실전 HTML5 가이드

윤석찬, 신현석, 정찬명, 경준호, 권정혁

한국 웹 표준 커뮤니티 일동
Web Standards Korea

서문

HTML5는 웹 개발 생산성 향상 및 차세대 웹 애플리케이션 플랫폼으로서 더 나은 사용자 경험을 위해 함께 만들어 나가는 개방형 웹 표준입니다. 2004년부터 전 세계 웹 개발자들이 직접 참여하여 만들어온 표준이기도 합니다.

2005년부터 국내에 웹 표준을 정착 시켜온 웹 표준 커뮤니티에서는 과거 국내 웹 표준 기술 도입 시 정보 부족 경험을 토대로 국내 웹 종사자들에게 새로운 웹 표준인 HTML5을 소개하기 위해 이 가이드를 준비하였습니다.

본 가이드에는 HTML5의 배경과 소개 웹 개발 방식의 변화, 새로운 마크업과 웹 폼 기능, 다양한 웹 애플리케이션 부가 API 그리고 CSS3와 모바일 기반 웹앱 개발에 대한 내용을 두루 넣었습니다. HTML5 오픈 콘퍼런스의 강사들이 직접 집필하였으며 누구나 읽을 수 있도록 크리에이티브 커먼즈 라이센스 하에 무료로 제공 합니다.

본 가이드는 2005년 무료 배포된 '실전 웹 표준 가이드'를 잇는 공개 자료로서 많은 관련 책들이 출판되겠지만 처음 공부를 시작하시는 분들게 부담없이 도움이 되기 위해 만들었습니다. 기꺼이 집필을 맡아준 커뮤니티 멤버들에게 감사드리고 다른 참여자에 의해 더욱 개선된 자료로 거듭날 수 있기를 바랍니다.

대표 저자 윤 석 찬

저자 소개

윤 석 찬 2002년부터 한국 Mozilla 커뮤니티 리더 및 Firefox 한국어 버전 개발에 관여하고 있다. 다음커뮤니케이션에서 사내 기술 전략 및 오픈 API 서비스를 담당했으며, 현재 서울대 박사과정에 재학 중이다.

```
http://channy.creation.net
Twitter: @channyun Me2day: channy
```

신 현 석 2005년경부터 웹표준을 적용한 사이트를 구축하면서 이를 효과적으로 적용하는 방법에 대해서 고민해 왔다. 현재는 Opera Software에서 웹 에반젤리스트로 활동하면서 웹표준과 웹접근성의 중요성을 알리는 노력을 하고 있다.

```
http://hyeonseok.com
Twitter: @hyeonseok, Me2day: hyeonseok
```

정 찬 명 웹 표준, 웹 접근성, 유니버설 디자인에 관심이 많고 현재 NHN에서 오픈소스 XE의 UI 개발업무를 담당하고 있다.

```
http://naradesign.net
Twitter: @naradesign, Me2day: naradesign
```

경 준 호 firejune.com을 운영중인 블로거, 프론드-엔드 엔지니어, 자바스크립트를 중심으로한 리치 웹 애플리케이션 개발에 관심과 흥미를 가지고 있다.

```
http://firejune.com  Twitter: @firejune
```

권 정 혁 xguru.net 을 운영중인 블로거. 아이폰/안드로이드와 같은 모바일 환경에서의 다각적인 웹 활용에 대해 연구중이다.

```
http://xguru.net  Twitter: @xguru
```

목차

1. HTML5 소개

HTML5는 웹 표준 기구인 W3C에서 만들고 있는 차세대 웹 표준안으로서 마이크로소프트, 모질라, 애플, 구글, 오페라 등 모든 웹 브라우저 벤더가 참여하고 있는 산업 표준이다.

2004년 웹 브라우저 벤더와 개발자 커뮤니티가 주축이 된 웹 하이퍼텍스트 애플리케이션 워킹그룹(WHATWG)의 초안으로부터 시작된 이 표준안은 웹 브라우저 호환성, 구조적이고 의미적 마크업 및 편리한 웹폼 기능을 제공하여 웹 개발자들의 생산성을 높임과 동시에 리치 웹 애플리케이션을 개발 할 수 있는 다양한 자바스크립트 API를 포함하고 있다. 2007년부터 W3C의 HTML W/G에서 HTML5 표준안으로 개발되고 있는 동시에 최신 버전의 웹 브라우저에서 빠르게 탑재되고 있어 차세대 웹 서비스 개발의 주요 기술로서 각광 받을 전망이다.

이 장에서는 HTML5가 만들어질 수 밖에 없었던 배경과 간단한 스펙 소개 그리고 웹 개발자로서 향후 접근 방법 및 웹 개발의 방향에 대해 공유하고자 한다.

1.1 HTML 5 의 배경

HTML5 표준안의 가장 큰 목적은 과거 HTML의 호환성을 유지하면서 웹 개발자들이 실질적으로 부딪히는 문제를 해결 하는 것이다.

웹을 통한 정보 공유가 폭발적으로 성장할 수 있었던 것은 HTML을 통해 정보(컨텐츠)와 의미(마크업)을 함께 손 쉬운 텍스트로 편집하도록 함으로서 쉽게 배우고 쓸 수 있었기 때문이다. HTML의 이런 단순함은 웹 상에 사람이 참여하는 토대를 낳게 하기에 충분했다.

하지만 이러한 장점에도 불구하고 90년대 후반 웹 브라우저 업체의 점유율 전쟁 중에 상용 브라우저 벤더들의 비표준 태그들이 남발되면서 HTML의 기본 정신을 훼손되었다. 실질적으로 IE vs. Netscape의 사이에서 피해를 본 것은 '웹 개발자'들이었다. 크로스 브라우징(Cross Browsing)라는 기법 때문에 고생을 했는가 하면 IE의 독주 상태 때문에 웹 서비스의 혁신이 늦어졌고 사실상 프론트 엔드의 기술 혁신은 플래시 같은 서드파티 플랫폼으로 넘어가 버렸다.

게다가 웹 표준 기구인 W3C는 견고한 웹 문서를 제공한다는 꿈을 가지고 XML을 기반으로 하는 XHTML로의 전환을 꾀하였다. 따라서 HTML은 4.01 버전을 끝으로 더 이상 업그레이드 되지 않는 낡은 표준으로 남았다.

1.1.1. 웹 표준과 웹 2.0

2000년대 중반부터 혁신의 단초를 제공하게 된 것은 바로 웹 표준과 웹 2.0이다.

우선 구글 같은 검색 엔진과 검색 광고의 성장과 특히, 블로그와 같은 사용자 생산 콘텐츠를 잘 검색하기 위하여 HTML과 CSS 레이아웃을 통한 웹 표준 기법이 각광 받기 시작했다. 이른바 구조(Structure)와 표현(Presentation) 그리고 동작(Behavior)를 분리하여 검색 크롤러(기계)가 콘텐츠를 읽고 쓸 수 있도록 하는 것은 매우 중요한 시작점이 되었다.

특히 이러한 방식은 웹 개발에 있어서 개발자와 디자이너 간의 역할 분담을 명확히 하고 코드 유지 보수 및 생산성에 큰 영향을 미쳤다. 게다가 장애인을 위한 웹 접근성에도 매우 뛰어난 개발 방법론이 되었다. 2004년부

터 실리콘밸리의 많은 웹2.0 스타트업들이 웹 표준 기법을 기반으로 다양한 웹 서비스를 선보이기 시작하였고 국내에서도 많은 영향을 미쳤다.

웹 2.0의 주 개념인 '플랫폼으로서 웹'은 웹 그 자체를 소프트웨어로 보는 웹 애플리케이션 시대를 열었다. 대표적인 기술이 바로 Ajax(Asynchronous JavaScript and XML)로서 지메일과 구글맵이 그 시초를 이루었다. 기존의 문서 형식의 정보의 제공이라는 틀에서 벗어나 데스크톱 소프트웨어와 같은 사용자 경험을 제공하는 것이다.

또한, 오픈API라는 데이터 기반 서비스는 전문 개발자뿐만 아니라 전문 사용자까지 웹 플랫폼에 끌어들였다. 오픈 API를 이용하면 자신의 블로그나 홈페이지에 네이버나 다음의 검색 결과나 구글 맵의 위성 지도, 이베이의 중고 상품 목록 같은 것을 쉽게 추가할 수 있다.

이미 데스크톱 소프트웨어 플랫폼 벤더들은 공개 웹 기술을 웹 애플리케이션에 접목하는 시도를 계속 하고 있다. XML과 (X)HTML, CSS, 자바 스크립트 같은 웹 표준 기술들을 리치 인터넷 애플리케이션(Rich Internet Application)을 만드는 데 사용하기 시작한 것이다. 대표적으로 모질라의 파이어폭스 확장 기능, 야후! 위젯, 마이크로소프트의 실버라이트(Silverlight), 어도비의 플렉스(Flex) 및 AIR 등이 여기에 속한다. 애플의 경우, Mac OS의 대시보드 위젯과 사파리에서 구동 가능한 웹 애플리케이션을 아이폰(iPhone)에서도 실행 할 수 있도록 하고 있다.

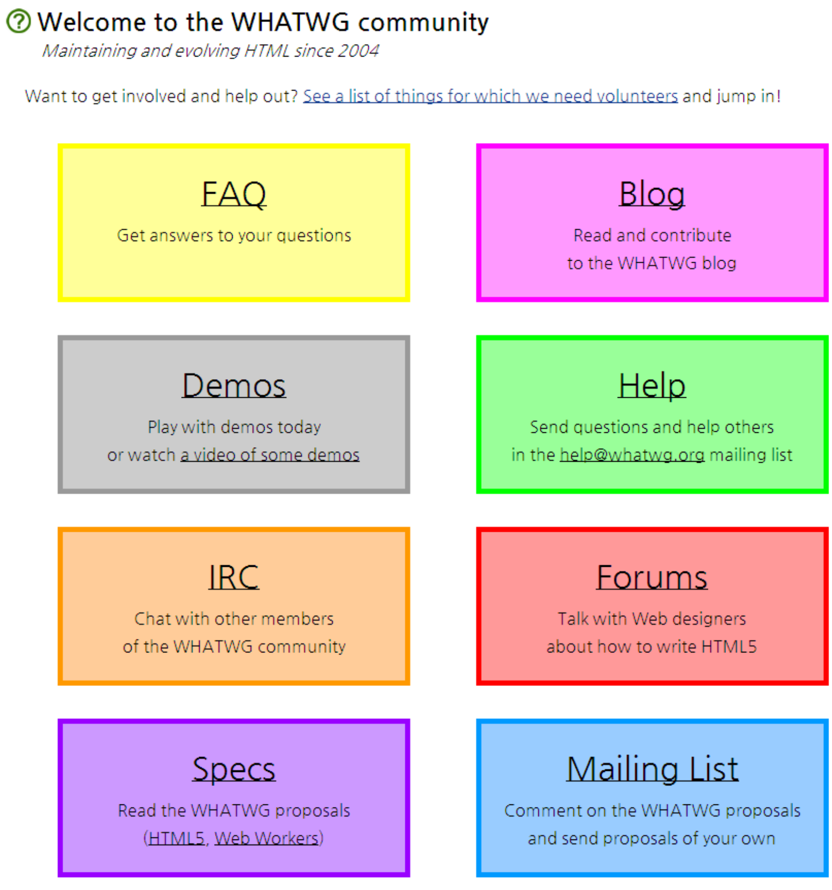
이들 응용 소프트웨어의 대표적인 특징은 XML 혹은 (X)HTML로 사용

자 인터페이스를 만들며 CSS로 디자인 및 스타일을 정의하고 자바 스크립트로 기능을 제어 하는 전형적인 웹 기술의 성공을 벤치마킹 했다는 데 있다.

1.1.2 HTML5의 시작

이러한 웹의 기술적 혁신은 웹 브라우저 업계에도 역시 시작 되었다. 오픈 소스 프로젝트인 모질라(Mozilla) 커뮤니티에서 개발한 파이어폭스와 애플의 사파리, 오페라 그리고 구글 크롬에 이르기 까지 2004년부터 다양한 웹 브라우저들이 시장에 쏟아져 나오기 시작했다.

2004년 비IE 브라우저 세계 점유율이 5% 안팎이던 것이 2010년 현재 거의 40%에 육박하고 있으며 유럽의 경우는 이미 50%를 넘었다. 웹 기술의 변화에는 이러한 마이너 웹 브라우저 업체의 혁신과 사용자들의 선택에 힘입은 바 컸으며 마이크로소프트가 2007년 IE 개발팀을 다시 만들 정도였다.



그러나, 웹 표준화 기구인 W3C는 이러한 변화를 수용할 준비를 하고 있지 못했다. 2004년 W3C의 한 워크샵에서 생긴 의견 차이 때문에 모질라,

애플, 오페라 등은 W3C 밖에서 새로운 버전의 HTML 표준을 준비하기 시작했다. W3C의 다른 표준화 기구 보다는 상대적으로 개방되어 있었지만, 다양한 웹 브라우저 환경에서의 웹 개발자의 고충과 웹 애플리케이션이라는 현실적인 변화를 받아들이지 못했다.

이들은 2006년 6월 웹 하이퍼텍스트 워킹그룹(WHATWG)이라는 공개 그룹을 형성하여 자신들이 만드는 새로운 표준안에 누구나 참여할 수 있도록 개방 하였다. W3C의 회원사 중심 표준안이 아닌 웹 개발자가 진정 원하는 표준을 만들기 위해서였다. 누구나 표준안 논의에 참여할 수 있었으며 이들은 오랜 공개 토론을 거쳐 Web Form 2.0과 Web Applications 1.0이라는 표준안을 만들어 냈다.

이들 표준안의 철학은 당시 전 세계 웹 사이트의 90%가 넘는 언어인 HTML을 혁신하자는 것이다. 웹 브라우저 업체 입장에서 W3C가 만들고 있던 XML기반 웹 표준(XHTML2.0, XPath, XML Events등)은 기존 웹 브라우저를 새로 작성해야 할 정도로 어려운 작업이라는 측면도 있었다. 무엇 보다 중요한 것은 기존 HTML이 가진 가치를 인정하고 장점을 그대로 살리면서 웹 브라우저 업체간 불명확했던 처리 방식을 재정의하고 새로운 마크업과 API를 통해 웹 개발자들이 콘텐츠 중심의 웹 어플리케이션 개발을 손쉽게 하려는 것이다.

1.1.3 Of the Web Developer, by the Web Developer and for the Web Developer

WHATWG 활동의 성공은 즉각 W3C에 영향을 주기 시작했다. 작년 10월 웹의 창시자이자 W3C를 이끌고 있는 팀 버너스 리(Tim Berners-Lee)는 ‘Reinventing HTML’이라는 글에서 XHTML의 전환 실패와 더불어 새 HTML 작업을 시작할 것을 천명하였다. 이에 제 3지대에서 활동하고 있던 WHATWG의 웹 개발자들과 벤더들은 W3C의 결정에 환영 하면서 2007년 3월 새로운 HTML 워킹 그룹에 하였다.

이 워킹 그룹 활동에는 몇 가지 고무스러운 점이 있다. 먼저 전직 IE 개발자 이며 최근 마이크로소프트의 IE7 이후의 개발을 총책임을 맡은 크리스 윌슨(Chris Wilson)이 워킹 그룹 의장이 되었다는 점이다. 또한, WHATWG의 표준 작업을 사실상 주도한 이안 힉슨(Ian Hickson)이 첫 표준 초안의 편집자가 된 것이다. 이안은 넷스케이프와 오페라를 거쳐 지금은 구글에서 풀타임 표준 작성가로 활동 중인 젊은 인재이다. 뿐만 아니라

WHATWG에서 활동하던 500여명의 웹 개발자들이 W3C의 초빙 전문가(Invited Expert)라는 제도를 활용해서 참여할 수 있게 되었고 메일링리스트도 완전 공개로 바뀌었다.

이러한 과감한 변화를 통해 W3C의 새 HTML 워킹 그룹은 새 표준의 이름을 'HTML5' 라고 명명 하고 WHATWG가 작업하던 대부분의 표준안을 그대로 수용하기에 이른다. 사실상 웹 개발자의 웹 개발자에 의한 웹 개발자를 위한 표준으로 거듭나게 된 순간이다.

Interaction domain

HTML Working Group Charter

The **mission** of the HTML Working Group, part of the HTML Activity, is to continue the evolution of HTML (including classic HTML and XML syntaxes).

Join the HTML Working Group.

End date	31 December 2010
Confidentiality	Proceedings are Public
Chair	*Chris Wilson, Microsoft,* *Sam Ruby, IBM*
Initial Team Contact (FTE %: 60)	*Dan Connolly, W3C/MIT and Midwest Web Sense,* *Michael Smith, W3C/Keio*
Usual Meeting Schedule	Teleconferences: up to 1 per week, as needed Face-to-face: up to 2 per year

많은 사람들이 HTML5에 대해서 회의적인 시각을 가지고 있는 것을 자주 본다. 그 대표적인 이유가 W3C 표준안이 되는 과정이 매우 길 뿐만 아니라 실제로 웹 브라우저에 적용되는 시기는 매우 오랜 기간이 걸릴 것이라는 이유에서다.

현실은 그렇지 않다. 이미 HTML5의 많은 기능들이 파이어폭스, 오페라, 사파리와 크롬에 이미 탑재되고 있으며 IE8과 IE9에서도 대거 포함될 것이 기정 사실화 되고 있기 때문이다. 본 가이드에서 앞으로 소개할 HTML5 관련 기술들이 실제로 다양한 웹 브라우저에서 볼 수 있게 된다는 것이다.

이러한 변화에도 불구하고 현재 국내에는 플래시나 실버라이트 등 각종 리치 인터넷 기술이 웹 애플리케이션의 미래인 듯 포장되고 있는 감이 없지 않다. 특히 아이폰이나 안드로이드 같은 스마트폰 애플리케이션이 모바일의 대세가 될 것처럼 여겨지는 부분도 존재한다.

하지만 웹이 가지고 있는 가장 큰 자산은 콘텐츠를 기반한 정보 공유의 수단으로 기본에 충실하면서 웹 애플리케이션 기능을 제공할 수 있는

HTML5의 등장으로 새로운 변화를 맞고 있다는 점이다. RIA에 비해 덜 풍부한 사용자 경험이나 낮은 개발 생산성으로 인해 웹의 낡은 애플리케이션 기술로 치부되어서는 안된다.

누구나 정보 공유와 애플리케이션 기능을 쉽게 만들고 제공할 수 있도록 웹 컨텐츠를 만들고 생산하는 고급 사용자와 웹 개발자들이 웹을 잘 가꾸어 나가야 할 책임을 느껴야 할 시점이다. HTML5가 중요한 것은 이러한 표준 웹의 근본적인 변화가 시도되고 있기 때문이고 지금이야 말로 우리가 함께 만들어 왔던 웹의 미래를 직면하게 되는 순간이다.

1.2 HTML5 개요

HTML5가 인터넷 업계에서 알려지게 된 계기는 바로 2009년 구글의 웹 개발자 콘퍼런스인 '구글 I/O'에서 자사의 서비스가 아닌 HTML5를 데모로 시연하면서 차세대 웹 기술로 지원하겠다는 천명을 하면서부터이다. 특히, 스티브 잡스가 애플 아이폰에 플래시 탑재를 거부하면서 대응 기술로 HTML5를 홍보하기 시작하였다.

또한, 수 많은 HTML5 데모들에서 기존의 RIA 기술을 능가할 만한 것을 보여줌으로써 마치 구글과 애플이 자사의 이익을 위해 플러그인 기반 RIA 기술의 대체 수단으로 홍보하고 있다는 생각이 널리 퍼져있다.

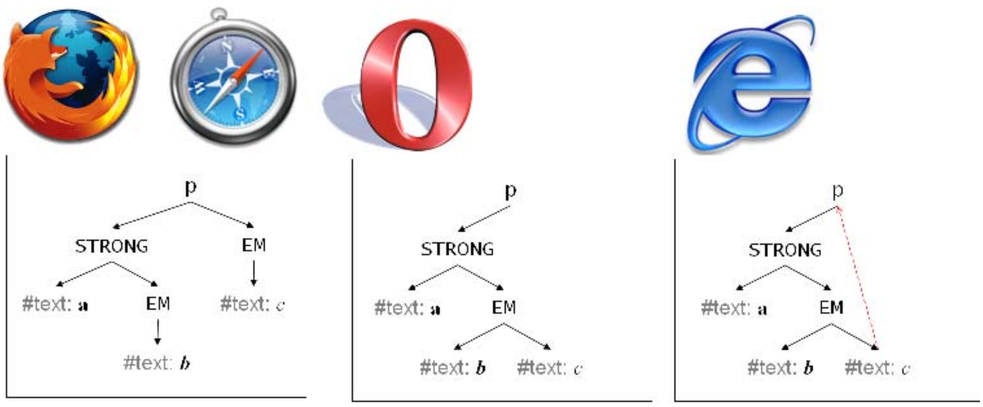
우리가 앞서 살펴 보았듯이 HTML5는 2004년 당시 시장 점유율 5% 미만이었던 마이너 웹 브라우저들이 웹 개발자들과의 토론을 통해 만들어 온 개방형 웹 표준으로서 웹 개발자의 생산성과 가치를 높이기 위한 다양한 기술들을 담았다. 이러한 점을 간략하게 살펴 보고자 한다.

1.2.1 디자인 원칙

W3C HTML5 W/G에서는 기존 표준 문서 외에도 웹 개발자의 이해를 돕기 위한 다양한 문서를 함께 만들고 있다. 그 중 HTML5 디자인 원칙이라는 문서에는 웹 표준을 만드는 데 있어, 의사 결정의 기본 원칙이 되는사항을 제시하고 있다.

첫째, 기존의 HTML 문법이랑 사용법을 최대한 지원하고 단계적 기능 축소(Graceful degradation)이 가능하도록 한다. <b>, <i> 같은 기존의 비

표준 태그의 사용도 용법을 정해 가능하게 했으며 <embed> 같은 이미 사용하던 표준도 재사용하도록 하여 웹 개발자들이 너무 문법에 억매이지 않도록 하는 '호환성(Compatibility)'을 제공한다.



웹 브라우저별로 <p>abc 대한 상이한 처리방법

둘째, 실제 웹 개발자들이 겪고 있는 가장 중요한 문제를 순위에 따라 나누되 문제점을 분리해서 독립적으로 해결 하는 유용성(Utility)의 원칙이다. 예를 들어, 웹폼(Web Form)에 email, number, date 같은 새로운 속성을 추가함으로써 사용자 입력 값의 유효성 확인에 드는 (항상 하는) 삽질을 줄일 수 있도록 하였다. <input> 태그에 datetime 속성을 넣어주면 웹 브라우저가 자동으로 달력을 표시해 준다. 또한 IE에서만 사용 가능 했던 contenteditable 속성이 표준화 되어 모든 HTML 요소를 사용자가 직접 편집할 수 있게 함으로서 위지윅 에디터의 호환성 문제도 사라질 것이다. 특히, 이미 웹 콘텐츠의 일부가 되어 버린 비디오와 오디오 콘텐츠 재생을 웹 브라우저에서 내부적으로 구현하여 보편적 접근이 가능하고 캔버스(Canvas)와 벡터 그래픽(SVG)를 통해 2차원 도표와 같은 콘텐츠도 마크업으로 표현 할 수 있도록 멀티미디어의 보편적 접근성을 높혔다.

셋째, 상호 호환성(Interoperability)으로 웹 브라우저가 상호 호환을 위해 최대한 자세하게 기술하되 오류 처리 방법을 명시하도록 하였다. HTML5의 기본 표준 문서 첫 부분은 웹 브라우저 간 HTML 문법 오류에 대한 자세한 사례와 이에 대한 브라우저의 처리 방법을 명시해 두었다. 따라서 웹 브라우저간 이러한 문법적 오류로 인해 웹 개발자들이 실질적인 어려움을 겪는 문제를 해결하였다.

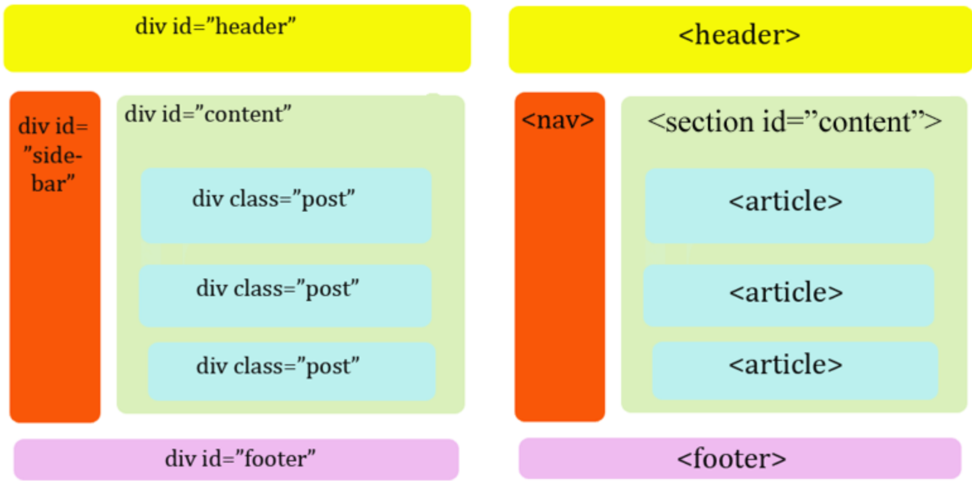
1.2.2 주요 기능

HTML5를 판별하는 기준은 바로 새로운 문서 형식(Doctype)에 있다. <!doctype html>을 선언함으로서 HTML5 문서로 인식할 수 있으며, 웹 브라우저에서는 가장 최신의 렌더링 엔진을 이용하게 된다.

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML5 마크업</title>
  </head>
  <body>
    <p>차세대 웹표준으로서 HTML5
  </body>
</html>
```

새로운 마크업

HTML5에서는 기존의 HTML4 보다 확장된 태그들을 지원한다. 특히, 문서 구조에 적합하게 header, footer, nav, section 같은 구조화 마크업을 사용할 수 있다. (자세한 내용은 2장에서 다룬다.)



HTML5의 새 구조적 마크업(출처: http://html5doctor.com/designing-a-blog-with-html5/)

특히, progress, time, mark, meter 등과 같은 의미 기반 태그들이 추가로 지원된다. 콘텐츠의 시맨틱 표현(Annotation)이 가능하도록 마이크로데이터(Microdata)와 RDFa라는 시맨틱 웹 기법도 포함하였다.

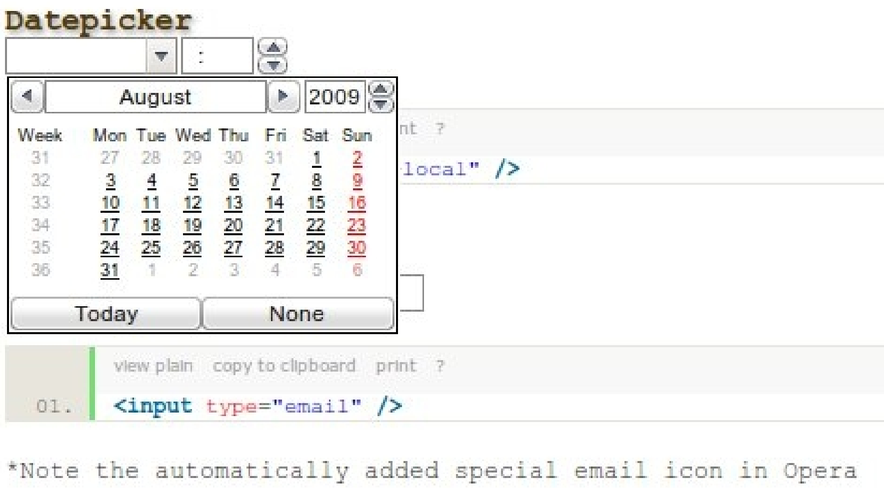
앞서 말한대로 비표준 문법이었으나 여전히 이용하고 있는<b>는 키워드,<i>는 학명에 사용하도록 하는 등 최대한 기존의 태그들을 그대로 이용할 수 있다. 대신 CSS로 완전 대체 가능한 big, center, font, s, strike 같은

스타일 기반 요소는 완전히 없어진다다. 또한, frame과 applet, acronym 같은 부정적인 요소들도 사용하지 않는다.

편리한 폼(Form) 속성

HTML5는 개발자의 수고를 덜어 줄 Form 기능 개선을 담고있다. input 태그의 각종 type 속성이 추가되어 다양한 기능을 제공해 준다.

datetime 속성값을 사용하면 달력을 웹 브라우저에서 제공해 주며, range 속성은 스크롤바를, url은 웹 사이트 목록, email은 메일 주소 유효성 확인을 해 준다. color 속성은 색상표를 별도 개발 없이 사용할 수 있다.



Form 양식은 모두 유효성 확인 기능을 켜거나 끌 수 있고, 정규식 표현을 사용할 수 있도록 하여 유효성 검증에 드는 자바스크립트 코드 개발 시간을 현저히 줄여 줄 것으로 기대한다.

리치 웹 애플리케이션

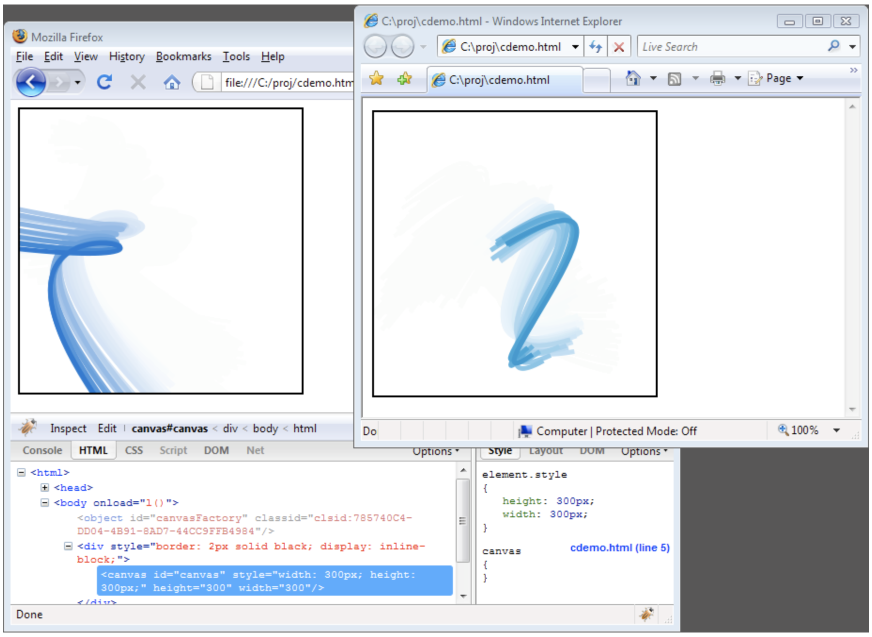
HTML 5는 웹 어플리케이션 작성에 도움을 줄 다양한 API를 제공 한다. 이는 새로 규정된 마크업 요소와 폼 속성들과 함께 더 좋은 애플리케이션 개발에 사용할 수 있다.

2차원 그래픽 API에 사용할 수 있는 canvas 요소와 내장 비디오 및 오디오 재생을 위한 video, audio 요소를 통해 멀티미디어 기능을 강화할 수 있다.

오프라인 웹 애플리케이션 지원을 위한 웹 페이지 내부 저장소

(AppCache). 키(Key)/값(Value) 기반 데이터 저장소(DOM Storage 및 IndexedDB)나 SQL 기반 데이터베이스 지원 기능(Web Databases API)를 이용하면 웹 기반 애플리케이션을 만들 수 있다.

또한, 웹 애플리케이션이 독립적으로 특정 프로토콜 및 미디어 형식을 등록할 수 있는 통신(Web Socket) API와 문서간 알림 기능(Sever-sent Event) 등으로 다양한 서버 통신을 처리 할 수 있다.



사용자 경험을 증대시키기 위한 contenteditable 속성과 함께 지원 되는 편집 API 및 draggable 속성과 함께 지원 되는 드래그앤 드롭 API 기능, 페이지 앞/뒤 네비게이션을 지원할 방문 기록 표시용 API 및 지역 정보 활용을 위한 지오로케이션(Geolocation) API등은 HTML5 지원 API로서 알려져 있다.

1.2.3 HTML vs. XHTML

2009년 7월 W3C에서는 XHTML2.0 W/G의 활동을 완전히 접었다. XHTML2는 XHTML1.0을 더 발전 시키기 위해 작업해온 표준안으로 HTML과 DOM, Form, Frames, Event 등 다양한 웹 요소들을 XML로 대체하기 위한 가장 큰 시도였다. 이에 따라 XForm, XFrames, XEvents 등의 표준도 함께 만들어졌다.

가장 큰 문제점은 HTML4와 XHTML1.0과 전혀 다른 새로운 표준이어서 하위 호환성에 대한 보장이 거의 없는 루비콘 강을 건너는 작업이었다

는 것이다.

XHTML2.0은 사라졌지만 XHTML이 사라진 것은 결코 아니다. XHTML이 가지고 있는 견고한(well-formed) 문서 규격은 대형 웹 서비스에서 이용되는 마크업의 개발자 생산성 및 유지 보수에 도움이 되고 있는 점이 입증되었기 때문이다. HTML5에서도 여전히 XHTML을 지원하며 application/xhtml+xml의 형식을 선언한다면 XHTML5로 렌더링한다.

```
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>… </title>
 </head>
 <body>
  <h1>…</h1>
  <p>…</p>
 </body>
</html>
```

아래 코믹은 DOM Scripting의 저자인 Jeremy Keith가 작성하고 Brad Colbow가 그린 "마크업의 오해"(Misunderstanding Markup: XHTML 2/HTML 5 Comic Strip)라는 만화로서 XHTML5에 대해 소개하고 있어 독자들의 이해를 돕기 위해 게재하였다.


MISUNDERSTANDING MARKUP
WRITTEN BY JEREMY KEITH
ADAPTED BY BRAD COLBOW
얼마전 W3C가 XHTML2 활동을 올해 안에 접는다고 발표했습니다.
전 솔직히 이 소식이 반가웠습니다.
그런데 많은 분들이 XHTML이 끝났다고 오해 하셨습니다.
전혀 다른 두 가지 기술을 비슷한 이름으로 썼기 때문에 오해가 생겨난 것입니다.
대표적으로 자바스크립트와 자바가 그렇습니다.
둘은 완전 달라 사실 햄과 햄스터랑 비슷하죠.
XHTML2가 그래요!
XHTML 1.0와 1.1와 XHTML2는 전혀 달라요.
헷갈리시죠?
SOY MILK
XHTML2


XHTML1.0은 HTML4를 XML 문법으로 바꾼 겁니다
HTML 4
XHTML 1
• 태그와 속성을 소문자로 쓴다.
• 속성에 따옴표를 쓴다.
• 태그를 시작했으면 꼭 닫는다. 〈p〉, 〈/p〉
• BR, IMG 등은 슬래시로 닫는다. 〈br /〉
XTHML 1.1은 다 똑같은데 단지 XML 문서 형식으로 제공한다는 단서를 달았습니다.
FAT FREE
근데 XHTML 2는 HTML4와는 아무 관련이 없습니다.
SOY MILK
XHTML2
호환성은 고려하지 않고 이론적으로 "순수한" 태그 용어를 만들려는 새로운 시도 입니다.
XHTML 2가 없어진다고 XHTML이 끝장나는 걸까요?
전혀 아닙니다.
다양한 의견들을 모아 멋진 결론을 내기 위한 작업이 진행됐습니다.
SEE LINKS 1 AND 2 BELOW
완벽한 문법을 선호하는 분들에게 포맷간 기술적 차이를 설명할 필요가 없습니다.
Drew의 글에 따르면 XHTML 문법을 써야 할 중요한 이유가 있죠.
이런 이야기가 나올때 마다 초보자는 간단한 문법이 더 쉽다는 가정을 한다. 황당하다. 내 경험에 따르면 규칙이 명확한 문법이 배우고 쓰기 더 쉽다. 난 XHTML을 좋아한다. 속성이 항상
따옴표 안에 있어 읽기 편하고 태그는 항상 닫겨 있다. 이게 예측 가능하고 읽기 쉬워 더 간단하다.
- Drew McLellan


XHTML 문법이 HTML 보다 더 낫다는 것은 아닙니다.
단지 개인의 선택입니다. 서로 다른 문법을 선호한다고 해서 그게 틀린것은 아니죠.
전 태국 음식을 좋아하고 여러분이 이태리 음식을 좋아하는 차이 정도죠
XHTML 2의 죽음이 곧 XHTML 언어의 죽음이 아닙니다.
앞으로 쭈욱... 태그를 닫고 속성에 따옴표를 표시하실 수 있습니다.
XHTML 1을 계속 쓰거나 HTML 5를 쓰실 수 있으니까!
그렇습니다. HTML 5는 여러분이 편한 방식 양쪽 모두 제공합니다. 브루스 박사말을 들어볼까요?
난 XHTML 문법을 좋아한다. 소문자로 적고 따옴표를 꼭 치고 태그를 늘 닫는 간단한 방법이다. 깔끔한 모습이 시원한 맥주 한잔을 연상시킨다. 하지만 여러분은 대문자로 태그를 쓰는게
자연스럽고 태그를 닫는 것을 귀찮을지도 모른다. HTML5에서는 선택 할 수 있다. 새로운 전환점이 된 이 원리는 여러분이 무엇을 좋아하든 가능하다.
- Bruce Lawson
여러분의 문서를 application/xhtml+xml로 제공하면 XHTML5에 충실한 문서입니다
아! 죄송합니다. 또 용어에 혼란을...
XHTML 2는 XHTML을 위한 것이 아니고 오히려 HTML5에서 살아 있음을 기억해 주세요.

필자에 의해 번역 되었으며 Creative Commons License하에서 사용하도록 허가 받았다.

출처 :http://www.smashingmagazine.com/2009/07/29/misunderstanding-markup-xhtml-2-comic-strip/

1.3 HTML5 표준 문서들

1.3.1 HTML5인 것과 아닌 것

최근 업계에서 HTML5를 일종의 마케팅 용어처럼 쓰고 있지만, 이에 대한 명확한 구분을 해야 할 필요가 있다. 현재 HTML5와 연계 표준은 W3C의 HTML W/G과 WebApps W/G 그리고 WHATWG에서 각각 진행하고 있다.

HTML5는 초기에 WHATWG에서 작업한 것을 받아들이면서 다양한 기능들을 모두 포함하고 있었다. W3C에서부터 표준안 제정 과정에서 기능별로 모듈화 되어 여러 표준안으로 나누어지게 되었고, 일부 자바스크립트 API표준안들은 WebApps W/G으로 넘겨져 작업하고 있다.

그러나, W3C에 참여하지 못하는 일반 웹 개발자들을 대상으로 WHATWG의 스펙은 W3C의 분리된 스펙을 합쳐서 그대로 제공 하고 있을 뿐만 아니라 정치적 이유로 제공되기 어려운 샘플코드나 구현 권고 등을 포함하고 있으며 메일링리스트를 통해 의견을 받고 있기도 하다.

이 장에서는 W3C가 만들고 있는 웹 표준 문서들을 간략히 소개하고자 한다. W3C의 표준 문서는 대체로 읽기 어려운 것으로 알려져 있다. 따라서 웹 개발자가 표준을 쉽게 이해할 수 있도록 다양한 관점에서 각기 다른 표준 문서를 제공하고 있다. 2009년 4월 이전에는 아래 소개된 문서가 HTML 5 표준안에 함께 담겨 있었으나 분량이 많고 기존 마크업 기반 내용과 혼란을 준다는 측면에서 분리해서 관리하고 있다.

1.3.2 HTML5 인 것

HTML W/G에서는 HTML5 그 자체를 다루는 표준안을 만들고 있으며 렌더링 호환성 제공 방법, HTML5 마크업과 그 속성에 대한 용법 정리 그리고 콘텐츠의 의미적 표현 및 문서간 데이터 알림 등의 스펙을 다룬다.

HTML 5: A vocabulary and associated APIs for HTML and XHTML

HTML 5의 원래 표준안으로 800여 페이지 분량으로 주로 내용이 웹 브라우저 개발자를 위해서 만들어져 있다. 따라서, 가급적웹 개발자용 문서를 따로 보는 것이 좋다.


W3C Working Draft

W3C®

HTML5

A vocabulary and associated APIs for HTML and XHTML

W3C Working Draft 24 June 2010

This Version:
  http://www.w3.org/TR/2010/WD-html5-20100624/
Latest Published Version:
  http://www.w3.org/TR/html5/
Latest Editor's Draft:
  http://dev.w3.org/html5/spec/Overview.html
Previous Versions:
  http://www.w3.org/TR/2010/WD-html5-20100304/
  http://www.w3.org/TR/2009/WD-html5-20090825/
  http://www.w3.org/TR/2009/WD-html5-20090423/
  http://www.w3.org/TR/2009/WD-html5-20090212/
  http://www.w3.org/TR/2008/WD-html5-20080610/
  http://www.w3.org/TR/2008/WD-html5-20080122/
Editors:
  Ian Hickson, Google, Inc.

HTML5 표준안을 설명하는 문구에서 Vocabulary는 사전과 마찬가지로 마크업 및 속성에 대한 정의, 용법에 대해 상세한 설명을 포함하고 있으며 이를 지원하는 Associated API로 구성되어 있다는 것은 기억하면 좋겠다.

HTML5 differences from HTML4

이 문서는 기존 HTML에 익숙하던 사람들이 HTML5에서 무엇이 바뀌었는지 알 수 있도록 만든 소개 문서이다. 이 문서는 HTML 5 입문자들이 읽기에 적당하며 연도별 주요 변경 내용도 담고 있으며 본 가이드의 부록에 포함되어 있다.

HTML: The Markup Language

기존 HTML 5 스펙은 웹 브라우저 개발 회사를 위해 기술된 표준안이다. 개발자 관점에서 무엇이 어떻게 바뀌었고 어떻게 사용할 수 있는지 보여줄 수 있는 문서가 필요하다. 과거 W3C 표준안들의 문제점이 바로 이용자가 아닌 개발자 위주로 만들어져 있어 읽기 어려웠다는 것이다.

이 문서는 바로 이용자 즉, 웹 개발자를 위한 스펙이다. 이 문서는 HTML5 표준안(웹 개발자 관점)의 하부 문서로서 HTML 문서를 주로 저작하는 웹 퍼블리셔 혹은 HTML 코더를 위해 만들어진 문서이다.

HTML+RDFa 1.1

XHTML2 워킹 그룹이 해산되고 나서 가장 애매해진 것이 RDFa라는 표준이다. 시맨틱 웹에서 콘텐츠의 의미적 표현을 위해 사용하는 RDF를 HTML에서도 사용할 수 있도록 만든 것이 RDFa인데 HTML5에 포함시켜야 한다는 주장이 나온 이후 논쟁이 뜨거웠다.

전반적으로 HTML5에 RDFa를 추가하는 것에 반대하는 기류가 강했는데 그 이유로 1) RDFa는 여전히 어렵다 2) XML namespace가 필요한데 HTML5에는 없다 3) 그러한 이유로 복사해서 붙여 넣기를 해도 RDFa 사용 선언이 없으면 쓸 수 없다. 하지만, 시맨틱 웹과의 연결 고리를 위해 수용되었고 백악관 웹사이트나 크리에이티브커먼즈 등에서 채용하는 등 데이터 웹을 위한 방법으로 이용 가능하다.

HTML Microdata

RDFa가 여전히 어렵고 이에 반해 태그의 id나 class를 이용해 의미적표현을 하는 마이크로포맷(Microformat)은 제한적이므로 이를 대체해 줄 수 있는 Microdata가 제안되었다. 쓰임새는 마이크르포맷과 거의 유사하고 item과 itemprop라는 별도 속성을 통해 의미를 표현할 수 있다.

HTML Canvas 2D Context

Canvas 태그 내 각종 객체를 회전 및 변환하고 그레디언트, 이미지 생성 등 각종 효과를 주는 기능 부분을 기술하고 있다. 태그 내 각종 객체를 그리고 생성하는 데 필요한 API도 포함한다.

HTML5: Techniques for providing useful text alternatives

범용적으로 사용하는 alt를 통해 콘텐츠 특성을 작성하는 방법을 규정한 규격이다. 원래 4장에 기술되어 있는 alt 컨텐트 속성에 대해 좀 더 자세하게 예제를 포함하여 이해하기 쉽도록 작성되었다.

Polyglot Markup: HTML-Compatible XHTML Documents

HTML5문서를 XML로 구문 오류 없이 함께 같이 쓰는 방법을 알려준다. HTML과 XML 모두 유효한(valid) 복합(Polyglot) 문서를 만들수 있다.문서 형식 선언은 <! DOCTYPE html> 이며 대소문자를 구별한다.

table 요소는tbody 요소를 마크업하고, 요소와 속성은 일부 예외를 제외하고는 소문자로 표기하며, 빈 요소에는 종료 슬래시를 넣는다. (예: <br/>) 또한, 콘텐트 속성 값은 따옴표로 꼭 하고 엔터티 참조로서 &amp, &lt, &gt, &apos, &quot 전용를 넣는 등 XHTML의 구문을 그대로 이용하면 된다

1.3.3 HTML5가 아닌 것

HTML5 주요 스펙에 포함되어 있었거나 차후에 분리된 표준안으로서 웹 애플리케이션 개발을 지원하기 위해 만들어진 것으로 HTML5의 범주 안에 포함된다고 간주되는 것들이다.

Server-Sent Events

이 문서는 웹 서버로 부터 전달(Push)되는 데이터 예를 들어 SMS 같은 것을 받을 수 있도록 EventSource를 정의하고 이벤트를 기다릴 수 있도록 하는 API를 기술하고 있고 HTML5 표준안에서 분리 중이다.

Communications

이 문서은 기존 Ajax의 단점으로 알려진 크로스 도메인 문서 접근을 가능하게 해 주는 스펙이다. 마이크로소프트의 XHR 때문에 약간 논의가 지지부진한 면이 있지만 텍스트를 위한 서버 통신을 지원해 준다. 물론 보안 사항에 대한 부분도 중요하게 다루어지고 있다.

Web SQL Database

자바 스크립트를 이용해 웹 브라우저 내장 데이터베이스에 SQL을 통해 질의하는 API이다. 오프라인 웹 애플리케이션 개발이나 모바일에서 로컬 데이터 캐싱이 필요할 때 유용하게 사용할 수 있으며, 일반적인 DB 라이브러리 수준의 메소드를 지원해 준다.

Web Sockets API

한 웹 페이지에서 서로 다른 서버에 있는 웹 페이지에 양방향 통신을 할 수 있는 별도 프로토콜을 정의할 수 있는 API이다.

Web Workers

웹 애플리케이션이 주 문서와 병렬적으로 스크립트를 백그라운드로 수행할 수 있게 해 주는 API. 쓰레드 기반 메시지 처리를 가능하게 해 준다. CPU 부하를 많이 잡는 작업을 여러 워커(worker)로 나누어 작업하거나 클라이언트 DB를 업데이트 하거나 나누어서 작업이 가능한 자바 스크립트 API를 제공해 준다.

그 밖의 표준들

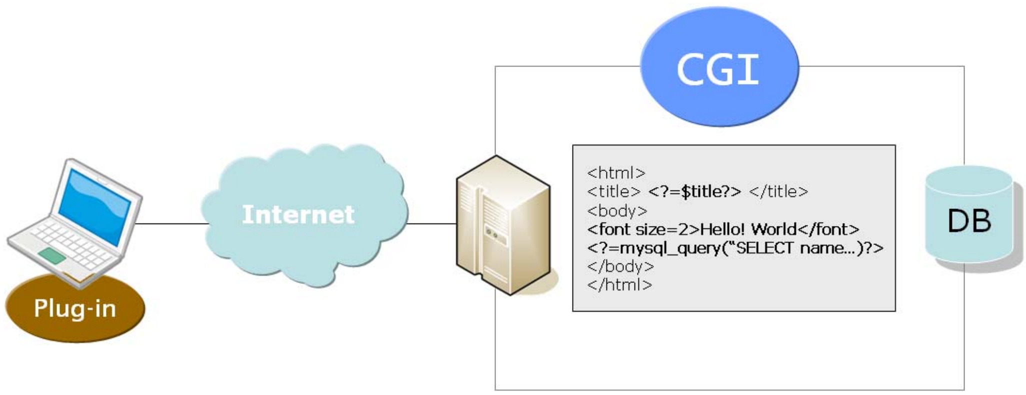
HTML5와 관련되 표준안에는 Content-Type Processing Model, The Web Concept, Geolocation API, SVG, MathML, XMLHttpRequest 등이 있으며, W3C와 별개로 WHATWG에서는 자막을 위한 WebSRT 기술 및 <device>요소를 통한 카메라나 USB제어 같은 기능도 준비하고 있다.

1.4 HTML5와 웹개발 방법론의 변화

이 장에서는 HTML5가 가져올 웹 개발 기술 및 방법론의 변화에 대해 알아보고자 한다. 지금까지의 웹 개발 플랫폼 변천 과정을 통해 현재의 변화를 확인해 보고자 한다.

웹 문서 시대(1990년대)

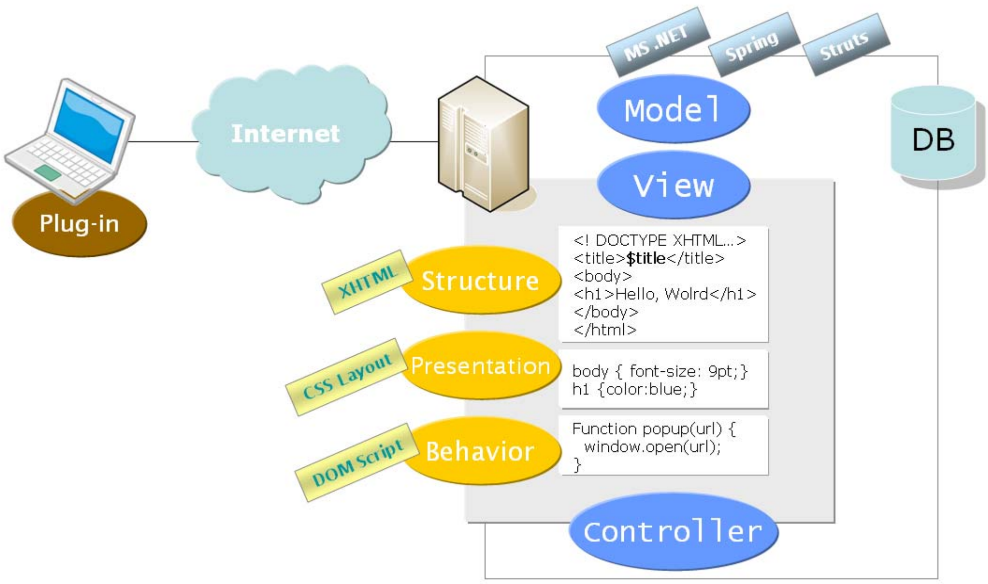
웹 서버와 웹 브라우저간 정적 HTML문서를 주로 보내거나 CGI(Common Gateway Interface)를 이용하여 개발하는 경우, 마크업과 프로그램 코드가 섞여있는 개발 방식을 사용했다. 이 때는 개발 직군간의 업무 분담이 전혀 이루어지지 않는 상태였다.



웹 표준 시대(2000년대 초반)

2000년대 초반으로 오면서 백엔드 개발에서는 이른바 MVC 모델이라는 기법을 통해 데이터 모델과 템플릿 그리고 비즈니스 로직이 분리된 코드를 통해 개발 생산성을 높이게 된다.

프론트엔드에서도 웹 표준 개발 방법론을 통해 구조(HTML), 표현(CSS), 동작(DOM Script)를 분리하고 CSS 레이아웃과 W3C 기반 DOM을 통한 웹 개발 방식을 많이 이용하게 된다.



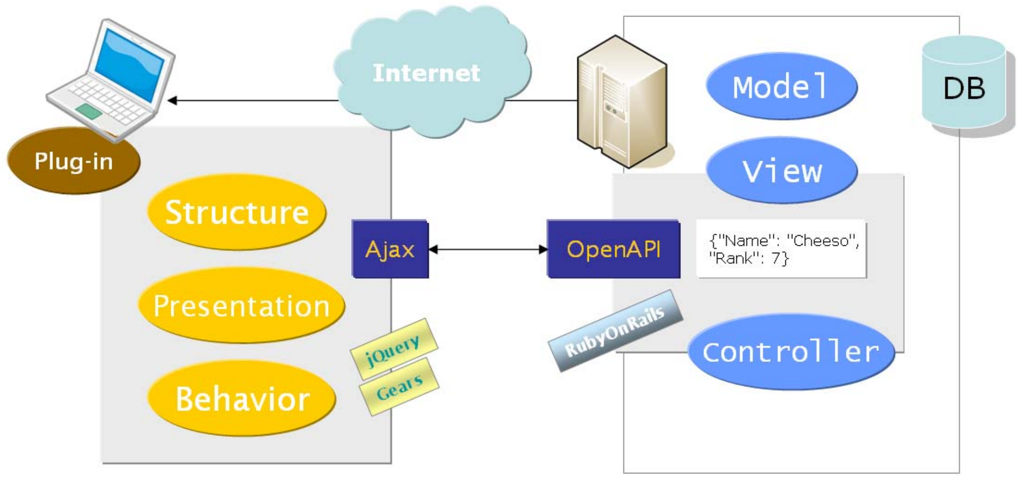
국내에서는 대략 2004년부터 웹 표준 기반 프론트엔드 개발 방식이 도입되기 시작했으며 주요 포털을 비롯하여 민간 기업 및 공공 웹 사이트에 웹 접근성과 더불어 확산되게 된다. 이를 통해 국내에서도 웹 퍼블리셔를 중심으로 한 직군이 형성되었다.

Ajax 시대 (2000년대 후반)

2004년 지메일과 구글맵스가 소개되면서 프론트 엔드 부분의 웹 기술의 혁신이 일어나게 된다. 즉, 프론트엔드 웹 콘텐츠가 고객의 PC에 일단 로딩이 된 후, 웹 서버에 Ajax 호출을 통해 데이터를 받아 온 후 기존 DOM을 갱신하는 개발 방법이다.

이 방법을 통하면 백엔드 개발자가 json과 같은 데이터 기반 응답만 하게 되므로 더 간단한 웹 개발이 이루어진다. 이에 반해 프론트엔드 개발자

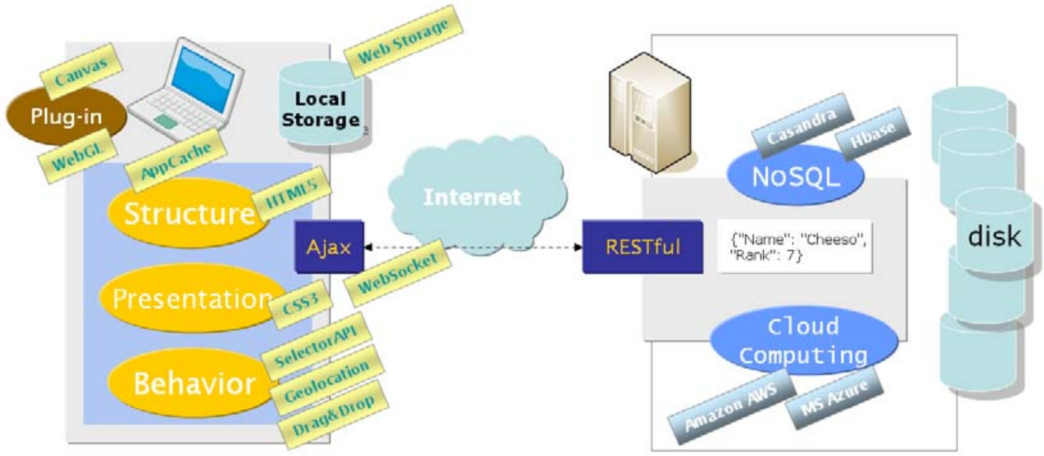
는 다양하고 풍부한 사용자 경험을 제공하는 웹 애플리케이션 개발이 가능해졌다.



하지만, 이 방식의 경우도 여전히 웹 서버에 종속적으며 독립적인 웹 애플리케이션 개발은 가능하지 않다.

HTML5시대 (2010년대 초반)

HTML5가 가져올 가장 큰 변화는 서버와 독립적인 웹 애플리케이션의 개발이 가능하다는 대목이다. 특히, 모바일 환경에서 오프라인 기능과 로컬 데이터베이스의 지원은 웹 서버와 독립할 수 있는 여건을 만들어 준다.



특히, HTML5의 Canvas, 드래그앤드롭, 지오로케이션, 파일API 등을 통한 사용자 경험을 확대해 줄 수 있다. 구글의 경우, 웹스토어라는 새로운 웹 애플리케이션 마켓플레이스를 준비하고 있기도 하다.

게다가 백엔드 부분의 기술적 변화도 눈에 띈다. 기존의 RDBMS와

MVC 기반 플랫폼에서 늘어나는 데이터를 잘 처리하기 위해 RESTful한 데이터 통신을 제공하고, 데이터는 좀 더 빠른 분산 파일 스토리지에 저장하는 방법이다. 구글, 아마존, 페이스북, 트위터가 이미 이러한 방식으로 웹 서비스를 제공하고 있기도 하다.

향후 5년간 다양한 웹 기술 요소들이 웹 개발 방법론 및 플랫폼을 변화시킬 것으로 예상된다. 간단한 프로토타입을 자사의 서비스에 조금씩 도입함으로서 새로운 변화가 연착륙할 수 있도록 하는 노력이 필요하다.

1.5 FAQ

HTML5를 현재 당장 사용할 수 있는가?

구글 및 애플닷컴 사이트의 첫 소스코드는 HTML5 Doctype을 이용하고 있다. 즉, 지금 당장 여러분의 웹 사이트에 HTML5 마크업을 이용할 수 있다는 이야기다. 물론 IE와 같이 오래된 브라우저의 경우 호환 스크립트로 기능 추가(Fallback)을 하는 방법을 사용하면 된다. HTML5 Shive, Modnizer, IE-CSS3 같은 스크립트를 통해 하위 호환성을 보장해 줄 수 있다. 단, HTML5를 통한 웹 사이트 제작이 여러분의 회사의 개발 생산성을 높히고 고객의 요구와 비지니스에 부합한다는 전제하에서 말이다. 미래 웹 기술을 미리 선도적으로 도입함으로서 부가 이익도 누릴 수 있다. 2004~2005년도에 웹 표준 기반 CSS 레이아웃 도입에 많은 토론이 있었지만 현재 결국 많이 도입되어 있다.

HTML5를 지향할 때 국내 IE 사용자가 현저히 줄어들 것인가?

IE 사용자도 고객이다. IE6 고객도 충분히 지원해야 한다. 다행히 윈도7의 판매 증가 및 마케팅으로 인해 IE8 혹은 앞으로 나올 IE9으로의 전환은 잘 이루어질 것 같다. 또한, 국내 모바일 단말기 시장에서 아이폰 및 안드로이드 고객의 증가로 인해 비 IE 브라우저 점유율도 높아질 것이다. 혁신은 빨리 온다. 미리 준비하는 사람에게 기회가 올 것이다.

HTML5가 현 시대에 비해 너무 과포장되어 않나?

과포장은 스티브잡스와 어도비의 논쟁이서 일정 부분 그렇게 된바가 있다. 하지만 웹 표준을 100% 준수하는 웹 브라우저도 없을 것이고 그런 시대가 오지도 않는다. 다만 상위 호환성(Forward Compatiblity)를 염두해

두고 특정 웹 브라우저 고객에게 좀 더 나은 서비스를 제공할 수 있다면, 고객은 바보가 아니기 때문에 더 많은 사람들이 향상된 기능을 사용할 것이다.

HTML5의 단점에는 어떤 것들이 있나?

단점은 역시 새로운 웹 기술이기 때문에 학습 비용(Learning Cost)가 든다는 것이다. 투자의 관점에서 접근해야 한다. 또한, 프론트 웹 개발 방식에 많은 자바스크립트 API가 제공되기 때문에 마크업 개발자들의 경우도 어느 정도는 자바스크립트 언어를 다룰 줄 알아야 한다는 점이다. 무론 백엔드 개발자들이 이전해 올 수도 있지만 프론트 엔드 개발자들이 어느 정도 자신의 역할을 넓혀야 한다고 본다.

웹 브라우저 벤더들 희망 사항 외에 실제 사용자 요구가 어떻게 반영되는가?

W3C안에서 표준안이 만들어지고 있기 때문에 일반 웹 개발자들이 의사결정에 직접 참여하기는 어렵다. 하지만 각 워킹 그룹이 운영하는 공개 메일링리스트인 public-html@w3.org나 public-whatwg@whatwg.org 등에 가입하여 충분히 의견 교환을 할 수 있다.

HTML5가 이슈이다 보니 고객들이 제작 요청을 하고 있다. 어떻게 접근해야 할까?

HTML5를 통한 콘텐츠 웹 서비스와 웹 애플리케이션 서비스를 분리하여 접근할 필요가 있다. 각 부분에 맞는 HTML5 기술셋을 정하여 고객이나 기획자에게 제시하고 이를 설득하는 것이 좋겠다. 예를 들어, 모바일 광고에서 HTML5를 적용하는 경우 동영상(video) 요소 및 캔버스(Canvas)를 이용하는 방법이 있다. 또한, 아이폰 앱 대용으로 제공하는 웹앱의 경우 오프라인 캐싱과 웹 스토리지를 이용하는 방법 등이다. 만약 콘텐츠 웹 서비스를 HTML5로 제공하려면 CSS3와 함께 접근하는 것도 좋은 접근이다.

HTML5에서 애니메이션이 된다면 플래시 개발자와 업무 분담은?

HTML5의 멀티미디어 기능은 초기의 플래시 프로토타입과 유사하다. 따라서 이 기능을 이용할 수 있는 분야는 플래시가 오용되는 부분 즉, 메뉴 네비게이션, 광고, 메인 페이지 애니메이션, 이벤트 페이지 애니메이션 같은 것들이다. 이들은 플래시 개발자들이 가진 영역이기도 하나 실질적으로

창조적은 콘텐츠로 분류되기는 어렵다. HTML5 멀티미디어 기능은 플래시 개발자와 UI 개발자의 업무 분담을 더욱 확실히 하여 각자 고유 영역에 더욱 매진할 수 있게 해 줄 수 있을 것이다.

참고 사이트

- W3C HTML Working Group http://www.w3.org/html/wg/
- HTML5 specification http://www.w3.org/TR/2010/WD-html5-20100624/
- HTML5 differences from HTML4 http://www.w3.org/TR/2010/WD-html5-diff-20100624/
- HTML: The Markup Language http://www.w3.org/TR/2010/WD-html-markup-20100624/
- HTML+RDFa 1.1 http://www.w3.org/TR/2010/WD-rdfa-in-html-20100624/
- HTML Microdata http://www.w3.org/TR/2010/WD-microdata-20100624/
- HTML Canvas 2D Context http://www.w3.org/TR/2010/WD-2dcontext-20100624/
- HTML5: Techniques for providing useful text alternatives http://www.w3.org/TR/2010/WD-html-alt-techniques-20100624/
- Polyglot Markup: HTML-Compatible XHTML Documents http://www.w3.org/TR/2010/WD-html-polyglot-20100624/

- HTML5Rocks http://html5rocks.com 구글이 만든 사이트
- HTML5Test http://html5test.com 브라우저별 지원 현황 파악 가능
- HTML5Doctor http://html5doctor.com HTML5 마크업 관련 블로그 및 Q&A
- HTML5gallery http://html5gallery.com HTML5 기반 웹 사이트 모음
- Mozilla Hacks http://hacks.mozilla.or.kr Mozilla 기반 웹 기술 블로그

2. HTML5 마크업

현재 HTML 는 아직 초안 상태이다. 기존에 표현의 용도로 사용되었던 요소들에 의미를 부여하는 작업이나, alt나 summary와 같은 접근성 기능들은 아직 논의 중에 있다.

HTML5는 브라우저 개발사를 위한 문서와 웹 저작자를 위한 문서가 나눠져 있다. 또한 기존 표준에서 명확하지 않았던 부분들을 새롭게 정의하면 현재 브라우저들의 동작 방식을 표준화하는데 노력했다. 결과적으로 새롭게 변경되거나 추가된 부분들도 하위 호환성을 가지고있기 때문에 조금만 신경을 쓰면 과거의 브라우저에서도 HTML5를 사용할 수 있다.

HTML5는 웹 디자이너와 웹 개발자로 하여금 마크업 언어를 쓰기 쉽게 만드는 목적을 가지고 있다.

2.1. 구조와 문법

HTML5는 HTML4와 XHTML1과 거의 완벽하게 호환할 수 있다. HTML 구문을 따르는 문서는 언제나 text/html 형식으로 배포할 수 있다. 또한 오류 복원 기능도 현재 가장 많이 사용되고 있는 방식을 참고로 상세하게 포함할 것이다.

HTML 형식의 문서

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Example document</title>
  </head>
  <body>
    <p>Example paragraph</p>
  </body>
</html>
```

iframe과 같은 요소가 외부 콘텐츠를 포함할 때 text/html-sanboxed 형식을 선언할 수 있게 할 것이다. XML 형식으로의 구성도 가능하다. XML로 구성된 문서는 반드시 application/xhtml+xml나 application/xml로 배포되어야 한다.

XML 형식의 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Example document</title>
  </head>
  <body>
    <p>Example paragraph</p>
  </body>
</html>
```

2.1.1 문자 인코딩

문서에 사용된 문자 인코딩의 인식은 다음의 우선 순위에 따라서 결정된다. 1) HTTP 헤더에서 선언 2) 유니코드 BOM에서 선언 3) meta 태그 사용

HTML5에서는 새로 추가된 간소화된 구문을 포함한 아래 두가지의

meta 태그를 모두 사용할 수 있다.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta charset="UTF-8">
```

2.1.2 DOCTYPE

기존의 HTML DOCTYPE은 SGML 기반이었기 때문에 DTD를 명시할 필요가 있었다. 하지만 HTML5에서 DOCTYPE은 브라우저가 표준 모드로 작동되게 하는 역할만 하면 되기 때문에 아주 간소해 졌다. 이미 브라우저들은 HTML5 DOCTYPE을 표준 모드로 작동되게 하고 있다.

HTML 4.01 Transitional DTD

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

XHTML 1.0 Strict DTD

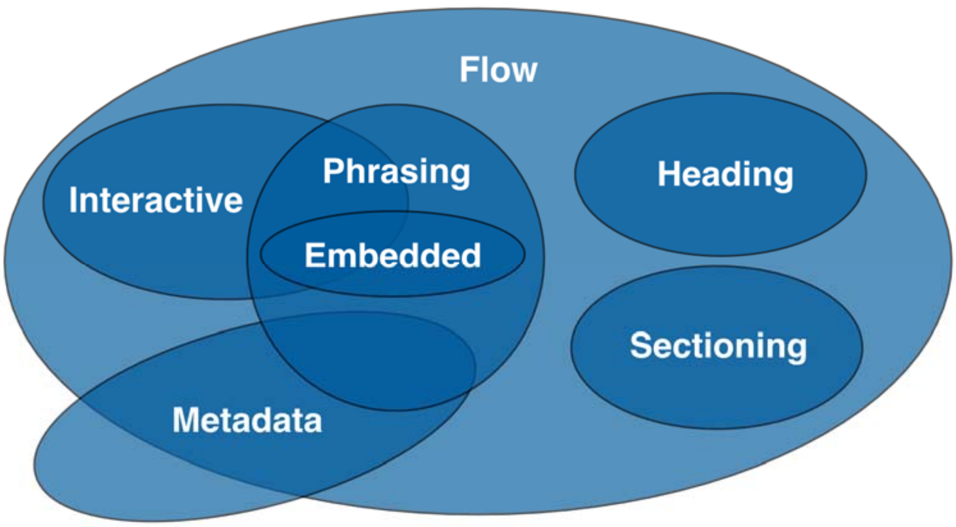
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

HTML5 DTD

```
<!DOCTYPE html>
```

2.1.3 콘텐츠 모델

HTML5의 각 요소들은 특성에 따라 분류된다. 한요소가 어떤 분류에도 속하지 않을 수도 있고 여러 분류에 속해있을 수도 있다. 분류는 아래 이미지와 같이 분포하고 있다.



분류	특성	예
메타데이타 콘텐츠 (metadata content)	콘텐츠의 모양, 동작을 설정하거나 다른 문서와의 관계를 나타낸다.	base, command, link, meta, noscript, script, style, title
플로우 콘텐츠 (flow content)	대부분의 body 요소 안의 요소들이 포함된다. 플로우 콘텐츠는 하위에 텍스트나 임베디드 콘텐츠를 포함한다.	a, abbr, address, map>area, article, aside, audio, b, bdo, blockquote, br, button, canvas, cite, code, command, datalist, del, details, dfn, div, dl, em, embed, fieldset, figure, footer, form, h1 ~ h6, header, hgroup, hr, i, iframe, img, input, ins, kbd, keygen, label, map, mark, math, menu, meter, nav, noscript, object, ol, output, p, pre, progress, q, ruby, samp, script, section, select, small, span, strong, style[scoped], sub, sup, svg, table, textarea, time, ul, var, video, wbr
섹셔닝 콘텐츠 (sectioning content)	헤딩과 푸터의 범위를 결정하는 요소이다. 모든 섹셔닝 콘텐츠는 헤딩과 아웃라인을 가지고 있다.	article, aside, nav, section
헤딩 콘텐츠 (heading content)	섹션의 헤더를 의미한다.	h1, h2, h3, h4, h5, h6, hgroup
프레이징 콘텐츠 (phrasing content)	문서의 텍스트를 의미한다. 프레이징 콘텐츠는 하위에 텍스트나 임베디드 콘텐츠를 포함한다.	a, abbr, map>area, audio, b, bdo, br, button, canvas, cite, code, command, datalist, del, dfn, em, embed, i, iframe, img, input, ins, kbd, keygen, label, map, mark, math, meter, noscript, object, output, progress, q, ruby, samp, script, select, small, span, strong, sub, sup, svg, textarea, time, var, video, wbr
임베디드 콘텐츠 (embedded content)	이미지, 비디오, 플래시 등 외부 콘텐츠를 문서내에 표현한다.	audio, canvas, embed, iframe, img, math, object, svg, video
인터액티브 콘텐츠 (interactive content)	사용자와 상호작용하는 요소들이다.	a, audio[controls], button, details, embed, iframe, img[usemap], input, keygen, label, menu, object[usemap], select, textarea, video[controls]
트랜스패런트 콘텐츠 (transparent content)	부모 요소의 콘텐츠에 따라 포함하는 콘텐츠의 분류가 바뀌는 요소를 말한다.	

2.1.4 MathML와 SVG

HTML5에서는 MathML이나 SVG를 문서 안에서 사용할 수 있다.

```
<!doctype html>
```

```
<title>SVG in text/html</title>
<p>
 A green circle:
 <svg> <circle r="50" cx="50" cy="50" fill="green"/> </svg>
</p>
```

2.1.5 문법 검사

W3C에서는 이미 HTML5 문법에 대한 검사 기능을 제공하고 있다. HTML4나 XHTML1의 문법 검사를 위해서 사용했던 W3C Markup Validation Service(http://validator.w3.org/)를 사용하면 HTML5의 문법 검사도 수행할 수 있다.

2.2. 요소와 속성

2.2.1 구조를 나타내는 요소

HTML4에서는 문서의 구조를 만들때 div, span 요소를 id, class와 함께 사용하였다. 문서의 구조를 나타내는 요소가 풍부하지 않아서 복잡한 페이지의 경우 많은 중첩된 div요소를 사용하였다. HTML5에서는 문서의 구조를 만드는 새로운 개념과 요소들이 추가되어서 보다 구조적인 문서를 만드는 것이 가능해졌다.

새로 추가된 요소들은 현재 파이어폭스3, 사파리3.1, 크롬2, 오페라 9.6 등 최신의 브라우저에서 문제없이 사용할 수 있다. 단 인터넷 익스플로러는 이러한 요소를 제대로 표현해 주지 못하기 때문에 createElement 메서드를 이용해서 활성화 해주는 과정이 필요하다.

예를 들어서 인터넷 익스플로러에서 header 요소를 사용하기 위해서는 아래와 같은 스크립트를 head 요소에서 먼저 실행해 주어야 한다.

```
document.createElement('header');
```

이렇게 HTML5에서 새로 추가된 요소들을 인터넷 익스플로러에서 사용할 수 있도록 미리 제작되어진 스크립트를 사용하는 것도 가능하다.

- HTML5 Enabling Script: http://code.google.com/p/html5shim/
- IE Print Protector: http://code.google.com/p/ie-print-protector/

section

문서나 어플리케이션의 섹션을 나타내는 section 요소가 추가되었다. 이

요소는 문서의 구조를 나타내기 위해 h1, h2, h3, h4, h5, h6 요소들과 같이 사용될 수 있다.

책의 1장, 2장이나 탭형식으로 되어 있는 콘텐츠의 각 탭에 section 요소를 쓸 수 있다. 만약 스타일링이나 스크립트를 위해서 감싸는 요소가 필요하다면 이른 용도로는 section을 사용해서는 안된다. 이러한 용도로는 문서 구조상으로 특별한 의미가 없는 div 요소를 사용해야 한다.

```
<article>
   <hgroup>
      <h1>Apples</h1>
      <h2>Tasty, delicious fruit!</h2>
   </hgroup>
   <p>The apple is the pomaceous fruit of the apple tree.</p>
   <section>
      <h1>Red Delicious</h1>
      <p>These bright red apples are the most common found in many
         supermarkets.</p>
   </section>
   <section>
      <h1>Granny Smith</h1>
      <p>These juicy, green apples make a great filling for apple
pies.</p>
   </section>
</article>
```

하나의 섹션 안에서는 그 섹션이 전체 문서 구조에서 같는 단계와 상관없이 h1 요소를 사용할 수 있다.

nav

네비게이션을 위해 구성된 섹션을 나타낸다.

페이지 않에 있는 모든 링크의 그룹이 nav 요소로 기술될 필요는 없다. nav 요소로는 그 페이지의 주요 네비게이션 링크들만 묶어주는 것이 좋다. 일반적으로 푸터에서 그 사이트의 주요 페이지로 이동하는 링크를 제공하는 경우가 많은데 이러한 링크는 푸터 안에 표시하는 것 만으로도 그 목적을 가늠할 수 있기 때문에 nav 요소를 사용할 필요는 없다.

그렇다고 한 페이지 안에서 nav 요소를 하나만 사용해야 하는 것은 아니다. 사이트 전반적인 이동을 위한 네비게이션과 페이지 전체를 이동하는 네베게이션이 있는 경우 각각을 nav 요소로 기술 할 수 있다.

article

문서내의 독립적인 글을 article로 표시할 수 있다. 블로그 글이나 뉴스 본문 등이 이에 해당한다.

article 요소 안에 article 요소가 들어갈 경우 안쪽의 article 요소는 밖의 article 요소의 내용과 관련이 있는 내용이라는 것을 의미한다. 예를 들어서 블로그의 글과 사용자가 작성한 댓글이 있는 경우 하나의 댓글은 안쪽 article로 기술하고 전체 블로그 글은 바깥쪽 article로 기술 할 수 있다.

aside

문서의 주 내용이 아닌 관련성이 낮은 내용들은 aside로 표시할 수 있다. 본문과 직접적으로 상관이 없는 관련 사이트 링크나 광고, nav 요소의 그룹 등이 aside 요소로 기술 될 수 있다.

hgroup

섹션의 제목을 나타낸다.

섹션의 제목이 여러 단계를 가지고 있을 때 이를 hgroup 요소로 기술할 수 있다. 문서의 구조를 나타낼 때에는 hgroup 안에 있는 가장 높은 레벨의 헤딩을 사용하고 나머지 요소들은 문서의 구조에서는 타나나지 않는다.

header

소개나 네비게이션 기능들의 묶음을 나타낸다.

header 요소는 보통 섹션의 제목(h1, h2, h3, h4, h5, h6)을 포함하지만 반드시 포함할 필요는 없다. 또한 header 요소는 목차나 검색창, 로고 등을 포함할 수도 있다.

header 요소는 섹션으로 간주되지 않는다. header 요소를 썼다고 문서 구조상 새로운 섹션이 생기지 않는다.

footer

섹션의 푸터를 나타내고 저자나 저작권 등을 포함할 수 있다.

footer 요소는 가장 가까운 선행하는 섹션의 푸터를 의미한다. 섹션의 저자나 관련 문서로의 링크, 저작권 정보 등을 나타낼 수 있다.

저자나 편집자의 연락처는 보통 footer 요소 안의 address 요소로 표현이 된다. 대부분의 푸터 정보는 섹션의 마지막에 위치하지만 반드시 마지막에 위치할 필요는 없다.

footer 요소도 header 요소와 마찬가지로 섹션으로 간주되지 않는다. footer 요소를 썼다고 문서 구조상 새로운 섹션이 생기지 않는다.

figure

그래픽이나 비디오를 위한 캡션을 표시할 수 있게 한다. figcaption 요소로 캡션을 표시한다.

본문에서 참조가 되지만 내용의 흐름에 크게 영향을 미치지 않는 일러스트, 다이어그램, 사진, 코드 등을 표시할 수 있다.

```
<figure>
   <video src="ogg"></video>
   <figcaption>Example</figcaption>
</figure>
```

2.2.2 헤딩과 섹션

h1, h2, h3, h4, h5, h6, hgroup 요소들은 헤딩을 의미한다.

섹셔닝 콘텐츠의 첫번째 헤딩 콘텐츠는 그 섹션의 헤딩을 의미한다. 그 이후의 같거나 더 높은 레벨의 헤딩은 새로운 섹션을 의미한다. 낮은 레벨의 섹션은 새로운 하위 섹션을 의미한다.

blockquote, body, details, fieldset, figure, td는 섹셔닝 루트(sectioning root) 요소들로, 이 요소들은 자기 자신만의 아웃라인을 가지고 상위 요소 문서의 아웃라인에 영향을 미치지 않는다.

```
<body>
   <h1>Foo</h1>
   <h2>Bar</h2>
   <blockquote>
      <h3>Bla</h3>
   </blockquote>
   <p>Baz</p>
   <h2>Quux</h2>
   <section>
```

```
        <h3>Thud</h3>
    </section>
    <p>Grunt</p>
</body>
```

위와 같은 문서의 구조는 다음과 같다.

- 1. Foo (Grunt 문단을 가지고 있는 body 섹션의 헤딩)
- 1. Bar (Baz 문단과 blockquote 요소 포함한 섹션의 헤딩)
- 2. Quux (내용이 없고 헤딩만 있는 섹션의 헤딩)
- 3. Thud (section 요소로 표시된 섹션의 헤딩)

섹션 헤딩의 단계는 크게 상관없지만 되도록 h1 요소를 사용하거나 섹션 구조에 맞는 단계의 헤딩을 사용할 것을 권장한다.

```
<body>
    <h4>Apples</h4>
    <p>Apples are fruit.</p>
    <section>
        <h2>Taste</h2>
        <p>They taste lovely.</p>
        <h6>Sweet</h6>
        <p>Red apples are sweeter than green ones.</p>
        <h1>Color</h1>
    <p>Apples come in various colors.</p>
    </section>
</body>
```

즉, 이렇게 헤딩의 단계가 순차적이지 않아도 유효한 구조적인 문서로 판단할 수 있다. 하지만 아래와같이 섹션을 명시적으로 지정해 주는 것이 더 좋다.

```
<body>
    <h1>Apples</h1>
    <p>Apples are fruit.</p>
    <section>
        <h2>Taste</h2>
        <p>They taste lovely.</p>
        <section>
            <h3>Sweet</h3>
            <p>Red apples are sweeter than green ones.</p>
        </section>
    </section>
    <section>
        <h2>Color</h2>
        <p>Apples come in various colors.</p>
    </section>
</body>
```

섹션을 구성하는 요소의 아웃라인은 포함된 섹션의 목록으로 이루어진다. 각각의 섹션은 하나의 헤딩을 포함할수 있고 다수의 다른 섹션을 포함할

수 있다.

```
<body>
    <h1>A</h1>
    <p>B</p>
    <h2>C</h2>
    <p>D</p>
    <h2>E</h2>
    <p>F</p>
</body>
```

이렇게 구성된 마크업은 다음과 같은 아웃라인을 생성한다.

body 요소로부터 하나의 섹션이 만들어지고 이 섹션은 헤딩 A와 문단 B를 갖는다. 이 섹션은 다은 두개의 섹션을 포함한다. h2 요소로부터 하나의 섹션이 만들어지고 헤딩 C와 문단 D를 갖는다. 그 다음 h2 요소로부타 하나의 섹션이 또 만들어지고 헤딩 E와 문단 F를 갖는다.

2.2.3 그 밖의 요소

video, audio

비디오나 오디오 콘텐츠를 넣기위해 사용된다. 스크립트로 제어할 수 있게끔 스크립트 API가 제공된다. source 요소로 여러개의 미디어를 추가할 수있다.

embed

플러그인 콘텐츠를 넣을 때 사용된다.

mark

중요 텍스트를 표기할 때 사용된다. 예를 들어 원 저자는 중요하다고 표현하지 않았지만 이 글을 따왔을 때 중요한 부분이 변경되거나 추가된다면 이 텍스트를 mark 요소로 표현할 수 있다.

progress

다운로드 상태바 처럼 작업의 진행상황을 나타낼 때 사용된다.

meter

분량이나 수량을 나타낼 때 사용된다.

time

날짜나 시간을 나타낼 때 사용된다.

ruby, rt, rp

일본어에서 한문에 음을 다는 것과 같은 루비 주석(ruby annotation)을 달 때 사용된다.

canvas

그래픽이나 게임과 같이 동적인 비트맵 이미지를 구현할 때 사용된다.

command

사용자가 행할 수 있는 명령 기능을 나타낼 때 사용된다.

details

사용자의 필요에 의해서 사용할 수 있게 제공되는 추가적인 정보나 기능을 나타낼 때 사용된다. summary 요소를 이용해서 요약, 제목, 캡션을 나타낼 수 있다.

```
<section class="progress window">
    <h1>"Really Achieving Your Childhood Dreams" 복사중...</h1>
    <details>
        <summary>복사중...
            <progress max="375505392" value="97543282"></progress> 25%
        </summary>
        <dl>
            <dt>초당 전송량:</dt> <dd>452KB/s</dd>
            <dt>복사할 파일명:</dt> <dd>/home/rpausch/raycd.m4v</dd>
            <dt>대상 파일명:</dt> <dd>/var/www/lectures/raycd.m4v</dd>
            <dt>걸린시간:</dt> <dd>01:16:27</dd>
            <dt>컬러 프로파일:</dt> <dd>SD (6-1-6)</dd>
            <dt>영상 크기:</dt> <dd>320x240</dd>
        </dl>
    </details>
</section>
```

datail 요소를 이용해서 상세한 내용을 사용자의 선택에 의해서 보이거나 감출 수 있다.

datalist

미리 정의되어 있는 option의 묶음을 나타낸다. input의 list, datalist의

id로 서로 연결된다.

keygen

서버에 공개키를 보내고 로컬의 키 저장소에 개인키를 저장하는데 사용된다.

output

서식이나 스크립트를 통해 생성된 결과물을 나타낼 때 사용된다.

input

input 요소의 type 속성으로 다음과 같은 다양한 형식을 사용할 수 있게 되었다. tel, search, url, email, datatime, date, month, week, time, datetime-local, number, range, color. 이러한 속성을 사용해서 달력이나 컬러 픽커와 같은 기능을 브라우저에서 제공할 수 있다.

type 속성 값에 따른 다양한 콘트롤 형식

값	상태	데이터 타입	콘트롤 타입
hidden	감춰짐	임의의 문자열	n/a
text	텍스트	줄바꿈 없는 텍스트	텍스트 필드
search	검색	줄바꿈 없는 텍스트	검색 필드
tel	전화번호	줄바꿈 없는 텍스트	텍스트 필드
url	URL	절대 IRI	텍스트 필드
email	이메일	이메일 주소나 이메일 주소 리스트	텍스트 필드
password	비밀번호	줄바꿈 없는 텍스트	데이터 입력이 나타나지 않는 텍스트 필드
datetime	날짜와 시각	UTC 날짜와 시각	날짜와 시각 콘트롤
date	날짜	시간대 없는 날짜	날짜 콘트롤
month	달	시간대 없는 년과 달	달 콘트롤
week	주	시간대 없는 주 번호	주 콘트롤
time	시각	시간대 없는 시각	시각 콘트롤

값	상태	데이터 타입	콘트롤 타입
datetime-local	로컬 날짜와 시각	시간대 없는 날짜와 시각	날짜와 시각 콘트롤
number	숫자	숫자 값	텍스트나 스피너 콘트롤
range	범위	숫자 값이나 정확한 숫자가 필요없는 의미상의 값	슬라이더 콘트롤
color	색	8-bit 적녹청 sRGB 칼라	컬러 웰
checkbox	체크박스	이미 설정된 리스트 값의 0또는 다른 값	체크박스
radio	라디오 버튼	지정된 값	라디오 버튼
file	파일 입력창	MIME type과 파일명이 있는 파일 목록	레이블과 버튼
submit	전송 버튼	폼 서식을 전송하는 지정된 값	버튼
image	이미지 버튼	폼 서식을 전송하는 이미지 좌표 값	클릭 가능한 이미지나 버튼
reset	리셋 버튼	해당사항 없음	버튼
button	버튼	해당사항 없음	버튼

style, script

style 요소와 script 요소의 type 속성이 생략 가능해졌다. style 요소의 기본 type 값은 "text/css", script 요소의 기본 type 값은 "text/javascript"이기 때문에 둘 다 type 요소를 생략할 수 있게 되었다.

```
<style>
#myelement {
   width: 100px;
}
</style>
<script>
function myfunction() {
   return 'Hello world';
}
</script>
```

2.2.4 새로운 속성

HTML4에서 사용되던 속성을 보다 넓게 사용할 수 있도록 확장하였다.

media

a와 area 요소에는 link 요소와 마찬가지로 media 속성을 지정할 수 있다.

ping

a와 area 요소에는 링크를 따라 갔을 때 참조될 ping 속성을 지정할 수 있다.

ping 속성을 지정하면 사용자 클릭 정보 수집과 같은 곳에 사용할 수 있고 사용자는 이를 옵션에서 제어할 수 있게 된다.

hreflang, rel

area 요소에는 link와 a 요소와 마찬가지로 hreflang과 rel 속성을 지정할 수 있다.

target

base 요소에는 a 요소와 마찬가지로 target 속성을 지정할 수 있다. HTML4에서 strict DTD를 사용할 때에는 target을 쓸 수 없었지만 iframe과 같이 웹에서 이미 유용하게 사용하고 브라우저에서 지원도 많이 되고 있기 때문에 HTML5에서는 사용할 수 있다.

value, start: li요소의 value 속성과 ol 요소의 start 속성은 구조와 관련된 속성이기 때문에 더이상 폐지된 속성이 아니다.

charset

meta 요소의 charset 속성이 추가되었다. 이미 많이 사용되고 잘 지원되고 있는 속성이다.

autofocus

input 요소의 type 속성이 hidden일 때를 제외하고 input, select textarea, button 요소에 autofocus 속성이 추가되었다.

이 속성은 문서가 로드되었을 때 폼에 포커스가 되게 한다. 사용자는 이러한 동작을 옵션에서 끌 수 있어야 한다.

placeholder

input 요소와 textarea 요소에 placeholder 속성이 추가되었다.

placehoder 속성의 값은 input이나 textarea 요소에 값이 입력되기 전에 힌트 정보로 표시된다. 포커스를 받기 전까지 placeholder 값이 표시되고 포커스를 받거나 값이 입력되면 표시된 값이 사라진다.

긴 힌트를 위해서는 placeholder 대신에 title 속성을 사용해야 하며, placeholder는 label을 대시할 수는 없기 때문에 label은 별도로 지정이 되어있어야 한다.

```
<input type="email" name="address" placeholder="abc@def.com">
```

form

input, output, select, textarea, button, fieldset 요소에 form 속성을 지정하여 form 요소의 밖에 콘트롤을 위치할 수 있게 되었다.

required

input 요소의 type 속성이 hidden, image 이거나 button 요소의 속성이 submit인 경우를 제외하고 input과 textarea 요소에 required 속성이 추가되었다.

이 속성은 사용자가 반드시 값을 입력해야 한다는 것을 의미한다. HTML5는 폼 값 유효성 검사(validation)와 같이 자주 사용되는 폼 관련 기능을 표준화 하여 더욱 강력하고 풍부한 웹 폼을 만들 수 있게 하고 있다.

```
<h1>회원 등록</h1>
<form action="/newaccount" method=post>
    <p>
        <label for="username">이메일 주소:</label>
        <input id="username" type=email required name=un>
    <p>
        <label for="password1">비밀번호:</label>
        <input id="password1" type=password required name=up>
```

```
    <p>
       <label for="password2">비밀번호 확인:</label>
       <input id="password2" type=password
onforminput="setCustomValidity(value != password1.value ? '비밀전호가
일치하지 않습니다.' : '')">
    <p>
     <input type=submit value="계정 생성">
</form>
```

이메일 주소와 비밀번호를 필수요소로 지정하였고, 비밀번화 확인 필드에서는 입력된 값이 비밀번호와 일치하는지를 추가적으로 확인한다.

disable

fieldset 요소에 disable 속성을 지정하여 전체 필드셋을 비활성화 할 수 있다.

autocomplete 등

input 요소에 autocomplete, min, max, multiple, pattern, step 속성이 추가되었다. 또한 datalist와 함께 사용될때 list 속성을 지정할 수 있다.

formaction 등

input과 button 요소에 formaction, formenctype, formmethod, formnovalidate, formtarget 속성을 지정할 수 있다. 이러한 속성들은 form 요소의 action, enctype, method, novalidate, target 속성을 재정의하게 된다.

type, label

menu 요소에 type과 label 속성을 지정할 수 있다. 이 속성들을 이용해서 브라우저에서 지원하는 컨텍스트 메뉴와 같은 기능을 제작할 수 있다.

scoped

style 요소에 scoped 속성을 지정하여 문서의 특정 부분에만 스타일이 적용되도록 할 수 있다.

async

style 요소에 async 속성을 지정하여 스크립트의 로딩과 실행 타이밍을

조절 할 수 있다.

manifest

html 요소에 manifest 속성을 지정하여 오프라인 애플리케이션 API에서 캐시 정보를 사용할 수 있다.

sizes

link 요소에 sizes 속성을 지정하여 아이콘의 기본 크기 정보를 표시할 수 있다.

reversed

ol 요소에 reversed 속성을 지정하여 번호 순서를 바꿀 수 있다.

sandbox 등

iframe 요소에 sandbox, seamless, srcdoc 속성을 지정하여 샌드박스로 지정할 수 있다.

2.2.4 글로벌 속성

HTML4의 글로벌 속성(class, dir, id, lang, style, tabindex, title)을 모든 요소에 지정할 수 있게 되었다.

contenteditable

사용자가 내용을 수정할 수 있는 요소를 표시할 수 있다.

contextmenu

컨텍스트 메뉴를 지정할 수 있다.

data-*

저작자가 새로운 속성을 지정할 수 있다.

draggable

새로운 드래그 앤 드롭 API를 사용할 수 있다.

hidden

특정 요소가 존재하지 않는 것임을 표시할 수 있다. 마크업으로 속성을 주는 것과 CSS의 display로 화면에 보이지 않게 하는 것은 엄연히 다르다. hidden 속성이 지정되면 그 요소는 문서상에서 없는 것과 같은 효과를 가진다.

role, aria-*

보조기기에서 접근성을 향상시키는데 사용할 수 있게 한다.

spellcheck

맞춤법 기능을 적용되거나 적용되지 않게 할 수 있다. 이 속성이 지정되어 있으면 사용자가 맞춤법 기능을 제어할 수 있게 된다.

onclick, onfocus 등의 event

HTML4의 on{이벤트이름} 속성을 사용할 수 있고 video나 audio 요소의 play와 같은 추가적인 이벤트 속성을 사용할 수 있다.

2.2.5 변경된 요소

a

href 속성이 없는 a 요소를 사용하여 링크 자리(placeholder link)를 표시할 수 있다.

a 요소는 안에 버튼과 같은 상호작용하는 요소가 없다면 전체 문단, 리스트, 표, 섹션 등을 모두 포함할 수 있다.

```
<aside class="advertising">
    <h1>광고</h1>
    <a href="http://ad.example.com/?adid=1929&amp;pubid=1422">
        <section>
            <h1>HTML5 오픈 컨퍼런스!</h1>
            <p>차세대 웹 서비스를 위한 새로운 웹 표준!</p>
            <p>최초 HTML5 오픈 컨퍼런스가 열립니다.</p>
        </section>
    </a>
    <a href="http://ad.example.com/?adid=375&amp;pubid=1422">
        <section>
            <h1>HTML5 가이드 북!</h1>
            <p>가이드 북을 함께 나눠드립니다.</p>
```

```
        <p>핵심만 잘 뽑은 핸드북!</p>
    </section>
  </a>
</aside>
```

전체 섹션을 하나의 a 요소로 감싸고 배너로 활용할 수 있다.

address

address 요소가 문서 구조(sectioning) 상에서 특정 범위에 적용된다.

b

b 요소는 문서상에서 중요한 의미는 없지만 문체적으로 다르게 나타내어져야 하는 텍스트를 위해서 사용된다. 예를 들어서 상품 설명 안에서의 상품의 이름이나 문서에 특정 키워드를 나타낼 때 사용할 수 있다.

hr

hr 요소는 문단 수준의 나눔을 의미하게 되었다.

i

i 요소는 어조나 분위기 또는 다른 일반 텍스트와 구분을 해야 하는 텍스트를 표시하는데 사용된다. 예를 들어서 특정 구분이나 기술적인 용어, 다른 언어 표현, 생각, 배의 이름 등과 같은 것을 표현할 때 사용된다. 사용 언어에 따라서 많은 차이가 있을 수 있다.

label

label 요소를 클릭 했을 때 포커스를 이동하는 기능이 플랫폼 수준에서의 표준이 아닌 이상 더이상 포커스를 콘트롤로 이동시켜서는 안된다.

menu

menu 요소가 툴바나 컨텍스트 메뉴를 위해서 개선되었다.

small

small 요소는 추가적인 코멘트나 법적인 표현 등과 같이 작게 출력되어

야 하는 내용을 나타낸다.

strong

strong 요소는 강한강조가 아니라 중요함을 나타낸다.

2.2.6 변경된 속성

기존의 속성들 중에 변경된 것들이 있다. 되도록이면 이러한 속성들은 다른 대안을 활용해 기술하는 것이 더 바람직 하다.

border

img 요소의 border 속성은 "0" 값을 지정해야 한다. CSS를 사용하는 것이 더 권장된다.

language

script 요소의 language 속성은 "Javascript" 값(대소문자 구분 없음)을 지정해야 해고 type 속서의 값과 일관되어야 한다. 특별한 목적이 있지 않은 한 language 속성은 생략할 수 있다.

name

a 요소의 name 속성 보다는 id 속성을 사용하는 것이 권장된다.

summary

table 요소의 summary 속성은 HTML5의 다른 대안을 이용해서 표시하는 것이 더 권장된다.

2.2.7 빠진 요소

basefont, big, center, font, s, strike, tt, u

이 요소들은 표현에 관련된 요소이기 때문에 HTML5에 정의되지 않았다.

frame, frameset, noframes

이 요소들은 콘텐츠를 네비게이션할 때 사용성과 접근성에 영향을 미치

기 때문에 HTML5에 정의되지 않았다.

acronym

이 요소는 많이 사용되지 않고 있고 혼동되는 경우가 많기 때문에 정의되지 않았다. 대신 abbr 요소를 사용할 수 있다.

applet

이 요소는 object 요소로 대체되었다.

isindex

이 오래된 요소는 다른 폼 콘트롤 요소로 대체되었다.

dir

이 요소는 ul 요소로 대체되었다.

2.2.8 빠진 속성

HTML5에서 없어진 속성

속성	적용 요소
rev, charset	link, a
shape, coords	a
longdesc	img, iframe
target	link
nohref	area
profile	head
version	html
name(id 사용 권장)	img
scheme	meta
archive, classid, codebase, codetype, declare, standby	object
valuetype, type	param
axis, abbr	td, th

속성	적용 요소
scope	td

표현에 관련된 없어진 속성

속성	적용 요소
align	caption, iframe, img, input, object, legend, table, hr, div, h1, h2, h3, h4, h5, h6, p, col, colgroup, tbody, td, tfoot, th, thead, tr
alink, link, text, vlink	body
background	body
bgcolor	table, tr, td, th, body
border	table, object
cellpadding, cellspacing	table
char, charoff	col, colgroup, tbody, td, tfoot, th, thead, tr
clear	br
compact	dl, menu, ol, ul
frame	table
frameborder	iframe
height	td, th
hspace, vspace	img, object
marginheight, marginwidth	iframe
noshade	hr
nowrap	td, th
rules	table
scrolling	iframe
size	hr
type	li, ol, ul
valign	col, colgroup, tbody, td, tfoot, th, thead, tr
width	hr, table, td, th, col, colgroup, pre

표현에 관련된 속성들은 CSS를 활용하여 대체 하는 것이 권장된다.

2.3 HTML5 예제

2.4.1 블로그 샘플

블로그 사이트의 첫페이지 예제 코드이다. 가장 최신에 작성된 글의 목록과 사이트 헤어, 푸터가 있는 간단한 페이지이다.

```
<body>
<header>
    <h1>HTML5 의 세계로!</h1>
    <nav>
        <ul>
            <li><a href="news.html">새소식</a></li>
            <li><a href="blog.html">블로그</a></li>
            <li><a href="example.html">예제</a></li>
        </ul>
    </nav>
</header>
<div>
    <article>
        <header>
            <h1>HTML5 오픈 컨퍼런스가 열립니다.</h1>
        </header>
        <p>2010 년 7 월 2 일 ...</p>
        <!-- 블로그 글 내용 -->
        <footer>
            <p>
                <time pubdate datetime="2010-06-20T14:36-
08:00">어제</time> 작성 되었음
            </p>
        </footer>
    </article>
    <article>    <!-- 생략 -->
    </article>
    <article>    <!-- 생략 -->
    </article>
</div>
<footer>
    <p>
        <a href="about.html">이 블로그에 관하여</a>
        | <a href="policy.html">개인정보 보호정책</a>
        | <a href="contact.html">연락처</a>
    </p>
</footer>
</body>
```

블로그 글의 모음을 div 요소로 묶어주었다. 이러한 경우에 새로 추가된 section 요소를 사용하는 경우가 많은데, section 요소는 하나의 주제에 대한 섹션을 표시하는 것이기 때문에 이 경우와 같이 하나의 주제가 아닌 여러 주제의 글을 단순히 묶은 경우에는 적절하지 않다. 이렇게 단순히 그루핑 만을 하기 위한 목적으로는 div 요소를 사용하는 것이 더욱 적절하다.

푸터의 링크 모음에 nav 요소를 사용하지 않고 그냥 p 요소만을 사용하였다. footer 요소로 페이지의 마무리 부분이라는 것을 명시 했기 때문에 굳이 nav 요소를 사용하지 않아도 의미 전달이 가능하기 때문에 이렇게 표현하는 것이 가능하다.

2.3.2 블로그 글 본문

블로그 사이트의 본문페이지의 분문 영역 예제 코드이다. 레이아웃이 동일하다면 이전 섹션의 메인 페이지 코드에 있는 중앙 div 요소 대신에 아래의 article 요소가 들어가게 될 것이다.

```
<article>
   <header>
      <h1>HTML5 오픈 컨퍼런스가 열립니다.</h1>
   </header>
   <p>2010 년 7 월 2 일 ...</p>
   <!-- 블로그 글 내용 -->
   <section>
      <h1>댓글</h1>
      <article>
         <footer><!-- footer 요소가 섹션의 처음에 위치할 수도 있다. -->
            <p>김민석님의 댓글</p>
            <p><time pubdate datetime="2010-06-20T19:10-
08:00"></time></p>
         </footer>
         <p>...댓글 내용...</p>
      </article>
      <article>
         <footer>
            <p>이명호님의 댓글</p>
            <p><time pubdate datetime="2009-06-20T19:15-
08:00"></time></p>
         </footer>
         <p>...댓글 내용...</p>
      </article>
      <!-- 생략 -->
   </section>
   <footer>
      <p>
         <time pubdate datetime="2010-06-20T14:36-08:00">어제</time>
         작성 되었음
      </p>
   </footer>
</article>
```

블로그의 글 본문은 독립적인 글 섹션이기 때문에 article 요소를 사용하였다. 전체 섹션의 제목인 h1 요소를 header 요소로 표시하였고 글 본문과 댓글 섹션과 글의 작성 일시를 표시한 footer 요소를 사용하였다.

하나의 댓글 또한 독립적인 글 섹션이기 때문에 article 요소를 사용하였

다. 이와같이 article 안에 또 다른 article 요소나 section 요소를 포함하는 것이 가능하고 반대로 section 요소도 또 다른 section 요소나 article 요소를 포함할 수 있다.

댓글의 article 요소를 보면 헤딩도 없고 footer 요소가 가장 처음에 나온 것을 볼 수 있는데, 이와같이 적절하다고 판단될 경우 footer 요소를 가장 먼저 표시할 수도 있다. 물론 header 요소를 사용할 수도 있지만 이는 전적으로 저작자의 표현 의도에 달려있다.

2.3.3 블로그 사이드 바

블로그 사이트의 본문 옆 부분에 본문과 직접적인 관련은 없지만 링크나 관련 정보를 표현하는 경우에 대한 예제 코드이다.

```
<aside>
   <nav>
      <h1>최근글</h1>
      <ol reversed>
         <li><a href="conference.html">HTML5 오픈 컨퍼런스가 
열립니다.</a></li>
         <li><a href="accessibility.html">HTML5 의 접근성</a></li>
         <li><a href="browser.html">HTML5 브라우저 
지원현황입니다..</a></li>
      </ol>
   </nav>
   <nav>
      <h1>최근 댓글</h1>
      <ol reversed>
         <li><a href="article/accessibility/cmt3384">표에 summary 
속성도 바뀐 것 같더라고요.</a></li>
         <li><a href="article/accessibility/cmt2452">longdesc 속성이 
없어진 이유가 뭘까요?</a></li>
         <li><a href="article/browser/cmt1562">브라우저 지원 현황을 보니 
저도 조금씩 시도를 해봐야 겠네요.</a></li>
      </ol>
   </nav>
   <nav>
      <h1>이웃 블로그</h1>
      <ul>
         <li><a href="http://channy.creation.net/">Channy's 
Blog</a></li>
         <li><a href="http://hyeonseok.com/">Hyeonseok.com</a></li>
         <li><a href="http://naradesign.net/">NARADESIGN</a></li>
         <li><a href="http://firejune.com/">Firejune Blog</a></li>
         <li><a href="http://xguru.net/">Guru's Blog</a></li>
         <li><a href="http://jhyun.wordpress.com/">삐돌이의 웹 
접근성</a></li>
      </ul>
   </nav>
```

```
</aside>
<aside>
    <section>
        <h1>미투데이</h1>
        <blockquote
cite="http://me2day.net/op****/2010/06/20#04:00:22">
            와 정말 오래 기다렸어요!
        </blockquote>
        <blockquote cite="
http://me2day.net/html****/2010/06/20#03:58:22">
            기대됩니다!
        </blockquote>
        <blockquote cite="
http://me2day.net/fx****/2010/06/20#03:57:22">
            빨리 신청해야 겠네요.
        </blockquote>
    </section>
    <section>
        <h1>트위터</h1>
        <blockquote
cite="http://twitter.com/sf****/status/17037990****">
            RT @ie**** 규모가 꽤 큰 HTML5 행사가 열리네요.
        </blockquote>
        <blockquote
cite="http://twitter.com/ie****/status/17037990****">
            규모가 꽤 큰 HTML5 행사가 열리네요.
        </blockquote>
        <blockquote
cite="http://twitter.com/op****/status/1703799****">
            이런 일정이 겹치네요.
        </blockquote>
    </section>
</aside>
```

블로그 글 본문과 직접적인 관련이 없기 때문에 aside 요소를 이용하여 콘텐츠를 표현하였다. 글이나 댓글 링크의 모음을 nav 요소로 표현하였고, SNS 서비스 사이트의 글들은 blockquote를 사용하여 다른 곳에서 가져온 글이라는 것을 표현하였다.

aside 요소도 섹션을 구성하는 요소이기 때문에 하위에 단일한 주제를 표현할 필요가 있다. 다시 말해서 aside를 사용할 때에는 단순히 레이아웃에서의 사이드바 영역을 aside로 표현하는 것이 아니라 주 콘텐츠와 직접적으로 연관이 없는 내용을 aside로 표현하는 것이다. 따라서 HTML5 요소를 사용할 때에는 콘텐츠의 내용을 먼저 파악해야 한다. 사이드바 영역의 콘텐츠들이 서로다른 주제를 가지고 있는 내용이어서 섹션이 나눠져야 할 필요가 있을 때는 위와 같이 두개의 aside 요소를 사용할 수도 있다. 이 aside 요소들을 div 요소로 묶어서 사이드바 영역으로 그루핑을 해주게 되면 레이아웃일 제작할 때에 활용할 수 있다.

구조를 구성하는 HTML5 요소들은 위치나 영역을 기준으로 사용하는 것이 아니라 표현하고자 하는 콘텐츠의 내용을 고려하여 적절한 요소를 선택하고 사용해야 한다.

2.4 HTML 4.01과의 비교

요소	HTML 4.01/XHTML 1.0	HTML 5	설명
a	strict	yes	하이퍼링크
abbr	strict	yes	약어
acronym	strict	-	두문자어
address	strict	yes	연락 정보
applet	transitional	-	자바 애플릿
area	strict	yes	이미지 맵의 영역
article	-	yes	독립적인 섹션
aside	-	yes	보조 섹션
audio	-	yes	오디오
b	strict	yes	굵은 텍스트
base	strict	yes	문서의 기본 URI
basefont	transitional	-	기본 폰트 스타일
bdo	strict	yes	텍스트 방향 설정
big	strict	-	큰 글자
blockquote	strict	yes	긴 인용
body	strict	yes	문서 본문
br	strict	yes	줄바꿈
button	strict	yes	버튼 콘트롤
canvas	-	yes	비트맵 캔바스
caption	strict	yes	표 캡션
center	transitional	-	가운데 정렬
cite	strict	yes	인용

요소	HTML 4.01/XHTML 1.0	HTML 5	설명
code	strict	yes	코드
col	strict	yes	표 컬럼
colgroup	strict	yes	표 컬럼 그룹
command	-	yes	사용자 커맨드
datalist	-	yes	기설정 콘트롤 값
dd	strict	yes	정의 설명
del	strict	yes	삭제
details	-	yes	상세정보
dfn	strict	yes	용어 정의
dir	transitional	-	디렉토리
div	strict	yes	블록 그루핑
dl	strict	yes	정의 목록
dt	strict	yes	정의 용어
em	strict	yes	강조
embed	-	yes	외부 콘텐츠
fieldset	strict	yes	서식 그룹
figcaption	-	yes	figure 요소의 캡션
figure	-	yes	캡션있는 그림
font	transitional	-	폰트 서식
footer	-	yes	섹션 푸터
form	strict	yes	서식
frame	frameset	-	프레임
frameset	frameset	-	프레임 셋
h1	strict	yes	제목1
h2	strict	yes	제목2
h3	strict	yes	제목3
h4	strict	yes	제목4

요소	HTML 4.01/XHTML 1.0	HTML 5	설명
h5	strict	yes	제목5
h6	strict	yes	제목6
head	strict	yes	문서 헤드
header	-	yes	섹션 헤더
hr	strict	yes	문단 구분
html	strict	yes	문서 루트
i	strict	yes	이탤릭 텍스트
iframe	transitional	yes	인라인 프레임
img	strict	yes	이미지
input	strict	yes	입력 콘트롤
ins	strict	yes	삽입
isindex	transitional	-	사용자 폼 생성
kbd	strict	yes	사용자 입력
keygen	-	yes	암호화 키 콘트롤
label	strict	yes	서식 레이블
legend	strict	yes	설명적인 캡션
li	strict	yes	목록 항목
link	strict	yes	자원 링크
map	strict	yes	이미지맵
mark	-	yes	표시된 텍스트
menu	transitional	yes	메뉴
meta	strict	yes	숨은정보
meter	-	yes	수량 포현
nav	-	yes	네비게이션
nobr	-	-	줄바꿈 금지
noembed	-	-	플러그인 대체 콘텐츠
noframes	frameset	-	프레임 대체 콘텐츠

요소	HTML 4.01/XHTML 1.0	HTML 5	설명
noscript	strict	yes	스크립트 대체 콘텐츠
object	strict	yes	외부 콘텐츠 삽입
ol	strict	yes	순서있는 목록
optgroup	strict	yes	선택상자 옵션 그룹
option	strict	yes	선택상자 옵션
output	-	yes	출력내용
p	strict	yes	문단
param	strict	yes	파라메터
pre	strict	yes	형식 유지 텍스트
progress	-	yes	처리상태
q	strict	yes	인라인 인용
rp	-	yes	루비 괄호
rt	-	yes	루비 텍스트
ruby	-	yes	루비 표현식
s	transitional	-	취소선
samp	strict	yes	프로그램 예제 출력
script	strict	yes	스크립트
section	-	yes	섹션
select	strict	yes	선택상자
small	strict	yes	작은 활자
source	-	yes	미디어의 소스 정보
span	strict	yes	인라인 그루핑
strike	transitional	-	취소선
strong	strict	yes	중요함
style	strict	yes	스타일 시트
sub	strict	yes	아래첨자
sup	strict	yes	위첨자

요소	HTML 4.01/XHTML 1.0	HTML 5	설명
summary	-	yes	detail 요소의 캡션
table	strict	yes	표
tbody	strict	yes	표 본문
td	strict	yes	표 셀
textarea	strict	yes	여러줄 입력 콘트롤
tfoot	strict	yes	표 푸터
th	strict	yes	표 헤더 셀
thead	strict	yes	표 헤더
time	-	yes	날짜나 시각
title	strict	yes	문서 제목
tr	strict	yes	테이블 줄
u	transitional	-	밑줄
ul	strict	yes	순서없는 목록
var	strict	yes	변수
video	-	yes	동영상
wbr	-	yes	줄바꿈
xmp	-	-	형식 유지 텍스트

3. CSS3 소개

CSS3는 HTML5와 더불어 최신 웹 브라우저들이 함께 채용하고 있는 스타일 표준이다. CSS3는 HTML5와 크게 관계 있는 부분은 없지만 현대적 웹 기술에 있어서 꼭 알아야할 웹 기술 요소이다. CSS3는 현재 표준 작업중이고 선택자에 대한 부분은 거의 완료되었다.

이 장에서는 CSS3를 이용하여 다양한 스타일 기능을 제공할 수 있는 방법을 중심으로 향후 HTML5 마크업을 작성 하거나 애플리케이션을 개발할 때 유용한 것들만 모아서 제공하고자 한다.

3.1 CSS2와 차이점

CSS2와 CSS3의 가장 큰 차이점은 CSS3가 모듈 기반으로 개발되고 있다는 점이다. 이것은 각종 브라우저나 디바이스가 필요에 따라 원하는 CSS 모듈만을 탑재하거나 또는 필요한 모듈만을 빠르게 자주 업데이트 하는 것을 돕는다. CSS3는 text, fonts, color, backgrounds & borders, transforms, transitions, animaitons와 같은 종류의 모듈들을 추가로 개발하고 있다. CSS3는 기존의 CSS2가 갖지 못했던 화려하고 역동적인 면모를 추가하여 포토샵과 자바스크립트 및 서버측 기술에만 완전히 의존하던 영역들을 개척했다. 상자의 크기에 따른 말줄임 표시, 투명한 배경, 그림자 효과, 둥근 모서리, 그라디언트, 도형의 회전과 비틀기, 애니메이션 효과 등이 가능해진 것이다. 특히 그래픽 디자인에만 의존하던 영역이 CSS3만으로도 상당부분 가능해지면서 웹 사이트의 성능 향상에 크게 기여할 수 있게 되었다.

3.2 CSS3 브라우저 지원현황

CSS3는 아직 개발이 완료되지 않은 초안이기 때문에 현재 시점(2010년 7월)을 기준으로 12 종류의 속성(CSS3의 모든 속성을 다루지 않았다) 및 CSS level 1~3 선택자 지원 현황을 파악했다.

3.2.1 CSS3 주요 속성들

아래 항목은 앞으로 소개할 CSS3의 유용한 속성에 대한 웹 브라우저 지원 현황을 정리한 도표이다.

Module	Properties	CR 5	SF 4	FF 3.6	OP 10.5	IE 8	IE 7	IE 6	Fallback for IE 6, 7, 8
Text	text-shadow	yes	Yes	yes	yes	Yes filter	Yes filter	Yes filter	DropShadow Filter
	text-overflow	yes	Yes	no	Yes -o-	yes	yes	yes	
	word-wrap	yes	Yes	yes	yes	yes	yes	yes	
Fonts	@font-face	yes	Yes	yes	yes	yes	yes	yes	IE를 위한 'eot' 포멧 필요
color	opacity	yes	Yes	yes	yes	yes filter	Yes filter	Yes filter	Alpha Filter

Module	Properties	CR 5	SF 4	FF 3.6	OP 10.5	IE 8	IE 7	IE 6	Fallback for IE 6, 7, 8
backgrounds & borders	box-shadow	Yes -webkit-	Yes -webkit-	Yes -moz-	yes	Yes filter	Yes filter	Yes filter	Shadow Filter
	border-radius	yes -webkit-	Yes -webkit-	Yes -moz-	yes	no	no	no	
	background(s)	yes	Yes	yes	yes	no	no	no	
	gradient	Yes -webkit-	Yes -webkit-	Yes -moz-	no	Yes filter	Yes filter	Yes filter	Gradient Filter
transforms	transform	Yes -webkit-	Yes -webkit-	Yes -moz-	Yes -o-	Yes filter	Yes filter	Yes filter	Matrix Filter
transitions	transition	Yes -webkit-	Yes -webkit-	No -moz- (3.7 or later)	Yes -o-	no	no	no	
animations	animation	Yes -webkit-	Yes -webkit-	no	no	no	no	no	

한편 fetchak.com은 ie-css3.htc(11.8KB) 파일을 사용하여 IE 6~8 브라우저에서 text-shadow, box-shadow, border-radius 렌더링을 흉내내는 스크립트를 제안하고 있다. 이 스크립트를 사용하면 IE 브라우저에서도 border-radius 표현이 가능해지고 text-shadow, box-shadow 표현은 더욱 풍부해 진다. 이 스크립트를 사용하면 DropShadow, Shadow filter를 사용할 필요가 없다. htc 포멧은 IE 브라우저에서만 기능하는 벤더 확장 자바스크립트 파일이다.

3.2.2 CSS 선택자들

아래 항목은 각종 CSS 버전에서 사용하고 있는 42가지의 유용한 선택자에 대한 웹 브라우저 지원 현황을 정리한 도표이다.

42 CSS3 Selectors **CR**:Chrome, **SF**:Safari, **FF**:Firefox, **OP**:Opera, **IE**:Internet Explorer

CSS level	Selectors	CR 5	SF 5	FF 3.6	OP 10.5	IE 8	IE 7	IE 6	Meaning
CSS3	E[attr^=val]	Yes	yes	yes	yes	yes	yes	no	'attr' 속성의 값이 'val'으로 시작하는 요소를 선택(공백으로 분리된 값이 일치해야 한다)

42 CSS3 Selectors **CR**:Chrome, **SF**:Safari, **FF**:Firefox, **OP**:Opera, **IE**:Internet Explorer

CSS level	Selectors	CR 5	SF 5	FF 3.6	OP 10.5	IE 8	IE 7	IE 6	Meaning
	E[attr$=val]	yes	yes	yes	yes	yes	yes	no	'attr' 속성의 값이 'val'으로 끝나는 요소를 선택(공백으로 분리된 값이 일치해야 한다)
	E[attr*=val]	yes	yes	yes	yes	yes	yes	no	'attr' 속성의 값에 'val'이 포함되는 요소를 선택(공백으로 분리된 값과 정확하게 일치하지 않아도 선택)
	E:root	yes	yes	yes	yes	no	no	no	문서의 최상위 요소(html)를 선택
	E:nth-child(n)	yes	yes	yes	yes	no	no	no	앞으로부터 지정된 순서와 일치하는 요소가 E 라면 선택(E 아닌 요소의 순서가 계산에 포함됨)
	E:nth-last-child(n)	yes	yes	yes	yes	no	no	no	뒤로부터 지정된 순서와 일치하는 요소가 E 라면 선택(E 아닌 요소의 순서가 계산에 포함됨)
	E:nth-of-type(n)	yes	yes	yes	yes	no	no	no	E 요소 중 앞으로부터 순서가 일치하는 E 요소를 선택(E 요소의 순서만 계산에 포함됨)
	E:nth-last-of-type(n)	yes	yes	yes	yes	no	no	no	E 요소 중 끝으로부터 순서가 일치하는 E 요소를 선택(E 요소의 순서만 계산에 포함됨)
	E:last-child	yes	yes	yes	yes	no	no	no	마지막에 등장하는 요소가 E 라면 선택(E 아닌 요소의 순서가 계산에 포함됨)
	E:first-of-type	yes	yes	yes	yes	no	no	no	E 요소 중 첫 번째 E를 선택(E 요소의 순서만 계산에 포함됨)
	E:last-of-type	yes	yes	yes	yes	no	no	no	E 요소 중 마지막 E를 선택(E 요소의 순서만 계산에 포함됨)
	E:only-child	yes	yes	yes	yes	no	no	no	E 요소가 유일한 자식이면 선택(E 아닌 요소가 하나라도 포함되면 선택 안함)
	E:only-of-type	yes	yes	yes	yes	no	no	no	E 요소가 유일한 타입이면 선택(E 아닌 요소가 포함되어도 E 타입이 유일하면 선택)

42 CSS3 Selectors **CR**:Chrome, **SF**:Safari, **FF**:Firefox, **OP**:Opera, **IE**:Internet Explorer

CSS level	Selectors	CR 5	SF 5	FF 3.6	OP 10.5	IE 8	IE 7	IE 6	Meaning
	E:empty	yes	yes	yes	yes	no	no	no	텍스트 및 공백을 포함하여 자식 요소가 없는 E를 선택
	E:target	yes	yes	yes	yes	no	no	no	E의 URI가 요청되면 선택(따라서 E는 ID가 지정되어 있어야 한다)
	E:enabled	yes	yes	yes	yes	no	no	no	사용 가능한 폼 콘트롤(input, textarea, select, button) E를 선택
	E:disabled	yes	yes	yes	yes	no	no	no	사용 불가능한 폼 콘트롤(input, textarea, select, button) E를 선택
	E:checked	yes	yes	yes	yes	no	no	no	선택된 폼 콘트롤(input checked="checked")을 선택
	E:not(s)	yes	yes	yes	yes	no	no	no	S가 아닌 E 요소를 선택
	E~F	yes	yes	yes	yes	yes	yes	no	E 요소가 앞에 존재하면 F를 선택(E가 F보다 먼저 등장하지 않으면 선택 안 함)
CSS2	*****	yes	yes	yes	yes	yes	yes	yes	모든 요소를 선택
	E[attr]	yes	yes	yes	yes	yes	yes	no	'attr' 속성이 포함된 요소 E를 선택
	E[attr=val]	yes	yes	yes	yes	yes	yes	no	'attr' 속성의 값이 정확하게 'val'과 일치하는 요소 E를 선택
	E[attr~=val]	yes	yes	yes	yes	yes	yes	no	'attr' 속성의 값에 'val'이 포함되는 요소를 선택(공백으로 분리된 값이 일치해야 한다)
	E[attr|=val]	yes	yes	yes	yes	yes	yes	no	'attr' 속성의 값이 'val' 또는 'val-' 으로 시작되는 요소 E를 선택
	E:first-child	yes	yes	yes	yes	yes	yes	no	첫 번째 등장하는 요소가 E 라면 선택(E 아닌 요소의 순서가 계산에 포함됨)
	E:lang(en)	yes	yes	yes	yes	yes	no	no	HTML lang 속성의 값이 'en'으로 지정된 요소를 선택

42 CSS3 Selectors **CR**:Chrome, **SF**:Safari, **FF**:Firefox, **OP**:Opera, **IE**:Internet Explorer

CSS level	Selectors	CR 5	SF 5	FF 3.6	OP 10.5	IE 8	IE 7	IE 6	Meaning
	E:before	yes	yes	yes	yes	yes	no	no	E 요소의 시작 지점에 생성된 요소를 선택
	E:after	yes	yes	yes	yes	yes	no	no	E 요소의 끝 지점에 생성된 요소를 선택
	E>F	yes	yes	yes	yes	yes	yes	no	E 요소의 자식인 F 요소를 선택
	E+F	yes	yes	yes	yes	yes	yes	no	E 요소를 뒤따르는 F 요소를 선택(E와 F 사이에 다른 요소가 존재하면 선택 안함)
CSS 1	**E**	yes	yes	yes	yes	yes	yes	yes	E 요소를 선택
	E:link	yes	yes	yes	yes	yes	yes	yes	방문하지 않은 앵커 E를 선택
	E:visited	yes	yes	yes	yes	yes	yes	yes	방문한 앵커 E를 선택
	E:hover	yes	yes	yes	yes	yes	yes	yes	E 요소에 마우스가 올라가 있는 동안 E를 선택
	E:active	yes	yes	yes	yes	yes	yes	yes	E 요소에 마우스 클릭 또는 키보드 엔터가 눌린 동안 E를 선택
	E:focus	yes	yes	yes	yes	yes	yes	yes	E 요소에 포커스가 머물러 있는 동안 E를 선택
	E:first-line	yes	yes	yes	yes	yes	yes	no	E 요소의 첫 번째 라인을 선택
	E:first-letter	yes	yes	yes	yes	yes	yes	no	E 요소의 첫 번째 문자를 선택
	.class	yes	yes	yes	yes	yes	yes	yes	클래스 이름이 class로 지정된 요소 선택
	#id	yes	yes	yes	yes	yes	yes	yes	아이디 이름이 id로 지정된 요소 선택

42 CSS3 Selectors **CR**:Chrome, **SF**:Safari, **FF**:Firefox, **OP**:Opera, **IE**:Internet Explorer									
CSS level	**Selectors**	**CR 5**	**SF 5**	**FF 3.6**	**OP 10.5**	**IE 8**	**IE 7**	**IE 6**	**Meaning**
	E F	yes	yes	yes	yes	yes	yes	yes	E 요소의 자손인 F 요소를 선택

3.3 CSS3 실전 적용

3.3.1 text-shadow


```
text-shadow:5px 5px 0 #ccc;
filter:progid:DXImageTransform.Microsoft.dropshadow(OffX=5, OffY=5,
Color=#cccccc, Positive=true);
display:inline-block; zoom:1; text-shadow:x offset y offset
blur_radius color
```

지원 브라우저 :

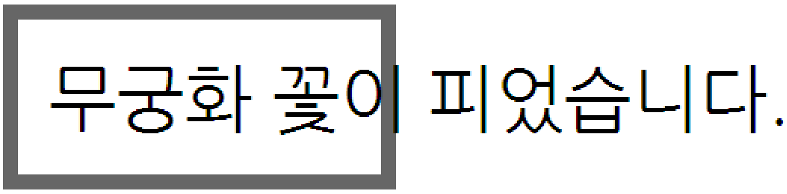
IE 브라우저를 제외한 모든 브라우저가 이미 text-shadow 속성을 지원하고 있다. blur 값은 생략할 수 있는데 생략하는 경우 기본값은 blur 스타일이 전혀 지정되지 않은 '0'이다. IE는 DropShadow Filter를 사용할 수 있지만 그림자의 blur 값이 '0'과 같은 표현으로 처리되고 blur 값을 지정할 수 없는 단점이 있다. IE 브라우저 버전간 호환(버그 해결)을 위해 display 속성을 block 또는 inline-block 으로 지정하고 zoom:1 속성을 추가로 부여해야 한다.

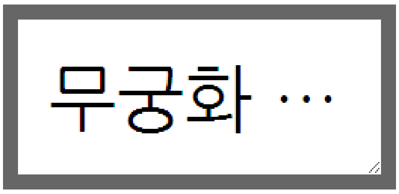
문법	
Name:	*text-shadow*
Value:	none | [<color>? <length> <length> <length>? | <length> <length> <length>? <color>?]
Initial:	None

Applies to:	all elements and generated content
Inherited:	Yes
Percentages:	N/A
Media:	Visual
Computed value:	a color plus three absolute <length>s

- CSS Text Level 3 › text-shadow - http://www.w3.org/TR/css3-text/#text-shadow
- MSDN › DropShadow Filter - http://msdn.microsoft.com/en-us/library/ms532985(VS.85).aspx

3.3.2 text-overflow






```
text-overflow:ellipsis;
-o-text-overflow:ellipsis;
```

지원 브라우저 : 

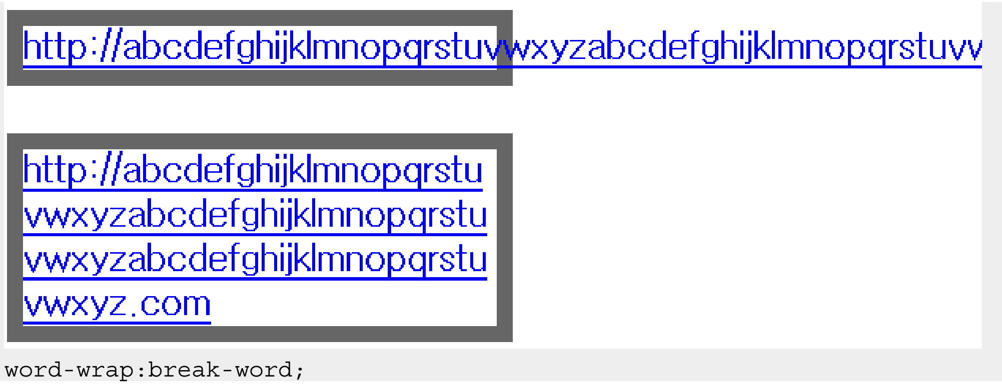
Firefox 브라우저는 아직 지원하지 않고 있지만 IE는 이미 IE6 버전부터 지원하고 있다. Firefox 브라우저는 overflow-hidden 속성에 의하여 넘치는 부분이 말 줄임표 없이 잘린다.

문법	
Name:	*text-overflow*
Value:	clip | ellipsis | <string>
Initial:	Clip
Applies to:	block-level, inline-block elements and table cells
Inherited:	No
Percentages:	N/A

Media:	Visual
Computed value:	as specified

- CSS Text Level 3 › text-overflow - http://dev.w3.org/csswg/css3-text/#text-overflow

3.3.3 word-wrap



지원 브라우저 :

1byte 문자열이 공백 없이 등장하는 경우 모든 브라우저들은 이를 하나의 단어로 해석하기 때문에 개행하지 않는다. 이것을 강제로 개행할 수 있다. 모든 브라우저가 이 속성을 지원한다.

문법	
Name:	*word-wrap*
Value:	normal | break-word
Initial:	normal
Applies to:	all elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	specified value

- CSS Text Level 3 › word-wrap - http://www.w3.org/TR/css3-text/#word-wrap

3.3.4 @font-face

The quick brown fox jumps over the lazy dog.
이 문서의 기본 글꼴은 나눔고딕 입니다.
0123456789

```
@font-face{ font-family:ngttf; src:url(NanumGothic.ttf);}
@font-face{ font-family:ngeot; src:url(NanumGothic.eot);}
body, input, textarea, select, button{ font-family:NanumGothic,
맑은고딕, ngttf, ngeot;}
```

지원 브라우저 :

ttf 파일과 eot 파일을 함께 준비한다. 비 IE 브라우저는 ttf 포멧과 otf 포멧을 지원하지만 eot 포멧을 지원하지 않는다. 반면 IE 브라우저는 eot 포멧만 지원하기 때문에 두 가지 포멧이 모두 필요하다. 나눔고딕 글꼴을 이미 설치한 사용자는 웹 폰트를 내려받지 않고 로컬 글꼴을 사용할 수 있도록 NanumGothic을 먼저 선언한다.

한국어 글꼴인 경우 오페라 브라우저는 font-family 이름에 반드시 한글을 사용해야 한다.

font-family 문법

Name:	*font-family*
Value:	<family-name>
Initial:	N/A

src 문법

Name:	*src*
Value:	[<uri> [format(<string> [, <string>]*)] | <font-face-name>] [, <uri> [format(<string> [, <string>]*)] | <font-face-name>]*
Initial:	N/A

- CSS Fonts Module Level 3 › @font-face - http://www.w3.org/TR/css3-fonts/#the-font-face-rule

3.3.5 opacity

```
opacity:0.5;
filter:alpha(opacity=50);
```

지원 브라우저 :

IE 브라우저를 제외한 모든 브라우저가 이미 opacity 속성을 지원하고 있다. IE는 MS전용 Alpha Filter를 적용하여 동일한 표현이 가능하다.

문법	
Name:	*opacity*
Value:	<alphavalue> | inherit
Initial:	1
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	The same as the specified value after clipping the <alphavalue> to the range [0.0,1.0].

- CSS Color Module Level 3 › opacity - http://www.w3.org/TR/css3-color/#transparency
- MSDN › Alpha Filter - http://msdn.microsoft.com/en-us/library/ms532967(VS.85).aspx

3.3.6 _ box-shadow

```
box-shadow:10px 10px 10px silver;
-moz-box-shadow:10px 10px 10px silver;
```

```
-webkit-box-shadow:10px 10px 10px silver;
filter:progid:DXImageTransform.Microsoft.Shadow(color=silver,directio
n=135, strength=10); box-shadow:x_offset y_offset blur_radius color
```

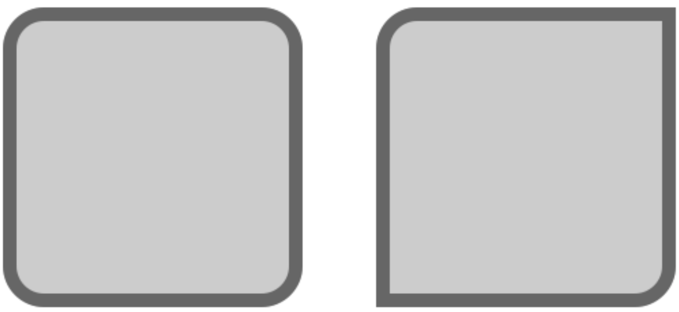
지원 브라우저 :

값은 'X좌표, Y좌표, blur값, 그림자색' 순으로 선언한다. blur 값은 생략할 수 있는데 생략하는 경우 기본값은 blur 스타일이 전혀 지정되지 않은 '0'이다. IE 브라우저는 Shadow Filter를 적용할 수 있으나 그림자를 한 방향으로만 표현할 수 있고 사방으로 표현할 수 없는 제약이 따른다. inset 값을 추가하면 그림자가 상자 안쪽으로 발생하는데 IE filter 로는 이런 표현이 불가능하다. Safari4는 inset 값을 지원하지 않지만 Safari5는 지원한다.

문법	
Name:	*box-shadow*
Value:	none | *<shadow>* [, *<shadow>*]*
Initial:	none
Applies to:	all elements
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	any <length> made absolute; any color computed; otherwise as specified

- CSS Backgrounds and Borders Module Level 3 › box-shadow - http://www.w3.org/TR/css3-background/#the-box-shadow
- MSDN › Shadow Filter - http://msdn.microsoft.com/en-us/library/ms533086(v=VS.85).aspx

3.3.7 border-radius



지원 브라우저 :

```
border-radius:30px;
-moz-border-radius:30px;
-webkit-border-radius:30px;
```

네 방향 모두 적용하는 경우.

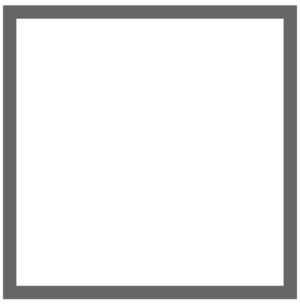
```
border-radius:30px 0;
-moz-border-radius:30px 0;
-webkit-border-top-left-radius:30px;
-webkit-border-bottom-right-radius:30px;
```

원하는 방향에만 적용하는 경우. 웹킷 브라우저 가운데 사파리 4는 단축 속성을 지원하지 않아 현재로써는 단축 속성을 사용할 수 없다.

<table>
<tr><th colspan="2">문법</th></tr>
<tr><td>Name:</td><td>border-radius</td></tr>
<tr><td>Value:</td><td>[<<i>length</i>> | <<i>percentage</i>>]{1,4} [/ [<<i>length</i>> | <<i>percentage</i>>]{1,4}]?</td></tr>
<tr><td>Initial:</td><td>0</td></tr>
<tr><td>Applies to:</td><td>all elements, except table element when 'border-collapse' is 'collapse'</td></tr>
<tr><td>Inherited:</td><td>no</td></tr>
<tr><td>Percentages:</td><td>Refer to corresponding dimension of the border box.</td></tr>
<tr><td>Media:</td><td>visual</td></tr>
<tr><td>Computed value:</td><td>see individual properties</td></tr>
</table>

- CSS Backgrounds and Borders Module Level 3 › border-radius - http://www.w3.org/TR/css3-background/#the-border-radius

3.3.8 background(s)



```
background:
url(bg.gif) no-repeat left top,
url(bg.gif) no-repeat right top,
url(bg.gif) no-repeat left bottom,
url(bg.gif) no-repeat right bottom;
```

지원 브라우저 : 

하나의 요소에 이런 배경 이미지가 4번 적용되었다. 속성의 값을 쉽

표(,)로 분리하면 배경 이미지의 사용 횟수를 무한대로 늘릴 수 있다.

문법	
Name:	*background*
Value:	[*<bg-layer>* ,]* *<final-bg-layer>*
Initial:	see individual properties
Applies to:	all elements
Inherited:	no
Percentages:	see individual properties
Media:	visual
Computed value:	see individual properties

- CSS Backgrounds and Borders Module Level 3 › background - http://www.w3.org/TR/css3-background/#background

3.3.9 gradient

```
background:#3EAF0E -webkit-gradient(linear, 0% 0%, 0% 100%,
from(#3EAF0E), to(#fff));
background:#3EAF0E -moz-linear-gradient(top, #3EAF0E, #fff);
filter:progid:DXImageTransform.Microsoft.gradient(startColorStr=#3EAF
0E, endColorStr=#ffffff);
```

지원 브라우저 :

웹킷(크롬, 사파리)과 파이어폭스 및 IE의 속성 및 값 선언 방식이 모두 다름에 유의한다. 웹킷은 (타입, X시작점 Y시작점, X종료점 Y종료점, 시작색, 종료색) 형식으로 선언하고 파이어폭스는 (시작점, 시작색, 종료색) 순으로 선언한다. gradient를 지원하지 않는 브라우저를 위하여 기본 배경색을 지정해두어야 함에 유의한다. IE는 IE 전용 Gradient Filter를 사용한다.

- Safari CSS Visual Effects Guide › gradients - http://developer.apple.com/safari/library/documentation/InternetWeb/Conceptual/SafariVisualEffectsProgGuide/Gradients/Gradients.html
- Mozila Developer Center › -moz-linear-gradient - https://developer.mozilla.org/en/CSS/-moz-linear-gradient
- MSDN › Gradient Filter - http://msdn.microsoft.com/en-

us/library/ms532997(v=VS.85).aspx

3.3.10 transform


```
-o-transform:rotate(90deg);-o-transform-origin:0 100%;
-moz-transform:rotate(90deg);-moz-transform-origin:0 100%;
-webkit-transform:rotate(90deg);-webkit-transform-origin:0 100%;
filter:progid:DXImageTransform.Microsoft.Matrix(M11=6.123031769111886
e-17, M12=-1, M21=1, M22=6.123031769111886e-17, sizingmethod='auto
expand');
```

지원 브라우저 : 

시계 방향으로 90도 회전시키는 코드. translate(이동), scale(크기), rotate(회전), skew(비틀기)가 가능하다. transform-oriain 속성은 transform의 축을 지정하는 속성으로써 값은 기본 값이 50%(X축) 50%(Y축) 이기 때문에 기본 값을 따르는 경우 생략할 수 있다. IE의 경우 IE 전용 Matrix Filter를 사용할 수 있으나 사용법이 다소 복잡하다.

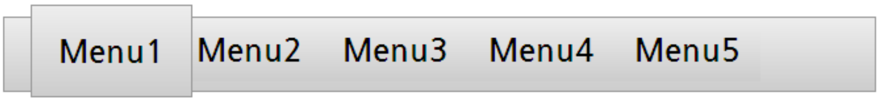
transform 문법

Name:	*transform*
Value:	none | <transform-function> [<transform-function>]*
Initial:	none
Applies to:	block-level and inline-level elements
Inherited:	no
Percentages:	refer to the size of the element's box
Media:	visual
Computed value:	Same as specified value.

transform-origin 문법	
Name:	*transform-origin*
Value:	[[<percentage> | <length> | left | center | right] [<percentage> | <length> | top | center | bottom]?] | [[left | center | right] || [top | center | bottom]]
Initial:	50% 50%
Applies to:	block-level and inline-level elements
Inherited:	no
Percentages:	refer to the size of the element's box
Media:	visual
Computed value:	For <length> the absolute value, otherwise a percentage

- CSS 2D Transforms Module Level 3 › transform - http://www.w3.org/TR/css3-2d-transforms/#transform-property
- MSDN › Matrix Filter - http://msdn.microsoft.com/en-us/library/ms533014(v=VS.85).aspx

3.3.11 transition




```
a{padding:10px 15px; margin:0;
  -o-transition:0.2s;
  -webkit-transition:0.2s;
  }

a:hover{padding:15px 20px; margin:-6px; opacity:0.8;}
```

지원 브라우저 :

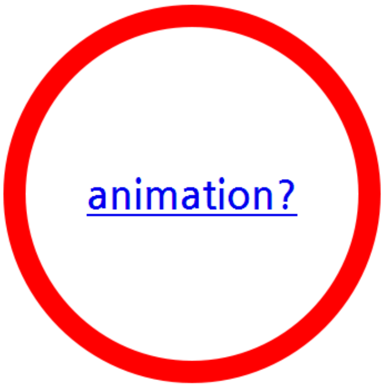
메뉴 버튼에 마우스를 올리면 플래시 메뉴와 같은 부드러운 움직임이 발생한다. transition 속성의 값으로는 어떤 속성을 몇 초간 진행할 것인지 정의한다. 어떤 속성을 transition 할 것인지 지정하지 않으면 기본 값 all이 지정되어 모든 속성을 transition 시킨다. 파이어폭스 브라우저는 3.7 버전부터 지원을 기대할 수 있다.

문법	
Name:	*transition*
Value:	[<'transition-property'> || <'transition-duration'> || <'transition-timing-function'> || <'transition-delay'> [, [<'transition-property'> || <'transition-duration'> || <'transition-timing-function'> || <'transition-delay'>]]*
Initial:	see individual properties

Applies to:	all elements, :before and :after pseudo elements
Inherited:	no
Percentages:	N/A
Media:	interactive
Computed value:	Same as specified value.

- CSS Transitions Module Level 3 › transition - http://www.w3.org/TR/css3-transitions/#the-transition-shorthand-property-

3.3.12 animation




```
.ani { -webkit-animation:webkitFire infinite 1s linear;}

@-webkit-keyframes 'webkitFire'{
  from{ -webkit-transform:rotate(0deg);}
  to{-webkit-transform:rotate(360deg);}
}
```

지원 브라우저 :

'webkitFire'라는 사용자 정의 에니메이션이 실행 된다. 일정한(linear) 속도로 1초(1s)에 한 바퀴(0~360deg)씩 영원히(infinite) 돌게 된다.

문법	
Name:	*animation*
Value:	[<animation-name> || <animation-duration> || <animation-timing-function> || <animation-delay> || <animation-iteration-count> || <animation-direction>] [, [<animation-name> || <animation-duration> || <animation-timing-function> || <animation-delay> || <animation-iteration-count> || <animation-direction>]]*
Initial:	see individual properties
Applies to:	block-level and inline-level elements
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	Same as specified value.

- CSS Animations Module Level 3 › animation - http://www.w3.org/TR/css3-animations/#the-animation-shorthand-property-

3.4 CSS3 명세 읽는 법

CSS3 명세 읽는 법	
용어	해설
Value	유효한 값 또는 문법
Initial	기본 값 (값을 지정하지 않은 경우의)
Applies to	속성을 적용할 수 있는 요소 (block, inline, inline-block ... all elements)
Inherited	속성이 자손에게 상속 되는지의 여부 (yes, no)
Percentage	퍼센티지 값을 적용할 수 있는지의 여부 (N/A 는 not applicable 또는 not available 으로써 적용할 수 없다는 의미)
Media	어떤 미디어 그룹에 적용할 수 있는 속성인지의 여부 (visual, interactive ...)
Computed value	절대 값이 어떻게 계산 되는지
<...>	포함되는 데이터 유형을 변수 형식으로 설명 (예를 들면 <length>, <percentage>)
a b	공백으로 분리되어 나열된 값들은 모든 값이 반드시 필요하고 순서를 지켜야 한다
|	둘 이상의 값이 이것으로 분리되어 있으면 하나의 값은 반드시 필요하고 하나만 필요하다
| |	둘 이상의 값이 이것으로 분리되어 있으면 하나 또는 하나 이상 반드시 필요하고 순서는 지키지 않아도 된다
&&	둘 이상의 값이 이것으로 분리되어 있으면 모든 값이 반드시 필요하고 순서는 지키지 않아도 된다
[...]	값을 그룹핑 한다
*	선행되는 유형의 값은 생략하거나 한 번 이상 등장할 수 있다
+	선행되는 유형의 값은 반드시 필요하고 한 번 이상 필요하다
?	선행되는 유형의 값은 생략하거나 한 번만 필요하다
{a,b}	선행되는 유형의 값은 적어도 'a' 번 필요하고 최대한 'b' 번 필요하다. (예를 들면 border-color 속성의 Value는 <color>{1,4} 이다)
(...)	인용부호 밖에 있는 괄호는 값의 묶음을 위해 사용되므로 문자 그대로 출력해야 한다 (예 background:url(...);)
/	인용부호 밖에 있는 슬러시는 값의 연결을 위해 사용되므로 문자 그대로 출력해야 한다 (예 font:12px/1.2 Sans-serif;)
,	인용부호 밖에 있는 쉼표는 값의 연결을 위해 사용되므로 문자 그대로 출력해야 한다 (예 font-family:Tahoma, Geneva, sans-serif;)

3.5 FAQ

CSS3는 언제 지원 되나?

CSS의 경우 공식 표준(de facto standard) 보다 사실 표준(de jure standard)이 더욱 의미가 있다. 그 이유는 현존하는 웹 브라우저들이 권고 후보(CR:Candidate Recommendation) 단계에 머물러 있는 CSS 2.1 명세의 대부분을 이미 잘 지원하고 있기 때문. CSS 2.1은 공식 표준(REC:Recommendation)이 되기 위하여 권고 제의(PR:Proposed Recommendation) 라는 단계를 한 번 더 거쳐야 하지만 이미 현실 세계에서는 사실상의 표준이 되어 있는 샘이다. 따라서 CSS3가 공식 표준이 될 때까지 기다릴 필요가 없다. 웹 브라우저들이 지원하기만 한다면 CSS3의 새로운 기술 명세를 현재의 실무에 즉시 적용할 수 있다. 이미 현존하는 최신 웹 브라우저들은 앞다투어 CSS3의 유용한 명세들을 지원하기 시작했다.

CSS3를 사용하면 이를 잘 지원하지 못하는 하위 버전 브라우저를 선택한 사용자를 차별하는 것이 아닌가?

전혀 그렇지 않다. CSS는 HTML 문서의 화면 표시 스타일을 기술하는 언어로써 HTML 구조를 변경하거나 의미를 교정하는 언어가 아니다. 따라서 CSS는 의도하지 않은 상황에서도 자연스럽게 우아한 낮춤(Graceful Degradation)과 점진적 향상(Progressive Enhancement) 원칙에 충실하다. 최신 웹 브라우저를 선택한 사용자에게는 더 세련되고 화려하면서 역동적인 모습을 보여줄 것이다. 그러나 오래된 웹 브라우저를 선택한 사용자도 HTML 문서를 읽거나 듣는데 전혀 문제가 되지 않는다. 이것은 상호 운용성(interoperability) 또는 웹 접근성(web accessibility)의 어느 측면에서 보더라도 차별에 해당하지 않는다. 더 좋은 도구를 선택한 사용자에게 더 좋은 사용자 경험을 제공하는 것은 당연한 이치로써 이것은 차별을 의미하지 않는다.

CSS3를 사용하면 CSS 문법 유효성 검사를 만족할 수 없지 않나?

그렇다. CSS3를 사용하면 CSS 문법 유효성 검사를 만족시킬 수 없다. 그러나 그것은 문제가 되지 않는다. CSS 문법 유효성 검사 결과는 현실 세계에서 중요하지 않다. CSS의 사용에 있어서 중요한 것은 사실상의 표준이고 이것은 현존하는 웹 브라우저들의 지원 여부에 따른다. 만약 누군가 CSS의 문법 유효성을 완벽하게 지켜야 한다고 주장한다면 W3C 웹 사이트의 CSS 소스 코드를 열어보라고 권해줄 수 있다. W3C 공식 웹사이트(w3.org)는 이미 CSS3 속성을 사용하여 그래픽 이미지를 최소화 하면서도 웹 페이지를 화려하게 디자인 했다. CSS3를 지원하지 않는 IE 브라우저 사용자에게는 조금씩 다르게 보이는 디자인이 존재한다. 그러나 누구도 W3C

가 IE 브라우저 사용자를 차별했다고 말하지는 않는다.

CSS3의 명세를 지금부터 공부해야 하나?

2010년 현재 CSS3는 아직 모든 명세가 확정되지 않았고 웹 브라우저들이 모든 명세를 잘 지원하는 것도 아니다. 따라서 최신 브라우저에서 지원하는 유용한 몇 가지 속성들을 먼저 사용해보면서 자연스럽게 점진적으로 익혀가는 것이 좋다. 또한 CSS2 버전에서 지원하던 거의 모든 속성과 선택자 사용법을 계승하고 있기 때문에 새 표준에서 추가된 명세들 가운데 브라우저들이 지원하는 명세들을 먼저 익히면 된다. 여러분이 CSS를 처음 배울때에도 아마 이런 방식으로 접근했을 것이다.

CSS3 명세가 갑자기 바뀌면 어떻게 하나?

CSS3 명세가 바뀌는 것은 중요한 사실이 아니다. 웹 브라우저가 바뀐 CSS3 명세를 지원하는지의 여부가 중요하다. 보통의 웹 브라우저들은 하위 호환성을 중요하게 여기기 때문에 새 표준을 수용하면서 낡은 명세의 지원을 차단해 버리는 어리석은 일은 하지 않을 것이다. 예를 들어 CSS3 명세는 상자의 한 쪽 귀퉁이에 둥근 모서리 효과를 지정할 때 border-top-left-radius 라는 속성을 사용할 것을 제시하고 있지만 파이어폭스 브라우저는 둥근 모서리를 표현할 때 -moz-border-radius-topleft 라는 속성을 사용해야 한다. 만약 border-top-left-radius 속성이 최종 권고안으로 확정 되더라도 파이어폭스 브라우저는 -moz-border-radius-topleft 속성의 지원을 멈추지 않을 것이다.

CSS3 공부했지만 대부분의 브라우저들이 지원하지 않는것 같다.

지원하지 않는 것이 아니라 사용 방법이 조금 다를 뿐이다. 웹 브라우저들은 CSS3 명세를 지원하지만 미처 발견하지 못했던 매우 특이한 경우의 문제점이나 개선 사항을 수집하기 위해 보다 안전한 방법으로 CSS3를 지원하는 방법을 선택했다. 그것이 바로 벤더 확장(Vendor Extensions) 속성이다. 벤더 확장 속성이란 CSS 속성 앞에 벤더 표시를 위한 접두사(prefix)를 붙인 상태의 속성 표기법이다. 오페라 브라우저는 -o-, 파이어폭스는 -moz-, 웹킷 엔진을 사용하는 사파리와 크롬은 -webkit- 이라는 접두사를 붙임으로써 현재 지원하는 속성이 공식적인 표준과 다를 수 있고 변경의 여지가 있을 수 있다는 점을 암시하고 있다. 예를 들면 CSS3의 box-shadow 속성을 작성할 때 파이어폭스는 -moz-box-shadow 라고 작성해

야 한다. 파이어폭스가 이 속성을 공식 지원하기 시작한다면 box-shadow와 -moz-box-shadow 표기법을 병행 지원 할 것이다. 다른 벤더들도 마찬가지다.

고객들이 IE 브라우저에서도 CSS3 기법들이 적용되길 원한다.

안타깝게도 현재(IE8) 및 하위 버전의 IE 브라우저는 CSS3를 지원하지 않는다. IE9부터 HTML5와 함께 CSS3를 지원할 계획이 발표되었고 IE9 브라우저는 Windows Vista, Windows 7 이후의 운영체제에만 설치 가능하다. IE9 브라우저는 아직(2010년 7월 현재) 공식 출시일을 밝힌 적이 없지만 빠르면 2011년에 출시가 될 것으로 전망하고 있다. CSS3가 최신 기술이기 때문에 하위 버전의 브라우저에서 똑같이 구현하려면 그래픽 이미지 또는 자바스크립트를 추가로 사용해야 한다. 때문에 무리하게 적용하는 경우 웹 페이지의 성능을 떨어뜨리는 결과를 감수해야 한다. 평균적으로 웹 페이지 로딩 속도의 80%는 이미지와 자바스크립트를 내려받는데 소요된다. 트래픽이 높은 웹 사이트 일수록 이러한 성능 문제는 치명적으로 작용하기 때문에 낮은 버전의 웹 브라우저에는 우아한 낮춤 전략으로 접근하는 것이 좋다. 예를 들면 낮은 버전의 웹 브라우저에서는 둥근 모서리 대신 각진 모서리 표현을 사용하는 것이다. 이것은 고객이나 디자이너의 욕심을 포기하라는 의미가 아니라 성능과 타협할 수 있도록 합의를 이끌어야 한다는 뜻이다. 결과적으로 모든 사용자들에게 높은 성능을 제공할 수 있고 최신 웹 브라우저를 선택한 사용자에게는 보다 향상된 경험을 제공할 수 있다. 낡은 브라우저 때문에 계속해서 시대에 뒤떨어진 낡은 기술만을 사용해야 한다면 향상된 경험도 제공할 수 없게 된다.

참고자료

- W3C Standards & Draft
  - About the CSS 2.1 Specification - http://www.w3.org/TR/CSS21/about.html
  - CSS3.info - http://www.css3.info/
  - CSS3 Working Group Editor's Draft - http://dev.w3.org/csswg/
  - Selectors Level 3 - http://www.w3.org/TR/css3-selectors/
  - CSS Text Level 3 - http://www.w3.org/TR/css3-text/
  - CSS Fonts Module Level 3 - http://www.w3.org/TR/css3-fonts/
  - CSS Color Module Level 3 - http://www.w3.org/TR/css3-color/
  - CSS Backgrounds and Borders Module Level 3 - http://www.w3.org/TR/css3-background/
  - CSS 2D Transforms Module Level 3 - http://www.w3.org/TR/css3-2d-transforms/
  - CSS 3D Transforms Module Level 3 - http://www.w3.org/TR/css3-3d-transforms/
  - CSS Transitions Module Level 3 - http://www.w3.org/TR/css3-transitions/

  - CSS Animations Module Level 3 - http://www.w3.org/TR/css3-animations/
- ALL STANDARDS AND DRAFTS › CSS - http://www.w3.org/TR/#tr_CSS
- Safari CSS Reference - http://developer.apple.com/safari/library/documentation/AppleApplications/Reference/SafariCSSRef/Introduction.html
- Mozilla CSS Extensions - https://developer.mozilla.org/en/CSS_Reference/Mozilla_Extensions
- Opera Web specifications support - http://www.opera.com/docs/specs/
- MSDN › Cascading Style Sheets - http://msdn.microsoft.com/en-us/library/ms531205(v=vs.85).aspx
- MSDN › Visual Filters and Transitions Reference - http://msdn.microsoft.com/en-us/library/ms532853(v=VS.85).aspx

4. HTML5 API

HTML5에서 공식으로 채택한 자바스크립트 API들에 대하여 이해하고 예제를 통한 사용방법을 알아보자. 여기에 소개된 예제들은 참고사이트에서 제공하는 내용을 번역 또는 실정에 맞게 의역한 것이다.

그리고 대부분의 현대 브라우저들이 지원 하거나 개발 진행중인 내용들만을 다루고 있어 본인이 직접 개발하는데에는 어려움이 없을 것으로 생각된다. 지면 관계상 세세한 명세까지는 모두 다루지 못한 점 양해 바라며, 응용력을 십분 발휘하여 보다 성능이 좋고 네이티브 애플리케이션다운 웹 애플리케이션을 개발하는데 도움되길 바라는 취지로 작성했다.

4.1 HTML5 미디어 요소

4.1.1 Audio와 Video 요소

지원 브라우저 :

HTML5에서는 새로운 미디어 요소인 <audio>와 <video> 요소를 지원한다. 이 요소들을 이용하여 별도의 브라우저 플러그인을 이용하지 않고 미디어를 웹 페이지에 쉽게 추가 할 수 있다. 그리고 스크립트를 이용하여 미디어를 직접 제어할 수 있다. HTML5를 지원하지 않는 브라우저라라도 크게 걱정할 필요없다. 종전과 같이 플래시 등과 같은 플러그인을 이용하여 미디어를 재생할 수 있기 때문이다. 특히, <canvas> 요소를 결합하면 영상의 실시간 비드맵 연산이 가능하기 때문에 다양한 그래픽 효과를 부여하거나 일종의 영상 판독기와 같은 애플리케이션을 오픈 웹기술만으로 개발할 수 있다.

미디어를 스크립트로 다루는 일은 <video> 요소와 <audio> 요소를 구분할 필요없이 동일하게 취급할 수 있다. HTML 문서에 <video> 요소를 삽입해 보고 스크립트를 이용한 제어 방법에 대하여 간단히 알아보자.

<video> 요소의 마크업

웹 페이지에 비디오 요소와 외부 컨트롤을 위한 요소를 작성한다. 만약, 요소의 속성으로 "controls"을 지정하면 브라우저에서 제공하는 기본 컨트롤러를 사용할 수 있다.

```
<video autoplay>
  <source src="foo.ogg" type="video/ogg">
  <source src="foo.mp4" type="video/mp4">
```

```
  <source src="foo.webm" type="video/webm">
  당신의 브라우저는 <code>&lt;video&gt;</code> 요소를 지원하지 않습니다.
</video>
<input type="button" onclick="playPause()" value="Play/Pause">
```

스크립트를 이용한 미디어 제어

playPause 함수의 내용은 다음과 같다.

```
function playPause() {
  var myVideo = document.getElementsByTagName('video')[0];
  if (myVideo.paused)
    myVideo.play();
  else
    myVideo.pause();
}
```

다양한 이벤트를 등록하여 미디어의 재생 상황을 모니터링하고 대응할 수 있다.

```
myVideo.addEventListener('ended', function () {
  alert('video playback finished')
});
```

데모: http://html5.firejune.com/demo/video.html

만약, 스크립트만으로 오디오를 재생하고 싶다면 다음과 같이 작성해도 무방하다.

```
var audio = new Audio("song.mp3");
audio.play();
```

브라우저에서 비디오 요소를 지원하는지를 검사하는 방법은 다음과 같다.

```
!!document.createElement('video').canPlayType
```

그리고 브라우저가 지원하는 코덱을 검사할 수도 있다.

```
var v = document.createElement('video');
var supported = v.canPlayType('video/mp4; codecs="avc1.58A01E,
mp4a.40.2"');
if ( supported == 'probably') { return true; }
```

미디어 요소의 이벤트들

Event name	Description
abort	Sent when playback is aborted; for example, if the media is playing and is restarted from the beginning, this event is sent.
canplay	Sent when enough data is available that the media can be played, at least for a couple of frames. This corresponds to the

	CAN_PLAY readyState.
canplaythrough	Sent when the ready state changes to CAN_PLAY_THROUGH, indicating that the entire media can be played without interruption, assuming the download rate remains at least at the current level.
canshowcurrentframe	The current frame has loaded and can be presented. This corresponds to the CAN_SHOW_CURRENT_FRAME readyState.
dataunavailable	Sent when the ready state changes to DATA_UNAVAILABLE.
durationchange	The metadata has loaded or changed, indicating a change in duration of the media. This is sent, for example, when the media has loaded enough that the duration is known.
emptied	The media has become empty; for example, this event is sent if the media has already been loaded (or partially loaded), and the load() method is called to reload it.
empty	Sent when an error occurs and the media is empty.
ended	Sent when playback completes.
error	Sent when an error occurs. The element's error attribute contains more information.
loadeddata	The first frame of the media has finished loading.
loadedmetadata	The media's metadata has finished loading; all attributes now contain as much useful information as they're going to.
loadstart	Sent when loading of the media begins.
pause	Sent when playback is paused.
play	Sent when playback starts or resumes.
progress	Sent periodically to inform interested parties of progress downloading the media. The progress event has three attributes: lengthComputable true if the total size of the media file is known, otherwise false. loaded The number of bytes of the media file that have been received so far. total The total number of bytes in the media file.
ratechange	Sent when the playback speed changes.
seeked	Sent when a seek operation completes.
seeking	Sent when a seek operation begins.
suspend	Sent when loading of the media is suspended; this may happen either because the download has completed or because it has been paused for any other reason.
timeupdate	The time indicated by the element's currentTime attribute has changed.
volumechange	Sent when the audio volume changes (both when the volume is set and when the muted attribute is changed).
waiting	Sent when the requested operation (such as playback) is delayed pending the completion of another operation (such as a seek).

위 표는 Mozilla Developer Center로부터 발췌한 이벤트 목록이다. 미디어

요소에는 매우 다양한 상황별 이벤트를 제공하는 것을 확인할 수 있다. HTML5 미디어 요소들의 명세는 아직도 진행중이며 코덱, 스트리밍 등 여전히 다양한 이슈들을 안고있다.

4.1.2 Canvas 요소

지원 브라우저 :

사실, <canvas> 요소는 HTML5가 나오기 전 부터 존재했었고 다양한 용도로 사용되어 왔었지만 이제서야 HTML5의 공식 명세로 자리잡았다. 이 요소를 사용하면 2차원의 비트맵 이미지 프로세싱이 가능하고 동적인 그래픽 렌더링을 스크립트로 제어할 수 있다. 이는 웹 페이지에 인터랙티브한 그래픽 컨텐츠를 만들어 제공할 수 있음을 뜻한다. 화려한 그래픽 기반의 게임이나, 다양한 종류의 그래프, 이미지 합성 또는 변형, 드로잉 애플리케이션 등 마치 플래시로 생성된 것과 같은 컨텐츠를 제공할 수 있게 되는 것이다.

간단한 예제

<canvas> 요소에 빨간색 사각형을 그려보자.

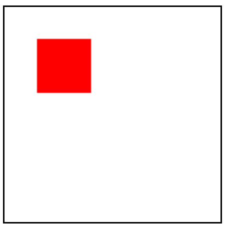
아래 코드는 HTML 문서에 <canvas> 요소를 작성한 것이다.

```
<canvas id="example" width="200" height="200">
이 메시지는 사용자의 브라우저에서 HTML5 캔버스를 지원하지 않는 경우 표시 됨
</canvas>
```

스크립트를 사용하여 <canvas> 요소에 사각형을 그려 넣는다.

```
var example = document.getElementById('example');
var context = example.getContext('2d');
context.fillStyle = "rgb(255,0,0)";
context.fillRect(30, 30, 50, 50);
```

다음과 같은 결과를 얻을 수 있다.



데모: http://html5.firejune.com/demo/canvas.html

4.1.3 SVG 요소

지원 브라우저 :

HTML5 명세에 포함되면서부터 표준으로 자리매김한 SVG는 확장 가능한 백터 그래픽(Scalable Vector Graphics)의 줄임말이다. 2차원 백터 그래픽만을 표현하며, XML형식으로 작성되고, SVG 뷰어를 이용하는 등 다양한 삽입 방법으로 사용자가 조회할 수 있다. SVG의 작성은 HTML과 매우 유사하다. <circle>, <rect>등과 같은 그래픽 태그들을 이용하여 작성하면 된다. SMIL 또는 스크립트를 이용하여 동적인 변화를 주거나 CSS를 지정하여 모양을 꾸밀 수도 있다.

SVG와 스크립트를 접목하여 상호작용이 발생하는 차트, 다이어그램, 일러스트레이트 등 선명한 화질을 가진 확대 가능한 자료를 웹 페이지에 삽입할 수 있으며, 마인드맵, 목업과 같은 애플리케이션을 개발할 수 있다.

SVG 요소의 마크업

보통 아래와 같은 형식으로 HTML 문서에 마크업 한다.

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
   <rect x="10" y="10" height="100" width="100"
         style="stroke:#ff0000; fill: #0000ff"/>
</svg>
```

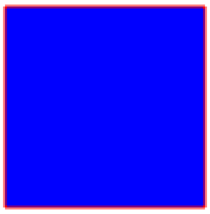
또는 리소스를 지정하는 형식으로도 삽입할 수 있다. 경우에 따라서는 svg 뷰어 브라우저 플러그인을 필요로 하기도 한다.

```
<!-- object 요소 사용 -->
<object data="/svg/examples.svg" width="300" height="100"
type="image/svg+xml"
       codebase="http://www.adobe.com/svg/viewer/install/" />

<!-- embed 요소 사용 -->
<embed src="/svg/examples.svg" width="500" height="200"
type="image/svg+xml"
      pluginspage="http://www.adobe.com/svg/viewer/install/" />

<!-- iframe 요소 사용 -->
<iframe src="/svg/examples.svg" width="300" height="100"></iframe>
```

삽입 결과는 다음과 같다.



데모: http://html5.firejune.com/demo/svg.html

SVG와 스크립트

SVG에 스크립트로 애니메이션을 하거나 변화를 주는 일은 DOM을 스크립트로 다루는 것과 별반 다르지 않다. 다음은 SVG가 가진 특정한 요소를 스크립트와 SMIL을 이용하여 애니메이션하는 예제이다.

```
var svgDocument;
var svgns = 'http://www.w3.org/2000/svg';
var xlinkns = 'http://www.w3.org/1999/xlink';

function startup(evt){
  P=document.getElementById("P")
  CL=document.getElementById("CL")
  animate()
  stop("S")
  stop("L")
}

limit=720
blu=4
speed=6
running=true
function animate(){
  if (!running) return
  B="rotate("+blu+" 360 150)"
  C="rotate("+(-blu/2)+" 360 150)"
  CL.setAttribute ("transform", B);
  P.setAttribute ("transform", C);
  blu=blu+speed
  if ((blu<0)||(blu>limit)) speed=-speed
  window.setTimeout("animate()",10)
}
runAnim=new Object
runAnim["S"]=false
runAnim["L"]=false
function stop(id){

  if (runAnim[id]) {
    document.getElementById(id).firstChild.nextSibling.endElement()
    document.getElementById("E"+id).endElement()
  }
  else{
    document.getElementById(id).firstChild.nextSibling.beginElement()
    document.getElementById("E"+id).beginElement()
```

```
  }
  runAnim[id]=!runAnim[id]
}
```

데모: http://html5.firejune.com/demo/svg-script.svg

4.2 HTML5 API

4.2.1 Offline Application Cache

지원 브라우저 :

HTML5는 오프라인 환경을 고려한 API들(Storage, Database)이 있다. 하지만 이 API들이 오프라인에서 정상적으로 작동하기 위해서는 결국 CSS, 이미지, 자바스크립트 등과 같은 리소스를 필요로한다. 이러한 환경를 궁극적으로 충족시켜 주는 것이 바로 Application Cache API이다. 오프라인 상태에서도 웹 애플리케이션으로의 접근을 가능케 하는 매우 중요한 기능이다. 이것은 기존 브라우저들이 사용하던 캐싱 메커니즘과는 차이가 있다. 오프라인 상태에서 개발자의 예상대로 작동하고 신뢰할 수 있는 새로운 인터페이스인 것이다. 이 것은 다음과 같은 3가지 장점을 가진다:

- 오프라인 브라우징 - 오프라인 상태에서도 사용자가 사이트에 접근할 수 있다.
- 속도 향상 - 로컬 영역에 저장된 리소스들은 매우빠른 로드 속도로 호출된다.
- 서버 부하 감소 - 브라우저는 오직 리소스가 변경된 경우에만 다운로드를 시도한다.

캐시 파일 목록 참조하기

`<html>` 태그에 manifest 속성을 지정하여 캐시할 파일들의 목록을 지정할 수 있다. 여기에 지정되는 파일은 간단한 텍스트 파일이며, 파일의 절대 또는 상대 경로를 기입하여 참조한다.

```
<html manifest="http://foo.example.com/example.manifest">
  ...
</html>
```

그리고 .manifest 파일의 mime-type은 반드시 "text/cache-manifest" 이어야 한다.

manifest 파일의 구조

다음은 간략하게 manifest를 구성한 파일이다.

```
CACHE MANIFEST
index.html
stylesheet.css
images/logo.png
scripts/main.js
```

manifest를 사용할 때 몇가지 주의해야할 점이 있다. "CACHE MANIFEST" 문자열은 항상 첫 번째 라인에 위치해야 하며, 사이트당 최대 5MB까지 캐시할 수 있다. 만약에 manifest 파일이나 또는 manifest에 명시된 파일의 다운로드가 실패할 경우 브라우저는 가장 최근에 성공적으로 다운로드한 파일을 그대로 사용하며, 실패 이벤트를 발생한다.

이제 조금 더 복잡하게 구성해 보자.

```
CACHE MANIFEST
# v2

# 명시적으로 캐시된 항목
CACHE:
index.html
stylesheet.css
images/logo.png
scripts/main.js

# 사용자가 온라인 상태가 되었을 때 필요한 리소스들
NETWORK:
login.php
/myapi
http://api.twitter.com

# static.html 파일은 main.py 파일에 접근할 수 없을 때 보여짐
FALLBACK:
/main.py /static.html
```

'#'으로 시작하는 문자열은 주석을 뜻한다. "CACHE:" 문자열 아래로 명시된 파일들은 간략하게 구성했을 때와 마찬가지로 로컬 영역에 파일을 캐시하는 것이다. "NETWORK:" 문자열 아래로 명시된 파일들은 온라인 상태가 되었을 때에만 접근을 허용하고 그렇지 않은 경우 우회한다. "FALLBACK:" 문자열 아래로 명시된 파일들은 해당 파일에 접근 할 수 없는 경우 대체 페이지를 지정한 것이다. 첫 번째 URI는 리소스 이며, 두 번째는 대체되는 파일이다.

주의해야 할 점은 섹션들을 임의의 순서로 나열해도 무방하지만 각 섹션은 하나 이상의 항목이 존재해야 한다는 점이다.

manifest에 명시된 파일들은 오직 manifest파일이 서버에서 갱신된 경우에만 다운로드를 시도한다. 그러나 스크립트를 이용하여 수동으로 갱신할

수도 있다. 이는 '스크립트로 캐시 갱신하기'에서 자세한 사용법을 다루도록 한다.

다음 예제는 모든 페이지를 정의한 것이다. offline.html은 오프라인 상태에서 루트("/")로 접근한 경우 보여지게 될 것이다. 그외 다른 리소스들은 몇몇 이미지 파일을 제외하고 모두 인터넷 연결을 필요로 한다.

```
CACHE MANIFEST
# v3

# 명시적으로 캐시된 항목
index.html
css/style.css

# offline.html 파일은 사용자가 오프라인이 되었을 때 보여짐
FALLBACK:
/ /offline.html

# 사이트의 모든 리소스는 온라인을 필요로 함
NETWORK:
*

# 추가적인 리소스 캐시
CACHE:
images/logo1.png
images/logo2.png
images/logo3.png
```

스크립트로 캐시 갱신하기

오프라인이 되는 순간 이벤트가 발생하며, 이러한 경우는 다음 중 하나이다.

사용자가 당신의 사이트에 대한 브라우저의 데이터 스토리지를 삭제한 경우

.manifest 파일이 수정된 경우

프로그램에 의해 캐시가 갱신된 경우

`window.applicationCache` 개체를 이용하여 캐시를 프로그램적으로 접근하여 관리할 수 있다. 그리고 `status` 프로퍼티를 확인하면 다음과 같이 현재 캐시 상태를 확인할 수 있다.

```
var appCache = window.applicationCache;

switch (appCache.status) {
  case appCache.UNCACHED: // UNCACHED == 0
```

```
    return 'UNCACHED';
    break;
  case appCache.IDLE: // IDLE == 1
    return 'IDLE';
    break;
  case appCache.CHECKING: // CHECKING == 2
    return 'CHECKING';
    break;
  case appCache.DOWNLOADING: // DOWNLOADING == 3
    return 'DOWNLOADING';
    break;
  case appCache.UPDATEREADY:  // UPDATEREADY == 5
    return 'UPDATEREADY';
    break;
  case appCache.OBSOLETE: // OBSOLETE == 5
    return 'OBSOLETE';
    break;
  default:
    return 'UKNOWN CACHE STATUS';
    break;
};
```

프로그램적으로 캐시를 갱신하려면 `applicationCache.update()`를 호출하면 된다. 이 명령은 사용자의 캐시를 업데이트하려고 시도할 것이다. 단, manifest파일이 변경되었을 경우이다. `applicationCache.status`를 확인한 결과가 "UPDATEREADY"(갱신 준비) 상태라면 `applicationCache.swapCache()`를 호출하여 오래된 캐시를 새로운 파일로 교체할 수도 있다.

```
var appCache = window.applicationCache;

appCache.update(); // 사용자의 캐시를 갱신하도록 시도함

...

if (appCache.status == window.applicationCache.UPDATEREADY) {
  appCache.swapCache();  // 가져오기에 성공한 경우 새로운 캐시로 교체
}
```

그리고 이벤트 리스너에 다음과 같은 다양한 이벤트를 할당하여 캐시 파일을 프로그램적으로 관리할 수 있다.

```
function handleCacheEvent(e) {
  //...
}

function handleCacheError(e) {
  alert('Error: 젠장! 캐시 갱신을 실패하였습니다.');
};

// 최초 manifest의 캐시가 완료된 경우 이벤트 발생
appCache.addEventListener('cached', handleCacheEvent, false);
```

```
// 갱신 확인, 항상  이벤트가 순차적으로 발생
appCache.addEventListener('checking', handleCacheEvent, false);

// 갱신이 필요함. 브라우저가 리소스를 가져올 때 발생
appCache.addEventListener('downloading', handleCacheEvent, false);

// manifest 에서 404 또는 410 로 응답시 발생, 다운로드 실패
// 또는 다운로드 진행중에 manifest 가 변경된 경우
appCache.addEventListener('error', handleCacheError, false);

// 최초 manifest 다운로드시 발생
appCache.addEventListener('noupdate', handleCacheEvent, false);

// manifest 파일에서 404 또는 410 으로 응답시 발생
// 이러한 경우 애플리케이션 캐시에서 삭제된다.
appCache.addEventListener('obsolete', handleCacheEvent, false);

// manifest 로 부터 각각의 리소스를 가져올 때 발생
appCache.addEventListener('progress', handleCacheEvent, false);

// manifest 의 리소스가 새롭게 다시 다운로드 된 경우 발생
appCache.addEventListener('updateready', handleCacheEvent, false);
```

다시 한번 언급하지만, manifest에 명시된 리소스들 중 하나라도 다운로드에 실패하면 전체 업데이트 역시 실패한다. 이 때 브라우저는 기존의 애플리케이션 캐시를 이용하여 실행되고 실패 이벤트를 발생하게 된다는 사실을 기억하자.

데모: http://html5.firejune.com/demo/manifest.html

4.2.2 Web Storage

지원 브라우저 :

Web Storage는 일종의 클라이언트-사이드 데이터베이스이다. 이 데이터는 서버가 아닌 각 사용자의 브라우저에 보관된다. 일반 데이터베이스와의 두드러진 차이점은 우리에게 익숙한 key-value 형식으로 보관/갱신/호출한다는 것이다. 이것은 Web Storage를 사용하기위해 별도의 쿼리 문법이나 복잡한 메커니즘을 이해하지 않아도 됨을 의미한다. 그렇기 때문에 우리는 한가지만 기억하면 된다. Web Storage는 Web Database와 마찬가지로 브라우저에서 제공하는 저장공간을 사용한다는 것이다. 만약에 사용자가 사파리에서 파이어폭스로 전환하는 경우 동일한 데이터를 가져올 수 없다는 것을 유념하자.

Web Storage는 `localStorage`와 `sessionStorage`로 구분된다.

이들의 차이점은 브라우저가 완전히 종료되고 난 후에도 데이터가 유지 되느냐 마느냐이다. 데이터의 용도에 따라서 적절한 방식을 선택하면 된다.

간단한 사용법

자, 이제 간단한 몇 가지 코드를 살펴보자. 다음은 localStorage의 기본적인 사용법이다.

```
localStorage.setItem("name", "Hello World!"); // key-value 형식으로
저장
document.write(localStorage.getItem("name")); // 저장된 값 호출
localStorage.removeItem("name"); // 스토리지로 부터 일치하는 아이템 삭제
```

첫 번째 라인에서 "name"이라는 키에 "Hello World!"라는 새 항목을 Web Storage에 저장한 것이다. 여기에서 주의해야 할 점은 setItem의 두 번째 인자는 항상 문자(String) 형식으로 전달해야 한다. 두 번째 라인에서는 Web Storage로 부터 "name"키에 저장된 값을 document.write로 출력한 것이다. 세 번째 라인은 Web Storage에서 "name"키에 해당하는 데이터를 삭제한 것이다.

만약, 할당량을 초과한 경우 첫 번째 라인에서 오류가 발생하며 데이터가 저장되지 않을 것이다. 다음은 이 오류를 대처하는 방법이다.

```
try {
   localStorage.setItem("name", "Hello World!"); // key-value 형식으로
저장
} catch (e) {
   if (e == QUOTA_EXCEEDED_ERR) {
     alert('할당량 초과!'); // 할당량 초과로 인하여 데이터를 저장할 수 없음
   }
}
```

이제 브라우저에서 localStorage를 지원하지 않는 경우를 구분하자.

```
if (typeof(localStorage) == 'undefined' ) {
  alert('당신의 브라우저는 HTML5 localStorage 를 지원하지 않습니다. 브라우저를
업그레이드하세요.');
} else {
  try {
    localStorage.setItem("name", "Hello World!"); // key-value 형식으로
저장
  } catch (e) {
     if (e == QUOTA_EXCEEDED_ERR) {
       alert('할당량 초과!'); // 할당량 초과로 인하여 데이터를 저장할 수 없음
     }
  }

  document.write(localStorage.getItem("name")); // 저장된 값 호출
```

```
  localStorage.removeItem("name"); // 스토리지로 부터 일치하는 아이템 삭제
}
```

이상으로 Web Storage에 데이터를 저장하고, 호출하고, 삭제하는 간한한 사용법에 대하여 알아 보았다.

데모: demo/storage.html

쿠키 대신 Web Storage사용하기

쿠키는 수 년 동안 사용자의 고유 데이터를 추적하는데 사용되어 왔지만 심각한 단점들이 있다. 그 중에도 가장 큰 결함은 모든 쿠키 데이터가 HTTP 요청 헤더에 포함되어 버린다는 점이다. 이는 결국 응답 시간에 나쁜 영향을 미친다. 특히, XHR이 많은 웹 애플리케이션은 더더욱 그렇다. 가장 좋은 사례는 역시 쿠키의 크기를 줄이는 것이지만 HTML5에서는 쿠키를 대체할 수 있는 Web Storage를 사용할 수 있다.

`localStorage`와 `sessionStorage` 이 두개의 웹 저장소 개체는 클라이언트-사이드에 사용자 데이터를 세션이 유지되는 동안 또는 무기한으로 유지하는데 사용 할 수 있다. 또한 개인 자료가 HTTP 요청에 전송되지도 않는다. 만약에 사용자 데이터를 쿠키에 저장하고 있다면 다음과 같이 개선해 보자.

```
// 브라우저의 localStorage 지원여부를 판단
if (('localStorage' in window) && window.localStorage !== null){

  // 개체에 프로퍼티를 할당하는 쉬운 방법을 사용
  localStorage.wishlist = '["Unicorn","Narwhal","Deathbear"]';


} else {

  // 브라우저에서 Web Storage 를 지원하지 않는다면
  // document.cookie 를 이용한다.
  var date = new Date();
  date.setTime(date.getTime()+(365*24*60*60*1000));
  var expires = date.toGMTString();
  var cookiestr = 'wishlist=["Unicorn","Narwhal","Deathbear"];'+
                 ' expires='+expires+'; path=/';
  document.cookie = cookiestr;
}
```

4.2.3 Web SQL Database

지원 브라우저 :

Web Database는 HTML5와 함께 새로 생겨난 것이다. 이제부터 클라이언트 웹 개발자들은 풍부한 쿼리 능력을 가진 웹 애플리케이션을 만들 수 있게 되었다. SQL 쿼리를 별도로 익혀야하는 노고가 뒤따르지만 온라인 또는 오프라인 여부에 상관없이 사용 가능하며, 클라이언트의 저장소에 영구히 보존할 수 있고, 리소스 점유율이 많은 덩치큰 데이터를 체계적으로 관리 할 수 있다.

지금 소개할 예제 코드들은 아주 간단한 할 일 목록 관리 애플리케이션을 만드는 과정을 다룬다.

변수 선언

예제에 사용될 데이터베이스 로직은 아래와 같은 네임스페이스를 사용한다.

```
var html5rocks = {};
html5rocks.webdb = {};
```

비동기와 트랜잭션의 이해

Web Database를 사용하는 대부분의 사례가 비동기 API를 사용한다. 비동기 API는 non-blocking 시스템이다. 그리고 리턴값을 통해서는 데이타를 얻지 못한다. 때문에 정의된 콜백 함수에 데이터를 전달하게 된다.

Web Database는 HTML을 통한 트랜잭션이다. 이것은 외부에서 SQL 문을 실행할 수 없다. 트랜잭션은 두 종류로 구분되는데, 읽고 쓰기위한 트랜잭션(`transaction()`)과 읽기 전용 트랜잭션(`readTransaction()`)이다. 그리고 주의해야 할 점은 데이터를 읽고 쓸 때 전체 데이터베이스가 잠겨버린다는 점이다.

데이터베이스 열기

데이터베이스에 접근하기 전에 먼저 해야할 일은 데이터베이스를 개설하는 것이다. 개설하기 위해서는 데이터베이스의 이름, 버전, 설명 그리고 크기를 정의 한다.

```
html5rocks.webdb.db = null;

html5rocks.webdb.open = function() {
  var dbSize = 5 * 1024 * 1024; // 5MB
  html5rocks.webdb.db = openDatabase('Todo', '1.0', 'todo manager',
dbSize);
}

html5rocks.webdb.onError = function(tx, e) {
  alert('예기치 않은 오류가 발생하였습니다: ' + e.message );
}

html5rocks.webdb.onSuccess = function(tx, r) {
  // 모든 데이터를 다시 그림
  html5rocks.webdb.getAllTodoItems(tx, r);
}
```

테이블 생성하기

"CREATE TABLE SQL" 쿼리문을 transaction 안에 실행하여 테이블을 만들 수 있다.

OnLoad 이벤트가 발생하는 지점에 테이블 생성함수를 정의했다. 테이블이 이미 존재하지 않는 경우에는 테이블이 생성된다. 이 테이블의 이름은 "todo"이고 아래와 같은 3개의 컬럼을 가진다.

ID - 순차적으로 증가하는 ID 컬럼

todo - 아이템의 몸체가 되는 텍스트 컬럼

added_on - 아이템이 만들어진 시간 컬럼

```
html5rocks.webdb.createTable = function() {
  html5rocks.webdb.db.transaction(function(tx) {
    tx.executeSql('CREATE TABLE IF NOT EXISTS ' +
                  'todo(ID INTEGER PRIMARY KEY ASC, todo TEXT, added on
DATETIME)', []);
  });
}
```

테이블에 데이터 추가하기

할 일 목록을 관리하기 위한 테이블이 준비되었다. 이제 테이블에 아이템을 추가하는 중요한 작업을 진행해 보자. `transaction` 내부에서 todo 테이블에 INSERT 쿼리를 수행해야 한다. 이 때 `executeSql`은 다수의 파라미터를 가진다. 그리고 SQL은 이 파라미터의 값을 컬럼에 입력하는 쿼리를 수행한다.

```
html5rocks.webdb.addTodo = function(todoText) {
  html5rocks.webdb.db.transaction(function(tx){
    tx.executeSql('INSERT INTO todo(todo, added on) VALUES (?,?)',
        [todoText, addedOn],
        html5rocks.webdb.onSuccess,
        html5rocks.webdb.onError);
    });
}
```

테이블에서 데이터 선택하기

이제 데이터베이스에 데이터가 존재한다. 이 데이터를 다시 밖으로 꺼내보자. Web Database는 표준 SQLite SELECT 쿼리를 이용하면 된다.

```
html5rocks.webdb.getAllTodoItems = function(renderFunc) {
  html5rocks.webdb.db.transaction(function(tx) {
    tx.executeSql('SELECT * FROM todo', [], renderFunc,
        html5rocks.webdb.onError);
  });
}
```

여기에 사용된 명령 예제는 모두 비동기이다. 이러한 경우 `transaction` 또는 `executeSql` 호출시 데이터가 반환되지 않는다. 데이터는 반드시 콜백을 통해 전달된다는 사실을 기억하자.

가져온 데이터 처리하기

데이터를 성공적으로 가져왔다면 `loadTodoItems` 함수가 호출되게 하자. `onSuccess` 콜백은 두개의 파라미터를 가진다. 첫 번째는 쿼리 트랜잭션이고 두 번째는 결과 묶음이다. 결과 묶음은 배열이며 데이터가 담겨있다.

```
function loadTodoItems(tx, rs) {
  var rowOutput = "";
  for (var i=0; i < rs.rows.length; i++) {
    rowOutput += renderTodo(rs.rows.item(i));
  }
  var todoItems = document.getElementById('todoItems');
  todoItems.innerHTML = rowOutput;
}

function renderTodo(row) {
  return '<li>' + row.ID  +
         '[<a onclick="html5rocks.webdb.deleteTodo(' + row.ID +
');"'>X</a>]</li>';
}
```

그리고 ID가 "todoItems"인 DOM 요소에 할 일 목록들이 그려지는 일을 수행한다.

테이블에서 데이터 제거하기

```
html5rocks.webdb.deleteTodo = function(id) {
  html5rocks.webdb.db.transaction(function(tx) {
    tx.executeSql('DELETE FROM todo WHERE ID=?', [id],
        loadTodoItems, html5rocks.webdb.onError);
  });
}
```

초기화 및 HTML 구성하기

페이지 로드가 완료되면, 데이터베이스를 열고, 테이블을 생성하고(필요한 경우), 데이터를 가져와 할 일 항목이 그려지게 하자.

```
<script>
....
function init() {
  html5rocks.webdb.open();
  html5rocks.webdb.createTable();
  html5rocks.webdb.getAllTodoItems(loadTodoItems);
}
</script>

<body onload="init();">

  <form type="post" onsubmit="addTodo(); return false;">
    <input type="text" id="todo" name="todo"
      placeholder="What do you need to do?" style="width: 200px;" />
    <input type="submit" value="Add Todo Item"/>
  </form>
```

`<input>` 요소로 부터 작성된 값을 가져와 전달하기 위한 함수가 필요하다. `html5rocks.webdb.addTodo` 메서드를 호출할 함수를 만들자.

```
function addTodo() {
  var todo = document.getElementById('todo');

  html5rocks.webdb.addTodo(todo.value);
  todo.value = '';
}
```

데모: http://html5.firejune.com/demo/webdb.html

4.2.4 Web Sockets

지원 브라우저 :

Web Socket은 꾸준한 성장과 인기를 얻고있는 Comet의 대안으로 고안되었다. 이 것은 웹 애플리케이션이 full-duplex 단일 소켓 연결을 가능케 한다. 이는 서버와 브라우저 사이에 진정한 양방향 통신 채널을 제공하는 것

을 의미하며, 연결 관리를 단순화 한다. 하지만 서버에서 Web Sockets 프로토콜을 지원하는 환경에서만 작동하며, 추가적으로 서버에 모듈을 설치하거나 독립적으로 이를 지원하는 서버에서 정상적으로 작동한다.

XHR보다 적은 대역폭을 가진 빠른 송/수신

WebSocket은 매우 가볍게 구성되어 XHR보다 대역폭 소모가 적다. 일부 보고서에 따르면 전송 대역폭의 35%의 절감효과가 발생하는 것으로 조사되었다. 또한, 메시지 전달 비교 실험에서 XHR이 WebSocket보다 3500% 느린 것으로 측정되 상당 수준의 성능 차이가 있는 것으로 밝혀졌다. 끝으로, Ericcson 연구소에서 만든 WebSockets와 HTTP 비교 동영상은 WebSockets보다 HTTP가 ping당 3-5배나 느린것으로 밝혀져 실시간 상호작용이 빈번하게 발생하는 웹 애플리케이션 개발에 사용하기 적합한 것으로 결론 내렸다.

지금부터 서버와 클라이언트간 메시지를 주고 받는 간단한 예제를 살펴보자.

클라이언트-사이드

클라이언트-사이드의 Web Socket은 매우 간단하게 사용할 수 있도록 고안되었다. 다음 코드가 하는 일은 서버의 9876 포트에 접속하고 수신한 데이터를 `alert`으로 출력한다.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Web Socket Example</title>
    <meta charset="UTF-8">
    <script>
      window.onload = function() {
        var s = new WebSocket("ws://localhost:9876/");
        s.onopen = function(e) { alert("opened"); }
        s.onclose = function(e) { alert("closed"); }
        s.onmessage = function(e) { alert("got: " + e.data); }
      };
    </script>
  </head>
    <body>
      <div id="holder" style="width:600px; height:300px"></div>
    </body>
</html>
```

서버-사이드

이제 서버 차례다. 그리고 서버단 언어는 파이선으로 작성되었다. 서버는 1초 간격으로 두개의 메시지를 보낸다. 단순성과 명확성을 위해 서버측 응답은 hard-coding된 것으로 한다. 이를 실제로 구현 한다면 유효성을 검사하고 동적인 응답이 이루어지도록 해야할 것이다.

```
#!/usr/bin/env python

import socket, threading, time

def handle(s):
  print repr(s.recv(4096))
  s.send('''
HTTP/1.1 101 Web Socket Protocol Handshake\r
Upgrade: WebSocket\r
Connection: Upgrade\r
WebSocket-Origin: http://localhost:8888\r
WebSocket-Location: ws://localhost:9876/\r
WebSocket-Protocol: sample
  '''.strip() + '\r\n\r\n')
  time.sleep(1)
  s.send('\x00hello\xff')
  time.sleep(1)
  s.send('\x00world\xff')
  s.close()

s = socket.socket()
s.setsockopt(socket.SOL SOCKET, socket.SO REUSEADDR, 1)
s.bind(('', 9876));
s.listen(1);
while 1:
  t,  = s.accept();
  threading.Thread(target = handle, args = (t,)).start()
To run the above, start the Web Socket server (./server.py) and start
a web server on port 8888 serving index.html:

./server.py &
python -m SimpleHTTPServer 8888
```

실행해 보면 "hello"와 "world"라는 메시지가 1초간격으로 수신되는 모습을 확인할 수 있다.

4.2.5 Web Workers

지원 브라우저 :

Web Worker는 두가지 중요한 장점을 가지고 있다. 첫 번째는 빠르다는 것이고, 두 번째는 브라우저에 부담을 주지않고 백그라운드에서 스크립트

연산을 수행하는 것이다. 이것이 가능한 이유는 브라우저가 OS-래벨의 스레드를 생성하기 때문이며, 동시 다발적으로 사용하는 경우 더욱 흥미로운 결과를 기대할 수 있다.

이제부터 Web Worker 기본적인 사용법에 대하여 알아보자.

Worker 생성하기

Worker를 생성하는 것은 간단하다. 백그라운드에서 작업할 스크립트를 별도의 파일에 작성하고 새로운 인스턴스에 URI를 기입하여 생성한다. 그리고 onmessage 속성에 함수를 대입하여 작업결과를 돌려 받을 수 있다.

```
var myWorker = new Worker('my worker.js');
myWorker.onmessage = function(event) {
  alert("Worker에 의해 실행된 콜백!\n");
};
```

Worker 종료하기

실행중인 Worker를 즉시 종료하려면 terminate() 메서드를 호출하여 즉시 종료할 수 있다. 이러한 경우, Worker는 남은 작업을 마무리하거나 메모리에서 찌꺼기를 청소한 후 자발적으로 사라진다.

```
myWorker.terminate();
```

백그라운드에서 피보나치 수열 계산하기

다음 예제는 Worker를 이용하여 피보나치 수열을 계산하는데 사용된다. 이것은 사용자 인터페이스의 스레드를 차단하지 않고 프로세서 집약적인 계산을 수행 할 수 있도록 하는 것이다.

다음은 "fibonacci.js"에 저장된 내용이다.

```
var results = [];

function resultReceiver(event) {
  results.push(parseInt(event.data));
  if (results.length == 2) {
    postMessage(results[0] + results[1]);
  }
}

function errorReceiver(event) {
  throw event.data;
}
```

```
onmessage = function(event) {
  var n = parseInt(event.data);

  if (n == 0 || n == 1) {
    postMessage(n);
    return;
  }

  for (var i = 1; i <= 2; i++) {
    var worker = new Worker("fibonacci.js");
    worker.onmessage = resultReceiver;
    worker.onerror = errorReceiver;
    worker.postMessage(n - i);
  }
};
```

onmessage 함수는 postMessage()를 호출한다. 이렇게 하므로써 반복적인 계산의 새로운 복사본을 만들어 수행하게 된다.

```
<!DOCTYPE html>
<html>
  <title>Test threads fibonacci</title>
  <body>

  <div id="result"></div>

  <script language="javascript">

    var worker = new Worker("fibonacci.js");

    worker.onmessage = function(event) {
      document.getElementById("result").textContent = event.data;
      console.log("Got: " + event.data + "\n");
    };

    worker.onerror = function(error) {
      console.log("Worker error: " + error.message + "\n");
      throw error;
    };

    worker.postMessage("5");

  </script>
  </body>
</html>
```

id가 "result"인 <div> 요소에 그 결과가 표시되며 worker.postMessage에 의해 Worker에 작업을 지시할 수 있다.

데모: http://html5.firejune.com/demo/worker.html

CPU 부하를 줄이기 위한 Web Worker를 적용할 상황들

스크립트를 이용하여 무거운 연산을 실행하면 브라우저는 먹통(응답 없음) 상태가 된다. 이러한 경우 이벤트 리스너가 제대로 작동하지 않아 오작동이 발생하거나 제때 콜백이 호출되지 않거나 짧은 시간동안 상호작용이 발생하는 프로그램 로직에 치명적인 오류를 안겨줄 수 있다.

이러한 상황은 Web Worker를 이용하여 우회할 수 있다는 사실을 기억하자.

- 긴 문서의 문자 서식 지정
- 문법 강조 기능
- 이미치 프로세싱
- 이미지 합성
- 덩치큰 배열 처리

4.2.6 Server-Sent Event

지원 브라우저 :

Server-sent Event는 일종의 푸시 테크놀로지이다. 이것은 브라우저가 서버로부터 지속적으로 데이터를 스트림하는 상태가 되는 것을 말한다. 서버에서 클라이언트로 전달할 이벤트가 발생한 경우 즉시 전달하여 사용자에게 알릴 수 있게 된다.

다음 예제는 서버로부터 이벤트를 스트림하는 방법을 다룬다. 현재 브라우저별로 사용법이 조금씩 다르기 때문에 웹킷 계열 브라우저를 중심으로 설명한다.

스크립트 작성하기

EventSource에 이벤트를 스트림 받을 URL을 입력한다.

```
var source = new EventSource('event.php');
source.onmessage = function (event) {
  alert(event.data);
};
```

event.php 작성하기

서버단 언어는 PHP이며, 서버의 시간을 스트리밍하도록 작성한 것이다.

이 때 mime-type은 "text/event-stream"이어야 한다.

```
<?php
header("Content-Type: text/event-stream");
    echo "data: " . time() . "\n";
?>
```

"\n"은 라인 변경을 의미한다. 수신 받은 데이터는 아래와 같다.

```
data: 1277717394\n
```

실행해 보면 `alert`에 "1277717394" 문자가 출력될 것이다. 그리고 이 과정은 계속 반복된다. 때문에 서버는 long-poll 형식으로 응답해 주는 것이 효과적이다.

오페라 브라우저 역시 이 기능을 지원하지만 `<event-source>` 요소에 "src"와 이벤트를 할당하는 방법으로 사용해야 하며, mime-type은 "application/x-dom-event-stream"이어야 하고 반드시 수신 데이터에 "Event: server-time"과 같은 이벤트를 명시해야 한다.

데모: http://html5.firejune.com/demo/sse.html

4.3 리치 웹 API

4.3.1 Selector API

지원 브라우저 :

HTML5에는 새로운 [Selector API](#)인 `querySelector`, `querySelectorAll` 메서드가 추가되었다. 이 메서드들을 이용하여 DOM으로 부터 요소를 빠르고 쉽게 찾아낼 수 있다. 기존 자바스크립트 라이브러리(Prototype, jQuery 등)들이 지원하던 DOM Selector의 네이티브 구현이라 할 수 있겠다.

`querySelector` 메서드는 인자로 받은 선택 조건을 DOM 트리로부터 검색하여 첫 번째 일치하는 요소 노드를 반환한다. 그리고 노드가 발견되지 않으면 null을 반환한다.

`querySelectorAll` 메서드는 인자로 받은 선택 조건을 DOM 트리로부터 검색하여 일치하는 모든 요소 노드를 반환한다. 일치되는 노드가 없

는 경우, 비어있는 목록을 반환한다.

그리고 우리가 그토록 바라던 getElementByClassName도 추가적으로 사용할 수 있게 되었다.

Selector API 사용법

두 메서드 모두 인자로 전달되는 검색 조건에는 우리가 일반적으로 많이 사용하는 CSS 선택 문법을 그대로 사용할 수 있으며, 쉼표(',')로 구분하여 하나 이상의 검색 조건을 추가할 수 있다.

```
// 클래스 이름이 'warning', 또는 'note'인 단락 요소(<p>)를 모두 찾음
var special = document.querySelectorAll("p.warning, p.note");

// id가 'main', 'basic', 'exclamation'인 요소들 중 첫 번째 발견된 요소를
찾음
var el = document.querySelector("#main, #basic, #exclamation");

// HTML 문서의 <body>에 속한 <style> 요소들 중
// 'type' 속성이 없거나, 'text/css'인 첫 번째 발견된 요소를 찾음
var style = document.body.querySelector("style[type='text/css'],
style:not([type])");

// id가 'fruits'인 요소의  <input> 요소(체크박스)들 중 선택된(checked) 요소를
찾음
var list = document.querySelectorAll("#fruits input:checked");
// 또는
var list =
document.getElementByID('fruits').querySelectorAll("input:checked");
```

4.3.2 Drag and Drop

지원 브라우저 :

Drag and Drop API가 없던 시절에도 "mousemove", "mousedown", "mouseup" 이벤트를 이용하여 요소를 특정한 요소에 끌어다 놓는 수준은 구현할 수 있었다. 그러나 잡다한 뒤처리를 해야 했기 때문에 자바스크립트 라이브러리를 추가적으로 이용해야 했고 이벤트 이상 증식현상이나 CPU 부하로 인한 오작동이 빈번하게 발생하여 널리 사용되고 있지는 않았다.

HTML5에서 새롭게 지원하기 시작한 Drag and Drop API는 더욱 향상된 끌어다 놓기 경험을 제공한다. 특히, File API를 함께 이용하면, 바탕화면 혹은 탐색기의 파일을 브라우저로 직접 끌어나 놓는 방식으로도 파일을 업로드 할 수 있게 되었다.

이 예제는 로컬에 위치한 파일을 특정한 HTML 요소에 끌어다 놓고 해당 파일을 직접 액세스하고 미리보기를 보여주는 예제이다.

드랍 영역 마크업하기

아이템을 드래그할 수 있는 영역과 미리볼 수 있는 이미지를 등록한다.

```
<div id="dropbox">
  <span id="droplabel">
    이곳에 파일을 드랍해 주세요...
  </span>
</div>
<img id="preview" alt="[ preview will display here ]" />
```

드랍 영역 이벤트 등록하기

그리고 아래와 같이 이벤트를 할당한다.

```
var dropbox = document.getElementById("dropbox")

// 이벤트 핸들러 할당
dropbox.addEventListener("dragenter", dragEnter, false);
dropbox.addEventListener("dragexit", dragExit, false);
dropbox.addEventListener("dragover", dragOver, false);
dropbox.addEventListener("drop", drop, false);
```

위 코드는 얼핏 보면 복잡해 보이지만 `dragEnter`, `dragExit`, `dragOver` 핸들러는 아래와 같은 이벤트의 이상 증식현상을 중지시키는 역할을 할 뿐이다.

```
event.stopPropagation();
event.preventDefault();
```

drop 이벤트 헨들러 작성하기

반환된 이벤트로 부터 `dataTransfer.files` 개체로 접근한 후 파일이 1개 이상 존재하면 `handleFiles` 함수를 호출한다.

```
event.stopPropagation();
event.preventDefault();

var files = event.dataTransfer.files;
var count = files.length;

// 오직 한개 이상의 파일이 드랍된 경우에만 처리기를 호출한다.
if (count > 0)
  handleFiles(files);
```

handleFiles 함수 작성하기

전달 받은 개체로 부터 파일들 선택하고, 파일 이름을 표시하고, FileReader(File API) 인스턴스를 생성하여 파일을 처리한다.

```
var file = files[0];

document.getElementById("droplabel").innerHTML = "Processing " +
file.name;

var reader = new FileReader();

// 파일 리더의 이베트 핸들러 정의
reader.onloadend = handleReaderLoadEnd;

// 파일을 읽는 작업 시작
reader.readAsDataURL(file);
```

readAsDataURL 메서드는 파일을 data URL 형식으로 만들어 준다. 이는 파일을 서버에 업로드하지 않고도 조작할 수 있음을 의미한다. 포멧을 변환하거나, 데이터를 분석하여 변조하는 일이 가능해 진다. 예를 들면, 이미지의 특정한 영역을 클라이언트-사이드에서 크롭한 후 서버에 업로드하는 것이 가능하다. 보다 자세한 내용은 '4.3.5 File API'에서 다루도록 한다.

handleReaderLoadEnd 함수 작성하기

handleReaderLoadEnd 함수의 내용은 아주 간단하다. 미리보기할 이미지 요소에 소스를 대입한다.

```
var img = document.getElementById("preview");
img.src = event.target.result;
```

데모: http://html5.firejune.com/demo/dnd.html

4.3.3 Geolocation

지원 브라우저 :

Geolocation API는 브라우저가 사용자의 지리적 위치를 찾아내고 그 정보를 애플리케이션에서 이용할 수 있도록 하는 기능이다. 사용자의 위치 정보를 이용하기 위해서는 먼저 승인 절차를 거쳐야 하며, 승인이 완료 된 상태라면 사용자 컨텐츠가 생성될 때 지오-태깅(geo-tagging)기능을 제공할 수 있고 근처에서 촬영된 사진 등에 대한 정보를 유기적으로 연결시켜 서비스

할 수 있다.

그리고 사용자의 위치가 변경 될 때 마다 콜백 메서드로 전달되어 항상 최신의 위치 정보를 유지하는 것이 가능하다. 이러한 지리 정보는 기본적으로 GPS 장치로 부터 얻어지는 것이 가장 정확하지만 그외 지리 정보를 얻을 수 있는 수단들을 단계적으로 이용하여 최소한 수도 또는 국가 단위의 지리 정보를 취득할 수 있다.

다음 예제는 이동 거리 측정기를 만드는 과정을 소개한다. 이 예제를 통해 Geolocation API의 자세한 사용방법을 알아보자.

브라우저 호환성 확인

geolocation 개체의 존재를 확인하는 방법으로 브라우저가 이를 지원하는지의 여부를 쉽게 확인 할 수 있다.

```
// check for Geolocation support
if (navigator.geolocation) {
  console.log('Geolocation 을 지원합니다.');
}
else {
  console.log('이 브라우저또는 OS 는 Geolocation 을 지원하지 않습니다.');
}
```

측정기의 HTML 마크업

아래는 이동 거리 측정기를 구성하는 HTML을 마크업 한 것이다.

```
<div id="tripmeter">
  <p>
    시작 위치 (위도, 경도):<br/>
    <span id="startLat"></span>°, <span id="startLon"></span>°
  </p>
  <p>
    현재 위치  (위도, 경도):<br/>
    <span id="currentLat"></span>°, <span id="currentLon"></span>°
  </p>
  <p>
    시작 위치로 부터의 거리:<br/>
    <span id="distance">0</span> km
  </p>
</div>
```

사용자의 현재 위치 확인

getCurrentPosition() 메서드를 이용하여 사용자의 현재 위치를 찾아낼 수 있다. 이 것은 페이지 로드가 완료되는 시점에 실행된다.

```
window.onload = function() {
  var startPos;
  navigator.geolocation.getCurrentPosition(function(position) {
    startPos = position;
    document.getElementById('startLat').innerHTML =
startPos.coords.latitude;
    document.getElementById('startLon').innerHTML =
startPos.coords.longitude;
  });
};
```

만약, 처음으로 위와 같은 과정이 발생하는 경우 브라우저는 사용자에게 위치 정보 사용을 허용할지에 대한 여부를 확인한다. 브라우저에 따라서는 환경설정에서 항상 허용 또는 거부할 수 있는 기능을 제공하기도 하기 때문에 이 과정이 무시될 수도 있다.

위 코드를 실행해 보자. 반환되는 `position` 개체에서 시작 위치의 좌표를 확인할 수 있어야 한다. `position` 개체는 위도(latitude)와 경도(longitude) 외에도 많은 정보가 포함되어 있는데, 사용자의 지리정보를 수신하는 기기 환경에 따라서는 고도(altitude)와 방향(direction) 정보까지 얻어낼 수 있다. `console.log`를 이용하여 이 값들의 포함 여부를 살펴보자.

오류 처리

불행하게도 위치 조회를 성공하지 못하는 경우가 발생한다. 어쩌면 갑자기 GPS를 찾을 수 없거나 위치 정보 사용 권한을 박탈당한 경우일 것이다. `getCurrentPosition()` 메서드의 두 번째 인자로 넘긴 콜백은 이러한 오류가 발생한 경우 호출되어 사용자에게 이 사실을 전달할 수 있다.

```
window.onload = function() {
  var startPos;
  navigator.geolocation.getCurrentPosition(function(position) {
    // 상기와 동일
  }, function(error) {
    alert('오류 발생. 오류 코드: ' + error.code);
    // error.code는 다음을 의미함:
    //   0: 알 수 없는 오류
    //   1: 권한 거부
    //   2: 위치를 사용할 수 없음 (이 오류는 위치 정보 공급자가 응답)
    //   3: 시간 초과
  });
};
```

사용자 위치 모니터링

getCurrentPosition()의 호출은 페이지가 로드되고 난 다음 딱 한 번만 하면 된다. 이후의 위치 변동사항을 추적하려면 watchPosition()을 사용한다. 이것은 사용자의 위치변동이 감지될 때마다 인자로 받은 콜백을 호출한다.

```
navigator.geolocation.watchPosition(function(position) {
  document.getElementById('currentLat').innerHTML =
position.coords.latitude;
  document.getElementById('currentLon').innerHTML =
position.coords.longitude;
});
```

이동한 거리 측정

이 예제는 Geolocation API와 직접적인 관계는 없다. 하지만 당신이 얻어낸 위치 데이터를 조금 더 구체적으로 이용할 수 있는 방법에 대하여 제시한다.

```
navigator.geolocation.watchPosition(function(position) {
  // 상기와 동일
  document.getElementById('distance').innerHTML =
      calculateDistance(startPos.coords.latitude,
startPos.coords.longitude,
                        position.coords.latitude,
position.coords.longitude);
});
```

지금부터 작성하게 될 calculateDistance() 함수는 두 좌표 사이의 거리를 확인하는 기하학적 알고리즘을 수행한다. 아래의 스크립트 코드는 Moveable Type에서 제공하는 것으로 Creative Commons 라이센스에 따라 이용할 수 있다.

```
function calculateDistance(lat1, lon1, lat2, lon2) {
  var R = 6371; // km
  var dLat = (lat2 - lat1).toRad();
  var dLon = (lon2 - lon1).toRad();
  var a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
          Math.cos(lat1.toRad()) * Math.cos(lat2.toRad()) *
          Math.sin(dLon / 2) * Math.sin(dLon / 2);
  var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
  var d = R * c;
  return d;
}
Number.prototype.toRad = function() {
  return this * Math.PI / 180;
}
```

이 이동 거리 측정기 예제는 페이지가 로드된 시점으로 부터 일정한 거리를 이동해 봐야 작동여부를 확인할 수 있다. 실질적으로 GPS가 장착된 최신 스마트폰에서 무난히 테스트 할 수 있을 것이다.

데모: http://html5.firejune.com/demo/geolocation.html

4.3.4 IndexedDB

지원 브라우저 :

IndexedDB(Indexed Database API)는 구조적 데이터 저장소이다. 이는 Web Storage나 Web Database와 마찬가지로 온/오프라인 상태에서 사용할 수 있는 클라이언트-사이드 데이터베이스이다. IndexedDB는 최초 Oracle에서 처음 제안된 것으로 사용 측면에서는 Web Database와 상당부분 겹친다. Web Database에 비하여 IndexedDB는 스크립트를 이용한 데이터베이스를 다루기에 최적화된 인터페이스를 제공하며, 알고리즘 방식의 입/출력을 지원하고 비동기/동기 처리 모두 API차원에서 제공하고 있으며 커서까지 지원하여 관계형 데이터베이스(RDB)형식으로 데이터 구조를 설계할 수 있다.

indexedDB 사용 예

```
var db = indexedDB.open('books', 'Book store', false);
if (db.version !== '1.0') {
  var olddb = indexedDB.open('books', 'Book store');
  olddb.createObjectStore('books', 'isbn');
  olddb.createIndex('BookAuthor', 'books', 'author', false);
  olddb.setVersion("1.0");
}
// db.version === "1.0";
var index = db.openIndex('BookAuthor');
var matching = index.get('fred');
if (matching)
  report(matching.isbn, matching.name, matching.author);
else
  report(null);
```

4.3.5 Notifications

지원 브라우저 :

구글 크롬은 새로운 방식의 독자적 알림 수단인 Notifications API를 제시했다. 이 것은 사용자에게 특정한 상황의 메시지를 전달하여 즉시 알릴 수 있는 기능으로 소프트웨어 업데이트 알림, 메신저의 메시지 도착 또는 친구

접속 알림 기능을 연상하면 이해하기 쉽다. 윈도 혹은 맥 OS의 작업 표시줄 근처에 풍선말이 출력되면서 이벤트 발생 사실을 문자로 알리는 것과 매우 흡사하기 때문이다. 그래서 새 이메일이 도착하거나 트위터에 답변이 작성되거나 혹은 캘린더의 일정과 같은 알림성 컨텐츠를 Server-Sent Event로 부터 받아 처리하기에 적합하다. 특히, 브라우저의 탭 또는 윈도의 활성화 여부에 상관없이 작동하기 때문에 페이지로의 접근을 용이하게 한다. 현재 Notifications는 아직 기초 단계에 있는 사양이며 표준안으로 채택될지는 불분명하다.

Notifications API 지원여부 확인

Notifications API가 지원되는 브라우저인지 확인하는 방법은 다음과 같다.

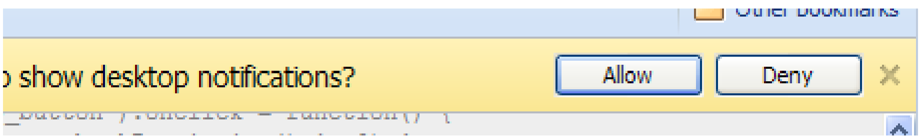
```
// notifications 를 지원하는지 확인
// 'window' 키워드는 생략 가능
if (window.webkitNotifications) {
  console.log("Notifications are supported!");
}
else {
  console.log("이 브라우저 또는 OS 는 Notifications 기능을 지원하지 않습니다.");
}
```

알림 인스턴스 생성하기

알림에 들어가는 컨텐츠는 일반 텍스트 또는 HTML 형식으로 작성할 수 있다. 이것을 옵션화 하여 두 종류 모두 사용할 수 있게 했다.

```
function createNotificationInstance(options) {
  if (options.notificationType == 'simple') {
    return window.webkitNotifications.createNotification(
        'icon.png', 'Notification Title', 'Notification content...');
  } else if (options.notificationType == 'html') {
    return
window.webkitNotifications.createHTMLNotification('http://someurl.com'
);
  }
}
```

사용자 승인여부 구분하여 출력하기



이 기능은 사용자의 승인을 얻어야만 작동하는 기능이다. 최신 브라우저들은 팝업이 기본으로 막혀있는데, 이를 사용자가 승인해 주어야지만 작동하는 모습과 비슷하다. 만약, 사용자가 이를 승인하지 않았다면 오류가 발생한다. 물론 try-catch를 사용할 수도 있지만 `checkPermission` 메서드를 사용하여 승인여부를 구분할 수 있다. 그리고 `ondisplay` 또는 `onclose` 프로퍼티에 상황에 따른 콜백함수를 정의할 수도 있다.

```
document.getElementById('show button').addEventListener('click',
function() {
  // 값이 "0"이면 승인을 얻어낸 것이다.
  if (window.webkitNotifications.checkPermission() == 0) {
    // 다음 단계에서 함수를 정의한다.
    notification test = createNotificationInstance({ notificationType:
'html' });
    notification test.ondisplay = function() { ... do
something ... };
    notification test.onclose = function() { ... do something
else ... };
    notification test.show();
  } else {
    window.webkitNotifications.requestPermission();
  }
}, false);
```

데모: http://html5.firejune.com/demo/notification.html

4.3.6 File API

지원 브라우저 :

File API는 웹 애플리케이션이 로컬 파일에 프로그램적으로 접근할 수 있게한다. 파일을 업로드하기 위해 웹 사이트의 특정 영역으로 파일을 드래그하거나 `<input>` 요소로부터 전달받은 파일에 접근하여 파일의 이름, 경로, 크기, 종류 등에 대한 정보를 취득할 수 있다. 물론, 읽기전용 상태로 접근된 것이기 때문에 실제 파일의 물리적 변형은 일어나지 않는다. 그러나 4.3.2 Drag and Drop에서 언급했듯이 `<canvas>`요소를 응용하면 파일을 서버에 업로드하지 않고도 포멧을 변환하거나, 변조한 후 서버로 업로드 하는 일이 가능하다. 또한 `FileReader`개체를 이용하면 바이너리 데이터를 분석하여 JPEG 파일의 EXIF 정보, MP3의 ID3 태그를 가져오는 등의 작업을 수행할 수 있다.

다음에 소개할 예제는 `<input>` 요소에 선택된 파일의 정보를 분석하는 과정을 설명할 것이다.

HTML 구성하기

<input type="file"> 요소는 일반적으로 단 하나의 파일만을 선택할 수 있지만 "multiple" 속성을 사용하면 여러개 파일을 선택할 수 있게 된다. 그러나 "multiple" 속성을 지원하지 않는 브라우저들이 있으므로 호환성을 위해서는 플래시의 File I/O와 연동하는 방법이 사용되기도 한다.

```
<h3>파일(들)을 선택하세요.</h3>
<!-- multiple 속성을 이용하면 파일을 다중으로 업로드 할 수 있음 -->
<input id="files-upload" type="file" multiple>

<h3>Uploaded files</h3>
<ul id="file-list">
  <li class="no-items">(파일이 선택되 않음)</li>
</ul>
```

스크립트 구성하기

<input> 요소에 변동사항이 발생하면 traverseFiles함수가 호출되며, 이 함수는 파일의 정보를 <ul> 요소에 출력할 것이다.

```
var filesUpload = document.getElementById("files-upload"),
    fileList = document.getElementById("file-list");

function traverseFiles (files) {
  var li,
      file,
      fileInfo;
  fileList.innerHTML = "";

  for (var i=0, il=files.length; i<il; i++) {
    li = document.createElement("li");
    file = files[i];
    fileInfo = "<div><strong>Name:</strong> " + file.name + "</div>";
    fileInfo += "<div><strong>Size:</strong> " + file.size + "
bytes</div>";
    fileInfo += "<div><strong>Type:</strong> " + file.type +
"</div>";
    li.innerHTML = fileInfo;
    fileList.appendChild(li);
  };
};

filesUpload.onchange = function () {
  traverseFiles(this.files);
};
```

데모: http://html5.firejune.com/demo/file.html

4.3.7 WebGL

지원 브라우저 :

WebGL은 브라우저가 플러그인의 도움을 받지 않고 3D 웹 그래픽을 표현하기 위한 OepnGL ES 2.0의 자바스크립트 바인딩이다. 이것은 하드웨어 가속이 되는 실시간 3D 그래픽을 표현하는 것이 가능하다. AMD, 구글, 모질라, 엔비디아 등의 굵직한 벤더들이 워킹 그룹에 참여하고 있다. 현재 WebGL은 개발 버전의 브라우저에서만 사용할 수 있으며, <canvas> 요소 위에 그려지도록 설계 되었다.

지금부터 하나의 정육면체를 생성하고 간단한 애니메이션 효과를 부여하는 예제를 작성할 것이다. 3차원 공간에서 정육면체를 생성하는 과정을 학습해 보자.

정육면체의 버텍스 위치 정의

첫 번째로 정육면체의 버텍스(꼭지점) 위치를 정의해야 한다. 아래와 같이 총 24개의 버텍스 위치의 배열을 만든다.

```
var vertices = [
    // 전면
    -1.0, -1.0,  1.0,
     1.0, -1.0,  1.0,
     1.0,  1.0,  1.0,
    -1.0,  1.0,  1.0,

    // 후면
    -1.0, -1.0, -1.0,
    -1.0,  1.0, -1.0,
     1.0,  1.0, -1.0,
     1.0, -1.0, -1.0,

    // 윗면
    -1.0,  1.0, -1.0,
    -1.0,  1.0,  1.0,
     1.0,  1.0,  1.0,
     1.0,  1.0, -1.0,

    // 아랫면
    -1.0, -1.0, -1.0,
     1.0, -1.0, -1.0,
     1.0, -1.0,  1.0,
    -1.0, -1.0,  1.0,

    // 우측면
     1.0, -1.0, -1.0,
     1.0,  1.0, -1.0,
```

```
    1.0,  1.0,  1.0,
    1.0, -1.0,  1.0,

   // 좌측면
   -1.0, -1.0, -1.0,
   -1.0, -1.0,  1.0,
   -1.0,  1.0,  1.0,
   -1.0,  1.0, -1.0
  ];
```

버텍스의 색상 정의

생성된 24개의 버텍스에 색상을 정의한다. 아래의 코드는 정육면체의 각 면단위로 색상을 정의할 것이다.

```
var colors = [
    [1.0,  1.0,  1.0,  1.0],    // 전면: 하양
    [1.0,  0.0,  0.0,  1.0],    // 후면: 빨강
    [0.0,  1.0,  0.0,  1.0],    // 윗면: 초록
    [0.0,  0.0,  1.0,  1.0],    // 아랫면: 파랑
    [1.0,  1.0,  0.0,  1.0],    // 우측면: 노랑
    [1.0,  0.0,  1.0,  1.0]     // 좌측면: 보라
  ];

  var generatedColors = [];

  for (j=0; j<6; j++) {
    var c = colors[j];

    for (var i=0; i<4; i++) {
      generatedColors = generatedColors.concat(c);
    }
  }

  cubeVerticesColorBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY BUFFER, cubeVerticesColorBuffer);
  gl.bufferData(gl.ARRAY BUFFER, new WebGLFloatArray(generatedColors),
gl.STATIC_DRAW);
```

요소 배열 정의

앞서 버텍스 배열을 생성했다. 이제 구성 요소들을 구축해야 할 단계이다.

```
cubeVerticesIndexBuffer = gl.createBuffer();
  gl.bindBuffer(gl.ELEMENT ARRAY BUFFER, cubeVerticesIndexBuffer);

  // 이 배열은 두개의 삼각형으로 하나의 면을 정의한다.
  // 각 삼각형의 버텍스 위치를 정의하여 순차적으로 지정한다.

  var cubeVertexIndices = [
    0,  1,  2,      0,  2,  3,    // 전
    4,  5,  6,      4,  6,  7,    // 후
    8,  9,  10,     8,  10, 11,   // 위
    12, 13, 14,     12, 14, 15,   // 아래
```

```
  16, 17, 18,     16, 18, 19,   // 우측
  20, 21, 22,     20, 22, 23    // 좌측
]

// 이제 요소의 배열을 GL에 보낸다.

gl.bufferData(gl.ELEMENT_ARRAY_BUFFER,
    new WebGLUnsignedShortArray(cubeVertexIndices), gl.STATIC_DRAW);
```

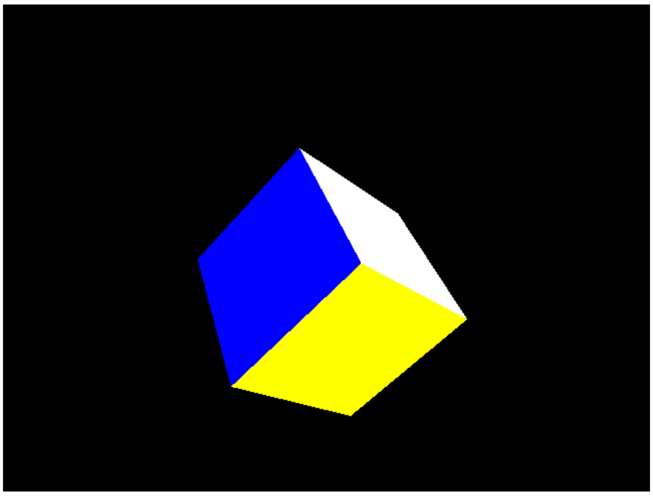
cubeVertexIndices 배열은 한 쌍이 삼각형으로 구성되어 있고 삼각형 두개로 하나의 면을 형성하는 구조로 정의한다. 삼각형은 세개의 버텍스를 순차적으로 지정한다. 결국 하나의 정육면체는 12개의 삼각형의 집합인 것이다.

정육면체 그리기

bindBuffer()을 추가하고 drawElements()메서드를 호출한다.

```
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, cubeVerticesIndexBuffer);
  setMatrixUniforms();
  gl.drawElements(gl.TRIANGLES, 36, gl.UNSIGNED_SHORT, 0);
```

한 면당 2개의 삼각형, 6개의 버텍스로 구성된다. 즉 정육면체는 총 36개의 버텍스로 이루어지며, 대부분 중복되는 위치에 있다. 그리고 이 버텍스들의 배열은 간단한 정수로 구성되어 있다.



데모: demo/webgl.html

4.4 FAQ

자바스크립트 API의 브라우저 지원은 언제쯤?

크로스-브라우저 이슈가 사라지는 그날이 과연 올 것인가? 대답은 '아니오'이다. 이 이슈는 지금까지도 줄곧 이어져 왔고 앞으로도 계속될 것이다. 왜냐하면 브라우저를 만드는 집단이 한 곳이 아니기 때문이다. 물론 웹 표준화 단체에서 권고사항을 문서화하고 표준이라는 단어를 앞세워 노력하고는 있지만 새로운 웹 기술을 제시하는 쪽은 대부분 브라우저 개발업체이거나 관련 업계여서 선/후발 주자가 생기고 또한 같은 기능을 만들어도 자신들만의 특징을 집어넣고 싶어하기 마련이다.

결국, 표준화 단체로 부터 기술을 인증받고 공식화 된 다음에야 비로소 모든 브라우저 개발사들이 새로운 기술을 구현하는 이 과정은 지속적으로 반복될 것이다. 그래서, 이러한 정책적인 문제는 웹 개발자들이 고민할 일도 아니고 한다고 해서 해결되지도 않는다. 일찌감치 포기하고 실현 가능성만을 바라보는 것이 훨씬 현실적이다. 예를 들어 Web Sockets 라는 멋진 API가 생겨났지만 일부 브라우저에서는 지원되지 않기 때문에 사용할 수 없다고 판단해버릴 문제가 아니라는 것이다. COMET이라는 대안이 있잖은가.

브라우저의 알파버전은 어디에서 다운로드하나?

대부분의 브라우저 개발사는 개발하는 과정에서 공개 테스트를 거친 후 안정적인 버전을 내 놓는다. 우리는 이 과정에서 배포된 버전을 베타 버전이라고 한다. 최근 들어서는 그 이전 내부 개발 단계에서 빌드된 버전들까지도 배포하는 풍토가 생겨났으며, 이 버전을 알파 또는 나이틀리 버전이라 한다.

알파버전은 곧 탑재될 새로운 기능 단위 테스트가 이루어지는 경우가 많아서 개발자들이 새로운 API를 미리 테스트하여 향후 계획을 설계하는데 사용되곤 한다. 여기에 소개한 내용의 일부 역시 알파버전에서만 작동하는 것이 있다. 브라우저별 알파버전을 다운로드하는 주소는 다음과 같다:

- IE9 - ie.microsoft.com/testdrive/
- Firefox: ftp.mozilla.org/pub/mozilla.org/firefox/nightly/latest-trunk/
- Chrome(Chromium): build.chromium.org/buildbot/continuous/
- Safari: nightly.webkit.org/

클라이언트-사이드의 개발분야가 너무 광범위하지 않나?

광범위해진 것이 사실이다. 웹 프론트-엔드 개발자가 생각치도 못했던 SQL, OpenGL을 다루게 생겼고 심지어 로컬 파일의 바이너리까지 분석할 수 있게 되었다. 이러한 멋진 기능들 중에 무엇부터 공부해야 할지, 어떠한 애플리케이션을 만들수 있는지, 정신을 못차릴 정도의 행복한 고민에 빠져들게 한다.

정말 눈부신 발전 속도다. 이 상태라면 다음 세대 HTML은 과연 어떤 모습으로 다가올지 두렵기까지 하다. 그렇다고 너무 걱정할 필요는 없지 않을까? 프로젝트를 혼자서 구축하지 않는 이상은 여럿이 협업하여 분담할 수 있고, 그러면서 각 분야의 전문가가 차차 나타나게 될 것이 분명하다.

오프라인을 고려하는 이유는 무엇인가?

HTML5에 새롭게 추가된 API들 중 상당부분이 오프라인을 염두하고 있음을 눈여겨 보지 않아도 알 수 있다. 데스크탑 컴퓨터에서 24시간 인터넷을 무제한으로 사용할 수 있는 환경이라면 선뜻 이해할 수 없는 부분이기도 하다.

그러나 긍정적으로 생각하면 기존 웹이 가진 특성상 서버나 브라우저 모두 자원 소모가 심한 비효율적인 통신 로직을 보완하여 성능 향상을 꾀할 수 있고, 상대적으로 통신 요금이 비싼 모바일 인터넷 사용자 또는 인터넷 인프라가 발전하지 못한 국가에는 패킷을 절약하여 요금 부담을 줄일 수도 있으며, 구글이 강력하게 밀고있는 웹 O/S기반에서 오프라인의 의미는 곧 애플리케이션을 설치한 개념으로 통할 수 있게 된다. 이정도만으로도 오프라인을 고려하려는 의지를 설명하기에 충분치 않겠는가.

향후 웹 개발 풍토는 어떻게 변하나?

프론트-엔드 개발자의 시각으로 볼 때 HTML5의 등장으로 두드러진 업무적 변화는 바로 스크립트로 하는 뒤치닥거리가 많이 줄었다는 점이다. CSS3의 등장으로 마우스 행동을 동반한 애니메이션을 스크립트 없이 만들어 낼 수 있고, Web Forms 2.0을 이용하면 유효성 검사 및 오류 메시지 출력, 인풋의 레이블 바꿔치기, 포커싱 등을 스크립트의 도움을 받지 않고 HTML 마크업만으로 작성할 수 있게 된 것이다.

그러나 새로운 스크립트 API들이 대거 추가되면서 서버에 의지하여 처리하던 작업들을 스크립트만으로 처리할 수 있게 되었다. 예를 들면, 프로파일 이미지의 크롭 및 리사이즈, 서버에서 생성한 데이터 차트 이미지 등의 리소스를 많이 잡아먹는 작업은 이제 클라이언트-사이드에서도 가능하다.

그리고 Server-Sent Event, Web Sockets 등의 출현으로 서버-사이드에는 새로운 개념의 통신 프로토콜이 마련되어야 하며, 실시간 상호작용을 위해 잦은 양방향 통신이 발생하는 웹 애플리케이션 개발에 어울리는 설계를 모색하게 될 것이다.

참고 자료

- www.whatwg.org/html5/ - HTML5 Working Draft
- dev.w3.org/html5/ - W3C HTML5
- html5doctor.com/ - HTML5 Doctor
- diveintohtml5.org/ - DIVE INTO HTML 5
- html5demos.com - HTML5 Demos and Examples
- www.html5rocks.com - HTML5 guides
- developer.mozilla.org/en/HTML/HTML5 - MDC
- www.canvasdemos.com/ - HTML5 Canvas Demos
- tutorials.jenkov.com/svg/index.html - SVG Tutorial
- srufaculty.sru.edu/david.dailey/svg/SVGAnimations.htm- SVG Animation

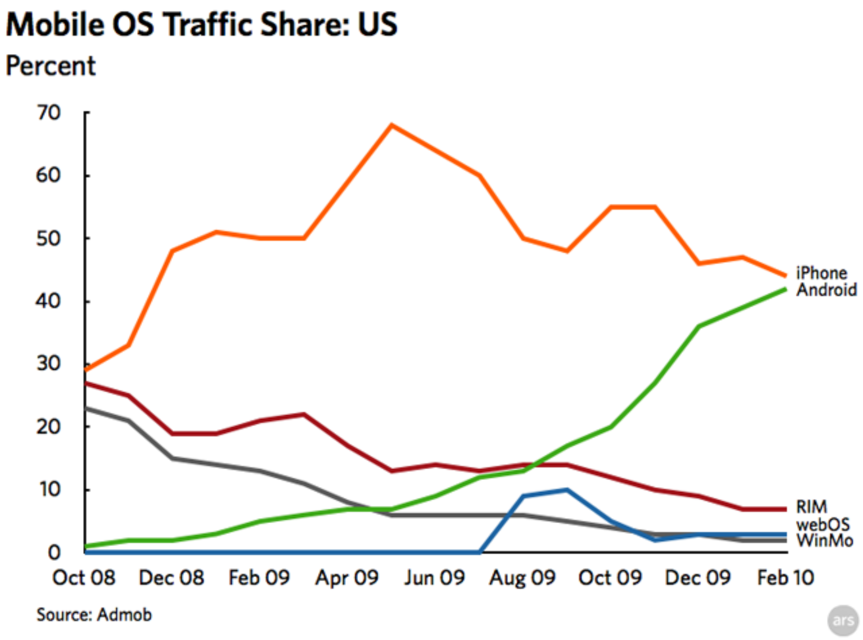
5. HTML5와 모바일

본 장에서는 HTML5에 대한 가장 최적의 적용 환경이라 할 수 있는 모바일 환경에서의 상황에 대해 알아보고, 아이폰에서의 HTML5 웹 어플리케이션 개발방법을 예제와 함께 살펴본다.

5.1 모바일에서의 HTML5

HTML5 를 이용한 많은 Web Application(웹 어플리케이션, 웹 앱)들이 만들어 지고 있지만, 데스크탑 환경에선 아직 이슈가 남아있다. 현재 브라우저 점유율에서 가장 큰 위치를 차지하고 있는 Internet Explorer (6.0 ~ 8.0) 이 아직 HTML5 지원이 미진하기 때문이다. 이 때문에 HTML5 웹 어플리케이션들은 Firefox, Chrome, Safari 같은 타 브라우저를 통해서만 확인이 가능하다.

하지만 모바일 환경으로 오면 얘기가 달라진다. 현재 모바일 웹 트래픽의 대부분을 차지하고 있는 것이 iPhone, Android 이고 이 두개의 플랫폼이 HTML5 를 잘 지원하고 있는 WebKit 기반의 브라우저를 사용하기 때문이다.



http://metrics.admob.com/2010/03/february-2010-mobile-metrics-report/

5.1.3 모바일 브라우저들의 HTML5 대응

현재 모바일 폰에서 HTML5 를 가장 잘 지원하고 있는 것은 아이폰 OS 4.0 (iOS4) 에 포함된 Mobile Safari 이다. Apple 은 HTML5 에 대한 지원을 공공연히 드러내기 전부터 Mobile Safari 에 조용히 HTML5

의 스펙을 지원하기 시작했다.

아래는 http://html5test.com 에서 제공하는 HTML5 테스트 결과점수로 살펴본 모바일 / 데스크탑 브라우저들의 HTML5 지원현황이다. (160점 만점으로 되어있으며, 현재는 좀더 항목을 세분화하여 300점 만점으로 개편되었다. 완벽한 기능테스트가 아니기 때문에 브라우저 별로 개체만 만들고 기능은 구현 안 한 경우도 있을 수 있다.)

OS or Browser	Version	Score (? / 160)
IE (Win)	6.0	11
IE (Win)	8.0.7600	19
Opera Mini	1.0	33
iPhone (Mobile Safari)	2.0	37
Android	1.6	39
iPhone (Mobile Safari)	2.1 - 2.2	45
Maemo microB	5 PR-1.1.1	55
Firefox Mobile	1.0	101
Firefox (Win)	3.6.3	101
Palm WebOS	1.4	107
iPhone (Mobile Safari)	3.0	110
iPhone (Mobile Safari)	3.1	113
Safari (Mac)	4.0.5	113
iPad (Mobile Safari)	3.2	115
Android	2.0 - 2.1	118
Android	2.2	122
iPhone (Mobile Safari)	4.0 Beta 4	133
Safari (Mac)	5.0	138
Chrome (Win , Mac)	6.0.422.0	142

현재로선 Android 2.2 와 iOS 4.0 의 브라우저가 거의 모든 HTML5 스

펙을 다 지원하기 때문에, HTML5 를 이용한 어플리케이션 작성에 있어선 모바일이 좀 더 나은 환경이라고 볼 수 있다.

5.1.2 모바일 웹 에서의 HTML5 Key Elements

모바일에서 HTML5 가 특별히 다른 태그를 활용하는 것은 아니다. 다만 API 중 몇 개가 모바일에 더욱 쓰기 좋은 형태일 뿐이다.

- Offline 지원 : LocalStorage , Web Database , App Cache
- 미디어 처리 : Video , Audio , Canvas
- 입력 지원 : Advanced Forms
- 위치 정보 : GeoLocation (연계표준)

Offline 지원의 경우 항상 인터넷에 연결되어 있는 데스크탑과 달리 모바일 환경은 꼭 3G 와 같은 네트웍에 항시 연결되어있지 않은 WIFI 전용기기(iPod Touch , iPad) 들도 있으며, 3G 환경이라 할지라도 네트웍 트래픽을 최소화 하는 것이 아주 중요하다.

또한 HTML5 의 중요 스펙중 몇 가지는 아직 주요 브라우저에서도 지원되지 않는다,

- WebSocket
- FileReader
- IndexedDB
- Web Workers

현재로선 모바일에서의 HTML5 사용은 주로 Offline 지원을 통한 Local App 으로서의 동작 및 트래픽 최적화, Geo Location 을 통한 위치 정보 연동이 가장 많이 쓰이고 있다.

5.1.3 Web App vs. Native App

Web App (웹앱)이라고 하면, 웹 기술을 이용하여 만들어진 어플리케이션을 말한다. 즉, 컨텐츠 리딩을 위해 사용되던 단방향성이던 웹사이트와 달리 사용자와의 인터랙션을 통하여 데스크탑에서의 Application 같은 사용성을 주는 앱을 말한다. 위젯이라는 단어도 종종 사용되는데 웹 앱은 이 위젯을 포함한 좀 더 넓은 범위로 보는게 맞다.

HTML5 를 이용해서 만들어진 Web App 과 iPhone / Android 환경

에서의 Native App 을 비교해보자.

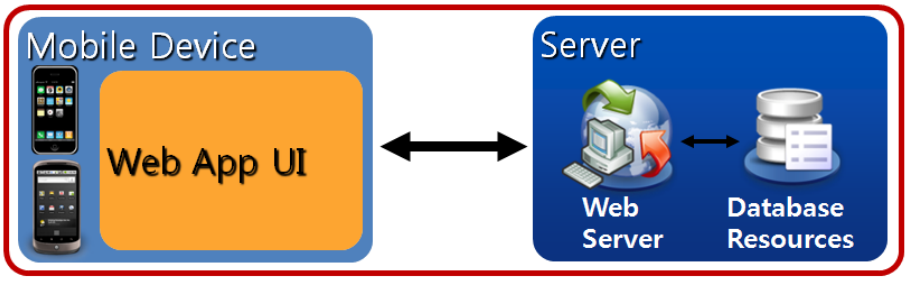
Web App	Native App
모바일 디바이스에 최적화된 웹사이트	모바일 디바이스 전용 앱
HTML , CSS , Javascript	Objective-C (iPhone) , Java (Android)
기존에 사용하던 웹 개발환경	XCode (iPhone) , Eclipse (Android)
웹 표준 컨트롤 , iUI , JQTouch ..	Cocoa Touch (iPhone), UI Framework (Android)
꼭 Mac 이 필요하지는 않음	Mac 이 필요 (iPhone) Android 는 멀티플랫폼 (Win , Mac , Linux)
App 개발자 등록 필요없음 1)	개발자등록 년 $99 (iPhone) or $35 (Android)
제한적인 디바이스 사용 - 카메라/마이크.. 2)	디바이스의 모든 기능을 활용
자체 결제시스템 구축필요 또는 광고	App Store/Market를 통한 판매/수익 & 광고
서버에서 바로바로 업데이트가능	업그레이드 할 때 마다 검수 (iPhone)
Android / Blackberry등으로도 바로 변환가능	실행속도가 빠르다

- 1) PhoneGap 등 Hybrid App Framework 사용하여 앱 만든후 등록시 필요함
- 2) honegap , QuickConnect 를 통하여 카메라/연락처 정보등 사용가능. 따로 진행중인 W3C DAP , OMTP BONDI , JIL 등의 스펙으로 디바이스 접근가능(차후)

5.1.4 Mobile Web App 의 종류

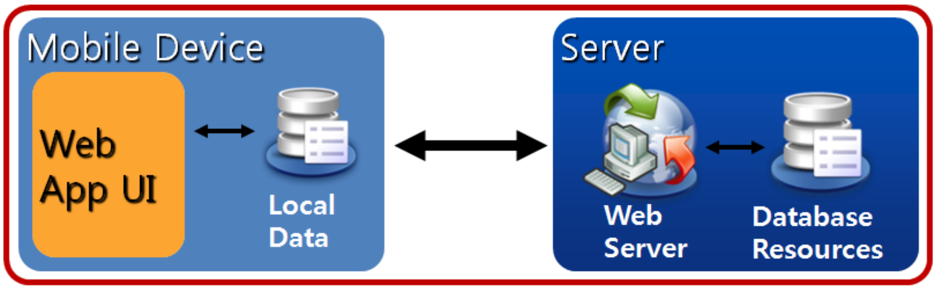
Web App 이라고 해서 특별한 것은 아니지만, 모바일 관점에서 볼 때 다음과 같이 나눠 볼수 있다.

Online Web Application



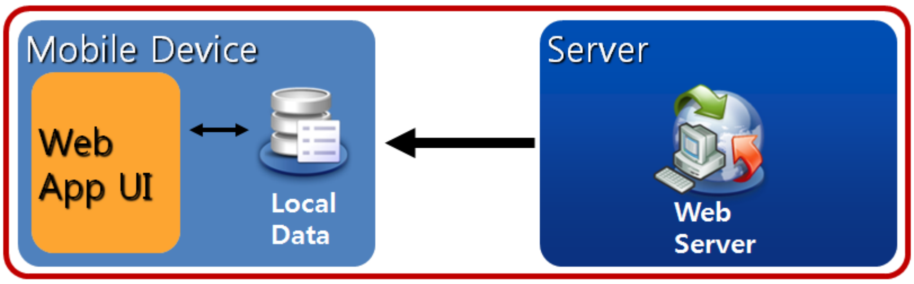
- 기존의 모바일 웹페이지를 포함한 웹 앱을 의미한다.
- HTML5, CSS3 등을 통하여 모바일 에서의 UI / UX 를 향상한다.
- GeoLocation API 를 이용하여 위치가 고정되지 않는 모바일의 장점을 활용한다.
- 주로 포털이나 컨텐츠를 리딩하는 형태의 앱이 많다.

Offline Enabled Web Application



- Application Cache , LocalStorage , Web SQL Database 를 활용하여 오프라인에서도 사용이 가능한 웹 앱을 의미한다.
- 처음 접속시에 주요데이타를 캐쉬하여 재 접속시에 네트웍 트래픽을 최소화한다.
- 오프라인상태에서 행한 동작들에 대해서 온라인시 서버의 데이터와 싱크한다.
- 이메일 어플리케이션과 같은 컨텐츠 리딩 & 작성앱에 적절하다.
- 모바일 Gmail 사이트는 이미 완벽한 오프라인 메일 앱으로 동작한다.

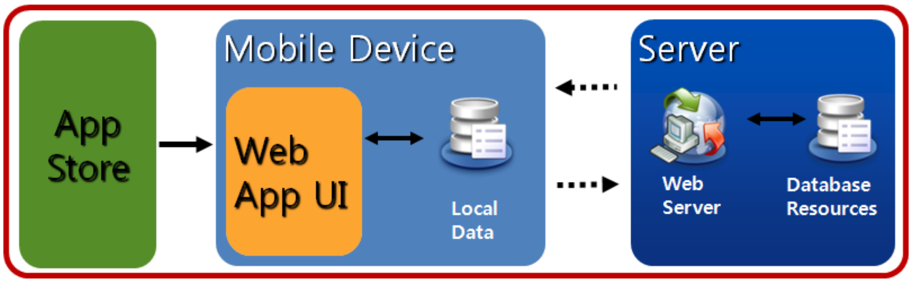
Offline Web Application



- 한번 서버에 접속해서 다운받으면 계속 오프라인으로 사용 가능한 앱.
- Application Cache 와 Web SQL Database 를 활용

- 서버와의 동기화를 필요로 하지 않는 게임, 유틸리티, EBook

Hybrid Web Application



- Native App 과 Web App을 합친 형태의 앱.
- 앱 스토어를 통한 다운로드 가능.
- 모바일 디바이스로 다운로드후에는 앱 속성에 따라 서버와의 AJAX 통신도 가능
- Native 수준의 다양한 앱 작성가능

모바일 웹 앱이 이렇게 4가지로 명확하게 분리되는 것은 아니지만, 이를 통해 모바일에서 어떤 종류의 웹 앱이 만들어 질 수 있는지를 알 수 있다.

5.2 아이폰 기반 HTML5 앱 개발

앞서 언급한대로 현재 Web App 을 가장 잘 지원하는 것은 아이폰이다. 이는 아이폰OS의 1.0 버전때에는 App Store 가 없었기에 애플측에서 Web App 의 사용을 권장했었기 때문이고, 아이폰 OS 2.0 버전부터 App Store 가 런칭되면서 사용용도가 많이 줄기는 했지만, 아직도 애플 웹사이트의 Web App Directory (http://www.apple.com/webapps/)에는 약 5천개 정도의 Web App 들이 등록되어 있다. 아이폰상에서 어떻게 HTML5 Web App 을 만드는지를 알아보자.

5.2.1 iPhone 환경에서의 Web App 지원

먼저, iPhone 환경에서는 데스크탑과 달리 사이트에 접근하는 방식이 차이가 있기 때문에 특이한 사항이 몇 개 있다. HTML5 지원과는 조금 다른 얘기이지만, HTML5 를 이용한 Web Application 을 만든다면 알아둬야 것 중의 하나다.

아이폰 사파리는 웹 사이트에 대해 "홈 화면에 추가 (Add to Home Screen)" 이라는 기능을 제공한다. 사파리 브라우저에서 맨 아래 + 버튼을 누르면 "책갈피추가 / 홈 화면에 추가" 선택팝업이 뜨고, 이를 통해 아

이폰 메인화면에 웹 사이트에 대한 바로가기 기능을 추가 할 수 있다. (안드로이드 에서도 비슷한 기능을 제공하긴 하지만 단계가 조금 복잡하다)

이렇게 + 를 눌러 "홈 화면에 추가"후 실행하면 iPhone 에서는 Web App 으로 동작하게 된다. 이때 Web App 의 지원을 위해 모바일 사파리는 몇 가지 태그를 지원한다.

<head> 섹션에 아래의 4가지 태그를 추가할 수 있다.

```
<link rel="apple-touch-icon" href="/apple-touch-icon.png"/>
<link rel="apple-touch-startup-image" href="/startup.png">
<meta name="apple-mobile-web-app-capable" content="yes" />
<meta name="apple-mobile-web-app-status-bar-style" content="black" />
```

각각은 다음과 같은 기능을 지원한다.

```
<link rel="apple-touch-icon" href="/apple-touch-icon.png"/>
```

등록되는 웹 사이트의 아이콘을 지정할 수 있다. apple-touch-icon.png 가 기본 이름이며 일반적으로 웹사이트 아이콘을 추가하게 되면 웹사이트 화면을 캡쳐한 내용을 아이콘으로 사용하는데 apple-touch-icon 이라는 링크를 추가하여 아이콘을 내가 지정한 것으로 사용할 수 있다. favicon 의 아이폰 버전이라고 생각하면 된다.

iPhone 은 57x57 , iPad 는 72 x 72 사이즈의 png 이미지를 사용한다.

이 아이콘은 기본적으로 아이폰이 제공하는 UI 처리 (모서리를 둥글게 하고 반원형의 밝은 부분을 추가해 주는 것) 가 된다. 원하지 않을 때는 파일 이름을 apple-touch-icon-precomposed.png 라는 이름으로 저장하여 사용하면 된다.

```
<link rel="apple-touch-icon" href="/apple-touch-icon-
precomposed.png"/>
```

➤ 이 아이콘기능은 안드로이드 에서도 지원된다. 단 precomposed 만 사용가능 (48x48)

```
<link rel="apple-touch-startup-image" href="/startup.png">
```

화면이 로딩될 때 스타트업 이미지를 지정할 수 있다. Web App 이지만 앱 처음 로딩시 로고화면 같은걸 보여줄 수 있다. 아이폰 기본 앱에 들어있는 Default.png 와 비슷한 역할이다.

```
<meta name="apple-mobile-web-app-capable" content="yes" />
```

Web App으로 선언하여 브라우저의 UI (URL 바) 를 안 보이도록 할 수 있다. 즉, Web App 이 마치 일반 Native App 처럼 화면 전체 (최상단 상태바 20px 제외) 를 활용할 수 있도록 한다.

```
<meta name="apple-mobile-web-app-status-bar-style" content="black" />
```

상태바의 색상을 지정할수 있다. 바탕화면이 검정색인 어플리케이션의 경우 상태바만 회색인 이질감을 줄이기 위해 사용한다. 3가지 스타일 : default (회색) , black , black-translucent (반투명)

이렇게 4가지 기능을 지원함으로써 HTML5 기반의 Local 또는 Online Web Application 들이 마치 Native App 처럼 실행하는 효과를 지원할 수 있다.

5.2.2 아이폰 개발 환경 꾸미기

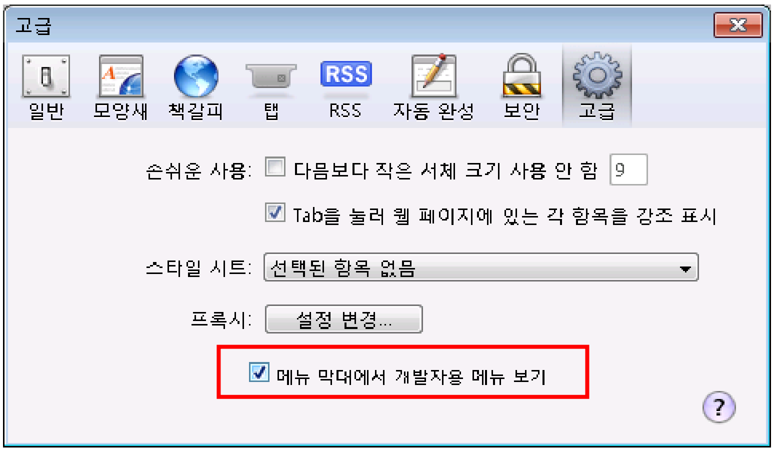
Web App 은 아이폰 Native App 과 달리 개발환경으로 꼭 Mac을 필요로 하지 않는다. 기존의 웹 개발 할때와 마찬가지로 손에 익은 툴을 이용하여 개발할 수 있다. 아이폰 상에서의 동작 화면을 테스트하기 위해서는 XCode 에 포함된 iPhone Simulator 가 테스트용으로 아주 적절하지만, Mac 사용자가 아니라면 사용이 어렵다. 또한 Web App 개발시 iPhone

Simulator는 디버깅을 하는 용도로는 적합하지 않다.

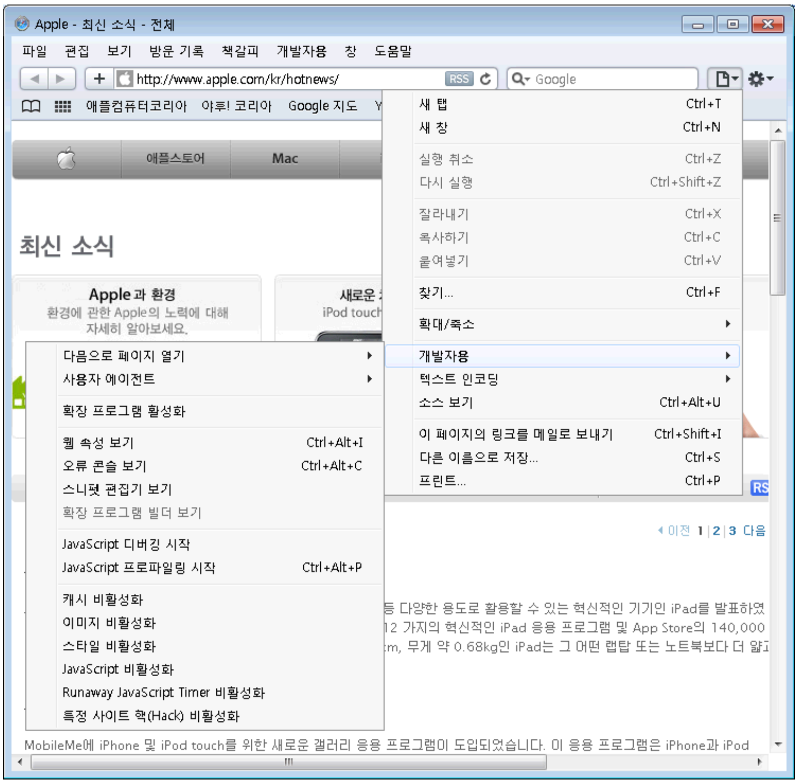
Safari 의 개발자 도구 활용하기

Webkit 기반의 브라우저 최신버전들에는 개발자 도구가 들어있다. 이를 활용해 보자.

Safari 의 메뉴에서 편집->기본설정 또는 Ctrl + , 를 눌러서 설정창을 연다. 우측 끝 고급탭을 선택하여 맨 밑에 "메뉴 막대에서 개발자용 메뉴 보기" 를 활성화 한다.



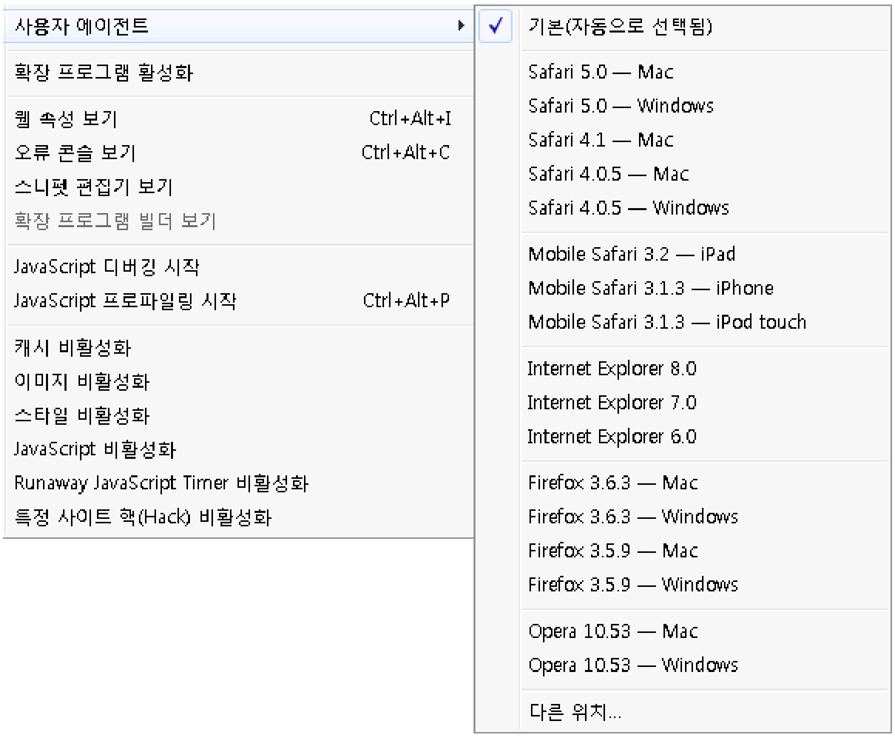
활성화 하면 아래와 같이 개발자용 메뉴가 추가된다. 이 개발자 도구는 같은 WebKit 기반인 Chrome 브라우저에도 비슷한 기능이 있지만, Safari 가 더 많은 기능을 가지고 있다.



개발자용 메뉴중 주요 기능을 몇 개 살펴보자.

사용자 에이전트

만들고 있는 웹 사이트를 iPhone / iPad 및 각 브라우저별로 어떻게 표시되는지를 테스트해보기 위해 UserAgent 를 쉽게 바꾸는 기능을 제공한다.



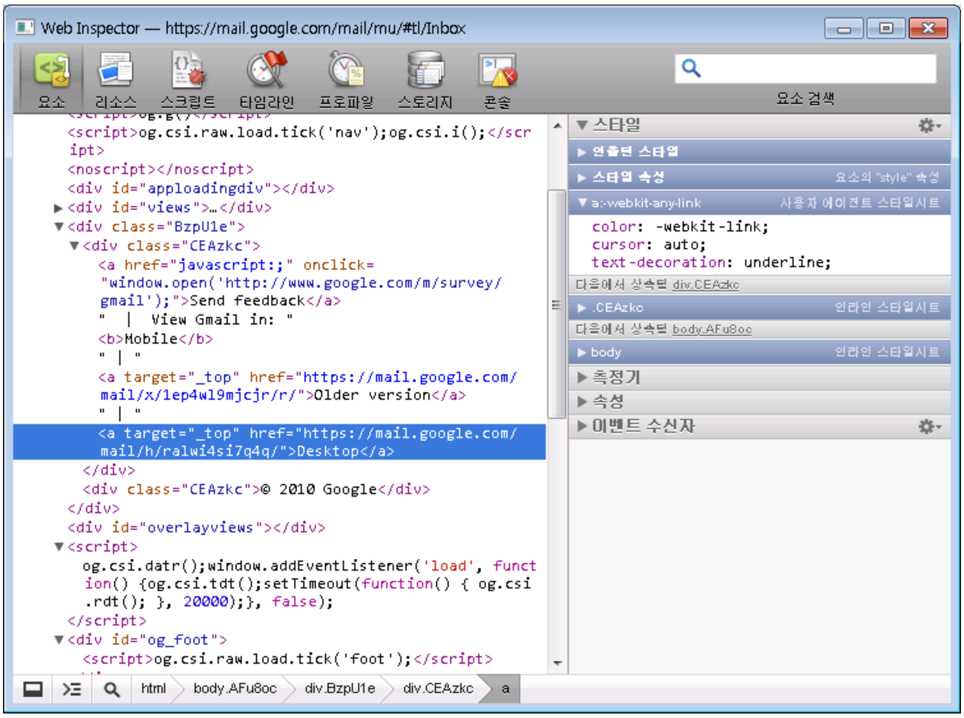
Chrome 이나 Firefox 에서는 확장기능을 설치해야 가능했던 동작인데 기본으로 포함하고 있다. Mobile Safari 로 설정시 거의 아이폰 화면과 비슷한 화면을 보여주기 때문에 아이폰용 Web App 개발후 테스트시 아주 유용하다.

아래는 사파리 브라우저 사이즈를 아이폰과 비슷하게 맞춘후, UserAgent 를 iPhone 으로 설정하고, 아이폰용 UI 라이브러리인 jQTouch 데모 사이트를 불러온 화면 이다.

거의 비슷한 형태의 화면을 볼 수 있으며, Webkit 애니메이션들도 그대로 볼 수 있다. 아이폰용 Web App 을 만든다면 꼭 알고 있어야 할 필수 테스트 방법이다.

웹 속성보기 – Web Inspector (Ctrl - Alt - I , □ - □ - I)

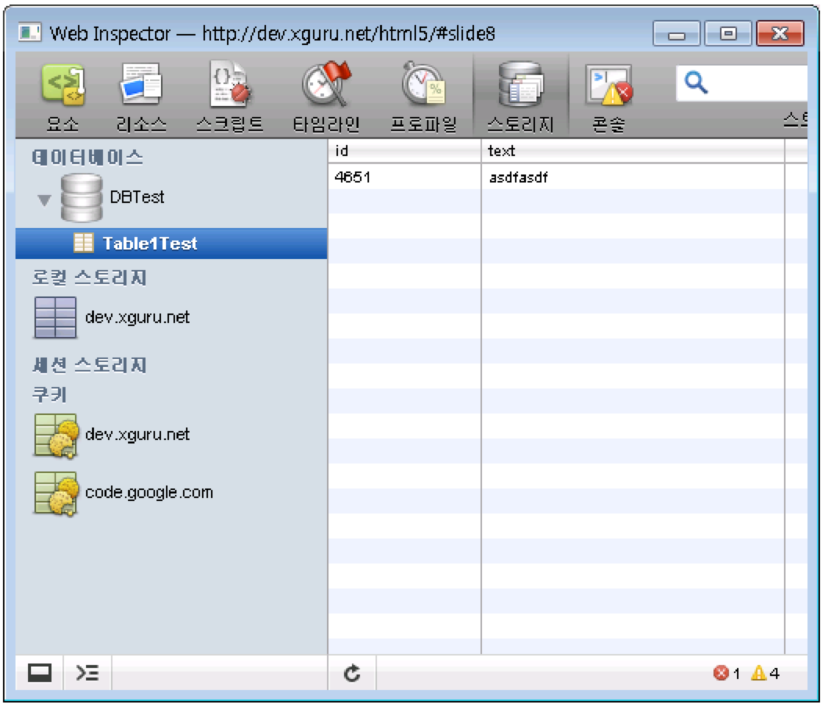
현재 웹 페이지의 상세 속성을 보는 창이다.



➢ 요소 : 현재 페이지의 HTML 소스를 보여준다.

➢ 리소스 : 현재 페이지에서 로딩한 리소스 파일들 (CSS, JS , 이미지 등) 및 각 리소스의 로딩에 걸린시간을 보여준다.
➢ 스크립트 : 현재 페이지내의 Javascript 소스를 확인하고 디버깅 할 수 있다.
➢ 타임라인 : 리소스 로드/스크립트수행/렌더링 시간들의 타임라인을 볼 수 있다.
➢ 프로파일 : 현재 페이지내의 스크립트가 CPU 자원을 얼마나 소모하는지 프로파일링
➢ 스토리지 : Database , Local/Session Storage , Cookie 들을 일목요연하게 볼 수 있다.
➢ 콘솔 : 에러 확인 및 자바스크립트 실행이 가능한 커맨드 라인 창

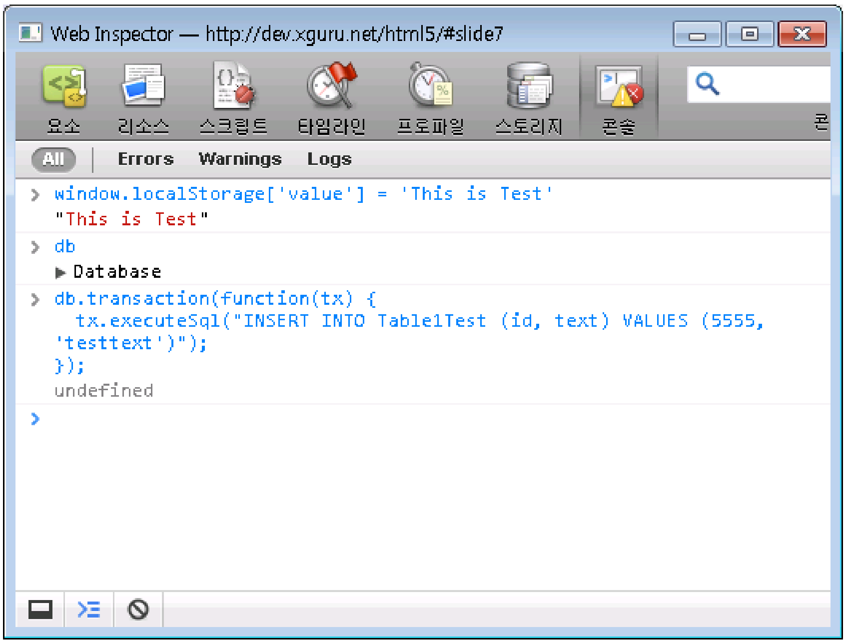
마치 Firefox 에서의 Firebug 플러그인을 보는 것처럼 잘 만들어져 있다. 다른 부분은 기존의 툴들과 비슷하니 HTML5 에 관련된 스토리지 메뉴만 살펴보도록 하자.



➢ 스토리지 메뉴의 좌측엔 데이터베이스 / 로컬 스토리지 / 세션 스토리지 / 쿠키 로 나뉘어져 있으며 현재 사이트에 대해 저장된 값들을 확인할 수 있다.
➢ 데이터 베이스 : HTML5 API중 Web SQL Database를 활용하여 작성된 Database / Table 을 확인할 수 있다. 각 테이블에 들어있는 컬럼 및 데이터를 볼수있지만 편집은 불가능하다.
➢ 로컬/세션 스토리지 : HTML5 API중 LocalStorage/SessionStorage 를 통해 저장된 Key/Value 쌍들을 확인할 수 있다. 직접 추가/수정/삭제도 가능하다.
➢ 쿠키 : 쿠키의 이름/값/도메인/만료일자 등 을 확인할 수 있다. 삭제는 가능하지만 편집은 불가능하다.

Database 와 Storage 의 내용들을 콘솔창을 이용하여 아래와 같이 직접

컨트롤이 가능하다.



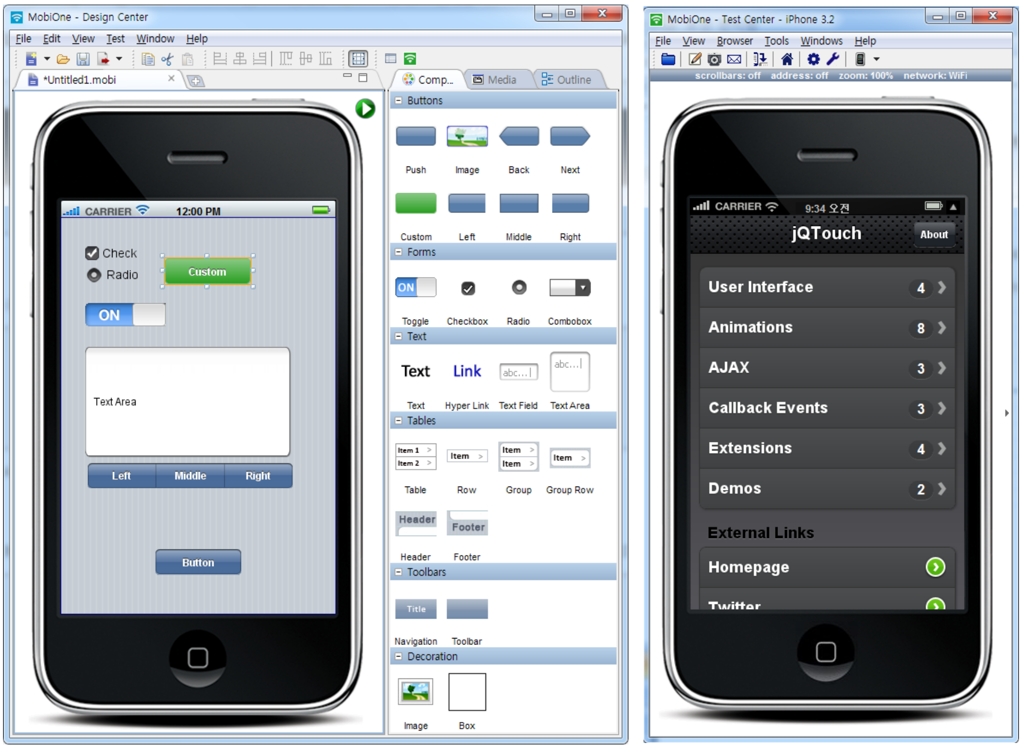
Windows용 iPhone Simulator – MobiOne

또 다른 방법으로 Genuitec 이라는 회사에서 제공하는 MobiOne이라는 프로그램을 이용하는 것이다. MobiOne 은 아이폰상에서의 Web App 을 디자인하기 위한 Design Center 와 테스트를 위한 Test Center 두가지 어플리케이션을 제공한다. (다운로드: http://www.genuitec.com/mobile/)

Design Center	Test Center
Visual Designer (WYSIWYG layout) 30+ UI components Design templates 60+ free iPhone app icons Code generation (currently HTML5) Test on Test Center iPhone emulator Test on your iPhone via cloud service	**iPhone emulator** Palm Pre emulator Multi-touch, gestures, orientation **PhoneGap API** Geolocation Live edit & update of local & remote resources Inspector: DOM, local data, JavaScript, CSS

Design Center

Design 센터에서는 GUI Drag & Drop 방식으로 iPhone Web App 의 UI 를 생성할 수 있게 해준다.



이렇게 작성된 UI 는 HTML5 형태로 CSS/JS/HTML 파일로 저장되어 바로 Web App 작성에 이용할 수 있다.

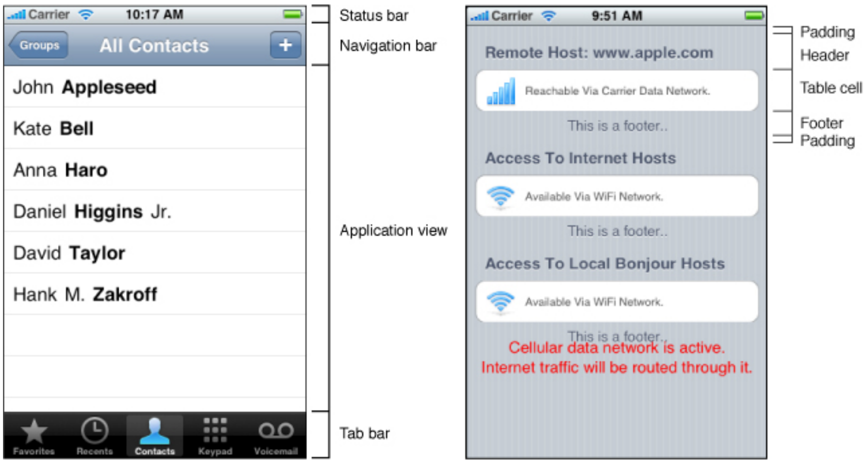
Test Center

Test Center 는 Design Center로 만든 UI 를 테스트하는 아이폰 에뮬레이터이다. URL 입력이 가능하기 때문에 사파리와 비슷하게 아이폰 웹앱 테스터로 유용하다.

5.2.3 웹 기반 아이폰 스타일 UI 만들기

아이폰은 터치기반의 CocoaTouch 프레임워크를 제공하여 터치에 최적화된 UI 컨트롤셋을 제공한다. 주로 사용하는 네비게이션바/탭바/테이블뷰 같은 것이 있는데, 이런 UI 를 Web App 에서도 제공하기 위한 다양한 UI 라이브러리/프레임워크 들이 출시되어 있다

JQTouch - http://jqtouch.com/



jQuery 를 사용하며, 터치기반의 UI 컨트롤을 제공하는 라이브러리. Native WebKit 애니메이션 효과를 지원하며, 성능이 좋아서 가장 많이 쓰인다.

jQTouch 개발자가 Ext JS & Raphael 개발자와 같이 Sencha Touch 라이브러리로 발전중이다. Sencha Touch 는 아이폰 만이 아닌 아이패드/안드로이드/데스크탑 까지 동시 지원을 목적으로 하는 HTML5 기반의 UI 프레임워크 이다.

iUI - http://code.google.com/p/iui/

아이폰 출시 초기부터 지원되었던 라이브러리. 주로 네비게이션/테이블뷰의 사용에 초점이 맞춰져 있다.

WebApp.Net - http://webapp-net.com/

최근에 출시된 라이브러리로 jQTouch 와 iUI 중간정도의 기능을 제공한다. 현재 가장 많이 사용되는 것은 jQTouch 이므로 간단히 사용방법을 알아보자.

jQTouch 설치 및 초기화

jQTouch 최신버전을 http://jqtouch.com 에서 다운받는다. 초기 index.html 파일에 아래 코드를 삽입하여 jQTouch 라이브러리를 인클루드 한다.

```
<head>
...
<style type="text/css" media="screen"> @import "jqtouch.min.css"; </style>
<style type="text/css" media="screen"> @import "themes/apple/theme.min.css";
</style>
<script src="jquery.1.3.2.min.js" type="text/javascript"> </script>
<script src="jqtouch.min.js" type="text/javascript"> </script>
<script type="text/javascript" > $.jQTouch();</script>
…
</head>
```

jQuery 에 기반하고 있기 때문에 먼저 jQuery 를 인클루드 해야한다. jQTouch의 초기화 함수는 $.jQTouch() 이다. 아래와 같이 초기화 옵션을 줄 수 있다.

```
$.jqTouch({
   icon: "path/to/apple-touch-icon.png",            // 홈 스크린 추가시
아이콘으로 지정
   startupScreen: "path/to/startup-image.png",      // 시작 스크린
   statusBar: "default|black|black-translucent",   // 상태바 의 색상
   addGlossToIcon: true|false,                      // 아이콘이
Precomposed 인지 설정
```

앞에서 살펴봤던 아이폰에서의 Web App 지원을 위한 <meta> 태그를 초기화 옵션으로 지정한다.

jQTouch 를 이용한 테이블 뷰 만들기

아이폰 UI중 데이터 네비게이션을 위한 테이블뷰는 div 와 ul 을 이용하여 만들수 있다. 아래 에제중 class 를 눈여겨 보기 바란다. toolbar 클래스는 상단의 툴바를, rounded 는 각 아이템의 테두리를 둥근 아이템으로 설정한다.

ul 클래스는 5가지의 스타일을 제공한다.

- rounded
- edgetoedge
- plastic
- metal
- form

```
<body>
<div id="home">
<div class="toolbar"><h1>My app</h1></div>
<ul class="rounded">
<li><a href="#foo">Foo</a></li>
<li><a href="#bar">Bar</a></li>
```

jQTouch는 각 화면간의 변환시 총 8가지의 Native Webkit Animation을 지원한다.

slide, slideup, dissolve, fade, flip, pop, swap, cube

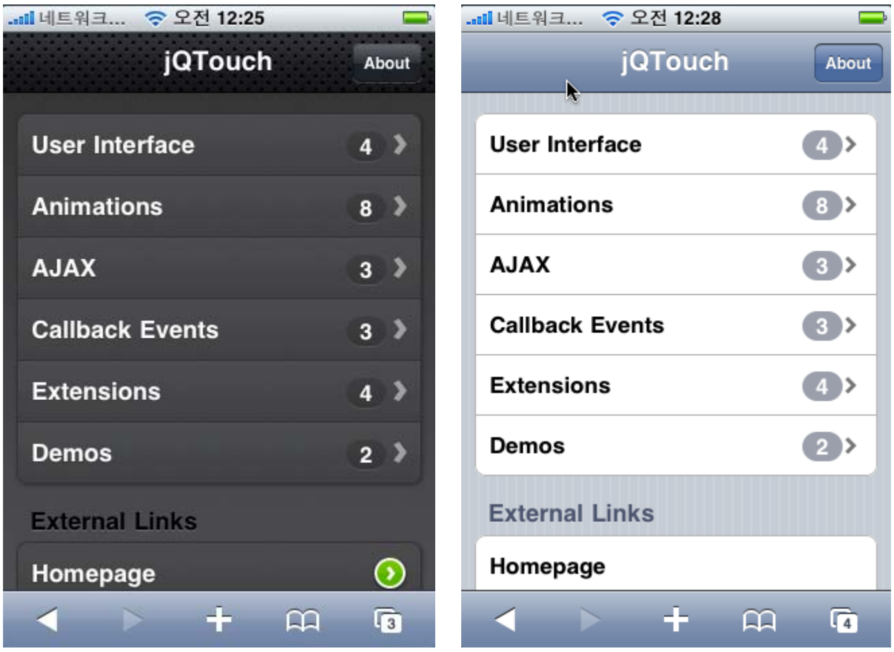
이 각각의 애니메이션 이름을 a 태그의 클래스명으로 지정하면 자동으로 애니메이션이 수행된다.

```
<a href="#foo" class="dissolve">Foo</a>
```

이렇게 애니메이션을 지정하면, Back 버튼으로 돌아올시 자동으로 반대쪽 애니메이션도 처리된다.

jQTouch 테마 지정

jQTouch 는 기본으로 2가지의 테마를 지원한다.



jQTouch Theme

```
<style type="text/css" media="screen"> @import
"themes/jqt/theme.min.css"; </style>
```

Apple Theme

```
<style type="text/css" media="screen"> @import
"themes/apple/theme.min.css"; </style>
```

CSS 만 변경함으로서 간단히 테마를 선택할 수 있다.

5.2.4 Phonegap

Web App 은 기본적으로 웹 서버에 올려지고 원격으로 모바일 디바이스에 다운로드 받는 방식으로 진행된다. 즉, 구조 자체는 기존의 Web Site 와 다른점이 없다.

여기서 소개하는 Phonegap 은 작성된 Web App 의 소스를 Native App 안에 내장하여 모바일 디바이스에는 앱처럼 설치하고 실행은 웹브라우저가 로컬에 있는 파일을 불러들이는 형태의 Hybrid App 을 만들 수 있게 한다.

Hybrid App 의 장점이 몇가지 있다.

- Multi Platform – Web App 하나로 iPhone,Android,Blackberry,PalmPre 등 다양한 디바이스동시 지원 가능
- Camera , Contacts 와 같은 Device API 에 접근할수 있는 방법이 제공
- App Store / Market 을 통한 어플리케이션 판매 가능

Hybrid App 를 만들수 있도록 하는 Cross Platform Mobile Application Framework 들도 다양하게 등장하고 있다.

- PhoneGap: iPhone, Android,Blackberry,Symbian,Palm,Windows Mobile (제한적) http://www.phonegap.com/
- Titanium Mobile http://www.appcelerator.com/ iPhone, Android
- QuickConnect http://quickconnectfamily.org/ iPhone, Android,Blackberry
- NimbleKit http://www.nimblekit.com/ iPhone

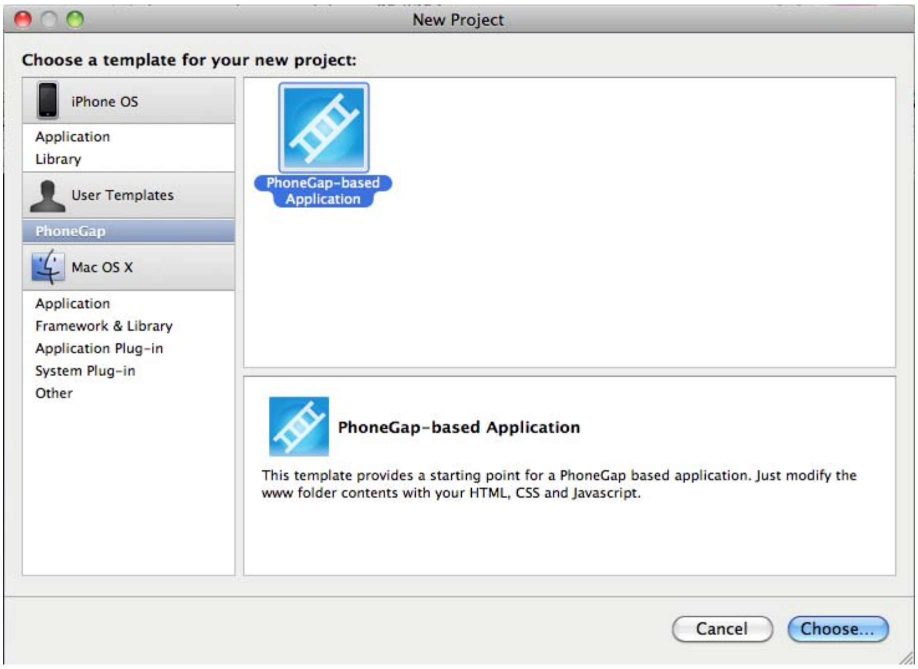
그나마 Phonegap 은 오픈 소스로 되어있어 많은 개발자에 의해 발전되고 있다. 여기서는 Phonegap 을 이용하여 WebApp 을 Native App 으로 만드는 방법을 알아보자.

Phonegap 소스를 http://github.com/phonegap/ 에서 내려받아 Mac 에 설치한다.

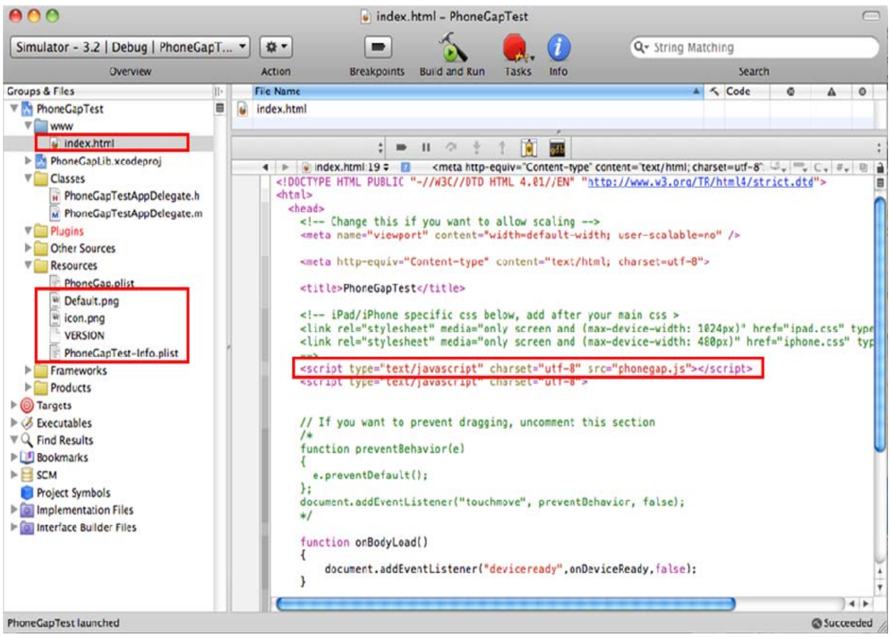
* Phonegap 은 NativeApp 들을 위한 Container 들을 제공하여 컴파일후 Native App 을 만드는 방법을 사용하고 있으므로, 기존 Web App 개발과는 달리 Mac 과 XCode 개발환경을 필요로 한다.

설치하고 나면 XCode 의 새 프로젝트 생성화면에 아래와 같이

PhoneGap based Application 이라는 항목이 생긴다.



템플릿을 선택하면 프로젝트 이름만 입력하면 된다. 그럼 아래와 같이 폰갭 프로젝트가 생성된다.



개발자가 눈 여겨 봐야 할 부분은 프로젝트 내의 www 폴더다. 이 폴더가 Web Root 역할을 하며 폰갭 어플리케이션이 실행되면 여기에 있는 index.html 을 로드해서 보여주게 된다. 즉, 자신이 작성한 Web App 소스를 프로젝트의 www 폴더에 복사하는 것 만으로 기본 준비는 끝나게 된다.

우측에 보면 javascript 를 하나 로드하는게 있다.

```
<script type='text/javascript' src='phonegap.js'></script>
```

이 Phonegap.js 는 모바일 디바이스의 웹 브라우저를 확장하여 Device 관련한 기능들 (Camera , Contacts ..) 에 접근할수 있도록 하는 역할을 한다. 이를 통해 확장되는 기능을 몇 개 살펴보자.

```
navigator.notification.alert ( message , title , button )       //
경고메시지 창을 띄운다.
navigator.notification.beep ( times )                          // 비프음을 n 회
울린다.
navigator.notification.vibrate ( duration )                    // 진동을 n 초간
울린다.
navigator.notification.loadingStart ( options )                // 로딩화면을
보여준다.
navigator.notification.loadingStop ();                         // 로딩화면을
닫는다.

device.platform            // Device 의 플랫폼을 읽어온다.
iPhone/Android ..
device.version             // 플랫폼 버전
device.name         // Device 의 이름
device.uuid         // iPhone 의 경우 UUID 값을 읽어온다.

navigator.camera.getPicture( success , fail , options );        //
카메라에서 사진을 가져온다.
  Options 에 sourceType = 0 은 폰의 포토앨범 , 1 은 카메라에서 사진을 가져온다.


// 폰에 있는 연락처 정보를 읽어오거나 추가/삭제 동작을 한다.
navigator.contact.contactsCount ( function(num) { alert(num); } ,
fail );
navigator.contact.getAllContacts( function(contactsArray) {}  , fail ,
options );
navigator.contact.chooseContact ( function(contact) {} , options );
navigator.contact.displayContact(contactsArray[x].recordID , fail,
options );
navigator.contact.removeContact(contactsArray[x].recordID , succ ,
fail );
navigator.contact.newContact(nc , succ );
```

위에서 보는 바와 같이 Javascript 에 확장기능을 추가함으로써, Web App 이 Device 의 기능을 사용할 수 있게 된다. 실제로 이런 시도는 W3C DAP , OMTP BONDI , JIL 에 의해 제안되고 표준화 작업이 진행되고 있으나, 아직 실제적용은 되지 않은 상태이다. 그리고 iPhone 에서 이 표준이 받아들여질지도 아직은 미지수 이기 때문에 현재로선 이런 Hybrid 프레임워크가 대안이라고 볼수 있다.

5.3 실전예제

5.3.1 오프라인 앱 어플리케이션 – 체크리스트

본 예제는 http://berttimmermans.com/checklist/에 대한것이다. Web App 를 좀더 간략하게 만든 것이 Javascript 만으로 실행 가능한 Web App 을 만들었다면 HTML5 의 App Cache 기능을 활용해서 Offline 에서도 실행이 가능한 Application 을 만들 수 있다.

Offline 에서도 실행 가능한 Web App 의 HTML 내용은 다음과 같다.

```
<!DOCTYPE html>
<html manifest="webapp.manifest">
<head>
 <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0"/>
 <meta name="apple-mobile-web-app-capable" content="yes" />
 <meta name="apple-mobile-web-app-status-bar-style" content="black" />
 <link rel="apple-touch-icon" href="iphone icon.png"/>
 <link rel="apple-touch-startup-image" href="startup.png" />

<script src="app.js"></script>
```

간단한 체크리스트를 Offline 에서 등록/관리 하는 Application 이다. 주요 기능은 app.js 에 있다.

```
// BUILD DATABASE --------------------------------

var db;

if (window.openDatabase) {
  db = openDatabase("Checklist", "1.0", "Checklist Item DB", 100000);
  if (!db)
    alert("DB Open Error");
  else
 var highestId = 0;
 var hideDelete = true;
}


function loaded()
{
// 페이지가 로드되면 테이블이 있는지 확인하여 없으면 새로 테이블을 생성한다.

  db.transaction(function(tx) {

    tx.executeSql("SELECT COUNT(*) FROM ToDos", [], function(result) {
      BuildList();
      //alert("loaded database");
    }, function(tx, error) {
      tx.executeSql(
```

```
"CREATE TABLE ToDos (id REAL UNIQUE, item TEXT, status REAL)", [],
function(result) {
       BuildList();
     });
    });
  });
}

// 페이지 로드후 이벤트에 등록한다.
addEventListener('load', loaded, false);


// NEW ------------------------------------------------------------

function newItem(){

 if(document.getElementById('item').value != ""){
 highestId ++;
 var item = document.getElementById('item').value;
 var status = 0;

// 신규 아이템 등록
 db.transaction(function (tx) {
   tx.executeSql("INSERT INTO ToDos (id , item, status) VALUES ("+
highestId
+", '"+ item +"', " + status + " )", [],
     function(result){
   document.getElementById('item').value = "";
   BuildList();
  } , function(tx, error) {
   alert(error);
  }
   );
  });
  }

}


// DELETE ------------------------------------------------------------

function deleteToDo(id){
// 아이템 삭제
 db.transaction(function (tx) {
   tx.executeSql("DELETE FROM ToDos WHERE id = " + id + ";", [],
     function(result){
   document.getElementById(id).style.display = "none";
  } , function(tx, error) {
   alert(error);
  }
   );
  });

}


// CHANGE STATUS -----------------------------------------------------
-----
```

```
function updateToDo(id, status){

if (status == '1') {
 status = '0';
 } else {
 status = '1';
 }

// 해당 체크리스트 아이템을 수행했는지 안했는지를 체크한다. 체크한 아이템은 흐리게
만든다.

 db.transaction(function (tx) {
    tx.executeSql("UPDATE ToDos SET status = " + status + " WHERE id =
" + id + " ;", [],
      function(result){
  if (status == '1') {
   document.getElementById(id+"box").removeAttribute("onclick");

   var newfunction = document.createAttribute("onclick");
    newfunction.nodeValue = "updateToDo(" + id + ", " + status + ")";
   document.getElementById(id+"box").setAttributeNode(newfunction);

   document.getElementById(id).style.opacity = '0.2';
  }
  if (status == '0') {
   document.getElementById(id+"box").removeAttribute("onclick");

   var newfunction = document.createAttribute("onclick");
    newfunction.nodeValue = "updateToDo(" + id + ", " + status + ")";
   document.getElementById(id+"box").setAttributeNode(newfunction);

   document.getElementById(id).style.opacity = '1';
  }
   } , function(tx, error) {
    alert(error);
   }
    );
  });

}


// BUILD LIST --------------------------------------------------

function BuildList(){

// DB 에서 전체 체크리스트 아이템을 읽어온다.

 document.getElementById('items').innerHTML = "";

  db.transaction(function(tx) {
    tx.executeSql("SELECT id, item, status FROM ToDos", [],
function(tx, result) {
      for (var i = 0; i < result.rows.length; ++i) {
        var row = result.rows.item(i);

        ToDo(row['id'], row['item'], row['status']);
```

```
      if (row['id'] > highestId)
        highestId = row['id'];
     }

     // if (!result.rows.length)

    }, function(tx, error) {
     alert('Failed to retrieve notes from database - ' +
error.message);
     return;
    });
  });
}



// TODO -----------------------------------------------------------

function ToDo(id, item, status){

// 각 아이템 등록

  var ToDoItem = "<div id='" + id + "' class='part ";
  if(status == 1) {
   ToDoItem += " done";
  }
  ToDoItem += "'> <input type='checkbox'";
  if(status == 1) {
   ToDoItem += "checked='checked'";
  }
  ToDoItem += " onclick='updateToDo(" + id + ", " + status + ")' id='"
+ id + "box' class='checked' />";
  ToDoItem +="<span>" + item + "</span>";
  ToDoItem +="<input type='button' value='Delete'
onclick='deleteToDo(" + id + ")' class='dbtn' id='Dbtn" + id +
"'/></div> ";

  document.getElementById('items').innerHTML =
document.getElementById('items').innerHTML + ToDoItem;
}



// TOGGLE HIDDEN -----------------------------------------------------
-

function toggleHidden(){
 if(hideDelete == true){
   hideDelete = false;
   document.getElementById('Edit').value = "Done";
 } else {
   hideDelete = true;
   document.getElementById('Edit').value = "Edit";
 }
 BuildList();
}
```

그리고, 오프라인 실행을 위해 webapp.manifest 를 만든다.

```
CACHE MANIFEST
app.js
startup.png
# 캐시를 선언한 index.html 은 자동으로
```

실행한 화면은 다음과 같다. 한번 온라인상태에서 로딩만 되면 오프라인 시에도 동작된다.


```
<meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0"/>
<meta name="apple-mobile-web-app-capable" content="yes" />
<meta name="apple-mobile-web-app-status-bar-style" content="black" />
<link rel="apple-touch-icon" href="iphone icon.png"/>
<link rel="apple-touch-startup-image" href="startup.png" />
```

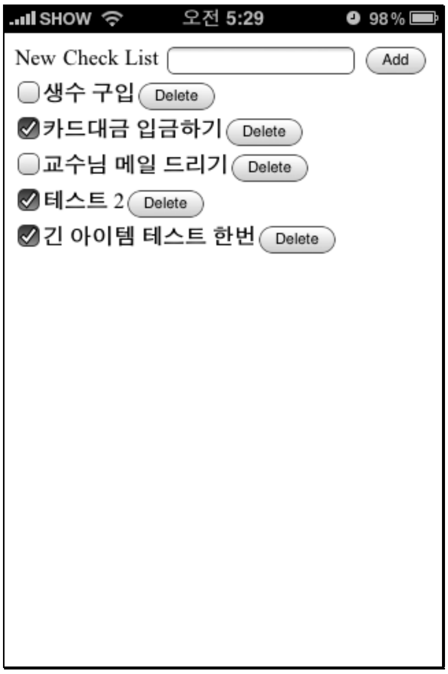
그리고, HTML의 <head> iPhone Web App 모드를 지원하는 태그를 추가했기 때문에 하단 + 버튼을 눌러서 추가하면 다음과 같이 지정한 iphone_icon.png 아이콘이 보이게 된다.

추가된 아이콘을 눌러서 실행하면 startup-image 로 지정한 화면이 로딩할 때 보이게되며, URL 바 없이 전체화면으로 실행된다. Status Bar 의 색상은 검정색이다.

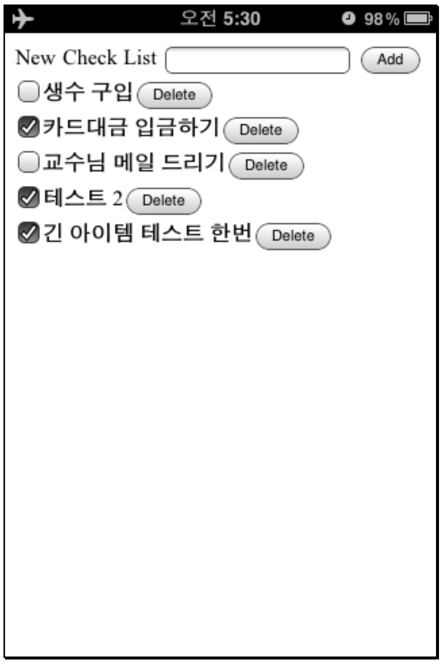
예제는 http://dev.xguru.net/checklist/ 에 올려져 있으므로, 아이폰에서 접속하면 바로 사용할수 있다.

[아이콘 추가]

[로딩 화면 이미지]



[전체화면 모드로 실행]



[Offline 모드]

5.3.2 GeoLocation 어플리케이션 - GeoChicken

이제 간단한 위치정보 Web App 을 만들어보자.

```
<!DOCTYPE html>
<html>
<head>
```

```
 <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0"/>
 <meta name="apple-mobile-web-app-capable" content="yes" />
 <meta name="apple-mobile-web-app-status-bar-style" content="black" />
 <link rel="apple-touch-icon" href="iphone icon.png"/>
 <link rel="apple-touch-startup-image" href="startup.png" />

// 예제를 위해 Daum 지도 API 사용
<script type="text/javascript"
src="http://apis.daum.net/maps/maps2.js?apikey=DAUM MAP APIKEY"></scri
pt>

<script src="app.js"></script>
 <title>GeoChicken WebApp</title>
</head>
<body>
<div id=map canvas style='width:300px;height:300px'></body>
</html>
```

역시 간단한 소스코드는 app.js 파일에 있다.

```
// 페이지가 로드되면 위치정보를 요청하여 바로 그 위치의 지도를 그린다.
function loaded()
{
navigator.geolocation.getCurrentPosition(
function(position) {
            var map = new DMap("map canvas", {point:new
DLatLng(position.coords.latitude,
position.coords.longitude), level:3});
},
function(){
alert('Error');
}
);
}

// 페이지 로드후 이벤트에 등록한다.
addEventListener('load', loaded, false);
```

페이지를 로딩하면 바로 현재 위치정보 사용 승인창이 나오고 승인 버튼을 누르면, 현재 있는곳 부근의 지도정보를 웹 페이지에 보여주게 된다.

지도정보만 보여서는 기존 지도 앱과 다를바 없으니, 다른 기능을 한번 추가해 보자. 데모 이름이 GeoChicken 이라는걸 확인했다면 무엇을 할지 바로 알 수 있다. 클릭 한번만으로 내 주위에 치킨 집을 바로 검색해보도록 하자.

```
<!DOCTYPE html>
<html>
<head>
 <meta name="viewport" content="user-scalable=no, width=device-width,
initial-scale=1.0"/>
 <meta name="apple-mobile-web-app-capable" content="yes" />
 <meta name="apple-mobile-web-app-status-bar-style" content="black" />
 <link rel="apple-touch-icon" href="iphone icon.png"/>
 <link rel="apple-touch-startup-image" href="startup.png" />

// 예제를 위해 Daum 지도 API 사용
<script type="text/javascript"
src="http://apis.daum.net/maps/maps2.js?apikey=DAUM MAP APIKEY"></scri
pt>
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
</script>

<script src="app.js"></script>

 <title>GeoChicken WebApp</title>
</head>
<body>
<div id=addr></div>
<ul id=items></ul>
// 지도에 위치표시 하는 기능은 생략.
</html>
```

역시 간단한 소스코드는 app.js 파일에 있다.

```
function loaded() {
navigator.geolocation.getCurrentPosition(
function(position) {

var lat = position.coords.latitude;
var lon = position.coords.longitude;

// 다음 좌표->주소 API 로 현재 좌표를 주소로 변환한다

var coord2addr =
"http://apis.daum.net/maps/coord2addr?apikey=DAUM MAP APIKEY&longitude
=" + lon + "&latitude=" + lat +
"&output=json&inputCoordSystem=WGS84&callback=?";

// AJAX 호출을 조금 편하기 표시하기 위해 jQuery 를 사용
$.getJSON(coord2addr, function(data){

$("#addr").html(data.fullName);
addr = encodeURI(data.fullName);

// 현재 국내 API 중 좌표만으로 근방 상호들을 검색해주는 API 는 없으므로
// 네이버의 지역검색 API 에 "주소 + 치킨" 형태로 검색을 호출하여 근처 치킨집을
찾는다.
// 네이버의 API 는 JSONP 지원이 안되서 Cross Domain 호출이 불가하므로 서버에서
대신 처리한다.
// 아래 주소는 네이버의 지역검색 API 호출후 결과를 JSON 으로 변경하여 보내준다.
//
http://openapi.naver.com/search?key=NAVERKEY&target=local&start=1&display=10&q
uery=
// 검색요청시 주소+" " +치킨
var getChicken =
"http://dev.xguru.net/geo/get.php?addr="+addr+"&callback=?";

$.getJSON(getChicken, function(data) {
$.each(data.stores, function(i,item){

store = "<li>" + item.title +
" <a href='tel:" + item.telephone + "'>" + item.telephone +
"</a></li>" ;

$("#items").append(store);

});
}
);
});
},
function(){
alert('Error');
}
);
}

// 페이지 로드후 이벤트에 등록한다.
addEventListener('load', loaded, false);
```

예제 소스를 http://dev.xguru.net/geo/에 올려두었다. HTML5 GeoLocation API를 지원하는 모든 브라우저에서 다 실행 가능하다. 실행하면 다음과 같은 화면을 볼수 있다.

이 사이트 역시 + 버튼을 이용하여 웹 앱으로 추가가능하며 한번의 클릭으로 원하는 치킨집을 찾을수 있다. 이와 같이 HTML5의 GeoLocation과 Web SQL Database API를 이용하여 간단한 Mobile HTML5 어플리케이션을 만들어 보았다.

부록.

[블로터포럼] HTML5가 개발자에게 '기회의 땅' 인 이유

출처: http://www.bloter.net/archives/24791

우연의 일치일까요. '블로터 포럼'을 진행한 뒤 애플 스티브 잡스가 때마침 제대로 한 방 날렸더군요. 아이폰에서 플래시를 지원하라는 어도비를 향해 '플래시 대안은 HTML5'라며 '어도비는 게으르다'고 심기를 건드린 겁니다.

왜 갑자기 여기저기서 HTML5를 외치는 걸까요. 특정분야 개발자들을 빼고는 대체로 비슷한 반응을 보입니다. HTML에 익숙한 사람도 HTML5 앞에선 꿀 먹은 벙어리마냥 얌전해집니다. 딱딱하고 어려운 게 사실이지만 알아두고 준비해야 할 기술. 이번 '블로터 포럼'에선 입문자 눈높이에 맞춰 HTML5를 들여다보고 싶었습니다.

- 일시 : 2010년 1월27일(목) 오후 5시~7시
- 장소 : SK커뮤니케이션즈 회의실
- 참석자 : 윤석찬 다음커뮤니케이션 DNA랩 팀장, 도안구 · 이희욱 · 주민영 블로터닷넷 기자

이희욱 | **오늘 주제가 참 어렵다. HTML5 문외한 입장에서 궁금한 점이 많다. 먼저 묻고 싶다. HTML5가 뭔가?**

윤석찬 | HTML5가 어느 날 갑자기 나타난 게 아니다. 사연이 길다. 1998년 HTML4.01 이후 웹표준을 개발하는 국제 컨소시엄인 W3C는 XHTML

표준 개발에 박차를 가했다. 웹브라우저 전쟁 이후 그 작업에서 웹브라우저 제조사들이 빠졌다. 이후 웹표준의 방향은 XML을 기반한 꽤 이상적인 표준을 만들기 시작했다. 2004년 파이어폭스가 나오고 아작스(Ajax)와 웹2.0이 활성화되면서 문서가 아닌 웹애플리케이션을 위한 웹표준의 재정비가 필요했다.

하지만 이러한 현실적 요구를 W3C가 받아들이지 않았다. 웹브라우저 제조사들에게는 W3C의 XHTML2.0과 XML기반 DOM 및 이벤트 핸들러 등은 루비콘 강을 건너는 것이다. 당시 XHTML 문서가 전체 웹에서 5%에 불과했고 웹브라우저 엔진들의 차이 탓에 개발자들은 '크로스 브라우징'에 생고생을 하고 있었다. 2004년 W3C의 한 워크샵에서 서로 틀어진 뒤 모질라와 오페라, 애플과 구글은 별도의 '웹 하이퍼텍스트 애플리케이션 테크놀로지 워킹그룹'(WHATWG)이라는 공개 표준 그룹을 만들고 새로운 HTML 표준안을 만들기 시작했다. 이것이 바로 HTML5의 시초다.

도안구 | **W3C와 웹브라우저 제조사 사이에 그런 의견 다툼이 있었나? 흥미롭다.**

윤석찬 기존 WHATWG 표준 초안을 가져오며 'HTML5'라 불렀다. 당시 IE7 개발을 맡았던 MS 유명 아키텍트인 크리스 윌슨이 워킹그룹 의장이 됐고 모질라, 오페라, 애플, 구글 등 모두 참여해 HTML5 표준을 만들고 있다.

HTML5 독타입은 매우 간단하다. '< !DOCTYPE HTML >' 이렇게 HTML 파일 맨 앞 줄에 넣어주면 끝이다. 이 뒤에 나오는 코드는 웹브라우저마다 HTML5에 맞춰 렌더링한다. HTML5 표준 초안은 웹브라우저 엔진 개발자들을 위해 만든 것으로, 보다 상세하게 구현 내용을 적고 있다.

두 번째 목적은 동적 웹애플리케이션을 지원할 리치 웹 기술을 담고 있다는 것이다. 멀티미디어를 다루는 'canvas', 'video', 'audio' 태그를 비롯해 웹브라우저 내 로컬 스토리지를 다루는 돔 API와 드래그앤드롭 API 등 일반 표준 문서에서는 보기 힘든 다양한 기술이 뒤섞여 있다. 특히 웹 개발자 수고를 덜어줄 '웹폼2.0'이라는 표준과 함께 쓰면 보다 멋진 리치 웹애플리케이션을 만들 수 있다. 웹브라우저 안에 DB를 탑재해 로컬 스토리지로 활용해 오프라인에서도 데이터를 싱크해 활용할 수 있다. 구글 G메일 '오프라인' 기능이 그렇게 구현돼 있다.

이희욱 **리치 웹애플리케이션이라면 플래시나 실버라이트를 얘기할 때 자주들 언급한다. HTML5가 리치 웹에 주목하는 이유는 무엇인가?**

윤석찬 웹브라우저 업체 입장에서 리치 웹 기술에 관심을 갖는 이유는 다양하다. 제가 참여하고 있는 모질라 커뮤니티의 경우, 웹은 읽을 수 있고(readable), 저장할 수 있고(Indexable), 편집할 수 있어야(editable) 한다고 믿는다. HTML 소스를 보고, 복사를 하고, 고칠 수 있었기 때문에 웹 문서가 비약적인 성공을 했다. 기존 플러그인 기반 리치 웹 기술들, 예컨대 플래시나 실버라이트는 그게 어렵다. 물론 이들도 XML 기술을 통해 이용자화면(UI)을 만들 때 스크립트 언어로 동작을 제어한다. 하지만 결국 읽을 수 없는 '바이너리'를 포함하고 있다. 이는 웹 본질과 일치하지 않는다. HTML5가 리치 웹 기술의 선택 가능한 대안으로 자리잡아야 한다.

물론 아직 플래시나 실버라이트에 비해 HTML5가 제한 사항이 많다. 하지만 궁극적으로 웹이 나아갈 방향이라고 본다. 구글이나 오페라와 애플도 이러한 점에 동의를 하고 있고 MS 역시 미온적이지만 참여를 하고 있다. 초창기 많은 사람들이 '리치 웹 환경에서 HTML5가 성공할 것인가'라는 물음엔 회의적이었다. 지금은 많이 바뀌었다. 이런 '블로터 포럼'에도 불려다니는 걸 보면. (웃음)

도안구 **HTML5가 주목 받게 된 특별한 계기나 사건이 있나?**

윤석찬 아무래도 구글 영향이 컸다. 지난 2009년 4월에 열린 '구글 I/O

컨퍼런스’가 전환점이 됐다. 구글은 2008년 첫 구글 I/O 컨퍼런스에서 안드로이드와 구글 기어스를 발표했다. 구글 기어스는 리치 웹애플리케이션을 만들기 위한 웹브라우저 플러그인이었다. 하지만 2009년 컨퍼런스에선 구글 CTO가 첫날 주제로 HTML5를 다루고, 둘쨋날 구글 웨이브를 다뤘다. 그런데 첫날 HTML5를 얘기하면서 ‘HTML5가 대세’란 분위기를 크게 조성했다. 자사 웹브라우저인 ‘구글 크롬’에도 아직 탑재 안 된 HTML5 기술을 파이어폭스와 사파리로 시연할 정도였다. 그러면서 인식이 많이 바뀌었다. 구글이 드디어 HTML5에 베팅하는구나. 시장에도 긍정적인 신호를 줬다.

모바일을 보면 완전히 다르다. 지금 PC의 웹브라우저 시장은 IE가 다수이고 파이어폭스, 크롬, 사파리가 따라오는 모양새다. 모바일 웹에서는 유럽 스마트폰 시장은 오페라가, 아이폰은 사파리를 기반하고 있다. 안드로이드폰이 나오면 크롬이 주력으로 들어간다. 모질라를 빼도 메이저 3사다. 결국 IE가 대세가 아니다. 모바일 애플리케이션 개발 환경은 PC 못지않게 폐쇄적이다. 이런 상황은 오래 가지 않을 것이고, 결국 범용 리치 웹 환경을 사용하는 것으로 바뀔 것이다. 특히 모바일 웹의 변화가 더욱 빠를 것 같다.

주민영 **허나 애플 아이폰이 촉발시킨 앱스토어도 개발자 입장에선 큰 기회다!**

윤석찬 물론 지금은 앱스토어가 유행이다. 돈벌이가 아니라 서비스를 만드는 관점에서 보면, 지금은 앱스토어용 따로, 웹애플리케이션 따로 만드는 식으로 과도기다. 결국 HTML 표준으로 웹 문서를 만들듯 웹애플리케이션도 표준으로 쉽게 만들고 서비스하는 환경이 와야 한다. 폐쇄적인 개발 플랫폼을 기반으로 하는 플레이어도 필요하지만 모두에게 이익이 되는 범용 개발 환경이 웹의 목표이고 지향하는 바다. 웹 개발자들은 이를 간과하면 안된다.

이희욱 **HTML5는 그럼 웹 개발자들을 위한 표준 기술 문서인가?**

윤석찬 앞서 말했듯이 HTML5는 웹브라우저 엔진 개발자를 위한 스펙이다. 하지만 이 안에는 렌더링 엔진 뿐만 아니라 중요한 리치 웹 기술이 포함돼 있다. 예컨대 크롬이 탭마다 적용한 병렬 프로세스 기능이나 외부와 데이터를 주고받을 때 웹브라우저가 어떻게 처리할 지 규약도 있고, 데스크톱에서 웹브라우저로 드래그앤드롭한 파일을 어떻게 처리할 지에 관한 스펙도 있다. HTML 뿐 아니라 방대한 내용들이 추가되고 있다. 초안 자체가 어렵기 때문에 웹 개발자들이 이를 쉽게 이해하고 이용할 수 있도록 설명 문서들

도 함께 만들고 있다.

도안구 **그 스펙은 계속 추가되고 실제 구현되고 있나?**

윤석찬 W3C 표준 제정 과정을 보면, HTML5는 현재는 초안 단계다. 한 단계 넘어가기 위해 준비 중이고 이는 정해진 내부 프로세스를 따라가는 것이다. 무엇보다 중요한 건, HTML5의 어떤 기술이 웹브라우저에서 구현되고 있고 얼마만큼 사용할 수 있느냐 하는 점이다. 현재 PC 기반 웹브라우저에서 HTML5의 주요 기능을 쓰는 데는 아직 무리가 있다.

주민영 **유튜브나 비메오 같은 동영상 공유 사이트가 플래시 대신 HTML5를 수용하겠다고 해서 화제가 되기도 했다.**

윤석찬 유튜브나 비메오 등이 수용한 건 HTML5의 일부다. ‘video’ 태그를 이용해 플러그인 도움 없이도 웹브라우저 만으로도 동영상을 서비스할 수 있다. 지금까지는 윈도우 미디어 플레이어나 플래시 플러그인을 깔아야만 가능했다. 문제는 동영상 코덱에 있다. 파이어폭스와 오페라는 오픈소스 기반 OGG 테오라(OGG Theora)를 지지해왔다. 하지만 크롬과 사파리는 특허료를 내야하는 H.264 MPEG 포맷을 지원한다. 유튜브와 비메오도 H.264 코덱을 지원하기 시작했다. 이 때문에 파이어폭스도 H.264 코덱을 지원해야 한다는 여론이 일었다. 이에 대해 모질라 제품담당 마이크 셰이버는 거부 의사를 분명히 했다.

파이어폭스가 H.264 코덱을 이용하는 데 1년에 500만 달러 정도의 특허료를 지불해야 하다. 모질라 입장에서 그리 큰 돈은 아니다. 하지만 문제는 이를 통해 서비스 개발자 및 업체 모두 2011년부터 특허료를 내야 한다. 이는 선택 가능한 대안을 중요시하는 모질라의 미션과 배치되는 것이다. 코덱은 물론 웹의 영역은 아니다. 하지만 플러그인들이 오픈웹에 큰 걸림돌이 되듯, 폐쇄형 코덱은 오픈 비디오 환경에 도움이 되지 않는다고 판단한다.

이희욱 **그럼 유튜브 HTML5 비디오 태그와 파이어폭스의 연동은 영영 안 되는 건가?**

윤석찬 가능성은 있다. 구글이 지난해 8월, 동영상 코덱 업체 ‘온투(On2) 테크놀로지’를 인수했다. 구글이 온투 코덱을 오픈소스와 특허 무료로 공개하는 거다. 온투 코덱은 플래시와 호환된다. 이러한 계획은 이미 구글도 밝힌 바 있다. 테오라 역시 온투의 과거 버전이 오픈소스화 된 것이다. 오

폰 비디오 환경은 이래저래 구글의 결정에 큰 영향을 받게 될 것이다.

도안구 **국내 웹사이트들의 HTML5 도입 현황은 어떤가?**

윤석찬 아직까지 국내에서는 HTML5에 대한 웹 개발자들의 관심이 높지는 않다. HTML5 독타입을 쓰면 표준 모드로 동작하므로 사용해도 지장은 없다. 우선 HTML5에 대한 문서자료와 HTML5갤러리나 HTML5닥터 웹사이트에서 다양한 예제를 살펴보고, 가능한 것부터 해보는 것이 좋겠다.

이희욱 **그럼 XHTML은 더 이상 개발 되지 않는 것인가?**

윤석찬 그렇지 않다. 물론 XHTML 2.0 표준 개발은 완전히 멈췄다. 지난해에 그룹이 해체됐다. 하지만 XHTML의 유용성은 그대로 있기에, HTML5 문서를 XHTML로도 표현할 수 있고 이를 위한 독타입을 선언하면 그대로 XHTML 문서로 유효하다. 이를 ‘XHTML5’라고 부른다. XHTML은 여전히 HTML5 안에서 유효하다.

주민영 **HTML5가 산업에 미치는 영향은 어떨 것으로 예상하나?**

윤석찬 가장 큰 수혜자는 기존 웹 개발자다. 요즘 모바일 애플리케이션 개발자는 중고 매킨토시를 산 뒤 코코아 개발환경을 익혀 앱스토어에 애플리케이션을 올리고, 안드로이드 애플리케이션 개발자는 자바를 배워야 한다고들 말한다. 하지만 자신이 가진 웹 기술에 조금만 더 보태면 감탄할 만 한 리치 애플리케이션을 만들 수 있다. 예컨대 ‘R그래프 ‘란 서비스를 보면 HTML5를 기반한 각종 비주얼 차트를 서비스 안에 넣을 수 있다.

그러니 웹 프론트엔드 개발자가 더 많은 생각을 갖고 HTML5를 적용하는 노력을 기울여야 한다. 그게 결국은 자기에게 보답으로 돌아온다. 전세계에 제공되는 범용 웹브라우저 기반으로 웹애플리케이션을 만들면 모든 개발자가 수혜를 받는다. 결국 이게 정석이다.

웹 산업에서 대형 주자가 폐쇄된 개발 환경과 플랫폼에서 비즈니스하는 것은 당연하다. 좋은 이용자경험(UX)을 주는 것은 칭찬할 만 하다. 중요한 것은, 선택 가능하고 범용적인 웹 기반 플랫폼도 제공돼야 한다. 표준은 죽기도 하고 산업에 밀리기도 한다. 100% 올바르지도 않다. 하지만 없는 것보다는 낫다.

이희욱 **HTML5 확산을 위한 과제가 있다면?**

윤석찬 국내에서는 일단 HTML5가 대형 포털이 적용할 만큼 매력이 있느냐의 문제가 있다. 국내에서 이용하는 대다수 웹브라우저가 아직 지원하지 않는 부분이 있기 때문이다. 파일럿 서비스나 모바일 웹 서비스를 준비하는 사람은 HTML5를 적용해보면 좋겠다. 아이폰용 웹 페이지를 만들 때 'video' 나 'canvas' 태그 혹은 오프라인 스토리지 기능을 이용하는 아이디어를 내야 한다. 천편일률적인 모바일 페이지는 식상하다. 기왕이면 모바일 웹페이지를 만들 때 '엣지있게' 만들면 좋잖나. 만약 누군가 'canvas' 태그를 이용해 그림을 그리고 이를 공유하는 서비스를 모바일 웹서비스로 만들었다 치자. 그는 회사에서 인정받는 사람이 될 수 있다. 그런 점들에 개발자가 좀 더 신경쓰면 좋겠다. 스스로 찾고 배워서 도전해 봤으면 한다.

이희욱 새롭고 흥미로운 얘기들을 많이 들었다. 아직은 어렵고 낯선 면이 많다. 리치 웹을 플러그인 없이 구현할 수 있는 기술을 내장했다는 얘기가 기억에 오래 남는다. 웹 개발자분들이 좋은 기회로 활용했으면 하는 생각이 든다. 새로운 정보들도 자주 알려주시길 기대한다.

[읽을꺼리] HTML5 동영상에 대한 전망

출처: http://channy.creation.net/blog/801

HTML5 동영상을 두고 애플과 어도비 그리고 웹 브라우저 벤더와 서비스 업체들의 혼란이 가중되고 있다.

유튜브를 비롯 많은 웹 서비스들은 아이폰과 아이패드 호환성을 위해 HTML5 <video> 기능을 제공하고 있는데 반해 최근 인터넷 TV 서비스인 Hulu는 <video>를 관망하고 플래시 동영상을 유지하겠다는 입장을 보였기 때문이다. 마치 과거 넷스케이프 vs. IE와 이에 따른 웹 서비스들의 줄서기 같은 데자뷰가 일어난다고 이야기 하는 분들도 있다. 중요한 것은 웹의 본질을 이해하고 스스로 판단해야 한다는 점이다.

콘테이너를 벗어나 웹의 일부가 되다

웹의 일부냐 아니냐를 가를 수 있는 구분자는 바로 웹에 있는 자원을 우리가 찾고(URL addressable), 읽고(Readable), 저장(Indexable)할 수 있느냐는 것이다.

웹이 처음 만들어졌을 때, 음성 및 동영상 같은 멀티미디어들도 웹의 일부가 될 수 있는 기술적 고려는 있었다. 하지만, 90년대말과 2000년대 초반 인터넷 방송을 해 봤던 경험에 따르면 당시 네트워크 속도, 저작권, 공유 기술의 부족으로 인해 플러그인의 콘테이너에 갇혀 제공될 수 밖에 없었다.

지금까지 웹 상의 동영상은 실버라이트, 리얼플레이어, 윈도우 플레이어 등에 갇혀 있었고 웹 2.0 조류를 타고 오디오의 경우 팟캐스트, 동영상은 유튜브가 나오고 나서야 사용자가 직접 만들어 공유 가능하게(sharable) 서비스가 나오기 시작했다. 특히 과거에 비하면 서비스 비용도 획기적으로 낮아졌고, 광고 시장의 성장도 눈부셔서 멀티미디어가 웹의 일부가 될 수 있는 토대가 된 것은 웹2.0의 또다른 혜택 중 하나다.

진정한 개방 동영상(Open Video) 시대로

동영상이 웹의 일부가 된다는 말은 무엇일까? 바로 웹 문서와 같이 콘테이너를 벗어나 URL을 통해 사용자나 검색 크롤러가 읽고(보여주고) 저장할 수 있다는 이야기이다.

얼마 전 CSS 창시자인 하콤비움리 국내 강연 중 "HTML5에서 동영상 저작권은 어떻게 해결해야 하나?"라는 질문을 던진 사람이 있다고 한다.

HTML5 동영상이 URL로 자유롭게 제공하여 크롤러나 사람이나 보고 저장할 수 있다는 것을 전제로 한다. 만일 저작권이 중요하다고 판단하는 서비스 제공자는 기존 처럼 콘테이너에 담거나 DRM 솔루션으로 꽁꽁 묶어 제공하면 된다. (Hulu가 HTML5 동영상에 소극적인 이유도 바로 이 때문이다.)

<video>와 <audio> 태그가 HTML5의 기술 사양에 포함된 것은 일찌기 이미지 파일이 그랬듯이 동영상을 통해 더 많은 사람들이 더 많은 정보 공유의 이익을 얻을 수 있기 때문이다.

유튜브에서 음성 인식을 통한 자동 자막 생성 기능을 경험해 보셨다면 이 기능이 얼마나 유용한지 알 수 있을 것이다. 누구나 웹 동영상을 접근할 수 있담면 이를 이용한 다양한 혁신 서비스가 나올 수 있다는 증거 중 하나다.

정보 접근성과 단계적 기능 축소

HTML5 동영상에 대한 또 하나의 관점은 코덱에 관한 것이다. 사파리, 크롬 (그리고 IE9)의 최신 버전은 특허료를 지급해야 하는 H.264(AVC)를 기본 지원하고, 파이어폭스, 오페라, 크롬은 특허 무료인 오픈소스 코덱 '오그 테오라' 를 지원하고 있다.

세간에서는 웹 코덱 전쟁이라고 칭하면서 어느 코덱이 대세가 될 것인지 이슈꺼리가 되고 있다. Mozilla나 오페라가 지향하는 바는 웹에 어떤 콘텐츠를 저작하거나 제공할 때, 누군가에게 돈을 내야 하는 것은 진정한 웹은 아니기 때문에 특히 무료 코덱을 주장하고 있다.

사실 웹 서비스 업체나 웹 개발자들이 누가 이길지 관망하면서 기다리는 것은 크게 도움이 되지 않는다. 단언하거나 HTML5 표준을 100% 구현하는 웹 브라우저도 없을 뿐더라 100% 같은 사양으로 구현한다는 것은 불가능하다. (과거 W3C의 웹 표준 사양을 100% 구현한 웹 브라우저도 없다.)

웹 개발의 중요한 고려 사항 중 하나는 단계적 기능 축소(Graceful Degradation)라는 개념이다. 자바스크립트를 꺼도 실행이 되고, CSS를 꺼도 보여주어 궁극적으로 HTML만 남아도 중요한 기능은 동작하게 하는 것이다. 우리가 웹 브라우저 호환성(Cross Browser compatibility)를 제공하는 것도 같은 이유이다. 이는 웹 개발자들의 필연적인 업무이다. 동영상 코덱의 경우, ogg나 mp4냐를 관망하는 건 의미가 없는 일이다. 접근성 향상에는 필연적으로 비용이 수반된다. HTML5 동영상 코덱 선택도 단계적 기능 축소 관점에서 접근할 필요가 있다는 의미이다.

구글이 이번 주 Google I/O에사 VP8 코덱을 오픈 소스로 제공하고, 여기에 애플을 제외한 어도비, 모질라, 오페라가 직접 참여하고, 마이크로소프트가 지원을 천명했기에 더욱 그렇다. 우리가 원하는 궁극적인 동영상 제공 방법은 바로 다음과 같이 한 줄의 코드로 요약할 수 있다.

```
<video>
<source src="openvideo.ogg" type="video/ogg"/>
<source src="openvideo.mp4" type="video/mp4"/>
<embed src="player.swf?openvideo.flv" type="application/x-shockwave-
flash" width="480" height="300" allowscriptaccess="always"
allowfullscreen="true"/>
</video>
```

HTML5 동영상은 HTML5의 극히 일부분에 지나지 않는다. HTML5의 수 많은 기술을 정확하게 이해하고 선택하는 것은 ‘웹 개발자’들의 몫이다. 주변에 휘둘리기 보다 내가 웹 개발자로서 어떤 판단을 해야 하는지가 더 중요하다는 의미이다. 그것이 우리가 웹 개발자로서 살아가야 하는 의무이자 책임이다.

국내 웹 표준 커뮤니티 목록

Web Standards Korea : webstandards.or.kr

국내 최초 웹 표준 커뮤니티로 2004년에 만들어져 다양한 강의,저술 활동, 공공 기관 지원을 하고 있다. 또한, 세계 웹표준 운동의 주축인 웹표준 프로젝트(Web Standards Project)의 지역화 단체이기도 하다.

“실전 웹 표준 가이드”(2005) 무료 배포, 한중일 공개SW포럼웹표준 W/G활동(2005~2008) 그리고 공공기관 웹 표준 평가 및 가이드 마련 등에 실질적인 도움을 주었다. 2007년 이후 겉으로 뚜렸하게 보이는 활동은 없지만 한국의 웹표준 커뮤니티 리더들이 필요한 경우, 공동 활동을 펼치기도 한다.

Mozilla 한국 커뮤니티: 웹표준 게시판 forums.mozilla.or.kr

모질라 재단에서 만든 웹 브라우저인 파이어폭스가 웹표준을 지키는 것을 중요하게 여기고 있어서 커뮤니티에 웹표준을 주제로 게시판이 생기게 되었다.

2004년 당시 웹표준에 대한 얘기를 할 수 있는 유일한 곳이어서 많은 사람들이 활동하고 토론하였고 이러한 활동을 지금까지고 계속되고 있다. 이곳에서는 표준을 지키지 않은 사이트나 웹표준 관련된 기술적, 이론적인 내용에 관련된 논의가 활발하게 이루어지고 있다.

CSS 디자인 코리아: forum.standardmag.org

웹표준과 CSS관련된 활동을 하던 사람들의 의견을 모아 장지윤님이 만든 사이트이다. 초기에는 토론게시판 형태로 운영이 되다가 최근에는 자료와 활동을 정리하는 위키를 도입하여 운영되고 있다. 현재 홍윤표님께서 총괄 운영을 맡아서 수고해 주고 계시며 게시판에서는 웹표준이나 웹 접근성 실무자 분들이 활발한 토의를 진행하고 있다.

최근에는 "한국 웹 표준의 날"이라는 행사를 시작하였고 한국의 웹표준을 이끌어가는 중심 행사로 자리 잡을 수 있도록 노력하고 있다.

클리어 보스(ClearBoth): clearboth.org

클리어보스는 추지호님을 중심으로 스터디 모임으로 시작하여 다양한 주제의 웹 표준 연구 모임이다. 친목과 공부 모임을 주관하면서 웹 표준 문서 번역를 번역하고 있기도 하다. 최근에 일본 CSS Nite를 국내에서 협력해서 행사를 개최하고 있다.

한국웹접근성그룹(Korea Web Accessibility Group): kwag.net

조훈님이 시작한 웹 접근성 전문 그룹이다. 일반인을 대상으로 한 워크샵을 진행하였고 한국 정보문화진흥원과 함께 한국의 웹접근성을 향상시키는데 중요한 역할을 하였다. 최근 다시 활동을 준비하고 있다.

네이버 카페 하코사: www.hacosa.com

하코사는 "하드코딩하는 사람들"의 준말로 네이버 카페에서 운영되고 있는 모임이다. 회원수가 3000명이 넘고 웹표준 뿐만 아니라 HTML을 다루는 다양한 직군의 회원들이 있다.

정보통신 접근성 향상 표준화 포럼: www.iabf.or.kr

국내의 웹접근성 전문가들의 포럼이다. 한국 정보문화 진흥원의 지원으로 운영되고 있고, 관련 세미나정보나 웹접근성 관련 자료들이 많다.

실전 HTML5 가이드

- 저자: 윤석찬, 신현석, 정찬명, 경준호, 권정혁

- 배포: http://webstandards.or.kr/html5