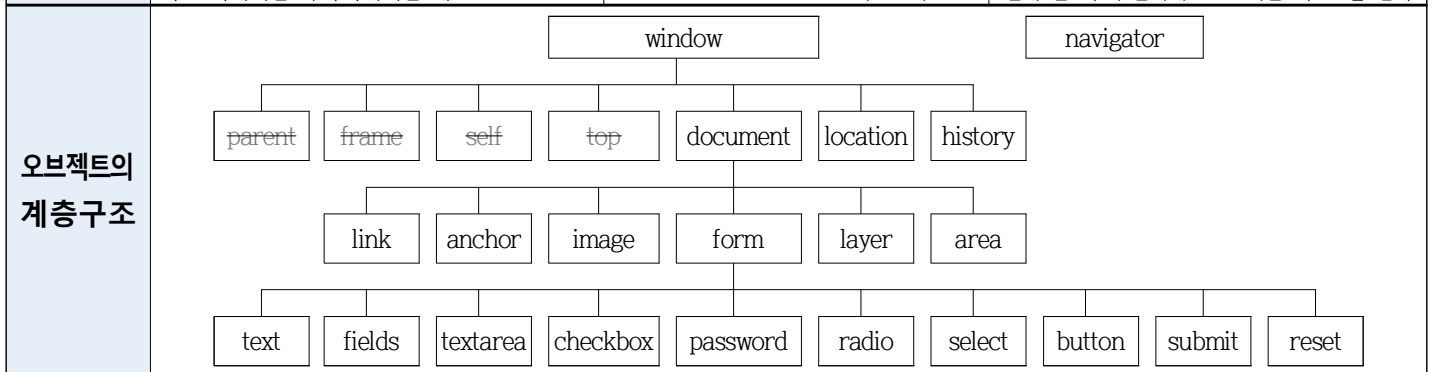


유형	용법	설명
HTML5 이전	HTML5 이후	
<pre><script language="javascript"> <!-- ~스크립트 내용~ // --> </script></pre>	⇒	<pre><script> ~스크립트 내용~ </script></pre>
행 입력형 (<TAG>의 안쪽)	문법 사용 예	<pre><태그이름 이벤트핸들러="스크립트 내용"></pre> <pre>click</pre>
내장형 (<HEAD>나 <BODY>의 내용)	문법 & 사용 예	<pre><head><script> function 함수이름() { ~스크립트 내용~ } </script></head> <body> <태그이름 onclick="함수이름()"> </body></pre> <p>함수 선언</p> <p>함수 호출</p>
외부 링크형 (js 파일을 링크)	문법 사용 예	<pre><script src="외부/파일주소"></script></pre> <pre><script src="myscript.js"></script></pre>

이름	오브젝트(Object ; 객체)	속성(Property)	메소드(Method) / 함수(Function)
설명	프로그램의 대상이 되는 모든 것	오브젝트의 속성이나 설명, 특징, 성격	오브젝트의 기능이나 동작, 역할
예시	window, document, form, button...	color, size, shape...	save, close...

오브젝트의 문법	문법	사용 예	설명
	부모객체이름.자식객체이름;	window.document;	현재 윈도우(브라우저)의 문서를 선택
	객체이름.속성;	window.name;	윈도우의 이름을 선택
	객체이름.속성 = "값";	window.name = "newPopup";	윈도우의 이름에 "newPopup"을 저장
	객체이름.메소드;	window.close();	현재 윈도우를 닫음
	부모객체이름.자식객체이름.속성;	window.document.frm1.desc.value;	"frm1"안에 있는 "desc"의 값을 선택
	부모객체이름.자식객체이름.속성="값";	window.document.frm1.desc.value = "text";	"frm1"안에 있는 "desc"의 값에 "text"를 저장
	부모객체이름.자식객체이름.메소드;	window.document.write("text");	현재 윈도우의 문서에 "text"라는 텍스트를 입력



JavaScript 주의사항 및 특징	대소문자를 반드시 구분하여야 한다. (특히 내장객체, 메소드, 속성, 변수 등은 대소문자를 혼동하면 작동하지 않는다) 매 구문(명령 한 줄)의 끝 마다 세미콜론(;)으로 구분하여야 한다. 객체, 속성, 메소드, 함수의 구분은 마침표(.) 연산자를 사용한다. (예: window.document.frm1.style.color = "red";) 문자열은 반드시 따옴표(")로 묶어야 한다. (큰 따옴표 작은 따옴표 상관없으나 중첩 시 종속관계 명확히 할 것) 스크립트는 쓰여진 순서대로 하나씩 실행된다. 다수의 공백은 무시한다. (가독성을 위해 공백을 넣어줄 수 있다.)
-----------------------------	---

이스케이프 문자	설명	이스케이프 문자	설명
\'	작은 따옴표	\t	탭
\"	큰 따옴표	\b	백 스페이스
\\	역 슬래시	\f	폼 피드
\n	줄 바꿈	// contents	주석 (한 줄)
\r	캐리지 리턴(줄의 맨 앞으로 이동)	/* contents */	주석 (여러 줄)

변수의 종류(Variables)	용법	설명
※ 숫자, 공백, 키워드, 함수, 오브젝트, 속성, 특수문자(" ", "\$"제외)는 변수의 첫 글자로 사용할 수 없다.		
숫자형(numbers)	var 변수이름 = 10;	계산할 수 있는 숫자를 변수에 대입한다.
부울형, 논리형(boolean)	var 변수이름 = true;	참(true, 1) 또는 거짓(false, 0) ※ 문자열이 아니므로 따옴표로 감싸지 않는다.
문자형(string)	var 변수이름 = "text";	계산할 수 없는 텍스트, 큰 따옴표나 작은 따옴표로 묶는다.
null	var 변수이름 = null;	null은 값이 존재하지 않는다는 뜻이다. (공백이나 0과는 다르다) 오브젝트의 종류로 간주된다. "자료형이 결정되었지만 비어있는 변수" "이 변수에 언젠가 객체를 할당하려고 하는데, 지금은 객체가 없습니다."
undefined	var 변수이름; var 변수이름 = undefined;	값이 할당되지 않은 변수 "자료형이 결정되지 않은 변수" "이 변수에 아직 아무 것도 정해진 것이 없습니다."
배열형(array)	var 변수이름 = [10, 20, 30];	var 변수이름 = new Array(); 변수이름[0] = 10; 변수이름[1] = 20; 변수이름[2] = 30;
	alert(변수이름[1]);	new Array() 키워드로 여러 개의 값들을 가진 배열(array)을 만들 수 있다. 각 데이터는 0부터 시작하는 고유의 색인 번호(index)를 가지고 있다. Link, Anchor, Image, Form 그리고 Class 는 색인번호를 가지고 있는 배열이다. ※ 색인 번호는 0부터 시작함
객체형(object)	var teacher = { firstName : "Jiho", lastName : "Han", height : 175, eyeColor : "black", sing : function(){ alert("lalala"); } };	alert(teacher.height + "," + teacher.eyeColor); teacher.sing(); 오브젝트도 변수의 일종이다. 여러 개의 속성과 값을 담을 수 있다는 점이 다르다. 속성과 값은 콜론(:)으로 구분한다. 함수를 담을 수도 있다. 이러한 경우에는 메소드를 생성하는 결과가 된다.
함수(function)	var 변수이름 = function () { 함수 내용 };	함수를 변수에 대입한다.

변수의 범위	사용 예	설명
지역 변수 Local variable	// myCar 변수 사용할 수 없음 function letsGoPicnic() { var myCar = "Benz"; // myCar 변수 사용할 수 있음 } // myCar 변수 사용할 수 없음 function otherFunction() { // myCar 변수 사용할 수 없음 }	<ul style="list-style-type: none"> 자바스크립트 함수 안에서 선언된 변수는 지역변수이며 변수가 선언된 그 함수 안에서만 인식되고 작동한다. 같은 이름의 변수라고 할지라도 다른 함수 안에 있거나 바깥에 있다면 전혀 다른 변수로 작동한다. 지역변수는 해당 함수가 끝나면 삭제된다.
전역 변수 Global variable	var myCar = "Benz"; // myCar 변수 사용할 수 있음 function letsGoPicnic() { // myCar 변수 사용할 수 있음 } // myCar 변수 사용할 수 있음 function otherFunction() { // myCar 변수 사용할 수 있음 }	<ul style="list-style-type: none"> 자바스크립트 함수 바깥에서 선언된 변수는 전역변수이며 현재 웹페이지의 모든 스크립트들과 함수들에서 사용할 수 있다. 전역변수는 페이지를 닫으면 삭제된다. 빠른 자바스크립트 구동을 위해 전역변수 사용을 최소화 한다.

대표적인 키워드 예약어		설명 (키워드 예약어 이름들은 변수, 함수 이름으로 사용할 수 없음)
break		반복문이나 스위치를 중지시키고 빠져나온다.
continue		반복문의 현재 위치에서 건너뛰고 반복문을 다시 시작한다.
for		조건이 참인 동안 블록을 반복한다. 초기화>조건식확인>블록실행>증감식실행>조건식확인>...
while		조건이 참인 동안 블록을 반복한다. 조건식확인>블록실행>조건식확인>...
do ... while		일단 먼저 내용을 한번 실행 시키고, 조건이 참인 동안 블록을 반복한다. (최소한 한번 이상 반복함)
if		조건이 참인지 거짓인지에 따라 해당 블록을 실행시킨다.
switch		서로 다른 조건의 결과에 따라 해당 블록을 실행시킨다.
return		함수를 종료하고 값을 반환한다.
var		변수를 선언한다.
new		객체를 생성한다.
delete		객체나 변수를 삭제한다.
function		함수를 선언한다.
this		현재 코드에서 자기 자신을 가리킨다.
typeof		변수의 타입을 판단한다.
기타 키워드 예약어	abstract, arguments, boolean, byte, case, catch, char, class, const, double, else, enum, eval, export, extends, false, final, finally, float, goto, implements, import, in, instanceof, int, interface, let, long, native, null, package, private, protected, public, short, static, super, synchronized, throw, throws, transient, true, try, typeof, void, volatile, with, yield	

대표적인 내장 함수 (메소드)		설명 (메소드 이름들은 변수, 함수 이름으로 사용할 수 없음)
alert("텍스트")		확인 버튼이 있는 메시지 창을 띄운다.
prompt("메시지", "기본 텍스트")		텍스트 입력 창이 있는 메시지 창을 띄운다.
confirm("메시지")		확인, 취소 버튼이 있는 메시지 창을 띄운다.
eval("텍스트")		텍스트를 계산할 수 있는 공식으로 바꾼다. 예) var a = eval("3+5"); // a의 값은 8
isNaN()		값이 계산할 수 있는 숫자인지 계산할 수 없는 텍스트인지 구별한다.
parseFloat()		값을 숫자 실수값으로 바꾼다. 예) var a = "12.345"; // parseFloat(a)의 값은 12.345
parseInt()		값을 숫자 정수값으로 바꾼다. 예) var a = "12.345"; // parseInt(a)의 값은 12
Number()		값을 숫자로 바꾼다. 예) var a = "12.345"; // Number(a)의 값은 12.345
String()		값을 텍스트로 바꾼다. 예) var a = 12.345; // String(a)의 값은 "12.345"
Object()		객체를 생성한다. 예) new Object()
Array()		배열을 생성한다. 예) new Array()
setInterval(함수, 반복시간)		해당 함수를 해당 시간 간격으로 반복한다. (밀리초 단위, 1000ms = 1s)
setTimeout(함수, 지연시간)		해당 함수를 해당 시간 이후에 실행시킨다. (밀리초 단위, 1000ms = 1s)
clearInterval()		setInterval()로 만든 타이머를 초기화시킨다.
clearTimeout()		setTimeout()으로 만든 타이머를 초기화시킨다.
close()		현재 창을 닫는다.
reload()		현재 문서를 새로고침한다.
...		

함수의 종류	용법	설명
기본 문법	<pre>function 함수이름(매개변수1, 매개변수2, ...){ /* 스크립트 내용 */ }</pre>	<p>함수는 어떤 일을 실행하기 위해 타이핑된 코드들의 한 덩어리(블록)이다.</p> <p>함수는 누군가가 호출했을 때에만 실행된다.</p>
매개변수가 없는 함수	<pre>function 함수이름() { document.backgroundColor = "orange"; }</pre> <input onclick="함수이름()" type="button" value="Change"/>	<div> <p>이 함수는 배경색을 오렌지색으로 변경시킵니다.</p> <p>배경색이 오렌지색으로 변경 됨</p> </div> <p>함수는 function이라는 키워드와 함수 이름, 괄호()로 선언할 수 있다. 함수를 실행시키면 중괄호{}로 감싸진 함수 내부의 코드들이 실행된다.</p>
매개변수가 있는 함수	<pre>function 함수이름(c) { document.backgroundColor = c; }</pre> <input onclick="함수이름(red)" type="button" value="Change"/>	<div> <p>빨강을 C에 대입</p> <p>이 함수는 배경색을 "C"라는 값으로 변경시킵니다.</p> <p>배경색이 빨강으로 변경 됨</p> </div>
매개변수가 있고 리턴값을 출력하는 함수	<pre>function 함수이름(a, b) { return a * b; }</pre> <input onclick="alert(함수이름(3, 2))" type="button" value="Change"/>	<div> <p>"3"과 "2"를 "a", "b"에 대입</p> <p>이 함수는 a*b의 값을 구하고 리턴값을 되돌려줍니다.</p> <p>호출자에게 6을 보냄</p> </div> <p>이 함수는 리턴값이라는 계산한 결과값을 배출한다. 그리고 그 리턴값은 호출한 요소에게로 되돌려진다.</p>
이름없는 함수	var 변수이름 = function() { /* 스크립트 내용 */ }	함수가 이름을 가지지 않고 변수로 선언될 수도 있다.

함수 호출 방법	용법
이벤트가 생겼을 때 (HTML 태그에 이벤트 핸들러 삽입)	<pre>function 함수이름() { document.backgroundColor = "orange"; }</pre> <div>함수 선언</div> <input onclick="함수이름()" type="button" value="Click here"/> <div>함수 호출</div> <input onclick="document.backgroundColor='orange';" type="button" value="Click here"/>
이벤트가 생겼을 때 (자바스크립트에서 이벤트 핸들러 사용)	<pre>document.getElementById('btn1').onclick = function() { document.backgroundColor = "orange"; }</pre>
자바스크립트 코드에서 직접 호출할 때	<pre>function 함수이름() { document.backgroundColor = "orange"; }</pre> <p>함수이름();</p> <div> <div>함수 선언</div> <div>함수 호출</div> </div>
자동 (스스로 호출 - 윈도우가 열릴 때)	<pre>window.onload = function() { document.backgroundColor = "orange"; }</pre>
이벤트 리스너를 이용	<pre>document.getElementById('myBtn').addEventListener("click", 함수이름); function 함수이름() { document.backgroundColor = "orange"; }</pre> <p>한 개의 요소에 다수의 이벤트를 연결시킬 수 있다. HTML과 자바스크립트를 완벽히 분리시킬 수 있다. (HTML태그에 이벤트 핸들러를 삽입할 필요가 없으므로) removeEventListener() 메소드를 이용하여 차후에 이벤트 리스너를 지울 수 있다. 이벤트 리스너는 이벤트 핸들러의 접두사를 삭제하고 사용한다. ("onclick" → "click")</p>
함수 호출 시 주의사항	<p>함수이름() → 함수의 결과값</p> <p>함수이름 → 함수의 내용 자체</p>

이벤트 핸들러 (Event handler)		설명 (http://www.w3schools.com/jsref/dom_obj_event.asp 참조)		
사 용 법	태그 안에서	<button onclick="스크립트내용"></button>		
	스크립트 안에서	document.frm1.btn1.onclick = function() { /*스크립트내용*/};	이벤트 이용	
		document.frm1.btn1.addEventListener("click", /*스크립트내용*/);	이벤트 리스너 이용	
마우스 이벤트	onclick	오브젝트를 클릭했을 때 (link, button, image, window, form...)		
	ondblclick	오브젝트를 더블클릭했을 때		
	oncontextmenu	오브젝트에서 마우스 오른쪽 버튼을 클릭했을 때 (마우스 오른쪽 버튼 메뉴가 나올 때)		
	onmouseover	마우스 포인터가 오브젝트의 위에 있을 때		
	onmouseout	마우스 포인터가 오브젝트의 위에서 내려왔을 때		
	onmousedown	마우스 버튼으로 오브젝트를 누르고 있는 그 순간		
	onmouseup	마우스 버튼으로 오브젝트를 누르고 있다가 떼는 그 순간		
	onmousemove	마우스 포인터가 움직일 때		
	ondragdrop	마우스 포인터로 오브젝트를 드래그 & 드롭 할 때		
	onwheel	마우스 휠을 움직일 때		
	마우스 이벤트 오브젝트			
	clientX	clientY	마우스 이벤트가 발생했을 때 마우스 포인터의 X/Y축 위치를 반환 (창 기준)	
	pageX	pageY	마우스 이벤트가 발생했을 때 마우스 포인터의 X/Y축 위치를 반환 (문서 기준)	
screenX	screenY	마우스 이벤트가 발생했을 때 마우스 포인터의 X/Y축 위치를 반환 (스크린 기준)		
용법	function showcoords(event) { var x = event.pageX; var y = event.pageY; document.getElementById("demo").innerHTML = x + ", " + y; }			
포커스 이벤트	onfocus	커서가 오브젝트의 안에 위치해 있을 때		
	onblur	커서가 오브젝트의 안에서 벗어났을 때		
키보드 이벤트	onkeydown	키가 눌릴 때		
	onkeypress	키가 눌리고 있는 그 순간		
	onkeyup	키가 눌리고 있다가 떼는 그 순간		
	키보드 이벤트 오브젝트			
	charCode	onkeypress 이벤트가 발생했을 때 눌러진 키의 유니코드 글자 코드(숫자)를 반환 글자 코드를 반대로 글자로 전환할때는 'fromCharCode()' 메소드를 사용함 onkeydown이나 onkeyup 이벤트에서 사용시 항상 0을 반환함		
	keyCode	which	onkeypress 이벤트가 발생했을 때는 눌러진 키의 유니코드 글자 코드를 반환(글자기준) onkeydown, onkeyup이벤트가 발생했을 때는 유니코드 키 코드를 반환 (키보드기준)	
	altKey	키보드 이벤트가 발생했을 때 [Alt]키가 눌렸는지 여부를 반환		
	ctrlKey	키보드 이벤트가 발생했을 때 [Ctrl]키가 눌렸는지 여부를 반환		
	key	키보드 이벤트가 발생했을 때 눌린 키의 이름을 반환 ("a", "A", "4", "\$", "F1", "Enter"...)		
	location	키보드 이벤트가 발생했을 때 눌린 키의 위치를 반환 (0:일반적인 키 / 1:왼쪽 [Ctrl] [Alt] 등 / 2:오른쪽 [Ctrl] [Alt] 등 / 3:숫자패드 키)		
	용법	function myFunction(event) { var x = event.key; document.getElementById("demo").innerHTML = "눌린 키: " + x; }		
폼 이벤트	onsubmit	폼에서 작성한 내용을 전송할 때		
	onreset	폼에서 작성한 내용을 초기화 할 때		
	onselect	텍스트 입력창의 글이나 체크박스, 라디오버튼 등의 항목을 선택했을 때		
	onchange	텍스트 입력창의 글이나 리스트박스의 아이템들이 바뀌었을 때		
윈도우 이벤트	onload	문서가 처음 로드 되었을 때 (열렸을 때)		
	onunload	문서를 닫을 때		
	onmove	브라우저를 움직일 때		
	onresize	브라우저의 크기를 변경할 때		
	onscroll	스크롤바를 움직일 때		
클립보드 이벤트	oncopy	사용자가 페이지 내 요소의 내용을 복사 할 때		
	oncut	사용자가 페이지 내 요소의 내용을 잘라내기 할 때		
	onpaste	사용자가 페이지 내 요소의 내용을 붙여넣기 할 때		

연산자(Operators)	문법	설명
산술 연산자 Arithmetic	A + B	더하기
	A - B	빼기
	A * B	곱하기
	A / B	나누기
	A % B	나머지
	++A	A가 1씩 증가 (A = A + 1)
	--A	A가 1씩 감소 (A = A - 1)
대입(할당) 연산자 Assignment	A = B	B라는 값을 A에 대입하라 (* A와 B가 같다는 뜻으로 쓰이지 않음)
	A += B	A = A + B A와 B를 더한 값을 A에 대입하라
	A -= B	A = A - B A에서 B를 뺀 값을 A에 대입하라
	A *= B	A = A * B A와 B를 곱한 값을 A에 대입하라
	A /= B	A = A / B A와 B를 나눈 값을 A에 대입하라
	A %= B	A = A % B A와 B를 나눈 값의 나머지를 A에 대입하라
비교 연산자 Comparison	A == B	A와 B의 값이 일치 하는가
	A === B	A와 B의 값과 유형이 일치 하는가
	A != B	A와 B의 값이 일치 하지 않는가
	A !== B	A와 B의 값과 유형이 일치 하지 않는가
	A > B	A가 B보다 크다
	A < B	A가 B보다 작으냐
	A >= B	A가 B보다 크거나 같은가
	A <= B	A가 B보다 작거나 같은가
논리 연산자 Logical	A && B	A와 B 둘 다 참일 경우 참을 반환, 하나라도 거짓일 경우 거짓을 반환
	A B	A나 B 중 하나 이상 참일 경우 참을 반환, 둘다 거짓이면 거짓을 반환
	!A	A가 참일 경우 거짓을 반환, A가 거짓일 경우 참을 반환
	(A) ? 값1 : 값2	A가 참일 경우 value1을 채택, A가 거짓일 경우 value2를 채택 예) var nightClub = (age < 20) ? “너무 어림” : “입장 가능함”
기타 연산자	A.B	A라는 부모의 자식 B를 선택
	A[B]	A라는 배열의 값 중 색인 B의 값을 선택 (B+1번째 값)

(if) 문법	흐름도
<pre>if (조건) { codes A; // 조건이 참일 경우 } codes B ;</pre>	<pre> graph TD A{조건} -- 참 --> B[codes A] A -- 거짓 --> C[codes B] </pre>
<pre>if (조건) { codes A; // 조건이 참일 경우 } else { codes B; // 조건이 거짓일 경우 }</pre>	<pre> graph TD A{조건} -- 참 --> B[codes A] A -- 거짓 --> C[codes B] </pre>
<pre>if (조건①) { codes A; // 조건 ①이 참일 경우 } else if (조건②) { codes B; // 조건 ①이 거짓이고 조건 ②가 참일 경우 } else { codes C; // 두 조건 모두 거짓일 경우 }</pre>	<pre> graph TD A{조건①} -- 참 --> B[codes A] A -- 거짓 --> C{조건②} C -- 참 --> D[codes B] C -- 거짓 --> E[codes C] </pre>

(switch / case) 문법	흐름도
<pre>switch (변수) { case value1 : codes A; break; // 변수가 value1의 내용과 같을 경우 case value2 : codes B; break; // 변수가 value2의 내용과 같을 경우 case value3 : codes C; break; // 변수가 value3의 내용과 같을 경우 case value4 : codes D; break; // 변수가 value4의 내용과 같을 경우 default : codes E; // 변수가 어떤 값과도 맞지 않을 경우 }; codes F;</pre>	<pre> graph TD A{변수의 값이} -- "value1과 같을 경우" --> B[codes A] A -- "value2와 같을 경우" --> C[codes B] A -- "value3과 같을 경우" --> D[codes C] A -- "value4와 같을 경우" --> E[codes D] A -- "어떤 값과도 맞지 않을 경우" --> F[codes E] B --> G[codes F] C --> G D --> G E --> G F --> G </pre>
예시	<pre>var x = 1; var y = 2; switch (x+y) { case 1 : alert("one"); break; case 2 : alert("two"); break; case 3 : alert("three"); break; case 4 : alert("four"); break; default : alert("알?"); }; alert("end!");</pre> <div style="display: flex; justify-content: space-around; align-items: center;"> </div>

(for) 문법			흐름도
for (시작값; 조건식; 증감식) { codes; }			
예시	<pre>var x = 0; for (i=0; i<10; i++) { x = x + "1"; }; // 10번 반복함 alert(x);</pre>	결과 0111111111	
시작값 > 조건식확인 > 블록실행 > 증감식실행 > 조건식확인 >...			
(while) 문법			
시작값; while (조건식) { codes; 증감식; }			
예시	<pre>var i = 0; var x = 0; while (i<10) { x = x + "1"; i++; } alert(x);</pre>	결과 0111111111	
조건식확인 > 블록실행 > 조건식확인 >...			
(do ~ while) 문법			흐름도
시작값; do { codes; 증감식; } while (조건식);			
예시	<pre>var i = 0; var x = 0; do { x = x + "1"; i++; } while (i<10); alert(x);</pre>	결과 0111111111	
블록실행 > 조건식확인 > 블록실행 > 조건식확인 > ... (처음에 일단 한번 블록을 실행하고 반복 시작함)			
상황에 따른 반복문 제어			흐름도
break	<pre>var text = ""; var i; for (i = 0; i < 10; i++) { if (i == 3) { break; } text += i; }</pre>	결과 012	
	결과	012	
continue	<pre>var text = ""; var i; for (i = 0; i < 10; i++) { if (i == 3) { continue; } text += i; }</pre>	결과 012456789	
	결과	012456789	

- 9 -

```

window.screen.속성;
screen.속성;
screen.속성 = 값;

```

screen 객체의 속성	설명
availHeight	윈도우 시작표시줄을 제외한 디바이스 화면의 높이
availWidth	윈도우 시작표시줄을 제외한 디바이스 화면의 너비
height	디바이스 화면의 전체 높이
width	디바이스 화면의 전체 너비

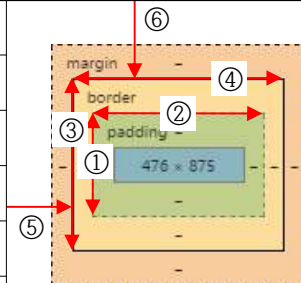
```

document.속성;
document.속성 = 값;
document.메소드();
document.메소드(매개변수);

```

document 객체의 속성	설명										
body	문서의 body태그										
title	문서의 제목										
URL	문서의 전체 주소 "http://www.naver.com/index.html"										
domain	서버의 도메인 이름 "www.naver.com"										
innerHTML	문서 내 특정 태그 안에 들어갈 내용 예) document.getElementById("idName").innerHTML = 5 + 6;										
readyState	현재 문서의 로딩 상태 <table> <tr> <td>uninitialized</td><td>아직 로딩이 시작되지 않음</td></tr> <tr> <td>loading</td><td>로딩 중</td></tr> <tr> <td>loaded</td><td>로딩 된 적이 있음</td></tr> <tr> <td>interactive</td><td>사용자가 상호작용할 만큼 충분히 로딩이 됨</td></tr> <tr> <td>complete</td><td>완전히 로딩 됨</td></tr> </table>	uninitialized	아직 로딩이 시작되지 않음	loading	로딩 중	loaded	로딩 된 적이 있음	interactive	사용자가 상호작용할 만큼 충분히 로딩이 됨	complete	완전히 로딩 됨
uninitialized	아직 로딩이 시작되지 않음										
loading	로딩 중										
loaded	로딩 된 적이 있음										
interactive	사용자가 상호작용할 만큼 충분히 로딩이 됨										
complete	완전히 로딩 됨										

document 객체의 메소드	설명
clear()	문서를 지운다.
open()	문서를 연다.
close()	"open()" 메소드로 열린 문서를 닫는다.
write("텍스트")	문서에 텍스트를 기입한다. (태그포함)
writeln("텍스트")	문서에 텍스트를 기입한다. (태그 및 태그 포함)
getElementById("아이디이름")	문서에서 "아이디이름" 아이디를 검색하여 해당 요소를 선택한다.
getElementsByTagName("태그이름") [색인번호]	문서에서 "태그이름" 태그 전체를 검색하여 해당 색인번호 순서의 태그를 선택한다.
getElementsByClassName("클래스이름") [색인번호]	문서에서 "클래스이름" 클래스 전체를 검색하여 해당 색인번호 순서의 클래스를 선택한다.
getElementsByName("네임이름") [색인번호]	문서에서 "네임이름" 네임 전체를 검색하여 해당 색인번호 순서의 네임을 선택한다.
querySelector("CSS선택자")	문서에서 CSS 선택자를 통해 첫 번째 해당 요소를 선택한다. 예) document.querySelector(".aaa > div");
querySelectorAll("CSS선택자") [색인번호]	문서에서 CSS 선택자를 통해 해당 요소 전체를 선택한다. 예) document.querySelectorAll(".aaa > div")[0];
hasFocus()	문서에 포커스가 위치하고 있는지 여부를 판단한다. (true / false)
blur()	문서에서 포커스를 제거한다.
focus()	문서에 포커스를 위치시킨다.

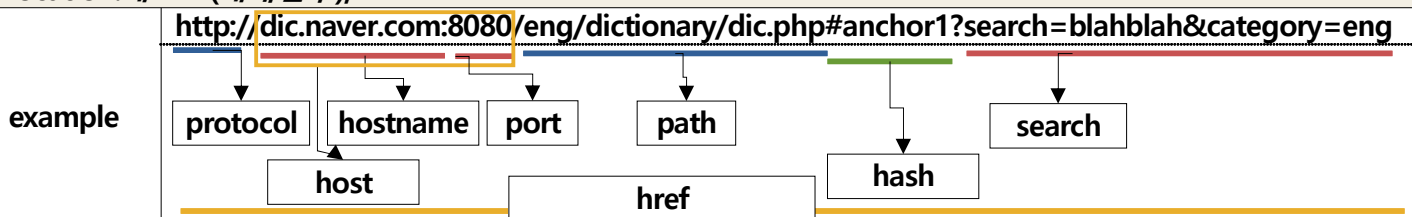
element.속성; element.속성 = 값 ; element.메소드(); element.메소드(매개변수);	모든 HTML 객체, 속성, document, 태그 안의 text, 코멘트 등(node)이 element객체가 될 수 있다.	
element 객체의 속성	설명	
id	요소의 아이디 속성 값	
attributes	요소에 적용된 속성	
className	요소에 연결된 클래스	
clientHeight	패딩을 포함한 요소의 높이 ①	
clientWidth	패딩을 포함한 요소의 너비 ②	
offsetHeight	패딩, 보더, 스크롤바를 포함한 요소의 높이 (마진은 포함하지 않음) ③	
offsetWidth	패딩, 보더, 스크롤바를 포함한 요소의 너비 (마진은 포함하지 않음) ④	
offsetLeft	부모의 안쪽으로부터 요소의 마진을 포함한 왼쪽 경계까지의 거리 ⑤	
offsetTop	부모의 안쪽으로부터 요소의 마진을 포함한 위쪽 경계까지의 거리 ⑥	
scrollHeight	패딩을 포함한 총 높이 (overflow 속성으로 감춰진 부분까지의 길이)	
scrollWidth	패딩을 포함한 총 너비 (overflow 속성으로 감춰진 부분까지의 길이)	
scrollLeft	왼쪽으로부터 스크롤 된 거리	
scrollTop	위쪽으로부터 스크롤 된 거리	
innerHTML	요소의 내용	예) 요소의 내용
accessKey	요소에 단축키를 지정 예) element.getElementById("homeBtn").accessKey = "h"; (Alt+h 키를 누르면 해당 요소가 실행 됨)	
style	CSS 스타일을 지정 예) element.style.borderBottom = "1px solid blue";	
tagName	요소의 태그 이름 (대문자로 반환 됨)	
title	요소의 타이틀 문구 (툴팁 표시 문구)	
length	요소가 가진 노드들의 개수	

element 객체의 메소드	설명	
<code>addEventListener("이벤트", function(){ 코드 })</code>	요소에 이벤트 추적자를 연결하여 이벤트가 발생했을 때 실행시킬 코드를 적용시킨다. (접두사 제거)	
<code>removeEventListener("이벤트")</code>	요소에 적용된 이벤트 핸들러를 지운다.	
<code>blur()</code>	요소에서 포커스를 제거한다.	
<code>focus()</code>	요소에 포커스를 위치시킨다.	
<code>cloneNode(deep)</code>	요소를 복제한다. (deep - true: 요소와 요소의 자식들도 복제 / false: 기본값, 노드만 복제)	
<code>setAttribute("속성명", "속성값")</code>	요소에 해당 속성과 값을 입력한다.	
<code>getAttribute("속성명")</code>	요소가 가진 속성의 값을 읽어온다.	
<code>removeAttribute("속성명")</code>	요소가 가진 해당 속성을 삭제한다.	
<code>hasAttribute("속성명")</code>	요소가 해당 속성을 가지고 있는지 여부를 판단한다. (true / false)	
<code>hasAttributes()</code>	요소가 어떤 속성이라도 가지고 있는지 여부를 판단한다. (true / false)	
<code>getElementsByName("태그이름")</code>	요소에서 "태그이름" 태그 전체를 검색하여 해당 색인번호 순서의 태그를 선택한다.	
<code>getElementsByClassName("클래스이름")</code>	요소에서 "클래스이름" 클래스 전체를 검색하여 해당 색인번호 순서의 클래스를 선택한다.	
<code>querySelector("CSS선택자")</code>	요소에서 CSS 선택자를 통해 첫 번째 해당 요소를 선택한다. 예) <code>document.querySelector(".aaa > div");</code>	
<code>querySelectorAll("CSS선택자")</code>	요소에서 CSS 선택자를 통해 해당 요소 전체를 선택한다. 예) <code>document.querySelectorAll("aaa > div");</code>	

attr. 속성;	모든 HTML 속성 객체가 attribute 객체가 될 수 있다.	
attribute 객체의 속성	설명	<pre><button id="button1" onclick="myFunction()">Push</button> <script>var btn = document.getElementById("button1");</script></pre>
<code>attributes[색인번호].name</code>	현재 속성의 이름	<code>var x = btn.attributes[0].name;</code> id
<code>attributes[색인번호].value</code>	현재 속성의 값	<code>var y = btn.attributes[0].value;</code> button1
<code>attributes.length</code>	현재 속성의 노드 개수	<code>var z = btn.attributes.length;</code> 2

객체.style 속성		※ CSS 정리 문건 참조	
style 객체의 속성	설명	style 객체의 속성	설명
animation	css 애니메이션 관련 속성 (animationDelay, animationDirection, animationDuration, animationFillMode, animationIterationCount, animationName, animationTimingFunction, animationPlayState)	background	배경그라운드 관련 속성 (backgroundAttachment, backgroundColor, backgroundImage, backgroundPosition, backgroundRepeat, backgroundClip, backgroundOrigin, backgroundSize)
border borderTop/borderBottom borderLeft/borderRight	외곽선 속성 (borderCollapse, borderColor, borderRadius, borderSpacing, borderStyle, borderWidth)	top/bottom/left/right	position 속성 (relative/absolute/fixed/..)이 들어간 요소의 해당 위치 속성
boxShadow	박스의 그림자 속성	boxSizing	박스의 가로/세로 사이즈 렌더링에 관한 속성
clear	float된 요소들의 포지션 속성	clip	요소에서 마스킹할 부분에 관한 속성
color	텍스트 색깔 속성	cursor	마우스 포인터 속성
direction	텍스트 방향 속성	display	요소의 디스플레이 타입 속성
margin marginTop/marginBottom marginLeft/marginRight	바깥 여백 속성	font	텍스트의 폰트 속성 (fontFamily, fontSize, fontStyle, fontVariant, fontWeight)
width/height	요소의 가로/세로 크기 속성	lineHeight	글줄 높이(간격) 속성
maxWidth/maxHeight	요소의 최대 가로/세로 크기 속성	cssFloat	요소의 수평 정렬 속성 (float)
minWidth/minHeight	요소의 최소 가로/세로 크기 속성	opacity	요소의 투명도 속성
overflow overflowX/overflowY	넘치는 내용 표시에 관한 속성	padding paddingTop/paddingBottom paddingLeft/paddingRight	안쪽 여백 속성
position	포지셔닝 방법에 관한 속성	textAlign	텍스트 정렬 속성
textDecoration	텍스트 꾸미기 속성	textIndent	텍스트 들여쓰기 속성
transform	2D, 3D 변형 속성	transition	전환 효과 속성
visibility	요소 숨기기 속성	verticalAlign	인라인 요소의 세로 정렬 속성
zIndex	포지션 속성이 들어간 요소의 중첩 순서 속성	whiteSpace	텍스트 줄바꿈 속성
기타	letterSpacing / wordSpacing / listStyle / textOverflow / textShadow / textTransform / userSelect /		

location.속성;
location.속성 = 값;
location.메소드();
location.메소드(매개변수);



location 객체의 속성	설명
href	전체 URL 주소
protocol	프로토콜의 종류
host	URL과 host name, port를 포함한 주소
hostname	도메인 이름(IP)
pathname	URL의 path 부분
port	포트 번호
hash	anchor 부분(#)의 이름
search	"?" 이후의 부분(쿼리 문자열)

location 객체의 메소드	설명
reload()	현재 페이지를 새로고침 한다.
assign("URL")	새 문서를 로딩 한다.
replace("새URL")	새 문서를 로딩 한다. (history에서 현재 문서의 URL을 삭제하기 때문에 뒤로 가기가 불가능함)

**navigator.속성;
navigator.메소드();**

navigator 객체의 속성	설명
appName	브라우저의 코드네임
appName	브라우저의 종류
appVersion	브라우저의 버전
userAgent	브라우저의 코드네임, 버전 및 OS 정보
platform	시스템 플랫폼 코드
navigator 객체의 메소드	설명
javaEnable()	자바스크립트의 사용 가능 여부를 판단한다.
taintEnable()	문서가 제대로 열렸는지 여부를 판단한다.

**history.속성;
history.메소드();
history.메소드(매개변수);**

history 객체의 속성	설명
length	현재 브라우저 히스토리의 링크 개수
history 객체의 메소드	설명
back()	이전 페이지로 이동한다.
forward()	다음 페이지로 이동한다.
go(n)	단계 만큼(n) 페이지 이동한다.
go(0)	페이지를 새로고침 한다.
go(1)	다음 페이지로 이동한다.
go(-1)	이전 페이지로 이동한다.

var 배열이름 = new Array(); 또는 var 배열이름 = [];**배열이름.속성;
배열이름.메소드();**

Array 객체의 속성	설명
length	배열에 저장된 총 데이터의 개수
Array 객체의 메소드	설명
reverse()	배열의 값들의 순서를 거꾸로 정렬한다.
sort()	배열의 값들을 오름차순으로 정렬한다.
pop()	배열의 마지막 색인번호에 저장된 값을 삭제한다.
shift()	배열의 첫 번째 색인번호에 저장된 값을 삭제한다.
push(새값, 새값2, 새값3...)	배열의 마지막 색인번호에 새 값을 삽입한다.
unshift(새값)	배열의 첫 번째 색인번호에 새 값을 삽입한다.
fill(값, 시작 색인번호, 끝 색인번호)	해당 값으로 배열 내의 값들을 교체한다. (색인번호 생략시, 배열 내의 모든 값들을 교체함)
forEach(함수)	배열 내의 각각의 값에 함수를 대입하여 각각 실행시킨다.
map(함수)	배열 내의 각각의 값에 함수를 대입하여 그 결과값으로 새 배열을 만든다.
reduce(함수)	배열 내의 각각의 값에 함수를 대입하여 하나의 결과값을 만든다. (배열이 아니라 일반 변수로)
indexOf("텍스트")	배열 내의 값 중 해당 텍스트와 일치하는 첫 번째 값의 색인번호를 찾는다. (못찾은 경우 -1을 반환)
lastIndexOf("텍스트")	배열 내의 값 중 해당 텍스트와 일치하는 마지막 값의 색인번호를 찾는다. (못찾은 경우 -1을 반환)
Array.isArray(객체이름)	객체가 배열인지 판단한다.
기존배열이름.concat(다른배열이름, ...)	2개 이상의 배열 객체를 하나로 결합시킨다.
join("연결텍스트")	배열의 값들을 연결텍스트를 삽입하여 1개의 문자형 데이터로 반환한다.
slice(시작 색인번호, 끝 색인번호)	배열의 값들 중 원하는 색인번호의 구간만큼 잘라서 새 배열을 반환한다.
splice(시작 색인번호, 지울 개수, 새값1, 새값2,...)	배열의 지정된 값을 삭제하거나 그 구간에 새 값을 삽입한다.

Number.속성; Number.메소드(); Number.메소드(매개변수);	
Number 객체의 속성	설명
MAX_VALUE	자바스크립트에서 사용할 수 있는 가장 큰 수 (1.7976931348623157e+308)
MIN_VALUE	자바스크립트에서 사용할 수 있는 가장 작은 수 (5e-324) (이보다 작은 수는 0)
POSITIVE_INFINITY	양수의 무한대 (MAX_VALUE보다 크다)
NEGATIVE_INFINITY	음수의 무한대 (그 어느 숫자보다 작다)
Number 객체의 메소드	설명
isInteger()	값이 정수인지 판단한다. (소수, 문자, 논리값, 무한대인 경우 \Rightarrow false)
isNaN()	값이 불가산 숫자인지 판단한다. (NaN, 0/0, 문자열 \Rightarrow true / 숫자, 문자(숫자로 이루어진), 논리값, undefined \Rightarrow false)
toFixed(소수점자리수)	값을 해당 소수점 자리수 만큼 반올림하고 이를 문자열로 바꾼다.
toPrecision(자리수)	값을 앞쪽으로부터 해당 자리수 만큼 남기고 반올림하고 이를 문자열로 바꾼다.
toString(진수)	값을 문자열로 바꾼다. (진수를 입력하면 해당 진수로 변환한다.)
숫자와 관련된 전역 메소드	설명
Number(값)	값을 숫자로 바꾼다.
parseInt(값)	값을 정수값으로 바꾼다.
parseFloat(값)	값을 실수값으로 바꾼다.

Math.속성; Math.메소드(); Math.메소드(매개변수);	
Math 객체의 속성	설명
PI	파이 원주율 (3.141592653589793)
SQRT2	$\sqrt{2}$ (1.4142135623730951)
Math 객체의 메소드	설명
random()	0과 1 사이의 무작위 실수를 반환한다. 50에서 100까지의 무작위 정수 구하는 법 <div> $\text{Math.floor}(\text{Math.random()} * (100 - 50) + 50);$ $\text{Math.floor}(\text{Math.random()} * (\text{최대값} - \text{최소값}) + \text{최소값});$ </div>
max(a, b, c, ...)	값들 중 가장 큰 값을 반환한다.
min(a, b, c, ...)	값들 중 가장 작은 값을 반환한다.
round(값)	값의 수를 반올림한 값을 반환한다.
ceil(값)	값의 수를 올림한 값을 반환한다.
floor(값)	값의 수를 버림한 값을 반환한다.
abs(값)	값의 절대값을 반환한다.
pow(X, Y)	X의 Y승 값을 반환한다. (X^Y)
sin(라디안)	

var 변수이름 = "텍스트"; 변수이름.속성; 변수이름.메소드(); "텍스트".메소드();		
string 객체의 속성	설명	
length	글자의 개수(공백 포함)	
string 객체의 메소드	설명	
indexOf("텍스트")	문자열 중 해당 텍스트의 위치를 찾는다. (왼쪽 첫 글자 부터 0으로 시작함, null=-1)	
	예시	결과
	<pre>var aaa = "abcdefg".indexOf("d"); var arr = ["apple", "orange", "banana", "nut"]; arr.indexOf("orange");</pre>	<pre>3 1</pre>
indexOf("텍스트", n)	문자열의 n번째 글자부터 해당 텍스트의 위치를 찾는다. (왼쪽 첫 글자 부터 0으로 시작함, null=-1)	
	<pre>var aaa = "abcdeabced".indexOf("b", 3);</pre> <p>0 ↦ 6</p>	6
lastIndexOf("텍스트")	문자열 중 왼쪽 방향으로 해당 텍스트의 위치를 찾는다. (왼쪽 첫 글자 부터 0으로 시작함, null=-1)	
	<pre>var aaa = "abcdeabced".lastIndexOf("a");</pre> <p>0 5 ←</p>	5
lastIndexOf("텍스트", n)	문자열의 n번째 글자부터 왼쪽 방향으로 해당 텍스트의 위치를 찾는다. (왼쪽 첫 글자 부터 0으로 시작함, null=-1)	
	<pre>var aaa = "abcdeabced".lastIndexOf("c", 5);</pre> <p>0 2 ←</p>	2
charAt(n)	n번째 텍스트를 찾는다.	
	<pre>var aaa = "abcdefg".charAt(3);</pre>	d
substring(n, m)	n번째 부터 m번째의 전까지의 텍스트를 찾는다.	
	<pre>var aaa = "abcdefg".substring(3, 6);</pre> <p>↘ ↙</p>	def
slice(n, m)	substring()과 비슷하나, 음수를 사용하여 오른쪽부터 탐색이 가능하다.	
substr(n, m)	n번째부터 m글자만큼 텍스트를 찾는다.	
	<pre>var aaa = "abcdefg".substr(3, 4);</pre> <p>↘ →→</p>	defg
split("분할텍스트")	분할텍스트를 기준으로 좌우로 string 객체를 분할한다. (분할되어 값이 2개가 되었으므로 배열 객체로 간주됨)	
	<pre>var aaa = "abcdefg".split("cd");</pre>	ab, efg
	aaa[1];	efg
concat("텍스트")	string 객체에 텍스트를 붙인다.	
	<pre>var aaa = "abcdefg".concat("zzz");</pre>	abcdefgzzz
toUpperCase()	모든 글자들을 대문자로 전환한다.	
toLowerCase()	모든 글자들을 소문자로 전환한다.	
eval()	텍스트를 수학 계산이 가능한 공식으로 전환한다.	
	<pre>var aaa = eval("3+5");</pre>	8
toString()	객체나 배열객체를 텍스트로 전환한다.	
	<pre>var arr = ["apple", "orange", "banana", "nut"]; arr.toString();</pre>	apple,orange,banana,nut
toString(n)	숫자를 n진수로 전환한다.	
	<pre>var aaa = 5; var bbb = aaa.toString(2);</pre>	101
search("텍스트")	해당 텍스트와 동일한 패턴을 찾고 해당 텍스트의 위치를 찾는다.	
	<pre>var aaa = "abcdefg".search("cd");</pre>	2
replace(a, b)	"a"를 찾아서 그 내용을 "b"로 바꾼다.	
	<pre>var aaa = "abcdefg".replace("cd", "뽕뽕");</pre>	ab뽕뽕efg
match(정규표현식)	정규표현식을 사용하여 해당 텍스트와 동일한 패턴을 찾고 배열로 반환한다.	
	<pre>var aaa = "abcdefgabcdABCD".match(/cd/g);</pre>	cd, cd


```

var 변수이름 = new Date(); /* 페이지가 로딩된 시점의 시간 */
var 변수이름 = new Date(1486085558000); /* 1970년 1월 1일 0시 0분 이후 지난 시간으로 설정(ms) */
var 변수이름 = new Date(2017, 1, 3, 10, 14, 20, 0); /* 2017년 2월 3일 10시 14분 20초 0ms로 설정 */
var 변수이름 = new Date(2017, 1, 3); /* 2017년 2월 3일로 설정 */
var 변수이름 = new Date("2017-02-03"); /* 2017년 2월 3일로 설정 */
var 변수이름 = new Date("2017-02-03T10:14:20Z"); /* 2017년 2월 3일 10시 14분 20초로 설정 */
var 변수이름 = new Date("February 03, 2017 10:14:20"); /* 2017년 2월 3일 10시 14분 20초로 설정 */
변수이름.메소드();

```

Date 객체의 메소드	설명
getFullYear()	1900년부터 현재까지의 연수를 계산한다. (2017년 기준 → 117)
getFullYear()	현재 연도를 네 자리 숫자로 반환한다. (yyyy)
getMonth()	현재 월을 숫자로 반환한다. (0~11) (1월 → 0 / 12월 → 11)
getDate()	현재 일을 숫자로 반환한다. (1~31)
getDay()	현재 요일을 숫자로 반환한다. (0~6) (일요일 → 0 / 토요일 → 6)
getHours()	현재 시간을 숫자로 반환한다. (0~23)
getMinutes()	현재 분을 숫자로 반환한다. (0~59)
getSeconds()	현재 초를 숫자로 반환한다. (0~59)
getMilliseconds()	현재 밀리초를 숫자로 반환한다. (0~999)
getTime()	1970년 1월 1일 0시부터 정해진 시간까지의 차이를 밀리초(ms)로 계산한다.
Date.now()	1970년 1월 1일 0시부터 현재까지의 시간을 밀리초(ms)로 계산한다.
setFullYear(연도, 월, 일)	특정 날짜로 시간을 설정한다.
setMonth(월)	특정 월로 시간을 설정한다. (0~11)
setDate(일)	특정 날짜로 시간을 설정한다. (0~31)
setHours(시간)	특정 시로 시간을 설정한다. (0~23)
setMinutes(분)	특정 분으로 시간을 설정한다. (0~59)
setSeconds(초)	특정 초로 시간을 설정한다. (0~59)
setTime(밀리초)	1970년 1월 1일 0시부터 지나온 시간을 밀리초(ms)로 설정한다.

/텍스트패턴/ 변경자,		http://regexper.com/ http://regexpr.com/ 참조	
정규표현식의 변경자	설명		
i	대소문자 구별안함	/abc/	ABCabcABCabc
		/abc/i	ABCabcABCabc
g	첫 번째 패턴 발견 이후에도 라인 전체에서 계속 패턴을 찾음	/abc/	ABCabcABCabc
		/abc/g	ABCabcABCabc
m	각각의 라인에서 시작문자(^)와 끝문자(\$)를 적용할 수 있음 항상 g변경자와 함께 씀(연속 탐색을 하지 않으면 의미 없으므로)	/abc\$/g	ABCabcABCabc ABCabcABCabc
		/abc\$/gm	ABCabcABCabc ABCabcABCabc

정규표현식의 패턴	설명		
abc	글자와 일치하는 부분을 찾는다.	/Lorem/g	Lorem ipsum 2dolor 7sit 9amet%
[abc]	대괄호 안에 있는 글자와 일치하는 글자를 찾는다.	/[abcd]/g	Lorem ipsum 2dolor 7sit 9amet%
		/[a-z]/g	Lorem ipsum 2dolor 7sit 9amet%
[123]	대괄호 안에 있는 숫자와 일치하는 숫자를 찾는다.	/[27]/g	Lorem ipsum 2dolor 7sit 9amet%
		/[0-9]/g	Lorem ipsum 2dolor 7sit 9amet%
[^abc]	대괄호 안에 있는 글자와 일치하지 않는 글자를 찾는다.	/[^abcd]/g	Lorem ipsum 2dolor 7sit 9amet%
[^123]	대괄호 안에 있는 숫자와 일치하지 않는 숫자를 찾는다.	/[^123]/g	Lorem ipsum 2dolor 7sit 9amet%
(x y)	소괄호 안에 있는 글자 중 x 또는 y와 일치하는 글자를 찾는다.	/(L l)/g	Lorem ipsum 2dolor 7sit 9amet%
.	아무 한 글자를 찾는다.	/2.../g	Lorem ipsum 2dolor 7sit 9amet%
\d	숫자를 찾는다.	/\d/g	Lorem ipsum 2dolor 7sit 9amet%
\s	공백을 찾는다.	/\s/g	Lorem ipsum 2dolor 7sit 9amet%
\w	글자를 찾는다. (a-z, A-Z, 0-9, _)	/\w/g	Lorem ipsum 2dolor 7sit 9amet%
\b	단어의 범위를(시작과 끝) 지정한다.	/\bipsum\b/g	Lorem ipsum 2dolor 7sit 9amet%
\D	숫자가 아닌 부분을 찾는다.	/\D/g	Lorem ipsum 2dolor 7sit 9amet%
\S	공백이 아닌 부분을 찾는다.	/\S/g	Lorem ipsum 2dolor 7sit 9amet%
\W	글자가 아닌 부분을 찾는다.	/\W/g	Lorem ipsum 2dolor 7sit 9amet%
\B	단어의 범위를 지정하지 않는다.	/\bsum\b/g	Lorem ipsum 2dolor 7sit 9amet%
n+	해당 글자가 한 개 이상 포함된 부분을 찾는다.	/lo+/g	Helloo World! Hello Daewoo Vocational school!
n*	해당 글자가 0 개 이상 포함된 부분을 찾는다.	/lo*/g	Helloo World! Hello Daewoo Vocational school!
n?	해당 글자가 없거나 한 개만 포함된 부분을 찾는다.	/lo?/g	Helloo World! Hello Daewoo Vocational school!
n{x}	해당 글자가 x개 만큼 반복된 부분을 찾는다.	/o{2}/g	Helloo World! Hello Daewoo Vocational school!
n{x, y}	해당 글자가 x개에서 y개 만큼 반복된 부분을 찾는다.	/o{2,3}/g	Helloo World! Hello Daewoo Vocational school!
n{x,}	해당 글자가 x개 이상 반복된 부분을 찾는다.	/o{2,}/g	Helloo World! Hello Daewoo Vocational school!
^n	라인에서 해당 글자로 시작하는 부분을 찾는다.	/^Hel/g	Helloo World! Hello Daewoo Vocational school!
n\$	라인에서 해당 글자로 끝나는 부분을 찾는다.	/\W\$/g	Helloo World! Hello Daewoo Vocational school!
?=n	뒤에 해당 글자가 붙는 부분을 찾는다.	/l(?!o)/g	Helloo World! Hello Daewoo Vocational school!
?!n	뒤에 해당 글자가 붙지 않는 부분을 찾는다.	/l(?!o)/g	Helloo World! Hello Daewoo Vocational school!

정규표현식의 속성과 메소드	설명	<pre>var text = "Lorem ipsum ipsum"; var pattern = /sum/g; // 정규표현식객체 var result = text.match(pattern);</pre>	
source	정규표현식의 텍스트패턴 부분	pattern.source;	sum (문자열)
match(정규표현식)	정규표현식으로 텍스트를 검색하여 배열을 반환한다.	text.match(pattern);	sum,sum (객체)
정규표현식객체.exec(텍스트)	정규표현식으로 첫 번째 텍스트를 검색하여 배열로 반환한다. 검색하지 못할 경우 null을 반환한다.	pattern.exec(text);	sum (객체)
정규표현식객체.test(텍스트)	정규표현식으로 텍스트를 검색하여 맞는 텍스트가 있을 경우 true를, 없을 경우 false를 반환한다.	pattern.test(text);	true (논리형)
정규표현식객체.toString()	정규표현식의 내용을 글자로 반환한다.	pattern.toString();	/sum/g (문자열)