



javascript 1강

양 명 속

[now4ever7@gmail.com]



목차

- 자바스크립트란
- 기본문법
 - 변수, 자료형, 연산자, 제어문 - 조건문, 반복문
 - 배열
 - 함수
 - 사용자 정의 함수
 - 내장함수
- 자바 스크립트에서 html 객체 접근
- DOM



자바스크립트란

- 선마이크로시스템즈와 넷스케이프에서 공동으로 개발한 클라이언트쪽 스크립트 언어
- html문서에 소스가 삽입된 형태로 실행되는 웹 브라우저 내에서만 돌아가는 스크립트 언어
- HTML에 포함되어서 웹 페이지에 **동적인 효과**를 주거나 사용자가 입력한 데이터의 **유효성 검사**를 하기 위해서 사용됨
 - html문서 내에서 마우스 클릭, 양식 입력, 페이지 이동, 사용자의 데이터입력 오류 등 **사용자의 이벤트를 확인하여 이에 대한 적절한 행동을 네트워크를 통하지 않고** 취하도록 하는데 많이 사용됨
 - 버튼 클릭에 반응하는 경고창 제작, 콤보상자를 이용한 사이트의 이동 및 선택, 즐겨찾기 추가 등



서버쪽 / 클라이언트쪽 스크립트

■ 웹 스크립트 언어

- 컴파일되어 실행파일의 형태로 작동하는 것이 아니라, **사용자의 요청이 있을 때마다 각 언어 해석기에 의해 프로그램이 실행되는** 스크립트로 이루어지는 인터프리터 언어의 형태가 대부분임
- **한줄씩 해석되고 가공** – 페이지를 실행할 때마다 동작하는 방식이므로 컴파일된 언어보다 처리속도가 느림
- Client Side Script, Server Side Script

■ Server Side Script

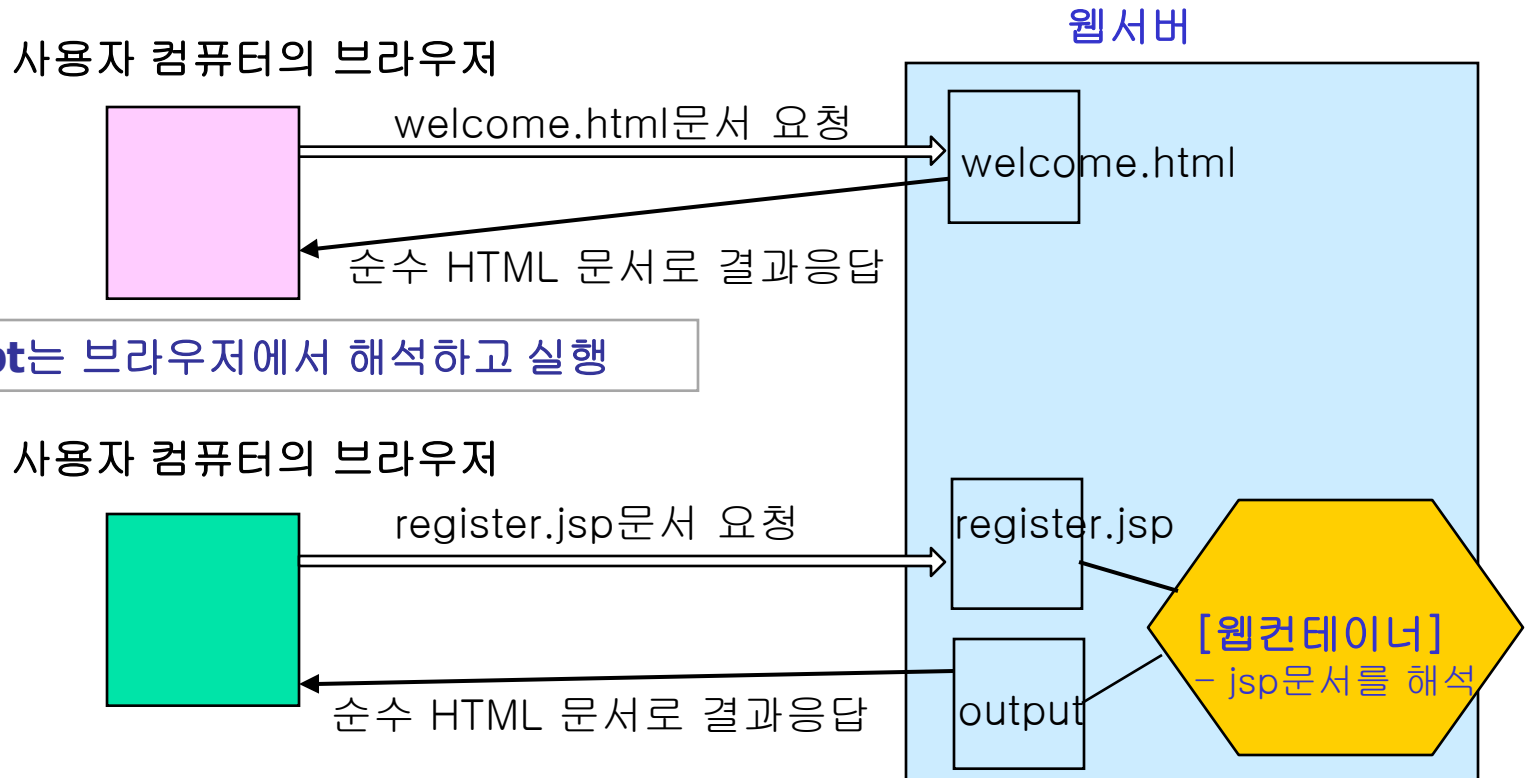
- 사용자의 요청이 있을 때 웹서버 상에서 DB등을 활용한 작업을 처리한 후 그 작업결과를 HTML의 형태로 가공하여 사용자의 브라우저에 전달
- JSP, ASP, PHP등

■ Client Side Script

- **실행위치가 클라이언트**의 웹 브라우저인 스크립트 언어
- **브라우저에 의해 해석되어 실행됨**
- 기존의 HTML처럼 클라이언트의 브라우저 자체가 스크립트 처리
- **JavaScript, Vbscript** 등

동적 웹 페이지

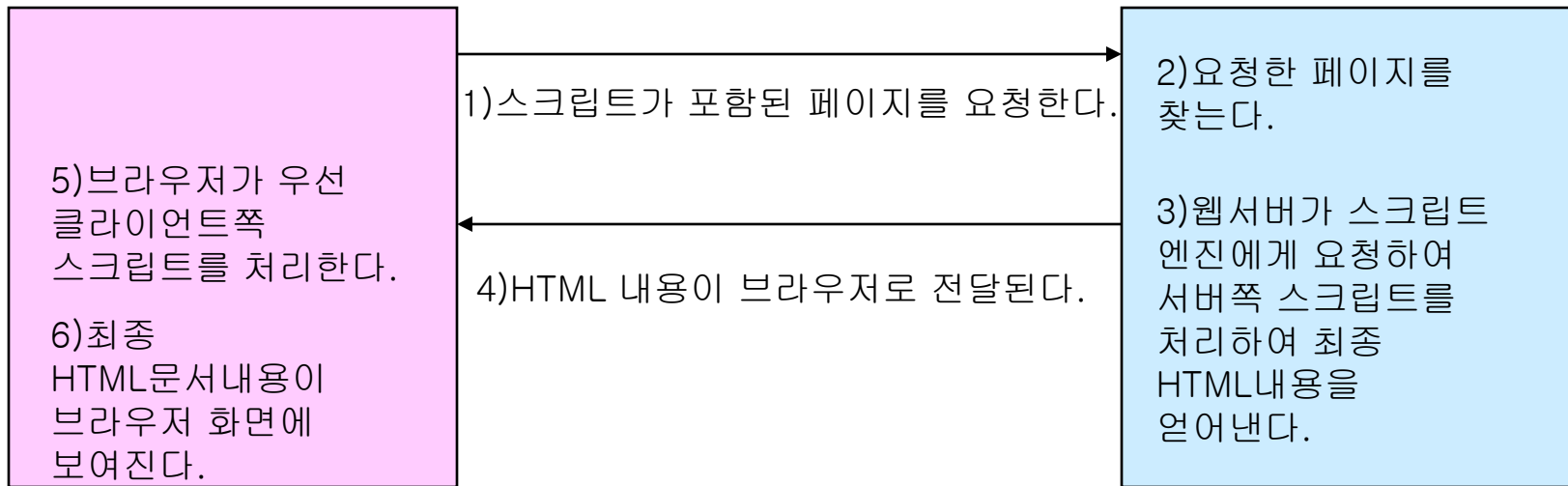
■ 동적 웹 페이지의 JSP



서버쪽 / 클라이언트쪽 스크립트

사용자 컴퓨터의 브라우저

웹 서버



- ✓ <http://www.naver.com/welcome.jsp>
 - <http://>로 시작하는 인터넷 프로토콜을 이용해서 접근하면 서버에서 해당 파일을 홈디렉토리에서 찾아 해석, 가공한 후 가공된 결과를 클라이언트에게 보내게 됨



자바스크립트

- 자바스크립트는 HTML문서에 <script>태그를 이용하여 자바스크립트 프로그램을 직접 삽입,
- 웹 브라우저는 HTML문서와 자바 스크립트 프로그램을 동시에 해석하면서 웹 브라우저 화면에 출력함
 - 자바스크립트를 포함한 HTML문서는 웹 브라우저의 자바 스크립트 인터프리터에 의해 실행됨
 - 웹 브라우저는 HTML문서들을 순서대로 해석하면서 <script>태그가 나타나면 자바스크립트 프로그램을 수행함
 - ※ 웹 브라우저 구성요소 - **html문서 처리 엔진, 자바스크립트 엔진**



자바스크립트

- 자바 스크립트 코드의 동작원리
 - 브라우저가 html문서를 처음부터 한 줄씩 태그들을 해석해 나가면서 결과를 화면상에 출력
 - html 문서 엔진이 <script>태그를 만나면 자바스크립트 엔진을 호출하여 코드의 실행을 지시하며, 자바스크립트 엔진은 <script>태그 내의 코드를 실행함
 - 브라우저는 문서의 끝까지 태그를 정상적으로 해석하여 출력하며, 사용자의 이벤트가 발생하면 그에 상응하는 이벤트 핸들러를 실행시킴



자바스크립트 특징

■ 자바 스크립트의 특징

- html문서 내에 그대로 추가하여 작성
- html과 같이 작성한 후 곧바로 브라우저에서 실행하는 인터프리터 방식의 언어 (컴파일 과정 없이 그대로 웹 브라우저로 실행함)
- 객체 지향언어
- 변수에 대한 특정 자료형을 선언하지 않고도 사용할 수 있음
- 운영체제(윈도우, 유닉스, 리눅스)에 제한 없이 사용 할 수 있음

■ 단점

- html 문서에 포함되어 소스 코드가 노출되므로 보안 유지 어려움
- 객체가 간소화 되어 있어 배우기 쉬우나, 그 만큼 한정된 객체와 메서드를 가지고 있기 때문에 다양한 프로그램을 만들기 어려움

자바스크립트의 동작방법

- [1] `<script> ~ </script>` 태그를 이용
 - 자바스크립트 프로그램을 `<script>` 태그 안에 기술
 - 하나의 html문서에 여러 개의 스크립트를 사용할 수 있음

자바스크립트의 위치

-`<script>`태그는 html내의 어느 곳에도 위치할 수 있으나

[1] 주로 자바 스크립트 함수는 `<head>`태그 사이에 정의, 함수 호출은 `<body>`태그 내에서 함

[2] 화면 요소들을 주로 컨트롤하게 되는 경우

- `<body>` 태그의 마지막에 위치
- `body`의 위쪽에 위치할 경우 `body` 내부의 엘리먼트(태그)에 접근하는 스크립트 구문들은 아직 해당 엘리먼트가 로딩이 되지 않았기 때문에 문제를 일으키게 됨

```
<script type="text/javascript">
```

```
<!--
```

```
자바스크립트 코드 구문
```

```
-->
```

```
</script>
```



자바스크립트의 동작방법

```
<HTML>
<HEAD>
  <script type="text/JavaScript">
    <!--
      document.write("<h2>안녕하세요</h2>");
    -->
  </script>
</HEAD>
<BODY>
<h1>html문서 - script 태그 이용</h1>
</BODY>
</HTML>
```

- document : html 문서를 의미하는 객체
- write : document 객체의 메서드
 - ()안의 태그를 html문서에 쓰라는 의미

안녕하세요

html문서 - script 태그 이용

자바스크립트의 동작방법

- [2] 외부 자바스크립트 불러오기
 - 내용이 많거나 복잡한 경우, js 확장자를 갖는 별도의 독립된 자바스크립트 파일을 외부에 만들어 두고, 이를 html문서에서 불러서 실행

```
<script type="text/javascript" src="파일명">
</script>
```

```
//독립된 자바스크립트 파일
//한 줄 주석처리
/* 여러 줄 주석 처리 방법 */
document.write("<h2>안녕하세요</h2>");
```

test.js

```
<HTML>
<HEAD>
  <script type="text/javascript" src="test.js" charset="utf-8">
  </script>
</HEAD>
```

자바스크립트의 동작방법

- [3] html 태그의 이벤트 핸들러를 자바 스크립트 코드로 표현하여 동작

```
<BODY>
  <input type="button" value="배경색 변경"
    onClick = "document.bgColor='skyblue'"><br>
  <a href="http://www.naver.com" onMouseOver = "document.bgColor='green'"
    onMouseOut="document.bgColor=" >naver</a>
</BODY>
```

- [4] 자바스크립트 코드를 URL로 사용 (웹 표준 이전에 사용)

```
<a href = "javascript:자바스크립트 코드"> ~ </a>
```

```
<a href="javascript:alert('안녕!!')">창이 실행됨</a>
```

```
<a href="#" onclick="alert('안녕!!')">창이 실행됨</a>
```

권장되는 방식

예제

```
<HTML>
<HEAD>
  <script type="text/javascript">
    <!--
      document.write("<h2>안녕하세요-자바스크립트 코드</h2>");
    //-->
  </script>
</HEAD>
<BODY>
  <input type="button" value="배경색 변경" onClick = "document.bgColor='skyblue'"><br>
  <a href="http://www.naver.com" onMouseOver = "document.bgColor='green'"
  onMouseOut="document.bgColor=''" >naver</a><br>

  <a href="#" onclick="alert('안녕!!')">창이 실행됨</a><br>

  <script type="text/javascript">
    document.write("<h1>body 문서내의 자바스크립트 코드</h1>");
  </script>
</BODY>
</HTML>
```

안녕하세요-자바스크립트 코드

배경색 변경

[naver](http://www.naver.com)

창이 실행됨

body 문서내의 자바스크립트 코드



자바스크립트 작성규칙

- 자바스크립트 작성규칙
 - 관련 코드는 한 줄에 모든 입력이 이루어져야 함
 - 대소문자 구분
 - 큰 따옴표(“) 안에 또 다른 따옴표가 필요한 경우에는 작은 따옴표(‘)를 사용
 - 문장을 구분하는 세미콜론(;) – 생략가능



기본 문법



기본 문법

■ 변수의 선언

- 변수를 선언할 필요 없이 변수가 필요한 곳에서 그냥 사용해도 됨
 - 변수명 = 초기값
- 변수를 선언하고 사용할 때의 선언 방법

```
var 변수명 [= 초기값]
```

- var a;
- var a, b, c;
- var a=5, b;

✓ 자바스크립트는 변수 선언시 데이터 타입을 설정할 필요가 없음

- 변수의 타입이 스크립트 실행 중에 자동으로 변환됨



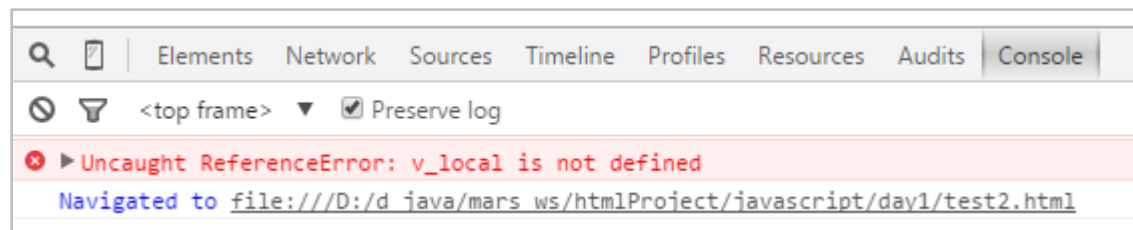
변수예제

```
<HEAD>
<script type="text/javascript">
  var a, b, result;
  a = 10;
  b = 30;
  result = a+b;
  document.write("a 값 = " + a + " <br>");
  document.write("b 값 = " + b + " <br>");
  document.write("result 값 = " + result + " <br>");

  //a 변수의 값 변경
  a = 20;
  document.write("a 값 = " + a + " <br>");
</script>
</HEAD>
```

```
a 값 = 10
b 값 = 30
result 값 = 40
a 값 = 20
```

예제



```
<script type="text/javascript">
```

```
var v_global="전역변수"
```

```
function f_vari(){  
    var v_local="지역변수";  
    document.write("v_global의 값 = " +v_global+" <br>");  
    document.write("v_local의 값 = " +v_local+" <br>");  
}  
f_vari();
```

```
//document.write("함수 밖에서 v_local의 값 = " +v_local+" <br>");  
document.write("함수 밖에서 v_global의 값 = " +v_global+" <br>");  
</script>
```

자료형

- 자바스크립트는 변수 선언 시 데이터 타입을 설정할 필요가 없음
 - 변수의 타입이 스크립트 실행 중에 자동적으로 변환됨

```
var text1, text2;  
text1 = '홍길동';  
text2 = '1+2';  
document.write(text1);  
document.write("<br><br>");  
document.write(text2 + "<br>");
```

```
var t_quot1, t_quot2;  
t_quot1 = "\" + "Hello" + "\"";  
t_quot2 = "취미는 \" 축구\"입니다.\\ n";  
t_quot2 += "\" t야구도 좋아합니다";  
document.write(t_quot1);  
document.write("<br>");  
document.write(t_quot2 + "<br>");  
alert(t_quot2);
```

```
var t_trans;  
t_trans = 250;  
document.write("변수 t_trans의 값=" + t_trans + "<br>");  
t_trans = "데이터형식이 문자형으로 변환됩니다";  
document.write("변수 t_trans의 값=" + t_trans);
```

홍길동

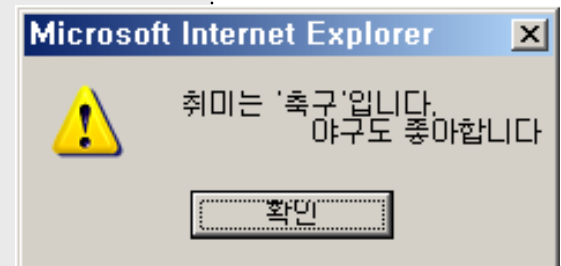
1+2

'Hello'

취미는 '축구'입니다. 야구도 좋아합니다

변수 t_trans의 값=250

변수 t_trans의 값=데이터형식이 문자형으로 변환됩니다





조건문

```
if(조건식1)
{
    조건식1이 참인 경우 실행할 내용;
}
else if(조건식2)
{
    조건식2가 참인 경우 실행할 내용;
}
else if(조건식3)
{
    조건식3이 참인 경우 실행할 내용;
}
else
{
    조건을 모두 만족하지 않는 경우
    실행할 내용;
}
```



반복문

```
while(반복 조건)
{
    반복 내용;
}
```

- 조건이 참인 동안은 계속 반복 수행
- while ~ 내부문장에서 조건이 거짓이 되도록 만들어서 loop를 빠져 나오게 해야 함
- 처음부터 조건이 거짓이면 반복되는 실행구문은 한번도 처리하지 않음.

```
do
{
    반복될 문장;
} while(조건);
```

- 조건을 먼저 체크한 후에 구문을 수행하는 것이 아니라, 일단 수행을 한 후에 조건을 체크
- 조건이 거짓이더라도 실행구문을 반드시 한번은 수행

```
for(초기화식;조건식;증감식)
{
    반복될 문장;
}
```

지정된 횟수만큼 반복해서 실행하는 구문

```
for(변수명 in 객체)
{
    실행블록;
}
```

- 객체에 있는 원소들을 차례대로 순회하는 반복문

객체 제어문

status는 window 객체의 속성입니다
onresize는 window 객체의 속성입니다
onmessage는 window 객체의 속성입니다
parent는 window 객체의 속성입니다
onhashchange는 window 객체의 속성입니다
defaultStatus는 window 객체의 속성입니다
name는 window 객체의 속성입니다
history는 window 객체의 속성입니다
maxConnectionsPerServer는 window 객체의 속성입니다
opener는 window 객체의 속성입니다
location는 window 객체의 속성입니다
screenLeft는 window 객체의 속성입니다
document는 window 객체의 속성입니다

- 객체 제어문 - 객체를 제어하고 조작하는 방법
- for in 문
 - 객체의 모든 속성을 변수에 할당하는 경우에 사용
 - 각 객체가 가지고 있는 속성의 개수만큼 지정된 문장을 수행함

```
for(변수 in 객체)
{
    문장
}
```

```
<script type="text/javascript">
    var property;
    for(property in window)
    {
        document.write("<b>" + property + "</b>는 window
객체의 속성입니다<br>");
    }
</script>
```

window객체의 속성을 property변수에 저장하고, window
객체가 가지고 있는 속성의 개수만큼 화면출력을 반복



new 연산자

■ new 연산자

- 사용자가 새로운 객체를 정의할 수 있도록 해줌

새로운 객체명 = **new** 객체형식(매개변수1, 매개변수2, ...)

```
<script type="text/javascript">  
  var d = new Date();  
  month1 = d.getMonth() + 1;  
  document.write(1900+d.getFullYear()+ "년" + month1+ "월" +d.getDate()+ "일"+"<br>");  
</script>
```

2014년3월10일

■ this 키워드

- this는 현재 객체를 나타내는 예약어
- 현재 객체를 참조하기 위해 사용, 자기 자신을 가리키는 것임
- this.객체명(매개변수)



with문

- with(객체)에서 지정한 객체는 { } 내에서 생략하고 사용할 수 있음
- 반복적으로 특정한 객체명을 자주 사용하는 경우에 유용하게 사용됨

with (객체)

{

처리문장

}

```
<script type="text/javascript">
  with(document)
  {
    write("hello<br>");
    write("안녕!!<br>");
  }
</script>
```



배열(array)

변수가 하나의 값을 가지는 것이 아니라 여러 개의 값을 동시에 가지는 경우

■ 배열

- 같은 타입의 데이터를 여러 개 저장할 수 있는 저장소
- 하나의 변수 이름에 여러 개의 값을 동시에 저장하는 것
- 메모리의 연속적인 공간이 할당됨

■ 배열의 선언

```
배열명 = new Array(인덱스번호);  
배열명 = new Array();
```

- 예) `var colors = new Array(3);`
 - 0~2 까지 3개의 요소가 만들어짐
 - 인덱스는 항상 0부터 시작



배열(array)

```
var colors = new Array(3);
```

• 자바

```
int[] arr=new int[2];  
arr[0]=1;  
arr[1]=2;
```

```
int[] arr={1, 2, 3};
```

- 배열에 값 넣기
 - colors[0] = “red”;
 - colors[1] = “blue”;
 - colors[2] = “green”;
- 배열의 초기화
 - 배열명 = new Array(초기값1, 초기값2);
 - 예) colors = new Array(“red”, “blue”, “green”);
 - var arr = [“red”, “blue”, “green”];



배열

- 배열 - 변수가 하나의 값을 가지는 것이 아니라 여러 개의 값을 동시에 가지는 경우
- Array객체 - 자바스크립트에서 배열을 사용하고자 할 때 이용되는 객체로 new 키워드를 사용하여 객체를 생성함
- Array 객체 생성 형식
 - 변수명 = new Array(인덱스 번호);
 - 인덱스 번호 - 배열에 넣을 값의 최대 길이를 의미, 생략하면 자동으로 배열을 설정해 줌



배열

- `var a = 10;`

a

10

- `var arr = [10, 20, 30, 40, 50];`

- `var arr = new Array(10, 20, 30, 40, 50);`

arr

10	20	30	40	50
----	----	----	----	----

`arr[0]` `arr[1]` `arr[2]` `arr[3]` `arr[4]`

배열

arr

10	20	30	40	50
----	----	----	----	----

arr[0] arr[1] arr[2] arr[3] arr[4]

- 배열 선언
 - `var arr = new Array(5);`
 - 또는 `var arr = new Array();`
- 배열에 값 넣기
 - `arr[0] = 10;`
 - `arr[1] = 20;`
 - `arr[2] = 30;`
 - `arr[3] = 40;`
 - `arr[4] = 50;`



배열

```
<script type="text/javascript">
```

```
    colors=new Array(3);
```

```
    colors[0]="red";
```

```
    colors[1]="blue";
```

```
    colors[2]="green";
```

```
    for(var i=0; i<3; i++)
```

```
        document.write(i + " : " + colors[i] + "<br>");
```

```
    document.write("<hr>");
```

```
// 배열 선언과 동시에 초기화
```

```
    num =new Array(100, 200, 300, 400);
```

```
    for(var i=0; i<num.length; i++)
```

```
        document.write(i + " : " + num[i] + "<br>");
```

```
</script>
```

0 : red
1 : blue
2 : green

0 : 100
1 : 200
2 : 300
3 : 400

num =[1100, 2100, 3100, 4100];
for(var i=0; i<4; i++)
document.write(i + " : " + num[i] + " ");



Array객체

■ 속성

속성	기능
<code>length</code>	배열의 크기를 알려줌

```
num = new Array(100, 200, 300, 400);  
document.write("num[]의 length : "+num.length + "<br>");  
  
for(var i=0; i<num.length; i++)  
    document.write(i + " : " + num[i] + "<br>");
```




실습 - 난수를 발생하여 명언 보여주기

불가능은 대개 시도하지 않기 때문이다

```
<script type="text/javascript">
  //난수발생하여 명언보여주기
  var msg = new Array(5);
  msg[0] = "너 자신을 알라";
  msg[1] = "불가능은 대개 시도하지 않기 때문이다";
  msg[2] = "삶을 변화시키려면 지금 당장 시작하라";
  msg[3] = "행복의 비결은 좋아하는 일을 해서가 아니라 해야 하는 일을 좋아하기 때문
    이다.";
  msg[4] = "인간이 실패하는 이유는 단 하나, 자기 자신에 대한 진정한 믿음이 부족하
    기 때문이다.";

  var num = Math.floor(Math.random() * 5);
  document.write(msg[num]);
</script>
```



자바 스크립트에서 html 객체 접근



자바 스크립트에서 html 객체 접근

웹 브라우저는 **html문서를 계층구조의 객체들로 관리**하며,
자바 스크립트는 **이 객체들의 정보에 접근하여 객체를 제어**할 수 있음

- 웹 브라우저는 html문서를 객체로 관리
 - html문서, 그림, form등의 각 객체를 대표하는 객체 인스턴스를 생성해서 관리
 - html문서를 위해서는 document라는 객체가 생성되고, html문서에 포함된 품은 document 객체에 포함된 객체들로 표현됨
 - 이러한 객체들은 자바 스크립트를 통해 속성과 함수에 접근해 사용할 수 있음

html 객체 계층도

```
<FORM name="frm1">
```

```
아이디 :<INPUT TYPE="text" name="id">
```

```
var userid = window.document.frm1.id.value;
```

■ html 객체 구조

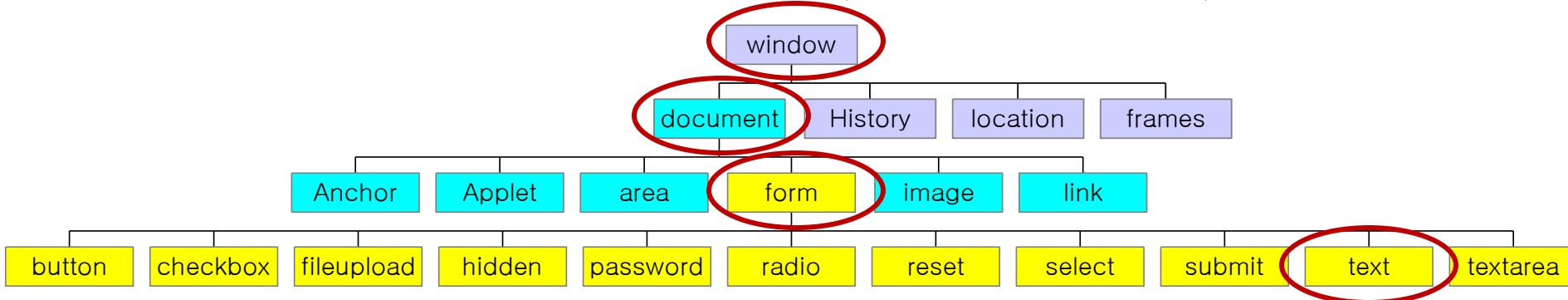
- 객체들은 계층구조로 이뤄져 있음
- 최상위 객체로는 Window 객체가 있으며 아래에는 frames, document, history, location 객체가 있으며 각각의 객체들은 아래로 또 다른 객체들로 파생

■ Window객체

- 자바스크립트의 브라우저 내장객체의 최상위 객체, 전체 윈도우에 적용될 내용들을 가지고 있다
- 윈도우마다 하나씩 존재하는 객체 (윈도우 창을 나타낼 때)

■ Document객체

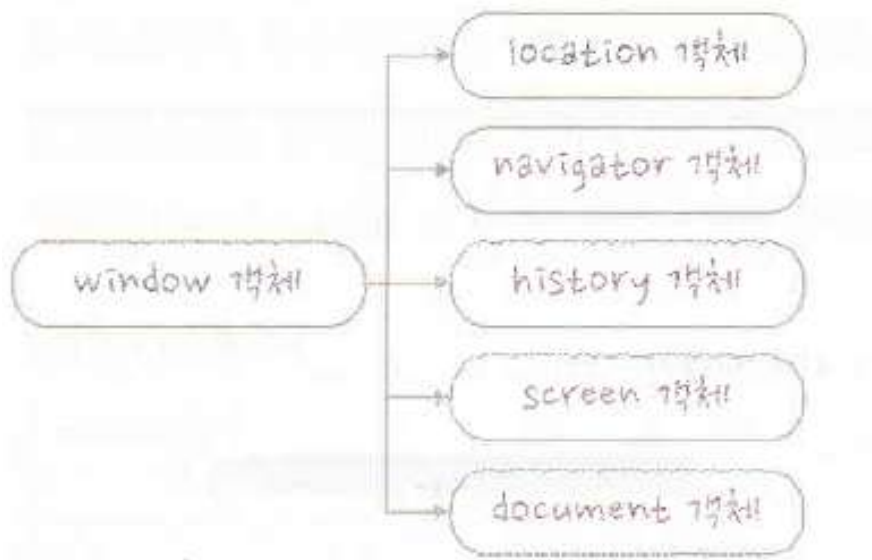
- 현재의 문서에 대한 내용을 가지고 있다.
- 제목, 배경색, 폼에 대한 속성을 가지고 있다
- html 페이지마다 하나씩 존재하는 객체 (웹 페이지를 나타낼 때)

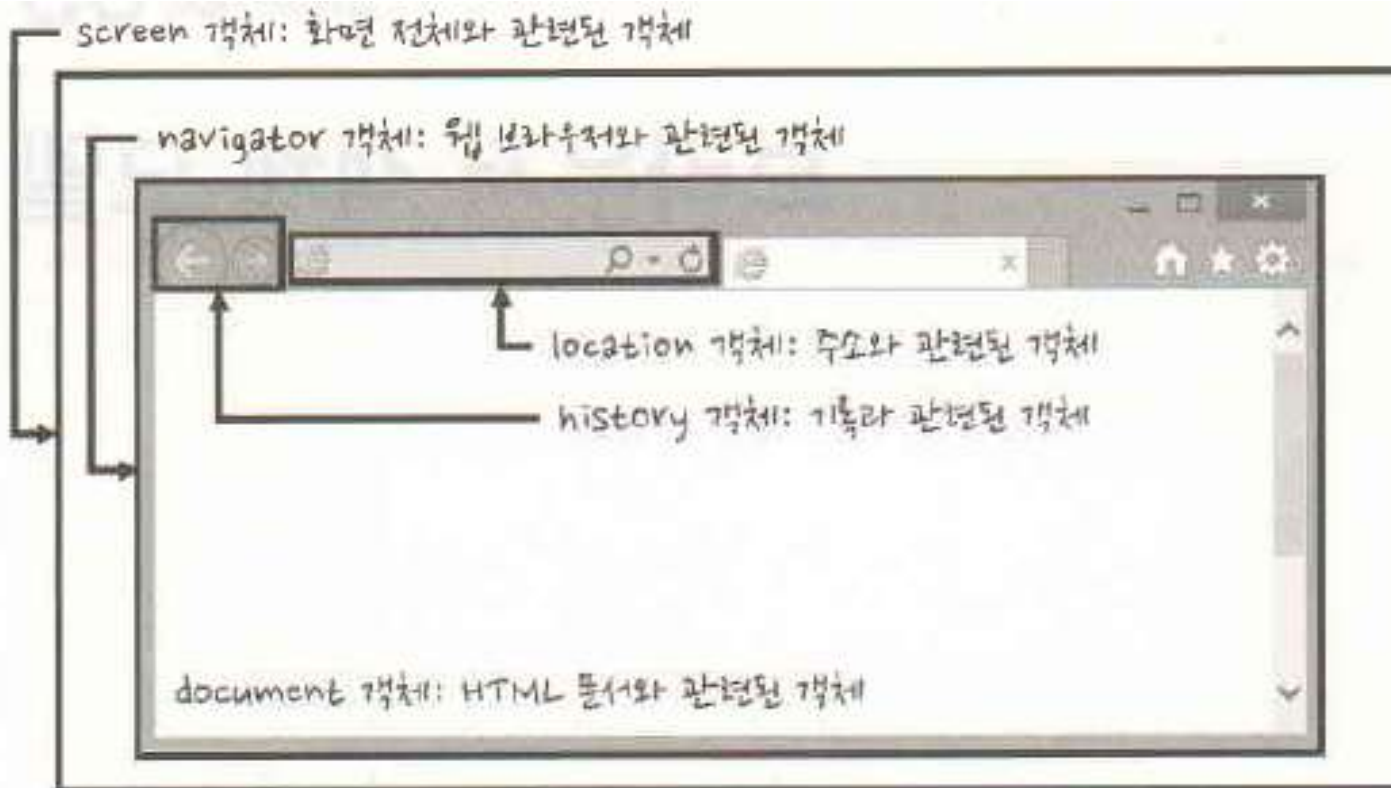


브라우저 객체 모델

■ 브라우저 객체 모델(BOM, Browser Object Model)

- 웹 브라우저와 관련된 객체의 집합
- 문서 객체 모델(DOM, Document Object Model)이라 통합해서 부르기도 하고,
- document 객체와 관련된 객체의 집합을 문서 객체 모델이라고 부르고, 그 이외의 객체 집합을 브라우저 객체 모델이라고 부르기도 함
- 대표적인 브라우저 객체 모델 - window, location, navigator, history, screen, document







객체, 속성, 메서드

- 자바스크립트 – 객체지향적인 언어
 - 웹 문서의 모든 요소들을 객체 개념으로 다루며 각각의 객체들은 속성과 메서드를 가짐
 - 자바스크립트에서 객체는 일반적으로 웹 브라우저와 관계되어 사용됨
 - 브라우저 객체 – 브라우저 윈도우, 웹 문서, 문서에 포함된 이미지, 폼 등과 같은 요소들이 하나의 객체가 됨
 - 내장 객체 – 시간(날짜), 배열 등과 같이 자바스크립트 자체에 내장된 객체도 있음
 - 사용자가 정의한 객체도 사용할 수 있음
 - 객체 지향언어 – 실세계의 사물(객체)을 모델링해서 이를 프로그래밍에 적용한 것
 - 객체들은 속성과 메서드를 가짐



속성(Property)

■ 속성

- 객체의 특징, 성질
- 자바스크립트에서 각각의 객체들은 속성을 가짐
- 하위 객체는 상위 객체의 속성이 됨
 - image는 하나의 객체이지만, document 객체의 속성이 됨
 - document 객체는 배경색, 문서의 제목 등과 같은 속성을 가짐
 - image객체는 너비, 높이 등의 속성을 가짐
 - 객체의 속성을 바꾸기 위해서는 도트(.)를 사용하여 구분

객체.속성 = “속성값”

예) document.bgColor = “green”;



메서드(Method)

- 메서드
 - 객체의 동작, 하는 일, 기능
 - 메서드는 ()를 사용

객체.메서드(값)
예) window.open()
document.write()

- 자바스크립트의 객체들(브라우저 객체, 내장객체)은 이미 이러한 속성과 메서드가 정의되어 있음



자바스크립트 객체 모형

- 자바스크립트에는 수많은 객체가 이미 정의되어 있음

- 웹 브라우저와 관련 있는 브라우저 객체

`window`, `document`, `frame`, `history`, `location`, `form`, `image`, `link`, `radio`, `text`, `navigator` 등

- 자바스크립트 자체에 내장된 내장 객체

`Date`, `Math`, `String`, `Array` 등



브라우저 객체 모형

- 브라우저 객체는 브라우저 화면에 나타나는 모든 요소들을 포함하는 것으로 계층적인 트리(tree)구조로 구성되어 있음
 - window라는 최상위 객체에서 파생된 수많은 하위 객체를 포함
 - 각각의 하위 객체들은 계층구조에 의해 정의되어 있으며, 접근할 수 있음
 - 브라우저 객체는 계층구조로 접근함



브라우저 객체에 접근하는 방법

- 브라우저 화면의 특정 객체(text 입력상자)에 접근하는 방법
 - [1] 객체 이름으로 객체에 접근
 - 객체 고유의 이름(태그의 name 속성에 지정된 값)을 사용
 - **window.document.form이름.입력상자이름**

```
<FORM NAME="frm1">  
아이디 :<INPUT TYPE="text" name="tel">
```

```
var phone = window.document.frm1.tel.value;
```

※window객체는 최상위 객체
이므로 생략가능



브라우저 객체에 접근하는 방법

■ [2] DOM Method 이용

■ document.getElementById()

- id값을 통해 해당 element에 접근
- id에 대한 요소(element)를 가져온다(get)
- 예) document.getElementById('tel')
 - 문서(document)내에서 'tel'이라는 id를 가진 요소를 가져와라

```
<FORM NAME="frm1">
```

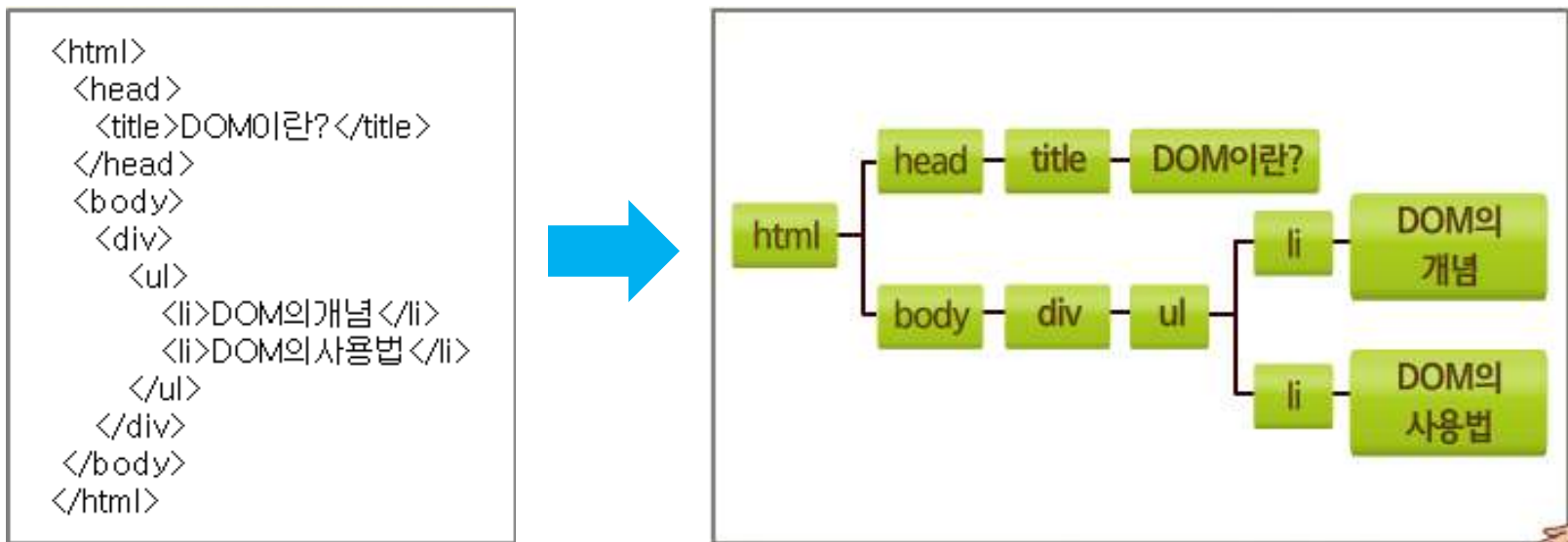
```
아이디 : <INPUT TYPE="text" name="tel" id="tel">
```

```
var phone = document.getElementById("tel").value;
```

DOM

■ DOM(Document Object Model)

- 문서 객체 모델, 하나의 웹 문서를 객체화해서 문서의 구조에 접근할 수 있는 방법



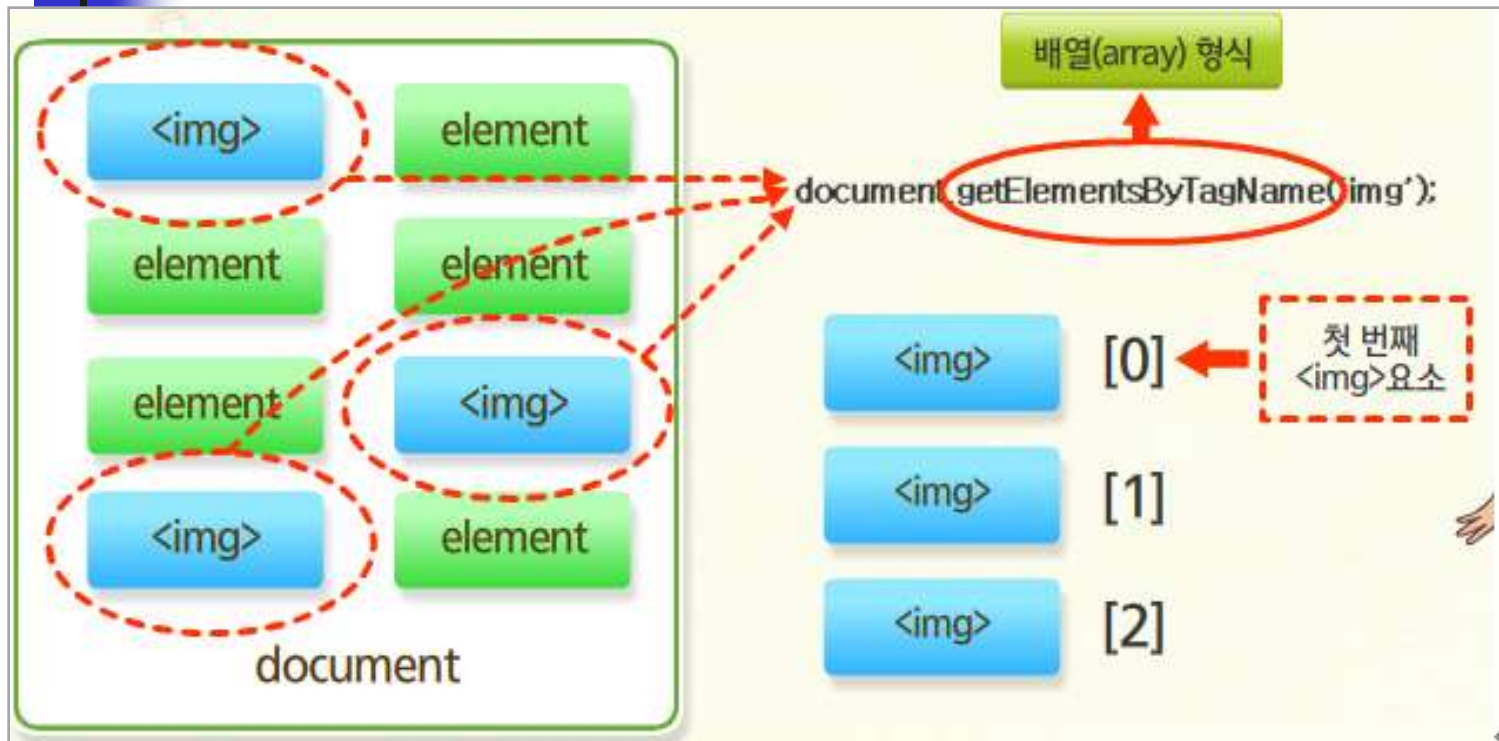
구조화된 하나 하나의 요소 = 객체(object)
객체들에 접근하여 활용할 수 있도록 하는 방법 = DOM



DOM Method

- DOM Method
 - id 접근 - `document.getElementById()`
 - tagName 접근 - `document.getElementsByTagName()`
- [1] `document.getElementById()`
 - id값을 통해 해당 element에 접근
 - id에 대한 요소(element)를 가져온다(get)
 - 예) `document.getElementById('tel')`
 - 문서(document)내에서 'tel'이라는 id를 가진 요소를 가져와라
- [2] `document.getElementsByTagName()`
 - 콘텐츠 내에 삽입된 태그명을 통해 접근
 - 복수형(Elements)을 사용함
 - tagName에 대한 요소(element)를 가져온다
 - 예) `document.getElementsByTagName('img')`
 - 문서(document)내에서 'img'라는 tagName를 가진 요소를 가져와라
 - => 여러 개 요소를 가져올 수 있으므로 배열로 처리

DOM Method



- id 접근 메서드
 - 항상 하나만 존재하기 때문에 단수형
- 태그 접근 메서드
 - 여러 개가 존재하기 때문에 복수형 => 항상 배열로 값을 받게 됨

예제



```

```

```

```

```

```

```
<script type="text/javascript">
```

```
    var img = document.getElementsByTagName("img");
```

```
    for(i=0;i<img.length;i++){
```

```
        img[i].style.border="3px solid green";
```

```
    }
```

```
</script>
```

예제

회원가입

아이디

전화번호

```
<SCRIPT type="text/javascript">
function send()
{
    var id = window.document.frm.userid.value;
    var tel = document.getElementById("tel").value;
    alert("아이디 : "+id + ", 전화번호 : "+tel);
}
</SCRIPT>
</HEAD>
<BODY>
<H3>회원가입</H3>
<FORM NAME="frm">
아이디 <INPUT TYPE="text" name="userid"><br>
전화번호 <INPUT TYPE="text" name="tel" id="tel"><br>
<INPUT TYPE="button" value="확인" onClick="send()">
</FORM>
</BODY>
```

html 객체 접근

```
<SCRIPT type="text/javascript">
function send()
{
    var name = window.document.frm.name.value;
    if(name == "")
    {
        alert("이름을 입력하세요");
    }
}
</SCRIPT>
</HEAD>
<BODY>
<H3>회원가입</H3>
<FORM NAME="frm">
이름 <INPUT TYPE="text" name="name"><br>
아이디 <INPUT TYPE="text" name="id"><br>
비밀번호 <INPUT TYPE="password" name="pwd"><br>
<INPUT TYPE="button" value="전송" onClick="send()">
</FORM>
</BODY>
```


회원가입

이름

아이디

비밀번호

Microsoft Internet Explorer

 이름을 입력하세요



html 객체 접근2

```
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
function send()
{
    if(window.document.frm.name.value == "")
    {
        alert("이름을 입력하세요");
        frm.name.focus();
        return;
    }

    //if(frm.id.value.length==0){
    //if(!(document.frm.id.value.length >=6 && document.frm.id.value.length <=10)){
    if(frm.id.value.length<6 || frm.id.value.length>10){
        alert("아이디는 6~10글자 사이여야 합니다...");
        //frm.id.focus();
        frm.id.select();
        return;
    }
}
```



html 객체 접근2

```
if(!document.frm.pwd.value){
    alert("비밀번호를 입력하세요....");
    document.frm.pwd.focus();
    return;
}

//frm.method="post";
//frm.action="test.jsp";
document.frm.submit();
}
</SCRIPT>
</HEAD>
<BODY>
<H3>회원가입</H3>
<FORM NAME="frm" method="post" action="test01.jsp">
이름 <INPUT TYPE="text" name="name"><br>
아이디 <INPUT TYPE="text" name="id"><br>
비밀번호 <INPUT TYPE="password" name="pwd"><br>
<INPUT TYPE="button" value="전송" onClick="send()">
</FORM>
</BODY>
</HTML>
```

예제-DOM method

DOM이란

색상 변경하기

```
<body>
  <h1 id="targetId">DOM0이란</h1>
  <button onclick="changeColor();">색상 변경하기</button>

  <script type="text/javascript">
    <!--
      function changeColor(){
        document.getElementById("targetId").style.color="red";
      }
    -->
  </script>
</BODY>
```

```
<input type="button" value="색상 변경하기" onclick="changeColor();">
```



CSS 조작하기

- HTML에서 style 속성 사용

```
<div id="ex" style="color:blue;background-color:lightblue">스타일 변경하기</div>
```

- 자바스크립트에서 style요소 처리

- 특정 노드의 style속성을 사용하면 CSS요소에 접근할 수 있음

```
var divEx = document.getElementById("ex");  
divEx.style.color = "red";  
divEx.style.backgroundColor = "yellow";
```



예제

- java
- jsp
- oracle

```
<BODY>
<ul id="lang">
  <li>java</li>
  <li>jsp</li>
  <li>oracle</li>
</ul>

<script type="text/javascript">
  var lang = document.getElementById("lang");
  var list = lang.getElementsByTagName("li");
  for(i=0;i<list.length;i++){
    list[i].style.color="blue";
  }
</script>
</BODY>
```

```
var liEls=langEl.getElementsByTagName("li");
    lang 속성을 가진 하위li만 선택됨
```

document.getElementById('wrap').getElementsByTagName('img')
- wrap 이라는 id를 가진 요소의 html 태그 중 img 태그만을 가져오도록 함

css와 자바스크립트의 스타일 속성명 비교

- style 오브젝트 - 스타일값들을 자바스크립트를 통해 컨트롤할 수 있음

CSS Property	JavaScript Reference
background	background
background-attachment	backgroundAttachment
background-color	backgroundColor
background-image	backgroundImage
background-position	backgroundPosition
background-repeat	backgroundRepeat
border	border
border-bottom	borderBottom
Float	styleFloat



MouseOver 효과

```
<table width="500" border="1">
  <tr>
    <td>제목1</td>
    <td>제목2</td>
  </tr>
  <tr
onMouseOver="this.style.backgroundColor='#e8e8ff';this.style.cursor='pointer'"
onMouseOut="this.style.backgroundColor=''">
    <td>내용1</td>
    <td>내용2</td>
  </tr>
  <tr
onMouseOver="this.style.backgroundColor='#e8e8ff';this.style.cursor='pointer'"
onMouseOut="this.style.backgroundColor=''">
    <td>내용1</td>
    <td>내용2</td>
  </tr>
</table>
```



예제-submit 버튼 이용

```
<script type="text/javascript">
    function send(){
        if(!frm1.title.value){
            alert("제목을 입력하세요");
            frm1.title.focus();
            return false;
        }else if(frm1.name.value==""){
            alert("작성자를 입력하세요");
            frm1.name.focus();
            return false;
        }else if(frm1.pwd.value.length==0){
            alert("비밀번호를 입력하세요");
            frm1.pwd.focus();
            return false;
        }

        //작성한 폼양식을 서버로 전송
        return true;
    }
</script>
```

```
<form name="frm1" method="post" action="test01.jsp" onsubmit="return send();">
```

```
<table width="500" class="box3">
```

```
<tr>
```

```
<th width="100">제목</th>
```

```
<td width="400"><input type="text" name="title" ></td>
```

```
</tr>
```

```
<tr>
```

```
<th>작성자</th>
```

```
<td><input type="text" name="name" ></td>
```

```
</tr>
```

```
<tr>
```

```
<th>비밀번호</th>
```

```
<td><input type="password" name="pwd" ></td>
```

```
</tr>
```

```
<tr>
```

```
<td align="center" colspan="2">
```

```
<input type="submit" value="등록" > &nbsp;
```

```
<input type="button" value="글목록" OnClick="location.href='list.html'" />
```

```
</td>
```

```
</tr>
```

```
</table>
```

```
</form>
```

(3) 이벤트 핸들러의 반환값- **true, false**

- **true**가 반환되면 그 이후의 행동을 계속함.
- **false**가 반환되면 더 이상 작업을 진행하지 않음



button 이용

```
<form name="frm1" method="post" action="test01.jsp" >
```

```
<input type = "button" value="등록2" onclick="send2();">
```

```
function send2(){
    if(!frm1.title.value){
        alert("제목을 입력하세요");
        frm1.title.focus();
        return;
    }else if(frm1.name.value==""){
        alert("작성자를 입력하세요");
        frm1.name.focus();
        return;
    }else if(frm1.pwd.value.length==0){
        alert("비밀번호를 입력하세요");
        frm1.pwd.focus();
        return;
    }

    //작성한 폼양식을 서버로 전송
    //frm1.action = "test01.jsp";
    //frm1.method="post";
    frm1.submit(); //전송
}
```



함수



함수(Function)

- 자바스크립트에서 제공하는 함수
 - 내장함수(Built in) – 자체적으로 정의되어 있는 함수
 - 객체와 연결되어 사용되는 메서드와는 달리 객체에 독립적으로 사용됨
 - 사용자 정의 함수



사용자 정의 함수

■ 함수 정의 – 함수 만들기

```
function 함수명 ( 입력 값을 받는 매개변수 )  
{  
    자바스크립트 코드;  
  
    return 반환값;  
}
```

- 매개변수=인수= argument= parameter=인자 : 입력값
- return 반환값 : 작업을 마친 후의 결과값을 return을 이용하여 호출자에게 돌려줌
 - 반환값은 1개 이하만 가능



사용자 정의 함수

```
<HEAD>
<SCRIPT type="text/javascript">
//함수 정의
function sample1()
{
    document.write("자바스크립트 - 함수 연습");
}
</SCRIPT>
</HEAD>
<BODY>
<h2>함수를 호출해보자</h2>

<SCRIPT type="text/javascript">
//함수 호출
    sample1();
</SCRIPT>
</BODY>
```



예제 1-매개변수가 있는 경우

```
<HEAD>
<SCRIPT type="text/javascript">
//함수 정의
function sample2(name){
    alert("내 이름은 " + name);
}
</SCRIPT>
</HEAD>
<BODY>
<h2>버튼 클릭시 함수를 호출해보자</h2>
<FORM NAME="frm">
<INPUT TYPE="button" value="확인" onClick="sample2('김유성')">
</FORM>
<SCRIPT type="text/javascript">
<!-- 함수 호출 - 매개변수 있는 함수 -->
    sample2("홍길동");
</SCRIPT>
</BODY>
```



예제2-매개변수, 리턴값이 있는 경우

```
<HEAD>
<SCRIPT type="text/javascript">
//함수 정의
function add(a, b)
{
    var sum;
    sum = a + b;

    return sum;
}
</SCRIPT>
</HEAD>
<BODY>
<h2>두 수의 합 구하기</h2>

<SCRIPT type="text/javascript">
    var num1, num2, result;
    num1 = 10, num2 = 20;
    result = add(num1, num2);
    document.write("두 수의 합 : " + result);
</SCRIPT>

<h3>여기서 부터 다음 문장</h3>
</BODY>
```



반환값 return

■ 형식

```
return 반환할 내용;  
return;
```

■ return 명령

- 반환값을 돌려주기 위해 사용
- 함수의 종료 명령도 같이 겸함
 - return 명령을 만나면 무조건 함수처리가 종료되고, 호출한 곳으로 복귀하도록 되어 있음
 - 반환값을 돌려줄 목적이 아닌 함수를 종료할 목적으로도 사용됨



함수의 호출과 반환값

■ 함수의 호출

- 만들어 놓은 함수를 실행시키는 것
- 형식

함수명(매개변수);

결과값을 받는 변수 = 함수명(매개변수);

- 1) 매개변수, 반환값이 모두 없는 함수 호출
 - 예) sample();
- 2) 매개변수는 있고, 반환값이 없는 함수 호출
 - 예) sample(“홍길동”);
- 3) 매개변수는 없고, 반환값이 있는 함수 호출
 - 예) result = sample();
 - 함수를 호출한 후 실행된 결과값을 왼쪽의 변수에 저장
- 4) 매개변수, 반환값이 모두 있는 함수 호출
 - 예) result = add(num1, num2);



함수의 호출 위치

- 1) 프로그램 내에서 호출하기
 - 프로그램을 진행하는 도중에 특정 함수의 기능이 필요한 경우 그 위치에서 함수를 호출
- 2) 웹 페이지가 실행되는 시점에서 호출하기
 - 자바스크립트를 포함하는 웹 페이지가 화면에 표시되는 순간 원하는 기능을 처리하고자 하는 경우 사용
 - <body> 태그 안에서 원하는 함수 호출 (onLoad 이벤트 핸들러 사용)

```
<HEAD>
<SCRIPT type="text/javascript">
function call()
{
    처리내용
}
</SCRIPT>
</HEAD>
<BODY onLoad = "call()">
```



함수의 호출 위치

- 3) 특정 행동이 있을 경우에 호출하기(이벤트 처리)
 - 사용자가 특정 행동(이벤트 발생)을 할 경우 어떤 기능을 수행하고자 할 때 사용

```
<HEAD>
<SCRIPT type="text/javascript">
function call()
{
    처리내용
}
</SCRIPT>
</HEAD>
<BODY>
<form>
<input type="button" value="회원가입" onClick = "call()">
</form>
```

예제

```
<SCRIPT type="text/javascript">
function getGrade()
{
    var score = window.document.frmScore.avg.value;
    var result="";
    if(score >=90)
        result = "A학점";
    else if(score >=80)
        result = "B학점";
    else if(score >=70)
        result = "C학점";
    else if(score >=60)
        result = "D학점";
    else if(score <60)
        result = "F학점";
    else
        result = "잘못입력";
    alert(result);
    document.frmScore.grade.value = result;
}
</SCRIPT>
```

학점구하기

평균 점수 입력

학점 :



예제

```
<BODY>
<H3> 학점 구하기</H3>
<hr>
<FORM NAME="frmScore">
평균 점수 입력 <INPUT TYPE="text" name="avg"><P>
<INPUT TYPE="button" value="학점구하기" onClick="getGrade()">
학점 : <INPUT TYPE="text" name="grade" size="15"><p>
</FORM>
</BODY>
```

```
function getGrade(frm)
{
    var score = frm.avg.value;
}
```

```
<INPUT TYPE="button" value="학점구하기" onClick="getGrade(this.form)">
```

실습

반올림 함수
- Math.round(실수)

내장 함수 이용
- parseInt(문자열)
- Number(문자형태의 숫자)

- 1. 미국 달러를 입력 받아 한국 원화로 변환해주는 환율 계산 함수 만들기
- 2. 인자로 두 수를 받아서 두 수 중 더 큰 수를 구한 후 그 결과값을 리턴해 주는 함수 만들기

환율계산

미국 달러입력

환율계산

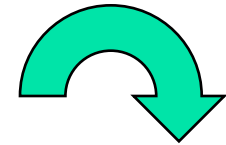
한국 원화

더 큰수 구하기

첫번째 수 입력

두번째 수 입력

더 큰수 구하기



더 큰수 구하기

첫번째 수 입력 14

14

두번째 수 입력 6

6

더 큰수 구하기

Microsoft Internet Explorer



더 큰 수는 14 입니다

확인

1 달러 => 1183.5 원

달러 * 1183.5 => 원화

내장 함수

■ 자바스크립트 내장 함수

- 자체적으로 정의되어 있는 함수
- 객체와 연결되어 사용되는 메서드와는 달리 객체에 독립적으로 사용됨

내장 함수명	기능
eval(수식 문자열)	문자열로 입력된 수식을 계산해 준다
parseInt(문자열)	부동 소수점이나 문자열을 2, 8, 10, 16진수의 정수로 변환해 준다
parseFloat(문자열)	문자열을 부동소수점(실수)으로 변환해준다
Number(문자형태의 숫자)	문자 형태의 숫자를 숫자로 변환
String(숫자)	숫자를 문자로 변환
escape(문자열)	ISO Latin-1 문자 세트를 ASCII 형태로 바꾸어 리턴한다 (인코딩)
unescape(문자열)	ASCII 형태의 문자를 ISO Latin-1 문자 세트로 바꾸어 준다 (디코딩)
isFinite(값)	값이 유한의 수인지 판별하여 유한이면 true, 무한이면 false
isNaN(값)	값이 문자인지 숫자인지 판별하여 순수 문자이면 true, 아니면 false



parseInt()

- parseInt(문자열)
 - 문자열을 정수로 변환시켜주는 함수
 - 매개 변수를 이용하여 2진수, 8, 10, 16진수의 수로도 변환이 가능
 - 숫자형과 문자형이 혼합된 문자열에서 숫자형 뒤에 오는 문자형은 숫자만 출력
 - 소수인 경우 소수점 이하를 버리고, 정수만 반환
 - 부동소수점형이나 정수로 바꾸지 못하면 NaN(Not a Number)를 반환



예제

```
<SCRIPT type="text/javascript">
  n = parseInt(prompt("정수를 입력하세요....",""));
  //n = prompt("정수를 입력하세요....","");
  sum=0;
  for(i=1;i<=n;i++){
    sum += i;
  }
  document.write("1~",n,"까지의 합 : ",sum);
</SCRIPT>
```



parseInt()

```
<SCRIPT type="text/javascript">
```

```
document.write("parseInt('15') + parseInt('25') => " + (parseInt('15') + parseInt('25')) + "<br>");
document.write("parseInt('15',8) => " + parseInt('15',8) + "<br>"); //8진수 15를 10진수로 변환
document.write("parseInt('15',16) => " + parseInt('15',16) + "<br>");//16진수 15를 10진수로 변환
document.write("parseInt('21Century') => " + parseInt('21Century') + "<br>");
document.write("parseInt('Century21') => " + parseInt('Century21') + "<br>");
document.write("parseInt('15') + parseInt('25.894') => " + (parseInt('15') + parseInt('25.894'))+
"<p>");
```

```
document.write("Number('21Century') => " + Number('21Century') + "<br>");
document.write("Number('Century21') => " + Number('Century21') + "<br>");
document.write("Number('15') + Number('25.894') => " + (Number('15') + Number('25.894'))+
"<br>");
```

[결과]

```
parseInt('15') + parseInt('25') => 40
parseInt('15',8) => 13
parseInt('15',16) => 21
parseInt('21Century') => 21
parseInt('Century21') => NaN
parseInt('15') + parseInt('25.894') => 40
```

```
Number('21Century') => NaN
Number('Century21') => NaN
Number('15') + Number('25.894') => 40.894
```



parseFloat()

- 문자열을 실수로 변환시켜줌
- 문자를 숫자로 변환하지 못하면 NaN 반환
- 진수를 지정할 수 없고, 오직 10진수로만 변환됨

//parseFloat

```
document.write("parseFloat('15') + parseFloat('25') => " + (parseFloat('15') + parseFloat('25'))  
+ "<br>");  
document.write("parseFloat('34.567.89') => " + parseFloat('34.567.89') + "<br>");  
document.write("parseFloat('34.567e3') => " + parseFloat('34.567e3') + "<br>");  
document.write("parseFloat('21Century') => " + parseFloat('21Century') + "<br>");  
document.write("parseFloat('Century21') => " + parseFloat('Century21') + "<br>");  
document.write("parseFloat('15') + parseFloat('25.456') => " + (parseFloat('15') +  
parseFloat('25.456')) + "<p>");
```

[결과]	parseFloat('15') + parseFloat('25') => 40
	parseFloat('34.567.89') => 34.567
	parseFloat('34.567e3') => 34567
	parseFloat('21Century') => 21
	parseFloat('Century21') => NaN
	parseFloat('15') + parseFloat('25.456') => 40.456



isNaN()

- isNaN(값)
 - NaN : Not a Number
 - 순수한 문자이면 true, 아니면 false를 반환
 - 문자 형태의 숫자도 false 반환
 - 숫자입력만 허용하도록 유효성 검사할 때 유용
 - true : 1, false : 0, null : 0, " " : 0 으로 변환
- isFinite(값)
 - 값이 유한의 수인지 판별하여 유한이면 true, 무한이면 false
- Number("숫자")
 - 문자 형태의 숫자를 숫자로 변환
- String(숫자)
 - 숫자 형태의 숫자를 문자로 변환

예제

```
isNaN('738493') => false  
isNaN(37.478) => false  
isNaN('자바스크립트') => true
```

```
//Number  
var sum;  
sum = Number('15') + Number('25');  
document.write("Number('15') + Number('25') => " + sum+"<p>");  
//String  
document.write("String(15) + String(25) => " + (String(15) + String(25))+"<p>");  
  
//isFinite  
document.write("isFinite(738493) ===>" + isFinite(738493) + "<br>");  
document.write("isFinite('738493') ===>" + isFinite('738493') + "<br>");  
document.write("isFinite(37.478) ===>" + isFinite(37.478) + "<br>");  
document.write("isFinite(3*5+2) ===>" + isFinite(3*5+2) + "<br>");  
document.write("isFinite(0738493) ===>" + isFinite(0738493) + "<br>");  
document.write("isFinite('자바스크립트') ===>" + isFinite('자바스크립트') + "<br>");  
document.write("isFinite('baseball') ===>" + isFinite('baseball') + "<br>");  
document.write("isFinite('738493자바스크립트') ===>" + isFinite('738493자바스크립트')+"<p>");  
  
//isNaN  
document.write("isNaN('738493') => " + isNaN('738493') + "<br>");  
document.write("isNaN(37.478) => " + isNaN(37.478) + "<br>");  
document.write("isNaN('자바스크립트') => " + isNaN('자바스크립트') + "<br>");
```

[결과]

Number('15') + Number('25') => 40

String(15) + String(25) => 1525

isFinite(738493) ===>true

isFinite('738493') ===>true

isFinite(37.478) ===>true

isFinite(3*5+2) ===>true

isFinite(0738493) ===>true

isFinite('자바스크립트') ===>>false

isFinite('baseball') ===>>false

isFinite('738493자바스크립트') ===>>false

예제2

- 숫자 입력만 허용하는 폼 문서
 - 사용자가 숫자 이외의 다른 문자를 입력하면 경고 메시지 출력하기
 - 전화번호, 우편번호 등의 경우 유효성 검사시 이용
- 자리수로 휴대폰 번호 타당성 검사하기

휴대폰번호 : - -

```
<BODY>
<FORM NAME="frmcal">
휴대폰번호 :<INPUT TYPE="text" name="tel1" size="3"> -
              <INPUT TYPE="text" name="tel2" size="5"> -
              <INPUT TYPE="text" name="tel3" size="10"> <P>
<INPUT TYPE="button" value="검사하기" onClick="checkNum(this.form)">
</FORM>
</BODY>
```



예제2

```
<SCRIPT type="text/javascript">
function checkNum(frm)
{ //숫자 입력 체크
  if(isNaN(frm.tel1.value)==true || isNaN(frm.tel2.value)|| isNaN(frm.tel3.value))
  {
    alert("숫자를 입력하세요");
    frm.tel1.value=""; frm.tel2.value=""; frm.tel3.value="";
    frm.tel1.focus();
    return;
  }
  else
  { //자리수 체크
    if(frm.tel1.value.length == 3 && (frm.tel2.value.length == 3 || frm.tel2.value.length == 4)
      && frm.tel3.value.length == 4)
      alert("잘 입력했어요");
    else
    {
      alert("휴대폰 번호가 잘못됐어요");
      frm.tel1.value=""; frm.tel2.value=""; frm.tel3.value="";
      frm.tel1.focus();
      return;
    }
  }
}
}
</SCRIPT></HEAD>
```

코드 실행 함수

- 문자열을 코드로 실행할 수 있는 함수를 제공
- `eval(string)` - `string`을 자바스크립트 코드로 실행함

```
<script>
  // 문자열을 생성합니다.
  var willEval = "";
  willEval += 'var number = 10;';
  willEval += 'alert(number);';

  // eval() 함수를 호출합니다.
  eval(willEval);
</script>
```

• `eval()` 함수는 문자열을 자바스크립트 코드로 실행하는 함수이므로 `eval()` 함수로 실행된 코드에서 정의한 변수도 활용할 수 있음

```
<script>
  // 문자열을 생성합니다.
  var willEval = ";
  willEval += 'var number = 10;';
  willEval += 'alert(number);';

  // eval() 함수를 호출합니다.
  eval(willEval);

  // eval() 함수로 호출한 코드의 변수를 사용합니다.
  alert(number);
</script>
```



eval() 함수

- eval(수식 문자열)

- 문자열로 입력된 계산식의 값을 구하거나 복잡한 스크립트 구문을 문자열로 만들어 실행시키고자 하는 경우 사용
 - 문자열을 수식으로 변환한 후에 계산을 수행함
- 입력양식(form)을 통해 입력 받은 수식이 문자열로 취급되어 계산이 불가능한 경우 eval()함수를 이용하면 계산이 가능해짐
- 수식이 아닌 일반 문자식도 변환하여 평가해 주기도 함
 - 따옴표와 연산자를 없애주는 형태로 변환해 줌



eval() 함수

```
<SCRIPT type="text/javascript">
    f_count = "25+6*10";
    document.write(f_count + " 의 계산결과는 " + f_count + " 입니다<br>");
    document.write(f_count + " 의 계산결과는 " + eval(f_count) + " 입니다<p>");

    f_count1 = "'document.myForm.id'+'.value'";
    document.write("eval()함수를 사용한 경우 : "+eval(f_count1)+"<br>");
    document.write("eval()함수를 사용하지 않은 경우 : "+f_count1+"<br>");
</SCRIPT>
```

[결과]

25+6*10 의 계산결과는 25+6*10 입니다

25+6*10 의 계산결과는 85 입니다

eval()함수를 사용한 경우 : document.myForm.id.value

eval()함수를 사용하지 않은 경우 : 'document.myForm.id'+'.value'



브라우저 객체의 함수



객체의 함수(메서드)

- window 객체의 메서드
 - window.alert()
 - window.confirm()
 - 2개의 버튼 중에서 사용자가 선택한 버튼에 따라서 true, false 값을 리턴함
 - window.prompt()



메서드 – 대화상자 표시

■ 경고창

- alert(“문자열”)
 - [확인]버튼을 가진 대화상자를 화면에 표시
 - 주로 특정 메시지를 전달하고자 하는 경우 사용
- confirm(“문자열”)
 - [확인],[취소]를 선택할 수 있는 대화상자를 표시
 - 주로 “예/아니오”로 대답을 유도하는 경우에 사용
 - [확인]버튼을 누르면 true 값이 반환되고, [취소]버튼을 누르면 false 값이 반환됨
- prompt(“대화상자에 표시할 메시지”, “입력상자에 표시할 초기 문자열”)
 - 문자열을 입력 받을 수 있는 대화상자를 표시
 - 주로 사용자로부터 특정 내용을 입력 받고자 할 때 사용
 - 대화상자를 닫을 때 [확인]버튼을 누르면 입력상자의 내용을 반환하고, [취소]버튼을 누르면 ‘null’값이 반환됨

예제

Explorer 사용자 프롬프트

스크립트 프롬프트:
저는 prompt() 메소드 입니다

확인

취소

초기값

<BODY >

window객체의 메소드를 실행하기 전<P>

<SCRIPT type="text/javascript">

window.alert("저는 alert() 메소드 입니다");

var b = window.prompt("저는 prompt() 메소드 입니다","초기값");

var c = window.confirm("저는 confirm() 메소드 입니다");

document.write(b + "
" + c + "<P>");

if(c==true)

document.write("확인버튼 선택
");

else

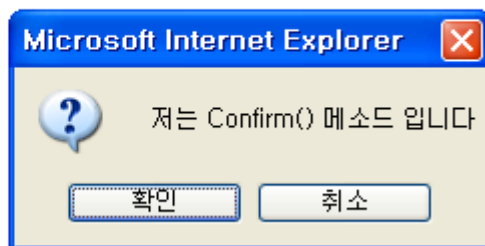
document.write("취소버튼 선택
");

</SCRIPT>

window.method()를 실행한 이후

</BODY>

window객체의 메소드를 실행하기 전



window객체의 메소드를 실행하기 전

150

true

확인버튼 선택

window.method()를 실행한 이후



객체의 함수(메서드)

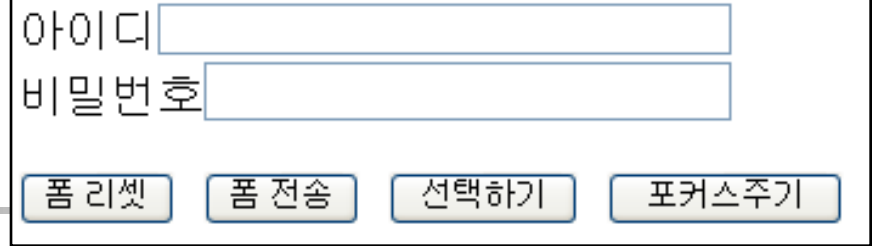
- document 객체의 메서드
 - document.write()
 - document.getElementById()
- 예)
 - document.write(i + "*" + j + "=" + i*j)
 - document.write(i, "*", j, "=", i*j)



객체의 함수(메서드)

- form 객체의 메서드
 - form이름.reset()
 - form이름.submit()
- text 객체의 메서드
 - form이름.입력상자이름.focus()
 - form이름.입력상자이름.select()

예제



아이디

비밀번호

```
<HTML>
<HEAD>
<META http-equiv="content-type" content="text/html; charset=euc-kr">
<TITLE> Form객체와 Text객체의 Method </TITLE>
<SCRIPT type="text/javascript">
    function hamsuA(){
        document.frm.reset();
    }
    function hamsuB(){
        document.frm.submit();
    }
    function hamsuC(){
        document.frm.id.select();
    }
    function hamsuD(){
        document.frm.pwd.focus();
    }
</SCRIPT>
</HEAD>
```



예제

<BODY>

<FORM name="frm" method="post" action="test.jsp">

아이디<INPUT type="text" name="id" size=32>

비밀번호<INPUT type="password" name="pwd" size=32> <P>

<input type = "button" value="폼 리셋" onclick="hamsuA();">

<input type = "button" value="폼 전송" onclick="hamsuB();">

<input type = "button" value="선택하기" onclick="hamsuC();">

<input type = "button" value="포커스주기" onclick="hamsuD();">

</FORM>

</BODY>

</HTML>



window 객체



window 객체

- window 객체
 - 최상위에 위치하는 Top 레벨 객체
 - 브라우저 윈도우나 프레임 윈도우를 표현하는 객체
 - 현재 화면에 나타나는 화면의 상태, 화면의 정보, 히스토리 정보, 배경, 색, 폰 등 다양한 정보 제공
 - 거의 대부분의 웹 브라우저 및 객체가 window객체를 기반으로 이루어짐
 - 보통 생략하고 사용하는 경우가 많음
 - 하나의 html 문서가 실행되면 자동으로 하나의 window 객체가 생성됨
 - 프레임으로 구성된 경우에는 프레임 당 하나씩의 window 객체가 더 생성됨 (3개의 프레임으로 구성된 문서 – 총 4개의 window 객체가 존재함)

window 객체

```
<head>
  <script>
    // 출력합니다.
    var output = "";
    for (var key in window) {
      output += key + '\n';
    }
    alert(output);
  </script>
</head>
```

■ window 객체

- 브라우저 기반 자바스크립트의 최상위 객체
- 메서드
 - alert()
 - prompt()
 - open() 등

top
window
location
external
chrome
document
output
key
speechSynthesis
webkitStorageInfo
indexedDB
webkitIndexedDB
crypto
localStorage
sessionStorage
applicationCache
CSS
performance
console
devicePixelRatio
styleMedia
parent
opener
frames
self
defaultStatus
defaultStatus
status
name
length
closed
pageYOffset
pageXOffset
scrollY
scrollX
screenTop
screenLeft

var 키워드로 선언한 일반 변수도 모두 **window** 객체의 속성이 됨

```
var num=100;  
//window.num
```

window 객체의 메서드

메서드	설명
alert()	경고 메시지를 가진 대화상자를 표시
confirm(문자열)	확인, 취소를 선택할 수 있는 대화상자를 표시함
prompt(메시지, 초기문자열)	문자열을 입력 받을 수 있는 대화상자를 표시함
open()	새로운 창을 연다
close()	열려진 창(윈도우)를 닫는다
setInterval()	일정 간격으로 함수를 호출하여 실행함 (주기적)
setTimeout()	정해진 시간이 지난 후 함수를 호출함 (1회성)
clearInterval(시간ID)	setInterval() 에 의해 설정된 타이머를 해제함
clearTimeout(시간ID)	setTimeout() 에 의해 설정된 타이머를 해제함
print()	화면에 있는 내용을 프린터로 출력함
blur()	윈도우의 포커스를 없앤다
focus()	윈도우에 포커스를 줌
moveBy(x좌표, y좌표)	윈도우를 현재 위치에서 상대적인 좌표 위치로 이동함
moveTo(x좌표, y좌표)	윈도우를 절대적인 좌표 위치로 이동함
resizeBy(폭, 높이)	윈도우를 현재 크기에서 상대적인 크기만큼 변경함
resizeTo(폭, 높이)	윈도우를 절대적인 크기로 변경함
scrollBy(x좌표, y좌표)	윈도우를 현재 위치에서 상대적인 좌표 위치로 스크롤시킴
scrollTo(x좌표, y좌표)	절대적인 좌표 위치로 스크롤 시킴



새로운 window 객체 생성

- window 객체는 open() 메서드로 새로운 window 객체를 생성함

open(URL, name, features)

- 새로운 window 객체를 생성함

- open() 메서드의 모든 매개변수는 옵션임(입력해도 되고 입력하지 않아도 되는 매개변수)

```
<script>  
    window.open();  
</script>
```

```
<script>  
    window.open('http://www.naver.com', 'child', 'width=500, height=300');  
</script>
```

자동으로 **naver**의 팝업창이 뜸

open() 메서드

- 문서가 로드되면서 별도의 브라우저 창을 띄우기 위해 사용되는 메서드
 - 공지창, 이벤트 창, 안내문구, 우편번호 검색 등의 표시에 주로 사용
- 형식

```
window.open(“연결할 URL”, “창이름”, “창의 속성”);
```

- 연결할 URL – 새로 만들어지는 창에 로드할 웹 문서를 입력
 - 공백(“ ”)을 주면 비어 있는 창이 만들어짐
- 창이름 – 새로 생성되는 창의 이름을 지정하는 것
 - html문서의 target 이름으로 사용됨
 - 윈도우 간 통신하는 데 사용하는 윈도우 이름
- 창의 속성
 - 새로 생성되는 창의 모양과 같은 속성을 지정
 - 윈도우를 어떠한 모양으로 출력할지 지정하는 옵션
 - 여러 속성들을 한번에 지정하는 경우에는 속성 사이에 빈칸 없이 쉼표(,)로 분리

open() 메서드 - 창의 속성

옵션	값	설명
menubar	yes/no 또는 1/0	윈도우의 menubar(메뉴표시줄) 표시 여부
toolbar	yes/no 또는 1/0	윈도우의 toolbar(도구모음) 표시 여부
location	yes/no 또는 1/0	윈도우의 location box(주소표시줄) 표시 여부
status	yes/no 또는 1/0	윈도우의 상태 표시줄 표시 여부
scrollbars	yes/no 또는 1/0	윈도우의 가로, 세로 scrollbar 표시 여부
resizable	yes/no 또는 1/0	윈도우의 크기가 조절될 수 있는지 결정
width	number (단위 pixel)	윈도우의 너비 결정
height	number (단위 pixel)	윈도우의 높이 결정
left	number (단위 pixel)	윈도우의 좌측 상단의 x좌표 위치 결정
top	number (단위 pixel)	윈도우의 좌측 상단의 y좌표 위치 결정

- width, height, left, top을 제외한 나머지 옵션은 익스플로러에서만 지정할 수 있음
- 크롬이나 파이어폭스에서는 location=yes, resizable=yes 가 항상 적용되어 나타남

open() 메서드

- open() 메서드

- 새로운 window 객체를 생성하는 메서드
- 단지 팝업창을 여는 것에서 끝나지 않고 **윈도 객체를 리턴함**
- 새로운 윈도 객체에 접근해 속성과 메서드를 사용할 수 있음

```
<script>  
    // 변수를 선언합니다.  
    var child = window.open("", "", 'width=300, height=200');  
  
    // 출력합니다.  
    child.document.write('<h1>From Parent Window</h1>');  
</script>
```



```
<script>
  // 변수를 선언
  var child = window.open("", "", 'width=300, height=200');

  // 출력
  if (child) {
    child.document.write('<h1>From Parent Window</h1>');
  } else {
    alert('팝업 차단을 해제해주세요.');
```

```
</script>
```

자바스크립트 알림

팝업 차단을 해제해주세요.

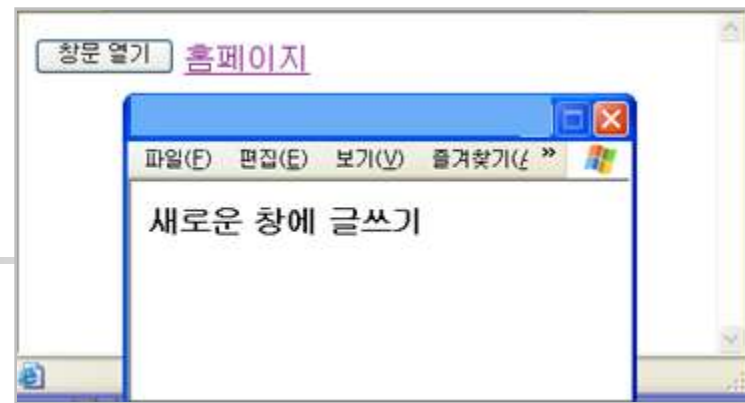
확인

예제

```
<HEAD>
<script>
var mywin;
function winOpen()
{
    window.open("event1.html", "ev", "width=300,height=200,location=yes,resizable=yes,
    left=0,top=0");
}
function winOpen2()
{
    mywin = window.open("", "notice", "width=300,height=200");
    mywin.document.write("새 창에 글쓰기");
    mywin.document.bgColor="skyblue";
}
function winClose()
{
    mywin.close();
}
</script>
</HEAD>
<BODY onload="winOpen()">
<input type="button" value="새창열기" onclick="winOpen2()">
<a href="#" onclick="winClose()">새창닫기</a>
<a href="http://www.naver.com" target="ev">네이버로 이동</a>
</BODY>
```

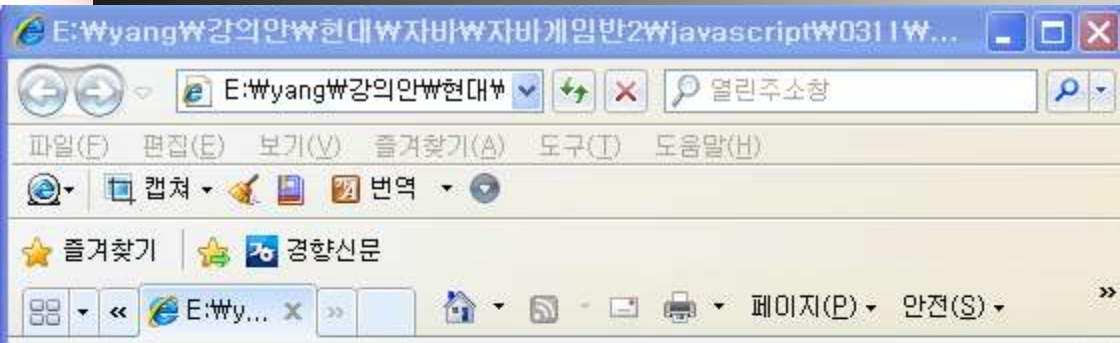


예제

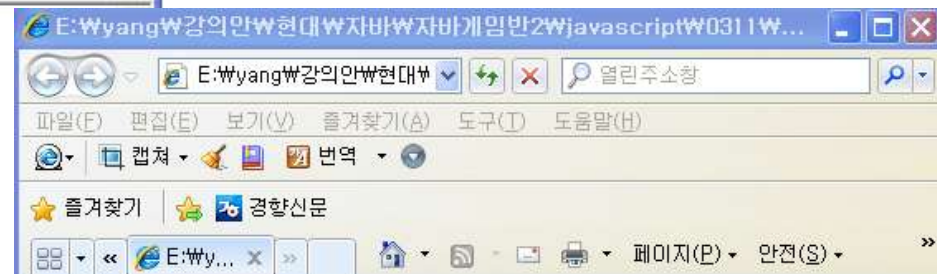
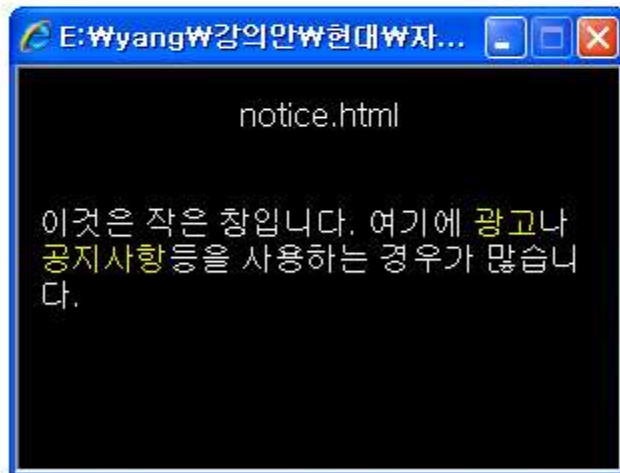


```
<HTML>
<HEAD>
<TITLE> open() 메소드의 3번째 매개변수 </TITLE>
<SCRIPT type="text/javascript">
    function hamsu(){
        win = window.open("", "op", "menubar=yes, width=300, height=150" );
        win.document.write("<H3>새로운 창에 글쓰기</H3>");
    }
</SCRIPT>
</HEAD>
<BODY>
    <BUTTON onclick="hamsu()"> 창문 열기 </BUTTON>
    <A href="http://www.naver.com" target="op"> 홈페이지 </A>
</BODY>
</HTML>
```

실습-공지사항 창 띄우기



새창에 네이버 띄우기



새창에 네이버 띄우기





opener 속성

■ opener 속성

- 현재 윈도우를 연 윈도우
- window 객체의 속성으로 open()메서드를 이용하여 자신을 생성한 부모 윈도우 객체를 가리킬 때 사용
- open()에 의해 생성된 윈도우가 자신을 생성한 윈도우를 제어하는 경우에 주로 사용



예제

```
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
    window.open("sample.htm", "fun", "width=400,height=400")
</SCRIPT>
</HEAD>
<BODY>
<form name="frmtest">
주소 : <input type="text" name = "address" size = "50">
</form>
</BODY>
</HTML>
```

예제

[sample.htm]

```
<HEAD>
<SCRIPT type="text/javascript">
function sam()
{
    window.opener.document.bgColor='cyan';
}
function send()
{
    opener.document.body.style.backgroundColor="skyblue";
    opener.document.frmtest.address.value=document.frm1.dong.value;
    self.close();
}
</SCRIPT>
</HEAD>
<BODY>
<A HREF="#" onclick="sam()">원래창의 배경색 바꾸기</A>
<form name="frm1">
주소 입력 : <input type="text" name = "dong">
<input type="button" value="확인" onclick="send();">
</form>
</BODY>
```





close() 메서드

- 열려진 창을 닫는 메서드로 다른 속성이나 메서드와는 달리 **닫고자 하는 창의 객체를 반드시 사용**해야 함

창 객체.close();


- window.close();
- self.close();
- myWin = window.open("", "sample", "width=400,height=300,left=0,top=0");
- myWin.close();

실습

우편번호 : -

주소 :

나머지 주소 :



동 입력 :



setInterval()메서드

■ setInterval()

- 일정한 시간 간격으로 원하는 작업을 실행하고자 할 때 사용
- setInterval() 메서드의 리턴값은 타이머ID로서 설정된 타이머를 해제하기 위한 clearInterval()메서드의 매개변수로 사용됨

```
TimerID = setInterval(“실행할 함수”, 시간)  
clearInterval(TimerID);
```

■ clearInterval()

- setInterval()메서드로 설정한 시간을 **해제**하는 기능
- setInterval() 메서드의 리턴값인 타이머ID를 매개변수로 사용함

```
clearInterval(TimerID);
```




setInterval()메서드

2014년 3월 12일 수요일 오전 10:56:12

```
<HEAD>
```

```
<SCRIPT type="text/javascript">  
setInterval('clock()', 1000);
```

```
function clock()  
{  
    today=new Date();  
    document.myForm.myClock.value=today.toLocaleString();  
}
```

```
</SCRIPT>
```

```
</HEAD>
```

```
<BODY>
```

```
<FORM NAME="myForm">
```

```
    <INPUT NAME="myClock" TYPE="Text" SIZE=30>
```

```
</FORM>
```

```
</BODY>
```

clearInterval()메서드

setInterval 과 clearInterval 예

지금은 11 시 7 분 55 초입니다.

```
<HEAD>
<SCRIPT type="text/javascript">
    var Time_ID;
    function displayTime()
    {
        today = new Date();
        document.getElementById("curTime").innerHTML = "지금은 " + today.getHours() + " 시 "
+ today.getMinutes() + " 분 " + today.getSeconds() + " 초입니다.";
    }
    function startTime()
    {
        Time_ID = setInterval("displayTime()", 1000);
    }
    function stopTime()
    {
        clearInterval(Time_ID);
    }
</SCRIPT></HEAD>
<BODY>
<H2> setInterval 과 clearInterval 예 </H2>
<FORM>
    <INPUT TYPE="button" VALUE="시간 표시" onClick="startTime()">
    <INPUT TYPE="button" VALUE="시간 중지" onClick="stopTime()"><br><br>
</FORM><div id="curTime"></div>
```



setTimeout()

■ setTimeout()

- 정해진 시간이 지나면 특정 함수를 **1회에 한해 호출**하는 메서드
- setTimeout() 메서드의 리턴값은 타이머ID로서 설정된 타이머를 해제하기 위한 clearTimeout()메서드의 매개변수로 사용됨

```
TimerID = setTimeout(“실행할 함수”, 일시 중지될 시간);
```

- 일시 중지될 시간 – 함수를 실행하기 위한 정해진 시간을 의미, 1/1000 초 단위로 지정함

■ clearTimeout()

- setTimeout()으로 설정한 시간을 해제하는 기능
- setTimeout() 메서드의 리턴값인 타이머ID를 매개변수로 사용함

```
clearTimeout(TimerID);
```



setTimeout()

- **setTimeout()** 메서드 – **한 번만 함수를 호출**
 - setTimeout() 메서드는 해당 함수 안에 존재하며 매번 설정을 해야 함
- **setInterval()** 메서드 – 함수를 **반복해서 호출**
 - 독립적으로 존재하며 한 번만 설정하면 됨

setTimeout()

<HEAD>

<SCRIPT>

```
function winClose()
{
    myWin.close();
}
```

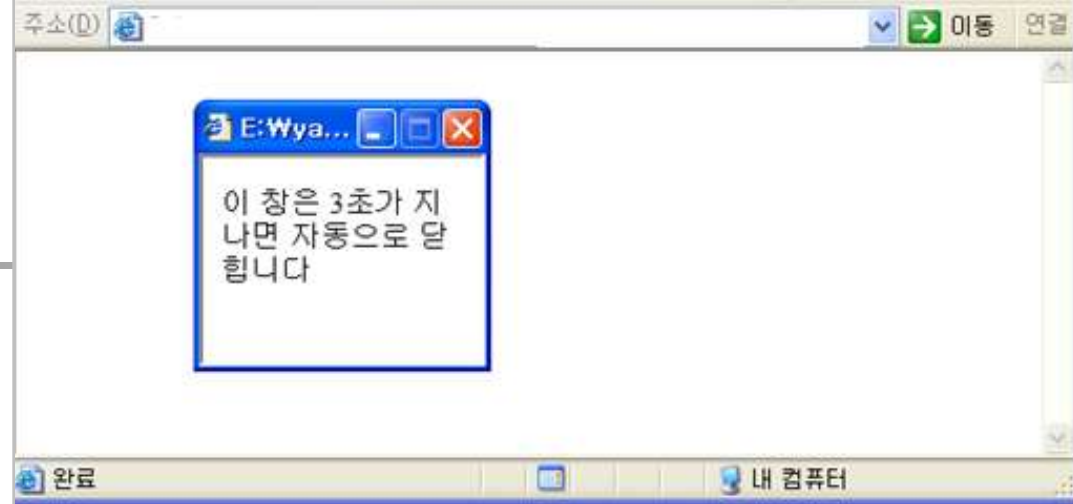
```
myWin = window.open("", "tinyWindow", 'width=150, height=110');
myWin.document.write("이 창은 3초가 지나면 자동으로 닫힙니다");
setTimeout("winClose()", 3000);
```

</SCRIPT>

</HEAD>

<BODY>

</BODY>



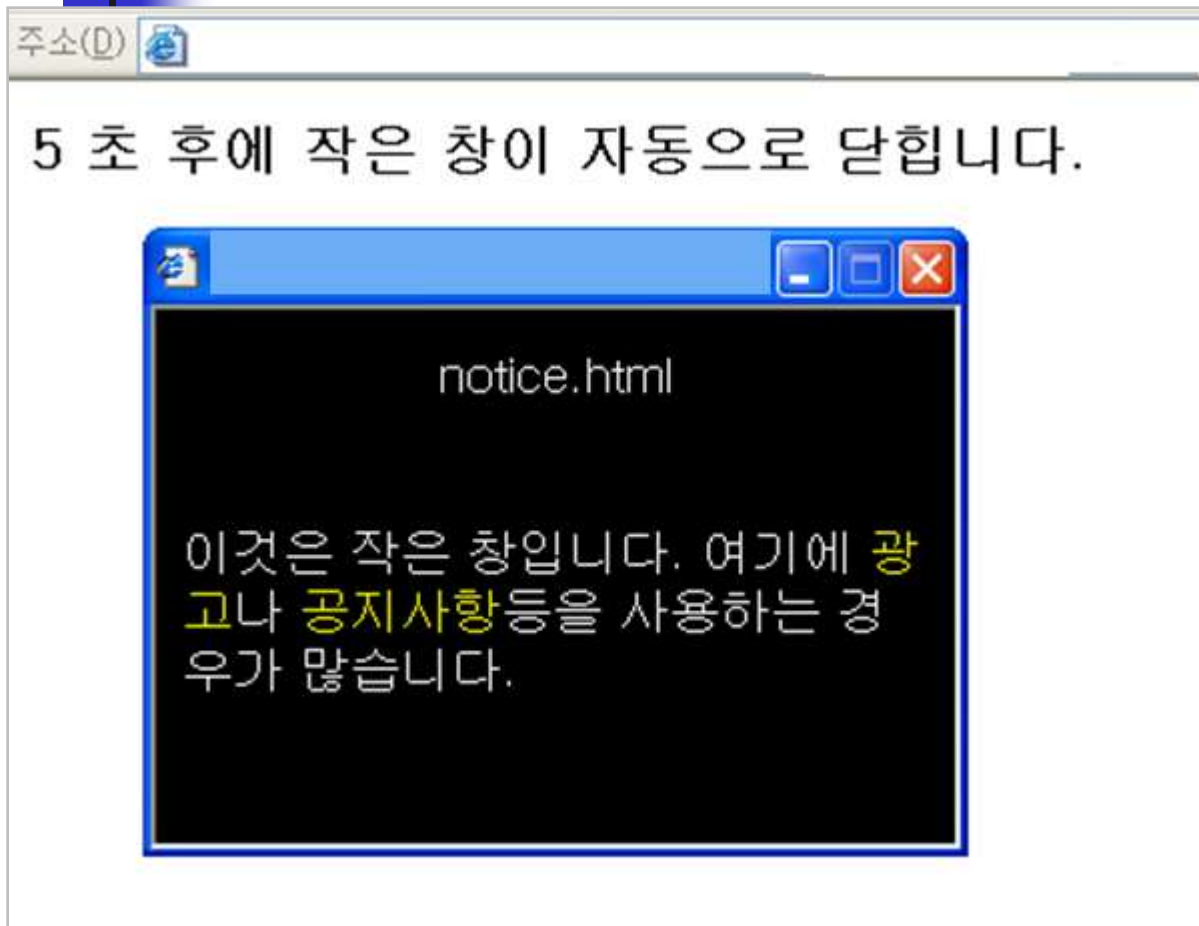
clearTimeout()

setTimeout 과 clearTimeout 예

지금은 11 시 14 분 9 초입니다.

```
<HEAD>
<SCRIPT type="text/javascript">
  function startTime()
  {
    today = new Date();
    document.getElementById("curTime").innerHTML = "지금은 " + today.getHours() + " 시 "
    + today.getMinutes() + " 분 " + today.getSeconds() + " 초입니다.";
    Time_ID = setTimeout("startTime()", 1000);
  }
  function stopTime()
  {
    clearTimeout(Time_ID);
  }
</SCRIPT>
</HEAD>
<BODY>
<H1> setTimeout 과 clearTimeout 예 </H1>
<FORM>
<INPUT TYPE="button" VALUE="시간 표시" onClick="startTime()">
<INPUT TYPE="button" VALUE="시간 중지" onClick="stopTime()">
</FORM>
<div id="curTime"></div>
</BODY>
```

실습 - 5초 후 자동으로 닫히는 윈도우





실습1

```
<div id="output">  
  0  
</div>
```

- 1초에 한 번씩 변수 값을 1씩 증가시키고 이 값을 #output 영역에 출력하기

3

실습2

- 1. 물고기 움직이기
 - 버튼을 누르면 x, y에 입력된 값만큼 물고기를 움직여주세요.
 - 단, x값이 0~500, y는 0~300이 넘는 값이 입력되면 "입력된 값이 너무 큼니다. 다시 입력해주세요." 라고 출력해 주세요.
- 물고기를 움직이는 방법
 - fish의 style 지정
 - left: x축 이동 위치 값,
 - top: y축 이동 위치 값



x값 입력 : 100

y값 입력 : 150

물고기 움직이기

실습3

이미지 배열 이용

for문에서
이미지 가로 위치 구하기
`x = 100 + 0`
`x = 100 + 150;`
`x = 100 + 300;`
.....

- 1. 이미지를 가로로 배열하기
- 버튼을 클릭하면 이미지를 요구사항에 맞게 배열해 주세요.

- 요구사항
- 01. 이미지 시작위치는 left:100, top:100 입니다.
- 02. 하나의 이미지 영역(이미지 크기와 여백 포함)은 150*150입니다.
- 03. 이미지를 가로로 배열해 주세요.

- 힌트 :
- 이미지 위치 설정하기
- style
- left:위치값,
- top:위치값

배열시작

Google

배열시작

Pinterest

facebook

twitter

Path

Google

실습4

setInterval()
Math.random() 이용
Img 태그에 접근해서 src 변경

- 1. 랜덤으로 배경이미지 변경하기
 - 1초에 한 번씩 배열에 들어 있는 이미지를 랜덤 순으로 전환되게 만들어 주세요.

```
var imgList1 = ["logo_01.jpg","logo_02.jpg","logo_03.jpg","logo_04.jpg","logo_05.jpg"];  
var imgList2 = ["logo_02.jpg","logo_05.jpg","logo_03.jpg","logo_04.jpg","logo_01.jpg"];
```





실습5

- 1. 타이머 함수를 활용한 “물고기 잡기 게임” 만들기
- 요구사항
- 1. 5초 동안 누가 많이 물고기를 클릭하는지 알아보는 간단한 게임 만들기
- 2. 물고기 클릭 시 1점 획득.
- 3. 게임이 종료된 후 물고기 클릭 시 게임 점수가 올라가서는 안 됨
- 4. 점수는 획득 시 바로 화면에 출력한다.

이미지를 클릭하면

`count` 변수의 값을 1씩 증가시켜서 현재점수 에 출력

단, `boolean` 변수의 값이 `true`일때만 진행

`setTimeout()`을 이용해서 5초 후에는 `boolean` 변수의 값을 `false`로 변경

현재점수 24

이 페이지 내용:

게임이 종료 되었습니다.

확인

