로그

15

□ alert("메시지") 함수

- ▣ 메시지'와 '확인' 버튼을 가진 다이얼로그 출력, 메시지 전달





데이터와 연산자

자바스크립트 식별자

17

□ 식별자

- 자바스크립트 프로그램의 변수, 상수(리터럴), 함수의 이름

- 식별자 만드는 규칙

- 첫 번째 문자 : 알파벳(A-Z, a-z), 언더스코어(_), \$ 문자만 사용 가능
- 두 번째 이상 문자 : 알파벳, 언더스코어(_), 0-9, \$ 사용 가능
- 대소문자는 구분되어 다루어짐
 - *myHome*과 *myhome*은 다른 식별자
- 자바스크립트 예약어 사용 불가
 - *false, for, if, null* 등 자바스크립트 예약어 사용 불가

- 식별자 사용 사례

```
6variable;      // (x) 숫자로 시작할 수 없음
student_ID;    // (0)
_code;         // (0) 맞지만 권하지 않음
if;            // (x) 예약어 if 사용 불가
%calc          // (x) % 사용 불가
bar, Bar;      // (0) bar와 Bar는 서로 다른 식별자임에 주의
```

자바스크립트 문장

18

□ 문장

- 자바스크립트 프로그램의 기본 단위는
- 문장과 문장을 구분하기 위해 세미콜론(;) 사용

```
i = i + 1           // (0) 한 줄에 한 문장만 있는 경우 세미콜론 생략 가능  
j = j + 1;          // (0)  
k = k + 1; m = m + 1; // (0) 한 줄에 여러 문장  
n = n + 1; p = p + 1; // (x) 첫 번째 문장 끝에 세미콜론이 필요함
```

□ 주석문

```
// 한 라인 주석. 라인의 끝까지 주석 처리  
/*  
    여러 라인 주석  
*/
```

데이터 타입

19

- 자바스크립트 언어에서 다루는 데이터 종류
 - ▣ 숫자 타입 : 정수, 실수(예: 42, 3.14)
 - ▣ 논리 타입 : 참, 거짓(예: true, false)
 - ▣ 문자열 타입(예: '좋은 세상', "a", "365", "2+4")
 - ▣ 객체 레퍼런스 타입 : 객체를 가리킴. C 언어의 포인터와 유사
 - ▣ null : 값이 없음을 표시하는 특수 키워드. Null, NULL과는 다름
- 특징
 - ▣ 자바스크립트에는 문자 타입 없음. 문자열로 표현

- 변수 : 자바스크립트 데이터 저장 공간
 - ▣ 변수 선언 : 변수 이름을 정하고, 저장 공간 할당
 - var 키워드로 선언하는 방법

```
var score;           // 변수 score 선언
var year, month, day; // year, month, day의 3 개의 변수 선언
var address = "서울시"; // address 변수를 선언하고 "서울시"로 초기화
```

- var 없이 선언

```
age = 21;           // var 없이, 변수 age를 선언하고 21로 초기화
```

- *age*가 이미 선언된 변수이면, 존재하는 *age*에 21 저장

- ▣ 자바스크립트에는 변수 타입 없음

- 변수 타입 선언하지 않음

```
var score; // 정상적인 변수 선언
int score; // 오류. 변수 타입 int 없음
```

- 변수에 저장되는 값에 대한 제약 없음

```
score = 66.8; // 실수도 저장 가능
score = "high"; // 문자열로 저장 가능
```

지역변수와 전역변수

21

지역변수	전역변수
함수 내에 var 키워드로 선언	함수 밖에 선언되거나, 함수 내에 var 키워드 없이 선언된 변수
선언된 함수 내에서만 사용	프로그램 전역에서 사용

```
var x;    // 전역변수 x
function f() {
  var y;  // 지역변수 y 선언
  x = 10; // 전역 변수 x에 10 저장
  y = 10; // 지역 변수 y에 10 저장
  z = 10; // 새로운 전역변수 z 선언. 10으로 초기화
}
```

this로 전역변수 접근

22

- 지역 변수와 전역 변수의 이름이 같을 때
 - ▣ 전역 변수에 접근하고자 할 때 : **this.전역변수**

```
var x;    // 전역변수
function f() {
  var x;   // 지역변수
  x = 1;   // 지역변수 x에 1 저장
  this.x = 100; // 전역변수 x에 100 저장
}
```

지역변수와 전역변수

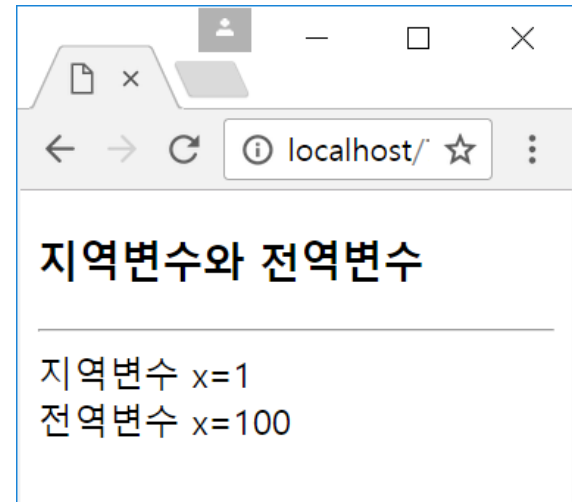
23

```
<!DOCTYPE html>
<html>
<head>
<title>지역변수와 전역변수</title> </head>
<body>
<h3>지역변수와 전역변수</h3>
<hr>
<script>
var x=100; // 전역변수 x

function f() { // 함수 f() 선언
    var x=1; // 지역변수 x

    document.write("지역변수 x=" + x);
    document.write("<br>");
    document.write("전역변수 x=" + this.x);
}

f(); // 함수 f() 호출
</script>
</body>
</html>
```



자바스크립트의 상수

24

- ▣ 상수(literal)
 - 데이터 값 그 자체
- ▣ 상수 종류

상수의 종류		특징	예
정수	8진수	0으로 시작	<code>var n = 015; // 8진수 15. 10진수로 13</code>
	10진수		<code>var n = 15; // 10진수 15</code>
	16진수	0x로 시작	<code>var n = 0x15; // 16진수 15. 10진수로 21</code>
실수	소수형		<code>var height = 0.1234;</code>
	지수형		<code>var height = 1234E-4; // $1234 \times 10^{-4} = 0.1234$</code>
논리	참	true	<code>var condition = true;</code>
	거짓	false	<code>var condition = false;</code>
문자열		""로 묶음	<code>var hello = "안녕하세요";</code>
		''로 묶음	<code>var name = 'kitae';</code>
기타	null	값이 없음을 뜻함	<code>var ret = null;</code>
	NaN	수가 아님을 뜻함	<code>var n = parseInt("abc"); // 이때 parseInt()는 NaN을 리턴</code>

문자열 상수

25

- ▣ 이중 인용 부호("")와 단일 인용 부호('') 모두 사용
- ▣ 문자열 내에 문자열

문자열 내 문자열

```
<p onmouseover = "document.body.style.color = 'brown' ">
```

- “ 문자를 그대로 사용하고자 하는 경우 ₩”로 사용할 것

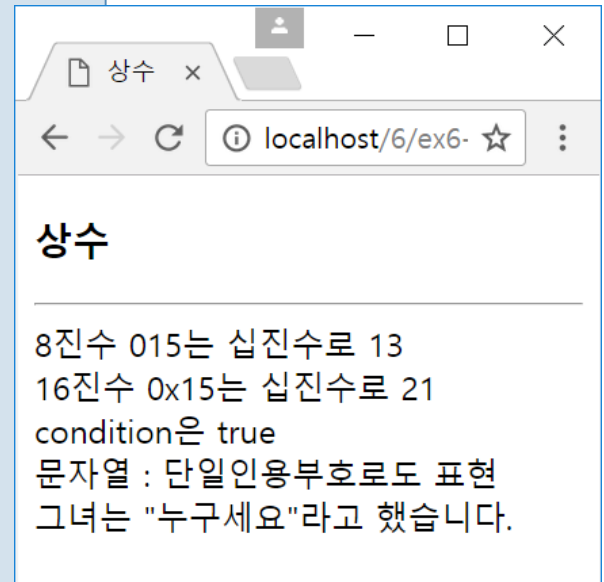
```
var cite="그녀는 ₩"누구세요₩"라고 했습니다.;"
```

상수

26

```
<!DOCTYPE html>
<html>
<head> <title>상수</title> </head>
<body>
<h3>상수</h3>
<hr>
<script>
  var oct = 015; // 015는 8진수. 10진수로 13
  var hex = 0x15; // 0x15는 16진수. 10진수로 21
  var condition = true; // True로 하면 안됨

  document.write("8진수 015는 십진수로 " + oct + "<br>");
  document.write("16진수 0x15는 십진수로 " + hex + "<br>");
  document.write("condition은 " + condition + "<br>");
  document.write('문자열 : 단일인용부호로도 표현' + "<br>");
  document.write("그녀는 ₩"누구세요₩"라고 했습니다.");
</script>
</body>
</html>
```



자바스크립트의 식과 연산

27

자바스크립트의 연산과 연산자 종류

연산 종류	연산자	연산 종류	연산자
산술	+ - * / %	대입	= *= /= += -= &= ^= = <=>= >>=
증감	++ --	비교	> < >= <= == !=
비트	& ^ ~	논리	&& !
시프트	>> << >>>	조건	? :

산술 연산자

- 5 가지 : 더하기(+), 빼기(-), 곱하기(*), 나누기(/), 나머지(%)

```
var x = 32;  
var total = 100 + x*2/4 - 3; // total은 113
```

- 연산의 결과는 항상 실수

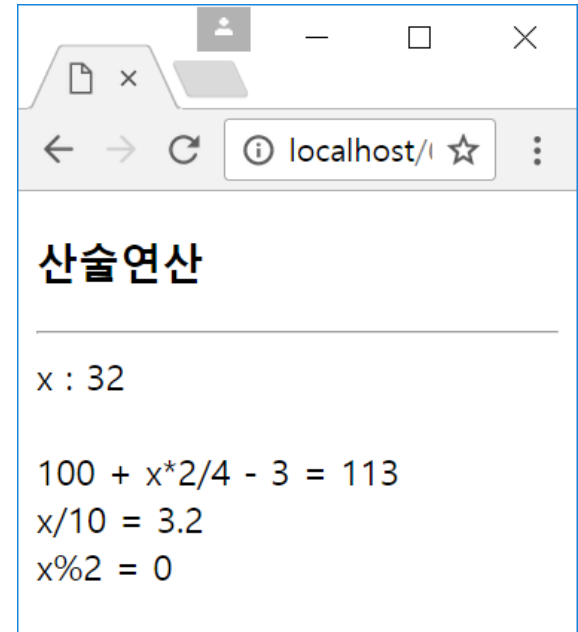
```
var div = 32/10;           // div = 3.2
```

산술 연산

28

```
<!DOCTYPE html>
<html>
<head> <title>산술연산</title> </head>
<body>
<h3>산술연산</h3>
<hr>
<script>
  var x=32;
  var total = 100 + x*2/4 - 3; // total은 113
  var div = x / 10; // div는 3.2
  var mod = x % 2; // x를 2로 나눈 나머지, 0

  document.write("x : " + x + "<br><br>");
  document.write("100 + x*2/4 - 3 = " + total + "<br>");
  document.write("x/10 = " + div + "<br>");
  document.write("x%2 = " + mod + "<br>");
</script>
</body>
</html>
```

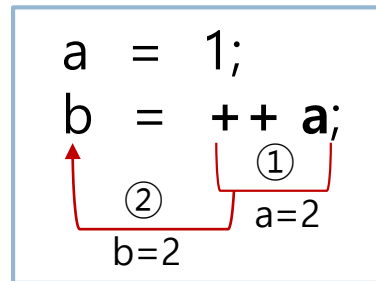


증감 연산자

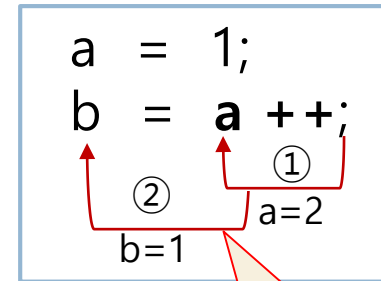
29

증감 연산자 : ++, --

(a) 전위연산자



(b) 후위연산자



a++의 연산은 증가
전의 값 1을 반환하
여 b의 값은 1이 됨

연산자	내용	연산자	내용
a++	a를 1 증가하고 증가 전의 값 반환	++a	a를 1 증가하고 증가된 값 반환
a--	a를 1 감소하고 감소 전의 값 반환	--a	a를 1 감소하고 감소된 값 반환

대입 연산자

30

□ 대입 연산 : 오른쪽 식의 결과를 왼쪽 변수에 대입

```
var a=1, b=3;  
a = b;    // a에 b의 값이 대입되어 a=3, b=3이 된다.  
a += b;   // a = a + b의 연산이 이루어져, a=6, b=3이 된다.
```

□ 대입연산자 종류

연산자	내용	연산자	내용
a = b	b 값을 a에 대입	a &= b	a = a & b와 동일
a += b	a = a + b와 동일	a ^= b	a = a ^ b와 동일
a -= b	a = a - b와 동일	a = b	a = a b와 동일
a *= b	a = a * b와 동일	a <<= b	a = a << b와 동일
a /= b	a = a / b와 동일	a >>= b	a = a >> b와 동일
a %= b	a = a % b와 동일	a >>>= b	a = a >>> b와 동일

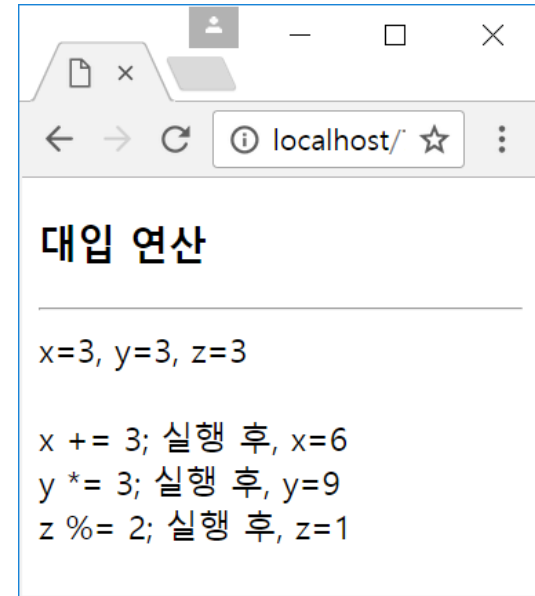
대입 연산

31

```
<!DOCTYPE html>
<html>
<head> <title>대입 연산</title> </head>
<body>
<h3>대입 연산</h3>
<hr>
<script>
  var x=3, y=3, z=3;
  document.write("x=" + x + ", y=" + y);
  document.write(", z=" + z + "<br><br>");

  x += 3; // x=x+3 -> x=6
  y *= 3; // y=y*3 -> y=9
  z %= 2; // z=z%2 -> z=1

  document.write("x += 3; 실행 후, x=" + x + "<br>");
  document.write("y *= 3; 실행 후, y=" + y + "<br>");
  document.write("z %= 2; 실행 후, z=" + z);
</script>
</body>
</html>
```



비교 연산자

32

- 비교 연산 : 두 값 비교, true나 false의 결과를 내는 연산

```
var age = 25;  
var result = (age > 20); // age가 20보다 크므로 result는 true
```

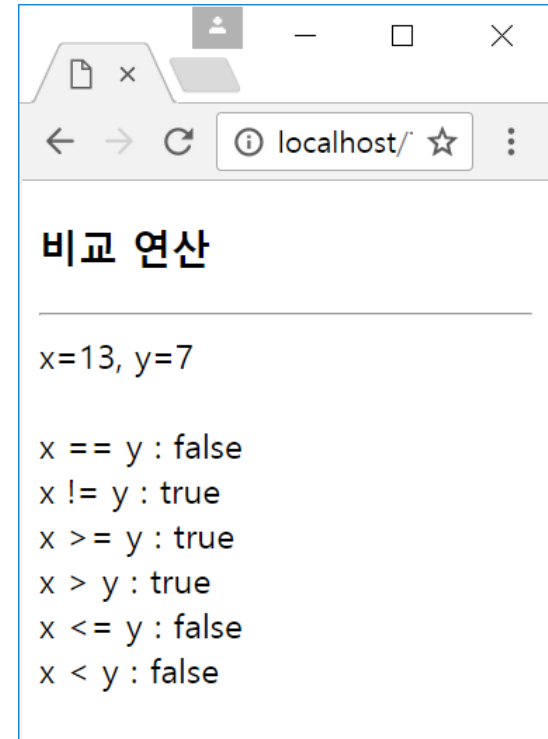
- 비교 연산자 종류

연산자	내용	연산자	내용
$a < b$	a가 b보다 작으면 true	$a \geq b$	a가 b보다 크거나 같으면 true
$a > b$	a가 b보다 크면 true	$a == b$	a가 b와 같으면 true
$a \leq b$	a가 b보다 작거나 같으면 true	$a != b$	a가 b와 같지 않으면 true

비교 연산

33

```
<!DOCTYPE html>
<html>
<head> <title>비교 연산</title> </head>
<body>
<h3>비교 연산</h3>
<hr>
<script>
  var x=13, y=7;
  document.write("x=" + x + ", y=" + y + "<br><br>");
  document.write("x == y : " + (x == y) + "<br>");
  document.write("x != y : " + (x != y) + "<br>");
  document.write("x >= y : " + (x >= y) + "<br>");
  document.write("x > y : " + (x > y) + "<br>");
  document.write("x <= y : " + (x <= y) + "<br>");
  document.write("x < y : " + (x < y) + "<br>");
</script>
</body>
</html>
```



논리 연산자

34

□ 논리 연산 : AND, OR, NOT

```
var score = 90;  
var age = 20;  
var res = ((score > 80) && (age < 25)); // res=true
```

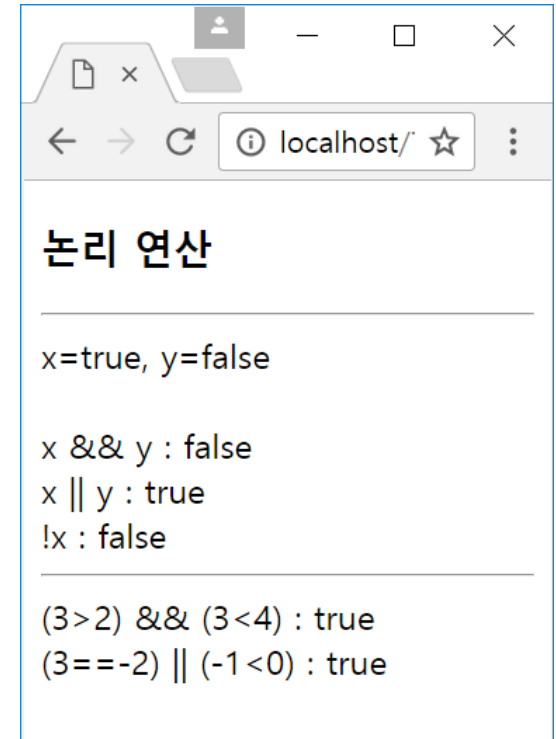
□ 논리 연산 종류

연산자	별칭	내용
a && b	논리 AND 연산	a, b 모두 true일 때 true 리턴
a b	논리 OR 연산	a, b 중 하나라도 true이면 true 리턴
!a	논리 NOT 연산	a가 true이면 false 값을, false이면 true 값 리턴

논리 연산

35

```
<!DOCTYPE html>
<html>
<head><title>논리 연산</title></head>
<body>
<h3>논리 연산</h3>
<hr>
<script>
  var x=true, y=false;
  document.write("x=" + x + ", y=" + y + "<br><br>");
  document.write("x && y : " + (x&&y) + "<br>");
  document.write("x || y : " + (x||y) + "<br>");
  document.write("!x : " + (!x) + "<br>");
  document.write("<hr>");
  document.write("(3>2) && (3<4) : " + ((3>2)&&(3<4)) + "<br>");
  document.write("(3==2) || (-1<0) : " + ((3==2)||(-1<0)));
</script>
</body>
</html>
```



조건 연산자

36

□ 조건 연산

▣ condition ? expT : expF

- condition이 true이면 전체 결과는 expT의 계산 값
- false이면 expF의 계산 값

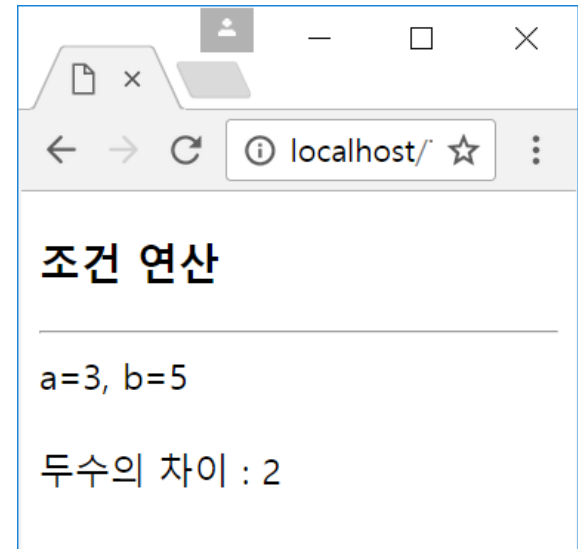
```
var x=5, y=3;
```

```
var big = (x>y) ? x : y; // (x>y)가 true이므로 x 값 5가 big에 대입된다.
```

조건 연산

37

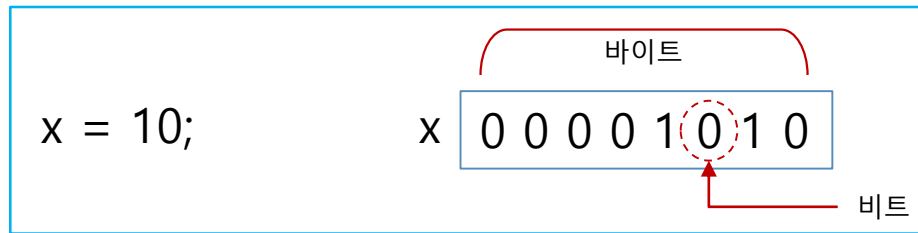
```
<!DOCTYPE html>
<html>
<head> <title>조건 연산</title> </head>
<body>
<h3>조건 연산</h3>
<hr>
<script>
  var a=3, b=5;
  document.write("a=" + a + ", b=" + b + "<br><br>");
  document.write("두수의 차이 : " + ((a>b)?(a-b):(b-a)));
</script>
</body>
</html>
```



비트 연산

38

□ 비트 개념



□ 비트 연산 종류

- ▣ 비트들끼리의 비트 논리 연산
- ▣ 비트 시프트 연산

비트 논리 연산

39

□ 비트 논리 연산

연산자	별칭	연산 설명
$a \& b$	비트 AND 연산	두 비트 모두 1이면 1, 그렇지 않으면 0
$a b$	비트 OR 연산	두 비트 모두 0이면 0, 그렇지 않으면 1
$a \wedge b$	비트 XOR 연산	두 비트가 다르면 1, 같으면 0
$\sim a$	비트 NOT 연산	1을 0으로, 0을 1로 변환

a = 106; 01101010

b = 77; 01001101

c = a & b;

```
  0 1 1 0 1 0 1 0
& 0 1 1 0 1 0 1 0
-----
c 0 1 1 0 1 0 1 0
```

둘 다 1,
결과 1

하나라도 0,
결과 0

c = a | b;

```
  0 1 1 0 1 0 1 0
| 0 1 0 0 1 1 0 1
-----
c 0 1 1 0 1 1 1 1
```

둘 다 0,
결과 1

하나라도 1,
결과 1

c = a ^ b;

```
  0 1 1 0 1 0 1 0
^ 0 1 0 0 1 1 0 1
-----
c 0 0 1 0 0 1 1 1
```

둘이 같으면
결과 0

둘이 다르면,
결과 1

c = ~a;

```
~ 0 1 1 0 1 0 1 0
c 1 0 0 1 0 1 0 1
```

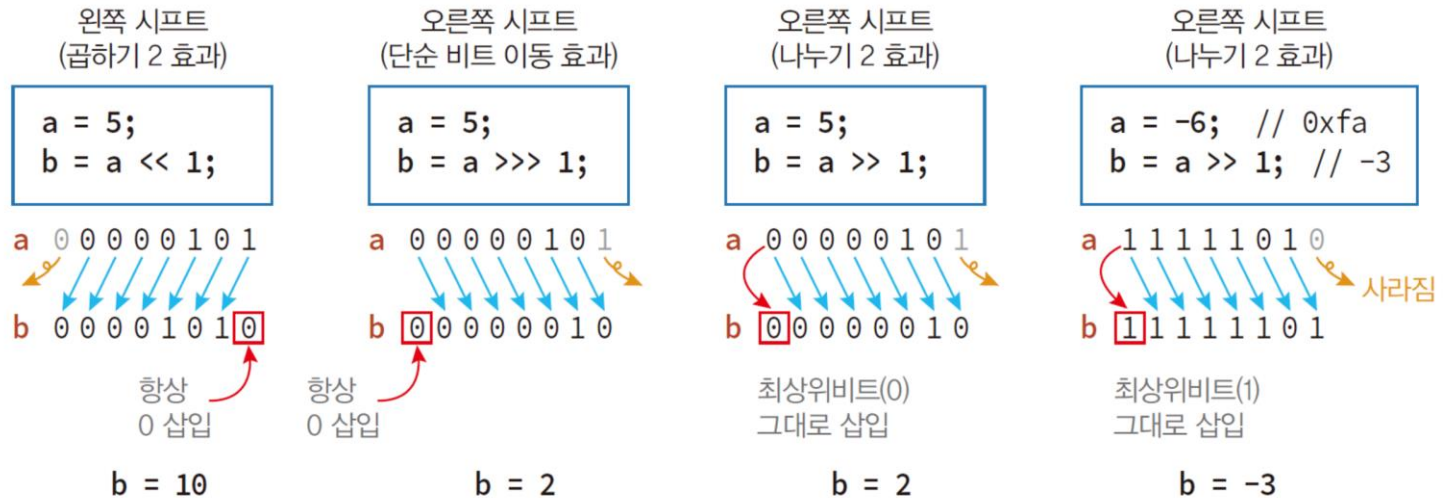
1은 0으로
바꿈

0은 1로
바꿈

비트 시프트 연산

40

□ 시프트 : 저장 공간에서 비트들의 오른쪽/왼쪽 이동



연산자	별칭	설명
$a \ll b$	산술적 왼쪽 시프트	a의 비트들을 왼쪽으로 b번 이동. 최하위 비트의 빈자리는 0으로 채움. 한 비트 시프트마다 곱하기 2의 효과 발생. a 값은 변화 없음
$a \gg b$	산술적 오른쪽 시프트	a의 비트들을 오른쪽으로 b번 이동. 최상위 비트의 빈자리는 시프트 전 최상위 비트로 채움. 한 비트 시프트마다 나누기 2의 효과 발생. a 값은 변화 없음
$a \ggg b$	논리적 오른쪽 시프트	a의 비트들을 오른쪽으로 b번 이동. 최상위 비트의 빈자리는 0으로 채움. a 값은 변화 없음

문자열 연산자

41

▣ 문자열 연결

■ + , +=

```
"abc" + "de"      // "abcde"
"abc" + 23        // "abc23"
23 + "abc"        // "23abc"
23 + "35"         // "2335"
23 + 35           // 58, 정수 더하기
```

■ 순서에 유의

```
23 + 35 + "abc";  // 23 + 35 -> 58로 먼저 계산, 58 + "abc" -> "58abc"
"abc" + 23 + 35;  // "abc" + 23 -> "abc23"로 먼저 계산, "abc23" + 35 -> "abc2335"
```

▣ 문자열 비교

■ 비교 연산자(!=, ==, >, <, <=, >=)는 문자열 비교에 사용

■ 사전 순으로 비교 결과 리턴

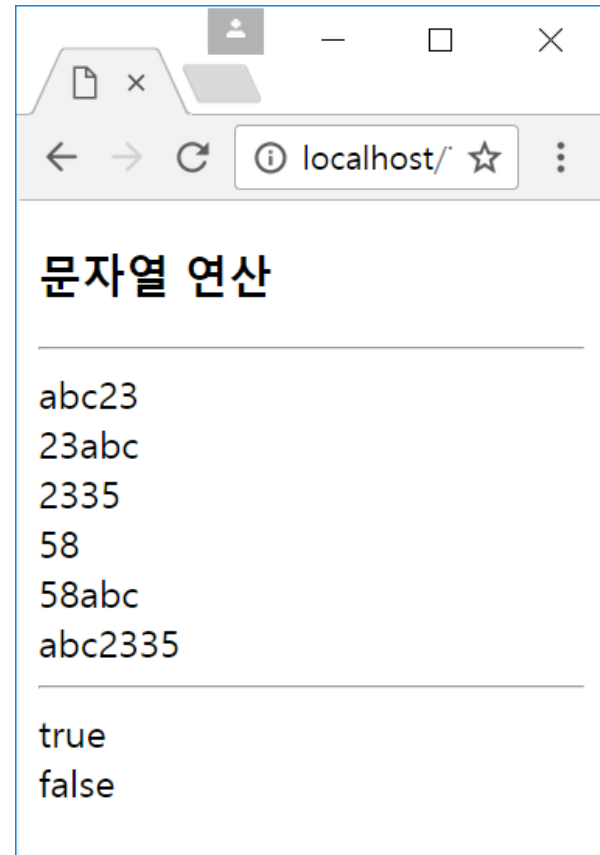
```
var name = "kitae";
var res = (name == "kitae"); // 비교 결과 true, res = true
var res = (name > "park");   // name이 "park"보다 사전순으로 앞에 나오므로 res = false
```

문자열 연산

42

```
<!DOCTYPE html>
<html>
<head> <title> 문자열 연산 </title> </head>
<body>
<h3>문자열 연산</h3>
<hr>
<script>
  document.write("abc" + 23 + "<br>");
  document.write(23 + "abc" + "<br>");
  document.write(23 + "35" + "<br>");
  document.write(23 + 35 + "<br>");
  document.write(23 + 35 + "abc" + "<br>");
  document.write("abc" + 23 + 35 + "<br><hr>");

  var name = "kitae";
  document.write(name == "kitae");
  document.write("<br>");
  document.write(name > "park");
</script>
</body>
</html>
```





제어문법

if, if-else

44

□ if, if-else 문

```
if(조건식) {  
    ... 실행문 ... // 조건식이 참인 경우  
}
```

```
if(a > b) {  
    document.write("a가 크다");  
}
```

```
if(조건식) {  
    ... 실행문1 ... // 조건식이 참인 경우  
}  
else {  
    ... 실행문2 ... // 조건식이 거짓인 경우  
}
```

```
if(a > b) {  
    document.write("a가 크다");  
}  
else {  
    document.write("a가 크지 않다");  
}
```

```
if(조건식1) {  
    실행문1 // 조건식1이 참인 경우  
}  
else if(조건식2) {  
    실행문2 // 조건식2가 참인 경우  
}  
.....  
else {  
    실행문n; // 앞의 모든 조건이 거짓인 경우  
}
```

```
if(a > b) {  
    document.write("a가 크다");  
}  
else if(a < b) {  
    document.write("b가 크다");  
}  
else  
    document.write("a와 b는 같다");
```

if-else 사용

45

```
<!DOCTYPE html>
<html>
<head> <title> if-else </title> </head>
<body>
<h3> if-else를 이용한 학점 매기기 </h3>
<hr>
<script>
  var grade;
  var score = prompt("황기태 님 점수를 입력하세요", 100);
  score = parseInt(score); // 문자열을 숫자로 바꿈
  if(score >= 90) // score가 90 이상
    grade = "A";
  else if(score >= 80) // 80 이상 90 미만
    grade = "B";
  else if(score >= 70) // 70 이상 80 미만
    grade = "C";
  else if(score >= 60) // 60 이상 70 미만
    grade = "D";
  else // 60 미만
    grade = "F";
  document.write(score + "는 " + grade + "입니다.<br>")
</script>
</body>
</html>
```

localhost 내용:

황기태 님 점수를 입력하세요

확인

취소

if-else를 이용한 학점 매기기

95는 A입니다.

switch 문

46

□ switch 문

- ▣ 값에 따라 서로 다른 코드를 실행할 때, switch 문 적합

```
switch(식) {  
  case 값1: // 식의 결과가 값1과 같을 때  
    실행 문장 1;  
    break;  
  case 값2: // 식의 결과가 값2와 같을 때  
    실행 문장 2;  
    break;  
  ...  
  case 값m:  
    실행 문장 m; // 식의 결과가 값과 같을 때  
    break;  
  default: // 어느 값과도 같지 않을 때  
    실행 문장 n;  
}
```

```
var fruits="사과";  
switch(fruits) {  
  case "바나나":  
    price = 200; break;  
  case "사과":  
    price = 300; break;  
  case "체리":  
    price = 400; break;  
  default:  
    document.write("팔지 않습니다.");  
    price = 0;  
}
```

// switch 문의 실행 결과 price=300

case 문의 '값'

47

▣ case 문의 '값'은 상수(리터럴)만 가능

■ 잘 작성된 case 문

```
case 1 :  
case 2.7 :  
case "Seoul" :  
case true :  
case 2+3 : // 2+3은 먼저 5로 계산되어 case 5:와 동일
```

▣ case 문의 '값'에 변수나 식은 사용 불가

■ 잘못 작성된 case 문

```
case a :           // 오류. 변수 a 사용 불가  
case a -> 3:       // 오류. 식(a>3) 사용 불가
```

switch 문에서 break 문의 역할

48

□ break 문

▣ switch 문 종료

- break; 문을 만날 때까지 아래로 코드 계속 실행

```
var city="Seoul";  
switch(city) {  
  case "Seoul":  
    document.write("서울");  
    break;  
  case "NewYork":  
    document.write("뉴욕");  
    break;  
  case "Paris":  
    document.write("파리");  
    break;  
}
```

서울뉴욕

(a) break;를 만날 때까지 아래로 실행을 계속하는 사례

```
var day="월";  
switch(day) {  
  case "월":  
  case "화":  
  case "수":  
  case "목":  
    document.write("정상영업");  
    break;  
  case "토":  
  case "일": document.write("휴일");  
    break;  
}
```

정상영업

(b) 여러 case에 대해 동일한 코드를 실행하도록 의도적으로 break;를 생략한 경우

switch 문 사용

49

```
<!DOCTYPE html>
<html>
<head> <title>switch</title> </head>
<body>
<h3>switch 문으로 커피 주문</h3>
<hr>
<script>
  var price = 0;
  var coffee = prompt("무슨 커피 드릴까요?", "");
  switch(coffee) {
    case "espresso" :
    case "에스프레소" : price = 2000;
      break;
    case "카푸치노" : price = 3000;
      break;
    case "카페라떼" : price = 3500;
      break;
    default :
      document.write(coffee + "는 없습니다.");
  }
  if(price != 0)
    document.write(coffee + "는 " + price + "원입니다.");
</script>
</body>
</html>
```

"espresso"나
"에스프레소"의 경우
모두 실행

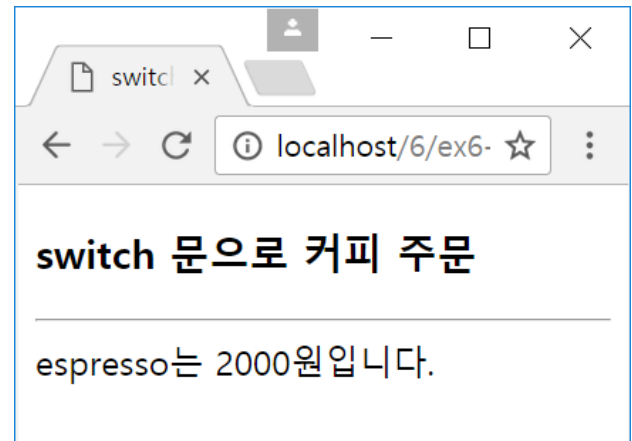
localhost 내용:

무슨 커피 드릴까요?

espresso

확인

취소



반복문

50

□ for 문

```
for(초기문; 조건식; 반복 후 작업) {  
    ... 작업문 ...  
}
```

```
// 0에서 9까지 출력  
for(var i=0; i<10; i++) {  
    document.write(i);  
}
```

0123456789

□ while 문

```
while(조건식) {  
    ... 작업문 ...  
}
```

```
var i=0;  
while(i<10) { // i가 0에서 9까지 출력  
    document.write(i);  
    i++;  
}
```

0123456789

□ do-while 문

```
do {  
    ... 작업문 ...  
} while(조건식);
```

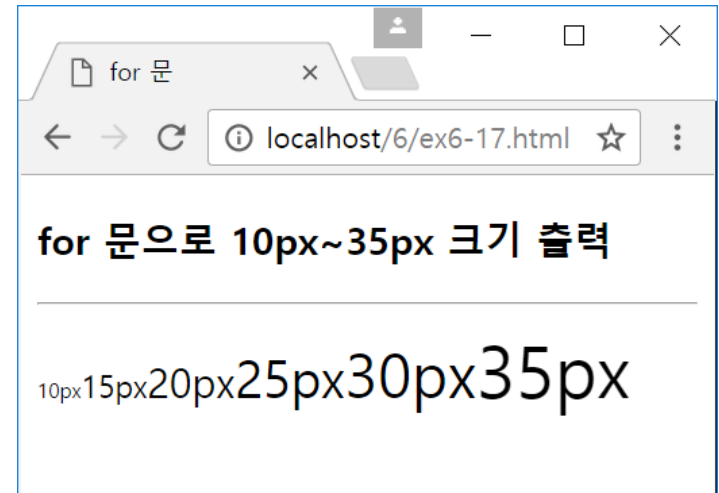
```
var i=0;  
do { // i가 0에서 9까지 출력  
    document.write(i);  
    i++;  
} while(i<10);
```

0123456789

for 문으로 10px~35px 크기로 출력

51

```
<!DOCTYPE html>
<html>
<head>
<title>for 문</title>
</head>
<body>
<h3>for 문으로 10px~35px 크기 출력</h3>
<hr>
<script>
  for(var size=10; size<=35; size+=5) { // 5씩 증가
    document.write("<span ");
    document.write("style='font-size:" + size + "px'>");
    document.write(size + "px");
    document.write("</span>");
  }
</script>
</body>
</html>
```



while 문으로 0~n까지의 합 구하기

52

```
<!DOCTYPE html>
<html>
<head>
<title>while 문</title>
</head>
<body>
<h3>while 문으로 0에서 n까지 합</h3>
<hr>
<script>
  var n = prompt("0보다 큰 정수를 입력하세요", 0);
  n = parseInt(n); // 문자열 n을 숫자로 바꿈

  var i=0, sum=0;
  while(i<=n) { // i가 0에서 n까지 반복
    sum += i;
    i++;
  }
  document.write("0에서 " + n + "까지 합은 " + sum);
</script>
</body>
</html>
```

prompt()가 리턴한 것은 문자열

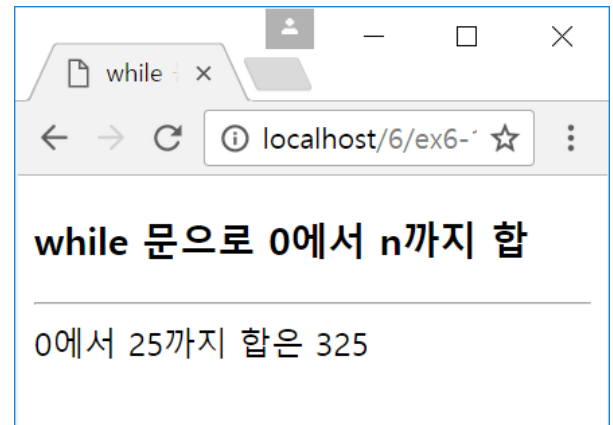
localhost 내용:

0보다 큰 정수를 입력하세요

25

확인

취소



do-while 문으로 0~n까지 합 구하기

53

```
<!DOCTYPE html>
<html>
<head>
<title>do-while 문</title>
</head>
<body>
<h3>do-while 문으로 0에서 n까지 합</h3>
<hr>
<script>
  var n = prompt("0보다 큰 정수를 입력하세요", 0);
  n = parseInt(n); // 문자열 n을 숫자로 바꿈

  var i=0, sum=0;
  do {
    sum += i;
    i++;
  } while(i<=n); // i가 0~n까지 반복
  document.write("0에서 " + n + "까지 합은 " + sum);
</script>
</body>
</html>
```

prompt()가 리턴한 것은 문자열

localhost 내용:

0보다 큰 정수를 입력하세요

25

확인

취소

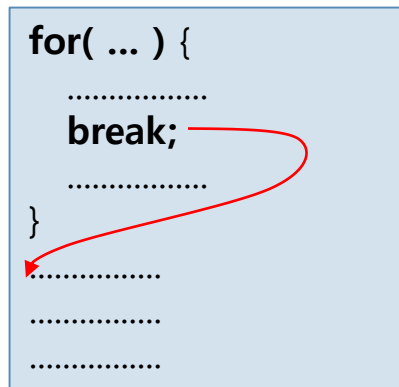
do-while 문으로 0에서 n까지 합

0에서 25까지 합은 325

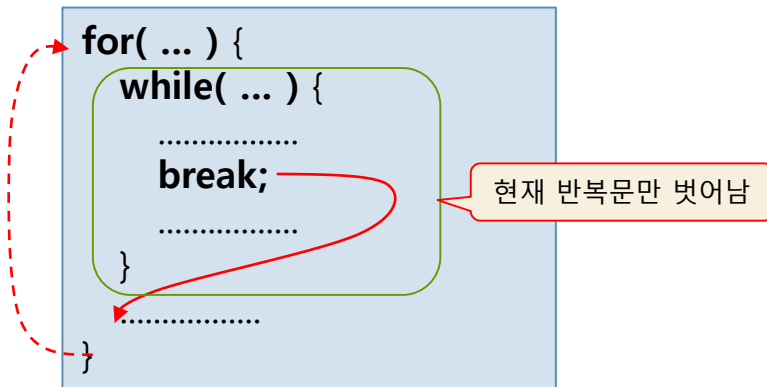
반복문 내의 break 문과 continue 문

54

- break 문 : 가장 안쪽 반복문 하나만 벗어나도록 제어

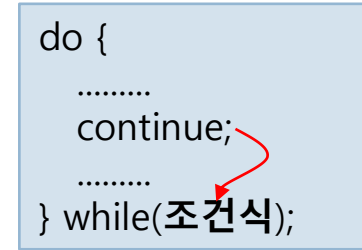
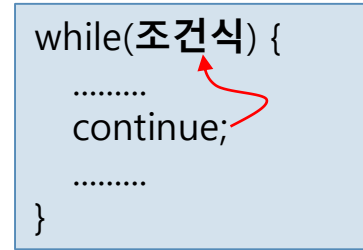
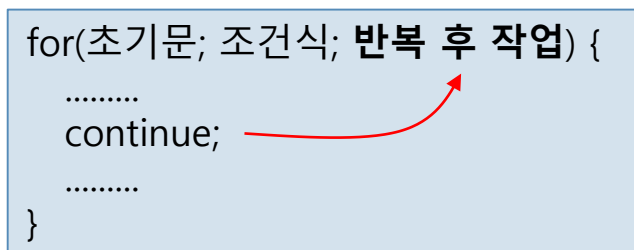


(a) 반복문 벗어나기



(b) 중첩 반복에서 현재 반복문만 벗어나

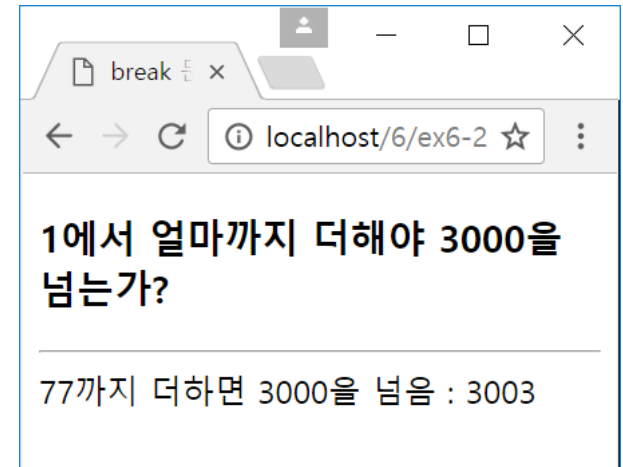
- continue 문 : 반복 코드 실행 중단, 다음 반복으로 점프



break 문 활용

55

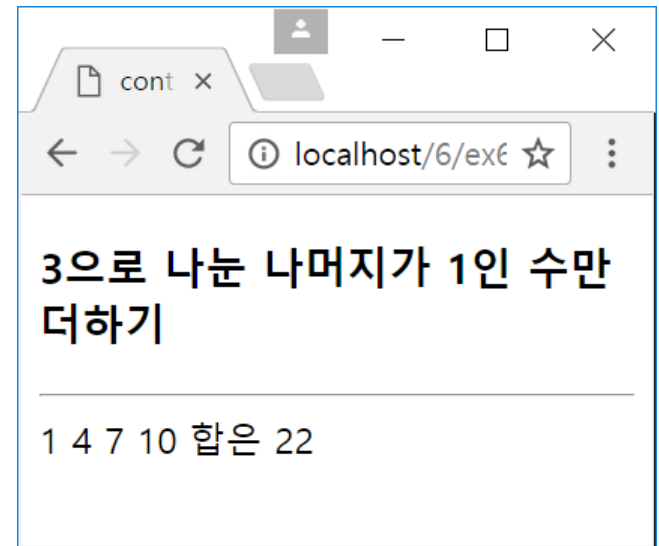
```
<!DOCTYPE html>
<html>
<head>
<title>break 문</title>
</head>
<body>
<h3>1에서 얼마까지 더해야 3000을 넘는가?</h3>
<hr>
<script>
  var i=0, sum=0;
  while(true) { // 무한 반복
    sum += i;
    if(sum > 3000)
      break; // 합이 3000보다 큼. 반복문 벗어남
    i++;
  }
  document.write(i + "까지 더하면 3000을 넘음 : " + sum);
</script>
</body>
</html>
```



continue 문 활용

56

```
<!DOCTYPE html>
<html>
<head>
<title>continue 문</title>
</head>
<body>
<h3>3으로 나눈 나머지가 1인 수만 더하기</h3>
<hr>
<script>
  var sum=0;
  for(i=1; i<=10; i++) { // i가 1에서 10까지 반복
    if(i%3 != 1) // 3으로 나눈 나머지가 1이 아닌 경우
      continue; // 다음 반복으로 점프(i++ 코드로)
    document.write(i + " ");
    sum += i;
  }
  document.write("합은 " + sum);
</script>
</body>
</html>
```



함수와 내장함수

- 함수란?
 - ▣ 목적을 가지고 작성된 코드 블록
 - ▣ 데이터 전달받아 처리한 후 결과를 돌려주는 코드 블록
- 함수 개념



함수의 구성과 호출

59

□ 함수의 구성

```
function 함수이름(arg1, arg2,..., argn) {  
    ...프로그램 코드...  
    결과를 리턴하는 return 문  
}
```

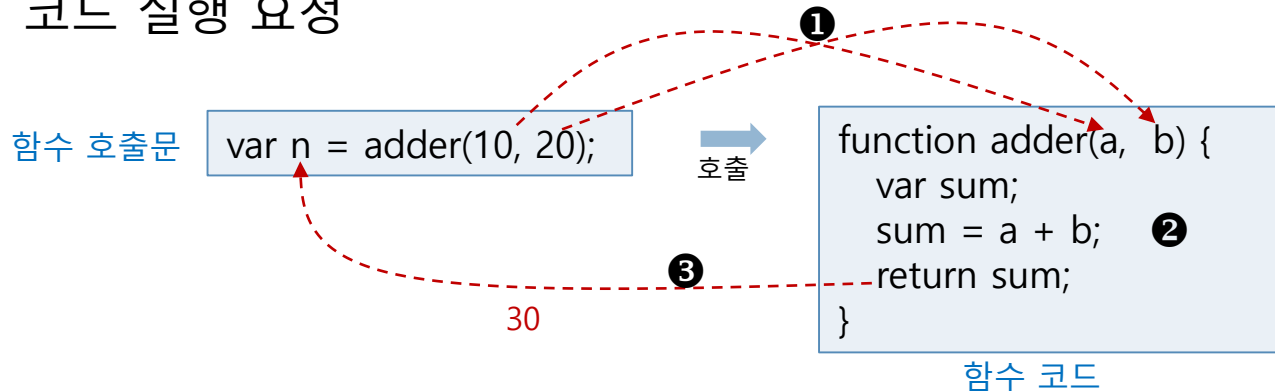
함수 선언 함수 이름 매개 변수

```
function adder ( a, b ) {  
    var sum;  
    sum = a + b;  
    return sum; // 덧셈 합 리턴  
}
```

반환 키워드 반환 값

□ 함수 호출

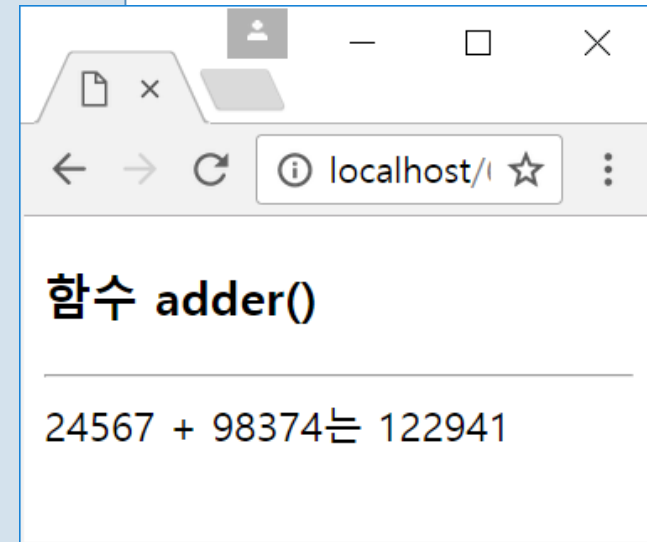
▣ 함수의 코드 실행 요청



adder() 함수 작성 및 호출

60

```
<!DOCTYPE html>
<html>
<head>
<title>함수</title>
<script>
function adder(a, b) { // 함수 작성
    var sum;
    sum = a + b;
    return sum;
}
</script>
</head>
<body>
<h3>함수 adder()</h3>
<hr>
<script>
    var n = adder(24567, 98374); // 함수 호출
    document.write("24567 + 98374는 " + n + "<br>");
</script>
</body>
</html>
```



자바스크립트에서 제공하는 전역 함수

61

□ 대표적인 자바스크립트 함수

▣ eval() 함수

예) `var res = eval("2*3+ 4*6");` // res는 30

▣ parseInt() 함수

예) `var l = parseInt("32");` // "32"를 10진수로 변환, 정수 32 리턴

`var n = parseInt("0x32");` // "0x32"를 16진수로 해석, 정수 50 리턴

▣ isNaN() 함수

예) `isNaN(32)` // false 리턴

`isNaN("32")` // false 리턴

`isNaN("hello")` // true 리턴

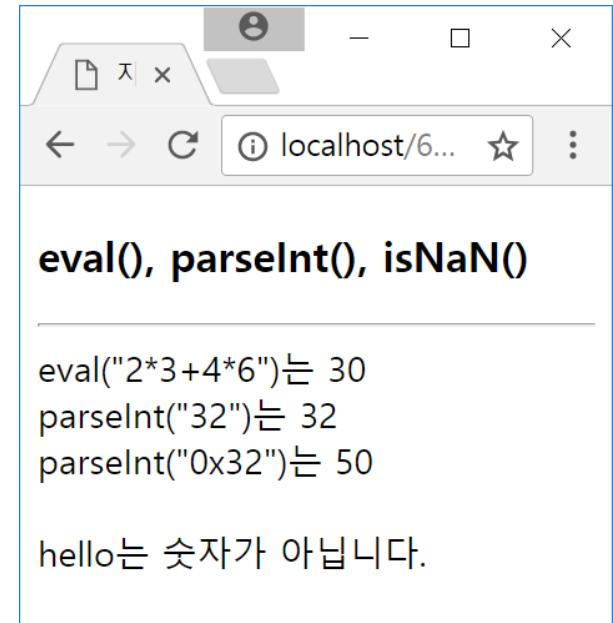
전역 함수명	설명
<code>eval(exp)</code>	exp의 자바스크립트 식을 계산하고 결과 리턴
<code>parseInt(str)</code>	str 문자열을 10진 정수로 변환하여 리턴
<code>parseInt(str, radix)</code>	str 문자열을 radix 진수로 해석하고, 10진 정수로 바꾸어 리턴
<code>parseFloat(str)</code>	str 문자열을 실수로 바꾸어 리턴
<code>isFinite(value)</code>	value가 숫자이면 true 리턴
<code>isNaN(value)</code>	value가 숫자가 아니면 true 리턴

eval(), parseInt(), isNaN()

62

```
<!DOCTYPE html>
<html>
<head>
<title>자바스크립트 전역함수</title>
<script>
function evalParseIntIsNaN() {
  var res = eval("2*3+4*6"); // res는 30
  document.write("eval(W"2*3+4*6W")는 " + res + "<br>");
  var m = parseInt("32");
  document.write("parseInt(W"32W")는 " + m + "<br>");
  var n = parseInt("0x32");
  document.write("parseInt(W"0x32W")는 " + n + "<br><br>");

  // "hello"는 정수로 변환할 수 없으므로 parseInt("hello")는 NaN 리턴
  n = parseInt("hello");
  if(isNaN(n)) // true
    document.write("hello는 숫자가 아닙니다.");
}
</script>
</head>
<body>
<h3>eval(), parseInt(), isNaN()</h3>
<hr>
<script>
  evalParseIntIsNaN();
</script>
</body>
</html>
```



구구단 출력 함수 만들기

63

```
<!DOCTYPE html>
<html> <head> <title> 함수 만들기 </title>
<script>
function gugudan(n) { // 함수 작성
    var m = parseInt(n); // 문자열 n을 숫자로 바꿈
    if(isNaN(m) || m < 1 || m > 9) {
        alert("잘못입력하셨습니다.");
        return;
    }
    for(var i=1; i<=9; i++) { // i는 1~9까지 반복
        document.write(m + "x" + i + "=" + m*i + "<br>");
    }
}
</script>
</head>
<body>
<h3>구구단 출력 함수 만들기</h3>
<hr>
<script>
    var n = prompt("구구단 몇 단을 원하세요", ""); // n은 문자열
    gugudan(n); // 함수 호출
</script>
</body>
</html>
```

n이 1~9사이의 숫자
가 아닌 경우 처리

localhost 내용:

구구단 몇 단을 원하세요

6

확인

취소

구구단 출력 함수 만들기

6x1=6
6x2=12
6x3=18
6x4=24
6x5=30
6x6=36
6x7=42
6x8=48
6x9=54