L6-javascript

예제 실습 및 보고서 작성



학 과 : 컴퓨터공학전공

과 목 : 웹프로그래밍

교수님 : 한영선 교수님

학 번 : 202030494

이 름 : 윤원재

제출일 : 2023년 04월 16일

1. **예제 6-1 HTML 이벤트 리스너에 자바스크립트 코드 작성**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**이벤트 리스너 속성에 자바스크립트 코드**</title>**

**</head>**

**<body>**

**<h3>**마우스 올려 보세요**</h3>**

**<hr>**

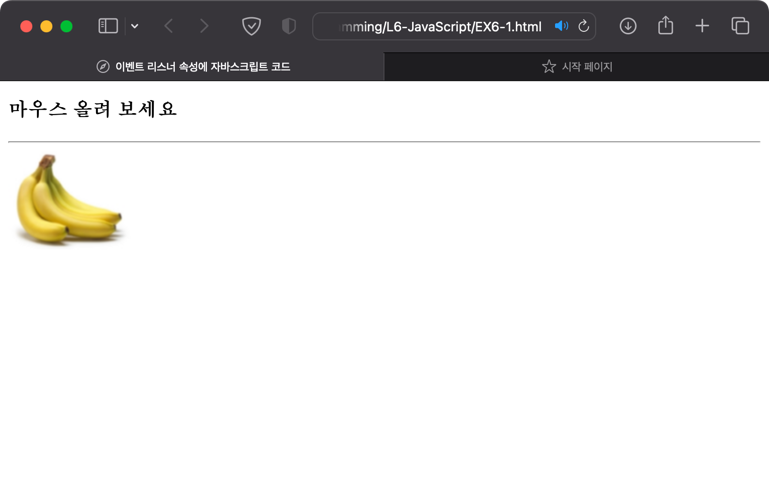
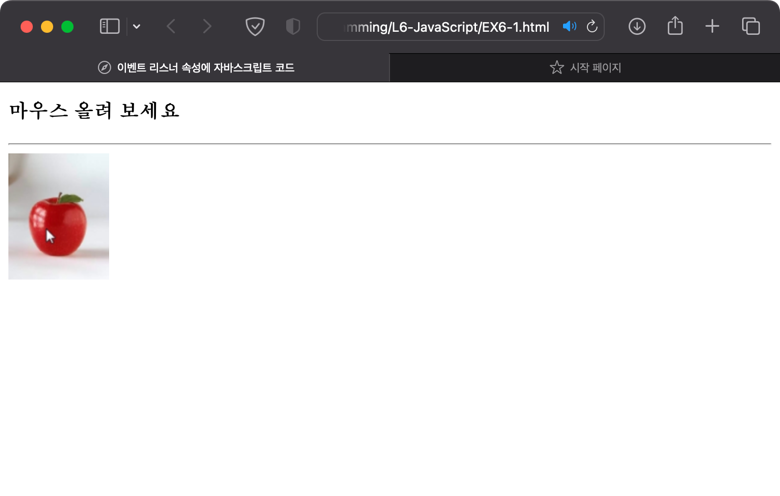
**<img src="media/apple.png" alt="이미지"**

**onmouseover="this.src='media/banana.png'"**

**onmouseout="this.src='media/apple.png'">**

**</body>**

**</html>**

****

<그림 1-1> 예제 6-1 결과

HTML에 자바스크립트 코드를 적용하기 위한 방법으로 HTML 태그의 이벤트 리스너 속성에 자바스크립트 코드를 작성하는 방법이 있다. <그림 1-1>에서 <img>태그에 ‘onmouseover’와 ‘onmouseout’ 이벤트 리스너 속성을 사용하여 자바스크립트 코드를 적용하였다. 이때 자바스크립트 코드의 ‘this.src’에서 ‘this’는 <img>태그를 의미하며 해당 이미지 위에 마우스를 올리면 바나나 이미지로 바뀌고 다시 마우스를 치우면 사과 이미지로 바뀌는 예제이다.

1. **예제 6-2 <script>태그에 자바스크립트 코드 작성**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<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>**

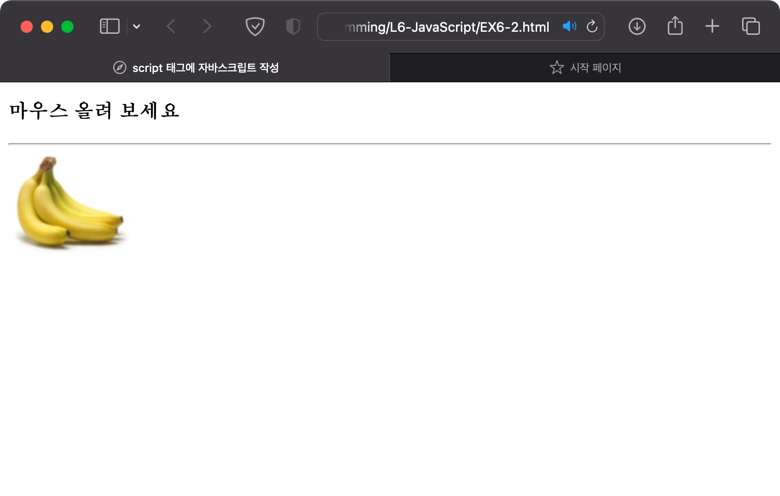
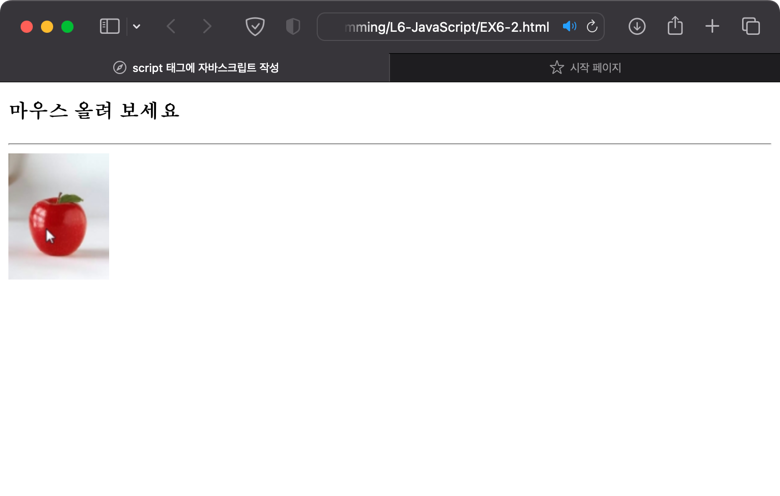
**<img src="media/apple.png" alt="이미지"**

**onmouseover="over(this)"**

**onmouseout="out(this)">**

**</body>**

**</html>**

****

<그림 2-1> 예제 6-2 결과

HTML에 자바스크립트 코드를 적용시키기 위한 두번째 방법으로 <script></script>태그에 작성하는 방법이 있다. <그림 2-1>에서 <script></script>태그 내에 자바스크립트 함수를 정의하고 <img>태그에서 이벤트 리스너 속성으로 over()함수와 out()함수를 호출하여 사용하였다. 함수와 관련된 자세한 내용은 ‘예제 6-22’에서 소개한다.

1. **예제 6-3 자바스크립트 파일 작성 및 불러오기**

function over(obj){

obj.src = "media/banana.png";

}

function out(obj){

obj.src = "media/apple.png";

}

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**외부 파일에 자바스크립트 작성**</title>**

**<script src="script.js"></script>**

**</head>**

**<body>**

**<h3>**마우스 올려 보세요**</h3>**

**<hr>**

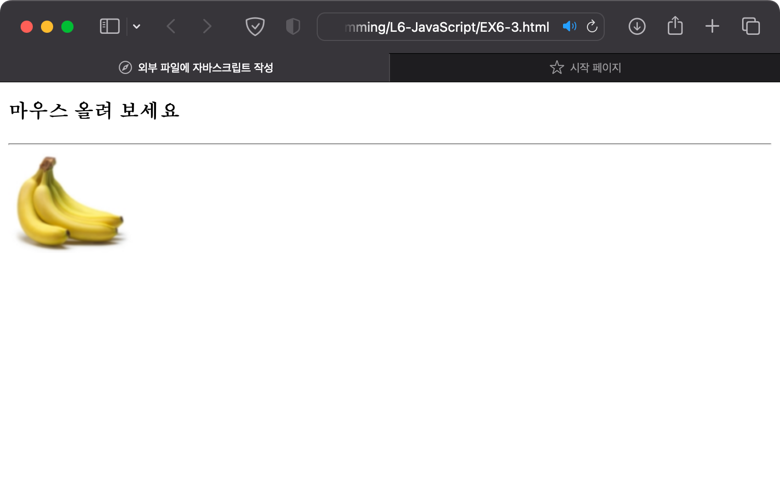
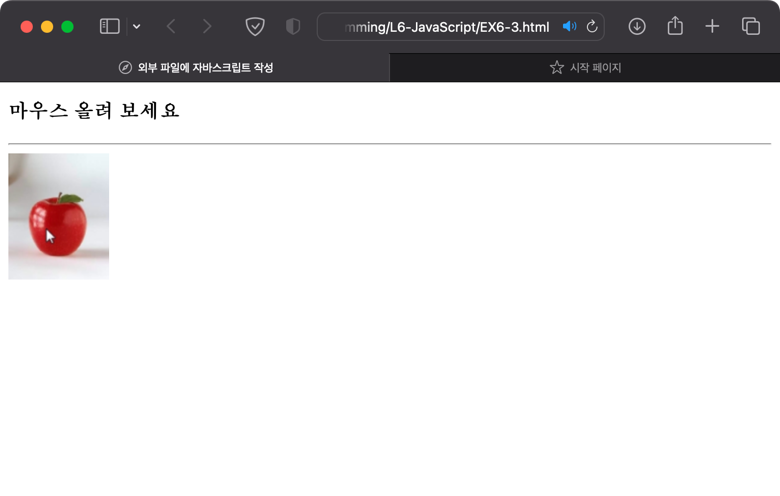
**<img src="media/apple.png" alt="이미지"**

**onmouseover="over(this)"**

**onmouseout="out(this)">**

**</body>**

**</html>**

****

<그림 3-1> 예제 6-3 결과

HTML에 자바스크립트 코드를 적용시키기 위한 세번째 방법으로 별도의 자바스크립트 코드 파일에 작성한 다음 해당 파일을 불러와서 사용하는 방법이 있다. <그림 3-1>에서 별도의 ‘script.js’ 코드 파일을 작성 후 이를 <script>태그에서 ‘src’속성을 사용하여 불러왔다.

단, 이렇게 사용할 경우 <script></script>태그에 추가적인 자바스크립트 코드를 작성해선 안된다.

1. **예제 6-4 링크의 href에 자바스크립트 코드 작성**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**URL에 자바스크립트 작성**</title>**

**</head>**

**<body>**

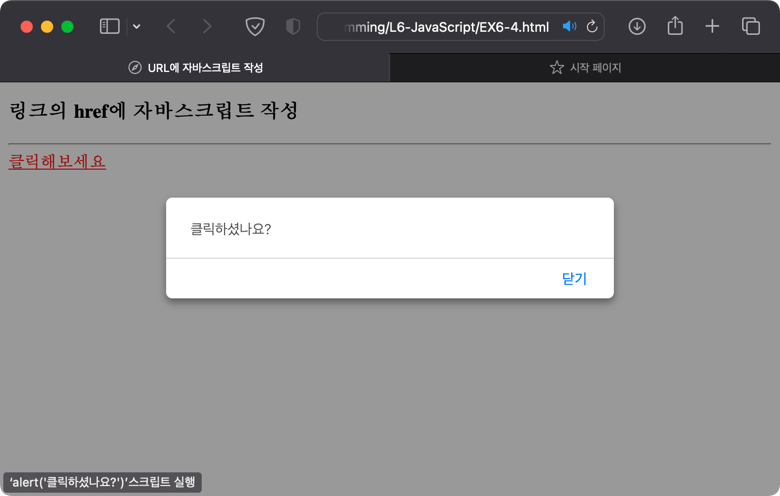
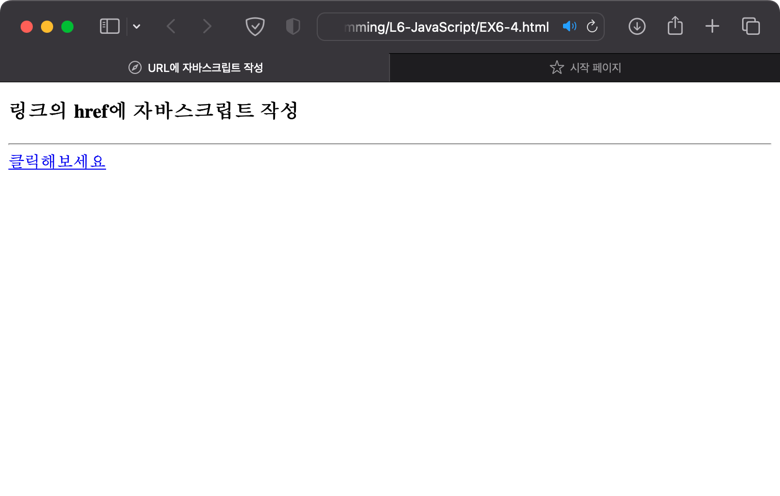
**<h3>**링크의 href에 자바스크립트 작성**</h3>**

**<hr>**

**<a href="javascript:alert('클릭하셨나요?')">**클릭해보세요**</a>**

**</body>**

**</html>**

****

<그림 4-1> 예제 6-4 결과

HTML에 자바스크립트 코드를 적용시키기 위한 네번째 방법으로 하이퍼링크의 ‘href’ 속성에 자바스크립트코드를 작성하는 방법이 있다. <그림 4-1>에서 하이퍼링크 태그인 <a>태그의 ‘href’ 속성에 ‘javascript:alert('클릭하셨나요?')’로 자바스크립트의 경고 다이얼로그를 작성하였다. ‘클릭해보세요’를 누르면 ‘클릭하셨나요?’라는 문구와 함께 경고창이 뜨게 된다.

1. **예제 6-5 document.write()로 웹 페이지에 HTML 콘텐츠 출력**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**document.write() 활용**</title>**

**</head>**

**<body>**

**<h3>**document.write() 활용**</h3>**

**<hr>**

**<script>**

document.write("<h3>Welcome!</h3>");

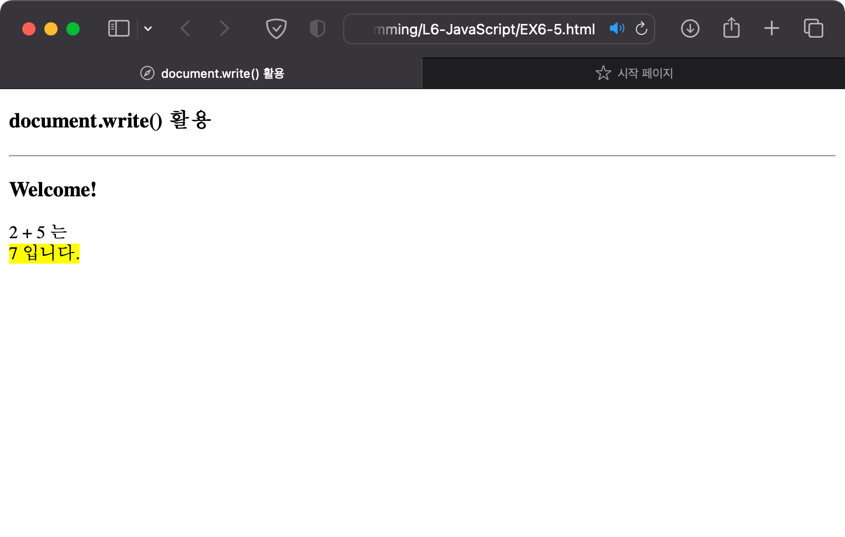
document.write("2 + 5 는 <br>");

document.write("<mark>7 입니다.</mark>");

**</script>**

**</body>**

**</html>**

****

<그림 5-1> 예제 6-5 결과

자바스크립트로 HTML 콘텐츠를 출력할 수 있다. ‘document.write()’를 사용하여 괄호 안에 HTML 콘텐츠를 작성하면 해당 HTML 콘텐츠가 화면에 출력된다. ‘document.writeln()’을 사용할 수도 있는데 작성한 HTML 콘텐츠 뒤에 빈칸을 추가해주는 기능을 수행한다.

<그림 5-1>에서 <h3>태그, <br>태그 및 <mark>태그를 포함한 텍스트가 ‘document.write()’를 통하여 잘 출력되는 것을 확인할 수 있다.

1. **예제 6-6 지역변수와 전역변수**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**지역변수와 전역변수**</title>**

**</head>**

**<body>**

**<h3>**지역변수와 전역변수**</h3>**

**<hr>**

**<script>**

var x=100; // 전역변수

function f() { // 함수 f() 선언

var x=1; // 지역변수

document.write("지역변수 x=" + x);

document.write("<br>");

document.write("전역변수 x=" + this.x);

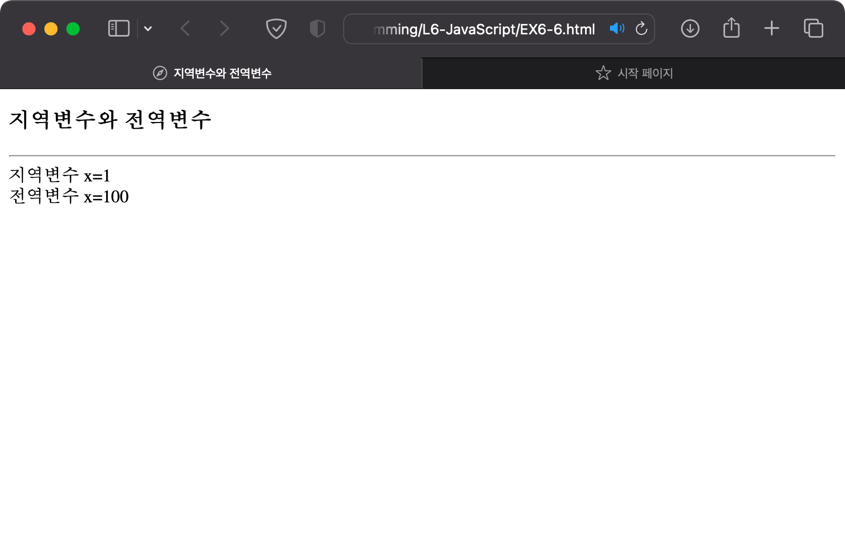
}

f(); // 함수 f() 호출

**</script>**

**</body>**

**</html>**

****

<그림 6-1> 예제 6-6 결과

자바스크립트에서 변수를 선언하여 사용할 수 있으며 변수 타입이 존재하지 않아 모든 변수는 ’var’ 키워드로 선언하여 사용할 수 있다. 따라서 저장되는 값에 대한 제약이 없다. 자바스크립트는 C언어의 문법과 매우 유사 지역변수와 전역변수의 차이를 두고있다. 함수 내에서 ‘var’ 키워드로 선언된 변수는 지역변수로 사용되며, 함수 밖에서 선언되거나 함수 내에서 ‘var’ 키워드 없이 선언된 변수는 전역변수로 사용된다. 만일 지역변수와 전역변수의 이름이 같을 경우 전역변수에 접근하기 위해선 ‘this.전역변수’의 형식으로 접근이 가능하다.

<그림 6-1>에서 x라는 이름으로 지역변수와 전역변수가 생성되었다. 전역변수에는 100의 값을 넣고 지역변수에는 1의 값을 넣었다. 이후 지역변수를 x로 호출하였고 전역변수를 this.x로 호출한 것을 확인할 수 있다.

1. **예제 6-7 상수**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**상수**</title>**

**</head>**

**<body>**

**<h3>**상수**</h3>**

**<hr>**

**<script>**

var oct = 015;

var hex = 0x15;

var condition = true;

document.write("8진수 015는 십진수로 " + oct + "<br>");

document.write("16진수 0x15는 십진수로 " + hex + "<br>");

document.write("condition은 " + condition + "<br>");

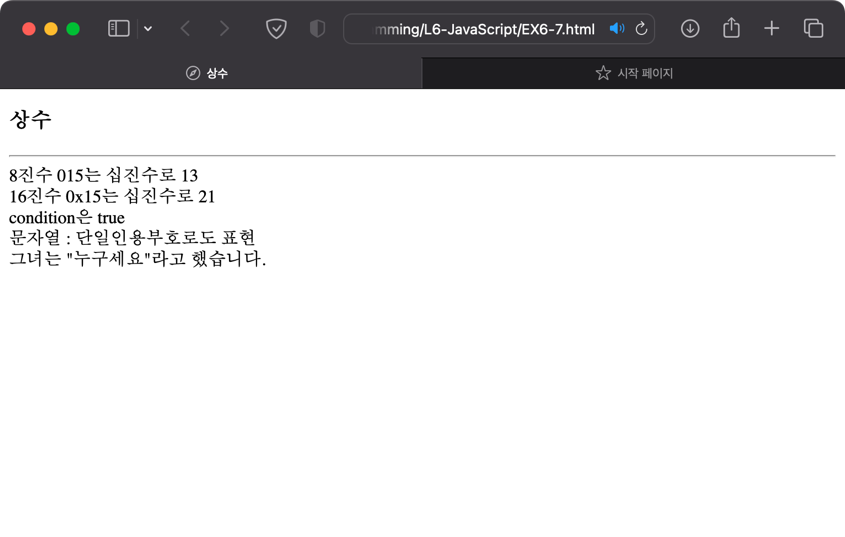
document.write('문자열 : 단일인용부호로도 표현' + "<br>");

document.write("그녀는 \"누구세요\"라고 했습니다." + "<br>");

**</script>**

**</body>**

**</html>**

****

<그림 7-1> 예제 6-7 결과

자바스크립트에서 리터럴을 표기하는 방법은 C언어의 문법과 유사하다. 리터럴의 종류로 정수, 실수, 논리, 문자열 등이 있다. 정수는 8진수, 10진수, 16진수로 나뉘며 ‘0’으로 시작하면 8진수, ‘0x’로 시작하면 16진수이다. 실수는 소수형과 지수형으로 나뉘며 지수형은 ‘E‘를 사용하여 표현한다. 논리는 참, 거짓으로 나뉘며 ‘true’, ‘false’로 표현한다. 자바스크립트에는 문자 타입 없이 문자는 문자열로 표현되며 문자열은 “”나 ‘’로 묶어 표현한다. 그 외 값이 없다는 것을 의미하는 null과 수가 아님을 뜻하는 NaN이 존재한다.

<그림 7-1>에서 “를 문자 그대로 표현하기 위해 \”로 사용한 것을 확인할 수 있다.

1. **예제 6-8 산술 연산**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<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>");

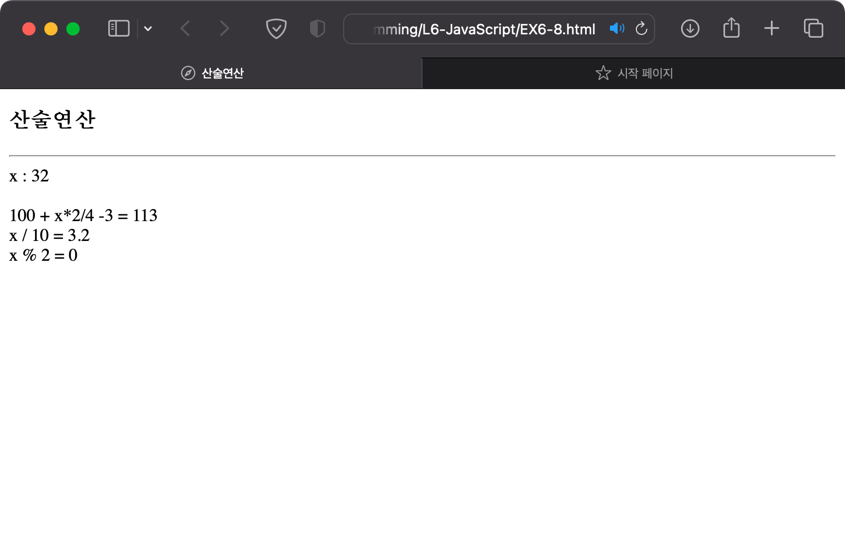
document.write("x / 10 = " + div + "<br>");

document.write("x % 2 = " + mod + "<br>");

**</script>**

**</body>**

**</html>**

****

<그림 8-1> 예제 6-8 결과

자바스크립트에서 산술 연산을 수행할 수 있다. 산술 연산자로 +, -, \*, /, %가 있으며 각각 덧셈, 뺄셈, 곱셈, 나눗셈, 나머지 연산을 수행하며 연산의 결과는 항상 실수이다. <그림 8-1>에서 산술 연산자를 사용한 예를 보이고있다.

1. **예제 6-9 대입 연산**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<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 + ", z=" + z + "<br><br>");

x += 3;

y \*= 3;

z %= 2;

document.write("x += 3; 실행 후, x=" + x + "<br>");

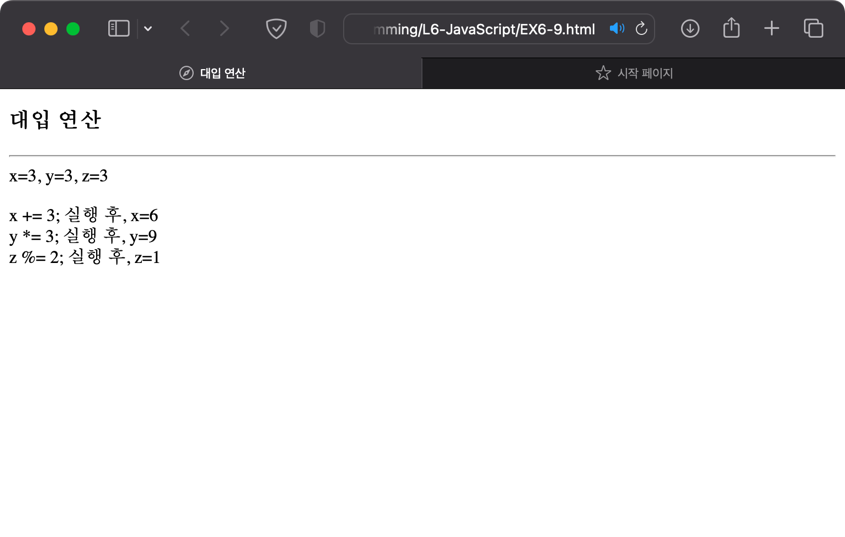
document.write("y \*= 3; 실행 후, y=" + y + "<br>");

document.write("z %= 2; 실행 후, z=" + z);

**</script>**

**</body>**

**</html>**

****

<그림 9-1> 예제 6-9 결과

자바스크립트는 대입 연산을 지원한다. 대입 연산이란 오른쪽 식의 결과를 왼쪽 변수에 대입하는 것을 의미한다. 대입 연산자는 ‘=’을 포함하여 ‘+=’, ‘-=’, ‘\*=’, ‘/=’, ‘%=’, ‘&=’, ‘^=’, ‘|=’, ‘<<=’, ‘>>=’, ‘>>>=’가 있다. ‘=’는 오른쪽 값을 왼쪽 변수에 대입하는 것을 의미하며 나머지 대입 연산자는 왼쪽 변수의 값과 오른쪽 값을 ‘=’기호 옆에 있는 연산자의 연산을 수행하여 왼쪽 변수에 대입하는 것을 의미한다. <그림 9-1>에서 대입 연산을 수행하고 그에 따른 결과를 출력하는 것을 확인할 수 있다.

1. **예제 6-10 비교 연산**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<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>");

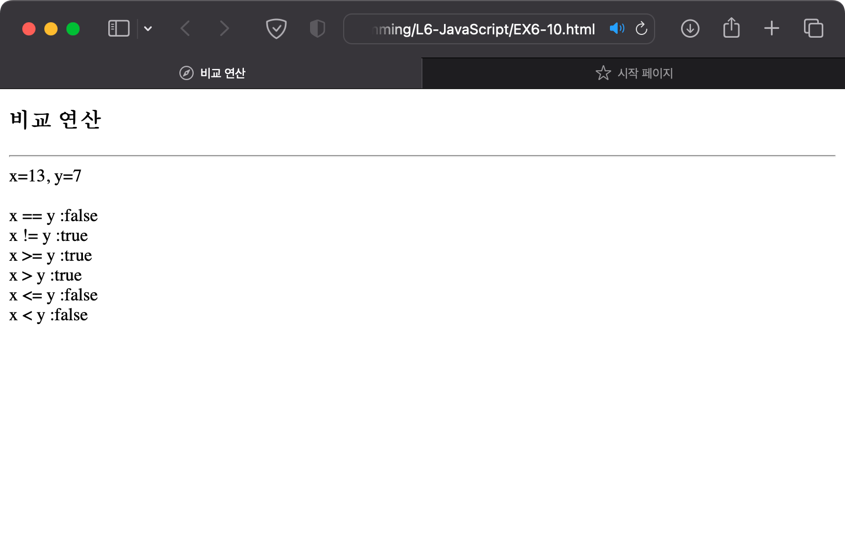
document.write("x <= y :" + (x <= y) + "<br>");

document.write("x < y :" + (x < y) + "<br>");

**</script>**

**</body>**

**</html>**

****

<그림 10-1> 예제 6-10 결과

비교 연산이란 두 값을 주어진 조건에 따라 비교하여 그 결과가 참이면 true를, 거짓이면 false의 결과를 내는 연산이다. 비교 연산자로 ‘<’ , ‘>’, ‘<=’, ‘>=’, ‘==’, ‘!=’가 있으며 왼쪽을 기준으로 각각 미만, 초과, 이하, 이상, 같음, 다름을 의미한다. <그림 10-1>에서 비교 연산자를 사용하여 비교를 하고 출력을 보이고 있다.

1. **예제 6-11 논리 연산**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<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>");

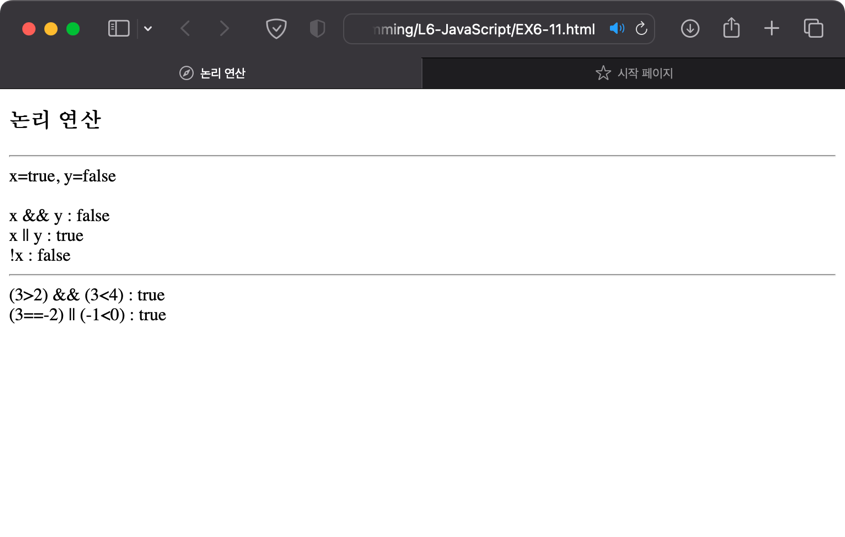
document.write("(3>2) && (3<4) : " + ((3>2)&&(3<4)) + "<br>");

document.write("(3==-2) || (-1<0) : " + ((3==-1)||(-1<0)) + "<br>");

**</script>**

**</body>**

**</html>**

****

<그림 11-1> 예제 6-11 결과

논리 연산이란 AND, OR, NOT연산을 수행하는 것을 의미한다. AND연산은 두 값이 모두 true일 경우 true를 반환한다. OR연산은 두 값 중 하나라도 true일 경우 true를 반환한다. NOT연산은 현재 상태의 반대를 반환한다. 예를 들어 true일 경우 false를 반환한다. AND연산은 ‘&&’를, OR연산은 ‘||’를, NOT연산은 ‘!’를 사용하여 표현한다. <그림 6-1>은 논리 연산을 비교 연산과 함께 수행한 결과를 보이고 있다.

1. **예제 6-12 조건 연산**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**조건 연산**</title>**

**</head>**

**<body>**

**<h3>**조건 연산**</h3>**

**<hr>**

**<script>**

var a=3, b=5;

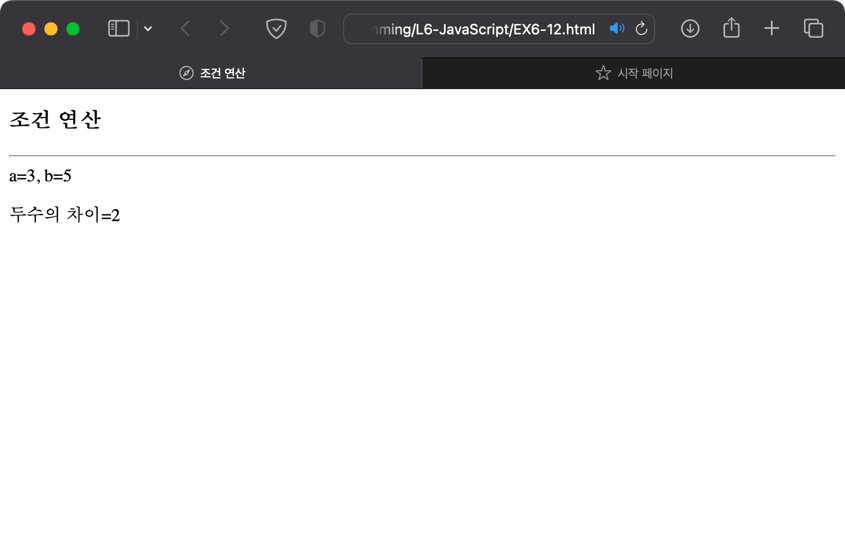
document.write("a=" + a + ", b=" + b + "<br><br>");

document.write("두수의 차이=" + ((a>b)?(a-b):(b-a)));

**</script>**

**</body>**

**</html>**

****

<그림 12-1> 예제 6-12 결과

조건 연산이란 조건에 따라 달라지는 결과를 연산자를 사용하여 표현하는 기법이다. 형식은 ‘condition ? ex\_true : ex\_false’로 condition이 true라면 결과는 ex\_true의 값이 되고 false라면 ex\_false의 값이 된다. <그림 12-1>에서 a=3과 b=5를 비교하여 큰 수에서 작은 수를 뺀 출력을 양수로 보이는 것을 확인할 수 있다.

1. **예제 6-13 비트 연산**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**비트 연산**</title>**

**<script>**

function digit8(v) {

var str="";

for(i=0; i<8; i++, v<<=1) {

if((v & 0x80)) str += "1";

else str += "0";

}

return str;

}

**</script>**

**</head>**

**<body>**

**<h3>**비트 연산**</h3>**

**<hr>**

**<script>**

var x=-10, y=3;

document.write("<pre>");

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

document.write("x : " + digit8(x) + "<br>");

document.write("y : " + digit8(y) + "<br><hr>");

document.write("x & y : " + digit8(x&y) + "<br>");

document.write("x | y : " + digit8(x|y) + "<br>");

document.write("x ^ y : " + digit8(x^y) + "<br>");

document.write("~x : " + digit8(~x) + "<br><hr>");

document.write("x << 1 : " + digit8(x<<1) + "(" + (x<<1) + ")" + "<br>");

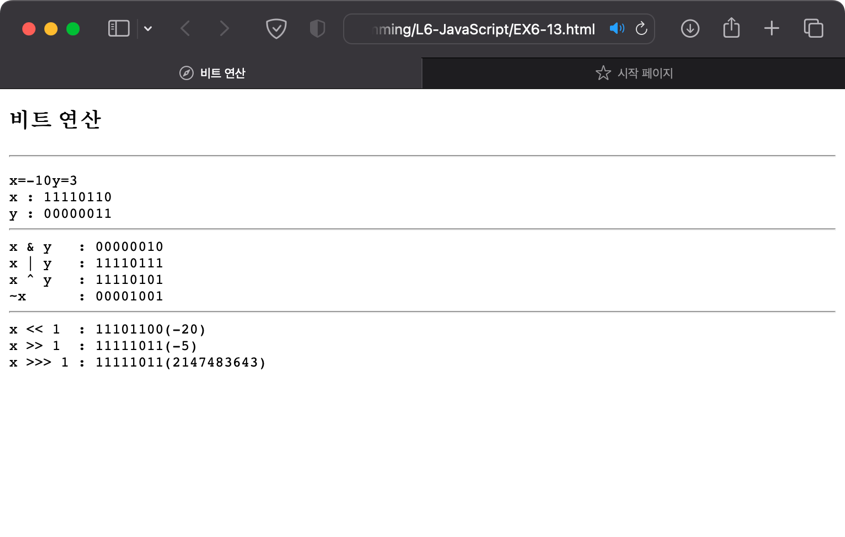
document.write("x >> 1 : " + digit8(x>>1) + "(" + (x>>1) + ")" + "<br>");

document.write("x >>> 1 : " + digit8(x>>>1) + "(" + (x>>>1) + ")");

**</script>**

**</body>**

**</html>**

****

<그림 13-1> 예제 6-13 결과

비트 연산이란 피연산자를 2진수로 바꿔서 비트 단위의 연산을 하는 기법이며 비트 논리 연산과 비트 시프트 연산이 있다. 비트 논리 연산은 앞서 살펴본 논리연산을 비트 단위로 연산하는 것이다. 비트 논리 연산자로 ‘&’, ‘|’, ‘^’, ‘~’가 있으며 각각 AND연산, OR연산, XOR연산, NOT연산을 수행한다. 여기서 XOR연산은 두 비트가 다를 경우 1로, 같으면 0으로 변환한다.

비트 시프트 연산은 비트들의 이동을 하는 연산이다. 연산자는 ‘>>’, ‘<<’, ‘>>>’가 있으며 산술적 오른쪽 시프트, 산술적 왼쪽 시프트, 논리적 오른쪽 시프트를 수행한다. 산술적 오른쪽 시프트는 최상위 비트의 빈자리를 최상위 비트로 채우며 나누기 2를 하는 효과를 발생시킨다. 산술적 왼쪽 시프트는 최하위 비트의 빈자리를 0으로 채우며 곱하기 2를 하는 효과를 발생시킨다. 논리적 오른쪽 시프트는 산술적 오른쪽 시프트와 비슷하지만 최상위 비트를 0으로 채우며 항상 양수의 결과를 얻을 수 있는 효과를 발생시킨다. <그림 13-1>에서 각 비트 연산의 사용을 확인할 수 있다.

1. **예제 6-14 문자열 연산**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<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");

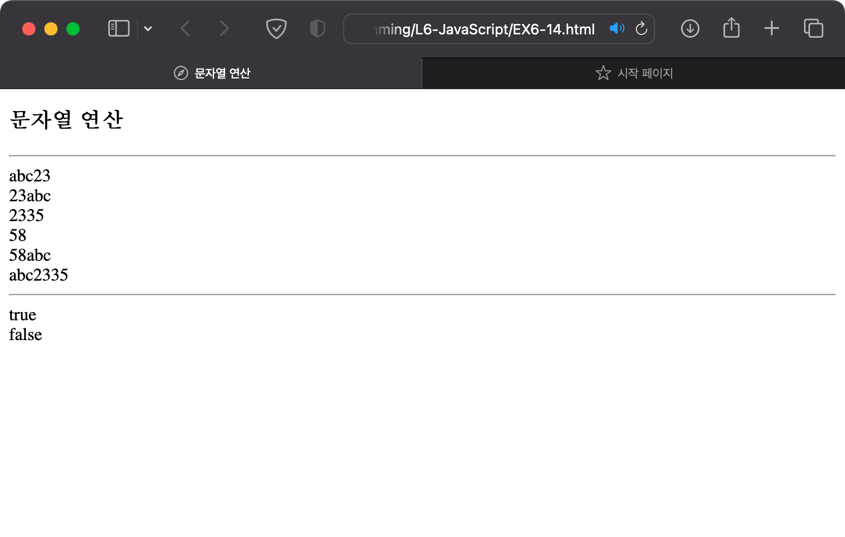
document.write("<br>");

document.write(name > "park");

**</script>**

**</body>**

**</html>**

****

<그림 14-1> 예제 6-14 결과

자바스크립트에서는 문자열 연산을 지원하는데 ‘+’와 ‘+=’를 사용하여 문자열 연결을 할 수 있고 비교 연산자를 사용하여 문자열 비교를 할 수 있다. <그림 14-1>에서 ‘+’을 사용하여 문자열 연결을 한 것을 확인할 수 있다. 이때 정수와 관련하여 순서에 유의해야 하는데, 23 + 25 + “abc”에서 23 + 25를 먼저 계산하여 결과적으로 58 + “abc”가 되어 “58abc”가 도출된다. “abc” + 23 + 25에서는 “abc23” + 25가 되어 “abc2325”가 도출된다.

문자열 비교는 사전 순으로 비교를 하여 먼저 나오는 순서대로 값이 증가한다. 예를 들어 “abc”와 “dcv”를 비교하면 사전 순에 따라 “abc”가 먼저 나오기 때문에 “abc” < “dcv”가 true로 반환된다.

1. **예제 6-15 if-else 사용**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**if-else**</title>**

**</head>**

**<body>**

**<h3>**if-else를 이용한 학점 매기기**</h3>**

**<hr>**

**<script>**

var grade;

var score = prompt("홍길동님 점수를 입력하세요", 100);

score = parseInt(score);

if(score >= 90) {grade = "A";}

else if(score >= 80) {grade = "B";}

else if(score >= 70) {grade = "C";}

else if(score >= 60) {grade = "D";}

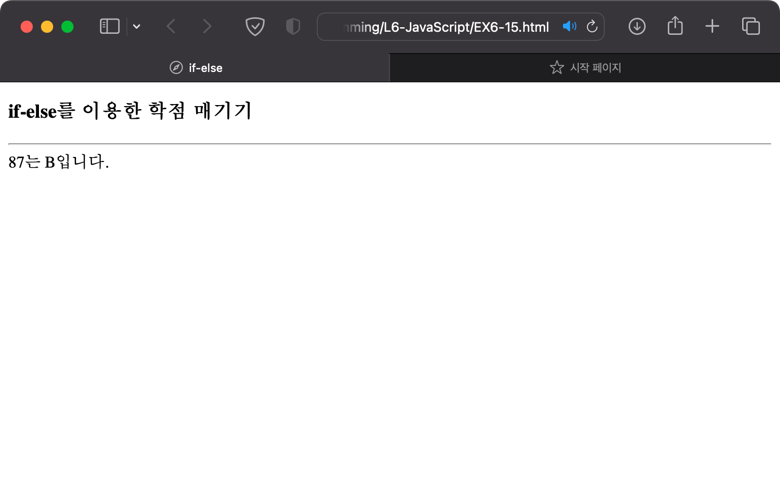
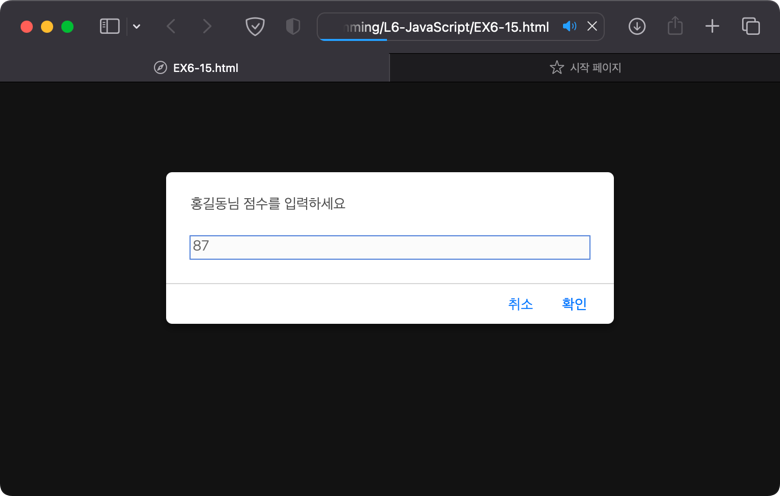
else {grade = "F";}

document.write(score + "는 " + grade + "입니다.<br>");

**</script>**

**</body>**

**</html>**

****

<그림 15-1> 예제 6-15 결과

특정 조건에 따라 실행하고자 하는 코드가 다를 때 조건문을 사용한다. 조건문에는 if문과 switch이 있다. if문의 키워드로 ‘if’, ‘else’, ‘else if’가 있다. ‘if(조건식){…}’의 형식으로 조건식이 참일 경우 중괄호 내의 코드가 실행된다. 거짓일 경우 실행되지 않는다. ‘if(조건식){실행1} else {실행2}’에서 조건식이 참일 경우 ‘실행1’이 실행되며 거짓일 경우 ‘실행2’가 실행된다. 여러 개의 조건을 사용할 경우에는 조건문 내에서 비교연산자를 사용하거나 ‘else if’를 사용할 수 있다. <그림 15-1>에서 점수가 90점 이상이 아닐 경우 ‘else if’를 사용하여 80점 이상, 70점 이상, 60점 이상을 비교하였고 그렇지도 않을 경우 ‘else’를 사용하여 F를 출력하는 것을 확인할 수 있다.

1. **예제 6-16 switch 문 사용**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<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.");

}

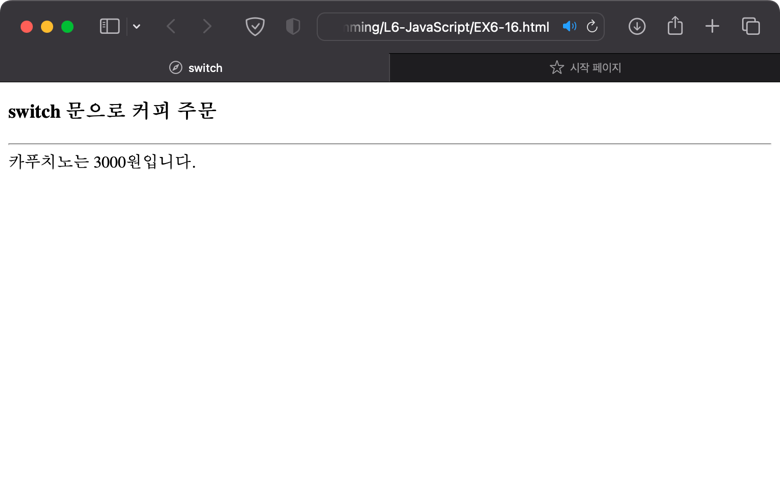
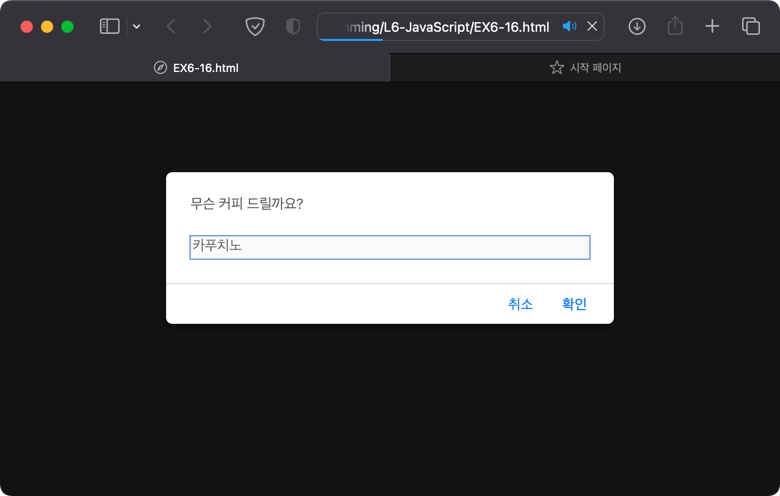
if(price != 0)

document.write(coffee + "는 " + price + "원입니다.");

**</script>**

**</body>**

**</html>**

****

<그림 16-1> 예제 6-16 결과

조건문에서 값에 따라 조건이 많이 발생하면 switch문을 쓰는 것이 유리하다. ‘switch(식) {case 값1: 실행문1; break; case 값2: 실행문2; break; …. default: 실행문d;}’의 형식으로 사용한다. 식의 결과값에 따라 각 case의 값과 비교하여 만약 같을 경우 해당 부분부터 아래로 실행된다. 이때 ‘break’를 만나면 switch문을 빠져나오게 된다. case의 값과 일치하는 것이 없을 경우 default의 실행문을 실행한다. case의 값은 리터럴만 가능하며 변수나 식은 사용할 수 없다. <그림 16-1>은 switch문의 예를 보여준다. 만약 해당 코드에 ‘에스프레소’를 입력하고 ‘break’를 모두 지우면 각 case에 대한 모든 결과가 출력된다.

1. **예제 6-17 for 문으로 10px~35px 크기로 출력**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**for 문**</title>**

**</head>**

**<body>**

**<h3>**for 문으로 10px~35px 크기 출력**</h3>**

**<hr>**

**<script>**

for(var size=10; size <= 35; size+=5){

document.write("<span ");

document.write("style='font-size: " + size + "px'>");

document.write(size + "px");

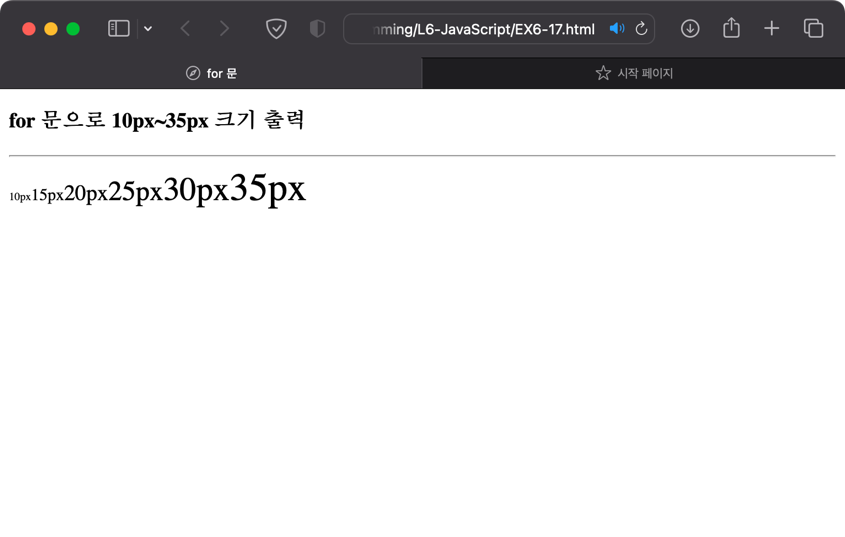
document.write("</span>");

}

**</script>**

**</body>**

**</html>**

****

<그림 17-1> 예제 6-17 결과

반복문으로 for문, while문, do-while문이 있다. for문은 ‘for(초기식; 조건식; 증감식;) { 실행문 }’의 형식으로 사용한다. 초기식에서 사용할 변수의 초기값을 지정한다. 조건식에서 조건을 만족할 경우 실행문이 실행되며 증감식으로 반복 후 작업을 지정하거나 변수의 값을 조정하여 반복의 횟수를 지정한다. <그림 17-1>에서 for문 내로 ‘size’라는 변수가 초기값 10으로 설정되었다. ‘size’의 값이 35 이하라면 해당 실행문은 계속 반복하여 실행하며 증감식에 따라 size는 5씩 증가한다. 이럴 경우 6번의 반복이 끝나면 size의 값은 40이 되므로 for문을 빠져나오게 된다.

1. **예제 6-18 while 문으로 0~n까지의 합 구하기**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<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);

var i = 0, sum = 0;

while (i <= n) {

sum += i;

i++;

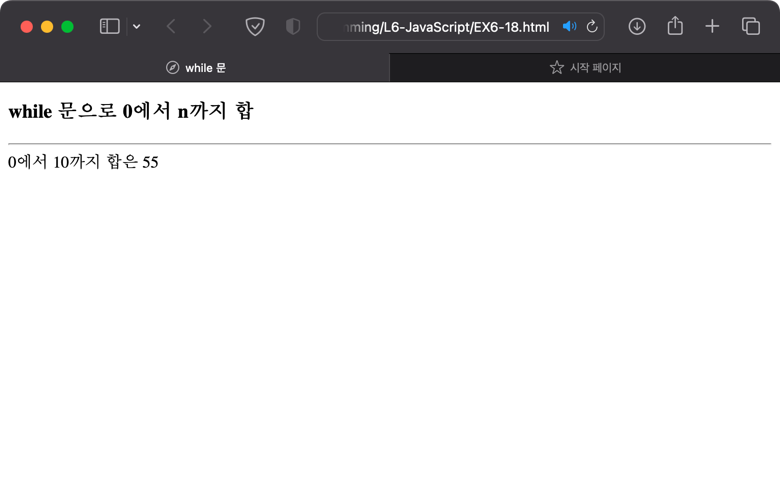
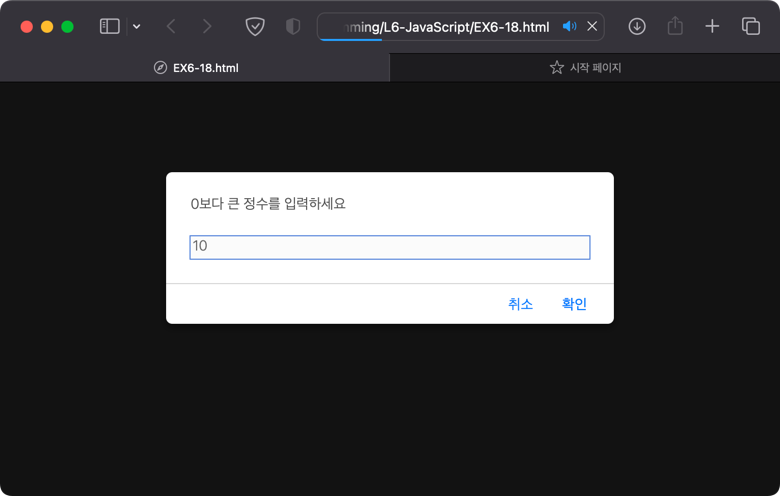
}

document.write("0에서 " + n + "까지 합은 " + sum);

**</script>**

**</body>**

**</html>**

****

<그림 18-1> 예제 6-18 결과

while문 ‘while(조건식) { 실행문 }’의 형식으로 사용된다. 조건식이 참일 경우 실행문이 계속 반복되며 거짓이라면 while문을 빠져나오게 된다. 자칫 실수하면 조건식이 영원히 참이 되어 무한반복에 빠질 수 있으므로 주의해야한다. <그림 18-1>에서 사용자가 입력한 n의 값과 변수 i의 값을 비교하여 i가 n 이하라면 while문이 계속 반복하게 된다. ‘i++;’을 통해 i를 1씩 증가하여 while문을 빠져나올 수 있는데 만약 i++가 없다면 무한반복에 빠지게 된다.

1. **예제 6-19 do-while 문으로 0~n까지의 합 구하기**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<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);

var i = 0, sum = 0;

do {

sum += i;

i++;

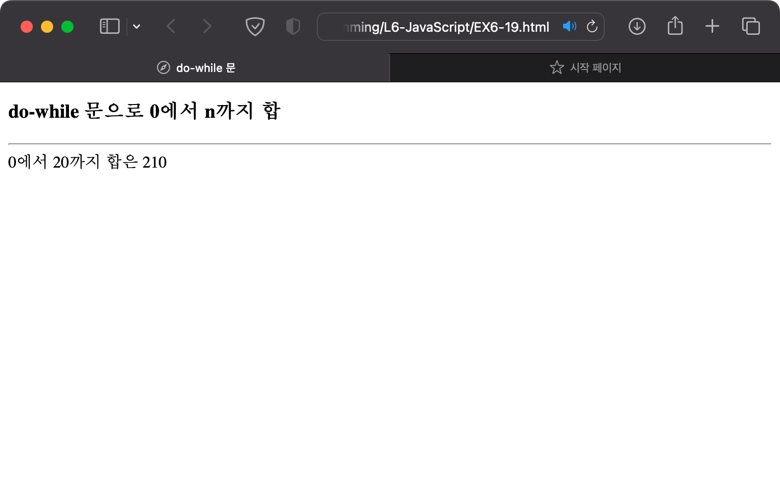
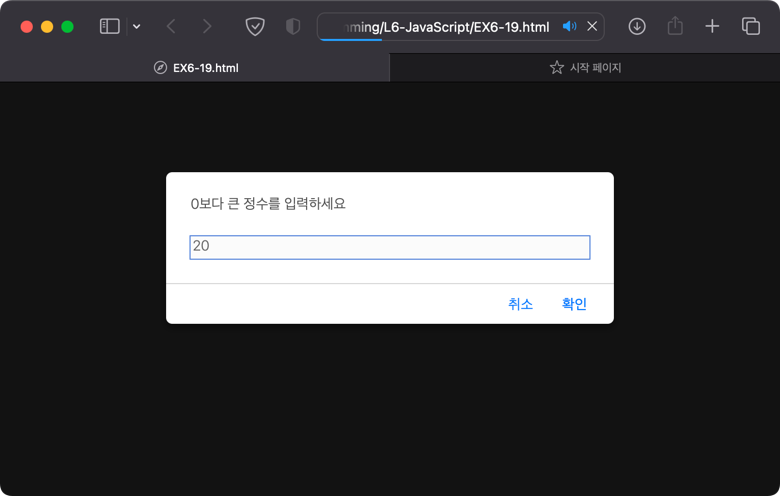
} while (i <= n)

document.write("0에서 " + n + "까지 합은 " + sum);

**</script>**

**</body>**

**</html>**

****

<그림 19-1> 예제 6-19 결과

do-while문은 while문과 한가지 차이점 빼고 모든 것이 동일하다. while문의 실행문은 조건식이 참일 경우에 실행되지만 do-while문의 경우 실행문이 일단 한번 실행되고 조건식을 비교하여 참일 경우 다시 반복하는 구조를 가지고 있다. <그림 19-1>은 do-while문을 사용한 반복문 예제를 보이고있다.

1. **예제 6-20 break 문**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**break 문**</title>**

**</head>**

**<body>**

**<h3>**1에서 얼마까지 더해야 3000을 넘는가?**</h3>**

**<hr>**

**<script>**

var i = 0, sum = 0;

while (true) {

sum += i;

if (sum > 3000) break;

i++;

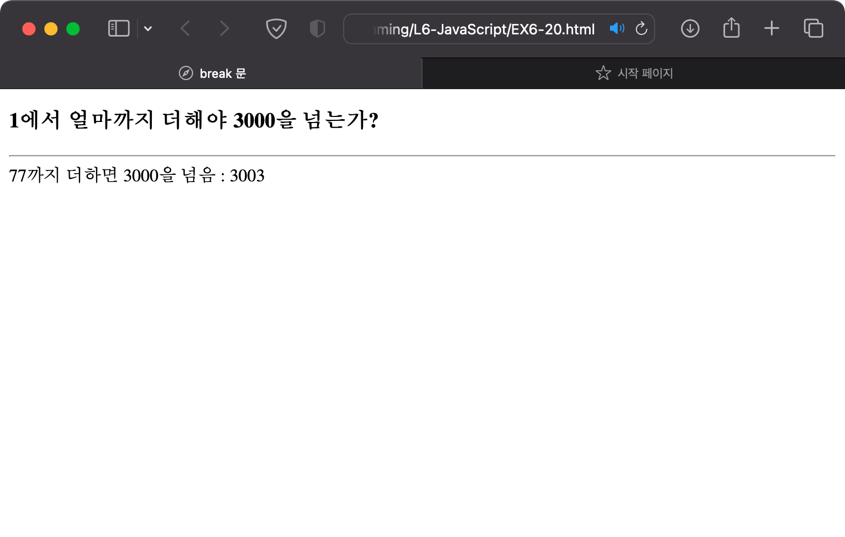
}

document.write(i + "까지 더하면 3000을 넘음 : " + sum);

**</script>**

**</body>**

**</html>**

****

<그림 20-1> 예제 6-20 결과

반복문 내에서 break를 만나면 가장 최근에 만난 반복문을 벗어나게 된다. ‘while(…){ for(…){… break; ….} …}’과 같이 반복문을 중첩하여 사용할 경우를 예를 들면, for문에서 break를 만났기 때문에 for문을 벗어나고 while문의 남은 코드를 마저 수행하게 된다. <그림 20-1>에서 sum의 값이 3000을 넘으면 break가 실행되어 while문을 빠져나오게 된다.

1. **예제 6-21 continue 문**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<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++){

if (i % 3 != 1)

continue;

document.write(i + ' ');

sum += i;

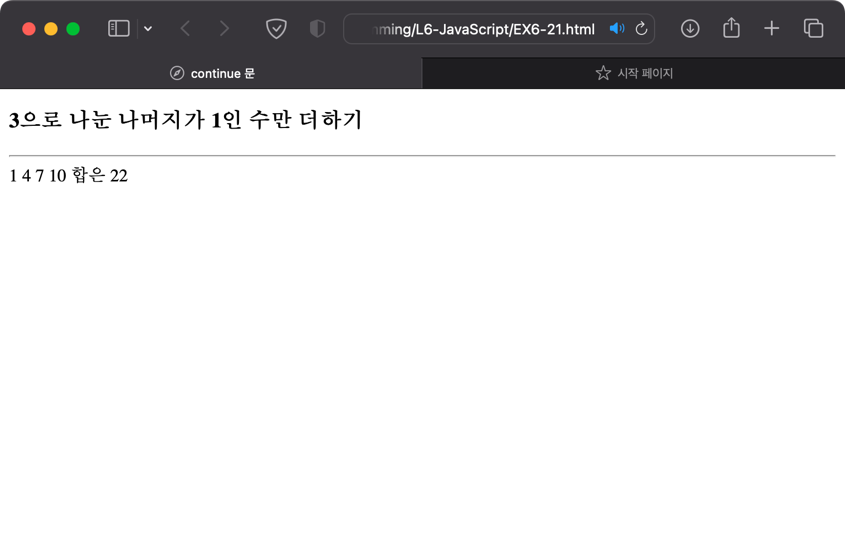
}

document.write("합은 " + sum);

**</script>**

**</body>**

**</html>**

****

<그림 21-1> 예제 6-21 결과

반복문 내에서 continue를 만나면 해당 반복을 중단하고 다음 반복문에 대한 수행으로 이동한다. for문의 경우 증감식의 단계로 이동하며, while문 및 do-while문의 경우 조건식의 단계로 이동한다. <그림 21-1>에서 i를 3으로 나눈 나머지가 1이 아닐 경우 if문 아래의 코드를 무시하고 증감식으로 이동하며, 나머지가 1일 경우 if문 아래의 코드를 수행한다.

1. **예제 6-22 adder() 함수 작성 및 호출**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**함수**</title>**

**<script>**

function adder(a, b) {

var sum;

sum = a + b;

return sum;

}

**</script>**

**</head>**

**<body>**

**<h3>**함수 adder()**</h3>**

**<hr>**

**<script>**

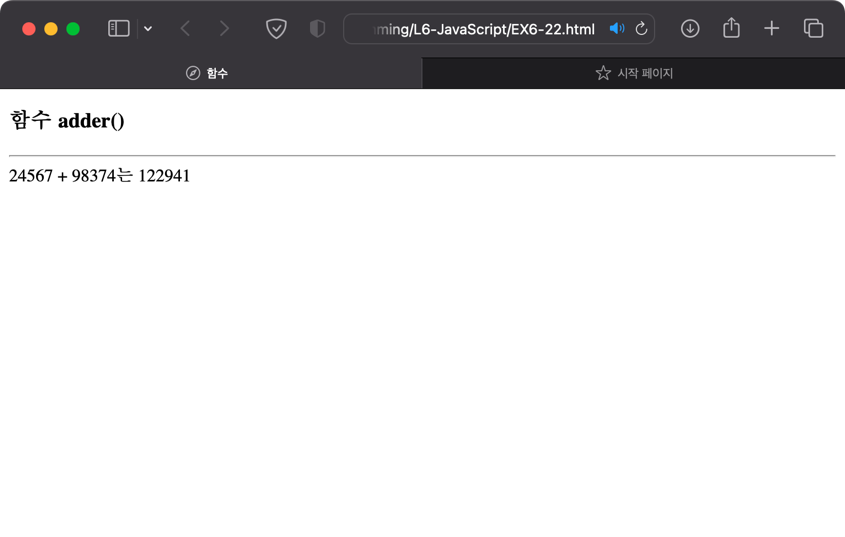
var n = adder(24567, 98374);

document.write('24567 + 98374는 ' + n + '<br>');

**</script>**

**</body>**

**</html>**

****

<그림 22-1> 예제 6-22 결과

자바스크립트에서 함수를 구성하는 방법은 <head>태그의 <script>태그 내에서 function 키워드를 통해 선언할 수 있다. ‘function 함수이름(매개변수1, 매개변수2, …){ 함수 코드; ...; return문;}’의 형식으로 사용한다. C언어의 문법과 유사하지만 함수 선언에서 매개 변수의 타입을 선언하지 않아도 된다는 것과 함수의 리턴 타입이 존재하지 않다는 차이가 있다. <그림 22-1>에서 adder() 함수를 선언하고 사용하는 예를 보이고 있다. 매개변수로 24567과 98374를 각각 a와 b에 전달하고 a+b의 결과인 sum을 반환하여 n에 대입한 후 n을 출력하는 것을 확인할 수 있다.

1. **예제 6-23 eval(), parseInt(), isNaN()**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**함수**</title>**

**<script>**

function evalParseIntIsNaN() {

var res = eval('2\*3+4\*6');

document.write('eval("2\*3+4\*6")는 ' + res + '<br>');

var m = eval('32');

document.write('parseInt("32")는 ' + m + '<br>');

var m = eval('32', 16);

document.write('parseInt("32", 16)는 ' + m + '<br>');

var n = eval('0x32');

document.write('parseInt("0x32")는 ' + n + '<br><br>');

n = parseInt('hello');

if (isNaN(n))

document.write('hello는 숫자가 아닙니다.');

}

**</script>**

**</head>**

**<body>**

**<h3>**eval(), parseInt(), isNaN()**</h3>**

**<hr>**

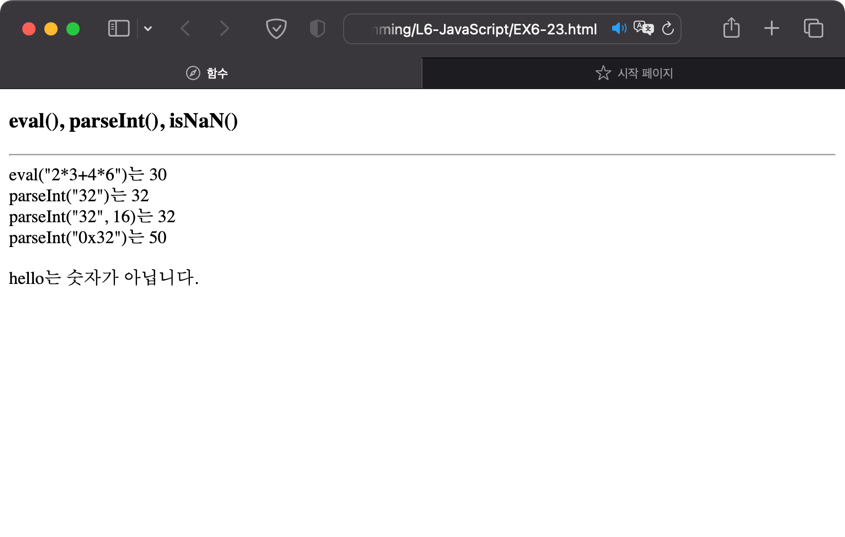
**<script>**

evalParseIntIsNaN();

**</script>**

**</body>**

**</html>**

****

<그림 23-1> 예제 6-23 결과

자바스크립트는 미리 정의된 전역함수를 제공한다. 여러 전역함수가 있지만 예제에 사용된 eval(), parseInt(), inNaN()을 살펴보면, 우선 eval() 함수는 문자열로 표현된 자바스크립트 식을 계산하고 결과를 반환하는 함수이다. eval(“2+3”)은 5를 반환한다. parseInt() 함수는 문자열을 정수로 반환하는 함수이다. parseInt(“10”)은 정수 10을 반환하고 parseInt(“0x10”)은 정수 16을 반환한다. inNaN() 함수는 함수로 전달된 값이 숫자인지 판단하여 숫자가 아니라면 true를 반환한다. 만약 함수로 전달된 값이 숫자가 아니라면 숫자로 강제 변환하여 검사를 수행한다. 예를 들어 isNaN("123");에서 문자열 ‘123’이 전달되었지만 숫자로 강제 변환하면 123이 되므로 false를 반환한다. <그림 23-1>에서 미리 정의된 전역함수의 사용 예를 확인할 수 있다.

1. **예제 6-24 구구단 출력 함수 만들기**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>**함수 만들기**</title>**

**<script>**

function gugudan(n) {

var m = parseInt(n);

if (isNaN(m) || m < 1 || m > 9) {

alert('잘못입력하셨습니다.');

return;

}

for (var i = 1; i <= 9; i++) {

document.write(m + 'x' + i + '=' + m \* i + '<br>');

}

}

**</script>**

**</head>**

**<body>**

**<h3>**구구단 출력 함수 만들기**</h3>**

**<hr>**

**<script>**

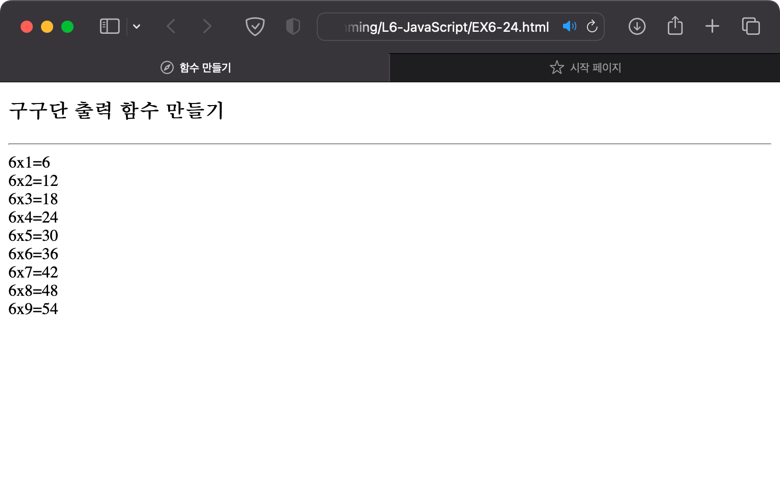
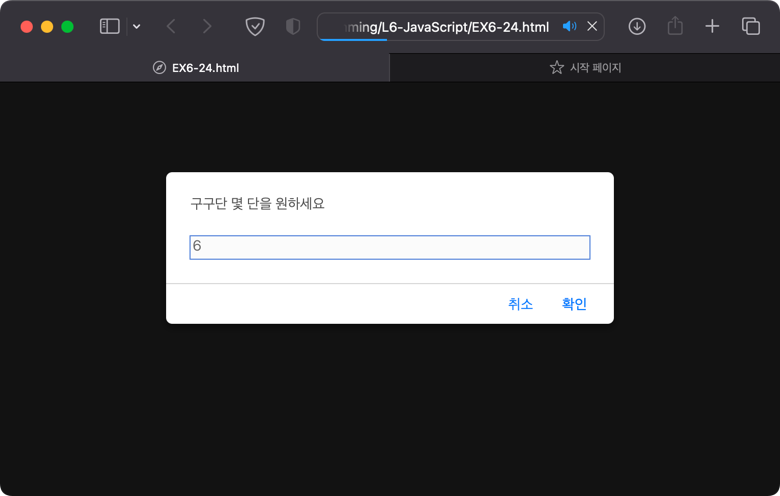
var n = prompt('구구단 몇 단을 원하세요', '');

gugudan(n);

**</script>**

**</body>**

**</html>**

****

<그림 24-1> 예제 6-24 결과

앞서 나온 변수, 연산자, 조건문, 반복문, 미리 정의된 전역함수 등을 활용하여 구구단 출력을 하는 함수를 만다는 예제이다. 사용자가 1~9 내의 수를 입력하면 정수로 변환하여 해당 수의 구구단을 출력하는 함수이다.

1. **참고문헌**

* W3schools, (n.d.), “JavaScript Tutorial”, <https://www.w3schools.com/js/default.asp>
* TCPSCHOOL, (n.d.), “자바스크립트”, <http://www.tcpschool.com/javascript/intro>