

Javascript

1. 자바스크립트 기초

1) 자바스크립트의 정의

클라이언트쪽에서 독립적으로 실행되는 프로그램을 작성하기 위한 스크립트 언어
넷스케이프 사의 브랜든 아이히 **Brendan Eich**에 의해 모카라는 이름으로 만들고 이후 라이브 스크립트라는 이름으로 개발

썬마이크로시스템사와 공동으로 라이브스크립트를 확장한 **JavaScript** 탄생

2) 자바스크립트의 특징

- 웹 문서(HTML)에 삽입해서 사용하는 스크립트 언어
- 웹 브라우저에서 웹 문서를 실행할 때 프로그램 코드가 해석됨
- 컴파일 과정을 거치지 않는 인터프리터 언어의 형태이기 때문에 비교적 자료형 조사를 철저하게 하지 않음
- 객체 지향적 특성을 모두 가지고 있다고 말할 수는 없지만 객체를 정의하여 사용할 수는 있음.

3) 자바스크립트의 장점 및 단점

(1) 장점

- 자바스크립트는 **HTML** 파일 내에서 작성할 수 있으므로 개발 속도가 빠름
- 운영체제의 제한을 받지 않음

(2) 단점

- 소스 코드가 노출됨. 컴파일하지 않는 언어이므로 복사하여 사용할 수 있음
- 한정된 객체와 객체 함수 제공

4) 자바스크립트(**ECMAScript**)의 표준화

브라우저에 따라 웹페이지가 정상적으로 동작하지 않는 크로스 브라우징 이슈가 발생.
자바스크립트의 파편화를 방지하고 모든 브라우저에서 정상적으로 동작하는 표준화된 자바스크립트의 필요성 대두. **1996년 11월**, 넷스케이프 커뮤니케이션즈는 컴퓨터 시스템의 표준을 관리하는 비영리 표준화 기구인 **ECMA** 인터내셔널에 자바스크립트 표준화 요청

ECMAScript는 자바스크립트의 표준 사양인 **ECMA-262**를 말하며, 프로그래밍 언어의 값, 타입, 객체와 프로퍼티, 함수, 표준 빌트인 객체(**standard built-in object**) 등 핵심 문법을

규정한다. 각 브라우저 제조사는 ECMAScript 사양을 준수해서 브라우저에 내장되는 자바스크립트 엔진을 구현한다.

버전	출시연도	특징
ES1	1997	초판
ES2	1998	ISO/IEC 16262 국제 표준과 동일한 규격을 적용
ES3	1999	정규 표현식, try... catch
ES5	2009	HTML5와 함께 출현한 표준안.
ES6(ECMAScript 2015)	2015	let/const, 클래스, 화살표 함수, 템플릿 리터럴, 디스트럭처링 할당, 스프레드 문법, Rest 파라미터, 심벌, 프로미스, Map/Set, 이터러블, for...of, 제너레이터, Proxy, 모듈 import/export
ES7(ECMAScript 2016)	2016	지수(**) 연산자, Array.prototype.includes, String.prototype.includes
ES8(ECMAScript 2017)	2017	async/await, Object 정적 메서드(Object.values, Object.entries, Object.getOwnPropertyDescriptors)
ES9(ECMAScript 2016)	2018	Object rest/spread 프로퍼티, Promise.prototype.finally, async generator, for await...of
ES10(ECMAScript 2019)	2019	Object.fromEntries, Array.prototype.flat, Array.prototype.flatMap, optional catch binding
ES11(ECMAScript 2020)	2020	String.prototype.matchAll, BigInt.globalThis, Promise.allSettled, null 병합 연산자, 옵셔널 체이닝 연산자, for...in enumeration order

5) 개발자 도구

크롬 브라우저가 제공하는 개발자 도구는 웹 애플리케이션 개발에 필수적인 강력한 도구다.

패널	설명
Elements	로딩된 웹페이지의 DOM과 CSS를 편집해서 렌더링된 뷰를 확인해 볼 수 있다. 단, 편집한 내용이 저장되지는 않는다. 웹페이지가 의도된 대로 렌더링되지 않았다면 이 패널을 확인해 유용한 힌트를 얻을 수 있다.
Console	로딩된 웹페이지의 에러를 확인하거나 자바스크립트 소스코드에 작성한 console.log 메서드의 실행 결과를 확인할 수 있다.

Source	로딩된 웹페이지의 자바스크립트 코드를 디버깅할 수 있다.
Network	로딩된 웹페이지에 관련된 네트워크 요청 정보와 성능을 확인할 수 있다.
Application	웹 스토리지, 세션, 쿠키를 확인하고 관리할 수 있다.

2. 자바스크립트의 기본 구조

1) 기본구조

<script>태그 안에서 코드 표시

```
<script type="text/javascript">
    document.write('head에서 실행<br>');
</script>
```

2) 자바스크립트 실행 위치

- <head>태그에 <script> 태그 명시

<head> 태그안에 <script> 태그를 명시하면 <body> 태그가 동작하기 전에 자바스크립트 코드가 실행됨

- <body> 태그에 <script> 태그 명시

<body> 태그 안에 <script> 태그를 명시하면 <head> 태그가 동작한 이후에 실행됨.

- 자바스크립트를 외부 파일로 사용

```
<script type="text/javascript" src="자바스크립트파일명.js"></script>
```

- HTML 태그에 인라인(inline) 형태로 삽입해서 사용

```
<input type="button" value="이동" onclick="location.href='index.html'">
```

3) 자바스크립트 출력

```
document.write('자바스크립트 출력 구문');
```

4) 자바스크립트 주석

```
// : 한줄 주석 처리  
/* ~ */ : 한줄 이상의 주석 처리
```

3. 변수와 자료형

변수 : 프로그래밍에서 데이터를 담을 수 있는 메모리 할당 영역

1) 변수명 지정 규칙

- 자바스크립트의 예약어는 사용할 수 없음(if, true, false, break, null 등).
- 영문자 혹은 밑줄(_)로 시작해야 하며, 숫자로 시작할 수 없음.
- 문자의 대문자(A~Z), 소문자(a~z), 숫자(0~9), 밑줄만 사용 가능.

```
var num = 45;  
var str = 'Hello';
```

* 자바스크립트 예약어(키워드)

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

double	import	public	let
--------	--------	--------	-----

2) 자료형(Data Type)

자바스크립트는 코드상에 자료형을 표시하지 않지만 내부적으로 데이터를 인식할 때 사용됨

<ul style="list-style-type: none"> - string 문자열 : “ 또는 ‘ 사이에 들어가는 문자들 ex) var str = ‘오전’; - number 숫자 : 정수형과 실수형으로 나눌 수 있음 ex) var a = 3; var b = 23.45 - boolean 논리 : 참(true)와 거짓(false)를 표현 ex) var a = true - 함수 : 함수 - 객체 : 객체 - undefined : 변수는 선언했지만 초기화 하지 않았을 때 - null : 값이 없다는 것을 의도적으로 명시할 때 사용하는 값 - symbol 심벌 : 고유하고 변경할 수 없는 값 (ES6에서 추가)
--

* 이스케이프 문자 : 자바스크립트에서 사용되는 특수한 문자형

특수문자	내용
\n	커서를 다음 줄로 이동
\t	커서를 탭(tab) 이동
\b	커서를 앞 문자를 지우며 이동
\f	커서를 다음 페이지 처음으로 이동
\r	커서를 그 줄의 처음으로 이동
\”	큰 따옴표
\’	작은 따옴표
\\	역슬래시

* **typeof** 연산자 : 자료형을 확인할 때 사용

<pre>document.write(typeof('String')); -> string document.write(typeof(1234)); -> number</pre>
--

3) 강제로 자료형 변환시키기

- 다른 자료형을 숫자로 - **Number()** 함수
- 다른 자료형은 문자열로 - **String()** 함수

4. 연산자

연산자 우선순위

1. ()
2. 단항 연산자(--, ++, !)
3. 산술 연산자(*, /, %, +, -)
4. 비교 연산자(>, >=, <=, ==, !=, !=)
5. 논리 연산자(&&, ||)
6. 대입(복합 대입) 연산자(=, +=, -=, *=, /=, %=)

1) 산술연산자

연산자	예	설명
+	$c = a + b$	a와 b의 합을 c에 저장, 덧셈 연산
-	$c = a - b$	a에서 b를 뺀 차를 c에 저장, 뺄셈 연산
*	$c = a * b$	a와 b의 곱을 c에 저장, 곱셈 연산
/	$c = a / b$	a를 b로 나눈 몫을 c에 저장, 나눗셈 연산
%	$c = a \% b$	a를 b로 나누었을 때 나머지를 c에 저장

2) 대입연산자

연산자	예	설명
+=	$c += a$	c와 a의 합을 c에 저장, 덧셈 연산
-=	$c -= a$	c에서 a를 뺀 차를 c에 저장, 뺄셈 연산
*=	$c *= a$	c와 a의 곱을 c에 저장, 곱셈 연산
/=	$c /= a$	c를 a로 나눈 몫을 c에 저장, 나눗셈 연산
%=	$c \% = a$	c를 a로 나누었을 때 나머지를 c에 저장

3) 비교연산자(관계연산자)

연산자	예	설명
-----	---	----

==	a == b	a와 b의 값이 같은지 비교
!=	a != b	a와 b의 값이 다른지 비교
===	a === b	a와 b의 값뿐만아니라 자료형도 같은지 비교
!==	a !== b	a와 b의 값뿐만아니라 자료형도 다른지 비교
>	a > b	a가 b 보다 큰지 비교
>=	a >= b	a가 b 보다 크거나 같은지 비교
<	a < b	a가 b 보다 작은지 비교
<=	a <= b	a가 b 보다 작거나 같은지 비교

4) 논리연산자

연산자	예	설명
&&	a && b	a와 b 둘 다 참일 때 참 (and 연산)
 	a b	a와 b 둘 중에 하나만 참이면 참 (or 연산)
!	!a	a가 참이면 거짓, 거짓이면 참으로 바꿈 (not 연산)

연산자	규칙
&&	좌측 피연산자 식의 평가값이 거짓이면, 우측 피연산자 식을 평가하지 않음
 	좌측 피연산자 식의 평가값이 참이면, 우측 피연산자 식을 평가하지 않음

true를 1로 false를 0으로 변환

a	b	a && b	a b	!a
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

5) 조건연산자(삼항연산자)

값 = 조건식 ? 참일 때 사용하는 문장(값) : 거짓일 때 사용하는 문장(값)

6) 증감연산자

연산자	예	설명
++	++a a++	a의 값을 1 증가 시킴
--	--b b--	b의 값을 1 감소 시킴

5. 제어문

1) 조건문

- if 문

if문

```
if(조건식){  
    문장  
}
```

if ~ else문

```
if(조건식){    문장  
}else{        문장  
}
```

if/else if ~ else 문(다중 조건 체크)

```
if(조건식){    문장  
}else if(조건식){    문장  
}else if(조건식){    문장  
}else if(조건식){    문장  
}else{        문장  
}
```

- switch 문

```
switch(변수){  
    case 상수1 : 문장1; break;  
    case 상수2 : 문장2; break;  
    case 상수3 : 문장3; break;
```



```
case 상수4 : 문장4; break;
default : 문장5;
}
```

2) 반복문

- for 문

```
for문

for(초기값;조건식;증감식){
    문장
}

다중 for문

for(초기값;조건식;증감식){
    문장
    for(초기값;조건식;증감식){
        문장
    }
}
```

- while 문

조건을 먼저 검사한 후 코드 블록 반복 실행

```
while(조건식){
    문장
}
```

- do ~ while 문

조건이 참인지 거짓인지와 상관없이 내부의 문장을 최소한 한 번은 실행해야 하는 경우 사용

```
do{
    문장
}while(조건식);
```

3) break문

특정조건일 때 반복문을 빠져나감

```
var output = 0;
```

```
for (var i = 1; i <= 10; i++) {
    if (i == 5) { // i가 5이면 현재 반복을 중지for문을 빠져 나감.
        break;
    }
    document.write(i+'<br>');
}
```

4) continue문

특정조건일 때 반복문에서 해당 반복 회차만 건너뛰어 다음 회차로 넘어가 수행문을 수행

```
var output = 0;
for (var i = 1; i <= 10; i++) {
    if (i % 2 == 1) { // 홀수이면 현재 반복을 중지하고 다음 반복을 수행
        continue;
    }
    document.write(i+'<br>');
}
```

6. 함수

함수의 역할

1. 호출 가능한 루틴으로서의 함수
2. 값으로서 함수
 - 1) 인자로 전달 가능
 - 2) 변수에 할당 가능
 - 3) 다른 함수의 반환값으로 사용 가능
3. 다른 인스턴스를 생성할 수 있는 요소, 객체 타입으로서의 함수

1) 함수 정의와 호출

- 선언적 함수 정의

```
function 함수 (매개변수명){
    코드
    return 반환값 // 반환할 값이 있을 경우 지정
}
```

[호출]

```
함수(매개변수에 전달할 값);
```

- 익명 함수 정의

```
var 함수 = function (매개변수명 ) {  
    코드  
    return 반환값 // 반환할 값이 있을 경우 지정  
};
```

[호출]

```
함수(매개변수에 전달할 값);
```

2) 매개 변수

- 함수를 호출하는 쪽과 호출된 함수를 연결하는 매개변수가 되는 변수
- 자바스크립트는 함수를 생성할 때 지정한 매개 변수보다 많거나 적은 매개 변수를 사용하는 것 허용

```
alert('원래 매개변수','추가된 매개 변수');<- 원래 매개변수외에 추가된 매개 변수
```

```
prompt('원래 매개 변수');<- 원래 두개의 매개변수를 명시해야 하지만 하나를 제거한 경우
```

3) 가변인자 함수

- 매개 변수의 개수가 변할 수 있는 함수
- 협의로는 매개 변수를 선언된 형태와 다르게 사용했을 때도 매개 변수를 모두 활용하는 함수

```
var array1 = Array();//빈 배열 생성  
var array2 = Array(10);//매개 변수만큼의 크기를 가지는 배열 생성  
var array3 = Array(273, 103, 57, 32);//매개 변수를 배열로 만듦
```

4) 함수는 내부에 자동 생성되는 변수 **arguments**

arguments는 객체의 자료형과 배열의 길이 출력

```
function sumAll() {  
    //배열은 객체임으로 Object 출력, 배열의 길이 출력  
    alert(typeof (arguments) + ': ' + arguments.length);  
}  
// 함수를 호출
```

```
sumAll(1, 2, 3, 4, 5, 6, 7, 8, 9);
```

5) return 값의 활용

return 키워드를 사용해 함수를 호출한 곳으로 값 넘김

```
function f(x) {  
    return x * x; //값은 반환함  
}  
alert(f(3)); //함수를 실행시켜 얻어진 결과값을 출력
```

return 키워드 사용시 값을 지정하지 않아도 함수를 호출한 곳으로 돌아감

```
function returnTest( ) {  
    alert('문장 A');  
    return; //함수를 호출한 곳으로 돌아감  
    alert('문장 B'); <-- 실행되지 않음  
}  
// 함수를 호출.  
returnTest();
```

6) 내부 함수

함수 내부에 선언한 함수

내부 함수를 사용하면 외부에 이름이 같은 함수가 있어도 내부 함수 우선

```
function 외부 함수(){  
    function 내부 함수1(){  
        코드  
    }  
  
    function 내부 함수2(){  
        코드  
    }  
  
    코드  
}
```

7) 함수를 반환하는 함수와 클로저

- 익명함수를 반환하는 함수

```
function outerFunction() {
    return function () {
        alert('Hello World .. !');
    };
}
outerFunction();
```

- 클로저의 사용

함수 안에 있는 변수는 지역 변수이므로 외부에서 사용할 수 없음. 클로저를 사용하면 이 규칙을 위반하여 지역변수를 사용할 수 있도록 할 수 있음.

익명함수를 반환하는 함수에 지역변수가 있으면 익명함수는 클로저 함수로서 지역변수를 가져다 쓸 수 있음.

클로저란?

- 지역 변수를 남겨두는 현상
- 함수 `outerFunction()`로 인해 생성된 공간
- 함수 `outerFunction()` 내부의 변수들이 살아있음
- 리턴되는 함수 자체
- 살아남은 지역 변수

```
function outerFunction(name) {
    var output = 'Hello ' + name + '.. !';
    return function () {
        alert(output);
    };
}
outerFunction('dragon');
```

8) 내장 함수

자바스크립트에서 자체 제공하는 기본 내장 함수

인코딩	문자를 컴퓨터에서 저장하거나 통신에 사용할 목적으로 부호화 – 한글 같은 유니코드 문자
-----	--

디코딩	부호화된 문자를 원래대로 되돌리는 것
-----	----------------------

인코딩, 디코딩과 관련된 내장 함수

함수	설명
<code>escape()</code>	데이터 인코딩 영문 알파벳, 숫자, 일부 특수 문자(<code>@</code> , <code>*</code> , <code>-</code> , <code>_</code> , <code>+</code> , <code>.</code> , <code>/</code>)를 제외한 모든 문자 1바이트 문자는 <code>%XX</code> 의 형태로, 2바이트 문자는 <code>%uXXXX</code> 의 형태로 변환
<code>unescape()</code>	데이터 디코딩 <code>escape()</code> 함수에서 인터넷 주소에 사용되는 일부 특수 문자(<code>:</code> , <code>;</code> , <code>/</code> , <code>=</code> , <code>?</code> , <code>&</code>)는 변환하지 않음
<code>encodeURIComponent(uri)</code>	최소한의 문자만 인코딩
<code>decodeURIComponent(encodedURI)</code>	최소한의 문자만 디코딩
<code>encodeURIComponent(uriComponent)</code>	대부분의 문자를 인코딩 알파벳과 숫자를 제외한 모든 문자 인코딩 UTF-8 인코딩과 같음
<code>decodeURIComponent(encodedURI)</code>	대부분의 문자를 디코딩

함수	설명
<code>eval(string)</code>	<code>string</code> 을 자바스크립트 코드로 실행
<code>isFinite(number)</code>	<code>number</code> 가 무한한 값인지 확인
<code>isNaN(number)</code>	<code>number</code> 가 NaN인지 확인
<code>parseInt(string)</code>	<code>string</code> 을 정수로 바꿈
<code>parseFloat(string)</code>	<code>string</code> 을 유리수로 바꿈

7. 배열

배열은 순서가 있는 요소의 집합. 자바스크립트 배열의 길이는 가변적이다. 요소를 말미에 더하면 배열의 길이가 자동으로 늘어난다.

배열 생성 예

```
var arr = [3,4,5];
for(var i=0;i<arr.length;i++){
    document.write(arr[i]);
}
```

정렬

sort 메서드를 이용하면 배열의 요소 값을 정렬할 수 있다. 인자 없이 **sort** 메서드를 호출하면 문자열 기준으로 정렬. 문자열 정렬은 **Unicode**의 코드 포인트 값의 대소 비교를 통해 이루어짐

문자열 정렬

```
var arr = ['one','two','three','four','five','six'];
arr.sort();
['five','four','one','six','three','two']
```

숫자 배열의 정렬

```
var arr = [1,0,20,100,55];
arr.sort(function(a,b){
    return a - b;
});
[0,1,20,55,100]
```

요소의 생성

```
var arr = [];
arr.push(요소);
```

요소의 제거

```
var arr = ['zero','one','xxx','two','three'];
delete arr[2];
zero,one,,two,three //두 번째 인덱스에 해당하는 요소는 빈 그대로 남음

arr.splice(2,1); //두 번째 인덱스에 해당하는 요소부터 하나의 요소를 제거
zero,one,two,three
```

8. 객체 생성

1) 객체 생성하기

```
- 객체를 생성할 때 key와 value의 쌍으로 속성 지정
var member = {
```

```
    name:'dennis Kim',  
    age:20,  
    job:'student'  
};
```

- 메서드 지정

```
var member = {  
    name:'dennis Kim',  
    age:20,  
    job:'student'  
    eat:function(){  
        코드  
    }  
};
```

- 생성된 객체에서 속성 호출

member.name 또는 **member['name']**

- 생성된 객체에서 메서드 호출

member.eat()

* 속성 호출시 대괄호 연산자만 쓸 수 있는 경우

(1) 식별자로 사용할 수 없는 프로퍼티명을 사용하는 경우

```
obj = {'foo-bar':5};
```

obj.foo-bar; //obj.foo와 bar의 뿔셈이라고 해석되어 에러가 발생

obj['foo-bar']; //[] 연산자를 사용해 문자열 값으로 지정

(2) 변수의 값을 프로퍼티명에 사용하는 경우

```
var key = 'x';
```

obj[key]; //프로퍼티 x (프로퍼티 key가 아님)

2) 객체 관련 키워드

- in : 해당 키가 객체 안에 있는지 확인

```
var output = "";
```

```
var student = {
```



```

        이름 : '홍길동',
        국어 : 92,
        수학 : 98,
        영어 : 96,
        과학 : 98
    };

    output += "'이름' in student: " + ('이름' in student) + '\n';
    output += "'성별' in student: " + ('성별' in student);

    alert(output);

```

- with : 객체명을 등록하고 속성에 접근할 때 속성명만 호출할 수 있도록 처리

```

var student = {
    이름: '홍길동',
    국어: 92,
    수학: 98,
    영어: 96,
    과학: 98
};

var output = "";

//with에 객체를 전달하면 with 블록내에서 속성명만으로 객체의 속성에 접근 가능
with (student) {
    output += '이름: ' + 이름 + '\n';
    output += '국어: ' + 국어 + '\n';
    output += '수학: ' + 수학 + '\n';
    output += '영어: ' + 영어 + '\n';
    output += '과학: ' + 과학 + '\n';
    output += '총점: ' + (국어 + 수학 + 영어 + 과학);
}
alert(output);

```

3) 동적 객체 생성과 추가

처음 객체를 생성하는 시점 이후에 객체의 속성 및 메서드 추가하거나 제거

- 속성 추가

```

//객체 생성
var student = {};

```

```
//객체에 속성 추가
student.이름 = '홍길동';
student.취미 = '악기';
student.특기 = '프로그래밍';
student.장래희망 = '프로그래머';
```

- 메서드 추가

```
//객체 생성
var student = {};

student.greet = function(){
    return 'Hello students!!';
};
```

- 속성 제거

```
delete(student.취미);
```

4) 생성자 함수를 이용한 객체 생성

new 키워드를 사용해 객체 생성할 수 있는 함수

```
function Student(name,korean,math,english,science){
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;

    //메서드 지정
    this.getSum = function(){
        return this.국어 + this.수학 + this.영어 + this.과학;
    };
    this.getAverage = function(){
        return this.getSum() / 4;
    };
    this.toString = function(){
        return this.이름 + ', ' + this.getSum() + ', ' + this.getAverage();
    };
}
```

```
//생성자 함수를 이용한 객체 생성
var student = new Student('홍길동',100,99,98,97);
```

5) 프로토타입을 이용한 메서드 생성

프로토타입은 생성자 함수를 사용해 생성된 객체가 공통으로 가지는 공간.

각기 객체를 생성할 때마다 동일한 함수를 계속 생성하는 것은 낭비이기 때문에 프로토타입을 이용해 공통으로 사용할 메서드를 지정

```
function Student(name,korean,math,english,science){
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;
}
Student.prototype.getSum = function(){
    return this.국어 + this.수학 + this.영어 + this.과학;
};
Student.prototype.getAverage = function(){
    return this.getSum() / 4;
};
Student.prototype.toString = function(){
    return this.이름 + ', ' + this.getSum() + ', ' + this.getAverage();
};
```

6) instanceof 연산자

- 생성자 함수를 통해 만들어진 객체가 인스턴스instance
- 해당 객체가 어떠한 생성자 함수를 통해 생성됐는지 확인할 때 사용

```
function Student(name) { this.name = name; }

var student = new Student('홍길동');

alert(student instanceof Student);
```

7) 캡슐화

객체 내부에 데이터가 저장되는 영역을 지역변수화해서 객체 외부에서 직접 접근을 불허하고 메서드를 이용해 접근하도록 처리함

```
// 생성자 함수를 선언
```

```
function Rectangle(w, h) {

    var width = w; //지역변수
    var height = h; //지역변수

    this.getWidth = function () { return width; };
    this.getHeight = function () { return height; };
    this.setWidth = function (w) {
        width = w;
    };
    this.setHeight = function (h) {
        height = h;
    };
}
```

8) 상속

- 기존의 생성자 함수나 객체를 기반으로 새로운 생성자 함수나 객체를 쉽게 만드는 것
- 기존의 객체를 기반으로 생성
- 상속을 통해 새로 만들어지는 객체에는 기존 객체의 특성 존재

9. 클라이언트 객체

1) 클라이언트 객체의 종류

객체	특징
window	최상위 객체로 자바스크립트에서 사용되는 모든 객체를 처리
navigator	웹 브라우저에 관련된 정보를 제공
link	하이퍼링크와 관련된 작업 처리
anchor	특정 위치로의 이동 객체
image	문서 내의 그림에 관련된 처리
location	현재 문서의 URL과 관련된 정보를 처리
history	방문기록에 관련된 처리
document	문서에 대한 정보를 처리
form	폼에 있는 양식 객체를 처리

2) window의 속성과 메서드

- 속성

속성	설명
defaultStatus	상태표시줄에 나타낼 초기 문자열 설정
status	상태표시줄에 나타낼 문자열 설정
frames	창에 포함된 프레임들을 배열로 지정
opener	새 창을 열도록 한 문서
parent	현재 프레임의 부모 문서 창(프레임)
self	현재 프레임 자신
top	프레임을 무시한 하나의 전체 창
screenX	창의 x 좌표값 반환
screenY	창의 y 좌표값 반환
closed	창이 닫힌 상태에 대한 true, false 반환
name	창의 이름 반환
length	창 안의 프레임수 반환

- 메서드

메서드	설명
alert('내용')	경고창 표시
prompt('제목','입력창에 미리 보여질 문구')	입력창 표시
confirm('내용')	확인창 표시
open('문서명','창이름','속성')	새 창 열기
close()	창 닫기
moveBy(x,y)	브라우저를 x,y 만큼 이동
moveTo(x,y)	브라우저를 x,y 로 이동
resizeBy(x,y)	브라우저의 크기를 상대적 x,y로 설정

resizeTo(x,y)	브라우저의 크기를 x,y 크기로 설정
scroll(x,y)	스크롤함
scrollBy(x,y)	현재 위치에서 스크롤을 x,y만큼 이동
scrollTo(x,y)	스크롤을 x,y로 이동
setTimeout(function,millisecond)	일정 시간 후에 함수를 한 번 실행
clearTimeout()	일정 시간 후에 함수를 한 번 실행하는 것을 중지
setInterval(function,millisecond)	일정 시간마다 함수를 반복해서 실행
clearInterval()	일정 시간마다 함수를 반복하는 것을 중단
print()	문서 출력

2. document 객체의 속성과 메서드

- 속성

속성	설명
bgColor	문서의 배경색을 지정
fgColor	문서의 글자색을 지정
linkColor	하이퍼링크의 문자 색상 지정
vlinkColor	방문한 링크의 문자 색상을 지정
lastModified	문서가 마지막으로 수정된 날짜를 저장
location	문서의 URL 주소를 저장
URL	문서의 URL 주소값을 반환
domain	서버의 도메인명을 지정 또는 반환
title	문서의 제목을 지정하거나 반환
cookie	쿠키 파일의 정보를 지정하거나 반환
images	문서에 삽입된 그림을 배열로 제공
links	문서의 링크를 배열로 제공
forms	문서의 form 태그로 이루어진 폼을 배열로 제공

embeds	문서에 포함된 플러그인을 배열로 제공
--------	----------------------

- 메서드

메서드	설명
open()	문서의 내용을 나타냄
close()	open()으로 나타낸 문서를 닫음
clear()	문서의 모든 내용을 삭제
write()	태그를 포함하여 문자열을 출력
writeln()	행 마지막에 new line 처리

3) history 객체의 속성과 메서드

-속성

속성	설명
length	히스토리에 보관되어 있는 URL 주소 수

- 메서드

메서드	설명
back()	이전 페이지로 이동
forward()	다음 페이지로 이동
go(n)	n=양수 : 히스토리 목록에서 n만큼 다음 페이지로 이동 n=음수 : 히스토리 목록에서 n만큼 이전 페이지로 이동

4) image 객체의 속성

속성	설명
name	그림에 설정된 name 속성값을 반환
length	문서에 삽입된 그림의 개수를 반환
src	그림의 src 속성값을 반환

hspace	설정된 그림의 좌/우 여백값을 반환
vspace	설정된 그림의 상/하 여백값을 반환
width	설정된 그림의 폭을 반환
height	설정된 그림의 높이를 반환
border	설정된 그림의 테두리 값을 반환

5) location 객체의 속성과 메서드

- 속성

속성	설명
host	호스트명과 포트 번호 정보를 표시(URL에 포트가 표시되어 있을 경우) (www.naver.com) or (localhost:8080)
hostname	호스트명을 표시 (www.naver.com)
href	전체 URL을 표시 (http://www.naver.com/index.html)
pathname	웹 문서의 경로를 표시 (/index.html)
port	포트 번호를 표시 (8080)
protocol	프로토콜명을 표시 (http:)
search	쿼리 파라미터 (?q=bar)
hash	해시 코크 (#baz)

- 메서드

메서드	설명
assign()	현재 표시 중인 페이지에서 다른 페이지로 이동

	(href 프로퍼티에 값을 설정하는 것과 같은 동작)
reload()	문서 새로고침 location.reload(true); //브라우저 캐시를 무시하고 리로드 location.reload(false); //브라우저 캐시를 이용해 리로드 location.reload(); //location.reload(false);와 같음
replace("URL")	현재 문서를 지정한 URL로 대체 (브라우저 이력이 남지 않기 때문에 '뒤로 가기'버튼으로 원래 페이지로 돌아가는 것 불가능)

6) form 객체의 속성과 메서드

- 속성

속성	설명
action	<form> 태그의 action 속성에 기록된 정보를 보관
elements	텍스트 입력상자, radio 버튼 등 폼 양식을 배열로 저장
encoding	encoding 속성에 기록된 정보를 보관
method	method 속성에 기록된 정보를 보관
target	target 속성에 기록된 정보를 보관
length	폼 양식의 개수
name	name 속성에 기록된 정보를 보관

- 메서드

메서드	설명
reset()	form 양식에 입력된 값을 초기화
submit()	form 양식에 입력된 값을 전달

(1) 텍스트 입력 태그(input)의 속성과 메서드

- 속성

속성	설명
----	----

type	텍스트 입력상자의 종류를 지정(text, password)
name	텍스트 입력상자의 이름
value	텍스트 입력상자에 입력되는 내용
defaultValue	텍스트 입력상자에 입력된 초기값
form	텍스트 입력상자를 포함하는 폼 객체

- 메서드

메서드	설명
focus()	호스트명과 포트 번호 정보를 표시
blur()	호스트명을 표시
select()	전체 URL을 표시

(2) select 객체의 속성

속성	설명
type	multiple 정보를 반환
length	목록의 개수를 반환
options	<option> 태그를 배열로 구성
selectedIndex	목록을 배열 번호로 표시하거나 배열 번호를 반환

(3) radio, checkbox 객체의 속성과 메서드

- 속성

속성	설명
form	입력 양식을 포함하는 폼 객체
name	입력 양식의 이름
value	입력 양식에 할당되는 값

length	입력 양식 수
checked	입력 양식 선택 여부
defaultChecked	기본적으로 선택된 상태로 표시될 입력 양식
type	type 속성값을 반환

- 메서드

메서드	설명
click()	버튼을 클릭하는 메서드

(4) Button 객체의 속성과 메서드

- 속성

속성	설명
form	버튼의 입력 양식을 포함하는 객체
name	버튼의 name 속성을 반환
value	버튼의 value 속성을 반환
type	버튼의 type 속성을 반환

- 메서드

메서드	설명
click()	버튼을 클릭하는 메서드

(5) form의 target 프로퍼티

값	결과를 표시하는 곳
_blank	새 윈도우
_self	현재 프레임(윈도우)
_parent	부모 프레임

_top	프레임 분할을 해제하고 윈도우 전체에 표시
프레임명,윈도우명	지정된 임의의 프레임(윈도우)

10. 내장 객체

1) 내장 객체의 종류

객체	특징
Array	여러 Data 를 하나의 이름으로 저장하고 다루기 위해 사용하는 객체
String	글자의 모양을 지정하거나 문자열을 처리하기 위해 사용하는 객체
Date	날짜와 시간을 표시하거나 설정
Event	로컬컴퓨터에서 발생하는 이벤트에 관한 속성 정보를 알려는 객체
Math	수학계산을 위해 사용되는 객체
Function	함수를 만들 때 사용하는 객체
Screen	Local 컴퓨터의 화면의 해당도나 색상을 알기위한 객체
Number	문자로된 숫자 단어를 실제 숫자로 바꿔주는 객체
Boolean	bool 형식의 값을 가지는 객체를 만들어주는 객체

2) Array 객체의 생성자 함수, 속성, 메서드

- 생성자 함수

생성자 함수	설명
Array()	빈 배열을 만듦
Array(number)	매개 변수만큼의 크기를 가지는 배열을 만듦
Array(mixed,...,mixed)	매개 변수를 배열로 만듦

- 속성

속성	설명
length	배열 요소의 개수를 알아냄

- 메서드

메서드	설명
concat()	매개 변수로 입력한 배열의 요소를 모두 합쳐 배열을 만들어 반환
join()	배열 안의 모든 요소를 문자열로 만들어 반환
pop()	배열의 마지막 요소를 제거하고 반환
push()	배열의 마지막 부분에 새로운 요소를 추가
reverse()	배열의 요소 순서를 뒤집음
slice()	배열 요소의 지정한 부분을 반환
sort()	배열의 요소를 정렬하고 반환
splice()	배열 요소의 지정한 부분을 삭제하고 삭제한 요소를 반환

3) String 객체의 속성, 메서드

- 속성

속성	설명
length	문자열의 길이

- 메서드

메서드	설명
charAt(position)	문자열에서 position 에 위치한 문자를 반환 인덱스 번호는 0부터 시작
charCodeAt(position)	문자열에서 position 에 위치한 문자의 유니코드 번호를 반환 인덱스 번호는 0부터 시작
concat(string,...,string)	매개 변수로 입력한 문자열을 연결해서 리턴
indexOf('문자',n)	문자열에서 해당 문자의 위치를 왼쪽 n번째부터 찾아서 찾은 위치를 반환

	n이 생략되면 왼쪽에서부터 찾고 없으면 -1dmf qksghks 인덱스 번호는 0부터 시작
lastIndexOf('문자',n)	문자열에서 해당 문자의 위치를 오른쪽 n번째부터 찾아서 찾은 위치를 반환 n이 생략되면 오른쪽에서부터 찾고 없으면 -1dmf qksghks 인덱스 번호는 0부터 시작
localeCompare(that)	로케일에 의한 문자열 비교. 비교 결과에 따라 양수, 0, 음수인 정수를 반환
match(regExp)	문자열 내에 regExp가 있는지 확인
replace(regExp,replacement)	regExp를 replacement로 바꾼 뒤 반환
search(regExp)	regExp와 일치하는 문자열의 위치를 반환
slice(start,end)	특정 위치의 문자열을 추출해 반환
split('구분자')	구분자로 문자열을 분리
substring(n,m)	문자열의 n번째 문자부터 m번째 문자까지를 반환
substr(n,m)	문자열의 n번째 문자부터 m개의 문자를 반환
substring(start,end)	start부터 end까지의 문자열을 새로운 문자열 값으로 반환. 동작 방식은 slice와 같지만 인자로 음수를 지정할 수 없음
toLocaleLowerCase()	문자열의 각 문자를 로케일을 고려해 소문자로 변환
toLocaleUpperCase()	문자열의 각 문자를 로케일을 고려해 대문자로 변환
toLowerCase()	모두 소문자로 변환
toString()	String 인스턴스에서 문자열 값으로 변환
toUpperCase()	모두 대문자로 변환
trim()	문자열 전후의 공백을 제거
valueOf()	String 인스턴스에서 문자열 값으로 변환

*** HTML** 관련 메서드

메서드	설명
anchor()	a 태그로 문자열을 감싸서 반환

big()	big 태그로 문자열을 감싸서 반환
blink()	blink 태그로 문자열을 감싸서 반환
bold()	문자열을 진하게 표현
fixed()	tt 태그로 문자열을 감싸서 반환
fontcolor('color')	문자열에 색상을 지정
fontsize(n)	문자열에 크기를 지정
italics()	문자열을 기울임꼴로 표현
link('url')	문자열에 하이퍼링크를 설정
small()	small 태그로 문자열을 감싸서 반환
sup()	문자열을 윗첨자로 표현
sub()	문자열을 아래첨자로 표현
strike()	strike 태그로 문자열을 감싸서 반환

4) Date 객체의 메서드

메서드	설명
getFullYear()	연도 정보를 반환
getMonth()	월 정보를 반환(0~11)
getDate()	일 정보를 반환(1~31)
getDay()	요일 정보를 숫자로 반환(일요일 0 ~ 토요일 6)
getHours()	시간 정보를 반환
getMinutes()	분 정보를 반환
getSeconds()	초 정보를 반환
getTime()	1970년 1월 1일 0시부터 지정한 시간까지를 1/1000초로 반환

5) Math 객체의 속성과 메서드

- 속성

속성	설명
PI	원주율(3.14159..)
E	오일러 상수 자연대수의 밑수(2.71828..)
LN10	밑수가 10인 자연로그(2.302585)
LN2	밑수가 2인 자연로그(0.6931471..)
SQR2	2의 제곱근(1.41421..)
LOG10E	밑수가 10인 E로그(0.43429448..)
LOG2E	밑수가 2인 E로그(1.442695)

- 메서드

메서드	설명
sin(x)	sine 함수
cos(x)	cosine 함수
tan(x)	tangent 함수
asin(x)	arc sine 함수
acos(x)	arc cosine 함수
atan(x)	arc tangent 함수
abs(x)	x의 절대값
exp(x)	E에 대한 지수 함수
log(x)	x 로그값
pow(x,y)	지수 함수, x의 y 제곱
sqrt(x)	제곱근 함수 x의 루트값
random()	난수 발생 함수
round(x)	x값에 대한 반올림 함수
floor(x)	x의 소수점 이하 값을 절삭하는 함수
ceil(x)	x의 소수점 이하 값을 무조건 반올림하는 함수

max(x,y)	x와 y 중 큰 값을 반환
min(x,y)	x와 y 중 작은 값을 반환

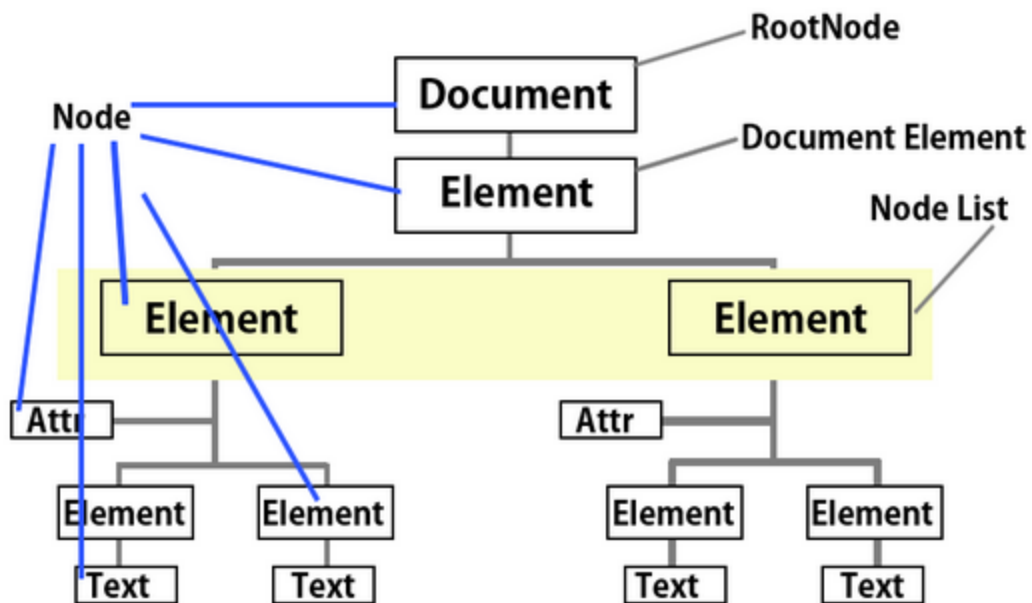
5) Number 객체의 속성

- 속성

prototype	프로토타입 체인용
length	값은 1
MAX_VALUE	64비트 부동소수점 수로 표현할 수 있는 최댓값
MIN_VALUE	64비트 부동소수점 수로 표현할 수 있는 최솟값
NaN	Not a Number를 나타내는 값
NEGATIVE_INFINITY	음의 무한대를 나타내는 값
POSITIVE_INFINITY	양의 무한대를 나타내는 값

11. 문서객체모델(DOM - Document Object Model)

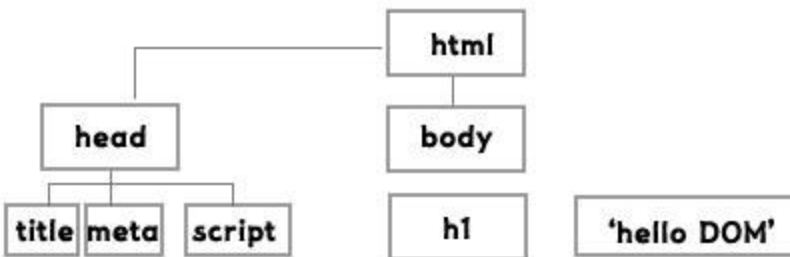
- 넓은 의미로 웹 브라우저가 HTML 페이지를 인식하는 방식
- 좁은 의미로는 **document** 객체와 관련된 객체 집합
- 사용시 HTML 페이지에 태그를 추가, 수정, 제거할 수 있음
- ‘태그’ - HTML 페이지에 존재하는 **html**이나 **body** 태그
- 문서 객체 - 태그를 자바스크립트에서 이용할 수 있는 객체로 만든 것



1) 문서객체의 동적 생성

메서드	설명
<code>createElement(tagName)</code>	요소 노드를 생성
<code>createTextNode(text)</code>	텍스트 노드를 생성

```
var header = document.createElement('h1'); //h1 태그 생성
var textNode = document.createTextNode('Hello DOM'); //텍스트 생성
```



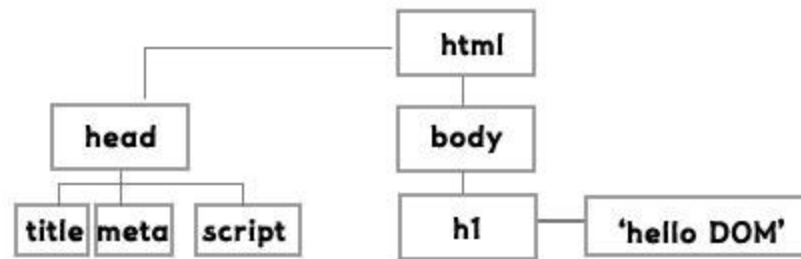
2) 화면에 문서객체 출력

메서드	설명
-----	----

appendChild(node)	객체에 노드를 연결
-------------------	------------

```
var header = document.createElement('h1'); //h1 태그 생성
var textNode = document.createTextNode('Hello DOM'); //텍스트 생성

//노드를 연결
header.appendChild(textNode);
document.body.appendChild(header);
```



3) HTML 태그를 자바스크립트로 가져오는 방법

메서드	설명
getElementById(id)	태그의 id 속성이 id와 일치하는 문서 객체를 가져옴
getElementsByName(name)	태그의 name 속성이 name과 일치하는 문서 객체를 배열로 가져옴
getElementsByTagName(tagName)	tagName과 일치하는 문서 객체를 배열로 가져옴

4) 노드의 탐색

프로퍼티명	취득할 수 있는 노드
parentNode	부모 노드
childNodes	자식 노드 리스트
firstChild	최초의 자식 노드
lastChild	최후의 자식 노드
nextSibling	다음 노드

previousSibling	하나 전의 노드
-----------------	----------

5) 문서 객체의 **innerHTML** 속성 사용해 객체 생성

innerHTML 프로퍼티에 값을 설정하면 브라우저에서는 그 내용을 파싱하고, 해석 결과를 해당 요소의 자식 요소로 만듦.

```
var output = '<h1>Hello World!!</h2>';
document.body.innerHTML = output;
```

6) **textContent** 속성

```
var output = '<h1>Hello World!!</h2>';
document.body.textContent = output;
//h1 요소는 생성되지 않음. 이대로 문자열로 브라우저에 표시됨
```

7) 문서 객체의 **style** 속성 사용

문서 객체의 **style** 변경 가능

```
var header = document.getElementById('header');

// 문서 객체의 스타일을 바꿈
header.style.border = '2px Solid Black';
header.style.color = 'Orange';
header.style.fontFamily = 'Helvetica';
```

8) 문서 객체 제거

메서드	설명
removeChild(child)	문서 객체의 자식 노드를 제거

12. 이벤트

1) 이벤트와 이벤트 속성

- 이벤트(event)

사용자에 의한 특정 행위의 결과로 발생하는 사용자와 프로그램 사이의 상호작용 처리의 요인
ex) 버튼 클릭, 하이퍼링크 위에 마우스 커서를 위치시킴

- 이벤트 속성

이벤트가 발생했을 때 이벤트 발생으로 호출되어야 할 이벤트 핸들러(실행 함수)를 연결하는 역할

	이벤트 속성	설명
마우스	onclick	버튼 등을 마우스로 클릭할 때
	ondblclick	버튼 등을 마우스로 더블 클릭할 때
	onmouseover	링크나 그림, 버튼 위에 마우스 포인터를 올릴 때
	onmouseout	링크나 그림, 버튼에서 마우스 포인터가 빠져나갈 때
	onmousedown	마우스 버튼을 클릭하는 순간
	onmouseup	마우스 버튼을 클릭했다가 떼는 순간
	onmousemove	마우스를 움직이는 순간
	ondragdrop	마우스를 클릭한 상태에서 움직일 때
포커스	onfocus	커서가 위치할 때
	onblur	커서가 다른 곳으로 옮겨갈 때
키보드	onkeydown	키보드를 누를 때
	onkeyup	키보드를 눌렀다가 떼는 순간
폼	onsubmit	전송(확인)버튼을 클릭할 때
	onreset	취소 버튼을 클릭할 때
	onselect	문자열을 선택하거나 체크 박스, 라디오 버튼 등을 선택할 때
	onchange	리스트 박스의 값을 변경할 때
윈도우	onload	브라우저에서 문서를 읽을 때
	onunload	브라우저에서 문서를 닫을 때
	onmove	브라우저를 이동했을 때
	onresize	브라우저의 크기를 변경했을 때

2) 이벤트 모델의 종류

- DOM Level 0

인라인 이벤트 모델

고전 이벤트 모델

- DOM Level 2

마이크로소프트 인터넷 익스플로러 이벤트 모델

표준 이벤트 모델

3) 인라인 이벤트 모델

HTML 태그의 이벤트 속성에 이벤트 핸들러(실행 함수)를 직접 입력하는 형태

```
<html>
<head>
<script type="text/javascript">
    function whenClick(){
        alert('CLICK');
    }
</script>
</head>
<body onload="init();">
    <div id="header" onclick="whenClick();">클릭</div>
</body>
</html>
```

4) 고전 이벤트 모델

```
<html>
<head>
<script type="text/javascript">
    window.onload = function(){
        var header = document.getElementById("header");

        function whenClick(){
            alert('CLICK');
        }

        header.onclick = whenClick; //이벤트 연결
    };
</script>
</head>
<body>
    <div id="header">클릭</div>
</body>
</html>
```

5) 이벤트 객체의 사용

```
var event = e || window.event;
```

-e가 존재하면 e를 변수 event에 넣고

-e가 undefined이면 window.event 속성을 변수 event에 넣음

* 인터넷 익스플로러 8 이하의 버전

이벤트가 발생시 이벤트 객체 window.event 속성으로 전달

다른 브라우저는 이벤트 핸들러의 매개 변수로 전달

사용 예)

```
window.onload = function(){
    document.body.onclick = function(e){
        //이벤트 객체를 설정
        var event = e || window.event;
        document.body.innerHTML = "";
        for(var key in event){
            document.body.innerHTML += '<p>' + key + ':' + event[key] + '</p>';
        }
    };
};
```

6) 기본 이벤트의 의미와 기본 이벤트 제거

일부 HTML 태그는 이미 이벤트 핸들러 가지고 있는데 이벤트가 발생할 때 이벤트 연결 없이 사용할 수 있는 이벤트를 기본 이벤트라고 함.

기본 이벤트가 있는 HTML 태그에 이벤트 연결을 하면 이벤트 연결에 의해 새롭게 호출되는 이벤트 핸들러와 기본 이벤트에 의해 고유의 이벤트 핸들러가 함께 호출되기 때문에 원하는 형태의 이벤트 처리가 어려워짐. 따라서 기본 이벤트를 제거함으로써 원하는 이벤트 연결이 가능하게 처리해야 함

```
<script>
    window.onload = function () {
        // 이벤트를 연결
        document.getElementById('my_form').onsubmit = function () {
            return false; //기본 이벤트 제거
        };
    };
</script>
```

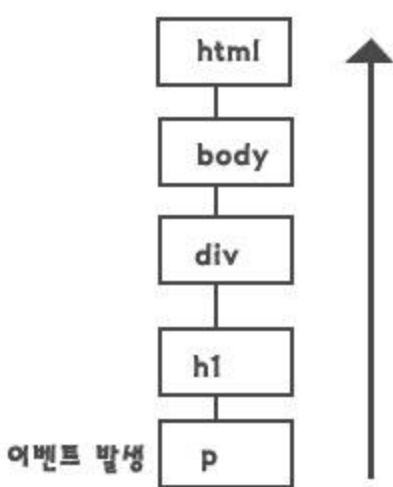
```

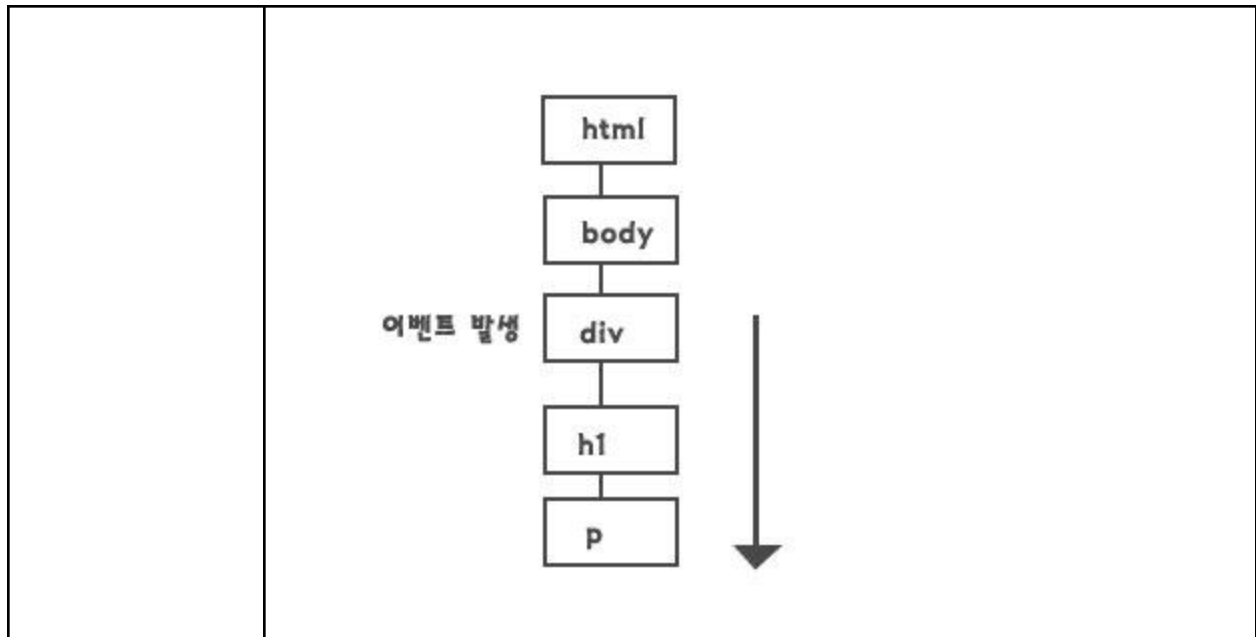
<form id="my_form">
  <label for="name">이름</label><br/>
  <input type="text" name="name" id="name"/><br/>
  <label for="pass">비밀번호</label><br/>
  <input type="password" name="pass" id="pass"/><br/>
  <label for="pass_check">비밀번호 확인</label><br/>
  <input type="password" id="pass_check"/><br/>
  <input type="submit" value="제출"/>
</form>

```

7) 이벤트 전달

어떠한 이벤트가 먼저 발생해 어떤 순서로 발생하는지 정하는 것을 이벤트 전달이라고 함
일반적으로 자바스크립트의 이벤트 전달 순서는 이벤트 버블링 방식

이벤트 버블링	<p>자식 노드에서 부모 노드 순으로 이벤트를 실행하는 것을 의미 (인터넷 익스플로러/비ie 지원)</p>  <p>이벤트 발생</p>
이벤트 캡처링	<p>이벤트가 부모노드에서 자식노드 순으로 실행되는 것을 의미 (비ie만 지원)</p>



8) 이벤트 전달 막기

```

window.onload = function () {

    document.getElementById('header').onclick = function () {
        alert('header');
    };
    document.getElementById('paragraph').onclick = function (e) {

        var event = e || window.event;

        alert('paragraph');

        // 이벤트 전달 제거
        event.cancelBubble = true;
        if (event.stopPropagation) {
            event.stopPropagation();
        }
    };
};

```

9) DOM Level 2

10) 인터넷 익스플로러 이벤트 모델

attachEvent(eventProperty,eventHandler)	이벤트 연결
---	--------

<code>detachEvent(eventProperty,eventHandler)</code>	이벤트 제거
--	--------

11) 표준 이벤트 모델

<code>addEventListener(eventName, handler, useCapture)</code>	이벤트 연결
<code>removeEventListener(eventName, handler)</code>	이벤트 제거

13. 예외 처리

1) 예외 처리란?

프로그램이 실행되는 동안 문제가 발생하면 프로그램 자동 중단되는데 사전에 문제 발생을 예상해서 프로그램이 안전하게 종료되도록 처리함

2) 기본 예외 처리

<pre>try { //예외 발생 가능 문장 } catch (exception) { //예외가 발생했을 때 실행될 문장 }</pre>
--

3) finally의 사용

finally 구문의 사용은 필수 사항은 아님

예외 발생 여부와 상관없이 수행돼야 하는 작업이 있을 때 사용

<pre>try { //예외 발생 가능 문장 } catch (exception) { //예외가 발생했을 때 실행될 문장 }finally{ //예외가 발생하건 발생하지 않건 꼭 수행해야 할 문장 }</pre>

4) 예외 객체

<pre>try {</pre>

```

} catch (exception) { //예외가 발생 예외에 관한 정보를 담은 예외 객체가 생성되고
                        //catch의 인자로 전달받아 예외 정보를 출력할 수 있음
}

```

예외 객체의 속성

속성	설명
message	예외 메시지
description	예외 설명
name	예외 이름

5) 예외 강제 발생

```

throw '예외 강제 발생';

```

사용 예)

```

try {
    var result = 10/0;

    if(result == 'Infinity'){
        throw 'DivideByZeroException';
    }
    alert(result);
} catch (exception) {
    alert(exception);
}

```