

JavaScript1.2

Park Yong Jun(RedPlus)
MCT, MVP, MCSD, MCAD, MCP,
정보처리기사, 전자상거래관리사
<http://www.dotnetkorea.com/>

◆ JavaScript 목차

■ JavaScript

- 자바스크립트 개요 - 자바스크립트란?
- 자바스크립트의 시작 및 작성
- 데이터와 정보 처리(변수, 연산자, 주석)
- 조건문(if문, switch문)
- 반복문(for문, while문)
- 함수(function) : 기본제공함수, 사용자정의함수
- 자바스크립트의 객체 모델(주요 객체)
- 배열(Array객체 사용)
- 이벤트(Event) : 마우스 제어, 키보드 제어
- 대화형 폼의 개발(HTML과의 연동)
- window, document, location, history, form 객체

◆ 자바스크립트 개요

- 자바스크립트란?
- 자바스크립트의 기능
- 자바스크립트와 자바와의 차이점
- 자바스크립트의 장점
- 자바스크립트의 단점

자바스크립트란?

■ 클라이언트 및 인터넷 응용 프로그램 개발을 위한 객체-지향 스크립팅 언어

■ 스크립팅(Scripting) 언어

- 인터프리터(Interpreter)에 의해 실행됨
- 손쉬운 프로그램 개발과 제한적이지만 배우기 쉬운 명령어들과 문장 구조
- 잘 정의된 일련의 작업을 수행하는 프로그램의 개발이 용이함

■ 단순하고 작은 프로그램을 위한 언어

- 단순하고 작은 프로그램 개발에 적합
- 분산된 CAD 문서 출력과 관리와 같은 환경에는 부적합

■ 반복적인 작업 수행

- 반복적이고 발생 사건(Event)에 의해 수행되는 일에 적합

자바스크립트란?

■ 사용자에게 의해 발생한 사건을 프로그래밍하기 위한 언어

- 웹 브라우저에 포함된 방식 및 **HTML** 문서와 직접 상호 작용 가능
- 마우스 선택과 폼에 데이터를 입력하는 등의 사용자에게 의해 발생하는 사건에 대해 응답하는 프로그램을 작성하는 것을 가능
- 웹 페이지에 대화형 기능 추가 및 역동적인 폼의 제작
- **CGI** 프로그래밍에 의해 야기된 서버의 부담을 줄이는 것이 가능

■ 손쉬운 디버깅

- 인터프리터 방식

■ 자바를 포함하는 언어

- 미래의 자바스크립트 버전에 포함될 내용

자바스크립트란?

■ 자바스크립트란?

- 자바스크립트는 자바의 기능을 그대로 사용하면서도, 매우 간편하고 쉽게 프로그래밍을 할 수 있도록 한 언어
- 프로그래밍에 별로 경험이 없는 사람도 쉽게 프로그래밍할 수 있는 언어
- MS 인터넷 익스플로러 3.0을 발표하면서 자바스크립트 지원.
- 원래 넷스케이프사에서는 클라이언트 쪽에서 독립적으로 실행되는 프로그램을 작성하기 위한 스크립트 언어. 그러나, 자바가 선풍적인 인기를 끌기 시작하자, 넷스케이프사는 자바를 만든 선 마이크로시스템즈사(Sun Microsystems)와 공동 프로젝트를 진행하여 LiveScript를 확장시킨 JavaScript를 제작

자바스크립트의 기능

■ 자바스크립트의 기능

- 자바스크립트는 객체 지향 스크립트 언어로 프로그램 코드가 **HTML** 문서 사이에 직접 들어가게 된다. 사용자가 마우스를 클릭하거나 키보드를 입력하는 것과 같은 작업을 즉시 처리.
- 클라이언트에서만 실행:데이터 전송 없이 모든 작업 처리
- 사용자가 페이지를 열거나 이동할 때를 알아내어 원하는 작업수행
- 프로그램 자체에 관한 정보를 알아낸다거나 플러그 인 정보를 알아낼 때에도 사용될 수 있다.
- 프레임을 조절한다거나, 독립윈도우를 띄워 특정 정보를 전달하거나 방문했던 페이지를 기록해 놓은 히스토리를 관리하는 등의 작업들도 가능

자바스크립트와 자바와의 차이점

자바스크립트	자바
컴파일되지 않은 소스로 클라이언트가 사용하는 browser에 의해 interpreter	프로그래머에 의해서 컴파일된 소스가 클라이언트에 의해 interpreter
HTML과 결합되어 사용	HTML과 별도로 존재
데이터형의 구별이 엄격하지 않음	데이터형의 구별이 엄격
동적으로 바인딩	정적으로 바인딩
소스를 숨길 수 없다.	컴파일되므로 소스를 감출 수 있다.
브라우저가 가지고 있는 객체들을 최대한 이용할 수 있다.	브라우저가 가지고 있는 객체나 기능을 이용할 수 없다.

자바스크립트의 장점

■ 빠른 개발이 가능하다

- 컴파일 과정이 필요없다

■ 배우기가 쉽다

- 자바와 유사한 점도 많지만 자바의 복잡한 문장 구조와 규칙들을 채택하지 않음

■ 사용 컴퓨터에 무관하게 프로그램이 수행된다

- 넷스케이프 2.0이 수행되는 모든 컴퓨터에서 동일하게 수행됨

■ 웹 서버에 주는 부담이 적다 : 성능향상

- 자바스크립트 프로그램이 **HTML** 코드와 동일한 파일에 존재하기 때문에 스크립트 전송을 위한 별도의 네트워크 접속이 필요없다
- 자바에 의해 생성된 바이트코드(Bytecode)보다 집약적이고 작기 때문에 사용자 컴퓨터에 전송하는데 걸리는 시간이 적다

자바스크립트의 단점

■ 자바스크립트의 단점

- 내장된 메서드의 기능이 제한적이다
 - 웹 문서와 클라이언트 윈도우를 처리하기 위한 메서드 부족
- 원시 코드를 감출 수 없다
 - 자바스크립트의 원시 코드는 **HTML** 코드의 일부로 포함되기 때문에 이 웹 페이지를 읽은 사람은 원시 코드를 복사해 재사용할 수 있다
- 디버깅 도구와 개발 도구의 부족

복습

■ 자바스크립트란?

- 클라이언트 및 인터넷 응용 프로그램 개발을 위한 객체-지향 스크립팅 언어
- 스크립팅(Scripting) 언어
- 단순하고 작은 프로그램을 위한 언어
- 반복적인 작업 수행
- 사용자에게 의해 발생한 사건을 프로그래밍하기 위한 언어

◆ 자바스크립트의 시작

■ 무엇부터 시작할 것인가?

- JavaScript 작성도구(편집툴) 준비
 - 메모장, 에디트플러스, 나모웍에디터, 드림위버, 기타
- JavaScript 실행 프로그램(웹브라우저)에 대한 이해(선수학습)
 - HTML/CSS, 인터넷 익스플로러, 넷스케이프 네비게이터

■ JavaScript의 완성

- 좋은 저작도구가 있는데 왜 소스를 공부하는가?
 - 하부적인 구조 이해, JavaScript/ASP&ASP.NET과 상호작용
- 어느 정도의 능력을 요구하는가?
 - 주요객체와 이벤트의 적절한 사용
 - ASP/ASP.NET과의 상호작용을 위한 기능 구현(입력양식 처리)

HTML 문서 안에 자바스크립트 삽입하기 1

■ HTML 문서 안에 자바스크립트 삽입하기

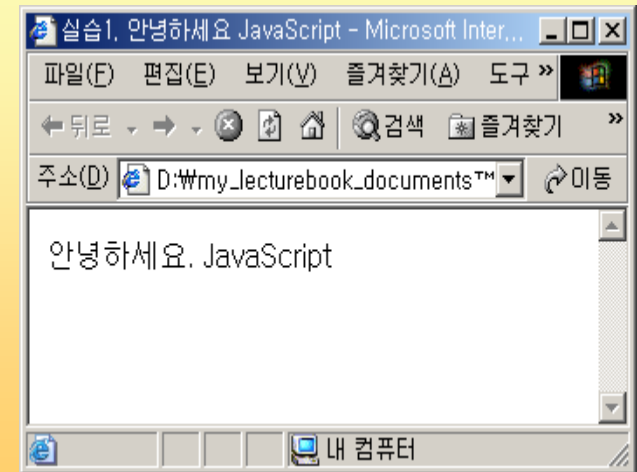
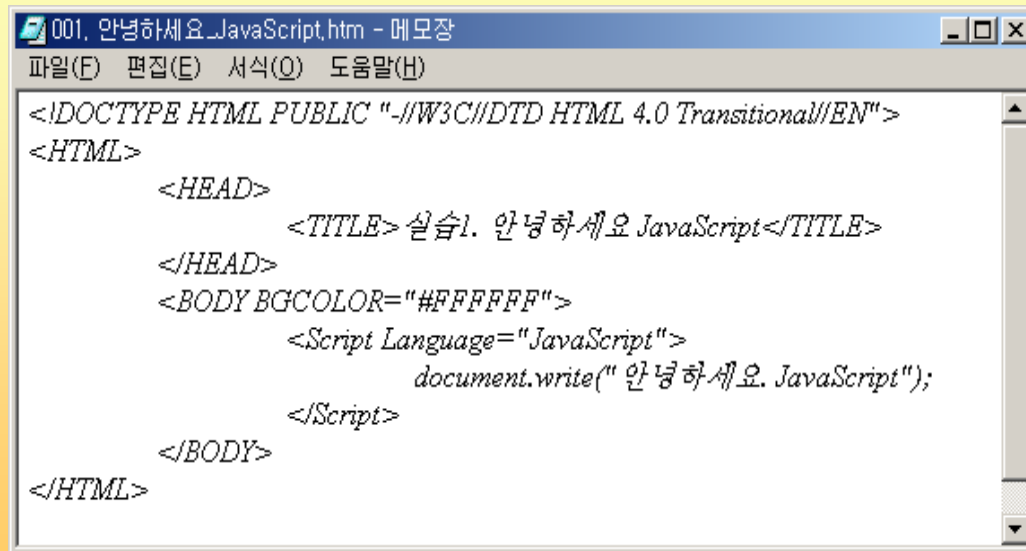
- `<SCRIPT LANGUAGE="JavaScript"> </SCRIPT>`를 이용
- 자바스크립트를 HTML 파일에 삽입하기

```
<SCRIPT LANGUAGE="JavaScript">  
.... JavaScript 프로그램 ....  
</SCRIPT>
```

실습 1 : 안녕하세요 JavaScript

■ 실습1

- `document`객체의 `write`라는 메서드를 사용해서 화면에 “안녕하세요 JavaScript”를 출력시켜보자.
- 현재 `document`객체니 `write`메서드는 배우지 않았다. 하지만, 바로 작성을 해보면 알 수 있을 것이다.



HTML 문서 안에 자바스크립트 삽입하기 2

■ 외부 파일에 존재하는 자바스크립트 프로그램 이용하기

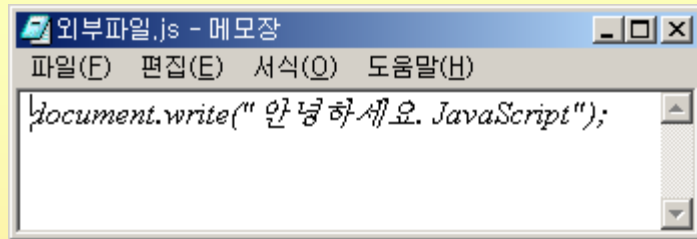
```
<SCRIPT LANGUAGE="JavaScript"  
    SRC="http://www.dotnetkorea.com/myScript.js">  
</SCRIPT>
```

- SRC 속성이 기능을 발휘하기 위해서는 자바스크립트를 포함하고 있는 파일의 이름이 반드시 '.js'로 끝나야 한다

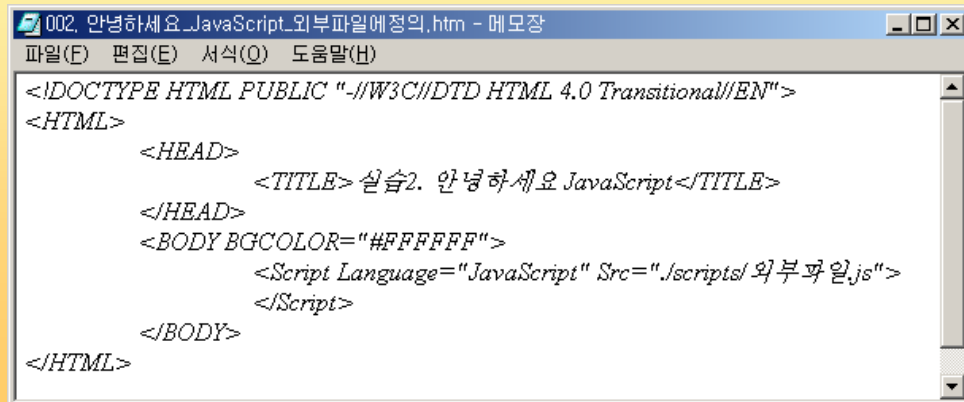
실습 2 : 안녕하세요 JavaScript – 외부파일에 정의

■ 실습 2

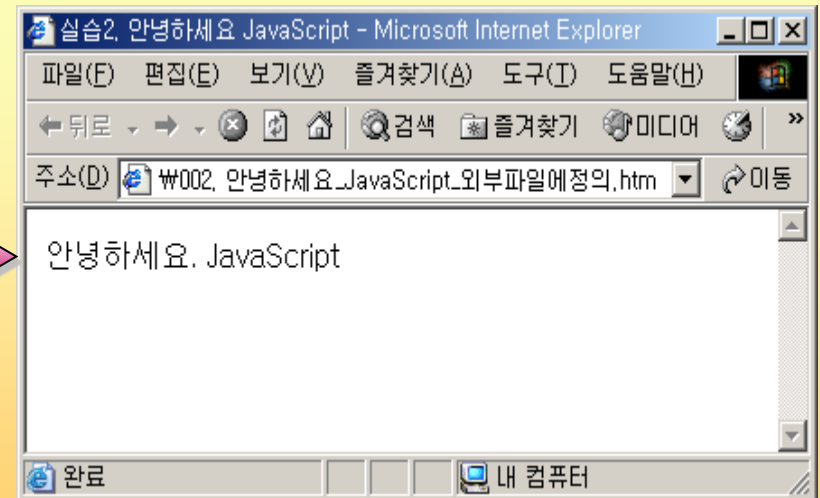
- “안녕하세요 JavaScript”를 외부파일에 미리 정의 시켜놓고, 원본 HTML문서에서 읽어오는 프로그램을 작성해보자.



```
document.write("안녕하세요. JavaScript");
```



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE>실습2. 안녕하세요 JavaScript</TITLE>
  </HEAD>
  <BODY BGCOLOR="#FFFFFF">
    <Script Language="JavaScript" Src="/scripts/외부파일.js">
    </Script>
  </BODY>
</HTML>
```



HTML 문서 안에 자바스크립트 삽입하기 3

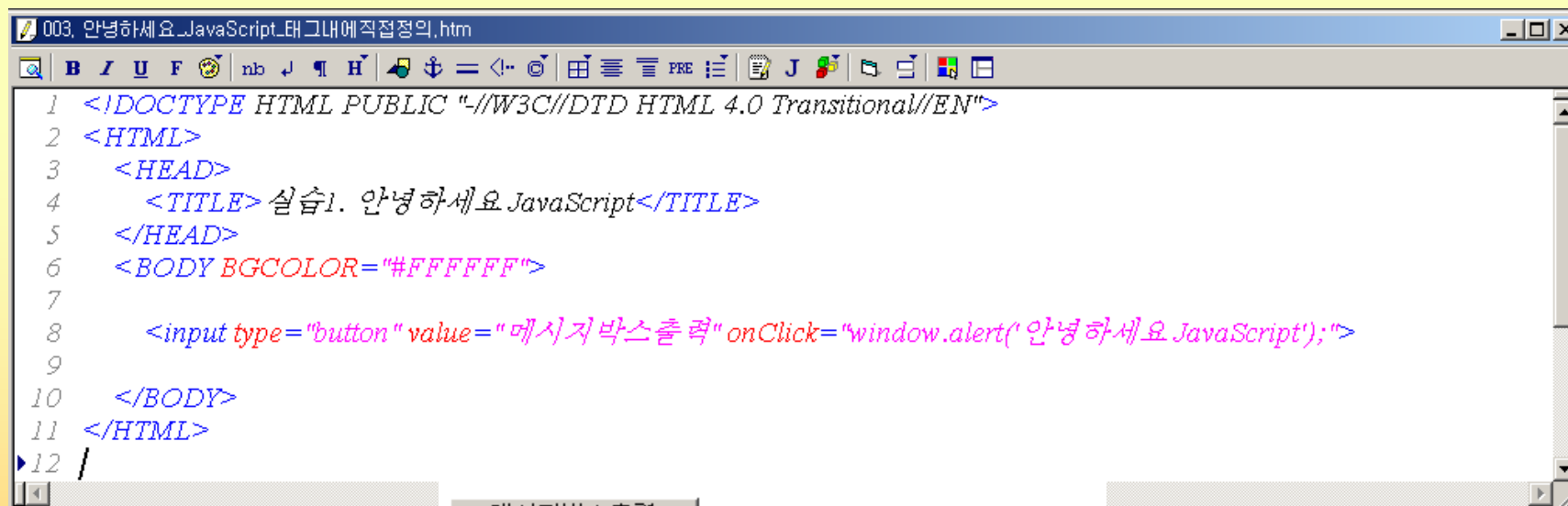
- HTML 태그의 속성(이벤트처리기)에 속성값(자바스크립트 코드)으로 사용

<태그명 이벤트핸들러="자바스크립트코드"> ... </태그명>

실습 3 : 태그내에 직접 자바스크립트 코드 삽입하기

■ 실습 3

- 버튼을 클릭하면 메시지박스를 띄운다.



The screenshot shows a web browser window titled "003. 안녕하세요 JavaScript_태그내에직접정의,html". The address bar shows a local file path. The browser's content area displays the HTML source code of the file. The code is as follows:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2 <HTML>
3   <HEAD>
4     <TITLE> 실습1. 안녕하세요 JavaScript</TITLE>
5   </HEAD>
6   <BODY BGCOLOR="#FFFFFF">
7
8     <input type="button" value="메시지 박스출력" onClick="window.alert('안녕하세요 JavaScript');">
9
10  </BODY>
11 </HTML>
12 /
```

메시지박스출력



복습

■ HTML에 자바스크립트 삽입

- `<Script>...</Script>` 사용 : `<head>` 또는 `<body>`태그내에 사용
- 외부파일에 자바스크립트 소스코드 작성 : 확장자는 반드시 `*.js`
- 태그내에 자바스크립트 코드 적용(이벤트 핸들러 사용) : `on`이벤트명

◆ 자바스크립트 작성시 주의할 점

- 자바스크립트의 버전 표시
- 자바스크립트 작성시 **Script**태그의 문제점
- 자바스크립트 문장의 구조
 - 주석
 - 기본 구조
- 자바스크립트 시작 전에 이해 해야 할 4가지 용어
 - 객체
 - 속성
 - 메서드
 - 이벤트

자바스크립트의 버전 표시

■ 자바스크립트의 버전 표시

- 1.0 : Netscape 2.0과 함께 발표된 초기 버전
- 1.1 : Netscape 3.0과 함께 발표된 버전
- 1.2 : Netscape 4.0과 함께 발표된 버전
- 1.3 : Netscape 4.5 등 자바스크립트 1.3을 지원하는 웹브라우저에서 사용
- 특정 버전은 아래와 같이 표시
 - `<script language = "JavaScript1.2"> ... </script>`

실습 1 : JavaScript의 버전 표시

■ 실습 1

- 자바스크립트 코드를 작성할 때 버전을 표시해 보자.

```
1  <HTML>
2  <HEAD>
3  <TITLE> 실습1. 자바스크립트 버전 표시</TITLE>
4  </HEAD>
5
6  <BODY BGCOLOR="#FFFFFF">
7
8  <SCRIPT LANGUAGE="JavaScript">
9      document.write("여기에 작성되는 자바스크립트 코드는 1.0<br>");
10 </SCRIPT>
11
12 <SCRIPT LANGUAGE="JavaScript1.1">
13     document.write("여기에 작성되는 자바스크립트 코드는 1.1<br>");
14 </SCRIPT>
15
16 <SCRIPT LANGUAGE="JavaScript1.2">
17     document.write("여기에 작성되는 자바스크립트 코드는 1.2<br>");
18 </SCRIPT>
19
20 <SCRIPT LANGUAGE="JavaScript1.3">
21     document.write("여기에 작성되는 자바스크립트 코드는 1.3<br>");
22 </SCRIPT>
23
24 </BODY>
25 </HTML>
```

JavaScript 작성시 Script태그의 문제점

■ Script태그의 문제점

- 넷스케이프 이외의 브라우저 중에서 **SCRIPT** 태그를 지원하지 않을 수 있다
- 자바스크립트를 지원하지 않는 브라우저는 이 태그에 의해 삽입된 내용을 그대로 화면에 표시하게 된다.
- 주석(<!-- 와 -->)을 사용하여 다른 브라우저에서는 스크립트 내용을 주석으로 해석하도록 한다
- *<SCRIPT LANGUAGE="JavaScript">
<!-- HIDE THE SCRIPT FROM OTHER BROWSERS
.... JavaScript Program ...
// STOP HIDING FROM OTHER BROWSERS -->
</SCRIPT>*

실습 2 : JavaScript 작성시 Script태그의 문제점

■ 실습 2

- 자바스크립트 코드를 실행하지 못하는 웹 브라우저를 위해서 <SCRIPT> 내에 HTML 주석을 삽입하자.

```
20 <SCRIPT LANGUAGE="JavaScript1.2">
21 <!--
22     document.write("여기에 작성되는 자바스크립트 코드는 1.2<br>");
23 //-->
24 </SCRIPT>
25
26 <SCRIPT LANGUAGE="JavaScript1.3">
27 <!--
28     document.write("여기에 작성되는 자바스크립트 코드는 1.3<br>");
29 //-->
```


자바스크립트 실행 문장의 구조

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

```
<!--
```

```
// 한줄 주석
```

```
/* 여러줄 주석 */
```

```
자바스크립트 코드;
```

```
//-->
```

```
<SCRIPT>
```

실습 3 : 자바스크립트 기본 구조 작성하기

■ 실습 3

- 자바스크립트 기본 구조를 작성하자.
 - 기본 구조 패턴을 몸에 익혀 매번 프로그램 작성할 때마다, 주석 처리 및 들여쓰기 등등의 습관을 들여놓자.

```
1 <HTML>
2 <HEAD>
3 <TITLE>실습3. 자바스크립트의 기본구조 표현</TITLE>
4 </HEAD>
5
6 <BODY BGCOLOR="#FFFFFF">
7
8 <SCRIPT LANGUAGE="JavaScript">
9 <!--
10
11     //자바스크립트 주석 부분
12
13     /*
14     자바스크립트 여러줄 주석 부분
15     */
16
17     //자바스크립트 코드 처리 부분 : 세미콜론(;)으로 끝맺는다.(옵션)
18     document.write("자바스크립트의 기본구조 표현<br>");
19
20 <!-->
21 </SCRIPT>
22
23 </BODY>
24 </HTML>
```

자바스크립트 시작 전에 이해 해야 할 4가지 용어

■ 객체(object) : 데이터를 지닐 수 있으며, 어떤 동작을 수행할 수 있는 하나의 단위

- 예) 사람, 컴퓨터, 자동차

■ 속성(property) : 객체가 지니는 성질, 특성, 색상 등의 표현

- 예) 자동차의 색상이 검정, 바퀴가 4개

■ 메서드(method) : 객체의 동작 수행 능력

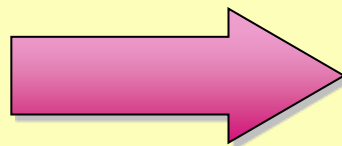
- 예) 자동차의 기어변속, 좌회전, 우회전, 후진

■ 이벤트(event) : 객체의 동작 수행의 결과/신호

- 예) 자동차가 전진하다 전신주를 받았을 때 어떤 결과 : 사고발생=이벤트

■ 객체.속성 = 속성값

■ 객체.메서드(내용)



이벤트(이벤트핸들러)

실습 4 : 객체, 속성, 메서드, 이벤트를 사용하자.

■ 실습 4

- 자바스크립트 소스 코드는 자바스크립트의 객체, 속성, 메서드, 이벤트로 이루어져 있다. 이를 이용한 테스트 페이지를 만들어 보자.

복습

- 자바스크립트의 버전 표시
- 자바스크립트 작성시 **Script**태그의 문제점
- 자바스크립트 문장의 구조
 - 주석
 - 기본 구조
- 자바스크립트 시작 전에 이해 해야 할 4가지 용어
 - 객체
 - 속성
 - 메서드
 - 이벤트

기본 출력문

■ 문서에 출력

- document.write() 메서드

■ 메시지박스(경고 대화 상자) 출력

- window.alert() 메서드

◆ 자바스크립트 기초 : 변수와 연산자

- 자료형(데이터 유형;Data Type)

- 변수(Variable)

- 연산자(Operator)

- 산술연산자
- 연결연산자
- 관계연산자
- 논리연산자
- 조건연산자
- 비트연산자
- 증감연산자
- 대입연산자

- 연산자 우선순위

자바스크립트 기초 – 자료형(데이터 유형)

■ 자료형(데이터 유형)

- 정수형
 - 10진수, 16진수, 8진수의 3가지 형태(양수나 음수)
- 부동소수점(실수형)
 - 12.43, -4.555 와 같이 소수 부분이 있는 10진수.
 - 지수를 사용하여 자리수가 큰 수도 표현. 예를 들어 2×10^4 같은 숫자는 2E4 로 표현.
 - 가끔 부동소수점의 계산이 정확하지 못한 경우가 있으므로 가급적이면 정수형을 사용
- Boolean
 - 크기가 1비트인 자료형으로 **true**와 **false**의 두 가지 값만을 가지며 주로 함수의 리턴 값이나 관계 연산자의 결과값으로 사용
- null
 - 아무것도 없다는 것을 의미, 예외처리에 자주 사용
 - 프롬프트 대화상자에서 사용자가 “Cancel” 버튼을 선택할 경우
- 문자열
 - 따옴표(“”, ‘ ’)에 들어가는 모든 문자

자바스크립트 기초 - 변수

■ 변수

- 변수의 정의
 - 프로그램 내에서 특정 자료형의 값을 가지고 있는 저장 장소를 의미하는 것.
- 변수의 규칙
 - 변수는 반드시 알파벳 문자나 `_` 로 시작.
(예 : `var aaa`, `ABc112`, `_finder`, 단 자바스크립트는 대문자와 소문자를 구별한다)
 - 예약어는 변수로서 사용할 수 없음.(다음 슬라이드에 예약어 표시)
- 변수의 선언
 - 변수에 값 할당시 : `str="JavaScript"`, `count=1` 등
 - `var` 명령어를 사용하여 선언
 - `Var`를 이용한 방법 : `var count = 1`
 - `Var`를 이용하면 같은 이름의 변수라도 현재 자신이 속해있는 영역 안에서만 효력이 발생됨.
 - 변수 선언시 데이터형을 같이 표기해주지 않는다.
 - 대/소문자 구별

자바스크립트 기초 – 자바스크립트 예약어

- 자바스크립트 예약어의 종류

예약어 종류

abstract	boolean	break	byte	case	catch
char	class	const	continue	default	do
double	else	extend	false	final	finally
float	for	function	goto	if	
implements		import	in	instanceof	intinterface
long	native	new	null	package	private
protected		public	return	short	static super
switch	synchronized	this	throw	throws	transient
true	try	var	void	while	with

자바스크립트 기초 - 데이터와 정보 처리

■ 데이터형 변환(casting)

- 데이터 유형에 대한 제약이 비교적 약한 언어
- 변수의 유형을 변수를 생성할 때 결정하지 않아도 되며, 변수의 내용에 따라 변수의 유형이 변경될 수 있다
- 선언되어 있는 형태와 다른 데이터형을 할당해도 대부분 치명적인 문제가 생기지 않는다

예) $3.5 + "10" = 13.5$, $"Count\ to" + 10 = Count\ to\ 10$

실습 1 : 변수 선언, 대입, 출력(사용) 1

■ 실습 1

- 변수를 선언 후, 값을 대입하고, 그 변수의 값을 출력하는 하나의 패턴을 익히자.

실습 2 : 변수 선언, 대입, 출력(사용) 2

■ 실습 2

- 변수를 선언함과 동시에 값을 대입해 보고, 문자열의 숫자형과 정수형의 숫자형을 더했을 때 어떤 결과가 나오는지 알아보고, 그 해결책을 찾아보자.

자바스크립트 기초 - 연산자

■ 연산자

● 산술연산자

더하기	$a + b$	빼기	$a - b$
곱하기	$a * b$	나누기	a / b
나머지	$a \% b$		

● 연결연산자

- '+' 기호를 사용해 문자열과 숫자를 연결

실습 3 : 산술연산자(+, -, *, /, %)

■ 실습 3

- 산술연산자의 5가지 연산자가 사용된 자바스크립트 코드를 작성해 보자.

자바스크립트 기초 - 연산자

- 관계(비교)연산자

- 두 피연산자를 조건에 따라 선택적으로 실행할 수 있도록
- 참이면 **true**, 거짓이면 **false** 값을 반환

==	a와 b가 같다	a == b	!=	a와b가 같지않다	a!=b
>	a보다b가 작다	a > b	<	a보다b가 크다	a < b
>=	a보다b가 작거나 같다	a >= b	<=	a보다b가 크거나 같다	a <= b

실습 4 : 관계연산자 : ==, !=, >=, >, <, <=

■ 실습 4

- 관계연산자를 사용해서 두 변수의 값을 비교해 보자.

자바스크립트 기초 – 논리연산자 : &&(AND), ||(OR), !(NOT)

- 논리연산자
 - true와 false 값을 가짐.
 - 연산자 논리표

좌변	우변	연산결과		
			&&	^
false(0)	false(0)	0	0	0
false(0)	true(1)	1	0	1
true(1)	false(0)	1	0	1
true(1)	true(1)	1	1	0

계산	boolean 연산
Or-ret = (3>6) (4 < 8)	true = false true
and_ret = (3>6) && (4 < 8)	false = false && true
Xor_ret = (3>6) ^ (4 < 8)	1 = false ^ true

실습 5 : 논리연산자

■ 실습 5

- 두개의 조건이 모두 만족할 경우에만 관리자 인증을 하는 로직을 구성해보자

자바스크립트 기초 – 조건연산자/비트식 연산자

- 조건연산자(3항연산자)

- 조건에 따라 두 값 중에 하나를 연산결과로 전달
- 형식 : (조건) ? 변수1 : 변수2
 - 조건인 참이면 앞 문장 수행, 거짓이면 뒤 문장 수행

- 비트식 연산자

- 비트는 0과1값을 갖는 최소의 단위로서 좌변과 우변의 비트값에 따라 or, and, xor이 결합되어 다음과 같은 논리값을 가짐.

좌변	우변	연산결과		
			&	^
false(0)	false(0)	0	0	0
false(0)	true(1)	1	0	1
true(1)	false(0)	1	0	1
true(1)	true(1)	1	1	0

실습 6 : 조건연산자 : ?:연산자

■ 실습 6

- 조건연산자(3항연산자)를 사용해서 관리자 로그인하는 로직을 구성해보자.

자바스크립트 기초 - 증감연산자

- !(not) 연산자는 값이 0인 경우는 1로 1인 경우는 0으로 바꿔줌.
- ~연산자는 1의 보수로 만들어줌.
- 논리연산자는 ||, &&.... 비트식 연산자는 한 개씩임.
- >>, <<, >>> 연산자는 비트를 이동하는 연산자

이동연산자	설명
<<	왼쪽으로 비트 이동
>>	오른쪽으로 비트 이동, 양수와 음수에 상관없이 왼쪽 끝 한 비트는 항상 0으로 채워짐.
>>>	음수인 경우에는 왼쪽 끝 한 비트가 1로 채워지고 양수인 경우에는 왼쪽 끝 한 비트가 0으로 채워짐.

- 증감연산자
 - 편리하게 1씩 증가하거나 1씩 감소하는데 사용
 - ++, --로 표시 (++A와, A++의 차이점?)

실습 7, 실습 8: 증감연산자

■ 실습 7, 실습 8

- 증감연산자를 사용해서 변수의 값을 증가 및 감소시켜보자.

자바스크립트 기초 – 대입연산자/연산자 우선순위

- 대입연산자(=, +=, -=, *=, /=, %=)

- 우변의 값을 좌변의 변수에 할당하는 연산자
- += : 앞에 변수에다 뒤의 변수값을 누적

```
a = 5; b = 10;
```

```
a += b;
```

```
//a = a + b; //+=연산자와 같다.
```

```
document.write(a); // a의값은? 15
```

- 연산자 우선순위

- 우선순위를 모르면 잘못된 식이나 결과를 만들어 버그의 원인이 됨.
- 우선 순위를 잘 모를 경우에는 괄호를 사용
- 자바스크립트의 연산자는 C언어의 연산자와 비슷함.

실습 9 : 대입연산자

■ 실습 9

- 대입연산자는 값을 누적시킬 때 사용하면 편리하다.

자바스크립트 기초 – 연산자 우선순위

종류	연산자	우선순위
괄호/대괄호	., [], ()	1
부정/증감	!, ++, --	2
산술	*, /, %, +, -	3
비트식	<<, >>, >>>	4
관계	<, >, <=, >=	5
	==, !=	6
비트식	&	7
	^	8
		9
논리	&&	10
		11
	? :	12
대입/할당	=, +=, -=, *=, >>=, ^=, ...	13

복습

■ 자료형(데이터 유형)

■ 변수

■ 연산자

- 산술연산자
- 연결연산자
- 관계연산자
- 논리연산자
- 조건연산자
- 비트연산자
- 증감연산자
- 대입연산자

■ 연산자 우선순위

◆ 자바스크립트 처리를 위한 대화상자

- 경고 대화상자(alert)
- 확인 대화상자(confirm)
- 입력 대화상자(prompt) : 거의 사용하지 않음

window객체의 3가지 주요 메서드 – 경고 대화상자

■ 경고 메시지박스(alert)

- 메시지를 사용자에게 알려서 경고나 문제가 되는 부분 또는 확인이 필요한 대화상자를 보여줌
- window객체의 메서드
- 주로 이벤트핸들러인 **onClick**과 함께 사용
- 메시지박스가 나타날 때 경고음 때문에 잦은 사용은 피한다.

■ 사용법

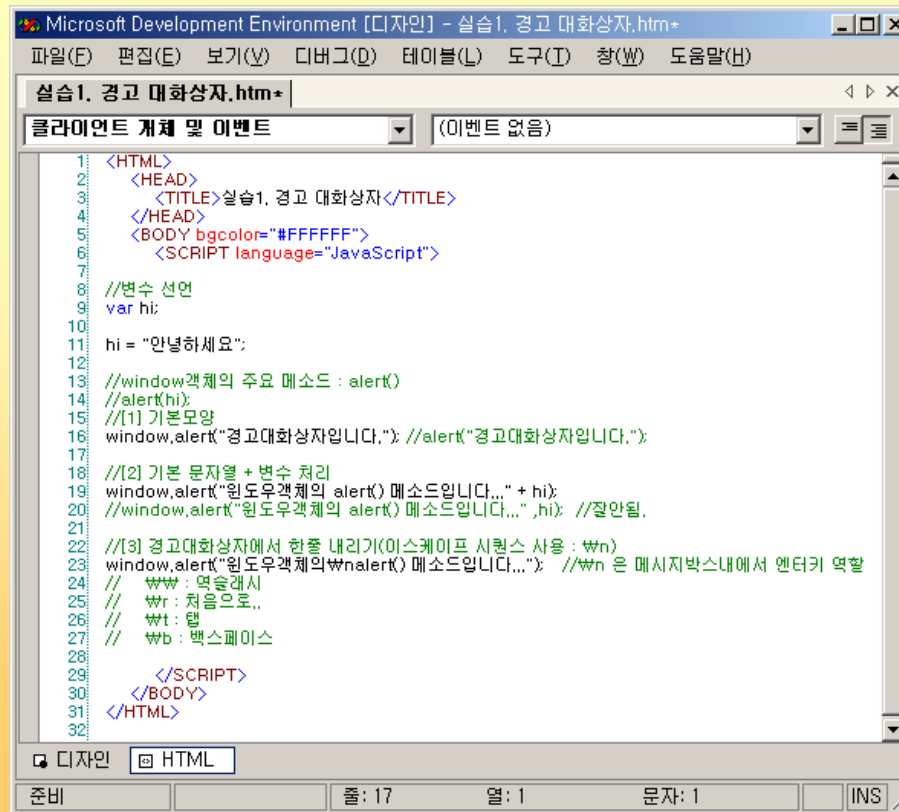
- **window.alert(문자열)**
- **alert(문자열)**
 - 일반적으로 자바스크립트 최상위 객체인 **window**는 생략해서 사용함.



실습 1 : 경고 대화상자

■ 실습 1

- 경고 대화상자를 사용해서 화면에 메시지를 출력하는 프로그램을 작성해보자.

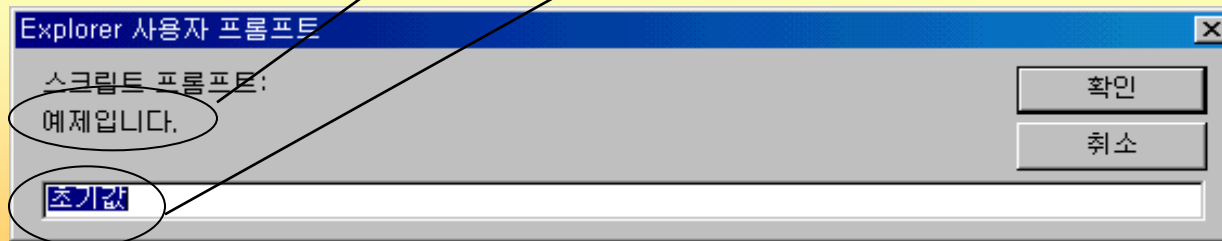


```
1 <HTML>
2 <HEAD>
3 <TITLE>실습1, 경고 대화상자</TITLE>
4 </HEAD>
5 <BODY bgcolor="#FFFFFF">
6 <SCRIPT language="JavaScript">
7
8 //변수 선언
9 var hi;
10
11 hi = "안녕하세요";
12
13 //window객체의 주요 메소드 : alert()
14 //alert(hi);
15 // [1] 기본모양
16 window.alert("경고대화상자입니다."); //alert("경고대화상자입니다.");
17
18 // [2] 기본 문자열 + 변수 처리
19 window.alert("윈도우객체의 alert() 메소드입니다..." + hi);
20 //window.alert("윈도우객체의 alert() 메소드입니다..." ,hi); //잘안됨.
21
22 // [3] 경고대화상자에서 한글 내리기(이스케이프 시퀀스 사용 : \n)
23 window.alert("윈도우객체의\ernalert() 메소드입니다..."); //\n 은 메시지박스내에서 엔터키 역할
24 // \t : 역슬래시
25 // \r : 처음으로..
26 // \t : 탭
27 // \b : 백스페이스
28
29 </SCRIPT>
30 </BODY>
31 </HTML>
```

window객체의 3가지 주요 메서드 – 입력 대화상자

■ 입력 메시지박스(prompt)

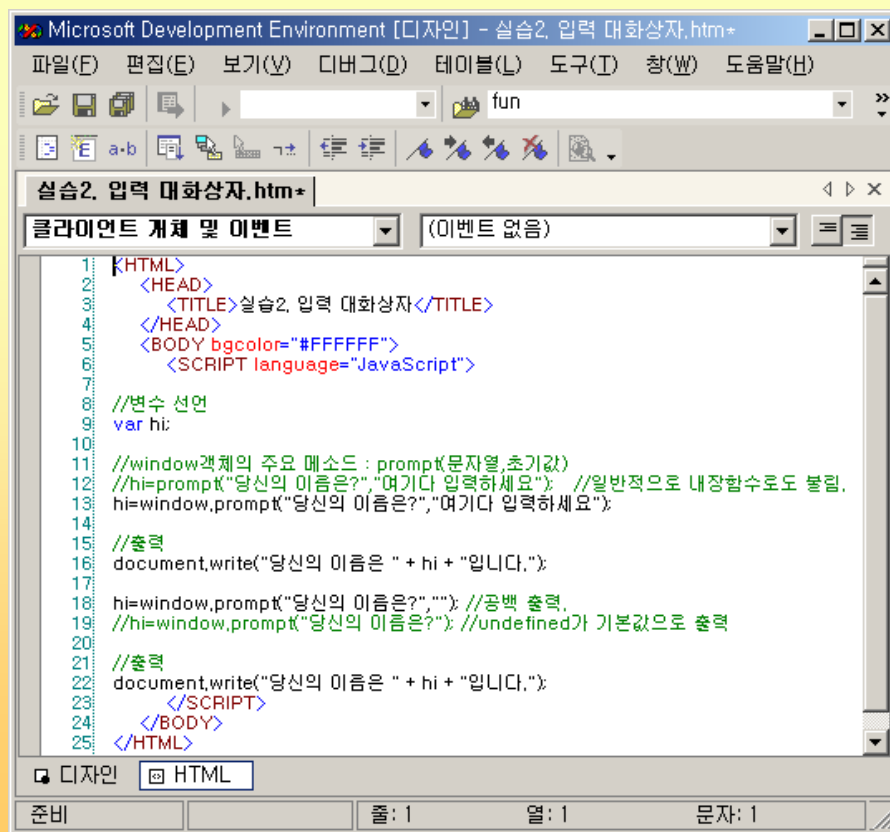
- window객체의 메서드
- 프롬프트라는 이름으로 사용
- 형식
 - 반환된 값 = **prompt** (문자열, 초기값)
 - 문자열 : 프롬프트 안에 들어갈 설명 문자열
 - 초기값 : 기본적으로 텍스트 박스에 들어갈 문자열이나 숫자를 설정
 - 사용예 : `exam = prompt("예제입니다.", "초기값")`



실습 2 : 입력 대화상자

■ 실습 2

- 입력 대화상자를 사용해서 화면으로부터 메시지를 입력하는 프로그램을 작성해보자.

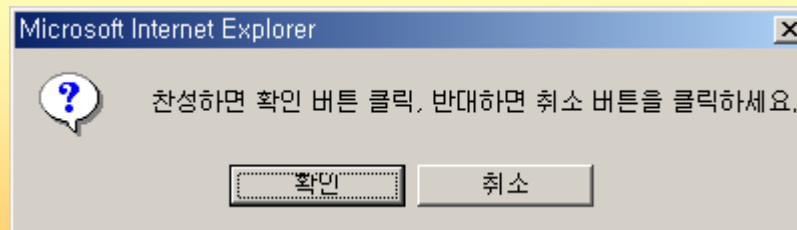


```
1 <HTML>
2 <HEAD>
3 <TITLE>실습2. 입력 대화상자</TITLE>
4 </HEAD>
5 <BODY bgcolor="#FFFFFF">
6 <SCRIPT language="JavaScript">
7
8 //변수 선언
9 var hi;
10
11 //window객체의 주요 메소드 : prompt(문자열,초기값)
12 //hi=prompt("당신의 이름은?", "여기다 입력하세요"); //일반적으로 내장함수로도 불림.
13 hi=window.prompt("당신의 이름은?", "여기다 입력하세요");
14
15 //출력
16 document.write("당신의 이름은 " + hi + "입니다.");
17
18 hi=window.prompt("당신의 이름은?", ""); //공백 출력,
19 //hi=window.prompt("당신의 이름은?", ""); //undefined가 기본값으로 출력
20
21 //출력
22 document.write("당신의 이름은 " + hi + "입니다.");
23 </SCRIPT>
24 </BODY>
25 </HTML>
```


window객체의 3가지 주요 메서드 – 확인 대화상자

■ 확인 메시지박스(confirm)

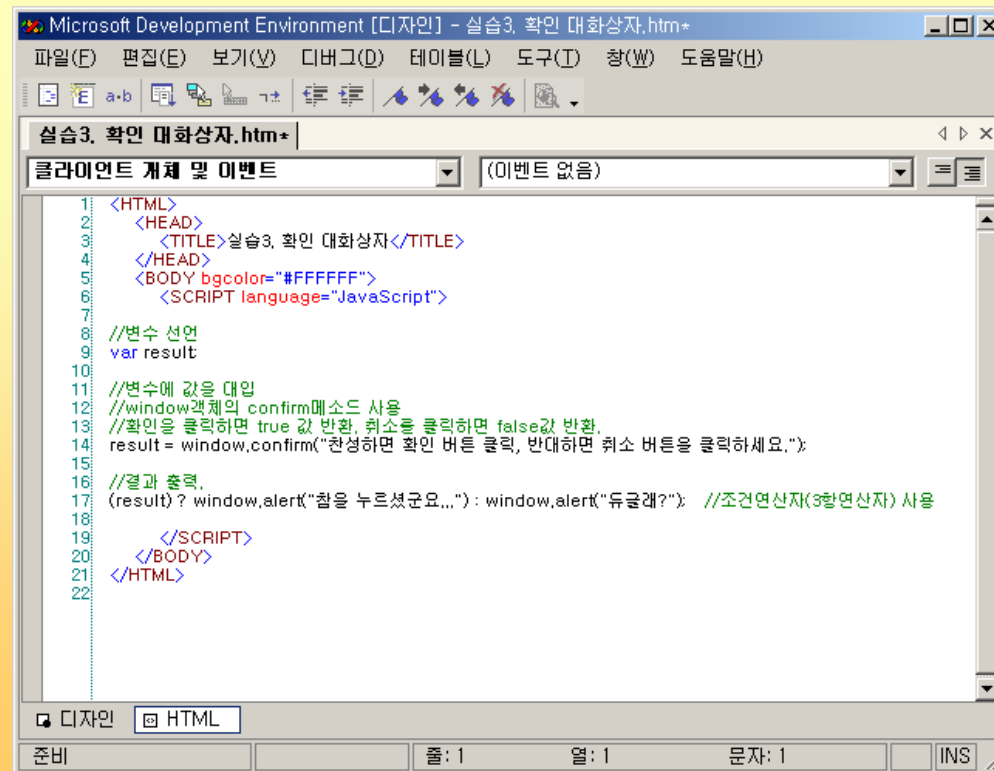
- 메시지를 사용자에게 알려서 [확인]이나 [취소]를 선택하여 적절하게 조치할 수 있게 해주는 대화상자
- window객체의 메서드
- 형식
 - **Confirm(문자열)**
 - 확인 버튼을 누르면 참(true)값을 취소 버튼을 누르면 거짓(false)값을 반환한다.



실습 3 : 확인 대화상자

■ 실습 3

- 확인 대화상자를 사용해서 확인버튼을 눌렀는지, 취소버튼을 눌렀는지를 알아보는 프로그램을 작성해 보자.



```
1 <HTML>
2 <HEAD>
3   <TITLE>실습3. 확인 대화상자</TITLE>
4 </HEAD>
5 <BODY bgcolor="#FFFFFF">
6   <SCRIPT language="JavaScript">
7
8   //변수 선언
9   var result
10
11  //변수에 값을 대입
12  //window객체의 confirm메소드 사용
13  //확인을 클릭하면 true 값 반환, 취소를 클릭하면 false값 반환.
14  result = window.confirm("천성하면 확인 버튼 클릭, 반대하면 취소 버튼을 클릭하세요.");
15
16  //결과 출력.
17  (result) ? window.alert("참을 누르셨군요...") : window.alert("뉘글래?"); //조건연산자(3항연산자) 사용
18
19  </SCRIPT>
20 </BODY>
21 </HTML>
22
```

복습

- 경고 대화상자(alert)
- 입력 대화상자(prompt)
- 확인 대화상자(confirm)

◆ 제어문 : 조건문/반복문/분기문

■ 조건문

- if~else문
- switch~case문

■ 반복문

- for문
- while문
- do문

■ 기타(점프문)

- continue문
- break문
- goto문

Statement Blocks

- Use braces As block delimiters

```
{  
    // code  
}
```

- A block and its parent block cannot have a variable with the same name

```
{  
    int i;  
    ...  
    {  
        int i;  
        ...  
    }  
}
```

- Sibling blocks can have variables with the same name

```
{  
    int i;  
    ...  
}  
...  
{  
    int i;  
    ...  
}
```

Types of Statements

Selection Statements

The if and switch statements

Iteration Statements

The while, do, for, and foreach statements

Jump Statements

The goto, break, and continue statements

조건문

■ 조건문

- 조건을 판단한 후 조건에 따라 분기하여 수행
- if 문
 - 조건을 만족하면 문장을 실행하고 다음 라인을 수행
 - if 문의 조건을 만족하지 않으면 수행을 한 번도 하지 않음.
 - 형식

```
if (조건식)
{
    문장;
}
```

실습 : if문_성적확인프로그램

■ 실습

- 문제 : 입력상자에서 입력된 값이 60보다 작으면 “F입니다.”를 출력하는 자바스크립트를 작성

조건문

- if ~ else 문

- 형식

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

- 조건을 만족하면 조건 다음의 문장1을 수행하고 조건을 만족하지 않으면 **else** 다음의 문장 2를 수행

실습 : if_else문_성적확인프로그램

■ 실습

- 문제 : 입력상자에서 입력된 값이 60보다 작으면 “F입니다.”, 100보다 크면 “잘못 입력하였습니다.”를 출력하는 자바스크립트 작성

조건문

- 다중 if 문

- 형식

```
if (조건1)
{
    문장 1 ;
}
else if (조건2)
{
    문장 2 ;
}
    (else if를 붙여 연속적으로 만들 수 있음)
else
{
    기본문장;
}
```

실습 : 다중if문_ 성적확인프로그램

■ 실습

- 문제 : 성적이 90에서100 사이면 "A"와 "잘했습니다.", 성적이 80에서 90사이이면 "B"와 "보통입니다." 70보다 작으면 "F"와 "좀 더 노력하세요"를 출력하는 자바스크립트를 작성하시오.

조건문(선택문)

■ switch 문

- 특징

- 다중 if문을 대신하여 switch문 사용
- 변수에 따라 선택적으로 [상수]로 분기하므로 소스를 이해하기 편하고 작성도 용이

- 형식

```
switch (표현식)
{
case 값1 : 문장 1 ; break;
case 값2 : 문장 2 ; break;
case 값3 : 문장 3 ; break;
.....
default : 문장 n;
}
```

- Switch문 활용하기

- Case : 선택적으로 분기하여 수행하기 위해 사용
- Break : switch()문을 중지

실습 4 : 매달 14일은 무슨 날

■ 실습 4

- 1월에서 12월 중 매달 14일은 무슨 날인지를 알아내는 프로그램을 작성해보자.

반복문 : 구간반복

■ for 문

● 특징

- [시작값], [최종값], [증가값]을 설정하여 반복적인 수행을 할 수 있는 문장
- 규칙적인 증가를 하는 경우에 많이 사용

● 형식

```
for (초기값; 최종값; 증가식)
{
    문장 ;
}
```

- 한 문장 이상인 경우에는 '{'와 '}' 기호를 이용하여 문장을 묶어줌.

실습 5 : 1~100까지 출력

■ 실습 5

- 1부터 100까지의 수를 화면에 출력시켜보자.

실습 6. 1~100까지의 합

■ 실습 6

- 1부터 100까지의 합을 구하는 프로그램을 작성하라.(+=연산자 사용)

실습 7 : 입력된 수의 제곱

■ 실습 7

- 입력 대화상자를 통하여 입력된 수의 제곱을 구하는 프로그램.

실습 8 : 1~100까지의 짝수의 합

■ 실습 8

- 1부터 100까지의 짝수의 합을 구하시오.(2550)

반복문

- 이중 for 문

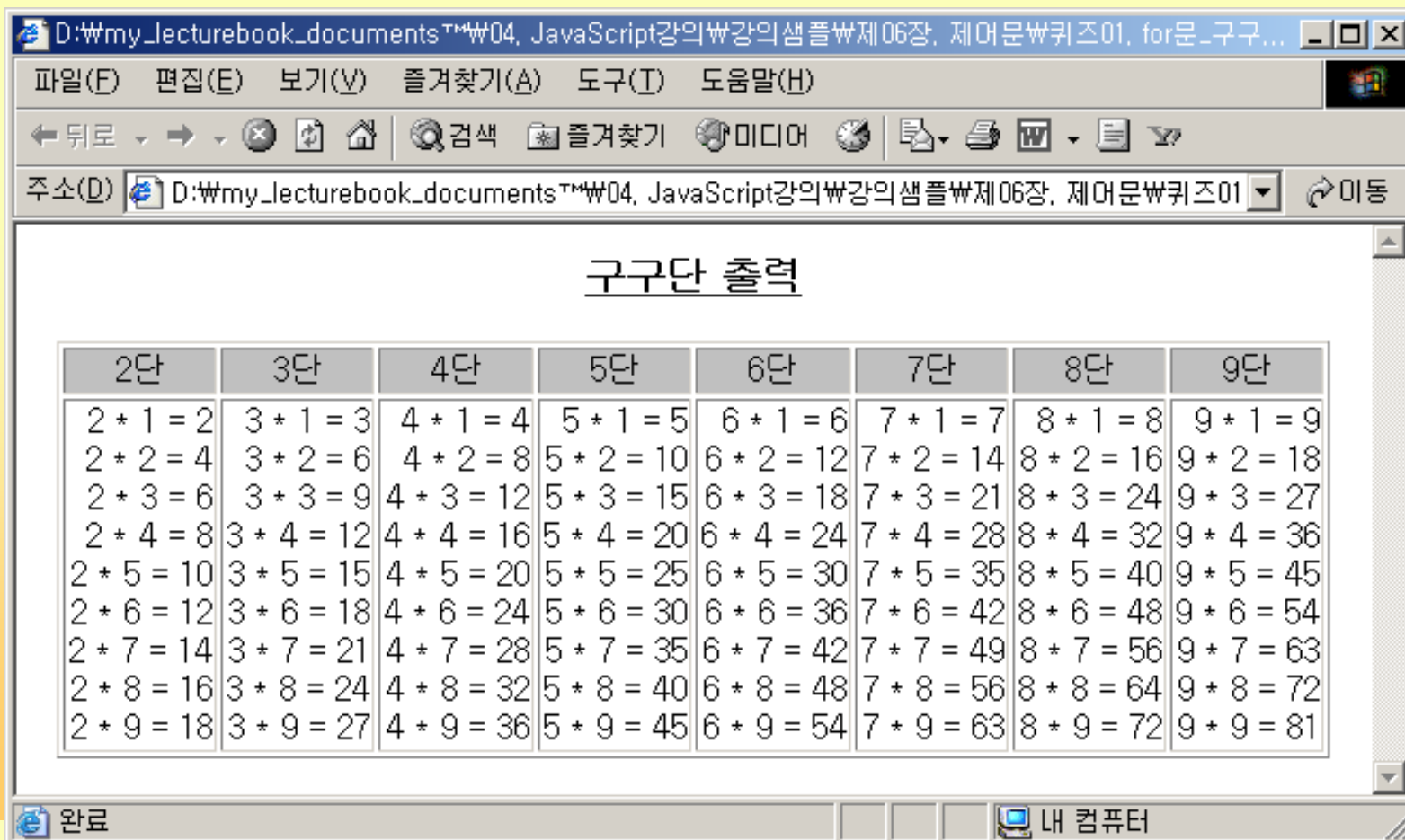
- 안쪽의 for문과 바깥쪽의 for 문이 쌍을 이루어 반복 수행
- 형식

```
for (초기값; 최종값; 증가식)
{
    문장 ;
    for (초기값; 최종값; 증가식)
    {
        문장 ;
    }
}
```

실습 9 : 구구단 출력 프로그램

■ 실습 9

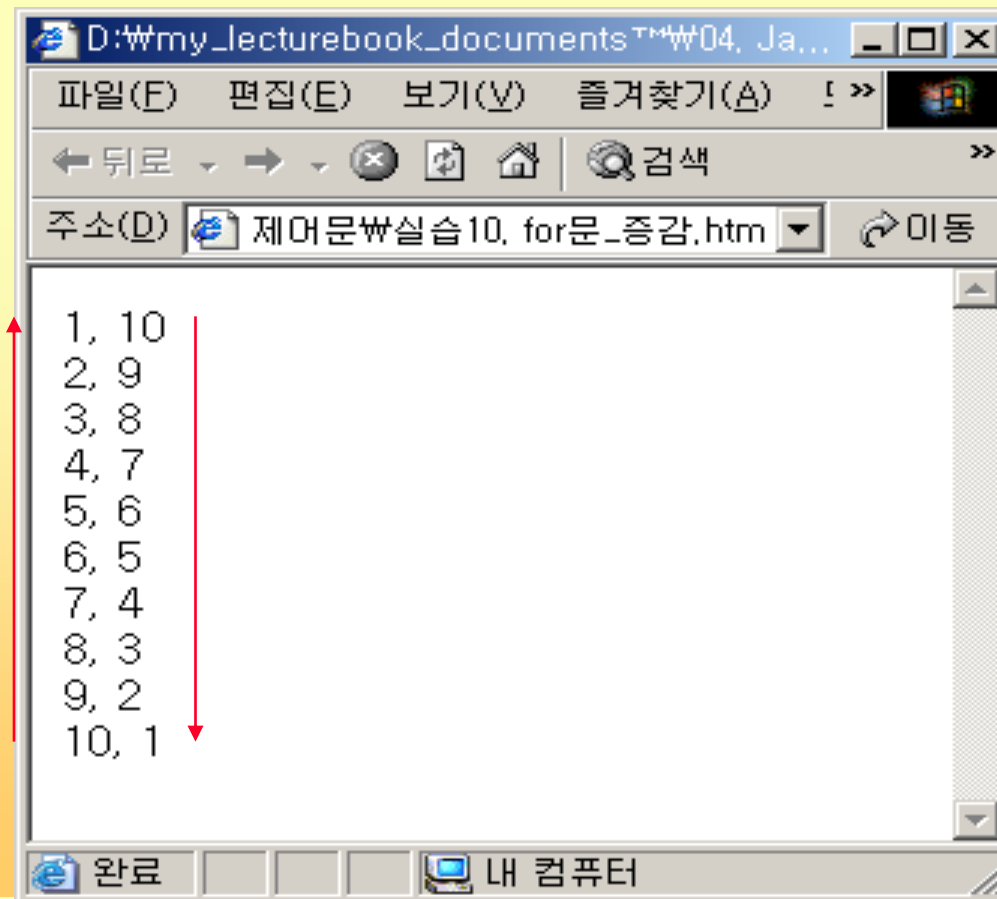
- 다중 for문을 이용해서 구구단을 출력하는 프로그램을 작성해 보자.



실습 10 : 증감

■ 실습 10

- 아래와 같이 1부터 10까지 증가되고, 10부터 1까지 감소되는 프로그램을 작성해 보자.



반복문 : 조건반복

■ while 문

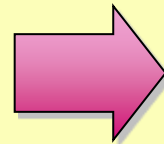
● 특징

- [조건만족]이 true인 동안은 문장을 반복하고 false일 경우에는 while 문 다음의 문장을 수행
- 조건을 먼저 확인하기 때문에 한 번도 수행하지 않는 경우도 있음.

● 형식

```
while (조건식)
{
    명령문;
}
```

```
초기값;
while (조건식)
{
    명령문;
    증감식;
}
```



```
for (초기값; 최종값; 증감식)
{
    명령문;
}
```

실습 11 : 1~100까지의 짝수의 합을 구하는 프로그램

■ 실습 11

- 문제 : **while**문을 이용하여 1부터 100까지의 짝수의 합을 구하는 프로그램을 작성

실습 12 : 무한루프

■ 실습12

- 무한 루프

반복문 : 실행반복

■ do~while 문

- 특징

- [반복문장]을 수행한 후에 [조건만족]을 확인
- [조건 만족]이 맞던, 틀리던 [반복 문장]을 한 번 수행

- 형식

```
do
{
    문장 ;
} while(조건) ;
```

실습 13 : 1~100까지의 합을 구하는 프로그램

■ 실습 13

- 문제 : do while문을 이용하여 1부터 100까지의 합을 구하는 프로그램을 작성

기타

■ continue 문

● 특징

- for문, while문, do while문의 반복중에 continue 문을 만나면 조건을 판단한 후 다음 구문을 수행
- 다시 조건식으로 이동하는 명령문

실습 14 : 1에서 100까지의 정수 중 3의 배수를 제외한 수들의 합을 구하는 프로그램 (continue문 사용)

■ 실습 14

- 1에서 100까지의 정수 중 3의 배수를 제외한 수들의 합을 구하는 프로그램 (continue문 사용)

기타

■ break문

● 특징

- for, while, do while 문의 반복중인 반복구문을 강제로 종료하는 경우에 사용
- 반복문을 빠져나오는 명령문

반복문

■ for ... in 반복문

- 특정 객체의 모든 프로퍼티에 대해서 자동적으로 조치를 취하고자 할 경우

```
for (j in testObject) {  
    실행부분  
}
```

- for 루프는 testObject 객체에 있는 프로퍼티의 배열 인덱스 번호가 마지막 프로퍼티에 이르기까지 j를 0부터 하나씩 증가시키면서 명령어 블록을 수행

복습

- if~else문
- switch~case문
- for문
- while문
- do문
- continue문
- break문

◆ 함수

■ 함수 개요

- 함수란?

■ 함수의 종류

- 사용자 정의 함수
 - 사용자의 필요에 의해서 만들어서 사용하는 함수
 - 함수를 잘 구성해야 좋은 프로그램을 만들 수 있다.
- 내장 함수
 - 자바스크립트 자체에서 지원하는 함수
 - 다른 프로그램 언어에 비해서 내장함수가 상당히 적음

함수개요

■ 함수란?

- 함수란 프로그램내에서 특정 작업을 수행하기 위해 독립적으로 만들어진 하나의 단위를 말한다.

■ 함수의 목적

- 특정 작업을 하나의 단위로 만들어 사용함으로써 반복적으로 사용할 수 있도록 하기 위해서이다.

사용자 정의 함수 만들기

```
function 함수명 ([매개변수리스트])  
{  
    문장;  
    [return 반환값;]  
}
```

사용자 정의 함수 : 함수 만들기

■ 특징

- 함수는 브라우저가 웹 페이지를 로드할 때 메모리로 로드되며, 자바스크립트의 다른 부분이나 이벤트 핸들러(event handler)에 의해 호출된다
- 반드시 함수 호출 이전에 정의해야 함
- <HEAD> 태그 안쪽에 함수를 정의하는 것을 권장
 - 스크립트에서 선언된 모든 함수를 사용자 이벤트가 발생하기 이전에 분석(parse)한다는 장점을 가짐
- 함수의 처리 결과는 return 문장 사용

■ 방법

- 함수를 정의하는 방법에는 두 가지가 있음. 하나는 매개변수를 전달받아서 단순히 수행만 하는 [반환값이 없는 함수]와 매개변수를 전달받고 함수를 수행한 후 값을 반환하는 [반환값이 있는 함수]
 - 실제 자바스크립트에서는 function으로만 처리하기 때문에 구분이 되어 있지않음.
- 위치 : <head>와 </head>사이에 넣어 함수가 정의되기 전에 함수가 사용되는 것을 가급적 막으려 함.
- 장점 : 반복되는 문장을 최소화할 수 있고 소스도 간결하게 되어 빠르게 코딩할 수 있음.

함수 만들기

■ 반환값이 없는 함수

- 단순히 문장을 수행하는 경우에 이용
- 형식

```
function함수명 (매개변수1, 매개변수2...)  
{  
    문장;  
}
```

- [매개변수]를 생략한 예

```
function html_add() {  
    var a, b, c  
    a = 10  
    b = 30  
    c = a + b  
    document.write(c) }
```

함수 만들기

- [매개변수]를 지정한 예

```
function html_add(a, b) {  
    var c  
    c = a + b  
    document.write(c) }
```

- 예제 : 사칙연산

■ 반환값이 있는 함수

- function 안에서 문장을 수행한 후 최종 결과값을 반환하는 경우에 사용
- return 문을 이용하면 문장 끝에 반환값이나 반환할 변수를 설정할 수 있음.

실습 : 사용자 정의 함수 - 매개변수 및 반환값이 없는 경우

자바스크립트의 내장 함수

■ 내장 함수

- `eval(expression)` : 연산식 `expression`을 계산하고 그 결과를 반환한다.
- `isNaN(value)` : `value`가 숫자인지 아닌지를 조사한다. 결과값으로 불리언값을 반환한다. UNIX 운영체제에서만 지원
- `parseFloat(string)` : `string`을 부동소수점 수로 전환하고 그 값을 반환한다.
숫자가 아닌 문자 이전의 문자열까지 변환하며, 첫 문자가 숫자가 아닐 경우 NaN을 반환한다.
- `parseInt(string, base)` : `base`에 명시된 진법에 따라 `string`을 정수로 변환하고 그 값을 반환한다. 나머지 부분은 `parseFloat`와 동일

자바스크립트의 내장함수

- `escape(character)` : `character`의 부호화된 ASCII 문자열을 전달한다. 이때, 각 문자는 `%xx`로 바뀌게 된다.
- `unescape(string)` : ASCII 부호화에 의해 부호화된 `string`에서 문자열을 추출해서 반환한다. `string`의 문자열은 `%xx` 또는 16진수로 구성되어 있다.

자바스크립트의 내장 함수

■ 데이터 처리 관련 함수

● eval 함수

- 문자열이나 수식으로 표현된 문자열을 숫자로 처리하여 계산하는 함수
- 형식

```
eval (수식 문자열)
```

실습 1 : 간단계산기 만들기

■ 실습 1

- `eval()` 함수를 사용해서 간단한 수식을 계산하는 계산기를 만들어 보자.

자바스크립트의 내장 함수

- parseFloat와 parseInt 함수

- parseFloat는 문자열을 부동소수점형으로 parseInt는 문자열을 정수로 바꾸어 주고 문자열을 정상적으로 부동소수점형이나 정수로 바꾸지 못하면 NaN(Not a Number)을 반환
- 형식

```
parseFloat(문자열)     /   parseInt(문자열,진수)
```

실습 2 : parseInt() 및 parseFloat() 사용하기

■ 실습 2

- 문자형 정수와 문자형 실수를 숫자형으로 변환시켜보자.

자바스크립트의 내장 함수

- escape와 unescape 함수

- escape함수는 ISO-Latin-1 문자셋을 ASCII 값으로 반환하고 unescape 함수는 반대로 아스키 값을 ISO-Latin-1 문자셋으로 반환
- 형식

escape (ISO-Latin-1 문자셋)

unescape (문자열)

- isNaN 함수

- 매개변수가 NaN(Not a Number)인지를 검사
- True 또는 false 를 반환

배열 : 자바스크립트의 내장 객체

■ Array 객체

- 변수(Variable) : 하나의 이름으로 하나의 데이터 형식을 하나만 보관해 놓는 그릇 역할
- 배열(Array) : 하나의 이름으로 하나의 데이터 형식을 여러 개 보관해 놓는 그릇 역할
- 같은 성격의 데이터를 저장할 때 반복적인 변수의 사용을 피하기 위해 배열을 사용
 - `test = new Array(3)`
 - `test[0] = "test1"`
 - `test[1] = "test2"`
 - `test[2] = "test3"`
- 배열 선언과 동시에 값을 할당할 수 있음.
 - `test = new array("test1", "test2", "test3")`
- JavaScript 1.2의 새로운 방법
 - `test = ["array value1", "array value2", "array value3"]`

자바스크립트의 내장 객체

- Array 객체의 메서드

join([str])	배열에 들어 있는 값들을 모두 붙여서 하나의 문자열로 test.join() //결과 "test1,test2,test3" test.join(":") //결과 "test1:test2:test3"
sort(function)	배열값들을 매개변수(값을 비교하는 함수)를 이용하여 정렬 function compare(a,b) { if (a<b) return -1 else if(a>b) return 1 else return 0 } test.sort(compare)
reverse	배열의 순서를 정반대로 바꾼다.
concat(array)	두 개의 배열을 합해 하나의 배열로 test2={"test4","test5","test6"} test3=test1.concat(test2)
slice(start, end)	배열의 일부를 추출하여 새로운 배열로 test4=test3.slice(0,2)

배열

■ 배열의 정의

- 하나의 이름으로 여러 개의 변수를 사용할 수 있는 구조.
- 배열 객체를 만들어 사용하는 방법
- Netscape 3.0 이상 버전에서만 사용할 수 있는 내재된 객체를 이용하는 방법

■ 배열 객체를 만들어 사용하는 방법

- 배열의 속성을 가지는 객체 클래스를 생성한다

```
function makeArray(n)    {  
    this.length = n;  
    for (var l = 1; l < n; ++l) {  
        this[l] = null;  
    }  
    return this;  
}
```

배열

- 배열 객체를 생성하기 위해 선언을 한다.

```
var Oarray = new makeArray(3);
```

- 배열에 값을 넣기 위해 각 셀에 값을 할당한다.

```
Oarray[1] = "JavaScript"
```

```
Oarray[2] = "Example"
```

- 배열에서 값을 가져오는 방법

```
alert(Oarray[1]);
```

배열

■ 내재된 객체를 이용하는 방법

- *Array*라는 내재된 객체를 사용한다.

```
var Narray = new Array();
```

```
Narray[0] = "JavaScript";
```

```
Narray[1] = "Example";
```

- 배열의 인덱스는 0부터 시작한다

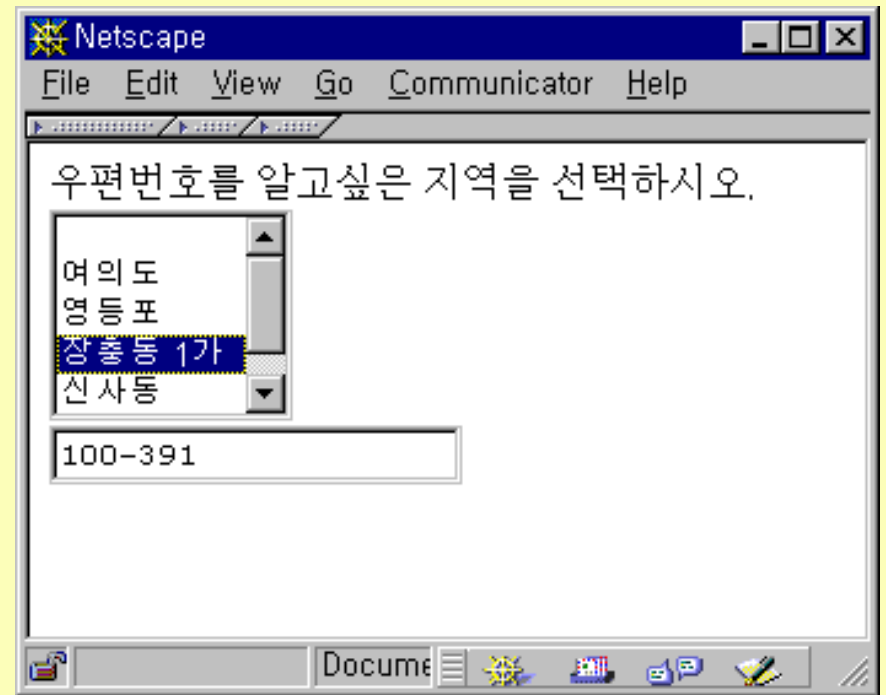
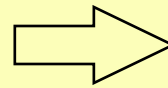
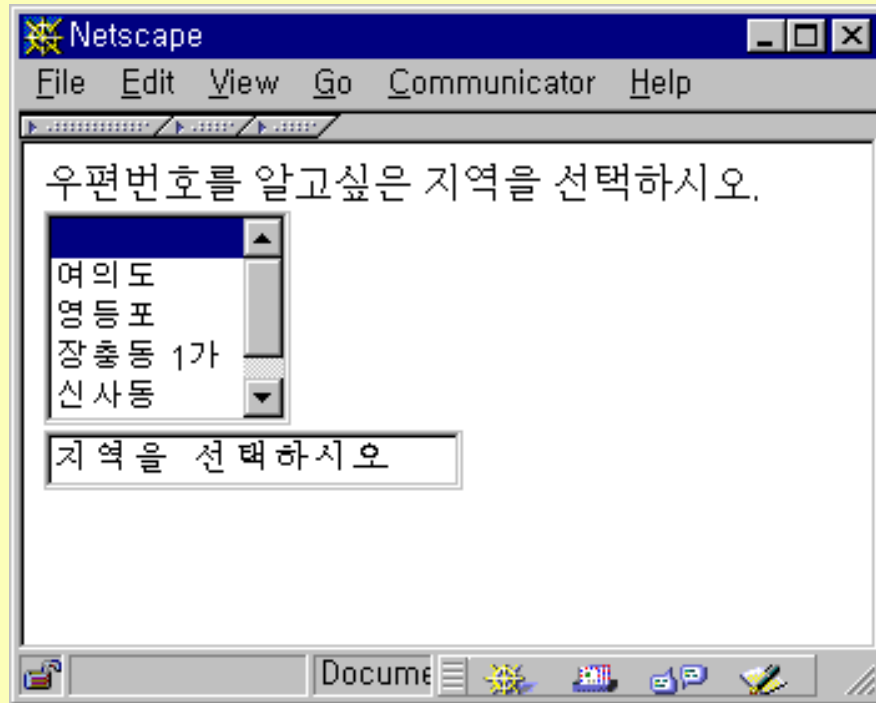
배열

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
function makeArray(n) {
    this.length = n;
    for (var i = 1; i <= n; ++i) {
        this[i] = null;
    }
    return this;
}
var cde = new makeArray(100);
var area = new makeArray(100);
area[1] = "여의도    ";   cde[1] = "150-010";
area[2] = "영등포    ";   cde[2] = "150-030";
area[3] = "장충동 1가";   cde[3] = "100-391";
area[4] = "신사동    ";   cde[4] = "135-120";
area[5] = "개포동    ";   cde[5] = "135-240";
//-->
</SCRIPT>
```

배열

```
<BODY>
<FORM>
    우편번호를 알고 싶은 지역을 선택하십시오.
    <SELECT name="address" size="5"
        onChange="form.returninfo.value = cde[form.address.selectedIndex];">
        <OPTION selected>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
        for(i = 1;i <= 5; ++i) {
            document.writeln("<option>" + area[i]);
        }
    //-->
    </SCRIPT>
    </SELECT><BR>
    <INPUT TYPE="text" NAME="returninfo" VALUE="지역을 선택하십시오">
</FORM>
</BODY>
</HTML>
```

배열



자바스크립트의 객체 모델

■ 자바스크립트 내장객체

- 자바스크립트에 내장된 객체

- 1) 현재 읽어 들인 **HTML** 문서의 구성 요소들과 정보를 주고받기 위한 객체

- 2) 산술 계산과 같은 유용한 작업을 수행하기 위한 객체

- 프로그래머가 생성한 객체

■ 웹브라우저(네비게이터) 내장객체 구조

- 네비게이터의 현재 상황과 현재 읽어 들인 웹 페이지와 그 내용에 대한 정보 제공

- 이러한 정보를 포함하는 프로퍼티(**Property**)를 이용해 작업을 수행하는 메서드 제공

- window, location, history, document, forms, anchors

우선적으로 살펴볼 5가지 주요 객체

- window 객체
- document 객체
- location 객체
- history 객체
- form 객체

자바스크립트의 객체 모델

■ 자바스크립트에서 객체 선언하기

- 함수 선언을 이용하여 객체를 선언

```
function student(name, age, grade)    {  
    this.name = name;  
    this.age = age;  
    this.grade = grade;  
}
```

- new 문장을 이용하여 객체 생성

```
student1 = new student("Bob", 10, 75);
```

- 자바스크립트에서는 객체를 다른 객체의 속성으로 사용할 수 있다

자바스크립트의 객체 모델

- 객체에 메서드 추가하기

```
function displayName() {  
    document.write("Name : " + this.name + "<BR>");  
}
```

```
function student(name, age, grade)    {  
    this.name = name;  
    this.age = age;  
    this.grade = grade;  
    this.displayName = displayName;  
}
```

자바스크립트의 객체 사용

■ 객체 정의

- 객체는 속성과 동작 방식에 해당하는 특성(property)과 메서드(method)를 가짐.
- 객체는 생성자 함수(constructor function)로 정의

```
function car(body, wheel, color){  
    This.body = body;  
    This.wheel = wheel;  
    This.color = color;  
}
```
- **this** : 생성자 함수를 통해서 만들어지는 객체 자기 자신을 가리킴.
- “.” 연산자는 객체에 속한 특성이나 메서드를 지정하는 역할을 하는 것.
- 왼쪽에는 객체, 오른쪽에는 메서드

자바스크립트의 객체 사용

■ 객체 만들기

- New라는 연산자를 사용하여 생성자 함수를 통해 객체 생성
 - mycar = new car("martiz", 12, "gold")
 - youcar = new car("Leganza", 16, "red")
- 위 객체들을 다시 표현해 보면
 - mycar.body = "martiz"
 - mycar.color="gold"
 - youcar.body="Leganza"
 - youcar.color="red"로 표현할 수 있다.

■ 객체 메서드 정의하기

- 메서드가 되는 함수를 먼저 정의
 - function print(){
 - document.write("BODY NAME:" + this.body + ",");
 - document.write("BODY WHEEL SIZE:" + this.wheel + ",");
 - document.write("BODY COLOR:" + this.color + "
");
 - }
- print()는 출력을 해주는 함수로 정의되었다.

자바스크립트의 객체 사용

- 생성자 함수에 이 함수를 메서드로 첨가

```
function car(body, wheel, color){  
    this.body = body;  
    this.wheel = wheel;  
    this.color = color;  
}
```

- 일일이 출력함수를 나열할 필요 없이 print메서드 호출사용

```
mycar.print();  
youcar.print()
```

두 문장의 출력결과는?

- 참조 배열로 객체의 특성 알기
 - mycar.body = mycar[0] = mycar["body"]="martiz"
 - mycar.wheel = mycar[1] = mycar["wheel"]=12
 - mycar.color = mycar[2] = mycar["color"]="gold"

자바스크립트의 객체 사용

■ 객체 제어문

● for ... in 문

- 객체 속에 들어 있는 모든 특성들을 반복해서 보여줌.

```
for (var each_property in mycar) {  
    document.write(mycar[each_property]);  
}
```

- 현재 객체 속에 몇 개의 특성이 들어 있는지 알 필요 없음.
- 객체의 특성들을 모두 읽을 때까지 **for...in** 제어문은 자동으로 반복 실행

자바스크립트의 내장 객체

■ Date 객체

- 날짜와 시간을 다루는데 사용
- `new` 연산자를 사용하여 만듦.
- `var now = new Date()`
Date 객체의 메서드

값 가져오기		값 설정하기	
<code>getFullYear()</code>	1970년도 이후의 연도	<code>setYear()</code>	1970년 이상 설정
<code>getMonth()</code>	월(0=1월, 1=2월, 2=3월.....)	<code>setMonth()</code>	월을 설정
<code>getDay()</code>	요일(0=일요일...)	<code>setDate()</code>	일을 설정
<code>getHours()</code>	시	<code>setDay()</code>	요일 설정
<code>getMinutes()</code>	분	<code>setTime()</code>	시간 설정
<code>getSeconds()</code>	초	<code>setHours</code>	시를 설정
<code>getTime()</code>	1970년1월1일 이후 시간을 1/1000초로 나타낸 값	<code>setMinutes</code>	분을 설정

자바스크립트의 내장 객체

■ Date객체를 이용한 예제

● 달력 만들기

- <table> 태그를 이용하여 달력 작성
- 현재 날짜에는 빨간색으로 표시
- 현재 시간이 그 날짜에 출력(오전/오후)

● 실행 결과



C:\w\kimsk\달력.HTML - Microsoft Internet Explorer

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

2000년 12월

일	월	화	수	목	금	토
						1 2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22 오후 4:10	23
24	25	26	27	28	29	30
31						

자바스크립트의 내장 객체

```
<html>
<head>
<script language="JavaScript">
//시간을 읽어서 오전 오후를 브라우저에 보여줌
function get_Time(){
    var now = new Date()
    var hour = now.getHours()
    var minute = now.getMinutes()
    var ampm
    now = null
    if (hour >= 12) {
        hour -=12
        ampm = "오후"
    } else
        ampm = "오전"
    hour = (hour == 0) ? 12:hour
    if (minute <10)
        minute = "0" + minute
    return ampa + hour + ":" + minute
}
```

```
//년과 달을 받아서 마지막 일을 알아 냄
function get_Day(year, month){
    var Last_Mon=new Array(31,29,31,30,
    31,31,31,31,30,31,30,31)
    var Mon2
    if (year %4 ==0)
        mon2 = true
    else
        mon2 = false
    Last_Mon[1] = (Mon2) ? 29:28
    return Last_Mon(month)
}
//table 태그를 이용하여 달력을 만듦.
function drawCal(firstDay, lastDate, date, year,
monthName){
    var text = ""
    text += "<CENTER>"
    text += "<TABLE>"
    text += "<TH COLSPAN=7 BGCOLOR= \"#F0F0F0\">"
    //년과 월을 출력
    text +=year+ "년" + (monthName+ 1)+ "월"
```

자바스크립트의 내장 객체

```
text+ = "</FONT>"
text+ = "</TH>"
var openCol = "<TD BGCOLOR='#FFFFFF' WIDTH=45
HEIGHT=40>"
openCol+ = "<FONT COLOR=darkblue>"
var closeCol = "</FONT></TD>"
text+ = "<TR ALIGH=center VALIGN= center>"
//달력의 일,월,... 요일 출력
var weekDay = new Array("일", "월", "화", "수",
"목", "금", "토")
for (var dayNum=0; dayNum<=6; dayNum++)
    text+ = openCol+ weekDay(dayNum) + closeCol
text+ = "</TR>"
var digit = 1
var curCell = 1
//달력 표를 만들
for(var row=1; row<=Math.cell(lastDate + fireDay-
1)/7); ++row) {
```

```
text+ = "<TR ALIGN=right VALIGH=top
BGCOLOR= '#FFFFFF'>"
    for (var col=1; col<=7; col++) {
        if(digit > lastDate) break
        if(curCell < firstDay) {
            text+ = "<TD> &nbsp; </TD>"
            curCell++
        }
        else
        {
            if (digit == date)
            {
                text+ = "<TD HEIGHT=40>"
                text+ = "<FONT COLOR=Red>"
                text+ = digit
                text+ = "</FONT><BR>"
                text+ = "<FONT COLOR= purple
SIZE=2>"
                text+ = "<CENTER>" + get_Time
+ "</CENTER>"
                text+ = "</FONT>"
                text+ = "</TD>"
            }
```

자바스크립트의 내장 객체

```
        else
            text+="  |
```

```
    //익스플로러가 아니면1900을 year에 붙여줌
    if(navigator.userAgent.indexOf("MSIE")==-1)
        year= 1900+ year
    var firstDayInstance=new Date(year, month, 1)
    var firstDay = firstDayInstance.getDay()+ 1
    firstDayInstance = null
    var days = get_Day(year,month)
    //달의 마지막 일을 구함.
    my_text=drawCal(firstDay,days,date,year, month)
    //최종적으로 만들어진 HTML 문서를 브라우저에 출력
    document.write(my_text)
    // -->
</SCRIPT>
</BODY>
</HTML>
```

자바스크립트의 내장 객체

■ string 객체

- new 연산자를 사용하지 않는다.

```
test = "test string"
```

- string 객체의 특성과 메서드는 string 객체 뒤에 "." 연산자와 함께 붙인다.

```
num = test.length
```

```
test.bold().blink()
```

- *string 객체의 메서드: 문자열 속성*

big()	글자를 좀더 크게	italics()	이탤릭체
small()	글자를 좀더 작게	strike()	글자가운데 줄긋기
blink()	깜박이게	sub()	아래첨자
bold()	볼드체	sup()	위 첨자
fixed()	타자기체	fontcolor("col")	글자색
fontsize("size")	글자크기	link("위치")	하이퍼텍스트통연결

자바스크립트의 내장 객체

“표식”.anchor(“특정”) //특정 위치 표시

“표식으로”.link(“#특정”) //특정 위치로 이동

● string 객체의 메서드: 문자열 처리

charAt(index)	매개변수로 입력된 숫자가 지정하는 곳에 있는 문자 리턴
indexOf(string)	문자열에서 자신이 검색하기를 원하는 문자가 나오는 곳의 위치를 리턴(왼쪽부터)
lastIndexOf(string)	지정된 위치의 문자 찾기(오른쪽 부터)
substring(index1, index2)	지정된 위치 사이에 있는 문자열을 추출
toLowerCase()	모든 문자를 소문자로
toUpperCase()	모든 문자를 대문자로
concat(string)	두 개의 문자열을 합하여 하나의 새로운 문자열
slice(시작 index, 끝index)	문자열의 일부를 추출하여 새로운 문자열
split([분리자],[개수])	분리자를 기준으로 여러 개의 문자열로

자바스크립트의 내장 객체

■ Number 객체

- 문자 로된 숫자 단어를 실제 숫자로 바꾸어주는 객체

```
document.write(number("3"));
```

이벤트

■ 이벤트란 ?

- 사용자로부터 특정 조작이 행해졌을 때 컴퓨터로부터 발생하는 일종의 신호

■ 자바스크립트에서 사용 가능한 이벤트 목록

- **blur** : 입력 포커스가 폼의 구성요소에 있다가 옮겨진 경우에 발생
- **click** : 링크나 폼의 구성원을 선택한 경우에 발생
- **change** : 입력 포커스가 폼 내의 필드에 주어진 경우 발생
- **focus** : 폼의 구성원의 필드에 입력을 하기 위해 선택하는 경우 발생
- **load** : 페이지를 브라우저로 읽을 때 발생하는 이벤트
- **mouseover** : 마우스 포인터를 하이퍼텍스트 링크 위로 옮길 때 발생

이벤트

- **select** : 품의 구성원의 필드를 선택한 경우에 발생하는 이벤트
- **submit** : 품을 등록한 경우에 발생하는 이벤트(품과 함께 제공되는 **submit**과 같은 입력이 끝났음을 알리는 버튼을 누르면 발생)
- **unload** : 지금 읽고 있는 페이지 외의 다른 페이지를 읽고자 할 경우 발생

■ 이벤트 핸들러(처리기)란 ?

- 특정 이벤트와 이에 대한 조치 내용을 담은 코드를 연결하는 역할
- 자바스크립트에서는 주로 사용자 입력과 관계된 **form**과 **anchor**에 대해서
- 이벤트 핸들러를 제공

이벤트

■ 이벤트 핸들러의 기본 형태

- **<태그명 태그_속성 이벤트_핸들러="JavaScript program">**

ex) `<INPUT TYPE="text" onChange="checkField(this)">`

- 스크립트 코드 전체가 태그 안에 들어갈 수 있다

ex) `<INPUT TYPE="text" onChange="`
 `if (parseInt(this.value) <= 5) {`
 `confirm("value error");`
 `}`
`">`

자바스크립트의 내장 객체

● event의 종류(1)

blur	입력양식 필드에서 포커스가 다른 곳으로 이동했을 때
click	입력양식이나 링크를 마우스로 클릭했을 때
change	입력양식 필드에 있는 값을 바꾸었을 때
focus	입력양식 필드로 포커스가 들어왔을 때
abort	이미지를 읽다가 중단시켰을 때
mouseover	링크위로 마우스가 지나갈 때
mouseout	마우스가 링크나 특정영역안에 있다가 나갔을 때
select	입력양식의 한 필드를 선택했을 때
submit	입력양식을 서버로 보낼때
load	브라우저가 문서를 읽을 때
unload	브라우저에서 문서가 없어질 때
error	문서나 이미지를 읽다가 에러 발생시
reset	입력양식에서 리셋했을 때

자바스크립트의 내장 객체

- event의 종류(2)

dblclick	마우스 더블 클릭시
dragdrop	마우스를 클릭한 상태에서 움직일 때
keydown	키를 입력했을 때
keypress	키를 눌렀을 때
keyup	키를 눌렀다 놓았을 때
mousedown	마우스 버튼을 눌렀을 때
mousemove	마우스를 움직였을 때
mouseup	마우스를 눌렀다 놓았을 때
move	윈도우나 프레임이 움직였을 때
resize	윈도우나 프레임의 크기를 바꾸었을 때

이벤트

■ 객체별 이벤트 핸들러의 종류

선택 목록(SELECT)	onBlur, onChange, onFocus
문자 필드(TEXT)	onBlur, onChange, onFocus, onSelect
문자 영역(TEXTAREA)	onBlur, onChange, onFocus, onSelect
버튼(BUTTON)	onClick
체크박스(CHECKBOX)	onClick
라디오 버튼(RADID BUTTON)	onClick
하이퍼텍스트 링크	onClick, onMouseover
RESET 버튼	onClick
SUBMIT 버튼	onClick
DOCUMENT	onLoad, onUnload
WINDOW	onLoad, onUnload
폼(FORM)	onSubmit

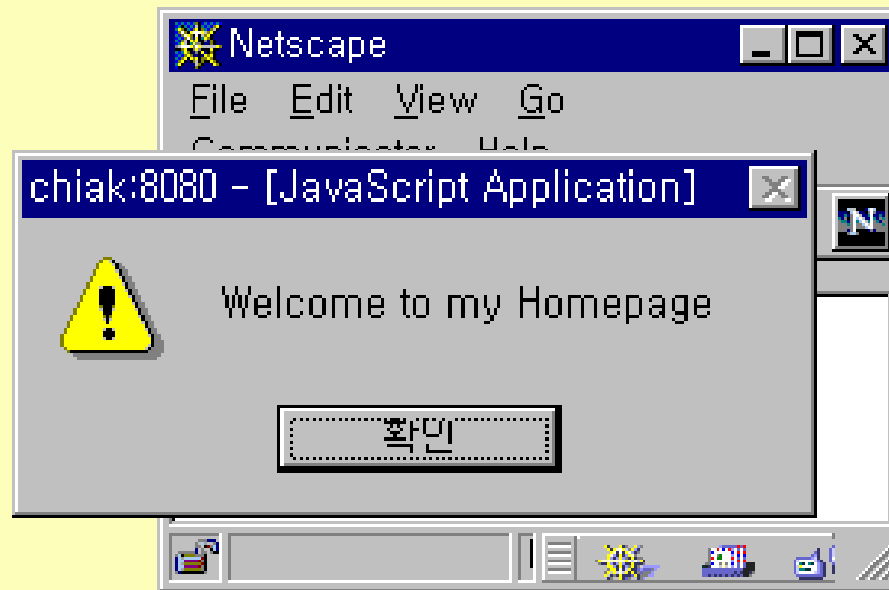
이벤트

<HTML>

<BODY onLoad="alert('Welcome to my Homepage');">

</BODY>

</HTML>



자바스크립트의 내장 객체

■ Event Handler(이벤트 핸들러)

- 이벤트에 대한 자바스크립트의 반응.
- 자바스크립트에 대한 동작 명령
- 각각의 이벤트를 처리
- 자바스크립트는 기본적으로 이벤트 핸들러에 의해 동작을 시작 (예외도 있음)
- HTML 태그의 한 속성으로 지정될 수 있다.
`<input type="button" value="꼭 눌러" onClick="function()">`
- 이벤트 핸들러는 클릭이 발생했을 때 (onclick), 입력양식에 포커스가 왔을 때 (onfocus), 문서가 로딩되었을 때 (onload), 링크위로 마우스가 왔을 때 (onmouseover), 입력양식의 한 필드가 선택되었을 때 (onselect), 페이지를 떠날 때 (onunload), 등으로 각각의 이벤트에 대한 반응 양식이다.
- 이벤트 핸들러는 각각의 이벤트에 on을 붙여주기만 하면 된다.

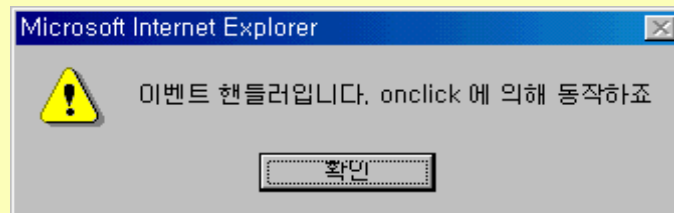
자바스크립트의 내장 객체

■ EventHandler의 사용예

```
// HEAD에 들어갈 자바스크립트
<script language="javascript">
    function greeting() {
        alert('이벤트 핸들러입니다. onclick 에 의해 동작하죠')
    }
</script>

// BODY에 들어갈 폼 태그

<input type="button" value="누르세요" onclick="greeting()">
```



자바스크립트 내장 객체

■ Event 객체

- 이벤트를 처리하기 위해서 넷스케이프 4.0부터 제공되는 객체로 넷스케이프와 익스플로러의 객체가 서로 조금씩 다름.

■ 이벤트 핸들러 함수

- **event**객체는 선언해서 사용하지 않지만 매개 변수를 전달
- **e**라는 매개변수는 다른 이름으로 사용해도 되지만 매개변수를 빼먹으면 안됨.

```
function 함수명(e)
{
    처리내용
    .
    .
}
```


자바스크립트 내장 객체

■ 넷스케이프 이벤트 속성

속성	설명
data	드래그앤 드롭 이벤트로 선택된 URL 주소를 알아냄.
layerX	레이어를 기준으로 이벤트가 발생한 위치의 X 좌표
layerY	레이어를 기준으로 이벤트가 발생한 위치의 Y 좌표
modifiers	마우스와 함께 눌려진 키를 알아냄
pageX	HTML 문서를 기준으로 이벤트가 발생한 X 좌표
pageY	HTML 문서를 기준으로 이벤트가 발생한 y 좌표
screenY	화면을 기준으로 이벤트가 발생한 y 좌표
screenX	화면을 기준으로 이벤트가 발생한 x 좌표
target	이벤트를 전달할 객체 설정
type	이벤트의 종류 설정
which	마우스 버튼의 종류를 알아내거나 입력한 ASCII 키 값으로 알아냄.
srcElement	이벤트가 전송된 원래 객체를 알아냄.

자바스크립트 내장 객체

■ 익스플로러의 이벤트 속성

속성	설명
altKey	alt상태를 true, false 값으로 알아냄.
button	버튼의 종류를 알아냄. 1번 왼쪽, 2번 오른쪽, 3번 가운데
clientX	HTML 문서를 기준으로 이벤트가 발생한 X좌표 위치를 알아냄.
clientY	HTML 문서를 기준으로 이벤트가 발생한 X좌표 위치를 알아냄.
ctrlKey	ctrl키 상태를 true, false 값으로 알아냄.
keyCode	키를 누른 상태를 알아냄
offsetX	컨테이너를 기준으로 이벤트가 발생한 위치의 x좌표
offsetY	컨테이너를 기준으로 이벤트가 발생한 위치의 y좌표
reason	데이터 소스 객체에 대한 데이터 전송상태를 알아내는데 사용
returnValue	이벤트 핸들러의 반환값을 true, false로 알아냄.
screenX	넷스케이프와 동일
screenY	넷스케이프와 동일

자바스크립트 내장 객체

■ 익스플로러의 이벤트 속성

속성	설명
shiftKey	shift키 상태를 true, false 값으로 알아냄
srcElement	넷스케이프와 동일
type	넷스케이프와 동일
X	상대적인 X좌표 위치
Y	상대적인 Y좌표 위치

■ 넷스케이프와 익스플로러 모두 지원하는 이벤트html 소스



자바스크립트 내장 객체

■ 이벤트 잡아내기(익스플로러 지원안됨)

속성	설명
captureEvents	HTML 문서의 특정 이벤트를 잡아냄
handleEvent	이벤트를 특정 객체의 이벤트 핸들러로 전달
releaseEvents	captureEvents 메서드로 지정된 이벤트를 중지
routeEvent	잡아낸 이벤트를 전달하기 위해서 사용

- 지원객체 : window, document, layer

- 형식

```
window.captureEvents(Event.이벤트)  
document.captureEvents(Event.이벤트)  
layer.captureEvents(Event.이벤트)
```

- 이벤트에는 이벤트 문자를 대문자로 넣는다.

- Event.CLICK

- window.captureEvents(Event.CLICK | Event.MOUSE



자바스크립트 내장객체

■ onError 이벤트

- HTML 문서를 읽어 올 때 발생한 오류를 처리하기 위해 사용.
- 익스플로러는 에러메시지가 한글로 출력

매개변수	설명
errorMessage	에러 문자열
url	URL 주소
line	에러가 발생한 라인 번호

■ 사용 예



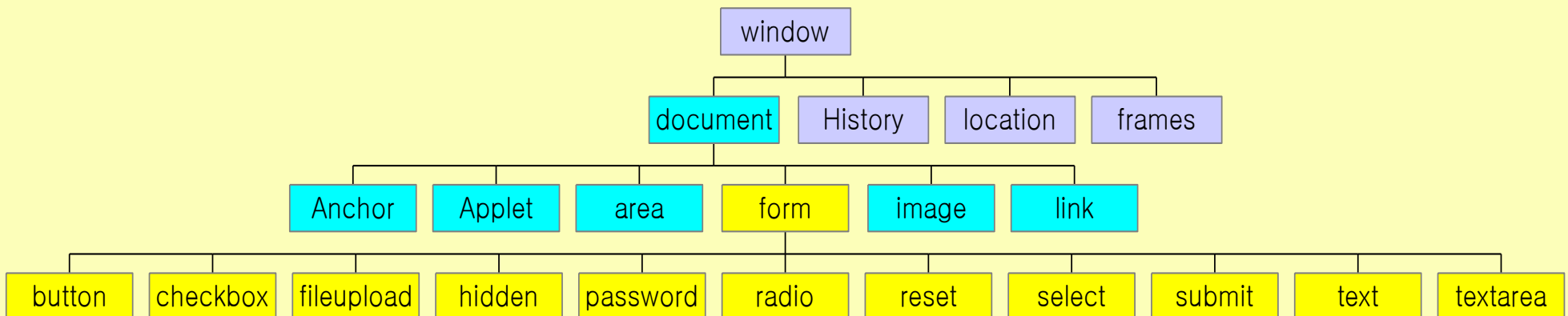
브라우저 내장 객체

■ 정의

- 현재 브라우저가 보여주고 있는 문서에 관한 여러 가지 정보뿐만 아니라 윈도우 정보, 히스토리 정보, 문서가 존재하는 위치에 관한 정보 등을 모두 포함

■ 브라우저 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



브라우저 내장객체

■ window 객체의 속성

● frame 객체와 중복되어 포함

매개변수	설명
classes	문서안에 정의된 모든 css 클래스들의 정보 포함
defaultsStats	브라우저 상태바에 초기 문자열을 설정하는 경우 사용
frames	windows 객체 안의 프레임들의 배열 정보를 가지고 있음.
innerHeight	브라우저 내용의 높이 값을 반환(익스플로러 지원 안 함)
innerWidth	브라우저 내용의 너비 값을 반환(익스플로러 지원 안 함)
opener	open() 메서드를 이용하여 연 문서를 말함
outerHeight	브라우저 높이 값을 반환(익스플로러 지원 안 함)
outerWidth	브라우저 너비 값을 반환(익스플로러 지원 안 함)
pageXOffset	현재 나타나는 페이지의 x 위치 반환
pageYOffset	현재 나타나는 페이지의 y 위치 반환
parent	계층 구조가 생길 때 바로 상위의 window 객체를 가리킬 때 사용

브라우저 내장객체

매개변수	설명
self	현재 활성중인 창의 자신 객체
status	브라우저의 상태바에 문자열을 출력하는 경우에 사용
tags	문서안에 정의된 모든 태그의 정보를 가짐
top	계층 구조가 생길 때 최상위의 window 객체를 가리킬 때 사용



- 예:status 속성 사용

■ window 객체의 메서드

- 새로운 창을 만들거나 메시지를 보여주거나 시간을 측정하는 등 다양하게 제공됨.

매개변수	설명
alert()	메시지를 대화상자에 보여줌
back()	이전화면으로 이동, 익스플로러 지원 안함.
close()	open() 메서드로 연 창을 닫습니다.

브라우저 내장객체

매개변수	설명
confirm()	확인,취소를 선택할 수 있는 대화상자를 보여줌.
find()	문자열에서 특정 문자열을 찾습니다. 익스플로러는 지원하지 않음.
forward()	다음 화면으로 이동, 익스플로러는 지원하지 않음.
home()	초기화 홈페이지로 이동, 익스플로러는 지원하지 않음.
moveBY()	상대적인 좌표로 이동
moveto()	절대적인 좌표로 이동
open()	새로운 창을 열어줌
print()	화면에 있는 내용을 프린터로 출력
prompt()	문자열을 입력받을 수 있는 대화상자를 보여줌.
resizeBy()	상대적인 좌표로 창의 크기를 설정
resizeTo()	절대적인 좌표로 창의 크기를 설정
scrollBy()	상대적인 좌표로 스크롤 위치값을 설정
scrollTo()	절대적인 좌표로 스크롤 위치값을 설정

브라우저 내장 객체

매개변수	설명
setInterval()	일정간격으로 함수를 호출하여 수행, 함수를 한번만 호출하면 됩니다. setTimeout() 메서드를 개선한 메서드
setTimeout()	일정간격으로 함수를 호출하여 수행, 함수를 재귀호출하여 사용
stop()	문서전송을 중지, 익스플로러는 지원하지 않습니다.

- alert/confirm/prompt 메서드

- window 객체를 앞에 삽입하지 않고 독립적으로 사용하기 때문에 내장함수로 포함하기도 함.
- 여러 라인의 문자열을 출력하거나 역슬래시(\)를 출력하고자 할 때는 문자기호를 사용

- open() 메서드

- open("문서명","창이름","윈도속성문자열")
- 현재 브라우저에서 새로운 브라우저가 출력되는 것을 의미
- 문서명:html 파일 설정
- 창이름 : 열리는 창의 이름
- 윈도 속성 문자열 : 툴바 메뉴, 상태바 등을 창에서 추가하거나 삭제할 수 있도록 문자열을 제공

브라우저 내장 객체

- open() 메서드 실행 예
- [윈도 속성 문자열]



속성	속성값	설명
directories	yes,no	디렉토리 메뉴(익스플로러만 지원)
location		주소(URL)
menubar		디렉토리 메뉴
scrollbars		스크롤바
status		상태바
toolbar		툴바 메뉴(뒤로, 앞으로, 중지 등)
copyhistory		히스토리 복사 정보
resizable		브라우저 크기 조절
Left, top		브라우저의 위치 정보
width/height	픽셀수	창의 너비/높이 설정

브라우저 내장 객체

- **close() 메서드**

- **open()** 메서드를 이용하여 만든 새로운 창을 닫기 위하여 사용
- **close()** 메서드는 매개 변수를 설정하지 않고 반환되는 값도 없음.
- **형식 : 객체.close()**
 - **open()** 메서드를 사용한 후 할당된 객체를 **close** 앞에 사용
 - 현재 열려있는 창인 경우에는 **self** 객체를 사용하여 닫아줌.
- **사용예 : self** 객체를 사용하여 창 열고 닫기

- **setTimeout 메서드 : 일정한 간격으로 함수를 호출하여 수행할 수 있게 해줌.**



- **clearTimeout 메서드 : 일정한 간격으로 수행하는 함수 중지**

- **형식 : 변수 = setTimeout(호출함수, 간격)**
clearTimeout(변수)

브라우저 내장 객체

- setInterval 메서드

- setTimeout() 메서드를 개선하여 나온 메서드
- setTimeout() 메서드는 일정한 간격으로 함수를 호출하기 위하여 재귀호출을 사용하여 매번 함수를 호출
- setInterval() 메서드는 함수를 한 번만 호출하면 자동으로 일정한 간격으로 함수를 호출.
- 형식 : 변수 = setInterval(호출함수, 간격)



- resizeBy/resizeTo 메서드

- 브라우저의 창 크기를 변경하는데 사용하는 메서드.
- resizeBy() : 상대적인 길이를 입력하여 현재 브라우저의 창크기를 축소/확대
- resizeTo() : 절대적 좌표를 지정하여 브라우저의 크기를 설정



브라우저 내장 객체

- moveBy/MoveTo 메서드

- 브라우저 위치를 변경하는데 사용
- moveBy() : 상대적인 위치를 입력하여 현재 브라우저의 위치를 변경
- moveTo() : 절대적 좌표를 지정하여 브라우저의 위치를 설정
- 형식 : moveBy(x 위치, y 위치)
moveTo(x좌표, y좌표)

- print 메서드

- 화면에 있는 내용을 모두 출력
- 형식 : window.print()



브라우저 내장 객체

■ window 객체의 이벤트

메서드	설명
onBlur	브라우저에 포커스를 잃을 때 발생합니다.
onDragDrop	사용자가 다른 곳에서 객체를 브라우저 안에 놓으려고 할 때 발생, (익스플로러는 지원 안 함)
onError	문서를 읽는 중에 에러가 생길 때 발생
onLoad	문서를 읽을 때 발생합니다.
onMove	브라우저의 위치를 변경했을 때 발생(익스플로러는 지원 안 함)
onResize	창의 크기를 변경했을 때 발생(익스플로러는 지원 안 함)
onUnload	현재 문서를 지우려고 할 때 발생

Window 객체

■ 다른 모든 객체의 최상의 계층

- `window.alert()` 대신에 `alert()`라고 해도 무방함

■ 윈도우 생성/소멸

- `open("URL", "윈도우 이름", "특성목록");`

`toolbar=[yes | no | 1 | 0]`

`scrollbars=[yes | no | 1 | 0]`

`location=[yes | no | 1 | 0]`

`resizable=[yes | no | 1 | 0]`

`directories=[yes | no | 1 | 0]`

`width=pixels`

`status=[yes | no | 1 | 0]`

`height=pixels`

`menubar=[yes | no | 1 | 0]`

- `close();`

브라우저 내장 객체

■ History 객체

- 정의
 - 브라우저를 이용하여 본 URL 사이트를 임시로 저장
 - 뒤로(BACK), 앞으로(FORWARD) 버튼 등을
 - 뒤로, 앞으로 버튼은 브라우저 창에서 만들어 낼 수 있음.
- 속성 : **length**는 브라우저로 읽어온 URL 주소의 개수를 알아 낼 수 있음.
 - `history.length()` (익스플로러는 0 부터)
- 메서드

메서드	설명
<code>back()</code>	브라우저로 본 이전 화면으로 이동
<code>forward()</code>	브라우저로 본 다음 화면으로 이동
<code>go()</code>	상대적인 숫자를 설정하여 본 화면을 이동

브라우저 내장 객체



- 형식 history.back()

history.forward()

history.go(번호, URL)

■ Location 객체

- 브라우저 창에 읽어온 문서를 제어할 수 있음.

URL 종류	프로토콜	사용예
자바스크립트 코드	javascript:	javascript:Date()
브라우저 정보	about:	about:cache
WWW	http:	http://www.yahoo.co.kr
File	file:	file:///javascript/meghods.html
FTP	ftp:	ftp://ftp.mine.com/home/mine
Mailto	mailto:	mailto:info@netscape.com
Gopher	gopher:	gopher.myhost.com

브라우저 내장 객체

● location 객체 속성

속성	설명
hash	표식이름을 설정하거나 알아냄.
host	URL 주소의 호스트 이름과 포트를 설정하거나 알아냄
hostname	URL 주소의 호스트 이름이나 호스트의 ip 주소를 설정, 알아냄
href	문서의 URL 주소를 설정하거나 알아냄.
pathname	문서의 디렉토리 위치를 설정하거나 알아냄
port	포트 번호를 설정하거나 알아냄.
protocol	프로토콜 종류를 설정하거나 알아냄
search	검색엔진을 호출할 때 사용

● location 객체 메서드



메서드	설명
reload()	현재 문서를 다시 읽어옴
replace()	현재 문서를 다른 URL 문서로 바꿔줌.

브라우저 내장 객체

■ Navigator 객체

● 정의 :

- 브라우저 객체 중에서 독립적으로 사용
- 브라우저 종류, 사용언어, 시스템 종류
- 실무에서 가장 많이 사용하는 부분은 어떤 브라우저로 홈페이지에 들어왔는가를 알아낼 때 사용

속성	설명
appName	브라우저의 코드명을 반환
appVersion	현재 사용중인 브라우저의 버전 반환
language	현재 브라우저가 사용하는 언어 반환(익스플로러 지원 안 함)
mimeType	mime 형식의 정보 반환
platform	사용중인 시스템 코드 반환

브라우저 내장 객체

속성	설명
plugins	플러그인 정보 반환(익스플로러 지원 안 함)
userAgent	브라우저의 이름, 버전, 코드를 포함하는 문자열 반환

● Navigator 객체 메서드

속성	설명
javaEnabled()	현재 사용하는 브라우저가 자바를 사용가능한지 상태를 알아냄.
taintEnabled()	현재 브라우저가 문서의 정보를 정상적으로 읽어 왔는지, 잘못되었는지를 알아냄.

Document 객체

■ 속성

- 링크 관련 속성이나 새 문서 속성으로 구성

속성	설명
alinkColor	누르고 있는 동안의 링크 문자열의 색을 설정
bgColor	창의 배경색을 설정합니다.
cookie	쿠키를 사용할 수 있게 해줍니다.
fgColor	글자색을 설정합니다.
images	이미지 배열 정보를 포함하고 있습니다.
lastModified	가장 최근에 수정한 문서 날짜를 알아냅니다.
layers	레이어 배열 정보를 포함하고 있습니다.
linkColor	링크색을 설정
referrer	링크된 이전 문서의 URL 정보를 알아냄.
title	문서의 <title></title> 태그 안에 있는 제목을 설정 또는 반환
vlinkColor	누른적 있는 링크 문자열의 색을 설정

실습 : document객체 사용 테이블 만들기

- 자바스크립트 document객체를 사용해서 2행2열의 테이블을 작성하시오.

Document 객체의 메서드

■ 메서드

속성	설명
clear()	문서의 내용을 지웁니다.
close()	open() 메서드로 연 문서를 닫아줌.
getSelection()	마우스로 드래그한 문자열을 반환
open()	문서의 데이터 정보를 출력하기 위해 준비
write()	태그를 포함하는 문자열을 문서에 출력
writeln()	태그를 포함하는 문자열을 문서에 출력하고 줄바꾸기를 포함. <pre> 태그 안에서 사용될 때 줄바꾸기를 함.

- write/writeln 메서드에 사용하는 문자열은 HTML 태그를 포함할 수 있는 문자열이고, 콤마를 이용하여 연속적으로 문자열을 넣어 사용할 수 있음.

Document에 포함된 객체

■ Document에 포함된 객체

- Anchor, Applet, Area, Form, Image, Link 객체 포함

■ anchor 객체

- 용도 : 링크된 문장을 특정위치에 이동시킬 수 있는 <a> 태그를 다룰 수 있게 해줌.
- 속성 : **length**를 이용하여 몇번 사용했는지 알아낼 수 있음.
 - document.anchors.length
 - document.anchors[인덱스 번호]

■ link 객체

- 용도 : <a href> 태그를 처리할 수 있게 속성과 이벤트 제공

Document 에 포함된 객체

- links 속성

- HTML 문서 안에 있는 <A HREF> 태그에 관한 정보를 배열로 포함.
- 형식 : document.links[인덱스번호]

document.links.length

- link 속성(운영체제 환경에 따라 다름)

속성	설명
hash	표식 이름을 알아냄
host	링크에 연결된 URL, 호스트 이름, 포트 번호를 알아냄.
hostname	링크에 연결된 URL과 호스트 이름을 알아냄.
href	링크에 연결된 문서의 URL을 알아냄.
pathname	디렉토리 위치를 알아냄.
port	포트 번호를 알아냄.
protocol	프로토콜 종류를 알아냄.

Document 에 포함된 객체

속성	설명
search	검색엔진을 불러낼 때 사용
target	지정된 URL에 연결된 후 보여줄 창을 알아냄.

- link 사용 예
- link 이벤트



속성	설명
onClick	마우스로 클릭했을 때 발생
onDbClick	마우스로 두 번 클릭했을 때 발생
onKeyDown	키를 눌렀을 때 발생
onKeyPress	키를 눌렀다 놓았을 때 발생
onKeyUp	키를 놓았을 때 발생
onMouseDown	마우스를 눌렀을 때 발생
onMouseOut	마우스가 링크 밖으로 벗어났을 때 발생

Document 에 포함된 객체

속성	설명
onMouse	마우스를 눌렀을 때 발생
onMouseOut	마우스가 링크 밖으로 벗어났을 때 발생

■ Image 객체

- 정의 : html 문서안에 있는 이미지를 처리할 수 있도록 속성과 이벤트 제공
- images 속성 : html 문서 안에 있는 태그의 정보를 가지고 있음.
 - document.images.length
 - document.images[l]
- image 속성

속성	설명
align	이미지 정렬 방식을 알아냄
alt	이미지 설명을 알아냄

Document 에 포함된 객체

속성	설명
border	이미지 테두리 값을 알아냄.
complete	이미지 전송이 끝났는지 여부를 알아냄.
height	이미지의 높이를 알아냄.
hspace	왼쪽 여백값을 알아냄.
lowsrc	이미지 파일의 위치, 해상도가 작은 파일로 이미지를 전송하는 동안에 보여줄 이미지 파일을 말함.
name	이미지 이름을 알아냅니다.
prototype	이미지 객체에 속성을 추가하기 위해서 사용
src	이미지 파일을 알아내거나 설정
vspace	오른쪽 여백값을 알아냄
width	이미지의 너비를 알아냄.



● 사용예

Document 에 포함된 객체



■ 이미지 이벤트를 이용한 실습

번호	Html 파일	이미지 파일명
1	INDEX	MENU.HTML, MAIN.HTML 파일을 읽어와서 창을 나눕니다.
2	MENU	왼쪽 메뉴를 보여줍니다.
3	MAIN	처음 화면의 오른쪽 내용을 보여줍니다.
4	MENU	images/back.gif
5		images/menu1.gif, images/menu1-1.gif
6		images/menu2.gif, images/menu2-1.gif
7		images/menu3.gif, images/menu3-1.gif
8	MAIN	Images/back1.gif
9		images/titlecopy1.gif
10		Images/pic01.jpg

Document 에 포함된 객체

■ 설명

- 이미지 객체를 사용하기 위해선 **new** 연산자를 이용하여 **image()**에 할당해 주고 **src** 속성을 이용하여 경로가 포함된 이미지 파일을 생성
 - `img.file0_off = new Image();`
 - `img.file0_off.src = "images/menu1.gif";`
- 메뉴에는 서로 두개의 상반된 이미지 사용
- 이미지 변경하기
 - **OnMouseOut** 이벤트는 마우스가 메뉴 항목밖으로 벗어날때 발생, **OnMouseOver** 이벤트는 마우스가 메뉴 위로 지나갈 때 발생
 - ``
 - ` `
 - **isimgact()**는 여러 개의 메뉴 항목을 쉽게 작성하기 위해 일정한 규칙을 갖는 함수가 필요하기 때문에...
 - 형식 : `isimgact(파일id, 액션)`

Document 에 포함된 객체

```
isamap = new Object();  
isamap[0] = "_off"  
isamap[1] = "_on"  
function isimgact(id, act)  
{  
    if(document.images)  
        document.images[id].src = eval( "img." + id + isamap[act] +  
        ".src");  
}
```

- OnMouseOver="isimgact('file0',1)' -> img.file0_on.src 생성

쿠키(Cookie)

■ 정의

- 사용자측에서 정보를 보관해 두었다가 웹 서버의 요청에 의해 그 정보를 원하는 순간에 사용하는 것
- 클라이언트와 서버의 관계에서 클라이언트는 서버쪽에서 보내는 데이터를 사용
- 쿠키를 이용하여 클라이언트측에 데이터를 저장하면, 서버쪽에서는 클라이언트측의 데이터 정보를 이용하여 다양한 형태로 처리할 수 있음.
- 사용자의 방문횟수, 방문 시기, 구매 품목 등
- 쿠키 크기 : **4KB** 이하, **300**개까지의 정보 배열 사용 가능($300 \times 4KB = 1,2MB$)
- 저장하는 정보는 간단한 문자열, 기호화된 코드 형태로
- **300**의 배열은 클라이언트에서 오래된 정보를 자동으로 삭제한다는 의미

쿠키(Cookie)

■ Set-Cookie 구조

- 'name=VALUE' 속성은 반드시 필요
 - 사용자로부터 쿠키정보 인가 단순한 문자열가를 식별하기 위해 사용(콤마, 세미콜론 등을 포함할 수 없음.)
- Name=VALUE; expries=DATE; path=PATH; domain= DOMAIN; secure
 - 쿠키의 문자열의 속성과 속성사이에는 세미콜론(;)으로 구분하고 마지막에 없음.

속성	설명
Name=value	Set-Cookie인지를 식별하기 위해서 사용
Expries=DATE	쿠키가 사용될 수 있는 유효기간을 설정
Path = PATH	저장된 쿠키의 URL 경로를 설정
Domain=DOMAIN	저장된 쿠키와 일치하는 URL 경로의 도메인 부분 설정
Secure	데이터 전송할 때의 보안여부를 true나 false로 설정

쿠키(Cookie)

■ SetCookie() 함수

- 속성값을 넣으면 자동으로 쿠키 형식의 문자열 작성
- SetCookie() 함수 소스

//쿠키 형식의 문자열을 만들어 줌.

```
function SetCookie(name, value) {  
    var argv = SetCookie.arguments  
    var argc = SetCookie.arguments.length  
    var expires = (2 < argc) ? Argv[2] : null  
    var path = (3 < argc) ? Argv[3] : null  
    var domain = (4 < argc) ? Argv[4] : null  
    var secure = (5 < argc) ? Argv[5] : null  
    document.cookie = name + "=" + escape(value) +  
        ((expires == null) ? "" : ("; expires=" + expires.toGMTString())) +  
        ((path == null) ? "" : ("; path=" + path)) +  
        ((domain == null) ? "" : ("; domain=" + domain)) +  
        ((secure == true) ? "" : "secure") ;  
}
```

쿠키(Cookie)

- argc변수 : SetCookie() 함수의 파라미터 정보의 배열
- argv : 파라미터 개수

● 예제 : SetCookie() 함수가 쿠키 형식의 문자열 생성

● 예제 : 사용자 방문 횟수 보여 주기

- 작성원칙 :

- 클라이언트 쪽에 쿠키가 생성되어 있지 않을 경우에는 방문수를 0으로 만들어줌.
- 클라이언트 쪽에 쿠키가 생성되어 있으면 쿠키 정보의 유효 기간을 확인한 후 유효기간 이내이면 방문자 횟수 증가

- 설명

- 방문자 횟수 처리
 - getTime()으로 날짜를 읽어 온 후 유효기간을 1달로 설정
 - setTime()으로 시간 설정
`myday.setTime(myday.getTime() + 30 * 24 * 60 * 60 * 1000))`
 - 클라이언트측에 쿠키 내용이 없으면 초기화 "sms = 0"



쿠키(Cookie)

- 날짜 정보와 쿠키 방문수를 `SetCookie()` 함수로 저장
`SetCookie("sms", sms, myday, "/", null, false)`
- 방문자 횟수가 저장된 `sms` 변수를 사용하여 방문수 출력
`document.write("이 홈페이지에" + 는 + "번째 방문...<p>")`
- 쿠키 정보 읽어 오기
 - 쿠키 이름을 사용하여 쿠키 길이 만큼 `while` 문을 이용하여 반복
 - "`sms=`"을 쿠키에서 찾으려면 `getCookieVal()` 함수 수행
 - `substring()`은 문자열 중에 일부분을 추출하기 위해 사용
 - `J`는 쿠키 배열 위치
 - `indexOf()` 메서드를 이용하여 문장 중의 인덱스 번호를 알아냄.
 - 횟수가 1이면 `sms = 1`중에 1을 얻기 위해
 - 쿠키의 문자열 구분은 "`;`"으로 하기 때문에 세미콜론의 위치를 `indexOf()` 메서드로 알아냄.

```
function GetCookie(name)
    var argv = name + "="
    var alen = arg.length
    // 쿠키 개수
    var clen = document.cookie.length
    var I = 0
    while (I < clen) {
        var j = I + alen
        if (document.cookie.substring(I, j) == arg)
            return getCookieVal (j)
        I = document.cookie.indexOf(" ", i) + 1
    }
}
```

```
function getCookieVal(offset) {
    var endstr = document.cookie.indexOf(";", offset)
    if (endstr == -1)
        endstr = document.cookie.length
    return unescape(document.cookie.substring(offset, endstr))
}
```

대화형 품의 개발

■ 품 객체

- 자바스크립트에서 가장 비중있게 사용되는 객체
- 품을 구성하는 요소 및 각 요소에 할당된 값 등을 알 수 있으며, 저장된 값을 필요에 따라 수정할 수 있다
- 자바스크립트의 품은 배열 **forms[]** 의 인덱스로 사용하여 여러 개의 품을 구별하거나, 또는 각 품에 주어진 이름으로 참조할 수 있다
- 품을 구성하는 구성원은 배열 **elements []** 로 지칭되어 인덱스를 사용하여 참조하거나, 또는 그 이름을 사용하여 참조할 수 있다

ex) *document.form_name.element_name*

document.forms[0].element_name

document.forms[0].elements[0]

대화형 품의 개발

■ 품 객체의 프로퍼티

- **action** : **FORM** 태그의 **action** 속성의 내용을 포함하는 문자열
- **elements** : 품을 구성하는 요소에 대한 내용을 담고 있는 배열(체크박스, 선택 목록 등의 구성요소)
- **encoding** : **MIME** 유형에 대한 정보를 담고 있어서 서버로 전달되는 품의 내용을 부호화하는데 이용된다. **FORM** 태그의 **enctype** 속성의 설정치를 반영한다
- **name** : **FORM** 태그의 **name** 속성의 값을 포함하는 문자열
- **target** : 품에 의해 영향을 받게 될 윈도우의 이름을 포함하는 문자열

대화형 폼의 개발

- 폼 객체와 관련된 메서드
 - **submit()** : onSubmit 이벤트 핸들러를 사용하지 않고 submit 버튼이 눌러진 것을 흉내낸다
- 폼을 구성하는 구성원은 그 자체가 객체로서 프로퍼티 및 메서드를 지닌다

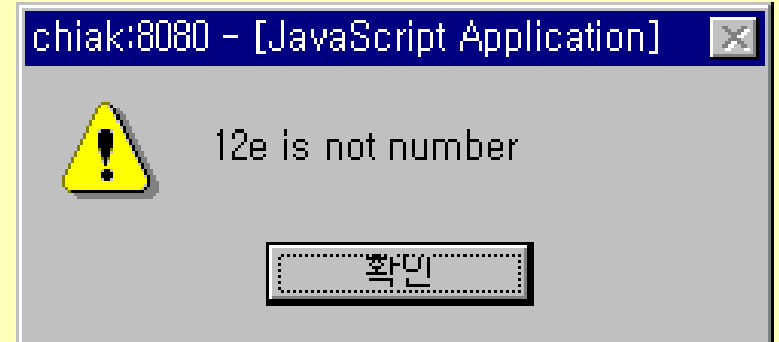
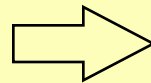
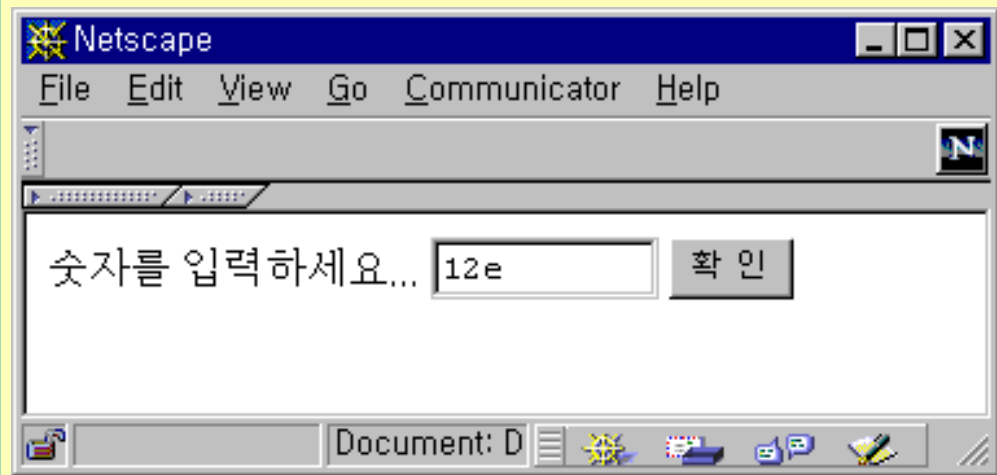
대화형 품의 개발

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!--
function isNumber(str) {
    for (i = 0; i < str.length; ++i) {
        if (str.charAt(i) < '0' || str.charAt(i) > '9')
            return false;
    }
    return true;
}
function checkNum(form) {
    if (!isNumber(form.num_field.value))
        alert(form.num_field.value + ' is not number');
    else
        form.submit();
}
//-->
</SCRIPT>
</HEAD>
```

대화형 품의 개발

```
<BODY>
  <FORM METHOD="POST"
    ACTION="chiak.yonsei.ac.kr/test.pl">
    숫자를 입력하세요...
    <INPUT TYPE="text" NAME="num_field" size=10>
    <INPUT TYPE="button" NAME="cnf" VALUE="확 인" onClick="checkNum(this.form);">
  </FORM>
</BODY>
</HTML>
```

대화형 품의 개발



자바스크립트의 입력 양식

■ Form 객체

- Html의 form 태그를 처리하기 위해 지원

● Form 객체에서 제공되는 속성, 메서드, 이벤트

구분		설명
속성	Action	<form> 태그의 action 속성과 동일하게 적용
	Elements	<form> 태그안에 있는 양식을 배열로 저장
	Encoding	<form> 태그의 enctype 속성과 동일하게 적용
	Length	<form> 태그안에 있는 양식의 수를 알아냄
	Name	<form> 태그의 name 속성과 동일하게 적용
	Method	<form> 태그의 method 속성과 동일하게 적용
	Target	<form> 태그의 target 속성과 동일하게 적용
메서드	Blur()	커서를 제거
	Reset()	양식을 새로 다시 입력받을 수 있게 초기화
	Submit()	입력한 양식에 서버에 전송

자바스크립트의 입력 양식

구분		설명
이벤트	onReset	리셋(reset) 버튼을 누를 때 발생
	onSubmit	서브미트(submit) 버튼을 누를 때 발생



- **action/encoding/length/name/method/target 속성**

- Form 태그의 속성을 제어할 수 있게 동일한 속성 제공
- Form 객체의 속성 형식

document.품명.action

document.품명.encoding

document.품명.length

document.품명.name

document.품명.method

document.품명.target



- **Element 속성**

- Form 태그안에 있는 양식에 어떤 값이 들어있는지 알아내거나 처리하기 위해서 사용

document.품명.elements[인덱스 번호]

자바스크립트의 입력 양식



- Forms 속성

- Html 문서안에서 사용된 **form** 태그에 관한 정보를 가짐
- 여러 개의 **form** 태그를 다룰 수 있음.
- **Length** 속성을 이용하여 **<form>** 태그를 몇 번 사용되었나 알아낼 수 있음.
 - `document.forms[인덱스번호]`
 - `document.forms.length`

- **reset()/submit()** 메서드

- **reset()** : 재입력할 수 있게 초기화
- **submit()** : 양식에서 입력한 내용을 서버에 전송

- **onReset/onSubmit** 이벤트

- 자바스크립트 에서 제공하는 **button** 객체를 사용한 후 **form** 객체의 **reset()**이나 **submit()** 메서드를 사용하는 방법



자바스크립트의 입력 양식

■ 양식에서 공통적으로 쓸수 있는 메서드

- toString() 메서드
 - 다양한 객체를 문자열로 변환해줌.
- valueOf() 메서드
 - 숫자, 문자열, 부울, **function** 객체의 원시값을 반환



■ Button 객체

- HTML의 버튼 입력 양식을 처리
- Button 객체의 속성/메서드/이벤트

구분		설명
속성	Name	<input> 태그의 name 속성과 동일하게 적용
	Type	<input> 태그에서 type 속성에 입력된 값 반환
	Value	<input> 태그의 value 속성과 동일하게 적용

자바스크립트의 입력양식

구분		설명
메서드	Blur()	특정한 버튼 양식에 포커스를 제거
	Click()	버튼을 클릭
	Eval()	수식으로 되어 있는 문자열을 계산하여 실수로
	Fcus()	버튼 양식에 포커스를 지정
이벤트	onBlur()	버튼양식이 포커스를 벗어날 때 발생
	onClick()	버튼양식을 클릭할 때 발생
	onFocus	버튼양식에 포커스가 설정되었을 때 발생

■ Text 객체

- 사용자가 입력하거나, 수정, 처리된 결과를 보여줄 수 있음.

자바스크립트의 입력양식

구분		설명
속성	defaultValue	defaultValue속성과 동일하게 적용. 초기의 value 속성에 설정된 문자열
	Name	Name 속성과 동일하게 적용
	Type	Type 속성과 동일하게 적용
메서드	Blur()	특정 HTML 텍스트 입력 양식에 포커스를 제거함.
	Value	텍스트 입력 양식에서 입력한 문자열을 가져옴.
	Eval()	수식으로 되어 있는 문자열을 계산하여 실수로
	Fcus()	버튼 양식에 포커스를 지정
이벤트	Select()	텍스트 입력 양식안에 있는 문자열을 모두 선택
	onBlur	버튼양식이 포커스를 벗어날 때 발생
	onChange	텍스트 입력 양식이 바뀔 때 발생
	onFocus	텍스트 입력 양식에 포커스가 설정되었을 때 발생
	onSelect	마우스를 이용하여 입력필드를 클릭하거나 선택했을 때

자바스크립트를 이용한 게임

25개의 불켜기 게임

게임개요

■ 게임명 : n개의 칸에 모두 불켜기



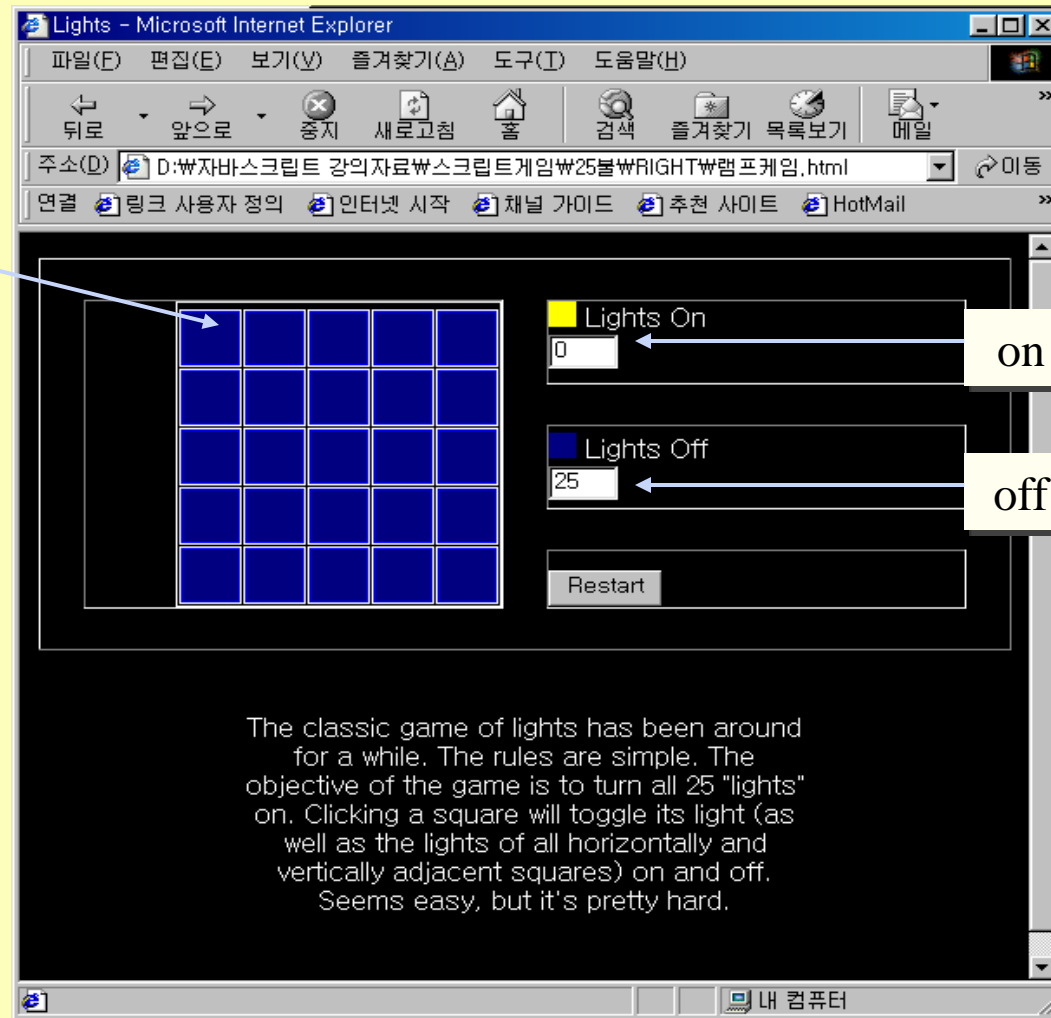
■ 개요

- $n \times n$ 개 사각표내에 off 상태인 칸을 모두 on 상태로 바꾸기
- 1회 클릭시 on/off, off/of 상태로 toggle
- $n \times n$ 개의 칸이 lights가 모두 on 상태면 승리
- 1회 toggle 된 것을 다시 click시 다시 toggle
- 불이 on 된 개수, off된 개수를 알려줌.
- 'restart'를 누르면 모두 off 상태로
- on 상태 : 노란색으로 전환
- off 상태 : 파란색으로 전환
- 브라우저가 자바를 지원 안하면 아무일도 하지 않음.

게임개요

■ 화면 설계

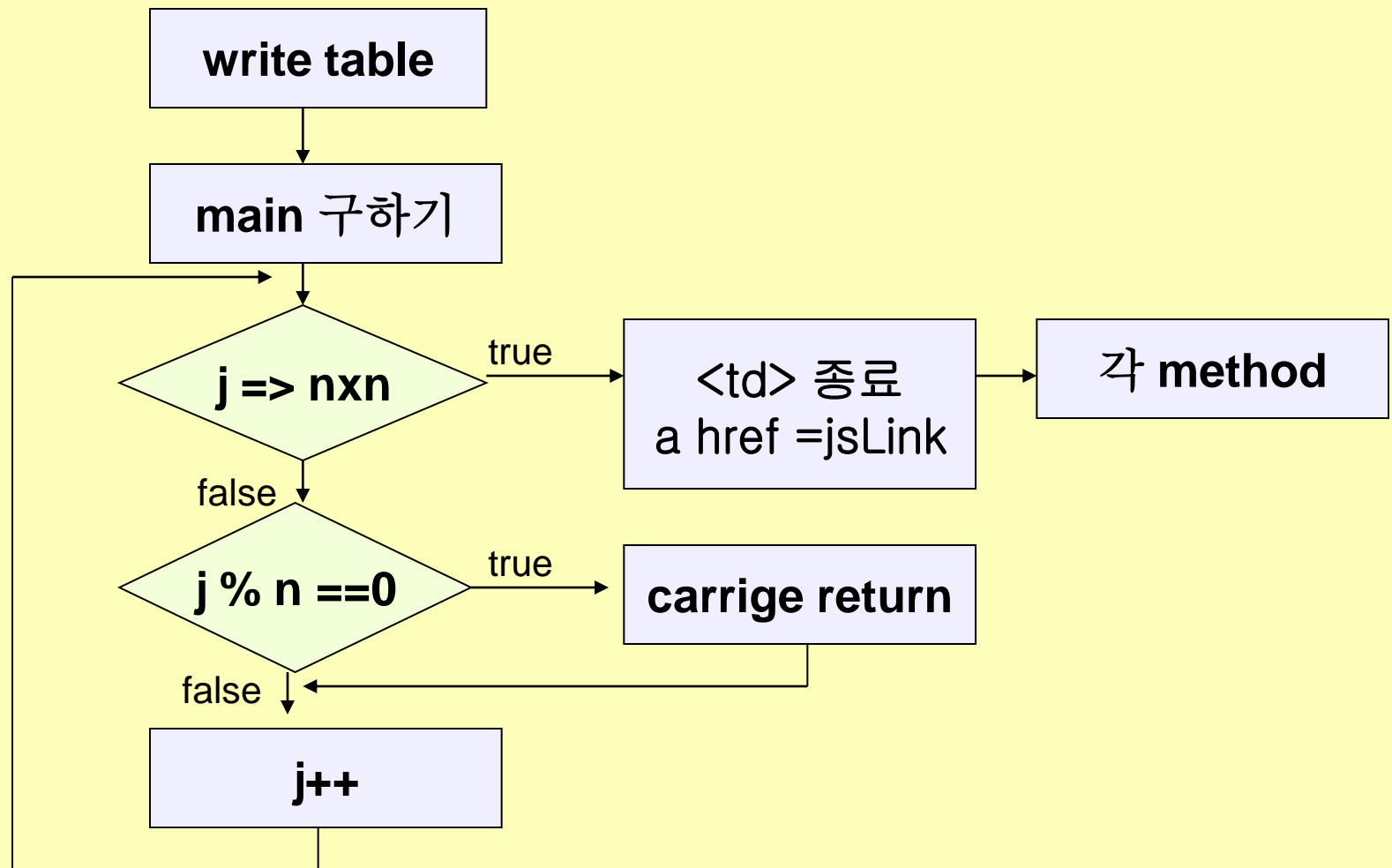
이미지 표시
칸



on 상태 개수

off 상태 개수

게임개요



게임개요

■ 필요 파일

- lights_0.gif : light_off
- lights_1.gif : light_on

■ 변수 정의(객체 정의)

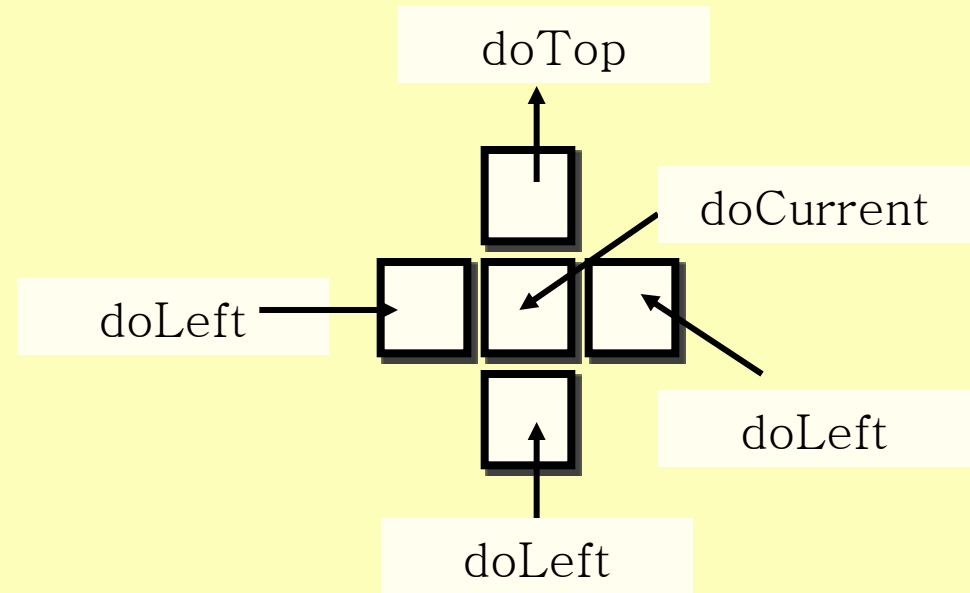
- lights_25 : Image 객체에서 “lights_” 파일 생성
- var off, var on : image 객체에서 on/off 상태
- browser : navigator 객체를 이용하여 java 지원 결정
- version : navigator 객체를 이용하여 브라우저 버전 생성
- numOff : 불의 on/off 개수
- turnlton : toggle 상태 저장(flag) 상태

게임개요

■ function의 정의

● main()

- 프로그램의 중심
- 세부 모듈(function)로 분기
 - doCurrent()
 - doBottom()
 - doTop()
 - doLeft()
 - doRight()
- 불의 켜진 개수 구함



게임개요

■ 각 method별 알고리즘

● doCurrent()

■ 현 마우스 포인터가 놓여진 element위치

```
- if (document.images[element].src == off.src) {  
    turnItOn = true;  
}  
if (document.images[element].src == on.src) {  
    turnItOn = false;  
}  
if (turnItOn) {  
    document.images[element].src = on.src; numOff--;  
}  
else {  
    document.images[element].src = off.src; numOff++;  
}
```

게임개요

- toTop()

- 현 element에서 -n한 요소
- element가 n 미만일 경우 doTop은 구하지 않음
 - if ((element >= n) {
 - if (document.image[element-n].src == off.src {
 - turnIfOn = true; }
 - if (document.image[element-n].src == on.src {
 - turnIfOn = false; }
 - if (turnIfOn) {
 - document.images[element - 5].src = on.src;
 - numOff--;
 - }
 - else {
 - document.images[element - 5].src = off.src;
 - numOff++
 - }

게임개요

- toBottom()

- 현 element에서 +n한 요소
- element가 ((nxn)-n) 한 값보다 클 경우는 구하지 않음.
 - if ((element) < 20) {
 if (document.images[element + 5].src == off.src) {
 turnItOn = true;
 }
 if (document.images[element + 5].src == on.src) {
 turnItOn = false;
 }
 if (turnItOn) {
 document.images[element + 5].src = on.src;
 numOff--;
 }
 else {
 document.images[element + 5].src = off.src;
 numOff++;
 }
}

게임개요

- toRight()

- 현 element에서 오른쪽 값 : element + 1위치
- element가 맨 우측 일 경우 right를 구하지 않음.

```
if (((element + 1) / 5) != Math.floor((element + 1) / 5)) {  
    if (document.images[element + 1].src == off.src) {  
        turnItOn = true;  
    }  
    if (document.images[element + 1].src == on.src) {  
        turnItOn = false;  
    }  
  
    if (turnItOn) {  
        document.images[element + 1].src = on.src;  
        numOff--;  
    }  
    else {document.images[element + 1].src = off.src;  
        numOff++;  
    }  
}
```

게임개요

- toLeft

- 현 element의 바로 왼쪽
- 맨 좌측 element인 경우는 toLeft를 구하지 않음.

```
if ((element / 5) != Math.floor(element / 5)) {  
    if (document.images[element - 1].src == off.src) {  
        turnItOn = true;  
    }  
    if (document.images[element - 1].src == on.src) {  
        turnItOn = false;  
    }  
    if (turnItOn) {  
        document.images[element - 1].src = on.src;  
        numOff--;  
    }  
    else {  
        document.images[element - 1].src = off.src;  
        numOff++;  
    }  
}
```