HTML5

HTML5

- HTML5
  - ✓ 웹의 접근성을 강화하기 위해 HTML4에서 진화한 마크업 언어로 모바일 기기와 클라우드 컴퓨팅과 네트워킹 시대의 웹 사이트와 애플리케이션 제작을 위한 새로운 개념
  - ✓ 단순하고 보편적인 접근성 강화
  - ✓ 디자인과 내용의 분리(디자인에 관련된 많은 부분을 CSS에서 처리하도록 권장)
  - ✓ 자바 , Activie X, 플래시와 같은 외부 플러그 인을 최소화 해서 구현
  - ✓ 마크 업 언어에 웹 애플리케이션을 만들 수 있는 API를 포함
  - ✓ 문서 구조의 의미나 문서 안에 삽입된 데이터의 의미 등을 명확하게 하기 위한 사양을 다수 포함
  - ✓ 사용자 우선 주의
  - ✓ HTML5 API: http://dev.w3.org/html5/html-author/

태그의 변화

- 새로운 태그
  - ✓ 문서 구조에 관련된 태그

    section, article, aside, header, hgroup, footer, nav
  - ✓ 외부 콘텐츠 삽입 관련 태그

    figure, audio, video, embed, canvas, source
  - ✓ 폼 관련 태그

    keygen, output, progress, meter
  - ✓ 텍스트 관련 태그

    mark, ruby/rt/rp, time, command, details, datalist

태그의 변화

❖ 변경된 태그

태그	이전 버전의 의미	HTML5에서의 의미
hr	가로 줄	주제 변경
menu	사용하지 않도록 권장	문서 메뉴 정보 제공
small	글자를 작게	작은 인쇄 정보
strong	글자 강조	중요한 정보

태그의 변화

❖ 추가된 속성

속성	적용 가능 태그	의미
media	a, area	링크할 미디어
hreflang	area	링크하는 대상의 언어를 지정
charset	meta	문자 인코딩
autofocus	input, select, textarea, button	포커스 설정
placeholder	input, textarea	보충 설명
required	input, textarea	필수 입력 필드 설정
min, max	input	숫자의 범위를 설정
multiple	input	값을 여러개 선택 가능
pattern	input	조건을 사용한 일반식 표시
step	input	값의 단계
novalidate	form	유효성을 보증할 수 없음
async	script	스크립트를 바로 실행
reversed	ol	목록을 역순으로 표기

태그의 변화

- 없어진 요소
  - ✓ applet → embed/object 요소를 이용
  - ✓ acronym → 약어나 이니셜을 표현할 때 사용하던 것으로 abbr을 이용
  - ✓ dir → 디렉토리 내용을 표시하는 요소로 ul을 이용
  - ✓ frame, frameset, noframes → iframe과 css를 이용하거나 서버 사이드 콘텐츠를 결합하는 것을 권장
  - ✓ isindex → form 과 텍스트 필드의 조합을 이용
  - ✓ listing, xmp → 있는 그대로 출력하는 것으로 pre와 code를 이용
  - ✓ noembed → object 요소를 이용
  - ✓ plaintext-> MIME 타입 text/plain을 이용
  - ✓ rb → ruby 요소에 기재
  - ✓ basefont, big, blink, center, font, marquee, s, space, strike, tt, u → CSS 대신
  - ✓ bgsound → audio 요소를 이용

태그의 변화

- 태그 변화
  - ✓ 종료 태그를 기술하면 안 되는 태그

    area, base, br, col, command, embed, hr, img, input, keygen, link, meta, param, source
  - ✓ 시작 태그와 종료 태그 모두 생략 가능한 태그

    html, head, body, colgroup, tbody
  - ✓ 종료 태그를 생략할 수 있는 태그

    li, dt, dd, p, rt, rp, optgroup, option, thead, tfoot, tr, td, th
  - ✓ 선택자 API 추가
    - querySelector(선택자): 선택자에 해당하는 하나의 개체 리턴
    - querySelectorAll(선택자): 선택자에 해당하는 모든 개체 리턴
  - ✓ window.JSON 지원

API 변화

❖ 추가된 API

API	내용
Web Storage	브라우저에 임시 저장 – 자동 소멸 가능
Web SQL	브라우저 안에서의 내장 데이터베이스 사용
Indexed DB	키-값 의 형태로 데이터베이스에 저장
Drag & Drop	웹 페이지 내에서 Darg&Drop 가능
Geolocation	디바이스의 위치 정보를 제공해주는 API
Canvas 2D와 3D	그리기 API
Web Worker	작업을 분할해서 수행 가능
MathML	수식을 html에 나타내는 방법
마이크로데이터	시맨틱 데이터 정의를 범용적으로 만든 것
Server Sent Event	서버로부터의 데이터 푸시
Web Socket API	브라우저와 사용자 간의 전이중 통신 API
XMLHttpRequest Level2	파일 업로드 및 progress 이벤트 등이 추가

HTML5 유의사항

- 권장 사항
  - ✓ 쿠키보다는 웹 저장소를 이용하는 것을 권장
  - ✓ 자바스크립트 애니메이션 대신 CSS의 전환 효과를 이용하는 것을 권장
  - ✓ 캐시보다는 로컬 데이터베이스를 이용하는 것을 권장
  - ✓ 부담을 많이 주는 작업에 웹 워커(스레드)를 이용하는 것을 권장

HTML5 문서 구조

- HTML5 문서 구조
  - ✓ DOCTYPE 선언

    <!DOCTYPE html>
  - ✓ 문서의 MIME 타입

    text/html
  - ✓ 문자 인코딩 지정

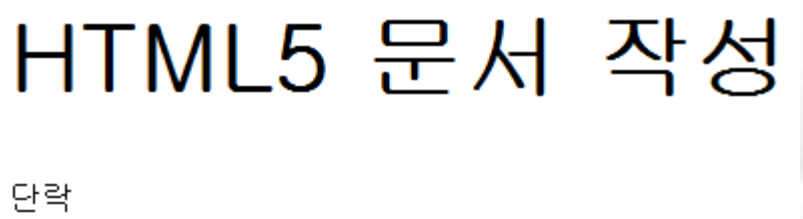
    <meta charset = "문자셋">

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

HTML5 문서 구조

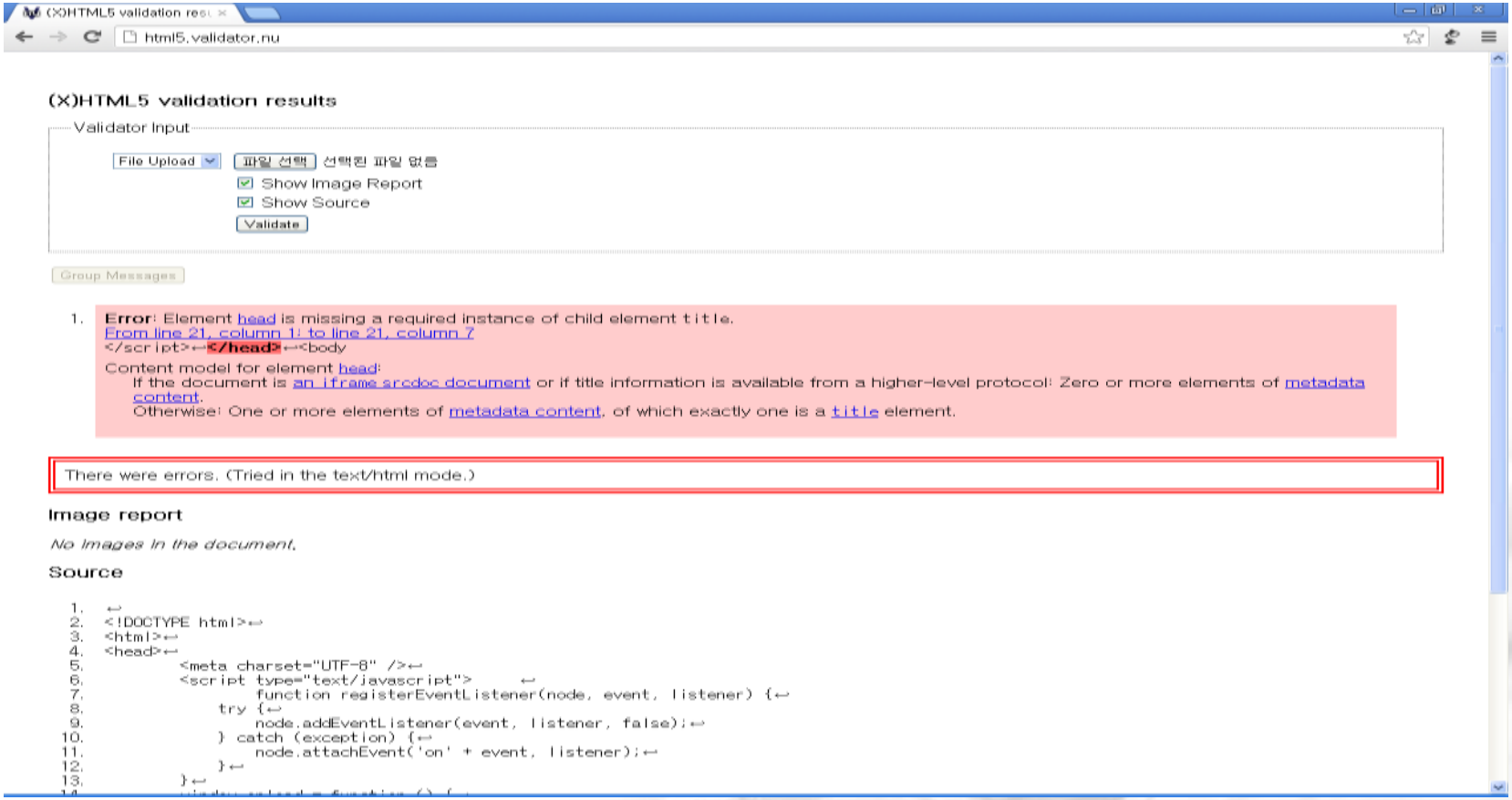
- basic.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>HTML5 Sample</title>
        <meta charset="utf-8" />
        <style type="text/css">h1 { font-size: 3em; }</style>
    </head>

    <body>
        <h1>HTML5 문서 작성</h1>
        <p>단락</p>
    </body>
</html>
```



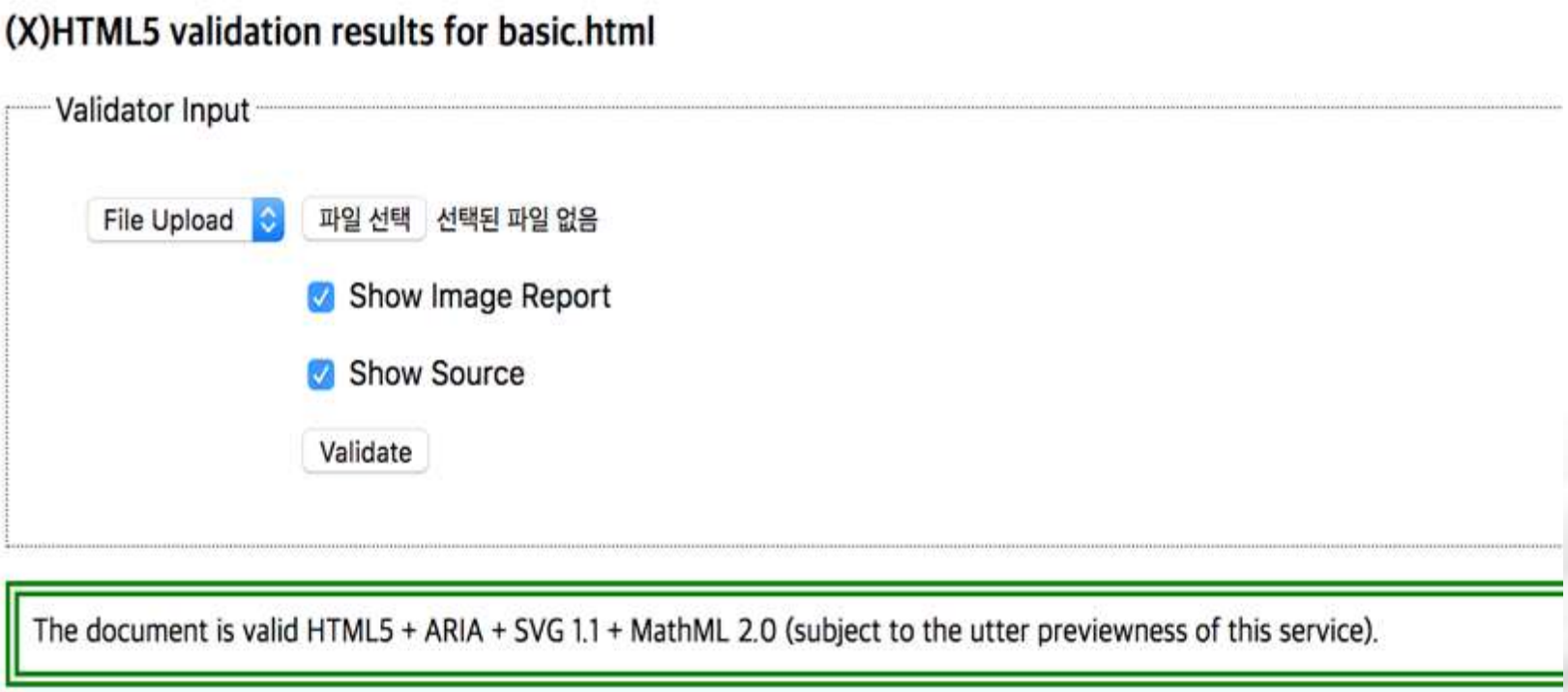
HTML5 문서 구조

- 문서 검증
  - ✓ http://html5.validator.nu 사이트에서 검증 가능



HTML5 문서 구조

- 문서 검증
  - ✓ http://html5.validator.nu 사이트에서 검증 가능



HTML5 문서 구조

- HTML5를 지원하지 않는 브라우저를 위한 코딩
  - ✓ 적용 범위 확인

    http://caniuse.com/#index
  - ✓ 블록 단위로 적용되는 새로운 태그는 CSS에서 display를 block으로 적용

    address, article, aside, figure, footer, header, hgroup, menu, nav, section{
    display: block; }
  - ✓ IE6에서 지원하지 않는 속성 들은 display를 hidden으로 설정하고 JavaScript를 이용해서 표시

    [hidden], command, datalist, menu[type=context], rp, source { display:none;}

    <!--[if IE]>

    document.createElement("command");

    <![endif]-- >

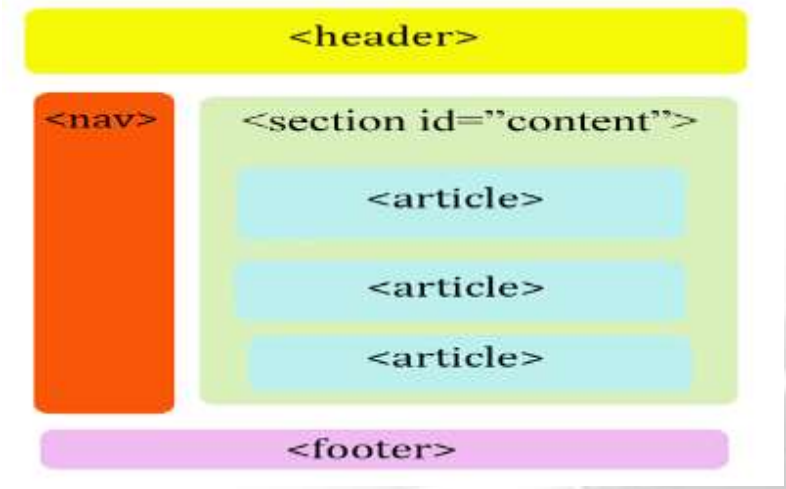
HTML5 문서 구조

- HTML5를 지원하지 않는 브라우저를 위한 코딩
  - ✓ 외부 라이브러리 이용

```
<!--[if lt IE 9]>
<script src="http://ie7-js.googlecode.com/svn/version/2.1(beta4)/IE9.js"></script>
<![endif]-->

<!--[if lt IE 8]>
<script src="http://ie7-js.googlecode.com/svn/version/2.1(beta4)/IE8.js"></script>
<![endif]-->

<!--[if lt IE 7]>
<script src="http://ie7-js.googlecode.com/svn/version/2.1(beta4)/IE7.js"></script>
<![endif]-->
```

Markup

- 추가된 섹션 요소 – 이전에는 일반적으로 div나 span 요소를 이용해서 작성
  - ✓ header: 사이트에 대한 소개 정보나 메인 메뉴, 사이트 로고 등을 포함하는 요소
  - ✓ section: 범용적인 섹션-여러 개의 콘텐츠를 묶어서 큰 논리적인 단위를 생성해주는 요소
  - ✓ article: 블로그의 본문 등 페이지의 다른 부분으로부터 독립된 섹션
  - ✓ aside: 내용에서 분리되어도 문제없는 섹션
  - ✓ nav: 다른 웹 페이지/웹 애플리케이션/웹 사이트와 관련된 네비게이션 메뉴를 표현
  - ✓ footer: 작성자 정보나 저작권 정보 또는 관련 문서 링크 등 부가 정보를 표현



Markup

- 추가된 섹션 요소

Markup

- 추가된 섹션 요소
  - ✓ semantic.html

```
<!doctype html>
<html>

<head>
    <meta charset="utf-8">
    <title>헤더 이용</title>
    <style>
        header {
            height: 280px;
            margin-top: 0;
            background-image: url(images/bg.png);
            background-repeat: no-repeat;
            background-position: left top;
        }
```

Markup

- 추가된 섹션 요소
  - ✓ semantic.html

```
.navi {
        background: #003E00;
        margin-top: -15px;
        width: 967px;
        height: 60px;
}

.navi ul {
        list-style: none;
        height: 40px;
        padding-top: 10px;
        padding-bottom: 5px;
}

.navi ul li {
        display: inline;
        float: left;
        font-size: 15px;
}
```

Markup

- 추가된 섹션 요소
  - ✓ semantic.html

```
                .navi a,
                .navi a:visited {
                        padding: 10px 5px 5px 35px;
                        display: block;
                        color: #fff;
                        width: 150px;
                        text-decoration: none;
                }

                .navi a:hover,
                .navi a:active,
                .navi a:focus {
                        text-shadow: 0px 2px 2px #000;
                        color: #FC0;
                }
        </style>
        <!--[if lt IE 9]>
                <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
        <![endif]-->
</head>
```

Markup

- 추가된 섹션 요소
  - ✓ semantic.html

```
<body>
    <div class="container">
        <header>
            <h1>HTML5 Sementic 태그</h1>
        </header>
        <nav class="navi">
            <ul>
                <li><a href="#">header</a></li>
                <li><a href="#">nav</a></li>
                <li><a href="#">section</a></li>
                <li><a href="#">article</a></li>
            </ul>
        </nav>
    </div>
</body>

</html>
```

Markup

- semantic
  - ✓ section 태그는 문서에서 가장 관련성 높은 정보를 배치하며 여러 개의 구역이나 열로 나눌 수 있음
  - ✓ aside는 웹 사이트의 레이아웃에서 사이드 바에 해당하는 태그로 메인 정보와 관련은 있지만 비교적 관련성이 높지 않고 덜 중요한 데이터가 있는 열이나 섹션
  - ✓ footer는 문서의 끝이나 하단에 표시하는 내용을 만들 때 사용하는 태그인데 일반적으로 저작권, 법규, 제약사항 같은 소유자나 회사와 관련된 정보를 공유하는데 사용
  - ✓ aside 나 footer는 section 태그 안에서 독립적인 내용을 표시할 때 사용
  - ✓ hgroup 태그는 header 태그 안에서 관련성있는 제목들을 묶을 때 사용

Markup

- semantic

웹 애플리케이션의 메인 제목

- 홈
- 사진
- 비디오
- 연락처

첫번째 큰 제목

첫번째 작은 제목

첫번째 내용

첫번째 본문

주석입니다.

두번째 큰 제목

두번째 작은 제목

두번째 내용

두번째 본문

주석입니다.

번호1

번호2

문서 전체의 바닥글

Markup

- semantic
  - ✓ html5semantic.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>HTML5 Semantic Tag</title>
</head>
<body>
    <header>
      <h1>웹 애플리케이션의 메인 제목</h1>
    </header>
    <nav>
      <ul>
            <li>홈</li>
            <li>사진</li>
            <li>비디오</li>
            <li>연락처</li>
      </ul>
    </nav>
```

Markup

- semantic
  - ✓ html5semantic.html

```
<section>
  <article>
        <header>
              <hgroup>
                    <h1>첫번째 큰 제목</h1>
                    <h2>첫번째 작은 제목</h2>
              </hgroup>
              <p>첫번째 내용</p>
        </header>
        첫번째 본문
        <footer>
              <p>주석입니다.</p>
        </footer>
  </article>
```

Markup

- semantic
  - ✓ html5semantic.html

```
<article>
    <header>
        <hgroup>
            <h1>두번째 큰 제목</h1>
            <h2>두번째 작은 제목</h2>
        </hgroup>
        <p>두번째 내용</p>
    </header>
    두번째 본문
    <footer>
        <p>주석입니다.</p>
    </footer>
</article>
</section>
<aside>
  <blockquote>번호1</blockquote>
  <blockquote>번호2</blockquote>
</aside>
<footer> 문서 전체의 바닥글 </footer>
</body>
</html>
```

Markup

- figure, figcaption
  - ✓ 이미지나 그래프, 사진, 예제 코드 등 자체적으로도 콘텐츠 형태를 이루는 것들을 문서 안에 삽입할 때 사용하는 태그로 figure 요소의 내부에 dt 요소를 사용해서 caption을 만들고 dd 요소를 이용하여 삽입할 내용을 작성
  - ✓ 캡션을 만들 때 figcaption이나 legend 태그도 가능

Markup

- 텍스트 강조
  - ✓ mark - 둘러싸인 텍스트를 강조하는 역할로 사용되는 태그
  - ✓ 태그는 이전에 사용하던 <i>
  - ✓ 태그는 중요 표시에 사용
  - ✓ <b>태그는 텍스트를 굵게

Markup

- figure.html

```
<!DOCTYPE html>
<html>
<head>
<title>시맨틱 태그</title>
<meta charset="utf-8" />
<!--[if lt IE 9]>
<script src="//html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
</head>
<body>
    <figure>
      <img src="http://cfile202.uf.daum.net/image/1960C644501781AD20E86B"
            alt="수지 사진">
      <figcaption>그림 실행 결과</figcaption>
    </figure>
    <br />
    <br /> 안녕하세요
    <mark> 박문석</mark>
    입니다.
</body>
</html>
```

Markup

- 루비
  - ✓ 글자 위에 조그마한 글자를 표시하는 기능으로 한자 등의 발음을 표시할 때 많이 사용
  - ✓ 본문과 루비로 표시될 문자를 ruby 태그로 감싸고 rt 태그로 루비로 표시될 문자를 감싸며 rp 태그에는 루비를 지원하지 않는 브라우저에서 출력되는 문자를 지정

Markup

- 루비
  - ruby.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>시맨틱 태그</title>
</head>

<body>
    <ruby>
      본
      <rp>(</rp>
      <rt>주</rt>
      <rp>)</rp>
      문
      <rp>(</rp>
      <rt>석</rt>
      <rp>)</rp>
    </ruby>
</body>
</html>
```

멀티미디어

- 멀티미디어 관련 태그
  - ✓ video, audio 요소는 멀티미디어에 관련된 태그
  - ✓ Video, Audio는 자바 스크립트 객체로도 추가되어 있음
  - ✓ HTML5이전에서는 외부 플러그 인을 이용해서 동영상이나 오디오를 재생했지만 HTML5에서는 외부 플러그인 없이 동영상이나 오디오를 재생하는 것을 권장
  - ✓ Firefox3.5, Safari4, Google Chrome3 등의 브라우저가 video/audio 요소를 지원
  - ✓ JavaScript를 이용하면 여러 가지 조작 가능(동영상을 재생하거나 중지 또는 재생 속도 변경 등)
  - ✓ 최근에 webm 파일이 추가됨
  - ✓ 브라우저가 지원하는 형식

브라우저	파일 형식	동영상 코덱	오디오 코덱
파이어폭스	Ogg	Theora	Vorbis
크롬	Ogg/MP4	Theora/H.264	Vorbis/AAC/MP3
사파리	MP4	MPEG/H.264	AAC/MP3
IE	MP4	H.264	AAC

멀티미디어

- video
  - ✓ 기본 연결: <video src = "비디오 파일 url"> </video>
  - ✓ 속성: loop(다시 재생), controls(컨트롤 바 출력), autoplay(자동으로 재생)
  - ✓ 태그 안에 문자열을 기술하면 재생되지 않는 브라우저 인 경우 표시
  - ✓ width: 동영상 재생 영역의 넓이
  - ✓ height: 동영상 재생 영역의 높이
  - ✓ poster: 동영상 프레임이 다운로드 될 때까지 대신 표시할 이미지 URL
  - ✓ type: 데이터의 MIME 타입
  - ✓ media: 미디어 쿼리를 지정

멀티미디어

- video

멀티미디어

- video
  - ✓ video.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>비디오 재생</title>
<style>
body {
    background-color: "#FFFFFF";
    color: "#000000"
}
</style>
</head>
<body>
    <video id="video" src="resource/Painting.mp4" width="320" height="320"
      controls="controls">
    </video>
</body>
</html>
```

멀티미디어

- video
  - ✓ video 태그의 src 속성에 경로를 지정하면 Codec 문제 때문에 재생이 안될 수 있음
  - ✓ video 태그 안에 source 태그를 이용해서 여러 비디오 파일들의 URL을 지정 할 수 있음
  - ✓ 여러 개의 source 태그를 설정하면 재생 가능한 파일을 만나면 재생하고 태그를 빠져 나감

    <source src="재생 파일 URL" />
  - ✓ video 태그를 지원하지 않는 브라우저에서 멀티미디어를 재생하고자 하면 object 태그 이용

멀티미디어

- video

멀티미디어

- video
  - ✓ videoplay.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>비디오 재생</title>
    <style>
      body {
            background-color: "#FFFFFF";
            color: "#000000"
      }
    </style>
</head>

<body>
    <video id="video" width="320" height="320" controls="controls">
      <source src="resource/IPhone3G.mov" />
      <source src="resource/Painting.mp4" />
    </video>
</body>
</html>
```

멀티미디어

- video
  - ✓ 속성 과 메서드
    - src: 재생할 파일의 경로
    - autoplay: 자동 재생 여부
    - loop: 반복 재생 여부
    - currentSrc: 실제로 읽어서 재생 중인 미디어 데이터의 URL
    - currentTime: 현재 재생 위치를 초 단위로 반환
    - startTime: 재생할 위치를 지정
    - duration: 데이터의 길이를 초 단위로 반환
    - paused: 일시 정지 여부
    - playbackRate: 재생 속도
    - ended: 종료 여부
    - muted: 음소거 여부
    - buffered: 얼만큼 버퍼에 로드했는지 저장하고 있는 속성
    - volume: 볼륨(0.0 – 1.0)

멀티미디어

- video
  - ✓ 속성 과 메서드
    - preload: none, metadata, auto를 설정할 수 있는데 none은 사용자가 조작하기 전까지 다운로드 받지 않으며 metadata를 설정하면 메타 데이터만 다운로드하고 auto는 브라우저가 재생하기 위해서 전부 다운로드
    - error: 에러가 발생한 경우에 에러에 대한 정보를 저장
    - load(): 동영상 다시 읽기
    - play(): 동영상 재생
    - pause(): 동영상 일시 정지

멀티미디어

- video
  - ✓ 이벤트
    - play
    - playing
    - timeupdate
    - waiting
    - ended
    - pause
    - emptied: 로딩과 함께 발생하는 이벤트
    - loadstart: 로딩 시작
    - progress: 데이터 로딩 중에 주기적으로 발생하는 이벤트
    - loadedmetadata: 미디어의 메타데이터를 읽어 들이고 있는 중
    - loadeddata: 데이터 로딩 완료
    - canplay: 재생 가능
    - canplaythrough: 마지막 까지 재생 가능
    - load: 다운로드 완료
    - error: 에러 발생

멀티미디어

- video

멀티미디어

- video
  - ✓ videocontrol.html

```
<!DOCTYPE html>
<html lang="ko">

<head>
	<title>Video Player</title>
	<style>
		body {
			text-align: center;
		}

		header,
		section,
		footer,
		aside,
		nav,
		article,
		figure,
		figcaption,
		hgroup {
			display: block;
		}
```

멀티미디어

- video
  - ✓ videocontrol.html

```
#player {
        width: 720px;
        margin: 20px auto;
        padding: 5px;
        background: #999999;
        border: 1px solid #666666;
        -moz-border-radius: 5px;
        -webkit-border-radius: 5px;
        border-radius: 5px;
}

nav {
        margin: 5px 0px;
}

#buttons {
        float: left;
        width: 85px;
        height: 20px;
}
```

멀티미디어

- video
  - ✓ videocontrol.html

```
#bar {
        position: relative;
        float: left;
        width: 600px;
        height: 16px;
        padding: 2px;
        border: 1px solid #CCCCCC;
        background: #EEEEEE;
}

#progress {
        position: absolute;
        width: 0px;
        height: 16px;
        background: rgba(0, 0, 150, .2);
}
</style>
```

멀티미디어

- video
  - ✓ videocontrol.html

```
<script>
    function initiate() {
        maxim = 600;
        mmedia = document.getElementById('media');
        play = document.getElementById('play');
        bar = document.getElementById('bar');
        progress = document.getElementById('progress');

        play.addEventListener('click', push, false);
        bar.addEventListener('click', move, false);
    }
```

멀티미디어

- video
  - ✓ videocontrol.html

```
function push() {
        if (!mmedia.paused && !mmedia.ended) {
                mmedia.pause();
                play.innerHTML = 'Play';
                window.clearInterval(loop);
        } else {
                mmedia.play();
                play.innerHTML = 'Pause';
                loop = setInterval(status, 1000);
        }
}
```

멀티미디어

- video
  - ✓ videocontrol.html

```
function status() {
    if (!mmedia.ended) {
        var size = parseInt(mmedia.currentTime * maxim /
mmedia.duration);
        progress.style.width = size + 'px';
    } else {
        progress.style.width = '0px';
        play.innerHTML = 'Play';
        window.clearInterval(loop);
    }
}
```

멀티미디어

- video
  - ✓ videocontrol.html

```
        function move(e) {
            if (!mmedia.paused && !mmedia.ended) {
                var mouseX = e.pageX - bar.offsetLeft;
                var newtime = mouseX * mmedia.duration / maxim;
                mmedia.currentTime = newtime;
                progress.style.width = mouseX + 'px';
            }
        }
        window.addEventListener('load', initiate, false);
    </script>
</head>
```

멀티미디어

- video
  - ✓ videocontrol.html

```
<body>
    <section id="player">
        <video id="media" width="720" height="400">
            <source src="resource/Painting.mp4">
            <source src="resource/IPhone3G.mov">
        </video>
        <nav>
            <div id="buttons">
                <button type="button" id="play">Play</button>
            </div>
            <div id="bar">
                <div id="progress"></div>
            </div>
            <div style="clear: both"></div>
        </nav>
    </section>
</body>

</html>
```

멀티미디어

- audio
  - ✓ <audio> 태그는 음악이나 오디오 스트림과 같은 사운드를 정의할 때 사용
  - ✓ <audio> 요소 내에 위치하는 텍스트는 사용자의 브라우저가 <audio> 요소를 지원하지 않을 경우 화면에 표시
  - ✓ <audio> 요소는 현재 다음 세 가지 포맷의 파일을 지원
    - MP3: Moving Picture Experts Group에 의해 개발되었으며 MPEG-1의 오디오 규격으로 개발된 손실 압축형 파일 형식
    - WAV: IBM과 Microsoft에 의해 개발되었으며 개인용 PC에서 오디오를 재생하기 위한 IBM과 Microsoft의 표준 오디오 파일 형식
    - Ogg: Xiph 재단에 의해 개발되었으며 MP3의 대안으로 개발된 특허권으로 보호되지 않는 개방형 공개 멀티미디어 파일 형식
  - ✓ video 태그에 존재하는 속성 과 메서드를 소유하고 있음

멀티미디어

- audio
  - ✓ audioplay.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>오디오 재생</title>
</head>

<body>
    <input type="button" id="play" value="재생" />
    <div id="time"></div>
</body>
```

멀티미디어

- audio
  - ✓ audioplay.html

```
<script>
    var audio = new Audio("resource/sound.mp3");
    audio.addEventListener("timeupdate", function () {
      var timeDisplay = document.getElementById("time");
      timeDisplay.innerHTML = Math.floor(audio.currentTime) + "/" +
            Math.floor(audio.duration) + ""
    }, false);

    audio.voulume = 0.5;
    audio.loop = true;

    document.getElementById("play").addEventListener("click", function () {
      audio.play()
    })
</script>

</html>
```

멀티미디어

- canvas
  - ✓ canvas는 그래픽을 자유롭게 출력 할 수 있는 html5에서 추가된 API
  - ✓ 고정된 넓이와 높이를 가지고 그 영역 내에 원이나 사각형 등의 도형 및 이미지 등의 2D 그래픽을 자유롭게 그릴 수 있으며 khronos 그룹에 의해서 WebGL 사양을 기반으로 하는 3D 그래픽 캔버스도 실험적으로 적용되고 있음(https://www.khronos.org/webgl/, http://webglsamples.org/)
  - ✓ internet explorer는 별도의 라이브러리(ExplorerCanvas, uuCanvas.js)를 이용해서 그릴 수 있음
  - ✓ 사용 방법
    - <canvas>태그를 이용해서 생성: 태그 안에는 캔버스를 지원하지 않는 경우 보여질 문자열을 설정

      <canvas id="캔버스id" width="너비" height="높이"> </canvas>
    - 2D 컨텍스트 생성: 캔버스에 그리기 위한 정보를 저장한 객체

      var context변수 = 캔버스객체.getContext('2d');
    - 자바스크립트에서 그리는 코드 작성 – context 변수를 이용해서 그리기 코드 작성
    - save(): 원본을 복구할 수 있게 수정되지 않은 상태의 컨텍스트 저장
    - restore(): 현재 컨텍스트를 마지막 저장된 상태로 복원

멀티미디어

- canvas
  - ✓ canvas.html

```
<!DOCTYPE html>
<html>

<head>
    <title>canvas 테스트</title>
    <meta charset="UTF-8" />
</head>

<body>
    <canvas id="canvas" width="300" height="200"> </canvas>
</body>
<script>
    var canvas = document.getElementById('canvas');
    var context = canvas.getContext('2d');
    context.fillRect(0, 0, 150, 100);
    context.fillText("안녕하세요 반갑습니다", 155, 110);
</script>

</html>
```

멀티미디어

- canvas
  - ✓ 사각형 그리기
    - strokeRect(x좌표, y좌표, 너비, 높이): 내부에 색이 칠해져 있지 않은 사각형
    - fillRect(x좌표, y좌표, 너비, 높이): 내부에 색이 칠해져 있는 사각형
    - clearRect(x좌표, y좌표, 너비, 높이): 지정한 영역의 픽셀을 지우는 사각형

멀티미디어

- canvas
  - ✓ canvasrect.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>캔버스 사각형</title>
</head>

<body>
    <canvas id="canvas" width="360" height="200"> </canvas>
</body>

<script>
    var canvas = document.getElementById('canvas');
    var context = canvas.getContext('2d');
    context.fillRect(10, 50, 140, 30);
    context.strokeRect(10, 100, 200, 30);
</script>

</html>
```

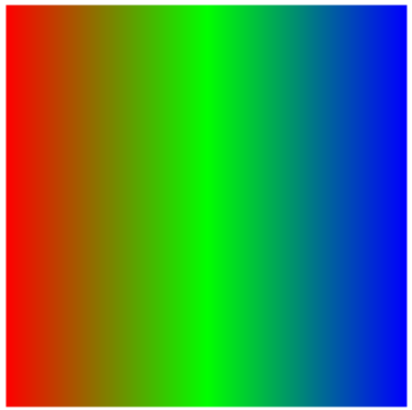
멀티미디어

- canvas
  - ✓ 색상
    - strokeStyle 속성을 이용해서 선의 색상이나 스타일을 지정
    - fillStyle 속성을 이용해서 도형 내부 색상을 선언
    - 색상 적용 방법
      - CSS에 의한 적용
      - 그라데이션 적용
      - 패턴 적용
    - CSS에 의한 색상 적용
      - 색상명으로 가능('red')
      - 색상값으로 적용 가능('#FF0000')
      - hsl(0,100%,50%)
      - rgb(255,0,0)
    - globalAlpha: 투명도 설정

멀티미디어

- canvas
  - ✓ 그라데이션
    - 그라데이션 객체 생성
      - createLinearGradient(sx,sy, ex,ey): (sx,sy) ~ (ex,ey) 까지 직선형 그라디언트 객체 생성
      - createRadialGradient(sx,sy,sr, ex,ey,er): (sx,sy) 중점의 반지름 sr인 원, (ex,ey)중점의 반지름 er인 원을 사용한 원형 그라디언트 객체 생성
    - 그라데이션 객체 생성 후 addColorStop(position, color)로 색 설정
      - position 시작점 : 0.0, 끝점 : 1.0
      - color에는 CSS 컬러를 문자열로 지정 가능
    - fillStyle에 그라디언트 객체 설정

멀티미디어



- canvas
  - ✓ canvasgradient.html

```
<!DOCTYPE html>
<html>
<head>
    <title>canvas 그라디언트</title>
    <meta charset="utf-8" />
</head>

<body>
    <canvas id="canvas" width="300" height="400"> </canvas>
</body>

<script>
    var canvas = document.getElementById('canvas');
    var context = canvas.getContext('2d');
    var g = context.createLinearGradient(0, 0, 200, 0);
    g.addColorStop(0, "red");
    g.addColorStop(0.5, '#00FF00');
    g.addColorStop(1, 'rgb(0,0,255)');
    context.fillStyle = g;
    context.fillRect(0, 0, 200, 200);
</script>
</html>
```

멀티미디어

- canvas
  - ✓ Path
    - beginPath(): 새로운 도형을 위한 패스 시작
    - closePath(): 패스를 닫고 마지막 점에서 원점까지 직선을 생성하는데 fill()을 호출하거나 오픈 패스를 그리고자 하는 경우 생략 가능
    - 패스를 선으로 표시: stroke()
    - 패스 내부를 채움: fill()
    - 패스 내부를 클리핑 영역으로 설정: clip()

멀티미디어

- canvas
  - ✓ Path
    - beginPath(): 새로운 도형을 위한 패스 시작
    - closePath(): 패스를 닫고 마지막 점에서 원점까지 직선을 생성하는데 fill()을 호출하거나 오픈 패스를 그리고자 하는 경우 생략 가능
    - 패스를 선으로 표시: stroke()
    - 패스 내부를 채움: fill()
    - 패스 내부를 클리핑 영역으로 설정: clip()

멀티미디어

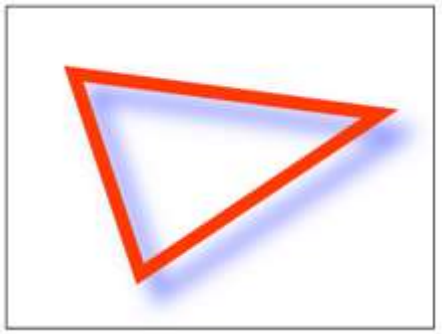
- canvas
  - ✓ Path
    - 선
      - moveTo(x좌표, y좌표): 원점 이동
      - lineTo(x좌표, y좌표): 원점에서 입력된 좌표까지 선 만들기
    - 사각형
      - rect(x좌표, y좌표, width, height)
    - 2차 베이지어 곡선
      - quadraticCurveTo(cpx, cpy, x, y): 현재 위치에서 곡선을 생성하는데 x,y는 마지막 지점의 좌표이고 cpx, cpy는 곡선이 생성되는 제어점(control point)의 좌표
    - 3차 베이지어 곡선
      - bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x,y): 제어점이 2개
    - 호 그리기
      - arcTo(x1, y1, x2, y2, 반지름): 현재 위치에서 x1,y1으로 직선을 그리고 이어서 x2, y2까지 이어진 직선을 그린 후 반지름 크기인 원 호를 그림
      - arc(x, y, radius, startAngle, endAngle, antiClockWise): x, y를 중점으로 해서 radius를 반지름으로 갖는 원을 그린 후 startAngle부터 endAngle까지만 antiClockWise 방향으로 원호를 그림

멀티미디어

- canvas
  - ✓ Path
    - 선 스타일
      - lineWidth: 선 두께
      - lineCap: 선의 끝점 속성(butt, round, square)
      - lineJoin: 교차점(bevel, round, miter)
      - miterLimit: miter 지정 시 끝점의 길이
    - 그림자 효과
      - shadowColor: 그림자 색상
      - shadowOffsetX: 그림자 x 좌표
      - shadowOffsetY: 그림자 y좌표
      - shadowBlur: 그림자 흐림의 정도

멀티미디어

- canvas
  - ✓ canvasshadow.html



```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>그림자</title>
</head>

<body>
    <canvas id="canvas" width="320" height="240" style="border: 1px black solid">
    </canvas>
</body>
```

멀티미디어

- canvas
  - ✓ canvasshadow.html

```
<script>
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");
    context.strokeStyle = "rgba(255,0,0,1)";
    context.lineWidth = 10;
    context.shadowColor = "rgba(0,0,255,0.5)";
    context.shadowBlur = 15;
    context.shadowOffsetX = 15;
    context.shadowOffsetY = 15;

    context.beginPath();
    context.moveTo(50, 50);
    context.lineTo(280, 80);
    context.lineTo(100, 200);
    context.closePath();
    context.stroke();
</script>

</html>
```

멀티미디어

- canvas
  - ✓ 텍스트
    - fillText(문자열, x, y, [가로 폭 너비]);
      - 채워진 텍스트로 x, y 좌표에 출력하게 되고 가로 폭 너비에 맞게 출력
      - fillStyle을 이용해서 색이나 스타일을 설정
    - strokeText(문자열, x, y, [가로 폭 너비]);
      - 외곽선만 채워진 텍스트로 x, y 좌표에 출력하게 되고 가로 폭 너비에 맞게 출력
      - strokeStyle을 이용해서 색이나 스타일을 설정
    - font = "크기 서체";
    - textAlign="값";
      - 가로 방향 표시 위치(left, right, center, start, end)
    - textBaseline="값";
      - 텍스트가 표시되는 위치의 기준선으로 세로 정렬(top, hanging, middle, alphabetic, ideographic, bottom)

멀티미디어

- canvas
  - ✓ canvastext.html


```
<!DOCTYPE html>
<html>

<head>
    <title>Canvas API</title>
</head>

<body>
    <section id="canvasbox">
      <canvas id="canvas" width="500" height="300">
      </canvas>
    </section>
</body>
```

멀티미디어

- canvas
  - ✓ canvastext.html

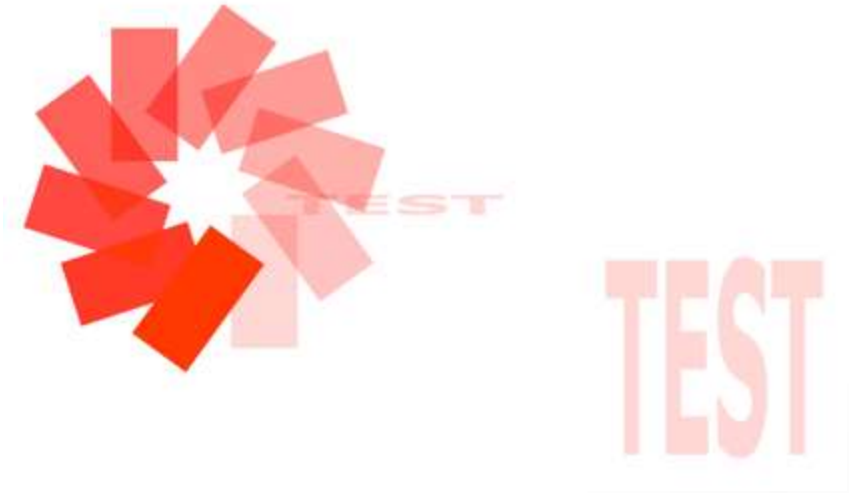
```
<script>
    var canvas = document.getElementById('canvas');
    context = canvas.getContext('2d');

    context.shadowColor = "rgba(0,0,0,0.5)";
    context.shadowOffsetX = 4;
    context.shadowOffsetY = 4;
    context.shadowBlur = 5;

    context.font = "bold 50px verdana, sans-serif";
    context.fillText("my message", 100, 100);
</script>

</html>
```

멀티미디어

- canvas
  - ✓ 위치 이동: translate(x,y);
    - 원점 위치 이동
  - ✓ 회전: rotate(각도);
    - 각도는 radian 사용
  - ✓ 확대/축소
    - scale(가로 배율, 세로 배율);
    - 캔버스 기본단위 : 1px이지만 2배 확대 하면 기본 단위가 2px
  - ✓ transform(m1, m2, m3, m4, m5, m6): 새로운 행렬 적용
    - m1: scale-x
    - m2: skew-x
    - m3: skew-y
    - m4: scale-y
    - m5: translate-x
    - m6: translate-y

멀티미디어

- canvas



멀티미디어

- canvas
  - ✓ canvastransform.html

```
<!DOCTYPE html>
<html>

<head>
    <title>canvas 테스트</title>
    <meta charset="utf-8" />
</head>

<body>
    <canvas id="canvas" width="1024" height="768"> </canvas>
</body>
```

멀티미디어

- canvas
  - ✓ canvastransform.html

```
<script>
    var canvas = document.getElementById('canvas');
    var context = canvas.getContext('2d');

    context.translate(150, 150);
    context.fillStyle = "red";
    for (var i = 0; i < 10; i++) {
      context.globalAlpha = (1 - (0.09 * i));
      context.rotate(((Math.PI / 180) * 360) / 10);
      context.fillRect(20, 20, 50, 100);
    }

    context.transform(3, 0, 0, 1, 0, 0);
    context.font = "bold 20px verdana, sans-serif";
    context.fillText("TEST", 20, 20);

    context.transform(1, 0, 0, 10, 0, 0);
    context.font = "bold 20px verdana, sans-serif";
    context.fillText("TEST", 100, 20);
</script>

</html>
```

멀티미디어

- canvas
  - ✓ 겹쳐지는 부분 처리: globalCompositeOperation = 값
    - source-over(기본) : source가 가림
    - source-in : source는 겹쳐지는 부분만 출력
    - source-out : source는 겹쳐지지 않는 부분만 출력
    - source-atop : 먼저 그린 그림+나중 그림 겹치는 부분

    - destination-over : 먼저 그린 그림이 덮음
    - destination-in : 먼저 그린 그림 중 겹쳐지는 부분만 표시
    - destination-out : 먼저 그린 그림 중 겹치지 않는 부분만 표시
    - destination+atop : 나중에그린 그림 + 먼저 그림 겹치는 부분

    - lighter : 모두 표시, 겹치는 부분은 색상값 합쳐서 결정 - or
    - darker : 모두 표시, 겹치는 부분은 색상값 빼서 결정 - and
    - xor : 모두 표시, 겹치는 부분은 비움

멀티미디어

- canvas
  - ✓ canvascomposition.html

```
<!DOCTYPE html>
<html>

<head>
    <title>canvas 테스트</title>
    <meta charset="utf-8" />

</head>

<body>
    <canvas id="canvas" width="1024" height="768"> </canvas>
</body>
```

멀티미디어

- canvas
  - ✓ canvascomposition.html

```
<script>
    var canvas = document.getElementById('canvas');
    var context = canvas.getContext('2d');
    context.globalCompositeOperation = "lighter";
    context.fillStyle = "red";
    context.fillRect(0, 0, 100, 100);
    context.fillStyle = "green";
    context.fillRect(50, 50, 100, 100);
```

멀티미디어

- canvas
  - ✓ canvascomposition.html

```
        /*
        context.fillStyle="#990000";
        context.fillRect(100,100,300,100);
        context.globalCompositeOperation="destination-atop";
        context.fillStyle="#AAAAFF";
        context.font="bold 80px verdana, sans-serif";
        context.textAlign="center";
        context.textBaseline="middle";
        context.fillText("TEST",250,110);
        */
    </script>
    </html>
```

멀티미디어

- canvas
  - ✓ Image 출력
    - Image 객체 생성

      new Image();
    - 이미지 경로 지정

      Image객체.src = "이미지 파일 경로";
    - 이미지가 전부 로드되면 발생하는 이벤트: load
    - 캔버스에 이미지 출력
      - context변수.drawImage(Image객체, x좌표, y좌표);
      - context변수.drawImage(Image객체, x좌표, y좌표, 너비, 높이);
      - context변수.drawImage(Image객체, x좌표, y좌표, 자르는 너비, 자르는 높이, x좌표, y좌표, 너비, 높이);
    - 주의 사항: 이미지가 전부 로드 되기 전에 drawImage()를 호출하면 안됨
      - 이미지를 출력할 때는 Image객체.onload = function() {draw 메서드 호출;} 의 형태로 작성

멀티미디어

- canvas

멀티미디어

- canvas
  - ✓ canvasimage.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
</head>

<body>
    <canvas id="canvas" width="400" height="400">
    </canvas>
</body>
```

멀티미디어

- canvas
  - ✓ canvasimage.html

```
<script>
    var canvas =
      document.getElementById("canvas")
    //컨텍스트 변수 만들기
    var context = canvas.getContext("2d")

    //출력할 이미지 객체 만들기
    var image = new Image()
    image.src = "http://cfile202.uf.daum.net/image/1960C644501781AD20E86B"


    //이미지 파일을 전부 읽어서 메모리에 로드 한 후 수행
    image.addEventListener("load", function () {
      context.drawImage(image, 50, 50, 300, 300)
      //setInterval(func,3000);
    })
```

멀티미디어

- canvas
  - ✓ canvasimage.html

```
/*
//이미지 스프라이트와 타이머
var canvas =
  document.getElementById("canvas")
//컨텍스트 변수 만들기
var context = canvas.getContext("2d")

//출력할 이미지 객체 만들기
var image = new Image()
image.src="images/img.jpg"


//이미지 파일을 전부 읽어서 메모리에 로드 한 후 수행
image.addEventListener("load", function(){
  setInterval(func,3000);
})
```

멀티미디어

- canvas
  - ✓ canvasimage.html

```
        var idx = 0
        var func = function(){
          context.drawImage(image,
                        133 * Math.floor(idx/3),
                        133 * Math.floor(idx%3),
                        133, 133,
                        10,10, 266,266)
          idx = idx + 1
          if(idx == 9)
                idx = 0
        }
        */
    </script>

    </html>
```

멀티미디어

- canvas
  - ✓ canvascliping.html

```
<!DOCTYPE html>
<html>

<head>
    <title>canvas 테스트</title>
    <meta charset="UTF-8" />
</head>

<body>
    <canvas id="canvas" width="400" height="400"> </canvas>
</body>
```

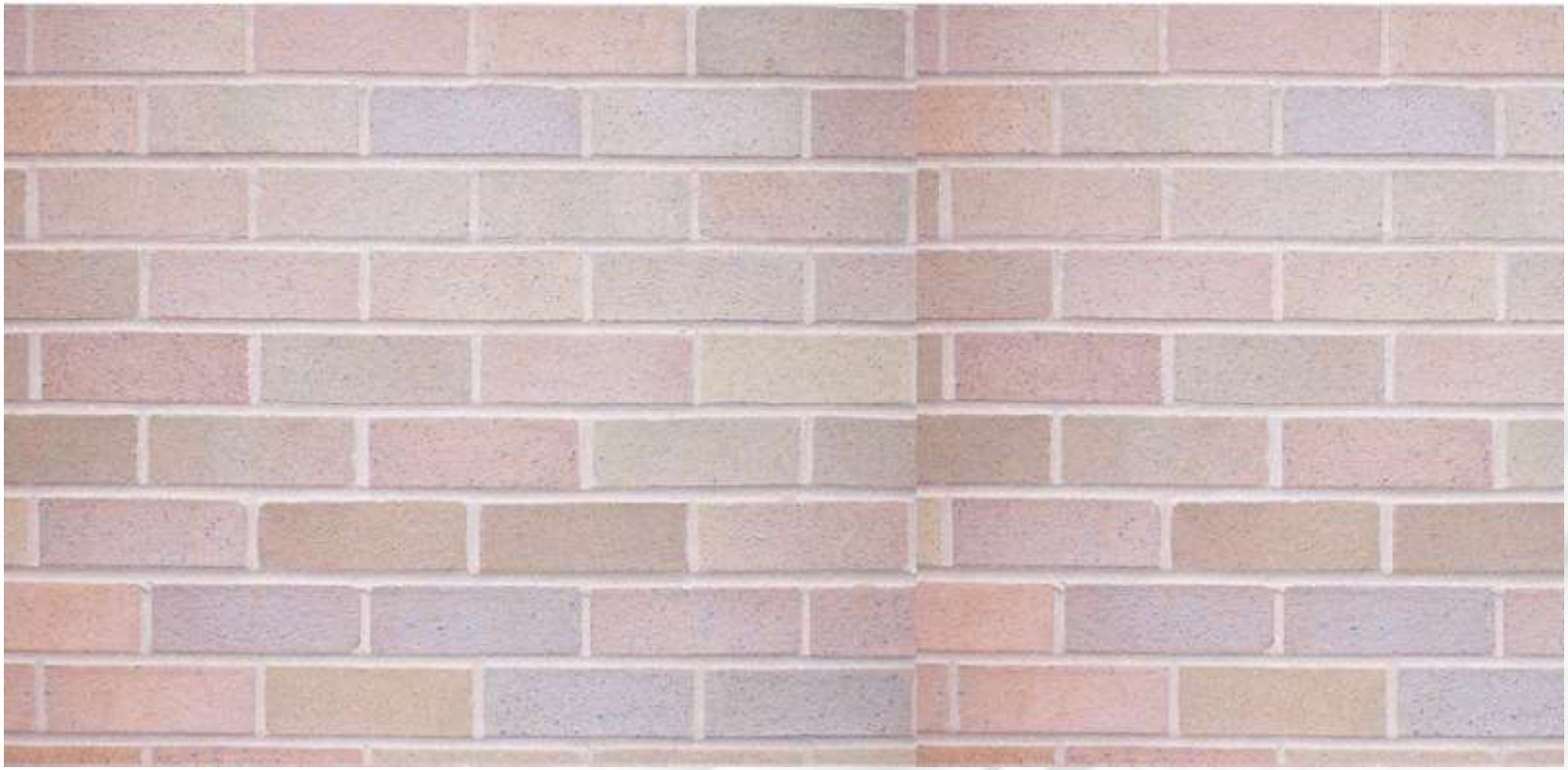
멀티미디어

- canvas
  - ✓ canvascliping.html

```
<script type="text/javascript">
    var canvas = document.getElementById("canvas");
    var cx = canvas.getContext("2d");
    cx.beginPath();
    cx.arc(200, 200, 150, 0, 2 * Math.PI, true);
    cx.clip();
    var image = new Image();
    image.onload = function () {
      cx.drawImage(image, 0, 0, canvas.width, canvas.height);
    };
    image.src = "http://cfile202.uf.daum.net/image/1960C644501781AD20E86B";
</script>

</html>
```

멀티미디어

- canvas
  - ✓ 이미지를 채워넣기
    - createPattern(객체명, type);
      - type : repeat, repeat-x, repeat-y, no-repeat



멀티미디어

- canvas
  - ✓ canvaspattern.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>Canvas API</title>

</head>

<body>
  <section id="canvasbox">
    <canvas id="canvas" width="2048" height="1024">
      Your browser doesn't support the canvas element
    </canvas>
  </section>
</body>
```

멀티미디어

- canvas
  - ✓ canvaspattern.html

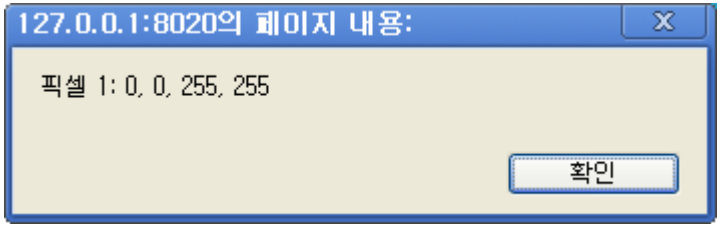
```
<script>
  var canvas = document.getElementById('canvas');
  var context = canvas.getContext('2d');

  var img = new Image();
  img.src = "images/bricks.jpg";
  img.addEventListener("load", modimage, false);

  function modimage(e) {
    img = e.target;
    var pattern = context.createPattern(img, 'repeat');
    context.fillStyle = pattern;
    context.fillRect(0, 0, 2038, 1014);
  }
</script>

</html>
```

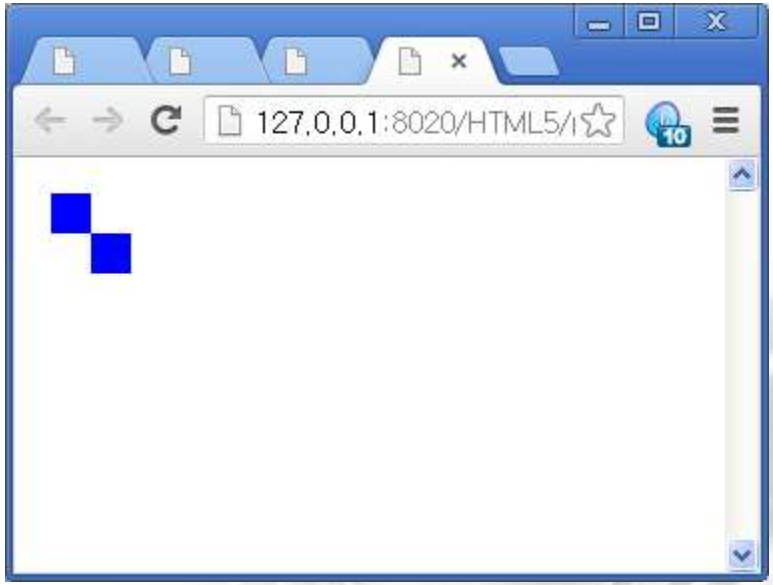
멀티미디어

- canvas
  - ✓ 픽셀 제어
    - getImageData(x좌표, y좌표, 너비, 높이)
      - 리턴되는 객체는 width, height, data 속성을 가짐
      - data는 rgba 값을 갖는 일차원 배열
    - putImageData(이미지 데이터, x좌표, y좌표): 이미지 데이터를 출력

멀티미디어

- canvas
  - ✓ 픽셀 제어





멀티미디어

- canvas
  - ✓ canvaspixel.html

```
<!DOCTYPE html>
<html>

<head>
<meta charset="UTF-8" />
<title>픽셀 데이터</title>
</head>

<body onload="init()">
<canvas id="canvas" width="300" height="300"> </canvas>
</body>
```

멀티미디어

- canvas
  - ✓ canvaspixel.html

```
<script>
function init() {
    var canvas = document.getElementById("canvas");
    if (canvas.getContext) {
      var canvas = document.getElementById("canvas");
      if (canvas.getContext) {
            var ctx = canvas.getContext("2d");
      }
      ctx.fillStyle = "rgb(0,0,255)";
      ctx.fillRect(10, 10, 20, 20);
      var Pixel = ctx.getImageData(29, 10, 2, 1);
      alert("픽셀 1: " + Pixel.data[0] + ", " + Pixel.data[1] + ", " +
            Pixel.data[2] + ", " + Pixel.data[3]);
      alert("픽셀 2: " + Pixel.data[4] + ", " + Pixel.data[5] + ", " +
            Pixel.data[6] + ", " + Pixel.data[7]);
```

멀티미디어

- canvas
  - ✓ canvaspixel.html

```
            alert("픽셀 너비: " + Pixel.width);
            alert("픽셀 높이: " + Pixel.height);
            Pixel = ctx.getImageData(10, 10, 20, 20);
            ctx.putImageData(Pixel, 30, 30);
        }
    }
    </script>

    </html>
```

멀티미디어

- canvas
  - ✓ 애니메이션

멀티미디어

- canvas
  - ✓ canvasanimation.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>Canvas API</title>
  <script>
    function initiate() {
      var elem = document.getElementById('canvas');
      canvas = elem.getContext('2d');

      window.addEventListener('mousemove', animation, false);
    }
```

멀티미디어

- canvas
  - ✓ canvasanimation.html

```
function animation(e) {
  canvas.clearRect(0, 0, 300, 500);
  var xmouse = e.clientX;
  var ymouse = e.clientY;
  var xcenter = 220;
  var ycenter = 150;
  var ang = Math.atan2(xmouse - xcenter, ymouse - ycenter);
  var x = xcenter + Math.round(Math.sin(ang) * 10);
  var y = ycenter + Math.round(Math.cos(ang) * 10);
  canvas.beginPath();
  canvas.arc(xcenter, ycenter, 20, 0, Math.PI * 2, false);
  canvas.moveTo(xcenter + 70, 150);
  canvas.arc(xcenter + 50, 150, 20, 0, Math.PI * 2, false);
  canvas.stroke();
  canvas.beginPath();
  canvas.moveTo(x + 10, y);
  canvas.arc(x, y, 10, 0, Math.PI * 2, false);
  canvas.moveTo(x + 60, y);
  canvas.arc(x + 50, y, 10, 0, Math.PI * 2, false);
  canvas.fill();
}
```

멀티미디어

- canvas
  - ✓ canvasanimation.html

```
    window.addEventListener("load", initiate, false);
  </script>
</head>

<body>
  <section id="canvasbox">
    <canvas id="canvas" width="500" height="300">
      Your browser doesn't support the canvas element
    </canvas>
  </section>
</body>

</html>
```

멀티미디어

- SVG
  - ✓ Scalable Vector Graphics 의 약자로 XML을 기반으로 한 Drawing 표준으로 벡터 그래픽을 표현할 때 사용하는 형식
  - ✓ HTML5 가 논의되기 이전부터 사용되던 기술로 HTML5 페이지 상에 선, 곡선, Path, 도형 등을 Vector 로 표현할 수 있다는 점이 Canvas 와는 다름
  - ✓ Web Page에서 Canvas는 Pixel 단위의 이미지 밖에 표현할 수 없지만 interactive 한 animation 이나 Vector graphic 을 SVG를 이용하여 표현할 수 있으며 Silverlight 나 Flash의 기능을 대체할 수 있고 일러스트레이터에서 제작이 가능
  - ✓ 표현 가능 효과
    - 기본 도형(사각형, 원, 다각형)
    - image
    - Bezier Pass, Curve
    - Text
    - Transparency
    - Transformation(회전, 기울이기, 확대/축소)
    - Gradient
    - Animation

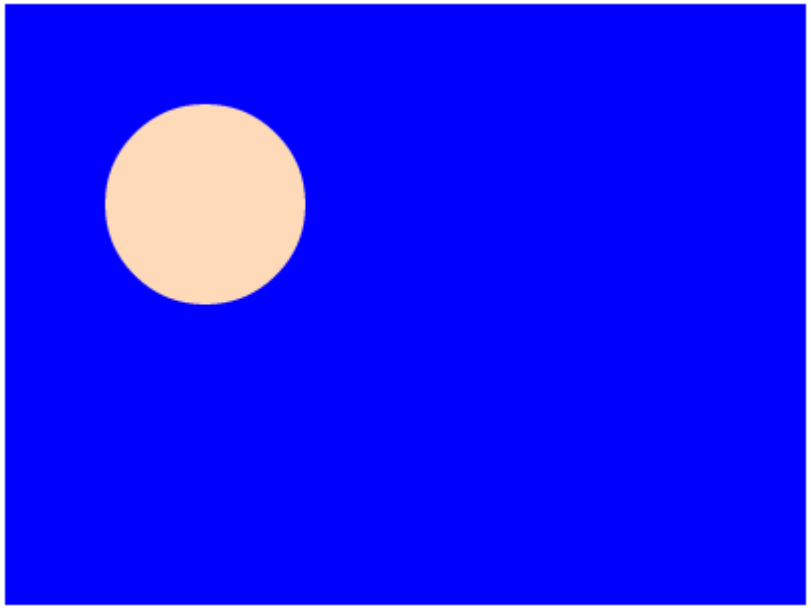
멀티미디어

- SVG
  - ✓ canvas 와 SVG 비교

	Canvas	SVG
이미지 처리방식	Bitmap	Vector
DOM	존재하지 않음	존재함(Script 로 Control 가능)
외부 이미지 편집	Bitmap image 편집 용이	Vector image 편집 용이
성능	높은 해상도의 이미지를 사용하면 성능 저하	이미지가 복잡해질수록 Markup 이 복잡해져 성능이 저하
Animation	Animation API 가 없으므로 Script 의 Timer 를 사용	높은 수준의 Animation 을 지원
Cross Browsing	모든 Browser 에서 지원하지 않음	모든 Browser 에서 지원되는 Drawing 표준
외부 이미지로 저장	jpg, png 등으로 저장 가능	불가
적합한 서비스	Game	Graph 구현 세밀한 해상도를 지원하는 UI 및 Application
부적합 서비스	Standalone Application UI	Game

멀티미디어

- SVG



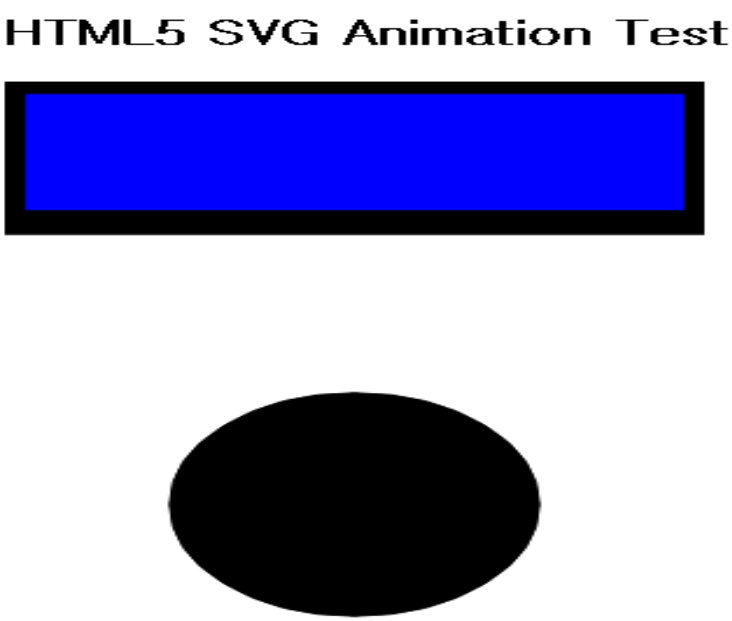
멀티미디어

- SVG
  - ✓ svg1.html

```
<!DOCTYPE html>
<html>
    <head>
      <meta charset="UTF-8" />
      <title>SVG 연습</title>
    </head>
    <body >
      <svg width="700" height="300">
       <rect width="400" height="400" fill="blue" />
       <circle cx="100" cy="100" r="50" fill="PeachPuff" />
    </svg>
    </body>
</html>
```

멀티미디어

- SVG



멀티미디어

- SVG
  - ✓ resource/svgtest.svg

```
<svg id="svgelem" height="600" 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>
```

멀티미디어

- SVG
  - ✓ svg2.html

```
<!DOCTYPE html>
<html>
<head>
    <title>SVG Test</title>
    <meta charset="utf-8" />
</head>
<body>
    <h2>HTML5 SVG Animation Test</h2>
   <img src="resource/svgtest.svg">
</body>
</html>
```

FORM

- HTML5 Form 기능의 변화
  - ✓ 여러가지 타입이 추가
  - ✓ JavaScript 라이브러리를 이용하지 않고 입력 보조나 입력 값 검증 가능
  - ✓ 입력 폼을 섬세하게 처리 가능
  - ✓ 진행 상황 표시를 위한 progress 와 meter 추가

FORM

- FORM
  - ✓ HTML5 Form 기능의 변화
    - 여러가지 타입이 추가
    - JavaScript 라이브러리를 이용하지 않고 입력 보조나 입력 값 검증 가능
    - 입력 폼을 섬세하게 처리 가능
    - 진행 상황 표시를 위한 progress 와 meter 추가

FORM

- FORM
  - ✓ 기존 input

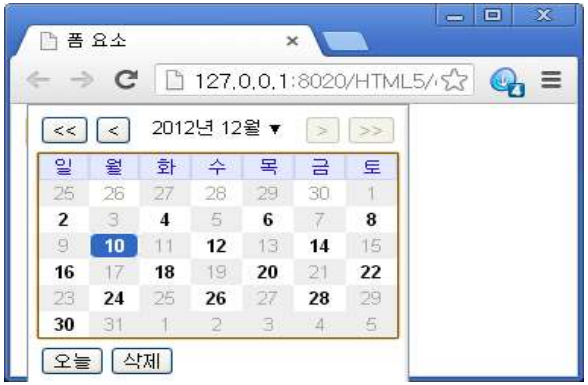
    <input type="값">
    - text : 텍스트
    - password : 패스워드
    - checkbox : 체크박스
    - radio : 라디오 버튼
    - file : 파일 업로드
    - button
    - image
    - submit : 전송 버튼
    - reset
    - hidden

FORM

- FORM
  - ✓ 추가된 input type
    - search(브라우저 표시 용)
    - tel(브라우저 표시 용)
    - url(절대 URL 검사)
    - email(email 유효성 검사)
    - number(입력이 숫자 인 경우만 유효하며 min, max, step 속성 사용 가능)
    - range(슬라이더나 화살표로 표시하여 값을 위아래 이동하면서 입력하는 것으로 아직까지 표준 디자인은 없음)
    - color
    - date, month, week, time, datetime-local

FORM

❖ FORM


폼 요소
127.0.0.1:8020/HTML5/
<<
<
2012년 12월 ▾
>
>>
일 월 화 수 목 금 토
25 26 27 28 29 30 1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31 1 2 3 4 5
오늘
삭제

FORM

- FORM
  - ✓ date.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>폼 요소</title>
    <script type="text/javascript">
      function move() {
            var calendar = document.getElementById("calendar");
            calendar.stepUp(2);
      }
    </script>
</head>

<body>
    <form>
      <input type="date" id="calendar" min="2012-01-01" max="2012-12-31" step="2"
    />
      <input id="button" type="button" value="이동" onclick="move()" />
    </form>
</body>
</html>
```

FORM

- FORM
  - ✓ date1.html

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>jQuery UI Datepicker - Default functionality</title>
    <link rel="stylesheet" href="//code.jquery.com/ui/1.10.4/themes/smoothness/jquery-
    ui.css">
    <script src="//code.jquery.com/jquery-1.9.1.js"></script>
    <script src="//code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
    <link rel="stylesheet" href="/resources/demos/style.css">
    <script>
      $(function () {
            $("#datepicker").datepicker();
      });
    </script>
</head>
<body>
    <p>
      Date: <input type="text" id="datepicker">
    </p>
</body>
</html>
```

FORM

- FORM
  - ✓ progress.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>폼 요소</title>
</head>

<body>
    <form>
      <progress id="bar" value="0" max="100"> </progress>
      <input type="button" value="시작" onclick="progress()">
    </form>
</body>
```

FORM

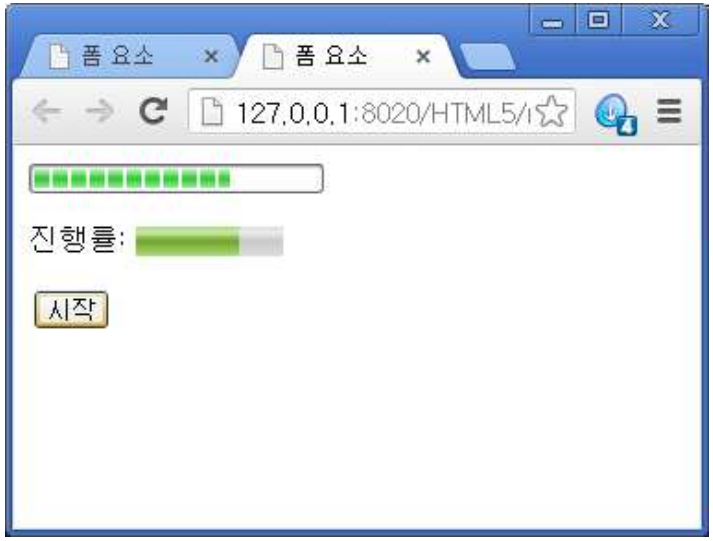
- FORM
  - ✓ progress.html

```
<script>
    function progress() {
      var bar = document.getElementById("bar");
      loaded = 0;
      var timer = setInterval(function () {
            loaded += 10;
            bar.value = loaded;
            if (loaded == 100)
                  clearInterval(timer);
      }, 1000);
    }
</script>

</html>
```

FORM

- FORM
  - ✓ meter.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>폼 요소</title>
</head>

<body>
    <form>
      <p>
            진행률:
            <meter id="meter" min="0" max="100"></meter>
      </p>
      <input type="button" value="시작" onclick="progress()">
    </form>
</body>
```



FORM

- FORM
  - ✓ meter.html

```
<script type="text/javascript">
    function progress() {
      var meter = document.getElementById("meter");
      loaded = 0;
      var timer = setInterval(function () {
            loaded += 10;
            meter.value = loaded;
            if (loaded == 100)
                  clearInterval(timer);
      }, 1000);
    }
</script>

</html>
```

FORM

- FORM
  - ✓ 변경된 Input type
    - checkbox: javaScript를 이용하여 값을 지정할 수 있는 indeterminate 속성이 추가되었으며 이 속성이 true가 되면 On/Off 가 아닌 상태의 시각효과
    - submit: formmethod(POST, GET), formaction(전송할 URL), formenctype(인코딩 방식), formnovalidate 속성이 추가
    - file: 파일을 여러 개 선택할 수 있게 되었으며(multiple 속성) accept 속성에 MIME 타입을 지정해서 선택할 수 있는 파일의 종류를 제한 할 수 있게 되었으며 files 속성을 이용해서 파일의 정보를 가져올 수 있도록 변경

FORM

- FORM
  - ✓ 추가된 속성
    - autocomplete 속성을 이용해서 자동 완성 기능 사용 여부 설정 – on 과 off 로 설정
    - placeholder 속성을 이용해서 보충 설명 추가 가능
    - autofocus 속성을 이용해서 첫번째 포커스 설정
    - list 속성과 datalist 태그를 이용해서 입력 값을 선택할 수 있도록 도와줌

```
<input type="tel" list="telephonelist">
<datalist id="telephonelist">
        <option value="010-3790-1997">
        <option value="010-3139-1997">
        <option value="010-3138-1997">
</datalist>
```

FORM

- FORM
  - ✓ 유효성 검사
    - 폼 요소의 입력 값 체크를 할 수 있음
    - 검증을 위한 속성
      - required : 필수 입력
      - pattern : 입력 값의 패턴을 정규 표현식을 지정해서 검증 가능 – 최신 버전의 chrome에서 동작하지 않음
      - maxlength: 최대 입력 글자 수
      - min, max
      - 위의 속성을 부여하면 입력 값의 검증이 송신할 때 자동으로 이루어지며 검증에 통과하지 못하면 폼을 전송하지 않는데 novalidate 속성을 이용해서 유효성 검사를 수행하지 않도록 하는 것이 가능

FORM

- FORM


127.0.0.1:8020/0311/
Google
Edit
Post to Blog
전화번호:
신용카드
이 항목을 필수로 입력해 주십시오.
가입


127.0.0.1:8020/0311/
Google
Edit
Post to Blog
전화번호: 01037901997
신용카드: 1234
가입
요청한 형식으로 입력해 주십시오. : 신용카드 번호는 숫자 16자리.

FORM

- FORM
  - ✓ validation.html

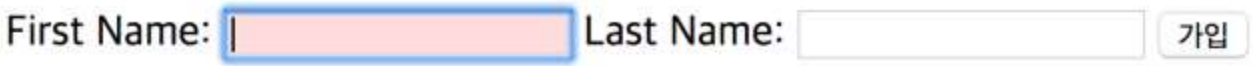
```
<!DOCTYPE html>
<html>
<head>
<title>validation test</title>
<meta charset="UTF-8">
</head>
<body>
    <form action="server">
      <p>
            <label>전화번호:</label> <input id="phone" type="tel"
                placeholder="(xxx) xxxx-xxx" required>
      </p>
      <p>
            <label>신용카드:</label> <input id="card"
            pattern="[0-9]{16}" required title="신용카드 번호는 숫자 16자리">
      </p>
      <p>
            <input type="submit" name="register" value="가입">
      </p>
    </form>
</body>
</html>
```

FORM

- FORM
  - ✓ 유효성 검사
    - setCustomValidity(메시지) 함수를 이용해서 유효성 검사를 통과하지 못했을 때의 메시지를 직접 설정 가능
    - form의 데이터가 유효성 검사를 통과하지 못하면 form에는 invalid 이벤트가 발생

FORM

- FORM
  - ✓ 유효성 검사



FORM

- FORM
  - ✓ html5validation.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Forms</title>
  <meta charset="utf-8" />
  <script>
    function initiate() {
      name1 = document.getElementById("firstname");
      name2 = document.getElementById("lastname");
      name1.addEventListener("input", validation, false);
      name2.addEventListener("input", validation, false);
      validation();
    }
```

FORM

- FORM
  - ✓ html5validation.html

```
        function validation() {
          if (name1.value == '' && name2.value == '') {
            name1.setCustomValidity('적어도 하나는 입력해야 합니다.');
            name1.style.background = '#FFDDDD';
          } else {
            name1.setCustomValidity('');
            name1.style.background = '#FFFFFF';
          }
        }
        window.addEventListener("load", initiate, false);
      </script>
    </head>
```

FORM

- FORM
  - ✓ html5validation.html

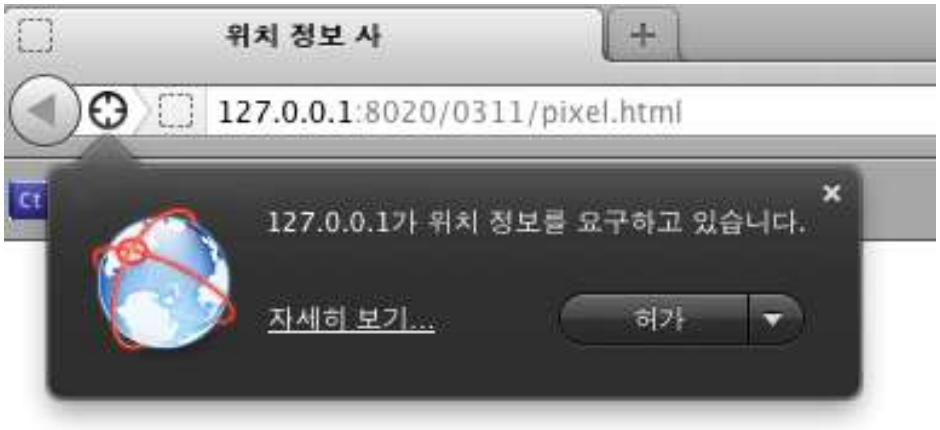
```
<body>
  <section id="form">
    <form name="registration" method="get">
      First Name:
      <input type="text" name="firstname" id="firstname">
      Last Name:
      <input type="text" name="lastname" id="lastname">
      <input type="submit" id="send" value="가입">
    </form>
  </section>
</body>

</html>
```

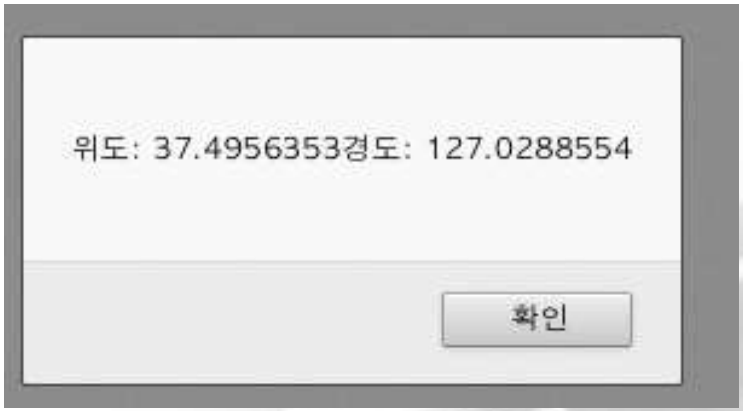
Geo Location

- Geo Location
  - ✓ 디바이스의 물리적 위치 정보를 파악하기 위한 JavaScript API
  - ✓ 위치 정보 사용 가능 여부: window.navigator.geolocation 의 값을 조사
  - ✓ 현재 위치 한번만 받아오기: navigator.geolocation.getCurrentPosition(위치정보를 가져오는데 성공했을 때 호출되는 메서드, 위치정보를 가져오는데 실패했을 때 호출되는 메서드, 위치 정보를 얻을 때의 옵션)
    - 위치 정보를 가져오는데 성공했을 때 호출되는 메서드에게는 객체가 1개 넘어가게 되는데 이 객체는 위치 정보 객체로 coords 객체를 소유하고 있으며 coords 안에 latitude, longitude, altitude, accuracy, altitudeAccuracy, heading, speed를 프로퍼티로 소유하고 있고 위치 정보를 가져온 시간을 저장하고 있는 timestamp 속성을 소유
    - 위치 정보를 가져오는데 실패했을 때 호출되는 메서드에게는 객체가 1개 넘어가게 되는데 이 객체는 code(UNKNOWN_ERROR, PERMISSION_DENIED, POSITION_UNAVALABLE, TIMEOUT)와 message를 소유

Geo Location

- Geo Location





Geo Location

- Geo Location
  - ✓ currentposition.html

```
<!DOCTYPE html>
<html>

<head>
    <title>위치 정보 사용</title>
    <meta charset="UTF-8">
</head>

<body>
</body>
<script>
    navigator.geolocation.getCurrentPosition(function (position) {
      alert("위도: " + position.coords.latitude + "경도: " +
            position.coords.longitude);
    }, function (error) {
      alert(error.message);
    });
</script>

</html>
```

Geo Location

- ✓ Geo Location
  - ✓ 위치 정보를 계속 확인하기 위해서는 watchPosition(성공했을 때 호출될 메서드, 실패했을 때 호출될 메서드, 옵션) 메서드를 이용
    - clearWatch(watchPosition 메서드가 반환한 값)를 호출할 때 까지 조사
    - 옵션
      - enableHighAccuracy: 정확도가 높은 위치 정보를 요청하도록 하는 옵션인데 기본 값은 false
      - timeout: 시간 제한을 설정
      - maximumAge: 위치 정보의 유효 기간을 설정(밀리 세컨드 단위)하는 속성으로 0을 지정하면 항상 새로운 위치 정보를 가져옴

Geo Location

- Geo Location
  - ✓ watchposition.html

```
<!DOCTYPE html>
<html>
<head>
    <title>위치 정보 사용</title>
    <meta charset="UTF-8">
    <script>
     navigator.geolocation.watchPosition(onSuccess, onError, {
            maximumAge: 5000,
            enableHighAccuracy: true
     });
     function onSuccess(position) {
            alert("위도: " + position.coords.latitude + "₩n경도: " +
    position.coords.longitude +
                "₩n방향: " + position.coords.heading);
     }
     function onError(error) {
            alert(error.message);
     }
    </script>
</head>
<body></body>
</html>
```

Geo Location

- Geo Location
  - ✓ 카카오 지도 활용

Geo Location

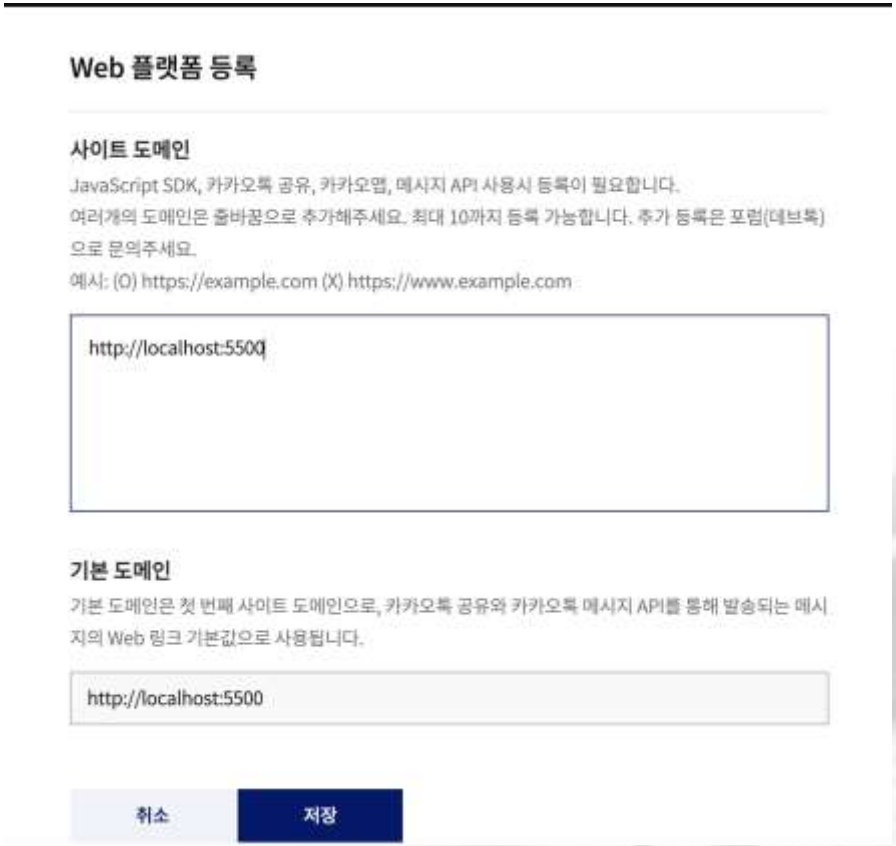
- Geo Location
  - ✓ 카카오 지도 활용
    - https://developers.kakao.com/ 에 로그인해서 애플리케이션을 생성하고 플랫폼(Web)을 설정해서 키를 받아야 함

Geo Location

- Geo Location
  - ✓ 카카오 지도 활용
    - https://developers.kakao.com/ 에 로그인해서 애플리케이션을 생성하고 플랫폼(Web)을 설정해서 키를 받아야 함

Geo Location

- ❖ Geo Location
  - ✓ 카카오 지도 활용
    - https://developers.kakao.com/ 에 로그인해서 애플리케이션을 생성하고 플랫폼(Web)을 설정해서 키를 받아야 함



Geo Location

- Geo Location
  - ✓ 카카오 지도 활용
    - https://developers.kakao.com/ 에 로그인해서 애플리케이션을 생성하고 플랫폼(Web)을 설정해서 키를 받아야 함 - Javascript

Geo Location

- ❖ Geo Location
  - ✓ 카카오 지도 활용
    - kakaomap.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>위치 정보 사용</title>
</head>

<body>
    <div id="map" style="width:500px;height:400px;"></div>
    <script type="text/javascript"
    src="//dapi.kakao.com/v2/maps/sdk.js?appkey=767bd1c0fb046671d16fc3574f70
    5521">
    </script>
</body>
```

Geo Location

- Geo Location
  - ✓ 카카오 지도 활용
    - kakaomap.html

```
<script>
    navigator.geolocation.getCurrentPosition(
     function (location) {
            var container = document.getElementById('map');
            var options = {
                center: new kakao.maps.LatLng(location.coords.latitude,
                    location.coords.longitude),
                level: 3
            };

            var map = new kakao.maps.Map(container, options);

            // 마커가 표시될 위치입니다
            var markerPosition = new kakao.maps.LatLng(location.coords.latitude,
    location.coords.longitude);

            // 마커를 생성합니다
            var marker = new kakao.maps.Marker({
                position: markerPosition
            });
```

Geo Location

- Geo Location
  - ✓ 카카오 지도 활용
    - kakaomap.html

```
                    // 마커가 지도 위에 표시되도록 설정합니다
                    marker.setMap(map);

                    // 지도에 교통정보를 표시하도록 지도타입을 추가합니다
                    map.addOverlayMapTypeId(kakao.maps.MapTypeId.TRAFFIC);

            },
            function (error) {
                    alert(error.code)
            }
        );
</script>

</html>
```

File API

- File API
  - ✓ HTML5의 로컬 파일에 대한 접근을 위한 API
  - ✓ input 태그 중에서 type 속성을 file로 설정하면 로컬 파일을 선택하도록 할 수 있는데 이 때 files 프로퍼티를 이용해서 파일을 전송
  - ✓ files 프로퍼티는 선택된 모든 파일이 포함되어 있는 배열
  - ✓ input 태그의 속성 중 multiple 속성이 없으면 하나의 파일만 선택 가능
  - ✓ 배열의 요소는 File 객체인데 아래와 같은 프로퍼티를 소유
    - name: 파일의 전체 이름
    - size: 파일의 크기를 바이트 단위로 리턴
    - type: MIME 타입으로 파일의 타입을 리턴

File API

- File API
  - ✓ FileReader: 파일을 읽기 위한 인터페이스

Attribute/Method	설명
readAsBinaryString(file)	File 내용을 읽어 Binary 문자열로 리턴 바이트 배열로 리턴
readAsText(file, encoding)	File 내용을 읽어 들여 문자열로 리턴 두번째 인수는 File의 문자 encoding
readAsDataURL(file)	File 내용을 읽어 dataURL 형식의 문자열로 저장
result	읽어들인 File 내용
error	error 발생시의 error 정보
load	읽기에 성공 했을 때 호출하는 event
progress	읽는 진행 상황을 얻을 수있는 event
error	읽기에 실패했을 때 호출되는 event
abort	읽기를 중단했을 때 호출되는 event
loadend	읽기에 성공하거나 실패했을 때 호출되는 event

File API

- File API
  - ✓ Error

상수	Code	설명
NOT_FOUND_ERR	1	읽을 File 을 찾지 못 할때
SECURITY_ERR	2	Web Application 이 Access 하기에 안전하지 못한 File일 때, File 에 너무 많은 읽기 호출이 있을 때, 사용자의 선택한 이후에 File 에 변경이 있을 때 등의 상황에서 발생
ABORT_ERR	3	abort() 함수 호출과 같은 것으로 인해 읽기가 중지되었을 때
NOT_READABLE_ERR	4	File 접근 권한 문제와 같은 것으로 인해 File 을 읽지 못할 때
ENCODING_ERR	5	동기적, 비동기적으로 readAsText() 함수를 사용하는 경우에 사용할 수 없을 때나 DataURL로 표현될 수 있는 File이나 Blob를 구현한 제한된 곳의 DataURL에 대한 URL 길이 제한에 걸렸을 때

File API

- File API
  - ✓ 선택한 파일 정보 확인

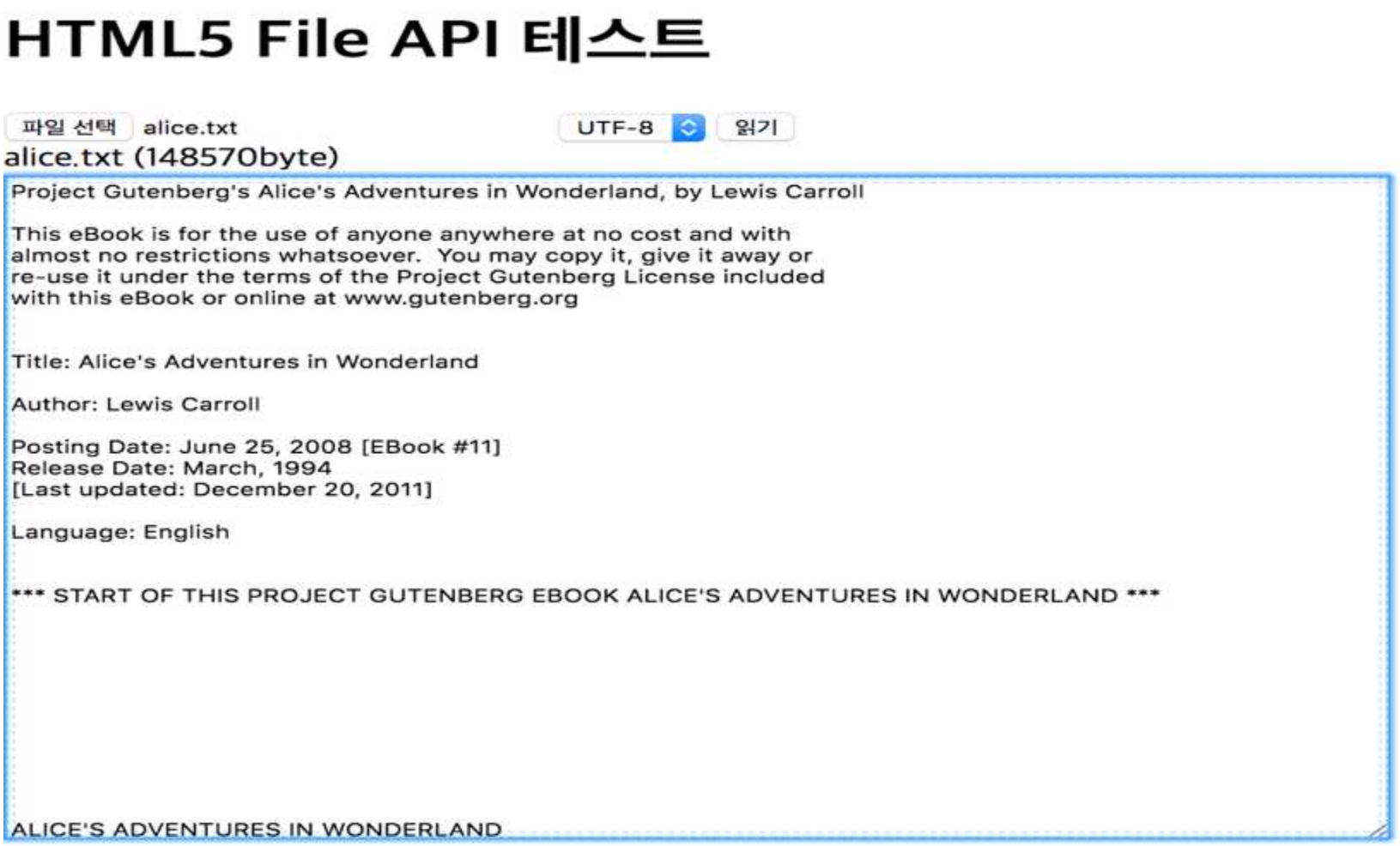
File API

- File API
  - ✓ file.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>선택한 파일 정보 확인</title>
</head>
<body>
    <form>
      <input type="file" id="file" multiple>
    </form>
</body>
<script>
    document.getElementById("file").addEventListener("change", function showfileinfo() {
      var selectFiles = document.getElementById("file").files;
      var fileAr = [];
      for (var i = 0; i < selectFiles.length; i++) {
            var file = selectFiles[i];
            fileAr.push("파일명:" + file.name + " 크기:" + file.size);
      }
      alert(fileAr.join("₩n"));
    });
</script>
</html>
```

File API

- File API
  - ✓ 텍스트 파일 읽기



File API

- File API
  - ✓ txtfileread.html

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML5 File API Test</title>
    <meta charset="utf-8" />
</head>
<body>
    <h1>HTML5 File API 테스트</h1>
    <input id="file" type="file">
    <select id="encoding">
      <option value="UTF-8">UTF-8</option>
      <option value="EUC-KR">EUC-KR</option>
    </select>
    <button onclick="readFile()">읽기</button>
    <br />
    <div>
      <span id="fileName">File Name</span>
      <span id="fileSize">File
            Size</span>
    </div>
    <textarea id="content" readonly style="width: 600px; height: 400px;"></textarea>
</body>
```

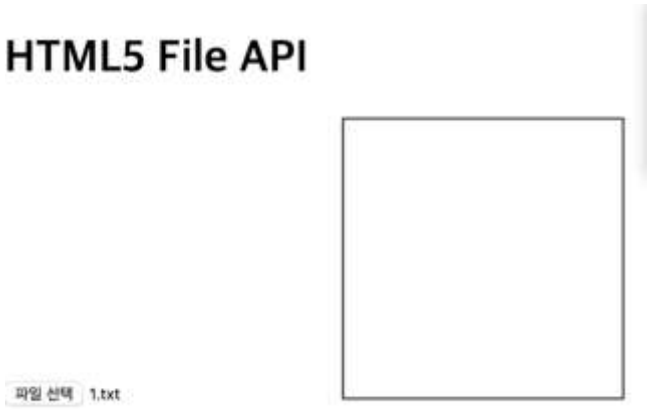
File API

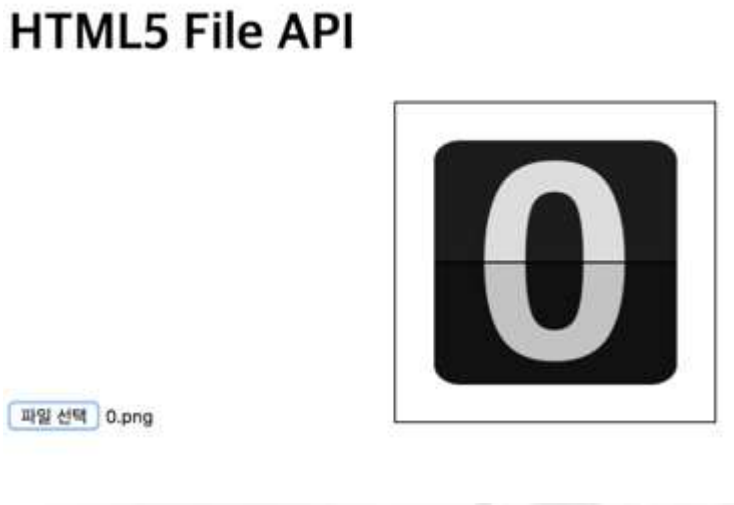
- File API
  - ✓ txtfileread.html

```
<script>
    function readFile() {
      var file = document.getElementById("file").files[0];
      document.getElementById("fileName").textContent = file.name;
      document.getElementById("fileSize").textContent = "(" + file.size +
            "byte)";
      var reader = new FileReader();
      reader.onload = function () {
            var display = document.getElementById("content");
            display.textContent = reader.result;
      };
      reader.onerror = function (evt) {
            alert(evt.target.error.code);
      };

      var encodingList = document.getElementById("encoding");
      var encoding = encodingList.options[encodingList.selectedIndex].value;
      reader.readAsText(file, encoding);
    };
</script>
</html>
```

File API

- File API
  - ✓ 이미지 미리보기





File API

- File API
  - ✓ imagepreview.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>이미지 미리보기</title>
</head>

<body>
    <h1>HTML5 File API</h1>
    <input type="file" id="imgInp" accept=".png, .jpg, .jpeg, .gif" />
    <!-- 위에서 선택한 파일이 이미지 일 때 미리보기 영역 -->
    <img id="img" width="200" height="200" border="1" />
</body>
```

File API

- File API
  - ✓ imagepreview.html

```
<script>
    var imgInp = document.getElementById("imgInp")
    var img = document.getElementById("img")

    //파일의 선택이 변경될 때 수행할 내용
    imgInp.addEventListener("change", function (e) {
      //파일이 선택된 경우만 수행
      if (imgInp.files && imgInp.files[0]) {
            //파일이름 가져오기
            var filename = imgInp.files[0].name

            //.뒤의 문자열 가져오기
            //split 메소드 이용
            var ar = filename.split(".")
            var ext = ar[ar.length - 1]
```

File API

- File API
  - ✓ imagepreview.html

```
<script>
        //그림 파일인지 확인 - 확장자:jpg, gif, png
        if (ext.toLowerCase() != "jpg" &&
            ext.toLowerCase() != "gif" &&
            ext.toLowerCase() != "png") {

            alert("그림 파일을 선택하세요 !!!!")
            return
        }

        //파일의 내용을 읽기 위한 객체 만들기
        var reader = new FileReader()
        reader.readAsDataURL(imgInp.files[0])

        //파일의 내용을 전부 읽으면
        reader.addEventListener("load", function (e) {
            img.src = reader.result;
        })
    }
  });
</script>
</html>
```

Drag & Drop API

- Drag & Drop API
  - ✓ 서로 다른 특정한 상황을 위해 도입된 7개의 이벤트 집합
  - ✓ 소스에서 발생하는 이벤트도 있고 타켓에서 발생하는 이벤트도 있음
  - ✓ 소스에서 발생하는 이벤트
    - dragstart
    - drag
    - dragend
  - ✓ 타켓에서 발생하는 이벤트
    - dragenter
    - dragover
    - drop
    - dropleave
  - ✓ dataTransfer
    - 드래그 앤 드랍 작업에서 데이터를 저장하는 객체
    - setData(type, data): 데이터를 전송하기 위해서 호출하는 메서드로 type에는 text/plain, text/html, text/uri-list, URL 또는 Text를 설정
    - getData(type): type 에 해당하는 데이터를 리턴
    - clearData(): 데이터 삭제

Drag & Drop API

- Drag & Drop API

Drag & Drop API

- Drag & Drop API
  - ✓ draganddrop.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Drag and Drop</title>
  <style>
    #dropbox {
      float: left;
      width: 500px;
      height: 300px;
      margin: 10px;
      border: 1px solid #999999;
    }
    #picturesbox {
      float: left;
      width: 320px;
      margin: 10px;
      border: 1px solid #999999;
    }
    #picturesbox>img {
      float: left;
      padding: 5px;
    }
  </style>
```

Drag & Drop API

- Drag & Drop API
  - ✓ draganddrop.html

```
<script>
  function initiate() {
    source1 = document.getElementById('image');
    source1.addEventListener('dragstart', dragged, false);

    drop = document.getElementById('dropbox');
    drop.addEventListener('dragenter', function (e) {
      e.preventDefault();
    }, false);
    drop.addEventListener('dragover', function (e) {
      e.preventDefault();
    }, false);
    drop.addEventListener('drop', dropped, false);
  }

  function dragged(e) {
    var code = '<img src="' + source1.getAttribute('src') + '">';
    e.dataTransfer.setData('Text', code);
  }
```

Drag & Drop API

- Drag & Drop API
  - ✓ draganddrop.html

```
        function dropped(e) {
          e.preventDefault();
          drop.innerHTML = e.dataTransfer.getData('Text');
        }
        window.addEventListener('load', initiate, false);
      </script>
    </head>
```

Drag & Drop API

- Drag & Drop API
  - ✓ draganddrop.html

```
<body>
  <section id="dropbox">
    Drag and drop the image here
  </section>
  <section id="picturesbox">
    <img id="image" src="images/network.png">
  </section>
</body>

</html>
```

Drag & Drop API

- Drag & Drop API

Drag & Drop API

- Drag & Drop API
  - ✓ dragenter.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>Drag and Drop</title>
  <style>
    #dropbox {
      float: left;
      width: 500px;
      height: 300px;
      margin: 10px;
      border: 1px solid #999999;
    }
```

Drag & Drop API

- Drag & Drop API
  - ✓ dragenter.html

```
#picturesbox {
  float: left;
  width: 320px;
  margin: 10px;
  border: 1px solid #999999;
}

#picturesbox>img {
  float: left;
  padding: 5px;
}
</style>
```

Drag & Drop API

- Drag & Drop API
  - ✓ dragenter.html

```
<script>
  function initiate() {
    source1 = document.getElementById('image');
    source1.addEventListener('dragstart', dragged, false);
    source1.addEventListener('dragend', ending, false);

    drop = document.getElementById('dropbox');
    drop.addEventListener('dragenter', entering, false);
    drop.addEventListener('dragleave', leaving, false);
    drop.addEventListener('dragover', function (e) {
      e.preventDefault();
    }, false);
    drop.addEventListener('drop', dropped, false);
  }
```

Drag & Drop API

- Drag & Drop API
  - ✓ dragenter.html

```
function entering(e) {
  e.preventDefault();
  drop.style.background = 'rgba(0,150,0,.2)';
}

function leaving(e) {
  e.preventDefault();
  drop.style.background = '#FFFFFF';
}

function ending(e) {
  elem = e.target;
  elem.style.visibility = 'hidden';
}
```

Drag & Drop API

- Drag & Drop API
  - ✓ dragenter.html

```
        function dragged(e) {
          var code = '<img src="' + source1.getAttribute('src') + '">';
          e.dataTransfer.setData('Text', code);
        }

        function dropped(e) {
          e.preventDefault();
          drop.style.background = '#FFFFFF';
          drop.innerHTML = e.dataTransfer.getData('Text');
        }
        window.addEventListener('load', initiate, false);
      </script>
    </head>
```

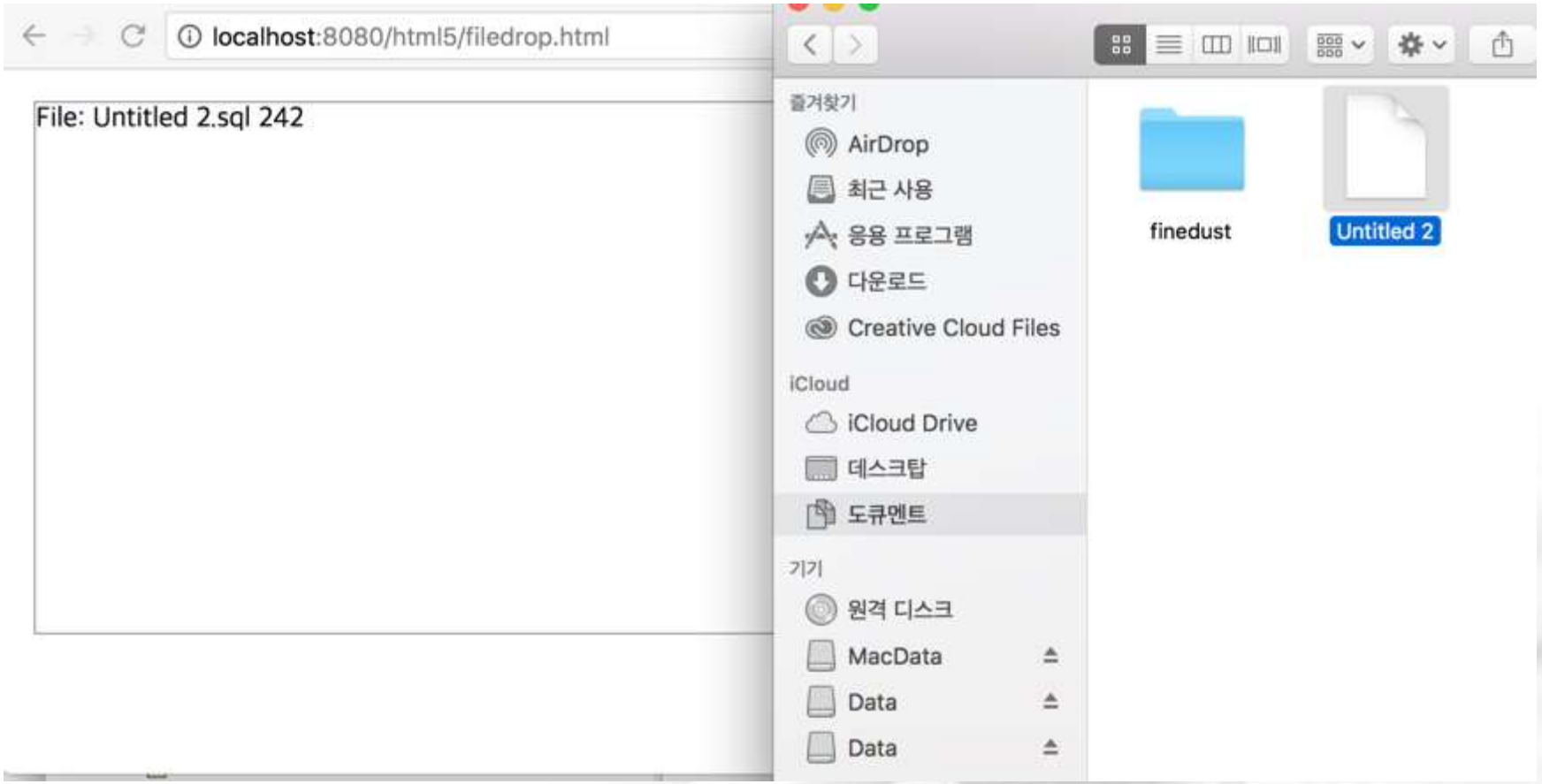
Drag & Drop API

- Drag & Drop API
  - ✓ dragenter.html

```
<body>
  <section id="dropbox">
    Drag and drop the image here
  </section>
  <section id="picturesbox">
    <img id="image" src="images/network.png">
  </section>
</body>

</html>
```

Drag & Drop API

❖ Drag & Drop API



Drag & Drop API

- Drag & Drop API
  - ✓ filedrop.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>Drag and Drop</title>
  <style>
    #dropbox {
      float: left;
      width: 500px;
      height: 300px;
      margin: 10px;
      border: 1px solid #999999;
    }
```

Drag & Drop API

- Drag & Drop API
  - ✓ filedrop.html

```
#picturesbox {
  float: left;
  width: 320px;
  margin: 10px;
  border: 1px solid #999999;
}

#picturesbox>img {
  float: left;
  padding: 5px;
}
</style>
```

Drag & Drop API

- Drag & Drop API
  - ✓ filedrop.html

```
<script>
  function initiate() {
    drop = document.getElementById('dropbox');
    drop.addEventListener('dragenter', function (e) {
      e.preventDefault();
    }, false);
    drop.addEventListener('dragover', function (e) {
      e.preventDefault();
    }, false);
    drop.addEventListener('drop', dropped, false);
  }
```

Drag & Drop API

- Drag & Drop API
  - ✓ filedrop.html

```
        function dropped(e) {
          e.preventDefault();
          var files = e.dataTransfer.files;
          var list = '';
          for (var f = 0; f < files.length; f++) {
            list += 'File: ' + files[f].name + ' ' + files[f].size + '<br>';
          }
          drop.innerHTML = list;
        }
        window.addEventListener('load', initiate, false);
      </script>
    </head>

    <body>
      <section id="dropbox">
        Drag and drop FILES here
      </section>
    </body>

    </html>
```

로컬 저장소

- Web Storage
  - ✓ 클라이언트의 디스크에 소량의 데이터를 저장하기 위한 저장소
  - ✓ 이전에는 일반적으로 Cookie를 사용
  - ✓ Cookie 와 다른 점
    - 크기에 제한이 없음
    - 데이터를 서버로 보내지 않음
    - 자바스크립트 객체의 복사본을 저장할 수 있음
  - ✓ 종류
    - 로컬 저장소
    - 세션 저장소
  - ✓ 메서드 및 속성
    - length 속성: 저장소에 저장된 데이터의 수를 반환
    - key(index): 인덱스의 키를 반환하고 없으면 null을 반환
    - getItem(key): 키에 대응되는 데이터를 반환(.key 로 가져오기 가능)
    - setItem(key, data): key로 저장소에 데이터를 저장(.key=데이터 형태로 설정 가능)
    - removeItem(key): key에 대응되는 데이터를 삭제
    - clear(): 저장소의 모든 데이터 삭제

로컬 저장소

- Web Storage
  - ✓ 저장
    - storage.키이름 = {속성:값, 속성:값....};
    - storage["키이름"] = {속성:값, 속성:값...};
    - storage.setItem("키이름", {속성:값, 속성:값...});
  - ✓ 읽기
    - localstorage.키이름;
    - localstorage["키이름"];
    - localstorage.getItem("키이름");
  - ✓ 삭제
    - delete storage.키이름;
    - delete storage["키이름"];
    - storage.removeItem("키이름");

로컬 저장소

- Web Storage
  - ✓ 이벤트 처리
    - 저장소에 변경이 발생하면 window에 storage 이벤트가 발생
    - 이벤트 객체 속성
      - key
      - oldValue
      - newValue
      - url
      - storageArea

로컬 저장소

- Web Storage
  - ✓ Local Storage
    - 웹 브라우저마다 별도로 생성되는 저장 영역 – 브라우저를 종료해도 유지
    - 명시적으로 지우지 않는 이상 데이터는 영구적으로 저장
    - 전역변수 localStorage를 이용
  - ✓ Session Storage
    - 접속한 브라우저 와 동일한 유효 범위, 생존 기간을 가지며 도메인 마다 따로 생성되는 스토리지 – 브라우저를 종료하면 소멸
    - 현재 창에서 새로은 창으로 다른 URL을 여는 경우 이전 세션 스토리지의 내용을 복사해서 세션 스토리지가 생성
    - 웹 사이트에서 사용자의 동작을 추적하고자 하는 경우 많이 사용

로컬 저장소

❖ Web Storage

로컬 저장소

- Web Storage
  - ✓ sessionstorage.html

```
<!DOCTYPE html>
<html>
<head>
    <title>sessiongStorage 테스트</title>
    <meta charset="UTF-8">
</head>
<body onload="init()">
    <fieldset>
      <legend>세션 스토리지 이용</legend>
      key: <input type="text" id="key" />
      value: <input type="text" id="value" />
      <input type="button" value="삽입" onclick="addData()" />
      <input type="button" value="삭제" onclick="removeData()" />
      <input type="button" value="윈도우 추가" onclick="window.open(location.href);" />
    </fieldset>

    <fieldset>
      <legend>세션 스토리지 데이터</legend>
      <select id="list" size=10 onchange="onListSelected()">
      </select> <input type="button" value="재출력" onclick="showData()" />
    </fieldset>
</body>
```

로컬 저장소

- Web Storage
  - ✓ sessionstorage.html

```
<script>
    function init() {
      var key = document.getElementById("key");
      var value = document.getElementById("value");
      var list = document.getElementById("list");
      showData();
    }

    function showData() {
      list.innerHTML = "";
      for (var i = 0; i < sessionStorage.length; i++) {
            var temp = sessionStorage.key(i);
            list.options[list.options.length] =
                  new Option(temp + ":" + sessionStorage[temp], temp);
      }
    }

    function addData() {
      sessionStorage[key.value] = value.value;
      showData();
    }
```

로컬 저장소

- Web Storage
  - ✓ sessionstorage.html

```
        function removeData() {
          delete sessionStorage[key.value];
          showData();
        }

        function onListSelected() {
          var selectData = list.options[list.selectedIndex];
          key.value = selectData.value;
          value.value = sessionStorage[selectData.value];
        }
    </script>

    </html>
```

로컬 저장소

❖ Web Storage

로컬 저장소

- Web Storage
  - ✓ idsave.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>웹 스토리지 사용</title>
</head>

<body>
    <form action="login.jsp" id="loginform">
      아이디:<input type="text" id="id" required="required" /><br />
      <input type="checkbox" value="check" id="idsave"> 아이디 저장<br />
      <input type="submit" value="로그인" />
    </form>
</body>
```

로컬 저장소

- Web Storage
  - ✓ idsave.html

```
<script>
    var idsave = document.getElementById("idsave");
    var ids = document.getElementById("id");

    //윈도우가 로드를 끝냈을 때
    window.addEventListener("load", function (e) {
      var event = e || window.event
      if (typeof localStorage.ids != 'undefined') {
            ids.value = localStorage.ids;
            idsave.checked = true;
      }
    });
```

로컬 저장소

- Web Storage
  - ✓ idsave.html

```
        document.getElementById("loginform").addEventListener(
          "submit",
          function (e) {
                var event = e || window.event
                //체크박스의 값 확인
                if (idsave.checked == true) {
                      //아이디에 입력된 값 저장
                      localStorage.ids = ids.value;
                } else {
                      //로컬 스토리지에 저장된 데이터 삭제
                      localStorage.clear();
                }
          })
</script>

</html>
```

로컬 저장소

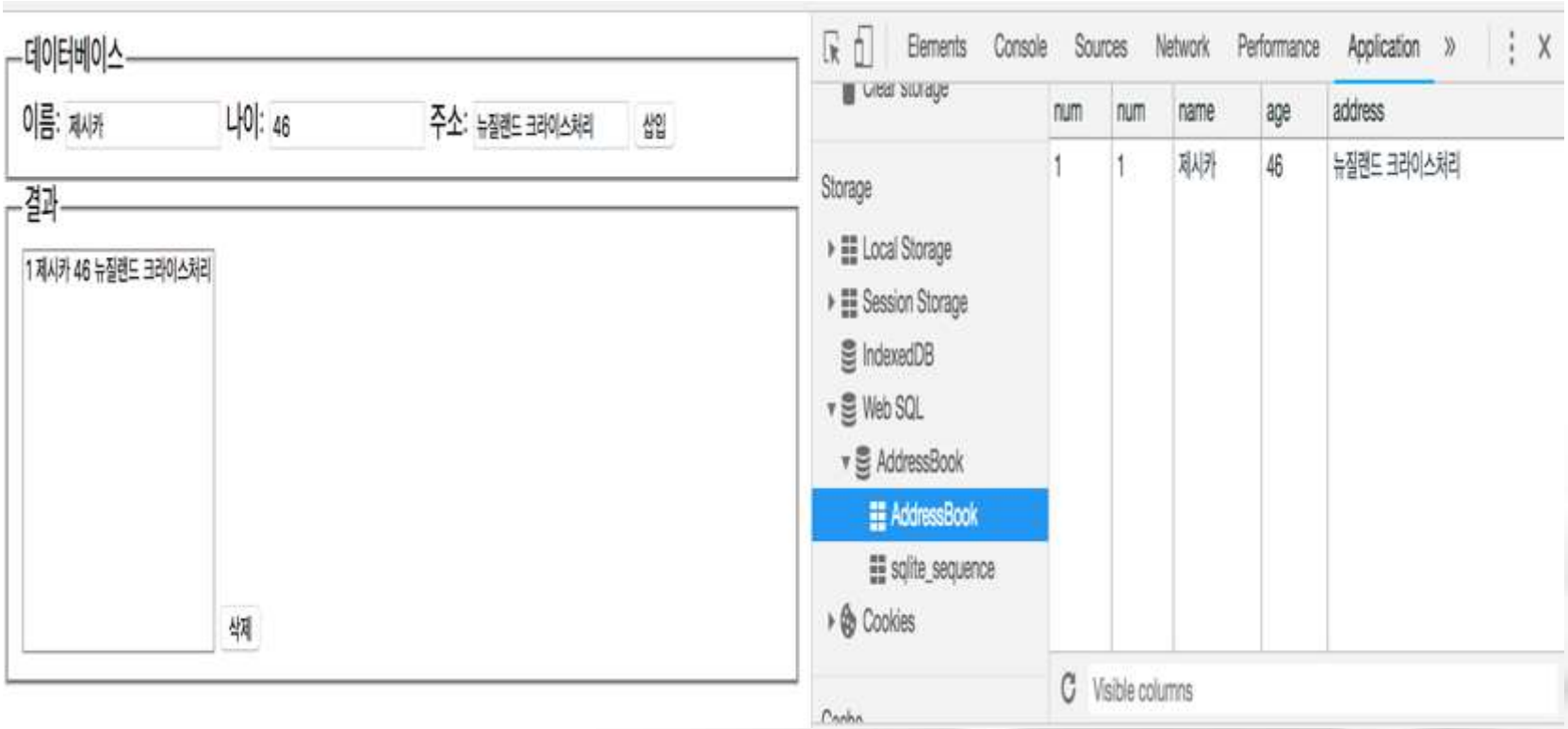
- Web SQL
  - ✓ 브라우저에서 사용할 수 있는 관계형 데이터베이스
  - ✓ 작업 순서
    - 데이터베이스 열기 → 트랜잭션 시작 → sql 실행
  - ✓ 데이터베이스 열기: openDatabase(데이터베이스 이름, 버전, 사용자에게 보여질 데이터베이스 이름, 데이터베이스의 크기, 처리가 완료된 경우 호출 메서드)를 호출해서 반환된 데이터베이스 객체를 이용해서 작업 수행

로컬 저장소

- Web SQL
  - ✓ 데이터베이스 관련 메서드
    - transaction(콜백 메서드, 에러가 발생했을 때 호출할 메서드, 성공했을 때 호출할 메서드): 데이터베이스 객체를 이용해서 호출하는데 콜백 메서드에는 SQL 작업을 위한 트랜잭션 객체가 전달되고 에러가 발생했을 때 호출될 메서드에는 에러 객체가 전달됨
    - transaction 메서드 호출 할 때 콜백 메서드 내부에서 sql 구문 수행
      - 트랜잭션변수.executeSql("sql 구문");
      - 트랜잭션변수.executeSql("sql 구문 내의 ?",[?의 실제 데이터]);
    - 결과 사용
      - sql 구문을 수행하게 되면 SQLResultSet 객체를 반환
      - 이 객체는 insertId(추가 시 행 번호), rowAffected(영향 받은 행의 개수), rows(select 구문 수행 시 결과)를 리턴

로컬 저장소

- Web SQL



로컬 저장소

- Web SQL
  - ✓ database.html

```
<!DOCTYPE html>
<html>

<head>
    <title>데이터베이스 테스트</title>
    <meta charset="UTF-8">
</head>
```

로컬 저장소

- Web SQL
  - ✓ database.html

```
<!DOCTYPE html>
<html>

<head>
    <title>데이터베이스 테스트</title>
    <meta charset="UTF-8">
</head>

<body>
    <fieldset>
      <legend>데이터베이스</legend>
      이름: <input type="text" id="name" /> 나이: <input type="text" id="age" />
      주소: <input type="text" id="address" /> <input type="button" value="삽입"
    onclick="addData()" />
    </fieldset>

    <fieldset>
      <legend>결과</legend>
      <select id="list" size=10>
      </select> <input type="button" value="삭제" onclick="removeData()" />
    </fieldset>
</body>
```

로컬 저장소

- Web SQL
  - ✓ database.html

```
<script>
    var db;

    function init() {
      db = window.openDatabase("AddressBook", "1", "AddressBookDB",
            1024 * 1024);
      db.transaction(function (tx) {
            tx.executeSql("create table if not exists AddressBook(" +
                  "num integer primary key autoincrement," +
                  " name, age integer, address)");
      });
      showData();
    }
```

로컬 저장소

- Web SQL
  - ✓ database.html

```
function showData() {
  db.transaction(function (tx) {
      tx.executeSql("select * from AddressBook", [], function (tx, rs) {
          var list = document.getElementById("list");
          list.innerHTML = "";
          var rows = rs.rows;

          for (var i = 0; i < rows.length; i++) {
              var row = rows.item(i);
              var option = new Option(row.num + "₩t" + row.name + "₩t" +
                  row.age + "₩t" + row.address, row.num);
              list.options[list.options.length] = option;
          }
      });
  });
}
```

로컬 저장소

- Web SQL
  - ✓ database.html

```
function addData() {
  var name = document.getElementById("name").value;
  var age = document.getElementById("age").value;
  var address = document.getElementById("address").value;
  db
      .transaction(function (tx) {
          tx.executeSql(
                  "insert into AddressBook(name, age, address) values(?,?,?)",
                  [name, age, address],
                  function (tx, rs) {
                      alert("레코드가 저장되었습니다");
                      showData();
                  });
      });
}
```

로컬 저장소

- Web SQL
  - ✓ database.html

```
        function removeData() {
          var list = document.getElementById("list");
          if (list.selectedIndex < 0) {
                return;
          }
          var selected = list.options[list.selectedIndex].value;
          db.transaction(function (tx) {
                tx.executeSql("delete from AddressBook where num=?", [selected],
                      function () {
                            showData();
                      });
          });
        }

        window.addEventListener("load", init());
    </script>

    </html>
```

로컬 저장소

- Indexed DB
  - ✓ 개요
    - 사용자의 브라우저 안에 데이터를 영구적으로 저장하게 해주는 방법 중 하나
    - IndexedDB는 "키(key)"로 지정된 객체를 저장하고 검색
    - IndexedDB는 동일 출처 정책 (same-origin policy)을 따르기 때문에 한 도메인의 데이터에 접근하는 동안 다른 도메인의 데이터에 접근할 수 없음
    - IndexedDB는 웹 워커를 포함하는 대부분의 문맥에 사용될 수 있는 비동기(asynchronous) API
    - IndexedDB와 WebSQL 둘 모두 데이터 저장을 위한 솔루션이지만 동일한 기능을 제공하지는 않는데 WebSQL Database는 관계형 데이터베이스 접근 시스템이고 IndexedDB는 인덱스 테이블 시스템
  - ✓ 특징
    - IndexedDB 데이터베이스는 키(key)-값(value) 한 쌍을 저장
    - IndexedDB는 트랜잭션 데이터베이스 모델을 내장
    - IndexedDB API는 비동기 방식
    - IndexedDB는 많은 request를 사용
    - IndexedDB는 결과를 사용 가능하게 되었을 때 알리기 위해 DOM 이벤트를 사용
    - IndexedDB는 객체 지향
    - IndexedDB는 구조화 질의어(SQL)를 사용하지 않음

로컬 저장소

- Indexed DB
  - ✓ indexedDB 속성의 open 메서드를 이용해서 데이터베이스를 생성
  - ✓ 객체 저장소 관리를 위한 API
    - createObjectStore(name, keyPath, autoIncrement): 이름과 설정 속성으로 새로운 객체 저장소를 생성하는데 keyPath는 모든 객체에 대한 공통 인덱스를 선언하는 옵션이고 autoIncrement 속성은 Boolean 값으로 keyPath에 해당하는 인덱스의 자동 증가 여부를 결정
    - objectStore(name): 객체 저장소에 있는 객체를 접근하려면 트랜잭션이 시작되어야 하고 객체 저장소는 이 트랜잭션에 대해 열려 있어야 하는데 이 메서드가 객체 저장소를 열어 줌
    - deleteObjectStore(name): 객체 저장소를 삭제
  - ✓ 인덱스 관련 API
    - createIndex(name, property, unique): 특정 객체에 대한 인덱스를 생성
    - index(name): 인덱스를 사용하려면 먼저 인덱스에 대한 참조를 생성한 다음 이 참조를 트랜잭션에 할당해야 하는데 이 메서드는 name 속성으로 선언된 인덱스의 참조를 생성
    - deleteIndex(name): name에 해당하는 인덱스 삭제

로컬 저장소

- Indexed DB
  - ✓ 트랜잭션
    - 브라우저에서 동작하는 데이터베이스 시스템은 다른 플랫폼에는 존재하지 않는 특수한 상황(브라우저 작업 실패, 브라우저 종료, 프로세스 중단 등)에 대비해야 하는데 이를 트랜잭션을 이용해서 수행
    - 생성하는 메서드는 transaction()
    - 속성
      - READ_ONLY
      - READ_WRITE
      - VERSION_CHANGE
  - ✓ 객체 저장소 메서드
    - add(object): 추가, 동일한 인덱스가 존재하면 에러
    - put(object) : 추가, 동일한 인덱스가 존재하면 수정
    - get(key): 데이터 가져오기
    - delete(key): 데이터 삭제

로컬 저장소

- Indexed DB

로컬 저장소

- Indexed DB
  - ✓ indexeddb.html

```
<!DOCTYPE html>
<html>
<head>
  <title>IndexedDB API</title>
  <link rel="stylesheet" href="resource/indexeddb.css">
  <script src="resource/indexeddb.js"></script>
</head>
<body>
  <section id="formbox">
    <form name="form">
      <p>Keyword:<br><input type="text" name="keyword" id="keyword"></p>
      <p>Title:<br><input type="text" name="text" id="text"></p>
      <p>Year:<br><input type="text" name="year" id="year"></p>
      <p><input type="button" name="save" id="save" value="Save"></p>
    </form>
  </section>
  <section id="databox">
    No Information available
  </section>
</body>
</html>
```

로컬 저장소

- Indexed DB
  - ✓ resource/indexeddb.css

```
#formbox{
  float: left;
  padding: 20px;
  border: 1px solid #999999;
}
#databox{
  float: left;
  width: 400px;
  margin-left: 20px;
  padding: 20px;
  border: 1px solid #999999;
}
#keyword, #text{
  width: 200px;
}
#databox > div{
  padding: 5px;
  border-bottom: 1px solid #999999;
}
```

로컬 저장소

- Indexed DB
  - ✓ resource/indexeddb.js

```
function initiate() {
    databox = document.getElementById('databox');

    window.indexedDB = window.indexedDB || window.mozIndexedDB
                        || window.webkitIndexedDB || window.msIndexedDB;
    window.IDBTransaction = window.IDBTransaction || window.webkitIDBTransaction ||
    window.msIDBTransaction;
    window.IDBKeyRange = window.IDBKeyRange || window.webkitIDBKeyRange ||
    window.msIDBKeyRange
    window.IDBCursor = window.IDBCursor || window.webkitIDBCursor ||
    window.msIDBCursor
```

로컬 저장소

- Indexed DB
  - ✓ resource/indexeddb.js

```
openDB = indexedDB.open('mydb', 1);
openDB.onupgradeneeded = function() {
    var db = {}
    db.result = openDB.result;
    db.store = db.result.createObjectStore('movies', {
                    keyPath : 'id'
        });
    db.index = db.store.createIndex('SearchYear', 'date', {
                    unique : false
        });
};

var button = document.getElementById('save');
button.addEventListener('click', addobject, false);

openDB.onsuccess = function(){
        show()
}
}
window.addEventListener('load', initiate, false);
```

로컬 저장소

- Indexed DB
  - ✓ resource/indexeddb.js

```
function addobject(){
    var keyword=document.getElementById('keyword').value;
    var title=document.getElementById('text').value;
    var year=document.getElementById('year').value;

    var db = {};
    db.result = openDB.result;
    db.tx = db.result.transaction("movies", "readwrite");
    db.store = db.tx.objectStore("movies");
    var request=db.store.add({id: keyword, name: title
            , date: year});
    //request.addEventListener('success', function(){ show(keyword) }, false);
    document.getElementById('keyword').value='';
    document.getElementById('text').value='';
    document.getElementById('year').value='';
    show()
}
```

로컬 저장소

- Indexed DB
  - ✓ resource/indexeddb.js

```
function show() {
    databox.innerHTML =''
    var db = {};
    db.result = openDB.result;
    db.tx = db.result.transaction("movies", "readwrite");

    var objectstore = db.tx.objectStore('movies');
    var cursor = objectstore.openCursor()
    cursor.addEventListener("success", function(e){
            var cur = e.result || e.target.result
            if(cur){
                    console.log(cur.value)
                    databox.innerHTML +='<div>'+cur.value.id+' - '+cur.value.name+' -
    '+cur.value.date +
                    '<button
    onclick="remove(₩''+cur.value.id+'₩')">remove</button></div>';
                    cur.continue()
            }
    })
}
```

로컬 저장소

- Indexed DB
  - ✓ resource/indexeddb.js

```
function remove(keyword){
    if(confirm('Are you sure?')){
            var db = {};
            db.result = openDB.result;
            db.tx = db.result.transaction("movies", "readwrite");
            db.store = db.tx.objectStore("movies");
        var request=db.store.delete(keyword);
        request.addEventListener('success', show, false);
    }
}
```

Web Worker

- Web Worker
  - ✓ 개요
    - 자바 스크립트 코드를 백그라운드에서 실행하기 위한 기술
    - 백그라운드에서 실행되는 코드를 worker라고 부르며 UI(DOM)와는 완전히 별개로 동작
    - 기존 웹 브라우저는 웹 페이지를 하나의 스레드로 실행
    - JavaScript로 만들어진 기능이나 DOM 조작에 의한 UI변경 처리도 하나의 스레드에서 동작시키므로 처리 시간이 긴 작업을 수행하게 되면 브라우저가 얼어버리는 현상이 발생할 수 있음
    - 작업 시간이 긴 JavaScript 코드를 실행하는 경우 Web Worker를 잘 이용하면 위와 같은 상황을 예방
    - worker에서는 DOM과 main browser code에 접근하지 못하지만 localStorage와 XMLHttpRequests 에는 접근할 수 있음
    - browser와 worker는 message를 주고 받음
    - message는 복사본이 전달되어 message의 변경은 browser, worker 상호 영향을 주지 않음

Web Worker

- Web Worker
  - ✓ 작성
    - 워커의 생성

      var 워커변수 = new Worker("자바스크립트 파일경로");
    - 워커와 브라우저 간의 메시지 전송은 postMessage

      워커변수.postMessage("메시지");

      postMessage("메시지");
    - 자바스크립트 파일에서 처리(message 이벤트)

      onmessage = function(e)

      { 처리 구문;}

      addEventListener("message", function(e) {

      //처리구문; }, false)
    - 데이터는 이벤트 객체의 data 속성에 전달
    - 에러가 발생했을 때는 error 이벤트가 발생
    - 워커의 중지는 terminate()를 호출하면 중지
    - 워커에 다른 자바 스크립트 파일의 내용 포함: importScripts("자바스크립트 파일 나열");

Web Worker

- Web Worker
  - ✓ resource/worker.js

```
onmessage = function(event)
{
  var num = event.data;
  var result = 0;
  for(var i=1; i<=num; i++)
  {
     result += i;
  }
  postMessage(result);
};
```

Web Worker

- Web Worker
  - ✓ worker.html

```
<!DOCTYPE html>
<html>

<head>
    <title>웹 워커 사용</title>
    <meta charset="UTF-8">
    <script type="text/javascript">
    var worker;

    function calculate() {
            if (worker)
                    worker.terminate();
            var num = document.getElementById("num").value;
            worker = new Worker("resource/worker.js");
            worker.onmessage = function (event) {
                    alert("합은 " + event.data + "입니다");
            };
            worker.postMessage(num);
    }
```

Web Worker

- Web Worker
  - ✓ worker.html

```
        function stopCalculation() {
            if (worker)
                worker.terminate();
            alert("중지");
        }
        </script>
    </head>

    <body>
        <h1> Web Worker </h1>
        <h2> 숫자의 합을 구해줍니다</h2>
        숫자 입력 <input type="text" id="num" />
        <button onclick="calculate()">합계 </button>
        <button onclick="stopCalculation()">계산 중지 </button>
    </body>

    </html>
```

Application Cache

- Application Cache
  - ✓ HTML5는 오프라인 환경을 고려한 API들(Storage, Database)이 있는데 이 API들이 오프라인에서 정상적으로 작동하기 위해서는 CSS, 이미지, 자바스크립트 등과 같은 리소스를 필요로 하는데 오프라인 상태에서도 웹 애플리케이션으로의 접근을 가능케 하는 매우 중요한 기능
  - ✓ Application Cache를 사용했을 때의 장점
    - 오프라인 브라우징 - 오프라인 상태에서도 사용자가 사이트에 접근할 수 있음
    - 속도 향상 - 로컬 영역에 저장된 리소스들은 매우 빠른 로드 속도로 호출
    - 서버 부하 감소 - 브라우저는 오직 리소스가 변경된 경우에만 다운로드를 시도
  - ✓ <html> 태그에 manifest 속성을 지정하여 캐시할 파일들의 목록을 지정할 수 있음

    <html manifest= " 파일경로">

Application Cache

- Application Cache
  - ✓ manifest 파일 구성 방법

```
CACHE MANIFEST
# v2

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

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

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

Application Cache

- Application Cache
  - ✓ cache 갱신

```
var appCache = window.applicationCache;

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

...

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

Communications

- Web Push - Server-Sent Events
  - ✓ HTML5가 등장하기 전까지는 HTML에 서버 푸시를 위한 표준화된 기술이 없었기 때문에 웹에서 실시간 정보를 받아와야 할 때 외부 플러그 인을 이용하거나 서버 푸시를 흉내 낸 Ajax 폴링(polling) 기법 등을 사용
  - ✓ 플러그인 종속적인 웹은 해당 플러그 인을 설치해야 하는 불편함이 있으며 폴링처럼 주기적인 요청을 통한 구현은 쓸모없는 요청의 발생으로 인한 대역폭의 낭비가 불가피
  - ✓ HTML5의 Server-Sent Events(이하 SSE)는 이러한 문제 없이 서버가 필요할 때마다 클라이언트에게 데이터를 줄 수 있게 해주는 서버 푸시 기술
  - ✓ SSE의 장점
    - 전통적인 HTTP를 통해 통신하므로 다른 프로토콜이 필요가 없음
    - 재접속 처리 같은 대부분의 저수준 처리가 자동
    - IE를 제외한 브라우저 대부분을 지원

Communications

- Web Push
  - ✓ 클라이언트에서는 다음 코드처럼 EventSource 객체를 생성하는 것만으로 주기적인 서버 호출이 발생

```
var eventSource = new EventSource("server.jsp");
```

  - ✓ 데이터 수신 이벤트: 서버로부터 전달받는 데이터를 처리하기 위해 수신 이벤트를 정의 하는데 EventSource 객체를 통해 정의하면 됨
  - ✓ 이벤트로 전달되는 data 속성으로 수신 데이터를 액세스 할 수 있음

```
eventSource.addEventListener("message",
    function(e){
      alert(e.data);
    }
,false);
```

Communications

- Web Push
  - ✓ 서버 측에서 클라이언트로 전달하는 데이터는 일반적인 텍스트 형태이지만 그 포맷이 정해져 있으며 몇 가지 규칙을 따라야 함
    - MIME 타입: 서버 데이터는 text/event-stream 라는 MIME 타입으로 제공
    - 문자 인코딩: 서버 데이터의 문자 인코딩은 UTF-8 형식
    - 빈 줄은 이벤트를 구분하는 역할
    - 주석은 :(콜론)
    - 데이터는 '필드 명: 필드 값' 형식
      - data : 서버가 전달할 실제 데이터를 정의
      - retry: 반복 주기를 설정(단위: millisecond)
      - event: 이벤트 이름을 지정(지정하지 않으면 기본값인 message 가 된다)
      - id: 이벤트 id를 지정

Communications

- Web Push
  - ✓ 서버 – push.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    response.setContentType("text/event-stream");
    response.setCharacterEncoding("UTF-8");

    java.util.Random r = new java.util.Random();
    out.write("data: " + (r.nextInt(46) + 1) + "₩n₩n");
    Thread.sleep(5000);
%>
```

Communications

- Web Push
  - ✓ 서버 – push.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <input type="button" value="웹 푸시 시작" onclick="pushStart()">
    <br /> 메시지:
    <span id="disp"></span>
</body>
<script>
    function pushStart() {
            var eventSource = new EventSource("push.jsp");
            eventSource.onmessage = function(event) {
                    document.getElementById('disp').innerHTML = event.data;
            };
    }
</script>
</html>
```

Communications

- Web Socket
  - ✓ 웹 환경에서 실시간 양방향 통신을 위한 Spec이 웹 소켓(Web Socket)
  - ✓ 웹 소켓은 웹 서버와 웹 브라우저가 지속적으로 연결된 TCP 라인을 통해 실시간으로 데이터를 주고 받을 수 있도록 하는 HTML5의 새로운 사양으로 웹 소켓을 이용하면 일반적인 TCP소켓과 같이 연결 지향 양방향 전이중 통신이 가능
  - ✓ 웹 소켓이 필요한 경우
    - 실시간 양방향 데이터 통신이 필요한 경우
    - 많은 수의 동시 접속자를 수용해야 하는 경우
    - 브라우저에서 TCP 기반의 통신으로 확장해야 하는 경우
    - 개발자에게 사용하기 쉬운 API가 필요할 경우
    - 클라우드 환경이나 웹을 넘어 SOA 로 확장해야 하는 경우

Communications

- Web Socket
  - ✓ 웹 소켓 역시 일반적인 TCP 소켓 통신처럼 웹 소켓 역시 서버와 클라이언트간 데이터 교환이 이루어지는 형태로 클라이언트 코드만으로 실행 가능한 코드를 만들 수 없음
  - ✓ 웹 소켓 클라이언트
    - 웹 소켓이 제공하는 클라이언트 측 API는 매우 심플
    - 기본적인 서버 연결, 데이터 송신, 데이터 수신만 정의하면 나머지는 일반적인 스크립트 로직
    - 웹 소켓이 동작하기 위해서 제일 처음 서버와 연결이 되어야 하는데 HTML5가 제공하는 WebSocket 객체를 통해 서버 연결을 수행
      - var wSocket = new WebSocket("ws://yourdomain/demo");
    - 서버와 연결이 되면 이제부터 데이터를 주고 받을 수 있게 되는데 WebSocket 객체의 send 함수로 데이터를 서버로 송신
      - wSocket.send(" 송신 메시지 ");
    - 서버에서 푸시(전송)하는 데이터를 받으려면 message 이벤트를 구현
      - wSocket.onmessage = function(e){ //매개변수 e를 통해 수신된 데이터를 조회
    - open 이벤트: 연결이 설정되면 발생
    - close 이벤트: 연결이 끊어지면 발생

Communications

- Web Socket
  - ✓ Java Web Server
    - web-socket-api.jar 파일을 프로젝트의 WebContent/WEB-INF/lib 디렉토리에 복사

Communications

- Web Socket
  - ✓ Java Web Server
    - src/main/java 디렉토리에 클래스를 추가하고 작성 - WebSocketService

```
@ServerEndpoint("/chat")
public class WebSocketService {
    static List<Session> list = new ArrayList<Session>();
    @OnMessage
    public void onMessage(String message, Session session) throws IOException,
    InterruptedException {
        System.out.println("Received: " + message);
        for (Session s : list) {
                    s.getBasicRemote().sendText(message);
        }
    }

    @OnOpen
    public void onOpen(Session session) {
        System.out.println("add:" + session);
        System.out.println("Client connected");
        list.add(session);
    }
```

Communications

- Web Socket
  - ✓ Java Web Server
    - src/main/java 디렉토리에 클래스를 추가하고 작성 - WebSocketService

```
    @OnClose
    public void onClose(Session session) {
       System.out.println("del:" + session);
       System.out.println("Connection closed");
       list.remove(session);
    }
}
```

Communications

- Web Socket
  - ✓ Client
    - chat.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>채팅</title>
</head>
<body>
    닉네임:
    <input type="text" id="nick" size="20" />
    <br /> 보내는 메시지:
    <input type="text" id="message" size="60" />
    <input type="button" onclick="start()" value="전송" />
    <br /> 받은 메시지:
    <textarea id="disp" cols="80" rows="20"></textarea>

</body>
```

Communications

- Web Socket
  - ✓ Client
    - chat.html

```
<script>
    var webSocket = new
    WebSocket('ws://172.30.67.250:9000/Communications/chat');
    webSocket.onerror = function(event) {
       onError(event)
    };
    webSocket.onopen = function(event) {
       onOpen(event)
    };
    webSocket.onmessage = function(event) {
       onMessage(event)
    };

    var disp = document.getElementById("disp");
    function onMessage(event) {
       disp.value = event.data + "₩n" + disp.value;
    }
    function onOpen(event) {
       disp.value = '연결 성공';
    }
```

Communications

- Web Socket
  - ✓ Client
    - chat.html

```
        function onError(event) {
          alert(event.data);
        }
      function start() {
          webSocket.send(document.getElementById("nick").value + ":" +
        document.getElementById('message').value);
          document.getElementById('message').value = "";
        }
      </script>
      </html>
```

HTML5 실습

❖ 메인 화면

HTML5 실습

- 메인 화면
  - ✓ index.html

```
<!DOCTYPE HTML>
<html>

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
    scale=1.0, user-scalable=no" />
    <title>ToDoList</title>
```

HTML5 실습

❖ 메인 화면

✓ index.html

```
<style>
        body {
                font-family: Tahoma, Verdana, Geneva, sans-serif;
                margin: 6em;
                font-weight: bold;
        }

        ul {
                list-style-type: none;
                margin: 0;
                padding: 0;
        }

        li {
                float: left;
                margin-bottom: 3em;
        }
```

HTML5 실습

❖ 메인 화면

✓ index.html

```
<style>
        a {
                display: block;
                width: 10em;
                margin: 0 5px;
                padding: .5em 1em;
                text-decoration: none;
                text-align: center;
                color: #fff;
                background-color: rgb(190, 190, 190);
                border: 2px solid rgb(175, 175, 175);
                border-radius: 6px;
                text-shadow: #666 .1em .1em .1em;
                box-shadow: 0 5px 3px rgba(0, 0, 0, .5);
                position: relative;
-webkit-transition: background-color 0.2s, border-color 0.2s, top .2s, box-shadow 0.2s;
-moz-transition: background-color 0.2s, border-color 0.2s, top .2s, box-shadow 0.2s;
-o-transition: background-color 0.2s, border-color 0.2s, top .2s, box-shadow 0.2s;
-ms-transition: background-color 0.2s, border-color 0.2s, top .2s, box-shadow 0.2s;
transition: background-color 0.2s, border-color 0.2s, top .2s, box-shadow 0.2s;
        }
```

HTML5 실습

❖ 메인 화면

✓ index.html

```
            a:hover,
            a:focus {
                        background-color: #fdca00;
                        border-color: #fda700;
            }

            a:active {
                        top: 3px;
                        box-shadow: 0 1px 2px rgba(0, 0, 0, .5);
            }
        </style>

    </head>
```

HTML5 실습

❖ 메인 화면

✓ index.html

```
<body>
    <ul>
        <li><a href="user/register.html">회원가입</a></li>
        <li><a href="user/login.html">로그인</a></li>
        <li><a href="todo/list.html">목록확인</a></li>
        <li><a href="todo/write.html">목록만들기</a></li>
    </ul>
</body>

</html>
```

HTML5 실습

❖ 회원 가입 화면 만들기(user/register.html)

HTML5 실습

- 회원 가입 화면
  - ✓ user/register.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
    scale=1.0, user-scalable=no" />
    <title>폼 스타일</title>
    <style>
        ul {
            list-style-type: none;
        }

        ul li {
            clear: both;
        }
```

HTML5 실습

- 회원 가입 화면
  - ✓ user/register.html

```
form {
    width: 50em;
    border: 1px solid #666;
    border-radius: 10px;
    box-shadow: .2em .2em .5em #999;
    background-color: #d0e9f6;
    padding: 1em;
    overflow: hidden;
}

label {
    display: block;
    float: left;
    width: 10em;
    text-align: right;
    margin-right: .5em;
    color: #04699d;
}
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
input.textinput {
    width: 30em;
    height: 2em;
    border: 1px solid #666;
    margin-bottom: 1em;
    display: block;
    float: left;
}

input.fileinput {
    width: 15em;
    height: 2em;
    border: 1px solid #666;
    margin-bottom: 1em;
    margin-right: 1em;
    display: block;
    float: left;
}
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
img {
    width: 7em;
    height: 7em;
    border: 1px solid #666;
    margin-bottom: 1em;
}

textarea {
    display: block;
    width: 30em;
    height: 5em;
    border: 1px solid #666;
    margin-bottom: 1em;
    line-height: 1.25;
    overflow: auto;
    resize: none;
}
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
input.textinput,
textarea {
    font-family: Georgia, "Times New Roman", Times, serif;
    font-size: .875em;
}

input[type="submit"],
input[type="button"] {
    display: block;
    width: 10em;
    height: 2em;
    float: left;
    background: white;
    font-size: inherit;
    border: 1px solid #04699d;
    border-radius: 4px;
    box-shadow: 2px 2px 3px rgba(0, 0, 0, .5);
    margin-top: 1em;
    margin-right: 1em;
    color: #C00;
}
```

HTML5 실습

- 회원 가입 화면
  - ✓ user/register.html

```
input[type="submit"] {
    margin-left: 5em;
}

#msg {
    color: red;
}
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
@media screen and (max-width: 960px) {
    form {
        width: 20em;
        border: 1px solid #666;
        border-radius: 10px;
        box-shadow: .2em .2em .5em #999;
        background-color: #d0e9f6;
        padding: 1em;
        overflow: hidden;
    }

    label {
        display: block;
        float: left;
        width: 4em;
        text-align: right;
        margin-right: .5em;
        color: #04699d;
    }
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
input.textinput {
    width: 13em;
    height: 2em;
    border: 1px solid #666;
    margin-bottom: 1em;
}

img {
    width: 7em;
    height: 7em;
    border: 1px solid #666;
    margin-bottom: 1em;
    margin-left: 5em;
}
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
textarea {
    display: block;
    width: 13em;
    height: 5em;
    border: 1px solid #666;
    margin-bottom: 1em;
    line-height: 1.25;
    overflow: auto;
    resize: none;
}

input.textinput,
textarea {
    font-family: Georgia, "Times New Roman", Times, serif;
    font-size: .875em;
}
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
            input[type="submit"],
            input[type="button"] {
                display: block;
                width: 10em;
                height: 2em;
                float: left;
                background: white;
                font-size: inherit;
                border: 1px solid #04699d;
                border-radius: 4px;
                box-shadow: 2px 2px 3px rgba(0, 0, 0, .5);
                margin-top: 1em;
                color: #C00;
                margin-left: 3em;
            }
        }
    </style>
</head>
```

HTML5 실습

- 회원 가입 화면
  - ✓ user/register.html

```
<body>
    <form method="post" action="server.jsp" id="registerform">
        <h2>회원가입</h2>
        <div align="center" id="msg"></div>
        <ul>
            <li>
                <label for="email">이메일</label>
                <input type="text" id="email" name="email" class="textinput" placeholder="
이메일 주소를 입력하세요" />
            </li>
            <li>
                <label for="userpw">비밀번호</label>
                <input type="password" id="userpw" name="userpw" class="textinput" />
            </li>
            <li>
                <label for="userpw1">비밀번호확인</label>
                <input type="password" id="userpw1" class="textinput" />
            </li>
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
                <li>
                    <label for="nickname">별명</label>
                    <input type="text" id="nickname" class="textinput" />
                </li>
                <li>
                    <label for="myimage">이미지</label>
                    <input type="file" id="myimage" class="fileinput" accept="image/*" />
                    <img id="img" />
                </li>
                <li class="buttons">
                    <input type="submit" value="회원가입" />
                    <input type="button" value="메인" id="mainbtn" />
                    <input type="button" value="로그인" id="loginbtn" />
                </li>
            </ul>
        </form>
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
<script>
    var mainbtn = document.getElementById("mainbtn");
    var loginbtn = document.getElementById("loginbtn");

    var registerform = document.getElementById("registerform");

    var email = document.getElementById("email");
    var userpw = document.getElementById("userpw");
    var userpw1 = document.getElementById("userpw1");
    var nickname = document.getElementById("nickname");
    var msg = document.getElementById("msg");
```

HTML5 실습

- 회원 가입 화면
  - ✓ user/register.html

```
document.getElementById("myimage").addEventListener("change", function () {
    readURL(this);
 })

function readURL(input) {
    if (input.files && input.files[0]) {
        var filename = input.files[0].name;

        var reader = new FileReader();

        reader.addEventListener("load", function (e) {
            document.getElementById('img').src = e.target.result;
        });

        reader.readAsDataURL(input.files[0]);
    }
 }
```

HTML5 실습

- 회원 가입 화면
  - ✓ user/register.html

```
mainbtn.addEventListener("click", function (event) {
    location.href = "../index.html";
});

loginbtn.addEventListener("click", function (event) {
    location.href = "login.html";
});
```

HTML5 실습

- 회원 가입 화면
  - ✓ user/register.html

```
registerform.addEventListener("submit", function (event) {
    msg.innerHTML = '';
    var flag = false;
    if (email.value.trim().length < 1) {
        msg.innerHTML = '이메일은 필수 입력입니다.<br/>';
        flag = true;
    } else {
        var emailRegExp =
            /^[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*@[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*.[a-zA-
Z]{2,3}$/i;
        if (!emailRegExp.test(email.value.trim())) {
            msg.innerHTML = '잘못된 이메일 형식입니다.<br/>';
            flag = true;
        }
    }
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
        if (userpw.value.trim().length < 1) {
            msg.innerHTML += '비밀번호는 필수 입력입니다.<br/>';
            flag = true;
        } else {
            var pwRegExp = /^.*(?=^.{8,15}$)(?=.*₩d)(?=.*[a-z])(?=.*[A-
Z])(?=.*[!@#$%^&+=]).*$/;
            if (!pwRegExp.test(userpw.value.trim())) {
                msg.innerHTML = '비밀번호는 숫자와 영문 대소문자 그리고 특수문자가 포
함되어야 합니다.<br/>';
                flag = true;
            } else {
                if (userpw.value.trim() !== userpw1.value.trim()) {
                    msg.innerHTML += '2개의 비밀번호는 같아야 합니다.<br/>';
                    flag = true;
                }
            }
        }
```

HTML5 실습

❖ 회원 가입 화면

✓ user/register.html

```
            if (nickname.value.trim().length < 1) {
                msg.innerHTML += '별명은 필수 입력입니다.<br/>';
                flag = true;
            } else {
                var nicknameRegExp = /^[a-zA-z가-힣0-9]{2,5}$/;
                if (!nicknameRegExp.test(nickname.value.trim())) {
                    msg.innerHTML = '닉네임은 영문 한글 숫자로 2자 이상 5자 이하이어야 합
    니다.<br/>';
                    flag = true;
                }
            }
            if (flag == true) {
                event.preventDefault();
            }
        });
    </script>
</body>

</html>
```

HTML5 실습

❖ 로그인 화면

✓ user/login.html

HTML5 실습

❖ 로그인 화면

✓ user/login.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>로그인</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
    scale=1.0, user-scalable=no"/>
<style>
h1{
    text-align:center;
}
a {
    color: #333;
    text-decoration: none;
}
```

HTML5 실습

❖ 로그인 화면

✓ user/login.html

```
input {
    -webkit-writing-mode: horizontal-tb !important;
    text-rendering: auto;
    color: initial;
    letter-spacing: normal;
    word-spacing: normal;
    text-transform: none;
    text-indent: 0px;
    text-shadow: none;
    display: inline-block;
    text-align: start;
    -webkit-appearance: textfield;
    background-color: white;
    -webkit-rtl-ordering: logical;
    cursor: text;
    margin: 0em;
    font: 400 13.3333px Arial;
    padding: 1px 0px;
    border-width: 2px;
    border-style: inset;
    border-color: initial;
    border-image: initial;
}
```

HTML5 실습

❖ 로그인 화면

✓ user/login.html

```
.inner_login {
    position: absolute;
    left: 50%;
    top: 30%;
    margin: -145px 0 0 -160px;
}

.login_myweb {
    position: relative;
    width: 320px;
    margin: 0 auto;
}

.screen_out {
    position: absolute;
    width: 0;
    height: 0;
    overflow: hidden;
    line-height: 0;
    text-indent: -9999px;
}
```

HTML5 실습

❖ 로그인 화면

✓ user/login.html

```
body, button, input, select, td, textarea, th {
    font-size: 13px;
    line-height: 1.5;
    -webkit-font-smoothing: antialiased;
}

fieldset, img {
    border: 0;
}
```

HTML5 실습

❖ 로그인 화면

✓ user/login.html

```
.login_myweb .box_login {
    margin: 35px 0 0;
    border: 1px solid #ddd;
    border-radius: 3px;
    background-color: #fff;
    box-sizing: border-box;
}

.login_myweb .inp_text {
    position: relative;
    width: 100%;
    margin: 0;
    padding: 18px 19px 19px;
    box-sizing: border-box;
}

.login_myweb .inp_text+.inp_text {
    border-top: 1px solid #ddd;
}
```

HTML5 실습

❖ 로그인 화면

✓ user/login.html

```
.inp_text input {
    display: block;
    width: 100%;
    height: 100%;
    font-size: 13px;
    color: #000;
    border: none;
    outline: 0;
    -webkit-appearance: none;
    background-color: transparent;
}

.btn_login {
    margin: 20px 0 0;
    width: 100%;
    height: 48px;
    border-radius: 3px;
    font-size: 16px;
    color: #fff;
    background-color: #000;
}
```

HTML5 실습

- 로그인 화면
  - ✓ user/login.html

```
.login_append {
    overflow: hidden;
    padding: 15px 0 0;
}

.inp_chk {
    display: inline-block;
    position: relative;
    margin-bottom: -1px;
}

.login_append .inp_chk {
    float: left;
}

.inp_chk .inp_radio {
    position: absolute;
    z-index: -1;
    top: 0;
    left: 0;
    width: 18px;
    height: 18px;
    border: 0;
```

HTML5 실습

❖ 로그인 화면

✓ user/login.html

```
.login_append {
    overflow: hidden;
    padding: 15px 0 0;
}

.inp_chk {
    display: inline-block;
    position: relative;
    margin-bottom: -1px;
}

.login_append .inp_chk {
    float: left;
}

.inp_chk .inp_radio {
    position: absolute;
    z-index: -1;
    top: 0;
    left: 0;
    width: 18px;
    height: 18px;
    border: 0;
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html)

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 본체)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>To Do</title>
<meta name="viewport"
		content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=no" />
</head>
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 본체)

```
<body>
		<h1>To Do</h1>
		<div class="inner_write">
			<div class="todo_write">
				<div id="msg" class="msg"></div>
				<form method="post" id="todoform" action="todowrite">
					<fieldset>
				<legend class="screen_out">To Do Write</legend>
				<div class="box_write">
					<div class="inp_text">
						<label for="date">날짜</label>
						<input type="date" id="date"
name="date" max="2030-12-31" />
					</div>
				<div class="inp_text">
					<label for="time">시간</label>
					<input type="time" id="time" name="time" />
				</div>
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 본체)

```
                    <div class="inp_text">
                            <label for="category">카테고리  </label>
                            <select name="category" id="category">
                            <option value="promise" id="promise">약속</option>
                            <option value="house">집안일</option>
                            <option value="study">학습</option>
                            <option value="etc">기타</option>
                            </select>
                    </div>
                    <div class="inp_text">
                            <label for="content">내용</label>
                            <textarea rows="5" cols="30" name="content"
id="content"></textarea>
                    </div>
            </div>
                    <button type="submit" class="btn_write">작성</button>
                    <button class="btn_write" id="btn_main">메인</button>
            </fieldset>
            </form>
            </div>
            </div>
</body>
</html>
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 스타일)

```
<style>
h1 {
		text-align: center;
}

a {
		color: #333;
		text-decoration: none;
}
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 스타일)

```
input {
        -webkit-writing-mode: horizontal-tb !important;
        text-rendering: auto;
        color: initial;
        letter-spacing: normal;
        word-spacing: normal;
        text-transform: none;
        text-indent: 0px;
        text-shadow: none;
        display: inline-block;
        text-align: start;
        -webkit-appearance: textfield;
        background-color: white;
        -webkit-rtl-ordering: logical;
        cursor: text;
        margin: 0em;
        font: 400 13.3333px Arial;
        padding: 1px 0px;
        border-width: 2px;
        border-style: inset;
        border-color: initial;
        border-image: initial;
}
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 스타일)

```
.inner_write {
		position: absolute;
		left: 50%;
		top: 30%;
		margin: -145px 0 0 -160px;
}

.todo_write {
		position: relative;
		width: 320px;
		margin: 0 auto;
}

.screen_out {
		position: absolute;
		width: 0;
		height: 0;
		overflow: hidden;
		line-height: 0;
		text-indent: -9999px;
}
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 스타일)

```
body, button, input, select, td, textarea, th {
		font-size: 13px;
		line-height: 1.5;
		-webkit-font-smoothing: antialiased;
}

fieldset, img {
		border: 0;
}

.todo_write .box_write {
		margin: 35px 0 0;
		border: 1px solid #ddd;
		border-radius: 3px;
		background-color: #fff;
		box-sizing: border-box;
}
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 스타일)

```
.todo_write .inp_text {
		position: relative;
		width: 100%;
		margin: 0;
		padding: 18px 19px 19px;
		box-sizing: border-box;
}

.todo_write .inp_text+.inp_text {
		border-top: 1px solid #ddd;
}

.inp_text input {
		display: block;
		width: 100%;
		height: 100%;
		font-size: 13px;
		color: #000;
		border: none;
		outline: 0;
		-webkit-appearance: none;
		background-color: transparent;
}
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 스타일)

```
.btn_write {
		margin: 20px 0 0;
		width: 100%;
		height: 48px;
		border-radius: 3px;
		font-size: 16px;
		color: #fff;
		background-color: #000;
}

.write_append {
		overflow: hidden;
		padding: 15px 0 0;
}

.write_append .link_find {
		float: left;
		color: #777;
		margin-left: .5em;
		margin-right: .5em;
}
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 스타일)

```
.msg {
		color: red;
		text-align: center;
		font-size: 1.5em;
}
</style>
```

HTML5 실습

❖ ToDo 작성 화면 만들기(todo/write.html - 스크립트)

```
<script>
		var date = document.getElementById("date");
		var time = document.getElementById("time");
		var current = new Date();
		var year = current.getYear() + 1900;
		var month = current.getMonth() + 1 > 9 ? current.getMonth() + 1 : "0" +
(current.getMonth() + 1);
		var day = current.getDate() > 9 ? current.getDate() : "0" + current.getDate();
		var today = year + "-" + month + "-" + day;
		date.value = today;
		var hour = current.getHours() > 9 ? current.getHours() : "0" + current.getHours();
		var minute = current.getMinutes() > 9 ? current.getMinutes() : "0" +
current.getMinutes();
		time.value = hour + ":" + minute

		var promise = document.getElementById("promise");
		promise.selected = true;

		var btn_main = document.getElementById("btn_main");
		btn_main.addEventListener("click", function(event){
			location.href = "../index.html";
			event.preventDefault();
		})
</script>
```