

웹 접근성을 고려한 신기술 콘텐츠 제작기법

- JavaScript, Flex, Flash를 중심으로 -

2008. 12

연구책임자 : 김석일(충북대학교 전기전자컴퓨터공학부 교수)
신인철(한국정보문화진흥원 디지털접근지원단장)
공동연구원 : 신덕식(한국정보문화진흥원 웹접근성TF팀장)
김요한(Ascent Networks Korea R&D센터 연구원)
도금호(디앤샵 기술본부 과장)
문준기(한국정보문화진흥원 웹접근성TF팀 과장)
박재표(한국정보문화진흥원 웹접근성TF팀 과장)
신현석(오페라 소프트웨어 코리아 과장)
조훈(케익소프트 실장)
최재영(아이Enter 책임연구원)
홍성원(한국어도비시스템즈 부장)

목 차

목차	i
그림목차	ix
I. 리치 인터넷 애플리케이션의 접근성	1
1. 연구목적	1
2. RIA 콘텐츠와 웹 접근성	6
2.1 RIA 콘텐츠	6
2.2 장애와 보조 기술	8
2.3 웹 콘텐츠 및 웹 애플리케이션 콘텐츠 접근성	11
3. WAI-ARIA 동향	13
3.1 RIA 기술로 인한 접근성 문제	15
3.2 WAI-ARIA 주요 내용	17
3.3 HTML 5.0 동향	18
4. RIA 콘텐츠 접근성 평가방법	20
4.1 보조 기술을 이용한 개발 단계 평가	22
4.2 자동평가 도구의 사용	24
4.3 전문가 및 사용자 평가	29

5. 웹 접근성 관련 사이트	29
5.1 웹 표준 관련 웹 사이트	29
5.2 JavaScript 관련 웹 사이트	30
5.3 Flex 관련 웹 사이트	32
5.4 Flash 관련 웹 사이트	34
5.5 MSAA 관련 웹 사이트	35
5.6 DFXP 자막 생성 소프트웨어 관련 웹 사이트	35
5.7 화면 낭독 프로그램 관련 웹 사이트	36

II. 접근성 있는 JavaScript 제작기법	37
1. JavaScript 개요	37
1.1 DOM과 DOM 스크립트	38
1.2 JavaScript 프로그램의 확장	41
2. 웹 접근성과 JavaScript	43
2.1 JavaScript에 관한 오해	43
2.2 접근성 있는 JavaScript 코딩 방법	44
3. JavaScript 접근성 지원 프로그래밍 지침	50
3.1 대체 텍스트 제공	50
3.2 키보드의 이용	54
3.3 반응시간의 조절	58
3.4 온라인 서식 구성	64
3.5 신기술의 사용	70
3.6 표준의 준수	71
4. JavaScript 프로그램의 접근성 평가방법	76
4.1 대체 텍스트 제공	78
4.2 키보드의 이용	80
4.3 반응시간의 조절	82
4.4 온라인 서식 구성	83
4.5 신기술의 사용	85
4.6 표준의 준수	86

5. JavaScript 애플리케이션 콘텐츠 구현 예	87
5.1 내비게이션 제작기법	87
5.2 접근성을 고려한 입력서식 제작기법	118
5.3 효과적인 UI 제작기법	131
5.4 JavaScript 대체 기법	142

III. 접근성 있는 Flex 제작기법	163
1. Flex 개요	163
2. 웹 접근성과 Flex 콘텐츠	165
2.1 Flex 버전	165
2.2 접근성 있는 Flex 콘텐츠 제작	166
3. Flex 콘텐츠 접근성 기술 지침	169
3.1 대체 텍스트 제공	169
3.2 자막 제공 방법	174
3.3 색상과 접근성	175
3.4 키보드의 이용	177
3.5 읽는 순서의 설정	179
3.6 플러그인 정보 제공	187
3.7 접근성 지원 컴포넌트의 사용	188
4. Flex 콘텐츠의 접근성 평가방법	190
4.1 대체 텍스트 제공	192
4.2 자막 제공 방법	194
4.3 색상과 접근성	195
4.4 키보드의 이용	198
4.5 읽는 순서의 설정	200
4.6 플러그인 정보 제공	202

5. 접근성 지원 Flex 컴포넌트	203
6. Flex 애플리케이션 콘텐츠 구현 예	209
6.1 접근성을 고려한 입력서식 제작기법	209
6.2 접근성을 고려한 차트 제작기법	219
6.3 사용자 UI를 고려한 제작기법	222

IV. 접근성 있는 Flash 제작기법	235
1. Flash 개요	235
2. 웹 접근성과 Flash 콘텐츠	238
2.1 접근성 있는 Flash 콘텐츠 제작	239
2.2 Flash 콘텐츠의 임베딩	241
3. Flash 접근성 지원 프로그래밍 지침	243
3.1 대체 텍스트 제공	243
3.2 자막 제공 방법	248
3.3 색상과 접근성	252
3.4 깜빡거리는 객체 사용 제한	253
3.5 키보드의 이용	255
3.6 접근성 지원 애니메이션	258
3.7 비디오/오디오 컨트롤	261
3.8 읽는 순서의 결정	264
3.9 구조의 제공	270
3.10 대체 페이지 제공	271
3.11 플러그인 정보 제공	273
3.12 접근성 지원 컴포넌트의 사용	274
4. Flash 콘텐츠의 접근성 평가방법	278
4.1 대체 텍스트 제공	280

4.2	자막 제공 방법	282
4.3	색상과 접근성	283
4.4	깜빡거리는 객체 사용 제한	285
4.5	키보드의 이용	286
4.6	접근성 지원 애니메이션	287
4.7	비디오/오디오 컨트롤	288
4.8	읽는 순서의 결정	289
4.9	구조의 제공	291
4.10	대체 페이지 제공	292
4.11	플러그인 정보 제공	293
5.	Flash 콘텐츠 구현 예	293
5.1	내비게이션 메뉴	294
5.2	사용자 UI를 고려한 콘텐츠	301

그 림 목 차

<그림 I - 1> 트리 컨트롤 예제	16
<그림 I - 2> Firefox의 에러 및 소스 보기 창을 연 화면	25
<그림 I - 3> Internet Explorer Developer Toolbar를 설치한 화면 ...	27
<그림 I - 4> Inspect32의 실행화면	28
<그림 II - 1> Internet Explorer에서의 웹 콘텐츠 DOM구조	41
<그림 II - 2> 스크립트가 작동할 때(좌)와 작동하지 않을 때(우)	45
<그림 II - 3> 웃는 얼굴(좌)을 클릭하면 우는 얼굴(우)로 변경되는 예제	53
<그림 II - 4> 두 개의 링크 탭과 전체리스트로 이동할 수 있는 버튼으 로 구성된 예제	62
<그림 II - 5> <그림 II - 4>에서 CSS를 제거한 경우	63
<그림 II - 6> 로그인 예제	65
<그림 II - 7> 게시판 기능 예제	68
<그림 II - 8> 레이블과 입력 창의 연결 순서	84
<그림 II - 9> 가로 1단 메뉴 예제	87
<그림 II - 10> 가로 2단 메뉴 예제	92
<그림 II - 11> 드롭다운 메뉴 예제	99
<그림 II - 12> Enter키에 의해 동작하는 가로 탭 메뉴 예제	106
<그림 II - 13> 롤오버 기능에 의해 동작하는 가로 탭 메뉴 예제	113
<그림 II - 14> 로그인 화면 예제	118
<그림 II - 15> 회원가입 예제	122
<그림 II - 16> 우편 번호 검색 예제	127
<그림 II - 17> 게시판 예제	130

<그림 II - 18> 롤링 배너 예제	131
<그림 II - 19> 스크롤 배너 예제	138
<그림 II - 20> 배경 이미지를 변경하여 롤오버 버튼을 구성한 예제	142
<그림 II - 21> 스타일을 변경하여 롤오버 버튼을 구성한 예제	145
<그림 II - 22> JavaScript 기능이 동작하는 환경에서 탭 메뉴의 화면	150
<그림 II - 23> JavaScript 기능이 동작하지 않는 환경에서의 탭 메뉴의 화면	152
<그림 II - 24> noscript 기능을 이용한 드롭다운메뉴	158
<그림 III - 1> Flex Builder 3을 이용한 접근성 지원방법	167
<그림 III - 2> toolTip 속성 사용 예제	171
<그림 III - 3> 트리 컨트롤(Tree control), 데이터 그리드(Data grid), 텍스트 영역(Text area)으로 구성된 Flex 콘텐츠 예제	182
<그림 III - 4> Tab 키에 의한 이동순서 예제	183
<그림 III - 5> 자막파일을 제공한 예제	194
<그림 III - 6> 가독성이 좋은 폰트로 변경한 모습	195
<그림 III - 7> 색상으로만 표시된 버튼	196
<그림 III - 8> 색상으로만 표시된 버튼	197
<그림 III - 9> Tab 키에 의한 초점 이동순서	201
<그림 III - 10> 로그인 창 의 예제	210
<그림 III - 11> 회원가입 예제	212
<그림 III - 12> 게시판 예제	216
<그림 III - 13> 차트 예제	220
<그림 III - 14> 상하 스크롤 바 예제	223

<그림 III - 15> 이미지 롤링 배너 예제	224
<그림 III - 16> 콘텐츠의 축소/확대 예제	230
<그림 IV - 1> ‘액세스 가능성’ 패널	240
<그림 IV - 2> 스테이지(stage)에 정적 텍스트와 이미지를 추가한 예제	244
<그림 IV - 3> 이름 필드에 대체 텍스트를 제공하는 예제	245
<그림 IV - 4> 무비의 예제(한국 Adobe 제공)	247
<그림 IV - 5> 전체 무비에 하나의 대체 텍스트를 제공하는 방법 ..	247
<그림 IV - 6> 자막파일을 지정하는 방법	250
<그림 IV - 7> 자막파일을 제공한 예제	250
<그림 IV - 8> 가독성이 좋은 폰트로 변경한 모습	251
<그림 IV - 9> 콘텐츠의 초점 이동 순서에 관한 예제(한국 Adobe 제 공)	266
<그림 IV - 10> 자막파일을 제공한 예제	282
<그림 IV - 11> 색상으로만 표시된 버튼	283
<그림 IV - 12> 색상으로만 표시된 버튼	284
<그림 IV - 13> 콘텐츠의 초점 이동 순서에 관한 예제(한국 Adobe 제 공)	290
<그림 IV - 14> 가로 1단 메뉴 예제	294
<그림 IV - 15> 접근성을 제공하기 위한 액세스 가능성 패널 화면 ..	296
<그림 IV - 16> 메뉴 전체를 접근성 있게 하는 방법	297
<그림 IV - 17> 가로 2단 롤오버 메뉴 예제	298
<그림 IV - 18> 6장의 배너로 구성된 자동 스크롤 배너 예제	302
<그림 IV - 19> 4개 버튼과 배너창으로 구성된 스크롤 배너 예제	305
<그림 IV - 20> 스크롤 버튼을 이용한 배너의 예제	307

I. 리치 인터넷 애플리케이션의 접근성

1. 연구목적

HTML(Hyper Text Markup Language)과 CSS(Cascade Style Sheet) 위주로 작성한 웹 콘텐츠는 클라이언트와 서버간의 정보교환이 페이지 단위로 이루어진다. 웹 페이지의 일부 정보를 갱신할 때에도 전체 페이지를 로딩해야 한다. 따라서 복잡한 기능을 수행하기 위해서는 많은 웹 페이지의 로딩과 전환이 필요하다. 이러한 HTML 환경은 사용자와 콘텐츠의 상호작용이 정적이어서 동적 콘텐츠나 데스크톱 프로그램과 같이 복잡한 기능을 구현하기 어렵다.

이러한 문제점을 해결하기 위하여 리치 인터넷 애플리케이션(RIA: Rich

Internet Application, 이하 RIA라 한다) 기술이 개발되었다. RIA 기술의 핵심은 첫째, 화면에 표시되는 웹 페이지의 일부분을 새로운 값으로 교체할 수 있다는 점과 둘째, 클라이언트와 웹 서버간의 정보 교환이 비동기 방식으로 이루어진다는 점이다. RIA 기술을 활용하면 서버가 사용자의 반응에 대응하여 화면의 정보를 변경할 수 있으므로 웹 상에서도 데스크톱 애플리케이션의 기능과 특징을 구현할 수 있다. 예를 들면 전체 화면을 데스크톱 프로그램처럼 구성하기, 그래픽 삽화와 상호작용이 수반되는 벡터 기반의 애니메이션 만들기, 프레젠테이션 기능, 드래그 앤 드롭(Drag and Drop)¹⁾ 기능, 컬럼 정렬, 차트 만들기 등과 같은 동적 콘텐츠가 그것이다.

이러한 RIA의 가능성이 알려지면서 RIA를 이용한 웹 콘텐츠 시장이 크게 확대되고 있다. 이에 따라 RIA 콘텐츠를 개발하기 위한 여러 가지 환경이 출시되었다. <표 I - 1>은 국내외에서 사용되고 있는 RIA 콘텐츠 개발 환경이다.

표 I - 1. RIA 개발 환경

명 칭	개발사	특징	비 고
Ajax	ASP.NET	<ul style="list-style-type: none"> • 웹 페이지 갱신 없이 고속으로 화면 전환 가능. • 웹 페이지의 일부분을 갱신하는데 필요한 데이터만을 수신할 수 있음. • 서버와의 비동기적 통신 지원함. 	
Flash	Macro-media	<ul style="list-style-type: none"> • 애니메이션, 비디오 등의 저작도로 개발되었으며, 웹 페이지를 동적으로 꾸미는데도 사용될 수 있음. 	
Flex	Adobe	<ul style="list-style-type: none"> • Flash Platform 상에서 동작하는 본격적인 RIA 도구임. • 이클립스 기반의 IDE를 지원함. • Flash와 호환되는 ActionScript 기반의 프로그래밍 환경을 지원함. 	
Silver-light	MS	<ul style="list-style-type: none"> • Visual Studio 2008 플랫폼임. • 닷넷프레임워크의 개발언어를 지원함. 	

1) 아이콘을 다른 아이콘 위에 포개어 놓음으로써 처리 내용을 지정하는 조작 개념

웹 콘텐츠 저작 기법이 전통적인 HTML 기반의 코딩방법으로부터 RIA 방식으로 진화함에 따라, HTML을 기반으로 하는 콘텐츠의 접근성 표준은 RIA 기술을 이용하여 구축한 동적 콘텐츠에 대한 접근성 표준으로 적용될 수 없게 되었다. 이에 따라 World Wide Web Consortium(W3C) 산하 WAI(Web Accessibility Initiative)에서는 웹 콘텐츠에 대한 접근성 표준을 개정하는 한편, RIA에 관한 접근성 가이드라인(Accessible Rich Internet Applications : WAI-ARIA 1.0, W3C Working Draft 6 August 2008)²⁾을 제안하였다.

W3C의 웹 접근성 표준 개정판(WCAG Technical Guidelines 2.0)³⁾은 오랜 보완작업을 거쳐 2008년 12월 11일자로 채택되었다. 기존의 WCAG 1.0⁴⁾은 HTML 기반의 웹 콘텐츠에 대한 접근성 제공 방법을 제시하고 있는데 반해 WCAG 2.0은 특정한 언어나 개발도구를 지정하지 않고 포괄적인 웹 콘텐츠에 대한 접근성 개념을 제시하고 이를 준수하는 방법을 설명하고 있다. 따라서 WCAG 2.0은 기존의 HTML 환경 뿐 아니라 RIA 기술을 이용한 웹 애플리케이션 콘텐츠의 경우에도 그대로 적용될 수 있다.

RIA에 관한 접근성 가이드라인(WAI-ARIA) 1.0에서는 접근 가능한 RIA 콘텐츠를 개발하기 위하여 고려해야 하는 규정과, HTML, CSS, JavaScript 및 Ajax 등의 기술을 이용하여 동적 콘텐츠를 개발하는 예를 제시하고 있다. 따라서 WAI-ARIA에서 제시하는 규정에 따라 RIA 콘텐츠를 구현한다면 접근 가능한 RIA 콘텐츠의 개발이 가능하다.

우리나라의 경우에 2008년 4월부터 시행된 「장애인 차별금지 및 권리구제 등에 관한 법률」(이하 '장차법')에 따라 2009년 4월부터 정부기관,

2) <http://www.w3.org/TR/wai-aria/>

3) <http://www.w3.org/TR/WCAG20/>

4) <http://www.w3.org/TR/WCAG10/>

지자체 및 공공기관을 시작으로 기관별 성격에 따라 단계적으로 이들 기관에서 제공하는 모든 웹 콘텐츠는 웹 접근성을 준수하여야만 한다. 뿐만 아니라 동적 콘텐츠에 대한 수요가 증가하면서 일부 웹 에이전시를 중심으로 Ajax, Flash, Flex, Silverlight 등과 같은 RIA 기술을 이용한 웹 콘텐츠가 개발되고 있다.

그러나 우리나라의 현실은 개발자가 참고할 수 있는 접근 가능한 RIA 콘텐츠의 개발방법이나 실무사례에 대한 자료가 매우 부족한 상태이다. 이러한 문제점을 해결하기 위한 방안으로 이 문서에서는 웹 개발자들이 처하게 될 어려움을 해소하는 데 필요한 사례들을 모아 RIA 콘텐츠 접근성 실무 지침을 개발하였다.

제 I 장 2절에서는 RIA 콘텐츠의 특징을 개괄하고, W3C의 동향을 조사하였다. 3절에서는 W3C에서 개발한 WAI-ARIA 적용사례를 요약하여 수록하였다. 4절에서는 RIA 접근성 평가방법을 기술하였으며, 5절에서는 RIA 콘텐츠 개발도구인 Flash, Flex, Ajax 등과 관련한 웹 사이트를 조사하여 수록하였다.

제 II장에서는 RIA의 대표적인 개발 기법인 Ajax, Javascript를 이용한 RIA콘텐츠의 접근성 준수방법을 기술하였다. 또한 웹 접근성을 제공하기 위한 개발 방법 및 실무사례를 수록하였다.

제 III장에서는 RIA 개발도구인 Adobe 사의 Flex 등 주요 웹 관련 신기술을 이용하여 웹 콘텐츠나 웹 애플리케이션을 제작하는 경우에 웹 접근성을 제공하기 위한 개발 방법 및 실무사례를 수록하였다.

제 IV장에서는 RIA 개발도구이자 멀티미디어 저작도구로 많이 사용되는 Flash 콘텐츠의 접근성을 제공하는 방법과 Flash 콘텐츠를 접근 가능

하게 만들기 위한 프로그래밍 기법 및 실무사례를 수집하여 수록하였다. 이 문서에서 상용 패키지인 Flash와 Flex에 관한 접근성 실무지침을 개발한 이유는 Flash 콘텐츠와 Flex 콘텐츠가 모두 Flash Player를 기반으로 접근성을 제공하기 때문이다.

Flash 콘텐츠와 Flex 콘텐츠는 아직까지 장애인들이 쉽게 접근하기 어렵다. 웹 개발자에게 이들 Flash 또는 Flex 콘텐츠의 사용을 자제하도록 요청한다고 해서 접근성 문제가 해결될 수 있는 것도 아니다. 오히려 향후 우리나라의 보조 기술 수준이 향상되었을 경우를 대비하여 지금부터 Flash 또는 Flex 콘텐츠를 접근 가능하게 구현하도록 유도하는 것이 바람직할 것이다. 이러한 판단에 따라 이 문서에서는 Flash와 Flex 환경에서 접근 가능한 웹 콘텐츠를 제작하는 방법을 제시하였다. 따라서 이 문서에서 설명한 바에 따라 웹 콘텐츠 또는 웹 애플리케이션 콘텐츠를 개발한다면 향후 보조 기술이 이들 콘텐츠를 지원하는 경우에 접근성 문제가 자동적으로 해결될 수 있을 것이다.

참고로 이 문서에서 사용하는 ‘신기술’이라는 용어의 의미는 우리나라 보조 기술이 지원하지 못하는 새로운 기술이라는 의미이다. 현재의 보조 기술 수준에서 본다면 RIA 기술은 신기술에 포함된다고 할 수 있다. 향후 우리나라의 보조 기술이 발전하여 Flash 또는 Flex 등의 웹 콘텐츠를 지원할 수 있게 되면, Flash 콘텐츠 또는 Flex 콘텐츠 개발기술은 더 이상 신기술로 분류되지 않을 것이다.

2. RIA 콘텐츠와 웹 접근성

2.1 RIA 콘텐츠

웹은 그동안 여러 과정을 거쳐 진화를 거듭해왔다. 처음에는 단순한 문자 정보를 제공하는데서 출발하였으나 점차 이미지, 사운드, 동영상 등의 멀티미디어 데이터를 제공하게 되었다. 2000년대에 들어와서는 웹 콘텐츠의 사용자 인터페이스 개선과 상호작용의 필요성이 요구되었고, 이에 따라 새로운 리치 인터넷 애플리케이션 콘텐츠(Rich Internet Application: RIA) 패러다임이 출현하게 되었다.

RIA 기술이란 웹에서 마치 윈도우용 데스크톱 프로그램을 사용하는 것과 같이 풍부한 사용자 인터페이스를 제공하는 웹 클라이언트 기술이다. 따라서 RIA 기술을 이용하면 클라이언트-서버 시스템의 특징과 웹의 특징을 모두 살린 웹 애플리케이션 콘텐츠의 개발이 가능하다. 다음 <표 I - 2>는 기존의 웹 콘텐츠와 데스크톱 프로그램, 그리고 RIA 콘텐츠를 상호 비교한 것이다.

표 I - 2. 웹 콘텐츠, 데스크톱, RIA 콘텐츠 비교

항 목	웹콘텐츠	데스크톱 프로그램	RIA 콘텐츠
빠른 설치와 정보 제공	○		○
크로스 플랫폼 특성	○		○
점진적인 다운로드	○		○
잡지형태의 표현	○		○
멀티미디어의 수용	○	○	○
표준 기반 (XML, SOAP, J2EE 등)	○		○
사용자 상호작용성		○	○
빠른 응답(페이지 갱신속도)		○	○
드래그 앤 드롭기능		○	○
확장성		○	○
온라인 및 오프라인 기능		○	○
전통적인 N-tier 개발 모델		○	○
통신기능의 간편한 추가			○

위 표에서 알 수 있듯이 기존의 웹 콘텐츠에서는 데스크톱 프로그램과 같은 풍부한 사용자 인터페이스를 표현하는 일은 거의 불가능에 가깝다. 또한 데스크톱 프로그램은 다양한 플랫폼에서 동작하도록 구현하는 것이 불가능하다. 그리고 잡지와 같은 사용자 인터페이스를 제공하기 위해서는 매우 많은 비용을 지불해야 한다. 그러나 위 표에서 알 수 있듯이 RIA 콘텐츠는 웹상에서 데스크톱 프로그램 수준의 기능을 구현할 수 있다. 일부에서는 RIA 기술로 말미암아 데스크톱 소프트웨어 시장이 몰락할 것이라는 성급한 예측을 내어 놓기도 한다.

RIA 콘텐츠는 기존의 웹 콘텐츠의 사용자 인터페이스와는 다른 새로운 사용자 인터페이스를 사용하고 있다. 예를 들면 정보의 계층적인 형태를

표현하는 트리 컨트롤(Tree Control)이나 지도를 표시하는 맵 컨트롤(Map Control) 등으로, 이는 기존의 HTML 환경에서는 사용하지 않았던 새로운 컨트롤들이다. 이러한 컨트롤들은 장애인, 특히 시각장애인에게는 더욱 생소하기 때문에 사용방법 조차 알기 힘들다.

이렇듯 RIA 콘텐츠는 장애인 접근성에 관한 새로운 패러다임을 제시하고 있다. 이러한 변화에 따라 W3C에서는 RIA 콘텐츠의 접근성에 관한 연구를 수행하고 있으며, WAI-ARIA(WAI-Accessibility Rich Internet Application)를 제안하고 있다. WAI-ARIA는 앞으로 보안을 거쳐 W3C 표준으로 채택될 전망이다.

2.2 장애와 보조 기술

컴퓨터를 다루는데 있어서 가장 많이 사용되는 감각 기관은 시각과 손, 그리고 청각이다. 컴퓨터 화면은 시각 정보를 제공하는 주요 수단으로 이용되며, 마우스와 키보드는 손을 사용하도록 형상화 되어 있다. 또한 멀티미디어는 시각과 청각을 사용하도록 구성된 콘텐츠이다.

어떤 웹 콘텐츠이든지 간에 콘텐츠를 사용하기 위해서는 시각과 청각 또는 손을 사용하는 일이 빈번하게 일어난다. 웹 콘텐츠의 내용을 인지하거나 이해하기 위해서는 컴퓨터 화면을 시각적으로 접근하여야 하며, 웹 콘텐츠를 탐색하거나 기능을 이용하기 위해서는 마우스와 키보드라는 보조적 입력 장치를 사용하게 된다. 또한 웹 콘텐츠에 포함된 사운드나 음향효과는 청각을 통하여 이용할 수 있다. 그런데 장애인들은 자신의 장애로 인하여 컴퓨터의 활용에 제약을 받는다.

시각장애인은 시각을 통하여 화면을 인지하는 능력이 떨어지거나 아예 없으므로 다른 감각기관을 이용하여 화면의 내용을 인지할 수 있도록 대

체 수단이 필요하다. 마찬가지로 청각장애인에게 제공되는 웹 콘텐츠의 음향정보는 다른 감각기관을 이용하여 인지할 수 있도록 대체 수단을 제공하여야 한다. 또한 마우스를 사용할 수 없는 장애인들에게는 키보드와 같은 보조 수단이 필요하다.

웹 콘텐츠를 다루는데 있어서 고려해야 할 장애 유형과 이들 장애를 극복하기 위하여 사용될 수 있는 수단은 다음과 같다.

- 청각장애 : 청각장애인은 마우스 또는 키보드의 사용에는 어려움이 없으나 사운드나 음향효과 등의 청각정보를 인지할 수 없으므로 청각정보를 시각정보 등과 같은 대체 방법으로 제공해 주어야 한다. 음성인식 기능을 이용하여 음성 정보를 텍스트로 변환하는 장치는 청각장애인을 위한 일종의 대체 수단이다.
- 전맹 : 전맹은 시각을 전혀 이용할 수 없기 때문에 마우스의 사용이 불가능하므로 키보드를 사용하여 모든 기능을 수행할 수 있어야 한다. 또한 시각적으로 제공되는 정보를 인지할 수 없으므로 시각으로 인지해야 하는 정보에 대한 텍스트를 제공하여 이를 인지할 수 있도록 도와야 한다. 즉 시각 정보의 대체 수단으로 텍스트 정보를 점자로 변환하거나 TTS(Text-to-Speech) 프로그램을 이용하여 음성으로 읽어줄 수 있다. 화면 낭독 프로그램은 TTS를 이용하여 음성으로 읽어주는 보조 기술의 하나이다.
- 저시력자 : 저시력자의 경우에는 마우스와 키보드를 병행하여 사용할 수 있다. 시각정보는 화면 확대 프로그램을 이용하여 판독할 수 있으며, 전맹의 경우와 같이 화면 낭독 프로그램을 병행하여 사용하면 문제를 해결할 수 있다.
- 지체장애 : 지체장애인 중 상지 장애인은 마우스의 사용이 불가능하며 키보드를 주로 사용한다. 필요시에는 자판이 큰 키보드를 사용하여 장애를 극복할 수 있다. 지체장애인이 사용하기 편리한 웹

콘텐츠는 가능한 한 키보드를 누르는 횟수를 줄일 수 있도록 단축키, 매크로 등의 기능이 제공되어야 하며 작업을 완수하는데 필요한 시간도 충분히 허용하여야 한다.

- 지적장애 : 지적장애자는 평이한 콘텐츠를 이해할 수 있다. 따라서 웹 콘텐츠에 사용하는 용어는 가능한 한 전문용어의 사용을 피하여야 한다. 어려운 용어는 풀어서 설명한다. 웹 콘텐츠를 전맹의 경우와 같이 화면 낭독 프로그램을 이용하여 읽어주는 것도 필요하다.
- 색각이상 : 색각이상자는 마우스 또는 키보드를 사용할 수 있으며, 청각의 사용이 가능하다. 웹 콘텐츠를 인지하는 과정에서 특정 색상을 구별하지 못하므로 웹 콘텐츠를 개발할 때에 색상의 차이가 확실히 구분되도록 하거나, 색상을 배제하더라도 판독이 가능하도록 웹 콘텐츠를 개발하여야 한다.

이상에서 살펴본 바와 같이 장애인이 웹 콘텐츠를 인지하고 이해하기 위해서는 키보드만으로 콘텐츠를 다룰 수 있어야 하며, 필요한 콘텐츠를 인지할 수 있는 보조 기술(Assistive Technology)이 제공되어야 한다. 또한 콘텐츠 자체의 구성 시 세심한 주의를 기울임으로써 그 내용을 충분히 인지할 수 있도록 해야 한다.

화면 낭독 프로그램이나 특수 키보드, 특수 마우스 등과 같이 장애유형별로 접근성을 향상시키는 많은 보조 기술들이 개발되어 있다. 보조 기술에 관련한 사항은 한국정보문화진흥원 정보통신 보조기기 체험관⁵⁾을 참고하라.

모든 웹 콘텐츠를 개발할 때에 접근 가능하도록 설계하는 일은 장애인 뿐 아니라 비장애인에게도 도움을 준다. 예를 들어 소음이 심한 지역에서

5) 한국정보문화진흥원 정보통신 보조기기 체험관 사이트 : <http://www.at4u.or.kr>

웹사이트에 접근하거나 웹 문서상의 음성정보를 이용하는 경우에 캡션이 제공된다면 제한적인 환경에도 불구하고 그 내용을 인지할 수 있다. 마찬가지로 영화에 자막이 제공된다면 검색 엔진을 활용하여 이 영화에 나오는 유명한 대사를 쉽게 찾을 수도 있다.

2.3 웹 콘텐츠 및 웹 애플리케이션 콘텐츠 접근성

장애인이 웹 콘텐츠에 접근하여 사용할 수 있으려면 2.2절에서 살펴본 바와 같이 적절한 보조 기술을 이용하여 해당 웹 콘텐츠에 접근할 수 있어야 하며, 웹 콘텐츠 자체도 장애인이 접근할 수 있도록 준비되어야 한다. 이를 위하여 우리나라를 비롯하여 많은 나라들이 접근 가능한 웹 콘텐츠를 제작하는 방법에 관한 기준이나 표준을 개발하여 준수하고 있다.

그리고 이들 표준이 적용되는 웹 콘텐츠의 유형도 기존의 HTML이나 CSS (Cascading Style Sheet)로 구성된 콘텐츠 뿐 아니라 JavaScript 코드가 포함된 대화형 웹 콘텐츠나 Flash 및 Flex 등의 RIA 콘텐츠도 포함된다.

우리나라의 웹 콘텐츠에 적용되는 접근성 지침은 2005년 12월 제정된 ‘인터넷 웹 콘텐츠 접근성 지침’(Internet Web Contents Accessibility Guideline; KICS.OT-10.0003; 2005. 12. 21)이다. 이 표준은 한국정보통신국가표준으로 제정되어 현재에 이르고 있다. 인터넷 웹 콘텐츠 접근성 지침(이하 ‘국가표준’이라고 한다)은 인터넷을 통하여 제공되는 웹 콘텐츠를 누구나 차별 없이 그 내용을 인지할 수 있고 운용할 수 있으며 이해할 수 있게 하기 위해서 필요한 사항을 규정하고 있다.

국가표준은 주로 HTML 및 CSS로 작성된 웹 콘텐츠의 접근성에 관한 사항을 주로 다루고 있다. JavaScript, Visual Basic 등과 같은 스크립트

코드가 포함된 웹 콘텐츠에 대해서는 스크립트를 사용하지 않아도 웹 콘텐츠 본래의 기능을 구현할 수 있도록 규정하였을 뿐, 스크립트 코드에 의하여 추가되는 기능이나 스크립트 코드 자체에 대한 접근성 제공 방법에 대해서는 아무런 규정이 없다. 마찬가지로 Ajax, Flash, Flex 또는 Silverlight를 이용하여 개발하는 RIA 콘텐츠에 대해서도 플러그인 (plug-in)에 대한 링크를 제공해야 한다는 규정 외에 RIA 콘텐츠 자체의 접근성 문제는 다루고 있지 않다.

이것은 국제적으로 통용되는 웹 접근성 표준의 경우에도 마찬가지이다. 2008년도 12월 이전까지 국제적으로 통용되어왔던 W3C(World Wide Web Consortium) 산하 WAI(Web Accessibility Initiative)에서 개발한 표준은 Web Content Accessibility Guidelines (WCAG) 1.0⁶⁾으로 우리나라 국가표준과 같이 RIA 콘텐츠의 접근성에 관한 사항을 다루고 있지 않다.

WAI에서는 웹 환경의 새로운 진화에 따라 기존의 정적 콘텐츠뿐만 아니라 RIA를 포함한 동적 콘텐츠에도 적용이 가능한 새로운 웹 접근성 지침을 개발하여 왔다. 그 결과로 2008년 12월 11일 WCAG 2.0을 W3C 표준으로 최종 확정하였다. WCAG 2.0은 4개 카테고리, 12개 지침 및 61개의 체크리스트로 구성되어 있다. 자세한 사항은 W3C 웹 사이트⁷⁾를 참고하라.

미국은 웹 접근성과 관련한 지침을 미국 장애인 복지법의 수정 조항인 508조에 규정하고 있다. 508조는 총 16개의 지침을 포함하고 있다. 508조에서도 우리나라 국가 표준과 마찬가지로 RIA에 대한 접근성을 규정하고 있지 않다. 이에 대한 자세한 내용은 미국 연방정부 508조 사이트⁸⁾를 참고하라. 미국도 웹 환경의 변화에 따라 508조의 개정 작업을 서두르고 있

6) <http://www.w3.org/TR/WCAG10/>

7) <http://www.w3.org/TR/WCAG20/>

8) <http://www.section508.gov>

다. 그 개정방향은 W3C의 경우와 마찬가지로 기존의 HTML과 CSS에 국한된 웹 콘텐츠 개발 방법을 탈피하여 스크립트 언어의 사용 추세와 RIA 콘텐츠로의 발전 추세를 감안할 것으로 알려지고 있다.

이러한 국내외적인 웹 관련 표준의 개정 작업에 발맞추어 우리나라에서도 국가표준의 개정과 함께 RIA 콘텐츠에 대한 접근성 제공 방법에 대한 연구가 필요한 상황이다. 이미 W3C 산하 WAI에서는 HTML, CSS, DOM 뿐 아니라 JavaScript, Ajax 등 신기술을 이용하여 개발하는 RIA 콘텐츠에 대한 장애인의 접근성 문제를 연구하고 있고, WAI-ARIA Suite를 그 결과물로 제출한 바 있다.

WAI에서 제공하고 있는 문서인 WAI-ARIA Best Practices는 WAI-ARIA 기술규격에 입각하여 접근성 있는 RIA 콘텐츠를 만드는 방법에 대하여 기술하고 있다. 3절에서는 WAI-ARIA의 동향에 대하여 보다 상세히 알아보기로 한다.

3. WAI-ARIA 동향

WAI에서는 리치 인터넷 애플리케이션(RIA: Rich Internet Application) 콘텐츠를 접근성 있게 개발하는데 따른 규정과 개발 사례 등을 모아 ARIA(Accessible Rich Internet Application) Suite로 묶어 제시하고 있다. WAI-ARIA Suite는 ARIA와 관련하여 WAI가 발간한 4가지 기술 자료 묶음으로, 최신판은 2008년 2월에 발간된 자료이다. WAI-ARIA Suite에 포함된 4가지 기술 자료는 다음과 같다.

- WAI-ARIA⁹⁾ : WAI-ARIA는 Ajax, HTML, JavaScript 및 관련 기술을 이용하여 동적 콘텐츠를 만들거나 고도의 사용자 인터페이스

9) <http://www.w3.org/TR/wai-aria/>

를 개발하는 과정에서 접근성을 제공하는 방법을 다룬다. 이 자료는 W3C의 웹 표준 권고안으로 채택될 예정이며, 웹 브라우저 개발자, 보조 기술 개발자, 사용자 도구나 툴킷 개발자, 접근성 평가 도구 개발자 등을 우선 고려 대상으로 삼는다.

- WAI-ARIA 프리미어(WAI-ARIA Primer)¹⁰⁾: WAI-ARIA 프리미어는 RIA 콘텐츠와 관련하여 해결하여야 할 접근성 문제, 접근성의 기본 개념 및 기술적 접근방법 등을 소개하는 문서이다. 이 문서는 W3C 워킹 그룹 규정으로 등재될 예정이다.
- WAI-ARIA 적용 기법(WAI-ARIA Best Practices)¹¹⁾: WAI-ARIA Best Practices는 웹 콘텐츠 개발자들이 WAI-ARIA를 이용하여 접근 가능한 RIA 콘텐츠를 개발하는 방법을 수록하고 있다. 이 문서는 웹 애플리케이션 개발자들을 대상으로 작성되었으나, 사용자 도구 개발자나 보조 기술 개발자들에게도 유용한 내용을 포함하고 있다. 이 문서는 W3C 워킹 그룹 규정으로 등재될 예정이다.
- WAI-ARIA 로드맵(WAI-ARIA Roadmap)¹²⁾: 이 문서는 RIA 콘텐츠를 접근 가능하게 만들기 위한 앞으로의 과정을 소개하고 있다. 뿐만 아니라 그동안 수행하여 온 내용, 앞으로 해결해야 하는 과제, 일정 등에 관한 사항도 기술하고 있다. 로드맵에 수록되었던 내용 중에서 해결된 사항들은 WAI-ARIA 프리미어 자료로 이관되어 수록된다.

WAI-ARIA에는 새로운 기술적 진화에 대응하는 접근성 향상 방법을 꾸준히 개발해 나갈 것으로 보인다. 이것을 위하여 WAI에서는 신기술에 대

10) <http://www.w3.org/TR/wai-aria-primer/>

11) <http://www.w3.org/TR/wai-aria-practices/>

12) <http://www.w3.org/TR/wai-aria-roadmap/>

한 접근성 향상 방법을 제안할 수 있는 기회를 모든 사람들에게 개방하고 있다.

3.1 RIA 기술로 인한 접근성 문제



웹 사이트들이 점점 다양한 신기술과 고도의 사용자 인터페이스 컨트롤을 제공하고 있다. 그 중 하나가 웹 사이트를 탐색하는데 사용되는 새로운 컨트롤인 트리 컨트롤(Tree Control)이다. 트리 컨트롤이란 문서의 구조를 <그림 I - 1>과 같이 계층 구조로 표시하는 방법이다. <그림 I - 1>은 공룡(Dinosauria) 메뉴의 모습으로 공룡의 이름 앞에 있는  또는  표시는 해당 공룡에 속한 종류를 표시하지 않거나 표시함을 의미한다. 따라서 Theropods는 상위계층인 Saurischians에 속한 공룡으로 그 하위계층이 Megalosaurus, Spinosaurus, ... , Maniraptors임을 보여주고 있다. 여기서 Maniraptors는 또다시 Oviraptors, Deinonychosaurs, Aves로 구분됨을 보여준다. 마찬가지로 Deinonychosaurs는 Velociraptors와 Dromaeosaurs로 구분됨을 보여주고 있다. 그러나 Oviraptors와 Aves는 하위계층을 보여주지 않고 있다.



그림 I - 1. 트리 컨트롤 예제

이상과 같이 트리 컨트롤은 문서의 구조를 시각적으로 보여줄 뿐 아니라 해당 아이টে를 클릭하면 해당 아이테의 하위 구조를 보여주거나 하위 문서를 열어 열람할 수 있도록 해준다. 그러나 문서를 계층구조로 표시하는 것은 새로운 방법이므로 이러한 계층 구조에 익숙하지 않은 사용자, 특히 시각장애인은 처음 사용하는 경우에 매우 당황스럽게 느낄 수 있다. 따라서 보조 기술들이 이러한 컨트롤에 반응할 수 있도록 개발되어야 하나, 아직까지 이러한 신기술을 따라가지 못하고 있다. 이러한 현상은 앞으로도 꾸준히 계속될 것이다.

마우스를 이용한 드래그 앤 드롭(drag-and-drop) 기능은 원래 마우스를 대상으로 개발된 기술이므로 이 기능을 키보드로 구현하는 것은 거의 불가능하다. 만일 드래그 앤 드롭 기능을 키보드만으로 구현한다고 하면, 아

무리 간단한 웹 사이트라고 할지라도 키보드를 누르는 횟수가 엄청나게 늘어날 것이다.

Ajax 또는 DHTML 등의 다양한 기술을 웹 콘텐츠 구현에 적용하는 것도 접근성을 저하시키는 원인이 된다. 예를 들어 웹 페이지가 사용자와의 상호작용을 하거나 시간에 따라 정보가 변화하는 경우에 일부 장애인들은 갱신된 콘텐츠에 대한 정보를 인지할 수 없다.

WAI에서는 이상에 열거한 경우와 같이 RIA 기술을 웹 콘텐츠에 적용함에 있어서 접근성을 제공하기 위하여 보조 기술에 어떤 정보를 어떻게 제공해야 하는가에 대한 연구 결과를 지속적으로 제시하고 있다. WAI-ARIA는 HTML5.0 임시판¹³⁾에 일부 반영되어 있다.

3.2 WAI-ARIA 주요 내용

WAI-ARIA는 사용자의 상호작용(예를 들면 상호간의 관계와 현재 상태 등과 같은 속성)을 추가하기 위한 프레임워크를 제공하고 있다. WAI-ARIA가 제시하는 방법을 적용하여 웹 페이지를 구성한다면 몇 개의 Tab 키를 누르는 것만으로도 지정된 영역으로 쉽게 이동할 수 있다.

WAI-ARIA 기술규격에서는 웹 콘텐츠 개발자에게 제공하는 6가지 사항을 규정하고 있다.

- 1) 메뉴, 트리아이템(treeitem), 슬라이더(slider), 프로그레시브미터(progressivemeter) 등과 같은 위젯의 용법
- 2) 헤딩(heading), 영역(region) 및 도표(table) 또는 그리드(grid) 와 같은 웹 페이지의 구조 표현 방법

13) <http://www.w3.org/TR/html5/>, Editor's Draft, 2009년 1월 14일

- 3) 위젯의 상태, 예를 들면 체크박스의 체크여부, 메뉴의 팝업여부(haspopup) 등의 프로퍼티
- 4) 증권 시세와 같이 갱신된 정보를 얻을 수 있는 페이지의 실시간 영역을 정의하는 프로퍼티. 또한 정보 갱신에 개입하여 매우 중요한 변화가 있을 경우에는 경고 대화 상자(alert dialog box)를 이용하여 표시하는 등에 관한 사항
- 5) 드래그 소스(drag source)와 드롭 타겟(drop target) 등과 같은 드래그 앤 드롭에 관한 프로퍼티
- 6) 웹 콘텐츠 상의 객체, 이벤트 간을 키보드만으로 이동하는 방법

이상의 6가지 규정에 대한 적용사례를 WAI-ARIA Best Practices에서 기술하고 있다. WAI-ARIA Best Practices에 기술한 적용 사례는 다음의 8가지이다.

- 1) 접근 가능한 RIA의 개발 방법
- 2) 키보드를 이용한 탐색방법
- 3) 읽는 순서와 레이블 제공방법
- 4) 문서 관리(Document Management)방법
- 5) 폼 프로퍼티 (Form Property)관련 기술
- 6) 드래그 앤 드롭(Drag-and-Drop) 지원 방법
- 7) 경고(Alert) 및 대화상자(Dialog) 사용방법
- 8) 접근 가능한 위젯(widget) 구현방법

3.3 HTML 5.0 동향

HTML은 당초 과학 문서를 웹에서 표현하는 언어로 시작되었으나 그 범용성으로 인하여 여러 가지 유형의 문서들을 표현하는데 사용되어 왔다. HTML 표준으로는 그동안 HTML 4.01¹⁴⁾이 국제 표준으로 사용되어

왔으나 HTML 4.01이 웹 애플리케이션 콘텐츠를 표현하는데 미흡하다는 점 때문에 HTML 5.0의 제정 필요성이 제기되었다.

HTML 5.0에 관한 표준 작업은 XForms 1.0의 기능을 HTML 4.01에 추가하기 위하여 2003년부터 시작되었다. 표준 작업 시작 당시에는 Opera만이 참여하였다. 2004년 초에는 Mozilla가 참여하였고, 이어서 Apple이 참여하여 본격적인 표준 개정작업이 진행되었다. 2007년에는 W3C가 표준화의 필요성을 인지하고 워킹 그룹을 구성하여 표준화 작업을 주도하였다. 그 결과 2008년 1월 22일에 W3C가 저작권을 소유한 첫 번째 초안이 발표되었다. 현재 HTML 5.0의 최신판¹⁵⁾은 2009년 1월 14일자 임시판(Editor's Draft)이다.

HTML 5.0 표준에는 정적인 웹 문서 뿐 아니라 동적 애플리케이션 콘텐츠를 개발하는 데 필요한 마크업 언어와 관련 스크립트용 API를 정의하는 내용이 담겨 있다. 애플리케이션 콘텐츠의 종류는 온라인 쇼핑몰, 검색 시스템, 게임, 온라인 전화번호부, 이메일이나 메신저와 같은 통신 프로그램, 인터넷 편집 프로그램 등과 같은 콘텐츠로 CPU 성능이 떨어지는 여건에서도 다양한 애플리케이션 콘텐츠를 접하게 되는 경우를 고려하고 있다.

RIA 개발 도구와 관련한 특정 제품(예를 들면 Mozilla의 XUL, Adobe의 Flash 및 Microsoft의 Silverlight)에 대해서는 언급하지 않고 있다. 이것은 특정 업체의 소프트웨어나 플랫폼을 지지하지 않는다는 W3C의 정책에 따른 것이다. HTML 5.0에서도 이러한 전략을 고수하고 있다.

브라우저와 관련해서도 HTML 5.0은 특정한 브라우저의 기능을 소개하

14) <http://www.w3.org/TR/html401/>, 1999년 12월 24일 W3C 표준

15) <http://www.w3.org/TR/html5/>, Editor's Draft, 2009년 1월 14일

기 보다는 브라우저가 가져야 하는 최소한의 기능에 대해서만 기술하고 있다. 특히 접근성을 저해할 수 있는 기능은 HTML 5.0에 포함시키지 않았다. 예를 들어 일부 브라우저가 blink 태그를 지원하고 있지만, HTML 5.0에서는 blink 태그를 표준으로 포함시키지 않는다. 그 대신 blink 기능을 웹 콘텐츠에 추가하려면 CSS를 사용하도록 함으로써 접근성 문제를 해결하고 있다.

4. RIA 콘텐츠 접근성 평가방법

이 문서는 접근성 있는 RIA 콘텐츠를 개발하는데 있어서 매우 일반적인 방법을 제공할 뿐이다. 따라서 이 문서에서 언급하는 사항들이 매우 개념적일 수 있다. 그러나 무엇보다도 중요한 것은 보조 기술 사용자의 관점에서 웹 콘텐츠를 평가해야 한다는 점이다.

‘웹 콘텐츠 접근성’ 개념과 비슷하지만 다르게 쓰이는 용어로 ‘웹 콘텐츠 사용성’이 있다. 웹 콘텐츠 사용성이란 사용자들이 웹 사이트가 제공하는 웹 콘텐츠의 주요 목적을 얼마나 편리하게 이용할 수 있는가를 의미한다. 예를 들면 웹 콘텐츠 사용성이 좋은 온라인 뱅킹 사이트는 온라인 뱅킹 업무를 빠르고 쉽게 처리할 수 있게 만들어져야 한다. 이 문서의 궁극적인 목표는 접근성도 좋고 사용성도 우수한 웹 사이트를 개발하는 방법을 제시하는 것이다. 여기서 접근성을 조금 떨어뜨리면 사용성이 크게 개선될 수 있는 경우와 사용성을 조금 떨어뜨리면 접근성이 크게 향상되는 경우 중에서 한 가지를 택해야 한다면 후자를 선택하여야 한다.

보조 기술이 충분한 기능을 제공하지 못하는 경우에도 웹 콘텐츠의 접근성이 떨어진다. 예를 들어 신기술이 계속 개발됨에도 불구하고 이들 신기술에 대한 접근성 지원 방법이 강구되지 않는다면, 이들 신기술을 적용한 웹 콘텐츠가 웹을 통하여 제공되는 경우에 기능이 떨어지는 보조 기술

을 사용해야 하는 장애인들은 이 웹 콘텐츠에 접근하기가 어렵다.

실제로 이 문서가 제시되는 현 시점에서 볼 때 우리나라에서 사용되는 화면 낭독 프로그램들이 Flash나 Flex, Silverlight와 같은 도구를 사용하여 개발한 RIA 콘텐츠에 대하여 충분한 접근성 기능을 지원하지 못하고 있는 점은 매우 아쉽다. 그러나 이 문서에서는 장애인들에게 필요한 보조 기술들이 RIA 콘텐츠에 접근하는데 필요한 기능이 충분하다고 가정하고 기술하고 있다. 또한 이 문서는 화면 낭독 프로그램의 기능이나 성능을 평가하는 것이 주목적이 아니므로, 화면 낭독 프로그램의 기능은 키보드 내비게이션에 따른 초점 이동문제와 초점이 제공된 객체에 대한 내용 읽기 기능을 위주로 기술하였다. 화면 낭독 프로그램의 고유 기능에 관해서는 이 문서에서 다루지 않았다.

웹 접근성을 평가하는 과정에서 항상 시각장애인의 접근성이 크게 부각된다. 그 이유는 시각장애인들이 화면을 볼 수 없기 때문에 화면에 표시되는 웹 콘텐츠를 음성으로 읽어주어야 한다는 점 때문이며, 또한 화면에 나타나는 내용을 읽어주는 과정에서 화면 낭독 프로그램이 웹 콘텐츠 개발자의 의도를 파악하여 읽어줄 수 있어야 하기 때문이다. 즉 어떤 그림을 감상할 때에 시각을 사용할 수 있는 사람들은 스스로 그림을 감상하면서 다양한 감정을 느낄 수 있지만 시각장애인들은 다른 사람의 해설을 통해서 그림을 감상할 수밖에 없으므로 그 해설이 그림을 정확하게 표현하고 있는가에 대한 논란을 피할 수 없다. 이러한 이유로 시각장애인의 웹 접근성이 중요하게 부각되고 있다. 동시에 다른 장애에 대한 고려도 필수적임을 잊지 말아야 한다.

웹 콘텐츠의 접근성을 평가하는 가장 기본이 되는 내용은 다음과 같다. 첫째 텍스트가 아닌 콘텐츠의 대체 텍스트가 제공되는가, 만일 제공되고 있다면 그 내용은 정확한가, 둘째 키보드만으로도 웹 콘텐츠가 제공하는

기능을 모두 사용할 수 있는가, 셋째 초점이 제공된 객체의 내용을 그대로 읽어주는가 하는 세 가지이다. 물론 이외에도 여러 가지 요구조건이 있지만 이들 세 가지만 정확하다면 웹 콘텐츠의 접근성이 크게 개선될 수 있다. 이는 RIA 콘텐츠의 경우에도 동일하다.

기본적으로 RIA 콘텐츠의 접근성을 평가하는 방법은 다음과 같이 세 가지(4.1 - 4.3)가 있다.

4.1 보조 기술을 이용한 개발 단계 평가

개발자는 RIA 콘텐츠를 지원하는 보조 기술을 사용하여 자신이 개발한 콘텐츠를 직접 테스트해 보아야 한다. 예를 들면 개발자가 화면 낭독 프로그램을 사용하여 개발한 콘텐츠를 테스트하는 과정을 통하여, 시각장애인이 웹 콘텐츠를 어떻게 사용하는지를 이해할 수 있기 때문이다. 참고로 보조 기술, 특히 화면 낭독 프로그램을 실행시키면 브라우저의 키보드의 조작방법이 달라지는 경우가 있다. 따라서 개발자는 화면 낭독 프로그램을 중지시키고 웹 콘텐츠를 테스트해야 할 뿐 아니라, 화면 낭독 프로그램을 실행시킨 상태에서도 테스트해 보아야 한다. 경우에 따라서는 동일한 기능을 수행하기 위하여 사용해야 하는 키보드 단축키가 달라질 수도 있다.

개발자는 웹사이트를 평가할 때에 키보드만으로 해당 웹 사이트가 제공하는 모든 기능을 사용할 수 있는가를 평가해보아야 한다. Tab 키와 Shift + Tab 키를 이용하여 객체 간을 이동하는 과정에서 초점이 이동하는 순서가 개발자의 의도대로 이동하는지, 그리고 초점이 주어진 객체의 모습이 화면상에서 구별 가능하도록 변화하는지를 확인한다. 일부 화면 낭독 프로그램의 경우에는 키보드 탐색 시 초점이 화면에서 사라진다. 이 문제가 화면 낭독 프로그램의 에러인지를 확인하려면, 화면 낭독 프로그램을

중지시키고 웹 콘텐츠를 키보드로 내비게이션하면서 초점이 표시되는지를 확인한다. 만일 화면 낭독 프로그램이 실행될 때에만 초점이 없어지는 현상이 발생한다면 이는 화면 낭독 프로그램의 에러로 간주할 수 있다. 화면 낭독 프로그램의 에러가 발견되면 해당 화면 낭독 프로그램 개발자에게 통보하여 버그를 수정할 수 있도록 한다.

다음으로 초점이 주어진 객체를 개발자가 의도한대로 읽어주는지를 확인하는 것이 중요하다. 예를 들어 Flash 콘텐츠에서는 텍스트 버튼의 레이블을 읽어주지만, 액세스 가능성 패널의 name 필드와 description 필드를 설정하면 레이블을 읽는 대신 name 필드와 description 필드를 읽어주기 때문이다.

웹 콘텐츠를 탐색하는 과정에서 마우스와 키보드를 병행하여 사용하는 경우에 마우스를 이용하여 객체를 선택하고, 이후에 키보드로 탐색을 계속하려고 하면, 키보드에 의한 탐색 시작 위치가 마우스로 선택한 객체로부터 시작되어야 한다. 그런데 이 과정에서 종종 임의의 객체에 초점이 주어지는 경우가 발생하는 것은 화면 낭독 프로그램의 에러이다. 따라서 웹 콘텐츠 개발자는 이러한 화면 낭독 프로그램의 문제점을 사전에 파악하고 웹 콘텐츠를 테스트해야 한다. 이러한 현상이 발견되면 그 내용을 해당 화면 낭독 프로그램 개발자에게 통보하여 버그를 수정하도록 한다.

우리나라에서 구입이 가능한 화면 낭독 프로그램은 엑스비전테크놀로지사의 센스리더(Sense Reader), 실로암시각장애인복지관에서 무료로 배포하는 드림보이스(Dream Voice) 6.0 및 미국 Freedom Scientific사의 Jaws for Windows 10 한국어 버전 등이다. 이 문서의 신기술을 이용한 웹 콘텐츠의 구현 예에서는, 이들 세 가지 화면 낭독 프로그램을 설치하여 사용할 때 키보드 내비게이션에 의한 초점 이동시 문제가 발생하는지 여부를 확인하였으며, 초점이동시 레이블 및 대체 텍스트를 읽어주는지 여부를

를 확인하였다.

이 문서의 주목적은 이들 세 가지 화면 낭독 프로그램의 기능이나 성능을 평가하는 것이 아니다. 그럼에도 불구하고 이 문서에서 화면 낭독 프로그램에 대한 시험 결과를 제시하는 것은, 개발자들이 자체평가 단계에서 어떤 문제점에 봉착했을 때에 이것이 웹 콘텐츠 자체의 문제인지, 아니면 화면 낭독 프로그램으로 인해 발생한 문제인지를 구별할 수 있는 기준을 제시하기 위함이다. 앞서 제시한 세 가지 화면 낭독 프로그램은 초점이 주어진 요소를 읽어주는 것 외에도 매우 많은 기능을 가지고 있다. 그러나 이 문서에서는 여타의 기능에 대해서는 실험하지 않았다. 이들 화면 낭독 프로그램에 대한 자세한 기능과 웹 콘텐츠의 내비게이션에 관한 기능 등은 각각의 화면 낭독 프로그램과 함께 제공되는 기술 자료나 웹사이트를 참고하라.

4.2 자동평가도구의 사용

웹 콘텐츠를 평가하는 가장 간단한 방법은 자동평가도구, 즉 기계적으로 접근성 준수 여부를 평가하여 그 결과를 제시해 주는 방법이다. 국내에서는 KADO-WAH라는 자동평가 도구를 무료로 다운받아 사용할 수 있으며 이 평가도구를 이용하면 국내외 웹 접근성 지침에 따라 웹 페이지를 자동으로 점검한 후에 그 문제점을 보고서로 확인할 수 있다.

요즈음 브라우저는 웹 콘텐츠를 읽어 들여 표시하는 과정에서 웹 콘텐츠의 문법 에러를 알려준다. 그중에서도 Firefox가 디버깅에 있어서 많이 사용되고 있다. Firefox는 다른 브라우저와는 달리 코드의 에러가 있는 곳을 지적할 뿐 아니라 코드 상의 잘못이 무엇인지도 정확히 설명한다. 특히 Firefox는 디버깅 콘솔을 제공하여 웹 콘텐츠의 디버깅을 편리하게 도와준다. 아래 그림은 Firefox의 에러 및 소스 보기 창을 연 상태이다.

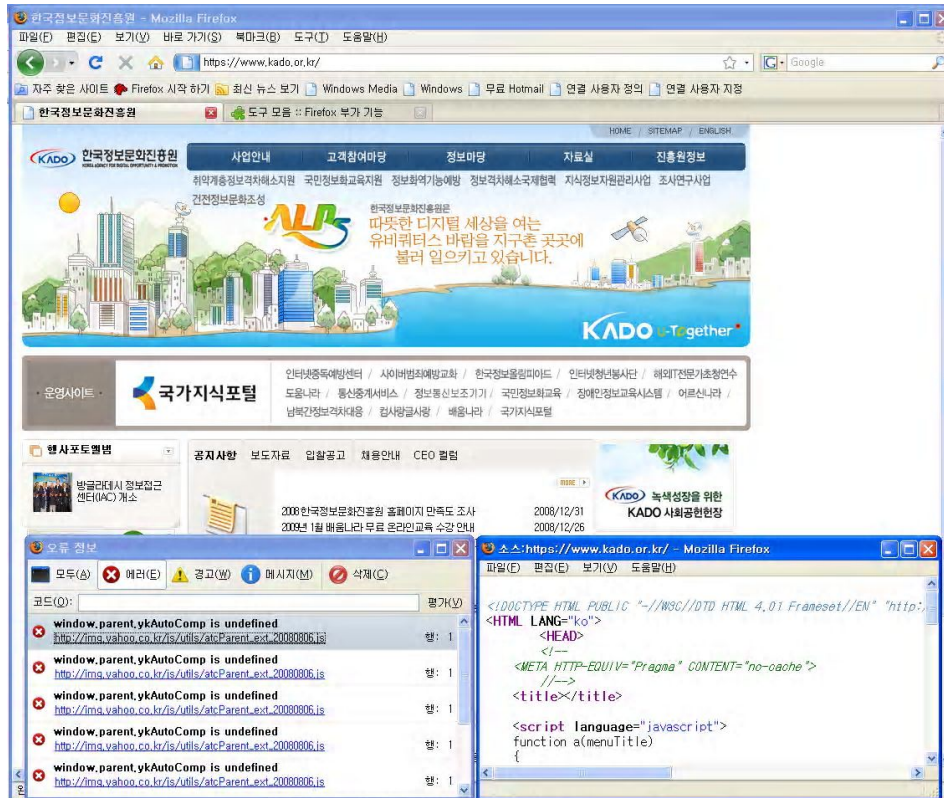


그림 I - 2. Firefox의 에러 및 소스 보기 창을 연 화면

Firefox가 디버깅 능력이 탁월하다고 하더라도 웹 콘텐츠의 논리적 에러를 찾아내는 것까지 도와주지는 않는다. 이 부분은 전적으로 개발자의 몫이다. 뿐만 아니라 자동평가도구는 대체 텍스트의 정확성과 같은 정성적인 평가는 불가능하므로 자동평가도구를 이용하여 평가한 결과가 완벽하다고 해서 접근성을 완전히 지원하는 것은 아니라는 점을 잊지 말아야 한다.

Microsoft사에서는 Internet Explorer와 함께 사용할 수 있는 HTML 태그를 정확하게 사용했는지를 평가할 수 있는 툴바(tool bar)인 Internet

Explorer Developer Toolbar¹⁶⁾를 제공한다. 이 툴바를 사용하면, 브라우저에 웹 콘텐츠의 DOM(Document Object Model)¹⁷⁾에 따른 문서 구조를 표시해준다. DOM이란 프로그램이나 스크립트가 문서 내용, 구조 및 스타일에 동적으로 접근하거나 갱신하기 위한 인터페이스이다.

뿐만 아니라 툴바는 Internet Explorer를 디버깅과 평가도구로 사용할 수 있도록 도와준다. 툴바를 이용하면 CSS를 보여주거나 이미지의 대체 텍스트를 표시할 수 있으며, HTML 태그의 이름을 찾거나 속성, CSS 프로퍼티를 검사하는데도 사용할 수 있다. Internet Explorer Developer Toolbar는 Microsoft 사이트로부터 다운받을 수 있다. 아래 <그림 I - 3>은 Internet Explorer Developer Toolbar를 설치한 화면으로 브라우저의 하단에 디버깅에 필요한 3개의 창이 설치되어 있다.

16) <http://www.microsoft.com/downloads/details.aspx?FamilyID=e59c3964-672d-4511-bb3e-2d5e1db91038&DisplayLang=en>

17) <http://www.w3.org/DOM/>

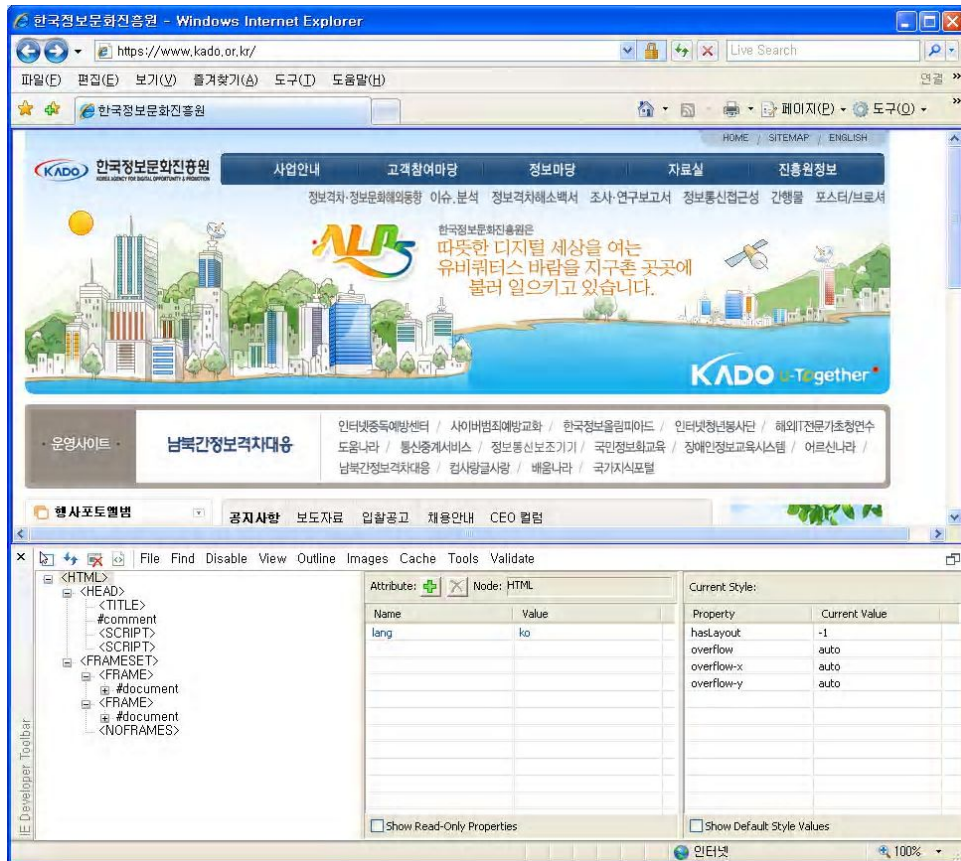


그림 1 - 3. Internet Explorer Developer Toolbar를 설치한 화면

자동평가도구는 아니지만 웹 콘텐츠의 MSAA(Microsoft Active Accessibility) 지원 여부를 평가하는 도구로 Microsoft사에서 개발한 도구인 Inspect32¹⁸⁾를 사용할 수 있다. 아래 그림은 Inspect32의 실행화면이다.

18) <http://www.microsoft.com/downloads/details.aspx?familyid=3755582A-A707-460A-BF21-1373316E13F0&displaylang=en>

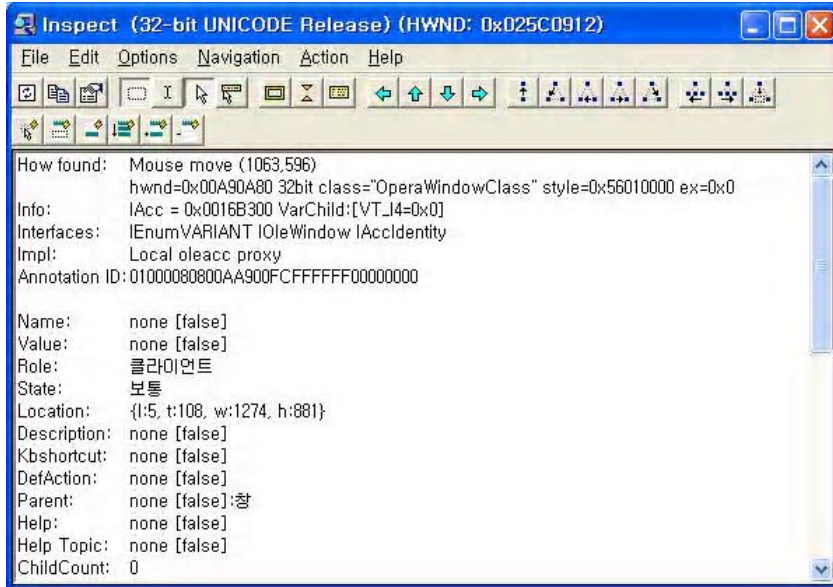


그림 I - 4. Inspect32의 실행화면

Inspect32는 웹 콘텐츠의 개발시 보조 기술을 이용하지 않고 접근성의 지원 여부를 평가할 수 있는 도구이다. Inspect32는 GUI 컴포넌트의 프로퍼티인 컴포넌트의 이름(name), 설명(description), 용도(role), 상태(state) 등을 파악하여 알려준다.

또한 Microsoft사의 AccEvent와 AccExplore를 Inspect32와 함께 사용하면, GUI나 문서 구성요소와 관련한 이벤트가 발생하였을 경우에 이를 모니터링하거나 보조 기술에 의하여 접근가능한 객체간의 계층구조를 탐색할 수 있다. Inspect32, AccEvent 및 AccExplore를 다운받을 수 있는 사이트는 관련 Microsoft 사이트를 참조하라.

Inspect32와 유사한 기능을 가지고 있는 MSAA 평가도구인 Parasoft사의 WebKing¹⁹⁾은 웹 애플리케이션 콘텐츠 개발자들이 웹 페이지를 수집

19) <http://www.parasoft.com/jsp/products/home.jsp?product=WebKing&itemId=86>

하여 각 페이지 별로 접근성을 분석한 후 그 결과를 보고해준다. WebKing은 적용할 접근성 표준에 적합한 보고서를 만들어 제공함으로써 다양한 접근성 표준을 적용할 수 있는 장점이 있다. WebKing에 관한 사용법 및 프로그램은 Parasoft사 웹사이트로부터 다운받을 수 있다.

4.3 전문가 및 사용자 평가

자동평가 결과는 기계적으로 접근성 준수여부를 평가하여 그 결과를 제시해주므로 참고용으로만 사용해야 한다. 가장 확실하고 최종적인 평가방법은 전문가 또는 장애인 사용자에 의한 평가이다. 이것은 개발된 콘텐츠의 수준과 질을 실제 사용자 관점에서 평가받을 수 있는 가장 확실한 방법이다. 이 방법을 통해서 자동평가도구를 이용한 평가에서 하지 못했던 대체 텍스트의 정확성과 웹 콘텐츠 사용상의 기능 오류 등도 찾아낼 수 있다.

그러나 전문가 평가는 비용이 수반되기 때문에 여러 차례 수행할 수 있는 방법은 아니다. 대개의 경우 사용자 평가는 웹사이트를 공개하기 직전에 한번 수행하게 된다.

5. 웹 접근성 관련 웹 사이트

5.1 웹 표준 관련 웹 사이트

- 정보통신 접근성 향상 표준화 포럼 : www.iabf.or.kr
- 한국 웹 접근성 연구소 : <http://www.iabf.or.kr/Lab>
- 미국 연방정부 508조 사이트 : <http://www.section508.gov>

- W3C Web Content Accessibility Guidelines (WCAG) 소개: <http://www.w3.org/WAI/intro/wcag.php>
- W3C WAI-ARIA : <http://www.w3.org/WAI/intro/aria.php>
- W3C Document Object Model (DOM) : <http://www.w3.org/DOM/>
- 영국 인권 위원회(The Web: Access and Inclusion for Disabled People) : <http://www.equalityhumanrights.com/en/publicationsandresources/Pages/webaccess.aspx>
- 캐나다 Common Look and Feel Standards for Internet 2.0 : <http://www.tbs-sct.gc.ca/clf2-nsi2/index-eng.asp>
- Japanese Industrial Standard X 8341-3 : <http://www.jisc.go.jp/index.html>

5.2 JavaScript 관련 웹사이트

- W3C DOM(Document Object Model) : <http://www.w3.org/DOM/>
- JavaScript Tutorial : <http://www.w3schools.com/JS/default.asp>
- JavaScript Objects Introduction : http://www.w3schools.com/JS/js_obj_intro.asp
- JavaScript Advanced : http://www.w3schools.com/JS/js_browser.asp
- JavaScript Examples : http://www.w3schools.com/JS/js_examples.asp
- JavaScript Reference : <http://www.w3schools.com/jsref/default.asp>
- HTML DOM Window Object : http://www.w3schools.com/html/dom/dom_obj_window.asp
- HTML DOM Examples : http://www.w3schools.com/html/dom/dom_examples.asp
- HTML DOM Window Object : http://www.w3schools.com/html/dom/dom_obj_window.asp
- JavaScript Tutorial for Programmers : <http://www.wdvl.com/Au>

thoring/JavaScript/Tutorial/

- JavaScript:세상에서 가장 오해가 많은 프로그래밍 언어 : <http://home.postech.ac.kr/~skyul/javascript.html>
- JAVASCRIPT KIT: <http://www.javascriptkit.com/cutpastejava.shtml>
- Mozilla JavaScript : <https://developer.mozilla.org/en/JavaScript>
- Mozilla DOM : <https://developer.mozilla.org/en/DOM>
- Mozilla AJAX : <https://developer.mozilla.org/en/AJAX>
- MicroSoft JScript : [http://msdn.microsoft.com/en-us/library/hbxc2t98\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/hbxc2t98(VS.85).aspx)
- JScript Fundamentals (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/6974wx4d\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/6974wx4d(VS.85).aspx)
- Advanced JScript (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/b9w25k6f\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/b9w25k6f(VS.85).aspx)
- Introduction to Regular Expressions (Scripting) : [http://msdn.microsoft.com/en-us/library/6wzad2b2\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/6wzad2b2(VS.85).aspx)
- Microsoft JScript Features - ECMA (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/d33e43t3\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/d33e43t3(VS.85).aspx)
- Microsoft JScript Features - Non-ECMA (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/4tc5a343\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/4tc5a343(VS.85).aspx)
- JScript Errors (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/7th8s2xk\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/7th8s2xk(VS.85).aspx)
- JScript Functions (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/6fw3zxcx\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/6fw3zxcx(VS.85).aspx)
- JScript Methods (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/c6hac83s\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/c6hac83s(VS.85).aspx)
- JScript Objects (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/htbw4ywd\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/htbw4ywd(VS.85).aspx)
- JScript Operators (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/htbw4ywd\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/htbw4ywd(VS.85).aspx)

[microsoft.com/en-us/library/ce57k8d5\(VS.85\).aspx](http://microsoft.com/en-us/library/ce57k8d5(VS.85).aspx)

- JScript Properties (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/xyad316h\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/xyad316h(VS.85).aspx)
- JScript Statements (Windows Scripting - JScript) : [http://msdn.microsoft.com/en-us/library/3xcfc93\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/3xcfc93(VS.85).aspx)
- JScript Blog : <http://blogs.msdn.com/jscript/>
- Prototype(JavaScript framework) : <http://prototypejs.org/>
- jQuery : <http://jquery.com/>

5.3 Flex 관련 웹 사이트

- Adobe 접근성 사이트 : <http://www.adobe.com/accessibility/products/flex>
- Adobe Accessibility Overview : <http://www.adobe.com/accessibility/products/flex/overview.html>
- Using Adobe Flex Application with JAWS : <http://www.adobe.com/accessibility/products/flex/jaws.html>
- Adobe Accessibility Best Practice for Flex : http://www.adobe.com/accessibility/products/flex/best_practices.html
- Adobe Accessibility Examples : <http://www.adobe.com/resources/accessibility/examples.html>
- Adobe White Papers : <http://www.adobe.com/macromedia/accessibility/whitepapers/>
- Adobe Accessibility Validate : <http://www.adobe.com/macromedia/accessibility/gettingstarted/validate.html>
- Adobe Accessibility Blogs : <http://blogs.adobe.com/accessibility/>
- Yahoo Flash와 Flex의 접근성 : <http://www.yswfblog.com/blog/2008/07/22/accessible-trueflash-and-flex-accessibility-docs/>

- Flex 접근성 개요 : <http://flexdocs.kr/docs/flex2/docs/00001025.html>
- Flex 접근성에 대한 best practice : <http://frankieloscavio.blogspot.com/2008/02/accessibility-best-practices-for-flex.html>
- RIA Accessibility with Flex : <http://www.javaworld.com/community/node/1746>
- Adobe Flex에서의 접근성 : <http://www.9xb.com/blog-accessibility-in-adobe-flex.html>
- Flex and Accessibility by Giorgio Natil : <http://www.onflex.org/ted/2008/09/360flex-sj-2008-flex-and-accessibility.php>
- Accessibility with Files FLEX : <http://kb.yworks.com/article295.html>
- Flex Documentation : http://livedocs.adobe.com/flex/3/html/accessible_5.html
- Flex 2.0 Keyboard Documentation : http://livedocs.adobe.com/flex/3/html/accessible_5.html
- Controls : <http://msdn.microsoft.com/library/en-us/dnwue/html/ch08c.asp>
- Guidelines for Keyboard User Interface Design : <http://msdn2.microsoft.com/en-us/library/ms971323.aspx>

5.4 Flash 관련 웹 사이트

- Adobe 접근성 사이트 : <http://www.adobe.com/accessibility/>
- Adobe Accessibility Overview : <http://www.adobe.com/accessibility/products/flash/author.html>
- Adobe Flash Accessibility Design Guideline : http://www.adobe.com/accessibility/products/flash/best_practices.html
- Adobe Flash White Papers : <http://www.adobe.com/accessibility/whitepapers/>

- Flash 자습서 페이지 : http://www.adobe.com/go/learn_fl_tutorials_kr
- Flash 접근성 웹 사이트 : http://www.adobe.com/go/Flash_accessibility_kr/
- Flash 샘플 페이지 : http://www.adobe.com/go/learn_fl_samples_kr
- 멀티미디어 접근성 지침 사이트 : http://ausweb.scu.edu.au/aw03/papers/arch__with_celic_/paper.html
- 접근 가능한 Flash 콘텐츠 제작 사이트 : <http://www.webaim.org/techniques/flash/>
- Flash의 접근성 문제 : <http://www.isolani.co.uk/blog/access/AccessibilityInTrouble1Flash>
- Flash와 접근성 : <http://www.usability.com.au/resources/flash.cfm>
- Flash 접근 : <http://www.alistapart.com/articles/unclear/>
- HP.com에서의 Flash 사용 : http://welcome.hp.com/country/us/en/flash_info.html
- Flash MX 접근성 문제 : http://welcome.hp.com/country/us/en/flash_info.html
- Macromedia Flash : <http://www.unc.edu/webaccess/flash.html>
- 장애인의 Flash 접근성과 사용성 : <http://www.nngroup.com/reports/accessibility/flash/>
- Detecting Assistive Technologies : <http://www.paciellogroup.com/blog/?p=61>
- Information and news about accessibility for people with disabilities in Adobe products : <http://blogs.adobe.com/accessibility/>

5.5 MSAA 관련 사이트

- Active Accessibility 2.0 Documentation : http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msaa/msaastart_9w2t.asp

- MS Active Accessibility 2.0 SDK Tools (Inspect32, AccExplore32, AccEvent32) : <http://www.microsoft.com/downloads/details.aspx?FamilyId=3755582A-A707-460A-BF21-1373316E13F0&displaylang=en>
- Microsoft Active Accessibility: Architecture : <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnacc/html/actvaccess.asp>

5.6 DFXP(Distribution Format Exchange Profile) 자막 생성 소프트웨어 관련 웹 사이트

- MAGpie : Media Access Generator (<http://ncam.wgbh.org/webaccess/magpie/>)
- Captionate : Manitu Group (<http://www.manitugroup.com/>)
- Caption Wrap : Mark Hall Sales Associates (<http://www.mhsa.us/flashcap.html>)
- Hi-Caption Studio : Hi-Software(<http://www.hisoftware.com/hmcc/>)
- Softel Swift : <http://www.softel.co.uk/>
- Subtitle-horse : <http://subtitle-horse.org/>

5.7 화면 낭독 프로그램 관련 웹 사이트

- 센스리더 엑스비전테크놀로지 센스리더 : http://www.xvtech.com/xvision_eng/
- 드림보이스 실로암시각장애인복지관 드림보이스 : <http://www.silwel.or.kr/>
- Jaws for Window : <http://freedomscientific.com/downloads/jaws/jaws-downloads.asp>

- Jaws for Window 한국어 음성파일 : <http://freedomscientific.com/downloads/RealSpeak-Solo-Direct-Voices/RealSpeak-Solo-Direct-Downloads.asp>

II. 접근성 있는 JavaScript 제작기법

1. JavaScript 개요

JavaScript는 HTML, CSS(Cascade Style Sheet)와 함께 현대적인 웹 페이지를 구축하기 위한 세 가지 요소 중 하나이다. HTML은 웹 페이지의 구조를 제공하며, CSS는 웹 페이지의 표현 스타일을 제공하고, JavaScript는 웹 페이지에서 특정 기능을 수행하는 역할을 담당한다. 이 과정에서 프로그램이나 스크립트는 웹 문서의 내용, 구조 및 스타일에 관한 정보에 접근하거나 갱신하기 위하여 DOM(Document Object Model)이라는 인터페이스를 이용한다.

웹 콘텐츠는 HTML과 CSS를 가지고도 어느 정도의 기능을 수행할 수

있으나 실제로는 그 기능이 매우 제한적이다. 이와는 달리 JavaScript는 사용자가 버튼을 클릭하거나 브라우저 창의 크기를 조정하기도 하고 텍스트 필드에 데이터를 입력하는 등과 같이 웹 페이지 내부에서 일어나는 이벤트를 감지하고 이에 따른 조치를 취할 수 있다. 또한 JavaScript는 일종의 프로그래밍 언어이므로 계산을 수행하거나 웹 페이지의 구성요소를 동적으로 변경할 수 있다. 특히 입력한 데이터의 유효성을 검증하는 것과 같은 상호작용도 가능하다. JavaScript의 이러한 기능을 이용하면 웹 페이지의 일부 정보를 교체하더라도 웹 페이지 전체를 리로드(reload)할 필요가 없어서 웹 페이지를 역동적으로 만들 수 있다.

제 II장에서는 JavaScript를 이용하여 웹 콘텐츠를 제작하는 과정에서 접근성을 제공하기 위하여 고려해야 하는 사항을 기술한다. 따라서 이 문서에서 제시한 절차에 따라 개발된 웹 콘텐츠는 보조 기술을 사용하는 장애인들에게 접근 가능한 웹 콘텐츠가 될 것이다. 그러나 이 문서에서는 JavaScript 코드를 작성하는 데 필요한 전반적인 내용을 기술하지는 않는다. 단지 JavaScript 코드를 개발하는 과정에서 접근성을 지원하기 위하여 지켜야 하는 최소한의 지침에 대하여 설명하고 필요한 예를 제공할 뿐이다. JavaScript, HTML, CSS 및 DOM에 관한 자세한 사항은 관련 자료나 W3C 사이트를 참고하도록 하라.

1.1 DOM과 DOM 스크립트

DOM(Document Object Model)은 프로그램이나 스크립트를 사용하여 문서의 내용, 구조 및 스타일에 접근하거나 갱신할 수 있도록 인터페이스를 정의한 것이다. DOM은 HTML이나 XML과 같은 마크업 언어의 문서 구조와 이들 문서의 일부분을 변경할 수 있는 수단에 관한 표준이라고 할 수 있다. 여기서 DOM을 다루는 프로그래밍 기법을 DOM 스크립트라고 한다. 일반적으로 DOM 스크립트는 JavaScript를 의미한다.

DOM 표준은 W3C에서 제정하였다. W3C는 1998년에 DOM Level 1을 발표하였다. DOM Level 1은 HTML이나 XML 문서를 완벽하게 지원할 뿐 아니라 문서의 어떤 부분도 변경할 수 있는 수단도 제공하였다. 2000년 하반기에 발표한 DOM Level 2는 처음으로 getElementById 함수와 이벤트, 그리고 XML 네임 스페이스와 CSS를 지원하기 시작하였다. 2004년 4월 발표한 DOM Level 3는 XPath와 키보드 이벤트 핸들링을 지원할 뿐 아니라 XML과 같이 문서를 시리얼라이징(serializing) 할 수 있는 인터페이스를 제공하였다.

2005년에는 JavaScript²⁰⁾를 지원하는 브라우저인 Internet Explorer 6.0, Mozilla, Firefox, Opera, Safari 등이 W3C DOM의 많은 부분을 지원하게 되었다. 이로써 DOM을 지원하는 브라우저들 간에는 문서의 호환성이 존재하게 되었다. 뿐만 아니라 DOM은 플랫폼, 운영체제 및 사용하는 스크립트 언어와는 독립적인 구조로 되어 있다.

JavaScript는 HTML과 CSS로 구성된 문서에 동적인 효과를 줄 수 있다는 점에서 많은 인기를 누리고 있다. 그동안 정적이었던 웹 페이지에 동적인 변화를 추가한 결과는 매우 충격적이어서 JavaScript를 이용하여 개발한 웹 콘텐츠를 DHTML(Dynamic HTML)이라고 부르기도 하였다. JavaScript는 초기에 주로 마우스에 반응하는 메뉴나 간단한 애니메이션 효과를 구현하는데 사용되었다. 예를 들어 마우스를 따라다니는 이미지를 만들어 넣는다거나 화면에 눈을 내리게 하는 효과 등이 그것이다.

DHTML과 DOM 스크립팅을 명확히 구분하는 것은 쉬운 일이 아니다. DOM 스크립팅은 DHTML에 뿌리를 두고 있다. 그러나 DOM 스크립팅

20) JavaScript는 Netscape가 ECMA-262를 기준으로 구현한 스크립트 언어임. 때론 ECMAScript라고도 함.

이 보다 구조적이라고 할 수 있다. DOM 개념과 DOM 메소드를 이용하기 이전의 스크립트 코딩 방법을 DHTML이라고도 하며, 그 이후의 스크립팅을 DOM 스크립팅이라고 구분하기도 한다. DHTML이 DOM에 비하여 시기적으로 먼저 제시되었으므로 크로스 브라우징의 개념이 상대적으로 약하다. DHTML에서는 브라우저별로 서로 다른 객체 참조 구문을 사용하기 때문에 서로 다른 브라우저에서는 잘 작동하지 않는 경우가 많았다. 이에 비하여 최신의 브라우저들은 모두 DOM을 지원하고 있으므로 DOM인터페이스를 사용하여 스크립트를 코딩할 경우에는 대부분의 브라우저에서 동일한 효과와 기능을 제공할 수 있다. 따라서 DOM 스크립팅에 의하여 크로스 브라우징이 가능한 웹 콘텐츠를 훨씬 편리하게 개발할 수 있게 되었다.

DOM 스크립트는 여러 가지 프로그램 언어로 구현이 가능하지만 이 문서에서는 JavaScript를 이용하는 것으로 간주한다. 그 이유는 모든 브라우저가 지원하는 스크립트 언어가 JavaScript이기 때문이다.

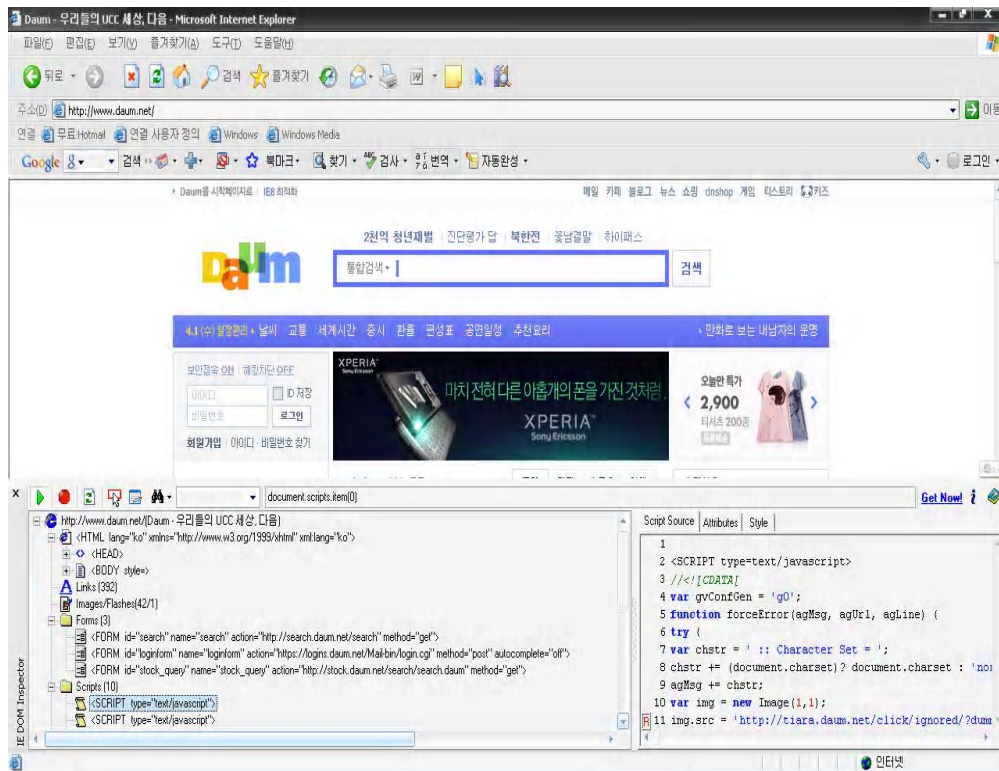


그림 II - 1. Internet Explorer에서의 웹 콘텐츠 DOM구조

1.2 JavaScript 프로그램의 확장

JavaScript가 보편화되고 적용 기법들이 계속 발전함에 따라서 JavaScript를 이용하는 새로운 개념들이 등장하게 되었다. 그 중에서 가장 대표적인 경우가 바로 Ajax(Asynchronous JavaScript and XML)이다. 이는 비동기 JavaScript와 XML을 의미하며, JavaScript가 웹 서버와의 비동기 통신을 통하여 필요한 데이터를 얻어서 이를 화면에 반영하므로 화면 제어를 보다 세밀하게 할 수 있고, 동시에 웹 콘텐츠와 사용자간의 상호작용을 계속할 수 있는 특징이 있다.

Ajax 기술의 핵심은 클라이언트와 웹 서버간의 비동기 통신(XMLHttpRequest) 방법²¹⁾이다. 또한 통신 방식의 표준을 제정할 필요성이 대두됨에 따라 W3C에서는 XMLHttpRequest에 관한 초안(2008년도 4월)을 제안한 바 있다.

Ajax 기술이 발전함에 따라 데스크 톱 소프트웨어(Desktop Software)를 웹에서도 제공할 수 있게 되었다. 웹에서 제공하는 애플리케이션 콘텐츠를 리치 인터넷 애플리케이션 콘텐츠(RIA: Rich Internet Application Content)라고 하며, 그 중심에 JavaScript와 비동기 통신 기법인 XMLHttpRequest가 위치하고 있다.

웹 콘텐츠를 개발함에 있어서 새로운 개념의 도입은 접근성 측면에서 새로운 부담이 될 수 있다. 그러나 이러한 우려는 접근성에 대한 잘못된 이해로부터 출발한다. 접근성을 준수하여 웹 콘텐츠 또는 웹 애플리케이션 콘텐츠를 개발하는 것은 다양한 사용자를 존중하고 다양한 환경에서도 무리 없이 작동하는 고품질의 콘텐츠를 만들기 위함이다. 따라서 좋은 웹 콘텐츠 또는 좋은 웹 애플리케이션 콘텐츠는 반드시 웹 접근성을 준수하여야 한다.

접근성을 준수하는 과정에서 부분적인 비용의 증가가 예상된다. 그러나 이 비용은 웹 콘텐츠의 완성도를 높이는 비용으로 보아야지 단순히 접근성을 제공하기 위한 비용으로 간주하는 것은 옳지 않다. 이를 반영하듯이 W3C에서는 Ajax나 RIA와 같이 JavaScript와 관련한 새로운 패러다임에 대응하는 접근성 표준을 계속 제안하고 이를 준수하도록 권고하고 있다.

21) <http://www.w3.org/TR/XMLHttpRequest/>

2. 웹 접근성과 JavaScript

2.1 JavaScript에 관한 오해

JavaScript는 잘 사용하면 사용자에게 좋은 경험을 제공할 수 있지만 잘못 사용할 경우에는 오히려 사용자에게 좋지 못한 경험을 제공하거나 접근성을 떨어뜨리는 요인이 된다. 접근성이 떨어지는 대부분의 이유는 JavaScript를 잘못 사용하거나 호환성이 없는 코딩 방법에 기인한다.

JavaScript는 다른 웹 개발 언어와는 달리 사용자 환경인 브라우저에서 작동하게 된다. 그런데 브라우저는 종류도 매우 다양하고 브라우저 별로 기능도 서로 다르다. 따라서 JavaScript의 접근성을 다룰 때에는 흔히 'JavaScript가 없는 상황'을 전제로 하는데 이는 JavaScript를 사용하지 않는 경우라기보다는 문제를 일으키지 않는 이상적인 환경을 의미하는 것으로 보아야 한다. 실제로 이상적인 환경을 가정하여 개발한 웹 콘텐츠는 접근성을 제공할 뿐 아니라 구조적인 견고함으로 인하여 모바일 환경과 같은 다양한 환경에서도 높은 호환성을 제공한다.

우리나라에서는 보안을 강화한다는 이유로 JavaScript를 이용하여 불필요한 처리를 하는 경우가 많다. 예를 들면 URL을 보이지 않게 한다든가, 링크를 JavaScript로 처리한다든가, 서식의 전송 기능을 JavaScript로 처리하는 등이 바로 그것이다. 결론적으로 말하자면 이러한 JavaScript 코딩 방법들은 콘텐츠의 보안에는 전혀 도움이 되지 않을 뿐 아니라 접근성과 사용성을 크게 저하시킨다. URL을 보이지 않게 하는 일은 사용자가 피싱에 노출되었을 때 스스로 방어할 수 없게 되는 문제를 야기하기도 한다. 이러한 잘못된 인식으로 인하여 정작 서버 측의 보안을 등한히 하는 경우도 종종 발생하고 있다.

2.2 접근성 있는 JavaScript 코딩 방법

접근 가능한 웹 콘텐츠를 개발하는 과정에서 JavaScript를 적용하는 방법에는 세 가지가 있다. 첫째는 JavaScript 코딩시에 접근성을 크게 고려하지 않고 우선 웹 콘텐츠를 개발하고, 이후에 접근성을 떨어뜨리는 부분을 하나씩 해결해 나가는 방법이다. 둘째는 HTML과 CSS를 사용하여 우선 웹 콘텐츠를 구현하고, 상호작용이 필요한 부분을 JavaScript로 코딩하는 방법이다. 세 번째 방법은 두 번째 방법에서 JavaScript로 코딩하는 부분을 스크립트를 제거하더라도 웹 콘텐츠의 이용에 문제가 없는 수준까지로 제한하는 방법이다. 이상의 세 가지 방법을 자세히 살펴보기로 하자.

2.2.1 적절한 낮춤(Graceful Degradation)

JavaScript를 이용하여 접근성을 지원하는 웹 콘텐츠를 개발하는 과정에서 HTML, CSS 및 JavaScript를 이용하여 코딩하고, 접근성이 떨어지는 부분의 JavaScript 부분을 대체기법을 이용하여 보완하는 방법이다. 이 방법은 고도의 JavaScript 코딩방법을 우선 적용한 후에 접근성을 제공할 수 없는 스크립트 부분을 HTML이나 CSS로 재작업을 하게 된다. 따라서 높은 기술 수준의 스크립트를 낮은 수준의 HTML이나 CSS 코딩으로 대체해 나가므로 이를 ‘적절한 낮춤’이라고 한다.

일반적으로 접근성을 지원하는 코딩방법이 어렵거나 보조 기술의 한계가 있는 개발방법을 높은 수준의 기술이라고 하고, 반대로 HTML 기술을 가장 낮은 단계의 기술이라고 할 수 있다. 예를 들어 Flash, Flex, JavaScript 등을 이용한 코딩 기술이 높은 수준의 기술이라고 할 수 있으며, 그 다음은 CSS, 그리고 HTML 기술의 순서로 기술수준이 낮다고 할 수 있다.

아래 보인 코드는 JavaScript로 구현한 프로그램으로 문서의 일부 영역에 도움말 링크를 동적으로 제공하는 코드이다. 이 부분은 JavaScript를 지원하지 않는 상황에서도 접근성을 보장하기 위해서 <noscript> 요소를 이용해서 대체 콘텐츠를 제공하고 있다. 따라서 기술 수준의 적절한 낮춤을 통해 접근성을 향상시키고 있다. <그림 II- 2>는 스크립트를 지원할 때와 지원하지 않을 때의 브라우저 모습을 보여준다.

O (Good)

```
<script type="text/JavaScript">
  var width = 600;
  document.write('<div id="sublayer" style="width:' + width + '">');
  document.write('<p><a href="help.html">도움말</a></p>');
  document.write('</div>');
</script>
<noscript>
  <p><a href="help.html">도움말</a></p>
</noscript>
```



그림 II - 2. 스크립트가 작동할 때(좌)와 작동하지 않을 때(우)

적절한 낮춤 방법을 적용하면 신기술을 사용할 때, 신기술을 지원하지 않는 환경에 대한 접근성을 보장할 수 있다. 특히 신기술이 제공하는 기

능이 매우 많고 복잡한 경우에 핵심기능만을 간추려서 대체 콘텐츠를 제공할 수 있기 때문에 효과적으로 접근성을 보장할 수 있다. 여기서 신기술이라 함은 우리나라의 보조 기술이 지원하지 못하는 기술을 의미한다. 예를 들어 우리나라의 화면 낭독 프로그램이 JavaScript를 충분히 지원하지 못하고 있으므로, JavaScript를 이용한 웹 콘텐츠 구현 기술은 신기술로 분류될 수 있다. 만일 우리나라에서 구입 가능한 화면 낭독 프로그램이 JavaScript를 충분히 지원하게 된다면 JavaScript는 더 이상 신기술로 분류되지 않을 것이다.

2.2.2 점진적 향상(Progressive Enhancement)

코딩 기술의 점진적 향상은 적절한 낮춤과는 반대 개념으로 접근성을 낮은 수준에서부터 제공하도록 하는 것이다. 접근 가능한 웹 콘텐츠나 웹 애플리케이션 콘텐츠를 구현하고자 할 경우에 핵심 기능과 부가 기능을 분리하여 핵심 기능은 낮은 수준의 기술로 구현하고, 부가 기능은 높은 수준의 기술로 구현하는 것이다. 예를 들면 웹 콘텐츠의 내용은 HTML로 구성하고, 웹 페이지의 디자인은 CSS를 이용하여 구현하며, 마지막으로 동적인 기능은 JavaScript를 이용하여 구현하는 것이다.

아래의 프로그램은 ‘도움말’ 링크를 클릭했을 때 해당 페이지로 이동하는 코드의 예를 보여준다. 이 예에서 보듯이 핵심 부분인 내용과 링크는 HTML로 구성하고, 나머지 기능은 JavaScript로 구현하였다.

O (Good)

```
<div id="sublayer">
  <p><a href="http://./help.html">도움말</a></p>
</div>
<script type="text/JavaScript">
  var width = 600;
  document.getElementById('sublayer').style.width = width + "px";
</script>
```

점진적 향상 방법을 적용하기 위해서는 먼저 콘텐츠나 기능의 핵심을 파악하는 것이 중요하다. 핵심 기능과 부가적인 기능을 어떻게 판단할 것인가가 점진적 향상 방법의 핵심이라고 할 수 있다. 요즘과 같이 대화형 웹 애플리케이션 콘텐츠를 선호하는 상황에서는 자칫 효과와 관련한 기능을 필수 기능으로 오해하기 쉬우나, HTML 수준에서 제공할 수 있는 아주 기본적인 기능들이 필수 기능임을 명심해야 한다.

2.2.3 겸손한 구현(Unobtrusive)

점진적 향상 방법은 결국 웹 콘텐츠를 내용, 표현, 기능의 세 가지로 분리하여 내용과 표현을 먼저 구현하고, 여기에 구현 기능을 추가하게 된다. 극단적으로는 이들 세 가지 웹 콘텐츠 구성요소를 완전히 분리하여 접근성을 높이려는 시도가 바로 겸손한 구현 방법이다. 따라서 2.2.2절에서 설명한 점진적 향상 기법을 기술적인 관점, 특히 JavaScript의 관점에서 해석한 것이 겸손한 구현방법이라고 볼 수 있다.

JavaScript는 DOM 메소드를 사용해서 동적으로 HTML이나 CSS 코드를 수정, 추가 또는 삭제할 수 있다. HTML 코드에 들어있는 이벤트 핸들러를 모두 JavaScript로 옮기는 것도 가능하다. 이렇게 기능과 관련한 JavaScript 이벤트 핸들러나 HTML 문서 내에 삽입된 JavaScript를 모두

분리해 낼 수 있다. HTML과 JavaScript가 완전히 분리된 상태에서는 HTML 문서가 독립적으로 의미를 갖고 기능을 할 수 있다. 이 상태에서 JavaScript는 HTML 문서 안에 포함되어 있지도 않고 지나친 개입도 하지 않는다. 자신의 역할이 필요할 때에만 HTML 문서와 기능적으로 교류하게 된다. 따라서 JavaScript가 지나치게 개입하지 아니하므로 이를 겸손한 구현방법이라고 부른다.

아래의 코드는 마우스를 이미지 위에 올렸을 때 이미지가 바뀌는 기능을 JavaScript로 구현한 것이다. 스크립트에서 먼저 함수를 선언하고 이를 HTML의 onmouseover 이벤트 핸들러를 이용하여 연결하였다. 여기서 이벤트는 JavaScript가 제공하는 기능이고 링크는 HTML이 제공하는 기능이다. HTML 속성 내에서 이벤트를 부여하였기 때문에 JavaScript 코드가 HTML내에 포함되어 있다.

X (Bad)

```
<script type="text/JavaScript">
  function changImage(img) {
    img.src = img.src.replace(".gif", "_on.gif");
  }
</script>
<p>
  <a href="about.html">
    </a>
  </p>
```

위의 예제를 겸손한 구현 방법을 이용하여 구현하려면, 우선 HTML 부분과 JavaScript 부분으로 구분한다. 다음은 HTML 코딩 부분으로 스크립트를 완전히 분리해 내었음을 알 수 있다. HTML 코딩 부분에서는 연결자(about-image)와 스크립트를 불러오는 구문만을 넣고 나머지 모든 기능

은 아래와 같이 JavaScript 파일인 changelImage.js로 분리하여 구현하였다.

O (Good)

```
<script type="text/JavaScript" src="changelImage.js"></script>
<p><a href="about.html"></a></p>
```

changelImage.js는 다음과 같다. JavaScript 부분에는 아무런 HTML 코드가 포함되어 있지 않음을 알 수 있다.

O (Good)

```
window.onload = function () {
    document.getElementById("about-image")
        .onmouseover= function changelImage() {
            this.src = this.src.replace(".gif", "_on.gif");
        }
}
```

JavaScript 코딩시에 간단한 기능을 수행하는 스크립트를 위의 예와 같이 분리하는 작업이 오히려 번거로울 수 있지만, 기능이 많아지고 복잡할수록 점차 장점이 늘어난다. 기능을 구별함으로써 각각의 코드들이 간단해지고 웹 콘텐츠의 크기도 줄어들게 된다. 그리고 HTML 자체적으로 모든 기능을 할 수 있기 때문에 자연히 접근성도 향상된다.

3. JavaScript 접근성 지원 프로그래밍 지침

이 지침에서는 JavaScript 프로그램을 포함하는 웹 콘텐츠가 접근성을 제공하기 위해서 JavaScript 프로그램이 지켜야할 코딩 방법을 제시한다. 제시하는 코딩 방법의 순서는 우리나라 국가 표준인 ‘인터넷 웹 콘텐츠 접근성 지침(Internet Web Contents Accessibility Guideline; KICS.OT-10.0003; 2005. 12. 21)’의 순서에 따르기로 한다. 이 지침에서는 인터넷 웹 콘텐츠 접근성 지침을 ‘국가표준’이라고 하며, 국가표준의 지침 순서에 의거하여 JavaScript 접근성 지원 프로그래밍 지침의 기술 순서를 정한다.

우선 접근성 준수 여부와 관계없이 웹 콘텐츠에 포함되는 JavaScript 코드는 JavaScript 문법을 준수하여야 한다. 뿐만 아니라 코드상의 오류나 실행시에 오류가 발생하지 않아야 한다. JavaScript에 관한 문법은 관련 도서를 참고하라.

이 절에서는 웹 콘텐츠가 접근성을 제공하기 위해서 JavaScript 프로그램이 지켜야할 코딩 방법을 제시한다.

3.1 (대체 텍스트 제공) ‘텍스트가 아닌 콘텐츠’에는 대체 텍스트를 제공하여야 한다.

국가표준 <항목 1.1>에 따르면, ‘텍스트가 아닌 콘텐츠’(예를 들어 이미지, 멀티미디어, 애니메이션 등)는 적절한 대체 텍스트를 제공하여야 화면 낭독 프로그램을 이용하는 장애인에게 그 내용을 전달할 수 있다. 이는 중요한 내용을 포함하고 있는 배경 이미지의 경우에도 마찬가지이다. 그러나 모든 ‘텍스트가 아닌 콘텐츠’에 대체 텍스트를 제공해야만 하는 것

은 아니다.

3.1.1 요구조건

- (1) 스크립트를 이용하여 웹 콘텐츠에서 '텍스트가 아닌 콘텐츠'를 스크립트를 이용하여 교체하는 경우에는 교체하는 콘텐츠의 대체 텍스트도 제공하여야 한다.

3.1.2 적용방법

가) 이미지 치환

JavaScript는 웹 페이지상의 이미지를 새로운 이미지로 치환할 수 있다. 이 과정에서 이미지만 치환한다면 이 이미지에 대한 대체 텍스트는 치환되기 이전의 대체 텍스트가 제공되므로 치환된 이미지에 적합한 대체 텍스트를 제공하지 못한다. 따라서 이미지를 치환할 경우에는 대체 텍스트도 함께 치환하도록 한다.

아래의 JavaScript 프로그램은 화면에 나타난 웃는 얼굴의 이미지 상에서 마우스를 클릭하면 기존의 이미지 `old_image.jpg`를 `new_image.jpg`로 치환하고, 대체 텍스트를 `new_image.jpg`에 적합한 내용 "우는 얼굴"로 치환하는 코딩의 한 예제이다.

O (Good)

```
<html>
  <head>
    <title> ... </title>
    <script type="text/JavaScript">
      var bool=0;
      function replaceImage() {
        if (bool == 0) {
          bool = 1;
          document.getElementById("image").src="new_image.jpg";
          document.getElementById("image").alt = "우는 얼굴";
        }
        else {
          bool = 0;
          document.getElementById("image").src="old_image.jpg";
          document.getElementById("image").alt = "웃는 얼굴";
        }
      }
    </script>
  </head>
  <body>
    <div style="margin-top:100px; text-align:center">
      <img id = "image" src=old_image.jpg" alt="웃는 얼굴"
        style="cursor:pointer" onclick="replaceImage(bool);" />
    </div>
  </body>
</html>
```



그림 II - 3. 웃는 얼굴(좌)을 클릭하면 우는 얼굴(우)로 변경되는 예제

나) 올바른 대체 텍스트

‘텍스트가 아닌 콘텐츠’와 함께 제공되는 대체 텍스트는 화면에 표시되는 이미지를 대신해서 보조 기술(예를 들어 화면 낭독 프로그램)에게 제공되는 정보이므로 그 내용이 적절하여야 한다.

바람직한 방법은 왜 이미지를 화면에 제공하느냐를 생각하여 대체 텍스트의 내용을 결정하는 것이 이미지의 모습을 설명하는 것보다 낫다. 예를 들어 투명한 플라스틱 병에 생수가 가득 들어있는 이미지에 대한 대체 텍스트를 ‘투명한 액체가 담긴 플라스틱 병’이라고 하기 보다는 ‘생수병’이라고 하는 것이 더 적합하다.

‘텍스트가 아닌 콘텐츠’ 요소에 대체 텍스트를 제공하는 것은 어려운 일이 아니다. 보다 중요한 점은 ‘언제, 어떤 내용의 대체 텍스트를 제공할 것인가’이다. 또한 대체 텍스트가 화면 낭독 프로그램 사용자에게 어떠한 영향을 주는가를 항상 고려하여야 한다.

3.2 (키보드의 이용) 키보드(또는 키보드 인터페이스)만으로도 웹 콘텐츠가 제공하는 모든 기능을 수행할 수 있어야 한다.

국가표준 <항목 2.4>에 따르면, 웹 콘텐츠는 키보드 또는 장애를 극복하도록 도와주는 여러 가지 입력 장치를 사용하는 경우에도 웹 콘텐츠가 제공하는 모든 기능을 사용할 수 있어야 한다. 예를 들어 마우스를 사용할 수 없는 장애인들도 마우스를 사용할 수 있는 사용자와 같이 키보드만으로 웹 콘텐츠가 제공하는 모든 기능을 동일하게 수행할 수 있어야 한다. 이 검사항목은 웹 브라우저의 이용 뿐 아니라 웹 콘텐츠가 제공하는 기능을 이용하는 경우에도 해당된다.

3.2.1 요구조건

- (1) 스크립트가 사용하는 이벤트 핸들러는 접근성을 보장하여야 한다.
- (2) 버튼에는 단축키를 제공하여 접근성을 높인다.

3.2.2 적용방법

가) 접근성 지원 이벤트 핸들러

모든 컨트롤은 마우스를 통해 작동될 뿐 아니라 키보드를 통해 접근할 수 있어야 한다. 이것은 화면 낭독 프로그램 사용자 뿐 아니라 마우스를 사용할 수 없는 사용자도 배려하기 위함이다.

JavaScript에서 접근성을 지원하는 마우스 이벤트와 이에 대응되는 키보

드 이벤트는 <표 II - 1>과 같다.

표 II - 1. 접근성 있는 마우스 이벤트

마우스 이벤트	대응되는 키보드 이벤트
onmouseover	onfocus ()
onmouseout	onblur ()
onmousedown	onkeydown ()
onmouseup	onkeyup ()
onclick	onkeypress ()

<표 II - 1>에서 onmouseover와 onmouseout 이벤트는 이미지 위에 마우스포인터를 위치시키거나 빠져나갈 때 발생한다. 그러나 이미지 롤오버(image rollover)에 키보드 이벤트 onfocus와 onblur 이벤트를 사용하더라도 화면 낭독 프로그램으로는 아무런 도움이 되지 못한다. 또한 CSS를 사용하여 롤오버 기능을 구현하는 경우에 CSS 이미지 치환 기술은 접근성을 지원하지 않는다. 그러나 텍스트의 치환시에는 문제가 없다.

일부 DHTML의 경우에 onkeypress 대신에 onkeydown과 onkeyup 이벤트를 사용할 수 있다. 예를 들어 화살표 키(arrow key)는 onkeypress 이벤트를 제대로 활성화 시키지 못한다. 그러나 onkeydown과 onkeyup 이벤트는 항상 활성화 된다. 여기서 노트북에는 화살표 키가 없으므로 화살표 키를 꼭 사용할 것인가를 생각해 보아야 한다. 화살표 키를 사용하기 위해서는 반드시 다른 키(기능 확장키)를 눌러야 하므로 이미 키보드 이벤트가 발생하기 때문이다. 뿐만 아니라 화면 낭독 프로그램이 이러한 형식의 인터페이스를 지원하는 것도 매우 어려운 일이다.

대부분의 브라우저는 Enter 키를 누를 때 onclick 이벤트를 활성화시킨다. 그러나 일부 브라우저는 이를 지원하지 않는다. 따라서 Enter 키를 지

원하기 위해서는 onclick 이벤트 보다는 onkeypress 이벤트를 사용하는 것이 좋다.

<표 II - 1>에 포함되지 않은 이벤트 핸들러를 사용할 경우에는 프로그램의 접근성 지원 여부를 세심하게 관찰하여야 한다. 예를 들어 onchange 이벤트는 입력 데이터의 형식을 수정하는 것과 같은 소소한 경우에는 사용할 수 있으나, Form(form)을 제출(submit)하기 위한 이벤트로는 사용하지 않는 것이 좋다.

나) 드래그 앤 드롭(Drag and Drop)

드래그 앤 드롭(drag-and-drop) 이벤트는 마우스를 사용하는 것을 전제로 제공한다. 그런데 전맹, 일부 저시력자, 지체장애인들은 마우스의 사용이 어렵다. 따라서 드래그 앤 드롭 기능은 키보드만으로도 사용할 수 있도록 하여야 한다. 예를 들어 온라인 쇼핑몰에서 쇼핑 카트에 상품을 드래그(drag)하는 과정을 Enter 키를 이용하도록 하는 것과 같다.

그 한 가지 방법은 링크를 이용하는 방법이다. 즉 상품 이미지에 링크를 제공하여 해당 상품 이미지에 대한 대체 텍스트를 읽어주는 동안 Enter 키를 치면 해당 상품이 쇼핑 카트에 담기도록 하는 것이다. 이것이 가능한 이유는 링크 컴포넌트가 키보드 접근을 지원하며, 이미지를 아이콘과 같이 표시할 수 있기 때문이다. 또한 이 링크에 추가적인 이벤트를 할당하여 이벤트가 발생하면 이 상품을 쇼핑 카트로 이동하게 한다.

본격적인 드래그 앤 드롭의 대체 수단은 이 문서의 부록에 수록한 WAI-ARIA 적용사례의 드래그 앤 드롭에 대한 설명을 참고하라.

다) 단축키 설정

다수의 컨트롤이 필요한 복잡한 애플리케이션 콘텐츠는 단축키를 이용하여 사용하도록 하는 것이 좋다. 일부 사용자에게는 하나의 키를 조작하는 것도 많은 노력을 필요로 한다. 따라서 가능한 한 이벤트를 발생시키기 위하여 누르는 키의 수를 줄일 수 있는 단축키를 제공하는 것이 바람직하다.

예를 들어 도움말 화면으로의 빠른 전환을 제공하기 위해 단축키로 '?' 키를 사용할 수 있다. 이것은 사용자들이 '?' 키를 누를 때마다 도움말 화면으로 이동시킨다. 이러한 기능을 수행하기 위해서는 HTML 코드는 다음과 같이 작성한다.

O (Good)

```
<input type="button" value="도움말"  
  onclick="window.open(http://./help.htm);"  
  onkeypress="verifyKey(this,event);"  
>
```

아래의 JavaScript 코드는 단축키를 '?'로 정의하고, 키보드에서 '?'를 누르면 도움말 버튼을 누른 것과 같은 효과를 제공한다.

O (Good)

```
// With onkeypress event, this verifies "?" key  
function verifyKey(oElement,oEvent){  
  if(oEvent.keyCode==63 && oElement.onclick){ // '?' = 63  
    oElement.onclick();  
  }  
}
```

3.3 (반응시간의 조절기능) 실시간 이벤트나 정해진 시간 내에 응답이 필요한 기능은 사용자가 시간에 구애받지 않고 읽거나 상호작용을 하거나 응답할 수 있어야 한다.

국가표준 <항목 2.6>에 따르면 자동적으로 갱신되도록 구성된 콘텐츠, 일정 시간이 경과하면 다른 페이지로 이동하도록 구성된 콘텐츠, 깜빡이는 텍스트나 스스로 스크롤 하도록 구성된 텍스트, 짧은 기간 동안 나타났다 일정시간 후에 사라지는 대화창, 일정시간 동안 사용하지 않으면 페이지에 대한 접근이 강제 차단되거나 사용할 수 없게 되는 콘텐츠 등과 같이 정해진 시간 내에 응답이 필요한 콘텐츠들은 사용하지 않아야 하며, 꼭 사용해야 할 경우에는 사용자가 그 기능을 조작할 수 있도록 해야 한다.

3.3.1 요구조건

- (1) 팝업(Pop-Up) 창을 열 때에는 사용자에게 팝업창의 열림을 공지하여야 한다.
- (2) 사용자가 의도하지 않은 브라우저 창의 크기 변경, 위치 이동 등은 발생하지 않아야 한다.

3.3.2 적용방법

가) 팝업창 열기

사용자가 요구하지 않은 팝업창을 열도록 허용해서는 안된다. 뿐만 아니라 팝업창을 열고자 하는 경우에도 사용자에게 이 사실을 알려주어야 한다. 팝업창의 열림을 알려주는 방법은 링크 요소의 직전에 알려주기, 또는 링크 텍스트를 이용하여 알려주기, 그리고 링크의 title 속성을 이용하여 알려주는 방법이 있다.

팝업창 또는 새 창 열기를 한 후에는 팝업창 또는 새 창이 정상적으로 열렸는지를 확인하여야 한다. 브라우저에서 새 창 열기를 차단한 경우에는 팝업창이 열리지 않는다. 따라서 팝업 창이 열리지 않아 아무런 콘텐츠가 표시되지 않았음에도 불구하고 팝업창이 열렸다고 알려준다면 화면 낭독 프로그램의 사용자에게 큰 혼란을 주게 된다.

아래의 코드는 onclick 이벤트 핸들러를 이용하여 팝업창을 열도록 구성한 프로그램이다. 만일 브라우저가 새 창 열기를 허용하지 않으면 새 창이 열리지 않는다.

X (Bad)

```
<!-- This is NOT OK with popup blocking setting -->  
href="example.htm" onclick="window.open (...); return false;"
```

새 창 열기가 차단된 브라우저에서 새 창 열기를 하는 경우에 새 창을 열지 않고 이미 열려있는 브라우저 화면을 이용하도록 하려면 다음과 같이 프로그램을 수정하여야 한다.

O (Good)

```
<!-- This is OK with popup blocking setting -->  
href="example.htm" onclick="return pop(this);"
```

여기서 pop() 함수는 새 창이 열리면 false 값을, 새 창이 열리지 않으면 true 값을 리턴하고, pop() 내부에서 새 창이 열리지 않으면 기존의

웹 브라우저 창에 콘텐츠를 표시하도록 코딩할 수 있다.

나) 하이퍼텍스트 레퍼런스의 제공

링크를 명시하는 것이 더 큰 효과를 보이는 부분은 팝업창이다. 여기서 말하는 팝업창은 전체 창이 새로 뜨는 것이 아니라 작은 창을 띄우는 것을 말한다. 팝업창을 띄우려면 JavaScript의 window.open 메소드를 사용한다.

아래의 예제는 이미지에 onclick 이벤트를 사용하여 작은 창을 열고 popup.html 페이지를 표시하는 것이다. 하지만 이 방법은 마우스 포인터가 바뀔다거나 키보드 초점이 제공되지 않기 때문에 사용자 입장에서 사용하기가 어렵다.

X (Bad)

```

```

키보드 초점을 제공하고 마우스 포인터가 바뀌는 등 보통의 링크와 가장 유사한 방법을 구현하기 위해서는 아래와 같이 프로그램을 작성하여야 한다.

X (Bad)

```
<a href="#">
  
</a>
```

그러나 위의 방법도 접근성에는 한계가 있다. 그 이유는 팝업창과 함께 열려야 하는 웹 페이지 주소가 이벤트 핸들러 안의 JavaScript 구문에 있

기 때문이다. HTML 상의 링크에는 페이지에 대한 정보가 제공되지 않고 웹 페이지 조각을 나타내는 #으로만 되어 있다. 그런데 접근성을 향상시키기 위해서는 HTML 상의 하이퍼텍스트 레퍼런스 속성에 팝업창의 주소를 제공하여야 한다. 또한 이벤트 핸들러 안의 JavaScript 구문의 팝업창 주소를 this.href로 표시한다. 이렇게 코딩하면 HTML 부분에 웹 페이지 경로에 대한 정보('popup.html')가 있기 때문에 JavaScript에 관계없이 웹 페이지의 링크가 제공되므로 접근성이 향상된다. 뿐만 아니라 브라우저에서 지원하는 새 창 열기나 북마크 기능 등에서도 HTML에 들어 있는 정보를 활용할 수 있다. 아래의 예제에서 "popup.html"이 HTML 부분에 나타나 있음을 알 수 있다.

O (Good)

```
<a href="popup.html"
  onclick="window.open(this.href,'popupName',
    'width=300,height=200'); return false;">
  
</a>
```

다) 링크의 사용

링크는 앵커(anchor) 요소를 사용하여 문서를 서로 연결시키는 역할을 한다. 사용법은 앵커 태그 <a>에 href 속성을 이용하여 연결할 링크의 URI를 표시한다. 이 때 하이퍼텍스트 레퍼런스 속성은 반드시 유효한 URI 값을 가지고 있어야 한다. 그러나 종종 JavaScript: myFunction()과 같이 URI가 아닌 다른 값을 사용하는 경우를 볼 수 있다. 'JavaScript:'로 시작하는 방법은 JavaScript를 바로 실행할 수 있는 장점이 있지만 HTML 문서 내에서 사용하는 href 값으로는 적절하지 않다. 이 방법은 폐기된 방법이기 때문에 정상적인 동작을 보장하지 못한다.

href 속성에 유효한 값이 아닌 다른 값이 들어갈 경우, 이는 올바른 링크가 아니기 때문에 여러 가지 혼란을 가져올 수 있다. 많은 브라우저에서 이러한 링크를 허용하고 있지만 일부 브라우저에서는 올바르게 동작하지 않는다. 예를 들어 href 값이 유효하지 않으면 마우스 오른쪽 클릭을 했을 때 새 창 열거나 북마크 기능 등을 사용할 수 없다.

접근성을 높이기 위해서는 href 값으로 유효한 값을 제공하고, 이벤트 핸들러를 이용해서 링크로 이동하거나 새 창 열기를 하는 등의 기능을 구현한다. 유효한 링크인지를 판단하려면 HTML 레벨에서의 적절한 동작이 무엇인지를 생각해 보아야 한다. 예를 들어 탭으로 여러 콘텐츠를 보여주는 기능을 구현하는 경우에, 탭은 지정된 콘텐츠 블록과 연결되어 있어야 하며, 탭을 클릭하는 것은 탭에 연결된 콘텐츠 블록을 확인하고자 하는 의도를 내포하고 있다. 따라서 탭의 링크는 콘텐츠의 위치를 나타낼 수 있어야 한다.

아래 그림은 '공지사항'과 '정보통신 접근성 뉴스'의 두 개 링크 탭에 이어서 '공지사항'에 해당하는 리스트와 '정보통신 접근성 뉴스'에 해당하는 리스트, 그리고 그 사이에 전체 리스트로 이동할 수 있는 'more' 버튼으로 이루어진 웹 콘텐츠이다.

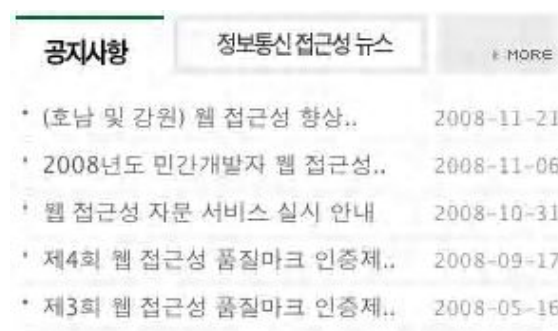


그림 II - 4. 두 개의 링크 탭과 전체리스트로 이동할 수 있는 버튼으로 구성된 예제

JavaScript를 사용하면 탭을 선택함에 따라 화면에 콘텐츠 블록을 바꾸어 표시할 수 있으므로 탭의 기능이 유효하다.

위의 예제에서 CSS를 제거하면 아래와 같이 표시된다. 따라서 JavaScript 기능을 비활성화 시키는 경우에는 각 탭이 어떤 콘텐츠 블록을 지시하는지 불분명해진다. 링크 속성이 정확히 표시되어 있다면 각각의 탭이 어느 리스트를 나타내는지 좀 더 명확히 알 수 있으므로 접근성이 향상된다.



그림 II - 5. <그림 II - 4>에서 CSS를 제거한 경우

3.4 (온라인 서식 구성) 온라인 서식은 작성에 필요한 정보, 서식 구성 요소, 필요한 기능, 작성 후 제출 과정 등 서식과 관련한 모든 정보를 제공해야 한다.

3.4.1 요구조건

- (1) 웹 페이지를 구성하는 온라인 서식 요소는 논리적이어야 한다.
- (2) 온라인 서식에서 Tab 키에 의한 이동 순서는 논리적이어야 한다.
- (3) 온라인 서식은 JavaScript 기능을 지원하지 않는 브라우저에서도 submit이 가능하여야 한다.

3.4.2 적용방법

온라인 서식은 사용자가 입력한 값을 서버에 전달하는 역할을 하는 웹 콘텐츠이다. 온라인 서식은 <form> 태그를 이용하여 구성하며, 서버로 전달하기 위해서는 <form>의 submit 기능을 이용한다.

국가표준 <항목 3.3>에 따르면, 온라인 서식을 구성하는 모든 서식 제어 요소(예를 들면 편집 상자(edit box), 라디오 버튼(radio button), 체크박스(check box)등의 레이블과 해당 서식 제어 요소)간의 표시 순서가 논리적이어야 한다. 또한 온라인 서식은 키보드로만 사용이 가능하여야 하며, Tab 키를 이용하여 서식 제어 요소 간을 이동할 경우에도 그 순서가 논리적이어야 한다. Shift + Tab 키는 Tab 키에 의한 이동 순서의 반대이다.

뿐만 아니라 온라인 서식은 JavaScript를 지원하지 않는 경우에도 submit이 가능하도록 구성하여야 한다. 그렇지 않으면 온라인 서식을 작성하였음에도 불구하고 스크립트를 지원하지 않는 환경 때문에 그동안 입력한 결과를 submit할 수 없어서 수포로 돌아갈 수 있기 때문이다.

가) 로그인



그림 II - 6. 로그인 예제

<input> 태그를 이용한 온라인 서식은 자체적으로 값을 보내는 submit 기능을 제공하고 있다. <input type="submit" />이나 <input type="image" />이 submit에 필요한 컨트롤이다. 그런데 Form에서 입력한 값의 유효성 여부를 동적으로 검사하기 위하여 스크립트로 코딩하게 되는데, 이 과정에서 Form의 submit 기능을 이용하지 않고 스크립트로 submit 기능을 대신하도록 코딩하는 경우가 있다. 아래의 코드는 스크립트를 이용한 submit 코드의 예제이다.

X (Bad)

```
<script type="text/javascript">
function submitForm() {
    loginForm.submit();
}
</script>
```

```

<form id="loginForm" name="loginForm" action="">
  User Id <input type="text" name="loginId">
  User Password <input type="password" name="loginPassword"><br>
  
</form>

```

이 경우에 Form에 submit 기능이 없기 때문에 JavaScript를 지원하지 못하는 일부 브라우저의 경우에는 submit이 불가능한 경우가 발생한다. 뿐만 아니라 submit 대신에 이미지가 들어가 있기 때문에 의미적으로도 맞지 않는다. 따라서 Form을 제작할 때에는 반드시 submit 기능을 <input> 태그로 구성하여야 한다. 아래의 코드는 <input>을 이용하여 재구성한 submit 방법의 예제이다.

O (Good)

```

<form id="loginForm" name="loginForm" action="">
  <p>
    <label for="loginId">User Id</label>
    <input type="text" id="loginId" name="loginId" /><br />
    <label for="loginPassword">User Password</label>
    <input type="password" id="loginPassword"
      name="loginPassword" />
  </p>
  <p>
    <input type="image" src="login.gif" alt="로그인" />
  </p>
</form>

```


나) 유효성 검증

클라이언트 측에서 JavaScript를 이용하여 유효성을 검증하는 기능은 <form>의 submit 이벤트를 캐치하는 방식으로 구현해야 한다. 만일 JavaScript가 Form을 submit 하도록 하면, <가) 로그인>에서 이야기 한 것과 같이 브라우저에 따라서 submit이 불가능한 경우가 발생한다.

아래의 코드는 JavaScript를 이용하여 Form의 유효성을 검증하는 프로그램으로 HTML만으로는 기능이 작동하지 않는다.

O (Good)

```
<script type="text/javascript">
function submitForm(formEl) {
    var errorMessage = null;
    var objFocus = null;

    if (formEl.loginId.value.length == 0) {
        errorMessage = "아이디를 넣어주세요.";
        objFocus = formEl.loginId;
    } else if (formEl.loginPassword.value.length == 0) {
        errorMessage = "비밀번호를 넣어주세요.";
        objFocus = formEl.loginPassword;
    }
    if(errorMessage != null) {
        alert(errorMessage);
        objFocus.focus();
        return false;
    }
    return true;
}
```

```

</script>

<form id="loginForm" name="loginForm" action=""
      onsubmit="return submitForm(this)">
    <label for="loginId">아이디</label>
    <input type="text" id="loginId" name="loginId" />
    <label for="loginPassword">비밀번호</label>
    <input type="password" id="loginPassword"
           name="loginPassword" /><br />
    <input type="image" src="login.gif" alt="로그인" />
  </form>

```

위의 프로그램에서는 onsubmit 이벤트를 이용해서 입력 필드의 유효성을 체크하고, 유효성 검사 결과를 true나 false로 리턴하는 예를 보여주고 있다. 따라서 입력 필드에 에러가 있을 경우에는 submit을 멈추고 에러가 발생한 입력 창에 초점이 주어지게 된다.

다) 게시판 기능






번호	제 목	파일	작성자	작성일	조회
10	공지사항		관리자	2008-09-23	731
9	공지사항		관리자	2007-05-15	1214
8	공지사항		관리자	2006-11-09	1546
7	공지사항		관리자	2006-02-06	1971
6	공지사항		관리자	2004-10-29	2384
5	공지사항		관리자	2004-08-10	2522
4	공지사항		관리자	2004-06-14	2745
3	공지사항		관리자	2004-03-09	4022
2	공지사항		관리자	2004-03-09	3437
1	공지사항		관리자	2003-10-21	2619

그림 II - 7. 게시판 기능 예제

게시판에서 JavaScript만으로 페이지 간을 이동하는 것을 종종 볼 수 있다. 아래의 프로그램은 사용자 input이 없는 빈 Form을 하나 만들고 JavaScript를 이용해서 내용을 보거나 페이지를 이동하도록 구현한 예제이다.

X (Bad)

```
<form method="post" name="vars">
  <input type="hidden" name="articleId" value="23" />
  <input type="hidden" name="page" value="3" />
  <input type="hidden" name="keysord" value="" />
  <input type="hidden" name="searchType" value="" />
  <!-- 등등 -->
</form>
.
.
<a href="javascript:ArticleRead()">글읽기</a>
<a href="javascript:GoList()">리스트 보기</a>
```

이 방법은 URL이 간단하고 다루기 쉽기 때문에 개발자들이 선호하는 경향이 있으나 절대로 사용해서는 안 된다. 왜냐하면 <그림 II - 7>의 프로그램과 같이 필요한 기능을 JavaScript로 구현하면, 스크립트 오류가 발생하거나 JavaScript가 정상적으로 작동하지 않을 경우 해당 기능에 접근할 수 없기 때문이다. 또한 모든 변수를 post로 전달하기 때문에 주소창에 URL이 표시되지 않아 해당 페이지를 따로 북마크하거나 저장할 수가 없다. 이 때문에 해당 게시물로의 접근이 원천적으로 차단되는 현상이 발생한다. 따라서 웹 콘텐츠를 <a> 태그와 URL만으로도 수행이 가능한 웹 페이지로 만들고, 필요시에 QueryString을 이용하여 필요한 정보를 검색하도록 하는 것이 가장 좋다.

3.5 (신기술의 사용) 스크립트, 애플릿 또는 플러그 인 (plug-in) 등과 같은 프로그래밍 요소들은 현재의 보조 기술의 수준에서 이들 프로그래밍 요소들의 내용을 사용자에게 전달해줄 수 있을 경우에만 사용하여야 한다.

국가표준 <항목 4.1>에 따르면, 보조 기술을 지원하지 않으면 스크립트 등의 프로그램 요소를 사용하지 않아야 한다. 만일 JavaScript 기능을 지원하지 않는 웹사이트는 JavaScript 기능을 정지시켰을 때에도 사용자에게는 필요한 정보가 차단되지 않아야 한다.

3.5.1 요구조건

- (1) JavaScript 프로그램은 보조 기술을 지원하도록 코딩하여야 한다. 만일 사용하는 스크립트 코드가 보조 기술을 지원하지 않을 경우에는 대체 방법을 제공하여야 한다.

3.5.2 적용방법

가) JavaScript가 아닌 대체 방법의 제공

JavaScript가 아닌 대체 방법을 제공하는 것은 noscript를 사용하라는 것이 아니다. 대개의 경우에는 noscript를 사용하지 않고도 접근성을 지원할 수 있는 방법들이 존재한다. 예를 들어 웹 콘텐츠의 내용을 화면에 표시하는 것은 HTML과 CSS를 이용하여 구현하도록 하고, 웹 페이지에서 상호작용이 필요한 경우에만 JavaScript를 사용하면 noscript를 사용하지 않고도 접근 가능한 웹 콘텐츠를 구현할 수 있다. noscript를 사용하는 경우는 더 이상 다른 방법이 없을 경우에만 사용하는 것이 좋다.

3.6 표준의 준수

국가표준에는 포함되어 있지 않으나 웹 콘텐츠를 개발하기 위해서는 기본적으로 표준에 따른 HTML 및 CSS 문법을 적용한다. 또한 JavaScript 표준 문법도 준수하여야 한다. 문법에 맞지 않는 스크립트가 웹 콘텐츠에 포함되어 있으면 브라우저 간의 호환성이 떨어질 뿐 아니라 접근성을 지원하지도 못한다. 스크립트에 버그가 있을 경우에도 브라우저에서 실행되지 않는 경우가 많으므로 세밀한 디버깅을 할 필요가 있다.

3.6.1 요구조건

(1) JavaScript 프로그램은 문법에 맞도록 코딩하여야 한다.

3.6.2 적용방법

가) JavaScript의 선언

JavaScript는 HTML의 스크립트 태그를 사용하여 문서에 삽입된다. 스크립트 태그는 문서가 로딩될 때 작동한다. 스크립트가 사용할 수 있는 속성은 charset, type, src, defer 등이다. 이 중에서 type 속성은 필수 속성이기 때문에 반드시 선언이 되어야 한다. 아래 예제는 type 속성을 이용하여 스크립트를 선언하고 있다.

O (Good)

```
<script type="text/JavaScript">  
  //code  
</script>
```

여기서 language 속성은 폐기된(deprecated) 속성이기 때문에 language 속성을 이용하여 스크립트를 선언하는 것은 잘못된 방법이다.

스크립트 요소는 <head> ... </head> 또는 <body> ... </body> 내에만 존재하여야 한다. 스크립트 요소를 <html> 태그 앞이나 </head>와 <body> 사이 등, 잘못된 곳에 포함시키면 정상적인 동작을 보장할 수 없다.

HTML 문서의 마지막에 스크립트를 삽입할 경우, 아래의 예와 같이 HTML을 닫는 태그 </html> 뒷부분에 삽입하지 않아야 하며, </body> 태그 앞부분에 삽입해야 한다.

O (Good)

```
...  
...  
...  
<script type="text/JavaScript">  
    //code  
</script>  
</body>  
</html>  
<!-- 이곳에는 스크립트 요소가 위치할 수 없다. -->
```

나) 페이지의 이동

웹사이트를 이용하다 보면 Form(form)에 값을 입력하고 submit 하는 순간 클릭을 여러 차례 한 것과 같이 "따다다닥" 하는 소리가 나는 경우가 있다. 그 이유는 submit 기능을 JavaScript로 수행하기 때문이다.

X (Bad)

```
<script type="text/JavaScript">
    document.location.href ="redirection.html";
</script>
```

이와 같은 현상은 아래와 같이 <form>을 이용해서 submit 하는 경우에도 발생한다.

X (Bad)

```
<form name="login_form">
    <input type="hidden" name="user_id" value="myid" />
    <input type="hidden" name="user_pwd" value="mypassword" />
    <input type="hidden" name="redirect_url" value=
        "http://mysite.com/login/" />
    <input type="hidden" name="somevalue" value="blahblah" />
    ...
</form>
<script type="text/JavaScript">
    f = document.forms.login_form;
    f.action = "http://login.oursite.com/login/";
    ...
    f.submit();
</script>
```

심한 경우 아래와 같이 전혀 의미 없는 Form을 이용하기도 한다.

X (Bad)

```
<form method="post" name="sg_form"
    action="http://www.qubi.com" target="_top"></form>
<script> sg_form.submit(); </script>
```

이상의 프로그램 예제는 분명한 HTML 문법상 에러이며, 이로 인하여 필요한 기능이 작동하지 못하는 경우도 발생한다. <form> 태그나 <script> 태그는 상위의 <body>나 <head> 태그가 있어야 하는데 위의 예에서는 이러한 태그가 없으므로 HTML 문장으로 해석이 안 되기 때문에 스크립트가 작동하지 않거나 submit이 불가능하다. 그리고 <form> 태그에는 submit을 위한 <input> 태그가 존재하지 않으므로 JavaScript로 submit을 할 수 없는 경우도 있다.

이와 같이 페이지를 이동하거나 submit이 필요할 때에 JavaScript를 이용하게 되면 클라이언트의 환경에 따라서 동작이 실패할 수 있다. 따라서 이러한 처리는 JavaScript 보다는 서버 측에서 http 헤더 정보를 이용해서 처리하는 것이 좋다.

이러한 중간과정에서의 처리는 서버 측에서 모두 처리하는 것이 가장 바람직 하지만 어쩔 수 없이 사용을 해야 할 경우에는 DTD 선언이나 <html> 태그, <head>, <body>와 같이 필수 태그들이 포함된 완전한 페이지로 구성한다. 또한 JavaScript가 작동하지 않는 경우를 위해서 <form>에 submit 버튼도 제공한다. submit 결과에 대한 메시지도 alert외에 일반 text와 <a>를 이용한 링크를 제공하도록 해야 한다.

O (Good)

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=euc-kr" />
<title>Redirect</title>
</head>
<body>
<script type="text/JavaScript">
/*
  some processes...
*/
```



```
alert(' ... 의 이유로 다시 돌아 갑니다. ');  
document.location.href="redirection.html";  
</script>  
  <p><a href="redirection.html"> ... 의 이유로 다시 돌아 갑니다.</a></p>  
</body>  
</html>
```

charset을 선언하지 않으면 브라우저는 초기 설정에 의하여 alert 기능이 작동한다. 또한 브라우저의 초기 설정을 ko-kr로 해야 한글이 깨지지 않는다. 문서의 mime-type도 text/html로 설정한다.

4. JavaScript 프로그램의 접근성 평가방법

JavaScript 프로그램은 HTML, CSS 등과 함께 웹 페이지를 구성한다. 따라서 웹 콘텐츠에서 JavaScript 프로그램만 분리하여 접근성을 평가할 수는 없다. 아래의 체크리스트는 JavaScript 코드가 포함된 웹 콘텐츠의 접근 가능여부를 평가하기 위하여 확인해야 하는 목록을 요약한 것이다.

일반적인 웹 콘텐츠의 평가방법에 대해서는 <I - 4. RIA 콘텐츠 접근성 평가방법>을 참고하라.

번호	항목	검사항목	국가 표준
1	대체 텍스트	(1) 스크립트로 교체하는 이미지는 교체되는 이미지에 알맞게 대체 텍스트도 교체하고 있는가?	1.1
2	키보드	(2) 키보드만으로 콘텐츠가 제공하는 모든 기능에 대한 제어가 가능한가? (3) 키보드로 Form 컨트롤, 링크, 버튼 등 객체 사이를 이동할 때 논리적 이동 순서가 적절한가? (4) 하위 메뉴를 제공하는 주 메뉴의 경우, 키보드만으로도 하위 메뉴를 이용할 수 있는가?	2.4
3	반응 시간	(5) 열리지 않는 팝업창을 ‘새 창열기’로 알려주지 않는가? (6) 사용자의 입력 없이 창의 크기를 변경하거나 이동시키지 않는가? (7) 사용자의 입력 없이 자동으로 발생하는 팝업창을 사용하는 경우는 없는가? (8) 사용자의 입력에 의해서 발생하는 팝업창은 미리 팝업창임을 알리고 있는가?	2.6

4	온라인 서식	(9) 온라인 서식의 구성이 논리적인가? (10) Tab 키에 의한 이동 순서가 논리적인가? (11) 키보드만으로 온라인 서식의 모든 기능을 이용할 수 있는가? (12) JavaScript를 지원하지 않더라도 submit 기능이 동작하는가?	3.3
5	신기술	(13) JavaScript 코드를 지원하지 않는 경우에 대체 콘텐츠를 제공하는가? (14) 대체 콘텐츠는 간단명료하면서 원본과 동일한 기능을 수행하고 있는가?	4.1
6	표준의 준수	(15) 사용하는 JavaScript코드는 표준을 준수하고 있는가?	

4.1 대체 텍스트 제공

4.1.1 체크리스트

(1) 스크립트로 교체하는 이미지는 아래와 같이 교체되는 이미지에 알맞게 대체 텍스트도 교체하고 있는가?

- 이미지, 멀티미디어 콘텐츠, 그래픽 아이콘, 그래픽 글자 등은 비록 텍스트의 형태를 하고 있더라도 코드로 사용될 수 없다. 이들은 보조 기술에 제공되더라도 그 내용을 알 수 없다. 따라서 보조 기술이 코드로 인식할 수 없는 모든 객체를 '텍스트가 아닌 콘텐츠'라고 정의할 수 있다. '텍스트가 아닌 콘텐츠'는 보조 기술이 인지할 수 있는 대체 텍스트를 추가로 제공하여야 한다.
- 만일 스크립트로 교체하는 '텍스트가 아닌 콘텐츠'가 있다면, 교체되는 '텍스트가 아닌 콘텐츠'에 대한 대체 텍스트도 교체해주었는지를 확인한다.
- ~링크, ~버튼, ~로고, ~바로가기, ~메뉴와 같이 중복될 수 있는 내용을 대체 텍스트로 제공하지 않는다. 화면 낭독 프로그램은 HTML 요소를 만나면 해당 요소의 용도를 알려준다. 예를 들어 어떤 그래픽 버튼을 W3C 웹사이트에 링크시켜 놓고, 대체 텍스트를 'W3C 바로가기 버튼'으로 설정하였다고 가정하면, 이 그래픽 버튼을 누를 때 마다 화면 낭독 프로그램은 "버튼, W3C 바로가기 버튼"이라고 읽어준다. 여기서 '버튼'을 두 번 읽지 않도록 하려면, 대체 텍스트를 'W3C 바로가기'나 'W3C 웹 사이트'로 설정하면 된다. 이것은 메뉴나 이미지 링크, 로고의 경우에도 해당된다.

- 대체 텍스트는 콘텐츠의 내용을 파악할 수 있는 핵심적인 설명을 제공한다. 대체 텍스트는 콘텐츠를 직관적으로 나타낼 수 있는 내용으로 구성하는 것이 바람직하다. <II - 3.1.2 나) 올바른 대체 텍스트>에서 예를 든 것처럼, 투명한 플라스틱 병에 생수가 가득 들어있는 이미지에 대한 대체 텍스트로는 ‘투명한 액체가 들어있는 플라스틱 병’이라고 하기보다는 ‘생수병’이라고 하는 것이 바람직하다.
- 대체 텍스트가 필요하지 않은 경우에는 alt=""로 제공한다. 때로는 대체 텍스트를 추가함으로 인하여 혼란을 야기시키는 경우가 있다. 예를 들어 글머리표는 “글머리표”라고 읽어주는 것이 도리어 혼란을 야기할 수 있다. 이런 경우에는 대체 텍스트를 null 값 (alt="")으로 제공한다. 이것은 기능적으로 보아 alt 속성을 생략하는 것과 같으나, alt 속성을 생략하면 화면 낭독 프로그램에 따라서 이미지의 파일 이름을 읽어주는 경우도 있으므로 alt=""이라는 대체 텍스트를 추가하는 것이 좋다.
- 교체되는 이미지의 대체 텍스트가 교체 전과 동일하면 대체 텍스트를 교체할 필요가 없다. 교체하는 이미지의 대체 텍스트가 교체하기 전과 동일할 경우에는 대체 텍스트를 교체할 필요가 없다. 스크립트를 이용하여 콘텐츠를 동적으로 교체하는 것은 자원 낭비이므로 가능한 한 대체 텍스트를 교체하지 않는 것이 좋다. 그러나 이미지의 내용이 교체 전과 비교하여 확연한 차이를 보일 경우에는 대체 텍스트를 교체하는 것이 당연하다.

4.1.2 관련 국가 표준 : 항목 1.1

4.1.3 JavaScript 프로그래밍 지침 : 항목 3.1

4.2 키보드의 이용

4.2.1 체크리스트

(2) 키보드만으로 콘텐츠가 제공하는 모든 기능에 대한 제어가 가능한가?

- 웹 콘텐츠는 키보드로만 사용이 가능하여야 한다. 또한 화면 낭독 프로그램과 같은 보조 기술과의 단축키 충돌이 없어야 한다.
- 특히 Tab 키와 Shift + Tab 키에 의한 초점 이동이 원활하여야 한다. 이 과정에서 가능한 한 불필요한 요소로 초점이 이동하지 않도록 함으로써 사용성을 높이는 것이 좋다. <II - 5.1.2 가로 2단 메뉴의 예제>에서 보여 주듯이, Shift +Tab 키를 누르면 하위 메뉴를 역순으로 이동하여 주 메뉴로 초점이 이동한 후, 주 메뉴 간을 역순으로 이동하는 것이 바람직하다.

(3) 키보드로 Form 컨트롤, 링크, 버튼 등 객체 사이를 이동할 때 논리적 이동 순서가 적절한가?

- 객체간의 이동은 Tab 키와 Shift + Tab를 이용하여 순방향 및 역방향 이동이 가능하다. 이때 이동하는 순서는 화면에 보인 객체의 순서와 동일하게 구성하는 것이 바람직하다. CSS를 사용하여 순서를 변경하더라도 초점이 이동하는 순서는 논리적으로 적절하여야 한다. 그 예로 <II - 5.2.2 회원가입>에서 '우편번호 검색' 버튼이 우편번호 입력란보다 앞서서 읽혀지도록 한 것을 보면 알 수 있다.

(4) 하위 메뉴를 제공하는 주 메뉴의 경우, 키보드만으로도 하위 메뉴를 이용할 수 있는가?

- 주 메뉴에서 하위 메뉴로 이동하는 것은 화면 낭독 프로그램별로 그 방법이 상이하다. 그러나 스크립트로 프로그램을 작성할 때에 이러한 기능이 정상적으로 동작하는지를 확인하여야 한다. 때에 따라서는 Tab 키나 Shift + Tab 키로 이동할 때에 하위 메뉴의 마지막에서 예상하지 못한 동작을 할 경우가 있다.
- 때때로 사용하는 브라우저를 달리하면 키보드 내비게이션에 의한 초점 이동결과가 달라지는 경우가 발생한다. 이것은 브라우저마다 스크립트의 사용방법이 다르기 때문이다. 크로스 브라우징에 관한 자료는 관련 사이트²²⁾를 참고하라

4.2.2 관련 국가 표준 : 항목 2.4

4.2.3 JavaScript 프로그래밍 지침 : 항목 3.2

22) <http://www.mozilla.or.kr/ko/docs/web-developer/standard/>

4.3 반응시간의 조절

4.3.1 체크리스트

(5) 열리지 않는 팝업창을 '새 창 열기'로 알려주지 않는가?

- 스크립트로 새 창 열기를 할 경우에 새 창이 열리지도 않으면서 새 창을 열었다고 알려준다면, 화면 낭독 프로그램 사용자는 열리지도 않은 창을 찾기 위하여 애를 쓰게 될 것이다. 따라서 새 창이 열린다는 내용을 읽어주었으면 반드시 새 창이 열리도록 스크립트를 작성하여야 한다. 이와 관련한 예는 <II - 3.3.2 가) 팝업창 열기>를 참조하라.

(6) 사용자의 입력 없이 창의 크기를 변경하거나 이동시키지 않는가?

- 스크립트가 창의 크기를 임의로 변화하게 한다면, 화면 확대 프로그램을 사용하는 사용자들에게 매우 큰 혼란을 주게 될 것이다. 예를 들어 웹 페이지의 어떤 부위를 화면 확대 프로그램으로 살펴보고 있는데 갑자기 화면이 축소되어 살펴보고 있는 부분이 시야에서 사라진다면, 사용자의 입장에서는 매우 당황스러울 것이다. 따라서 창의 크기는 사용자가 의도적으로 변경하지 않는 한, 그 크기를 일정하게 유지하도록 프로그래밍 하여야 한다.

(7) 사용자의 입력 없이 자동으로 발생하는 팝업창을 사용하는 경우는 없는가?

- (6)과 마찬가지로 스크립트가 임의로 팝업창을 열거나 닫는다면,

화면 확대 프로그램 사용자들을 매우 당황하게 할 뿐 아니라 화면 낭독 프로그램 사용자들도 초점의 갑작스러운 이동으로 혼란에 빠질 것이다. 따라서 웹 콘텐츠는 사용자가 의도하지 않은 팝업창의 열림과 닫힘을 허용하지 않아야 한다.

(8) 사용자의 입력에 의해서 발생하는 팝업창은 미리 팝업창임을 알리고 있는가?

- 어떤 버튼을 동작시키면 새 창이 열리는 웹 콘텐츠의 경우에, 새 창을 무조건 열기보다는 새 창이 열린다는 것을 사용자에게 알린 다음, 열리게 하는 것이 좋다. 예를 들어 대화 상자(Dialog Box)를 이용하여 ‘새 창을 열 것인가?’를 사용자에게 결정하도록 기회를 제공하는 것도 좋은 코딩방법의 하나이다.

4.3.2 관련 국가 표준 : 항목 2.6

4.3.3 JavaScript 프로그래밍 지침 : 항목 3.3

4.4 온라인 서식 구성

4.4.1 체크리스트

(9) 온라인 서식의 구성이 논리적인가?

- 온라인 서식은 사용자와의 상호작용이 필요한 가장 중요한 형식의 콘텐츠이다. 그런데 사용자와의 상호작용을 위해서는 사용자가 서식을 작성해야 하는데, 이 때 서식의 작성 요령이 제공되지 않

거나 제공되는 내용을 듣기 위해서 서식의 작성이 완료되어야 한다면 매우 불편할 것이다. 따라서 서식 작성요령, 레이블 등은 반드시 입력창 앞에 위치해야 한다. 아래 <그림 II - 8>은 레이블이 각각 입력창 앞에 위치한 것을 보여주는 예이다.



그림 II - 8. 레이블과 입력 창의 연결 순서

(10) Tab 키에 의한 이동 순서가 논리적인가?

- 객체간의 이동은 Tab 키와 Shift + Tab를 이용하여 순방향 및 역방향 이동이 가능하다. 이때 이동하는 순서는 화면에 보인 온라인 서식을 구성하는 객체의 순서와 동일하게 구성하는 것이 바람직하다. CSS를 사용하여 순서를 변경하더라도 초점이 이동하는 순서는 논리적으로 적절하여야 한다. 그 예로 <II - 5.2.2 회원가입>에서 '우편번호 검색' 버튼이 우편번호 입력란보다 앞서서 읽혀지도록 한 것을 보면 알 수 있다.

(11) 키보드만으로 온라인 서식의 모든 기능을 이용할 수 있는가?

- 온라인 서식도 키보드로만 사용이 가능하여야 한다. 또한 화면 낭독 프로그램과 같은 보조 기술과의 단축키 충돌이 없어야 한다.
- Tab 키와 Shift + Tab 키에 의한 초점 이동이 원활하여야 한다.

(12) JavaScript를 지원하지 않더라도 submit 기능이 동작하는가?

- 이 항목과 관련해서는 <II - 3.4 온라인서식 구성>을 참고하시오.

4.4.2 관련 국가 표준 : 항목 3.3

4.4.3 JavaScript 프로그래밍 지침 : 항목 3.4, 3.2

4.5 신기술의 사용

4.5.1 체크리스트

(13) JavaScript 코드를 지원하지 않는 경우에 대체 콘텐츠를 제공하는가?

- 브라우저에 따라서 스크립트를 지원하는 못하는 경우도 있다. 이 경우에 JavaScript의 기능을 중지시키더라도 해당 웹 콘텐츠가 제공하려는 기능을 이용할 수 있어야 한다. 스크립트에 대한 대체 콘텐츠는 HTML과 CSS로 가능하므로 noscript를 제공한다. 이와 관련한 예는 <II - 5.4 JavaScript 대체 기법>을 참고하라.

(14) 대체 콘텐츠는 간단명료하면서 원본과 동일한 기능을 수행하고 있는가?

- (13)에서 설명한 바와 같이 JavaScript의 대체 콘텐츠는 대부분 HTML과 CSS로 비슷한 기능을 수행하도록 코딩할 수 있다. 이 부분에 관한 예제는 <II - 5.4 JavaScript 대체 기법>을 참고하라.

4.5.2 관련 국가 표준 : 항목 4.1

4.5.3 JavaScript 프로그래밍 지침 : 항목 3.5

4.6 표준의 준수

4.6.1 체크리스트

(15) 사용하는 JavaScript코드는 표준을 준수하고 있는가?

- 접근 가능한 웹 콘텐츠를 개발하기 위해서는 W3C 표준을 준수하여야 한다. 마찬가지로 크로스 브라우징을 위해서도 W3C 표준을 준수하여야 한다.
- 참고로 준수하여야 하는 표준은 WCAG 1.0, WCAG 2.0, DOM 표준, WAI-ARIA 1.0, CSS Level 2, HTML 4.01을 포함한다. JavaScript 표준으로 ECMA(European Computer Manufacturers Association)가 제정한 ECMA-262가 있다. JavaScript는 Netscape가 ECMA-262를 나름대로 구현한 것이다. 한편 Microsoft사는 ECMA-262를 자사의 목적에 맞게 구현하여 JScript라고 부른다. 이렇듯 이해 당사자들이 서로 다른 전략을 추구하는 바람에 크로스 브라우징의 어려움이 발생하고 있다. 그러나 모든 브라우저가 JavaScript 버전 1.5를 지원하고 있으므로 호환성을 위해서는 버전 1.5를 기반으로 코딩해야 한다.

4.6.2 관련 국가 표준 : 없음

4.6.3 JavaScript 프로그래밍 지침 : 항목 3.6

5. JavaScript 애플리케이션 콘텐츠 구현 예

5.1 내비게이션 제작기법

5.1.1 가로 1단 메뉴

<그림 II - 9>는 가로 1단, 5개의 주 메뉴만으로 구성된 메뉴의 예로, Tab 키를 이용하여 메뉴간의 이동이 가능한 예제이다.

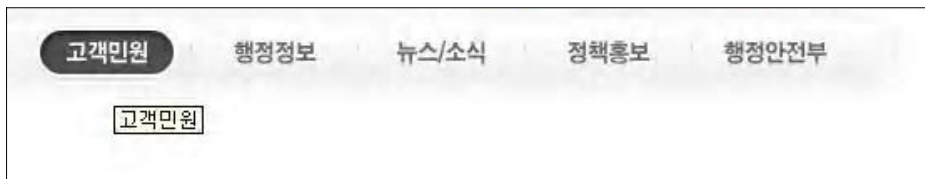


그림 II - 9. 가로 1단 메뉴 예제

<그림 II - 9>에서 Tab키를 누를 때마다 초점이 ‘고객민원’, ‘행정정보’ ... ‘행정안전부’ 등의 순서로 이동한다.

화면 낭독 프로그램을 사용하는 경우에는 초점의 이동 순서에 따라 메뉴의 내용을 읽어주어야 한다. Shift +Tab 키를 누를 때마다 초점의 이동 순서는 Tab 키의 이동 순서와 반대가 되어야 한다.

이 문서에서는 우리나라에서 구입할 수 있는 화면 낭독 프로그램인 센스리더 Professional Edition(센스), 드림보이스 7.0²³⁾(드림) 및 Jaws for Window 10.0(JAWS)의 세 가지 화면 낭독 프로그램을 이용하여 초점의 이동이 가능한가를 확인하였다. 참고로 이 문서는 화면 낭독 프로그램의

23) 이번 평가에서는 드림보이스 7.0 베타 버전을 사용하였음.

기능이나 성능을 평가하는 것이 목적이 아니다. 이 문서에서 화면 낭독 프로그램에 대한 시험 결과를 제시하는 것은 개발자들이 자체평가 단계에서 어떤 문제점에 봉착했을 때에 이 문제가 웹 콘텐츠 자체의 문제인지, 아니면 화면 낭독 프로그램의 문제인지를 구별할 수 있는 기준을 알려주기 위함이다.

위에 예로 든 세 가지 화면 낭독 프로그램은 초점이 주어진 요소를 읽어주는 것 외에도 매우 많은 기능을 가지고 있다. 그러나 이 문서에서는 여타의 기능에 대해서는 언급하지 않을 것이다. 이들 화면 낭독 프로그램에 대한 자세한 기능과 웹 콘텐츠의 내비게이션에 관한 기능 등은 각각의 화면 낭독 프로그램과 함께 제공되는 기술 자료나 웹 사이트를 참고하라.

참고

- 센스(센스리더) : 센스리더는 이 예제에서 정상적으로 동작한다. 참고로 센스리더를 설치하였을 때에 초기 값은 Tab 키에 의한 이 동시 초점을 화면에 보여주지 않도록 설정되어 있으므로 실험에서는 센스리더의 웹 브라우저 설정을 '자동포커스=선택'으로 변경하였다.
- 드림(드림보이스) : 드림보이스에서도 이 예제는 정상적으로 동작한다. 한 가지 주의할 점은 드림보이스의 경우에 Up 키와 Down 키가 각각 Shift + Tab 키 및 Tab 키와 대응한다는 점이다. 또한 Up 키와 Down 키를 빠르게 두 번 누르면 초점이 이동하여 가리키는 부분을 읽어주고, 한번 누를 경우에는 초점이 이동하지 않고 주어진 객체의 레이블, 대체 텍스트 등을 읽어준다.
- JAWS(Jaws for Windows) : JAWS는 정상적으로 초점이 이동하며, 초점이 주어진 객체의 레이블 또는 대체 텍스트를 읽어준다.

웹 페이지는 HTML의 하이퍼텍스트 링크로 구성되어 있으며 메뉴 선택 기능은 JavaScript로 구성하였다. <그림 II - 9>에 대한 HTML 코드는 다음과 같다.

1) HTML 코드 부분

O (Good)

```
//HTML Document

<div id="TopMenu">
  <div id="TopMenuSub">
    <ul>
      <li><a href="./고객민원.html"></a></li>
      <li><a href="./행정정보.html"></a></li>
      <li><a href="./뉴스소식.html"></a></li>
      <li><a href="./정책홍보.html"></a></li>
      <li><a href="./행정안전부.html"></a></li>
    </ul>
  </div>
</div>
<script type="text/javascript">
  var TopMenu1 = new fnTopMenu1_Type1;
  TopMenu1.DivName = "TopMenuSub";
  TopMenu1.fnName = "TopMenu1";
  TopMenu1.DefaultMenu = 0;

  TopMenu1.Start();
</script>
```

위의 HTML 문서와 같이 사용하는 JavaScript는 다음과 같다.

2) 스크립트 코드 부분

O (Good)

```
// JavaScript Document

function fnTopMenu1_Type1() {
    this.Start = function() {
        this.MenuBox = document.getElementById(this.DivName
        .getElementsByTagName("ul")[0].getElementsByTagName("li");

        // 메뉴의 갯수를 파악하는 부분
        this.MenuLength = this.MenuBox.length;

        // 메뉴의 링크부분에 마우스나 키보드의 반응을 넣는 부분
        for ( var i=0; i<this.MenuLength; i++ ) {
            this.MenuLink = this.MenuBox.item(i)
            .getElementsByTagName("a")[0];
            this.MenuLink.i = i;
            this.MenuLink.fnName = this.fnName;
            this.MenuLink.onmouseover =
            this.MenuLink.onfocus = function() {
                eval( this.fnName +
                    ".fnMouseOver("+this.i+")" )
            }
            this.MenuLink.onmouseout
            = this.MenuLink.onblur
            =function() {eval( this.fnName
                +".fnMouseOver()")
            }
        }
    }
}
```



```

        if ( this.DefaultMenu != 0 ) {
            this.fnMouseOver(this.DefaultMenu - 1);
        }
    }

    //메뉴의 링크부분에 마우스나 키보드의 반응에 의해 실행하는 부분
    this.fnMouseOver = function(val) {
        for ( var i=0; i<this.MenuLength; i++ ) {
            this.MenuImg
                =this.MenuBox.item(i).getElementsByTagName("a")[0]
                .getElementsByTagName("img")[0];
            if ( i == val ) {
                this.MenuImg.src
                    = this.MenuImg.src.replace("_off.gif","_on.gif");
            } else {
                this.MenuImg.src
                    = this.MenuImg.src.replace("_on.gif","_off.gif");
            }
        }
    }
}

```

5.1.2 가로 2단 메뉴

<그림 II - 10>은 가로 2단, 5개의 아이템으로 만든 메뉴의 구성 예로써 Tab 키를 이용하여 메뉴를 사용할 수 있도록 만든 것이다. 주 메뉴에 초점이 주어지면 하위 메뉴가 자동적으로 펼쳐지며, Tab에 의해 이동하는 경우 이동 순서는 초점이 주어진 주 메뉴를 읽고, 이어서 Tab 키를 누를 때마다 하위 메뉴로 하나씩 이동한다. 또한 하위 메뉴를 마지막까지 이동한 후에는 다음번 주 메뉴로 이동한다.

Shift + Tab 키를 이용하는 경우에는 펼쳐있는 하위 메뉴를 역순으로 이동한 후에 이전 주 메뉴로 이동한다. 이후에는 주 메뉴 간을 역순으로 이동한다.



그림 II - 10. 가로 2단 메뉴 예제

화면 낭독 프로그램이 사용되더라도 Tab 키에 의한 초점 이동과정은 화면 낭독 프로그램이 설치되지 않았을 때와 동일하여야 한다. 그러나 Shift + Tab 키를 이용하여 역순으로 이동하는 경우에는 열려있는 하위 메뉴를 역순으로 하나씩 이동한 후에 주 메뉴로 초점이 이동한다. 이후에는 주 메뉴에 초점이 있는 경우에 Shift + Tab 키를 누르면 직전 주 메뉴로 초점이 이동하여야 한다.

참고

- 센스 : 센스리더 또는 드림보이스를 사용하는 경우에 Tab키에 의한 이동은 정상적이다. 그러나 Shift + Tab 키에 의한 역방향 이동시에는 하위 메뉴를 역순으로 주 메뉴 간을 이동하지 않고, 직전 주 메뉴에 속한 하위 메뉴의 제일 마지막으로 이동한다. 이 부분은 센스리더의 개선이 필요한 부분이다.
- 드림 : Tab 키 및 Shift + Tab 키의 경우 초점이동이 정상적이다. 그러나 초점이 주어지는 요소를 읽어주지 않는다. Down 키는 Tab 키와 동일한 기능을 하며, 정상적으로 동작한다. 그러나 Up 키는 Shift + Tab 키와 동일한 기능을 수행하나, Up 키를 이용하여 역방향 이동시에는 하위 메뉴를 역순으로 주 메뉴 간을 이동하지 않고, 직전 주 메뉴에 속한 하위 메뉴의 제일 마지막으로 이동한다. 이 부분은 드림보이스의 개선이 필요한 부분이다.
- JAWS : Tab 키 및 Shift + Tab 키의 경우 초점이동이 정상적이다.

가로 2단 메뉴의 HTML 코드와 JavaScript 코드는 하위 메뉴가 있다는 것을 제외하면 가로 1단 메뉴의 코드와 그 구조가 동일하므로 이 문서에서 코드에 대한 자세한 설명을 기술하지 않는다.

1) HTML 코드 부분

O (Good)

```
//HTML Document

<body>
<div id="TopMenu">
  <div id="TopMenuSub">
    <ul>
```

```

<li class="menu1">
    <a href="/menu1.html">
</a>
    <div class="TopSubMenu">
        <ul>
            <li><a href="/고객센터.html">
</a></li>
            <li><a href="/질의응답.html">
</a></li>
            <li><a href="/국민제안.html">
</a></li>
            <li><a href="/국민신고.html">
</a></li>
        </ul>
    </div>
</li>
.
.
<li class="menu5">
    <a href="/행정안전부.html">
    </a>
    <div class="TopSubMenu">
        <ul>
            <li><a href="/소개.html">
</a></li>
            <li><a href="/조직.html">
</a></li>
            <li><a href="/장관소개.html">
</a></li>
            <li><a href="/차관소개.html">
</a></li>
        </ul>
    </div>
</li>

```

```

        </ul>
    </div>
</div>
<script type="text/javascript">
    var TopMenu1 = new fnTopMenu1_Type1;
    TopMenu1.DivName = "TopMenuSub";
    TopMenu1.fnName = "TopMenu1";
    TopMenu1.DefaultMenu = 0;
    TopMenu1.DefaultSubMenu = 0;

    TopMenu1.Start();
</script>
</body>

```

2) 스크립트 코드 부분

O (Good)

```

// JavaScript Document

function fnTopMenu1_Type1() {
    this.menu = new Array();
    this.menuseq = 0;
    this.Start = function() { this.MenuBox = document
    .getElementById(this.DivName).getElementsByTagName("ul")[0].childNodes
    ;

        // 메뉴의 갯수를 파악하는 부분
        this.MenuLength = this.MenuBox.length;

        // 메뉴의 주 메뉴 링크부분에
        // 마우스나 키보드의 반응을 넣는 부분
        for ( var i=0; i<this.MenuLength; i++ ) {
            if(this.MenuBox.item(i).tagName != "LI" )
                {continue;}
        }
    }
}

```

```

this.MenuLink
    = this.MenuBox.item(i).getElementsByTagName("a")[0];
this.MenuLink.i = i;
this.MenuLink.fnName = this.fnName;
this.MenuLink.onmouseover
    = this.MenuLink.onfocus
    = function() {
        eval(this.fnName+".fnMouseOver("+ this.i + ")")
    }

this.MenuSubBox
    = this.MenuBox.item(i).getElementsByTagName("div")[0];
this.MenuSubMenu
    = this.MenuSubBox.getElementsByTagName("ul")[0]
        .getElementsByTagName("li");
this.MenuSubMenuLength
    = this.MenuSubMenu.length;

// 메뉴 2탭스 링크에 마우스나 키보드의 반응 넣는 부분
for ( var j=0; j<this.MenuSubMenuLength; j++ ) {
    this.MenuSubLink = this.MenuSubMenu.item(j)
        .getElementsByTagName("a")[0];
    this.MenuSubLink.i = i;
    this.MenuSubLink.j = j;
    this.MenuSubLink.fnName = this.fnName;
    this.MenuSubLink.onmouseover =
    this.MenuSubLink.onfocus = function() {
        eval(this.fnName + ".fnMouseSubOver
            (" + this.i + "," + this.j + ")") }
    this.MenuSubLink.onmouseout
        = this.MenuSubLink.onblur = function() {
            eval(this.fnName + ".fnMouseSubOut
                (" + this.i + "," + this.j + ")")    }
        }
}

```

```

        this.MenuSubBox.style.display = "none";
        this.menuseq++;
        this.menu[this.menuseq] = i
    }
    if ( this.DefaultMenu != 0 ) {
        this.fnMouseOver(this.menu[this.DefaultMenu]);
    }
    if ( this.DefaultSubMenu != 0 ) {
        this.fnMouseSubOver(this.menu[this.DefaultMenu];
        this.DefaultSubMenu- 1);
    }
}

// 메뉴의 주 메뉴 링크부분에
// 마우스나 키보드의 반응에 의해 실행하는 부분
this.fnMouseOver = function(val) {
    for ( var i=0; i<this.MenuLength; i++ ) {
        if ( this.MenuBox.item(i).tagName != "LI" )
            {continue;}
        this.MenuImg
= this.MenuBox.item(i)
        .getElementsByTagName("a")[0]
        .getElementsByTagName("img")[0];
        this.MenuSDiv
= this.MenuBox.item(i)
        .getElementsByTagName("div")[0];
        if ( i == val )
            { this.MenuImg.src
= this.MenuImg.src.replace("_off.gif","_on.gif");
            this.MenuSDiv.style.display = "block"; }
        else
            { this.MenuImg.src
= this.MenuImg.src.replace("_on.gif","_off.gif");

```

```

        this.MenuSDiv.style.display="none";  }
    }

    // 메뉴의 하위 메뉴 링크부분에
    // 마우스나 키보드의 반응에 의해 실행하는 부분
    this.fnMouseSubOver = function(mnum,snum) {
        this.SubMenuImg
            = this.MenuBox.item(mnum).getElementsByTagName("div")[0]
              .getElementsByTagName("ul")[0].getElementsByTagName("li")
                [snum].getElementsByTagName("a")[0]
                  .getElementsByTagName("img")[0];
        this.SubMenuImg.src
            = this.SubMenuImg.src.replace("_off.gif","_on.gif");
    }

    this.fnMouseSubOut = function(mnum,snum) {
        this.SubMenuImg
            = this.MenuBox.item(mnum).getElementsByTagName("div")[0]
              .getElementsByTagName("ul")[0].getElementsByTagName("li")
                [snum].getElementsByTagName("a")[0]
                  .getElementsByTagName("img")[0];
        this.SubMenuImg.src
            = this.SubMenuImg.src.replace("_on.gif","_off.gif");
    }
}

```

5.1.3 드롭다운 메뉴

드롭다운(Drop-down)메뉴는 마우스 롤오버(mouse rollover)에 의하여 하위 메뉴가 펼쳐지는 형태의 메뉴이다. 따라서 키보드 조작에 따른 결과는 앞에 예로 든 가로 2단 메뉴의 경우와 동일하다.



그림 II - 11. 드롭다운 메뉴 예제

드롭다운 메뉴의 키보드 내비게이션은 다음과 같다. Tab 키를 이용하여 ‘관련사이트’ 링크 메뉴로 초점이 이동하면, Enter 키를 눌러 하위 메뉴를 화면에 보여주고, 첫 번째 하위 메뉴(‘정부청사관리소’)로 초점이 이동한다. 이어서 Tab 키를 누를 때마다 다음 하위 메뉴로 이동한다. 마지막 하위 메뉴(‘소청심사위원회’)로 초점이 이동한 후에는 드롭다운 메뉴를 빠져나가면서 드롭다운 메뉴가 비활성화 된다.

Shift + Tab 키에 의한 이동시에는 현재의 메뉴가 주 메뉴(‘관련사이트’)이면 드롭다운 메뉴를 빠져나가면서 드롭다운 메뉴가 비활성화 되어야 하며, 하위 메뉴에 초점이 있을 경우에는 Shift + Tab 키를 누를 때마다 역순으로 이동하여 주 메뉴로 초점이 이동한다.

화면 낭독 프로그램을 사용할 경우에는 Tab 키와 Shift + Tab 키의 조작에 따라 초점이 주어지는 메뉴를 읽어주어야 한다. 예를 들어 ‘관련사이트’ 링크 메뉴에 초점이 주어지면 대체 텍스트(‘관련사이트, 엔터키를 누르시오’)를 읽어주어야 한다.

참고

- 센스 : 센스리더는 Tab 키와 Shift + Tab 키에 의한 초점 이동시 읽어주는 내용은 정상적이다. 그러나 마지막 하위 메뉴(‘소청심사위원회’)에서 Tab 키에 의한 초점이동시 드롭다운 메뉴를 빠져나가지 못하는 현상이 발생한다. 이 부분은 개선이 필요한 부분이다.
- 드림/JAWS : 드림보이스와 JAWS for Windows는 Tab 키와 Shift + Tab 키에 의한 초점이동이 정상이며, 이 때 읽어주는 내용도 정확하다.

1) HTML 코드 부분

O (Good)

```
// HTML Document

<body>

<div id="Content">
</div>

<div id="LinkSite">
  <div id="LinkSiteSub">
    <dl>
      <dt><a href="#LinkSiteList">
        
      </a></dt>
      <dd id="LinkSiteList">
        <ul>
          <li><a href="http://./정부청사관리소.html">
            
          </a></li>
```

```

<li><a href="#">
    </a></li>
<li><a href="#">
    </a></li>
<li><a href="#">
    </a></li>
<li><a href="#">
    </a></li>
<li><a href="#">
    </a></li>
</ul>
</dd>
</dl>
</div>
</div>
<script type="text/javascript">
    var isShiftDown = false;
    document.body.onkeydown = function(e){
        var eCode = (window.netscape) ? ev.which
            : event.keyCode;
        if(eCode == 16)
            isShiftDown = true;
    }
    document.body.onkeyup = function(e){ isShiftDown = false;}

    var LinkSite1 = new fnLinkSite_Type1;
    LinkSite1.DivName = "LinkSiteSub";
    LinkSite1.fnName = "LinkSite1";
    LinkSite1.Start();

```

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

2) 스크립트 코드 부분

O (Good)

```
// JavaScript Document

function fnLinkSite_Type1() {

    this.Start = function() {

        this.LinkSite
            = document.getElementById(this.DivName)
              .getElementsByTagName("dl")[0];
        this.MainLink
            = this.LinkSite.getElementsByTagName("dt")[0]
              .getElementsByTagName("a")[0];
        this.SubMenuBox
            = this.LinkSite.getElementsByTagName("dd")[0];

        this.MainLink.fnName = this.fnName;

        // 관련사이트 메뉴의 클릭에 반응 하는 부분
        this.MainLink.onclick = function() {
            eval( this.fnName + ".fnMouseOver(0,this)") }
        // 관련사이트 메뉴의 MouseOut시 발생하는 부분
        this.MainLink.onmouseout = function() {
            eval( this.fnName + ".fnMouseOut(0,this)") }
        ;
        // 관련사이트 서브메뉴의 박스 부분에
        // 마우스나 키보드의 반응을 넣는 부분
        this.SubMenuBox.style.display = "none";
        this.SubMenuBox.fnName = this.fnName;
```

```

        this.SubMenuBox.onmouseover
        = this.SubMenuBox.onfocus = function() {
            eval( this.fnName + ".fnMouseOver(1,this)" )
        }
        this.SubMenuBox.onmouseout = this.SubMenuBox.onblur
        = function() { eval( this.fnName + ".fnMouseOut(1,this)" )
    }

    this.SubMenuList
    = this.SubMenuBox.getElementsByTagName("ul")[0]
    .getElementsByTagName("li");
    this.SubMenuListTotal = this.SubMenuList.length;

    // 관련사이트 서브메뉴의 링크 부분에
    // 마우스나 키보드의 반응을 넣는 부분
    for ( var i=0; i<this.SubMenuListTotal; i++ ) {
        this.SubLink
        = this.SubMenuList.item(i)
        .getElementsByTagName("a")[0];
        this.SubLink.i = i;
        this.SubLink.fnName = this.fnName;
        this.SubLink.onmouseover = this.SubLink.onfocus
        = function() {
            eval( this.fnName + ".fnSubMouseOver("+this.i+")" )
        }

        this.SubLink.onmouseout = this.SubLink.onblur
        = function() {
            eval( this.fnName + ".fnSubMouseOut("+this.i+")" )
        }
    }
}

// 관련사이트 메뉴의 마우스나 키보드의 반응에 의해 실행하는 부분
this.fnMouseOver = function(num,val) {
    if ( num == 0 ) {
        this.MainMenuImg
    }
}

```

```

        = val.getElementsByTagName("img")[0];
    } else if ( num == 1 ) {
        this.MainMenuImg
        =this.MainLink.getElementsByTagName("img")[0];
    }
    this.SubMenuBox.style.display = "block";
    this.MainMenuImg.src
        = this.MainMenuImg.src.replace("_off.gif","_on.gif");
}

this.fnMouseOut = function(num,val) {
    if ( num == 0 ) {
        this.MainMenuImg
        = val.getElementsByTagName("img")[0];
    } else if ( num == 1 ) {
        this.MainMenuImg
        =
this.MainLink.getElementsByTagName("img")[0];
    }
    this.SubMenuBox.style.display = "none";
    this.MainMenuImg.src
        = this.MainMenuImg.src.replace("_on.gif","_off.gif");
}

// 관련사이트 서브메뉴의 링크 부분에
// 마우스나 키보드에 의해 실행하는 부분
this.fnSubMouseOver = function(val) {
    for ( var i=0; i<this.SubMenuListTotal; i++ ) {
        this.MenuImg
        = this.SubMenuList.item(i)
        .getElementsByTagName("a")[0]
        .getElementsByTagName("img")[0];
        if ( i == val ) {
            this.MenuImg.src

```

```

        = this.MenuImg.src.replace("_off.gif","_on.gif");
    } else {
        this.MenuImg.src
        = this.MenuImg.src.replace("_on.gif","_off.gif");
    }
    }
}

// 관련사이트 서브메뉴에서 포커스를 벗어날 때
this.fnSubMouseOut = function(val) {

    if(isShiftDown)
        return;

    if(val == this.SubMenuListTotal - 1){

        this.SubMenuBox.style.display = "none";
        this.MainMenuImg.src
        =this.MainMenuImg.src.replace("_on.gif","_off.gif");

        for ( var i=0; i<this.SubMenuListTotal; i++ ) {
            this.MenuImg
                = this.SubMenuList.item(i)
                    .getElementsByTagName("a")[0]
                    .getElementsByTagName("img")[0];
            if ( i == val ) {
                this.MenuImg.src
                = this.MenuImg.src.replace("_off.gif","_on.gif");
            } else {
                this.MenuImg.src
                = this.MenuImg.src.replace("_on.gif","_off.gif");
            }
        }
    }
}

```

```

    }
}

```

5.1.4 가로 탭 메뉴

(1) 가로 탭 메뉴와 Enter키의 조합

<그림 II - 12>는 Tab 키에 의하여 주 메뉴를 한차례 이동한 후에 순차적으로 화면에 나타난 하위 메뉴 및 '더보기' 링크로 이동하는 '가로 탭' 메뉴이다. Tab 키로 이동 중에 Enter 키를 누르면 해당 주 메뉴에 속한 하위 메뉴가 펼쳐지고, 하위 메뉴의 첫 번째 메뉴로 초점이 이동한다.

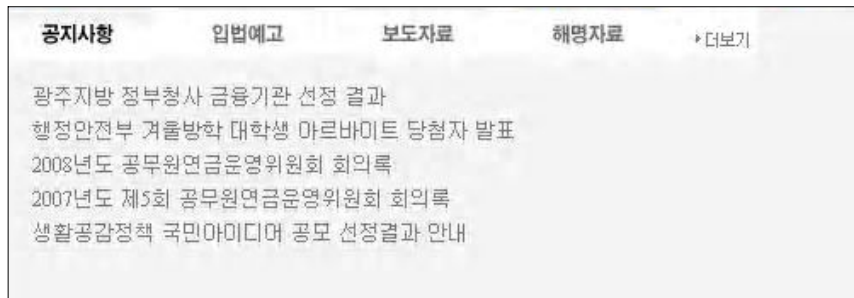


그림 II - 12. Enter키에 의해 동작하는 가로 탭 메뉴 예제

Shift + Tab 키에 의한 이동 순서는 다음과 같다. 초점이 주 메뉴에 있을 경우에는 주 메뉴를 역순으로 이동한다. 만일 초점이 하위 메뉴에 있으면 해당 하위 메뉴를 역순으로 이동한 후에 주 메뉴를 읽어주고, 이어서 주 메뉴 간을 역순으로 이동한다. 화면 낭독 프로그램이 설치되어도 Tab 키와 Shift + Tab 키에 의한 초점 이동이 정상적이어야 하며, 초점이 주어지는 요소의 텍스트 또는 대체 텍스트를 읽어주어야 한다.

참고

- **센스** : 센스리더는 Tab 키에 의한 초점 이동시 콘텐츠를 정상적으로 읽어준다. 그러나 Shift + Tab 키를 이용한 초점 이동시에는 탭을 역순으로 읽어야 하나, 하위 메뉴를 역순으로 내비게이션 한다. 이 부분은 센스리더의 개선이 필요한 부분이다.
- **드림** : 드림보이스는 Tab 키와 Shift + Tab 키에 의하여 초점이 정상적으로 이동하지만 읽어주지는 않는다. Up 키와 Down 키는 초점이동과 읽어주는 것이 정상이다. 그러나 주 메뉴에서 Enter 키를 치면 하위 메뉴로 이동하지 않고 연결된 하이퍼텍스트로 이동한다. 그러므로 Enter 키를 사용하여 하위 메뉴를 나타내도록 웹 콘텐츠를 구성해서는 안된다. 이 부분은 개선이 필요하다.
- **JAWS** : Up 키와 Down 키로 초점을 이동시킬 수 있다. Tab 키와 Shift 키는 주 메뉴의 이동에만 사용된다.

1) HTML 코드 부분

O (Good)

```
//HTML Document

<head>
<title>가로 탭 메뉴 Type1</title>
<link rel="stylesheet" href="./common/style.css" type="text/css" />
<script type="text/javascript" src="./common/script.js"></script>
</head>

<body>
<div id="TabMenu">
  <div id="TabMenuSub">
    <ul>
      <li><a href="#TabNotice0">
        </a></li>
```

```

        <li><a href="#TabNotice1">
            </a></li>
        <li><a href="#TabNotice2">
            </a></li>
        <li><a href="#TabNotice3">
            </a></li>
    </ul>
</div>
<div id="TabNotice0" class="TabNoticeStyle">
    <ul>
        <li><a href="/notice.html" >광주지방 정부청사 금융기관 선정
            결과</a></li>
        <li><a href="/notice.html" >행정안전부 겨울방학 대학생
            ... </a></li>
        <li><a href="/notice.html">2008년도 공무원연금운영위원회
            회의록</a></li>
        <li><a href="/notice.html">2007년도 제5회
            공무원연금운영위원회...</a></li>
        <li><a href="/notice.html">생활공감정책 국민아이디어
            공모 ...</a></li>
    </ul>
    <a href="/noticemore.html" class="more" onblur="checkLink(0)">
        </a>
</div>
.
.
</div>
<script type="text/javascript">
    var TabMenu1 = new fnTabMenu_Type1;
    TabMenu1.MenuName = "TabMenuSub";
    TabMenu1.DivName = "TabNotice";
    TabMenu1.fnName = "TabMenu1";
    TabMenu1.Start();
</script>

```

```
</body>
```

2) 스크립트 코드 부분

O (Good)

```
// JavaScript Document

function fnTabMenu_Type1() {
    this.CurrentMenu = 0;

    this.Start = function() {
        this.MenuBox
            = document.getElementById(this.MenuName)
              .getElementsByTagName("ul")[0].getElementsByTagName("li");

        // 탭 메뉴의 갯수를 파악하는 부분
        this.MenuLength = this.MenuBox.length;

        // 탭 메뉴의 링크부분에 마우스나 키보드의 반응을 넣는 부분
        for ( var i=0; i<this.MenuLength; i++ ) {
            this.MenuLink
                = this.MenuBox.item(i).getElementsByTagName("a")[0];
            this.MenuLinkBtn
                = this.MenuLink.getElementsByTagName("img")[0];
            if ( i == this.CurrentMenu ) {
                document.getElementById(this.DivName + i).style.display = "block";
                this.MenuLinkBtn.src
                    = this.MenuLinkBtn.src.replace("_off.", "_on.");
            } else {
                document.getElementById(this.DivName + i).style.display = "none";
                this.MenuLinkBtn.src
                    = this.MenuLinkBtn.src.replace("_on.", "_off.");
            }
            this.MenuLink.i = i;
        }
    }
}
```

```

this.MenuLink.fnName = this.fnName;
this.MenuLink.onclick = function() {
    eval( this.fnName + ".fnMouseOver(" + this.i + ")");
    return false;
}

// 주 메뉴에서 Enter 키를 눌렀을 경우
this.MenuLink.onkeydown=function(e){
    if(event.keyCode == 13){
        eval( this.fnName + ".fnMouseOver2(" + this.i + ")");
    }
}

// 주 메뉴의 가장 마지막 링크에서 포커스를 검사한다.
// 현재 주 메뉴가 선택되어 있고 링크가 하위 메뉴에 있다면,
// 이전에 선택한 주 메뉴로 포커스를 이동시킨다.
if(i == this.MenuLength-1){
    this.MenuLink.onfocus = function(){
        try{
            if(currentTab < i && isSubMenu){
                document.getElementById("TabMenuSub")
                    .getElementsByName("a")[currentTab].focus();
                isSubMenu = false;
            }
        }catch(e){
        }
        return false;
    }
}
}

//탭 메뉴의 링크부분에 마우스나 키보드 반응에 의해 실행하는 부분
this.fnMouseOver = function(val) {

```

```

for ( var i=0; i<this.MenuLength; i++) {
    this.MenuLink=this.MenuBox.item(i)
        .getElementsByTagName("a")[0];
this.MenuLinkBtn
    = this.MenuLink.getElementsByTagName("img")[0];
if ( i == val ) {
    document.getElementById(this.DivName+i).style.display
        = "block";
    this.MenuLinkBtn.src=this.MenuLinkBtn.src.replace
        ("_off.", "_on.");
} else {
    document.getElementById(this.DivName+i).style.display
        = "none";
    this.MenuLinkBtn.src=this.MenuLinkBtn.src.replace("_on.", "_off.");
}
}
}

this.fnMouseOver2 = function(val) {
    currentTab = val;
    isSubMenu = true;
    for ( var i=0; i<this.MenuLength; i++) {
        this.MenuLink
            = this.MenuBox.item(i).getElementsByTagName("a")[0];
        this.MenuLinkBtn
            = this.MenuLink.getElementsByTagName("img")[0];
        if ( i == val ) {
            document.getElementById(this.DivName+i).style.display="block";
            this.MenuLinkBtn.src
                = this.MenuLinkBtn.src.replace("_off.", "_on.");
        } else {
            document.getElementById(this.DivName+i).style.display="none";
            this.MenuLinkBtn.src
                = this.MenuLinkBtn.src.replace("_on.", "_off.");
        }
    }
}

```

```

    }

    // 하위 메뉴의 첫번째 링크로 초점이 이동
    if(i == val) {
        document.getElementById("TabNotice"
            + val).getElementsByTagName("a")[0].focus(); }
    }
}

// 더보기를 눌렀을 경우 주 메뉴에서
// 선택되지 않은 다음 메뉴가 있는지 검사한다.
function checkLink(id){
    try{ var obj = document.getElementById("TabMenuSub")
        .getElementsByTagName("a")[id + 1];
        obj.focus();
    }catch(e){
    }
}
}

```

(2) 가로 탭 메뉴와 롤오버 기능의 조합

<그림 II - 13>은 Tab 키를 이용하여 주 메뉴로 초점을 이동시키면, 하위 메뉴가 자동적으로 나타나는 롤오버 기능을 이용한 ‘가로 탭’ 메뉴이다. 이 문서의 <II - 5.1.4(1) 가로 탭 메뉴와 Enter키의 조합>과의 차이점은, Tab 키로 이동할 때 해당 주 메뉴에 속한 하위 메뉴가 자동적으로 펼쳐지고, 이후에 Tab 키를 누르면 하위 메뉴의 첫 번째 메뉴로 초점이 이동한다는 점이다.

예를 들어 아래 그림에서 Tab 키에 의한 이동순서는 ‘공지사항’, ‘광주 지방 ... 선정 결과’, ‘행정안전부 ... 당첨자발표’, ... ‘생활 공감정책 ... 안

내’의 순서이다. 이어서 ‘더보기’ 링크로 이동하고, 다음 주 메뉴인 ‘입법예고’ 링크로 이동한다. 이어서 자동적으로 ‘입법예고’의 하위 메뉴가 펼쳐지고, 하위 메뉴를 하나씩 이동한 후, 마지막으로 ‘더보기’ 링크로 이동한다.

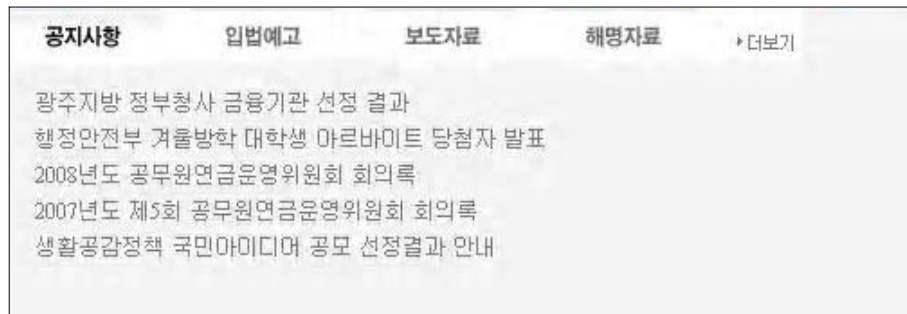


그림 II - 13. 롤오버 기능에 의해 동작하는 가로 탭 메뉴 예제

Shift + Tab 키에 의한 이동 순서는 Tab 키에 의한 이동순서의 역순과 차이가 있다. 만일 초점이 하위 메뉴에 있을 경우에는 하위 메뉴를 역순으로 읽어준 후에 주 메뉴를 읽어주고, 이어서 주 메뉴 이전의 주 메뉴로 초점이 이동한다. 이 때 자동적으로 하위 메뉴가 펼쳐지며, Tab 키에 의하여 하위 메뉴의 마지막 메뉴로 초점이 이동한다. ‘더보기’ 링크는 Shift + Tab 키로 이동시에 건너뛰게 된다. 화면 낭독 프로그램을 사용하는 경우에는 초점이 주어지는 요소의 텍스트 또는 대체 텍스트를 읽어주어야 한다.

참고

- 센스 : 센스리더는 이 예를 정상적으로 읽어준다. 다만 초점이 주어지는 부분이 화면에 보인 초점의 위치를 벗어난다. 이 부분은 개선이 필요하다.
- 드림/JAWS : 정상적으로 읽어준다.

전체적인 기능을 비교해 보면, 롤오버에 의한 가로 탭 메뉴가 Enter 키를 이용한 가로 탭 메뉴에 비하여 화면 구성과 화면 낭독 프로그램과의 관계 면에서 더 직관적이다.

1) HTML 코드 부분

O (Good)

```
//HTML Document

<head>
<title>가로 탭 메뉴 Type2</title>
<link rel="stylesheet" href="./common/style.css"
  type="text/css" />
<script type="text/javascript"
  src="./common/script.js"></script>
</head>

<body>
<div id="TabMenu">
  <div class="TabNoticeStyle">
    <a href="#TabNotice0">
      
    </a>
    <ul id="TabNotice0">
      <li><a href="./notice.html">광주지방 정부청사 금융기관
```



```

                선정 결과</a></li>
            <li><a href="./notice.html">행정안전부 겨울방학 대학생
                ...</a></li>
            <li><a href="./notice.html">2008년도 공무원연금운영위원회
                ...</a></li>
            <li><a href="./notice.html">2007년도 제5회 공무원연금운영
                ...</a></li>
            <li><a href="./notice.html">생활공감정책 국민아이디어 공모
                ...</a></li>
        </ul>
        <a id="TabNotice0More" href="./notice.html" class="more">
            </a>
    </div>
    .
    .
</div>

<script type="text/javascript">
    var TabMenu1 = new fnTabMenu_Type1;
    TabMenu1.MenuName = "TabMenu";
    TabMenu1.DivName = "TabNotice";
    TabMenu1.fnName = "TabMenu1";

    TabMenu1.Start();
</script>

</body>

```

2) 스크립트 코드 부분

O (Good)

```
// JavaScript Document

function fnTabMenu_Type1() {
    this.CurrentMenu = 0;
    currentTab = 0;
    this.Start = function() {
        this.MenuBox
            = document.getElementById(this.MenuName)
              .getElementsByName("ul");
        this.MenuLength = this.MenuBox.length;
        this.MenuLayer
            = document.getElementById(this.MenuName)
              .getElementsByName("div");

        for ( var i=0; i<this.MenuLength; i++ ) {
            this.MenuLink
                = this.MenuLayer.item(i).getElementsByName("a")[0];
            this.MenuLinkBtn
                = this.MenuLink.getElementsByName("img")[0];
            if ( i == this.CurrentMenu ) {
                document.getElementById(this.DivName+i).style.display
                    ="block";
                document.getElementById(this.DivName+i+"More")
                    .style.display = "block";
                this.MenuLinkBtn.src=this.MenuLinkBtn.src.replace
                    ("_off.", "_on.");
            } else {
                document.getElementById(this.DivName+i).style.display
                    ="none";
                document.getElementById(this.DivName+i+"More")
                    .style.display = "none";
                this.MenuLinkBtn.src=this.MenuLinkBtn.src.replace
```

```

        ("_on.", "_off.");
    }
    this.MenuLink.i = i;
    this.MenuLink.fnName = this.fnName;
    this.MenuLink.onmouseover = this.MenuLink.onfocus
    = function() {
        eval( this.fnName + ".fnMouseOver(" + this.i + ")") }
    }
}

this.fnMouseOver = function(val) {
    isBack = false;
    if(currentTab > val)
        { isBack = true; }
    currentTab = val;
    for ( var i=0; i<this.MenuLength; i++) {
        this.MenuLink
        = this.MenuLayer.item(i).getElementsByTagName("a")[0];
        this.MenuLinkBtn
        = this.MenuLink.getElementsByTagName("img")[0];
        if ( i == val ) {
            document.getElementById(this.DivName+ i).style.display
            = "block";
            document.getElementById(this.DivName+i+"More")
            .style.display = "block";
            this.MenuLinkBtn.src
            = this.MenuLinkBtn.src.replace("_off.", "_on.");
        } else {
            document.getElementById(this.DivName
            + i).style.display = "none";
            document.getElementById(this.DivName+i+"More")
            .style.display = "none";
            this.MenuLinkBtn.src
            = this.MenuLinkBtn.src.replace("_on.", "_off.");
        }
    }
}

```

```

    }

    // 하위 메뉴의 제일 마지막부터 차례로 이동하는 부분
    // 다음의 3줄을 빼면 메인 메뉴로만 이동합니다.
    if(isBack && i== val) {
        this.MenuLayer.item(i).getElementsByTagName("a")
        [this.MenuLayer.item(i).getElementsByTagName("a")
        .length-2].focus();
    }
}
}
}

```

5.2 접근성을 고려한 입력서식 제작기법

5.2.1 로그인

<그림 II - 14>에는 <form> 태그를 이용하여 Submit하는 전형적인 온라인 서식의 예제이다.

그림 II - 14. 로그인 화면 예제

II - 3.4.2절에서 이야기한 대로 <input type="image"...> 태그를 이용하여 Submit 할 수 있도록 하였으며, Submit 시에 JavaScript로 '아이디'와 '비밀번호'의 유효성을 검증하는 submitForum errorMessage 함수를 호출하고 있다.

submitForum errorMessage 함수의 실행결과 '아이디'가 빈공간일 경우에는 errorMessage를 경고 창으로 출력한 후에 '아이디' 입력창으로 초점이 이동하여 아이디를 입력할 수 있도록 한다. '비밀번호'가 빈 공간일 경우에도 마찬가지로 절차를 따른다.

이 프로그램에서 JavaScript는 기능을 제공하는 것에 국한되어 있으므로 Tab 키에 의한 이동순서는 '아이디' 입력창, '아이디저장' 체크박스, '비밀번호' 입력창, '로그인' 버튼, '회원이입' 링크, '아이디/비밀번호 찾기' 링크로 되어있다. Shift + Tab 키에 의한 이동순서는 역순이다. 화면 낭독 프로그램은 초점이 주어지는 구성요소를 읽어주어야 한다.

참고

- 센스/드림/JAWS : 정상적으로 동작한다.

1) HTML 코드 부분

O (Good)

```
//HTML Document

<head>
<title>로그인</title>
<link rel="stylesheet" href="./common/style.css" type="text/css" />
<script type="text/javascript" src="./common/script.js"></script>
</head>
```

```

<body>
<div id="MemberLogin">
  <div id="MemberLoginSub">
    <form id="LoginForm" name="LoginForm" method="post"
      action="" onsubmit=" return submitForum(this);">
      <fieldset>
        <legend>
          </legend>
          <p>
            <label for="LoginId">아이디</label>
            <input type="text" id="LoginId" name="LoginId"
              maxlength="15" value="" />
            <span>
              <input type="checkbox" id="LoginCheck"
                name="LoginCheck" value="1" />
              <label for="LoginCheck">아이디저장</label></span>
          </p>
          <p>
            <label for="LoginPass">비밀번호</label>
            <input type="password" id="LoginPass"
              name="LoginPass" maxlength="15" value="" />
            <input type="image" src="/img/login_submit.gif"
              class="loginbtn" alt="로그인" />
          </p>
        </fieldset>
      </form>
      <div class="MemberBox">
        <a href="/회원가입.html" class="point">회원가입</a>
        <a href="/비밀번호.html">아이디 / 비밀번호 찾기</a>
      </div>
    </div>
  </div>
</body>

```

2) 스크립트 코드 부분

O (Good)

```
// JavaScript Document

function submitForum(formEl) {
    var errorMessage = null;
    var objFocus = null;

    if ( formEl.LoginId.value.length == 0 ) {
        errorMessage = "아이디를 넣어 주세요";
        objFocus = formEl.LoginId;
    } else if ( formEl.LoginPass.value.length == 0 ) {
        errorMessage = "비밀번호를 넣어 주세요";
        objFocus = formEl.LoginPass;
    }

    if ( errorMessage != null ) {
        alert(errorMessage);
        objFocus.focus();
        return false;
    }
    return true;
}
```

5.2.2 회원가입

<그림 II - 15>에 보인 회원가입 화면도 전형적인 온라인 서식의 하나이다. 온라인 서식의 기본 구성은 <input> 태그를 이용하고 있으며 Submit 직전에 입력창에 입력된 정보의 정확성을 검증한다. 따라서 기본 구성은 로그인 창과 예와 같다.

회원가입 창에서 관심을 가져야 하는 내용은 화면에 시각적으로 나타난 '필수입력사항'을 보조 기술로 제공하는 방법이다. 즉 아래 그림에서 필수 입력사항인 '아이디', '비밀번호', '비밀번호 확인', '이름', '주소', '이메일 주소' 등이 그것이다.

이 경우 <input> 태그의 title 속성에 '필수'라는 단어를 추가하여 읽어 주도록 하거나, 입력창에 우선하는 이미지의 대체 텍스트와 레이블을 순서대로 읽어 주도록 하여 해결할 수 있다.

그림 II - 15. 회원가입 예제

참고

- 센스/드림/JAWS : 정상적으로 동작하지만, 레이블만 읽어주므로 필수항목임을 읽어줄 수 있는 방법이 강구될 필요가 있다.

우편번호 검색 기능은 JavaScript로 작성하였다(스크립트에서 밑줄로 표시한 부분). 따라서 JavaScript를 지원하지 않는 브라우저에서는 사용자가 직접 입력할 수 있다.

또한 우편번호 검색이 우편번호 입력창 보다 앞에 위치하도록 함으로써 우편번호를 입력한 후에 우편번호를 검색하도록 하는 것은 바람직하지 못하다. 따라서 '우편번호 검색' 버튼을 화면과 같이 우편번호 입력창 앞에 나타나도록 수정하는 것이 바람직하다. 물론 HTML 문서상에서는 '우편번호 검색' 버튼이 주소 창보다 앞에 위치하지만, CSS를 이용하여 뒷부분에 배치하는 것도 좋은 방법이다.

1) HTML 코드 부분

O (Good)

```
// HTML Document

<head>
<link rel="stylesheet" href="./common/style.css" type="text/css" />
<script type="text/javascript" src="./common/script.js"></script>
</head>

<body>

<div id="JoinWirte">
  <form id="WirteForm" name="WirteForm" method="post" action=""
    onsubmit="return submitForum(this);">
    <p class="check">
       표시 필수 입력사항
    </p>
    <fieldset>
      <legend>아이디 기본정보</legend>
      <table>
```

```

<tr>
<th><label for="JoinID">
    
    아이디 </label></th>
<td colspan="3">
    <input type="text" id="JoinID" name="JoinID"
    style="width:160px;" title="필수 아이디" value="" />
</td>
</tr>
<tr>
<th><label for="JoinPass">
    
    비밀번호</label></th>
<td><input type="password" id="JoinPass" name="JoinPass"
    style="width:160px;" title="필수 비밀번호" value="" /></td>
<th><label for="JoinPassRe">
    
    비밀번호 확인</label></th>
<td><input type="password" id="JoinPassRe" name="JoinPassRe"
    style="width:160px;" title="필수 비밀번호 확인" value="" /></td>
</tr>
</table>
</fieldset>
<fieldset>
<legend>개인 기본정보</legend>
<table>
<tr>
<th><label for="JoinName">
    
    이름(한글) </label></th>
<td>
<input type="text" id="JoinName" name="JoinName"
    style="width:180px;" title="필수 이름" value="" />
</td>

```

```

</tr>
<tr id="JoinPostJsApply">
  <th><label for="JoinPost1">
    주소
  </label></th>
  <td>
    <p class="Type1"> </p>
    <p class="Type1">
      <input type="text" id="JoinPost1" name="JoinPost1"
style="width:40px;" title="필수 우편번호 앞자리" value="" />
      - <input type="text" id="JoinPost2" name="JoinPost2"
style="width:40px;" title="필수 우편번호 뒷자리" value="" />
    </p>
    <p class="Type1">
      <input type="text" id="JoinAddress" name="JoinAddress"
style="width:90%;" title="필수 우편번호를 제외한 주소" value="" />
    </p>
  </td>
</tr>
<tr>
  <th><span>연락처</span></th>
  <td>
    <p>
<label for="TelA1" class="Type1">휴대폰</label>
<input type="text" id="TelA1" name="TelA1" style="width:40px;"
title="통신사 번호" value="" />
- <input type="text" id="TelA2" name="TelA2" style="width:40px;"
title="휴대폰번호 국번" value="" />
- <input type="text" id="TelA3" name="TelA3" style="width:40px;"
title="휴대폰번호 뒷네자리" value="" />
    </p>
    <p>
<label for="TelB1">전화번호</label>
<input type="text" id="TelB1" name="TelB1" style="width:40px;"

```

```

        title="전화번호 지역번호" value="" />
- <input type="text" id="TelB2" name="TelB2" style="width:40px;"
  title="전화번호 국번" value="" />
- <input type="text" id="TelB3" name="TelB3" style="width:40px;"
  title="전화번호 뒷네자리" value="" />
      </p>
    </td>
  </tr>
<tr>
  <th><label for="BoardEmail1">
    
    이메일 주소</label></th>
  <td>
    <input type="text" id="BoardEmail1" name="BoardEmail1"
      style="width:100px;" title="필수 이메일 아이디 입력" value="" />
    @ <input type="text" id="BoardEmail2" name="BoardEmail2"
      style="width:250px;" title="필수 이메일 서비스 입력" value="" />
  </td>
</tr>
</table>
</fieldset>
<p>
  <input type="image" src="./img/join_submit.gif" alt="회원가입" />
</p>
</form>
</div>

<script type="text/javascript">
// 우편번호 부분에 Javascript가 실행되었을 경우
// "우편번호 검색" 버튼을 생성
// Javascript가 실행되지 않았을 경우
// 우편번호 검색 버튼을 생성시키지 않고 사용자가 입력하게 함
  var PostSearch_Type1 = new fnPostSearch_Type1;
  PostSearch_Type1.DivName = "JoinPostJsApply";

```

```

        PostSearch_Type1.Start();
    </script>

</body>

```

우편 번호를 검색하여 우편번호를 자동 입력하고 주소를 입력하는 기능은 대부분의 회원 입력창에서 구현하고 있으므로 이 부분은 생략하도록 한다.

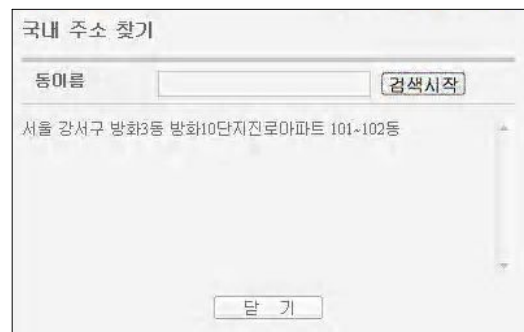


그림 II - 16. 우편 번호 검색 예제

아래의 JavaScript는 입력창의 입력 정보에 대한 유효성을 검사하기 위한 함수와 우편 번호 검색 버튼을 JavaScript로 생성하는 부분이다. 앞에서 이야기한 대로 우편번호 검색 버튼은 JavaScript 기능이 지원되지 않는 브라우저에서는 생성되지 않는다.

2) 스크립트 코드 부분

O (Good)

```

// JavaScript Document

function submitForum(formEl) {

```

```

var errorMessage = null;
var objFocus = null;

if ( formEl.JoinID.value.length == 0 ) {
    errorMessage = "아이디를 넣어 주세요";
    objFocus = formEl.JoinID;
} else if ( formEl.JoinPass.value.length == 0 ) {
    errorMessage = "비밀번호를 넣어 주세요";
    objFocus = formEl.JoinPass;
} else if ( formEl.JoinPassRe.value.length == 0 ) {
    errorMessage = "비밀번호 확인을 넣어 주세요";
    objFocus = formEl.JoinPassRe;
} else if ( formEl.JoinPass.value != formEl.JoinPassRe.value ) {
    errorMessage = "비밀번호와 비밀번호 확인이 다릅니다.  
확인해 보시기 바랍니다.";
    objFocus = formEl.JoinPass;
} else if ( formEl.JoinName.value.length == 0 ) {
    errorMessage = "이름을 넣어 주세요";
    objFocus = formEl.JoinName;
} else if ( formEl.JoinPost1.value.length == 0 ) {
    errorMessage = "우편번호 앞자리";
    objFocus = formEl.JoinPost1;
} else if ( formEl.JoinPost2.value.length == 0 ) {
    errorMessage = "우편번호 뒷자리";
    objFocus = formEl.JoinPost2;
} else if ( formEl.JoinAddress.value.length == 0 ) {
    errorMessage = "주소";
    objFocus = formEl.JoinAddress;
} else if ( formEl.BoardEmail1.value.length == 0 ) {
    errorMessage = "이메일 아이디를 넣어 주세요";
    objFocus = formEl.BoardEmail1;
} else if ( formEl.BoardEmail2.value.length == 0 ) {
    errorMessage = "이메일 서비스 회사의 주소를 넣어주세요";
    objFocus = formEl.BoardEmail2;
}

```

```

    }

    if ( errorMessage != null ) {
        alert(errorMessage);
        objFocus.focus();
        return false;
    }

    return true;
}

function postForum() {
    window.open('./popup.html','PostSearch','width=400,height
        =250,scrollbars=no,resizable=no');
}

function postApply(JoinPost1,JoinPost2,JoinAddress) {
    opener.document.getElementById("JoinPost1").value = JoinPost1;
    opener.document.getElementById("JoinPost2").value = JoinPost2;
    opener.document.getElementById("JoinAddress").value
        = JoinAddress;

    self.close();
}

// Javascript가 실행되었을 경우 "우편번호 검색" 버튼을 생성 하는 부분
function fnPostSearch_Type1() {
    this.Start = function() {
        this.TargetObj = document.getElementById(this.DivName)
            .getElementsByTagName("td")[0].getElementsByTagName("p")[0];
        this.PostBtn = document.createElement('a');
        this.PostBtn.href = "http://./postBtn.html";
        this.PostBtnImg = document.createElement('img');
        this.PostBtnImg.src = "./img/join_post.gif";
        this.PostBtnImg.alt = "우편번호 검색(팝업)";
    }
}

```

```

        this.PostBtnImg.className = "joinpost";

        this.TargetObj.appendChild(this.PostBtn);

        this.PostBtn.onclick = function() {
            postForum();
            return false;
        }
        this.PostBtn.appendChild(this.PostBtnImg);
    }
}

```

5.2.3 게시판 기능

<그림 II - 17>에 보인 게시판도 전형적인 온라인 서식으로, 앞에 기술한 로그인 창과 회원가입의 경우와 동일하므로 별도의 설명은 생략한다.

그림 II - 17. 게시판 예제

5.3 효과적인 UI 제작기법

5.3.1 롤링 배너

<그림 II - 18>에 보인 롤링배너는 좌우 또는 상하로 배너를 순차적으로 교체하는 컨트롤이다. 마우스를 사용할 경우에는 좌우 또는 상하 이동 버튼을 클릭하여 사용한다. Tab 키를 이용할 경우에는 배너를 하나씩 순서대로 이동하므로 이동 버튼을 사용하지 않는다.



그림 II - 18. 롤링 배너 예제

그러므로 Tab 키에 의하여 이동 버튼을 이동하지 않도록 해야 한다. 따라서 이동 버튼을 이미지로 바꾸고, 마우스 포인터가 위치할 경우에는 포인터의 형태를 변경하도록 함으로써 동일한 효과를 내게 한다. 또한 이동 버튼의 대체 텍스트를 `null(alt="")`로 제공하여 그것을 읽지 않도록 한다. 프로그램에서는 이 부분을 밑줄로 표시하였다.

Shift +Tab 키에 의한 이동 방향은 Tab 키의 역순이다. 또한 세로 롤링 배너의 코딩 방법도 가로 롤링 배너의 경우와 동일하므로 따로 설명하지 않았다.

또한 이 예제에서는 이동과 관련한 기능을 JavaScript로 작성하여 JavaScript를 지원하지 않는 브라우저에서는 모든 배너들이 화면에 보이도록 함으로써 접근성을 향상시킨다. 화면 낭독 프로그램은 Tab 키에 의한 이동결과를 읽어주어야 한다.

참고

- 센스/드림/JAWS : 읽어주는 방법이 화면에 나타나는 순서와 정확히 일치한다. 때때로 화면에서 초점이 사라지는 현상이 나타나기도 한다.

1) HTML 코드 부분

O (Good)

```
//HTML Document

<body>

<div id="BannerBox">
    <div id="BannerList">
        <div id="BannerListSub">
            <ul>
                <li><a href="./정보화마을.html">
                    
                </a></li>
                <li><a href="./주민서비스.html">
                    
                </a></li>
                <li><a href="./전자민원.html">
                    
                </a></li>
                .
                .
                <li><a href="./국민신문고.html">
                    
                </a></li>
                <li><a href="./청와대.html">
                    
                </a></li>
```

```

        </ul>
    </div>
</div>
<div id="BannerListMore">
    <a href="./banner.html"></a>
</div>
</div>

<script type="text/javascript">
    var fnRolBanner = new fnRolMove_Type1;
    fnRolBanner.BoxName = "BannerBox";
    fnRolBanner.DivName = "BannerListSub";
    fnRolBanner.fnName = "fnRolBanner";
    fnRolBanner.DateWidth = 140;

    fnRolBanner.Speed = 0.2;
    fnRolBanner.Start();
</script>
</body>

```

2) 스크립트 코드 부분

O (Good)

```

// JavaScript Document

function fnRolMove_Type1() {
    this.GoodsSetTime = null;
    this.BannerCurrent = 0;
    this.DateNum = 6;

    this.Start = function() {
        this.Obj = document.getElementById(this.BoxName);
        this.ObjBox = document.getElementById(this.DivName);
    }
}

```

```

this.ObjUI = this.ObjBox.getElementsByTagName("ul")[0];
this.ObjLi = this.ObjUI.getElementsByTagName("li");

// 배너 갯수를 파악하는 부분
this.ObjLiNum = this.ObjLi.length;

// 배너 부분의 총 넓이를 파악하는 부분
this.TotalWidth = this.DateWidth * this.ObjLiNum;

this.ObjBox.style.width = this.TotalWidth + "px";

if ( this.ObjLiNum % this.DateNum == 0 ) {
    this.BannerEnd
        = this.TotalWidth - ( this.DateWidth * this.DateNum );
} else {
    this.BannerEnd = this.TotalWidth - this.DateWidth;
}

// 배너를 쌓고 있는 부분의 박스에 CSS를 입히기
this._Style();
// Javascript 작동시 다음, 이전 버튼을 삽입하기
this._ControlAdd();

// 다음, 이전 버튼을 클릭시 이동해야 할 위치 계산하기
this.BannerPrevLeft = this.BannerEnd;
this.BannerNextLeft = this.DateWidth * this.DateNum ;

this.PrevBtnLinkImg.Num = this.BannerPrevLeft;
this.PrevBtnLinkImg.onclick = function() {
    eval(this.fnName + "._moveFrame(" + this.Num + ",'prev')");
    return false;
}

```

```

this.NextBtnLinkImg.Num = this.BannerNextLeft;
this.NextBtnLinkImg.onclick = function() {
    eval(this.fnName + "._moveFrame
        (" + this.Num + ",'next')");
    return false;
}
}

this._ControlAdd = function() {
this.NewControl = document.createElement('div');
this.NewControl.id = "BannerListCon";
this.Obj.appendChild(this.NewControl);

    //이전보기 버튼으로 구현한 경우
    //this.PrevBtnLink = document.createElement('a');
    //this.PrevBtnLink.fnName = this.fnName;
    //this.PrevBtnLink.href = "#";
    //this.NewControl.appendChild(this.PrevBtnLink);

    //이전보기 버튼을 가상 이미지 버튼으로 만든 경우.
    this.PrevBtnLinkImg = document.createElement('img');
    this.PrevBtnLinkImg.fnName = this.fnName;
    this.PrevBtnLinkImg.className = "btn";
    this.PrevBtnLinkImg.src = "./img/btn_prev.gif";
    this.PrevBtnLinkImg.alt = ""; //대체 텍스트를 null로 만든다
    this.NewControl.appendChild(this.PrevBtnLinkImg);

    //다음보기 버튼으로 구현한 경우
    //this.NextBtnLink = document.createElement('a');
    //this.NextBtnLink.fnName = this.fnName;
    //this.NextBtnLink.href = "#";
    //this.NewControl.appendChild(this.NextBtnLink);

    //다음보기 버튼을 가상 이미지 버튼으로 만든 경우.

```

```

        this.NextBtnLinkImg = document.createElement('img');
        this.NextBtnLinkImg.fnName = this.fnName;
        this.NextBtnLinkImg.className = "btn";
        this.NextBtnLinkImg.src = "./img/btn_next.gif";
        this.NextBtnLinkImg.alt = ""; //대체 텍스트를 null로 만든다
        this.NewControl.appendChild(this.NextBtnLinkImg);
    }

    this._Style = function() {
        this.BoxStyle = this.ObjBox.parentNode;
        this.BoxStyle.className = "BannerListStyle";
        this.BoxUIStyle
            = this.ObjBox.getElementsByTagName("ul")[0];
        this.BoxUIStyle.className = "BannerUIStyle";
    }

    // 다음, 이전 버튼을 클릭시 배너를 이동시키는 부분
    this._moveFrame = function(val,fnmove) {
        clearTimeout(this.GoodsSetTime);

        if ( Math.abs(val - this.BannerCurrent) > 5 ) {
            this.BannerCurrent
            = this.BannerCurrent+(val-this.BannerCurrent) * this.Speed;
        } else {
            this.BannerCurrent = val;
        }

        this.ObjUl.style.left = ( -1 * this.BannerCurrent ) + "px";

        if ( this.BannerCurrent != val ) {
            this.GoodsSetTime = setTimeout(this.fnName
                + "_moveFrame(" + val + "," + fnmove + ")",10);
        } else {
            this.CurrentPicNum = this.BannerCurrent / this.DateWidth;

```

```

        this.BannerPrevLeft
        = this.BannerCurrent - ( this.DateWidth * this.DateNum );
        this.BannerNextLeft
        = this.BannerCurrent + ( this.DateWidth * this.DateNum );

        if ( this.BannerCurrent == 0 ) {
            this.BannerPrevLeft = this.BannerEnd;
        } else if ( this.BannerCurrent == this.TotalWidth
                    - ( this.DateWidth * this.DateNum ) ) {
            this.BannerNextLeft = 0;
        }
        this.PrevBtnLinkImg.Num = this.BannerPrevLeft;
        this.PrevBtnLinkImg.onclick = function() {
            eval(this.fnName+"._moveFrame("+this.Num+", 'prev')");
            return false;
        }

        this.NextBtnLinkImg.Num = this.BannerNextLeft;
        this.NextBtnLinkImg.onclick = function() {
            eval(this.fnName+"._moveFrame("+this.Num+", 'next')");
            return false;
        }
    }
}

```

5.3.2 스크롤 배너

스크롤 배너란 아래 <그림 II - 19>와 같이 마우스 포인터를 배너에 위치시키면 해당 배너가 전면에 나타나고, 클릭하는 경우에는 해당 링크로 이동하는 컨트롤이다.



그림 II - 19. 스크롤 배너 예제

스크롤 배너는 Tab 키를 누를 때마다 순방향으로 초점이 이동하면서 해당 배너가 전면에 나타난다. Shift +Tab 키에 의해서는 Tab 키의 역순으로 이동한다. 화면 낭독 프로그램은 초점이 주어지는 배너의 내용을 읽어준다.

참고

- 센스/드림/JAWS : 정상적으로 동작한다.

여기서 주의할 점은 이미지의 대체 텍스트를 너무 장황하지 않게 작성하는 것이다. 배너를 통해서 많은 내용을 전달하려고 애쓰기 보다는 핵심적인 내용을 전달해야 하기 때문이다. 또한 설명이 필요한 경우에는 title 을 사용하는 것이 바람직하다.

1) HTML 코드 부분

O (Good)

```
//HTML Document
```

```
<body>
```

```
<div id="Banner_Box">
```



```

<ul>
  <li><a href="/질의응답.html">질의응답을 신청할 수 있는 공간</a></li>
  <li><a href="/나의질의.html">나의질의조회-담당자의 답변을 확인하실
    수 있는 공간 </a></li>
  <li><a href="/FAQ.html">고객센터 FAQ - 유사질문에 대한 답변확인
    공간 </a></li>
  <li><a href="/고객상담.html">고객상담안내 - 고객의 상담을 위한 안내
    공간 </a></li>
  <li><a href="/국민제안.html">국민제안 - 행정정책과 제도에 대한 의견
    제안 공간 </a></li>
  <li><a href="/여론광장.html">여론광장 </a></li>
</ul>
</div>

<script type="text/javascript">
  var BannerName = "BannerScroll";
  var BannerBox = "Banner_Box";
  var BannerWidth_S = 54;
  var BannerWidth_B = 154;
  var BannerScroll = new BannerOver();
</script>

</body>

```

2) 스크립트 코드 부분

O (Good)

```

// JavaScript Document

function BannerOver() {
  this.BannerWrap = document.getElementById(BannerBox);
  this.BannerList =
    this.BannerWrap.getElementsByTagName("ul")[0]

```

```

        .getElementsByTagName("li");
this.BannerTotal = this.BannerList.length;
this.BannerLocation = new Array();
this.BannerTarget = new Array();
this.BannerTempWidth;

for ( var i=0; i<this.BannerTotal; i++ ) {
    this.BannerLocation[i] = BannerWidth_S*i;
    this.BannerList.item(i).style.left
        = this.BannerLocation[i] + "px";

    this.BannerLink
        = this.BannerList.item(i)
        .getElementsByTagName("a")[0];
    this.BannerLink.Num = i;
    this.BannerLink.onmouseover
        = this.BannerLink.onfocus = function() {
        eval(BannerName+".BannerMove("+this.Num+")");
    }
}

this.BannerRandom
    = Math.floor(Math.random()*this.BannerTotal);
this.BannerMove(this.BannerRandom);
}
BannerOver.prototype.BannerMove = function(val) {
    this.BannerTargetSetting(val);
    this.BannerLiMove(val);
}
BannerOver.prototype.BannerTargetSetting = function(val) {
    for ( var i=0; i<this.BannerTotal; i++ ) {
        this.BannerLiw = this.BannerList.item(i);
        this.BannerImg
            = this.BannerLiw.getElementsByTagName("a")[0];

```

```

        if ( (val+1) == i ) { this.BannerTempWidth
            = BannerWidth_B }
        else { this.BannerTempWidth = BannerWidth_S }
        if ( i == 0 ) {
            this.BannerTarget[i] = 0;
        } else {
            this.BannerTarget[i]
            = this.BannerTarget[i-1]+this.BannerTempWidth;
        }
        if ( val == i ) {
            this.BannerLiw.className = "widthb";
            this.BannerImg.className = "line_" + i + "_on";
        } else {
            this.BannerLiw.className = "widths";
            this.BannerImg.className = "line_" + i + "_off";
        }
    }
}

BannerOver.prototype.BannerLiMove = function(val) {
    for ( var i=0; i<this.BannerTotal; i++ ) {
        this.BannerLocation[i] +=
            (this.BannerTarget[i] - this.BannerLocation[i]) * 0.2;
        if (Math.abs((this.BannerTarget[i]-this.BannerLocation[i]))< 2)
        {
            this.BannerTarget[i] = this.BannerTarget[i];
            this.BannerList.item(i).style.left = this.BannerTarget[i] + "px";
        } else {
            this.BannerList.item(i).style.left
            = parseInt(this.BannerLocation[i],10) + "px";
            setTimeout(BannerName+".BannerLiMove("+val+)",100);
        }
    }
}
}

```

5.4 Javascript 대체 기법

5.4.1 배경 이미지 변경

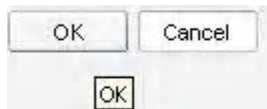
(1) JavaScript를 사용한 경우

아래 <그림 II - 20>은 JavaScript를 이용하여 초점이 있는가 여부에 따라 이미지 버튼('OK')과 텍스트 버튼('Cancel')의 배경 이미지를 변경하는 예이다.

초점이 주어짐에 따라 버튼('OK')의 배경 이미지가 <그림 II - 20(a)>에서 <그림 II - 20(b)>로 변화함을 알 수 있다.



(a) 'OK'버튼에 마우스
롤오버를 하기 전



(b) 'OK'버튼에 마우스
롤오버를 한 후

그림 II - 20. 배경 이미지를 변경하여 롤오버 버튼을 구성한 예제

1) 스크립트 코드 부분

O (Good)

```
<script type="text/javascript">
<!--
var on = function(e)    { e.getElementsByTagName("img")[0].src
= e.getElementsByTagName("img")[0].src.replace(".gif", "_on.gif"); }
var out = function(e) { e.getElementsByTagName("img")[0].src
= e.getElementsByTagName("img")[0].src.replace("_on.gif", ".gif"); }
//-->
</script>
```

2) HTML 코드 부분

O (Good)

```
<body>

<a href="#" onmouseover="on(this)" onmouseout="out(this)">
  </a>
<a href="#" onmouseover="on(this)" onmouseout="out(this)">
  </a>

</body>
```

(2) CSS로만 구성한 경우

다음 코드는 (1)에서 보인 프로그램을 JavaScript를 사용하지 않고 CSS로만 작성한 것이다.

O (Good)

```
<head>
<style type="text/css" title="">
/*
    Rollover, out
    CSS에서 A 태그의 배경그림을 위치를 바꿔 롤오버 효과를 구현
*/
a.bg-button {
    display: block;
    width: 60px;
    height: 21px;
    text-align: center;
    color: black;
    text-decoration: none;
    font-size: 11px;
    font-family: arial;
    line-height: 22px;
```

```

        background: url("image_pack.gif");

        float: left;
        margin-right: 5px;
    }
    a.bg-button:hover      { background-position: 0 21px; }
    a.bg-button:active     { background-position: 0 42px; }
</style>
</head>

<body>
<h1>CSS만을 이용해서 Rollover, out, click 되었을 때를 표현</h1>
<a href="http://../ok.html" class="bg-button">OK</a>
<a href="http://../cancel.html" class="bg-button">Cancel</a>
</body>

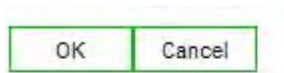
```

5.4.2 스타일 변경

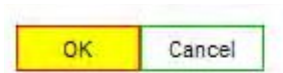
(1) JavaScript를 사용한 경우

아래 <그림 II - 21>은 초점이 주어진 버튼의 색깔을 class값('OK' 버튼)과 style('Cancel' 버튼)을 이용하여 변경하는 예이다.

마우스 롤오버(mouse rollover)를 함에 따라 'OK' 버튼과 'Cancel' 버튼이 노란색으로 변화하는 것을 알 수 있다.



(a) 'OK'버튼에 마우스
롤오버를 하기 전



(b) 'OK'버튼에 마우스
롤오버를 한 후

그림 II - 21. 스타일을 변경하여 롤오버 버튼을 구성한 예제

<그림 II - 21>을 JavaScript로 작성한 예제는 다음 프로그램과 같다.

1) CSS 코드 부분

O (Good)

```
<head>
<link type="text/css" href="reset.css" rel="stylesheet"/>
<style type="text/css" title="">
a {
    float: left;
}

a.bg-button {
    display: block;
    width: 60px;
    height: 21px;
    text-align: center;
    color: black;
    text-decoration: none;
    font-size: 11px;
    font-family: arial;
    line-height: 22px;

    background: url("ok_bg.gif");
```

```
float: left;
}
</style>
</head>
```

2) 스크립트 코드 부분

O (Good)

```
<script type="text/javascript">
<!--
var bindChangeClassName = function() {
    var selected_class_name = "hover";

    var as = document.getElementsByTagName("a");
    for (var i=0; i<as.length; i++) {
        var a = as[i];

        if (a.className == "class") {
            a.onmouseover = a.onfocus = function(event) {
                this.className += " " + selected_class_name; }
            a.onmouseout = a.onblur = function(event) {
                this.className = this.className
                    .replace(selected_class_name, "");}
        }
    }
}

var bindChangeStyle = function() {
    var as = document.getElementsByTagName("a");
    for (var i=0; i<as.length; i++) {
        var a = as[i];

        if (a.className == "style") {
```



```

        a.onmouseover = a.onfocus = function(event) {
            this.style.borderColor = "red";
            this.style.backgroundColor = "yellow";
        }
        a.onmouseout = a.onblur = function(event) {
            this.style.borderColor = "#00cc00";
            this.style.backgroundColor = "white";
        }
    }
}
window.onload = function() {
    bindChangeClassName();
    bindChangeStyle();
}
//-->
</script>

```

3) HTML 코드 부분

O (Good)

```

<body>
<a href="#" onmouseover="on_1(this)" onmouseout="out_1(this)">
    </a>
<a href="#" class="bg-button">Cancel</a>
</body>

```

(2) CSS로만 구성한 경우

다음 코드는 JavaScript를 사용한 경우를 CSS로 구현한 것이다. 버튼의 기능이 동일하게 동작하는 것을 볼 수 있다.

1) CSS 및 HTML 코드 부분(전체)

O (Good)

```
<link type="text/css" href="reset.css" rel="stylesheet"/>
<style type="text/css" title="">

/* 배경 색을 바꿔서 롤오버 효과 구현 */
a.ss-button {
    display: block;
    width: 60px;
    height: 21px;
    text-align: center;
    color: black;
    text-decoration: none;
    font-size: 11px;
    font-family: arial;
    line-height: 22px;
    border: 1px solid #00CC00;
    float: left;
}
a.ss-button:hover      { border-color: red; background-color: yellow; }
a.ss-button:active     { border-color: red; background-color: orange; }
}

/* position */
.position {
    clear: both;
}
.position ul {
    margin: 0;
    padding: 0;
}
.position li {
    float: left;
```

```

        margin: 0;
        padding: 0;
        list-style: none;
    }
    .position .menu a {
        display: block;
        height: 21px;
        width: 100px;
        text-align: center;
        color: black;
        text-decoration: none;
        font-size: 11px;
        font-family: arial;
        line-height: 22px;
        background-color: yellow;
        border: 1px solid blue;
    }
    .position .menu a:hover { border-color: red; background-color: yellow; }
    .position .menu a:active { border-color: red; background-color: orange; }
    }
    .position .menu .sub {
        position: absolute;
        display: none;
    }
    .position .menu .sub a {
        background-color: #CCFFFF;
    }
</style>
</head>

<body>
<div class="section">
    <div class="section">
        <dl>

```

```

        <dd>
            <a href="#" class="ss-button">OK</a>
            <a href="#" class="ss-button">Cancel</a>
        </dd>
    </dl>
</div>
</div>
</body>

```

5.4.3 탭 메뉴의 대체

아래 예제는 가로 탭 메뉴로써, JavaScript 기능이 동작하는 환경에서는 선택된 탭 메뉴에 해당하는 하위 목록을 보여준다. 그러나 JavaScript를 지원하지 않는 브라우저에서는 모든 탭 메뉴의 하위 목록을 화면에 보여준다. 아래 <그림 II - 22>는 JavaScript를 지원하는 브라우저의 화면 모습으로 탭 메뉴 중 'Menu 03'이 선택된 상태를 보여주고 있다.



그림 II - 22. JavaScript 기능이 동작하는 환경에서 탭 메뉴의 화면

주 메뉴 간의 이동은 Tab 키를 이용할 수 있다. 즉 Tab 키를 연속해서 누르면, 'Menu 01', 'Menu 02', 'Menu 03'의 순서로 이동하고, 다시 Tab 키를 누르면 아래의 'Top' 버튼으로 초점이 이동한다.

주 메뉴에 초점이 주어졌을 때 Enter 키를 누르면 해당 탭 메뉴의 하위 목록이 펼쳐진다. 이어서 Tab 키를 누르면 다음 주 메뉴로 이동한다.

마지막 주 메뉴로 이동한 후에는 화면에 보인 탭 메뉴의 마지막 부분에 있는 'Top' 버튼으로 초점이 이동하며, 버튼을 클릭(Enter 키를 누르면)하면 다시 첫 번째 탭 메뉴로 이동한다.

화면 낭독 프로그램이 실행되어도 초점이 이동함에 따라 그 내용을 읽어주어야 한다.

만일 브라우저가 JavaScript를 지원하지 않을 경우에는 다음 <그림 II - 23>과 같이 모든 하위 메뉴가 화면에 순서대로 나타나게 된다.

이때 Tab 키를 누르면, 초점이 주 메뉴인 'Menu 01', 'Menu 02', 'Menu 03'의 순서로 이동하고, 다시 Tab 키를 누르면 아래의 첫 번째 'Top' 버튼으로부터 마지막 세 번째 버튼으로 초점이 순차적으로 이동한다.

주 메뉴에 초점이 주어졌을 때 Enter 키를 누르면 해당 탭 메뉴의 콘텐츠로 화면이 이동한다. 이어서 Tab 키를 누르면 해당 하위목록의 마지막에 있는 'Top' 버튼으로 초점이 이동한다. 여기서 버튼을 클릭(Enter 키를 누르면)하면 다시 첫 번째 탭 메뉴로 이동한다.



그림 II - 23. JavaScript 기능이 동작하지 않는 환경에서의 탭 메뉴의 화면

참고

- 센스/드림/JAWS : 정상적으로 동작한다.

아래에 보인 코드 자체는 평이하므로 설명을 생략한다. JavaScript 부분을 삭제하거나 브라우저가 지원하지 않으면 CSS와 HTML 코드만으로도 동작한다.

1) CSS 코드 부분

O (Good)

<pre> <style type="text/css" title=""> body { font: 12px arial; } a { text-decoration: none; } img { border: 0; } /* tab menu */ #tab_menus { margin: 0; padding: 0; list-style: none; } #tab_menus li { margin-bottom: 16px; float: left; padding: 0px; } #tab_menus li a { padding: 8px 10px; color: #505050; border-width: 1px 0 1px 1px; border-style: solid; border-color: #ae9f96; background-color: #eae4e0; background-image: none; </pre>	<pre> #tab_menus li a:hover { color: black; background-color: #c6b5aa; } #tab_menus li.selected a { border-bottom: 0 solid white; background: white url("blt_arrowdown.gif") no-repeat center bottom; color: black; } .tab_right { margin-bottom: 16px; border-width: 0 0 1px 1px; border-style: solid; border-color: #ae9f96; width: 80px; height: 31px; float: left; } /* content */ .content { clear: both; padding-left: 10px; } .content h3 { margin: 0; padding: 0; font-size: 1.2em; </pre>
---	---

<pre> font-weight: bold; font-size: 0.9em; display: block; } /* 우측에 계속 */ </pre>	<pre> } .content h3 a { color: #ff4800; } </style> </pre>
--	---

2) 스크립트 코드 부분

O (Good)

```

<script type="text/javascript">
<!--
var selected_menu = 0;
var content_count = 0;

var selected_class_name = "selected";

var menus
    = document.getElementById("tab_menus").getElementsByTagName("li");
var dives = document.getElementsByTagName("div");
var contents = new Array;

for (var i=0; i<dives.length; i++) {
    var div = dives[i];

    if (div.className == "content") {

        // Javascript를 이용하여 CSS초기화
        if (selected_menu == content_count) {
            div.style.display = "block";
            menus[content_count].className += " " +
                selected_class_name;
        }
    }
}

```



```

        else {
            div.style.display = "none";
            menus[content_count].className
                = menus[content_count]
                    .className.replace(selected_class_name, "");
        }
        // -->

        contents.push(div);
        content_count++;
    }
}

for (var i=0; i<menus.length; i++) {
    var quick_link = menus[i];
    quick_link.content_index = i;

    quick_link.onclick = quick_link.onkeypress = function(event) {
        contents[selected_menu].style.display = "none";
        menus[selected_menu].className
            = menus[selected_menu].className.replace
                (selected_class_name, "");
        contents[this.content_index].style.display = "block";
        this.className += " " + selected_class_name;
        selected_menu = this.content_index;
        return false;
    }
}
//-->
</script>

```

3) HTML 코드 부분

O (Good)

```
<body>
<menu id="tab_menus">
  <li><a href="#QuickMenu_01">Menu 01</a></li>
  <li><a href="#QuickMenu_02">Menu 02</a></li>
  <li><a href="#QuickMenu_03">Menu 03</a></li>
</menu>
<div class="tab_right"></div>

<div id="Menu_01" class="content">
  <h3><a id="QuickMenu_01" href="#QuickMenu_01">Menu 01
  </a></h3>
  <p>
    Menu 01...<br />
    Menu 01...<br />
    ...
    Menu 01...<br />
    Menu 01...<br />
  </p>
  <a href="#tab_menus" class="go_tab_menu">
    </a>
</div>
<div id="Menu_02" class="content">
  <h3><a id="QuickMenu_02" href="#QuickMenu_02">Menu 02
  </a></h3>
  <p>
    Menu 02...<br />
    Menu 02...<br />
    ...
    Menu 02...<br />
    Menu 02...<br />
  </p>
</div>
```