

나만의 윈도우 라이브 가젯 만들기

# 이상한 나라의 자바스크립트

Windows Live 가젯은 웹에서 동작하는 간단한 프로그램이다. 가젯 제작을 위해서 가장 중요한 것은 자바스크립트와 브라우저의 DOM 구조를 이해하는 것이다. 이번 시간에는 자바스크립트의 특징과 자바스크립트로 DOM 구조를 조작하는 방법을 소개한다.

## 목차

목차.....	1
연재 가이드.....	1
연재 순서.....	2
필자 소개.....	2
필자 메모.....	2
Intro.....	2
var.....	4
var를 꼭 써야 하나요?.....	4
함수 객체, 스코프, 클로저.....	7
객체 생성 방법.....	9
객체에 메소드 추가하기.....	10
DOM.....	12
DOM 노드 찾기.....	13
DOM 노드 생성.....	15
DOM 노드 추가, 삭제.....	15
도전 과제.....	17
참고자료.....	17

## 연재 가이드

운영체제: 윈도우 2000/XP

개발도구: Editplus, IE7 or FireFox

기초지식: Javascript, HTML, CSS

응용분야: Windows Live Gadget 프로그램, AJAX 프로그램

# 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

## 연재 순서

### 2007. 02 이상한 나라의 자바스크립트

2007. 03 체스 기보 뷰어 만들기

2007. 04 Hello, World 가젯 만들기.

2007. 05 StockViewer 가젯 만들기

## 필자 소개

신영진 pop@jiniya.net, <http://www.jiniya.net>

시스템 프로그래밍에 관심이 많으며 다수의 보안 프로그램 개발에 참여했다. 현재 데브피아 Visual C++ 섹션 시삽을 맡고 있으며, Microsoft Visual C++ MVP로 활동하고 있다. 최근에는 python과 lua같은 스크립트 언어를 배우려고 노력하고 있다.

## 필자 메모

요즘 프로그램들은 크고 복잡하다. 대부분의 프로그램들이 수십 명에서 수천 명의 공동 작업으로 이루어진다. 전체적인 내부 구조를 이해하고 자신의 코드를 작성하는 개발자는 정말 손에 꼽을 만큼 작다. 이렇게 복잡함 속에 빠져 살다 보면 단순하고 간단한 것이 정말 그리워진다. 혼자 푹푹푹 몇 일 작업해서 프로그램이 만들어진다면 정말 재미있을 것 같다. 이런 생각을 조금이라도 해 본 개발자라면 가젯 제작에 한번 빠져보도록 하자. 잃어버린 프로그래밍의 즐거움을 찾아줄 것이다.

가젯은 쉽다. 가젯은 단순한 프로그램이다. 가젯은 손쉽게 설치하고 제거할 수 있다. . 아이디어만 있으면 누구나 몇 일간의 작업으로 근사한 프로그램을 만들 수 있다.

## Intro

가젯이란 사용자에게 유용한 정보를 제공하는 간단한 프로그램이다. 가젯에는 총 세 가지 종류가 있다. 웹 가젯, 사이드바(Sidebar) 가젯, 사이드쇼(SideShow) 가젯이 그것이다. 웹 가젯은 앞으로 우리가 제작해 볼 것으로 웹 상에서 플러그인 형태로 동작하는 작은 프로그램이며, live.com에 설치해서 사용할 수 있다(<화면 1> 참고). 사이드바 가젯은 윈도우 비스타에서 사용되는 가젯으로 비스타의 사이드바에 설치해서 사용할 수 있는 가젯이다(<그림 1> 참고). 사이드쇼 가젯은 휴대용 디바이스의 보조 출력 장치에 사용되는 가젯이다. 이러한 사이드쇼 기능을 탑재한 대표적인 제품으로 ASUS 노트북(<그림 2> 참고)이 있다.

# 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트



화면 1 live.com에 설치된 웹 가젯들



그림 1 비스타의 사이드바 가젯



# 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

## 그림 2 비스타의 사이드쇼 기능을 탑재한 ASUS 노트북

웹 가젯을 제작하는데 필요한 지식은 자바스크립트, 브라우저 DOM 구조, HTML, CSS가 전부다. 전부 그렇게 어려운 내용이 아니기 때문에 프로그래밍 경험이 있다면 쉽게 배울 수 있는 것이다. 앞으로 연재는 위의 네 가지 기본적인 지식을 알고 있다는 가정하에서 진행될 것이다. 이번 시간에는 가젯 작성에 도움이 될만한 자바스크립트 특징과 자바스크립트로 브라우저 DOM 구조를 조작하는 방법에 대해서 설명한다.

### var

자바스크립트는 타입 체킹이 없는(typeless) 언어다. 모든 데이터 타입은 var에 저장된다. 이러한 특징은 필자처럼 강력한 타입 체킹을 하는 언어만 다뤄온 개발자들에게는 굉장히 이상하게 보인다. 정수도, 문자도, 실수도, 함수도 모두 var에 저장한다. <리스트 1>은 이러한 특징을 보여준다. 명심하자. 모든 데이터는 var에 저장된다.

#### 리스트 1 다양한 자료형

```
var i = 3;
var str = "abcdef";
var array = new Array();
var word = new Word();
var func = function(str)
{
    alert(str);
}
```

### var를 꼭 써야 하나요?

자바스크립트의 변수는 필요한 시점에 자동으로 생성된다. <리스트 2>는 <리스트 1>과 동일한 코드다. 실행해 보면 에러 없이 잘 실행되는 것을 볼 수 있다. var를 통해 명시적으로 선언하지 않더라도 필요한 시점에 자동적으로 변수가 생성된다.

#### 리스트 2 var 없이 선언한 변수들

```
i = 3;
str = "abcdef";
array = new Array();
word = new Word();
func = function(str)
{
    alert(str);
}
```

## 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

그렇다면 var는 필요 없는 키워드일까? 물론 아니다. 위와 같은 전역 변수의 경우 var로 선언한 것과 그렇지 않은 것의 차이는 없다. 하지만 함수 안으로 들어오면 둘의 차이는 명확하다. <리스트 3>이 그 차이를 보여주는 코드다. foo1은 var없이 지역 변수를 선언했고, foo2는 var를 사용해서 지역 변수를 선언했다. 각 함수를 수행한 다음 사용한 지역 변수를 출력해 보면 foo1 함수의 지역 변수인 f1은 함수 스코프를 벗어 나서도 존재한다는 것을 알 수 있다.

### 리스트 3 var로 선언한 변수의 차이를 보여주는 코드

```
function foo1()
{
    for(f1=0; f1<10; ++f1)
        ;
}

function foo2()
{
    for(var f2=0; f2<10; ++f2)
        ;
}

foo1();
alert(f1);
foo2();
alert(f2);
```

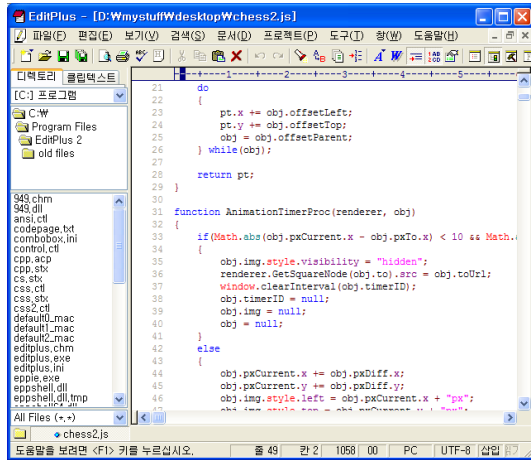
함수 안에서 변수를 선언할 때 var를 사용하지 않으면 전역(global) 변수 저장소에 변수가 저장된다. 함수가 끝나도 변수는 계속 생존하는 것이다. 이런 문제는 코드가 커지면 알아내기 쉽지 않기 때문에 반드시 변수를 선언할 때에는 var를 사용해서 선언하는 습관을 가지는 것이 좋다.

### 박스 1 자바와 자바스크립트

자바와 자바스크립트는 전혀 관련이 없는 언어다. 자바는 썬(Sun)에서 개발한 객체지향 언어이고, 자바스크립트는 ECMAScript 표준안을 구현한 스크립트 언어이다. 자바스크립트는 처음 넷스케이프에서 라이브스크립트(LiveScript)라는 이름으로 개발되었다가 후에 홍보 효과를 위해서 명칭을 자바스크립트로 변경한 것이다. 현재 IE7과 FireFox에서 동작하는 자바스크립트는 모두 ECMA-262 3차 개정판에 표준에 기반을 두고 있다.

# 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트


## 박스 2 가젯 개발에 사용하면 좋은 도구들



The screenshot shows the EditPlus 2 text editor with a file named 'chess2.js'. The code is a JavaScript function for an animation timer. It includes a 'do' loop that updates the position of a piece (obj) based on its offset and a 'while' loop that continues until the piece reaches its target position. The function also handles rendering and clearing the interval.

### 화면 2 EditPlus 실행 화면

Editplus는 정말 뛰어난 편집기다. 필자는 핵사 에디팅 기능이 없다는 점을 제외하고는 단점이 거의 없다고 생각한다. 가젯 개발에도 상당히 유용한데, 그 이유는 개발한 코드를 바로 브라우저에서 테스트 해 볼 수 있기 때문이다. 아직 손에 익은 에디터가 없는 개발자들에게는 추천해주고 싶은 제품이다.

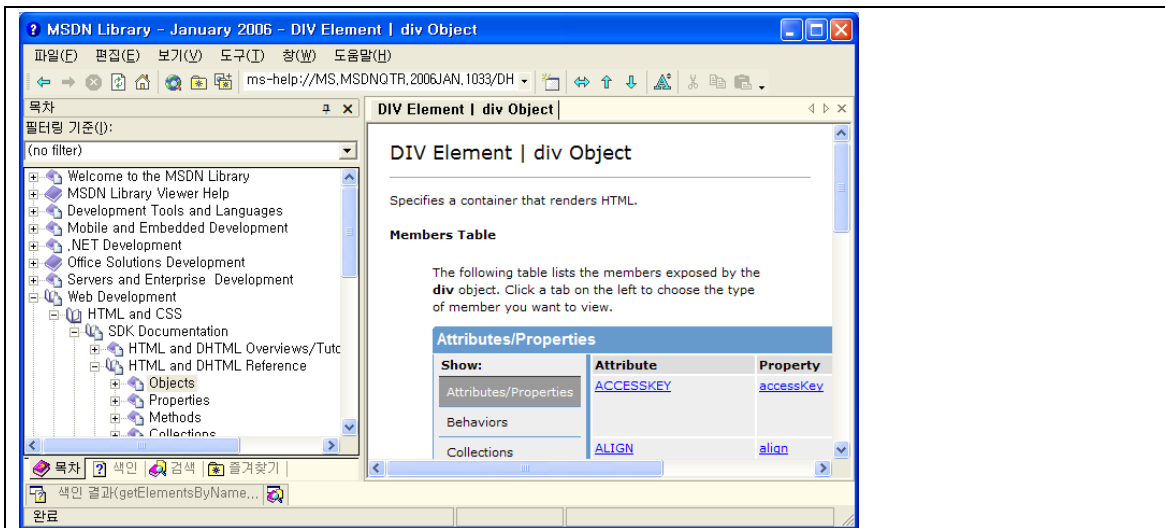


The screenshot shows the Mozilla Firefox browser displaying a website. The address bar shows 'http://www.imaso.co.kr/'. The page content includes a header for 'Software' and a main section titled 'micro software 23주년 기념' (micro software 23rd anniversary) featuring a 'ThinkPad' laptop. There is also a 'TOP STORY' section with an article about 'Office Open XML'.

### 화면 3 FireFox 실행 화면

FireFox는 워낙에 유명하기 때문에 많은 개발자가 사용하고 있으리라고 생각된다. 자바스크립트 오류를 쉽게 찾을 수 있고, DOM Inspector가 제공되기 때문에 개발자에게 장점이 많은 브라우저다. 특히나 IETab이라는 플러그인을 설치하면 IE도 같이 사용할 수 있기 때문에 더욱 유용하다.

## 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트



### 화면 4 MSDN 실행 화면

DOM 객체들에 관해서 가장 상세히 설명된 자료를 가르쳐달라고 한다면 필자는 주저 없이 MSDN을 추천해 줄 것이다. 목차에서 Web Development => HTML and CSS => SDK Documentation => HTML and DHTML Reference로 들어가면 된다. 모든 메소드와 각 객체가 가지고 있는 속성에 대해서 자세히 확인할 수 있다. MS 기술 자료에 회의적인 개발자들이 많이 있다. 왜냐하면 MS가 그 동안 웹 표준을 준수하지 않고 독자적인 노선을 많이 걸었기 때문이다. 하지만 여전히 많은 사람들이 IE를 사용하고 있음을 기억하자.

## 함수 객체, 스코프, 클로저

자바 스크립트의 큰 특징 중에 하나가 함수도 객체라는 점이다. 일반적으로 생성하는 함수도 모두 객체다. 그래서 <리스트 4>와 같이 함수를 변수에 저장하는 것도, 그 내용을 출력하는 것도 가능하다.

### 리스트 4 함수 생성 방법

```
var bar = function(obj)
{
    alert(obj);
}
alert(bar);
```

### 리스트 5 자바스크립트 스코프

```
var gv = 0;
function foo()
{
    var lv = 1;
```

## 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

```
for(var i=0; i<3; ++i)
{
    // scope 1
    var lv2 = 2;
}

// scope 2
}

// scope 3
```

자바스크립트는 C++에 비해서 단순한 형태의 스코프 룰을 가진다. 가장 큰 차이는 블록 스코프가 없다는 점이다. <리스트 5>을 보자. C++이라면 scope 2에서 i와 lv2는 찾을 수 없다. 왜냐하면 그것들의 생존 범위는 scope 1로 제한되기 때문이다. 하지만 자바스크립트는 그렇지 않다. scope 2에서 모든 지역 변수를 볼 수 있다. 블록 스코프가 없기 때문이다.

자바 스크립트에는 세 가지 형태의 스코프만 있다. 전역, 함수, eval 스코프가 그것이다. 전역 스코프는 말 그대로 전역 변수가 존재하는 영역이다. 함수 스코프란 함수가 실행될 때 생성되는 스코프다. 함수 내에서 var로 선언되는 모든 변수는 함수 스코프에 저장된다. 앞서 설명 했듯이 var를 사용하지 않으면 전역 스코프에 저장된다. 함수 스코프는 또한 계층적으로 구성된다. eval 스코프는 eval 함수로 수행되는 코드에 대한 스코프다.

### 리스트 6 중첩된 함수들

```
var gv = 0;
function foo1()
{
    var lv = 1;

    function foo2()
    {
        var lv2 = 2;

        function foo3()
        {
            var lv3 = 3;
        }
    }
}
}
```

객체는 언제나 생성될 수 있다. 함수도 객체이기 때문에 이 원칙이 동일하게 적용된다. 따라서 함수 내에서도 함수를 생성할 수 있다. <리스트 6>은 이러한 중첩 함수의 사용을 보여주고 있다. <리스트 6>의 변수들은 일반적인 C++의 블록 스크립트와 동일하게 해석하면 된다. foo3 내에서는 모든 변수를 볼 수 있다. 하지만 foo1 스코프 내에서는 lv2와 lv3을 볼



## 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

수 없다.

### 리스트 7 누산기 생성 코드

```
function foo(n)
{
    var func = function(i)
    {
        return n += i;
    }

    return func;
}

var gen = foo(3);
```

<리스트 7>은 폴 그레이엄의 책, "해커와 화가"에 나와서 유명해진 코드다. foo는 누산기를 생성해서 리턴하는 함수다. 누산기는 foo로 들어간 인자 n에 누산기로 들어온 인자 i만큼 누적 시킨 결과를 리턴 하는 함수다. <리스트 7> 에서 gen(3)을 실행하면 6이, 다시 gen(2)를 실행하면 8이 리턴 된다.

<리스트 7>은 자바스크립트의 큰 특징인 클로저의 대표적인 예다. 클로저란 함수 스코프가 함수 수행 후에도 사라지지 않는 것을 의미한다. foo 수행이 끝나면 foo 스코프는 사라져야 한다. 또한 그 스코프에 저장된 n이란 변수도 사라지는 것이 일반적이다. 하지만 여기서는 n은 사라지지 않는다. 왜냐하면 전역 변수인 gen이 func를 참조하고 있고, func의 스코프가 foo의 스코프를 참조하고 있기 때문이다. 스코프와 클로저에 대한 좀 더 자세한 내용을 알고 싶다면 참고 자료의 "자바스크립트 클로저"와 "ECMAScript 표준안"을 참고하자.

### 객체 생성 방법

자바스크립트로 객체를 생성하는 방법은 무척 간단하다. <리스트 8>과 같이 생성자를 사용하는 방법이 일반적이다. 함수 내부에서 this의 멤버로 데이터를 저장하면 멤버 변수가 된다. 배열에 여러 개의 객체를 생성하기 위해서는 마지막 줄과 같이 하면 된다.

### 리스트 8 Word 객체 생성자

```
function Word(name, meaning)
{
    this.name = name;
    this.meaning = meaning;
}

var w = new Word("apple", "사과");
```

## 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

```
alert(w.name + " " + w.meaning);
w.meaning = "먹는 사과";

var arr = new Array(new Word("apple", "사과"), new Word("cat", "고양이"));
```

자바스크립트에서 객체를 만드는 다른 방법으로 JSON이 있다. JSON은 자바스크립트 객체 표기법(Javascript object notation)의 약자다. 중괄호({})로 묶인 부분이 하나의 객체를 나타낸다. 각 요소는 콜론(:)을 기준으로 멤버 명, 데이터로 구분된다. 끝으로 배열을 만들기 위해서는 대괄호([])를 사용해서 표기한다. <리스트 9>는 <리스트 8>을 JSON 표기법을 사용해서 바꾼 것이다.

### 리스트 9 JSON을 사용하는 방법

```
var w = {name:"apple", meaning:"사과"};
alert(w.name + " " + w.meaning);
w.meaning = "먹는 사과";

var arr = [ {name:"apple", meaning:"사과"},
            , {name:"cat", meaning:"고양이"} ];
```

## 객체에 메소드 추가하기

앞서 자바스크립트의 함수도 객체라는 점을 배웠다. 따라서 객체에 메소드를 추가하는 것은 멤버 변수를 쓰는 것과 크게 다르지 않다. <리스트 10>은 이러한 기본적인 원칙 그대로 메소드를 추가한 것이다. this.GetArea에 함수를 할당한다. 하지만 이 방법은 묵시적으로 클로저를 생성한다는 단점이 있다. this.GetArea 함수가 Rect 함수의 스코프를 참조하기 때문이다. 또한 이 방법은 객체가 생성될 때마다 GetArea 함수를 매번 새로 만든다. 동일한 기능을 하는 함수가 동적으로 계속 생성되는 것이다. 이것 또한 낭비라 할 수 있다.

### 리스트 10 멤버 변수와 동일한 방법으로 메소드를 추가하는 코드

```
function Rect(width, height)
{
    this.width = width;
    this.height = height;

    this.GetArea = function()
    {
        return this.width * this.height;
    }
}

var r = new Rect(3, 4);
alert(r.GetArea());
```

## 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

똑똑한 독자라면 벌써 <리스트 11>과 같은 코드를 생각하고 있을 것이다. 함수를 외부로 분리해서 동적으로 생성하지 않도록 만든 것이다. 이 경우 클로저가 생성 되지 않고, 함수도 여러 개가 생성되지 않기 때문에 정답처럼 보인다. 하지만 아직도 낭비가 있다. 바로 this.GetArea가 낭비다. 이 것은 RectGetArea를 참조하는 레퍼런스다. 객체가 생성될 때 마다 동일한 곳을 참조하는 레퍼런스가 계속 생기는 것이다. 이러한 문제를 해결하기 위해서 생긴 키워드가 prototype이다. <리스트 12>는 prototype을 사용해서 메소드를 추가하는 방법을 보여준다. JSON을 사용하는 경우에는 <리스트 13 >과 같은 방법으로 멤버를 추가하면 된다.

### 리스트 11 함수를 동적으로 생성하지 않는 코드

```
function RectGetArea()
{
    return this.width * this.height;
}

function Rect(width, height)
{
    this.width = width;
    this.height = height;

    this.GetArea = RectGetArea;
}
```

### 리스트 12 prototype을 사용해서 메소드를 추가하는 방법

```
function Rect(width, height)
{
    this.width = width;
    this.height = height;
}

Rect.prototype.GetArea = function()
{
    return this.width * this.height;
}
```

### 리스트 13 JSON을 사용할 때 메소드를 추가하는 방법

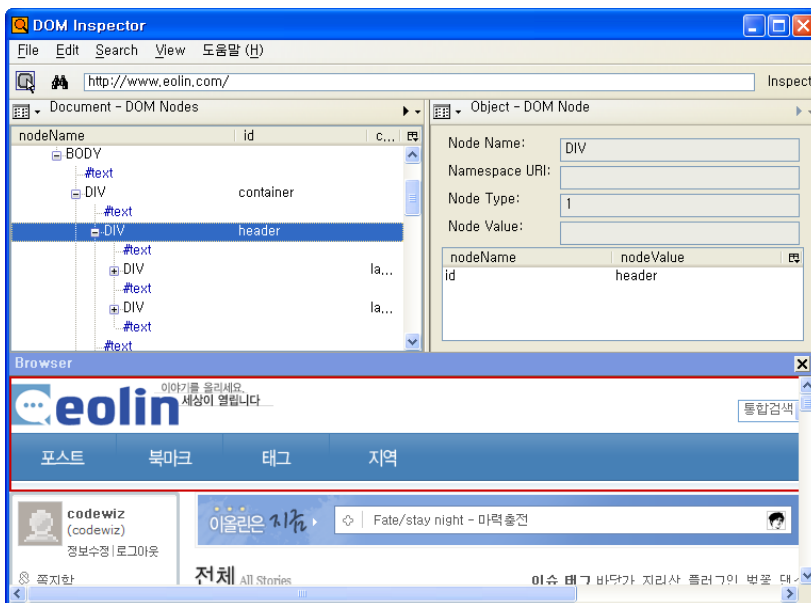
```
function RectGetArea()
{
    return this.width * this.height;
}

var r = {width:3, height:4, GetArea:RectGetArea};
alert(r.GetArea());
```

# 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

## DOM

DOM이란 Document Object Model의 약자다. 이는 HTML이나 XML 문서를 표현하는 한 가지 방법이다. 해당 문서의 구조를 모두 메모리에 올려놓고, 트리 형태로 손쉽게 사용할 수 있도록 한 것이 특징이다. <화면 5>는 FireFox의 DOM Inspector를 사용해서 eolin.com을 분석해 본 화면이다. DOM Inspector는 FireFox의 기본 설치에서 제거되어 있다. DOM Inspector를 사용하려면 반드시 설치할 때에 사용자 설정으로 들어가서 같이 설치하도록 해주어야 한다. 같이 설치했다면 FireFox의 도구 메뉴에서 DOM Inspector를 선택하면 사용할 수 있다.

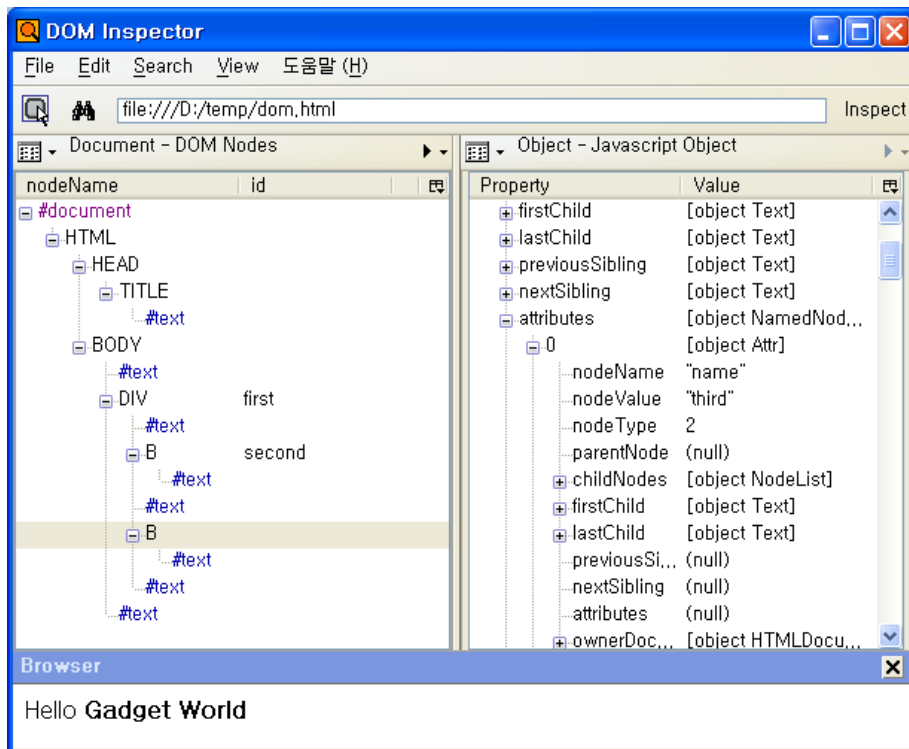


화면 5 FireFox DOM Inspector

## 리스트 14 DOM 테스트 HTML

```
<html>
<head>
<title>DOM 테스트</title>
</head>
<body>
<div id="first">
  Hello
  <b id="second">Gadget</b>
  <b name="third">World</b>
</div>
</body>
</html>
```

## 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트



화면 6 테스트 HTML의 DOM 구조

### DOM 노드 찾기

DOM 구조를 탐색하기 위해서 우리가 가장 먼저 익혀야 할 것은 DOM 노드를 찾는 일이다. 노드의 ID를 통해서 찾는 방법이 가장 많이 사용된다. `getElementById` 메소드를 사용하면 이러한 작업을 할 수 있다. 아래는 `getElementById` 메소드의 원형이다.

```
oElement = document.getElementById(sIDValue)
```

`sIDValue`에 찾고자 하는 DOM 노드의 ID를 넣어주면 해당 노드가 `oElement`에 들어온다. <리스트 14>의 `first` 노드를 찾고 싶다면 `var node = document.getElementById("first")`와 같이 사용하면 된다.

```
collobjects = document.getElementsByName(sNameValue)
```

`getElementsByName`은 이름(name)을 통해서 노드를 찾는 메소드다. 이름의 경우 중복될 수 있기 때문에 같은 이름을 가진 모든 노드를 찾아서 리턴해 준다. 리턴되는 `collobjects`는 배열이다. <리스트 14>의 `third` 노드를 찾고 싶다면 `var nodes = document.getElementsByName("third")`와 같이 사용하면 된다. `third`노드 객체는 `nodes[0]`에

## 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

들어있다.

IE에서의 `getElementsByName`은 동작이 조금 이상하다. 위에서 `third` 노드를 찾아서 출력하려면 에러가 난다. IE에서는 B태그가 `name` 속성을 지원하지 않기 때문에 `third`란 이름을 가진 노드를 찾지 못해서 에러가 나는 것이다. IE에서 `name` 속성을 지원하는 태그는 사용자 입력을 받는 `input` 관련 태그들뿐이다. `name` 속성을 지원하지 않는 태그들은 `getElementsByName`으로 검사할때 `id`속성을 사용해서 검사한다. 반면에 FireFox는 우리가 예상한 대로 모든 태그에 대해서 일관되게 `name` 속성을 사용해서 검사해 준다. 두 브라우저의 동작 방식이 틀리기 때문에 `getElementsByName`을 사용할 때에는 신경 써서 테스트를 해야 한다. 가장 좋은 방법은 이 메소드를 사용하지 않는 것이다.

```
coll0bjects = object.getElementsByTagName(sTagName)
```

마지막 함수는 태그 이름을 통해서 노드를 찾는 함수다. `getElementsByTagName`은 `sTagName`과 일치하는 태그의 DOM 노드를 모두 찾아서 반환해 준다. 앞서 살펴본 것과 달리 이 메소드는 `object`의 멤버라는 점에 주의해야 한다. <리스트 14>에서 `div` 태그를 모두 찾고 싶다면 `var nodes = document.body.getElementsByTagName("div")`와 같이 사용하면 된다. html에서 같은 태그를 가진 노드는 무수히 많기 때문에 이 방법은 그다지 실용적이지 않다. 또한 이러한 성능상의 이유 때문에 MS에서는 가젯 개발에 `getElementsByTagName`을 사용하지 않도록 권고하고 있다.

DOM 노드는 <화면 6>에 나타난 것처럼 트리 구조로 서로 연결되어 있다. 이러한 DOM 노드를 탐색하기 위해서는 <표 1>에 나와있는 속성들을 자주 사용한다. 특정 노드의 모든 자식 노드들은 `childNodes`를 참조하면 된다. `childNodes[0]`이 첫 번째 자식 노드를 가리킨다. `parentNode`는 자신의 부모 노드를 가리킨다. `nodeValue`는 해당 노드의 값이다. 이 값을 변경하면 화면 나타나는 글자가 바뀐다.

표 1 DOM 노드의 공통된 속성들

속성명	역할
<code>childNodes</code>	모든 자식 노드를 가진 배열
<code>parentNode</code>	부모 노드
<code>nodeName</code>	노드 태그명
<code>nodeValue</code>	노드 값
<code>nodeType</code>	3인 경우 텍스트 노드, 1인 경우 태그(element) 노드

# 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

## DOM 노드 생성

DOM 노드를 생성하기 위해서는 `createElement`나 `createTextNode` 메소드를 사용하면 된다. `createElement` 메소드는 태그 노드를 만드는 역할을 하고, `createTextNode`는 텍스트 노드를 만드는 역할을 한다. 각 메소드의 원형은 아래와 같다.

```
oElement = document.createElement(sTag)  
oTextNode = document.createTextNode( [sText] )
```

`div` 태그 노드를 만들고 싶다면 `document.createElement("div")`, `img` 태그 노드를 만들고 싶다면 `document.createElement("img")`와 같이 사용한다. 비어있는 텍스트 노드를 만드려면 `document.createTextNode()`, `Hello`라는 문자열을 가진 텍스트 노드를 만들고 싶다면 `document.createTextNode("Hello")`와 같이 사용한다.

`document.createElement`, `document.createTextNode`를 매번 입력하는 작업이 무척 귀찮다. 그래서 필자는 <리스트 15>과 같은 래퍼 함수를 만들어서 사용했다. 익숙해지면 코드가 더 읽기 쉽고 간결해진 느낌이 든다. 일부 개발자들은 `$`과 같은 기호로 함수 명을 지어서 더 간단하게 사용하기도 한다.

### 리스트 15 `createElement`, `createTextNode` 래퍼 함수들

```
// 태그 노드를 생성한다.  
function Node(tag)  
{  
    return document.createElement(tag);  
}  
  
// 텍스트 노드를 생성한다.  
function TNode(text)  
{  
    return document.createTextNode(text);  
}  
  
// 특정 id를 가진 노드를 찾는다.  
function FNode(id)  
{  
    return document.getElementById(id);  
}
```

## DOM 노드 추가, 삭제

지금까지 한 모든 작업을 실제로 HTML 문서에 반영하기 위해서는 기존의 DOM 객체의 자

## 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

식으로 추가하거나 삭제를 해야 한다. 이러한 작업에 사용하는 함수를 살펴보자.

**oElement = object.appendChild(oNode)**

appendChild 메소드는 object의 자식으로 oNode를 추가하는 역할을 한다. 추가한 노드 (oNode)를 리턴 한다. <리스트 14>의 first 노드에 "!!!"를 문자열로 가지는 텍스트 노드를 추가하기 위해서는 다음과 같이 하면 된다.

```
var firstNode = document.getElementById("first"); // first 노드를 찾는다.  
var node = document.createTextNode("!!!"); // 추가할 텍스트 노드를 만든다.  
firstNode.appendChild(node); // 노드를 추가한다.
```

**oElement = object.insertBefore(oNewNode [, oChildNode])**

insertBefore 메소드는 object의 자식으로 oNewNode를 추가하는 역할을 한다. appendChild와의 차이점은 oChildNode를 지정할 경우 해당 노드 앞에 추가를 한다는 점이다. oChildNode를 지정하지 않으면 appendChild와 동일하게 동작한다. 추가한 노드 (oNewNode)를 리턴 한다. <리스트 14>의 first 노드 앞쪽에 ">>>"를 문자열로 가지는 텍스트 노드를 추가하기 위해서는 다음과 같이 하면 된다.

```
var firstNode = document.getElementById("first"); // first 노드를 찾는다.  
var node = document.createTextNode(">>>"); // 추가할 텍스트 노드를 만든다.  
firstNode.insertBefore(node, firstNode.childNodes[0]); // 노드를 추가한다.
```

**oRemove = object.removeChild(oNode)**

removeChild 메소드는 특정 자식 노드를 제거한다. object의 자식 중에 oNode를 제거한다. 제거된 노드(oNode)를 리턴 한다. <리스트 14>의 second 노드의 첫 번째 자식인 텍스트 노드를 제거하려면 아래와 같이 하면 된다.

```
var secondNode = document.getElementById("second"); // second 노드를 찾는다.  
secondNode.removeChild(secondNode.childNodes[0]); // 첫 번째 자식 노드를 제거한다.
```

**oRemoved = object.removeNode( [bRemoveChildren])**

removeNode는 자기 자신을 DOM 트리에서 제거하는 메소드다. bRemoveChildren은 자식 노드도 모두 지울지 말지를 결정하는 값이다. true를 넘기면 자식 노드도 모두 삭제되고, false를 넘기면 자식 노드는 DOM 트리에 그대로 남아 있게 된다. <리스트 10>의 second 노드를 삭제하려면 다음과 같이 하면 된다.



# 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트

```
var secondNode = document.getElementById("second"); // second 노드를 찾는다.  
secondNode.removeNode(true); // second 노드를 제거한다.
```

## 도전 과제

지금까지 배운 지식을 토대로 자바스크립트 간단한 달력을 만들어 보자 (<http://www.jiniya.net/tt/421> 참고). 거창한 것은 아니고 페이지를 열었을 때 달력이 보인다면 성공한 것이다. <리스트 16>은 간단한 뼈대 코드다. 여기다 살을 붙여서 멋진 달력을 만들어 보자.

### 리스트 16 달력 뼈대 코드

```
<html>  
<head></head>  
<body>  
<div id="cal"></div>  
<script>  
function Calendar(year, mon)  
{  
  this.node = ???; // 여기다 만들어진 노드 루트 저장  
}  
  
var cal = new Calendar(2007, 2); // 2007년 2월 달력 생성  
var obj = document.getElementById("cal"); // 추가할 노드 찾기  
obj.appendChild(cal.node); // 달력 추가  
</script>  
</body>  
</html>
```

## 참고자료

- 참고자료 1. 자바스크립트 클로저 - [http://www.jibbering.com/faq/faq\\_notes/closures.html](http://www.jibbering.com/faq/faq_notes/closures.html)
- 참고자료 2. Paul Graham <<해커와 화가>> 13장 - [http://lib.aldebaran.ru/author/graham\\_paul/graham\\_paul\\_hackers\\_and\\_painters\\_big\\_ideas\\_from\\_the\\_computer\\_age/graham\\_paul\\_hackers\\_and\\_painters\\_big\\_ideas\\_from\\_the\\_computer\\_age\\_13.html](http://lib.aldebaran.ru/author/graham_paul/graham_paul_hackers_and_painters_big_ideas_from_the_computer_age/graham_paul_hackers_and_painters_big_ideas_from_the_computer_age_13.html)
- 참고자료 3. JSON - <http://www.json.org>
- 참고자료 4. ECMAScript 표준안(ECMA-262 3차 개정판) - <http://www.ecma-international.org/publications/standards/Ecma-262.htm>
- 참고자료 5.

## 나만의 윈도우 라이브 가젯 만들기: 이상한 나라의 자바 스크립트