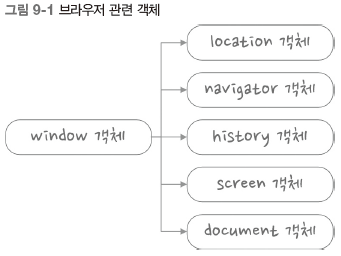
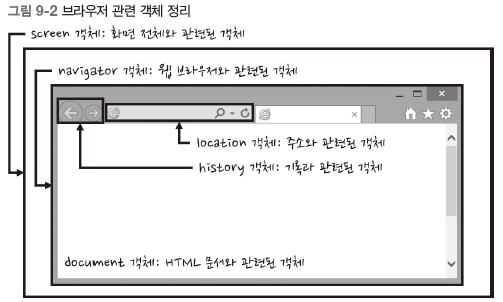
**Ch09 브라우저 객체 모델**

**브라우저 객체 모델(BOM, Browser Object Model)은 웹 브라우저와 관련된 객체의 집합을 의미한다. 대표적인 브라우저 객체 모델은 다음 그림과 같다**





9.1 window 객체

<script>

var output = '';

var i = 0;

for (var key in window) {

output += (++i)+' ★ ' + key + ':' + window[key] + '<br>';

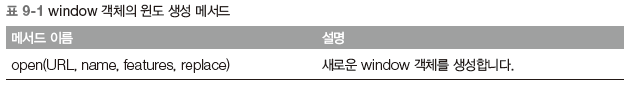
}

document.write(output);

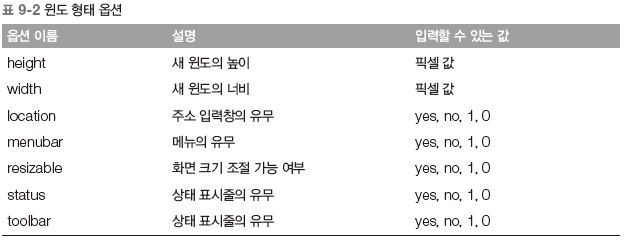
</script>

* 위의 예제에서 보듯 window 객체는 이렇게 많은 속성과 메서드가 있으므로 자주 사용되는 속성과 메서드만 본다
* window 객체는 브라우저 기반 자바 스크립트의 최상위 객체. 지금까지 함수처럼 사용한 alert(), prompt()함수 모두 window객체의 메서드이다. Var 키워드로 선언한 일반 변수도 모두 window 객체의 속성이 된다.

9.2 새로운 window 객체 생성



* window 객체의 open()메서드의 매개변수는 모두 옵션이다.
  + URL : 열고자 하는 HTML 페이지의 URL
  + name : 윈도우 간 통신하는데 사용하는 윈도우 이름 또는 target속성
  + features : 윈도우 출력 모양을 지정하는 옵션



<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<script>

var child = window.open('', '', 'width=450, height=500');

/\* window.open(속성1, 속성2, 속성3)

\* 속성1 : 새창의 url (내부자원, 외부 url)

\* 속성2 : 윈도우 이름 지정(윈도우 간에 통신하는데 사용)

\* 속성3 : 새창옵션 - width(300), height, location(1yes,0no)

\* menubar(1yes,0no), resiable(창크기변경-1yes,0no)

\* left, top(창의 왼쪽위치, 윗쪽위치),

\* scollbar(1yes,0no)

\*/

if (!child) { // 팝업차단 해제

alert('팝업차단을 해제하셔야 새창이 열립니다');

} else {

child.document.write('<h1>새창</h1>');

}

</script>

</head>

<body>

</body>

</html>

* ※ Window의 이름은 다음과 같은 경우 사용되기도 한다 open() 메서드는 새로운 window 객체를 생성하는 메서드. 단지 팝업창을 여는 것에서 끝나지 않고 윈도우 객체를 리턴한다. 따라서 다음의 예제처럼 새로운 윈도 객체에 접근해 속성과 메서드를 사용할 수 있다

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<script>

function openWindow() {

var child = open('', 'childWin', 'width=800,height=600');

if (child) {

child.document.write(child.name);

}

}

</script>

</head>

<body>

<button onclick="openWindow()">새창 열기 테스트</button>

<hr />

<a href="http://www.naver.com" target="childWin">열린창 경로 바꾸기</a>

</body>

</html>

※ 작은 사진 위에 마우스를 over하면 큰 사이즈의 사진이 나오고, 3초후 큰 사이즈 사진이 닫히는 로직

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<script>

var child;

function bigopen() {

child = window.open('5\_child.html', '', 'width=450,height=550');

if (!child) {

alert('큰그림을 원하면 팝업차단해제 요망');

}

}

function bigclose() {child.close();}

</script>

</head>

<body>

<img src="img/small.jpg" alt="작은그림" onmouseover="bigopen()" onmouseout="bigclose()"/>

</body>

</html>

5\_child.html

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<style>button { margin-left:150px;}</style>

<script>

for (var key in window) {

document.write(key + ':' + window[key] + '<br>');

}

// 객체 속성 중 on으로 시작하는 속성을 이벤트 속성이라 하고 함수를 할당해야 함.

//원래는 onload에 null이 있으나 아래와 같은 로직을 추가하면 해당 이벤트가 발생할 때 해당 함수가 실행

window.onload = function () {

this.setTimeout(function () { window.close(); }, 3000);

};

</script>

</head>

<body>

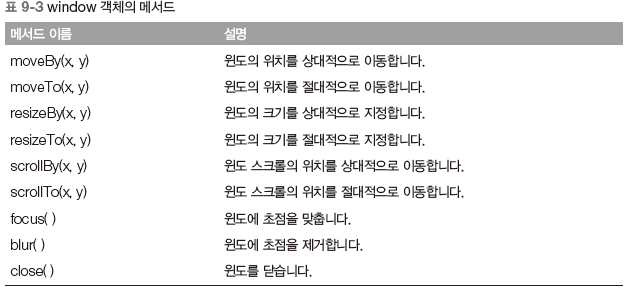
<img src="img/large.jpg" alt="큰그림"/><br />

<button onclick="window.close()">닫기</button>

</body>

</html>

9.3 window 객체의 기본 메소드 ; window 객체는 자신의 형태와 위치를 변경할 수 있게 다음과 같은 메서드를 제공한다. OOby() 형태의 메서드는 현재 위도를 기준으로 상대적으로 속성을 변화하는 메서드며, OOTo()형태의 메서드는 절대적인 기준으로 속성을 변화해 준다.



<script>

var child = window.open('', '', 'width=300, height=200');

child.moveTo(0, 0);

setInterval(function () { child.moveBy(10, 10); }, 1000);

</script>

9.4 screen 객체

- screen 객체는 웹 브라우저의 화면이 아니라 운영체제 화면이 속성을 가지는 객체

<script>

var child = window.open('', '','width=300, height=300');

var i = 0;

var w = screen.width/2;

var h = screen.height/2;

child.resizeTo(w, h);

child.moveTo(screen.width/4, screen.height/4);

setInterval(function () {

if (i) {

child.moveBy(100, 100); i = 0;

} else {

child.moveBy(-100, -100); i = 1;

}

}, 1000);

var output = '';

for (var key in screen) {

output += key + ' : ' + screen[key] + '<br>';

}

child.document.write(output);

</script>

9.5 location 객체

* 브라우저의 주소 표시줄과 관련된 객체

<script>

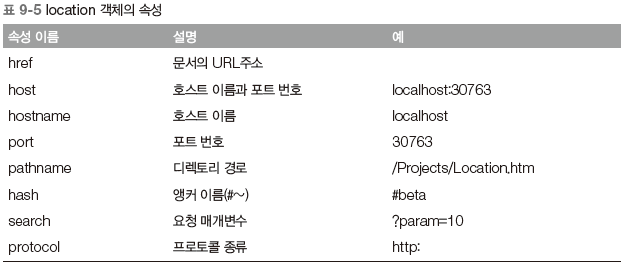
var output = '';

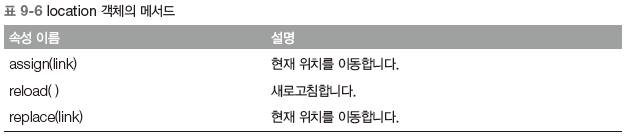
for (var key in location)

output += key + ' : ' + screen[key] + '\n';

alert(output);

</script>





* location 객체는 페이지를 이동할 때 많이 사용한다. 다음 네가지 방법으로 페이지를 이동할 수 있으나 assign() 메서드와 다르게 replace()메서드는 뒤로 가기 버튼을 사용할 수 없다.
  + location = ‘http://hanb.co.kr’;
  + location.href = ‘http://hanb.co.kr’;
  + location.assign(‘http://hanb.co.kr’);
  + location.replace(‘http://hanb.co.kr’);

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<script>

setTimeout(function () {

location.href = 'http://hanb.co.kr';

}, 3000);

</script>

</head>

<body>

잠시후 페이지 이동이 있겠습니다

</body>

</html>

* location객체와 history 객체

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<script>

function loc7() {

//location = '7.html';

//location.href = '7.html';

location.assign('7.html');

//location.replace('7.html');

}

</script>

</head>

<body>

<button onclick="loc7()">7로 가기</button>

</body>

</html>

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<script>

function historyback() {

history.go(-1);

//history.back();

}

</script>

</head>

<body>

<button onclick="historyback()">뒤로가기(바로전)</button>

</body>

</html>