실전 웹 표준 가이드

The Practice Guide for Development
based on Web Standards

2005.12

한국소프트웨어진흥원

Web Standard Practice by KIPA

Edited by Seokchan Yun, Jungsik Shin, Hyeonseok Shin, Sungno Lee

목차

그림 목차

저자 소개

윤석찬 (E-mail) channy@gmail.com (Blog) http://channy.creation.net/blog

신현석 (E-mail) hyeonseok@gmail.com (Blog) http://hyeonseok.com/blog

이성노 (E-mail) eouia0819@gmail.com (Blog) http://eouia0.cafe24.com/blog

신정식 (E-mail) jshin@i18nl10n.com

서론

1993년 4월 22일 미국 일리노이 대학에서 일단의 학생들이 개발한 모자이크(Mosaic)라는 작은 공개 소프트웨어 웹브라우저는 오늘날 웹이 전 세계에 영향을 끼치게 하는 혁명적인 첫 출발이었다. 이 웹브라우저의 근본 아이디어를 기초로 마이크로소프트와 넷스케이프사 등에서 개발한 유수의 웹브라우저 들이 나와 각축을 벌이고, 이제는 넷스케이프 네비게이터가 시장 선점에 실패하면서 MS사의 인터넷 익스플로러가 시장 지배적인 위치에 들어서 있다.

브라우저 시장점유 전쟁 동안 서로간의 웹브라우저에 배타적인 기술을 도입하던 나머지 똑 같은 웹페이지가 다르게 보이고 서로에서 구현하지 못하는 기술 때문에 많은 혼란을 겪어 왔다. 이러한 상호 호환 미성숙으로 말미암아 웹기술이 혼란 상태에 있어 왔던 것이 사실이다. 현재에는 이미 마이크로소프트가 IE로 시장 지배력을 넓히고 있는 이 시점인데다 더 이상 웹브라우저가 신기한 도구이지 않기 때문에, 다양한 인터넷 환경에서 어떠한 웹브라우저가 가장 최적의 구현을 제공하느냐가 관건이 되었다.

그림 1 역사상의 웹 브라우저

그러나 넷스케이프사가 자사의 웹브라우저 소스를 공개 소프트웨어로 전환시키면서 탄생한 모질라(Mozilla) 재단은 전 세계 개발자들의 노력에 힘입어 경량의 오픈 소스 웹브라우저인 파이어폭스(Firefox) 1.0을 내 놓으면서, 출시 된지 몇 개월 만에 인터넷 익스플로러의 브라우저 점유율을 90% 아래로 끌어 내리고 넷스케이프 이후 사상 최초로 10% 점유율을 바라보고 있다. 노르웨이의 Opera 브라우저는 가볍고, 각종 OS 플랫폼과 표준 호환성이 뛰어난 기능을 무기로 시장을 개척해 나가고 있다. 뿐만 아니라 유닉스 기반 오픈 소스 프로젝트 KDE(K Desktop Environment)에서 개발한 KHTML 브라우징 엔진은 애플이 사파리 브라우저에 채택되었고 PDA 및 Embedded Linux 등 소형 기기에 탑재될 수 있는 가능성으로 웹 브라우저의 다양한 엔진 전쟁이 예고 되고 있다.

이러한 현재 상황을 두고 많은 웹 서비스 개발자들이 마치 90년대 중반의 브라우저 전쟁 중에 있었던 비호환성을 고려해야 하는 크로스 브라우징 문제가 다시 대두되는 게 아닌가 염려하고 있다. 즉, 다양한 브라우저 지원이 결국 시간과 비용을 들여야 하는 문제이며 소수 브라우저에 굳이 비용을 투여할 필요가 있는가 하는 오해가 존재하는 것이다.

본 가이드에서는 현대 웹브라우저들이 과거와는 달리 웹 표준을 준수하고 있고 계속적으로 이를 지원하기 시작했기 때문에 이를 배우고 따르기만 하면 보다 저렴한 비용으로 홈

페이지를 구축 및 유지 보수 할 수 있다는 것을 보여 줄 수 있음을 보여 주고자 한다. 또한, 웹의 근본적인 목표인 기기 및 운영 체제 독립적인 보편적인 웹을 만들 수 있도록 하고 있다. 전 세계의 많은 웹사이트들이 이미 웹 표준에 기반한 홈페이지 제작 방식을 도입하고 있으며 이를 통해 매우 효과적인 웹서비스 체계를 갖추기 시작하고 있다.

이 가이드의 목적

1989년 웹(World-wide Web)을 처음 발명한 팀 버너스 리(Tim Berners-Lees)는 CERN 연구소의 수 천명 연구자들이 이기종 OS와 개발 환경에서 정보를 공유할 수 있도록 기종이나 환경과 상관없이 어떤 컴퓨터에서도 정보 자원에 접근할 수 있는 웹을 만들게 되었다. 이러한 웹의 근본 취지에는 보편적 디자인(universal design), 보편적 접근(universal access) 개념이 뿌리내려 있으며 웹을 사용하고 접하는 다양한 환경과 사람들을 위해 공통적인 정보 소통 통로를 만드는 것은 정보의 가치와 비례하여 중요하다고 하겠다. 이러한 꿈을 이루기 위해 Tim과 CERN은 웹의 발명품을 아무나 사용할 수 있는 무제한으로 특허를 사용할 수 있도록 하였다.

우리들 중에는 노인, 장애인, 어린이 등의 다양한 계층과 윈도우, 매킨토시, 리눅스 등 다른 OS, IE와 파이어폭스, 오페라 등 다른 브라우저를 사용하는 사람이 존재한다. 또한, 향후 TV 브라우저, PDA 등의 다양한 휴대 기기를 사용하는 사람과 시력이 약해서 화면을 확대해 봐야 하는 사람들이 있다. 다양한 사용자들이 어떻게 하면 편하게 웹을 이용할 수 있을까? 어떻게 하면 저비용으로 접근성 문제를 해결할 수 있을지를 고민하여 실마리를 풀어 나가야 한다. 특정 플랫폼의 사용자만이 향유할 수 있는 웹을 만들면 만들수록 더 많은 정보가 웹으로 쏟아져 나오기 때문에, 장애인, 노인, 외국인, 매킨토시 사용자, 리눅스 사용자는 점점 더 소외 계층으로 전락하여 이른바 정보화의 폐해가 드러나게 되기 때문이다.

이러한 문제를 해결하기 위한 구체적인 방법은 무엇일까? 본 가이드에서 설명하고자 하는 핵심 요지는 아래에서 말하는 세 가지로 요약 할 수 있다.

미션1. 웹 표준을 지켜라

전 세계적인 웹 기술 표준을 주도하고 있는 W3C의 HTML4.1, XHTML1.0, CSS1/2, DOM 등의 구현 스펙이 매우 상세하고 이를 지원하는 브라우저들이 계속 늘어 남에 따라 더 이상 웹페이지가 다르게 보이거나 동작하지 않는 현상은 거의 사라지게 되었다. 다만 웹브라우저간 이종 기능이 아직은 상존하고 있고 예전에 개발되어 사용된 오래된 브라우저 사용자들의 불편함을 고려해 같은 기능이라도 호환 가능하도록 해 주는 표준을 통한 웹페이지 제작이 요구되고 있다.

그러나 이것이 웹브라우저에서 지원하는 버전 호환성 유지(backward compatibility)라는 괴물과 충돌하게 된다. 일반적으로 프로그램을 개발할 때, 버전이 올라가면 갈수록 새로운 기능을 추가하고 이전 기능은 폐기하게 된다. 그러나 사용자의 측면에서는 예전 기능을 계속 유지해 주어 개발 호환성을 유지해 줄 필요성이 생긴다. 즉, 버전 호환성 유지는 예전에 사용되는 기능이나 태그를 그대로 사용하도록 해 주는데 이 때 사용된 비표준 문법들이 계속 확대 재생산 되어 결국 접근성에 심각한 위해를 주게 된다. 여기에는 취약한 우리나라 웹 생산 시스템의 문제도 있다. 주로 나모 웹 에디터, 드림 위버와 같은 저작 도구에

의존하여 표준을 무시하고, 그냥 남의 코드를 따다가 적당히 익스플로러에서만 돌려보고 개발을 끝내는 풍토와 그런 정보 가공자들을 계속 양산 하는 교육 시스템에 문제가 있다.

이와 반대로 XHTML 1.x이나 HTML 4.x 표준에 맞추어진 문서는 99% 접근성이 높은 사이트들이다. 기존에 흔히 사용되는 table 구조를 div 바꾸고 font, b 같은 태그들을 스타일시트(CSS)로 사용하게 되면, HTML 코드 양은 약간 과장해서 반 이하로 줄어든다. 구조와 표현이 엄격히 분리되면, 사이트의 로딩 속도도 빨라지며, 코딩과 유지 보수의 효율성은 두 배로 늘어난다. 표준을 지킨 사이트에서는 오히려 코드의 양이 줄고 속도가 늘어나며 재개발 효율성이 증대 된다.

미션2. 구조와 표현, 동작을 분리하라

웹 표준에서 HTML과 함께 중요한 또 다른 요소인 스타일시트(CSS)는 단순히 링크의 색상, 글자 모양 바꾸는 정도만 할 수 있는 것이 아니고, 문서의 배치, 여백 조정, 색깔, 요소 자체의 성격 변화, 클래스를 통한 디자인의 일관성 확보, 서로 다른 미디어에 따른 최적화된 디자인 템플릿 적용 등 이루 말할 수 없이 많은 역할을 할 수 있다.

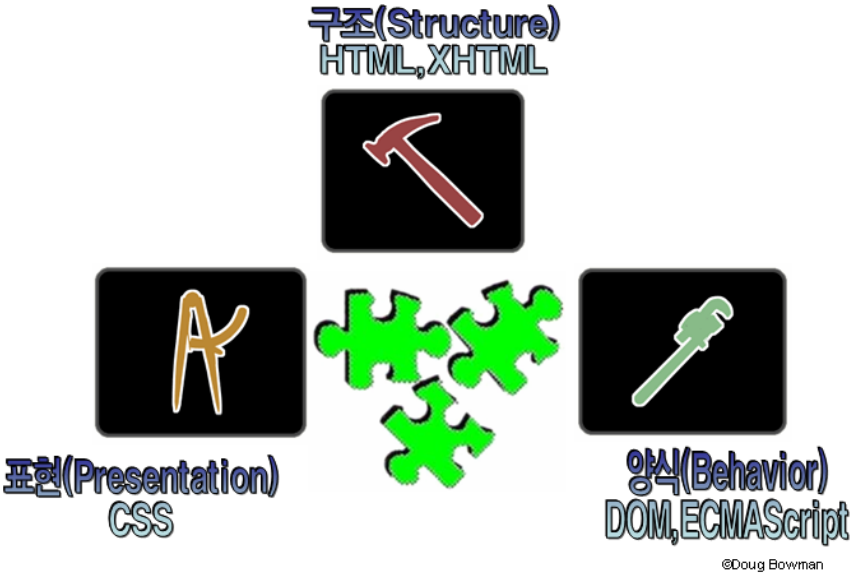


그림 2 웹 표준에서 구조, 표현, 행동의 분리

HTML 에서는 철저하게 구조화된 마크업만을 사용하고, 모양이나 디자인에 관한 것은 CSS로 완전히 분리함으로써, 구조는 변하지 않은 채 여러 가지 디자인을 적용한다거나, 상황에 따라 쉽게 디자인을 변경하는 것이 가능해진다. 또한, 똑같은 디자인 템플릿에 다른 내용을 담는 여러 가지 문서를 만드는 것도 가능해진다. 또는 디자인에 전혀 영향을 주지 않고 문서의 내용을 바꾸는 것이 쉬워지기 때문에 장애인의 접근성에 엄청난 도움을 준다. 예를 들어, 한 개의 HTML 문서에 시각 장애인용 CSS와 텍스트용 CSS 그리고 기존의 CSS 등 세 개의 스타일시트를 만들어 변경해 줌에 따라 문서의 구조가 내용과 섞이지 않고 제공 될 수 있다.

우리나라 대부분의 웹사이트들이 문서의 구조적인 요소인 h1, ul, strong, blockquote, cite 등 보다는 font, table, br 등의 요소가 의미적인 내용과 섞여 있어 문서 중에서 도대체 어

떤 것이 중요한 지, 제목에 해당되는 지, 강조할 것과 인용된 것이 어떤 것이지 하는 것을 이해할 수 없게 된다. 문서의 내용이 구조화 되어 있지 않다는 것은 결국 장애인용 보조기기, 또는 PDA, TV, 음성 브라우저 등을 통해서도 접근할 수 없거나, 접근했을 때에 바른 결과를 얻어 낼 수 없다는 이야기가 된다. 디자인 요소가 CSS로 완전히 분리되면, 의미 있고 구조화된 내용만 남게 되며, 문서의 표현에도 제약 사항이 발생하지 않게 되는 것이다. 최악의 경우, CSS가 제거되어도 의미를 이해하는 데에는 아무런 문제가 없게 되며 이것이 웹 접근성 문제를 해결하는데도 도움을 준다.

또한, 구조와 표현에서 행동을 분리하는 것도 매우 중요하다. 웹 문서의 내용을 자바 스크립트를 통해 동적으로 사용하고, 사용자의 반응을 주고 받는 행동(Behavior)도 중요한 부분으로 인식되고 있다. 이러한 클라이언트 스크립팅을 기본으로하는 동적인 행동 양식이 사실 접근성 및 사용성을 저해하고 및 복잡성을 높이는 요소로 생각되어 왔다. 그러나, 최근 구조에서 행동 양식을 분리하는 다양한 방법론이 제안되면서 접근성을 해치지 않으면서 코드를 간단하게 하는 다양한 방법들을 적용할 수 있게 되었다. 자바스크립트를 홈페이지에 마구 쑤셔 넣는 개발 방식에서 변화할 수 있다는 것이다.

미션3. 최소한의 디버깅을 거쳐라.

W3C에서 제시한 HTML 혹은 XHTML 표준을 지키고, CSS를 통해 구조와 표현을 분리하였다 하더라도 이것을 검증해 볼 수 있는 방법이 없다면 역시 그 문제 해결이 쉽지 않을 것이다. 다행히 W3C에서는 이러한 문제를 검증 해주는 유효성 검사(Validation) 도구들이 제공되고 있다. doctype에 규정된 문서 형식에 따라 유효한 코드가 사용되었는지를 알 수 있게 해준다. (http://validator.w3.org)

HTML과 XHTML 문서뿐만 아니라 CSS 역시 자신이 사용한 표현식과 문법이 제대로 되어 있는지 알려 주는 도구가 존재 한다. 처음 이 검사기를 돌려 보면 수 많은 문제들이 나오게 될 수도 있다. 그러나 대부분의 문제들은 요소를 따옴표 등으로 묶지 않았거나, ?, & 같은 특수 문자를 특수 코드로 쓰지 않는 등 반복되는 몇 가지 실수를 포함 하고 있으므로 간단한 수정 만으로 유효성 검사를 통과 할 수 있다.

또한, 문서 안에 들어 있는 요소를 객체화 시켜 사용하는 DOM(Document Object Method)과 이를 이용하는 자바 스크립트(JavaScript) 등을 사용하는데 있어도 표준 문법을 사용했는지 확인하는 과정이 필요하다. 오픈소스 브라우저인 Mozilla Firefox의 탑재된 자바 스크립트 디버거 만으로도 인터넷 익스플로러와 공통으로 생기는 문제점을 발견 해결할 수 있고, 비표준으로 사용된 문법을 판별해 낼 수 있다. 그러나 웹사이트 가공자들이 이러한 디버깅 과정을 빠뜨리고 납품을 하거나 완성하는 경우가 대부분이기 때문에, 웹 사이트 관리자들이 검수(Quality Assurance) 과정에서 이러한 유효성 통과와 표준 문법 사용 검증 결과를 첨부하도록 가이드 한다면 보다 좋은 품질을 가진 결과물을 얻을 수 있을 것이다.

그림 3 W3C HTML Validator의 실행 모습

미션4. 효율적인 웹 개발 방법론을 가져라.

웹을 생산하는 시스템 안에는 웹 기획자와 웹 개발자라는 직군이 함께 존재한다. 웹 생산 공정에서 웹 디자인까지 이들은 삼각 구도를 이루고 있으며, 서로 간의 업무 역할이나 관심 영역에서 확연한 차이가 드러난다. 따라서 웹사이트를 만드는 이들이 서로 협력하지 않는 다면 프로젝트는 산으로 가며 따라서 웹사이트는 제대로 된 품질을 지킬 수 없다.

웹사이트를 만드는 일은 기획->디자인->개발이라는 컨베이어 벨트 위에 놓은 물건과 같다. 이게 공장이라면 컨베이어 벨트만 통과하면 완성품이 하나 나오지만, 문제는 이 컨베이어 벨트에 탄 물건은 사용자 테스트나 요구 사항 변경에 따라 이 컨베이어 벨트를 몇 번씩 옮겨진다는 데 있다. 이 과정에서 누가 컨베이어 벨트에 새로 올려 놓았느냐에 따라 서로 간에 피해 의식이 생기게 된다.

서로를 탓할 순 없고 맘속으로만 삭일 수 밖에 없다. 문제를 해결하는 방법은 서로 영역을 나누어 공동 작업을 하는 것이다. 마치 시계의 부품을 각자 작업해서 조립만 될 수 있도록 하는 것이다. 그러기 위해서는 웹 표준을 통한 공정이 매우 생산적이다.

웹 표준에서 말하는 웹은 내용을 담는 구조(HTML, XHTML), 표현(CSS), 양식(자바스크립트, 서버 측 개발)을 분리시켜 개발하도록 권장하고 있다. 구조는 단순한 HTML로 이 페이지에서 넣고 싶은 내용만을 간추릴 수 있다. 간단한 템플릿을 사용하면 기획자들이 쓰는 스토리 보드처럼 레이아웃을 만드는 것이 가능하다.

HTML에 표현과 내용을 분리하지 않고 다 집어 넣는 풍토가 바뀌면 기획자와 디자이너, 개발자가 같은 시간에 같은 일을 하는 것이 가능하게 된다. 기획자가 기획안을 넘기고 다른 기획에 투입되며 디자이너는 디자인만 넘기고 다른 프로젝트에 투입되고, 개발자가 코딩만 하고 다른 개발을 하게 되는 일이 없어지면 상호간 하나의 팀으로서 같은 일을 같은 시선으로 바라볼 수 있는 것이다.

기획자와 개발자, 그리고 디자이너들은 서로 다른 사고 방식과 생각을 가진 사람들이다. 기획자들은 비즈니스를 생각하고 디자이너들은 시각적인 요소를 더 중시한다. 개발자들은

그들이 사용하는 기술에 관심이 있는 것은 두말할 나위가 없다.

그럼에도 불구하고 그들을 모두 엮는 것은 웹(Web)이라는 테두리이다. 웹 기획자가 아니라면 그들이 HTML을 이해할 필요도 없고, DB와 서버를 알 필요도 없다. 따라서 이러한 직군에 근무하는 사람들은 웹에 대한 의미와 인식을 같이하는 공감대를 가질 필요가 있다.

웹 표준에 대한 오해

웹 표준과 웹 접근성을 강조하다 보면 사람들로부터 다음과 같은 질문을 받게 된다. 웹 표준과 접근성을을 지키는 것이 어떤 혜택을 얻을 수 있는지 알아 본다.

화려하고 세련된 웹페이지를 만들 수 없다?

흔히들 우리나라 웹페이지는 외국보다 역동적이고 화려하게 하는 것을 좋아하기 때문에 접근성을 강조하면 그것을 달성하기 어렵다고들 한다. 그러나 어떠한 접근성 지침에서도 그림이나 멀티미디어 쓰지 말라고 하지 않으며, 그림을 못 보는 사람들을 위해 대체 텍스트를 함께 넣으라는 것뿐이다. 레이아웃 장식을 위해 그림을 많이 사용하는 경우, CSS의 배경(background)으로 그림을 넣어버리면 가능하며 플래쉬를 이용해 메뉴를 구성 하는 경우, 대체 텍스트 메뉴를 플래쉬를 표현하는 object 사이에 포함시켜 주기만 해도 된다.

접근성 높은 사이트가 온통 텍스트로만 구성된 지루하고, 멋없는 사이트라고 생각하는 것은 잘못된 편견이다. 접근성 지침에서 결코 디자인과 타협하거나 상충되는 부분은 거의 없으며, 이미 CSS를 활용한 외국의 사이트들은 접근성이 높으면서도 화려하고, 깔끔한 사이트가 대부분이다. 우리 나라는 오히려 웹 서비스 가공자들에 대한 재교육 부재와 기존의 비효율적이고 표준을 활용하지 못하는 작업 행태에서 문제가 시작 되는 것이다.

접근성이 높은 사이트는 비용이 훨씬 증가한다?

물론 기존에 접근성을 전혀 고려하지 않고 만든 사이트를 고치려면 상당한 비용이 들어가는 것이 사실이다. 그러나 대부분의 웹사이트들이 여러 번 리뉴얼이라는 과정을 통해 계속 변경하고 있으며, 이러한 리뉴얼을 시작할 때 처음부터 접근성을 염두해 두고 엄격한 웹 표준에 입각해 개발하면, 결코 비용이 많이 들어가지 않는다.

이 렇게 개발된 접근성이 높은 사이트를 다시 재개발하는데 드는 효율성도 매우 높다. 왜냐하면, 구조와 표현을 분리하여 만들어져 있으므로, 컨텐츠 담당자는 오로지 구조와 내용에만 신경을 쓰고, 디자인 담당자는 내용을 표현하기 위한 외양적 디자인에만 관심을 기울일 수 있기 때문이다. 프로그래머는 이미 HTML과 CSS로 UI가 분리되어 있으므로, 복잡한 HTML 코드를 이해하고 다뤄야 하는 대신 프로그램 코드에만 신경 쓰면 된다. 즉, 내용이 바뀌면 디자인까지 다 뜯어고쳐야 한다거나, 디자인이 바뀌면서 내용이 타협을 해야 한다거나 하는 일들이 줄어들게 된다. 그리고 한 번 이렇게 만들어진 사이트를 다른 사람이 유지 보수를 할 때에도 소스 코드도 훨씬 이해하기가 쉽고 간결해서 완전 재개발 해야 하는 경우가 발생하지 않아 유지 보수 비용도 줄일 수 있다.

특수 계층을 위한 별도의 사이트가 필요하다?

가뜩이나 다수의 논리에 의해 소수는 쉽게 무시되어버리는 우리 나라 사회에서 웹 접근성

을 이야기하면 효율과 속도를 중시하는 풍조에서 누가 과연 몇 퍼센트나 있을 지 모르는 장애인과 매킨토시, 리눅스 사용자와 비 IE 사용자를 배려할 수 있을 것인가라는 의문을 제기한다. 그리고는 마치 그들을 위해 뭔가 선심이라도 써야 할 듯이 별도의 사이트를 만들고 비용과 인력을 들여야 할 듯이 말한다. 웹에서 표준안의 사용과 접근성 가이드에 대한 준수 만으로도 이들 계층들을 위한 별도의 웹사이트는 필요치 않다. 별도의 사이트를 만드는 것이야 말로 오히려 비용을 증가 시켜 효율성을 떨어 뜨리는 것이다.

접 근성이 높은 웹사이트는 결코 이들 특수 계층에게만 좋은 것이 아니다. 사용 편의성이 높아지고, 문서가 분명하며 이해하기 쉽게 되면 아이들이나 노인들에게도 도움이 되며, 이것을 개선하여 작업하는 웹 사이트 운영자도 편해 진다. 곳곳에 설명 도구(tooltip)들이 생겨서 일반인들에게도 웹을 친근하게 사용하는 데에 도움이 되며, 그림과 멀티미디어 요소가 의미있는 대체 텍스트를 달게 하면 검색 엔진이 그림과 멀티미디어를 일반인들이 검색하는 데에 큰 도움을 준다.

웹 표준이란 무엇인가?

전 세계적인 웹 기술 표준을 주도하고 있는 W3C의 HTML4.0, XHTML, CSS1/2 등의 구현 스펙이 매우 상세하고 이를 지원하는 현대적인 브라우저들이 계속 늘어남에 따라 더 이상 웹페이지가 다르게 보이거나 동작되지 않는 현상은 거의 사라지게 되었다. 옛날 웹브라우저간 이종 기능이 아직은 상존하고 있기 때문에 오래된 브라우저 사용자들의 불편함을 고려해 주는 상호 호환성(Cross Browsing)과 최신 웹 표준 기술 적용 그리고 접근성 높은 웹페이지를 통해 향후 표준 기술에 적합하게 만드는 상위 호환성(Forward Compatibility)가 현재 웹 서비스 제공자들의 공통된 숙제가 되고 있다.

이것은 하위 버전 호환성 유지(Backward Compatibility)와 구분해야 한다. 일반적으로 프로그램을 개발할 때, 버전이 올라가면 갈수록 새로운 기능을 추가하고 이전 기능은 폐기하게 된다. 그러나 사용자의 측면에서는 예전 기능을 계속 유지해 주어 개발 호환성을 유지해 줄 필요성이 생긴다. 웹브라우저에서 하위 버전 호환성 유지는 예전에 사용되는 기능이나 태그를 표준 태그로 치환해 주는 것이다. 이를 통해 대부분의 웹디자이너는 예전 지식에 따라 웹페이지를 코딩해 주어도 그대로 구현되는 것으로 생각하게 되는 것이다. 그러나, 하위 버전 호환성 유지는 웹브라우저의 벤더에 따라 지원 가능 정도가 약해 질 뿐만 아니라 웹 표준 기술에 대한 지식 습득을 가로막는 장애가 된다.

이에 반해 상호 호환성은 표준 웹 기술을 채용하여 다른 기종 혹은 플랫폼에 따라 달리 구현되는 기술을 비슷하게 만듦과 동시에 어느 한쪽에 최적화되어 치우치지 않도록 공통 요소를 사용하여 웹페이지를 제작하는 기법을 말하는 것이다. 또한, 지원할 수 없는 다른 웹브라우저를 위한 장치를 만들어 모든 웹브라우저 사용자가 방문했을 때 정보로서의 소외감을 느끼지 않도록 하는 방법론적 가이드를 의미하는 것이다. 또한, 장기적인 웹 표준을 지원하는 상위 호환성은 미래에 어떠한 웹 브라우저나 단말 장치가 나오더라도 웹을 이용할 수 있다는 측면에서 매우 중요하다고 하겠다.

웹 표준이 만들어 지는 방법

웹사이트에 적용하는 HTML, CSS, 자바스크립트 같은 것은 어디에서 정해져서 사용되는 것일까? 이 같은 승인된 개방형 인터넷 표준은 즉 World Wide Web Consortium (W3C,

http://www.w3c.org) 에서 만들어 진다. W3C는 1994년 10월 미국의 MIT 컴퓨터 과학 연구소(MIT LCS), 정보 수학 유럽 연구 컨소시움(ERCIM), 그리고 일본의 게이오 대학이 연합하여 만들어진 국제적인 웹 기술 표준 기구이다.

그림 4 웹 표준을 제정하는 표준화 기구, 웹 컨소시움(http://w3.org)

언뜻 보기에는 연구 기관으로만 이루어진 것 같으나 웹과 관련된 510여 개의 국제적인 다국적 IT 기업체가 참여하여 자사의 하드웨어와 소프트웨어에 웹 표준 기술을 탑재하거나 자사의 기술을 표준화 하고자 하는 치열한 전투장 이기도 하다. W3C의 역할은 정보, 의견 교환, 아이디어 창출, 독립적 사고, 그리고 공동의 이해를 위하여 명세, 가이드 라인, 소프트웨어, 그리고 도구 및 규칙 등의 표준안을 제정함으로써 웹의 모든 잠재력을 이끌어 내는 것이다.

웹에 관련한 표준에는 우리가 흔히 말하는 표준(Standard)은 존재하지 않으며, W3C의 토론을 통해 나온 권고안(Recomendation)이 가장 최상위 이다. 표준의 종류에는 제안된 표준(Draft), 작업하는 표준(Working Draft, WD), 확정될 권고안(Candidate Recommendation, RC), 확정된 권고안(Recommendation)이 있다.

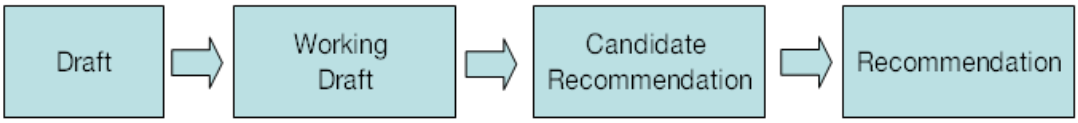


그림 5 웹 표준이 정해 지는 순서

권고안을 제정하는 방법은 1) 어떤 기능을 Draft로서 제안하고, 2) 드래프트를 실제로 적용할 수 있게 기술적인 작업을 하고(Working Draft), 3) 이를 다시 논리 오류가 없는지, 실제 하드웨어에서 지원이 가능한지를 살피고, 4) 정식 권고안이 되기 전에 기업체에 공개하여 토론을 거친 후(Recommendate Candidation), 5) 마지막으로 권고안(Recommendation)을 확정한다.

웹브라우저 중 모질라 파이어폭스, 오페라 등은 문서로서 확정된 권고안(Recomendation) 등 다양한 표준 가운데 확정된 표준을 지원한다. 다시 말해 모질라는 HTML 4.0도 지원하지만, HTML4.01을 더 잘 지원해 준다. 특히 W3C 전용 브라우저인 Amaya 브라우저가 했던 표준의 기술 지원 시험을 요즘에는 모질라에서 하고 있어 더욱 빠르게 표준 기술이 적용되고 있다. 모질라 계열 제품들의 최신 버전은 XHTML, CSS 1/2를 모두 지원하고 있다. 오페라도 비슷한 정도로 표준을 지원해 주지만, 빠른 속도를 유지하기 위해 MathML 등을 제대로 해석하지 못하기도 한다.

인터넷 익스플로러는 지원하는 표준의 종류만을 보자면, 다른 두 가지 브라우저보다 더 뛰어나다. 그러나, 지원하는 표준이 권고안(Recomendation)이 아니라는 데 문제가 있다. IE는 Microsoft가 제안했던 내용만을 지원하는데, 권고안에 자신들이 제안했던 내용이 적고, 권고안 후보나 작업안 또는 기초안에 자신들이 제안한 내용이 많다면 그것을 지원한다. 대표적인 예가 HTML 4.0과 XHTML 1.0/1.1이다. HTML 4.0은 거의 모든 부분에서 마이크로소프트의 의견이 반영되어 있다. 그럼에도 불구하고 HTML4.0의 후속 버전인 XHTML 1.0/1.1은 제대로 지원하지 않고 있다. 왜냐 하면, HTML을 모듈화하면서 마이크로소프트의 의견이 상당 부분 표준에 채택되지 못했기 때문이다. 또한 CSS Level 2(흔히 CSS2) 지원도 미흡하다. 그러나, 마이크로소프트는 CSS3 표준안에 열심히 참여하고 있다. 웹브라우저가 W3C의 권고안을 지원하는 데는 이러한 복잡한 관계가 얽혀 있다.

이에 반해 모질라 파이어폭스 1.5의 최신 버전은 XTHML 1.0을 모두 지원할 뿐만 아니라 Draft상태에 있는 CSS2.1 전체 기능과 CSS3의 일부 기능을 이미 구현했다. 인터넷 익스플로러 6.0이 출시된 지 4년 만에 IE팀을 새로 꾸린 마이크로소프트는 IE7 버전에 CSS2에 대한 지원을 강화하기로 하는 등 표준 지원에 대해 모든 브라우저가 활발히 움직이고 있다.

웹 표준 스펙 소개

지금부터는 W3C에서 제공하는 각종 웹 표준 권고안에 대해 간단하게 살펴본다.

(X)HTML (eXtensible Hypertext Markup Language)

HTML(Hypertext Markup Language) 는 웹페이지를 표시하는데 기본 언어로서 사용된다. 웹 컨텐츠의 내용은 표준 HTML 포맷으로 적용해야 하며 정보가 독점적인 고유 포맷으로 제공되는 경우, HTML 포맷도 제공되어야 한다. 브라우저 호환성은 모든 경우에 있어 고려되어야 하며, 웹사이트는 단일 웹 브라우저에 맞추어 제작되어서는 안되며, 클라이언트 그룹에 의해 빈번하게 사용되는 웹 브라우저에서 올바르게 작동해야 한다.

최신의 HTML 표준은 4.01이지만 HTML을 XML과 결합한 XHTML이 권고안으로 나와 있다. HTML2/3와 달리 최신 XHTML 표준은 <font>, <b>, <i> 같은 표현 요소들을 배제하고, 태그를 모두 닫도록 권고하는 등 정확한 문서 규격을 요구하고 있다. 이것은 손으로 코딩을 하는 게 아니라 점점 전문적인 저작 도구를 사용함에 따라 구조적인 HTML템플리트를 생성하고 스타일(Cascade Style Sheet, CSS)을 관리함으로서 비 전문 설계자도 웹 페이지를 손쉽게 제작 관리 할 수 있게 해 준다.

CSS(Cascading Style Sheets)

CSS는 사용자 정의의 디자인 속성, 즉 글꼴, 크기, 색상, 이벤트 등을 지정할 수 있으며 CSS를 사용한 모든 페이지는 기존 버전과의 호환성 되게 어떤 브라우저에서도 내용을 열람할 수 있다. CSS를 이용하여 설계자는 서로 다른 화면 해상도와 브라우저 상에서, 테이블 없이도 동일하게 보여질 수 있는 페이지를 생성할 수 있다. 단 IE4.0 이하와 넷스케이프4 이하의 오래된 웹브라우저에서는 CSS를 지원하지 못한다. CSS를 사용하여 생성한 페이지와 템플리트는 다양한 브라우저, 화면 해상도 및 액세스 기술을 사용하여 테스트하여야 하며, 최신 시스템 사용자가 아니더라도 적합한 접근이 보장되어야 한다.

XML(eXtensible Markup Language)

XML(eXtensible Markup Language)은 HTML이나 CSS로서 표현되지 못하는 영역을 DTD를 이용하여 정의하여 사용자 정의의 태그를 생성하여 제작할 수 있는 메타 마크업 언어이다. XML 사용 분야를 검토하여 적절한 용도에 맞게 사용하여야 한다. XML이 고려되는 애플리케이션은 사용자가 필요한 정보를 얻기 위해 하나 이상의 데이터베이스와 상호 작용할 필요가 있는 경우, 작업이 사용자에게 전달되어 사용자가 자신의 기록 혹은 문서에 액세스할 것이 예상되는 경우, 서로 다른 세트의 데이터가 서로 다른 사용자에게 디스플레이 되어야 하는 경우, 정보 검색 및 디스플레이와 관련하여 사용자 선호 프로파일을 구축해야 할 필요가 있는 경우, 각 개인이 스타일 시트를 사용하여 다양한 포맷으로 문서를 갱신해야 할 필요가 있는 경우에 사용 가능하다. XML은 다양한 인터넷 비즈니스 환경에 손쉽게 적응 가능하여 웹 표준 분야에서 가장 활발한 표준 제정 활동이 이루어 지고 있다.

DOM(Document Object Model)

DOM(Document Object Model)은 웹페이지에 표현되는 모든 속성에 대해 객체화 하여 이를 자유 자재로 사용할 수 있도록 만든 것이다. document, from, window 등 각각의 속성을 객체화 하여 트리 구조로 형상화 하여 이에 대한 이벤트 처리 같은 것이 가능하다. DOM에는 크게 W3C DOM과 MS DOM이 있는데, IE6.0은 아직 하위 버전 호환성을 위해 MS DOM을 지원하고 있지만, IE6.0 이전 브라우저를 제외하고는 거의 모든 브라우저가 표준 W3C DOM을 지원한다.

ECMAScript

자바 스크립트는 W3C 표준으로 제정된 것은 아니다. 또한, 모든 웹 브라우저 사용자가 자바 스크립트를 볼 수 있는 것은 아니다. 특정 방화벽은 자바 스크립트가 통과하는 것을 허용하지 않는다. 그럼에도 자바 스크립트는 DOM이 표준화 되면서 웹 브라우저에 널리 쓰이고 있다. 주의할 점은 클라이언트 측 스크립트는 여러 브라우저에서 폭 넓게 검사되어야 한다. 핵심 기능은 자바 스크립트에 의해서만 제공되어서는 안 된다. 또 자바 스크립트는 주석 코드를 사용하여 비 호환성의 웹 브라우저로부터 숨겨져야 한다. 자바 스크립트는 HTML 문서의 Head 내에 위치해야 제대로 동작한다 따라서 문서의 Body 내에 자바 스크립트를 위치시키는 것은 피해야 한다.

자바스크립트의 경우, 넷스케이프사가 ECMA라는 표준 기구로 제안하여 채택된 바 있어

ECMA -262 표준안(http://www.ecma-international.org/publications/standards/Ecma-262.htm)을 공부하면 된다. ECMASCript는 IE6.0, Firefox 1.0, Safari 1.0, Opera8에서 거의 100% 지원하고 있다.

웹 표준 목록

웹 표준을 지킨다고 하는 것은 W3C의 표준안을 지킨다는 것을 말한다. W3C의 표준 활동을 지속적으로 살펴보고 표준 권고안을 공부할 필요가 있는 것이다. 또한 웹 브라우저에 따라 지원 정도가 다르므로 부록에서 제공하는 호환 차트(Web Standard-Broweser Compatiblity Chart)를 통해 공부 해야 한다. 아래는 웹사이트 개발에 필요한 주요 표준안들에 대한 목록이다.

- [HTML 4.01] "HTML 4.01 Recommandation", David Raggett, Arnaud Le Hors, Ian Jacobs, http://www.w3.org/TR/1999/REC-html401-19991224 , HTML 4.0의 수정 표준 (http://www.w3.org/TR/1998/REC-html40-19980424)
- [XHTML1.0] "XTHML 1.0 Recommandation", 26 January 2000, revised 1 August 2002, Steven Pemberton http://www.w3.org/TR/2002/REC-xhtml1-20020801 최신 버전: http://www.w3.org/TR/xhtml1
- [XHTML 1.1] " Module-based XHTML", 31 May 2001, Murray Altheim, Shane McCarron http://www.w3.org/TR/2001/REC-xhtml11-20010531
- [CSS1] "CSS, level 1 Recommendation", B. Bos, H. Wium Lie, eds., 17 December 1996, revised 11 January 1999. CSS1 권고안: http://www.w3.org/TR/1999/REC-CSS1-19990111 . CSS1 최신 버전: http://www.w3.org/TR/REC-CSS1 .
- [CSS2.1] "CSS, level 2.1 Draft", Bert Bos, Tantek Çelik , Ian Hickson, Håkon, Wium Lie eds., 12 May 1998. CSS2 드래프트: http://www.w3.org/TR/2005/WD-CSS21-20050613 . CSS2.1 최신 버전: http://www.w3.org/TR/CSS21.
- [DOM Level 1, Level 2, Level 3] "Document Object Model (DOM) Level 1,2,3 Recomandation", 최신 버전 http://www.w3.org/DOM/DOMTR
- [XML] "Extensible Markup Language (XML) 1.0.", T. Bray, J. Paoli, C.M. Sperberg-McQueen, eds., 10 February 1998. http://www.w3.org/TR/1998/REC-xml-19980210 .[XML 1.1] XML 1.1 (http://www.w3.org/TR/xml11
- [HTTP] http://www.w3.org/Protocols/, HTTP 1.0, 1.1을 비롯한 여러 가지 표준 웹 프로토콜에 대한 IETF(http://www.ietf.org) 표준이 있음.

웹에서 쓰이는 표준 중 HTTP등과 같은 프로토콜 표준은 W3C에서 만드는 것이 아니다. W3C 뿐만 아니라 인터넷에서 사용되는 모든 표준을 RFC라는 문서로 규정하는 IETF도 중요한 표준 기구이다. W3C 표준뿐 아니라 IETF에서 제정하는 표준 가운데 적어도 HTTP와 MIME에 대한 표준은 매우 중요하다. 왜냐하면 우리가 만드는 웹 문서를 인터넷으로 전달하는 기본적인 방법에 대한 것이기 때문이다.

다행히 이들 웹 표준 문서 중 일부는 W3C 한국사무국(http://w3c.or.kr)을 통해 한국어 번역(http://w3.org/2003/03/Translations/byLanguage?language=ko)을 접할 수 있다. 또한 이들 중 일부는 한국정보통신표준협회(TTA)의 국내 표준으로 정해진 것도 있다.

- TTA, TTAS.KO-10.0085, "웹 구축 지침서", 1998-10
- TTA, TTAS.W3-XHTML1.0, "XHTML 1.0 표준", 2001.12.

- TTA, TTAS.W3-DOM "문서 객체 모델 레벨 1", 2003.10.
- TTA, TTAS.OT-10.0003 "한국형 웹 콘텐츠 접근성 지침", 2004.12

아래는 웹사이트의 접근성을 높이기 위한 표준안들 이다.

- [WCAG 1.0] Web Content Accessibility Guidelines 1.0, 5 May 1999, Wendy Chisholm, Gregg Vanderheiden, Ian Jacobs, http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505
- [WAI-AUTOOLS] "Authoring Tool Accessibility Guidelines", J. Treviranus, J. Richards, I. Jacobs, C. McCathieNevile, eds. 저작 도구 접근성 지침에 대한 가장 최신 작업 초안(Working Draft): http://www.w3.org/TR/WAI-AUTOOLS/
- [WAI-UA-SUPPORT] 이 문서에서는 (보조 기술을 포함하여) 웹 표시 장치들이 여기에서 언급된 접근성 관련 기능을 얼마나 지원하는지에 대해 언급하고 있다. 문서 있는 곳: http://www.w3.org/WAI/Resources/WAI-UA-Support
- [WAI-USERAGENT] "User Agent Accessibility Guidelines", J. Gunderson and I. Jacobs, eds. 접근성이 높은 웹 표시 장치를 설계하기 위한 이 지침에 대한 최신 작업 초안: http://www.w3.org/TR/WAI-USERAGENT
- [TECHNIQUES] "Techniques for Web Content Accessibility Guidelines 1.0", W. Chisholm, G. Vanderheiden, I. Jacobs, eds. 이 문서에서는 "웹 콘텐츠 접근성 지침 1.0"에서 정의한 체크포인트를 어떻게 구현하는지에 대해 설명하고 있다. 이 기술 문서의 최신 버전: http://www.w3.org/TR/WAI-WEBCONTENT-TECHS

웹 표준 검사 방법

W3C에서는 웹페이지가 표준안에 따라 만들어 졌는지, 접근성에 대한 고려가 이루어 졌는지 유효성 검사(Validation)에 대한 정보를 제공하고 있다. 개발의 맨 첫 단계에서부터 여러 가지 검사를 시작하면, 초기에 식별한 접근성 관련 문제점은 더 수정하기 쉽고, 피해갈 수 있다.

아래는 몇 개의 중요한 유효성 검사 방법으로 제시되는 것이다. 먼저 자동화된 접근성 검사 도구와 브라우저 유효성 검사 도구(http://validation.w3.org)를 사용한다. 한국어 번역이 제공되는 http://validator.kldp.net 을 이용하면 유효성 에러에 대한 설명과 해결을 한국어로 볼수 있다. 이 외에도 CSS 유효성 확인(http://jigsaw.w3.org/css-validator/)페이지와 XML에 대한 유효성 확인(http://www.stg.brown.edu/service/xmlvalid)를 사용할 수 있다.

이러한 웹 표준 문법에 대한 유효성 확인 도구가 모든 접근성 관련 문제점을 다룰 수는 없다는 점을 유의 해야한다. 예를 들면 링크 텍스트의 의미가 적절한지 여부나, 대체 텍스트(text equivalent)의 적용 가능성(applicability) 등은 다룰 수 없다. 따라서 W3C의 웹 접근성 체크 리스트를 체크해 볼 필요가 있는데, 한국형 웹 접근성 평가도구인 KADO-WAH(http://www.iabf.or.kr/web/kadowah.asp)라는 소프트웨어를 이용할 수 있다. 이 프로그램은 웹 페이지의 접근성 준수여부를 평가하고 접근성의 오류들을 바로 잡아주는 수정 과정을 통해서 웹 개발자와 콘텐츠 제작자들에게 장애인들이 웹페이지 접근이 용이한 웹 사이트를 만들 수 있도록 도와준다.

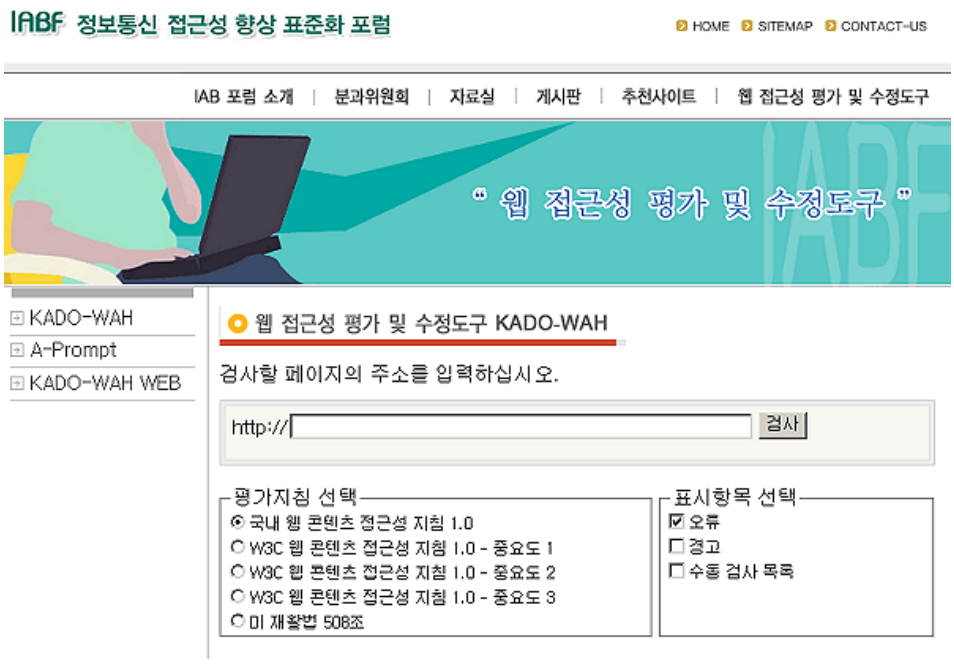


그림 6 웹 접근성 평가 도구, KADO-WAH

만약 좀 더 상세하게 접근하고자 한다면 텍스트만 나오는 브라우저 또는 에뮬레이터로 시험하여 페이지의 레이아웃이 올바른지 살펴 보고 여러 개의 그래픽 브라우저를 써서, 소리와 그래픽을 모두 받는 설정, 그래픽을 받지 는 설정, 소리를 받지 않는 설정, 마우스를 쓰지 않는 설정, 프레임, 스크립트, 스타일 시트, 애플릿을 사용하지 않는 설정 등을 통해 얼마나 접근도가 좋은지 체크해 볼 필요가 있다. 또한, 최근에 나온 것뿐 아니라 오래된 브라우저를 포함하여 여러 개의 브라우저로 시험해 본다면 더 좋을 것이다.

또한, 음성 브라우저(self-voicing browser), 화면 음성 변환기, 화면 확대 장치, 낮은 해상도의 화면 등을 써보면 자신의 웹페이지 접근도에서 문제되는 점을 고칠 수 있다. 마지막으로 철자법과 문법 검사기를 사용한다. 음성 합성기를 통해 페이지를 읽는 사람들은 철자법이 틀린 단어에 대해서 합성기가 읽어주는 것으로는 무슨 단어인지 추측할 수가 없을 것이다. 문법적인 오류도 없어야 이해하기가 쉽다.

기본적인 접근도 검사가 수행되었다면 문서가 간단 명료하게 작성되었는지 다시 점검한다. 일부 워드 프로세서가 생성해 주는 가독성 통계치같은 것들이 명확성이나 간결성에 대한 좋은 지표로 쓰일 수 있다. 그보다 더 나은 방법은 경험 있는 편집자에게 명료성을 검토해 주도록 부탁하는 것이다. 또한 경험있는 편집자는 특정 언어(단어나 표현)나 아이콘 사용이 야기할 수도 있는 잠재적으로 민감한 문화적인 문제점을 가려내 문서의 사용자 편의성(Usability)을 높일 수도 있다.

웹 표준 홈페이지 방법론

우리가 사용하는 HTML의 태그들은 구조(structure)를 나타내는 것과 표현(presentation)을 나타내는 것으로 분류할 수 있다. 구조를 나타내는 것의 대표적인 것이 H1, H2, UL, DL, OL, LI, LINK, ADDRESS, STRONG, BLOCKQUOTE, CODE, Q, DIV, P 등이다. 일반적인 브라우저에서는 구조적인 마크업을 사용할 때에 적당하게 모양

(표현)도 바꾸어 표현해준다. 예를 들면, H1이라는 마크업을 사용하면 보통은 글자 크기가 커지고 두꺼워진다. 그렇다고 해서 H1을 단순히 글자를 키울 목적으로 사용하는 것은 잘못된 것이다.

H1은 문서의 가장 중요한(또는 가장 상위의) 제목 부분을 표시하라는 "의미"를 가지고 있는 것이지, 그냥 글자를 키울 목적으로 제공되는 것이 아니기 때문이다. 이에 반해 B, FONT, I, BR 등은 문서의 표시 방법을 결정하는 표현 요소들이다. 즉 이것들은 아무런 의미는 없지만 단지 시각적으로 표현하는 방법만 결정한다. 예를 들어 B 요소를 사용하면 글자가 두꺼워진다.

초기 HTML이 계속 진보하면서 표현 마크업이 담당하던 모양에 대한 것은 CSS가 담당하게 되고, 이제 HTML에서 표현 마크업은 점점 쓸모가 없게 되었다. 따라서 1부에서는 문서 내에 FONT, B, BR 등을 되도록 쓰지 않도록 권고했다. 이런 것들로 표현하고자 하는 모양은 100% CSS를 이용해 훨씬 정교하게 표현할 수 있기 때문이다. 이렇게 HTML은 구조, CSS는 표현이라는 역할 분담을 함으로써, 구조를 고치기 위해 표현 방법을 바꾼다든지, 표현 방법을 바꾸기 위해 구조를 전부 뜯어고쳐야 할 일이 없어지게 된다.

구조와 표현의 분리

구조를 위한 마크업(structural markup)과 표현을 위한 마크업(presentational markup)이란 무엇인가? 이 둘을 구분하는 것은 매우 중요하다. 겉으로는 똑같이 보이는 웹페이지를 만들었더라도 속에 보이는 구조는 천차만별로 다를 수 있고, 이 다른 구조는 다양한 환경의 사용자에게, 또는 다양한 기기에게, 또는 검색 엔진이나 의미 해석 엔진(semantic parser)에게 엄청난 차이를 가져다 준다.

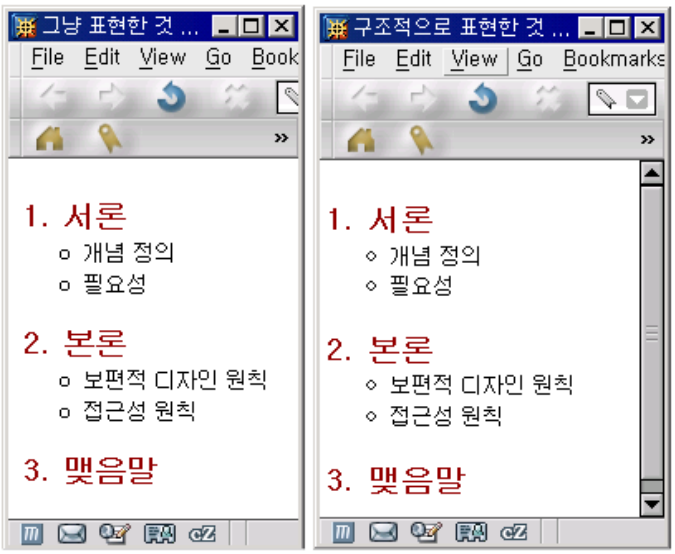


그림 7 구조적으로 표현한것과 태그로만 표현한 페이지

위의 그림은 웹 브라우저에 표시된 두 개의 문서를 보여주고 있다. 왼쪽 브라우저와 오른쪽 브라우저에 표시된 문서의 겉보기 모양은 거의 동일하다. 거의 같은 모양과 내용을 포함하고 있으나 이 두 페이지가 담고 있는 HTML은 완전히 다르다.

아마도 흔히 우리가 코딩하는 방식이나 간단하게 나모나 드림위버로 만들어버리면 아래와 같은 html이 만들어질 것이다. 이것은 그림의 왼쪽 화면의 모양과 같다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="ko">
<head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr">
<title>그냥 표현한 것</title>
</head>
<body>
<br>
<font size="4" color="darkred"><b>1. 서론</b></font><br>
    <font size="2" color="black">1. 개념
정의</font><br>
    <font size="2" color="black">2.
필요성</font><br>
<font size="4" color="darkred"><b>2. 본론</b></font><br>
    <font size="2" color="black">2.1 보편적 디자인
원칙</font><br>
    <font size="2" color="black">2.2 접근성
원칙</font><br>
<font size="4" color="darkred"><b>3. 맺음말</b></font><br>
</body>
</html>
```

이 예제를 보면 계층적인 제목을 나타내기 위해 글꼴을 바꾼 것과 줄 간격, 들여 쓰기 한 것, 그리고 번호를 붙인 것, 불릿 이미지로 동그라미를 붙인 것과 같은 표현 방법(presentation)들이 내용(contents)과 섞여 있어서 나중에 표현 방법만 살짝 바꾸거나 아니면 내용만 바꾸거나, 아니면 구조(structure)를 현재의 2단계에서 3단계 체제로 바꾸거나, 아니면 본론이 없어지고 결론이 바로 나오도록 하려면 번호도 다 바꾸어 주어야 한다.

이와 같이 구조와 표현과 내용이 한데 얽혀 있으면 나중에 소스를 관리하고 수정하기가 매우 어려울 뿐만 아니라, 컴퓨터가 이 문서의 구조를 파악하기도 매우 힘들다. 예를 들어 위의 문서에서 가장 큰문서의 구조는 서론, 본론, 결론의 세 가지 항목으로 이루어졌다고 사람은 알 수 있는데, 컴퓨터는 글자 색깔이 어떻고, 크기가 어떻고 하는 것이 의미가 없으므로 시각 장애인이 사용하는 음성 브라우저에서도 위와 같은 글자 크기가 어떻고 하는 것은 거의 의미가 없다.

따라서 아래의 HTML 소스와 같이 수정을 해주면, 컴퓨터나 검색 엔진이 문서의 의미적인 구조를 파악하기가 쉽고, 기계를 통해 문서의 구조에 쉽게 접근할 수 있다. 이것은 그림7의 오른쪽 화면을 나타낸다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="ko">
<head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr">
<style type="text/css">
```

```
* { line-height: 138%;}
ol>li {
font-size: 1.2em;
color: darkred;
font-weight: bold;
padding-top: 0.8em;
margin-left: -1em;
}
ul>li {
font-size: 0.7em;
color: black;
font-weight: normal;
margin-left: -2em;
}
</style>
<title>구조적으로 표현한 것</title>
</head>
<body>
<ol>
  <li>서론</li>
    <ul>
      <li>개념 정의</li>
      <li>필요성</li>
    </ul>
  <li>본론</li>
    <ul>
      <li>보편적 디자인 원칙</li>
      <li>접근성 원칙</li>
    </ul>
  <li>맺음말</li>
</ol>
</body>
</html>
```

이 HTML 소스는 스타일 지정 부분이 따로 있기 때문에 이 부분만을 바꿈으로써 문서의 구조는 그대로 유지한 채 모양을 자유자재로 바꿀 수가 있고, 심지어 이 스타일 파일만을 여러 벌 만들어 한 벌은 모바일용으로, 한 벌은 음성 브라우저용으로, 한 벌은 그래픽 브라우저용으로 최적화해서 만들 수도 있다. 그러면, 구조와 내용은 아주 간결하게 놔둔 채로 표현 방법만 상황에 맞게 여러 가지로 할 수 있게 된다.

이런 경우 인터넷 익스플로러에서는 CSS의 선택자(selector)를 제대로 해석하지 못해 색깔이나 글자 크기가 제대로 표현되지 않을 수 있다. 그러나 모질라나 오페라와 같은 다른 브라우저에서는 원래 의도한 대로 정확히 보일 것이다. 그럼에도 불구하고 이렇게 쓰는 것이 font를 이용하여 쓰는 것보다 더 좋은 방법이다.

여기에서 사용된 마크업은 표준에 따라 만든 것이므로 미래에는 인터넷 익스플로러도 이 기능을 지원할 것이고, 설사 이 기능이 지원되지 않는다고 하더라도 문서를 구조적으로 이해하는 데에 아무런 문제가 없기 때문이다. 이것은 브라우저 이미 언급한 하위 버전 호환

성뿐만 아니라 향후 미래에 나올 브라우저에 대한 상위 버전 호환성(Forward Compatibility)로 매우 중요하다는 점이다.

즉, 잠재적으로 스타일시트 기능을 지원하지 않는 어떤 브라우저에서 보더라도, 또는 스타일 시트를 완전히 제거하더라도 이 문서를 이해하는 데에는 아무런 문제가 없기 때문이다. HTML에서는 표현과 구조를 나타내는 마크업이 완전하고 명확하게 구분되지는 않았다. 그래서 XML이 나오게 된 것이고, XML은 표현을 위한 마크업이 아예 존재하지 않는다. XML 문서 내에서는 문서의 구조만을 나타내고 표현을 위한 부분은 CSS, XSL(T) 등으로 아예 따로 떼어내어서 표현해야 한다.

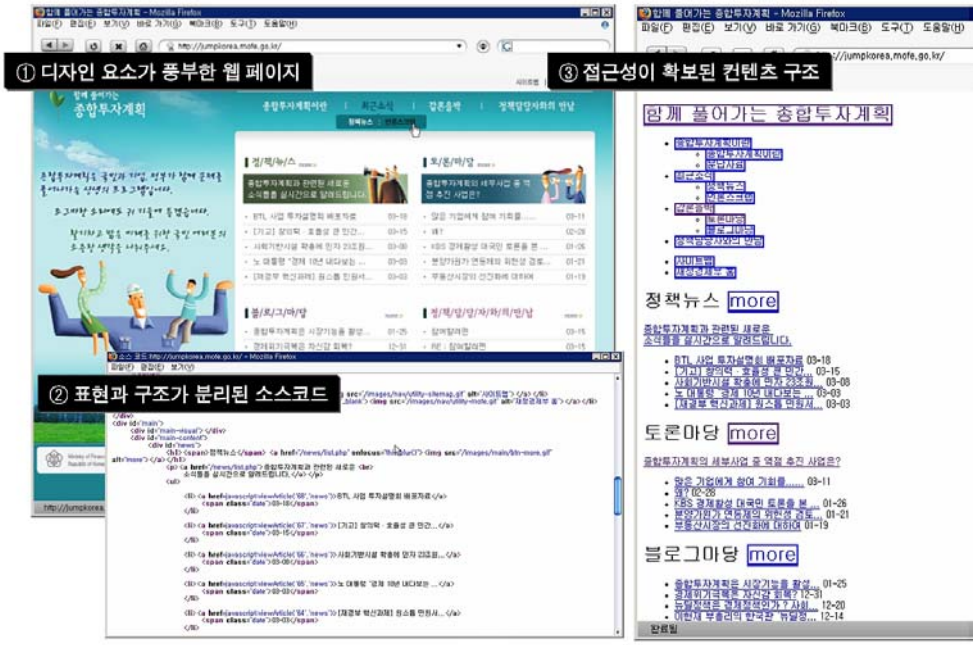


그림 8 구조와 표현의 분리를 통해 디자인 및 접근성이 확보된 웹 페이지 (1)은 기본 스타일 양식, (2)는 구조적 마크업을 통한 HTML소스 (3)은 스타일을 걷어 냈을 때 시맨틱 내용만 있는 모습

구조와 동작(Behavior)의 분리

우리가 흔히 접하는 웹 문서에는 이전에 말한 대로 구조와 표현만 있는 것이 아니다. 바로 사용자 액션에 따른 반응 및 동적인 문서 구조 변경 등 스크립팅(Scripting)이라고 불리우는 클라이언트 프로그램이 있다. 일반적으로 자바 스크립트(JavaScript) 또는 VBScript 등으로 불리는 이러한 스크립팅을 통한 제어를 동작(Behavior) 이라고 통칭한다. 구조에서 동작을 분리해 내는 일은 구조에서 표현을 분리해내는 일보다 더 어렵다. 특히, CSS는 선언적인데 반해 스크립트 언어는 그렇지 않기 때문이다.

특히 웹이 문서를 표현 하는 수단으로 여기지기 때문에 웹 문서에서 DHTML이나 DOM 스크립팅 같은 행동 제어를 하는 것에 대해 많은 문제점이 있다고 지적되었다.

- ❑ 사용성: 팝업이나 상태창 메시지, 슬라이딩 메뉴 등은 사용성을 크게 해친다.
- ❑ 접근성: 자바스크립트를 꺼놓았거나 시각 장애인인 경우 접근성이 떨어진다.
- ❑ 복잡성: HTML에 스크립트가 끼어 들어가 코드의 복잡도가 높아진다.

- 중복성: CSS에서 스크립트가 하는 전통적인 기능을 수행할 수 있는 데도

이러한 문제점에도 불구하고 스크립트를 통한 DOM 핸들링은 매우 광범위하게 사용되고 있다. 위에서 말한 문제점을 해결 하기 위해서는 구조에서 스크립트를 분리해야 될 필요가 있다.

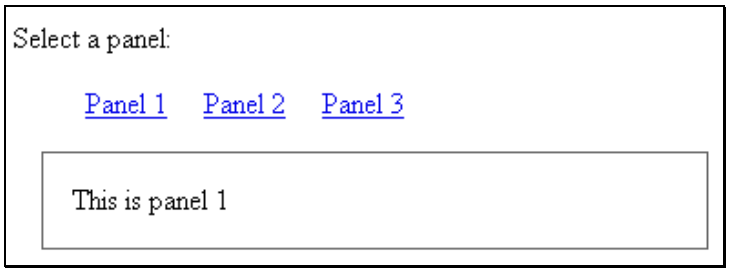


그림 9 패널 번호를 누르면 내용이 바뀌는 예제

아래 코드는 위와 같이 패널 번호를 누르면 내용이 바뀌는 기능을 구현한 것이다.

```
<script type="text/javascript">
Function changePanel (id) {
 document.getElementById("panel").innerHTML="This is panel"+id;
}
<ul>
 <li><a href="#panel1" onClick="changePanel(1);">Panel 1</a></li>
 <li><a href="#panel2" onClick="changePanel(2);">Panel 2</a></li>
 <li><a href="#panel3" onClick="changePanel(3);">Panel 3</a></li>
</ul>

<div id="panel"> </div>
```

통상적으로 위와 같이 구현을 하고 나면 panel을 누를 때 마다 나타나는 값에 대한 접근성이 떨어지게 된다. 또한, 구조 내에 onClick과 같은 이벤트 요소를 넣을 수 밖에 없다.

구조에서 행동을 분리하기 위해서는 우선 구조적인 태그와 함께 반응에 따라 얻어지는 결과값도 함께 제공한다.

```
<script src="easytoggle.js"></script>
<p>Select a panel:</p>
<ul>
 <li><a href="#panel1" class="toggle">Panel 1</a></li>
 <li><a href="#panel2" class="toggle">Panel 2</a></li>
 <li><a href="#panel3" class="toggle">Panel 3</a></li>
</ul>

<div id="panel1">This is panel 1</div>
<div id="panel2">This is panel 2</div>
<div id="panel3">This is panel 3</div>
```

이렇게 제공 한 후에 자바 스크립트에서는 사용자의 이벤트를 감시하고 제어하는 기능을 초기화 하여 사용자가 panel을 누를 때 마다 나타나는 내용을 변화 시켜 주면 된다. 아래

소스 코드는 그러한 예이다.

```
/* easytoggle.js, by Simon Willison */

addEvent(window, 'load', et_init); // 이벤트 감시

var et_toggleElements = [];

/* 초기화 */
function et_init() {
    var i, link, id, target, first;
    first = true;
    for (i = 0; (link = document.links[i]); i++) {
        if (/\btoggle\b/.exec(link.className)) {
            id = link.href.split('#')[1];
            target = document.getElementById(id);
            et_toggleElements[et_toggleElements.length] = target;
            if (first) {
                first = false;
            } else {
                target.style.display = 'none';
            }
            link.onclick = et_toggle;
        }
    }
}
// 토글일 경우 디스플레이 변경
function et_toggle(e) {
    /* Event handling code adapted from
       http://www.quirksmode.org/js/events_properties.html
    */
    if (typeof e == 'undefined') {
        var e = window.event;
    }
    var source;
    if (typeof e.target != 'undefined') {
        source = e.target;
    } else if (typeof e.srcElement != 'undefined') {
        source = e.srcElement;
    } else {
        return true;
    }
    /* For most browsers, targ would now be a link element; Safari
       however returns a text node so we need to check the node
       type to make sure */
    if (source.nodeType == 3) {
        source = source.parentNode;
    }
    var id = source.href.split('#')[1];
    var elem;
```

```
    for (var i = 0; (elem = et_toggleElements[i]); i++) {
        if (elem.id != id) {
            elem.style.display = 'none';
        } else {
            elem.style.display = 'block';
        }
    }
    return false;
}

/* 브라우저별 이벤트 처리*/
function addEvent(obj, evType, fn){
    if (obj.addEventListener) {
        obj.addEventListener(evType, fn, true);
        return true;
    } else if (obj.attachEvent) {
        var r = obj.attachEvent("on"+evType, fn);
        return r;
    } else {
      return false;
    }
}
```

위의 예에서 사용성을 만족 시키면서도 접근성과 복잡성을 모두 해결한 방법을 간단하게 배웠다. 구조에서 행동을 분리하는 것은 아직 많은 시도가 이루어지고 있지 않은 분야이기도 하다. 그러나, 현대 웹이 어플리케이션으로 진화함에 따라 문서로 인식되던 웹에서 구조 요소를 제외하는 것은 무엇보다 중요하게 되었다. 특히, Ajax와 플러그인 등 다양한 웹 어플리케이션 기술에서 어떻게 하면 접근성 높은 상태에서 웹의 특징을 보존할 수 있을 것인가 하는 문제는 향후에도 중요한 이슈가 될 것이다.

우리 나라 웹 표준 현실 및 과제

국내 웹 사이트의 웹 표준 여부를 고려해 보면 현실은 매우 심각하다. 2004년에 공공기관 정보접근성 현황(KIPA 조사 자료)을 조사한 결과는 아래와 같다.

- 조사 대상: 정부 및 공공기관, 금융기관 웹 사이트 1,000개
- 조사 기간: 2004년 10월~12월
- 조사 방법: 조사대상 웹사이트를 윈도즈, 리눅스, 맥킨토시 3종의 운영체제하에서 익스플러로, 모질라, 사파리를 이용하여 접속 및 정상작동 여부 조사
- 조사 항목: ① 초기 화면의 정상 작동 여부, ② MS익스플로러 최적화 문구 표기, ③ 글꼴의 깨짐 없는 내용의 전달, ④ 그림 및 사진 디스플레이 가능, ⑤ 동영상 작동, ⑥ 자료 다운로드 가능, ⑦ 스크립트 오류 발생 여부, ⑧ 인증여부, ⑨ 인터넷뱅킹 작동여부 등 9개 항목
- 조사 결과: 전체의 12.3%인 110개 기관 인터넷 익스플러어 최적화 표기, 조사 대상기관의 99.8%가 모질라(리눅스), 사파리(맥킨토시) 부분장애 또는 심각한 장애 발생

웹 표준에따라 홈페이지를 구축하는 분위기가 정립되어 있을 뿐 만 아니라, 정보 접근 문제에 대한 인식도 매우 저조하기 때문이다. 정보 접근성 문제를 장애인 접근성에 한정되게

생각하는 것도 문제이다. 정보 접근성 문제를 가지고 있는 대상자는 장애인 뿐만 아니라 소수 운영체제 및 브라우저 사용자, 비 PC 단말기 사용자 (PDA, Phone), 인터넷 환경이 열악한 제3 세계 재한국인, 정보 소외 계층(노인 및 장애인, 농민 및 빈민) 등 다양하기 때문이다. 이는 전 국민이 잠재적인 대상자라고 볼 수 있다.

국내 웹 표준 문제 현황 및 요인

국내 정보 접근성의 주요 문제는 크게 웹 정보 전달 행태 즉, 고기능을 위주(과잉 개인 정보 및 기능 요구, 스크립트의 무분별한 사용), 고사양을 위주(고해상도 색상 및 플래쉬, 고사양 PC 위주)의 정보 제공 문제와 웹 정보 가공 행태 즉, 비표준에 따른 웹페이지 가공 (IE 종속적인 기능)과 과도한 플러그인 기능 사용 (ActiveX)으로 나뉜다.

현황 1. 비표준 웹 페이지 가공

공공기관 웹사이트에서 사용하고 있는 HTML 문서에 사용된 태그들이 90년대 중반에 양산된 IE용 비표준 태그들을 그대로 채용하고 있는 곳이 많다. 비표준 태그는 IE에서만 지원 가능하므로 IE 이외의 웹브라우저에서 웹페이지를 제대로 표시할 수 없는 문제를 야기시키고 있으며, HTML문서를 구성하는 표준 형식을 생략함으로서 브라우저가 글꼴 및 그림을 제대로 표현 할 수 없는 상태가 되고 있다. HTML 뿐만 아니라 비표준 JavaScript는 로그인, 회원 가입, 검색, 주소 입력 등 웹페이지 상의 다양한 활동을 제약시키고 있다.

우선 먼저 비표준 태그를 표준 태그로 변환 시키는 전반적인 HTML 리팩토링 작업을 통해 간단하게 웹사이트 접근성 향상이 가능하다.

현황 2. 과도한 플러그인 사용

전자 정부 사이트를 비롯 공공 기관 웹사이트 온라인 민원 서비스를 이용하기 위해서는 본인 확인을 위해 공인 인증서를 사용하여야 한다. 공인 인증서를 확인하고 처리하는 프로그램은 웹브라우저에 포함되어 있지 않으므로 별도의 브라우저 플러그인으로 제공하고 있다. 인증서 플러그인이 Microsoft사의 윈도우즈를 기반으로 하는 ActiveX 기술로만 제작되어 있어 매킨토시, 리눅스 등 비윈도우즈용 OS에서 온라인 민원 서비스를 이용할 수 없다.

윈도우즈 사용자라 하더라도 IE이외 모질라, 사파리 웹브라우저 사용자는 MS의 ActiveX 기술을 사용하지 못하므로 접근성에 제약을 받고 있다. 이는 공인 인증에 기반이 되는 정부 전자 서명 체계는 OS나 브라우저에 관계없이 구현 가능한 기술이나, 예산상의 문제를 들어 타 OS 및 브라우저를 지원하지 않고 있기 때문이다. IE 사용자라도 동일한 기능을 하는 ActiveX 플러그인이 공인 인증 기관 6개, 은행 3~4개 등 10여개의 플러그인이 중복 개발되고 있으며, 해외의 경우 표준 SSL과 간단한 기능의 공통 ActiveX 및 Java Plugin 만으로 인터넷 뱅킹을 구현하여 서비스하고 있다. 뿐만 아니라 민간 부문에서도 우리 나라 ActiveX 사용 비율은 전세계적으로도 가장 높은 편에 속해 있다.

그 밖에, 공공 기관 웹사이트에서 제공하는 인터넷 방송 및 미디어 배포 자료가 모두 Microsoft사에서 제공하는 윈도우즈 미디어(Windows Media) 포맷으로 제공되고 있는데 윈도우즈 미디어 포맷이더라도 비 IE 브라우저에서 재생 될 수 있도록 재생을 위한 HTML 태그를 지원해 주고 있지 않고 있다. 윈도우즈 미디어의 스트리밍 방식을 사용함

으로서 저속도 사용자나 비윈도우즈 미디어 플레이어 사용자가 접근성에 제약을 받고 있는 것이다.

리눅스 및 매킨토시에서도 재생 가능한 WAV, MPEG 파일에 대한 지원 및 다운로드가 제공되고 있지 않고 있다. 이러한 현상 들은 공공 기관뿐만 아니라 일반 기업 웹 사이트에서도 흔히 볼 수 있는 문제이다.

웹 표준 문제 발생 요인

이미 언급한 바 대로 1990년대 중반 소위 브라우저 전쟁기간 동안 인터넷 익스플로러와 넷스케이프간의 비표준을 기반한 경쟁 이후, 시장이 IE 독점 상태가 되면서 IE 전용 기술만 잔재로 남게 되었다. 2000년에 들어와서 웹 표준 기술이 비약적으로 발전하였으나, 독점 브라우저인 IE 의 하위 버전 호환 기능(Backward Compatibility) 으로 인해 신 기술에 대한 추가가 잘 되지 못하였다.

뿐만 아니라 표준 기술에 대한 국내 웹디자이너/UI 개발자 등 웹 생산 종사자 재교육 및 자기 개발 부재도 큰 요인이 되었다.

특히, 국내 브로드 밴드 인터넷 환경의 급격한 성장과 공공 부문에서 자체 공인 인증 시스템으로 인해 플러그인 기술이 광범위하게 도입되었으며 인터넷 산업화로 인한 엔터테인먼트 인터넷으로 진화하면서 표준 웹 문서 교환이라는 고유의 모습이 잘 지켜지지 못했다. 과다한 홈페이지 구축열로 인해 SI를 통해 고정화된 열악한 국내 웹 생산 시스템 구조 속에서 웹에 대한 기본 인식 및 개발 방식에 대한 이해 및 교육 부재, 비용과 효율만 중요시하는 행태 등 공공재로서의 웹을 바라보는 인식이 매우 부족했다고 할 수 있다.

외국의 웹 표준 제도 동향

우리 나라 밖에서는 대부분 웹사이트들이 웹 표준을 채택 하고 있는 반면, 정보 접근이 제약 받는 다양한 계층을 위한 제도를 가지고 있다.

미국

미국 재활법(The Rehatilitation Act Amendment) 508조는 연방 우편업무를 포함하여 연방부처나 기구가 전자 및 정보기술을 개발, 조달, 유지, 사용할 대는 지나친 부담(Undue Burden)이 되지 않는 한 사용하는 기술의 종류에 상관없이 장애를 지닌 연방정부 직원도 비장애인과 동등한 수준으로 정보와 자료에 접근하여 이용할 수 있어야 함을 규정하고 있다.

동 조항은 연방정부가 준수하는 조항이기 때문에 미 연방 조달시장 진입을 위해서는 필수적으로 준수하여야 하며, 전자 및 정보기술의 접근성 표준안(Electronic and Information Technology Accessibility Standards)을 만들어 지침으로 활용(WCAG기반 작성)하고 있다.

영국

영국은 전자정부를 추진하는 e-Envoy에서 영국 정부의 웹사이트 접근성 준수를 위한 가이드라인을 제정/공포 하고 있다. 이는 W3C의 WCAG1.0을 기반으로 가이드라인이 작

성되어 보편적 접근성 보장하고 있으며 시각 장애인 기관인 RNIB(Royal National Institute for the Blind)에서 웹접근성 인증마크 제도(See it Right) 시행하고 있다.

호주

호주는 W3C WCAG 기준을 활용하여 웹접근성 표준 지침으로 활용하고 있으며, 2001년 NOIE(National Office for the Information Economy)에서 연방 정부 및 기관이 지켜야 할 표준 가이드 제정하고 있다.

국내 웹 표준 제도 소개

2003년 말부터 우리 나라에서도 공공 기관을 시작으로 웹 표준에 대한 인식을 새로이 하기 시작하여, 전자정부 사업 공개 SW 도입 권고안(2003), 공개 SW 기반 정보 시스템 구축 사용자 가이드(2004), 행정기관 홈페이지 구축 운영 표준 지침(2005), 행정기관 홈페이지 평가 지표(2005), 소수 정보통신 환경 사용자 정보 접근성 제약 개선 방안(2005) 등에 정보 접근성, 웹 표준, 웹 접근성에 대한 다양한 가이드라인들이 제정 되었다.

특히, 한국정보 통신 표준 협회(TTA)에 XHTML 1.0 표준(2001.12.), 문서 객체 모델 레벨 1(2003.10.), 한국형 웹 콘텐츠 접근성 지침(2004.12) 등이 산업 표준으로 제정되기도 했다.

이들 중 많은 공공 기관에서 홈페이지를 구축할 때 표준 지침으로 사용하고 있는 행정 자치부의 행정 기관 홈페이지 구축 가이드 및 평가 지표를 통해 웹 표준 문제에 대한 요구사항과 해결 방법을 이 가이드에 기초하여 간단하게 살펴 보도록 한다.

2005년 행정 기관 홈페이지 구축 가이드

본 가이드는 행정자치부 전자정부 본부에서 발간한 홈페이지 구축 가이드 중 일부이다.

지표 항목	준수 방법
6. 개인별 맞춤 서비스 ○ 장애인, 노인 등을 위한 웹 접근성 준수	텍스트 버전을 별도로 제공하여 접근성을 높이기 위한 지침으로서, 표준 기반으로 구조와 표현을 분리한 후 스타일 변경 만으로도 텍스트 페이지를 구성할 수 있다.
9. 저작권, 개인 정보 보호 등을 위한 고려 ○ 장애인, 노인 등 다양한 이용자의 용이한 접근을 확보하기 위해 한국형 웹 콘텐츠 접근성 지침 중 다음 사항을 고려하여 제작 - 그림에 대한 대체 텍스트 제공(지침1), 탭 및 화살표 등 최소 키보드로 접근 가능성 확보(지침5), 각 프레임별 별도 제목 부여(지침7), 스타일을 제거하더라도 문서 구조를 쉽게 파악 가능해야 함(지침11) 등 ※ 별첨 2「한국형 웹 콘텐츠 접근성 지침」참조	장애인 접근성을 위해 한국형 웹 콘텐츠 접근성 지침 중 꼭 필요한 사항만을 제시한 것으로 별첨한 지침을 참고하면 도움이 될 것이다.
별첨1. 3. 기술 표준 ○ 인터페이스- 브라우저: 익스플로러 6.0 이상(윈도우), 사파리 1.0(매킨토시) 등에서 작동되어야 하며 리눅스 등 공개 소	지원해야할 웹 브라우저의 버전을 명시한 것으로 웹 표준에 대한 기술셋이 유사한 주요 주요 운영 체제의 브라우저를 대상으로 명기하였

프트웨어 사용자의 웹브라우저에서도 작동되어야 함. (예 모질라 파이어폭스 1.0)	다.
별첨1. 3. 기술 표준 ○ 뷰어 실행 프로그램- 브라우저 : 익스플로러 6.0 이상(윈도우), 사파리 1.0(매킨토시) 및 리눅스 등 공개소프트웨어 (예 모질라 파이어폭스 1.0)	지원해야할 웹 브라우저의 버전을 명시한 것으로 웹 표준에 대한 기술셋이 유사한 주요 주요 운영 체제의 브라우저를 대상으로 명기하였다.
별첨1. 3. 기술 표준 ○ 뷰어 실행 프로그램- 문서 뷰어: MS word viewer 97, Ms Excel viewer 97, Ms Powerpoint viewer 97, 한글뷰어 2002, OpenOffice, Acrobat PDF Reader	모든 운영 체제와 브라우저에서 지원되고 있는 PDF 뷰어가 추가 되었다.

2005년 행정기관 홈페이지 평가 지표안

본 지표는 행정자치부 전자정부 본부에서 매년 시행하는 홈페이지 평가 지표 중 일부다.

지표 항목	준수 방법
□ 홈페이지 구축성 1. 정보 접근성 (Information Accessibility) 1-1 국제 표준 준수 - 홈페이지가 HTML4.01 또는 XHTML 1.0 표준을 준수하고 있는가 - 디자인 표현을 위해 CSS 1.0 표준을 준수하고 있는가 - 정보 제공을 위해 XML 1.0, XSL 1.0 표준을 준수하고 있는가 - 사용자 기능 제공을 위해 W3C 객체모델(DOM) 2.0과 ECMAScript 표준(자바 스크립트)를 준수하고 있는가 * 위의 항목들을 W3C Validator 또는 Venkman등 표준 자바 스크립트 디버거로 조사하여 오류가 없어야 한다.	국제 웹 표준이 준수되고 있는 지 여부를 명기하여 유효한 웹 제작이 되도록 하였다. 본 가이드의 XHTML, CSS, DOM, Javascript 부분을 잘 고려하여 웹 개발을 하면 문제 없이 만들 수 있다.
1-2 브라우저 및 운영 체제 호환성 - 공인 인증 서비스를 제외한 전 페이지가 익스플로러 6.0(윈도), 모질라 파이어폭스 1.0(리눅스), 사파리 1.0(매킨토시), 에서 이상 없이 동작 하는가 - 공인 인증 서비스를 제외한 전 페이지가 윈도, 리눅스, 매킨토시 등 주요 운영 체제에서 이상 없이 동작하는가	지원해야할 웹 브라우저의 버전을 명시한 것으로 웹 표준에 대한 기술셋이 유사한 주요 주요 운영 체제의 브라우저를 대상으로 명기하였다. 공인 인증 서비스를 제외하고 ActiveX같은 종속적 기능은 배제하거나 대체 기술을 제공하도록 함을 명기하였다.
1-3 한국형 웹 콘텐츠 접근성 지침 주요 항목 적용 - 모든 비 텍스트 콘텐츠에 대해서 대체 텍스트를 제공하는가 (지침1) - 탭 및 화살표 등 최소한의 키보드로 홈페이지 네비게인션이 가능한가 (지침5) - 프레임을 제공하지 않거나 제공 할 경우 각 프레임별 별도 제목을 부여하고 있는가 (지침 7) - 스타일을 제거하더라도 일렬로 문서 구조를 한눈에 쉽게 파악 가능한가 (지침 11)	장애인 접근성을 위해 한국형 웹 콘텐츠 접근성 지침 중 꼭 필요한 사항만을 제시한 것으로 별첨한 지침을 참고하면 도움이 될 것이다.

* 위의 항목들을 웹접근성 평가 및 수정 도구를 통해 조사하여 오류가 없어야 한다.	
2-2 해상도 - 사용자가 다양한 해상도를 사용하더라도 문서의 내용을 이해하거나 사용자 기능을 사용할 수 있는가? - 사용자가 다양한 해상도를 사용할 경우를 대비하여 팝업창을 아예 사용하지 않거나 사용할 경우, 문서의 내용을 이해하거나 닫을 수 있는 방법을 제공 하고 있는가?	해상도에 따라 팝업창을 닫을 수 없거나 내용을 못보는 것을 방지하는 것이 필요하다. 또한, 다양한 해상도에서도 내용을 이해할 수 있도록 레이 아웃을 고정하는 것을 지양 시킨다.
□ 홈페이지 기술 평가 1. 정보 접근성 (Information Accessibility) 1.1. 표준 및 주요 운영 체제 지원 - 주요 운영 체제 및 웹브라우저 지원 여부 - 장애인, 텍스트 브라우저 지원 여부 - Validator.w3.org에 검증하여 에러 여부 - 표준 자바스크립트 디버거로 검증 하에 에러 여부 - 웹 접근성 평가 및 수정 도구로 주요 항목을 검증 하에 에러 여부	기술 평가를 위한 각종 품질 관리(QA) 항목을 설정 하였다. 이 검증 도구를 통해 손쉽게 웹 표준 준수 여부를 확인 할 수 있다.
3. 성능 (Performance) 1-1. 웹페이지 로딩 속도 - 단위 웹페이지 HTML 크기로 로딩 속도 산출 (페이지 당 70kbyte 이내 권고)	테이블 레이아웃이 아닌 CSS 레이아웃을 이용함으로서 코드량이 줄기 때문에 웹 페이지(HTML) 용량를 줄일 수 있다.
4. 유지 보수성 (Maintainability) 1-2 유지 보수 용이성 - 웹페이지를 쉽게 관리할 수 있도록 표현 요소를Cascading Style Sheet(CSS)로 설계 했는지 유무 - 기본 페이지를 CSS 레이아웃을 통해 제작하여 스타일변경으로 텍스트, 장애인, 모바일 페이지로 재사용 하고 있는지 여부.	CSS 레이아웃을 이용하여 유지 보수에 용이하게 개발 했는지 여부와 스타일 변경만으로 다양한 별도 웹페이지 기능이 제공 되는지 여부를 확인한다.
□ 홈페이지 특성 평가 3. 사용성 3.1 W3C 표준을 준수하는 주요 플래폼의 주요 브라우저가 전 페이지 사용 가능한가? 3.2 모든 해상도에서 전 페이지의 내용 확인 및 기능 사용이 가능한가?	W3C 표준과 브라우저 지원, 구조적인 컨텐츠 전달이 가능한가를 확인하고 있다.
4. 내용 4.4 다양한 멀티미디어 자료(동영상, VOD등)을 주요 브라우저에서 사용 가능 하도록 제공하는가? 4.5 원문이나 제공 자료를 다운로드 할 수 있으며 주요 운영체제별 뷰어 프로그램을 제공하는가?	동영상 및 문서 자료를 다양한 웹 브라우저에서도 볼 수 있도록 하고 있으며, 본 가이드의 플러그인 사용 방법 및 웹 서버 설정 방법을 통해 설정 할 수 있다.

실전 XHTML 가이드

XTHML 소개

웹이 계속해서 성장, 변화, 성숙하는 것과 마찬가지로, HTML 역시 계속 진화하고 있는 것이다. 표준화 그룹이나 브라우저 회사에서 표준화 작업을 했었지만 HTML의 변화를 막지는 못했다. 초기 웹은 조잡하고 제대로 된 형태를 이루지 못하다가, 새로운 브라우저 버전이 출시될 때마다 새로운 태그가 정의되곤 했다.

HTML 3.2는 가장 흔히 쓰이는 태그와 이의 속성을 하나로 묶어 주어 웹 개발이라는 세계가 어느 정도 안정을 찾는 듯했다. HTML 3.2가 널리 보급되면서 HTML 4.0이 생겨나게 되었는데 HTML 4.0은 좋으며 간결하고 잘 지원되는 최초의 HTML 표준이다. HTML 4.0은 어떤 태그가 올바른 것인지 말해 줄 뿐 아니라 어떤 태그가 사라져 가고 있는지도 알려 주기 때문에 개발자는 이에 따라 작업해 나가면 된다.

웹 표준화를 맡고 있는 W3C는 HTML 4.0을 표준으로 발표한 바로 직후, 더 큰 모듈성, 유연성, 성능이 필요함을 절감하게 되었다. HTML이 처음 들어 왔을 때 HTML이 이처럼 많은 문서와 브라우저, 미디어를 다룰 것이라고는 아무도 예상하지 못했다. 웹 사용자들의 끝도 없는 요구에 응답하면서 HTML 4.0은 끊임 없이 확장해서 새로운 기술을 수용해야만 했는데 이를 위해 XTHML이 등장 했다.

XTHML이란 무엇인가?

XHTML은 eXtensible HyperText Markup Language의 약자이다. HTML을 대체하기 위한 목적으로 만들어졌지만 HTML 4.01 규약에 "거의" 준한다. 쉽게 말하자면 XHTML은 HTML에 비해 일반 HTML에 비해 좀더 명확하고 구조적인 특징을 가지고 있다. HTML은 정해진 태그 집합만을 사용하는 정적인 마크업 언어라면, XHTML은 XML(Extensible Markup Language: 확장 마크업 언어)로 정의된 단 하나의 집합만을 사용한다.

XML을 사용하면 HTML을 넘어서는 모든 종류의 데이터와 문서를 표현할 수 있다. 다양한 데이터셋을 만들어 표현 할 수 있는 것이다. 현재의 브라우저에서도 작동하는 혼성 문서(hybrid documents)를 생성하여, 사용자가 새로운 마크업 태그를 XHTML과 통합할 수도 있다. XSL(Extensible Style Sheets)을 함께 사용하면, 브라우저에 사용자의 새로운 태그를 표시하는 방법도 변경할 수 있다.

XML은 웹이 성장 및 확장할 수 있는 플랫폼이며 XHTML은 XML을 사용하여 HTML을 혁신한 것으로 XML이 사용되는 모든 새로운 툴에 XHTML을 결합해 준다. 하지만 HTML을 잘 사용한다고 해서 XHTML 역시 잘 사용하리라는 보장은 없다. XHTML이 HTML과 비슷한 사용법을 가지기 위해 노력을 기울이긴 했지만, 이것만 믿고 있다가는 머리 아픈 문제를 많이 만나게 될 것이다. XHTML을 잘 사용하기 위한 유일한 방법은 최신 버전인 HTML 4.01 표준을 사용해 보면서, XHTML의 모든 세부 사항을 익히는 것이다.

2000년 1월에 W3C의 공식 표준으로 지정된 이후 HTML의 표준이라하면 XHTML 1.0을 가리킵니다. 당연히 최근의 모든 HTML 브라우저는 XHTML 1.0을 완벽히 지원하고 있다. 대부분의 브라우저는 일반 HTML을 써도 상관없다. 게다가 HTML 문법이 상당히 느

슨하기 때문에 어쩌면 XHTML의 딱딱한 규정을 지키는 것이 오히려 번거로울 수도 있다. 그렇다면 왜 굳이 HTML대신 XHTML을 써야 하는 것인가?

왜 XTHML을 사용해야 하는가?

XHTML 1.0으로 전이함으로써 얻는 잇점은 위에서 기술하였다. 이 XHTML로 전이의 몇 가지 일반적인 잇점은 다음과 같다:

호환성 및 확장 가능성

XHTML은 단순히 HTML4를 업그레이드한 것이 아니라 XML 애플리케이션을 가장 폭넓게 사용하는 웹 페이지에 적용할 수 있다는 데에서 의미를 찾을 수 있다. XML 어플리케이션이 사용 가능하다는 말은 바로 기계가 이해할 수 있는 언어라는 것이다. 기계가 이해하려면 우리가 흔히 색상, 글꼴 형식, 레이아웃 등 눈으로 보는 표현적인 요소가 완전히 배제되어 있다는 것이다. HTML4.01 보다 더 표현과 구조가 엄격하게 분리되고 있다는 뜻이다. 다만 브라우저 지원 문제가 걸림돌인데, 인터넷 익스플로러도 XML을 충분히 지원한다고 보기에 아직 부족한 면을 찾을 수 있다. XHTML은 HTML 문서의 하위 호환성 유지와 함께 더욱 강력하게 확장할 수 있도록 해주는 것이다.

XHTML 1.0 스펙에 무슨 내용이 들어있는지 막상 뚜껑을 열어보면 그 단순함에 당황하게 된다. HTML 4.01 스펙은 389쪽, CSS2 스펙은 338쪽 정도로 책 한권 분량의 문서로 접할 수 있으며, 그 자체로 충분한 레퍼런스 역할을 해준다. 이러한 스펙과 달리 XHTML 1.0은 30쪽 정도밖에 안되므로 지나칠 정도로 단순한 것이 아니냐는 생각을 가질 수도 있으며, 레퍼런스로 삼기에 부실하다는 사람도 많이 있다. HTML4.01 스펙을 참고하여 XML적인 요소를 좀 더 이해 한다면 XHTML 스펙에 대한 이해도를 높힐 수 있다.

유지 비용의 감소 및 재생산성 확대

사실 국내에서 웹 페이지 제작은 "IE를 통해 사용자가 눈으로 보는 것"을 목적으로 만들어지는 경향이 있다. 따라서 일단 눈으로 보이는 부분만 멀쩡하면 내부적으로 HTML 문서가 어떻게 구성되어있든 아무도 신경 쓰지 않는다. 이것은 HTML의 재활용과 생산성의 문제이며 장차 나오는 비용의 문제이다. 일반적인 관습대로 작성된 HTML 문서는 내용과 디자인, 문서 구조가 모두 뒤범벅이 되어있다.

디자인을 바꾸려면 일일이 HTML문서를 수정해야 한다. 기존의 HTML 제작 방법으로는 이러한 변화에 대처하기 힘들며, 똑같은 내용이라고 하더라도 모바일 환경을 위해서 따로 만들고, PDA버전을 위해 따로 만들고, 심지어는 같은 PC환경이라도 브라우저 버전별로 따로 만들기도 한다. 뿐만 아니라 장애인용 저속 사용자용 텍스트 버전 심지어 회사의 PR 사이트를 만들면서 영어와 중국어 버전이 각각 필요할 때. 이런 경우에도 각각의 페이지를 다 따로 만들어 주어야 한다. 이는 매우 효율성이 낮은 반복 작업과 막대한 수정 비용을 요구하는 것이다. XHTML과 CSS 기반 사이트는 이러한 수고를 덜고 유지비용이 감소한다.

경량의 로딩 속도

XHTML 규격을 따르면 저절로 "구조화된" 문서로 만들어진다. 구조화된 문서의 특징은 "

가독성"이 높아지게 되는데 단지 생성된 코드를 사람이 읽기 편하다는 뜻뿐만 아니라, 다른 기계나 프로그램도 읽기 수월해 진다. 만약 디자인 부분을 CSS 파일로 분리해낸다면 훨씬 더 간단해진다. 지금까지 만들어온 일반적인 HTML 코드를 보면 아마도 실제 내용보다 디자인 요소가 차지하는 부분이 더 클 것이다. 문서 내 이러한 디자인 요소들은 파일의 용량이 키움에 따라, 웹페이지 로딩과 렌더링 속도를 느리게 하고 있다.

실제로 XHTML+CSS 레이아웃으로 첫화면을 개편한 미국 야후!닷컴은 기존과 똑같은 UI를 유지 하면서도 첫화면 HTML 파일 크기를 1/3 가량 줄었다. ESPN.com의 경우 50kb의 파일 크기가 감소했고, MSN.com과 Wired.com은 각각 64%, 62% 가량 줄었다. MSN.com의 경우 하루 940GB의 트래픽 감소 효과를 보았다.

이제 점점 더 많은 사이트들이 테이블을 사용하지 않고 CSS의 배치(layout) 기능을 이용하여 홈페이지를 재설계 하고 있다. W3C, Web Standards Project, Mozilla, WebAIM와 같이 비영리 사이트나 개인 사이트가 아닌 상업적인 홈페이지도 얼마든지 CSS를 이용해 디자인을 바꿀 수 있다는 실제적인 증거가 곳곳에서 늘어나는 것이다. 미국 야후와 한국 야후, ABC News, Novell: Suse Linux, Chevrolet, Disney Store UK가 그 대열에 동참했고, ESPN, SitePoint, Max Design, RedHat, Wired News, Opera 등은 오래 전부터 테이블을 사용하지 않고 설계되어 있었다. Macromedia사의 홈페이지가 테이블을 쓰지 않고 설계가 되을 뿐 아니라 Flash, 드림 위버 등 최근 제품들은 대부분 접근성을 고려한 기능이 상당히 정교하게 들어가 있다. Adobe사의 홈페이지도 오늘 확인해보니 완벽하진 않지만 테이블을 배제하고 CSS의 레이아웃 기능을 활용하여 디자인되어 있다. 이러한 추세는 앞으로도 계속 될 것으로 보인다.

뿐만 아니라 XHTML 사용은 웹표준을 지원하는 모든 현대적인 웹브라우저를 모두 지원함에 따라 사용자 층을 넓힐 수 있을 뿐 아니라, 검색 엔진이 접근하여 자료를 분석하고 색인 하는 데도 도움을 주므로 좋은 검색 결과를 얻어 미래의 사용자층을 확대할 수도 있다.

XTHML 문서 구조

올바른 DOCTYPE 사용

대부분의 웹페이지 들이 <html>로 시작하여 <head>와 <body> 태그를 사용하여 웹페이지를 표현한다. 그러나 웹페이지를 표현 하는 방식을 제대로 표현 하기 위해 웹 브라우저가 적절한 문서 형태를 표기하도록 할 필요가 있다. 즉 문서의 루트 요소 앞에는 공백 없이 DOCTYPE 선언이 있어야 하며, 이 선언은 XHTML에 대한 세 DTD(Document Type Definition) 파일(strict, transitional, frameset) 중 하나를 참조해야 한다.

올바른 문서 형식 선언을 해 주는 것은 다양한 브라우저에 따른 렌더링 차이를 최소화 할 수 있기 때문에 매우 중요하다. HTML 버전에 따라 해석되는 방식이 브라우저에 따라서도 다르기 때문에 이를 지정해 주는 것은 매우 중요하다.

1) HTML 2.0 표준 문서 형식

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

2) HTML 3.2 표준 문서 형식

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

3) HTML 4.01 표준 문서 형식

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

4) XHTML 1.0 표준 문서 형식

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

5) XHTML 1.1 표준 문서 형식

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

표준 문서 구조

표준 문서 형식(Doctype)을 기반한 웹페이지에 대한 정확한 사용법은 다음과 같다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko-KR">
<head>
<title> ... </title>
</head>
<body>
...
</body>
</html>
```

가장 흔히 사용되는 DOCTYPE 코드는 일반 형식(Transitional)과 엄격한 형식(Strict)으로 나누어 지게 된다. HTML 4.01 Transitional은 예전에 있었거나 없어진 태그도 지원하며, <font>에 지정된 스타일도 제대로 표현하여 준다. HTML 4.01 Strict은 HTML을 엄격하게 적용한다. <font> 태그에 적용된 스타일 보다는 CSS파일에서 지정된 스타일을 지켜 표현 해야 한다. 이러한 DOCTYPE 선언에 차이는 다음과 같다. 만약 웹사이트에 BODY {font-family: Helvetica, sans-serif; font-size: 200%;}와 같은 스타일을 지정한 다음 서로 다른 DOCTYPE을 지정한 경우 아래와 같이 표현된다.

DOCTYPE을 규정하는 가장 좋은 방법은 Strict 형식을 사용하는 것이다. 이것은 CSS를 통해 모든 HTML 태그의 속성을 모두 자유 자재로 규정할 수 있기 때문이다. 즉, b {font-weight: normal;} 라고 적는 다면 더 이상 <b>는 굵은체로 표시되지 않는다. 그러나, 아직 브라우저 호환성 때문에 <embed>나 비표준 태그를 사용해야할 필요가 있으므로 현재 상태에서 가장 최상의 브라우저 호환성을 제공해 주는 문서 형식은 XHTML 1.0 Transitional을 사용하는 것이다.

XHTML 일반 문법 준수

정확한 문서 구조 준수

문서의 루트 요소는 html이 되어야 하며, 이 html 요소는 XHTML 네임스페이스를 지정해야 한다.

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

표준 문서에는 head, title 및 body 구조 요소가 포함되어야 한다. 프레임 세트 문서에는 head, title 및 frameset 구조 요소가 포함되어야 한다

모든 요소는 완벽하게 중첩되어야 한다.

모든 요소들이 완벽하게 내포(nest) 되어야 하는 것은 필수적이다. 중첩(overlapping)이 부적합(illegal)한 것임에도 불구하고 기존 웹 브라우저들에서 널리 관대하게 사용되었다.

```
<p>This is a <i>bad example.</p></i>
<p>This is a <i>good example.</i></p>
```

모든 속성 값은 인용 부호(“나 ‘)로 묶어야 한다.

코드를 생성하거나 XHTML을 정리할 때 코드에서 속성 값을 인용 부호로 묶는다.

```
<a href=http://sample.com>틀린 경우</A>
<a href="http://sample.com">맞는 경우</a>
```

모든 요소와 속성은 소문자여야 한다.

XHTML 코드를 생성하거나 정리할 때 태그 및 속성의 대/소문자 환경 설정에 상관 없이 XHTML 코드에서 HTML 요소 및 속성의 이름을 소문자로 강제로 설정해야 한다. 이러한 차이는 XML은 대소문자를 구별(case-sensitive)하므로 필수적이다. 예를 들어, 와 <LI>는 서로 다른 태그들이다

```
<A HREF="http://sample.com">틀린 경우</A>
<a href="http://sample.com">맞는 경우</a>
```

모든 요소는 닫아야 한다.

DTD에서 EMPTY로 선언된 경우를 제외하고 모든 요소에는 종료 태그가 포함되어야 한다. 코드를 생성하거나 XHTML을 정리할 때 코드에 닫기 태그를 삽입한다.

빈 요소에는 종료 태그가 포함되거나 시작 태그가 />로 끝나야 한다. 예를 들어,
은 잘못된 것이며
</br> 또는
이 올바른 형식이다. 빈 요소로는 area, base, basefont, br, col, frame, hr, img, input, isindex, link, meta 및 param이 있다.

또한, XML을 사용할 수 없는 이전 브라우저와의 호환성을 위해 /> 앞에 공백이 있어야 한다(예:
가 아니라 <br />).

```
<img src="http://sample.com/wrong.jpg">
<img src="http://sample.com/right.jpg" />
```

모든 속성값은 속성이 함께 선언되어야 한다.

모든 속성은 최소화되어 표기 되면 안 된다. XML은 속성의 최소화를 지원하지 않는다. 속성 값의 짝들은 모두 작성되어야 한다.

a, applet, form, frame, iframe, img, map 등의 요소에는 name 속성뿐만 아니라 id 속성도 있어야 한다. 예를 들어, Introduction는 잘못된 것이며 Introduction 또는 Introduction.가 맞다.

또한 <td nowrap>은 잘못된 것이며 <td nowrap="nowrap">이 올바른 형식이다. 최소화될 수 없는 속성으로는 checked, compact, declare, defer, disabled, ismap, multiple, noresize, noshade, nowrap, readonly 및 selected가 있다.

```
<option value="wrong" selected>틀린 경우</option>
<option value="right" selected="selected">맞는 경우</option>
```

참고: HTML 브라우저에서 HTML 4를 지원하지 않는 경우, 부울 속성이 전체 형식으로 표시되면 브라우저에서 이들 속성을 해석하지 못할 수도 있다.

모든 script 및 style 요소에는 type 속성이 포함되어야 한다.

language 속성이 사용되지 않는 HTML 4 이후로는 script 요소의 type 속성을 반드시 지정해야 한다. 코드를 생성하거나 XHTML을 정리할 때 script 요소에서 type 및 language 속성을 설정하고 style 요소에서 type 속성을 설정한다.

```
<script type="text/javascript” language="javscript"></script>
<style type="text/css"></style>
```

모든 img 및 area 요소에는 alt 속성이 포함되어야 한다.

코드를 생성하거나 XHTML을 정리할 때 코드에서 이들 속성을 설정하고, 찾을 수 없는 alt 속성을 보고한다.

모든 SCRIPT내의 태그는 Escape 시켜야 한다.

자바 스크립트에서 HTML 태그 쓰기에서 많은 경우 오류를 낸다. 자바스크립트 내에 데이터는 CDATA 형식으로 간주되기 때문에 HTML태그가 들어가게 되면 오류를 내게 되어 있다. 예를 들어 아래 예제는 잘못된 방식이다.

```
<script type="text/javascript">
<!--
  // 틀린 표현!
  document.write("</P>");
// -->
</script>
```

HTML4에서는 SCRIPT내에 데이터 중 시작 태그나 코멘트 부분은 인식이 안되지만 종료 태그는 인식이 되기 때문에 이를 역슬래시로 표시해야 한다.

```
<script type="text/javascript">
<!—
// 맞는 표현!
  document.write("<\/P>");
// -->
</script>
```

XHTML에서, 스크립트와 스타일 요소들은 #PCDATA 컨텐트를 갖는 것으로 선언된다. 결과적으로, <과 &는 마크업의 시작으로 처리되고, <과 &와 같은 개체(entities)들은 XML 프로세서(processor)에 의해 각각 <과 &로의 개체 참조로서 인식되므로 CDATA로 마크업 하여 표시하는 게 좋다.

```
<script type="text/javascript">
   <![CDATA[
    ... <h1>데이터</h1> ...
   ]]>
   </script>
```

모든 문서 내 URL에서 &를 쓰면 안 된다.

URL에 &가 포함되어 있는 경우 에러를 낼 수 있다. 이것은 &가 XML 엔티티의 시작으로 인식 하기 때문에 생기는 문제이다. 기존 웹브라우저는 이러한 에러를 복구해 주고 있지만 유효성 검사기에서는 에러를 내게 된다.

```
<!—에러! --> <a href="foo.cgi?chapter=1&section=2">...</a>
<!—적합! --> <a href="foo.cgi?chapter=1&amp;section=2">...</a>
```

HTML 문서 내에서만 &를 &로 바꾸어야 하며 브라우저 주소창이나 이메일 본문에서는 &를 써야 한다. 웹 서버에서는 &가 아니라 &만을 인식하기 때문이다.

구조적 XHTML 사용 방법

잘못 사용하고 있는 태그

XHTML에는 컨텐츠의 의미를 나타내는 많은 태그들이 있다. 간단하게는 문단은 나타내는 <p>부터 사용하는 것을 많이 보지 못했던 <var> 등에 이르기 까지 다양한 태그들이 있다. 그러나 많은 수의 사람들이 이러한 다양한 태그를 사용해서 문서의 메타 데이터를 풍부하게 하기 보다는 <table>만을 사용해서 시각적인 것만을 고려 하여 웹 페이지를 제작한다.

무분별한 테이블 사용

```
<table width="640" cellpadding="0" cellspacing="0" border="0">
<tr>
   <td height="25" valign="top">
     <img src="title_01.gif">
   </td>
</tr>
<tr>
     <td>
레이아웃에 table element 를 사용하는 이유는 아마도 쉽기 때문일 것이다. 쉽다는
것 보다는 "익숙해서" 가 더 큰 이유일지도 모른다. 대다수의 사람들은 처음부터 아주
당연하게 table 을 사용해 왔고 그렇게 되어 있는 사이트를 더 많이 봤을 것이다.
   </td>
</tr>
</table>
```

보통의 웹사이트에서 위와 같은 코드는 아주 쉽게 찾아 볼 수 있다. <table>은 행과 열이 있는 2차원의 데이터를 표시 하는 데에 사용되는 태그 이다. 그러나 위와 같이 단순히 제목과 문단을 구분 하는 데에도 표를 사용하는 것을 많이 볼 수 있다. 이와 같은 데이터는 표가 아니기 때문에 <table> 태그를 사용해서는 안되고 의미에 맞는 <h1>이나 <p>와 같은 태그를 사용해 주어야 한다.

잘못된 위치, 태그, 스크립트 사용

```
<table cellpadding="0" cellspacing="0">
<form action="/search/" method="get">
<tr>
<td>
<select>
<option>제목</option>
<option>내용</option>
<option>작성자</option>
</select>
</td>
<td>
<input type="text" size="10" class="type-text" />
</td>
```

```
<td>
<a href="javascript:search()"><img src="button_search.gif"></a>
</td>
</tr>
</form>
</table>
```

<form>을 사용해서 제작하다 보면 디자인상에서 나타나서는 안되는 공백이 생기는 것을 볼 수 있다. 이와 같은 현상을 피하기 위해서 많은 사람들이 <form>태그를 <tr>태그 사이에 넣는다. 하지만 validator를 이용해 보면 이와 같은 사용은 잘못된 문법 오류라는 것을 알 수 있다. <form>의 여백을 없애기 위해서 스타일을 사용하면 아주 간단하게 문제를 해결 할 수 있다.

```
form {    margin: 0; }
```

<form>에 마진을 없앰으로서 간단하게 여백을 없앨 수 있고 위와 같이 <tr>사이에 <form>태그를 넣는 잘못된 xhtml사용을 피할 수 있다.

또한 많은 사람들이 <form>의 서밋을 javascript로 하는 것도 잘못된 xhtml의 사용이다. <form>의 서밋을 하기 위해서는 <input type="submit">을 사용하면 되고 일반적으로는 이미지를 서밋버튼으로 사용하기 때문에 <input type="image">를 사용하게 된다. 하지만 상당수의 사람들이 이와 같은 방법을 사용하지 않고 <img>태그를 이용해서 이미지 버튼을 삽입하고 이것에 <a>태그와 javascript를 이용해서 서밋을 하는 방법을 사용한다. 첫째로, 이 방법은 javascript가 작동하지 않는 상황에서는 서밋이 일어날 수가 없기 때문에 접근성을 아주 떨어뜨리는 방법이다. 그리고 href의 값도 올바른 값이 아니다. href는 하이퍼텍스트 주소를 값으로 할 수가 있는데 위에서 사용된 "javascript:search()"는 올바른 하이퍼텍스트 주소가 아니다.

그리고 위의 경우는 단순히 <select>와 텍스트를 입력받는 <input>, 그리고 서밋 버튼으로 이루어져 있는데 이를 각각<td>안에 넣는 것도 불필요한 <table>의 사용이다. 위의 코드를 간결한 xhtml로 바꾸면 아래와 같다.

```
<form action="/search/" method="get" onsubmit="validation(this)">
<div class="search">
<select>
<option>제목</option>
<option>내용</option>
<option>작성자</option>
</select>
<input type="text" size="10" class="type-text" />
<inpyt type="image" src="button_search.gif" alt="검색" />
</div>
</form>
```

불필요한 테이블을 없애고 중앙정렬이나 여백을 없애는 것과 같은 디자인적인 요소를 CSS로 처리 함으로써 아주 간결하고 의미에도 맞는 마크업을 할 수 있다.

XHTML을 작성함에 있어서 가장 중요하게 생각하고 있어야 하는 것은 "의미에 맞는 태그를 사용"하는 것이다.

그룹 요소: div, span

페이지를 제작할 때에는 각각의 의미를 갖고 있는 컨텐츠를 묶거나 표시를 해줄 필요가 있다. 이러할 때에 사용하는 태그가 와 이다. 가 레이어를 만드는 것에 사용되는 태그인 것으로 알고 있는 사람이 있을 정도로 많은 사람들이 <table>에 익숙해져 있는 것이 사실이다. 그리고 <table>을 사용하지 않고 페이지를 제작하는 사람들을 보면 생각의 틀은 바뀌지 않은 채 <table>을 단순히 로 옮기는 식으로 제작을 하는 경우도 있다. 의미에 맞는 태그를 사용해서 페이지를 제작한 다음에는 나 을 이용해서 단위별로 구분을 해 주어야 한다.

block과 inline

와 은 둘다 그루핑에 사용되는 엘리먼트이지만 는 block이고 은 inline이라는 큰 차이점이 있다. 이 block과 inline은 화면에서 엘리먼트들이 랜더링 되는 기본 모양을 지칭한다. block과 inline은 쉽게 생각하면 이 엘리먼트 뒤에 개행이 이루어지는지로 구분 할 수 있다. 를 이용하면 뒤에 개행이 이루어 지고 을 이용하면 개행이 이루어 지지 않는다.

```
<div class="name">홍길동</div> <div class="age">24 세</div>
```

위의 코드를 화면에서 보면 "홍길동" 다음에 줄이 바뀌어서 "24세"라고 나오는 것을 볼 수 있다.

```
<span class="name">홍길동</span> <span class="age">24 세</span>
```

하지만 위와 같이 작성을 하게 되면 "홍길동"뒤에 개행이 되지 않고 한줄에 "홍길동 24세"와 같이 나오는 것을 볼 수 있다.

, <form>, <ul>, <ol>, <li>, <dl>, <dt>, <dd> 등이 block들이고 <a>, <img>, <select>, <input> 등이 대표적인 inline 엘리먼트들 이다.

표제(Heading)

Heading 태그는 각 부분에서 중요한 역할을 하는 제목들을 표시할 때에 사용하게 된고, <h1> ~ <h6> 까지 6단계의 heading태그가 있다. 페이지의 한 부분에서 가장 중요한 제목을 <h1>으로 묶어주게 되고 그 아래의 중요도를 갖는 것은 <h2>, <h3>등을 이용해서 제목을 표기해 주게된다.

주의할 것은 <h1>의 하위 heading은 <h2>여야지 <h3>나 <h4>와 같이 단계를 건너뛰는 것은 좋지 않다.

문단(paragraph)

<p>태그는 문단을 나타낼 때에 사용한다. 어느 웹스타일 가이드를 보면 브라우져 별로 <p>태그의 간격이 다르기 때문에 <p>태그를 사용하면 안되고
두번을 사용하라는 경우가 있는데 이는 마크업을 비주얼 적인 요소로 잘못 이해한 것이다. 문단은 <p>로 구분을 하고 간격을 CSS로 제어하게 되면 브라우져와 상관없이 일관된 디자인을 유지할 수

있다.

를 사용할 때에는 반드시 닫는 태그도 사용해 주어야 한다.

```
<p>        첫번째 문단
<p>        두번째 문단
<p>        세번째 문단
```

위와 같이 단락의 경계를 로 구분 하는 것이 아니라 하나의 문단을 와 로 감싸는 식으로 작성을 해야 한다.

```
<p>첫번째 문단</p>
<p>두번째 문단</p>
<p>세번째 문단</p>
```

그리고 는 하위에 block요소를 포함 할 수가 없다. 텍스트나 이미지와 같은 inline요소만을 하위에 포함 할 수 있기 때문에 주의해서 사용해야 한다.

구문(em, strong, dfn, code, samp, kbd, var, cite, abbr, acronym)

강조 (strong, em)

과 은 문장안에서 강조를 나타낸다. 보통의 브라우저에서 은 이탤릭으로, 은 볼드체로 나타나게 된다. 문장안에서 중요도가 있을 때에는 이 과 을 사용해야 하고 단지 이탤릭이나 볼드를 표현하고자 할 때에는 <i>와 <b>를 사용해야 한다. 스크린리터중에는 이나 태그를 사용하면 그 부분을 큰소리로 강조해서 읽어주는 제품도 있다.

정의

<dfn>태그는 정의를 나타낼 때에 사용된다.

코드

<code>는 컴퓨터 코드를 나타내는 태그이고 <samp>는 코드의 결과 출력물을 나타낼때에 사용한다.

값의 표시

<kbd>는 유저의 키보드 입력을 나타내고, <var>는 프로그램에서의 변수를 나타낸다.

<kbd>Enter</kbd>키를 누르세요

출처

<cite>는 인용이나 출처를 밝힐 때에 사용한다.

축약

<abbr>는 축약어를 나타내고 <acronym>은 두문자어를 나타낸다. 이 두가지는 상당히 혼동하기가 쉬운데 보통 그 약어를 그대로 발음하는 것은 <acronym>을 사용하고 한글자씩 읽는 것은 <abbr>을 사용한다.

인용(blockquote, q)

다른 인용문을 표시할 때에 사용한다. <blockquote>는 block요소의 인용문이고 <q>는 inline요소의 인용을 나타낸다. 보통 <blockquote>는 인덴트를 해서 보여주게 되고 <q>는 인용문 앞 뒤로 따옴표를 나타내 준다.

첨자(sup, sub)

위첨자나 아래첨자를 나타내고자 할때 하용한다.

```
x<sup>2</sup> + 4x + 4 = (x+2)<sup>2</sup>
```

형식을 가지고 있는 컨텐츠 (pre)

미리 형식을 가지고 있는 내용을 나타내고자 할 때에는 <pre> 태그를 사용한다. 이 태그를 사용하게되면 공란도 갯수에 맞게 다 나오고 글자폭이 일정한 폰트로 화면에 나오게 된다. 소스 코드등을 나타낼 때에 많이 사용한다. 단, 자동 줄바꿈이 되지 않기 때문에 너비가 제한 적일 때에는 주의해서 사용해야 한다.

```
<pre class="code">function menuOn(imgEl)
{
imgEl.src = imgEl.src.replace(&quot;.gif&quot;,
&quot;_on.gif&quot;);
}
function menuOut(imgEl) {
imgEl.src = imgEl.src.replace(&quot;_on.gif&quot;,
&quot;.gif&quot;);
}</pre>
```

인덴트나 개행을 위해서 다른 태그등을 사용하지 않아도 화면에 그대로 랜더링 되는 것을 볼 수 있다.

추가 및 삭제(ins, del)

문서에 추가된 내용이나 삭제된 내용을 명시할 때에 사용한다.

문서에 <ins>새로 추가</ins>되거나 ~~삭제된 내용~~을 표시 할 수 있습니다.

보통 <ins>는 밑줄을, <del>은 취소선으로 표현이 된다.

목록 (ul, ol, dl)

리스트에는 <ul>, <ol>, <dl>세가지가 있다.

Unordered List

은 하위로 <li> 엘리먼트를 갖게 되고 각 엘리먼트의 앞부분에는 불렛이 나타나게 된다.

Ordered List

은 하위로 <li> 엘리먼트를 갖게 되고 각 엘리먼트의 앞부분에는 자동적으로 숫자가 나오게 된다.

Definition List

<dl>은 하위로 <dt>와 <dd>엘리먼트를 갖게 된다. <dt>는 term을 <dd>는 definition을 나타낸다.

실전 CSS 레이아웃

CSS 제대로 사용하기

현재 대부분의 사이트에서 CSS는 폰트나 링크 스타일에 대한 정의 정도로만 사요이 되고 있다. 이러한 사용도 사이트의 디자인을 효과적으로 관리하고 제작하는 것에 도움이 되고 있지만, CSS의 가장 큰 장점은 문서의 구조와 디자인의 분리에 있다. CSS를 폰트에 대해서만 제한적으로 사용하게 되면 HTML에 레이아웃 이라든가 배경이미지 같은 디자인 적인 요소가 많이 들어갈 수 밖에 없다. 여기서는 이러한 디자인 적인 요소를 HTML에서 분리해 냄으로서 얻어질 수 있는 여러가지 장점에 대해서 알아본다.

웹은 초기에 다양한 환경의 사용자들이 환경에 영향을 받지 않으면서 정보나 자료를 서로 쉽게 교환을 할 수 있게 하기 위해서 만들어 졌다. 웹이 처음 만들어질 당시에는 지금과 같은 인터넷이라는 것도 없었고 지역적으로 멀거나 사용하는 컴퓨터 기종이 다를 경우 상호간에 자료를 교환하는 것이 쉬운일이 아니었다. 이 불편한 상황을 극복하기 위해서 웹이라는 것이 고안되고 계속 발전을 해서 지금과 같은 엄청난 규모의 정보 전달 매체로 만들어지게 된 것이다. 자료를 손쉽게 교환하기 위해서 정보를 기록하는 공통된 약속이 필요했고 이 약속이 발전한 것이 우리들이 현재 사용하고 있는 HTML, XHTML 등 이다.

HTML 의미 바로 알기

HTML은 HyperText Markup Language의 약자 이다. HyperText 는 일반적으로 많이 알고 있는 것 이다. 쉽게 얘기 하자면 링크로 연결되는 문서를 의미 한다. 보통의 종이에 인쇄되어 있는 책들과 비교해 볼 때 HyperText 는 많은 장점을 가지고 있다. 문서들끼리 서로 연결 되어 있기 때문에 참조나 설명이 용이 하고 자신이 원하는 내용으로의 이동이 아주 간편하다. 우리는 HyperText의 장점을 잘 살려서 Web Site 라고 하는 정보의 집합체이자 효과적인 정보 전달을 할 수 있는 구조적인 문서를 만들어 낼 수 있는 것이다.

HTML에서 한번쯤은 생각해 봐야 하는 것이 HTML은 markup language라는 것이다. 출판이 전산화 되기 전에는 글을 쓰는 사람이 원로글 작성하고 자신의 원고에 표현하고자 하는 내용의 의미과 형식에 대해서 표기를 하였고 이것을 바탕으로 인쇄를 할 활자를 짰다. 여기서 저자가 원고에 내용의 의미를 표기하는 작업을 "마크업한다(mark it up)." 라고 했다. 즉 마크업이라는 것은 눈으로 보이는 글자 외에 원고에서 특정 부분의 의미가 무엇인가를 설명하고 그것을 표기하는 것을 말한다. 이 표기는 문서의 표현 형식 - 페이지를 나눈다거나 글씨의 모양을 지정하는 - 일 수도 있고, 특정 부분이 제목이나 문단과 같이 문서내에서 뜻하는 것이 어떠한 것이라는 것의 설명일 수도 있다. 즉, 우리들이 문서를 작성을 할 때에는 글자 모양외에도 문서가 가지고 있는 정보를 확실하게 전달하기 위해서 겉으로는 보이지 않지만 추가적인 정보를 기술해 주어야 한다.

그러나 보통 HTML 문서를 작업할 때에는 작업자들이 HTML을 markup 하는 용도로 사용하기 보다는 겉으로 어떻게 문서가 보이는 지 만을 나타내기 위해서 사용하고 있다. 사실 더 직설적으로 말하자면 대부분의 HTML 작업자 들은 HTML이 markup이라는 것을 간과하고 무의식 중에 <table cellpadding="0" cellspacing="0"> 부터 적기 시작한다. 실제로 HTML 작업자에게 "H1 태그가 무엇인가요?" 라고 물어보면 "매우 크고 두꺼운 폰트" 라고 답변하는 사람이 있을 것이다. 그 사람에게 "H2 태그가 무엇인가요?" 라고 질문을 하게 되면 아마도 "두번째로 크고 두꺼운 폰트" 라는 답변을 하게 될 것이다. 그리고

"그러한 태그를 사용하신 적이 있습니까?" 라고 물으면 "처음에 배울때에는 사용했었는데 글씨가 두껍고 못생겨서 요즘에는 사용하지 않다." 라고 할 것이다. 물론 완전히 틀린 말은 아니다. 실제로 이 태그들을 사용해서 페이지를 만들게 되면 크고 두꺼운 글씨들이 나타나고 그렇게 H1~H6태그를 설명하는 책들도 서점에서 찾아볼 수 있다.

하지만 HTML은 마크업 언어이고 문서의 구조를 나타내는 언어이기 때문에 이러한 답변보다는 "H1은 해당 부분에서 가장 중요한 타이틀이고 H2는 H1바로 하위의 타이틀을 나타내고 H3~6 등은 하위 뎁스의 타이틀이다. 그리고 글씨의 모양은 제가 원하는 대로 CSS를 이용해서 조절을 한다." 라는 답변을 해야 - 혹은 알고 있어야 - 프로 HTML 작업자라고 할 수 있을 것이다. 어느순간부터 대다수의 웹사이트 제작자들은 HTML 태그들이 구조적인 의미를 갖는 다는 것은 잊어 버리고 화면상에서 원하는 대로 보이게 하기 위해서 HTML언어를 사용하고 있다. 대다수의 사람들이 많은 구조를 나타내는 태그들은 사용을 안하고 오직 <table>, <img>, <form> 태그들만 이용해서 웹페이지를 제작하고 있는 것이 현실이다.

HTML과 CSS의 관계

HTML은 컨텐츠의 내용과 구조를 표시 한다. 컨텐츠의 내용은 텍스트와 <img> 엘리먼트 등으로 나타나게 되고 이러한 컨텐츠가 하나의 문단을 이루고 있는지, 인용문인지, 리스트 형태 인지에 따라서 각각 <p>, <blockquote>, <ul> 등의 의미를 나타내는 엘리먼트로 표기한다. 그리고 각 컨텐츠들이 페이지에서 어떠한 의미를 갖는지 엘리먼트와 적절한 id, class 속성으로 표기한다. 이렇게 작성된 markup은 사용자 뿐만이 아니라 검색엔진과 같이 페이지에서 내용의 의미를 파악하는 기계들도 쉽게 이해 할 수 있는 언어가 된다. 이렇게 나타내고자 하는 컨텐츠를 의미에 맞게 기술하고 웹페이지로의 접근성을 높일 수 있는 방법으로 markup을 만드는 것이 웹표준의 목적이기도 하다. 그리고 이렇게 제작된 페이지를 어떻게 표현해 내는지에 대한 것은 CSS에서 담당하게 된다.

CSS를 익히고 접근할때 사람들이 가장 흔하게 하는 실수가, CSS라는 것이 기존과는 다른 새로운 것이기 때문에 CSS에만 너무 집중하는 것이다. CSS에 너무 집중을 하게 되면 CSS를 사용하는 원래의 목적에서 멀어지고 컨텐츠의 의미도 제대로 전달하지도 못하고 제작만 더 어렵고 크로스 브라우징도 잘 되지 않는 애물단지의 페이지가 만들어 지게 된다.

CSS의 가장 큰 목적은 문서의 내용과 문서의 표현을 분리하는 것에 있다. 문서의 내용은 HTML로 작성 하고 문서의 표현은 CSS를 이용해서 나타내는 것이다. CSS에만 너무 촛점을 맞추다 보면 문서의 내용을 기술하는 HTML을 소홀이 하기 쉽다. CSS를 처음 접할때 CSS를 적용하는 것이 주 목적이 아니라 컨텐츠를 의미에 맞게 기술하기 위한 HTML 작성을 위해서 CSS를 사용한다는 것을 항상 염두에 두어야 한다.

CSS 개념 및 소개

CSS(Cascading Style Sheet)란 무엇인가?

CSS 는 구조적으로 짜여진 문서(HTML, XML)에 style (글자, 여백, 레이아웃) 등을 적용하기 위해서 사용하는 language이다. CSS는 문서의 구조와 디자인을 분리할 수 있게 해 줌으로써 웹 제작이나 유지관리를 간단하게 해 준다. 또한 미디어 (화면, 프린트, 보이스머신 등) 별로 스타일을 적용 할 수 있기 때문에 각 기기별로 다른 스타일이 적용된 모습을 만들 수 있다.

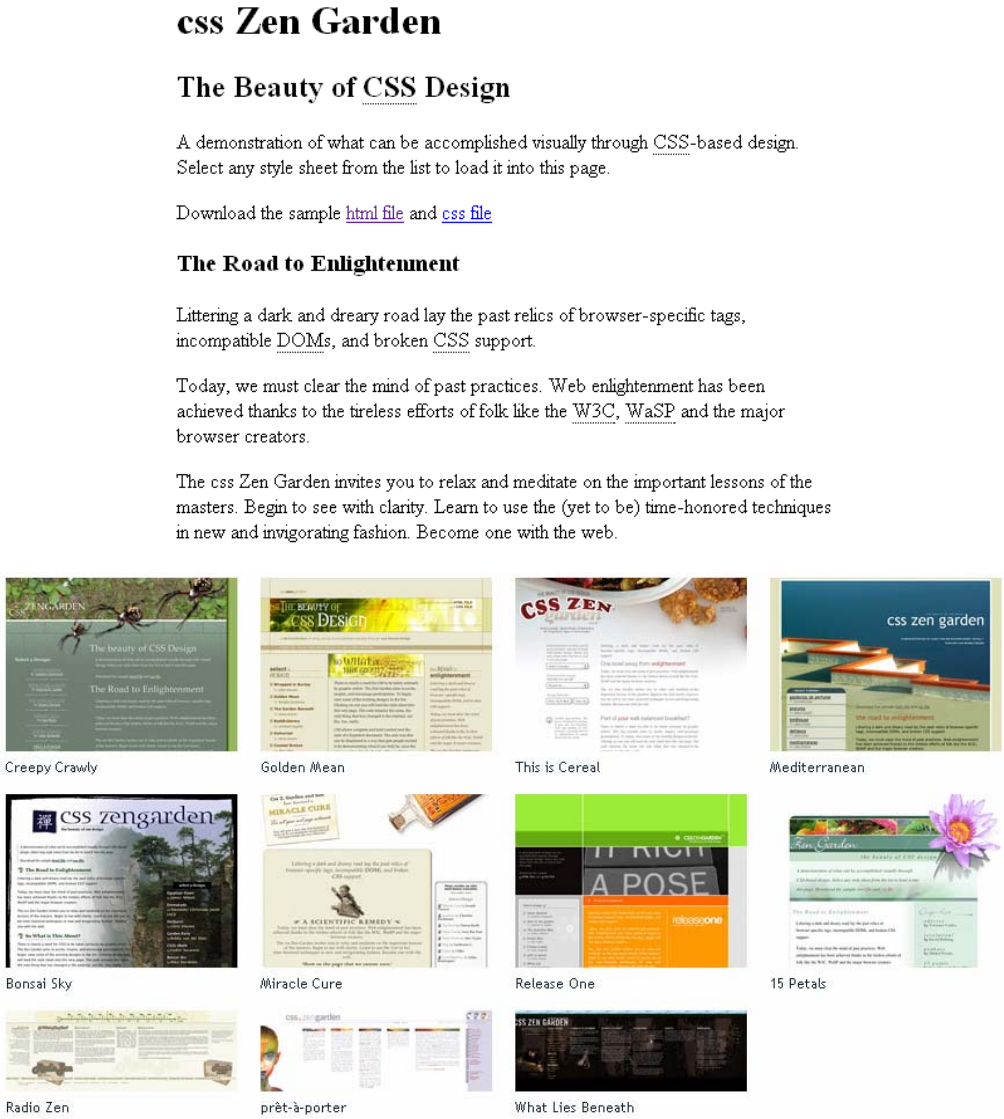


그림 10 CSS Zen Garden은 하나의 HTML에 스타일 변경 만으로 다양한 디자인을 선보이고 있다.

이러한 특징을 잘 보여주는 사이트로 css zengarden (http://csszengarden.com) 사이트가 있다. 위의 사이트들은 다 전혀 다른 사이트 같지만 실제적으로는 동일한 사이트이다.

동일한 HTML을 사용하면서 CSS 만 바꿔서 사이트의 디자인을 다르게 표현한 것이다. 내용의 구조와 디자인이 완벽하게 분리되어 있기 때문에 디자인 요소인 CSS만 바꾸어도 다른 사이트처럼 보이게 할 수 있게 된다.

CSS의 선언

```
p.text {
    margin: 0;
}
```

CSS의 선언(rule)은 선택자(selector)와 선언부로 이루어져 있다. 위의 rule에서 "p.text" 부분이 바로 selector이고 그 뒤에 나오는 {}안의 내용이 선언부이다. selector에 해당하는 HTML엘리먼트에 선언부에 명시된 것과 같은 스타일을 적용하라는 의미로 되어 있다.

```
p.text,
span.name {
    color: #666;
}
```

selector들은 ,(comma)를 이용하여 구분할 수 있다. "p.text"와 "span.name"은 ,(comma)로 구분되어 있다. 이경우 "text"라는 class를 갖는 <p> 엘리먼트와 "name"이라는 class를 갖는 엘리먼트는 둘다 폰트의 색이 rgb #666666으로 나타나게 된다.

```
h1 { font-weight: bold }
h1 { font-size: 3em }
h1 { color: #333 }
h1 { margin: 1.5em 0 1em }
h1 { padding: 0 0 0 8px }
```

동일한 selector에 서로 다른 선언들이 있을 경우 이를 하나의 셀렉터에 ;(semi-colon)으로 구분하여 선언 할 수 있다. 위의 선언과 아래의 선언은 동일하다.

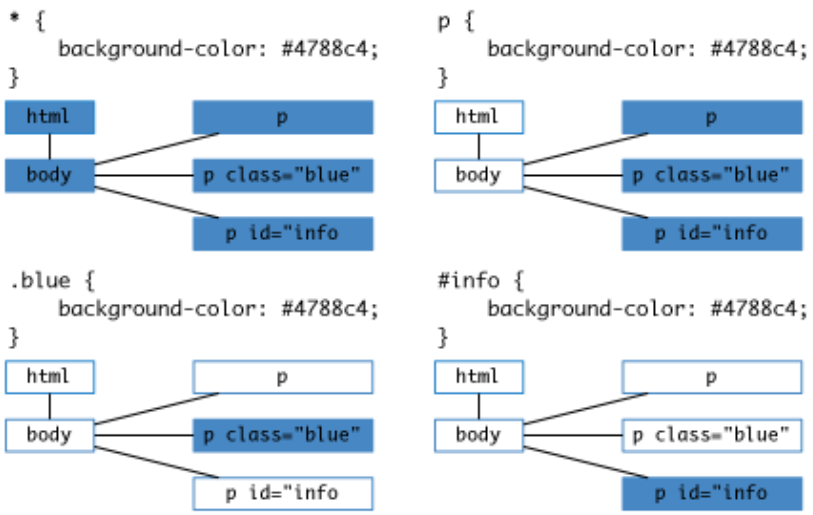
```
h1 {
    font-weight: bold;
    font-size: 3em;
    color: #333;
    margin: 1.5em 0 1em;
    padding: 0 0 0 8px;
}
```

CSS 선택자(Selector)

CSS는 속성을 잘 알아서 사용하는 것도 중요하지만 구조화된 문서에 효과적으로 속성을 적용하기 위해서는 selector를 정확히 이해하고 사용해야만 한다. CSS를 풍부하게 사용하다보면 "내가 만든 CSS class 가 다른 사람의 것과 겹치면 어떻게 하지?", 또는 "CSS class 가 너무 많아서 복잡하고 이름 정하기가 힘들어." 라는 생각을 하게 된다. 하지만 CSS selector를 잘 알고 능숙하게 사용하게 되면 오히려 이러한 고민들이 보다 구조화된 문서를 제작하는 것에 도움을 주게 된다.

일반 선택자

일반 선택자는 네 가지 종류가 있다.



공용 선택자	*	모든 태그를 지정
타입 선택자	A	태그 A를 지정
클래스 선택자	.A	클래스가 A인 태그를 지정
ID 선택자	#A	아이디가 A인 태그를 지정

그림 11 일반 선택자 개념도 (출처: http://andsite.net)

대부분 한번쯤은 접해 본 타입, 클래스, ID 선택자는 CSS1 에서 채택되었고, 이후 공통 선택자와 다중 클래스가 CSS2 에서 추가되었다. 안타깝게도 윈도용 인터넷 익스플로러 6(IE6/win)는 다중 클래스를 제대로 지원 못해 가장 마지막 클래스만 인식한다.

이 오류가 안타까운 이유는 다중 클래스를 쓸 수 있다면 CSS 에서 반복된 코드를 획기적으로 줄일 수 있기 때문이다. 예를 들어 사이트에 전반적인 색깔을 .main-color, .sub-color 로 정하고 폰트 크기를 .main-size, .info-size 로 정한 후에 <p class="main-color info-size">로 지정하면 두가지 속성을 동시에 적용할 수 있어서 중복된 코드가 필요 없어지기 때문이다. (출처 http://andsite.net/)

*(공용 선택자)

모든 엘리먼트를 선택한다.

```
* {
   margin: 0;
   padding: 0;
}
```

이와 같은 선언을 할 경우 페이지 내의 모든 <h1>, <h2>, <p>, <form>, <blockqoute> 등의 브라우저 기본 마진과 패딩을 갖고 있는 엘리먼트들이 여백이 없어지게 된다.

```
div.search * {
   vertical-align: middle
}
```

이 경우 div.search안의 모든 엘리먼트가 세로로 가운데 정렬이 되게 된다.

E (타입 선택자)

E엘리먼트를 선택한다. 예를 들어서 "body"와 같이 사용하면 body 엘리먼트를, “div”와

같이 사용하면 div 엘리먼트를 선택한다.

.E (클래스 선택자)

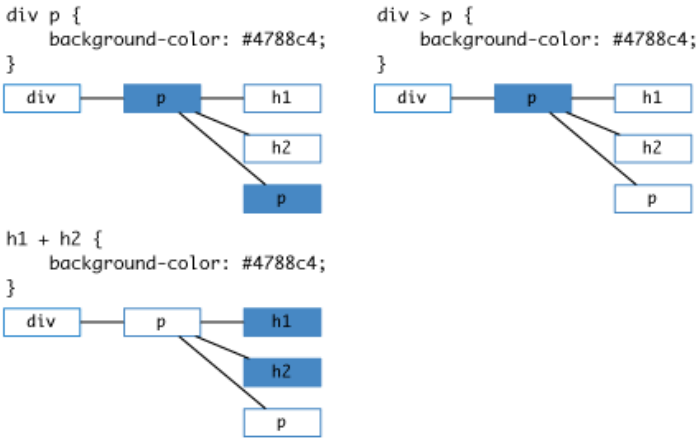
HTML에서만 사용할 수 있으며, warning이라는 class를 갖고 있는 DIV 엘리먼트를 선택한다. class는 하나의 페이지에서 여러번 사용할 수 있기 때문에 반적적으로 여러번 나오는 스타일의 경우 class를 지정해서 정의하여 사용하게 된다.

#E(아이디 선택자)

ID가 myid인 E 엘리먼트를 선택한다. descendant selector와 같이 사용하여 우선순위의 조정에 많이 사용된다. id를 사용하지 않고 class만을 사용하게 되면 다른 페이지나 다른 정의에서 사용된 셀렉터와 겹치게 될 수가 있다. 이를 방지하기 위해서 id를 하나 선언을 하고 descendatd selector를 사용하여 CSS를 정의하면 우선순위를 확실하게 구별할 수 있기 때문에 이러한 염려를 하지 않아도 된다.

복합 선택자

복합 선택자는 크게 세 가지 종류가 있다.



하위 선택자	A B	태그 A로 감싸져 있는 태그 B를 지정
자식 선택자	A > B	태그 A로 감싸져 있는 태그 B중 한 단계 밑에 것을 지정
인접 선택자	A + B	태그 A와 B가 연속으로 나와 있는 것을 지정

그림 12 복합 선택자의 종류 (출처: http://andsite.net/)

CSS1 에서 복합 선택자가 CSS2 로 오면서 하위 선택자로 바뀌고 자식, 인접 선택자가 추가됐다. 복합 선택자를 잘쓰면 불필요한 클래스의 남발을 막을 수 있다. 예를 들어 #header h1 은 머릿말 부분의 제목이고 #footer h1 은 꼬릿말 부분의 제목으로 각각의 의미가 분명한데, 굳이 <h1 class="header-title">나 <h1 class="footer-title">로 쓸 이유가 없는 것이다.

IE6/win 는 CSS2 에서 추가된 부분은 지원하지 않고, 그나마 지원하던 하위 선택자 역시 공통 선택자로 시작할 경우 첫 공통 선택자를 무시한다 2. 이 오류로 IE6/win 는 *

html 은 html 로, * * body 는 * body 로 인식하게 되는데, 때때로 IE6/win 의 여러가지 오류들을 고치기 위한 편법으로 쓰인다. (출처: http://andsite.net/)

E F (하위 선택자)

E 엘리먼트의 하위에 있는 F 엘리먼트들을 선택한다. 보통 ID Selector와 함께 사용하여 중복 선언을 피하고자 할때 많이 사용한다. 또한 불필요하게 class를 많이 적어주지 않아도 많은 하위 엘리먼트를 한번에 선택 할 수 있기 때문에 유용하고 많이 사용한다.

```
<ul id="list">
    <li><a href="list.html?id=34&amp;type=blah">item 34</a></li>
    <li><a href="list.html?id=35&amp;type=blah">item 35</a></li>
    ...
    ...
    <li><a href="list.html?id=99&amp;type=blah">item 99</a></li>
</ul>
```

위와 같은 코드에서 a 엘리먼트에 스타일을 적용하고자 할 때 일일이 a 엘리먼트에 class를 기입하지 않아도 아래와 같이 하면 모든 하위 a 엘리먼트에 스타일을 적용 할 수 있다.

```
ul#list a:link,
ul#list a:visited {
    color: #999;
}
ul#list a:hover,
ul#list a:active {
    color: #000;
}
```

E > F (자식 선택자)

E 엘리먼트의 자식 엘리먼트인 F 엘리먼트를 선택한다. descendant selector의 경우 하위에 있는 모든 F 엘리먼트를 선택하는데 비해 child selector는 바로 하위에 있는 F 엘리먼트만을 선택해 온다.

사이트 맵과 같은 중첩된 을 사용할때 유용하게 사용할 수 있다.

```
<ul class="depth1">
   <li>
           <a href="about.html>Company</a>
           <ul class="depth2">
                   <li>
                           <a href="overview.html">Overview</a>
                   </li>
                   <li>
                           <a href="ceo.html">Ceo.html</a>
                   </li>
           </ul>
   </li>
</ul>
```

위의 코드 에서 ul.depth1과 ul.depth2의 에 스타일을 적용할 경우,

```
ul.depth1 li {
   background: #f9f9f9;
   border-bottom: 1px solid #ddd;
}
```

이렇게 정의 하게 되면 ul.depth2의 까지 스티일이 적용 되는 것을 볼 수 있다.

```
ul.depth1>li {
   background: #f9f9f9;
   border-bottom: 1px solid #ddd;
}
```

이 때, child selector를 사용해 주면 첫번째 뎁스의 만 선택을 해서 스타일을 적용 할 수 있게 된다. IE6에서는 구현되어 있지 않다.

E + F (인접 선택자)

E와 F엘리먼트가 서로 근접해 있는 형제관계(sibling)일 경우를 선택한다. 제목 바로 아래의 문단에만 특정 스타일을 적용하는 경우와 같이 다른 엘리먼트가 디자인 적으로 영향을 미칠 때에 유용하게 사용할 수 있다. IE6에서는 구현되어 있지 않다.

```
<h2>브라우져 워</h2>
<p>웹스탠다드를 보다 잘 이해하기 위해서는 브라우져 워에 대해서 짚어볼 필요가
있습니다. 초창기에는 많은 브라우져들이 있었고 그중 사용자가 가장 많은 표준
브라우져는 넷스케이프 네비게이터(이하 NN)였습니다. 그리고 이 시장에
마이크로소프트(이하 MS)가 진입을 했습니다. 초기 windows(win95?) 사용자는
윈도우를 설치 하더라도 … 즉 많은 브라우져가 있었고 IE 에 의한 독점이 이루어지지
않고 서로 시장 점유율을 높이려고 노력하던 시대를 Browser War 라고 합니다.</p>
<h2>브라우져 독점의 폐단과 우리의 웹시장</h2>
<p>우리의 웹시장이 IT 에 대한 지원과 더불어 엄청난 속도로 발전을 했다는 것은
모든 사람이 아는 사실입니다. 그리고 그러한 발전이 양적인 발전이지 질적인 발전은
떨어진 다는 것도 대부분의 사람들이 인식하고 있는 사실입니다. …그리고 약간
비약하자면 이러한 현상에 브라우져의 독점 현상이 일조를 했습니다. 사람들은 MS 의
IE 에서 돌아가는 javascript 를 구현하고 MS 에서 제시하는 방법론을 아무 저항없이
받아 들였습니다.</p>
```

제목 바로 다음에 나오는 문단의 첫글자를 크게 나타내고자 할 경우 첫번째 글자를 등을 이용해서 따로 선택을 해 주어야 스타일을 적용 할 수 있다. 이럴 경우에 이 셀렉터를 사용하면 별도의 추가 마크업 없이 쉽게 구현 할 수 있다.

```
h2+p:first-letter {
   float: left;
   font-size: 2.2em;
}
```

별도의 코드 없이 타이틀 바로 다음의 "웹" 이라는 글자와 "우"자를 선택하여 스타일을 적용하면서 다른 문단에는 영향이 없는 것을 볼 수 있다.

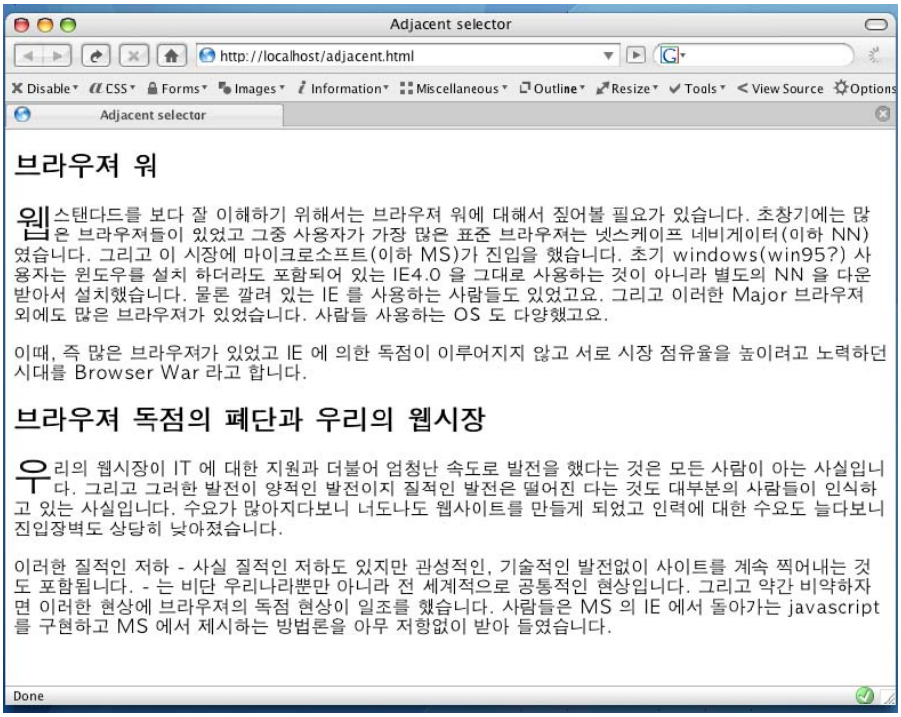
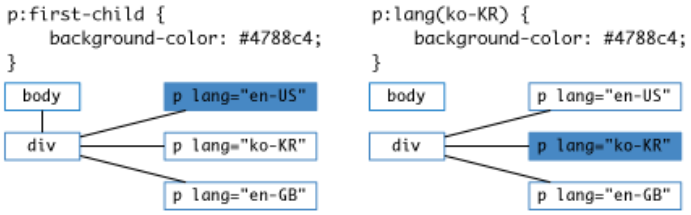


그림 13 인접 선택자 사용 예제

가상 클래스 선택자



:first-child 선택자	A:first-child	태그 A로 감싸져 있는 가장 처음 태그를 지정
언어 선택자	A:lang(B)	태그 A중 언어가 B로 설정된 것을 지정
링크 선택자	A:link	태그 A중 링크가 걸려있으면 지정
	A:visited	태그 A중 링크가 걸렸고 사용자가 이미 클릭한 태그를 지정
동적 선택자	A:active	태그 A중 사용자가 마우스를 누르고 있는 태그를 지정
	A:hover	태그 A중 사용자가 마우스 포인터를 위에 올려 놓은 태그를 지정
	A:focus	태그 A중 사용자의 키보드 입력을 받는 태그를 지정

그림 14. 가상 클래스 선택자 종류 (출처: http://andsite.net/)

E:first-child (:first-child 수도 클래스)

E 엘리먼트 중에서 맨처음에 나오는 E 엘리먼트를 선택한다. 리스트 등을 디자인 할 때

유용하게 사용할 수 있다. IE6에서는 구현되어 있지 않다.

E:lang(c) (언어 수도 클래스)

언어가 c 인 E엘리먼트를 선택한다. 문서 안에 한국어, 일본어, 중국어가 섞여 있고 이 글자들에 각각 다른 스타일을 적용해야 할때 유용하게 사용할 수 있는 셀렉터이다. IE6에서는 구현되어 있지 않다.

E:link, E:visited (링크 수도 클래스)

링크인 E 엘리먼트와 방문한 E 링크 엘리먼트를 선택한다. HTML 4.01 이나 XHTML 1.0 에서는 a 엘리먼트가 해당된다.

E:active, E:hover, E:focus (동적 수도 클래스)

사용자 액션이 active, hover, focus 인 E 엘리먼트들을 선택한다.

```
input:hover,
input:focus {
   background: #ffe;
}
```

와 같이 사용하면 input에 마우스 포인터가 오버 되거나 커서가 위치해 있을때의 배경색을 지정 할 수 있다. 아래를 보면 글을 작성하고 있는 폼의 배경색이 다른 것을 볼 수 있다.

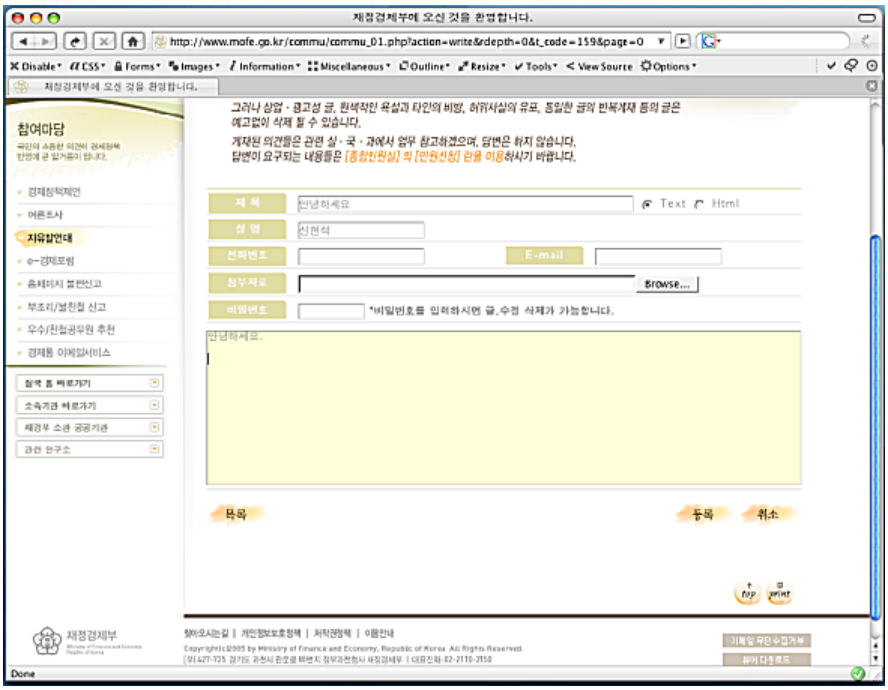


그림 15 동적 수도 클래스 사용 예제

IE6는 a 엘리먼트에서만 구현 되어 있고, :focus는 구현되어 있지 않다.

E[foo], E[foo="warning"], E[foo~="warning"], E[lang|="en"] (속성 선택자)

foo라는 속성과 warning이라는 값을 가진 E엘리먼트를 선택한다.

```
<input type="text" />
<textarea></textarea>
```

위와 같은 폼 컨트롤의 경우 사이트 전반적으로 일정한 보더를 주는 경우가 있는데 이럴 경우 아래와 같이 손쉽게 스타일을 적용 할 수 있다. IE6에서는 구현되어 있지 않다.

```
input[type="text"],
textarea {
    border: 1px solid #eee;
}
```

동적 선택자

CSS는 선언적인 특성을 가지고 있기 때문에 문서에 동적으로 스타일을 적용하기는 힘들지만 pseudo class, pseudo element라는 것을 제공함으로써 몇가지 경우에 있어서는 동적으로 스타일을 적용 할 수 있다. 첫번째 엘리먼트라든가 첫번째 글자 등을 따로 선택한다든가 마우스 액션에 따른 스타일을 적용한다든가 하는 것을 이를 이용해서 할 수 있다.

```
p:first-line {
    text-transform: uppercase
}
p:first-letter {
    font-size: 2em;
}
```

THE :FIRST-LETTER PSEUDO-element must select the first letter of the first line of a block,

```
p:before {
    text-content: "[";
}
p:after {
    text-content: "]";
}
<p>The :before, :after pseudo-element can used to insert</p>
```

[The :before, :after pseudo-element ca used to insert]

:first-line 선택자	A:first-line	태그 A의 문단중 첫번째 줄을 지정
:first-letter 선택자	A:first-letter	태그 A의 문단중 첫번째 글자를 지정
:before 선택자	A:before	태그 A의 문단 앞을 지정
:after 선택자	A:after	태그 A의 문단 뒤를 지정

그림 16 동적 선택자 종류 (출처: http://andsite.net/)

:link, :visited, :hover, :active, :focus 수도 클래스

:link, :visited 수도 클래스는 이미 대부분의 사람들이 사용하고 있는 수도 클래스 이다. 바로 링크가 되어 있는 엘리먼트와 방문한 링크를 선택할 수 있는 수도 클래스 인데, 수도 클래스는 이와 같이 순차적으로 적용되는 것이 아니고 특정 조건을 가진 엘리먼트를 선택해 올 수 있는 기능을 제공한다.

:hover와 :active 수도 클래스는 마우스의 사용과 함께 보다 다양한 스타일 적용을 할 수 있게 해 준다. MSIE에서는 이 수도클래스가 <a>에서만 작동을 해서 링크 스타일 외의 사용은 많지 않지만 대부분의 브라우저들은 이 수도클래스를 <a>이외의 엘리먼트에도 사용할 수 있다. 예를 들어서 <tr>에 마우스가 오버되었을때 컬러를 바꾸는 것을 :hover를 이

용해서 표현이 가능하다.

```
tr:hover td {
    background: #eee;
}
```

이와같이 하면 마우스가 올라갔을때 배경색이 바뀌는 기능을 javascript를 사용하지 않고 CSS만으로 구현할 수 있다.

이보다 더 유용한 것으로 :focus 수도 클래스가 있다. 이 수도 클래스는 엘리먼트에 포커스가 이동되어 왔을 때를 선택 할 수 있다. 이를 <input>이나 <select>등에 적용하면 사용자는 자신의 포커스가 현재 어디에 있는지 화면상에서 확실하게 알 수 있다. 아래의 회원가입 폼을 보면 현재의 포커스가 이동해 있는 성명 부분의 <input>의 배경 색이 다른 것들과 다른 것을 볼 수 있다.

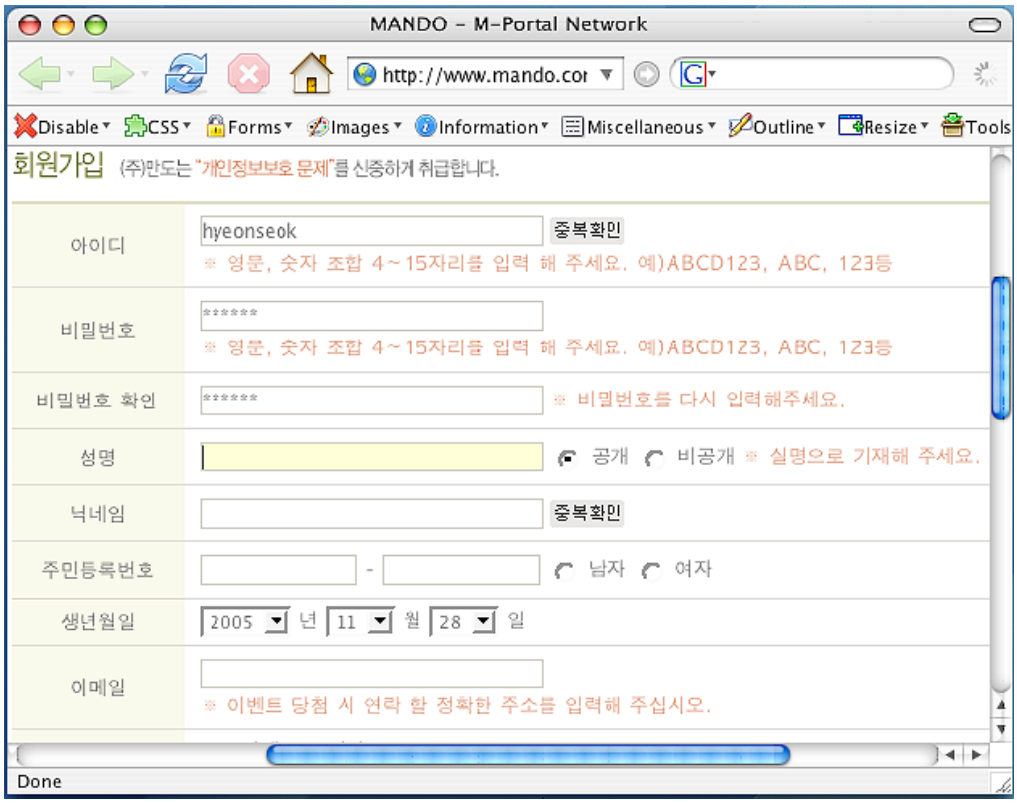


그림 17 focus 수도 클래스 사용 예제

:first-child 수도 클래스

:first-child 수도 클래스는 첫번째 엘리먼트를 선택해 옴으로써 디자인 적용에 동적인 기능을 제공한다.

위의 리스트 디자인을 보면 각 항목간에 구분선이 있는 것을 볼 수 있다. 엘리먼트에 border-top 속성을 이용해서 구현을 할 경우 처음 나오는 리스트 항목에도 위에 줄이 생겨서 원하는 결과를 얻기는 힘들다. 그래서 이와 같은 경우 첫번째 항목에만 특정 class를 적용한다든가 해서 해결을 하게 된다.

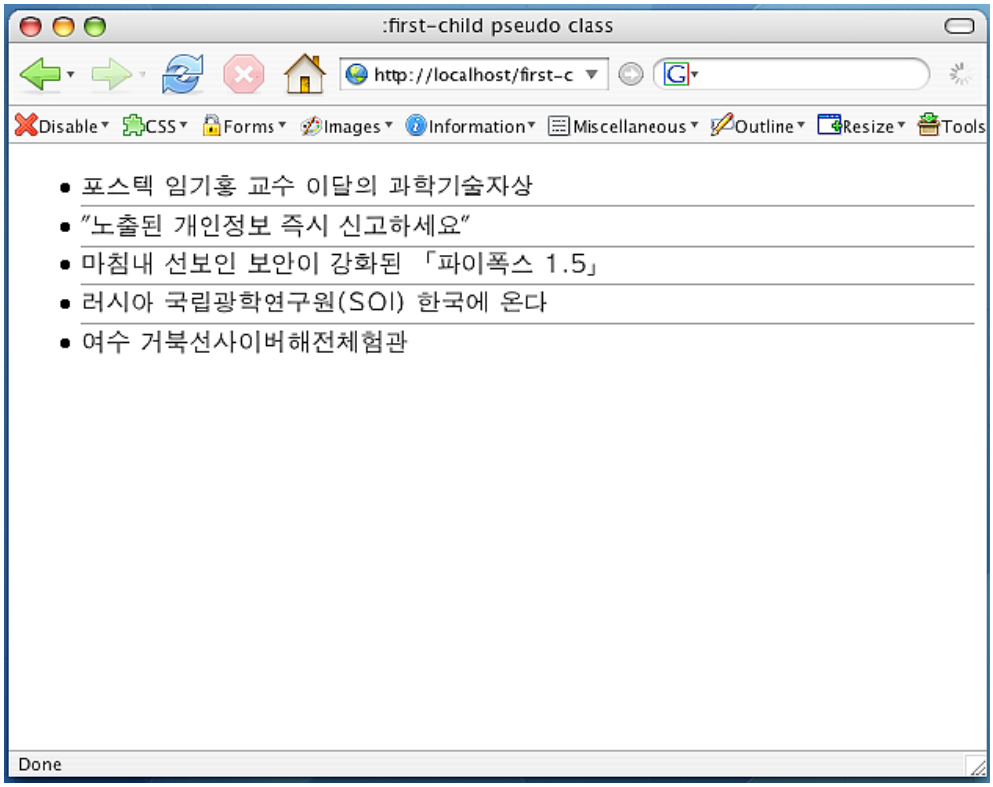


그림 18 :first-child 수도 클래스 사용 예제

하지만 이 리스트가 동적으로 생성되는 것이라면 첫번째 항목과 나머지 항목들을 구분하기 위해서 서버사이드나 클라이언트 사이드에서 별도의 작업을 해 주어야 한다. 이러한 추가 작업을 :first-child 수도 클래스를 사용하면 간단히 CSS만으로 구현할 수 있다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<title>:first-child pseudo class</title>
<style type="text/css">
li {
    border-top: 1px solid #999;
    padding: 0.2em 0;
}
li:first-child {
    border-top: 0 none;
}
</style>
</head>
<body>
<ul>
    <li>포스텍 임기홍 교수 이달의 과학기술자상</li>
    <li>"노출된 개인정보 즉시 신고하세요"</li>
    <li>마침내 선보인 보안이 강화된 「파이폭스 1.5」</li>
    <li>러시아 국립광학연구원(SOI) 한국에 온다</li>
    <li>여수 거북선사이버해전체험관</li>
</ul>
</body>
</html>
```

이와 같이 :first-child 수도 클래스를 사용하면 추가적인 로직 작업 없이 HTML과 CSS만으로 간편하게 원하는 디자인을 적용 할 수 있다. IE6는 아직 구현되어 있지 않다.

:first-line, :first-letter 수도 클래스

:first-line, :first-letter 수도 엘리먼트는 :first-child 수도 클래스와 비슷한 기능을 가지고 있다. :first-child 수도 클래스가 첫번째 나오는 하위 항목을 선택하는 것과 비슷하게 :first-line 수도 클래스는 첫번째 라인을, :first-letter 수도 클래스는 첫번째 글자를 선택할 수 있다. 차이가 있다면 수도 클래스의 경우는 기존에 생성되어 있는 엘리먼트를 선택해 오지만 수도 엘리먼트의 경우는 코드상에는 별도의 엘리먼트가 없지만 마치 다른 엘리먼트로 구별 되어 있는 것 같이 해당하는 엘리먼트를 생성해서 선택해 온다.

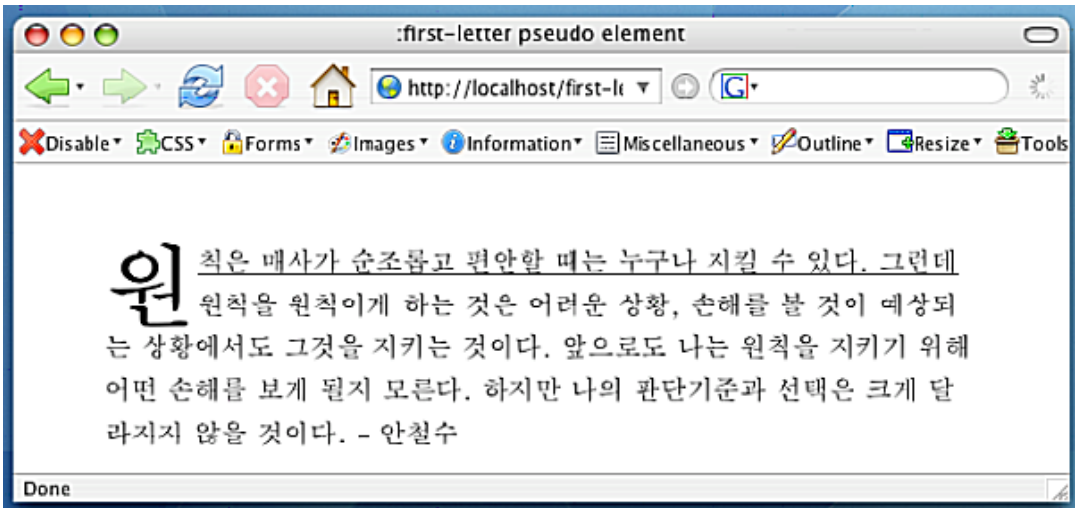


그림 19 first-line, first-letter 수도 클래스 사용 예제

문단의 첫번째 글자인 "원"자의 클씨 크기를 크게 하고 문단의 첫번째 줄에 밑줄이 그어져 있다. 이러한 디자인을 적용하기 위해서는 아래와 같은 코드를 작성해서 별도의 클래스를 적용해 주어야 한다.

```
<p><span class="first-line"><span class="first-child">원</span>칙은
매사가 순조롭고 편안할 때는 누구나 지킬 수 있다. 그런데</span> 원칙을 원칙이게
하는 것은 어려운 상황, 손해를 볼 것이 예상되는 상황에서도 그것을 지키는 것이다.
앞으로도 나는 원칙을 지키기 위해 어떤 손해를 보게 될지 모른다. 하지만 나의
판단기준과 선택은 크게 달라지지 않을 것이다. - 안철수</p>
```

그러나, :first-line 수도 엘리먼트와 :first-letter 수도 엘리먼트를 사용하면 위와 같이 엘리먼트를 삽입한 것과 같은 효과를 CSS만으로 구현 할 수 있다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<title>:first-letter pseudo element</title>
<style type="text/css">
p {
    margin: 3em;
    line-height: 1.6em;
    font-family: AppleMyungJo, serif;
}
p:first-line {
    text-decoration: underline;
}
p:first-letter {
    float: left;
    font-weight: bold;
```

```
    font-size: 3.5em;
}
</style>
</head>
<body>
<p>원칙은 매사가 순조롭고 편안할 때는 누구나 지킬 수 있다. 그런데 원칙을 원칙이게
하는 것은 어려운 상황, 손해를 볼 것이 예상되는 상황에서도 그것을 지키는 것이다.
앞으로도 나는 원칙을 지키기 위해 어떤 손해를 보게 될지 모른다. 하지만 나의
판단기준과 선택은 크게 달라지지 않을 것이다. - 안철수</p>
</body>
</html>
```

CSS 선언 방법

CSS는 우선순위에 따라서 3가지의 선언 방법이 있다.

외부 선언 (external css)

```
<link rel="stylesheet" type="text/css" href="myCss.css" />
```

HTML <head> 엘리먼트에 위와 같이 선언하여 외부에 별도의 파일로 되어 있는 CSS 정의를 불러온다. 여러 HTML 파일이 하나의 CSS 파일을 공유할 수 있어서 표현에 일관성을 갖게 해 준다. 우선 순위는 가장 낮다.

문서 안에 포함 (embeded css)

```
<head>
<style type="text/css">
body {
    margin: 0;
    padding: 0;
}
</style>
</head>
```

위와 같이 HTML <head> 엘리먼트 안에 <style> 엘리먼트를 사용하여 하나의 문서 안에서 CSS를 정의 한다.

엘리먼트에 직접 선언 (inline css)

```
<div style="padding: 10px; border: 1px solid #eee;">
    <p>contents</p>
</div>
```

HTML 엘리먼트에 style 속성을 이용하여 직접 선언하는 방법이다.

사용자 정의 스타일 (user defined css)

가장 우선 순위가 높은 선언으로 웹페이지 제작자가 선언하는 것이 아니라 웹사이트를 이용하는 사용자가 직접 자신에게 맞는 스타일을 선언하는 방법이다.

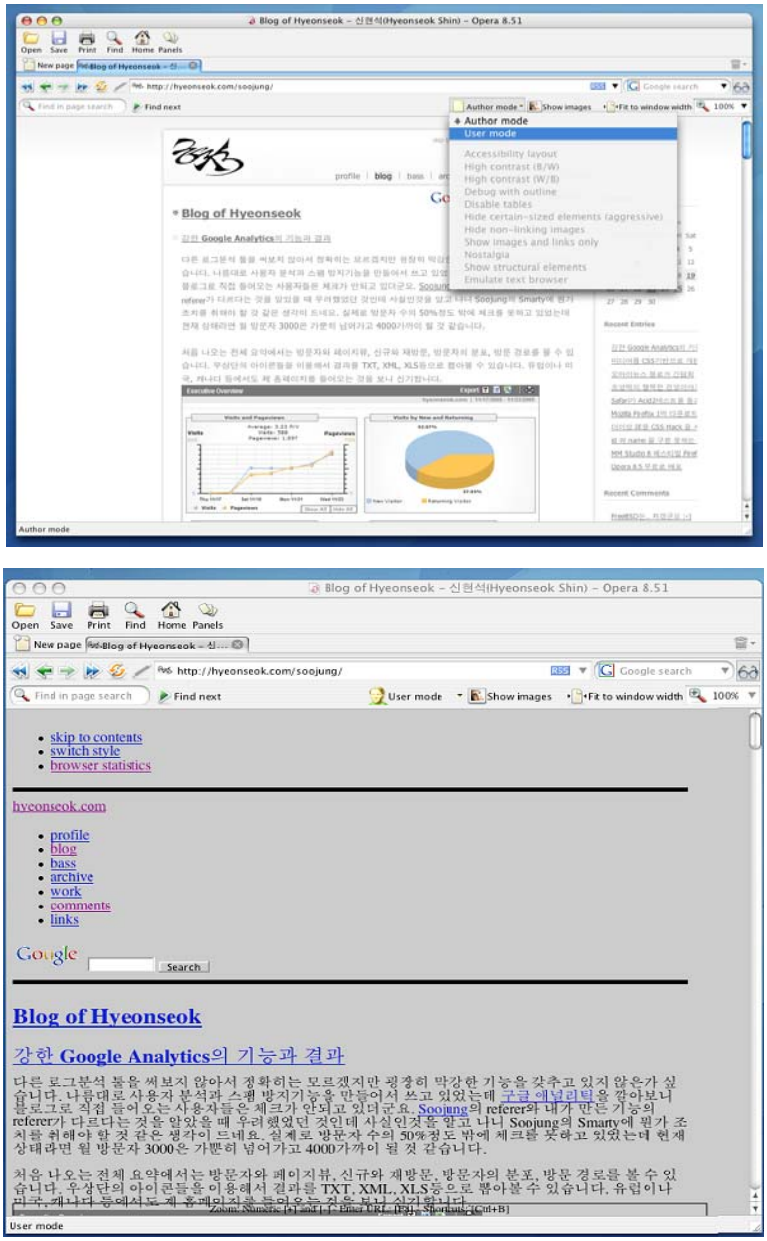


그림 20 사용자가 스타일을 선택 가능하도록한 표준 기반 예제

Opera의 경우 스타일을 author 모드와 user 모드를 선택하여 적용하는 기능을 제공한다. 좌측의 원래 사이트가 사용자의 스타일 재정의에 의해서 완전히 다른 모습으로 보이는 것을 볼 수 있다. 사용자의 정의가 가장 우선 순위가 높다는 것은 문서의 구조와 디자인을 분리 하여서 개별 사용자가 스스로에게 가장 적합한 형태의 스타일을 선택하여 쉽게 문서에 접근할 수 있게 하기 위함이다.

CSS 적용의 체크 포인트 4가지

CSS를 이용해서 페이지에 디자인을 적용하기 전에 반드시 지켜야 하는 사항 4가지

XHTML이 표준 문법이어야 한다.

CSS를 적용하기 전에 반드시 XHTML이 표준 문법인지를 검사해야 한다. XHTML 문법이 표준이 아니면 제작 시에는 랜더링이 정상적으로 된다고 하더라도 테스트하고 있는 브라우저가 아닌 다른 브라우저에서는 랜더링이 같게 나올 것이라고 보장 할 수 없게 된다. XHTML 문법을 확인하는 가장 잘 알려진 방법은 W3C의 Markup Validation Service(http://validator.w3.org)를 이용하는 것이다.

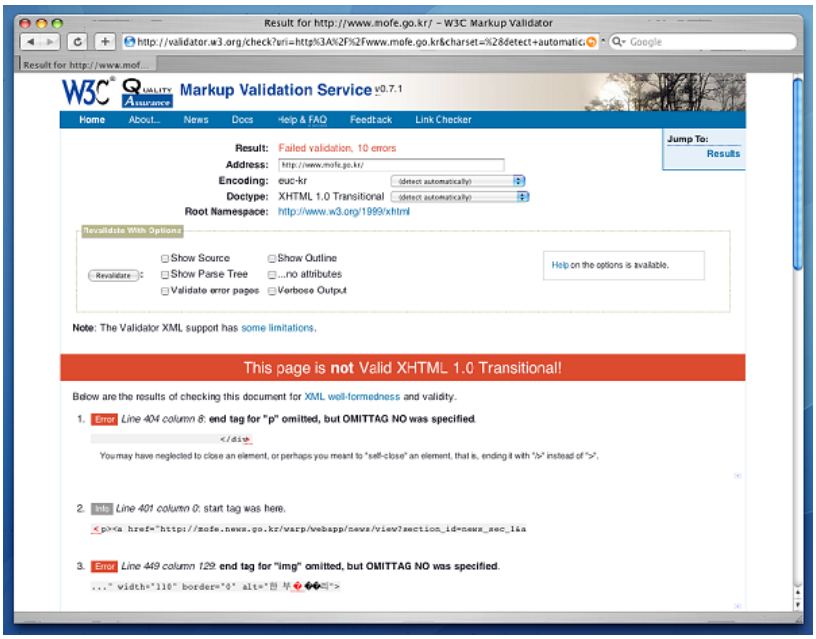


그림 21 W3C CSS Validator

HTML, XHTML, XML등의 markup문서의 문법을 체크하는 툴로서 markup 문서의 DTD 선언에 기초하여 문법을 검사해 준다. 현재 버전은 0.7.1로 URL을 직접 입력하는 방법, 파일을 업로드 하는 방법, textarea에 코드를 직접 넣는 방법, 세가지로 문법검사를 할 수 있다.

XHTML 문서가 의미와 구조적으로 구성되어야 한다.

XHTML 문서를 작성할 때에는 표현하고자 하는 컨텐츠에 맞게 태그를 사용하여 문서를 표현해야 한다. 국내의 많은 사이트들은 이러한 컨텐츠의 의미를 나타내어 주는 태그를 사용하기 보다는 단순히 화면상에서 어떻게 표현되는 지만을 고려하여 제작되어 있는 경우가 많다.

예를 들어서 문서의 전체적인 내용을 나타내고자 하면 <title>을 사용하게 된다. 많은 사이트들이 이 <title>에 동일한 내용을 넣고 있는데 이는 잘못된 태그의 사용이고 <title>에는 해당 페이지의 내용을 간략하게 넣어주어야 한다. 또한 문서의 내용들 중에서 각 제목에 해당하는 내용은 각 단계에 맞게 <h1>~<h6>를 이용해서 표현을 해 주어야 한다. 이 <h1>~<h6>들은 사용자가 문서의 구조를 파악하는데 많은 도움을 준다. 이 외에도 <ol>, <ul>, <p>, <blockquote>, <ins>, <del> 등과 같은 태그들을 이용해서 컨텐츠가 의미하는 바를 표현해 주어야 한다. 이렇게 의미에 맞는 태그를 적절히 사용해 주게 되면 문서의 스타일을 적용하지 않고 브라우저 기본스타일 만으로도 충분히 가독성 높은 문서

를 만들 수 있다.

이렇게 의미에 맞는 태그로 문서가 제작이 되면 다음으로는 문서중에서 의미에 따라서 나눌 수 있는 부분들을 그 의미에 맞게 구분을 해 주어야 한다. 통상 이러한 구분은 , 태그와 id, class를 부여하는 것으로 구현 되고 이 구분들을 CSS Selector 를 사용해서 디자인을 적용하게된다.

여기에 추가적으로 <script>, <style>과 같은 태그들은 가능하다면 외부문서로 분리하여 HTML크기를 줄이는 것이 좋다. 문서를 제작 할때에 CSS를 적용하는 것보다는 markup을 완성도 있게 만드는 것이 무엇보다도 중요하다는 것을 항상 염두해야 한다.

CSS가 표준 문법이어야 한다.

XTHML이 표준 문법이어야 하는 것과 마찬가지로 CSS도 표준 문법을 사용해야 한다. 아래는 자주 볼 수 있는 잘못 사용된 CSS 문법들이다.

주석구문

CSS의 주석 구문은 아래와 같다.

```
/* comment */
```

HTML 주석과 다른 언어들의 한줄 주석은 CSS의 주석 문법이 아니니 사용해서는 안된다.

```
<!-- wrong CSS comment -->
```

한줄 주석

```
// wrong CSS comment
```

단위표기

CSS의 값들 중에 0을 제외한 모든 값들은 단위를 표기 하여야 한다.

```
<td style="padding: 15 20 10 30;">
```

많이 볼 수 있는 실수가 위와 같이 단위를 안 적는 것이다. MSIE는 단위가 없으면 pixel로 해석을 해주어서 많은 사람들이 이와 같은 오류 구문을 사용 하는데 반드시 단위를 기입해 주고 세미콜론(;)으로 구문의 끝을 표시해 주어야 한다.

```
<td style="padding: 15px 20px 10px 30px;">
```

잘못된 컬러값 사용

CSS에서 컬러 값을 입력 할때에는 3가지 방법을 사용할 수 있다.

컬러의 이름을 넣는 방법 - color: red;

```
#rrggbb 방법 - color: #ff0000; (=color: #f00;)
rgb(r,g,b) 방법 - color: rgb(255,0,0); (=color: rgb(100%,0%,0%);)
```

간혹 color: ff0000; 이나 color=#ff0000 과 같이 표현하는 경우를 볼 수 있는데 잘못된 구문이므로 주의해야 한다.

CSS를 표준대로 잘 구현한 브라우저를 이용해야 한다.

XHTML과 CSS를 표준 문법으로 작성해도 IE4나 NN4와 같이 CSS가 제대로 구현되지 않은 브라우저를 사용하게 되면 표준대로 랜더링 되지 않는 것이 당연한 일이다. CSS를 사용하여 사이트를 제작할 때에는 표준을 잘 준수하는 브라우저를 이용하고 제작이 끝난 후에 가장 많이 사용되는 메이저 브라우저에 대한 튜닝을 진행하는 것이 효과적이다. CSS 지원이 미약한 브라우저에서 제작을 할 경우 이미 표준대로 랜더링이 되고 있지 않기 때문에 모든 브라우저에대한 튜닝 작업을 해야 하는 상황이 발생하게 되고 이는 불필요한 자원의 낭비일 수 밖에 없다.

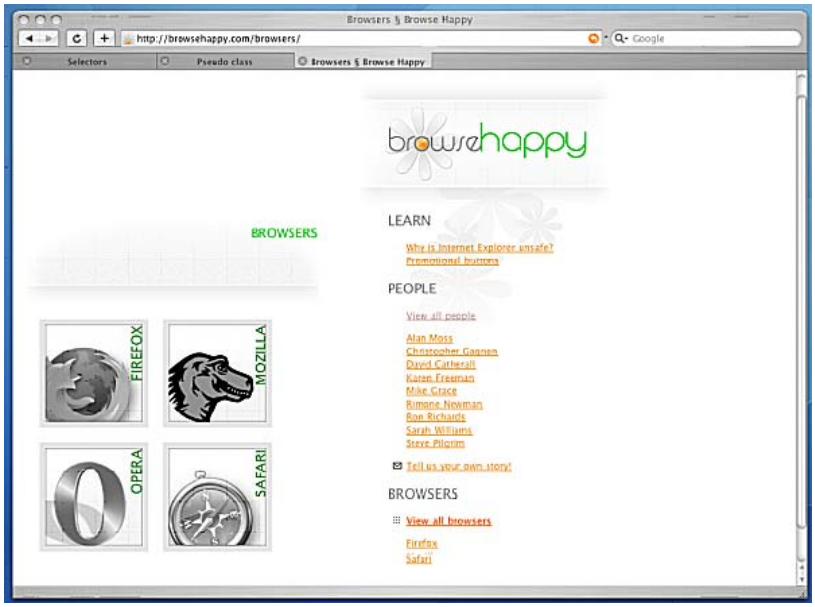


그림 22 일반인들에게 표준 웹 브라우저를 홍보하는 BrowseHappy

Web Standards Project (http://webstandards.org)의 캠페인 사이트인 Browse Happy (http://browsehappy.com)에서는 Firefox, Mozilla Suite, Opera, Safari 4개의 브라우저를 표준준수 브라우저로 일반인들에게 추천하고 있다.

CSS 레이아웃(Layout) 기초

CSS를 이용하여 사이트를 제작 할 때에 가장 먼저 접하게 되는 것이 CSS를 이용한 레이아웃의 제작이다.

테이블 레이아웃

웹 표준이라고 불리는 XHTML은 HTML 4.01에 기반을 두고 있다. 웹 표준을 담당하는 W3C의 문서를 살펴보면 테이블에 대한 사양을 알 수 있다.

> HTML 테이블 모델은 데이타를 행과 열의 셀로 정렬하는 것이 목적이다. 이 데이터에는 텍스트, 스타일이 있는 텍스트, 이미지, 링크, 폼, 폼 필드, 테이블 등이 있다. Tables in HTML docuemnts

테이블에 이미지나 또다른 테이블이 들어가도 표준에 어긋나지 않는다. 하지만 테이블을 레이아웃의 용도로 쓰지 말 것을 권장하고 있다.

> 테이블은 비 시각적 미디어에서 랜더링시 문제가 있기 때문에 문서 내용의 레이아웃을 정하는 목적으로 사용하지 않는 편이 좋다. 그리고 테이블에 이미지를 사용한다면, 디자이너의 화면이 사용자의 화면보다 넓을 경우 사용자가 수평으로 스크롤을 해야하는 문제가 발생할 수도 있다. 이런 문제들을 최소화하기 위해서 레이아웃은 테이블보다 스타일쉬트를 사용하는 것이 바람직하다. Tables in HTML docuemnts

인터넷 익스플로러 전용이라고 당당하게 홈페이지에 박아 놓는 우리나라 실정과는 동떨어진 이야기인게 문제다.

일반적으로 CSS만을 이용해 글이나 그림을 수평으로 정렬하는 것은 쉽지 않다 1. 게다가 75.5%의 점유율을 차지하고 있는 마이크로소프트의 인터넷 익스플로러(IE)는 2 CSS 버그들로 악평이 높고 언제쯤 이 버그들이 고쳐질지도 알 수가 없다. 그럼에도 불구하고 CSS를 써야하는 몇가지 이유가 있다.

예전의 테이블을 이용한 레이아웃 디자인은 테이블을 겹쳐서 원하는 위치에 글과 그림을 놓는 방법이었다. 글이나 그림의 간격은 투명 gif 파일로 조정하고, 테이블 속에 테이블이 서너개씩 들어간 것을 쉽게 볼 수 있었다.

이런 이유로 전체적인 데이타량이 늘어나고 구조가 복잡해져서 전송 시간과 페이지 랜더링 시간이 길어지며 보수와 유지도 힘들어진다. CSS를 이용하면 데이타량과 구조의 문제에서 해방될 수 있다. 게다가 CSS가 웹표준으로 받아들여지기 때문에 브라우저들이 표준을 지원할수록 페이지의 랜더링은 최적화되기 마련이다.

CSS 레이아웃이란?

그리드(grid)가 아닌 구성요소의 집합

보통의 테이블을 이용한 퍼블리시에 익숙한 사람이라면 웹페이지의 디자인을 보고 화면을 그리드로 나누는 것을 처음에 시도할 것이다. 하지만 이러한 접근은 웹페이지를 디자인 결과물로만 바라보는 시각이 강하고 웹의 본연의 목적인 정보 전달의 측면을 간과한 것이다.

웹페이지를 제작할 때에 가장 먼저 생각해야 하는 것은 화면의 분할이 아니라 웹페이지의 구성요소들과 이 구성요소들의 관계를 정립하는 것이다. 이렇게 구성요소와 구성요소들의 그룹을 확실하게 이해 해야 의미에 맞는 마크업을 이용하여 웹페이지를 퍼블리시 할 수 있게 된다.

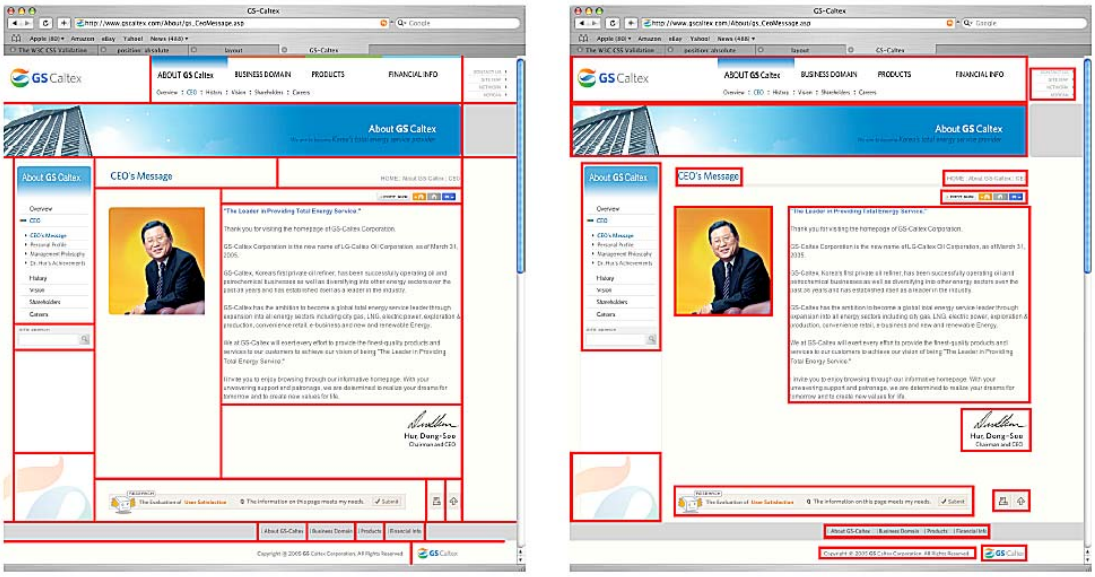


그림 23웹페이지를 그리드로 바라보고 접근한 것(좌측)과 구성요소로 구분하여 접근한 측면(우측)

좌측의 경우에는 구성요소들이 서로간의 레이아웃 종속성이 높은 것을 볼 수 있다. 디자인과 구성요소를 구분하여 생각하기 쉽고 이를 이용하여 보다 접근성 높고 디자인 변경이 쉬운 웹페이지를 제작 할 수 있다.

레이아웃에 사용되는 두가지 속성 position vs. float

CSS의 여러 속성 중에 position 속성과 float 속성을 이용하여 레이아웃을 제작할 수 있다. position 속성은 단어의 뜻이 말해 주듯이 위치를 지정하여 원하는 위치에 엘리먼트를 배치하는 속성이다. float 속성은 대상 엘리먼트를 띄워서 현재의 위치에서 좌측이나 우측에 배치하는 속성이다. float 속성은 레이아웃을 위한 속성이라기 보다는 주로 텍스트 안에 이미지를 삽입할 때에 사용되지만 다른 속성들과 함께 레이아웃을 작성하는 곳에 요긴하게 사용할 수 있다.

position

position은 static, relative, absolute의 세가지 값을 가질 수 있다.

- static : 기본값.
- relative : static과 같지만 offset을 지정 할 수 있고 하위 엘리먼트 offset의 기준점이 된다.
- absolute : 화면상에서 다른 컨텐츠에 위치에 영향을 미치지 않고 위치 지정이 가능하다. 보통 레이어라고 불리우는 것.

float

float은 left, right, none의 세가지 값을 가질 수 있다.

- left : 엘리먼트를 좌측으로 배치함.
- right : 엘리먼트를 우측으로 배치함.
- none : float시키지 않음

레이아웃을 작성 할 때에 position을 이용할 것인지, float를 이용할 것인지 선택하는 문제는 명확한 정답이 있는 것은 아니다. 각 방법들이 장단점을 가지고 있기 때문에 주어진 환경에 맞게 적절한 속성을 선택할 수 있어야 한다. position은 블록의 크기가 크게 유동적이지 않고 코드 상에서의 위치에 구애 받지 않고 블록을 위치 시킬때 사용한다. position을 사용하면 코드상에서 제일 하단에 있는 블록을 페이지의 상단으로 이동 시킬 수 있을 정도로 자유로운 블록의 배치가 가능하다. 이에 반해 float은 지정된 위치에서 좌측 또는 우측으로만 배치가 가능하기 때문에 position과 같이 자유로운 블록 배치는 힘들다. 하지만 float된 블록의 높이가 유동적으로 변경되어도 레이아웃 조정이 손쉽기 때문에 보통 컬럼을 사용해야 하는 레이아웃에 사용된다.

기본 레이아웃

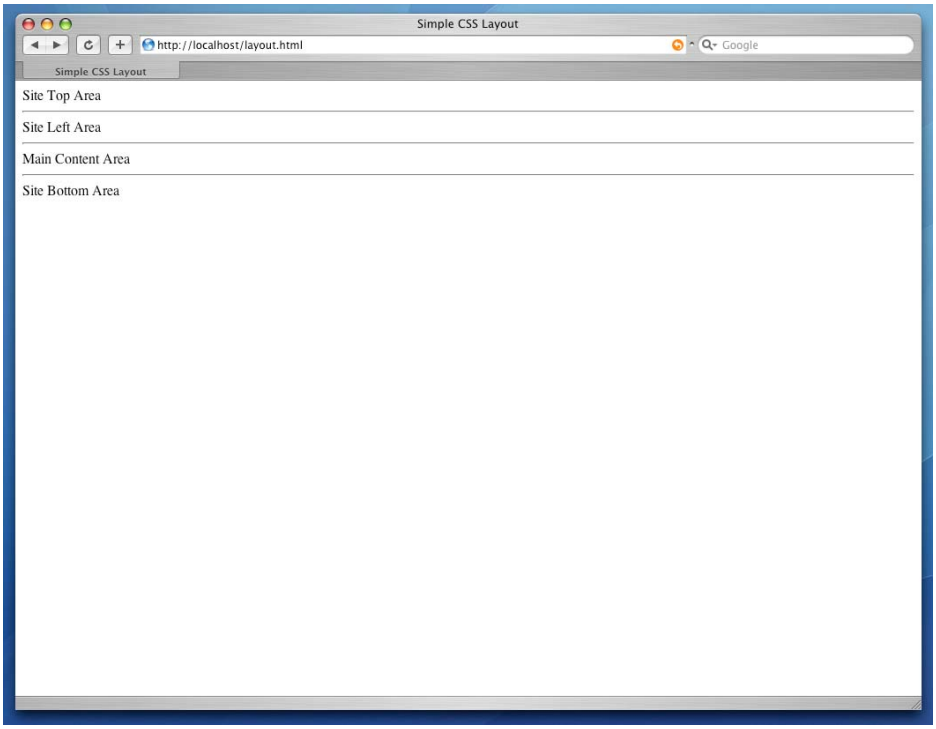
국내 웹사이트에서 일반적으로 많이 볼 수 있는 간단한 레이아웃을 position 속성을 이용해서 제작 한다.

마크업(Markup)

페이지 제작을 위해 먼저 markup을 작성한다. 이 markup작성 단계에서는 디자인이나 CSS를 고려하지 않고 제작을 한다. 페이지를 구성하는 구성요소가 무엇이고 각 구성요소들의 그룹에 따라서 의미에 맞는 태그와 id, class를 적용하여 markup을 구성한다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<title>Simple CSS Layout</title>
</head>

<body>
<div id="head">Site Top Area</div>
<hr>
<div id="sub">Site Left Area</div>
<hr>
<div id="body">Main Content Area
</div>
<hr>
<div id="foot">Site Bottom Area</div>
</body>
</html>
```



레이아웃 구성을 보기 위해서 상단(#head), 서브(#sub), 내용(#body), 하단(#foot) 만으로 이루어진 페이지 이다. 제작된 markup을 보면 디자인적인 markup이 전혀 포함되어 있지 않고 단지 영역을 구분지어 주는 div들과 id로만 이루어 진 것을 알 수 있다. <table>을 이용하여 레이아웃을 작성하게 되면 markup에서 영역이 좌측에 위치 하게 되는지 우측에 위치하게 되는지 이미 결정지어지게 된다. 하지만 CSS를 이용한 레이아웃에서는 markup에는 이러한 디자인 적인 정보가 없어서 CSS만을 이용하여 레이아웃을 쉽게 바꿀 수 있게 된다. 제작된 결과를 브라우져에서 보면 오른쪽 같이 단순히 위에서 아래로 나열되는 모습을 볼 수 있다.

CSS

markup이 의미에 맞게 작성이 되었으면 CSS를 이용하여 디자인을 적용한다. CSS를 이용하여 디자인을 적용 할때에 가장 먼저 고려 해야 하는 것은 브라우져 고유의 스타일이다. 모든 브라우져들은 태그마다 고유의 스타일을 가지고 있다. 예를 들어서 h1은 페이지에서 가장 중요한 타이틀이기 때문에 가장 큰 글씨를 보여준다든가 ul과 li를 사용하면 li마다 앞에 불렛(bullet)을 표시한다든가 하는 것이다. 레이아웃을 작성할 때에는 이러한 기본 스타일중에서 body의 기본 스타일을 고려해 주어야 한다. 스타일 없이 HTML문서를 제작할 경우 화면의 가장자리에 여백이 있는 것을 볼 수 있다. 이 body의 여백을 아래과 같이 처리한 경우를 쉽게 볼 수 있다.

```
<body topmragin="0" leftmargin="0" marginheigth="0" marginwidth="0">
```

하지만 이 속성들은 HTML 4.01 이상의 버전에서는 사라진 속성이기 때문에 이렇게 사용해서는 안되고 CSS를 이용해서 처리해 주어야 한다. CSS로는 아래와 같이 하여 화면가장자리의 여백을 없앨 수 있다.

```
body {
    margin: 0;
```

```
    padding: 0;
}
```

보통은 margin만 사용하면 여백이 없어지지만 여백의 조절을 padding으로 하는 브라우저가 있기 때문에 두개 다 적용시키는 것이 좋다.

다음으로 각 부분을 구분하는 <hr>은 디자인을 적용하면 화면상에서는 없어져야 하기 때문에 안보이게 처리를 한다. <hr>같은 엘리먼트는 텍스트 환경에서 각 부분을 구분지어주는 역할을 하기 때문에 삭제하지는 말고 코드에는 적용을 하고 스타일로 보이지 않게 해주는 것이 좋다.

```
hr {
    display: none;
}
```

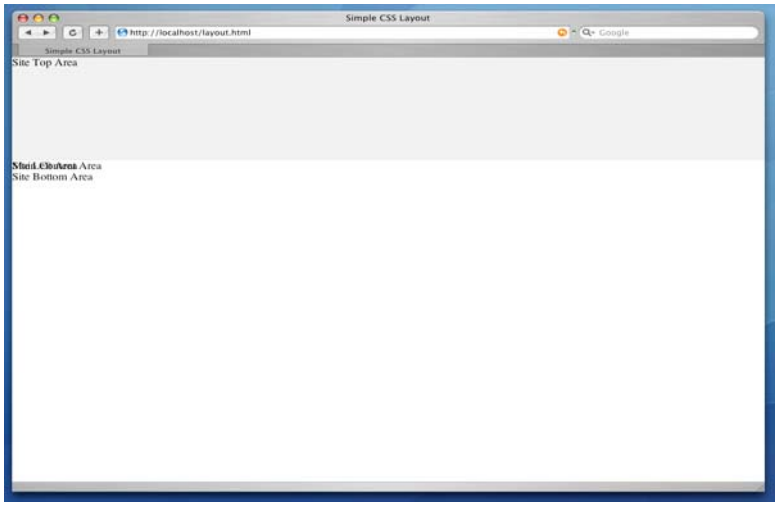
Site Top Area(#head)는 사이트 로고나 상단 메뉴, 비주얼이 들어가는 영역인데 고정된 높이를 갖는 경우가 많다. #head에는 높이만을 적용하고 화면에서 구분을 할 수 있게 바탕에 색만 지정을 해 준다.

```
#head {
    height: 170px;
    background: #eee;
}
```

Site Left Area(#sub)는 사이트의 좌측 영역으로 서브네비게이션이나 섹션의 타이틀, 배너, 검색 등이 들어가게 된다. 이 #sub는 비교적으로 높이가 고정적이고 컨텐츠 영역의 좌측에 항상 위치하기 때문에 고정적으로 absolute position을 사용하여 위치 시킨다.

```
#sub {
    position: absolute;
    top: 170px;
    left: 0;
    width: 160px;
}
```

여기까지를 적용해 보면 오른쪽과 같다.



#sub에 absolute position을 사용했기 때문에 Site Left Area와 Main Content Area가 겹쳐진 것을 볼 수 있다. absolute position의 경우 화면에서 위치를 차지하지 않기 때문에 Site Top Area 바로 아래에 Main Content Area가 위치하게 된 것이고 Site Left Area가 그 위에 겹쳐지게 된 것이다. 보통 레이어라고 일컬어 지는 것이 바로 이

absolute position을 이용한 block의 다른 이름이다.

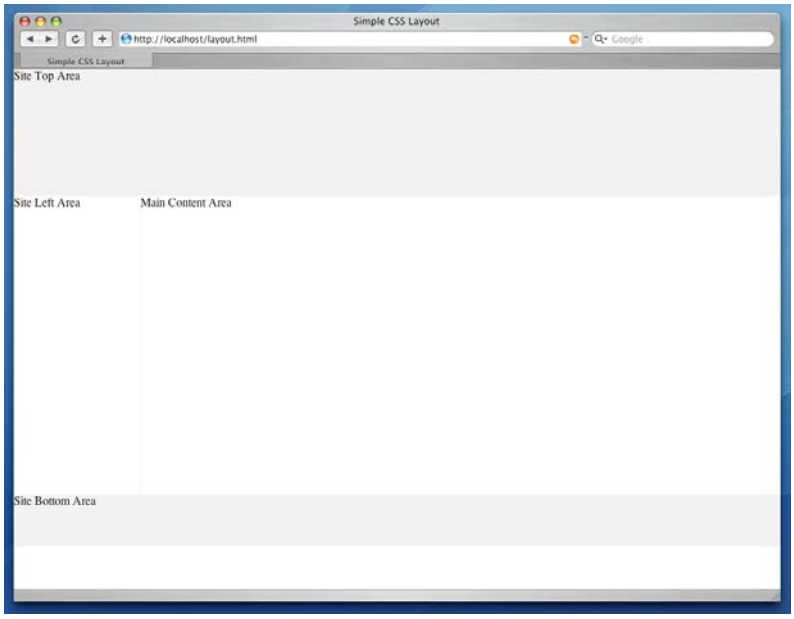
Main Content Area는 padding을 써서 간단하게 Site Left Area와 분리할 수 있다. 여기서 margin을 사용하지 않고 padding을 사용한 이유는 padding을 사용하여 Site Left Area까지 포함하여 #body에 background 속성을 사용할 수 있게 하기 위함이다.

```
#body {
    padding-left: 170px;
    width: 700px;
}
```

보통의 이러한 2단 레이아웃의 경우 좌측 영역과 컨텐츠 영역 사이에 구분선과 같은 디자인이 들어가게 되는 Site Left Area는 absolute position을 사용했기 때문에 높이가 유동적이지 못하고 구분선을 사용할 수가 없다. 그래서 구분선을 background-image를 사용하여 #body에 적용해 주게 된다. 또한, Site Left Area에 absolute position을 사용했기 때문에 Main Content Area의 높이가 Site Left Area의 높이보다 작을 경우 Site Left Area가 Site Bottom Area를 덮어버릴 수가 있다. 그래서 Main Content Area가 항상 Site Left Area보다 높게 지정해 준다.

```
#body {
    padding-left: 170px;
    width: 700px;
    background: url(body.gif) repeat-y 170px 0;
    min-height: 400px;
}
```

마지막으로 Site Bottom Area에 높이와 색을 지정해주면 간단한 레이아웃이 완성된 모습을 볼 수 있다.



relative와 absolute의 관계

대부분의 사이트가 이 position 속성을 absolute만 사용을 해서 단지 레이어 개념으로만 사용하고 있다. 이 absolute position은 relative position과 같이 사용하면 보다 다양하고 자유로운 위치 조정이 가능해 진다. absolute position을 사용했을때, top, left, right, bottom의 속성으로 offset을 지정하게 된다. 보통의 경우는 이 absolute position 블록의

상위에 아무것도 없기 때문에 이 offset이 브라우저의 좌상단을 기준으로 지정이 된다. 하지만 absolute position 블록의 상위에 relative position 블록이 있으면 이 offset의 기준이 상위의 relative position 블록의 좌상단이 된다.

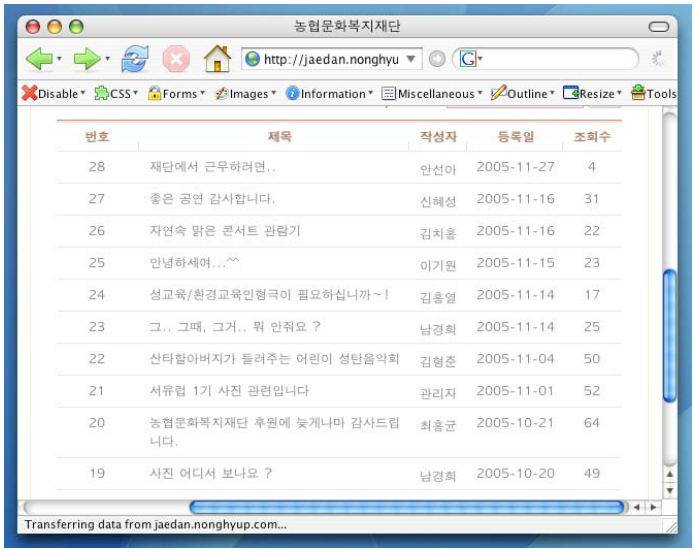
```
<div id="board-list" class="freeboard-item">
    <ul id="board-list-item">
          <li>
                <div class="number">
                      26
                      <!-- ArticleSeq: 271 -->
                </div>
                <div class="title">
                <a href="view.jsp?articleSeq=271">자연속 </a>
                </div>
                <div class="name">
                      김치홍
                </div>
                <div class="date">
                      2005-11-16
                </div>
                <div class="hits">
                      22
                </div>
          </li>
          ...
          ...
          ...
    </ul>
</div>
```

각 에는 div.number, div.title, div.name, div.date, div.hits 블록이 있다.

```
#board-list-item li {
    position: relative;
    width: 548px;
    border-bottom: 1px solid #EBDDD4;
    color: inheret;
}
#board-list li div.number,
#board-list li div.date,
#board-list li div.hits {
    top: 7px;
}
#board-list li div.title {
    padding-top: 7px;
    padding-bottom: 5px;
}
#board-list div.number {
    position: absolute;
    left: 0;
    width: 79px;
    text-align: center;
}
#board-list div.title {
    margin-left: 90px;
    width: 297px;
}
#board-list div.date {
    position: absolute;
    right: 57px;
    width: 92px;
    text-align: center;
}
#board-list div.hits {
    position: absolute;
    right: 0;
    width: 57px;
    text-align: center;
}
div.freeboard-item div.name {
```

```
    position: absolute;
    top: 10px;
    right: 150px;
    width: 55px;
    height: 1.5em;
    text-align: center;
    overflow: hidden;
}
div.freeboard-item div.title {
    width: 252px !important;
}
```

는 relative position이고 하위의 div들은 div.title을 제외하고는 absolute position 이다. 그러면 이 하위의 div들은 브라우저의 좌상단을 기준으로 top과 left같은 offset이 지정되는 것이 아니라 의 좌상단을 기준으로 위치가 지정되게 된다. 따라서 좌상단을 기준으로 정렬 되는 것 처럼 서로 겹치게 되는 것이 아니라 안에서 위치가 결정 되므로 이와 같은 들을 이용해서 게시판의 리스트 화면을 만들 수도 있다. 단, 이 예에서 div.title은 static position을 사용해서 의 기본적은 크기를 유지해 주어야 하고, 다른 div들은 div.title보다 높이가 높아져서는 안된다.



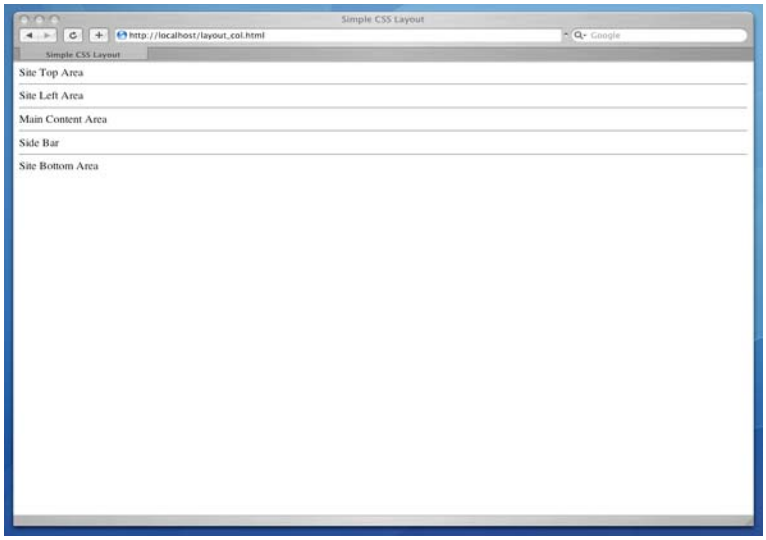
컬럼형 레이아웃

블로그와 같은 사이트에 볼 수 있는 컬럼형 레이아웃 제작에는 float 속성이 적합하다.

Markup

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<title>Simple CSS Layout</title>
</head>
<body>
<div id="wrapper">
    <div id="head">Site Top Area</div>
    <hr>
    <div id="sub">Site Left Area</div>
    <hr>
    <div id="body">Main Content Area</div>
    <hr>
    <div id="sidebar">Side Bar</div>
```

```
    <hr>
    <div id="foot">Site Bottom Area</div>
</div>
</body>
</html>
```



float 속성을 이용해서 레이아웃을 작성 할때에는 컬럼전체의 폭을 고정하기 위해서 바깥쪽에 감싸는 블록 하나를 두어야 한다. 여기서는 #wrapper로 전체 사이트를 감싸 주었다. 겉으로 보기에는 컬럼을 차지하는 Side Bar만 추가되었다. 랜더링된 화면은 이전의 예와 크게 다를 것 없이 수직적으로 해당 섹션들이 나열된 형태인 것을 볼 수 있다.

CSS

기본적인 CSS 설정은 이전에 다루었던 부분과 같다. body의 margin을 없애주고, hr을 보이지 않게 한다. 그리고 사이트 전체의 폭과 중앙 정렬을 위해서 #wrapper와 body에 스타일을 적용한다.

```
#wrapper {
    width: 700px;
    border: 1px solid #eee;
    margin: 20px auto;
}
```

이렇게 하면 화면의 중앙에 정렬된 모습을 볼 수 있다. 추가적으로 Site Top Area와 Site Bottom Area를 구분하기 위해서 배경을 넣어준다.

```
#head {
    height: 80px;
    background: #eee;
}
#foot {
    height: 30px;
    background: #eee;
}
```

float 속성을 이용해서 블록을 배치하는 방법은 아주 간단하다. 단지 폭을 정해주고 좌측이나 우측에 정렬을 해주면 하나의 블록이 하나의 컬럼을 형성하면서 레이아웃을 작성 할 수 있다. 그리고 #foot에서는 float된 블록들을 clear시켜 주어서 #foot블록과 float된 블록들이 겹치지 않게 해 준다.

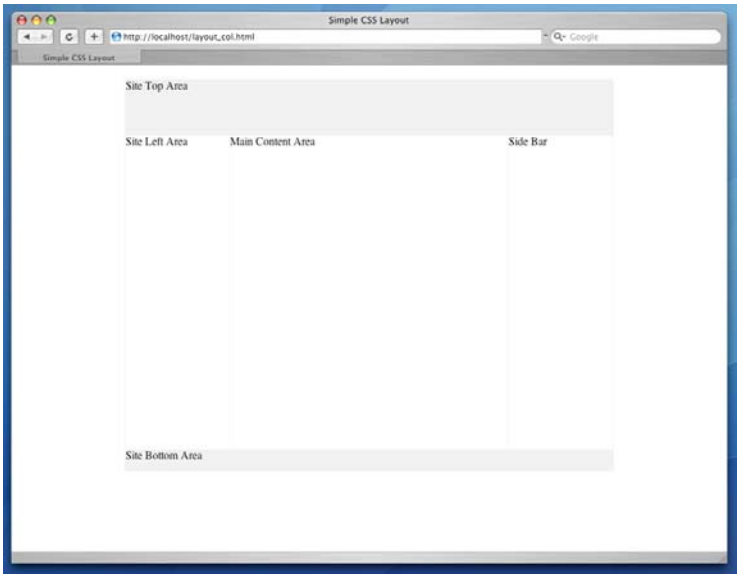
```
#sub,
#body,
#sidebar {
    float: left;
}
#sub,
#sidebar {
    width: 150px;
}
#body {
    width: 400px;
    height: 450px;
}
#foot {
    clear: both;
}
```

블록이나 엘리먼트를 float할때에는 항상 width나 height를 지정해 주어서 확실한 영역을 차지하도록 하게 하고 float이 끝난 위치 바로 다음에 오는 엘리먼트에서는 clear시켜주어서 전체 모양이 일그러 지지 않게 주의 해야 한다.

컬럼으로 이루어진 레이아웃에서는 각 컬럼들 사이에 경계선을 넣는 경우가 많다. 하지만 지금과 같은 경우 float된 블록에 border로 경계선을 넣게 되면 원하는 결과가 나오지 않게 된다. 가장 긴 블록을 기준으로 경계선이 아래까지 내려와야 하는데 float되면 자신이 포함하고 있는 컨텐츠의 높이 만큼만 영역이 확보 되기 때문에 하단까지 border가 생기지 않는다. 그래서 지금과 같은 경우는 #wrapper에 background 속성을 이용해서 경계선을 표현하게 된다.

```
#wrapper {
    background: url(body-col.gif) repeat-y 150px 0;
}
```

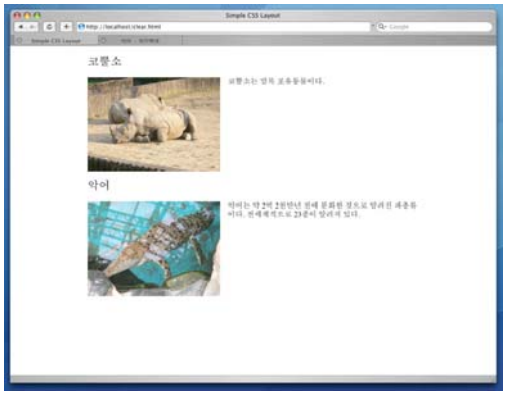
이렇게 하면 3단의 컬럼형 레이아웃이 만들어 진다.



float와 clear

float 속성은 하위의 블록에 영향을 미치기 때문에 이 float 속성을 없애주는 clear 속성을 같이 사용해야 한다. 가장 기본적인 clear 속성의 사용은 float된 블록이나 이미지가 더이상 하위에 영향을 미치지 않게 하기 위해서 사용한다.

컨텐츠 이미지에 float 속성을 사용하게 되면 이미지를 원하는 방향에 정렬 할 수 있다 하지만 컨텐츠의 길이에 따라서 원하지 않는 결과가 나오기도 한다.



왼쪽의 경우 float된 이미지가 하위의 블록에 영향을 미쳐서 페이지 레이아웃이 이상하게 된 것을 볼 수 있다. 이럴때에는 하위블록의 엘리먼트에 clear 속성을 주어서 float된 이미지의 속성을 없애주면 오른쪽과같이 정상적으로 정렬을 할 수 있다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Float and Clearing</title>
<style type="text/css">
body {
    margin: 0;
    padding: 0;
}
#wrapper {
    width: 700px;
    margin: 20px auto;
}
img {
    width: 280px;
    height: 200px;
    float: left;
    margin: 0 1em 1em 0;
}
h2 {
    clear: both;
}
</style>
</head>
<body>
<div id="wrapper">
    <h2>코뿔소</h2>
    <img src="rhinoceros.png" alt="코뿔소">
    <p>코뿔소는 말목 포유동물이다.</p>
    <h2>악어</h2>
    <img src="alligator.png" alt="악어">
    <p>악어는 약 2 억 2 천만년 전에 분화한 것으로 알려진 파충류이다. 전세계적으로
23 종이 알려져 있다.</p>
</div>
</body>
</html>
```

이것은 일반적인 clear의 사용이고, float 속성을 이용해서 레이아웃을 작성하게 되면 아래와 같은 경우를 접하게 된다.

```
<div id="articles">
    <ul id="notice">
    ...
    </ul>
    <ul id="news">
    ...
    </ul>
    <ul id="stats">
    ...
    </ul>
</div>
```

article은 세개의 하위 리스트를 포함하는 블록이고 하위의 ul은 모두 float를 사용하여 가로로 배치되어 있다. 이때에 상위 블록인 #articles에 background속성으로 색을 지정해 주면 색이 나타나지 않는 것을 알 수 있다. float된 ul들은 화면상에서 공간을 차지 하지 않기 때문에 #articles가 float된 ul만을 하위에 가지고 있으면 #articles의 높이가 0이 되어서 마치 배경색이 적용이 안되는 것같이 보이게 된다. 이를 해결하기 위해 과거에는 새로운 엘리먼트를 추가하거나 content속성을 이용하는 등 여러가지 방법이 있었지만 지금은 overflow 속성을 이용해는 방법이 가장 좋은 방법으로 여겨지고 있다.

```
#articles {
    width: 700px;
    margin: 20px auto;
    background: #ddd;
    overflow: auto;
}
```

위와 같이 상위 엘리먼트에 overflow에 auto값을 주게 되면 원하는 배경색이 나오는 것을 볼 수 있다.

목록(List)

웹사이트 컨텐츠의 절반은 리스트라고 할 수 있을 정도로 리스트는 컨텐츠에서 상당히 많이 사용되는 형태이다.

세가지 목록

- ul (unordered list) : 순서가 없는 리스트
- ol (ordered list) : 순서가 있는 리스트
- dl (definition list) : term(<dt>), definition(<dd>) 쌍으로 이루어진 리스트

목록의 여백과 모양

list를 나타내는 엘리먼트의 가장 큰 특징은 불렛이나 번호 등이 자동으로 나온다는 것이다. 그래서 이렇게 자동으로 출력되는 것들의 스타일을 원하는대로 정할 수 있어야 한다. <dl>은 자동으로 출력되는 것이 없고 단지 <dd>에 기본마진이 있는 것만 신경써서 스타일을 정의해 주면 된다. , 은 하나의 <li> 좌측에 기본적으로 여백이 생기고 이 여백에 불렛 등이 나오게 된다. 문제는 이 여백의 조정인데 각 브라우저들 마다 이 여백의 조정이 다르게 구현되어 있다. 예를 들어서 Firefox와 Safari는 padding을 이용해서 이 여백을 조정하게 되는 반면에 Internet Explorer나 Opera 등에서는 margin을 이용해서 이 여백을 조정하게 된다. 그리고 이 불렛이나 숫자의 모양을 직접적으로 제어할 수 있게 되어 있는 list-style 속성도 브라우저별로 차이가 많다. list-style을 위부 이미지 등으로 대체 하여도 정확한 위치를 조절할 수 있는 속성이 없기 때문에 여백의 위치를 조정하게

되면 이 역시 브라우저 별로 다를 모양으로 나오게 된다. 스펙에는 marker를 이용해서 이 문제를 해결 하는 방법을 제시해 놓고 있지만 실제적으로 많은 브라우저에서 이를 구현해 놓고 있지 않기 때문에 기존의 list관련된 속성을 이용해서 list에 스타일을 적용하는 문제는 쉬운 일이 아니다.

background 속성을 이용한 bullet의 표시

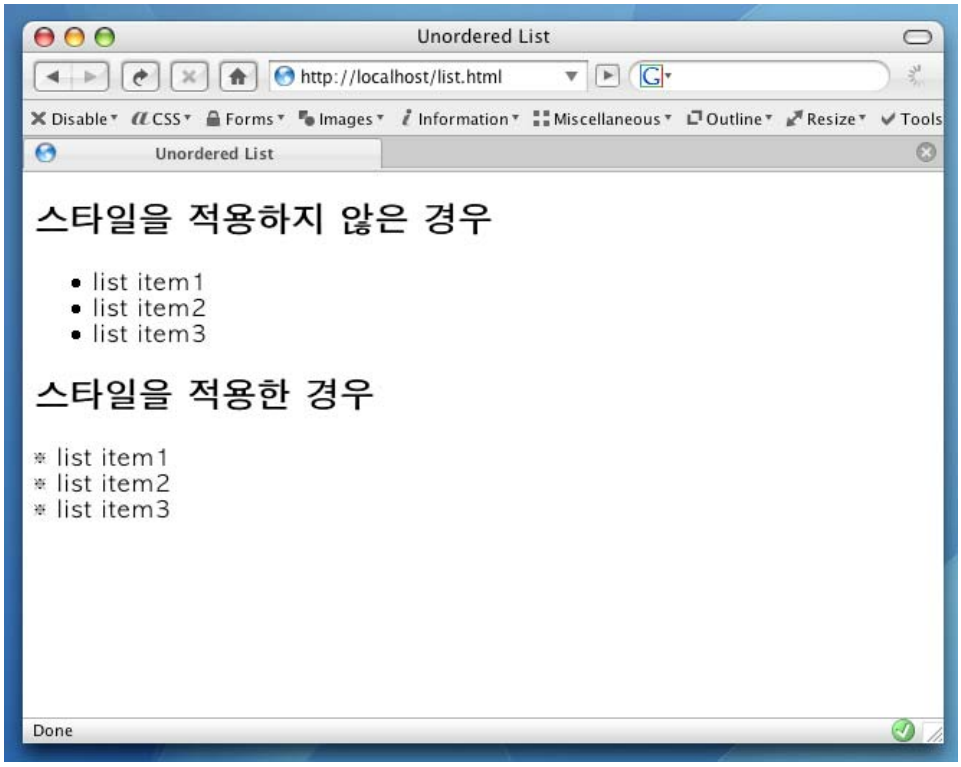
브라우저 호환성과 list관련된 CSS속성의 특성때문에 이들을 이용해서 list에 디자인을 적용하기는 힘들고 보통의 경우 background속성을 이용하여 디자인을 적용한다. background속성을 이용하기 위해서 우선은 브라우저 기본 속성을 초기화 해 준다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>Unordered List</title>
<style type="text/css">
ul {
    margin: 0;
    padding: 0;
    list-style: none;
}
</style>
</head>

<body>
<ul>
    <li>list item1</li>
    <li>list item2</li>
    <li>list item3</li>
</ul>
</body>
</html>
```

이렇게 하면 아무 스타일도 적용되지 않은 세줄의 텍스트와 같이 나오게 된다. 이 <li> 엘리먼트에 background 속성을 이용해서 불렛디자인을 적용하게 된다.

```
li {
    background: url(bullet.gif) no-repeat 0 0.25em;
    padding-left: 15px;
}
```



박스 모델(Box Model)

Box model

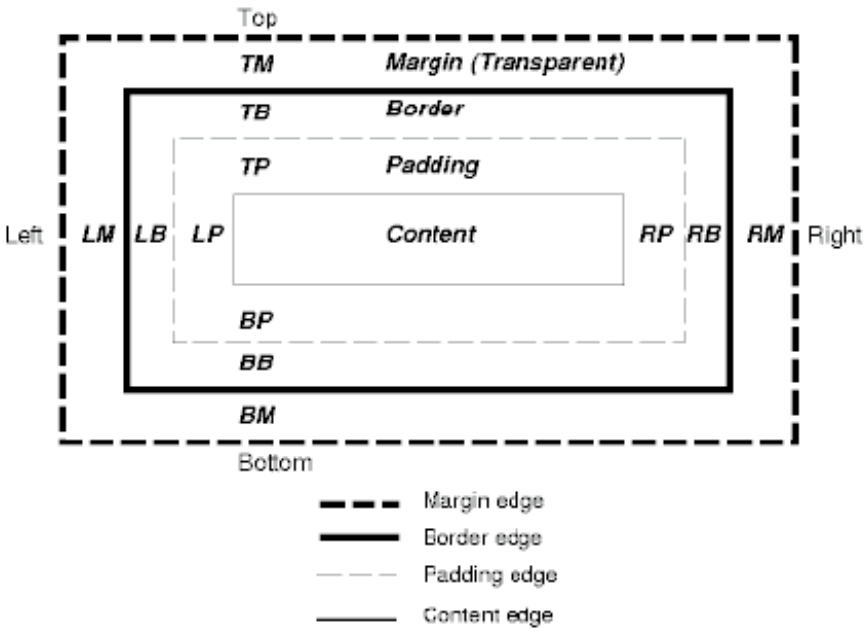


그림 24 CSS 박스 모델

위의 그림은 CSS Box model을 한장의 도표로 나타낸 것이다. Box는 컨텐츠 영역(width, height), 패딩영역(padding), 보더영역(border), 마진영역(margin), offset(top, right, bottom, left)으로 구성되어 있다. 가장 안쪽의 컨텐츠 영역은 width 속성과 height 속성으로 그 크기가 결정된다. 그리고 그 밖으로 패딩, 보더, 마진 영역이 있다. 그리고 position 속성과 함께 지정되는 offset이 있게 된다. 실제적으로 화면에서 보이는 영역은 컨텐츠, 패딩, 보더 영역이고 그 외곽의 마진과 offset은 실제적으로 화면상에서 box라고 인식 되어지지는 않는다.

여기서 가장 주의 해야 하는 것은 컨텐츠 영역의 크기이다. 보통 table을 이용해서 작업하는 것에 익숙한 사람들이 가장 많이 혼동하는 것이 width의 크기를 컨텐츠 영역의 너비로 인식하지 않고 실제로 눈에 보이는 box의 크기로 인식을 하는 것이다. 즉, width를 실제로 눈에 보이는 영역인, width + padding + border의 영역으로 인식하는 것이다. 이것이 CSS2의 box 랜더링과 IE의 box 랜더링이 가장 큰 차이를 보이는 사항이다. CSS2에서는 width나 height를 컨텐츠가 들어갈 수 있는 영역으로 나타내기 때문에 padding이나 border를 적용하게 되면 눈에 보이는 box의 크기는 커지게 된다.

IE의 DOCTYPE Switching

IE는 box모델을 비롯하여 다른 여러 랜더링 특성이 표준과 다르기 때문에 독특한 방법으로 표준을 지원하고 있다. IE의 경우 오랫동안 표준과 다른 랜더링을 유지해 왔기 때문에 갑자기 표준을 지원한다고 하여 랜더링 특성을 바꾸게 되면 이미 IE에 맞춰진 수 많은 사이트들이 IE에서 정상적으로 나오지 않게 될 것이다. 그래서 IE의 trident 엔진은 호환 랜더링과 표준 랜더링 두개의 랜더링 모드를 지원한다. 호환 랜더링 모드는 IE의 하위버전 호환성을 위해서 이전의 IE랜더링을 유지한 모드이고 표준 랜더링 모드는 W3C의 CSS 스펙에 의거한 랜더링을 해주는 모드이다. 이 호환 랜더링과 표준 랜더링의 선택은 문서에 DOCTYPE을 어떻게 선언하는지에 따라서 달라지게 된다.

HTML과 같은 markup언어는 현재 문서에서 사용되고 있는 엘리먼트나 속성들을 따로 정의하고 그 기준에 따라서 작성하게 된다. 이러한 정의를 document type definition이라고 하고 markup 문서의 최 상단에 아래와 같은 방식으로 선언을 하게 된다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

IE는 이 DOCTYPE 선언이 표준대로 선언이 되었는가, 아니면 선언이 되지 않았거나 다른 DTD를 선언하였는가에 따라서 랜더링 모드를 선택하게된다. 따라서 이 선언을 하지 않거나 다른 DTD를 선언하게 되면 호환 랜더링 모드를 선택하게 되고 box에 padding이나 border를 적용하게 되면 컨텐츠 영역인 width나 height가 줄어들게 된다. 그러나 이 DOCTYPE선언을 HTML 4.01이나 XHTML 1.0과 같이 W3C의 선언을 정확하게 할 경우 CSS스펙대로 width나 height영역이 변화 없이 컨텐츠의 영역을 나타내게 된다.

또 하나 주의해야 하는 것이 DOCTYPE 선언을 정확히 했다고 하더래도 DOCTYPE 선언 이전에 어떠한 문자라도 나오게 되면 호환 모드로 랜더링이 되게 된다. 보통은 DOCTYPE 선언 이전에 어떠한 이유로 인해 주석이 나타나게 되면 호환랜더링 모드로 전환이 되어서 페이지가 원하는 대로 나오지 않는 경우가 많다. DOCTYPE 선언 이전에 서버사이드 코드가 들어가게 되면 출력되는 문자가 없게 주의해야 한다.

DOCTYPE 선언이 없으면 유효한 HTML문서가 아니기 때문에 반드시 DOCTYPE을 사

용해야 한다. 하지만 호환성의 문제로 표준 랜더링을 사용하지 않아야 하는 경우가 있는데 이럴 경우에는 역으로 상단에 주석 구문을 추가 해서 호환 랜더링을 사용할 수 있다. 하지만 이러한 버그 현상은 IE7에서는 이미 수정되었고, 새로 구축되는 사이트의 경우 앞으로의 호환성에 더 중점을 두어 표준 랜더링을 사용하는 것이 좋다.

중앙 정렬

사이트가 화면의 중앙에 정렬되어야 하는 레이아웃의 경우 <table> 엘리먼트에 고정된 폭을 지정해 주고 align="center"를 이용해서 정렬 할 수 있었다. 하지만 CSS를 이용한 레이아웃에서는 이러한 방법으로는 구현이 안되고 margin 속성을 이용해서 동일할 효과를 낼 수 있다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<title>center alignment</title>
<style type="text/css">
#content {
    width: 300px;
    padding: 1em;
    border: 1px solid #999;
    margin: 0 auto;
    line-height: 1.5em;
}
</style>
</head>
<body>
<div id="content">
    <p>margin 의 auto 값을 이용해서 보다 다양한 레이아웃을 제작 할 수
있다.</p>
</div>
</body>
</html>
```

폭이 일정한 엘리먼트에 좌우 margin을 auto로 설정하면 브라우저는 최적의 margin 값을 찾게 되고 엘리먼트는 화면의 가운데에 정렬이 되게 된다.

IE5 Tuning

IE5는 위와 같이만 하면 중앙정력이 되지 않고 추가적인 속성이 더 필요 하다. IE5에서는 margin: auto가 정상적으로 작동하지 않기 때문에 텍스트 자체를 중앙정렬 해 줄 필요가 있다.

```
<style type="text/css">
body {
    text-align: center;
}
#content {
    width: 300px;
    padding: 1em;
    border: 1px solid #999;
    margin: 0 auto;
    line-height: 1.5em;
}
</style>
```

이렇게 하면 box는 가운데 정렬이 되었지만 box안의 텍스트가 모두 가운데 정렬이 된 것을 볼 수 있다. 이 텍스트 정렬을 기본값으로 돌리기 위해서 #content에 다시 텍스트 정

렬을 해 준다.

```
#content {
    width: 300px;
    padding: 1em;
    border: 1px solid #999;
    margin: 0 auto;
    line-height: 1.5em;
    text-align: left;
}
```

화면 정 중앙에 위치 시키기

<table> 엘리먼트를 이용할 때에는 valign이나 height="100%" 와 같은 속성들을 이용해서 세로 정렬을 자유롭게 할 수 있지만 CSS에서는 이것이 그렇게 쉽지만은 않다.

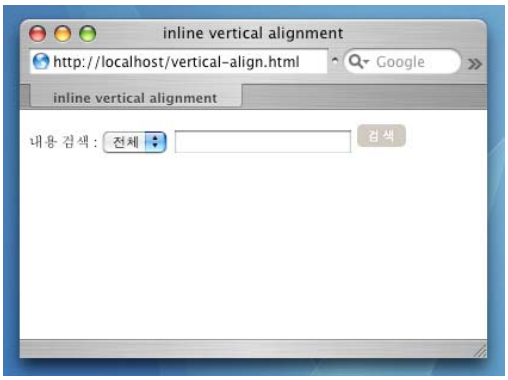
vertical-align 속성

CSS의 vertical-align속성은 적용되는 엘리먼트에 따라서 다른 기능을 보인다. 가장 일반적으로 많이 사용되고 이해하기 쉬운것은 <td> 엘리먼트에 적용되는 것이다. <td>에 적용된 vertical-align 속성은 HTML의 valign 속성과 같은 기능을 한다. 즉, 셀 안의 컨텐츠를 셀 높이의 중앙에 위치 시킨다. vertical-align 속성을 <td> 엘리먼트 외의 사용할 수 있는 곳은 inline 엘리먼트이다. inline 엘리먼트의 가장 대표적인 예는 일반 텍스트, <img>, <input> 엘리먼트 등이다. HTML에서 텍스트를 입력하게 되면 텍스트는 하나의 inline 엘리먼트를 생성하고 그 안에 위치 하게 된다. vertical-align 속성은 이렇게 생성된 inline 엘리먼트에서의 세로 정렬을 의미한다.

<img> 엘리먼트도 inline 엘리먼트인데 <img> 엘리먼트에 vertical-align 속성의 값을 middle로 지정하면 텍스트 inline 엘리먼트 안에서 세로로 가운데에 정렬이 된 것을 볼 수 있다. 예를 들어서 검색 컨트롤을 만들게 될 때 <select> 엘리먼트와 <input> 엘리먼트와 텍스트가 가운데로 정렬이 안되서 <table>을 따로 만들어 그 안에 넣고 중앙 정렬을 해본 경험들이 아마 있을 것이다. 이럴때에 유용하게 사용할 수 있는 속성이 vertical-align 속성이다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<title>inline vertical alignment</title>
<style type="text/css">
body {
    font-size: 0.75em;
}
</style>
</head>
<body>
<form action="">
    <p>
          내용 검색 :
          <select>
                <option>전체</option>
                <option>제목</option>
                <option>이름</option>
                <option>내용</option>
          </select>
          <input type="text">
          <input type="image" src="btnSearch.gif" alt="검색">
```

```
    </p>
</form>
</body>
</html>
```



vertical-align이 적용되지 않은 좌측은 검색 버튼이 약간 올라갔지만 우측은 같은 선상에 정렬 된 것을 볼 수 있다.

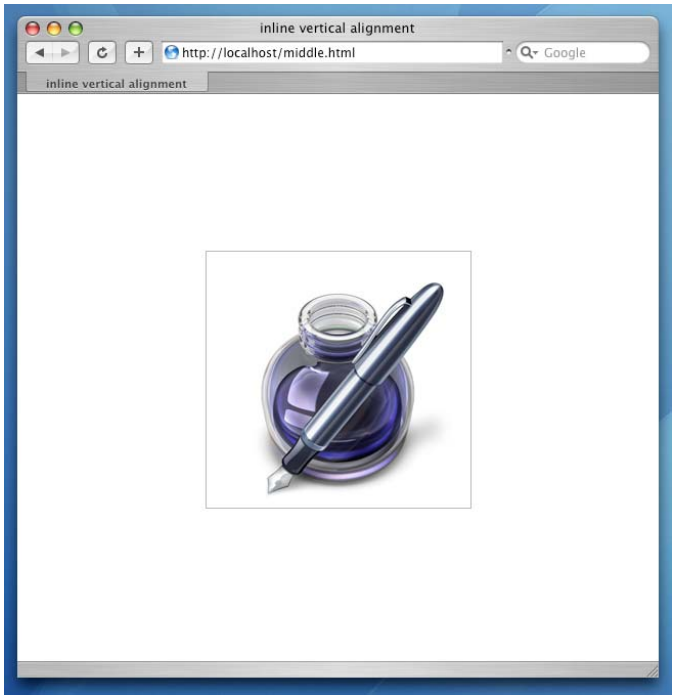
```
* {
    vertical-align: middle;
}
```

vertical-align 속성은 inline 엘리먼트에 적용되는 속성이기 때문에 가로세로 완전히 중앙에 블록을 위치 시키기에는 적합하지 않은 속성이다.

position 속성과 negative margin

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
<title>inline vertical alignment</title>
<style type="text/css">
#middle {
    position: absolute;
    top: 50%;
    left: 50%;
    width: 258px;
    height: 250px;
    margin: -125px 0 0 -129px;
    border: 1px solid #aaa;
}
</style>
</head>
<body>
<div id="middle">
    <img src="pages.png" alt="iWork Pages">
</div>
</body>
</html>
```

화면의 중앙에 위치 시켜야 하기 때문에 absolute position을 사용하고 offset을 50%씩 준다. 그러면 위치시키고자 하는 엘리먼트의 좌상단이 화면의 정 중앙에 오게 될 것이다. 이 상태에서 블록의 크기의 절반 만큼을 margin으로 적용하되 음수로 적용하여 반대로 이동하게 한다. 그러면 위치하고 자 하는 엘리먼트의 정 중앙이 화면의 정중앙과 일치 하게 된다.



이 방법은 정확히 화면의 정 중앙에 엘리먼트를 위치 시키고 브라우저 창의 크기를 변화시켜도 중앙을 유지 하지만 margin값으로 음수를 사용했기 때문에 브라우저의 크기가 엘리먼트의 크기보다 작아질 경우 화면 밖으로 위치해버리게 되어 엘리먼트의 일부분이 보이지 않게 된다. 그래서 이 방법을 사용할 때에는 너무 큰 엘리먼트에는 적용하지 않도록 주의 해야 한다.

100%의 높이를 유지하는 레이아웃

사이트 푸터 부분을 브라우저 크기와 상관 없이 항상 하단에 위치 시키고 컨텐츠의 높이가 브라우저의 높이를 넘어가면 자동으로 스크롤 바가 생기면서 푸터 부분이 브라우저 아래로 넘어가야 한다. 우선 전체 높이를 유지하기 위해서는 높이가 100%인 블록을 이용하고 100%를 넘어가면 브라우저 화면 아래로 넘어가게 하기 위해서 height 속성을 이용하지 않고 min-height 속성을 이용한다.

 엘리먼트를 높이 100%로 유지하기 위해서는 먼저 height에 % 단위로 값을 지정하는 것의 의미를 파악해야 한다. % 단위로 값을 넣을 때 이 % 값의 기준은 상위 엘리먼트가 된다. 다시 말해서 상위 엘리먼트의 높이가 100px라면 100%의 높이는 100px가 되고 50%의 높이는 50px가 된다. 레이아웃을 작성하기 위해서는 <body> 엘리먼트 바로 하위에 100% 높이를 유지하는 엘리먼트가 와야 한다. 하지만 엘리먼트에 height: 100%를 적용하여도 높이가 브라우저 높이만큼 유지되지 않는 것을 볼 수 있다. 이 이유는 상위 엘리먼트인 <body>의 높이가 100%가 아니기 때문이며, 마찬가지로 <body> 엘리먼트 상위의 <html> 엘리먼트도 높이가 100%가 아니기 때문이다. 그래서 100% 높이를 이용하고자 할 때에는 <html> 엘리먼트와 <body> 엘리먼트의 높이를 우선 100%로 고정해 주어야 한다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>Full height layout</title>
<style type="text/css">
html,
body {
    height: 100%;
    margin: 0;
    padding: 0;
}
#head {
    height: 100px;
    background: #ddd;
    position: relative;
    z-index: 1;
}
#body {
    min-height: 100%;
    margin: -100px 0 -50px;
}
#content-area {
    padding: 100px 0 50px;
    height: 300px;
}
#foot {
    height: 50px;
    background: #ddd;
}
</style>
<!--[if IE]>
<style type="text/css">
#body {
    height: 100%;
}
</style>
<![endif]-->
</head>

<body>
<div id="head">
    head(height 100pixel)
</div>
<div id="body">
    <div id="content-area">
         content area
    </div>
</div>
<div id="foot">
    foot(height 50pixel)
</div>
</body>
</html>
```

화면의 높이를 100%로 확보하고 사이트의 전체 높이를 100%로 유지 하기 위해서 #body에 min-height: 100%를 적용한다. min-height 속성은 최소의 높이를 지정해주는 속성인데 컨텐츠의 높이가 지정된 min-height보다 낮으면 그 값을 유지하고 높이가 넘칠 경우 auto로 높이를 설정해 준다. 이렇게 하면 #body의 높이가 브라우저의 높이와 일치되지만 IE는 min-height 속성이 구현되어 있지 않기 때문에 다른 방법으로 구현을 해야 한다. IE의 경우 min-height 속성은 구현되어 있지 않지만 height 속성이 이 min-height 속성과 같은 역할을 한다. IE에서는 컨텐츠가 블록의 높이나 너비보다 크게 되면 블록의 크기가 커지는 것을 볼 수 있다. 이와 같은 역할을 해 주는 것이 height이지만 height를 지정할 경우 다른 브라우저에서는 고정된 높이를 갖기 때문에 하나의 속성으로 이 두가지를 구현 할 수 있는 방법은 없다.

IE는 자체적으로 IE만을 위한 코드를 적용하기 위해서 conditional comment라는 기능을

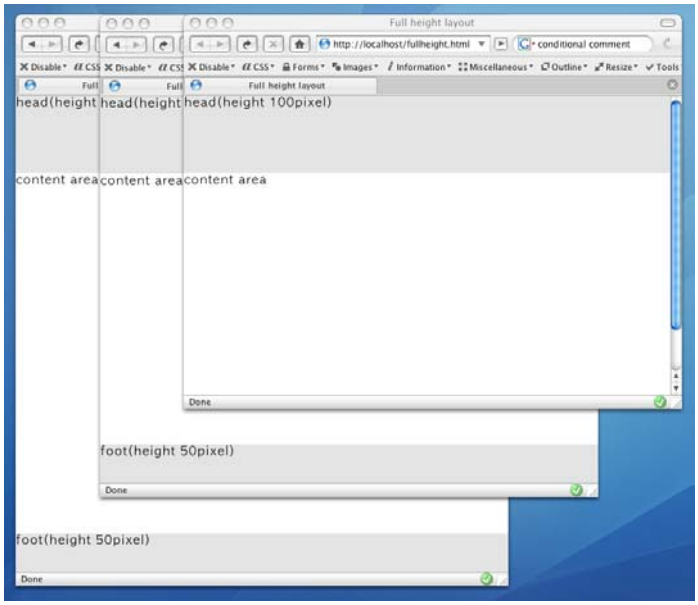
제공하고 있다. 코멘트와 같이 표시된 부분이 조건에 따라서 다르게 작동하게 되는 기능을 말한다. 컨디셔널 코멘트는 아래와 같이 사용한다.

```
<!--[if IE]>
Code for Internet Explorer
<![endif]-->
```

그냥 보기에는 주석구문이지만 [if IE], [endif] 구문으로 IE에서만 작동하는 코드를 넣을 수 있다. 이 구문에는 not 연산자(!)나 less than 연산자(lt), 버전 표시(IE 5)와 같이 IE의 종류에 따라서 적용되는 코드들을 넣을 수 있다. IE의 경우에는 이 구문을 해석하지만 다른 브라우저에서는 이 구문은 그냥 주석 구문이기 때문에 건너뛰게 된다. 이 컨디셔널 커멘트를 이용해서 IE에서는 min-height를 height로 대체하여 작동 될 수 있게 한다.

```
<style type="text/css">
#body {
    height: 100%;
}
</style>
<![endif]-->
```

위와 같이 하면 IE에서는 min-height 속성은 구현이 안되어 있으므로 무시되고 height: 100%가 적용된다. 다음에는 이미 높이가 확보되어 있는 #head와 #foot을 100% 높이에서 제외시기키 위해 negative margin을 이용하고 #content-area에서는 padding을 이용해서 영역을 확보 해 준다.



가장 좌측은 브라우저의 높이가 충분한 경우이고 우측은 브라우저의 높이가 충분하지 않아서 세로 스크롤 바가 생긴 경우이다. 브라우저 사이즈를 변화시켜 보면 브라우저의 높이에 따라서 푸터의 위치가 변하는 것을 볼 수 있다.

테이블(Tables)

CSS레이아웃을 적용할 때 사람들이 의도적으로 테이블 사용을 피하는 것을 볼 수 있다. 하지만 이는 잘못된 행동이며 표 형태로 표현해야 하는 데이터의 경우는 당연히 <table> 엘리먼트를 사용해야 한다. 데이터 테이블은 <thead>, <tbody>, <tfoot>, <th>, <td> 등

다양한 태그를 사용해서 구성하게 되고 이렇게 다양한 태그를 이용해서 제작된 테이블은 접근성도 높아지고 CSS에서도 참조가 쉽위지기 때문에 의미에 맞는 태그를 사용해서 테이블을 표현하는 것이 중요하다.

테이블에 디자인을 적용할 때 대부분의 사람들은 아래와 같이 cellpadding과 cellspacing 값을 조정한다.

```
<table cellpadding="0" cellspacing="0" border="0">
    <tr>
          <td>example</td>
    </tr>
</table>
```

cellpadding은 셀 경계면과 컨텐츠 사이의 패딩 영역을 나타내고 cellspacing은 셀간의 간격을 나타낸다. border는 셀 경계선을 나타내고 기본값이 0이므로 생략해도 무방하다. <table> 엘리먼트에 대부분 기본적으로 적용하는 속성은 다 디자인 적인 요소이고 의미를 나타내는 것이 아니기 때문에 CSS로 처리 하는 것이 더욱 적합하다. cellpadding은 <th> 나 <td> 엘리먼트의 padding으로 제어하고 cellspacing은 border-collapse 속성으로 제어 한다.

```
table {
    border-collapse: collapse;
}
table th,
table td {
    padding: 0;
}
```

<table> 엘리먼트는 기본적으로 복잡한 width calculation algorithm을 거쳐서 랜더링 되는데 이 알고리즘은 셀안의 컨텐츠와 셀에 지정된 width를 바탕으로 최적의 테이블 레이아웃을 계산하는 것이다. 대부분 <table>과 <td>에 정확히 width를 지정을 해도 지정한 대로 width가 나오지 않는 경험을 한 적이 있을 것이고 이때문에 <img src="blank.gif" width="100"> 을 이용해서 width를 강제로 고정하는 방법도 본적이 있을 것이다. 이렇게 정확한 폭을 지정하기가 힘들고 복잡한 알고리즘을 거쳐서 랜더링 속도가 떨어지게 되는데 이를 table-layout이라는 속성을 이용해서 보완 할 수 있다.

table-layout 속성의 기본 값은 auto인데 기본 값일 때에는 테이블이 많이 중첩될 수록 랜더링 속도도 떨어지고 정확한 width 적용이 힘들다. 이 속성의 값을 fixed로 하게 되면 width 계산과정도 없고 정확하게 지정된 값으로 width가 정해지게 된다.

```
table {
    table-layout: fiexd;
}
```

table-layout: fixed를 사용할때 주의 해야 할 점은 이 속성이 적용된 테이블은 셀의 너비를 첫번째 줄의 셀의 너비에 맞게 강제로 정해진다는 것이다. 즉, 첫번째 줄의 속성 외의 다른 줄의 width 값은 무시가 되고, 특정 값이 정해 지지 않으면 정해지지 않은 셀의 너비 만큼 균등하게 나누어서 셀의 너비가 정해진다. 만약에 첫번째 줄에 colspan등을 사용하여서 모든 셀의 너비를 지정할 수 없을 때에는 <colgroup> 엘리먼트와 <col> 엘리먼트를 이용해서 너비를 지정해 주게 된다.

```
<table>
    <colgroup>
          <col style="width: 150px;">
          <col style="width: 100px;" >
          <col style="width: 50px;" >
```

```
    </colgroup>
    <tr>
          <td colspan="2">colspan 이 적용 되어서 원하는 너비를 지정할 수
없다.</td>
          <td>cell2</td>
    </tr>
    <tr>
          <td style="width: 50px;">이 셀의 너비는 무시된다.</td>
          <td style="width: 100px;" >이 셀의 너비는 무시된다.</td>
          <td style="width: 150px;" >이 셀의 너비는 무시된다.</td>
    </tr>
</table>
```

마지막 <tr>의 <td>에 너비를 지정 하였지만 table-layout: fixed;가 적용된 상태에서는 첫번째 줄의 너비만을 참조 하기 때문에 150pixel, 100pixel, 50pixel 순으로 너비가 결정된다.

오른쪽과 같은 데이터는 다른 태그를 사용해서 표현 하는 것 보다 테이블 태그를 사용해서 표현 하는 것이 좋다.

```
<table class="data product-data">
    <thead>
          <tr>
                <th colspan="2"><img
src="/Product/img/dataDivision.gif" alt="구분"></th>
                <th><img src="/Product/img/dataStandard.gif"
alt="규격치"></th>
          </tr>
    </thead>
    <tbody>
          <tr>
```

```
                <th colspan="2">외관</th>
                <td>노란색</td>
            </tr>
            <tr>
                <th colspan="2">옥탄가</th>
                <td>91 이상~94 미만</td>
            </tr>
            <tr>
                <th rowspan="5">증류성상(℃)</th>
                <th>10% 유출온도</th>
                <td>70 이하</td>
            </tr>
            <tr>
                <th>5% 유출온도</th>
                <td>125 이하</td>
            </tr>
            <tr>
                <th>90% 유출온도</th>
                <td>175 이하</td>
            </tr>
            <!-- 중략 -->
            <tr>
                <th colspan="2">메탄올 함량(무게%)</th>
                <td>0.1 이하</td>
            </tr>
        </tbody>
</table>
```

CSS Hack

browser issue

HTML이나 CSS는 W3C에서 제정한 표준이다. 그리고 이 표준을 토대로 하여 브라우저 만드는 회사에서 자신의 브라우저에 구현을 하게 된다. 이 과정에서 브라우저 만드는 회사들이 W3C 의 표준을 완전히 잘 구현을 해 낸다면 별로 문제될 것이 없지만 실제로 브라우저 만드는 회사들 마다 CSS의 구현 정도에 차이가 있다. Microsoft의 Internet Explorer 6는 우리가 가장 많이 사용하고 있는 브라우저이지만 출시가 된지 많은 시간이 지나서 CSS 의 구현 정도가 다른 브라우저보다는 미약하다. 그리고 다른 브라우저들 끼리도 구현된 CSS 스펙이 다르고, 어떠한 브라우저는 스펙을 잘못 구현한 경우도 있다. 가장 좋은 것은 최신의 스펙을 구현한 브라우저를 모든 사람들이 사용하는 것이겠지만 현실적으로 불가능 하기 때문에 웹사이트를 제가할 경우 반드시 하위버젼 호환성과 브라우저 호환성을 염두에 두어야 한다.

Browser specific code

CSS hack은 표준은 아니고 브라우저들 간의 CSS 해석 오류나 차이를 이용하여 특정 브라우저만을 위한 CSS를 만드는 방법이다. 일차적으로는 표준을 준수하여 CSS를 작성을 하고 브라우저들에서 테스트를 해 보고 CSS를 해석하는데 있어서의 차이를 각 브라우저별로 hack을 사용하여 없애는 것이다. CSS Hack 자체가 고의적으로 에러를 발생시키는 것이라고 볼 수도 있기 때문에 CSS Hack의 사용에 대해서는 아직 논쟁이 많다. 하지만 현재까지 나온 CSS Hack들은 경험적으로 여러번 테스트를 거쳤기 때문에 사용해도 큰 문제가 없고 다양한 브라우저를 지원하기 위해서 어쩔 수 없이 사용해야 하는 경우도 있다.

Microsoft Internet Explorer 용 CSS hack

```
div.ie-hack {
    width: 100px;
    padding: 20px;
}
* html div.ie-hack {
    width: 140px;   /* IE width */
}
```

IE의 경우 표준 DTD 를 사용하여 표준 모드의 랜더링을 사용하지 않으면 padding과 border의 해석이 표준과 다르게 처리된다. 표준에서는 100pixel width에 20px의 padding을 적용하면 화면에서 차지하는 블록의 너비는 140pixel이 되지만 IE 는 여전히 100pixel로 표현 된다. 따라서 표준에 맞게 랜더링이 되게 하기 위해서 IE에서는 width를 140pixel로 해주어야 한다. "* html E" selector를 사용하게 되면 다른 브라우져들은 html 상위 element가 존재하지 않기 때문에 인식이 안되지만 IE는 인식을 하게 되고 블록의 크기가 140pixel이 된다.

Microsoft Internet Explorer 5 용 CSS hack

```
div.ie5-hack {
    padding: 20px;
    width: 140px;
    voice-family: "\"}\"";
    voice-family: inherit;
    width: 100px;   /* IE5 ignore this line */
}
```

IE6 는 표준 모드와 호환모드 두개의 랜더링 모드를 가지고 있다. 이 것은 <!DOCTYPE>을 선언 하는 것으로 제어가 되는데 <!DOCTYPE> 을 표준으로 선언하게 되면 IE6는 표준 스펙을 따라서 랜더링이 되고 <!DOCTYPE> 선언을 다르게 하거나 하지 않으면 호환모드로 랜더링이 된다. IE5는 이와 같은 doctype switching을 지원하지 않기 때문에 표준모드를 사용하게 되면 IE6와 IE5는 서로 다르게 화면을 랜더링 하게 된다. 그래서 표준 모드를 지원 하지 않는 IE5를 위한 CSS hack을 사용해야 하는 경우가 발생한다. 위의 코드를 이용하게 되면 IE 5는 첫번째 voice-family 선언에서 "}" 을 CSS 선언의 종료로 인식하고 CSS 선언이 끝난 것으로 해석하게 된고 width는 140pixel이 된다. 반면 다른 현대 브라우져들은 그 다음줄도 인식을 하기 때문에 width는 100pixel이 된다.

이 외에도 CSS Hack은 브라우져 별로 많은 종류가 있다. 하지만 대부분의 브라우져 사용자들은 최신 버전으로 자신들의 브라우져를 업데이트 하여 사용하는 빈도가 많기 때문에 다른 브라우져용 hack을 사용하는 경우는 매우 드물다. hack을 사용해야 하는 대상들은 보통 컴퓨터 사용에 익숙하지 않아서 처음 설치되어 있는 IE 브라우져를 그냥 사용하는 사람들이기 때문에 요즘에는 거의 IE용 hack외에는 사용되는 경우가 드물다. 그리고 만약 사용자가 구형 브라우져를 사용하여 CSS가 제대로 해석되지 않다라도 마크업이 완전하면 디자인은 완전하지 않더라도 컨텐츠를 이용하는 데에는 불편이 없기 때문에 충분한 접근성을 보장 할 수 있다.

실전 예제를 통한 CSS 레이아웃

지금까지는 간략하고 부분적인 CSS디자인 적용 방법들을 살펴 보았다. 이 장에서는 실제 디자인된 사이트를 분석해 보고 각 단계에서 필요한 부분들을 알아 봄으로써 하나의 완성된 페이지를 만드는데 필요한 사항들을 살펴 보겠다.

설전 예제로 분석해볼 페이지는 재정경제부의 재경부 안내페이지이다

URL: http://www.mofe.go.kr/about/about_01.php

전체적인 구조와 마크업

전체 페이지 구조를 제작할 때에는 가장 중요한 기준은 화면 구성요소의 의미와 그들 간의 그룹핑이다. 페이지를 구성하고 있는 요소들의 관계와 그룹핑 데이터가 마크업에 반영이 되어야 한다. 기존의 테이블을 이용한 레이아웃에서는 이러한 관계와 그룹핑 정보를 마크업에 넣는 것이 불가능 하지만 CSS와 div를 이용한 레이아웃에서는 이러한 데이터를 포함 시키는 것이 가능하고 이렇게 함으로써 보다 의미에 맞는 마크업을 제작할 수 있게 된다. 재경부의 서브페이지는 크게 4부분으로 구분이 되어 있다.

- 사이트의 로고와 상단 메뉴
- 좌측메뉴
- 컨텐츠 영역
- 사이트 하단

각 부분들은 하나의 div 안에 구성되어 있고 각자의 고유한 id를 갖게 된다. 이 id는 해당 div의 이름이면서도 그 div의 역할과 의미를 표현하는 적절한 것을 선택해야 한다. 예를 들어서 id="b_e-13" 과 같은 id는 그 id만을 봐서는 무슨 의미 인지 전혀 알 수 없고 코드를 읽는 사람에게 어려움을 주게 되므로 피해야 하고 간결하면서도 의미를 잘 나타내 주는 단어를 선택해야 한다. 이 id는 javascript가 DOM을 통해 엘리먼트에 접근할 때에나 CSS selector에서 rule을 정의할 때 사용하게 된다.

DTD(Document Type Definition)의 선택

마크업 작성에 있어서 가장 먼저 고려해야 하는 것은 어떠한 DTD를 따라서 페이지를 작성할 것인가 하는 점이다. DTD는 문서의 작성방법과 여러 태그네임과 속성등을 경정 짓기 때문에 선택에 있어서 신중을 기할 필요가 있다.

우선은 HTML을 이용할 것인지, XHTML을 이용할 것인지를 정하게 된다. HTML과 XHTML의 가장 큰 차이는 XML의 문법 규칙을 따르게 되는지 그렇지 않는지 이다. XML문법을 딸게 된다면 XHTML을 사용하게 될 것이고 그렇지 않다면 HTML을 사용하게 될 것이다. 그리고 그 다음으로는 디자인 적인 요소를 완전히 사용하지 않고 CSS로 대체할 것인지 아닌지를 판단해야 한다. 디자인 적인 요소를 완전히 사용하지 않을 것이라면 Strict DTD를 사용하고 그렇지 않다면 Transitional DTD를 사용하게 된다.

많은 사람들이 웹표준이라고 하면 당연히 XHTML을 사용해야 하는 것으로 생각하는데 이는 올바른 생각이 아니다. DTD를 선택한다는 것은 정해진 문법 규약을 따르겠다의미이기 때문에 선택한 DTD에 따라서 표준이거나 비표준이 아니고 반드시 XHTML을 사용할 필요는 없다. 그리고 XHTML을 사용하게 된다면 XML문법을 완벽하게 지켜서 페이지를 제작해야 하기 때문에 만약 그러한 준비가 되어 있지 않은 상황이라면 XHTML을 사용하는 것 보다는 HTML을 사용하는 것이 더 바람직하다.

그리고 XHTML은 단순히 문법뿐만이 아니라 mime-type도 XML의 것을 따르겠다는 의미이다. 특히나 XHTML 1.1은 application/xml+xhtml mime-type으로 배포되어야 함에도 불구하고 많은 사람들이 text/html 로 배포하고 있는 경우가 많다. 이것과 같이 규약을 정확히 지키지 못하게 된다면 차라리 HTML 4.01 Strict를 사용하고 정확한 문법을 지키는 것이 훨씬 더 좋은 선택이다.

재정경제부는 XHTML 1.0 Transitional DTD로 제작이 되었다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr"
/>
<title>재정경제부에 오신 것을 환영한다.</title>
</head>
<body>
<div id="head">
</div>
```

```
<hr />
<div id="sub">
</div>
<hr />
<div id="body">
</div>
<hr />
<div id="foot">
</div>
</body>
</html>
```

각 부분들은 <hr />을 이용하여 의미적으로 명확하게 구분을 해 준다. 텍스트 브라우저와 같이 완전한 스타일을 볼 수 없는 상황에서 페이지 구분을 명확하게 해 준다. 이 구분은 일반 브라우저에서는 스타일로 보이지 않게 처리 된다.

상단 부분(#head)

상단 부분인 #head에는 사이트 로고와 사이트메뉴, 상단메뉴가 위치하게 된다.

```
<div id="head">
    <h1><a href="/"><img src="/images/nav/logo.gif" alt="재정경제부"
/></a></h1>
    <div id="site-menu">
         <ul>
              <li><a href="http://english.mofe.go.kr/"><img
src="/images/nav/siteMenu01.gif" alt="English" /></a></li>
              <li><a href="http://kids.mofe.go.kr/"><img
src="/images/nav/siteMenu02.gif" alt="어린이 홈페이지" /></a></li>
              <li><a href="/guide/sitemap.php"><img
src="/images/nav/siteMenu03.gif" alt="사이트맵" /></a></li>
         </ul>
    </div>
    <div id="top-navigation">
         <ul class="depth1">
              <li class="menu-1"><a
href="/news/news_01_latest.php">재경부뉴스</a>
                   <ul class="depth2">
                        <li class="menu-1-1"><a
href="/news/news_01_latest.php">보도자료</a></li>
                        <li class="menu-1-2"><a
href="http://mofe.news.go.kr/" target="_blank">재경부뉴스 </a></li>
                        <li class="menu-1-3"><a
href="/news/news_03.php?action=list">언론보도해명</a></li>
                        <!-- 이하 메뉴 생략 -->
                   </ul>
              </li>
              <li class="menu-2"><a href="/lib/">경제정보</a>
                   <ul class="depth2">
                        <li class="menu-2-2"><a
href="/lib/lib_02.php?action=list">함께읽고싶은보고서</a></li>
                        <li class="menu-2-1"><a
href="/lib/lib_01_01.php">간행물</a></li>
                        <!-- 이하 메뉴 생략 -->
                   </ul>
              </li>
              <!-- 메뉴 생략 -->
         </ul>
         <script type="text/javascript">
         initTopNavigation();
         </script>
    </div>
</div>
```

사이트 로고는 페이지에서 가장 중요하기도 하고 상단 부분에서도 가장 중요한 부분이기

때문에 <h1> 태그를 사용하였다. 그리고 나머지 메뉴 부분들은 메뉴의 리스트 이기 때문에 <ul> 태그를 사용하였고 하나의 메뉴 아이템은 하나의 <li> 태그로 나타내어 진다. 그리고 그 하위의 메뉴가 있을 경우에는 <li> 안의 메뉴 다음에 을 이용해서 표기가 되어 있다. 그리고 끝부분에 javascript를 이용하여 전체 메뉴의 효과를 지정해 준다. 이 javascript 부분은 상당히 중요한 역할을 하는 부분이지만 이 글에서 논하기에는 범위를 벗어나는 부분이기 때문에 상세히 다루지는 않겠다. 한가지 특기할 만한 사항은 이 javascript가 현재 text로 되어 있는 메뉴 항목을 이미지로 바꾸어 주는 역할을 한다는 것이다. 메뉴를 텍스트로 넣고 javascript로 바꾸는 방법을 택한 것은 표준이나 접근성과는 별개의 문제이고 사이트의 유지 관리의 효율성을 위한 것으로 이 역시 이 글에서 다룰 범위는 아니기 때문에 자세한 언급은 하지 않겠다.

기본 CSS구조

한개의 페이지라면 그렇지 않겠지만 사이트 전체를 CSS를 이용해서 제작하게 되면 CSS가 상당히 덩치가 커지고 복잡해 지게 된다. 따라서 CSS의 구조와 그에 따른 범위를 명확히 정하지 않게 되면 차후에 유지관리가 매우 힘들어 지기 때문에 이를 미리 명확하게 규정지을 필요가 있다.

CSS파일들은 각 파일들의 범위에 따라서 구분을 지어주는 것이 좋다. 물론 전체 CSS파일이 일 이백줄 내외로 간단하다면야 굳이 파일을 분리할 필요는 없겠지만 사이트가 커지게 되면 하나의 파일로는 관리도 안되고 다른 사람과 협업을 할 때에도 conflict가 많이 발생하기 때문에 의미와 섹션별로 구분을 해 주는 것이 좋다. 그리고 실제 디자인과는 관련이 없지만 항상 기본적으로 설정해 주어야 하는 파일 같은 것도 따로 구분을 해 주는 것이 좋다. 아래는 재정경제부의 기본 element 설정을 해 주는 base.css 파일이다.

```
@charset "euc-kr";
/*
 * default definition
 */
body {
    margin: 0;
    padding: 0;
    font-size: 0.75em;
    line-height: 1.5em;
    font-family: Dotum, "돋움", sans-serif;
}
form {
    margin: 0;
    padding: 0;
}
hr {
    display: none;
}
li img {
    vertical-align: middle;    /* for IE imge margin */
}
p, div, th, td, select {
    color: #78777C;
}
a:link, a:visited {
    color: #78777C;
    text-decoration: none;
}
a:active, a:hover {
    text-decoration: underline;
}
img,
input.type-image {
```

```
    border: 0 none;
}
input.type-text,
textarea {
    border-top: 1px solid #797979;
    border-right: 1px solid #D4D1C8;
    border-bottom: 1px solid #D4D1C8;
    border-left: 1px solid #797979;
    background: #fff;
}
input.type-text:hover,
input.type-text:focus,
textarea:hover,
textarea:focus {
    background-color: #FFFFCE;
}
input, select, textarea {
    vertical-align: middle;
    font-size: 1em;
    color: #78777C;
}
span.button,
img.button,
a.button {
    cursor: pointer;
    vertical-align: middle;
}
```

base.css에서는 <body>엘리먼트나 <form>, <img> 엘리먼트등 기본 설정이 필요한 엘리먼트들과 폰트정의 등과 같이 사이트 전반에 따라서 설정해 주어야 하는 rule들이 있다. <form>을 사용하면 공백이 생겨서 <tr> 사이에 넣는 경우가 있는데 이는 HTML상으로도 오류이기 때문에 잘못된 이용이다. <form>에 기본 여백이 있기 때문인데 이러한 것과 같이 기본 설정이 필요한 것들의 설정들이 포함되어 있다. 이 base.css는 모든 페이지에서 포함해야 하는 파일이다. 실제적으로 지금 디자인작업을 하게되는 파일은 layout.css이고 이 layout.css 상단에서 아래와 같이 base.css를 포함하게 된다.

```
@charset "euc-kr";
@import url("base.css");
```

폭이나 높이가 고정되어 있는 경우

페이지의 상단이나 하단과 같이 높이가 고정되어 있는 경우, 그리고 좌측과 같이 너비가 고정되었는 경우와 같이 유동적인 크기를 반영해 줄 필요가 없는 경우에는 position 속성을 이용해서 위치를 설정해 주기가 수월하다.

우선은 #head의 크기와 배경을 지정하고 하위 엘리먼트에서 absolute position을 사용하기 위해서 position속성을 relative로 설정해 준다.

```
#head {
    position: relative;
    height: 92px;
    overflow: hidden;
    background: url(/images/nav/head.gif) repeat-x 0 64px;
}
```

로고가 들어 있는 h1은 absolute position을 사용해서 위치를 설정해 준다.

```
#head h1 {
    position: absolute;
    top: 8px;
    left: 14px;
    margin: 0;
    z-index: 5;
}
```

<h1>~<h6>까지의 태그에 스타일을 적용할때에는 항상 이 엘리먼트에 기본적으로 margin이 있다는 것을 염두에 두어야 한다. 이것을 정확히 파악하지 못할 경우 "알 수 없는 이유"로 여백이 발생하여 어려움을 겪을 수 있다. 이것은 다른 <p>나 과 같은 엘리먼트에도 적용 된다.

사이트 메뉴는 가로로 배치되어 있는 리스트 이다. 일단은 로고와 같이 absolute position으로 위치를 지정하고 를 float시켜주어서 가로로 배치하게 된다. 은 대부분의 경우 커스터마이징 된 불렛을 사용하기 때문에 보통 margin, padding, list-style등을 초기화 한 후에 디자인을 적용한다.

```
#site-menu {
    position: absolute;
    top: 7px;
    left: 558px;
    width: 170px;
    z-index: 2;
}
#site-menu ul {
    width: 170px;
    margin: 0;
    padding: 0;
    list-style: none;
}
#site-menu ul li {
    float: left;
}
```

상단 메뉴는 좌측 하단의 그라데이션 부분을 처리하기 위해서 로고 영역부터 사이트 우측 경계까지 넓게 자리 잡고 있다. 그리고 하위의 메뉴들을 위해서 relative position으로 되어 있다.

```
#top-navigation {
    position: relative;
    width: 939px;
    height: 91px;
    background: url(/images/nav/topNav.gif) no-repeat 0 64px;
}
```

하위의 메뉴들은 리스트 형태이고 이를 absolute position으로 위치를 잡는다. 그리고 좌측으로 배열하기 위해서 <li> 엘리먼트를 float 시켜준다.

```
#top-navigation ul.depth1 {
    position: absolute;
    top: 24px;
    left: 211px;
    margin: 0;
    padding: 0;
    list-style: none;
    z-index: 2;
    height: 40px;
}
#top-navigation li {
    float: left;
}
```

그 하위의 서브 메뉴는 기본적으로는 탑메뉴와 동일하다. 단, 메뉴의 구동 자체가 javascript로 이루어져 있기 때문에 일부 javascript의 도움을 얻어서 디자인을 적용한 것이 있다. 각 서브메뉴 항목들을 구분지어주는 사선의 경우 서브 메뉴의 에 background 속성을 이용하여 이미지를 삽입한 것인데 첫번째 는 사선이 없어야 하기 때문에 이것을 javascript로 처리 하였다. 그리고 각 메뉴별로 위치는 특별한 규칙이 없기 때문에 길이에 따라서 CSS로 정의 하였다.

```
#top-navigation ul.depth2 {
    margin: 0;
    padding: 0;
    list-style: none;
    position: absolute;
    left: 0;
    width: 800px;
    bottom: -27px;
    display: none;
}
#top-navigation ul ul.depth2 li {
    background: url(/images/menu/top.gif) no-repeat;
    padding-left: 6px;
}
#top-navigation li.menu-1 ul.depth2 {
    left: 0;
}
#top-navigation li.menu-2 ul.depth2 {
    left: -10px;
}
#top-navigation li.menu-3 ul.depth2 {
    left: 135px;
}
#top-navigation li.menu-4 ul.depth2 {
    left: 115px;
}
#top-navigation li.menu-5 ul.depth2 {
    left: -20px;
}
#top-navigation li.menu-6 ul.depth2 {
    left: -165px;
    width: 1040px;
}
#top-navigation li.menu-7 ul.depth2 {
    left: 190px;
}
```

좌측 영역 (#sub)

좌측은 페이지의 현재 메뉴명과 서브메뉴, 검색, 외부 링크 영역으로 이루어져 있다.

```
<div id="sub">
    <div id="visual"></div>

    <h1><a href="/about/about_01.php"><img
src="/images/nav/heading/about.gif" alt="재경부안내" /></a></h1>
    <div id="sub-navigation">
         <ul class="depth2">
              <li class="menu-7-1"><a
href="/about/about_01.php">인사말</a></li>
              <li class="menu-7-2"><a
href="/about/about_02_1.php">부총리/차관</a>
                   <ul class="depth3">
                        <li class="menu-7-2-1"><a
href="/about/about_02_1.php">부총리 프로필</a></li>
                        <li class="menu-7-2-2"><a
href="/about/about_02_2.php">제 1 차관 프로필</a></li>
                        <!-- 메뉴 항목 생략 -->
                   </ul>
              </li>
              <li class="menu-7-3"><a
href="/about/about_03_1.php">재정경제부는</a>
                   <ul class="depth3">
                        <li class="menu-7-3-1"><a
href="/about/about_03_1.php">소개</a></li>
                        <li class="menu-7-3-2"><a
href="/about/about_03_2.php">연혁</a></li>
                        <!-- 메뉴 항목 생략 -->
```

```
                    </ul>
               </li>
               <!-- 메뉴 항목 생략 -->
          </ul>
     </div>
     <script type="text/javascript">
     initSubNavigation();
     </script>
     <form action="/about/man_search.php" method="post" name="member-
search-form">
          <div id="search-member">
               <h2><label for="layout-memger-search-name"><img
src="/images/nav/searchMemberH2.gif" alt="직원검색" /></label></h2>
               <input id="layout-memger-search-name" name="name"
type="text" class="type-text" />
               <input name="image" type="image"
src="/images/nav/btnGo.gif" alt="GO" />
               <input type="hidden" name="flag" value="3" />
          </div>
     </form>
     <div id="ext-link">
          <div class="link1">
               <h2><a href="#ext-link1">실국 홈 바로가기</a></h2>
               <div id="ext-link1" class="list">
                    <ul>
                         <li class="menu-8"><a
href="/division/off_pm/">정책홍보관리실</a></li>
                         <li class="menu-9"><a
href="/division/off_tc/">세제실</a></li>
                         <!-- 메뉴 항목 생략 -->
                    </ul>
               </div>
          </div>
          <div class="link2">
               <h2><a href="#ext-link2">소속기관 바로가기</a></h2>
               <div id="ext-link2" class="list">
                    <ul>
                         <li><a
href="http://www.publicfund.go.kr/kor_pf/index.html">공적자금관리위원회
</a></li>
                         <li><a
href="http://www.ntt.go.kr/">국세심판원</a></li>
                         <!-- 메뉴 항목 생략 -->
                    </ul>
               </div>
          </div>
          <div class="link3">
               <h2><a href="#ext-link3">재경부 소관 공공기관</a></h2>
               <div id="ext-link3" class="list">
                    <ul>
                         <li><a
href="http://www.shinbo.co.kr">신용보증기금</a></li>
                         <li><a
href="http://www.komsco.com/">한국조폐공사</a></li>
                         <!-- 메뉴 항목 생략 -->
                    </ul>
               </div>
          </div>
          <div class="link4">
               <h2><a href="#ext-link4">관련 연구소</a></h2>
               <div id="ext-link4" class="list">
                    <ul>
                         <li><a
href="http://www.kdi.re.kr/">한국개발연구원</a></li>
                         <li><a
href="http://www.kiep.go.kr/">대외경제정책연구원</a></li>
                         <!-- 메뉴 항목 생략 -->
```

```
                    </ul>
                </div>
            </div>
            <script type="text/javascript">
            initExtLink();
            </script>
        </div>
</div>
```

좌측 영역에서 가장 중요한 현재 메뉴 타이틀은 <h1> 태그를 사용하였다. 그리고 메뉴들은 상단과 마찬가지로 중첩된 <ul> 엘리먼트를 사용하였다. 검색 부분에서의 타이틀은 좌측 영역의 서브격의 타이틀 이기 때문에 <h2> 태그를 사용하였고 <form>과 <label>, <input>을 이용해서 구성되고 있다. 외부 링크는 링크의 나열이기 때문에 을 사용하고 태그로 각각을 그룹핑해주고 있다. 그리고 javascript로 좌측과 외부 링크 영역을 제어하고 있기 때문에 <script> 태그가 사용되고 있다.

서브 영역은 absolute position을 이용해서 위치가 정해져 있다.

```
#sub {
    position: absolute;
    top: 92px;
    left: 0;
    width: 209px;
    z-index: 5;
}
```

그리고 비주얼 영역과 타이틀 영역의 크기와 여백을 정해 준다. 비주얼 영역은 처음에는 계획 되었지만 현재는 별 의미가 없는 블록이다. 즉 현재는 신경쓰지 않아도 되는 garbage 코드이다.

```
#visual {
    width: 209px;
    height: 121px;
}
#sub h1 {
    margin: 33px 0 22px 12px;
}
```

좌측 메뉴는 여백과 배경 이미지 만으로 구현 되어 있다.

```
#sub #sub-navigation {
    margin: 15px 0 0;
}
#sub #sub-navigation ul.depth2 {
    margin: 0;
    padding: 0 0 2px;
    list-style: none;
    background: url(/images/nav/subNavigation.gif) no-repeat 0 100%;
}
#sub #sub-navigation ul.depth3 {
    margin: 0 0 0 24px;
    padding: 2px 0 3px 4px;
    list-style: none;
    background: url(/images/menu/subD3.gif) no-repeat 0 100%;
    display: none;
}
* html #sub #sub-navigation ul.depth2 li {
    height: 1%;
}
* html #sub #sub-navigation ul.depth2 li img {
    float: left;
}
```

마지막 두개의 CSS rule은 앞에도 소개가 되어 있는 IE용 CSS hack이다.

IE의 hasLayout 속성과 리스트 오류

IE의 랜더링 오류중의 상당수가 hasLayout이라는 속성과 관련이 있다. 보통의 블록들은 hasLayout 속성일 false 인데 이 경우 CSS의 랜더링이 정상적으로 이루어 지지 않는 경우가 많다. 이때에 해당 블록에 width나 height를 지정해 주게 되면 hasLayout속성이 true로 바뀌게 되고 이렇게 되면 표준대로 랜더링 되는 것을 자주 경험할 수 있다. 여기서도 li에 height: 1%를 적용함으로써 hasLayout 속성을 true로 바꿔주고 랜더링 오류를 고친 것이다.

그리고 아래의 <img> 엘리먼트를 float시켜준 것은 안에 margin이 없음에도 불구하고 <li> 엘리먼트 끼리 공간이 생기는 것을 방지하기 위한 것이다. float시키는 방법 외에도 line-height 속성을 조정하거나 font-size: 1px와 같이 안의 것의 크기를 아주 작게 만들면 해결이 되곤 한다.

검색 부분은 배경 이미지와 여백의 조정으로 구현 된다.

```
#search-member {
    background: url(/images/nav/searchMember.gif) no-repeat;
    padding: 8px 0 13px 15px;
    margin: 10px 0 10px 10px;
}
#search-member h2 {
    margin: 0 0 10px;
}
```

외부 링크는 같은 형태의 리스트 4개가 있는 모양인데 클릭이 이루어 지는 부분은 타이틀인 <h2>로 되어 있고 화면에 나오는 리스트 부분은 로 되어 있다. 이들 각각은 하나의 relative position 블록 안에 위치 하게 된다.

```
#ext-link {
    background: #FAF9F2;
    padding: 3px 3px 0;
    width: 173px;
    margin: 5px 10px 0;
    overflow: visible;
}
#ext-link h2 {
    width: 164px;
    font-size: 0.9em;
    line-height: 21px;
    background: url(/images/nav/extLinkH2.gif) no-repeat;
    padding: 2px 0 0 9px;
    margin: 0 0 3px;
}
#ext-link div.link1,
#ext-link div.link2,
#ext-link div.link3,
#ext-link div.link4 {
    position: relative;
}
```

이 리스트들은 클릭했을 때에 다른 것들 보다 맨 위에 있어서 리스트 내용 전체가 나와야 하는데 나중에 생성된 블록의 높이가 높기 때문에 현재와 같은 상태에서는 아래의 타이틀이 리스트 위로 올라오는 것을 볼 수 있다. 이를 해결 하기 위해서 타이틀과 리스트의 z-index를 조절해 준다.

```
#ext-link div.link1 {
    z-index: 7;
}
#ext-link div.link1 div {
    z-index: 8;
```

```
}
#ext-link div.link2 {
    z-index: 5;
}
#ext-link div.link2 div {
    z-index: 6;
}
#ext-link div.link3 {
    z-index: 3;
}
#ext-link div.link3 div {
    z-index: 4;
}
#ext-link div.link4 {
    z-index: 1;
}
#ext-link div.link4 div {
    z-index: 2;
}
```

화면상에서위에 위치하는 것의 z-index는 아래의 것보다 크고 클릭시 나오는 리시트의 z-index는 자신을 포함하는 블록의 z-index보다 커야 한다.

여기까지 되었으면 나머지는 여백과 위치, 색등을 지정해 주면 된다.

```
#ext-link div.link1 h2 a {
    color: #534C41;
}
#ext-link div.link2 h2 a {
    color: #676054;
}
#ext-link div.link3 h2 a {
    color: #7D7465;
}
#ext-link div.link4 h2 a {
    color: #948876;
}
#ext-link div.list {
    position: absolute;
    top: 21px;
    left: 0;
    display: none;
    background: url(/images/nav/extLinkUl.gif) no-repeat 0 100%;
}
#ext-link div.list ul {
    margin: 0;
    padding: 5px 9px 2px;
    width: 156px;
    list-style: none;
}
```

한가지 특징적인 것은 리스트의 디자인을 나타내기 위해서 이미지를 단 하나만을 사용했다는 것이다. 원래대로라면 리스트의 아래 부분과 리스트의 중간 부분을 두개로 나누어서 해야 했겠지만 리스트의 길이가 아주 유동적인 것도 아니고 이미지가 복잡하지 않아서 용량 문제가 별로 심하지 않을 경우 하나의 이미지로 제작하는 것이 HTML도 간소화 해지고 작업에 있어서의 효율성도 늘어 나게 된다. 그리고 이미지의 갯수가 줄어들기 때문에 서버의 부하도 줄어들게 된다는 장점이 있다.

본문 영역 (#body)

보통의 본문 영역의 경우 좌측 영역과의 구분이라든가 본문 영역의 구분을 위해서 배경이미지를 사용하게 된다. 그리고 본문 영역은 본문의 길이의 제한이 없어야 되기 때문에 이 배경 이미지는 repeat-y 속성을 이용해서 세로의 구분을 해 주는 경우가 많다. 재정경제

부 홈페이지 역시 세로의 구분선이 존재 하지만 이에 더에서 emboss와 drop shadow를 이용한 이미지도 구분을 위해서 사용이 되고 있다. 배경이미지는 하나의 div에 하나밖에 사용할 수가 없기 때문에 이와같은 레이아웃을 구현하기 위해서 #body-wrapper라는 div블록을 하나 더 추가 한다.

그리고 #body안에는 오늘의 날짜를 표현해 주는 #current-date와 페이지의 현재 위치를 나타내는 #location, 페이지의 타이틀을 나타내는 h1.content가 모든 페이지에 존재 하게 된다.

```
<div id="body-wrapper">
    <div id="body">
         <div id="current-date">2005.12.11 (일)</div>
         <div id="location">
              <a href="/">HOME</a> :
              <a href="/about/about_01.php">재경부안내</a> :
              <a href="/about/about_01.php"
class="current">인사말</a>
         </div>
         <h1 class="content"><img
src="/images/about/heading/greeting.gif" alt="인사말" /></h1>
    </div><!-- endof #body -->
</div><!-- endof #body-wrapper -->
```

#body-wrapper는 세로 구분선을 나타내는 이미지를 repeat-y로 넣는다. 이렇게 하면 실제적으로는 좌측 영역과 컨텐츠 영역을 모두 포함하면서 세로 구분선을 나타내게 된다.

```
#body-wrapper {
    background: url(/images/nav/bodyWrapper.gif) repeat-y 209px 0;
    width: 934px;
    margin: 0 0 10px;
}
```

그리고 그 위에 있는 #body에 drop shadow가 표현된 이미지를 넣고 여백을 조정한다.

```
#body {
    padding: 35px 0 68px 229px;
    min-height: 600px;
    background: url(/images/nav/news.gif) no-repeat;
}
* html #body {
    height: 600px;
}
```

여기서 주의 해야 할 점은 #body의 최소 높이를 설정하는 것이다. 좌측영역(#sub)을 absolute position을 사용해서 위치 시켰기 때문에 본문 영역이 좌측 영역보다 높이가 낮아 지게 되면 좌측 영역은 아래 부분이 하단의 푸터 영역을 덮게 된다. 그렇기 때문에 본문 영역을 좌측 영역보다 항상 높은 위치로 유지 시켜주는 것이 중요하다. 만약 정확한 본문 영역의 최소 높이를 판단 할 수 없을 때에는 javascript를 이용해서 좌측 영역의 높이가 본문 영역의 높이보다 높아지게 되는 경우 자동으로 본문 영역이 늘어날 수 있게 해주는 것이 좋다.

오늘 날짜와 현재 위치는 레이아웃에서 항상 같은 위치에 있기 때문에 absolute position을 이용하여 위치를 설정한다.

```
#current-date {
    position: absolute;
    top: 104px;
    left: 212px;
    width: 720px;
    text-align: right;
```

```
    font-weight: bold;
}
#location {
    position: absolute;
    top: 136px;
    left: 212px;
    width: 720px;
    text-align: right;
    line-height: 25px;
    font-size: 0.9em;
}
#location a.current {
    font-weight: bold;
}
```

본문의 타이틀은 높이와 여백, 선등을 설정해 주면 된다.

```
#body h1.content {
    height: 18px;
    font-size: 14px;
    line-height: 40px;
    margin: 0 0 12px;
    padding: 9px 0 9px 8px;
    border-bottom-width: 2px;
    border-bottom-style: solid;
}
```

하단 영역 (#foot)

하단 영역에는 프린트와 탑버튼이 있고 푸터용 로고, 기타링크, 사이트 주소 등이 있다.

```
<div id="foot">
    <div id="page-menu">
         <ul>
              <li><a href="#head"><img src="/images/nav/btnTop.gif"
alt="Top" /></a></li>
              <li><a id="page-print" href="#page-menu"><img
src="/images/nav/btnPrint.gif" alt="Print" /></a></li>
         </ul>
    </div>
    <h1><img src="/images/nav/footLogo.gif" alt="재정경제부" /></h1>
    <div id="foot-link">
         <ul>
              <li><a href="/about/about_07.php"><img
src="/images/nav/footLink01.gif" alt="찾아오시는길" /></a></li>
              <li><a href="/guide/privacy.php"><img
src="/images/nav/footLink03.gif" alt="개인정보보호정책" /></a></li>
              <li><a href="/guide/copyright.php"><img
src="/images/nav/footLink04.gif" alt="저작권정책" /></a></li>
              <li><a href="/guide/"><img
src="/images/nav/footLink05.gif" alt="이용안내" /></a></li>
              <li class="email"><a href="/guide/nospam.php"><img
src="/images/nav/footLink06.gif" alt="이메일 무단수집 거부" /></a></li>
              <li class="viewer"><a href="/guide/viewer.php"><img
src="/images/nav/footLink07.gif" alt="뷰어다운로드" /></a></li>
         </ul>
    </div>
    <div id="copyright"><img src="/images/nav/footCopyright.gif"
alt="Copyright &copy; 2005 by Ministry of Financial and Economy.
Republic of Korea. All Rights Reserved." /></div>
    <address><img src="/images/nav/footAddress.gif" alt="(우) 427-
725 경기도 과천시 관문로 88 번지 정부과천청사 재정경제부 | 대표전화 02-2110-
2332" /><br />
         <a href="mailto:forumnet@mofe.go.kr"><img
src="/images/nav/footEmail.gif" alt="Mailto:forumnet@mofe.go.kr"
/></a></address>
</div>
```

프린트 버튼과 탑버튼은 컨텐츠 영역의 하단에 위치하고 버튼 리스트 이기 때문에 #foot 맨 위의 list로 표현 된다. 그리고 푸터용 로고는 푸터에서 가장 중요한 제목 이므로 <h1>을 이용한다. 기타링크는 리스트 이므로 을 사용하고 사이트 주소는 <address>태그를 이용한다.

#foot은 #head와 마찬가지로 높이가 고정되어 있으므로 relative position을 사용하면 하위의 구성 요소들을 위치 잡기가 편리하다.

```
#foot {
    position: relative;
    clear: both;
    height: 88px;
    background: url(/images/nav/foot.gif) repeat-x;
    z-index: 0;
}
```

프린트 버튼과 탑버튼(#page-menu)은 푸터의 영역 밖에 있기 때문에 top offset을 음수로 지정해 준다. 그리고 다른 가로형 리스트들과 마찬가지로 float을 이용한다.

```
#page-menu {
    position: absolute;
    top: -41px;
    left: 852px;
}
#page-menu ul {
    padding: 0;
    margin: 0;
    list-style: none;
}
#page-menu li {
    float: left;
    margin-right: 2px;
}
```

푸터용 로고는 배경 이미지를 이용하기 위해서 #foot의 좌상단에 꽉 차게 위치를 정하고 #fff로 배경색을 지정하여 가로라인을 가린다. 그리고 안의 <img> 엘리먼트의 여백을 이용해서 위치를 잡아준다.

```
#foot h1 {
    position: absolute;
    top: 0;
    left: 0;
    margin: 0;
    padding: 14px 0 0;
    background: #fff;
    width: 209px;
}
#foot h1 img {
    margin-left: 32px;
}
```

기타 링크들은 다른 가로형 리스트와 같다.

```
#foot-link {
    padding: 13px 0 6px 209px;
}
#foot-link ul {
    padding: 0;
    margin: 0;
    list-style: none;
    height: 17px;
}
#foot-link li {
    float: left;
}
```

이메일 무단 수집 거부 버튼과 뷰어 다운로드 버튼은 그 위치가 상이 하기 때문에 absolute position을 이용해서 다시 위치를 잡아 준다.

```
#foot-link li.email {
    position: absolute;
    top: 22px;
    left: 840px;
}
#foot-link li.viewer {
    position: absolute;
    top: 45px;
    left: 840px;
}
```

나머지 카피라이트 부분과 주소 부분은 여백을 설정해 주는 것으로 간단하게 마무리 할 수 있다.

```
#copyright,
#foot address {
    margin-left: 209px;
}
#copyright img,
#foot address img {
    margin-bottom: 3px;
}
```

완료

이렇게 현재 구현되어 있는 페이지의 스타일을 구간별로 짚어 보면서 제작되어있는 코드를 살펴 보았다. 사실 CSS디자인이 그 개념을 이해하기 까지는 낯설고 어렵게 느껴질 수도 있지만 실상 작업을 계속 하다 보면 굉장히 단순하고 반복적인 작업이 많은 것이 사실이다. 오히려 CSS보다는 XHTML을 구조적으로 작성하고 id, class의 이름을 알아보기 쉽게 정하는 것이 더 어려운 작업이다. 사실 CSS 레이아웃이나 CSS를 이용해서 디자인을 적용하는 목적 자체가 의미에 맞는 XHTML 제작이라는 것을 생각해 보면 당연한 결과이기도 하다. 항상 XHTML에 초점을 맞추고 완성도 높은 마크업을 제작할 수 있다면 CSS를 이용해서 디자인을 적용하는 일은 크게 어렵지 않을 것이다.

고급 CSS 레이 아웃

CSS를 이용한 디자인 팁

CSS를 이용하면 기존에 우리가 하기 어려웠던 다양한 디자인 효과를 얻을 수 있다. 이 중 몇 가지를 소개해 보고자 한다.

라운드형 박스 디자인

CSS를 이용하여 테이블을 통해 여러 개 파일로 구성해야 하는 디자인도 쉽게 구현이 가능하다 예를 들어 아래 그림과 같이 제목과 내용이 있는 박스를 디자인 하려면 어떻게 해야 할까? 대부분 테이블을 떠 올릴 것이다.

그림 25 라운드 박스 표현을 위한 배경 분리

그러나 라는 박스 안에 제목 부분을 <h3>, 목록 부분을 로 나누어 우선 시맨틱 마크업을 해보자.

```
<div class="box">
<h3>Gifts and Special Offers</h3>
<ul>
<li><a href="/purchase/">Purchase Gift Subscription</a></li>
<li><a href="/redeem/">Redeem Gift Subscription</a></li>
<li><a href="/view/">View Purchase History</a></li>
</ul>
</div>
```

이렇게 한 다음 각각 윗 배경과 아랫 배경 그림을 <h3>과 속성에 스타일을 지정하고 표현 하면 테이블로 복잡하게 작성하는 라운드형 박스를 제작할 수 있다.

```
<style>
.box {
  width: 273px;
  background: url(img/div-bottom.gif) no-repeat bottom left;
```

```
  }
.box h3 {
  margin: 0;
  padding: 6px 8px 4px 10px;
  font-size: 130%;
  color: #333;
  border-bottom: 1px solid #E0CFAB;
  background: url(img/h3-bg.gif) no-repeat top left;
  }
.box ul {
  margin: 0;
  padding: 14px 10px 14px 10px;
  list-style: none;
  }
.box li {
  margin: 0 0 6px;
  padding: 0;
  }
</style>
```

이미지의 갯수 줄이기

빠른 로딩, 사용자 친화적인 랜더링)

상속을 고려한 CSS 구조 설계

CSS에서 선언된 속성들은 상속이 가능하다. 즉, 초기에 선언된 내용을 기초로 다양한 선언들로 재사용 가능한 것이다. 웹 사이트의 메뉴에 따라서 색상을 변경하는 기능을 구현해 보고자 한다. 그러면, 우선 사이트 기본 CSS 파일 'default.css'를 작성한다. 여기에는 레이아웃과 글자 크기에 대한 속성만 정의 한다.

그런 다음 색상 CSS 파일을 'color.css'라고 하고 여기에 색상을 정의 한다.

Default.css

```
h1 {
font-size: 1.4em;
font-weight: bold;
}
```

Color.css

```
h1 {
background-color: #f30;
color: #fff;
}
```

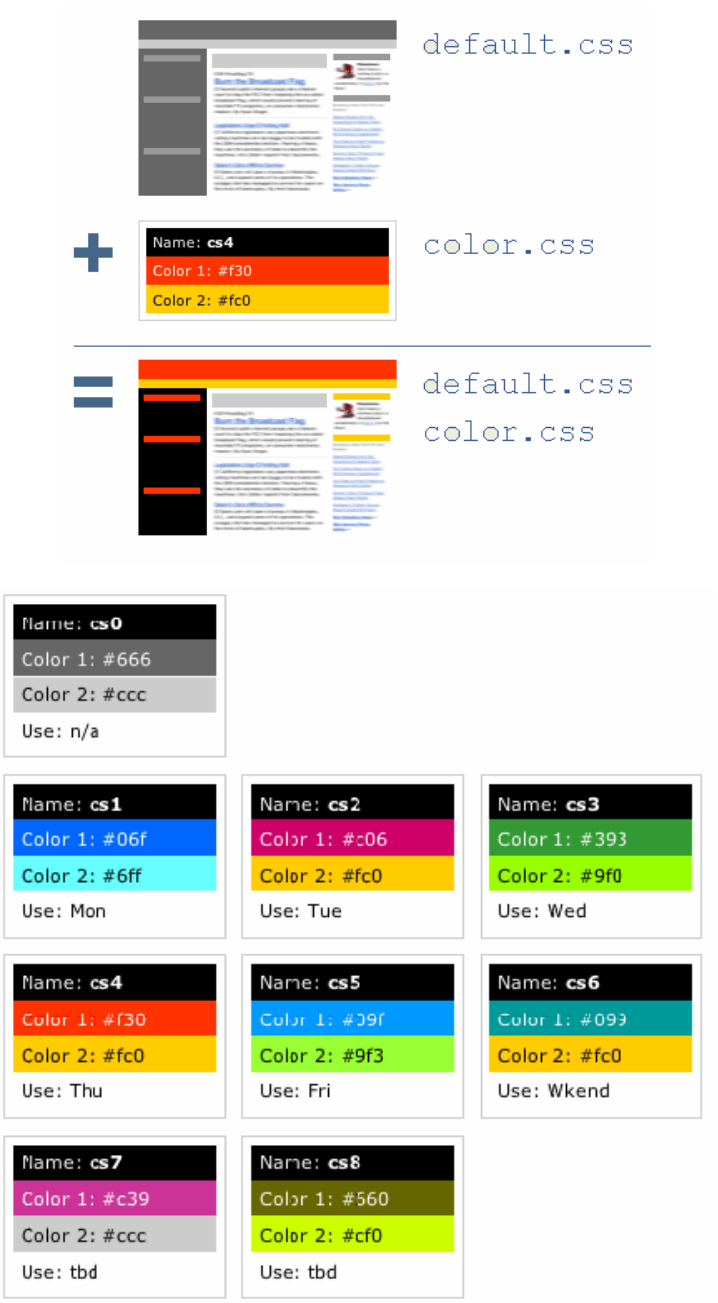


그림 26 Wired.com을 통해 본 CSS 파일 상속 사례

CSS Switching으로 다양한 페이지 만들기

웹 사이트에서 요구 사항에 보면 하나의 컨텐츠인데도 불구하고 다양한 계층과 단말기를 위한 웹 페이지를 요구하는 경우가 있다. 예를 들어, 유아나 노인을 위해 확대 축소 기능이 가능한 텍스트 전용 페이지라던지 장애인을 위한 음성 서비스가 포함된 웹페이지, 혹은 모바일 단말기를 위한 모바일 페이지 같은 것들이다.

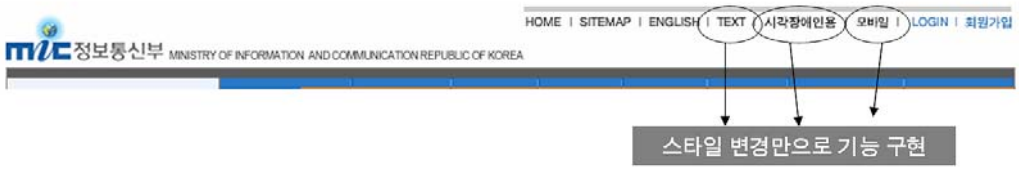


그림 27 정보통신부의 텍스트, 시각 장애인, 모바일 페이지

지금까지는 하나의 웹페이지를 만든 다음, 디자인을 변경하여 Copy&Paste 방식의 웹페이지 과잉 생산을 해왔었다. 이들의 내용은 모두 같아야 하는데도 중복 페이지가 생성되어 관리에 어려움이 있었던 것이 사실이다.

만약 하나의 웹 페이지에 다양한 스타일을 주고 쉽게 변경할 수 있으면 하나의 웹페이지에서 다양한 요구를 반영할 수 있다. 다국어 홈페이지처럼 내용이 바뀌는 것이 아니라 단지 디자인이 변경 된다면 더더욱 그렇다.

우선 홈페이지에 다양한 디자인 양식을 아래와 같이 넣는다.

```
<link rel="stylesheet" type="text/css" href="원본.css"
title="default" />
<link rel="alternate stylesheet" type="text/css" href="텍스트.css"
title="Text" />
<link rel="alternate stylesheet" type="text/css" href="장애인.css"
title="Accessiblity" />
<link rel="alternate stylesheet" type="text/css" href="인쇄용.css"
title="Print" />
<link rel="alternate stylesheet" type="text/css" href="모바일.css"
title="Mobile" />
```

만약 텍스트 페이지인 경우 레이 아웃은 같지만 이미지가 없는 경우가 된다. 따라서 텍스트 CSS는 아래와 같은 태그를 넣어 이미지를 없앨 수 있다.

```
img { display: none; }
```

그런 다음 자바 스크립트를 이용하여 CSS 자동 변경을 해보고자 한다.

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

이렇게 한 다음 페이지 내에 직접 사용자가 클릭할수 있는 링크를 넣고 싶은곳에 레이아웃 변경 페이지를 넣으면 된다.

```
<a href="#" onclick="setActiveStyleSheet('Text'); return
false;">텍스트용</a>
<a href="#" onclick="setActiveStyleSheet('Accessiblity'); return
false;">장애인용</a>
<a href="#" onclick="setActiveStyleSheet('Print'); return
false;">인쇄용</a>
<a href="#" onclick="setActiveStyleSheet('Mobile'); return
false;">모바일용</a>
```

Styleswitcher.js 의 내용은 아래와 같다.

```
function setActiveStyleSheet(title) {
  var i, a, main;
  for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
```

```
    if(a.getAttribute("rel").indexOf("style") != -1 &&
a.getAttribute("title")) {
      a.disabled = true;
      if(a.getAttribute("title") == title) a.disabled = false;
    }
  }
}

function getActiveStyleSheet() {
  var i, a;
  for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
    if(a.getAttribute("rel").indexOf("style") != -1 &&
a.getAttribute("title") && !a.disabled) return
a.getAttribute("title");
  }
  return null;
}

function getPreferredStyleSheet() {
  var i, a;
  for(i=0; (a = document.getElementsByTagName("link")[i]); i++) {
    if(a.getAttribute("rel").indexOf("style") != -1
       && a.getAttribute("rel").indexOf("alt") == -1
       && a.getAttribute("title")
       ) return a.getAttribute("title");
  }
  return null;
}

function createCookie(name,value,days) {
  if (days) {
    var date = new Date();
    date.setTime(date.getTime()+(days*24*60*60*1000));
    var expires = "; expires="+date.toGMTString();
  }
  else expires = "";
  document.cookie = name+"="+value+expires+"; path=/";
}

function readCookie(name) {
  var nameEQ = name + "=";
  var ca = document.cookie.split(';');
  for(var i=0;i < ca.length;i++) {
    var c = ca[i];
    while (c.charAt(0)==' ') c = c.substring(1,c.length);
    if (c.indexOf(nameEQ) == 0) return
c.substring(nameEQ.length,c.length);
  }
  return null;
}

window.onload = function(e) {
```

```
  var cookie = readCookie("style");
  var title = cookie ? cookie : getPreferredStyleSheet();
  setActiveStyleSheet(title);
}
window.onunload = function(e) {
  var title = getActiveStyleSheet();
  createCookie("style", title, 365);
}

var cookie = readCookie("style");
var title = cookie ? cookie : getPreferredStyleSheet();
setActiveStyleSheet(title);
```

동적인 메뉴 레이아웃 구성

CSS 스타일 변경 만으로 레이아웃을 구성하는 각종 항목의 위치를 자유 자재로 변경할 수 있다. 여기서는 가장 알맞은 예제 사이트를 소개한다.

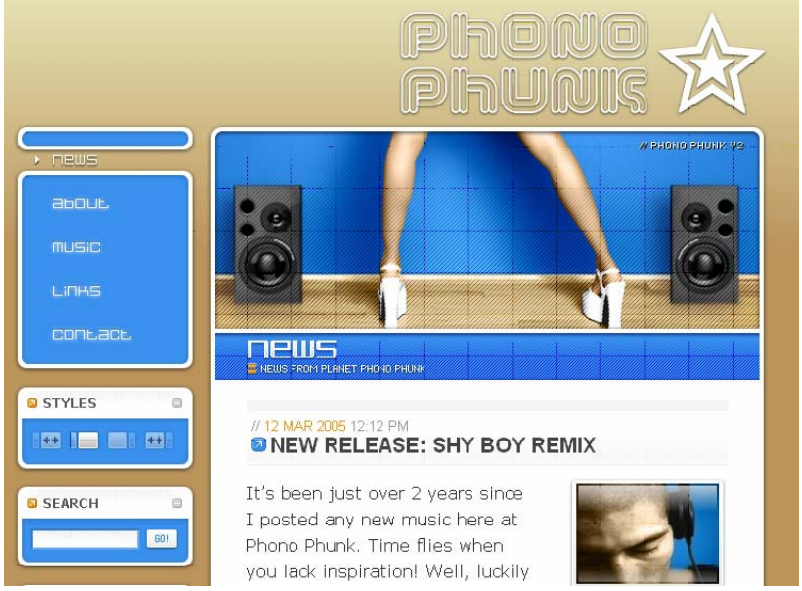


그림 28 스타일 변경으로 레이아웃 변경 사례 (http://PhonoPhunk.phreakin.com)

CSS 개발 및 검증 도구

많은 웹 개발자들이 XHTML/CSS 개발을 메모장에서 하고 있는 경우가 많다. 왜냐하면, 나모 웹에디터나 드림위버 같은 웹 에디터들이 원하지 않는 코드를 생산해 내는 경험 때문이다. 이로 인해, 코드 생산성이 더 떨어진다고 생각하게 된다. 그러나 아래에 소개할 웹 에디터들은 표준 마크업 및 CSS 레이아웃을 지원하는 것들 이므로 이것들을 이용하면 보다 효율적인 표준 CSS 개발이 가능할 것이다.

또한, CSS 문법을 제대로 사용 했는지 개발 시 확인하고 추후 QA 과정에서 검증할 수 있는 다양한 툴도 소개한다.

표준 에디터 소개

드림 위버 8 및 MX2004

드림위버는 대표적인 웹 에디터이다. 드림위버 8 이상 MX2004 버전에서는 표준CSS를 을 거의 완벽하게 지원한다.

또한 간단한 드래그 앤 드롭 워크플로를 통해 RSS 피드와 같은 XML 기반의 데이터를 웹 페이지에 통합할 수 있을 뿐만 아니라, 코드 뷰로 바로 이동하여 XML 및 XSLT를 지원하는 향상된 코드 힌트 기능을 사용하여 사용자 요구에 맞게 변형 작업을 수행할 수 있다.

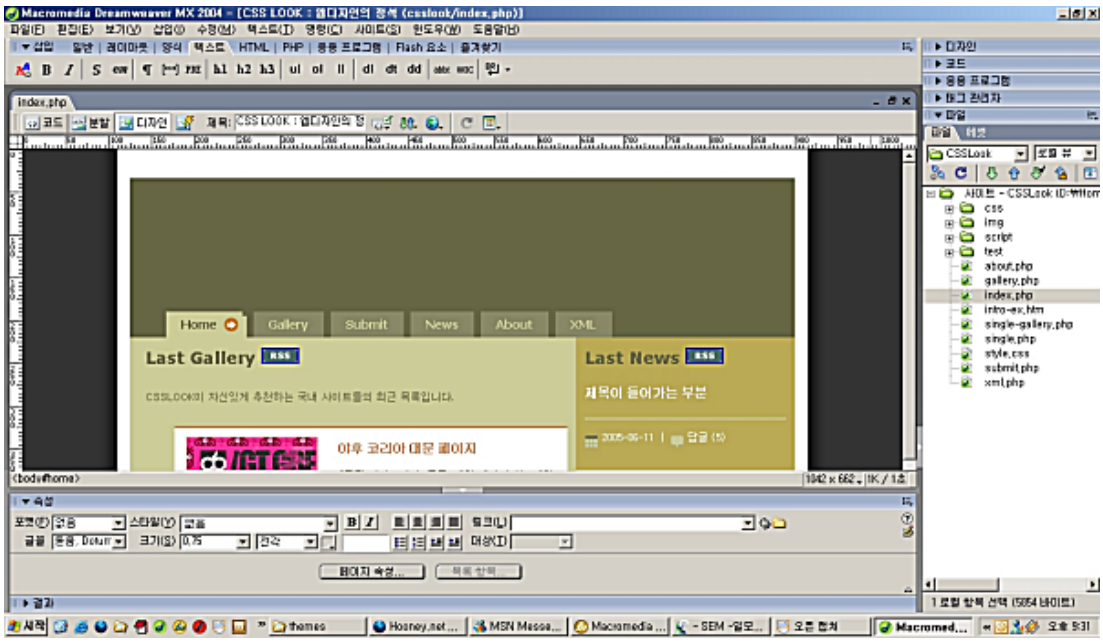


그림 29 드림위버 MX 2004를 통해 본 CSS 레이아웃 기능

모든 CSS 기능이 하나의 패널 세트로 통합되어 CSS를 사용한 작업이 훨씬 수월해졌으며 생산성이 크게 향상되었다. 이 새로운 인터페이스를 사용하면 특정 요소에 적용된 계단식 배열의 스타일을 손쉽게 볼 수 있고 속성이 정의된 위치를 쉽게 파악할 수 있다. 또한 속성 그리드를 통해 신속하게 편집할 수 있다.

디자인 시 시각적인 보조 도구를 사용하여 CSS 레이아웃 테두리의 외곽선을 표시하거나 CSS 레이아웃의 색상을 지정하여 복잡하게 중첩된 구조를 구분할 수 있고 손쉽게 항목을 선택할 수 있다. CSS 레이아웃을 클릭하면 ID, 패딩, 여백, 테두리 설정과 같은 유용한 툴팁을 확인할 수 있다.

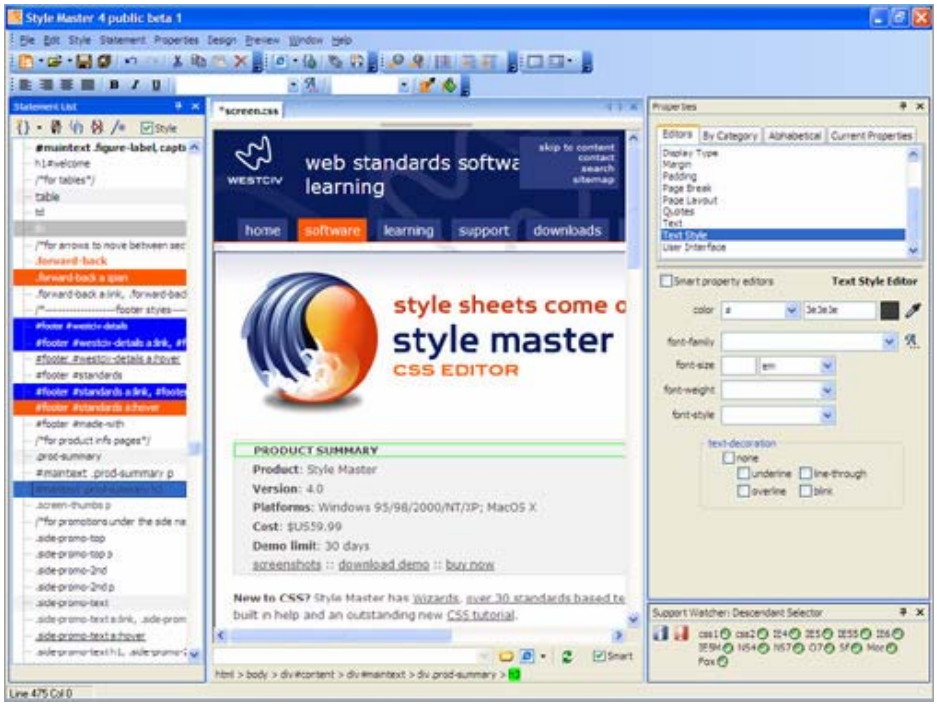
드림위버는 CSS 미디어 유형을 새롭게 지원하므로 컨텐츠 전달 방법에 관계없이 최종 사용자가 보게 될 컨텐츠와 동일한 컨텐츠를 볼 수 있다. 또한 스타일 렌더링 툴바를 통해 디자인 뷰로 전환하여 핸드헬드 또는 스크린에서의 인쇄 시 모습이 어떤지 확인할 수 있다.

정확성이 대폭 향상된 디자인 뷰를 통해 아무리 복잡한 CSS 레이아웃도 대부분의 브라우저에서 렌더링되도록 할 수 있다. Dreamweaver는 이제 오버플로, 의사(pseudo)-요소 및

양식 요소와 같은 고급 CSS 기법을 완벽하게 지원한다.

Section 508과 WCAG Priority 1 체크포인트를 위한 통합된 액세스 가능성 평가 도구 이외에도 Dreamweaver는 현재 WCAG Priority 2 체크포인트를 포함하는 업데이트된 평가 도구를 통해 CSS 및 액세스 가능성을 모두 지원합니다.

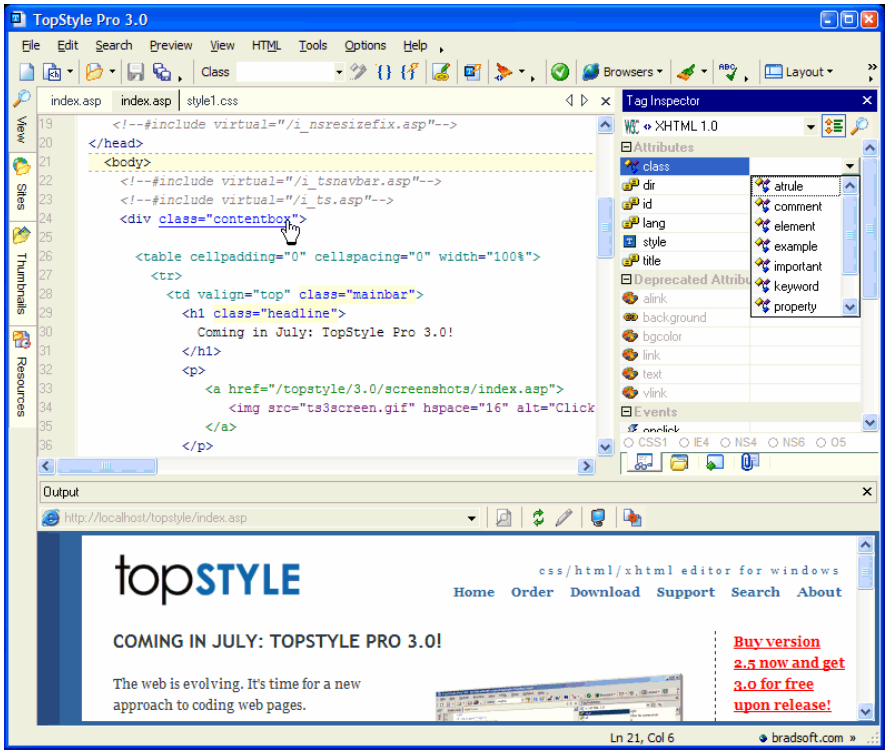
Style Master



http://www.westciv.com/ 에서 다운 받아 사용할 수 있는 CSS Style Master는 매우 훌륭한 CSS 에디터이다.

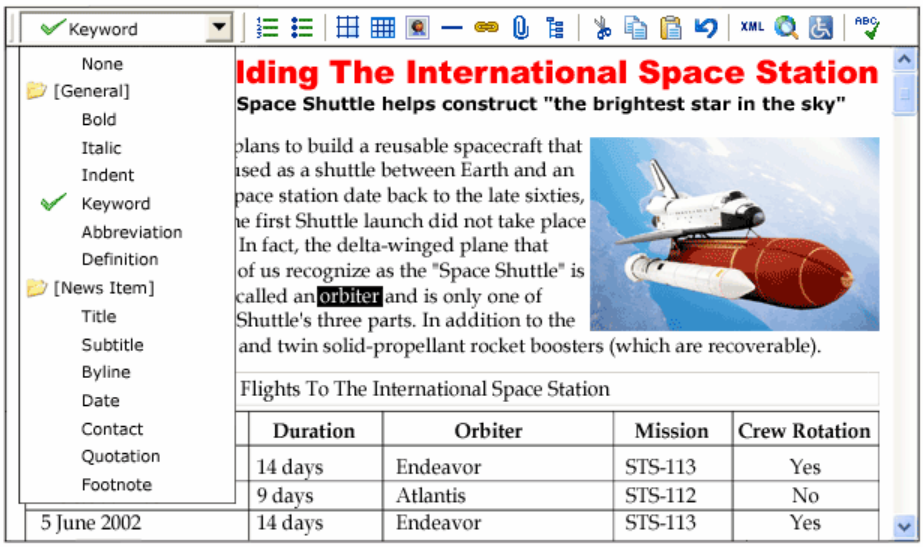
Top Style Pro

Top Style Pro는 메모장에 길들어져 있는 개발자에게 제격이다. 각종 CSS 표준문법들을 코딩 즉시 알 수 있다. http://www.bradsoft.com 에서 다운로드 받을 수 있다.



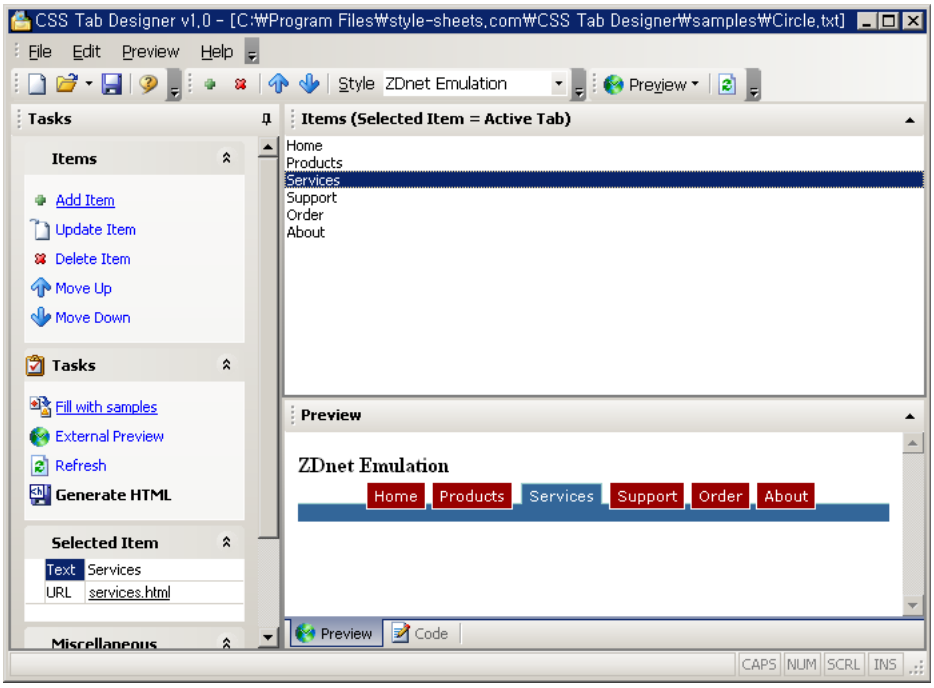
XStandard Editor

XStandard Editor는 XML 에디터이다. XHTML 및 CSS도 지원하며 웹 에디터로 사용할 수도 있다. http://www.xstandard.com 에서 다운받을 수 있다.



CSS Tab Designer

CSS Tab Designer는 태그를 이용하여 탭 메뉴를 자동으로 만들어 주는 간단한 프로그램이다. CSS로 다양한 효과를 얻을 수 있는 여러 가지 데모들을 볼 수 있다. http://www.style-sheets.com 에서 다운로드 받을 수 있다.



유효성 검증 도구

W3C CSS Validator

W3C의 CSS Validator는 가장 많이 사용하는 CSS 문법 검증 도구 이다. 이를 통해 CSS 오류 등을 알 수 있다. (http://jigsaw.w3.org/css-validator)

Web Developer 확장 기능

파이어폭스에는 Web Developer 확장 기능이라는 것이 있다. 여기서 CSS 문법 오류를 간단하게 체크해 볼 수 있다.

- 다운로드: http://chrispederick.com/work/webdeveloper/

Web Accessiblity 툴바

웹 접근성 툴바를 통해서도 CSS 문법 오류를 체크해 볼 수 있다.

- 다운로드: http://www.vinfoaxia.com/tools/wat/ko

HTML Tidy

HTML Tidy(http://tidy.sourceforge.net)를 이용하는 방법도 있다. HTML Tidy는 HTML의 문법을 체크하고 잘못된 문법에 대해서는 수정도 가능하게 하는 HTML문법을 위한 도구 이다. 현재는 오픈소스로 관리가 이루어지고 있어서 많은 수의 참가자들이 공동으로 제작에 참여하고 있다.

이 HTML Tidy를 직접 설치하여 이용할 수도 있겠지만 조금은 번거롭고 누구나 쉽게 할 수 있는 방법은 아니다. 하지만 이 HTML Tidy가 Firefox 확장기능으로 제공되고 있어서 이를 이용하면 손쉽게 HTML, XHTML문법의 유효성을 체크해 볼 수 있다.

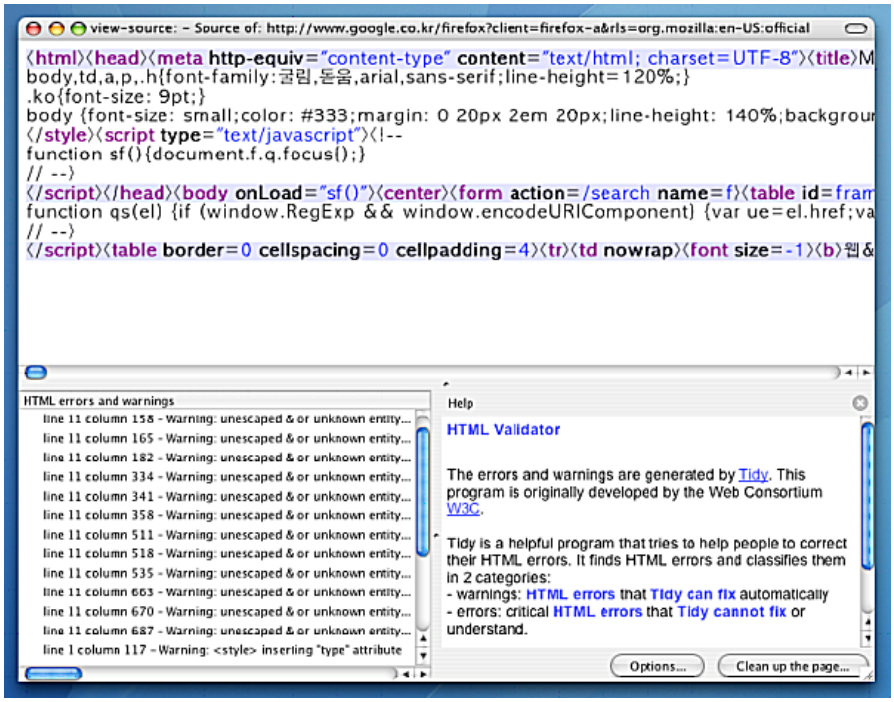


그림 30 HTML Tidy를 통한 유효성 검사

브라우저의 우측 하단에서 validation 결과를 바로 확인 할 수 있고, 소스보기에서 해당하는 부분과 에러 내용을 살펴볼 수 있다.

실전 DOM/Script 가이드

표준 DOM 기반 개발

문서 객체 모델(DOM; Document Object Model)은 HTML내에 들어 있는 요소를 구조화 객체 모델로 표현하는 형식이다. DOM은 플랫폼/언어 중립적으로 구조화된 문서를 표현하는 W3C 표준 모델이 기반이 된다.

DOM은 HTML 문서의 요소(Element)를 조작하기 위해 웹 브라우저에서 처음 지원됐다. DOM은 동적으로 문서의 내용, 구조, 스타일에 접근하고 변경하는 수단이었다. 브라우저 간의 DOM 구현이 호환되지 않음에 따라 W3C에서 DOM 표준 명세를 작성하게 되었다.

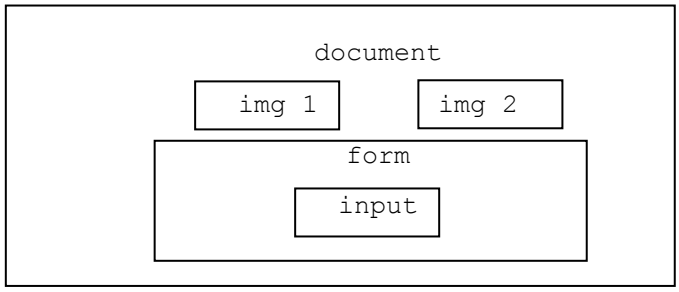
DOM은 문서의 기반이 되는 데이터 구조에 제한을 두지 않는다. 잘 구조화된 문서는 DOM을 사용하여 트리 구조(Tree Structure)를 얻어낼 수 있다. 대부분의 XML 해석기와 XSL 처리기는 이러한 트리 구조를 기반으로 개발 되었는데, 이 같은 구현에서는 문서의 전체 내용이 해석되어 메모리 저장되어야 한다. 이 때문에 DOM은 문서 요소가 임의적으로 접근되고 변경 가능해야 하는 응용프로그램에 가장 적합하다. 한 번 해석 시 단 한 번의 선택적인 읽기 및 쓰기가 이루어지는 XML 기반 응용프로그램에서 DOM은 메모리에 상당한 부하를 가져 오기도 한다.

현재 널리 사용되고 있는 DOM 표준안은 Level 2이다. 일부 Level 3 명세서 역시 W3C의 권고안으로 나와 있다.

- Level 0: DOM이 만들어지기 이전의 모든 벤더 종속적인 DOM을 포함한다. 예: document.images, document.forms, document.layers, document.all. 이것은 W3C에 의해 공식적으로 출판된 명세가 아니며, 표준화 과정 이전에 있었던 단계에 대한 표현이다.
- Level 1: DOM 문서에 대한 탐색과 조정에 대한 최초의 표준 명세
- Level 2: XML 네임스페이스(Namespace) 지원, 필터링된 뷰(view)와 이벤트.
- Level 3: 6가지 다른 명세로 구성: 1) Core; 2) Load and Save; 3) XPath; 4) Views and Formatting; 5) Requirements; 6) Validation;

W3C DOM vs. MS DOM

DOM이 처음 구현 된 것은 최초의 브라우저인 Netscape 2 에서이다. 이 때는 문서 내에 들어있는 태그를 그대로 접근하는 방식이었다. 예를 들어 아래와 같은 문서 구조가 있다고 하자.



그러면 위의 각 문서 내 객체들은 다음과 같이 접근한다.

```
document.images['thefirst'] // name 이 있는 경우
document.images[0]
document.images[1]
document.forms['contactform'] //name 이 있는 경우
document.forms[0].elements['address']
document.forms[0].elements[0]
```

그러나 이렇게 접근할 수 있는 HTML 태그들은 제한 되어 있다. 예를 들어 <h1>, <p> 등은 접근할 수 없는 것이다. Netscape 4와 IE4 버전이 나올 무렵 좀 더 근본적인 방법으로 문서 객체에 접근할 수 있는 방법을 따로 내놓았다. 그것이 바로 document.layer와 document.all 이다. 예를 들어, id를 기초로 각 객체를 부르는 방법을 다음과 같이 다르게 사용하게 된 것이다.

```
<DIV ID="stuff"><IMG NAME="testimage"></DIV>
document.all['stuff'].style.left = 200;  // IE 표현
document.all.stuff.style.left = 200; // IE 표현
document.layers['stuff'].left = 200; // Netscape 표현
document.stuff.left = 200; // Netscape 표현
```

이렇게 자신들이 원하는 형태로 객체 모델을 만들어 퍼트리게 되니 1990년대 중반에는 이른바 DHTML이라는 기법으로 홈페이지를 만들려면 고도의 브라우저 구현 스펙을 알아야 하는 문제까지 생겼다. 따라서, 브라우저 전쟁 기간 동안 마이크로소프트와 넷스케이프 양쪽 다 표준 경쟁의 와중에서 비표준 기능을 퍼트리게 된 것에 대한 책임을 피할 수 없다.

어쨌든 브라우저간의 DOM 구현의 차이 때문에 상호 운용성 문제가 생기게 되었다. 이 때문에 MS와 넷스케이프는 DOM Level 1 표준 스펙을 작성하게 되었고, IE5.0 에서는 W3C DOM을 지원하기 시작했다. 그러나, IE가 브라우저 시장에서 독점이 되면서 IE가 W3C DOM을 지원함에도 불구하고 MS DOM이 일반적으로 쓰이게 되었다. 그럼으로 인해 최근에 나온 파이어폭스나 오페라 등과 같은 표준 호환 브라우저에서 비 표준 DOM 사용에 따라 웹 페이지가 제대로 표시 되지 못하는 문제가 발생하고 있다.

그러나 위와 같은 브라우저 전용 DOM 접근 방식은 W3C DOM에서 아래와 같이 정의한다. 이는 이미 IE 5.0 이후에 나온 모든 브라우저가 지원하므로 거의 99%의 브라우저가 지원한다고 볼 수 있다.

```
document.getElementById['stuff'].style.left = 200; // W3C 표준 표현
```

따라서 일부 몇 가지 DOM 스펙만을 제외하면, 표준 W3C DOM을 사용하는 것이 중요하다. 만약 웹 개발자가 IE의 MS DOM 확장을 사용한다면 표준 준수에 대한 신뢰성을 잃을 수 있으며, 반대의 경우라면 비표준 확장을 사용하지 않음으로 생기는 기능적 제약 때문에 사용자가 이탈할 수도 있다. 그러나 표준 호환 브라우저가 웹 시장에서 주목할 만한 점유율을 차지하게 된다면 이 같은 상황이 바뀌게 될 것이며, 비표준 확장을 사용하는 것이 작성자에게 상업적 불이익으로 다가올 것이라는 것에 대해서는 일반적으로 의견이 일치하고 있다.

IE도 W3C DOM을 지원하고 있기 때문에 MS DOM을 사용하지 않고 대체 기능을 사용하면 충분히 표준 호환 브라우저에서도 서비스를 제공할 수 있다.

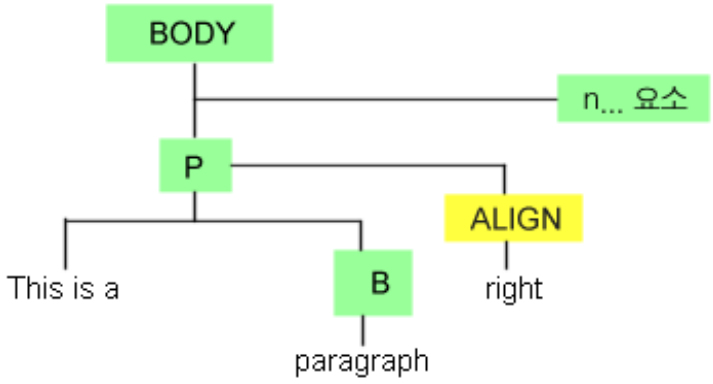
DOM 기본 기능

W3C DOM에서는 HTML 문서를 XML로 바라본다. 즉, 각 문서에 있는 태그들을 노드(Node)가 있는 트리 구조로 해석해 낼 수 있다고 가정하는 것이다. 각 태그들은 요소 노드(Element Node)와 텍스트 노드(Text Node)가 있다고 생각하며, 각 요소 노드에 있는 속성 값들은 속성 노드(Atrribute Node)로 인식한다.

예를 들어 아래와 같은 문서 구조가 있다고 하자.

```
<BODY>
<P ALIGN="right">This is a <B id="dynatext">paragraph</B></P>
</BODY>
```

위의 문서는 아래와 같은 구조로 표현된다.



객체 접근 방법

Level 0이라고 명명한 과거의 DOM에서 객체에 접근하는 방식은 document.form 과 같이 HTML 글로벌 네임 스페이스를 바로 선언 하는 것이었다. 그렇지 않으면 document.layer (NS4), document.all(IE4)를 통하는 것이었다. Level 1에서는 이를 위해 다음 두 가지의 객체 접근 방식을 정했다.

- document.getElementById(aId)
- document.getElementByTagName(aTagName)

따라서 이전에 썼던 모든 객체 접근법은 위의 방식으로 바꾸어야 한다. 만약 getElementById를 지원하지 않는 IE4의 경우, 아래의 객체 접근 함수를 만들어 사용하면 유용하다.

```
function getObject(objectId) {
if(document.getElementById && document.getElementById(objectId) {
return document.getElementById(objectId); // check W3C DOM
}
else if (document.all && document.all(objectID) {
   return document.all(objectID); // IE4
}
else if (document.layers && document.layers[objectID] {
```

```
   return document.layer[objectID]; // NN4
}
else {
   return false;
   }
}
```

위의 스크립트를 통해 getObject(objectId)를 이용하여 모든 브라우저의 DOM 객체를 얻을 수 있다.

객체 사용 방법

DOM 트리 구조 중 각 요소의 컨텐트는 일련의 자식 노드(child node)로 분할되어 있으며, 각 노드는 단문과 그 자식 요소로 구성되어 있다. 즉, 텍스트를 변경하고자 요소의 노드를 조정하는 것이 표준적인 방법이다. 노드의 구조 및 지원 메소드는 W3C DOM 레벨 1 권고로 정해져 있다. 이 메소드들은 이미 IE5.0 이상 브라우저에서 모두 지원한다. 따라서 문제 없이 사용할 수 있다.

예를 들기 앞서서, 우리가 흔히 사용하는 요소 사용 방법중 중에 구성요소의 내용을 바꾸거나 수정하는 innerText, innerHTML, outerText, outerHTML을 사용하는 것은 원래 잘못된 것이다. 왜냐하면 이는 W3C DOM 표준이 아니고 MS DOM 이기 때문이다. 그러나, 많은 웹 브라우저들이 만은 지원해 왔기 때문에 일반적으로 사용할 수 있다. 모질라 계열에서는 innerHTML 외에 outerHTML, innerText와 outerText등 다른 메소드는 지원하지 않기 때문에 사용하는 것을 지양해야 한다. W3C DOM을 이용해서도 이 기능을 구현 할 수 있는 방법이 있다.

innerText와 innerHTML을 W3C DOM으로 구현하는 방법을 통해 객체 사용 방법을 알아 보도록 하자.

```
<p ALIGN="right">This is a <span id="dynatext">paragraph</span></p>
```

위의 dynatext라는 id에 텍스트 내용을 변경하는 예를 통해 DOM에 접근하는 방법을 알아 보자. 먼저 id dynatext의 요소를 먼저 span_el에 담는다.

```
<script type="text/javascript">
var span_el = document.getElementById("dynatext");
```

자식 노드에서 그 이하 특정한 요소를 가지지 않고 텍스트 밖에 없다면 (통상) element.childNodes[0] 으로서 접근할 수 있는 1차 노드를 가진다. 즉, 우리가 element.innerText을 대체할 방법으로서는 element.childNodes[0].nodeValue 가 사용할 수 있다.

즉, span_el.innerText = "a brand new bag"을 실현하려면, 다음과 같다.

```
   var new_txt = document.createTextNode("a brand new bag");
   span_el.replaceChild(new_txt, span_el.childNodes[0]);
```

span_el.innerHTML = "a brand new bag" 처럼 HTML 태그가 포함된 문장을 표준적으로 수용하는 방법은 새로운 요소를 만든 후, 이에 3개의 노드를 추가한다. 처음의 텍스트 노드, 자신의 텍스트 노드를 가지는 B 요소, 마지막 텍스트 노드에 각각 내용을 넣

은 다음 원래 요소에 치환하는 방법이다

```
var new_el = document.createElement(span_el.nodeName);
   new_el.appendChild(document.createTextNode("a brand "));
   var bold_el = document.createElement("B");
   bold_el.appendChild(document.createTextNode("new"));
   new_el.appendChild(bold_el);
   new_el.appendChild(document.createTextNode(" bag"));
   span_el.parentNode.replaceChild(new_el, span_el);
 </script>
```

DOM객체와 노드 및 그 속성을 조작 하는 방법은 일반적인 W3C DOM 표준을 따른다. createElement(), createTextNode() 같은 노드 생성 및 접근, nodeName, nodeType, tagName 같은 노드 정보, childNode, firstChild같은 트리 구조 파악 등은 비교적 표준이 잘 지켜 지고 있다.

그러나, getAttribute(), removeAttribute() 같은 속성 정의 인터페이스들은 브라우저 마다 다른 점이 매우 많아서 사용에 주의를 요한다. 이러한 차이점은 부록에 나와 있는 DOM 브라우저 호환 차트를 참고해야 한다.

일반적인 중요 DOM 메소드에 대해서 아래와 같다.

Property/Method	Description	비고
childNodes	요소내 모든 노드의 배열을 반환	
firstChild	요소내 첫 번째 노드를 반환	
getAttribute(aAttributeName)	특정 속성 값을 반환	오페라 버그
hasAttribute(aAttributeName)	특정 속성 값이 있는지 여부를 판별.	IE지원 안함
hasChildNodes()	자식 노드가 있는지 여부를 판별	
lastChild	요소내 마지막 노드를 반환	
nextSibling	상위 노드의 다음 자식 노드	
nodeName	현재 노드의 이름을 반환	
nodeType	현재 노드의 형식을 반환한다. 예를 들어 1번은 요소 노드, 2번은 속성 노드 3번은 텍스트, 4번은 CDATA, 5번은 참조 엔티티 등.	
nodeValue	현재 노드의 값을 반환한다. 노드 값이 텍스트이면 텍스트를, 속성이면 속성값을 기타는 null을 반환한다.	
ownerDocument	현재 노드를 포함하고 있는 문서 객체를 반환	
parentNode	현재 노드의 상위 노드를 반환	
previousSibling	상위 노드의 이전 자식 노드	
removeAttribute(aName)	노드의 특정 속성을 지운다.	IE 버그 있음

setAttribute(aName, aValue)	특정 노드의 특정 속성에 대한 값을 설정한다.	IE 버그 있음

속성을 읽어 오는 가장 좋은 방법을 예를 들어 보자.

```
id="test" align="center" style="border: 1px solid #0000cc"
```

1. x.id 나 x.style 같은 방법으로 먼저 속성을 찾는다.
2. 값이 나오지 않으면 x.getAttribute("align")나 x.getAttributeNode("align").value 로 찾는다.
3. 그래도 나오지 않으면 다른 속성 인터페이스를 시도하되, attributes[]는 절대 사용하지 않는다.

객체 요소를 다루는 메소드에 있어서도 IE에서만 사용되는 MS DOM 확장 메소드들이 있다. 이들에 대해서도 사용에 주의를 요하며 이를 대체할 수 있는 W3C DOM을 사용한다.

MS DOM 전용 확장	설명	W3C 대체 표준
applyElement()	다른 상위 노드에 새 노드 생성	appendChild()
clearAttributes()	노드의 모든 속성 삭제	removeAttribute()
mergeAttributes()	특정 속성을 특정 노드에 복사	cloneNode()
removeNode()	노드 삭제	removeChild()
replaceNode()	특정 노드를 다른 노드로 대체	replaceChild()
swapNode()	두 노드를 바꾸기	

DOM 호환 기능

DOM Level1과 2에서는 거의 모든 HTML/CSS 객체의 속성을 이용할 수 있다. x를 객체라고 가정하면 x.className이나 x.id, x.title 등으로 속성을 읽고 쓸 수 있다. 또한, x.style 등으로 스타일 설정도 가능하다. 그러나, 아직 브라우저 비 호환 DOM 속성들이 많이 남아 있어서 아래 몇 가지 예에서는 브라우저 판별법을 사용해야 될 필요가 있다. 아래 사항을 알아 두면 웹 개발 시 도움이 될 것이다.

윈도우 위치 파악

DOM을 사용하다 보면 윈도우 위에 각종 DOM 레이어들을 배치 시키고 위치를 잡도록 해야될 필요가 있다. 각 브라우저 마다 윈도우의 크기와 높이 위치를 정하는 방식이 다르기 때문에 이에 대한 호환 방식을 알아둘 필요가 있다.

Inner width 알기

윈도우나 프레임의 내부 크기를 알아내는 방법이다.

```
var x,y;
if (self.innerHeight) { // IE 외 모든 브라우저
   x = self.innerWidth;
   y = self.innerHeight;
```

```
}
else if (document.documentElement &&
document.documentElement.clientHeight) { // Explorer 6 Strict 모드
          x = document.documentElement.clientWidth;
   y = document.documentElement.clientHeight;
}
else if (document.body) { // 다른 IE 브라우저
   x = document.body.clientWidth;
   y = document.body.clientHeight;
}
```

스크롤 위치 파악

페이지가 얼마나 스크롤 됐는지 알아내는 방법이다.

```
var x,y;
if (self.pageYOffset) { // IE 외 모든 브라우저
   x = self.pageXOffset;
   y = self.pageYOffset;
}
else if (document.documentElement &&
document.documentElement.scrollTop) {
   // Explorer 6 Strict
   x = document.documentElement.scrollLeft;
   y = document.documentElement.scrollTop;
}
else if (document.body) { // IE 브라우저
   x = document.body.scrollLeft;
   y = document.body.scrollTop;
}
```

스타일 가져오기

대부분의 스타일 속성은 x.style을 통해 읽을 수 있지만 인라인(inline) 속성만 사용할 수 있기 때문에 그렇지 않은 스타일 속성을 찾기 어렵다. MS에서는 x.currentStyle이라는 메소드를 지원하지만 W3C에서는 getComputedStyle()을 사용하므로 브라우저간 호환성 문제가 생긴다.

```
#test {font-size: 16px;
   padding: 10px;
   width: 50%;
   border-width: 1px;
   border-style: solid;
   border-color: #cc0000;
}
```

위와 같은 test라는 클래스의 font-size 속성을 읽어 오려면 다음과 같이 한다.

```
testProp=getStyle("test","width");
function getStyle(el,styleProp)
{
   var x = document.getElementById(el);
```

```
   if (x.currentStyle)
           var y = x.currentStyle[styleProp];
   else if (window.getComputedStyle)
           var y =
document.defaultView.getComputedStyle(x,null).getPropertyValue(style
Prop);
   return y;
}
```

DOM에서 pixel 처리

W3C DOM2에서는 style.left 이나 style.top 속성이 돌려주는 값은 CSS의 단위("px"등)를 포함한다. 그러나, 넷스케이프4의 element.left 나 IE4/5 의 element.style.pixelLeft 는 정수치를 돌려준다. 요소의 왼쪽 혹은 위의 내부 스타일 설정을 정수값으로 읽기 위해서는 parseInt() 을 사용해 문자 라인에서 정수값을 받는다. 반대로 설정하고자 하면 px과 같은 단위를 꼭 설정해야 한다.

```
x.style.pixelLeft = x; // IE4/5
x.style.pixelTop = y; // IE4/5
x.style.left = value + "px"; // DOM Level 2
x.style.top = value + "px"; // DOM Level2
```

W3C DOM과 innerHTML 성능

이미 언급한 바와 같이 W3C의 노드를 생성하는 방법으로 DOM을 생성할 때 실제 브라우저에서 속도가 많이 느린 것을 경험 할 수 있다. 50x50 정도의 테이블을 그릴 때, W3C DOM 방식과 HTML 테이블 DOM 방식, innerHTML 방식을 비교하여 실험해 본 결과 innerHTML의 성능이 가장 빠른 것을 알 수 있다. (http://www.quirksmode.org의 실험 결과)

방식 및 브라우저별 성능	IE5	IE6	Firefox 1.0	Safari1.3	Opera8
W3C DOM1 노드 생성	380	3000	340	330	510
W3C DOM2 노드 생성	330	3000	320	205	340
Table 그리기 방식	3600	9000	290	150	400
innerHTML 방식	100	100	110	100	110

일반적으로 자바스크립트의 수행 속도는 상당히 빨라서 성능에 영향을 주지는 않는다. 다만 문제되는 경우 위와 같은 DOM을 조작하고 렌더링 하는 부분에서 성능 차이가 나고 있다. 이러한 성능 차이를 볼 수 있는 사이트는 아래와 같다.

1. http://www.oreillynet.com/javascript/2003/05/06/examples/dyn_table_benchmarker_ora.html (Table Benchmarer)
2. http://www.umsu.de/jsperf/ (DOM Performance Test)
3. http://msdn.microsoft.com/library/default.asp?url=/workshop/author/perf/dhtmlperf.asp (MSDN DHTML Performance)

이벤트(Events) 기능

MS DOM과 W3C DOM이 극단적으로 차이를 보이는 부분이 바로 이벤트(Events) 부분이다. 마우스의 클릭, 움직임 등을 파악하여 액션을 구현하는 이벤트 부분은 웹에서 가장 많이 사용되고 있는 부분이기도 하다. IE는 이벤트가 나오면 window.event 를 통해 event 객체를 전달 하는데 반해 파이어폭스는 event 객체를 바로 전달 받는다. 따라서, 이벤트 핸들러를 사용하는데 있어 차이가 발생하게 된다.

```
<script type="text/javascript">
function handleEvent(aEvent){
  // aEvent 가 null 이면 IE 이기 때문에 window.event 를 반환한다.
  var myEvent = aEvent ? aEvent : window.event;
}
</script>
<div onclick="handleEvent(event)">Click me!</div>
```

브라우저 호환 이벤트 캐칭을 위해서는 위의 스크립트가 항상 초기화 되어 있어야 한다. 아래 표즌 MS DOM Event와 W3C Dom Event 의 차이점을 설명한 것이다. 같이 사용할 수 있는 부분도 있으나 대부분 달리 쓸 수 밖에 없다.

키 이벤트 속성

키 이벤트를 나타내는 속성은 아래와 같다. Keycode가 권장 된다.

MS DOM Event	W3C DOM Event	사용 설명
altKey (altLeft)	altKey	Alt키 사용 여부
shiftKey(shiftLeft)	shiftKey	Shift키 사용 여부
ctrlKey(ctrlLeft)	ctrlKey	Ctrl키 사용 여부
keyCode	keyCode	ASCII코드 값으로 키 인식 (a=65)

마우스 위치 속성

마우스 위치를 나타내는 속성은 다음과 같다. clientX/Y가 권장 된다.

MS DOM Event	W3C DOM Event	사용 설명
clientX/Y	clientX/Y	윈도우에서 이벤트가 일어난 상대적 x /y좌표 (Safari는 document 기반으로 측정)
screenX/Y	screenX/Y	전체 화면에서 이벤트가 일어나는 위치
offsetX/Y	-	마우스 근방에 요소에서 상대적 위치 (모질라에서 작동 안함)
-	PageX/Y	전체 문서에서 상대적 위치 (IE에서 작동 안함)

위와 같이 표준 지원에 대한 범위가 다르므로 아래와 같은 코드로 마우스의 위치를 파악할 수 있다.

```
function getPosition(e)
{
   var posx = 0;
   var posy = 0;
   if (!e) var e = window.event; // 이벤트 검사
   if (e.pageX || e.pageY) {  // pageX/Y 표준 검사
           posx = e.pageX;
           posy = e.pageY;
   }
   else if (e.clientX || e.clientY) { //clientX/Y 표준 검사 Opera
           posx = e.clientX;
           posy = e.clientY;
           if (isIE) { // IE 여부 검사
                   posx += document.body.scrollLeft;
                   posy += document.body.scrollTop;
           }
   }
}
```

이벤트 핸들러 등록

MS DOM 및 W3C DOM 지원 브라우저 간에 완전히 다른 이벤트 핸들러 인터페이스를 가지고 있다.

MS DOM Event	W3C DOM Event	사용 설명
attachEvent()	addEventListener()	요소에 이벤트 핸들러 추가 x.addEventListener('click',doSomething,false)
detachEvent()	removeEventListner()	요소에서 이벤트 핸들러 삭제

양쪽을 모두 지원할 수 있는 attachEvent 판별 스크립트는 아래와 같다.

```
function attachEvent (obj, evt, fuc, useCapture) {
 if(!useCapture) useCapture=false;
   if(obj.addEventListener) {  // W3C DOM 지원 브라우저
      return obj.addEventListener(evt,fuc,useCapture);
   } else if(obj.attachEvent) { // MSDOM 지원 브라우저
      return obj.attachEvent("on"+evt, fnc);
   } else { // NN4 나 IE5mac 등 비 호환 브라우저
      MyAttachEvent(obj, evt, fnc);
      obj['on'+evt]=function() { MyFireEvent(obj,evt) };
   }
 }

function MyAttachEvent(obj, evt, fuc) {
```

```
  if(!obj.myEvents) obj.myEvents= {};
  if(!obj.myEvents[evt]) obj.myEvents[evt]=[];
  var evts = obj.myEvents[evt];
  evts[evts.length]=fnc;
 }

function MyFireEvent(obj, evt) {
  if(!obj )} !obj.myEvents || !obj.myEvents[evt]) return;
  var evts = obj.myEvents[evt];
  for (var i=0;len=evts.length; i<len;i++) evts[i]();
 }
```

XML 기능

여기에서는 표준 DOM에서 XML을 처리하는 방법을 알아보고자 한다.

XML 데이터 핸들링

W3C DOM의 XML 기능을 가장 잘 탑재하고 있는 것이 바로 모질라 계열 브라우저들이다. 비표준적인 처리 방식에서 IE와 크게 다른 점은 텍스트 노드에 공백이 들어 있는 경우 처리 방식이다. XML 노드 사이에 공백이 들어 있는 경우 IE는 XMLNode.childNodes[]속에 공백을 포함하지 않는다.

```
//XML:
 <?xml version="1.0"?>
 <myXMLdoc xmlns:myns="http://myfoo.com">
   <myns:foo>bar</myns:foo>
 </myXMLdoc>

 // JavaScript:
 var myXMLDoc = getXMLDocument().documentElement;
 alert(myXMLDoc.childNodes.length);
```

위의 예제에서 documentElement를 통해 myXMLDoc에 XML 문서를 로드한 후에 이를 표시하는 내용이다. Mozilla인 경우 이 자식 노드의 길이에서 공백을 포함하기 때문에 3이 나오지만 IE인 경우에는 1이 나오게 된다.

또한 모든 노드는 nodeType을 가지는데, 요소 노드인 형식1과 문서 노드인 형식 9를 가질 때, 텍스트 노드를 분리해 내기 위해 텍스트 노드 형식 3과 코멘트 노드 형식 8여부를 아래와 같이 확인해야 한다.

```
// XML:
 <?xml version="1.0"?>
 <myXMLdoc xmlns:myns="http://myfoo.com">
   <myns:foo>bar</myns:foo>
 </myXMLdoc>

// JavaScript:
 var myXMLDoc = getXMLDocument().documentElement;
 var myChildren = myXMLDoc.childNodes;
```

```
 for (var run = 0; run < myChildren.length; run++){
   if ( (myChildren[run].nodeType != 3) &&
(myChildren[run].nodeType != 8) ){
  // not a text or comment node
   }
 }
```

XML data island 처리

IE에서는 XML data islands라는 비표준 기능이 있다. 이것은 HTML 문서내에서 XML을 임베딩 시키는 것인데 <xml>이라는 비표준 태그를 사용하며 다른 브라우저에서는 지원하지 않는다. XHTML을 사요해서 같은 기능을 구현해 볼 수 있다.

```
<xml id="xmldataisland">
  <foo>bar</foo>
 </xml>
```

방법은 XML 문서를 생성하고 파싱하는 DOM 파서를 사용하는 것인데 모질라에서는 DOMParser라는 구현을 이용한다. IE에서는 ActiveX로 된 Microsoft.XMLDOM에서 이러한 작업을 처리할 수 있다.

```
var xmlString = "<xml
id=\"xmldataisland\"><foo>bar</foo></xml>";
 var myDocument;

 if (document.implementation.createDocument){
   // Mozilla 에서 DOMParser 를 이용한다.
   var parser = new DOMParser();
   myDocument = parser.parseFromString(xmlString, "text/xml");
 } else if (window.ActiveXObject){
   // IE 에서 XMLDOM 객체를 이용한다.
   myDocument = new ActiveXObject("Microsoft.XMLDOM")
   myDocument.async="false";
   myDocument.loadXML(xmlString);
 }
```

XMLHttpRequest 처리

IE에서는 IE5.0부터 MSXML데이터를 통신으로 처리하기 위해 XMLHTTPRequest라는 객체를 ActiveX Object에 포함해서 비표준으로 제공하고 있다. 이것을 사용하기 위해서는 ActiveXObject("MSxml2.XMLHTTP) 혹은 ActiveXObject("Microsofot.XMLHTTP)라는 객체를 불러서 사용할 수 있다.

그런데 이 기능이 모질라와 사파리 등 다른 브라우저에서 XMLHttpRequest라는 자바스크립트 객체로 지원하게 됨에 따라 비동기 통신 기능을 하는 표준으로 자리 잡게 되었다. 이른바 Ajax(Asynchronous Javascript and XML)이라는 것으로 잘 알려진 이 기능은 페이지 내에서 사용자의 데이터를 비동기적으로 받을 수 있도록 할 수 있다는 점에서 획기적인 기능이라는 평가를 받았다.

```
// 객체 판별
var xmlhttp = false;
if (window.XMLHttpRequest) {
    myXMLHTTPRequest = new XMLHttpRequest();
} else {
    myXMLHTTPRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
// 동기 통신 요청
  myXMLHTTPRequest.open("GET", "data.xml", false);
  myXMLHTTPRequest.send(null);

  var myXMLDocument = myXMLHTTPRequest.responseXML;

//비동기 통신 요청

  function xmlLoaded() {
    var myXMLDocument = myXMLHTTPRequest.responseXML;
  }

  function loadXML(){
    myXMLHTTPRequest = new XMLHttpRequest();
    myXMLHTTPRequest.open("GET", "data.xml", true);
    myXMLHTTPRequest.onload = xmlLoaded;
    myXMLHTTPRequest.send(null);
  }
```

표준 JavaScript 사용 방법

XHTML, CSS와 함께 웹 문서에서 가장 많이 사용되는 것이 JavaScript이다. 자바스크립트는 DOM을 핸들링 할 수 있는 클라이언트 사이드 언어로서 매우 가볍고 쉽지만 개발하는 데 고려 사항이 매우 많기 때문에 충분한 경험이 필요하다.

자바스크립트는 플래시의 ActionScript, 모질라 확장 기능, 위젯 등 최근의 많은 리치 인터넷 어플리케이션의 기반 언어가 되고 있으므로 이를 잘 알아 두는 것이 여러 모로 도움이 된다. 특히 웹 개발에서 자바 스크립트는 매우 중요한 요소이며 브라우저에 따른 호환성 문제에도 관심을 가져야 한다.

국내 웹사이트 중 대부분의 경우 브라우저에 따른 DOM 핸들링과 자바스크립트 오류로 인해 잘 작동되지 않기 때문이다.

ECMAscript vs. Jscript?

ECMA스크립트(ECMAScript)는 ECMA 인터내셔널의 ECMA-262 기술 명세에 정의된 표준화된 스크립트 프로그래밍 언어이다. 이 언어는 웹 상에서 널리 쓰이며, 흔히 자바스크립트 혹은 JScript로 간주되지만 두 용어는 특별한 의미 차이가 있다. ECMA스크립트와 자바스크립트, JScript의 관계를 이해하기 위해서는 ECMA스크립트의 역사를 알 필요가 있다.

1996년 3월, 넷스케이프에서 넷스케이프 네비게이터 2.0을 출시하면서 자바스크립트를 지원하기 시작했다. 웹 페이지 동작을 향상시키는 언어로서 자바스크립트의 성공 때문에 마이크로소프트가 이와 "적당히" 호환되는 JScript를 개발하는 계기가 되었다. JScript는 1996년 8월 인터넷 익스플로러 3.0에 포함되어 출시되었다.

이에 넷스케이프는 표준화를 위해 자바스크립트 기술 명세를 ECMA 인터내셔널에 제출하였고, 이 명세에 대한 작업은 ECMA-262의 이름으로 1996년 11월부터 시작됐다. ECMA-262의 초판은 ECMA 일반 회의에서 1997년 6월 채택됐다.

ECMAScript는 ECMA-262에 의해 표준화된 언어의 이름이다. 자바스크립트와 JScript는 모두 ECMA스크립트와의 호환을 목표로 하면서 ECMA 명세에 포함되지 않는 다른 확장 기능을 제공한다. 1997년 6월과 1998년 6월에 각각 1,2 버전이 발표 되었으며 2000년 7월 자바스크립트 1.5를 기반으로 3 버전(ECMA-327)가 발표되었다. 2004년 6월에 ECMAscript for XML을 포함한 E4X 명세(ECMA-357)이 발표 되었다.

자바스크립트	JScript	ECMA스크립트
1.0 (넷스케이프 2.0, 1996년 3월)	1.0 (IE 3.0 - 초기 버전, 1996년 8월)	
1.1 (넷스케이프 3.0, 1996년 8월)	2.0 (IE 3.0 - 후기 버전, 1997년 1월)	
1.2 (넷스케이프 4.0, 1997년 6월)		
1.3 (넷스케이프 4.5, 1998년 10월)	3.0 (IE 4.0, 1997년 10월)	초판 (1997년 6월) / 2판

		(1998년 6월)
1.4 (넷스케이프 서버에만 사용됨)	4.0 (비주얼 스튜디오 6, IE에는 사용되지 않음)	
	5.0 (IE 5.0, 1999년 3월)	
	5.1 (IE 5.01)	
1.5 (넷스케이프 6.0, 2000년 11월; 이후 넷스케이프와 모질라 포함)	5.5 (IE 5.5, 2000년 7월)	3판 (1999년 12월)
	5.6 (IE 6.0, 2001년 10월)	
	JScript(ASP.NET; IE에는 포함되지 않음)	
자바스크립트 2.0 (제안)		4판 (진행중)

- 출처: 위키퍼디아(ko.wikiperdia.org)

- 참고: http://www.ecma-international.org/publication/standard/Ecma-357.htm

스크립트 개발시 유의점

브라우저 스니핑

웹 표준 기반 개발은 웹브라우저와 관계 없이 통일된 웹페이지를 제공하는 데 목표가 있지만 실제로 웹브라우저에 따라 달리 표현하는 부분이 있기 때문에 사용자의 웹브라우저의 벤더와 버전을 확인하여 이에 따라 적절하게 웹페이지를 표시하거나 대응할 필요가 있다. 이는 오래된 웹브라우저를 사용하거나 특정 브라우저에서만 동작하는 기능을 제공할 때 특히 그렇다.

1994~2000년도 사이에 나온 브라우저들은 브라우저 시장 경쟁에서 이기기 위한 목적으로 출시된 것들이어서 W3C에서 제정하는 표준을 지키는 브라우저는 아니었다. 브라우저간 비호환성은 웹서비스 발전에 가장 중대한 도전이기 때문에 이를 표준적으로 지원하는 브라우저의 출현은 필수 불가결한 것이었다. 현재 모질라 1.0 (넷스케이프6) 이상, IE5.5이상 버전의 브라우저들은 W3C의 웹 페이지 표현에 대한 표준인 HTML4.0, CSS1/2, W3C DOM 시 방식을 지원하고 있다.

크로스 브라우징을 통해 웹페이지를 완벽하게 개발을 하기 위해서는 브라우저의 기능을 동작시에 판별할 수 있어야 한다. 즉, 에러를 일으키지 않고 다양한 방문자들이 폭 넓게 사용해 주기 위한 것이다. 일반적으로 사용되는 방법은 번거럽지만 브라우저를 식별하여 설계 시에 브라우저의 능력에 띠라 웹페이지를 만드는 것이다. 그렇지만, 다양한 브라우저의 다른 기능을 개발자가 알아서 판단하고 제공한다는 것은 쉬운 일은 아니다. 그러나, 지금까지 나열된 웹브라우저 차이점을 숙지하여 브라우저에 따라 판별 해 준다면 매우 유용할 것이다.

다음은 브라우저를 판별하는데 사용하는 몇 가지 방법들이다.

```
if (navigator.appName == "Microsoft Internet Explorer") {
   document.all(id).style.visibility = "visible";
} else if (navigator.appName == "Netscape") {
    if (parseInt(navigator.appVersion) < 5) {
   document.layers[id].visibility = "show";
} else {
   document.getElementById(id).style.visibility = "visible";
  }
}
```

위의 예에서는 navigator 객체의 appName 이라고 하는 속성 값을 따라 "Microsoft Internet Explorer" 혹은 "Netscape"를 판별하여 대응하는 코드를 실행하게 된다. 그러나, Opera와 같이 navigator.appName나 navigator.appVersion의 값을 간단하게 변경할 수 있는 브라우저도 있고 개개의 브라우저를 하나하나 판별해야 하기 때문에 좋은 방법이라 할 수 없다.

그래서 대부분 객체 기반의 브라우저 판별법을 사용한다. 지원하는 브라우저에 객체모델이 존재하는지 여부를 통해 간단하게 구현 기능을 확인하는 것이다.

```
if (document.getElementById)  {    // NS6+, IE 5+, Opera 5+
    elm = document.getElementById(id);
  }
  else if (document.all)  {   // IE4, Opera
    elm = document.all[id];
  }
  else if (document.layers)  {   // NN4
    elm = document.layers[id];
  }
```

이 예는 document.getElementById이라고 하는 객체를 가지고 있는 브라우저에 대해서는 같은 코드를 실행한다. document.getElementById 객체는 W3C이 규정되어 있는 DOM의 표준으로 최근 웹브라우저는 대부분 지원하므로 통상 이 방법을 사용해야 한다. 따라서 W3C DOM을 사용하는 표준 웹브라우저에서 다음과 같이 … 로 규정된 영역을 이동하는 간단한 스크립트를 생성할 수 있다.

```
function moveElement(id, x, y){   // W3C DOM Browser
  var elm = document.getElementById(id);
  if (elm)   {
    elm.style.left = x + 'px';
    elm.style.top  = y + 'px';
  }
}
```

Browser sniffing으로 불리는 이러한 방법은 흔히 ECMAScript 함수에 의해 다루어져 아래와 같은 스크립트로 브라우저의 버전과 제품 벤더를 확인할 수도 있다.

```
// convert all characters to lowercase to simplify testing
 var agt=navigator.userAgent.toLowerCase();

 // *** BROWSER VERSION ***
```

```
 // Note: On IE5, these return 4, so use is_ie5up to detect IE5.
 var is_major = parseInt(navigator.appVersion);
 var is_minor = parseFloat(navigator.appVersion);

 // Note: Opera and WebTV spoof Navigator
 var is_nav = ((agt.indexOf('mozilla')!=-1) &&
(agt.indexOf('spoofer')==-1)
 && (agt.indexOf('compatible') == -1) && (agt.indexOf('opera')==-1)
 && (agt.indexOf('webtv')==-1) && (agt.indexOf('hotjava')==-1));
 var is_nav2 = (is_nav && (is_major == 2));
 var is_nav3 = (is_nav && (is_major == 3));
 var is_nav4 = (is_nav && (is_major == 4));
 var is_nav4up = (is_nav && (is_major >= 4));
 var is_navonly = (is_nav && ((agt.indexOf(";nav") != -1) ||
 (agt.indexOf("; nav") != -1)) );
 var is_nav6 = (is_nav && (is_major == 5));
 var is_nav6up = (is_nav && (is_major >= 5));
 var is_gecko = (agt.indexOf('gecko') != -1);

 var is_ie = ((agt.indexOf("msie") != -1) && (agt.indexOf("opera")
== -1));
 var is_ie3 = (is_ie && (is_major < 4));
 var is_ie4 = (is_ie && (is_major == 4) && (agt.indexOf("msie 4")!=-
1) );
 var is_ie4up = (is_ie && (is_major >= 4));
 var is_ie5 = (is_ie && (is_major == 4) && (agt.indexOf("msie
5.0")!=-1) );
 var is_ie5_5 = (is_ie && (is_major == 4) && (agt.indexOf("msie
5.5") !=-1));
 var is_ie5up = (is_ie && !is_ie3 && !is_ie4);
 var is_ie5_5up =(is_ie && !is_ie3 && !is_ie4 && !is_ie5);
 var is_ie6 = (is_ie && (is_major == 4) && (agt.indexOf("msie
6.")!=-1) );
 var is_ie6up = (is_ie && !is_ie3 && !is_ie4 && !is_ie5
&& !is_ie5_5);

 // KNOWN BUG: On AOL4, returns false if IE3 is embedded browser
var is_aol = (agt.indexOf("aol") != -1);
 var is_aol3 = (is_aol && is_ie3);
 var is_aol4 = (is_aol && is_ie4);
 var is_aol5 = (agt.indexOf("aol 5") != -1);
 var is_aol6 = (agt.indexOf("aol 6") != -1);

 var is_opera = (agt.indexOf("opera") != -1);
 var is_opera2=(agt.indexOf("opera 2") != -1 ||
agt.indexOf("opera/2") != -1);
 var is_opera3=(agt.indexOf("opera 3") != -1 ||
agt.indexOf("opera/3") != -1);
 var is_opera4=(agt.indexOf("opera 4") != -1 ||
agt.indexOf("opera/4") != -1);
 var is_opera5=(agt.indexOf("opera 5") != -1 ||
agt.indexOf("opera/5") != -1);
 var is_opera5up=(is_opera && !is_opera2 && !is_opera3
&& !is_opera4);
```

```
var is_webtv = (agt.indexOf("webtv") != -1);

var is_TVNavigator = ((agt.indexOf("navio") != -1)
|| (agt.indexOf("navio_aoltv") != -1));
var is_AOLTV = is_TVNavigator;

var is_hotjava = (agt.indexOf("hotjava") != -1);
var is_hotjava3 = (is_hotjava && (is_major == 3));
var is_hotjava3up = (is_hotjava && (is_major >= 3));
```

코드 작성 시 주의 사항

일반적인 코딩 규칙

자바스크립트를 사용할 때는 <script> 태그에 language="JavaScript"를 속성으로 선언해 사용하는 경우가 있는데 반드시 type="text/javascript"를 사용해 준다. JScript, VBscript 등은 IE에서만 사용하므로 사용하지 않도록 한다. 또한, 텍스트 브라우저나 비 스크립트 브라우저를 위해 NOSCRIPT라는 요소를 사용하여 대체 텍스트나 링크를 제공하거나, 클라이언트측 스크립트 대신에 서버측 스크립트를 사용해 호환성을 높여 주는 것이 좋다.

```
<SCRIPT type="text/javascript"><!--   //  // --> </SCRIPT>
  <NOSCRIPT>
  <UL>
    <LI><A HREF="choice1.html">Choice1</A></LI>
    <LI><A HREF="choice2.html">Choice2</A></LI>
  </UL>
</NOSCRIPT>
```

또한, 내용은 꼭 코멘트를 사용하여 텍스트 브라우저에서도 잘 표현 되도록 해야한다. 코멘트를 사용할 때는 <!--- -Comment-------> 로 쓰는 것은 잘못된 방법으로 <!--로 시작하여 -->로 끝내고, 주석 내용 안에는 하이폰(-)이 두개 이상 들어가지 않도록 한다. 즉, <!--==Comment==-->, <!-- Comment --> 방식이 바른 표현이다.

getYear()의 Y2K 문제

모든 브라우저가 ECMAscript의 기본 함수와 기능을 모두 지원하고있다. 따라서 ECMAScript만으로 스크립트 프로그래밍을 하는 것이 바람직 하다. 하나의 예를 들면, 국민은행 홈페이지에 비 IE 브라우저로 접속 하면 날짜를 확인할 수 없다는 에러창이 뜬다. 이 에러 창은 날짜를 받는 Jscript 전용 함수를 사용해서 그렇다.

getYear() 라는 함수는 IE인 경우 2005, 비 IE 브라우저인 경우 2005년을 105로 반환한다. 특히 1998년의 경우는 모두 98로 반환한다. 이런 에러를 없애기 위해서는 ECMAscript Spec에 있는 getFullYear()함수를 사용해야 한다.

속성 입력할 때 주의점

스크립트나 애플릿, 또는 다른 프로그램 객체를 사용하지 않거나 지원하지 않는 경우에도 페이지의 내용을 이해할 수 있어야 한다. 그것이 불가능하다면, 대안적으로 접근 가능한

페이지에 그들을 대체할만한 정보를 제공하는 것이 좋다.

예를 들면, 스크립트 기능이 꺼져 있거나 지원되지 않을 경우에도 스크립트를 활성화하는 링크가 작동하도록 해야 한다. (예를 들어, 링크의 목적지로 "javascript:"를 쓰지 않아야 한다. href 속성의 값으로 "javascript:"를 쓰는 것은 접근성 지침 위반일 뿐 아니라 HTML 표준 위반이기도 하다. 이런 경우, onClick 등을 사용해야 한다.

```
<a href="javascript:goURL(here);">틀린 표현</a>
<a href="#" onClick="javascript:goURL(here);">맞는 표현</a>
```

사용자 form에서 action을 받은 후 나온 결과에 자바스크립트 만을 제공해 자동 전환하는 결과는 될 수 있으면 사용하지 않는다.

```
<script> href.location="test.html"; </script> // 나쁜 표현
```

같은 내용만 담는 것은 권장하지 않으며 만약 한다면, window.href.location 이라고 정확하게 표현하거나, <meta> 태그의 refresh를 사용하거나 하는 것이 옳다. 또한, 결과에 링크로 직접 POST 하지 않고 document.form.submit(); 같은 방식을 쓰는 것도 지양해야 한다. 모질라에서는 사용자의 액션이 없는 자동 forum.submit()을 지원하지 않는다.

스크립트 블록 실행

자바스크립트 블록에 실행 스크립트를 넣어 바로 실행하게 하는 것도 올바르지 않은 방법이다.

```
// 올바르지 않은 방법
<div id="foo">Loading...</div>
<script type="text/javascript">
  document.getElementById("foo").innerHTML = "Done.";
</script>
// 올바른 방법
<body onload="doFinish()">
<div id="foo">Loading...</div>
<script type="text/javascript">
  function doFinish() {
    var element = document.getElementById("foo");
     element.innerHTML = "Done.";
  }
</script>
```

반드시 실행할 함수를 onLoad 함수로 넣어 실행하도록 한다.

Strict 모드에서 document.write()

자바스크립트는 document.write로 HTML 내용을 생성할 수 있다. 그런데 <script>태그 안에 <script>를 생성 할때는 문제가 발생된다. 일반적으로 Transtional 모드에서는 다음과 같이 할 수 있다.

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

```
...
<script>
  document.write("<script>alert("Hello")</script>")
</script>
```

Strict 모드에서는 Mozilla의 렌더링 엔진이 이를 허용 하지 않게 때문에 아래와 같이 표현 해야 한다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
...
<script>
  document.write("<script>alert("Hello")</" + "script>")
</script>
```

디버깅 및 품질 관리

지금까지 각종 웹 표준을 알아보고 이에 대한 각 웹 브라우저의 특성과 표준 지원 정도. 올바른 웹페이지 코딩 방법 등을 살펴보았다. 그러나, 이러한 가이드를 충분히 숙지하고 있어도 오류가 나는 것이 웹페이지이다. 가이드를 잘 익히는 것도 중요하지만, 결국 웹 개발자가 개발 중이나 최종 작업을 마치고 어떤 방식으로 디버깅을 하고 품질 관리(QA)를 하는 가 하는 점도 매우 중요하다. 이 장에서는 최신 디버깅 및 품질 관리 방법을 알아 본다.

기본 디버깅 방법론

웹 브라우저 기반 디버깅

우선 개발 중 가장 빠르게 해 볼 수 있는 것이 바로 여러 웹브라우저에서 기능을 구현하여 동작 여부를 체크하는 것이다. IE4.0, IE5.5, IE6.0, Firefox 1.5, Nescape7, Opera8, Safari1.3, Lynx2.8 등의 브라우저에서 확인해 볼 려면, 아래 링크를 따라가면, 각 웹브라우저의 예전 버전까지 제공해 준다.

- 인터넷 익스플로러: http://browsers.evolt.org/?ie/
- 모질라 파이어폭스: http://browsers.evolt.org/?mozilla/
- 오페라: http://browsers.evolt.org/?opera/
- 넷스케이프: http://browsers.evolt.org/?navigator/
- 사파리: http://browsers.evolt.org/?safari/
- 링스(Lynx): http://browsers.evolt.org/?lynx/

웹 브라우저를 통한 디버깅에서 가장 쉬운 것은 웹페이지의 간단한 스크립트 오류를 알아내기 위해서는 파이어폭스에 있는 자바스크립트 콘솔을 이용하는 방법이 있다. 이 콘솔을 이용하면, 표준안에 근접한 방법으로 웹페이지를 디버깅 할 수 있는 장점이 있다. 파이어폭스는 W3C 표준 DOM과 ECMAscript를 지원하기 때문에 가장 먼저 개발 시 적용해 보고 다른 웹 브라우저로 확인 하는 방법을 이용하면 좋다.

그러나, 각 브라우저 버전별로 DOM, CSS JavaScript Core 등이 조금씩 다르기 때문에 호환성 테스트를 할 때 여러 문제들이 있다. 아래 사이트에서 PC에 동시에 설치할 수 있는 다양한 버전의 InternetExplorer를 다운로드 받을 수 있다.

http://www.skyzyx.com/downloads/

자동화된 UnitTest를 통해 웹 브라우저별로 한번씩 실행시켜주기만 하면 되니까 편리 하게 디버깅이 가능하다. 아래 그림은 IE 5.01, 5.5, 6.0, FireFox 1.0에서 자동화된 테스트를 통해 자바스크립트를 테스트 하고 있다.

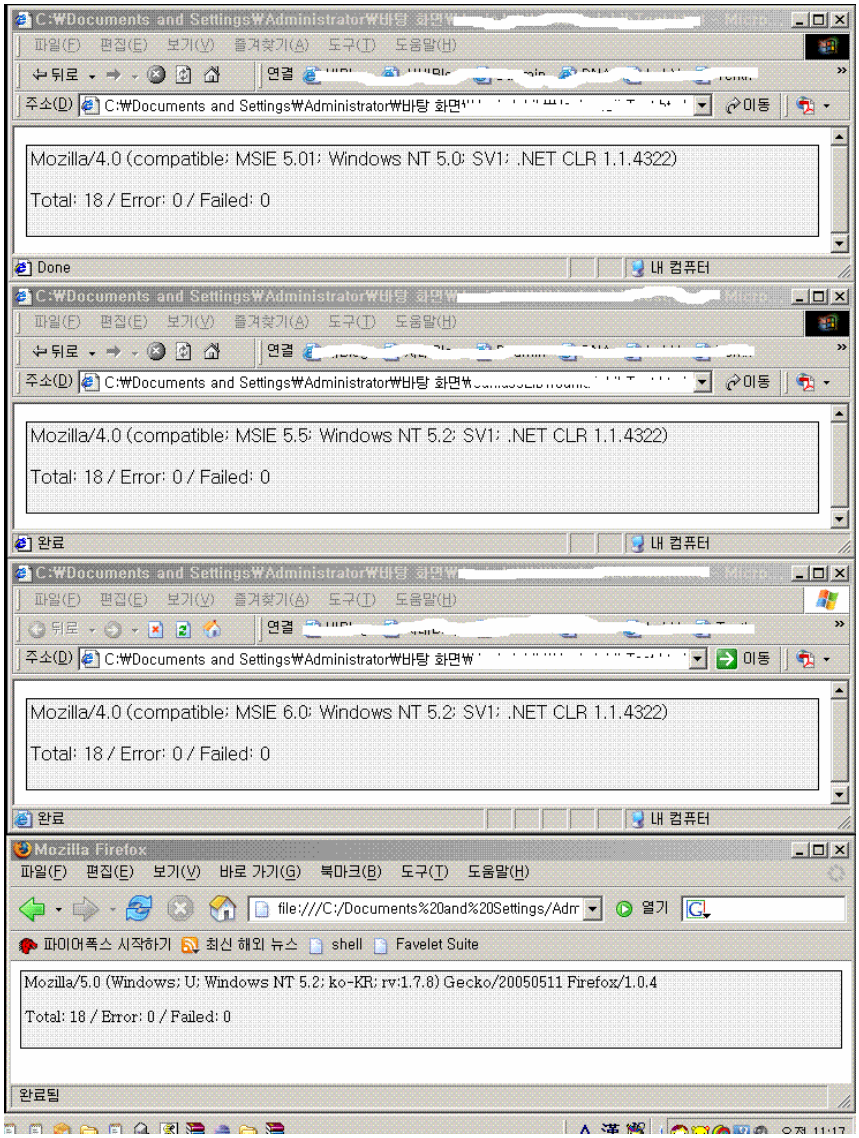


그림 31 다양한 웹 브라우저를 한번에 띄워 테스트 하는 모습

디버깅 도구 이용

앞서 언급한 대로 각종 웹브라우저에는 확장 스크립트 디버거들이 있다.

- ❑ Firefox Javascript Debugger : 브라우저 내장
- ❑ Microsoft Script Debugger: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sdbug/Html/sdbug_1.asp
- ❑ Internet Explorer Developer Toolbar : http://www.microsoft.com/downloads/details.aspx?FamilyID=e59c3964-672d-4511-bb3e-2d5e1db91038&displaylang=en
- ❑ Visual Studio Script Debugger : http://msdn.microsoft.com/vstudio/
- ❑ Venkman Javascript Debugger: http://www.mozilla.org/projects/venkman/

이러한 스크립트 디버거로 알 수 있는 것은 DOM 요소와 속성 사용에 대한 에러 처리 같은 것이다. 만약 Javascript 문법에 대해서도 확인하고자 한다면, Strict로 처리한다.

var response = true; var response = false; 라는 코드를 Strict모드로 한 경우 흔히 나타나는 "redeclaration of var response" 에러의 경우 위의 문법을 아래와 같이 수정해야 에러가 없어진다. var response = true; response = false;

유사한 기능을 하는 IE에서도 MS 스크립트 디버거라는 프로그램이 있다. 기본적으로 자바스크립트에 에러가 나면 아래와 같은 경고창이 나온다. 여기에는 에러가 생긴 곳(Breakpoint)의 행과 문자 위치만 나오며 특별한 에러 메시지가 표시되지 않기 때문에 오류를 찾아내는 것이 쉽지는 않다.

그리고, HTML의 표현상 오류는 소스를 간단히 살펴 봄으로서 해결이 되는 경우가 많다. 그러나 자바스크립트 문법과 DOM의 사용상의 오류는 쉽게 알아내기 힘들다. 따라서 이러한 경우를 대비하여 디버거를 사용할 수 있다.

IE Developer Toolbar

MS에서 2004년부터 새로운 IE7 버전을 만들기 위해 꾸린 IE팀에서 웹 개발자를 위해 만든 툴바이다. 이 툴바를 이용하면 DOM Explorer, Validator, Ruler 등 다양한 기능을 이용할 수 있다.

다운로드: http://www.microsoft.com/downloads/details.aspx?FamilyID=e59c3964-672d-4511-bb3e-2d5e1db91038&displaylang=en

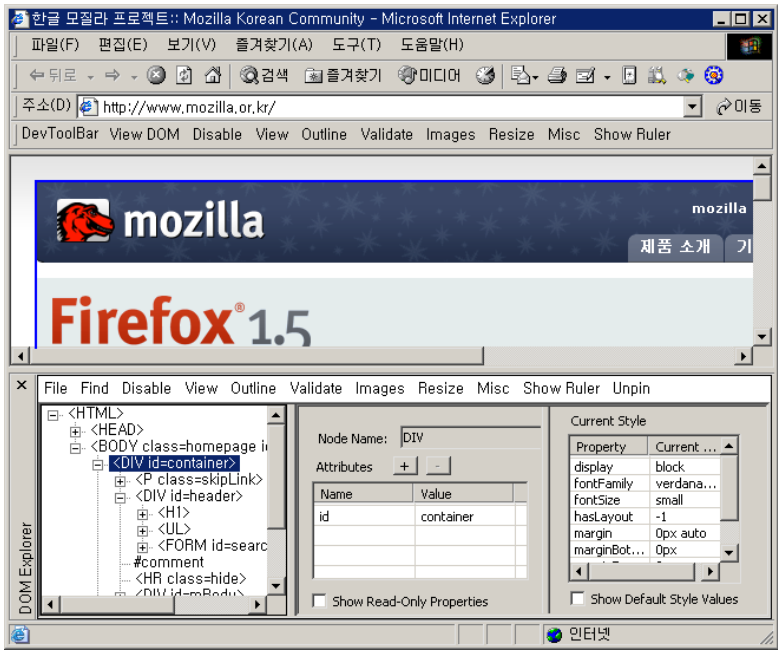


그림 32 IE 개발자 툴바를 통한 DOM 스크립트 디버깅

Venkman 스크립트 디버거

Mozilla 프로젝트 중에는 자바스크립트 콘솔 및 벤크맨(Venkman)으로 불리는

JavaScript 디버거가 내장 되어, 스크립트 개발자들에게 이용되고 있다. 이것은 화면 표시와 콘솔 양쪽에서 조작할 수 있는 디버거이다. 스크립트에서 잘못된 코드로 인해 만들어지는 종료점(beakpoint)관리, Call Stack 감시 변수/객체 감시라고 하는 기능을 화면 콘솔 커멘드로 이용 가능하며, 대화형 콘솔에서는 임의의 JavaScript 코드를 실행시킬 수도 있다. 키보드 쇼트 컷은 기존의 비주얼 디버그 환경에 맞추고 있어 gdb 의 사용자이면 벤크맨의 break, step, next, finish, frame 및 where 커멘드를 자연스럽게 사용할 수 있다.

이 JavaScript 디버거는 Windows 환경에서의 비주얼 상호 개발 환경이나 다른 대규모 웹 개발도구보다 뛰어나 Mac OS 나 Unix 를 포함해 다른 플랫폼에 대해서는 비주얼 디버그 환경에서 이 정도까지 포괄적으로 적용 가능하다. 왼쪽의 스크린샷은 벤크맨의 실행 모습이다.

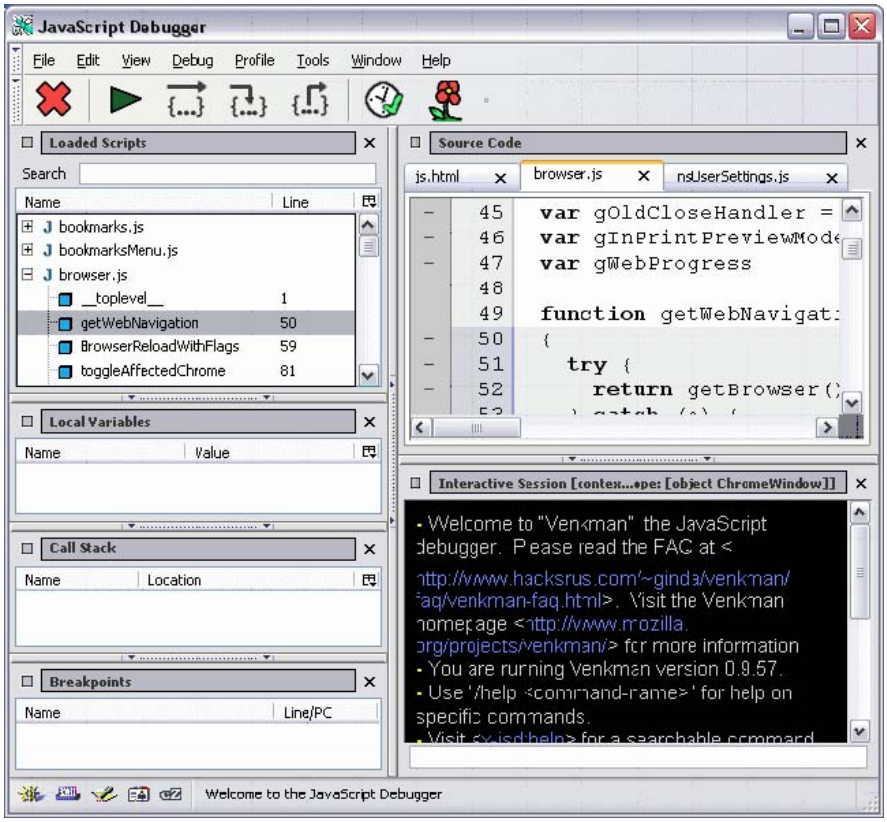


그림 33 Venkman을 통한 스크립트 디버깅

DOM Inspector 검사기

W3C의 표준 권고안인 DOM에 대한 체계적인 구조도와 웹페이지 상의 잘못된 사용 방법을 알려주는 도구인 DOM Inspector가 모질라에 역시 내장되어 있다. 임의의 웹 문서나 XUL 어플리케이션으로 이용 중 DOM 을 정밀 조사 하거나 수정하거나 하는데 사용할 수가 있는 도구로사, 문서 및 내부의 노드를 다종 다양한 시점에서 보는 볼 수 있는 윈도우를 이용해 DOM 계층을 탐색할 수 있다. 아래쪽 스크린샷은 전형적인 DOM 정밀 조사 작업의 모습이다. 이 프로그램은 Firefox에 자체적으로 내장되어 있다. Firefox를 설치할 때 설치 내용을 묻는 대화 상자에서 고급 사용자 정의로 설치 하고 '개발 도구'를 체크하면 DOM Inspector가 설치 된다.

그렇지 않더라도 DOM Inspector 확장 기능을 http://update.mozilla.org 에 접속하여 다운로드 받을 수도 있다. 매우 편리한 기능을 가지고 있다.

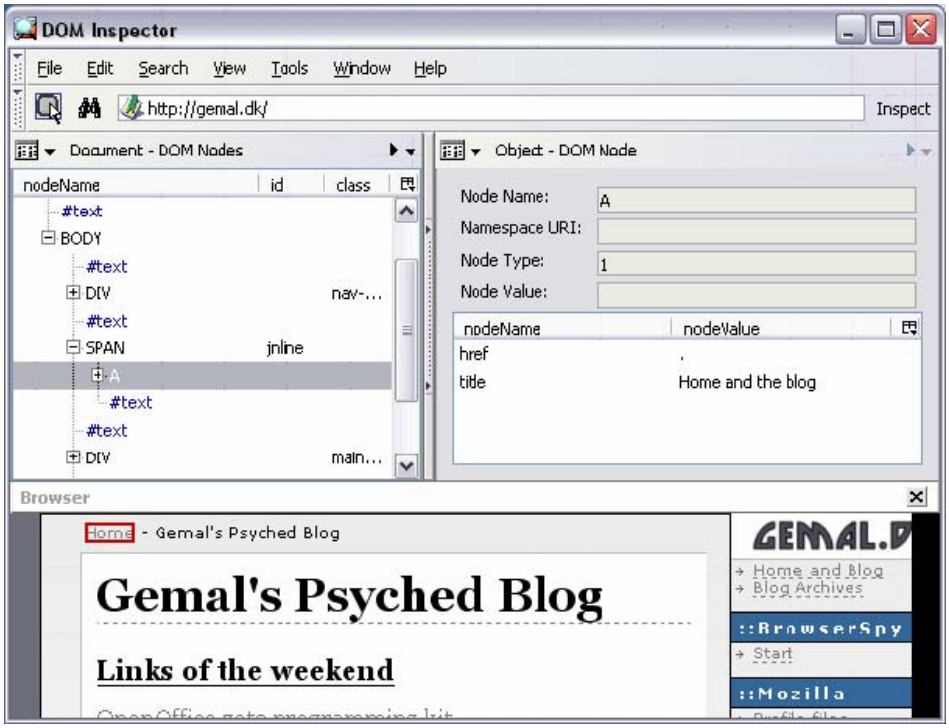


그림 34 DOM Inspector를 통한 DOM 디버깅 (Firefox 내장)

그 밖에 모질라에서는 IE와 달리 페이지 소스 보기에서 문법을 하이라이트 처리하여 별도로 확인 할 수 있으며, 캐쉬 관리자, HTTP 헤더 보기를 통해 웹서버와의 통신 과정에서 일어나는 일련의 과정을 모두 디버깅 해 볼 수 있다. 이러한 다양한 디버깅 방법들을 활용하여 보다 웹 표준에 가까운 웹페이지 구현이 가능하다.

Interactive Shell

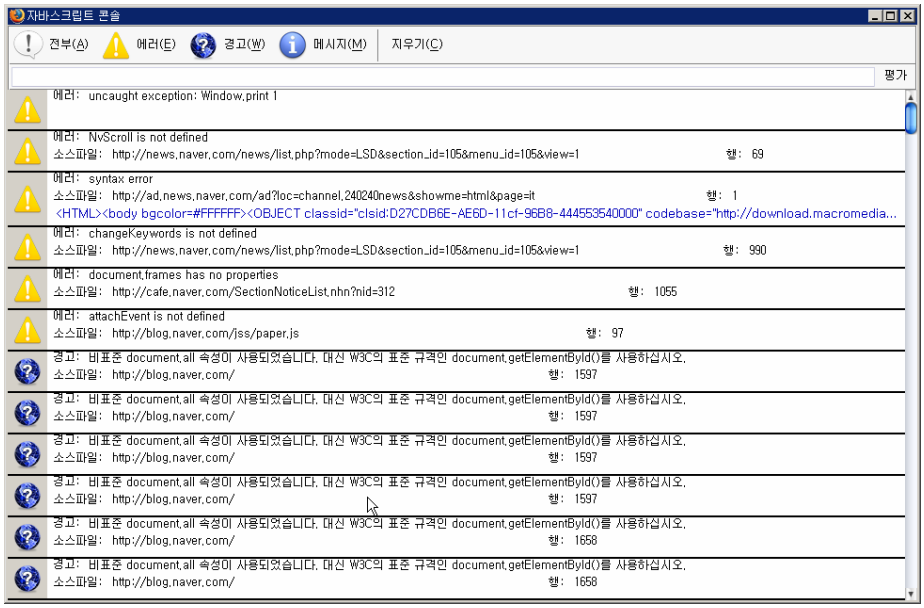
자바 스크립트 명령을 실행해 볼 수 있는 흥미로운 프로그램이다. Interactive Shell (http://www.squarefree.com/shell/)은 Firefox에서 BookMarklet으로 등록해두면 원하는 페이지에서 Shell을 띄워 DOM inspection을 할 수 있다.

Favelet Suite

HTTP Response Header Viewer, Hidden Field Modifier, DOM Inspector 등 유용한 디버깅 툴을 모아둔 BookMarklet 이다. http://slayeroffice.com/index.php

Firefox 자바스크립트 콘솔

Firefox에는 자체 내장되어 있는 자바 스크립트 콘솔이 있다. 이 콘솔을 이용하면 웹 페이지내에 에러와 경고를 상세하게 보여 주며 정확한 위치의 소스를 보여 주는 기능 까지 가지고 있다.



```
에러: uncaught exception: Window.print 1
에러: NvScroll is not defined
소스파일: http://news.naver.com/news/list.php?mode=LSD&section_id=105&menu_id=105&view=1          행: 69
에러: syntax error
소스파일: http://ad.news.naver.com/ad?loc=channel.240240news&showme=html&page=it          행: 1
<HTML><body bgcolor=#FFFFFF><OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="http://download.macromedia...
에러: changeKeywords is not defined
소스파일: http://news.naver.com/news/list.php?mode=LSD&section_id=105&menu_id=105&view=1          행: 990
에러: document.frames has no properties
소스파일: http://cafe.naver.com/SectionNoticeList.nhn?nid=312          행: 1055
에러: attachEvent is not defined
소스파일: http://blog.naver.com/jss/paper.js          행: 97
경고: 비표준 document.all 속성이 사용되었습니다. 대신 W3C의 표준 규격인 document.getElementById()를 사용하십시오.
소스파일: http://blog.naver.com/          행: 1597
경고: 비표준 document.all 속성이 사용되었습니다. 대신 W3C의 표준 규격인 document.getElementById()를 사용하십시오.
소스파일: http://blog.naver.com/          행: 1597
경고: 비표준 document.all 속성이 사용되었습니다. 대신 W3C의 표준 규격인 document.getElementById()를 사용하십시오.
소스파일: http://blog.naver.com/          행: 1597
경고: 비표준 document.all 속성이 사용되었습니다. 대신 W3C의 표준 규격인 document.getElementById()를 사용하십시오.
소스파일: http://blog.naver.com/          행: 1597
경고: 비표준 document.all 속성이 사용되었습니다. 대신 W3C의 표준 규격인 document.getElementById()를 사용하십시오.
소스파일: http://blog.naver.com/          행: 1658
경고: 비표준 document.all 속성이 사용되었습니다. 대신 W3C의 표준 규격인 document.getElementById()를 사용하십시오.
소스파일: http://blog.naver.com/          행: 1658
```

그림 35 Firefox 자바스크립트 콘솔을 통한 디버깅

올바른 플러그인(Plugin) 사용

외부 객체 이용 방법

올바른 OBJECT의 사용 방법

Cross browsing에서 가장 난감한 문제에 봉착했다. 바로 Plugin이라는 문제이다. Plugin 이란 HTML상에 특정 어플리케이션의 기능을 실행할 수 있도록 해주는 기술을 말하는 것으로 매크로미디어사의 Flash, 어도비의 Acrobat Reader, 리얼네트웍스의 Real Player, 마이크로소프트의 Windows Media Player 등이 여기에 속한다. 이들은 간단한 실행형 파일만으로도 웹브라우저 내에서 응용 프로그램을 실행할 수 있다.

이는 브라우저 전쟁 당시에 넷스케이프와 마이크로소프트가 각각 NSplugin과 ActiveX라는 상호 배타적인 기술을 브라우저에 탑재하면서부터 시작되었다. NSplugin은 OS와 관계없이 제작 실행될 수 있고 Opera나 Sapari 같은 웹브라우저에 채용되는 반면, ActiveX는 IE가 설치된 윈도우즈 환경에서만 실행된다. 당시 넷스케이프는 <embed>, <applet>을 마이크로소프트는 <object>라는 별도의 태그를 만들면서 이 기능을 지원해 왔다. HTML 4.01에서는 object가 표준으로 제정되었기 때문에, object만 사용하면 되지만 예전 NN4버전과 표준 사용방법에 대해 여전히 논란이 제기되고 있다.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/sw
flash.cab#version=5,0,0,0" width="366" height="142" id="myFlash">
    <param name="movie" value="test.swf">
    <param name="quality" value="high">
    <param name="swliveconnect" value="true">
</object>
```

위의 코드는 IE에서 ActiveX를 불러오기 위한 HTML로서 classid는 clsid로 Flash의 고유 프로그램 식별자이고, codebase는 Flash ActiveX가 설치되지 않은 경우 설치 파일이 위치한 경로를 지정한다. param은 문서 객체 모델에서 자식 노드로서 참고 할수 있는 값들을 의미한다. Netscape6이나 Opera 4이상 부터는 object를 표준으로 지원하기는 하나 윈도우 미디어 플레이어 등 몇 가지에 국한된다. 또한, classid 같은 식별자를 사용하는 것이 아니라 mime-type을 기반으로 하기 때문에 응용프로그램을 인식하는 방법이 IE의 object 기술방법과 크게 다르다.

```
<object type="application/x-shockwave-flash"
data="test.swf" width="366" height="142" id="myFlash">
    <param name="movie" value="test.swf">
    <param name="quality" value="high">
    <param name="swliveconnect" value="true">
<p>You need Flash -- get the latest version from
<a href="http://www.macromedia.com/downloads/">here.</a></p>
</object>
```

위의 예제는 Mozilla 기반의 웹브라우저에서 flash를 찾아 실행하기 위한 object 의 사용 방법이다. mime-type을 통해 프로그램을 인지하고, IE와 같이 codebase를 통해 자동 설

치하는 기능을 넣고 있지 않다. 특히, param을 무시하는 웹브라우저가 있기 때문에 data=" ..swf" 파일 형식으로도 설정해 주고 있다. object를 사용하는데 있어, 모질라 기반 브라우저들이 더 표준에 가깝게 기술되고 있으나 워낙 논란이 많고 Plugin이 IE 기반이 많기 때문에 모두 같이 쓰는 것이 유리하다. 그러나 두 가지 코드를 동시에 사용하는 것은 불가능 하므로 위의 두가지 코드를 지원 여부에 따라 나누어 쓰거나 아예 두가지 정보를 함께 담는 방법도 있다.

```
if (window.ActiveXObject) {
// IE ActiveX Code
document.write("<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-
444553540000">
} else {
// Mozilla based Plugin Code
document.write("<object type="application/x-shockwave-flash"..">
}
```

<embed>와 <applet>은 지금까지 널리 쓰이긴 하였으나, HTML4.01에서 권고하지 않는 방법으로 되어 될 수 있으면 사용하지 않는 것이 좋다. 그러나 NN4와 같은 브라우저의 버전 호환성 유지 차원에서 필요할 수도 있다. 따라서 자바 애플릿을 불러올 경우에도 object 태그를 사용해야 하며, 아래와 같은 예제를 사용하여 불러온다.

```
<object classid="java:NervousText.class" width="534" height="50">
<param name="text" value="Java 2 SDK, Standard Edition v1.4">
<p>You need the Java Plugin. Get it from
<a
href="http://java.sun.com/products/plugin/index.html">here.</a>></p>
</object>
```

위와 같은 방법으로 object 태그의 호환성을 높일 수는 있으나 기본적으로 NSplugin과 ActiveX의 기술적인 호환성 때문에 제일 중요한 목표인 응용프로그램 실행에는 문제가 있다. 이를 해결하기 위해서는 ActiveX 개발자가 NSPlugin API를 참조하여 같은 기능의 Plugin을 개발해 줄 필요가 있다.

또한, 윈도우 환경에서는 ActiveX for Plugin, Plugin for ActiveX 등의 양쪽을 호환할 수 있는 plugin(http://www.iol.ie/~locka/mozilla/mozilla.htm)을 설치하거나, 리눅스 데스크탑 환경에서는 wine (http://www.winehq.org) 같은 윈도우 에뮬레이터를 설치하여 상호 기술을 사용 가능하다.

외부 객체 특허로 인한 문제 해결

최근 object와 plugin 사용과 관련하여 IE가 다른 업체의 특허를 침해했다는 법원 판결로 인터넷 사용 환경상 변화가 예상되는 가운데 인터넷 표준 언어인 HTML도 특허 논란에 휘말렸다. 2003년 9월 미국 시카고 법원은 지난달 "MS의 IE가 에올라스의 플러그인 기술 관련 특허를 침해한 것이 인정된다" 며 5억2000만달러를 배상하라고 판결했다. 이 소송의 핵심은 HTML 플러그인 사용에 대한 특허를 UC가 92년도에 받아서 발명만 전담으로 하는 회사인 이올라스에 94년에 팔았고, 이에 대해 소송을 제기해 W3C HTML 4.01의 object의 구동 방법에 대한 심각한 타격을 입힌 것으로 보인다. MS는 즉각 항소 의지를 밝히는 한편, W3C에 IE를 일부 수정할 뜻을 밝혔다. 이 문제는 IE 뿐만 아니라 모든 웹

브라우저에 관련된 중요한 사안으로 특허를 피해 나가는 방법으로 아래 제안된 방법을 사용하도록 권고 하고 있다.

Objec사용을 HTML내에 넣지 않고 JS파일에 넣어 실행하는 방법으로 아래와 같다.

```
// HTML File
<html>
  <body>
    <script src="embedControlOuterHTML.js"></script>
  </body>
</html>

// embedControlOuterHTML.js
embedControlLocation.outerHTML =  '<embed src="examplecontrol">';
```

아래 예제는 ActiveX 컨트롤에 Parameter 값을 설정하기 위해 createElement를 사용한 것이다.

```
// HTML File
<html>
  <body>
    <div id="embedControlLocation">
      <script id="elementid" src="embedControl.js"></script>
    </div>
  </body>
</html>


// embedControl.js
var myObjectElement = document.createElement('<object id="elementid"
  classid="clsid:098F2470-BAE0-11CD-B579-08002B30BFEB"></object>');

var myParamElement1 = document.createElement('<PARAM NAME=movie
  value="example.avi">');
var myParamElement2 = document.createElement('<Param name=quality
value=high>');
var myParamElement3 = document.createElement('<Param name=bgcolor
value=#FFFFFF>');

myObjectElement.appendChild(myParamElement1);
myObjectElement.appendChild(myParamElement2);
myObjectElement.appendChild(myParamElement3);

embedControlLocation.appendChild(myObjectElement);
```

그리고, PARAM값을 DATA 객체로 사용할 때는 BASE64로 인코딩 하거나 js파일에 넣어 document.write를 해야 된다.

- 참고URL

http://msdn.microsoft.com/library/?url=/workshop/author/dhtml/overview/activating_activex.asp

ActiveX와 대안 Plugin 기술

1995년 넷스케이프가 웹브라우저와 외부 프로그램과의 통신과 좀 더 다이나믹한 인터넷 경험을 제공하기 위해 플러그인(Plugin)이라는 기술을 선보였다. 그러나 마이크로소프트가 인터넷 익스플로러를 시장에 도입하면서 이에 대응하는 액티브X(ActiveX) 기술을 발표하게 된다. 액티브X는 윈도우의 COM/DCOM 환경에서 인터넷을 자유 자재로 사용할 수 있는 기술 플랫폼으로 인터넷 익스플로러에 임베딩 되어 실행 가능하다. 인터넷 익스플로러와 윈도우가 브라우저와 운영 체제를 독점하게 됨에 따라 윈도우 환경에서는 자바 애플릿이나 넷스케이프 플러그인 기술을 대체하게 되었다

인터넷 익스플로러가 웹브라우저 시장 독점 체제를 구축하는 동안 우리 나라에서는 세계에서도 유래를 찾아 볼 수 없을 정도로 급속한 인터넷 환경이 갖추어 지게 되었다. 이러한 성장의 이면에는 몇 가지 문제점이 발견되었는데 바로 특정 운영 체제와 플랫폼에 종속성이 심화된 것이다. 웹페이지를 만들 때도 현재의 W3C에서 발표한 웹 표준 스펙을 이용하지 않고, 당시 넷스케이프와 인터넷 익스플로러에서 경쟁적으로 사용되는 비표준 태그(Tag)와 마이크로소프트 표준을 기반으로 하는 문서 객체(Document Object Model) 사용 행태가 현재에도 고쳐지지 않고 있다. 인터넷 익스플로러도 최소한의 W3C 표준을 지키고 있어 표준을 따르기만 하면, 쉽게 모든 브라우저를 지원하게 된다.

또한, 플러그인(Plugin) 기술 사용에 있어 액티브X 종속성이 더 심화 되어 비 윈도우, 비 IE 환경에서는 사용이 거의 불가능하다는 것이다. 액티브X 종속성은 외국에 비해 우리 나라의 경우 매우 심각하다. 액티브X 콘트롤을 배포할 때 사용하는 코드사인(Codesign) 인증서의 경우, 베리사인으로부터 국내에 공급되는 양은 거의 800여개가 된다. 인증서 하나를 하나의 회사에서 사용한다고 본다면 적어도 800개의 업체에서 800개의 액티브X 콘트롤이 배포되고 있다는 것을 의미한다. 이는 세계에서 최대 규모이며 적어도 코드사인 인증서의 경우 세계 최대 시장이라고 들었다. 필자도 윈도우에 최소로 액티브X를 설치 하는데도 30여개 정도가 설치되어 있다.

왜 액티브X가 문제인가?

액티브X를 우리 나라에서 급속도로 쓸 수 밖에 없었던 이유가 몇 가지가 있다. 첫번째는 초고속 인터넷의 급속한 확대이다. 플러그인이 구동 되기 위해서는 별도의 프로그램을 다운로드 받아 설치를 해야 하는데 다운로드 크기가 큰 것은 모뎀 수준의 연결 속도에서는 다운로드 받을 때까지 기다릴 수 없는 것이다. 이런 이유로 아직 외국에서는 액티브X 같은 플러그인 기술을 사용하는 웹사이트가 거의 드물다. 플래쉬와 윈도우 미디어, 리얼 플레이어 등과 어도비 애크로뱃 리더처럼 설치시 제공하는 플러그인 정도가 대부분이다. 이에 반해 우리나라는 로그인, 채팅, 파일 첨부, 광고, 카드 결제, 인터넷 뱅킹, 사이버 트레이딩, 게임, 정부 민원 업무까지 그야말로 쓰이지 않는 곳이 거의 없을 지경이다. 초고속망 국가로 빨리 진입한 것이 플러그인 들을 부담 없이 사용하게 한 원인이다.

두번째는 웹과 어플리케이션을 구별이 모호한 이유이다. 웹은 정보의 자유로운 공유라는 측면에서 발전 되어 왔는데, 우리 나라에서는 웹을 어플리케이션이 하는 기능의 연장 선상에서 바라보고 있으며 그에 따라 부가 기능들이 계속적으로 필요로 하게 된 것이다. 소프트웨어의 모든 기능을 웹이 할 수 있다는 생각에 여러 기능이 덕지덕지 붙어서 결국 웹도 어플리케이션도 아닌 어중간한 웹사이트들이 만들어 지는 것이다.

세번째는 국가 차원에서의 플러그인 기술 장려를 들 수 있다. 우리 나라에서 플러그인 기술이 가장 먼저 도입된 것도 인터넷 뱅킹과 공인 인증 기술에 있다. 1999년 당시 128비트 암호화가 탑재된 웹브라우저가 미국 밖으로 수출이 안되고 있는 사이, 우리 나라에서는 128비트 암호화가 가능한 SEED라는 알고리듬을 개발했고 이를 국가 인증 기술로 탑재하는 과정에서 플러그인 기술을 사용하게 된 것이다. 당시에는 넷스케이프와 인터넷 익스플로러 양쪽에 모두 포팅 했었으나 브라우저 점유율 확대에 따라 지금은 액티브X만 남게 되었다. 플러그인은 자신의 PC에 다른 프로그램을 설치하는 것으로 보안 때문에라도 신중하게 설치 여부를 검토해야 하는 데도 인터넷 뱅킹, 사이버 트레이딩, 공인 인증 등 액티브X 프로그램을 보안 경고창의 "예" 버튼 한번으로 설치했던 경험으로 일반인들에게 거부감 없이 받아들여 지고 있다. 최근 인터넷 익스플로러의 보안 이슈와 악성 플러그인이 범람함에 따라 좀 더 신중하게 액티브X를 설치하는 사람들이 늘어나고 있기는 하지만.

대안 없는 무분별한 액티브X 사용은 그것이 자랑할만한 독자 기술이였다 하더라도 다시 특정 기술 플랫폼 종속적이 될 수 밖에 없다는 실례를 우라나라 공인 인증 기술에서 찾아 볼 수 있다. 99%가 사용하고 있으니 그것만 지원하는 것이 효율적이라는 생각에는 종속성이 커짐에 따라 증대하는 관성과 더딘 기술의 진보, 그리고 상상할 수 없는 비용의 증가를 예고하는 것이다. 얼마 전 윈도우XP 서비스팩2 출시에 따른 액티브X 설치 방법의 변경으로 인해 생긴 문제 때문에 우리 나라만 서비스팩2 출시를 몇 달 간 연기하고서 우리나라 모든 웹사이트들이 이 문제에 매달렸던 것만 봐도 알 수 있다. 그래서 운영 체제나 디바이스 플랫폼에 독립적인 데다 오픈 소스이기도 한 모질라 플랫폼에서 제공하는 대안 기술을 주목해야 하는 것이다.

ActiveX Plugin 기술

많은 사람들이 파이어폭스가 액티브X를 지원하면 많은 문제가 해결되는데 왜 그렇게 하지 않는지 의문을 가진다. 액티브X는 윈도우 종속적인 기술이고 모질라는 크로스 플랫폼을 지향하는 특성상 그렇게 할 수가 없다. 그러나 같은 기능을 두 가지로 개발하는 비용을 들이는데 비해 우선 사용자층이 두터운 윈도우 플랫폼에서 액티브X를 사용할 수 있게 한다면 또한 윈도우 개발자들도 모질라 플랫폼을 쉽게 가져다 쓸 수 있다면 서로를 이해하고 접근 하는 데 더 용이하지 않을까? 그래서 시작된 것이 바로 Mozilla ActiveX Project(http://www.iol.ie/~locka/mozilla/mozilla.htm)이다. 이 프로젝트는 성격상 모질라의 공식 프로젝트는 아니지만 자원 봉사로 꾸준히 업데이트 되고 있다.

이 프로젝트의 결과물은 크게 두 가지 부분으로 나누어 진다. 그 첫번째는 액티브X 플러그인(Plug-in For ActiveX controls) 기술이다. 이것은 기존의 모질라 플러그인 기술을 사용하여 윈도우에 설치된 액티브X를 감지하고 실행하게 해 주는 플러그인이다. 즉, 파이어폭스에서 액티브X를 실행할 수 있게 해준다. 이를 위해서는 먼저 인터넷 익스플로러와 파이어폭스가 플러그인을 인식하는 방법을 맞출 필요가 있다. 똑같이 표준 태그인 <object>를 사용하고 있지만 그 속성을 인식할 때 인터넷 익스플로러는 classid="CLSID:XXXX...", 파이어폭스는 type="application/x-oleobject" 등과 같은 방법을 사용한다. 액티브X 플러그인은 일단 classid를 인식하게 해준다. 그리고 clsid에 해당하는 액티브X를 호스팅(hosting)하여 실행한다. 만약 비표준 자바스크립트나 VBScript로 액티브X를 제어 하면 제대로 동작하지 않겠지만, 그렇지 않은 경우 대부분 잘 동작한다.

이 프로그램은 모든 액티브X가 실행되는 것은 아니다. 기본으로 윈도우 미디어 플레이어가 실행될 수 있게 만약 실행하고 싶은 액티브X가 있으면 파이어폭스 디렉토리 내에 activex.js 파일에 클래스 ID를 추가해 주어야 한다.

```
pref("general.useragent.vendorComment", "ax");
pref("security.xpconnect.activex.global.hosting_flags", 9);
pref("security.classID.allowByDefault", false);
pref("capability.policy.default.ClassID.CID6BF52A52-394A-11D3-B153-00C04F79FAA6", "AllAccess");
pref("capability.policy.default.ClassID.CID22D6F312-B0F6-11D0-94AB-0080C74C7E95", "AllAccess");
```

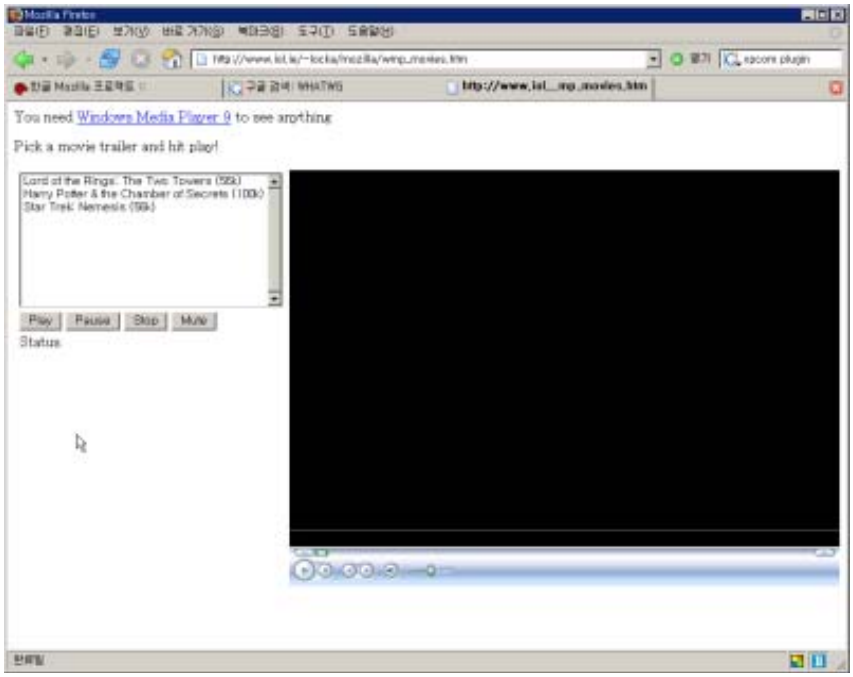
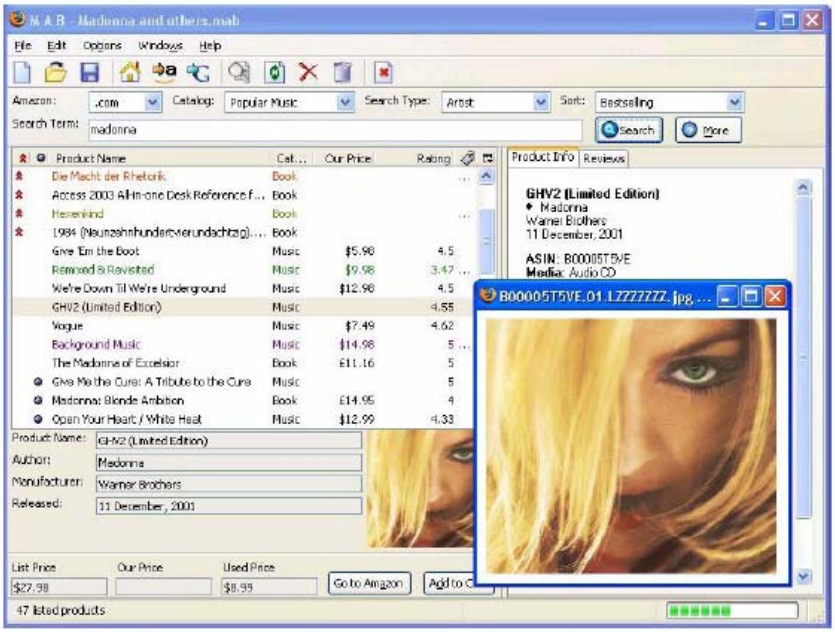


그림 36 파이어폭스에서 윈도우 미디어 플레이어 액티브X가 실행되는 모습

이렇게 하는 이유는 보안상의 문제로 사용자가 실행할 액티브X를 결정할 수 있도록 하기 위해서이다. 아직 테스트 프로젝트이므로 편리한 인터페이스가 필요하다는 생각이 든다.

XPCOM Plugin 기술

액티브X와 모질라를 이어주는 플러그인 기술은 그 자체로서 크로스 플랫폼을 지원하지 않기 때문에 매우 불안정한 대안이라고 밖에 볼 수 없다. 모질라가 가지고 있는 근본적인 대안은 앞서 계속해서 다루어 온 대로 XPCOM을 이용하여 모든 플랫폼을 동시에 지원하면서 XUL, 자바스크립트, CSS 등을 통해 인터넷 어플리케이션(Rich Internet Application)을 만드는 기술의 확대를 꾀하는 것이다. 예전 플러그인이 수행했던 외부와의 통신 기능도 XML-RPC, SOAP, XMLHTTP 등의 기술로서 이미 해결되었기 때문에 확장 기능 등에서 활용도는 매우 높아지고 있다.



Mozilla Amazon Browser: http://www.faser.net/mab/

그림 37 XUL로 개발한 아마존 서비스 브라우저

그러나, 그밖에 외부 기술을 도입하고자 한다면 모질라 내부에 구현해야 한다. XPCOM Plugin기술은 XPIDL로 정의한 인터페이스를 구현한 플러그인 API와 이를 호출하는 XUL 및 자바스크립트로 구성되어 자유 자재로 구현 가능하다. 각 API의 경우 각 OS별로 고려하고 컴파일 하는 수고를 제외하고는 크로스 플랫폼에서 훌륭하게 돌아갈 수 있다.

Flash (Flex)

Flex는 Flash라는 클라이언트 플러그인을 무기로 매크로 미디어에서 만든 서버와 클라이언트의 중간 개념인 미들티어(middle-Tier) 플랫폼이다. FLEX 는 개발자들이 플래시개발툴(Flash IDE) 없이도 태그로 간단하게 플래시를 만들수 있게 해준다

Flex는 Flash를 대체하는 게 아니라 그 기능의 확장으로서 flash로 FLEX에서 쓰일 컴포넌트를 개발 하게 될것이다. Flex는 리치 인터넷 어플리케이션(RIA) 나 온라인 프리젠테이션을 쉽고 간단하게 만들고자 하는 서버쪽 개발자를 위한 맞춤복 같은 솔루션이다. FLEX 는 쉬운 MXML을 사용하므로 기존의 ASP/JSP/Coldfusion 개발자이 Flash 를 사용하지 않고도 화면구성을 컨트롤 할수 있게 한다

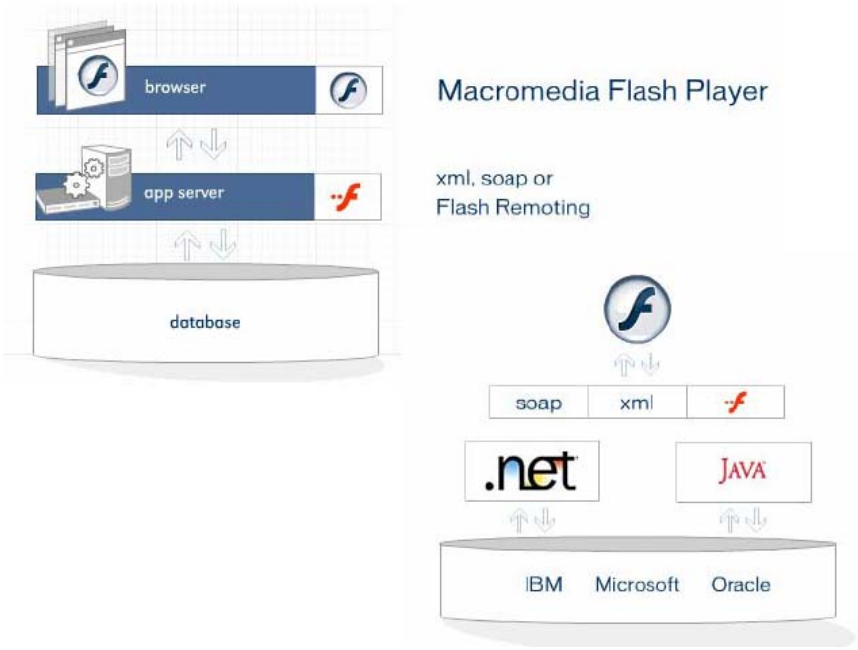


그림 38 Flex의 서비스 플랫폼 구조

FLEX의 개발 환경은 ASP/JSP/Coldfusion 등 외부 개발 환경을 사용할 수 있다. 즉, 비지니스 로직은 기존과 같이 ASP/JSP/Coldfusion 등이 담당하고, FLEX가 화면표현쪽을 담당하는 식으로 분리된다. Flex는 Flash로 가는 프로젝트에서 플래시로만 개발 했을때는 소스가 복잡해지는 것을 막는다. 이에 Flash 개발자는 컴포넌트 만들기에만 집중하고 서버측 개발자는 그 컴포넌트를 사용해 화면 구성을 한다면 상당히 깔끔한 플랫폼이 만들어질 가능성이 크다.

브라우저 내장 기술

Ajax

Ajax(Asynchronous JavaScript and XML)는 대화식 웹 어플리케이션의 제작을 위해 아래와 같은 조합을 이용하는 웹 개발 기법이다:

- 표현 정보를 위한 HTML (또는 XHTML) 과 CSS
- 동적인 화면 출력 및 표시 정보와의 상호작용을 위한 DOM, 자바스크립트
- 웹 서버와 비동기적으로 데이터를 교환하고 조작하기 위한 XML, XSLT, XMLHttpRequest (Ajax 어플리케이션은 XML/XSLT 대신 미리 정의된 HTML 이나 일반 텍스트, JSON, JSON-RPC를 이용할 수 있다).

DHTML이나 LAMP와 같이 Ajax는 자체가 하나의 특정한 기술을 말하는 것이 아니며, 함께 사용하는 기술의 묶음을 지칭하는 용어이다. 실제로 AFLAX와 같이 사실상 Ajax에 바탕을 두고 있는 유사/복합 기술들이 속속 나타나고 있다.

Ajax 어플리케이션은 실행을 위한 플랫폼으로 위에서 열거한 기술들을 지원하는 웹 브라우저를 이용한다. 이것을 지원하는 브라우저로는 모질라 파이어폭스, 인터넷 익스플로러, 오페라, 사파리 등이 있다. 단, 오페라는 현재 XSL 포맷팅 객체와 XSLT 변환을 지원하지

않는다.

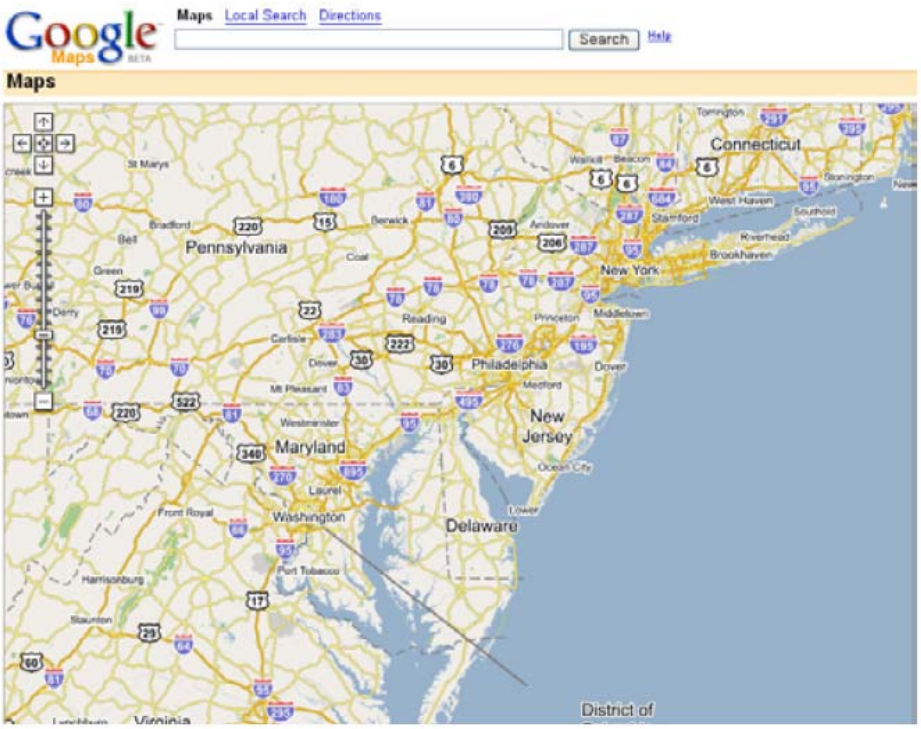


그림 39 Ajax로 구현한 Google Maps

기존의 웹 어플리케이션은 폼을 채우고 제출(submit)을 하면, 웹 서버로 요청을 보내도록 한다. 웹 서버는 전송된 내용에 따라서 새로운 웹 페이지로 결과물을 되돌려준다. 이때 둘 사이에 중복되는 HTML 코드로 인해 많은 대역폭이 낭비된다. 게다가 이러한 방식으로는 네이티브 어플리케이션에 비해 고도로 대화형 사용자 인터페이스를 작성하기가 힘들다.

반면에 Ajax 어플리케이션은 필요한 데이터만을 주도록 웹 서버에 요청할 수 있다. 보통 SOAP나 XML 기반의 웹 서비스 언어를 사용하며, 웹 서버의 응답을 처리하기 위해 클라이언트 쪽에서 자바스크립트를 쓴다. 그 결과로 웹 브라우저와 웹 서버 사이의 교환되는 데이터량이 줄어들기 때문에 어플리케이션의 응답성이 좋아진다. 요청을 주는 수많은 컴퓨터에서 이 같은 일이 일어나기 때문에, 전체적인 웹 서버 처리량도 줄어들게 된다..

DHTML 어플리케이션과 같이, Ajax 어플리케이션에서는 브라우저마다의 편차를 고려해서 엄격한 테스트가 이루어져야 한다. 이 기술로써 얻는 이득은 어플리케이션의 속도와 응답성의 개선이다

Canvas

Canvas는 모질라 게코 렌더링 엔진에 HTML 엘리먼트로 추가된 기능이다. 이 새로운 엘리먼트는 웹 컨텐츠 제공자가 웹페이지의 원하는 영역에 비트맵 혹은 벡터 그래픽을 그릴 수 있는 스크립트를 이용할 수 있도록 한다. 캔버스 엘리먼트는 웹 애플리케이션 1.0 스펙의 일부로, 웹 하이퍼텍스트 애플리케이션 기술 워킹 그룹(Web Hypertext Application Technology Working Group)이 만들었다. WHATWG는 월드 와이드 웹을 통해 풍부한

애플리케이션을 전달하기 위한 신기술을 개발하는 것을 목표로 하는 그룹이다. 모질라 재단, 오페라 소프트웨어, 애플 컴퓨터 등이 WHATWG의 회원이다. 캔버스 엘리먼트는 최초 애플 맥 OS X의 대시보드(Dashboard)에서 이용하기 위해 만들어졌다.

Gecko의 HTML 캔버스 지원에 대해서 이것이 W3C 표준이 아닌 만큼 논쟁의 여지가 있으나, WHATWG는 추후에 이를 표준화하기로 결정하였다. 일반적으로 WHATWG 기술은 HTML을 더 향상시키는 반면, W3C 표준은 선행적으로 XML을 향한 이전에 필요한 급진적인 변화의 경향이 있다. 몇몇 사람들은 이로 인해 두 그룹이 충돌할 수 있다고 주장한다. WHATWG의 실용주의적인 접근이 W3C의 광범위하고 복잡한 표준에 맞서 궁극적으로 웹 개발자들 사이에서 승리를 거둘지에 대한 논쟁이 일어나곤 한다.

그림 40 Canvas를 이용한 3D 게임

모질라 캔버스 구현은 크로스 플랫폼 Cairo 벡터 그래픽 라이브러리를 이용한다. Cairo는 모질라의 SVG(Scalable Vector Graphics)의 렌더링 백엔드로도 쓰이며 추후에 모질라의 그래픽 능력에 힘을 더해줄 계획을 갖고 있다. 작년 Cairo는 Mozilla Public License로 라이선스가 변경되었으며(더불어 LGPL로도), 모질라 재단의 요구에 편의를 도모하기 위한 것이다. 이 기능을 통해 화려한 데스크탑 효과를 만들기 위해 새로운 캔버스 엘리먼트와 XUL의 알파 채널 지원을 결합하였다.

웹 어플리케이션 표준화 동향

WHATWG

그간 웹 표준화 기구인 W3C(www.w3.org)를 통해 수 많은 웹 표준들이 제정되었음에도 불구하고 플러그인 기술 즉 웹 어플리케이션 기술에 대한 표준은 만들어 지지 못했다. 표준은 상호 운용성 및 개방성 측면에서 매우 중요하며 공정한 경쟁을 유도할 수 있는 제도적 장치이다. 이러한 필요성에 따라 W3C는 작년 4월 웹 어플리케이션에 대한 워크샵을 개최하고 이에 대한 의견을 나누었다. (http://www.w3.org/2004/04/webapps-cdf-

ws/)

이 워크샵에서 크게 쟁점은 마이크로소프트와 오페라/모질라 재단 연합이 발표한 웹어플리케이션 방향에 대한 것이다. 마이크로소프트는 XAML/아발론 등으로 대별되는 롱혼 전략과 자사가 제안한 CSS3에 대한 이야기만 한 반면, 오페라/모질라 연합은 기존의 웹 표준 기술을 활용한 중간 단계의 웹 어플리케이션 표준을 빨리 만들자는 제안을 하였다. 이에 대해 많은 참석자들은 부정적인 반응을 나타냈다. 이런 문제를 다룰 워킹 그룹이 아직 존재하지 않는다는 이유를 달았지만 이미 W3C는 데스크탑 환경에서 웹 표준 문제 보다는 오히려 모바일 같은 비PC 디바이스에서의 상호 운용성에 관심을 가진 사람들이 많았으며 그곳에 초점을 두고 있었기 때문이다.

이에 2004년 6월 오페라와 모질라의 Ian Hickson과 David Baron, 애플의 David Hyatt 등이 주축이 되어 W3C와 별도로 표준안을 만들기 위한 웹어플리케이션 기술 워킹 그룹(Web Hypertext Application Techology Working Group, http://whatwg.org) 을 조직하고 활동에 들어갔다. 이 워킹 그룹은 W3C 형식에 준하는 표준안 작업을 한 후, 향후 IETF나 W3C에 기초안(Draft)를 제안할 예정이다.

이 워킹 그룹이 작업 중인 표준안은 크게 세가지 영역으로 나누어져 있다. 그 중 가장 관심을 끄는 Web Applications 1.0은 어플리케이션 개발을 하기 위해 기존의 HTML을 확장하는 것이다. 흔히 HTML5라고 불리는 이 표준안은 XUL의 경험을 기초로 복잡한 XML을 사용하기 보다는 기존의 HTML을 확장 하여 UI를 만드는 것을 목표로 하는 것으로 기존의 파이어폭스 확장 기능보다도 더 쉽게 웹 어플리케이션을 만드는 방법을 제안할 것이다.

예를 들어 메뉴바를 만들기 위해서 아래와 같이 우리에게 익숙한 HTML태그와 그 확장 태그를 쓰겠다는 것이다.

```
<menubar>
 <li>
  <a href="#file">File</a>
  <menu id="file">
   <li><button type="button" onclick="fnew()">New...</button></li>
   <li><button type="button" onclick="fopen()">Open...</button></li>
   <li><button type="button" onclick="fsave()"
id="save">Save</button></li>
   <li><button type="button" onclick="fsaveas()">Save
as...</button></li>
  </menu>
 </li>
</menubar>
```

이 표준안은 XHTML, CSS 같은 기존 표준안과는 연동 하지만, XUL/XAML 과는 독립적으로 구성하겠다는 뜻을 가지고 있어 웹 표준으로서 기술을 주도하겠다는 생각을 가지고 있다. Web Forms 2.0은 HTML4.01의 Form 부분의 기능을 확장하는 작업으로 Web Application에서 사용하는 데이터 입력 및 출력 그리고 추가 등을 위해 기초적으로 필요한 작업이다. 따라서 이 표준안 작업은 꽤 많이 진척되어 작년 연말까지 거의 완성되게 되었다.

W3C에는 이와 유사한 IBM이 제안하여 표준안이 된 XForm이 있다. XForm은 기계들간의 데이터 교환 표준으로 삼고, WebForm은 사용자 인터페이스에 사용하게 하여 어플리케이션 개발자들이 이해하기 쉽도록 하자는 것이 이 표준안의 취지이다. 이 표준안이 W3C나 IETF에 받아 들여 진다면 비 마이크로소프트 진영의 브라우저 어플리케이션 개발에 보다 획기적인 플랫폼이 만들어질 것으로 예상된다.

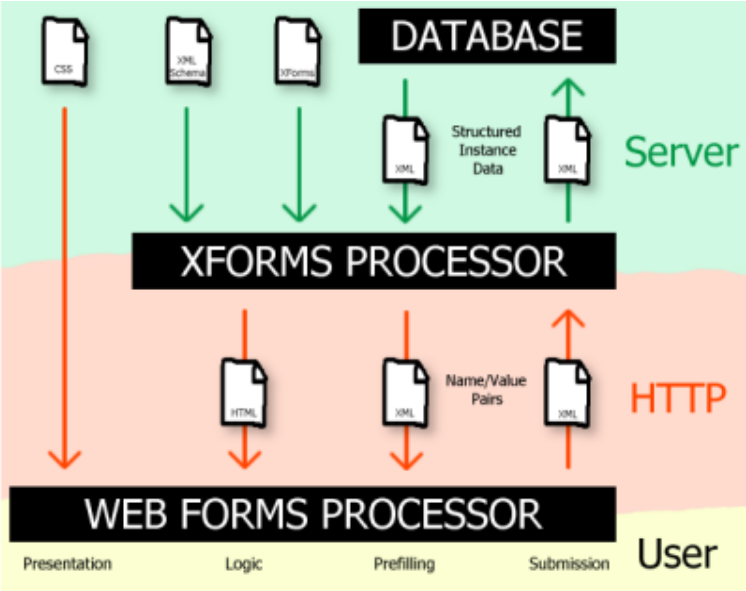


그림 41 XForm과 WebForm의 관계도

마지막으로 Web Controls 1.0은 새로운 WebForm과 UI에 대한 콘트롤과 외양 표시가 가능하도록 DOM과 CSS의 기존 표준을 확장하는 작업이다. 이 부분은 기존의 DOM/CSS의 객체 사용 방법이 어느 정도 성숙되어 있기 때문에 제일 마지막에 할 작업이 될 것이다.

표준은 표준으로만 존재하는 것이 아니라 구현 되었을 때 빛을 발하게 된다. 모질라와 오페라 애플이 만드는 새로운 표준은 웹 개방성과 상호 운용성 문제를 해결 할 수 있는 중요한 시발점이 될 것이다. 표준이 되더라도 과연 마이크로소프트가 이를 수용할 지 여부는 알 수 없지만 검증된 개발 방법론을 기초로 대안을 제시할 수 있다고 생각한다.

W3C Rich Web Application W/G

Ajax, Web2.0 등 리치 웹 어플리케이션에 대한 관심이 뜨거워 지지 W3C에서도 2005년 9월 워킹 그룹 활동 계획을 시작으로 하여 11월에 두개의 워킹 그룹을 공식적으로 설립하였다.

Robin Berjon (Expway)가 의장을 맡은 Web APIs Working Group은 AJAX나 XMLHttpRequest 같은 표준 어플리케이션 인터페이스를 표준화 하는 논의를 시작한다.

또한, Art Barstow (Nokia)가 의장을 맡은 Web Application Formats Working Group은 최근 XUL, XAML, MXML 등 각종 클라이언트 어플리케이션에 사용되는 XML 인터

페이스 언어와 XBL 등의 표준을 다룬다.

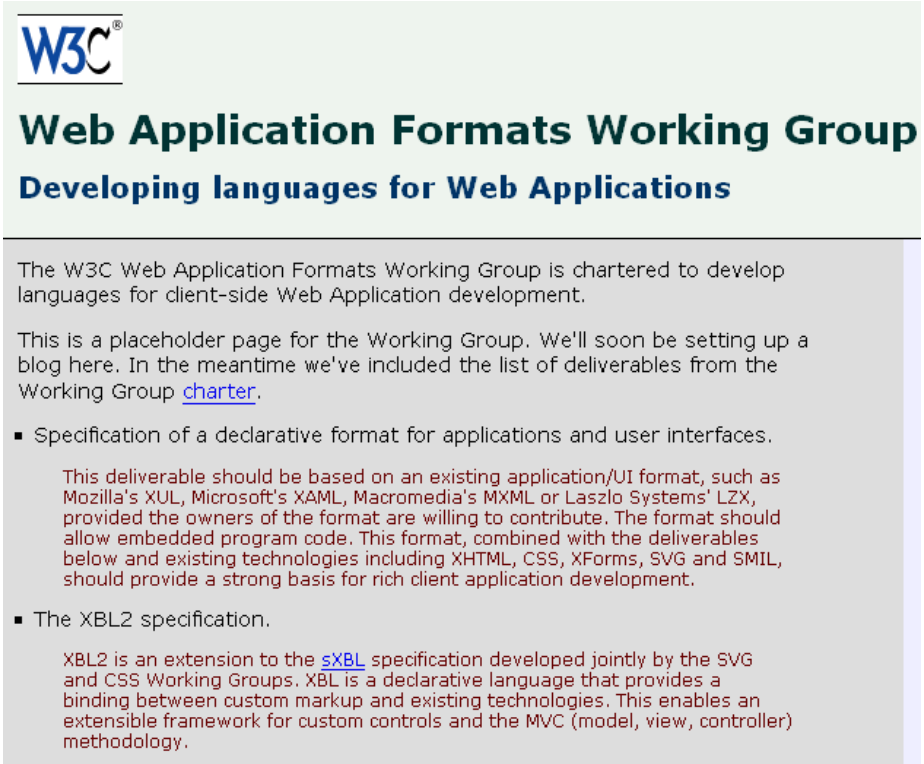


그림 42 W3C의 웹 어플리케이션 포맷 워킹 그룹

W3C에는 유력한 인터넷 브라우저 업체와 기술 업체가 모두 참여 하고 있기 때문에 향후 표준화 움직임에 따라 RIA 시장이 움직을 것으로 예상된다. 따라서, 브라우저 플러그인 대용 기술에 대한 다양한 고민과 표준화의 방향에 따라 움직일 필요가 있다.

실전 표준 웹 프로그래밍

표준 MIME 타입 설정

MIME 타입은 인터넷 상의 파일 교환에서 - 메일을 통해서이든 웹을 통해서이든 - 파일 송신자가 수신자에게 '지금 보내는 파일은 이러이러한 형식이다'라고 알려 주기 위해 사용한다. 파일 수신자가 그 파일을 온전히 해석하도록 하기 위해서 송신지가 꼭 전해 주어야 하는 매우 중요한 정보이므로, 웹 서버 관리자나 웹 개발자들은 올바로 이 정보를 전달하기 위한 방법을 숙지하고 있어야 한다.

인터넷 익스플로러로 대표되는 몇몇 브라우저는 송신자가 보내는 MIME 타입 정보를 무시하고, 파일의 첫 부분을 직접 들여다 보고 파일 타입을 스스로 결정한다. 하지만, 이런 방법은 인터넷 표준에 부합하는 것이 아니므로, 모든 웹 클라이언트가 지원하지 않다. 따라서, 장치 독립적이고, 상호 운용성 있는 웹 사이트를 구축하려면, 웹 서버 관리자와 웹 개발자는 파일 송신 시에 필히 송신하려는 파일에 맞는 MIME 타입을 지정해서 보내야 한다.

웹 서버가 내보내는 MIME 타입을 확인하기 위해서는 HTTP 헤더 엿보기 프로그램이나 엿보기 사이트를 이용할 수 있다. 검색 엔진에서 'HTTP sniffer'라고 치면 이 서비스를 제공하는 곳이나 프로그램을 쉽게 찾을 수 있다. 예를 들어, SniffURI.org에 가서 HTTP 헤더의 내용을 알고 싶은 파일의 URL을 치면 그 파일을 제공하는 웹 서버가 내보내는 응답(response) 내용(HTTP 헤더를 포함한)을 모두 볼 수 있다. 그 가운데에서 Content-Type으로 시작하는 줄을 보십시오. 아래는 http://www.w3.org/StyleSheets/home.css를 SniffURI에 치고 얻은 결과 중 일부를 보인 것이다. Content-Type으로 시작하는 줄에서 text/css라는 값을 서버가 돌려주었다.

```
HTTP/1.1 200 OK
Date: Mon, 12 Dec 2005 11:53:28 GMT
Server: Apache/1.3.33 (Unix) PHP/4.3.10
P3P: policyref="http://www.w3.org/2001/05/P3P/p3p.xml"
Expires: Mon, 12 Dec 2005 17:53:28 GMT
Last-Modified: Mon, 16 May 2005 15:59:15 GMT
.... 생략 ...
Content-Length: 1545
Content-Type: text/css
```

MIME 타입을 잘 지정하는 것이 중요한 보기를 두 가지만 들겠다. 국내 유수의 어떤 신문사 웹 사이트에서는 iframe으로 불러 들이는 html 파일에 'inc'라는 확장자를 붙여 놓았다. 그런데, 그곳에서 쓰는 웹 서버는 따로 MIME 타입을 설정해 놓지 않은 확장자가 붙은 파일은 모두 application/octet-stream으로 처리하도록 되어 있다. 따라서, inc가 붙은 파일을 iframe에서 불러 들일 경우 HTTP 표준을 잘 준수하는 브라우저(예를 들어, firefox)는 파일을 저장하라는 (application/octet-stream은 바이너리 실행 파일로 브라우저에서 처리할 수 없으므로), 대화 상자가 뜬다.

이 경우 'inc'라는 확장자를 'html'로 바꾸거나 (html이란 확장자는 거의 대부분의 웹 서버에서 text/html로 이미 설정되어 있다.), 'inc'라는 확장자를 아래에서 설명하는 방법에 따라 'text/html'로 대응시켜 주어야 한다. 또, 어떤 웹 사이트를 firefox로 보면 html 소

스가 그대로 드러나는 경우가 있다. 이것은 MIME 타입을 지정하지 않은 경우 무조건 'text/plain'으로 처리하도록 웹 서버가 설정되어 있기 때문이다. 이 경우에도 'text/html'로 모든 웹 클라이언트가 처리하도록 하려면, 써버가 이 파일에 대해 'text/html'이라고 클라이언트에게 알리도록 설정을 고쳐야 한다. 또, 상당수의 국내 웹 서버가 CSS에 대해 MIME type을 지정해 놓지 않아서 application/octet-stream 혹은 text/plain을 서버가 내보내고 있다. 이 역시 text/css로 지정해 두어야 한다.

본래 인터넷 메일을 위해서 개발되었지만(RFC 2045-2049), 그 뒤에 인터넷 메일과 (RFC 2821/2822, RFC 2045-2049)과 웹(HTTP 1.x) 상의 정보 교환에서 공히 쓰이고 있다. 두 경우 모두 'Content-Type'이라는 헤더 필드를 통해 다음과 같이 지정한다. 웹 상에서 파일 교환을 할 때에는 HTTP 헤더를 통해 그 헤더 필드를 내보내고, 메일에서는 메일 헤더에 그 헤더 필드를 내보내야 한다. 다음은 몇 가지 보기이다.

```
Content-Type: type/subtype; param1=value1; param2=value2
```

MIME 타입	확장자	파라미터	설명	HTTP header
text/html	html, htm		HTML (인코딩은 문서 내부에서 지정)	Content-Type: text/html
text/html	html, htm	charset=EUC-KR	HTML (EUC-KR 인코딩)	Content-Type: text/html; charset=EUC-KR
text/html	html, htm	charset=UTF-8	HTML (UTF-8 인코딩)	Content-Type: text/html; charset=UTF-8
text/css	css		CSS 스타일시트(인코딩은 파일 내부에서 지정)	Content-Type: text/css
application/xml	xml		XML 문서	Content-Type: application/xml
text/plain	txt		일반 텍스트 파일	Content-Type: text/plain
text/rtf	rtf		Rich Text Format	Content-Type: text/rtf
text/javascript	js		ECMAscript/Javascript	Content-Type: text/javascript
image/png	png		PNG 그림 파일	Content-Type: image/png
application/pdf	pdf		Adobe PDF	Content-Type: application/pdf
application/vnd.ms-excel	xsl		엑셀 파일	Content-Type: application/vnd.ms-excel
application/vnd.sun.xml.writer	sxw		오픈/스타 오피스 워드 프로세서 파일	Content-Type: application/vnd.sun.xml.writer
application/octet-stream	exe		바이너리 실행 패일	Content-Type: application/octet-stream

audio/mpeg	mp3	mp3	Content-Type: audio/mpeg
audio/x-mp3	mp3	mp3	Content-Type: audio/x-mp3
video/x-ms-wmv	wmv	윈도우 미디어 비디오	Content-Type: video/x-ms-wmv
application/x-shockwave-flash	swf	플래시	Content-Type: application/x-shockwave-flash
audio/x-realaudio	ra	리얼 미디어	Content-Type: audio/x-realaudio

아직 IANA (인터넷에서 쓰는 여러 가지 번호나 형식 등을 등록하는 곳)에 공식으로 등록되지 않은 파일 타입에 사용한다. MIME 타입 가운데 'text/*'(application/javascript 등 일부 'application/*'에도)에는 'charset'이란 parameter를 써서 문서에 쓰인 문자 인코딩(character encoding)을 지정할 수 있다. 이에 대한 자세한 내용은 아래 문자 인코딩 설정을 참고하면 된다.

Apache에서 설정 방법

Apache 웹 서버에서 MIME 타입을 지정하는 방법에는 몇 가지가 있다. 다음 두 문서를 참고하십시오.

- http://httpd.apache.org/docs/2.1/mod/mod_mime.html
- http://httpd.apache.org/docs/1.3/mod/mod_mime.html#addtype

Apache 1.x나 2.x에서 공히 IANA에서 등록된 지 충분한 시간이 지난 (최근에 등록된 경우에는 아직 Apache에 반영되지 않았을 가능성이 있다.) MIME 타입에 대해서는 mime.types 파일에 모두 나열되어 있으므로, 흔히 쓰이는 확장자를 IANA에 등록된 흔히 쓰이는 MIME 타입에 대해 쓴다면 특별히 다른 일을 할 필요가 없다. (mime.types의 위치는 써버 설치 방법에 따라 다르다.

/etc/, /etc/httpd/conf, /usr/local/etc/httpd/conf 등에 있을 것이다. Apache의 httpd.conf에서 TypesConfig 디렉티브를 써서 다른 곳에 있도록 지정할 수도 있다.) 하지만, IANA에 등록되지 않았지만, 흔히 쓰이는 파일 형식 (예를 들어, mp3, Windows Media, HWP, real audio 등)을 제공하는 경우나 흔히 쓰지 않는 확장자를 사용하는 경우(위에서 보기로 든 'inc'를 'text/html'에 쓰는 경우)에는 아래에서 설명한 방법에 따라 따로 지정해 주어야 한다.

서버 전체에서 지정하기

이 방법은 서버 관리자 권한이 있는 경우에만 쓸 수 있다. 두 가지 방법이 있다. 첫째는 'mime.types' (TypesConfig 디렉티브가 가리키는) 파일에 다음과 같은 줄을 더하는 것이다. (물론, 이미 이런 항목이 있다면 더할 필요가 없겠지요.) Apache 문서에서는 이 중 두 번째 방법을 추천하고 있다.

```
application/xml xml
```

```
application/xhtml+xml xhtml
application/javascript js
application/ecmascript es
text/css    css
video/x-ms-asf asf asx
audio/x-ms-wma wma
video/x-ms-wmv wmv
application/x-pn-realaudio ram rm
application/x-realaudio ra
application/ogg        ogg
application/x-hwp      hwp
# 보기로 든 모 신문사 웹 서버에는 다음과 같은 설정을 더해 주어야 한다.
text/html              inc
```

두번째는 웹 서버 설정 파일인 httpd.conf에 다음과 같은 줄을 더하는 것이다. 이 경우에 변경 후에 웹 서버를 다시 시작해야 한다. AddType 디렉티브를 써서 더한 내용이 mime.types 파일에 있는 내용보다 더 우선 순위가 높다는 데에 유의해야 한다.

```
# XML 에는 다른 MIME 타입을 쓸 수도 있지만, application/xml 권장
AddType application/xml .xml

# XHTML 문서에 사용할 수 있는 MIME 타입은 'text/html'을
# 비롯해서 이외에 몇 가지가 더 있음.
AddType application/xhtml+xml .xhtml

# Javascript, ECMAscript
AddType application/javascript .js
AddType application/ecmascript .es
# CSS
AddType text/css    .css
# Windows Media format
AddType video/x-ms-asf .asf .asx
AddType audio/x-ms-wma .wma
AddType video/x-ms-wmv .wmv

# Real Media 관련 포맷
AddType application/x-pn-realaudio .ram .rm
AddType application/x-realaudio .ra
AddType application/ogg        .ogg
AddType application/x-hwp      .hwp
AddType application/octet-stream .rar

# 보기로 든 모 신문사 웹 서버에는 다음과 같은 설정을 더해 주어야 한다.
AddType text/html              .inc
```

웹 사이트 별로 지정하기

AddType을 써서 가상 호스트 별로 서로 다르게 지정할 수 있다.

디렉토리 별로 지정하기

서버 설정 파일 내에서 Directory로 묶인 부분에서 AddType을 써서 지정하거나, MIME 타입을 더하고자 하는 디렉토리 계층의 최상위 디렉토리에 '.htaccess' 파일을 만들고, 거기에 AddType 을 써서 더할 수 있다. 후자의 경우에는 서버 관리자 권한이 없는 경우에도 MIME 타입 설정을 변경할 수 있으므로, 웹 호스팅 서비스를 쓰는 경우에 특히 유용할 것이다. 단, 이 경우에 서버 관리자가 서버 설정에서 .htaccess 파일에서 FileInfo 관련 값을 변경할 수(AllowOverride) 있도록 설정해 주어야 한다.

기본 MIME 타입 지정

어떤 버전의 Apache 서버의 경우 따로 MIME 타입을 지정하지 않은 확장자에 대해 기본 MIME 타입(DefaultType)으로 text/plain을 쓰도록 설정되어 있는 경우가 있다. 서버에서 전송할 모든 파일 타입에 대해 MIME 타입 지정을 해 놓은 경우에는 별 문제가 없지만, 그렇지 않은 경우 바이너리 파일이 브라우저 창에 뿌려지는 일이 생길 수 있다. 이런 일을 방지하려면 DefaultType을 application/octet-stream으로 설정해야 한다. 이 설정은 서버 전체의 설정 파일, 가상 호스트 설정 파일과 디렉토리별 설정 파일인 .htaccess에서 할 수 있다.

IIS에서 설정 방법

다음은 마이크로소프트 테크넷의 아래 주소에서 가져온 것이다.
http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/cd72c0dc-c5b8-42e4-96c2-b3c656f99ead.mspx

웹 서버 전체에 대해 특정 확장자를 갖는 파일에 대해 특정 MIME 타입을 지정하려면 다음과 같이 한다.

1. IIS 관리자에서 'local computer'를 클릭하고, 다시 '속성'을 선택한다.
2. MIME 타입 단추를 누릅니다
3. '새로 만들기'를 누릅니다.
4. '확장자' 상자에 확장자 이름 (예를 들어, 'css', 'inc', 'hwp')을 입력한다.
5. 'MIME 타입' 상자에 해당 파일의 MIME 타입을 입력한다. 예를 들자면, 'text/css', 'text/html', 'application/x-hwp' 등이다.
6. '확인'을 누릅니다.
7. 웹 써버 전체가 아니라 웹 서버가 호스팅하는 일부 웹 사이트나 일부 디렉토리에 대해서만 MIME 타입을 더하려면 다음과 같이 한다.
8. IIS 관리자에서 MIME 타입을 더하고자 하는 웹 사이트나 디렉토리를 골라서 오른쪽 마우스 버튼을 누르고, 거기에서 '속성'을 선택한다.
9. HTTP 헤더 탭을 클릭한다.
10. MIME 타입을 클릭한다
11. 새로 만들기를 클릭한다
12. '확장자' 상자에 확장자 이름 (예를 들어, 'foo')을 입력한다.
13. 'MIME 타입' 상자에 해당 파일의 MIME 타입을 입력한다. 예를 들자면, 'text/plain', 'text/html', 'application/x-foo' 등이다. 이미 상위 수준(예를 들어 웹 서버 전체)에서

정의한 MIME 타입을 더하려고 할 경우 지금 더하는 정의를 어느 수준에서부터 적용할 것인지 정하라는 질문에 답해야 한다.

14. '확인'을 누릅니다.

설정해 놓은 MIME 타입을 제거하기 위해서는 다음과 같이 한다.

1. IIS 관리자에서 MIME 타입을 제거하고자 하는 웹 사이트나 디렉토리를 골라서 오른쪽 마우스 버튼을 누르고, 거기에서 '속성'을 선택한다.
2. HTTP 헤더 탭을 클릭한다.
3. MIME 타입을 클릭한다
4. 등록된 MIME 타입 목록에서 제거하고자 하는 타입을 클릭하고, '제거하기'를 누릅니다.
5. '확인' 단추를 세 번 누릅니다.

PHP에서 설정 방법

CGI에서 사용되는 다양한 프로그램 언어들이 있지만 PHP를 예를 들어 다른 언어에 적용할 수 있을 것이다. http://kr.php.net/header에서 자세히 설명하고 있는 header() 함수를 써서 Content-Type을 지정해 주어야 한다.

```
<?php
// PDF 파일을 전송할 경우
header('Content-type: application/pdf');

// 이름은 'my resume.pdf'
// Content-Disposition 헤더 필드의 filename 파라미터에 쓸 파일 이름
// 중간에 공백이 있을 경우 따옴표로 꼭 묶어야 함.
header('Content-Disposition: attachment; filename="my resume.pdf"');

// 서버 측에 있는 PDF 문서의 이름은  original.pdf
readfile('original.pdf');
?>
<?php
// UTF-8 로 된  text/html 파일을 보낼 때
// 이 경우 이후에 따라 나오는 HTML 파일에서 meta tag 을 써서 문자 인코딩을
// 지정할 필요는 없음. 지정하는 경우에는 header 에서 지정한 값이 우선임을
// 명심할 것. MS IE 는 이 표준을 무시하고, html 파일 내부의 값에 우선 순위를
// 두므로, 브라우저 독립성을 위해서 둘 사이에 모순이 없도록 유의할 것.
header('Content-type: text/html; charset=UTF-8');
....

?>
```

Apache Tomcat에서 설정 방법

다른 JSP/Servlet container도 비슷한 방식을 사용할 것이다.

정적 내용물(Static Contents)

conf 디렉토리 아래에 있는 web.xml 파일에 보면 확장자들을 MIME 타입에 대응시켜 놓은 다음과 같은 부분을 볼 수 있다.

```
<mime-mapping>
  <extension>zip</extension>
  <mime-type>application/zip</mime-type>
</mime-mapping>
<mime-mapping>
  <extension>css</extension>
  <mime-type>text/css</mime-type>
</mime-mapping>
```

Apache에서 배포하는 Tomcat에는 IANA에 등록되지 않은 wma, wmv, rar, hwp 등에 대한 MIME 타입 대응을 해 놓지 않았으므로, 이들에 대해 위와 비슷한 방식으로 web.xml 파일에 추가한 후, 서버를 다시 시작해야 한다. HWP의 경우 application/x-hwp로 할 경우 다운로드가 안 될 수도 있다고 한다. 그 경우에는 application/octet-stream을 대응시켜야 한다.

```
<mime-mapping>
  <extension>hwp</extension>
  <mime-type>application/octet-stream</mime-type>
</mime-mapping>
<mime-mapping>
  <extension>wma</extension>
  <mime-type>audio/x-ms-wma</mime-type>
</mime-mapping>
```

동적 내용물(Dynamic Contents)

JSP에서는 다음과 같은 식으로 파일 선두에서 MIME 타입을 지정할 수 있다. 서블렛 엔진이 생성해서 외부로 전송하는 파일의 문자 인코딩은 @page 디렉티브의 contentType에서 지정한다. JSP 소스 파일의 인코딩은 pageEncoding으로 지정한다. 따로 pageEncoding을 지정하지 않으면 contentType에서 지정한 인코딩을 JSP 소스 파일 인코딩으로 가정한다. 반대로 pageEncoding만 지정하면 contentType에서 charset을 지정하지 않으면 외부로 전송하는 파일의 인코딩도 pageEncoding의 값이 된다.

```
<%@page contentType="text/css" %>
<%@page contentType="text/plain" %>
<%@page contentType="text/html;charset=UTF-8" %>
<%@page contentType="text/html;charset=EUC-KR" %>
<%@page pageEncoding="EUC-KR" contentType="text/html;charset=UTF-
8" %>
<%@page pageEncoding="UTF-8" contentType="text/html;charset=EUC-
KR" %>
```

3(4)행은 JSP 소스 파일을 UTF-8 (EUC-KR)로 해석하고 HTML 파일도 UTF-8(EUC-KR)로 내보냅니다. 5(6)행은 JSP 소스 파일은 pageEncoding에서 지정한 EUC-KR(UTF-8)로 해석하고, 외부로 내보내는 HTML 파일은 UTF-8(EUC-KR)로 인코딩한다. 처음 두

경우에는 서버의 기본 설정 값인 ISO-8859-1을 문자 인코딩 값으로 사용하므로, 그렇지 않은 경우에는 꼭 명시적으로 지정해 주어야 한다.

어느 경우에나 JSP 소스 파일의 실제 인코딩이 JSP 엔진이 생각하는 문자 인코딩과 일치하도록 주의를 기울여야 한다. *항상 UTF-8을 쓰기를 강력히 권장한다.* JSP를 쓸 경우에는 PDF, zip 파일 등 바이너리 파일은 내보낼 수 없고, 오직 text 파일만 내보낼 수 있다. 바이너리 파일을 내보낼 때에는 servlet을 써야 한다.

Servelet에서는 setContentType을 써서 MIME 타입을 지정할 수 있다.

```
//response 가 ServletResponse type object 일 때
response.setContentType("text/html; charset=UTF-8");
response.setContentType("text/plain; charset=EUC-KR");
response.setContentType("application/pdf");
response.setContentType("application/x-hwp");
response.setCharacterEncoding("UTF-8"); // Java servlet 2.4 이후에서
```

Perl/Python 등으로 쓴 CGI

CGI 스크립트에서는 파일의 본체 전송을 시작하기 전에 HTTP 헤더 필드를 차례로 print 문이나 그와 동등한 방법으로 출력해 주면 된다. 단, HTTP 헤더 전송이 다 끝나고, 파일 본체 전송을 시작하기 전에 빈 줄을 하나 넣어 주어야 한다는 사실을 잊지 말아야 한다.

```
print "Content-Type: text/html; charset=UTF-8\n";
print "Content-Type: application/pdf\n";
print "Content-Type: application/x-hwp\n";
print "Content-Type: application/xhtml+xml\n";
```

표준 문자 인코딩 지정

텍스트 형식의 파일(HTML, XML, CSS, Javascript, RTF, 일반 텍스트 등)을 송신하는 서버는 그 파일의 MIME 타입과 아울러 그 파일이 어떤 문자 인코딩으로 쓰인 것인지를 수신 측에 알려 주어야 한다. 이 값이 없이는 문서를 수신 측에서 올바로 해석할 수 없기 때문에 HTML, XML, CSS 문법 검사 프로그램은 이 값이 제대로 설정되지 않은 경우 유효하지 않은 문서로 간주한다. 문자 인코딩을 수신 측에 알려 주는 방법에는 크게 다음 3가지가 있다.

- HTTP 헤더의 Content-Type에 charset parameter를 더해서 지정하는 방법. 아래는 몇몇 보기이다.

```
Content-Type: text/html; charset=UTF-8
Content-Type: text/html; charset=EUC-KR
Content-Type: text/css; charset=UTF-16LE
Content-Type: text/xml; charset=UTF-8
```

- Content-Type: application/javascript; charset=UTF-8
- 파일 형식 고유의 내부적인 방법. HTML, XML, CSS 등은 이런 방법을 쓸 수 있다.
- 현재 문서를 참조한(refer to) 문서와의 관계로부터 현재 문서의 인코딩을 결정. 예를 들어, HTML 문서에서 CSS 파일을 참조한 경우 앞선 두 가지 방법으로 인코딩을 결정할 수 없을 경우, CSS 파일은 HTML 문서의 인코딩이라고 간주한다. Javascript/ECMAscript나 text/plain처럼 문서 형식에서 두번째 방법을 지원하지 않는 경우에도 (Unicode에 기반을 둔 몇몇 문자 인코딩은 Byte Order Mark를 써서 이런 경우에도 문서 인코딩을 결정할 수 있기도 한다.) 첫번째 방법을 통한 인코딩 지정이 없으면, 이 방법이 쓰이다.

웹 서버 설정을 변경하여 *모든 정적인* 문서에 대해 HTTP 헤더로 *항상 같은* charset 값을 내보내도록 하는 것은 좋은 생각이 아닙니다. 왜냐하면, 그렇게 할 경우 HTTP 헤더에서 지정한 값의 우선 순위가 문서 내부에서 지정한 값보다 더 높아서 개별 문서별로 문자 인코딩을 지정할 수 없기 때문이다. MS IE는 HTTP 표준을 어기고 HTML 문서의 경우, HTTP 헤더에서 지정한 값보다 문서 내부에서 지정한 값에 더 높은 우선 순위를 부여한다.

하지만, 대부분의 다른 웹 브라우저는 문서 내부에서 지정한 값과 HTTP에서 지정한 값이 다를 경우 전자를 따릅니다. 따라서, 문서 형식이 제공하는 인코딩 지정 방법을 사용해서 문서 내부에서 지정하는 것이 많은 경우에 더 좋다. 하지만, PHP, CGI, JSP, ASP 등으로 *동적*으로 생성하는 문서에서는 서버 측 프로그램에서 각 문서 송신에 앞서 HTTP 헤더를 동적으로 변경할 수 있으므로, HTTP 헤더를 통해 문자 인코딩을 지정하는 것이 더 좋다.

단, XML 문서의 경우에는 문서 내부에서 지정하는 것을 권장한다. 또, 대부분의 경우에 HTTP 헤더와 문서 내부 모두에서 지정하는 것은 불필요한다. 하지만, 그렇게 하는 것을 권장하는 의견도 있다. 웹 서버를 *통하지 않고* 문서를 볼 경우(예를 들어, CD-ROM이나 디스크에 있는 문서를 볼 경우)에 대비하여 문서 내부에서 문자 인코딩을 지정해 놓는 것은 좋은 생각이다. 그렇게 할 경우에는 두 값이 *일치*하도록 주의를 기울여야 한다.

한국어 문서를 제공하는 웹 사이트에서 흔히 쓰는 문자 인코딩은 EUC-KR이다. 마이크로소프트 기반 제품에서는 "ks_c_5601-1987"을 쓰기도 한다만, 이것은 올바른 이름이 아닙

니다. EUC-KR은 2byte로 표현할 수 있는 한글 음절의 수가 2350자로 제한되어 있다. 따라서,' ',' ',' ' 등의 글자를 표현하기 위해서는 8byte를 써야 한다.

하지만, KS X 1001 부록 3에서 규정한 이 방법은 mozilla firefox 등 gecko 기반 브라우저만 지원한다. 따라서, *모든* 브라우저에서 현대 한국어의 모든 음절을 불편 없이 쓰기 위해서는 유니코드에 바탕을 둔 인코딩 방법인 UTF-8, UTF-16LE (LE는 Little Endian. 일부 Windows 기반 프로그램에서 'Unicode'라고 부르는 인코딩 방법은 실제로는 UTF-16LE이다), UTF-16BE (BE는 Big Endina) 등을 써야 한다. 몇 년 전과는 달리 UTF-8, UTF-16 등을 지원하는 문서 편집기(Windows XP에서는 노트 패드나 워드 패드도 지원)와 웹 저작 도구 (예를 들어, Dreamweaver, Nvu, MS FrontPage 등)를 쉽게 구할 수 있다.

또한, Oracle, DB2 등 상용 DBMS는 물론이고, MySQL, Postgres 등 open source DB도 UTF-8을 잘 지원하며, Linux의 기본 인코딩도 UTF-8이다. 따라서, *UTF-8 (혹은 UTF-16)을 사용할 것을 강력하게 권고한다.*. HTML 문서에 UTF-8을 사용하면 한글 이름을 지닌 파일을 HTML 문서에서 참조할 때 EUC-KR 문서에서 하듯이 한글 부분을 %-escape하지 않아도 된다.

즉, EUC-KR 문서에서는 'http://www.example.com/%B0%A1%B0%A2.jpg'라고 *해야 하지만*, UTF-8 문서에서는 'http://www.example.com/가각.jpg'라고 할 수 있다. 이외에도 한국어가 아닌 다른 언어를 지원하거나 (예를 들어, 한국인을 위한 중국어/일본어/러시아어 사전이나 언어 교육용 웹 사이트), 장차 해외 시장으로 진출할 때 유니코드 기반으로 작업하면 훨씬 편리한다.

문자 인코딩 지정에 대한 정보와 유니코드 지원 도구에 대해서는 웹 페이지와 거기에서 언급하고 있는 관련 페이지를 참고하면 된다.

- http://www.w3.org/International/O-HTTP-charset : HTTP에서 문자 인코딩 지정 방법
- http://www.w3.org/International/O-charset.html : XML과 (X)HTML에서 문자 인코딩 지정 방법
- http://www.alanwood.net/unicode/index.html : 유니코드를 지원하는 각종 도구 (편집기, 워드 프로세서 등), 글꼴, 브라우저 등에 대한 자세한 정보를 얻을 수 있다.
- convmv (http://osx.freshmeat.net/projects/convmv/?branch_id=39772&release_id=144059) : 파일 시스템 인코딩 변환 도구. 리눅스에서 EUC-KR 파일 시스템 인코딩을 쓰고 있다면, 이 도구를 써서 UTF-8로 변환할 수 있다. 이 도구는 파일 이름을 변경하는 도구이다. 파일 내용의 문자 인코딩 변경은 iconv 등을 쓰거나 위에서 언급한 문서 편집기, 저작 도구를 사용하십시오.
- fileiri (http://www.w3.org/2003/06/mod_fileiri/) : Linux/Unix에서 설치된 Apache 2.x용 파일 시스템 인코딩 실시간 변환 모듈. convmv로 파일 시스템 인코딩을 변환하는 대신 사용 가능. 참고 문헌에 있는 '국제화된 도메인 이름과 리소스 식별자 튜토리얼'도 참고하기 바랍니다.
- PuTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/) : Windows용 무료 SSH 클라이언트로 UTF-8을 매우 잘 지원한다. 컬럼비아 대학에서 만든 Kermit95도 잘 지원하지만, 유료이다.

문서에서 지정

문서 형식에 따른 문자 인코딩의 결정의 우선 순위와 내부에서 지정하는 방법은 다음과 같다.

HTML

HTML의 경우 meta tag을 써서 다음과 같이 문자 인코딩을 내부에서 지정할 수 있다. 이때, 꼭 문자 인코딩을 지정하는 meta 태그가 head의 *맨 처음* 요소가 되도록 해야 한다. 많은 웹 사이트에서 title 다음에 문자 인코딩을 지정하고 있다.

대부분의 경우 별 문제가 없지만, 매우 미미할지라도 좀더 빠른 웹 페이지 렌더링을 원한다면 꼭 표준에 따라 문자 인코딩을 지정하는 meta를 head의 *맨 처음* 요소로 넣으시기 바랍니다. 특히, 문자 인코딩을 지정하는 meta의 위치가 파일 선두에서 지나치게 떨어진 곳에 있을 경우 브라우저가 이 값을 감지하지 못 할 가능성이 높으므로 그렇게 해서는 절대 안 된다.

```
<html lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>문서 제목</title>
... 다른 헤더 태그들 : style, script, link 등 ..
</head>
```

문자 인코딩 결정의 우선 순위는 다음과 같다.

HTTP 헤더를 통해 전송된 Content-Type 필드의 "charset" 파라미터

문서 내부에서 meta로 지정한 값

대부분의 브라우저가 이외에도 참조한 문서의 인코딩 값, 웹 브라우저의 기본값, 인코딩 자동 감지 등의 방법을 채용하고 있다. 하지만, 명시적으로 문자 인코딩을 지정하면, 이런 방법의 구현 여부에 관계 없이 항상 문서가 원하는 방식으로 웹 브라우저나 다른 웹 클라이언트에 의해 해석될 것이다.

XML

모든 XML 문서는 문서의 최선두에서 다음과 같은 방식으로 문자 인코딩을 선언해야 한다. 단, 'encoding="UTF-8"' 부분이 없고, 문자 인코딩에 대한 다른 외부 정보가 없는 경우에는 기본으로 UTF-8을 가정한다.

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML 문서가 HTTP를 통해 전송될 경우 문자 인코딩 결정의 우선 순위는 다음과 같다.

HTTP 헤더를 통해 전송된 Content-Type 필드의 "charset" 파라미터. 단, Content-Type으로 text/xml을 쓰면서 charset 파라미터를 생략하면 US-ASCII가 기본값이 되고, US-ASCII 범위 밖의 문자를 포함한 모든 XML 문서가 유효하지 않게 되므로, charset을 생략하지 않도록 주의해야 한다. application/xml을 MIME 타입으로 쓸 경우 이런 문제가 없다.

BOM(Byte Order Mark) 혹은 문서 최선두의 XML 선언에서 지정한 값

더 자세한 내용은 XML 1.1 규약의 4.3.3과 부록 F를 참고하면된다.

XHTML

원칙적으로 XML과 같지만, 호환성을 위해 meta 태그를 통한 인코딩 선언과 XML을 통한 인코딩 선언을 같이 써 주어야 한다. 이 경우 두 값은 일치하여야 한다. 일치하지 않을 경우 표준은 XML 선언의 값에 우선 순위를 부여하지만, XHTML을 지원하지 않은 구식 브라우저에서 잘못 해석할 수도 있으므로, 두 값은 꼭 일치시키십시오. 또, meta 태그를 통한 인코딩 선언에서 meta tag을 닫을 때 ">"가 아니라 "/>"를 써야 한다는 점에 유의하십시오. X(HT)ML에서는 UTF-8이 기본 문자 인코딩이므로, UTF-8을 쓸 경우에는 'encoding="UTF-8"'은 생략 가능한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ko" xml:lang="ko">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
/>
<title>문서 제목</title>
... 다른 헤더 태그들 : style, script, link 등 ..
</head>
```

문자 인코딩 결정의 우선 순위는 XML과 같다.

CSS

CSS 문서 내부에서 encoding을 지정하기 위해서는 CSS 파일은 다음과 같이 시작해야 한다. Unicode에 기반한 인코딩 (UTF-8, UTF-16LE, UTF-16BE, UTF-32BE 등)을 쓸 경우에는 BOM(Byte Order Mark)을 파일 선두에 붙여서 문자 인코딩을 표시할 수도 있다.

```
@charset="UTF-8";    /* UTF-8 로 지정 */
@charset="EUC-KR";   /* EUC-KR 로 지정 */
```

문자 인코딩 결정의 우선 순위는 XML과 같다.

- HTTP 헤더를 통해 전송된 Content-Type 필드의 "charset" 파라미터
- BOM(Byte Order Mark) 혹은 @charset를 써서 CSS의 최선두에서 지정한 값
- CSS 문서를 참조하는 HTML 문서에서 <link charset="">로 지정한 값
- CSS 문서를 참조하는 HTML/XML 문서의 문자 인코딩
- UTF-8로 가정

웹 서버 프로그램에서 지정

Apache 서버

MIME 타입 지정과 마찬가지로 DefautlCharset 디렉티브를 써서 기본값을 지정할 수 있다. 하지만, 이 값은 웹 서버 전체에 대해서는 지정하지 않는 것이 좋다. 특정 디렉토리나 특정 가상 호스트에서 제공되는 모든 정적인 텍스트 문서가 이 기본값을 문자 인코딩으로 사용하는 *특수한 경우*에만 이 값을 지정하는 것을 고려해 볼 수 있다. 또, 웹 서버 관리자가 이 값을 설정해 놓았고, 일반 사용자나 가상 호스트 사용자가 자신의 웹 영역에 있는 모든 정적인 문서에 대해 다른 값을 일관되게 쓰고 싶다면, .htacces에서 이 값을 지정할 수 있다.

```
DefaultCharset UTF-8
```

AddType을 써서 확장자를 MIME 타입에 대응시킨 것과 마찬가지로 AddCharset을 써서 확장자를 문자 인코딩에 대응시킬 수 있다. 이 대응에 대해서는 다음 문서를 참고하십시오.

- http://httpd.apache.org/docs/2.1/mod/mod_mime.html#addcharset
- http://httpd.apache.org/docs/1.3/mod/mod_mime.html#addcharset

```
AddCharset UTF-8 .utf8
AddCharset EUC-KR .euckr
AddCharset EUC-JP .eucjp
AddCharset ISO-8859-1 .iso88591
AddCharset GB2312 .gb2312
AddCharset UTF-16LE .utf16le
AddCharset UTF-32BE .utf32be
```

Apache가 송신하는 문서는 복수의 확장자를 가질 수 있으며, Apache 서버는 복수의 확장자에 대응하는 MIME 타입, 문자 인코딩을 찾아 HTTP 헤더의 Content-Type 필드의 값으로 내보낸다. 예를 들어, "test.html.utf8.ko"란 파일은 다음과 같은 HTTP 헤더를 가지고 (다른 헤더 필드는 생략) 송신된다. 복수의 확장자가 붙는 순서는 중요하지 않다. "test.html.ko.utf8"이나 "test.ko.utf8.html" 파일도 아래와 같은 HTTP 헤더를 가지고 송신된다.

```
Content-Type: text/html; charset=UTF-8
Content-Language: ko
```

위의 보기에서 AddLanguage 디렉티브를 써서 확장자 'ko'를 언어 코드 'ko'에 대응시켰다고 가정했다.

JSP/Java servlet

동적 문서의 경우에는 MIME 타입 설정에서 설명한 방식으로 문자 인코딩을 지정할 수 있다. 혹은, setLocale을 사용하고, 다음처럼 locale-encoding-mapping-list에 locale에 대응하는 인코딩을 지정하는 방법도 있다고 한다. 아래와 같은 설정이 서버 설정 파일에 있을 때, setLocale("ko")는 문자 인코딩을 'UTF-8'로 만들고, setLocale("ja")는 'Shift_JIS'로 만듭니다. 또, JSTL의 지역화 관련 태그를 쓸 때에도 locale이 암묵적으로 설정되는 경우가 있으므로 주의해야 한다. 하지만, 이런 경우에도 page 디렉티브에서 contentType으

로 지정한 값이 우선권을 가집니다. 일부 옛 버전(JSTL 1.0)에서는 이 우선 순위가 바뀌어 있었지만, 최신 버전에서는 고쳐졌다. 옛 버전을 써야 하는 경우라면, 문자 인코딩을 지정한 직후 (암묵적으로 문자 인코딩을 바꿀 가능성이 있는 JSTL 태그를 쓰기 전에) ServletResponse.flushBuffer를 써서 버퍼를 비워주면 된다.

```
<locale-encoding-mapping-list>
  <locale-encoding-mapping>
    <locale>ko</locale>
    <encoding>UTF-8</encoding>
  </locale-encoding-mapping>
  <locale-encoding-mapping>
    <locale>ja</locale>
    <encoding>Shift_JIS</encoding>
  </locale-encoding-mapping>
</locale-encoding-mapping-list>
```

JSP 2.0에서는 deployment descriptor에서 다음과 같이 JSP 소스 코드 인코딩을 지정할 수도 있다. 이렇게 지정한 경우 소스 파일 내의 page 디렉티브로 지정한 값보다 우선 순위가 높다는데에 유의하십시오.

```
<jsp-property-group>
  <page-encoding>UTF-8</page-encoding>
</jsp-property-group>
```

HTML form을 통해 POST로 전달되는 파라미터의 문자 인코딩은 또다른 방법으로 지정해야 한다. ServletRequest object의 characterEncoding property를 지정할 수 있다. JSP에서는 이 방법 이외에 JSTL의 <fmt:requestEncoding>를 써서 지정할 수도 있다.

단, 이 경우에도 Apache tomcat5 등을 비롯한 JSP container는 GET으로 넘어온 파라미터를 해석할 때 이 값을 쓰지 않다. 이 문제를 해결하기 위해서는 server.xml 파일에서 useBodyEncodingForURI를 true로 설정해야 한다. Tomcat container의 경우 URIEncoding이나 QueryStringEncoding을 쓰는 방법도 있다. UTF-8을 항상 쓴다면 이 값을 UTF-8로 설정하는 방법도 고려할 수 있다.

더 자세한 내용은 http://issues.apache.org/bugzilla/show_bug.cgi?id=23929 혹은 http://java.sun.com/developer/technicalArticles/Intl/MultilingualJSP/index.html 를 참고하면 된다.

JSP 파일에서 디렉티브를 파일 선두에 사용할 때에는 원하지 않는 빈 줄이 생성되어 전송되는 파일의 선두에 들어가지 않도록 주의해야 한다. 복수의 디렉티브를 쓸 경우에는 아래처럼 뒤에 오는 디렉티브가 앞에 오는 디렉티브의 후미와 같은 줄에 오도록 해야 한다. 이렇게 해야만 파일 선두에 공백 없이 와야하는 XML 선언문이나 DTD 선언 등이 수신 측에 제대로 인식이 된다. 혹은 DTD나 XML 선언문을 page 디렉티브보다 먼저 쓰는 방법도 있다.

```
<%@ page pageEncoding="UTF-8"
   contentType="text/html; charset=UTF-8"
   import="java.io.*, java.net.*" %><%@
   taglib prefix="my"
   uri="http://jakarta.apache.org/tomcat/jsp2-example-
taglib"%><!DOCTYPE html
```

```
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<html>
   <head>
   <meta http-equiv="Content-Type" content="text/html charset=UTF-
8" />
   <title><%= application.getServerInfo() %></title>
   </head>
<body>
.....

</body>
</html>

<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%@ page pagecontentType="text/html charset=UTF-8"
   import="java.io.*, java.net.*" %>
<%@ taglib prefix="my"
   uri="http://jakarta.apache.org/tomcat/jsp2-example-taglib"%>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">
<html>
   <head>
   <meta http-equiv="Content-Type" content="text/html charset=UTF-
8" />
   <title><%= application.getServerInfo() %></title>
   </head>
<body>
<% request.setCharacterEncoding("UTF-8");
      response.flushBuffer();
      String names[];
      names = request.getParameterValues("name");
%>
</body>
</html>
```

IIS and ASP

http://www.w3.org/International/O-HTTP-charset에 있는 내용을 참고하였다. 정적인 문서에 대해서 문자 인코딩을 지정하고자 한다면, MIME 타입 지정과 같은 방법을 쓰면서 각 MIME 타입에 대응하는 확장자에 대해서 'text/html;charset=UTF-8'(예를 들어, html/htm에 대해서)과 같은 식으로 charset parameter를 더해 주십시오. 이 때, 공백을 넣지 않도록 주의해야 한다.

ASP로 만드는 동적인 문서의 출력 인코딩은 response.Charset으로 다음처럼 지정할 수 있다. response.Codepage는 ASP 소스 문서의 인코딩을 지정하는데 쓰이다. (65001은 UTF-8, 949는 확장 완성형)

```
   <% response.Charset= "UTF-8" %>
```

```
<% response.CodePage=65001 %>
<% response.CodePage=949 %>
```

ASP.Net의 경우에는 설정 파일(Machine.config나 Web.config)의 globalization 요소에서 responseEncoding (출력 인코딩), fileEncoding (asp 소스 파일 인코딩), requestEncoding (get/post request의 문자 인코딩) 등을 설정할 수 있다.

PHP

PHP의 경우 MIME 타입 설정에서 설명한 대로 header() 함수를 써서 Content-Type 헤더 필드를 내보낼 때에 "charset" 파라미터를 같이 써 주면 문자 인코딩을 지정할 수 있다.

참고 문헌

1. W3C 의 HTTP 표준 관련 자료 모음 (http://www.w3.org/Protocols) : HTTP 1.0, HTTP 1.1 등에 대한 랑크 포함
2. HTML 4.01 표준(http://www.w3.org/TR/html401/) : 5.2에서 문자 인코딩 선언법과 결정 방법을 다루고 있다. 부록 B.2에서 URI 에 대해 언급하고 있지만, RFC 3987에서 이와 약간 다른 방식을 표준으로 지정하였음에 유의하십시오.
3. CSS(http://www.w3.org/Style/CSS/)
4. CSS 2.1 표준(http://www.w3.org/TR/CSS21/) : 4.4에서 문자 인코딩 선언법과 결정 방법을 다루고 있다.
5. XML 1.1 표준(http://www.w3.org/TR/xml11) : 부록 F 와 4.3.3에서 문자 인코딩 선언법과 클라이언트에서 어떻게 문자 인코딩을 알아 내는지 다루고 있다.
6. RFC 2616 (http://ietf.org/rfc/rfc2616.txt?number=2616) : HTTP 1.1
7. 자세한 MIME 타입 목록(http://www.utoronto.ca/webdocs/HTMLdocs/Book/Book-3ed/appb/mimetype.html): 1997년판으로 낡았음.
8. IANA MIME 타입 목록 (http://www.iana.org/assignments/media-types/) : application/* 타입만 포함
9. 또다른 MIME 타입 목록 (http://www.webmaster-toolkit.com/mime-types.shtml) : 규범적인 것은 아님
10. RFC 2045(http://www.faqs.org/rfcs/rfc2045.html) : RFC 2045에서 2049까지 문서에서 MIME 의 기본을 정의한다.
11. IANA charset registry(http://www.iana.org/assignments/character-sets) : EUC-KR 이 preferrred MIME 이름으로 등록되어 있음. 하지만, 앞으로 만드는 모든 한국어 문서는 UTF-8이나 UTF-16 사용이 바람직한다.
12. http://web-sniffer.net : HTTP Sniffer
13. http://sniffuri.org : 또다른 HTTP sniffer
14. Apache 2.x MIME 타입 설정 방법 (http://httpd.apache.org/docs/2.1/mod/mod_mime.html)
15. Apache 1.x MIME 타입 설정 (http://httpd.apache.org/docs/1.3/mod/mod_mime.html#addtype)
16. Apache 2.x 문서 (http://httpd.apache.org/docs/2.1/) : AllowOverride, MIME, htaccess, DefaultCharset 등의 키워드를 치면 Apache 2.x 서버의 해당 항목에 대한 자세한 설명을 볼 수 있음.
17. Apache 1.x 문서 (http://httpd.apache.org/docs/1.3/) : AllowOverride, MIME, htaccess, DefaultCharset 등의 키워드를 치면 Apache 1.x 서버의 해당 항목에 대한 자세한 설명을 볼 수 있음.
18. Windows Media File MIME Types(http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwmt/html/mime.asp): 여러 가지 웹 서버에서 Windows Media 파일을 위한 MIME 타입 설정법. 다른 파일 형식에도 응용할 수 있다.
19. W3C 국제화 워킹 그룹 (http://www.w3.org/International/) : 웹의 국제화 관련 풍부한 자료를 구할 수 있음.
20. 문자 인코딩 지정 방법 튜토리얼(http://www.w3.org/International/tutorials/tutorial-char-

enc/)

21. 문자 인코딩 지정 방법 (http://www.w3.org/International/O-HTTP-charset) : 간략한 설명
22. RFC 3987 (http://www.ietf.org/rfc/rfc3987) : IRI (국제화된 리소스 식별자)를 규정한 RFC. HTML 4.01의 B.2와는 약간 다른 방법이 표준으로 채택되었음에 유의.
23. 국제화된 도메인 이름과 리소스 식별자(IRI)(http://www.w3.org/International/articles/idn-and-iri/ : IDN 과 IRI 에 대한 여러 가지 정보. RFC 3986(URI), RFC 3987(IRI), Apache 2.x 용 fileiri 모듈을 비롯한 여러 자료에 대한 링크가 있음
24. RFC 3023 (XML Media type)(http://www.ietf.org/rfc/rfc3023.txt)
25. XHTML media type(http://www.w3.org/TR/2002/NOTE-xhtml-media-types-20020801/)
26. Javascript/ECMAscript MIME type(http://www.ietf.org/internet-drafts/draft-hoehrmann-script-types-03.txt)
27. XHTML 호환성 지침(http://www.w3.org/TR/2002/REC-xhtml1-20020801/#guidelines)
28. Unicode 용어집(http://www.unicode.org/glossary)
29. JSP 를 이용한 다국어 지원 웹 사이트 개발 튜토리얼 (http://java.sun.com/developer/technicalArticles/Intl/MultilingualJSP/index.html) : JSP 개발자라면 꼭 읽어야 할 문서이다.
30. Java Server Page 표준 (http://java.sun.com/products/jsp/reference/api/index.html) : JSP 1.1, 1.2, 2.0의 표준 문서를 구할 수 있다. JSP 2.0의 4장에서 문자 인코딩에 대해서 자세히 다루고 있다.
31. JSP 를 이용한 웹 사이트의 국제화 (http://java.sun.com/developer/technicalArticles/Programming/jspwebsites/index.html)
32. Java 국제화(http://java.sun.com/j2se/corejava/intl/index.jsp)
33. IBM web sphere 에서 문자 인코딩 지정 방법(http://www-306.ibm.com/software/globalization/j2ee/encoding.jsp)
34.

실전 웹 표준 개발 프로세스

기존 웹 개발 프로세스

일반적으로 웹 개발 프로젝트란 웹의 다양한 기술을 바탕으로 브라우저 상에서 이용할 수 있는 문서 혹은 서비스를 기획, 개발하는 일련의 작업과정을 가리킨다. 이전에는 주로 Internet Explorer나 Mozilla같은 데스크탑 브라우저 클라이언트 상에서 동작하는 것만을 목표로 했으나, 기술이 발달하면서 데스크탑 브라우저 이외에도 모바일 및 유비쿼터스 환경에서의 이용, 여러 프로그램 및 기계장치에서의 입,출력 데이터 포맷 등 다양한 용도로의 활용이 요구되고 있다. 이러한 다양한 욕구를 충족시키기에는 기존의 데스크탑 브라우저 중심의 개발 개념/기술/환경은 부족함이 있으며 이에 따라 새로이 웹 표준화의 중요성이 강조되고 있다.

그러나 실무현장에서는 업무종사자들이 아직 기존의 낡은 개발방법론을 따르고 있어 웹 표준화를 준수하는데 구조적인 문제가 발생할 수 있다. 즉, 단순히 (x)HTML이나 CSS등의 웹 표준 기술이나, 웹 접근성 등에 대한 이해가 있다 하더라도, 기존의 개발 프로세스 상에서는 이를 적용시키기가 어려울 뿐더러 오히려 기회비용의 증가로 인해 작업효율성이 떨어지게 되어 웹 표준을 거부하는 현상도 발생하게 된다. 따라서 웹 표준화를 따르기 위해서는 기존의 방식보다 개선되고 발전된 새로운 개발 방법론을 채택할 필요가 있다. 그러나 일부 대형 사업장 및 전문 업체만이 이 분야에 대해 관심을 가지고 자신들의 상황에 맞추어 실무에 적용하고 있을 뿐, 우리나라의 대부분의 웹 개발 관련 업체들의 영세성이나 후진성 때문에 이에 대한 체계적인 접근 및 고민은 전무하다고 해도 과언이 아니다.

이에 이 문서에서는 기존 방법론의 문제점을 확인하고 웹 표준화를 준수하기 위해 필요한 새로운 방법론을 제시하고자 한다.

주의 : 이 문서에서 소개 및 제시되는 모든 개발 방법론들은 모든 경우에 반드시 맞아떨어지는 방법론이라 할 수는 없다. 그러한 방법론은 존재하지 않는다, 개인, 조직, 기업은 저마다 고유한 특성과 환경을 가지고 있어서, 일률적으로 한가지 방법이 최선의 선택일 수는 없다. 따라서 이 문서는 어디까지나 자신들의 고유한 방법론을 개발하기 위한 참고용으로만 활용되기 바란다.

현재 프로세스 소개(Waterfall 방식)

이전에 사용하던 일반적인 웹 개발 업무의 흐름이다. 대략적으로 살펴보자면, 다음과 같은 순서를 따른다.

1. 기획자가 기획작업을 거쳐 스토리보드를 산출해내면
2. 디자이너가 이미지 편집 프로그램을 이용해 필요한 이미지를 생산해 낸 후,
3. 코더가 적절히 구획된 테이블 안에 해당 이미지와 컨텐트들을 배치한 HTML 코드를 생산해내고,
4. 프로그래머가 산출된 코드를 받아 웹 프로그래밍 언어와 조합하여 프로그램이 적용된 최종 결과를 완성한다.

기존의 프로세스의 특징이라면, 이른바 선형 개발 과정을 따르기 때문에, 컨베이어 벨트처럼 밀어내기식 개발을 하게 된다. 마치 폭포처럼 보인다 해서 폭포수(Waterfall) 방법론이라고 불리우기도 한다.

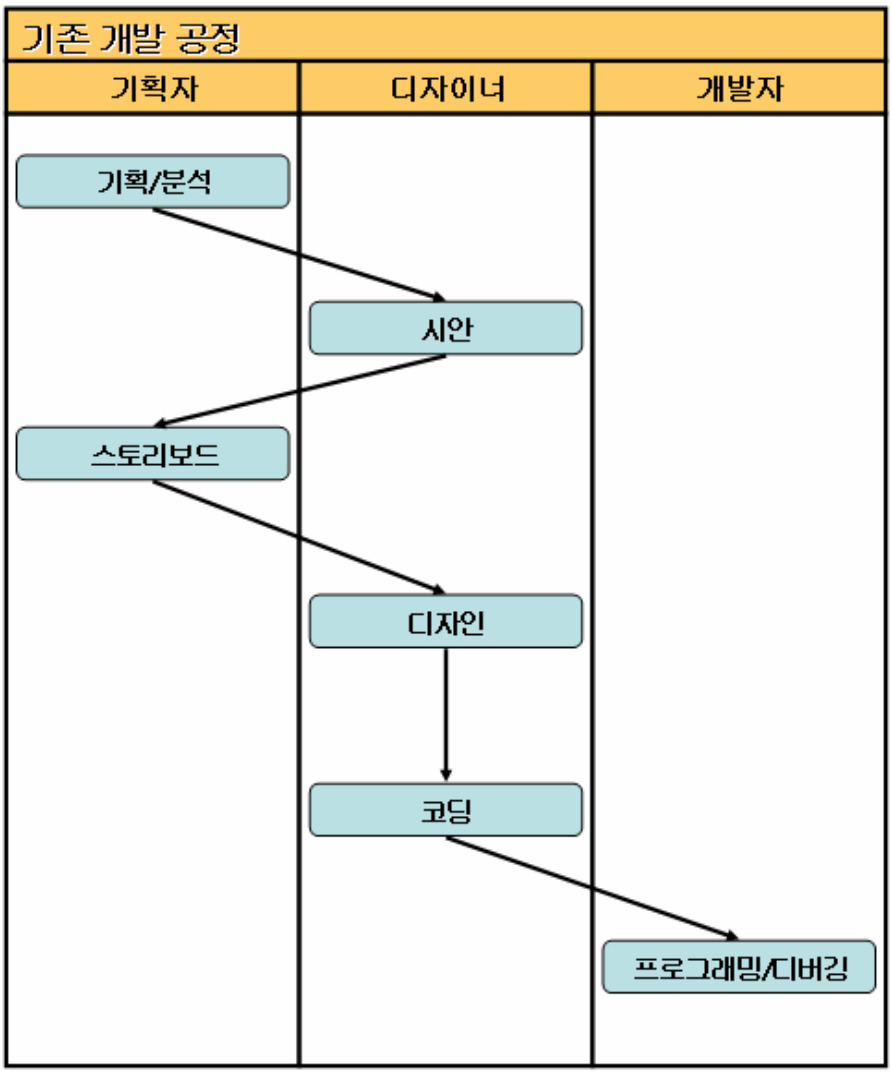


그림 43 기존 웹 개발 공정표

이 방식은 각 프로세스별 단계 파악이 용이하고, 비교적 작업이 정형화되어 대단한 기술이나 능력을 요구하지 않는 특징이 있다. 그러나, 웹 표준화를 준수하는 데 있어 이러한 기존 방법론은 구조적인 문제에 봉착하게 된다.

문제점(1) 병목현상

기존 프로세스의 중요한 문제점 중 첫번째는, 밀어내기식 선형 작업 진행에 의해 필연적으로 발생하는 자원관리의 비효율성과 병목현상을 들 수 있다.

디자이너가 디자인을 완성하기 전까지는 HTML 코드를 생산해낼 수 없다. HTML 코드가 생산되지 않으면 프로그래머는 프로그램 개발을 진행할 수 없다. 기존 프로세스 방식을 따랐을 때의 일반적인 상황이다.

예를 들어보자.

전체 개발 기간으로 30일을 산정한 웹 사이트 개발 프로젝트가 있다. 프로젝트 매니저가 주의깊게 프로젝트를 검토한 결과, 기획단계에 10인-일 단위가, 디자인 단계에 10인-일 단위가, 개발 단계에 10인-일 단위가 필요하다는 결론을 내렸다. 이에 따라 인력이 이 프로젝트에 적절히 투입된다. (실제로는, 기획자가 스토리보드를 완성할 때까지 프로젝트의 규모를 추산해내지 못하는 경우도 허다하다.)

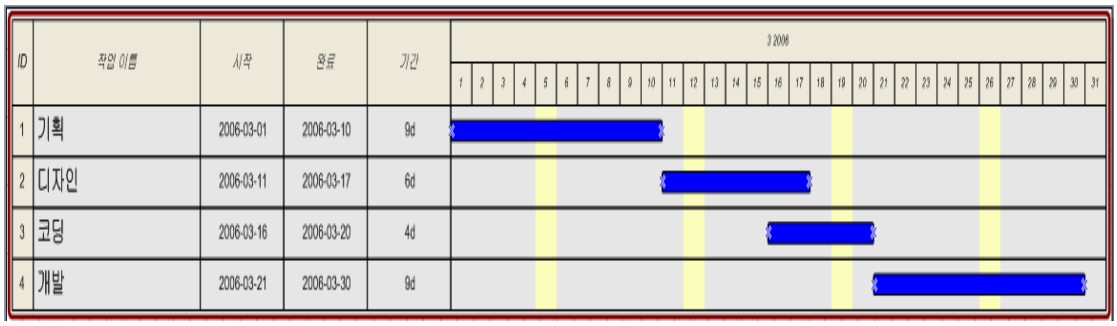


그림 44 협업을 이루어 내지 못하는 개발 공정표

대개의 경우 위의 그림처럼 프로젝트 일정은 밀어내기식 순차진행을 보이게 된다. 이전 단계의 작업이 어느정도 끝나야 다음 단계의 작업을 시작할 수 있다.

기획단계에서 스토리보드를 작성해내고 디자이너가 디자인을 만들어 HTML 코드를 생산해줄 때까지 프로그래머는 약 20일간 할 일이 없게 된다. 물론 어느 단계에서 예상 외의 지연상황이 발생하게 되면 후속 단계들도 그만큼 늦어지게 된다. 선행 단계에서의 지연 누적은 후속 단계로 갈 수록 일정 준수에 대한 부담으로 쌓이게 되며, 대개의 경우 가장 마지막 단계에 속하는 프로그래머들이 프로그래밍에 착수할 때 즈음에는 작업을 위한 충분한 일정을 확보하지 못한 채 프로젝트 종료일정의 압박을 받으며 작업해야 한다.

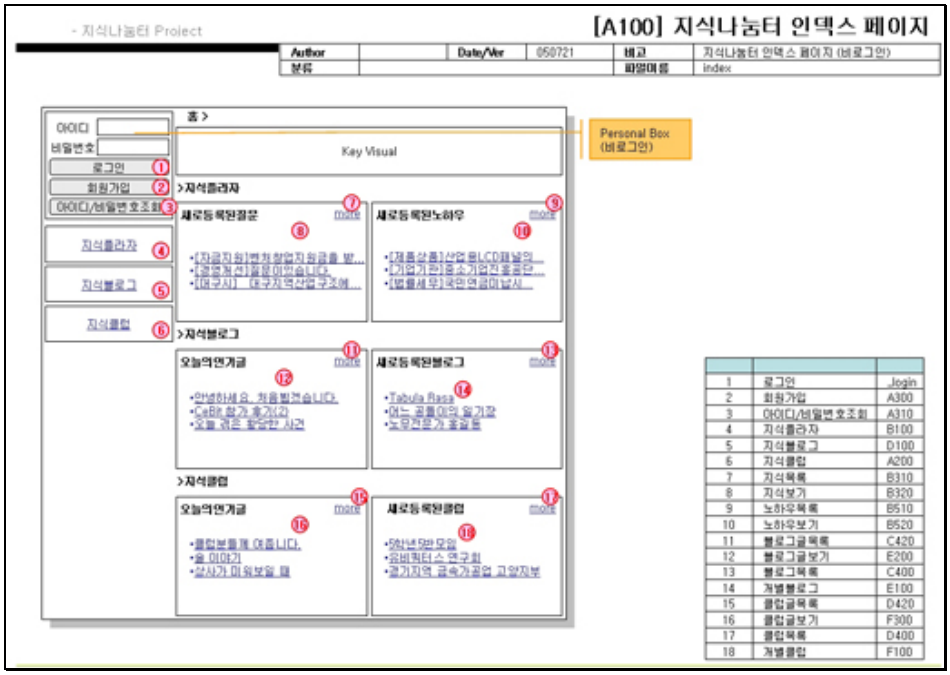
이러한 문제점을 해결하기 위해 일반적으로 쉽게 취할 수 있는 방법 중 한가지는, 자신의 작업 단계가 가능할 때까지, 해당 인력을 다른 프로젝트에 할당하는 것이다. 그러나 이 방법은 이론만큼 효율적이지는 못하다. 어떠한 실무자도 두 개의 프로젝트에 50%씩 투입되기 보다는 하나의 프로젝트에 100% 투입되기를 원한다. 만약 계획대로 일정이 지켜진다 하더라도 아무런 준비없이 다른 프로젝트의 중간단계를 바로 시작할 수 없기 때문이다. 게다가 이 방법은 병목현상에 대해 아무런 효과를 볼 수 없다. 일정은 언제나 계획보다 지연되기 마련이다. 프로젝트 기간의 대부분 동안 다른 프로젝트 개발에 투입되어 있다가 원래 프로젝트로 돌아와 며칠 남지 않은 마감을 앞에 두고 개발을 시작해야하는 프로그래머의 입장을 상상해보라.

해결 방법 중 또다른 한가지는, 선행단계의 일정을 단축시키는 것이다. 스토리보드가 빨리 나오면 그만큼 디자인이 빨리 완성된다. 디자인이 빨리 나오면 그만큼 코딩이 앞당겨진다. 코딩이 앞당겨지면 프로그래밍도 그만큼 앞당겨 진다. 그런 믿음으로 관리자들은 언제나 실무자들의 일정을 닥달하게 된다. 그러나 이것은 곧잘 무리한 공정 단축으로 이어지게 되며, 선행 단계에서 충분히 검토되고 완성되어야 할 작업들을 빠트리게 되는 핑계가 된다. 결국 잦은 변경과 수정으로 인해 작업진행이 체계적이지 못하게 되고, 원래 일정보다 오히려 더 지연되는 결과를 초래하거나 혹은 불완전한 결과물을 만들어내는 요인이 된다.

문제점(2) 스토리보드

스토리보드란 원래 시나리오의 흐름(스토리)을 기술하는 유스케이스 모델링 기법의 일종이다. 그러나 현재 국내의 일반적인 웹 개발 스토리보드란 작업지시명세서 및 페이퍼 시뮬레이션의 성격이 더 강하다. 기획이란 이렇다는 특정한 법칙이 있는 것은 아니니, 스토리보드를 이용한다는 그 자체가 문제가 되는 것은 아니다.

일반적인 스토리보드는 다음 그림들과 같다.



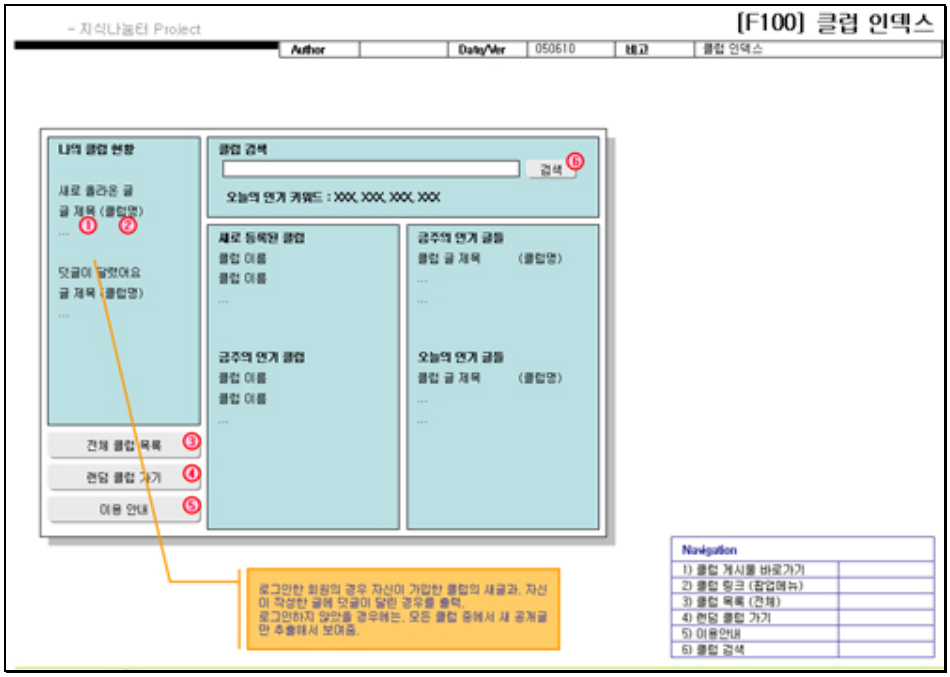


그림 45 의존적인 스토리 보드의 전형적인 예

실제 현장에서 사용되는 스토리보드 문서들의 형태는 대략 이와 같으며, 주로 웹페이지의 외형과 컨텐츠, 동작 기능에 대해 기술하고 있다.

문제는 스토리보드에 지나치게 의존적인 작업형태라는 점이다. 기획단계에서 인터뷰/회의를 통해 대강의 흐름과 요구사항을 확인한다 하여도 그 결과가 스토리보드에 반영되어 확정되기 전까지 다른 사람들은 설계에 들어갈 수 없다. 관행상 혹은 일정상이라는 이유로 기획자들은 스토리보드만을 산출해낸다. 예를 들어 DBA가 DB 스키마 설계를 하려 해도, 어떤 객체들과 어떤 기능, 어떤 속성들이 있을지 사전 정리가 안되어 있기 때문에 설계가 힘들어진다. 메인 프로그래머가 전체 시스템에 대한 프레임워크를 적용하려 해도 비즈니스 로직이 명확히 정의되지 않았기 때문에 스토리보드의 완성을 기다려야만 한다. 디자이너 역시 디자인 지시사항이 스토리보드에 명시되어 있기 때문에 스토리보드 없이 디자인은 불가능하다.

스토리보드가 산출되어야 디자이너는 스토리보드에 따라 디자인을 시작할 수 있고, 프로그래머들은 스토리보드를 보고 전체적인 프로세스 플로우와 비즈니스 로직을 뽑아낼 수 있으며, DBA는 DB설계에 필요한 요소들을 잡아낼 수 있다. 심지어 이러한 이유로, 스토리보드가 완성되고 나서야 프로젝트의 정확한 규모, 일정, 소요인력등이 결정되기도 한다.

선형 순차 개발 방법론의 디자인->코딩->프로그래밍이라는 구조상 이를 해결할 수 있는 방법은 그다지 다양하지 못하다. 애초에 프로그램 설계를 먼저 한다 하더라도 실제 코딩 결과물이 산출되지 않는 한, 프로그래머로서는 설계와 대략적인 프레임워크는 만들 수 있어도, 실제 인터페이스 개발은 불가능하기 때문이다.

실질적으로 스토리보드 외의 레퍼런스를 만들기 어려운 현실 상, 상세하고 내용이 많을 수록 좋은 스토리보드로 평가받게 된다. 이는 또다른 문제점을 야기시키는데, 스토리보드가 상세하면 상세할 수록, 디자이너와 프로그래머들의 스토리보드 의존도가 높아진다는 점이다. 스토리보드 의존도가 높아지면 높아질 수록, 더욱 더 상세한 스토리보드를 기획자에게 요구하게 된다. 이 무한루프는 기획자들의 가장 큰 고민거리이며, 기획자들이 프로젝트 일정 내내 파워포인트를 붙잡고 몇백장씩의 세세한 스토리보드를 그려내는 일에만 몰두하게 만드는 원인이다. 그러나 아무리 뛰어난 기획자라 해도 완벽한 스토리보드를 그려내기란 거의 불가능하다.

다음 발언은 스토리보드 의존도가 심화된 상태의 현장에서의 흔히 발견되는 모습이라 할 수 있다.

> "스토리보드에는 그에 대한 지시사항이 없는 걸요" - 아주 기본적인 에러확인 로직을 스토리보드에 없다는 이유만으로 개발하지 않은 프로그래머의 반문
>
> "저는 스토리보드에 그려진 대로 그린 것 뿐인데요..." - 디자인이 참신하지 못하다는 지적을 받은 디자이너의 변명
>
> "어째서 스토리보드에 있는 그대로 하지 않았지?" - 발전적 창의성을 적용한 프로그래머/디자이너를 인정하지 않는 기획자의 질책

디자인 지시서 / 설계 지시서로 스토리보드의 역할이 고정되어 있는 한 이러한 문제점을 해결하기는 힘들다.

문제점(3) 구조화의 어려움

앞서 언급한 문제점들은 웹 표준화 패러다임과는 크게 관련없어 보일 수도 있다. 웹 표준화를 위한 문제점이라기 보다는, 개발 방법론 자체의 후진성에 기인하기 때문이다. 이제 좀 더 실질적으로 우리의 관심사인 웹 표준화라는 관점에서의 문제점을 짚어보자.

웹 프로젝트의 투입인력을 대략의 역할(role)에 따라 크게 구분해보면 기획/디자인/코딩/개발로 나뉘게 된다. 사실 대개의 경우 **코딩**을 따로 두는 경우도 많지 않다. 대부분 디자이너가 겸하고 있다. 체계적인 개발 방법론이 확립되지 않은 많은 현장에서 코딩이란, "디자이너가 Photoshop으로 그려낸 이미지를 적당히 잘라 Dreamweaver등의 도구를 이용해 적당히 배치한 후 HTML형식으로 저장하는 것"으로 정의되어 있는 셈이다. 실제 현장에서 지연된 일정을 만회하기 위해 택하는 가장 손쉬운 방법 중의 하나는 "코더를 추가투입하여 디자이너의 작업 중 단순반복적인 작업을 덜어주는 것"으로 알려져 있다.

그러나 웹 표준화라는 관점에서 보자면, HTML 코드의 생산은 가장 중요한 부분이라 할 수 있다.

주지하다시피 웹 표준화는 크게 웹 접근성의 확보 및 의미론적 웹이라는 두마리 토끼를 잡으려는 셈이다. CSS를 이용한 디자인, 크로스 브라우징 기술, 재활용가능한 웹, Web 2.0 등등 현재 이슈가 되고 있는 주제들도 결국은 웹 표준화의 수단과 목적이라고 할 수 있다.

이를 위해 가장 기초적으로 요구되는 것은, well-formed(structured) document라 할 수 있다. CSS를 이용한 디자인의 필요조건은 (X)HTML 문서가 제대로 구조화되어 있어야 한다는 점이며, 크로스 브라우징을 위해서도 표준규약에 맞는 (X)HTML 문서를 필수로 한다. 의미에 맞는 태그를 적절히 사용해야만 symantic web을 구현할 수 있다.

따라서, HTML코드의 생산은 전문적으로 훈련된 인력에 의해 주의깊게 작성되어야 하며, 디자인과는 독립된 하나의 작업공정으로 인정되어야 한다.

역할을 중심으로 한 개발 공정

이상에서 살펴본 바와 같이, 기존의 웹 개발 공정은 웹 표준화를 준수하는데 있어, 구조적으로 어려운 실정임을 알 수 있다. 물론, 개인과 조직의 역량에 따라 기존 방법론으로 완벽히 불가능한 작업이라고는 할 수 없으나, 더 나은 개발 방법론에 대한 연구가 필요한 시점이다.

이제, 일반적인 웹 개발 프로젝트 구성원의 역할에 따라 4가지 모델을 새롭게 제시한다. 각각의 모델은 장단점이 있으며, 또한 이 모델들이 최선의 선택이라고 할 수는 없다. 여기에 제시되는 모델들은 예시이며, 개인의 능력과 조직의 구성, 기술숙달정도, 조직문화, 프로젝트의 성격 등에 따라, 각 개별 프로젝트 단위마다 최적화된 다른 방법들이 존재할 것이다. 여기에 제시된 모델들을 참고로 자신에게 최적화된 방법론을 연구해보는 일이 필요하다.

전통적으로, 웹 개발 공정에는 다음 4가지 역할을 담당할 구성원들을 필요로 한다.

기획 : 한마디로 '기획'이라고 표현하긴 하지만, 각각 다른 여러 가지 분야를 포함하고 있다. 예컨데, 비즈니스 목표를 달성하기 위한 계획을 세우는 '마케팅 기획'이나 '경영

기획'등이 있고, 필요한 기반 기술들을 결정짓고 감독하는 '기술 기획'도 있으며, 실질적으로 웹 페이지의 컨텐트를 채우는 '컨텐트 기획'등, 기획에 포함되는 분야는 다양하다. 여기에서는 실질적으로 웹 페이지 개발시 중요한 '웹 컨텐트 기획'을 중심으로 논의한다.

디자인 : 웹 페이지는 1 차적으로는 시각적 매체이기에 시각디자인은 매우 중요한 요소이다. 그러나 '디자인(design)'에는 '설계'의 의미도 들어있으며, 실제로 웹 페이지 개발에 있어서 디자인이란, '아름답게 보이는 것'을 넘어서 사용자의 이용 편의성, 정보의 효율적인 전달, 웹 페이지/사이트의 목적에 부합하는 미적 표현 등이 포함된 개념이다.

개발 : 프로그래머 역시 그 역할에 따라 다양한 직능으로 분화가 가능하다. 웹 사이트 개발과 운영에 필요한 환경을 관리하는 시스템 엔지니어에서부터 최종 사용자에게 rich interface 를 제공하는 플래쉬 스크립터까지 여러 역할이 존재한다. 여기에서는, 실제로 웹 사이트 자체 제작에 필요한 서버사이드/클라이언트사이드 프로그래밍 역할을 중점적으로 살펴본다.

코딩/퍼블리싱 : 종래의 코딩이라는 작업은, "디자이너가 그린 그림을 HTML 로 옮김"이라는 개념이 강하였으며, 일반적으로 단순노동으로 저평가되었던 작업이었다. 실제로 특정프로그램을 이용한 HTML 코드 생산이라는 작업과정만 놓고 보면 그러한 저평가가 틀린 것만은 아니었다. 그러나 애초 HTML 의 목표자체가 인터넷 상에서 의미있는 정보를 교환하기 위함이라는 점을 감안한다면, 웹 상에 정보를 출판(publish)하기 위해 필요한 기술을 갖추고 이를 목적에 맞게 활용할 수 있는 전문가로서의 퍼블리셔(publisher)의 존재가 중요하다.

위와 같이 웹 개발 공정에 필요한 역할들을 나누었을 때, 웹 표준화를 적용시키기 위한 주도적인 역할을 누가 맡느냐에 따라 세세한 작업 절차가 달라지게 된다. 여기에서는 각 역할별로 4가지 모델을 제안한다. 각 모델은 이해를 돕기 위한 수단으로서의 제안이며, 실제 현장에서 적용시에는 반드시 프로젝트의 종류와 개개인의 역량에 따라 현실에 맞게 개별적으로 맞춤해야 할 것이다.

여기에서는 기획 -> 프로그래밍/디버깅 까지의 과정만 논의하며, 제안, 수주 등의 프리프로덕션 과정 및 검수, 인도 등의 포스트프로덕션 과정은 생략한다.

디자이너 중심

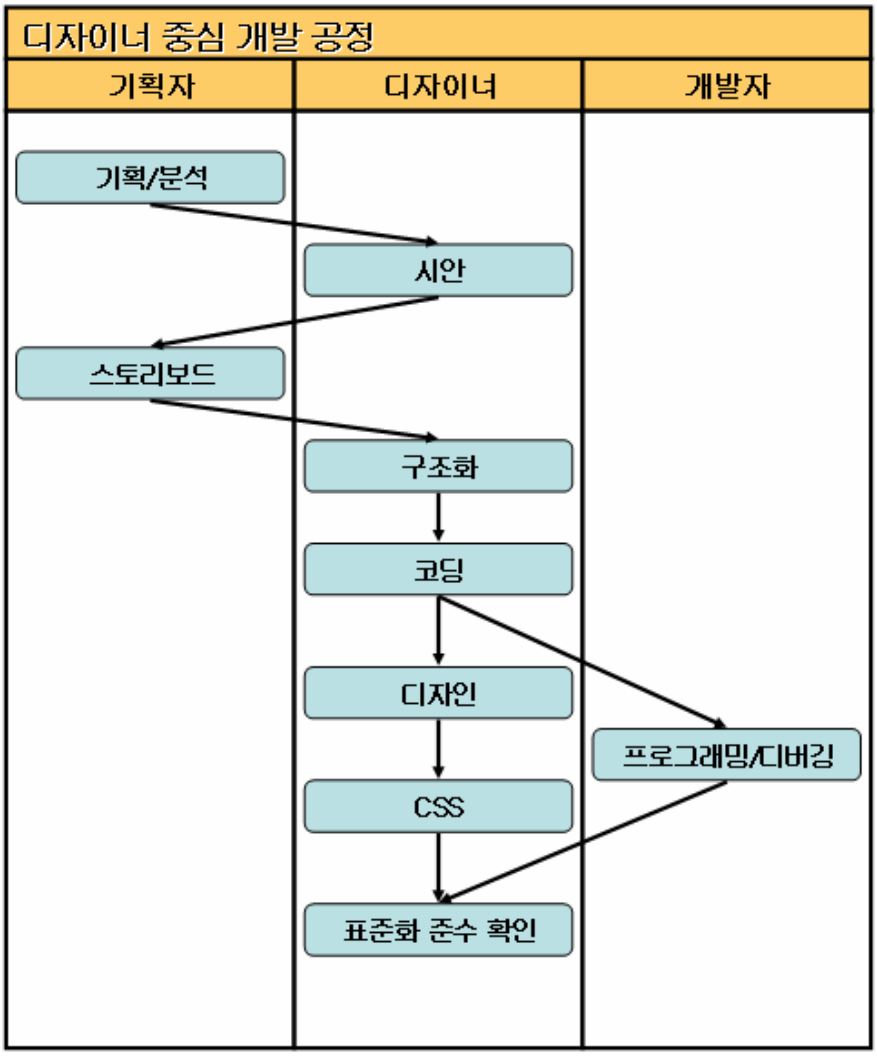


그림 46 디자이너 중심 개발 공정표

디자이너를 중심으로 한 개발 공정 모델의 경우, 디자이너가 웹 표준화의 중심 작업 (구조화, HTML 코딩)을 맡아하는 경우를 말한다. 기존 방식의 공정 모델과 비교하여, 변경되는 부분이 작으므로 가장 쉽게 적용할 수 있는 방법이라 할 수 있겠다.

디자이너 중심의 모델을 적용하기 위해서는 디자이너가 웹 표준화에 관한 필요기술을 추가적으로 습득해야 할 필요가 있다. 기존의 방법에서 대개의 경우 디자이너가 코딩까지 겸하는 경우가 많으므로, 작업의 일관성을 유지하기 쉽다는 장점이 있다. 또한, 디자인보다 코딩이 먼저 끝나게 되므로, 프로그래머의 작업 시작이 빨라지게 된다.

그러나, 웹 표준화를 위해서 추가로 익혀야 할 기술에는 ECMA 스크립트 표준 등의 기술 중심적인 내용들이 많이 있기 때문에, 제대로 훈련받지 않은 디자이너에게는 상당히 어려운 과정이 될 수 있으며, 감성적인 부분을 중시하는 디자이너 특성상 논리적, 구조적인 능력을 필요로 하는 웹 표준화를 적용하는 것이 무리한 요구일 수도 있다. 예를 들어 정보의 구조와 가치에 따라 구조적 HTML을 생성해야함에도, 디자인적 감성으로 시각적인 요소

에 집중하여 논리적 구조화보다는 시각적 구조화에 그치는 등의 문제점이 발생할 수 있다.

또한, 기존의 모델에서 크게 변경된 부분이 없으므로, 여전히 순차적 진행에 따른 작업 지연 등의 문제점은 개선되지 못한다.

프로그래머 중심

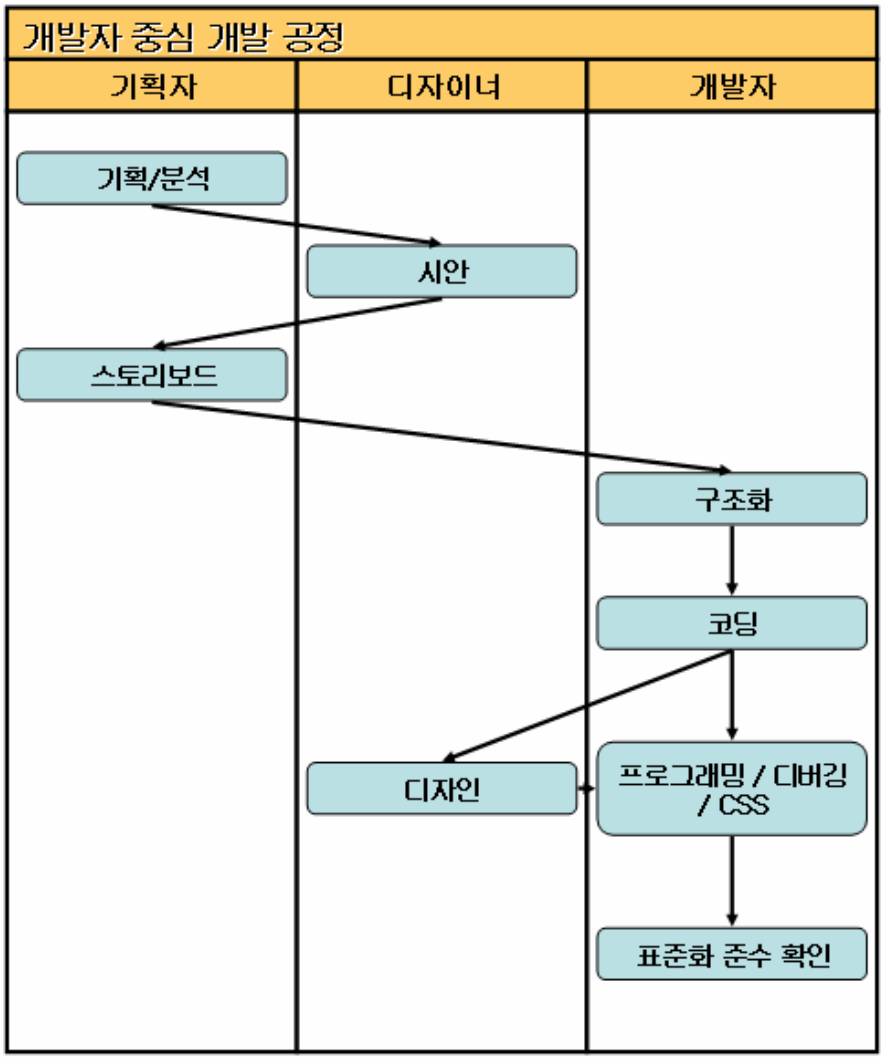


그림 47 개발자 중심 개발 공정표

논리와 구조, 표준을 중요하게 여기는 프로그래밍의 특성상, 프로그래머들은 다른 어떤 역할을 맡은 사람보다 웹 표준화에 대한 이해가 빠르며, 관련 기술에 대한 습득이 용이하다. 따라서 프로그래머 중심의 개발 공정 모델은 디자이너 중심의 개발 공정 모델보다 성공할 가능성이 높다. 또한 자신이 생산한 코드를 바로 프로그래밍에 사용할 수 있으므로 빠른 개발 속도를 확보할 수 있는 장점이 있다.

그러나 프로그래머가 웹 표준화에 대해 충분히 이해하고 있지 않다면, 단지 프로그램 적용을 위한 HTML코드만을 의도적으로 생산할 가능성이 있으며, 이는 제대로 구조화되지 않은 웹 페이지를 생산해내는 결과를 가져올 수 있다. 웹 페이지의 구조화는 개발의 용이성

등의 문제와는 별개로 문서 자체의 가치와 내용에 따라 결정되어야 하기 때문이다.

이에 더하여, 프로그래머에게 너무 많은 작업이 배정되므로 충분한 인력과 시간을 확보하지 못한다면 프로그래머에게 큰 부담이 되는 모델이다. 특히, 디자인과 프로그래밍이 동시 진행되게 되므로, 1인 개발 등의 소규모 프로젝트에서는 CSS 관련 작업에 대한 시간 배분이 힘들어지는 단점이 있다.

기획자 중심

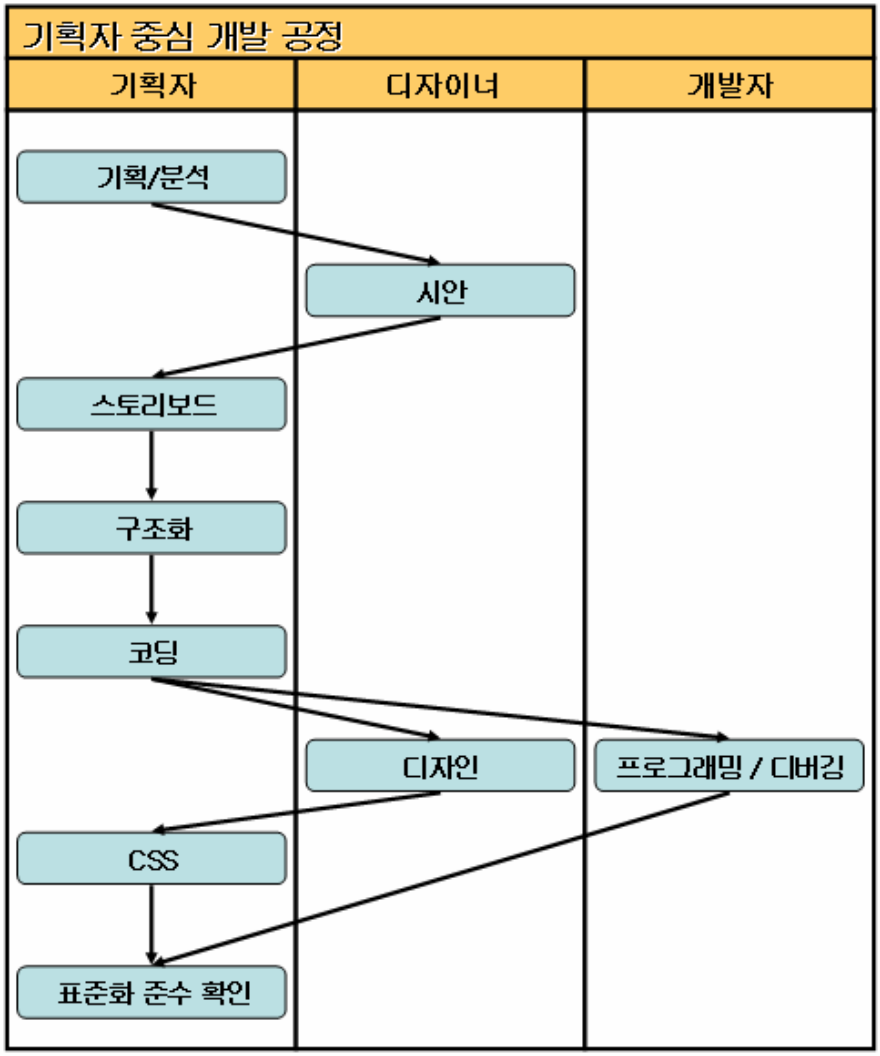


그림 48 기획자 중심 개발 공정표

기획자 중심 개발 공정 모델의 장점은 다른 모델들 보다 기획자의 기획의도에 맞게 웹 페이지의 논리적 구조화가 가능하다는 점이다. 디자인이나 개발에 의해 구조화자체가 왜곡되는 현상을 최소화시킬 수 있다는 점은 웹 표준화 개발에 있어 다른 모델들에 비해 큰 장점이라 할 수 있다.

그러나 역시 디자이너 중심 모델과 마찬가지로, 기획자가 기술지향적인 관련기술을 습득해

야 하는 부담이 있고, 과도한 스토리보드 작업에 더해 기획자의 작업부담이 늘어난다는 문제점이 존재한다.

이상에서 제안한 세가지 기존 역할 중심 모델은 기존 개발 공정 모델에 비해 접근하기 쉬우나, 중심 역할을 맡은 인력에게 추가 작업 부담이 더해지기 때문에 오히려 전체 공정 기간이 늘어나고 업무 편중이 과도해지는 단점이 있다. 또한, 앞 장에서 말한 기존 모델의 문제점을 개선하는 데 구조적인 한계를 지니게 된다. 이를 해결하기 위해서는 기존의 역할에서 벗어나 웹 표준화 작업을 담당할 새로운 역할이 필요하며, 이에 따라 새로운 개발 공정 모델을 제시해보도록 한다.

개선된 모델(퍼블리셔 중심)

우선, 기존의 개발 공정 모델 및 기존 역할을 중심 모델들의 개선해야 할 부분들을 찾아보자.

과도한 스토리보드

앞에서 언급한 바와 같이, 지나친 스토리보드 중심의 기획문서 생산과정은 기획자의 업무를 과중하게 하고, 다른 역할들의 능동적, 적극적 참여를 제한하며, 작업 일정 지연 및 대기기간 증가 등의 단점을 불러오게 된다.

이를 해결하기 위해서는 스토리보드를 경량화하고, 필요에 따라 각기 다른 종류의 문서로 분할하여, 전체적인 작업 일정을 단축시킬 수 있도록 해야 한다. 또한 과감히 다른 역할을 맡은 인력들이 프로젝트에 적극적이고 능동적인 참여가 가능하도록 하여 기획자의 부담을 줄이고 창의적이고도 효율적인 작업 결과를 산출해낼 수 있도록 해야 한다. 디자인 지시서로서의 스토리보드의 성격을 폐기함으로써, 디자인은 온전히 디자이너 고유의 영역으로 남겨주어야 한다.

개선된 기술의 도입

대표적인 웹 프로그래밍 언어 중 하나인 PHP를 예로 들자면, 간단히 HTML에 스크립트를 추가하는 것만으로 프로그램 개발이 가능하며 문법도 쉽다는 장점이 있으나, 오히려 그 장점이 구조적이고 관리하기 용이한 프로그램을 작성하는 데 걸림돌이 되는 형편이다. 국내의 많은 PHP 프로그래머들이 아직도 HTML코드와 PHP코드를 섞어 뷰(view)와 모델(model)을 구분할 수 없고, 유지보수가 힘든 코드를 양산해내는 현실상, 웹 표준이 적용된 결과물을 얻어내기에는 많은 노력과 시간이 소요된다.

디자인과 컨텐트를 분리하는 CSS의 개념처럼 PHP에서도 템플릿 등을 이용하여 뷰와 모델을 분리하는 MVC 기법 등을 적용하면, HTML 코드와 독립된 개발이 가능할 수 있다. 이는 PHP만이 아닌 다른 모든 웹 개발 언어에도 마찬가지로 적용된다.

작업 과정의 재분배

기획/디자인/개발의 기존 역할 외에, 최종 사용자에게 보여질 내용의 코디네이팅을 전적으로 담당할 퍼블리셔(publisher)를 두고 작업 과정을 재분배하도록 한다.

퍼블리셔란 기존의 단순 HTML 코드생산을 담당하던 코딩작업을 벗어나, 웹상에 컨텐트

를 게제하는 방식을 책임지는 전문가를 말한다. 즉, 이전과는 달리, "디자인과 프로그램의 조합으로 웹상에 컨텐트를 보여준다"는 개념을 벗어난, 편집 겸 코디네이터로서의 역할이 추가된 개념이다.

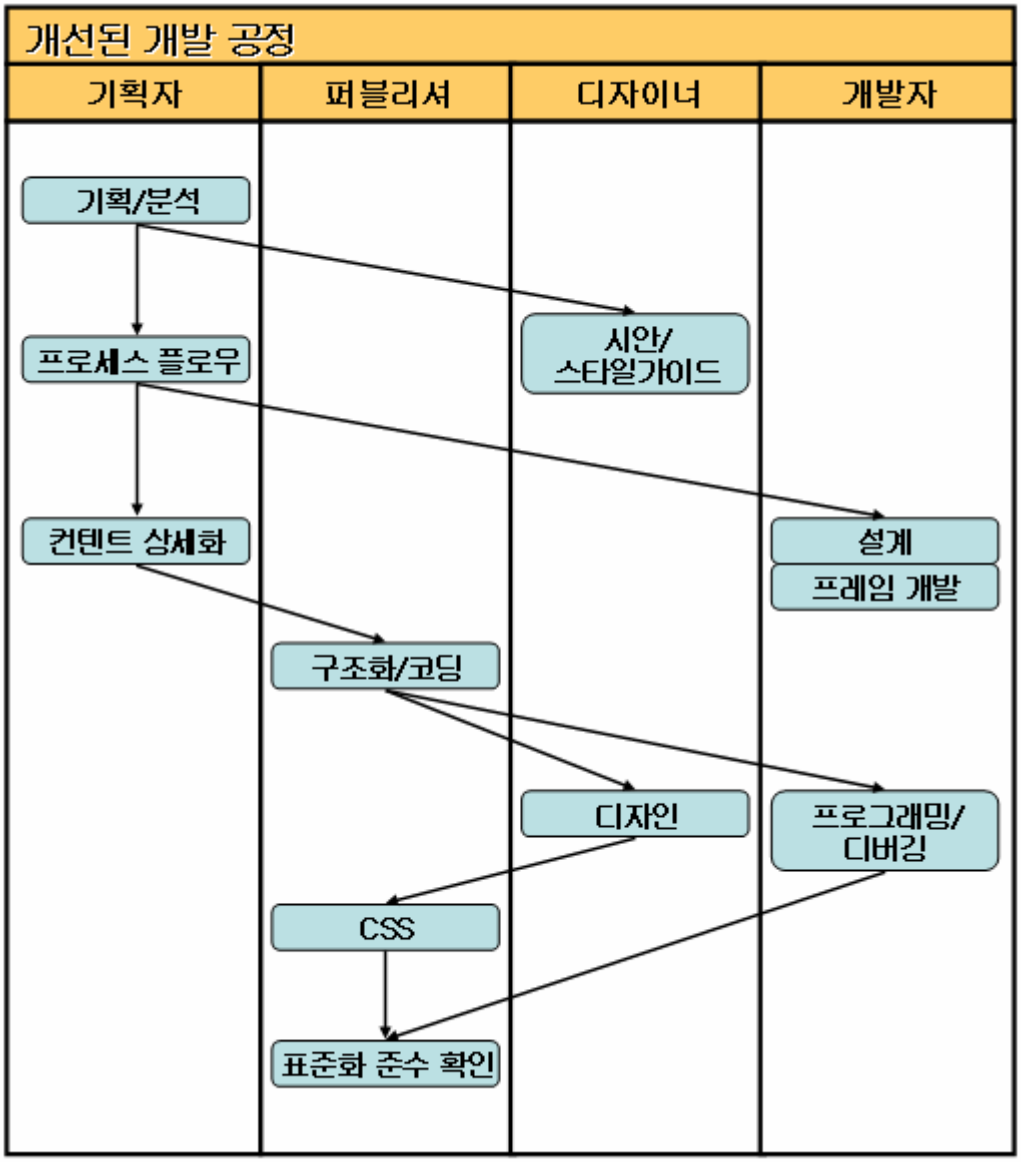


그림 49 개선된 개발 공정 도표

퍼블리셔는, 기획자가 만들어낸 컨텐트를 기획의도에 따라 디자인을 적용시켜 프로그램의 도움을 받아 웹 상에 게제하며, 이를 위해 퍼블리셔는 최종사용자가 대면하는 모든 부분을 책임져야 하므로, 웹 개발의 경우, 최소한 (x)HTML/CSS/JavaScript 부분을 전적으로 담당해야한다. 필요하다면, 프로그래밍에 이용된 템플릿 사용도 가능해야할 것이다.퍼블리셔를 추가함으로써, 기존 역할을 맡은 인력의 업무를 경감시켜주고, 작업 강도와 기간이 단축되는 효과를 기대할 수 있다.

그림은 이상과 같은 개선점이 반영된 개선된 개발 공정 모델로써, 앞에서 소개된 모델들과는 차이를 보이고 있음을 알 수 있다.

우선, 스토리보드 생산 작업을 프로세스 흐름도와 컨텐트 명세서 작업으로 나누어 프로그래머나 디자이너, 퍼블리셔 들의 작업 착수 기간을 앞당길 수 있게 하였다. 또한 스토리보드의 폐기 혹은 경량화를 가능케 하여 전체적인 업무 부담을 줄일 수 있다.

또한 디자인 지시서로서의 스토리보드의 역할을 축소시킴으로써, 디자인을 디자이너의 고유영역으로 보장해준다. 이를 위해서는 기존의 형식적이었던 시안/스타일가이드 작업을 실질적으로 디자인 및 CSS에 사용할 수 있도록 만들어야 한다. 프로그래머의 경우, MVC 기법을 도입하여 스토리보드나 HTML코드의 생산을 기다리지 않고도 전체적인 프로그램 설계 및 개발을 가능하도록 하였다.

다음 장에서는 개선된 모델에 따른 새로운 개발 프로세스의 각 업무에 대한 상세한 설명이 이어진다.

새로운 개발 프로세스

기획/분석/회의

기획이란 기획자의 "혼자만의 상상"을 문서화해내는 것을 말하지는 않는다. 비록 우리나라의 대부분의 사업장에서 알게 모르게 그러한 관행이 이어지고 있지만, 본래 기획이란 의미는 Plan(계획)과 Design(설계)과정을 뜻한다. 이를 위해서는 프로젝트 참여자 모두의 협력이 필요하며, 기획/분석/회의는 그것을 위한 수단이다.

기존 방식의 회의시간은 주로 일방적인 발표와 오류찾기 시간이나 마찬가지였다. 기획자는 며칠간 머리를 싸매고 기획안을 들고 나오고, 다른 참가자들은 그제서야 기획문서를 놓고 예상되는 문제점을 지적한다. 그럴 바에야 애초부터 같이 협력하여 기획을 잡아나가는 것이 효율적이지 않을까?

바람직한 개발 과정은 다음 다섯가지 요소로 정리된다.

- 해결하고자 하는 문제를 프로젝트팀이 명확하게 이해할 수 있도록 한다.
- 한 팀은 여러 가지 역할의 여러 멤버로 이루어짐을 고려한다.
- 각각의 역할을 맡은 멤버간의 의사소통이 원활하도록 한다.
- 프로젝트 진행중에 각 단계간의 피드백을 수용하고 고려한다.
- 클라이언트에게 보여줄 수 있는 작업결과물을 만들어내되, 불필요한 문서화는 하지 않는다.

이를 위한 방법론은 여러가지가 있지만, 여기에서 소개할 것은 그 중 한가지인 **RAD³ : GRAPPLE(Guidelines for Rapid APPlication Engineering)** 방법론의 변형이다. RAD3 : GRAPPLE에 관한 상세한 내용에 관한 설명은 본 문서의 취지와는 부합하지 않으므로, 애플리케이션 공학과 관련된 별도의 자료로 학습하는 것이 필요할 것이다. 여기에서는 RAD3 : GRAPPLE 등의 객체지향 개발방법론을 알지 못하는 국내의 일반적인 상황에서도 통용될 수 있는 방식으로 응용하여 설명하도록 한다.

기획/분석/회의 단계에서 이루어져야 하는 내용들은 다음과 같다,

요구사항 수집 (Requirement Gathering) : 프로젝트의 목적과 의미를 파악하고, 필요한 기능들에 대한 정의와 목록을 만들어낸다. 프로젝트 결과물의 형태와 용도를 명확히 하여 프로젝트 참가자들 모두가 요구사항 및 결과에 대해 동일한 수준의 이해가 이루어져야 한다. 이 과정을 통해 추상적 수준의 로직 분석과 설계구성요소들을 정리할 수 있다. (만약 객체 지향 개발 방법론을 사용중이라면, 활동/클래스/컴포넌트/배포 다이어그램 등이 스케치될 것이다.)

분석 (Analysis) :: 분석은 요구사항 수집 단계에서 부분적으로 시작된다. 수집된 요구사항을 각자의 역할에 맞추어 프로젝트 진행과정에 필요한 요소들을 찾아낸다. 디자이너라면, 어떠한 컨셉의 디자인을 만들어낼 것인가, 프로그래머라면, 필요한 시스템 환경, 데이터 모델 등에 대한 요점을 잡아낼 수 있어야 한다. (객체지향 개발방법론에 따른다면 산출물은 유스케이스/상태/시퀀스/협력 다이어그램등이 될 것이다.)

설계 (Design) : 관점에 따라서는 분석 단계와 설계 단계를 별도로 분리하지 않을 수도 있다. 실질적인 문서화 작업은 여기에서부터 시작되며, 개선된 개발 공정에 따른다면 프로세스 플로우/스타일가이드/개발설계 단계의 초입에 해당한다.

이러한 작업이 가능하도록 기획/분석/회의의 실제 진행은 프로젝트 인원 전체(JAD:Joint Application Development)가 참여하도록 하며, 인터뷰, 브레인 스토밍, 페이퍼 시뮬레이션, 아이디어 스케치 등의 기법을 사용하도록 한다.

기획자 공정

UML

UML(Unified Modeling Language)은 객체지향 설계 분석 전문가인 그래디 부치(Grady Booch), 제임스 럼비(James Rumbaugh), 이바 야콥슨(Ivar Jacobson)에 의해 제안되었으며, 시스템 개발 세계에서 표준으로 인정받은 표기 시스템이다. UML은 기획자에게 클라이언트, 프로그래머, 그리고 개발 과정에 참여한 모든 사람들이 각자의 시점에서 이해할 수 있는 다방면의 설계도를 그릴 수 있는 표준을 제공하며, 제안하는 그래픽 요소를 조합하여 다이어그램을 그릴 수 있도록 되어있다.

흔히, UML은 Java나 C++같은 객체지향 개발도구를 사용할 때 프로그래머들만 사용하는 설계분석도구로 잘못 알려져 있으나, 실제로는 기획자나 시스템 분석가들이 불필요한 문서화 작업 대신, 실제 개발에 바로 사용될 수 있는 지시서를 만드는 데 아주 좋은 도구이다.

예를 들어, 잘 그려진 UML 다이어그램은 그 자체를 바로 OOP(Object-Oriented Programming) 코드로 변환할 수 있다.(그러한 과정을 도와주는 애플리케이션 도구들이 있다.) 복잡하고 지루한 문서화 과정을 거치고, 프로그래머가 다시 이 문서를 해독, 이해한 후 나름대로 분석,설계,개발하는 과정보다 훨씬 빠르고 정확하게 기획의도를 프로그램으로 산출해낼 수 있다는 뜻이다.

현재 UML은 DEC, Hewlett-Packard, Intellicorp, Microsoft, Oracle, Texas Instruments, Rational 등이 주축이 된 UML 컨소시엄에서 발전되고 있으며 OMG(Object Manage Group)의 표준 모델링 언어로 인정받고 있다. 97년 표준 모델링 언어로 채택된 UML 1.1에서 계속 발전하여 현재에는 UML 2.0이 승인된 상태이다. (참고:http://www.omg.uml/)

그렇다면, 과연 웹 기획자가 UML을 반드시 배워야만 하는가?

UML을 사용하기 위해서는 프로젝트 참여자 모두가 UML을 알고 있어야 한다는 전제가 붙는다. 우리나라의 웹 개발 환경의 여건상 완벽한 UML을 웹 개발 분야에 적용하는 것은 현재까지는 무리일. 수도 있다

그러나 UML의 기본 개념을 차용하면 기획자의 기획단계에서의 업무 부담을 경감시킬 수 있으며, 기획 의도를 더 명확하고 확실하게 다른 팀원들에게 이해시킬 수 있기 때문에, 프로그래머의 개발효율을 높여주고, 디자이너나 퍼블리셔의 작업에 영향을 주지 않고도 스토리보드를 경량화시킬 수 있다. 따라서 UML을 직접적으로 프로젝트에 적용시키지 않더라도 기획자들은 UML의 기본 개념에 대한 이해를 갖추어야할 필요가 있다.

스토리보드

현실적으로, 스토리보드가 없으면 웹 개발 자체가 거의 불가능하다는 인식이 국내 웹 개발

분야에 널리 퍼져있는 상황이다. 그러나 앞서 말했듯이 스토리보드는 그 복잡함과 상세함에 비해 실제 개발에서의 효용성은 그다지 높다고 말할 수 없다. 오히려, 스토리보드라도 있기 때문에 겨우 개발이 가능하다는 표현이 더 어울릴 것이다. 다음은 복잡한 스토리보드 대신 UML 기법을 차용한 프로세스 플로우로 대체할 경우이다.

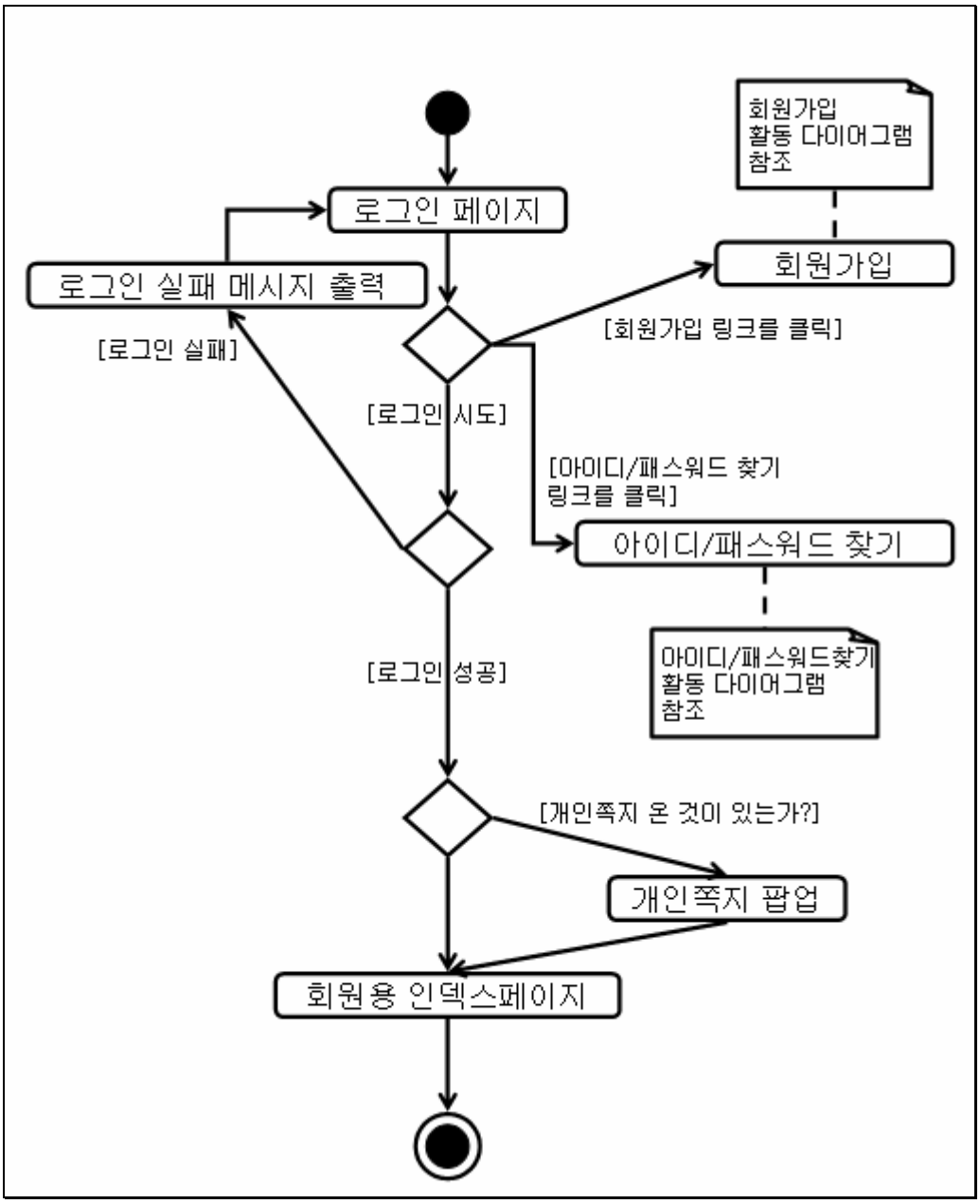


그림 50 스토리 보드 예제

이 프로세스 플로우는 어떤 사이트에서 로그인 과정을 UML의 활동 다이어그램을 이용하여 정리하였다. 기존의 여러페이지로 구성된 스토리보드보다, 이러한 한장의 프로세스 플로우가 개발단계에서의 프로그래머에게는 더 필요한 문서라 할 수 있다.

이러한 프로세스 플로우가 먼저 산출된다면, 프로젝트 참여 인원들은 전체적인 프로젝트의 흐름을 한눈에 알아보기 쉬우며, 자신이 어떤 작업들을 수행해야 하는지 쉽게 확인할 수

있다. 프로그래머라면 비즈니스 로직을 설계하는데 큰 도움이 될 것이고, 디자이너라면 어떠한 페이지들에 어떠한 디자인이 필요할지 미리 알 수 있게 된다.

프로세스 플로우는 반드시 UML 규칙을 따라야 하는 것은 아니며, 의도와 내용을 충분히 확인할 수 있고 팀원간의 의사소통에 문제가 없다면 나름대로의 고유한 방법을 사용해도 충분하다. 여기에서는 예시를 위해 표준규격인 UML을 이용하였다.

이렇게 프로세스 플로우가 먼저 완성이 된다면 기획자에게 필요한 남은 작업은 각 페이지에 들어갈 내용을 정리하는 컨텐트 명세서의 작성이다.

컨텐트 명세서는, 위의 프로세스 플로우에서 미리 선언된 뷰 페이지마다, "이 페이지에는 이러저러한 메뉴가 있고, 이러저러한 내용들이 보여야 하며, 이러저러한 기능들이 있어야 한다." 라는 것을 정리해놓는 문서를 말한다. 역시 마찬가지로 지정된 형식이 존재하는 것은 아니며, 각 개인, 조직, 기업, 프로젝트에 따라 목적에 부합하는 형식이면 충분하다.

페이지 이름 : 영화 정보 서비스 공통 구성요소 설명 : 영화 정보 서비스의 모든 페이지에 대해 다음 요소들을 공통으로 포함한다. 사이트 메뉴 (메일/카페/플래닛/블로그/쇼핑/뉴스/검색/전체보기/로그인) 서비스 로고 서비스 메뉴 (영화홈/상영정보/예매/매거진/재밌는DB/커뮤니티/시사이벤트/마이무비) 검색 (영화검색, 인물검색, 통합검색), 인기검색어 4-5건, 재밌는DB 신규 등록내용 1-2건을 같이 보여준다. 영화 클릭 순위 : daily(default)/weekly 변경가능, 5건 정도 제목과 링크 제공. 상위 1건에 대해 이미지 썸네일 제공 영화기사목록 : 요즘뜨는이영화/Photo & Talk/뉴스매거진 각 5개씩 최근 등록 순서로, 상위 1건에 이미지가 있을 경우 이미지 썸네일 포함. Poll 서비스 크레딧 카피라이트 프로모션 배너 1 프로모션 배너 2 프로모션 배너 3 프로모션 배너 4
페이지 이름 : 개별 영화 정보 보기 URL : ./movieinfo?mkey=영화id 설명 : 이 페이지는 개별 영화 정보 보기 페이지로 검색결과 및 개별 영화 정보의 기본 링크가 된다. 전체 화면배치는 영화사이트 기본 레이아웃을 따른다. (공통 구성요소 및 UI 스타일 가이드 참고) 컨텐트 : 각 영화 정보 보기 페이지 및 그 서브 페이지에 공통으로

전체보기/동영상,포토/영화지식/매거진/네티즌평 의 서브메뉴를 제공한다. 영화 타이틀, 원제, 제작년도, 제작국가의 정보 제공 영화 정보 제공 포스터 / 감독 / 출연 / 관람점수 / 장르 / 개봉일 / 상영시간 / 관람등급 / 관련정보 / 사이트 등 동영상 프리뷰 및 스틸컷 썸네일 4~5장 제공 -> 갤러리 페이지로 링크 평점 관람포인트 : 200자 내외의 텍스트 설명 줄거리 : 400자 내외의 텍스트로 된 줄거리 영화지식 : 해당 지식검색으로 연결되는 링크 모음 매거진 : 해당 뉴스 기사로 연결되는 링크 모음(종류, 기사제목, 출처, 날짜 등 부가 정보 필요) 네티즌리뷰 : 해당 네티즌 리뷰로 연결되는 링크 모음(제목, 작성자, 날짜등 부가 정보 필요) 400자평 보기 : 생략…

위의 내용은 컨텐트 명세서의 일례이다. 좀 더 자세하고 명확하게 서술해야 하나, 여기에서는 예시를 위해 간략히 표기하였다. 실제 현장에서 사용하기 위해서는 형식이나 내용의 추가가 필요할 것이다. . 각 회사의 사정에 따라 내부 문서 규격도 있을 터이고. 디자이너의 창의성과 퍼블리셔의 구조화를 저해하지 않는다면 기존의 스토리보드 형태의 파워포인트 문서래도 상관없다.

이렇게 프로세스 플로우와 컨텐트 명세서를 작성하게 되면 굳이 기존의 스토리보드를 유지할 필요가 없어진다. 프로그래머와 퍼블리셔, 디자이너는 이렇게 만들어진 프로세스 플로우와 컨텐트 명세서만 가지고도 실제 개발작업에 들어갈 수 있으며, 기획자로서는 업무의 양과 시간이 크게 줄어들 수 있는 좋은 방법이라 할 수 있겠다.

물론, 클라이언트에게 보여주기 위한 페이퍼 시뮬레이션으로써 스토리보드가 필요하다면 별도로 작성하는 과정이 필요할 수도 있다. 그러나 예전처럼 스토리보드 하나에 모든 것을 채워넣기 위한 문서화 작업에 큰 수고를 들이지 않아도 될 것이다..

퍼블리셔 공정

구조화

웹 표준화에 맞는 (x)HTML 코드 생성을 위한 컨텐트 구조화 작업을 기획자가 맡을 것인가 퍼블리셔가 맡을 것인가에 대한 질문은 우문이라 할 수 있다. 애초에 명세서 작업시 구조화를 염두에 두고 작성한다면 별도의 구조화 작업은 필요하지 않을 수도 있다.

앞에서 만든 컨텐트 명세서를 바탕으로 실제 (x)HTML 구조화방법을 예시를 통해 익혀보도록 한다.

구조화의 요점은, “덩어리로 분할해서 나누어 공략한다.” Divide & Conquer라는 오래된 – 그러나 확실한 전략을 따른다. 효율적인 작업을 위해서 공통 레이아웃 등은 별도로 작업하는 것이 좋겠지만, 여기에서는 이해를 돕기 위해 한번에 같이 다룬다.

명세서를 받은 퍼블리셔는 해당 페이지의 목적과 컨셉과 스타일에 맞게 내용을 묶어 분할하기 시작한다. 우선, 디자이너가 처음 잡은 스타일 가이드에 전형적인 2단 레이아웃 구조를 이용하기로 결정했다고 가정하면 크게 보아 이 문서는 다음과 같이 러프하게 구조화할 수 있을 것이다. (2단 레이아웃이 아니더라도 사실 대부분 1단계 분할은 다음 형태처럼 되기 마련이다.)

```
* Header 영역
* Content 영역
* Footer 영역
```

2단 레이아웃을 지시했으므로, Content영역은 좀 더 나눌 필요가 있을 것이다.

```
* Header
* Content
  * MainContent
  * SideContent
* Footer
```

퍼블리셔가 파악하기에, 위의 명세서에 들어간 내용들 중 Header에 속하는 것은 다음과 같다.

```
* Header
    * SiteMenu
    * ServiceLogo
    * ServiceMenu
    * Search
    * Promotion_1
    * MovieRank
```

같은 방식으로 나머지를 구조화한다.

```
* Header
    * SiteMenu
    * ServiceLogo
    * ServiceMenu
    * Search
    * Promotion_1
    * MovieRank
* Content
    * MainContent
    * SideContent
            * Promotion_2
            * ArticleBox
            * Poll
            * Promotion_3
```

```
* Footer
    * Credit
    * Copyright
```

이제 대략적인 공통 페이지 구조는 다 잡은 셈이다. MainContent에 들어갈 내용만 페이지 별로 상세화하면 된다.

이 페이지의 주된 컨텐트는 크게 “제목”, “메뉴”와 “내용”으로 나뉘어진다. 그러므로 그에 맞게 구조화하자.

```
* MainContent
    * ContentTitle
    * ContentMenu
    * ContentBody
```

ContentBody에 들어갈 내용은 다음과 같다.

```
* ContentBody
    * MovieInfo
            * Poster
            * Director
            * Casting
            * MovieField_1
            * MovieField_2
            * MovieField_3
            ...
    * Score
    * Point
    * Synopsis
    * Knowhow
...
```

이 구조가 정답이라는 소리는 아니다. 이런 식으로 계층적으로 내용을 분할해 들어갈 수 있다면 다른 방식의 구조화도 가능하다.

어쨌거나, 보이는 바와 같이, 결국 명세서에 써있는 내용을 잘 정리한 것 뿐이다. 애초에, 명세서에 내용을 이런 식으로 정리해놓았다면 별도의 구조화도 거의 필요없다. 그러나 기획자가 처음부터 구조화를 염두에 두지 않고 기획을 진행했다면, 퍼블리셔는 그러한 기획안을 가지고도 위와 같은 구조화 결과를 만들어 낼 수 있어야 한다.

코딩

이제 (x)HTML코딩을 해보자.

```
...
<body>
```

```
<div id="header">
   <div id="sitemenu"></div>
   <div id="servicelogo"></div>
   <div id="servicemenu"></div>
   <div id="search"></div>
   <div id="promotion_1"></div>
   <div id="movierank"></div>
</div>

<div id="content">
   <div id="maincontent">
           <div id="contenttitle"></div>
           <div id="contentmenu"></div>
           <div id="contentbody>
                   <div id="movieinfo">
                           <div id="poster"></div>
                           <div id="director"></div>
                           <div id="casting"></div>
                           <div id="moviefield_1"></div>
                           …
                   </div>
                   <div id="score"></div>
                   <div id="point"></div>
                   <div id="synopsis"></div>
                   <div id="knowhow"></div>
           </div>
   </div>
   <div id="sidecontent">
           <div id="promotion_2"></div>
           <div id="articlebox"></div>
           <div id="poll"></div>
           <div id="promotion_3"></div>
   </div>
</div>

<div id="footer">
   <div id="credit"></div>
   <div id="copyright"></div>
</div>
</body>
</html>
```

이 정도 결과가 나오면 절반 이상 도달한 셈이다. 보면 알겠지만, 위에 "구조화"의 결과를 그대로 HTML코드만 써서 붙인 셈이다.

이제 남은 것은 아직도 비어있는 각 블록의 안쪽을 세세하게 컨텐트에 맞춰 채워넣는 것이다. 원리는 지금까지와 동일하다.

예를 들어 sitemenu를 채워보자.

이 사이트 및 패밀리 사이트들이 공유하는 최상위 메뉴는 다음과 같다.

```
메일, 카페, 플래닛, 블로그, 쇼핑, 뉴스, 검색, 전체보기, 로그인
```

메일~뉴스까지는 패밀리사이트 링크들의 모음이므로 하나의 목록으로 묶을 수 있을 것이다. 검색은 form이 들어가므로 별도.

전체보기는 사이트맵으로 가는 링크이니 앞의 메일~뉴스 링크들과는 성격이 다르고, 로그인 역시 사용자 계정과 관련있는 링크이므로 별도로 분리하는 것이 낫다.

이와 같은 내용을 (x)HTML로 표현하면 다음 같은 구조가 되는 것이다.

```
<div id="sitemenu">
<ul id="familysite>
          <li><a href="">메일</a></li>
          <li><a href="">카페</a></li>
          <li><a href="">플래닛</a></li>
          <li><a href="">블로그</a></li>
          <li><a href="">쇼핑</a></li>
          <li><a href="">뉴스</a></li>
</ul>
<form id="form_search" action="" method="">
          <input type="text" id="txt_search" name="txt_search"
value="" />
                  <input type="image" src="" alt="검색"
id="btn_search" name="btn_search" />
</form>
<div id="link_sitemap">
<a href="">전체보기</a>
</div>
<div id="link_login">
<a href=""><img src="" alt="로그인" /></a>
</div>
</div>
```

믿기지 않겠지만, sitemenu 부분의 코딩은 이걸로 끝났다. 나머지 부분들도 이런 식으로 상세화해나가면 된다.

만약 프로그래머들이 템플릿을 사용하고 있다면, 해당 템플릿 시스템의 템플릿 태그를 익혀 (x)HTML 코드 생성시 퍼블리셔가 직접 템플릿을 생성해주는 과정도 필요하다. 현재의 개발 추세는 점점 템플릿을 이용한 프레임워크 형태로 발전하고 있으며, 최종 사용자에게 보이는 결과물을 책임져야 하는 퍼블리셔의 특성상, 이러한 템플릿 사용법은 웹 퍼블리셔로서 갖추어야될 기본 기술이 될 것이다.

다음은 SixApart사에서 제작한 MovableType이란 blogtool 형태의 CMS 프레임워크에 사용되는 템플릿의 한 예이다. HTML코드에 미리 프로그래머에 의해 만들어진 템플릿 태

그를 같이 사용하여 프로그래밍에 대해 모르더라도 퍼블리셔가 웹 페이지를 구체화하는 방식을 보이고 있다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" id="sixapart-standard">
<head>
   <meta http-equiv="Content-Type" content="text/html;
charset=<$MTPublishCharset$>" />
   <meta name="generator" content="Movable Type <$MTVersion$>" />

   <link rel="stylesheet" href="<$MTBlogURL$>styles-site.css"
type="text/css" />
   <link rel="alternate" type="application/atom+xml" title="Atom"
href="<$MTBlogURL$>atom.xml" />
   <link rel="alternate" type="application/rss+xml" title="RSS 2.0"
href="<$MTBlogURL$>index.xml" />

   <title><$MTBlogName encode_html="1"$>: <$MTEntryTitle
remove_html="1"$></title>

   <link rel="start" href="<$MTBlogURL$>" title="Home" />
   <MTEntryPrevious><link rel="prev" href="<$MTEntryPermalink$>"
title="<$MTEntryTitle encode_html="1"$>" /></MTEntryPrevious>
   <MTEntryNext><link rel="next" href="<$MTEntryPermalink$>"
title="<$MTEntryTitle encode_html="1"$>" /></MTEntryNext>

   <$MTEntryTrackbackData$>

   <MTBlogIfCCLicense>
   <$MTCCLicenseRDF$>
   </MTBlogIfCCLicense>

   <script type="text/javascript" src="<$MTBlogURL$>mt-
site.js"></script>
</head>
<body class="layout-one-column"
onload="individualArchivesOnLoad(commenter_name)">
   <div id="container">
      <div id="container-inner" class="pkg">

         <div id="banner">
            <div id="banner-inner" class="pkg">
               <h1 id="banner-header"><a href="<$MTBlogURL$>"
accesskey="1"><$MTBlogName encode_html="1"$></a></h1>
               <h2 id="banner-description"><$MTBlogDescription$></h2>

… (이하 생략)
```

CSS 스타일 가이드

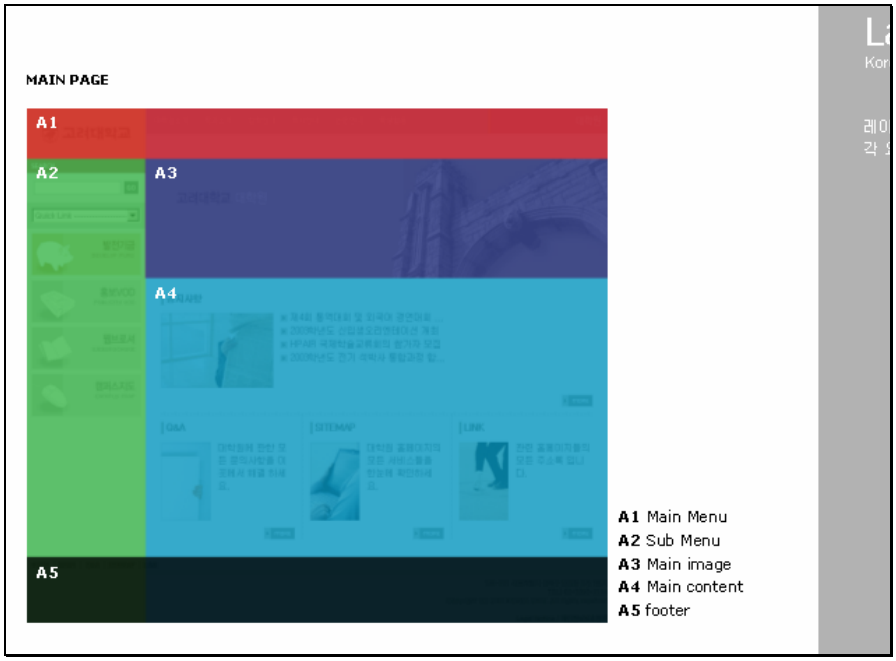


그림 51 CSS 스타일 가이드 예제

위의 그림들은 현장에서 사용되는 스타일 가이드의 일례이다. 물론 스타일 가이드의 형식 역시 각 개인, 조직, 기업, 프로젝트 별로 다양하므로 일률적으로 이렇다라고 단정짓기는 어려우나, 스타일 가이드의 목적 자체가 웹 페이지의 사용자 인터페이스 및 컨텐트 표현 방식에 대한 규칙이기 때문에 이를 CSS로 변환하는 작업이 필요하다.

잘 만들어진 스타일 가이드는 그 자체만으로 전체 CSS 작업의 절반 이상을 줄여 줄 수

있다.

이를 위해서는 CSS의 기본 문법과 규칙을 이해하며, 특히 각 엘리먼트(element)와 셀렉터(selector), 그리고 이용되는 프로퍼티(property)에 대해 충분히 숙지하고 있어야 한다.

여기에서는 간단하게 다음과 같은 스타일 가이드를 CSS로 적용하는 방법을 예시한다.

```
…
박스기사 본문은 다른 영역과 최소 3px 이상 간격을 두고 배치되어야 하며, 1px
크기의 #EEEEEE 색상의 경계선으로 구획지어진다. 경계선과 본문 내용과의 여백은
10px 이상이어야 한다. 배경색은 따로 지정하지 않고 상위 영역의 배경색 및
배경이미지를 그대로 사용한다.
기본 글꼴은 “sans-serif”로 한다. 글꼴 크기는 11.5pt 이며, 들여쓰기 하지 않는다.
본문 중의 링크는 사이트 전체의 일반 링크 표시 규칙을 따르되 밑줄(underline)을
표시한다.
…
```

위처럼 기술된 스타일 가이드를 CSS로 표현하면 다음과 같다.

```
…
.box_article .content{
   margin:3px;
   border:1px solid #EEEEEE;
   padding:10px;
   font-family:sans-serif;
   font-size:11.5pt;
}
.box_article .content a{
   text-decoration:underline;
}
…
```

레이아웃이나 링크 규칙, 이미지 규칙등도 이런 방식에 준하여 CSS로 변환해두면, 나중에 개별 (x)HTML 코드에 적용될 CSS 코드를 작성할 때 작업량이 크게 줄어든다. 반대로 스타일 가이드 작성시, 처음부터 CSS를 염두에 두고 작성하는 것도 좋은 방법이다.

JavaScript

JavaScript의 올바른 사용법은 본 문서의 **3장 – 실전 DOM/SCRIPT 가이드**를 참고하도록 한다.

Validation

완성된 코드가 웹 표준에 부합하는지 여부를 체크하는 방법은 여러가지가 있다. 그 중 대표적인 것을 소개하자면 다음과 같다.

(x)HTML Validator

완성된 코드가 (x)HTML 표준을 따르는지는 웹 표준이 준수되었음을 증명하는 최소한의 조건이다. 많은 (x)HTML Validator가 존재하지만 여기에서는 W3C HTML Validator만 소개하도록 한다.

W3C HTML Validator : http://validator.w3.org/

간단히 이용하는 방법은, Mozilla(Firefox)의 확장기능 중 WebDeveloper Extension을 설치(http://chrispederick.com/work/webdeveloper/) 하거나, MS InternetExplorer 용 인 InternetExplorer Developer Toolbar를 설치' http://www.microsoft.com/downloads/details.aspx?FamilyID=e59c3964-672d-4511-bb3e-2d5e1db91038&DisplayLang=en)하거나, 또는 Opera 브라우저에 기본 기능으로 내장되어있는 HTML Validator 기능을 이용하면 된다.

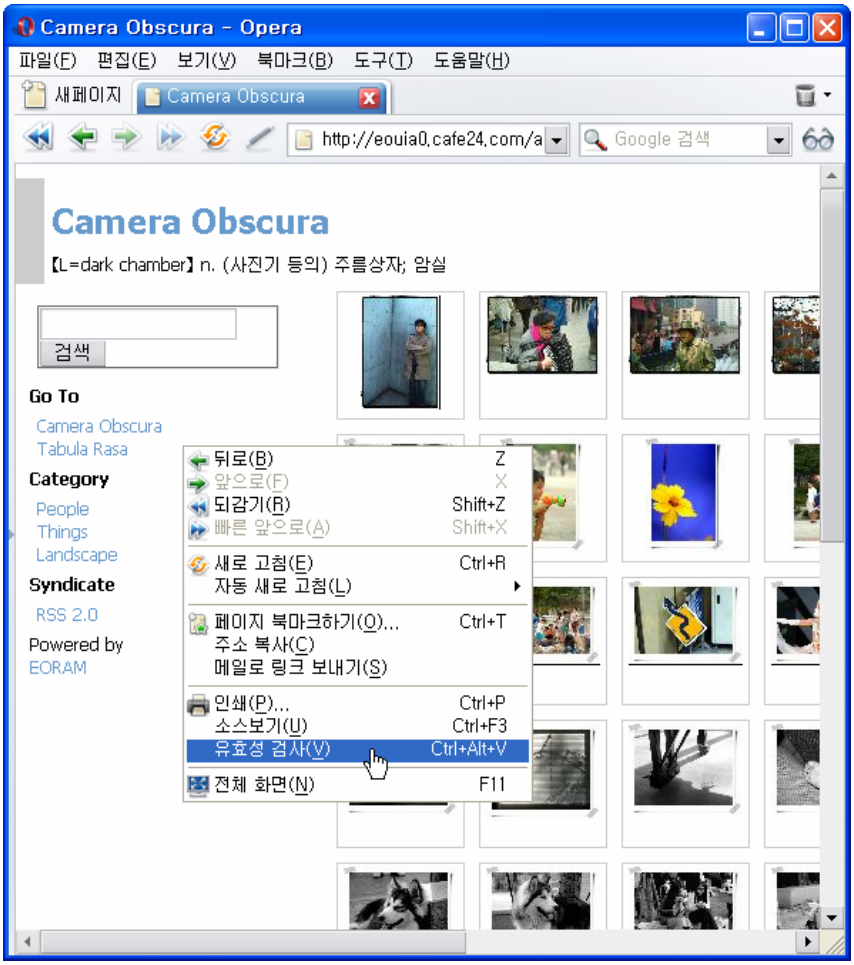


그림 52 Opera에 내장된 디버거

주의할 점은, 웹상에서 링크를 입력하여 사용하는 (x)HTML Validator의 경우, 방화벽을 사용중이거나 동적으로 생성된 페이지에서는 제대로 검사되지 않으므로, 방금 소개한 브라우저 연동 검사기나 별도의 검사 애플리케이션을 이용하는 것이 좋다.

CSS Validator

CSS의 경우에도 (x)HTML과 마찬가지로, W3C의 Validator가 유용하다.

W3C CSS Validator : http://jigsaw.w3.org/css-validator/

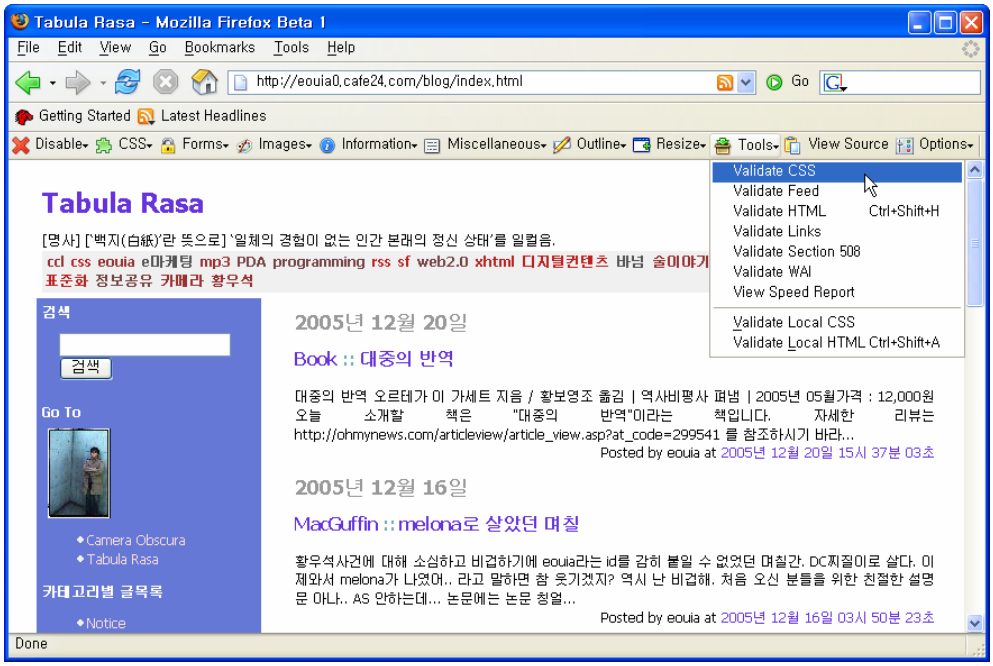


그림 53 Firefox Web Developer Extensions을 이용한 디버깅

역시 마찬가지로, , Mozilla(Firefox)의 확장기능 중 WebDeveloper Extension을 설치 (http://chrispederick.com/work/webdeveloper/)하거나,

MS InternetExplorer용 인 InternetExplorer Developer Toolbar를 설치' http://www.microsoft.com/downloads/details.aspx?FamilyID=e59c3964-672d-4511-bb3e-2d5e1db91038&DisplayLang=en)하거나, 또는 Opera 브라우저에 기본 기능을 이용하는 것이 손쉽다.

접근성(Accessibilty) Validator

웹 표준화가 준수된 후, 실제로 다종다양한 환경에서의 웹 접근성을 확보할 수 있는지 여부를 판단한다.

웹 접근성을 고려하기 위해서는 다음 문서를 참고하도록 한다.

- ❑ 웹접근성을 고려한 콘텐츠 제작기법 (http://www.mozilla.or.kr/docs/web-developer/content_authoring_for_accessibility.pdf)
- ❑ W3C Web Content Accessibility Guideline (http://www.w3.org/WAI/intro/wcag.php)

웹 접근성을 검사하는 도구 역시 다양한데, 그 중 국내에서 개발된 KADO-WAH (http://www.iabf.or.kr/web/kadowah.asp)를 사용할 수 있다.

이외에도 Webxact Accessibility Validator (http://webxact.watchfire.com) 등 강력한

도구들이 있으며 좀 더 많은 목록을 원한다면,

http://www.w3.org/WAI/ER/existingtools.html 을 참조하도록 한다. 대개의 경우 (x)HTML Validator와 CSS Validator를 함께 제공하는 경우들도 많다.

디자이너 공정

UI 스타일 가이드

국내의 디자이너들 중 많은 수가 스타일 가이드를 작성하지 않고 바로 디자인 작업에 들어가는 경우가 있다. 또한 기업이나 프로젝트 팀 단위에서도 스타일 가이드를 요구하지 않는 경우도 있다.

그림 54 다음커뮤니케이션에서 사용하는 UI 가이드라인

그러나 스타일 가이드를 작성하지 않는다면, 사이트 전체의 디자인 일관성을 유지하기가 어려우며, 여러 명의 디자이너가 공동 작업시에도 통일감있는 디자인을 생산해내는 것이 어려워진다. 일반적으로 스타일 가이드에는 다음과 같은 내용들이 포함되어야 한다.

- 디자인 목표, 컨셉
- Color Scheme (사용되는 색상 일람)
- Font, Typography (글꼴, 크기, 색상, 자간, 장평, 행간, 문단형태, 정렬방식 등)
- Layout (문서 구조, 크기, 위치, 형태, 성격 등)
- Graphic Element (아이콘, 이미지, 불릿, 버튼 등의 크기, 형태, 색상, 용도 등)
- 기타 (표, Flash, 멀티미디어 등에 대한 상세한 규칙)

스타일 가이드 작성시 주의해야할 점은, 스타일 가이드가 단지 외부에 전시용으로 구색맞추듯 작성해서는 안되고, 실제 작업에 사용할 수 있도록 세부 사항을 꼼꼼히 기록해야 한다는 것이다. 또 CSS 적용이 가능하도록 CSS를 염두에 둔 스타일 가이드 작성이 되어야 한다. 실제로, 잘 작성된 스타일 가이드는 CSS와 내용이 동일하며, CSS로 기록될 내용을 누구나 보기 쉽게 풀어 설명했다는 개념으로 이해하면 충분할 것이다.

프로그래머 공정

비즈니스 로직 분석

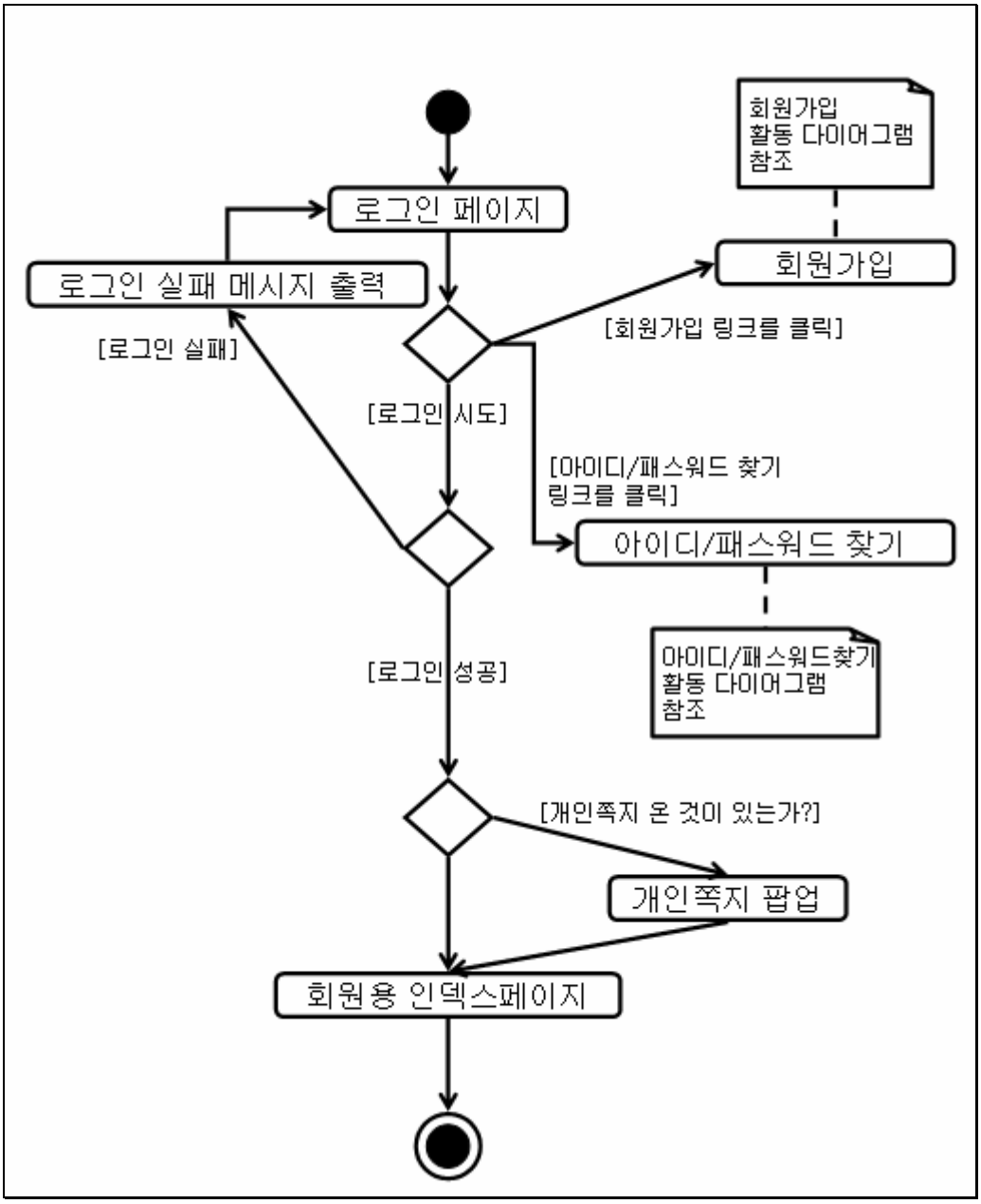


그림 55 비지니스 로직 분석도

앞서 기획자 공정의 프로세스 플로우를 다시 살펴보자. 세세한 데이터 형식과 기능 구현은 컨텐트 명세서 및 (x)HTML 코드를 함께 살펴보아야겠지만, 전체적인 비즈니스 로직의 설계는 프로세스 플로우만으로도 가능하다.

예를 들어 위 그림의 경우에는,

- 각 페이지 출력
- 로그인에 필요한 데이터 모델
- 사용자 입력값 전달
- 로그인 판정
- 개인쪽지에 필요한 데이터 모델
- 페이지 전환
- 팝업 기능
- 에러 처리 등

등의 분석이 가능하다. 이를 바탕으로 DataBase 설계나, 프레임워크의 구현, 결과 페이지의 시뮬레이션, 유스케이스 시나리오 생성 등의 작업을 진행시킬 수 있다. 스토리보드가 완성되어야만 분석이 가능했던 기존 방식에 비해 일정 초기부터 프로그램 작업에 들어갈 수 있으므로 상당한 시간적 여유를 확보할 수 있다.

MVC 모델

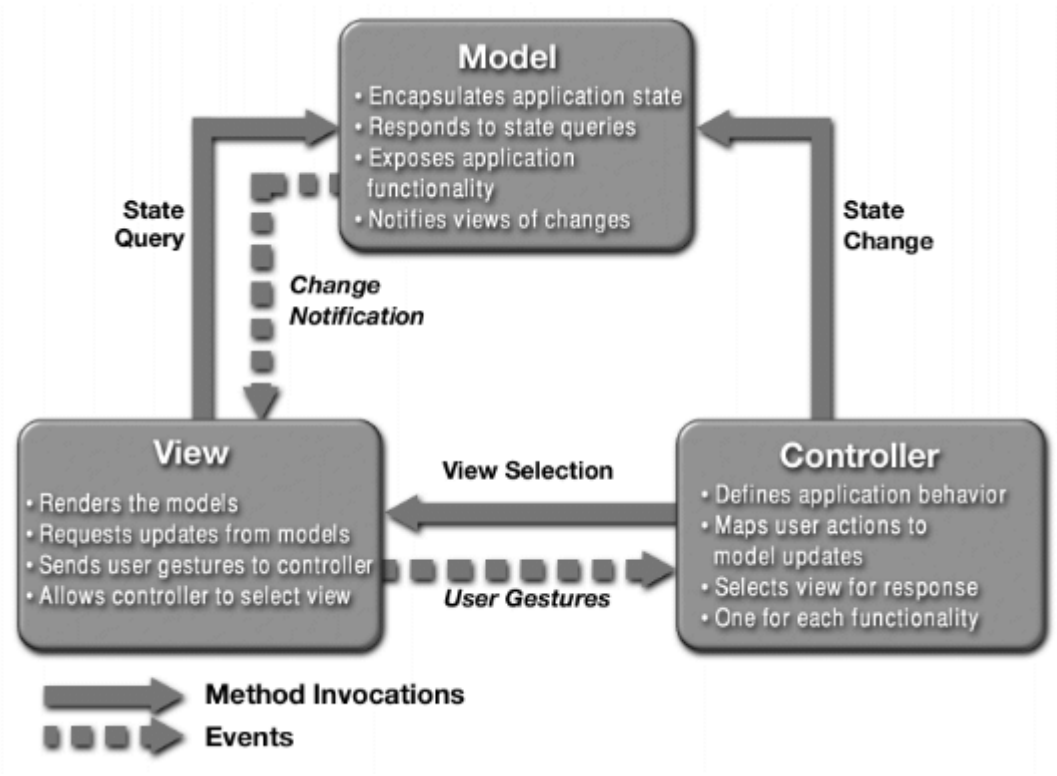


그림 56 MVC 모델 설명도

MVC 모델이란 Model-View-Controller의 세 가지 구성요소로 시스템을 구축하는 방법

을 말한다.

웹 개발 과정에 대응하여 해석하자면,

- View : 최종 사용자에게 보여지는 HTML 결과물
- Model : 인터페이스와는 상관없이 정해진 기능들을 수행하고 그 결과를 갱신하는 프로그램
- Controller : 사용자의 Action 을 받아 해당하는 Model 을 수행시키기 위한 관리 프로그램

이라 해석할 수 있다.

애플리케이션 개발에서는 이미 널리 알려진 방법론이며, 웹 개발에 있어서도 Java 나 .NET 진영 등을 중심으로 이를 이용한 프레임워크나 기법이 많이 개발되어 왔다. 그러나 PHP, ASP, PERL등에서 상용화된 MVC 프레임워크가 없다 하여도, 이 개념을 이용하면 훨씬 구조적이고 용이한 개발이 가능하다.

MVC모델을 도입하기 위해 가장 기본이 되는 개념은 Model과 View를 분리해야한다는 점이다. 웹 개발의 경우라면, 사용자에게 보여줄 HTML 코드를 출력하는 부분과 데이터를 처리하는 부분은 분리되어야 한다고 표현할 수 있다.

PHP나 ASP, 심지어 JSP 프로그래머 중에도, HTML코드 안에 각종 스크립트 처리문을 그대로 코딩하는 프로그래머들이 많이 있는데, 이는 프로세스의 흐름을 확히 파악하는데 어려움을 주며, 프로그램과 HTML코드가 분리되지 않아 유지/보수가 용이하지 못하고, 코드의 가독성을 떨어뜨리는 데 일조를 한다. 이러한 개발 방식은 HTML코드에 의존하기 때문에, 애써 퍼블리셔를 두더라도, 퍼블리셔의 HTML코드 산출을 기다리느라 프로그래밍 단계에서 작업이 지연되는 결과를 가져온다.

예를 들자면 로그인 프로세스의 경우

로그인 form 페이지 출력 -> 입력받은 ID/Password 를 확인하여 결과 처리

보다는

로그인 form 페이지 출력 -> *입력받은 ID/Password 확인* -> **로그인 결과 페이지 출력**
(볼드체가 view / 이탤릭체가 model 에 해당)

방식이 MVC모델을 도입하여 좀 더 효율적인 개발이 가능해질 수 있다. 템플릿을 활용하는 것도 이러한 MVC모델 개념을 이용한 개발 방법 중 하나이다.

맺음말

우리나라에서 어느 정부기관이 조사한 바에 따르면, 각 운영체제별 웹 브라우저에 따른 정부 및 공공기관, 금융기관의 정보접속성 현황은 대부분의 웹 사이트가 윈도우즈 환경 하에 익스플로러에 최적화 되어 리눅스 및 맥 OS 사용자는 정보접근에 제약이 따르는 것으로 나타났다. 특히 인터넷뱅킹과 관련한 문제는 윈도우즈 기반의 공인인증서만을 현재 대부분 금융기관에서 사용하고 있어, 다른 운영체제를 사용하는 사용자들은 인터넷뱅킹을 위해서 윈도우즈 운영체제로 다시 접속해야 하는 번거로움이 있다.

이것은 표준 기술에 대한 이해 없이 시장 기술에 따라 인터넷 산업이 이끌려 옴에 따라 생긴 부작용이라고 할 수 있다. 웹 개발자들이 자신도 모르는 사이에 표준에 어긋나는 개발을 하게 되는데, 이것은 표준안에 대한 재교육과 학습과정이 결여되어 있었던 이유이기도 하다. 이 가이드는 표준안에 대한 완벽한 설명을 담고 있지는 않지만 적어도 각 웹브라우저의 차이로 인해 야기되는 문제를 거의 대부분 다루고 있으며 이를 해결할 수 있는 방법을 제시하고 있기 때문에 이런 점들을 꼭 숙지한다면 보다 접근성이 향상된 웹사이트가 제작될 수 있을 것이다.

웹개발자가 표준 환경에 맞는 웹사이트를 구축할 수 있으려면 기본적인 마인드의 전환이 필요하다. 먼저 내용(Content: html/xhtml/xml)과 그 표현 방법(Presentation Method : CSS/XSL), 행동 양식(Behavior: DOM Scripting)을 엄격하게 분리하여야 한다. 이것이 명확해야만 웹 사이트의 유지, 관리, 보수가 용이하고, 장치 독립성, 플랫폼 독립성, 접속 방법 독립성, 장애 정도와 무관한 내용에 대한 접근 가능성이 보장된다.

Tim Berners-Lee가 제창한 웹의 기본 정신은 내용에 대한 '보편적 접근 가능성'(상호 운용성, 플랫폼/장치 독립성 등을 포함해서)이다. 보편적 접근 가능성(Universal Accessibility)는 결코 글자 모양, 페이지 폭, 색깔 등이 언제 어디서나 다 똑같이 보여야 한다는 것을 의미하는 것이 아니다. XML/RDF 등을 이용한 Symantic Web의 구현에도 보편적 접근 가능성은 중요하게 적용 되고 있다.

웹은 계속해서 발전하고 있다. 그러나, 한국의 웹은 상업성과 화려함에 가려져 웹이 처음 만들어 졌던 기본 정신을 외면하고, 세계적인 표준 동향을 바로 찾아가지 못한 채 한국 내부의 웹으로 전략하고 있다. 이 가이드가 국내 웹 환경의 접근성과 브라우저 호환성을 좀 더 높이는 계기가 되기를 희망한다.

부록. 웹 표준 브라우저 호환표

웹 브라우저 현황

국내의 대부분 웹사이트들은 IE를 기준으로 만들어졌다는 비난 아닌 비난을 받고 있다. 대부분의 사용자들이 IE를 쓰는 만큼 3~4% 내외인 비 IE 사용자를 위해 웹페이지나 서비스를 바꾼다는 것은 쉽지 않다는 것은 사실이다. 그러나, 최근 모질라 파이어폭스가 해외에서 IE의 시장 점유율을 90% 이하로 끌어 내리며 15~20%의 시장점유율을 획득하기 시작했다. 또한, 맥킨토시에서도 MS가 IE5.2Mac 버전을 더 이상 지원하지 않기로 결정함으로서 사파리 브라우저에 대한 사용도가 늘어 나고 있다.

2005	IE 6	IE 5	Ffox	Moz	NN 7	O 8	O 7
October	67.5%	6.0%	19.6%	2.6%	0.4%	1.2%	0.2%
September	69.8%	5.7%	18.0%	2.5%	0.4%	1.0%	0.2%
August	68.4%	6.3%	18.9%	2.4%	0.4%	0.8%	0.3%
July	67.9%	5.9%	19.8%	2.6%	0.5%	0.8%	0.4%
June	65.0%	6.8%	20.7%	2.9%	0.6%	0.7%	0.5%
May	64.8%	6.8%	21.0%	3.1%	0.7%	0.7%	0.6%
April	63.5%	7.9%	20.9%	3.1%	0.9%	0.4%	1.0%
March	63.6%	8.9%	18.9%	3.3%	1.0%	0.3%	1.6%
February	63.9%	9.5%	17.9%	3.3%	1.0%		1.7%
January	64.8%	9.7%	16.6%	3.4%	1.1%		1.9%
2004	**IE 6**	**IE 5**	**Moz**	**NN3**	**NN 7**	**NN 4**	**O 7**
December	65.5%	9.9%	17.0%	0.2%	1.2%	0.2%	1.8%
November	66.0%	10.2%	16.5%	0.2%	1.2%	0.3%	1.6%
October	67.3%	10.8%	14.7%	0.3%	1.3%	0.3%	1.6%
September	67.8%	11.2%	13.7%	0.3%	1.4%	0.3%	1.7%
August	67.0%	13.0%	12.7%	0.4%	1.4%	0.4%	1.6%

그림 57 브라우저 시장 점유율 (2005.10현재)

게다가 파이어폭스의 성공에 힘입어 오페라 브라우저도 무료 배포로 전환함으로서 웹 브라우저들 간의 새로운 신선한 경쟁의 바람이 불고 있다. 이 경쟁은 과거와 달리 보다 나은 웹 표준 기술을 선보여 사용자의 관심을 불러 일으키는 것이다. 이 장에서는 대표적인 웹 브라우저들을 소개하고자 한다.

인터넷 익스플로러7

마이크로소프트는 파이어폭스의 시장 점유율이 높아지자 위기감과 아울러 새 OS인 비지티에 탑재할 새로운 IE 개발에 착수하고 최근에 윈도우 비스타 베타1과 함께 인터넷 익스플로러(IE) 7을 함께 선보였다 IE7의 가장 눈에 띄는 특징은 보다 깔끔해진 툴바라 할 수 있다. 이전, 다음 페이지를 오가기 위해 "Back", "Forward"와 같은 버튼으로 구분된 목록을 갖는 대신에, 리스트를 통해서 최근에 본 페이지들을 하나로 결합해서

보여준다. 이와 같은 방식은 페이지간의 이동을 보다 논리적으로 만들뿐만 아니라 도구바를 깔끔하게 만든다.

IE7에서는 웹 브라우저에서 가장 많이 찾는 기능인 탭 브라우징을 갖추게 되었다. 파이어폭스, 오페라, 사파리 등 경쟁 브라우저가 이미 탑재한 기능인 탭 브라우징을 사용하면 하나의 창에서 여러 웹 페이지를 볼 수 있으며, 링크에서 오른쪽 클릭해서 "새 탭 링크 열기"를 선택해서 새로운 탭에서 링크를 볼 수 있다

또한, IE7의 유용한 기능은 RSS(Really Simple Syndication)를 지원하는 것이다. IE7의 다음 베타 버전(베타2)에서는 RSS 0.9x, 1.0, 2.0, Atom 0.3/1.0을 지원하게 될 것이다. 현재 베타 버전에서는 Atom을 지원하지 않는다. IE7에서는 인기있는 검색 엔진을 사용하는 검색 기능을 내장했다.(그림12) 원하는 검색 엔진을 선택할 수 있다. 직접 다른 검색엔진을 선택하지 않는한 기본 검색 엔진이 항상 사용될 것이다. 원하는 검색 엔진을 기본값으로 설정하기 위해 Search Settings를 선택할 수 있다.

IE7 베타1에는 피싱(Phishing) 필터를 포함하고 있다. 주소를 로드할 때마다, IE7은 피싱사이트로 알려진 블랙리스트 URL 데이터베이스와 입력된 주소를 비교한다. 피싱사이트로 결정되면 차단할 것인지, 보고할 것인지를 선택할 수 있다. 피싱 필터는 윈도 XP버전의 IE7 베타1에서만 이용할 수 있다. 윈도우 비스타의 IE7에서 이 기능을 지원하는 시기는 베타2가 될 것이다. IE7 작업이 끝난 것은 아니지만, 베타1을 통해 향상된 브라우저와 새로운 기능들을 미리 살펴볼 수 있다. IE7 출시 임박은 브라우저 메이커들에게 자신들의 브라우저를 개선하게 하는 자극이 될 것이다.

모질라(Mozilla) 계열 웹브라우저 : 파이어폭스

비 IE계열의 대표적인 웹브라우저가 Netscape이다. Netscape는 4.58버전일 끝으로 Navigaor라고 불리는 브라우저 시대를 끝내고, 소스를 공개함으로서 공개 소프트웨어로 전환하였다. 이 공개 소프트웨어 프로젝트를 모질라(Mozilla)라고 명명하고, Gegko라는 브라우저 엔진을 통해 웹브라우저를 발전시켜 왔다. Netscape6/7 버전은 Mozilla의 1.0.2, 1.4를 기반으로 한 것이며 모질라 기반 브라우저라고 할 수 있다. Mozilla, Netscape, Mozilla Firebird, Kameleon 등은 모두 모질라 기반 브라우저로서 모질라에 대한 정보만 제공하여도 비 IE 사용자의 상당수를 지원할 수 있다.

모질라에서 브라우저 설정을 어떻게 하며 어떤 폴더에 저장되는지 궁금해 하는 경우가 있다. 과거에 Netscape Communicator를 설치 해본 사람이라면, 프로필이라고 하는 개념은 친숙할지도 모른다. 모질라 기반 브라우저에서 프로필은 북마크 주소장, 캐시 메일 사용자 정의 설정등의 개인적인 데이터를 정리해 보존해 두는 폴더이다. Mozilla 는 복수의 프로필을 사용할 수 있게 되어 있어 프로필의 변환에 프로필 관리자라고 하는 전용의 컴퍼넌트를 사용한다. 프로필 관리자는 Mozilla 프로그램의 실행전 혹은 종료후가 아니면 실행할 수 없다.

파이어폭스

모질라 파이어폭스 (Mozilla Firefox)는 모질라 프로젝트에서 떨어져나온, 게코 엔진 기반의 작고 가벼운 자유 소프트웨어 웹 브라우저이다. 모질라가 웹 브라우저 및 전자 우편 관리 기능 등을 포함하면서 확장해나가, 덩치가 매우 커졌다. 그래서 많은 사람들이 보다 가벼운 웹 브라우저를 원함에 따라 웹 브라우저만 따로 떼어내어 피닉스(phoenix)를 만들었다.

나중에는 하위 프로젝트인 피닉스에서 기술을 개발하여 나중에 모질라에 적용시키며 앞서 나갔다. 상표권 문제로 Firebird, Firebird™, Mozilla Firebird로 바꾸다가 결국은 Mozilla Firefox로 이름을 바꿨다. 2004년 현재 모질라는 개발을 중단하고 웹 브라우저인 모질라 파이어폭스와 전자 우편 관리 프로그램인 썬더버드, 두 하위 프로젝트를 계속 진행 중이다.

파이어폭스는 2005년 11월 출시 이후 1년 만에 1억 다운로드를 기록하였고 전 세계 시장 점유율 10~15%를 기록하고 있는 인기 있는 브라우저이다.

주요 특징

- 탭 브라우징, 팝업 광고 차단
- 700여개의 사용자 확장 프로그램 및 다양한 동적인 테마의 전환
- 개인 정보 보호 및 보안 및 폼의 자동 완성 기능
- 빠르고 편리한 사이드바, 툴바의 검색창, 라이브 북마크 기능
- Canvas 기능을 통해 2D/3D 그래픽 기능 브라우저에 내장
- SVG(Scalable Vector Graphic) 표준 브라우저 내장 기능으로 탑재
- CSS2, CSS3, Javascript 1.6의 새로운 웹 표준 기능 지원

오페라 브라우저

오페라(Opera)는 1990년 초, 노르웨이 통신회사, 텔레너의 연구소에서 근무하던 3명의 로부터 시작하여 1995년에 오페라소프트웨어가 설립되었다. 그리고 1996년에 윈도우 오페라 2.1의 오페라 최초 버전이 발표되었다. 1998년에, 오페라는 윈도우를 넘어 다른 플랫폼으로 오페라 브라우저를 확대하여 2000년과 2001년에 Linux, Macintosh, BeOS, Symbian OS (EPOC) 및 QNX와 같은 대중적인 플랫폼에 대해 출시되었다. 2000년 12월에 윈도우 오페라 5 버전이 광고가 지원되는 무료 버전으로 출시되었다. 첫달에 무료 오페라 5 버전은 2백만개가 다운로드되어 설치되었다. 오페라는 개발 초기부터 W3C의 표준 사양을 준수하고, 브라우징 속도를 가장 빠르게 한 특징을 가지고 사용자 층을 이끌어 나왔다.

오페라 한글판이 출시되기 앞서 이 한글 언어팩을 이용하여 한글로 오페라를 사용할 수 있다. 한글 언어팩을 클릭하여 다운로드 받은 후 오페라가 설치된 Opera 디렉토리에 압축을 풀면 된다. 다음에 오페라를 실행하여 "파일(File)/환경설정(Preferences)/언어(Language)"에서 사용자 인터페이스 언어 설정에서 옆에 버튼을 누르고 Opera 디렉토리 안에 owxxx_xxxxko.lng 파일을 찾아 선택하고 확인 후 설정한다. 그리고 오페라를 다시

시작하면 한글로 오페라를 사용할 수 있다.

오페라 사용시 웹페이지 내에서 한글을 제대로 표현하려면 다음과 같이 한다. 메뉴바 파일(File) -> 환경설정(Preferences) -> 글꼴 및 색(Fonts and colors)를 선택하면 오른쪽 화면 나의 글꼴 및 색(my fonts and color)에서 먼저 일반글자(normal) 선택하고 변경을 누르면 대화창이 나타나는데 거기서 글꼴은 TT굴림 또는 TT돋음을 선택하고 유형에서는 보통, 크기는 10 또는 11(최적) 로 선택하고 확인을 선택한다. 그리고 다시 환경설정 창에서 아래 적용을 눌러 설정한다. 다른(css font-family 를 제외한) 항목들도 같은 방법으로 설정한다. 다음에 [환경설정-언어]에서 인코딩유지 항목에서 html 을 euc-kr 로 선택하고 설정하여 사용한다. 메뉴바, 도구모음 글꼴도 변경할 수 있다. 환경설정> 브라우저 모양에서 "글꼴 및 색" 항목에서 '시스템 기본설정 사용' 에 체크 지우고, "일반 텍스트", "비 활성 텍스트", "북마크창 글꼴"를 각각 누르고 차례로 글꼴을 바꾸면 된다. 크기는 동일하게 "9" 로 하면 되며, 글꼴 색상도 변경 가능하다.

- 오페라 브라우저를 사용할 때, 어떤 페이지는 다른 브라우저와 다르게 표시되는 것을 볼 수 있다. 대부분의 경우에서 그 차이는 그 표시되는 페이지에서 표준을 지원하지 않는 오류에 의해 나타난다. 어느정도까지 오페라는 넷스케이프 및 인터넷 익스플로러로 표시되는 오류를 그대로 복제하도록 시도하지만, 오페라는 우선적으로 표준 체계를 그대로 적용한다. 오페라와 넷스케이프/ 인터넷익스플로러 사이에 표시되는 차이에 대해서는 아래와 같다.
- 오페라에서 <HR>의 색은 배경 특징 이다, 그래서 오페라는 생성하는 내용 뿐만 아니라 모든 배경스타일을 승인한다. NN4 및 IE에서 그것은 전경 특징이다 (색). NN6은 역시 바르게 한다 (전에 내용을 생성하고 사용된 후라면 오페라와 NN6 사이에는 차이가 있다).
- 링크 밑줄은 오페라 및 NN6 이전 버전에서와는 다르다 (하나의 색으로 밑줄이 사용되고, 텍스트는 다른 색으로 사용된다). 이것은 텍스트에 대한 CSS2의 결과이다.
- IE5/Windows는 오페라 (및 NN6)에서 상자를 보다 크게 보이게 하는 높이와 폭에서 오류가 있다. 이것은 표준 양식으로 IE6에서 수정되었다.
- IE5는 또한 위치에 대해 문제를 갖고 있다. 위치한 요소는 포함하는 요소가 아닌 가장 가까이 위치한 요소에 위치되어야 한다. 배경 이미지의 위치는 창이 아닌 요소 상자와 관계한다. 이것은 오페라에서 body 와 함께 위치한 이미지 (background-position: center)는 창의 중앙에서가 아닌 페이지의 중앙에서 보기 좋지 않게 나타난다는 것을 의미한다.
- 보통 padding은 그 body 요소에 적용된다. 그리고 여백을 두지 않는다 (body 및 head/html 요소 사이의 여백).

오페라에 대한 소개 및 한글 지원 페이지는 http://opera114.pe.kr 를 참고하면 된다.

사파리

사파리(Safari)는 애플 컴퓨터가 자사의 맥 오에스 텐(Mac OS X) 운영체제를 위해 개발한 웹 브라우저이다. 사파리는 맥 오에스 텐 v10.3(팬서)의 기본 브라우저로 포함되었고, 맥 오에스 텐 v10.4(타이거)에서는 기본 탑재되어 있는 유일한 브라우저이다.

사파리는 아이튠즈 음악 감상 소프트웨어와 유사한 북마크 관리 체계를 가지고 있고, 애플의 퀵타임 멀티미디어 기술과 통합되어 있으며, 모질라와 유사한 탭 브라우징 인터페이스를 사용한다. 구글 검색 상자는 사파리 인터페이스의 기본 요소이며, 웹 주소 자동완성과 웹 페이지 텍스트 영역의 맞

춤법 검사를 지원한다.

1997년까지 애플 매킨토시 컴퓨터는 넷스케이프 네비게이터를 기본으로 제공해왔다. 이후 마이크로소프트의 맥용 인터넷 익스플로러가 기본 브라우저로 포함되었다. 그러나 2003년 6월에 사파리 출시에 따른 마이크로소프트의 대응은 맥용 인터넷 익스프로러의 개발중단 선언이었다. 넉달 뒤에 맥 오에스 텐 v10.3에 맥용 인터넷 익스플로러가 들어 있긴 했지만 기본 브라우저에 대한 대체 브라우저로써 포함된 것이었다. 맥 오에스 텐 v10.4의 도래와 함께, 사파리는 이 운영체제에 포함된 유일한 웹 브라우저이다.

사파리는 웹페이지 렌더링 및 자바 스크립트 실행에 애플의 웹키트를 사용한다. 웹키트는 웹코어(컹커러의 KHTML 엔진에 기반한 것)와 자바스크립트코어(KDE의 kjs 자바스크립트 엔진에 기반한 것)로 구성되어 있다. KHTML과 kjs와 마찬가지로 웹코어와 자바스립트코어는 자유 소프트웨어이며, LGPL(약소 일반 공중 사용 허가서) 라이선스로 배포된다. KHTML 코드로부터 애플이 개선한 일부 코드는 컹커러 프로젝트에 합쳐진다. 애플은 또한 2절로 된 BSD 라이선스와 유사한 오픈 소스로 추가 코드를 공개한다.

2005년 4월 29일에 나온 사파리 2.0판은 RSS와 Atom 읽기 기능을 내장하고 있다. 다른 기능으로는 보안 브라우징, 웹페이지의 저장 및 이메일 전송, 북마크 검색 기능을 들 수 있고, 1.2.4판에 비해 1.8배의 속도 증진이 있었다는 보고가 있다.

사파리의 현재 개발자 버전은 Acid2 테스트, 즉 CSS2의 일부 기능(특히 에러 핸들링 부분에서)을 점검하는 테스트에 통과한 최초의 브라우저이다. 그렇지만 이와 같은 개선점은 웹키트 소스를 다운로드해서 컴파일해야 얻을 수 있기 때문에 아직은 실사용자들과는 거리가 멀다.

장애인 웹 접근성 체크 리스트

웹 표준 범주에는 레이아웃 및 기술적 공통성을 추구하는 면이 있는 가 하면, 일반적이지 않는 웹사용자에 대한 지원이라는 포괄적인 의미도 함축하고 있다. 예를 들어, 청각 장애자나 시각 장애자가 웹페이지를 보기 위해 필요한 것들이나 어린이 노약자를 위한 배려 같은 것들이 그것이다.

이러한 기능 옵션에 대한 중요한 사항은 웹 사이트를 기획 운영하는 웹마스터와 웹디자이너 및 개발자들이 http://www.w3.org/TR/WAI-WEBCONTENT/에 제시되어 있는 'W3C web accessibility initiatives'의 규정한 지침에 유의해야 한다.

1) 텍스트

- ❑ 핵심 정보는 반드시 텍스트/HTML 포맷으로 제공되어야 한다. 특히 Flash 같은 것으로 전체화면을 구성하거나 메뉴를 구성하는 것은 피해야 한다. 만약 꼭 사용해야 한다면 비 Flash 버전을 만들어야 한다.
- ❑ 텍스트는 반드시 사용된 배경색에 대해 뚜렷이 대비되는 색으로 표시되어야 한다. (다양한 환경의 256 COLOR 지원 그래픽 카드에서 식별 가능여부가 테스트 되어야 한다)
- ❑ 텍스트 색상은 텍스트를 표시하는 곳에서 사용자가 원하는 색상을 선택할 수 있으므로, 색상별로 별도의 의미를 함축하지는 않는다.

2) 폰트 설정(Font)

- ❑ 글자에 대한 형식은 <font> Tag를 사용하기 보다는 CSS을 통해 지정해서 사용한다. HTML4.0에서는 FONT를 사용하는 것을 추천하고 있지 않다.
- ❑ CSS에는 일반적으로 사용 가능한 글자꼴을 Face 속성에서 지정해야 한다. 예를 들면, 굴림, 굴림체, 돋움, 돋움체 등 Arial, Helvetica, Times New Roman등이 있다. 또한, 가변폭과 고정폭의 글꼴 선택에 있어 글자의 크기를 사용자가 임의로 조정할 수 있도록 가변폭 글꼴을 우선한다.
- ❑ 영문의 경우 모두 대문자로 표기하거나 이탤릭체를 과도하게 사용하는 것은 피해야 한다. 밑줄 친 글자는 하이퍼링크와 혼동될 우려가 있으므로 사용을 피한다.
- ❑ 색상 속성은 인쇄 시 나타나지 않으므로 흰색이나 지나치게 밝은색으로 설정하지 않으며, 쉽게 읽을 수 있도록 배경색과 대비가 되어야 한다. 특히, 'Color'가 특정 의미 부여의 유일한 방법이어서는 안 된다.
- ❑ 어떤 정보가 특정 글꼴로 표현되어야 한다면, 해당 정보는 이미지로 표현되어야 하고 텍스트 형식의 ALT 값을 제공해야 한다. 정보를 표현하는데 이미지를 사용하는 것은 최소화해야 한다.

3) 테이블 (TABLE)

- ❑ 브라우저에 따라, 특정 사용자의 경우에는, 복잡한 테이블을 생성하는 것이 어렵거나 레이아웃이 틀려 보이는 경우가 매우 많다. 따라서, <TABLE> 태그 방식의 레이아웃 보다는 <DIV>와 CSS과 접목된 레이아웃 방식으로 변경하도록 노력한다.
- ❑ 웹페이지 내에 테이블을 아예 사용하지 않는다는 정책을 고집하는 것은 현실적으로 불가능하므로, 최소한 디자이너는 복잡한 테이블 사용 시 일어날 수 있는 문제에 대해 인식하고 있어야 한다.

- 컬럼 수는 최소로 효과적으로 유지해야 하고, 중첩 테이블은 가능한 한 피하고, 다른 대안이 없는 경우 사용한다
- 테이블 내의 정보는 가능하면 수평으로 읽혀져야 하며, 서로 다른 웹 브라우저에 따라 가능한 동일하게 표현되어야 하고 호환성이 확인되어야 한다.
- Ending 태그는 절대 생략해서는 안 되며, 셀 내의 배경 이미지는 구버전의 브라우저에서는 지원되지 않으므로 피해야 한다.

4) 대안 TAG의 정의

- <img>, <applet>, <input> ,<object>, <applet> 태그 등에는 이미지를 보지 않거나 볼 수 없는 사용자나 검색 엔진 위치설정에 매우 유용한 ALT나 LONGDESC, TITLE 같은 텍스트 정의를 반드시 삽입한다.
- 웹사이트는 그래픽을 연결시키지 않은 상태로도 사용이 가능해야 하며, 이미지를 볼 수 없는 사람들의 비용과 이익간의 균형을 반드시 고려해야 한다.
- 대안 태그는 항상 포함되어야 하며, 이미지의 외관뿐만 아니라 기능을 설명해야 한다. 내용은 100 문자를 초과하지 않아야 한다.
- 중요한 로고가 처음으로 사용되는 곳에는(예를 들면 웹사이트 상에), 완전한 공식적인 설명("X 정보컨텐츠 팀 로고 : ....을 나타내는 로고" 등)을 제공하는 것이 권장된다. 이후 로고가 반복될 때는 ALT 텍스트 내에 "X 정보컨텐츠 팀 로고"로 명명할 수 있다.
- 때때로 중요 정보(예를 들면, 차트, 테이블 또는 다이어그램)를 나타내는 어떤 이미지의 내용에 대해 자세한 설명이 제공될 필요가 있다. 이 설명은 주요 웹 페이지 내에 텍스트로 포함되거나, IMG 요소의 LONGDESC 속성에 의해 링크된 웹 페이지에 위치시킬 수도 있다.
- IMG와 A에서 사용 실례

```
<img src="access.gif" alt="[Description]"
longdesc="imgdesc_a.html"><a href="w.htm" title="Description of
Accessbility">[D]</a>
```

- OBJECT에서의 사용 실례

```
<object data="accessbrdlogo.gif" type="image/gif"> The Access
Board's <a href="projected.html">projected budget</a> for Fiscal
2005 is ... </object>
```

- IMAGE MAP에서의 사용 실례

```
<OBJECT data="navigation.gif" type="image/gif" usemap="#mapnav">
<MAP name="map1"><P>Navigate the Access Board site.
[<A href="guidelines.html" shape="rect" coords="0,0,118,28">Access
Guide</A>]
[<A href="news.html" shape="rect" coords="118,0,184,28">Go</A>]
[<A href="search.html" shape="circle"
coords="184.200,60">Search</A>]
</MAP>
</OBJECT>
```

- TABLE에서 사용 실례

```
<TABLE border="1" summary="This table charts the number of web pages
analyzed by each agency head, what kind of media the pages contain,
and whether or not the page is part of the Executive Branch.">
<CAPTION>Web pages Analyzed by Agency Heads</CAPTION>
<TR>
<TH id="header1">Agency Head</TH>
. . .
```

```
</TR>
</TABLE>
```

5) 접근성 시험

웹사이트 접근성은 Website garage (http://www.websitegarage.com) 혹은 Bobby (http://www.cast.org/bobby) 등에서 테스트해 볼 수 있다. 좋은 방법은 텍스트 브라우저인 Lynx를 활용하여 웹사이트를 시범적으로 조사해 보는 방법도 있다.

참고 사이트

각 웹 표준 브라우저별 호환 여부

HTML 호환 차트

- http://nanobox.chipx86.com/browser_support_html.php

CSS 호환 차트

- http://nanobox.chipx86.com/browser_support_css.php
- http://www.quirksmode.org/css/contents.html

DOM 호환 차트

- http://nanobox.chipx86.com/browser_support_dom.php
- http://www.quirksmode.org/dom/contents.html

ECMAScript 호환 차트

- http://nanobox.chipx86.com/browser_support_ecmascript.php
- http://www.quirksmode.org/js/contents.html

국내 웹 표준 커뮤니티

웹 표준화 프로젝트

http://webstandard.or.kr

모질라 웹 표준 게시판

http://forums.mozilla.or.kr/viewforum.php?f=9

CSS 디자인 코리아

http://css.macple.com

실전 웹 표준 가이드

한국소프트웨어진흥원

(138-711) 서울특별시 송파구 가락본동 79-2 KIPA 빌딩
발행일: 2005. 12
발행처: 한국소프트웨어진흥원 공개SW지원센터
전 화: 02-2141-5000 FAX: 02-2141-5199
E-mail: webmaster@oss.or.kr URL: http://www.oss.or.kr