



CSS3+JavaScript와 함께 하는 HTML5 웹 프로그래밍

6장 자바스크립트 기본

CHAPTER

06

HTML5+CSS3+JavaScript



자바스크립트 기본

6.1 자바스크립트 개요

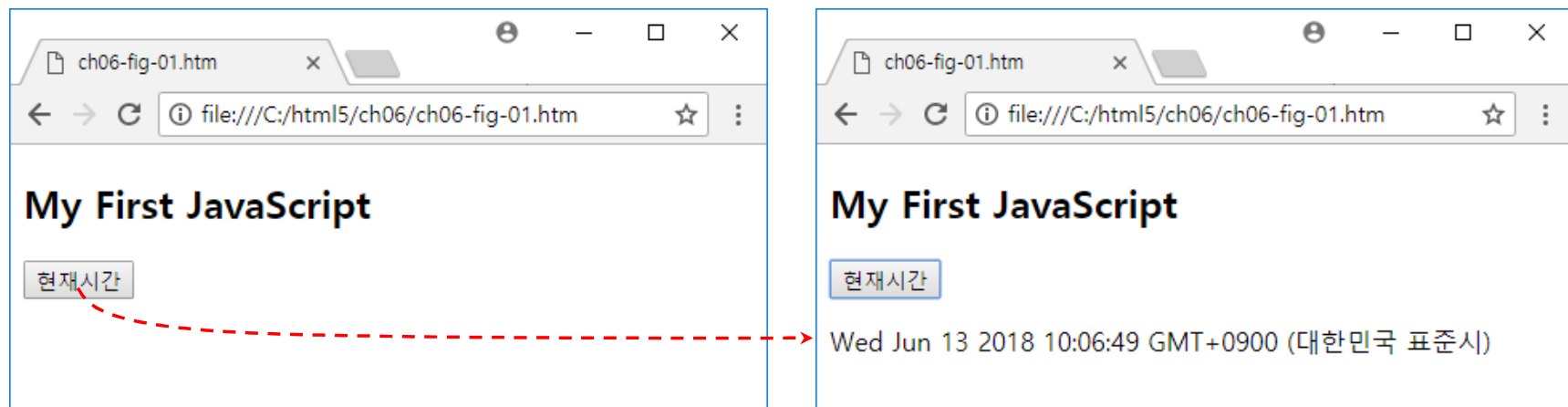
6.2 자바스크립트의 기본 문법

6.3 자바스크립트 함수

6.1 자바스크립트 개요

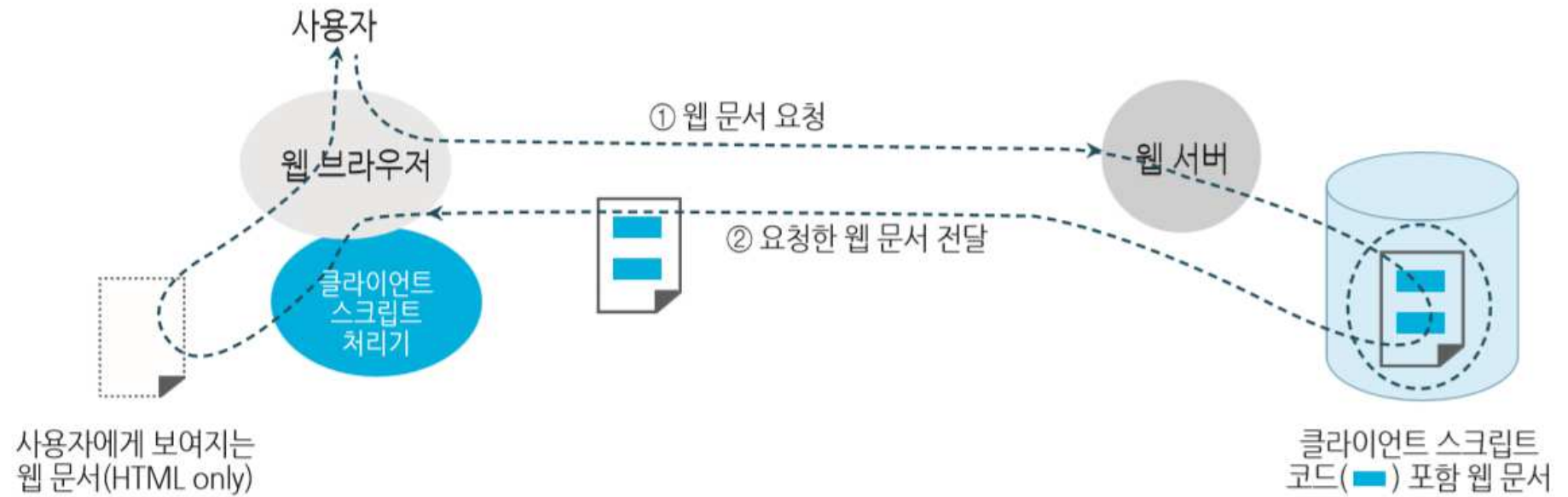
6.1.1 자바스크립트란?

- 웹 문서 작성 : HTML + CSS3
HTML : 웹 문서의 내용 구성
CSS : 웹 문서의 출력 스타일 지정, 가상클래스를 이용한 간단한 애니메이션 효과 지정 가능
- HTML + CSS3 만을 이용한 웹 문서의 한계 : 사용자들에 따른 문서 내용의 동적 구성은 불가함
- 웹 문서 내용의 동적 구성을 위해서는 웹 문서에 프로그래밍 코드를 포함시켜야 함

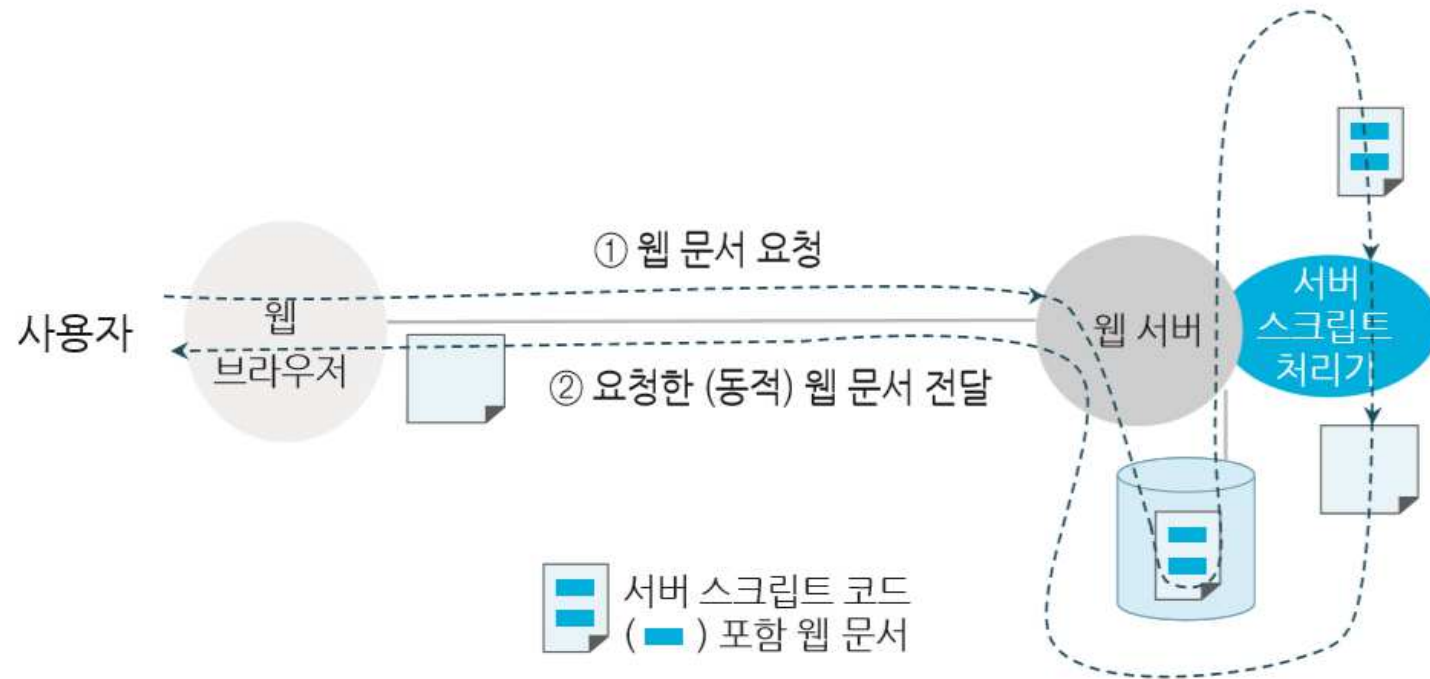


- 웹 문서내용의 동적 구성을 위해 웹 문서에 포함되는 프로그래밍 코드 : 웹 스크립트 코드
 - 웹 문서 구성 : HTML + CSS3 + 웹 스크립트 코드
 - 웹 문서의 작성 = 웹프로그래밍(프로그래밍 구현 기능들의 웹 문서 추가)
- 웹 문서 내용의 동적 구성 방법
 - 클라이언트 웹 스크립트 작성
 - ✓ 사용자와의 상호작용에 따른 웹브라우저에서의 동적 문서내용 구성
 - ✓ 클라이언트 웹 스크립트 작성 언어 : 자바스크립트(표준)
 - ✓ 웹 서버는 사용자들에게 동일한 웹 문서를 제공하지만 사용자의 환경(시간 등)과 상호작용(사용자와의 인터페이싱 결과)에 따라 다른 문서내용을 나타냄
 - 서버 웹 스크립트 작성
 - ✓ 사용자 정보를 이용한 웹 서버에서의 동적 문서내용 구성
 - ✓ 서버 웹스크립트 작성 언어 : PHP, JSP, ASP, Node J, 파이썬 등
 - ✓ 웹서버는 사용자들의 정보에 따라 사용자별로 동적 웹문서를 작성해서 전달함
따라서, 사용자들은 자신의 정보를 기반으로 구성된 웹문서를 전달받게 됨

- 클라이언트 스크립트(자바스크립트) 코드 포함 웹 문서의 처리(그림 1.14)

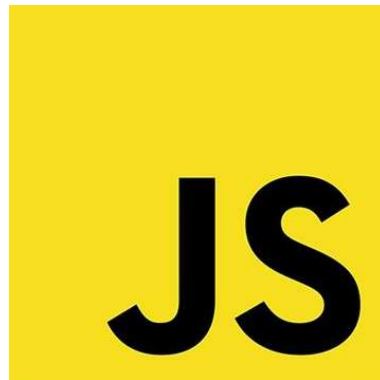


- 서버 스크립트 코드 포함 웹 문서의 처리(그림 1.15)



● 자바스크립트는 ?

- Netscape 사의 Brendan Eich가 "Livescript" 라는 이름으로 최초 개발함
- Sun micro systems의 개발권한 인수, 1995년 "자바스크립트" 로 이름변경후 발전시킴
 - 현재, HTML5의 표준 스크립트 언어 지정(모든 웹 브라우저들에서 기본으로 지원함)
 - 자바스크립트 표준은 ECMA 에서 관리
 - 공식명칭 : ECMA 스크립트(ECMA-262 / ISO 16262 표준) 언어
- 자바스크립트 기반의 다양한 라이브러리들, 웹 응용 프레임워크들 출현 및 경쟁
 - jQuery
 - Node.js, Angular.js, D3(DataDrivenedDocument).JS, React.js 등

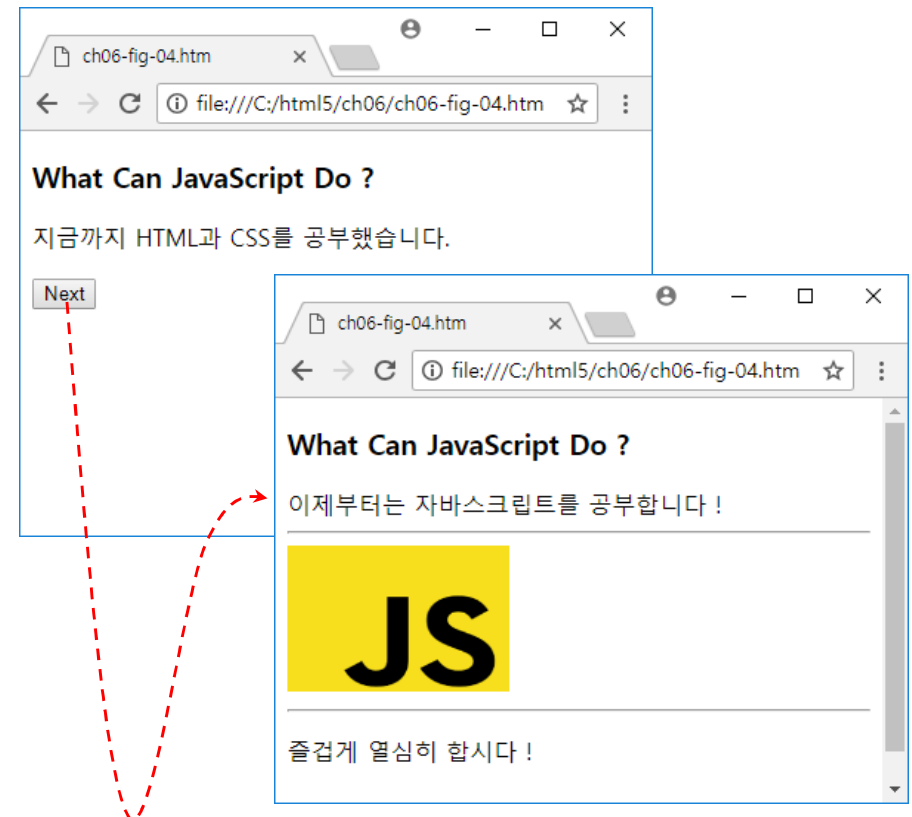
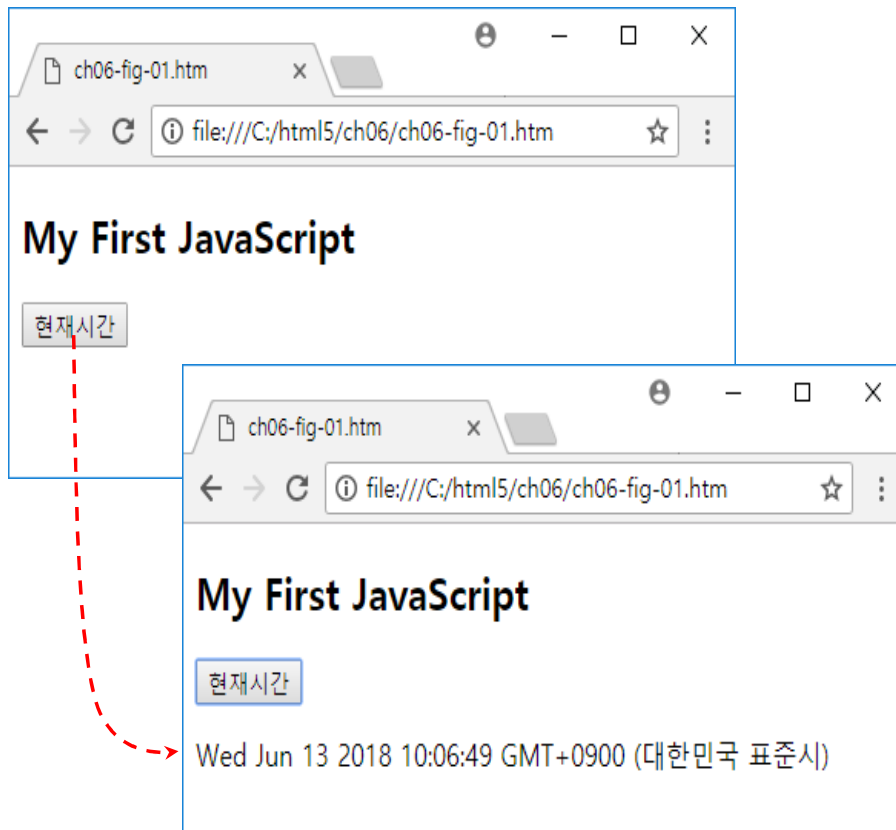


● 자바스크립트 특성

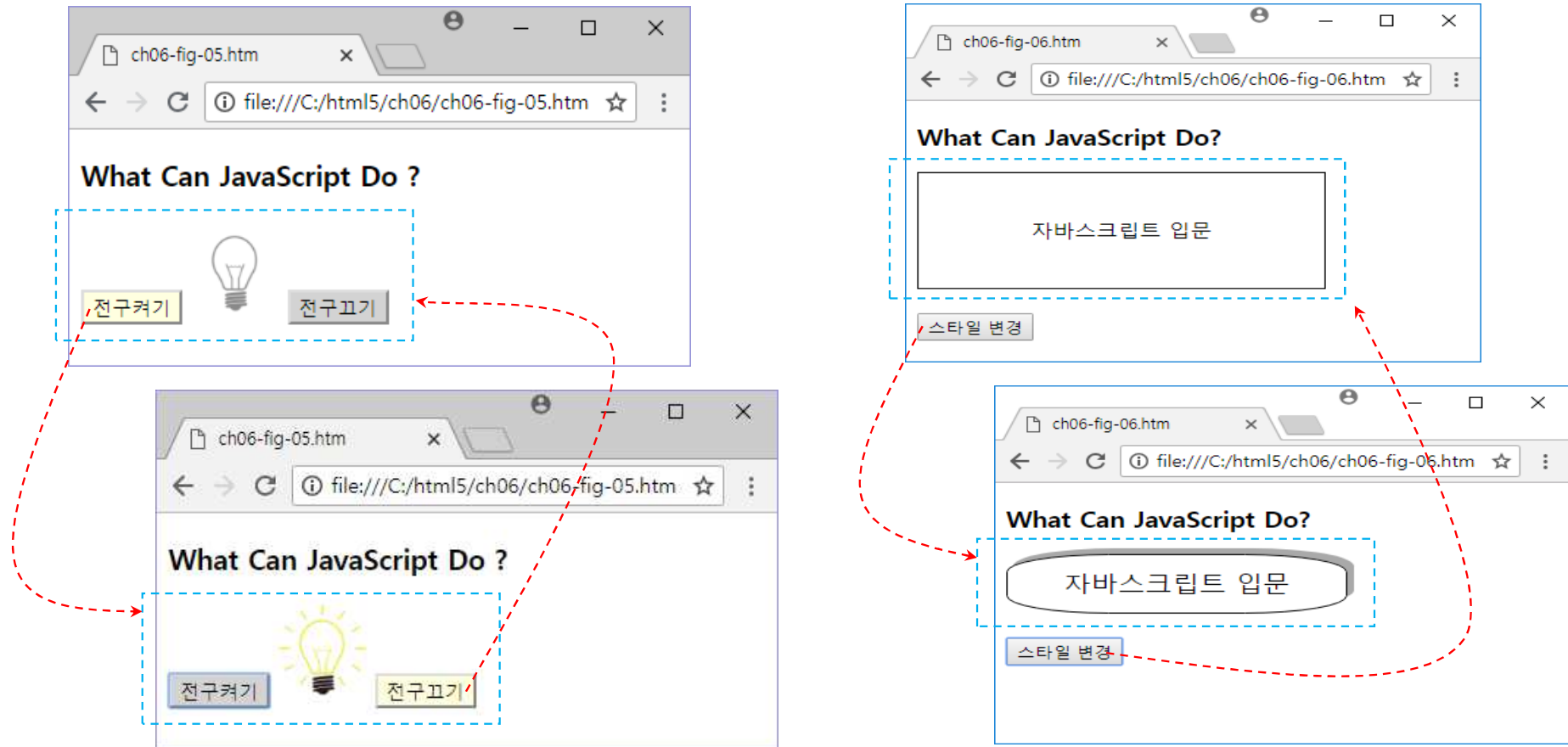
- 웹의 클라이언트 환경에서 웹 문서 내용의 동적 처리를 지원하는 언어
- 플랫폼 독립적인 인터프리터 언어 : 모든 웹 브라우저에서 곧바로 실행됨
- 스크립트 언어
 - HTML 문서에 필요한 부분 코드만을 포함시킴
 - 별도의 정형화된 프로그램 구조는 없음
- 동적 타이핑(dynamic typing) 지원 : 변수들의 자료형을 자유롭게 변환할 수 있음
- 객체 기반(object based)의 언어
 - C/C++, java 등과 유사한 문법적 특성
 - 웹 브라우저 및 HTML 문서 관련 다양한 객체들(DOM, BOM) 제공
- 웹 브라우저 환경에서 발생하는 이벤트 처리 지원
 - 웹 브라우저에서 HTML 문서와 사용자들과의 다양한 상호작용에 필요한 처리 지원

- 자바스크립트를 이용해서 작성된 HTML 문서들의 예

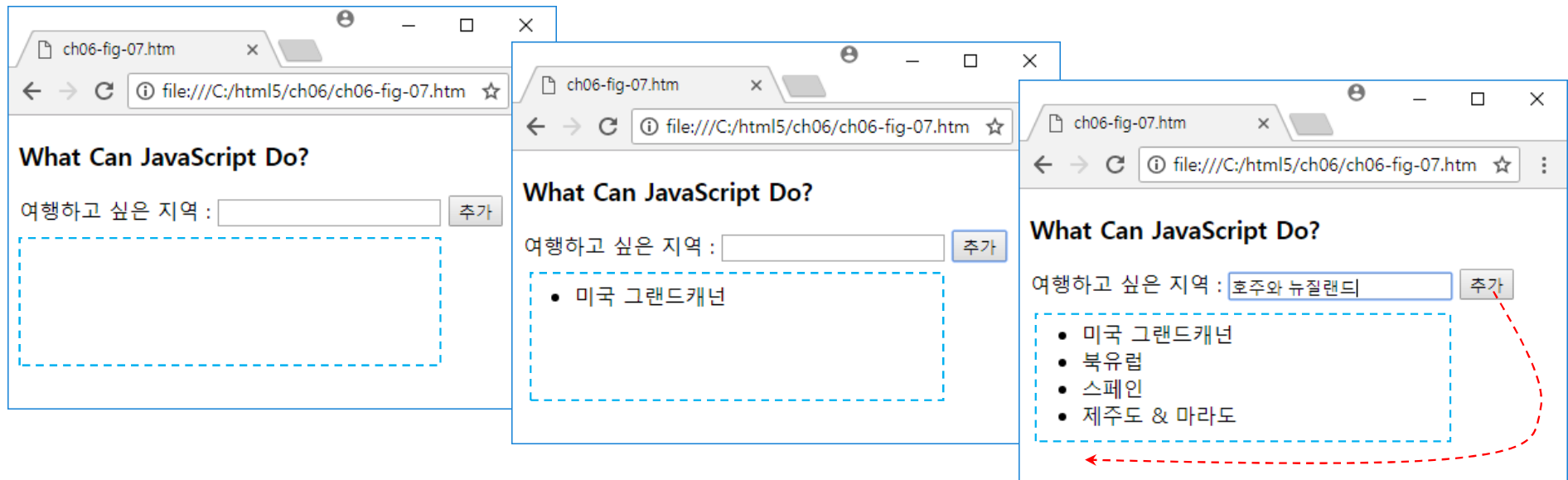
- 문서 내용의 변경& 추가(그림 6.3. 6.4)



- 속성 또는 CSS3 스타일 속성 변경(그림 6.5, 그림 6.6)



- 문서 내용의 동적인 구성(사용자 입력을 문서 내용으로 추가함(그림 6.7))



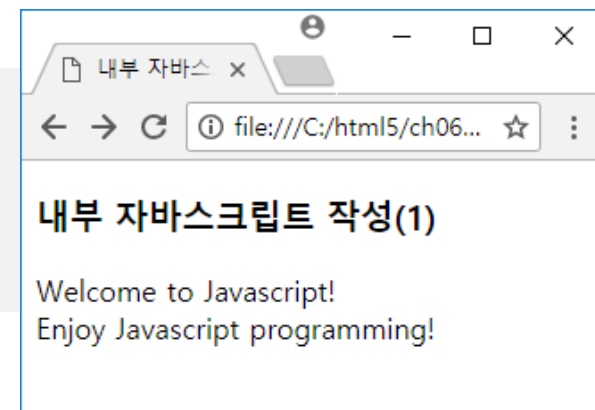
6.1.2 자바스크립트 코드의 작성 방법

- 자바스크립트는 HTML 문서에서 HTML, CSS3 코드들과 함께 작성됨
- 웹 문서를 웹 브라우저로 확인(로드)할 때 실행됨
- 자바스크립트 코드의 작성방법 : 내부/외부/인라인 코드의 세가지 방법 있음

(1) 내부 자바스크립트

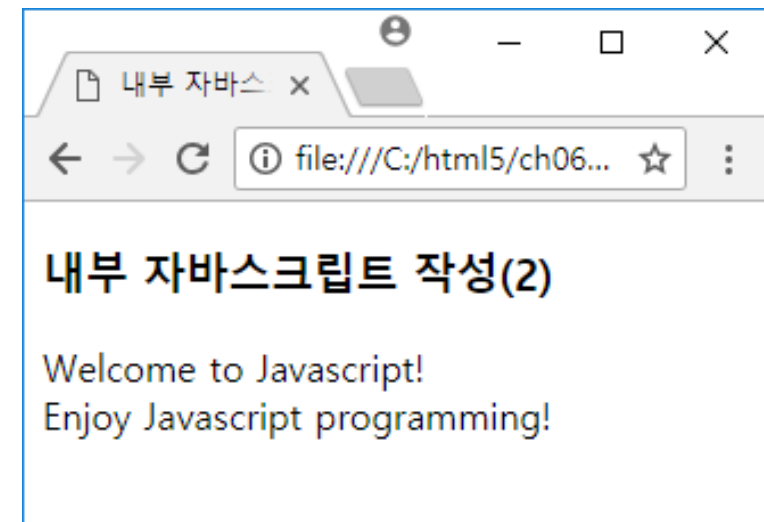
- <script>를 이용해서 직접 HTML 문서에 자바스크립트 코드 작성함
- 즉, 자바스크립트 코드가 실행되는 HTML 문서 위치에서
- <script>와 </script> 사이에 자바스크립트 코드를 포함시킴

```
<h3> 내부 자바스크립트 작성(1) </h2>  
<script>  
  document.write("Welcome to Javascript! <br> ");  
  document.write("Enjoy Javascript programming!");  
</script>
```



```
<head>
  <script>
    var subject1="Welcome to Javascript! <br> ";
    var subject2="Enjoy Javascript programming!";
    function welcome(target) {
      document.write(target);
    }
  </script>
</head>

...
<h3> 내부 자바스크립트 작성(2) </h3>
<script>
  welcome(subject1);
  welcome(subject2);
</script>
```



(2) 외부 자바스크립트

- 자바스크립트 코드를 별도의 파일 (확장자 .js인 텍스트 파일)로 작성해 두고, `<script>` 태그의 `src` 속성으로 이 파일을 지정해서 연결함

```
<scriptsrc="자바스크립트파일경로"></script>
```

- CSS의 외부 스타일시트 지정 방법과 유사함
- 코드 가독성과 재활용성을 높이고, 코드 유지보수를 용이하게 함

[HTML 파일 : ch06- ex03.htm]

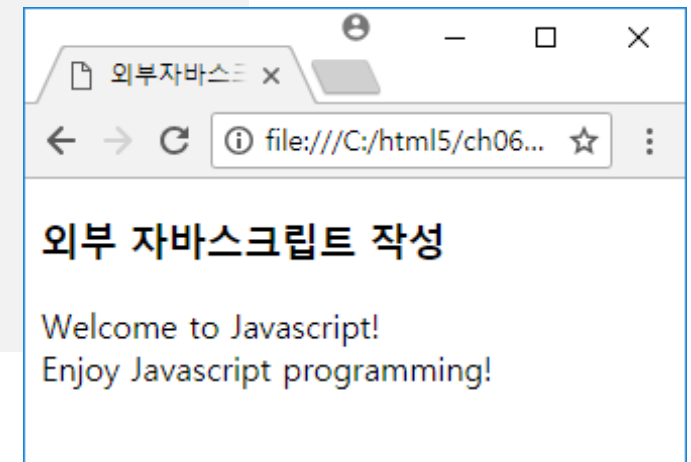
```
<!DOCTYPE html>
<html>
<head>
  <title> 외부자바스크립트 </title>
</head>
<body>
  <h3> 외부 자바스크립트 작성 </h3>
  <script src="welcome.js"></script>
</body>
</html>
```

[자바스크립트 파일 : welcome.js]

```
var subject1="Welcome to Javascript! <br> ";
var subject2="Enjoy Javascript programming!";

function welcome(target) {
  document.write(target);
}

welcome(subject1);
welcome(subject2);
```



(3) 인라인 자바스크립트

- 특정 HTML 요소에서 이벤트 속성(예를 들면, onclick 속성)이나 <a> 요소의 href 속성값으로 자바스크립트 코드를 나타냄

**<태그이벤트속성="자바스크립트코드">...</태그> 또는
<ahref="javascript:자바스크립트코드">...**

- 이벤트 속성 : 해당 이벤트가 발생되면 지정된 자바스크립트 코드들(속성값)을 실행시킴
 - 이벤트 속성명은 "on"으로 시작함
 - (예) onclick – 사용자가 마우스를 클릭하면 발생하는이벤트(click)에 대한 이벤트 속성

```
<!DOCTYPE html>
<html>
<head> <title> 인라인 자바스크립트 </title> </head>
<body>
<h3> 인라인 자바스크립트 작성 </h3>
<button onclick="alert('Welcome to Web Programming!');" >
  웹프로그래밍
</button>
<a href="javascript: alert('Welcome to Javascript!');" > 자바스크립트 </a>
</body>
</html>
```

이 페이지 내용:

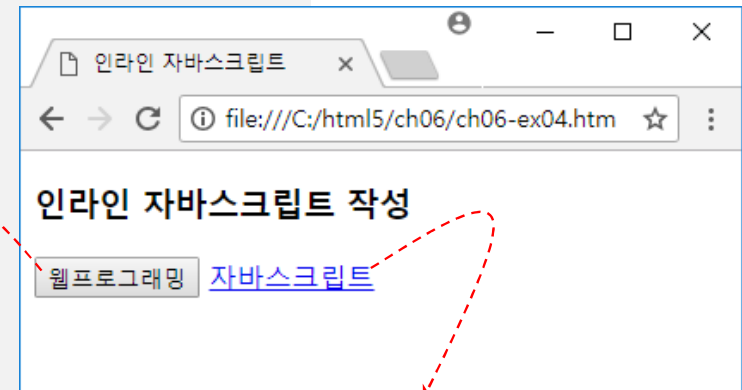
Welcome to Web Programming!

확인

이 페이지 내용:

Welcome to Javascript!

확인



[예 6.6] 자바스크립트 코드의 다양한 작성

[HTML 파일 1]

```
<head>
  <script src="welcome1.js"> </script>
</head>
<body>
  <h4> 외부 & 내부 자바스크립트 작성 </h4>
  <script>
    welcome(subject1);
    welcome(subject2);
  </script>
</body>
```

[자바스크립트 파일(welcome1.js)]

```
var subject1="Web Programming!";
var subject2="Javascript!";

function welcome(target) {
  document.write("Welcome to " + target);
}
```

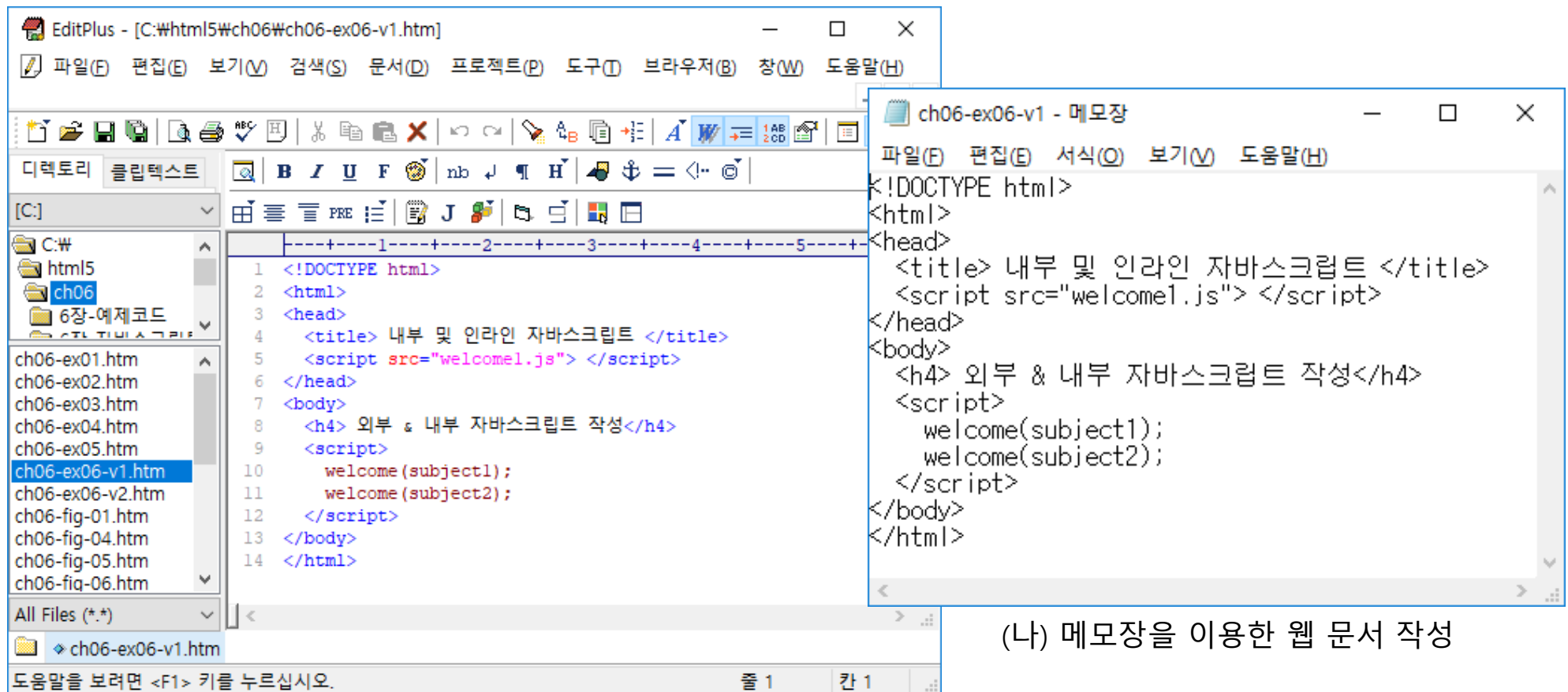
[HTML 파일 2]

```
<head>
  <script src="welcome1.js"> </script>
</head>
<body>
  <h4> 외부 & 인라인 자바스크립트 작성 </h4>
  <button onclick="welcome(subject1); "> 웹프로그래밍 </button>
  <button onclick="welcome(subject2); "> 자바스크립트 </button>
</body>
```

6.1.3 자바스크립트 실습환경

- 자바스크립트 프로그래밍 실습을 위한 도구
 - 웹 문서 작성 소프트웨어 : 자바스크립트 코드 작성
 - 웹 브라우저 : 자바스크립트 코드 실행 및 결과 확인
- 웹 문서 작성 소프트웨어
 - 텍스트 편집기, EditPlus3, 나모 웹 에디터, 드림위버 등
 - 윈도우 시스템의 메모장 프로그램 사용
- 웹 브라우저
 - 구글 크롬, 인터넷 익스플로러 11 등
 - 웹 브라우저의 개발도구 이용(F12)
 - ✓ HTML 문서의 처리 과정에서의 HTML, CSS3, 자바스크립트의 오류 등 확인 가능
 - ✓ 현재 웹 문서의 수정, 자원파일 정보 등 확인기능 등 지원

[그림 6.8] HTML 문서 작성 프로그램 비교

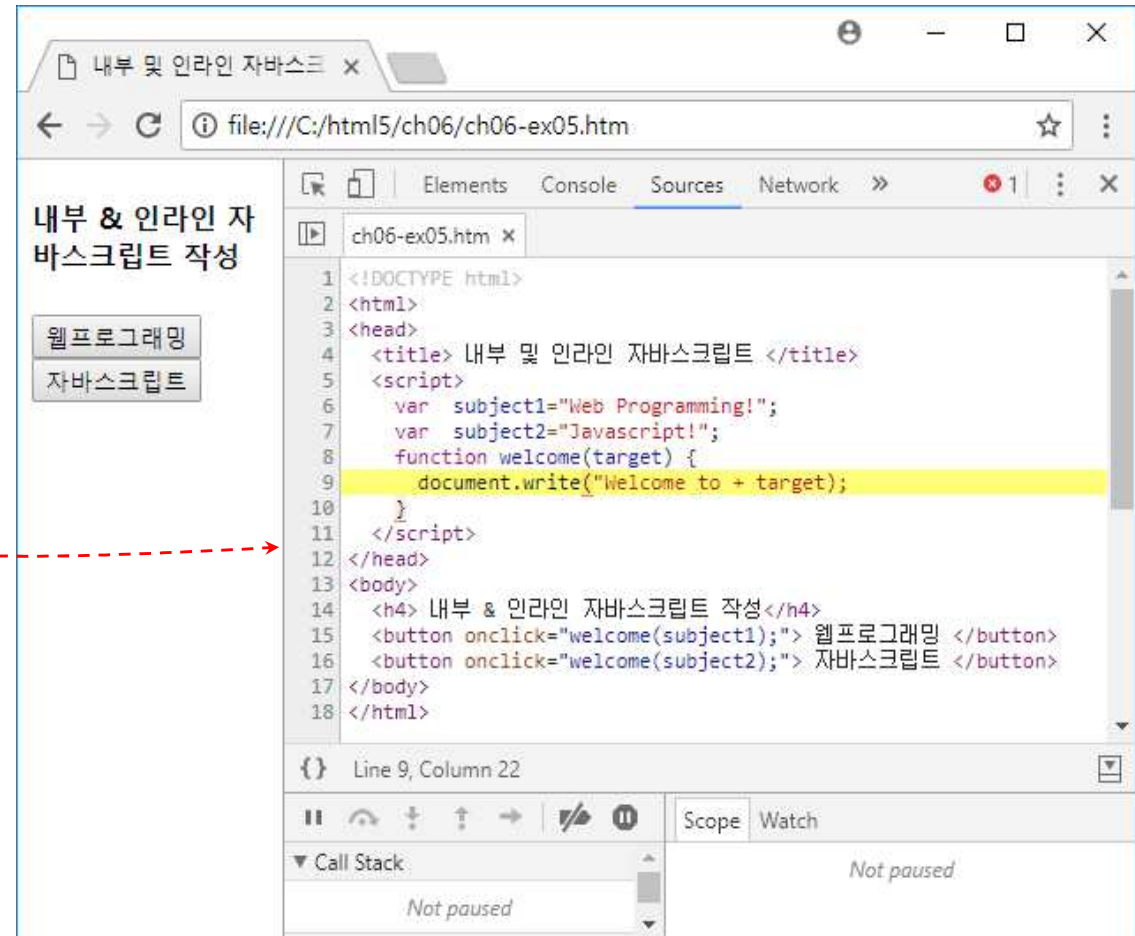
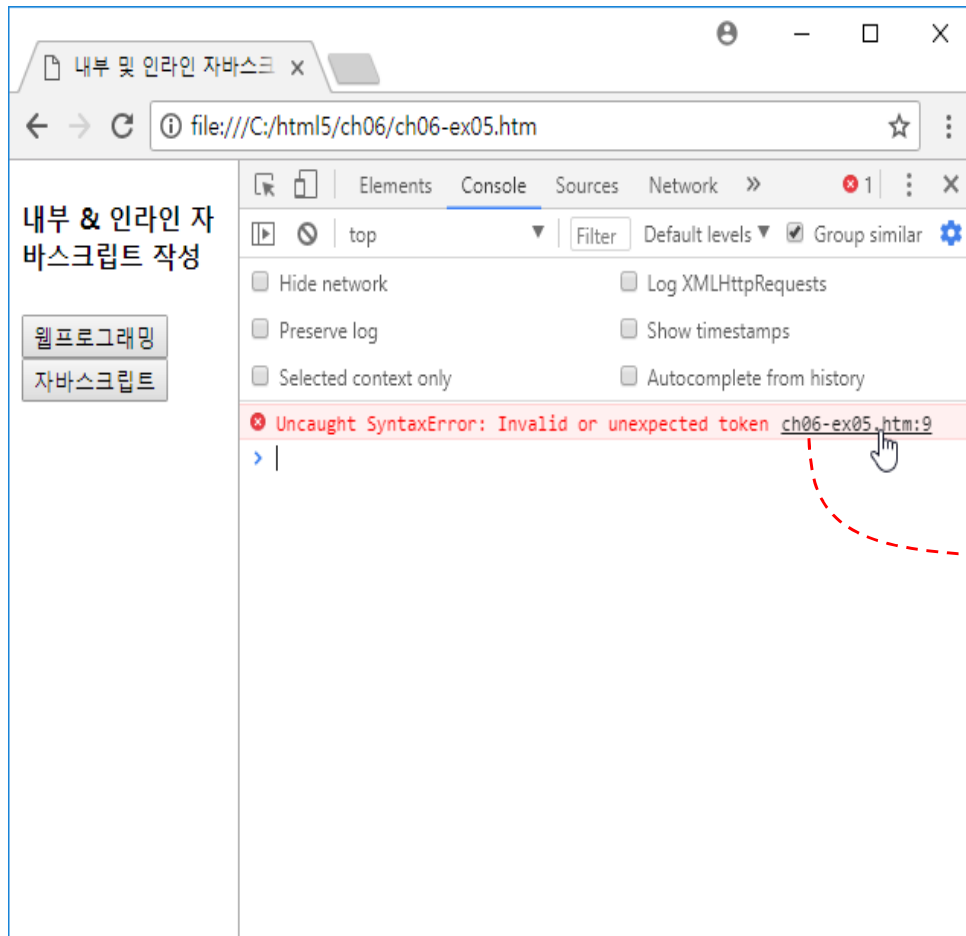


(나) 메모장을 이용한 웹 문서 작성

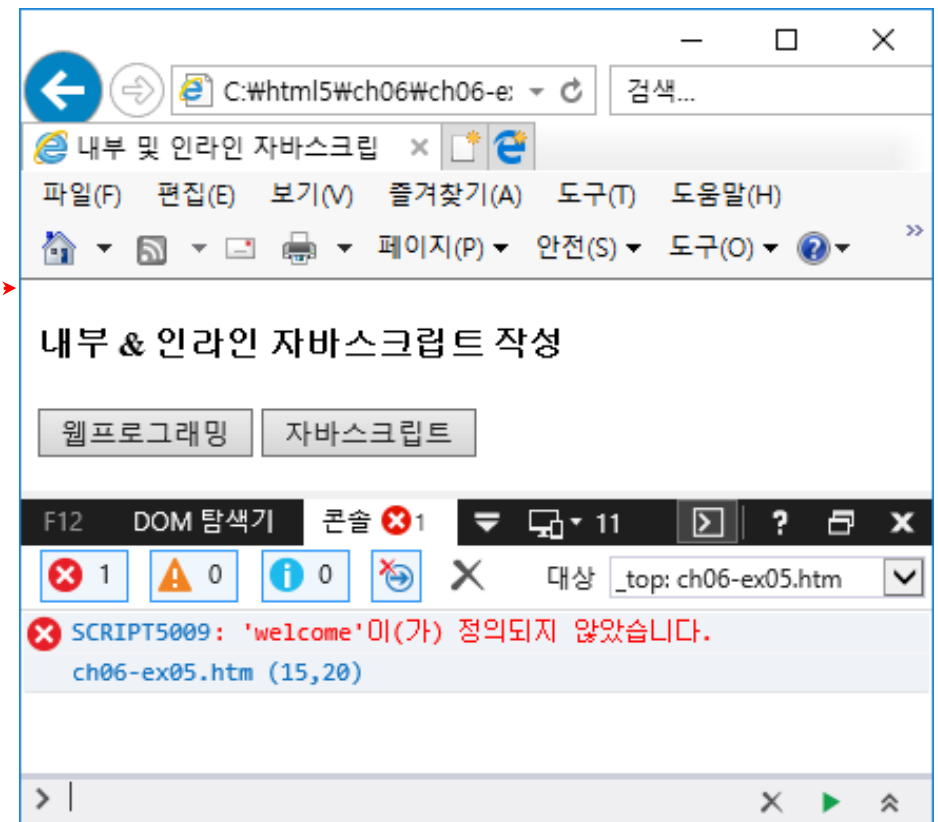
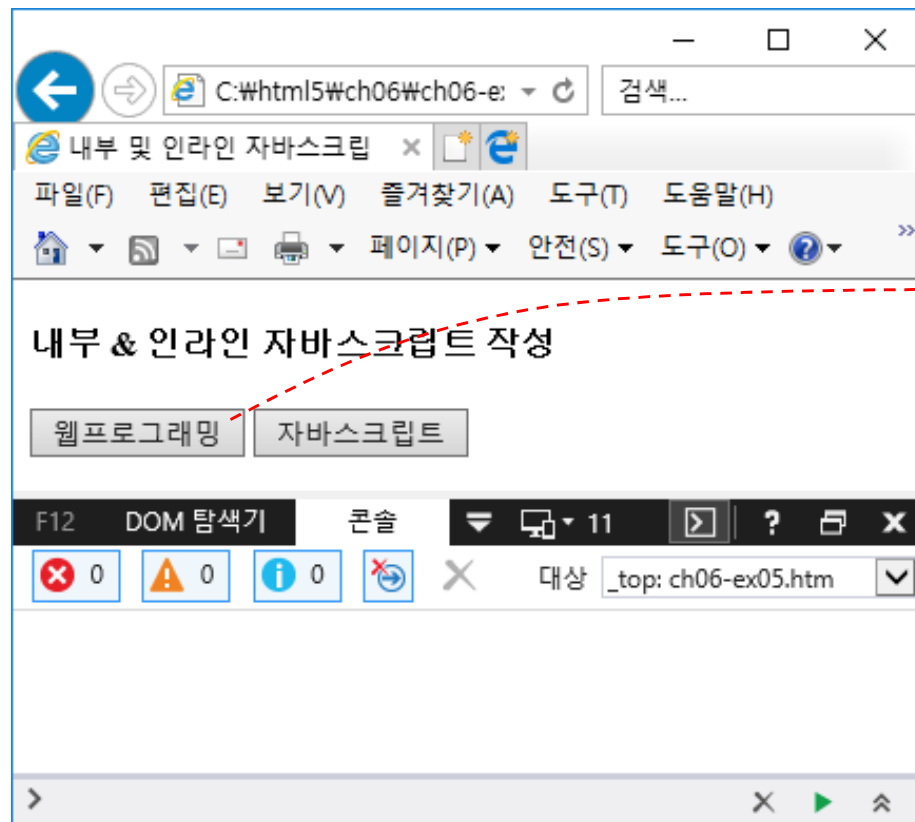
(가) EditPlus를 이용한 웹 문서 작성

[그림 6.9] 크롬과 인터넷 익스플로러 11의 개발자 도구 및 콘솔창

(가) 크롬 웹브라우저의 개발도구(F12) 및 콘솔창



(나) 인터넷 익스플로러 11의 개발도구(F12) 및 콘솔창



6.2 자바스크립트의 기본 문법

- C/C++ 또는 Java와 사용환경은 다르지만 기본적인 문법 구조는 거의 같음
- 스크립트 언어, 변수 자료형, 객체 기반 언어 등의 특성

6.2.1 주요 구성요소

(1) 자바스크립트 코드 구조

- 정형화된 프로그램 구조 없음(HTML 문서의 실행 위치에 필요한 자바스크립트 코드를 포함시키는 방식으로 사용함)
- 일반적으로
 - ✓ <head> 부분 : 변수와 함수정의, HTML 문서를 로딩할 때 실행되는 코드들 작성함
 - ✓ <body> 부분 : 함수호출 등과 같은 실행 코드들 작성함

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<script>
```

```
var obj1 = "CSS3";
```

```
function welcome(obj) {  
    document.write(obj);
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<h3> Welcome! &nbsp; HTML5 웹 프로그래밍 </h3><hr>
```

```
<ol>
```

```
<li> Using HTML! + <script> welcome(obj1); </script> ! </li>
```

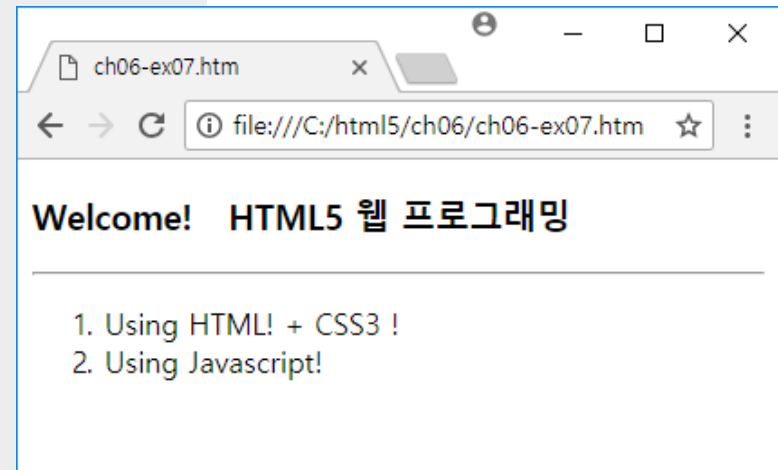
```
<li> <script> welcome("Using Javascript!"); </script> </li>
```

```
</ol>
```

```
</body>
```

```
</html>
```

자바스크립트 코드



(2) 자바스크립트의 문자 집합

- 유니코드 문자세트(unicode ver.3) 사용
- 자바스크립트 명령 코드들은 주로 영문자와 특수문자 등의 아스키 코드 문자들로 구성됨
 - **영문 대소문자들을 구별하고**, 하나 이상의 공백문자들은 무시함
 - 자바스크립트의 예약어들이나 변수명, 함수명 등은 반드시 영문 대소문자를 구별해야 함
- 한글을 포함해서 그 외의 문자들은 단순히 문자열 자료로 취급함
 - HTML 문서에서 한글은 utf-8 인코딩을 사용함(영문 1바이트, 한글 3바이트)

(3) 주석문

- ❶ // 한 라인의 주석문 (// 이후의 문자열은 모두 주석으로 처리됨)
- ❷ /* 여러 라인에 걸쳐 있는 주석문 */

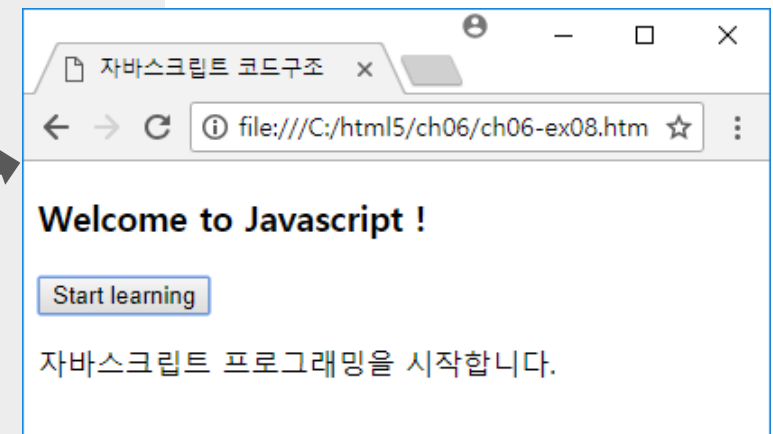
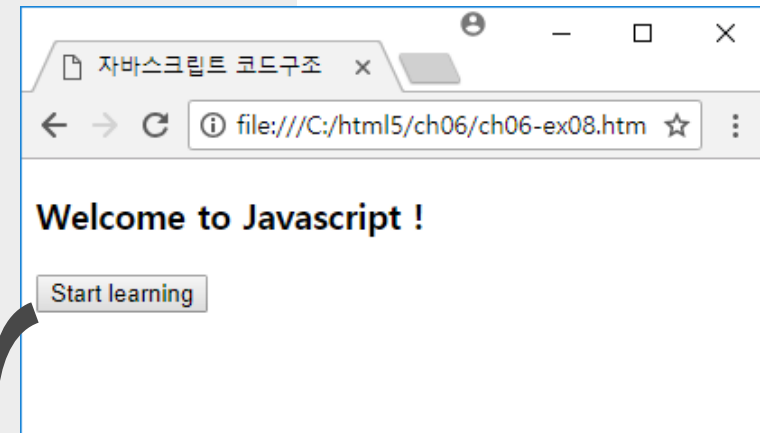
(4) 기본 출력 함수: document.write()

- ❶ document.write("출력문자열"); // 지정된 출력문자열을 웹브라우저 화면에 나타냄
- ❷ document.write(변수); // 지정된 변수값(문자열로 변환)을 나타냄

■ 주석문 사용 예

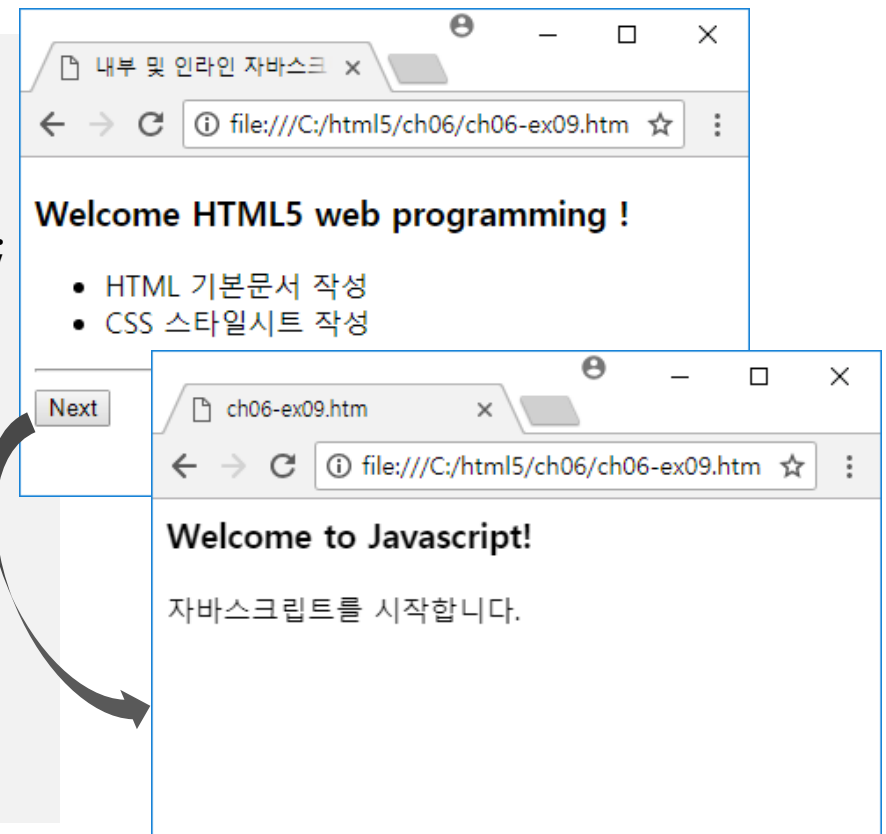
```
<!DOCTYPE html>
<html>
<head>
  <title>자바스크립트 코드구조</title>
  <script>
    // 문자열 변수 정의
    var txt = " 자바스크립트 프로그래밍을 시작합니다.";

    /* 자바스크립트 함수 welcome() 정의
       <p> 태그내용으로 문자열변수 txt 를 지정함 */
    function welcome() {
      document.getElementById("demo").innerHTML = txt;
    }
  </script>
</head>
<body>
  <h3> Welcome to Javascript !</h3>
  <button type="button" onclick="welcome();">
    Start learning
  </button>
  <p id="demo"> </p>
</body>
</html>
```



■ document.write() 함수의 사용 예 (1)

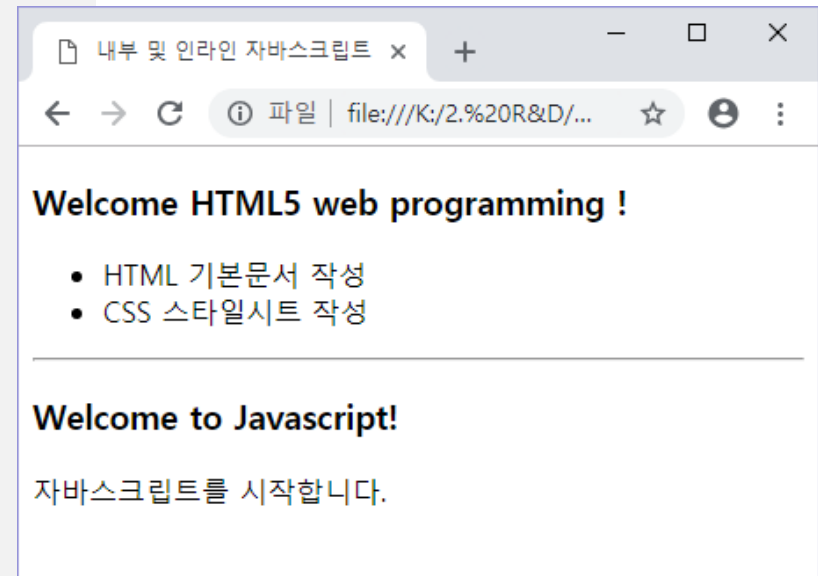
```
<head>
<script>
  function welcome() {
    document.write("<h3>Welcome to Javascript! </h3>");
    document.write("<p>자바스크립트를 시작합니다.</p>");
  }
</script>
</head>
<body>
  <h3> Welcome HTML5 web programming !</h3>
  <ul>
    <li> HTML 기본문서 작성 </li>
    <li> CSS 스타일시트 작성 </li>
  </ul> <hr>
  <button onclick="welcome();" > Next </button>
</body>
```



[주의] HTML 문서를 로딩된 이후 실행되는 document.write() 함수는 현재 웹브라우저 화면에 출력된 내용들을 모두 삭제하고 출력함

▪ document.write() 함수의 사용 예 (2)

```
<head>
<script>
  function welcome() {
    document.write("<h3>Welcome to Javascript! </h3>");
    document.write("<p>자바스크립트를 시작합니다.</p>");
  }
</script>
</head>
<body>
  <h3> Welcome HTML5 web programming !</h3>
  <ul>
    <li> HTML 기본문서 작성 </li>
    <li> CSS 스타일시트 작성 </li>
  </ul> <hr>
  <script>
    // <button onclick="welcome();" > Next </button>
    welcome();
  </script>
</body>
```



[주의] **HTML 문서를 로딩될 때 실행되는 document.write() 함수는 웹 문서의 내용들과 함께 웹브라우저 화면에 출력됨**

(5) 변수 정의 및 선언

- 변수(variable) : 자료를 저장하는 메모리 공간
- 변수의 사용
 - 미리 변수명을 정의하지 않고 필요한 부분에서 바로 사용할 수 있음
 - 변수명을 미리 정의할 경우, 자료형은 정의하지 않음
 - 변수의 자료형은 저장되는 값에 따라 자동으로 결정됨

```
❶ var 변수명; // 변수값을 지정하지 않는 변수 정의  
❷ var 변수명 = 값; // 변수명과 변수값의 지정  
❸ var 변수1 = 값1, 변수2, 변수3 = 값3, ... ; // 여러 변수들의 정의  
❹ var 객체변수명 = new 객체생성자함수(); //객체형 변수 정의
```



- 변수명의 정의
 - 숫자가 아닌 문자 또는 \$, _로 시작해야 하고, 공백문자를 포함할 수 없음(영문자, 수자, \$, _로 구성)
 - 변수명의 영문 대소문자들은 구별됨(예를 들면, 변수 kor, Kor, KOR 은 서로 다른 변수들임)
 - 변수명은 영문자 소문자들과 숫자들만으로 정의하고, 기억하기 쉽게 작성하는 것이 바람직함
 - 낙타체 변수명
 - 첫번째 단어는 소문자로 시작하고, 나머지 단어의 첫 글자는 대문자로 시작해서 연결함
 - 예 : myNewCar, scoreMinsu, scoreWoojin, nameStudent, nameProfessor 등

(6) 수식과 연산자

- 수식(expression) : 자바스크립트에서 어떤 한 값을 나타내는 것
- 변수, 상수, 함수, (수식), 수식 연산자 수식 -> 수식
- 자바스크립트의 연산자들

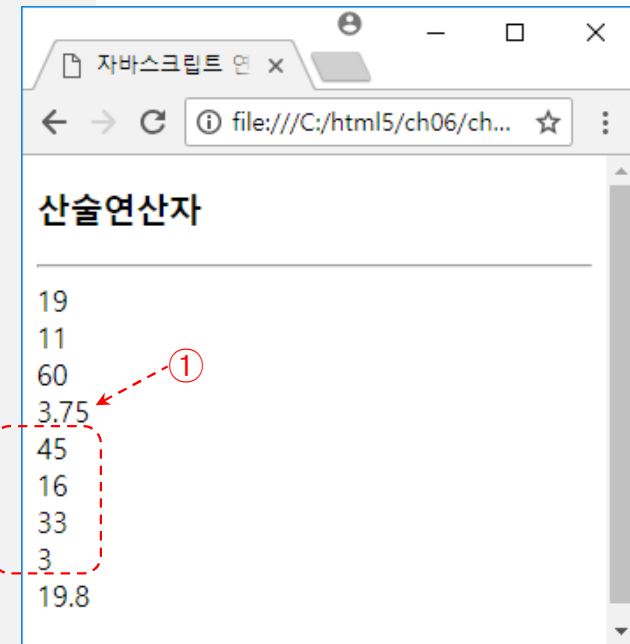
연산종류	연산자	참고
증감연산자	++, --	각각 변수값을 1씩 증가, 감소시킴
산술연산자	+, -, *, /, %	덧셈, 뺄셈, 곱셈, 나눗셈, 나머지연산
관계(비교)연산자	>, >=, <, <= ==, != ===, !==	크다, 크거나 같다, 작다, 작거나 같다 같다, 같지 않다 타입과 값이 같다, 타입과 값이 같지 않다
논리연산자	!, , &&	부정, 논리합, 논리곱
조건연산자	식 ? 값1 : 값2	식이 참이면 값1, 그렇지 않으면 값2
대입연산자	=, +=, -=, *=, /=, %=	치환, 덧셈/뺄셈/곱셈/나눗셈/나머지 치환
타입연산자	typeof,	피연산자의 자료형
인스턴스연산자	instanceof	피연산자 객체형의 인스턴스 여부

- 자바스크립트의 연산자들은 C/C++, Java 등의 언어와 크게 다르지 않음
- 단, /(나눗셈 연산자), +(덧셈 또는 문자연결 연산자), ===, !== 은 차이점이 있음

[예 6.10] 자바스크립트의 산술연산자와 산술식

```
<h3> 산술연산자 </h3> <hr>
<script>
  var num1 = 15, num2 = 4;
  var grade1 = 3.8, grade2 = 4.3, grade3 = 15.0;
  var str1 = "Javascript", str2="Programming", str3= "80";

  document.write( num1 + num2 + '<br>' );
  document.write( num1 - num2 + '<br>' );
  document.write( num1 * num2 + '<br>' );
  document.write( num1 / num2 + '<br>' );
  document.write( num1++ + 30 + '<br>' );
  document.write( num1 + '<br>' );
  document.write( --num2 + 30 + '<br>' );
  document.write( num2 + '<br>' );
  document.write(num1 + grade1 + '<br><br>' );
</script>
```



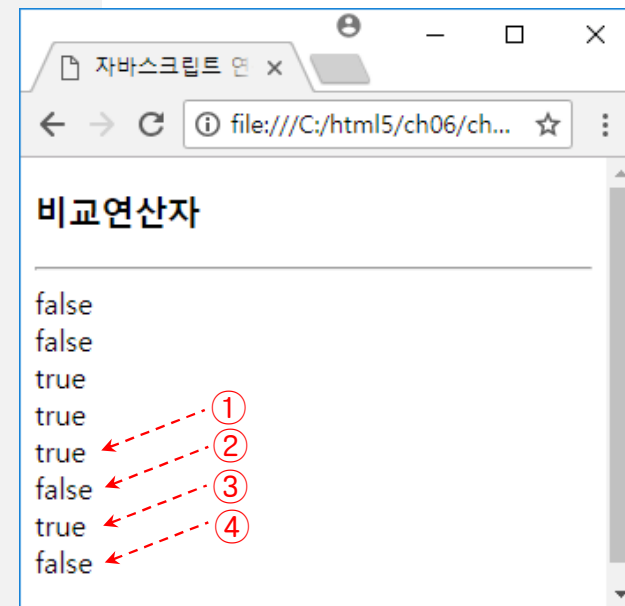
● 자바스크립트의 비교연산자

- 관계연산자들과 논리연산자들의 연산결과 : 수학의 관계식과 논리식의 연산 결과와 동일함
- 즉, 그 연산결과가 참이면 true, 거짓이면 false로 표현됨
- 피연산자의 값과 자료형을 동시에 검사하는 비교연산자들 : **===** , **!==**

[예 6.11]

```
<h3> 비교연산자 </h3> <hr>
<script>
  var num1 = 15, num2 = 4, num3 = 15;
  var grade = 15.0;
  var str= "80";

  document.write( (num1 <= num2) + '<br>' );
  document.write( (num1 !== num3) + '<br>' );
  document.write( (num3 == grade) + '<br>' );
  document.write( (num3 === grade) + '<br>' );
  document.write( (num3 == 15) + '<br>' );
  document.write( (num3 === "15") + '<br>' );
  document.write( (str == 80) + '<br>' );
  document.write( (str === 80) + '<br><br>' );
</script>
```



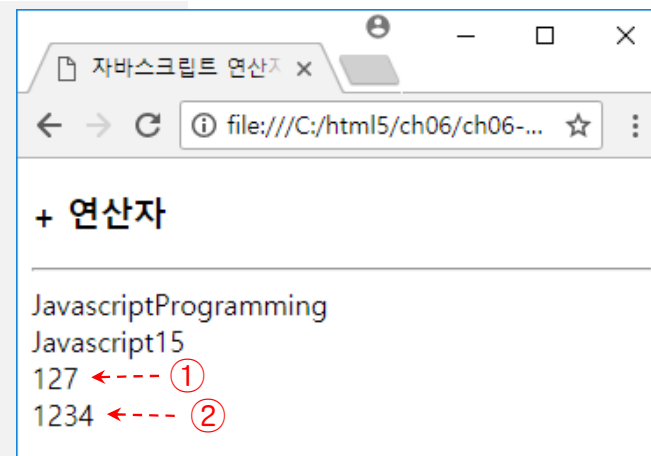
[참고] str="80"과 상수 80은 값만 고려하면 같으므로 ③의 연산결과는 true 이지만, 자료형까지 고려하면 문자열과 수치로 서로 다르므로 ④의 연산결과는 false 임

● + 연산자

- 두 피연산자가 수치형이면 덧셈 연산자임
- 두 피연산자들중 어느 하나라도 문자열이면 문자열 연결 연산자임

```
<h3> + 연산자 </h3> <hr>
<script>
  var num1 = 15, num2 = 4;
  var str1 = "Javascript", str2="Programming" ;

  document.write( str1 + str2 + '<br>' );
  document.write( str1 + num1 + '<br>' );
  document.write( 123 + num2 + '<br>' );
  document.write( "123" + num2 + '<br>' );
</script>
```



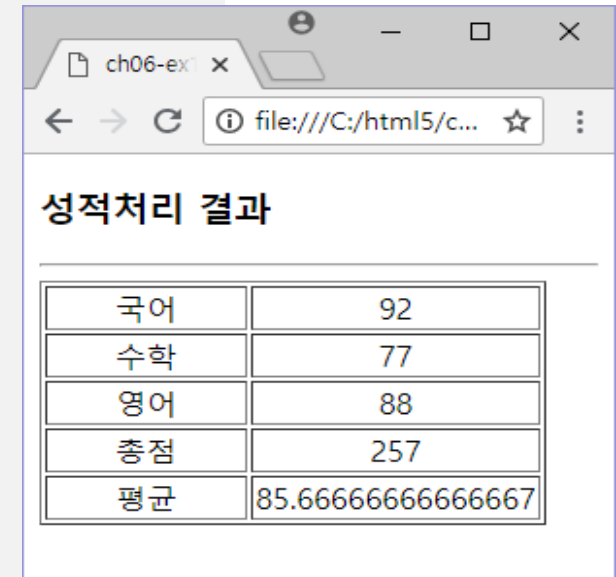
(7) 문장의 표현

- 문장(또는 명령문) : 특정 작업의 수행을 나타내는 것
- 한 명령문 의 끝에는 ;(세미콜론)을 붙임 (한 라인에 한 명령문을 작성한다면 ; 을 생략할 수 있음)
- 모든 수식은 명령문이 될 수 있음

[치환식 명령문] 변수 = 식; (예) x=y+5;

[함수식 명령문] 함수식(인수1,...); (예) document.write("welcometojavascript!");

```
<head> <style> td { width : 100px; text-align : center; } </style> </head>
<body>
  <h3> 성적처리 결과 </h3> <hr>
  <script>
    var kor=92, math=77, eng=88, total, avg;
    total = kor + math + eng; // 치환문
    avg = total / 3; // 치환문
    document.write('<table border=1>'); // 함수식 명령문
    document.write('<tr> <td>국어</td><td>' + kor + '</td></tr>'); // 함수문
    document.write('<tr> <td>수학</td><td>' + math + '</td></tr>'); // 함수문
    document.write('<tr> <td>영어</td><td>' + eng + '</td></tr>'); // 함수문
    document.write('<tr> <td>총점</td><td>' + total + '</td></tr>'); // 함수문
    document.write('<tr> <td>평균</td><td>' + avg + '</td></tr>'); // 함수문
    document.write('</table>'); // 함수문
  </script>
</body>
```



국어	92
수학	77
영어	88
총점	257
평균	85.66666666666667

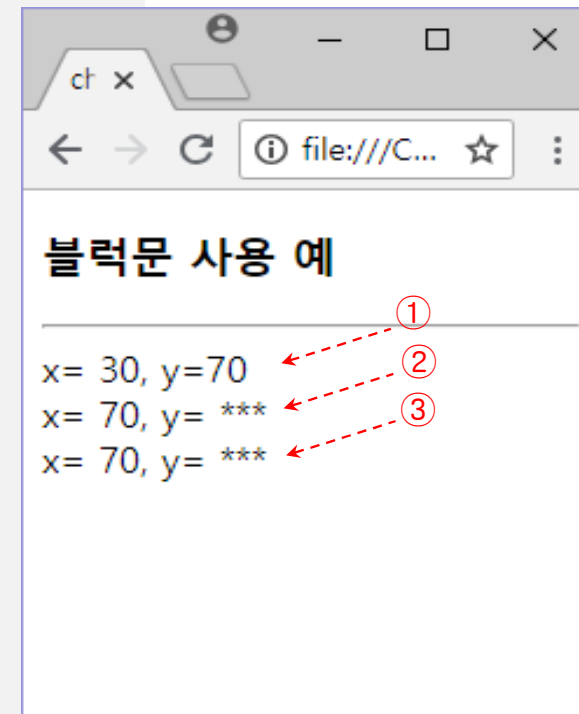
- 블록문(block statement)

- 여러 명령문들을 중괄호({ })로 그룹핑해서 나타낸 것
- 논리적으로 한 명령문으로 취급함

```
<h3> 블록문 사용 예 </h3> <hr>
<script>
  var x, y, temp;
  // ① 블록문 사용
  x=30, y=70, temp="***";
  if (x > y) { temp = x ; x = y ; y = temp ; }
  document.write("x= " + x + ", y= " + y + "<br>");

  // ②
  x=30, y=70, temp="***";
  if (x > y) temp = x ; x = y ; y = temp ;
  document.write("x= " + x + ", y= " + y + "<br>");

  // ③
  x=30, y=70, temp="***";
  if (x > y) { temp = x ; } x = y ; y = temp ;
  document.write("x= " + x + ", y= " + y + "<br>");
</script>
```



6.2.2 자료형

- 자료(data) : 컴퓨터로 처리해야 하는 현실세계의 대상들
- 자료형(data type) : 자료를 컴퓨터 메모리에 저장해서 표현하는 방식
- 컴퓨터 언어에서 자료형 구분
 - 메모리 크기와 저장 방식, 값들의 저장방식(유형) 등에 따라 구분함
 - 기본자료형(primitive data type)
 - ✓ 간단히 한 값으로 표현하는 되는 자료들 (예) 시험성적, 이름, 합격 여부 등
 - ✓ 문자열, 수치값, 논리값으로 표현
 - 객체 자료형 (object data type)
 - ✓ 여러 속성값들과 동작들을 그룹핑해서 표현하는 자료들 (예) 자동차, 학생, 학교
 - ✓ 속성(property)들과 메서드(method)들로 그룹핑해서 표현함
- 자바스크립트의 자료형 : 6가지
 - 기본 자료형 : string, number, boolean, undefined
 - 객체 자료형 : object, function

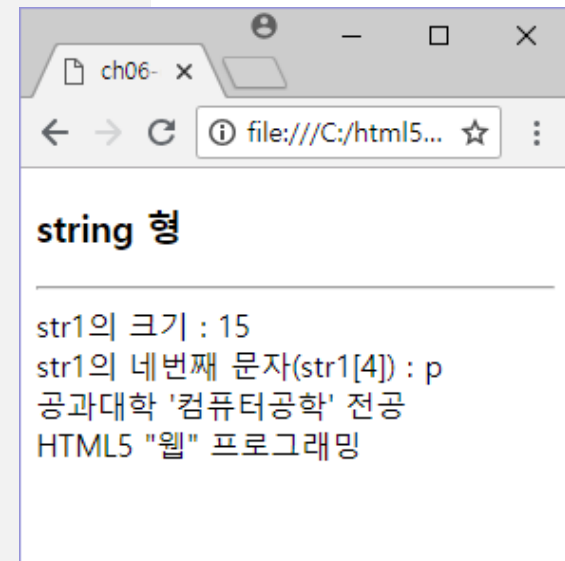
[표 6.3] 자바스크립트의 자료형

자료형	변수의 자료형	구 분	표현 예/방법
string	문자열	primitive type 또는 object	"문자들"
number	수치값(정수, 실수)	primitive type 또는 object	356, 3.14
boolean	참, 거짓의 논리값	primitive type 또는 object	true, false
object	호출 불가능한 일반 객체	object	객체 상수 또는 객체 변수
function	호출가능한 함수 객체	object	함수명() 또는 함수형 객체 변수
undefined	자료형이 정의되지 않음	primitive type	undefined

(1) string 형

- 문자열 즉, 일련의 문자들로 표현되는 자료들
- 문자열 리터럴(문자열 상수)은 작은 인용부호 또는 이중인용부호를 이용하여 나타냄
 - (예) 'html5', "javascript", "컴퓨터 '전공' 학생", '컴퓨터 "전공" 학생' 등
- 문자열은 내부적으로 시작인덱스가 0인 문자 배열로 저장됨
 - 문자열 배열의 length 속성 : 문자열 길이(문자열의 문자 개수)
 - 배열원소 연산자 [] : 문자열의 각 문자열 참조 가능
 - (예) "Javascript".length == 10(문자열 크기)
 - (예) "Javascript"[0] == 'J' , "Javascript"[4] == 's' , "Javascript"[9] == 't'

```
<h3> string 형 </h3> <hr>
<script>
  var str1 = "web programming";
  var str2 = "공과대학 '컴퓨터공학' 전공";
  var str3 = 'HTML5 "웹" 프로그래밍';
  document.write("str1의 크기 : ");
  document.write(str1.length );
  document.write("<br>");
  document.write("str1의 네번째 문자(str1[4]) : " + str1[4]);
  document.write("<br>");
  document.write(str2 + "<br>");
  document.write(str3 + "<br>");
</script>
```

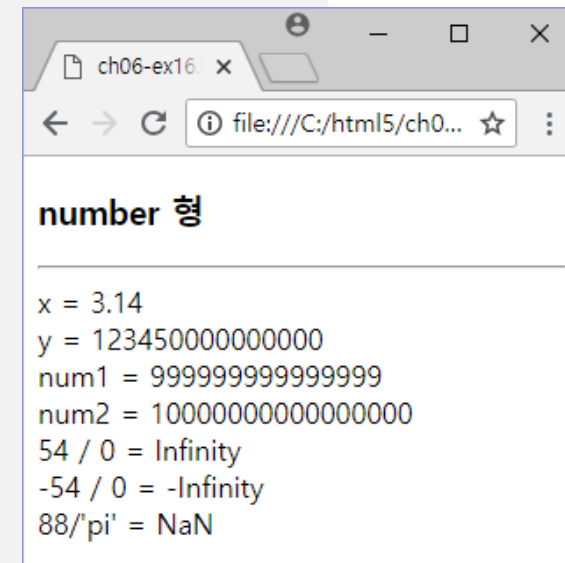


(2) number 형

- 정수 또는 실수 등의 수치값들
- 정수 : 소수점을 포함하지 않는 수치값
 - 10진수 또는 16진수(예 : 0xFF(=255) 등) 표현 사용
 - 최대 15자리까지의 정확도(precision) 표시, 그 이상을 넘으면 오버플로우됨
 -
- 실수 : 소수점을 포함하는 수치값
 - 10진수 (예: 3.14, 45.0 등) 또는 지수표현(예: 123e⁻³ 등) 사용
- 정수와 실수는 내부적으로는 모두 64비트 실수형(floating point)으로 저장됨
- number 형 예약어
 - NaN(Not a Number) : 자료값이 수치형이 아님
 - Infinity(또는 -Infinity) : 시스템에서 표현가능한 최대값(또는 최소값)을 벗어난 값임
 - 수치를 0으로 나누면 그 결과는 Infinity가 됨

```
<h3> number 형 </h3> <hr>
<script>
var x=314e-2, y=12345e10;
var num1=9999999999999999, num2 = 9999999999999999;

document.write("x = " + x + "<br>");
document.write("y = " + y + "<br>");
document.write("num1 = " + (num1) + "<br>");
document.write("num2 = " + (num2) + "<br>");
document.write("54 / 0 = " + (54/0) + "<br>");
document.write("-54 / 0 = " + (-54/0) + "<br>");
document.write("88/'pi' = " + ( 88/ 'pi'));
</script>
```

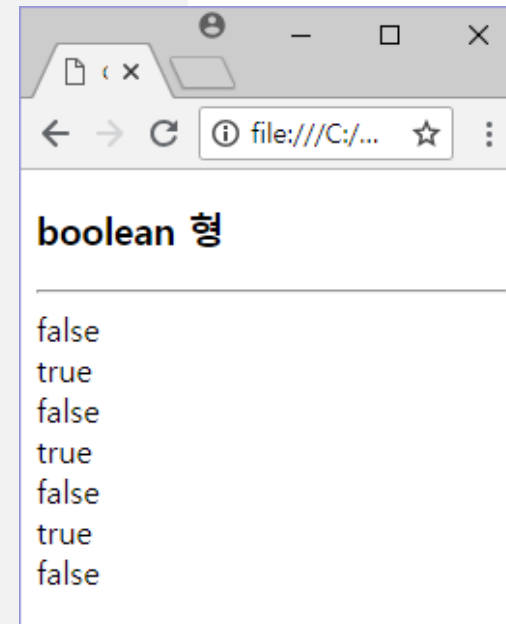


(3) boolean 형

- 참 또는 거짓의 논리값(true, false)을 나타내는 것
- 관계식이나 논리식으로 표현되고, 조건문과 반복문에서 실행조건을 표현하는데 사용됨
- 형변환 내장 함수 Boolean() 제공
 - ✓ 0, -0, 공백문자열(""), undefind 변수, null 객체 : false
 - ✓ 그 외 : true

```
<h3> boolean 형 </h3> <hr>
<script>
  var state, num1=0, num2=88;
  var str1=""; str2="Javascript"
  var obj1 = null;
  var obj2 = new Object();

  document.write(Boolean(num1) + '<br>');
  document.write(Boolean(num2) + '<br>');
  document.write(Boolean(str1) + '<br>');
  document.write(Boolean(str2) + '<br>');
  document.write(Boolean(obj1) + '<br>');
  document.write(Boolean(obj2) + '<br>');
  document.write(Boolean(state) + '<br>');
</script>
```

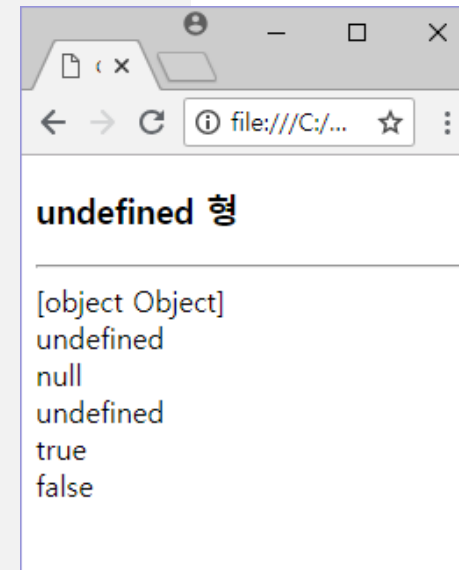


(4) undefined 형

- 변수가 자료형을 결정할 수 없음을 나타내는 것
- 변수명 선언후 아직 값을 저장하지 않았거나 변수값으로 undefined를 저장한 변수들
- undefined : 기본 자료형 변수가 비어있음을 나타냄(자료형: undefined)
- null : 객체형 변수가 비어있음을 나타냄(자료형: object)

```
<h3> undefined 형 </h3> <hr>
<script>
  var obj = new Object();
  var num;

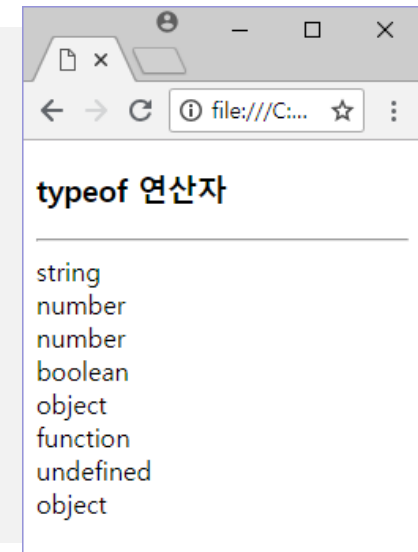
  document.write (obj + '<br>');
  document.write (num + '<br>');
  obj=null;
  document.write (obj + '<br>');
  obj=undefined ;
  document.write (obj + '<br>');
  document.write( (undefined == null ) + '<br>');
  document.write( (undefined === null));
</script>
```



(5) typeof 연산자와 형변환

- typeof 연산자 : 연산결과로 인수의 자료형을 나타냄

```
<h3> typeof 연산자 </h3> <hr>
<script>
  document.write(typeof "John" + '<br>'); // 문자열
  document.write(typeof 3.14 + '<br>'); // 실수
  document.write(typeof NaN + '<br>'); // NaN
  document.write(typeof false + '<br>'); // 논리형
  document.write(typeof [1,2,3,4] + '<br>'); // 배열
  document.write(typeof function () {} + '<br>'); // 함수
  document.write(typeof myCar + '<br>'); // 정의되지 않은 변수
  document.write(typeof null); // null
</script>
```



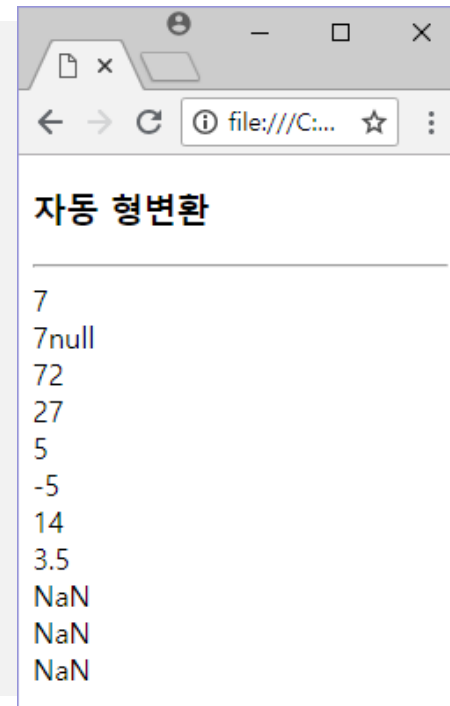
- 형변환(type conversion)

- 특정 자료형의 값을 다른 자료형의 값으로 변환 해서 나타내는 것
- 자동 형변환
 - ✓ 연산자들은 일정한 규칙을 가지고 피연산자들에 대해 자동으로 형변환을 수행해서 연산을 수행함
- 명시적 형변환 : 형변환 함수 이용

형변환 메서드	기능	예
String()	수치값을 문자열로 변환함	String(123)→"123"
Number()	문자열을 수치로 변환함	Number("3.14")→3.14 Number("88")→88
Boolean()	비논리형 값을 논리값으로 변환함	Boolean(0)→false Boolean(3.14)→true
parseFloat()	정수 또는 숫자로 구성된 문자열을 실수로 변환함	parseFloat("3.14")→3.14
parseInt()	실수 또는 숫자로 구성된 문자열을 정수로 변환함	parseInt("92")→92
toString()	수치값을 문자열로 변환함	(123).toString()→"123"

[예 6.20] 자동 형변환

```
<h3> 자동 형변환 </h3> <hr>
<script>
  document.write((7 + null) + '<br>');
  document.write(("7" + null) + '<br>');
  document.write(("7" + 2) + '<br>');
  document.write((2 + "7" ) + '<br>');
  document.write(("7" - 2) + '<br>');
  document.write((2 - "7") + '<br>');
  document.write(("7" * "2") + '<br>');
  document.write(("7" / "2") + '<br>');
  document.write(("abc" - 2) + '<br>');
  document.write(("abc" * 2) + '<br>');
  document.write(("abc" / 2) + '<br>');
</script>
```

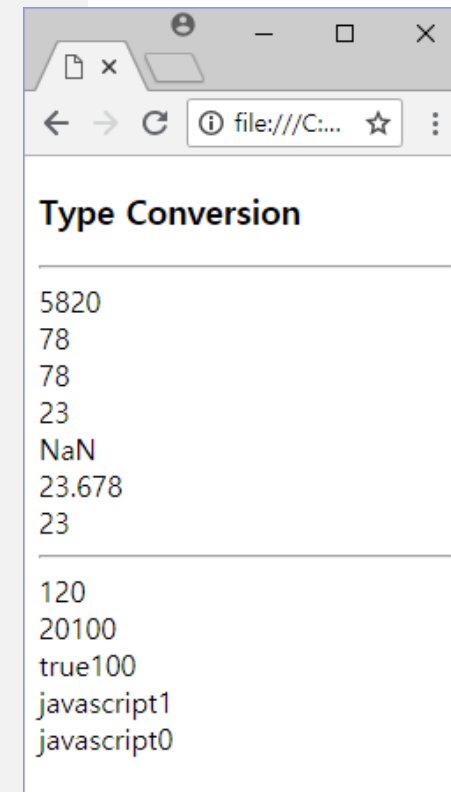


[예 6.21] 형변환 메서드

```
<h3> Type Conversion </h3> <hr>
<script>
  var num1= 3.145, num2=20, num3=100;
  var str1="58", str2="3.678", str3="javascript";
  var flag1=true, flag2=false;

  document.write( str1 + num2 + '<br>');
  document.write( Number(str1) + num2 + '<br>');
  document.write( parseInt(str1) + num2 + '<br>');
  document.write( parseInt(str2) + num2 + '<br>');
  document.write( parseInt(str3) + num2 + '<br>');
  document.write( parseFloat(str2) + num2 + '<br>');
  document.write( parseInt(num1) + num2 + '<br><hr>');

  document.write( num2 + num3 + '<br>');
  document.write( num2 + num3.toString() + '<br>');
  document.write( Boolean(num2) + num3.toString() + '<br>');
  document.write( str3 + Number(flag1) + '<br>');
  document.write( str3 + Number(flag2) + '<br>');
</script>
```



6.2.3 이벤트 속성

- 자바스크립트는 이벤트 처리(event processing)를 지원함
- 이벤트(event)
 - 웹 문서를 로드한 웹 브라우저에서 발생하는 어떤 동작이나 상태변화를 나타내는 것
 - 웹 브라우저에 로드된 HTML 문서와 사용자 사이의 상호작용에 의해 발생함
- 이벤트 예
 - 사용자가 마우스로 특정 태그 영역을 클릭함(click 이벤트)
 - 사용자가 마우스를 특정 태그 영역 위로 이동시킴(mouseover 이벤트)
 - 특정 <input> 태그의 사용자 입력값이 변경됨(change 이벤트)
 - 이미지 파일 또는 HTML 문서가 웹 브라우저로의 로드를 마침(load 이벤트)
 - 사용자가 브라우저 창의 크기를 변경함(resize 이벤트)
- 이벤트 처리
 - 이벤트 발생 -> (웹브라우저) -> 이벤트 처리함수(이벤트핸들러) 호출 & 실행 -> 이벤트 처리
 - 이벤트 등록 : 이벤트와 이벤트핸들러를 연결하는 과정
 - 자바스크립트는 이벤트 등록을 위해 이벤트 속성들을 지원함
 - 이벤트 속성
 - ✓ 자바스크립트 코드를 속성값으로 갖고, 이벤트가 발생하면 자바스크립트 코드를 실행함
 - ✓ 속성이름은 "on이벤트명" 형식임

[표 6.5] 자주 사용하는 마우스 관련 이벤트 속성

이벤트 분류	이벤트 속성	이벤트 의미
마우스 관련	onclick	마우스를 클릭함
	onmouseover	마우스가 해당 영역 위에 위치함
키보드 관련	onkeypress	키보드를 누른 상태임
	onkeydown	한 키를 입력함
문서 관련	onload	HTML 문서가 웹 브라우저에 적재됨
	onunload	HTML 문서가 웹 브라우저에서 제거됨
폼(<form>) 관련	onchange	사용자가 입력값을 변경함
	onfocus	사용자가 데이터를 입력할 수 있는 상태임

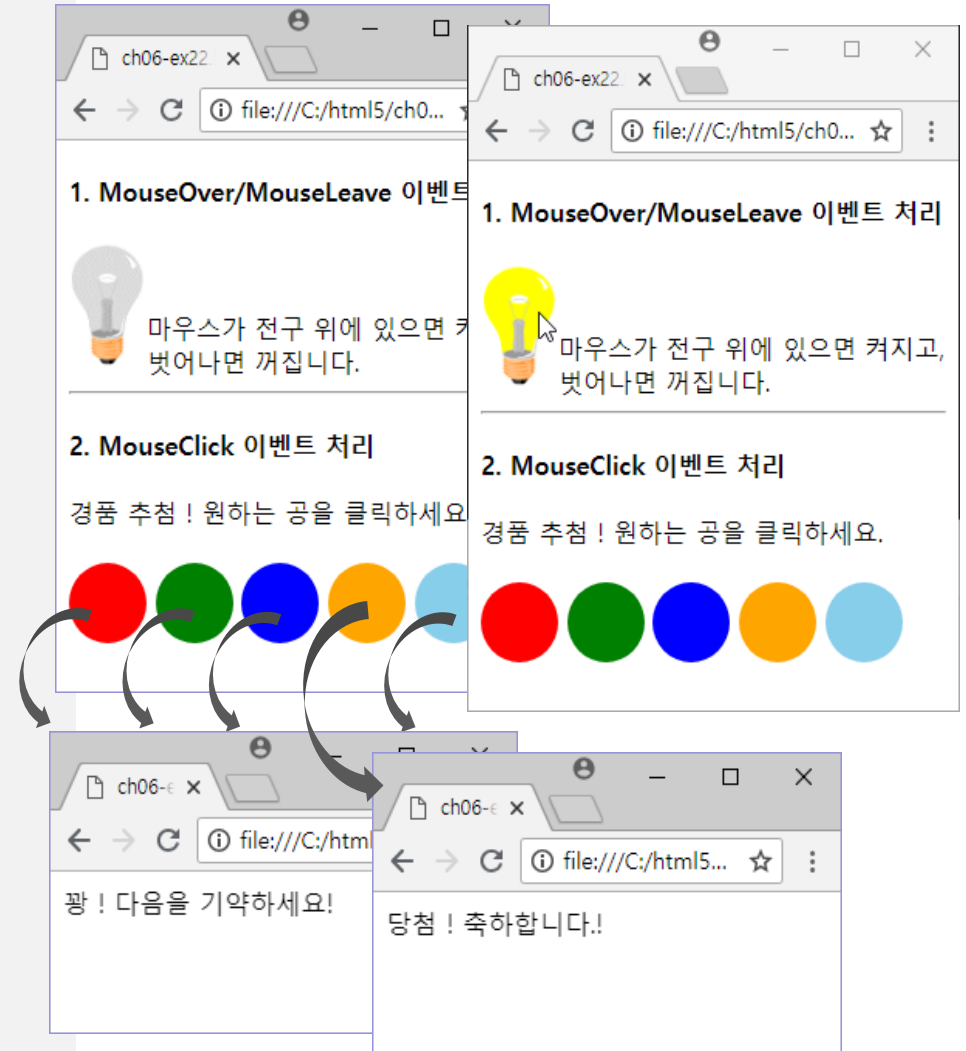
- 자바스크립트의 이벤트 처리 방법 : 이벤트 속성값으로 "이벤트처리 코드"를 지정함

<HTML태그 이벤트속성="자바스크립트 코드 (이벤트리스너)" ... > </HTML태그>

예 <h1 onclick="document.write ('마우스 클릭을 했음!');" > 타이틀 </h1>

[예 6.22] 이벤트 속성의 사용 예

```
<script>
  var str1 = "꽝 ! 다음을 기약하세요!";
  var str2 = "당첨 ! 축하합니다.! ";
</script>
<style>
  div { width: 50px; height:50px; border-radius: 50%;
        display: inline-block; }
  img { width: 50px; height: 80px; float : left; }
  #b1 { background: red; } #b2 { background: green; }
  #b3 { background: blue; } #b4 { background: orange; }
  #b5 { background: skyblue; }
</style>
....
<h4> 1. MouseOver/MouseLeave 이벤트 처리 </h4>
 <br> <br>
마우스가 전구 위에 있으면 켜지고, 벗어나면 꺼집니다. <hr>
<h4>2. MouseClick 이벤트 처리 </h4>
경품 추첨 ! 원하는 공을 클릭하세요.<br> <br>
<div id="b1" onclick="document.write(str1);"> </div>
<div id="b2" onclick="document.write(str1);"> </div>
<div id="b3" onclick="document.write(str1);"> </div>
<div id="b4" onclick="document.write(str2);"> </div>
<div id="b5" onclick="document.write(str1);"> </div>
```



6.2.4 입력 함수

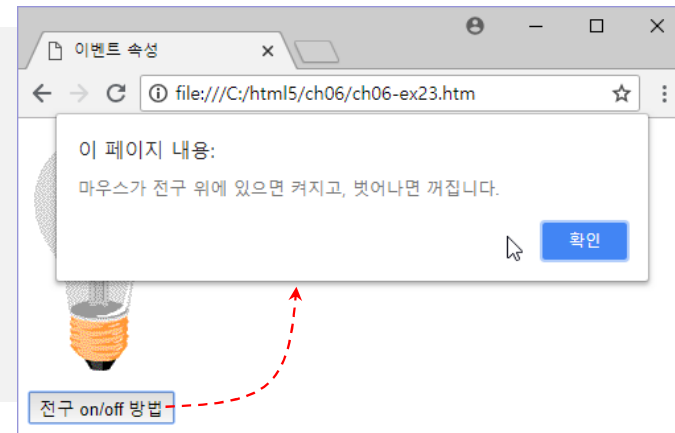
- 자바스크립트는 일반적으로 HTML 입력양식을 통해 입력된 데이터들을 처리함
- 직접 키보드를 통해 데이터를 입력받아 처리하는 경우는 극히 제한적 임
- 메시지 대화상자를 통해 직접 사용자들로부터 입력받는 함수 제공
 - alert() : 경고(안내) 메시지 확인 응답 입력
 - confirm() : yes 또는 no 의 응답 입력
 - prompt() : 문자열 입력

(1) alert() 함수

- 안내메시지, 경고메시지 등을 나타내는 대화상자를 출력하고 사용자가 [확인(OK)] 버튼을 누르면 대화상자를 종료함

```
<script>
  var txt = "마우스가 전구 위에 있으면 켜지고,
            벗어나면 꺼집니다.";
</script>

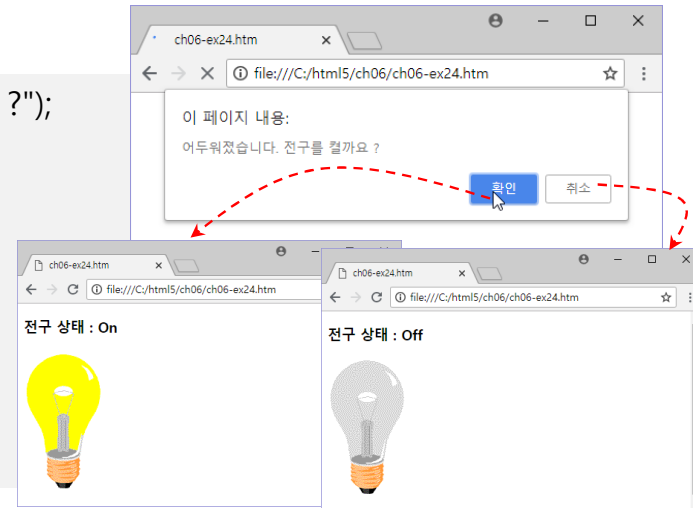
...
 <br>
<button onclick="alert(txt)"> 전구 on/off 방법</button>
```



(2) confirm() 함수

- 대화상자를 통해 사용자에게 확인(Yes, OK) 또는 취소(No, Cancel) 의 응답을 요구하는 메시지를 제공하고 응답을 받아 반환함

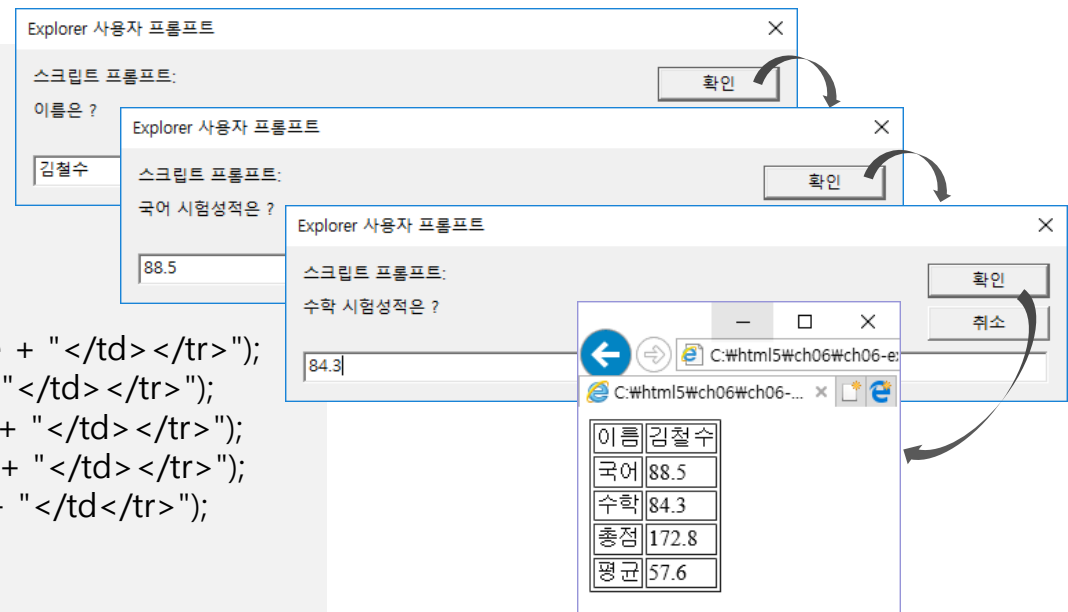
```
var answer = confirm("어두워졌습니다. 전구를 켜까요 ?");  
if (answer == true) {  
    document.write("<h3>전구 상태 : On</h3>");  
    document.write("<img src='lighton.png' />");  
}  
else {  
    document.write("<h3>전구 상태 : Off</h3>");  
    document.write("<img src='lightoff.png' />");  
}  
</script>
```



(3) prompt() 함수

- 대화상자를 통해 사용자에게 문자열의 입력을 요구하고 사용자가 입력 한 문자열을 반환함
- 수치값들은 문자열 변수에 저장한 후, 수치형으로 변환해서 사용해야 함
 - 문자열 -> 수치형 변환함수들 : parseInt(), parseFloat(), Number()

```
<script>
var name = prompt("이름은 ?");
var kor = prompt("국어 시험성적은 ?");
var math = prompt("수학 시험성적은 ?");
total = Number(kor) + Number(math) ;
avg = total /3;
document.write("<table border=1 >");
document.write("<tr><td>이름</td> <td>" + name + "</td></tr>");
document.write("<tr><td>국어</td> <td>" + kor + "</td></tr>");
document.write("<tr><td>수학</td> <td>" + math + "</td></tr>");
document.write("<tr><td>총점</td> <td>" + total + "</td></tr>");
document.write("<tr><td>평균</td> <td>" + avg + "</td></tr>");
document.write("</table>");
</script>
```

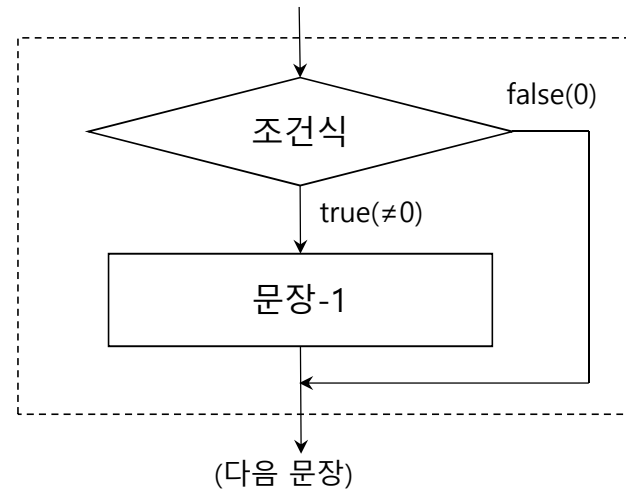


6.2.5 제어문

- 명령문들의 실행 순서, 즉 제어 구조를 나타냄
- 제어문 종류
 - 순차 구조 : 명령문들을 순차적 나열(별도의 순차구조 제어문은 없음)
 - 선택문 : if, if~else , if~else if, switch
 - 반복문 : while, for, do~while, for~in
 - 그 외 : continue, break

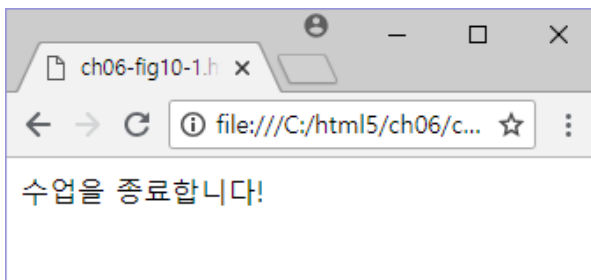
▪ if 문 : 단일 선택 구조

```
if (조건식) { 문장-1; ... }
```

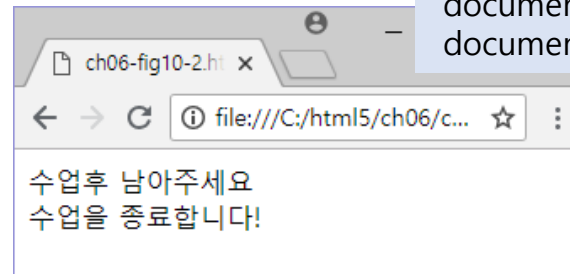


```
if (score <= 70) {
    document.write("추가과제를부여합니다.");
    document.write("수업후남아주세요.<br>");
}
document.write("수업을종료합니다!");
```

```
if (score <= 70) {
    document.write("추가과제를부여합니다.");
}
document.write("수업후남아주세요.<br>");
document.write("수업을종료합니다!");
```



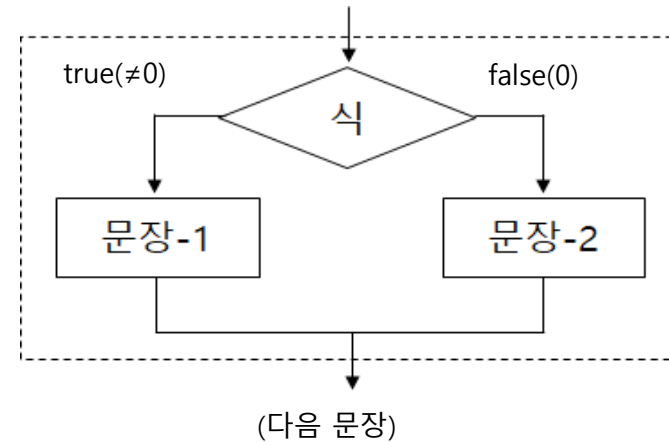
가) [예 1]의 실행결과(score=88)



나) [예 1]의 실행결과(중괄호 생략, score=88)

▪ if ~ else 문 : 양자택일 구조

```
if (조건식) { 문장-1; ... }  
else { 문장-2; ... }
```



```
if (score >= 80) {  
    document.write("합격 입니다");  
    document.write("추가과제는 없습니다.<br>");  
} else {  
    document.write("불합격 입니다");  
    document.write("추가과제가 있습니다.<br>");  
}  
document.write("수업을 종료합니다!");
```

▪ 다중 선택 구조

[예 3] 다중 선택의 예

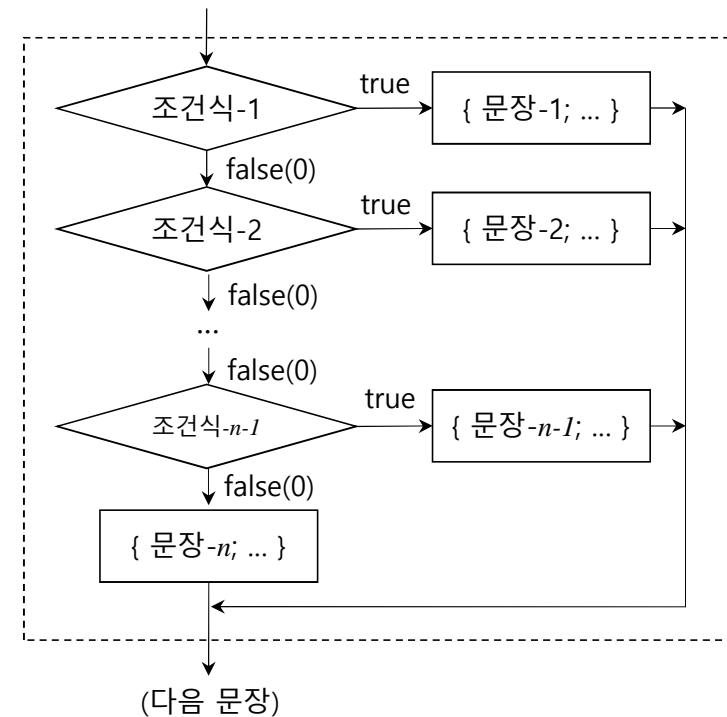
```
{시험성적(score)이
  90점 이상이면, ❶ 학점(grade)은 A 이다.
  80~89점 사이 이면, ❷ 학점(grade)은 B 이다.
  70~79점 사이 이면, ❸ 학점(grade)은 C 이다.
  60~69점 사이 이면, ❹ 학점(grade)은 D 이다.
  그 외, ❺ 학점(grade)은 F 이다.
}
```

[예 4] if, if~else를 이용한 표현

```
{
  if (score >= 90) grade="A";
  if (score < 90 && score >= 80) grade="B";
  if (score < 80 && score >= 70) grade="C";
  if (score < 70 && score >= 60) grade="D";
  else grade="F";
}
```

▪ if~else if 문

```
if (조건식-1) { 문장-1; ... }
else if (조건식-2) { 문장-2; ... }
else if (조건식-3) { 문장-3; ... }
...
else if (조건식-n-1) { 문장-n-1; ... }
else { 문장-n; ... }
```



- **if~else if 문**

[예 5] 배타적인 조건식 사용

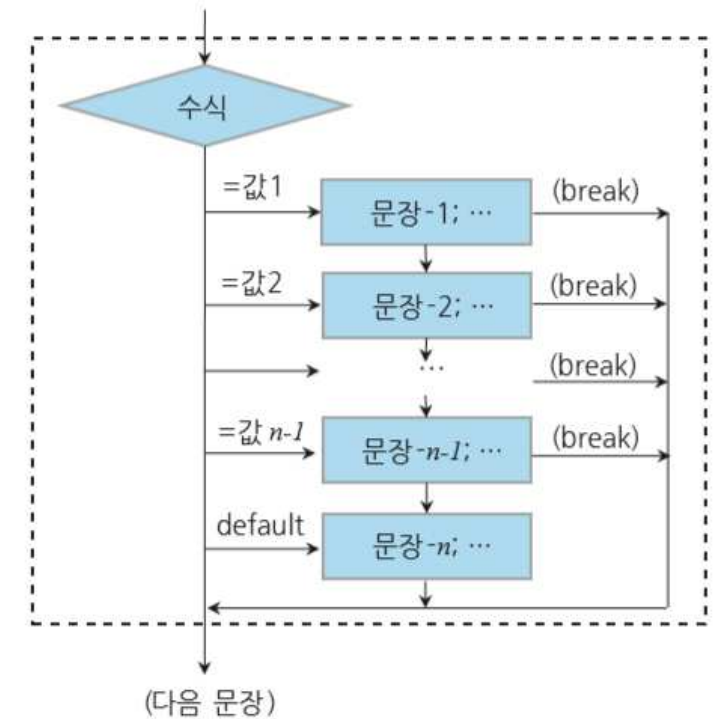
```
if (score >= 90) grade = "A";  
else if (score < 90 && score >= 80) grade = "B";  
else if (score < 80 && score >= 70) grade = "C";  
else if (score < 70 && score >= 60) grade = "D";  
else grade = "F";
```

[예 6] 나열 순서를 고려한 조건식 사용

```
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";
```


▪ switch 문

```
// break문은생략할수있음
switch(수식) {
    case 값1 : 문장-1; ... ; (break;)
    case 값2 : 문장-2; ... ; (break;)
    ...
    case 값n : 문장-n; ... ; (break;)
    default : 문장-n ; ... ;
}
```



[예 7] switch 문(1)

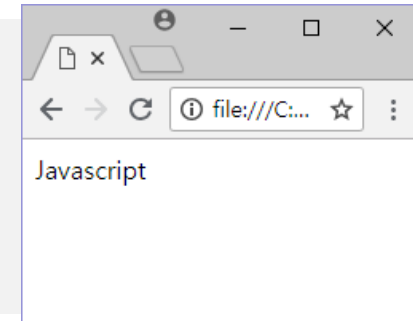
```
switch (parseInt(score/10)) {
    case 10: grade="A"; break;
    case 9: grade="A"; break;
    case 8: grade="B"; break;
    case 7: grade="C"; break;
    case 6: grade="D"; break;
    default: grade="F";
}
```

[예 7] switch 문(2)

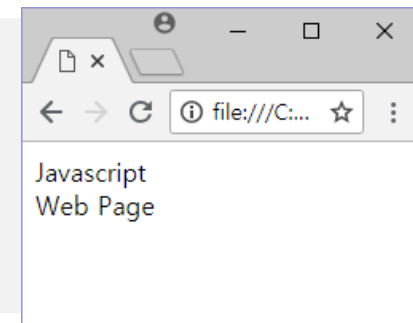
```
switch (parseInt(score/10)) {
    case 10, 9: grade="A"; break;
    case 8: grade="B"; break;
    case 7: grade="C"; break;
    case 6: grade="D"; break;
    default: grade="F";
}
```

[예 6.26] switch 문의 실행결과 비교(choice=2)

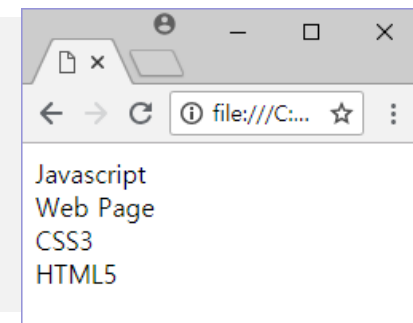
```
① switch (choice) {  
    case 0: document.write("HTML5 <br>"); break;  
    case 1: document.write("CSS3 <br>"); break;  
    case 2: document.write("Javascript <br>"); break;  
    default: document.write("Web Page <br>");  
}
```



```
② switch (choice) {  
    case 0: document.write("HTML5 <br>");  
    case 1: document.write("CSS3 <br>");  
    case 2: document.write("Javascript <br>");  
    default: document.write("Web Page <br>");  
}
```



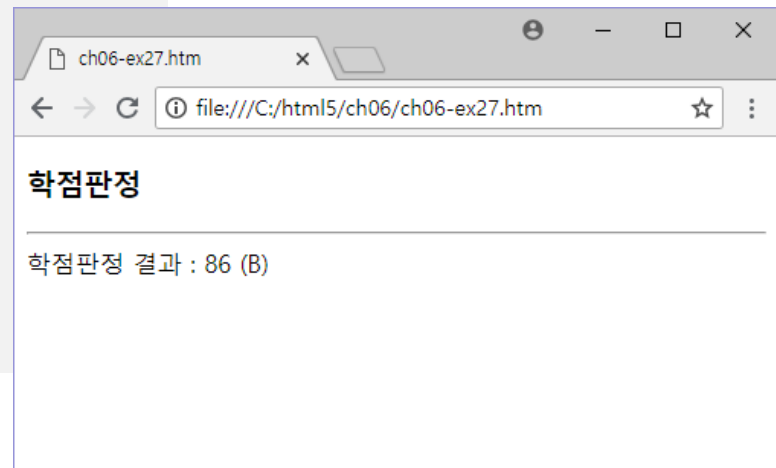
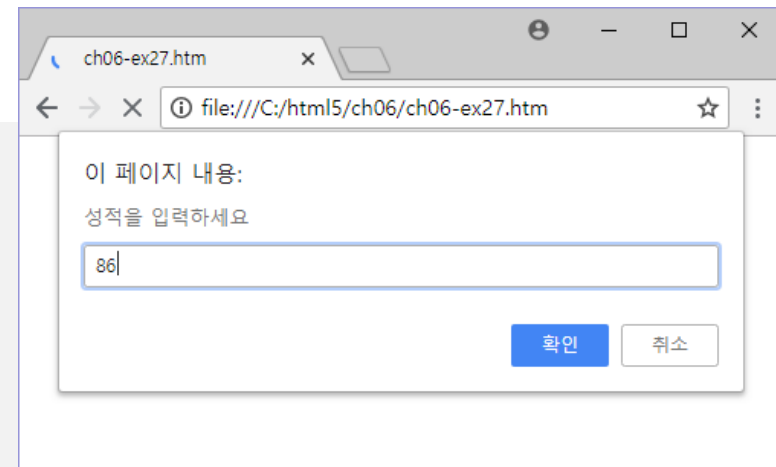
```
③ switch (choice) {  
    case 2: document.write("Javascript <br>");  
    default: document.write("Web Page <br>");  
    case 1: document.write("CSS3 <br>");  
    case 0: document.write("HTML5 <br>");  
}
```



[예 6.27] switch 문을 이용한 학점 판정

```
<script>
  var score;
  var grade;

  document.write("<h3> 학점판정 </h3>");
  score = prompt("성적을 입력하세요");
  switch (parseInt(score/10)) {
    case 10, 9: grade="A"; break;
    case 8: grade="B"; break;
    case 7: grade="C"; break;
    case 6: grade="D"; break;
    default: grade="F";
  }
  document.write("<hr>학점판정 결과 : ");
  document.write(score + " (" + grade + ")");
</script>
```



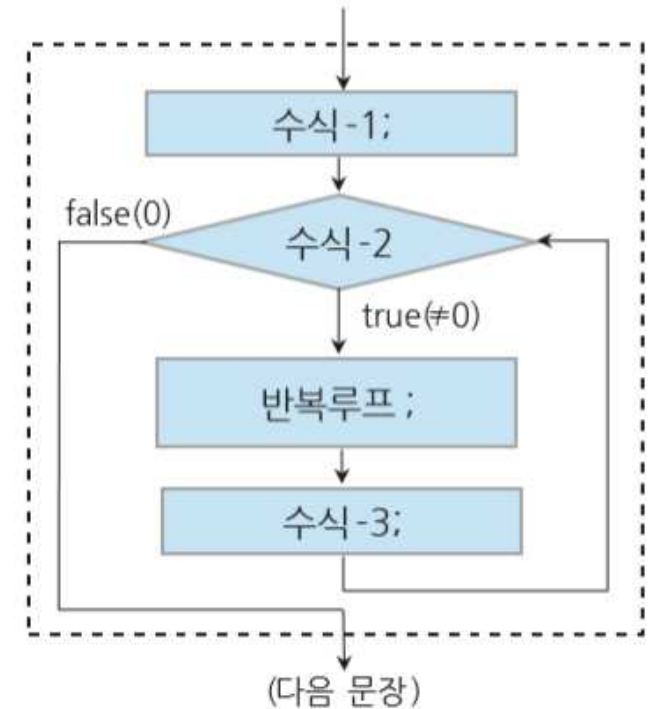
(2) 반복문

- for 문

```
for (수식-1; 수식-2; 수식-3) {  
    반복루프;  
}
```

- 3개의 수식들 이용해서 반복루프의 반복횟수를 나타냄
- 실행 순서

```
수식-1 →  
수식-2 (true 또는 ≠0) → 반복루프 → 수식-3 →  
수식-2 (true 또는 ≠0) → ... → 수식-3 →  
수식-2 (false 또는 0)
```



[문장 document.write("welcome to javascript!");의 10회 반복]

- ❶ // i=1,2,...,10일 때까지 반복하고, i=11 이 되면 종료함
for (i=1; i<=10; i=i+1) { document.write("welcometojavascript!"); }
- ❷ // j=1,3,...,19 까지 반복하고, i=21 이 되면 종료함
for (j=1; j<20; j=j+2) { document.write("welcometojavascript!"); }

- 반복제어 변수는 반복횟수와 반복 루프 내에서의 사용을 함께 고려해서 변경해야 함

③ // 1 부터 100 사이의 모든 짝수들의 합 계산

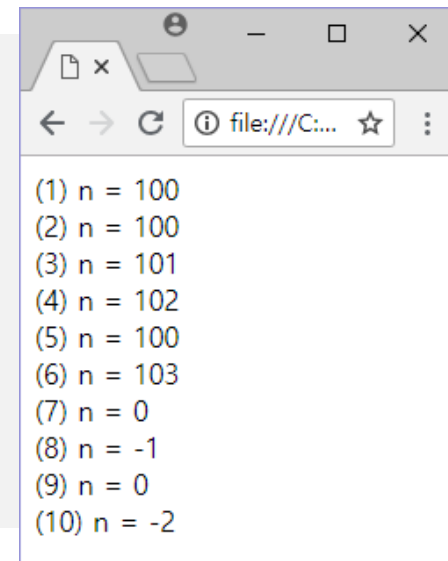
```
for (sum=0, i=2; i<=100; i=i+2) { sum=sum+i; } // 50회 반복 ( i = 2, 4, ... , 100)
```

④ // 1 부터 100 사이의 5의 배수들의 합 계산

```
for (sum=0, i=5; i<=100; i=i+5) { sum=sum+i; } // 20회 반복 ( i = 5, 10, 15, ... , 100)
```

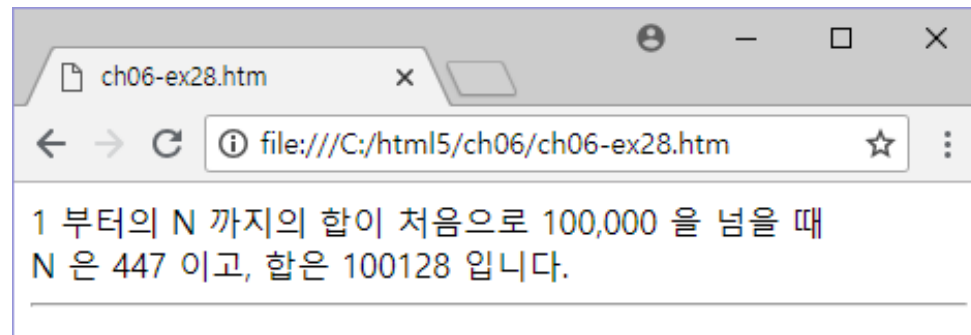
- for 문의 실행이 끝난 직후 반복제어변수 값을 정확히 예측할 수 있어야 함

```
<script>
  for (n=0; n<100; n++) ; document.write("(1) n = " + n + "<br>");
  for (n=0; n<100; n++) n++; document.write("(2) n = " + n + "<br>");
  for (n=0; n<=100; n++); document.write("(3) n = " + n + "<br>");
  for (n=0; n<=100; n=n+2); document.write("(4) n = " + n + "<br>");
  for (n=1; n<100; n++); document.write("(5) n = " + n + "<br>");
  for (n=1; n<=100; n++) n= n+2; document.write("(6) n = " + n + "<br>");
  for (n=100; n>0; n--); document.write("(7) n = " + n + "<br>");
  for (n=100; n>=0; n--); document.write("(8) n = " + n + "<br>");
  for (n=100; n>0; n--) n--; document.write("(9) n = " + n + "<br>");
  for (n=100; n>=0; n--) n=n-2 ; document.write("(10) n = " + n + "<br>");
</script>
```



[예 6.28] 1부터 N까지의 합이 100,000을 넘을 때의 N과 합 구하기

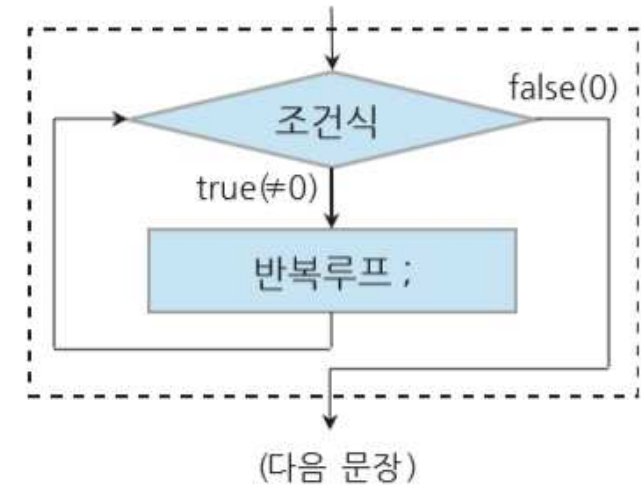
```
<script>
  sum=0;
  for (i=1; sum<=100000; i=i+1) {
    sum = sum + i;
  }
  document.write("1 부터의 N 까지의 합이 처음으로 100,000 을 넘을 때 <br> ");
  document.write("N 은 " + (i-1) + " 이고, 합은 " + sum + " 입니다. <hr>");
</script>
```



- while 문

- 조건식을 통해 반복루프의 반복 여부를 나타냄
- 조건식을 검사해서 참이면 반복루프를 실행하는 과정을 반복하고, 조건식이 거짓이 되면 다음 문장으로 이동함

```
while (조건식) {  
    반복루프;  
}
```



[문장 document.write("welcome to javascript!");의 10회 반복]

```
❶ i=1;  
while (i<=10) {  
    document.write("welcometojavascript!");  
    i=i+1;  
}
```

```
❷ i=1;  
while (i<20) {  
    document.write("welcometojavascript!");  
    i=i+2;  
}
```

[예 10] for 문과 while 문의 비교

```
// ❶ while 문
sum=0; i=1;
while (i<=100) {
    sum = sum+i;
    i = i+1;
}
```

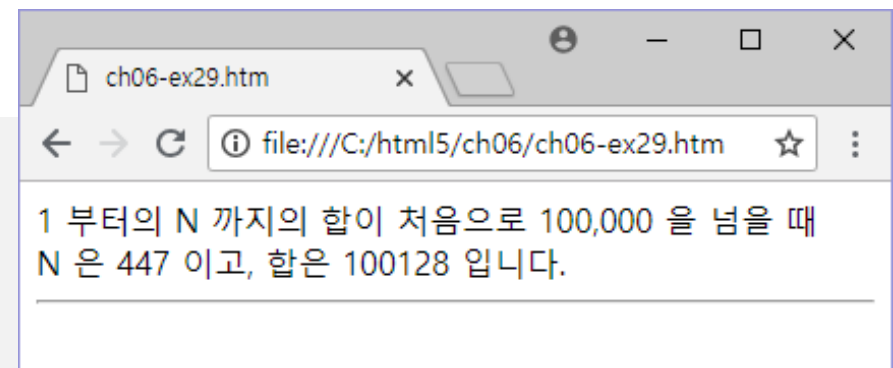
```
// ❷ for 문(1)
sum=0;
for (i=0; i<=100; i=i+1) {
    sum = sum+i;
}
```

```
// ❸ for 문(2)
sum=0; i=0;
for ( ; i<=100; ) {
    sum = sum+i;
    i = i+1;
}
```

[예 6.29]

1부터 N까지의 합이 100,000을 넘을 때의 N과 합 구하기

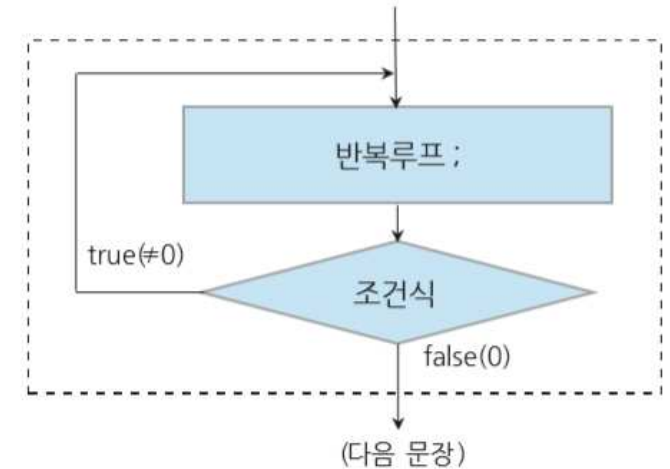
```
<script>
    sum=0; i=0;
    while (sum <= 100000 ) {
        i = i + 1;
        sum = sum + i;
    }
    document.write("1 부터의 N 까지의 합이 처음으로 100,000 을 넘을 때 <br> ");
    document.write("N 은 " + i + " 이고, 합은 " + sum + " 입니다. <hr>");
</script>
```



● do~while 문

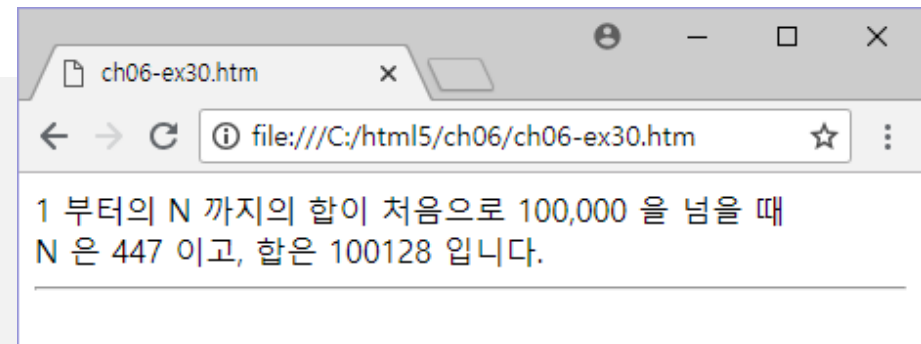
```
do {  
    반복루프;  
} while (조건식);
```

- while 문과는 달리 먼저 반복루프를 실행한 후에 조건식을 검사함
- do~while 문은 반복루프를 적어도 한번은 실행함
- 후조건 반복 구조



[예 6.30]

```
<script>  
    sum=0; i=0;  
    do {  
        i = i + 1;  
        sum = sum + i;  
    } while ( sum <= 100000 )  
    document.write("1 부터의 N 까지의 합이 처음으로 100,000 을 넘을 때 <br> ");  
    document.write("N 은 " + i + " 이고, 합은 " + sum + " 입니다. <hr>");  
</script>
```



(비교) 1부터 N까지의 합이 100,000을 넘을 때의 N과 합 구하기

```
<script>
  sum=0;
  for (i=1; sum<=100000; i=i+1) {
    sum = sum + i;
  }
  document.write("1 부터의 N 까지의 합이 처음으로 100,000 을 넘을 때 <br> ");
  document.write("N 은 " + (i-1) + " 이고, 합은 " + sum + " 입니다. <hr>");
</script>
```

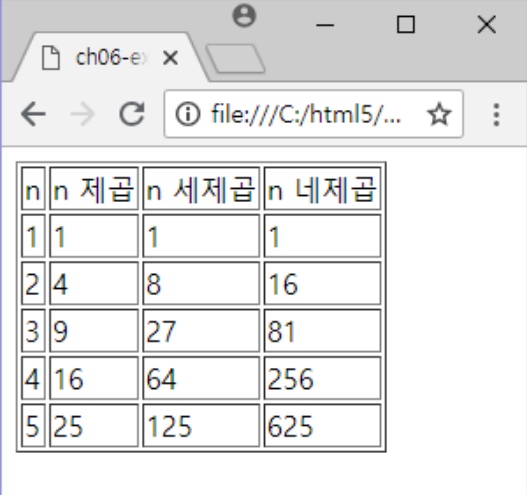
```
<script>
  sum=0; i=0;
  while (sum <= 100000 ) {
    i = i + 1;
    sum = sum + i;
  }
  document.write("1 부터의 N 까지의 합이 처음으로 100,000 을 넘을 때 <br> ");
  document.write("N 은 " + i + " 이고, 합은 " + sum + " 입니다. <hr>");
</script>
```

```
<script>
  sum=0; i=0;
  do {
    i = i + 1;
    sum = sum + i;
  } while ( sum <= 100000 )
  document.write("1 부터의 N 까지의 합이 처음으로 100,000 을 넘을 때 <br> ");
  document.write("N 은 " + i + " 이고, 합은 " + sum + " 입니다. <hr>");
</script>
```

- 반복문의 중첩

- 반복문의 반복루프에 또 다른 반복문을 중첩해서 사용할 수 있음
- 1~2회 정도의 중첩으로 제한해서 사용하는 것이 바람직함
- 동일한 반복문을 중첩해서 사용하는 것이 바람직함

```
<script>
// for 문의 중첩
document.write("<table border=1 text-align=center>");
document.write("<tr><td> n </td><td>n 제곱</td><td>n 세제곱</td><td>n 네제곱</td></tr>");
for (n=1; n<=5; n++) {
    document.write("<tr>");
    result = n;
    document.write("<td>" + result + "</td>");
    for (k=2; k<5; k++) {
        result = result * n ;
        document.write("<td>" + result + "</td>");
    }
    document.write("</tr>");
}
document.write("</table>");
</script>
```



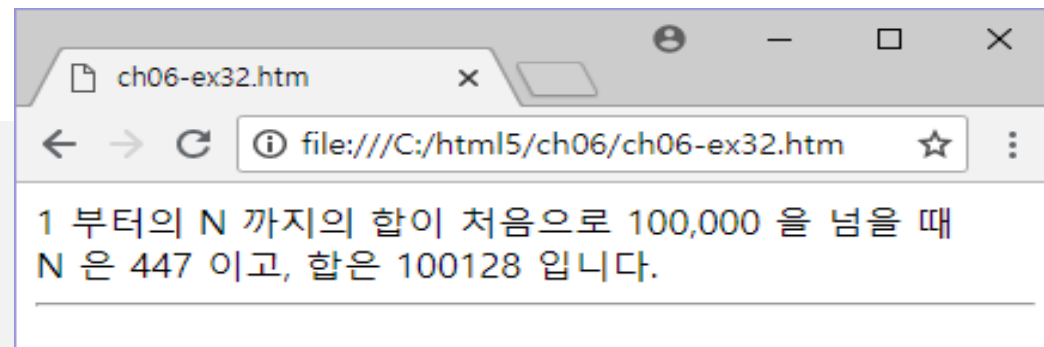
n	n 제곱	n 세제곱	n 네제곱
1	1	1	1
2	4	8	16
3	9	27	81
4	16	64	256
5	25	125	625

(3) break, continue 문

- break 문
 - 현재 실행중인 문장을 종료하고 다음 문장으로 이동함
 - 선택문과 반복문에서 사용함

[break문] break; 또는 break 라벨; //지정된라벨위치로이동함

```
<script>
sum = 0;
for (i=1; i <100000; i=i+1) {
    sum = sum + i;
    if (sum>100000) break;
}
document.write("1 부터의 N 까지의 합이 처음으로 100,000 을 넘을 때 <br> ");
document.write("N 은 " + i + " 이고, 합은 " + sum + " 입니다. <hr>");
</script>
```



● continue 문

[continue문] continue;

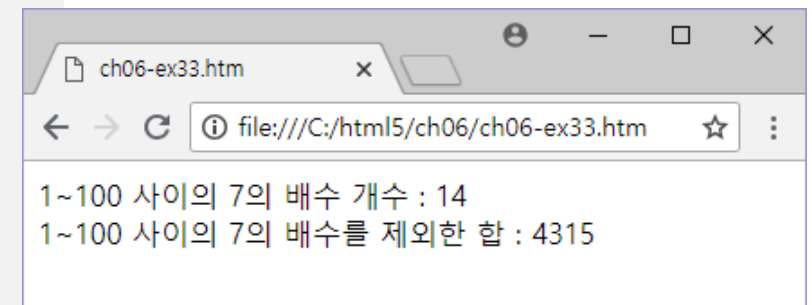
- 반복루프에서 나머지 문장들의 실행은 생략하고, 반복루프 다음의 실행 위치로 이동함
- 반복문에서 사용함
- continue 문을 for, while, do~while 문에서 사용하면, 다음 실행 위치가 각각 다름

```
for ( i=1; i<= 100; i++ ) {  
    ...  
    if (조건식) continue;  
    ...  
}
```

```
while ( i<=100 ) {  
    ...  
    if (조건식) continue;  
    ...  
}
```

```
do {  
    ...  
    if ( 조건식 ) continue;  
    ...  
} while ( i<=100 );
```

```
<script>  
sum=0; count=0;  
for (i=1; i<=100; i=i+1) {  
    if (i%7==0) { count++;  
        continue;  
    }  
    sum = sum  + i ;  
}  
document.write("1~100 사이의 7의 배수 개수 : " + count + "<br>");  
document.write("1~100 사이의 7의 배수를 제외한 합 : " + sum);  
</script>
```



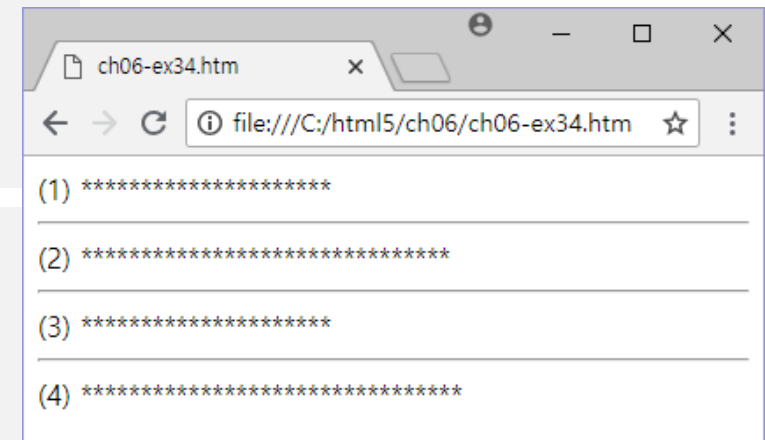
[예 6.34] for, while, do~while 문의 비교

```
str="";  
for (i=1; i++ <=50; i=i+1) {  
    str=str+ '*';  
    if (i%5 !=0) continue;  
    i=i+1;  
}  
document.write("(1) " + str);
```

```
str=""; i=1;  
do {  
    str=str+ '*';  
    if (i%5 !=0) continue;  
    i=i+1;  
    i=i+1;  
} while (i++ <=50);  
document.write("(4) " + str);
```

```
str=""; i=1;  
while (i++ <=50) {  
    str=str+ '*';  
    if (i%5 != 0) continue;  
    i=i+1; i=i+1;  
}  
document.write("(2) " + str);
```

```
str=""; i=1;  
do {  
    i=i+1;  
    str=str+ '*';  
    if (i%5 !=0) continue;  
    i=i+1;  
} while (i++ <=50);  
document.write("(3) " + str);
```



6.2.6 배열

(1) 배열의 정의와 사용

- 배열(array)
 - ✓ 서로 관련된 여러 변수 메모리들을 모아놓은 것
 - ✓ 배열을 이용하면 한꺼번에 많은 변수 메모리들을 연속적으로 확보해서 사용할 수 있음
 - ✓ 배열은 객체임
- 배열 정의 방법

// 리터럴들을 나열한 배열 정의

❶ var a1 = [10, 20, 30, 40, 50]; // 크기 5 인 배열 생성

❷ var a2 = []; // 빈 배열 생성

❸ var a3 = ["자료구조", 84, "웹프로그래밍", 92, "알고리즘", 76]; // 크기 6 인 배열 생성

// 객체생성자함수Array()를 이용한 배열정의

❹ var b1 = new Array(10,20,30,40,50); // ❶과 같음

❺ var b2 = new Aarray(); // ❷와 같음

❻ var b3 = new Array("자료구조", 84, "웹프로그래밍", 92, "알고리즘", 76); // ❸과 같음

❼ var b4 = new Aarray(5); // 크기가 5 인 배열 생성함

[그림 6.11] 예 ①~⑦의 배열들의 메모리 상태



● 배열 참조

(1) 배열원소 참조 : 배열명[인덱스]

- ✓ 0부터 1, 2, 3 등의 정수인덱스를 이용해서 개별적으로 접근 가능함 수 있음
- ✓ 배열원소값들을 변경할 수 있음

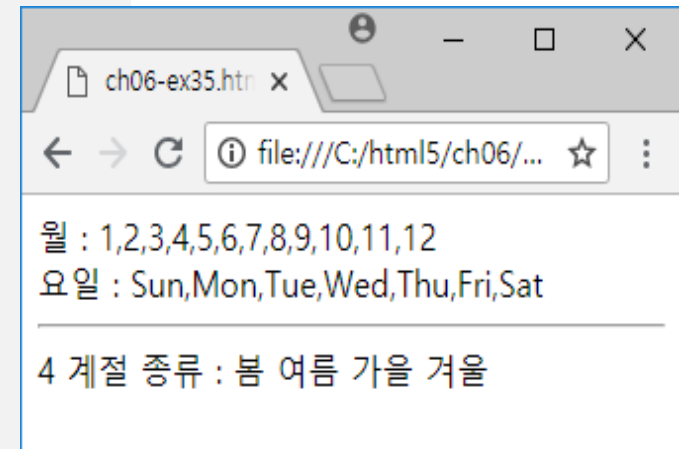
(2) 배열 전체 참조 : 배열명

- ✓ 모든 배열원소 값들을 순차적으로 연결한 문자열을 참조함
- ✓ 각 배열원소 값들을 변경할 수는 없음

(3) 배열명.length : 배열 원소 개수

```
<script>
var months = [1,2,3,4,5,6,7,8,9,10,11,12];
var days = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"];
var season = ["봄", "여름", "가을", "겨울"];

document.write("월 : " + months + "<br>"); // 배열명 참조
document.write("요일 : " + days + "<br>"); // 배열명 참조
document.write("<hr>" + season.length + " 계절 종류 : ");
for (i=0; i<season.length; i++) {
    document.write(season[i]+ ' '); // 배열원소 참조
}
</script>
```

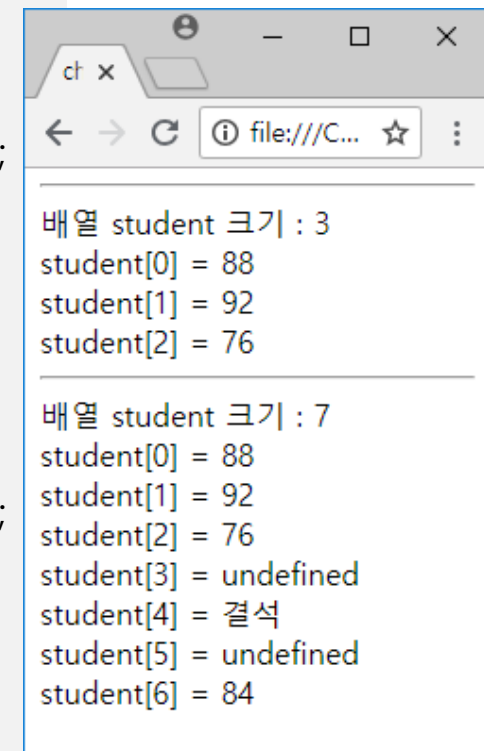


- 배열은 미리 정의하지 않고 사용할 수 있고, 배열 크기도 동적으로 변경할 수 있음
- 현재 배열크기보다 더 큰 정수 인덱스로 배열원소를 참조하면, 배열 크기는 자동으로 최대 인덱스 크기로 확장되고, 확장된 배열의 중간 인덱스의 배열원소들은 undefined 형이 됨
- 배열원소들의 자료형은 같지 않아도 됨

```
<script>
  var student=[88, 92, 76];

  document.write("<hr> 배열 student 크기 : " + student.length + "<br>");
  for (i=0; i<student.length; i++) {
    document.write("student[" + i + "] = ");
    document.write(student[i] + "<br>");
  }

  student[6]= 84; // 동적 배열크기 확장
  student[4]= "결석";
  document.write("<hr> 배열 student 크기 : " + student.length + "<br>");
  for (i=0; i<student.length; i++) {
    document.write("student[" + i + "] = ");
    document.write(student[i] + "<br>");
  }
</script>
```



- 자바스크립트의 2차원 배열

- "배열들의 배열" 즉, 각 배열원소들이 배열인 배열로 정의함

//2차원배열정의예

⑧ var c1 = [[10,15,20,30], [40,50,60], [70,75,80,90]]; // 크기 3x4 인 2차원 배열

⑨ var c2 = new Array(Array("자료구조", 84), Array("웹프로그래밍", 92),
Array("알고리즘", 76), Array("웹프로젝트", 88)); // 크기 4x2 인 2차원 배열

2차원 배열 c1

	[0]	[1]	[2]	[3]
[0]	10	15	20	30
[1]	40	50	60	-
[2]	70	75	80	90

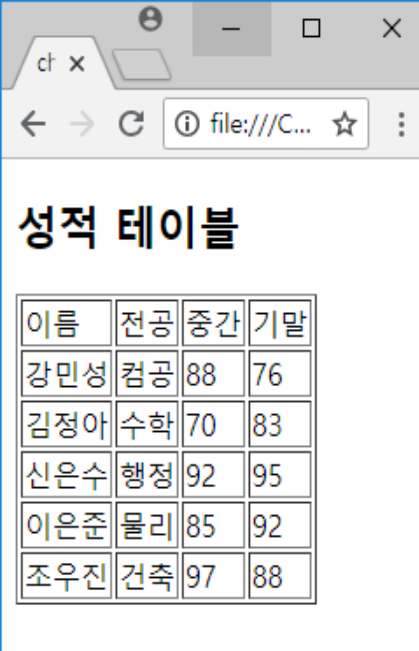
2차원 배열 c2

	[0]	[1]
[0]	"자료구조"	84
[1]	"웹 프로그래밍 "	92
[2]	"알고리즘"	76
[3]	"웹 프로젝트 "	88

```

<script>
  var student = new Array( Array('강민성', '컴공', 88, 76),
    Array('김정아', '수학', 70, 83), Array('신은수', '행정', 92, 95),
    Array('이은준', '물리', 85, 92), Array('조우진', '건축', 97, 88)
  );
  document.write("<h2> 성적 테이블 </h2>");
  document.write("<table border=1>");
  document.write("<tr><td>이름</td><td>전공</td>"
    + "<td>중간</td><td>기말</td></tr>");
  for (i=0; i<5; i++) {
    document.write("<tr>");
    for (j=0; j<4; j++) {
      document.write("<td>" + student[i][j]+ "</td>");
    }
    document.write("</tr>");
  }
</script>

```



성적 테이블

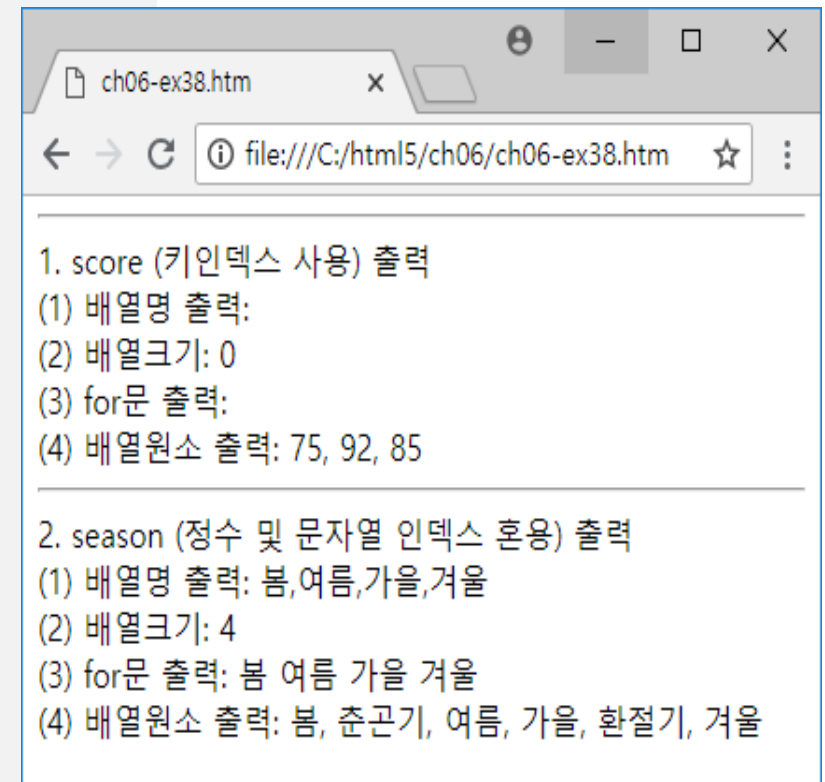
이름	전공	중간	기말
강민성	컴공	88	76
김정아	수학	70	83
신은수	행정	92	95
이은준	물리	85	92
조우진	건축	97	88

- **문자열 인덱스**(named index 또는 key)를 이용해서 배열원소들의 접근할 수 있음
- 문자열 인덱스가 저장된 string 변수를 인덱스로 사용하는 것도 가능함

```
<script>
var score=new Array(), season=new Array("봄","여름","가을","겨울");
var subject="java";
score["web"] = 75; score[subject]=92; score['project']=85;

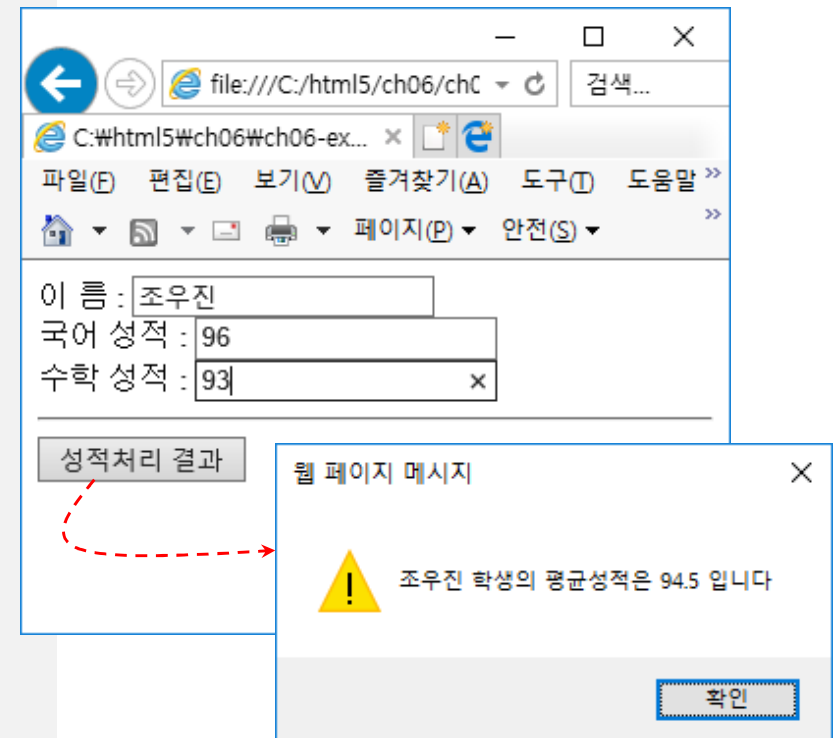
// 배열 score 출력
document.write("<hr>1. score (키인덱스 사용) 출력 <br>");
document.write("(1) 배열명 출력: " + score + "<br>");
document.write("(2) 배열크기: " + score.length + "<br>");
document.write("(3) for문 출력: ");
for (i=0 ; i<score.length; i++) document.write(score[i] + " ");
document.write("<br>(4) 배열원소 출력: ");
document.write( score['web'] + ", " + score[subject] );
document.write(", " + score['project'] + "<hr>");

season["mar"]="춘곤기"; season["sep"]="환절기";
document.write("2. season (정수/키인덱스 혼용) 출력<br>");
document.write("(1) 배열명 출력: " + season + "<br>");
document.write("(2) 배열크기: " + season.length + "<br>");
document.write("(3) for문 출력: ");
for (i=0 ; i<season.length; i++) document.write(season[i]+" ");
document.write("<br>(4) 배열원소 출력: ");
document.write(season[0] + ", " + season['mar'] + ", " + season[1] + ", ");
document.write(season[2] + ", " + season['sep'] + ", " + season[3]);
</script>
```



[예 6.39] 문자열 인덱스를 이용한 배열원소 참조

```
<form name="form1">
  이름 : <input type="text" name="stuName"/> <br>
  국어 성적 : <input type="text" name="kor"/> <br>
  수학 성적 : <input type="text" name="math"/> <br>
  <button onclick="compute();" > 성적처리 결과 </button>
</form>
<script>
function compute() {
  var total;
  total = parseInt(form1['kor'].value) +
          parseInt(form1['math'].value);
  avg = total / 2 ;
  alert(form1['stuName'].value
        + " 학생의 평균성적은 " + avg + " 입니다");
}
</script>
```



(2) for~in 문

- 배열의 모든 배열원소들이나 객체의 모든 속성들에 대해 반복적으로 동일한 처리를 할 때 사용함
- 배열의 원소 개수, 객체의 모든 속성명들을 모르더라도 빠짐없이 처리할 수 있음
- 즉, 배열이나 객체의 모든 자료값들을 빠짐없이 참조할 수 있는 방법을 제공함

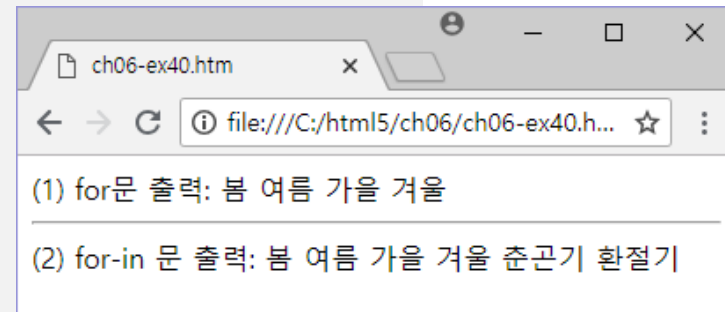
```
// obj : 배열 또는 객체변수
// index : 인덱스 또는 객체속성을 나타내는 변수
for ( var index in obj ) {
    obj[index]를 처리하는 코드;
}
```

```
var subject = ["algorithm", "DataStructure", "ComputerNetwork", "DB"];
❶ for (i=0; i<=subject.length; i++) document.write(subject[i]+"<br>");
❷ for ( var i in subject) document.write(subject[i]+"<br>");
❸ for ( var i in subject) document.write(subject.i+"<br>");
```

[예 6.40] for, for~in 문의 비교(1)

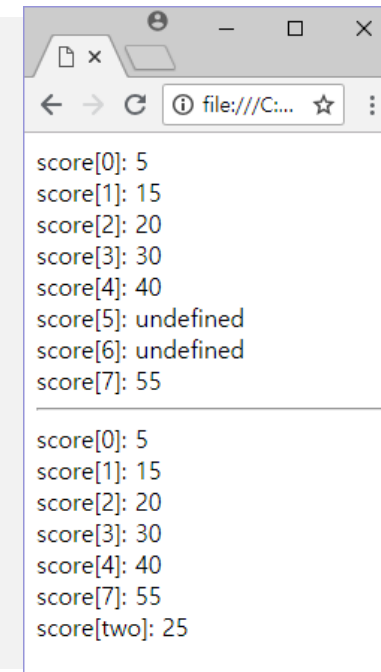
```
<script>
  var season=new Array("봄","여름","가을","겨울");

  season["mar"]="춘곤기"; season["sep"]="환절기";
  document.write("(1) for문 출력: ");
  for (i=0 ; i<season.length; i++)
    document.write(season[i]+" ");
  document.write("<hr>");
  document.write("(2) for-in 문 출력: ");
  for ( var key in season)
    document.write(season[key]+" ");
</script>
```



[예 6.41] for, for~in 문의 비교(2)

```
<script>
  var score = [0, 10, 20, 30, 40];
  score[0] = 5;
  score["1"] = 15;
  score["two"] = 25;
  score[7] = 55;
  for (i=0; i<score.length; i++) {
    document.write("score[" + i + "]: " + score[i] + "<br>");
  }
  document.write("<hr>");
  for (var key in score) {
    document.write("score[" + key + "]: " + score[key] + "<br>");
  }
</script>
```

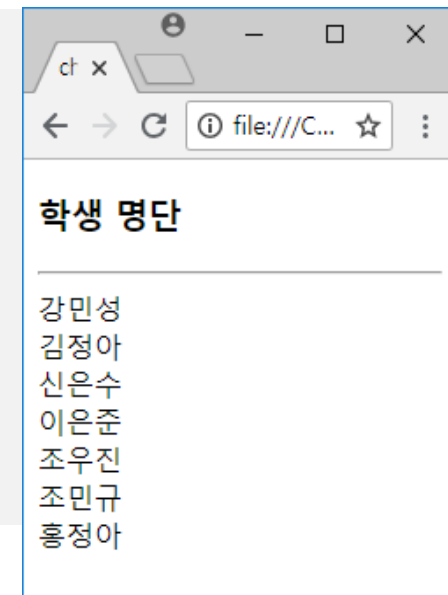


[예 6.42] for~in 문의 사용

```
<script>
  var students = ['강민성', '김정아', '신은수', '이은준', '조우진'];

  students['전입생1'] = '조민규';
  students['전입생2'] = '홍정아';

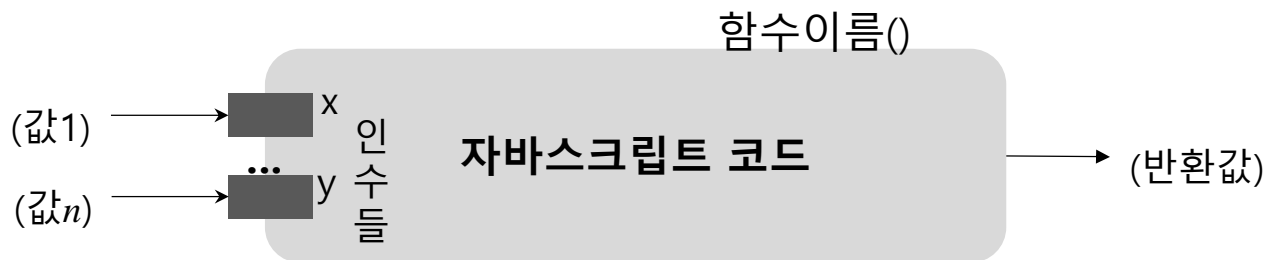
  document.write("<h3> 학생 명단 </h3> <hr>");
  for ( var i in students ) {
    document.write(students[i] + "<br>");
  }
</script>
```



6.3 자바스크립트 함수

6.3.1 함수 정의와 사용

- 함수(function)
 - 특정기능이나 작업을 수행하는 프로그램 단위(program unit)
 - 인수들을 통해 외부로부터 자료값을 전달받고, 종료될 때 한 값을 반환함
 - 함수는 사용 전에 미리 정의해야 하고, 프로그램의 다른 위치에서 이 함수를 호출해야 실행됨



```
function 함수이름(인수1,...,인수n){  
    지역변수 선언;  
    함수의 명령코드들;  
    return 반환값; //함수값 반환 및 종료  
}
```

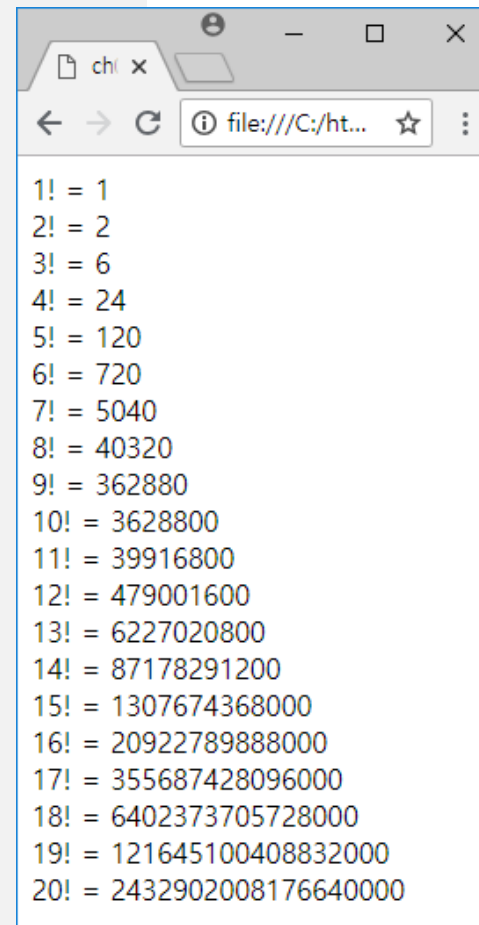
```
function sum(a,b){  
    var c;  
    c=a+b;  
    return(c);  
}
```

[예 6.43] 함수 정의 및 호출

```
<script>
  function f1(n) {
    var result = 1;
    var num = Number(n);
    for (i=1; i<=num; i++ ) result *= i;
    return result;
  }

  function f2(n) {
    var result=1;
    num = Number(n);
    for (i=1; i<=num; i++ ) result *= i;
    document.write(n + "! = " + result + "<br>");
  }

  for (j=1; j<=20; j++) {
    if ( j/2*2 == j)
      document.write(j+"! = "+ f1(j) + "<br>"); // ①
    else f2(j); // ②
  }
</script>
```



1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
20! = 2432902008176640000

(1) 전역 변수와 지역 변수

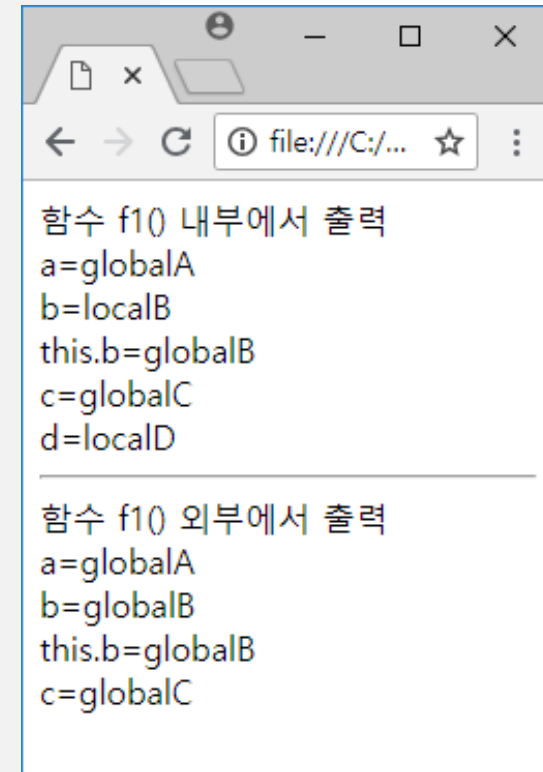
- 전역 변수(global variable)
 - ✓ 함수들 밖에서 정의한 변수들
 - ✓ 해당 웹 페이지를 적재하는 동안 계속 유지, 웹 문서내의 모든 자바스크립트 코드에서 참조할 수 있음
- 지역 변수(local variable)
 - ✓ 특정 함수의 내부에서 정의된 변수들
 - ✓ 해당 함수가 실행되는 동안만 유지, 해당 함수 내의 코드들에서만 참조할 수 있음
- 함수 내부에서 정의되지 않고 사용되는 변수는 지역변수임(전역변수는 함수 밖에서 정의해야 함)
- 함수 내부에서 전역변수와 지역변수명이 같으면 지역변수로 취급함
- 이 경우, 전역변수로 취급하려면 "**this.전역변수명**" 으로 사용해야 함

[예 6.44] 지역 변수, 전역 변수 구분 예

```
<script>
  var a="globalA", b="globalB", c="globalC";
  function f1() {
    var b="localB";
    var d="localD";

    document.write("함수 f1() 내부에서 출력<br>");
    document.write("a=" + a + "<br>");
    document.write("b=" + b + "<br>");
    document.write("this.b=" + this.b + "<br>");
    document.write("c=" + c + "<br>");
    document.write("d=" + d + "<br><hr>");
  }

  f1();
  document.write("함수 f1() 외부에서 출력<br>");
  document.write("a=" + a + "<br>");
  document.write("b=" + b + "<br>");
  document.write("this.b=" + this.b + "<br>");
  document.write("c=" + c + "<br>");
  document.write("d=" + d + "<br><hr>");
</script>
```



(2) 자바스크립트 내장 함수

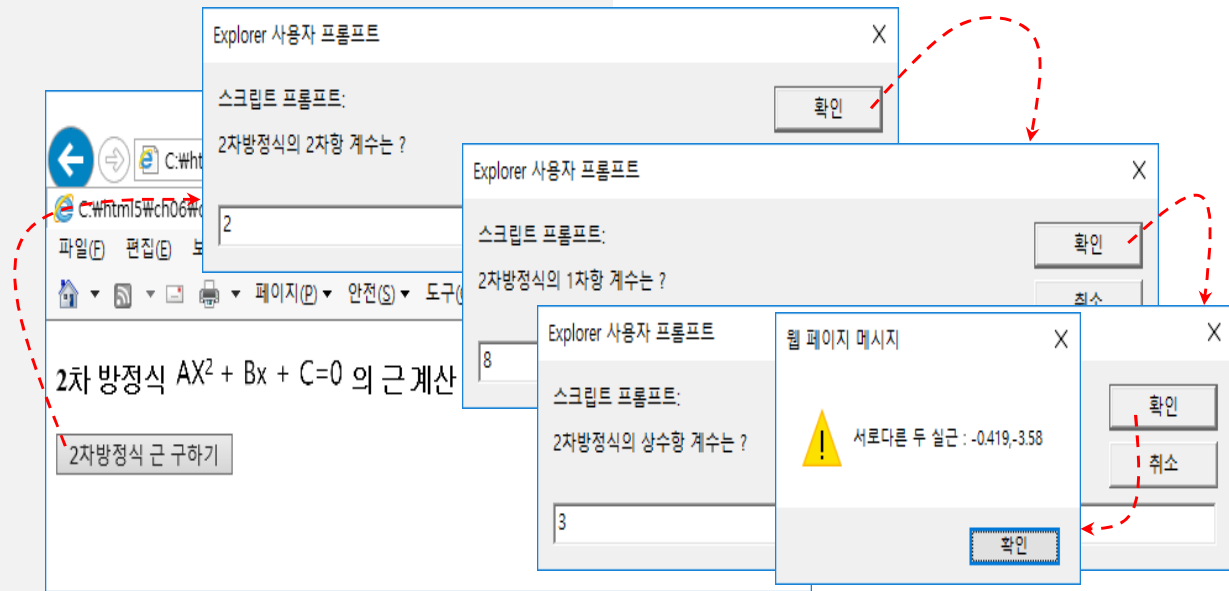
- 자바스크립트에서 미리 정의해 둔 함수들, 별도의 함수 정의없이 사용할 수 있음
- 대부분 내장 객체들의 메서드(method) 형태로 제공됨

속성 및 메서드	의미 또는 기능
<code>parseInt(s)</code>	문자열 <code>s</code> 를 10진 정수로 변환해서 반환함
<code>parseFloat(s)</code>	문자열 <code>s</code> 를 10진 실수로 변환해서 반환함
<code>isFinite(x)</code>	수치 <code>x</code> 의 유한수 여부를 반환함(true, false)
<code>isNaN(x)</code>	<code>x</code> 의 비수치(Not-a-Number) 여부를 반환함(true, false)
<code>eval(exp)</code>	문자열 수식 <code>exp</code> 의 계산결과를 반환함
<code>toFixed(n)</code>	수치를 소수점 이하 <code>n</code> 자리까지 나타낸 문자열을 반환함
<code>toPrecision(n)</code>	수치를 크기 <code>n</code> 의 문자열로 변환해서 반환함
<code>valueOf()</code>	수식(수치)을 계산한 값(수치) 반환함
<code>MATH.sqrt(x)</code>	<code>x</code> 의 제곱근을 계산해서 반환함

[예 6.45] 자바스크립트 내장 함수들을 이용한 2차방정식 근 구하기

<h3> 2차 방정식 의 근 계산</h3>
<button onclick="slove();" >2차방정식 근 구하기</button>

```
<script>
function slove(){
  var a1= prompt("2차방정식의 2차항 계수는 ? ");
  var b1= prompt("2차방정식의 1차항 계수는 ? ");
  var c1= prompt("2차방정식의 상수항 계수는 ? ");
  a=parseInt(a1); b=parseInt(b1); c=parseInt(c1);
  d = b*b - 4*a*c;
  if (a==0 ) { alert("2차방정식이 아님!"); }
  else if ( d < 0 ) { alert("근 없음!"); }
  else if ( d==0 ) {
    root1 = (-1*b + Math.sqrt(d)) /(2*a);
    alert("중근 : " + root1.toPresion(3));
  }
  else {
    root1 = (-1*b + Math.sqrt(d)) /(2*a) ;
    root2 = (-1*b - Math.sqrt(d)) /(2*a) ;
    alert("서로다른 두 실근 : " + root1.toPrecision(3) + "," + root2.toPrecision(3));
  }
}
</script>
```



6.3.2 자바스크립트 함수의 특성

(1) 함수 인수들에 대한 느슨한 검사

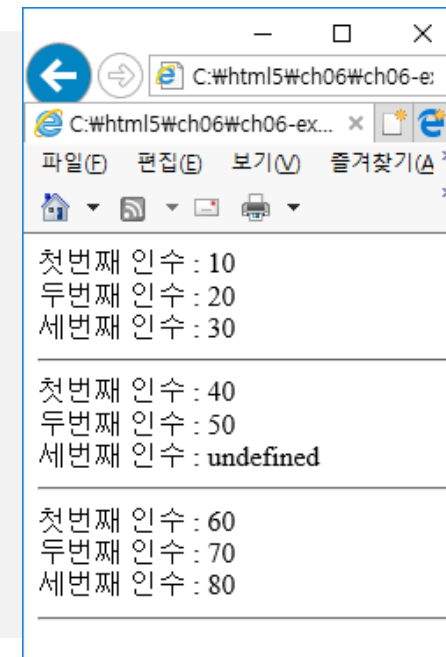
- 함수를 호출할 때 인수들의 자료형과 개수 등을 엄격하게 확인하지 않음
- 인수값 개수가 부족하면 순차적으로 대응하고 남겨진 인수들은 undefined형으로 처리함
- 초과되는 인수값들은 무시함

[예 6.46]

```
<script>
function f1(x, y, z) {
    document.write("첫 번째 인수 : " + x + "<br>");
    document.write("두 번째 인수 : " + y + "<br>");
    document.write("세 번째 인수 : " + z + "<br>");
    document.write("<hr>");
}

f1(10, 20, 30); // ①
f1(40, 50);    // ②
f1(60, 70, 80, 90); // ③

</script>
```



(2) 함수 선언 위치와 사용 위치의 독립성

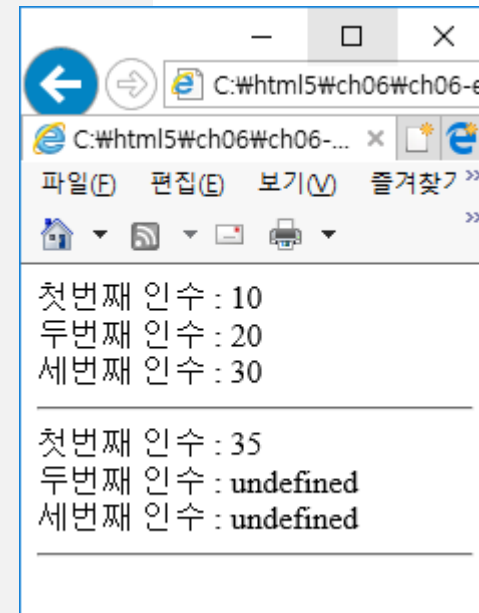
- 자바스크립트에서는 함수와 변수들을 먼저 사용하고 뒤에 정의를 하는 것도 가능함
- 변수 및 함수 호이스팅(function hoisting) 기능 제공
 - ✓ 초기화된 변수정의를 제외한 모든 전역 변수들과 함수들의 정의를 HTML 문서의 앞부분으로 자동 이동시킨 후 실행을 시작함
- 다만, 특별한 경우가 아니면 함수와 변수들은 먼저 정의하고 사용하는 것이 바람직함

[예 6.47]

```
<script>
  x=10; y=20; z=30; a=35;
  f1(10, 20, 30);
  f1(a, b);

  function f1(x, y, z) {
    document.write("첫번째 인수 : " + x + "<br>");
    document.write("두번째 인수 : " + y + "<br>");
    document.write("세번째 인수 : " + z + "<br>");
    document.write("<hr>");
  }

  var x, y, z;
  var a, b=45;
</script>
```



(3) 익명 함수와 콜백 함수

- 자바스크립트는 함수를 호출가능한 객체 자료로 취급함
- 객체 자료를 나타낼 수 있는 곳에는 함수들도 나타낼 수 있음
 - 예) 함수를 객체 변수에 저장해서 사용(function형 변수)
 - 예) 특정 함수의 인수 또는 반환값으로 사용할수 있음

❶ `var f1 = function(a,b){ return a*b }; var x = f1(4,3);`

❷ `window.onload = function(){ alert('callmebaby'); }; // 콜백함수`

- 콜백 함수(callback function) : 어떤 함수의 인수로만 사용되는 함수
- 클로저(closure) : 반환 값으로만 사용되는 함수들
- 익명 함수(anonymous function) : 함수명이 없는 작성되는 함수 코드들
 - ✓ 일반적으로 익명함수는 함수의 인수값으로 사용함
 - ✓ 이벤트처리함수, 애니메이션 기능 구현 등에서 많이 사용함

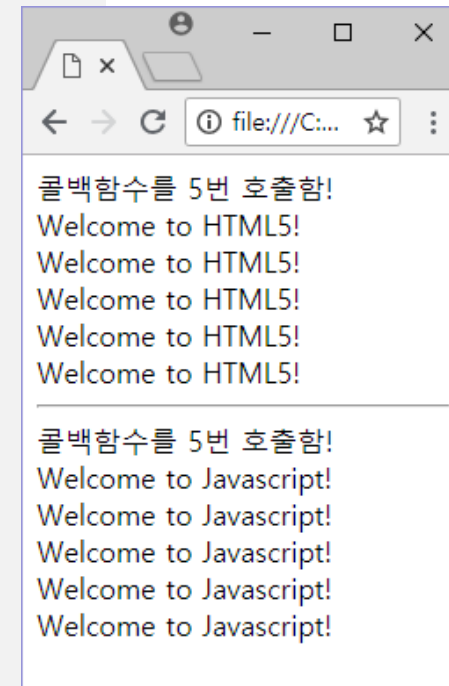
[예 6.48] 콜백 함수의 사용 예

```
<script>
  function f1(f2, n) {
    document.write('콜백함수를 '+n+'번 호출함!' + '<br>');
    for ( i=0; i<n; i++) f2(); ;
  }

  var f3 = function () {
    document.write('Welcome to HTML5 ! <br>');
  }

  f1(f3, 5); // ①

  document.write('<hr>');
  f1( function() { // ②
    document.write('Welcome to Javascript ! <br>');
  }, 5);
</script>
```



- ✓ 함수호출 ① : 외부에서 정의된 함수 f3을 인수로 전달함
- ✓ 함수호출 ② : 직접 함수 코드(익명 함수)를 작성해서 인수로 전달함

Next 7장



자바스크립트 객체와
브라우저 객체 모델