JavaScript

자바 스크립트 – 개요

JavaScript

자바스크립트 – 스크립트의 등장과 역할

- 최초의 스크립트 : 1987년 애플사의 HyperCard
- 발전계기 : 1990년대초에 MS사에서 VB에서 사용할 수 있는 VBA(VBApplication) 개발
- SunMicrosystems사가 인터넷 프로그래밍 언어로 Java를 개발
- 넷스케이프사는 선사와 전략적 제휴를 통하여, HTML 기능을 수용하면서 프로그래밍 개념을 대폭 수용한 JavaScript 개발

JavaScript

자바스크립트 – 소개

- 넷스케이프 네비게이터(또는 IE)에서 사용할 수 있는 스크립트 언어
- 넷스케이프는 Java를 만든 Sun Microsystems과 공동 프로젝트를 진행하여 라이브스크립트 (LiveScript)를 확장 시킨 자바스크립트를 만듦
- 객체 지향 스크립트 언어로서 프로그램 코드가 HTML 문서 사이에 직접 삽입
  - HTML 문서 내에서 바로 작성할 수 있으므로 빠르게 문서를 작성할 수 있다.
- 자바스크립트로 만든 프로그램에서는 사용자가 마우스를 클릭하거나 키보드를 입력하는 것과 같은 작업을 즉시 처리
  - 동적인 홈페이지 제작이 가능한 스크립트 언어다.
- 오직 클라이언트쪽에서만 실행
- HTML 소스 코드 내에서 바로 인식하여 전송하기 때문에 윈도우, 유닉스, 리눅스 등에 제한 없이 동작한다.

JavaScript

자바스크립트- 소개

- 라이브스크립트(LiveScript)
  - 썬 마이크로시스템사의 자바를 넷스케이프에서 스크립트 언어로 사용하기 위해 개발
- 객체 기반 스크립트 언어
  - 클라이언트 측의 브라우저에서 실행
  - 네트워크를 통한 데이터의 전송 없이 모든 작업을 처리
  - 서버의 업무 부담이 감소

JavaScript

자바스크립트 특징

- HTML 문서에 포함되어 실행되는 클라이언트 사이드 스크립트 언어
  - 웹 브라우저에서 웹 문서를 실행할 때 프로그램 코드가 해석
- 컴파일 과정 없이 스크립트를 직접 실행하는 인터프리터 언어
  - 클라이언트의 웹 브라우저에 의해 해석되고 실행
- 모든 플랫폼에서 인터프리터에 의해 실행이 가능
  - 컴파일 과정을 거치지 않는 인터프리터 언어의 형태이기 때문에 비교적 자료형 조사를 철저하게 하지 않음
- 간단한 문법과 사용법
  - HTML에 연산, 제어 등 프로그래밍적 요소를 만들 수 있음
- 동적인 웹 페이지 작성
  - 이벤트 중심의 프로그래밍 언어
  - DOM과 이벤트 제어를 위해 사용되며 클라이언트 자원도 활용할 수 있음
- 객체 지향적 특성을 모두 가지고 있다고 말 할 수 없지만 객체를 정의하여 사용할 수 있는 객체 기반의 언어

JavaScript

자바스크립트 – 사용 목적

- interactive한 홈페이지
- 경제적인 가격의 컴퓨터로 서버 구축
- 플랫폼이 독립적이다.
- 역동적인 홈페이지 제작
- 웹 프로그램 사용 시 반드시 필요하다.

JavaScript

자바스크립트 용도

- 이벤트에 반응하는 동작을 구현
- AJAX를 통하여 전체 페이지를 다시 로드 하지 않고서도 서버로 부터 새로운 페이지 콘텐츠를 받거나 데이터를 제출할 때 사용
- HTML 요소들의 크기나 색상 등 style을 동적으로 변경
  - HTML 내용과 이미지를 변경
- 게임이나 애니메이션과 같은 상호 대화적인 콘텐츠를 구현할 수 있다.
- 사용자가 입력한 값들을 검증

JavaScript

자바스크립트 – 자바와 자바스크립트 차이점

구 분	자바스크립트	자 바
작성/사용 방법	HTML에 작성하며 가공하지 않고 그대로 사용	자바 에디터에서 작성 후 컴파일하며, 자바 가상 머신을 사용하여 읽음
방식	인터프리터 방식	컴파일러 방식
OOP	OOP 기반으로 되어 있지만 클래스가 없고 상속할 수도 없음	완벽한 OOP로 클래스를 가지고 있으며 상속할 수 있음
보안성	소스가 노출되어 보안성이 없음	컴파일된 실행 파일로 만들어져 보안성이 있음

JavaScript

자바스크립트 – 장단점

① 장점
- 컴파일 과정 없이 HTML 파일 내에서 작성할 수 있으므로 개발 속도가 빠르고 또한 다른 언어에 비해 간단하고 배우기 쉽다.
- 웹 브라우저에서 동작하는 스크립트 언어로 시스템에 부담이 적고 HTML과 같이 플랫폼에 독립적이어서 운영체제에 제한을 받지 않는다.

② 단점
- 코드의 보안성
  - 브라우저 상에서 소스 코드가 노출된다.
  - 컴파일하지 않는 언어이므로 복사하여 그대로 사용할 수 있다.
- 내장 함수의 한계: 한정된 객체와 메서드(객체 함수) 제공한다.
- 디버깅 및 개발 도구의 부족

JavaScript

자바스크립트 기본 구조

- <script>와 </script>는 자바스크립트가 시작하고 종료되는 위치를 표시
- 자바스크립트는 이 태그 사이에 위치

```
형식1 :  <script>
             JavaScript_코드
         </script>
```

```
형식2 :  <script src=myscript.js>
             JavaScript_코드
         </script>
```

JavaScript

자바스크립트 출력 방법 – document.write()

- document.write() : '+' 기호로 출력할 값 연결

```
a = 10;  b = 5;
result = a + b;
document.write( a + "  과  " +  b + "를 더한 결과는 " + result );
```

JavaScript

자바스크립트 출력 방법 – alert()

- alert() 내장 함수 사용 – 메시지와 확인 버튼만으로 구성되는 대화상자로서 경고나 인사말 등과 같은 사용자의 요구를 받을 필요가 없는 메시지를 출력하는 경우

```
a = 10;  b = 5;
result = a + b;
alert( a + "  과  " +  b + "를 더한 결과는 " + result );
```

형 식	alert (메시지)
용 도	사용자의 요구를 받을 필요가 없는 정보가 담긴 메지만을 출력시켜 주는 대화상자이다.
인 수	메시지 : 대화상자에 나타나는 메시지이다.
	확인(ok) 버튼을 누를 경우 alert() 함수 명령이 종료한다.

JavaScript

자바스크립트의 위치

- CSS와 마찬가지로 자바스크립트도 다음과 같은 3가지 방법으로 HTML 문서를 삽입
  1. 내부 자바스크립트 (내장형)
  2. 외부 자바스크립트 (링크형)
  3. 인라인 자바스크립트 (행 입력형과 함수형)
- HTML 내부 어디라도 자바스크립트를 삽입이 가능하지만 <head> section에 넣은 것을 권장

JavaScript

자바스크립트의 위치(...Cont'd)

- <HEAD> </HEAD> 태그 사이에 자바스크립트 코드를 넣는 이유
  - 다른 곳에 넣어도 됨
  - <HEAD>부분을 먼저 읽고 <BODY>부분을 읽어 화면에 출력함
  - 미처 해석되지 않은 자바스크립트 코드를 실행하는 일이 발생하지 않도록 하기 위한 것임
  - 웹 페이지가 화면에 나타나기 전에 자바스크립트 코드를 읽어 들여 해석하는 것이 바람직함

JavaScript

자바스크립트의 위치(...Cont'd)

1. 내부 자바스크립트 (내장형)
   - document.write()는 document 객체의 write() 메서드를 호출하는 문장
   - 여기서 객체는 속성과 동작을 한데 모아둔 것으로 메서드는 동작에 해당

```
<!DOCTYPE HTML>
<html>
<head>

   <title>My First Javascript </title>

   <script>
      document.write("Hello World!");
   </script>

</head>
<body></body>
</html>
```

JavaScript

자바스크립트의 위치(...Cont'd)

2. 외부 자바스크립트 (링크형)
   - <head>나 <body>에 놓일 수 있다.

```
<!DOCTYPE html>
<html>
<head>
    <script src="myscript.js"></script>
</head>
<body>
</body>
</html>
```

- myscript.js

```
document.write("Hello World!");
```

JavaScript

자바스크립트의 위치(...Cont'd)

3. 인라인 자바스크립트 (행 입력형과 함수형)
   - 자바스크립트는 HTML 태그 내부에 이벤트 속성으로 삽입
   - 다음은 대표적인 이벤트 처리 코드로 사용자 입력에 반응
     - onclick 이벤트가 발생하면 alert()를 호출

```
<!DOCTYPE html>
<html>
<body>
    <button type="button" onclick="alert('반갑습니다.')">버튼을 누르세요!</button>
</body></html>
```

JavaScript

자바스크립트의 위치(...Cont'd)

- 행 입력형: 문서의 태그 내 사용, 단순한 경우 편리
  - 형식: **<태그 이벤트핸들러="자바스크립트 소스">**
- 함수형: <head> 태그나 <body> 태그 사이
  - 형식:

```
<script>
 function 함수명( )
 {
        자바스크립트 소스
 }
</script>
```

  - 실행:

```
<태그명 이벤트핸들러="함수명()">
```

JavaScript

<script> 태그 밖에서 이밴트 핸들러 사용

- <TAG eventHandler="자바스크립트 명령이나 함수">
  - <INPUT type="button" value="더하기 버튼" onclick="addition()">
    - 사용자가 "더하기 버튼"을 클릭하여 이벤트를 발생시키면 onclick 이벤트 핸들러가 자바스크립트 명령으로 정의된 addition() 함수가 실행된다.
  - <INPUT type="button" value="배경색 변경" onclick="document.bgColor ='blue'">
    - 사용자가 "배경색 변경" 버튼을 클릭하면 onclick 이벤트 핸들러가 자바스크립트 명령인 "document.bgcolor='blue'"를 실행시켜 문서의 배경 색상을 파란색으로 변경시킨다.
  - <body onLoad="window.defaultStatus='어서오세요';">
    - HTML 문서가 처음 실행될 때 자동으로 발생되는 onLoad 이벤트 핸들러가 자바스크립트 명령인 "window.defaultStatus='어서오세요';"를 실행시켜 브라우저 하단의 상태 표시줄에 "어서오세요"와 같은 초기 메시지를 출력한다.

JavaScript

Event Handler 종류

이벤트 핸들러	이벤트 핸들러가 동작되는 경우
onClick	사용자가 하이퍼 링크나 버튼, 체크상자, 리셋(reset)이나 제출(submit) 버튼 등을 클릭할 때 동작한다.
onFocus	사용자가 클릭이나 탭 키를 이용하여 다른 입력 양식 요소로 초점(Focus)을 옮길 때 동작하는 이벤트 핸들러로서, 이는 사용자가 데이터를 입력할 때와는 다른 경우이다.
onBlur	focus 이벤트 핸들러와는 정반대로서 현재 focus 상태에 있는 입력 양식에서 focus를 다른 곳으로 이동할 때 동작한다.
onChange	텍스트(text), 텍스트 영역(text area)의 입력 양식에서 기존 입력되어 있는 자료를 사용자가 변경할 때 동작한다.
onLoad	특정 웹페이지가 접속되어 실행될 때 자동으로 동작한다.
onUnload	접속한 페이지를 떠날 경우에 자동으로 동작한다..
onMouseOver	하이퍼 링크로 마우스 커서를 위치시킨 경우에 동작한다..
onMouseOut	하이퍼 링크로 마우스 커서를 벗어나게 할 때 동작한다..
onSubmit	특정 폼의 자료를 서버로 전송시킬 때 동작한다.

JavaScript

JavaScript 구성요소

- 기본 문법
  - 변수, 자료형, 연산자, 제어문, 함수 등
- 객체
  - 객체는 속성과 메서드를 가짐
  - 객체의 종류 :
    - 기본적으로 제공하는 **내장 객체**
    - 사용자가 정의하여 사용할 수 있는 **사용자 정의 객체**
    - 브라우저 지원을 이용하는 **브라우저 객체**
- 이벤트
  - 이벤트와 이벤트 핸들러를 처리하게 되면 상호작용이 가능한 웹 페이지 작성

JavaScript

자바 스크립트 – 기본 문법

- 주석문 작성방법
- 변수의 사용방법
- 예약어
- 자료형
- 연산자

JavaScript

자바스크립트 주의 사항

- **각 문장 끝에는 ;(세미콜론)을 붙인다.**
  - 세미콜론을 사용해 한 줄에 여러 개의 문장을 사용 가능
- **자바스크립트는 대소문자를 구별한다.**
  - init() 함수와 INIT() 함수와 동일하지 않다.
  - myName 변수와 MyName 변수는 다르다.
  - onclick 속성과 onClick 속성은 다르다.
    - 특히 JavaScript 에서는 camel case 를 종종 소문자로 시작
      - firstName, lastName, masterCard, interCity
- **Hyphens(-)은 JavaScript 에서 허용되지 않는다.**
  - 뺄셈(subtractions)에 할당되어 있다.
- **공백 문자를 모두 무시한다.**
  - 자바스크립트는 프로그램 내 토큰 사이에 존재하는 스페이스, 탭(Tab), 줄바꿈(newline) 등을 무시한다.

JavaScript

자바스크립트 주의사항(…Cont'd)

- **식별자 (identifier)**
  - 식별자는 **변수**나 **함수**의 이름을 작성할 때 사용하는 이름을 의미
    - 자바스크립트 식별자는 **변수**나 **함수**에 이름을 붙이거나 자바스크립트 코드 내 루프 문에 레이블을 붙이는 데 사용된다.
  - 첫번째 문자는 알파벳(letter), 밑줄(_) 혹은 달러 표시($)여야 함
  - 이어지는 문자들은 알파벳(letter), 숫자, 밑줄(_) 혹은 달러 표시여야 한다.
  - 자바스크립트가 숫자와 식별자를 쉽게 구별할 수 있게 하기 위해, 숫자는 첫 번째 문자로 허용되지 않는다.
  - 다음은 모두 올바른 식별자다.
    - **i, my_variable_name, v13, _dummy, $str**
  - 올바른 식별자가 되기 위한 규칙은 자바나 다른 수많은 언어의 규칙과 동일하다.

JavaScript

자바스크립트 – 주석달기

- 한 행을 주석문 처리
  //주석 처리할 행, 문장
- 두 행 이상에 걸치는 주석문 처리
  /* 주석 처리할 영역 */

JavaScript

변수 (Variable)

- 프로그램 실행 시 처리 대상이 되는 자료나 처리된 결과를 기억 시킬 기억 장소의 이름
- 특정 자료형의 값을 가지고 있는 저장 장소
- 선언부와 정의부로 구분되며, var라는 지시어로 선언
  - 예제):

```
var a, b, c;   // 변수 a, b, c 선언
var t = true;  // 변수의 선언과 함께 값을 정의
```

- 자료형에 따라 구분되어 사용되지 않고, 하나의 변수에 다양한 자료의 종류 저장 가능
  - 예제)

```
var a = 100; //정수값 100을 저장하기 위한 기억 공간인 a를 변수로 선언
a = "a는 문자열"; //100을 할당한 후, 다시 변수 a에 문자열 값을 할당 가능
```

JavaScript

자료형 - Data type

- 수치 자료형 – 정수, 실수
- 문자열 자료형 – "문자열"
- boolean 자료형 – true, false
- Null

```
var javaScript          // javaScript 정수형 변수이다.
var pi = 3.14159        // pi는 실수형 변수이다.
var name = "홍길동"     // name은 문자열 변수이다.
var bool = false        // bool은 불리언 변수이다.
var empty = null        // empty는 null 값을 갖는 변수이다.
```

JavaScript

자료형 - Data type

- 자바스크립트는 다른 프로그래밍 언어와 다르게 변수에 자료형을 선언하지 않고, 변수에 자료를 할당하기만 하면 해당하는 4가지 자료형(data type) 중의 하나인 변수로 자동 선언되어 사용된다.

JavaScript

자료형 - Data type

- Numeric (숫자형)
  - 정수형 (Integer):

```
10진수 :    0,   255,   65535
 8진수 :   017,  0377, 0177777
16진수 :   0xf,  0xff,  0Xffff
```

  - 실수형 (Float):

```
고정소수점 수 :  1.414,     3.1415924,    0.00001
부동소수점 수 :  0.1414e01, 0.1414E+10,   5E-5
```

JavaScript

자료형 – Data type (Cont'd)

- String (문자열형) – 텍스트
  - '(quote, 인용기호)나 "(double quote, 이중인용기호) 사이에 표현
  - 문자열 연결을 위한 연산자로 '+' 기호 사용
- Boolean (논리형)
  - 비교식의 결과로 true아니면, false 값임
  - 둘 중 하나를 표현하는 데 사용
- null
  - 자바스크립트에서 널은 아무 것도 없음을 표현하는 용도로 사용
  - 객체가 아님을 뜻하는 특수한 값
- undefined
  - 값이 정해지지 않은 상태를 나타냄 즉, 값이 없음을 나타내는 값
  - 값 자체가 없음 즉, 값이 결정되지 않았다는 것을 나타내는 것으로 변수가 선언되었지만 아직 값이 대입되지 않았으면 undefined 상태가 된다.
  - 초기화 되어 있지 않거나 존재하지 않는 값에 접근 하려 할 때 얻는 변수의 값
  - **시스템 레벨에서는 undefined, 일반적으로 프로그램 레벨에서는 null을 사용**

JavaScript

변수의 사용 범위

- 지역 변수(local variable)
  - 함수의 내부에서 var로 선언한 변수
  - 변수를 선언한 함수의 내부에서만 사용 가능
- 전역 변수(global variable)
  - 함수의 내부에서 선언되지 않은 변수
  - 또는 함수 내부에서 var로 선언하지 않은 변수
  - 페이지의 어디에서나 사용 가능

JavaScript

자료형 – Data type (Cont'd)

- 자바스크립트에서 변수를 선언할 때는 자료형이 필요 없다.
- 하지만 특정한 값이 변수에 저장되는 순간 자료형이 내부적으로 결정된다. (동적 바인딩)
- 자바스크립트에서 변수는 어떤 타입의 값이라도 가질 수 있다.
- 즉 자비스크립트에서는 자료형을 표현하지 않지만 변수의 내부 속성으로 자료형을 가지고 있다.
- 예)

```
var x;          // x의 값은 undefined가 된다.
x = 100;        // x는 숫자를 가진다.
x = "홍길동";   // x는 문자열을 가진다.
```

JavaScript

자료형 – Data type (Cont'd)

- object(객체)형
  - **객체(object)**는 사물의 속성과 동작을 묶어서 표현하는 기법
  - 예) 자동차는 메이커, 모델, 색상, 마력과 같은 속성도 있고 출발하기, 정지하기 등의 동작도 가지고 있다.

```
var myCar = { model: "bmz", color: "red", hp: 100 };

document.write(myCar.model + "<br>");
document.write(myCar.color + "<br>");
document.write(myCar.hp + "<br>");
```

JavaScript

예약어

- 특별한 용도로 미리 지정한 단어

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

JavaScript

자료형(데이터 형) 변환

- 예)

```
i = 10;        //수치형 (number)
i = "ten";     //문자열 (string)
i = "10";      //문자열 (string)
```

- 데이터 형을 규정해 놓지 않은 언어
  - 데이터 형이 엄격하지 않고 동적으로 변하는 언어
  - 어떤 형식의 데이터라도 가질 수 있다.
  - 부족한 데이터형을 보완하기 위해서 필요에 따라 데이터 형을 자동 변환: 자동으로 형 변환
  - 스크립트 언어에 합당한 유연성과 단순성을 제공

JavaScript

자료형(데이터 형) 변환 함수

- parseInt()
  - 문자열을 숫자로 변환할 때 사용하는 함수
- String()
  - 숫자를 문자열로 변환할 때 사용하는 함수

JavaScript

자료형(데이터 형) 변환 함수

- 내장함수 prompt() 사용(1)
  - 사용자가 입력한 내용을 문자열로 반환
- 내장함수 prompt() 사용(2)
  - '확인'을 선택하면 입력한 값이 변수에 저장
  - '취소'를 선택하면 null값이 변수에 저장

js_prompt.html

```
<script>
  var age = prompt("나이를 입력하세요", "만 나이로 입력합니다.");
</script>
```

JavaScript

연산자 (Operator)

- 산술 연산자 (Arithmetic Operator): +, −, *, /, %, ++, −−
- 비교/관계 연산자 (Relational Operator): ==, !=, >, <, >=, <=
- 논리 연산자 (Logical Operator): &&, ||, !
- 비트 연산자 (Bitwise Operator): &, |, ^, ~, <<, >>, >>>
- 배정 연산자 (Assignment Operator): =, +=, −=, *=, /=, %=, <<=, =>>, >>>=, ^=
- 조건 연산자 (Conditional Operator): (조건) ? 결과 1 : 결과 2
- 문자열 (연결) 연산자 : +, +=
  - "Java" + "Script"

JavaScript

연산자 (Operator) – 비교/관계 연산자

- 값의 대소를 비교

비교연산자	의미
a == b	a와 b는 같다
a != b	a와 b는 같지 않다
a < b	a는 b보다 작다
a <= b	a는 b보다 작거나 같다
a > b	a는 b보다 크다
a >= b	a는 b보다 크거나 같다

JavaScript

연산자 (Operator) – 논리 연산자

- AND, OR, NOT 등 논리연산을 수행하는 연산자

논리요소		NOT	AND	OR
A	B	!A	A && B	A || B
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

JavaScript

연산자 (Operator) – 비트 연산자

[표 3.10] 비트 연산자

연산자	연산 의미	연산 예
&	비트 논리곱	x & y
|	비트 논리합	x | y
^	비트 배타적 논리합	x ^ y
~	1의 보수	~x
<<	왼쪽 이동	x << y
>>	오른쪽 이동	x >> y
>>>	오른쪽 이동-왼쪽 비트 0	x >>> y

JavaScript

연산자 (Operator) – 배정 연산자

[표 3.11] 배정 연산자

연산자	연산 예	의미
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y
&=	x &= y	x = x & y
|=	x |= y	x = x | y
^=	x ^= y	x = x ^ y

JavaScript

연산기호의 우선순위

우선순위	연산자
1	[], ()
2	++, --, !
3	*, /, %
4	+, -
5	<, <=, >, >=
6	==, !=
7	&
8	|
9	&&
10	||

JavaScript

문장 (Statement) - 기본 실행문

- 변수 선언문: var i = 10
- 대입/배정문: i = 10, i = "SeoulTech"
- 조건문:

```
if (i < 10)
  document.write("조건만족")
```

- 순환문 (반복문):

```
for (var i = 0; i < 10; i++){
        document.write(i)
}
```

JavaScript

자바스크립트 - 문장 (Statement) 종류

- 자바스크립트 문장
  - 배정문 : var = exp
  - 혼합문 : { }
  - 제어문
    - 조건문 : if 문, switch 문
    - 반복문 : for 문, while 문, do-while 문
    - 분기문 : break 문, continue 문, return 문
    - 객체 제어문 : for in 문, with 문

JavaScript

문장 (Statement) - 배정문

- 프로그래밍언어의 가장 기본적인 문장
- 새롭게 계산되어진 값을 변수에 저장하는 기능
- = 기호를 중심으로 좌측에는 변수, 우측에는 식으로 구성
  - 의미: 먼저 식이 연산된 후, 그 결과가 좌측의 변수에 저장
- 예제)

```
remainder = dividend - (dividend / divisor) * divisor;
sum += i;          // sum = sum + i
i = j = 0;
```

JavaScript

문장 (Statement) - 혼합문

- 여러 문장을 한데 묶어 하나의 문장으로 나타내는 역할
- { } 기호로 표시하며, 그 사이에 <선언>과 <문장> 기술
- 예제)

```
if (a > b) {
    a--;
    b++;
}
```

JavaScript

문장 (Statement) - 제어문

- **프로그램의 실행 순서를 바꾸는데 사용되는 문장**
- **종류**
  - **조건문**
  - **반복문**
  - **분기문**
  - **객체 제어문**

JavaScript

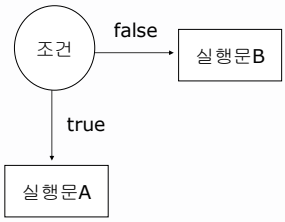
문장 (Statement) - 조건문

- 주어진 조건에 따라 수행되는 부분이 다를 때 사용되는 문장
- 종류
  - if문
  - switch문

JavaScript

문장 – IF 조건문 (1)

```
if (조건)
  실행문A
else
  실행문B
```



JavaScript

문장 – IF 조건문 (2)

- 형식1

```
if(조건)
    명령문
```

- 형식2

```
if(조건){
    명령문1
    명령문2
}
else{
    명령문1
    명령문2
}
```

- 중첩 IF문

```
if(조건)
    명령문
else if(조건)
    명령문
else if(조건)
    명령문
else(조건)
    명령문
```

JavaScript

조건문 – SWITCH 문

- switch문은 여러 가지 경우 중 하나를 선택하는데 적합한 문장 구조
- switch문의 조건은 if문과 같은 조건문이 아니라 특정 값으로 결정되는 식이 온다.
- 필요한 만큼 case문이 올 수 있다.
- default는 식에 해당하는 값이 없을 경우에 사용된다.
- default는 없어도 된다.
- break문이 없으면 다음 case도 실행된다.

JavaScript

조건문 – SWITCH 문

```
switch(표현식){
    case value1:
            명령문1;
            break;
    case value2:
            명령문;
            break;
    ……
    default
            명령문n
}
```

JavaScript

문장 – 반복 문

- 프로그램의 일정한 부분을 주어진 조건이 만족할 때까지 반복해서 실행하는 문장

- 종류
  - for문
  - while문
  - do~while문

JavaScript

반복문 – FOR 문 (1)

- 초기값과 조건식, 증감이 한 문장에 같이 온다.
- 조건이 참인 동안 반복 수행
- 기본 형식

```
for(초기 값;조건부;증감식){
    코드부 문장
}
```

JavaScript

반복문 – FOR 문 (2)

- 예제)

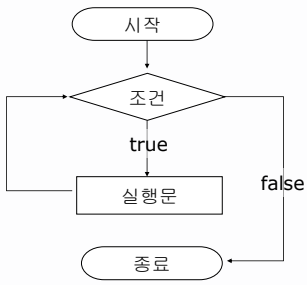
```
for(a=1;a<11;a++){
    document.write(a+ "*" +a+ "="+a*a+"<br>")
}
```

```
sum = 0;
for(a = 1;a <=10; a++)
    sum = sum + a;
document.write("1부터 10까지의 합은
                " + sum + "이다.");
```

```
a = 1; sum = 0;
while(a <=10) {
    sum = sum + a;
    a++;
}
```

JavaScript

반복문 – FOR 문 (3)



JavaScript

FOR 문 – 중첩 반복문

- 반복문 안에 다시 반복문을 사용

```
for( ;  ; ){
    for( ; ; ){
       문장1;
    }
    문장2;
}
```

```
<script language="JavaScript">
   for(a = 0; a <= 3; a++) {
      document.write("a  = " + a + "<br>");
      for(b = 0; b <=5; b++)
         document.write("b = " + b + " ");
      document.write("<br>");
   }
</script>
```

JavaScript

반복문 – WHILE 문 (1)

- 주어진 조건식이 참인 경우에만 프로그램의 일정한 부분을 반복해서 실행하는 반복문
- 기본 형식

```
while(조건){
    명령문
}
```

- FOR 문과의 비교:

◑ for 문 사용 형태 :

```
for (<초기식>;<조건식>;<증가식> )
   <반복할 문장>
```

◑ while 문 사용 형태 :

```
<초기식>;
while ( <조건식> ) {
   <반복할 문장>
   <증가식>
}
```

JavaScript

반복문 – WHILE 문 (2)

- 예제 1)

```
a=1
while(a<11){
    document.write(a+ "*" +a+ "="+a*a+"<br>")
    A++
}
```

- 예제 2)

```
a = 1; sum = 0;
while(a <=10) {
    sum = sum + a;
    a++;
}
document.write("1부터 10까지의 합은 " + sum + "이다.");
```

JavaScript

반복문 – DO WHILE 문

- 반복되는 문장을 먼저 실행한 후에 조건식을 검사하여 참이면 계속 반복하고, 그렇지 않으면 반복을 종료하고 제어 이동
- 조건이 참인 동안 반복 수행 : 조건을 나중에 확인
- 조건은 참,거짓을 판별할 수 있는 조건식이 온다.
- 조건을 나중에 확인하므로 문장을 한번은 수행한다.
- 기본 형식

```
do{
    명령문
} While(조건)
```

```
a = 0; sum = 0;
while(a > 1) {
    sum = sum + a;
    a++;
}
```

```
a = 0; sum = 0;
do {
    sum = sum + a;
    a++;
} while(a > 1);
```

JavaScript

문장 – 분기문

- 지정된 곳으로 제어를 옮기는 문장
  - **break문, continue문, return문**
  - **위 3가지 문은 줄 바꿈을 하지 말아야 하며, 줄 바꿈을 하면 자동으로 ; 이 적용 되어 원치 않는 예외 발생**
- break문: 제어문 종료
  - while문이나 for문, switch문에서 현재의 실행을 멈추고 제어를 블록 밖으로 옮기는 역할
- continue문: 제어문 반복
  - 반복문 내에서 사용하며, 다음 반복이 시작되는 곳으로 제어를 이동하는 기능
- return문
  - 함수의 실행을 종료하고 호출한 함수에게 제어를 넘겨주는 문장

JavaScript

문장 - 객체 제어문

- **객체를 조작하는 문장을 제공**
- **다른 언어들 즉, C, C++, Java에서는 제공하지 않는 특수한 문장**
- **종류:**
  - **FOR ~ IN**
  - **WITH**

JavaScript

객체 제어문 – FOR ~ IN

- 객체의 모든 속성에 대해서 순서적으로 조회
  - 객체가 가지는 속성 정보를 알려준다.
- 객체의 모든 속성을 차례로 처리하고자 할 때 사용
  - 만약 객체의 모든 속성이 5개라면 5번 반복된다.
- 자바 스크립트는 완성된 언어가 아니므로 버전업 되면서 새로 추가된 객체의 속성 정보를 알 수 있다.
- 형식:

```
for (variable in 객체) //변수는 객체의 모든 속성 참조
{
  수행할 작업
}
```

JavaScript

객체 제어문 – WITH

- 하나의 객체에 대해 여러 가지 속성들을 한꺼번에 조작할 때 사용한다.
- 객체의 이름은 생략하고 매소드와 속성만 사용함으로서, 반복적인 객체이름에 대한 번거로움을 덜 수 있다.
- 형식:

```
with (객체)
{
  조작 내용
}
```

```
with (document)
{
        bgcolor = "white";
        fgcolor = "red";
}
```

JavaScript

자바 스크립트 – 함수

JavaScript

함수 (Function)

- 함수란?
  - 프로그램에서 특정한 기능을 수행하도록 분리하여 구현된 코드
  - 특정 작업/일을 수행하는 명령어들의 집합
- 하나의 module/block으로 실행될 수 있는 문장들
  - 함수(function)란 호출에 의해서 실행되거나, 마우스 클릭 등의 이벤트에 의해서 실행시킬 수 있도록 만든 코드 블록
- 독립적으로 처리할 수 있는 작업을 기술하는 프로그램 단위
- 메서드나 사용자 정의 객체를 정의할 때 이용

JavaScript

함수 (Function) 종류

- **사용자 정의 함수 (User-defined Function)**
  - 사용자가 필요에 따라 만들어 사용하는 함수
- **내장 함수 (Built-in Function)**
  - 공통으로 사용하는 기능을 미리 만들어서 제공되는 함수

JavaScript

함수 - 사용자 정의 함수

- 함수의 형태 :

```
function functionName(argList) {
    statements; // function body
}
```

  - 예약어 function으로 시작
  - 매개변수의 type을 기술하지 않음
  - return type을 명시하지 않음
- 정의는 head부문에
- 호출은 body내 스크립트 태그 안에서 직접 호출하거나, 페이지 내 이벤트 (event)에 의해 호출

JavaScript

함수 - 사용자 정의 함수의 종류

- 매개변수가 없는 함수

```
function test() {
...
}
```

- 매개변수가 있는 함수

```
function test(name) {
...
}
```

- 리턴 값이 있는 경우

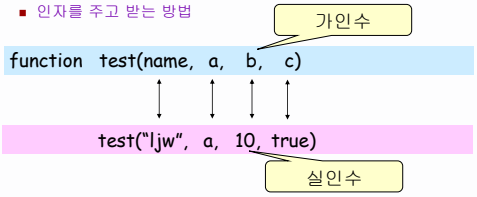
```
function test(question) {
    ans = confirm(question)
    return ans
}
```

- 리턴 값이 없는 경우

JavaScript

매개 변수가 있는 함수

- 인자 전달이 필요한 함수
  - 함수를 호출하는 곳과 해당 함수와의 자료교환을 위하여 인자(argument,인수)를 사용
  - 인자를 통한 자료 교환을 매개변수 전달(parameter passing)이라고 함
  - 인자를 주고 받는 방법



JavaScript

매개 변수가 있는 함수 (Cont'd)

- 인자 전달 방법
  - 값에 의한 호출(call by value)
    - 실인수의 값을 가인수로 전달
    - 전달받은 값을 변경하여도 실인수에는 영향이 없음
  - 주소에 의한 호출(call by reference)
    - 실인수의 주소를 가인수로 전달
    - 전달받은 가인수를 변경하면 실인수도 변경됨
  - 자바스크립트는 값에 의한 호출이 기본
    - 객체 처리시 주소에 의한 호출 사용

JavaScript

함수 - 내장 함수 (built-in function)

- 미리 정의되어 있는 global function/독립적인 함수
- 객체의 참조 없이 어느 곳에서든지 임의로 호출하여 사용할 수 있음
- 내장 함수의 종류:
  - parseInt(): 문자열을 정수로 변환
  - parseFloat(): 문자열을 부동 소수점으로 변환
  - eval(): 문자열을 숫자로 변환한 후 그 결과 값을 리턴
  - escape(): ISO LATIN-1 문자 ===> %16진수의 ASCII 코드 값으로 변환
    - 예) escape("&") ===> %26 = $(26)_{16}$
  - unescape(): ASCII 코드 값 ===> ISO LATIN-1 문자로 변환
    - 예) unesacpe("%26") ===> &

JavaScript

함수 – 내장 함수의 종류

- 대표적인 내장 함수

대화상자기능	수식계산	문자열과 숫자	시간조정
alert() prompt() confirm()	eval()	parseInt() parseFloat() isNaN()	setTimeout() clearTimeout() setInterval()

JavaScript

함수 – 내장 함수 confirm()

- confirm() : 질문에 대한 응답
  - '확인'을 선택하면 true, '취소'를 선택하면 false가 변수에 저장된다.
- confirm() 내장 함수는 사용자가 에게 질문 메시지를 보여 주고 사용자가 선택한 확인 또는 취소 버튼에 따라 프로그램을 진행해 나갈 때 사용
- 사용자가 확인(ok) 버튼을 누를 경우에 ok_no 변수에 true 값이 할당되고, 만일 취소(cancel) 버튼을 누를 경우에는 false 값이 할당된다.

형식	confirm(메시지)
용도	사용자에게 질문 메시지를 보여주고 응답을 확인하고 싶을 때 사용하는 대화상자이다.
인수	메시지 : 대화상자에 나타나는 메시지이다.
반환값 (return value)	· 확인(ok) 버튼을 누를 경우 : true 값이 반환되어 조건으로 사용하거나 변수에 할당할 수 있다. · 취소(cancel) 버튼을 누를 경우 : false 값이 반환되어 조건으로 사용하거나 변수에 할당할 수 있다..

JavaScript

함수 – 내장 함수 eval()

- 수식을 문자열 인자로 받아 이를 수식으로 변환하여 계산을 수행하고 결과를 돌려준다.
- 문자열로 입력된 수식을 계산하여 주는 편리한 함수
- document.write(eval(10 * 5)); // 50이 출력된다.
  eval() 함수는 prompt() 함수와 같은 입력 대화상자를 통해 입력 받은 수식을 처리할 때 아주 편리하다.

형식	eval(문자열 수식)
용도	문자열 수식을 인수로 기술하면 문자열을 수식으로 변환한 후에 수식 계산을 수행하여 준다.
인수	문자열 수식: 수식 계산에 사용할 문자열을 상수 또는 변수로 기술한다.

JavaScript

함수 – 내장 함수 isNaN()

- isNaN() : 인자가 숫자인지 판단(is Not a Number)
  - isNaN(= is Not a Number) 함수는 문자인지 숫자인지를 구별하는 함수로서 테스트 값이 문자이면 "true"를 숫자이면 "false"로 결과를 나타낸다.
  - 문자일 경우 true, 숫자일 경우 false를 반환한다.
  - a는 false, b는 true를 반환한다.

```
a = isNaN("100");
b = isNaN("12abc");
```

JavaScript

함수 – 내장 함수 isNaN()

```
// "100"은 false로 출력되어 숫자임을 알 수 있다.
num = isNaN("100");
document.write ( num ); // false로 출력된다.

// "string"은 true가 출력되어 문자임을 알 수 있다.
document.write ( isNaN("string") );

// "10kkk"와 같이 숫자와 문자가 혼합되어 있으면 true가 출력되어 문자로 판단한다.
document.write ( isNaN("10kkk") );
```

JavaScript

함수 – 내장 함수 parseInt() & parseFloat()

- parseInt() :문자열을 정수로 변환
  a = parseInt("10");
  a = parseInt("10", 2); //진수 표시:2, 8, 10, 16; 생략하면 10진수
  a = parseInt("123ab45");// 문자 이후 부분은 무시; 결과는 123
- parseFloat() : 문자열을 실수로 변환
  a = parseFloat("3.14");

JavaScript

시간 조정 내장 함수 종류

- setTimeout()
  - 일정시간이 경과 후 명령 실행
  - a = setTimeout("alert('시간경과')", 2000);
  - 2000ms(2초)가 지난 후 alert() 실행
- setInterval()
  - 일정 시간 간격마다 명령 반복 실행
  - a = setInterval("window.status=new Date()", 1000);
    - 1초마다 status 라인에 시간 표시
- clearTimeout()
  - setTimeout(), setInterval()로 설정한 시간 명령을 해제
  - clearTimeout(a);

JavaScript

자바 스크립트 – 객체

- 객체의 개념
- 객체의 이용방법
- 객체의 구조
- 객체의 사용
- 자바스크립트 내장객체

JavaScript

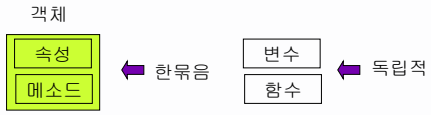
객체(object) 란?

- 사전적 의미
  - 실생활에서 식별할 수 있는 사물(책상,의자..)
  - 생각이나 관념
- 프로그래밍에서는
  - 프로그래밍에 필요한 요소를 미리 만들어 제공
  - 버튼, 스크롤바, 윈도우 등
  - 미리 만들어 제공되는 객체를 조립하여 프로그래밍
  - 데이터 + 함수

JavaScript

객체(object)의 구조

- 속성(attribute) + 메소드(method)
  - 데이터 + 함수
- 속성 : 객체의 특성을 나타내는 데이터
- 메소드 : 객체의 동작을 나타내는 함수



JavaScript

객체 표현법/사용법

- 객체명.속성="값"
  window.status="GO!"
  document.bgColor
- 객체명.메소드
  window.close( )
  human.go()
- 상위객체이름.하위객체이름.속성="값"
  window.document.frm1.id.value="컴공과"

JavaScript

객체 (Object) 의 종류

- 사용자 정의 객체 (User-defined Object)
  - 자바 스크립트 작성자가 직접 정의한 객체
- 자바스크립트 내장 객체 (Built-in Object)
  - 자바 스크립트 언어 자체 내에서 지원되는 객체
- 브라우저 내장객체: 표준 객체 (Standard Object)
  - 브라우저의 정보를 관리 및 처리할 수 있는 객체

JavaScript

사용자 정의 객체

- 예)
```
function Book(title, author, pages) {
        this.title = title;
        this.author = author;
        this.pages = pages;
}
        ~~
myBook = new Book("JavaScript", "John", 255);
myBook.author = "Jane"; // author 멤버변수에 "Jane" 입력
```
  - title, author, pages --- 객체의 속성 변수로 정의
  - this --- 객체 자신을 가리킴
  - new --- 인스턴스 생성 연산자 (객체 생성)
  - myBook --- 변수로 객체 생성
- 객체는 생성자 (constructor) 함수를 통해 정의
- 함수의 이름이 객체의 이름

JavaScript

자바스크립트 내장 객체

객체	용도
Date	날짜와 시간용 객체
String	문자열 처리용 객체
Math	일반 수학 연산에 사용하는 객체
Array	배열 처리에 사용하는 객체
Arguments	함수의 인자(arguments) 정보처리용 객체
Function	함수정의용 객체
Number	수치 상수에 대한 객체
screen	화면에 대한 정보를 처리하는 객체

JavaScript

내장 객체 (Built-in Object)

◈ 내장 객체
- 객체를 정의하지 않고 사용할 수 있는 자바스크립트의 객체
  - 자바스크립트에 미리 정의되어 있는 객체
  - Java에서 미리 정의되어진 클래스와 유사
- 사용 빈도가 높은 많은 메소드들을 포함 즉 내장 함수
- 내장 객체의 종류
  - Array, Boolean, Date, Function, Math, Number, Object, RegExp, String, Arguments, etc

JavaScript

내장 객체의 사용

❖ 인스턴스 생성
- 인스턴스 : 객체정의로부터 생성된 객체
- new 연산자를 통하여 객체의 인스턴스를 생성

  인스턴스명 = new 생성자함수()
- 예)

  today = new Date();
  - Date 객체로 부터 생성된 인스턴스 today
- new 연산자에 의해 내장 객체를 사용하여 인스턴스를 생성

  someString = "JavaScript";

  myArray = new Array("hello", someString, 3.14);

JavaScript

내장 객체의 사용(Cont'd)

❖ 정적객체 사용
- new연산자와 생성자함수 없이 객체 사용
- 프로그램 전역에서 사용하는 객체

❖ 예
- Math.PI
- Math.sin(30)
- document.write("출력")

JavaScript

내장 객체 - 배열 (array)과 Array 객체

- 같은 형, 같은 길이의 데이터를 2개 이상 붙여서 동일한 변수로 처리하는 것
  - 같은 성격의 많은 자료를 동일한 이름의 변수명에 저장하고 사용하기 위한 기법
  - 40명의 성적을 처리한다면 40개의 변수가 필요한데 이를 성적처리를 위한 하나의 변수로 사용
- 동일한 자료 객체 여러 개가 모인 구조체: 각 자료 객체를 배열의 원소(element)라 한다.
- 자바스크립트에서는 배열을 객체로 취급
- 명시적인 배열이 아닌 객체의 속성을 이용한 연관 배열을 지원
  - 연관배열 (associative array): 배열의 인덱스로 속성의 이름을 사용하여 특정원소를 참조하는 형태
- 정의:
  - 사용자가 배열을 정의 하여 사용
  - 내장객체의 Array를 이용하여 사용

JavaScript

내장 객체 - 배열 변수 선언법

- 예)

```
function MakeArray(n) {
   this.length = n;
   for (var i = 1; i <= n; i++) {
       this[i] = 0
   }
   return this
}
object1 = new MakeArray(3);  //배열 객체를 생성
```

- 사용 방법:
  - 사용자 정의:
    `objectname = new MakeArray(n);`
  - Array 객체 사용:
    `objectname = new Array(n);`

JavaScript

내장 객체 – Array 객체 사용법

- 예 1) 배열의 크기를 정하지 않은 경우

```
var 배열명 = new Array( )
배열명[0]=값
배열명[1]=값
배열명[2]=값
```

- 예 2) 배열의 길이인 인덱스 첨자를 정한 경우

```
var 배열명 = new Array(10)
```

- 예 3) 각 셀의 초기값과 함께 정의 할 경우

```
var 배열명 = new Array(값1, 값2, 값3)
```

```
someString = "JavaScript";
myArray = new Array("hello", someString, 3.14);
```

JavaScript

내장 객체 – Array 객체 사용법

- 크기 지정으로 생성
  - a = new Array(크기)
  - a = new Array(5) -> 5개의 값을 저장 가능하고 첨자(index)는 0로 부터 시작
- 초기값 지정으로 배열 생성
  - week = new Array (“일”,”월”,”화”,”수”,”목”,”금”,”토”)
  - a = new Array (10,20,30,40,50)

JavaScript

내장 객체 - Date 객체를 생성하는 다양한 방법

- now=new Date( )
  - 현재 날짜와 시각을 갖는 now 객체를 생성한다.
- now=new Date(year, month, day)
  - 연,월,일 정보를 갖는 now 객체를 생성한다.
- now=new Date(year, month, day, hours, minutes, seconds)
  - 연, 월, 일, 시, 분, 초의 정보를 갖는 now 객체를 생성한다.
- now=new Date("month, day, year hours: minutes: seconds")
  - 월, 일, 연 시:분:초의 정보를 갖는 now 객체를 생성한다.

JavaScript

Date 객체(1)

메소드	기능
getYear()	Date객체에서 연도 추출
getMonth()	월 추출(0 ~ 11)
getDate()	일 추출(1~31)
getDay()	요일 추출(0~6)
getHours()	시간 추출(0~23)
getMinutes()	분 추출(0~59)
getSeconds()	초 추출(0~59)
getTime()	1970.1.1을 기준으로 ms단위로 추출

JavaScript

Date 객체(2)

메소드	기능
setYear(연도)	연도 설정
setMonth(월)	월 설정(0 ~ 11)
setDate(일)	일 설정(1~31)
setDay(요일)	요일 설정(0~6)
setHours(시간)	시간 설정(0~23)
setMinutes(분)	분 설정(0~59)
setSeconds(초)	초 설정(0~59)
setTime(ms)	1970.1.1을 기준으로 ms단위로 설정

JavaScript

Date 객체 – 날짜와 시간을 표시하는 메소드

메소드명	설명
getYear( )	1970년 이후의 년도를 구함
getFullYear( )	년도를 구함
getMonth( )	월을 구함(0~11, 1월은0, 12월은11)
getDate( )	일을 구함(1~31)
getDay( )	요일에 해당하는 숫자를 구함(0~6, 일요일은0, 토요일은6)
getHours( )	시간을 구함(24시간 기준, 0~23)
getMinutes( )	분을 구함 (0~59)
getSeconds( )	초를 구함(0~59)
GetTime( )	1970년 1월 1일 00:00:00부터 지정한 시간까지를 1/1000초로 표시
GetMilliseconds( )	1/100초로 표시

JavaScript

문자열 객체 - String

- String 객체 생성 방법
  - new 연산자 사용
    - str = new String("문자열");
  - 간편한 방법
    - str = "문자열";

JavaScript

문자열 객체 – String(Cont'd)

속성	기능
length	문자열의 크기를 알려준다.

메소드	기능	HTML 태그
big()	글자를 크게	<BIG>글자</BIG>
small()	글자를 작게	<SMALL>글자SMALL>
blink()	깜박임	<BLINK> 글자 </BLINK>
bold()	볼드체	<B> 글자 </B>
fixed()	타자기체	<TT> 글자 </TT>
italics()	이탤릭체	<I> 글자 </I>
strike()	글자에 줄 긋기	<STRIKE> 글자 </STRIKE>
sub()	아래 첨자	_{글자}
sup()	위 첨자	^{글자}
fontcolor("색")	글자 색	<FONT COLOR="색">글자</FONT>
fontsize("크기")	글자 크기	<FONT SIZE="크기">글자</FONT>

JavaScript

문자열 객체 – String(Cont'd)

메소드	기능
charAt(위치)	위치로 지정한 문자 추출(0부터)
indexOf("문자열")	지정된 문자열의 위치(왼쪽으로부터)
lastIndexOf("문자열")	지정된 문자열의 위치(오른쪽으로부터)
substring(시작,마지막)	시작부터 마지막위치까지 부분 문자열 추출
substr(시작,길이)	시작부터 길이만큼 부분 문자열 추출
toLowerCase()	문자열을 소문자로 변환
toUpperCase()	문자열을 대문자로 변환
concat("문자열")	두 개의 문자열을 하나로 병합
split(분리문자)	분리문자를 기준으로 문자열 분할하여 배열화
split(분리문자,범위)	범위만큼만 분할하여 배열화

JavaScript

내장 객체의 사용 - Math

◈ 수학 Math 객체

- 수학 연산에 사용하는 객체
- 정적 객체로 new 없이 사용

```
// Math 객체의 수학 함수
with (Math)  {
    document.write("abs(-3.14) is " + abs(-3.14) + "<BR>");
    document.write("sin(0.5) is " + sin(0.5) + "<BR>");
    document.write("Random number is " + random() + "<BR>");
}
```

속성	기능
PI	원주율(3.14159…)

JavaScript

내장 객체의 사용 – Math(Cont'd)

메소드	기능
sin(x)	sine 함수
cos(x)	cosine 함수
tan(x)	tangent 함수
abs(x)	절대값
random()	난수발생(0 < 난수 < 1)
max(a,b)	a,b중 큰 값 리턴
min(a,b)	a,b중 작은 값 리턴
round(x)	소수부분이 0.5보다 크면 반올림
floor(x)	소수이하를 버리고 가장 가까운 정수로 변환(6.7->6, -6.7->-7)
ceil(x)	무조건 소수 이하를 반올림(6.1->7, -6.1->-6)

JavaScript

내장 객체 – Function 객체

- 한 줄 정도의 간단한 함수를 생성

변수 = new Function([인자1][,인자2]..., 함수 몸체)

add = new Function("a", "b", "return a+b")

```
function add(a, b) {
   return a+b;
}
```

JavaScript

내장 객체 – Function 객체

속성	기능
arguments	함수에 전달된 인자를 가지고 있는 배열
arguments.length	함수로 실제 전달된 인자의 개수
length	함수에 정의된 인자의 수

JavaScript

내장 객체의 사용 – Boolean, Date, Function

◈ **Boolean 객체**

```
// Booloean 객체를 사용해 참/거짓의 할당
isTrue = new Boolean(false);
document.write(isTrue + "<BR>");
```

◈ **Date 객체**

```
// Date 객체의 getYear와 getMonth 메소드의 사용
today = new Date("Oct 28, 1999 12:00:00");
document.write(today.getYear(), ", ", today.getMonth(), "<BR>");
```

◈ **Function 객체**

```
// Function 객체를 사용해 새로운 함수의 생성
var println = new Function("str", "document.write(str + '<BR>')");
println("This is from println function.");
```

JavaScript

내장 객체의 사용 - Number

◈ **Number 객체**

```
// Number 객체를 사용해 숫자에 설명을 추가
myNum = new Number(3342);
myNum.description="Phone number : ";
document.write(myNum.description + myNum + "<BR>");
```

JavaScript

내장 객체의 사용 – object, RegExp

◈ **Object 객체: 모든 자바스크립트 객체의 특징을 포함**

```
document.write("(Decimal : Binary) ==> ");
for (x = 0; x < 5; x++) {
    document.write("(", x.toString(10), " : ", x.toString(2), ")");
    if (x!=4) document.write(", ");
}
document.write("<BR>");
```

◈ **RegExp 객체: 문자열을 검색해 주는 객체**

```
// RegExp 객체를 이용해 특정 문자 패턴을 검색.
document.write(/ab+c/.exec("cabbcbsbz") + " is same as ");
myRe= new RegExp ("ab+c");
token = myRe.exec("cabbcbsbz");
document.write(token + "<BR>");
```

JavaScript

브라우저 내장 객체 - 표준 객체 (Standard Object)

◈ **표준 객체 – 브라우저 내장객체**

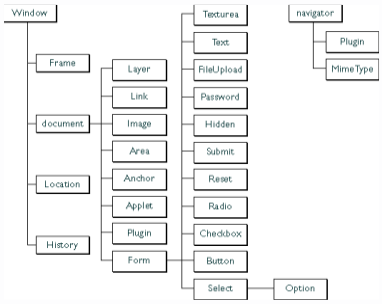
- HTML 문서가 브라우저에 로드 되면 브라우저는 HTML 문서의 <FORM> 태그나 <A>(anchor) 태그 등 관련된 정보에 기초하여 많은 자바스크립트 객체를 생성하는 것을 말한다.
- 넷스케이프 네비게이터를 동적으로 제어할 수 있다.
- 계층적 구조를 갖고 있다.

◈ **계층적 구조**

- 하위 객체는 상위 객체의 속성으로 정의되어 있다.
- 예를 들어, form1이라는 이름의 객체는 document 객체의 속성인 form 객체의 인스턴스이며, 자바스크립트에서 document.form1 형태로 참조할 수 있다.
- 네비게이터 객체의 모든 하위 객체는 많은 속성과 메소드 그리고 이벤트 핸들러를 갖고 있다.

JavaScript

브라우저 내장 객체 - 계층적 구조



JavaScript

Browser 내장 객체 – Cont'd

- 브라우저 내장 객체가 수행하는 기능

▪ navigator 객체	: 사용하고 있는 브라우저에 대한 정보를 제공한다.
▪ window, frame 객체	: 브라우저 윈도우의 상태를 제어한다.
▪ document 객체	: 문서 자체를 다루는 객체이다.
▪ form 객체	: 사용자의 입력 폼을 제어한다.
▪ location 객체	: 현재 문서의 URL에 기반한 정보를 가지는 객체이다.
▪ history 객체	: 브라우저가 이미 방문했던 사이트들에 대한 URL 값을 갖는다.

JavaScript

HTML DOM 객체

- HTML DOM 객체
  - HTML 문서의 요소(element)에 접근하거나 조작하기 위해 정의된 객체

이름	설명
Document	문서 객체 처리
Elements	문서의 요소 처리
Events	문서의 요소에 이벤트 처리기 등록
Body	body 요소 처리
Image	이미지 요소 처리
Table	테이블 요소 처리
Form	폼 요소 처리
Frame/IFrame	프레임 요소 처리
Frameset	프레임 셋 요소 처리

이름	설명
Input Text	텍스트 입력 필드 처리
Input Button	입력 버튼 처리
Input Checkbox	체크박스 처리
Input Password	패스워드 필드 처리
Input Radio	라디오 버튼 처리
Button	버튼 요소 처리
Textarea	텍스트 영역 요소 처리
Anchor	앵커 객체 처리
Link	링크 요소 처리

JavaScript

Browser 내장 객체 – document 객체

- 가장 많이 사용하는 객체
- HTML 문서에 대한 정보와 문서에 포함된 객체에 대한 정보를 보유
- 속성
  - 웹 문서의 색상 설정과 관련된 속성
    fgColor, bgColor, alinkColor, linkColor, vlinkColor
  - 웹 문서와 관련된 정보를 다루는 속성
    lastModified, location, referrer, title
  - 웹 문서에 포함된 내용과 관련된 속성
    anchors, cookie, forms, links
- 메소드
  - document 객체의 메소드
    open(), close(), clear(), write(), writeln()

JavaScript

Browser 내장 객체 – document 객체

- document 객체의 주요 속성과 메소드

구분	이름	설명
속성	title	문서의 제목
	lastModified	문서를 마지막으로 수정한 날짜와 시각
메소드	getElementsByName("이름")	지정된 이름의 모든 요소들 접근
	getElementsById("ID")	지정된 ID의 처음 요소에 접근
	write("문자열")	문자열 출력
	writeln("문자열")	문자열 출력과 줄바꿈

JavaScript

자바 스크립트 – 이벤트

JavaScript

이벤트와 이벤트 핸들러

- JavaScript
  - 이벤트(event)에 의해 실행되는 언어
  - 이벤트 처리기
    - 이벤트가 발생할 경우 이를 처리
    - 이벤트의 이름 앞에 'on'이라는 접두어를 붙여서 사용
- 이벤트 & 이벤트 핸들러
- 이벤트 핸들러의 종류

JavaScript

이벤트와 이벤트 핸들러

- 마우스의 동작과 관련된 이벤트 처리기를 이용한 예
  - 마우스 이벤트 처리기의 기능(1)

```
onMouseOver
: 특정한 요소에 마우스 포인터가 위치할 때

사용 예 :
<a href = "url" onMouseOver = "마우스 포인터가 링크 영역에 위치할 때
실행되는 함수"> 링크 텍스트 또는 링크 이미지 </a>
```

  - 마우스 이벤트 처리기의 기능(2)

```
onMouseOut
: 특정한 요소로부터 마우스 포인터가 벗어날 때

사용 예 :
<a href = "url" onMouseOut = "마우스 포인터가 링크 영역에서 벗어날 때
실행되는 함수"> 링크 텍스트 또는 링크 이미지 </a>
```

JavaScript

이벤트와 이벤트 핸들러 (Cont'd)

- **이벤트(Event)**
  - 이벤트(사건)에 의해 실행되는 언어
  - 사용자에 의한 특정 행위의 결과로 발생.
  - 일반적으로 사용자와 프로그램 사이에 상호작용을 처리
  - 예)
    - 하이퍼링크 위에 마우스 커서가 위치하는 경우
    - 사용자의 버튼 클릭
  - 이벤트 드리븐(Event driven) 방식
    - 이벤트 발생에 따라 이벤트 핸들러가 실행되는 방식
- **이벤트 핸들러(Event Handler)**
  - 이벤트 처리를 위해 정의된 함수나 메소드
  - 이벤트 앞에 on을 붙이면 event handler가 된다.
  - 형식 :
    - <INPUT TYPE="button" VALUE="확인" **onClick="JavaScriptCode()"**>

JavaScript

이벤트와 이벤트 핸들러 (Cont'd)

- 사용자가 마우스를 움직이거나 키를 누르는 등의 동작을 event라 한다.
- 이 이벤트 앞에 on을 붙이면 event handler가 된다.
- 사용자의 행위 자체는 이벤트
- 사용자의 행위를 전달하는 시점은 이벤트 핸들러

JavaScript

이벤트와 이벤트 핸들러 (Cont'd)

- **이벤트의 종류(11개)**
  - abort, blur, click, change, error, focus, load, mouseout, mouseover, select, submit
- **이벤트 핸들러**
  - 이벤트 이름 앞에 on을 붙임 (예: onClick)
  - **의미**
    - 마우스 관련 – onClick, onMouseout, onMouseover
    - 포커스와 관련 – onBlur, onFocus
    - 로딩 중 – onAbort, onError, onLoad
    - 선택과 관련 – onChange, onSelect
    - 전송과 관련 – onSubmit

JavaScript

이벤트와 이벤트 핸들러 (Cont'd)

- 상태바에 메시지 나타내기
  - onLoad
    - 웹 페이지를 불러올 때 발생하는 이벤트
    - Ex) OnLoad = " status= '메시지' "
  - onMouseOver
    - 마우스가 특정 위치에 있을 때 발생하는 이벤트
    - Ex) OnMouseOver = " status= '메시지' "
  - onMouseOut
    - 마우스가 특정 위치를 벗어날 때 발생하는 이벤트

JavaScript

이벤트와 이벤트 핸들러 (Cont'd)

- 상태바에 메시지 나타내기
  - onLoad

```
<HTML>
<HEAD>
<TITLE> 상태바에 메시지 나타내기 </TITLE>
</HEAD>
<BODY onLoad="status='환영합니다!'"
onUnload="alert('안녕히 가세요')" >
</BODY>
</HTML>
```

JavaScript

이벤트와 이벤트 핸들러 (Cont'd)

- 상태바에 메시지 나타내기
  - onMouseOver

```
<HTML><HEAD>
<TITLE>Status Bar(상태선)에 글자 출력</TITLE>
<BODY>
 <a href=http://www.seoultech.ac.kr onMouseOver=
   "window.status='서울과학기술대학교 홈페이지로 이동합니다';
   return true">서울과학기술대학교</a>
</BODY></HTML>
```

JavaScript

이벤트와 이벤트 핸들러 (Cont'd)

- 상태바에 메시지 나타내기
  - onMouseOut

```
<HTML>
 <HEAD>
 <TITLE> 마우스 이벤트에 따른 이미지 변환 </TITLE>
 </HEAD>
 <BODY>
      <A HREF="http://www.seoultech.ac.kr"
        onMouseOver = 'document.image(0).src = "on.gif" '
        onMouseOut = 'document.image(0).src = "off.gif" '>
       <IMG SRC = 'off.gif' border=0></A>
</BODY><HTML>
```

JavaScript

이벤트 핸들러의 종류

- onLoad( ) : 웹 브라우저 홈페이지를 불러올 때
- onUnload( ) : 현재 홈페이지에서 빠져 나가려할 때
- onClick( ) : 마우스를 클릭할 때
- onFocus( ) : 양식이나 홈페이지에 커서나 포커스가 위치했을 때
- onBlur( ) : 양식이나 홈페이지에서 커서나 포커스가 다른 곳으로 이동할 때
- onMouseover( ) : 마우스가 위로 왔을 때
  - 특정한 링크 또는 영역에 마우스 포인터가 위치하게 될 때의 동작을 정의한다.
- onMouseout( ) : 마우스가 영역을 벗어났을 때
  - 특정한 링크 또는 영역으로부터 마우스 포인터가 벗어나게 될 때의 동작을 정의한다.