

02

CHAPTER

HTML5 마크업

- 01 마크업 기본 규칙
- 02 더 시맨틱하게…
- 03 무엇이든 그려내는 Canvas 요소
- 04 SVG
- 05 쉽고 강력해진 HTML5 웹 품
- 06 액티브엑스가 필요 없는 내장 미디어
- 07 구형 익스플로러를 위한 대비책

HTML5에는 새로운 마크업 요소가 많이 추가되었습니다. 기존에는 단순히 UL이나 DIV로 페이지의 각 영역을 나누었던 것에 반해 헤더와 푸터, 사이드바 그리고 콘텐츠 영역의 아티클까지도 각자의 역할을 가진 요소들로 관리할 수 있게 되었습니다. 반면에 없어진 요소도 많습니다. 웹페이지에 있는 텍스트나 테이블, 이미지 등을 꾸며 주기 위한 <center>, <big>, <u>와 같은 요소나 <align>, <bgcolor> 등의 속성은 과감하게 HTML5에서 사라질 것입니다.

이는 HTML에는 웹페이지의 데이터만을 담고 디자인이나 스타일과 관련된 내용은 전적으로 CSS에서 처리하도록 만들겠다는 W3C의 확고한 의지를 보여 줍니다.

새로이 추가된 마크업 요소 중 특히 눈에 띠는 요소는 <canvas>나 <video>, <audio>와 같은 멀티미디어 요소입니다. 특히 <canvas>는 외부 플러그인 없이 스크립트만을 이용해서 웹페이지에서 바로 그래픽을 구현할 수 있습니다. 2D, 3D 그리고 더 강력한 엔진을 동원한다면 높은 퀄리티의 3D 이미지도 웹페이지에서 바로 구현할 수 있습니다.

자, 그럼 HTML5 마크업의 세계로 떠나 볼까요?

01 마크업 기본 규칙

기존에 마크업을 잘 하던 사람들은 익히 알고 있는 기본적인 내용을 다루는 장입니다. 그러나 기본은 너무나 중요하기 때문에 HTML 마크업의 개념에 대해서 잘 모르는 사람들은 물론 이미 잘 아는 사람들도 한 번은 짚고 넘어가야 하는 장입니다. HTML5 마크업 규칙의 기본 뼈대는 이전 버전의 HTML이나 XHTML 문서의 것과 크게 다르지 않습니다. 다만 바뀌거나 추가된 부분 그리고 꼭 숙지해야 하는 부분이 있습니다.

HTML5는 HTML4 버전대의 문법도 호환하고 XHTML1 버전대의 문법도 호환합니다. 앞서 1장에서 HTML5를 소개할 때 언급한 부분입니다. 따라서 HTML 문법에 익숙한 사람들은 HTML 방식으로 마크업 코딩을 하면 되고 XHTML 문법에 익숙한 사람들은 XHTML 방식으로 마크업을 하면 됩니다.

필자는 개인적으로 꽤 오랫동안 다양한 프로젝트들을 해 오면서 XHTML 문서를 주로 사용해 왔습니다. 따라서 HTML4.x 버전의 문법은 따로 다루지 않겠습니다. XHTML 문서의 작성 규칙을 중심으로 HTML5의 기본 문법과 마크업 규칙을 공부해 보겠습니다.

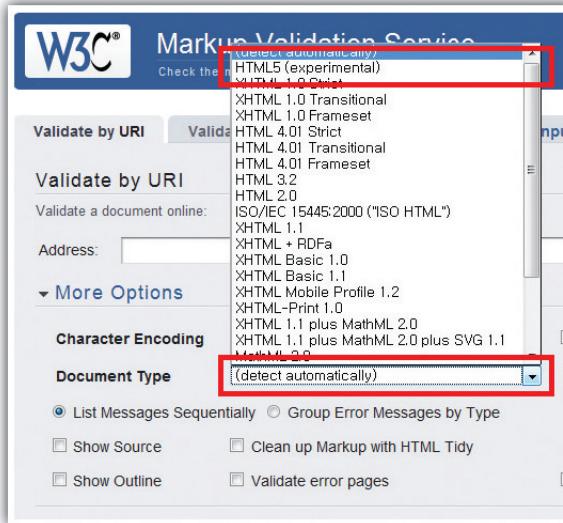
HTML5 기본 코딩 규칙과 문법

HTML5 문서를 잘 만들려면 아주 기초적인 규칙부터 잘 지켜야 합니다. 규칙을 잘 지켜서 잘 만들어진 HTML5 문서는 여러 가지 측면에서 유용합니다. 우선 만들어진 웹페이지는 앞으로도 몇 년간 PC에서는 물론이고 다양한 기기에서 읽힐 것입니다. 표준을 준수하여 만든 웹페이지라면 지금 당장이라도 다양한 기기에서 그 문서를 읽을 수 있습니다.

또한 CSS를 이용하여 웹페이지를 스타일링할 때도 잘 만들어진 HTML5 문서일 수록 스타일링하기가 수월합니다. 물론 관리하기도 좋습니다.

기본적인 HTML5 작성 규칙을 잘 지켰는지 확인하고 싶다면 W3C의 유효성 검

사 서비스를 이용하면 됩니다. W3C에서는 HTML5의 유효성 검사 기능도 제공하고 있으므로 지금 당장이라도 이용해 볼 수 있습니다.



☞ 그림 2.1 W3C 마크업 유효성 검사 서비스입니다. Document Type 항목에서 HTML5를 선택하면 HTML5 마크업이 규칙에 맞게 만들어졌는지 유효성 검사를 할 수 있습니다.

XHTML 방식의 마크업, 몇 가지 규칙

앞서도 잠시 말하였지만 필자는 XHTML 방식의 문서작성을 선호해 왔기 때문에 XHTML 방식으로 문서를 코딩할 때 지켜야 할 기본적인 몇 가지 규칙을 설명하겠습니다.

- ① 짹이 있는 태그는 열리는 태그가 있으면 반드시 닫아 줍니다.

<p>p태그는 열리는 태그와 닫히는 태그 짹이 있습니다.</p>

그러나 짹이 없는 태그는 HTML 4.x 방식으로 슬래시로 닫지 않아도 되고, XHTML 1.x 방식으로 슬래시로 닫아도 됩니다

이렇게 써도 되고
이렇게 써도 됩니다.

- ② 독타입을 잊지 말고 선언합니다. 책의 도입부부터 계속하여 이야기하고 있는 규칙입니다.

HTML5의 독타입은 간결합니다. HTML 문서의 최상단에 다음 코드 한 줄만 넣어 주면 됩니다.

```
<!DOCTYPE html>
```

- ③ 캐릭터셋을 설정합니다. 문서의 캐릭터셋 설정도 이전보다 간단해졌습니다.

HTML 문서의 <head>에 다음과 같은 메타 태그를 삽입합니다.

```
<meta charset="utf-8">
```

- ④ 열리고 닫히는 태그의 순서가 꼬이지 않도록 합니다. 다른 태그를 담고 있는 태그와 그 안에 담겨 있는 태그는 명확하게 자신의 위치를 지켜야 합니다. 무슨 말인지 다음 코드를 보면서 설명하겠습니다.

```
<div><strong>…</div></strong> --> (구조가 잘못되었음)
```

div 안에 strong 태그가 포함되도록 만들어야 하는데, div가 strong보다 앞서 서 닫혀 있습니다. 이렇게 되면 W3C 유효성 검사를 통과하지 못할 뿐 아니라 페이지가 심하게 왜곡되어 디자인상으로도 깨지고 의미상으로도 많은 문제가 발생할 수 있습니다. 반드시 각 태그가 작성된 구조에 맞도록 열고 닫는 것을 생활화하세요. 위의 코드를 정상적으로 고친다면 다음과 같이 될 것입니다.

```
<div><strong>…</strong></div>
```

- ⑤ 태그 이름은 대문자와 소문자를 가리지 않습니다. HTML5는 태그 이름을 사용할 때, 소문자를 사용하든 대문자를 사용하든 상관없습니다. 태그 이름으로는 원래 XHTML 문법에서는 소문자를 사용하고, HTML 문법에서는 대문자로 써야 하지만, HTML5에서는 양쪽 하위 버전 모두를 호환합니다.

- ⑥ 특수 기호는 엔티티코드를 사용합니다. 예를 들면 앰페샌드라고 불리는 ‘&’ 기호는 그대로 쓰지 말고 ‘&’라고 쓰세요. 저작권 부분을 나타낼 때 많이 사용하는 기호 ‘©’ 역시 ‘©’로 표기하세요.

엔티티 코드에 대한 더 많은 정보는 <http://www.entitycode.com/>에서 얻을 수 있습니다.

- ⑦ 인라인 태그 안에 블록레벨 태그를 담지 않습니다. <div>나 <p>와 같은 태그는 블록레벨 태그입니다. 블록레벨 태그는 다른 태그들을 담을 수 있습니다. 이나 <a>와 같은 태그는 인라인 레벨 태그입니다. 문법상 인라인 레벨의 태그 안에 블록레벨 태그가 들어가면 올바르지 않으므로 주의해서 사용합니다.
- ⑧ 확장태그 속성을 사용할 때 빈 값인 상태로 두면 안 됩니다. 예를 들어서 태그에 alt 태그를 사용할 때, 와 같이 사용하면 안 됩니다. 이렇게 하면 문법적으로도 문제가 있지만 해당 태그가 가지는 속성의 의미가 많이 희석됩니다. 확장태그에는 반드시 값이 들어갈 수 있도록 마크업하는 습관을 들입시다.
- ⑨ 태그에는 width와 height 값을 지정하고 alt와 title 속성을 사용합니다. 표준화 작업에 대한 실력이 많이 향상된 독자들은 대부분 이미지를 CSS로 처리할 것입니다. 그러나 만약 부득이 하게 HTML에서 바로 이미지를 사용해야 한다면 이미지의 높이와 너비를 지정해 주도록 합니다. 이것은 태그의 ‘디자인 속성은 HTML에서 사용하지 마라’는 것과는 약간 다릅니다. 이렇게 하는 이유가 있습니다. 높이와 너비를 미리 지정하면 페이지가 로드되는 속도가 조금 더 빨라집니다. 컴퓨터가 이미지의 너비와 높이를 따로 계산할 필요가 없기 때문입니다. 또한 이미지에는 반드시 alt 속성과 title 속성을 이용해서 이미지가 의미하는 바를 정확하게 명시하도록 해야 웹 접근성도 준수할 수 있습니다.



UTF-8을 사용하는 이유가 뭘까요?

어느 나라에서나 한글을 문제없이 표현하고 싶고 미래지향적인 웹페이지를 만들고 싶다면 문서의 캐릭터셋은 UTF-8로 설정할 것을 권장합니다. EUC-KR은 한글, 한자, 영어를 제외한 다른 나라의 언어를 표현하지 못합니다. 반면, UTF-8은 한글은 물론이고 세계의 다양한 언어를 문제 없이 표현할 수 있습니다. 한글이라고는 구경해 본 적도 없는 미국의 컴퓨터 아무 곳에서나 열어도 한글이 제대로 출력됩니다. 다만 UTF-8은 한글 한 자를 3바이트로 처리하여 문서 크기가 다소 커지지만, 공백이나 영문 한 자는 1바이트로 처리하는 유연함을 보여 줍니다. 캐릭터셋을 UTF-8로 사용한다면 HTML 파일을 저장할 때 파일 타입도 UTF-8로 저장해야 합니다.

닫는 태그를 사용하면 안 되는 태그

 태그나
, <meta>와 같은 태그들은 대표적인 홀태그입니다. ‘셀프 클로징’ 태그라고도 불리는 이 태그들은 짹이 없기 때문에 당연히 짹을 지어서 마크업하면 문법적으로 올바르지 않은 문서가 됩니다.

예제 2.1 짹이 없는 태그에 짹을 지어 준 틀린 마크업

```
</img>
```

이와 같이 하면 틀린 마크업이 됩니다. 과 같이 짹이 없는 홀태그들은 다음과 같은 방법으로 처리하면 됩니다.

예제 2.2 홀태그의 올바른 사용법

	XHTML 방식
	HTML 방식

위의 두 가지 방법대로 사용하면 됩니다. 윗줄의 XHTML 방식처럼 ‘/>’으로 마무리해도 되고, 기존 HTML4 방식처럼 그냥 닫는 태그 없이 사용해도 됩니다.

닫는 태그를 사용하면 안 되는 홀태그의 목록은 다음과 같습니다.

```
area, base, meta, img, br, embed, hr, param, source, keygen, link, input, command, col
```

HTML5 템플릿

앞으로는 HTML5를 이용해서 웹페이지를 만들 때 기본적인 문서 템플릿이 있어야 할 것입니다. HTML5의 문서 템플릿은 의외로 간결합니다. 앞서 1장에서 배운 독타입만 유의해서 사용하면 됩니다.

예제 2.3 HTML5 문서의 기본 뼈대

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset= "UTF-8">
```

```
<title>HTML5 문서 템플릿</title>
</head>
<body>
<!-- 여기 문서의 콘텐츠가 들어갑니다 -->
</body>
</html>
```

이것이 HTML5 기본 템플릿의 전부입니다. HTML5 독타입을 선언하고 캐릭터셋과 타이틀 태그로 문서의 타이틀을 정해 주면 그만입니다. <head> 태그와 <body> 태그로 문서의 머리 부분과 몸통 부분을 정확하게 나누어 HTML5 문서를 만들기 위한 작업 준비를 마칩니다.



타이틀 태그를 낭비하지 마세요

타이틀 태그는 SEO('검색엔진최적화') 레벨이 높습니다. 검색엔진의 봇(Bot)들이 웹페이지의 링크에 링크를 타고 돌아다니면서 웹페이지를 크롤링하여 수집합니다. 이렇게 수집된 페이지들은 이용자가 검색어를 넣고 검색을 하였을 때 특정한 알고리즘에 따라 순서대로 출력됩니다. 검색엔진에서 상위 랭킹에 출력되는 것은 사이트 방문객을 늘리는 데 매우 중요합니다. 이때 검색 순위를 매길 때 중요한 요소가 타이틀 태그와 헤드라인(H) 태그입니다. 특히 타이틀 태그는 중요도가 매우 높습니다. “저의 홈페이지에 오신 것을 환영합니다.”와 같은 타이틀을 사용하여 흥보 리소스를 낭비하지 마세요. 정확한 키워드를 타이틀 태그에 담으세요. 특히 타이틀 태그에는 단순히 서비스 이름만 넣지 말고 각 페이지의 주요 키워드가 변화무쌍하게 들어가도록 하면 검색엔진에 노출되는 빈도가 훨씬 높아집니다.

콘텐츠의 흐름, 블록레벨 요소와 인라인 요소

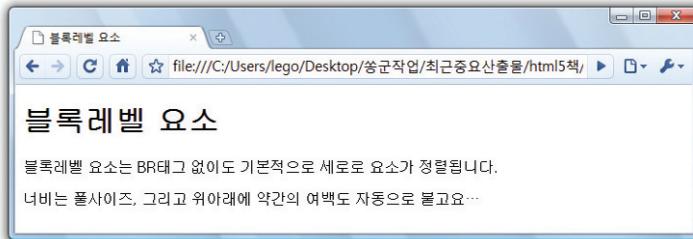
HTML5도 엄연히 하나의 문서입니다. 당연히 이 문서 안에는 글도 들어갈 수 있고 그림도 들어갈 수 있습니다. 글과 그림은 일반 워드프로세서의 문서처럼 나열되면서 흐르게 됩니다. 그러나 일반 워드프로세서와 다른 점이 있습니다. 그것은 바로 HTML 문서에는 블록레벨(block level) 요소와 인라인(inline) 요소라고 하는 두 개의 큰 흐름의 줄기가 있다는 것입니다.

블록레벨 요소

<div>나 <p>, <h1>과 같은 요소들은 블록레벨 요소입니다. 블록레벨 요소의 특징은 줄 바꿈 태그인
과 같은 요소를 쓰지 않아도 스스로 줄 바꿈이 된다는 데 있습니다. 이 요소들은 기본적으로 문서 내에서 가로로 흐르지 않고 세로로 흐릅니다. 그리고 주변에 일정량의 공간을 만듭니다. 또한 너비를 지정해 주지 않으면 폴사이즈가 되어 가로로 가득 찹니다. CSS를 적용하지 않고 이 요소들을 브라우저에서 확인해 보면 세로로 쭉 나열되는 것을 볼 수 있습니다. 블록레벨 요소는 가로, 세로 일정량의 여백을 가지고 있는데 이것은 브라우저마다 조금씩 다릅니다.

예제 2.4 블록레벨 요소의 예

```
<h1>블록레벨 요소</h1>
<p>블록레벨 요소는 BR태그 없이도 기본적으로 세로로 요소가 정렬됩니다.</p>
<div>너비는 폴사이즈, 그리고 위아래에 약간의 여백도 자동으로 붙고요…</div>
```



☞ 그림 2.2 스타일을 적용하지 않은 블록레벨 요소의 기본 흐름

스타일을 일체 적용하지 않고 이 코드를 화면으로 확인해 보면 이 그림과 같이 출력됩니다.

인라인 요소

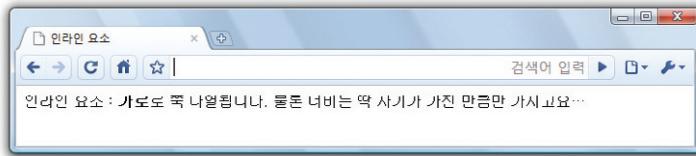
, , 과 같은 요소들은 인라인 요소입니다. 이 요소들의 특징은 줄 바꿈 태그를 사용하지 않거나 블록레벨 태그 안에 단독으로 포함되지 않는 한 가로로 쭉 나열된다는 데 있습니다. 그리고 이 요소들의 주변에는 공간이 생기지 않습니다.

문법적으로는 인라인 요소 안에 블록레벨 요소가 들어가면 안 됩니다. 블록레벨

요소 안에 인라인 요소가 포함되도록 마크업해야 문법을 바르게 준수할 수 있게 됩니다.

예제 2.5 인라인 레벨 태그의 예

```
<span>인라인 요소 : </span>
<strong>가로</strong>로 쭉 나열됩니다. 물론 너비는 딱 자기가 가진 만큼만 가지고요.
```

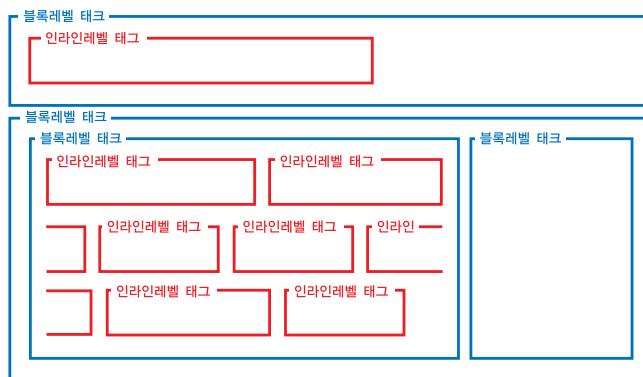


☞ 그림 2.3 스타일을 적용하지 않은 인라인 요소의 기본 흐름

스타일을 일체 적용하지 않고 이 코드를 화면으로 확인해 보면 이 그림과 같이 출력됩니다.

적절한 태그의 선택과 사용 방법

블록레벨 태그 안에는 인라인 레벨 태그가 들어갈 수 있습니다. 아니 들어가야만 하는 경우가 아주 많습니다. 그러나 문법적으로 인라인 레벨 태그는 블록레벨 태그를 담을 수 없습니다. 다음 그림은 아마도 인라인레벨 태그와 블록레벨 태그의 관계를 가장 잘 표현해 주는 다이어그램일 것입니다.



☞ 그림 2.4 블록레벨 태그와 인라인 레벨 태그를 쉽게 이해할 수 있는 구조도

인라인 레벨 태그는 기본적으로 옆으로 흐릅니다. 블록레벨 태그는 아래로 흐릅니다. 너비 또한 화면 전체를 차지합니다. 자, 그렇다면 블록레벨 태그와 인라인 레벨 태그와의 관계는 어느 정도 감을 잡았습니다. 그러면 도대체 언제 어떤 태그를 사용해야 할까요?

특정 단어를 강조하고 싶을 때는 **** 태그보다 **** 태그를 사용하는 것이 좋습니다. 외관상으로는 두 태그 모두 텍스트를 굵게 보이게 하여 강조해 주지만, **** 태그는 단지 굵게 만들어 줄 뿐 강조하는 아무 의미가 없는 태그입니다. 반면에 **** 태그는 외형적으로도 텍스트를 굵게 해 주고 실제로 의미 있는 텍스트라고 강조해 주는 역할도 합니다.

웹사이트의 메뉴나 목록 형태의 콘텐츠를 만들 때는 ****과 **** 태그를 조합합니다. 만약에 사이트의 이용약관과 같이 목록 앞에 일정한 숫자를 순차적으로 붙이고 싶을 때는 ****, ****를 이용하면 관리하기도 쉬운 훨씬 의미 있는 마크업이 됩니다.

****이나 **<mark>**와 같은 태그는 특정한 부분에 별다른 의미를 부여하지는 않아도 시각적으로 무언가 효과를 주고 싶을 때 사용합니다. ****으로 감싸고 CSS로 접근을 하여 스타일링을 하면 되겠습니다.

이 밖에도 주로 레이아웃을 짜거나 박스를 만들 때는 **<div>**를 많이 사용했는데, HTML5에서는 이것이 조금 더 세분화되었습니다. 부분별로 **<header>**나 **<footer>**, **<section>**과 같은 태그를 사용하면 됩니다. 이 부분에 대해서는 뒤에서 조금 더 자세히 다루겠습니다.

모든 태그가 가지는 고유의 특징들이 있어서 이 섹션에서 일일이 설명하기는 어렵습니다. 이 책으로 공부를 하면서 각 태그를 언제 어느 때 사용하는 것이 적절한지 익히기 바랍니다. 또한 책의 맨 끝에 있는 부록을 통해서 HTML5의 모든 태그와 그 역할에 대해서 한눈에 파악해 보는 것도 공부에 많은 도움이 될 것입니다.

HTML5 문서 계층 구조

HTML의 문서 계층 구조는 반드시 이해하고 넘어가야 합니다. HTML 문서의 계층 구조를 이해하지 못하면 장차 CSS나 자바스크립트를 절대 다룰 수 없기 때-

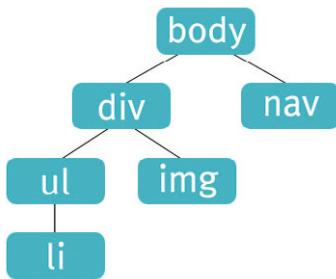
문입니다. 다음 코드를 봅시다.

구조를 중시하는 트리구조

예제 2.6 트리구조 마크업의 예

```
<body>
  <div>
    <ul>
      <li></li>
    </ul>
    <img>
  </div>
  <nav>
  </nav>
</body>
```

위의 코드와 같은 구조를 가진 문서가 있다고 가정합시다. 쉽게 설명하기 위해 다른 코드는 생략하였습니다. 이 코드를 다이어그램으로 표현해 보면 다음 그림과 같습니다.



☞ 그림 2.5 앞서 코드를 다이어그램으로 표현한 모습

박씨 집안의 시조가 박혁거세이듯이 `<head>` 부분을 제외하고 이 문서에 있는 모든 태그들의 시조는 `<body>`입니다. `<body>` 요소는 문서의 모든 요소들을 담고 있습니다. 말을 만들자면 시조태그인 셈입니다. `<div>`는 `<body>` 요소의 자식태그입니다. 반대로 `<div>`에게 `<body>` 태그는 부모태그입니다. 한 계층 차이가 나기 때문입니다. 그러면 이 예시에서 `<div>`와 ``의 관계는 어떻게

될까요? 눈치가 빠른 분은 금방 이해하셨으리라 생각합니다. <div>에게 요소는 손자태그입니다. 반대로 에게 <div> 요소는 할아버지뻘이 됩니다.

자, 그럼 마지막 문제입니다. 이 문서에서 과 요소는 어떤 관계일까요? 바로 형제지간입니다. 같은 높이의 계층에 위치하고 있기 때문입니다. 별로 어렵지 않죠?

아웃라인과 콘텐츠 구성을 중시하는 구조

이미 HTML 페이지를 마크업해 온 사람들은 익숙하겠지만, 방금 설명한 것은 가장 기본적인 트리 형태의 구조입니다. 물론 HTML5에서도 이런 단순 트리 형태의 구조를 사용할 수 있지만 추가된 개념이 있습니다.

기존에는 문서 하나에 H1 헤더 하나가 붙고, 각 하위 콘텐츠별로 H2 태그가 붙는 탑다운 방식으로 문서 하나를 완성했습니다. 그러나 HTML5 문서는 각 섹션이 별도의 H1 태그를 가져도 무방합니다. 박스별 콘텐츠들은 독립된 하나의 문서가 될 수 있습니다. 여러 문서가 합해져 하나의 HTML 파일이 된다고 생각하면 이해하기가 쉬울까요?

또한, 기존에는 <div>를 이용해서 레이아웃을 구성했습니다. 헤더에 해당하는 부분은 <div class="header"></div>, 푸터에 해당하는 부분은 <div class="footer"></div>, 이런 식으로 많이 사용했습니다. HTML5에서는 이런 부분들을 조금 더 세분화하여 <header> 태그가 추가되었고 <footer> 태그도 추가되었습니다. 이 밖에도 많은 구성요소들이 추가되어 아웃라인을 살린 마크업 페이지를 만들 수 있게 되었습니다.

책에서는 이해를 돋기 위해서 페이지의 ‘구조’라는 단어를 계속 사용하겠지만 HTML5에서는 콘텐츠의 ‘구성’이 더 중요해졌습니다. 이것에 대한 설명은 다음 섹션의 ‘문서의 구조화’에서 자세히 하겠습니다.

HTML5에 추가된 글로벌 속성

HTML5에는 새로운 태그들이 추가되었고 자바스크립트 API들도 추가되었습니다. 또한 HTML5에서는 HTML에서 사용할 수 있는 속성들도 추가되었습니다.

각 태그마다 고유하게 사용할 수 있는 속성들은 공부를 하면서 하나씩 살펴보도록 하고, 이번에는 모든 태그에서 사용할 수 있는 글로벌 속성에 대해서 알아보겠습니다. 새로이 추가된 HTML5 글로벌 속성은 유용한 것들이 많으므로 꼼꼼히 공부하여 익혀 둡시다.

웹페이지를 바로 수정할 수 있는 `contenteditable`

이 속성을 사용하면 폼의 텍스트 필드가 아닌데도 사용자가 웹페이지에 있는 문구를 즉각적으로 수정할 수 있습니다.

☞ 표 2.1 브라우저별 `contenteditable` 속성 지원 현황

 익스플로러	 파이어폭스	 사파리	 크롬	 오페라
6.0 버전부터 일부지원	3.5 버전부터 지원	3.2 버전부터 지원	3.0 버전부터 지원	10.1 버전부터 지원할지 미지수

예제 2.7 `contenteditable`을 사용하도록 설정한 섹션

```
<body>
<section contenteditable="true">
<hgroup>
  <h1>박스 텍스트 수정</h1>
  <h2>contenteditable을 사용했습니다.</h2>
</hgroup>
<article>
  <h3>브라우저에서 수정해 보세요.</h3>
  <ul>
    <li>정말로 수정할 수 있나요?</li>
    <li>네, 수정할 수 있어요. 지금 클릭해서 수정해 보세요.</li>
  </ul>
</article>
</section>
</body>
```

자, 이렇게 마크업을 해 보았습니다. 커다란 섹션 박스가 하나 있고 이 텍스트들은 모두 섹션 박스 안에 포함됩니다. 이들을 감싸고 있는 섹션에 강조된 코드를 주목해 주세요. ‘`contenteditable`’ 속성이 ‘`true`’ 값을 가지도록 설정해 두었습니다.

이 마크업을 브라우저로 확인해 보겠습니다.



☞ 그림 2.6 웹페이지에 있는 텍스트를 브라우저상에서 직접 편집하고 있습니다.

어때요? 신기하죠? 이 속성을 지정하면 하위에 있는 모든 요소들에 영향을 미칩니다. 이 예시에서는 맨 위에 있는 `<section>` 태그에 이 속성을 적용했습니다. 그래서 `<section>` 태그 아래에 있는 모든 텍스트를 수정할 수 있게 된 것입니다. 만일 `<body>` 태그에 이 속성을 적용한다면 페이지에 있는 모든 요소들을 편집 할 수 있게 됩니다. 이 속성이 가지는 값은 'true' 와 'false' 두 개뿐입니다.

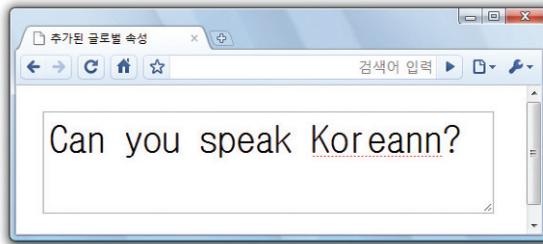
틀린 스펠링을 체크해 주는 `spellcheck`

이 속성은 인풋 타입을 텍스트로 사용하거나 `<textarea>` 태그로 사용자에게 텍스트를 입력받을 때 사용자가 틀린 텍스트를 입력하면 빨간색 밑줄로 알려 주는 속성입니다. 흔히 워드프로세서 프로그램에서 많이 제공하는 기능으로, HTML5에서도 유용하게 쓰일 것입니다. 아쉬운 점은 이 속성은 영어 단어나 영어 문장에만 해당한다는 점입니다. 한글 체크 속성도 있으면 좋겠는데, 아쉽습니다.

예제 2.8 `<textarea>`에 틀린 영문 스펠링을 잡아 줍니다.

```
<body>
  <textarea spellcheck="true"></textarea>
</body>
```

강조된 부분처럼 사용하면 됩니다. 이 코드를 브라우저로 확인해 보았습니다.



☞ 그림 2.7 틀린 영어문장을 사용자가 입력하면 빨간색 밑줄로 지적해 줍니다.

이 속성 역시 'true' 와 'false' 2가지 값을 가집니다.

HTML5에 추가된 글로벌 속성 중 대표적인 2가지를 알아보았습니다.

추가된 속성 중 'hidden' 속성을 사용하면 해당 태그 안에 있는 콘텐츠를 감출 수 있습니다. 메모리에서 지우는 것이 아니라 단지 감추는 기능입니다. <div hidden>, 이렇게 사용하면 됩니다. 사용자가 임의로 태그의 속성을 만들어서 사용해야 할 때는 'data-' 속성을 사용합니다. 접두어로 'data-'를 사용하는 이유는 향후 HTML5 그리고 그 이후의 HTML이 버전 업되면서 추가되는 속성과 사용자가 임의로 만든 속성이 충돌하는 것을 피하게 하는 것입니다. 이 속성은 다음과 같이 사용하면 됩니다. <div data-yourage="28"> .

설명한 글로벌 속성 외에도 추가된 속성이 몇 가지 더 있지만 특정 태그나 특정 API와 연동할 때 자주 사용하는 속성은 뒤에서 따로 설명하겠습니다.

02

더 시맨틱하게…

기존의 HTML4.x와 XHTML1.x 버전이 거의 전 세계 웹페이지에 사용되고 있을 때, 이미 디자인을 CSS로 떼어내 웹페이지를 제작하는 방법이 널리 애용되었습니다. 많은 사람들이 시맨틱한 웹페이지와 나아가 웹표준, 웹접근성까지 지키려고 참 많은 노력을 했습니다. 그래서 이제는 어느 정도 마크업과 스타일의 분리

가 보편적으로 통용되고 있는 느낌입니다.

그러나 HTML5 이전 버전의 HTML에서는 `<u>`와 같은 요소나 `'bgcolor'`와 같이 단지 꾸며 주기 위해 존재하던 HTML 마크업 속성이 완전히 제거되지 않고 아직까지 존재합니다. HTML5에서는 조금 더 시맨틱한 웹페이지를 제작할 수 있도록 단지 꾸며 주기만 하는 요소들을 모두 제거해 버렸습니다. 이제 스타일링이나 디자인, 꾸밈에 필요한 것은 전적으로 CSS에서 하도록 완전히 강제로 분리해버린 것입니다.

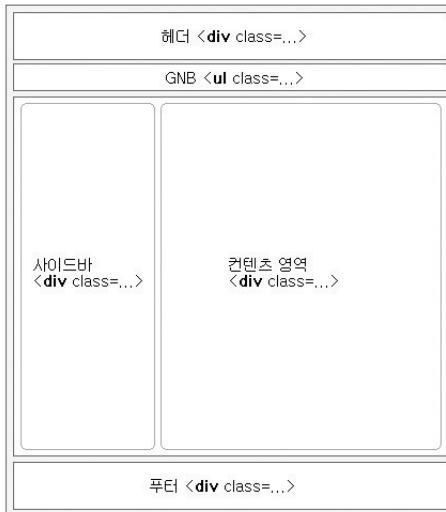
또한 이번에 새로이 추가된 `<aside>`, `<header>`, `<article>`과 같은 속성은 문서를 조금 더 구조적으로 만들 수 있게 해 줍니다. 기존에 `<div>`나 ``등의 요소로 페이지의 각 부분을 구조화했다면 이제는 헤더면 헤더, 푸터면 푸터, 조금 더 각 부분의 적합한 요소들을 사용하여 배치할 수 있게 된 것입니다.

또, `<time>`, `<meter>`와 같이 의미 있는 특정 단위의 시간이나 숫자를 입력할 수 있는 요소도 추가되었습니다. 그럼, 조금 더 시맨틱해진 HTML5 요소들에 대해서 살펴보겠습니다.

아웃라인 구성과 문서의 구조를 지키는 마크업

HTML5 이전의 웹페이지는 다음과 같이 구조화하였습니다. 일단 문서의 가장 큰 제목은 H1, 부제목은 H2, 섹션별로 H3의 제목을 부여하였습니다. 그리고 헤더와 바디, 푸터와 기타 사이드바나 각 모듈은 `<div>`요소나 `` 등의 요소를 이용해서 구조화하였습니다. 과거의 구조화된 문서를 이미지로 그려 보면 그림 2.8과 같습니다.

이 그림에서 보다시피 페이지의 구조를 아우르는 레이아웃은 대부분 `div` 혹은 `ul`로 만들어져 있습니다.



☞ 그림 2.8 HTML5 이전의 기존 웹페이지들은 푸터나 헤더나 콘텐츠 영역에 상관없이 div나 ul 등의 요소를 이용하여 각 영역을 구조화하였습니다.

HTML5에 추가된 문서 구조화 태그

그러나 HTML5에는 문서의 구조화와 관련되어 몇 가지 요소가 추가되었습니다. 과거보다는 조금 더 세밀하게 문서의 구조화를 할 수 있게 된 것입니다. 앞서도 잠시 언급했지만 여러분들의 이해를 쉽게 하기 위해서 ‘구조화’라는 단어를 계속하여 사용하고 있습니다. 그러나 HTML5 문서는 ‘구성’을 중시합니다. 용어에 큰 의미를 두지 않아도 될 것 같습니다. 문서의 의미 있고 체계적인 구조화를 위해서 일단 HTML5에 추가된 요소들을 살펴보겠습니다. 다음 표를 봐 주세요.

☞ 표 2.2 HTML5에 추가된 주요 문서 구조화 관련 요소

요소 이름	역할	HTML5 추가
body	문서의 콘텐츠 전체를 담습니다. 부모 요소는 오직 <HTML>뿐인 2인자 태그입니다.	X
section	문서 또는 애플리케이션의 콘텐츠를 포함합니다. 주로 제목(h)과 함께 사용. 같은 테마를 가진 여러 개의 콘텐츠를 그룹핑합니다.	O
nav	페이지의 섹션 한 부분을 구성하는 요소로, 페이지의 GNB(글로벌 내비 게이션)를 담당합니다.	O
article	블로그의 글 하나나, 사이드바의 위젯처럼 독립적인 개별 콘텐츠를 담습니다.	O

◆ 표 2.2 계속

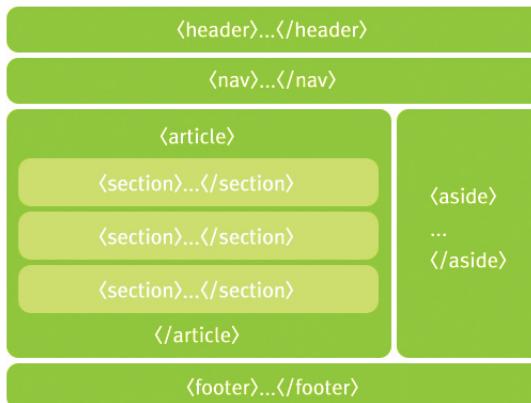
요소 이름	역할	HTML5 추가
aside	문서의 주요 콘텐츠에 포함되지 못하고 남아 있는 잔여 콘텐츠를 담습니다. 주로 사이드바로 많이 활용될 것 같습니다.	O
h1~h6	문서의 헤드라인입니다. H1이 중요도가 가장 높습니다. hgroup 요소에 포함되면 헤더 섹션에서 다수의 문서 제목을 할 수 있습니다.	X
hgroup	헤더 요소에 포함되어 여러 개의 헤드라인(h)을 사용할 수 있습니다.	O
header	문서나 각 섹션, 아티클 등의 헤더 부분에 사용 가능하며, 그것들의 제목이나 간단한 소개 콘텐츠를 담을 수 있습니다.	O
footer	문서 하단에 들어가는 저작권 정보, 서비스 제공자 정보 등을 담을 수 있는 푸터 요소들을 포함합니다.	O
address	콘텐츠 작성자나 사이트 소유자의 연락처 정보 등을 담을 수 있습니다. 만일 개별 <article>에 이 요소가 사용된다면 그 아티클 작성자의 연락처 정보가 됩니다.	X

이 표에서 소개한 요소들을 이용하여 HTML5 방식으로 문서의 구조를 만들어 본다면 어떤 모습일까요? 간단한 마크업을 참고해 주세요.

예제 2.9 아웃라인을 살린 HTML5 마크업 뼈대의 예

```
<body>
  <header>...</header>
  <nav>...</nav>
  <article>
    <section>...</section>
    <section>...</section>
    <section>...</section>
  </article>
  <aside>...</aside>
  <footer>...</footer>
</body>
```

이 코드를 어떤 레이아웃으로 만들 것인지는 다음 그림을 주목해 주세요.



☞ 그림 2.9 HTML5에 새로 추가된 요소들을 이용해서 HTML5 방식으로 페이지의 구조화를 단행한 모습입니다. 앞서 만들었던 코드로 이 다이어그램과 같은 레이아웃을 만들 수 있습니다. 레이아웃의 각 부분이 DIV와 같은 단순한 하나의 요소가 아니라 하나하나 최적화되어 있다는 것을 알 수 있습니다.

문서의 구조를 조금 더 의미 있게 만드는 부분에 대해서는 이제 감이 잡히시죠? 지금까지 본 것은 콘텐츠를 담는 각 컨테이너의 전체적인 구조였습니다. 컨테이너 못지않게 중요한 것이 문서의 제목입니다. 문서의 제목과 컨테이너의 구성을 잘 하면 웹표준을 준수하여 HTML 마크업을 하는 데 있어서 절반은 성공했다고 생각하면 됩니다.

HTML5 이전의 웹문서들이 문서 내에서 헤드라인, 즉 H태그를 어떻게 다루었는지 알아보고 HTML5 스타일로 바꾸어 보는 작업을 해 보겠습니다.

<h1> 일반적으로 서비스의 이름</h1>

<h2> 일반적으로 메인 카테고리 이름</h2>

<h3> 섹션별 타이틀</h3>

컨텐츠 블라블라...

<h3> 섹션별 타이틀</h3>

컨텐츠 블라블라...

☞ 그림 2.10 HTML5 이전에 권장되던 헤드라인 사용방식의 예

헤드라인 태그를 이전처럼 꼭 어느 한쪽으로만 국한해 사용할 필요는 없습니다. 그럼 2.10에 보이는 것처럼 보통 h1태그는 문서의 가장 큰 제목으로 회사명이나 서비스명, 슬로건 등을 사용하는 경우가 많고, h2태그는 그것보다는 한 단계 아래인 카테고리의 제목, 그리고 그보다 더 한 단계 아래인 h3태그는 섹션별 타이틀을 가져가는 경우가 많았습니다. 이것으로 온전히 HTML 문서 한 장이 일반 워드프로세서의 문서 한 장처럼 의미 있는 문서가 되는 것이었습니다.

HTML5에서도 이처럼 마크업을 하면 됩니다. 웹표준과 웹접근성 측면에서 괜찮은 방식이라고도 생각합니다. 다만, 헤드라인 사용에 있어서 HTML5에서는 이것과 관련해서 생각해야 할 것이 하나 있습니다. 바로 콘텐츠의 구성에 대한 부분입니다.

헤드라인을 여러 개로 묶을 필요성이 있을 때 사용하는 `hgroup` 요소

만일 페이지에 산발적으로 여러 개의 헤드라인을 묶어 줘야 할 필요가 있을 경우 HTML5에서는 다음과 같이 hgroup 요소를 이용할 수 있습니다.

예제 2.10 <hgroup> 태그의 사용 예

```
<header>
  <hgroup>
    <h1>쏭군 블로그</h1>
    <h2>열심히 살아가는 88만월 세대의 이야기</h2>
  </hgroup>
</header>
<article>
  <hgroup>
    <h1>최근 올라온 글</h1>
    <h2>최근에 올라온 글을 시간 순으로 표시합니다.</h2>
  </hgroup>
  <section>
    <hgroup>
      <h1>봉사활동</h1>
      <h2>나도 그들도 짠했던 이야기</h2>
    </hgroup>
  </section>
  <section>
    <hgroup>
      <h1>자전거 구입</h1>
```

```

<h2>싼 녀석을 구하기 위한 사투스토리!</h2>
</hgroup>
</section>
</article>
<aside>
<hgroup>
<h1>위젯 사이드바</h1>
<h2>개인적으로 좋아하는 위젯들 모음</h2>
</hgroup>
</aside>

```

앞서 과거에 사용되던 방식을 사용하는 것도 괜찮습니다. 그러나 HTML5에 추가된 `<hgroup>` 요소 덕분에 이 예제 코드와 같은 방식으로도 헤드라인 태그를 운용할 수 있게 되었습니다. H1이나 H2와 같은 높은 단계에 있는 헤드라인 요소들도 이제는 문서의 최상단의 전유물이 아니라, 각 섹션이나 아티클 등 어디에서나 꾸릴 수 있게 된 것입니다. 이렇게 하면 각 콘텐츠가 훨씬 독립적으로 보입니다. 위의 코드를 다이어그램으로 만들어 보면 다음 그림과 같습니다.



☞ 그림 2.11 `<hgroup>`을 사용한 코드를 브라우저에서 확인하면 아마 다음과 같은 구조로 출력될 것입니다.

문서의 의미만 잘 구현된다면 헤드라인을 어떻게 사용하여도 큰 문제는 없습니다. 다만, 앞서 설명한 방식대로 콘텐츠를 쪼개 각 콘텐츠에 높은 레벨의 헤드라인을 부여하여 의미를 담을 수 있다는 것 정도를 알아 둔다면 나중에 요긴하게 활용할 때가 있을 것입니다.

독립된 개별 콘텐츠를 담는 article, section 요소

섹션 요소에는 서로 다른 의미를 가진 콘텐츠를 담을 수 있습니다. 예를 들면 블로그에서 각 포스트를 요약해 둔 메인페이지가 꾸며진 경우를 볼 수 있는데, 그 요약된 포스트들의 묶음을 <section> 태그로 담고 각 개별 포스트는 <article> 태그로 묶으면 됩니다.



☞ 그림 2.12 테크크런치 (techcrunch.com)

이 그림은 발 빠른 IT 소식이 올라오는 블로그 테크크런치입니다. 빨간색 박스로 되어 있는 부분은 `<section>` 태그로 감싸고, 파란색 부분은 `<article>` 태그로 담아내면 될 것입니다. 보다시피 파란색 박스는 개별 콘텐츠를 담고 있습니다. 빨간 박스와 파란 박스를 마크업한다면 다음과 같습니다.

예제 2.11 독립적인 콘텐츠를 담는 `<section>`과 `<article>` 태그 사용 예

```
<section>
  <article>
    <h1>Google Secretly...</h1>
    ...
  </article>
  <article>
    <h1>It's As Is Apple...</h1>
    ...
  </article>
  <article>
    <h1>DevHub Now...</h1>
    ...
  </article>
</section>
```

`<article>` 요소에는 완전히 떼어내 다른 사이트에 붙여도 그 의미를 사용자가 알 수 있는 콘텐츠를 담으면 됩니다.

내비게이션을 만들 때 사용하는 nav 요소

문서의 글로벌 내비게이션이나 사이드 내비게이션 등 페이지의 이동을 위한 내비게이션을 만들 때는 `<nav>` 태그에 담도록 합니다. 이 태그는 다음과 같은 구성에 사용합니다.



☞ 그림 2.13 빌보드차트 웹사이트의 내비게이션 (billboard.com)

이 그림은 벨보드차트 사이트의 내비게이션입니다. 이 내비게이션은 `<nav>` 태그를 이용해서 다음 예제와 같이 마크업할 수 있습니다.

예제 2.12 `<nav>` 태그의 사용 예

```
<nav>
  <ul>
    <li>CHARTS</li>
    <li>NEWS</li>
    <li>VIDEO</li>
    <li>REVIEWS</li>
    <li>LIVE</li>
  </ul>
</nav>
```

`<nav>` 태그는 페이지의 ‘이전’, ‘다음’과 같이 페이지의 페이징 부분의 섹션을 담을 때 사용할 수도 있습니다.

잔여 콘텐츠를 담는 `aside` 요소

`<aside>`는 아웃라인 요소들 중에서도 가급적 조심스럽게 사용해야 하는 요소입니다. 웹페이지에 들어가는 모든 콘텐츠를 전부 다 담고도 들어갈 곳이 없는 잔여 콘텐츠는 이 곳에 담습니다. 아주 간단하게 사이드바를 생각하면 됩니다. 보통 사이드바에는 메인 콘텐츠와 직접 연관이 없는 위젯과 같은 것을 담게 되는데 이런 것을 `<aside>` 요소로 처리하면 됩니다.

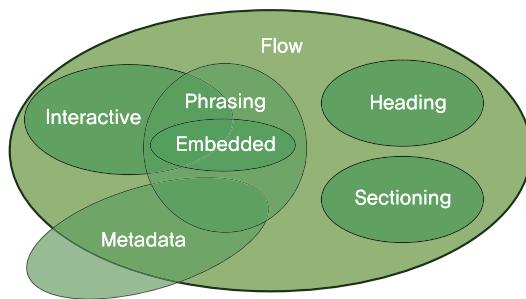
div 요소 사용

아웃라인을 위한 새로운 태그들이 대거 추가되면서 `<div>`를 사용할 필요가 많이 사라졌습니다. 기존에는 콘텐츠를 담거나 레이어를 만들기 위해 언제 어디서나 `<div>`나 ``과 같은 태그를 사용해 왔습니다. 그러나 이것은 의미에 그다지 맞지 않는 것이었습니다. 이제는 의미에 맞는 아웃라인 태그들이 `<div>`가 하던 역할을 대신하게 되었습니다.

`<div>`는 사실상 존재의 의미가 많이 퇴색되었습니다. `<div>`를 아예 쓰지 않아도 무방합니다. 굳이 `<div>`를 사용해야 한다면 아웃라인을 위한 태그나 기타 의미에 맞는 마크업을 위한 태그들이 아예 쓰이지 않는 곳에서 사용하면 될 것입니다.

콘텐츠 타입별 태그의 분류

아웃라인의 구성이 중시되고 각각의 콘텐츠가 가지는 의미가 더 중요해진 HTML5 문서에서 각 태그가 가지는 역할과 의미를 확실히 이해하는 것은 더 중요할 것입니다. 그리고 큰 흐름에서 어떤 분류로 콘텐츠가 나뉘는지도 알고 있다면 더 유익할 것입니다. 아래의 다이어그램과 도표는 WHATWG에 올라온 콘텐츠를 가져온 것입니다.



☞ 그림 2.14 HTML5의 콘텐츠 타입별 분류 (이미지 출처 : WHATWG 공식 사이트)

이 그림을 보면, 플로우형 콘텐츠 그룹에 속하는 태그에는 일부 메타데이터 콘텐츠 그룹에 속하는 태그를 제외하고 모든 태그가 포함된다는 것을 알 수 있습니다.

☞ 표 2.3 콘텐츠 타입별 태그 분류 (자료 출처 : WHATWG)

콘텐츠 타입 분류	설명	속한 태그
콘텐츠 흐름형 Flow	HTML 문서의 흐름과 관련된 모든 태그가 이 부류에 속합니다. 일부 메타데이터형 태그를 제외하고 <body> 태그 아래에 포함되는 거의 모든 태그가 포함됩니다.	a, abbr, address, area, article, aside, audio, b, bdo, blockquote, br, button, canvas, cite, code, command, datalist, del, details, dfn, div, dl, em, embed, fieldset, figure, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup, hr, i, iframe, img, input, ins, kbd, keygen, label, link, map, mark, math, menu, meta, meter, nav, noscript, object, ol, output, p, pre, progress, q, ruby, samp, script, section, select, small, span, strong, style, sub, sup, svg, table, textarea, time, ul, var, video, wbr, text

☞ 표 2.3 계속

콘텐츠 타입 분류	설명	속한 태그
섹션형 Sectioning	헤더와 푸터 범위 내에서 콘텐츠를 그룹화하는 모든 섹션형 태그가 이 콘텐츠 부류에 포함됩니다. 모든 섹션형 콘텐츠는 잠재적으로 헤드라인과 아웃라인을 가지게 됩니다.	article, aside, section, nav
메타데이터형 Metadata	콘텐츠의 모양을 담당하는 외부 CSS, 동작을 담당하는 스크립트를 사용할 수 있습니다. 이외에도 외부 문서와의 관계를 담당하는 태그들이 이 부류에 속합니다.	base, command, link, meta, noscript, script, style, title
임베디드형 Embedded	HTML문서에서 외부의 동영상이나 사진, 사운드, 드로잉 이미지를 사용할 필요가 있을 때 사용하는 태그들이 이 콘텐츠 부류에 속합니다.	audio, canvas, iframe, video, embed, img, math, object, svg, video
상호작용형 Interactive	사용자가 직접 컨트롤할 수 있고 사용자와 상호작용을 할 수 있도록 해 주는 태그들이 이 콘텐츠 부류에 속합니다.	a, audio, button, details, embed, iframe, img(이미지 맵 사용시), input(hidden속성 제외), keygen, label, menu(툴바 탑입 사용시), object, select, textarea, video(컨트롤 패널 사용시)
텍스트, 표현형 Phrasing	문서의 텍스트나 문장 내부의 텍스트를 마크업할 때 사용되는 태그들이 이 부류에 속합니다.	a, abbr, area, audio, b, bdo, br, button, canvas, cite, code, command, datalist, del, dfn, em, embed, i, iframe, img, input, ins, kbd, keygen, label, link, map, mark, math, meta, meter, noscript, object, output, progress, q, ruby, samp, script, select, small, span, strong, sub, sup, svg, textarea, time, var, video, wbr, text
제목형 Heading	각 섹션의 헤드라인에 속하는 부분을 담당하는 콘텐츠 타입입니다. 스스로 헤드라인을 담당한다는 것을 의미적으로 가지고 있습니다.	h1, h2, h3, h4, h5, h6, hgroup

의미를 부여받은 특별한 텍스트

HTML5는 텍스트를 위한 요소도 많이 추가되었습니다. 과거의 `` 태그와 같이 의미 없는 태그는 사라졌습니다. 그러나 이제는 조금 더 디테일하고 많이 사용될 일종의 템플릿과도 같은 의미 있는 요소가 많이 추가되었습니다. 사실 과거에는 헤드라인이면 헤드라인, 레이블이면 레이블, 일반 텍스트면 텍스트, 목록이면 목록, 기껏해야 이 정도 수준에서 텍스트를 해석할 수 있었습니다. 그러나 지금 공부할 요소들을 사용하면 이 텍스트가 거리를 의미하는지, 시간을 의미하는지, 대화 부분 중 하나인지 등 조금 더 자세한 정보를 알 수 있습니다.

대화를 표현하는 `dialog` 태그

대화를 의미 있는 콘텐츠로 만들고자 할 때 `<dialog>` 요소를 사용하면 됩니다. 일대일 대화도 좋고, 여러 사람의 회의록 같은 자료도 의미에 맞게 저장할 수 있습니다. 일단 다음 예제 코드를 살펴보겠습니다.

예제 2.13 `<dialog>` 태그에 개미와 배짱이의 대화를 표현한 예

```
<dialog>
  <dt>배짱이</dt>
  <dd>야 개미야. 넌 왜 맨날 일만하나?</dd>
  <dt>개미</dt>
  <dd>응, 지금 열심히 일해야, 겨울에 따뜻하게 쉴 수 있거든</dd>
  <dt>배짱이</dt>
  <dd>야야, 겨울일은 겨울에 신경쓰라구…</dd>
  <dt>개미</dt>
  <dd>훗, 넌 나한테 밥 얻으려 오지마라~</dd>
  <dt>배짱이</dt>
  <dd>넌 나한테 월세나 밀리지 말고 잘내라~</dd>
</dialog>
```

구조는 간단합니다. `<dt>`는 말하는 사람의 이름을 작성합니다. `<dd>`는 그 사람이 한 말을 작성합니다. 이전 버전의 HTML과 XHTML에서 `ul`이나 `ol`이 많이 사용되었던 반면 `dl`을 활용하는 사례는 잘 찾아보지 못했습니다. 물론 포털사이트 다음과 같이 `dl` 요소를 잘 활용하는 사이트도 많습니다. 어쨌건, `<dialog>` 요소 덕분에 `<dt>`와 `<dd>` 요소가 빛을 보게 되었습니다. `<dt>` 요소와 `<dd>` 요

소는 이전 버전의 HTML 문서에서도 의미 있는 요소였기 때문에 HTML5가 지원되지 않는 환경에서도 충분히 대화 내용을 인지하도록 하는 데 문제가 없을 것이라고 생각합니다.



☞ 그림 2.15 <dialog> 태그 안에 포함된 **div** 요소, 대화를 표현하기에 적당합니다.

이 요소는 글로벌 속성 이외에 단독으로 지원하는 속성이 없습니다.

이미지와 캡션을 묶어 블록으로 만들어 주는 figure 태그

사실 이 요소는 이미지와 캡션을 묶어 준다고 초기에 알려졌지만, 이미지만 블록으로 만들어 주는 것이 아닙니다. 비디오나 소스 코드 리스트, 일러스트, 다이어그램, 일반 텍스트 등을 모두 하나의 단위로 묶어 줄 수 있습니다. 즉, 텍스트와 다른 요소간 태그들을 하나의 블록으로 묶고 캡션을 포함할 필요가 있을 때 유용하게 사용할 수 있습니다.

하나의 단위로 묶을 때 캡션이 필요하게 되는데, 이때 함께 사용되는 태그가 <figcaption> 태그입니다. <figcaption> 태그는 <figure>로 묶인 박스를 설명하는 캡션으로 사용합니다. 일단 아래의 예제 코드를 보겠습니다.

예제 2.14 <figure> 태그의 사용 예

```
<figure>
  <figcaption>세계 최고의 검색엔진</figcaption>
  
</figure>
```

만약에 아우디를 소개하기 위한 `figure`가 있다고 하면, 다음과 같이 HTML 마크업을 만들 수 있습니다.

예제 2.15 `<figcaption>` 태그의 추가

```
<figure>
  <figcaption>아우디</figcaption>
  <p>아우디는 1899년 설립됐다. 1901년 첫 모델을 발표했다…</p>
  
</figure>
```

이렇게 하면 이 `<figure>` 블록은 ‘아우디’에 대한 것임을 알 수 있습니다.

시각적 주목 효과를 텍스트에 주기 위한 `mark` 태그

`<mark>` 요소는 문장 내에서 특정 텍스트를 강조해야 할 필요가 있을 때 사용합니다. 그러나 혼동하지 말아야 할 것이 있습니다. 바로 `` 태그와의 차이점입니다. `` 태그는 그 의미도 강조하지만 `<mark>` 요소는 의미까지 강조하지는 않습니다. 단지 스타일을 적용하여 시각적으로만 강조하고 싶을 때 `<mark>` 요소를 사용하면 됩니다. 우리가 흔히 형광펜으로 ‘마킹한다’고 할 때 사용하는 바로 그 용도입니다. `<mark>` 요소는 `<m>`으로 줄여서도 사용할 수 있습니다. 그러나 가급적 `<mark>`로 사용하길 권합니다. `<m>` 요소의 존재 여부는 지금도 논쟁이 많습니다. 이런 논쟁 중 사라질 수도 있기 때문입니다.

이렇게 마크업을 하면 됩니다. 간단하지요?

예제 2.16 무언가 시각적으로 강조하고 싶을 때 `<mark>` 태그를 사용

```
<p>이번 월드컵에서 한국은 반드시 <mark>16강</mark>에 진출할 것입니다.</p>
```

이 태그는 시각적으로나 의미적으로나 아무 의미가 없습니다. 단지 CSS를 이용해서 시각적 변화를 줄 수 있습니다.

예제 2.17 `<mark>` 처리된 부분을 CSS로 강조

```
mark {font-weight:bold;}
```

이렇게 하면 ‘16강’ 부분이 굽게 표시될 것입니다.

아마도 이전에 ``이 하던 역할과 비슷한 역할을 많이 할 것 같습니다.

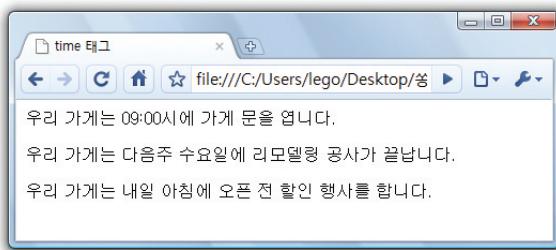
텍스트를 시스템이 인지하는 시간으로 만들어 주는 `time` 태그

이 태그를 사용하면 특정한 시간을 표시할 수 있습니다. 즉, 검색엔진이나 브라우저가 단순하게 생각해 버리는 평범한 텍스트가 아니라, 특정한 날짜와 시간이라는 점을 인지할 수 있도록 해 주는 태그입니다. 다시 말해, 평범한 텍스트를 시스템이 인지할 수 있는 시간이나 날짜 정보로 만들어 주는 태그입니다.

예제 2.18 `<time>` 태그 사용 방법의 예

```
<p>우리 가게는 <time>09:00</time>시에 가게 문을 엽니다.</p>
<p>우리 가게는 <time datetime="2010-04-14">다음주 수요일</time>에 리모델링 공사가 끝납니다.</p>
<p>우리 가게는 <time datetime="2010-04-20T09:00+10:00">내일 아침</time>에 오픈 전 할인 행사를 합니다.</p>
```

`<time>` 요소를 사용한 예제 코드입니다. 이 코드를 브라우저로 확인하면 다음 그림과 같습니다. 스타일링을 따로 해 주지 않는 이상 실제 브라우저상으로 알 수 있는 부분은 없습니다. 그러나 의미상으로는 큰 변화가 생겼습니다. 시스템이나 웹브라우저가 `<time>` 태그로 감싸인 부분을 시간이나 날짜로 인지하게 된 것이죠. 시맨틱한 텍스트가 된 것입니다!



☞ 그림 2.16 `<time>` 태그로 시간과 날짜의 의미를 가지게 된 텍스트들입니다. 일단 따로 CSS 스타일은 적용하지 않았기 때문에 겉으로는 아무런 변화가 없습니다.

첫 번째 문장에서 ‘09:00’이라는 정보가 온전하게 오전 9시라는 의미를 가진 시간 정보가 되었습니다. 두 번째 문장에서 ‘다음주 수요일’이라는 단어가 ‘2010년 4월 14일’이라는 시간 정보를 가지게 되었습니다. 마지막 세 번째 문장에서 ‘내일 아침’이라는 단어가 정확하게 2010년 4월 20일 오전 9시부터 10시까지의 한 시간 동안이라는 의미를 가지게 되었습니다.

<time> 요소는 글로벌 속성 외에 datetime이라는 하나의 속성을 가집니다.

이 datetime 속성에는 시스템이 인지할 수 있는 올바른 형태의 시간을 작성해야 합니다. 만일 이 속성을 사용하지 않을 경우에는 소스 코드의 첫 번째 줄처럼 시스템이 인지하는 시간 정보를 가진 텍스트를 감싸서 사용하면 됩니다.

특정 수치값을 가질 때 사용하는 meter 태그

반드시 두께나 양, 통화, 인구, 점수나 치수와 같은 숫자들과 같이 특정 범위 내에 있는 소소한 숫자 값을 시스템에 인지시킬 필요가 있을 때 <meter> 태그를 사용합니다. ‘%’나 분수 형태로도 사용할 수 있습니다. 혹은 ‘mm’나 ‘kg’과 같은 치수 정보를 가질 필요가 있을 때도 이 <meter> 태그를 이용합니다. 일단 간단한 사용 예제 몇 가지를 살펴보겠습니다.

예제 2.19 가장 기초적인 <meter> 태그의 마크업

```
<p>수학 : <meter>80</meter>점</p>
```

기본적으로는 이렇게 사용하면 됩니다. 하지만 <meter> 태그에 있는 속성들을 추가적으로 사용한다면 조금 더 정밀하고 진정한 의미를 가진 값을 지정해 줄 수 있습니다.

예제 2.20 최저값과 최대값을 정해 줌

```
<p>수학 : <meter min="0" max="100">80</meter>점</p>
```

min 속성과 max 속성을 추가했습니다. 가장 낮은 점수는 0점이고, 가장 높은 점수는 100점입니다.

예제 2.21 경험상 최대값과 실제의 값을 추가

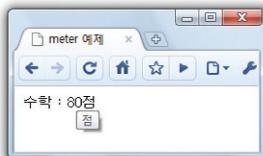
```
<p>수학 : <meter min="0" max="100" value="80" optimum="95">80점</meter></p>
```

내가 받았으면 하는 최고 점수는 95점인데 실제로는 80점을 받았습니다.

예제 2.22 <meter> 태그에 title값을 추가

```
<p>수학 : <meter min="0" max="100" value="80" optimum="95" title="점">80점</meter></p>
```

이번에는 툴팁을 추가해 보았습니다. 이 텍스트 위에 마우스를 올리면 다음 그림과 같은 툴팁을 볼 수 있습니다.



☞ 그림 2.17 <meter> 태그를 이용해서 만든 문장. title 태그를 이용하면 툴팁을 볼 수 있습니다.

기본적인 코드들을 보니 조금은 이해가 가죠?

☞ 표 2.4 <meter> 태그의 특별한 속성

속성명	속성의 기능
value	실제로 측정된 진짜 데이터를 지정합니다.
title	툴팁을 입력할 수 있습니다.
high	입력된 데이터들 중의 사용자가 허용할 수 있는 최대값을 지정합니다
low	입력된 데이터들 중의 사용자가 허용할 수 있는 최저값을 지정합니다.
max	meter에서 인식할 수 있는 최고값을 지정합니다
min	meter에서 인식할 수 있는 최저값은 지정합니다.
optimum	최적의 측정값을 지정해 줍니다. min으로 설정된 최저값과 max로 설정된 최대값 사이에서 가능한 한 최고 적합한 값을 지정해 줍니다. 예를 들면 시험점수는 0점에서 100점까지 나올 수 있지만 내 나름대로 95점만 받아도 선방했다고 생각하면 optimum값으로 95를 입력해 주면 됩니다. 가능한 높은 쪽 기준이겠죠. 반대로 어떤 권투선수가 시합에서 질 수 있는 최대한의 수는 10경기고 최소한의 수는 0경기라고 가정합시다. 이 선수가 스스로 2경기만 져도 선방했다고 생각한다면 optimum값은 낮은쪽 기준으로 2를 입력해 주면 됩니다. 덜 질수록 좋은 것이니까요.

<meter> 태그는 앞서 본 형태 외에도 다양한 형태의 ‘수치’ 정보를 가질 수 있습니다.

예제 2.23 <meter> 태그에 수치 정보를 담는 예

```
<p>높이 : <meter min="100" max="400" title="밀리미터">250mm</meter></p>
```

이렇게 사용할 수도 있고,

예제 2.24 <meter> 태그에 점유율 정보를 담는 예

```
<p>점유율 : <meter min="0" max="500000" value="100000">20%</meter></p>
<p>판매율 : <meter min="0" max="100" value="50">1/2</meter></p>
```

이런 식으로 ‘%’나 ‘분수’ 형태로도 사용할 수 있습니다.

다운로드 진척도를 알려 주는 progress 태그

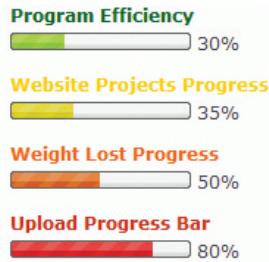
이 요소는 다운로드 진척 상태를 알려줄 때 유용하게 사용할 수 있는 태그입니다.

예제 2.25 <progress> 태그의 사용 예

```
파일을 다운로드 중입니다 :
<progress value="50000" max="100000">
  <span id="status">50</span>%
</progress>
```

이 요소는 글로벌 속성과 함께 자체적으로 2개의 속성을 가지고 있습니다. 우선 max 속성은 다운로드해야 할 최대값을 지정합니다. 그리고 value값은 현재까지 다운로드한 값을 나타냅니다.

그리고 이 요소는 필연적으로 자바스크립트와 연동해서 사용해야 합니다. 자바스크립트와 연동하여 사용하면 페이지의 다운로드 정보를 동적으로 제공할 수 있고 페이지가 로드되는 동안 사용자의 지루함을 덜 수 있고, 페이지 로드에 대한 정보를 제공할 수 있습니다.



☞ 그림 2.18 <progress> 태그를 적극적으로 이와 같이 활용할 수 있습니다. (이미지 출처 : amolwable.com)

책을 쓰고 있는 현재는 제대로 지원하는 브라우저가 없지만 추후 브라우저들이 이 요소를 제대로 지원한다면 이 그림과 같이 progress 막대를 만들어서 사용할 수 있을 것입니다.



게이지 막대 형태의 값을 사용할 때는 ...

주의할 점이 있습니다. <progress> 태그로 게이지 막대 형태의 값을 사용하는 데는 사용하지 않기를 권장합니다. 게이지에 적합한 태그는 <meter> 태그입니다.

상호작용을 위한 요소

다음에 설명할 요소들은 사용자와 직접 상호작용을 위한 요소들입니다. 이 요소들은 WHATWG의 웹 애플리케이션1.0 스펙 안에 포함됩니다. 웹 애플리케이션 1.0 스펙은 기본적으로 시맨틱한 웹페이지를 위한 마크업을 지향합니다. 특징으로 기존 서버에서 처리하던 것을 브라우저 선에서 처리할 수 있도록 만드는 기술의 추가, 브라우저 간 통신 기능의 향상을 들 수 있습니다. 이 섹션에서 설명할 것은 현재 위젯으로 통칭되고 있는(사실은 웹 애플리케이션의 형태 중 일부인) 곳에서 유용한 툴바나 명령어 박스와 같이 직접 상호작용하는 요소에 관한 것입니다.

각주로 활용하기에 편리한 details 태그

<details> 태그는 HTML 문서에서 추가적인 설명을 붙일 때 사용합니다. 워드 프로세서의 ‘각주’ 역할과 비슷합니다. HTML 문서 전체만 아니라 문서의 각 부분마다 ‘각주’ 기능이 필요한 경우에는 <details> 태그를 사용하면 됩니다.

예제 2.26 <details> 태그와 <summary> 태그를 사용해 각주 생성

```
<details>
  <summary>이 문서에 대한 추가 정보</summary>
  <p>이 웹문서는 2010년 4월 15일에 만들어졌습니다. 문서 저작자에게 연락주세요.</p>
</details>
```

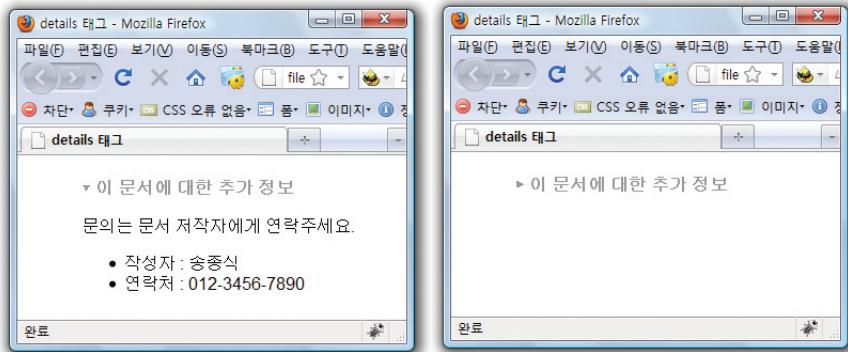
<details> 요소는 <summay> 태그와 함께 사용됩니다. <summary> 가 하는 역할은 <legend> 나 <caption>, <figcaption>과 비슷합니다. <details> 박스에 대한 간략한 설명을 담을 수 있습니다.

<details> 요소는 기본적으로 ‘감춤’ 상태입니다. <summary> 부분만 보이도록 되어 있고 그 외 콘텐츠는 기본적으로 감춰져 있습니다. 이 <summary> 부분을 클릭하면 추가 콘텐츠를 보여 주거나 감출 수 있습니다. 그런데 기본적으로 <details> 콘텐츠를 모두 펼쳐 놓기 위한 속성이 있는데 ‘open’ 속성입니다. 글로벌 속성 이외에 <details> 태그가 가지고 있는 유일한 속성입니다.

예제 2.27 <details> 태그에 open 속성을 적용하여 각주와 설명이 기본적으로 펼쳐져 있게 함

```
<details open="open">
  <summary>이 문서에 대한 추가 정보</summary>
  <p>문의는 문서 저작자에게 연락주세요.</p>
  <ul>
    <li>작성자 : 송종식</li>
    <li>연락처 : 012-3456-7890</li>
  </ul>
</details>
```

이렇게 해 주면 이 각주 박스는 기본적으로 펼쳐지게 됩니다. open 속성이 가지는 값은 open뿐입니다. <summary>를 클릭해서 <details> 내용을 보였다 감췄다 토글하는 것은 자바스크립트를 이용해서 동작시켜야 합니다.



☞ 그림 2.19 <detail> 태그의 open 속성이 적용되어 각주가 열려 있는 모습(왼쪽)과 각주가 닫혀 있어서 <summary>만 보이는 모습(오른쪽)

훌륭한 UI 개발자인 레미가 만든 샘플 사이트(<http://remysharp.com/demo/details-with-js.html>)를 방문해 보면 잘 만들어진 <details> 태그 샘플을 볼 수 있습니다. 이 샘플을 보면 알겠지만 꽤 많은 양의 스크립트와 여러 가지 특수 CSS3 선택자가 사용되었습니다. 지금 이 섹션에서 설명하기에는 진도가 맞지 않고 지면상 문제도 있습니다. 그러나 이 책을 끝까지 공부한다면 여러분도 이 정도 샘플을 쉽게 만들 수 있게 됩니다.

데이터 그리드를 직접 컨트롤할 수 있는 datagrid 태그

datagrid 요소는 기초 자료를 HTML 태그로부터 가져옵니다. 이렇게 가져온 데이터는 자바스크립트 등을 이용해서 동적으로 정보를 갱신할 수 있습니다. 테이블로부터 다음과 같이 자료를 가져올 수 있습니다.

예제 2.28 <table> 태그로 만든 일반적인 표

```
<table>
<tr>
<td>영어</td>
<td>국어</td>
</tr>
<tr>
<td>90점</td>
<td>85점</td>
</tr>
</table>
```

이렇게 하면 일반적인 테이블에 불과합니다. 그냥 정적인 테이블이죠.

예제 2.29 <datagrid>로 감싸 테이블의 값을 외부에서 컨트롤할 수 있도록 함

```
<datagrid>
<table>
<tr>
<td>영어</td>
<td>국어</td>
</tr>
<tr>
<td>90점</td>
<td>85점</td>
</tr>
</table>
</datagrid>
```

이렇게 하면 datagrid용 데이터가 됩니다. 이제 자바스크립트를 활용하면 사용자가 브라우저를 통해서 직접 셀을 늘리거나 줄일 수 있게 됩니다. 셀을 선택하여 삭제하거나 추가하고 데이터를 변경할 수도 있습니다.

이 요소는 동적인 데이터를 꼭 표 형태로만 표현하는 것이 아닙니다. 트리구조 형태나 목록 형태의 데이터도 얼마든지 동적으로 처리할 수 있습니다.

예제 2.30 테이블뿐 아니라 목록형태의 값도 컨트롤할 수 있고 한 번에 여러 행을 컨트롤하는 것이 가능

```
<datagrid multiple="multiple">
<ol>
<li>데이터그리드 0행</li>
<li>데이터그리드 1행</li>
<ol>
<li>데이터그리드 1행 0열</li>
<li>데이터그리드 1행 2열</li>
</ol>
<li>데이터그리드 2행</li>
</ol>
</datagrid>
```

이처럼 목록형 데이터도 사용할 수 있습니다.

◆ 표 2.5 datagrid의 속성과 기능

속성명	속성값	기능
disable	기본 빈 값, disabled	datagrid를 비활성화시킵니다.
multiple	기본 빈 값, multiple	사용자가 여러 행을 한 번에 선택할 수 있게 해 줍니다.
글로벌 속성	-	-

웹 애플리케이션 툴바를 만들 수 있는 menu와 command 태그

<menu> 태그는 폼 컨트롤 목록이나 명령어 목록을 만들 때 사용합니다. <command> 태그는 라디오버튼이나 체크박스, 버튼을 이용해서 명령어 박스를 만들 수 있게 해 줍니다. <command> 태그는 반드시 <menu> 태그 안에 포함되어야 합니다.

흔동하지 말아야 할 것은 아웃라인 태그인 <nav>와의 차이점입니다. <nav> 태그는 사이트의 글로벌 내비게이션에 사용할 수 있으며 URL링크를 사용할 수 있습니다. 다만 <menu>와 <command> 태그는 클라이언트 측 애플리케이션 실행을 위한 것이므로 URL링크나 폼 액션은 처리하지 않습니다. 이 부분은 명확한 차이점으로 <nav> 태그와 흔동하지 말기 바랍니다.

이것들의 기본적인 사용 형태는 다음과 같습니다.

예제 2.31 <menu> 태그와 <command> 태그의 기본적인 사용 예

```
<menu>
  <command type="command">새 문서</command>
</menu>
```

가장 기본적인 형태의 사용법입니다. <menu> 요소 안에 <command> 요소가 포함되어 있다는 것을 잊지 마세요.

<menu> 태그는 사용하고자 하는 목적에 따라 몇 가지 타입을 지정해 줄 수 있습니다. <menu> 태그에서 사용가능한 속성은 다음 표를 참고해 주세요.

☞ 표 2.6 <menu> 태그에서 사용가능한 속성 확장태그들과 값

속성명	속성값	설명
type	list (기본값) toolbar context	사용할 <menu>의 타입을 지정합니다. 툴바 형태로 사용할 것인지 목록 형태로 사용할 것인지를 지정할 수 있습니다.
label	레이블 텍스트	메뉴에 대해서 알려 줄 레이블입니다.

<menu>를 이용해서 다음과 같이 툴바를 만들 수 있습니다.

예제 2.32 <menu> 태그를 이용해서 툴바를 만드는 마크업의 예

```
<menu type="toolbar">
<li>
  <menu label="파일">
    <button type="button" onclick="newfile()">새 파일</button>
    <button type="button" onclick="fileopen()">열기</button>
    <button type="button" onclick="fileopen()">저장</button>
  </menu>
</li>
<li>
  <menu label="편집">
    <button type="button" onclick="btncopy()">복사</button>
    <button type="button" onclick="btncut()">잘라내기</button>
    <button type="button" onclick="btnpaste()">붙여넣기</button>
  </menu>
</li>
<li>
  <menu label="보기">
    <button type="button" onclick="vbig()">크게</button>
    <button type="button" onclick="vsmall()">작게</button>
    <button type="button" onclick="vfullscreen()">전체화면</button>
  </menu>
</li>
</menu>
```

이러한 구조의 마크업으로 <menu>를 이용한 툴바를 만들 수 있습니다. 눈여겨 볼 것은 <menu>안에 다시 <menu>가 포함될 수 있다는 것입니다. 그리고 <command> 태그의 경우에는 반드시 <menu> 태그 안에 포함되어야 하지만 <menu> 태그 안에는 <command>뿐 아니라 여러 가지 인풋 요소나 콤보박

스, 와 같은 목록이 들어갈 수 있다는 것입니다.



☞ 그림 2.20 <menu> 태그를 사용하여 만든 툴바 예시 화면 (이미지 출처 : W3C 공식사이트)

<command>를 이용하면 사용자가 요청하는 명령을 즉각적으로 실행할 수 있습니다. <command> 태그는 <menu> 태그 안에 포함되지 않으면 아마도 그 내용이 브라우저에서 보이지 않을 것입니다.

☞ 표 2.7 <command> 태그에서 사용 가능한 확장태그들과 값

속성	값	설명
type	command (기본값) checkbox radio	<command> 태그에서 사용할 명령어 형태의 타입을 지정합니다. <command>가 기본값입니다. 라디오버튼이나 체크박스로 지정해 줄 경우 명령어 박스는 버튼 형태가 아니라 항목을 확인해 주는 형태로 사용할 수 있습니다.
label	명령어 설명, 이름	<command>의 각 명령어 항목별 설명 텍스트 혹은 이름을 입력합니다.
disabled	disabled	
icon	아이콘 이미지 URL	각 커맨드 항목에 아이콘을 만들어 줄 수 있습니다. 아이콘은 외부 이미지 URL 주소로 가져옵니다.
radiogroup	라디오 그룹 이름	타입을 라디오 버튼으로 지정했을 경우 이들 라디오 버튼을 묶어서 이름을 지정해 줄 수 있습니다.
checked	checked	타입을 라디오버튼이나 체크박스로 지정했을 경우 이들을 기본적으로 선택상태로 할 것인지를 설정할 수 있습니다.

예제 2.33 <menu>와 <command>를 함께 사용하여 애플리케이션 툴바를 마크업하는 예

```
<menu type="toolbar">
  <command type="command" icon="icon_lock.gif" label="잠금" onclick="elock()"/>
  <command type="command" icon="icon_open.png" label="열기" onclick="elock()"/>
</menu>
```

이 코드는 <command> 태그의 사용 예입니다. 만일 우리가 흔히 사용하는 블로그나 이메일 편집기 상단에 있는 툴바를 만들고 싶다면 이와 같이 메뉴의 타입은 'toolbar'로, 커맨트 타입은 'command'로 지정해 주고 마크업하면 됩니다.

그러나 아쉽게도 아직까지 <command>와 <menu> 태그를 제대로 랜더링할 수 있는 브라우저가 없어서 어떤 결과 화면을 얻기는 어려울 것입니다. 조만간 브라우저들이 <command> 태그를 잘 지원해 주리라 믿습니다.

03

무엇이든 그려내는 Canvas 요소

canvas는 이미 마음만 먹으면 이전 버전의 HTML에서도 api를 이용할 수 있었습니다. 그럼에도 불구하고 HTML5의 새로운 스펙들이 하나씩 공개되면서 가장 큰 주목을 받은 요소 중 하나이기도 합니다. 이 녀석이 하는 역할은 이름에 그대로 묻어납니다. 간단하게 그림을 그리거나 사진합성을 할 수 있습니다. 그래프를 그릴 수도 있고 애니메이션이나 심지어 게임을 출력할 수도 있습니다. HTML5 자체가 리치웹을 추구하지만, 특히 이 canvas 요소 덕분에 플래시와 HTML5 간의 대결의 장이 부각되었습니다. 표준 Canvas 2D(3D) API로 웹의 무궁무진한 그래픽을 랜더링해 주는 소중한 요소입니다.

☞ 표 2.8 브라우저별 canvas 요소 기본 지원 현황 (2D 컨텍스트)

 익스플로러	 파이어폭스	 사파리	 크롬	 오페라
9.0 버전부터 지원 할지 미지수	3.0 버전부터 지원	3.1 이상 버전부터 지원할지 미지수	2.0 버전부터 지원	9.6 버전부터 지원



Canvas의 기본 속성

- **Canvas 기본 배경색** : 투명(transparent)
- **Canvas 기본 크기** : 너비 300픽셀, 높이 150픽셀 (크기 지정을 하지 않을 경우)

Canvas 기본 사용법

canvas 요소를 다루기 위해서는 약간의 자바스크립트를 다룰 수 있어야 합니다. 물론, HTML에 대한 지식도 가지고 있어야 합니다. canvas 요소는 파이어폭스1.5 이상의 버전이나 오페라9, 사파리4, 크롬, 인터넷 익스플로러9 버전 이상의 브라우저에서 작동합니다.

예제 2.34 캔버스 마크업

```
<canvas id="exam" width="200" height="200">
```

가장 기본적인 형태의 canvas 사용 방법입니다. canvas 요소는 딱 두 가지의 속성만 가지고 있습니다. 너비를 지정해 주는 width와 높이를 지정해 주는 height 속성이 그것입니다. 이 두 속성이 외에 어떤 속성도 지원하지 않습니다. 단, canvas 요소를 외부의 스크립트나 CSS 등으로 식별하기 위한 클래스나 아이디와 같은 글로벌 속성은 지원합니다. 특히, canvas는 스크립트와 함께 사용하지 않으면 무용지물이 되므로 아이디 값을 항상 지정해 주는 습관을 들이면 좋습니다.

그 밖에 고려해야 할 점이 있습니다. canvas 요소를 지원하지 않는 구형 브라우저에서는 어떻게 해야 할까요? 그런 경우에는 대체 콘텐츠를 제공하면 됩니다. 이때 canvas 태그는 홀태그가 아니라 짹이 있는 태그로 사용합니다.

예제 2.35 캔버스를 지원하지 않는 브라우저를 위한 대체 이미지 제공

```
<canvas id="exam" width="200" height="200">

</canvas>
```

기본적인 코드와 달리 여기서는 </canvas>가 추가되어 확실히 닫아 주었습니다. 여는 canvas 태그와 닫히는 canvas 태그 사이에 콘텐츠가 들어갈 수 있습니다.

canvas 요소를 읽을 수 있는 브라우저라면 대체 콘텐츠를 무시하고 canvas 내용만 출력합니다. 반대로 canvas 요소를 읽을 수 없는 브라우저라면 canvas 요소를 무시하고 가운데 있는 대체 콘텐츠만 출력합니다. 대체 콘텐츠는 텍스트나 이미지 등 구형 브라우저가 인식할 수 있는 어떤 콘텐츠라도 상관없지만 가급적 canvas

요소에서 원래 보여 주려고 했던 사항과 크게 다르지 않은 콘텐츠면 좋습니다. 앞의 코드에서는 canvas가 지원되지 않으면 GIF로 된 예제이미지를 대체하여 출력합니다.

만일 대체 콘텐츠가 필요하지 않다면 아래와 같이 처리하면 됩니다.

예제 2.36 기본적인 캔버스 마크업

```
<canvas id="exam" width="200" height="200"></canvas>
```

Canvas를 이용해서 그림 그리기 준비

canvas 요소는 getContext()라고 하는 DOM 메소드를 제공합니다. 말이 조금 어렵지만, 막상 예제 코드를 한 번만 타이핑해 보면 쉽게 이해할 수 있을 것입니다. 또한 이용하기도 쉽습니다.

canvas 태그를 작성하면 텅 비어 있는 요소가 하나 생깁니다. 텅 비어 있는 이 canvas 안에 그림을 그려야 하는데 그림을 그리는 도구로는 자바스크립트 같은 스크립트가 동원됩니다. 자바스크립트에 대해 조금만 알아도 쉽게 이해할 수 있습니다. 자, 시작해 볼까요?

우선 앞에서 사용했던 HTML5 마크업을 가져와서 사용하겠습니다.

예제 2.37 앞서 만들어 두었던 캔버스 마크업과 캔버스가 지원되지 않는 브라우저를 위한 대체 이미지 제공

```
<canvas id="exam" width="200" height="200">
  
</canvas>
```

그리고 canvas가 지원하는 그리기 컨텍스트에 접근하기 위해서 다음과 같이 자바스크립트를 만들어 보겠습니다.

예제 2.38 자바스크립트로 캔버스에 접근

```
var exam = document.getElementById('exam'); // id값이 exam인 캔버스를 불러와서 exam이라는 변수에 담습니다.
var ctx = exam.getContext('2d');
```

마크업에서 canvas에 부여해 준 아이디 값 'exam' 을 자바스크립트의 첫째 문장에서 getElementById를 통해서 받아왔습니다. 이로써 canvas의 DOM 노드를 가져와서 'exam' 변수에 담아 둘 수 있게 되었습니다. 그리고 평면에 그림을 그릴 수 있도록 '2d' 컨텍스트를 둘째 줄의 'ctx' 라는 변수에 담았습니다.



기대되는 3D context (WebGL)

초기에는 getContext의 컨텍스트 아이디에 '2d' 컨텍스트가 가장 많이 쓰일 것으로 생각합니다. 아직은 2d 컨텍스트를 지원하기에도 구형 브라우저의 점유율이 너무나 낮은 실정입니다. 그래서 이 책에서도 2d 컨텍스트에 대해서만 다룹니다. 그러나 Open GL 그리고 Open GL의 임베디드 버전인 Open GL ES를 기반으로 하는 WebGL이 공개됨에 따라 '3d' 컨텍스트를 문제 없이 사용할 수 있도록 자리 잡힌다면 HTML5로 할 수 있는 것은 무궁무진하게 많아질 것입니다. 10여 년 전 VRML이 이루어지 못한 여러 가지 꿈을 HTML5가 이루어 줄지도 모르는 일입니다. WebGL에 대해서는 '플래시를 대체하기 힘들다'는 등 논란의 여지가 많지만 구글에서 공개한 오픈 소스인 O3D를 보면 3D 그래픽이 상당히 정교하게 웹브라우저에서 돌아간다는 것을 알 수 있습니다. 물론 자바스크립트 기반입니다! 브라우저상에서 자바스크립트만으로 WebGL이 얼마나 정교하게 처리될 수 있겠느냐 하는 논란은 O3D의 등장으로 조금 잠잠해졌습니다. 3D 온라인 쇼핑몰, 3D 온라인 모델하우스, 웹게임 등의 분야에서 유용하게 쓰일 것으로 기대됩니다.

구글 O3D 웹사이트 : <http://code.google.com/apis/o3d/>

앞서 HTML5의 canvas 태그를 지원하지 않는 브라우저를 위해서 대체 콘텐츠를 사용하는 방법을 배웠습니다. 브라우저가 canvas 요소를 지원하는지 지원하지 않는지는 자바스크립트를 이용해서 체크할 수도 있습니다.

예제 2.39 캔버스에 그림을 그리기 위해 2D 컨텍스트에 접근

```
var exam = document.getElementById('exam');
if (exam.getContext) {
    var ctx = exam.getContext('2d');
    // 그림을 그려낼 코드를 여기에 작성합니다.
} else {
    // Canvas가 지원되지 않는 경우 출력할 코드를 만듭니다
}
```

이 자바스크립트 코드를 이용하면 브라우저가 canvas 요소를 지원하는지 여부를 체크할 수 있습니다.

자, 이렇게 canvas에 그림을 그리기 위한 메소드 동원까지 끝났습니다. 본격적으로 도형 그리기를 시작해 보겠습니다.

예제 2.40 페이지가 로드되면 캔버스에 그림이 그려지도록 준비 완료

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Canvas 공부</title>
    <script type="text/javascript">
      var make_canvas = function() { // 페이지가 열리자마자 호출될 make_canvas 함수
        var exam = document.getElementById('exam');
        var ctx = exam.getContext('2d');
      };
    </script>
    <style type="text/css">
      #exam { border: 1px solid #000000; }
    </style>
  </head>
  <body onload="make_canvas();">
    <canvas id="exam" width="200" height="200"></canvas>
  </body>
</html>
```

이 예제 코드를 만들어 앞서 설명한 canvas가 지원되는 브라우저에서 열어 보세요. 아직은 본격적으로 그림을 그리지 않았습니다. canvas를 사용하기 위한 뼈대만 잡은 상태이기 때문에 너비 200픽셀, 높이 200픽셀의 검정색 박스 하나만 보일 것입니다. canvas 요소는 기본적으로 테두리나 배경색을 가지고 있지 않기 때문에 눈에 잘 띄도록 스타일시트를 이용해서 테두리 경계선을 넣어 주었습니다. 그리고 <body>에 onload 이벤트 핸들러를 사용해서 페이지가 열리자마자 ‘make_canvas’라는 이름을 가진 자바스크립트 함수가 작동되도록 하였습니다.

Canvas에 사각형 그리기

예제 2.41 캔버스에 간단한 사각형 그리기

```
<!DOCTYPE HTML>
<html>
<head>
<title>Canvas 공부</title>
<script type="text/javascript">
var make_canvas = function() {
var exam = document.getElementById('exam');
var ctx = exam.getContext('2d');
ctx.fillStyle = "rgb(0, 0, 255)"; // 어떤 색을 채울 것인지 RGB값 설정
ctx.fillRect (10, 10, 100, 100); // 사각형 그리기 시작점, 끝점 좌표 설정
};
</script>
<style type="text/css">
#exam { border: 1px solid #000000; }
</style>
</head>
<body onload="make_canvas()">
<canvas id="exam" width="200" height="200"></canvas>
</body>
</html>
```

이 코드에서 파란색 박스를 그리기 위해 두 줄의 문장이 추가되었습니다. 우선 `fillStyle` 속성에 대해서 알아보겠습니다. 이 함수는 도형을 어떤 색으로 채울지 정해 줄 수 있습니다.

`ctx.fillStyle = "rgba(0, 0, 0, 0.5)";`

순서대로 0~255까지의 RGB값
0~1까지 투명도 조절

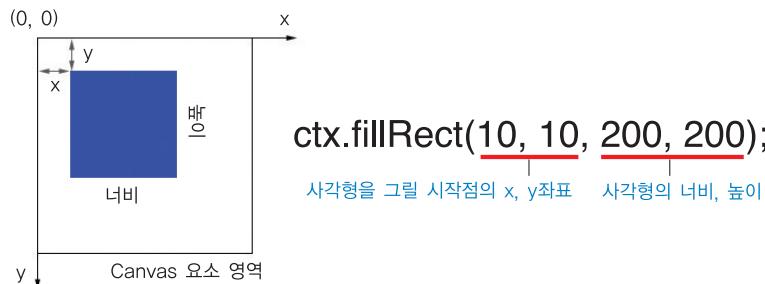
- ☞ 그림 2.21 `fillStyle` 속성은 Red, Green, Blue의 0부터 255 사이의 값을 받아서 도형의 색상을 채웁니다. RGBA에서 A에 해당하는 인자의 네 번째 숫자는 투명도를 조정하는 것으로 0부터 1 사이의 값을 사용할 수 있습니다. 0은 완전 투명, 1은 불투명, 0.5는 반투명입니다. 적당한 값을 사용하면 됩니다. ctx는 앞서 설명한 코드 예제에서 canvas의 context를 담고 있는 변수입니다.



알아 두면 좋은 팁

`fillStyle` 함수를 이용할 때 반투명 효과가 필요 없는 경우에 RGBA값에서 알파값 A는 생략할 수 있습니다. 또한 앞의 소스처럼 RGB값을 사용할 수도 있고 `ctx.fillStyle = '#0000FF'`;와 같이 16진수 코드를 사용할 수도 있습니다.

`fillRect` 속성은 속이 가득 찬 사각형을 그릴 때 사용합니다. 사용 방법은 다음과 같습니다.



☞ 그림 2.22 `fillRect` 속성은 속이 가득 찬 사각형을 그려 줍니다. 4개의 인자 값 중에서 앞의 두 개 값은 상자 왼쪽 상단 모서리 위치를 지정합니다. 이 부분에서 사각형을 그려 나갑니다. 뒤의 인자 두 개는 사각형의 높이와 너비를 지정합니다. 사각형을 그려낼 시작점 x, y 좌표는 HTML 페이지 기준이 아니고, canvas의 왼쪽 위 모서리를 기준으로 위치합니다. canvas 요소가 HTML 페이지 어디에 존재하든 상관 없이 canvas 왼쪽 상단 모서리를 기준으로 하므로 참고해 주세요. 좌표와 사각형의 크기를 지정할 때 기준 단위는 픽셀입니다. ctx는 앞서 설명한 코드 예제에서 canvas의 context를 담고 있는 변수입니다.

canvas에 사각형 도형을 그리기 위해 필요한 속성은 다음 표에 나타낸 세 가지입니다. 별로 어렵지 않아 익히기도 쉽습니다. 인자는 모두 동일합니다. 사각형의 왼쪽 상단 모서리 좌표를 지정해 주고 크기를 정해 주면 끝입니다.

☞ 표 2.9 Canvas에 사각형을 그릴 때 사용되는 함수 속성

함수 속성 이름	역할	인자
<code>fillRect</code>	가득 찬 사각형을 그릴 때 사용	<code>x, y, width, height</code>
<code>strokeRect</code>	사각형의 테두리만 그릴 때 사용	<code>x, y, width, height</code>
<code>clearRect</code>	지정된 영역을 투명하게 지울 때 사용	<code>x, y, width, height</code>

패스를 이용해서 선 긋기, 선 스타일 지정하기

canvas를 이용해서 선을 그어 보겠습니다. 선을 그을 때 canvas API가 제공하는 다양한 스타일을 이용할 수 있습니다.

기본 선 긋기

기본 사용법 : `lineTo(x, y);`

이 메소드는 사실 선이 끝나는 부분의 값을 가지고 있습니다. 이전 `lineTo` 메소드의 좌표를 가져와 선을 쭉쭉 이어 나가면서 그릴 수 있습니다.

예제 2.42 간단한 선 긋기

```
ctx.beginPath();           // 새로운 도형을 그림  
ctx.lineTo(10,10);       // 선이 시작할 좌표 설정  
ctx.lineTo(100,100);     // 선이 끝나는 곳 좌표 설정  
ctx.stroke();            // 선 긋기
```

가장 기본적인 형태의 `lineTo` 메소드의 사용 방법입니다. 첫 번째 줄에 있는 `lineTo` 메소드에서 선을 긋기 위해 시작하는 시작점의 위치를 지정해 줍니다. 그리고 두 번째 줄에 있는 `lineTo` 메소드에 입력된 좌표로 선을 긋습니다.

예제 2.43 처음 그린 선과 연결하여 두 번째 선 긋기

```
ctx.beginPath();           // 새로운 도형을 그림  
ctx.lineTo(10,10);       // 선이 시작할 좌표 설정  
ctx.lineTo(100,100);     // 선이 끝나는 곳 좌표 설정  
ctx.lineTo(50,150);      // 이전 좌표에서 연결하여 선 긋기  
ctx.stroke();            // 선 긋기
```

이렇게 하면 x좌표 10픽셀, y좌표 10픽셀에서 출발한 선이 100, 100까지 그어지고, 다시 거기서 연결된 선이 50, 150좌표를 향해 그어지게 됩니다. 지금까지 만든 코드를 브라우저로 확인하면 2개의 선이 보일 것입니다. 이 코드에 `closePath` 메소드를 추가해 보겠습니다.

예제 2.44 선 굵기 종료

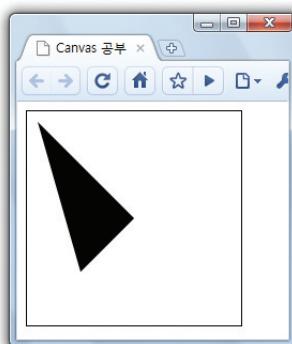
```
ctx.beginPath();           // 새로운 도형을 그림  
ctx.lineTo(10,10);        // 선이 시작할 좌표 설정  
ctx.lineTo(100,100);      // 선이 끝나는 곳 좌표 설정  
ctx.lineTo(50,150);        // 이전 좌표에서 연결하여 선 굵기  
ctx.closePath();          // 도형 그리기를 종료합니다  
ctx.stroke();             // 선 굵기
```

이 예시에서 패스를 종료하는 `closePath` 메소드가 사용되었습니다. 그러면 선 굵기가 종료된 지점과 선 굵기가 최초로 시작된 지점이 연결됩니다. 이 예시를 브라우저에서 실행해 보면 삼각형이 그려져 있을 것입니다.

그런데 이 삼각형의 색을 채우고 싶다면 어떻게 해야 할까요? 방법은 간단합니다. 아래 코드를 확인해 주세요.

예제 2.45 그어진 선분 사이를 채우기

```
ctx.beginPath();           // 새로운 도형을 그림  
ctx.lineTo(10,10);        // 선이 시작할 좌표 설정  
ctx.lineTo(100,100);      // 선이 끝나는 곳 좌표 설정  
ctx.lineTo(50,150);        // 이전 좌표에서 연결하여 선 굵기  
ctx.closePath();          // 도형 그리기를 종료합니다  
ctx.fill();                // 색 채우기
```



☞ 그림 2.23 라인이 그어진 곳들을 선으로 굿는 것이 아니라 안쪽을 채우면 이렇게 됩니다.

이렇게 하면 색이 채워진 삼각형이 완성됩니다. 만일 ctx.fill()을 사용하지 않고 ctx.stroke()를 사용한다면 안이 채워지지는 않고 테두리만 그려진 삼각형이 완성될 것입니다.

의문점이 하나 생깁니다. 만약에 선을 연결해서 그리지 않고 따로 분리되어 있는 선을 그리고 싶다면 어떻게 해야 할까요?

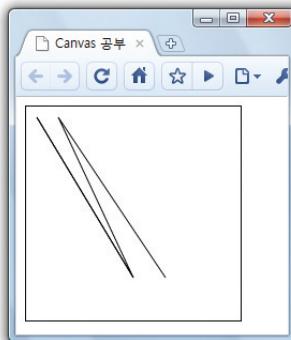
예제 2.46 2개의 선을 분리하여 긋기

```
ctx.beginPath();           // 새로운 도형을 그림

// 첫 번째 선
ctx.lineTo(10,10);
ctx.lineTo(100,160);
ctx.stroke();

// 두 번째 선
ctx.lineTo(30,10);
ctx.lineTo(130,160);
ctx.stroke();
ctx.closePath();          // 도형 그리기를 종료
```

2개의 선을 그리기 위해 위와 같이 코드를 만들었습니다. 이 코드를 브라우저에서 확인해 보겠습니다.



☞ 그림 2.24 2개의 선을 그었는데 3개의 선이 출력되었습니다.

이 그림에서처럼 분명히 2개의 선을 긋도록 스크립트를 만들었는데 3개의 선이 그어져 있는 것을 확인할 수 있습니다. 그 이유는 canvas에서의 도형 그리기는 한번에 하나만 그리는 것을 허용하기 때문입니다. 그래서 첫 번째 선이 끝나는 부분에서 두 번째 선이 시작되는 부분에 우리가 원하지 않는 선이 그어진 것입니다. 코드를 아래와 같이 수정해 보겠습니다.

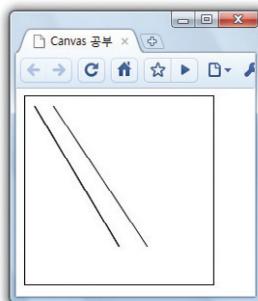
예제 2.47 moveTo()메소드를 이용한 보정

```
ctx.beginPath();           // 새로운 도형을 그림

// 첫 번째 선
ctx.lineTo(10,10);
ctx.lineTo(100,160);
ctx.stroke();

// 두 번째 선
ctx.moveTo(30,10);      // 두 번째 선이 시작할 좌표
ctx.lineTo(130,160);
ctx.stroke();
ctx.closePath();          // 도형 그리기를 종료
```

두 번째 선이 시작하는 좌표를 `lineTo()`가 아니라 `moveTo()`를 이용해서 지정해 주었습니다. 이 함수는 선을 그리지는 않지만 원하는 좌표에서 그리기를 시작할 수 있도록 해 줍니다. 우리가 연필로 그림을 그릴 때 펜을 종이에서 뗐다가 다른 곳으로 이동시킬 때가 있지요? `moveTo()` 함수는 그런 역할을 합니다. 이렇게 코드를 고치고 나서 브라우저로 확인해 보면 우리가 원하는 결과를 얻을 수 있습니다.



☞ 그림 2.25 `moveTo()` 함수를 이용하여 우리가 원하는 두 줄의 선을 긋는 데 성공했습니다.

자, 이제 기본적인 선 긋기 방법에 대해서는 여기까지 알아보고 선을 꾸미는 방법에 대해서 공부해 보겠습니다.

선 스타일

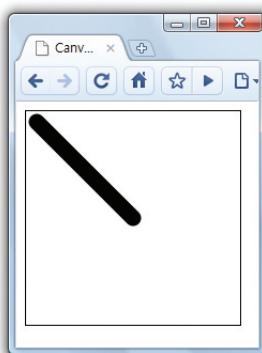
예제 2.48 선 두께 지정

```
ctx.beginPath();           // 새로운 도형을 그림  
ctx.lineWidth = 15;        // 선 두께 지정  
ctx.lineTo(10,10);        // 선이 시작할 좌표 설정  
ctx.lineTo(100,100);      // 선이 끝나는 곳 좌표 설정  
ctx.stroke();             // 선 긋기
```

lineWidth라는 속성이 추가되었습니다. 이 속성은 선의 두께를 설정할 수 있게 해줍니다. 선의 두께를 따로 설정해 주지 않으면 기본 두께는 1.0입니다. 위의 예시에서는 15까지 두께를 늘려 보았습니다. 꽤 굵직한 선이 그어졌습니다.

예제 2.49 선이 끝나는 부분을 둥글게

```
ctx.beginPath();           // 새로운 도형을 그림  
ctx.lineWidth = 15;        // 선 두께 지정  
ctx.lineCap = 'round';     // 선의 끝 부분을 둥글게  
ctx.lineTo(10,10);        // 선이 시작할 좌표 설정  
ctx.lineTo(100,100);      // 선이 끝나는 곳 좌표 설정  
ctx.stroke();             // 선 긋기
```



☞ 그림 2.26 끝이 둥글고 두께가 15픽셀 정도 되는 검정색 선이 그어졌습니다.

끝이 둥글거나 각지도록 선의 스타일을 지정해 줄 수 있고, 선의 끝부분과 꺾이는 부분의 모양을 조절할 수 있습니다. ‘lineCap’ 속성을 사용하면 됩니다. 기본적인 사용 방법은 앞선 예제에서 살펴보았습니다. lineCap에 사용할 수 있는 스타일에는 3가지가 있습니다.



`LineCap = 'butt' LineCap = 'round' LineCap = 'square'`

기본값

☞ 그림 2.27 사용 가능한 3가지의 선 스타일입니다. 선의 끝부분 기준을 약간 넘어가는 round값과 square값에 주목해 주세요. 선의 원래 길이는 시작 기준점과 끝 기준점만큼의 거리입니다.

이렇게 lineCap 속성을 사용하면 모서리의 모양을 바꿀 수 있습니다. 단 ‘butt’ 스타일을 제외한 다른 스타일은 기준이 되는 곳에서부터 불록하게 튀어나오므로 다소 튀어나갑니다. 선의 색상을 바꾸기 위해서는 앞서 박스를 그릴 때처럼 `ctx.strokeStyle`를 사용하면 됩니다.

lineJoin 속성을 사용하면 선이 꺾이는 부분의 모양을 처리할 수 있습니다. 사용 방법은 간단합니다.

예제 2.50 선의 끝부분 모양을 정해 줄 수 있는 lineJoin 속성

```
ctx.lineJoin = "round";
```

이렇게만 지정해 주면 선이 꺾이는 부분에서 원하는 모양으로 출력할 수 있습니다. lineJoin 속성은 3가지 값을 가집니다. 각 값이 의미하는 모양은 다음 그림과 같습니다.



☞ 그림 2.28 lineJoin 속성에 들어가는 3가지 값이 가지는 꺾이는 모양입니다. 기준선 부분까지가 실제 길이입니다.

베지어 곡선 그리기

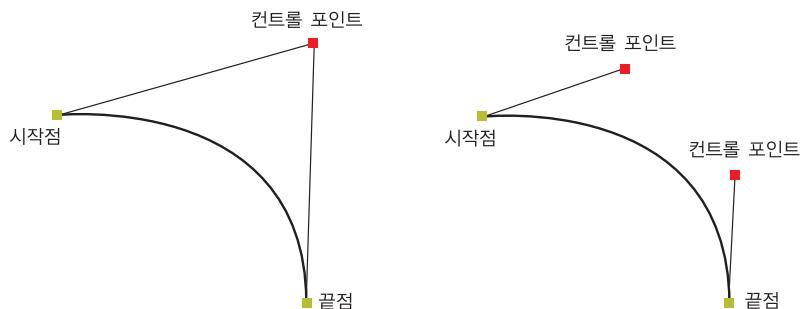
canvas의 2D 컨텍스트는 베지어 곡선 그리기도 지원합니다. 2차 베지어 곡선과 3차 베지어 곡선 모두를 사용할 수 있는데 기본적인 사용 형식은 다음과 같습니다.

예제 2.51 베지어 곡선의 기본 사용법

```
ctx.quadraticCurveTo(cpx, cpy, x, y);           // 2차 베지어 곡선  
ctx.bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y); // 3차 베지어 곡선
```

2차 베지어 곡선, 3차 베지어 곡선이라고 해서 3차 베지어 곡선이 3차원 이미지를 그린다는 것은 아닙니다. 2차 베지어 곡선과 3차 베지어 곡선은 곡선 패스를 컨트롤할 수 있는 컨트롤 포인트의 개수가 다를 뿐입니다. 당연히 컨트롤 포인트가 하나 더 많은 3차 베지어 곡선이 더 디테일한 그림을 그릴 수 있겠지요.

기본 사용 형식에서 cpx와 cpy 부분에는 컨트롤 포인트의 좌표가 들어갑니다. 컨트롤 포인트의 위치를 바꾸어 가면서 곡선의 위치를 만들어 낼 수 있을 것입니다. 그리고 x와 y는 곡선이 끝나는 지점의 좌표입니다. 3차 베지어 곡선에 있는 cp2x와 cp2y는 두 번째 컨트롤 포인트의 좌표값을 입력합니다.



2차 베지어 곡선

quadraticCurveTo

3차 베지어 곡선

bezierCurveTo

☞ 그림 2.29 2차 베지어 곡선과 3차 베지어 곡선의 차이는 컨트롤 포인트 개수의 차이입니다. 당연히 3차 베지어 곡선이 더 디테일한 그림을 그릴 수 있습니다.

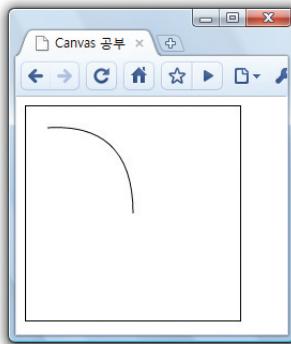
우선 간단한 2차 베지어 곡선을 그려 보겠습니다.

예제 2.52 2차 베지어 곡선 기본 사용 예

```
<script type="text/javascript">
    var make_canvas = function() {
        var exam = document.getElementById('exam');
        var ctx = exam.getContext('2d');

        // 2차 베지어 곡선 그리기
        ctx.beginPath();                                // 새 패스
        ctx.moveTo(20, 20);                            // 곡선의 시작 좌표
        ctx.quadraticCurveTo(100,15,100,100);          // 2차 베지어 곡선 생성
        ctx.stroke();                                  // 완성된 패스 출력
    };
</script>
```

브라우저로 확인해 보겠습니다.



◆ 그림 2.30 간단하게 그려 본 2차 베지어 곡선

가로 20픽셀, 세로 20픽셀에서 패스를 그리기 시작하여 가로 100픽셀, 세로 100픽셀에서 곡선을 끝냅니다. 컨트롤 포인트는 가로 100픽셀 세로 15픽셀에 위치해 있습니다. 오른쪽으로 휘어 있는 곡선이 만들어졌습니다. 직접 좌표값을 바꾸어 가면서 2차 베지어 곡선 그리기를 익혀 보세요.

이제 간단한 3차 베지어 곡선을 하나 그려 보겠습니다.

예제 2.53 3차 베지어 곡선 기본 사용 예

```
<script type="text/javascript">
    var make_canvas = function() {
        var exam = document.getElementById('exam');
        var ctx = exam.getContext('2d');

        // 3차 베지어 곡선 그리기
        ctx.beginPath();                                // 새 패스
        ctx.moveTo(20, 20);                            // 곡선의 시작 좌표
        ctx.bezierCurveTo(50,15,100,15,100,100);      // 3차 베지어 곡선 생성
        ctx.stroke();                                  // 완성된 패스 출력
    };
</script>
```

컨트롤 포인트 좌표가 하나 더 추가된 것 말고는 2차 베지어 곡선과 사용 방법은 동일합니다. 이 코드 역시 직접 타이핑해서 브라우저로 확인해 보기 바랍니다.

베지어 곡선을 연달아 사용하면 복잡한 그림을 그릴 수 있습니다.

요소 합성하기

canvas에서는 다양한 방식으로 그려진 요소들을 합성할 수 있습니다. 이 섹션의 공부를 위해서 간단한 도형을 하나 그려 보겠습니다. 이미 널리 알려진 canvas 합성용 치트시트의 모양을 참고해서 만들어 보겠습니다. 원과 네모가 겹쳐져 있고, 뺨간색과 파란색이 겹쳐져 있기 때문에 다른 도형을 쓰는 것보다도 처음 공부하는 분들은 이해하기 쉬울 것입니다.

예제 2.54 사각형과 동그라미를 겹쳐지게 드로잉

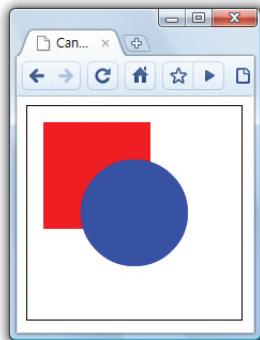
```
<script type="text/javascript">
    var make_canvas = function() {
        var exam = document.getElementById('exam');
        var ctx = exam.getContext('2d');

        // 뺨간색 사각형 그리기
        ctx.fillStyle = "#ff0000";
        ctx.fillRect(15, 15, 100, 100);
```

```
// 파란색 동그라미 그리기
ctx.fillStyle = "#0000ff";
ctx.beginPath();
ctx.arc(100, 100, 50, 0, Math.PI*2, false);
ctx.fill();
};

</script>
```

마크업은 앞선 예제에서 사용하던 것을 계속하여 사용하겠습니다. 이렇게 캔버스 용 스크립트를 만들고 브라우저로 확인해 보겠습니다.



☞ 그림 2.31 캔버스의 합성 학습을 위해서 만든 기본 도형입니다.

자, 아주 유명한 도형입니다. 캔버스의 합성 예제를 말할 때 빠지지 않고 등장하는 겹쳐진 네모와 동그라미입니다. 이제 이 도형들의 겹쳐진 부분을 합성해 보겠습니다. 합성을 위해서 코드를 한 줄 추가합니다.

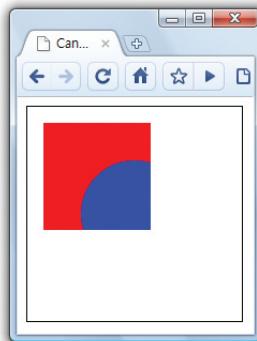
예제 2.55 두 도형이 겹쳐지는 부분의 합성

```
<script type="text/javascript">
    var make_canvas = function() {
        var exam = document.getElementById('exam');
        var ctx = exam.getContext('2d');

        // 빨간색 사각형 그리기
        ctx.fillStyle = "#ff0000";
        ctx.fillRect(15, 15, 100, 100);
```

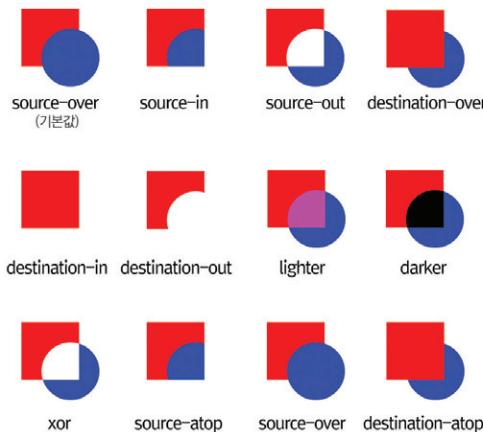
```
// 합성  
ctx.globalCompositeOperation = "source-in";  
  
// 파란색 동그라미 그리기  
ctx.fillStyle = "#0000ff";  
ctx.beginPath();  
ctx.arc(100, 100, 50, 0, Math.PI*2, false);  
ctx.fill();  
};  
</script>
```

'globalCompositeOperation' 속성을 사용합니다. 이 예제에서는 'source-in' 값을 사용하였습니다. 브라우저로 테스트해 보겠습니다.



☞ 그림 2.32 합성 타입을 'source-in'으로 하여 브라우저로 확인해 본 모습입니다. 겹쳐지는 부분을 제외하고 원의 나머지 부분이 잘려 나깁니다.

합성하는 방법이 간단하죠? 'globalCompositeOperation' 속성에서 사용할 수 있는 값은 다음 그림과 같습니다. 여러분도 여기 있는 값을 직접 입력해 보면서 공부하기 바랍니다.



☞ 그림 2.33 캔버스에서 지원하는 다양한 합성 속성의 결과물

원, 호 구현하기

이번에는 원이나 부채꼴과 같은 호를 간단하게 그려 보겠습니다.

`arcTo` 메소드를 사용하면 직선을 기반으로 한 곡선을 그을 수 있습니다. `arcTo` 메소드의 기본 사용 방법은 다음과 같습니다.

```
ctx.arcTo(x1, y1, x2, y2, radial);
```

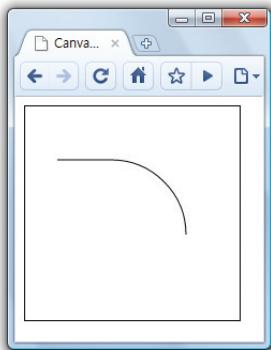
`x1`은 선이 시작하는 위치에서 선이 가로로 뻗을 만큼의 길이, `y1`은 선이 시작하는 부분의 세로 위치, `x2`, `y2`는 선이 끝나는 위치입니다. `radial`값을 조정해서 선을 둥글게 할 수 있습니다. 우선 다음 예제를 보겠습니다.

예제 2.56 직선 끝의 곡선 처리 방법의 예

```
var make_canvas = function() {
    var exam = document.getElementById('exam');
    var ctx = exam.getContext('2d');

    ctx.beginPath();
    ctx.moveTo(30, 50);
    ctx.arcTo(150, 50, 150, 100, 70);
    ctx.stroke();
};
```

예제 2.56 코드를 브라우저로 확인해 보겠습니다.



☞ 그림 2.34 arcTo를 이용해서 만든 선입니다.

예제 2.56은 `moveTo`를 이용해서 그림을 그리기 위한 펜의 위치를 가로 30, 세로 50픽셀 위치로 가져왔습니다. 여기서 `arcTo`를 이용해서 세로 50픽셀 위치에서 선을 그어 가로 30픽셀에서 150픽셀만큼 오른쪽으로 선이 더 그어지게 하였습니다. 그리고 선이 끝나는 부분은 세로 100픽셀만큼 아래로 내려가고 둑글기의 값은 70 정도를 주었습니다. 이 값을 조작해 가면서 익혀 보기 바랍니다.

직선과 곡선을 그려주는 `arcTo` 메소드를 알아보았습니다. 이제는 원을 그려 주는 `arc` 메소드에 대해서 알아볼 차례입니다. `arc`의 기본적인 사용법과 받아오는 값은 다음과 같습니다.

원 그리기 | context.arc 구성

```
ctx.arc(x, y, radius, startAngle, endAngle, anticlockwise);
```

원을 그리기 위해서 사용하는 메소드는 `arc`입니다. `arc` 메소드에 들어가는 각 값에 대해서 설명하겠습니다. `x`와 `y`는 원의 중심이 위치할 좌표입니다. `radius`에는 원의 반지름값을 지정해 줍니다. `startAngle`과 `endAngle`은 원 그리기를 시작하고 끝내는 부분의 각도를 지정해 줍니다. 라디안을 사용해야 하는데 라디안을 쉽게 이해하는 방법을 알려면 '라디안 쉽게 이해하기' 텁을 읽어 보세요. `anticlockwise`

는 'true' 와 'false' 2개 값을 가집니다. anticlockwise값을 'true' 로 지정해 주면 시계반대 방향으로 원을 그려 나가고 'false' 로 지정해 주면 시계방향으로 원을 그려 나갑니다.



라디안 쉽게 이해하기

'라디안'이라는 개념은 다소 어렵지만, 이해하기 쉬운 방법이 있습니다. $\text{Math.PI}/180$ 가 1도의 각도를 가진다고 생각해 보세요. 당연히 이 1도의 라디안에 10을 곱하여 다음과 같은 식을 만들면 $10 * \text{Math.PI}/180$ 라디안은 10도가 되겠습니다. 90도를 만들기 위해서는 $90 * \text{Math.PI}/180$ 라디안 식을 만들어 주면 됩니다.

다음 예제는 간단한 원을 그리는 소스입니다.

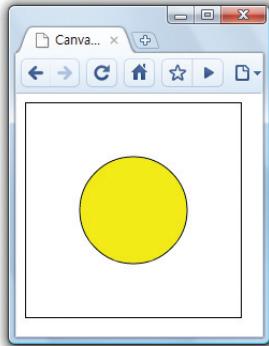
예제 2.57

간단한 원 그리기의 예

```
var make_canvas = function() {
    var exam = document.getElementById('exam');
    var ctx = exam.getContext('2d');

    // 간단한 원 그리기
    ctx.strokeStyle = "#000000";
    ctx.fillStyle = "#ffff00";
    ctx.beginPath();
    ctx.arc(100, 100, 50, 0, 360*Math.PI/180, false);
    ctx.closePath();
    ctx.fill();
    ctx.stroke();
};
```

검정색 실선과 노란색으로 채워진 원을 그릴 것입니다. 원의 중심점은 가로 100픽셀, 세로 100픽셀 좌표에 위치합니다. 반지름이 50, 지름이 100픽셀짜리인 원이 만들어질 것입니다. 라디안을 360도짜리로 만들어 온전한 원이 그려지게 했고, 원은 시계방향으로 그릴 것입니다.



◆ 그림 2.35 노란색 원이 만들어졌습니다.

이 그림은 예제 2.57을 브라우저에서 확인한 모습입니다.

여러분 나름대로 값을 이리저리 조절해 보면서 원 그리기를 숙달해 보기 바랍니다. 부채꼴을 그리기 위해서는 `moveTo()` 메소드를 동원해야 합니다. 일단 다음 예제 코드를 살펴보겠습니다.

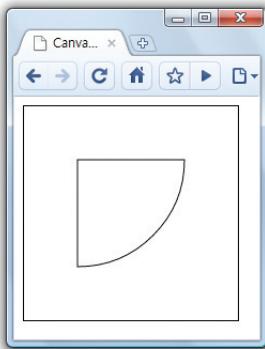
예제 2.58 `moveTo()` 메소드를 동원하여 부채꼴 그리기의 예

```
var make_canvas = function() {
    var exam = document.getElementById('exam');
    var ctx = exam.getContext('2d');

    // 부채꼴 그리기
    ctx.strokeStyle = "#000000";
    ctx.beginPath();
    ctx.moveTo(50, 50);
    ctx.arc(50, 50, 100, 0, 90*Math.PI/180, false);
    ctx.closePath();
    ctx.stroke();
};
```

`arc` 부분의 코드를 보면 원의 중심좌표는 가로 50픽셀, 세로 50픽셀 부분에 위치합니다. 원은 시계방향으로 90도만큼만 돌고 반지름 100픽셀짜리로 그리게 됩니다. 그런데 이렇게만 하면 원의 바깥 부분은 정확하게 그려지지만 안쪽 부분은 제

대로 표현되지 않습니다. 그래서 `moveTo()` 메소드를 동원했습니다. `moveTo()` 메소드로 원을 그리기 전에 펜의 위치를 가로 50픽셀, 세로 50픽셀 위치에 놓습니다. 이렇게 하면 부채꼴이 다음 그림과 같이 제대로 그려지게 됩니다.



☞ 그림 2.36 완성된 부채꼴

텍스트 구현하기

canvas에 텍스트를 작성하고 싶을 때도 있을 것입니다. 이번에는 canvas 요소 내에서 텍스트를 작성해 보겠습니다.

☞ 표 2.10 브라우저별 canvas 텍스트 API 지원 현황

인스플로러	파이어폭스	사파리	크롬	오페라
9.0 버전부터 지원 할지 미지수	3.5 버전부터 지원	4.0 이상 버전부터 지원 할지 미지수	2.0 버전부터 지원	10.5 버전부터 지원

먼저 텍스트를 입력한 간단한 코드를 확인해 보고 공부를 계속하겠습니다. HTML 마크업은 캔버스를 공부하는 이 섹션 초반부에서부터 계속 사용하고 있는 마크업으로 하겠습니다. 스크립트만 다음과 같이 넣어 보겠습니다.

예제 2.59 캔버스에서 기본적인 텍스트 구현의 예

```
<script type="text/javascript">
var make_canvas = function() {
    var exam = document.getElementById('exam');
    var ctx = exam.getContext('2d');

    // 속이 꽉 찬 텍스트
    ctx.font = "20px Malgun Gothic";
    ctx.textAlign ="center";
    ctx.fillText("헬로 월드!", 100, 50);

    // 속이 텅 빈 텍스트
    ctx.font = "25px Arial Black";
    ctx.textAlign ="center";
    ctx.strokeText("Hello! World!", 100, 110);
};

</script>
```

캔버스에 두 개의 텍스트를 넣어 본 코드입니다. 이렇게 코드를 만들고 브라우저로 확인해 보면 다음 그림과 같습니다.



그림 2.37 기본 텍스트를 캔버스에 그려낸 모습입니다.

우선 살펴볼 것은 ctx.font 속성입니다. 이 속성은 폰트의 정보를 지정해 줄 수 있는데 CSS에서 사용하는 'font' 속성과 사용 방법은 동일합니다. 만약에 굵은 20픽셀짜리 굴림체를 만들고 싶다면 다음과 같이 지정해 주면 됩니다.

예제 2.60 캔버스의 font API 사용 방법

```
ctx.font = "bold 20px Gulim";
```

말한 대로 CSS의 font 속성과 동일하다는 것을 알 수 있습니다. canvas 폰트 API의 font 속성이 가지는 폰트의 기본값은 10픽셀의 산세리프체입니다. 한글이라면 굴림체로 나올 것입니다.

다음 그림에 있는 선에는 텍스트의 가로 정렬을 위한 textAlign 속성이 선언되어 있습니다. 이 속성 역시 CSS와 문법적으로는 비슷하지만 결과는 약간 다르게 나옵니다. 다음 그림에서 캔버스 텍스트 API의 텍스트 정렬을 한눈에 볼 수 있습니다.



☞ 그림 2.38 'ctx.textAlign' 속성에 어떤 정렬 방식을 적용하느냐에 따라 달라지는 텍스트의 포지션입니다.
파란색 선은 기준선입니다.

보면 알겠지만 CSS와는 방향 조작 방법이 약간 다릅니다. CSS의 경우에는 텍스트 정렬을 'left'로 해 주면 텍스트가 왼쪽으로 흐 하고 이동합니다. 그러나 캔버스의 텍스트 정렬은 조금 다릅니다. 텍스트 자체가 왼쪽으로 가는 것이 아니라 기준선을 중심으로 텍스트의 왼쪽 끝 부분이 왼쪽에 딱 붙습니다. 이 속성의 기본값은 'start'입니다.

텍스트의 가로 정렬에 대해서 알아보았습니다. 캔버스에서 작성하는 텍스트는 세로 정렬도 해 줄 수 있습니다. 사용 방법은 다음과 같습니다.

예제 2.61 캔버스 텍스트의 baseline 설정 방법의 예

```
ctx.textBaseline = "top";
```

가로로 정렬하는 방법과 마찬가지로, 기본적인 사용 방법은 간단합니다. 이 속성에 어떤 값을 주느냐에 따라 텍스트의 세로 정렬 위치가 바뀝니다. 물론 어떤 문

자를 쓰는지에 따라서도 다소 차이가 있습니다. 다음 그림을 참고해 주세요.



☞ 그림 2.39 베이스라인에 설정된 값에 따라 텍스트의 기준선이 달라지는 모습입니다. 왼쪽의 경우는 파이어폭스, 오른쪽은 크롬과 사파리의 출력 결과 모습인데, hanging의 포지션이 완전히 정반대임을 볼 수 있습니다. Ideographic의 베이스라인 간격도 약간 차이가 있습니다.

베이스라인에 설정된 값과 문자의 종류에 따라 텍스트가 어떻게 위치하는지 알 수 있을 것입니다.



☞ 그림 2.40 텍스트 베이스라인 기준 (이미지 출처 : W3C)

이 그림은 W3C의 유명한 이미지입니다. 각 문자 크기별로 베이스라인이 어떻게 위치하고 있는지 더 정확하게 알아볼 수 있습니다.

텍스트에 색상을 입히는 방법은 앞에서 네모와 같은 기본 도형들을 그릴 때의 방법과 동일한 방법을 사용하면 됩니다. ctx.fillText를 사용해서 속이 꽉 찬 텍스트를

사용한다면 ctx.fillStyle에 색상을 지정해 주면 되고, ctx.fillText를 사용해서 속이 비어 있는 텍스트를 사용한다면 ctx.strokeStyle에 색상을 지정해 주면 됩니다.

그림자와 그라데이션 구현하기

캔버스에서 그림자와 색상을 구현해 보도록 하겠습니다. 먼저 그림자를 구현하는 방법입니다. 그림자를 구현하기 위해 사용되는 4개의 속성을 알아보겠습니다.

- Context.shadowColor : 그림자의 색상을 지정. 16진수와 RGB값 모두 사용 가능합니다.
- Context.shadowOffsetX : 그림자의 가로 위치를 지정해 줍니다.
- Context.shadowOffsetY : 그림자의 세로 위치를 지정해 줍니다.
- Context.shadowBlur : 그림자를 어느 정도 흐리게 할지 정해 줍니다.

예제 2.62 기본적인 그림자 구현 방법의 예

```
<script type="text/javascript">
    var make_canvas = function() {
        var exam = document.getElementById('exam');
        var ctx = exam.getContext('2d');

        // 그림자 구현
        ctx.shadowColor = "#333333";
        ctx.shadowOffsetX = "3";
        ctx.shadowOffsetY = "3";
        ctx.shadowBlur = "5";

        ctx.font = "40px Arial Black";
        ctx.textAlign = "center";
        ctx.fillStyle = "#f423d9";
        ctx.fillText("SSONG", 100, 100);
    };
</script>
```

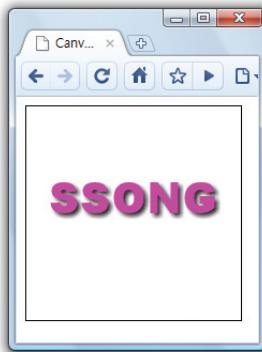
간단하게 텍스트에 그림자를 구현해 본 코드입니다. 그림자 색상은 짙은 회색이고 글자로부터 오른쪽으로 3픽셀, 아래로 3픽셀 밀려나 있는 그림자가 있고, 블러

는 5픽셀을 주었습니다. 블러는 값이 커질수록 더 넓게 퍼지고 흐려집니다. 색상의 경우 이 예제처럼 16진수를 직접 사용하여도 되지만, RGBA값을 사용할 수도 있습니다.

```
ctx.shadowColor = "rgba(0, 0, 0, 0.5);
```

이렇게 쓰면 되는데, 마지막에 있는 숫자는 알파값이고 0부터 1 사이의 값을 주어 가며 조정해서 반투명 효과를 줄 수도 있습니다. RGBA 컬러의 자세한 사용법은 4장 CSS3의 컬러를 컨트롤하는 부분을 참고해 주세요.

브라우저로 확인해 보겠습니다.



☞ 그림 2.41 텍스트에 그림자가 적용되었습니다. 디자인이 예쁘지 않더라도 이해해 주세요. 쉽게 설명하기 위해서 퀄리티가 다소 낮은 색감을 사용했습니다.

canvas에서 색을 칠할 수도 있습니다. 색을 칠하는 것은 앞선 예제들을 만들면서도 배웠지만 fillRect, strokeStyle과 같이 색을 칠하거나 테두리의 색을 지정해 주는 속성을 사용하면 됩니다. 곧바로 그라데이션 사용법에 대해서 설명하겠습니다.

그라데이션을 위해서는 2가지의 메소드를 사용할 수 있습니다.

☞ 표 2.11 그라데이션을 위해 canvas에서 사용할 수 있는 2가지 메소드

메소드	값	역할
createLinearGradient	x0, y0, x1, y1	선형 그라데이션
createRadialGradient	x0, y0, r0, x1, y1, r1	원형 그라데이션

그라데이션을 위한 이 2가지의 메소드는 채워지는 색상을 컨트롤할 때 사용하는 `fillStyle` 속성과 선의 스타일을 컨트롤할 때 사용하는 `strokeStyle` 속성에 적용할 수 있습니다.

먼저 선형으로 그라데이션 색을 사용하기 위해서는 `createLinearGradient` 메소드를 사용해야 합니다. 사용할 수 있는 4가지 값은 색상이 시작되고 끝나는 부분을 지정해 줍니다. 그라데이션을 x_0 , y_0 좌표에서 시작하여 x_1 , y_1 좌표에서 끝냅니다.

예제 2.63 기본적인 선형 그라데이션의 사용 예

```
<script type="text/javascript">
    var make_canvas = function() {
        var exam = document.getElementById('exam');
        var ctx = exam.getContext('2d');

        // 그라데이션을 생성합니다
        var grd = ctx.createLinearGradient(0, 0, 200, 0); // grd 변수에 선형 그라데이션 생성
        grd.addColorStop(0, "#000000"); // 그라데이션 시작색
        grd.addColorStop(1, "#FFFFFF"); // 그라데이션 끝색

        // 텍스트에 그라데이션을 입힙니다
        ctx.fillStyle = grd; // 그라데이션 정보가 있는 grd 색상을 칠합니다.
        ctx.font = "40px Arial Black";
        ctx.textAlign ="center";
        ctx.fillText("SSONG", 100, 100);
    };
</script>
```

그라데이션 정보를 담기 위해서 ‘`grd`’라는 임의의 변수를 만들었습니다. 그라데이션은 선형으로 선택했고 그라데이션 시작 좌표는 $0, 0$ 으로, 끝나는 좌표는 $200, 0$ 으로 설정하였습니다. `canvas`의 왼쪽 상단 모서리에서 시작해서 우측으로 200픽셀만큼 그어지는 그라데이션이 완성될 것입니다. 눈여겨보아야 할 것은 `addColorStop` 메소드입니다. 이 메소드는 그라데이션에서 사용할 색을 설정해 줄 수 있는데, 0에서 1까지의 값을 설정할 수 있습니다. 0이 시작, 1이 끝부분입니다. 이 예시에서는 시작 부분의 색상은 검정색으로, 끝나는 부분의 색상은 하얀색으로 지정해 주었습니다. 끝으로 `fillStyle`에 `grd` 변수를 대입하여 그라데이션 정보

를 뿐만 아니라 주도록 하였습니다. 이 예제를 브라우저로 확인해 보면 다음 그림과 같이 출력될 것입니다.



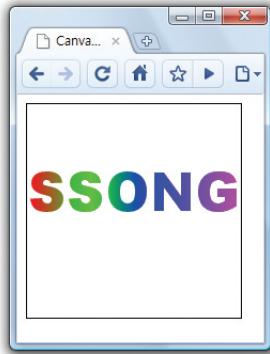
그림 2.42 앞서 만들어 본 그라데이션을 적용한 텍스트 모습

addColorStop 메소드에 대해서 간단한 테스트 하나를 해 보고 넘어가겠습니다. 앞선 예제는 시작과 끝점이 2개뿐인 그라데이션이지만 만약에 여러 개의 변곡점이 있는 그라데이션을 사용하고 싶다면 어떻게 해야 할까요?

예제 2.64 addColorStop 메소드를 이용한 다중 그라데이션 사용법

```
grd.addColorStop(0, "#ff0000");           // 그라데이션 시작색  
grd.addColorStop(0.3, "#00ff00");  
grd.addColorStop(0.6, "#0000ff");  
grd.addColorStop(1, "#ffff00");           // 그라데이션 끝색
```

그라데이션 색을 4개로 추가해 보았습니다. 언제나 시작점은 0, 끝점은 1의 값을 가집니다. 가운데 소수점의 거리만큼 해당 색이 변화하게 됩니다. 이렇게 addColorStop 메소드를 바꾸고 결과물을 브라우저로 확인해 보겠습니다.



☞ 그림 2.43 4개의 색상이 사용된 그라데이션

선형 그라데이션을 사용하는 방법을 알아보았습니다. 원형 그라데이션의 사용법도 이와 비슷합니다.

예제 2.65 원형 그라데이션 기본 사용 예

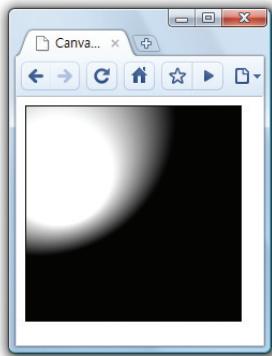
```
<script type="text/javascript">
    var make_canvas = function() {
        var exam = document.getElementById('exam');
        var ctx = exam.getContext('2d');

        // 그라데이션을 생성합니다
        var grd = ctx.createRadialGradient(10, 10, 130, 30, 60, 50); // grd 변수에 원형
                                                               // 그라데이션 생성
        grd.addColorStop(0, "#000");
                                                               // 그라데이션 시작색
        grd.addColorStop(1, "#fff");
                                                               // 그라데이션 끝색

        // 텍스트에 그라데이션을 입힙니다
        ctx.fillStyle = grd;
                                                               // 그라데이션 정보가 있는 grd 색상을 칠합니다
        ctx.fillRect(0,0,200,200);
                                                               // 채워진 사각형을 생성합니다
    };
</script>
```

원형 그라디언트를 위해 사용하는 `createRadialGradient` 메소드는 6개의 값을 가집니다. 앞에 있는 2개의 값은 원의 바깥쪽 그라데이션 좌표를 의미합니다. 좌표는 원의 중심을 기준으로 합니다. 세 번째 값은 끝나는 값의 원의 반지름, 그 다음

값은 안쪽 원의 가로 좌표값과 세로 좌표값입니다. 마지막에 있는 값은 그라데이션이 시작하는 색에 해당하는 원의 반지름입니다. 직접 값을 바꾸어 가면서 시험해 보세요. 이 코드를 브라우저로 확인해 보면 다음과 같은 화면을 얻을 수 있습니다.



☞ 그림 2.44 원형 그라디언트를 캔버스에 뿌린 모습

외부 이미지 삽입하기

이번에는 캔버스에 외부 이미지를 삽입하는 공부를 해 보겠습니다. 외부에 있는 이미지를 삽입하기 위해서는 'drawImage' API를 이용합니다. 이 drawImage를 이용해서 캔버스에 이미지를 삽입하는 방법은 3가지입니다.

```
ctx.drawImage(img, dx, dy);
```

첫 번째 방법은 가장 간단한 방법입니다. 불러올 이미지 파일을 선택하고, 이미지를 위치시켜 줄 가로 dx값과 세로 dy값을 넣어 주면 됩니다. 실제로 다음과 같이 사용할 수 있습니다.

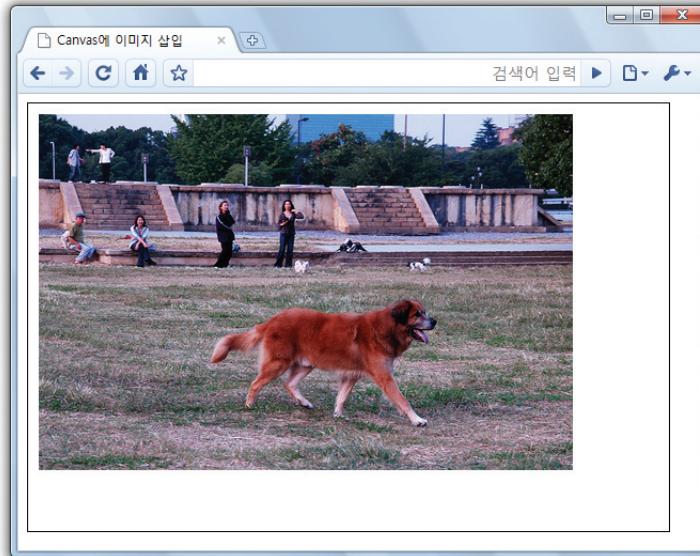
예제 2.66 캔버스에 외부 이미지를 불러오는 가장 기본적인 방법

```
<script type="text/javascript">
    var make_canvas = function() {
        var exam = document.getElementById('exam');
        var ctx = exam.getContext('2d');
```

```
// 이미지 삽입
var doggy = new Image();           // doggy 변수를 이미지 객체로 만들
doggy.src = 'bowwow.jpg';         // doggy에 외부 이미지 주소 입력
ctx.drawImage(doggy, 10, 10);      // 캔버스에 이미지 뿌려 주기
};

</script>
```

캔버스에서 위로 10픽셀, 왼쪽으로 10픽셀 떨어뜨려 강아지 사진을 뿌려 주는 코드입니다. 이 소스를 브라우저에서 확인해 보았습니다.



☞ 그림 2.45 캔버스에 강아지 사진을 삽입해 본 모습

```
ctx.drawImage(img, dx, dy, dw, dh);
```

두 번째 방법으로, 첫 번째 방법에서 dw인자값과 dh인자값이 추가되었습니다. dw에는 사진 이미지의 너비를, dh에는 이미지의 높이를 지정해 주면 됩니다.

예제 2.67 캔버스에 외부 이미지를 불러오면서 사이즈 정해 주기

```
<script type="text/javascript">
    var make_canvas = function() {
        var exam = document.getElementById('exam');
        var ctx = exam.getContext('2d');

        // 이미지 삽입
        var doggy = new Image();          // doggy 변수를 이미지 객체로 만들
        doggy.src = 'bowwow.jpg';        // doggy에 외부 이미지 주소 입력
        ctx.drawImage(doggy, 10, 10, 100, 100); // 캔버스에 이미지 뿌려 주기
    };
</script>
```

앞서 추가한 강아지 사진을 가로 100픽셀, 세로 100픽셀로 만들고 브라우저로 확인해 보았습니다.

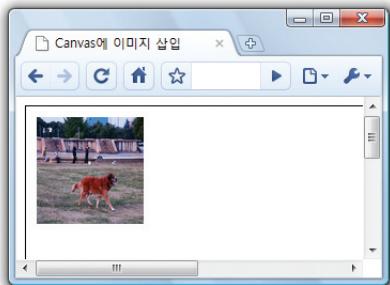


그림 2.46 강아지 사진을 가로 100픽셀, 세로 100픽셀로 바꾸었습니다.

강아지 사진이 가로 100픽셀, 세로 100픽셀을 가지게 되어 정사각형으로 바뀌었습니다. 이때 사진의 비율은 고려하지 않고 사진이 일그러지는 모습을 확인할 수 있습니다.

```
ctx.drawImage(img, sx, sy, sw, sh, dx, dy, dw, dh);
```

이것은 세 번째 방법입니다. 셋 중 가장 복잡한 방법으로 sx, sy, sw, sh 인지값이 추가되었습니다. 어렵게 보이지만 사실 쉽습니다. 여러분들도 잘 알겠지만 포토샵에는 'crop' 기능이 있습니다. 이미지에서 원하는 부분만 잘라내서 쓰는 방법입니다.

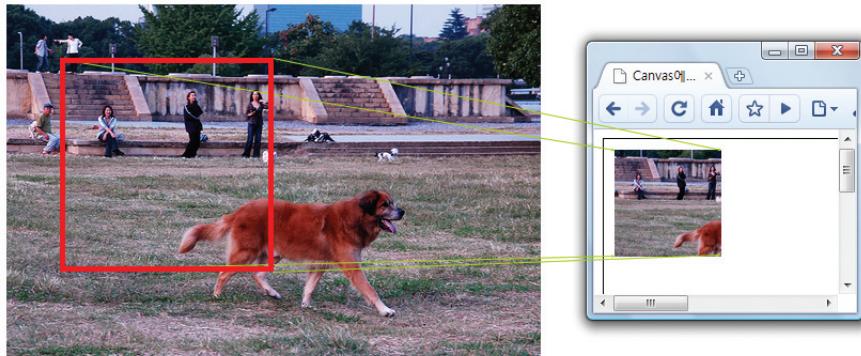
니다. 추가된 4가지 값을 잘 사용하면 바로 이 크롭 기능을 사용할 수 있습니다. sx와 sy에는 이미지를 잘라내기 시작할 부분의 왼쪽 위 모서리 좌표를 지정합니다. 그리고 sw와 sh에는 이미지 잘라내기를 끝낼 부분의 좌표를 지정해 주면 됩니다. 이 좌표들은 원본 이미지를 기준으로 지정해 줍니다.

예제 2.68 외부 이미지의 원하는 부분만 크롭하여 뿌려 주기

```
<script type="text/javascript">
var make_canvas = function() {
    var exam = document.getElementById('exam');
    var ctx = exam.getContext('2d');

    // 이미지 삽입
    var doggy = new Image();           // doggy 변수를 이미지 객체로 만듦
    doggy.src = 'bowwow.jpg';         // doggy에 외부 이미지 주소 입력
    ctx.drawImage(doggy, 50, 50, 200, 200, 10, 10, 100, 100); // 캔버스에 이미지 뿌려 주기
};
</script>
```

위와 같이 사진을 크롭하기 위한 값을 넣고 브라우저로 확인해 보았습니다.



☞ 그림 2.47 강아지 사진을 원하는 부분만 크롭해 보았습니다.

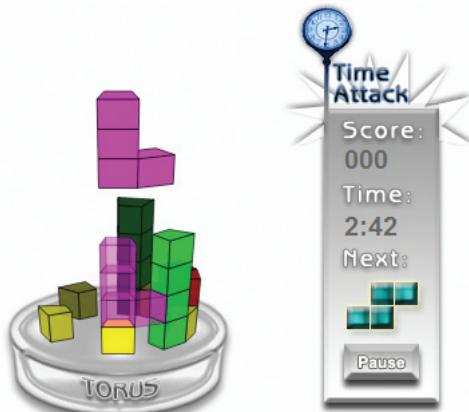
강아지 사진을 크롭한 모습을 보니 이해가 훨씬 잘 되죠? 기억해야 할 것은 크롭을 시작하는 부분에 지정한 50, 50좌표는 사진의 왼쪽 상단 모서리를 기준으로 한다는 것입니다. 그리고 크롭이 끝나는 부분의 좌표 200, 200은 사진의 왼쪽 상단

을 기준으로 하지 않고 크롭이 시작되는 지점을 기준으로 합니다. 그림 2.47에서 크롭된 빨간 박스의 크기는 가로세로 200픽셀입니다.

Canvas 샘플로 살펴보는 무궁무진한 가능성

canvas에서 구현한 놀라운 예제들을 소개하겠습니다. canvas를 지원하는 브라우저의 예제 사이트에 접속해서 예제들을 즐겨 보세요. 그리고 소스를 열어 보고 내 PC로 내려받기해서 직접 분석해 보세요. 자바스크립트의 방대한 소스를 이 책에서 일일이 열거하여 분석하는 과정은 이 책의 컨셉과 맞지 않고 지면상 이유로 넘어가겠습니다. 관심이 많은 독자들은 HTML5 API를 집중적으로 다루는 책과 자바스크립트를 심층적으로 다루는 책으로 공부의 깊이를 더해 보세요.

벤저민 조프의 ‘토러스’

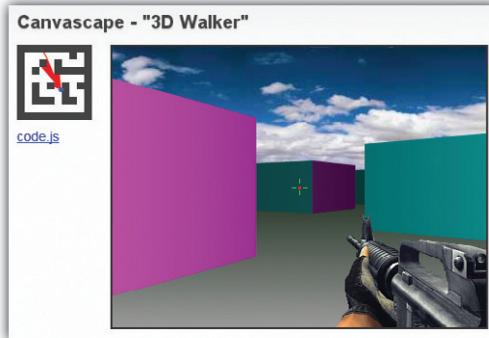


☞ 그림 2.48 3차원 테트리스 토러스

실험적인 작품을 많이 만들어서 유명해진 벤저민 조프의 작품입니다. 토러스는 테트리스와 비슷한 룰을 가지고 있습니다. 키보드의 방향키로 조작하면 됩니다.

<http://www.benjoffe.com/code/games/torus/v1/>

벤저민 조프의 '3D 워커'



☞ 그림 2.49 WebGL 부립지 않은 온라인 스크립트의 결과물

한때 우리나라에서 선풍적인 인기를 끌었던 FPS 게임의 뼈대만 구현했다고 보면 됩니다. 고전 게임 둘과 퀘이크를 기억나게 만드는 예제입니다. 좌측 위에 있는 맵도 정교하게 잘 만들어졌습니다. B를 누르면 총알이 발사되고 스페이스키는 점프, 방향키를 이용해서 캐릭터를 운용하면 됩니다.

<http://www.benjoffe.com/code/demos/canvasscape/>

온라인 포토샵? MugTug



☞ 그림 2.50 너무나 유명한 'MugTug'

거의 완벽에 가까운 웹 애플리케이션입니다. 어지간한 그래픽 툴 못지않은 기능을 제공합니다. 또한 작업을 완료한 이미지는 이미지 파일로 저장도 할 수 있습니다. HTML5의 미래를 가장 잘 보여 주는 웹 애플리케이션의 하나입니다.

<http://mugtug.com/sketchpad/>

라이덴의 향수가… ‘솔리튜드’

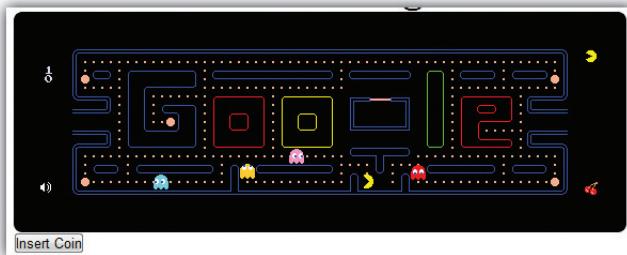


☞ 그림 2.51 캔버스에서 흘러나오는 라이덴의 향수

그 예전 오락실에 동전을 쌓아 놓고 즐기던 라이덴의 향수가 떠오르는 게임입니다. Z키를 누르면 게임이 시작됩니다. HTML5 캔버스에 게임을 훌륭하게 컨버팅하였습니다.

<http://www.kesiev.com/akihabara/demo/game-solitude.html>

팩맨



☞ 그림 2.52 실제 구글 메인페이지 로고를 장식했던 팩맨

팩맨 30주년 기념으로 구글에서 만든 게임입니다. 한동안 전 세계 구글 첫 페이지의 로고를 장식했었습니다. 그때도 작동이 되었습니다.

<http://html5games.net/game/google-pacman/>

구글의 웨이크2 시연 화면



☞ 그림 2.53 구글에서 GWT를 이용해 브라우저에서 구현한 웨이크2

HTML5에 조금이라도 관심이 있는 독자에게는 이미 성지가 된 영상입니다. 구글에서 만든 이 샘플에는 HTML5의 새로운 API들이 총동원되었습니다. 그리고 WebGL 베이스로 캔버스의 3D 컨텍스트를 적극 활용한 샘플입니다. 이 샘플 역시 HTML5의 미래를 잘 보여 주는 샘플이라고 생각합니다. 언제가 될지는 속단할 수 없지만 온라인 게임도 웹 애플리케이션 형태로 돌아가는 세상이 언젠가는 올 것입니다.

<http://www.youtube.com/watch?v=XhMN0w1ITLk>

HTML5와 관련해서 웹서핑을 하다 보면 더 훌륭한 예제들을 많이 접할 수 있을 것입니다. 몇 가지 예제들만 소개하였는데 처음 접하는 독자들은 ‘굉장하다’고 생각할지도 모르겠습니다. HTML5는 이제 걸음마 단계에 있으며, 앞으로 상용화 되는 서비스들이 HTML5를 차용하기 시작하면 기대 이상의 많은 서비스와 기술을 소화해 내는 모습을 볼 수 있을 것입니다.

04 SVG

SVG는 Scalable Vector graphics의 약자로 XML 기반의 드로잉 표준입니다. 캔버스와 비슷한 역할을 한다고 보면 됩니다. SVG는 HTML5와는 별개로 이전부터 사용되던 기술이기 때문에 간단하게 다루고 이 섹션을 마무리하겠습니다.

SVG 소개

캔버스로는 벡터 이미지를 그려낼 수 없습니다. 픽셀 단위의 이미지를 지원하는 웹페이지에서 픽셀의 영향을 받지 않는 이미지를 사용하기 위해서 SVG를 사용합니다. 플래시의 경우가 비슷한 경우에 해당합니다. 또한 SVG는 매우 인터랙티브한 애니메이션이나 벡터 그래픽을 만들어 낼 수 있으므로 유용하게 활용할 수 있습니다.

캔버스에서도 SVG를 읽어들여 표현할 수 있지만 사실 SVG와 캔버스가 직접적으로 관련이 있는 것은 아닙니다. 캔버스와 SVG의 차이점은 무엇일까요? 다음 표를 참고해 주세요.

◆ 표 2.12 드로잉 표준, Canvas와 SVG 비교 도표

	Canvas	SVG
이미지 처리 방식	비트맵	벡터
DOM	존재하지 않음(DOM 컨트롤이 불가능하기 때문에 픽셀 컨트롤)	존재함(외부 스크립트로 각 개체를 컨트롤 가능)
외부 이미지 편집	비트맵 이미지 편집 용이	벡터 이미지 편집 용이
성능	높은 이미지 해상도를 가진 이미지를 사용하면 성능이 저하됨	이미지가 복잡해질수록 마크업이 복잡해지고 성능이 많이 저하됨
애니메이션	애니메이션 API가 없기 때문에 외부 스크립트의 타이머 등 사용	높은 수준의 애니메이션 지원
크로스브라우징	IE를 위한 핵이 존재하나 근본적으로 모든 브라우저에서 지원하지는 않음	모든 브라우저에서 지원되는 드로잉 표준
외부 이미지로 저장	jpg, png 등으로 가능	-
적합한 서비스	그래프 구현, 게임	그래프 구현, 매우 세밀한 해상도를 지원하는 UI, 애플리케이션
적합하지 않은 서비스	독립 애플리케이션 UI	게임



VML

IE8과 그 이전의 버전에서는 SVG 대신에 VML을 지원합니다. 물론 HTC핵을 사용하면 SVG를 IE에서도 사용할 수 있지만 아직 IE에서는 완벽하게 지원하지 않습니다. IE9이 나오면 SVG를 웹 페이지에서 완전히 지원할 것으로 기대해 봅니다.

간단한 SVG 이용 방법

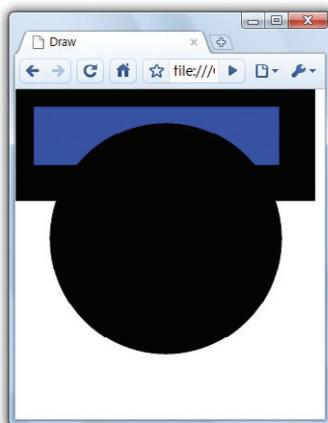
SVG를 이용하는 방법을 간단하게 살펴보겠습니다. SVG를 이용해서 도형을 만들고 간단하게 움직이도록 하면서 캔버스와 SVG에 접근하는 방법이 어떻게 다른지 알아보겠습니다.

예제 2.69 간단한 도형의 간단한 애니메이션 – 파일명 : example.svg

```
<svg version="1.1" width="500" height="500" xmlns="http://www.w3.org/2000/svg">
  <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:1px;
  stroke:rgb(0,0,0)">
    <animate attributeName="stroke-width" values="1px;50px;1px;" dur="2s"
repeatCount="indefinite" />
  </rect>
  <ellipse stroke="#000" cx="50%" cy="50%" rx="50%" ry="50%">
    <animate attributeName="rx" values="0%;50%;0%" dur="2s" repeatCount="indefinite" />
    <animate attributeName="ry" values="0%;50%;0%" dur="2s" repeatCount="indefinite" />
  </ellipse>
</svg>
```

너비 500픽셀, 높이 500픽셀의 SVG 스테이지를 생성합니다. XML 네임스페이스는 w3.org의 경로를 참조하고 있습니다. 너비 300픽셀, 높이 100픽셀의 파란색 박스를 생성하고 그 박스의 테두리가 1픽셀에서 50픽셀로, 다시 1픽셀로 2초 동안 굵기가 변하도록 애니메이션을 적용했습니다. 반복횟수는 무한대입니다. ellipse로 타원형을 만들어 크기가 변하도록 애니메이션을 적용했습니다.

이 소스코드를 확장자 svg 파일로 저장하고 브라우저로 확인해 보면 다음 그림과 같은 결과 화면을 볼 수 있습니다. 동그라미와 네모에 약간의 애니메이션이 적용되어 있습니다. 부드러운 움직임을 볼 수 있을 것입니다.



☞ 그림 2.54 SVG에서 간단한 도형과 간단한 애니메이션을 구현한 모습

SVG는 이렇게 마크업 한줄 한줄로 각 개체를 생성하거나 컨트롤할 수 있습니다. SVG에 대해서 자세히 다룬 책이 시중에 많이 나와 있으니, 자세한 것이 궁금하면 SVG 관련 서적을 탐독해 보기 바랍니다.

05

쉽고 강력해진 HTML5 웹 폼

기존의 HTML4.x 버전이나 XHTML1.x에서는 폼을 이용할 때 불편한 점이 많았습니다. HTML5에서는 불합리한 부분이 많이 개선되었습니다. 새로운 속성과 인풋 타입이 대거 추가되었고, 기존 버전의 HTML 문서들과 호환되고 구형 브라우저를 지원하는 방법도 많아졌습니다. 그러나 여러 UI 개발자나 그 외 개발자들이 우려할 만큼 새로운 폼의 인풋 요소나 속성이 많이 생겼습니다. 형태가 새로운 것도 있고, 기존에는 자바스크립트를 사용해야 가능했던 것이 요소 자체에 내장되어 있는 기능으로 처리되기도 합니다.

새로 추가된 인풋 타입

기존의 인풋 요소에는 텍스트를 입력받는 `text`, 암호를 입력받는 `password`, 라디오버튼, 체크박스, 이미지나 서밋 버튼 등 몇 개 되지 않는 형태가 전부였습니다. 특히 `text` 타입의 경우에는 그 자체만으로는 응용될 수 있는 한계가 너무 명확해서 주변의 프레임워크나 스크립트 언어를 동원해야 하는 경우가 많았습니다.

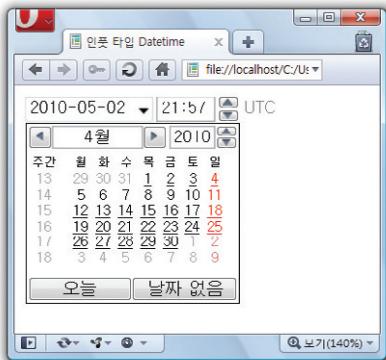
달력을 생성하고 시간을 컨트롤하게 해 주는 `datetime`, `time`, `date` 등

기존에는 달력을 만들기 위해서 꽤 많은 양의 스크립트가 동원되었습니다. 이번에 추가된 `datetime` 인풋 요소 덕분에 이제는 캘린더 같은 컴포넌트를 만들기 위해서 들이던 시간을 다른 데 활용할 수 있게 되었습니다.

예제 2.70 달력 생성

```
<input type="datetime" />
```

가장 기본적인 형태의 `datetime` 태입은 위와 같이 사용하면 끝입니다. 이 코드를 브라우저에서 읽어 보면 다음 그림과 같습니다. 꽤 잘 만들어진 달력과 시간 컴포넌트를 외부 스크립트 등의 지원 없이도 브라우저 내부에서 바로 사용할 수 있습니다.



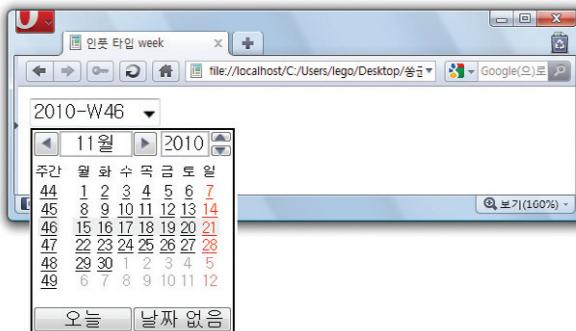
☞ 그림 2.55 HTML5에서 추가된 `datetime` 인풋 타입을 이를 지원하는 오페라 브라우저에서 실행한 모습

인풋 타입을 `datetime`으로 지정하지 않고 `date`라고만 지정하면 시간 부분이 빠지고 달력만 출력하게 됩니다. 일 단위의 날짜를 선택할 수 있습니다.

다음에 소개할 `week` 속성으로 달력의 선택 단위를 주간으로 설정할 수 있습니다.

예제 2.71 주간 단위로만 선택이 가능한 달력

```
<input type="week" />
```



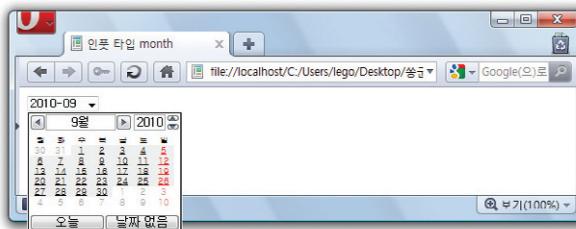
☞ 그림 2.56 인풋 타입을 `week`로 설정한 모습. 달력이 주 단위로 선택됩니다.

기본 시간 표현 방식은 ‘연도–주차’입니다. 예를 들어 2010년 16번째 주라면 2010-W16과 같이 표기되는 것이 기본 표기 방식입니다.

이번에는 인풋 타입 속성을 month로 설정해 보겠습니다.

예제 2.72 월간 단위로 선택이 가능한 달력

```
<input type="month" />
```



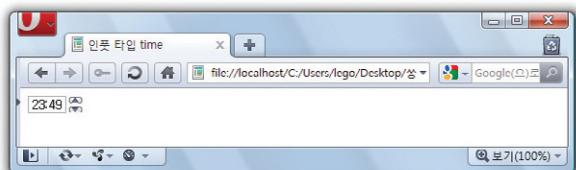
☞ 그림 2.57 인풋 타입 속성을 month로 설정한 모습

인풋 타입 속성을 ‘month’로 설정하면 월 단위의 기간을 선택할 수 있습니다.

이번에는 조금 더 세밀한 설정을 위해 인풋 타입을 ‘time’으로 설정해 보겠습니다. 이 속성을 사용하면 시간값만 입력받을 수 있습니다. 시간과 분 단위의 시간을 입력받을 수 있습니다.

예제 2.73 시간만 선택할 수 있는 기본 시간폼

```
<input type="time" />
```



☞ 그림 2.58 인풋 타입을 time으로 설정하고 오페라 브라우저로 확인한 모습



표준시에 대한 상식

세상에서는 저마다 다른 기준으로 시간을 이용합니다. 그래서 국제 표준시가 필요하게 되었는데, 대표적인 것이 GMT와 UTC입니다. 우리나라에서는 동경135°를 기준으로 한 KST를 사용하고 있습니다. 일본 표준시 JST를 따라가고 있기 때문에 실제 시간보다 약 32분 정도 빠른 시간을 사용하고 있고, GMT보다 9시간 빠른 시간을 가지고 있어 GMT+9로 표기하기도 합니다.

이메일 형식을 체크해 주는 email

사용자가 이메일 주소를 입력했을 때 그 이메일 주소가 유효한 것인지 판단하는 기준의 방법은 매우 성가신 것이었습니다. 불필요한 코드 또한 많아 리소스 낭비를 초래했습니다. 기본적인 텍스트 인풋 박스를 넣고 거기에 자바스크립트를 연동하였습니다. 그 자바스크립트는 이메일 주소가 유효한지 여부를 판단하여 여러 가지 액션을 취할 수 있도록 만들었습니다.

그러나 HTML5에서는 그런 불필요한 작업을 하지 않아도 됩니다. 바로 이메일 유효성 검사를 해 주는 속성이 추가되었기 때문입니다. 사용 방법은 간단합니다.

예제 2.74 이메일 형식을 받는 email 속성

```
<input type="email" />
```

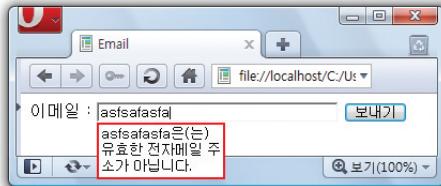
인풋 타입에 ‘email’이라고 넣어 주기만 하면 됩니다. 그러면 사용자가 입력한 텍스트가 유효한 이메일 주소인지 자동으로 체크해 줍니다. 작동 원리는 '@' 와 '.' 기호의 양쪽에 어떤 텍스트 문자열이 있는지를 판별하여 액션을 취할지 말지를 결정합니다.

마크업을 하나 만들어 보겠습니다.

예제 2.75 이메일 형식을 서밋하기 위한 마크업

```
<form>
    이메일 :
    <input type="email" name="user_email" />
    <input type="submit" value="보내기" />
</form>
```

다음과 같이 HTML 마크업 소스를 만들었습니다. 오페라에서 테스트를 해 보겠습니다.



☞ 그림 2.59 유효하지 않은 이메일 주소를 입력해 보았습니다.

이 그림에서 볼 수 있듯이 이메일 주소가 올바르지 않으면 경고 문구를 띠워 줍니다. 뿐만 아니라, 폼을 작동시키지도 않습니다. 올바른 이메일 주소를 입력해 주어야 폼이 값을 올바르게 정해진 곳으로 넘겨줍니다.



☞ 그림 2.60 이메일 필드를 아이폰에서 선택하면 나타나는 아이폰 키패드. '@' 와 '.' 을 쉽게 입력할 수 있게 되어 있습니다.

URL 체크

URL 주소 체크의 사용법 역시 앞서 보았던 이메일 체크 사용법과 동일합니다. 이것 역시 유효한 URL 체크를 위한 외부의 스크립트를 동원하지 않고도 URL 체크를 해 주는 속성입니다. 기본적으로 다음과 같이 사용합니다.

예제 2.76 URL 형식을 체크해 주는 URL 속성

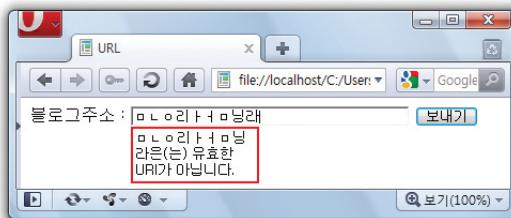
```
<input type="url" />
```

인풋 타입에 'url' 이라고만 입력해 주면 됩니다. 마크업을 하나 해서 오페라를 통해서 테스트를 해 보겠습니다. 여러분도 직접 마크업 코드를 따라 만들어 보세요.

예제 2.77 URL 서밋을 하기 위한 마크업

```
<form>
    블로그주소 :
    <input type="url" name="user_url" />
    <input type="submit" value="보내기" />
</form>
```

이 HTML 마크업을 오페라로 확인해 보겠습니다. 결과 화면은 다음 그림과 같습니다.



☞ 그림 2.61 인풋 타입을 URL로 하여 오페라에서 테스트한 모습

이 인풋 타입은 'http://' 존재 여부를 체크합니다. 위의 그림과 같이 올바른 URL이 아닌 경우 포커스를 잃는 순간 'http://'를 자동으로 붙여 줍니다.



☞ 그림 2.62 URL 인풋 속성을 아이폰에서 선택하면 키패드 모양은 다음과 같습니다. '/' 와 '.com' 버튼이 눈에 띕니다.

컬러 팔레트를 제공해 주는 color 속성

HTML5 폼에서는 컬러피커를 제공합니다. 컬러피커의 사용 방법 역시 간단합니다.

예제 2.78 컬러피커를 생성하기 위한 color 속성

```
<input type="color" />
```

컬러피커는 아직 거의 모든 브라우저에서 완벽하게 지원하지 못하고 있습니다. 그러나 많은 브라우저가 점차 컬러피커의 기능을 제공할 것으로 전망됩니다. 미리 익혀 두면 유용하게 쓰일 것입니다.



☞ 그림 2.63 블랙베리의 웹킷 브라우저에서 실행한 컬러피커의 예 (가상 화면)

검색을 위한 search 속성

블로그를 운영하는 운영자들은 블로그에 있는 글을 쉽게 찾아볼 수 있도록 검색박스를 제공합니다. 또 자그마한 쇼핑몰에서도 사용자가 물건을 쉽게 찾을 수 있도록 검색박스를 제공합니다. 이런 검색박스들은 일반 텍스트 인풋 박스로 만들어져 있습니다. 마크업으로 보면 다음과 같이 되어 있을 것입니다.

예제 2.79 기본적인 텍스트 필드

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

이 방법은 거의 모든 웹사이트에서 사용되고 있습니다. 그렇지만 앞으로 검색박스는 다음과 같이 마크업을 고쳐서 쓰는 것이 좋을 것입니다.

예제 2.80 검색을 위한 search 속성

```
<input type="search" />
```

애플사의 웹사이트(apple.com)에 접속해 보면 우측 상단에 검색박스가 있습니다. 애플사에서는 이 검색박스에 이미 새로운 방식의 검색박스 ‘search’ 인풋 타입을 사용하고 있습니다. 이 방식이 훨씬 의미에 맞는 방식입니다. 만약 이 인풋 타입을 지원하지 않는 브라우저로 사용자가 접속한다면 일반 텍스트 박스로 출력됩니다.

예제 2.81 검색 필드를 서밋하기 위한 마크업

```
<form>
  <input type="search">
  <input type="submit" value="검색">
</form>
```

여러분들도 직접 코드를 타이핑해서 브라우저에서 테스트해 보세요.

외관으로 일반 텍스트 필드와 검색 필드를 구별하는 가장 쉬운 방법은 박스 오른 쪽의 ‘x’ 표시입니다. 검색창에 텍스트를 입력해 보면 다음 그림과 같이 ‘x’ 박스가 나타남을 알 수 있습니다. ‘x’ 아이콘을 클릭하면 텍스트를 초기화합니다.



☞ 그림 2.64 일반 텍스트와 외관상 구별되는 점은 텍스트를 입력하면 우측에 ‘x’ 아이콘이 생긴다는 점입니다. (크롬에서 테스트)

특정 숫자값만 입력받고자 할 땐 number

종종 정해진 범위 내의 특정한 숫자값만 입력받아야 하는 경우가 있습니다. 그런 경우에 유용하게 사용할 수 있는 인풋 타입 속성입니다. 기본적으로 다음과 같이 사용합니다.

예제 2.82 숫자만 입력받기 위한 number 타입 속성

```
<input type="number" />
```

number 태입은 공용 속성 이외에 다음과 같은 4가지 속성을 사용할 수 있습니다.

☞ 표 2.13 인풋 태입 number 속성과 함께 사용 가능한 추가 속성

속성	받는 값	의미
min	숫자	입력받을 수 있는 최소값을 정의합니다.
max	숫자	입력받을 수 있는 최대값을 정의합니다.
step	숫자	게이지 버튼을 조작할 때 숫자가 얼마씩 증가 혹은 감소하게 할지 지정합니다.
value	숫자	기본값을 지정합니다.

이 4가지 속성을 사용하여 HTML 마크업을 해 보겠습니다.

예제 2.83 특정 구간의 숫자만 입력받기 위한 속성값 부여

```
<input type="number" min="12" max="60" step="6" value="24" />
```

이 마크업을 설명하겠습니다. 이 인풋 필드는 숫자값만을 입력받습니다. 입력받을 수 있는 가장 작은 값은 12입니다. 11이하의 값은 입력할 수 없습니다. 입력받을 수 있는 최대값은 60입니다. 61이상의 값은 입력할 수 없습니다. 값을 올리거나 내리는 게이즈를 조절하면 6단위로 값이 조절됩니다. 12, 18, 24, 이런 식으로 값이 조정될 것입니다. 이 필드가 로드될 때 초기에 입력된 기본값은 24입니다. 오페라 브라우저로 확인해 보면 다음 그림과 같습니다.



☞ 그림 2.65 number 요소를 오페라에서 확인한 모습

수치를 제공된 슬라이드 막대에서 선택하도록 해 주는 range

앞서 알아보았던 number 속성이 시각적으로 발전한 단계라고 볼 수 있습니다.

range 속성을 사용하면 간편하게 수치를 제어하는 막대를 만들어 낼 수 있습니다. 최소값과 최대값을 지정하여 특정한 구간의 값을 사용자가 선택하게 하도록 할 수 있습니다.

예제 2.84 가장 기본적인 range 속성 사용

```
<input type="range" />
```



☞ 그림 2.66 인풋 타입의 range 속성을 브라우저로 확인한 모습

range 속성은 min과 max를 이용하여 최소값과 최대값을 설정하여 게이지 값을 지정해 줄 수 있습니다.

예제 2.85 최소값과 최대값 부여

```
<input type="range" min="0" max="10" />
```

이 예제에서 슬라이더는 최소값으로 0을 가지게 되고, 최대값으로 10을 가지게 됩니다. 슬라이더는 기본적으로 10등분될 것입니다. 10단계의 값 중 하나를 이용자가 선택할 수 있습니다.

예제 2.86 짹수만 입력 가능

```
<input type="range" min="0" max="10" step="2" />
```

이렇게 step이라는 속성에 2라는 값을 주게 되면 슬라이드 게이지는 이등분됩니다. 최소값이 0이고 최대값이 10으로 설정되어 있으니까 게이지는 0이나 5, 10을 선택할 수 있게 될 것입니다.

예제 2.87 기본적으로 2가 선택되어 있도록 함

```
<input type="range" min="0" max="10" value="2" />
```

이 예제에서는 `value`라는 속성이 추가되었습니다. `value` 속성에 2라는 값을 주었습니다. 이 예제를 브라우저에서 실행해 보면 최소값은 0, 최대값은 10인 10조각을 가진 슬라이드 막대가 생깁니다. 그리고 `value`에서 지정해 준 값 2에 게이지가 기본적으로 위치해 있음을 확인할 수 있습니다.

전화번호를 입력받고자 할 때

`tel` 속성은 전화번호를 입력받고자 할 때 사용할 수 있습니다. 기본적인 사용법은 다음과 같습니다.

예제 2.88 가장 기본적인 `tel` 속성 사용 예

```
<input type="tel" />
```

`tel` 속성을 테스트해 보기 위한 예제 마크업을 하나 만들어 보겠습니다.

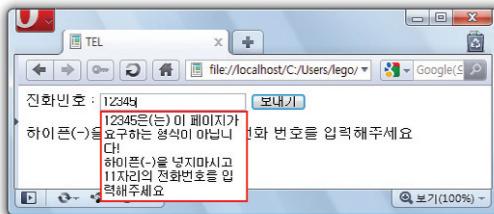
예제 2.89 `pattern` 속성과 정규식을 사용해서 11자리의 숫자만 입력받도록 한 예

```
<form>
    전화번호 :
    <input type="tel" pattern="[0-9]{11}" name="user_tel" title="하이픈(-)을 넣지 말고
    11자리의 전화번호를 입력해 주세요" />
    <input type="submit" value="보내기" />
    <p>하이픈(-)을 넣지마시고 11자리의 전화번호를 입력해주세요</p>
</form>
```

전화번호만 입력받기 위해서 인풋 타입 속성을 ‘`tel`’로 지정하였습니다. 주목해야 할 부분은 `pattern` 속성과 `title` 속성입니다. `pattern` 속성은 사용자가 아무 값이나 입력하지 못하도록 막아 줍니다. 이 코드에서는 오직 11자리의 숫자만 입력받도록 만들었습니다. 정규표현식이라고 불리는 부분인데 HTML5에서는 이와 같이 사용할 수 있습니다. 정규표현식에 대해서는 이 장의 뒷부분에서 간단히 공부하겠습니다.

title 부분에 입력된 문구는 해당 필드에 대한 설명을 담고 있습니다. 필드가 오류 메시지를 출력할 경우 title 부분의 문구를 출력해 줄 수 있습니다.

이 예제 코드를 오페라로 확인해 보겠습니다.



☞ 그림 2.67 방금 만든 예제 코드를 오페라에서 확인한 모습입니다. 잘못된 값을 입력하자 오류 안내 메시지를 자동으로 출력해 줍니다.



☞ 그림 2.68 아이폰이나 아이팟터치에서 TEL 요소에 값을 입력하기 위해 선택할 경우의 키패드의 모양입니다. 전화번호를 입력하기에 좋은 UI입니다.

기능이 강화된 file

file 인풋 태입은 이전 버전의 HTML에도 있었습니다. 다만 이 책에서 소개하는 이유는 file 태입의 기능 중 강화된 부분이 있고, 이 부분이 매우 유용하게 사용될 것이기 때문입니다. 이 file 속성의 가장 강화된 부분은 여러 개의 파일을 한 번에 선택할 수 있다는 것입니다. 이전에는 한 번에 하나의 파일만 선택이 가능했기 때문에 플래시와 같은 외부의 기술을 빌려서 꼼수를 써야 했지만 이제는 그럴 필요가 없어졌습니다.

예제 2.90 여러 개의 파일을 선택할 수 있는 multiple 속성이 적용된 file 속성

```
<input type="file" multiple />
```

나중에 multiple 속성에 대해 설명하겠지만 `multiple="multiple"` 과 같이 사용해도 되고 그냥 ‘multiple’이라고만 써도 됩니다. 이 속성을 첨가하면 한 번에 여러 개의 파일을 업로드할 수 있습니다.

예제 2.91 accept 속성을 사용해서 이미지 파일만 선택할 수 있도록 필터링하는 예

```
<input type="file" accept="image/*" />
```

이 예제 2.91처럼 accept 속성을 사용하면 선택할 수 있는 파일의 종류를 필터링 할 수 있습니다. 이때 필터링할 수 있는 값으로 MIME 타입을 사용하는데, 현재는 거의 모든 브라우저에서 지원하지 않지만 미래에는 ‘image/*’, ‘video/*’와 같은 특별한 MIME 타입도 지원할 것입니다.



MIME 타입

Multi-propose Internet Mail Extentions의 약자로 원래는 이메일 확장 규격이었습니다. 이메일에 첨부파일을 함께 주고받을 때 이 MIME 규격을 사용합니다. 그러던 것이 웹 전방위적으로 사용되는 규격이 된 것입니다. 웹서버가 클라이언트 웹브라우저에 HTTP 헤더를 통해서 여러 가지 정보를 전달할 때 보내지는 자원의 콘텐츠 타입이 어떤 것인지를 MIME 타입으로 알려 줍니다. 가장 기본적인 표준 MIME-Type은 ‘text/html’입니다. 일반 텍스트 파일의 MIME-Type은 ‘text/text’이고 GIF 이미지는 ‘image/gif’과 같은 MIME 타입을 가지게 됩니다. 확장자별로 가지게 되는 MIME 타입이 다양한데 자세한 것은 다음 주소를 참고하세요.

http://www.w3schools.com/media/media_mimeref.asp

인풋 요소에 새로 추가된 속성

HTML5에는 편리한 속성이 많이 추가되었습니다. 대부분 기존에도 사용하던 기능이지만 자바스크립트를 동원하여 복잡한 코드로 처리해야만 하던 부분이 이제는 속성 하나만 지정해 주면 간단하게 해결되는 것이 많아졌습니다. 단, 이 속성들을 실무에 적용하기에는 아직 난제가 있습니다. 속성마다 지원하는 브라우저가 제각각이기 때문입니다. 이 속성들이 자리를 잡으려면 시간이 조금 더 걸리겠지만 기술의 흐름은 급변하니 지금부터라도 미리 배우고 익혀 두면 독자 여러분에게 많은 도움이 되리라 확신합니다.

그러면 HTML5에 추가된 인풋 요소의 속성 중 유용한 몇 가지만 소개하겠습니다.

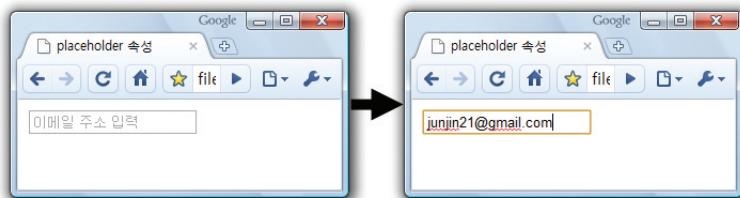
placeholder 속성

HTML5에 추가된 인풋 속성 중 가장 유용한 속성 중 하나입니다. 기존에는 로그인하는 부분에 ‘아이디를 입력하세요’, ‘비밀번호를 입력하세요’와 같은 안내 문구를 이미지로 제작하거나 임의의 텍스트로 넣어야 했습니다. 그리고 사용자가 아이디를 입력하기 위해서 텍스트 필드를 선택하여 포커싱되면 안내문구가 사라지는 효과를 위해 외부의 자바스크립트까지 동원해야 했습니다. 그러나 이제는 placeholder 속성 하나면 그 부분이 말끔하게 해결됩니다. 간단한 사용법을 보겠습니다.

예제 2.92 기본적인 placeholder 속성 사용 예

```
<input type="email" placeholder="이메일 주소 입력" />
```

이렇게 입력하고 브라우저를 통해 확인해 보면 다음 그림과 같습니다.



☞ 그림 2.69 placeholder 속성 구현 모습 (크롬에서 테스트)

최초에는 ‘이메일 주소 입력’이라는 문구가 보입니다. 그러면 사용자는 이메일을 입력해야 하는 공간으로 인식합니다. 그리고 사용자가 이메일을 입력하기 위해 포커싱을 가하면 해당 안내문구는 사라집니다.

autocomplete 속성

독자 여러분도 이런 경험이 많을 것입니다. 회원가입을 하거나 방대한 정보를 입력하다가 페이지가 날아가서 처음부터 다시 입력해야 하는 경우 말이죠. 이런 경우 많은 이용자의 불쾌지수가 올라가고 마음 먹고 회원 정보를 입력하려고 했

던 이들조차 발길을 돌리게 됩니다. `autocomplete` 속성은 그런 일을 막아 주는 속성으로 HTML5에서 새로이 추가되었습니다. 이것 역시 기존에는 외부의 스криプ트나 쿠키를 동원하여 해결을 하긴 했지만 이제는 이 속성 하나로 손쉽게 해결됩니다.

`autocomplete`는 `on`과 `off` 2의 속성값을 가집니다.

- `on` : 입력된 정보를 저장하고 다른 페이지에 갔다가 돌아오는 경우에 그 값을 다시 채워 넣습니다
- `off` : 폼에 입력된 정보를 저장하지 않습니다.

예제 2.93 기본적인 `autocomplete` 속성 사용 예

```
<input type="text" autocomplete="on" />
```

위의 코드를 작성하여 오페라에서 테스트해 보세요. 값을 입력하고 다른 페이지로 갔다가 다시 뒤로 돌아와도 입력한 값이 그대로 살아 있을 것입니다. 이 속성은 텍스트 필드 종류와 같은 요소 외에도 체크박스나 라디오박스와 같은 폼 필드에서도 적용됩니다.

autofocus 속성

네이버나 구글과 같은 검색 사이트에 접속하면 가운데 커다란 검색창이 보입니다. 이 검색창을 굳이 클릭하지 않아도 자동으로 포커싱이 되어 있어서 키워드만 입력하면 되니까 편리합니다. 이 방법 역시 기존에는 자바스크립트를 동원하여 제작을 했습니다. 그러나 HTML5에서 추가된 `autofocus` 속성을 사용하면 손쉽게 이 방법을 구현할 수 있습니다. 페이지가 로드되면 원하는 인풋 요소에 자동으로 포커싱이 되도록 해 보겠습니다.

예제 2.94 기본적인 `autofocus` 속성 사용 예

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

간단하죠? `autofocus`에는 `autofocus` 하나의 값이 붙습니다. 빈 값을 주면 `autofocus`가 작동하지 않습니다.

그럼, 여기서 호기심이 하나 생깁니다. 만일 다음과 같이 모든 인풋 요소에 autofocus가 적용되어 있다면 어떤 인풋 요소에 포커싱이 가 있을까요? 여러분도 브라우저를 통해서 테스트해 보세요. 크롬에서 테스트해 보세요.

예제 2.95 여러 개의 autofocus가 지정된 경우

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

테스트해 보셨나요? 정답은 바로 마지막에 있는 인풋 요소가 포커싱된다는 것입니다. 왜 그렇게 포커싱되는지 약간 아리송합니다. 논리적으로는 가장 앞에 있는 인풋요소가 포커싱되는 것이 맞는 것 같기도 하지만 일단은 정답을 알아 두기 바랍니다.

단, hidden 요소에는 사용할 수 없는 속성으로 hidden 요소에 autofocus를 적용해도 작동하지 않습니다.

list 속성

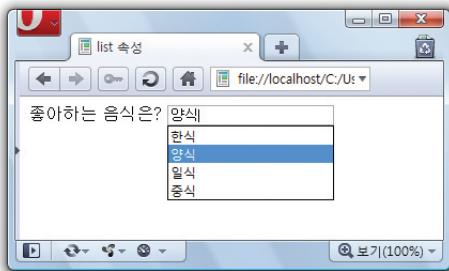
이 속성은 네이버의 검색어 자동완성 서비스와 비슷할 수 있습니다. datalist 요소에서 가지고 있는 값을 사용자에게 미리 뿐려 주는 방식입니다. 한편으로는 셀렉트 박스와 비슷할 수도 있지만, datalist에서 가지고 있는 값을 미리 보여 주어 선택하게 하되 사용자가 임의의 텍스트 값을 입력하도록 할 때 유용하게 사용할 수 있는 요소입니다. 일단 간단한 사용 방법을 살펴보겠습니다.

예제 2.96 datalist를 이용한 자동 완성 사용 예

```
<label> 좋아하는 음식은?
<input list="foods" />
<datalist id="foods">
  <option value="한식">
  <option value="양식">
  <option value="일식">
  <option value="중식">
</datalist>
</label>
```

datalist의 id값을 인풋 요소의 list 속성에서 받아옵니다.

이 예제 코드를 직접 타이핑하여 만들어 보고 오페라에서 테스트해 보세요. 다음 그림과 같이 datalist에 있는 값들이 자동으로 드롭다운되어 출력됩니다.



☞ 그림 2.70 datalist 요소와 결합하여 인풋의 list 속성을 구현한 모습

이 속성의 훌륭한 점은 자바스크립트로 구현하면 꽤 복잡해지는 키보드 액션까지도 훌륭하게 처리한다는 것입니다.

width, height 속성

인풋 타입으로 이미지를 사용할 경우 이미지의 높이와 너비를 지정해 줄 수 있습니다. 높이와 너비를 지정하기 위해 사용할 수 있는 단위는 다음과 같습니다.

- 픽셀 : 단위를 붙이지 않으면 기본값인 픽셀로 높이와 너비를 인식합니다.
- % : 단위에 %를 붙이면 %단위로 너비와 높이를 인식합니다.

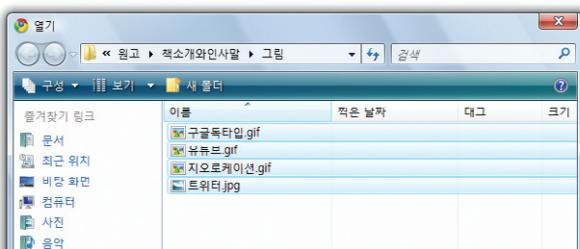
multiple 속성

이 속성은 INPUT 요소 중에서 어떤 아이템(목록, 파일 등)을 선택하는 요소를 대상으로 동작하는 속성입니다. 예를 들어 파일을 업로드하기 위해 인풋 타입 속성으로 'file'을 사용한다고 하면 파일을 업로드할 때 한 번에 여러 개의 파일을 올릴 수 있습니다.

예제 2.97 multiple 속성을 사용한 다중 파일 선택의 예

```
<form>  
    파일 업로드(한꺼번에 여러 개 가능) : <input type="file" name="upload" multiple>  
</form>
```

파일 업로드(한꺼번에 여러 개 가능) : [파일 선택] 선택된 파일 없음



☞ 그림 2.71 파일을 업로드하기 위해 file 인풋 타입을 사용하였고 multiple 속성을 추가하여 한꺼번에 여러 개의 파일을 업로드할 수 있는 모습을 크롬으로 시현한 모습

이 예제에서는 파일을 업로드할 수 있는 'file' 인풋 타입을 사용하였습니다. 코드 마지막 부분에 multiple 속성을 주었습니다. 크롬으로 테스트해 보면 파일을 업로드할 때 한꺼번에 여러 개의 파일을 업로드할 수 있음을 알 수 있습니다. 기존에는 플래시와 같은 외부 기술을 사용해야 가능했던 기능입니다. 이제는 multiple 속성 하나만 추가하면 됩니다.

min, max 속성

이 속성은 인풋 타입을 date나 number, range와 같이 특정 범위 안의 숫자를 컨트롤 가능한 도구로 설정하였을 경우 사용할 수 있습니다. min은 해당 인풋 요소가 가질 수 있는 최소값을, max는 해당 인풋 요소가 가질 수 있는 최대값을 지정해 줍니다. 두 속성 모두 숫자 값만을 받습니다. 일단 간단한 예제를 살펴보겠습니다.

예제 2.98 기본적인 min, max 속성 사용 예

```
<input type="range" min="0" max="2" />
```

최소값은 0, 최대값은 2로 설정된 슬라이더를 만드는 예제입니다. 이 예제 코드를 브라우저로 확인하면 눈금이 3개밖에 없는 것을 볼 수 있습니다. 당연히 max값을

늘려주면 슬라이더는 더 부드러워질 것입니다. 특정 구간에 있는 값만 설정받고 싶을 때 유용하게 사용할 수 있습니다.

슬라이더 말고 달력에도 min, max 속성을 적용할 수 있습니다. 다음 예제 코드를 봐주세요.

예제 2.99 달력에서도 사용할 수 있는 min, max

```
<input type="date" min="2010-01-01" max="2010-02-01">
```

이 예제는 여러분도 직접 타이핑해 보고, 오페라 브라우저에서 확인해 보세요. 인풋 요소로 달력을 사용합니다. 그리고 달력은 2010년 1월 1일부터 2010년 2월 1일 까지만 선택 가능하도록 제한하였습니다. 이와 같이 min, max 속성은 특정 범위를 지정해야 할 때 유용한 속성입니다.

formaction 속성

이 속성은 폼이 데이터를 넘겨줄 페이지를 input 태입선에서 바꾸어 버릴 수 있는 속성입니다.

예제 2.100 formaction 속성을 사용해 데이터를 넘겨줄 페이지를 바꾸는 예

```
<form action="go_america.jsp" method="post">
  <input type="submit" formaction="go_korea.jsp" value="제출" />
</form>
```

이 코드를 보면 폼에서 기본적으로 데이터를 넘겨주는 페이지는 ‘go_america.jsp’ 파일입니다. 그러나 인풋 서밋 버튼에서 폼값이 넘어가는 페이지를 ‘go_korea.jsp’ 페이지로 바꾸어 버렸습니다. formaction 속성은 이와 같이 폼값을 넘기는 기본 목표 페이지를 폼 안에 있는 인풋 태입선에서 바꾸어 줄 수 있는 속성입니다.

required 속성

이 속성은 인풋 요소에 값을 반드시 채워야만 서밋이 작동하도록 합니다. 만일 값이 채워지지 않았다면 서밋이 되지 않습니다. 이 속성은 다음의 인풋 태입들과 함께 사용할 수 있습니다.

required 사용 가능한 인풋 타입 : text, search, password, data pickers, number, checkbox, radio, email, url, telephone, file

예제 2.101 required 속성이 지정되어 있으면 반드시 값을 입력해야만 합니다

```
<form action="demo_form.asp" method="get">  
    찾으실 동 이름을 입력해 주세요 <input type="text" name="usr_name" required="required" />  
    <input type="submit" value="찾기" />  
</form>
```

이렇게 해 두면 텍스트 필드에 아무 값도 입력되지 않은 경우 폼을 서밋하지 않습니다. 즉 아무 반응도 없을 것입니다. 텍스트 필드에 무언가 입력되어야만 폼이 동작할 것입니다.

disable 속성

이 속성은 폼을 비활성화시킵니다. 이 속성은 새로이 추가된 속성은 아니지만 알고 있으면 좋은 속성인데 의외로 모르고 있는 디자이너들이 있어서 임의로 추가하였습니다.

예제 2.102 폼 사용 불능으로

```
<input type="submit" value="검색" disabled>
```

다음과 같이 ‘disabled’ 라고 쓰거나 disabled= "disabled" 라고만 붙여 주면 됩니다. 이 속성은 이렇게 원하는 인풋 박스에 써도 되고 폼을 감싸고 있는 <fieldset> 태그에 사용해도 됩니다.

예제 2.103 필드셋 덩어리를 사용 불능으로

```
<fieldset disabled>  
    <legend>검색하세요</legend>  
    <input type="search">  
    <input type="submit" value="검색">  
    <label><input type="checkbox"> 고급검색</label>  
</fieldset>
```

이 마크업처럼 인풋 폼 요소들을 감싸고 있는 `<fieldset>` 요소에 ‘disabled’ 를 설정해 주면 안에 포함되는 모든 폼 요소가 비활성화됩니다. 실무에서도 편리하게 쓰이고 있습니다. 아쉽게도, 인풋 요소에 일일이 disabled 값을 지정해 주는 것은 모든 브라우저에서 쓸 수 있지만 `<fieldset>`에 적용된 disabled 속성은 오페라에서만 동작합니다.

indeterminate 속성

이 속성은 체크박스에 사용할 수 있습니다. 체크박스에 이 속성을 부여하면 체크한 상태도 아니고 체크하지 않은 상태도 아닌 중간 상태가 됩니다. 이 속성은 스크립트를 이용해서 조절할 수 있습니다.

예제 2.104 체크박스의 중간 상태를 만드는 방법의 예

```
<!DOCTYPE html>
<input type="checkbox" id="chk">
<script>
  document.getElementById('chk').indeterminate = true;
</script>
</html>
```

스크립트로 체크박스의 DOM에 접근하여 indeterminate 속성에 ‘true’ 값을 부여하였습니다.



그림 2.72 체크박스의 indeterminate 속성이 true일 때 모습 (파이어폭스 3.6)

브라우저로 확인해 보면 이 그림과 같습니다. 이제 체크박스의 속성상태가 2가지가 아닌 3가지가 된 셈입니다.

자바스크립트 정규표현식

'자바스크립트 정규표현식'이라는 소제목을 붙였지만 정규표현식은 다른 언어에서도 사용할 수 있는 규약입니다. PHP나 자바 등 정규표현식을 지원하는 다른 언어에서도 큰 차이 없이 사용할 수 있습니다.

앞서 이메일을 체크하거나 URL을 체크하는 등의 편리한 요소들을 공부했습니다. 추가된 새로운 요소들만으로도 웹페이지를 만들거나 UI를 개발할 때 많은 부담을 덜 수 있게 되었습니다. 그러나 만약 이것들 외에 다른 형태의 규칙을 추출해야 한다면 자바스크립트 정규표현식을 사용하면 됩니다. 기존에는 자바스크립트를 이용해서 정규식을 이용할 때 불필요한 요소가 많이 들어갔고 손이 많이 가는 방식으로 정규표현식을 사용해야 했습니다. 하지만 HTML5에서는 조금 더 편리한 방법으로 자바스크립트 정규식을 사용할 수 있습니다.

HTML5 인풋 속성에는 pattern이라는 속성이 추가되었습니다. 이 속성은 자바스크립트의 정규표현식을 지원합니다. 이 속성을 이용하여 자바스크립트의 정규식을 사용하면 예전처럼 많은 노력을 기울이지 않고도 간단하게 원하는 형태의 값인지 체크할 수 있습니다.

예를 들어 우편번호를 체크하고 싶다면 다음과 같은 정규식을 사용하면 됩니다.

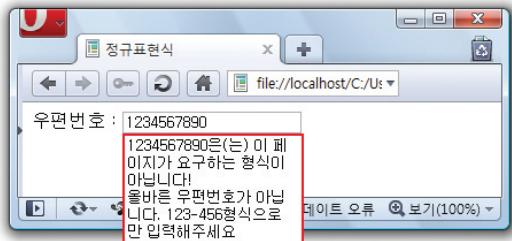
예제 2.105 우편번호 형식만 입력받도록 하는 정규표현식의 예

```
<input type="text" pattern="\d{3}(-\d{3})?" />
```

세 자리의 숫자와 하이픈, 그리고 다시 세 자리의 숫자를 입력받는 정규식입니다. 이 형식대로만 사용자에게서 값을 입력받고 다른 형식의 값은 입력받지 않습니다. 다음과 같이 코드를 만들어서 오페라로 확인해 보겠습니다.

예제 2.106 올바른 값이 아닌 경우 출력할 메시지

```
<input type="text" pattern="\d{3}(-\d{3})?" title="올바른 우편번호 형식이 아닙니다." />
```



☞ 그림 2.73 올바른 형식의 우편번호가 아닌 경우, 서밋을 하지 않습니다.

이 정규식은 우편번호를 받는 정규식으로 ‘123–456’ 형태의 값만 입력받도록 되어 있습니다. 이 형태를 벗어나는 값을 사용자가 입력하고 서밋을 하려고 하면 서밋이 되지 않고 오류메세지가 출력됩니다. 이것을 구현하는 데 HTML 단 한 줄이면 된다는 것이 놀랍습니다. 어떤 스크립트도 필요 없습니다. 발전한 HTML5의 디테일한 부분은 알면 알수록 놀랍습니다.

정규표현식은 처음 보면 이게 무슨 외계어인가 하는 느낌이 듭니다. 하지만 정규표현식에도 일정한 규칙이 있습니다.

☞ 표 2.14 정규표현식에서 사용하는 규칙의 몇 가지 예

기호와 문자	규칙
/ 와 /	/와 /를 감싸면 정규표현식을 사용한다는 의미입니다.
\	\는 바로 뒤에 오는 기호를 문자로 만듭니다. 가령 '*'은 꼽하기 연산자를 의미하지만 '*'로 사용되면 단지 문자로서 '*'을 의미합니다.
\d	0부터 9까지의 숫자. 단 계산은 되지 않고 문자로 인식합니다.
{n}	n개의 문자를 의미합니다. a{3}이라고 하면 aaa를 의미합니다. \d{3}은 숫자형 문자 3개를 의미하겠죠.
[0~9]	0부터 9까지의 숫자. 계산 가능한 숫자를 추출합니다.
	or와 같은 의미입니다.
\$	문자열의 끝부분을 의미합니다.
[abc]	문자열의 범위로 [abc]의 경우에는 a와 b와 c만을 의미합니다.
[a~z]	문자열의 범위로 ‘~’는 범위를 설정합니다. [a~z]는 a에서 z까지를 의미합니다.
?	0개 또는 1개를 의미. a?b는 b도 되고, ab도 됩니다.
-	~부터 ~까지, 즉 영어의 to와 같습니다.
^	문자열의 처음을 의미합니다.

이 표에 보인 규칙은 정규표현식의 규칙 중 일부에 불과합니다. 정규표현식만 따로 공부를 한다고 해도 분량이 꽤 됩니다.

자바스크립트 정규식은 어렵기는 하지만 천천히 익혀 두면 매우 편리하게 사용할 수 있습니다. 이 책에서 다루기에는 주제가 다소 어긋나므로 정규표현식에 대한 이야기는 이 정도로 정리하겠습니다. 정규표현식에 대해 호기심이 많은 독자들은 시중에 나와 있는 책으로 따로 공부해도 재미있을 것입니다.



브라우저별 새로운 품 요소와 속성 지원 현황 체크

이 장에서 소개한 예제들을 직접 타이핑해 보았나요? 아마 직접 코드를 입력해 가면서 공부했다면 느끼셨겠지만 브라우저들이 저마다 지원하는 요소와 속성이 아직은 천차만별입니다. 어떤 브라우저에서는 placeholder 속성이 잘 동작하는 반면에 어떤 브라우저에서는 placeholder 속성이 잘 작동하지 않는 것을 확인할 수 있습니다. 또 어떤 브라우저에서는 range 요소가 잘 나오는 반면 어떤 브라우저에서는 range 요소가 잘 표현되지 않을 것입니다. 이 문제는 브라우저에 따른 문제도 있지만 같은 브라우저라고 해도 버전별로 많은 차이를 보입니다. 그런 이유로 아직 HTML5의 품 요소를 실제 서비스에 적용하기에는 많은 무리가 따릅니다. 그렇지만 머지않은 미래를 대비해 공부하는 것입니다. 여러분들이 HTML5의 새로운 품 요소와 새로운 속성들을 공부할 때 도움이 될 사이트 하나를 소개하겠습니다.

The screenshot shows a browser window with the title "HTML5 inputs and attribute support". The address bar shows the URL "http://miketaylr.com/code/input-type-attr.html". The page content includes a red header box with the text "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.5 (KHTML, like Gecko) Chrome/4.1.249.1064 Safari/532.5". Below the header, there is a note: "A red input (followed by frowny faces) indicates the browser does not support the input type." A table follows, comparing various input types (text, search, url, tel) against attributes: type, autocomplete, autofocus, list, maxlength, and pattern. The table uses red and green colors to highlight support or lack thereof.

<input>	type	autocomplete	autofocus	list	maxlength	pattern
	text	false	true	false	true	true
	search	false	true	false	true	true
	url	false	true	false	true	true
	tel	false	true	false	true	true

☞ <http://miketaylr.com/code/input-type-attr.html>

이 사이트에 접속하면 사용하는 브라우저가 어떤 품 요소와 인풋 속성을 지원하는지 한눈에 볼 수 있습니다. HTML5 품을 공부할 때 유용할 것입니다.

HTML5의 인풋 속성 중에서 pattern 속성과 정규표현식을 이용하면 외부 스크립트를 이용하지 않고도 사용자로부터 원하는 형식의 값만 입력받을 수 있습니다.

추가된 요소

공개키와 비밀키 생성을 위한 keygen 요소

사용자들 간에 통신을 하거나 금전을 거래하기 위해 공개키와 비밀키 한 쌍을 생성하여 공개키는 폼으로 전송하고 비밀키는 클라이언트측에 저장해 주는 요소입니다. 공개키는 보안을 위하여 우리가 통신하는 데이터를 암호화한 키라고 생각하면 됩니다. 이 공개키는 비밀키를 아는 사람만이 복호화하여 원래 데이터를 해석할 수 있도록 합니다.

예제 2.107 공개키를 만들어 전송하도록 마크업한 예

```
<form action="kg.jsp">
  <keygen name="kgmoney" challenge>
  <input type="submit" value="키전송" />
</form>
```

이 예제와 같이 마크업을 하고 브라우저로 확인해 보면 다음 그림과 같은 콤보박스를 확인할 수 있습니다.

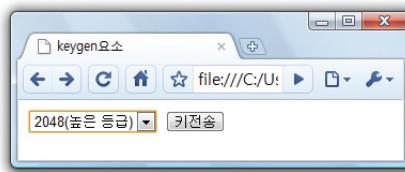


그림 2.74 <keygen> 요소를 브라우저로 확인해 본 모습

책을 쓰고 있는 현재까지 크롬에서는 높은 등급과 중간 등급 2개의 공개기 전송만을 지원하고 있습니다. 당연히 1024비트보다 높은 등급의 2048비트의 공개기가 암호화 수준이 높습니다. HTML5의 웹폼을 가장 잘 지원하고 있는 오페라의 경우에는 이보다 훨씬 다양한 레벨의 공개기 암호화 레벨을 지원합니다.

데이터를 출력하여 보여 주는 output 요소

<output> 요소는 스크립트 등을 이용해서 폼 내에 있는 다양한 값을 출력할 수 있습니다. 기존에 화면에는 출력하지 않지만 데이터를 임의로 가지고 있던 <hidden> 요소와는 달리 <output> 요소는 가지고 있는 값을 적극적으로 화면에 출력해 주고 서밋을 하면 가지고 있는 값을 넘겨줄 수도 있습니다.

간단한 예제를 하나 만들어 보겠습니다.

예제 2.108 range 속성과 output 요소를 결합하여 사용한 예

```
<form>
  <input type="range" required name="range" min="0" max="100" value="0" />
  <output onforminput="value=range.value">0</output>
</form>
```

폼 안에 있는 <input> 요소 중 값이 변하는 요소가 있으면 onforminput 이벤트가 발생합니다. <output> 요소에 해당 이벤트가 들어가 있습니다. onforminput 이벤트가 발생하면 input range로 만들어져 있는 슬라이더의 값을 <output> 태그에 출력하도록 만든 소스입니다. 결과 화면은 다음 그림과 같습니다. 오페라10 이상의 버전에서 테스트해 보기 바랍니다.



☞ 그림 2.75 슬라이더와 <output> 요소를 결합하여 만든 예제의 결과 화면 (오페라 10)

06

액티브엑스가 필요 없는 내장 미디어

이름에서 알 수 있듯이 오디오 요소와 비디오 요소는 사운드의 재생과 동영상의 재생을 위한 요소입니다. 두 요소 모두 HTML4.01 이전에는 없었고 HTML5에 새로 추가된 요소입니다. 이전 버전의 HTML 문서에서도 사운드 파일과 비디오 파일을 재생할 수 있었습니다. 그러기 위해서 이전에는 embed 요소와 object 요소를 사용했습니다.

그것은 표면적인 것에 불과하고 사실 큰 문제는 따로 있었습니다. HTML 문서도 우리가 읽는 책과 같이 온전한 하나의 문서임에는 틀림없습니다. 그리고 전자적으로 출력할 수 있는 전자문서인 만큼 문자 이외에 동영상이나 사운드와 같은 요소들도 문서에 포함되어야 합니다. 그러나 지금껏 이들은 HTML 문서의 하나로 존재하지 못했습니다. 플래시나 실버라이트, 미디어플레이어나 쿼타임 플레이어와 같은 외부 플러그인 안에 갇혀 있었습니다. 많은 개발자와 디자이너, 퍼블리셔가 웹표준을 지향하고 있고 모두가 공유할 수 있는 웹을 만들고자 하는 시대에 이것은 장애와도 같은 것이었습니다.

하지만 우리는 동영상이나 음악과 같은 멀티미디어 요소들을 공유할 수 있는 시대를 유튜브나 스티큐브와 같은 서비스로 체험해 볼 수 있었습니다. 더 나아가 이제는 오디오 요소와 비디오 요소를 사용하면 외부의 플러그인을 사용하지 않고 내장된 플레이어로 음악이나 동영상을 감상할 수 있습니다. 멀티미디어 요소가 웹페이지의 일부분이 되는 것입니다. 이렇게 되면 웹의 기본 정신에도 부합되는 것이고 시맨틱한 웹페이지를 만드는 데 있어서 동영상도 그 영역 안에 포함할 수 있게 됩니다. 또한 검색엔진이 동영상을 크롤링하여 찾을 수 있게 해 줍니다. 이미지 태그에 alt를 달아 설명을 달 수 있듯이, 동영상에도 텍스트 컨텐츠를 부여하여 누구나 공유하고 이용할 수 있도록 도와줄 것입니다.

사운드 재생을 위한 오디오 요소

▶ 표 2.15 브라우저별 오디오 요소 지원현황

 익스플로러	 파이어폭스	 사파리	 크롬	 오페라
9.0 버전부터 지원	3.5 버전부터 지원	3.2 버전부터 지원	3.0 버전부터 지원	10.5 버전부터 지원

오디오 요소의 사용 방법은 매우 간단합니다.

예제 2.109 가장 기본적인 오디오 요소 사용 예

```
<audio src= "sound.wav"></audio>
```

가장 간단한 사용 방법입니다. 이렇게 하면 사운드를 재생하기 위한 플레이어가 올라와 있는 상태가 됩니다. 하지만 아무런 소리도 들리지 않을 것입니다. `autoplay` 속성을 추가해 주면 플레이어는 보이지 않지만 페이지가 로드되자마자 사운드가 재생될 것입니다. 여러분도 가지고 있는 ogg 파일이나 wav 파일을 링크하여 시험해 보세요.

예제 2.110 페이지가 열리면 자동으로 사운드를 재생

```
<audio src= "sound.wav" autoplay="autoplay"></audio>
```

자, 자동으로 사운드가 재생되나요? 그런데 만약 사용자가 HTML5의 오디오 요소를 지원하지 않는 구형 브라우저를 사용한다면 어떻게 해야 할까요? 그런 경우에는 열리는 오디오 태그와 닫히는 오디오 태그 사이에 다음과 같이 안내문구를 넣어 주면 됩니다.

예제 2.111 오디오 태그를 지원하지 않는 브라우저의 경우 에러 문구 출력

```
<audio src= "sound.wav" autoplay="autoplay">
    죄송합니다. 이 브라우저는 HTML5의 오디오 태그를 지원하지 않습니다.
</audio>
```

간단하죠? 이렇게 하면 오디오 태그가 지원되는 브라우저에서는 소리가 재생되고 문구는 보이지 않습니다. 그러나 오디오 요소가 지원되지 않는 브라우저에서는

소리가 재생되지 않지만 안내문구가 보이게 됩니다.

여기서 생각해 볼 문제가 하나 있습니다. 우리는 웹표준과 웹접근성의 중요성에 대해 알고 있습니다. 모두가 평등하게 정보를 소비해야 한다고 믿고 있습니다. 그런데 구형 브라우저를 사용하고 있다고 해서 단지 안내문구만 띄워 주면 안 될 것입니다. 궁극적이고 깔끔한 방법이라고 할 수는 없지만 구형 브라우저를 사용하는 사람들을 위해서 다음과 같이 처리해 주면 됩니다.

예제 2.112 오디오 태그를 지원하지 않는 브라우저는 embed 태그로 대체 사운드 재생

```
<audio src= "sound.wav" autoplay="autoplay">  
  <embed src="sound.wav"></embed>  
</audio>
```

이렇게 해 두면, 오디오 태그를 지원하는 브라우저에서는 오디오 태그를 통해서 사운드를 재생하고, 오디오 태그를 지원하지 않는 브라우저는 embed 태그를 통해서 사운드를 재생할 것입니다.

자, 이제 기본적인 오디오 요소의 사용 방법에 대해서는 감을 잡았습니다. 그런데 문득 플레이어가 어떻게 생겼는지 몹시 궁금해집니다. 방법은 간단하지요. 확장 태그 하나만 붙이면 플레이어 컨트롤 패널을 표시할 수 있습니다.

예제 2.113 컨트롤 패널 보기

```
<audio src= "sound.wav" autoplay="autoplay" controls="controls">  
  <embed src="sound.wav"></embed>  
</audio>
```



☞ 그림 2.76 오디오 태그를 이용하여 플레이어를 크롬에서 구동한 모습

`controls` 속성만 추가하면 플레이어의 모습을 확인할 수 있습니다. 이 그림은 크롬에서 확인한 내장 플레이어의 모습입니다. 재생, 프로그레스 막대, 재생시간 표시, 볼륨 조절 버튼 등 기본적으로 필요한 것은 다 갖춘 플레이어입니다. 말이 나온 김에 HTML5의 오디오 요소에서 사용할 수 있는 속성을 정리해 보겠습니다.

☞ 표 2.16 글로벌 속성을 제외한 오디오 태그에서 지원하는 속성

속성명	값	기능 및 역할
src	파일 경로	재생할 사운드 파일의 경로를 지정
autoplay	빈 값, autoplay	페이지가 로드되자마자 사운드를 재생할지 지정
controls	빈 값, controls	플레이어를 표시
loop	재생횟수(숫자)	사운드를 반복 재생할 횟수를 지정. 기본값은 1임
preload	none, auto, meta	페이지가 열리면 비디오를 미리 로드해 둠 none – 프리로드를 사용하지 않음 auto – 페이지가 모두 열리면 비디오 프리로드 meta – 페이지가 모두 열리면 비디오 메타정보만 로드

오디오 코덱 정리

오디오 요소도 비디오 요소와 마찬가지로 코덱이 문제인데 브라우저마다 지원하는 파일 포맷이 약간씩 다릅니다. 다음 표를 참고해 주세요. 책을 쓰고 있는 시점 기준입니다.

☞ 표 2.17 브라우저별 지원하는 사운드 포맷, HTML5 `<audio>` 태그

	크롬	파이어폭스	사파리	익스플로러	오페라
Vorbis (oga, ogg)	지원함	지원함	지원 안함	지원 안함	지원함
(wav, wma)	지원 안함	지원함	지원함	지원함	지원함
(mp3)	지원함	지원 안함	지원함	지원함	지원 안함
AAC	지원함	지원 안함	지원함	지원함	지원 안함

비디오 재생을 위한 비디오 요소

☞ 표 2.18 브라우저별 비디오 요소 지원현황

 익스플로러	 파이어폭스	 사파리	 크롬	 오페라
9.0 버전부터 지원	3.5 버전부터 지원	3.2 버전부터 지원	4.0 버전부터 지원	10.5 버전부터 지원

비디오 요소의 사용 방법도 오디오 요소의 사용 방법과 크게 다르지 않습니다. 동일한 플레이어니 영상이 나오고 안 나오고의 차이일 뿐 거의 같다고 봐도 무방합니다.

예제 2.114 기본적인 비디오 태그 사용법과 지원하지 않는 브라우저를 위한 안내문구

```
<video src= "movie.ogg" controls="controls" width="400" height="350">
  사용하시는 브라우저에서는 비디오를 볼 수 없습니다.
</video>
```

비디오 태그를 이용하여 비디오를 재생하기 위한 코드입니다. 간단하게 훑어봐도 오디오 태그와 사용형식이 똑같은 것을 알 수 있습니다. 비디오 요소에서 지원하는 속성들을 바로 알아보겠습니다.

☞ 표 2.19 글로벌 속성을 제외한 비디오 태그에서 지원하는 속성

속성명	값	기능 및 역할
src	파일 경로	재생할 비디오의 경로
autoplay	빈 값, autoplay	페이지가 열리면 비디오를 자동으로 재생
controls	빈 값, controls	비디오를 제어할 수 있는 컨트롤 패널 사용
preload	none, auto, meta	페이지가 열리면 비디오를 미리 로드해 둠 none – 프리로드를 사용하지 않음 auto – 페이지가 모두 열리면 비디오 프리로드 meta – 페이지가 모두 열리면 비디오 메타정보만 로드
poster	이미지 URL	비디오가 로드 되기전에 보여 줄 이미지 설정
loop	반복횟수, 숫자	비디오를 몇 번 반복 재생할 것인지 설정
width	픽셀단위 숫자	비디오 스테이지의 너비
height	픽셀단위 숫자	비디오 스테이지의 높이



☞ 그림 2.77 H.264코덱을 쓰는 mp4파일을 크롬에서 구동하는 모습(왼쪽), ogg theora코덱을 쓰는 ogv 비디오를 파이어폭스에서 구동한 모습(오른쪽). 브라우저에서 지원하지 않는 코덱을 사용하는 동영상을 보려면 플레이어 자체가 노출이 안 됩니다.

비디오 코덱 정리

아직까지는 코덱이 통일되지 않아서 비디오 태그에서 사용할 수 있는 비디오 타입이 브라우저마다 조금씩 다릅니다. 다음 표를 참고해 주세요. 책을 쓰고 있는 시점 기준입니다.

☞ 표 2.20 브라우저별 지원 비디오 코덱과 동영상 포맷, HTML5 <video> 태그

	크롬	파이어폭스	사파리	익스플로러	오페라
Theora (ogv, ogg)	지원함	지원함	지원 안함	지원 안함	지원함
H.264 (mp4 등)	지원함	지원 안함	지원함	지원함	지원 안함
VP8 (Web M)	지원함	지원함	미정	지원함	지원함



HTML5 비디오와 관련된 여러 이야기들(표준 포맷 문제 그리고 플래시…)

canvas와 더불어서 비디오 요소는 플래시와 HTML5 간의 비교를 불러일으킨 핵심 요소입니다. 여기에는 긴 사연이 있습니다.

동영상 재생 코덱 싸움의 핵심에 H.264 코덱과 Ogg theora 코덱이 있습니다. H.264 코덱은 사파리와 크롬에서 기본으로 지원하지만 파이어폭스에서는 특허료 지불 등의 문제로 지원하지 않고

있습니다. 반면, 오픈소스이며 라이선스 비용이 들어가지 않는 Ogg theora 코덱은 파이어폭스와, 오페라, 크롬에서 지원합니다. 크롬은 양 진영 모두를 지원하며 구글은 최근 On2테크놀로지사를 인수하였습니다. 구글은 이 회사를 인수하여 역시 오픈소스인 V8 기반의 '웹M'이라고 하는 새로운 동영상 포맷을 개발하였고 크롬에서 이를 지원한다고 하였습니다. 가장 느긋한 쪽은 구글입니다. 이미 막대한 특허료를 내면서도 H.264 코덱을 지원하고 Ogg theora의 ogv 포맷도 지원합니다. 게다가 독자적인 포맷까지 만들어 냈으나 구글이 선택할 수 있는 카드가 가장 많아 보입니다. 웹M 프로젝트에는 파이어폭스를 만드는 모질라와 오페라까지 가세하였습니다. 비싼 특허료를 내기며 H.264 코덱 사용을 고집하는 애플과 MS는 고심에 빠졌습니다. 구글은 동영상도 웹페이지의 일부가 되어야 한다고 주장하며 개방형 비디오를 지향한다고 말하고 있습니다. 웹M 포맷을 자사의 유튜브와 구글 TV 등 전방위적으로 사용할 것입니다. 이미 구글은 유튜브를 비롯하여 자사의 굵직한 서비스를 HTML5로 컨버팅하여 서비스하고 있습니다.

자, 그럼 플래시와의 비교 문제는 왜 불거졌을까요? 아마도 canvas와 플래시가 비교되는 이유와 비슷할 것입니다. 동영상 플레이를 위해 플래시의 힘을 빌리지 않아도 되었다는 것이 가장 큰 이유인 것 같습니다. 유튜브의 HTML5 지원이 기쁨을 부었고요. 아마도 외부 플러그인 없이도 자유롭게 동영상을 볼 수 있게 된다면 사용자 입장에서도 훨씬 편리하고 좋은 환경이 되지 않을까 생각합니다.

스크립트를 이용한 내장 미디어 컨트롤

자바스크립트를 이용해서 <audio>나 <video>와 같은 내장 미디어 요소들의 컨트롤을 간단하게 해 보겠습니다. 오디오와 비디오를 컨트롤하는 방법은 동일합니다. 먼저 외부에서 비디오를 재생시키고 멈추게 하는 스크립트를 만들어 보겠습니다.

예제 2.115 기본적인 play 이벤트와 pause 이벤트를 이용한 외부 컨트롤의 예

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>비디오 컨트롤</title>
<script>
    // 비디오 재생을 위한 play 함수
    var play = function() {
        var videoTag = document.getElementsByTagName('video')[0];      // 비디오 태그 가져옴
        videoTag.play();           // videoTag 변수에 플레이 이벤트를 부여
    }
    // 비디오 멈춤을 위한 pause 함수
    var pause = function() {
        var videoTag = document.getElementsByTagName('video')[0];
        videoTag.pause();
    }
</script>
```

```
};

// 비디오 일시 정지를 위한 pause 함수
var pause = function() {
    var videoTag = document.getElementsByTagName('video')[0];      // 비디오 태그 가져옴
    videoTag.pause();        // videoTag 변수에 일시정지 이벤트를 부여
};

</script>
</head>
<body>
    <video src="sample.ogv"></video>
    <input type="button" value="비디오 재생" onclick="play();" /> <!-- play() 함수 호출 -->
    <input type="button" value="일시정지" onclick="pause();" /> <!-- pause() 함수 호출 -->
</body>
</html>
```

샘플 비디오 파일을 비디오 태그로 읽어들였습니다. 외부의 버튼으로 비디오를 컨트롤 하기 위해서 일부러 비디오 컨트롤 패널은 사용하지 않았습니다. 비디오 태그 아래에 임의의 버튼 2개를 만들었습니다. 하나는 재생 버튼이고 하나는 일시 정지 버튼입니다. 재생 버튼을 클릭하면 play() 함수를 호출하고 일시정지 버튼을 클릭하면 pause() 함수를 호출합니다.



☞ 그림 2.78 비디오 하단에 추가된 2개의 버튼을 이용해서 비디오를 컨트롤해 보고 있습니다.

`document.getElementsByTagName('video')`를 이용해서 HTML 마크업상에 있는 비디오 태그를 스크립트 안으로 가져옵니다. 이렇게 가져온 비디오 태그는 ‘videoTag’라는 임의의 변수에 담고 재생 버튼이나 일시정지 버튼이 클릭될 경우 각 함수는 비디오 태그에 내장된 이벤트를 실행합니다.

HTML5의 내장 미디어는 `play()`, `pause()` 이외에도 다양한 이벤트를 제공합니다.

☞ 표 2.21 HTML5 내장 미디어에서 사용할 수 있는 이벤트 핸들러

이벤트 이름	이벤트 설명
loadstart	미디어의 로딩이 시작될 때 발생하는 이벤트입니다.
progress	미디어 다운로드 진행 상황의 중요한 부분을 주기적으로 알려주기 위해 발생하는 이벤트로 세 가지 속성을 가집니다. • lengthComputable : 미디어의 전체 사이즈를 알면 ‘true’이고 아닐 경우는 ‘false’ • loaded : 지금까지 받은 미디어 크기의 바이트 단위 숫자 • total : 미디어 파일 전체 크기의 바이트 단위 숫자
suspend	미디어의 로딩이 중단되었을 때 발생하는 이벤트입니다. 미디어의 로딩이 완료되었거나 인터넷이 끊기는 등의 여러 가지 이유로 로딩이 중단되었을 때 발생합니다.
abort	재생이 중단되었을 때 발생하는 이벤트, 동영상이 재생되고 있다가 처음부터 다시 시작될 때 발생하는 이벤트입니다.
error	에러가 발생한 경우에 발생하는 이벤트입니다.
emptied	미디어가 이미 다 로드되었거나 혹은 부분적으로 로드되었을 때 발생하는 이벤트입니다. 그리고 <code>load()</code> 메소드가 미디어를 다시 로드하라고 호출했을 때 발생합니다. 이는 미디어가 더 이상 로드할 것이 없어서 비어 있을 때를 의미합니다.(미디어가 완전히 없어서 빈 것과는 다릅니다.)
play	미디어를 새로 재생하거나 일시정지되어 있는 상태에서 연결하여 재생하는 이벤트입니다.
pause	재생 중인 미디어를 일시정지시키는 이벤트입니다.
loadedmetadata	미디어의 메타데이터가 모두 로드되었기 때문에 모든 속성은 앞으로 사용할 수 있는 유용한 정보를 담게 됩니다. 미디어의 메타데이터가 로드되었을 때 발생합니다.
loadeddata	미디어의 첫 번째 프레임의 로딩이 끝났을 때 발생합니다.
waiting	이미 요청된 작업(예를 들어 재생)을 잠시 보류하고 다른 작업(예를 들어 재생 위치 변경)을 먼저 끝낼 때 발생하는 이벤트입니다.
canplay	미디어가 재생되기에 충분한 데이터가 로드되었을 때 발생하는 이벤트입니다. 적어도 몇 개 정도의 프레임이 로드되어야 발생합니다.

◆ 표 2.21 계속

이벤트 이름	이벤트 설명
canplaythrough	미디어를 재생하기 위해 준비되어 있는 현재 상황이 CAN_PLAY_THROUGH로 변했을 때 발생합니다. 즉, 끊김없이 전체 미디어가 플레이할 수 있는 상태를 의미합니다. 적어도 현재 보고 있는 미디어의 위치와 다운로드되는 미디어의 속도가 동일해야 합니다.
seeking	미디어 탐색 작업을 시작할 때 발생합니다.
seeked	미디어 탐색 작업이 완료되었을 때 발생합니다.
timeupdate	비디오 요소의 currentTime 속성이 변하면서 시간이 지정될 때 발생하는 이벤트입니다.
ended	재생이 완료되었을 때 발생하는 이벤트입니다.
empty	미디어가 비어 있어서 에러가 생기면 발생하는 이벤트입니다.
ratechange	재생속도가 변할 때 발생하는 이벤트입니다.
durationchange	메타데이터가 로드되었거나 바뀌었을 때 미디어의 길이가 변하면 발생하는 이벤트입니다. 예를 들면, 미디어가 충분히 로드되어서 미디어의 길이를 알 수 있을 때를 의미합니다.
volumechange	음소거 속성이 변하거나 볼륨이 바뀔 때 발생하는 이벤트입니다.

07

구형 익스플로러를 위한 대비책

해외에서는 많은 사이트들이 HTML5 문서 형식으로 웹페이지를 만들고 있습니다. 우리나라에도 조금씩 변화의 물결이 다가오고 있습니다. 그러나 아직은 대대적으로 HTML5를 사용하기에 무리가 있습니다. 최신 브라우저들마저도 HTML5의 모든 스펙을 지원하지는 않기 때문입니다. 아직은 브라우저별로 지원하는 속성이나 마크업 요소도 천차만별입니다. 현황은 이렇습니다.

하지만 몇몇 뛰어난 개발자들 덕분에 HTML5의 일부 요소들을 모든 브라우저에서 사용할 수 있는 방법이 있기는 합니다. 특히 익스플로9부터 HTML5를 전면적으로 지원하는 익스플로러의 경우는 8 이전 버전에서도 HTML5의 일부 태그를 인식할 수 있도록 하는 방법이 있습니다.

익스플로러에서 아웃라인 태그 인식시키기

익스플로러에서는 새로이 추가된 header, footer와 같은 아웃라인 태그의 DOM을 제대로 인식하지 못하는 문제가 있습니다. 이를 해결하기 위해서 다음과 같은 자바스크립트를 사용합니다.

예제 2.116 익스플로러를 위한 아웃라인 태그 객체 생성

```
<script type="text/javascript">
    document.createElement('header');
    document.createElement('nav');
    document.createElement('article');
    document.createElement('section');
    document.createElement('aside');
    document.createElement('footer');
</script>
```

이렇게 만든 자바스크립트를 HTML5 문서의 <head> 안에 포함시키면 익스플로러에서도 위의 태그들을 인식할 수 있습니다. 제대로 동작하는지 테스트부터 해 보겠습니다.

일단 테스트를 위해서 다음과 같이 HTML5 마크업을 하나 만들어 보겠습니다.

예제 2.117 아웃라인 태그는 익스플로러에서는 기본적으로 지원되지 않습니다

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset=“UTF-8”>
    <title>익스플로러를 위해!!</title>
    <style type="text/css">
        header {background-color:red;}
        article {background-color:green;}
        footer {background-color:blue;}
    </style>
</head>
<body>
    <header>헤더입니다</header>
    <article>아티클입니다</article>
    <footer>푸터입니다</footer>
</body>
</html>
```

이렇게 HTML 마크업을 하고 나서 익스플로러와 크롬에서 확인해 보았습니다.



☞ 그림 2.79 왼쪽이 크롬, 오른쪽이 익스플로러로 확인한 모습입니다. 크롬에서는 스타일이 적용된 반면에 익스플로러에서는 스타일을 적용하지 못하는 모습입니다.

이 그림에서 볼 수 있듯이 크롬에서는 `<header>`, `<footer>`와 같이 HTML5에 새로 추가된 아웃라인 태그들을 잘 인식합니다. 그래서 스타일 시트에서 각 부분마다 정해 준 색상도 잘 표현됩니다. 반면 익스플로러는 스타일시트를 적용하지 못하고 있는데, 익스플로러가 새로이 추가된 이 태그들을 인식하지 못해서 그렇습니다. 익스플로러에서도 이 태그들을 인식할 수 있도록 앞서 소개한 자바스크립트를 추가해 보겠습니다.

예제 2.118 익스플로러를 위한 아웃라인 객체 생성 스크립트 삽입

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset= "UTF-8">
    <title>익스플로러를 위해!!</title>
    <style type="text/css">
        header {background-color:red;}
        article {background-color:green;}
        footer {background-color:blue;}
    </style>
    <script type="text/javascript">
        document.createElement('header');
        document.createElement('nav');
        document.createElement('article');
        document.createElement('section');
```

```
document.createElement('aside');
document.createElement('footer');
</script>
</head>
<body>
<header>헤더입니다</header>
<article>아티클입니다</article>
<footer>푸터입니다</footer>
</body>
</html>
```

이렇게 익스플로러에서도 `<header>`, `<footer>`, `<article>`과 같은 태그를 인식할 수 있도록 스크립트를 추가하였습니다. 다시 브라우저로 확인해 보았습니다.



☞ 그림 2.80 원쪽이 크롬, 오른쪽이 익스플로러입니다. 스크립트를 추가하니 익스플로러에서도 새로운 태그들을 인식할 수 있게 되었습니다.

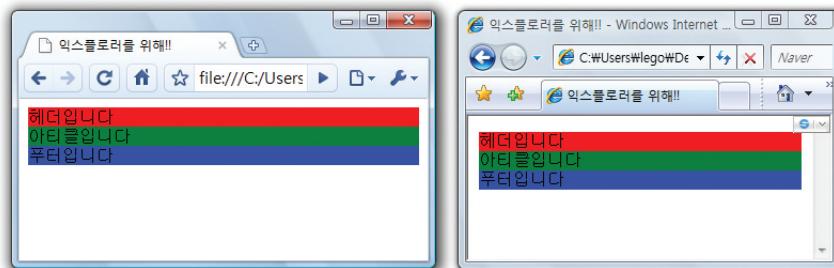
자, 이제 익스플로러에서도 새로운 태그들을 인식할 수 있게 되었습니다. 그런데, 문제가 하나 있는 듯 보입니다. `<header>`, `<article>`, `<footer>`와 같은 태그들은 블록레벨 태그입니다. 그런데 익스플로러에서는 인라인 레벨 태그로 인식을 합니다. 익스플로러에서도 이들이 블록레벨 태그 속성을 가질 수 있도록 하는 조치가 필요할 것 같습니다. 다음과 같이 CSS를 추가해 보겠습니다.

예제 2.119 생성한 아웃라인 객체를 블록레벨화함

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

```
<meta charset= "UTF-8">
<title>익스플로러를 위해!!</title>
<style type="text/css">
    header, nav, article, section, aside, footer, menu, hgroup, figure {display:block;}
    header {background-color:red;}
    article {background-color:green;}
    footer {background-color:blue;}
</style>
<script type="text/javascript">
    document.createElement('header');
    document.createElement('nav');
    document.createElement('article');
    document.createElement('section');
    document.createElement('aside');
    document.createElement('footer');
</script>
</head>
<body>
    <header>헤더입니다</header>
    <article>아티클입니다</article>
    <footer>푸터입니다</footer>
</body>
</html>
```

새로운 태그들을 익스플로러가 블록레벨 태그로 인지할 수 있도록 하기 위해서 이와 같이 CSS를 한 줄 추가하였습니다. 그리고 다시 브라우저로 확인해 보겠습니다.



☞ 그림 2.81 왼쪽이 크롬, 오른쪽이 익스플로러입니다. 이제 블록레벨 속성까지 적용이 완료되었습니다.

자, 어떤가요? 블록레벨 속성을 적용하고 나니 이제는 익스플로러에서도 새로운 태그들을 확실히 인지하고 표현하기 시작하는 모습을 볼 수 있습니다.

이들 스크립트와 초기화 CSS를 매번 페이지마다 사용할 수는 없는 노릇입니다. 따로 .js 파일과 .css 파일을 만들어서 <head> 태그 안에서 불러서 사용하면 편리하겠습니다.

아, 잠시만요! 익스플로러에서 HTML5의 몇 가지 태그들을 인식시키기 위해서 더욱 세련된 스크립트를 사용할 수 있습니다. 유명한 UI 개발자인 레미샤프 씨가 제작하여 구글 코드를 통해서 배포하고 있는 자바스크립트가 있습니다.

예제 2.120 HTML5의 몇 가지 추가적인 태그를 익스플로러에서 인식시키기 위한 라이브러리 링크

```
<!--[if lt IE 9]>
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

HTML 문서의 <head> 안에 위와 같이 컨디셔널 코멘트를 넣어서 사용하면 됩니다. 'if lt IE 9'은 IE 9 이하 버전의 IE에 이 스크립트를 적용한다는 뜻입니다. 이 html5shiv 자바스크립트에서 익스플로러를 위해 지원하는 HTML5 태그는 abbr, article, aside, audio, canvas, details, figcaption, figure, footer, header, hgroup, mark, meter, nav, output, progress, section, summary, time, video입니다.

예제 2.121 익스플로러6, 7 버전의 웹표준 향상을 위한 스크립트

```
<!--[if lt IE 7]>
<script src="http://ie7-js.googlecode.com/svn/version/2.1(beta4)/IE8.js"></script>
<![endif]-->
```

익스플로러7 이하의 브라우저를 위해서 IE8.js 파일을 로드합니다. 이 스크립트는 안 써도 크게 상관은 없지만 익스플로러6에서 투명처리가 되지 않는 문제 등 웹표준 향상을 위한 편리한 기능을 제공합니다.

익스플로러에서 <canvas> 사용하기

익스플로러9에서는 <canvas> 가 완벽하게 지원되겠지만 구형 익스플로러는 이를 지원하지 못합니다. 하지만 부족하나마 이것 역시 해결할 방법이 있습니다.

<http://code.google.com/p/explorercanvas/>에 접속하여 excanvas.zip 파일을 다운로드합니다. 다운로드한 파일의 압축을 풀면 excanvas.js라는 파일이 있을 것입니다. 이 파일을 HTML 문서 <head> 안에 다음과 같이 불러옵니다. 이때 컨디셔널 코멘트를 이용해서 익스플로러에서만 해당 자바스크립트 파일이 로드되도록 합니다.

예제 2.122 익스플로러에서 캔버스를 사용하기 위한 excanvas.js 라이브러리 링크

```
<head>
<!--[if IE]><script type="text/javascript" src="excanvas.js"></script><![endif]-->
</head>
```

<head> 안에 위와 같이 익스플로러용 컨디셔널 코멘트를 넣어 excanvas.js 파일을 로드하였습니다. 그리고 캔버스에 파란색 박스를 그리고 익스플로러에서 잘 작동하는지 확인해 보았습니다.

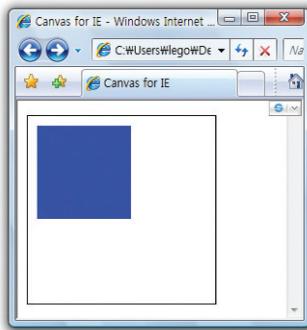


그림 2.82 익스플로러에서 성공적으로 캔버스를 작동시킨 모습

이처럼 간단한 방법으로 <canvas>를 익스플로러에서 사용할 수 있습니다.

캔버스에서 작성한 텍스트를 익스플로러에서 사용하기 위해서는 스크립트 라이브러리를 하나 더 추가해야 합니다.

<http://code.google.com/p/canvas-text/>에 접속해서 canvas.text.js 파일을 다운로드합니다.

HTML페이지의 <head> 안에 다음과 같이 다운로드한 .js 파일을 호출합니다.

예제 2.123 익스플로러에서 캔버스 텍스트 API를 구현하기 위한 라이브러리 로드

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

이렇게 하면 이제 <canvas>에서 구현한 텍스트까지도 익스플로러에서 잘 지원하게 됩니다.

<http://canvas-text.googlecode.com/svn/trunk/examples/index.html>

이 링크는 <canvas>에 구현한 텍스트와 애니메이션 등 몇 가지 샘플을 소개하는 페이지입니다. 이 페이지를 익스플로러에서 열어 보세요. 앞서 소개한 자바스크립트 라이브러리를 이용해서 만들어져 있고 익스플로러에서도 <canvas>의 여러 가지 기능이 잘 구현되는 것을 볼 수 있습니다.

익스플로러에 CSS3 인기 속성 적용하기

익스플로러9에서는 CSS3의 속성 대부분을 지원합니다. 이 방법은 익스플로러6, 7, 8버전을 위한 것입니다. 그리고 CSS3의 모든 기능을 적용하지는 못하고 CSS3의 새로운 속성 중 가장 인기 있는 3가지 속성만을 사용할 수 있습니다. 겉보기는 별반 차이가 없지만 실제로는 CSS3를 익스플로러에 적용하는 것이 아니라 VML을 이용해서 CSS3로 처리한 것 같은 효과를 준 것뿐입니다. 사용 방법을 알아보겠습니다.

<http://fetchak.com/ie-css3/ie-css3.htc> 경로에 위치해 있는 ie-css3.htc 파일을 다운로드합니다. 그리고 다음과 같이 CSS를 작성합니다.

예제 2.124 CSS3의 인기 속성을 구형 익스플로러에서 사용하기 위한 핵 파일 호출

```
.box {
    -moz-border-radius: 15px; /* for 파이어폭스 */
    -webkit-border-radius: 15px; /* for 크롬, 사파리 */
    border-radius: 15px; /* for 오페라 10.5, 미래 브라우저들 */

    -moz-box-shadow: 10px 10px 20px #000; /* for 파이어폭스 */
    -webkit-box-shadow: 10px 10px 20px #000; /* for 크롬, 사파리 */
    box-shadow: 10px 10px 20px #000; /* for 오페라 10.5, 미래 브라우저들 */

    behavior: url(ie-css3.htc); /* for 구형 익스플로러를 위한 스크립트 핵 */
}
```

이와 같이 CSS3 코드를 사용하면 되고 구형 익스플로러를 위해서 강조되어 있는 가장 마지막 줄의 behavior를 이용해서 ie-css3.htc 파일을 읽어오는 부분만 추가해 주면 익스플로러에서도 CSS3의 대표적인 3가지 속성을 이용할 수 있습니다.

☞ 표 2.22 ie-css3.htc에서 지원하는 3가지 CSS3 속성과 제약사항

지원하는 CSS3 속성	지원	지원 안함
border-radius	4개 모서리 한번에 둥글기 설정 외곽선	각 모서리 둥글기 개별 설정
box-shadow	그림자 높도 그림자 위치	#000(검정)을 제외한 다른 색
text-shadow	그림자 높도 그림자 위치 그림자 색상	

CSS3에 대한 더 많은 부분은 4장에서 공부하도록 하겠습니다. 일단 ‘익스플로러에서 CSS3의 등근 모서리 박스와 그림자 기능 일부를 사용할 수 있구나’ 하는 정도만 기억하고 이 장을 넘어가도 좋습니다. CSS3에 대해서 공부를 하고 혹시라도 나중에 필요할 때 이 부분을 다시 펼쳐서 활용해 보기 바랍니다.



CSS3 프리픽스

CSS3의 속성은 아직 모든 브라우저에서 동일하게 작동하지 않습니다. 아직은 CSS3의 속성이 많이 불안한 상태입니다. 눈치가 빠른 분들은 방금 본 CSS3 코드에서 이상한 점을 발견했을 것입니다. 박스에 그림자를 적용하기 위해 box-shadow 속성을 그냥 사용하면 될 텐데 동일한 코드가 두 줄이 더 있습니다. 하나에는 `-moz-`라는 녀석이 붙어 있고, 다른 하나에는 `-webkit-`이라는 녀석이 붙어 있습니다.

이것들의 이름은 ‘프리픽스’입니다. 아직은 CSS3 속성이 온전히 지원되지 않기 때문에 각 브라우저에 맞는 프리픽스를 붙여 주어야 합니다. `-moz-`를 붙이면 파이어폭스를 위한 코드가 되고, `-webkit-`를 붙이면 웹킷 엔진을 사용하는 크롬과 사파리를 위한 코드가 됩니다. 참고로 오페라는 `-o-`입니다.

마지막으로, 아무것도 안 붙은 것은 나중에 프리픽스 없이도 CSS3를 모든 브라우저에서 쓸 수 있을 때를 대비해서 만들어 둔 코드입니다.

HTML5 API를 익스플로러에서 쓸 수 있을까?

익스플로러에서도 HTML5의 API나 더욱 다양한 CSS3 선택자를 사용할 수 있도록 다양한 라이브러리가 나오고 있습니다. 그러나 HTML5의 API들은 라이브러리들을 사용하는 데 다소 무리가 따릅니다. 구글에서 기어즈 프로젝트(<http://gears.google.com/>)를 중단한 상태이고, 익스플로러 사용자들에게 ‘우리 서비스를 이용하려면 구글 크롬 프레임(<http://code.google.com/chromeframe/>)을 설치하세요’라고 요구하는데 사실은 이 요구에도 많은 무리가 따릅니다. 이 부분은 익스플로러9이 나와 줘야 해결될 것 같고, 구형 익스플로러에서 HTML5의 API들을 온전하게 지원할 방법은 아직까지 없는 것으로 보입니다.

아마 이 책이 출판되고 난 이후에도 익스플로러에서 HTML5와 CSS3를 사용할 수 있도록 도와주는 라이브러리는 계속 개발되어 나올 것입니다. 그런 라이브러리 정보가 있다면 필자의 블로그에서 소개하도록 하겠습니다.