



# HTML5/CSS3/JavaScript Programming(중급)

- 진정원 / [meteorstartup@gmail.com](mailto:meteorstartup@gmail.com)
- Meteor Startup 대표 (<http://meteorstartup.com>)
- 임베디드 / 모바일
- 리눅스 엔지니어링 / DB
- 웹 (with Meteor.js)



- 미림 마에스터고 리눅스 네트워크
- KT DataSystem OS X/iOS
- Meteor.js Getting Started
- RHCSA / RHCE



1. 제1장 웹 아키텍처의 이해

2. 제2장 HTML5

3. 제3장 JavaScript와 jQuery

4. 제4장 HTML5 확장 API

5. 제5장 HTML5 구현 사례

6. 제6장 Extend Javascript



## HTML5?

- HTML 2.0 (IETF) ~ HTML 4 (W3C)
- HTML 5 (WHATWG)
- 변형중인 HTML

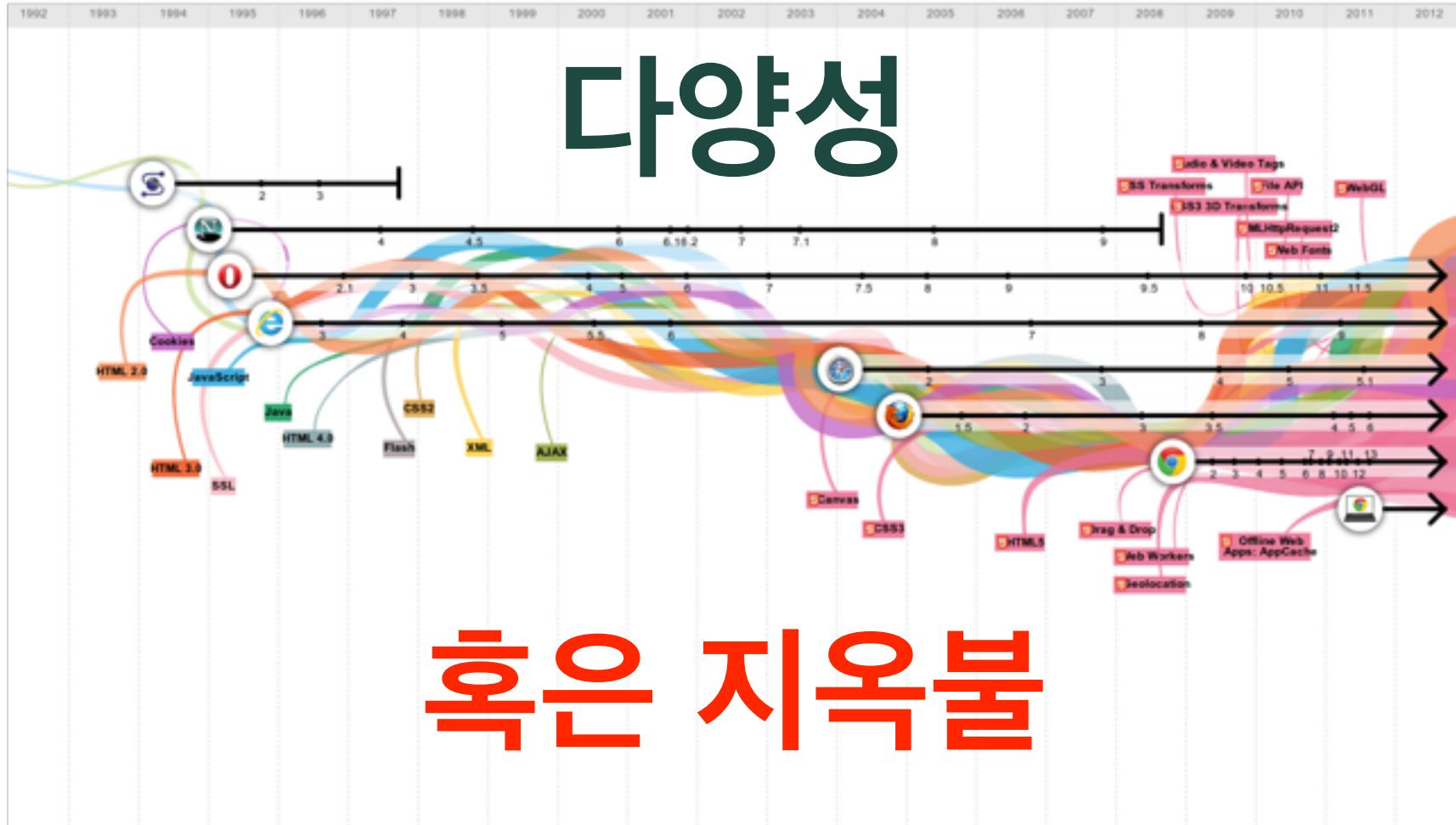


# HTML?

- 선언형 언어
- 각 브라우저가 구동
- 다양성



# 다양성



# WEB

- 범용성
- 간단할 수 있는 구성
- 낮은 초기 개발 비용



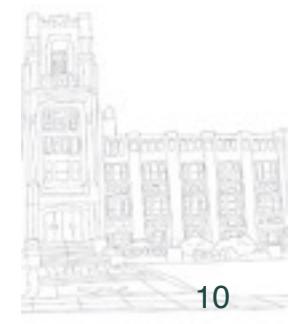


## 제1장 웹 아키텍처의 이해

1. Web Architecture & Client/Server 환경

2. Web Programming

3. 자바 기반 서버 환경 구축 및 사용



# Web Architecture & Client/Server 환경

## □ 웹 어플리케이션 정의

- 인터넷으로 연결된 환경에서 웹 브라우저를 이용해서 사용할 수 있는 프로그램

## □ 웹 어플리케이션 구성 요소

### ■ 클라이언트

- 서비스를 요청
- 웹브라우저 사용 (예 : 인터넷 익스플로러, 크롬, 사파리, 파이어폭스, 오페라 등)

### ■ 서버

- 서비스를 제공
- 웹서버 (예 : IIS, Apache, Apache Tomcat 등)
  - HTTP프로토콜을 기반으로 하여 클라이언트의 웹브라우저 요청을 받아 결과를 바로 웹브라우저에 전송하거나 웹 어플리케이션에서 실행한 결과를 전달 받아 웹브라우저에 전송하는 서비스 기능 수행
- 웹 어플리케이션 서버(WAS : Web Application Server)
  - 웹 서버의 요청으로 프로그래밍 언어(C, JAVA, C#, ASP, JSP, PHP등)로 만들어진 웹 어플리케이션을 실행하고 그 결과를 웹서버로 전달하는 역할 수행
- 데이터베이스 시스템 (DBMS)
  - 웹 어플리케이션 서버를 통해 제공되는 서비스에 필요한 데이터를 저장, 관리 및 운영

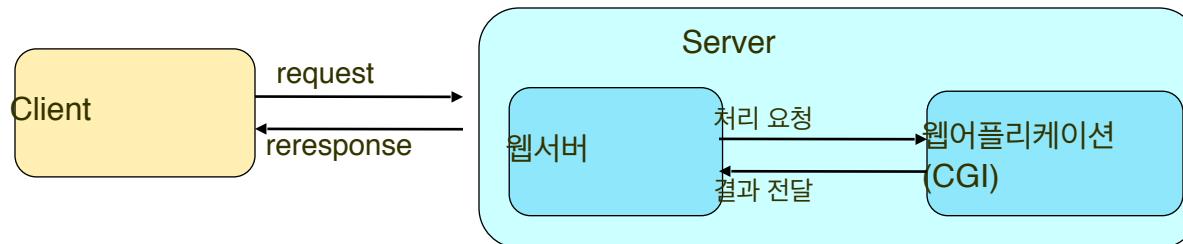


# Web Architecture & Client/Server 환경

## □ 웹 어플리케이션의 구현 방식

### ■ CGI 방식

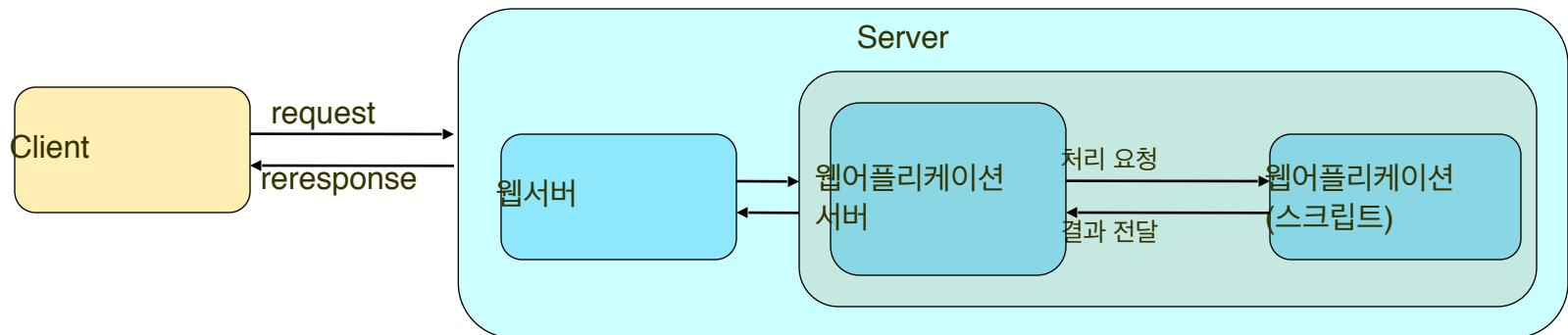
- 웹서버가 직접 어플리케이션을 호출하는 구조
- 웹브라우저의 요청수 만큼 프로세스를 생성해서 실행하기 때문에 메모리 소모가 큼
- 실행코드를 사용해서 동작
  - 별도로 컴파일 된 실행프로그램을 이용해서 동작하며 코드 변경시 별도로 재 컴파일을 해야 한다.



### ■ 어플리케이션 서버 방식

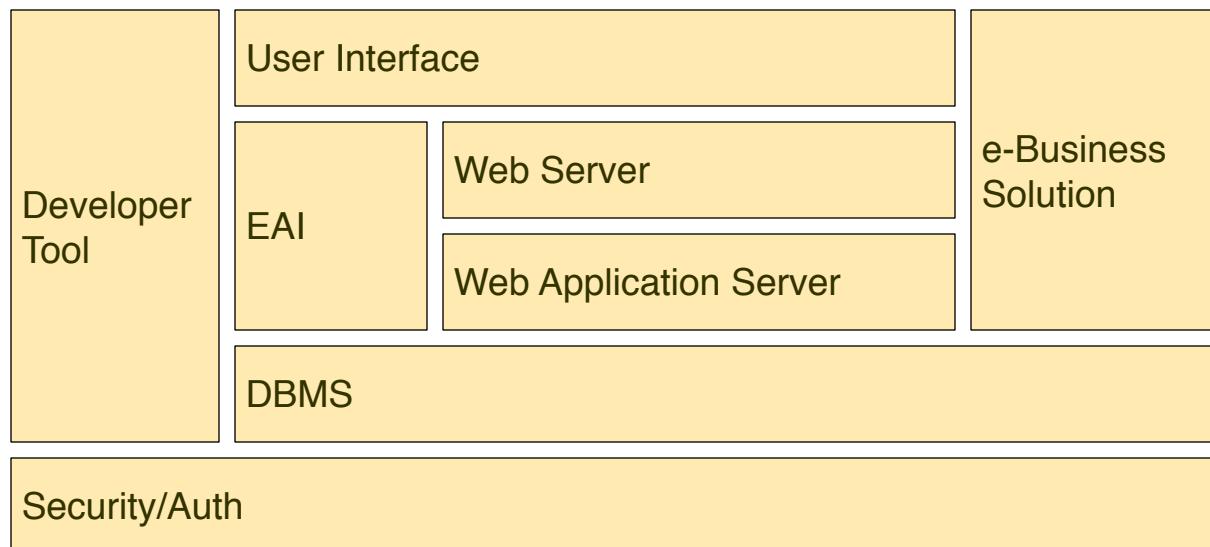
- 웹서버가 웹 어플리케이션 서버를 통해서 간접적으로 호출하는 구조
- 웹브라우저가 같은 프로그램을 동시에 요청해도 하나의 프로세스를 통해서 처리하기 때문에 CGI방식에 비해 메모리 소모가 적음
- 스크립트 코드를 사용해서 동작
  - 스크립트 코드는 최초 실행시에 한번만 컴파일하는 과정을 수행하며 그 다음부터는 컴파일 과정을 생략하고 바로 실행하며 코드 변경시에는 변경된 코드에 대해서 한번 컴파일 하는 과정을 수행한다.

# Web Architecture & Client/Server 환경



## □ Web Architecture 프레임워크

- 일반적인 웹 프레임워크의 구성요소는 다음과 같다.

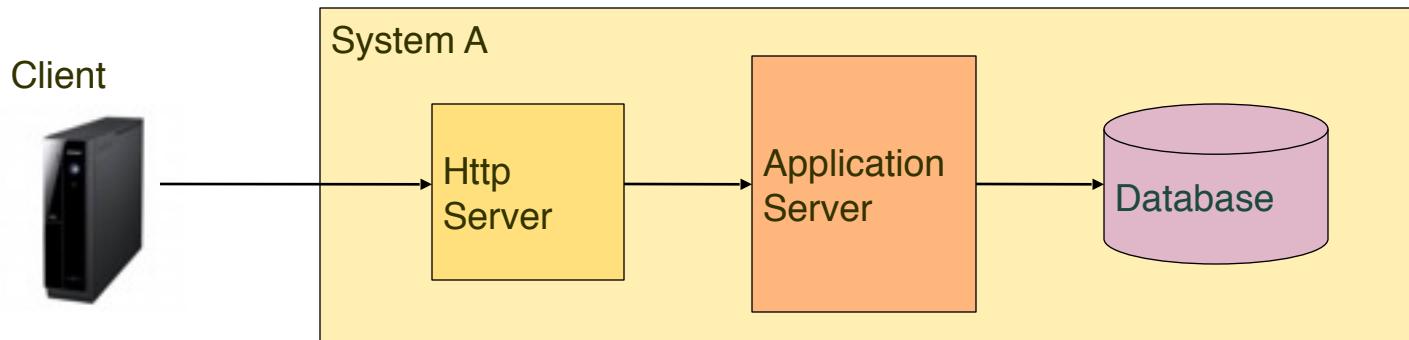


# Web Architecture & Client/Server 환경

## □ Web Architecture 구조

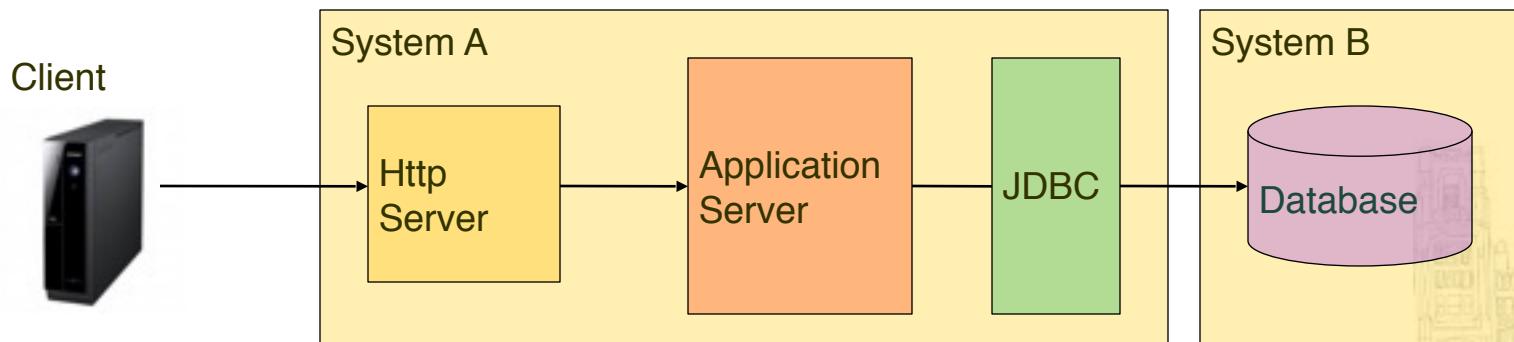
### ■ 1-Tier

- 하나의 시스템에 웹기반 아키텍처 요소를 모두 포함하는 구조



### ■ 2-Tier

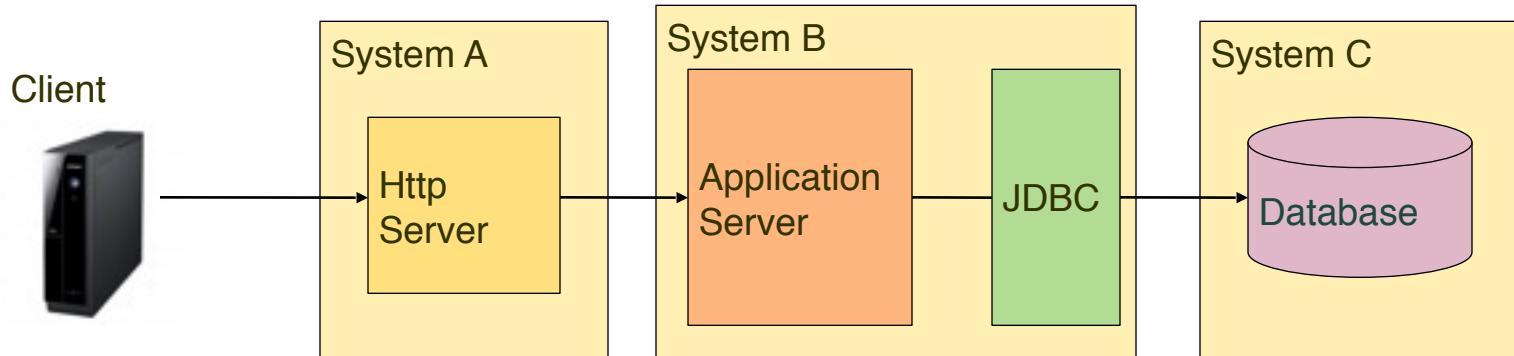
- 웹 서버와 WAS를 하나의 머신에 배치하고 DBMS를 나머지 머신에 배치하는 구조



# Web Architecture & Client/Server 환경

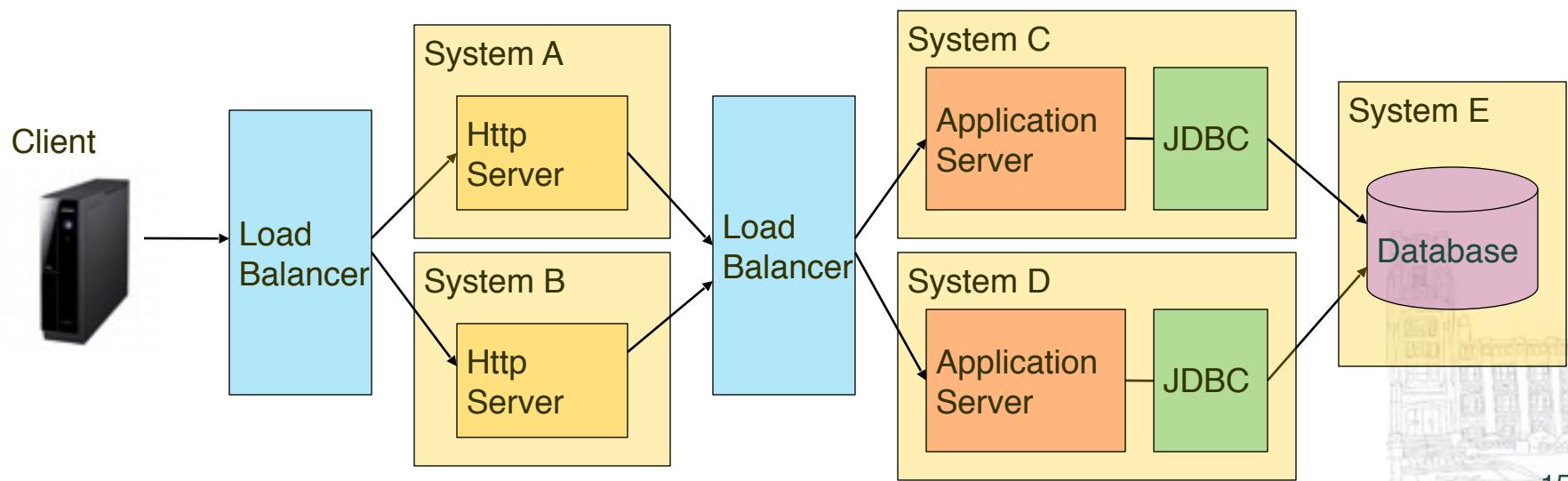
## ■ 3-Tier

- 서버, WAS, DBMS 를 모두 다른 머신에 배치하는 구조



## ■ 대규모 3-Tier

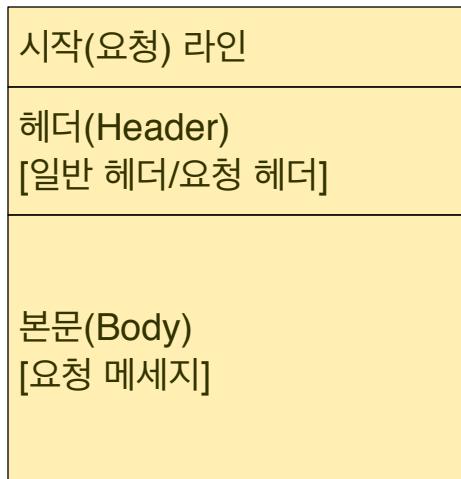
- 로드 밸런스를 이용하여 3-Tier구조를 확장하는 구조



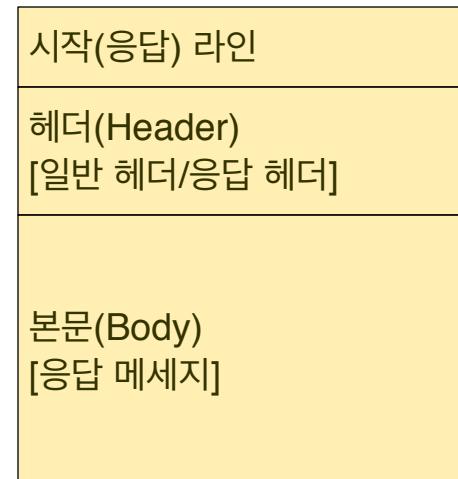
# Web Architecture & Client/Server 환경

## □ HTTP 프로토콜

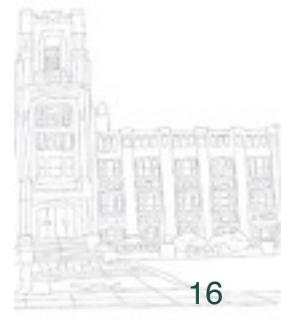
- 웹에서 클라이언트와 서버간에 하이퍼 텍스트(Hyper Text) 문서를 교환하기 위한 통신 규약
- HTTP 메시지는 시작라인, 헤더, 본문으로 구성
  - 시작라인
    - 작성된 메시지가 요청인지 응답인지 구분하는 정보, 요청 URL
  - 헤더
    - 서버정보, 수행 날짜, 브라우저 버전 등 부가적인 정보
  - 본문
    - 요청이나 응답에 필요한 내용
- 종류



- HTTP 요청 메시지 -



- HTTP 응답 메시지 -



# Web Architecture & Client/Server 환경

## ■ Http 요청(Request) 메시지

- HTTP method(GET/POST), URL, 서버에 전송할 폼(form) 파라미터로 구성
- GET 방식
  - 전송할 파라미터 값을 URL 정보에 연결해서 서버로 전송
  - 최대길이 256바이트
  - 본문(Body)가 필요없으므로 전송속도는 POST보다 빠름
  - 폼 입력정보 외부 노출되어 보안에 취약

시작(요청) 라인	GET /home/form.jsp?irum=hong&email=hong@daum.net HTTP/1.1
요청 헤더	Host : www.example.com User-Agent : Mozilla/5.0... ....

### • POST 방식

- 전송할 파라미터 값을 본문(Body)에 저장해서 서버로 전송
- 전송길이 제약 없음
- 폼 입력정보가 감추어져 서버로 전송되므로 GET 방식보다 안전

시작(요청) 라인	POST /home/form.jsp HTTP/1.1
요청 헤더	Host : www.example.com User-Agent : Mozilla/5.0... ....
본문(Body)	irum=hong&email=hong@daum.net ....

# Web Architecture & Client/Server 환경

## ■ HTTP 응답(Response) 메시지

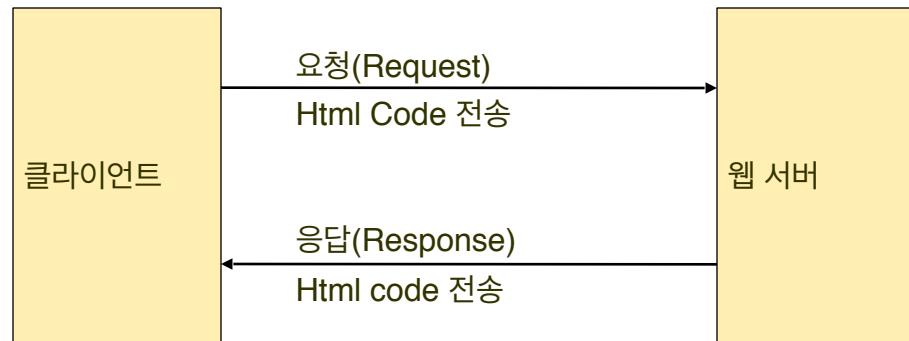
- 요청에 대한 서버의 처리 여부를 표시하는 상태코드(예 : HTTP 200(OK), 404(Not Found), 500(Internal Server Error) 등)와 웹 서버가 응답해주는 컨텐츠의 타입정보(예 : text/html), 컨텐츠의 내용으로 구성

시작(응답) 라인	HTTP/1.1 200 OK
응답 헤더	Server : Apache-Coyote/1.1 Content-Type : text/html;charset=utf-8 ....
본문(Body)	....

# Web Programming

## □ 웹 프로그래밍 이란?

- 웹 상에서 사용자와 기업 또는 사용자들간의 연결을 가능하게 하는 프로그래밍 언어
- 일반적으로 웹 프로그래밍은 클라이언트(Client)/서버(Server)의 방식으로 구축되어 동작
- 클라이언트가 서버에 정보를 요청하면 서버는 요청에 대한 응답을 위해 필요한 작업을 수행한 후 HTML코드 형식(HTML, CSS, Javascript 등)으로 응답 정보를 클라이언트에게 전송한다



- 클라이언트/서버 구조 -

# Web Programming

## □ 웹 프로그래밍 언어

- HTML(HyperText Markup Language)
  - 인터넷상에서 문서들을 링크에 의해 서로 연결하기 위해 만들어진 언어
  - 텍스트 기반의 웹 문서 작성이나 정적인 데이터들을 처리
- CGI(Common Gateway Interface)
  - Common Gateway Interface의 약어로 응용 프로그램과 웹 서버 사이의 정보를 주고받는 방식이나 규약들을 정해 놓은 것을 말함
  - 초기 CGI 프로그래밍에 사용된 언어
    - C, Perl
  - 주로 파일형식의 DB 사용(텍스트 파일, MDB 등)했으며 데이터베이스 연동이 불편
- ASP(Active Server Page)
  - 동적인 웹 페이지의 구현을 위해 Visual Basic 언어를 기반으로 만들어진 VBScript라는 스크립트 언어를 사용해서 구성된 웹 프로그래밍 언어
  - IIS(Internet Information Server)에서만 실행
- PHP(Personal Hypertext Preprocessor)
  - C를 기반으로 만들어진 언어로 빠르게 동작
  - 서버 측의 지원 인프라가 부족하며 확장성이 떨어지고, 기업형의 복잡한 시스템 구조에 적용하기가 힘들며 보안이 취약



## ■ JSP

- 초창기에 서블릿(Servlet)이라는 동적 웹 구현 기술로 발표
- 인터페이스의 효율적인 구현을 위해 JSP 개발
- 서블릿과 함께 구동함으로써 서블릿의 기능을 그대로 사용 가능
- 자바빈즈(JavaBeans), EJB같은 기술로 보다 강력한 객체 지향적 지원 가능
- JSTL을 지원하게 되면서 웹 프로그램의 가독성이 좋아지고 유지 및 보수가 편리



- WebStorm: <https://www.jetbrains.com/webstorm/download/> (Windows)
  - <https://git-scm.com/download/win> (64bit Git for Windows)
  - <https://www.meteor.com/install> (On Windows)
- » 각 프로그램 다운로드 및 설치. (설치 중 옵션들은 Default 상태로 Next 클릭)



# 자바 기반 서버 환경 구축 및 사용

## ▣ 자바 기반 서버 환경

### ■ 개발도구

- JDK
- Apache Tomcat
- Eclipse

## ▣ 설치 순서 및 관련 사이트

### ■ JDK 설치

- 사이트 - <http://www.oracle.com/technetwork/java/index.html>

### ■ Apache Tomcat 설치

- 사이트 - <http://tomcat.apache.org/>

### ■ Eclipse

- 사이트 - <http://www.eclipse.org/>

### ■ Eclipse에 Apache Tomcat 서버 등록



## JDK 설치

- 1. 오라클사이트 접속후 파일 다운로드(jdk-8u20-windows-i586.exe)
  - 사이트 주소 : <http://www.oracle.com/technetwork/java/index.html>

The screenshot shows the Oracle Java Technology Network homepage. At the top, there's a navigation bar with links for Sign In/Register, Help, Country, Communities, I am a..., I want to..., Search, Products, Solutions, Downloads, Store, Support, Training, Partners, About, and OTN. Below the navigation is a banner for the Virtual Technology Summit. On the right side, there's a "Software Downloads" section with a red header. This section includes "Top Downloads" (Java SE, Java EE and GlassFish, JavaFX, Java ME, JDeveloper 11g and ADF, Enterprise Pack for Eclipse, NetBeans IDE, Pre-Built VM for Java Devs) and "New Downloads" (Java SE 8 Update 45, Java SE 7 Update 80 (PSU), Java SE 7 Update 79 (CPU), Java SE Embedded 8 Update 33, Java SE Embedded 7 Update 75). A blue oval highlights the "Java SE 8 Update 45" link under "New Downloads". At the bottom of the page, there are links for Essential Links, Developer Spotlight, Blogs, and Technologies.

Virtual Technology Summit

Join the next VTS on July 14! You will learn about Java 8 lambdas, 3D modeling and printing and Docker

Posted 06/10/15 // Tags: java, JavaOne, VTS // Headlines Archive

What's New

Java in the Cloud: Rapidly develop and deploy Java business applications in the cloud. Try it FREE for 30 days.

Essential Links

Developer Spotlight

Blogs

Technologies

Software Downloads

View All Downloads

New Downloads

Java SE 8 Update 45  
Released 3/03/15

Java SE 7 Update 80 (PSU)  
Released 1/20/15

Java SE 7 Update 79 (CPU)  
Released 1/20/15

Java SE Embedded 8 Update 33  
Released 1/16/15

Java SE Embedded 7 Update 75  
Released 10/16/15

\*Java CPU and PSU Releases Explained

# JDK 설치

www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html

The screenshot shows the Oracle Java SE Downloads page. On the left, there's a sidebar with links to Java SE, Java EE, Java ME, Java SE Support, Java SE Advanced & Suite, Java Embedded, Java DB, Web Tier, Java Card, Java TV, New to Java, Community, and Java Magazine. The main navigation bar includes Sign In/Register, Help, Country, Communities, I am a..., I want to..., Search, Products, Solutions, Downloads, Store, Support, Training, Partners, About, and OTN. Below the navigation is a breadcrumb trail: Oracle Technology Network > Java > Java SE > Downloads. The central area features tabs for Overview, Downloads (which is selected), Documentation, Community, Technologies, and Training. A large section titled "Java SE Downloads" contains two main download options: "Java" (with a "DOWNLOAD" button) and "NetBeans" (with a "DOWNLOAD" button). To the right, there's an advertisement for "JavaOne" with the date Junho 23-25, 2015, São Paulo, Brasil, and a "#javatech" tag. At the bottom, there's a section for "Java Platform, Standard Edition" with a summary of Java SE 8u45, a "Learn more" link, and a list of links: Installation Instructions, Release Notes, and Oracle License. The "JDK DOWNLOAD" button in this section is circled with a blue oval.

# JDK 설치

## ■ 운영체제에 따라 선택

www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

**Java SE Development Kit 8 Downloads**

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK MD5 Checksum

**Looking for JDK 8 on ARM?**  
JDK 8 for ARM downloads have moved to the [JDK 8 for ARM download page](#).

**Java SE Development Kit 8u45**

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement    Decline License Agreement

Product / File Description	File size	Download
Linux x86	146.89 MB	jdk-8u45-linux-i586.rpm
Linux x86	166.88 MB	jdk-8u45-linux-i586.tar.gz
Linux x64	145.19 MB	jdk-8u45-linux-x64.rpm
Linux x64	165.24 MB	jdk-8u45-linux-x64.tar.gz
Mac OS X x64	221.98 MB	jdk-8u45-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	131.73 MB	jdk-8u45-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	92.9 MB	jdk-8u45-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	139.51 MB	jdk-8u45-solaris-x64.tar.Z
Solaris x64	95.88 MB	jdk-8u45-solaris-x64.tar.gz
Windows x86	175.98 MB	jdk-8u45-windows-i586.exe
Windows x64	180.44 MB	jdk-8u45-windows-x64.exe

**Java Resources**

- Java ME
- Java Card
- NetBeans IDE
- Java Mission Control

**Java APIs**

**Technical Articles**

**Demos and Videos**

**Forums**

**Java Magazine**

**Java.net**

**Developer Training**

**Tutorials**

**Java.com**

**ORACLE TECHNOLOGY NETWORK**

VIRTUAL TECHNOLOGY SUMMIT

February 11th  
February 25th  
March 4th

**REGISTER!**

## 아파치 톰캣 설치

- 1. 아파티치 톰캣 사이트 접속후 파일 다운로드 – Tomcat 8.0 설치
  - 사이트 주소 : <http://tomcat.apache.org/>

The screenshot shows the Apache Tomcat website at <http://tomcat.apache.org/>. The page features the Apache logo (a yellow cat) on the left and the Apache feather logo on the right. A search bar is at the top right. The main content area has a title "Apache Tomcat" and a brief description of the software. On the left, there's a sidebar with links for "Apache Tomcat", "Download", and "Documentation". The "Download" section includes a dropdown menu for selecting a version, with "Tomcat 8.0" highlighted and circled in blue.

Apache Tomcat™ is an open source software implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. The Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket specifications are developed under the [Java Community Process](#).

Apache Tomcat is developed in an open and participatory environment and released under the [Apache License version 2](#). Apache Tomcat is intended to be a collaboration of the best-of-breed developers from around the world. We invite you to participate in this open development project. To learn more about getting involved, [click here](#).

Apache Tomcat powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations. Some of these users and their stories are listed on the [PoweredBy](#) wiki page.

Apache Tomcat, Tomcat, Apache, the Apache feather, and the Apache Tomcat project logo are trademarks of the Apache Software Foundation.

# 아파치 톰캣 설치

## □ 1. 아파티치 톰캣 사이트 접속후 파일 다운로드 – Tomcat 8.0 설치

- windows 환경이라면 Core 파트의 32bit/64bit Windows Service Installer 를 다운로드한다.

The screenshot shows the Apache Tomcat download page at [tomcat.apache.org/download-80.cgi](http://tomcat.apache.org/download-80.cgi). The page displays the 'Mirrors' section for version 8.0.23. It includes a note about using a specific mirror, a dropdown for other mirrors set to <http://mirror.apache-kr.org/>, and a 'Change' button. Below this, the 'Binary Distributions' section lists several download links for Core, Full documentation, and Deployer components. A blue oval highlights the link for the '64-bit Itanium Windows zip (pgp, md5, sha1)' under the Core distributions.

Documentation

- Tomcat 8.0
- Tomcat 7.0
- Tomcat 6.0
- Tomcat Connectors
- Tomcat Native
- Wiki
- Migration Guide

Problems?

- Security Reports
- Find help
- FAQ
- Mailing Lists
- Bug Database
- IRC

Get Involved

- Overview
- SVN Repositories
- Buildbot
- Reviewboard
- Tools

Media

- Blog
- Twitter

Mirrors

You are currently using <http://mirror.apache-kr.org/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are backup mirrors (at the end of the mirrors list) that should be available.

Other mirrors:  [Change](#)

8.0.23

Please see the [README](#) file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
  - [zip \(pgp, md5, sha1\)](#)
  - [tar.gz \(pgp, md5, sha1\)](#)
  - [32-bit Windows zip \(pgp, md5, sha1\)](#)
  - [64-bit Windows zip \(pgp, md5, sha1\)](#)
  - [64-bit Itanium Windows zip \(pgp, md5, sha1\)](#)
  - [32-bit/64-bit Windows Service Installer \(pgp, md5, sha1\)](#)
- Full documentation:
  - [tar.gz \(pgp, md5, sha1\)](#)
- Deployer:
  - [zip \(pgp, md5, sha1\)](#)

## 이클립스 설치

- 1. 이클립스 사이트에서 파일 다운로드
  - 사이트 : <http://www.eclipse.org/>

The screenshot shows the official Eclipse website at [www.eclipse.org](http://www.eclipse.org). The page features a large banner with the text "Eclipse Is..." and a subtext about an amazing open source community. A prominent orange "DOWNLOAD" button is highlighted with a blue oval. The top navigation bar includes links for "GETTING STARTED", "MEMBERS", "PROJECTS", and "MORE". A search bar is also visible.

## 이클립스 설치

- Eclipse IDE for Java EE Developers 를 선택



Package Solutions

Eclipse Luna SR2 (4.4.2) Release for Windows ▾

**Eclipse IDE for Java Developers**

 155 MB 4,000,958 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a CVS client, Git client, XML Editor, Mylyn, Maven integration...

Windows  
32 bit | 64 bit

### Eclipse IDE for Java EE Developers



254 MB 2,161,188 DOWNLOADS

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn...

Windows



32 bit | 64 bit

### JRebel for Eclipse IDE



See Java Code Changes Instantly. Save Time. Reduce Stress.  
Finish Projects Faster!



★  
Promoted  
Download

# Thank You

**HTML**



**JS**



**CSS**



## 제2장 Web 표준과 HTML5 개요

1. Web 표준
2. HTML5 개요
3. HTML5 문서 구조
4. <video>, <audio> 태그
5. Web Form
6. contenteditable 속성





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

- 팀 버너스 리([www 창시자](#)) -



# Web 표준

## □ Web 표준이란

- 특정 브라우저에서만 사용하는 비 표준화된 기술을 배제하고, W3C라는 조직에서 정한 표준 기술만을 사용하여, 웹 문서의 구조와 표현 그리고 동작을 구분해서 사용하는 것이다.
- 웹 문서의 구조를 담당하는 것은 HTML이고 표현을 담당하는 것은 CSS이며 동작을 담당하는 것은 자바스크립트이다.
- 3가지 요소가 유기적으로 결합하여 작동하게 되면, 웹 문서가 가벼워지며, 유지보수 및 수정 시에도 간편하고 빨리 처리할 수 있게 된다.



# Web 표준

## □ Web 표준의 목표

[Web은 어디서든, 누구나 정보를 함께 공유하고 즐길 수 있어야 한다.]



# HTML5 개요

## □ 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 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 화려한 그래픽 효과를 낼 수 있다.



# HTML5 개요

## □ HTML5 의 도입 배경



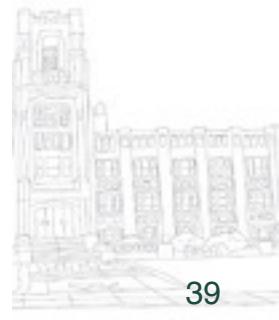
Web Hypertext Application  
Technology Working Group(WWHATWG):  
Apple, Mozillar, Opera 등이 공동으로 발족한  
워킹 그룹으로 HTML을 진화시키고 그 성과를 표  
준화 단체에 제출하는 것을 목적으로 하였다.



# HTML5 개요

## □ HTML5의 주요 특징

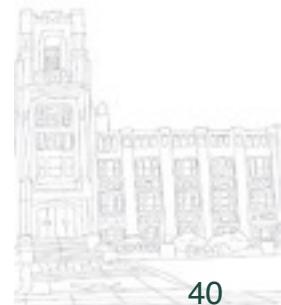
- 웹의 애플리케이션화
  - HTML과 CSS, JavaScript API를 이용한 애플리케이션 개발 가능
- 새롭고 간단한 DOCTYPE과 문자셋(charset) 선언
  - DOCTYPE  
`<!DOCTYPE html>`
  - 문자 셋  
`<meta http-equiv="Content-Type" content="text/html; charset=utf-8">, <meta charset="utf-8">`
- 구조화되고 다양한 기능의 HTML태그 제공
  - 문서를 의미 있게 구조화 및 조직화 할 수 있게 되었다.
- 향상된 웹 폼 양식 제공
  - 기존에는 입력 태입이 text로 제한된 반면 HTML5에서는 email, tel, uri, datetime 등 다양한 형식을 제공
- 멀티미디어 기능 지원
  - 내장된 비디오, 오디오 기능 지원으로 별도의 플러그인 프로그램 없이 동영상이나 음악을 재생할 수 있다.
- Canvas, SVG, WebGL 등을 이용한 다양한 2D/3D 그래픽 기능 제공
- CSS3를 이용한 UI기능 강화
- 오프라인 캐시 기능지원
  - 기존에 방문한 사이트는 오프라인 상태에서도 이용
- 로컬 저장소 제공
- 소켓 프로그래밍 지원



# HTML5 개요

## □ HTML5의 영향

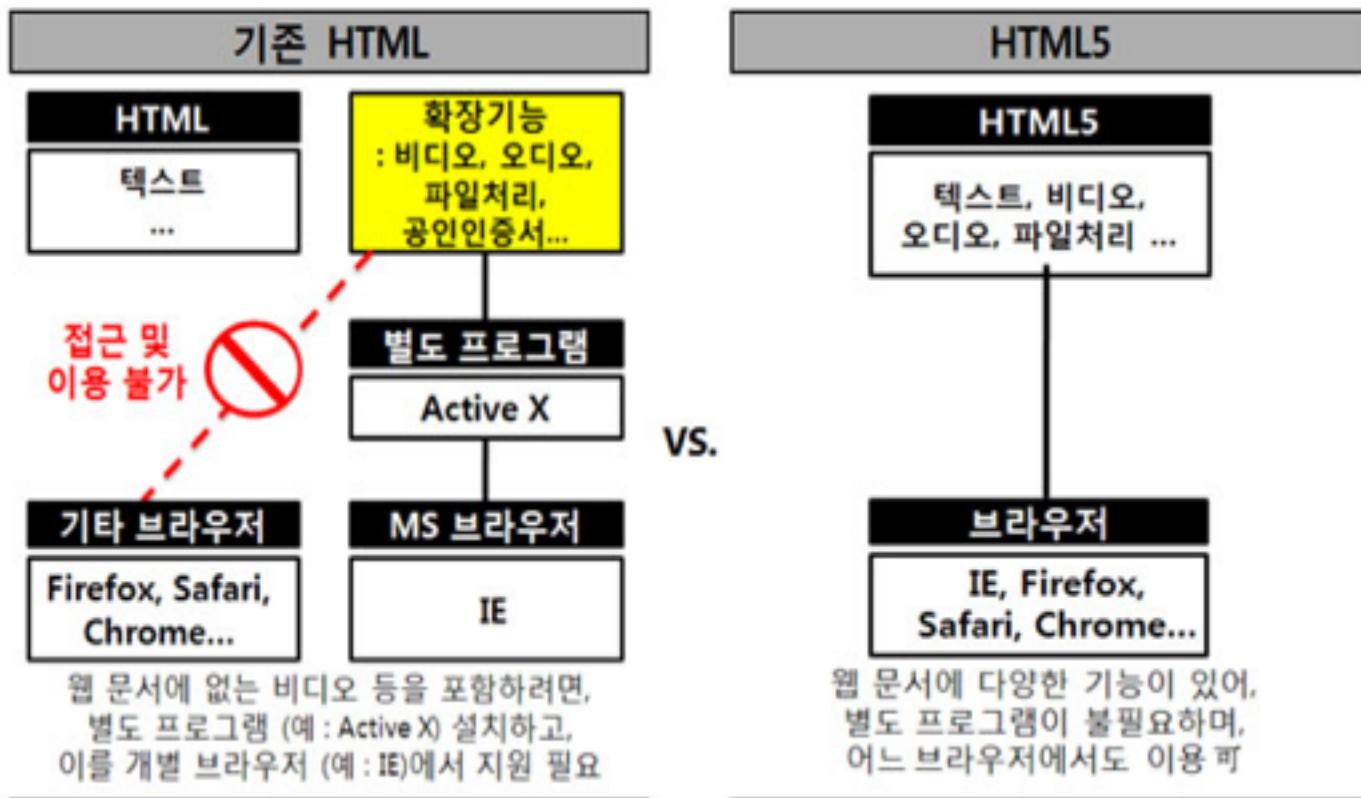
- HTML5에서는 기본적으로 문서의 구조는 HTML 태그(시멘틱태그)를 이용하고 표현은 CSS3 그리고 웹 페이지에서의 동작 구현은 JavaScript로 처리하게 하여 웹 표준에 기반한 웹 사이트를 개발하도록 지원하며 브라우저에 무관하고 디바이스에 무관한 웹 사이트 개발이 가능하게 한다.



# HTML5 개요

## □ HTML5의 영향

- Active-X 와 같은 확장 프로그램에 의존하여 처리되었던 기능들을 HTML5 에서는 표준 API 를 이용하여 개발함으로써 모든 브라우저에서 지원하는 표준 Web 페이지 개발이 가능하다.



▲ 기존 HTML과 HTML5 비교 - 방송통신위원회 제공

# HTML5 개요

## □ HTML5 기본요소

- HTML5 는 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>

# HTML5 개요

## □ HTML5 태그

- 태그 자체에 의미를 부여하는 시맨틱 마크업(Semantic Markup) 사용

## □ HTML5 새로운 태그

- **section**

- 일반적인 문서나 애플리케이션 영역 표시
  - 섹션의 제목을 나타내는 h1~h6와 함께 사용

- **article**

- 뉴스 기사나 블로그 글 같은 독립적인 콘텐츠 표시

- **aside**

- 문서의 주요 부분을 표시하고 남는 사이드 바 콘텐츠 표시

- **hgroup**

- 제목과 그에 관련된 부제목 그룹핑

- **header**

- 헤더 부분으로, 사이트 소개나 내비게이션 등을 표시하거나 내용 중간에서는 머리글 역할 수행

- **footer**

- 사이트 제작자나 저작권 정보 등을 나타낼 때 주로 사용

- **nav**

- 사이트 안의 내비게이션 요소 표시

- **audio, video**

- 멀티미디어 콘텐츠를 표시

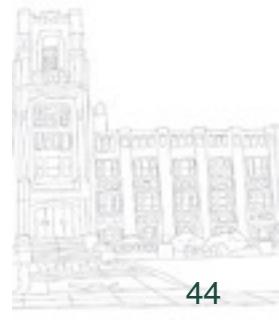


# HTML5 개요

- canvas
  - 웹에 그래픽 표시. API와 함께 사용해 다양한 애플리케이션 구현 가능.
- mark
  - 텍스트에 형광펜으로 칠한 것과 같은 강조 효과
- datalist
  - 사용자가 텍스트 필드에 내용을 입력할 때 선택할 수 있는 값들을 목록으로 보여줍니다.

## □ 새로 추가된 속성

- charset
  - 적용태그 : meta
  - 문자 인코딩 선언
- autofocus
  - 적용태그 : input(type이 hidden일 때를 제외), select, textarea, button
  - 입력 필드에 포커스 설정
- placeholder
  - 적용태그 : input, textarea
  - 입력 필드에 힌트 내용 표시 입력 위해 필드 내부 클릭시 힌트 내용 사라짐.
- required
  - 적용태그 : input(type이 hidden, image, submit 같은 버튼일 때 제외), textarea
  - 필수 입력 필드를 설정



# HTML5 개요

- autocomplete
  - 적용태그 : input
  - 자동 완성 기능 설정
- min, max
  - 적용태그 : input
  - 숫자나 범위를 지정할 때 값의 최소값이나 최대값 지정
- pattern
  - 적용태그 : input
  - 조건을 사용한 일반식(패턴) 표시
- step
  - 적용태그 : input
  - 숫자나 범위를 지정할 때 증감값 표시
- novalidate
  - 적용태그 : form
  - 서버로 양식을 전송하는 동안 유효했는지 보증할 수 없음을 나타냄
- label
  - 적용태그 : menu
  - 메뉴명을 지정
- menifest
  - 적용태그 : html
  - 오프라인 웹 캐시를 사용할 경우 menifest 파일의 경로 지정

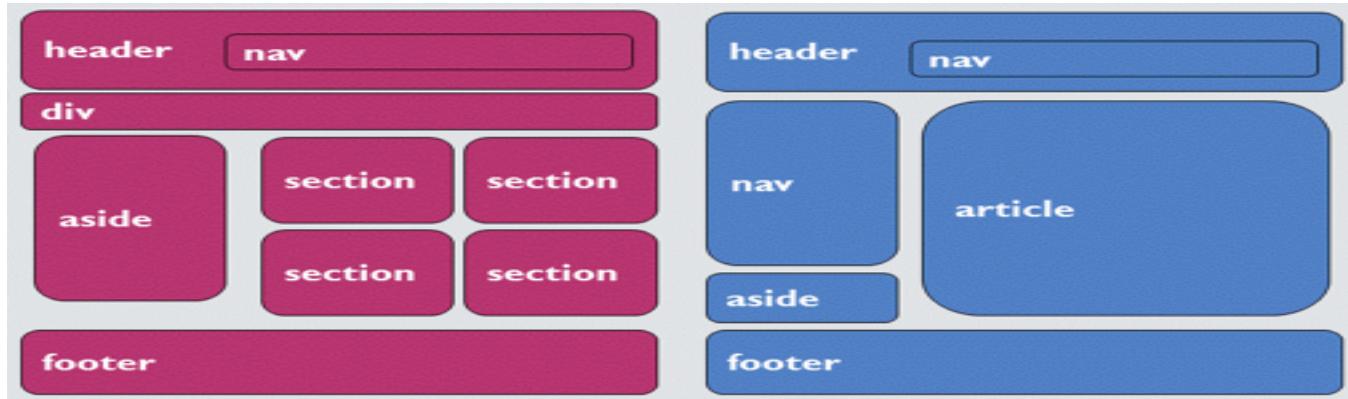


# HTML5 문서 구조

## □ HTML5에서의 DOCTYPE

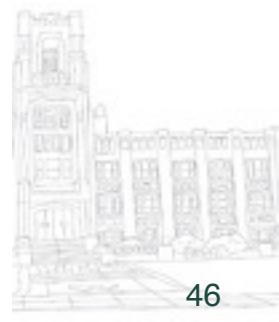
- <!DOCTYPE html>
- 대소문자 상관없이 기술

## □ 시맨틱 태그를 이용한 문서구조



### ▪ <header> 태그

- 머리말 지정
- 사이트 전체의 머리말이나 블로그 전체의 머리말로 사용
- 사이트 전체의 머리말로 사용될 경우 화면 상단이나 좌측에 주로 배치
- 페이지 제목, 소개 글이나 로고 이미지 등이 포함
- <body>,<section>,<article> 태그 안에도 여러 번 포함 가능



# HTML5 문서 구조

- <nav> 태그
  - 페이지 이동을 위한 메뉴 영역을 지정
  - 이전 페이지, 다음 페이지, 특정 페이지나 사이트에 대한 연결 등 네비게이션 요소로 구성
  - <header>,<aside>,<footer> 태그 안에도 여러 번 포함 가능
- <article> 태그
  - 독립적인 개별 내용 영역을 지정
  - 블로그나 댓글, 신문, 잡지의 기사 등을 제공
  - 별도로 배포되거나 재사용 가능한 내용으로 그룹화
  - <section> 태그와 <article> 태그 사이의 중첩 가능
- <section> 태그
  - 형식적인 구분영역으로서 제목을 갖는 연관된 내용 영역을 지정
  - 하나의 제목(<h1>~<h6> 태그)을 중심으로 내용들을 그룹화
  - <section> 태그 안에 또 다른 <section> 태그를 포함한 다양한 태그들이 중첩되어 사용
- <aside> 태그
  - 페이지의 좌,우 측면 공간 같은 보조 영역을 지정
  - 광고나 즐겨찾기 링크, 관련 이미지 정보 등을 제공
  - <section>,<article> 태그 안의 내용과 간접적으로 관련된 내용을 포함
- <footer> 태그
  - 페이지(혹은 내용) 하단의 꼬리말을 지정
  - 작성자와 작성 날짜, 저작권 등 웹 페이지 관련 추가 정보들을 포함



# HTML5 문서 구조

## ▣ 기존 레이아웃 방식

- 모든 레이아웃 영역을 `<div>` 태그를 사용하므로 세부적인 구별이 어려움
- `<div>` 태그의 `id` 속성값으로 의미를 표시하거나 `class` 속성값으로 의미를 표현

## ▣ 시맨틱 레이아웃 방식

- 레이아웃 영역을 시맨틱 태그를 이용하여 구분
- `<div>` 태그를 여러 시맨틱 태그로 세분화하여 표시
- 아이디(또는 클래스) 이름들을 표준 시맨틱 태그로 정의함으로써 문서의 의미 구조를 명확하고 간결하게 표현하도록 개선

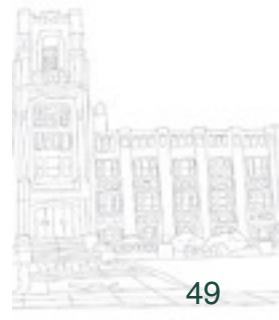
```
<!-- 기존 레이아웃 방식 -->
<body>
<div id="header"> ... </div>
<div id="nav"> ... </div>
<div id="sidebar"> ... </div>
<div id="section1"> ...
    <div id="article"> ... </div>
</div>
<div id="section2"> ...
<div id="section2_1">
    ...
</div>
</div>
<div id="footer"> ... </div>
</body>
```

```
<!-- 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>
```

# <video>, <audio> 태그

## □ 비디오 파일 형식

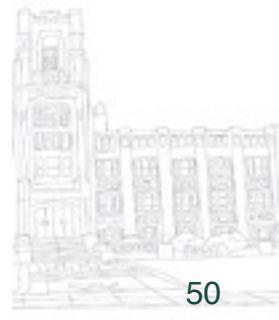
- MPEG4(\*.mp4, \*.m4v)
  - 영상, 음성을 디지털 데이터로 전송, 저장하기 위한 규격의 하나로 동영상 인코딩 방식을 가리킴
  - .mp4와 .m4v는 MPEG-4의 part14로 규정된 파일 형식
  - 최근 H.264 코덱이 part10 AVC로 규격화 됨
  - 아이폰이나 안드로이드폰에서 지원하며 가장 널리 사용중
- 플래시 비디오(\*.flv)
  - 어도비 시스템즈가 개발하고 있는 동영상 파일 형식
- Ogg(\*.ogg)
  - 멀티미디어 컨테이너 형식으로 오픈 파일 형식을 따르고 있으며 멀티미디어 비트 스트림을 효율적으로 전송하고 처리할 수 있게 하기 위해 Xiph.org재단에서 개발
  - 오그 테오라(Ogg Theora) 코덱 사용
- AVI(\*.avi)
  - 마이크로소프트사가 개발한 멀티미디어 형식
  - 화질이 좋은 반면 용량이 커서 실시간 전송에는 적합하지 않음
- WebM(\*.webm)
  - 구글이 2010년 구글 I/O 개발자 컨퍼런스에서 발표한 공개 소스 방식의 파일 형식
  - VP8 비디오와 Vorbis 오디오 스트림으로 구성
  - 구글과 여러 브라우저가 지원하기로 함에 따라 추후 활용도가 높을 것으로 예상.



## <video>, <audio> 태그

### □ 비디오 , 오디오 코덱 지원 여부

브라우저	버전	비디오코덱	오디오코덱
Chrome	10.0	ogg/theora, WebM, H.264*	ogg/vorbis, mp3, wav, AAC
Internet Explorer	9	H.264, WebM**	mp3, AAC
	10	H.264	mp3, AAC
Safari	3	H.264	mp3, wav, AAC
Opera	10.5	ogg/theora	ogg/theora
	11.1	ogg/theora, WebM	ogg/theora, WebM
Firefox	3.6	ogg/theora	ogg/vorbis, wav
	4	ogg/theora, WebM, H.264*	ogg/vorbis, wav



# <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



## <video>, <audio> 태그

- 예제 : play\_video.html



## <video>, <audio> 태그 (chap 2-1)

### [ HTML 태그 소스 ]

```
<header>
  <h1>HTML5 비디오 태그 테스트</h1>
</header>
<section id="player">
  <video id="media" width="600" height="400" preload controls loop poster="poster.jpg">
    <source src="trailer.mp4">
    <source src="trailer.ogg">
  </video>
</section>
```

### [ CSS 구현 코드 ]

```
section, h1 {
  text-align : center;
}
h1 {
  background-image : linear-gradient(to right, red,
    #f06d06, rgb(255, 255, 0), green);
  border-radius : 6px;
  text-shadow : 4px 4px 4px #ffffff;
  width : 80%; height : 80px;
  margin : auto; padding-top : 20px;
  font-size : 2.2em;
}
header {
  margin-top : 10px;
}
```



## 웹 폼 (chapter2-2)

### □ <input>에 추가된 요소들

- <input type="email">

- 이메일 주소 입력시 사용
- 서버로 전송시 이메일 형식 자동 체크

email

- <input type="url">

- 웹 사이트 주소 입력시 사용

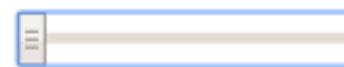
- <input type="number">

- 숫자를 스피너 박스를 이용해서 입력가능
- min : 최소값, max : 최대값, step : 간격, value : 초기값

number  

- <input type="range">

- 슬라이드 막대를 숫자 선택
- min : 최소값, max : 최대값, step : 간격, value : 초기값으로 생략시 중간에 위치.

range 

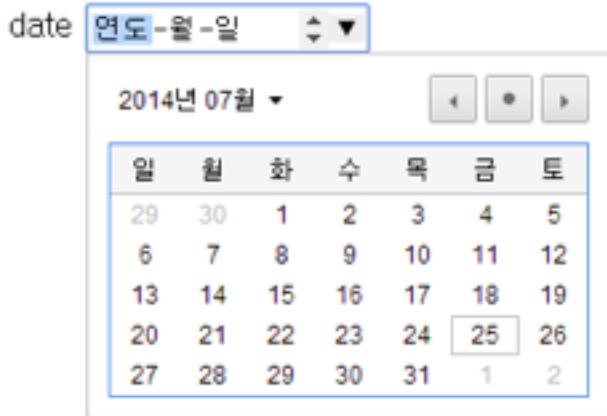
- <input type="search">

- 검색 상자 삽입
- 검색어 입력하면 오른쪽에 x가 표시됨

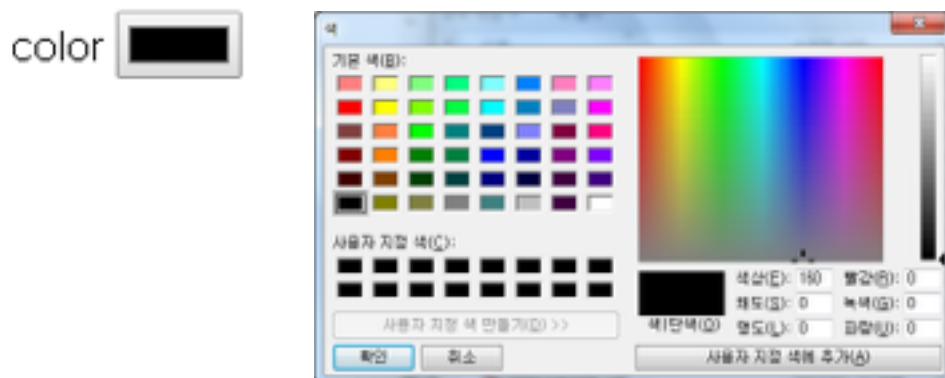
search  



- <input type="date">, <input type="month">, <input type="week">, <input type="time">
  - 달력에서 날짜를 선택하거나 스픈 박스에서 시간을 선택



- <input type="color">
  - 색상 선택 상자 표시



## □ <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

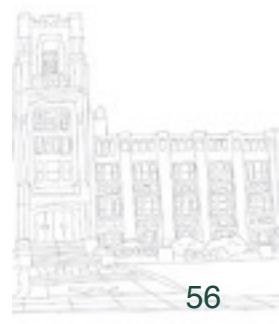
- <input> 태그의 필드 안에 적당한 힌트 내용 표시
  - 실제로 값을 입력하면 힌트표시는 자동으로 사라진다.

- 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 속성이다.



## 웹 품

- 예제 : book\_rental.html

**도서 대출 예약**

성명 :

전화 : 00-000-0000

이메일 : \*\*\*@\*\*\*.\*\*\*

도서명 :

예약권수 :  1권

예약 회망일 : 연도-월-일

수령 시간 : — :— :—에서 — :— 사이

**예약하기**

**도서 대출 예약**

성명 :

전화 : 00-000-00  이 입력란을 작성하세요.

이메일 : \*\*\*@\*\*\*.\*\*\*

도서명 :

예약권수 :  1권

예약 회망일 : 연도-월-일

수령 시간 : — :— :—에서 — :— 사이

**예약하기**

**도서 대출 예약**

성명 :

전화 : 00-000-0000

이메일 : \*\*\*@\*\*\*.\*\*\*

도서명 :

예약권수 :  1권

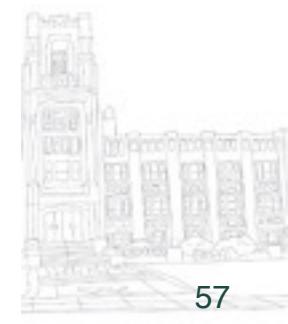
예약 회망일 : 연도-월-일 

수령 시간 :

**예약하기**

2015년 06월   

일	월	화	수	목	금	토
31	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	1	2	3	4



# 웹 폼 (chapter2-3)

## [ HTML 태그 소스 ]

```
<fieldset>
<legend>도서 대출 예약</legend>
<form method="get" action=".....">
    성명 : <input type="text" name="p_name" required autofocus /> <br>
    전화 : <input type="tel" name="p_tel" placeholder="00*-000*-0000" required
            pattern="[0-9]{2,3}-[0-9]{3,4}-[0-9]{4}" /> <br>
    이메일 : <input type="email" name="p_mail" required placeholder="***@**.**"/><br>
    도서명 : <input type="text" size="40" name="book_title" /><br>
    예약권수 : <input type="range" value="1" min="1" max="3" name="amount"
                  onchange="updateRange(this);"/>
                <output id="rangevalue">1권</output><br>
    예약 희망일 : <input type='date' name='p_date'><br>
    수령 시간 : <input type="time" name="time_from" min="09:00" max="18:00" >
    에서 <input type="time" name="time_until" min="09:00" max="18:00" >사이<br>
    <hr>
    <input type="submit" value="예약하기"/>
</form>
</fieldset>
```

## [ JavaScript 소스 ]

```
function updateRange(obj) {
    document.getElementById('rangevalue').innerHTML = obj.value + '권';
}
window.onload = function() {
    var domobj = document.querySelector("input[type=date]");
    var d = new Date();
    var year = d.getFullYear();
    var month = d.getMonth()+1;
    var date = d.getDate();
    if (month <= 9)
        month = "0"+month;
    if (date <= 9)
        date = "0"+date;
    var datestr = year+"-"+month+"-"+date;
    domobj.min = datestr;
}
```

# contenteditable 속성

## □ contenteditable 속성

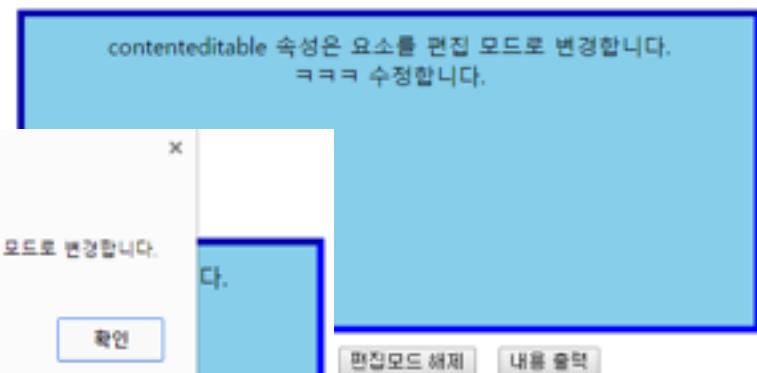
- 요소(Element)를 편집 모드로 변경하는 기능을 제공한다.
- 설정값
  - true : 요소 편집 가능 상태
  - false : 요소 편집 불가능 상태
  - inherit : 부모 요소의 값 상속. 즉, 부모가 편집 가능하다면 이 요소도 편집 가능(디폴트)

## □ 예제(contenteditable.html)

### contenteditable 속성 테스트



### contenteditable 속성 테스트



# 웹 품 (chapter2-4)

## [ HTML 태그 소스 ]

```
<style>
.edit_content {
    width : 500px; height : 200px;
    margin : 5px auto; padding : 10px;
    border : 1px dashed red;
}
.edit_content[contenteditable=true] {
    border : 5px inset blue; background-color : skyblue;
}
button { margin : 5px; }
</style>
<h2>contenteditable 속성 테스트</h2>
<div class="edit_content"> contenteditable 속성은 요소를 편집 모드로 변경합니다. </div>
<button id="edit_btn" onclick="edit_content()">편집모드 설정</button>
<button id="finish_btn" onclick="finish_content()">편집모드 해제</button>
<button id="show_btn" onclick="show_content()">내용 출력</button>
<script>
var editTarget = document.querySelector(".edit_content");
function edit_content() {
    editTarget.setAttribute("contenteditable", true);
}
function finish_content() {
    editTarget.setAttribute("contenteditable", false);
}
function show_content() {
    alert(editTarget.textContent);
}
</script>
```

# Cascade Style Sheet

## □ CSS(Cascading Style Sheets) 란?

- 구조적으로 짜여진 문서(HTML,XML)에 Style(글자,여백,레이아웃)을 적용하기 위해 사용 하는 언어(Language)이다.
- CSS 스타일시트는 HTML 문서의 요소에 적용되는 CSS 스타일 정의를 포함하며 CSS 스타일은 요소 표시 방법 및 페이지에서의 요소 위치를 지정한다.
- W3C의 표준이며 HTML구조는 그대로 두고 CSS 파일만 변경해도 전혀 다른 웹사이트처럼 꾸밀 수 있다.

The image shows a screenshot of the CSS Zen Garden website, which displays four different CSS styles applied to the same basic HTML template. The top left shows a dark blue 'HTML' logo with a subtitle '웹페이지의 데이터 보유'. The top right shows a large 'CSS' logo with a subtitle '웹페이지의 디자인 정보 보유'. Below these are four distinct web pages:

- Top Left:** A dark purple design featuring a large white 'HtmL' logo and several white lotus flowers on a dark background.
- Top Right:** A light green design featuring a red rocket ship launching over a blue sky with white clouds.
- Bottom Left:** A blue ocean-themed design featuring a yellow fish at the bottom and a blue gradient background.
- Bottom Right:** A pink and purple design featuring stylized flower icons and a pink gradient background.

Each page contains placeholder text and navigation links, demonstrating how CSS can transform the visual presentation of the same underlying HTML code.

# Cascade Style Sheet

## □ CSS 의 작성 방법

- 인라인 방법 - HTML 엘리먼트에 style 이라는 속성으로 정의하는 방법
- 전역적 방법 - <style> 이라는 태그에 웹 페이지의 태그들에 대한 스타일을 정의하는 방법
- 외부 파일 연결 방법 - 독립된 파일(확장자 .css)을 만들어서 HTML 문서에 연결하는 방법

### [ CSS 스타일 선언 형식 ]

#### - 선택자(selector)

스타일을 적용할 대상을 지정

#### - 스타일 속성(property) 블록

선택자에 의해 선택된 영역에

적용할 색상, 크기 등의 스타

일을 명세

스타일 속성 블록은 중괄호({ })

로 둘러싸며 하나 이상의 스타일  
속성 선언을 포함



# Cascade Style Sheet

## □ CSS 사용의 이점

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

## □ CSS의 역사

### ■ CSS1

- 첫 CSS 규격은 공식 W3C 권고안이 되었으며 그 이름은 CSS1이다.
- 1996년 12월에 출시되었다.

### ■ CSS2

- W3C가 개발하였으며 1998년 5월에 권고안으로 출시되었다.

### ■ CSS3

- 2005년 12월 5일 이후 개발 중에 있다.
- 모듈 기반으로 개발이 진행중이며 선택적으로 지원할 수 있다.
- CSS3의 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.
- 현재 모든 브라우저가 CSS3를 완벽하게 지원하는 것은 아니다. 일부 기능은 브라우저에 따라 지원 방식이 달라 별도의 접두어를 붙여야만 사용할 수 있다.

# Cascade Style Sheet

## □ CSS3 소개

- CSS2와 CSS3의 가장 큰 차이점은 CSS3가 모듈 기반으로 개발되고 있다는 점이다. 이것은 각종 브라우저나 디바이스가 필요에 따라 원하는 CSS 모듈만을 탑재하거나 또는 필요한 모듈만을 빠르게 자주 업데이트 하는 것을 돋는다.
- CSS3는 text, fonts, color, backgrounds & borders, transforms, transitions, animations와 같은 종류의 모듈들을 추가로 개발하고 있다.
- CSS3는 기존의 CSS2가 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 CSS3만으로도 상당부분 가능해지면서 웹 사이트의 성능 향상에 크게 기여할 수 있게 되었다.

### [ CSS3의 주요 기능 ]

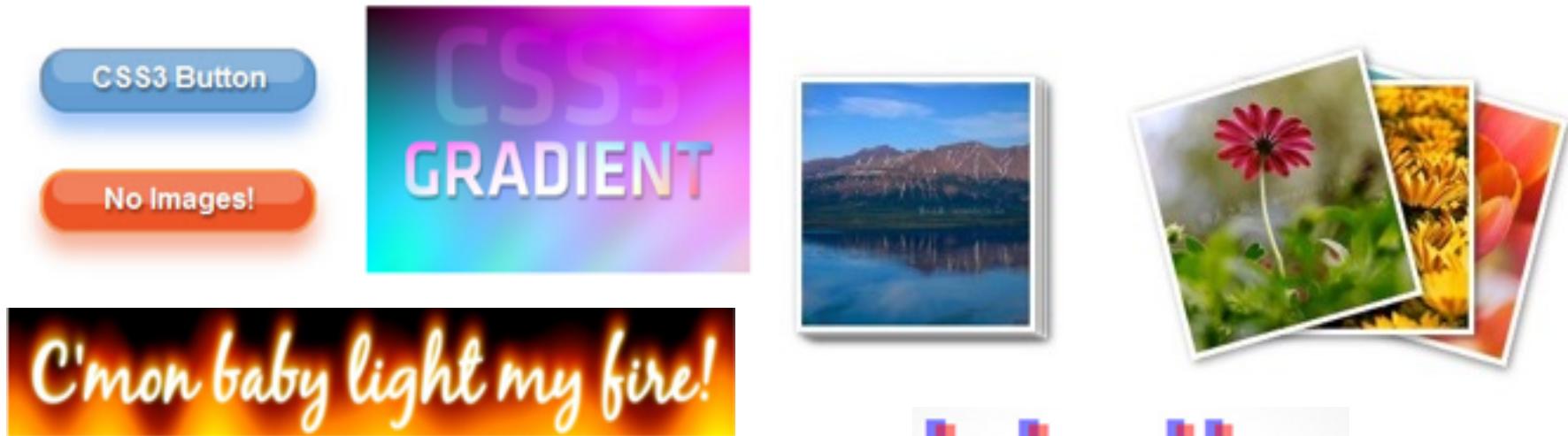
- 구조 선택자 지원
- 가상 선택자 지원
- 둥근 모서리의 경계선 지원
- 그라데이션 배경 지원
- 변환, 전환, 애니메이션 지원
- 글자와 박스 뒤에 그림자 효과 지원



# Cascade Style Sheet

## □ CSS3의 주요 구현 예

- CSS3 기술을 이용하면 다음의 화면들을 이미지를 사용하지 않고 제작하는 것이 가능하다.



Hello

CSS3와 미디어 쿼리 기능을 사용하면 클라이언트의 화면 사이즈에 알맞게 요소들의 레이아웃이 유동적으로 적용되는 반응형 웹 구현도 가능하다.



# Thank You

**HTML**



**CSS**



**JS**



## 제3장 JavaScript와 jQuery

1. JavaScript의 데이터 타입

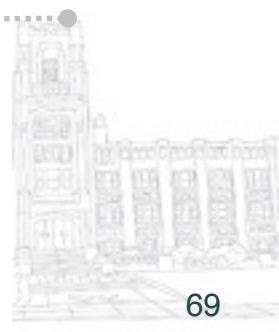
2. JavaScript의 주요 연산자

3. JavaScript의 제어문

4. JavaScript의 배열 정의와 활용

5. JavaScript의 함수 정의와 활용

6. JavaScript의 객체 정의와 활용



7. JavaScript의 표준 내장 객체

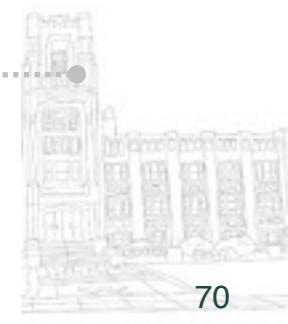
8. JavaScript의 BOM 객체

9. JavaScript의 DOM 객체

10. JavaScript의 이벤트 모델

11. jQuery 라이브러리 사용

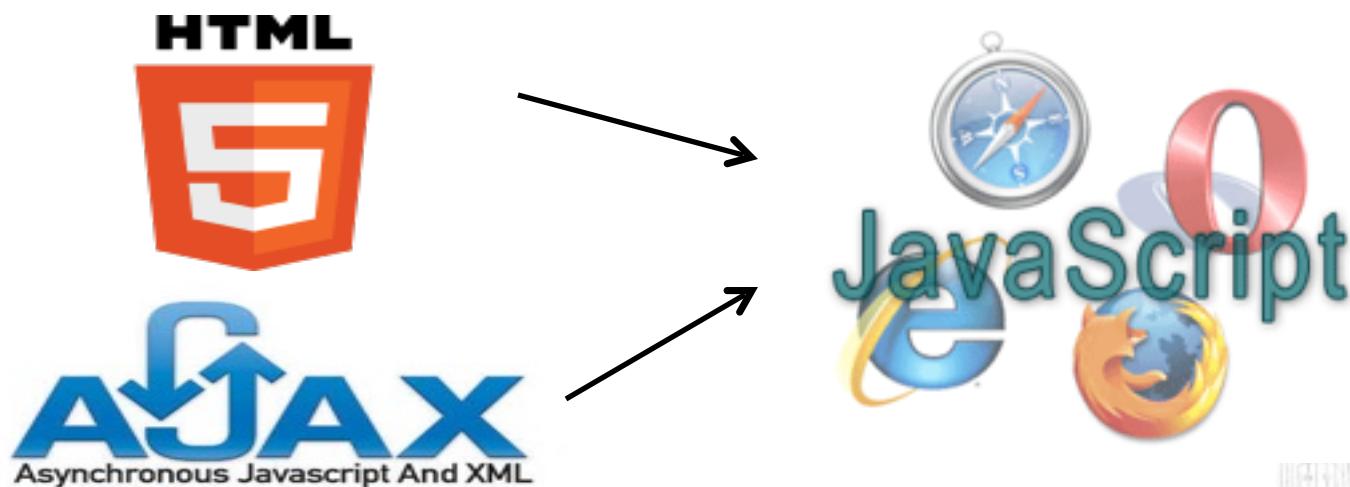
12. jQuery 기반의 DOM 프로그래밍



# JavaScript 소개

## □ JavaScript 란

- JavaScript는 넷스케이프 커뮤니케이션즈 코퍼레이션의 브랜던 아이크(Brendan Eich)가 처음에는 모카(Mocha)라는 이름으로, 나중에는 라이브스크립트(LiveScript)라는 이름으로 개발하였으며, 최종적으로 JavaScript라는 이름으로 발표되었다.
- JavaScript는 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 JavaScript의 가치는 재검토되었다.
- HTML5에서 HTML5의 API로 JavaScript를 공식 채택함으로써 JavaScript는 세계에서 가장 인기 있는 프로그래밍 언어 중 하나로 자리 잡아가고 있다.



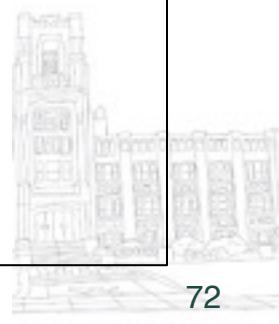
# JavaScript 소개

## [ JavaScript 소스1 ] (chapter3-1)

```
<script>
  document.writeln("JavaScript 테스트");
  window.alert("HTML5 테스트");
  document.writeln("CSS3 테스트");
  console.log('AJAX 테스트');
  document.writeln(this);
</script>
```

## [ JavaScript 소스2 ] (chapter3-2)

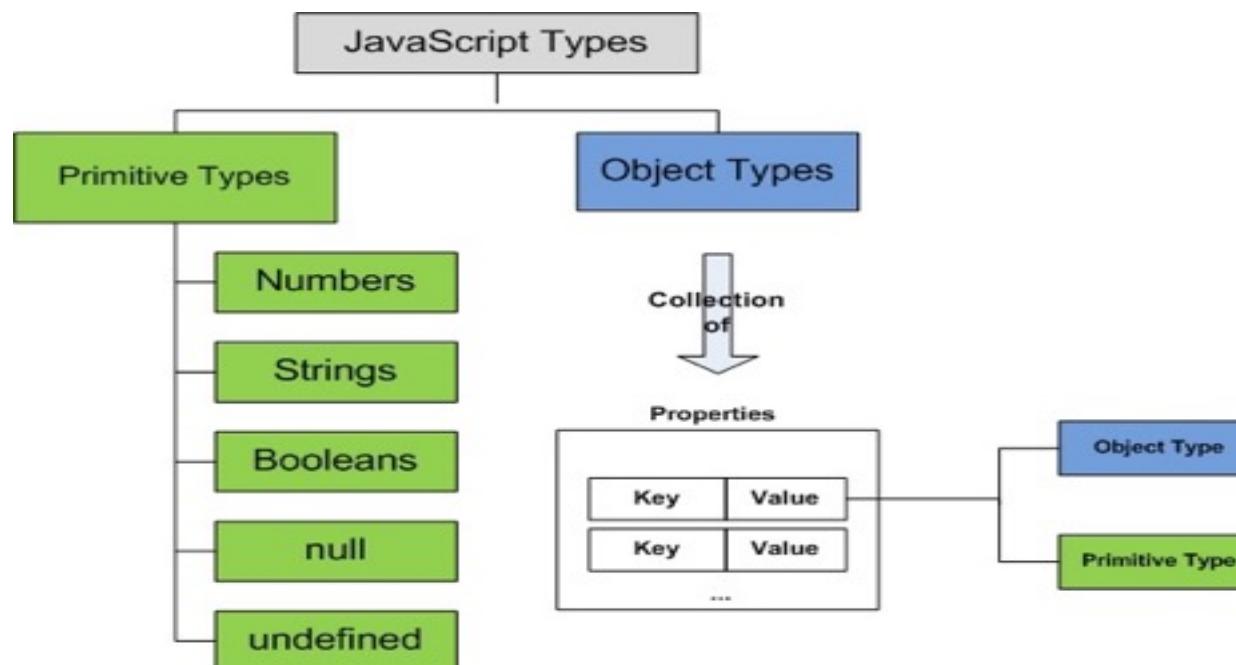
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script>
  alert("ONE");
</script>
</head>
<body>
  <h1>자바스크립트 학습 START</h1>
  <script>
    alert("TWO");
  </script>
  <h1>자바스크립트 학습 END</h1>
</body>
</html>
```



# JavaScript의 데이터 타입

## □ 데이터 타입

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



# JavaScript 소개

## [ JavaScript 소스1 ] (chapter3-3)

```
<script>
var v_test;
window.alert(v_test);
v_test = 100;
window.alert(v_test + 200);
v_test = "100";
window.alert(v_test + 200);
</script>
```

## [ JavaScript 소스2 ] (chapter3-4)

```
<script>
document.writeln("100 : " + typeof 100);
document.writeln("3.14 : " + typeof 3.14);
document.writeln("자바 : " + typeof "자바");
document.writeln("정의되지 않은 변수 : " + typeof sum);
document.writeln("true : " + typeof true);
document.writeln("function() {} : " + typeof function() { });
document.writeln("[1,2,3] : " + typeof [ 1, 2, 3 ]);
document.writeln("new Object() : " + typeof new Object());
var result;
document.writeln("초기화되지 않은 변수 : " + typeof result);
</script>
```

# JavaScript의 주요 연산자

## □ JavaScript의 연산자

- 수치 연산자

덧셈(+), 뺄셈(-), 곱셈(\*), 나눗셈(/), 나머지(%), 증가 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`

- 삭제 연산자

`delete`



# JavaScript의 주요 연산자

## [ JavaScript 소스 ] (chapter3-5)

```
<script>
document.writeln(100 + 200);
document.writeln("100"+200);
document.writeln('100'+'200');
document.writeln("결과 : " + (100+200));
document.writeln("결과 : " + 100*200);
document.writeln((true > false) + "<br>");
document.writeln((true == 1) + "<br>");
document.writeln((true === 1) + "<br>");
var result;
document.writeln((result || "ㅋㅋㅋ") +"<br>");
</script>
```

# JavaScript의 제어문

## □ JavaScript의 제어문

- 조건제어문 if, 다중 분기문 switch  
switch 문에 사용되는 비교식에 데이터 타입의 제한이 없다.
- 반복제어문 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.
- 분기제어문 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 제어문

## [ JavaScript 소스 ] (chapter3-6)

```
<script>
if(100 > 50)
    document.writeln("100 은 50보다 크다");
var number = 10;
if(number % 2 == 0)
    document.writeln(number + "는 짝수");
else
    document.writeln(number + "는 홀수");

var name;
while (true) {
    name = window.prompt("좋아하는 프로그래밍 언어의 이름을 입력하세요", "javascript");
    if (name == 'javascript') {
        window.alert('자바스크립트를 좋아하는군요^^');
    } else if (name == 'java') {
        window.alert('자바를 좋아하는군요^^');
        break;
    }
    else if (name == 'c++')
        window.alert('C++을 좋아하는군요^^');
    else
        window.alert('????');
}
</script>
```

# 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.

### ■ JavaScript의 배열 생성 방법은 2가지 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)  
join(del)  
del로 연결

slice(start [,end])  
요소들을

pop()  
push(data)  
shift()  
삭제

unshift(data,...)  
reverse()  
sort([fnc])  
toString()

지정 배열을 현재의 배열에 추가  
배열 내의 요소들을 구분 문자

해서 문자열 리턴  
start 부터 end-1번째까지의

추출하여 배열 객체를 리턴  
배열 끝의 요소를 취득하여 삭제  
배열 끝에 요소를 추가  
배열 선두의 요소를 취득하여

배열 선두에 지정 요소를 추가  
역순으로 정렬(반전)  
요소를 오름차순으로 정렬  
요소, 요소, ... 의 형식으로 문자열  
리턴

# JavaScript의 배열 정의와 활용

## [ JavaScript 소스1 ] (chapter3-7)

```
<pre>
<script>
var ary1 = [ ];
var ary2 = new Array( );
var ary3 = new Array(10); // 크기 new Array('A')
var ary4 = new Array(10, 20, 30);
var ary5 = [10];
var ary6 = [10, 20, 30];
document.writeln("ary1.length - " + ary1.length);
document.writeln("ary2.length - " + ary2.length);
document.writeln("ary3.length - " + ary3.length);
document.writeln("ary4.length - " + ary4.length);
document.writeln("ary5.length - " + ary5.length);
document.writeln("ary6.length - " + ary6.length);
</script>
</pre>
```

## [ JavaScript 소스2 ] (chapter3-8)

```
<script>
var ary = [100, 200, 'javascript'];
for(var i=0; i < ary.length; i++)
  document.write(ary[i] + " ");
document.write("<br>");
for(var index in ary)
  document.write(ary[index] + " ");
</script>
```

# JavaScript의 배열 정의와 활용

## [ JavaScript 소스 ] (chapter3-9)

```
<pre>
<script>
var ary = [5, 25, 10];
//var ary = ["abc", "xyz", "b"];
document.writeln(ary.sort( ));
document.writeln(ary.sort(
  function(x, y) {
    return x - y;
  }
));
document.writeln(ary.sort(
  function(x, y) {
    return y - x;
  }
));
</script>
</pre>
```

# JavaScript의 함수 정의와 활용

## □ JavaScript의 함수 정의 방법

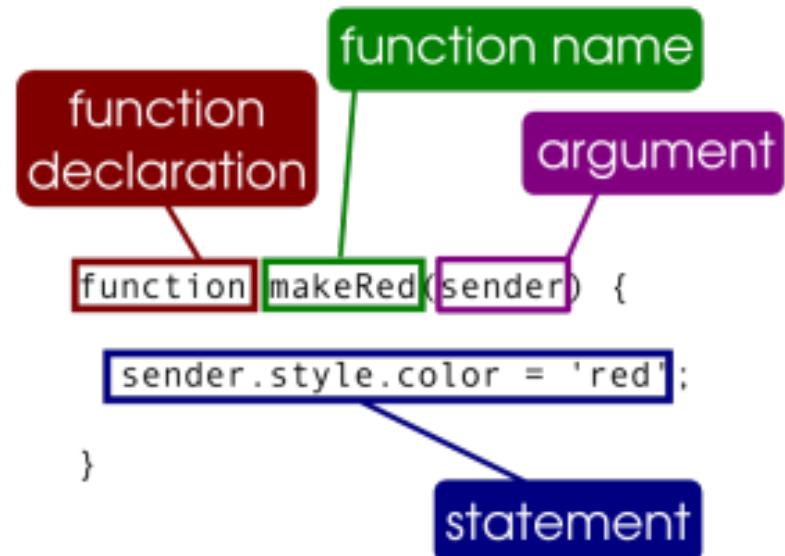
- 함수(function)란 하나의 로직을 재실행 할 수 있도록 하는 것으로 코드의 재사용성을 높여준다.

- 선언적 함수 정의 방법

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

- 표현적 함수 정의 방법

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



# JavaScript의 함수 정의와 활용

## □ JavaScript의 함수의 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 );
```

# JavaScript의 함수 정의와 활용

## [ JavaScript 소스1 ]

```
<pre>
<script>
myFunc1();
//myFunc2();
function myFunc1( ) {
    document.writeln("명시적 함수 정의");
}
var myFunc2 = function ( ) {
    document.writeln("익명 함수(함수 리터럴) 정의");
}
myFunc1();
myFunc2();
</script>
</pre>
```

# JavaScript의 함수 정의와 활용

## [ JavaScript 소스 ]

```
<pre>
<script>
var triangle1 = new Function('base', 'height', 'return base * height / 2;');
document.writeln('삼각형의 면적(Function생성자) : ' + triangle1(5, 2));
var param = 'height, width';
var formula = 'return height * width / 2;';
var diamond = new Function(param, formula);
document.writeln('마름모의 면적 : ' + diamond(5, 2));

var triangle2 = function(base, height) {
    return base * height / 2;
};
document.writeln('삼각형의 면적(함수 리터럴 정의) : ' + triangle2(5, 2));

function triangle3(base, height) {
    return base * height / 2;
}
document.writeln('삼각형의 면적(명시적 정의) : ' + triangle3(5,2));
</script>
</pre>
```

# JavaScript의 함수 정의와 활용

## [ JavaScript 소스 ]

```
<pre>
<script>
var g_v = "전역변수";
function myFunc1() {
    var l_v = "myFunc1()의 지역 변수";
    document.writeln("myFunc1() 함수의 스코프 영역");
    document.writeln("l_v : " + l_v);
    document.writeln("g_v : " + g_v);
}
var myFunc2 = function () {
    var l_v = "myFunc2()의 지역 변수";
    document.writeln("myFunc2() 함수의 스코프 영역");
    document.writeln("l_v : " + l_v);
    document.writeln("g_v : " + g_v);
}
myFunc1();
document.writeln();
myFunc2();

document.writeln();
document.writeln("전역(글로벌) 스코프 영역");
document.writeln("g_v : " + g_v);
document.writeln("l_v : " + l_v);
</script>
</pre>
```

# JavaScript의 함수 정의와 활용

## [ JavaScript 소스1 ]

```
<script>
function test() {
    document.writeln("<h1>" + arguments.length + "</h1>")
    var i;
    document.writeln("<ul>");
    for(i=0;i < arguments.length; i++)
        document.writeln("<li>" + arguments[i]);
    document.writeln("</ul>");
}
test();
test(100);
test(10, 20, 30);
test(10, 20, 30, 40, 50, 60, 70, 80);
</script>
```

## [ JavaScript 소스2 ]

```
<script>
try {
    showMessage('javascript', 'css3', 'html5');
} catch (e) {
    alert(e);
} finally {
    console.log('수행 완료');
}
</script>
```

# JavaScript의 함수 정의와 활용

## [ JavaScript 소스 ]

```
<h1>파라미터 타입 테스트</h1>
<hr>
<button onclick="testParaType(100);">숫자파라미터</button>
<button onclick="testParaType('test');">문자열파라미터</button>
<button onclick="testParaType({ })">객체파라미터</button>
<button onclick="testParaType(function(){ })">함수파라미터</button>
<button onclick="testParaType()";>전달안함</button>
<button onclick="testParaType(true);>논리파라미터</button>
<script>
function testParaType(p) {
    if(typeof p == 'number')
        alert('숫자를 전달했네요');
    else if(typeof p == 'string')
        alert('문자열을 전달했네요');
    else if(typeof p == 'object')
        alert('객체를 전달했네요');
    else if(typeof p == 'function')
        alert('함수를 전달했네요');
    else if(typeof p == 'undefined')
        alert('전달된 파라미터가 없네요');
    else
        alert('????');
}
</script>
```

# JavaScript의 함수 정의와 활용

## [ JavaScript 소스1 ]

```
<script>
function highOrderFunction(p) {
  if (typeof p == 'function') {
    p('ㅋㅋㅋ');
  } else {
    alert(p);
  }
}
highOrderFunction(function (msg) {document.write(msg);});
highOrderFunction(function (msg) {console.log(msg);});
highOrderFunction();
highOrderFunction('ㅋㅋㅋ');
</script>
```

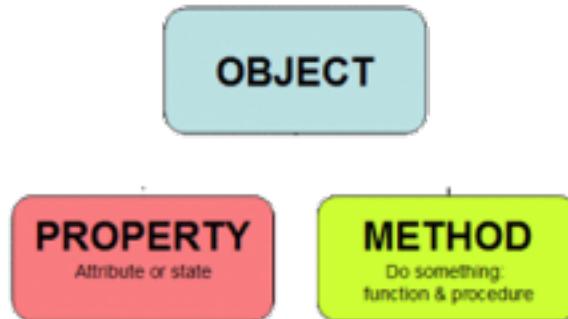
## [ JavaScript 소스2 ]

```
<script>
function outerFunction() {
  return function () {
    document.write('<H1>Hello World .. !</h1>');
  };
}
outerFunction( )();
var result = outerFunction( );
result( );
result( );
</script>
```

# 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.



- JavaScript에서 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 JavaScript에서 생성되는 모든 객체들은 조상 객체로 Object 객체를 갖는다.
- JavaScript의 함수는 실행 가능한 코드와 연결된 객체라 할 수 있다.



# JavaScript의 객체 정의와 활용

## □ JavaScript 객체의 정의 방법

JavaScript의 객체 생성 방법은 다음과 같이 2가지 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; ...  
}
```



# JavaScript의 객체 정의와 활용

## [ JavaScript 소스1 ]

```
<script>
var person = {
  name : '듀크',
  eat : function(food) {
    alert(this.name + "가 " + food + "를 먹어요!!");
  }
};
person.eat("사과");
</script>
```

## [ JavaScript 소스2 ]

```
<script>
var obj = {
  name: '자바스크립트',
  age: 19,
  kind: '웹앱 개발 스크립트 언어'
};
var jsonStr = JSON.stringify(obj);
alert(jsonStr);
var copy = JSON.parse(jsonStr);
alert(copy.name + '\n' + copy.age + '\n' + copy.kind);
</script>
```

# JavaScript의 객체 정의와 활용

## □ JavaScript의 객체의 다양한 활용

```
var person1 = new Object();
person1.firstName = "Ankita";
alert( person1.firstName );
```

```
var people = {};
people[ "person1" ] = person1;
alert( people[ "person1" ].firstName );
```

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

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

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

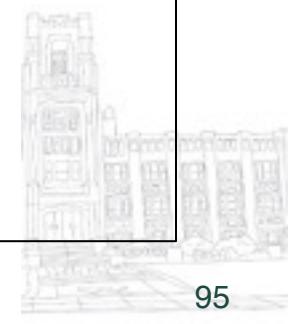
myCar = new car ("Ford", "Focus", "Red");
myCar.displayCar();
myCar.make = "BMW";
myCar.displayCar();
```



# JavaScript의 객체 정의와 활용

## [ JavaScript 소스 ]

```
<script>
function Student(name, korean, math, english, science) {
    // 속성
    this.name = name;
    this.kor = korean;
    this.math = math;
    this.eng = english;
    this.sci = science;
    // 메서드
    this.getSum = function () {
        return this.kor + this.math + this.eng + this.sci;
    };
    this.getAverage = function () { return this.getSum() / 4; };
    this.toString = function () {
        return this.name + ' ' + this.getSum() + ' ' + this.getAverage();
    };
}
var students = [ ];
students.push(new Student('홍길동1', 87, 98, 88, 95)); students.push(new Student('홍길동2', 92, 98, 96, 98));
students.push(new Student('홍길동3', 76, 96, 94, 90)); students.push(new Student('홍길동4', 98, 92, 96, 92));
students.push(new Student('홍길동5', 95, 98, 98, 98)); students.push(new Student('홍길동6', 64, 88, 92, 92));
students.push(new Student('홍길동7', 82, 86, 98, 88)); students.push(new Student('홍길동8', 88, 74, 78, 92));
var output = '이름    총점    평균\n';
for (var i in students) {
    output += students[i].toString() + '\n';
}
alert(output);
</script>
```



# JavaScript의 내장 객체

## □ JavaScript 내장 객체들의 종류

- 표준 내장 객체

JavaScript 언어 자체에 정의되어 있는 객체들이다.

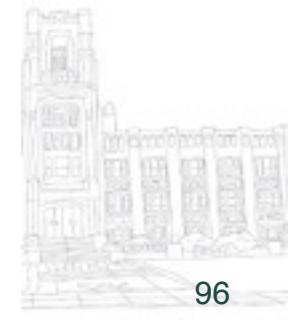
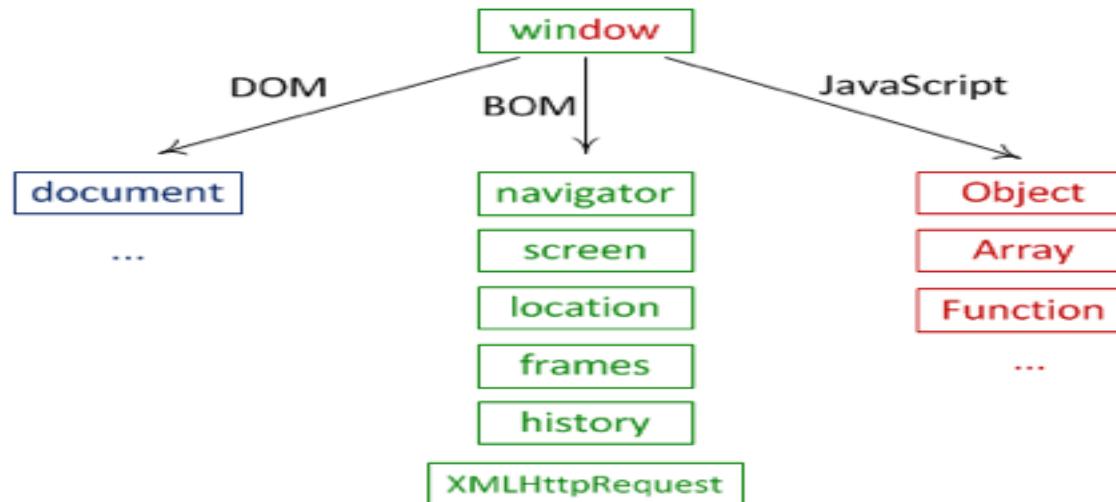
- BOM

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

- DOM

Document Object Model. Web 페이지의 내용을 제어한다. window의 프로퍼티인 document 프로퍼티에 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 JavaScript가 기본적으로 가지고 있는 객체들을 의미 한다.
- 프로그래밍이라는 것은 언어와 호스트 환경에서 제공하는 기능들을 통해서 새로운 소프트 웨어를 만들어내는 것이므로 내장 객체에 대한 이해는 프로그래밍의 기본이라고 할 수 있다.
- JavaScript는 아래와 같은 내장 객체를 가지고 있다.

Object : 최상의 객체로서 JavaScript의 모든 객체들은 이 객체를 상속하게 된다.

Function : 함수정의시 사용되는 객체이다.

Array : 배열 정의시 생성되는 객체이다.

String : 문자열 데이터에 대한 Wrapper 객체이다.

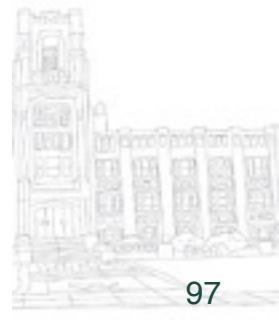
Boolean : 대수형 값에 대한 Wrapper 객체이다.

Number : 수치값에 대한 Wrapper 객체이다.

Math : 다양한 수학 함수 기능을 제공하는 객체이다.

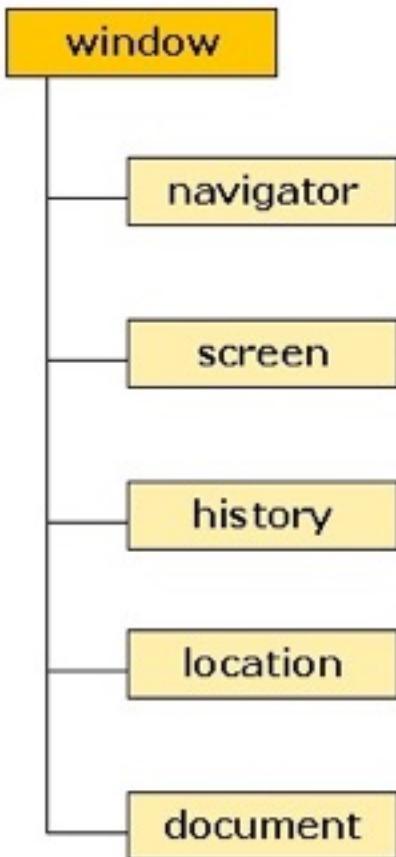
Date : 날짜와 시간 정보 추출과 설정 관련 기능을 제공하는 객체이다.

RegExp : 정규 표현식(패턴)을 이용하여 데이터를 처리하려는 경우 사용되는 객체이다.



# JavaScript의 BOM 객체

## □ BOM 객체



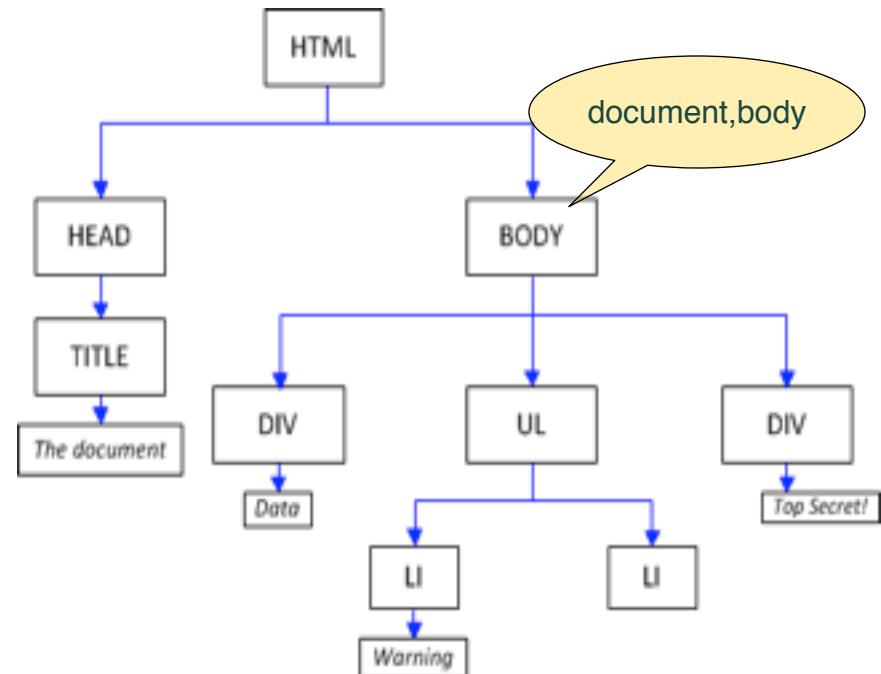
- window : 최상위 객체로, 각 탭별, iframe 별로 하나씩 존재
- 
- navigator : 브라우저(이름, 버전등)정보를 보관하는 객체
- document : 현재의 문서에 대한 정보를 보관하는 객체
- location : 현재 보여지고 있는 웹 페이지에 대한 URL 정보를 보관하는 객체
- history : 현재의 브라우저가 접근했던 URL의 정보를 보관하는 객체



# JavaScript의 DOM 객체

## DOM 이란?

- 문서 객체 모델(DOM; Document Object Model)은 객체 지향 모델로써 구조화된 문서를 표현하는 형식이다.
- DOM은 플랫폼/언어 중립적으로 구조화된 문서를 표현하는 W3C의 공식 표준이다. 또한 W3C가 표준화한 여러 개의 API의 기반이 된다.
- 브라우저는 서버로 부터 응답된 웹 컨텐트 내용을 파싱한 후 트리구조로 각 HTML 태그마다 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 ('id속성값')`

태그에 정의된 id 속성의 값으로 DOM 객체 찾음

- `document.getElementsByClassName('class속성값')`

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')`

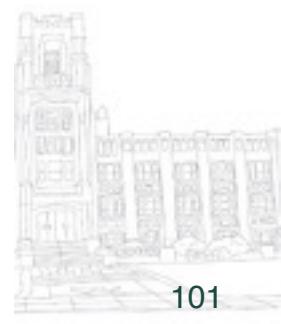
선택자에 알맞은 DOM 객체들을 찾음

- 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

- DOM의 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 2가지 가능하다.
  - 간단한 컨텐트의 편집에는 `node.innerHTML` 또는 `node.innerText` 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
    - `removeChild()` : 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
  - JavaScript는 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.
  - JavaScript가 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**

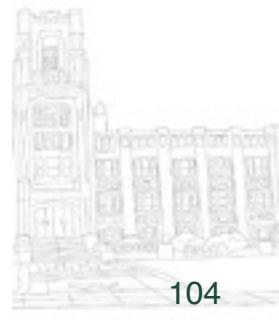
```
<script>
function btn_onclick(){
    window.alert('안녕?');
}
</script>
<input type="button" value="다이얼로그 표시" onclick="btn_onclick()">
```

- **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">
```



# JavaScript의 이벤트 연결

## [ JavaScript 소스 ]

```
<div id="main">
  <p>클릭하세요. 전역적 이벤트 모델로 핸들러가 수행됩니다.</p>
  <p onclick="alert('인라인 이벤트 모델로 처리합니다.!')">클릭하세요. 인라인 이벤트 모델로 핸들러가 수행됩니다.</p>
  <p>클릭하세요. 어떠한 기능도 수행되지 않아요.</p>
</div>
<script>
  function showalert(){
    alert('전역적 이벤트 모델로 처리합니다.!');
  }
  function clickme(){
    document.getElementsByTagName('p')[0].onclick=showalert;
  }
  window.onload=clickme;
</script>
```

# JavaScript의 이벤트 연결

## [ JavaScript 소스 ]

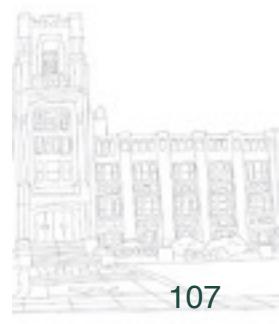
```
<div id="main">
  <p>JavaScript</p>
  <p>HTML5</p>
</div>
<p>CSS3</p>
<script>
  function clickme() {
    var list = document.querySelectorAll("div>p");
    for (var i = 0; i < list.length; i++) {
      list[i].onclick = showalert;
      list[i].onmouseover = function() {
        this.style.color = 'red';
      };
      list[i].onmouseout = function() {
        this.style.color = 'black';
      };
    }
  }
  function showalert(e) {
    alert(e.type + " 이벤트 발생!! - " + e.target.innerHTML);
  }
  window.onload = clickme;
</script>
```

# JavaScript의 이벤트 연결

## [ JavaScript 소스 ]

```
<input type="button" id='btn' value="다이얼로그표시">
<script>
function addListener(elem, ev, listener) {
    if(elem.addEventListener) {
        elem.addEventListener(ev, listener, false);
    } else if(elem.attachEvent) { // IE 8 이하버전
        elem.attachEvent('on' + ev, listener);
    } else {
        throw new Error('이벤트 리스너에 미대응입니다.');
    }
}
addListener(window, 'load', init);

function init() {
    addListener(document.getElementById('btn'), 'click', function() {
        window.alert('버튼이 클릭되었습니다.');
    });
}
</script>
```



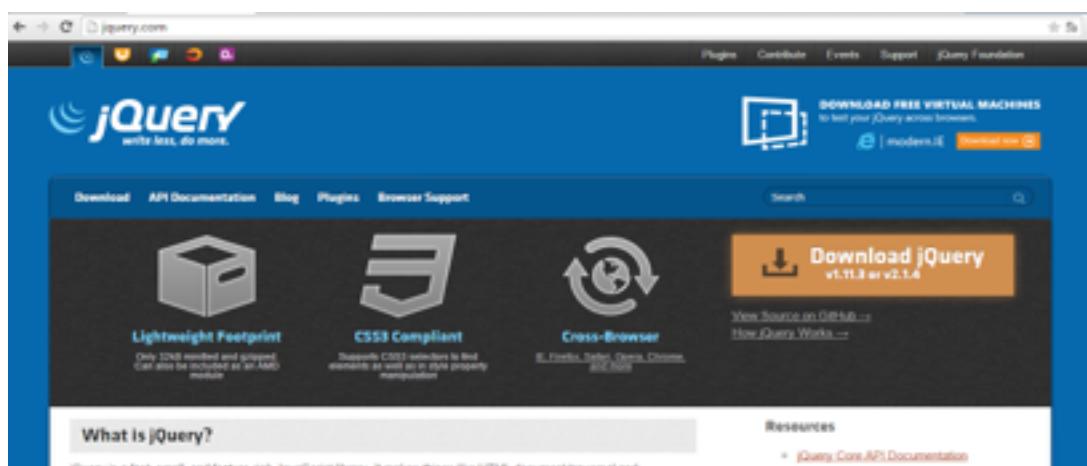
# jQuery 라이브러리 사용

## ▣ jQuery 란?

- HTML 문서 안의 스크립트 코드를 단순화하도록 설계된 자바스크립트 라이브러리이다.
- 존 레식(John Resig)이 2006년 자바스크립트를 쉽게 이용할 목적으로 제안하였다.
- 웹 브라우저마다 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. (Sizzle을 이용)
- 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()

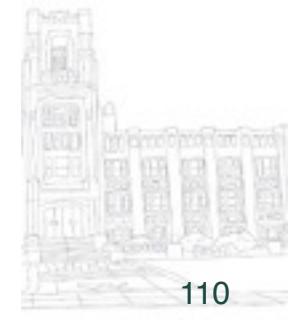
jQuery( html [, ownerDocument ] )

jQuery( html, attributes )

jQuery( callback )

jQuery → \$

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



# jQuery 라이브러리 사용

## □ \$(() 함수의 사용

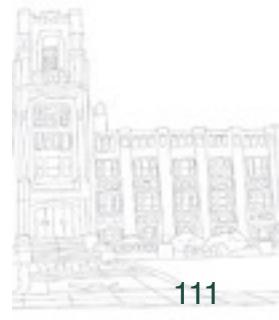
- 시작 이벤트 핸들러 등록

```
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~价值]  
[attribute]  
[attribute2]

#### Forms

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

#### Form Filters

:enabled  
:disabled  
:checked  
:selected



# jQuery 라이브러리 사용

## □ 스타일 처리 관련 메서드

형식	기능
<code>\$( ).css(CSS속성명)</code>	선택된 엘리먼트에 적용된 CSS 스타일 속성값을 반환
<code>\$( ).css(CSS속성명, CSS속성값)</code> <code>\$( ).css({CSS속성집합})</code>	선택된 엘리먼트에 CSS 스타일을 적용 속성값을 한꺼번에 설정(맵형식( {'속성명':'속성값', '속성명':'속성값', ... } ))
<code>\$( ).addClass(CSS클래스명)</code>	선택된 엘리먼트에 class 속성값을 설정(CSS 클래스명으로 선언된 스타일을 적용)
<code>\$( ).removeClass(CSS클래스명)</code>	선택된 엘리먼트의 class 속성값을 제거(적용된 CSS 클래스 스타일을 제거)
<code>\$( ).toggleClass(CSS클래스명)</code>	선택된 엘리먼트에 class 속성값이 존재하면 제거, 없으면 추가(CSS 클래스 스타일을 적용/해제를 전환)
<code>\$( ).hasClass(CSS클래스명)</code>	선택된 엘리먼트에 class 속성값 존재 유무를 반환(CSS 클래스 스타일 적용 유무를 반환)
<code>\$( ).width()</code>	선택된 엘리먼트의 너비 값을 반환
<code>\$( ).width(너비값)</code>	선택된 엘리먼트의 너비 값을 설정
<code>\$( ).height()</code>	선택된 엘리먼트의 높이 값을 반환
<code>\$( ).height(높이값)</code>	선택된 엘리먼트의 높이 값을 설정

# jQuery 라이브러리 사용

## □ DOM 탐색 메서드

- 선택자 이외에도 여러 DOM 탐색 메서드를 통해 엘리먼트에 접근 가능
- DOM 트리 관련 탐색 메서드와 필터링 메서드는 선택자의 기능을 확장 및 보완
- 탐색(traversing) 메서드 : DOM 트리의 선택된 위치를 기준으로 원하는 노드를 찾음
- 선택자 대신 현재의 참조 엘리먼트를 기준으로 탐색 메서드를 사용하는 것이 효과적인 경우에 사용  
(예: 이전 또는 다음 엘리먼트를 접근하는 경우)
- `$()` 함수 선택자를 이용 특정 노드를 찾은 후, 그 위치를 기준으로 다른 노드를 탐색

형식	기능
<code>\$( ).find( )</code>	선택된 엘리먼트 중에서 조건을 충족하는 모든 자손 엘리먼트를 반환
<code>\$( ).children( )</code>	선택된 엘리먼트 중에서 [조건을 충족하는] 모든 자식 엘리먼트들을 반환
<code>\$( ).parent( )</code>	선택된 엘리먼트 중에서 부모 엘리먼트를 반환
<code>\$( ).parents( )</code>	선택된 엘리먼트 중에서 [조건을 충족하는] 모든 조상 엘리먼트들을 반환
<code>\$( ).siblings( )</code>	선택된 엘리먼트 중에서 자신을 제외한 [조건을 충족하는] 모든 형제 엘리먼트들을 반환
<code>\$( ).prev( )</code>	선택된 엘리먼트 중에서 바로 앞에 위치한 [조건을 충족하는] 형제 엘리먼트를 반환
<code>\$( ).prevAll( )</code>	선택된 엘리먼트 중에서 앞에 위치한 모든 [조건을 충족하는] 모든 형제 엘리먼트들을 반환
<code>\$( ).next( )</code>	선택된 엘리먼트 중에서 바로 다음에 위치한 [조건을 충족하는] 형제 엘리먼트를 반환
<code>\$( ).nextAll( )</code>	선택된 엘리먼트 중에서 다음에 위치한 [조건을 충족하는] 모든 형제 엘리먼트들을 반환

# jQuery 라이브러리 사용

## □ DOM 필터링 메서드

- DOM 트리의 선택된 노드 집합(jQuery 객체 집합)에서 다시 특정 조건을 만족하는 노드만을 탐색하는 필터링 메서드

형식	기능(첨자가 0부터 시작됨)
<code>\$( ).filter( )</code>	선택된 엘리먼트들 중에서 필터링 조건을 충족하는 엘리먼트를 반환
<code>\$( ).slice(시작첨자[, 종료첨자])</code>	선택된 엘리먼트들 중에서 시작첨자부터 종료첨자 이전까지의 엘리먼트를 반환
<code>\$( ).first( )</code>	선택된 엘리먼트들 중에서 첫 번째 엘리먼트를 반환
<code>\$( ).last( )</code>	선택된 엘리먼트들 중에서 마지막 엘리먼트를 반환
<code>\$( ).eq(n)</code>	선택된 엘리먼트들 중에서 첨자 n인 엘리먼트를 반환
<code>\$( ).has( )</code>	선택된 엘리먼트들 중에서 특정 자손 엘리먼트를 갖는 엘리먼트를 반환
<code>\$( ).is( )</code>	선택된 엘리먼트들 중에서 특정 조건을 만족하는 엘리먼트가 있으면 'true' 반환
<code>\$( ).not( )</code>	선택된 엘리먼트들 중에서 특정 조건을 만족하지 않는 엘리먼트를 반환

# jQuery 라이브러리 사용

## □ DOM 트리 엘리먼트 조작 메서드

- 정적인 HTML5 문서에 동적인 특성을 제공하는 jQuery의 대표적 기능
- DOM 트리에 새로운 노드를 추가하거나 변경 또는 제거를 자유롭게 수행

형식	기능
<code>\$( ).append( )</code>	선택된 엘리먼트의 마지막 자식 엘리먼트로 추가
<code>\$( ).prepend( )</code>	선택된 엘리먼트의 첫 번째 자식 엘리먼트로 추가
<code>\$( ).after( )</code>	선택된 엘리먼트의 뒤에 형제 엘리먼트로 추가
<code>\$( ).before( )</code>	선택된 엘리먼트의 앞에 형제 엘리먼트로 추가
<code>\$( ).empty( )</code>	선택된 엘리먼트의 내용을 비움(자식 엘리먼트만 제거)
<code>\$( ).remove( )</code>	선택된 엘리먼트를 제거(자식 엘리먼트도 포함하여 제거)
<code>\$( ).replaceWith( )</code>	선택된 엘리먼트를 특정 엘리먼트로 바꿈
<code>\$( ).clone( )</code>	선택된 엘리먼트를 복사
<code>\$( ).wrap( )</code>	선택된 엘리먼트를 특정 엘리먼트로 둘러쌈(부모 엘리먼트로 삽입)
<code>\$( ).unwrap( )</code>	선택된 엘리먼트의 부모 엘리먼트를 제거

# jQuery 라이브러리 사용

## □ 프로그래밍 관련 메서드

- 사용자와의 상호 작용을 위한 DOM 트리와의 정보 교환에 관련된 jQuery 메서드

형식	기능
<code>\$( ).html( )</code>	선택된 엘리먼트의 내용을 HTML5 형식의 문자열로 반환(마크업 포함)
<code>\$( ).html(HTML5문자열)</code>	선택된 엘리먼트 밑에 HTML5 문자열을 엘리먼트로 변환하여 추가
<code>\$( ).text( )</code>	선택된 엘리먼트의 텍스트 내용을 텍스트 형식의 문자열로 반환
<code>\$( ).text(문자열)</code>	선택된 엘리먼트 밑에 텍스트 내용으로 문자열을 추가
<code>\$( ).size( )</code>	선택된 엘리먼트의 개수를 반환 <code>\$( ).length</code> 와 기능 동일
<code>\$( ).get(첨자)</code>	선택된 엘리먼트 중에서 첨자(0부터 시작)에 해당하는 엘리먼트 객체를 반환
<code>\$( ).index( )</code>	선택된 (첫)엘리먼트의 형제 엘리먼트와의 상대적인 첨자를 반환
<code>\$( ).each(콜백함수)</code>	선택된 엘리먼트들을 차례로 순환하면서 콜백 함수를 반복 호출

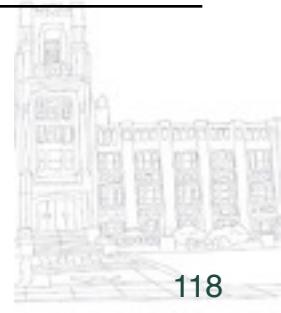


# jQuery 라이브러리 사용

## □ DOM 트리 속성 조작 메서드

- 엘리먼트의 시작 태그 안에 포함된 속성 이름과 속성값은 DOM 트리에서 엘리먼트 노드의 하위 노드
- 속성에 대해서도 메서드를 통해 DOM 트리에 추가하거나 제거 가능

형식	기능
<code>\$( ).attr(속성명)</code>	선택된 엘리먼트들 중 첫 번째 엘리먼트의 특정 속성값을 반환
<code>\$( ).attr(속성명, 속성값)</code> <code>\$( ).attr({속성집합})</code> <code>\$( ).attr(속성명, 함수())</code>	선택된 모든 엘리먼트에 속성값을 설정 여러 속성값을 한꺼번에 설정(맵형식( {속성명:속성값, 속성명:속성값, ...} )) 함수 반환 값을 속성값으로 설정
<code>\$( ).removeAttr(속성명)</code>	선택된 모든 엘리먼트의 특정 속성을 제거
<code>\$( ).val()</code>	선택된 첫 번째 엘리먼트(폼 관련)의 value 속성값을 반환
<code>\$( ).val(속성값)</code>	선택된 모든 엘리먼트(폼 관련)의 value 속성값을 설정



# jQuery 라이브러리 사용

## □ each( ) 메서드

- 보통 이름없는 콜백 함수(반복 함수)를 입력 인자로 사용
- 콜백 함수는 \$( )가 반환하는 jQuery 객체 집합의 개수만큼 반복해서 호출됨
- 콜백 함수 안에서 반복 호출할 때마다 각 jQuery 객체(엘리먼트)들을 \$(this)로 접근

```
<ul>      <li>봄</li><li>여름</li><li>가을</li><li>겨울</li>
</ul>
```

```
$('li').each(function(){
    alert($(this).text());
});
```

# jQuery 라이브러리 사용

## □ 이벤트 핸들러 등록과 해제

- 이벤트 타입명에 해당하는 단축형 메서드를 사용하여 이벤트핸들러 연결
- on() 메서드를 사용하여 이벤트 타입과 이벤트 핸들러 연결

```
function clickEventFunc() {
    alert('hi');
}

$('div').click(clickEventFunc);
$('div').on('click', clickEventFunc);
```

- 연결된 이벤트 핸들러를 해제하는 경우에는 off() 메서드를 이용

```
$(선택자).off('이벤트유형') ;           // 선택자 객체에 연결된 이벤트를 해제
$(선택자).off() ;                      // 선택자 객체에 연결된 모든 이벤트를 해제
```

# jQuery 라이브러리 사용

## □ 이벤트 연결/해제 메서드

- 이벤트 발생시 실행할 함수를 이벤트와 연결하거나 연결을 해제하는 jQuery 메서드

형식	기능
<code>\$( ).on( )</code>	특정 엘리먼트(jQuery 객체)에 이벤트 핸들러를 연결
<code>\$( ).off( )</code>	<code>on()</code> 로 연결된 이벤트 핸들러를 해제
<code>\$( ).bind( )</code>	특정 엘리먼트(jQuery 객체)에 이벤트 핸들러를 연결
<code>\$( ).unbind( )</code>	<code>bind()</code> 로 연결된 이벤트 핸들러를 해제
<code>\$( ).one( )</code>	특정 엘리먼트(jQuery 객체)에 이벤트 핸들러를 단 한번 연결 후 해제(일회용)
<code>\$( ).trigger( )</code>	특정 엘리먼트(jQuery 객체)에 직접 이벤트를 발생시켜 이벤트 핸들러 함수를 실행
<code>\$( ).live( ), \$( ).delegate( )</code>	동적으로 미래에 생성될 특정 엘리먼트(jQuery 객체)에 이벤트 핸들러를 연결
<code>\$( ).die( ), \$( ).undelegate( )</code>	<code>live()</code> 로 연결된 이벤트 핸들러를 해제

# jQuery 라이브러리 사용

## □ 이벤트 핸들러 단축형 메서드

- 이벤트 타입명을 메서드 이름으로 사용 가능
- 직접 이벤트 핸들러를 설정하는 단축형 이벤트 메서드

분류	이벤트 메서드	기능
마우스	<code>\$( ).click( )</code>	엘리먼트 표시 영역을 마우스로 클릭할 때 동작
	<code>\$( ).hover( )</code>	엘리먼트 표시 영역 안으로 마우스 포인터가 들어올 때(또는 나갈 때) 동작
	<code>\$( ).toggle( )</code>	마우스를 클릭할 때마다 두 함수를 번갈아 동작
폼	<code>\$( ).focus( )</code>	폼 엘리먼트 표시 영역이 포커스를 얻을 때 동작
	<code>\$( ).blur( )</code>	폼 엘리먼트 표시 영역이 포커스를 잃을 때 동작
	<code>\$( ).change( )</code>	폼 엘리먼트 표시 영역의 값이 변경될 때 동작
	<code>\$( ).select( )</code>	폼 엘리먼트 표시 영역의 텍스트 일부를 선택할 때 동작
키보드	<code>\$( ).keydown( )</code>	키보드를 눌렀을 때 동작
문서	<code>\$( ).ready( )</code>	브라우저에 문서가 읽혀질 때 동작 DOM을 완전히 로드 했을 때 실행할 함수를 지정
	<code>\$( ).load( )</code>	브라우저에 문서 관련 모든 자원이 읽혀질 때 동작 엘리먼트의 모든 하위 엘리먼트를 로드 했을 때 실행할 함수를 지정
	<code>\$( ).unload( )</code>	브라우저에서 문서가 사라질 때 동작 현재 페이지를 떠나거나 이동할 경우 실행할 함수를 지정

# jQuery 라이브러리 사용

## □ 효과 메서드

- 특정 영역을 서서히 사라졌다가 다시 나타나게 하는 등의 간단한 애니메이션 효과를 메서드로 제공한다. (다양한 효과를 일정 시간 동안 계속 수행)

효과 유형	기능
show([ms][,function( )])	선택된 엘리먼트를 화면에서 보이게 함
hide([ms][,function( )])	선택된 엘리먼트를 화면에서 사라지게 함
toggle([ms][,function( )])	선택된 엘리먼트가 화면에 보였다가 사라지는 상태를 반복함 show()와 hide()를 번갈아 수행함
slideUp([ms][,function( )])	선택된 엘리먼트의 높이를 점차 위로 감소시켜 화면에서 사라지게 함 위로 접는 효과
slideDown([ms][,function( )])	선택된 엘리먼트의 높이를 점차 아래로 증가시켜 화면에서 보이게 함 아래로 펼치는 효과
slideToggle([ms][,function( )])	선택된 엘리먼트의 높이를 변경하여 화면에서 사라지거나 보이게 함
fadeIn([ms][,function( )])	선택된 엘리먼트의 불투명도를 점차 높여서 보이게 함
fadeOut([ms][,function( )])	선택된 엘리먼트의 불투명도를 점차 낮춰서 사라지게 함
fadeToggle([ms][,function( )])	선택된 엘리먼트의 불투명도를 변경하여 사라지거나 보이게 함

- 효과 메서드들의 세가지 입력 인자

```
show( ) ;                                // (1) 빈 입력인자
show(600) ; 또는 show('slow') ;          // (2) 수치, 문자열 입력인자
show(200, function( ) { . . . } ) ;        // (3) 콜백함수 입력인자
```



# jQuery 라이브러리 사용

## □ animate( ) 메서드

- 기본적인 효과 이외에 맞춤형 효과를 사용자가 직접 정의하여 사용
- 수치값을 사용하는 CSS3 스타일 속성값은 모두 사용 가능

효과 유형	기능
animate([properties][,ms][,function( ))]	선택된 엘리먼트에 대해 직접 설정한 효과를 통해 맞춤형 애니메이션 효과를 적용함

- 첫 번째 입력 인자인 특성(properties) 객체

- 다양한 움직임 효과를 줄 수 있는 CSS3 속성과 속성값을 지정
- CSS3 속성 중 크기나 길이, 비율 등 숫자를 사용하는 속성들을 맵 방식으로 명세  
예) 지속시간 : 1/1000초 단위의 숫자나 slow, normal, fast 문자열 중 하나

animate(특성객체, 지속시간, 콜백함수)

```
$('p').animate({font-size: "3em"}, 2000);           // 문단 글자 크기를 3배로 확대하는 효과
$('div').animate({height: "25%"}, "slow");        // div 영역의 높이를 1/4로 축소하는 효과
$('div:has(img)').animate({width: "0px"}, 1000);   // 이미지가 포함된 div 영역을 왼쪽으로 접는 효과
```



# Thank You

**HTML**



**CSS**



**JS**



## 제4장 HTML5 API

1. XMLHttpRequest API

2. Canvas API

3. Drag&Drop API

4. Multimedia API

5. File API

6. Web Storage AP



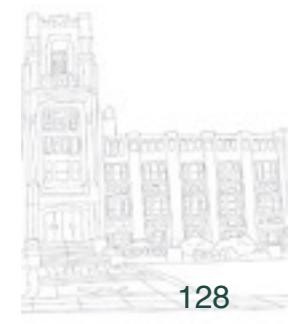
7. Indexed DB API

8. Web Messaging API

9. Server-Sent Events API

10. Web Socket API

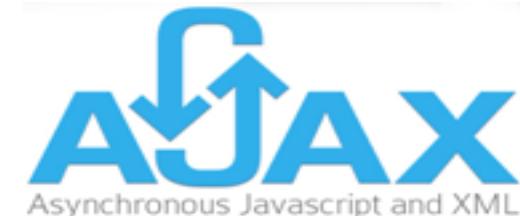
11. Web Worker API



# XMLHttpRequest API

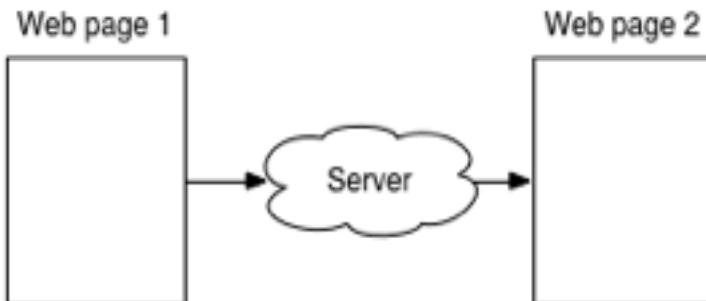
## □ AJAX 란?

- 전통적인 Web 의 통신 방법은 Web 페이지의 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 동적인 Web 페이지를 생성하는 기술이다.



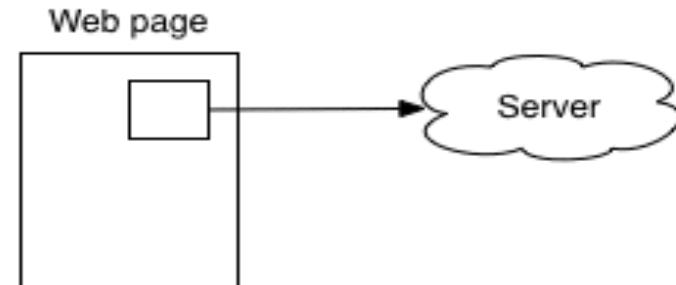
### Before Ajax

The whole page changes on an update



### With Ajax

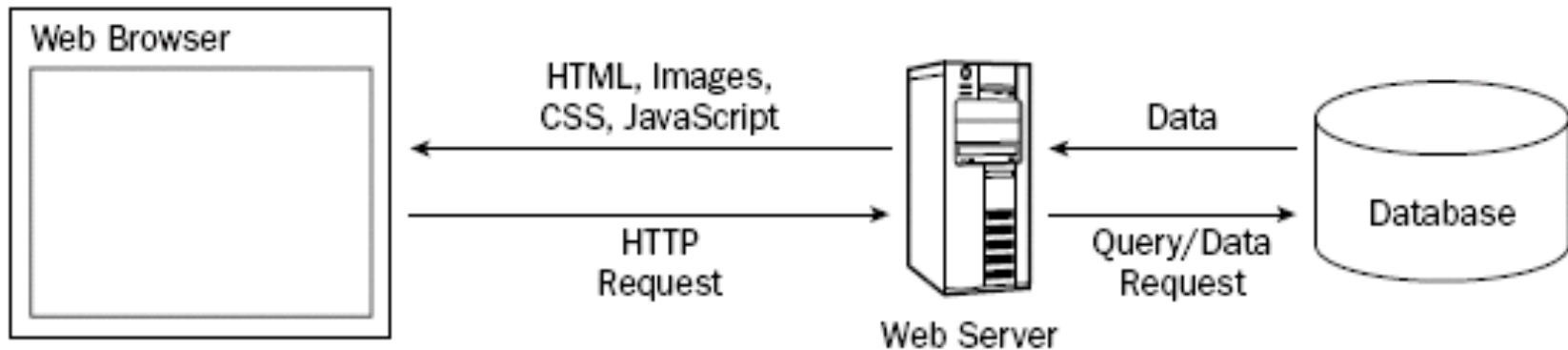
Only parts of the web page change on an update



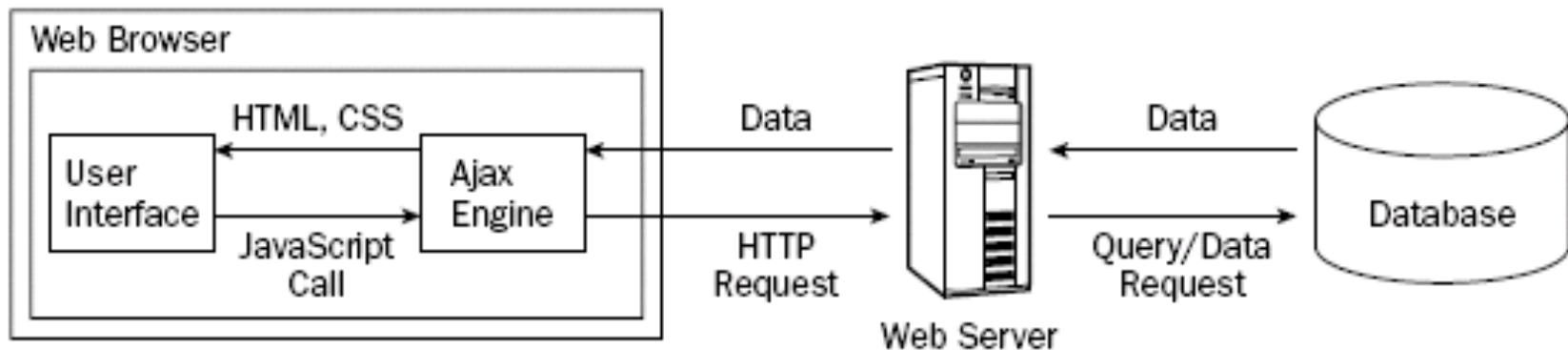
# XMLHttpRequest API

## □ 전통적인 Web 통신과 AJAX Web 통신

Traditional Web Application Model

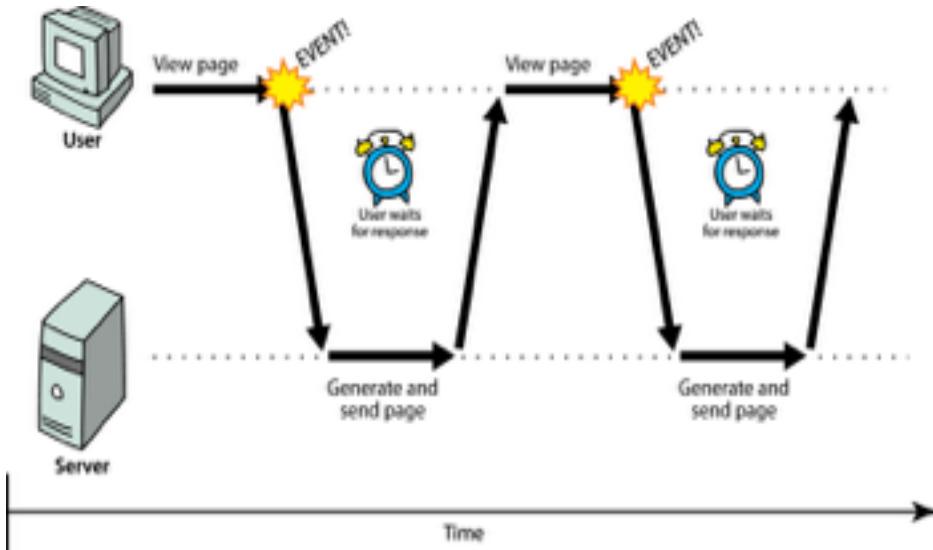


Ajax Web Application Model



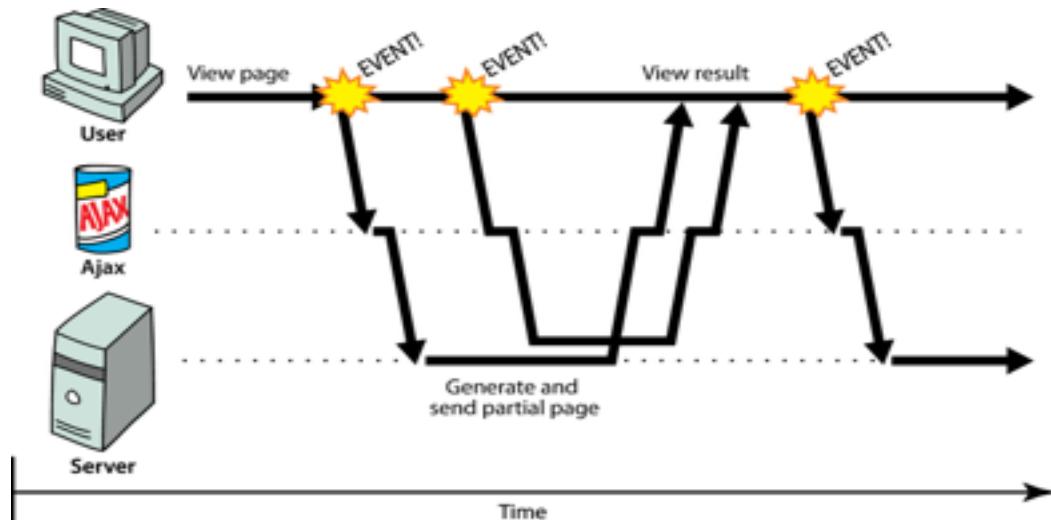
# XMLHttpRequest API

## □ 동기 통신과 비동기 통신



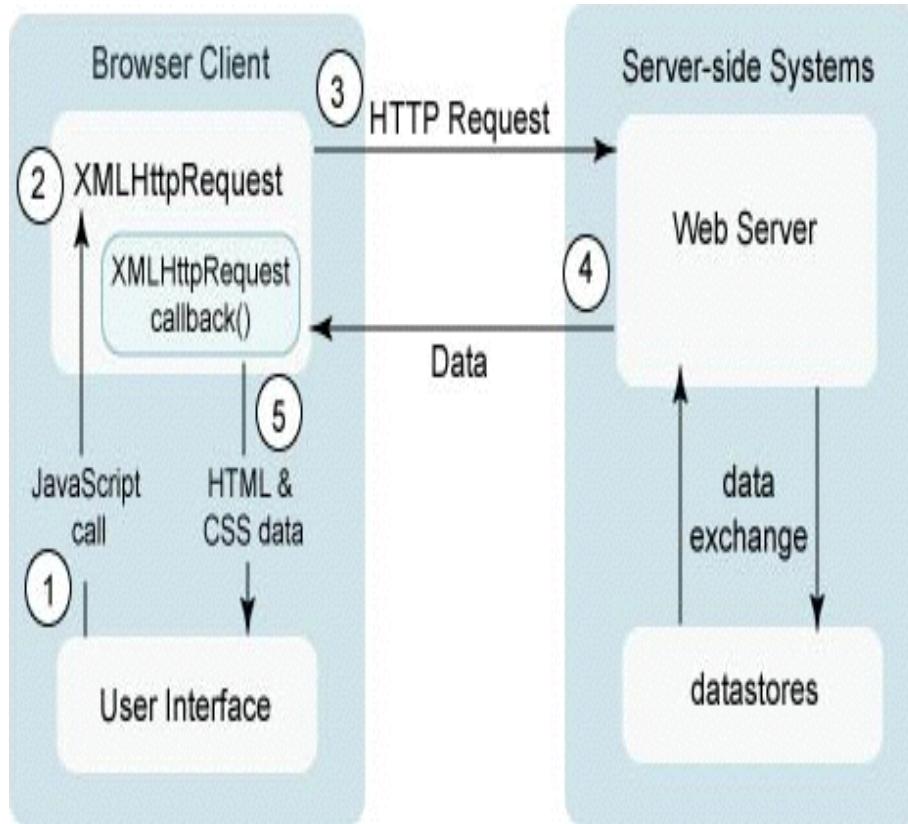
서버에 정보를 요청하면 서버로 부터 응답이 올 때까지 대기하게 된다. 또한 현재 페이지 전체가 응답된 내용으로 변경된다.(refresh)

AJAX 엔진을 통해서 서버에 요청하게 되며 서버로 부터 응답이 올 때까지 대기하지 않는다. 현재 페이지의 내용은 유지하면서 이벤트 핸들러를 통해 응답 내용을 현재 페이지에 동적으로 반영한다.



# XMLHttpRequest API

## □ 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 요청 받은 서버는 요청 결과를 적당 한 데이터로 구성하여 응답한다.
- ⑤ XMLHttpRequest 객체에 의해 등록된 콜백함수를 호출하여 응답 결과를 Web 페이지에 반영한다.

# XMLHttpRequest API

## □ 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()
- XMLHttpRequest 객체의 주요 멤버들

onreadystatechange	통신 상태가 변화된 타이밍에 호출되는 이벤트 핸들러
readyState	HTTP 통신 상태를 취득
status	HTTP Status 코드를 취득
statusText	HTTP Status의 상세 메시지를 취득
responseText	응답 본체를 plaintext로 취득
responseXML	응답 본체를 XML(XMLDocument 객체)로 취득
abort()	현재의 비동기 통신을 중단
getAllResponseHeaders()	수신한 모든 HTTP 응답 헤더를 취득
getResponseHeader(header)	지정한 HTTP 응답 헤더를 취득
open( ... )	HTTP 요청을 초기화
setRequestHeader(header,value)	요청 시에 송신하는 헤더를 추가
send(body)	HTTP 요청을 송신(인수 body는 요청 본체)
upload	XMLHttpRequestUpload 객체 추출

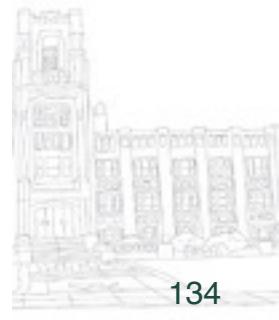
# XMLHttpRequest API

## □ status 프로퍼티의 반환 값

200	OK(처리 성공)
401	Unauthorized(인증이 필요)
403	Forbidden(액세스가 거부되었다)
404	Not Found(요청된 자원이 존재하지 않는다)
500	Internal Server Error(내부 서버 에러)
503	Service Unavailable(요구한 서버가 이용 불가)

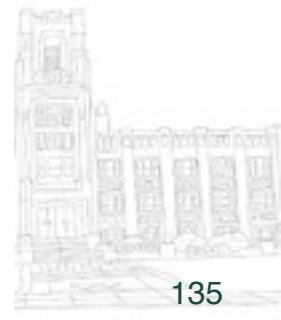
## □ readyState 프로퍼티의 값

- 0 - 미초기화(open()) 이 호출되지 않음)
- 1 - 로드 중(open 메서드는 호출됐지만, send 메서드는 호출되지 않았다)
- 2 - 로드 완료(send 메서드는 호출됐지만, 응답 스테이터스/헤더는 미취득)
- 3 - 일부 응답을 취득(응답 스테이터스/헤더만 취득, 본체는 미취득)
- 4 - 모든 응답 데이터를 취득 완료



# XMLHttpRequest API

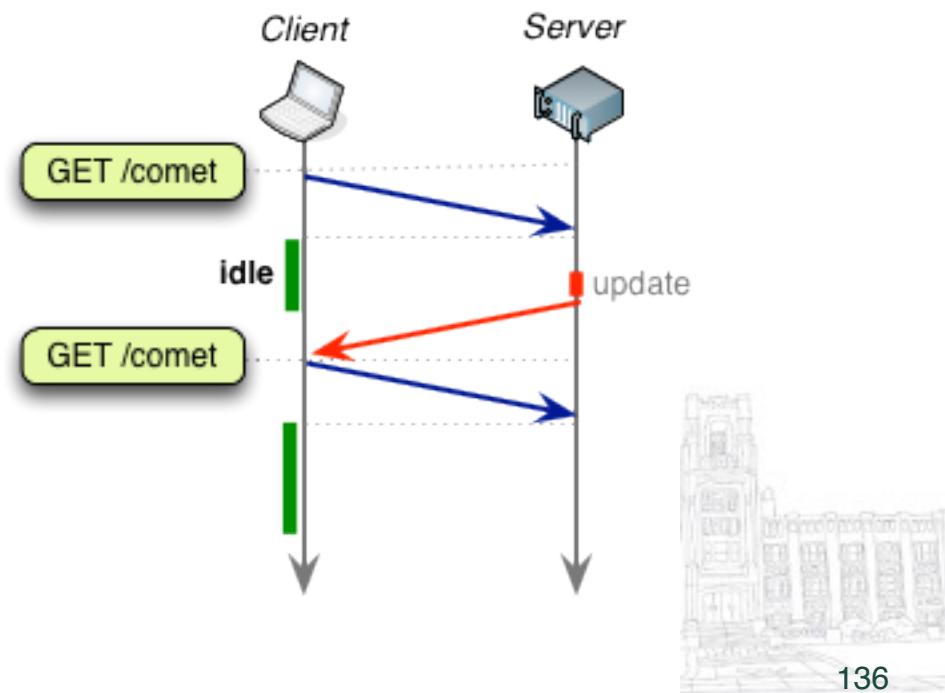
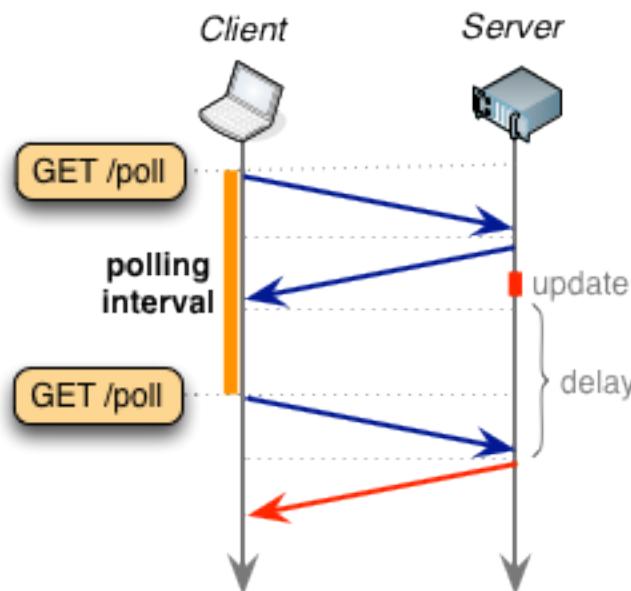
- open()와 send() 메서드
  - open(HTTP 메서드, URL [, 비동기 모드 통신 여부])
    - HTTP 메서드 : 요청 방식(GET, POST, PUT, DELETE..)
    - URL : AJAX로 요청하려는 서버의 대상 페이지
    - 비동기 모드 통신 여부 : true(비동기통신), false(동기통신)
  - send([요청 파라미터])
    - POST의 경우 Query 문자열을 인수로 지정
    - ArrayBufferView, Blob, Document, DOMString, FormData, null이 올 수 있다.
- XMLHttpRequest 객체에서 제공되는 이벤트 관련 속성
  - onloadstart
  - onprogress
  - onabort
  - onerror
  - onload
  - ontimeout
  - onloadend
  - onreadystatechange



# XMLHttpRequest API

## □ 서버 푸시

- 푸시 기법 또는 서버 푸시(server push)는 인터넷 상에서 어떤 전송 요청이 중앙 서버에서 시작되는 정보 전달 방식이다. 이것은 전송 요청이 클라이언트에서 시작되는 풀 기법과 대비되는 것이다.
- 사용자가 원하든 원치 않든 방송처럼 뉴스를 제공하는 기술로서 사용자가 원하는 정보를 직접 찾는 풀 기법과는 상반되는 개념이다.
- AJAX 기술을 이용해서도 서버 푸시 효과를 구현할 수 있으며 Polling 과 Long Polling (comet)이 사용된다.

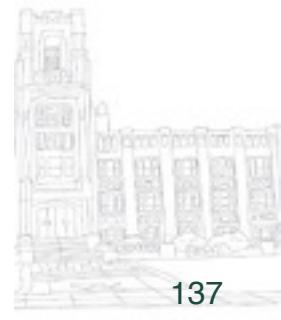


# XMLHttpRequest API

## □ jQuery의 AJAX 지원

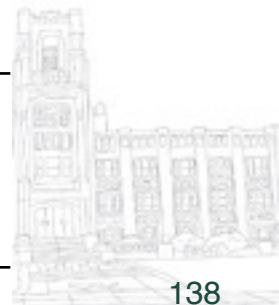
- jQuery는 웹 브라우저 종류에 상관없이 같은 방식으로 Ajax 기능을 구현하도록 다양한 메소드를 제공한다.
- jQuery Mobile은 페이지 이동을 위해 Ajax 기술을 사용한다.
- AJAX를 지원하는 jQuery 메서드들

Ajax 메소드	기능/예
<code>\$.ajax( )</code>	모든 Ajax 메소드의 기본이 되는 메소드 예) <code>\$.ajax({     url: 'service.php',     success: function(data) {         \$('#area').html(data);     } });</code>
<code>\$.get( )</code>	GET 방식의 ajax( ) 메소드 예) <code>\$.get('sample.html', function(data) {     \$('#area').html(data); });</code>
<code>\$.post( )</code>	POST 방식의 ajax( ) 메소드 예) <code>\$.post('sample.html', function(data) {     \$('#area').html(data); });</code>



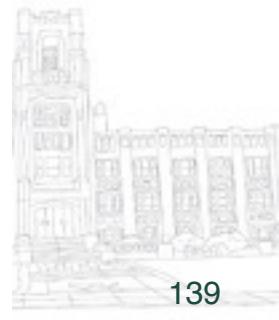
# XMLHttpRequest API

<code>\$.getJSON( )</code>	JSON 형식으로 응답 받는 ajax( ) 메소드 예) <code>\$.getJSON('sample.json', function(data) {     \$('#area').html('&lt;p&gt;' + data.age + '&lt;/p&gt;'); });</code>
<code>load( )</code>	서버로부터 데이터를 받아서 일치하는 요소 안에 HTML을 추가 예) <code>\$('#area').load('sample.html', function() {     ; });</code>
<code>\$.getScript( )</code>	자바스크립트 형식으로 응답 받는 ajax( ) 메소드 예) <code>\$.getScript('sample.js', function() {     ; });</code>
<code>\$.ajaxSetup( )</code>	ajax( ) 메소드의 선택 사항들에 대한 기본값 설정 예) <code>\$.ajaxSetup({     url: 'service.php' });</code>
<code>ajaxStart( )</code>	첫 번째 Ajax 요청이 시작될 때 호출되는 이벤트 메소드 예) <code>\$('#img1').ajaxStart(function(){     \$(this).show(); });</code>
<code>ajaxStop( )</code>	모든 Ajax 요청이 끝날 때 호출되는 이벤트 메소드 예) <code>\$('#img1').ajaxStop(function(){     \$(this).fadeOut(2000); });</code>



# XMLHttpRequest API

ajaxSend( )	특정 Ajax 요청을 보내기 전에 호출되는 이벤트 메소드 예) \$("#msg").ajaxSend(function(event, request, settings){ \$(this).append("<p>" + settings.url + "페이지 요청 시작</p>"); });
ajaxSuccess( )	특정 Ajax 요청이 성공적으로 완료될 때마다 호출되는 이벤트 메소드 예) \$("#msg").ajaxSuccess(function(event, request, settings){ \$(this).append("<p>요청 성공</p>"); });
ajaxError( )	Ajax 요청들에 대한 오류 발생시 호출되는 이벤트 메소드 예) \$("#msg").ajaxError(function(event, request, settings){ \$(this).append("<p>" + settings.url + "페이지 요청 실패</p>"); });
ajaxComplete( )	Ajax 요청들이 완료되면(성공/실패 관련 없이) 호출되는 이벤트 메소드 예) \$("#msg").ajaxComplete(function(event,request, settings){ \$(this).append("<p>요청 완료</p>"); });



# XMLHttpRequest API

## □ \$.ajax( ) 메소드

- 모든 Ajax 메소드가 내부적으로는 사용하는 기본 메소드
- Ajax 요청을 기본적인 부분부터 직접 설정하고 제어할 수 있어 다른 Ajax 메소드로 할 수 없는 요청도 수행 가능
- \$.ajax() 메소드의 기본 형식 : \$.ajax( url [, settings ] )
- settings : 선택항목으로서 JavaScript 객체 형식으로 지정

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

# XMLHttpRequest API

## \$.getJSON( ) 메소드

- GET 요청 방식으로 서버로부터 JSON 형식의 데이터를 요청
- \$.getJSON( url [, data ] [, success ] )

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)
success(data)	요청 성공 콜백 함수 (data: 서버 반환 값)

## \$.load( ) 메소드

- 서버로부터 데이터를 받아오는 가장 간단한 메소드
- 서버로부터 데이터를 받아 메소드를 실행하는 대상 엘리먼트에 컨텐트로 추가

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)
success(data)	요청 성공 콜백 함수(data: 서버 반환 값)

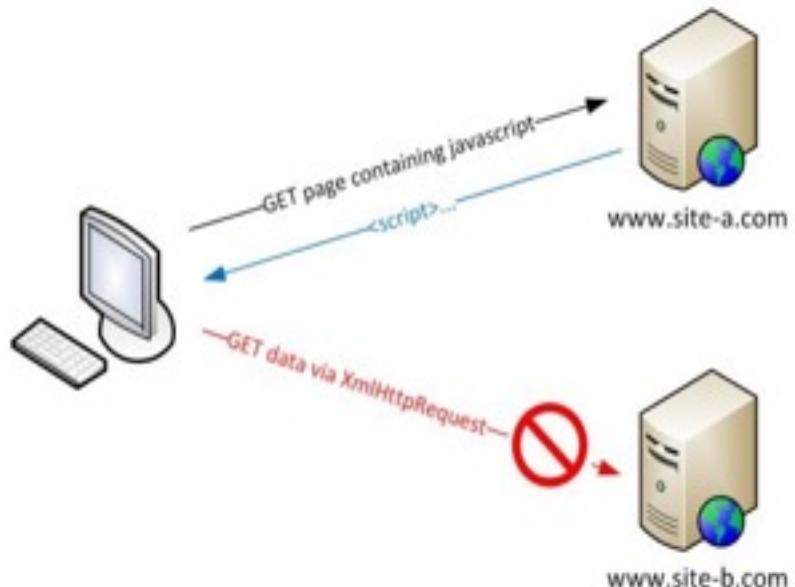
# XMLHttpRequest API

## □ AJAX 의 크로스 도메인 통신

- Same Origin Policy(SOP) 제약  
브라우저에서 보안상의 이유로 동일 사이트의 자원(Resource)만 접근해야 한다는 제약이다.

- SOP를 해결해야 하는 배경
  - Open API 활성화  
여러 서비스 회사에서 Open API를 널리 보급하게 됨
  - In-House에서도 공통 서비스를 Open API 형태로 만들고 재활용

- RIA 및 모바일 활성화  
클라이언트 단에서 OpenAPI를 이용하여 풍부한 기능 제공(메시업)  
모바일은 기존 서비스를 재활용하면서 가공하여 제공하는 경향이 높  
하이브리드 앱의 출현으로 file:// 에서 서버 접근 필요

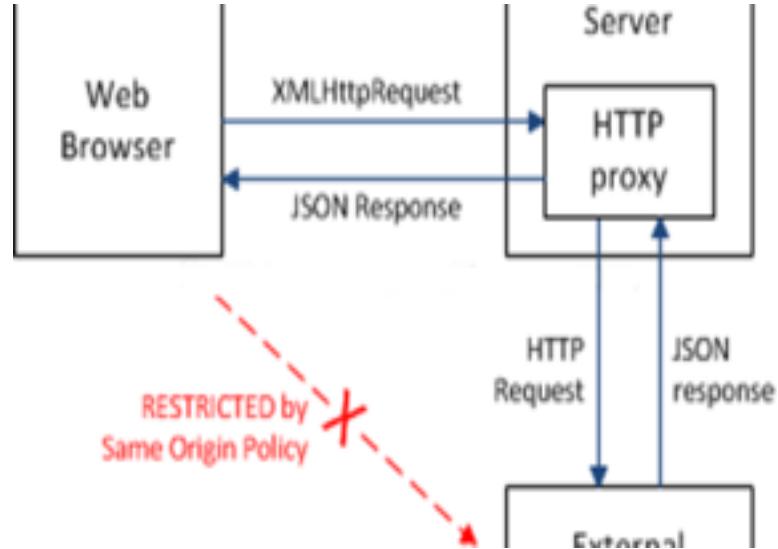


# XMLHttpRequest API

## □ AJAX 통신에서 SOP 해결 방법 – 1

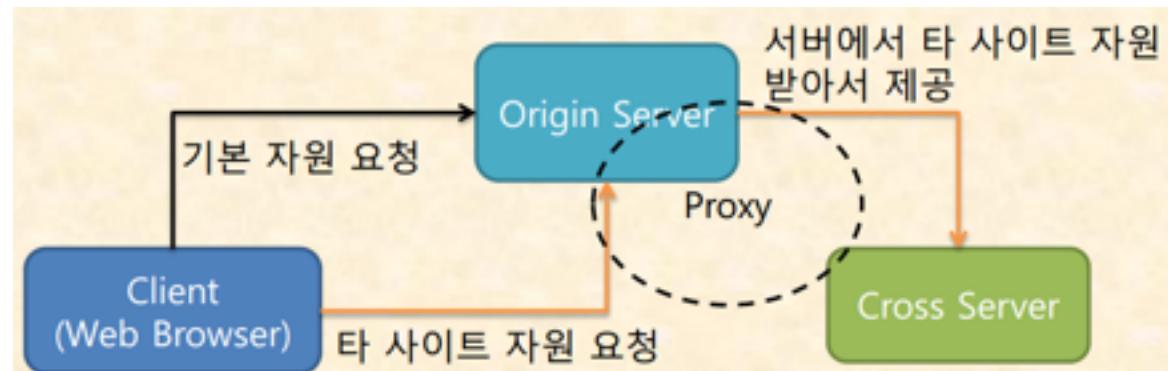
- Client는 모든 요청을 Origin 서버로 요청한다.
- 타 사이트(서버)의 자원은 Origin 서버에서 타 서버의 자원을 받아서 응답을 줌

## Proxy서버 개발



### [ 이슈 ]

- Proxy처리를 위한 기능을 Origin 서버에 개발해야 함
- 로컬 파일(file://)은 원천 서버가 없으므로 사용 불가



# XMLHttpRequest API

## □ AJAX 통신에서 SOP 해결 방법 – 2

### JSONP 적용

#### ■ JSONP의 장점

- Cross Origin 제약과는 아무 상관 없음
- 로컬 파일(file://)에서도 사용 가능
- Client에서 적절한 API를 사용하면 일반 AJAX와 별 차이없이 구현 가능

JSON

```
{  
  "roses": "red",  
  "violets": "blue",  
  "grass": "green"  
}
```

JSONP

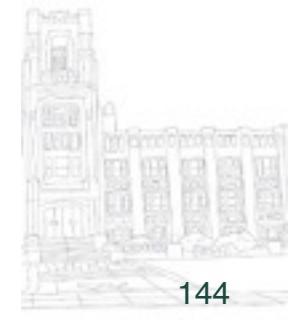
P for padding

grab({

```
  "roses": "red",  
  "violets": "blue",  
  "grass": "green"  
})
```

#### [ 이슈 ]

- 기술 원리상 GET만 가능함
- Script 보안을 교묘하게 회피하는 방법이므로 추후에 문제가 될 가능성성이 많음



# XMLHttpRequest API

[ 클라이언트 소스 ]

```
<html>
<script>
    function parseResponse(data) { ... }
</script>
<body>
...
<script type="application/javascript"
src="http://서버주소/xxx/yyy?jsonp=parseResponse">
</script>
```

[ Server 응답(텍스트) ]

```
parseResponse({
    "Name": "Foo",
    "Id": 1234,
    "Rank": 7
});
```

[ 클라이언트 소스 ]

```
<script type="text/javascript">
    function dataHandler(objData) {
        console.log(objData.one);
    };
</script>
<script src="http://서버주소/xxx/dataP.json"> </script>
```

[ Server 응답(텍스트) ]

```
dataHandler( {
    one: "Singular sensation",
    two: "Beady little eyes",
    three: "Little birds pitch by my
           doorstep"
});
```



# XMLHttpRequest API

- jQuery 라이브러리로 구현하는 JSONP 통신

```
$getJSON("http://서버주소/server/data.jsp?callback=?", function(d) {  
    .....  
});  
  
$.ajax({  
    url : "http://서버주소/xxx/data.jsp",  
    dataType : "jsonp ",  
    jsonp : "callback",  
    success : function(d){ ..... }  
});  
  
$('a').click(function() {  
    $.ajax({  
        url: "http://서버주소/jsonp.json",  
        dataType: 'jsonp',  
        jsonpCallback: "myCallback",  
        success: function(data) { console.log('성공 - ', data); },  
        error: function(xhr) { console.log('실패 - ', xhr); }  
    });  
});  
  
function dataHandler(objData) {  
    console.log(objData.one);  
};  
  
$(function() {  
    $.ajax({  
        type: 'GET',  
        url: 'http://서버주소/xxx/dataP.json',  
        async: true,  
        jsonpCallback: 'dataHandler',  
        contentType: "application/json",  
        dataType: 'jsonp',  
        success: function(objData) {  
            console.log(objData.one);  
        },  
        error: function(e) {  
            console.log(e.message);  
        }  
    });  
});
```

# XMLHttpRequest API

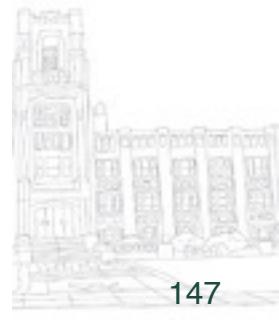
[ AJAX 로 XML 문서를 요청하여 처리하는 JavaScript 소스 ]

```
var request = new XMLHttpRequest();
request.onreadystatechange = function (event) {
    alert('readyState 값 : ' + request.readyState + ' HTTP 응답 값 : ' + request.status);
    if (request.readyState == 4) {
        if (request.status == 200) {
            var xml = request.responseXML;
            var rootE = xml.getElementsByTagName('testxml');
            var output = "";
            for(var i=1; i < rootE[0].childNodes.length; i+=2)
                output += "<h1>" + rootE[0].childNodes[i].firstChild.nodeValue + "</h1>";

            document.body.innerHTML += output;
        }
    }
};
request.open('GET', 'content/testxml.xml', true);
request.send();
```

[ XML 문서 ]

```
<?xml version="1.0" encoding="utf-8" ?>
<testxml>
    <name>자바스크립트</name>
    <age>19</age>
    <kind>웹앱개발 전용 OOP 언어</kind>
</testxml>
```



# XMLHttpRequest API

[ AJAX 로 JSON 문서를 요청하여 처리하는 JavaScript 소스 ]

```
var request = new XMLHttpRequest();
request.onreadystatechange = function (event) {
    if (request.readyState == 4) {
        if (request.status == 200) {
            var str = request.responseText;
            alert(str);
            var result = JSON.parse(str);
            var output = "";
            for(var i in result)
                output += "<h1>" + result[i] + '</h1>';
            document.body.innerHTML += output;
        }
    }
};
request.open('GET', 'content/testjson.txt', true);
request.send();
```

[ JSON 문서 ]

```
{ "name" : "자바스크립트", "age" : 19, "kind" : "웹앱개발 전용 OOP 언어"}
```

# XMLHttpRequest API

[ AJAX 로 XML 문서를 요청하여 처리하는 jQuery 활용 소스 ]

```
$.ajax('content/testxml.xml', {
  success : function(data) {
    alert(data);
    $(data).find('testxml').each(function() {
      $('body').append("<h1>name :" + $(this).find('name').text() + '</h1>');
      $('body').append("<h1>age :" + $(this).find('age').text() + '</h1>');
      $('body').append("<h1>kind :" + $(this).find('kind').text() + '</h1>');
    });
  }
});
```

[ AJAX 로 JSON 문서를 요청하여 처리하는 jQuery 활용 소스 ]

```
$.ajax('content/testjson.txt', {
  success : function(data) {
    var result = JSON.parse(data);
    $.each(result, function(key, value) {
      $('body').append("<h1>" +key+ " :" +value + '</h1>');
    });
  }
});
```

```
$.getJSON('content/testjson.txt', function(data) {
  $.each(data, function(key, value) {
    $('body').append("<h1>" +key+ " :" +value + '</h1>');
  });
});
```

# XMLHttpRequest API

## □ XMLHttpRequest Level 2( XHR 2)

- HTML5에 추가된 AJAX 의 향상된 스펙

- CORS (AJAX 로 Cross Domain 을 지원하기 위한 스펙)
- 이벤트 처리를 통한 AJAX 구현(load, error, loadstart, progress...)
- 요청하면서 받고자 하는 데이터의 타입을 지정할 수 있다.

responseType : "", "arraybuffer", "blob", "document", "json", "text" 중 하나를 설정한다.

response : 응답된 내용을 추출하는 용도로 사용한다.

- FormData 지원 : multipart/form-data 타입으로 구성된 데이터를 POST 방식으로 전달하면서 AJAX 요청하는 프로그램을 쉽게 구현할 수 있다.

문자열 뿐만아니라 바이너리도 처리 가능하다.

```
Content-Length: 117262
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryhBNdxb00beVuLete
Cookie: JSESSIONID=43AA44E847AA8B741DEE47C87A5A5347
Host: localhost:8080
Origin: http://localhost:8080
Referer: http://localhost:8080/edu/ajaxhtml5exam/new/exam4.html
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.130 Safari/537.36
```

### ▼ Request Payload

```
-----WebKitFormBoundaryhBNdxb00beVuLete
Content-Disposition: form-data; name="myuploadfile"; filename="duke_evolution.png"
Content-Type: image/png
```

```
-----WebKitFormBoundaryhBNdxb00beVuLete
Content-Disposition: form-data; name="myuploadfile"; filename="html5.jpg"
Content-Type: image/jpeg
```



# XMLHttpRequest API

[ FormData 객체를 사용하여 POST 방식으로 요청하는 예제 1 ]

```
<section id="formbox">
  <form name="form">
    <p><input type="button" name="button" id="button" value="AJAX요청"></p>
  </form>
</section>
<section id="databox"></section>
<script>
function initiate(){
  databox=document.getElementById('databox');
  var button=document.getElementById('button');
  button.addEventListener('click', send, false);
}
function send(){
  var data=new FormData();
  data.append('name','JavaScript');
  data.append('age','19');
  data.append('kind', "OOP Based Web Programming Language");
  var url="test.jsp";
  var request=new XMLHttpRequest();
  request.addEventListener('load',show,false);
  request.open("POST", url, true);
  request.send(data);
}
function show(e){
  databox.innerHTML=e.target.responseText;
}
window.addEventListener('load', initiate, false);
</script>
```

# XMLHttpRequest API

[ FormData 객체를 사용하여 POST 방식으로 요청하는 예제 2 ]

```
<h1>AJAX 로 FormData 를 구성하여 전송하는 예제</h1>
<h2>좋아하는 프로그래밍 언어에 대하여 입력하세요...</h2>
<form id="myform" action="test.jsp">
    언어명칭 : <input type="text" name="name" required autofocus><br>
    나이 : <input type="number" name="age" required><br>
    종류 : <input type="text" name="kind" size="50" required><br>
    <input type="submit" onclick="return sendForm(this.form);">
</form>
<div id="result"></div>
<script>
function sendForm(form) {
    var formData = new FormData(form);
    formData.append('secret_token', '1234567890');
    var xhr = new XMLHttpRequest();
    xhr.open('POST', form.action, true);
    xhr.responseType = 'text';
    xhr.onload = function(e) {
        if (this.status == 200) {
            document.getElementById("result").innerHTML = "<h2>응답 결과</h2>" + this.responseText;
        }
    };
    xhr.send(formData);
    return false;
}
</script>
```

# XMLHttpRequest API

[ FormData 객체를 사용하여 POST 방식으로 요청하는 예제 3 ]

```
<h1 onclick="getImage();">클릭해보세요... AJAX로 이미지를 요청하고 출력해요....</h1>
<script>
function getImage() {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', 'duke_luau.png', true);
    xhr.responseType = 'blob';

    xhr.onload = function(e) {
        if (this.status == 200) {
            var blob = this.response;
            var img = document.createElement('img');
            img.width=200;
            img.height=200;
            img.onload = function(e) {
                window.URL.revokeObjectURL(img.src);
            };
            img.src = URL.createObjectURL(blob);
            document.body.appendChild(img);
        }
    };
    xhr.send();
}
</script>
```

# XMLHttpRequest API

## [ FormData 객체를 사용하여 POST 방식으로 요청하는 예제 4 ]

```
<h1>HTML5 의 FormData 객체로 파일업로드 하는 예제</h1>
<hr>
<h2>업로드하려는 파일을 선택해 주세요.</h2>
<form id="testform">
<input type="file" name="myuploadfile" multiple><br><br>
<input type="button" id="uploadbutton" value="전송" >
</form>
<script>
function process() {
    var btn = document.querySelector("#uploadbutton");
    btn.onclick = function() {
        var form = document.querySelector("#testform");
        var xhr = new XMLHttpRequest();
        var formData = new FormData(form);
        xhr.open("POST", "process.jsp", true);
        xhr.onload = function(oEvent) {
            alert("전송완료");
        };
        xhr.send(formData);
    }
}
window.onload = process;
</script>
```

-----  
-----WebKitFormBoundaryhBNdxb00beVuLete  
Content-Disposition: form-data; name="myuploadfile"; filename="duke\_evolution.png"  
Content-Type: image/png

-----WebKitFormBoundaryhBNdxb00beVuLete  
Content-Disposition: form-data; name="myuploadfile"; filename="html5.jpg"  
Content-Type: image/jpeg

-----WebKitFormBoundaryhBNdxb00beVuLete  
Content-Disposition: form-data; name="myuploadfile"; filename="구글PPAPI.docx"  
Content-Type: application/haansoftdocx

-----WebKitFormBoundaryhBNdxb00beVuLete--

# XMLHttpRequest API

## □ Cross Origin Resource Sharing(CORS)

- 초기에는 Cross Domain이라고 하였다.(동일 도메인에서 포트만 다른 경우, 로컬 파일인 경우 등으로 인해 Origin이라는 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 응답 헤더에 다음과 같이 CORS 와 관련된 항목을 추가한다. 다음에 제시된 헤더 내용은 어떠한 오리진을 갖는 페이지라 해도 접근을 허용한다는 의미이다.

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

[ Access-Control 과 관련된 헤더 ]

```
response.setHeader("Access-Control-Allow-Methods", "POST, GET, OPTIONS, DELETE");  
POST, GET, OPTIONS, DELETE 요청에 대해 허용하겠다는 의미이다.
```

```
response.setHeader("Access-Control-Allow-Headers", "x-requested-with");
```

이는 표준화된 규약은 아니지만, 보통 AJAX 호출이라는 것을 의미하기 위해 비공식적으로 사용되는 절차로  
jQuery 또한 AJAX 요청 시, 이 헤더(x-requested-with)를 포함한다.

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

\* 는 모든 도메인에 대해 허용하겠다는 의미이며 \* 대신 특정 도메인만을 지정할 수도 있다.

# Canvas API

## □ Canvas API 란?

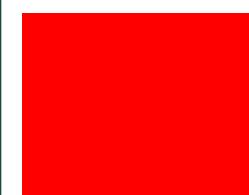
- 웹 페이지에 그림을 그릴 수 있도록 지원하는 HTML5 API 이다.
- <canvas> 엘리먼트를 사용하여 그림을 그리기 위한 영역을 정의하고 스크립트로 그림을 그린다.
- 직선, 박스, 원, 베지에 곡선 등 다양한 그림을 직접 그릴 수 있으며 원하는 사이즈 그리고 칼라의 이미지 출력을 처리할 수 있다.
- <canvas> 엘리먼트 작성 방법 : 그림을 그릴 수 있는 사각형 영역이 만들어진다.

```
<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.

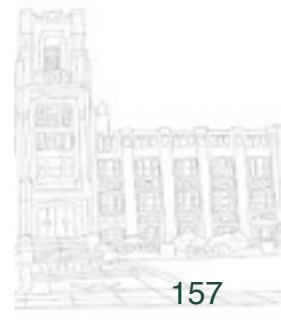
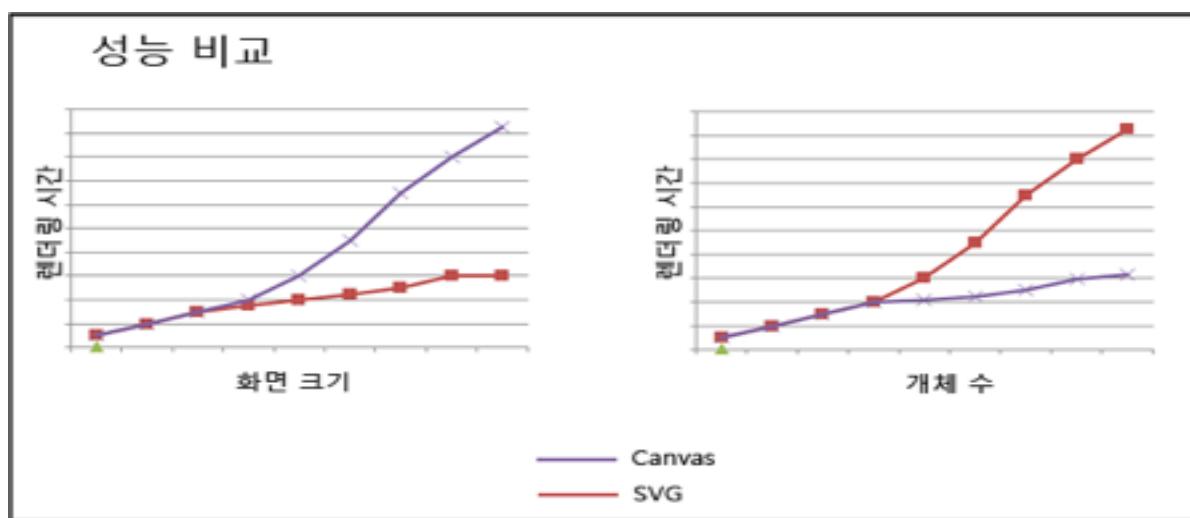
```
var area = document.getElementById("draw");
var ctx = area.getContext("2d");
ctx.fillStyle = "rgb(255,0,0)";
ctx.fillRect (10, 10, 100, 100);
```



# Canvas API

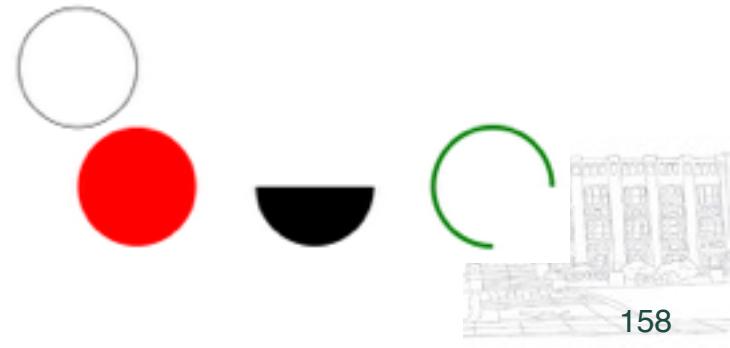
## □ Canvas API 와 SVG 비교

	Canvas	SVG
그래픽 시스템	픽셀 기반의 즉시 모드 그래픽 시스템	모양 기반의 유지모드 그래픽 시스템
이미지 처리방식	Bitmap (해상도에 의존적)	Vector (해상도에 독립적)
DOM	존재하지 않음 (DOM Control 불가)	존재함 (Script로 Control 가능)
외부 이미지 편집	Bitmap image 편집 용이	Vector image 편집 용이
성능 저하 요인	높은 해상도의 이미지를 사용하면 성능 저하	이미지가 복잡해질수록 Markup 이 복잡해져 성능이 저하
Animation	Animation API 가 없으므로 Script 의 Timer를 사용	높은 수준의 Animation을 지원
외부 이미지로 저장	jpg, png 등으로 저장 가능	불가
적합한 서비스	Graph구현, Game, 실시간 데이터 출력, 동영상 조작	Graph구현, 매우 세밀한 해상도를 지원하는 UI 및 Application
적합하지 않은 서비스	Standalone Application UI	Game, 실시간 데이터 출력



## □ 그리기 기능을 지원하는 메서드들

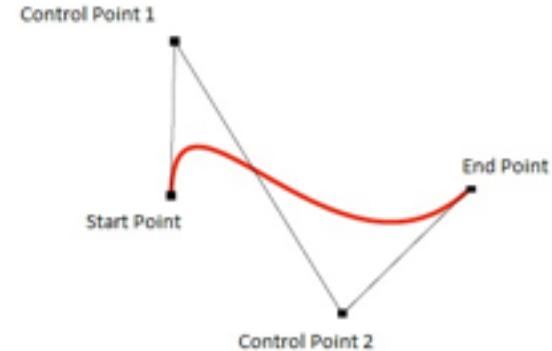
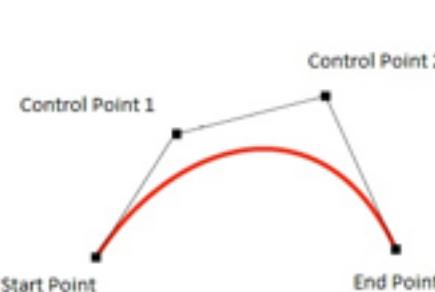
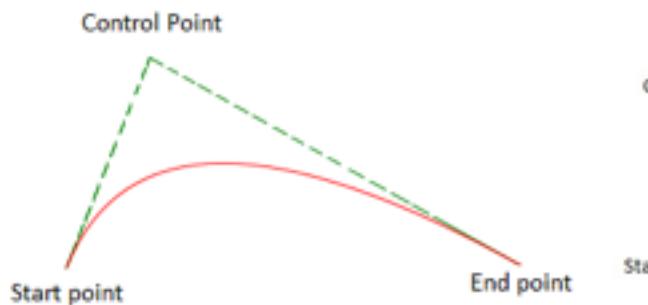
- fillRect(x, y, width, height) : 색이 칠해진 사각형을 그린다.
- strokeRect(x, y, width, height) : 테두리만 있는 사각형을 그린다.
- clearRect(x, y, width, height) : 특정 영역을 지우고 완전히 투명하게 만든다.
- beginPath() : 경로를 시작한다.
- closePath() : 경로를 종료한다.
- stroke() : 경로를 따라서 테두리 선을 그린다.
- fill() : 설정된 스타일로 도형을 채운다.
- moveto(x,y) : (x,y) 위치로 시작점을 옮긴다.
- lineto(x,y) : x에서 y까지 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.



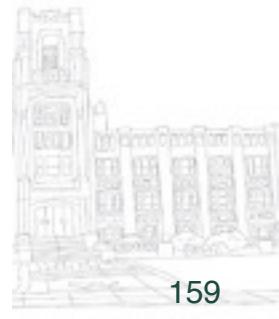
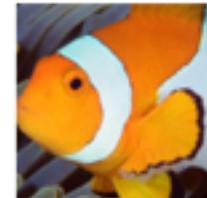
# Canvas API

## □ 그리기 기능을 지원하는 메서드들

- `quadraticCurveTo(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.



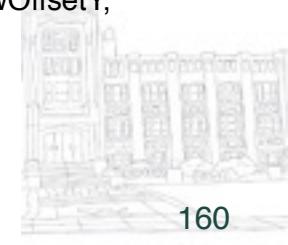
# Canvas API

## □ 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
    - strokeStyle과 fillStyle, globalAlpha, lineWidth, lineCap, lineJoin, miterLimit, shadowOffsetX, shadowOffsetY, shadowBlur, shadowColor, globalCompositeOperation 속성
- 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 , 1이 기본값

- `lineCap` : 선의 끝모양을 결정한다.

- `butt` : 기본값으로 아무런 효과 없음

- `round` : 선 너비의 1/2을 반지름으로 하는 반원이 선 양쪽 끝에

그려서 표시

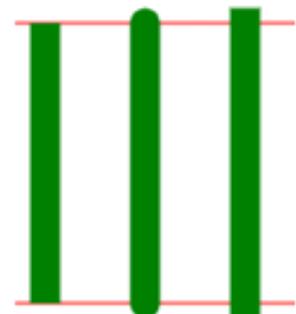
- `square` : 선 양쪽 끝에 사각형을 그려서 표시

- `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

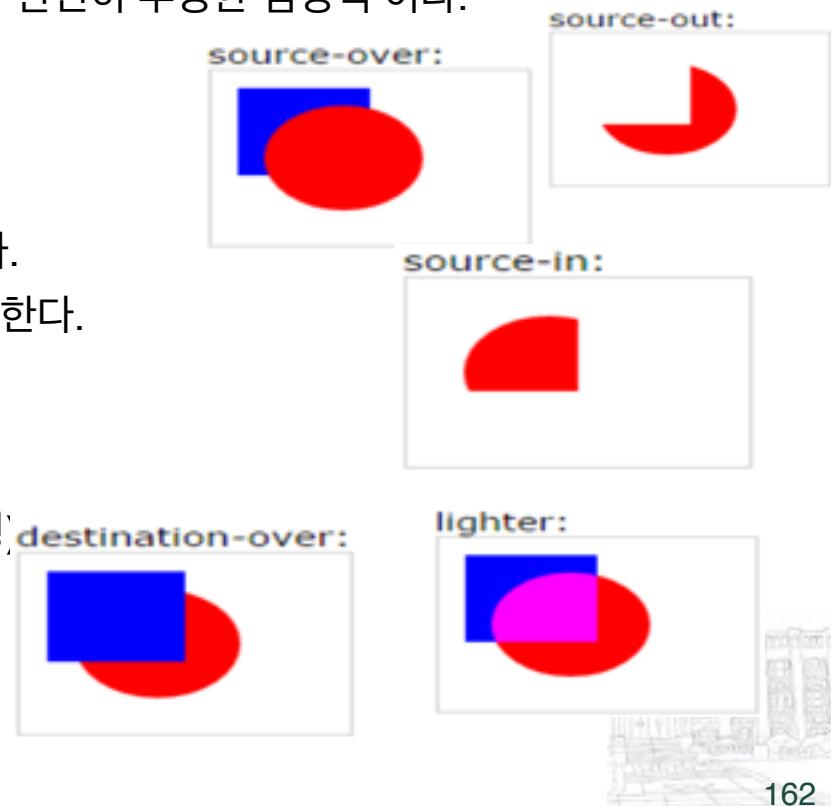
- shadowOffsetX
  - 객체로부터 그림자가 x축 방향으로 얼마나 떨어져 있는지 표시한다.(기본값은 0, 음수이면 왼쪽)
- shadowOffsetY
  - 객체로부터 그림자가 y축 방향으로 얼마나 떨어져 있는지 표시한다.(기본값은 0, 음수이면 위쪽)
- shadowBlur : 그림자가 얼마나 흐릿한지 나타낸다.(기본값은 0)
- shadowColor : 그림자 색상을 지정한다. 기본값은 완전히 투명한 검정색이다.

## □ 도형 변형

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

## □ 도형 합성

- globalCompositeOperation : 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.

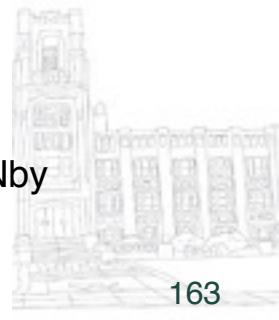
- putImageData(ImageData 객체, dx, dy) :

<canvas> 객체의 (dx, dy) 위치에 비트맵 이미지 객체(ImageData)의 데이터를 출력한다.

### ■ HTMLObjectCanvas 객체 저장

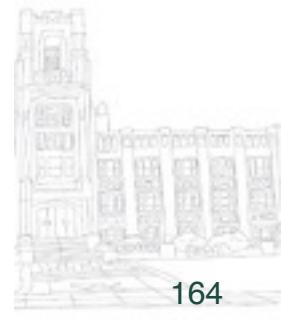
toDataURL() : <canvas> 태그 영역의 모든 내용을 png 형식의 URI 문자열로 변환하여 리턴한다.

```
var canvas = document.getElementById("draw");
var dataURL = canvas.toDataURL();
console.log(dataURL);
// "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAUAAAAFCAYAAACNby
// bIAAAADEIEQVQlWNgoBMAAABpAAFEI8ARAAAAAEIFTkSuQmCC"
```



## Canvas API

- 예제 : memo\_canvas.html



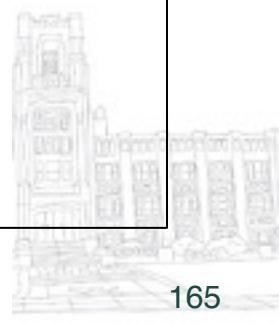
# Canvas API

## [ HTML 소스 ]

```
<button onclick="setColor('red');">RED</button>
<button onclick="setColor('blue');">BLUE</button>
<button onclick="setColor('green');">GREEN</button>
<button onclick="setColor('yellow');">YELLOW</button>
<button onclick="setColor('black');">BLACK</button><br><br>
<canvas id="myCanvas" width="580" height="450"></canvas>
```

## [ JavaScript 소스 ]

```
function setColor(color) {
drawColor = color;
}
function initialize() {
    context.clearRect(0,0,580,450);
    context.beginPath();
    context.rect(0,0,580,450);
    context.strokeStyle = "silver";
    context.fillStyle = "LightGoldenrodYellow";
    context.fill();
    context.lineWidth = 0.5;
    for(i=1;i<=8;i++) {
        context.moveTo(5,i*50);
        context.lineTo(575, i*50);
    }
    context.stroke();
}
```



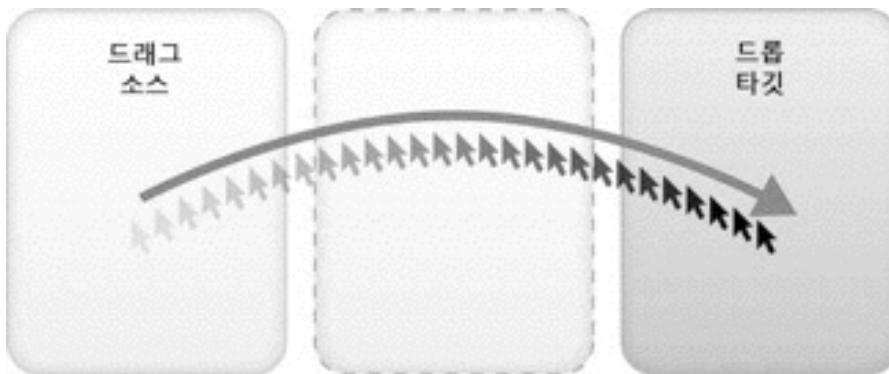
# Canvas API

```
function startDrawing() {  
    drawing = true;  
    context.beginPath();  
    context.strokeStyle = drawColor;  
    context.lineWidth = 1;  
    context.arc(event.clientX - rect.left, event.clientY - rect.top, 3, 0, 2*Math.PI);  
    context.stroke();  
    context.fillStyle = drawColor;  
    context.fill();  
    context.closePath();  
    context.beginPath();  
    context.moveTo(event.clientX - rect.left, event.clientY - rect.top);  
    context.lineCap = "round";  
    context.lineWidth = 6;  
}  
function keepDrawing() {  
    if (drawing) {  
        var x,y;  
        if (device == "mobileDevice") {  
            x = event.targetTouches[0].pageX;  
            y = event.targetTouches[0].pageY;  
        }  
        else {  
            x = event.clientX;  
            y = event.clientY;  
        }  
        context.lineTo(x - rect.left, y - rect.top);  
        context.stroke();  
    }  
}  
  
function stopDrawing() {  
    if (drawing) {  
        context.stroke();  
        drawing = false;  
    }  
}
```

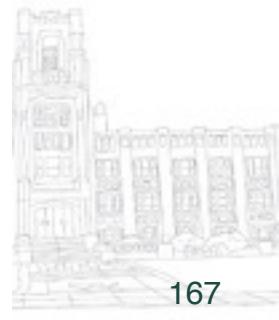
# Drag & Drop API

## □ Drag & Drop API 란?

- 웹 페이지상에서 원하는 항목을 드래그할 수 있게 해주는 API
- 드래그 소스(drag source)
  - 드래그를 시작하는 아이템이나 영역
- 드롭 타깃(drop target)
  - 버튼을 떼면서 포인터를 내려놓고 동작이 완성되었을 때 최종적으로 선택되는 영역이나 아이템



- 데이터 전송(data transfer)
  - 드래그 앤 드롭 동작을 위한 드래그 데이터 저장소를 보여주기 위해 사용되는 객체



# 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

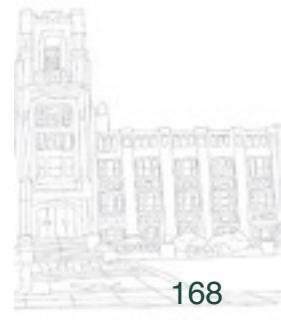
- 드래그된 엘리먼트가 dropzone 영역에서 움직일 때 발생하는 이벤트
- 드래그 소스에서 호출되는 drag 이벤트와는 달리 이 이벤트는 마우스의 현재 타깃에서 호출된다.

### ■ 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

## □ 드래그 기능 제어

- 특정 엘리먼트가 드래그된다는 사실을 나타내려면 `draggable`이라는 어트리뷰트를 추가하고 드래그 가 가능하도록 하기 위해서 `dragable` 속성을 `true`로 설정하면 된다.
- 예) `<div id='dragsource' draggable='true'>`

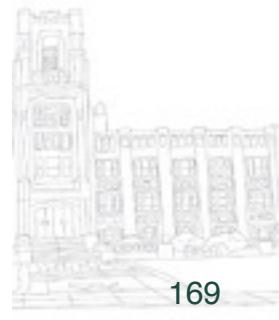
## □ 전송 및 제어

### ▪ `dataTransfer` 객체

- 드래그 앤 드롭 이벤트가 일어날 때마다 사용
- 소스와 타깃 사이에 교섭이 벌어지는 동안 실제 드롭 데이터를 얻어내 설정하는 데 사용

### ▪ `dataTransfer` 객체 함수

- `setData(format, data)`
  - `draggable`의 `dragstart` 이벤트에서 `draggable`의 정보를 저장하기 위해 사용
  - `key/value` 쌍으로 MIME 타입과 데이터 저장
  - 텍스트인 경우 : `text`, URL이인 경우 : `url`
- `getData(format)`
  - `droppable`의 `drop` 이벤트에서 `draggable`의 정보를 꺼내기 위해 사용
  - 지정한 MIME 타입의 데이터 반환
- `clearData([format])`
  - 매개변수 없이 이 함수를 호출하면 등록된 모든 데이터가 지워진다.
  - 포맷 매개변수를 넣어서 이 함수를 호출하면 특정 사항만 삭제된다.



# Drag & Drop API

- **dataTransfer** 객체 속성
  - **types**
    - 현재 등록된 모든 포맷의 배열을 반환
  - **items**
    - 모든 아이템 목록과 관련 포맷을 함께 반환
  - **files**
    - 드롭 동작과 관련된 모든 파일을 반환
- 드래그를 하는 동안 일어나는 피드백을 바꾸기 위해 사용되는 함수
  - **setDragImage(element, x, y)**
    - 기존 이미지 엘리먼트를 드래그 이미지로 사용하라고 브라우저에 알려준다.
    - 이 이미지는 커서 옆에 나란히 표시되어 사용자에게 드래그 동작의 결과를 알려준다.
    - (x, y) 좌표가 제공된다면 그것은 마우스의 드롭 지점일 가능성이 높다.
  - **addElement(element)**
    - 주어진 페이지 엘리먼트에서 이 함수를 호출하면 그 엘리먼트를 드래그 피드백 이미지로 사용하라고 브라우저에 알려준다.
- 드래그 동작 유형
  - **effectAllowed**
    - none, copy, copyLink, copyMove, link, linkMove, move, all 중 하나로 설정하면 그 연산(들)만 사용자가 쓸 수 있다 고 브라우저에 알려준다.
    - copy가 설정되면 copy 동작만 사용이 가능하며 move나 link 동작은 사용이 제한된다.
  - **dropEffect**
    - 현재 실행 중이거나 특정 동작을 하도록 설정된 연산이 어떤 것인지 결정하는 데 사용



# Drag & Drop API

## ■ 데이터 전송에 쓰이는 MIME 타입

- text/plain
  - text에 대한 표준 MIME 타입
- image/png
  - png를 위한 표준 MIME타입
- image/jpg
  - jpg를 위한 표준 MIME타입
- image/gif
  - gif를 위한 표준 MIME타입

## □ dropzone 속성

### ■ 이벤트 핸들러를 코딩하지 않고도 엘리먼트가 드롭 동작을 할 수 있게 한다.

```
<div dropzone="copy s:text/plain s:text/html" ondrop="dropevent(event, this)">
```

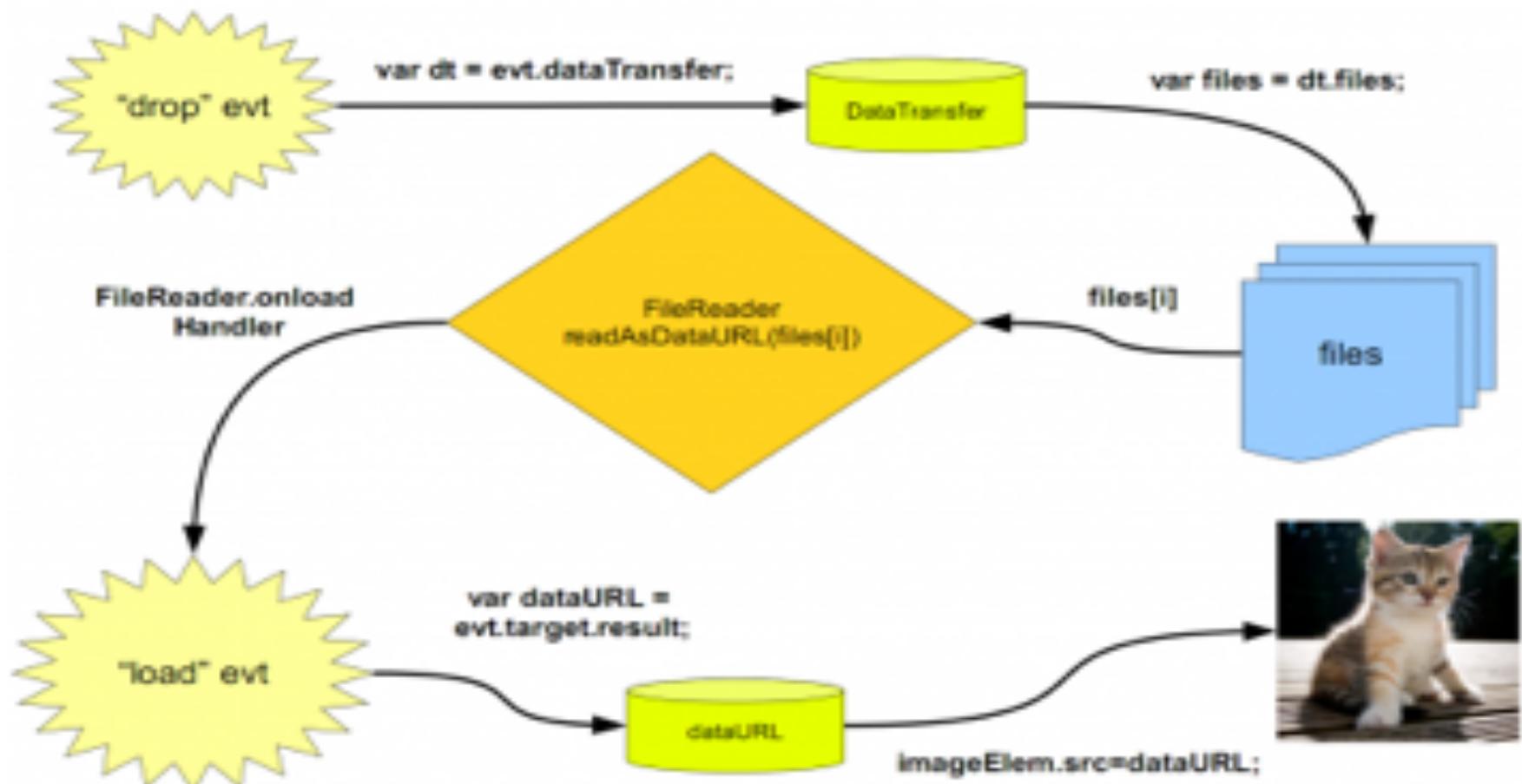
### ■ dropzone 어트리뷰트 기능

- copy, move, link
  - 세 가지 동작 중 하나만 사용가능
  - 아무 것도 지정되지 않으면 copy 동작이 실행
- s:<mime>
  - 문자 s: 뒤에 MIME 타입 기술
- f:<mime>
  - 문자 f: 뒤에 MIME 타입 기술



# Drag & Drop API

## 파일에 대한 Drag & Drop



# Drag & Drop API

- 예제 : card\_dnd.html



# Drag & Drop API

## [ HTML 소스 ]

```
<section id="dropbox">
  <canvas id="canvas" width="500" height="540"></canvas>
</section>
<section id="databox">
  
  
  
  
  
  
</section>
```

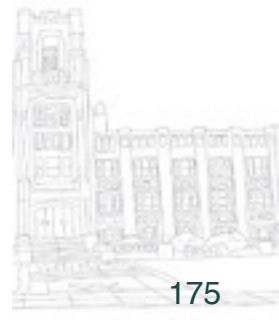
## [ JavaScript 소스 ]

```
var canvas, drop;
function initiate(){
  var images=document.querySelectorAll('#databox > img');
  for(var i=0; i<images.length; i++){
    images[i].addEventListener('dragstart', dragged, false);
  }
  drop=document.getElementById('canvas');
  canvas=drop.getContext('2d');

  drop.addEventListener('dragover', function(e){ e.preventDefault(); }, false);
  drop.addEventListener('drop', dropped, false);
}
```

# Drag & Drop API

```
function dragged(e){  
    elem=e.target;  
    e.dataTransfer.setData('Text', elem.getAttribute('id'));  
    e.dataTransfer.setDragImage(e.target, 0, 0);  
}  
function dropped(e){  
    e.preventDefault();  
    var id=e.dataTransfer.getData('Text');  
    var elem=document.getElementById(id);  
  
    var posx=e.pageX-drop.offsetLeft;  
    var posy=e.pageY-drop.offsetTop;  
  
    canvas.drawImage(elem,posx,posy, 200, 200);  
}  
window.addEventListener('load', initiate, false);
```



# Drag & Drop API

- 예제 : dnd\_fileread.html

## Drag & Drop 그리고 File I/O 예제

텍스트 파일을 끌어다 놓아보세요.

파일 선택 선택된 파일 없음      UTF-8 ▾      읽기

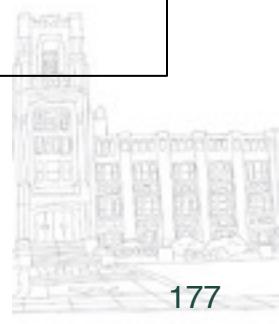
File Name	File Size



# Drag & Drop API

## [ HTML 소스 ]

```
<style type="text/css">
.hover {
    border: 10px solid #000;
    background-color: #bbb;
}
</style>
<section id="content">
    <h1>Drag & Drop 그리고 File I/O 예제</h1>
    <p>텍스트 파일을 끌어다 놓아보세요.</p>
    <article>
        <input id="file" type="file">
        <select id="encoding">
            <option>EUC-KR</option>
            <option>UTF-8</option>
        </select>
        <button id="read">읽기</button><br />
        <div>
            <span id="fileName">File Name</span>
            <span id="fileSize">File Size</span>
        </div>
        <textarea id="droparea" class="droparea" readonly style="width:600px; height:400px;"></textarea>
    </article>
</section>
```



# Drag & Drop API

## [ JavaScript 소스 ]

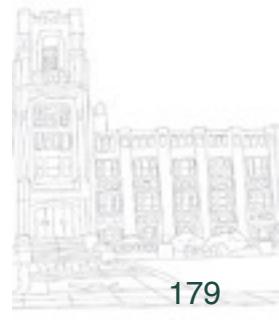
```
window.onload = function() {
    var readButton = document.getElementById("read");
    readButton.onclick = function(event) {
        readFile(document.getElementById("file").files[0]);
    }
    var target = document.getElementById("droparea");
    target.ondragover = function(event) {
        if(event.preventDefault) event.preventDefault();
    }
    target.ondragenter = function(event){
        target.classList.add('hover');
    }
    target.ondragleave = function(event){
        target.classList.remove('hover');
    }
    target.ondrop = function(event){
        if(event.preventDefault) event.preventDefault();
        target.classList.remove('hover');
        var filedata = event.dataTransfer.files[0];
        readFile(filedata);
    }
}
function readFile(file) {
    document.getElementById("fileName").textContent = file.name;
    document.getElementById("fileSize").textContent = "(" + file.size + "byte)";
    var reader = new FileReader();
```

# Drag & Drop API

```
reader.onload = function() {
    var display = document.getElementById("droparea");
    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);
};
```



# 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)

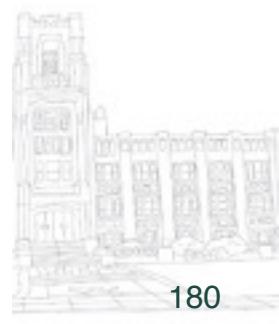
브라우저가 해당 미디어 타입을 재생할 수 있는지를 나타내는 문자열이다.

- play()

현재 위치에서 재생을 시작한다.

- pause()

오디오가 재생 중일 경우 일시 정지한다.



## Multimedia API

- 예제 : video\_play.html



# Multimedia API

## [ HTML 소스 ]

```
<video id="media" width="720" height="400">
  <source src="trailer.mp4">
  <source src="trailer.ogg">
</video>
<nav>
  <div id="buttons">
    <button type="button" id="play">재생</button>
  </div>
  <div id="bar">
    <div id="progress"></div>
  </div>
  <div style="clear: both"></div>
</nav>
```

## [ JavaScript 소스 ]

```
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);
}
```

# Multimedia API

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

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';
}
}

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);
}
}

window.addEventListener('load', initiate, false);
```

# File API

## □ File

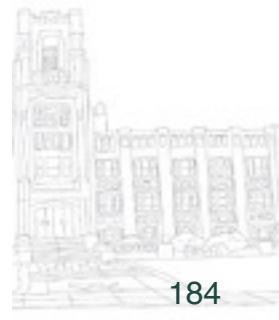
- 웹 브라우저에서 로컬 영역의 특정 파일을 접근, 조작할 수 있는 방법 제공
- 속성과 메서드
  - name : 파일 이름
  - type : 파일의 MIME Type (알 수 없을 때는 null)
  - size : 파일 크기
  - lastModifiedDate : 파일 마지막 수정일
  - slice(start, length) : 시작위치(start)부터 length만큼 지정하여 파일의 내용을 추출

## □ FileReader

- 파일의 내용을 읽는 기능 제공
- 메서드
  - readAsBinaryString(fileBlob)
    - 파일 내용을 읽어들여 바이너리 문자열로 설정
  - readAsText(fileBlob, encoding)
    - 파일 내용을 읽어들여 문자열로 설정
    - 두번째 인자에는 파일의 문자 인코딩을 지정할 수 있음(생략 시는 UTF-8)
  - readAsDataURL(file)
    - 파일 내용을 읽어들여 dataURL 형식의 문자열로 설정

### ■ 속성

- result : 읽어들인 파일의 내용
- error : 에러 발생 시의 에러 정보



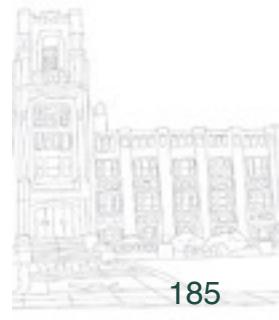
# File API

## ■ 이벤트

- **onload**
  - 읽기에 성공했을 때 호출
- **onerror**
  - 읽기 오류 발생시 호출

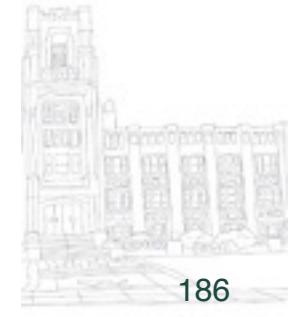
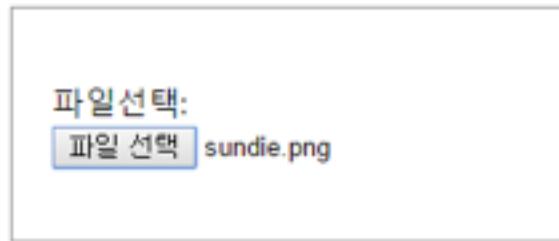
## ■ Error 상수

- **NOT\_FOUND\_ERR(1)**
  - 읽을 File 을 찾지 못할때
- **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
- **NOT\_READABLE\_ERR(4)**
  - 파일 접근 권한 문제와 같은 것으로 인해 File 을 읽지 못할때
- **ENCODING\_ERR(5)**
  - DataURL 로 표현될 수 있는 File 이나 Blob 을 구현한 제한된 곳의DataURL 에 대한 URL 길이 제한에 걸렸을 때



# File API

- 예제 : imgfile\_read.html



# File API

## [ HTML 소스 ]

```
<section id="formbox">
  <form name="form">
    <p>파일선택:<br><input type="file" name="myfile" id="myfile"></p>
  </form>
</section>
<section id="databox">
  여기에 이미지를 출력합니다.
</section>
```

## [ JavaScript 소스 ]

```
function initiate(){
  databox=document.getElementById('databox');
  var myfiles=document.getElementById('myfile');
  myfiles.addEventListener('change', process, false);
}
function show(e){
  var result=e.target.result;
  databox.innerHTML+='
```

# File API

```
function process(e){  
    var files=e.target.files;  
    databox.innerHTML="";  
    var file=files[0];  
    if(!file.type.match(/image.*\./)){  
        alert('이미지를 선택해 주세요.');//이미지 파일만 처리  
    }else{  
        databox.innerHTML+='Name: '+file.name+'<br>';  
        databox.innerHTML+='Size: '+file.size+' bytes<br>';  
  
        var reader=new FileReader();  
        reader.addEventListener('load', show, false); //reader.onload=show 와 동일  
        reader.readAsDataURL(file);  
    }  
}
```

# Web Storage API

## □ 쿠키와 웹 스토리지

### ■ 쿠키

#### • 개요

- 특정 홈페이지를 접속할 때 생성되는 정보를 담은 임시파일
- 로컬 값을 애플리케이션에 저장해서 다음에 페이지를 로딩할 때 다시 사용하는 용도로 사용

#### • 쿠키 단점

- 쿠키는 크기가 작다. 일반적으로 쿠키의 최대 크기는 4KB 정도로 더 큰 데이터는 저장할 수 없다.
- 같은 사이트에서 둘 이상의 탭을 열었을 때 둘 이상의 트랜잭션(transaction)을 추적하기 어렵다.
- 사이트간의 교차 스크립트 같은 기법을 통해 쿠키를 오용하고 결국 보안문제를 일으키기 쉽다.

### ■ 웹 스토리지

#### • 개요

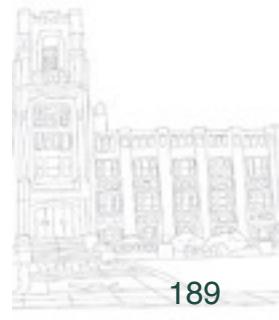
- 나중에 사용할 수 있도록 키/값 형태의 데이터 쌍을 세션이 끝날 때까지(세션 스토리지), 또는 세션 후에도(로컬 스토리지) 저장해 두는 것

#### • 웹 스토리지의 특징

- 쿠키처럼 웹 사이트를 돌아보고 난 후, 브라우저 탭을 닫고 난 후, 또는 브라우저를 빠져나온 후에 저장된다.
- 쿠키와 다른 점은 웹 스토리지의 데이터들은 사용자가 일부러 보내지 않는 한 웹 서버로 전송되지 않는다.

#### • 웹 스토리지 종류

- 로컬 스토리지
- 세션 스토리지



# Web Storage API

## □ 세션(Session) 스토리지

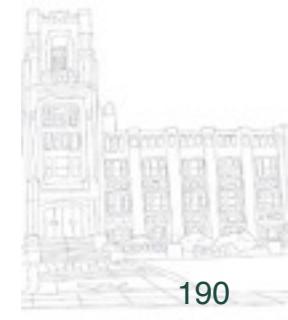
- 세션 동안 저장된 데이터를 저장하고 있다가 창(또는 탭)을 닫으면 모두 삭제된다.
- 관련 메서드
  - 데이터 저장 : `sessionStorage.setItem(key, value);`
  - 데이터 검색 : `sessionStorage.getItem(key)`
  - 데이터 삭제 : `sessionStorage.removeItem(key)`
  - 모든 데이터를 삭제 : `sessionStorage.clear();`

## □ 로컬(Local) 스토리지

- 세션이 끝나도 계속해서 데이터를 저장해야 할 때 사용한다.
- 데이터 저장, 검색, 삭제는 세션 스토리지와 동일

## □ 스토리지 이벤트

- 로컬 스토리지 영역이 바뀔 때마다 발생
- 속성
  - `storageArea` : 어떤 스토리지인지 알려준다.(세션인지 로컬인지)
  - `key` : 변경된 키
  - `oldValue` : 키의 이전 값
  - `newValue` : 키의 새로운 값
  - `url` : 키가 변경된 페이지의 URL



# Web Storage API

## ▣ 로컬(Local) 스토리지에 데이터를 저장, 읽기, 삭제하는 다양한 방법

- 저장

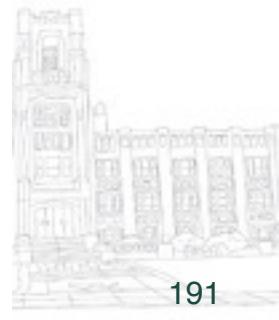
```
localStorage.mykey = "myvalue";  
localStorage["mykey"] = "myvalue";  
localStorage.setItem("mykey", "myvalue");
```

- 읽기

```
var mykey = localStorage.mykey;  
var mykey = localStorage["mykey"];  
var mykey = localStorage.getItem["mykey"];
```

- 삭제

```
delete localStorage.mykey;  
delete localStorage["mykey"];  
localStorage.removeItem("mykey");
```



# Web Storage API

## □ 세션(Session) 스토리지에 데이터를 저장, 읽기, 삭제하는 다양한 방법

- 저장

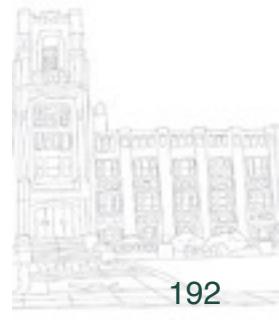
```
sessionStorage.mykey = "myvalue";  
sessionStorage["mykey"] = "myvalue";  
sessionStorage.setItem("mykey", "myvalue");
```

- 읽기

```
var mykey = sessionStorage.mykey;  
var mykey = sessionStorage["mykey"];  
var mykey = sessionStorage.getItem["mykey"];
```

- 삭제

```
delete sessionStorage.mykey;  
delete sessionStorage["mykey"];  
sessionStorage.removeItem("mykey");
```



# Web Storage API

- 예제 : memo\_storage.html

## 메모 편집

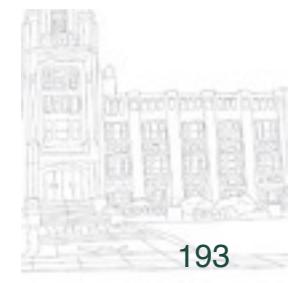
로컬스토리지를 테스트합니다.  
텍스트 입력시마다 로컬스토리지 영역에 보관합니다.

저장된 메모 비우기

메모내용이 자동 저장되었습니다.

The screenshot shows the Chrome DevTools Resources panel with the Local Storage section selected. A single item named 'memo' is listed, with its value being the text from the previous screenshot: "로컬스토리지를 테스트합니다. 텍스트 입력시마다 로컬스토리지 영역에 보관합니다."

Key	Value
memo	로컬스토리지를 테스트합니다. 텍스트 입력시마다 로컬스토리지 영역에 보관...



# Web Storage API

## [ HTML 소스 ]

```
<h2>메모 편집</h2>
<textarea id="txtBox" onKeyDown="saveText()" onKeyUp="saveText();"
          cols="100" rows="10">
</textarea>
<br>
<input type="button" value="저장된 메모 비우기" onClick="clr()" onKeyUp="clr();">
<p id="info" style="display: none;">메모내용이 자동 저장되었습니다.</p>
```

## [ JavaScript 소스 ]

```
function saveText() {
    info = document.getElementById("info");
    info.style.display = "block";
    localStorage.setItem("memo", msg.value);
};

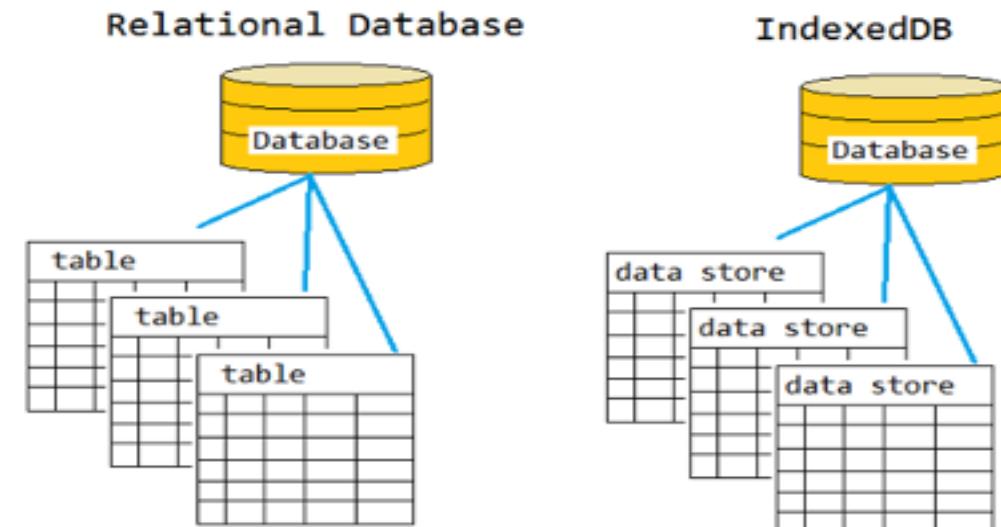
function pageload() {
    msg = document.getElementById("txtBox");
    msg.value = localStorage.getItem("memo");
};

function clr() {
    msg.value = "";
    localStorage.removeItem("memo");
    info.style.display = "none";
};
```

# Indexed DB API

## IndexedDB의 특징

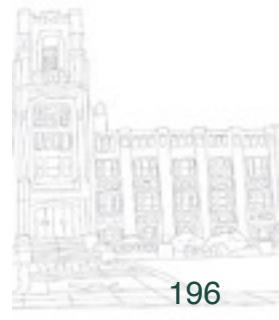
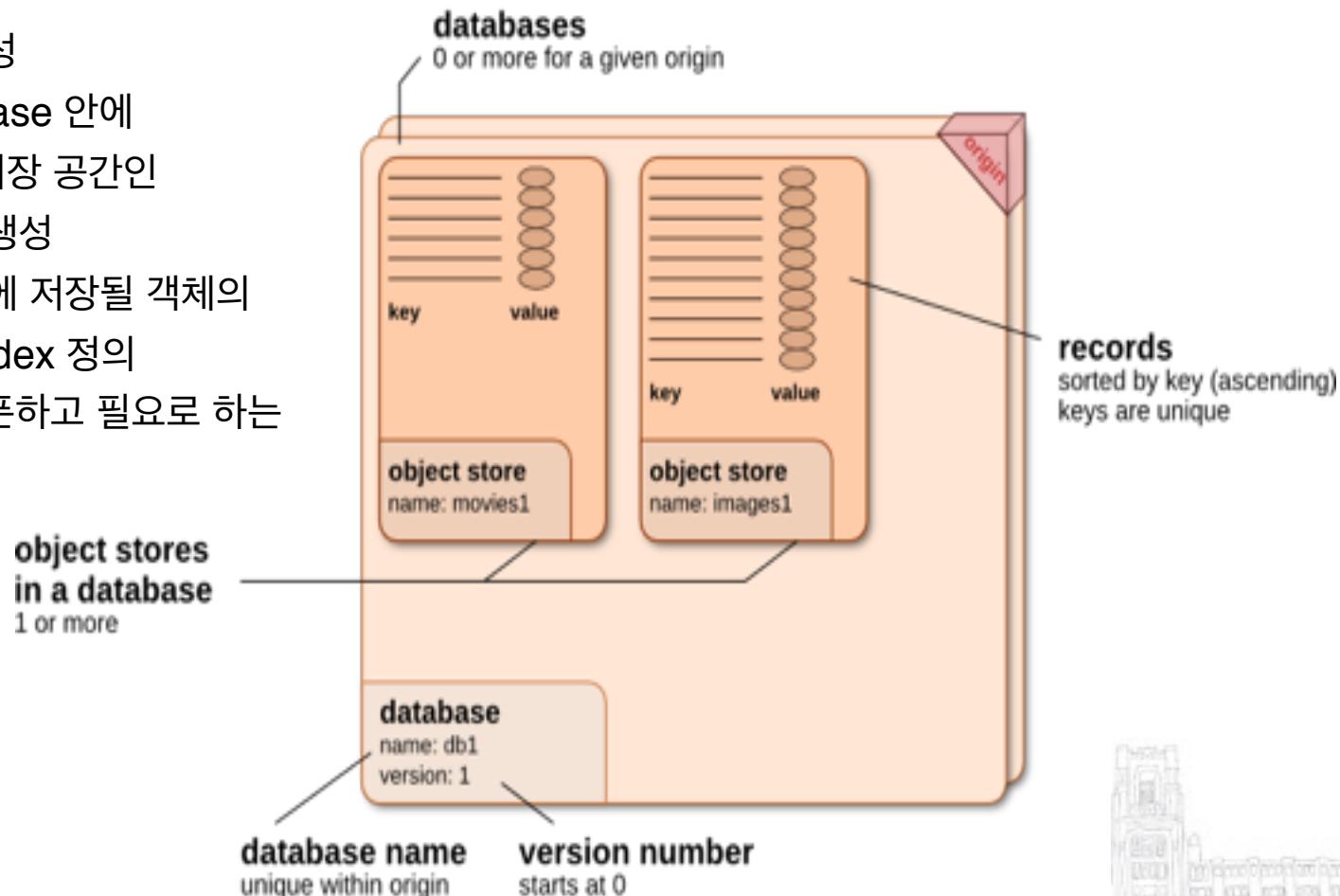
- key, value 형식의 데이터들을 구조화 시켜 보관하고 처리할 수 있다.
- 쿠키나 웹 스토리지와는 달리 JavaScript 객체를 구조적으로 보관하는 것이 가능하다.
- key 또는 인덱스를 이용한 데이터 저장과 검색 기능을 제공하며 데이터 검색시 뛰어난 성능을 지원한다.
- 트랜잭션 데이터베이스 모델로서 IndexedDB에서 일어나는 모든 CRUD 는 트랜잭션 내부에서 실행되어야 한다.
- 일반적으로 사용되고 있는 IndexedDB API 는 비동기이며 SQL을 사용하지 않는다. 각각의 이벤트 발생에 대한 콜백 함수를 구현하여 처리한다.
- Same Origin Policy 정책이 적용된다.



# Indexed DB API

## □ 구현 과정

1. DataBase 생성
2. 생성된 DataBase 안에  
객체(데이터) 저장 공간인  
ObjectStore 생성
3. ObjectStore 에 저장될 객체의  
사양에 따라 Index 정의
4. 트랜잭션을 오픈하고 필요로 하는  
CRUD 구현



# Indexed DB API

## □ DataBase 생성

- IDBFactory 타입인 window.indexedDB 객체를 사용한다.

- 데이터베이스와 커넥션을 처리하는 객체

- open(string name) : IDBOpenDBRequest

- open(string name, long version) : IDBOpenDBRequest

- deleteDatabase(string name) : IDBOpenDBRequest

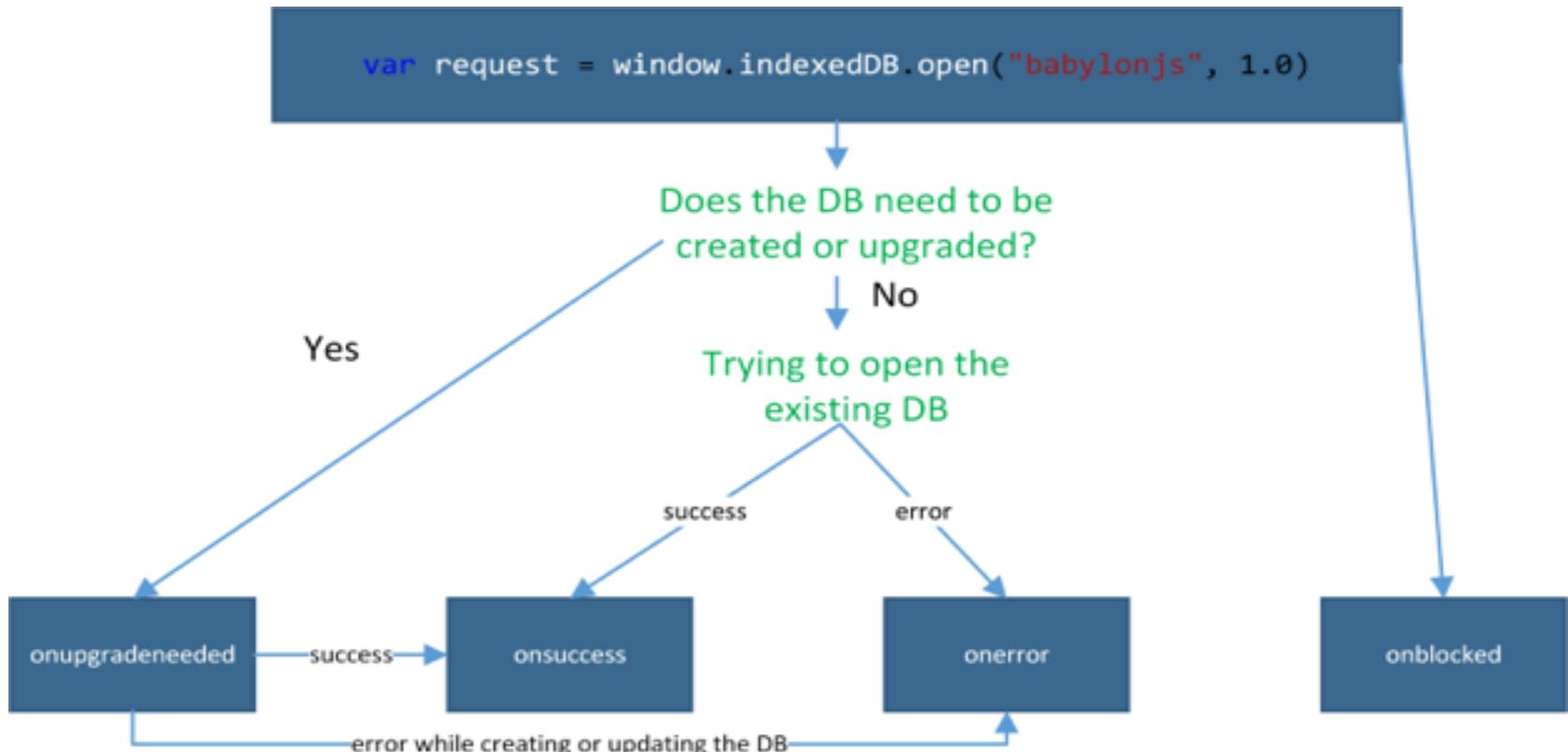
- open() 메서드는 비동기로 데이터베이스와의 커넥션을 생성하는 API로서 동일한 이름의 데이터 베이스가 있으면 데이터베이스와 커넥션을 연결하고, 없으면 주어진 이름으로 데이터베이스를 생성하고 success 이벤트 실행한다.
    - deleteDatabase() 메서드는 비동기로 데이터베이스를 삭제한다.  
데이터베이스와의 커넥션이 열려있으면 blocked 이벤트를 실행하고 커넥션이 닫힌 이후에 삭제한다.(IDBDatabase.close())
    - 모든 메서드들은 IDBOpenDBRequest 객체를 리턴한다. 이 객체에 success 이벤트와 error 이벤트 등에 대한 핸들러를 구현하여 기능을 처리한다.

```
var db;  
var request = indexedDB.open("MyTestDatabase");  
request.onerror = function(event) {  
    alert("Why didn't you allow my web app to use IndexedDB?!");  
};  
request.onsuccess = function(event) {  
    db = request.result;  
};
```

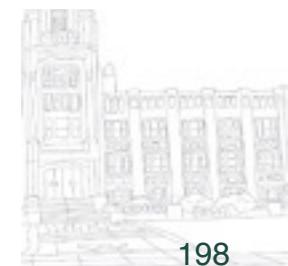


# Indexed DB API

## DataBase의 생성 또는 버전업 과정



```
request.onupgradeneeded = function(event) {  
    // Update object stores and indices ....  
};
```



# Indexed DB API

## □ ObjectStore 생성

- IDBDatabase 객체를 사용한다.

- 데이터베이스와의 커넥션이 성공적으로 이루어지면 success 이벤트를 통해 추출 가능한 객체이다.
- ObjectStore 객체의 생성과 삭제 기능의 메서드를 제공한다.
- 데이터베이스와의 트랜잭션을 시작하는 메서드를 제공한다.

`createObjectStore(string name, object optionalParameters) : IDBObjectStore`

optionalParameters : string keyPath, boolean autoIncrement  
(autoincrement : 레코드가 객체 저장소에 추가될 때마다 value 값을  
자동 증가시킨다.)

`deleteObjectStore(string name) : IDBRequest`

`setVersion(string version) : IDBVersionChangeRequest ----- deprecated`

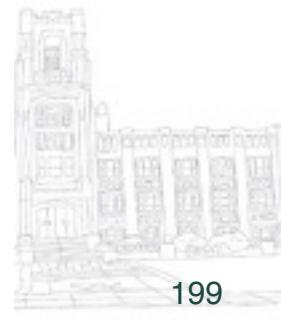
`transaction(any storeName, short mode) : IDBTransaction`

`close() : void`

`name(readonly string)`

`version(readonly long)`

`objectNames(readonly stringList)`



# Indexed DB API

## □ 인덱스 생성과 트랜잭션 생성

### ■ 인덱스 생성

- 객체 저장소에 인덱스를 생성한다.
- versionchange 트랜잭션(upgradeneeded 이벤트 발생) 내에서만 호출될 수 있다.

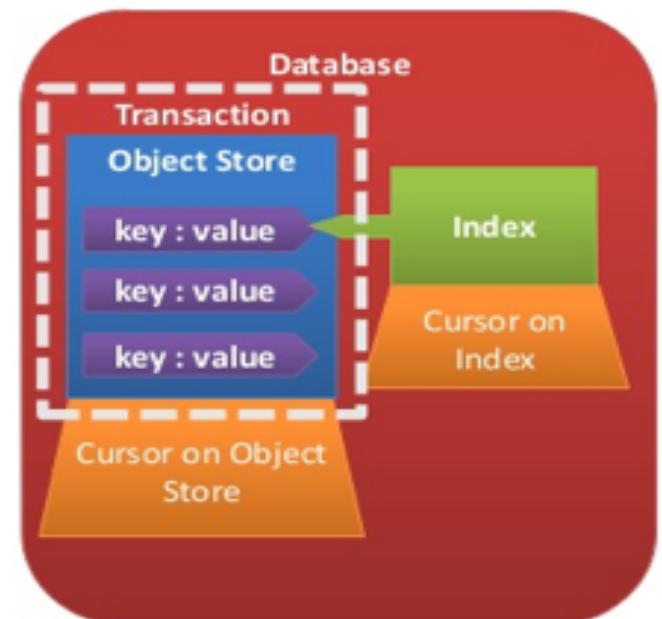
IDBObjectStore.createIndex(string name, string keyPath, object optionalParameters) : IDBIndex

### ■ 트랜잭션 생성

- IDBTransaction transaction ((DOMString or sequence<DOMString>)
- IDBTransaction 는 데이터베이스의 비동기 트랜잭션 인터페이스이다.
- 데이터를 읽고 쓰는(CRUD) 모든 작업은 트랜잭션내에서 수행된다.

```
var tranx = database.transaction(objStoreName, "readonly");
var objectStore = tranx.objectStore(objStoreName);
var request = objectStore.get(key)
```

```
var tranx = database.transaction(objStoreName, txMode2);
var objectStore = tranx.objectStore(objStoreName);
var data = {
    title : q_title,
    price : q_price,
    desc : q_desc
}
var request = objectStore.add(data);
```



# Indexed DB API

## □ DataBase의 생성 또는 버전업 과정

```
var request = indexedDB.open(dbName, dbVersion);
request.onsuccess = function(event) {
    database = this.result;
    document.getElementById("output").textContent = "bookDB 오픈 완료";
};

request.onerror = function(event) {
    console.log(event)
    alert('bookDB 오픈 중 에러 발생');
};

request.onupgradeneeded = function(event) {
    var objectStoreMember = event.currentTarget.result.createObjectStore(
        'book', {keyPath:'bookNum', autoIncrement:true});
    objectStoreMember.createIndex('titleIndex', 'title', {unique:true});
    objectStoreMember.createIndex('priceIndex', 'price', {unique:false});
    for(var i=0, len=bookData.length; i<len; i++) {
        objectStoreMember.add(bookData[i]);
    }
    alert('객체 저장소(book)를 생성하고 데이터 초기화 완료');
};
```

# Indexed DB API

## □ KeyRange 값

- 인덱스를 이용하면 인덱스를 생성한 속성에 대해 범위를 지정해서 빠르게 검색할 수 있다. 범위를 지정하려면 IDBKeyRange를 이용한다.

[ IDBKeyRange 의 주요 메서드 ]

- `only(value)`

`value`만을 포함한 범위를 생성한다

- `lowerBound(value, [, open])`

`value`를 최소값으로 하는 범위를 생성한다 (`open`에 `true`를 지정하면 `value`는 범위에 포함되지 않는다)

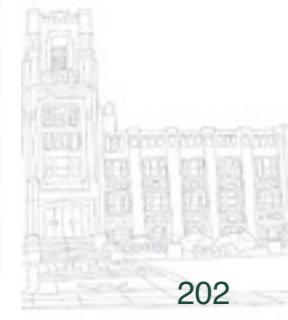
- `upperBound(value, [, open])`

`value`를 최댓값으로 하는 범위를 생성한다 (`open`에 `true`를 지정하면 `value`는 범위에 포함되지 않는다)

- `bound(lower, upper, [, lowerOpen, upperOpen])`

`lower`를 최소값, `upper`를 최댓값으로 하는 범위를 생성한다 (`lowerOpen, upperOpen`으로 각각 경계값을 범위에 포함하지 않도록 지정하거나)

Range	Code
<code>Value ≤ x</code>	<code>IDBKeyRange.upperBound(x)</code>
<code>Value &lt; x</code>	<code>IDBKeyRange.upperBound(x, true)</code>
<code>Value ≥ y</code>	<code>IDBKeyRange.lowerBound(y)</code>
<code>Value &gt; y</code>	<code>IDBKeyRange.lowerBound(y, true)</code>
<code>y ≤ Value ≤ x</code>	<code>IDBKeyRange.bound(y, x)</code>
<code>y &lt; Value ≤ x</code>	<code>IDBKeyRange.bound(y, x, true)</code>
<code>y ≤ Value &lt; x</code>	<code>IDBKeyRange.bound(y, x, false, true)</code>
<code>y &lt; Value &lt; x</code>	<code>IDBKeyRange.bound(y, x, true, true)</code>
<code>Value = z</code>	<code>IDBKeyRange.only(z)</code>



# Indexed DB API

- 예제 : crud\_db.html

## IndexedDB CRUD

bookDB 생성과 초기화

bookDB 삭제

검색할 key입력

데이터 추출(key 사용)

도서명

가격

간단설명

데이터 삽입

삭제하려는도서명

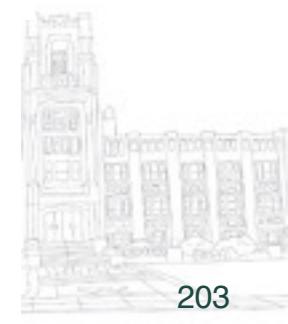
데이터 삭제

추출하려는도서명

데이터 추출(index 사용)

전체 도서 추출

30,000원 미만의 도서 금액높은순 추출



# Indexed DB API

## [ HTML 소스 ]

```
<h2>IndexedDB CRUD</h2>
<button onclick="creatIdxDB();">bookDB 생성과 초기화</button><br>
<button onclick="deleteIdxDB();">bookDB 삭제</button><br>
<input type="text" id="keyValue" placeholder="검색할 key입력">
<button onclick="getDataWithKey();">데이터 추출(key사용)</button><br>
<input type="text" id="title" placeholder="도서명">
<input type="text" id="price" size="10" placeholder="가격">
<input type="text" id="desc" size="30" placeholder="간단설명">
<button onclick="addData();">데이터 삽입</button><br>
<input type="text" id="deleteNameValue" placeholder="삭제하려는도서명">
<button onclick="deleteData();">데이터 삭제</button><br>
<input type="text" id="nameValue" placeholder="추출하려는도서명">
<button onclick="getDataWithBookName();">데이터 추출(index사용)</button><br>
<button onclick="listData();">전체 도서 추출</button><br>
<button onclick="listPriceData();">30,000원 미만의 도서 금액높은순 추출</button><br>
<hr>
<div id="output"></div>
```

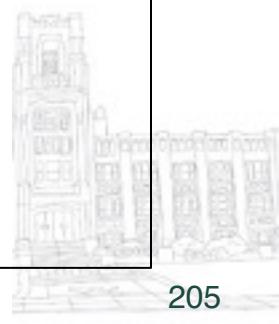
## [ JavaScript 소스 ]

```
var dbName = 'bookDB';
var dbVersion = 1;
var objStoreName = 'book';
var txMode1 = 'readonly';
var txMode2 = 'readwrite';
var database;
```

# Indexed DB API

## [ JavaScript 소스 ]

```
var bookData = [
    {title : 'HTML5 프로그래밍', price : 25000, desc : "HTML5의 주요 태그와 API 설명"},
    {title : '자바의 정석', price : 32000, desc : "Java 프로그래밍 방법 설명"},
    {title : 'HTML5&CSS3사전', price : 20000, desc : "HTML5의 주요 태그와 CSS3 구현 방법 설명"},
    {title : 'Oracle SQL', price : 24000, desc : "Oracle 기반의 SQL 작성 방법 설명"}
];
var createIdxDB = function() {
    var request = indexedDB.open(dbName, dbVersion);
    request.onsuccess = function(event) {
        database = this.result;
        document.getElementById("output").textContent = "bookDB 오픈 완료";
    };
    request.onerror = function(event) {
        console.log(event)
        alert('bookDB 오픈 중 에러 발생');
    };
    request.onupgradeneeded = function(event) {
        var objectStoreMember = event.currentTarget.result.createObjectStore('book', {keyPath:'bookNum',
autoIncrement:true});
        objectStoreMember.createIndex('titleIndex', 'title', {unique:true});
        objectStoreMember.createIndex('priceIndex', 'price', {unique:false});
        for(var i=0, len=bookData.length; i<len; i++) {
            objectStoreMember.add(bookData[i]);
        }
        alert('객체 저장소(book)를 생성하고 데이터 초기화 완료');
    };
};
```



# Indexed DB API

```
var deleteIdxDB = function() {
    var request = indexedDB.deleteDatabase(dbName);
    request.onsuccess = function(event) {
        alert('bookDB 삭제 완료');
    };
    request.onerror = function(event) {
        alert('bookDB 삭제 중 에러 발생');
    };
    request.onblocked = function() {
        alert('DB가오픈된 상태이므로 Blocked된 상태임 페이지 DB가 닫힐 때 삭제됨!!')
    }
};

var addData = function() {
    var q_title = document.getElementById("title").value;
    var q_price = document.getElementById("price").value;
    var q_desc = document.getElementById("desc").value;
    if(database && q_title && q_price && q_desc) {
        var tranx = database.transaction(objStoreName, txMode2);
        var objectStore = tranx.objectStore(objStoreName);
        var data = {
            title : q_title,
            price : q_price,
            desc : q_desc
        }
        var request = objectStore.add(data);
        request.onsuccess = function(event) {
            document.getElementById("output").textContent = "데이터 삽입 성공";
        };
    }
};
```

# Indexed DB API

```
request.onerror = function(event) {
    document.getElementById("output").textContent = "데이터 삽입 실패";
};

} else { alert('bookDB 오픈과 삽입될 데이터 입력은 필수입니다.'); }

};

var deleteData = function() {
    var q_name = document.getElementById("deleteNameValue").value;
    if(database && q_name) {
        var tranx = database.transaction(objStoreName, txMode2);
        var objectStore = tranx.objectStore(objStoreName);
        var index = objectStore.index("titleIndex");
        var getrequest = index.openCursor(IDBKeyRange.only(q_name));
        var key;
        getrequest.onsuccess = function(event) {
            var cursor = event.target.result;
            if(cursor) {
                key = cursor.value.bookNum;
                var delrequest = objectStore.delete(key);
                delrequest.onsuccess = function(event) {
                    document.getElementById("output").textContent = "데이터 삭제 성공";
                };
                delrequest.onerror = function(event) {
                    document.getElementById("output").textContent = "데이터 삭제 실패";
                };
            }
        }
    } else {
        alert('bookDB 오픈과 삭제하려는 도서명 입력은 필수입니다.');
    }
};
```

# Indexed DB API

```
var getDataWithKey = function(event) {
    var key = document.getElementById("keyValue").value
    if(database && key) {
        var tranx = database.transaction(objStoreName, txMode2);
        var objectStore = tranx.objectStore(objStoreName);
        var request = objectStore.get(key)
        request.onsuccess = function(event) {
            if (this.result){
                document.getElementById("output").textContent = JSON.stringify(this.result);
            } else {
                alert('추출된 데이터가 없음');
            }
        };
    } else {
        alert('bookDB 오픈과 추출하려는 데이터의 키 입력은 필수입니다.');
    }
};

var getDataWithBookName = function() {
    var q_name = document.getElementById("nameValue").value;
    if(database && q_name) {
        var tranx = database.transaction(objStoreName, txMode1);
        var objectStore = tranx.objectStore(objStoreName);
        var index = objectStore.index("titleIndex");
        var request = index.openCursor(IDBKeyRange.only(q_name));
        request.onsuccess = function(event) {
            var cursor = event.target.result;
```

# Indexed DB API

```
if(cursor) {
    var row = '<table border="1"><tr><td>도서명</td><td>' + cursor.value.title + '</td></tr>';
    row += '<tr><td>가격</td><td>' + cursor.value.price + '원</td></tr>';
    row += '<tr><td>설명</td><td>' + cursor.value.desc + '</td></tr>';
    row += '</tr></table>';
    document.getElementById("output").innerHTML = row;
}
};

} else {
    alert('bookDB 오픈과 추출하려는 도서명 입력은 필수입니다.');
}
};

request.onerror = function(event) {
    document.getElementById("output").textContent = "데이터 삽입 실패";
};

} else {
    alert('bookDB 오픈과 삽입될 데이터 입력은 필수입니다.');
}
};
```



# Indexed DB API

```
var listData = function() {
    if(database) {
        var tranx = database.transaction(objStoreName, txMode1);
        var objectStore = tranx.objectStore(objStoreName);
        var request = objectStore.openCursor();
        var tablelist = "<table border='1'>";
        request.onsuccess = function(event) {
            var cursor = event.target.result;
            if(cursor) {
                var row = '<tr>';
                row += '<td>' + cursor.value.bookNum + '</td>';
                row += '<td>' + cursor.value.title + '</td>';
                row += '<td>' + cursor.value.price + '</td>';
                row += '<td>' + cursor.value.desc + '</td>';
                row += '</tr>';
                tablelist += row;
                cursor.continue();
            } else {
                tablelist += "</table>";
                document.getElementById("output").innerHTML = tablelist;
            }
        };
    } else {
        alert('bookDB를 오픈하세요.');
    }
};
```

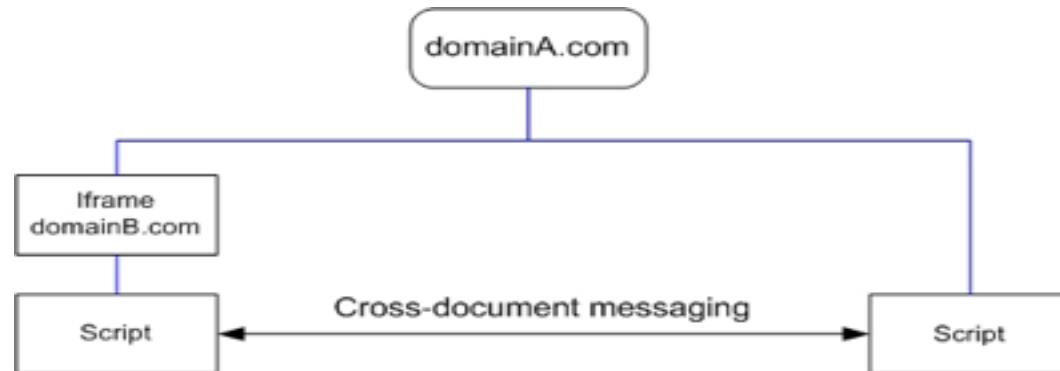
# Indexed DB API

```
var listPriceData = function() {
    if(database) {
        var tranx = database.transaction(objStoreName, txMode1);
        var objectStore = tranx.objectStore(objStoreName);
        var index = objectStore.index("priceIndex");
        var request = index.openCursor(IDBKeyRange.upperBound(30000, true), "prev");
        var tablelist = "<table border='1'>";
        request.onsuccess = function(event) {
            var cursor = event.target.result;
            if(cursor) {
                var row = '<tr>';
                row += '<td>' + cursor.value.bookNum + '</td>';
                row += '<td>' + cursor.value.title + '</td>';
                row += '<td>' + cursor.value.price + '</td>';
                row += '<td>' + cursor.value.desc + '</td>';
                row += '</tr>';
                tablelist += row;
                cursor.continue();
            } else {
                tablelist += "</table>";
                document.getElementById("output").innerHTML = tablelist;
            }
        };
    } else {
        alert('bookDB를 오픈하세요.');
    }
};
```

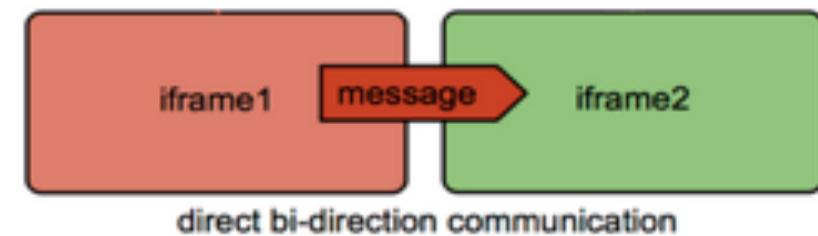
# Web Messaging API

## □ Web Messaging 이란?

- 실행중인 브라우저에서 iframe, 탭, 그리고 윈도우 사이의 통신은 보안상의 이유로 제한되어 있다. HTML5의 Messaging API는 다른 도메인에서 로드된 컨텐트들끼리 메시지를 주고 받는 것을 지원한다.



- 메시지 전송의 표준 방식은 `postMessage()` 메서드를 사용하는 것이다.
- 메시지 수신의 표준 방식은 `message` 이벤트의 핸들러를 구현하는 것이다.
- 두 개의 웹 페이지가 서로 데이터를 주고 받을 수 있는 기능으로 1 : 1 통신이다.
- SOP 가 적용되지 않으므로 서로 다른 도메인으로 부터 전달된 문서들 간의 통신이 가능하다.



# Web Messaging API

## ▣ 메시지 송신

- postMessage() 메서드를 이용하면 자바스크립트 컨텍스트 사이에 비동기 메시지 통신을 처리할 수 있다.
- postMessage() 메서드는 동일 도메인 문서 사이의 통신에도 사용되지만 서로 다른 도메인(cross document)을 갖는 문서 사이의 통신에도 사용 가능하다.
- 메시지를 보내는 쪽에서는 postMessage(data, [ports,] targetOrigin) 메서드를 사용. postMessage() 메서드의 각 파라미터는 다음과 같다.
  - data : 송신할 메시지(자바스크립트 객체)이다.
  - ports : 채널 메시지 전송 시 공유할 포트배열이며, 생략 가능하다.
  - targetOrigin : 수신처의 도메인이다.

```
document.getElementById(id).contentWindow.postMessage(msg, '*');
window.opener.postMessage(id, "*");
```

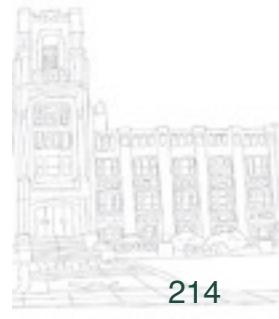


# Web Messaging API

## ▣ 메시지 수신

- 메시지 수신측에서는 메시지를 수신할 때마다 message 이벤트가 발생된다.
- message 이벤트의 핸들러 메서드에 인자로 넘어오는 MessageEvent 객체의 속성을 이용하여 수신된 메시지를 사용한다.
- MessageEvent 객체의 주요 속성
  - data 송신되는 메시지의 내용이 되는 데이터이다.
  - origin 메시지 송신처의 도메인입니다. 신뢰할 수 없는 도메인에서 온 메시지는 무시할 수 있다.
  - source 메시지를 송신하는 윈도우 객체이다.
  - ports 메시지 송신 시 지정한 포트의 복사본이다. Channel Messaging에서 사용한다.
  - lastEventId 마지막 이벤트 ID이다. Server-Sent Event에서 사용한다.
- 다음과 같이 message 이벤트에 대한 핸들러를 구현한다.

```
window.onmessage = function(me){  
    if(!me) me = window.event;  
        // 원하는 도메인에서만 오는 경우만 처리하고자 할 경우  
        if(me.origin == "http://xxx.yyy.zzz"){  
            document.getElementById("display").innerHTML = me.data;  
        };  
};
```



## Indexed DB API

- 예제 : comm\_msg.html



# Indexed DB API

## [ HTML 소스 ]

```
<iframe id="frame_1" src="sub1.html" style="width: 320px; height: 240px;"></iframe>
<iframe id="frame_2" src="sub2.html" style="width: 320px; height: 240px;"></iframe>
<br><br>
<button onclick="sendMessage('frame_1', 'main: hello?');">프레임1로 보내기</button>
<button onclick="sendMessage('frame_2', 'main: hello?');">프레임2로 보내기</button>
<br>
<div id="data" style="border: 1px solid black; width: 650px; height: 200px; overflow: auto;"></div>
```

## [ JavaScript 소스 ]

```
function sendMessage(id, msg) {
    document.getElementById(id).contentWindow.postMessage(msg, '*');
}
window.onmessage = function(e) {
    document.getElementById('data').innerHTML += e.data + '<br>';
}
```

# Indexed DB API

[ sub1.html ]

```
<script>
dow.onmessage = function(e) {
  document.getElementById('data').innerHTML += e.data + '<br />';
}
function sendMessage(isParent) {
  if (isParent) window.parent.postMessage('frame_1: hello?', '*');
  else
    window.parent.document.getElementById('frame_2').contentWindow.postMessage('frame_1: hello?', '*');
}
</script>
</head>
<body>
<button onclick="sendMessage(true);">부모창으로 보내기</button>
 
<button onclick="sendMessage(false);">frame_2로 보내기</button>
<br />
<div id="data" style="border: 1px solid red; width: 300px; height: 180px; overflow: auto;"></div>
```

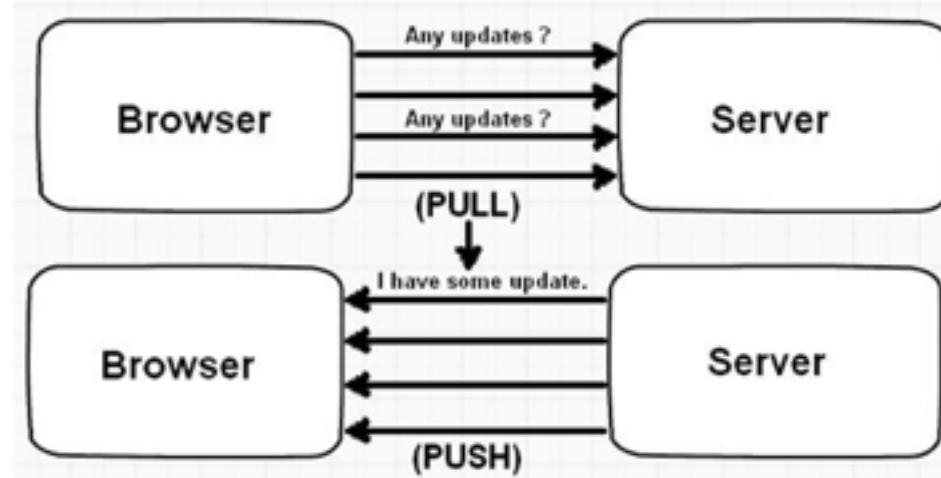
[ sub2.html ]

```
<script>
window.onmessage = function(e) {
  document.getElementById('data').innerHTML += e.data + '<br />';
}
function sendMessage(isParent) {
  if (isParent) window.parent.postMessage('frame_2: hello?', '*');
  else
    window.parent.document.getElementById('frame_1').contentWindow.postMessage('frame_2: hello?', '*');
}
</script>
</head>
<body>
<button onclick="sendMessage(true);">부모창으로 보내기</button>
 
<button onclick="sendMessage(false);">frame_1로 보내기</button>
<br />
<div id="data" style="border: 1px solid red; width: 300px; height: 180px; overflow: auto;"></div>
```

# Server Sent Events API

## □ Server-Sent Events란?

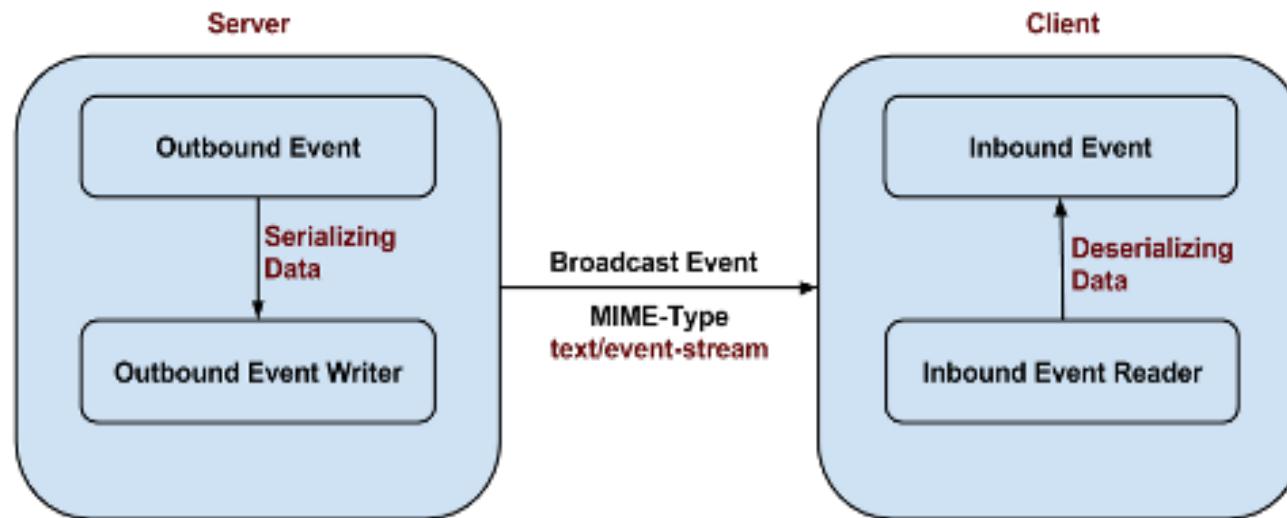
- 웹 환경에서 서버 푸시(Server Push)를 구현하기 위해 제안된 표준 기술
- 서버 푸시(Server Push)
  - 서버에서 클라이언트로 데이터를 능동적으로 전송해 주는 통신 방식
  - 관련 기술
    - 외부 플러그인 이용
    - Ajax 폴링(polling) 기법 사용 : 클라이언트의 주기적인 요청요구 필요
- SSE(Server-Sent Event)는 폴링과 유사하게 클라이언트에서 서버로 반복적으로 질의를 하는 방식으로 서버 푸시 효과를 내기 위한 기술임
- 사용 예) 페이스 북/트위터 업데이트, 주식 정보, 뉴스 Feeds 등



# Server Sent Events API

## □ EventSource 객체

- EventSource 객체를 통해 서버로의 접속과 주기적인 자동 요청이 처리되며 message 이벤트를 처리하여 서버로 부터 전달된 데이터를 받을 수 있다.  
예) var event\_source = new EventSource("push.jsp");
- EventSource 객체에서 발생 가능한 이벤트들
  - open : 서버와의 접속이 이루어질 때마다 발생되는 이벤트이다.
  - message : 서버로부터 데이터(이벤트의 종류가 지정되지 않은)가 전송되었을 때마다 발생되는 이벤트이다.
  - error : 오류 발생시 클라이언트에서는 발생되는 이벤트이다.
  - XXX : 서버로부터 데이터(XXX라는 이벤트와 함께 전달된)가 전송되었을 때마다 발생되는 이벤트이다.



# Server Sent Events API

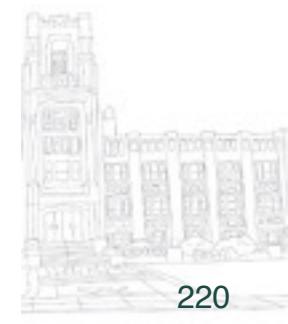
## □ 서버로 부터 전송되는 SSE 응답 형식

- 응답 데이터는 text/event-stream 라는 MIME 타입으로 제공되어야 한다.
- 응답 데이터의 문자 인코딩은 반드시 UTF-8 형식이어야 한다.
- 데이터 형식
  - 빈 행은 이벤트를 구분하는 역할 수행
  - retry : 반복주기(단위 : millisecond)
  - event : 클라이언트에 전달할 이벤트 이름 지정(디폴트 : message)
  - data : 서버가 저단한 실제 데이터
  - 예) **retry: 2000**

```
data: 데이터1_1  
data: 데이터1_2
```

```
event: first  
data: 데이터2_1  
data: 데이터2_2
```

```
event: second  
data: 데이터3_1  
data: 데이터3_2
```



# Server Sent Events API

- 예제 : stockdata\_push.html

[ HTML 소스 ]

```
<h1>Server-Sent Events 예제</h1>
<p>주가를 실시간으로 표시합니다.</p>
<table border="1">
  <tr>
    <th>A 회사</th><th>B 회사</th><th>C 회사</th>
  </tr>
  <tr>
    <td id="A">55</td>
    <td id="B">70</td>
    <td id="C">100</td>
  </tr>
</table>
```

## Server-Sent Events 예제

주가를 실시간으로 표시합니다.

A 회사	B 회사	C 회사
55	70	100

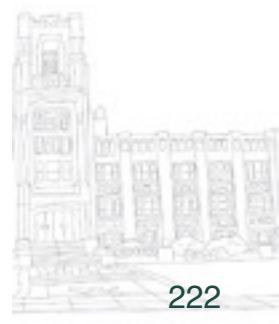
[ JavaScript 소스 ]

```
var source = new EventSource("datapush.jsp");
source.addEventListener("open", function(event) {
  console.log("서버와 접속되었습니다.");
}, false);
source.addEventListener("message", function(event) {
  console.log(event.data);
  var data = JSON.parse(event.data);
  var delta = parseInt(data.value, 10);
  var cell = document.getElementById(data.cname);
  var currentValue = parseInt(cell.textContent, 10);
  if (delta < 0) {
    cell.style.color = "red";
  } else {
    cell.style.color = "blue";
  }
  cell.textContent = currentValue + delta;
}, false);
```

# Server Sent Events API

## [ 서버 소스 ]

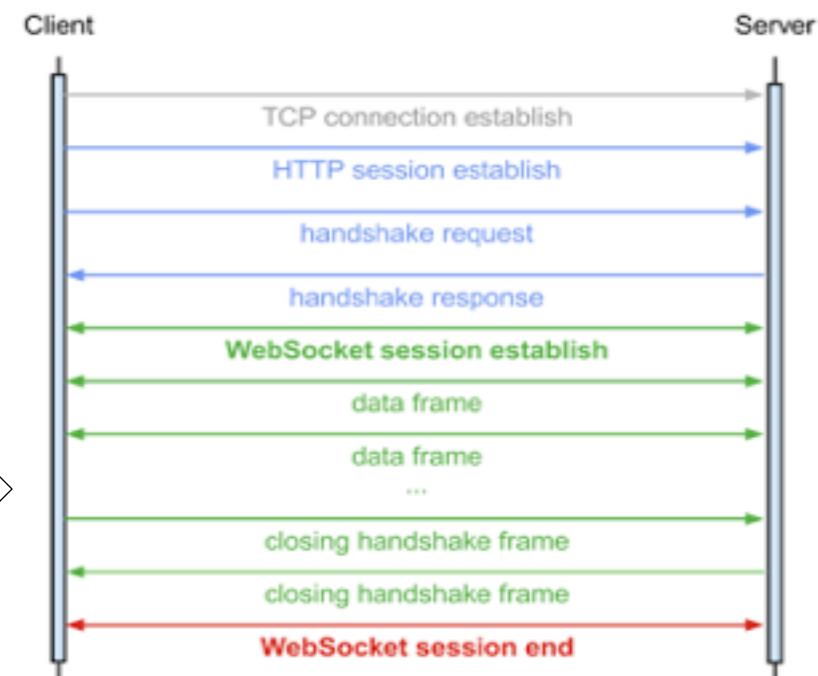
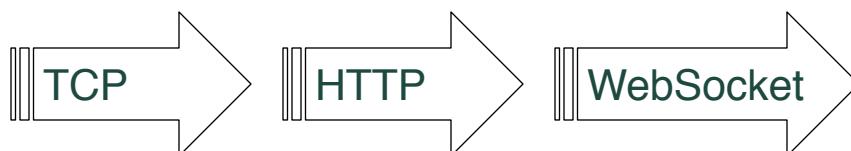
```
<%@page contentType="text/event-stream; charset=utf-8"%>
<%-- 재시도 간격: 1-5초 --%>
retry: <%= ((int)(Math.random()*5)+1)*1000 %>
<%
String[] symbols = new String[] { "A", "B", "C" };
for (String symbol : symbols) {
    int delta = (int) (Math.random() * 10);
    if (delta < 7 && delta > 0) {
        if (System.currentTimeMillis() % 2 == 0)
            delta = -delta;
    }
    data: {"cname" : "<%=symbol%>" , "value" : "<%=delta%>"}<%= "\n" %>
}
<%>
```



# Web Socket API

## □ 웹 소켓이란?

- 웹 소켓은 웹 서버와 웹 브라우저가 지속적으로 연결된 TCP 라인을 통해 실시간으로 데이터를 주고 받을 수 있도록 하는 HTML5의 새로운 사양이다.
- 웹 소켓을 이용하면 일반적인 TCP 소켓과 같이 연결지향 양방향 전이중 통신이 가능하다.
- WebSocket을 사용하면 더 이상 ActiveX를 사용하지 않고도 TCP/IP 소켓통신을 구현할 수 있으며 네트워크의 과부하를 줄이고 애플리케이션의 반응성을 높일 수 있게 된다.
- WebSocket은 HTTP의 단점을 보완하기 위해 등장하기는 했으나 WebSocket이 HTTP를 대체하는 것은 어려며 다만 HTTP가 적합치 않은 메세징, 애플리케이션 특성 상 트래픽이 높고 지연시간이 낮은 환경에서 유용하다.



# Web Socket API

## ■ 웹 소켓의 구현 방법

### ■ 서버연결

HTML5가 제공하는 WebSocket 객체를 통해 서버 연결을 수행한다. 일반 통신은 ws, 보안 통신은 wss 프로토콜을 이용한다

```
var ws = new WebSocket("ws://echo.websocket.org/echo");
```

### ■ 데이터 송신

WebSocket 객체의 send() 메서드로 데이터를 서버로 송신한다.

```
ws.send("웹소켓님 반가워요!")
```

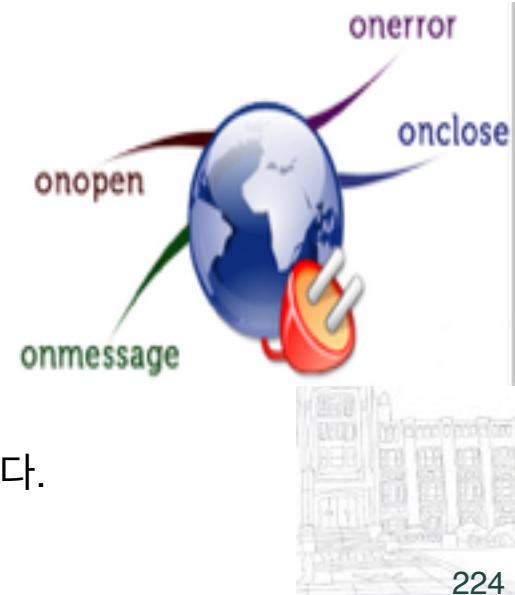
### ■ 데이터 수신

서버에서 전송되는 데이터를 받으려면 message 이벤트를 구현한다.

```
ws.onmessage = function(e) {  
    log("웹소켓 에코서버로 부터 메시지 도착: " + e.data);  
}
```

### ■ 웹 소켓 관련 이벤트

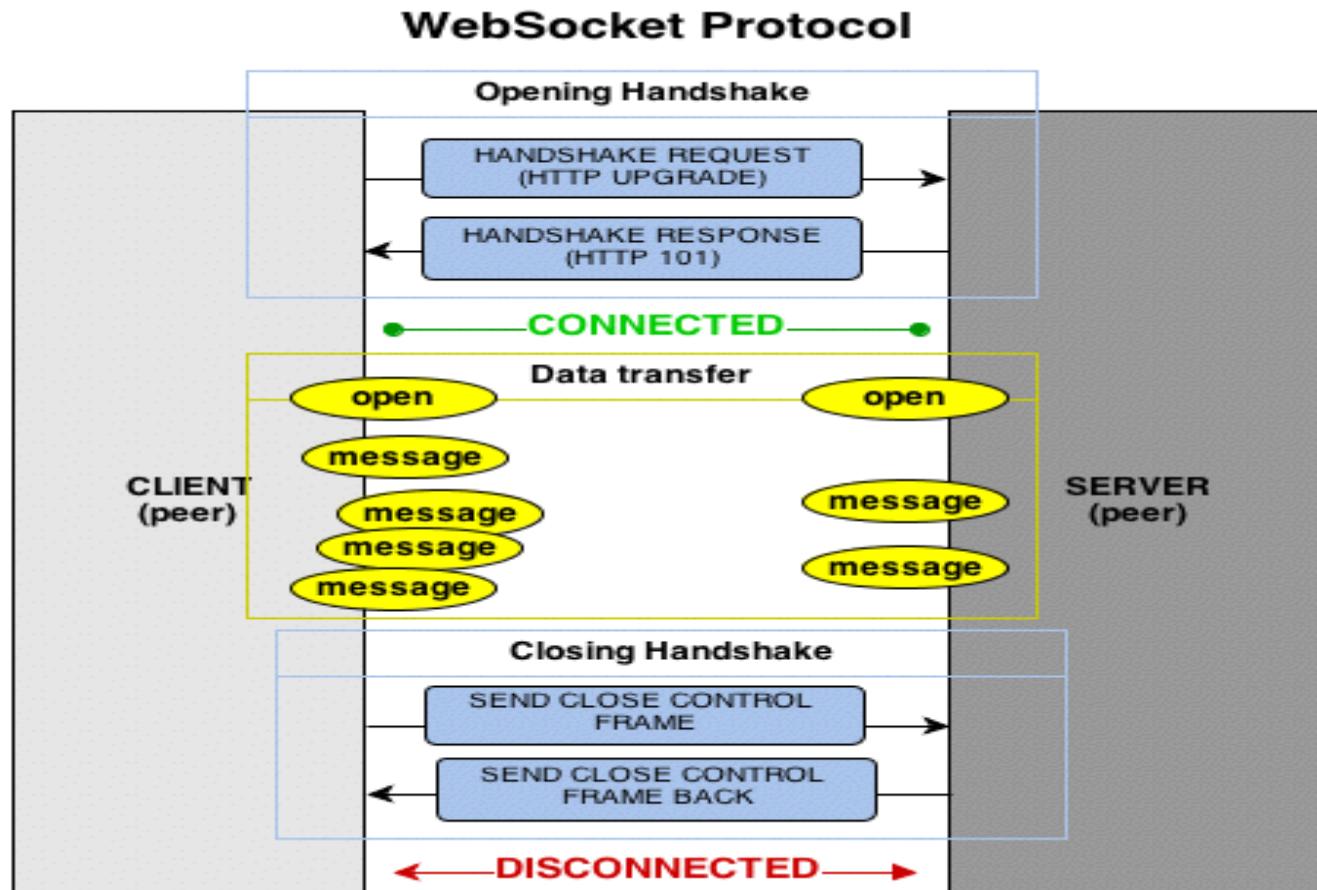
- open : 웹 소켓 서버와 접속이 일어나면 발생하는 이벤트이다.
- close : 웹 소켓 서버와 접속이 해제되면 발생되는 이벤트이다.
- message : 웹 소켓 서버로 부터 메시지가 전송되면 발생되는 이벤트이다.
- error : 오류가 발생되면 발생되는 이벤트이다.



# Web Socket API

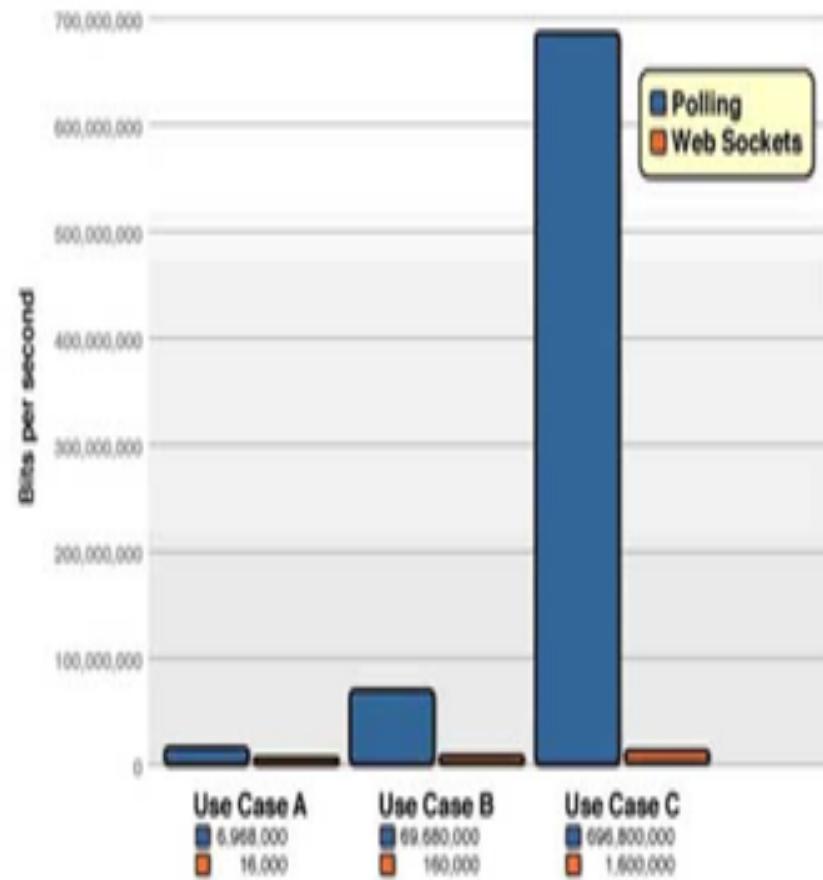
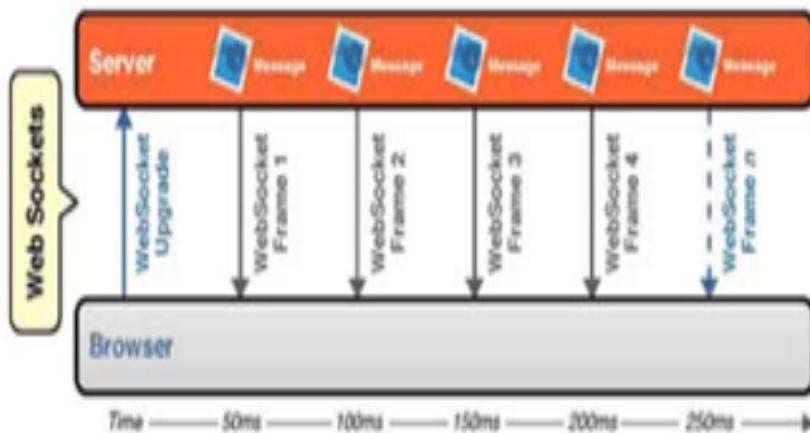
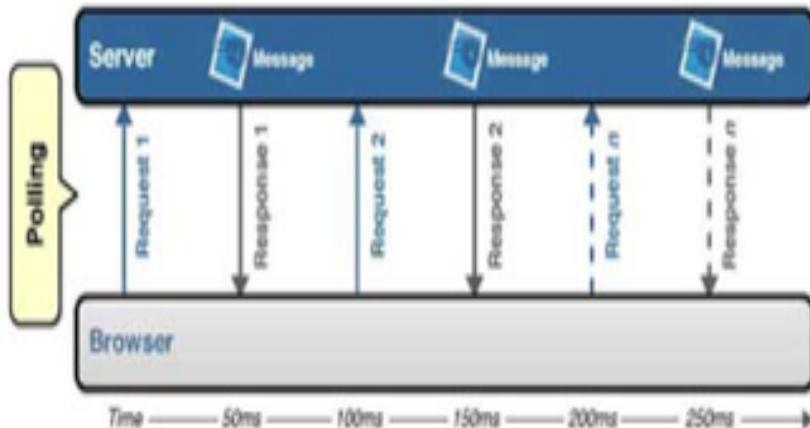
## ■ 웹 소켓 프로토콜의 통신과정

- TCP의 상위 레벨 프로토콜로서 IP 주소와 포트번호대신 ws: 로 시작하는 URL 문자열을 사용한다.
- HTTP 프로토콜로 요청된 다음 websocket 프로토콜로 프로토콜 스위칭이 발생한다.



# Web Socket API

## □ 웹 소켓과 폴링 기술의 비교



# Web Socket API

## □ 웹 소켓과 폴링 기술의 비교

### ■ 폴링(Polling) 방식:

요청/응답 헤더 데이터 용량: (871 Byte)

1. 1000명 \* 헤더 데이터 용량 = 871,000 Byte
2. 10000명 \* 헤더 데이터 용량 = 8,710,000 Byte
3. 100000명 \* 헤더 데이터 용량 = 87,100,000 Byte

### ■ WebSocket 방식:

메시지 데이터 용량: (2 Byte)

1. 1000명 \* 메시지 데이터 용량 = 2,000 Byte
2. 10000명 \* 메시지 데이터 용량 = 20,000 Byte
3. 100000명 \* 메시지 데이터 용량 = 200,000 Byte

## [ 테스트 결과 ]

폴링 방식 데이터(위의 경우 헤더 데이터만 있으며, 바디가 빠진상태)처리는 웹 소켓처리 방식보다 주고 받는 데이터양에 의해 상당한 네트워크 오버헤드가 발생한다.



# Web Socket API

- 예제 : comm\_websocket.html

## 메시지를 입력

multimessage 입력시 2초  
간격으로 서버 푸시가 10회 발생합니다.  
close 입력시 연결종료됩니다.

multimessage

전송

open 이벤트 발생 : 서버와 접속됨

message 이벤트 발생 : 환영합니다. 서블릿으로 구현한 웹소켓 서버입니다.

message 이벤트 발생 : [테스트합니다.] - 웹소켓 서버로 부터의 응답!!

message 이벤트 발생 : \*\*\* 2초에 한번씩 서버푸쉬를 10번 진행합니다. \*\*\*

message 이벤트 발생 : 1 - 웹소켓 서버로 부터의 응답!!

message 이벤트 발생 : 2 - 웹소켓 서버로 부터의 응답!!

message 이벤트 발생 : 3 - 웹소켓 서버로 부터의 응답!!

message 이벤트 발생 : 4 - 웹소켓 서버로 부터의 응답!!

message 이벤트 발생 : 5 - 웹소켓 서버로 부터의 응답!!

message 이벤트 발생 : 6 - 웹소켓 서버로 부터의 응답!!

message 이벤트 발생 : 7 - 웹소켓 서버로 부터의 응답!!

message 이벤트 발생 : 8 - 웹소켓 서버로 부터의 응답!!

message 이벤트 발생 : 9 - 웹소켓 서버로 부터의 응답!!

message 이벤트 발생 : 10 - 웹소켓 서버로 부터의 응답!!

# Web Socket API

## [ HTML 소스 ]

```
<section id="formbox">
  <form name="form">
    <h2>메시지를 입력</h2>
    multimessage 입력시 2초 <br>간격으로 서버 푸시가 10회 발생합니다.<br>
    close 입력시 연결종료됩니다.<br>
    <input type="text" name="command" id="command"></p>
    <p><input type="button" name="button" id="button" value="전송"></p>
  </form>
</section>
<section id="databox"></section>
```

## [ JavaScript 소스 ]

```
function initiate(){
  databox=document.getElementById('databox');
  var button=document.getElementById('button');
  button.addEventListener('click', send, false);
  socket=new WebSocket("ws://localhost:8080/edu/TheWebSocketServlet");
  socket.addEventListener('open', opened, false);
  socket.addEventListener('message', received, false);
  socket.addEventListener('close', closed, false);
  socket.addEventListener('error', error, false);
}
```

# Web Socket API

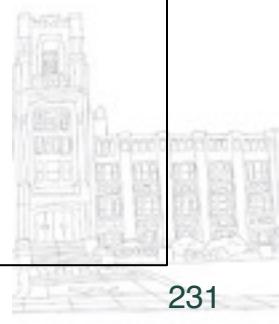
```
function opened(){
    databox.innerHTML+='open 이벤트 발생 : 서버와 접속됨<br>';
}
function received(e){
    databox.innerHTML+='message 이벤트 발생 : '+e.data+'<br>';
}
function closed(){
    databox.innerHTML+='close 이벤트 발생 : 서버와 접속 해제됨<br>';
    var button=document.getElementById('button');
    button.disabled=true;
}
function error(event){
    databox.innerHTML+='error 이벤트 발생<br>';
}
function send(){
    var command=document.getElementById('command').value;
    if(command=='close'){
        socket.close();
    }else{
        socket.send(command);
    }
}
window.addEventListener('load', initiate, false);
```



# Web Socket API

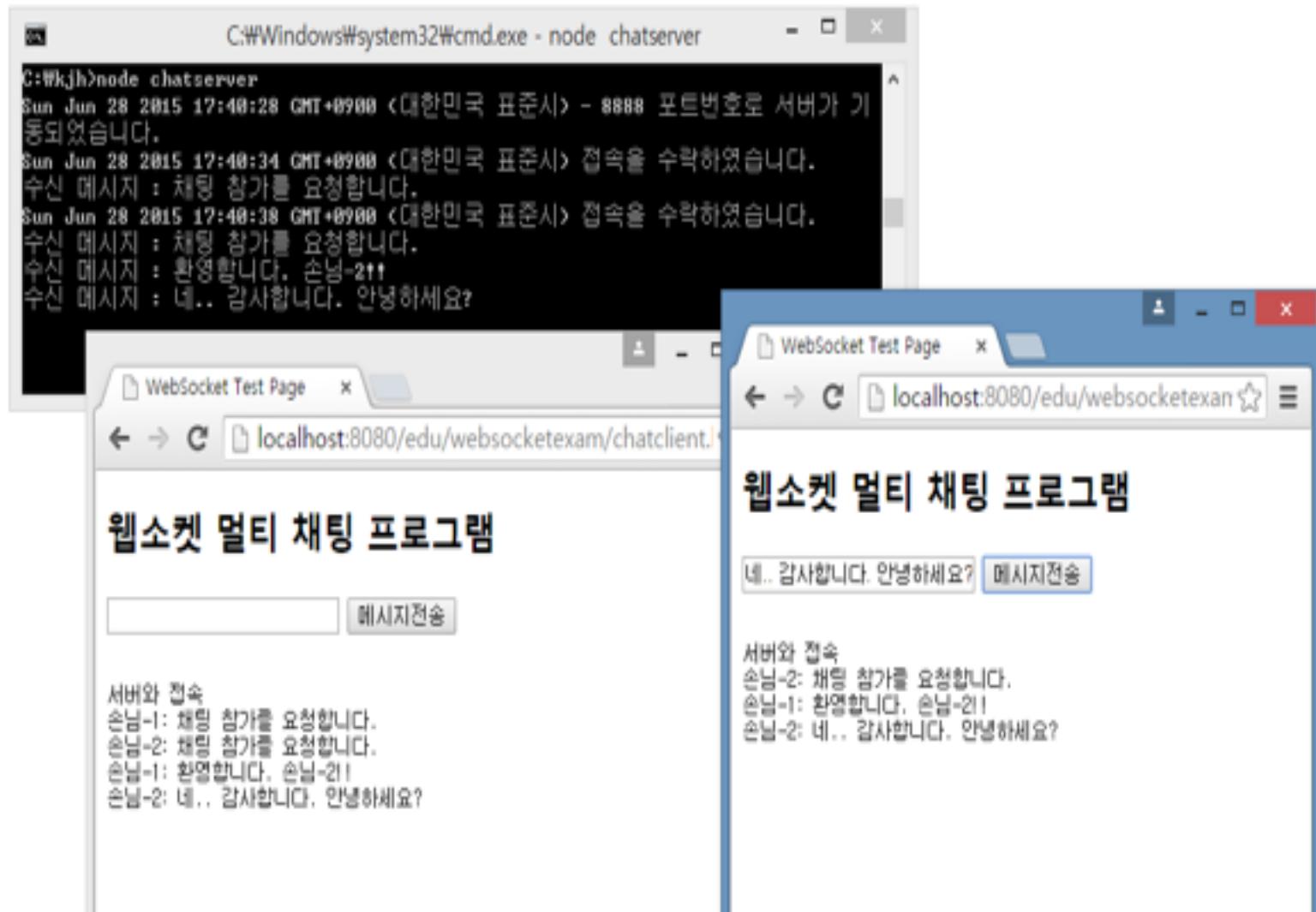
## [ 웹소켓 서블릿 소스 ]

```
@ServerEndpoint(value = "/TheWebSocketServlet")
public class TheWebSocketServlet {
    @OnOpen
    public void onOpen(Session session) {
        try {
            session.getBasicRemote().sendText("환영합니다. 서블릿으로 구현한 웹소켓 서버입니다.");
        } catch (IOException e) { e.printStackTrace(); }
    }
    @OnMessage
    public void onMessage(Session session, String message) {
        String s = " - 웹소켓 서버로 부터의 응답!!";
        try {
            if (message.equals("multimessage")) {
                session.getBasicRemote().sendText("**** 2초에 한번씩 서버푸쉬를 10번 진행합니다. ****");
                for (int i = 1; i < 11; i++) {
                    try {
                        Thread.sleep(2000);
                    } catch (Exception e) { System.out.println("오류 발생 : " + e); }
                    session.getBasicRemote().sendText(i + s);
                }
            } else {
                session.getBasicRemote().sendText("[" + message + "]" + s);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```



# Web Socket API

## ■ 예제 : chatclient.html



# Web Socket API

## [ HTML 소스 ]

```
<h2>웹소켓 멀티 채팅 프로그램</h2>
<input type="text" id="inputMessage">
<button id="sendButton">메시지전송</button>
<pre id="output"></pre>
```

## [ JavaScript 소스 ]

```
var log = function(s) {
  console.log(s);
  if (document.readyState !== "complete") {
    log.buffer.push(s);
  } else {
    document.getElementById("output").innerHTML += (s + "\n")
  }
}
log.buffer = [];
var url = "ws://localhost:8888";
var ws = new WebSocket(url);
ws.onopen = function() {
  log("서버와 접속");
  ws.send("채팅 참가를 요청합니다.");
}
```

# Web Socket API

```
ws.onmessage = function(e) {
  console.log(e.data);
  log(e.data);
}
ws.onclose = function(e) {
  log("서버와의 접속해제");
}
window.onload = function() {
  log(log.buffer.join("\n"));
  document.getElementById("sendButton").onclick = function() {
    console.log(document.getElementById("inputMessage").value);
    ws.send(document.getElementById("inputMessage").value);
  }
  document.getElementById("inputMessage").onkeypress = function() {
    if (event.keyCode == '13') {
      value = document.getElementById("inputMessage").value
      ws.send(value);
      document.getElementById("inputMessage").value = "";
    }
  }
}
```

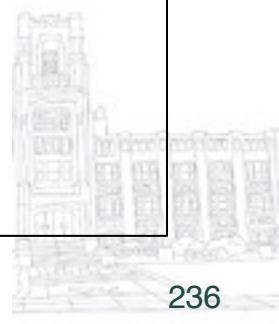
# Web Socket API

## [ chatserver.js ]

```
var WebSocketServer = require('websocket').server;
var http = require('http');
var clients = [];
// 임의로 ID부여하기 위함
var idlist = [];
var id = 1;
var server = http.createServer(function(request, response) {
    response.writeHead(404);
    response.end();
});
server.listen(8888, function() {
    console.log((new Date()) + ' - 8888 포트번호로 서버가 기동되었습니다.');
});
wsServer = new WebSocketServer({
    httpServer: server,
    autoAcceptConnections: false
});
function originIsAllowed(origin) {
    return true;
}
```

# Web Socket API

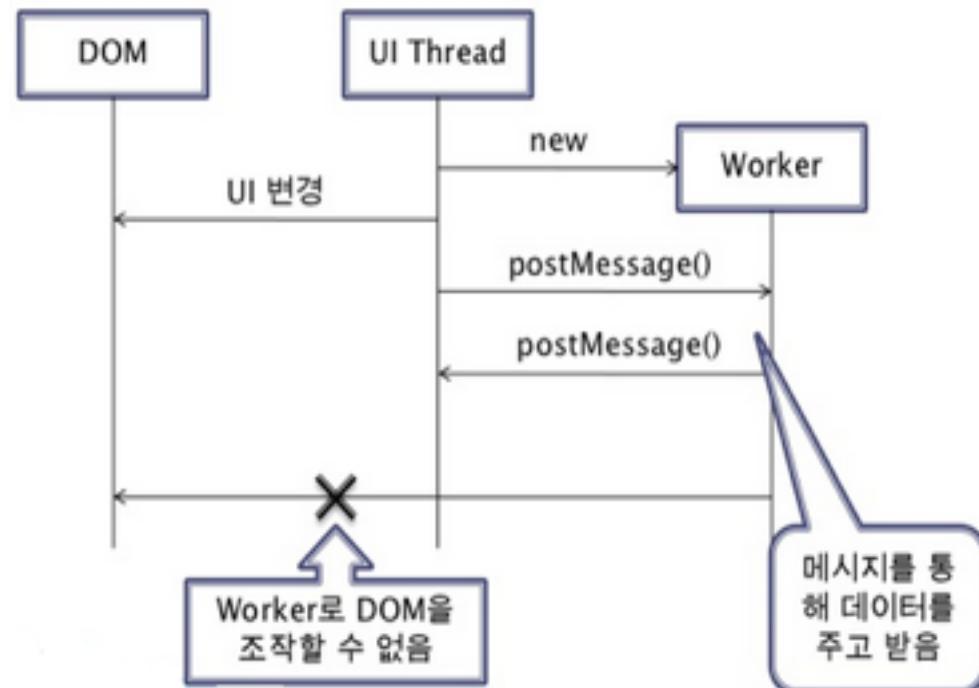
```
wsServer.on('request', function(request) {
  if (!originIsAllowed(request.origin)) {
    request.reject();
    console.log((new Date()) + request.origin + '로 부터의 접속 요청을 거절하였습니다.');
    return;
  }
  var connection = request.accept(null, request.origin);
  clients.push(connection);
  // 임의로 id값을 할당함. request.key값으로 client 구분
  idlist[request.key] = '손님-' + id++;
  console.log((new Date()) + ' 접속을 수락하였습니다.');
  connection.on('message', function(message) {
    if (message.type === 'utf8') {
      console.log('수신 메시지 : ' + message.utf8Data);
      clients.forEach(function(cli) {
        msg = idlist[request.key] + ':' + message.utf8Data;
        cli.sendUTF(msg);
      });
    }
    else if (message.type === 'binary') {
      console.log('바이너리 메시지를 전달받았습니다 : ' + message.binaryData.length + ' 바이트');
    }
  });
  connection.on('close', function(reasonCode, description) {
    console.log((new Date()) + connection.remoteAddress + ' 와의 접속상태가 해제되었습니다.');
  });
});
```



# Web Worker API

## □ 웹 워커(Web Worker) 란?

- 워커란 백드라운드에서 동작하는 스크립트이다.
- 오래 걸리는 작업 수행 시 워커를 이용할 수 있으나 워커에서는 DOM 조작과 window 나 document 와 같은 일부 내장 객체 사용이 불가하다.
- 웹 페이지내에 포함된 JavaScript 코드는 기본적으로 하나의 스레드로 동작하는데 이 스레드를 메인 스레드 또는 UI 라 하며 이때 추가로 기동되는 스레드를 워커라고 한다.
- 워커를 사용하면 JavaScript 코드 수행을 병렬처리 할 수 있다.
- 워커로 구현하면 좋은 예
  - AJAX
  - 네트워크를 통한 데이터 처리
  - 시간이 오려걸리는 산술연산
  - UI Thread에 방해없이 지속적으로 수행해야 하는 작업
  - 실시간적인 채크 작업



# Web Worker API

## 워커의 구현

- 워커에서 수행하고자 하는 기능을 .js 라는 확장자를 같은 스크립트 파일로 작성하거나 type="javascript/worker" 을 지정한 <script> 태그 안에 작성한 후에 BLOB 객체로 만들어 Worker 객체를 생성한다.

[ 첫 번째 방법 ]

```
var worker = new Worker("calcworker.js");
```

[ 두 번째 방법 ]

```
<script id="workercode" type="javascript/worker">  
    // 워커가 수행하게 될 코드  
</script>  
var blob = new Blob([document.querySelector('#workercode').textContent]);  
var worker = new Worker(window.URL.createObjectURL(blob));
```

- Worker 객체의 postMessage() 를 호출하여 워커를 수행시작 시킨다.  
(이 때 워커 수행에 필요한 메시지를 전송할 수 있다.)
- 워커로 부터 송신되는 메시지를 받기 위해 워커 객체에 message 이벤트에 대한 핸들러를 등록한다.
- 워커의 기능이 더 이상 필요 없으면 Worker 객체의 terminate() 메서드를 호출한다.
- 워커에서의 실행 오류를 처리하려면 Worker 객체에 error 이벤트에 대한 핸들러를 등록한다.

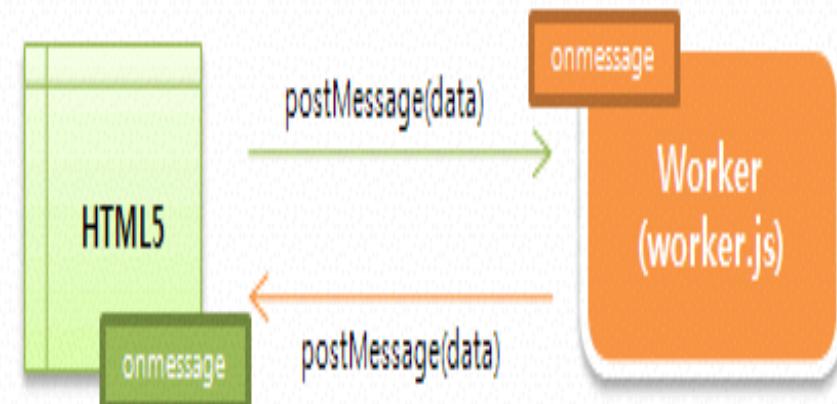
# Web Worker API

## 워커 객체의 주요 메서드

- **postMessage()**
  - 워커에게 메시지 전송시 또는 워커에서 처리한 결과값을 되돌려주기 위해 사용한다.
- **terminate()**
  - 워커를 종료하기 위해 사용한다.
- **onmessage** 에 등록되는 이벤트 핸들러
  - 워커로 보내온 메시지를 수신하거나 웹워커에서 보내온 메시지를 수신하기 위해 사용한다.
- **onerror** 에 등록되는 이벤트 핸들러
  - 워커 수행중 오류가 발생할 때 자동 호출된다.

## 워커의 코드 작성

```
self.onmessage = function(e) {  
    // 메인 스레드(UI 스레드)에서 전달된  
    // 데이터 추출  
    var data = e.data;  
  
    // 워커의 기능을 구현한 코드  
  
    var result = 수행 결과;  
    self.postMessage(result);  
};
```



# Web Worker API

## 워커의 코드 작성시 주의사항

- 워커에서는 모든 기능을 구현할 수 없다.
  - 메인 페이지에 정의된 전역 변수/함수 사용이 불가하며 window 객체, document 객체등을 사용할 수 없다.(UI 처리 불가)
  - 워커에사 사용 가능한 미리 정의된 전역 변수는 self이며 self 는 워커 자신을 나타내는 객체이다. (메인 페이지의 window 객체와 비슷한 역할 수행)
  - XMLHttpRequest 객체, WebSocket 객체 그리고 JavaScript 의 표준 내장 객체의 사용이 가능하다.
  - **self** 객체의 주요 멤버들
    - self.postMessage(msg) : 메시지를 보낸다.(메시지는 값으로 전달된다.)
    - self.importScripts(url[, url...]) : 지정된 스크립트 파일을 로드하여 포함한다.
    - self.close() : 워커를 종료한다.
    - onmessage : message 이벤트의 핸들러를 등록한다.
    - onerror : error 이벤트의 핸들러를 등록한다.
    - on.navigator : 브라우저의 정보를 담은 WorkerNavigator 객체를 제공한다.
    - self.location : 워커의 스크립트 파일에 대한 URL 정보를 제공하는 WorkerLocation 객체를 제공한다.
    - self.setTimeout(callback, timeout), self.clearTimeout() : window 객체의 메서드와 동일하게 타이머를 설정하거나 해제하는 기능을 제공한다.
    - self.setInterval (callback, timeout), self.clearInterval() : window 객체의 메서드와 동일하게 반복 타이머를 설정하거나 해제하는 기능을 제공한다.

# Web Worker API

## ■ 예제 : calcworker.html

계산중지

Sum :

2015/6/28 20:49:28

계산중지

Sum : 50000005000000

2015/6/28 20:50:23

메인 스레드에서 초단위로 시간을 측정하고 있는 상태에서 위와 같이 비교적 큰 숫자를 입력해 본다. 워커에서 1부터 이 숫자값까지의 합을 계산하여 메인 스레드에 전달해주므로 초단위의 시간 출력이 멈추지 않고 병렬 처리되는 것을 체크해 볼 수 있는 예제이다.

[ HTML 소스 ]

```
<input type="text" id="num">
<button onclick="calculate()">계산</button>
<button onclick="stopCalculate()">중지</button><br/>
<p id="sum">Sum :</p>
<p id="msg"></p>
```

# Web Worker API

## [ JavaScript 소스 ]

```
window.onload = function() {
    setInterval(timedisplay, 1000);
}
function timedisplay(){
    var date = new Date();
    var dateStr = date.getFullYear() + "/" + (date.getMonth() + 1) + "/" +
        date.getDate() + " " + date.getHours() + ":" + date.getMinutes() + ":" + date.getSeconds();
    var number = Math.floor(Math.random() * 100);
    document.getElementById("msg").innerHTML = dateStr;
}
var worker;
function calculate() {
    if(worker) { worker.terminate(); }
    var num = document.getElementById("num").value;
    worker = new Worker("calcworker.js");
    worker.onmessage = function(evt) {
        document.getElementById("sum").innerHTML = "Sum : " + evt.data;
    };
    worker.onerror=function(evt) {
        alert("Error : " + evt.message + " (" + evt.filename + ":" + evt.lineno + ")");
    };
    worker.postMessage(num);
}
function stopCalculate() {
    if(worker) { worker.terminate(); }
    alert("Worker is Stopped");
}
```

# Web Worker API

[ calcworker.js 소스 ]

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

**HTML**



**CSS**



**JS**



## 제5장 HTML5 구현 실습

## 구현 실습 -1 : 칼라 이미지를 흑백 이미지로 출력하기

**캔バス를 이용하여 비디오를 그레이톤으로 출력**



[ 적용된 주요 기술 ]

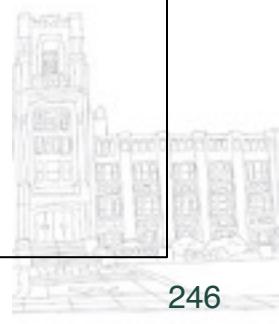
- Video API
- Canvas API



## 구현 실습 -1 : 칼라 이미지를 흑백 이미지로 출력하기

### [ HTML 소스 ]

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Video on Canvas</title>
<style>
.boxes{
  display: inline-block; margin: 10px; padding: 5px; border: 1px solid #999999;
}
</style>
<script src="canvasvideo.js"></script>
</head>
<body style="text-align:center;">
<header>
  <h1>캔バス를 이용하여 비디오를 그레이톤으로 출력</h1>
</header>
<section class="boxes">
  <video id="media" width="483" height="272">
    <source src="trailer2.mp4"> <source src="trailer2.ogg">
  </video>
</section>
<section class="boxes">
  <canvas id="canvas" width="483" height="272">
    Your browser doesn't support the canvas element
  </canvas>
</section>
</body>
</html>
```



## 구현 실습 -1 : 칼라 이미지를 흑백 이미지로 출력하기

### [ JavaScript 소스 ]

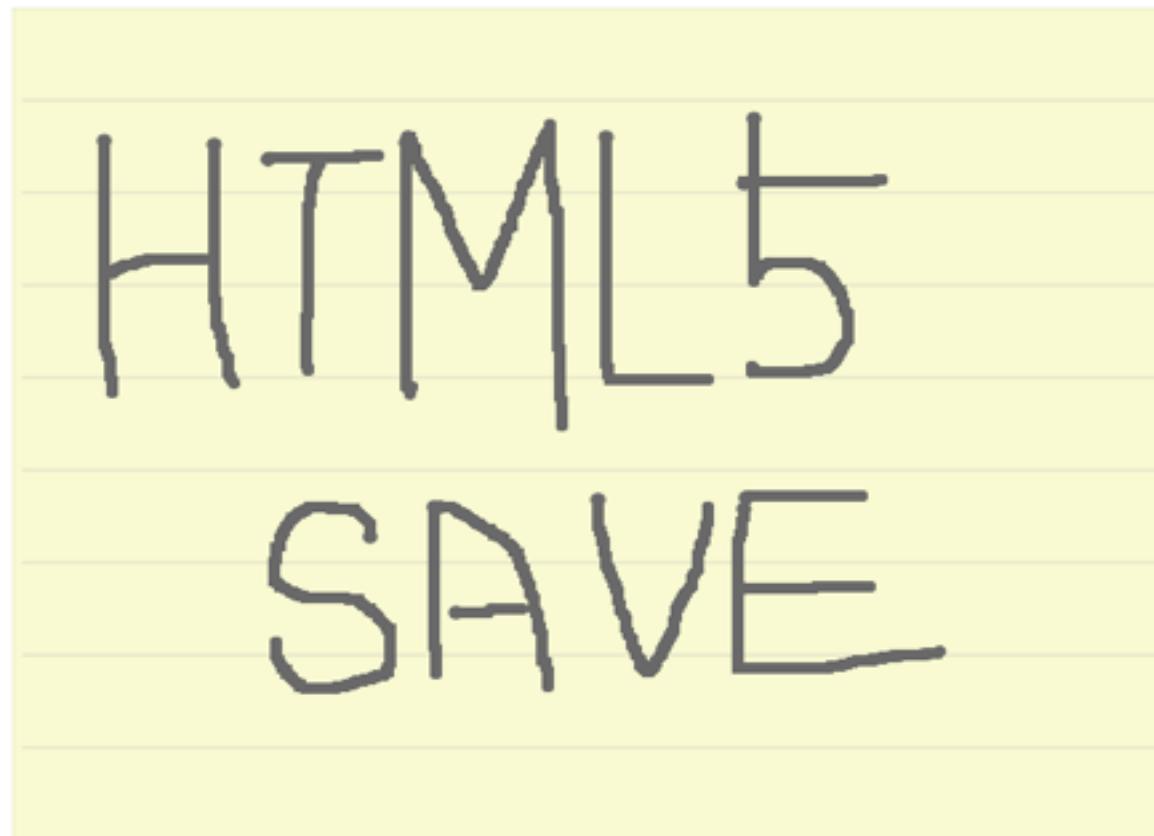
```
var canvas, video;
function initiate(){
    var elem=document.getElementById('canvas');
    canvas=elem.getContext('2d');
    video=document.getElementById('media');
    video.addEventListener('click', push, false);
}

function processFrames(){
    canvas.drawImage(video,0,0);
    var info=canvas.getImageData(0,0,483,272);
    var pos;
    var gray;
    for(x=0;x<483;x++){
        for(y=0;y<272;y++){
            pos=(info.width*4*y)+(x*4);
            gray=parseInt(info.data[pos]*0.2989 + info.data[pos+1]*0.5870 + info.data[pos+2]*0.1140);
            info.data[pos]=gray;
            info.data[pos+1]=gray;
            info.data[pos+2]=gray;
        }
    }
    canvas.putImageData(info,0,0);
}
window.addEventListener("load", initiate, false);

function push(){
    if(!video.paused && !video.ended){
        video.pause();
        window.clearInterval(loop);
    }else{
        video.play();
        loop=setInterval(processFrames, 33);
    }
}
```

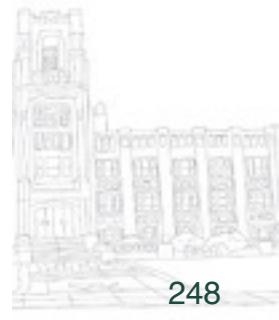
## 구현 실습 -2 : 메모 캔버스 저장과 업로드

Save Restore Clear Upload DownImage



[ 적용된 주요 기술 ]

- Canvas API, Web Storage API
- XMLHttpRequest 2



## 구현 실습 -2 : 메모 캔버스 저장과 업로드

### [ HTML 소스 ]

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8"/>
<title>Draw memo on canvas</title>
</head>
<body>
  <button onclick="save();">Save</button>
  <button onclick="restore();">Restore</button>
  <button onclick="initialize();">Clear</button>
  <button onclick="upload();">Upload</button>
  <button onclick="downimage();">DownImage</button><br><br>
  <canvas id="myCanvas" width="580" height="450"></canvas>

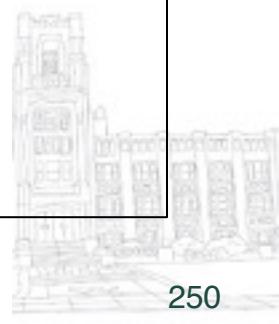
  <script type="text/javascript" src="memocanvasproc.js"></script>
</body>
</html>
```

## 구현 실습 -2 : 메모 캔버스 저장과 업로드

### [ JavaScript 소스 ]

```
var device
var drawing = false;
var canvas;
var context;
var rect;
function initialize() {
    context.clearRect(0,0,580,450);
    context.beginPath();
    context.rect(0,0,580,450);
    context.strokeStyle = "silver";
    context.fillStyle = "LightGoldenrodYellow";
    context.fill();

    context.lineWidth = 0.5;
    for(i=1;i<=8;i++) {
        context.moveTo(5,i*50);
        context.lineTo(575, i*50);
    }
    context.stroke();
}
function startDrawing() {
    if (device == "mobileDevice") event.preventDefault();
    event.preventDefault();
    drawing = true;
    context.beginPath();
    context.strokeStyle = "dimgray";
    context.lineWidth = 1;
```



## 구현 실습 -2 : 메모 캔버스 저장과 업로드

```
context.arc(event.clientX - rect.left, event.clientY - rect.top, 3, 0, 2*Math.PI)
context.stroke();
context.fillStyle = "dimgray";
context.fill();
context.closePath();

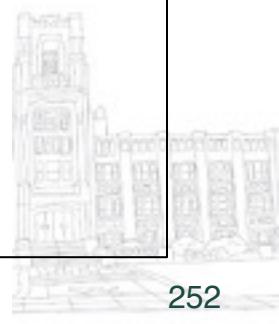
context.beginPath();
context.moveTo(event.clientX - rect.left, event.clientY - rect.top);
context.lineCap = "round";
context.lineWidth = 6;
}

function keepDrawing() {
if (drawing) {
    var x,y;
    if (device == "mobileDevice") {
        x = event.targetTouches[0].pageX;
        y = event.targetTouches[0].pageY;
    }
    else {
        x = event.clientX;
        y = event.clientY;
    }
    context.lineTo(x - rect.left, y - rect.top);
    context.stroke();
}
}
```

## 구현 실습 -2 : 메모 캔버스 저장과 업로드

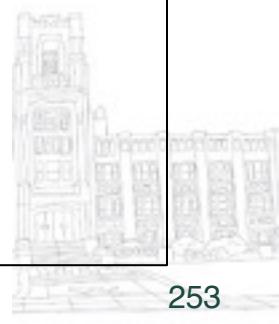
### [ JavaScript 소스 ]

```
function stopDrawing() {
    if (drawing) {
        context.stroke();
        drawing = false;
    }
}
function save() {
    localStorage.canvas = canvas.toDataURL();
}
function upload() {
    var data = new FormData();
    var myblob = new Blob([canvas.toDataURL()], {type : 'text/plain'});
    data.append('file', myblob, "test.txt");
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "canvasupload.jsp", true);
    xhr.send(data);
    xhr.onload=function() { alert("업로드완료");};
}
function downimage() {
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "canvasdownimage.jsp", true);
    xhr.send();
    xhr.onload=function(e) {
        var img = new Image();
        img.src = this.responseText;
        context.drawImage(img, 0, 0);
    }
}
```



## 구현 실습 -2 : 메모 캔바스 저장과 업로드

```
function restore() {
    var img = new Image();
    img.src = localStorage.canvas;
    img.onload = function() {
        context.drawImage(img, 0, 0);
    }
}
function getDeviceType() {
    var str = navigator.userAgent;
    if (str.match(/(ipad)|(iphone)|(ipod)|(android)|(webos)/i))
        device = "mobileDevice";
    else
        device = "desktopPC";
}
window.onload = function() {
    canvas = document.getElementById("myCanvas");
    context = canvas.getContext("2d");
    rect = canvas.getBoundingClientRect();
    getDeviceType();
    if(device == "mobileDevice") {
        canvas.ontouchstart = startDrawing;
        canvas.ontouchmove = keepDrawing;
        canvas.ontouchend = stopDrawing;
    } else {
        canvas.onmousedown = startDrawing;
        canvas.onmousemove = keepDrawing;
        canvas.onmouseup = stopDrawing;
    }
    initialize();
}
```



## 구현 실습 - 3 : 드래그 앤 드롭 쇼핑몰

### 드래그 & 드롭 쇼핑몰

The interface features a grid of nine flower arrangements arranged in three rows of three. The top row contains a round arrangement with purple and white flowers, a square arrangement with pink roses, and a circular arrangement with various colorful flowers. The middle row contains a bouquet wrapped in red paper with yellow and orange flowers, a bouquet wrapped in purple paper with small pink flowers, and a small arrangement with red roses. The bottom row contains a cluster of pink carnations and a small arrangement with green plants.

#### 주문

#### Shopping Cart

1	꽃바구니-화목	32000
2	꽃다발-우정	22000
1	화분-기쁨	10000

합계: 86000원

구입하려는 제품을  
이곳에 드래그하여 드롭하세요.

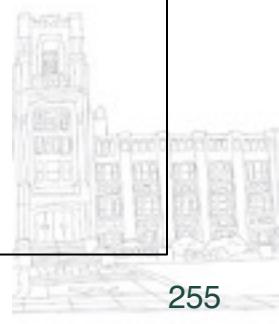
[ 적용된 주요 기술 ]

- Drag&Drop API, Web Storage API
- XMLHttpRequest 2, CSS3, jQuery

## 구현 실습 - 3 : 드래그 앤 드롭 쇼핑몰

[ HTML/JavaScript 소스 ]

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Drag and Drop Shopping Cart</title>
    <link rel="stylesheet" href="default.css" />
    <script>
      window.onload = function() {
        var submit = document.querySelector("input[type=submit]");
        for(i=0; i<localStorage.length; i++) {
          var id = localStorage.key(i);
          var item = $('#' + id),
              cartList = $("#cart ul"),
              total = $("#total span"),
              price = $('p:eq(1) span', item).text(),
              prevCartItem = null,
              notInCart = (function () {
                var lis = $('li', cartList), len = lis.length, i;
                for (i = 0; i < len; i++ ) {
                  var temp = $(lis[i]);
                  if (temp.data('id') === id) {
                    prevCartItem = temp;
                    return false;
                  }
                }
                return true;
              }()),
              quantLeftEl, quantBoughtEl, quantLeft;
```

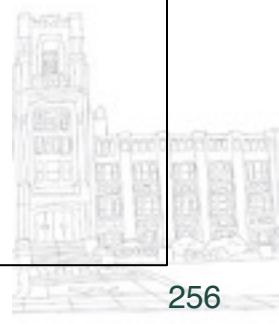


## 구현 실습 - 3 : 드래그 앤 드롭 쇼핑몰

```
$("h2").fadeOut('fast');
if (notInCart) {
    prevCartItem = $('<li />', {
        text : $('p:first', item).text(),
        data : { id : id }
    }).prepend('<span />', {
        'class' : 'quantity',
        text : '0'
    }).prepend('<span />', {
        'class' : 'price',
        text : price
    }).appendTo(cartList);
}
quantLeftEl = $('p:last span', item);
var cnt = parseInt(localStorage.getItem(id), 10);
quantLeft = parseInt(quantLeftEl.text(), 10) - cnt;
quantBoughtEl = $('.quantity', prevCartItem);

quantBoughtEl.text(parseInt(quantBoughtEl.text(), 10) + cnt);
quantLeftEl.text(quantLeft);

if (quantLeft === 0) {
    item.fadeOut('fast');
}
total.text(parseInt(total.text()) + parseInt(price));
}
submit.onclick = function() {
    var obj = {};
```



## 구현 실습 - 3 : 드래그 앤 드롭 쇼핑몰

```
for(i=0; i<localStorage.length; i++) {
    var id = localStorage.key(i);
    var value = localStorage.getItem(id);
    obj[id] = value;
}
$.get("order.jsp", $.param(obj), function(data) {
    alert(data);
    localStorage.clear();
    $("#listarea").empty();
    $("#total span").text(0);
});
}
}
</script>
</head>
<body>
<header>
    <div>드래그 & 드롭 쇼핑몰</div>
</header>
<section>
<ul id="products">
    <li><a class="item" href="#" id="basket1" draggable="true">
        
        <div>
            <p><strong>꽃바구니-사랑</strong></p>
            <p><strong>가격</strong>: <span>35000</span></p>
            <p><strong>남은수량</strong>: <span>10</span></p>
        </div>
    </a></li>
```

## 구현 실습 - 3 : 드래그 앤 드롭 쇼핑몰

```
<li><a class="item" href="#" id="basket2" draggable="true">
    
    <div>
        <p><strong>꽃바구니-화목</strong></p>
        <p><strong>가격</strong>: <span>32000</span></p>
        <p><strong>남은수량</strong>: <span>16</span></p>
    </div>
</a></li>
<li><a class="item" href="#" id="basket3" draggable="true">
    
    <div>
        <p><strong>꽃바구니-행복</strong></p>
        <p><strong>가격</strong>: <span>30000</span></p>
        <p><strong>남은수량</strong>: <span>9</span></p>
    </div>
</a></li>
<li><a class="item" href="#" id="bouquet1" draggable="true">
    
    <div>
        <p><strong>꽃다발-우정</strong></p>
        <p><strong>가격</strong>: <span>22000</span></p>
        <p><strong>남은수량</strong>: <span>4</span></p>
    </div>
</a></li>
<li><a class="item" href="#" id="bouquet2" draggable="true">
    
```

## 구현 실습 - 3 : 드래그 앤 드롭 쇼핑몰

```
<div>
  <p><strong>꽃다발-축하</strong></p>
  <p><strong>가격</strong>: <span>15000</span></p>
  <p><strong>남은수량</strong>: <span>20</span></p>
</div>
</a></li>
<li><a class="item" href="#" id="bouquet3" draggable="true">
  
  <div>
    <p><strong>꽃다발-감사</strong></p>
    <p><strong>가격</strong>: <span>20000</span></p>
    <p><strong>남은수량</strong>: <span>13</span></p>
  </div>
</a></li>
<li><a class="item" href="#" id="flowerpot1" draggable="true">
  
  <div>
    <p><strong>화분-기쁨</strong></p>
    <p><strong>가격</strong>: <span>10000</span></p>
    <p><strong>남은수량</strong>: <span>18</span></p>
  </div>
</a></li>
```



## 구현 실습 - 3 : 드래그 앤 드롭 쇼핑몰

```
<li><a class="item" href="#" id="flowerpot2" draggable="true">
    
    <div>
        <p><strong>화분-건강</strong></p>
        <p><strong>가격</strong>: <span>30000</span></p>
        <p><strong>남은수량</strong>: <span>15</span></p>
    </div>
</a></li>
</ul>
</section>
<section id="cartarea">
    <div id="submitarea">
        <input type="submit" value="주 문 ">
    </div>
    <div id="cart">
        <h1>Shopping Cart</h1>
        <ul id="listarea"></ul>
        <div id="total"><strong>합계:</strong> <span>0</span>원</div>
        <div id="msg">
            <h3>구입하려는 제품을</h3>
            <h3>이곳에 드래그하여 드롭하세요.</h3>
        </div>
    </div>
</section>
<script src="jquery.min.js"></script>
<script src="jquery.ndd.js"></script>
<script src="dragdrop.js"></script>
</body>
</html>
```

## 구현 실습 - 4 : 스티키 노트

색상: 노란색 ▾ 내용: CSS3

스티키 노트 추가 스티키 노트 전부 삭제

스티커노트를 테스트합니다.

행복한 하루 되세요!!

즐거운 하루....

ㅋㅋㅋㅋㅋㅋ

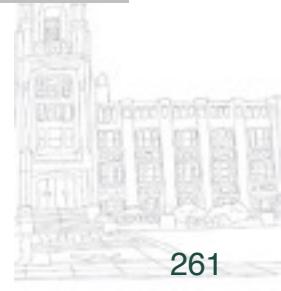
HTML5를 학습해요!!

JavaScript & jQuery

CSS3

[ 적용된 주요 기술 ]

- Web Storage API, JSON API
- CSS3의 주요 기술



## 구현 실습 - 4 : 스티키 노트

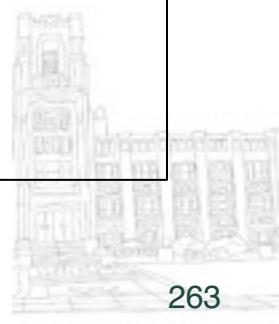
### [ HTML 소스 ]

```
<!DOCTYPE html>
<html>
<head>
<title>스티키 노트</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
<link rel="stylesheet" href="note.css">
<script src="note.js"></script>
</head>
<body>
<form>
  <label for="note_color">색상: </label>
  <select id="note_color">
    <option value="Gold">노란색</option>
    <option value="PaleGreen">녹색</option>
    <option value="LightPink">분홍색</option>
    <option value="LightBlue">파란색</option>
  </select>
  <label for="note_text">내용:</label> <input type="text" id="note_text">
  <input type="button" id="add_button" value="스티키 노트 추가">
  <input type="button" id="clear_button" value="스티키 노트 전부 삭제">
</form>
<ul id="stickies">          </ul>
</body>
</html>
```

## 구현 실습 - 4 : 스티키 노트

### [ CSS3 소스 ]

```
body {  
    background-color: silver;  
}  
form input#note_text {  
    width: 350px;  
}  
ul#stickies li {  
    display: block; list-style: none;  
    z-index: 1;  
    float: left;  
    margin: 30px;  
    padding: 15px 15px 50px 15px;  
    width: 100px; height: 100px;  
    border: 1px solid #bfbfbf;  
    background-color: Gold;  
    color: black; text-decoration: none;  
    box-shadow: 2px 2px 4px rgba(0, 0, 0, 0.4);  
    transition: all 0.5s ease-in;  
    overflow: hidden;  
}  
ul#stickies li span.sticky {  
    font-family: Verdana, Helvetica, sans-serif;  
}  
ul#stickies li:nth-child(even) {  
    transform: rotate(2deg);  
}  
  
ul#stickies li:nth-child(odd) {  
    transform: rotate(-1deg);  
}  
ul#stickies li:nth-child(3n) {  
    transform: rotate(1deg);  
}  
ul#stickies li:hover {  
    box-shadow: 5px 5px 6px  
        rgba(0, 0, 0, 0.4);  
    transform: rotate(0deg) scale(1.25);  
    z-index: 10;  
}
```



## 구현 실습 - 4 : 스티키 노트

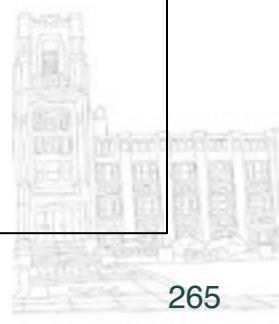
### [ JavaScript 소스 ]

```
window.onload = init;
function init() {
    var button = document.getElementById("add_button");
    button.onclick = createSticky;
    var clearButton = document.getElementById("clear_button");
    clearButton.onclick = clearStickyNotes;
    var stickiesArray = getStickiesArray();
    for (var i = 0; i < stickiesArray.length; i++) {
        var key = stickiesArray[i];
        var value = JSON.parse(localStorage[key]);
        addStickyToDOM(key, value);
    }
}
function getStickiesArray() {
    var stickiesArray = localStorage.getItem("stickiesArray");
    if (!stickiesArray) {
        stickiesArray = [];
        localStorage.setItem("stickiesArray", JSON.stringify(stickiesArray));
    } else {
        stickiesArray = JSON.parse(stickiesArray);
    }
    return stickiesArray;
}
function createSticky() {
    var stickiesArray = getStickiesArray();
    var value = document.getElementById("note_text").value;
    var colorSelectObj = document.getElementById("note_color");
```

## 구현 실습 - 4 : 스티키 노트

```
var index = colorSelectObj.selectedIndex;
var color = colorSelectObj[index].value;
var currentDate = new Date();
var key = "sticky_" + currentDate.getTime();
var stickyObj = {
    "value": value,
    "color": color
};
localStorage.setItem(key, JSON.stringify(stickyObj));
stickiesArray.push(key);
localStorage.setItem("stickiesArray", JSON.stringify(stickiesArray));
addStickyToDOM(key, stickyObj);
}

function deleteSticky(e) {
    var key = e.target.id;
    if (e.target.tagName.toLowerCase() == "span") {
        key = e.target.parentNode.id;
    }
    var stickiesArray = getStickiesArray();
    if (stickiesArray) {
        for (var i = 0; i < stickiesArray.length; i++) {
            if (key == stickiesArray[i]) {
                stickiesArray.splice(i, 1);
            }
        }
        localStorage.removeItem(key);
        localStorage.setItem("stickiesArray", JSON.stringify(stickiesArray));
        removeStickyFromDOM(key);
    }
}
```



## 구현 실습 - 4 : 스티키 노트

```
function addStickyToDOM(key, stickyObj) {
    var stickies = document.getElementById("stickies");
    var sticky = document.createElement("li");
    sticky.setAttribute("id", key);
    sticky.style.backgroundColor = stickyObj.color;
    var span = document.createElement("span");
    span.setAttribute("class", "sticky");
    span.innerHTML = stickyObj.value;
    sticky.appendChild(span);
    stickies.appendChild(sticky);
    sticky.onclick = deleteSticky;
}
function removeStickyFromDOM(key) {
    var sticky = document.getElementById(key);
    sticky.parentNode.removeChild(sticky);
}
function clearStickyNotes() {
    localStorage.clear();
    var stickyList = document.getElementById("stickies");
    var stickies = stickyList.childNodes;
    for (var i = stickies.length-1; i >= 0; i--) {
        stickyList.removeChild(stickies[i]);
    }
    var stickiesArray = getStickiesArray();
}
```

# Thank You

**HTML**



**CSS**



**JS**



## 제6장 Extend JavaScript

## Use === Instead of ==

- “If two operands are of the same type and value, then === produces true and !== produces false.”  
by JavaScript: The Good Parts



## Eval = Bad

- Not only will this decrease your script's performance substantially, but it also poses a huge security risk because it grants far too much power to the passed in text. Avoid it!



## Don't Use Short-Hand

- ```
if(someVariableExists)
    x = false
    anotherFunctionCall();
```
- ```
if(someVariableExists) {
    x = false;
}
anotherFunctionCall();
```



## Utilize JS Lint

- Before signing off on a script, run it through JSLint just to be sure that you haven't made any mindless mistakes.



## Place Scripts at the Bottom of Your Page

```
</div><!-- end container-->
<script type="text/javascript" src="?>
<script type="text/javascript" src="?>
</body>
</html>
```



# Declare Variables Outside of the For Statement

Bad

```
1 for(var i = 0; i < someArray.length; i++) {  
2     var container = document.getElementById('container');  
3     container.innerHTML += 'my number: ' + i;  
4     console.log(i);  
5 }
```

Better

```
1 var container = document.getElementById('container');  
2 for(var i = 0, len = someArray.length; i < len; i++) {  
3     container.innerHTML += 'my number: ' + i;  
4     console.log(i);  
5 }
```



## The Fastest Way to Build a String

- `var arr = ['item 1', 'item 2', 'item 3', ...];`
- `var list = '<ul><li>' + arr.join('</li><li>') + '</li></ul>';`



# Reduce Globals

```
1 var name = 'Jeffrey';
2 var lastName = 'Way';
3
4 function doSomething() {...}
5
6 console.log(name); // Jeffrey -- or window.name
```

Better

```
1 var DudeNameSpace = {
2   name : 'Jeffrey',
3   lastName : 'Way',
4   doSomething : function() {...}
5 }
6 console.log(DudeNameSpace.name); // Jeffrey
```



## Comment Your Code

```
1 // Cycle through array and echo out each name.  
2 for(var i = 0, len = array.length; i < len; i++) {  
3     console.log(array[i]);  
4 }
```



## Don't Use the "With" Statement

```
1 | being.person.man.bodyparts.arms = true;  
2 | being.person.man.bodyparts.legs= true;
```

```
1 | with (being.person.man.bodyparts) {  
2 |     arms = true;  
3 |     legs = true;  
4 | }
```

```
1 | var o = being.person.man.bodyparts;  
2 | o.arms = true;  
3 | o.legs = true;
```



## Use {} Instead of New Object()

```
1 var o = new Object();
2 o.name = 'Jeffrey';
3 o.lastName = 'Way';
4 o.someFunction = function() {
5     console.log(this.name);
6 }
```

```
1 var o = {
2     name: 'Jeffrey',
3     lastName = 'Way',
4     someFunction : function() {
5         console.log(this.name);
6     }
7 };
```

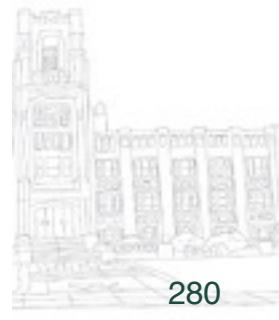
# Use [] Instead of New Array()

Okay

```
1 | var a = new Array();
2 | a[0] = "Joe";
3 | a[1] = 'Plumber';
```

Better

```
1 | var a = ['Joe', 'Plumber'];
```



## Always, Always Use Semicolons

- Sip of coffee could solve this!



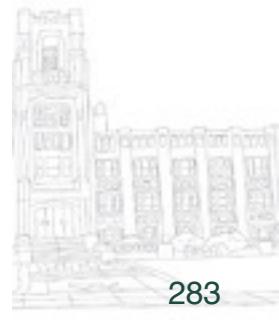
## "For in" Statements

```
1 | for(key in object) {  
2 |   if(object.hasOwnProperty(key)) {  
3 |     ...then do something...  
4 |   }  
5 | }
```



# Self-Executing Functions

```
1 (function doSomething() {  
2     return {  
3         name: 'jeff',  
4         lastName: 'way'  
5     };  
6 })();
```



## Raw JavaScript Can Always Be Quicker Than Using a Library

- jQuery's "each" method is great for looping, but using a native "for" statement will always be an ounce quicker.



## Remove “type” & "Language"

- <script type="text/javascript"  
language="javascript">



# call

```
1 | function sum(num1, num2) {  
2 |     return num1 + num2;  
3 | }  
4 |  
5 | function callSum(num1, num2) {  
6 |     return sum.call(this, num1, num2);  
7 | }  
8 |  
9 | alert(callSum(10, 10)); // 20
```



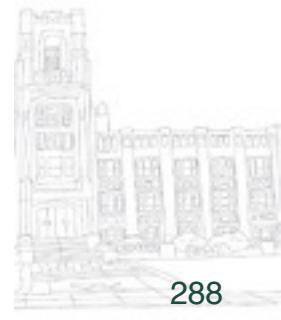
## apply

```
function sum(num1, num2) {  
    return num1 + num2;  
}  
  
function callSum1(num1, num2) {  
    return sum.apply(this, arguments); // arguments 객체를 넘깁니다.  
}  
  
function callSum2(num1, num2) { // 배열을 넘깁니다.  
    return sum.apply(this, [num1, num2]);  
}  
  
alert(callSum1(10, 10)); // 20  
alert(callSum2(10, 10)); // 20
```



# bind

```
1 window.color = "red";
2 var o = {color: "blue"};
3
4 function sayColor() {
5     alert(this.color);
6 }
7
8 var objectSayColor = sayColor.bind(o);
9 objectSayColor();      // blue
```



# garbage collection

```
1 function f(){
2     var o = {};
3     var o2 = {};
4     o.a = o2; // o는 o2를 참조한다.
5     o2.a = o; // o2는 o를 참조한다.
6
7     return "azerty";
8 }
```



## WebStorm Tip

- annotation
- coffee compile (preference plugin install first)
- live template (temple example)
- global search
- clipboard compare
- ctrl + g (add selection for next occur alt+j?)
- file search ctrl+shift+n

