

# ◆ Java Script란?

◆ 단순히 보여주거나 이동하는 명령의 집합인 HTML을 프로그래밍 수준에서 처리할 수 있게 해주는 클라이언트용 프로그래밍 언어이다.

## ◆ 역사

- 넷스케이프(Netscape)의 LiveScript를 독립적으로 개발
  - 서버를 거치지 않고도 클라이언트쪽에서 독립적으로 실행되는 프로그램
- 넷스케이프(Netscape)사와 썬 마이크로 시스템(Sun Microsystems)가 공동으로 개발

## ◆ 장점

- 작업이 빠르고 생산성이 좋음 – HTML코드 안에 삽입, 컴파일이 없음
- 운영체제에 대한 제한을 받지 않음
- 학습이 쉬움

## ◆ 단점

- 소스 코드 노출
- 한정된 객체와 메소드

# ◆ Java Script란?

## ◆ JavaScript 와 Java 의 차이점

종 류	JavaScript	Java
실행 방법	브라우저에서 해석됨	서버에서 컴파일된 byte code를 불러 클라이언트(브라우저)에서 해석됨
성격	OOP를 기반으로 되어 있지만 클래스가 없고 상속할 수 없음. 객체 기반 언어	완벽한 OOP로 클래스와 상속이 지원됨
형태	HTML 파일 내에서 같이 기술 (HTML과 결합되어 사용)	HTML파일에 작은 공간을 얻어 수행 (HTML과 별도로 존재할 수 있음)
변수	변수형을 미리 정할 필요가 없음	미리 변수형을 정해야 함
보안성	소스가 노출되어 보안성이 없음	컴파일된 실행파일이므로 보안성이 있음

# ◆ Java Script의 특징

- ◆ HTML문서 자체에 포함 컴파일 과정을 거치지 않고 브라우저에서 실행
- ◆ 웹 페이지가 서버에 가하는 부담을 줄여 줌.
- ◆ 넷스케이프가 실행 가능한 모든 플랫폼(Windows 3.1 과 Mac)에서 실행
  - Netscape2.00이상, Internet Explorer 3.00이상에서 지원됨
- ◆ 하나의 HTML문서에 여러 개의 < script >....< /script >가 들어갈 수 있음.
- ◆ 자바스크립트는 대문자와 소문자를 구별

# ◆ 주석

```
// 한줄이 주석처리입니다.
```

```
/* 주석 처리 시작  
여기는 여러줄 주석입니다^^  
주석 처리 끝 */
```

# ◆ 기본구조

- ◆ `< script >....< /script >` 태그 내에 표현
- ◆ HTML 태그의 이벤트 핸들러를 자바스크립트 코드로 표현
- ◆ 자바스크립트 코드를 URL로 사용
- ◆ 자바스크립트 파일의 내장

# ◆ 기본구조

◆ < script >....< /script> 태그 내에 내용

```
<SCRIPT language="스크립트 종류" src="파일경로">  
<!--
```

스크립트 소스

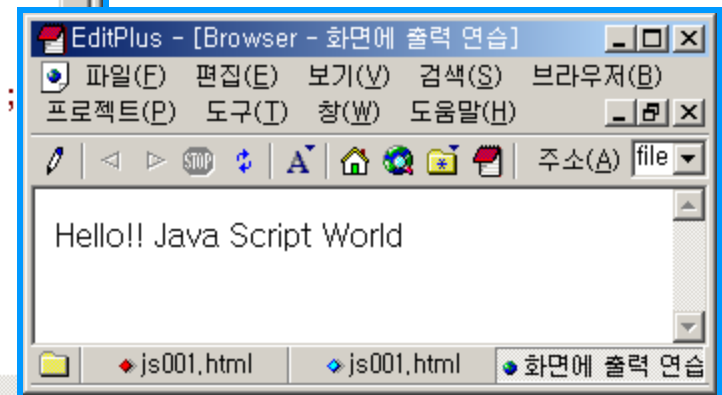
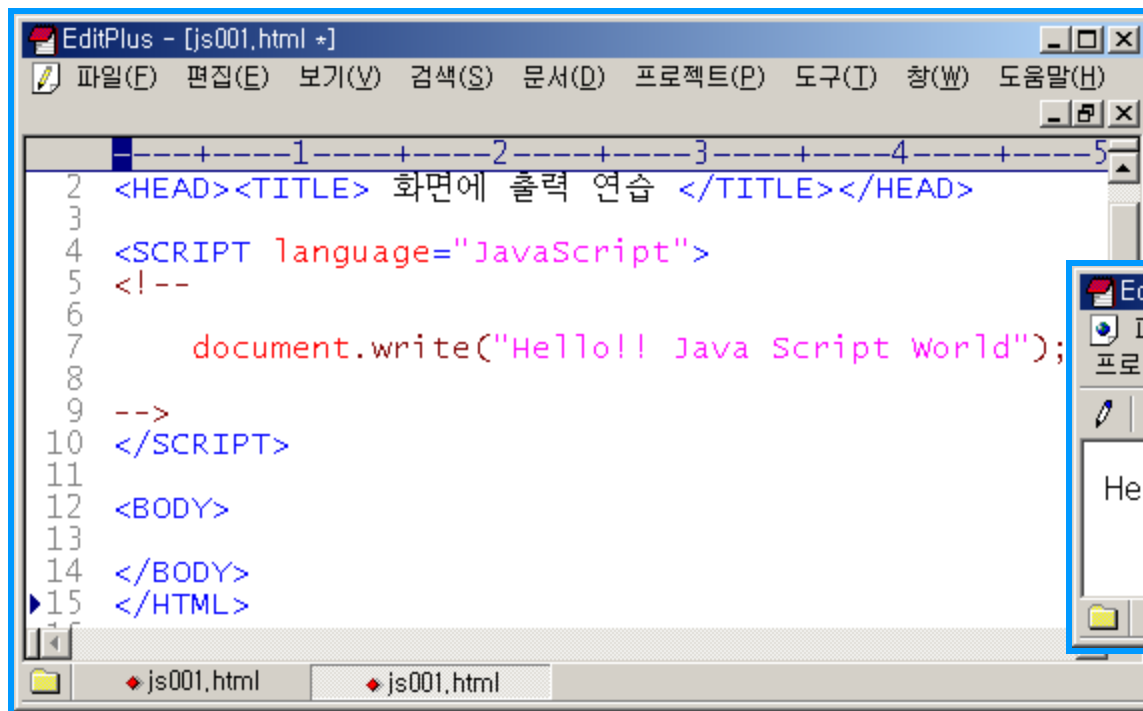
```
//-->  
</SCRIPT>
```

속성	내 용
language	스크립트 언어의 종류를 설정함 자바스크립트는 JavaScript 비주얼베이직 스크립트는 VBScript
src	스크립트가 있는 경로를 포함한 파일명을 설정함

# ◆ 기본예제

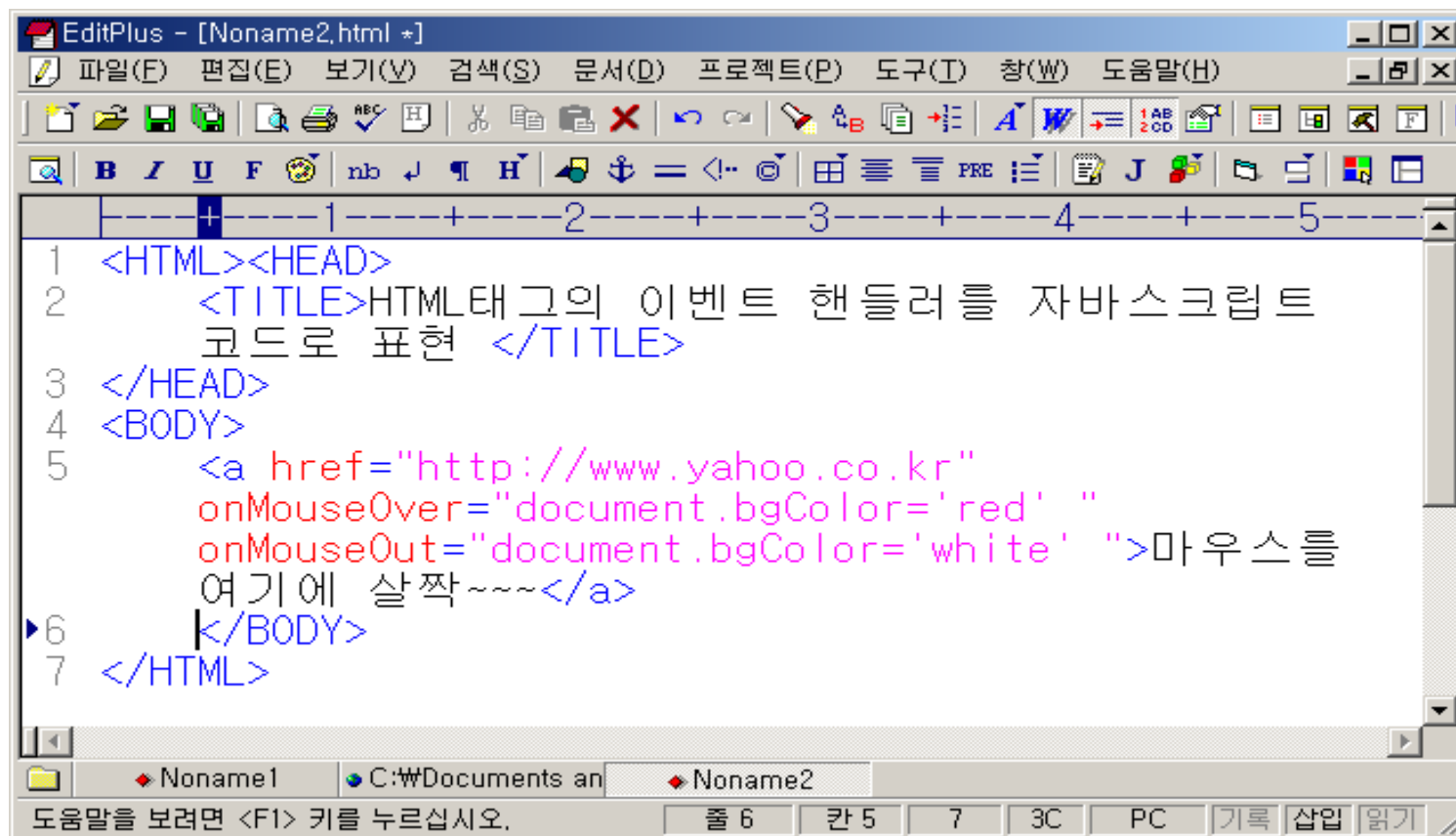
## ◆ “Hello World!!”를 화면에 출력하는 문장

```
<SCRIPT language="JavaScript" >
<!--
    document.write("Hello World!!");
-->
</SCRIPT>
```



# ◆ 기본구조

- ◆ HTML 태그의 이벤트 핸들러를 자바스크립트 코드로 표현
  - <태그이름 ... **onEventName**= "자바스크립트 코드" ....>



The screenshot shows the EditPlus text editor with a file named [Noname2.html]. The code is as follows:

```
1 <HTML><HEAD>
2     <TITLE>HTML태그의 이벤트 핸들러를 자바스크립트
   코드로 표현 </TITLE>
3 </HEAD>
4 <BODY>
5     <a href="http://www.yahoo.co.kr"
   onMouseOver="document.bgColor='red' "
   onMouseOut="document.bgColor='white' ">마우스를
   여기에 살짝~~~</a>
6 </BODY>
7 </HTML>
```

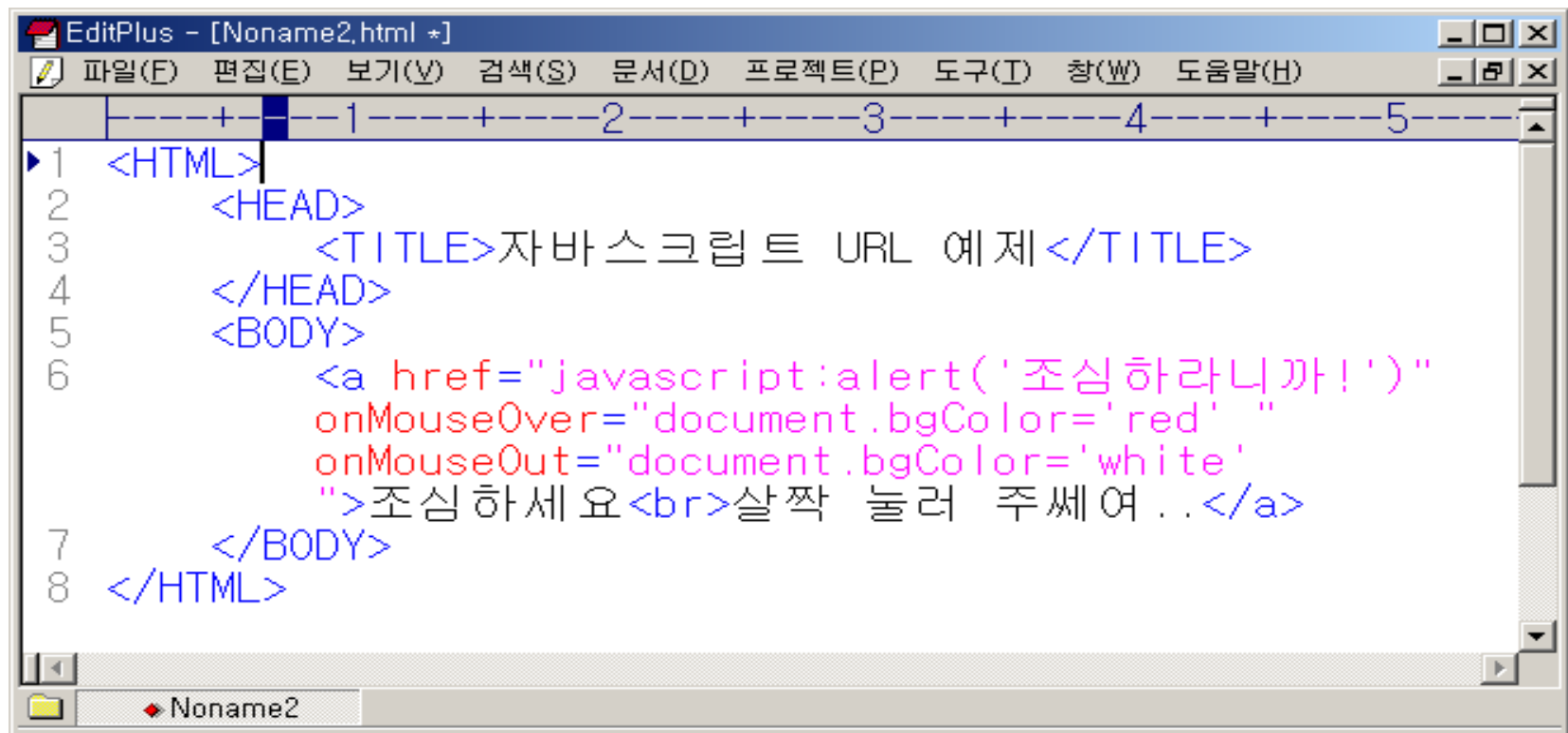
The status bar at the bottom indicates the current position is 줄 6 (Line 6), 칸 5 (Column 5). The bottom-most status bar shows "도움말을 보려면 <F1> 키를 누르십시오." (Press <F1> key to see help).



# ◆ 기본구조

## ◆ 자바스크립트 코드를 URL로 사용

- `< a href="javascript:자바스크립트코드" ....> ..... </a>`

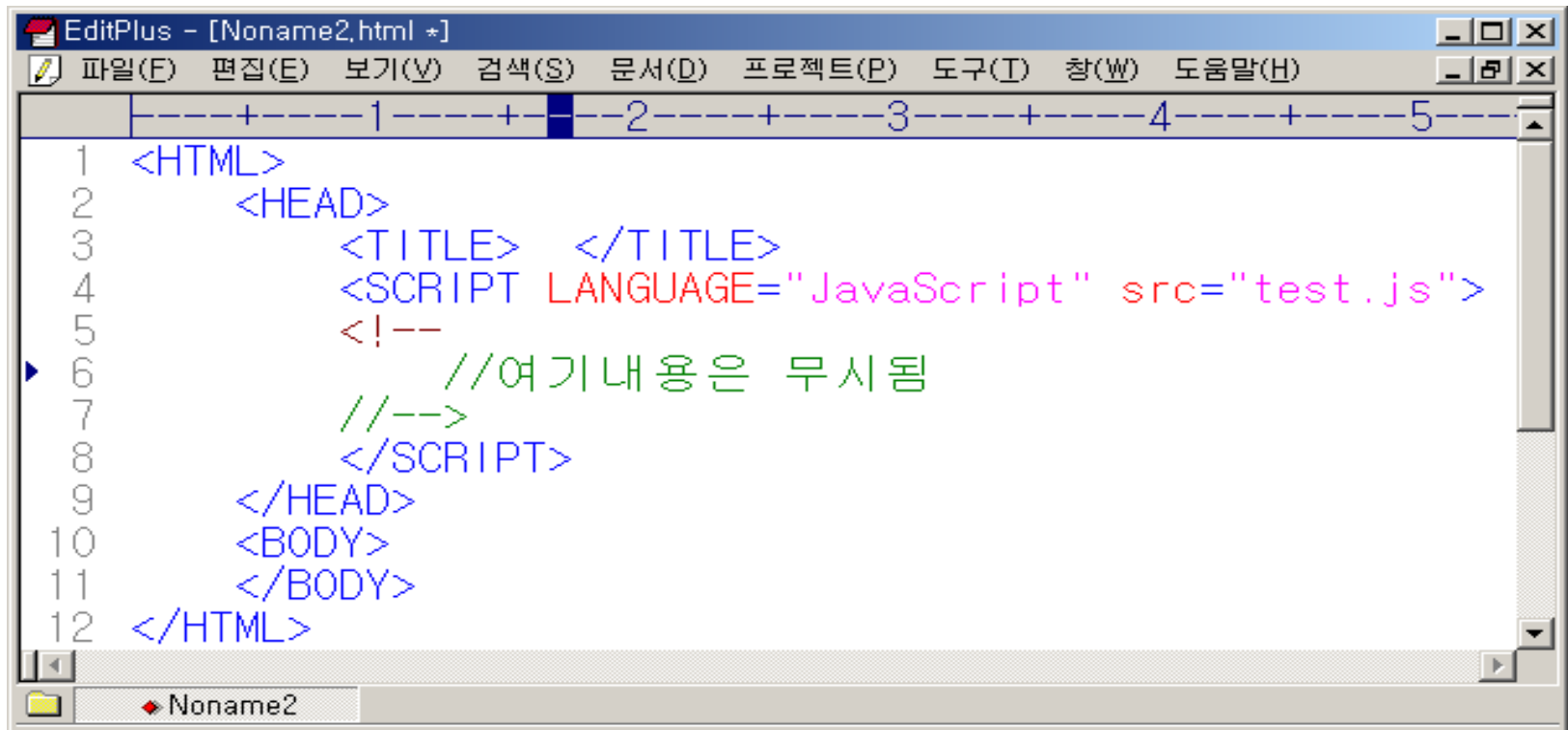


```
1 <HTML>
2   <HEAD>
3     <TITLE>자바스크립트 URL 예제</TITLE>
4   </HEAD>
5   <BODY>
6     <a href="javascript:alert('조심하라니까!')"
7       onMouseOver="document.bgColor='red' "
8       onMouseOut="document.bgColor='white'
       ">조심하세요<br>살짝 눌러 주세요..</a>
9   </BODY>
10 </HTML>
```

# ◆ 기본구조

## ◆ 자바스크립트 파일 내장

- `< script src="test.js">여기 내용은 무시됨 </script>`



The screenshot shows the EditPlus text editor with a file named [Noname2.html \*]. The menu bar includes File (F), Edit (E), View (V), Search (S), Document (D), Project (P), Tools (T), Window (W), and Help (H). The editor content is as follows:

```
1 <HTML>
2     <HEAD>
3         <TITLE> </TITLE>
4         <SCRIPT LANGUAGE="JavaScript" src="test.js">
5             <!--
6                 //여기내용은 무시됨
7                 //-->
8         </SCRIPT>
9     </HEAD>
10    <BODY>
11    </BODY>
12 </HTML>
```

Sample

◆ 외부정의.html

# ◆ 변수(Variable)

- ◆ 임의 값을 저장하는 임시 기억장소로 데이터를 처리하기 위해서 사용하는 수를 말함
- ◆ 상수는 한번 선언하면 바꿀 수 없지만 변수는 프로그램 수행중에 언제든지 그 값을 바꿀 수 있음

var 변수명

- ◆ 변수명 규칙
  - 영문자, 숫자, 밑줄(\_)문자만 사용가능하고 특수기호는 사용못함
  - 미리 정해진 예약어는 변수명으로 사용할 수 없음
  - 첫번째 문자는 반드시 영문자여야 함
  - 한글 변수명은 사용할 수 없음
  - 대소문자구분
  - 전역변수: 함수의 밖에서 선언, 모든 함수에서 참조, 선택적으로 var 키워드 사용
  - 지역변수: 함수의 안에서 선언, 해당 함수에서만 참조, 반드시 var 키워드 사용 (var 키워드없으면 전역변수로 인식)

# ◆ 데이터 형(Data Type)

- ◆ 별도의 데이터 형이 없으며 선언하지 않음
- ◆ 숫자나 문자를 변수에 넣으면 자동으로 인식함
- ◆ 정수
  - 10진수, 8진수, 16진수로 표현가능
- ◆ 부동소수점
  - 소수점이 있는 수
- ◆ Boolean(논리형)
  - True, False 값을 가짐
- ◆ 문자열(스트링)
  - 큰따옴표(“)나 작은따옴표(‘)안에 넣어서 사용함
- ◆ null값
  - 값이 없음을 표시하는 특수 키워드, Null, NULL과는 다른 값임

## Sample

- ◆ 지역Exam.html
- ◆ 전역지역Exam.html

# ◆ 연산자

## ◆ 산술연산자

- 사칙 연산자와 나머지 연산자가 있음

속성	내 용
+	더하기 연산을 함
-	빼기 연산을 함
*	곱하기 연산을 함
/	나누기 연산을 함
%	나머지 연산을 함

Sample

◆ 산술연산자.html

# ◆ 연산자

## ◆ 연결연산자

- 문자열을 연결해주는 연산자

속 성	내 용
+	문자열을 연결해주는 연산자

Sample

◆연결연산자.html

# ◆ 연산자

## ◆ 비교연산자

- 두 피연산자를 조건에 따라 선택적으로 실행할 수 있도록 해줌
- 참이면 True, 거짓이면 False값을 반환함

속 성	내 용
==	A와 B가 같음
!=	A와 B가 같지 않음
<	A가 B보다 작음
<=	A가 B보다 작거나 같음
>	A가 B보다 큼
>=	A가 B보다 크거나 같음

Sample

◆비교연산자.html

# ◆ 연산자

- ◆ 대입연산자
  - 우변의 값을 좌변에 넣을 때 사용

속 성	내 용
=	왼쪽 변수에 오른쪽 값을 할당함
+=	더하기 연산 후에 할당
-=	빼기 연산 후에 할당
*=	곱하기 연산 후에 할당
/=	나누기 연산 후에 할당
%=	나머지 연산 후에 할당

Sample

◆대입연산자.html



# ◆ 연산자

## ◆ 논리연산자

- 참과 거짓을 나타내는 연산자

속 성	내 용
&&	A와 B를 모두 만족할 때 참
	A와 B 중에 하나만 만족하면 참
^	A와 B가 둘다 같은 값이면 거짓, 다르면 참

- 연산자 논리표

좌변	우변	&&		^
False	False	False	False	False(0)
False	True	False	True	True(1)
True	False	False	True	True(1)
True	True	True	True	False(0)

# ◆ 연산자

## ◆ 조건연산자

- 조건에 따라 두 값 중에 하나를 연산결과로 전달함

(조건) ? 참 : 거짓

Sample

◆논리연산자.html

# ◆ 연산자

## ◆ 증감연산자

- 1씩 증가, 감소하기 편리하게 하기 위해서 사용됨

속 성	내 용
++	1씩 증가함
--	1씩 감소함

- ++a 와 a++의 차이점
  - ++a : 1을 증가한 후에 일 처리함(선증가 후대입)
  - a++ : 일을 처리한 후에 1씩 증가함(선대입 후증가)

Sample

◆증감연산자.html



교사 : 장 희 정

# ◆ 함수(Function)

- ◆ 프로그램을 수행하기 위한 일련의 과정들을 하나의 단위로 묶어줄 수 있는 기능을 제공해 주는 방법
- ◆ 대소문자를 구별
- ◆ 내부함수와 외부함수가 있다.
- ◆ 내부함수(내장함수)
  - 미리 자바스크립트 내에 만들어져 있는 함수
- ◆ 외부 함수
  - 사용자가 자신이 원하는 함수를 만들어서 사용하는 것,
  - 키워드로 function이라는 것을 사용함.
  - 사용자 정의 함수라고도 함

# ◆ 내장함수

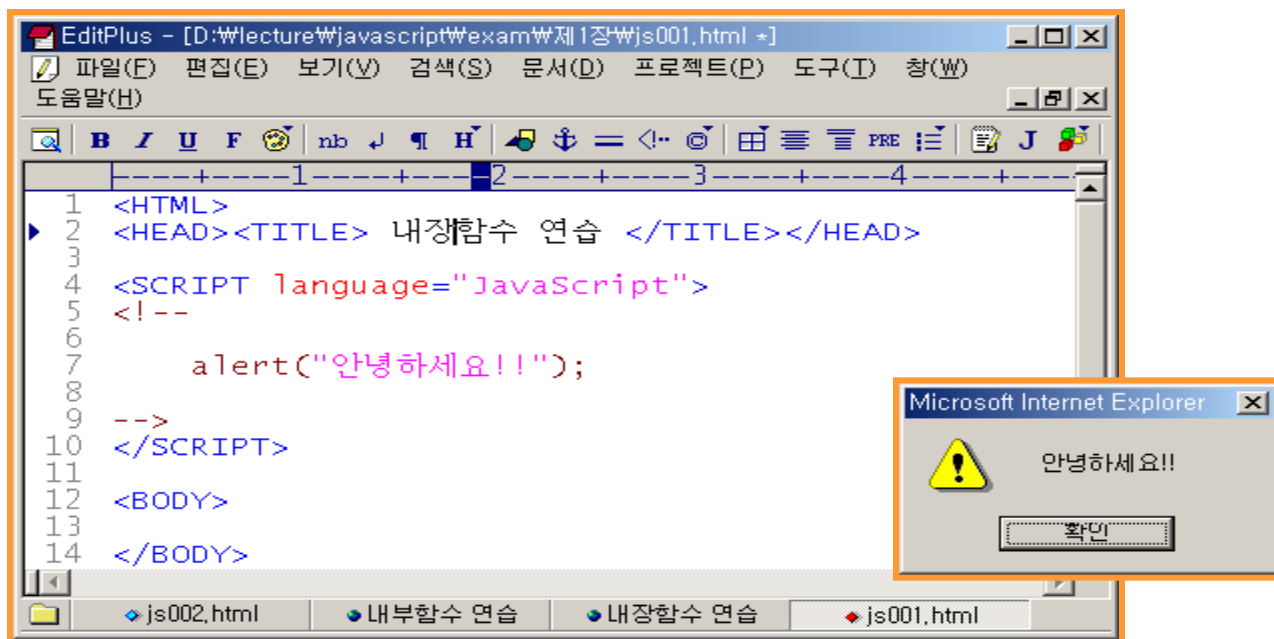
- ◆ 자바 스크립트에서 미리 만들어 사용자에게 제공되는 함수를 말함
- ◆ Alert, Prompt, Confirm등이 있음

# ◆ 내장함수

## ◆ alert()

alert( "문자열" )

- 문자열에 원하는 내용을 입력해주면 대화상자가 나타남
- 문자열을 항상 큰따옴표(")를 사용해야 함

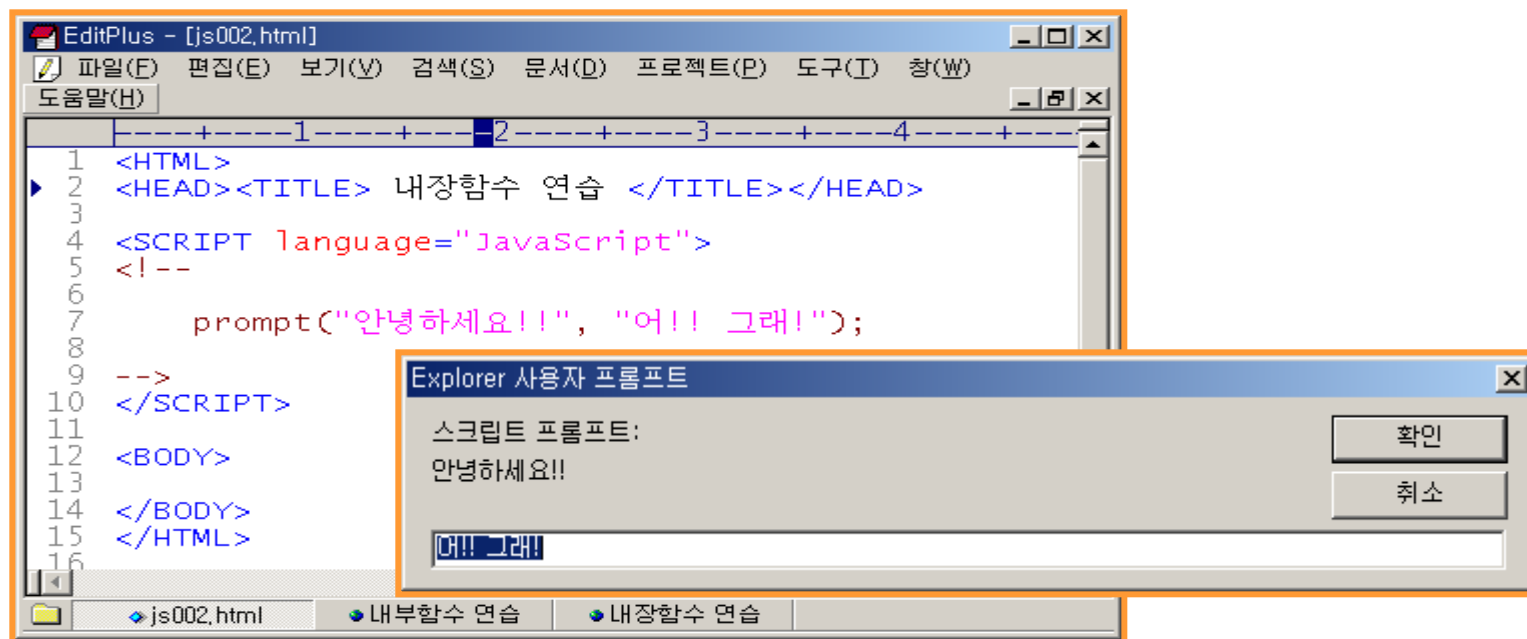


# ◆ 내장함수

## ◆ prompt()

**prompt( "문자열" , " 초기값 " )**

- 화면에 대화상자를 표시하고 사용자로부터 원하는 정보를 키보드로 입력 받을 수 있음



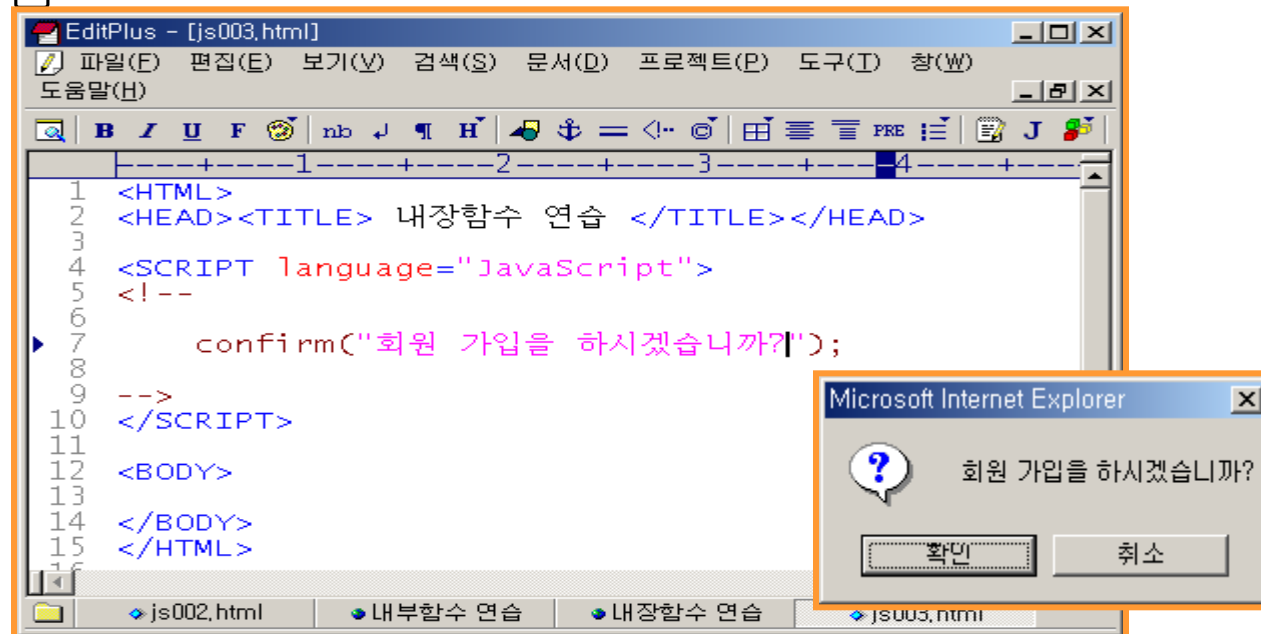


# ◆ 내장함수

## ◆ confirm()

`confirm( "문자열" )`

- 문자열에 원하는 내용을 넣으면 대화상자가 나타남
- 사용자가 “확인(True)”, “취소(False)”버튼 중에 하나를 선택할 수 있음



Sample

◆ 내장함수.html

# ◆ 내장함수

## ◆ 문자열을 숫자로 바꾸기

### ▪ parseFloat()

parseFloat( 문자열 )

- ()숫자로 구성된 문자열을 입력받아 실수값을 돌려준다.

### ▪ parseInt()

parseInt( 문자열 , 진수 )

- 문자열을 정수형으로 바꾸어주는 함수
- 문자열 대신 실수를 넣으면 정수형으로도 바꾸어줌
  - (소수점자리 버림)

# ◆ 내장함수

## ◆ eval()

eval( 문자열 )

- 문자열로 입력된 수식을 계산하는 함수
- 숫자로 구성된 문자열을 입력 받아 숫자로 변환하여 결과값을 돌려준다.
- 문자열은 표현식, 숫자, 변수를 포함할 수 있다

## ◆ String()

String( 객체 )

- 객체를 문자열로 변환하는 함수

Sample

◆ Parse예제.html

# ◆ 외부함수

- ◆ 외부함수는 HTML문서상 < HEAD >부분에 써줘야 함.
  - 함수가 정의되기 전에 함수가 사용되는 것을 방지하기 위해서
- ◆ 반복되는 문장을 최소화할 수 있고 소스도 간결해져 빠르게 코딩할수있다.
- ◆ 매개변수(parameter)
  - 함수를 호출하는 곳에서 사용하는 목적에 따라 적절한 값을 지정하여 함수를 호출함.
- ◆ 매개변수가 없는 함수 - **function** 함수이름( ){ }
- ◆ 매개변수가 있는 함수 - **function** 함수이름(param) [,param]...){ }
- ◆ 값을 돌려주기 위해 return을 사용 할 수도 있다.

```
function 함수이름( [param] [,param]...[,param]){  
    실행문장 ;  
  
    .....  
    [ return 반환값이나 반환변수 ; ]  
}
```

▪반환값이 있는 함수일 경우에만 사용하고  
반환값이 없는 함수에는 사용하지 않는다.

# ◆ 외부 함수

## ◆ 매개변수 없는 함수

```
<HTML><HEAD><TITLE>매개변수없는 함수</TITLE>
<SCRIPT language="javascript">
  <!--
  function test() {
    document.write("이건 매개변수가 없어요!!!<p>")
    document.write("안녕~~~")
  }

  document.write("함수를 호출해보자"+"<hr>")

  test(); //함수호출하는 부분
  //-->
</script></head>
</HTML>
```

Sample

◆매개변수없는함수.html



# 외부 함수

## ◆ 매개변수 있는 함수

```
<HTML><HEAD><TITLE>매개변수 있는 함수</TITLE>
<SCRIPT language="javascript">
<!--
    function test(a) {
        document.write("당신의 이름은 무엇입니까?<p>")
        document.write("제 이름은 " + a + "입니다.")
    }
//-->
</SCRIPT>
</HEAD><BODY>
<SCRIPT language="javascript">
<!--
    test("이쁜 희정이~~^^"); //함수호출하는 부분
//-->
</SCRIPT>
</BODY></HTML>
```

### Sample

◆매개변수 있는 함수.html

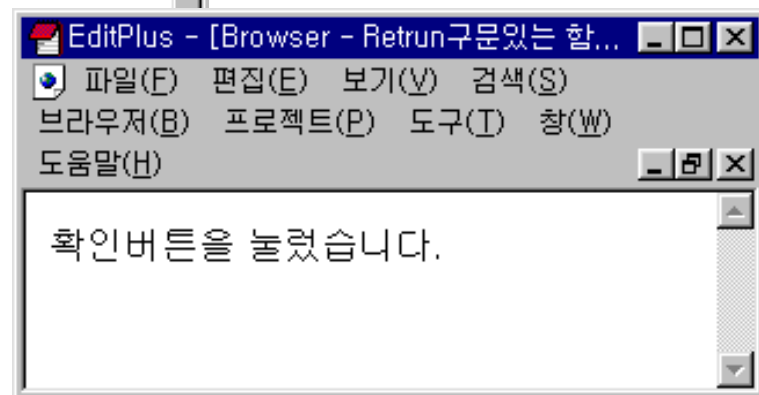
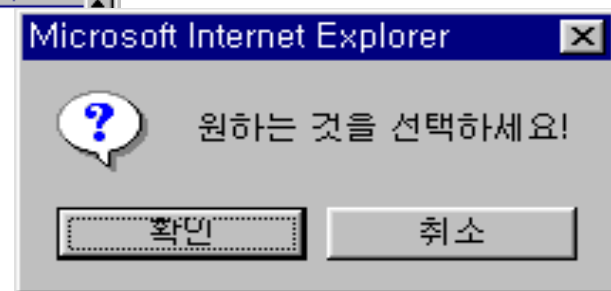
◆매개변수 있는 함수2.html

# ◆외부함수

## ◆ Return구문 있는 함수

```
EditPlus - [Return함수.html *]
파일(F) 편집(E) 보기(V) 검색(S) 문서(D) 프로젝트(P) 도구(T) 창(W) 도움말(H)

1
2 <HTML><HEAD><TITLE>Retrun구문있는 함수</TITLE>
3   <SCRIPT language="javascript">
4     <!--
5       function confirm_text( a ) {
6         var b=confirm(a);
7         if(b==true)
8           return "확인버튼을 눌렀습니다.";
9         else
10          return "취소버튼을 눌렀습니다.";
11       }
12     <!-->
13   </SCRIPT>
14 </HEAD>
15 <BODY>
16   <SCRIPT language="javascript">
17     <!--
18       var result = confirm_text("원하는 것을 선택하세요!");
19       document.write(result);
20     <!-->
21   </SCRIPT>
22 </BODY></HTML>
23
```



Sample

◆Return함수.html

# ◆외부함수

## ◆ 함수를 이용하여 문제를 푸세요^^

- 이름을 입력받아 이름출력(prompt이용) -인수/리턴없음
- 색깔을 입력받아 바탕색 바꾸기 (prompt이용) - 색상인수있음
- 이름과 태어난 년도를 입력받아 alert()에 이름과 나이 출력
  - - 이름과 년도 인수 받음.
- Confirm을 이용하여 성별구별하기.
  - - 성별 리턴
- onClick="함수이름()" 이용하여 바탕색 바꾸기
  - - 두번째 함수를 이용함

Sample

◆함수Test.html



# JavaScript

교사 : 장 희 정

# ◆ 제어문

## ◆ 문장의 흐름을 제어하는 문장

## ◆ 종류

### ■ 조건문

- 주어진 조건에 따라 프로그램을 다르게 수행하도록 하는 문장.
- if~else문, switch~case문.

### ■ 반복문

- 주어진 일을 빠르게 반복하기 위해 반복문을 사용한다.
- while문, do-while문, for문

# ◆조건문

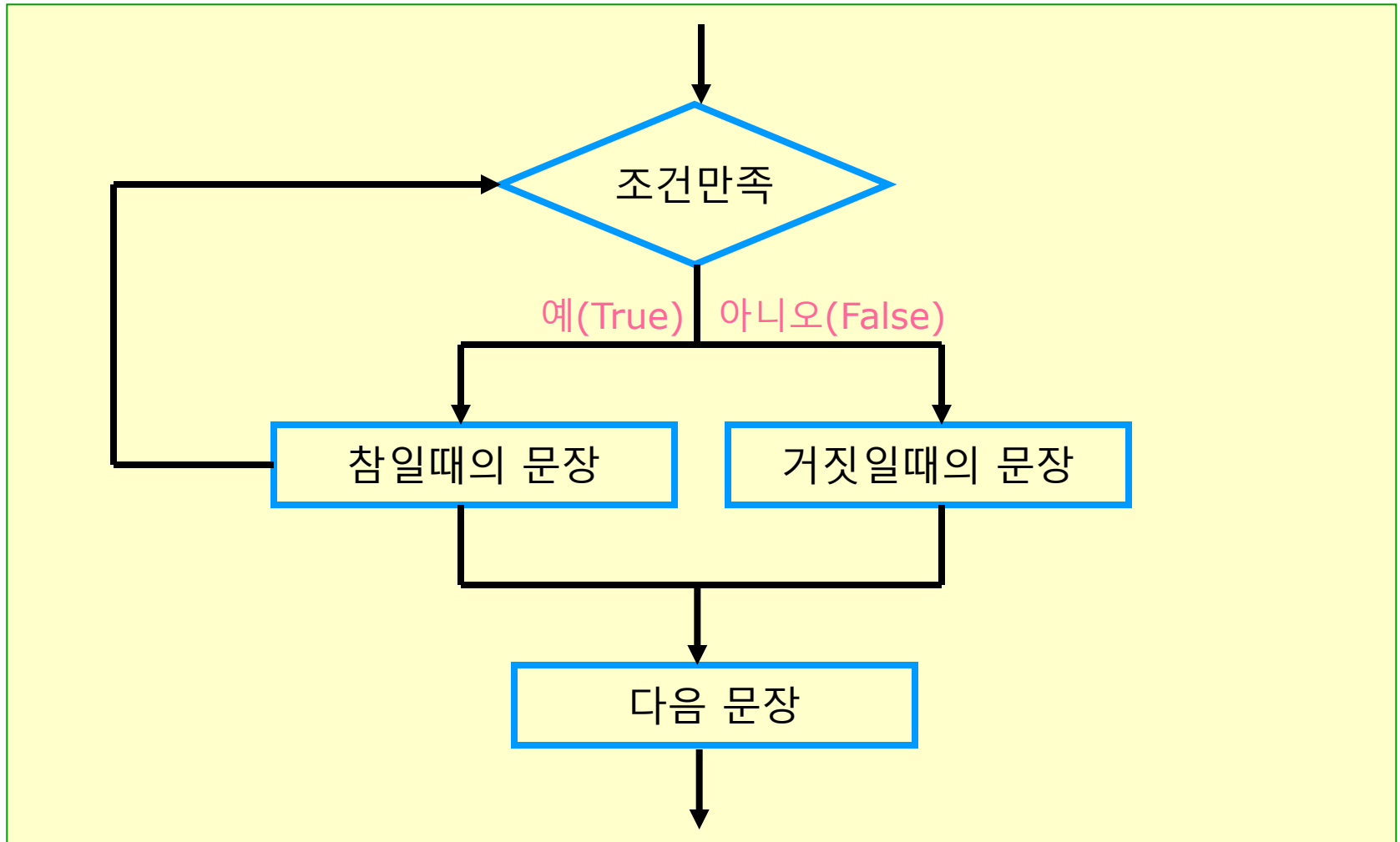
## ◆ IF문

- 조건을 판단한 후에 조건에 따라 분기하는 조건문
- 관계연산자와 논리연산자 등을 이용하여 조건을 만들 수 있다.

```
if ( 조건1 ) {  
    실행문 1 ;  
    실행문 2 ;  
}  
else if ( 조건2 ) {  
    실행문 3 ;  
    실행문 4 ;  
}  
else {  
    실행문 5 ;  
    실행문 6 ;  
}
```

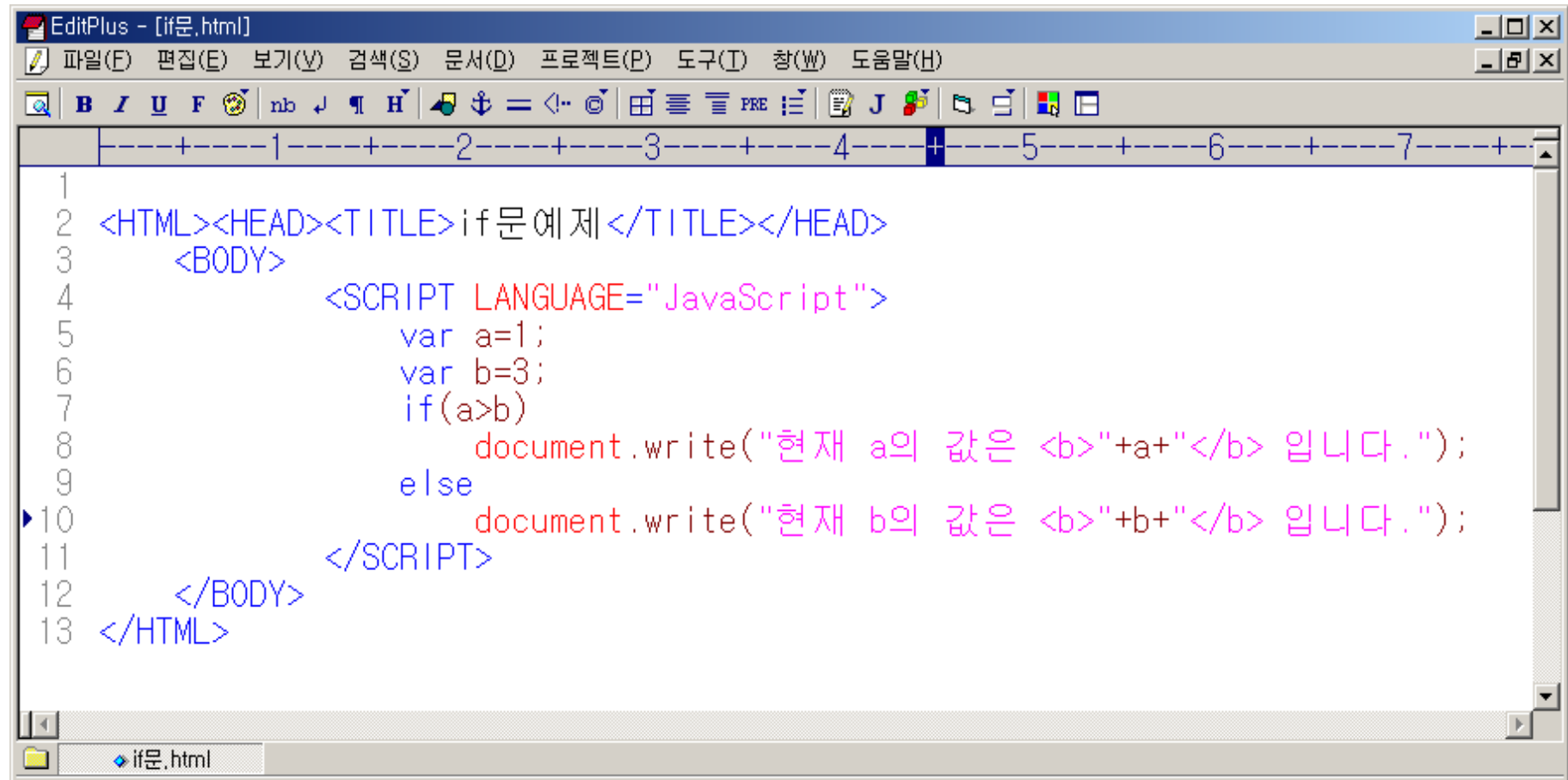
# ◆ 조건문

## ◆ IF문



# ◆ 조건문

## ◆ IF문 예제



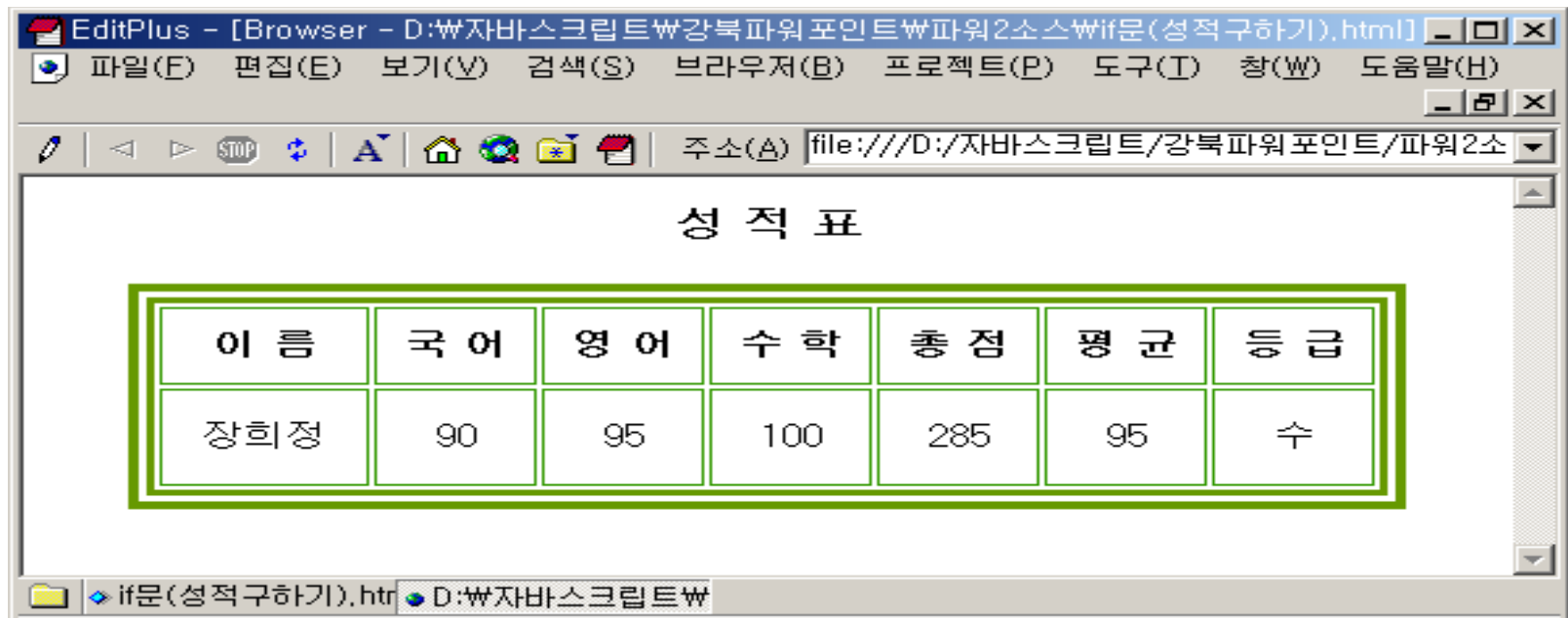
The screenshot shows the EditPlus text editor with a file named 'if문.html'. The code is as follows:

```
1
2 <HTML><HEAD><TITLE>if문예제</TITLE></HEAD>
3   <BODY>
4       <SCRIPT LANGUAGE="JavaScript">
5           var a=1;
6           var b=3;
7           if(a>b)
8               document.write("현재 a의 값은 <b>"+a+"</b> 입니다.");
9           else
10              document.write("현재 b의 값은 <b>"+b+"</b> 입니다.");
11      </SCRIPT>
12  </BODY>
13 </HTML>
```

# ◆ 조건문

## ◆ IF문 예제

- 홀수, 짝수 구분하기(prompt이용) – [홀짝구분.html](#)
- prompt을 이용하여 이름, 국어, 영어, 수학 점수를 각각 입력 받아 아래와 같이 출력하세요.
  - [if문\(성적구하기\).html](#)



# ◆조건문

## ◆Switch문

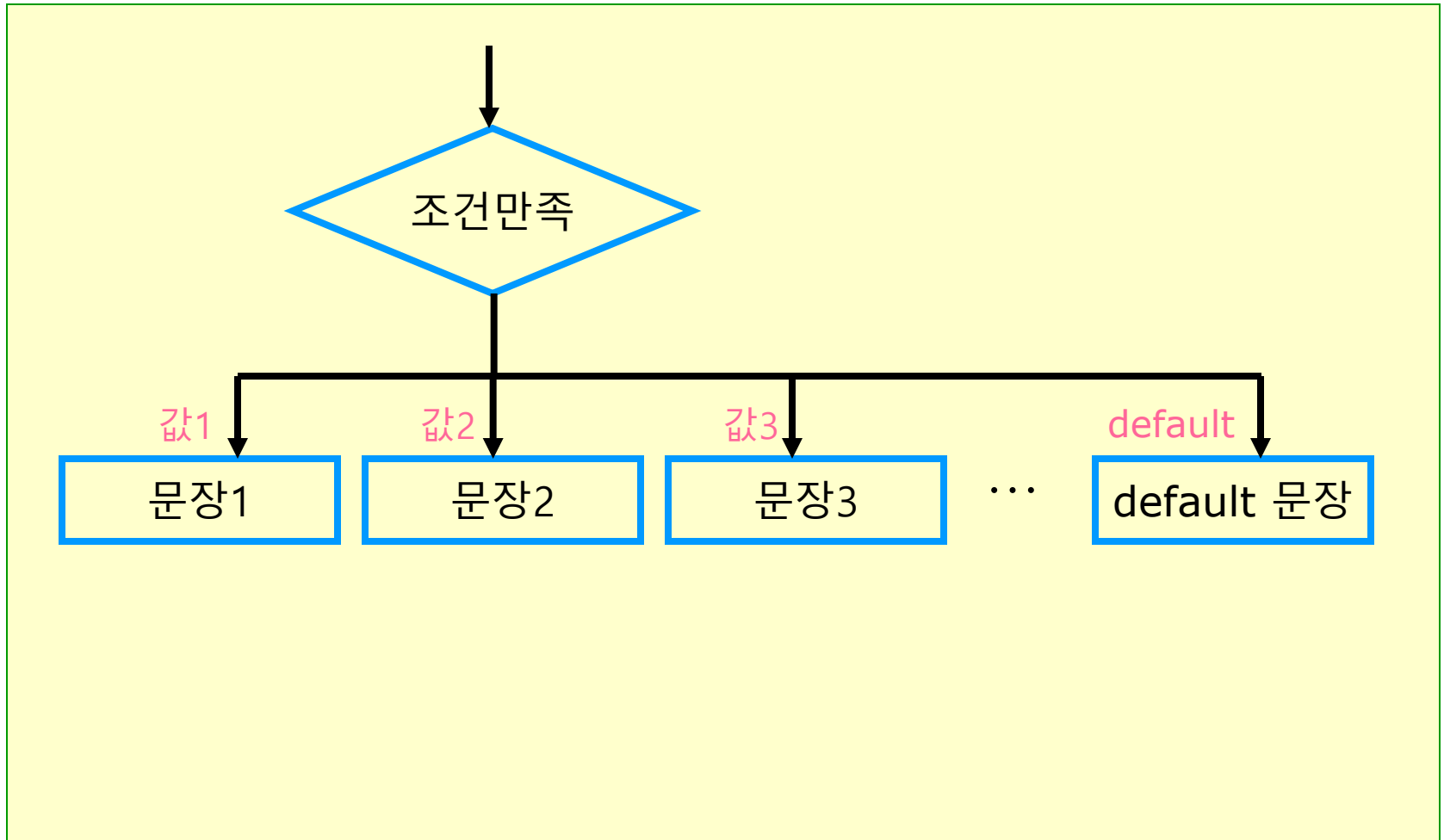
- 여러 단계의 조건을 비교하는 경우 다중 if문을 사용하는 대신 간편하게 사용할 수 있음

```
switch (변수) {  
    case 상수1 :  
        문장1;  
        break;  
    case 상수2 :  
        문장2;  
        break;  
    default :  
        문장3;  
}
```

- 반복문의 제어를 담당하는 명령문
- 진행되던 작업이 중단됨

# ◆ 조건문

## ◆ Switch문





# ◆ 조건문

## ◆ Switch문 예제

- Prompt를 이용하여 두개의 수와 연산기호를 입력받아 입력된 연산기호에 따라 두수를 연산한다.
  - [switch\(연산\).html](#)
- Prompt를 이용하여 달수를 입력받아 입력된 달수의 마지막 일을 구하세요.
  - [Switch\(달수\).html](#)
- prompt을 이용하여 이름, 국어 ,영어,수학 점수를 각각 입력 받아 아래와 같이 출력하세요.
  - [switch문\(성적구하기\).html](#)

# ◆반복문

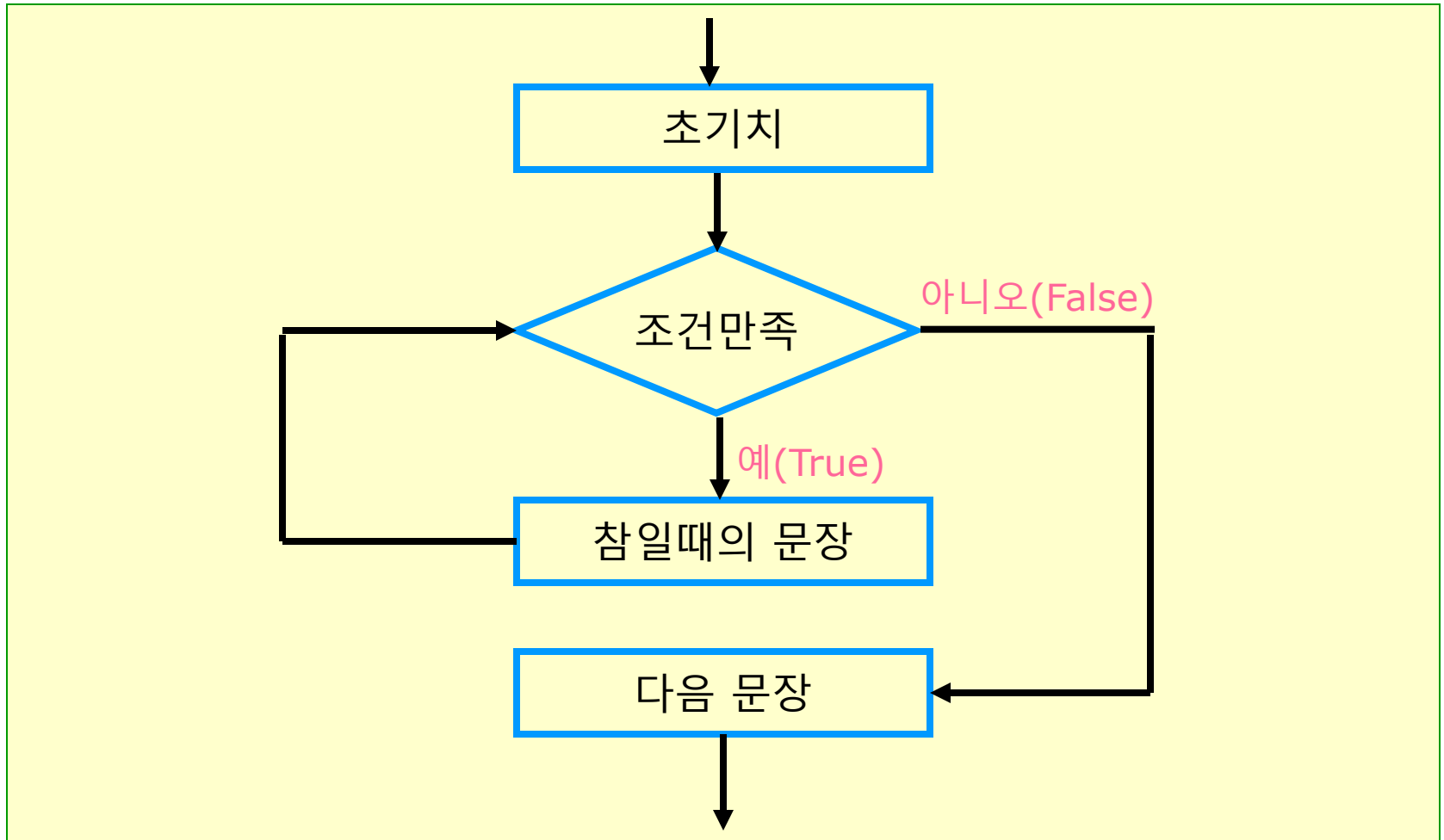
## ◆For문

- 초기값, 조건식, 증감식을 설정하여 반복적인 수행을 할 수 있다.
- 규칙적인 증가를 할 때 많이 사용함

```
for (초기값;  조건식;  증감식) {  
    문장  ;  
}
```

# ◆ 반복문

## ◆ For문



# ◆반복문

## ◆For문 예제

- 1 ~ 100까지 출력 – for문(1~100).html
- 구구단 짜기 – Prompt사용 (for문(구구단).html)
- 구구단 짜기 – Table사용 (for문(구구단테이블).html)

# ◆반복문

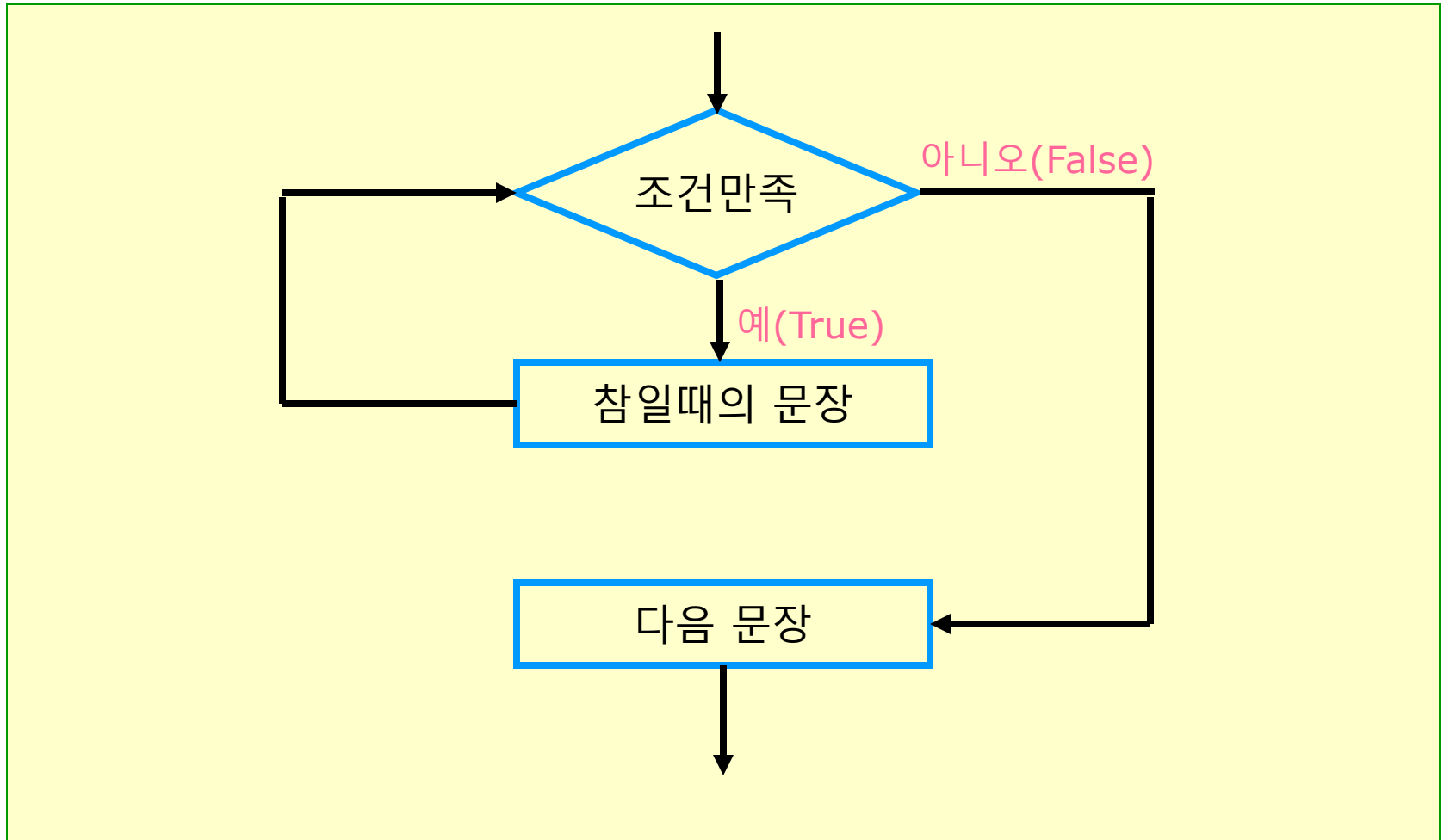
## ◆While문

- 조건이 만족(True)일때만 문장을 반복하고 만족하지(False) 않을 때에는 While문 다음 문장을 수행함
- 조건을 먼저 확인하기 때문에 처음 조건을 만족하지 않으면 반복 문장을 한번도 수행하지 않는다.

```
while ( 조건식 ) {  
    문장;  
}
```

# ◆ 반복문

## ◆ While문



# ◆ 반복문

## ◆ While문 예제

- 1 ~ 100까지 출력 – while문(1~100).html
- 구구단 짜기 – Prompt사용 – while문(구구단).html
- 구구단 짜기 – Table사용 – (while문(구구단테이블).html)

# ◆ 반복문

## ◆ Do-While문

- 반복문장을 수행한 후에 조건만족(True)을 확인함
- 조건이 만족되던, 틀리던 무조건 한번은 수행을 함

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



# ◆ 반복문

## ◆ Do - While문 예제

- 1 ~ 100까지 출력 - do-while문(1~100).html
- 구구단 짜기 - Prompt사용 - do-while문(구구단).html
- 구구단 짜기 - Table사용 - (do-while문(구구단테이블).html)

# ◆ Break/Continue

◆ 조건문이나 반복문에서 순서의 흐름을 제어하는데 사용

◆ break

- 조건문이나 반복문에서 반복구간이나 조건문안을 빠져나갈 때 사용
- Break문장을 만나면 break이하의 문장 실행하지 않고 그 반복문이나 조건문을 빠져나간다.

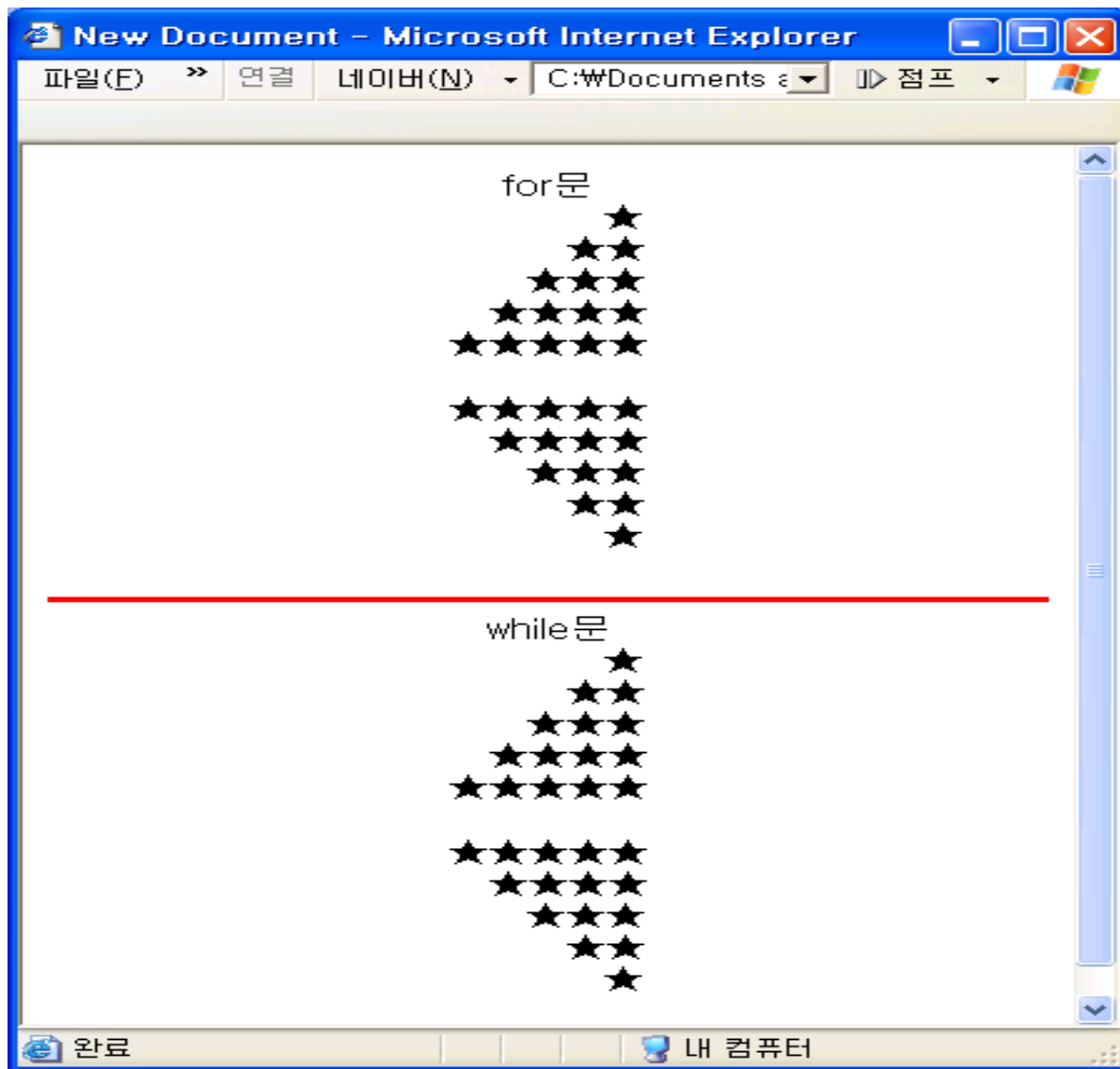
◆ continue

- 조건문이나 반복문에서 다시 조건이나 반복의 위치로 다시 되돌아갈 때 사용
- Continue문장을 만나면 continue이하의 문장 실행하지 않고 조건이 남아있으면 다시 반복문 실행한다.

Sample

◆ Break, continue 예제 .html

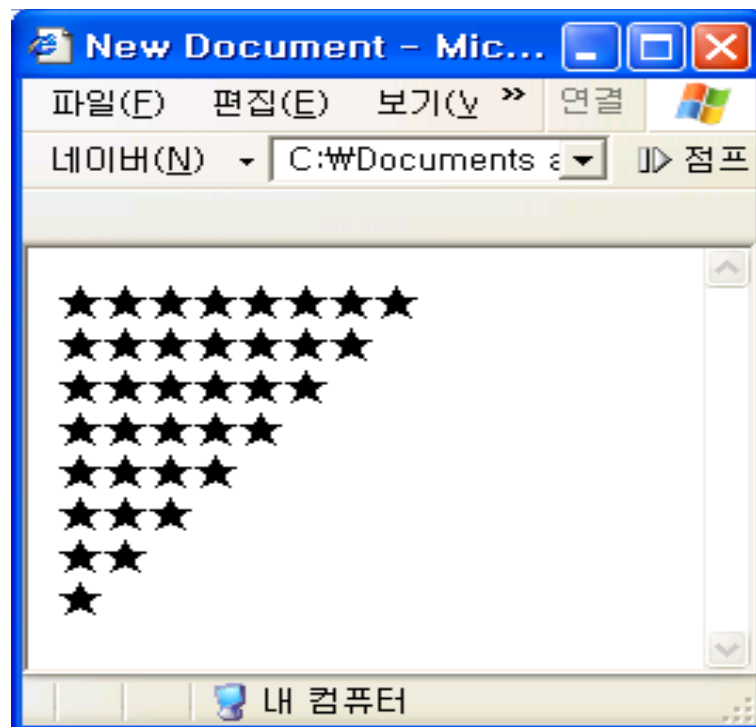
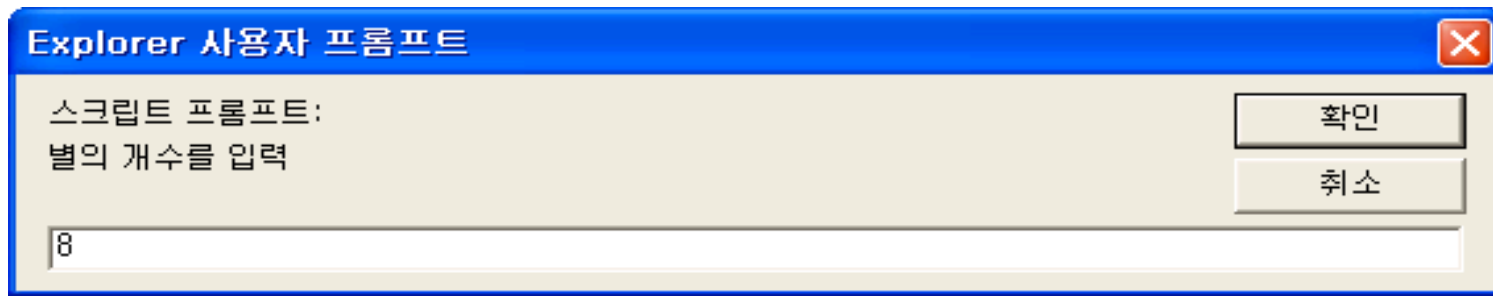
# ◆ 별찍기 테스트



Sample

◆ 별찍기.html

# ◆ 별찍기 테스트



Sample

◆ 별찍기2.html

# ◆ 객체

## ◆ 자바 스크립트는 객체 기반의 언어임

- 객체지향 언어에서 클래스와 상속성을 사용하지 않은 언어임

## ◆ 객체(Object)

- 공통적인 특성을 가진 속성이나 동작들을 묶어놓은 그룹.
- 사용자 정의 객체
  - 사용자가 임의적으로 정의하여 사용하는 객체
- 내장 객체
  - 자바스크립트에 이미 정의 된 객체
  - (예) Array(), String(), navigator객체, window객체, date객체, document객체, form객체 등이 있다.
- 객체는 그 속성(Property)을 표현하는 데이터와 그 데이터를 이용해 특별한 작업을 하는 메소드(Method)로 구성되어 있다.

# ◆객체의 구성

◆객체의 구성 = 속성 + 행동(Method)

◆속성(Property)

- 다른 것과 어떤 객체를 구분하고, 그 객체의 다른 성질, 상태, 모습 등을 결정하는 개별적인 것

◆메소드(Method)

- 개체의 동작을 지정하는 함수

# ◆ 객체 사용법

- ◆ 마침표(.)를 사용함
- ◆ '객체명 . 속성명' 또는 '객체명 . 메소드명'
  - ex) document . write(~~~)
  - 객체이름.속성 혹은 메소드

document.lastModified	document.write("문자열")
<ul style="list-style-type: none"><li>➤ document객체를 이용하여 문서가 마지막으로 변경된 날짜를 구할 때 사용.</li><li>➤ 객체이름을 쓰고 다음에 그 객체의 속성을 적어 주면 됨</li><li>➤ 객체의 속성은 그 객체에서만 사용할 수 있음</li></ul>	<ul style="list-style-type: none"><li>➤ document객체의 메소드를 이용하여 문자열을 출력하는 것.</li><li>➤ write메소드는 document객체에서만 사용할 수 있다.</li></ul>

# ◆ 객체 사용법

## ◆ 객체 사용하기

- 상위객체이름.하위객체이름.속성 혹은 메소드

`document.form.text1.value`

- 현재객체에 상위객체가 존재한다면 같이 써주어야 함.
- 상위객체를 지정해 주지 않는 경우가 있는데 이는 그것은 객체가 현재 스크립트가 쓰여져 있는 그 객체를 가리킨다는 것을 의미함



# ◆ 객체 조작성

## ◆ New()

- 만들어져 있는 객체 혹은 내장객체를 이용하여 새로운 객체 인스턴스 (instance)를 만들 때 사용하는 객체 생성자

```
객체변수 = new 객체명();  
객체변수 = new 객체명( 매개변수1, 매개변수2, ... );
```

## ◆ This

- 이것은 객체 타입에 상관없이 현재의 객체를 지칭하는 키워드

```
this.속성  
this.하위객체
```

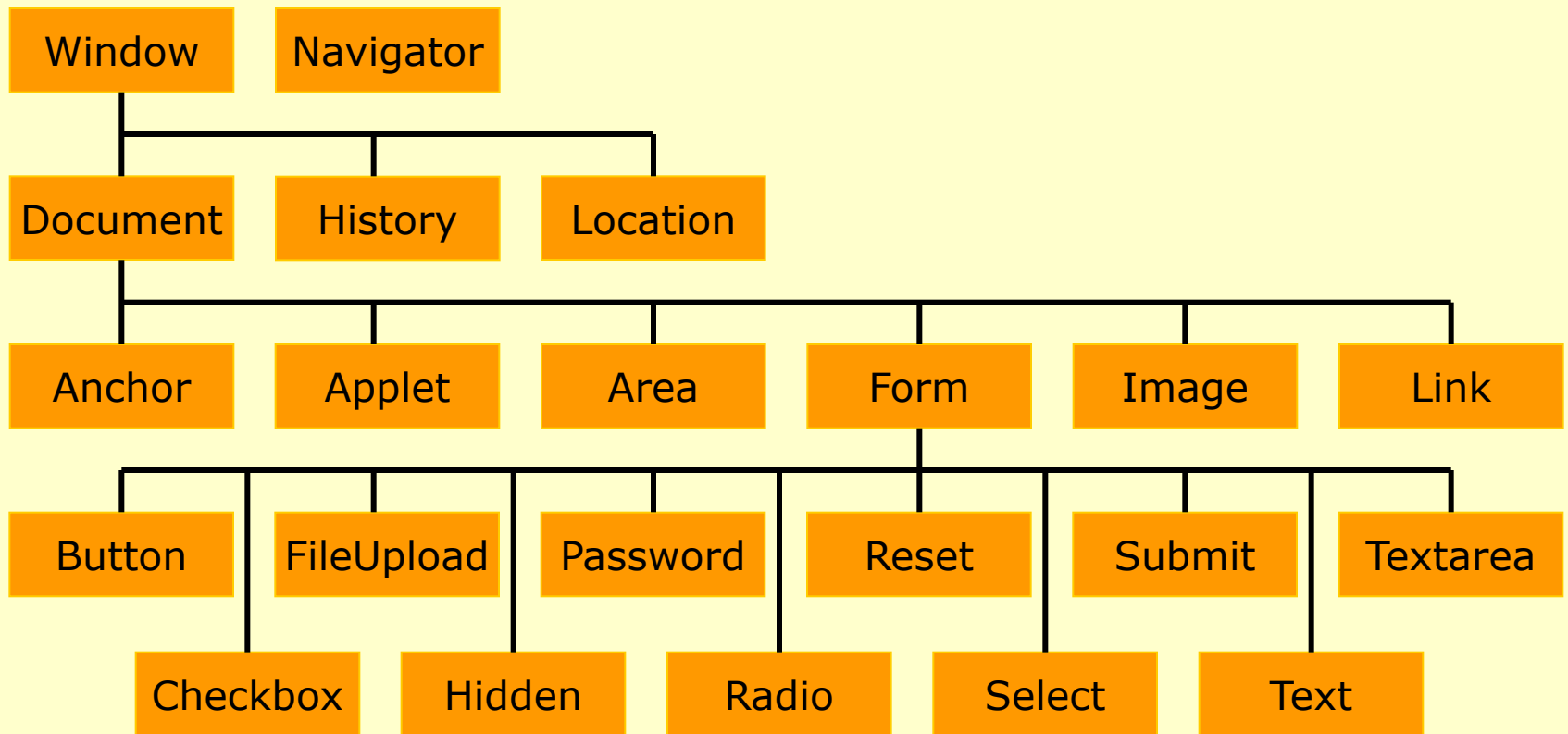
# ◆ 자바스크립트 내장객체

- ◆ 자바 스크립트에서 화면정보, 날짜, 배열 등을 알아내거나 사용할 수 있게 만들어놓은 객체

객체이름	설명
Arguments	함수가 호출될 때 함수내의 매개변수 정보를 읽어옴 Explorer에서 지원안됨
Array	배열을 사용할 수 있게 함
Date	날짜와 시간을 다룰 수 있게 함
Function	함수를 정의하여 객체와 동일한 형태로 사용할 수 있게 함
Math	수학적인 함수와 특수 함수를 제공함
Number	문자로 된 숫자를 숫자로 바꾸어줌
Screen	현재 사용중인 웹 페이지의 해상도, 색등의 정보를 보여줌
String	문자열을 다룰 수 있게 함

# ◆ 웹 브라우저의 객체

## ◆ 웹 브라우저의 객체 구조



# ◆ 내장객체

## ◆ Math

- 수학에서 사용되는 삼각함수, 파이, 지수, 절대값 등을 처리할 수 있게 속성이나 메소드를 제공함
- new 연산자를 사용하지 않음
  - Math.메소드()
  - Math.속성

속성	설명
E	자연로그 밑에 사용하는 오일러 상수
PI	원주율 ( 3.141592... )
LN2	자연로그
SQRT2	제곱근

# ◆ 내장객체

## ◆ Math

Method	설명	Method	설명
abs(x)	x의 절대값을 반환	acos(x)	역코사인(arc cosine)
asin(x)	역사인(arc sine)	atan(x)	역 탄젠트(arc tangent)
ceil(x)	x값의 소수부분올림	cos(x)	cosine
floor(x)	x값의 소수부분버림	log(x)	로그
max(x, y)	x와 y중에 큰 수를 반환	exp(x)	지수함수
min(x, y)	x와 y중에 작은 수를 반환	pow(x, y)	x의 지수 y
random()	난수를 발생	round(x)	x를 반올림
sin(x)	sine	sqrt(x)	제곱근
tan(x)	tangent		

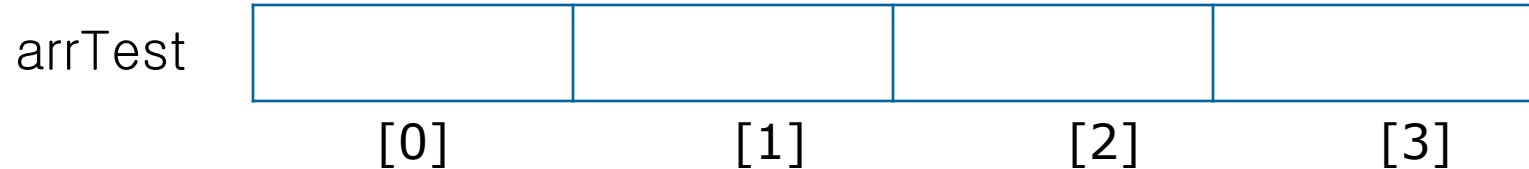
Sample

◆Math예제.html

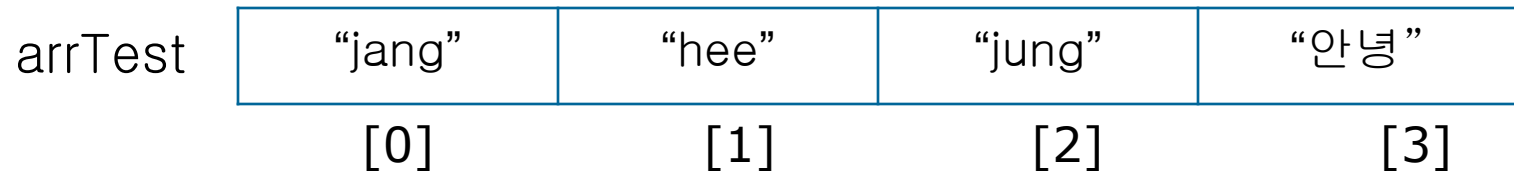
# ◆ 내장객체

## ◆ Array

```
arrTest = new Array(4)
```



```
arrTest [0] = "jang" ;  
arrTest [1] = "hee" ;  
arrTest [2] = "jung" ;  
arrTest [3] = "안녕" ;
```



# ◆내장객체

## ◆ Array

Method	설명
join([문자])	배열을 하나의 문자열로 만들어줌
reverse()	배열의 순서를 역순으로 바꾸어 줌
sort()	배열 값을 정렬함
slice(시작 , 끝)	배열의 일부분을 선택하여 새로운 배열을 만 들
concat(배열이름)	두개의 배열을 하나의 배열로 만들어 줌

배열이름.length

### Sample

- ◆배열선언.html
- ◆배열함수.html
- ◆배경색배열이용.tml
- ◆배경색바꾸기.html

# ◆ 내장 객체

## ◆ Date

- 시스템에 있는 날짜와 시간을 자바 스크립트에서 사용할 수 있게 해줌

Method	설 명
getFullYear()	1970년 이상의 년을 구함
getMonth()	월을 구함(0->1월, 1->2월, ... , 11->12월)
getDate()	일을 구함
getDay()	요일을 구함 (0->일, 1->월, 2->화, ... , 6->토)
getTime()	1970년 1월 1일 이후 시간을 ms로 나타낸 값
getHours()	오전/오후를 표시하지 않는 시를 구함
getMinutes()	분을 구함
getSeconds()	초를 구함





# 내장객체

## ◆ Date

Method	설명
setYear()	1970년 이상의 년을 구함
setMonth()	월을 설정(0->1월, 1->2월, ... , 11->12월)
setDate()	일을 설정
setDay()	요일을 설정 (0->일, 1->월, 2->화, ... , 6->토)
setTime()	1970년 1월 1일 이후 시간을 ms로 나타낸 값
setHours()	오전/오후를 표시하지 않는 시를 설정
setMinutes()	분을 설정
setSeconds()	초를 설정
toString()	'Tue Feb 5 09:10:34 UTC+0900 2002'로 표시됨
toLocaleString()	'02/05/2002 09:10:34'로 표시됨
toGMTString()	Explorer는 UTC 날짜로 반환됨

# ◆내장객체

## ◆Date객체 예제

Microsoft Internet Explorer window showing a calendar for September 2011. The address bar shows the file path: F:\자바스크립트\강북과워포인트\파워4소스\Date\달력.html.

2011년 9월

일요일	월요일	화요일	수요일	목요일	금요일	토요일
★	★	★	★	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15 AM 10:00:13	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	※

### Sample

- ◆Date.html
- ◆현재시간날짜.html
- ◆dayNumber.tml
- ◆상태바에날짜출력html
- ◆timer.html
- ◆달력.html

# ◆ 내장객체

## ◆ String

- 글자모양, 위치이동, 문자열 다루기 등을 할 수 있게 다양한 메소드를 제공함
- new 연산자를 사용하지 않음
  - “문자열”.메소드()
  - “문자열”.속성

Method	태 그 명 령	설 명
big()	<big></big>	글자를 조금크게함.
bold()	<b></b>	굵은체
fontcolor(“색상”)	<font color=“색상”></font>	글자색
italics()	<i></i>	이탤릭체
strike()	<strike></strike>	가운데 줄 그은글자체
fixed()	<tt></tt>	타자기체
fontsize(“크기”)	<font size=“색상”></font>	글자 크기
small()	<small></small>	글자를 조금 작게
sub(), sup()	<sub></sub>, <sup></sup>	아래첨자, 위첨자
link(“이동위치”)	<a href=“이동할위치”></a>	다른위치나 다른사이트로이동

# ◆ 내장객체

## ◆ String

Method	설명	사 용 예
charAt(index)	index 위치의 문자를 알아냄	"scr ipt".charAt(4)→ p
concat("문자열")	문자열에 문자열을 연결시켜줌	
indexOf("문자열")	문자열중에 찾는 문자열의 인덱스 위치를 왼쪽에서부터 찾아냄	"script".indexOf("r") → 2
lastIndexOf("문자열")	문자열중에 찾는 문자열의 인덱스 위치를 오른쪽에서부터 찾아냄	"script".lastIndexOf("c") → 1
slice(index1, index2)	index1에서 index2사이의 문자열을 추출	
split(구분문자, 개수)	문자열에서 구분문자를 이용해 개수만큼 구분해줌	"sc ric pc t".split(" ",3)
substr(index, length)	문자열에서 인덱스위치부터 길이까지 문장 추출	"script".substr(1,3) → cri
substring(index1, index2)	index1에서 index2사이의 문자열 추출	"script".substring(1,3) → cr
toLowerCase()	소문자로 바꾸어줌	
toUpperCase()	대문자로 바꾸어줌	

# ◆내장객체

## ◆ String예제

- 문자열체크 – charAt(index번호)와 indexOf(“찾을문자열”)이용
- 주민번호체크 – charAt(index번호) 와 onkeyup이벤트이용
- 메일체크 – indexOf(“@”) 이용

### Sample

- ◆String태그.html
- ◆String메소드.html
- ◆문자열체크.html
- ◆주민번호.html
- ◆메일체크.html

# ◆객체조작문

## ◆typeof

- 변수의 데이터형을 알아내기 위해서 사용됨
- 특정 문자열이 반환되어 숫자인지 문자열인지 확인 가능함

속성	반환하는 문자열
typeof “문자열”	“string”
typeof 숫자	“number”
typeof 부울값(True, False)	“boolean”
typeof 객체	“object”
typeof null	“object”

# ◆ Event란?

## ◆ Event

- 사용자가 키를 누르거나 마우스의 버튼을 누른 경우에 발생됨
- 특정한 동작이 발생할 경우에 발생하는 신호를 말함

## ◆ Event Handler

- Event를 처리할 수 있는 함수나 메소드
- 이벤트 핸들러는 "**on이벤트타입**" 과 같이 사용하여 이벤트를 다루는 것으로, "어떤 어떠한 이벤트가 발생하면..." ~해라. 라는 식으로 해석한다.
- 이것은 어떤 이벤트가 발생했을 때 지정한 함수나 명령을 실행하도록 하는 것이다.

# ◆Event

Event	Event핸들러	설 명
load	onload	문서가 브라우저에 의해 읽혀지는 순간 함수나 명령줄을 실행하고자 할 때 발생
Unload	onUnload	현재 문서를 떠날때 발생
Abort	Onabort	문서나 이미지로딩이 중지되는 때 발생한다.
Error	Onerror	문서나 이미지로딩이 에러가 발생할때 이다.
Focus	Onfocus	창이나 텍스트 박스등을 선택하여 포커스가 주어질 때이다
Blur	Onblur	창이나 텍스트박스를 떠나 포커스가 해제 될때이다.
Chagne	Onchagne	입력양식필드에서 값이 바꿨을때
Click	Onclick	버튼이나 링크등을 마우스로 클릭할때
Dbclick	Ondblclick	마우스를 더블클릭 할때
move	Onmove	창이나 프레임이 움직일때
resize	onresize	창이나 프레임의 크기가 변경 될때

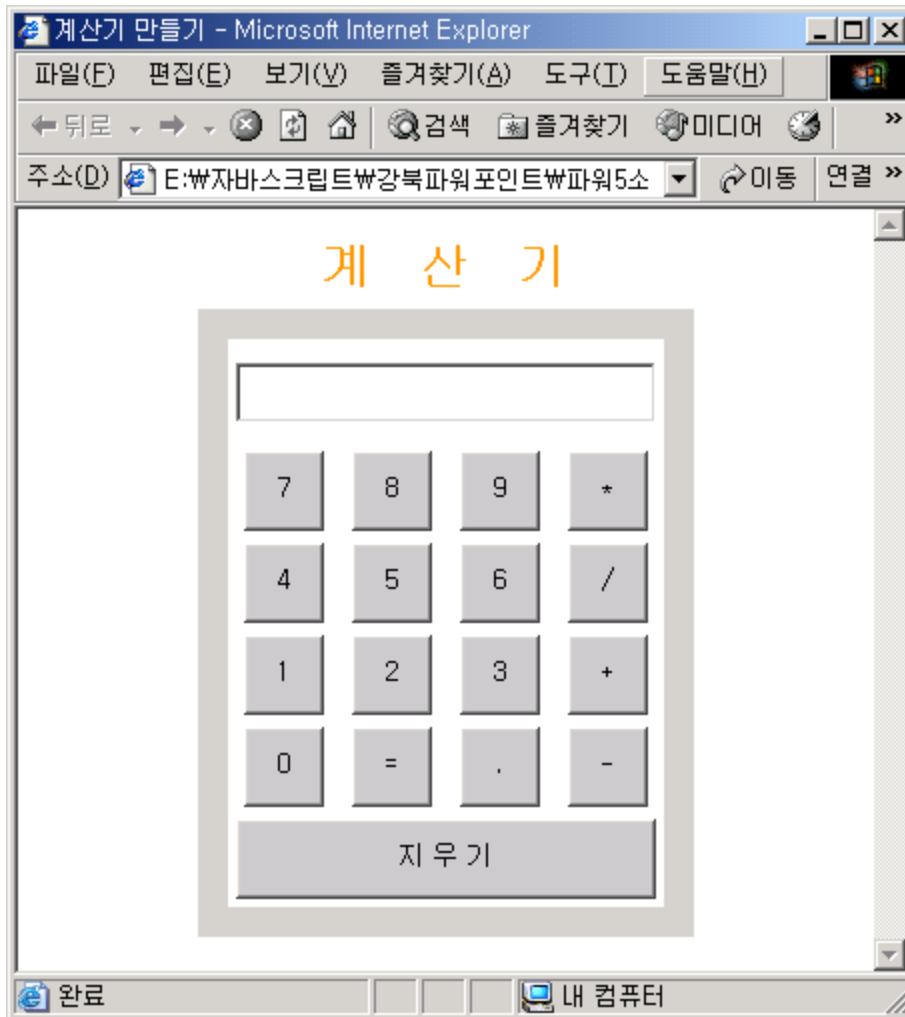


# ◆ Event

Event	Event핸들러	설명
Mouseover	onMouseover	그림이나 링크에 마우스가 올라 갔을때
Mouseout	onMouseout	마우스 커서가 그림이나 링크영역 밖으로 나갔을때
Mousedown	onMousedown	마우스 버튼을 누르는 순간
Mouseup	onMouseup	마우스 버튼을 눌렀다가 때는 순간
Keydown	onKeydown	키보드의 키를 눌렀을때
KeyPress	onKeyPress	키보드의 키를 누르는 순간
keyup	Onkeyup	키보드의 키를 눌렀다가 때는 순간
Dragdrop	onDragdrop	마우스를 클릭한 상태에서 드래그 했을 때 발생
Select	Onselect	입력 양식의 한 필드를 선택했을 때 발생
Submit	Onsubmit	Submit버튼을 눌러 폼을 넘길때
Reset	onreset	Reset버튼을 눌러 폼의 내용을 초기화 할때

# ◆ Event

## ◆ Event예제



### Sample

- ◆onLoad와 onUnload.html
- ◆onBlur,onfocus예제.html
- ◆onBlur,onfocus예제2.html
- ◆onChange.html
- ◆onClick,onDbIcIck.html
- ◆onMouseDown,onMouseUp.html
- ◆계산기 만들기

# ◆ Event객체

- ◆ 이벤트를 처리하기 위해서 제공되는 객체 상위객체이름.하위객체이름.속성 혹은 메소드

속성	설 명
altKey	alt 키 상태를 True, False값으로 알아냄
Button	버튼의 종류를 알아냄, 1-> 왼쪽, 2-> 오른쪽, 3-> 가운데
clientX	HTML문서를 기준으로 이벤트가 발생한 X좌표위치를 알아냄
clientY	HTML문서를 기준으로 이벤트가 발생한 Y좌표위치를 알아냄
ctrlKey	ctrl 키 상태를 True, False 값으로 알아냄
keyCode	키를 누른 상태를 알아냄
offsetX	컨테이너를 기준으로 이벤트가 발생한 X좌표 위치를 알아냄
offsetY	컨테이너를 기준으로 이벤트가 발생한 Y좌표 위치를 알아냄
reason	데이터 소스 객체에 대한 데이터 전송상태를 알아냄
returnValue	이벤트 핸들러의 반환값을 True, False로 알아냄
screenX	화면을 기준으로 이벤트가 발생한 X좌표 위치를 알아냄
screenY	화면을 기준으로 이벤트가 발생한 Y좌표 위치를 알아냄

# ◆ Event객체

Event	설 명
shiftKey	shift 키 상태를 True, False 값으로 알아냄
srcElement	이벤트가 전송된 원래 객체를 알아냄
type	이벤트의 종류를 설정함
X	상대적인 X좌표 위치를 알아냄
Y	상대적인 Y좌표 위치를 알아냄

## Sample

- ◆String태그.html
- ◆String메소드.html

# ◆ Window 객체

- ◆ 브라우저 내장 객체들의 계층구조 중에서 가장 상위에 있는 객체
- ◆ 자바 스크립트로 하는 모든 작업이 window 객체 안에서 이루어짐
- ◆ window 하위에 있는 객체를 가리킬 때 window를 생략해도 됨

# ◆ Window 객체

속 성	설 명
defaultStatus	상태바에 초기 문자열을 설정하는 경우
frames	window 객체 안에 들어갈 프레임들의 배열정보
opener	open() 메소드로 윈도우를 연 문서가 있는 윈도우
parent	window 객체간에 계층구조가 생길 때 바로 상위 객체
self	현재 활성화중인 창 의 자신 객체
top	window 객체간에 계층구조가 생길 때 최상위 객체
status	브라우저 상태바에 나타날 문자열
tags	HTML문서에 사용된 모든 태그들
classes	HTML문서에 정의된 모든 스타일시트 클래스들

# ◆ Window 객체

Method	설 명
alert()	메시지를 대화상자에 보여줌
confirm()	확인, 취소를 선택할 수 있는 대화상자를 보여줌
prompt()	문자열을 입력받을 수 있는 대화상자를 보여줌
moveBy(x위치, y위치)	상대적인 좌표로 이동
moveTo(x좌표, y좌표)	절대적인 좌표로 이동
resizeBy(x위치, y위치)	상대적인 좌표로 창의 크기 설정
resizeTo(x좌표, y좌표)	절대적인 좌표로 창의 크기 설정
open()	새로운 창을 열어줌
close()	open()으로 연 창을 닫음
print()	화면에 있는 내용을 프린터로 출력함

# ◆ Window 객체

Method	설 명
scrollBy(x위치, y위치)	상대적인 좌표로 스크롤 위치값을 바꿈
scrollTo(x좌표, y좌표)	절대적인 좌표로 스크롤 위치값을 바꿈
setInterval(호출함수, 간격)	일정간격으로 함수를 호출하여 수행함 함수 한번만 호출
setTimeout(호출함수, 간격)	일정간격으로 함수를 호출하여 수행함 함수를 재귀 호출하여 사용함
clearTimeout(변수)	setTimeout()이나setInterval()메소드에 의 해 일정한 간격으로 수행하는 함수를 중지

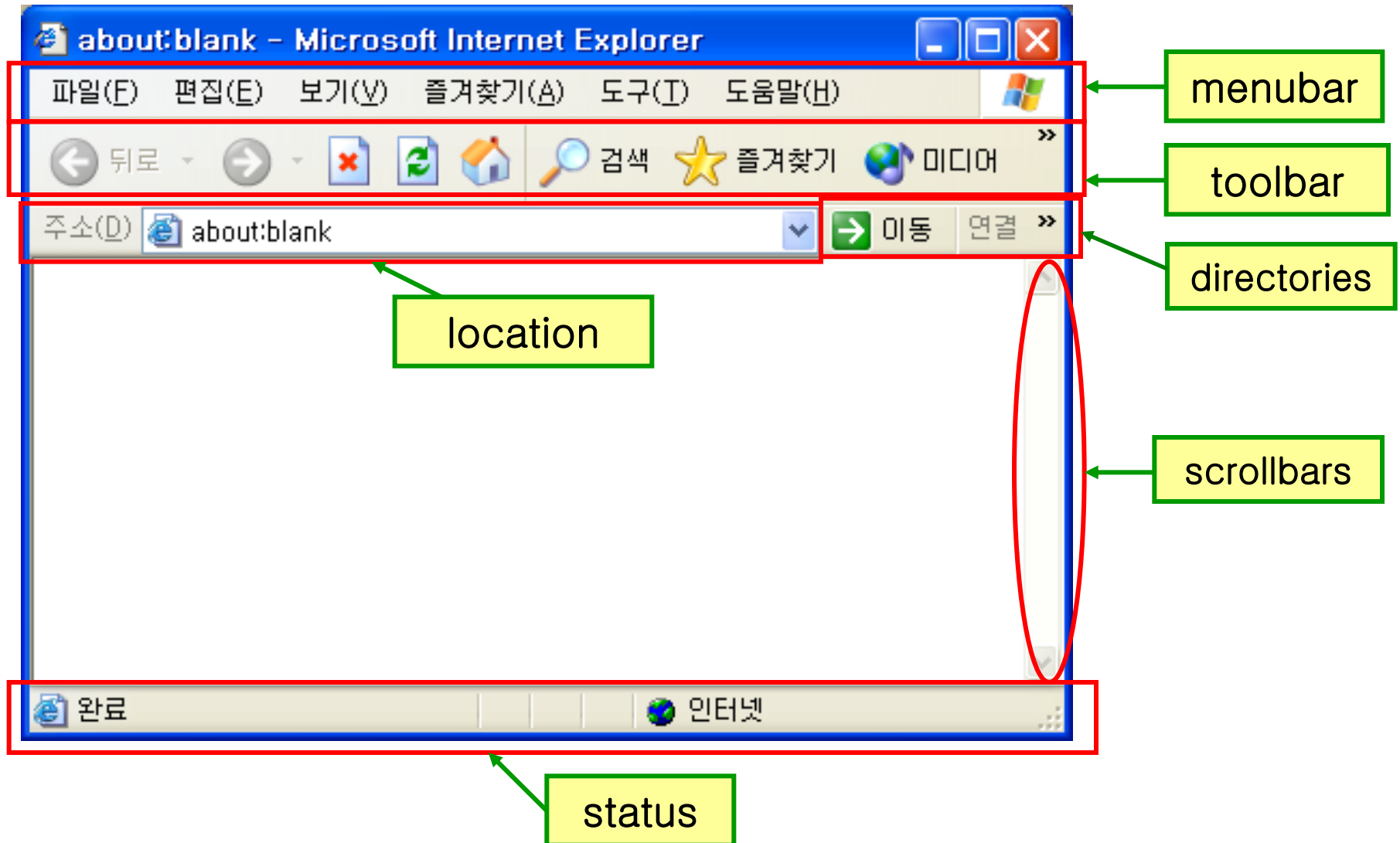


# ◆ Window 객체

open(“문서이름” , ”창이름” , ”윈도우속성문자열”)

속성	값	설 명
directories	yes, No  OR  1,0	디렉토리 메뉴
Location		주소(URL)
menubar		메뉴바
scrollbars		스크롤바
toolbar		툴바메뉴(뒤로,앞으로,중지등)
resizable		브라우저 크기조절
status		상태바
width	픽셀수	창의 너비설정
height		창의 높이 설정

# ◆ Window 객체



# ◆ Window 객체

## ◆ 실습

### Sample

- ◆ 창 의 모양.html
- ◆ resizeBy.html
- ◆ moveBy.html
- ◆ moveBy2.html
- ◆ remotemain.htm
- ◆ remote.htm

# ◆ History 객체

- ◆ 브라우저를 이용하여 본 URL 사이트를 임시로 저장하고 있음
- ◆ 뒤로(Back), 앞으로(Forward) 버튼을 이용하여 볼 수 있음
- ◆ 속성

속성	설명
length	브라우저로 읽어온 URL 주소의 개수를 알아낼 수 있음

## ◆ 메소드

Method	설명
back()	브라우저로 본 이전 화면으로 이동
forward()	브라우저로 본 다음 화면으로 이동
Go(숫자)	상대적인 숫자를 설정하여 본 화면을 이동

Sample

◆ history.html

# ◆ Location 객체

- ◆ 브라우저 창에 읽어온 문서를 제어할 수 있음
- ◆ 속성

속성	설명
hash	표식 이름을 설정하거나 알아냄
host	URL 주소의 호스트 이름과 포트를 설정하거나 알아냄
hostname	URL 주소의 호스트 이름, IP 주소를 설정하거나 알아냄
href	문서의 URL 주소를 설정하거나 알아냄
pathname	문서의 디렉토리 위치를 설정하거나 알아냄
port	포트 번호를 설정하거나 알아냄
protocol	프로토콜 종류를 설정하거나 알아냄
search	검색엔진을 호출할 때 사용

# ◆ Location 객체

Method	설명
reload()	현재 문서를 다시 읽어옴
replace()	현재 문서를 다른 URL 문서로 바꾸어줌

Sample

◆ location.html

# ◆ Navigator 객체

- ◆ 브라우저 객체 중에 독립적으로 사용됨
- ◆ 브라우저 종류, 사용언어, 시스템 종류 등을 알아낼 수 있음
- ◆ 속성

속 성	설 명
appName	브라우저의 코드명을 반환
appVersion	현재 사용중인 브라우저의 이름을 반환
language	현재 사용중인 브라우저의 버전을 반환
mimeTypes	현재 브라우저가 사용하는 언어를 반환
platform	mime 형식의 정보를 반환(익스플로러지원안함)
plugins	사용중인 시스템 코드를 반환
userAgent	플러그인 정보를 반환
	브라우저의 이름, 버전, 코드를 포함하는 문자열을 반환

# ◆Navigator 객체

Method	설명
javaEnabled()	현재 사용하는 브라우저의 자바 사용가능 여부
taintEnabled()	현재 브라우저가 문서의 정보를 정상적으로 읽어왔는지, 잘못되었는지를 알아냄(익스플로러지원안함)

Sample

◆Navigator.html



# ◆ Document 객체

속성	설명
linkColor	링크색을 설정함
alinkColor	누르고 있는 동안의 링크 문자열의 색을 설정
vlinkColor	방문한 적 있는 링크의 문자열의 색을 설정
bgColor	창의 배경색을 설정
fgColor	글자색을 설정함
images	이미지 배열 정보를 포함
lastModified	가장 최근에 수정한 문서 날짜를 알아냄
layers	레이어 배열 정보를 포함
cookie	쿠키를 사용할 수 있게 해줌
referrer	링크된 이전 문서의 URL정보를 알아냄
title	문서의 제목을 설정하거나 반환

# ◆ Document 객체

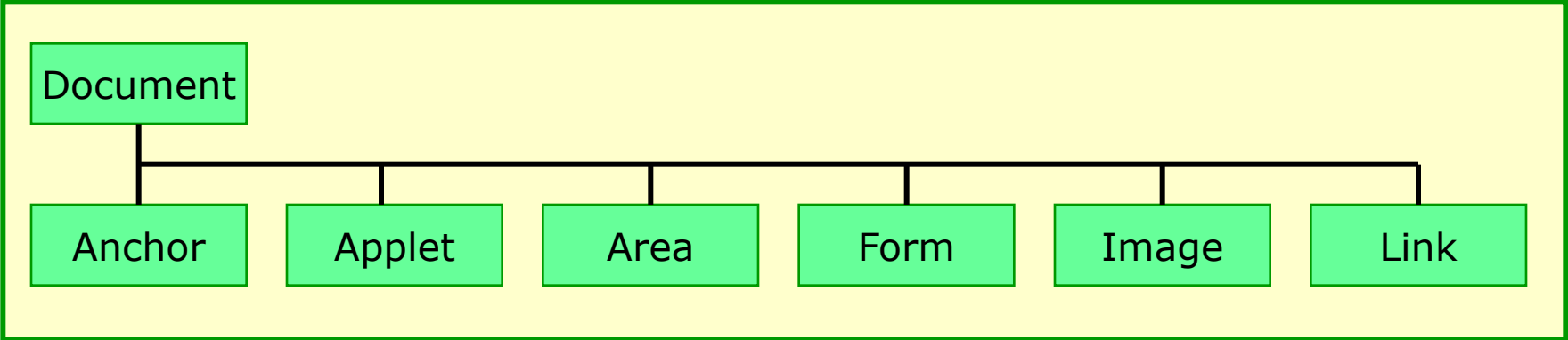
이벤트	설명
onBlur	브라우저에 포커스를 잃을 때 발생
onClick	마우스로 클릭했을 때 발생
onDbClick	마우스로 더블클릭했을 때 발생
onFocus	브라우저에 포커스를 얻을 때 발생
onKeyDown	키를 눌렀을 때 발생
onKeyPress	키를 눌렀다 놓았을 때 발생
onKeyUp	키를 놓았을 때 발생
onMouseDown	마우스를 눌렀을 때 발생
onMouseUp	마우스를 놓았을 때 발생

# ◆ Document 객체

Method	설 명
clear()	문서의 내용을 지움
getSelection()	마우스로 드래그한 문자열을 반환(익스플로우 지원안함)
write()	태그를 포함한 문자열을 문서에 출력

# ◆ Document 객체

## ◆ Document 객체에 포함된 객체



객 체	설 명
anchors	문서 안에 특정 부분에 대한 정보를 배열로 포함하고 있음
applets	자바 애플릿에 대한 정보를 배열로 포함하고 있음
forms	<FORM> 태그에 관한 정보를 배열로 포함하고 있음
images	이미지에 대한 정보를 배열로 포함하고 있음
links	링크에 관한 정보를 배열로 포함하고 있음
location	문서의 URL 위치를 알아냄

# ◆ Form 객체

- ◆ HTML 입력양식을 뜻하는 객체
- ◆ 속성

속 성	설 명
action	<FORM> 태그의 action 속성과 같음
elements	<FORM> 태그 안에 있는 양식을 배열로 저장
encoding	<FORM> 태그의 enctype 속성과 같음
length	<FORM> 태그 안에 있는 양식의 개수
name	<FORM> 태그의 name 속성과 같음
method	<FORM> 태그의 method 속성과 같음
target	<FORM> 태그의 target 속성과 같음

# ◆ Form 객체

## ◆ 메소드

Method	설 명
blur()	커서를 제거시킴
reset()	양식을 새로 다시 입력 받을 수 있게 초기화 시켜줌
submit()	입력된 양식 내용을 서버에 전송함

## ◆ 이벤트

이벤트	설 명
onReset	Reset 버튼을 누를 때 발생
onSubmit	Submit 버튼을 누를 때 발생

## ◆ 양식에서 공통적으로 쓸 수 있는 메소드

이벤트	설 명
toString()	객체를 문자열로 반환
valueOf()	객체를 원시값으로 반환

# ◆Form 객체

## ◆예제

### Sample

- ◆글자 옮기기.html
- ◆Radio.html
- ◆checkbox이용한창띄우기.html
- ◆select.html

# ◆Anchor 객체

- ◆ 링크된 문장을 특정 위치에 이동시킬 수 있음
- ◆ Anchor객체는 같은 문서 내에서 하이퍼 링크를 시킬때 사용되는 <a name> 태그에 의해 지정된 anchor에 관한 정보 및 속성을 추출하고 제어하고 설정할 수 있도록 하는 객체
- ◆ 속성

속성	설 명
length	anchor객체를 몇 번 사용했는지를 알아냄
name	anchor객체의 이름값을 알아냄

Sample

◆anchor.html



# ◆ Link 객체

- ◆ 다른 곳이나 URL 위치로 이동할 수 있게 해줌
- ◆ link객체는 <a href>태그에 의해 지정된 하이퍼 링크에 관한 정보 및 속성을 추출하고 제어하고 설정 할 수 있도록 하는 객체이다.
- ◆ 속성

속 성	설 명
hash	Href에 지정된 URL의 앵커 이름을 설정하거나 알아낸다.
host	URL 주소의 호스트 이름과 포트를 설정하거나 알아냄
hostname	URL 주소의 호스트 이름, IP 주소를 설정하거나 알아냄
href	문서의 URL 주소를 설정하거나 알아냄
pathname	문서의 디렉토리 위치를 설정하거나 알아냄

# ◆ Link 객체

## ◆ 속성

속 성	설 명
port	포트 번호를 설정하거나 알아냄
protocol	프로토콜 종류를 설정하거나 알아냄
search	Href에 지정된 URL의 ?로 시작하는 질의 정보를 설정하거나 알아낸다.
target	지정된 URL에 연결된 후 보여줄 창을 알아냄

Sample

◆ link.html

# ◆ Image 객체

- ◆ HTML 문서 안에 있는 이미지를 처리할 수 있음
- ◆ Image객체는 <img>태그를 이용해 만들어진 이미지에 관한 정보 및 속성을 추출하고 제어하고 설정 할 수 있도록 하는 객체로서 화면에 보이고 있는 이미지의 이름, 크기, 테두리굵기, 이미지 전송 시 에러발생여부 등 이미지와 관련한 다양한 정보를 처리
- ◆ 속성

속 성	설 명
align	이미지 정렬 방식을 알아냄
alt	이미지 설명을 알아냄
border	이미지의 테두리 값을 알아냄
complete	이미지 전송이 끝났는지 여부를 알아냄
height	이미지의 높이를 알아냄
width	이미지의 너비를 알아냄

# ◆ Image 객체

## ◆ 속성

속 성	설 명
lowsrc	이미지 전송하는 동안에 보여줄 이미지 파일의 위치
name	이미지 이름을 알아냄
prototype	이미지 객체에 속성을 추가하기 위해서 사용함
src	이미지 URL주소 알아냄
vspace	이미지의 세로 여백값을 알아냄
hspace	이미지의 가로 여백값을 알아냄

### Src속성과 lowsrc속성의 차이점

해상도가 비교적 높은 이미지인 경우에는 이미지를 웹 문서에서 로드 할 때 시간이 많이 소요되는데 이런 경우 이미지가 완전히 로드 될 때 까지 해상도가 낮은 그림을 하나 더 준비하여 임시적으로 보여주는 그림을 lowsrc속성으로 이용

# ◆ Image 객체

이벤트	설 명
onAbort	이미지 전송을 중지시켰을 때 발생
onError	이미지 전송중에 에러가 나타날 때 발생
onLoad	창에 이미지가 나타날 때 발생
onKeyDown	키를 눌렀을 때 발생
onKeyPress	키를 눌렀다 놓았을 때 발생
onKeyUp	키를 놓았을 때 발생

## Sample

◆ image속성.html

◆ changeImage.html

# ◆Frame객체

- ◆ 프레임은 하나의 웹 브라우저 창에 하나 이상의 HTML 문서를 볼 수 있도록 해주는 기능이다.
- ◆ 프레임을 사용하기 위해서는 **<FRAMESET>** 태그와 **<FRAME>** 태그를 사용한다.
- ◆ 프레임은 HTML문서를 가지고 있기 때문에 프레임에 document 객체가 포함된다.
- ◆ 프레임을 사용하는 경우에 한 프레임에서 다른 프레임을 접근하기 위해서는 상위 프레임이나 창으로 이동한 다음에 원하는 프레임을 기술한다.
- ◆ **parent** : 자신의 상위 객체
- ◆ **top** : 계층구조상 top레벨의 윈도우 객체

# ◆Frame객체

```
<frameset cols="30% , *">  
    <frame src="left.htm" name="left">  
    <frame src="right.htm" name="right">  
</frameset>
```

```
<FORM method="post" target="right" >  
    <INPUT type="button" name="yahoo" value="YAHOO">  
</Form>
```

## Sample

- ◆frame.htm
- ◆frame1.htm
- ◆frame2.htm

# ◆ Layer

- ◆ 레이어를 이용하면 이미지나 텍스트와 같은 객체들의 절대위치를 지정할 수 있다.
- ◆ HTML 페이지상에서 이러한 객체들을 이동시킬 수 도 있으며 그 객체들을 상황에 따라 숨기거나 보여줄 수 도 있다.

## ◆ 레이어 접근하기

```
<DIV ID= "idname" style="position: absolute; left: 50; top: 100  
width: 30;" >
```

레이어테스트

```
</DIV>
```

## ◆ 사용법

```
document.all.idname.style.속성  
= document.all["idname"].style.속성  
= idname.style.속성
```



# ◆ Layer

## ◆ 자바스크립트 레이어 속성

속성	설 명
background	레이어 배경색. 값(색 이름 혹은 색 코드)
Left / top	레이어 왼쪽/위로부터의 위치 값(픽셀)
right / bottom	레이어 오른쪽/아래로 부터의 위치. 값(픽셀)
width / height	레이어의 넓이/높이. 값(픽셀)
visibility	레이어를 보이거나 숨김(hidden, visible)
zindex	겹친 레이어들의 순서. 값(숫자 1 ~)

### Sample

- ◆레이어이벤트.htm
- ◆마우스따라다니기.htm
- ◆레이어움직이기.html

# ◆ 테스트

## [ 기능 ]

아래와 같은 오류를 테스트하여 체크하여 보자.

1. 입력 유무 체크.
2. 두개의 암호 일치 여부.
3. 주민번호 자리 수 체크하기.
4. 주민번호 두번째 자리  
커서자동이동.
5. 성별 자동으로 체크 하여 주기.
6. Email안에 @문자 유무 체크.
7. 자기소개 클릭하면 박스 안에  
문자 자동으로 clear.

New Document - Microsoft Internet Explorer

파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)

주소(D) E:\자바스크립트\강북파워포인트\파워6소스\layer\회원가입.htm 이동 연결 >>

### 회원가입

이름	<input type="text"/>
사용자 id	<input type="text"/> <input type="button" value="아이디중복체크"/>
암호	<input type="password"/>
암호확인	<input type="password"/>
주민등록번호	<input type="text"/> <input type="text"/>
성별	남자 <input type="radio"/> 여자 <input type="radio"/>
직업	선택하세요 ▼
email	<input type="text"/>
자기소개	<div>간단하게 입력하세요</div> <div></div>

완료 내 컴퓨터

ES6(ECMAScript 6) and JS

# ES5

- ES5에서 변수를 선언할 수 있는 유일한 방법은 var 키워드를 사용하는 것.
- . var 키워드로 선언된 변수의 특징
  - 함수 레벨 스코프(Function-level scope)
    - 전역 변수의 남발
    - for loop 초기화식에서 사용한 변수를 for loop 외부 또는 전역에서 참조할 수 있다.
  - var 키워드 생략 허용
    - 의도하지 않은 변수의 전역화
  - 중복 선언 허용
    - 의도하지 않은 변수값 변경
  - 변수 호이스팅
    - 변수를 선언하기 전에 참조가 가능하다

# ES5

- 대부분의 문제는 전역 변수로 인해 발생한다. 전역 변수는 간단한 애플리케이션의 경우, 사용이 편리하다는 장점이 있지만 불가피한 상황을 제외하고 사용을 억제해야 한다. 전역 변수는 유효 범위(scope)가 넓어서 어디에서 어떻게 사용될 것인지 파악하기 힘들며, 비순수 함수(Impure function)에 의해 의도하지 않게 변경될 수도 있어서 복잡성을 증가시키는 원인이 된다. 따라서 변수의 유효 범위(scope)는 좁을수록 좋다.
- ES6는 이러한 var의 단점을 보완하기 위해 let과 const 키워드를 도입하였다.

# ES6의 let

- **1.1 블록 레벨 스코프**
- 대부분의 C-family 언어는 블록 레벨 스코프(Block-level scope)를 지원하지만 자바스크립트는 함수 레벨 스코프(Function-level scope)를 갖는다.

# ES6의 let

- 함수 레벨 스코프(Function-level scope)
  - 함수 내에서 선언된 변수는 함수 내에서만 유효하며 함수 외부에서는 참조할 수 없다. 즉, 함수 내부에서 선언한 변수는 지역 변수이며 함수 외부에서 선언한 변수는 모두 전역 변수이다.
- 블록 레벨 스코프(Block-level scope)
  - 코드 블록 내에서 선언된 변수는 코드 블록 내에서만 유효하며 코드 블록 외부에서는 참조할 수 없다.

# ES6의 let

- 블록 레벨 스코프를 지원하지 않는 `var` 키워드의 특성상, 코드 블록 내의 변수 `foo`는 전역변수이다. 그런데 이미 전역변수 `foo`가 선언되어 있다. `var` 키워드를 사용하여 선언한 변수는 중복 선언이 허용되므로 위의 코드는 문법적으로 아무런 문제가 없다. 단, 코드 블록 내의 변수 `foo`는 전역변수이기 때문에 전역에서 선언된 전역변수 `foo`의 값 123을 대체하는 새로운 값 456을 재할당한다..

```
console.log(foo); // undefined
var foo = 123;
console.log(foo); // 123
{
  var foo = 456;
}
console.log(foo); // 456
```



# ES6의 let

- ES6는 **블록 레벨 스코프**를 갖는 변수를 선언하기 위해 let 키워드를 제공한다.
- let 키워드로 선언된 변수는 블록 레벨 스코프를 갖는다. 위 예제에서 코드 블록 내에 선언된 변수 foo는 블록 레벨 스코프를 갖는 지역 변수이다. 전역에서 선언된 변수 foo와는 다른 변수이다. 또한 변수 bar도 블록 레벨 스코프를 갖는 지역 변수이다. 따라서 전역에서는 변수 bar를 참조할 수 없다.

```
let foo = 123;  
{  
  let foo = 456;  
  let bar = 456;  
}  
console.log(foo); // 123  
console.log(bar); // ReferenceError: bar is not defined
```

# ES6의 let

- var 키워드로는 이름이 같은 변수를 중복해서 선언할 수 있었지만, let 키워드로는 이름이 같은 변수를 중복해서 선언하면 문법 에러 (SyntaxError)가 발생한다.

```
var foo = 123;
```

```
var foo = 456; // 중복 선언 허용
```

```
let bar = 123;
```

```
let bar = 456; // Uncaught SyntaxError: Identifier 'bar' has already been declared
```

# 호이스팅(Hoisting)

- 자바스크립트는 ES6에서 도입된 let, const를 포함하여 모든 선언(var, let, const, function, function\*, class)을 호이스팅한다.
- 호이스팅(Hoisting)이란, var 선언문이나 function 선언문 등을 해당 스코프의 선두로 옮긴 것처럼 동작하는 특성을 말한다.
- 하지만 var 키워드로 선언된 변수와는 달리 let 키워드로 선언된 변수를 선언문 이전에 참조하면 참조 에러(ReferenceError)가 발생한다. 이는 let 키워드로 선언된 변수는 스코프의 시작에서 변수의 선언까지 **일시적 사각지대(Temporal Dead Zone; TDZ)**에 빠지기 때문이다.

# 호이스팅(Hoisting)

```
console.log(foo); // undefined
var foo;

console.log(bar); // Error: Uncaught ReferenceError: bar is not defined
let bar;
```

```
let foo = 1; // 전역 변수

{
  console.log(foo); // ReferenceError: foo is not defined
  let foo = 2; // 지역 변수
}
```

# 전역객체와 let

- 전역 객체(Global Object)는 모든 객체의 유일한 최상위 객체를 의미하며 일반적으로 Browser-side에서는 window 객체
- var 키워드로 선언된 변수를 전역 변수로 사용하면 전역 객체의 프로퍼티가 된다.

```
var foo = 123; // 전역변수
```

```
console.log(window.foo); // 123
```

# 전역객체와 let

- let 키워드로 선언된 변수를 전역 변수로 사용하는 경우, let 전역 변수는 전역 객체의 프로퍼티가 아니다. 즉, window.foo와 같이 접근할 수 없다. let 전역 변수는 보이지 않는 개념적인 블록 내에 존재하게 된다.

```
let foo = 123; // 전역변수
```

```
console.log(window.foo); // undefined
```

# const

- const는 상수(변하지 않는 값)
- let은 재 할당 가능하나 const 재할당 금지

```
const F00 = 123;
```

```
F00 = 456; // TypeError: Assignment to constant variable.
```

# const

- 주의할 점은 **const**는 반드시 선언과 동시에 할당이 이루어져야 한다는 것이다. 그렇지 않으면 다음처럼 문법 에러(SyntaxError)가 발생한다.

```
const F00; // SyntaxError: Missing initializer in const declaration
```



# const

- const는 let과 마찬가지로 블록 레벨 스코프를 갖는다

```
{  
  const F00 = 10;  
  console.log(F00); //10  
}  
console.log(F00); // ReferenceError: F00 is not defined
```

# const

- 상수는 가독성과 유지보수의 편의를 위해 적극적으로 사용해야 한다

```
// 10의 의미를 알기 어렵기 때문에 가독성이 좋지 않다.  
if (rows > 10) {  
}
```

```
// 값의 의미를 명확히 기술하여 가독성이 향상되었다.  
const MAXROWS = 10;  
if (rows > MAXROWS) {  
}
```

# 7.1 생성자 함수 개요

## ❖ 생성자 함수란?

- new 키워드를 사용해 객체 생성할 수 있는 함수

## ❖ student 생성자

- student 생성자 함수를 만드는 코드

코드 7-1 생성자 함수 생성

```
<script>  
    function Student() {  
  
    }  
</script>
```



# 7.1 생성자 함수 개요

## ❖ new 키워드

- new 키워드로 객체 생성

코드 7-2 객체 생성

```
<script>
    function Student() {

    }

    var student = new Student();
</script>
```



# 7.1 생성자 함수 개요

## ❖ this 키워드

- 생성자 함수로 생성될 객체의 속성 지정

코드 7-3 속성 생성

```
<script>
function Student(name, korean, math, english, science) {
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;
}
var student = new Student('윤하린', 96, 98, 92, 98);
</script>
```



## 7. 생성자 함수 개요

### ❖ 메서드 생성

코드 7-4 메서드 생성

```
<script>
function Student(name, korean, math, english, science) {
    // 속성
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;

    // 메서드
    this.getSum = function () {
        return this.국어 + this.수학 + this.영어 + this.과학;
    };
    this.getAverage = function () {
        return this.getSum() / 4;
    };
    this.toString = function () {
        return this.이름 + '\t' + this.getSum() + '\t' + this.getAverage();
    };
}

var student = new Student('윤하린', 96, 98, 92, 98);
</script>
```



## 7. 생성자 함수 개요

### ❖ 생성자 함수를 사용한 객체 배열 생성

코드 7-5 생성자 함수를 사용한 객체 배열 생성

```
<script>
    function Student(name, korean, math, english, science) {
        /* 생략 */
    }

    // 학생 정보 배열을 만듭니다.
    var students = [];
    students.push(new Student('윤하린', 96, 98, 92, 98));
    /* 생략 */
    students.push(new Student('윤인아', 96, 96, 98, 92));

    // 출력합니다.
    var output = '이름\t총점\t평균\n';
    for (var i in students) {
        output += students[i].toString() + '\n';
    }
    alert(output);
</script>
```



## 7.2 프로토타입

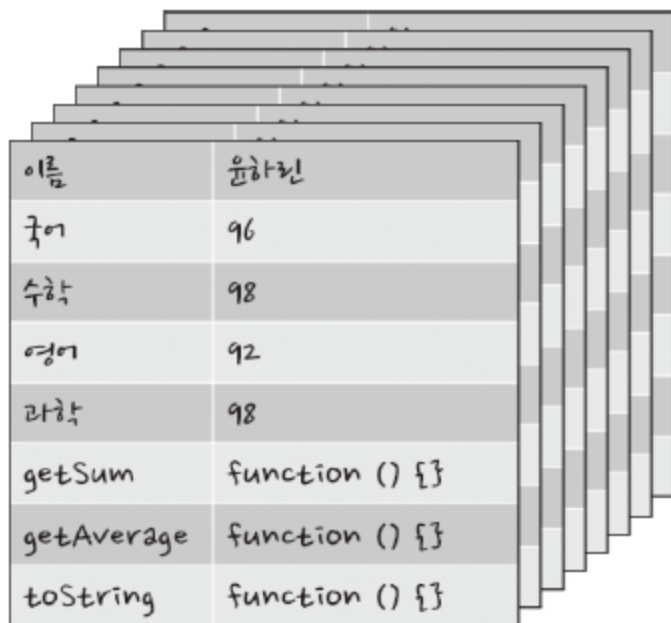
### ❖ 생성자 함수

#### ■ 기존의 객체 구조

→ 이름, 국어, 수학, 영어, 과학 속성

→ getSum ( ), getAverage ( ), toString ( ) 메서드

그림 7-3 기존의 객체 구조



이름	윤하린
국어	96
수학	98
영어	92
과학	98
getSum	function () {}
getAverage	function () {}
toString	function () {}

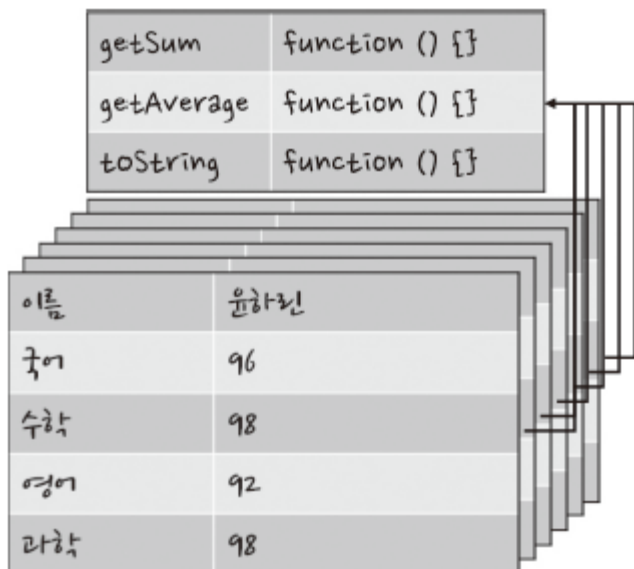




## ■ 프로토타입

→ 생성자 함수로 생성된 객체가 공통으로 가지는 공간

그림 7-4 프로토타입을 사용한 객체 구조



## 7.2 프로토타입

### ❖ 생성자 함수 구성

- 한 개의 메서드로 모든 객체가 사용
- 생성자 함수로 객체를 만들 때 → 생성자 함수 내부에 속성만 넣음

코드 7-7 생성자 함수 구성

```
<script>
  function Student(name, korean, math, english, science) {
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;
  }
</script>
```

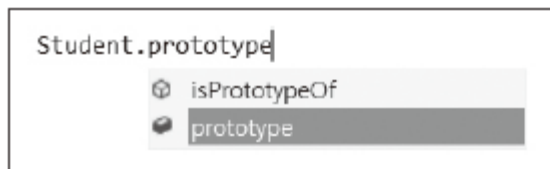


## 7.2 프로토타입

### ❖ 프로토타입

- 자바스크립트의 모든 함수는 변수 prototype을 갖음
- prototype은 객체

그림 7-5 모든 자바스크립트의 함수는 prototype 객체를 갖습니다

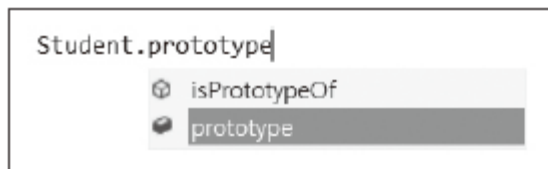


## 7.2 프로토타입

### ❖ 프로토타입

- 자바스크립트의 모든 함수는 변수 prototype을 갖음
- prototype은 객체

그림 7-5 모든 자바스크립트의 함수는 prototype 객체를 갖습니다



## 7.3 new 키워드

### ❖ new 키워드

코드 7-9 new 키워드

```
<script>
  // 생성자 함수를 선언합니다.
  function Constructor(value) {
    this.value = value;
  }

  // 변수를 선언합니다.
  var constructor = new Constructor('Hello');

  // 출력합니다.
  alert(constructor.value);
</script>
```



## 7.3 new 키워드

### ❖ new 키워드를 사용하지 않으면?

- this 키워드 사용 → window 객체를 나타냄
- 일반 함수 호출과 같이 new 키워드를 사용하지 않으면  
→ 함수 실행 중 window 객체에 속성이 추가한 것이 됨
- new 키워드로 함수 호출  
→ 객체를 위한 공간 생성(this 키워드가 해당 공간을 의미)



### ❖ 캡슐화

- 다양한 사람이 많아서 만들어진 기술
- 예제 : Rectangle 객체를 만들어 봄

코드 7-11 생성자 함수 Rectangle 선언

```
<script>
  // 생성자 함수를 선언합니다.
  function Rectangle(width, height) {
    this.width = width;
    this.height = height;
  }
  Rectangle.prototype.getArea = function () {
    return this.width * this.height;
  };

  // 변수를 선언합니다.
  var rectangle = new Rectangle(5, 7);

  // 출력합니다.
  alert('AREA: ' + rectangle.getArea());
</script>
```



### ❖ 캡슐화

#### ■ 잘못된 속성의 사용

코드 7-12 잘못된 속성의 사용

```
// 변수를 선언합니다.  
var rectangle = new Rectangle(5, 7);  
rectangle.width = -2;  
  
// 출력합니다.  
alert('AREA: ' + rectangle.getArea());
```





### ❖ 캡슐화

- 캡슐화는 잘못 사용될 수 있는 객체의 특정 부분을 사용자가 사용할 수 없게 막는 기술

코드 7-13 캡슐화

```
// 생성자 함수를 선언합니다.
function Rectangle(w, h) {
    // 변수를 선언합니다.
    var width = w;
    var height = h;

    // 메서드를 선언합니다.
    this.getWidth = function () { return width; };
    this.getHeight = function () { return height; };
    this.setWidth = function (w) {
        width = w;
    };
    this.setHeight = function (h) {
        height = h;
    };
}
```



### ❖ 캡슐화

- 게터 : get○○ ( ) 형태의 메서드와 같이 값을 가져오는 메서드
- 세터 : set○○ ( ) 형태의 메서드와 같이 값을 입력하는 메서드
- 게터와 세터를 만드는것이 캡슐화는 아님
- 캡슐화는 만일의 상황에 특정 속성이나 메서드를 사용자가 사용할 수 없도록 숨겨 놓는 것임

