HTML & CSS3

고 덕 한

과정 개요

- HTML 에 대한 기초 지식 습득
- HTML 구성 요소 파악
- CSS 이해 및 활용
- JavaScript 개요 및 사용법 파악
- DHTML을 적용한 페이지 구성

HTML Structure

1. HTML 이해
2. CSS (Cascading Style Sheet)
3. JavaScript
4. 브라우저 내장객체
5. 폼(FORM) 관련 객체
6. 레이어(Layer)와 DHTML

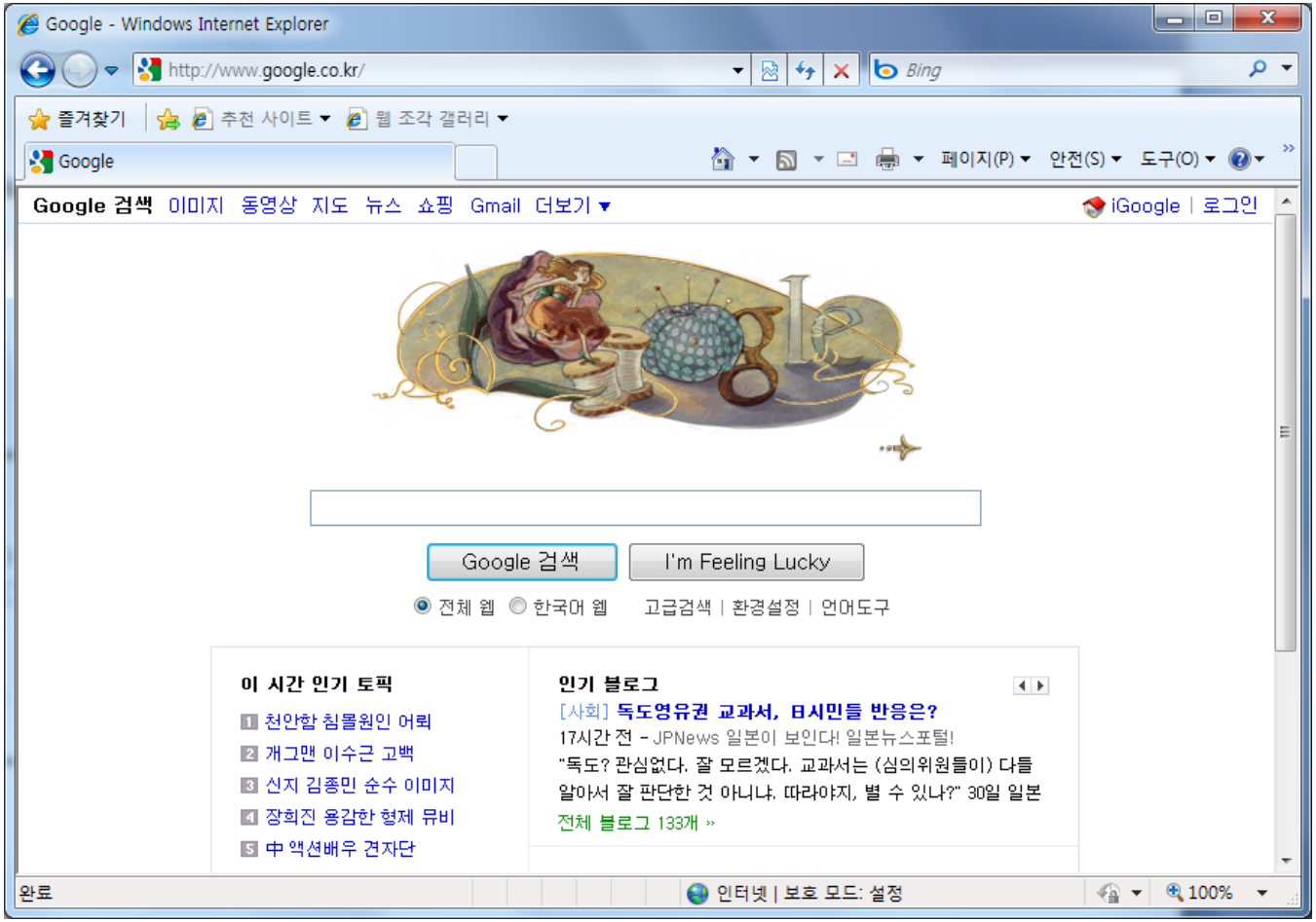
HTML 개요

◆ HTML

- Hyper Text Markup Language의 약어

◆ HTML 의 기본 구조

```
<HTML>
  <HEAD>
    <TITLE>
    </TITLE>
  </HEAD>
  <BODY>
  </BODY>
</HTML>
```



HTML 개요

◆ 태그의 역할

- **<HTML> ... </HTML>**
  - 문서의 시작과 끝
- **<HEAD> ... </HEAD>**
  - 문서 전체의 정보를 제공
  - <TITLE>, <META>, <SCRIPT>. <STYLE> 태그 포함 가능
- **<META> ... </META>**
  - 문서 정보 및 사이트 이동, 화면 전환 효과, 웹 페이지 검색 기능
- **<TITLE> ... </TITLE>**
  - 브라우저의 제목 표시줄에 나타날 문서 제목 표시
- **<BODY> ... </BODY>**
  - 웹 페이지 몸체부의 시작과 끝을 나타내는 문서

HTML 개요

◆ 태그의 역할

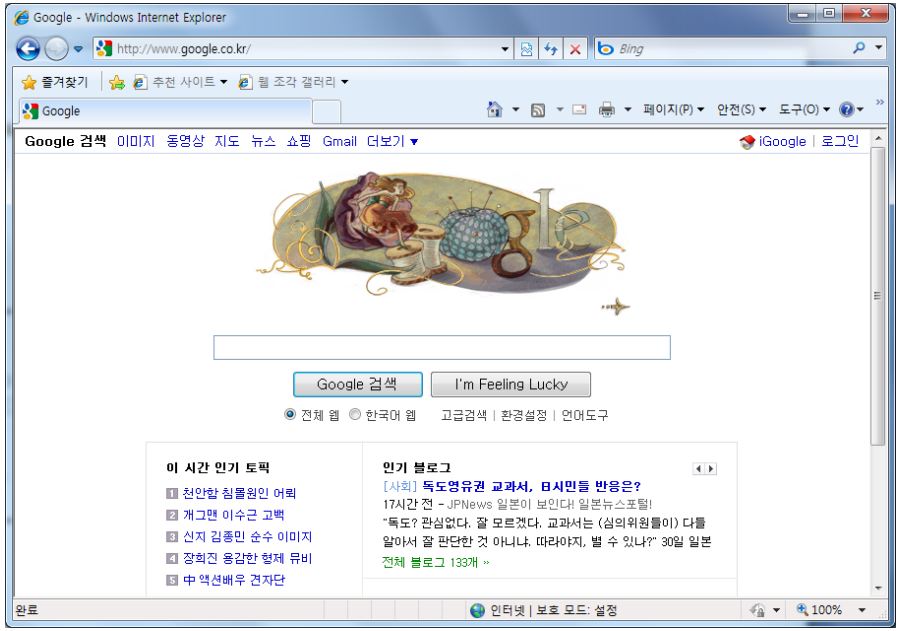
- **<HTML> ... </HTML>**
  - 문서의 시작과 끝
- **<HEAD> ... </HEAD>**
  - 문서 전체의 정보를 제공
  - <TITLE>, <META>, <SCRIPT>. <STYLE> 태그 포함 가능
- **<META> ... </META>**
  - 문서 정보 및 사이트 이동, 화면 전환 효과, 웹 페이지 검색 기능
- **<TITLE> ... </TITLE>**
  - 브라우저의 제목 표시줄에 나타날 문서 제목 표시
- **<BODY> ... </BODY>**
  - 웹 페이지 몸체부의 시작과 끝을 나타내는 문서

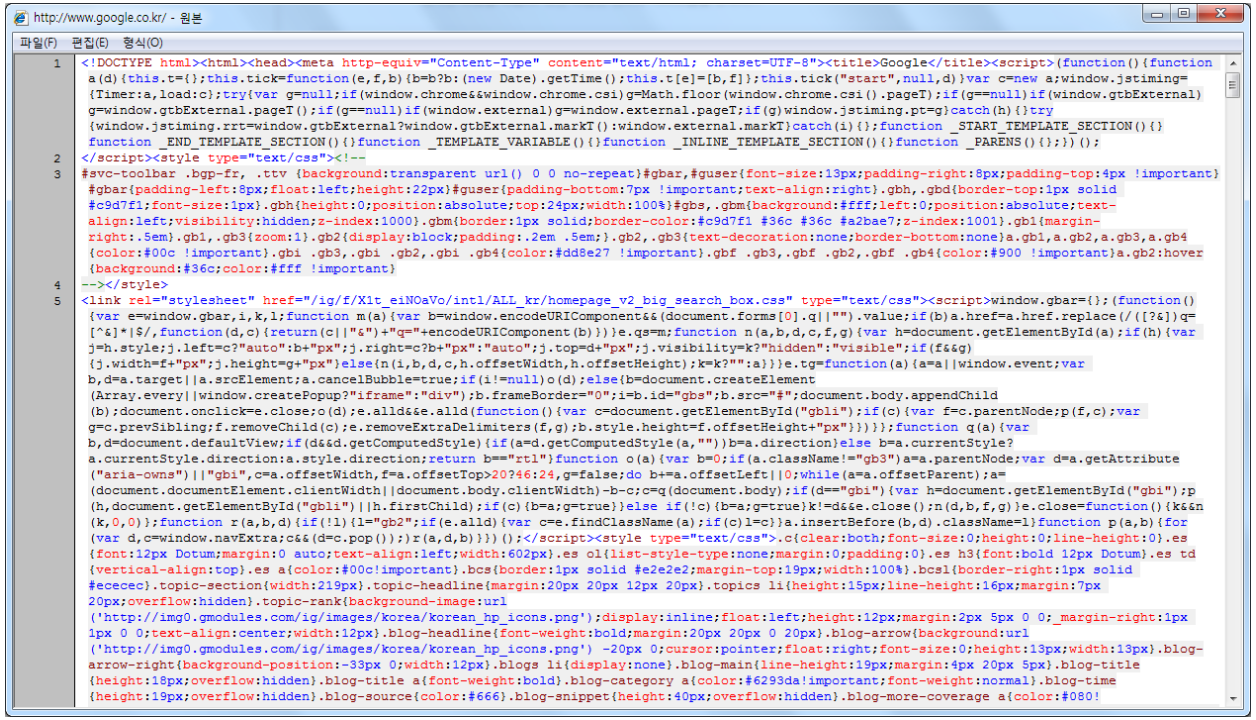
HTML 개요

◆ 태그의 역할

● HTML 소스와 화면

– HTML 에서 태그와 화면에 보이도록 처리하는 부분





문서관련 기본 태그

◆ 배경 관련 태그

● <BODY> 태그의 Element 로 배경을 처리

기능	속성	속성값	사용 예
배경색 지정	bgcolor	색상	<body bgcolor ="red">
배경 이미지 지정	background	이미지 파일명	<body background="이미지파일명">
배경 글자색 지정	text	색상	<body text="white">
하이퍼링크 글자색 지정	link	색상	<body link="red">
클릭될 하이퍼링크 글자색 지정	alink	색상	<body link="blue">
클릭 후 하이퍼링크 글자색 지정	vlink	색상	<body vlink="black">
문서의 윗 여백 지정	topmargin	픽셀	<body topmargin="100">
문서의 왼쪽 여백 지정	leftmargin	픽셀	<body leftmargin="50">
배경 이미지 고정	bgproperties	fixed	<body bgproperties="fixed">
스크롤 바 비활성화	scroll	no	<body scroll="no">

문서관련 기본 태그

◆ 글꼴 관련 태그

● <FONT>태그의 속성과 속성값

기능	속성	속성값	사용 예
글자색 지정	color	색상	<font color =“red”>
글꼴 지정	face	글자체	<font face = “굴림체”>
글자 크기 지정	size	숫자	<font size = “2”>

● 글꼴 설정 태그

기능	태그	기능	태그	기능	태그
굵은 글자체	<b>	최소선 표시	<strike>	아래첨자	<sub>
이탤릭체	<i>	타이핑체	<tt>	한단계 크게	<big>
밑줄 표시	<u>	윗첨자	<sup>	한단계 작게	<small>

문서관련 기본 태그

◆ 글꼴 관련 태그

● 글꼴 설정 태그(계속)

기능	태그	기능	태그
강조할 부분 이탤릭 체로 표시		키보드 입력 표시 시 고정폭 글자체	<kbd>
강조할 부분 굵은 문자로 표시		변수이름 표기 시 이탤릭체로 표시	<var>
코딩 시 고정 폭 글자체로 표시	<code>	인용구를 이탤릭체로 표시	<cite>
샘플 출력 시 고정폭 글자체로 표시	<samp>	이탤릭체로 정의 표시	<dfn>

● 특수문자

표기법	설명	표기법	설명
&lt;	여는 괄호	&copy;	저작권 표시
&gt;	닫는 괄호	&trade;	상표 표시
	공백 문자	&reg;	등록상표 표시
&amp;	& 기호 표기	&#163;	프랑 표시
&quot;	쌍따옴표	&#165;	엔 표시

문서관련 기본 태그

◆ 문단 관련 태그

기능	속성	속성값	사용 예
줄바꿈	 		내용
문단 구분 및 줄바꿈		align	내용 // 내용 후 줄바꿈 내용 // 내용을 중앙 정렬
문단 구분		align	내용
제목 출력	<h숫자>	align	<h2 align=center>내용</h2> //h2 크기로 중앙 정렬해서 내용 출력 숫자는 1~6까지 사용 가능
중앙 정렬			내용
수평선	<hr>	align size width color Noshade	<hr size=굵기 width=길이 align=정렬방식 color=색상 Noshade>
키보드로 입력한 형식 유지	<pre>		<pre>내용</pre> //내용은 그대로 출력되므로 공백을 위한 특수 문자는 사용하지 않음

문서관련 기본 태그

◆ 문단 관련 태그(계속)

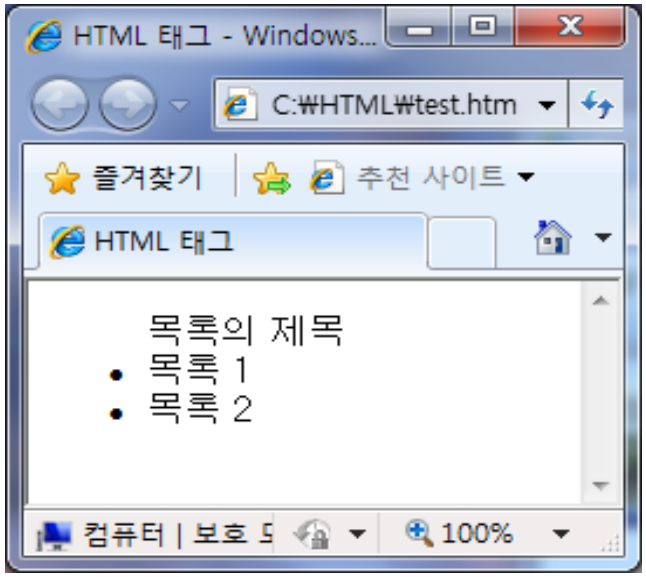
기능	속성	속성값	사용 예
태그 그대로 출력	<xmp>		<xmp>내용</xmp> //내용에 있는 태그 그대로 출력되므로 html 태그 강의록 작성 시 유리함
주석문	<!-- -->		<!--설명문 -->
문자열 흘러가기	marquee	width height behavior direction loop scrolldelay scrollmount bgcolor	<marquee width=너비 height=높이 behavior=scroll/slide/alternate direction=left/right/up/down loop=횟수 scrolldelay=시간 간격 scrollamount=움직이는 거리 bgcolor=배경색>내용</marquee>

문서관련 기본 태그

◆ 목록 관련 태그

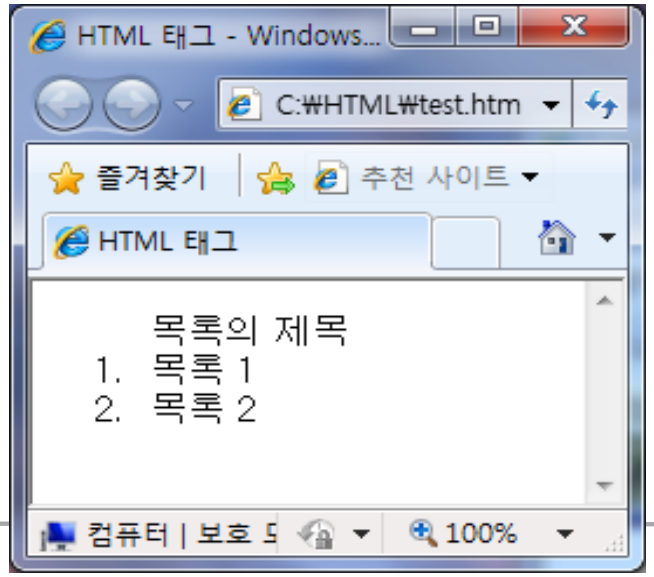
● 비순서 목록

```
<ul type = “서블릿 종류” >
    <lh>목록의 제목
    <li>목록 1
    <li>목록 2
</ul>
```



● 순서 목록

```
<ol type = “서블릿 종류” >
    <lh>목록의 제목
    <li>목록 1
    <li>목록 2
</ol>
```



문서관련 기본 태그

◆ 목록 관련 태그

● 목록 태그 속성

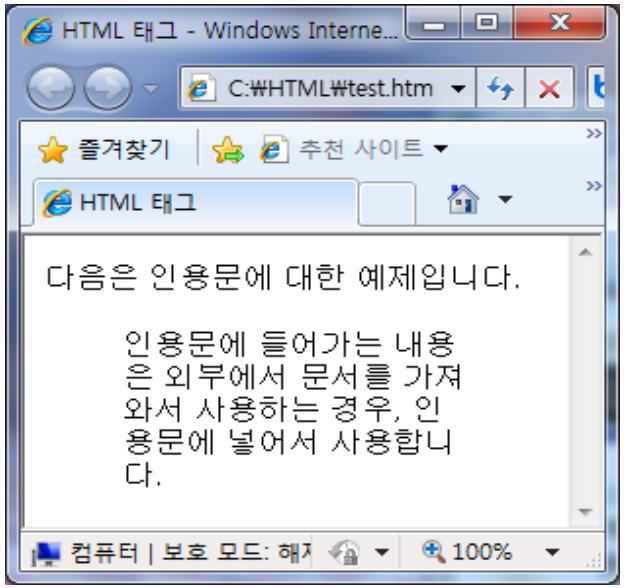
기능	속성	사용 예
시작 번호 지정	start	<ol start=10> //10부터 번호 부여
해당 목록의 시작 번호 지정	value	<li value=10> //10부터 목록 번호를 부여
목록 번호 스타일 지정	type	<ol type=a> //소문자로 번호를 부여 <li type=i> //로마자 소문자로 해당목록의 번호를 부여

● 인용문

```
<blockquote>

인용문

</blockquote>
```



문서관련 기본 태그

◆ 하이퍼링크(<a href>) 관련 태그

- 동일 도메인 내의 다른 웹 페이지 연결

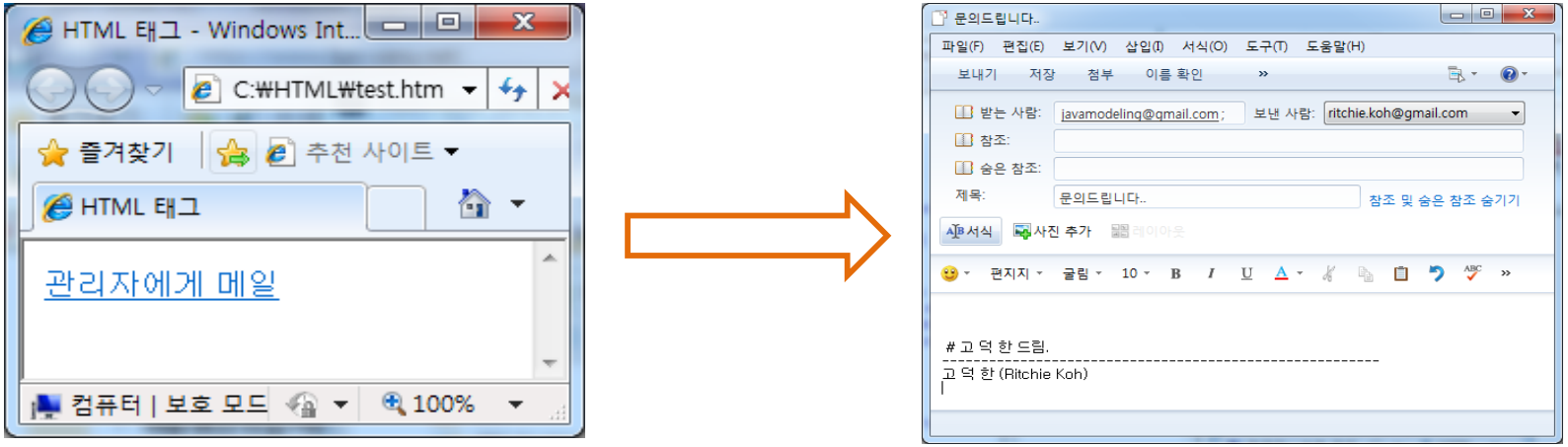
```
<a href="연결할 파일명" title="설명" target="창이름"> 화면에 표시될 글자 </a>
```

- 다른 도메인 내의 다른 웹 페이지 연결

```
<a href="전체 URL" title="설명" target="창이름"> 화면에 표시될 글자 </a>
```

- mail 주소와 연결

```
<a href="mailto:이메일주소" title="설명" target="창이름"> 화면에 표시될 글자 </a>
```



문서관련 기본 태그

◆ 이미지(<img>) 관련 태그

● 이미지 태그

```
<img src="그림파일명" alt="설명문" width="가로"height="세로"
 border="테두리"  align="정렬방식" vspace="상하여백"
 hspace="좌우여백">
```

● 이미지맵 태그

– 하나의 이미지에 여러 개의 링크를 설정

– <map> 태그 사용

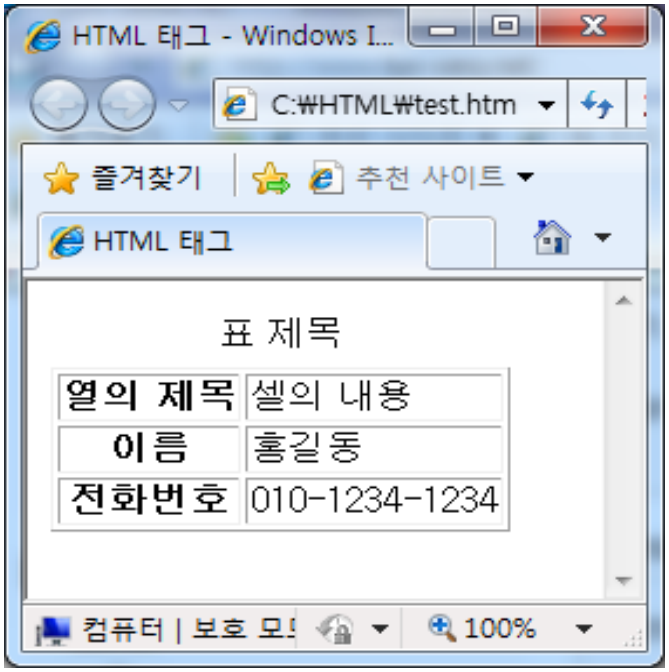
```
<img src="그림파일명" usemap="#맵이름">
<map name="이름">
<area shape="맵의 종류" coords="좌표값" href="링크될 주소"
 alt="설명문">
</map>
```

테이블(<table>) 관련 태그

◆ 테이블의 속성

● 테이블 태그의 형식

```
<table>
  <caption> 표 제목 </caption>
   <tr> --> 행 구분
    <th>열의 제목 </th>
    <td>셀의 내용 </td>
   </tr>
</table>
```



테이블(<table>) 관련 태그

◆ 테이블의 속성

● 테이블 태그 속성

기능	속성	사용 예
전체 너비 지정	width	<table width="600"> //테이블 전체 너비를 600픽셀로 지정
전체 높이 지정	height	<table height="600"> //테이블 전체 높이를 600픽셀로 지정
배경색 지정	bgcolor	<table bgcolor="red"> //테이블 전체 색상을 빨간색으로 지정
배경 이미지 지정	background	<table background="bg.gif"> //테이블 전체의 이미지 지정
정렬 방식 지정	align	<table align="center"> //테이블 전체의 정렬 방식을 중앙으로 지정
셀 안의 여백 지정	cellpadding	<table cellpadding="5"> //셀 안의 내용과 경계선 사이 여백을 5픽셀로 지정
셀 간의 여백 지정	cellspacing	<table cellspacing="5"> //셀 간의 여백을 5픽셀로 지정
테이블의 테두리 색 지정	bordercolor	<table bordercolor="red"> //테이블 테두리를 빨간색으로 지정

테이블(<table>) 관련 태그

◆ 테이블의 속성

● 테이블 태그 속성(계속)

기능	속성	사용 예
테이블의 그림자	bordercolordark	<table bordercolordark="red"> //테이블 우측과 아랫부분에 그림자를 빨간색으로 지정
테두리선의 유무 지정	frame	<table frame="voide/hside/vside/above/below/lhs/rhs/box"> //테이블선의 유무를 지정
테이블 안쪽 경계선의 유무 지정	rules	<table rules="none/rows/cols/group/all"> //테이블 안쪽 경계선의 지정
테이블과 주변 글들과의 여백 지정	hspace/vspace	<table hspace/vspace="left/top"> //테이블 내용은 왼쪽 정렬이고, 수직으로는 위쪽 정렬

테이블(<table>) 관련 태그

◆ 행과 셀 내용의 속성

● 행의 속성

기능	속성	사용 예
전행 너비 지정	width	<tr width="600"> //행 너비를 600픽셀로 지정
행 높이 지정	height	<tr height="600"> //행 높이를 600픽셀로 지정
행의 배경색 지정	bgcolor	<tr bgcolor="red"> //행 전체 색상을 빨간색으로 지정
정렬 방식 지정	align/valign	<tr align="center" valign="top"> //행의 가로는 중앙, 세로는 상단으로 정렬

테이블(<table>) 관련 태그

◆ 행과 셀 내용의 속성

● 셀의 속성

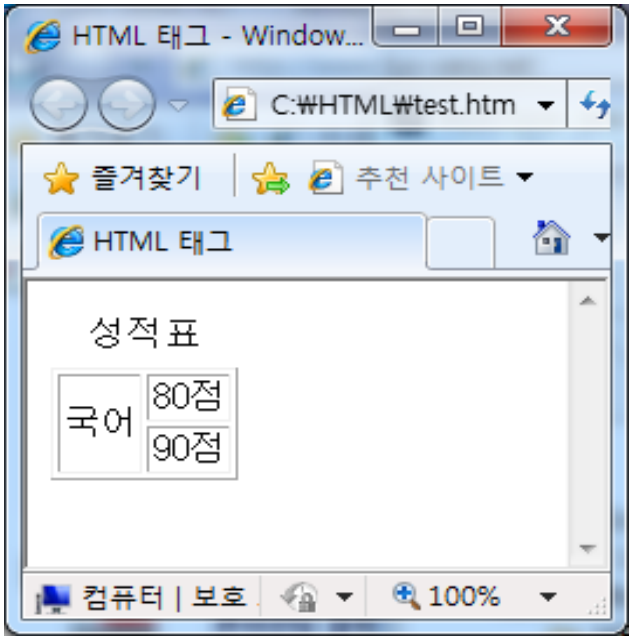
기능	속성	사용 예
셀 내용 너비 지정	width	<td width="600"> //셀 내용 너비를 600픽셀로 지정
셀 내용 높이 지정	height	<td height="600"> //셀 내용 높이를 600픽셀로 지정
셀 내용의 배경색 지정	bgcolor	<td bgcolor="red"> //셀 내용 전체 색상을 빨간색으로 지정
셀 내용의 배경 이미지 지정	background	<td background="bg.gif"> //셀 내용의 이미지 지정
정렬 방식 지정	align/valign	<td align="center" valign="top"> //셀 내용의 가로는 중앙, 세로는 상단으로 정렬
여러 행에 있는 칸들을 통합	rowspan	<td rowspan="2"> //셀 내용이 담긴 2행을 1행으로 합침
여러 열에 있는 행들을 통합	colspan	<td colspan="2"> //셀 내용이 담긴 2열을 1열로 합침

테이블(<table>) 관련 태그

◆ 행과 열의 셀 합치기

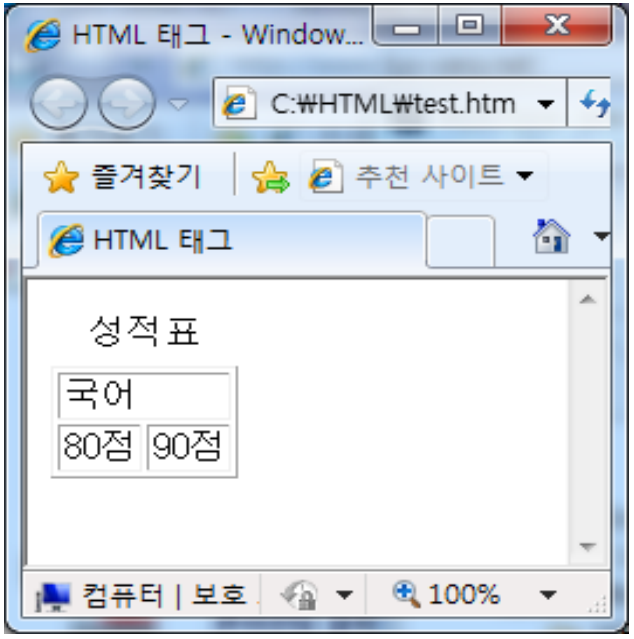
● 행 합치기(rowspan)

```
<table border>
<caption>성적표 </caption>
<tr><td rowspan="2">국어 </td>
    <td>80점 </td>
</tr>
<tr><td>90점</td>
</tr>
</table>
```



● 열 합치기(colspan)

```
<table border>
<caption>성적표 </caption>
<tr><td colspan="2">국어 </td></tr>
<tr>
    <td>80점 </td>
    <td>90점</td>
</tr>
</table>
```



폼(form) 관련 태그

◆ 폼(form) 태그

- **웹 페이지에서 사용자에게 정보를 입력받는 형식**
  - 브라우저 안에서 사용자에게 받은 정보를 서버에게 전달하는 동적인 처리 가능
- **<form> 태그 종류 : 텍스트 입력, 체크박스, 목록상자 등**
- **폼 태그 형식**

```
<form name="폼이름" action="연결될 서버 프로그램명“ method="전송방식">
    <input type= ..... >
    <select> .....</select>
    <textarea>.....</textarea>
</form>
```

  - method : 전송 방식 지정
    - get, post 사용
  - action : 폼 양식에 입력된 값을 처리할 서버 프로그램명 지정(주로 URL)
    - PHP, JSP, ASP, C 등

폼(form) 관련 태그

◆ 폼(form) 태그

- **웹 페이지에서 사용자에게 정보를 입력받는 형식**
  - 브라우저 안에서 사용자에게 받은 정보를 서버에게 전달하는 동적인 처리 가능
- **<form> 태그 종류 : 텍스트 입력, 체크박스, 목록상자 등**
- **폼 태그 형식**

```
<form name="폼이름" action="연결될 서버 프로그램명“ method="전송방식">
    <input type= ..... >
    <select> .....</select>
    <textarea>.....</textarea>
</form>
```

  - method : 전송 방식 지정
    - get, post 사용
  - action : 폼 양식에 입력된 값을 처리할 서버 프로그램명 지정(주로 URL)
    - PHP, JSP, ASP, C 등

폼(form) 관련 태그

◆ Input 형식

● 텍스트 입력 시 사용하는 형태

```
<form>
  <input type= "입력종류" name="이름" size="길이" value="기본 문자열">
</form>
```

● 입력 종류 속성

기능	속성		사용 예
text	name size maxlength value title	이름 화면에 나타날 크기 최대 입력 글자수 기본 문자값 설명문	<input type="text" size="10" value="학생">
password	name size maxlength value title	이름 화면에 나타날 크기 최대 입력 글자수 기본 문자값 설명문	<input type="password" size="10">

폼(form) 관련 태그

◆ Input 형식

● 입력 종류 속성(계속)

기능	속성		사용 예
radio	name value checked	이름 선택한 값을 서버에 전달 초기 체크 표시	<input type="radio" checked>
checkbox	name value checked	이름 선택한 값을 서버에 전달 초기 체크 표시	<input type="checkbox" checked>
button	name value	이름 버튼 위에 표기할 문자	<input type="button" value="전송">
submit	name value	이름 버튼 위에 표기할 문자	<input type="submit" value="쿼리전송">
reset	name value	이름 버튼 위에 표기할 문자	<input type="reset" value="원래대로">
file	name value	이름 버튼 위에 표기할 문자	<input type="file" value="찾아보기">
image	name value src	이름 사용할 이미지 파일 설정	<input type="image" value="버튼" src="button.gif">
hidden	name value	이름 초기값	<input type="hidden" value="홍길동"> //보이지 않는 형식

폼(form) 관련 태그

◆ Select 형식

- 목록상자 입력 형식에 사용되는 태그
- <option> 태그로 하위 목록 리스트 표현

```
<form>
  <select name="이름">
  <option value="해당 목록선택 시 전송될 값"> 화면에 표시할 내용
   ..........................
  </select>
</form>
```

- 사용 가능한 속성
  - name : 이름
  - value : 목록 선택 시 전송될 값
  - selected : 최초에 설정된 기본값
  - size : 목록상자의 크기 지정
  - multiple : 동시에 2개 이상의 기본값을 지정할 수 있음

폼(form) 관련 태그

◆ Textarea 형식

● 여러 줄의 입력을 가능하게 하는 형식

```
<form>
  <textarea name="이름" rows="행의 수" cols="열의 수">
    ..................
  </textarea>
</form>
```

– name : 이름

– rows : 입력 가능한 전체 행의 수

– cols : 입력 가능한 전체 열의 수

– wrap : 자동 줄바꿈 사용

- physical : 화면상의 줄바꿈 상태 그대로 서버에 전송
- virtual : 화면상에서는 줄바꿈되지만, 서버로 전송될 때는 한 줄로 이어서 전송

프레임(frame)

◆ 프레임 형식

- 브라우저 화면을 여러 개로 나누어서 이용하는 것을 의미
- 분할 방식 : 좌/우, 여러 개의 영역으로 분할 가능

```
<frameset rows="각 프레임 크기" | cols ="각 프레임 크기">
  <frame src="보여줄 파일명" name="프레임명">
    ..........................
  </frame>
</frameset>
```

– <Frameset>…</frameset> : 프레임 분할

– <frame>…</frame> : 분할 영역의 내용

– rows : 상하 분할

– cols : 좌우 분할

– 프레임 크기 : 픽셀, %, 상대적 크기 지정 가능

프레임(frame)

◆ 프레임 형식

● 프레임 나누기 예

```
<frameset rows=" 50, 30, * ">
    화면을 상하로 3등분하는데 크기는 50픽셀, 30 픽셀, 나머지로 분할한다.
</frameset>
```

```
<frameset rows=" 50%, 50% ">
    화면을 상하로 50%씩 분할한다.
</frameset>
```

```
<frameset rows=" *, 2* ">
    화면을 상하 2개로 분할하는데 아래가 위의 2배가 되도록 분할한다.
</frameset>
```

프레임(frame)

◆ 프레임 형식

● <frameset> 태그 속성

속성	내용
cols	프레임을 수직 분할
rows	rows 프레임을 수평 분할
border	border 분할 프레임의 경계선에 대한 설정을 하고, 0일 경우 선이 보이지 않음

● <frame> 태그 속성

속성	내용
src	해당 프레임에 출력될 내용을 지정
name	프레임의 이름을 지정
noresize	프레임의 크기를 조절하지 못하도록 지정
scrolling	해당 프레임에서 스크롤 사용 여부 지정(YES|NO|AUTO)
marginheight	해당 프레임의 상하 여백을 픽셀 단위로 지정
marginwidth	해당 프레임의 좌우 여백을 픽셀 단위로 지정

프레임(frame)

◆ iFrame

- 프레임을 나누지 않고 문서 안에서 여러 개의 문서를 보여줄 때 사용

- <iframe> 태그 속성

속성	내용
src	내부 프레임 안에 출력될 문서 지정
name	내부 프레임의 이름을 지정
width	내부 프레임의 너비를 지정
height	내부 프레임의 높이를 지정
frameborder	내부 프레임의 경계선 표시 유무 지정
align	내부 프레임에서 정렬 방식 지정
scrolling	내부 프레임에서 스크롤 사용 여부 지정(YES|NO|AUTO)
marginheight	내부 프레임의 상하 여백을 픽셀 단위로 지정
marginwidth	내부 프레임의 좌우 여백을 픽셀 단위로 지정

스타일 쉬트(Cascading Style Sheet)

1. HTML 이해
2. CSS
3. JavaScript
4. Event

HTML 에 대한 기초 지식

1. HTML 이해
2. CSS (Cascading Style Sheet)
3. JavaScript
4. 브라우저 내장객체
5. 폼(FORM) 관련 객체
6. 레이어(Layer)와 DHTML

스타일(Cascade Style Sheet:CSS)

◆ 스타일시트의 종류

● 내부 스타일시트

```
<head>
  <style type="text/css" 또는 type="text/javascript">
  <!--
    선택자 {속성 : 속성값}
    ..................
   -->
  </style>
</head>
```

● 외부 스타일시트

```
<link rel="stylesheet" type="text/css" 또는 type="text/javascript" href="파일명.css">
```

● 행 스타일시트

```
<태그 style = "속성 : 속성값"> ..................... </태그>
```

스타일(Cascade Style Sheet:CSS)

◆ 스타일시트 정의 방법

● 기본 사용 형식

- 스타일을 정의하기 위한 기본 요소는 속성과 속성값으로 구성
- 스타일을 적용할 대상을 선택자(selector)라 함

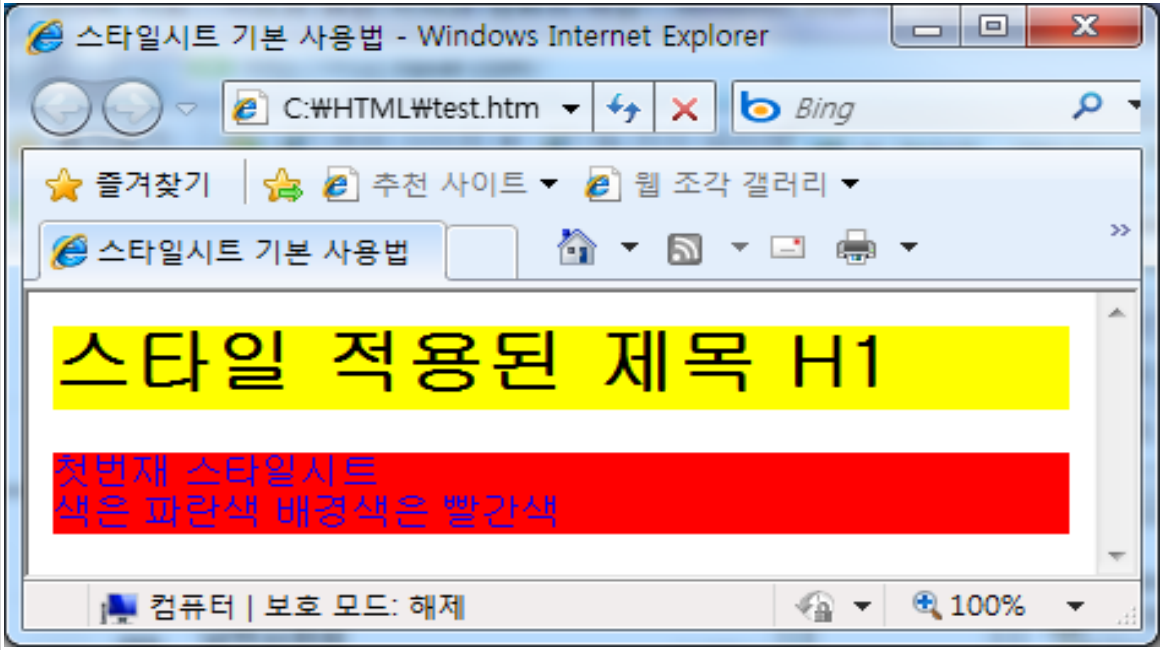
```
선택자1, ...., {속성1 : 속성값1; 속성2: 속성값2; .....속성n:속성값n}
```

스타일(Cascade Style Sheet:CSS)

◆ 다양한 스타일시트 적용 예제

- 기본으로 적용

```
<HTML>
<HEAD>
    <TITLE> 스타일시트 기본 사용법</TITLE>
    <STYLE type="text/css">
      <!--
      p {color:blue; background:red}
      h1{background:yellow}
      -->
    </STYLE>
</HEAD>
<BODY>
    <H1>스타일 적용된 제목 H1 </H1>
    <P>첫번재 스타일시트 <br>
    색은 파란색 배경색은 빨간색</P>
</BODY>
</HTML>
```

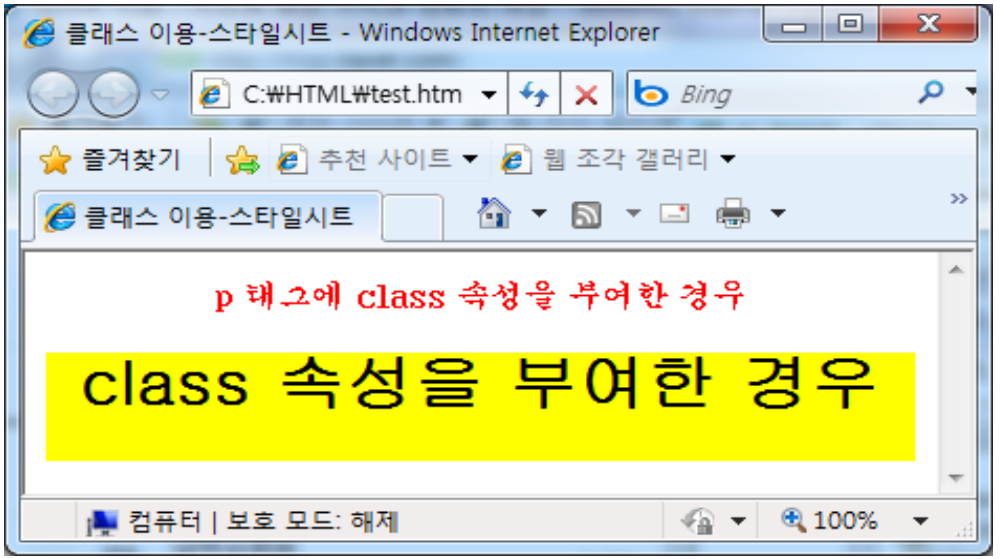


스타일(Cascade Style Sheet:CSS)

◆ 다양한 스타일시트 적용 예제

- class 이용

```
<HTML>
<HEAD>
<TITLE> 클래스 이용-스타일시트</TITLE>
<STYLE type="text/css">
<!--
   p.content1 {font-family:궁서;color:red;}
   .content2 {font-family:굴림;background:yellow;}
-->
</STYLE>
</HEAD>
<BODY><center>
<p CLASS="content1"> p 태그에 class 속성을 부여한 경우
</p>
<h1 CLASS="content2"> class 속성을 부여한 경우</p>
</BODY>
</HTML>
```

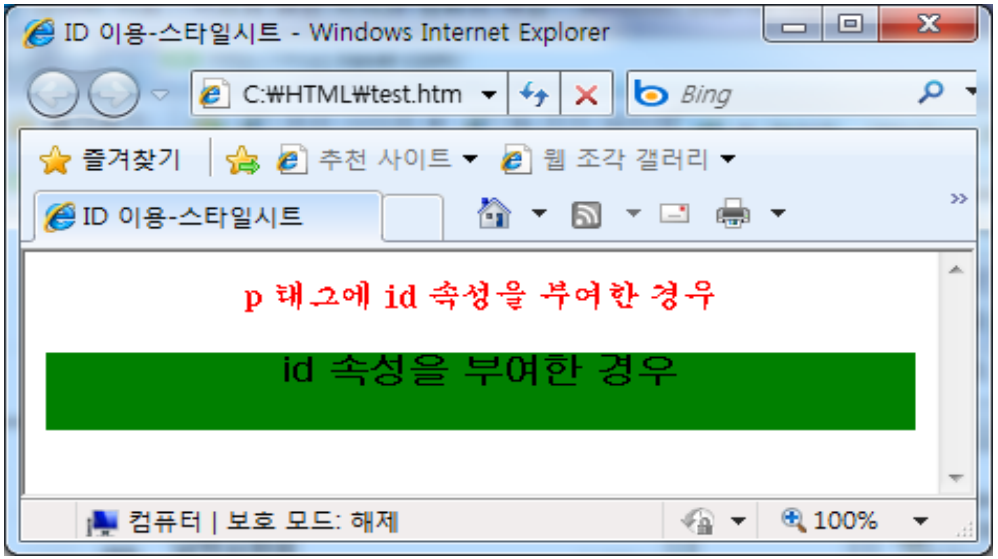


스타일(Cascade Style Sheet:CSS)

◆ 다양한 스타일시트 적용 예제

● id 이용

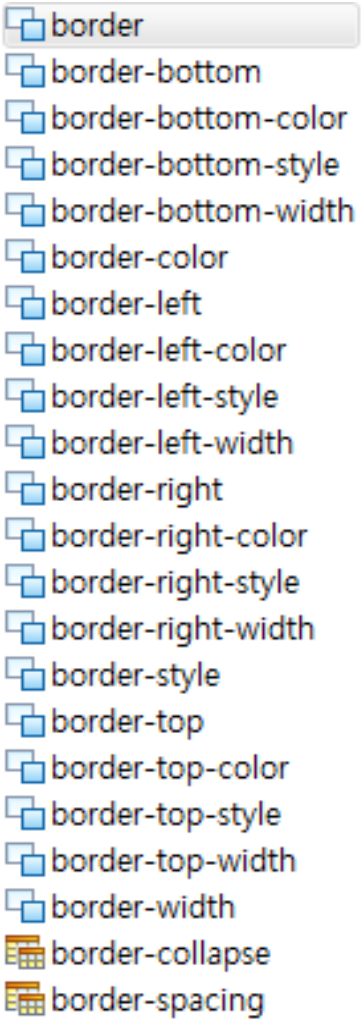
```
<HTML>
<HEAD>
<TITLE> ID 이용-스타일시트</TITLE>
<STYLE type="text/css">
<!--
   #con1 {font-family:궁서;color:red;}
   #con2 {font-family:굴림;background:green;}
-->
</STYLE>
</HEAD>
<BODY><center>
<p id="con1"> p 태그에 id 속성을 부여한 경우</p>
<h3 id="con2"> id 속성을 부여한 경우</p>
</BODY>
</HTML>
```

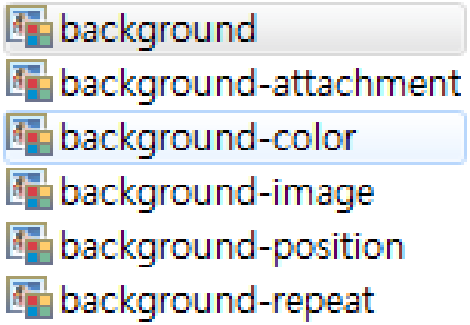


스타일(Cascade Style Sheet:CSS)

◆ 스타일시트 적용 속성

- **border** **backgroup**

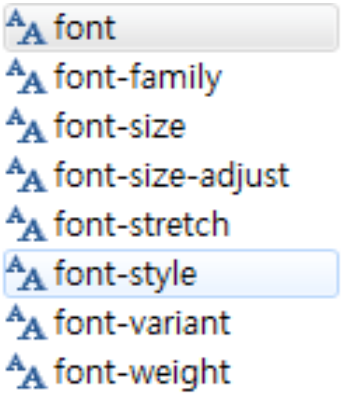




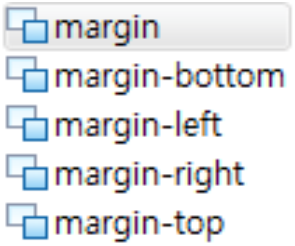
스타일(Cascade Style Sheet:CSS)

◆ 스타일시트 적용 속성

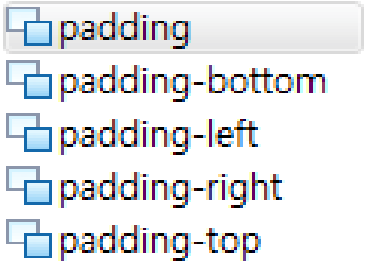
- **font**



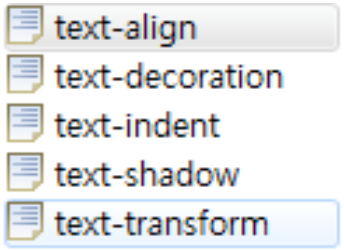
margin



- **padding**



text



스타일(Cascade Style Sheet:CSS)

◆ 스타일시트 적용 속성

style[1].css - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
body { margin:0; padding:0; font-family:Dotum; font-size:12px; color:#959595; background:url(
.bodynone { background:none; }
table, div { font-family:Dotum; font-size:12px; color:#959595; }

h2, h3, h4, h5, h6, div, span, p, dl, dt, dd, ul, ol, li, form, table { margin: 0px; padding:
img { border: none; }
.b { font-weight:bold; }
.n { font-weight:normal; }
.u { text-decoration:underline; }
.aL { text-align:left;}
.aC { text-align:center;}
.aR { text-align:right;}
.aT { vertical-align:top; }
.aM { vertical-align:middle; }
.aB { vertical-align:bottom; }
.just{text-align:justify}
.clear { clear:both; }
.radio{margin:1 0 -1 0;border:none;background:}

.input { border:1px solid #7f9db9; background-color:#ffffff; height:20px; font-family:Dotum;f
.input_customer { border:1px solid #d9d8d8; background-color:#e7eff3; height:18px; font-famil



/* 기본링크 */
a:link { color: #959595; text-decoration: none; }
a:visited { color: #959595; text-decoration: none; }
a:hover { color: #959595; text-decoration: none; }
a:active { color: #959595; text-decoration: none; }

/* MAIN */
.main_02 a:link { font-size:11px; color: #73748e; text-decoration: none; }
.main_02 a:visited { font-size:11px; color: #73748e; text-decoration: none; }
.main_02 a:hover { font-size:11px; color: #73748e; text-decoration: none; }
.main_02 a:active { font-size:11px; color: #73748e; text-decoration: none; }

.main_list02_title a:link { font-size:12px; color: #4f72b9; text-decoration: none; font-weigh
.main_list02_title a:visited { font-size:12px; color: #4f72b9; text-decoration: none; font-we
.main_list02_title a:hover { font-size:12px; color: #4f72b9; text-decoration: none; font-weig
.main_list02_title a:active { font-size:12px; color: #4f72b9; text-decoration: none; font-wei

.main_list02_text a:link { font-size:11px; color: #73748e; text-decoration: none; }
.main_list02_text a:visited { font-size:11px; color: #73748e; text-decoration: none; }
.main_list02_text a:hover { font-size:11px; color: #73748e; text-decoration: none; }
.main_list02_text a:active { font-size:11px; color: #73748e; text-decoration: none; }


/* paging */
.paging a:link { font-size:11px; color: #898989; text-decoration: none; }
.paging a:visited { font-size:11px; color: #464646; text-decoration: none; font-weight:bold;
.paging a:hover { font-size:11px; color: #464646; text-decoration: none; font-weight:bold; }
.paging a:active { font-size:11px; color: #464646; text-decoration: none; font-weight:bold; }
```

style[1].css - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
/* PRESS CENTER */
.press a:link { font-size:12px; color: #959595; text-decoration: none; }
.press a:visited { font-size:12px; color: #f27322; text-decoration: none; }
.press a:hover { font-size:12px; color: #f27322; text-decoration: none; }
.press a:active { font-size:12px; color: #f27322; text-decoration: none; }

.press01 a:link { font-size:12px; color: #f27322; text-decoration: none; }
.press01 a:visited { font-size:12px; color: #f27322; text-decoration: none; }
.press01 a:hover { font-size:12px; color: #f27322; text-decoration: none; }
.press01 a:active { font-size:12px; color: #f27322; text-decoration: none; }

/* CUSTMOER CENTER */
.customer a:link { font-size:12px; color: #959595; text-decoration: none; }
.customer a:visited { font-size:12px; color: #8ec050; text-decoration: none; }
.customer a:hover { font-size:12px; color: #8ec050; text-decoration: none; }
.customer a:active { font-size:12px; color: #8ec050; text-decoration: none; }

.customer01 a:link { font-size:12px; color: #8ec050; text-decoration: none; }
.customer01 a:visited { font-size:12px; color: #8ec050; text-decoration: none; }
.customer01 a:hover { font-size:12px; color: #8ec050; text-decoration: none; }
.customer01 a:active { font-size:12px; color: #8ec050; text-decoration: none; }

/* SITEMAP */
.menu01_01 a:link { font-size:12px; color: #4baec8; text-decoration: none; font-weight:bold;
.menu01_01 a:visited { font-size:12px; color: #4baec8; text-decoration: none; font-weight:bol
.menu01_01 a:hover { font-size:12px; color: #4baec8; text-decoration: none; font-weight:bold;
.menu01_01 a:active { font-size:12px; color: #4baec8; text-decoration: none; font-weight:bold

.menu01_02 a:link { font-size:12px; color: #7496d9; text-decoration: none; font-weight:bold;
.menu01_02 a:visited { font-size:12px; color: #7496d9; text-decoration: none; font-weight:bol
.menu01_02 a:hover { font-size:12px; color: #7496d9; text-decoration: none; font-weight:bold;
.menu01_02 a:active { font-size:12px; color: #7496d9; text-decoration: none; font-weight:bold

.menu01_03 a:link { font-size:12px; color: #a483df; text-decoration: none; font-weight:bold;
.menu01_03 a:visited { font-size:12px; color: #a483df; text-decoration: none; font-weight:bol
.menu01_03 a:hover { font-size:12px; color: #a483df; text-decoration: none; font-weight:bold;
.menu01_03 a:active { font-size:12px; color: #a483df; text-decoration: none; font-weight:bold

.menu01_04 a:link { font-size:12px; color: #f9a64a; text-decoration: none; font-weight:bold;
.menu01_04 a:visited { font-size:12px; color: #f9a64a; text-decoration: none; font-weight:bol
.menu01_04 a:hover { font-size:12px; color: #f9a64a; text-decoration: none; font-weight:bold;
.menu01_04 a:active { font-size:12px; color: #f9a64a; text-decoration: none; font-weight:bold

.menu01_05 a:link { font-size:12px; color: #8db658; text-decoration: none; font-weight:bold;
.menu01_05 a:visited { font-size:12px; color: #8db658; text-decoration: none; font-weight:bol
.menu01_05 a:hover { font-size:12px; color: #8db658; text-decoration: none; font-weight:bold;
.menu01_05 a:active { font-size:12px; color: #8db658; text-decoration: none; font-weight:bold


/************ 스크롤바 ************/
.scr {
scrollbar-face-color: #EDEDED;
```

HTML 에 대한 기초 지식

1. HTML 이해
2. CSS (Cascading Style Sheet)
3. JavaScript
4. 브라우저 내장객체
5. 폼(FORM) 관련 객체
6. 레이어(Layer)와 DHTML

자바스크립트(Javascript) 개요

◆ 자바스크립트의 개요

- 자바스크립트는 웹 브라우저에서 사용할 수 있는 스크립트 언어
- HTML 문서 내에 함께 존재

◆ 자바스크립트의 특징

- 브라우저 해석기에서 실행하므로 별도 프로그램 필요 없음
- HTML로는 표현이 불가능했던 프로그램적인 활용이나 동적인 표현이 가능해 좀더 역동적인 홈페이지를 만들 수 있음
- 서버로 전송될 자료의 검증을 웹 프로그램이 실행되기 전인 클라이언트 단계에서 실행
- AJAX 기술을 사용한 비동기 통신 가능

자바스크립트(Javascript) 사용법

◆ 자바스크립트의 기본 구조

- <HEAD> 태그 내에서 선언하고 <BODY> 태그 내에서 실행
- <SCRIPT> . . . . . </SCRIPT> 태그 내에서 구성

```
<SCRIPT>
 자바스크립트 내용
</SCRIPT>
```

```
 <SCRIPT src="자바스크립트 파일명">
</SCRIPT>
```

- 주석문-프로그램 내용 설명

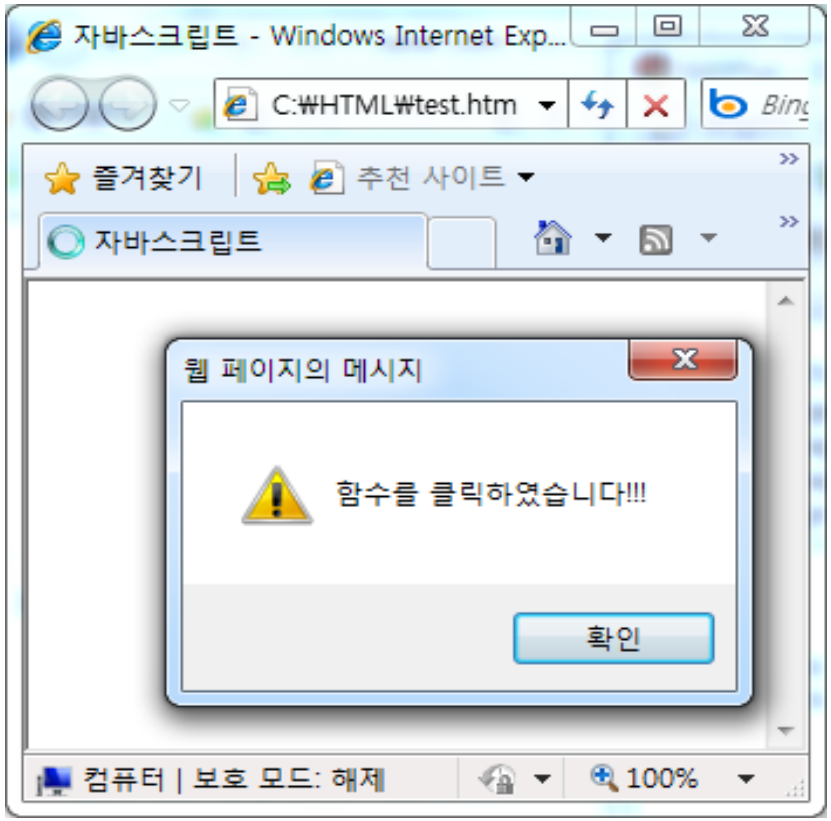
```
/* . . . . . */  => 여러 줄 주석
//  =>한 줄 주석
```

자바스크립트(Javascript) 사용법

◆ 자바스크립트의 실행

● 자바스크립트 함수를 호출하여 실행

```
<HTML>
  <HEAD><TITLE>자바스크립트 </TITLE>
    <SCRIPT LANGUAGE="JavaScript">
    function 함수명( ) {
        alert("함수를 클릭하였습니다!!!");
    }
    </SCRIPT>
  </HEAD>
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      함수명()
    </SCRIPT>
  </BODY>
</HTML>
```

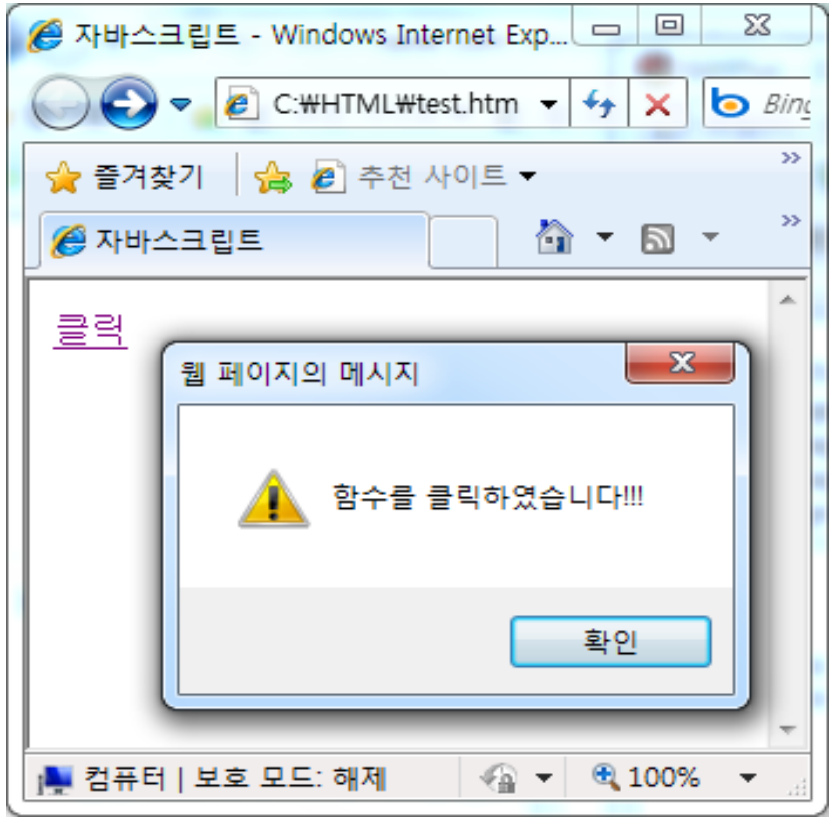


자바스크립트(Javascript) 사용법

◆ 자바스크립트의 실행

● 이벤트가 발생할 때 자바스크립트 함수를 호출하여 실행

```
<HTML>
  <HEAD><TITLE>자바스크립트 </TITLE>
    <SCRIPT LANGUAGE="JavaScript">
    function 함수명( ) {
          alert("함수를 클릭하였습니다!!!");
    }
    </SCRIPT>
  </HEAD>
  <BODY>
       <a href="javascript:함수명();">클릭</a>
  </BODY>
</HTML>
```



자바스크립트(Javascript) 사용법

◆ 자바스크립트 사용 시 주의 사항

- 한 줄에 한 개 이상의 항과 기호로 구성
- 새로운 줄은 새로운 문장으로 시작
- 한 줄에서 두 문장 이상을 작성 할 때 세미콜론(;)사용
- 동일한 실행문들은 { }로 둘러싼다.
- 명령문이 길어서 한 줄을 넘어 갈 때 '_'로 연결
- 따옴표가 중복되는 경우 외부에는 큰 따옴표, 내부에는 작은 따옴표 사용

자바스크립트(Javascript) 기본 문법

◆ 변수 개요

```
var kim // 변수 선언 변수명은 kim
var i, j, k // 동시에 여러 개의 변수 선언
var i=1 // 변수 선언과 동시에 기본값 할당
```

- 대소문자를 구별하며, 반드시 첫 자는 영문자로 시작
- 문자와 숫자만으로 구성, 한글과 밑줄(_)를 제외한 특수 문자와 공백은 사용 불가
- 예약어, 함수명, 객체명, 속성 등은 변수로 사용 불가

◆ 변수의 종류

- 전역변수와 지역변수
- 전역변수 : 스크립트 어디에서나 사용 가능
- 지역변수 : 해당 함수 내에서만 사용

자바스크립트(Javascript) 기본 문법

◆ 연산자

- **산술 연산자 : 수치 계산에 사용**
  - +, -, *, /
- **관계 연산자 : 값을 비교하는 연산자**
  - <, >, <=, >=, ==, !=
- **조건 연산자 : 조건에 따라 참과 거짓으로 나타내는 연산**
  - 조건식 ? 표현문 1 : 표현문 2
- **비트 연산자 : 두 수의 비트 사이에 일어나는 연산**
  - A | B, A & B, A ^ B

자바스크립트(Javascript) 기본 문법

◆ 연산자

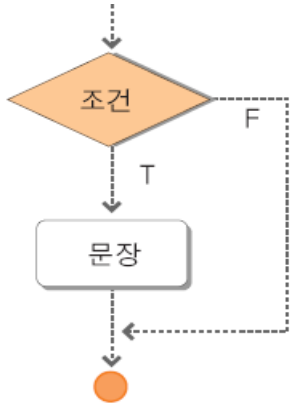
- **대입 연산자 : 변수에 값을 대입할 때 사용하는 연산자**
  - =, +=, -=, ...
- **논리 연산자 : 조건의 참, 거짓을 판단**
  - &&(AND), ||(OR), !(NOT)
- **문자열 연산자 : 문자들을 연결시키는 역할**
  - , +

자바스크립트(Javascript) 기본 문법

◆ 조건문

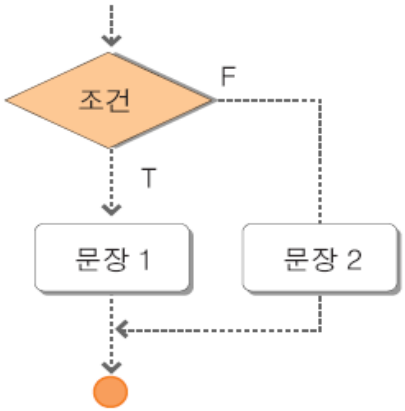
- 조건이 참인 경우에만 처리

```
if (조건) {
    문장;
}
```



- 조건에 만족하는 경우와 그렇지 않은 경우의 처리

```
if (조건) {
    문장 1;
}
else {
    문장 2;
}
```

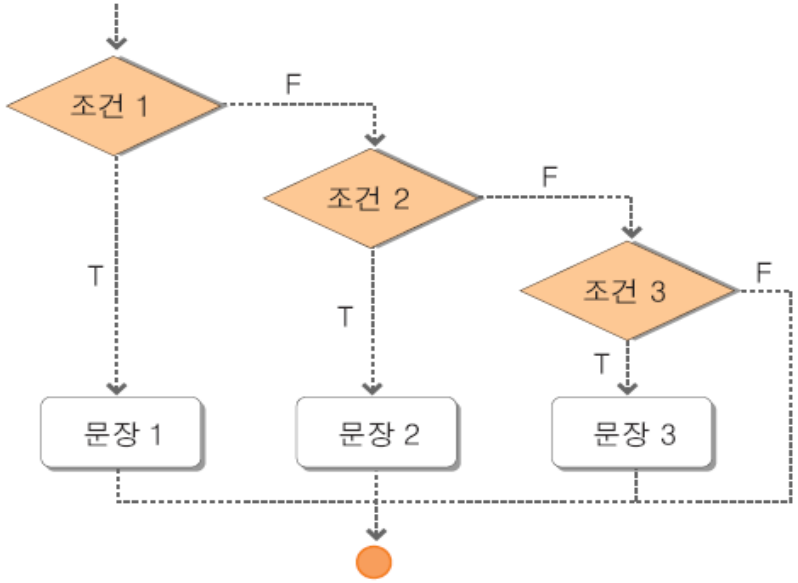


자바스크립트(Javascript) 기본 문법

◆ 조건문

● 조건이 참인 경우와 조건이 거짓인 경우 n개일 때의 처리

```
if (조건 1) {
    문장 1;
}
else if (조건 2) {
    문장 2;
}
else if (조건 3) {
    문장 3;
}
.....
```

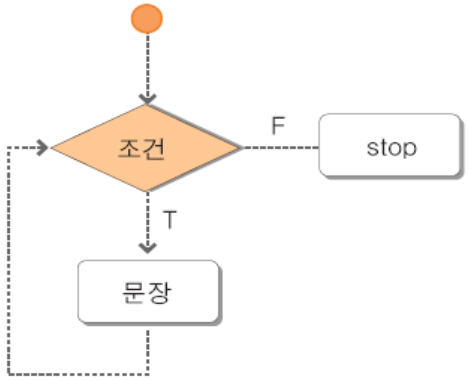


자바스크립트(Javascript) 기본 문법

◆ 반복문

- for문 : 초기값에 의해 시작되고, 조건을 만족할 때까지 실행

```
for(초기값; 조건; 증감값) {
        문장;
}
```



- while 문 : 조건이 참인 동안 문장을 수행

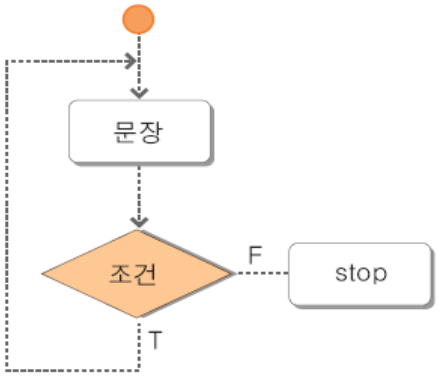
```
while (조건) {
   문장;
}
```

자바스크립트(Javascript) 기본 문법

◆ 반복문

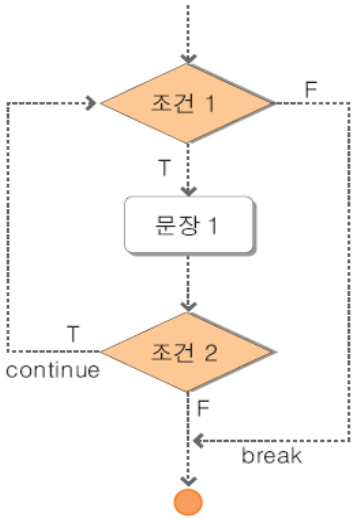
- do..while 문 : while 문과 다르게 반드시 문장을 한번은 수행

```
do {
   문장;
} while(조건);
```



- break(블럭 종료)와 continue(조건문으로 분기)문

```
while (조건 1) {
   문장;
   if (조건 2) {
      continue;
   }
   else {
      break;
   }
문장 2;
}
```



함수의 개요

◆ 함수의 형식

● 자바스크립트 내장 함수

– alert() : 경고 창을 띄워주는 내장 함수

– prompt() : 사용자 입력창을 띄워 입력한 값을 리턴

– confirm() : 다이얼로그 박스를 띄워 확인/취소에 따라 결과값을 리턴

– eval() : 문자열로 입력된 수식을 계산해주는 내장 함수

● 사용자 정의 함수

– 내장함수 이외의 기능을 사용하고 할 때, 사용자 정의 함수를 사용

```
<HTML>
<HEAD><TITLE>함수의 형식</TITLE>
  <SCRIPT LANGUAGE="JavaScript" >
    function 함수명(매개변수1, 매개변수2,....) {
      함수 처리 내용;
    }
</SCRIPT>
</HEAD>
```

함수의 개요

◆ 매개변수가 없는 함수

- 가장 단순한 형태로 호출하고 처리를 종료하는 경우 사용

```
function 함수명() {
  함수 처리 내용;
}
```

◆ 매개변수가 있는 함수

- 함수에 사용자가 값을 주는 형태로 여러 값을 가지고 호출

```
function 함수명(매개변수1,매개변수2,.....) {
  함수 처리 내용;
}
```

함수의 개요

◆ 매개변수가 없는 함수

- 가장 단순한 형태로 호출하고 처리를 종료하는 경우 사용

```
function 함수명() {
  함수 처리 내용;
}
```

◆ 매개변수가 있는 함수

- 함수에 사용자가 값을 주는 형태로 여러 값을 가지고 호출

```
function 함수명(매개변수1,매개변수2,.....) {
  함수 처리 내용;
}
```

◆ 결과를 돌려주는 사용자 정의 함수

- 함수에서 처리한 결과를 다시 호출 부분에서 처리
- return을 사용해서 함수 호출부에 결과를 돌려주는 역할

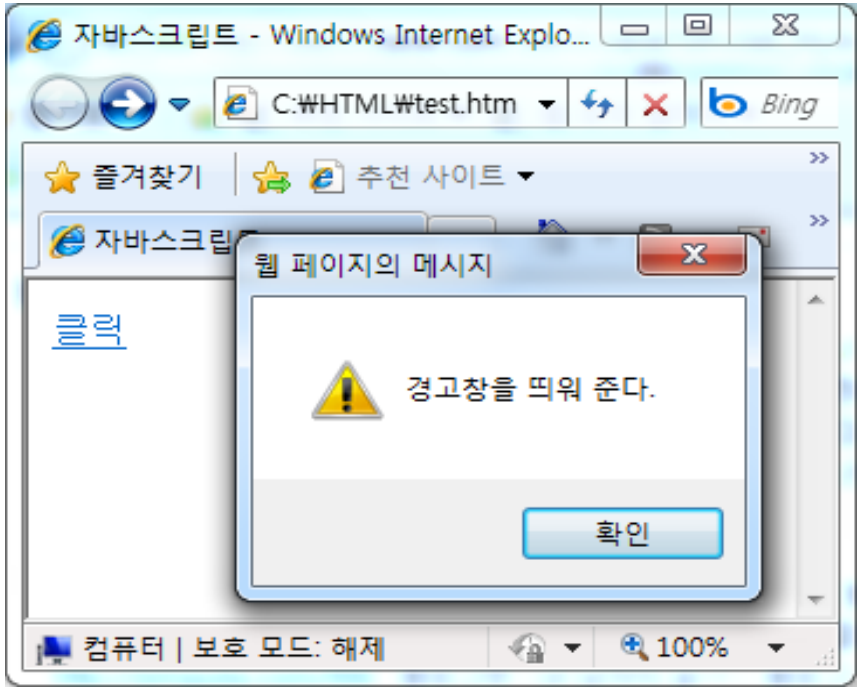
```
function 함수명(매개변수1,매개변수2,.....) {
  return 결과값;
}
```

자바스크립트 내장 함수

◆ 내장 함수

- **alert() : 경고 창을 띄워주는 내장함수**

```
<HTML>
  <HEAD><TITLE>자바스크립트 </TITLE>
    <SCRIPT LANGUAGE="JavaScript">
    function 함수명() {
          alert("경고창을 띄워 준다.");
    }
    </SCRIPT>
  </HEAD>
  <BODY>
       <a href="javascript:함수명();">클릭</a>
  </BODY>
</HTML>
```

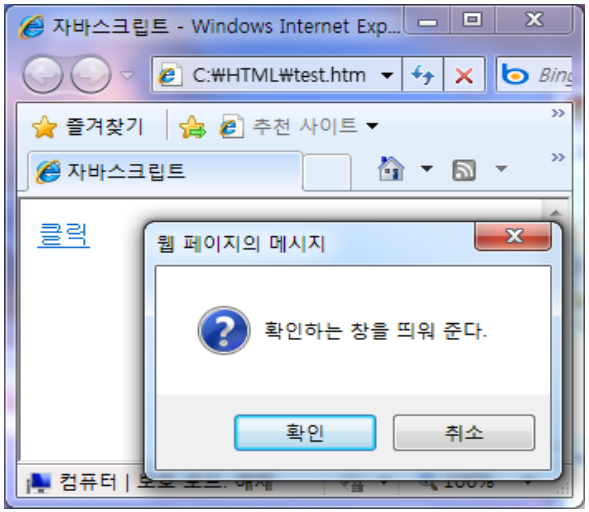


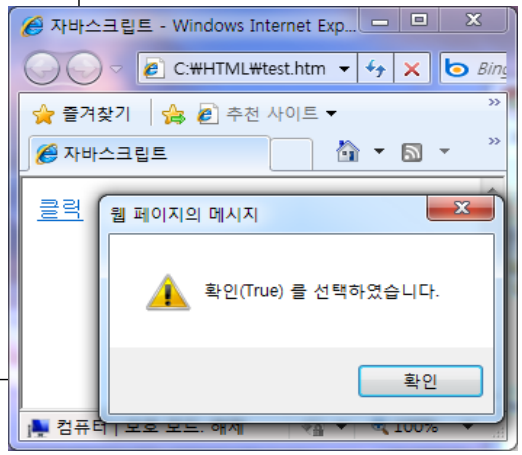
자바스크립트 내장 함수

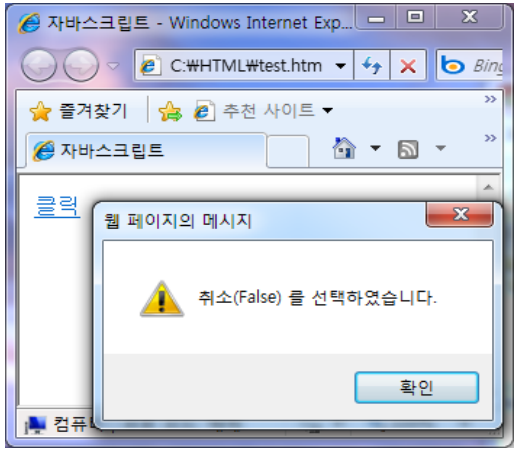
◆ 내장 함수

- **conform() : True/False 값으로 확인하는 창을 띄워주는 내장함수**

```
<HTML>
  <HEAD><TITLE>자바스크립트 </TITLE>
    <SCRIPT LANGUAGE="JavaScript">
    function 함수명() {
        var result = confirm("확인하는 창을 띄워 준다.");
            if(result) {
                alert("확인(True) 를 선택하였습니다.");
            } else {
                alert("취소(False) 를 선택하였습니다.");
            }
    }
    </SCRIPT>
  </HEAD>
  <BODY>
        <a href="javascript:함수명();">클릭</a>
  </BODY>
</HTML>
```





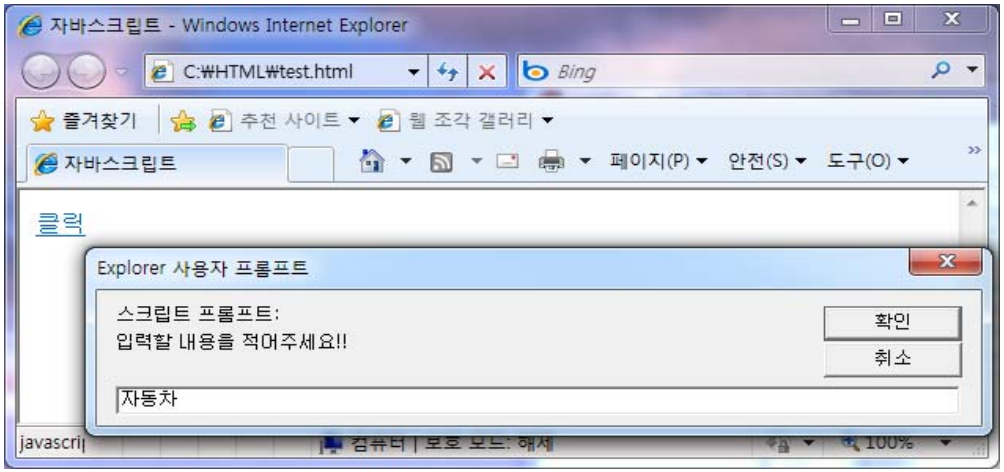


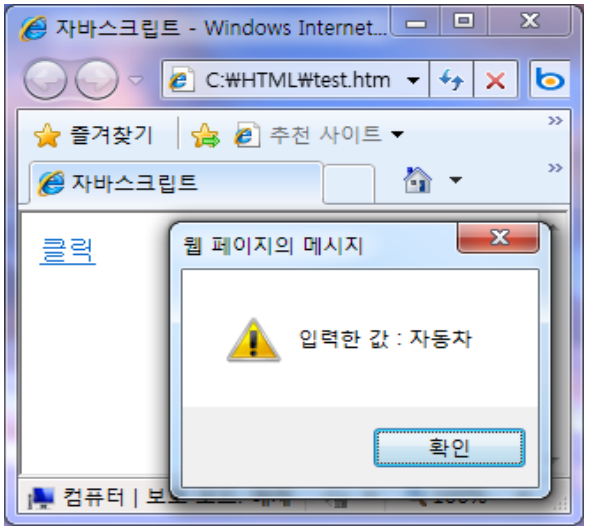
자바스크립트 내장 함수

◆ 내장 함수

- **prompt() : 입력 창을 띄워주는 내장함수**

```
<HTML>
  <HEAD><TITLE>자바스크립트 </TITLE>
    <SCRIPT LANGUAGE="JavaScript">
    function 함수명() {
        var result = prompt("입력할 내용을 적어주세요!!");
        alert("입력한 값 : " + result);
    }
    </SCRIPT>
  </HEAD>
  <BODY>
      <a href="javascript:함수명();">클릭</a>
  </BODY>
</HTML>
```



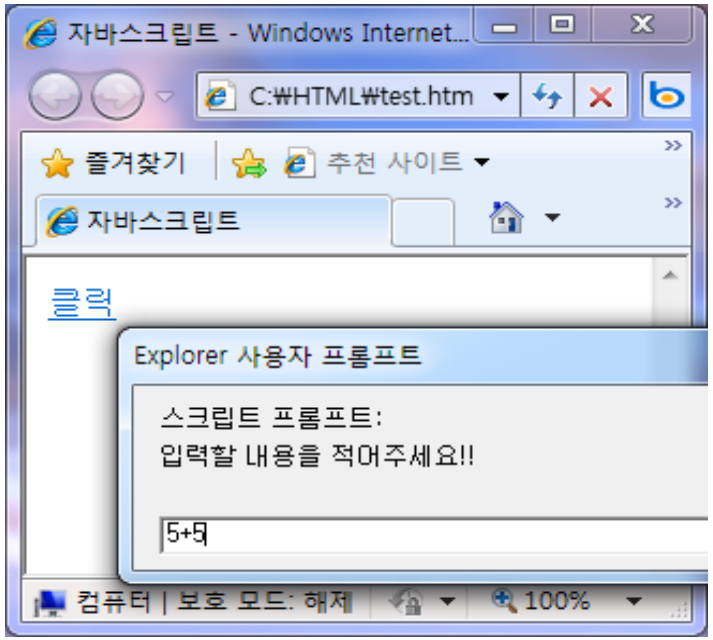


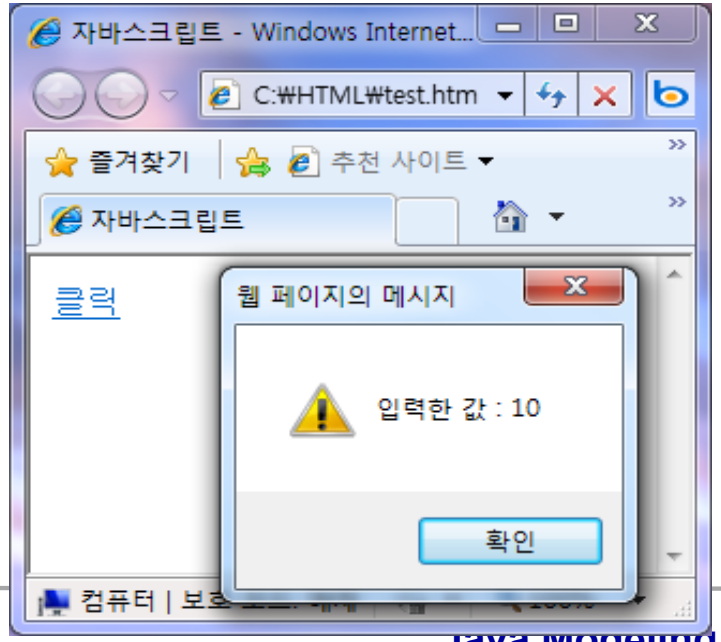
자바스크립트 내장 함수

◆ 내장 함수

- eval() : 문자열려 입력된 수식을 계산해주는 내장함수
- 입력양식을 통해 입력된 값을 계산할 때 유용

```
<HTML>
  <HEAD><TITLE>자바스크립트 </TITLE>
    <SCRIPT LANGUAGE="JavaScript">
    function 함수명() {
        var input = prompt("입력할 내용을 적어주세요!!");
        var result = eval(input)
        alert("입력한 값 : " + result);
    }
    </SCRIPT>
  </HEAD>
  <BODY>
      <a href="javascript:함수명();">클릭</a>
  </BODY>
</HTML>
```





자바스크립트 내장 함수

◆ 다른 유용한 내장 함수

- **parseInt( ) 함수 – 문자열을 정수로 반환**
- **parseFloat( ) 함수 – 문자열을 실수로 반환**

```
parseInt(문자열, 변환 진법)
parseFloat(문자열)
```

- **isNaN( ) 함수 – 문자판단. 숫자면 false 문자면 true 반환**

```
isNaN("판별할 내용")
```

- **escape( )/unescape( ) 함수 – ASCII코드와 문자 교환**

```
escape(문자)
unescape(문자열)
```

자바스크립트 객체

◆ 객체란

- **다른 프로그래밍 언어와 같이 자바스크립트도 객체를 정의**
- **자바스크립트에서 객체는 생성자 함수를 이용해서 생성**
  - 생성자 함수는 객체의 속성과 메소드를 정의하기 위해서 사용되는 함수

```
function 객체이름([전달인수 리스트]) {
   속성을 위한 변수 [= 초기값];
   ……
   메소드를 위한 함수 = 함수이름;
   ……
}
```

```
Member = function(id, name, address) {
        this.id = id;
        this.name = name;
        this.address = address;
 }
```

자바스크립트 객체

◆ 객체란

- 객체에 함수를 정의
- 두 가지 방법으로 함수 정의 가능

```
클래스명.prototype = {
   함수이름:function(파라미터) {
        함수내용;
          ……
    } // 함수와 함수는 콤마(,)로 구분
}
```

```
Member.prototype = {
        show:function() {
            return this.id + ":" + this.name + ":"
                    + this.address;
        },
        home:function(address) {
            return "집주소 : " + address;
        }
    }
```

```
클래스명.prototype.함수명 = function(파라미터) {
     함수내용;
          ……
}
```

```
Member.prototype.setName = function(name) {
        this.name = name;
    }
```

자바스크립트 객체

◆ 객체란

● 객체 사용

– new 연산자를 사용하여 객체를 생성

```
var 변수명 = new 클래스명(파라미터);
```

```
var member1 = new Member("java", "홍길동", "서울시 강남구");
```

자바스크립트 객체

◆ 객체란

● 객체 사용 : 예제

– new 연산자를 사용하여 객체를 생성

```
<HTML>
  <HEAD><TITLE>자바스크립트 </TITLE>
    <SCRIPT LANGUAGE="JavaScript">

    Member = function(id, name, address) {
        this.id = id;
        this.name = name;
        this.address = address;
    }

    Member.prototype = {
        show:function() {
            return this.id + ":" + this.name + ":" + this.address;
        },
        home:function(address) {
            return "집주소 : " + address;
        }
    }

    Member.prototype.setName = function(name) {
        this.name = name;
    }
```

자바스크립트 객체

◆ 객체란

● 객체 사용 : 예제

– new 연산자를 사용하여 객체를 생성

```
    function 함수명() {

        var member1 = new Member("java", "홍길동", "서울시 강남구");

        alert(member1.show());
        alert(member1.home("서울시 서초구"));

        member1.setName("이순신");
        alert(member1.show());
    }
    </SCRIPT>
</HEAD>
<BODY>
    <a href="javascript:함수명();">클릭</a>
</BODY>
</HTML>
```

◆ 내장 객체

● Date 객체

– 프로그램에서 날짜와 시간에 대한 정보 제공

```
객체명 = new Date();
객체명 = new Date(년,월,일);
객체명 = new Date(년, 월, 일, 시, 분, 초);
```

– Date 객체의 속성과 메소드

속성	내용
prototype	Date 객체에 속성 부여
메소드	**내용**
getYear( )	년도(2개의 숫자로 1970년 이후의 년도를 표시)
getMonth( )	월(0=1월, 1=2월, 2=3월로 표시)
getDate( )	일(1에서 31의 정수로 표시)
getDay( )	요일(0=일요일, 1=월요일로 표시)
getHours( )	시(0과 23 사이의 수로 표시)
getMinute( )	분(0과 59 사이의 수로 표시)
getSeconds( )	초(0과 59 사이의 수로 표시)
getTime( )	1970.1.1.일 0시 이후의 시간을 ms(1000분의 1초)로 표시

◆ 내장 객체

● Date 객체

– Date 객체의 속성과 메소드 (계속)

메소드	내용
setDate(날짜값)	1에서 31까지의 숫자로 날짜를 변경
setHours(시간)	0에서 23까지의 숫자로 시간을 변경
setMinute(분)	0에서 59까지의 숫자로 분을 변경
setMonth(월)	0에서 11까지의 숫자로 월을 변경
setSeconds(초)	0에서 59까지의 숫자로 초를 변경
setTime(시간)	1970.1.1. 0시 이후의 시간을 ms(1000분의 1초)의 숫자로 변경
setYear(년도)	1900보다 큰 년도를 4개의 숫자로 변경
parse(날짜 문자열)	1970.1.1. 0시 이후의 ms의 숫자를 문자열과 변경
toGMTString( )	인터넷의 GMT 규약을 사용해 현재의 날짜와 시간을 변경
toLocaleString( )	현재 지역 시간을 현재의 Date 객체의 값에 반환

자바스크립트 객체

◆ 내장 객체

● String 객체

– " "를 통해서 String 객체 선언

```
변수= "문자열";
"문자열".속성[메소드()];
```

– String 개체의 속성과 메소드

속성		내용	
length		문자열이 가지고 있는 문자의 개수	
prototype		String 객체에 속성 부가	
메소드		**내용**	**동일 기능 HTML 태그**
문자 모양	big( )	글자를 좀더 크게	〈BIG〉 글자 〈/BIG〉
	small( )	글자를 좀더 작게	〈SMALL〉 글자 〈/SMALL〉
	bold( )	볼드체	〈B〉 글자 〈/B〉
	fixed( )	타자기체	〈TT〉 글자 〈/TT〉
	italics( )	이탤릭체	〈I〉 글자 〈/I〉
	strike( )	글자 가운데 줄긋기	〈STRIKE〉 글자 〈/STRIKE〉
	sub( )	아래 첨자	〈SUB〉 글자 〈/SUB〉
	sup( )	윗 첨자	〈SUP〉 글자 〈/SUP〉
	fontcolor(색)	글자색	〈FONT COLOR="색"〉 글자 〈/FONT〉
	fontsize(크기)	글자 크기	〈FONT SIZE="크기"〉 글자 〈/FONT〉

◆ 내장 객체

● String 객체

– String 개체의 속성과 메소드(계속)

메소드		내용	동일 기능 HTML 태그
하이퍼텍스트 연결	anchor("표식")	표식 지정	〈A NAME="표식"〉 글자 〈/A〉
	link("위치")	하이퍼텍스트 연결	〈A HREF="위치"〉 글자 〈/A〉
문자 처리	charAt(n)	n번째에 해당하는 문자 찾기	
	indexOf("문자열")	지정된 문자열의 위치값 계산(왼쪽부터)	
	lastIndexOf("문자열")	지정된 문자열의 위치값 계산(오른쪽부터)	
	subString(n, m)	문자열의 n번째 문자부터 m번째 문자까지 표시. 음수값은 무시	
	concat("문자열")	문자열을 문자열 객체에 연결	
	slice(n, m)	subSrting과 동일, 음수값은 오른쪽부터 순번으로 계산	
	split("구분문자")	구분문자로 문자열 분리	
	toUpperCase( )	대문자로 만들기	
	toLowerCase( )	소문자로 만들기	

◆ 내장 객체

● Math 객체

– 삼각, 지수, 로그 함수와 같은 연산을 위한 속성과 메소드 제공

```
Math.속성[메소드()];
```

– Math 객체의 속성과 메소드

속성	내용	속성	내용
E	오일러 상수	PI	π (약 3.1415)
LN10	1의 자연 로그(약 2.302)	SQRT 2	$\sqrt{2}$ (약 1.414)
LN2	2의 자연 로그(약 0.693)		
메소드	**내용**	**메소드**	**내용**
abs(수)	수의 절대값	floor(수)	수와 같거나 작은 정수값
acos(수)	코사인 값을 호도법으로 반환	log(수)	수의 자연 로그값
asin(수)	사인 값을 호도법으로 반환	max(수1, 수2)	두 수 중 큰 값
atan(수)	탄젠트 값을 호도법으로 반환	min(수1, 수2)	두 수 중 작은 값
ceil(수)	수와 같거나 큰 정수값을 반환	pow(수1, 수2)	수1의 수2 승
cos(수)	코사인 값	random(수)	0에서 1 사이의 난수
sin(수)	사인 값	round(수)	수에서 가장 가까운 정수
tan(수)	탄젠트 값	sqrt(수)	$\sqrt{수}$
exp(수)	수의 e 승		

◆ 내장 객체

● Array 객체

– Array()는 길이가 정해지지 않은 배열 생성

```
변수명 = new Array(첨자);
변수명 = new Array();
```

– Array 객체의 속성과 메소드

속성	내용
length	배열 내의 항목 수
prototype	Array 객체에 속성을 부가하는 수단 (comment는 설명문을 추가하는 변수로 자주 사용됨)

메소드	내용
join(문자열)	배열의 각 요소를 포함한 문자열을 반환
reverse( )	배열의 순서를 반대로 변경
sort(함수)	함수에 따라 배열을 정렬
slice(시작, 끝)	배열의 일부를 추출
concat(배열)	두 개의 배열을 연결

◆ 내장 객체

● Function 객체

```
함수명 = new Function(매개변수1, 매개변수2, ..... , 함수처리 내용)
```

– Function 객체의 속성

속성	내용
arguments	함수의 매개변수 정보를 배열로 제공
prototype	function 객체의 속성을 정의

자바스크립트 객체

◆ 내장 객체

● Screen 객체

– 컴퓨터의 해상도나 색상에 관한 정보를 제공

```
SCREEN.속성;
```

– Screen 객체의 속성

속성	내용
height	화면의 높이
width	화면의 너비
availHeight	작업 표시줄을 제외한 화면의 높이
availWidth	작업 표시줄을 제외한 화면의 너비
availTop	화면 표시 영역의 y 좌표 표시
availLeft	화면 표시 영역의 x 좌표 표시
colorDepth	컴퓨터에서 사용 가능한 색상 수
pixelDepth	화면의 컬러 해상도를 표시

● Number 객체

– 입력된 값을 숫자로 바꾸어 주는 객체

이벤트와 이벤트 핸들러

◆ 이벤트란?

- 특별한 사건이 일어났을 때 발생하는 일종의 신호(행위자체)

◆ 이벤트 핸들러

- 이벤트를 통해 설정된 일정한 작업을 수행(행위전달시점)

- 사용 형식

```
<~~ on이벤트이름="이벤트 핸들러" ……..>
```

- 이벤트와 이벤트 핸들러

이벤트	이벤트 핸들러	내용
그림 관련 이벤트	abort(onAbort)	사용자가 이미지를 읽다가 중단할 경우
	error(onError)	웹 페이지가 불러들일 때 에러가 발생한 경우
포커스 관련 이벤트	blur(onBlur)	사용자로부터 입력받는 양식에서 커서가 다른 곳으로 이동한 경우
	focus(onFocus)	사용자의 입력 양식 안으로 커서가 들어왔을 경우

이벤트와 이벤트 핸들러

◆ 이벤트 핸들러

● 이벤트와 이벤트 핸들러(계속)

마우스 관련 이벤트	click(onClick)	사용자가 링크나 입력 양식을 클릭한 경우
	dbclick(onDblick)	사용자가 마우스를 더블 클릭한 경우
	mousedown(onMousedown)	사용자가 마우스 버튼을 누른 경우
	mouseover(onMouseover)	사용자의 마우스가 링크된 곳을 지나가는 경우
	mouseup(onMouseup)	사용자가 마우스를 눌렀다 놓을 경우
	mousemove(onMousemove)	사용자가 마우스를 움직일 경우
	dragdrop(onDragdrop)	사용자가 마우스를 클릭한 상태에서 움직였을 경우
폼 관련 이벤트	change(onChange)	사용자로부터 입력받는 양식의 값이 바뀌는 경우
	submit(onSubmit)	사용자가 입력한 양식을 서버로 전송할 경우
	select(onSelect)	사용자가 입력 양식의 한 항목을 선택한 경우
키보드 관련 이벤트	keydown(onKeydown)	사용자가 키를 입력했을 경우
	keypress(onKeypress)	사용자가 키를 눌렀을 경우
	keyup(onKeyup)	사용자가 키를 눌렀다 놓았을 경우
윈도우 관련 이벤트	load(onLoad)	네스케이프에서 웹페이지를 열 경우
	move(onMove)	사용자가 윈도우를 움직일 경우
	resize(onResize)	사용자가 윈도우나 프레임의 크기를 변경한 경우
	unload(onUnload)	사용자가 현재의 웹 페이지에서 다른 곳으로 이동하는 경우

이벤트와 이벤트 핸들러

◆ 이벤트 핸들러

● 객체별 이벤트 핸들러

객체	이벤트 핸들러
Select	onBlur, onChange, onFocus
Text	onBlur, onChange, onFocus, onSelect
Textarea	onBlur, onChange, onFocus, onSelect
Button	onClick
Checkbox	onClick
Radio	onClick
Hypertext	onClick, onMouseOver, onMouseOut
Reset	onClick
Submit	onClick
Document	onLoad, onUnload, onError
Window	onLoad, onUnload, onBlur, onFocus
Frameset	onBlur, onFocus
Form	onsubmit, onReset
Image	onLoad, onError, onAbort

이벤트와 이벤트 핸들러

◆ 이벤트 핸들러

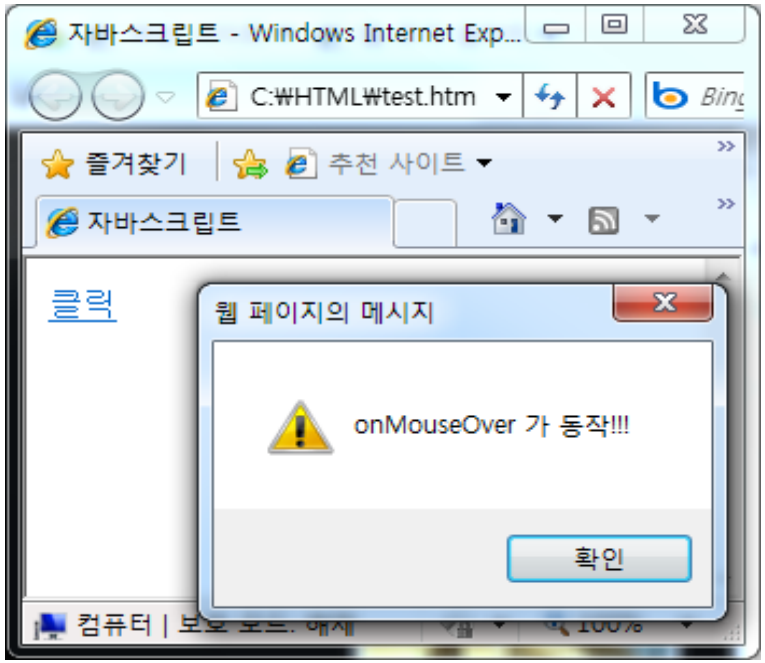
● 마우스 이벤트 예제

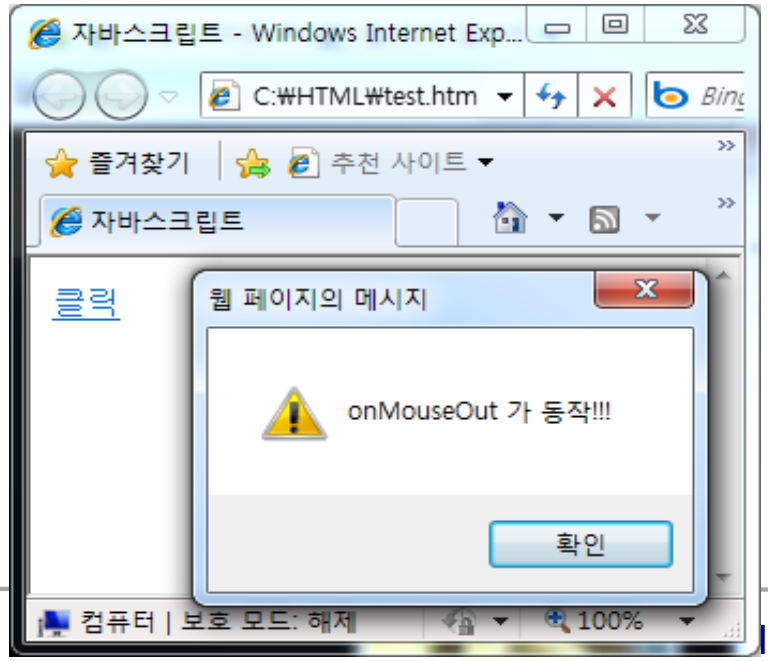
```
<HTML>
  <HEAD><TITLE>자바스크립트 </TITLE>
    <SCRIPT LANGUAGE="JavaScript">

        function mover() {
            alert("onMouseOver 가 동작!!!");
        }

        function mout() {
            alert("onMouseOut 가 동작!!!");
        }


    </SCRIPT>
  </HEAD>
  <BODY>
      <a href="javascript:함수명();"
          onmouseover="javascript:mover();"
          onmouseout="javascript:mout();">클릭</a>
  </BODY>
</HTML>
```





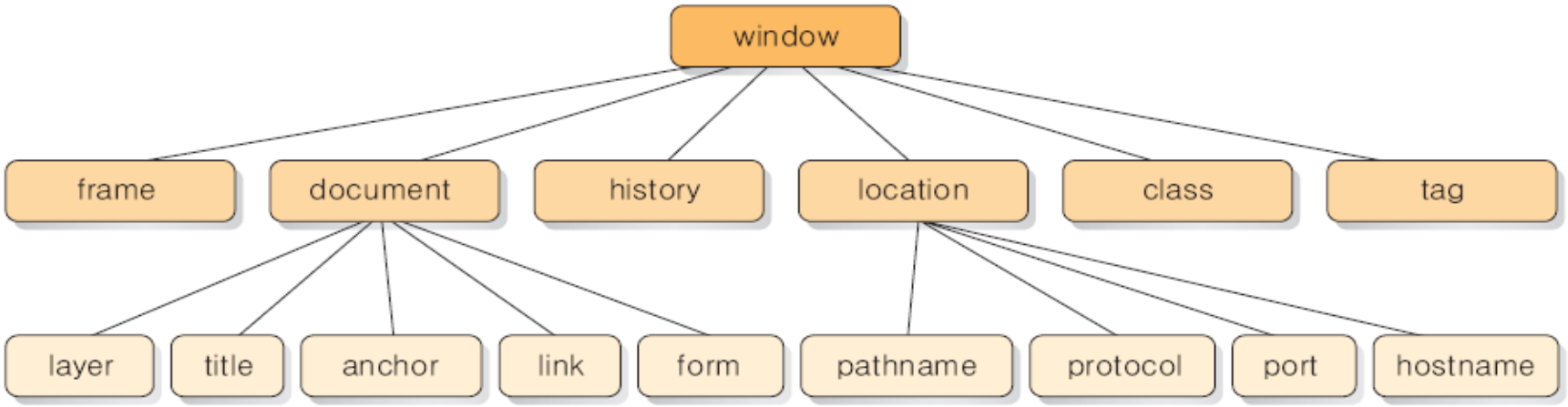
HTML 에 대한 기초 지식

1. HTML 이해
2. CSS (Cascading Style Sheet)
3. JavaScript
4. 브라우저 내장객체
5. 폼(FORM) 관련 객체
6. 레이어(Layer)와 DHTML

브라우저 내장 객체

◆ 브라우저 내장 객체란?

- 브라우저에서 지원되는 여러 가지 객체로써, 브라우저에서 표현되는 컨텐츠를 제어
- window 객체는 최상위 객체
- 여러 개의 객체를 나타낼 때는 점(.)연산자로 구분
- 내장 객체 계층 구조도



Window 객체

◆ Window 객체

- 브라우저 내장 객체 중에서 최상위에 있는 객체
- window 하위에 있는 객체를 가리킬때 window 생략 가능

```
window.속성
window.속성=값
window.메소드()
window.메소드(값)
```

- Window 객체의 속성, 메소드, 이벤트핸들러

분류	명령어	내용
속성	defaultStatus	상태 바에 나타날 기본값을 설정
	frames	윈도우 내에 들어 있는 프레임 객체
	length	윈도우 창에 있는 프레임의 수
	name	윈도우나 프레임의 이름
	opener	현재 열려 있는 윈도우

◆ Window 객체

● Window 객체의 속성, 메소드, 이벤트핸들러(계속)

속성	parent	윈도우에 계층일 경우 상위 객체
	self	현재 자신의 윈도우
	status	상태 바에 출력되는 문자열
	top	윈도우가 계층 구조일 때 최상위 객체
	windows	현재의 윈도우
	innerHeight	윈도우 안에서 내용의 높이
	innerWidth	윈도우 안에서 내용의 너비
	outerHeight	윈도우 밖의 테두리 높이
	outerWidth	윈도우 밖의 테두리 너비
	pageXOffset	윈도우에 현재 나타낸 페이지의 x 좌표
	pageYOffset	윈도우에 현재 나타낸 페이지의 y 좌표
	locationbar	윈도우의 locationbar를 나타낼지 여부
	menubar	윈도우의 메뉴 바를 나타낼지 여부
	personalbar	윈도우의 personalbar를 나타낼지 여부
	scrollbars	윈도우의 scrollbar를 나타낼지 여부
	statusbar	윈도우의 상태선을 나타낼지 여부
	toolbar	윈도우의 툴 바를 나타낼지 여부
	tags	웹 페이지에 사용된 모든 태그 정보
	classes	웹 페이지에 정의된 모드 스타일시트 클래스 정보

Window 객체

◆ Window 객체

● Window 객체의 속성, 메소드, 이벤트핸들러(계속)

메소드	alert( )	경고 메시지를 나타내는 대화창 출력
	blur( )	윈도우로부터 입력 커서를 없앰
	close( )	윈도우 닫기
	confirm( )	사용자에게 확인/취소를 선택하는 대화창 출력
	open( )	윈도우 열기
	prompt( )	사용자가 메시지를 입력함
	setTimeout( )	1000분의 1초 단위로 지정된 명령 실행
	clearTimeout( )	setTimeout을 해제
	setInterval( )	일정한 간격으로 명령을 실행
	scroll( )	특정 위치로 스크롤시킴
	find( )	윈도우 안에서 원하는 문자 검색
	print( )	윈도우 내용을 프린터로 출력
이벤트 핸들러	onBlur	윈도우에서 커서가 없어졌을 때
	onError	문서를 읽다가 error가 발생될 때
	onFocus	윈도우에 커서가 있을 때
	onLoad	브라우저로 문서가 읽힐 때
	onUnload	브라우저에서 문서가 사라질 때
	onDragDrop	브라우저에서 드래그앤 드롭시켰을 때
	onMove	윈도우를 움직였을 때
	onResize	윈도우의 크기가 변경될 때

Window 객체

◆ Window 객체의 메소드

● 새로운 창 열기(Window.open( ))

```
open("URL", "창 이름", " 창의 특성") ;
```

● 창의 특성

특성	내용
toolbar	표준 툴 바 메뉴를 생성
location	문서의 위치 정보 생성
directories	디렉토리 버튼을 생성
status	상태 표시줄을 생성
menubar	메뉴 바를 생성
scrollbars	스크롤 바를 생성
resizable	윈도우의 크기를 사용자가 조절
copyhistory	현재 윈도우의 히스토리 정보를 복사
width	픽셀수를 기준으로 윈도우의 폭을 지정
height	픽셀수를 기준으로 윈도우의 높이를 지정

Window 객체

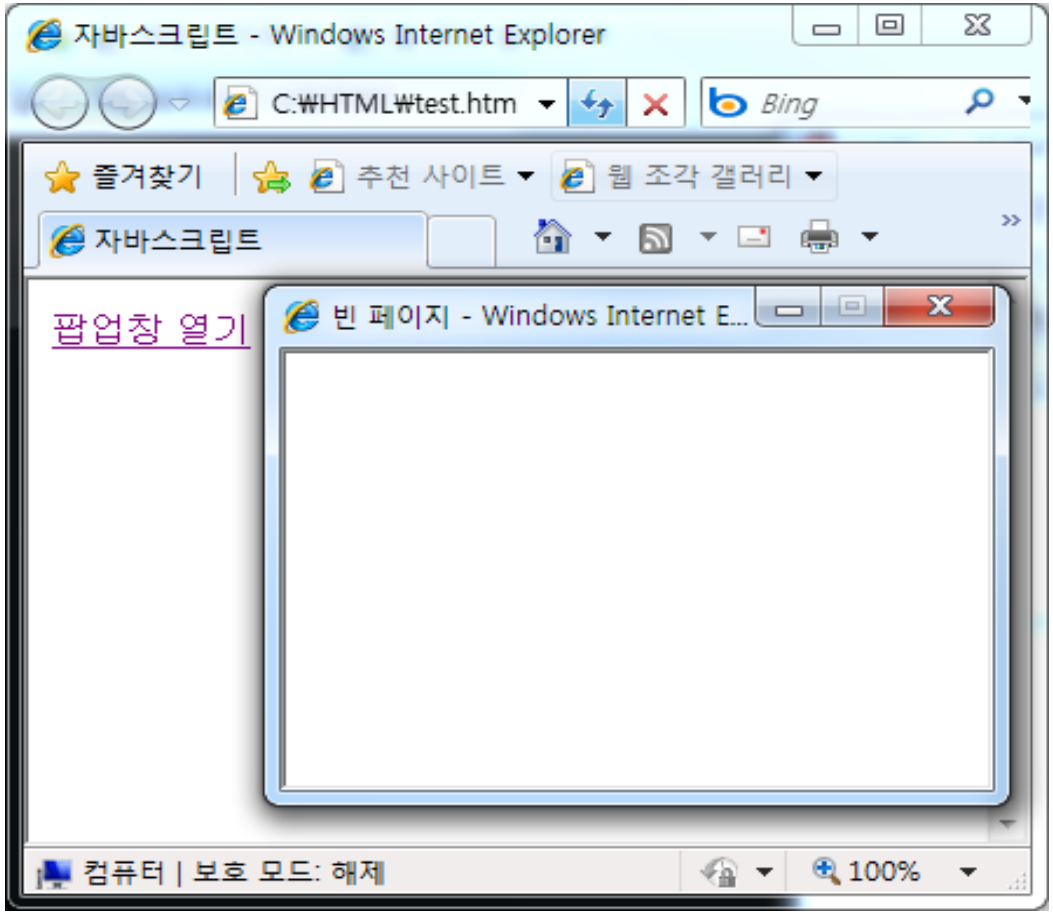
◆ Window 객체의 메소드

● 새로운 창 열기(Window.open( ))

– 예제

```
<HTML>
  <HEAD><TITLE>자바스크립트 </TITLE>
    <SCRIPT LANGUAGE="JavaScript">

        function wingen(){
            window.open("","win1","width=300,
              height=200, toolbar=no, location=no,
              toolbar=no, location=no, menubar=no,
              scrollbars=no, resizable=no");
        }
    </SCRIPT>
  </HEAD>
  <BODY>
      <a href="javascript:wingen();" >팝업창 열기</a>
  </BODY>
</HTML>
```



Window 객체

◆ Window 객체의 메소드

● 새로운 창 열기(Window.open( ))

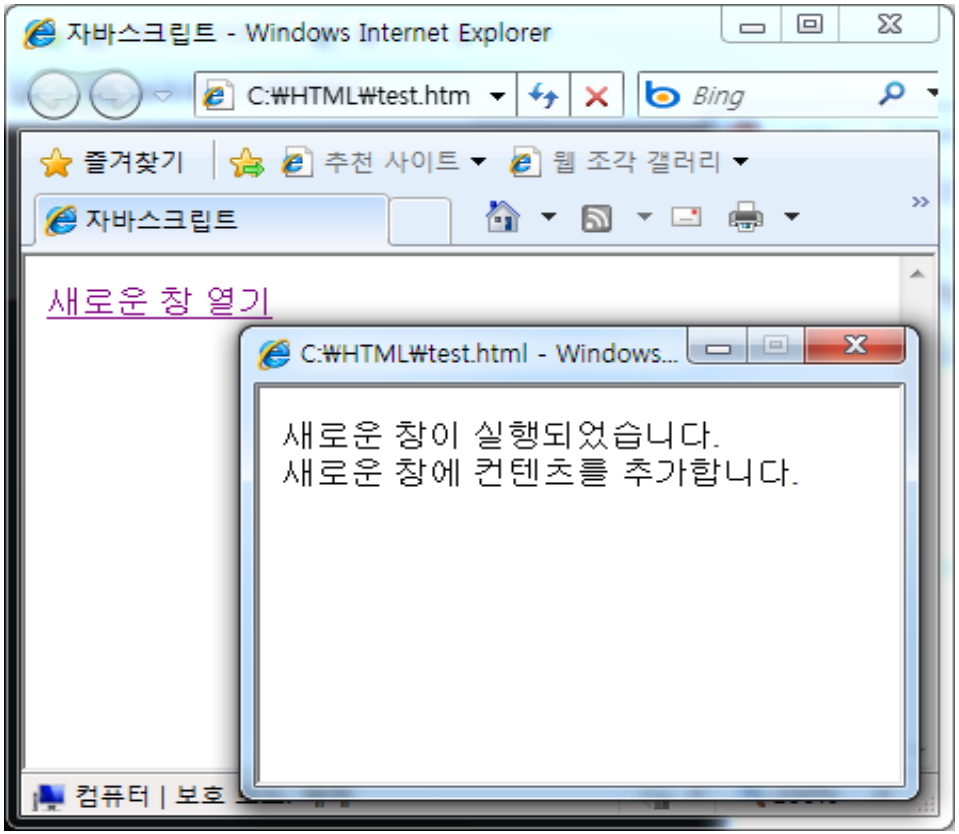
– 예제 : 새로운 창에 컨텐츠 추가

```
<HTML>
  <HEAD><TITLE>자바스크립트 </TITLE>
    <SCRIPT LANGUAGE="JavaScript">

        function wingen(){
            var newWin1 = window.open("","win1","width=300,
              height=200, toolbar=no, location=no,
              toolbar=no, location=no, menubar=no,
              scrollbars=no, resizable=no");

            var contents = "새로운 창이 실행되었습니다.<br>";
            contents = contents + "새로운 창에 컨텐츠를 추가합니다.";

            newWin1.document.write(contents);
        }
    </SCRIPT>
  </HEAD>
  <BODY>
      <a href="javascript:wingen();" >새로운 창 열기</a>
  </BODY>
</HTML>
```



Window 객체

◆ Window 객체의 메소드

● 새로운 창 닫기(Window.close( ))

```
window.close()
self.close()
this.close()
```

● 특정 창 열고 닫기

```
객체명=window.open("","","") // 열기
객체명.close() // 닫기
```

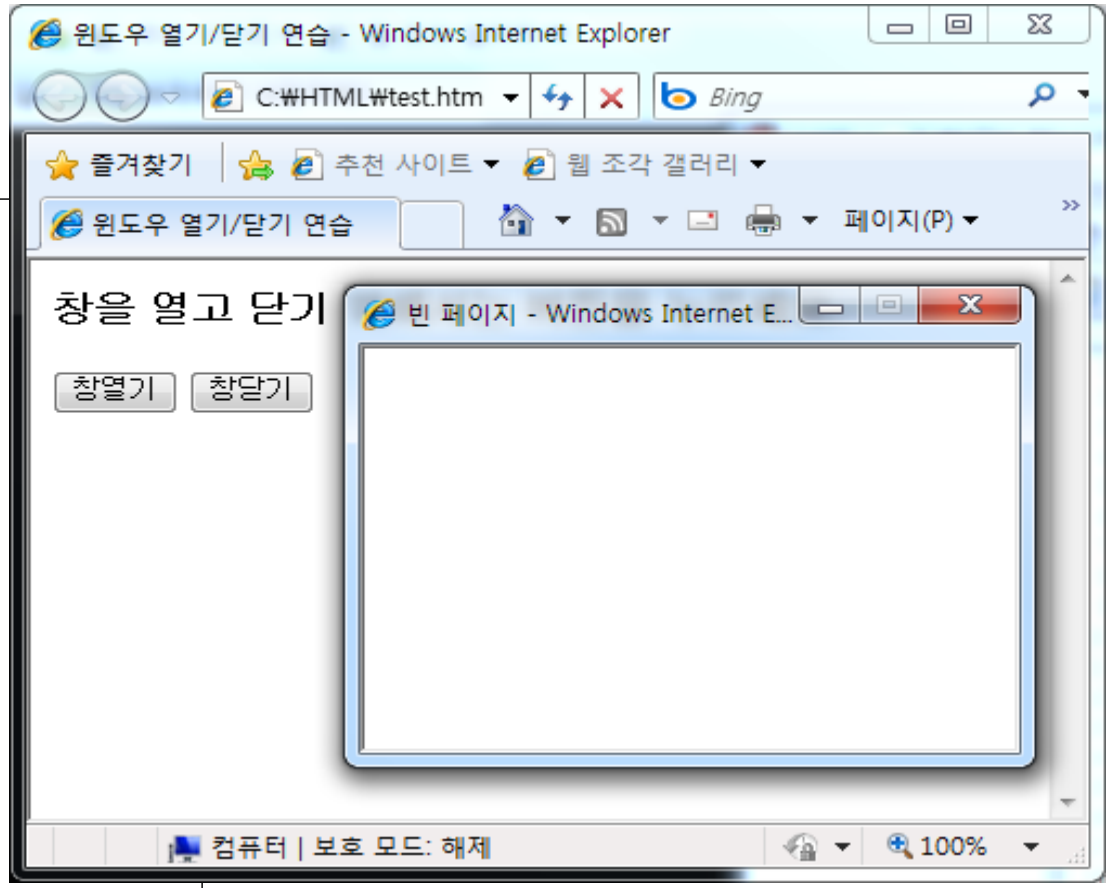
Window 객체

◆ Window 객체의 메소드

● 새로운 창 닫기(Window.close( )) : 예제

```
<HTML><HEAD>
<TITLE>윈도우 열기/닫기 연습</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
  var newwin = null
  function Winopen() {
    newwin = window.open("", "", "width=300,
                                  height=200,menubar=yes")
  }

  function Winclose() {
    if(newwin != null)
      newwin.close()
  }
</SCRIPT>
</HEAD>
<BODY>
    <H3>창을 열고 닫기 위해서는 버튼을 누르세요.</H3>
    <FORM>
        <INPUT TYPE="button" VALUE="창열기" onClick="Winopen()">
        <INPUT TYPE="button" VALUE="창닫기" onClick="Winclose()">
    </FORM>
</BODY></HTML>
```



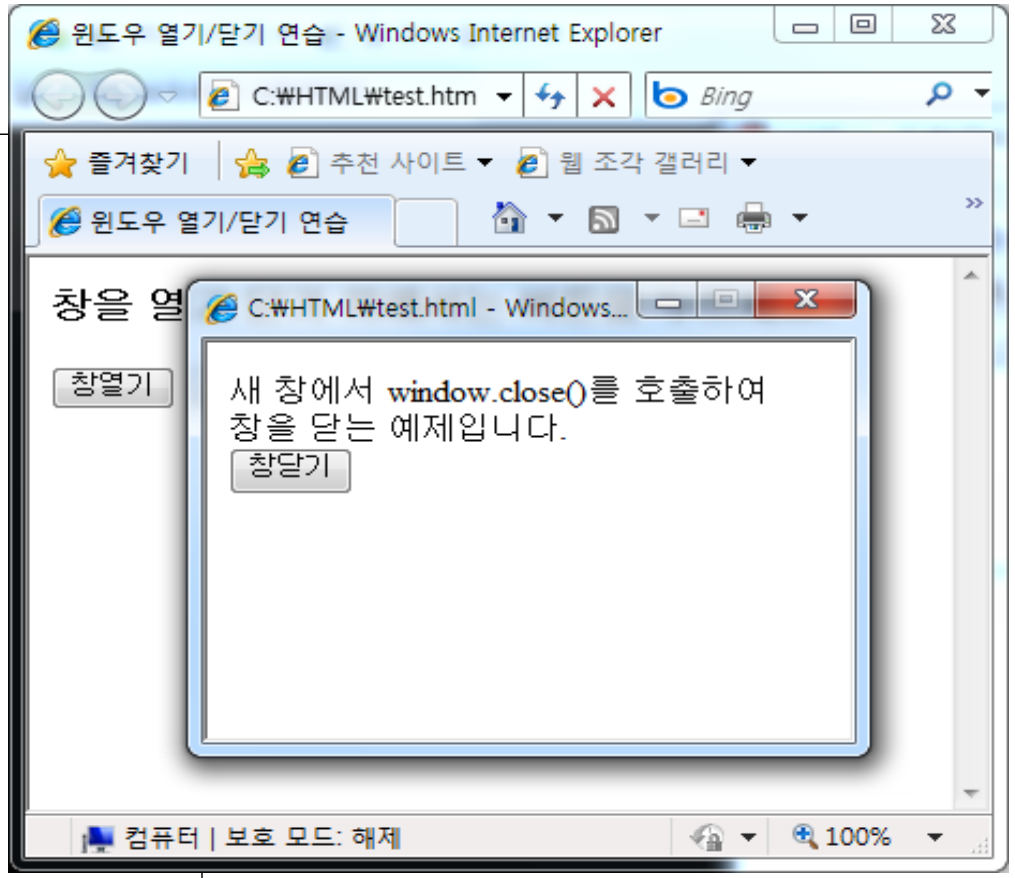
Window 객체

◆ Window 객체의 메소드

● 새로운 창 닫기(Window.close( )) : 예제

```
<HTML><HEAD>
<TITLE>윈도우 열기/닫기 연습</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
  var newwin = null
  function Winopen() {
    newwin = window.open("", "", "width=300,height=200,
                                  menubar=yes");
    var m = "새 창에서 window.close()를 호출하여<br>";
    m = m + "창을 닫는 예제입니다.</br>";
    newwin.document.write(m);
     newwin.document.write("<input type='button' value='창닫기'
        onclick='javascript:window.close()'>");
  }

</SCRIPT>
</HEAD>
<BODY>
    <H3>창을 열고 닫기 위해서는 버튼을 누르세요.</H3>
    <FORM><INPUT TYPE="button" VALUE="창열기" onClick="Winopen()">
          <INPUT TYPE="button" VALUE="창닫기" onClick="Winclose()">
    </FORM>
</BODY></HTML>
```



Location 객체

◆ Location 객체의 기본 사용법

- 현재 열려진 윈도우의 URL 주소에 관한 정보를 제공하는 객체
- frame 으로 나누ㅜ어져 있다면 최상위에 있는 문서의 URL 주소를 갖게 됨
- Location 객체 형식

```
windows.location.속성
 location.속성=값
 location.메소드()
```

- **속성**
  - href : 문서의 URL 주소
  - port : 포트번호
  - host : URL 주소의 호스티이름과 포트번호
  - hostname : URL 주소의 호스트이름
- **메소드**
  - reload() : 브라우저의 현재문서를 다시 읽도록 하는 메소드
  - replace() : 현재 브라우저의 특정 URL 에 있는 문서로 바꾸어주는 메소드

Location 객체

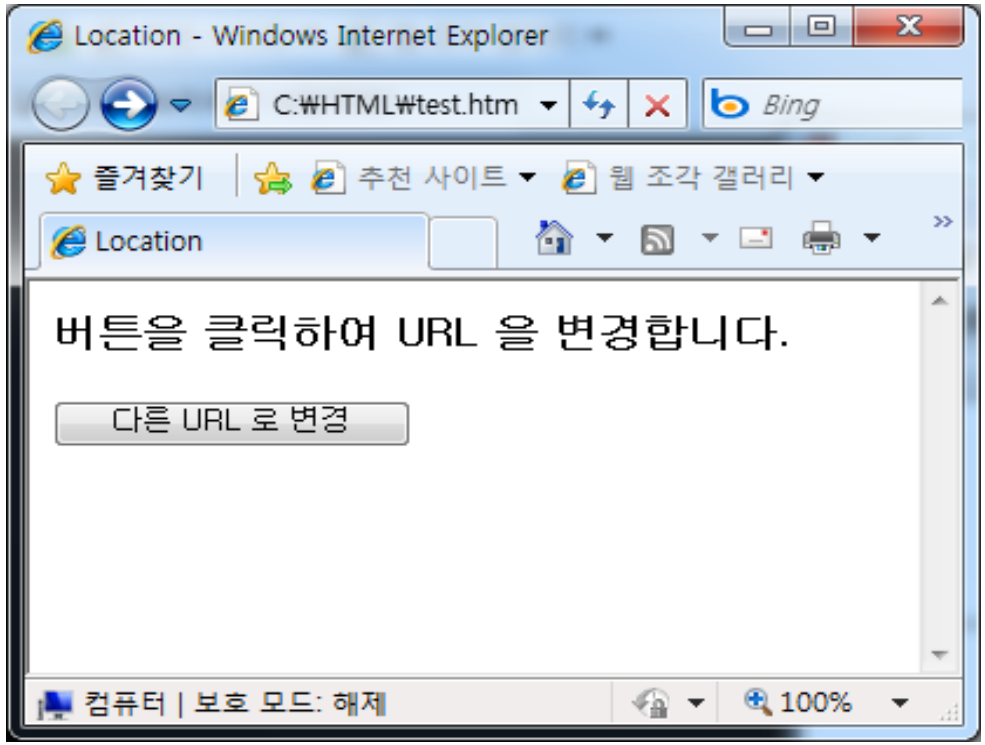
◆ Location 객체의 기본 사용법

● Location 객체 형식

```
windows.location.속성
 location.속성=값
 location.메소드()
```

```
<HTML><HEAD>
<TITLE>Location</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
  function changeLocation() {
     window.location.href="http://www.ktds.com";
  }

</SCRIPT>
</HEAD>
<BODY>
    <H3>버튼을 클릭하여 URL 을 변경합니다.</H3>
    <FORM><INPUT TYPE="button" VALUE="다른 URL 로 변경"
                 onClick="changeLocation()">
    </FORM>
</BODY></HTML>
```



History 객체

◆ History 객체의 기본 사용법

- 브라우저의 히스토리 리스트 정보를 저장해 두는 곳
- 히스토리 정보는 브라우저가 최근 방문했던 url 주소를 의미
- history 객체 형식

```
windows.history.속성
history.메소드()
history.메소드(값)
```

- 속성
  - length : 히스토리 리스트에 포함되어 있는 URL 주소의 개수
- 메소드
  - back() : 히스토리 리스트에서 한단계 뒤로 이동
  - forward() : 히스토리 리스트에서 한단계 앞으로 이동
  - go() : 히스토리 리스트에서 임의의 위치로 이동

Document 객체

◆ Document 객체의 기본 사용법

- 브라우저 내장 객체 중에서 window 객체 아래에 위치
- html 문서의 body 태그 안에 있는 내용을 지정하고 있음
- Document 객체는 이미 선언 되어 있으므로 따로 선언하지 않고 사용

```
document.속성
document.메소드()
```

◆ Document 주요 속성

- location : 문서의 URL 위치
- title : 문서의 제목
- layers : 문서에 있는 레이어들의 배열
- embeds : 문서에 있는 플러그인들의 배열
- applets : 문서에 있는 자바 애플릿의 배열
- images : 문서에 있는 이미지들의 배열

Document 객체

◆ Document 주요 속성(계속)

- cookie : 클라이언트 PC 에 저장한 정보
- links : 문서에 있는 링크들의 배열
- forms : 문서에 있는 입력양식의 배열
- anchors : 문서에 있는 표식들의 배열
- vlinkColor : 이전에 방문했던 링크를 표시하는 색
- alinkColor : 링크를 클릭했을 때 나타내는 색
- linkColor : 링크를 표시하는 색
- fgColor : 문서의 전경색
- bgColor : 문서의 배경색
- referer : 링크로 현재문서에 왔을 때 이전 문서의 URL 위치정보
- lastModified : 문서를 마지막으로 수정한 날짜

Document 객체

◆ Image 객체

- 이미지에 대한 정보를 제공하는 객체

◆ Image 주요 속성

- name : 이름
- src : 이미지 파일 위치
- lowsrc : 이미지 파일 위치
- height : 이미지의 높이 크기
- width : 이미지의 너비 크기
- border : 이미지 테두리 크기
- hspace : 이미지 가로 여백
- vspace : 이미지 세로 여백

Image 객체

◆ Document 주요 메소드

- open() : 문서에 데이터를 출력 시키기 위한 준비
- close() : 문서에 데이터를 출력 시키는 것을 마무리
- write() : 문서에 데이터를 출력
- writeln() : 문서에 데이터를 출력한 후 줄을 바꿈
- clear() : 브라우저에서 문서 지우기
- getSelection() : 현재 선택된 문자열을 리턴

Links 객체

◆ Links 객체

- HTML 문서에 들어있는 모든 링크에 대한 정보를 제공하는 객체
- document 객체의 하위 객체

◆ Links 주요 속성

- target : 정보를 보여줄 윈도우나 프레임
- href : 문서의 URL 주소
- port : 포트 번호
- host : URL 주소의 호스트이름과 포트번호
- hostname : URL 주소의 호스트 이름
- protocol : 프로토콜 종류
- pathname : 디렉토리 위치

HTML 에 대한 기초 지식

1. HTML 이해
2. CSS (Cascading Style Sheet)
3. JavaScript
4. 브라우저 내장객체
5. 폼(FORM) 관련 객체
6. 레이어(Layer)와 DHTML

Form 객체의 입력 양식

◆ Form(폼) 객체

- 사용자의 입력 사항을 입력 받는 객체
- Form 의 하위 객체의 값을 서버로 전송하기 위해서 사용

◆ Form 객체의 형식

```
<FORM [NAME = "양식 이름"]
          [TARGET="출력 창"]
          [ACTION="CGI_URL"]
          [METHOD= GET | POST]
          [ENCTYPE="MIME_TYPE"]
          [onSubmit="처리할 이벤트 핸들러"]>
 </FORM>
```

Form 객체의 입력 양식

◆ Form 객체의 속성

- target : 데이터를 서버로 보낸 후 받아보는 결과를 어느 윈도우에서 받아볼 것인지를 지정
- action : 입력한 데이터를 처리할 서버의 URL 지정
- method : 데이터를 어떤 HTTP 프로토콜로 보낼 것인지를 지정(GET/POST)
- encoding : 서버로 보내질 데이터이의 인코딩을 지정
- elements : Form 태그 안에 있는 모든 입력 양식을 배열로 저장하고 있는 속성

◆ Form 객체의 메소드

- submit() : 입력된 데이터를 서버로 전송하는 메소드
- reset() : 입력된 데이터를 리셋시키는 메소드

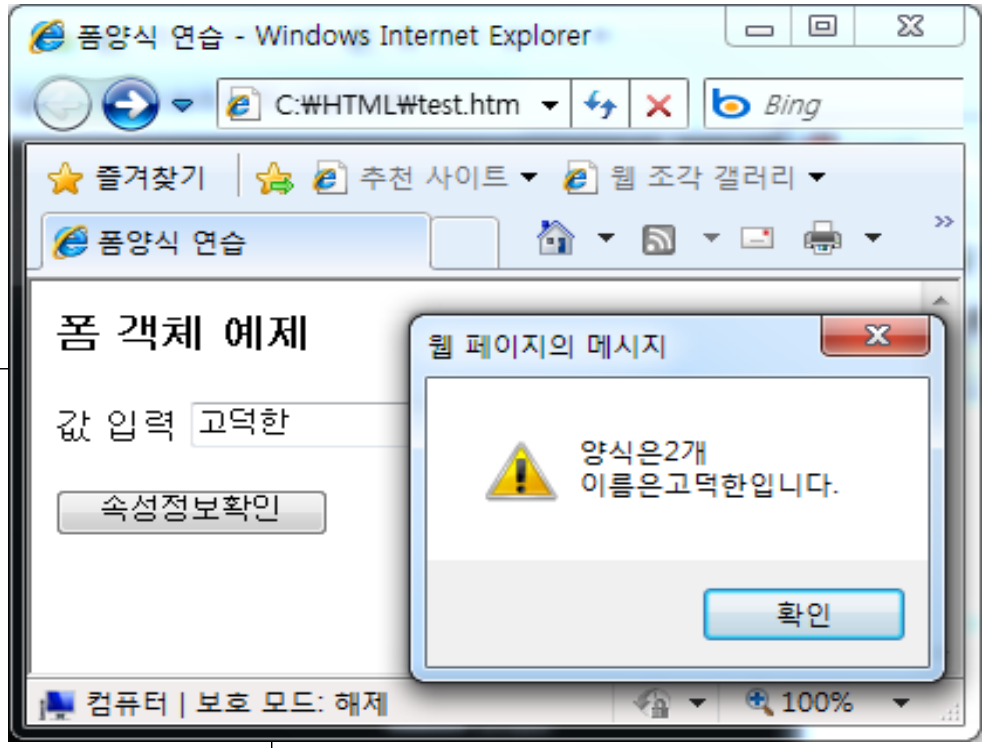
◆ Form 객체의 이벤트 핸들러

- onSubmit : 데이터를 서버로 보내는 명령이 내려졌을 때 실행되는 이벤트핸들러
- onReset : 리셋 버튼을 눌렀을 때 실행되는 이벤트 핸들러

Form 객체의 입력 양식

◆ 폼 객체의 속성

- Form 을 활용한 간단한 예제




```
<HTML><HEAD><TITLE>폼양식 연습</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
  function view(){
    var name = document.mm.elements[0].value;
    if(name == ""){
      alert("값을 입력하세요.");}
    else{
      tt = "양식은" + document.mm.elements.length + "개 ₩n";
      tt += "이름은" + document.mm.elements[0].value + "입니다.";
      alert(tt);}
  }
</SCRIPT></HEAD>
<BODY><H3> 폼 객체 예제 </H3>
<FORM NAME="mm">값 입력
  <INPUT TYPE="text"><br><br>
  <INPUT TYPE="button" VALUE="속성정보확인" onClick="view();">
</FORM></BODY></HTML>
```

Input 객체

◆ Input 객체

- 사용자의 입력 값을 처리하기 위한 다양한 형식을 포함한 객체
- <INPUT> 태그 형식

```
<INPUT [TYPE="Input-type"]
       [NAME="Input-name"]>
 </INPUT>
```

- <Input>태그의 type 종류

TYPE 형태	내용
text	텍스트 형태의 입력 양식 제공
password	입력한 문자를 * 형태로 제공
radio	라디오 버튼 형태의 입력 양식 제공
checkbox	체크 박스 모양의 입력 양식 제공
button	버튼 모양의 입력 양식 제공
file	사용자가 파일을 선택할 수 있게 제공
hidden	숨겨진 입력 양식 제공
reset	리셋 형태의 입력 양식 제공
submit	전송 입력 양식 제공

Input 객체

◆ Text 객체

● 한줄짜리 문자열을 입력 받기 위한 형식

```
<INPUT TYPE="text"
        NAME="텍스트 입력 양식의 이름"
        [VALUE="텍스트 입력 양식에 기본 문자열"]
        [SIZE="텍스트 입력 양식의 크기 기본은 20"]
        [MAXLENGTH="사용자가 최대로 입력한 글자수"]
        [onChange="이벤트 핸들러"]
        [onBlur="이벤트 핸들러"]
        [onFocus="이벤트 핸들러"]
        [onSelect="이벤트 핸들러"] >
```

● text type 주요 속성

– name : 이름

– value : 입력 양식에 입력한 값

– defaultValue : 처음 디폴트로 value 에 입력된 값

– size : 입력창의 크기

Input 객체

◆ Password 객체

● 글 내용을 ** 표시로 나타내는 한 줄짜리 문자열을 입력 받기 위한 형식

```
<INPUT TYPE="password"
        NAME="패스워드 입력 양식 이름"
        [VALUE ="패스워드 입력 양식에 지정하고 싶은 문자열"]
        [SIZE = "패스워드 입력 양식의 크기"]>
```

● password type 주요 속성

– name : 이름

– value : 암호 입력 양식에 입력한 값

– size : 입력창의 크기

Input 객체

◆ Radio 객체

● 여러 개의 항목 중에서 하나를 선택하기 위한 형식

```
<INPUT TYPE="radio"
    NAME="라디오 버튼 그룹 이름"
    VALUE="라디오 버튼에 할당할 값"
    [CHECKED]
    [onClick="선택할 경우 이벤트 핸들러"]> 라디오버튼에 나타나는 문자열
```

● radio type 주요 속성

– name : 이름

– value : Radio 에 지정된 값

– checked : 해당 항목이 선택되었는지를 판단

Input 객체

◆ Checkbox 객체

● 여러 개의 항목 중에서 여러 개를 선택하기 위한 형식

```
<INPUT TYPE="checkbox"
      NAME="체크박스 양식의 이름"
      VALUE="체크박스에 할당할 값"
      [CHECKED]
      [onClick="선택할 경우 이벤트 핸들러"]> 체크박스에 나타나는 문자열
```

● checkbox type 주요 속성

– name : 이름

– value : checkbox에 지정된 값

– checked : 해당 항목이 선택되었는지를 판단

Input 객체

◆ Button 객체

● 여러 개의 항목 중에서 여러 개를 선택하기 위한 형식

```
<INPUT TYPE="button"
        NAME="버튼 양식의 이름"
        VALUE="버튼 위에 쓰여질 문자열"
        [onClick="버튼을 선택할 경우 이벤트 핸들러"]>
```

● button type 주요 속성

– name : 이름

– value : 버튼에 쓰여질 문자열

Input 객체

◆ File 객체

- 파일을 전송하기 위해서 파일을 선택하는 창을 보여준다

```
<INPUT TYPE="file"
        NAME="파일 업로드 양식의 이름 지정">
```

Input 객체

◆ Hidden 객체

● 브라우저에 화면으로 나타내지 않고, 값을 서버로 전송하기 위해 사용하는 형식

```
<INPUT TYPE="hidden"
        NAME="이름" VALUE="값">
```

Input 객체

◆ Reset 객체

● 브라우저에 화면으로 나타내지 않고, 값을 서버로 전송하기 위해 사용하는 형식

```
<INPUT TYPE="reset"
        NAME="리셋 버튼 이름"
        VALUE="리셋 버튼 위에 찍히는 문자값"
        [onClick="선택할 경우 이벤트 핸들러"]>
```

● reset 객체 주요 속성

– name : 이름

Input 객체

◆ Submit 객체

● <form> 태그에 포함된 입력 값들을 서버로 전송하는 객체

```
<INPUT TYPE="submit"
        NAME="보내기 양식의 이름"
        VALUE="보내기 버튼 위에 찍히는 문자열"
        onClick="클릭하면 수행되는 이벤트 핸들러">
```

● submit 객체 주요 속성

– name : 이름

Select 객체

◆ Select 객체

- 여러 항목을 목록으로 나열하여 선택하기 위한 형식

◆ Select 객체의 기본 사용법

- option : 목록 열거

```
<SELECT NAME="SELECT 입력 양식 이름"
                [SIZE="양식 크기"]
                [MULTIPLE]
                [onFocus="이벤트 핸들러"]
                [onBlur="이벤트 핸들러"]
                [onChange="이벤트 핸들러"]
<OPTION [SELECTED] [VALUE="목록에 설정할 값"]> 목록
  ..................
</SELECT>
```

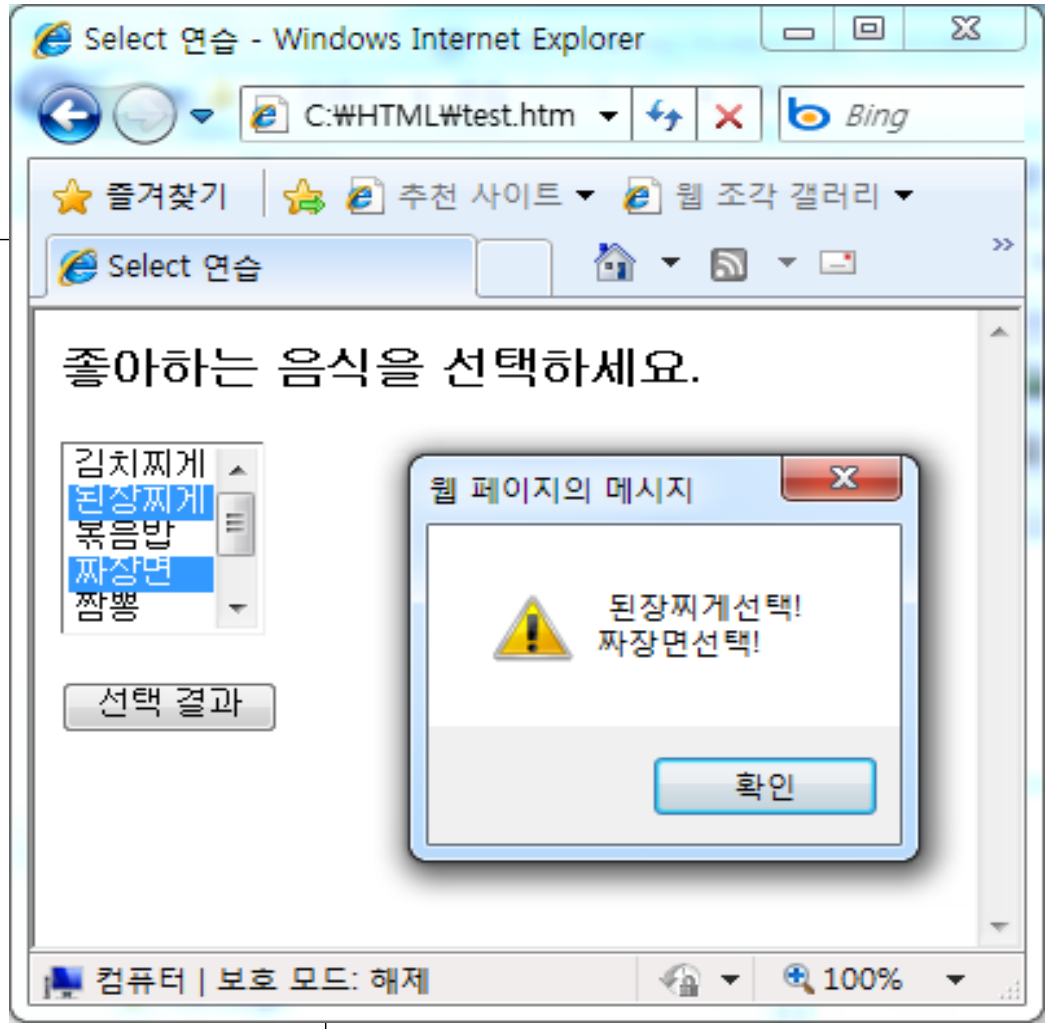
- selected목록이 선택된 상태로 만듦
- multiple : 목록을 중복 선택 여부를 지정
- size : 보여지는 목록 개수

Select 객체

◆ Select 객체

● 예제

```
<HTML><HEAD><TITLE>Select 연습</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
  function resultmus(form) {
    var m = " "
    for(i = 0; i < form.food.length; i++) {
      if(form.food.options[i].selected == true){
      m += form.food.options[i].text+ "선택!₩n"
    } }
    alert(m)
  }
</SCRIPT></HEAD>
<BODY><H3> 좋아하는 음식을 선택하세요.</H3><P>
<FORM><SELECT name="food" size=5 multiple>
  <option value="김치찌게"> 김치찌게
  <option value="된장찌게" selected> 된장찌게
  <option value="볶음밥"> 볶음밥
  <option value="짜장면" selected> 짜장면
  <option value="짬뽕"> 짬뽕
  <option value="냉면"> 냉명
  <option value="칼국수"> 칼국수
</SELECT><p>
  <INPUT TYPE="button" VALUE="선택 결과" onClick="resultmus(this.form);">
</FORM></BODY></HTML>
```



Textarea 객체

◆ Textarea 객체

- 여러 줄의 텍스트를 입력하기 위한 객체

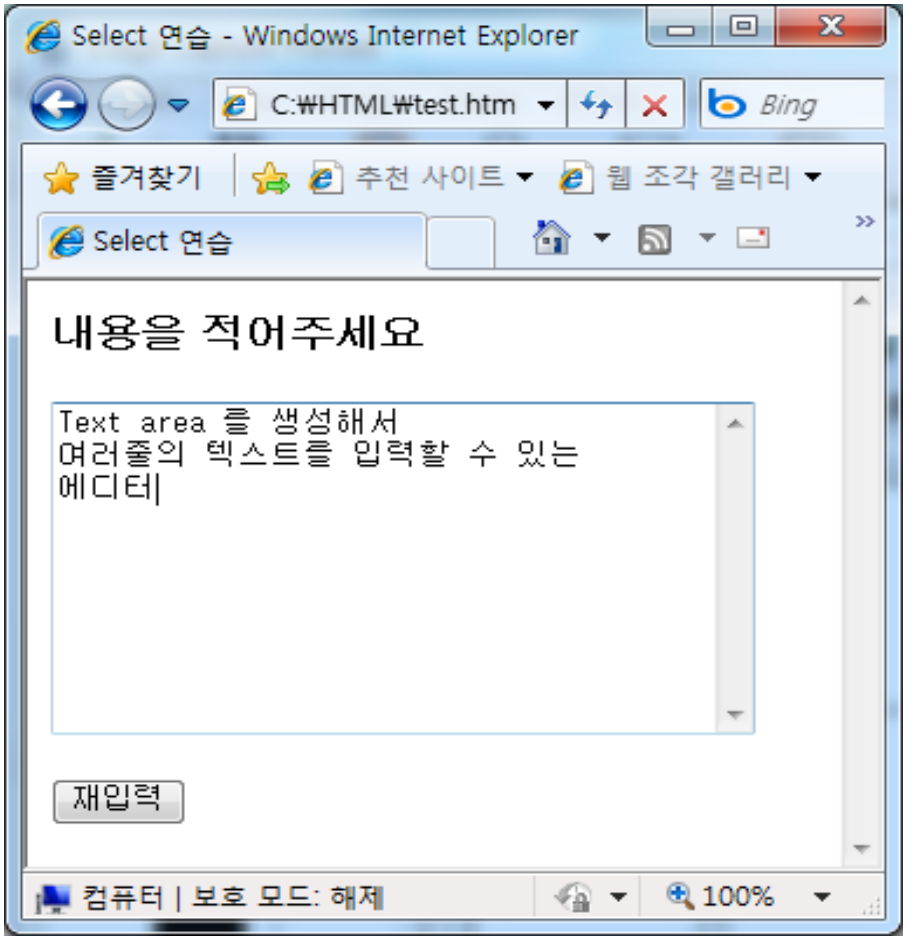
```
<TEXTAREA NAME="텍스트 영역 입력 양식의 이름"
          [ROWS="텍스트 영역 입력 양식의 가로"]
          [COLS="텍스트 영역 입력 양식의 세로"]
          [WRAP="off" |"virtual" |"physical"]
          [onChange="이벤트 핸들러"]
          [onBlur="이벤트 핸들러"]
          [onFocus="이벤트 핸들러"]
          [onSelect="이벤트 핸들러"] >
           텍스트 영역에 쓰일 기본 문자열
</TEXTAREA>
```

Textarea 객체

◆ Textarea 객체

● 예제

```
<HTML><HEAD><TITLE>Select 연습</TITLE>
</HEAD>
<BODY><H3> 내용을 적어주세요</H3><P>
<FORM>
   <TEXTAREA rows="10" cols="40"></TEXTAREA><p>
   <input type="reset" value="재입력">
 </FORM>
</FORM></BODY></HTML>
```



HTML 에 대한 기초 지식

1. HTML 이해
2. CSS (Cascading Style Sheet)
3. JavaScript
4. 브라우저 내장객체
5. 폼(FORM) 관련 객체
6. 레이어(Layer)와 DHTML

DHTML 개요

◆ Dynamic HTML

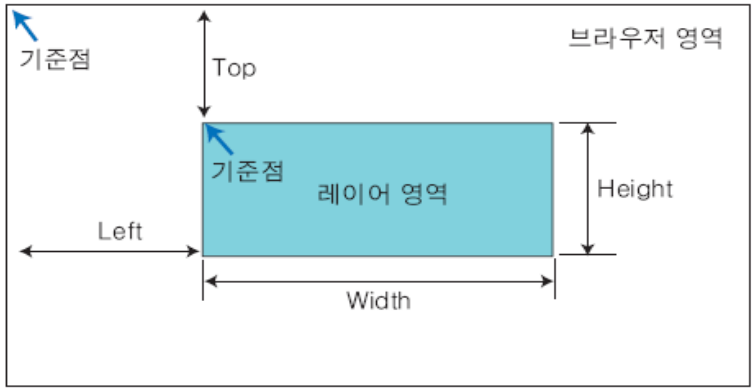
- **HTML 을 동적으로 처리**
- **DHTML = HTML + CSS + JavaScript**

- **문서 객체 모델(DOM)**
  - 웹 페이지 구성 요소를 처리하기 위하여 내장 객체와 브라우저에 표현
- **스크립트 언어(JavaScript)**
  - 스크립트는 명령어만 사용해서 편리하지만 복잡한 응용 프로그램을 만들기 어려움
- **스타일시트(CSS)**
  - 웹 페이지의 통일감과 일관성을 유지하고 제작시간과 문서의 용량이 줄어듬
- **레이어(Layer)**
  - 문자, 그림, 테이블을 원하는 위치에 겹치게 함

레이어(Layer) 객체

◆ 레이어 객체 선언

- 레이어 객체는 와 태그를 사용
- 레이어 사용 좌표



- 레이어 사용 형식

```
<style>
   #레이어 이름 {position:absolute;..... : }
</style>

<div id="레이어 이름">
   텍스트 또는 이미지
</div>
```

레이어(Layer) 객체

◆ 레이어 객체의 속성과 메소드

속성	의미
name	레이어의 이름
width	레이어의 가로 크기 픽셀
height	레이어의 세로 크기 픽셀
left	브라우저 영역 기준점에서 레이어가 위치할 x 좌표
top	브라우저 영역 기준점에서 레이어가 위치할 y 좌표
visibility	레이어가 표시될지 여부를 지정
pageX	레이어가 위치할 x 좌표로서 기준은 항상 전체 화면으로 계산
pageY	레이어가 위치할 y 좌표로서 기준은 항상 전체 화면으로 계산
X	화면의 왼쪽을 기준으로 한 레이어의 위치
Y	화면의 위쪽을 기준으로 한 레이어의 위치
clip	레이어가 보여질 위치 지정 left, top, right, bottom, width, height 값을 지정
zindex	레이어의 실행 순서 값 지정, 클수록 위에 표시
src	레이어에 불러들일 외부 문서

레이어(Layer) 객체

◆ 레이어 객체의 속성과 메소드

bgcolor	레이어의 배경색을 지정하는 기능
background	레이어의 배경 이미지를 지정하는 기능
parentLayer	현재 레이어의 상위 레이어의 이름을 알려주는 기능을 가지는 속성으로, 상위 레이어가 없는 경우 window 객체를 의미
above	현재 레이어의 상위 레이어 정보 제공
below	현재 레이어의 하위 레이어 정보 제공
siblingAbove	특정 레이어에 포함된 두 개의 레이어 중 위쪽 레이어
siblingBelow	특정 레이어에 포함된 두 개의 레이어 중 아래쪽 레이어

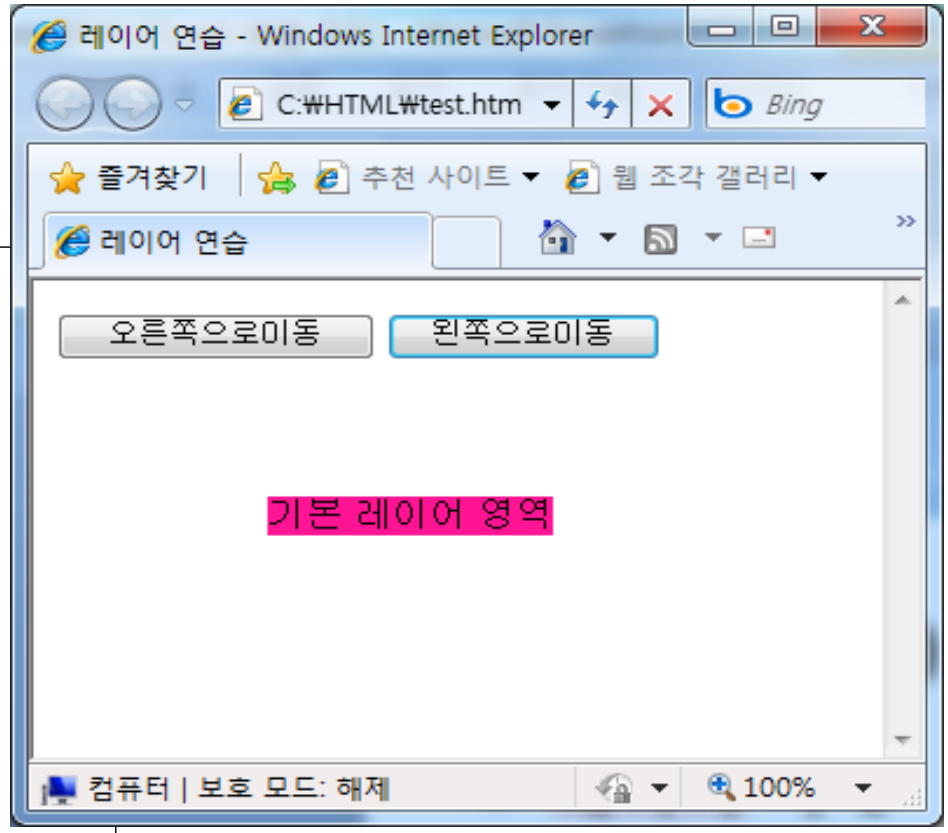
메소드	의미
moveBy(x, y)	레이어의 위치를 상대값만큼 변경
moveTo(x, y)	레이어가 속한 상위 레이어 기준으로 이동
moveToAbsolute(x, y)	화면 전체를 기준으로 레이어 이동
moveToAbove(레이어)	지정한 레이어 위로 이동
moveBelow(레이어)	지정한 레이어 아래로 이동
resizeBy(x, y)	레이어의 크기를 상대적인 크기로 변경
resizeTo(x, y)	레이어의 크기를 절대적인 크기로 변경
load(src, width)	레이어에 나타날 문서를 지정한 넓이로 호출
clipTo(레이어, t, r, b, l)	레이어에 top, right, bottom, left 좌표로 오려냄
clipBy(레이어, t, r, b, l)	레이어에 top, right, bottom, left 크기만큼 오려냄

레이어(Layer) 객체

◆ 레이어 객체

● 간단한 예제

```
<HTML><HEAD><TITLE>레이어 연습</TITLE>
<SCRIPT LANGUAGE="javascript">
  function R() {
    var loc=parseInt(document.all["myLay"].style.left);
    document.all["myLay"].style.left=loc+150;
  }
  function L() {
    var loc=parseInt(document.all["myLay"].style.left);
    document.all["myLay"].style.left=loc-150;
  }
</SCRIPT></HEAD>
<BODY><FORM>
  <INPUT type=button value="오른쪽으로이동" onClick="R()">
  <INPUT type=button value="왼쪽으로이동" onClick="L()"><br>
</FORM>
  <DIV id="myLay" style="position:absolute;top:100px;
left:100px; background-color: #ff1493;">
기본 레이어 영역 </DIV>
</BODY></HTML>
```



레이어(Layer) 객체

◆ 레이어와 스타일시트 활용

- 레이어는 고유한 기름으로 구별
- 레이어 사용 형식

```
<STYLE>
 #ID 이름 {position : 속성값(absolute, relative);
       위치 속성(left, top) : 속성값(pt, px, in, cm)
       크기 속성(width, height) : 속성값(pt, px, in, cm)
       }
</STYLE>
```

- absolute : 레이어의 문서 전체에 대한 절대적 위치 지정
- relative : 레이어의 현재 위치를 기준으로 지정

레이어(Layer) 객체

◆ 레이어를 활용한 웹 페이지 작성

- 자바스크립트를 이용하여 레이어 보기와 감추기와 위치 이동이 가능
- 레이어 감추고 보이기 형식

```
(레이어 이름).style.visibility="visible"
(레이어 이름).style.visibility="hidden"
```

실전 예제 프로그램 작성

1. 예제 프로그램 작성

실습예제

◆ 아래 그림과 같이 HTML 을 디자인