브라우저 호환성과 웹표준 기반 사이트 제작을 위한

웹표준 기반 홈페이지 구축 가이드

한국소프트웨어진흥원 공개 SW지원센터

목차

저자: 윤석찬 (다음커뮤니케이션 R&D 센터, 한글 모질라 프로젝트 운영자)

들면서

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 시점인 데다 더 이상 웹브라우저가 신기한 도구이지 않기 때문에, 다양한 인터넷 환경에서 어떠한 웹브라우저가 가장 최적의 구현을 제공하느냐가 관건이 되었다.

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 오픈소스 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 제작사들이 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. 이 가이드에서는 현대적인 웹브라우저들이 과거와는 달리 웹 표준을 준수하고 있기 때문에 이를 배우고 따르기만 하면 보다 저렴한 비용으로 홈페이지를 구축 및 유지 보수 할 수 있다는 것을 보여 주기 위해 작성되었다. 전 세계의 많은 웹사이트들이 웹 표준에 기반한 홈페이지 제작 방식을 도입하고 있으며 이를 통해 매우 효과적인 웹서비스 체계를 갖추기 시작하고 있다

1부 브라우저 호환성 유지 방안

1. 현대 웹사이트 제작의 방향

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 웹페이지 제작이 현재 웹 서비스 제공자들의 공통된 숙제가 되고 있다.

이것은 흔히 하위 버전 호환성 유지(Backward Compatibility)와 혼동하는 경우가 있다. 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 버전 호환성 유지는 예전에 사용되는 기능이나 태그를 표준 태그로 치환해 주는 것이다. 이를 통해 대부분의 웹 디자이너는 예전 지식에 따라 웹페이지를 코딩해 주어도 그대로 구현되는 것으로 생각하게 되는 것이다. 그러나, 버전 호환성 유지는 웹브라우저의 벤더에 따라 지원 가능 정도가 약해 질 뿐만 아니라 웹 표준 기술에 대한 지식 습득을 가로막는 장애가 된다.

2005	IE 6	IE 5	O 7/8	Ffox	Moz	NN 4	NN 7
March	64.0%	3.9%	1.8%	21.5%	3.7%	0.2%	1.0%
February	64.8%	4.2%	1.9%	20.4%	3.9%	0.2%	1.1%
January	65.3%	4.4%	2.1%	19.3%	4.0%	0.3%	1.1%
2004	**IE 6**	**IE 5**	**O 7**	**Moz**	**NN 3**	**NN 4**	**NN 7**
December	66.0%	4.8%	2.0%	21.3%	0.2%	0.3%	1.2%
November	68.5%	5.0%	2.2%	19.3%	0.2%	0.2%	1.2%
October	69.5%	5.7%	2.2%	17.5%	0.2%	0.2%	1.3%
September	69.6%	6.2%	2.3%	16.9%	0.2%	0.2%	1.3%
2003	**IE 6**	**IE 5**	**O 7**	**Moz**	**NN 3**	**NN 4**	**NN 7**
November	71.2%	13.7%	1.9%	7.2%	0.5%	0.5%	1.6%
September	69.7%	16.9%	1.8%	6.2%	0.6%	0.6%	1.5%
2002	**IE 6**	**IE 5**	**IE 4**	**AOL**	**NN 3**	**NN 4**	**NN 5+**
March	36.7%	49.4%	0.7%	3.0%	1.2%	4.1%	2.4%
January	30.1%	55.7%	1.0%	2.8%	1.3%	4.4%	2.2%

그림 1 W3school의 브라우저 통계 (http://www.w3schools.com/browsers/browsers_stats.asp) IE Internet Explorer , Ffox Firefox (identified as Mozilla before 2005), Moz Mozilla, O Opera, NN Netscape, AOL America Online (based on both Internet Explorer and Mozilla)

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.

즉, 웹 서비스의 모든 잠재 사용자들이 사이트에 접근할 수 있어야 한다는 것은 매우 중요한 요소라는 것이다. 표준 웹 기술의 범주에는 레이아웃 및 기술적 공통성을 추구하는 면이 있는가 하면, 일반적이지 않는 웹 사용자에 대한 지원이라는 포괄적인 의미도 함축하고 있다. 따라서, 웹 표준에 따라 브라우저와 관계없이 웹페이지 기능을 구현하려는 상호 호환성 확보는 독점 브라우저 시장에서 매우 중요한 문제라고 하겠다.

2. W3C 표준안

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 만들어진다. 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.

언뜻 보기에는 연구 기관으로만 이루어진 것 같으나 웹과 관련된 510여개의 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.

웹브라우저 중 모질라, 넷스케이프 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.

최신의 HTML 표준은 4.01이며, HTML을 XML과 결합한 XHTML이 권고안으로 나와 있다. HTML2/3와 달리 최신 HTML 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 비전문 설계자도 웹 페이지를 손쉽게 제작 관리 할 수 있게 해 준다.

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.

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, IE 6.0은 아직 하위 버전 호환성을 위해 MS DOM을 지원하고 있다. IE6.0 이전 브라우저를 제외하고는 거의 무든 브라우저가 표준 W3C DOM을 사용한다.

자바스크립트는 표준으로 제정된 것은 아니다. 또한, 모든 웹브라우저 사용자가 자바스크립트를 볼 수 있는 것은 아니다. 특정 방화벽은 자바스크립트가 통과하는 것을 허용하지 않는다. 그럼에도 자바스크립트는 DOM이 표준화 되면서 웹브라우저에 널리 쓰이고 있다. 주의할 점은 클라이언트측 스크립트는 여러 브라우저에서 폭 넓게 검사되어야 한다. 핵심 기능은 자바스크립트에 의해서만 제공되어서는 안 된다. 또 자바스크립트는 주석 코드를 사용하여 비호환성의 웹 브라우저로부터 숨겨져야 한다. 자바스크립트는 HTML 문서의 Head 내에 위치해야 제대로 동작한다 따라서 문서의 Body 내에 자바스크립트를 위치시키는 것은 피해야 한다.

웹 표준을 지킨다고 하는 것은 W3C의 표준안을 지킨다는 것을 말한다. W3C의 표준 활동을 지속적으로 살펴보고 표준 권고안을 공부할 필요가 있는 것이다. 이러한 표준 번역본은 W3C 한국 사무국 (http://www.w3c.or.kr)에서 찾을 수 있다. 또한, 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.

3. 정확한 문서 형식 사용

대부분의 웹페이지 들이 <html>로 시작하여 <head>와 <body> 태그를 사용하여 웹페이지를 표현한다. 그러나 웹페이지를 표현 하는 방식을 제대로 표현 하기 위해서는 표준은 아니더라도 준수하는 표준 규격이나 문자 인코딩들을 표현 하여 브라우저가 적절한 문서 형태를 표기하도록 할 필요가 있다. 이런 경우 사용하는 <!DOCTYPE> 태그 역시 제대로 알지 못하고 사용하는 경우가 많다. 이것은 HTML 파일 맨 처음에 공백 없이 적는 것으로 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/xh
```

```
tml1/DTD/xhtml1-frameset.dtd">
```

5) XHTML 1.1 표준 문서 형식

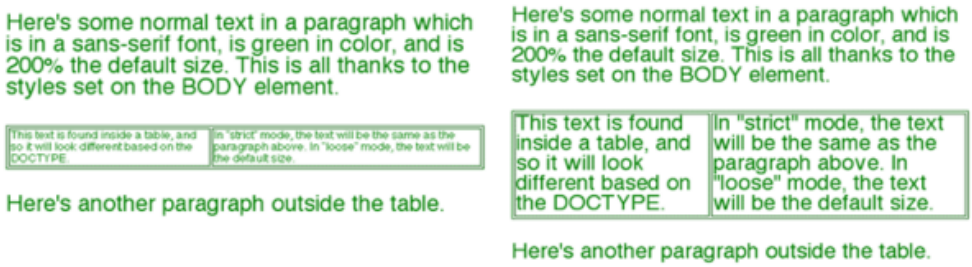
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD
/xhtml11.dtd">
```

표준 문서 형식(Doctype)을 기반한 웹페이지에 대한 정확힌 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="en-US">
<head>
<title> ... </title>
</head>
<body>
...
</body>
</html>
```

가장 흔히 사용되는 DOCTYPE 코드는 일반 형식과 엄격한 형식으로 나누어 지게 된다. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">은 HTML 4.01에서 예전에 있었거나 없어진 태그도 지원하며, <font>에 지정된 스타일도 제대로 표현하여 준다. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN">은 HTML 4.01을 엄격하게 적용한다. <font> 태그에 적용된 스타일 보다는 CSS파일에서 지정된 스타일을 지켜 표현 하도록 한다. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">은 HTML 4.01에서 Frames을 포함하는 웹 페이지에서 사용한다. 이러한 표현 방식에는 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html401/loose.dtd">처럼 W3C에서 제공하는 DTD파일을 포함하기도 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.



(a) Transitional 형식으로 규정한 경우　　(b) Strict 형식으로 규정한 경우

그림 2 DOCTYPE에 따른 HTML 표시 방법

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.

인터넷 익스플로러는 meta 태그를 통해 선언되어 있지 않더라도 문자코딩을 자동적으로 감별하여 표시하지

만 모질라는 그렇게 하지 않는다. 따라서 meta태그의 charset을 euc-kr 등으로 아래와 같이 지정해 주어야 한다. 만약 이것을 빠뜨리면 모질라에서 한글이 깨지는 상황이 생겨 마치 오류처럼 보일 수 있는 가장 첫번째 요소이다.

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

또한, form에서 post 후에 통상적으로 이전 문자셋 코드를 그대로 사용하는 데 모질라는 데이터를 POST로 송신한 후 페이지에 대해 자동 판별을 하지 않는다. 단, POST후에 표시되는 페이지에 문자 코드 선언이 있는 경우 (HTTP헤더 혹은HTML안에서의 meta) 모질라는 그 지정된 문자 코드로 페이지를 표시한다.

뿐만 아니라, 웹서버에서 브라우저로 데이터를 보낼 때에는 HTTP 헤더에 데이터 컨텐트 형식을 적어주게 돼 있다. 대부분의 아파치 웹서버는 text/plain이 기본으로 설정돼 있으며 MIME 타입에 설정돼 있지 않는 확장자로 데이터를 보내면 웹브라우저는 어떤 형식인지 인식할 수 없게 된다. 인터넷 익스플로러에는 MIME 스니핑이라는 기능이 내장돼 있어 헤더에 text/plain이더라도 html 구문이 포함돼 있으면 html 문서로 읽어주는 역할을 해 왔다. 이것은 그림 파일이나 기타 다른 애플리케이션 파일 포맷도 마찬가지다.

따라서 문서 형식에 따라 적절한 MIME 형식을 지정해 주는 것이 중요하다. 이것은 확장자를 통해 별도로 문서 형식을 정해야 한다. 그렇지 않고 PHP, JSP 같은 CGI 프로그램을 사용하는 경우, 문서 다운로드나 그림 표시를 하는 경우 웹프로그래머들이 문서 헤더에 정확하게 MIME 형식을 문자 인코딩 등을 지정해 주어야 한다. MIME 형식 지정이 제대로 안된 경우, 비 IE 브라우저에서는 HTML소스가 그대로 보인다든지 그림 파일 등이 제대로 표현되지 않는다.

또한, 윈도우 XP 서비스팩2에서는 이런 자동적인 MIME 스니핑은 더 이상 작동하지 않는다. 만약 그렇지 못한 파일이 IE에 전달되면 저장 기능만 작동되며 바로 실행은 이제 불가능하다. MIME 타입이 보내지면 이에 따라서만 데이터를 표시하게 된다이미 모질라나 오페라 같은 브라우저에서는 이렇게 작동됨으로써 정체 불명의 파일이 다운로드돼 실행되는 것을 막고 있다. MIME 타입을 알 수 없는 경우에는 스니핑이 그대로 동작하고 있기 하지만 웹개발자들에게 HTTP 헤더와 데이터 포맷을 정확히 쓰는 버릇을 들이도록 해야 한다.

5. 올바른 HTML 및 CSS 사용 방법

각 웹브라우저에 따라 지원하는 HTML 태그가 다를 수 있다. 대표적으로 <col>과 <colgroup>에서 span, width만 넷스케이프는 지원하나 인터넷 익스플로러가 지원하는 align, valign은 하지 않고 있다. 또한 <iframe>의 align도 넷스케이프는 지원하지 않는데 이는 align, valign 같은 속성들이 CSS로 합쳐지기 때문이다. 따라서 align 혹은 valign 같은 것은 쓰지 않고 CSS에서 설정하여 사용하는 것이 좋다. <legend>의 경우 인터넷 익스플로러만 지원하며 <table>의 hspace, vspace는 HTML3에 있는 기능이나 인터넷 익스플로러에서

문서 표시 형식	text/html	application/xhtml+xml	application/xml	text/xml
HTML4	필수	불가	불가	불가
XHTML1.0 (HTML호환)	가능	필수	가능	가능
XHTML1.0 (일반)	불가	필수	가능	가능

표 1 표준 방식에 따른 문서 형식

는 지원하지 않는다.

HTML 표준은 아니나 지원이 다른 경우를 살펴보면, <body> marginwidth, marginheight는 넷스케이프에서 지원하고 topmargin, leftmargin은 인터넷 익스플로러에서 각각 다르게 지원하므로 같이 설정해 주어야

바르지 않는 표현	추천 하는 표현
NS4 <layer> </layer>	HTML 4.0
NS4 <nolayer> </nolayer>	N/A
NS4 <ilayer> </ilayer>	HTML 4.0 <iframe> </iframe>
HTML3 <dir>, <menu>	HTML 4.0
NS4 , <layer> 에서 src=	HTML 4.0 <iframe> 의 src=
IE3/4/5 <marquee>	HTML 4.0 DOM1의 Javascript로 표현.
IE3/4/5 <bgsound>	HTML 3.2 넷스케이프4-에서는 <embed> 기타 웹 브라우저에서는 <object>
HTML3	CSS1 text-align:center
HTML3 <strike>	CSS1 text-decoration:line-through
HTML3 <u>	CSS1 text-decoration:underline
NS4 <blink> </blink>	CSS1 text-decoration: blink;

표 2 HTML태그의 올바른 사용법

한다. <hr>의 color와 <frame>의 framespacing, <table>의 bordercolor-light, bordercolordark 그리고 배경음악을 들려주는 <bgsound>, <script>의 id는 인터넷 익스플로러만 지원되는 것이므로 가급적 사용하지 않거나 대체되는 자바스크립트 기능을 이용하는 것이 팔요하다. 특히 <bgsound>의 경우 <object>를 사용하여 활용 가능하다.

넷스케이프와 인터넷 익스플로러의 브라우저 시장경쟁 당시에 각 사의 대표적인 비교 태그인 <blink>, <marquee>는 글자를 깜박이거나 흐르는 기능을 해주는 것들로 이들은 요즘 자바스크립트로 충분히 구현 가능하다. 인터넷 익스플로러만 지원하는 기능 중에는 <comment>, <xml>등의 태그가 있으며, <spacer>은 넷스케이프에서만 지원하는 등 차이가 있다. 이러한 웹브라우저 사이의 특이성은 테이블(table)에서 표시하는 방식에서 확연히 다른 결과를 보여준다. <그림2>는 테이블에 background 이미지 파일을 설정하였을 때 달리 나타나는 현상을 표현한 것이다

그림 3 Table에 background image를 삽입한 경우

브라우저들 간에 나타나는 이러한 차이점은 HTML 태그의 속성을 각기 다른 렌더링 엔진을 통해 표현하기 때문에 생기는 문제들로 디자인을 달리 보이게 하는 요소들로서 지적되고 있다. 이런 태그들은 아예 사용하지 않거나 align같은 스타일 속성에 의존하고 있는 것은 CSS를 사용하여 문제를 해결한다. 브라우저의 특이성은 당연히 존재하는 것이며 이러한 특징들을 알아두고 확인하여 최대한 표현하고자 하는 컨텐츠 형태로 출력하는 것이 중요하다.<표2>은 HTML과 CSS에서 필수적으로 바꾸거나 사용하지 말아야 하는 표현들을 정리하였다. 비 권장 요소들은 대체할 수 있는 제안들이 이미 나와 있는 경우이다.

<object>를 사용하여 동영상 및 미디어 포맷 데이터를 브라우저에 표현 할 때에는 인터넷 익스플로러가 표준 속성을 따르지 않기 때문에 다음과 같이 부득이 <embed>와 함께 표기하는 것이 좋다.

```
<OBJECT ID="MediaPlayer1" width=300 height=300 classid="CLSID:22D6F312-B0F6-11D0-94
AB-0080C74C7E95" type="application/x-oleobject">
<PARAM NAME="FileName" VALUE="mms://vodboard.hanafos.com/test.wmv">
<EMBED SRC=" mms://vodboard.hanafos.com/test.wmv" type="application/x-mplayer2" nam
e="MediaPlayer1" width=300 height=300>
</EMBED>
</OBJECT>
```

스타일을 지정할 때 특정 클래스를 만들어 별도로 사용한다는 생각을 버리고 기존의 HTML TAG에 스타일을 선언하여 잘 활용할 필요가 있다. 대부분의 웹디자이너들은 CSS를 적용할 때 새로운 class로 선언하고 사용하는 경우가 많다. 아래의 font 속성 클래스를 지정하는 것 같은 것이다.

```
<STYLE type="text/css">
<!--
.title { font-size:12pt, font-color: red }
h2 { font-size:12pt, font-color: red } -->
 -->
</STYLE>
<h1>Red Title</h1>
<font class="title">Red Title</font>
```

이러한 CSS 사용은 별로 바람직하지 못하다. 기존의 <h1>, <b>, <p> 등의 HTML Tag에 CSS를 적용하면 HTML코드도 간단하고, 웹브라우저 간 다르게 지정된 HTML 태그에 대한 표현도 통일될 수 있다.

폰트 크기나 사이즈 단위에서 %, em 등을 사용하지 않고 pixel만 사용하는 것이 좋다. line-height는 쓰지 않는 것이 좋으며, class명에 일반 키워드 같은 것은 혼돈되므로 쓰지 않는 것이 좋다. BODY에는 margin-left. margin-right만 설정하고 LI, DD, DT 등에는 스타일을 설정하지 않는 것이 좋다. 또한, <h1>, <h2> 같은 태그를 사용하여 문서의 구조의 역할을 부여할 필요가 있다. 단순히 글꼴이나 글자의 크기를 배치하기 위해 사용하는 것이 아니라 HTML 태그의 쓰임새에 따라 CSS를 설정해야 한다는 의미이다.

CSS 파일을 외부 파일로 지정하면 브라우저 캐쉬를 통해 속도가 향상되는데 이 파일을 link속성에서 읽어오는 경우, 아래와 같이 title을 통해 여러 대체 스타일 시트를 사용할 수 있다. title이 없는 스타일 시트가 표시의 기본 형식이 되고 title속성에 따라 또는 rel의 속성에 따라 대체 스타일 시트를 표시할 수 있다. 그러나, 인터넷 익스플로러의 경우 모든 스타일 시트를 다 읽어 오기 때문에 주위를 요한다. 모질라의 경우 각 스타일 시트를 통해 텍스트의 레이아웃을 변경할 수 있기 때문에, 기본모드, 텍스트모드, 인쇄모드 등으로 레이아웃을 간편하게 바꿀 수 있는 장점이 있다. 모질라의 방식이 표준에 부합하는 지는 논쟁의 여지가 있다.

```
<link rel="stylesheet" href="foo.css" type="text/css">
<link rel="stylesheet" href="hge.css" type="text/css" title="노약자용">
<link rel="alternate stylesheet" href="alt.css" type="text/css" title="인쇄용">
```

link를 통해 style파일을 불러오는 경우, 이미 불러온 HTML과 다른 파일이므로 문자코드가 선언되지 않았다면, 스타일 시트에 문자코드를 선언해 주어야 한다. 이는 font-family에서 한글 글꼴을 사용한다면 더더욱 필요한 것이다.

```
@charset "euc-kr"; /* charset */
```

CSS에 의한 Length값은 0의 경우를 제외하고 항상 단위를 적어주어야 한다. border-width나 width, height 등에 0 이외의 값에 단위가 없으면 표준 웹브라우저에서는 부정한 구문이 되어 속성 설정이 무효화 된다. 그러나, 인터넷 익스플로러에서는 이러한 값을 px로서 보완하게 되어 있으므로 가능하면 px로서 단위를 꼭 사용하는 것이 좋다.

색상을 표시할 때도 #3399ff과 같에 번호 앞에 #을 꼭 붙이도록 한다. 이를 붙이지 않아도 색상을 표현하는 브라우저가 있으나 이것은 바람직한 렌더링 결과가 아니며, 이러한 차이로 다른 브라우저 간에 다른 색이 표현되기도 한다.

또한, 스타일을 선언할 때 어떤 것은 순서에 유의해야 하는 것이 있다. :hover의 경우, 마우스를 특정 링크에 올렸을 때 나타나는 색상을 지정할 때 사용하는 데, 아래의 경우는 인터넷 익스플로러와 모질라가 다른 결과를 나타낸다.

```
a:hover{ color:red; } a:link{ color:black;} a:visited{ color:green;}
```

위의 CSS를 사용한 경우, 마우스를 올렸을 때 visited 속성에는 붉은색이 표시되지 않고 link에만 붉은색이 표시된다. 이것은 CSS 단계화 규칙이라는 것이 있어 어느 조건에 대해서 값을 여러 차례 지정하면 마지막에 지정된 것이 유효하게 되기 때문이다. 반면, hover를 제일 나중에 두면 제대로 동작하게 된다. 따라서, link, visited, hover, active, focus의 순서대로 CSS를 정해야 한다.

우리가 문장의 스타일을 제어할 때 사용하는 text-decoration의 경우에도 브라우저에 따라 많은 차이가 난다. CSS속성에도 트리 구조가 있어 상위에서 실행된 내용이 하위에서 취소되거나 할 수 없게 되어 있다. text-decoration은 표준 사양에 비하면 오페라와 인터넷 익스플로러가 버그가 있는 것으로 보이므로, 이것을 사용할 때는 최대한 여러 브라우저를 통해 살펴 볼 필요 가 있다.

윈도우와 맥 및 매킨토시 등 운영 체제에 따라 사용되는 기본 글꼴은 모두 다르다, 따라서 같은 내용도 글꼴에 따라 매우 다른 모양을 나타낼 수 있다. font-family 속성을 정할 때 고려해야 할 점은 각 OS플랫폼에서 유사한 글꼴을 집합하여 보여 줌으로서 최대한 통일 성을 유지 하게 하는 것이다.

한국어 글꼴 이름은 '굴림, gulim, 돋음, dotum'처럼 로마자와 한글로 모두 표시한다. 한국어 설정을 쓰는 브라우저가 많을 것이므로 한글로 표기한 글꼴 이름을 더 앞에 둔다. 아래 A.는 모든 운영 체제를 지원하는 명조체 글꼴을, B는 고딕체 글꼴을 표시한 것이다.

```
1) 명조체 글꼴: font-family: 바탕, Batang, "Baekmuk Batang", UnBatang, AppleMyeongjo, "Tim
es New Roman", "Bitstream Vera Serif", Times, serif;

2) 고딕체 글꼴: font-family: 돋음, Dotum, Arial, "Helvetica", "Bitstream Vera Sans", "UnDot
um", "Baekmuk Dotum", AppleGothic, sans-serif;
```

CSS1과 CSS2는 웹브라우저에 따라 구현되어 있거나, 또는 덜 구현되어 있거나 지원하지 않는 경우가 있다. 또한, 구현되어 있더라도 문법에 따라 버그로서 존재하는 것이 있기 때문에 그 구현 정도를 확인해 볼 필요가 있다.

5. 올바른 객체 모델 및 자바스크립트 사용 방법

문서 객체 모델(DOM)이라는 방식을 통해 HTML내에 있는 거의 모든 요소를 스크립트 처리 가능한 객체로서 활용하여 만들어 자바 스크립트에서 활용하거나 CSS에서 스타일을 지정해 줄 수도 있다. 그런데 이러한 객체를 읽어 올 때 브라우저에서는 독자적인 객체 모델을 가지고 있다. 예를 들어 넷스케이프4 에서는 객체 모델 안의 특정의 요소에 액세스 하는데 document.tags 가 사용되며 인터넷 익스플로러 문서 객체 모델(MS DOM)에서는 document.all[]이 사용된다. 이러한 차이로 인해 상당수가 W3C의 표준 DOM 의 사양에 포함되지 않아 모질라나 오페라와 같은 표준에 준거한 브라우저에서는 자바 스크립트 에러의 요인이 된다.

W3C 표준 DOM에서는 getElementByID라는 객체 모델을 사용하고 있으며, 현재 대부분의 브라우저들이

이를 지원하고 있다. 구형 브라우저를 지원하기 위해서는 다음과 같은 객체 판별 스크립트를 사용하면 된다.

```
function getObject(objectId) {
// checkW3C DOM, then MSIE 4, then NN 4.
if(document.getElementById && document.getElementById(objectId)) {
    return document.getElementById(objectId); // 대부분의 브라우저
    }
else if (document.all && document.all(objectId)) {
    return document.all(objectId); // IE4와 5.0
    }
else if (document.layers && document.layers[objectId]) {
    return document.layers[objectId];  // Netscape 4.x
} else {
    return false;
    }
}
```

사용 방법은 원하는 객체에 대해서 getObject('content').style.visibility="hidden"; 와 같은 방법으로 표현하면 된다. 이런 방법을 활용한 다양한 브라우저 지원 방식은 http://www.cross-browser.com/toys/ 에서 찾을 수 있다.

자바스크립트를 사용할 때는 script 태그에 LANGUAGE="JavaScript"를 선언해 주거나 type="text/javascript"를 사용해 준다. JScript, VBscript 등은 인터넷 익스플로러에서만 사용하므로 사용하지 않도록 한다. 또한, 내용은 꼭 코멘트를 사용하여 텍스트 브라우저에서도 잘 표현 되도록 해야 한다. 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.

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

바르지 않는 표현	추천 하는 표현
Netscape4 document.layers[], document.tags[], document.ids[], document.classes[], document.elementName IE 4/5 document.all	DOM1 document.getElementById()
IE 4/5 element.visibility = value;	DOM2 element.style.visibility = value;
Netscape4 handleEvent()	DOM2 dispatchEvent()
Netscape4 element.left, element.top IE 4/5 element.style.pixelLeft, element.style.pixelTop	DOM2 parseInt (element.style.left) DOM2 parseInt (element.style.top)
Netscape4 element.moveTo(x, y); IE4/5 element.style.pixelLeft = x; element.style.pixelTop = y;	DOM2 element.style.left = value + "px"; DOM2 element.style.top = value + "px";

표 3 정확한 DOM 표현 방식

제공하는 것이 좋다. 예를 들면, 스크립트 기능이 꺼져 있거나 지원되지 않을 경우에도 스크립트를 활성화하는 링크가 작동하도록 해야 한다. (예를 들어, 링크의 목적지 로 "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.

어떤 사용자 form에서 action을 받은 후 나온 결과에 자바스크립트 만을 제공해 자동 전환하는 결과는 될 수 있으면 사용하지 않는다. <script> href.location=□test.html□; </script> 같은 내용만 담는 것은 권장하지 않으며 만약 한다면, window.href.location 이라고 정확하게 표현하거나, <meta> 태그의 refresh를 사용하거나 하는 것이 옳다.

아래 표는 웹페이지의 다양한 HTML 요소에 정의된 스타일 규정을 변경할 때, 표준에 의거한 메소드를 알려주고 있다.

Form 객체를 사용할 때 <form id="testfrm"><input type=text name="userid"></form>에서 document.forms("userid")로 표현하는 실수를 할 때가 있다. document.form은 메소드가 아니라 배열이기 때문에 document.forms[0]로 표현하는 것이 정확하다. 또는 document.form.userid 로 표현한다.

6. 웹페이지 디버깅 도구 사용

웹 표준과 호환성 확보 방안을 충분히 숙지하고 있어도 오류가 나는 것이 웹페이지이다. 가이드라인을 잘 익히는 것도 중요하지만, 결국 웹 개발자가 최종 작업을 마치고 여러 웹브라우저에서 기능을 구현하여 동작 여부를 체크하는 것이 필요하다. http://browsers.evolt.org 링크를 따라가면, 각 웹브라우저의 예전 버전까지 제공해 준다. 호환을 브라우저 버전을 정한 후 테스트를 할 필요가 있다.

웹페이지의 자바 스크립트 오류를 알아 내기 위해서는 모질라나 파이어폭스에 있는 자바스크립트 콘솔을 이용하는 방법이 있다. 이 콘솔을 이용하면, 표준안에 근접한 방법으로 웹페이지를 디버깅 할 수 있는 장점이 있다. 파이어버드는 가벼운 웹브라우저 이므로, 파이어버드와 인터넷 익스플로러 그리고 오페라 정도로도 웹페이지 디버깅을 할 수 있다.

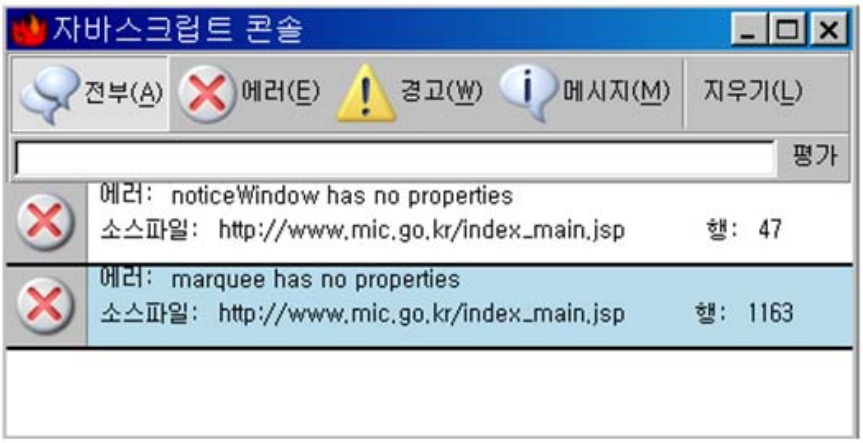


그림 4 파이어폭스의 자바스크립트 콘솔

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 인터넷 익스플로러에서도 MS 스크립트 디버거라는 프로그램이 있다. 기본적으로 자바스

크립트에 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. MS에서 제공하느 최신의 스크립트 디버거는 아래에서 다운로드 받을 수 있다. (http://www.microsoft.com/downloads/details.aspx?FamilyID=2f465be0-94fd-4569-b3c4-dffdf19ccd99) 모질라에는 벤크맨(Venkman)으로 불리는 JavaScript 디버거가 내장 되어, 스크립트 개발자들에게 이용되고 있다. 이것은 화면 표시와 콘솔 양쪽에서 조작할 수 있는 디버거이다. 웹페이지에 대한 또 하나의 디버깅 방법은 W3C의 HTML 유효성 검사 기능 (http://validator.w3.org/) 을 사용하는 것이다. 이 기능을 사용하면 웹페이지가 얼마나 표준에 맞는지 확인해 볼 수 있다.

2부 웹표준 기반 페이지 제작 방법

우리가 사용하는 HTML의 마크업은 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 표시하라□는 □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.

1. 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.

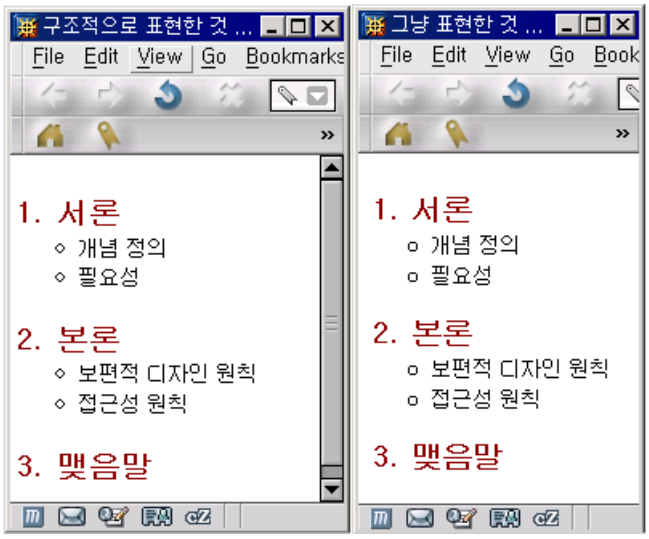


그림 5 다른 레이아웃 코드로 작성된 화면

그림5는 두 개의 브라우저에 표시된 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 그림5에서 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">ㅇ 개념 정의</font><br>

    <font size="2" color="black">ㅇ 필요성</font><br>
<font size="4" color="darkred"><b>2. 본론</b></font><br>
    <font size="2" color="black">ㅇ 보편적 디자인 원칙</font><br>

    <font size="2" color="black">ㅇ 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 그림5의 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 것입니다. 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 호환성(Backward Compatibility) 뿐만 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.

구조와 표현이 CSS를 이용해 아주 잘 분리된 사례는 http://www.csszengarden.com에 있다. 여기에 제시되는 페이지들은 모두 동일한 구조의 HTML 소스를 가지고 있으나, 별도로 분리된 CSS만을 바꿈으로써 완전히 다른 모양으로 탈바꿈하고 있다. 이 장에서는 XHTML과 CSS를 이용하여 구조적 마크업과 표현의 분리 그리고 이를 통한 사용성 및 접근성 높은 웹페이지 구축 방법을 알아본다.

2. XHTML를 사용해야 하는 이유

W3C는 HTML 4 스펙 자체와 그것을 정리한 문서에 포함된 일부 오류를 수정 하여 HTML 4.01을 내놓기는 했으나, 여전히 HTML4가 HTML의 마지막 버전이라는 사실은 달라지지 않았다고 할 수 있다. 즉 HTML4.01을 재구성한 메이저 업그레이드 버전으로 XHTML 1.0을 내놓은 것이다. XHTML은 eXtensible HyperText Markup Language의 약자이다. HTML을 대체하기 위한 목적으로 만들어졌고, HTML 4.01 규약에 준한다. 쉽게 말하자면 XHTML은 HTML의 XML 버전이며 일반 HTML에 비해 다음과 같은 점에서 사용이 권장된다.

호환성 및 확장 가능성

XHTML은 단순히 HTML4를 업그레이드한 것이 아니라 XML 애플리케이션을 가장 폭 넓게 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 신경쓰지 않는다. 이것은 HTML의 재활용과 생산성의 문제이며 장차 나오는 비용의 문제이다. 일반적인 관습대로 작성된 HTML 문서는 내용과 디자인, 문서구조가 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 디자인 되어 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.

3. XHTML의 주요 특징

모든 요소들은 정확힌 형식(well-formed)로 완벽하게 사용한다

일반 HTML문서라면 테스트 처럼 쓰더라도 아무 문제가 없으나, XHTML 문서는 반드시 테스트처럼 중첩구조가 완벽 해야 한다. HTML에서는 가끔 닫는 태그를 빼먹어도 별 문제없이 동작하지만, XHTML에서는 </xxx>처럼 닫는 태그들을 반드시 사용해야 한다. 특히 태그를 사용할 때 를 닫아야 한다.

또한, HTML에서는 마침 태그를 생략할 수 있는 경우가 많았다. 브라우저가 페이지를 렌더링 하는 속도를 향상시키기 위해 마침 태그를 적어주면 좋다는 식이 아니라 유효한 문서를 작성하기 위해 꼭 붙여서 마크업 해야 하는 것이다. HTML4에서도 CSS를 적용할 때 마침 태그를 붙여줄 필요성이 있다. Empty Element라고 불리우는 단독 태그들 - br, hr, img 등은
, <hr />, 처럼 써야한다. /앞의 공백은 붙이는 것을 권장한다.

모든 태그와 속성들은 소문자를 사용한다

HTML에서는 대소문자를 특별히 구분하지 않으므로 대문자로 태그를 처리하는 경우가 많았는데, XML의 특성을 받아들이는 XHTML에 맞추어 소문자로 적는 습관을 들이는 것이 좋다. <DIV NAME="xxx"> 처럼 쓰면 안되고 반드시 처럼 써야한다. 물론 대문자로 쓰더라도 해석은 되다 어플리케이션 호환성을 위해 반드시 소문자로 쓴다.

속성값을 줄 때는 겹따옴표(")를 적는다

HTML에서는 문자열(string)이나 숫자를 적어줄 때 따옴표를 생략해도 브라우저가 디스플레이하는데 무리가 없었으므로 따옴표로 묶어주는 것을 생략하는 경우가 많았는데, XHTML에서는 항상 적어줄 필요가 있다. 즉, <table width=100>은 틀린 XHTML 문법이며, <table width="100"> 처럼 써야만 한다.

단축형 속성값은 사용할 수 없다

흔히 HTML Form에서 checked, disabled 같은 단축형 속성값을 많이 사용했으나 요소를 함께 지정해 주어야 한다. 즉 <option selected> 대신 <option selected="selected">처럼 써야만 한다.

name 속성대신 id 속성을 사용한다

<input type="password" name="password"> 대신 <input type="password" id="password" />처럼 사용한다. name과 id의 가장 큰 차이점은 하나의 문서 내에서 동일한 name값을 가지는 요소들이 있어도 상관없지만, id값은 같은 문서에서 동일한 id값을 가질 수 없다.

자바스크립트에서 문서 내의 객체에 접근할 때, 이런 요소의 name을 바로 객체로 사용하는데 이것은 잘못된 습관이다. password.value나 document.form.password.value처럼 쓰는 것은 옳지 않으며, getElementsByName이나 getElementById 같은 메쏘드를 써서 해당 오브젝트를 반환 받아 써야한다. 즉, getElementById('password').value 처럼 쓰는 것이 옳다. 물론 옛날 브라우저와의 호환을 위해 id와 name을 동시에 쓰거나 브라우저별 객체 판별법 스크립트를 이용하는 것도 권장한다.

lang 속성의 사용 한다

만약 문서내에 특정 언어로 쓰여진 부분이 있다면, lang 속성을 이용해 언어를 구분해줍니다. This is written by in Korean(한국어)처럼 한국어 부분에 한국어로 된 글자라는 부분을 인식 시킨다.

<!DOCTYPE>을 정확히 사용한다

실제로 이 <!DOCTYPE>은 XHTML 문법에는 포함되지 않지만 해석기나 브라우저로 하여금 이후 이어지는 문서가 XHTML이며 어떤 DTD(Document Type Definition)에 의해 해석되어야하는 지를 지정해주는데 이용된다. 통상 Strict, Transitional, Frameset의 세 종류가 있으며, Strict는 엄격한 XHTML문법을 따르고 CSS와 같이 이용될 때 쓰이며, Transitional은 틀린 XHTML 문법이 있어도 HTML문법에 준해서 오류를 허용할 때 쓰인다. Frameset은 말그대로 HTML의 frame을 이용할 때 사용되는데 각각의 경우에 대해 아래의 내용을 문서의 가장 첫줄에 공백없이 적어야 한다.

다음은 이러한 XHTML 규칙에 따라 생성된 XHTML 코드의 예이다.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>simple document</title>
</head>
<body>
<p>a simple paragraph</p>
</body>
</html>
```

XHTML 혹은 HTML Strict DTD에서는 사용할 수 없는 태그들이 있다. <applet> 대신 애플릿을 이용하기 위해서는 <object>를 사용한다. <applet>태그는 HTML4.01 이상에서 더 이상 사용하지 않는다. <frame>, <frameset>, <noframes>은 Frameset DTD로 선언하지 않으면 사용할 수 없다. 따라서 Frame 태그를 포함하고 있는 페이지만 Frameset DTD를 선언하여 사용하면 된다.

, <font>, <basefont>, <s>, <strike>, <u>들은 모두 HTML에서 사용되던 '디자인 요소'들이므로 사용할 수 없다. 이 부분은 모두 CSS를 통해 처리해야 한다. <iframe>, <dir>, <menu> 등은 XHTML 문서에서는 구조화된 문서를 만드는데 해가 되므로 사용하지 않는다.

요소 외에 속성을 살펴 보면, class, id, style, title은 대부분의 태그에서 사용되는 핵심 속성이다. class는 클래스 규칙이나 스타일 규칙을 적용하는 속성이며, id속성은 해당 엘리먼트를 다른 엘리먼트와 구분 짓게 하고, style은 행 단위(Inline) 스타일을 적용할 때 사용한다. title은 해당 요소에 대한 설명으로 <image>의 alt처럼 보통 툴팁 도움말을 띄울 때 사용한다.

```
<div id="greeting" class="text" style="border:1px solid #F00;margin:5px" title="Hel
lo, World"> Hello World</div>
```

이 밖에 텍스트 출력방향을 지정해주는 dir속성과 사용되는 문자코드를 지정해주는 lang속성이 있으며, 엘리먼트에 키보드 단축키를 지정해주는 accesskey 속성과 탭 인덱스 순서를 지정해주는 tabindex 속성이 올 수 있습니다.

지금 까지 계속 설명해온 구조적 마크업을 위한 몇 가지 태그의 올바른 쓰임새를 알아보도록 하자. 기본이 되는 것은 □태그의 쓰임에 따라 사용하라□는 단순한 원칙이다. 문단을 나타낼 때는 , 긴 인용문을 표현할 때는 <blockqoute>, 단어나 문장을 강조하고 싶을때는 ,을 사용한다.

은 줄바꿈이 쉬워 가장 많이 사용되는 태그입니다. 그러나 이 태그 사용은 가급적이면 피하는 것이 좋다. 문장이 끝날 때마다
을 쓰는것은 효율적이지 못하다. 웹 문서가 지저분해지고 유지관리하기가 불편하기 때문이다. 흔히 문장의 줄 간격을 넓히기 위해서 연속적으로 사용하는 경우가 있는데 이 때는 줄 간격(line-height)속성을 이용해서 조정해 준다. 문장이 끝나거나 줄 길이를 줄이기 위해
태그를 쓰는경우가 아주 빈번히 있는데 그때는 CSS의 넓이(width)속성이나 패딩(padding)을 이용한다. 문단을 나눌때는
 대신 태그를 사용한다.
태그를 겹쳐 사용하지 말고 문단을 문단의 내용… 로 감싸서 사용한다.

짧은 문장이나 항목을 나타내기 위해
 태그를 사용 하지 말고 <ul>,
,
태그를 이용하여 나타낸다. , <li>, 는 들여 쓰기(indent)용 태그라고 생각하지만 항목성 자료를 나타낼 때 주로 사용하는 태그이다. CSS을 함께 사용하여 이 태그를 사용하면 간단한 메뉴도 구성할 수 있으며, 간단한 뉴스 목록을 표현 할 수 있다.

<blockquote>는 긴 인용문을 나타내기 위해 쓰이는데 기본적으로 들여 쓰기(indent)가 된다. 들여 쓰기 위해서 이 태그를 사용하지 않아야 한다. 문단 단위(block-level)이므로 태그와 같이 쓰이면 적합성검사를 통과하지 못하므로 유의 해야 한다. 이 태그를 쓸 때 인용된 곳을 링크하는 cite 속성과 인용 원문의 제목을 기입해 주는 title 속성을 사용할 수 있다. 이와 달리 <cite>라는 태그는 짧은 인용문을 나타낼 때 사용하는데 <blockquote>와 달리 행 단위(inline) 요소이다.

```
<blockqoute cite="http://user.oss.or.kr/standardguide" title="웹 표준 가이드"> 여기 있
는 웹 표준 가이드를 이용하시면 표준에 입각한 웹페이지 제작이 가능합니다. </blockquote>
```

기울임꼴(italic)이나 굵은꼴(bold)을 사용하기 위해 ,태그를 사용하지 않는다. 둘 다 강조의 뜻을 나타냅니다. 은 보다 더 강조 할 때 쓴다. 이들 태그들이 사용된 곳에서는 검색 엔진이 중요한 키워드로 인식하여 검색 결과를 더 높일 수 있다.

4. CSS 사용 방법

구조적 마크업에 대해 이해를 했다면 이제 표현과 레이아웃을 신경 쓸 단계이다. 하나의 HTML문서를 이루는 성분으로는 구조, 내용, 디자인이 있다. 구조란 이 문서의 형식, 구성요소들을 말한다. 상단 영역이 있고, 메뉴가 있고, 네비게이션 영역이 있고, 로그인 박스가 있고, 컨텐츠가 있고, 하단 영역이 있는 문서의 구조를 가리킨다. 대부분의 웹사이트는 이러한 영역의 조합으로 이루어져 있다.

내용이란 이 구조를 통해 실제로 전달하고자 하는 텍스트나 그림 자료라고 생각하면 된다. 디자인이란 이러한 구조와 내용을 가진 문서가 실제로 브라우저 등을 통해 어떻게 보여지느냐에 관한 것입니다. 예를 들어, 어떤 문서에 박스기사를 삽입하는 것은 문서의 구조를 변경시키는 것이며, 박스 기사에 디자인에 관한 글이 들어 가야 한다면 그것은 문서 내용에 관한 영역이 된다. 이 박스기사가 브라우저 출력시에 어느 위치에 어떤 식으로 보일 것인가를 결정하는 것이 바로 디자인이다.

웹프로그램이 내용을 다루고, XHTML이 구조를 다룬다면 CSS는 어떻게 표현될 것인가 하는 부분이 전문 영역이다. CSS에서 Style은 글자의 크기, 색상 등의 표현적 요소를 의미하고 Cascading은 폭포가 계단을 내려 오는 것처럼 여러 Style Sheets들이 단계적으로 중첩되어 적용됨을 뜻한다. 즉, 우리가 기본 스타일 시트를 정하고 HTML 파일 내에 또는 줄 특성 내에 다른 스타일을 사용하면 기본 스타일 파일이 기초로 적용되고 다른 스타일도 함께 적용된다는 것을 의미한다.

Style은 크게 4가지 층위로 Cascading된다.

1. 브라우저 기본 (브라우저에서 기본적으로 사용되는 스타일이다.)
2. External Style Sheet (<head>안에 <link> 태그로 외부에서 파일로 링크 된다.)
3. Internal Style Sheet(<style>이라는 태그를 써서 문서 안에서 정의된다.)
4. Inline Style (각 태그에 style=""이라는 속성을 이용해 정의된다.)

따라서 <table style="border:none">일 경우 Inline Style인 border:none이 브라우저에서 디폴트값으로 정해진 것 border:2px에 우선해서 적용되는 것이다. 실제로 문서에 적용될 때는 위의 네 가지 층위의 Style이 모두 합쳐져서 한개의 "Virtual Style"로 적용되어 사용된다. 그러므로 Inline Style로 border:none이 적용되었지만 다른 부분은 다른 층위에서 적용된 스타일을 따르게 된다.

구조적 마크업에서 3,4번이 HTML문서 내에 존재하는 것은 권장하지 않는다. CSS요소들은 모두 외부 css 파일에 위치 시켜 관리하는 것이 바람직하다. CSS의 문법은 selector, property, value의 세가지 요소로 구성된다.

```
selector {
property: value;
}
```

selector는 CSS의 스타일을 지정하는 이름이므로 selector는 일반적으로 HTML태그, class 이름, id 이름 등이 올 수 있다.

```
p { /* HTML 태그의 예 */
text-align:left;
}
.paragraph { /* class속성으로 쓸 경우의 예. 앞에 dot(.)가 붙음. */
text-align:right;
}
#main_contents { /* id속성으로 쓸 경우의 예. 앞에 #이 붙음. */
text-align:center;
```

```
}
```

실제 HTML에서 사용할 때는 다음과 같이 사용한다.

```
<p>
Test
</p>
<p class="paragraph">
Test2
</p>
<p id="main_contents">
Test3
</p>
```

원래 class selector는 앞에 적용되는 HTML 태그를 붙여 표현 하는 게 바르나 생략해도 된다. 위의 .paragraph는 p.paragraph에서 p태그가 생략된 예입니다. 왜 생략을 하느냐 하면 이 스타일이 반드시 <p>에서만 쓰이지 않을 수도 있기 때문이다. 그러므로 처럼 써도 통용되도록 하기 위해 일반적으로 HTML 태그는 생략하여 정의한다. 반대로 HTML태그를 붙여서 정의한다면 각각의 HTML 태그별로 같은 class 이름이지만 다른 형태의 스타일을 지정할 수 있습니다. id selector도 역시 앞에 HTML태그를 붙여야 하지만 생략 가능하다. class와의 차이라면, id로 정의된 스타일은 오직 한 요소에서만 사용될 수 있다.

구조적 마크업에서 class를 별도의 이름을 붙이는 것은 최대한 피하고 정해진 HTML 요소를 최대한 사용한 후 여기에 스타일을 추가하는 것을 선호한다.

5. CSS 레이아웃 vs Table 레이아웃

아마도 전체 웹사이트의 90% 이상이 화면 레이아웃에 테이블을 사용하고 있을 것이다. 레이아웃에 table element 를 사용하는 이유는 아마도 쉽기 때문일 것이다. 쉽다는 것 보다는 "익숙해서" 가 더 큰 이유일지도 모른다. 대다수의 사람들은 처음부터 아주 당연하게 테이블을 사용해 왔고 그렇게 되어 있는 사이트를 더 많이 봤을 것이다. 속도나 성능을 중요시 하는 포털 사이트 조차도 테이블을 사용하고 있으니까. 테이블을 사용해서 레이아웃을 잡으면서 테이블을 사용하면 불가능한 레이아웃이 없다는 것을 느낄 지도 모르나 CSS 를 사용해도 그것이 가능하고 그것이 더 의미에 맞고 효율적이다.

테이블은 레이아웃을 만드는데 사용하는 element 가 아니라 표를 나타내는 element 이다. 따라서 레이아웃에 이것을 사용하면 상대적으로 CSS 레이아웃 에 비해서 코드가 길어진다. 또한 테이블을 브라우져에서 표현을 할때에는 상당히 복잡한 width 와 height 계산 알고리즘을 거치게 된다. 따라서 화면이 랜더링 되는데 시간이 많이 걸리게 된다.

의미에 맞지 않다는 것은 어떻게 보면 별로 중요하지 않게 느껴질 수도 있다. 하지만 웹표준에서 이것은 굉장히 중요한 것이고 의미에 맞는 markup 을 사용한다는 것은 그 문서가 오랜 시간이 지나도 생명력을 가질 수 있다는 의미이다. 코드의 길이가 길어진다는 것은 불필요한 tr 이나 td 가 많이 들어간다는 것이다 CSS 를 사용하게 되면 의미 없느 row 나 column 을 표기해 주지 않아도 되기 때문에 코드의 길이가 줄어들게 된다. 코드의 길이가 줄어들면 구축이나 운영에 들어가는 비용을 절감할 수 있고, 네트웍을 통해서 전송되어야 하는 양이 줄어들기 때문에 보다 적은 서버부하, 빠른 페이지 전송을 할 수 있다.

랜더링 시간은 컴퓨터가 고사양이 되면 의미가 약해 질 수는 있지만 이것도 익숙함의 문제이다. 이제까지는 대부분의 사이트가 table 이어서 느리다는 것을 느끼지 못했겠지만 CSS 레이아웃을 사용하면 몸으로 체험할 수 있을 정도로 화면 랜더링 속도가 개선 된다. 테이블을 사용 한다고 하더래도 랜더링 속도를 계산 할

수는 있는데 바로 width 나 height 계산 알고리즘을 간소화 하는 것이다. 이것은 css 의 table-layout 속성을 이용하는 것이고 이것만 잘 사용해도 랜더링 속도는 비약적으로 향상된다.

그럼 CSS레이아웃을 사용하려면 어떻게 해야 하는가? 이것 역시 단점이 있다. table 을 이용해서 layout 을 만드는 것을 배운 만큼 CSS layout 만드는 것을 공부해야 한다. 또한, 전세계적으로 가장 많이 사용되는 IE6 가 아직 CSS 를 완전히 지원하지 않는다.그래서 어떠한 layout 은 table 을 이용하는 것이 더 제작이 쉽다.

이미 웹사이트를 만들 능력이 있는 사람이 새로운 기술을 위해서 교육을 받아야 한다는 것이 어떻게 보면 손실일 수도 있다. 이미 잘하고 있는데 방법을 바꾸는 것이라고 생각이 들 수도 있다. "보다 좋은 방법을 위해서 투자를 할 것이냐", 아니면 "다른 것에 투자를 할 것이냐" 의 문제이다. 개인 이나 회사도 모두 마찬가지라고 본다. 회사에서 CSS 레이아웃을 사용하기로 하면 그에 따른 교육이 필요할 것이다.

CSS 레이아웃을 사용하기 위해서는 CSS 가 제대로 지원이 되어야 한다. 너무나 당연한 말이다. 초기 브라우져들도 테이블을 나타내는 방법들이 달라서 이 부분이 이슈였던 적이 있다. 지금은 거의 테이블을 비슷한 수준으로 지원하고 있어서 대다수의 사람들이 느끼지 못하는 것일 뿐이다. 웹 표준이나 CSS 레이아웃은 브라우져 이슈와 밀접한 관계가 있다. CSS 가 지원이 안되면 CSS 레이아웃을 사용 못하고 그러면 어쩔 수 없이 기존의 테이블 레이아웃을 사용해야 한다. 이 때문에 많은 수의 웹표준에 관심을 가지고 있는 사람들이 이를 연구하고 문서들을 만들어 내고 있다. 이제 CSS를 이용하여 간단한 레이아웃을 가진 웹페이지를 만들어 보도록 하자. 우선 상단 영역, 메뉴영역, 컨텐츠 영역, 하단 영역으로 나누어진 페이지를 XHTML을 기반으로 아래와 같이 작성할 수 있다..

```
<!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=euc-kr" />
<title>뉴스 예제</title>
<link rel="stylesheet" href="layout.css" type="text/css">
</head>
<body>
<div id="container">
    <div id="header">
        <h1>뉴스 헤더</h1>
    </div>
    <div id="menu">
        <ul>
            <li>home</li>
            <li>뉴스</li>
            <li>방명록</li>
            <li>링크</li>
        </ul>
    </div>
    <div id="contents">
        <h3>박지성, 아인트 호벤 계약 연장</h3>
            <p>네덜란드 프로축구에서 활약하는 박지성(24.에인트호벤)이 소속팀에 3년 더 남을 전망이다.</p>
            <p>PSV 에인트호벤은 내년 시즌을 끝으로 계약이 만료되는 박지성과 3년 재계약을 맺기로 원칙적으로 합의했다고 현지 신문 알게메네 다흐 블라드가 12일(한국시간) 보도했다.</p>
    </div>
    <div id="footer">
        2005 news.com All rights reserved.
    </div>
</div>
```

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

위의 웹페이지는 구조적 마크업으로만 웹페이지를 작성한 것이다. 이 페이지를 CSS 없이 나타낸 것이 다음과 같다.

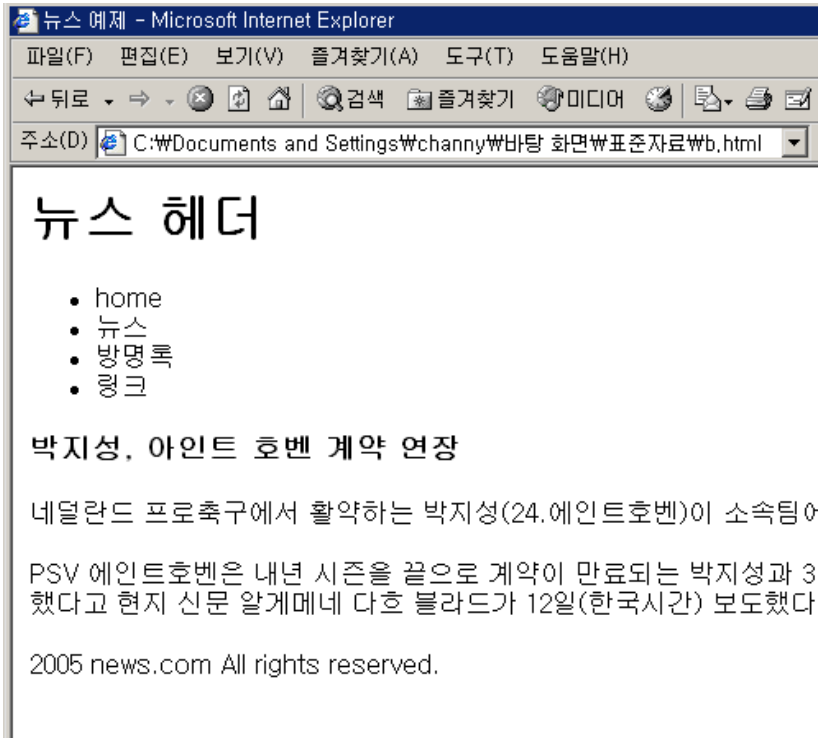


그림 6 CSS를 배제한 구조적 마크업

이제 이 웹페이지의 배치 및 스타일 지정은 CSS를 통해 가능하다.

```
#head {
    postion: relative;
    height: 50px;
    background: #fcc;
}
#menu {
    position: absolute;
    width: 180px;
    top: 60px;
    left: 10px;
    background: #cfc;
}
#contents {
    min-height: 450px;
    padding-left: 210px;
    background: #ccf;
}
#footer {
    height: 40px;
    background: #ffc;
}
```

레이아웃에 사용되는 CSS 속성은 float, position, top, left, right, bottom 등이다. 그리고 크게 float와 position을 사용한 레이아웃으로 크게 나뉜다. float는 하위 버전 호환성이 좋지만 마크업에 해당하는 섹션의 위치

가 중요하게 된다. 따라서 position을 이용한 레이아웃이 가변성 및 이용이 쉽다. 위의 예제를 보면 각각의 요소를 선택하는데 id를 사용하였다. 크기나 여백에 대한 정의를 한 후 위치를 설정해 주었다.

#header와 #footer는 높이만을 설정해 주었다. #menu는 absolute position이 설정되어 있다. position은 보통 레이어라고 부르는 것과 같다. 메뉴는 절대 영역으로 크기와 x,y 좌표를 가진 박스 형태로 존재하고 #content는 #menu를 위해 210px 정도를 왼쪽에 두고 구성한다. 이렇게 되면 왼쪽에 메뉴와 오른쪽에 컨텐츠를 배치하는 웹페이지가 완성된다. 이렇게 완성된 레이아웃은 그림과 같다.

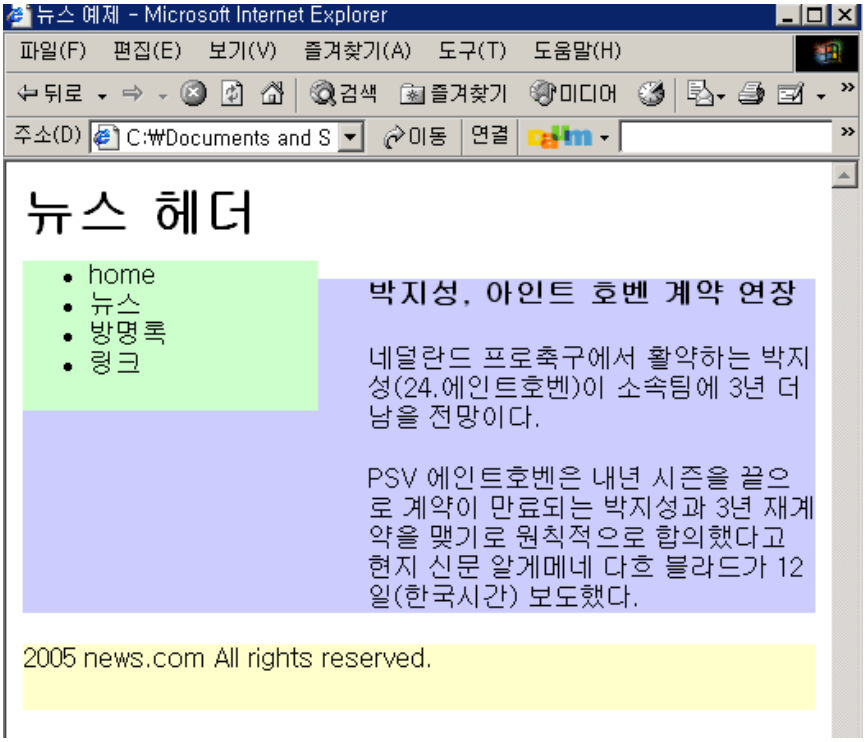


그림 7 CSS를 포함한 레이아웃

CSS 레이아웃은 브라우저 간에 차이가 존재한다. 이것은 브라우저 마다 CSS 표준안을 지원하는 구현 정도가 차이가 있기 때문이다. IE6.0은 가장 많이 사용되는 브라우저이나 3년 전에 업데이트 된 것이 마지막이기 때문에 표준 지원 정도가 파이어폭스, 오페라 같은 최신 브라우저보다 많이 떨어진다. 따라서 HTML의 크로스 브라우징과 마찬가지로 CSS를 사용할 때도 이를 피해 나가는 몇 가지 방법이 존재하며 이것을 CSS Hack이라고 한다. CSS Hack은 표준은 아니지만, CSS 해석 오류를 고려해 특정 브라우저만을 위한 코드를 넣는 방법이다. 표준으로 CSS를 작성하고 그 외 브라우저를 위한 코드를 넣어 주는 것이다.

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

IE의 경우 표준 DTD를 사용하지 않으면 padding이 표준과 다르게 렌더링 된다. 표준에서 width 100px과 padding 20px이면 총 넓이는 140px이 되지만 IE는 여전히 100px로 인식하므로 이를 고쳐 주기 위해서 IE에서만 사용되는 * html Selector를 사용하여 위와 같이 표현 하면, IE에서만 인식한다.

IE6.0은 Doctype을 이용해 Transitional 모드와 Strict 모드를 선택하여 표시할 수 있지만 IE5는 Strict 모드를 지원하지 않기 때문에 다음과 같이 수정할 필요가 있다.

```
div.ie5-hack {
    padding: 20px;
    width: 140px;
    voice-family: "\"}\"";
    voice-family: inherit;
    width: 100px;  /* IE5는 이줄을 인식하지 못한다. */
}
```

위의 코드를 이용하면 IE5는 voice-family 선언에서 }가 선언되었으므로 CSS가 종료된 것으로 인식하지만 다른 브라우저는 그 다음 줄도 인식하여 width 140px;인 것을 width 100px로 다시 인식하기 때문에 최종적으로 width는 100px이 된다. 이 두가지가 IE를 위한 가장 흔히 사용되는 CSS hack이다. 또한, IE는 아직 min-height를 지원하지 않기 때문에 height를 이용해서 높이를 정해 준다.

CSS를 이용하여 여러 가지 그림 파일로 구성하던 디자인 요소들도 쉽게 구현이 가능하다 예를 들어 다음과 같은 코드를 통해 박스를 만들 수 있다. <h3>, 에 각각 윗 배경과 아랫 배경 그림을 속성에 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>
<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>
```

이 밖에도 테이블 레이아웃이 하지 못하는 다양한 효과를 CSS 레이아웃에서 가능하다. 정부 기관 홈페이지

그림 8 CSS 레이아웃을 이용한 BOX 구조 제작

로서 CSS 레이아웃을 사용한 대표적인 웹사이트가 재정 경제부의 함께 풀어가는 종합 투자 계획(http://jumpkorea.mofe.go.kr)이라는 웹사이트이다. 이 웹사이트는 밖으로 보기에는 다른 웹페이지와 거의 차이가 없어 보이지만 레이아웃을 CSS를 사용하여 작성한 것이다.

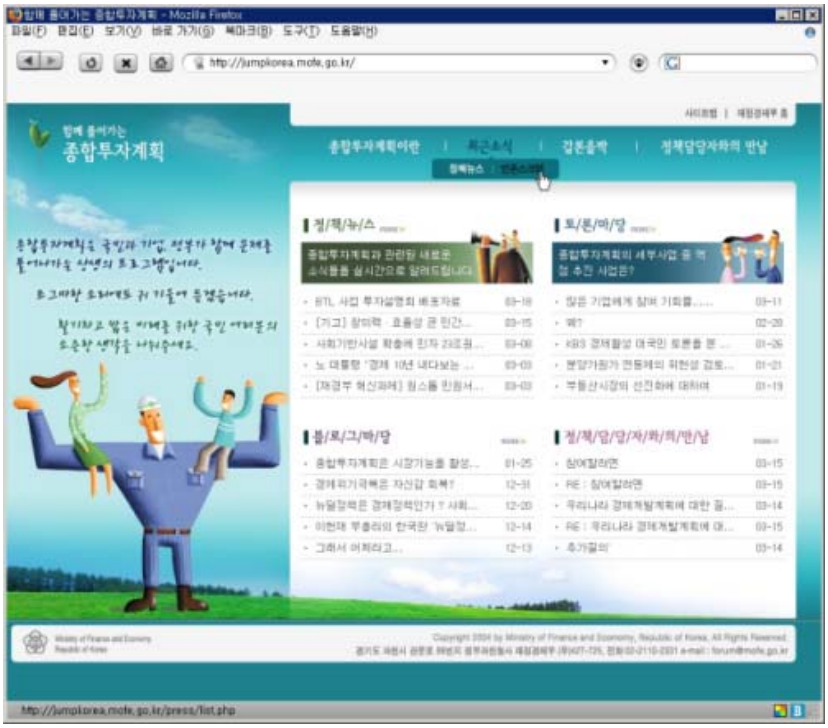


그림 9 함께 풀어가는 투자 계획을 브라우저로 본 화면

이 웹사이트에서 CSS를 사용하지 않은 경우, (파이어폭스에서는 보기->페이지 형식->형식 없음 으로 볼 수 있고, IE에서는 스크립트를 이용해 스타일을 없앨 수 있다) 복잡한 구조가 나오는 것이 아니라 그림 10과 같이 단순하고 이해하기 쉬운 정보만 나열이 된다.

이 정보 페이지를 PDA용, 텍스트용, 장애인 용으로 나누어 작성하고 싶으면 CSS 파일만 수정하여 제공하면

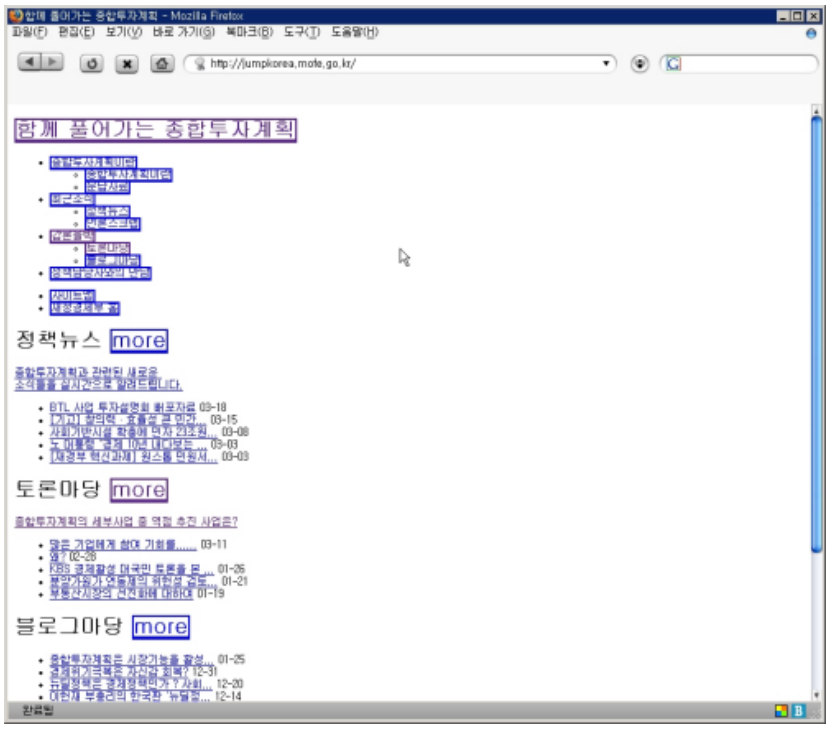


그림 10 CSS를 배제한 경우 구조적 마크업만 남은 화면

된다. 실제로 파이어 폭스에서는 이러한 스타일 변경 기능이 존재한다.

```
<link rel="stylesheet" type="text/css" href="/default.css" title="기본 형식" media="
all">
<link rel="stylesheet" type="text/css" href="/pda.css" title="PDA용" media="pda">
<link rel="stylesheet" type="text/css" href="/print.css" title="프린트용" media="scr
een">
```

위의 소스를 파이어폭스를 통해 적용하면 다양한 화면 레이아웃을 표시할 수 있다. 또 다른 예제는 재정 경제부 뉴스 웹사이트 (http://mofe.news.go.kr/)를 통해서도 확인할 수 있다.

이처럼 CSS 레이아웃이라는 웹 표준에 입각한 웹사이트는 장애인 및 노약자 같은 접근성 취약 계층에도 도움을 줄 수 있다. W3C의 웹접근성 지침 역시 XHTML과 CSS의 표준을 지키면 거의 95% 이상 지킬 수 있게 된다.

참 고 문 헌

1. Browser Archive, http://browsers.evolt.org
2. W3C, http://www.w3.org/TR
3. BBC Website Guide, http://www.bbc.co.uk/guidelines/webdev/
4. Mozilla Web Developers, http://www.mozilla.org/docs/web-developer
5. Mozilla Korean Community, http://www.mozilla.or.kr
6. Opera Developer, http://www.opera.com/doc/
7. Web standards, http://www.webstandards.org/learn/resources/dom/index.html
8. Any Browser Campaign, http://www.anybrowser.org/campaign/
9. HTML Reference, http://www.blooberry.com/indexdot/html/index.html
10. Quirksmode, http://www.quirksmode.org/viewport/compatibility.html

마치면서

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.

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 강의에서는 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.

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 강좌가 국내 웹 환경의 접근성과 브라우저 호환성을 좀 더 높이는 계기가 되기를 희망한다.