

# 자바스크립트 (웹 프로그래밍)

## □ 자바스크립트 프로그래밍 기초

### ※ 자바스크립트 시작하기

#### ○ 자바스크립트 개요와 특징

##### \* 개요 및 특징 // ECMAScript 라고도 불림

- 동적인 웹 문서, 웹 응용프로그램을 위한 사용자 인터페이스 개발
- 작성 및 실행이 매우 간편함 / 인터프리터 (interpreter) 방식

#### ○ 객체 기반의 자바스크립트

##### \* 자바 언어와 차이점

	자바스크립트	자바 언어
실행 방식	웹 브라우저에서 실행 - 자바스크립트 코드를 해석하고 바로 실행(스크립트/인터프리터 기반 언어)	자바 가상머신에서 실행 - 자바 프로그램을 중간코드로 컴파일 후 변환된 object code를 실행하는 방식(컴파일 기반)
성격	객체기반 (object-based)	객체지향 (object-oriented)
작성 형태	HTML 파일 내에 포함되어 작성	별도의 자바 프로그램 파일로 작성
변수형 선언 및 타입 검사	변수의 선언이 따로 필요 없음, 타입 검사도 매우 느슨함	변수의 선언이 필요, 변수 타입의 검사가 엄격

#### ○ 자바스크립트 작성

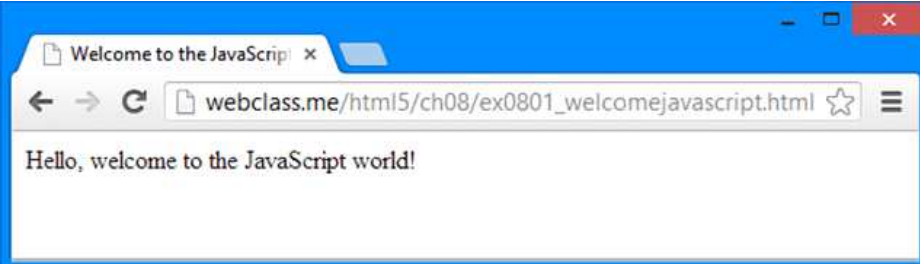
##### ◎ 웹문서 내장 방식

/// 웹문서 내장 방식	
01	<!-- HTML documents... -->
02	...
03	<script type = "text/javascript">
04	<!--
05	// 자바스크립트 코드
06	// -->
07	</script>
08	....
09	<!-- HTML documents... -->

##### ◎ 외부 파일 참조 방식

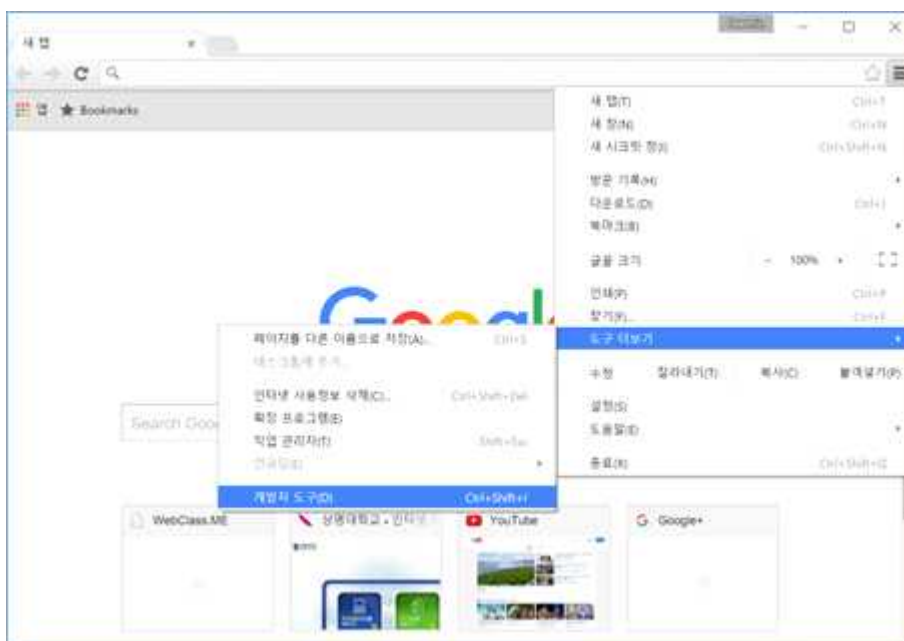
/// 외부 파일 참조 방식	
01	<!-- HTML documents... -->
02	...
03	<script type = "text/javascript" src="myscript.js"> </script>
04	...
05	<script type = "text/javascript"
	src="http://webllass.me/html5/javascript_example.js"> </script>
07	....
08	<!-- HTML documents... -->

\* 간단한 자바스크립트 예제

/// 기본 자바스크립트 예제	
01	<!DOCTYPE html>
02	<!-- hello.html
03	간단한 인사말을 화면에 표시하는 HTML/자바스크립트 기본 예제
04	-->
05	<html>
06	<head>
07	<title> Hello JavaScript </title>
08	</head>
09	<body>
10	<script type="text/javascript">
11	<!--
12	document.write("Hello! JavaScript!");
13	// -->
14	</script>
15	</body>
16	</html>
출력	

○ 자바스크립트 실행 및 디버깅

\* [Chrome 설정 및 관리] 버튼(☰) -> [도구 더보기] -> [개발자 도구] -> [콘솔] 실행



## ※ 자바스크립트 기본 문법

### ○ 자바스크립트 기본 변수

#### \* 기본 변수 타입

- 대부분의 경우 미리 선언할 필요가 없음
- 타입도 지정할 필요가 없음

#### \* 내부적인 변수의 다섯가지 기본 형식

- Number, String, Boolean, Undefined, Null

#### \* 숫자의 표현 형태 (내부적으로 숫자는 모두 실수로 저장)

- 125    1.25    0.125    .125    125.    12.e5    1.2e-5    12E5    12e5    .12e5

### ○ 자바스크립트 기본 변수 타입

기본 변수 타입	변수 값	비고
Number	정수, 실수 등 숫자 값을 가짐	
String	연속된 글자들로 이루어진 문자열(공백도 가능), 문자열의 시작과 끝은 작은 따옴표 (') 혹은 겹따옴표 (")로 지정	숫자 (Number)와 문자열 (String)타입 간에는 숫자 값에 대해 자동 형변환을 제공한다.
Boolean	true 혹은 false	조건식에서 사용
Undefined	undefined 만 가능	
Null	null 만 가능	

#### \* 자바스크립트 변수형은 typeof() 연산자를 이용해서 확인 가능

- typeof(123) -> "Number"를 반환
- typeof("123") -> "String"을 반환

### ○ 자바스크립트 변수 선언

#### \* 변수를 사전에 선언 없이 사용하는 것이 가능

- 전역 변수로 사용할 때는 미리 선언해야함

#### \* 별도의 변수 타입이 없으며 var 선언 한가지만 제공

#### \* 변수에 값이 저장될 때, 그 값에 따라 내부적으로 변수 타입 결정

- 문자열, 정수, 실수 등 타입 상관 없음  
    // ex) var 변수명; 혹은 var 변수명 = 변수 값;
- 변수의 선언 방식 : 대소문자 구분

/// 변수 선언 예제	
01	var index, name = "모바일 웹";
02	var start = 0, end = 100.0;
03	var message, condition, sender, receiver;
04	
05	var a = "3";
06	var b = 2;
07	c = b + 3 + a;    // c 값은 "53"이 됨, 더하기 후 문자열 붙이기 연산
08	d = a + b;        // d 값은 "3"+"2"="32", 문자열 연산(Concatenation)

## ○ 기본 연산자

종류	연산자	설명	비고
사칙연산	+ - * / %	더하기, 빼기, 곱하기, 나누기, 나머지	-내부적으로 실수 값으로 변환 후 처리
대입 연산자	+= -= *= /= %=	왼편 변수에 우측 값을 연산 후 왼편 변수에 대입	-자바 언어와 동일한 연산 우선순위
증감 연산자	++ --	기존 변수 값에 +1 혹은 -1 연산 수행	
논리연산자	> <	왼편의 값이 크다, 작다	결과값이 true 혹은 false이다.
	>= <=	왼편의 값이 크거나 같다, 작거나 같다	
	== !=	양편이 같다, 다르다 (값만 비교)	
	=== !==	양편의 같다, 다르다 (값과 타입 모두 비교)	
	!    &&	논리부정 (NOT), 논리합 (OR), 논리곱 (AND) 연산	

### \* 문자열 붙이기 (Concatenation) 연산

- '+' 연산자를 이용해서 두 문자열을 붙임

```
var first_name = "Steve";
var last_name = "Jobs";

var full_name1 = first_name + " " + last_name;
// full_name1: "Steve Jobs"

var full_name2 = last_name + ", " + first_name;
// full_name2: 'Jobs, Steve'
```


## ○ 변수 형 변환

### \* 문자열 타입 -> 숫자 타입

◎ parseInt (문자열 변수) 혹은 parseFloat (문자열 변수) 함수

### \* 숫자 타입 -> 문자열 타입

◎ toString (숫자 형 변수) 메소드를 이용

/// 변수 형 변환 예제	
01	var length = 123, length_num, length_str;
02	
03	length_num = length + 10;
04	length_str = length.toString() + 10;
05	
06	document.write("Length in Number: " + length_num + "cm" ");
07	document.write("Length in String: " + length_str + "cm" ");
08	
09	var num = parseInt(lenth_str) + 20;
10	document.write("Length in Integer: " + num + "cm");
출력	

## ○ 화면 출력

◎ **document.write()** : 화면 출력 명령어

\* HTML 문서에 콘텐츠를 추가하여 화면에 출력

- HTML 태그를 추가할 경우에는 그 태그도 해석되어 화면에 출력

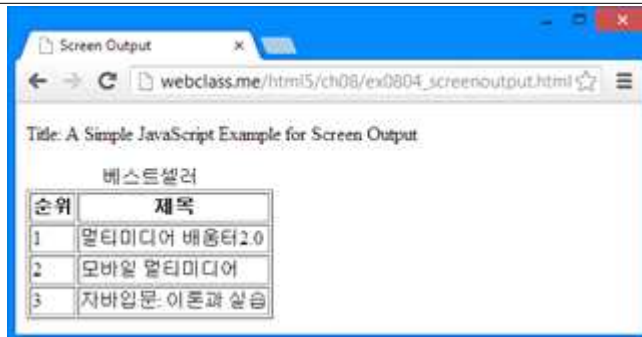
\* HTML 문서는 Document라는 객체로 모델링 되어 있음

- document라는 이름으로 접근 => Document 객체의 write() 메소드

/// 화면 출력 예제

```
01 var title1 = "멀티미디어 배움터 2.0";
02 var title2 = "모바일 멀티미디어";
03 var title3 = "자바입문: 이론과 실습";
04
05 document.write("<caption> 베스트셀러 </caption>");
06 document.write("<tr>");
07 document.write("<th> 순위 </th>");
08 document.write("<th> 제목 </th>");
09 document.write("</tr>");
10 document.write("<tr> <td> 1 </td> <td> " + title1 + "</td> </tr>");
11 document.write("<tr> <td> 2 </td> <td> " + title2 + "</td> </tr>");
12 document.write("<tr> <td> 3 </td> <td> " + title3 + "</td> </tr>");
```

출력



## ○ 대화상자로 메시지 출력

\* 대화상자 (dialog box)로 입력을 받는 세 가지 방법

- alert(), prompt(), confirm()

◎ **alert()** 명령어

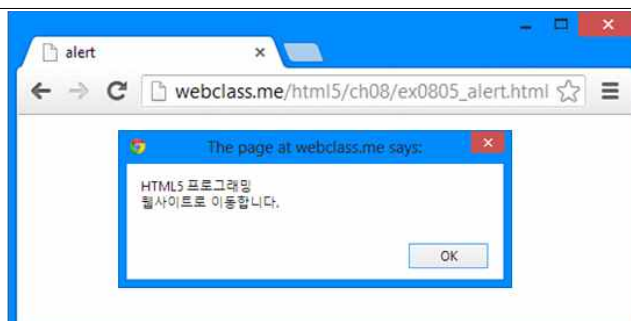
- 사용자에게 경고 사항이나 메시지를 전달

- “확인” 버튼을 클릭하지 않으면 다음 자바스크립트 문장이 실행되지 않음

/// alert() 명령어 예제

```
01 alert("HTML5 프로그래밍 \n 웹사이트로 이동합니다.");
```

출력



### ◎ confirm() 명령어

- 사용자에게 Yes/No 선택을 입력 => “확인”과 “취소” 버튼
- “확인” 버튼을 누르면 true, 취소 버튼을 누르면 false를 반환

/// alert() 명령어 예제	
01	var answer = confirm("주문한 서적을 결제하시겠습니까?");
02	document.write("Answer = " + answer + " ");
출력	

### ◎ prompt() 명령어

- 사용자에게 키보드를 통해 문자열을 입력
- “확인” 버튼을 누르면 입력된 문자열, “취소”를 누르면 null을 반환

/// alert() 명령어 예제	
01	var answer = prompt("서적 제목을 입력해 주세요.", "모바일 멀티미디어");
02	document.write("Answer = " + answer + " ");
출력	

### ※ 자바스크립트 제어문 및 반복문

- \* if / for / while

### ※ 자바스크립트 함수

- \* function(변수){} 사용,

## □ 자바스크립트 객체와 DOM

※ 자바스크립트 내장 객체 다루기

※ 자바스크립트 사용자 정의 객체 다루기

※ DOM으로 HTML 문서 다루기

### ○ DOM의 정의 및 문서 구조

#### \* DOM (Document Object Model)

- 웹 문서를 자바 스크립트 입장에서 구조적 문서 (document) 객체 형태로 다루는 모델
- 자바스크립트는 HTML 문서를 객체로 바라보고 다룬다.
- HTML 문서 뿐만 아니라 CSS 속성도 변경 가능

#### \* 자바스크립트 활용

- 자바스크립트를 이용하여 HTML 문서의 내용을 변경
- 사용자 입력을 받아 처리

### ○ HTML 태그 요소와 DOM

#### \* 태그 요소는 DOM의 객체로 표현 됨

- 태그 요소에 포함된 다른 요소는 객체 내에 소속된 객체 형태로 표현 (하위 객체)

#### \* 태그 속성은 DOM 객체의 속성으로 표현 됨

#### \* 요소 전체가 하나의 객체

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

- type과 name은 속성
- “text”와 “username”은 type과 name의 속성 값

### ○ DOM을 통한 HTML 문서 접근

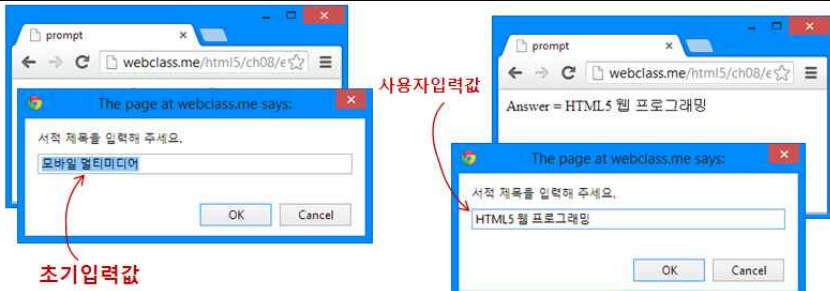
\* 자바스크립트 입장에서는 웹 브라우저 환경과 HTML 문서를 모두 객체로 바라보고 처리

#### \* DOM 접근 방법

- 1. document의 forms 속성을 이용해서 접근하는 방법
  - 2. 요소 이름을 이용해 접근하는 방법
  - 3. document 객체가 제공하는 getElementByld() 등의 메소드를 이용해서 접근하는 방법
- // 가장 사용이 쉽고 많이 사용되는 방법인 getElementByld() 메소드 방법을 중심으로 설명

#### /// DOM 접근 방법 예제 1

출력

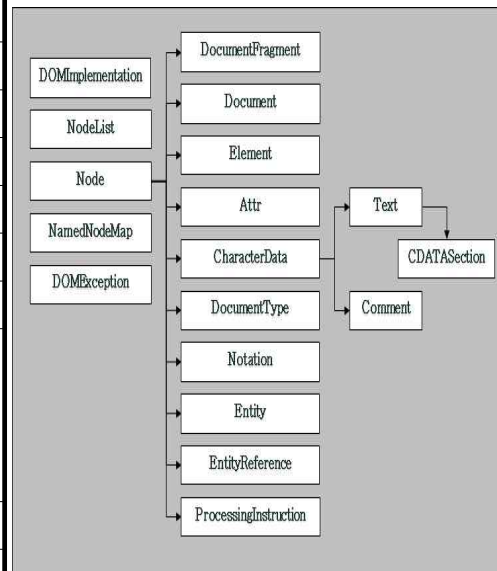


## □ DOM (Document Object Model)

### ○ Document Interface : Document 불러오기

- \* Document 객체는 문서 자체를 나타내며 문서도 일종의 노드로 취급되어 Node의 모든 멤버를 사용가능
- \* `var xmlDocument.load("mydoc.xml");` // load할 수 있는 스크립트 필요
- \* `var rootEl = xmlDocument.documentElement;` // 해당 문서의 최상위 엘리먼트를 불러서 rootEl에 할당

속성	설명
<code>title</code>	문서의 제목
<code>URL</code>	문서의 주소
<code>inputEncoding</code>	문서의 인코딩 정보
<code>links</code>	링크의 배열
<code>images</code>	이미지의 배열
<code>forms</code>	입력 양식의 배열
<code>anchors</code>	앵커의 배열
<code>cookie</code>	이름과 값의 쌍으로 된 쿠키의 집합
<code>lastModified</code>	문서의 최후 수정 시간
<code>readyState</code>	문서의 로드 상태 uninitialized : 시작하지 않음 loading : 로딩 중 interactive : 사용자 사용 가능 complete : 로드 완료
<code>documentElement</code>	루트인 html 엘리먼트를 리턴
<code>body</code>	body 엘리먼트를 리턴



### ○ NodeType (Interfaces)

속성	설명	nodeName	nodeValue
<code>Attr</code>	태그의 속성	속성 이름	속성 값
<code>CDATASection</code>	파서에 의해 해석되지 않는 섹션	#cdata-section	노드의 내용
<code>CharacterData</code>	문자 데이터	#character-data	null
<code>Comment</code>	주석	#comment	노드의 내용
<code>Document</code>	전체 문서	#document	null
<code>DocumentFragment</code>	문서 조각	#document fragment	null
<code>DocumentType</code>	문서 종류	doctype name	null
<code>DOMImplementation</code>	DOM 메소드 제공	?	null
<code>Element</code>	요소 노드이며, 태그이다.	태그 이름	null
<code>Entity</code>	엔터티	엔터티 이름	null
<code>EntityReference</code>	엔터티 참조	엔터티 참조 이름	null
<code>NamedNode</code>	노드의 이름	노드 이름	null
<code>Node</code>	노드	노드 이름	null
<code>NodeList</code>	노드의 목록	노드 이름(목록)	null
<code>Notation</code>	DTD에 선언된 노테이션	노테이션 이름	null
<code>ProcessingInstruction</code>	처리 지시문	target	노드의 내용
<code>text</code>	태그에 포함된 문자열 내용	#text	노드의 내용



## ○ Node Methods

속성	설명
<code>appendChild(node)</code>	가지고 있는 노드 뒤에 새로운 노드를 추가 (이미 존재하면 기존 노드는 삭제되고 새로운 위치에 옮겨짐)
<code>appendData(text)</code>	노드 마지막에 text를 추가
<code>cloneNode(true)</code>	노드를 복제 (처일드까지 같이 복사됨)
<code>contains(node)</code>	주어진 노드가 다른 노드의 자식인지 조사
<code>createAttribute(attributename)</code>	새로운 속성 노드를 생성
<code>createCDATASection(cdata-section)</code>	새로운 섹션을 추가
<code>createComment(text)</code>	새로운 주석을 추가
<code>createDocumentFragment</code>	새로운 문서 조각을 추가
<code>createElement(nodename)</code>	새로운 태그를 추가
<code>createEntityReference()</code>	새로운 엔터티 참조를 추가
<code>createProcessingInstruction</code>	새로운 처리문을 추가
<code>createTextNode(text)</code>	새로운 텍스트 노드를 추가
<code>deleteData(offset, count)</code>	offset부터 count만큼 데이터 삭제
<code>getElementByClassName(classname)</code>	요소ID로 자식을 찾음
<code>getElementById(elementID)</code>	요소ID로 자식을 찾음
<code>getElementsByName(name)</code>	이름으로 자식을 찾음
<code>getElementsByTagName(tagname)</code>	태그 이름으로 자식을 찾음
<code>hasChildNodes</code>	자식 노드가 존재하는지 조사
<code>hasFeature</code>	
<code>hasFocus()</code>	document가 포커스가 있는지 조사
<code>innerHTML</code>	태그까지 읽어 들임
<code>innerText</code>	태그는 빼고 순수 텍스트만 읽음
<code>insertBefore(newnode, existingnode)</code>	삽입할 노드 위치의 앞쪽에 추가 (이미 존재하면 기존 노드는 삭제되고 새로운 위치에 옮겨짐)
<code>insertData(offset, text)</code>	offset 위치에서 text를 삽입
<code>importNode()</code>	노드를 다른 document에서 가져옴
<code>isEqualNode()</code>	같은 노드인지 조사
<code>item</code>	노드의 내용
<code>nomalize</code>	빈 텍스트 노드를 찾으려면 제거, 텍스트 노드끼리 형제인 경우 두 노드를 하나로 합침
<code>outerText</code>	태그 제외 텍스트에 노드자체 태그까지 읽음
<code>outerHTML</code>	모든 태그와 노드 자체 태그까지 읽음
<code>querySelector(sel)</code>	선택자를 이용해 자식을 찾음
<code>querySelectorAll(sel)</code>	선택자를 이용해 모든 자식을 찾음
<code>removeChild(node)</code>	자식 노드를 제거(부모 노드에서 가능)
<code>removeNamedItem</code>	특정 이름을 가진 노드를 제거
<code>replaceChild(newnode, oldnode)</code>	기존 노드를 다른 노드로 교체
<code>replaceData(offset, count, text)</code>	offset부터 +count까지 텍스트를 text로 교체
<code>setNamedItem</code>	배열에 노드를 추가함, 이미 있으면 덮어씀
<code>splitText(offset)</code>	offset 위치를 기준으로 텍스트 노드를 둘로 나눔
<code>substringData(offset, count)</code>	offset 위치부터 +count까지 텍스트를 꺼냄
<code>textContent</code>	자신과 자신의 자식 텍스트를 전부 합쳐 읽음

## ○ Attributes (Node Interface)

속성	설명
<code>attributes</code>	해당 노드의 속성 값 반환
<code>baseURI</code>	기초 url을 반환
<code>childNodes</code>	자식 노드를 반환
<code>data</code>	해당 노드의 데이터(text) 반환
<code>doctype</code>	문서의 종류를 반환
<code>documentElement</code>	루트인 html엘리먼트를 반환
<code>firstChild</code>	첫 번째 노드를 반환
<code>implementation</code>	DocumentImplementation 객체로 이 문서를 처리하는 DOMImplementation개체를 반환
<code>innerHTML</code>	태그의 내용물
<code>lastChild</code>	마지막 노드를 반환
<code>length</code>	노드의 길이를 반환
<code>namenextSibling</code>	'네임'의 다음 노드를 반환
<code>nextSibling</code>	다음 형제노드를 반환
<code>nodeName</code>	노드의 이름을 반환
<code>nodeType</code>	노드 타입을 반환 (밑 NodeType 참조)
<code>nodeValue</code>	노드의 값을 반환
<code>notationName</code>	DTD에 선언된 노테이션 이름을 반환
<code>Notations</code>	DTD에 선언된 노테이션을 반환
<code>ownerDocument</code>	소속된 문서를 반환
<code>parentNode</code>	부모의 노드를 반환
<code>previousSibling</code>	이전의 노드를 반환 (sibling : 자매 라는 뜻)
<code>publicId</code>	공개된 id를 반환
<code>specified</code>	값이 지정되어 있는지 조사
<code>specifiedsystemId</code>	값이 지정되어있는 시스템 id를 반환
<code>systemId</code>	시스템 id를 반환
<code>style</code>	엘리먼트 스타일
<code>tagName</code>	해당 노드의 태그 이름을 반환
<code>Target</code>	지정한 노드를 반환
<code>value</code>	노드의 값을 반환

- `previousSibling` : 자매 라는 뜻, 이전의 노드를 반환
- `curNode.firstChild.firstChild` : 노드 안의 노드를 반환
- `element.childNodes.length` : 총 노드의 길이를 반환해줌(for문에서 사용)

## ○ NodeList Interface

\* NodeList에 있는 노드들은 `item()` 메소드로 얻음

\* NodeList의 길이는 `length` 특성을 구함

/// nodeList interface	
01	<code>var x = element.childNodes; // x에 childNodes의 속성을 넣음</code>
02	<code>if (x &lt; element.childNodes.length){</code>
03	<code>var current = x.item(0);</code>

○ NodeType : 노드 타입에 오류가 날 수 있으므로

/// NodeType		
Interget value	Defined Constant	설명
01	Element node	엘리먼트 노드
02	Attribute node	속성 노드
03	Text node	텍스트 노드
04	CDATA section node	섹션 노드
05	Entity node	엔터티 노드
06	Processing Instruction node	처리 지시문 노드
07	Comment node	주석 노드
08	Document Type node	문서 타입 노드
09	Document Fragment node	문서 조각 노드
10	Notation node	

/// nodeType	
01	var xnode = myDoc.childNodes;
02	document.write(xnode.(3).nodeType); // nodeType의 3번째 노드를 출력

○ nodeName 과 nodeValue

/// nodeName 과 nodeValue	
01	var xnode = myDoc.childNodes;
02	document.write(xnode.(i).nodeName); // nodeType의 i번째 노드이름을 출력
03	document.write(xnode.(i).nodeValue); // nodeType의 i번째 노드 값을 출력

○ Factory Methods와 Node Interface (/숫자는 argument 수)

\* insertBefore/2, replaceChild/2,

○ Factory Methods와 Document Interface (/숫자는 argument 수)

\* createElement/TagName, createDocumnetFragment/0

○ Attributes

속성	설명
name	속성의 이름
value	속성의 값
isId	ID속성인지 조사
ownerElement	속성이 소속된 엘리먼트
specified	값이 지정되어 있는지 조사

/// Attributes	
01	<greeting type="cordial" position="first">
02	x = anElenent.attributes(0); // anElenent = greeting
03	

\* Element와 Attr 메소드

속성	설명
<code>createAttribute(attributename)</code>	새로운 속성 노드를 생성
<code>getAttribute(attributename)</code>	이름으로부터 속성을 찾음
<code>getAttributeNode</code>	이름으로부터 속성 노드를 찾음(속성의 이름을 문자열 형태로 지정)
<code>getElementsByClassName(classname)</code>	요소ID로 자식을 찾음
<code>getElementById(elementID)</code>	요소ID로 자식을 찾음
<code>getElementsByName(name)</code>	이름으로 자식을 찾음
<code>getElementsByTagName(tagname)</code>	태그 이름으로 자식을 찾음
<code>getNamedItem</code>	이름으로부터 속성을 읽음
<code>hasAttributes()</code>	속성이 존재하는지 조사
<code>removeAttribute</code>	속성을 제거
<code>removeAttributeNode</code>	속성의 노드를 제거
<code>removeNamedItem(nodename)</code>	노드를 제거
<code>setAttribute(attributename, attributevalue)</code>	속성 값을 변경(속성의 이름을 문자열 형태로 지정)
<code>setAttributeNode</code>	속성 노드를 변경
<code>setNamedItem(node)</code>	배열에 노드를 추가, 이미 있으면 덮어씀

○ Sample JavaScript Code

```

/// 자바스크립트 예제 1 - xml 문서파일
01 <APEC_SATMAN ID="IA1001">
02     <NAME>조광민</NAME>
03     <TITLE>학생</TITLE>
04     <ADDRESS>부산</ADDRESS>
05 </APEC_SATMAN>
    
```

```

/// 자바스크립트 예제 1 - 파일 불러오기
06 <script type = "text/javascript" language="JavaScript">
07     var xmlDocument = new ActiveXObject("Microsoft.XMLDOM");
08     xmlDocument.load("hument.xml");
09     //get the root element / 최상위 태그를 xmlDocument로 연동
10     var element = xmlDocument.documentElement;
    
```

○ 역파일 : 유관 검색어가 이 파일에 있음

## □ BOM (Browser Object Model)

### ○ 자바스크립트 명령어

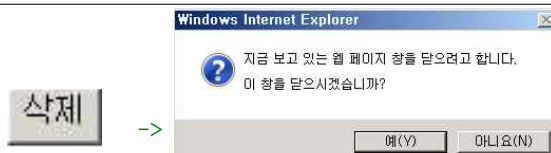
#### \* Window 객체 : 자바스크립트 최상위 객체 (C#의 Object 클래스)

속성	설명
Window.Alert()	경고 대화상자
Window.Confirm()	확인 대화상자
Window.Prompt()	입력 대화상자 (잘 안씀)
Window.Open()	새 창 열기
Window.Close()	현재 창 닫기
Window.Status = "상태바에 문자열 출력"	속성
Window.setTimeout(code, delay)	해당 시간 이후 명령어 실행

#### /// Window 객체 예제

```
01 <script>
02     function CheckDelete(){
03         //확인 버튼 클릭하면 true, 취소 버튼 : false
04         var flag = windw.confirm("정말로 삭제?");
05         if (flag) { alert("삭제 진행"); return true; }
06         } else { alert("멈춤"); return false; }
07     }
08 </script>
09 <input type="button" value="삭제" onclick="CheckDelete()" />
```

출력



#### \* Document 객체

속성	설명
title	타이틀 바에 출력
Write()	문서에 출력
bgColor	배경색
fgColor	전경색
getElementById()	id 속성을 통해서 해당 객체 가져오기

#### \* Location 객체

속성	설명
href	링크 이동
reload()	새로고침

\* history 객체

속성		설명
go(n)		n번 째 방문정보로 이동
	back()	뒤로
	forward()	앞으로

\* form 객체

속성	설명
value	폼 관련 태그 안에 저장된 값
focus()	해당 컨트롤(개체)에 포커스
submit()	폼 내용 전송 - action이 지정된 곳으로 전송
action	전송될 서버 측 URL을 동적으로 재지정
length	해당 컨트롤의 값의 길이
select()	해당 컨트롤의 텍스트를 선택 (블록 씌우기)
checked	체크박스 컨트롤이 체크되었다면 true / false
disabled	모든 컨트롤의 비활성화 여부를 결정 true/false
selectedIndex	드롭다운리스트 컨트롤의 현재 선택된 인덱스 값 반환

\* style 객체

속성		설명
style		모든 객체의 CSS속성을 변경 가능
	visibility	존재하는지 여부
	display	화면에 보여줌

\* 내장 객체(함수)

속성	설명
replace(“원본”, “바꿀 값”);	해당 값을 바꿈

## □ 이벤트 처리와 동적 웹 문서

### ※ 이벤트 처리하기

#### ○ 이벤트 처리 예제

```

1 <body>
2 <form>
3   <input type="button" value="Yes" onclick="alert('You pressed Yes');" />
4   <input type="button" value="No" onclick="alert('You pressed No');" />
5 </form>
6 </body>

```

클릭 이벤트에 반응하여 alert() 함수를 호출하는 부분

#### ○ 이벤트의 종류

속성	설명
load, unload	페이지 시작 및 종료
click, doubleclick	마우스 클릭, 더블 클릭
mousedown, mouseup	마우스 버튼 누름, 땀
mousemove	마우스 이동
mouseover, mouseout	마우스가 경계 안으로 들어오거나 나감
contextmenu	오른쪽 마우스 클릭
focus, blur	포커스 얻음, 잃음
submit	폼 전송
reset	폼 초기화
resize	윈도우와 프레임의 크기 변경
select	텍스트 선택
scroll	스크롤
abort	이미지 로드 중단
ctrlKey, altKey, shiftKey	조합키의 눌림 여부
keyCode	눌려진 키 코드 값
screenX, screenY	화면 좌표
clientX, clientY	작업영역 좌표

#### ◎ 마우스 이벤트

이벤트 이름	태그 속성	설명
click	onclick	HTML 문서내의 요소를 클릭했을때 발생한다.
dblclick	ondblclick	HTML 문서내의 요소를 더블클릭했을때 발생한다.
mousedown	onmousedown	마우스 커서를 HTML 문서내의 요소 위에 위치시키고 마우스 버튼을 누를때 발생한다.
mousemove	onmousemove	마우스 커서를 HTML 문서내의 요소 위에서 이동시킬때 발생한다. 마우스 커서를 움직이는 동안에는 계속해서 이벤트가 발생한다.
mouseup	onmouseup	사용자가 마우스 커서를 HTML 문서내의 요소 위에 위치시키고 마우스 버튼을 땀때 발생한다.
mouseover	onmouseover	마우스 커서가 해당 요소 위에 위치할 경우에 발생한다. 요소 위에 위치할때 1회만 발생하며 연속해서 발생하지 않는다.
mouseout	onmouseout	마우스 커서가 해당 요소 위를 벗어날때 발생하는 이벤트이다.

## ◎ 키보드 이벤트

이벤트 이름	태그 속성	설명
keypress	onkeypress	키보드를 타이핑할때 발생하는 이벤트이다. 키보드를 누를 때 1회 발생하고 손을 떼기 전까지 주기적으로 계속 이벤트가 발생한다.
keydown	onkeydown	키보드를 누를때 발생하는 이벤트이다. 키보드를 눌러서 내려갈때 1회 발생한다.
keyup	onkeyup	키보드를 누른 후 떼때 발생하는 이벤트이다. 키보드에서 손을 떼때 키보드가 올라올때 1회 발생한다.

## ◎ 프레임 / 객체 이벤트

이벤트 이름	태그 속성	설명
load	onload	문서, 프레임, 객체 등이 웹 브라우저상에 로드가 완료 되었을때 발생하는 이벤트이다.
resize	onresize	문서 창, 문서 뷰의 크기가 리사이즈(resize) 되었을 경우 발생한다.
scroll	onscroll	문서 창, 문서 뷰가 스크롤 되었을 경우 발생한다.

## ◎ 폼 이벤트

이벤트 이름	태그 속성	설명
change	onchange	<input>, <selection>, <textarea>등 폼 요소 콘텐츠의 내용이 변경되었을 때 발생하는 이벤트이다.
focus	onfocus	요소가 포커스 되었을때 발생하는 이벤트이다. 마우스로 선택되거나 입력 커서가 해당 요소에 위치할때 발생한다.
blur	onblur	focus 이벤트의 반대 개념으로 요소에서 포커스가 없어질 때 발생하는 이벤트이다. 즉, 요소가 마우스 선택이 해제되거나 입력 커서가 다른 곳으로 이동할 때 발생한다.
select	onselect	<input> 과 <textarea> 요소 내의 텍스트의 일부 혹은 전부가 선택 되었을 때 발생하는 이벤트이다.

## ○ 이벤트 핸들링 및 이벤트 등록

### \* 이벤트 핸들러(handler)

- 이벤트가 발생 시 실행하고자 하는 자바스크립트 함수나 코드
- 사용자가 입력한 내용이 맞는지 검사하거나 입력한 내용에 따라 웹 문서를 수정하는 등의 작업을 통해 동적 웹 문서를 만든다.

### \* 이벤트 등록

- 이벤트의 종류와 이를 처리할 이벤트 핸들러를 연결시키는 작업
- 두 가지 등록 방법
  1. 태그 속성에 직접 이벤트 핸들러 기술
  2. 객체의 이벤트 속성 값에 이벤트 핸들러 함수 기술

## ※ 폼 다루기

## ※ 동적 웹 문서 만들기

## ※ 다양한 방법으로 폼 다루기

////////////////////////////////////



속성	설명
<code>appendChild(node)</code>	가지고 있는 노드 뒤에 새로운 노드를 추가 (이미 존재하면 기존 노드는 삭제되고 새로운 위치에 옮겨짐)
<code>appendData()</code>	
<code>cloneNode(true)</code>	노드를 복제 (차일드까지 같이 복사됨)
<code>createAttribute(attributename)</code>	새로운 속성 노드를 생성
<code>createCDATASection(cdata-section)</code>	새로운 섹션을 추가
<code>createComment(text)</code>	새로운 주석을 추가
<code>createDocumentFragment</code>	새로운 문서 조각을 추가
<code>createElement(nodename)</code>	새로운 태그를 추가
<code>createEntityReference()</code>	새로운 엔터티 참조를 추가
<code>createProcessingInstruction</code>	
<code>createTextNode(text)</code>	새로운 텍스트 노드를 추가
<code>deleteData</code>	
<code>getAttribute(attributename)</code>	이름으로부터 속성을 찾음
<code>getAttributeNode</code>	이름으로부터 속성 노드를 찾음(속성의 이름을 문자열 형태로 지정)
<code>getElementsByClassName(classname)</code>	요소ID로 자식을 찾음
<code>getElementById(elementID)</code>	요소ID로 자식을 찾음
<code>getElementsByName(name)</code>	이름으로 자식을 찾음
<code>getElementsByTagName(tagname)</code>	태그 이름으로 자식을 찾음
<code>getNamedItem</code>	이름으로부터 속성을 읽음
<code>hasAttributes()</code>	속성이 존재하는지 조사
<code>hasChildNodes()</code>	자식 노드가 존재하는지 조사
<code>hasFeature</code>	
<code>innerHTML</code>	태그까지 읽어 들임
<code>innerText</code>	태그는 빼고 순수 텍스트만 읽음
<code>insertBefore(newnode, existingnode)</code>	삽입할 노드 위치의 앞쪽에 추가 (이미 존재하면 기존 노드는 삭제되고 새로운 위치에 옮겨짐)
<code>insertData</code>	
<code>item</code>	
<code>nomalize</code>	
<code>outerText</code>	태그 제외 텍스트에 노드자체 태그까지 읽음
<code>outerHTML</code>	모든 태그와 노드 자체 태그까지 읽음
<code>querySelector(sel)</code>	선택자를 이용해 자식을 찾음
<code>querySelectorAll(sel)</code>	선택자를 이용해 모든 자식을 찾음
<code>removeAttribute</code>	속성을 제거
<code>removeAttributeNode</code>	속성의 노드를 제거
<code>removeChild(node)</code>	자식 노드를 제거(부모 노드에서 가능)
<code>removeNamedItem</code>	특정 이름을 가진 노드를 제거
<code>replaceChild(newnode, oldnode)</code>	기존 노드를 다른 노드로 교체
<code>replaceData</code>	
<code>setAttribute(attributename, attributevalue)</code>	속성 값을 변경(속성의 이름을 문자열 형태로 지정)
<code>setAttributeNode</code>	
<code>setNamedItem</code>	배열에 노드를 추가함, 이미 있으면 덮어씀
<code>splitText</code>	
<code>substringData</code>	