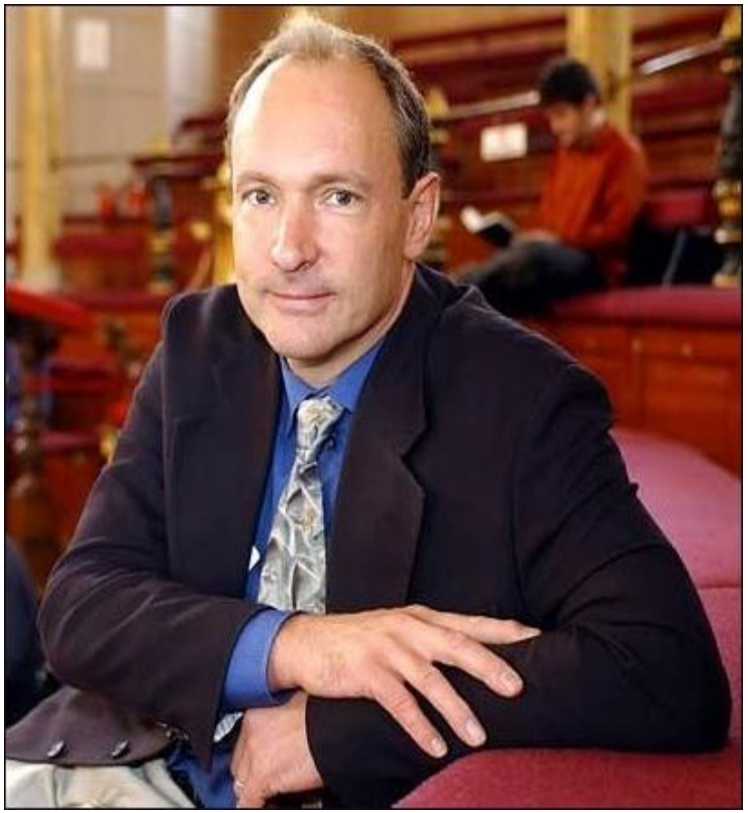
Web Client 기술

- HTML5
- CSS3
- JavaScript
- jQuery
- HTML5 API

  (canvas, multimedia, drag & drop, web storage, geolocation)
- AJAX

작성자 : 김 정현



"웹은 모든 사람들이 손쉽게 정보를 공유할 수 있는 공간이며 어떤 장애도 없이 이를 이용할 수 있어야 한다."

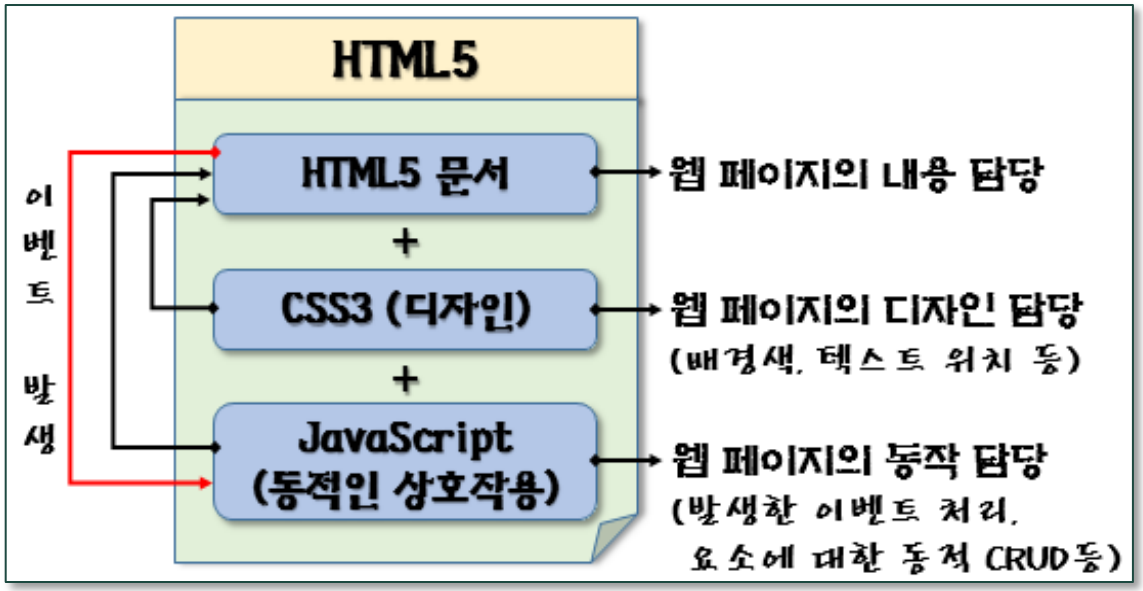
- 팀 버너스 리(WWW 창시자) -

웹 표준이란

특정 브라우저에서만 사용하는 비 표준화된 기술을 배제하고, **W3C**라는 조직에서 정한 표준 기술만을 사용하여, 웹 페이지 작성시 문서의 구조와 표현 방법 그리고 상호 동작을 구분하여 구현하는 것을 말함

웹 문서의 구조를 담당하는 것은 **HTML** 이고 표현을 담당하는 것은 **CSS**이며 동작을 담당하는 것은 자바스크립트임

3가지 요소가 유기적으로 결합하여 작동하게 하면, 웹 문서가 가벼워지며, 유지보수 및 수정 시에도 간편하고 빨리 처리할 수 있게 됨



Web 표준의 목표

Web 은 어디서든, 누구나 정보를 함께 공유하고 즐길 수 있어야 한다.

웹 접근성

- 월드 와이드 웹(World Wide Web)을 창시한 팀 버너스 리(Tim Berners-Lee)는 웹이란 '장애에 구애 없이 모든 사람들이 손쉽게 정보를 공유할 수 있는 공간'이라고 정의하였으며, 웹 콘텐츠를 제작할 때에는 장애에 구애됨이 없이 누구나 접근할 수 있도록 제작하여야 한다고 하였다.

- 누구든지 신체적·기술적 여건과 관계없이 웹사이트를 통하여 원하는 서비스를 이용할 수 있도록 접근성을 보장하는 것으로, 장애인, 고령자 등 정보이용접근이 어려운 사용자뿐만 아니라 일반적인 사용자들도 편리하게 웹 서비스를 이용할 수 있도록 견고한 컨텐츠를 만드는 것이다.

HTML5 란

- HTML5는 W3C(월드와이드웹콘소시엄)의 HTML WG(Working Group)을 통해서 만들어지고 있는 차세대 마크업 언어 표준이다.

- 문서 작성 중심으로 구성된 기존 표준에 그림, 동영상, 음악 등을 실행하는 기능 그리고 다양한 기능의 클라이언트 애플리케이션 구현 API 까지 포함시켰다.

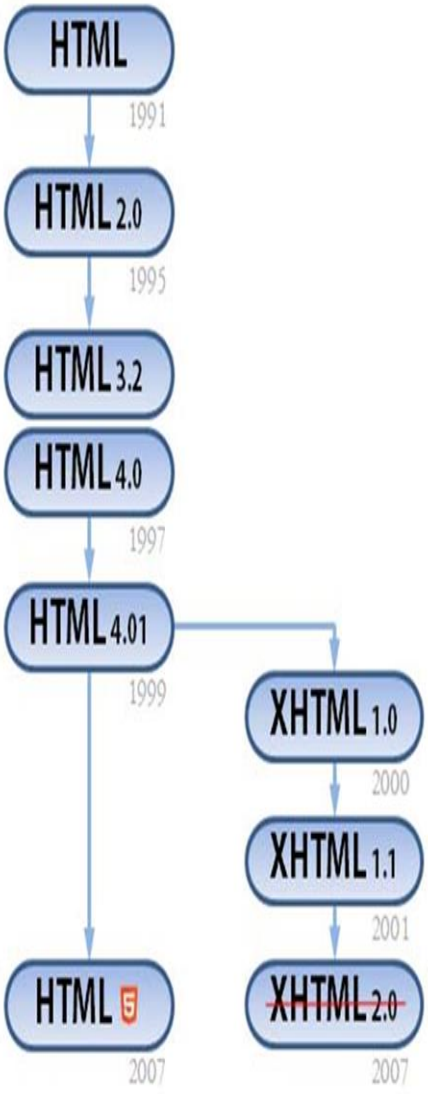
- 플랫폼 중립적이며 특정 디바이스에 종속되지 않으므로 스마트폰/태블릿 그리고 향후에는 스마트TV 나 웨어러블 등을 포함한 대부분의 스마트기기 환경에서의 공통 콘텐츠 플랫폼으로 활용될 것으로 전망된다.

- HTML5는 액티브X(Active-X)를 설치하지 않아도 동일한 기능을 구현할 수 있고, 특히 플래시(flash)나 실버라이트(Silverlight), 자바FX(JAVA FX) 없이도 웹 브라우저(web browser)에서 다양한 기능과 화려한 그래픽 효과를 낼 수 있다.

HTML5의 도입 배경

Web Hypertext Application Technology Working Group(WHATWG):
Apple, Mozillar, Opera 등이 공동으로 발족한 워킹 그룹으로 HTML을 진화시키고 그 성과를 표준화 단체에 제출하는 것을 목적으로 하였다.

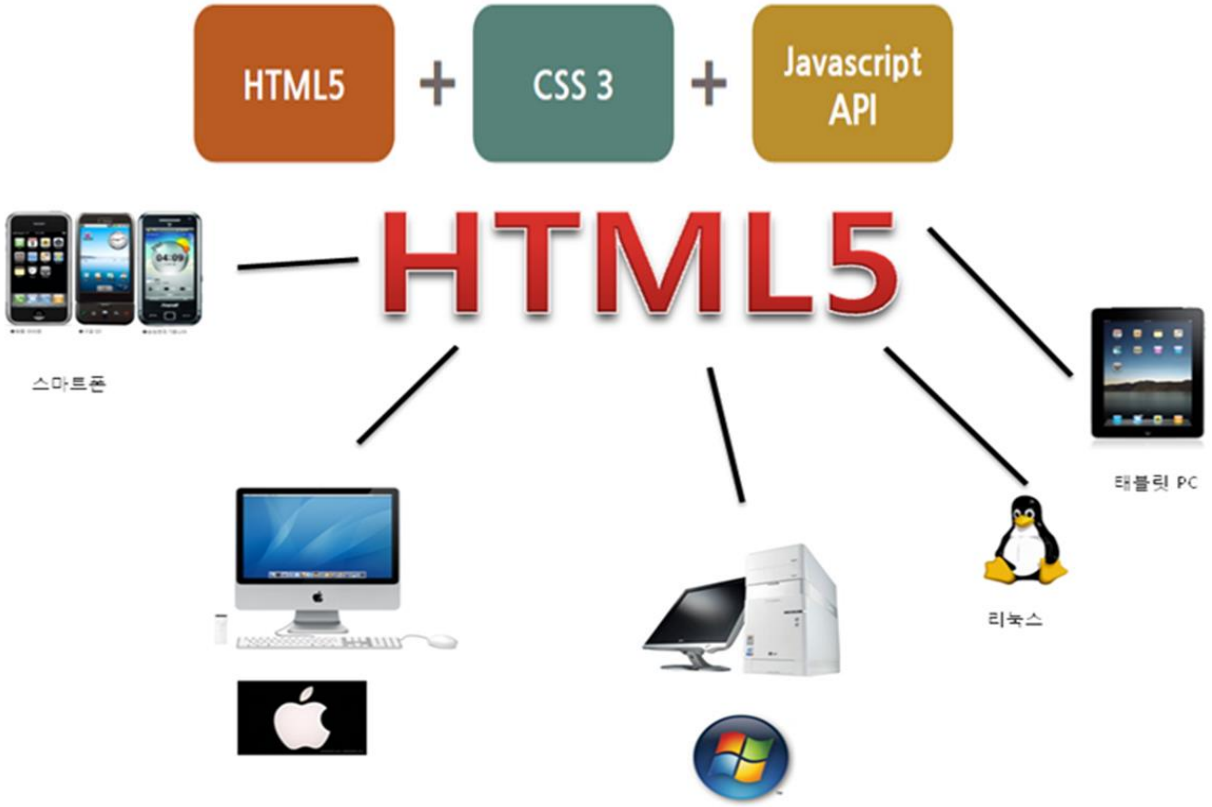
웹과 HTML



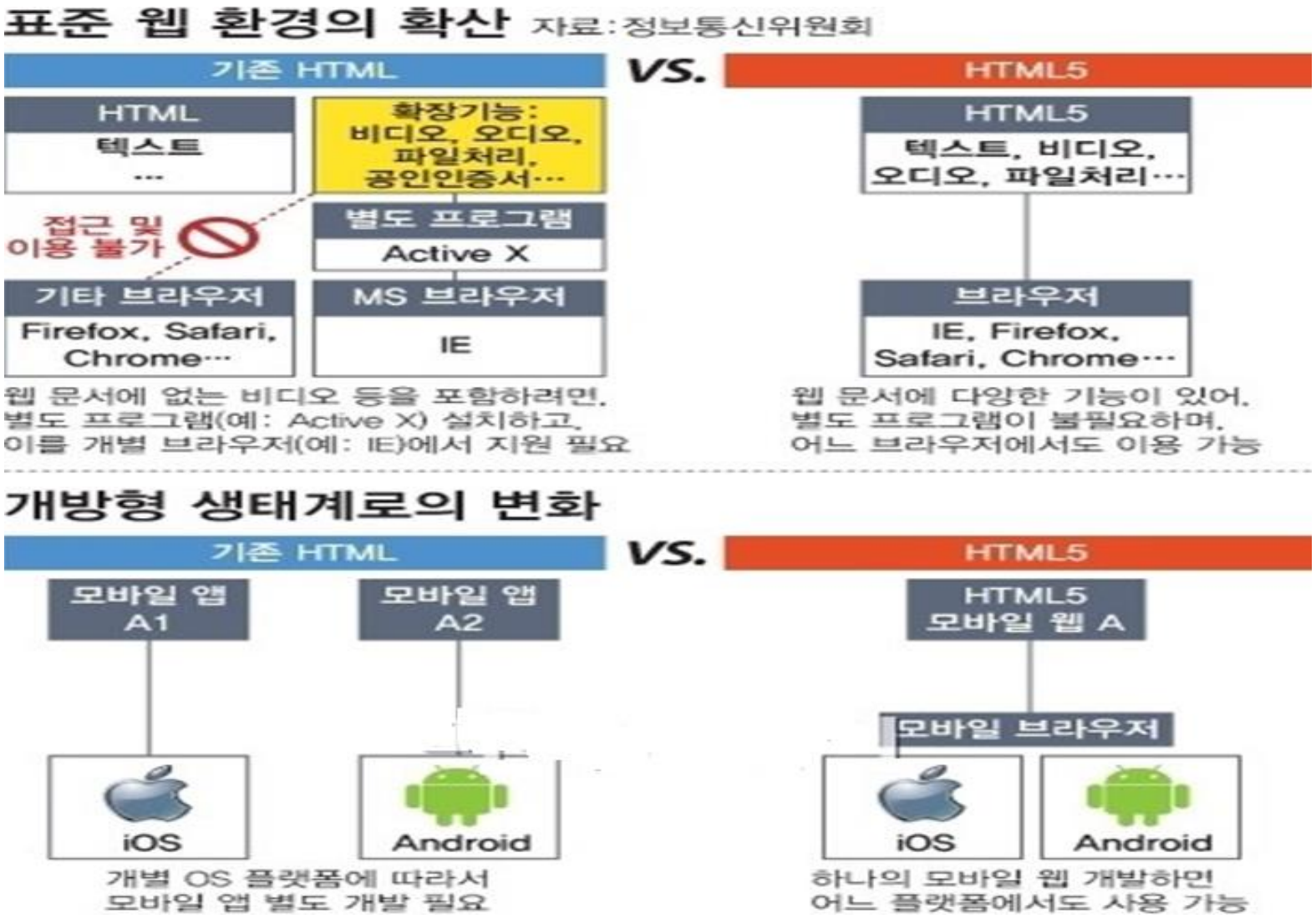
- HTML이라는 용어는 1991년에 처음 사용되었고, 1995년 World Wide Web Consortium(이하 W3C) 에서 정식으로 HTML 2.0 표준안을 발표했다.
- HTML은 웹(인터넷)에서 여러 가지 콘텐트를 보여주기 위하여 등장했던 언어로서 초기엔 텍스트 기반의 정보 전달이 목적이었으나 시간이 지나면서 텍스트 뿐만 아니라, 이미지, 오디오, 멀티미디어를 보여 주기 위한 목적으로 확정되었다.
- HTML의 자체적인 발전은 없었지만 Active-X와 플러그인과 같은 외부 기술에 주로 의존해 서 HTML의 한계를 벗어 나기 위한 방향으로 웹은 발전되어 왔다.
- 웹 브라우저 제조업체들이 해당 브라우저에서만 지원하는 확장 HTML과 부가기능 들을 경쟁적으로 개발하고 이를 독립 기술로 분류해 도입하면서 크로스 브라우징 문제가 발생하게 되었다.

HTML5의 영향

- **HTML5**는 문서의 내용과 구조는 **HTML** 태그(시멘틱태그)를 이용하고 스타일에 관련된 표현은 **CSS3** 그리고 웹 페이지에서의 다양한 동작 구현은 **JavaScript** 로 처리하게 하여 웹 표준에 기반한 웹 사이트를 개발하도록 지원하며 브라우저에 무관하고 디바이스에 무관한 웹 사이트 개발이 가능하게 한다.



HTML5의 영향


표준 웹 환경의 확산 자료:정보통신위원회
기존 HTML
VS.
HTML5
HTML
텍스트
...
확장기능:
비디오, 오디오,
파일처리,
공인인증서…
별도 프로그램
Active X
접근 및
이용 불가
기타 브라우저
Firefox, Safari,
Chrome…
MS 브라우저
IE
HTML5
텍스트, 비디오,
오디오, 파일처리…
브라우저
IE, Firefox,
Safari, Chrome…
웹 문서에 없는 비디오 등을 포함하려면,
별도 프로그램(예: Active X) 설치하고,
이를 개별 브라우저(예: IE)에서 지원 필요
웹 문서에 다양한 기능이 있어,
별도 프로그램이 불필요하며,
어느 브라우저에서도 이용 가능
개방형 생태계로의 변화
기존 HTML
VS.
HTML5
모바일 앱
A1
모바일 앱
A2
HTML5
모바일 웹 A
모바일 브라우저
iOS
Android
iOS
Android
개별 OS 플랫폼에 따라서
모바일 앱 별도 개발 필요
하나의 모바일 웹 개발하면
어느 플랫폼에서도 사용 가능

HTML5 문서 구성

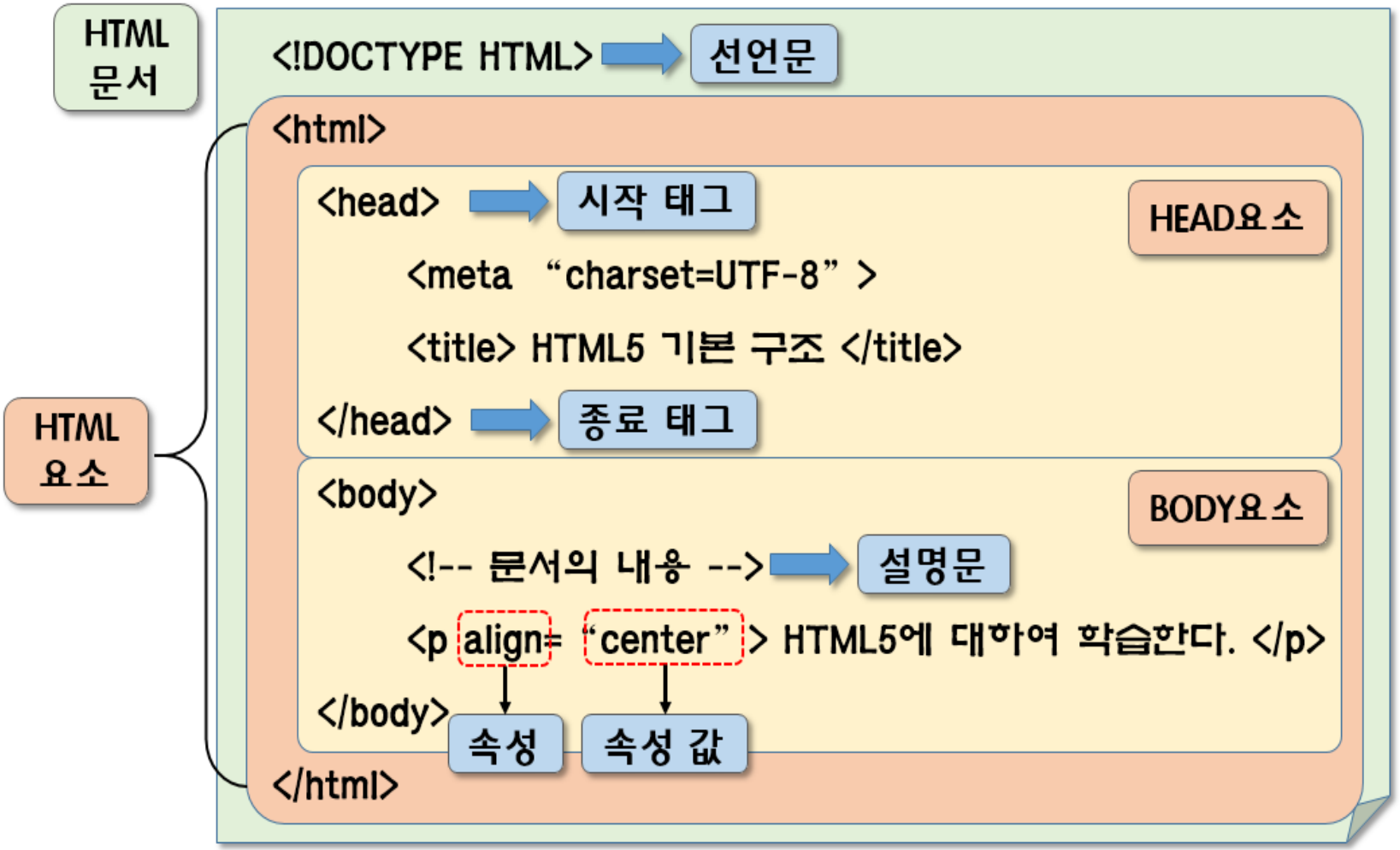
- **HTML5 문서의 작성은 다음과 같은 5가지의 기본 요소들로 구성된다.**

요소	의미	코드 예
태그(tag)	'<'와 '>'로 둘러싸인 문자열 시작태그(< >)와 종료태그(</ >)로 구성 문서 표현 방식을 지시	<title>웹문서내용</title>
내용(content)	태그로 둘러싸인 문자열	<title>웹문서내용</title>
엘리먼트(element)	태그와 내용을 포함한 전체 문자열 HTML 문서의 기본 구성 단위 (상위)엘리먼트 안에 (하위)엘리먼트가 계층적으로 포함될 수 있음	<title>웹문서내용<title>
속성(attribute)	엘리먼트의 상세한 표현(기능) 설정 사항을 지시 시작 태그 안에 사용	<title color="red"></title>
속성값(value)	속성값(' '또는 " "로 감싸야 함)	<title color="red"></title>

HTML5의 태그 리스트

분류	내용
Metadata Content	문서의 정보를 포함하여 현재 문서와 다른 문서의 관계를 설정하거나 나머지 내용의 표현이나 행동을 설정 또는 분류되지 않은 정보들을 포함
	base, link, meta, noscript, script, style, template, title
Flow Content	Body 요소 내에서 사용되는 대부분의 요소들이 포함되지만, 일부는 조건부로 포함
	a, abbr, address, area, article, aside, audio, b, bdi, bdo, blockquote, br, button, canvas, cite, code, data, datalist, del, details, dfn, dialog, div, dl, em, embed, fieldset, figure, footer, form, h1, h2, h3, h4, h5, h6, header, hr, i, iframe, img, input, ins, kbd, keygen, label, link, main, map, mark, math, menu, meta, meter, nav, noscript, object, ol, output, p, pre, progress, q, ruby, s, samp, script, section, select, small, span, strong, style, sub, sup, svg, table, template, textarea, time, u, ul, var, video, wbr, Text, picture
Sectioning Content	Header 요소와 Footer 요소의 범위 내에서 콘텐츠를 그룹화하여 구조
	article, aside, nav, section
Heading Content	Section 요소의 헤더(제목)를 정의
	h1, h2, h3, h4, h5, h6

HTML5의 문서구조


HTML 문서
<!DOCTYPE HTML>
선언문
<html>
<head>
시작 태그
HEAD요소
<meta “charset=UTF-8” >
<title> HTML5 기본 구조 </title>
</head>
종료 태그
HTML 요소
<body>
BODY요소
<!-- 문서의 내용 -->
설명문
 HTML5에 대하여 학습한다.
</body>
속성
속성 값
</html>

기존 HTML과 HTML5 비교

사용 예제

기존 HTML 방식

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<HTML lang="ko">
  <html>
  <head>
      <title> HTML5 기본 구조 </title>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <link rel="stylesheet"  href="path/to/stylesheet.css"  type="text/css" />
      <script type="text/javascript"  src="path/to/script.js"><script>
</head>
  <body>
   HTML5 기본 구조를 학습합니다.
  </body>
</HTML>
```

HTML5 방식

```
<!DOCTYPE HTML>
<HTML lang="ko">
  <html>
  <head>
      <title> HTML5 기본 구조 </title>
      <meta charset="UTF-8">
      <link rel="stylesheet"  href="path/to/stylesheet.css" />
      <script src="path/to/script.js"><script>
</head>
  <body>
   HTML5 기본 구조를 학습합니다.
  </body>
</HTML>
```

HTML5에 추가된 태그들

요소	설명
header	머리말(제목)을 나타낸다.
footer	제작자의 정보나 저작권의 정보를 나타낸다.
hgroup	제목과 부제목을 묶어주는 역할을 한다.
nav	메뉴 부분을 나타낸다.
section	문서 영역을 구성할 때 사용되고 H1~H6과 함께 사용하며 제목별로 나눌 수 있다.
article	개별 콘텐츠(뉴스 기사 또는 블로그의 내용 글과 같은)를 나타낸다.
aside	주요 콘텐츠 이외의 참고가 될 수 있는 좌우측의 콘텐츠를 나타낸다.

요소	설명
main	Body 영역의 주요 콘텐츠 블록을 그룹화할 때 사용한다.
figure	그림이나 비디오 같은 멀티미디어 요소를 나타낸다.
figcaption	Figure 요소에 대한 제목(개략적인 설명)을 나타낸다.
ficture	반응형 웹에 알맞은 이미지를 정의한다. (해상도에 가변적인)
ol (reversed 속성)	Li 요소들의 순서를 바꾸도록(앞의 순번이 뒤의 순번으로) 한다.

HTML5에 추가된 태그들

요소	설명
a (download 속성)	href 속성에 지정된 파일을 다운로드 할 수 있도록 한다.
a (ping 속성)	링크를 클릭했을 때, 링크의 정보를 자동으로 송신할 수 있도록 한다.
strong	아주 중요하거나 심각하거나 긴급을 요하는 내용을 표시하도록 한다.
time	날짜와 시간 표현에 의미를 주고자 할 때 사용하며 날짜와 시간을 나타낸다.
mark	특정 텍스트의 강조 효과를(형광펜으로 칠한 것과 같은) 나타낸다.
ruby, rt, rp, rb, rtc	Ruby 언어를 표시할 때 사용한다.
bdi	텍스트의 방향을 격리하여 나타나도록 한다.
wbr	줄 바꿈 지점을 나타낸다.

요소	설명
details	사용자 요청에 따라 얻은 콘트롤이나 추가적인 정보를 나타낸다.
summary	Details요소의 하위 요소로써 머리글을 나타낸다.
menu	명령들의 목록을 정의하고 명령들의 단축 메뉴 같은 것들을 목록화 한다
command	사용자가 호출할 수 있는 명령어를 나타낸다.
dialog	대화를 의미 있는 콘텐츠로 만들고자 할 때 사용하며 대화상자 또는 창을 나타낸다.

<video>, <audio> 태그

☐ <video> 태그 속성

- **src**
  - 동영상 파일의 경로를 지정
- **poster**
  - 동영상이 화면에 나타나지 않을 때 대신 표시할 그림을 지정
- **preload**
  - 동영상이 백그라운드에서 다운로드 되어 재생 단추를 눌렀을 때 재생
- **autoplay**
  - 동영상 자동 재생
- **loop**
  - 반복 재생 횟수 지정
- **controls**
  - 동영상 화면에 컨트롤 기능 추가
- **width**
  - 동영상 화면 너비 지정
- **height**
  - 동영상 화면 높이 지정

Web Form

<input>에 추가된 요소들

- **<input type= "email" >**

  email example@domain.com

  - 이메일 주소 입력시 사용
  - 서버로 전송시 이메일 형식 자동 체크
- **<input type= "url" >**
  - 웹 사이트 주소 입력시 사용
- **<input type= "number" >**

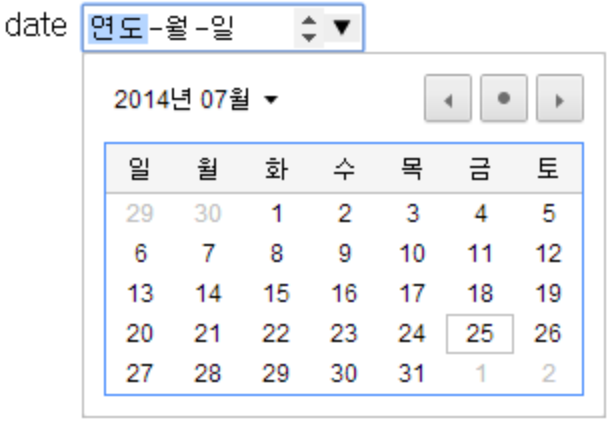
  number 1
  - 숫자를 스핀 박스를 이용해서 입력가능
  - min : 최소값, max, : 최대값, step : 간격, value : 초기값
- **<input type= "range" >**
  - 슬라이드 막대를 숫자 선택
  - min : 최소값, max, : 최대값, step : 간격, value : 초기값으로 생략시 중간에 위치.

  range
- **<input type= "search" >**
  - 검색 상자 삽입
  - 검색어 입력하면 오른쪽에 x가 표시됨

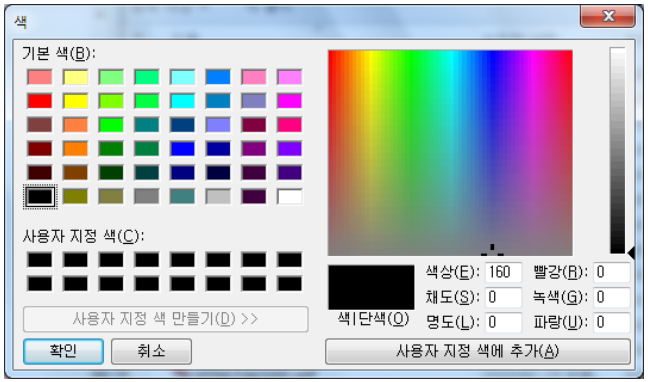
  search HTML5

Web Form

- **<input type= “date” >, <input type= “month” >, <input type= “week” >, <input type= “time” >**
  - 달력에서 날짜를 선택하거나 스핀 박스에서 시간을 선택



- **<input type= “color” >**
  - 색상 선택 상자 표시

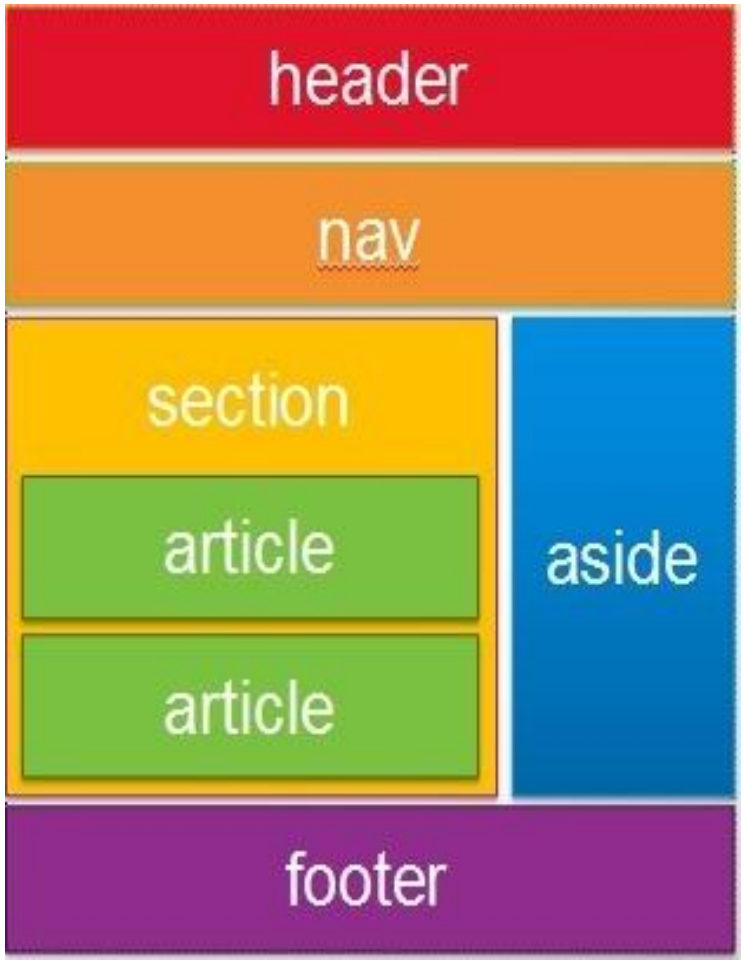


Web Form

☐ <input> 태그의 새로운 속성들

- **autocomplete**
  - 자동완성제어기능
  - 기본값 : on
- **autofocus**
  - 웹페이지 로딩 완료시 자동으로 포커스 이동
- **입력 값 제한**
  - min, max, step
  - <input> 태그의 유형이 date, month, week, time, number, range일 경우 사용 가능
- **placeholder**
  - <input> 태그의 필드 안에 적당한 힌트 내용 표시
  - 실제로 값을 입력하면 힌트표시는 자동으로 사라짐
- **required**
  - 서버로 폼을 전송하기 전에 필수 필드에 내용들이 모두 채워졌는지를 검사
- **multiple**
  - file 타입의 <input> 태그의 속성으로 사용 가능
  - 다중 파일 업로드를 처리하려는 경우 사용되는 속성

HTML5에서 문서의 구조를 정의하는 태그들



<header>
<header>는 주로 머리말, 제목을 표현하기 위해 쓰인다.

<nav>
네비게이션이라고 불린다. 콘텐츠를 담고 있는 문서를 사이트간에 서로 연결하는 링크의 역활 을 담당한다. 주로 메뉴에 사용되고 위치에 영향을 받지 않아 어디에서든 사용이 가능하다.

<section>
<body>영역은 콘텐트를 <Header>,<section>,<footer>의 3가지 공간에 저장하는데 그 중 <section>은 본문 콘텐트를 담고 있다. <section>안에 <section>을 넣는 것도 가능하다.

<article>
<section>이 콘텐트를 분류한다면 <article>태그 안에는 실질적인 내용을 넣는다. 뉴스로 예를 들면 정치/ 연예/ 사회의 대분류는 <section>이고, 정치의 기사내용과 연예의 기사내용들을 <article>에 넣는 것이다.

<aside>
사이드바라고 부르기도 하며, 본문 이외의 내용을 담고 있는 시맨틱 태그입니다. 주로 본문 옆에 광고를 달거나 링크들을 이 공간에 넣어 표현한다.

<footer>
화면의 구조 중 제일 아래에 위치하고, 회사소개 / 저작권 / 약관 / 제작정보 들이 들어간다. 연락처는 <address>태그를 사용하여 표시한다.

** **
** 는 HTML5에 와서 글자나 사진등 콘텐트들을 묶어서 CSS 스타일을 적용시킬 때 사용한다.**

HTML5 문서 구조

□ 기존 레이아웃 방식

- 모든 레이아웃 영역을 ** ** 태그를 사용하므로 세부적인 구별이 어려움
- ** ** 태그의 **id** 속성값으로 의미를 표시하거나 **class** 속성값으로 의미를 표현

□ 시맨틱 레이아웃 방식

- 레이아웃 영역을 시맨틱 태그를 이용하여 구분
- ** ** 태그 대신 여러 시맨틱 태그로 변경하여 표시
- 아이디(또는 클래스) 이름들을 표준 시맨틱 태그로 정의함으로써 문서의 의미 구조를 명확하고 간결하게 표현하도록 개선

```
<!-- 기존 레이아웃 방식 -->
  <body>
    <div id="header">   . . .  </div>
    <div id="nav">      . . .  </div>
    <div id="sidebar">  . . .  </div>
    <div id="section1"> . . .
      <div id="article"> . . . </div>
    </div>
    <div id="section2"> . . .
      <div id="section2_1">
          . . .
      </div>
    </div>
    <div id="footer">   . . .  </div>
   </body>
```

```
<!-- 시맨틱 레이아웃 방식 -->
  <body>
    <header>   . . .   </header>
    <nav>      . . .   </nav>
    <aside>    . . .   </aside>
    <section id="section1">    . . .
      <article>   . . .   </article>
    </section>
    <section id="section2">    . . .
      <section id="section2_1">
          . . .
      </section>
    </section>
    <footer>  . . .   </footer>
   </body>
```

contenteditable 속성

☐ contenteditable 속성

- 요소**(Element)**를 편집 모드로 변경하는 기능을 제공한다**.**
- 설정값
  - true : 요소 편집 가능 상태
  - false : 요소 편집 불가능 상태
  - inherit : 부모 요소의 값 상속. 즉, 부모가 편집 가능하다면 이 요소도 편집 가능(디폴트)

☐ 예제(contenteditable.html)

HTML5는...

- 더 풍부한 웹 애플리케이션
  - 비디오/오디오 재생을 위한 요소(video/audio), 2D, 3D 그래픽 처리(canvas)
  - 오프라인에서도 작동되는 애플리케이션(Application Cache)
  - 도메인 간의 통신 구현(Cross Document Messaging, XMLHttpRequest Level2)
  - 클라이언트 측에 데이터 저장(Web Storage, Indexed DB)
  - 백그라운드 처리 수행(Web Workers)
  - 서버로부터의 데이터 푸시나 서버와의 양방향 통신 수행
  - (Server-Sent Events, Web Sockets)
  - 로컬 파일의 내용을 읽어들임(File API) <-- 서버가 아닌 클라이언트에서 가능
- 더 시맨틱한 마크업
  - 기계가 분석해서 해석 가능 <= 문서의 의미를 부여하는 마크업 언어 추가됨
  - 디자인영역 구분하기 위한 태그 사용 → 각 용도별 의미있는 태그
  - 문서 구조의 의미나 문서 안에 삽입된 데이터의 의미 등을 명확히 하기 위한 사양이 다수 포함되어 시맨틱 웹의 구현

    <head>, <section>, <footer>, <article>, <nav>, <address>, <time>...

Cascade Style Sheet

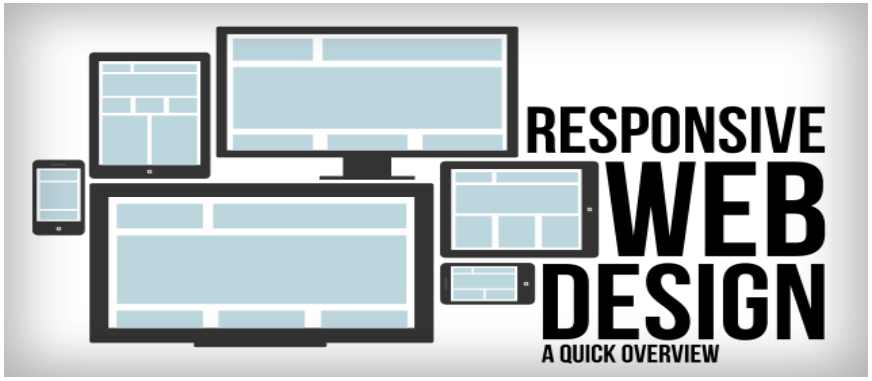
☐ CSS(Cascading Style Sheets) 란?

- 구조적으로 짜여진 문서**(HTML,XML)**에 **Style(**글자,여백,레이아웃**)**을 적용하기 위해 사용하는 언어**(Language)**이다**.**
- **CSS** 스타일시트는 **HTML** 문서의 요소에 적용되는 **CSS** 스타일 정의를 포함하며 **CSS**
- 스타일은 요소 표시 방법 및 페이지에서의 요소 위치를 지정한다**.**
- **W3C**의 표준이며 **HTML**구조는 그대로 두고 **CSS** 파일만 변경해도 전혀 다른 웹사이트처럼 꾸밀 수 있다**.**

Cascade Style Sheet

CSS(Cascading Style Sheets) 사용한 웹 페이지 개발

- 웹 표준에 기반한 웹 사이트를 개발할 수 있다. (페이지의 내용과 디자인을 분리)
- 클라이언트 기기에 알맞는 반응형 웹 페이지를 개발 할 수 있다.
- 이미지의 사용을 최소화시켜 가벼운 웹 페이지 개발을 가능하게 한다.



Cascade Style Sheet

□ CSS 사용의 이점

- 확장성 : 표현을 더욱 다양하게 확장하거나 표현 기능의 변경 가능
- 편의성 : 훨씬 간편하게 레이아웃 등의 스타일 구성
- 재사용성 : 독립된 스타일 모듈 작성, 여러 **HTML** 문서에 공통으로 활용
- 생산성 : 역할 분담에 따른 전문화, 모듈 단위의 협업과 생산성의 향상

□ CSS 의 역사

- **CSS1**
  - 첫 **CSS** 규격은 공식 **W3C** 권고안이 되었으며 그 이름은 **CSS1**이다.
  - **1996**년 **12**월에 출시되었다.
- **CSS2**
  - **W3C**가 개발하였으며 **1998**년 **5**월에 권고안으로 출시되었다.
- **CSS3**
  - **2005**년 **12**월 **5**일 이후 개발 중에 있다.
  - 모듈 기반으로 개발이 진행중이며 선택적으로 지원할 수 있다.
  - **CSS3**의 경우 그림자 효과, 그라데이션, 변형 등 그래픽 편집 프로그램으로 제작한 이미지를 대체할 수 있는 기능이 추가되었다. 또한 다양한 애니메이션 기능이 추가되어 플래시를 어느정도 대체하고 있다.
  - 재 모든 브라우저가 **CSS3**를 완벽하게 지원하는 것은 아니다. 일부 기능은 브라우저에 따라 지원 방식이 달라 별도의 접두어(밴더프리픽스)를 붙여야만 사용할 수 있다.

Cascade Style Sheet

☐ CSS 의 작성 방법

- 인라인 방법 – **HTML** 엘리먼트에 **style** 이라는 속성으로 정의하는 방법

  <tag style="property: value">

- 전역적 방법 – **<style>** 이라는 태그에 웹 페이지의 태그들에 대한 스타일을 정의하는 방법

  <style type="text/css">
  selector {property: value;}
  </style>

- 외부 파일 연결 방법 – 독립된 파일(확장자 **.css**)을 만들어서 **HTML** 문서에 연결하는 방법

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

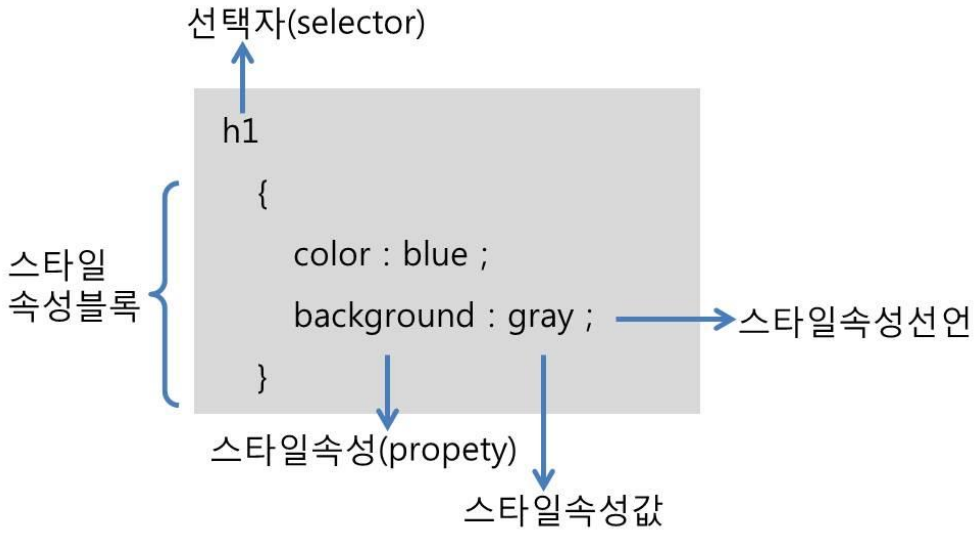
[CSS 스타일 선언 형식 **]**

– 선택자**(selector)**

스타일을 적용할 대상을 지정

– 스타일 속성**(property)** 블록

선택자에 의해 선택된 영역에 적용할 색상, 크기 등의 스타일을 명세로 중괄호**({ })** 로 둘러싸며 세미콜론**(;)** 으로 구분하여 하나 이상의 스타일 속성 선언을 포함



Cascade Style Sheet

☐ CSS3 소개

- **CSS2**와 **CSS3**의 가장 큰 차이점은 **CSS3**가 모듈 기반으로 개발되고 있다는 점이다. 이것은 각종 브라우저나 디바이스가 필요에 따라 원하는 **CSS** 모듈만을 탑재하거나 또는 필요한 모듈만을 빠르게 자주 업데이트 하는 것을 돕는다.
- **CSS3**는 **text, fonts, color, backgrounds & borders, transforms, transitions, animaitons**와 같은 종류의 모듈들을 추가로 개발하고 있다.
- **CSS3**는 기존의 **CSS2**가 갖지 못했던 화려하고 역동적인 면모를 추가하여 포토샵과 **JavaScript** 및 서버측 기술에만 완전히 의존하던 영역들을 개척했다.
- 상자의 크기에 따른 말줄임 표시, 투명한 배경, 그림자 효과, 둥근 모서리, 그라디언트, 도형의 회전과 비틀기, 애니메이션 효과 등이 가능해진 것이다. 특히 그래픽 디자인에만 의존하던 영역이 **CSS3**만으로도 상당부분 가능해지면서 웹 사이트의 성능 향상에 크게 기여할 수 있게 되었다.

[CSS3의 주요 기능 **]**

- 구조 선택자 지원
- 가상 선택자 지원
- 둥근 모서리의 경계선 지원
- 그라데이션 배경 지원
- 변환, 전환, 애니메이션 지원
- 글자와 박스 뒤에 그림자 효과 지원

Cascade Style Sheet

CSS 선택자(selector)란

- 스타일을 적용하기 위해 대상을 선택하는 방법

전체 선택자

- 페이지에 있는 모든 요소를 대상으로 스타일을 적용할 때 사용
- 다른 선택자와 함께 모든 하위 요소에 한꺼번에 스타일을 적용하려고 할 때 주로 사용

예) *** { margin:0; padding:0;}**

태그 선택자

- 문서 안의 특정 태그에 스타일이 모두 적용됨

예) **p { font-size:12px; font-family:** "돋음"**; }**

클래스 선택자 :

- 문서 안에서 여러 번 반복할 스타일이면 클래스 선택자로 정의하며 **.** 뒤에 클래스 이름 지정

예) **.redtext { color:red; }**

id 선택자

- 문서 안에서 한번만 사용한다면 **id** 선택자로 정의하며 파운드**(#)** 다음에 **id** 이름 지정

예) **#pic2 { clear:both; float:left; }**

Cascade Style Sheet

☐ 하위 선택자(descendant selector)

- 부모 요소에 포함된 모든 하위 요소에 스타일이 적용된다.
- 하위 선택자를 정의할 때는 상위 요소와 하위 요소를 나란히 작성한다.

예) **section p { color:blue; }**

☐ 자식 선택자(child selector)

- 부모 요소의 자식 요소에만 스타일이 적용된다.

예) **section > p { color:blue; }**

☐ 인접 형제 선택자(adjacent selector)

- 문서 구조상 같은 부모를 가진 형제 요소 중 첫 번째 동생 요소에만 스타일이 적용된다.
- 같은 부모 요소를 가지는 요소들을 형제 관계라고 부른다. 먼저 나오는 요소를 '형 요소', 나중에 나오는 요소를 '동생 요소'라고 한다.

예) **h1 + p { text-decoration : underline; }**

☐ 형제 선택자(sibling selector)

- 형제 요소들 중에서 모든 동생 요소들에 스타일이 적용된다.

예) **h1~ p { text-decoration : underline; }**

Cascade Style Sheet

□ 그룹 선택자(descendant selector)

- 같은 속성을 적용해야 할 경우 똑같은 스타일을 두번 정의하지 않고 한번에 묶어서 정의한다.
- 쉼표로 선택자 구분

예) **a, p { color: #fffff; }**

□ 속성 선택자(property selector)

- 태그에 정의된 속성과 속성의 값을 가지고 대상을 정하는 선택자이다.

표기	설명
[속성]	지정한 '속성'을 가지고 있는 요소를 찾아 스타일을 적용
[속성 ~= 값]	'속성'과 '값'을 체크해 여러 개의 값 중 하나만 일치해도 스타일을 적용
[속성 ^= 값]	'속성'의 '값'이 지정한 문자로 시작하는 속성값에 대해서만 스타일을 적용
[속성 $= 값]	속성'의 '값'이 지정한 문자로 끝나는 속성에 대해서만 스타일을 적용
[속성 *= 값]	속성 값 중에 '값'의 일부가 포함되어 있는 속성에 스타일을 지정

Cascade Style Sheet

☐ 가상 선택자(pseudo selector)

- 웹 문서의 소스에는 실제로 존재하지 않지만 필요에 의해 임의로 가상의 선택자를 지정하여 사용하는 것을 말한다.

표기	설명
:first-letter	첫 번째 문자에 스타일을 적용한다.
:first-line	첫 번째 행에 스타일을 적용한다.
:first-child	첫 번째 자식 요소에 스타일을 적용한다.
:last-child	마지막 자식 요소에 스타일을 적용한다.
:before	특정 요소의 내용 앞에 지정한 내용을 만든다.
:after	특정 요소의 내용 뒤에 지정한 내용을 만든다.
:hover	사용자가 대상 요소를 가리키고 있을 때 스타일을 적용한다.롤 오버 등
:focus	대상 요소가 포커스 되었을 때 스타일을 적용한다.

Cascade Style Sheet

☐ 주요 CSS 속성들

GENERAL	
Class	String preceded by a period
ID	String preceded by a hash mark
div	Formats structure or block of text
span	Inline formatting
color	Foreground color
cursor	Appearance of the cursor
display	block; inline; list-item; none
overflow	How content overflowing its box is handled visible, hidden, scroll, auto
visibility	visible, hidden

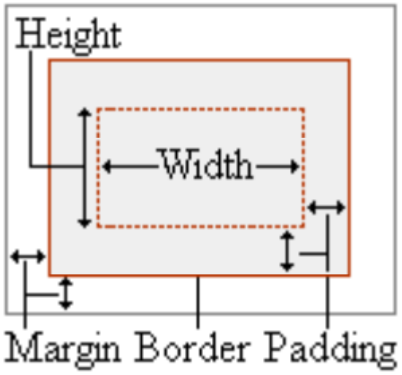
TEXT	
letter-spacing	Space between letters
line-height	Vertical distance between baselines
text-align	Horizontal alignment
text-decoration	blink, line-through, none, overline, underline
text-indent	First line indentation
text-transform	capitalize, lowercase, uppercase
vertical-align	Vertical alignment
word-spacing	Spacing between words

FONT	
font-style	Italic, normal
font-variant	normal, small-caps
font-weight	bold, normal, lighter, bolder, integer (100-900)
font-size	Size of the font
font-family	Specific font(s) to be used

Cascade Style Sheet

☐ 주요 CSS 속성들

BOX MODEL



height; width; margin-top; margin-right; margin-bottom; margin-left; padding-top; padding-right; padding-bottom; padding-left;

BORDER

border-width	Width of the border
border-style	dashed; dotted; double; groove; inset; outset; ridge; solid; none
border-color	Color of the border

BACKGROUND

background-color	Background color
background-image	Background image
background-repeat	repeat, no-repeat, repeat-x, repeat-y
background-attachment	Background image scroll with the element? scroll, fixed
background-position	(x y), top, center, bottom, left, right

Cascade Style Sheet

☐ 주요 CSS 속성들

POSITION	
clear	Any floating elements around the element? both, left, right, none
float	Floats to a specified side left, right, none
left	The left position of an element auto, length values (pt, in, cm, px)
top	The top position of an element auto, length values (pt, in, cm, px)
position	static, relative, absolute
z-index	Element above or below overlapping elements? auto, integer (higher numbers on top)

LIST	
list-style-type	Type of bullet or numbering in the list disc; circle; square; decimal; lower-roman; upper-roman; lower-alpha; upper-alpha; none
list-style-position	Position of the bullet or number in a list inside; outside
list-style-image	Image to be used as the bullet in a list

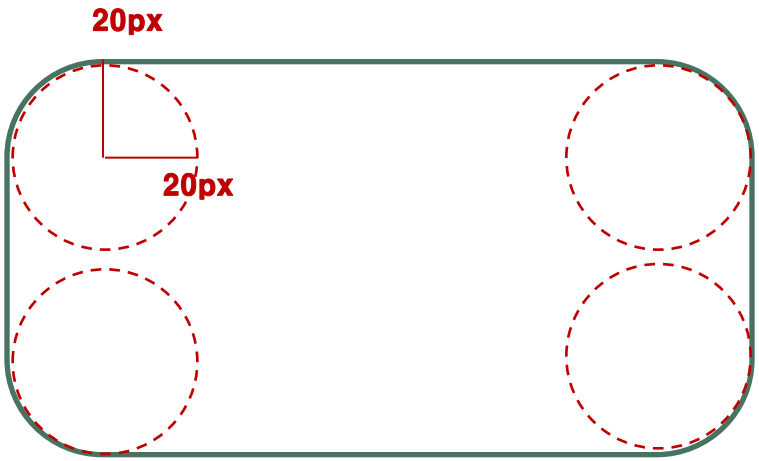
Cascade Style Sheet

CSS3에서 추기된 속성들

- 새로운 **CSS3** 속성을 부분적으로만 지원할 수 있기 때문에 대부분의 **CSS3** 프로퍼티를 사용하려면 프로퍼티 앞에 브라우저를 식별할 수 있는 접두어를 붙여야 하는 것도 있다.

-webkit-	사파리, 크롬 등
-moz-	모질라, 파이어폭스 등
-o-	오페라
-ms-	인터넷 익스플로러

- **border-radius**




```
div.rounded {
    background-color: #666;
    color: #fff;
    width: 400px;
    padding: 10px;
    -webkit-border-radius: 20px;
    -moz-border-radius: 20px;
    border-radius: 20px;
}
```

Cascade Style Sheet

CSS3에서 추기된 속성들

- **box-shadow**

가로 오프셋	양수값은 오른쪽, 음수값은 왼쪽에 그림자 생김
세로 오프셋	양수값은 아래쪽, 음수값은 위쪽에 그림자 생김
blur radius	그림자의 번지는 정도. 0이 최소값.
그림자 색상	16진수나 RGB값, 색상 이름 모두 사용 가능

```
<style type="text/css">
a img:hover {
    box-shadow: 0 5px 10px rgba(0,0,0,0.6);
    -moz-box-shadow:0 5px 10px rgba(0,0,0,0.6);
    -webkit-box-shadow:0 5px 10px rgba(0,0,0,0.6);
}
</style>
```

Cascade Style Sheet

CSS3에서 추기된 속성들

- **text-shadow**

가로 오프셋	양수값은 오른쪽, 음수값은 왼쪽에 그림자 생김
세로 오프셋	양수값은 아래쪽, 음수값은 위쪽에 그림자 생김
blur radius	그림자의 번지는 정도. 0이 최소값.
그림자 색상	16진수나 RGB값, 색상 이름 모두 사용 가능

```
<style type="text/css">
.text1 {
    color:#06F;
    text-shadow:3px 3px 5px #000;
}
.text2{
    color:#C30;
    text-shadow:3px -3px 5px #000;
}
</style>
```

HTML5 & CSS3

HTML5 & CSS3

Cascade Style Sheet

CSS3에서 추기된 속성들

- **text-shadow**

```
<style>
body {
    background:#066;
    font-family: Arial Black;
    font-size:50px;
}
.text1 {
    color:#fff;
    text-shadow:0px 1px 8px #fff;
}
.text2 {
    color:#fff;
    text-shadow:1px -3px 8px
#6F0;
}
.text3 {
    color:#000;
    text-shadow:1px 1px 4px #fff;
}
</style>
```


```
<style>
body {
    background:#000;
    font-family: Arial Black;
    font-size:50px;
}
.text3 {
    color:#000;
    text-shadow:0 0 4px #ccc,
              0 -5px 4px #ff3,
              2px -10px 6px #fd3,
             -2px -15px 11px #f80,
              2px -19px 18px #f20;
}
</style>
```

Cascade Style Sheet

CSS3에서 추기된 속성들

- **gradient** 속성값

  background : linear-gradient(direction, color1, color2, …, color3);

to bottom
위에서 아래로(기본값)
to top
아래에서 위로
to left
오른쪽에서 왼쪽
to right
왼쪽에서 오른쪽
ndeg
n도의 방향

Cascade Style Sheet

☐ CSS3에서 추기된 속성들

- **opacity** 속성

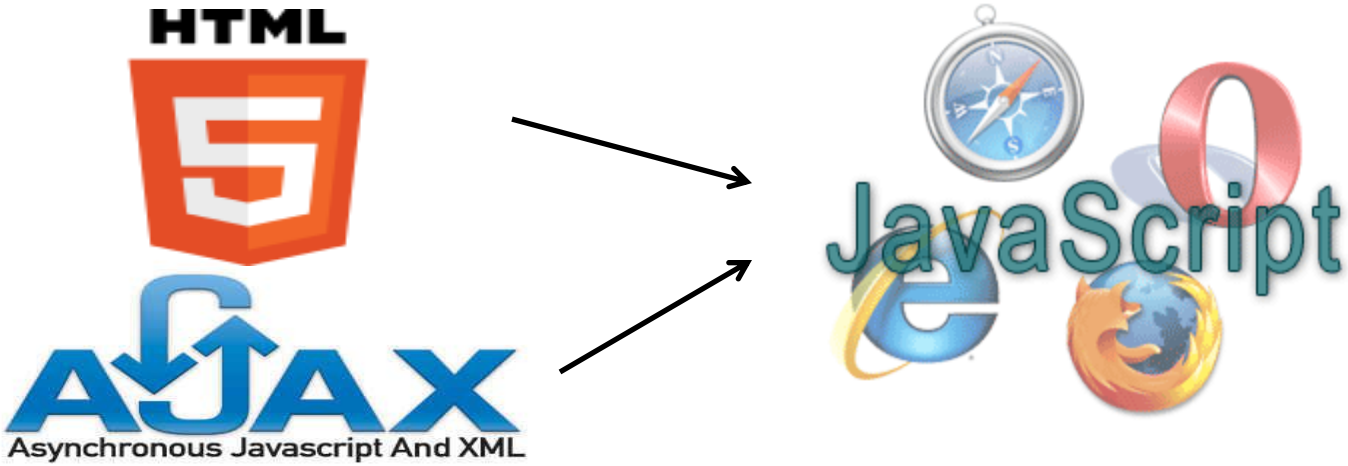
  칼라나 이미지의 투명도를 설정하는 속성으로 **0.0 ~ 1.0** 사이의 값을 설정한다.

```
<style>
.noopa {
    background-color:#fff;
    color:#000;
}
.opa {
    background-color:#fff;
    opacity:0.5;
    color:#000;
}
</style>
```

JavaScript 소개

JavaScript 란

- **JavaScript**는 넷스케이프 커뮤니케이션즈 코퍼레이션의 브렌던 아이크**(Brendan Eich)**가 처음에는 모카**(Mocha)**라는 이름으로, 나중에는 라이브스크립트**(LiveScript)**라는 이름으로 개발하였으며, 최종적으로 **JavaScript**라는 이름으로 발표되었다.
- **JavaScript**는 객체 기반의 스크립트 프로그래밍 언어이다. 이 언어는 웹브라우저 내에서 주로 사용한다.
- 프로그래밍 언어로서 저평가 받는 시기도 있었으나 리치 콘텐츠**(Rich Content)**를 작성할 수 있는 **AJAX(Asynchronous JavaScript + XML)**의 등장으로 인해 **JavaScript**의 가치는 재검토되었다.
- **HTML5**에서 **HTML5**의 **API** 로 **JavaScript**를 공식 채택함으로써 **JavaScript**는 세계에서 가장 인기 있는 프로그래밍 언어 중 하나로 자리 잡아가고 있다.



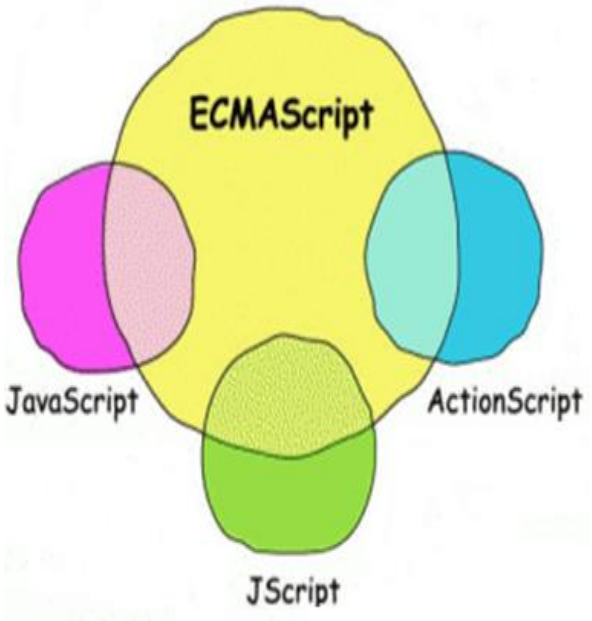
ECMAScript란?

- 웹의 발전과 함께 브라우저 회사들의 과도한 경쟁으로 크로스 브라우징 구현이 어려웠으나 국제적인 표준화단체인 ECMA 에서 착실하게 표준화를 진행하여 언어로서의 완성도를 높여 나갔다.

- ECMAScript 는 Ecma 인터내셔널의 ECMA-262 기술 규격에 정의된 스크립트 프로그래밍 언어의 표준화된 스펙이다.

- 2015년 ECMAScript 6판이 발표되었다.

- 현재 대부분의 브라우저들이 지원하는 JavaScript는 ECMAScript 6 표준을 따르고 있다.



ECMA(European Computer Manufacturer Association: 유럽 전자계산기 공업회)

JavaScript의 활용범위

□웹 클라이언트 개발

웹이 발전하면서 서버에서 처리되던 많은 기능들이 클라이언트로 이동되었으며, HTML5에서는 웹 클라이언트에서 처리하려는 기능들을 표준적인 방법으로 구현할 수 있게 지원하는 API들을 JavaScript로 제공한다. 그러므로 웹 클라이언트에서는 JavaScript 의 필요성은 더욱 강조되고 있다.

□웹 서버 개발

Node.js의 출현으로 JavaScript를 활용한 서버 개발도 가능하게 되었다. express, soicket.io등의 라이브러리는 보다 쉽게 JavaScript 로 서버를 개발할 수 있는 환경을 제공해 준다.

□어플리케이션 개발

웹이 하나의 플랫폼으로 진화하면서, 웹 OS를 표방한 여러 가지 프로젝트가 진행되고 있다. 구글에서는 크롬OS라는 브라우저 기반의 OS를 선보였고, 모바일에서도 HP에서 웹OS라는 이름으로 휴대폰과 같은 모바일 기기에도 웹 기반의 각종 기술이 활용되고 있다. 이러한 웹 기반 플랫폼에서 구동되는 애플리케이션 개발에 JavaScript는 없어서는 안 될 핵심 언어가 되었다. 향후 완벽한 웹OS가 출현한다면 JavaScript의 중요성은 더욱 커질 전망이다.

JavaScript의 정의방법

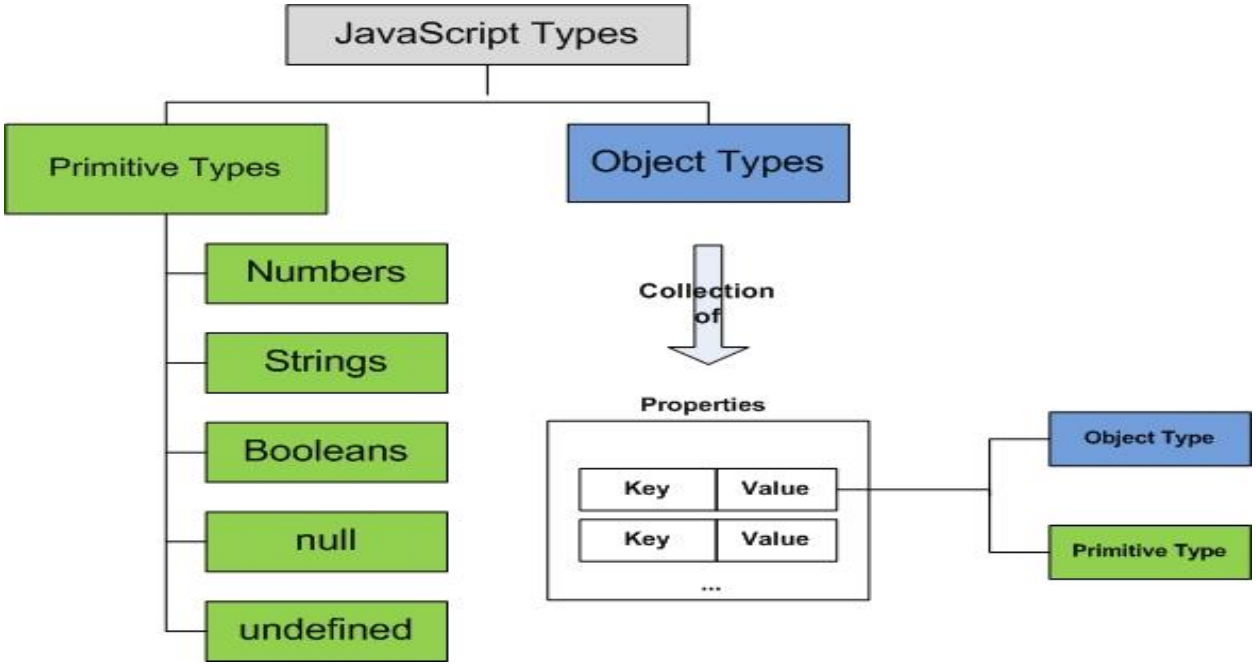
JavaScript 코드는 HTML 내의 어느 부분에 삽입해도 가능하나 주로 필요에 따라 <head> 태그나 <body> 태그에 삽입하게 되는데 헤더 부분에 위치한 JavaScript 코드는 컨텐트 (<body> 태그)내용의 랜더링 이전에 브라우저에 의해 먼저 해석되어 랜더링을 지연시키는 결과가 될 수도 있다.

이벤트 핸들러 기능의 JavaScript 코드는 가급적 <body 태그의 마지막 부분 즉, **</body> 태그의 바로 위에 삽입**한다.

```
<!DOCTYPE html>
<html>
  <head>
    <title>script tag의 위치</title>
    <!-- script 태그의 위치 1 -->
    <script>
      alert("테스트1");
    </script>
  </head>
  <body>
    즐거운 저녁입니다.
    <!-- script 태그의 위치 2 -->
    <script>
       alert("테스트2");
    </script>
  </body>
</html>
```

JavaScript의 데이터 타입

- JavaScript는 데이터 타입이 number, string, boolean, null 그리고 undefined 로 구분되는 기본형 타입과 객체 타입으로 나뉜다.
- 숫자 타입 : 100, 3.14
- 문자열 타입 : "가나다", 'abc'
- 논리 타입 : true, false



JavaScript의 주요 연산자

- 수치 연산자

  덧셈(+), 뺄셈(-), 곱셈(*), 나눗셈(/), 나머지(%), 중가 연산자(++,--), 단항 연산자(-)

  문자열 연산자 + : 문자열을 합하여 하나의 문자열 생성

  str = "ABCD" +"1234"; ==> "ABCD1234"

- 비교 연산자

  <,>,<=, <=, ==, ===, !=, !==

- 조건 연산자

  AND 연산자(&&), OR연산자(||), NOT 연산자(!), ? 연산자

- 대입 연산자

  =, += -=, *=, /=, %=

- 비트 연산자

  비트 AND(&), 비트 OR(|),비트 XOR(^), 비트 좌우 이동(<<,>>)

- 타입 점검 연산자

  typeof, instanceof

- 삭제 연산자

  delete

JavaScript의 제어문

- **조건 제어문 if, 다중 분기문 switch**
  switch 문에 사용되는 비교식에 데이터 타입의 제한이 없다.

- **반복 제어문 for, while, do-while**
  for…in 반복문 사용이 가능하다(for-each 문이라고도 한다.)
  for…in 명령은 지정된 배열이나 객체 내의 요소/멤버에 대해 선두부터 마지막까지 순서대로
  반복 문장을 수행한다.

- **분기 제어문 break, continue**
  중첩된 반복문에서 사용될 때 레이블을 사용하여 외부 반복문에 대한 제어가 가능하다.

- **예외처리 구문이 지원된다.**
  try – catch – finally 구문을 사용하여 실행 오류 발생시의 대비 코드 구현이 가능하다.

JavaScript의 배열 정의와 활용

☐ JavaScript 배열의 특징과 정의 방법

- **JavaScript** 배열의 특징
  - 객체로 취급된다.
  - 배열을 구성하는 각 데이터들을 요소라고 한다.
  - 배열의 요소 개수를 가변적으로 처리할 수 있다. 배열을 생성할 때 크기를 지정하더라도 필요하다면 배열을 구성하는 요소의 개수를 늘리는 것이 가능하다.
  - 배열에 저장할 수 있는 데이터의 타입에 제한이 없다.
    배열을 구성하는 각 요소마다 다른 타입의 데이터를 저장하고 사용하는 것이 가능하다.
  - **length** 라는 속성을 사용하여 배열을 구성하고 있는 요소의 개수를 추출할 수 있다.
  - 배열을 생성하여 변수에 담아 사용한다.

- **JavaScript**의 배열 생성 방법은 **2**가지 방법이 지원된다.
  - 배열 리터럴을 사용하는 방법**(**자동으로 배열 객체가 된다.**)**
    [1, 2, 3, 4, 5]
  - **Array()** 라는 생성자 함수를 호출하여 배열 객체를 생성하는 방법
    new Array(10)

JavaScript의 배열 정의와 활용

□ 배열의 활용

```
var array_example1 = new Array( "hello", "world" );
var array_example2 = [ "hello", "world" ];
var array_example3 = [];
array_example3.push( "5" );
array_example3.push( "7" );
array_example3[ 2 ] = "2";
array_example3[ 3 ] = "12";
var array_example4 = [];
array_example4.push( 0 ); // [ 0 ]
array_example4.push( 2 ); // [ 0 , 2 ]
array_example4.push( 7 ); // [ 0 , 2 , 7 ]
array_example4.pop();     // [ 0 , 2 ]
var array_example5 = [ "world" , "hello" ];
// [ "hello", "world" ]
array_example5.reverse();
var array_example7 = [ 3, 4, 6, 1 ];
array_example7.sort(); // 1, 3, 4, 6
```

concat(ary)	**지정 배열을 현재의 배열에 추가**
join(del)	**배열 내의 요소들을 구분문자 del로 연결 해서 문자열 리턴**
slice(start [,end])	**start 부터 end-1번째까지의 요소들을 추출하여 배열 객체를 리턴**
pop()	**배열 끝의 요소를 취득하여 삭제**
push(data)	**배열 끝에 요소를 추가**
shift()	**배열 선두의 요소를 취득하여 삭제**
unshift(data,...)	**배열 선두에 지정 요소를 추가**
reverse()	**역순으로 정렬(반전)**
sort([fnc])	**요소를 오름차순으로 정렬**
toString()	**요소, 요소, ... 의 형식으로 문자열 리턴**

JavaScript의 함수 정의와 활용

JavaScript의 함수 정의 방법

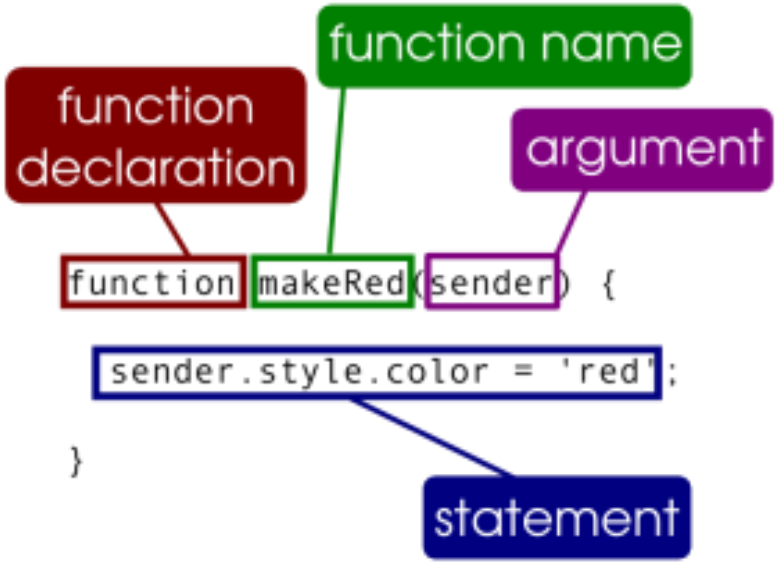
- 함수(**function**)란 하나의 로직을 재실행 할 수 있도록 하는 것으로 코드의 재사용성을 높여준다.

- 선언적(명시적) 함수 정의 방법

```
function myFunction([인자...[,인자]]) {
    /* do something */
}
```

- 표현식(익명) 함수 정의 방법

```
var myFunction = function([인자...[,인자]]) {
    /* do something */
}
```



JavaScript의 함수 정의와 활용

☐ JavaScript의 함수 의 다양한 활용

[함수의 정의와 호출 예1]

```
var msg = function( person, greeting ) {
    var text = greeting + ", " + person;
    alert( text );
};
msg("자바스크립트", "안녕하세요?" );
```

[함수의 정의와 호출 예2]

```
var msg = function( person, greeting ) {
    var text = greeting + ", " + person;
    return text;
};
alert(msg("자바스크립트", "안녕하세요?" ));
```

[함수의 정의와 호출 예3]

```
var myFn = function( fn ) {
    var result = fn();
    console.log( result );
};

myFn( function() {
    return "hello world";
});
```

[함수의 정의와 호출 예4]

```
var myFn = function( fn ) {
    var result = fn();
    console.log( result );
};

var myOtherFn = function() {
    return "hello world";
};

myFn( myOtherFn );
```

함수의 아규먼트 활용

- 함수의 호출과 인수(argument)
  - 함수 호출 시에는 함수의 매개변수 사양과 리턴 값의 존재여부를 파악하여 다음과 같이 다양한 형식으로 호출하는 것이 가능하다.

```
func1();  func4();
func2(100);  func5('안녕');
var result = func3();
```

  - JavaScript는 메서드 호출 시 전달되는 인수(argument)의 개수와 함수 정의 시 선언되어 있는 매개변수의 개수를 일치하는지 체크하지 않는다.

- 함수 호출시, 매개변수에 인수를 전달하지 않으면 이 매개 변수의 값은 **undefined**가 된다.
- 선언된 매개변수보다 더 많은 인수가 전달된 경우 매개변수를 통해서는 모든 인수를 추출할 수 없다. 전달된 인수를 모두 추출하려면 **arguments** 라는 함수의 내장 변수를 사용한다.
- arguments 변수는 객체 타입으로서 함수 호출 타이밍에 생성되고 전달되는 인수를 보관한다. length 라는 속성을 사용하여 전달된 인수의 개수를 추출할 수 있다. 이 때 함수에 선언되어 있는 실제 매개변수의 개수를 알고자 할 때는 **함수이름.length** 를 사용한다.

5. 함수의 활용

- 고계(고차) 함수(High-Order Function : **함수를 데이터로 다루는 함수**)
  - 함수 호출 시 인수로 또 다른 함수를 전달할 수 있으며 리턴 값으로 함수를 전달할 수도 있다. 이렇게 정의되는 함수를 고계(고차)함수라고 한다.
  - JavaScript에서의 함수는 데이터 타입의 한 종류이다. 즉, 함수 자체도 다른 수치 타입이나 문자열 타입 등과 같이 함수의 인수로 전달하거나 반환 값으로서 리턴할 수 있다.

```
function test1(callback) {
    if(callback != undefined)
            callback(3);
};

test1(function (num) {
    if( num % 2 == 1)
            alert(num + ': Odd');
    else
            alert(num + ': Even');
});
```

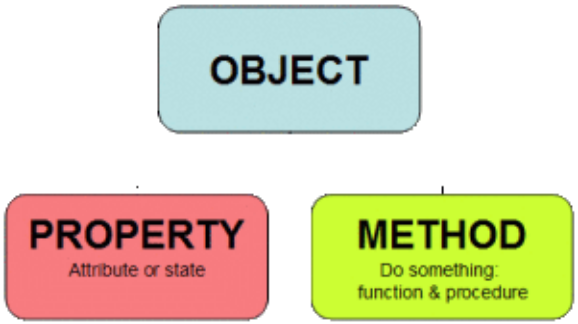
```
function test2() {
    return function(str) {
            alert(str + '!!');
    };
};

var result1 = test2();
result1();
result1();
var result2 = test2();
result2();
```

JavaScript의 객체 정의와 활용

☐ JavaScript의 객체의 특징

- 객체란 이름과 값을 가진 **data(Property)** 의 집합 및 **data** 를 조작하기 위한 **Method** 가 하나로 묶인 것이다.



- **JavaScript** 에서 객체는 **Property** 의 집합과 하나의 **prototype object** 을 가지고 있다 **.**
- **Method** 는 함수가 값으로 저장된 객체의 **Property** 로서**,** 객체의 속성을 취득 및 변경 하기 위한 창구이다**.** 객체의 프로퍼티에 할당되어 객체를 통해서 호출되는 함수를 메서드라 부른다**.**
- 객체의 속성과 메서드는 동적으로 추가하거나 삭제하는 것이 가능하다**.**
- 상속구문도 적용되어 **JavaScript** 에서 생성되는 모든 객체들은 조상 객체로 **Object** 객체를 갖는다**.**
- **JavaScript** 의 함수는 실행 가능한 코드와 연결된 객체라 할 수 있다**.**

JavaScript의 객체 정의와 활용

☐ JavaScript 객체의 정의 방법

JavaScript의 객체 생성 방법은 다음과 같이 **2**가지 지원된다.

- 객체 리터럴을 사용하는 방법
- 생성자 함수를 사용하는 방법

- **객체 리터럴을 사용하는 방법**

```
{
  속성명 : 속성값, 속성명 : 속성값, …
}
```

- 생성자 함수를 사용하는 방법

  생성자 함수란 객체를 초기화(속성과 메서드를 정의)하기 위해 사용되는 함수로서 관례적으로 생성자 함수의 명칭은 첫 글자를 대문자로 사용한다.

```
function 함수명([매개변수]) {
   this.속성명 = 값;
   this.속성명 = 값; …
}

new 함수명()
```

JavaScript의 객체 정의와 활용

JavaScript의 객체 의 다양한 활용

```
var person1 = new Object();
person1.firstName = "duke";
alert( person1.firstName );
alert( person1["firstName"] );
```

```
var person2 = {
    firstName: "duke",
    lastName: "java"
};
alert( person2.firstName + " " +
person2.lastName );
```

```
var people = { };
people[ "person3" ] = person1;
People.persion4 = persion2;
alert( people[ "person3" ].firstName );
alert( people.person3.firstName );
alert( people[ "person4" ].firstName );
alert( people.person4.firstName );
```

```
function Car (make, model, color) {
  this.make = make;
  this.model = model;
  this.color = color
  this.displayCar = displayCar;
}

function displayCar() {
     document.writeln("Make = " + this.make)
}

Var myCar = new Car ("Ford", "Focus", "Red");
myCar.displayCar();
myCar.make = "BMW";
myCar.displayCar();
```

객체의 생성과 활용

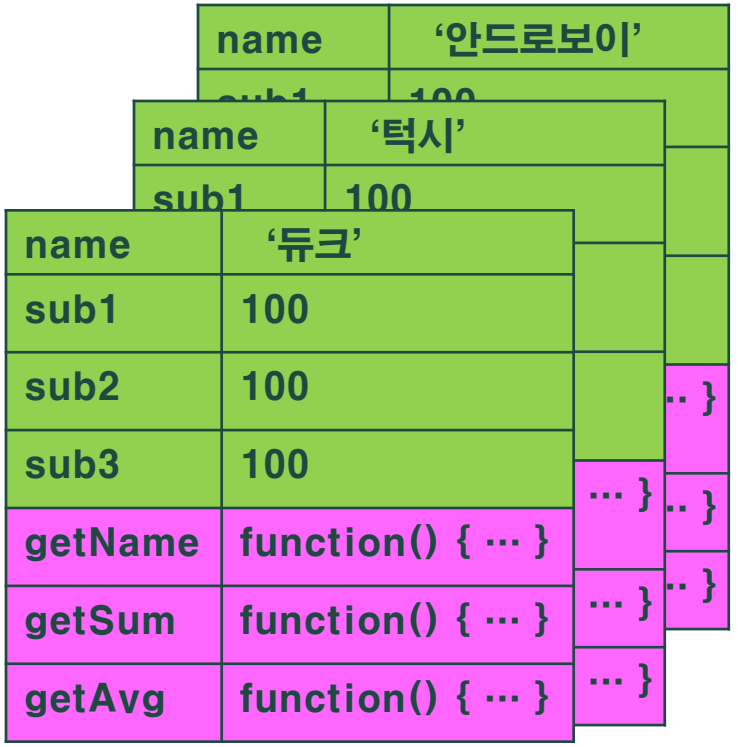
- 객체 리터럴 방식과 생성자 함수 방식의 비교

객체리터럴 방식	생성자 함수 방식
- 객체 리터럴 방식은 하나의 객체만을 만들 수 있게 된다. (싱글톤 객체) - prototype 속성 사용이 불가하다.	- 동일한 속성 사양을 갖는 객체들을 여러개 생성하는 것이 가능하다. - prototype 속성 사용이 가능하다. - 정적 멤버를 정의할 수 있다. - 캡슐화나 정적 멤버와 같은 OOP 구문을 적용하여 객체를 생성하는 것이 가능하다.

```
function Student(name, sub1, sub2, sub3)
   this.name = name;
   this.sub1 = sub1;
   this.sub2 = sub2;
   this.sub3 = sub3;
   this.getName = function() {
      return this.name;
   }
   this.geSum = function() {
      return this.sub1+this.sub2+this.sub3;
   }
   this.getAvg = function() {
      return this.getSum() / 3;
   }
 }

var st1 = new Student('듀크', 100, 100, 100);
var st2 = new Student('턱시', 90, 90, 90);
var st3 = new Student('안드로보이', 80, 80, 80);
                    :

}
```

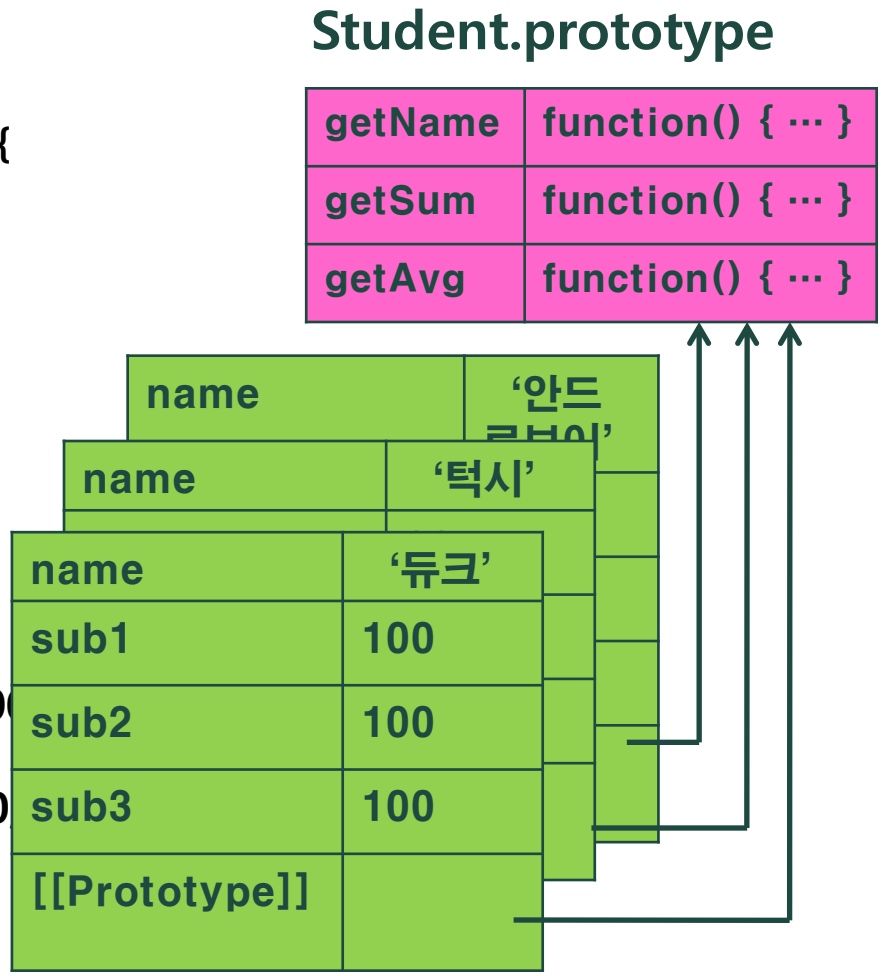


```
function Student(name, sub1, sub2, sub3)
   this.name = name;
   this.sub1 = sub1;
   this.sub2 = sub2;
   this.sub3 = sub3;
}

Student.prototype.getName = function() {
   return this.name;
}
Student.prototype.geSum = function() {
   return this.sub1+this.sub2+this.sub3;
}
Student.prototype.getAvg = function() {
   this.getSum() / 3;
}


var st1 = new Student('듀크', 100, 100, 10
var st2 = new Student('턱시', 90, 90, 90);
var st3 = new Student('안드로보이', 80, 80
                  :

}
```



this 객체의 활용

- this 객체 : **this 객체는 함수가 어떻게 실행되느냐에 따라 결정된다.**
  - 함수가 객체의 메서드로서 불리게 되면, this는 메소드를 부른 객체로 설정된다.
  - new 연산자를 통해 인스턴스(객체)를 생성해 함수를 부를 때도 같은 원리다. 이런 방법으로 부르게 되면 함수 스코프 내에서 this의 값은 새로 만들어진 객체로 설정된다.
  - 함수가 단독으로 호출되면, this는 기본적으로 브라우저의 window 객체가 된다.
  - 함수명으로 호출 가능한 call() 과 apply() 메서드를 사용해도 함수 호출이 가능하며 이 때는 함수가 어떤 컨텍스트로 실행할지 즉 함수내에서 사용되는 this 가 어떠한 객체를 참조하게 될지 자유롭게 설정해줄 수 있다.

JavaScript의 내장 객체

JavaScript 내장 객체들의 종류

- 표준 내장 객체

  JavaScript 언어 자체에 정의되어 있는 객체들이다.

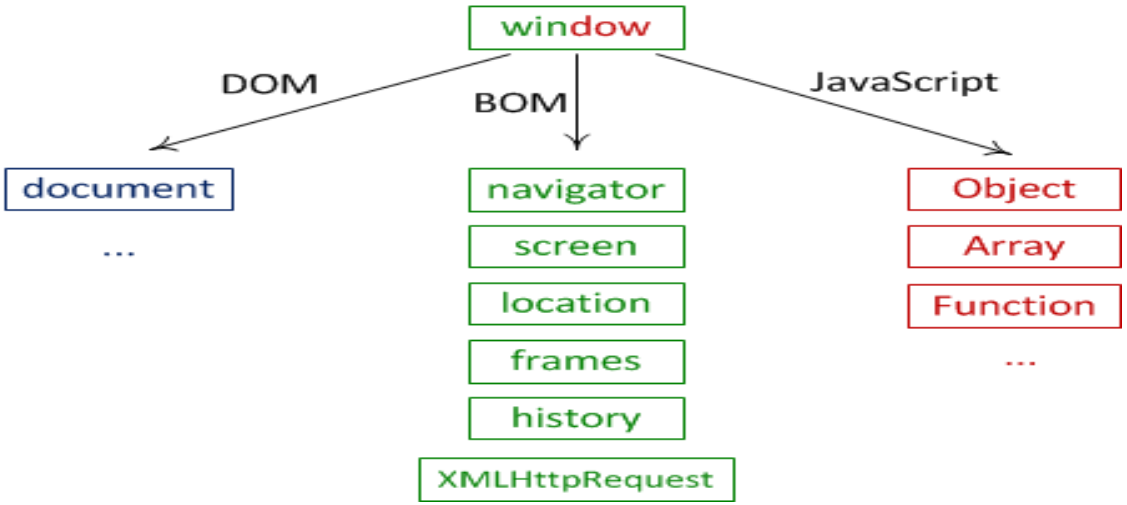
- **BOM**

  Browser Object Model. Web 페이지의 내용을 제외한 브라우저의 각종 요소들을 객체화시킨 것이다.

- **DOM**

  Document Object Model. Web 페이지의 내용을 제어한다. **window**의 프로퍼티인 **document** 프로퍼터에 할당된 **Document** 객체를 통해 사용한다.

  Document 객체의 프로퍼티는 문서 내의 주요 엘리먼트에 접근할 수 있는 객체를 제공한다.



JavaScript의 표준 내장 객체

□ 표준 내장 객체

- 표준 내장 객체(**Standard Built-in Object**)는 **JavaScript**가 기본적으로 가지고 있는 객체들을 의미한다.

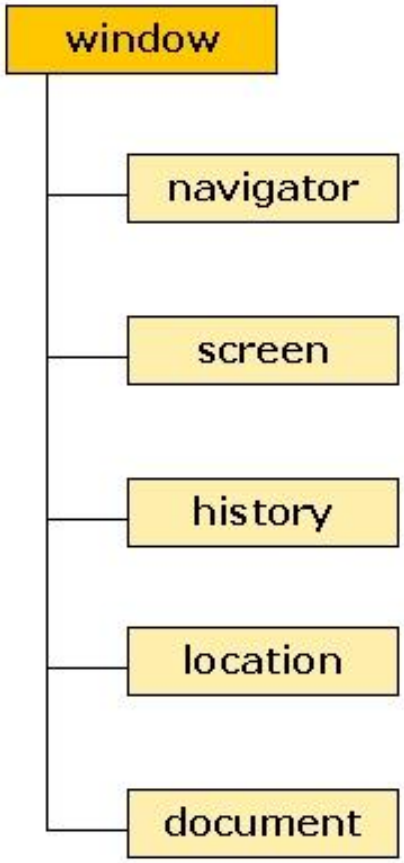
- 프로그래밍이라는 것은 언어와 호스트 환경에서 제공하는 기능들을 통해서 새로운 소프트웨어를 만들어내는 것이므로 내장 객체에 대한 이해는 프로그래밍의 기본이라고 할 수 있다.

- **JavaScript**는 아래와 같은 내장 객체를 가지고 있다.

  Object : 최상의 객체로서 **JavaScript**의 모든 객체들은 이 객체를 상속하게 된다.
  Function : 함수정의시 사용되는 객체이다.
  Array : 배열 정의시 생성되는 객체이다.
  String : 문자열 데이터에 대한 **Wrapper** 객체이다.
  Boolean : 대수형 값에 대한 **Wrapper** 객체이다.
  Number : 수치값에 대한 **Wrapper** 객체이다.
  Math : 다양한 수학 함수 기능을 제공하는 객체이다.
  Date : 날짜와 시간 정보 추출과 설정 관련 기능을 제공하는 객체이다.
  RegExp : 정규 표현식(패턴)을 이용하여 데이터를 처리하려는 경우 사용되는 객체이다.

JavaScript의 BOM객체

□ BOM 객체



- window : 최상위 객체로, 각 탭별, iframe 별로 하나씩 존재
- navigator : 브라우져(이름, 버전등)정보를 보관하는 객체
- document : 현재의 문서에 대한 정보를 보관하는 객체
- location : 현재 보여지고 있는 웹 페이지에 대한 URL 정보를 보관하는 객체
- history : 현재의 브라우저가 접근했던 URL의 정보를 보관하는 객체
- screen : 클라이언트 머신의 화면 크기나 해상도 등의 정보를 얻을 수 있는 객체

JavaScript의 DOM 객체

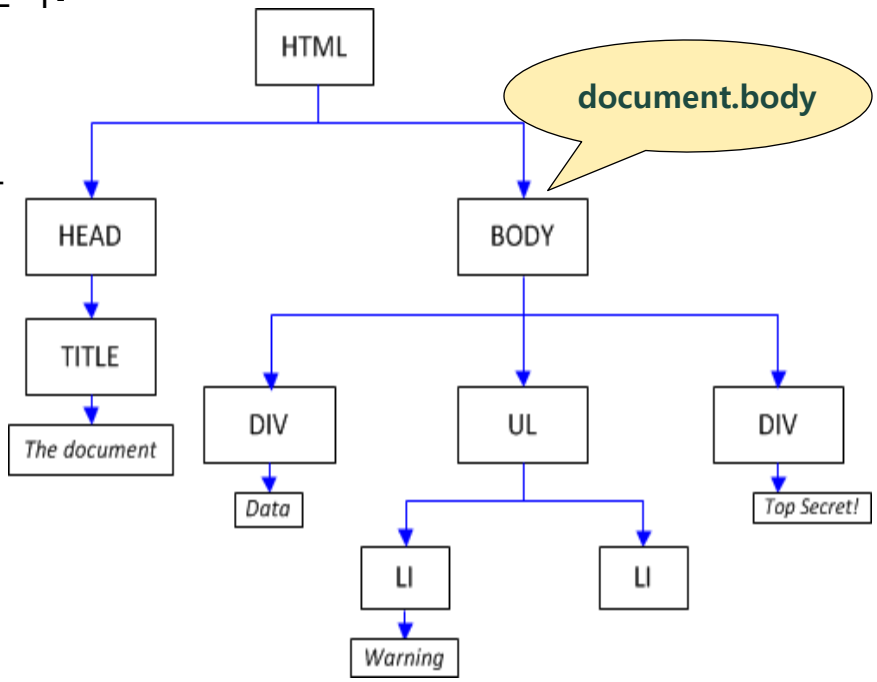
DOM 이란?

- 문서 객체 모델(**DOM; Document Object Model**)은 객체 지향 모델로써 구조화된 문서를 표현하는 형식이다.

- **DOM**은 플랫폼/언어 중립적으로 구조화된 문서를 표현하는 **W3C**의 공식 표준이다. 또한 **W3C**가 표준화한 여러 개의 **API**의 기반이 된다.

- 브라우저는 서버로 부터 응답된 웹 컨텐트 내용을 파싱한 후 트리구조로 각 **HTML** 태그 마다 **DOM** 기술을 적용하여 **JavaScript** 객체를 생성하는데 바로 이 객체들을 **DOM** 객체라 한다.

- **DOM** 객체를 통해서 **HTML** 문서의 내용을 접근하여 읽는 기능 뿐만 아니라 내용을 수정, 삭제, 추가 등 변경하는 기능을 처리 할 수 있다.



JavaScript의 DOM 객체

□ DOM 객체 접근

- **DOM** 객체를 접근할 때는 직접 접근 방법과 노트 워킹 접근 방법이 사용될 수 있다.

  [직접 접근 방법 **]**

  원하는 **DOM** 객체에 접근하기 위해서는 **document** 객체에서 제공되는 다음 메서드들을 사용한다.

  - **document.getElementsByTagName(**'태그명')

    태그명으로 **DOM** 객체들을 찾음

  - **document.getElementById ('id**속성값')

    태그에 정의된 **id** 속성의 값으로 **DOM** 객체 찾음

  - **document.getElementsByClassName('class**속성값')

    태그에 정의된 **class** 속성의 값으로 **DOM** 객체들을 찾음

  - **document.querySelector(**'찾고자 하는 **DOM** 객체에대한 **CSS** 선택자')

    선택자에 알맞은 **DOM** 객체를 찾음

  - **document.querySelectorAll(**'찾고자 하는 **DOM** 객체에대한 **CSS** 선택자')

    선택자에 알맞은 **DOM** 객체들을 찾음

- 문자열로 정의된 엘리먼트의 컨텐트 내용을 추출하려면 **node.nodeValue** 를 사용한다.
- 종류에 관계없이 엘리먼트의 컨텐트 내용을 추출하려면 **node.innerHTML** 을 사용한다.
- 엘리먼트에 정의된 속성을 접근하기 위해서는 **node.getAttribute(**'속성명')을 사용한다.

JavaScript의 DOM 객체

□ DOM 객체의 내용 편집

- **DOM**의 역할은 기존의 노드를 참조하는 것만이 아니며 문서 트리에 대하여 신규의 노드를 추가/치환하거나 기존의 노드를 삭제할 수도 있다.

- **DOM** 객체로 웹 페이지의 내용을 편집하는 방법도 **2**가지 가능하다.
  - 간단한 컨텐트의 편집에는 **node.innerHTML** 또는 **node.textContent** 속성을 사용한다.
  - 복잡한 컨텐트의 편집에는 **document.createXXX()** 메서드를 사용하여 직접 **DOM** 객체를 만든다. 원하는 편집 기능에 따라 다음 메서드들을 사용한다.

    appendChild() : 마지막 자식으로 추가
    insertBefore() : 지정된 자식 앞에 삽입
    replaceChild() : 지정된 자식을 다른 노드로 대체
    remodeChild() : 지정된 자식을 삭제
    cloneNode() : 지정된 자식을 복제한 노드를 반환

- 엘리먼트에 속성을 추가하기 위해서는 **node.setAttribute(**'속성명', '속성값'**)**을 사용하고 삭제시에는 **node.removeAttribute(**'속성명'**)**을 사용한다.

JavaScript의 이벤트 모델

☐ JavaScript의 이벤트 모델

- **JavaScript**는 이벤트 드리븐 모델에 기반하여 동작한다**. Web** 페이지 안에서 발생한 여러 가지 사건**(**이벤트**)**에 따라 대응하는 처리**(**이벤트 핸들러**)**를 호출하여 실행하는 모델이다 **.**

> - event는 어떤 사건을 의미한다. 브라우저에서의 사건이란 사용자가 클릭을 했을 '때', 필드의 내용을 바꾸었을 '때'와 같은 것을 의미한다.
> - event target은 이벤트가 일어날 객체를 의미한다.
> - event type은 이벤트의 종류를 의미한다. 위의 예제에서는 click이 이벤트 타입이다.
> - event handler는 이벤트가 발생했을 때 동작하는 코드를 의미한다.

- **JavaScript**가 지원하는 이벤트는 애플리케이션 사용자가 발생시키는 이벤트와 애플리케이션이 스스로 발생시키는 이벤트로 나뉜다**.**

> load, click
> mousedown, mousemove, mouseover, mouseup
> keydown, keypress, keyup
> change, reset, submit
> blur, focus

JavaScript의 이벤트 모델

□ DOM 객체에 이벤트를 연결하는 다양한 방법

- 인라인 이벤트 모델
  - 이벤트 핸들러를 등록하고자 하는 대상의 **HTML** 태그에 속성으로 정의하는 모델이다.
  - 등록된 이벤트 핸들러를 해제할 수 있는 방법은 없으므로 이벤트 핸들러 역할의 함수에서 프로그램적으로 해결해야 한다.

- 전역적 이벤트 모델
  - 이벤트 핸들러를 등록하고자 하는 대상의 **DOM** 객체를 찾아서 이벤트 핸들러를 등록하는 모델이다.
  - 핸들러를 등록하려는 이벤트에 대한 속성을 사용한다.
  - 등록된 이벤트 핸들러를 해제하려면 핸들러를 등록한 속성의 값에 **null** 을 재할당한다.

- 표준 이벤트 모델

  다음과 같은 이벤트 연결/해제 메서드들을 모든 **DOM** 객체들이 지원하므로 이벤트 핸들러를 등록하려는 **DOM** 객체에 대하여 다음 메서드들을 사용한다.
  - **addEventListener(eventName, handler, useCapture)** : 이벤트 핸들러 등록시 사용
  - **removeEventListener(eventName, handler)** : 이벤트 핸들러 해제시 사용

JavaScript의 이벤트 모델

- 인라인 이벤트 모델

```
<script>
function btn_onclick(){
      window.alert(  '안녕?');
}
</script>
<input type="button" value="다이얼로그 표시" onclick="btn_onclick()  ">
```

- 고전 이벤트 모델

```
<script>
      window.onload = function() {
        document.getElementById('btn').onclick = function(){
          window.alert('안녕?');
        };
      };
</script>
<input id="btn" type="button" value="다이얼로그 표시  ">
```

- 표준 이벤트 모델

```
<script>
      window.addEventListener('load', function() {
        document.getElementById('btn').addEventListener('click', function(){
              window.alert('안녕?');}, false}, false);
</script>
<input id="btn" type="button" value="다이얼로그 표시">
```

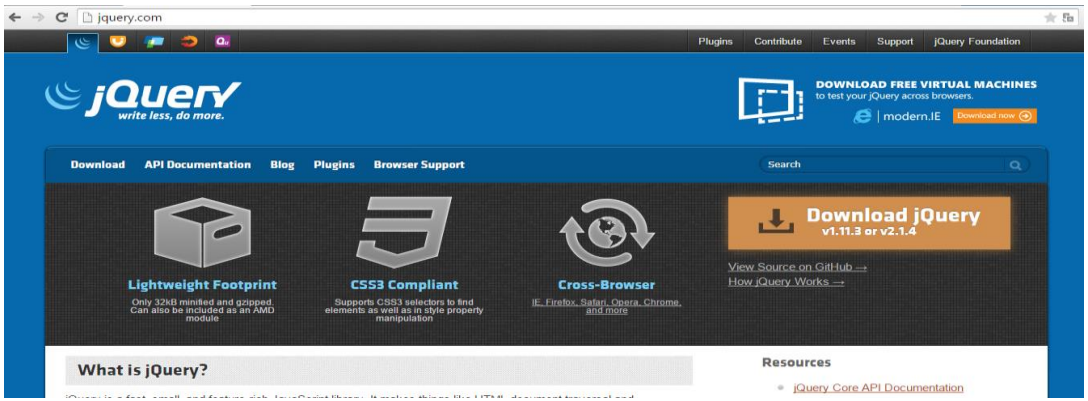
jQuery 라이브러리 사용

jQuery 란?

- **HTML** 문서 안의 스크립트 코드를 단순화하도록 설계된 자바스크립트 라이브러리이다.
- 존 레식**(John Resig)**이 **2006**년 자바스크립트를 쉽게 이용할 목적으로 제안하였다.
- 웹 브라우저 마다 다르게 작성해야 되는 크로스 브라우징 자바스크립트 코드를 **jQuery** 라이브러리를 사용하여 최대한 쉽게 작성할 수 있도록 지원하는 것을 목표로 한다.

[주요 기능]

- **DOM** 요소 선택 기능, **DOM** 탐색 및 수정
- **CSS** 셀렉터에 기반한 **DOM** 조작
- 크로스 브라우징 이벤트 처리
- 특수효과 및 애니메이션
- **AJAX**
- **JSON** 파싱
- 플러그인을 통한 확장성



jQuery 라이브러리 사용

☐ jQuery 의 주요 활용 기능

DOM 프로그래밍, 이벤트 처리 프로그래밍, AJAX 프로그래밍, 화면 효과 프로그래밍

분류	기능	지원 메서드
Core	핵심개념	jQuery( ) 선언 함수 정의 및 활용 방법
Selectors	선택자	DOM 트리의 노드 선택 표현식
CSS	스타일	CSS 스타일 속성값 변경 메서드
Traversing	탐색	DOM 트리의 계층 구조를 이용한 노드 탐색 메서드
Manipulation	조작	DOM 트리의 노드 변경 메서드
Attributes	속성	엘리먼트 속성값의 조회 및 변경 메서드
Events	이벤트	마우스, 키보드, 폼 및 문서 관련 이벤트 메서드
Effects	효과	동적 스타일 변화를 위한 메서드
Ajax	비동기교환방식	Ajax 관련 메서드
UI	사용자 인터페이스	사용자 인터페이스용 라이브러리

jQuery 라이브러리 사용

☐ jQuery 라이브러리 선언

- **jQuery** 라이브러리를 포함하는 **<script>** 태그를 작성한다.

```
<script src="http://code.jquery.com/jquery-xxx.js"></script>
```

- **jQuery** 에서 제공되는 메서드들은 두 가지 방식으로 호출된다.

```
jQuery(자바스크립트객체).xxx()
jQuery.xxx()
```

- **jQuery()** 함수의 주요 아규먼트

```
jQuery( selector [, context ] )
jQuery( element )
jQuery( elementArray )
jQuery( object )
jQuery( selection )
jQuery()
jQuery( html [, ownerDocument ] )
jQuery( html, attributes )
jQuery( callback )
```

jQuery → $

$는 자바스크립트 객체를 jquery로 포장해준다.

$(자바스크립트 객체)
$('css 선택자')
$()
$()
$()
$()

코드 안의 jQuery() 함수는
식별이 어렵고 불편하기 때문에
줄여서 $( )로 표시

jQuery 라이브러리 사용

$() 함수의 사용

- 시작 이벤트 핸들러 등록

```
jQuery(document).ready(function() { . . . . . });
  → $(document).ready(function() { . . . . . });
  → $(function() { . . . . . });
```

- 선택자에 알맞은 **DOM** 객체 추출
  - **$(**선택자**)** 형태로 **jQuery** 선택자를 입력 인자로 받아 들여 선택자 조건을 만족하는 엘리먼트 노드들을 **DOM** 트리에서 찾아 **'jQuery** 객체**'** 형식으로 반환
  - 일치하는 **DOM** 노드들이 여러 개이면 배열 형태의 **'jQuery** 객체 집합**'**을 반환
  - 반환하는 **DOM** 엘리먼트들을 **jQuery** 객체 개념으로 감싸고 미리 준비된 메서드를 사용할 수 있도록 확장

```
jQuery('div').html('<h1>테스트</h1>')
                    →  $('div').html('<h1>테스트</h1>')
```

jQuery 라이브러리 사용

- $() 함수에 사용 가능한 선택자들

Selectors

Basics
#id
element
.class,
.class.class
*
selector1,
selector2

Hierarchy
ancestor descendant
parent > child
prev + next
prev ~ siblings

Basic Filters
:first
:last
:not(selector)
:even
:odd
:eq(index)
:gt(index)
:lt(index)

Content Filters
:contains(text)
:empty
:has(selector)
:parent

Visibility Filters
:hidden
:visible

Child Filters
:nth-child(expr)
:first-child
:last-child
:only-child

Attribute Filters
[attribute]
[attribute=value]
[attribute!=value]
[attribute^=value]
[attribute$=value]
[attribute*=value]
[attribute|=value]
[attribute~=value]
[attribute]
[attribute2]

Forms
:input
:text
:password
:radio
:checkbox
:submit
:image
:reset
:button
:file

Form Filters
:enabled
:disabled
:checked
:selected

Canvas API

Canvas API 란?

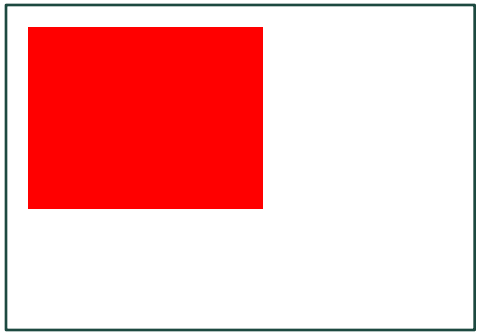
- 웹 페이지에 그림을 그릴 수 있도록 지원하는 **HTML5 API** 이다.
- **<canvas>** 엘리먼트를 사용하여 그림을 그리기 위한 영역을 정의하고 스크립트로 그림을 그린다.
- 직선, 박스, 원, 베지에 곡선 등 다양한 그림을 직접 그릴 수 있으며 원하는 사이즈 그리고 칼라의 이미지 출력을 처리할 수 있다.
- **<canvas>** 엘리먼트 작성 방법 : 그림을 그릴 수 있는 사각형 영역이 만들어진다.

```
<canvas  id= "draw"  width= "400"  height= "300"></canvas>
```

HTMLCanvasObject 객체 접근

- 웹 스크립트로 그림을 그리기 위해서는 **<canvas>** 태그를 **DOM** 객체로 접근해야 한다.
- **<canvas>** 엘리먼트를 사용하여 그림을 그리기 위한 영역을 정의하고 스크립트**(JavaScript** 코드**)**로 그림을 그린다.

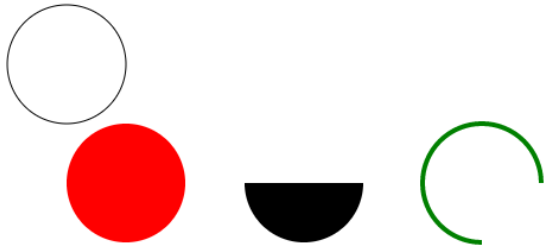
```
var area = document.getElementById("draw");
var ctx = area.getContext("2d");
ctx.fillStyle = "rgb(255,0,0)";
ctx.fillRect (10, 10, 100, 100);
```



Canvas API

그리기 기능을 지원하는 메서드들

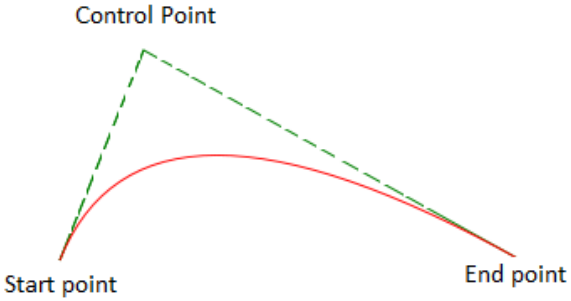
- **fillRect(x, y, width, height) :** 색이 칠해진 사각형을 그린다.
- **strokeRect(x, y, width, height) :** 테두리만 있는 사각형을 그린다.
- **clearRect(x, y, width, height) :** 특정 영역을 지우고 완전히 투명하게 만든다.
- **beginPath() :** 경로를 시작한다.
- **closePath() :** 경로를 종료한다.
- **stroke() :** 경로를 따라서 테두리 선을 그린다.
- **fill() :** 설정된 스타일로 도형을 채운다.
- **moveto(x,y) : (x,y)** 위치로 시작점을 옮긴다.
- **lineto(x,y) : x**에서 **y**까지 직선을 그린다.
- **strokeText(msg, x, y) : (x,y)** 위치에 텍스트를 테두리선만 그린다.
- **fillText(msg, x, y) : (x,y)** 위치에 텍스트를 색을 채워서 그린다.
- **measureText(msg) :** 측정된 문자열의 길이정보를 저장한 **TextMetrics** 객체를 리턴한다.
- **arc(x, y, r, startAngle, endAngle, anticlockwise) :**
  (x,y)에서 시작하여 반시계방향
  (anticlockwise)으로 반지름**(r)**만큼의 원을 그린다.

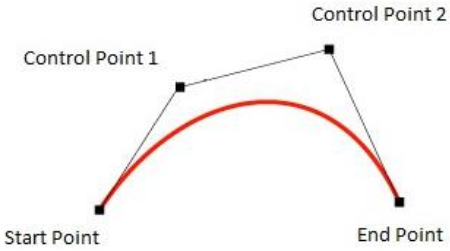


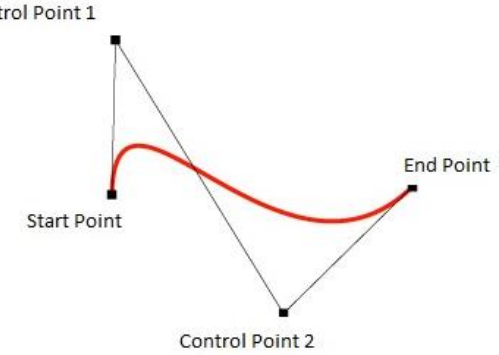
Canvas API

☐ 그리기 기능을 지원하는 메서드들

- **quardraticCurveTo(cp1x, cp1y, x, y) :**
  한 개의 조절점**(cp1x,cp1y)**을 이용해 **(x,y)**까지의 곡선을 그린다
- **bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y) :**
  두 개의 조절점**(cp1x,cp1y)**와 **(cp2x,cp2y)**를 이용해 **(x,y)**까지의 곡선을 그린다





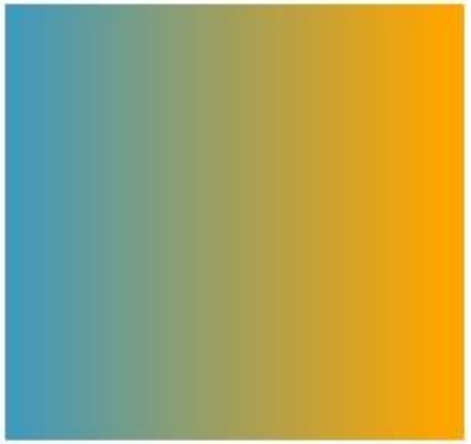


- **drawImage(image, sx, sy)**
- **drawImage(image, sx, sy, sWidth, sHeight)**
- **drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth, dHeight)**
  이미지 파일을 읽어서 주어진 위치에 주어진 크기로 또는 슬라이스하여 그린다.

Canvas API

□ 그라디언트와 패턴

- 그라디언트 : **CanvasGradient** 객체를 생성한다.
  - **createLinearGradient(x1, y1, x2, y2)** : 선형그라디언트 객체를 생성한다.
  - **createRadialGradient(x1, y1, r1, x2, y2, r2)** : 원형그라디언트 객체를 생성한다.
  - **CanvasGradient** 객체의 메서드
    addColorStop(position, color) : **position(0.0~1.0)** 위치에 **color** 를 설정한다.
- 패턴 : **CanvasPattern** 객체를 생성한다.
  - **createPattern(image, type)** : **image** 와 **type** 에 알맞은 패턴 객체를 생성한다.
    image 에는 **CanvasImageSource** 객체를 지정하며
    type 는 **repeat, repeat-x, repeat-y, no-repeat** 중
    한 개를 설정한다.



- **save ()**
  - 캔버스의 상태정보를 스택에 저장
  - 스택에 저장되는 정보
    - 회전이나 크기 조절과 같이 캔버스에 적용된 변형 내용
- **restore()**
  - 스택에 저장된 상태 정보를 읽어온다.

Canvas API

□ 스타일

[색상]

- **fillStyle** : 채워질 색상 지정
  칼라, **CanvasGradient** 객체, **CanvasPattern** 객체를 지정할 수 있다.
- **strokeStyle** : 테두리 색상 지정
  칼라, **CanvasGradient** 객체, **CanvasPattern** 객체를 지정할 수 있다.
- **globalAlpha** : 투명도 지정한다. **0**(완전투명)에서 **1**(완전불투명)사이 값을 가짐

[선]

- **lineWidth** : 선의 두께 , **1**이 기본값
- **lineCap** : 선의 끝모양을 결정한다.
  - **butt** : 기본값으로 아무런 효과 없음
  - **round** : 선 너비의 **1/2**을 반지름으로 하는 반원이 선 양쪽 끝에 그려서 표시
  - **square** : 선 양쪽 끝에 사각형을 그려서 표시
- **lineJoin** : 두 개의 선이 만날 때 선의 교차점 표시한다.
  - **round** : 선과 선이 만나는 부분이 둥글게 처리
  - **bevel** : 두 선 연결 부분에 단면으로 표시
  - **miter** : 연결한 흔적이 남지 않고 마치 처음부터 하나의 선이었던 것처럼 연결. 기본값.

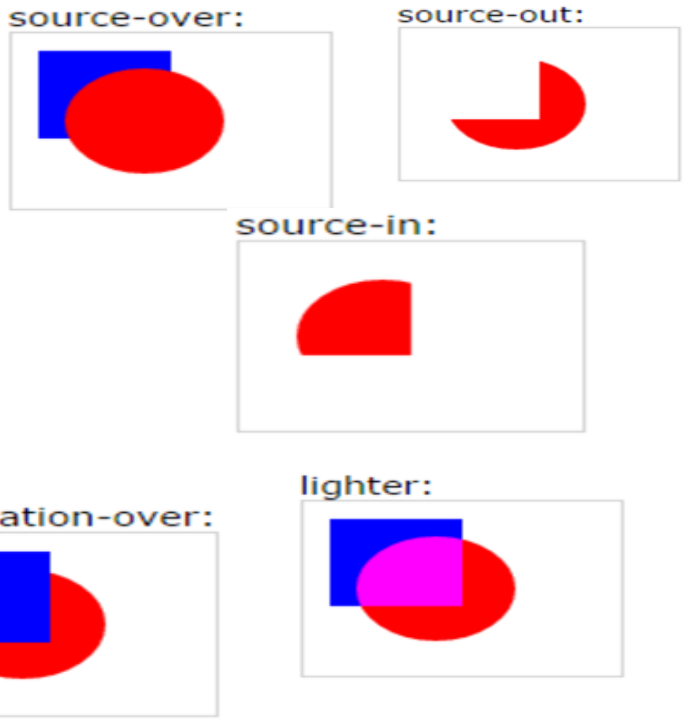
Canvas API

그림자 효과

- **shadowOffsetX**
  - 객체로부터 그림자가 x축 방향으로 얼마나 떨어져 있는지 표시한다.(기본값은 0, 음수이면 왼쪽)
- **shadowOffsetY**
  - 객체로부터 그림자가 y축 방향으로 얼마나 떨어져 있는지 표시한다.(기본값은 0, 음수이면 위쪽)
- **shadowBlur** : 그림자가 얼마나 흐릿한지 나타낸다.**(**기본값은 **0)**
- **shadowColor** : 그림자 색상을 지정한다. 기본값은 완전히 투명한 검정색 이다.

도형 변형

- **scale(x, y)** : 도형의 크기를 조정한다.
- **rotate(angle)** : 주어진 각도만큼 도형을 회전한다.
- **translate(x, y)** : 도형을 그리는 기준 위치를 이동한다.



도형 합성

- **globalCompositeOperatio** : 원본(먼저 그린 도형) 도형과 대상(나중에 그린 도형) 도형의 겹쳐진 형태에 따른 표시 방법을 정의한다.

Canvas API

□ 비트맵이미지 관리와 HTMLObjectCanvas 객체 저장

- 비트맵이미지 관리
  - **createImageData(sw, sh)**
  - **createImageData(ImageData** 객체**)**
    비트맵 이미지 객체**(ImageData)**를 생성한다**.**
  - **getImageData(sx, sy, sw, sh) :**
    <canvas>객체의 주어진 영역의 데이터를 비트맵 이미지 객체**(ImageData)**로 추출한다**.**
  - **putImageData(ImageData** 객체**, dx, dy) :**
    <canvas>객체의 **(dx, dy)** 위치에 비트맵 이미지 객체**(ImageData)**의 데이터를 출력한다**.**

- **HTMLObjectCanvas** 객체 저장
  toDataURL() : <canvas> 태그 영역의 모든 내용을 **png** 형식의 **URI** 문자열로 변환하여 리턴한다**.**

```
var canvas = document.getElementById("draw");
var dataURL = canvas.toDataURL();
console.log(dataURL);
// "
// blAAAADElEQVQImWNgoBMAAABpAAFEI8ARAAAAAElFTkSuQmCC"
```

Multimedia API

<video> 와 <audio> 태그 관련 API

- **controls, autoplay, loop**
  설정 여부를 조정하는 **boolean** 타입의 속성이다.
- **currentTime**
  현재의 재생 위치를 초 단위로 나타내는 속성이다.
- **duration**
  오디오 또는 비디오 파일의 길이를 초단위로 나타내는 속성이다.
- **ended/paused**
  재생의 종료여부 또는 일시 정지 여부를 나타내는 속성이다.
- **canPlayType(type)**
  브라우저가 해당 미디어 타입을 재생할 수 있는지를 나타내는 문자열이다.
- **play()**
  현재 위치에서 재생을 시작한다.
- **pause()**
  오디오가 재생 중일 경우 일시 정지한다.

Web Storage API

- 웹 스토리지란?
  - 웹 브라우저에 자료를 저장하기 위한 기능으로 로컬스토리지와 세션스토리지로 나뉜다.
  - 기존의 쿠키와 비슷한 기술이지만 일부 기능에서 차이를 가지고 있다.
  - 저장하려는 데이터마다 유일한 이름(키)을 같이 저장한다.
  - 저장하려는 데이터의 종류에는 제한이 없으며 저장시에는 문자열로 저장된다
  - 로컬스토리지(local storage) : 영구 보관
  - 세션스토리지(session storage) : 브라우저가 종료될 때까지 보관
  - W3C는 Same Origin Policy에 따라 도메인당 5MB를 권장하고 있으며 추가 용량이 필요할 경우 사용자의 동의를 얻어 용량을 확장할 수 있다.(초과시 QUOTA_EXCEEDED-ERR 발생)
  - Same Orign Policy 정책이 적용된다.

Web Storage API

- window.localStorage 와 window.sessionStorage 의 주요 멤버
  - length : 스토리지에 저장된 key/value 쌍의 개수를 추출하는 속성이다.
  - key(index) : 숫자형 인덱스에 해당하는 key를 리턴한다.
  - getItem(key) : 스토리지로 부터 key 에 해당하는 value 를 추출한다.
  - setItem(key, value) : 스토리지에 key 에 해당하는 value 를 저장한다.
  - removeItem(string key) : 스토리지에 key 에 해당하는 value 를 제거한다.
  - clear() : 현재 스토리지의 모든 데이터를 제거한다.
  - onstorage : 로컬 스토리지의 내용이 변경될 때마다 발생되는 이벤트로 로컬 스토리지의

    변경 사항을 모니터링 하는 것이 가능하다. StorageEvent 객체가 생성된다.

    [StorageEvent 객체의 주요 속성]
    - key : 추가, 삭제, 변경된 키 이름
    - oldValue : 업데이트되기 전의 값으로 새로 추가된 값이면 null
    - newValue : 새로 업데이트된 값으로 기존 값을 삭제한 경우에는 null
    - url : 변경사항이 발생된 페이지의 URL

Web Storage API

- 로컬 스토리지의 데이터 관리
  - 저장

    localStorage.mykey = "myvalue";

    localStorage["mykey"] = "myvalue";

    localStorage.setItem("mykey", "myvalue");
  - 읽기

    var mydata = localStorage.mykey;

    var mydata = localStorage["mykey"];

    var mydata = localStorage.getItem("mykey");
  - 삭제

    delete localStorage.mykey;

    delete localStorage["mykey"];

    localStorage.removeItem("mykey");

Web Storage API

- 세션 스토리지의 데이터 관리
  - 저장

    sessionStorage.mykey = "myvalue";

    sessionStorage["mykey"] = "myvalue";

    sessionStorage.setItem("mykey", "myvalue");
  - 읽기

    var mydata = sessionStorage.mykey;

    var mydata = sessionStorage["mykey"];

    var mydata = sessionStorage.getItem("mykey");
  - 삭제

    delete sessionStorage.mykey;

    delete sessionStorage["mykey"];

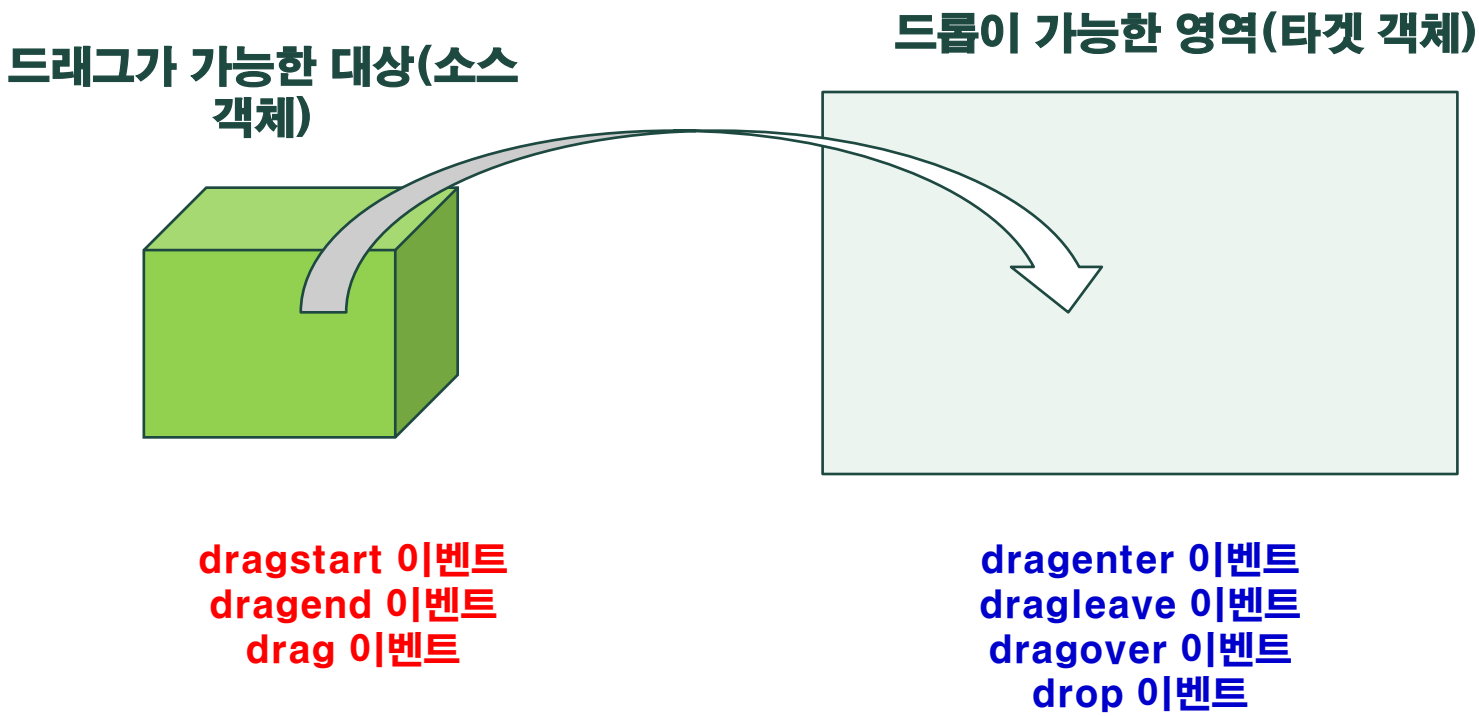
    sessionStorage.removeItem("mykey");

Drag&Drop API

- Drag&Drop 이란?
  - Drag&Drop 은 사용자 편의성을 고려한 UI 기능이다.
  - 마우스를 사용하여 애플리케이션 간에 파일이나 애플리케이션의 데이터를 전달하는 기능으로서 다양한 이벤트의 핸들러를 구현하여 처리한다.
  - 웹 애플리케이션에서는 화면 상에 나타나는 요소를 옮기거나 외부에 있는 파일을 읽어 웹 페이지에 출력 또는 업로드 하는 용도로 사용된다.
  - HTML5 의 엘리먼트들은 draggable 속성의 값을 true 로 지정하면 드래그가 가능한 소스객체가 된다.
  - <img> 엘리먼트는 디폴트로 draggable 속성의 값이 true 이며 다른 엘리먼트들은 설정해주어야 한다.

Drag&Drop API

- Drag&Drop 관련 이벤트



Drag & Drop API

☐ 드래그 앤 드롭 이벤트

- **dragstart** 이벤트
  - 엘리먼트에서 드래그를 시작할 때 발생
- **drag** 이벤트
  - 드래그하는 동안 일어나는 연속적인 이벤트
  - 마우스 커서를 움직일 때 드래그 이벤트가 드래그 소스에서 반복적으로 호출된다.
  - drag 이벤트가 일어나는 동안 나타나는 드래그 피드백의 형태는 바꿀 수 있지만 dataTransfer에 있는 데이터에는 접근할 수 없다.
- **dragenter** 이벤트
  - 드래그된 요소가 드롭 동작을 수행하기 위해 dropzone 영역에 들어 갔을 때 발생하는 이벤트
- **dragleave** 이벤트
  - 드래그된 엘리먼트를 드롭 동작을 하지 않고 dropzone 영역을 벗어날 때 발생하는 이벤트
- **dragover** 이벤트
  - 드래그된 엘리먼트가 dropzone 영역에서 움직일 때 발생하는 이벤트
  - 드래그 소스에서 호출되는 drag 이벤트와는 달리 이 이벤트는 마우스의 현재 타깃에서 호출된다.
- **drop** 이벤트
  - 사용자가 마우스 버튼을 놓을 때 현재 마우스 타깃에서 호출
- **dragend** 이벤트
  - 연쇄 작용의 마지막 단계에서 일어나는 이벤트로 drop 이벤트 후 발생한다.
  - 드래그 소스에서 발생하여 드래그가 완료되었다는 것을 나타낸다.

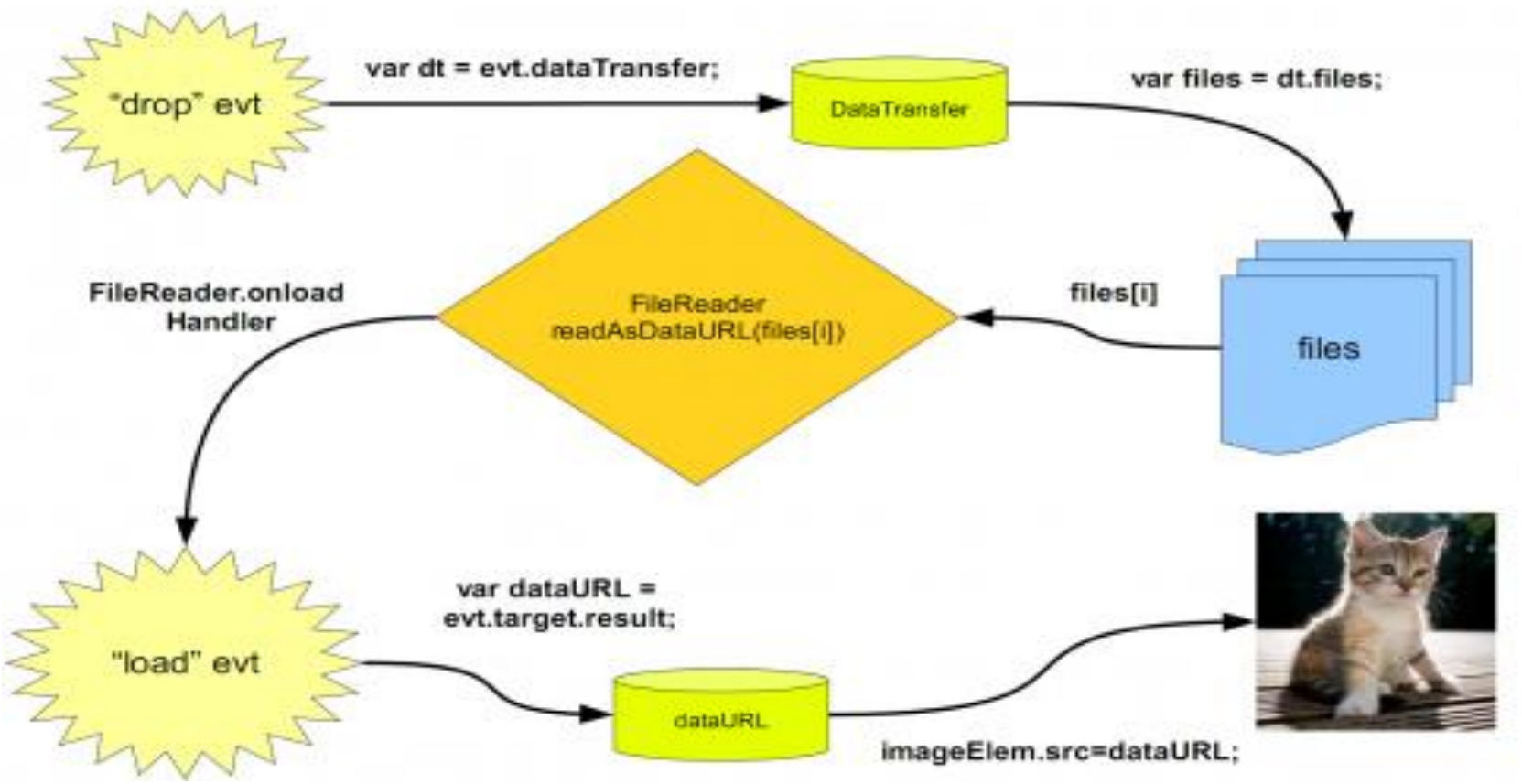
Drag&Drop API

- **dataTransfer 객체**

  - 드래그되는 소스객체에서 드롭이 일어나는 타겟 객체로 전달하려는 데이터를 저장하는 객체이다.
  - dragstart 이벤트 발생시 전달되는 이벤트 객체의 dataTransfer 속성을 사용한다.
  - dataTransfer 객체의 주요 속성과 메서드
    - files 속성 : FileList 타입으로, 드래그 대상이 파일일 때 사용된다.
    - types 속성 : StringList 타입으로, 전달되는 데이터들의 타입명을 추출할 수 있다.
    - clearData(type) : type 명에 해당되는 데이터를 삭제한다.
    - getData(type) : type 명에 해당되는 데이터를 추출한다.
    - setData(type, data) : type 명으로 데이터를 저장한다.
    - setDragImage(image, x, y) : 드래그하는 동안 커서를 따라다니는 이미지를 (x,y) 위치에 출력되도록 설정한다.

Drag & Drop API

□ 파일에 대한 Drag & Drop



File API

- HTML5 File API의 특징
  - HTML5에서는 로컬 컴퓨터에 있는 파일을 웹 애플리케이션이 읽을 수 있는 기능을 제공한다.
  - 텍스트 형식 뿐만 아니라 이진(binary)형식도 읽는 것이 가능하다.
  - 읽는 기능 뿐만 아니라 출력과 생성기능도 스펙이 제안되기도 했지만 현재로서는 읽는 기능의 스펙만 남고 출력이나 생성 기능의 API는 제거되었다.
  - File API
    - 파일에 대한 정보를 알아내기 위한 기능을 지원하는 File 객체
    - 파일의 내용을 읽어들이는 기능을 지원하는 FileReader 객체

File API

- File 객체
  - 로컬하드디스크에 존재하는 파일의 정보를 추출하거나 조작할 수 있는 멤버를 제공한다.

  - name : 파일의 이름을 추출하는 기능의 속성이다.
  - type : 파일의 MIME Type (알 수 없을 때는 null)정보를 추출하는 기능의 속성이다.
  - size : 파일의 크기를 추출하는 기능의 속성이다.
  - lastModifiedDate : 파일의 마지막 수정일을 추출하는 기능의 속성이다.
  - slice(start, length, contentType) : 파일에서 시작위치(start)부터 length만큼 파일의 내용을 Blob 객체로 리턴한다.

File API

- FileReader 객체
  - 로컬하드디스크에 존재하는 파일의 내용을 읽는 기능을 제공한다.

  - readAsText(Blob 또는 File 객체, encoding) : 파일 내용을 텍스트 문자열로 읽는다. 두 번째 인자로는 파일의 문자 인코딩을 지정한다.(생략 시는 UTF-8)
  - readAsDataURL(Blob 또는 File 객체) : 파일 내용을 data: 으로 시작하는 DataURL 형식의 문자열로 읽는다.
  - readAsBinaryString(Blob 또는 File 객체) : 파일의 내용을 바이너리 형식으로 읽는다.
  - readAsArrayBuffer(Blob 또는 File 객체) : 파일의 내용을 읽어서 ArrayBuffer 객체에 저장한다.
  - abort() : 읽는 작업을 중간에 중지한다.
  - result : 읽어들인 파일의 내용 추출하는 용도로 사용되는 속성이다.
  - error : 에러 발생 시의 에러 정보를 추출하는 용도로 사용되는 속성이다.

File API

- FileList
  - File 객체들을 담고 있는 배열과 같은 객체이다.
  - 파일에 대한 드래그 앤 드롭에서 dataTransfer 객체를 통해 또는 <input type="file"> 태그의 DOM 객체를 통해서 사용되는 files 속성의 타입이다.
  - length : 저장된 File 객체들의 크기를 추출할 수 있는 속성이다.
  - item(index) : index 위치의 File 객체를 추출하는 기능의 메서드이다.

- Blob
  - 실제 데이터들을 표현하는 객체이다.
  - type : 파일의 MIME Type (알 수 없을 때는 null)정보를 추출하는 기능의 속성이다.
  - size : 파일의 크기를 추출하는 기능의 속성이다.
  - slice(start, length, contentType) : 파일에서 시작위치(start)부터 length만큼 파일의 내용을 Blob 객체로 리턴한다.

File API

- 이벤트 관련 속성
  - onload
    load 이벤트 발생시 호출되는 이벤트 핸들러를 등록하는 속성이다.
  - onprogress
    progress 이벤트 발생시 호출되는 이벤트 핸들러를 등록하는 속성이다.
  - onerror
    error 이벤트 발생시 호출되는 이벤트 핸들러를 등록하는 속성이다.
  - onloadend
    loadend 이벤트 발생시 호출되는 이벤트 핸들러를 등록하는 속성이다.
  - onloadstart
    loadstart 이벤트 발생시 호출되는 이벤트 핸들러를 등록하는 속성이다.
  - onabort
    abort이벤트 발생시 호출되는 이벤트 핸들러를 등록하는 속성이다.

Geolocation API

☐ window.navigator.geolocation 객체

- 위치 요청에 대해 사용자가 동의하면 브라우저의 위치 정보가 반환된다.
- 지오로케이션을 지원하는 브라우저가 데스크탑이나 휴대전화 같은 디바이스에서 **IP**주소 또는 **GPS** 그리고 **WiFi** 기지국, 스마트폰 기지국을 이용해 위치 정보를 브라우저에 전달한다.
- 위치 정보는 추가적인 메타데이터와 함께 위도와 경도 좌표로 전달된다.

- **getCurrentPosition() :** 현재 위치정보를 추출한다.
  getCurrentPosition(successCallback[, errorCallback, options])
  - successCallback : 위치정보를 성공적으로 추출했을 때 호출되는 콜백 함수이다.
  - errorCallback : 위치정보를 추출하는데 실패했을 때 호출되는 콜백 함수이다.
- **watchPosition() :** 주기적으로 반복해서 위치정보를 구하는 작업을 수행한다.
  watchCurrentPosition(successCallback[, errorCallback, options])

Geolocation API

successCallback 함수

사용자 위치를 **Position** 객체를 통해 파라미터로 전달 받으며 다음과 같은 필드 포함한다.

- **timestamp :** 위치를 가져올 당시의 시간을 제공한다.
- **coords :** 위치 정보를 제공하는 **Coords** 객체이다.
  - event.coords.latitude — 위도를 수치로 반환(단위는 도)
  - event.coords.longitude — 경도를 수치로 반환(단위는 도)
  - event.coords.accuracy — 위도 및 경도의 정밀도를 반환(단위는 미터)
  - event.coords.altitude — GPS 고도를 반환(단위는 미터)
  - event.coords.altitudeAccuracy — GPS 고도의 정밀도 반환(단위는 미터)
  - event.coords.heading — 진행방향을 0부터 360 사이의 수치로 반환(북쪽은 0도, 동쪽은 90도, 남쪽은 180도, 서쪽은 270도)
  - event.coords.speed — 이동 속도를 수치로 반환(단위는 m/s)
  - event.timestamp — 위치 정보를 얻었을 때의 시간을 나타내는 Date 객체를 반환
- **address :** 나라나 시, 거리 등 주소를 표시하는 **Address** 객체이다.

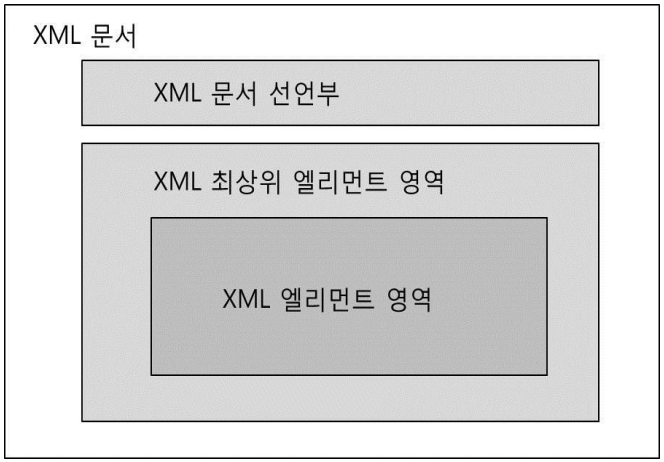
errorCallback 함수

파라미터로 **PositionError**를 전달 받으며 포함된 필드는 다음과 같다.

- **code : UNKNOWN_ERROR, PERMISSION_DENIED, POSITION_UNAVAILABLE , TIMEOUT** 중 하나의 값이다.
- **message :** 로그와 디버깅을 위해 사람이 읽을 수 있는 오류 메시지를 제공한다.

XML(Extensible Markup Language)

- XML 문서
  - XML 선언부를 제외하고는 기존 HTML5의 기본 구조와 사용 방법이 거의 유사
  - XML 문서 선언부
    - 반드시 맨 앞에 명세, XML 문서 유형을 지정
    - XML 문서 구조를 정의한 DTD(또는 XML Schema) 선언, 스타일을 정의한 CSS 연결에 대한 선언도 명세



  - 하나의 최상위 엘리먼트의 <시작태그>로 시작해서 </종료태그>로 끝남
  - 최상위 엘리먼트를 포함하여 XML 문서의 모든 태그들은 자유롭게 정의
  - 엘리먼트의 시작 태그 안의 속성도 자유롭게 정의

JSON

- JSON 이란?

JSON(제이슨, JavaScript Object Notation)은, 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법이다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수값을 표현하는 데 적합하다. 형식은 자바스크립트의 구문 형식을 따르지만, 프로그래밍 언어나 플랫폼에 독립적이므로 C, C++, C#, 자바, 자바스크립트, 펄, 파이썬 등 많은 언어에서 이용할 수 있다.

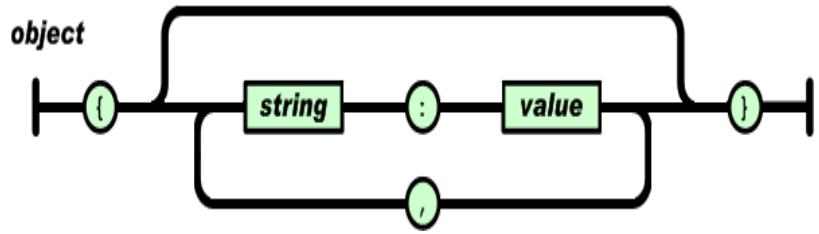
JSON 문법은 자바스크립트 표준인 ECMA-262 3판의 객체 문법에 바탕을 두며, 인코딩은 유니코드로 한다. 표현할 수 있는 기본 자료형으로는 수, 문자열, 참/거짓, null이 있고, 집합 자료형으로는 배열과 객체가 있다.

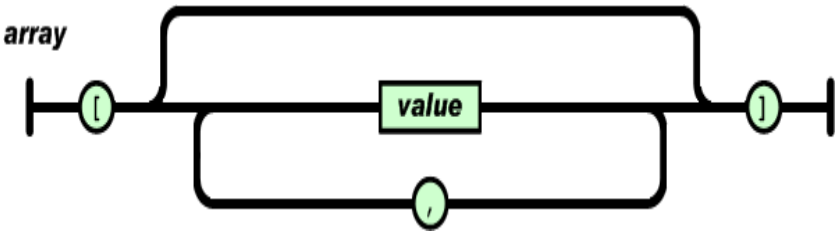
[JSON의 장점]

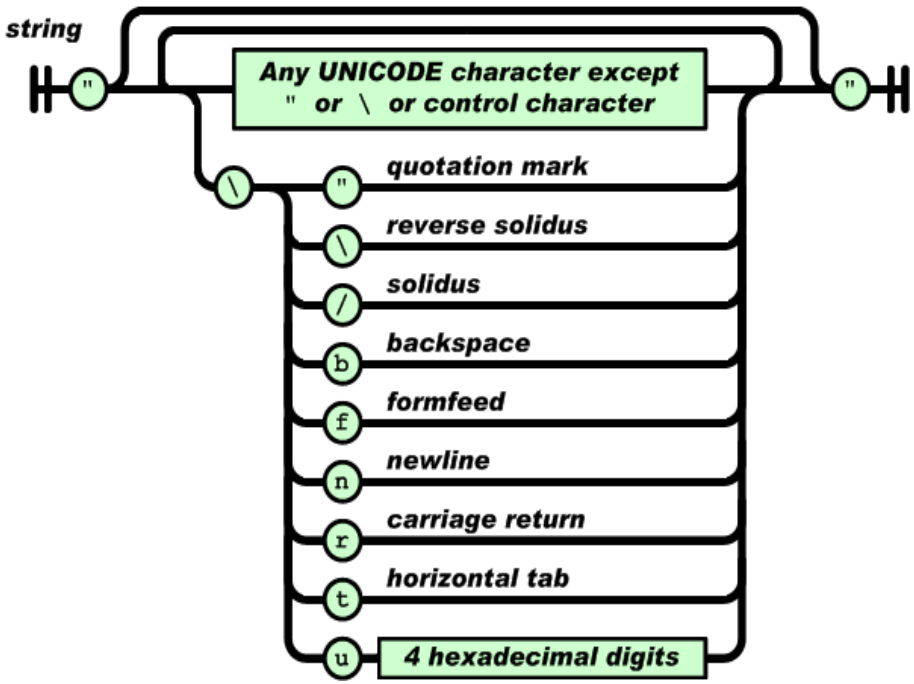
- JSON은 텍스트로 이루어져 있으므로, 사람과 기계 모두 읽고 쓰기 쉽다.
-프로그래밍 언어와 플랫폼에 독립적이므로, 서로 다른 시스템간에 객체를 교환하기에 좋다.
- JSON은 개방형 표준이며, 읽기 및 쓰기가 쉽고 가볍다.

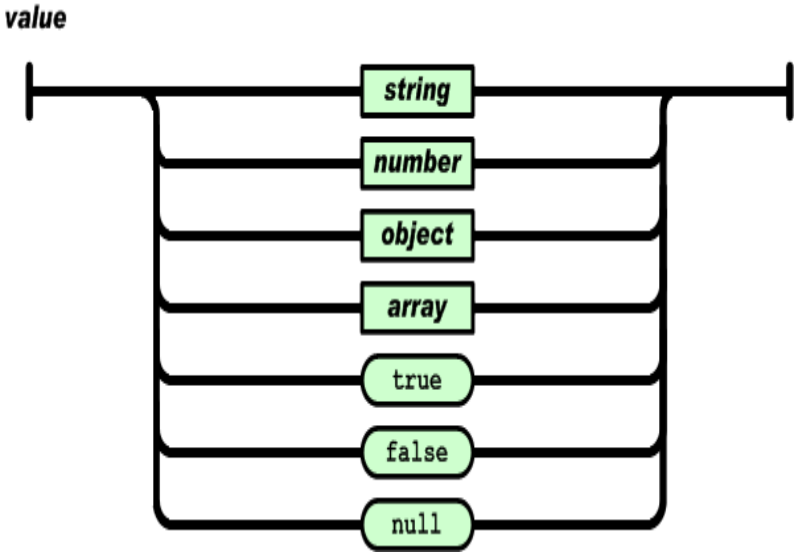
```
{
    "status": "OK",
    "data": {
      "trends": {
        "uv": [
          {"date": "200906", "value": 90714948},
          {"date": "200907", "value": 98292793},
          {"date": "200908", "value": 103509116},
          ...
        ]
      },
      "trends_low_sample": false,
      "query_cost": 13,
      "trends_frequency": "monthly"
    }
  }
```

JSON


object
{
string
:
value
}
,


array
[
value
]
,


string
"
Any UNICODE character except
" or \ or control character
"
\
"
quotation mark
\
reverse solidus
/
solidus
b
backspace
f
formfeed
n
newline
r
carriage return
t
horizontal tab
u
4 hexadecimal digits

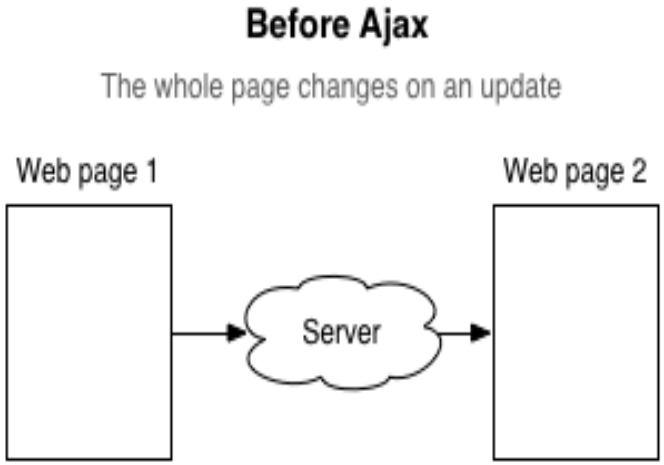

value
string
number
object
array
true
false
null

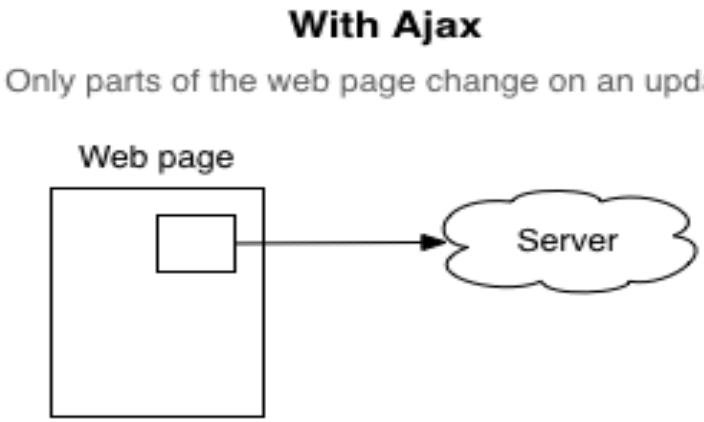
XML과 JSON 비교

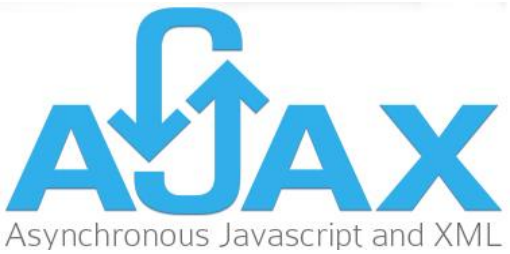
JSON 형식	XML 형식
{ "students" : { "student" : [{"name":"홍길동", "gender":"남"}, {"name":"홍길순", "gender":"여"},] } }	<students> <student> <name>홍길동</name> <gender>남</gender> </student> <student> <name>홍길순</name> <gender>여</gender> </student> </students>

AJAX

- **AJAX = Asynchronous JavaScript and XML**
  - 고전적인 웹의 통신 방법은 웹페이지의 일부분을 갱신하기 위해서는 페이지 전체를 다시 로드 해야 했다.
  - AJAX 의 핵심은 재로드(refresh 재갱신) 하지 않고 웹페이지의 일부만을 갱신하여 웹서버와 데이터를 교환하는 방법이다. 즉, 빠르게 동적 웹페이지를 생성하는 기술이다.

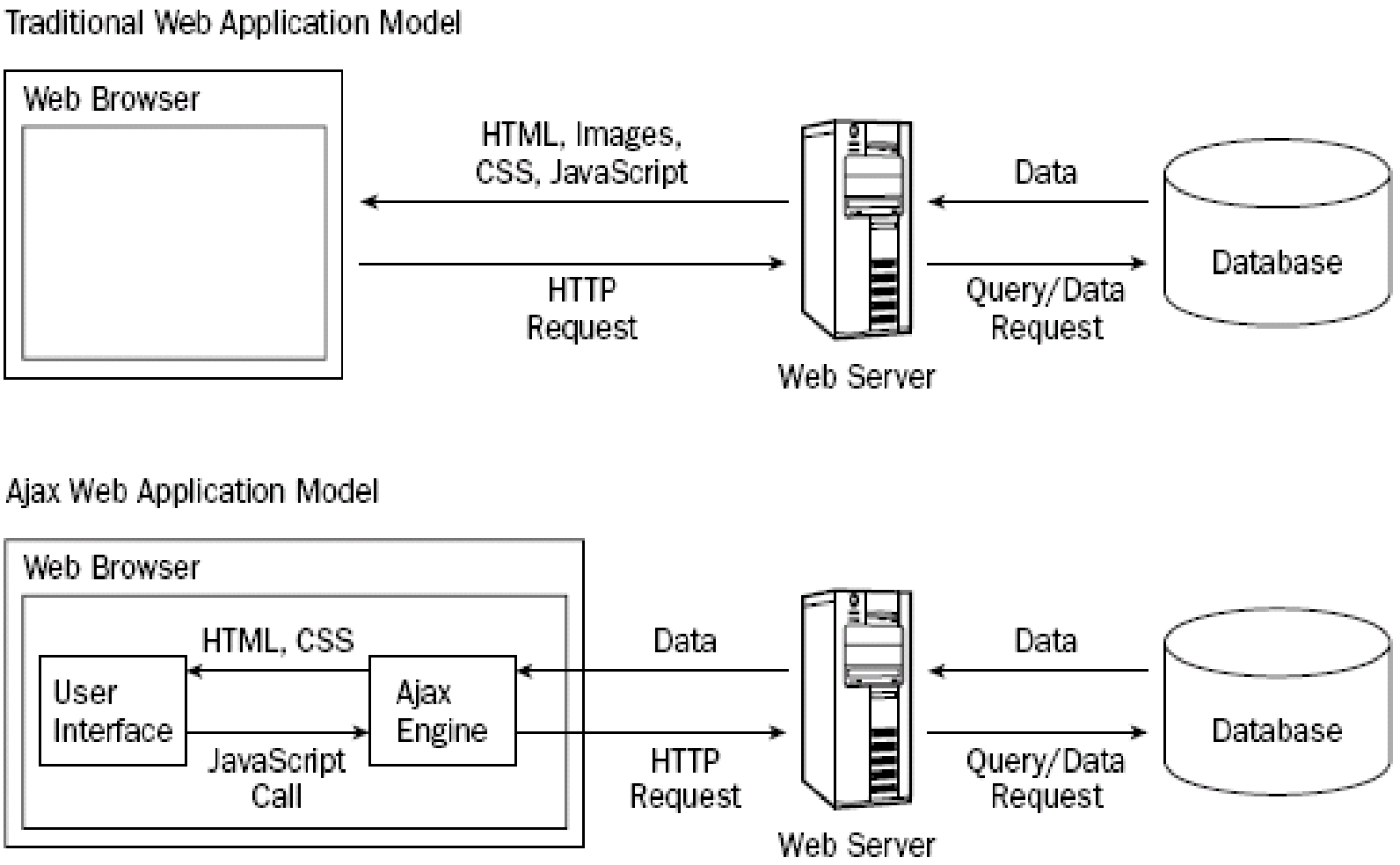






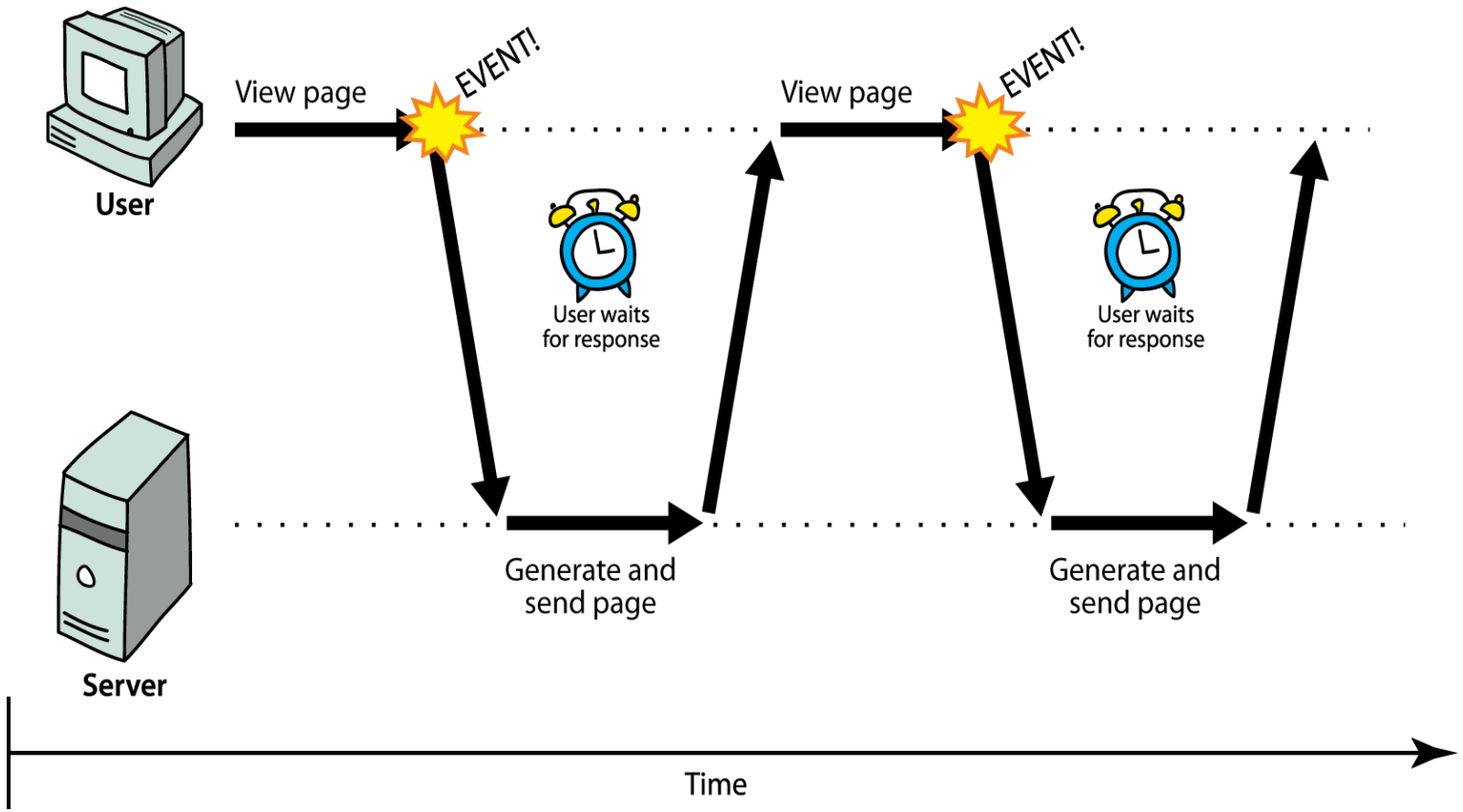
AJAX

- 고전적 웹 통신과 AJAX 웹 통신



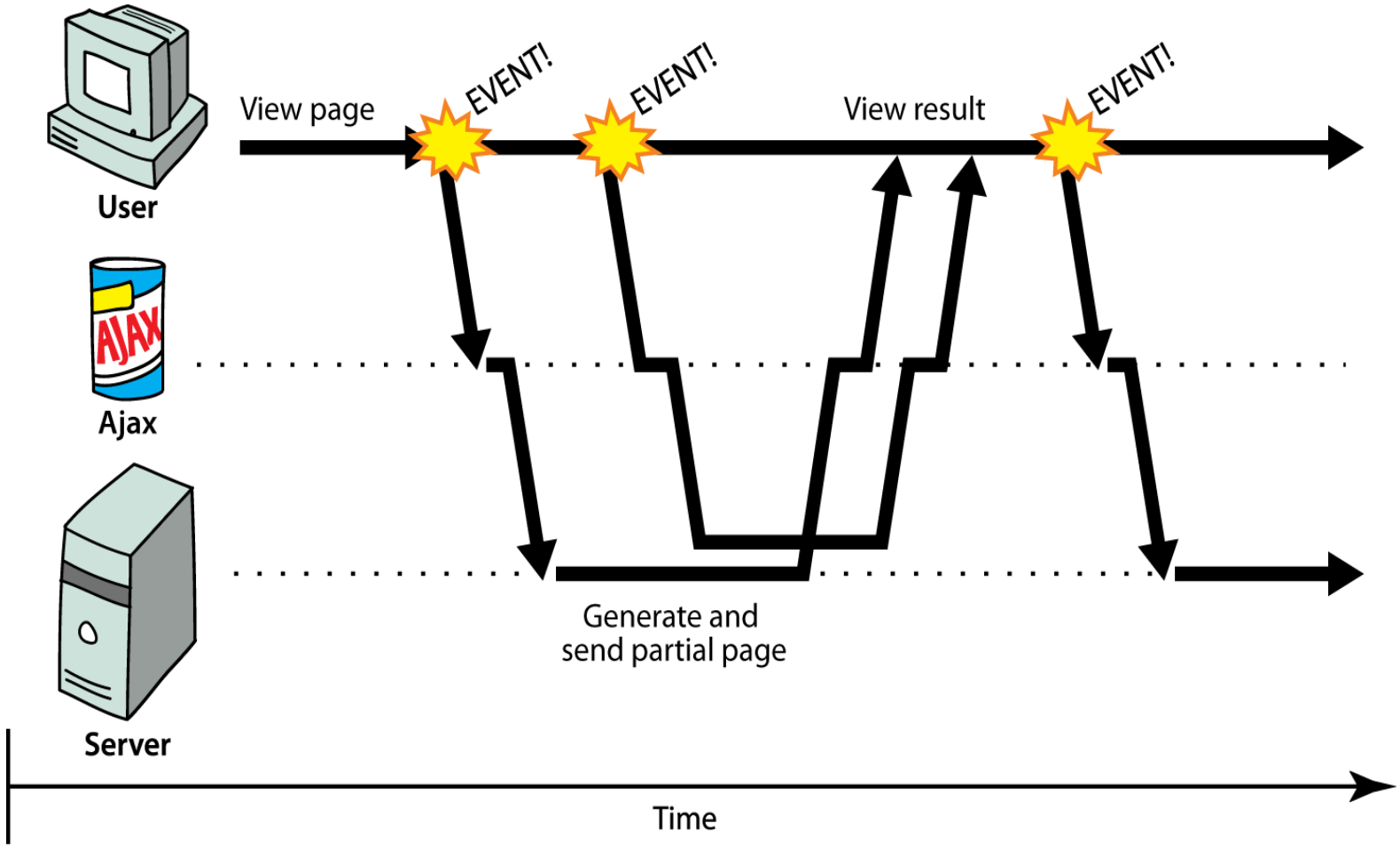
AJAX

- 고전적 웹 통신(동기)



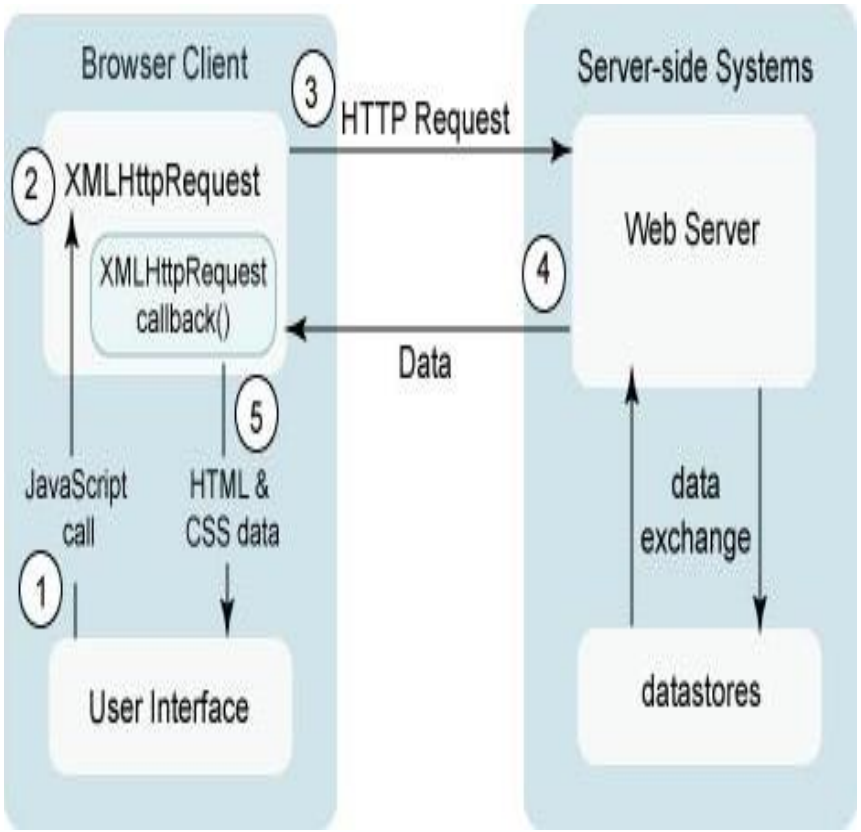
AJAX

- AJAX 웹 통신(비동기)



AJAX

- AJAX의 동작과정



① 이벤트 발생에 의해 이벤트핸들러 역할의 JavaScript 함수를 호출한다.

② 핸들러 함수에서 XMLHttpRequest 객체를 생성한다. 요청이 종료되었을 때 처리할 기능을 콜백함수로 만들어 등록한다.

③ XMLHttpRequest 객체를 통해 서버에 요청을 보낸다.

④ 요청을 받은 서버는 요청 결과를 적당한 데이터로 구성하여 응답한다.

⑤ XMLHttpRequest 객체에 의해 등록된 콜백함수를 호출하여 응답 결과를 현재 웹 페이지에 반영한다.

AJAX

- XMLHttpRequest 객체
  - 서버 측과의 비동기 통신을 제어하는 것은 XMLHttpRequest 객체의 역할이다.
  - XMLHttpRequest 객체를 이용함으로써 지금까지 브라우저가 실행해 온 서버와의 통신 부분을 JavaScript가 제어할 수 있게 된다.
  - XMLHttpRequest 객체 생성 : new XMLHttpRequest()

분류	멤버	개요
프로퍼티	**onreadystatechange**	통신상태가변화된타이밍에호출되는이벤트핸들러
	readyState	HTTP 통신상태를취득
	status	HTTP Status코드를취득
	responseType	응답받으려는 컨텐트 타입. "arraybuffer", "blob", "document", "json", and "text"
	responseText	응답본체를 plaintext로취득
	responseXML	응답본체를 XML(XMLDocument 객체)로취득
	response	지정된 응답 타입에 따른 응답객체
	upload	XMLHttpRequestUpload 객체 제공
메서드	abort()	현재의비동기통신을중단
	getAllResponseHeaders()	수신한모든 HTTP 응답헤더를취득
	getResponseHeader(header)	지정한 HTTP 응답헤더를취득
	open(...)	HTTP 요청을초기화
	setRequestHeader(header, value)	요청 시에송신하는헤더를추가
	send([body])	HTTP 요청을송신(인수 body는요청본체)

AJAX

- FormData 객체
  - FormData 객체는 폼의 각 필드와 값을 나타내는 키/값 쌍들의 집합을 쉽게 구성할 수 있는 방법을 제공하며, 이를 이용하면 데이터를 "multipart/form-data" 형식으로 XMLHttpRequest의 send() 메소드를 사용하여 쉽게 전송할 수 있다.

```
// aFile은 input type="file" 이나 드래그 앤 드롭된 파일로 부터 온 값
var formdata = new FormData();
formdata.append("nickname", "Foooobar");
formdata.append("website", "http://hacks.mozilla.org");
formdata.append("media", aFile);
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://foo.bar/upload.php");
xhr.send(formdata);
```

  - HTML from 엘리먼트의 DOM 객체는 폼의 데이터를 FormData 객체로 얻게 해 주는 getFormData() 메소드를 제공한다.

```
var formElement = document.getElementById("myFormElement");
formData = formElement.getFormData();
formData.append("serialnumber", serialNumber++);
xhr.send(formData);
```

AJAX

- readyState 값

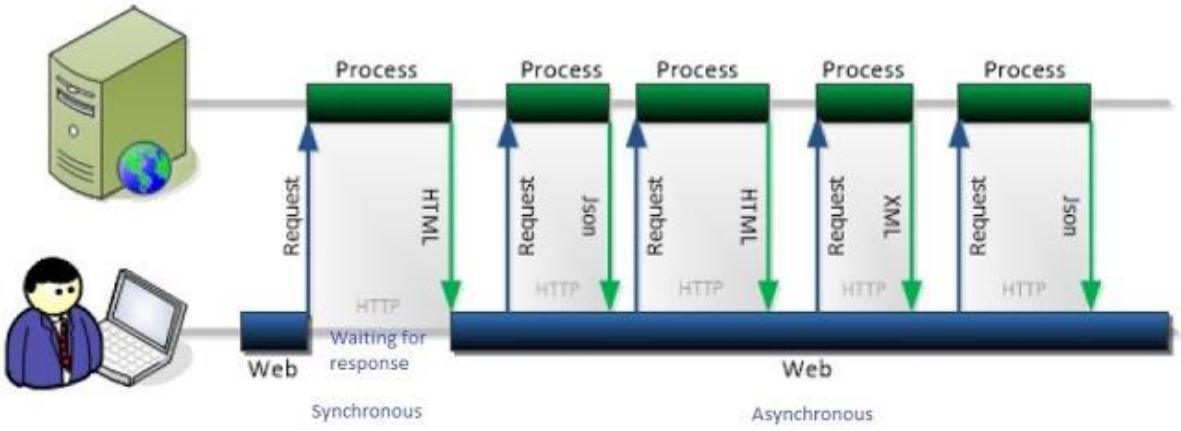
반환값	개요
0	미초기화(open 메서드가호출되지않음)
1	로드 중(open 메서드는호출됐지만, send 메서드는호출되지않았다)
2	로드 완료(send 메서드는호출됐지만, 응답스테이터스/헤더는 미취득)
3	일부 응답을 취득(응답스테이터스/헤더만취득, 본체는미취득)
4	모든 응답데이터를 취득 완료

[XMLHttpRequest 객체에서 제공되는 이벤트 관련 속성]

- onloadstart
- onprogress
- onabort
- onerror
- onload
- ontimeout
- onloadend
- onreadystatechange

AJAX

- open()와 send() 메서드
  - open(HTTP 메서드, URL [, 비동기 모드 통신 여부])
    - HTTP 메서드 : 요청 방식(GET, POST, PUT, DELETE..)
    - URL : AJAX 로 요청하려는 서버의 대상 페이지
    - 비동기 모드 통신 여부 : true(비동기통신), false(동기통신)
  - send([요청 파라미터])
    - POST 의 경우 Query 문자열을 인수로 지정
    - ArrayBufferView, Blob, Document, DOMString, FormData, null 이 올 수 있다.



jQuery 의 AJAX 지원 API

Ajax 메소드	기능/예

$.ajax( )	모든 Ajax 메소드의 기본이 되는 메소드
	예) $.ajax({
	url: 'service.php',
	success: function(data) {
	$('#area').html(data);
	}
	});

$.get( )	GET 방식의 ajax( ) 메소드 예) $.get('sample.html', function(data) { $('#area').html(data); });
$.post( )	POST 방식의 ajax( ) 메소드 예) $.post('sample.html', function(data) { $('#area').html(data); });
$.getJSON( )	JSON 형식으로 응답 받는 ajax( ) 메소드 예) $.getJSON('sample.json', function(data) { $('#area').html(' ' + data.age + ' '); });
load( )	서버로부터 데이터를 받아서 일치하는 요소 안에 HTML을 추가 예) $('#area').load('sample.html', function( ) { ; });
$.getScript( )	자바스크립트 형식으로 응답 받는 ajax( ) 메소드 예) $.getScript('sample.js', function( ) { ; });
$.ajaxSetup( )	ajax( ) 메소드의 선택 사항들에 대한 기본값 설정 예) $.ajaxSetup({ url: 'service.php' });

jQuery 의 AJAX 지원 API

- $.ajax( ) 메서드
  - 모든 AJAX 메서드가 내부적으로는 사용하는 기본 메서드
  - AJAX 요청을 기본적인 부분부터 직접 설정하고 제어할 수 있어 다른 AJAX 메서드로 할 수 없는 요청도 수행 가능
  - $.ajax() 메서드의 기본 형식

```
$.ajax( options ) ;
```

```
$.ajax({ url: URL주소 [,type: 요청방식] [,data: 요청내용] [,timeout: 응답제한시간] [,dataType: 응답데이터유형] [,async: 비동기여부] [,success: 성공콜백함수] [,error: 실패콜백함수] });
```

- $.ajax( ) 메서드 선택 항목들(options)을 맵 형식으로 명세

선택항목 : 항목값	의미
url : URL 주소	요청이 보내질(주로 서버)의 URL 주소(필수 항목, 기본값: 현재페이지) 예) "sample.php", "sample.html", "sample.xml"
type : 요청방식	요청을 위해 사용할 HTTP 메소드 예) "get"(기본값), "post"
data : 요청내용	서버로 전달되는 요청 내용(제이쿼리 객체맵이나 문자열)
timeout : 응답제한시간	요청 응답 제한 시간(밀리초) 예) 20000
dataType : 응답데이터유형	(서버로부터의) 반환될 응답 데이터의 형식 예) "xml", "html", "json", "jsonp", "script", "text"
Async : 논리값	요청이 비동기식으로 처리되는지 여부(기본값: true)
success : function(data)	요청 성공 콜백함수(data: 서버 반환 값)
error : function( )	요청 실패 콜백함수

jQuery 의 AJAX 지원 API

- $.getJSON( ) 메서드
  - GET 요청 방식으로 서버로부터 JSON 형식의 데이터를 요청

```
$.getJSON( url [, data] [,function(data)] ) ;
```

  - $.getJSON 메서드 입력인자

인자	의미
url	요청이 보내질(주로 서버)의 URL 주소(필수 항목, 기본값: 현재 페이지) 예) "sample.json"
data	서버로 전달되는 요청 내용(제이쿼리 객체 맵이나 문자열)
function(data)	요청 성공 콜백 함수 (data: 서버 반환 값)

jQuery 의 AJAX 지원 API

- $.load( ) 메서드
  - 서버로부터 데이터를 받아오는 가장 간단한 메서드로 많이 이용
  - 서버로부터 데이터를 받아 메서드를 실행하는 대상 엘리먼트에 직접 추가 -> 복잡한 선택 사항을 설정하지 않고도 빠르고 간단하게 웹 페이지의 동적 갱신이 가능
  - 요청이 성공하면 메서드가 실행되는 대상 엘리먼트 내용이 서버에서 응답 받은 HTML5 마크업 데이터로 대체

```
$.load( url [, data] [,function(data)] ) ;
```

  - $.load( ) 메서드 선택 항목

인자	의미
url	요청이 보내질(주로 서버)의 URL 주소(필수 항목, 기본값: 현재페이지) 예) "sample.html"
data	서버로 전달되는 요청 내용(제이쿼리 객체맵이나 문자열)
function(data)	요청 성공 콜백 함수(data: 서버 반환 값)

AJAX

- Same Origin Policy(SOP)
  - 브라우저에서 보안상의 이슈로 동일 사이트의 자원(Resource)만 접근해야 한다는 제약이다.
  - AJAX는 이 제약에 영향을 받으므로 Origin 서버가 아니면 AJAX 로 요청한 컨텐츠를 수신할 수 없다.

- Cross Origin Resource Sharing(CORS)
  - 초기에는 Cross Domain이라고 하였다.(동일 도메인에서 포트만 다른 경우, 로컬 파일인 경우 등으로 인해 Origin이라는 으로 용어 통일됨)
  - Origin 이 아닌 다른 사이트의 자원을 접근하여 사용한다는 의미이다.
  - Open API 의 활성화와 공공 DB 의 활용에 의해서 CORS 의 중요성이 강조되고 있다.
  - HTTP Header에 CORS 와 관련된 항목을 추가한다.

```
response.addHeader("Access-Control-Allow-Origin", "*");
```