서울시 인재개발원
최신 자바스크립트 요약교재

작성자 : 김은옥(probemedia@gmail.com)

* 수업파일 다운로드 위치 - https://github.com/probemedia/web
* 수업파일 - 2024_web.zip

* 과정목적 : 기관의 사이트 유지보수
* 학습목표 : 최신 웹사이트를 이해하고 활용할 수 있음

- 목차 -

1. 웹 아키텍처와 웹 표준 기술 개요

가. 웹 아키텍처

1) 웹 앱, 데스크탑 앱, 모바일 앱

- 웹 앱(웹사이트) : 웹서버의 프로그램을 네트워크를 통해서 웹브라우저에서 실행
- 데스크탑앱, 모바일 앱 : 컴퓨터에 설치되어 실행
  - 모바일 앱: 안드로이드/iOS 운영체제가 설치된 컴퓨터에서 실행되는 프로그램(애플리케이션, 앱)
  - 데스크탑 앱 윈도우즈/Mac 운영체제가 설치된 컴퓨터에서 실행되는 프로그램(애플리케이션 앱). 예)한글 엑셀.

2) 웹 어플리케이션의 구조 및 구성요소

가) 웹 어플리케이션의 구조

- 웹 어플리케이션이란 웹을 기반으로 실행되는 프로그램, 웹 프로그래밍을 통해 구현.

- 웹 어플리케이션의 처리 구조

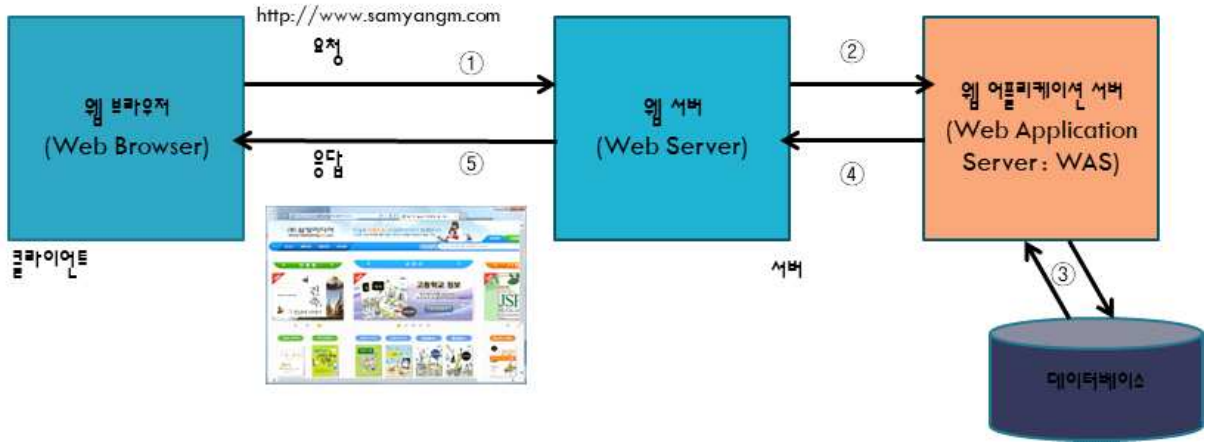
① 웹 브라우저가 웹 서버에 어떠한 페이지를 요청하게 되면

② 해당 웹 서버는 웹 브라우저의 요청을 받아서 요청된 페이지의 로직 및 데이터베이스와의 연동을 위해 웹 어플리케이션 서버에 이들의 처리를 요청

③ 이때 웹 어플리케이션 서버는 데이터베이스와의 연동이 필요한 경우 이를 수행.

④ 로직 및 데이터베이스 작업의 처리 결과를 웹 서버에 돌려보냄

⑤ 웹 서버는 결과를 다시 웹 브라우저에 응답.



나) 웹 어플리케이션 구성요소

- 웹 브라우저: 웹에서 클라이언트이며, 사용자의 작업 창. 예) IE, Chrome, Safari, FireFox 등
- 웹서버: 클라이언트의 요청을 받은 웹 페이지를 서비스 예) 아파치(Apache), IIS(Internet Information Server), Tomcat
- 웹컨테이너: jsp, servlet을 해독. 예)Tomcat cf) EJB컨테이너 - jboss
- 웹 애플리케이션 서버(WAS, Web Application Server): 로직처리, DB연동. 예) Tomcat, Jetty, Weblogic, Jeus, JBoss
- 데이터베이스 : 데이터의 저장소. 예) Oracle, Sybase, Informix, DB2, Mssql, MySQL

나. 웹 표준 기술 개요

1) 웹페이지 구성

가) 클라이언트 사이드 : html , css, js

- 사용자가 보는 화면을 구성 : UI

- html : 문서의 구조(내용)를 태그를 사용해서 기술
- css : 태그를 표현. 반응형 웹 구현 예)태그를 버튼, 메뉴로 표현
- js : 태그에서 발생한 동작처리, Ajax구현

나) 서버사이드 : jsp

- 로직을 처리하고 DB연동

- 컨트롤러 : 서블릿클래스
- 뷰 : jsp
- 모델 : 로직 - 자바클래스, 자바빈(DB연동 로직)

- 그 밖의 서버사이드 : CGI, ASP, PHP

2) 웹페이지 실행 : 웹브라우저 사용

- 웹서버 : 웹페이지를 서비스(제공)
- 웹브라우저 : 웹 페이지를 실행하는 프로그램

2. 개발환경 설정 : 서버사이드(jsp) 확장을 고려한 세팅

- 웹브라우저, 자바(JDK), 톰캣(Tomcat), 이클립스(Eclipse IDE for JAVA EE Developers) 를 설치 후 작업환경 설정

가. 웹 브라우저 설치 : 웹 페이지를 실행하는 프로그램

1) 구글 크롬 : html5권고안을 가장 잘 반영

- 구글 크롬 다운로드 사이트 https://www.google.com/chrome/ 로 이동
- [Chrome 다운로드]버튼 클릭, 설치 진행
  - [크롬 개발자 도구]표시 : Ctrl+Shift+i
  - [크롬 개발자 도구]해제 : [크롬 개발자 도구]가 표시된 상태에서 Ctrl+Shift+i

참고: 애플 사파리 - 맥pc에서만 최신 브라우저 제공

참고: 크롬에 Web Developer설치

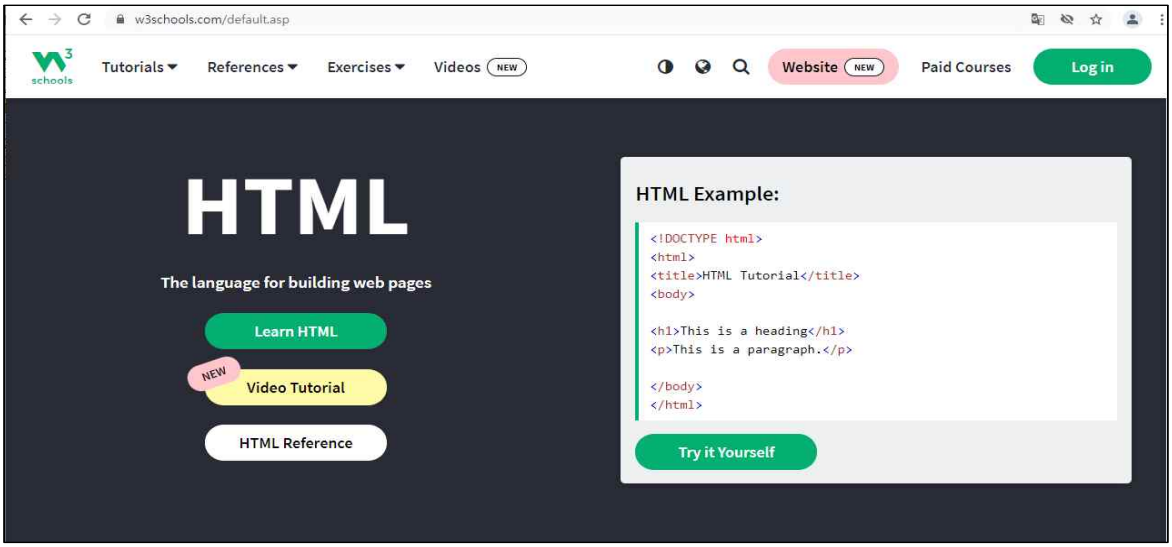
- 크롬브라우저에서 [웹스토어] 선택
- Web Developer 입력 후 검색
- 확장프로그램 Web Developer 추가 설치

나. 클라우드 기반 에디터 사용 - 무설치

- 다음의 웹 기반 에디터 사이트를 활용해서 설치하지 않고 웹 기반 코딩 수행

1) w3schools.com

- https://www.w3schools.com 사이트로 이동

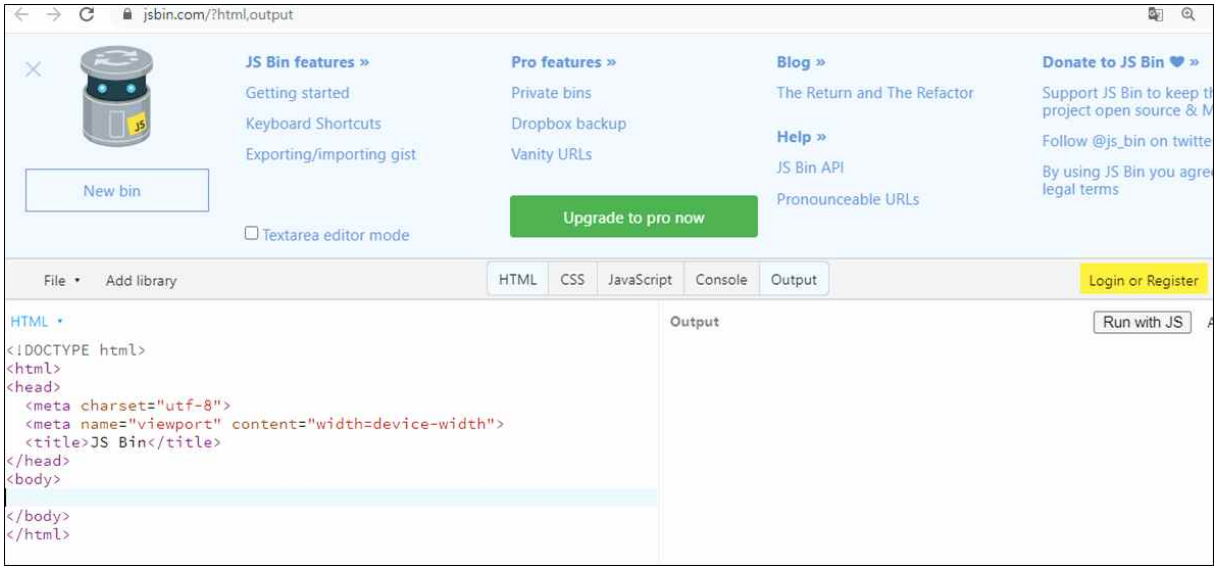


- 에서 [Try it Yourself]버튼 클릭

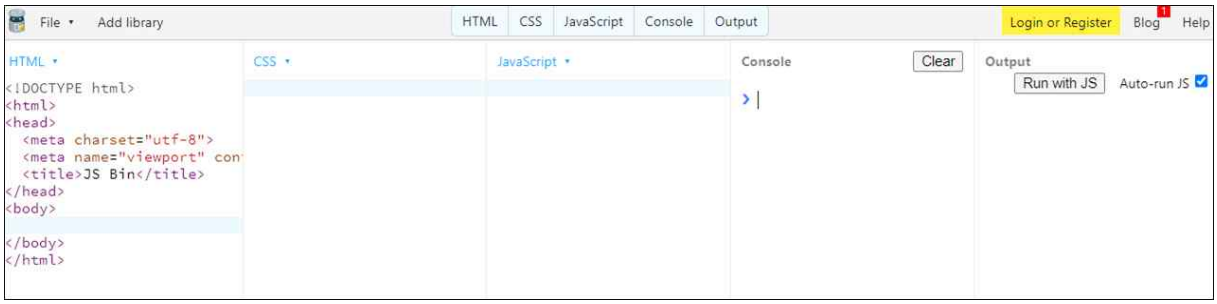
- 코드 수정 가능하며 [Run]버튼을 눌러 실행

2) jsbin.com

- http://jsbin.com/ 사이트로 이동

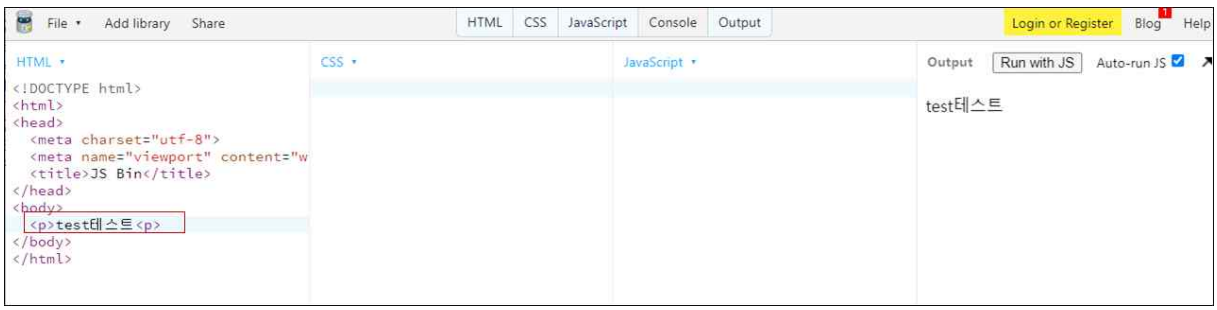


- 기본 선택된 [HTML], [Output]버튼 이외에 필요할 경우 [CSS], [JavaScript], [Console]버튼 클릭



- 코드를 추가하면 자동으로 실행이 확인됨.

예) <p>test테스트<p>



3) 구글 Colab사용

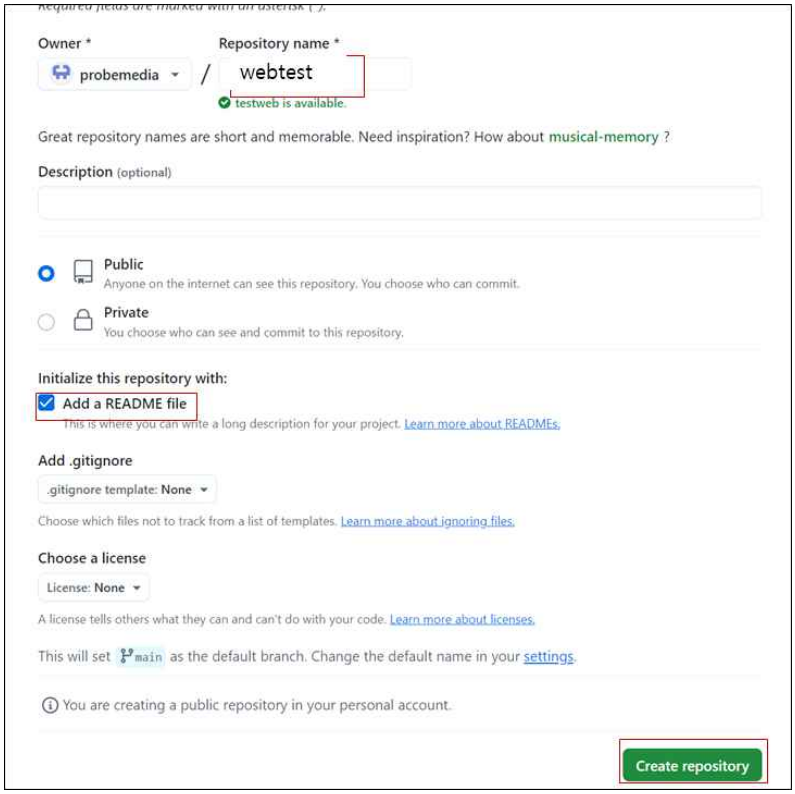
- Google Colab에서 html 실행: %%html 매직을 셀의 첫줄에 입력 후 사용

예시)

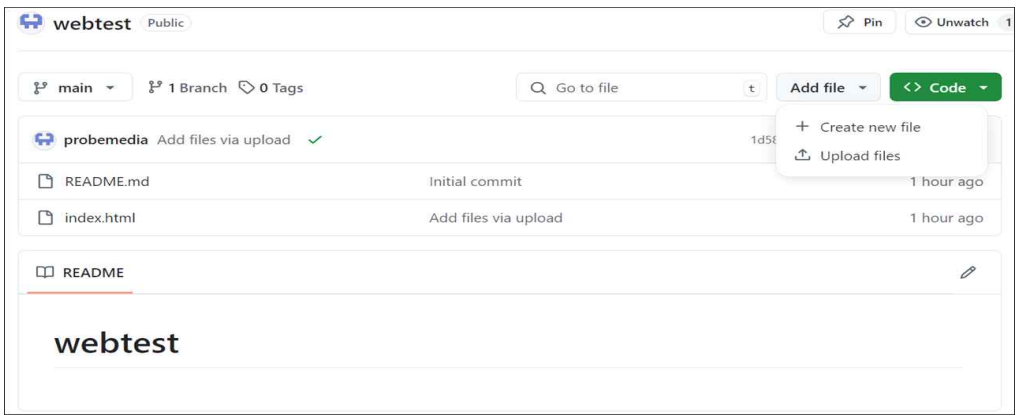
```
%%html
<h1>Hello, Colab!</h1>
<p>This is a paragraph.</p>
```

- github 레포지터리에서 웹서비스 : 회원가입 후 로그인해서 사용

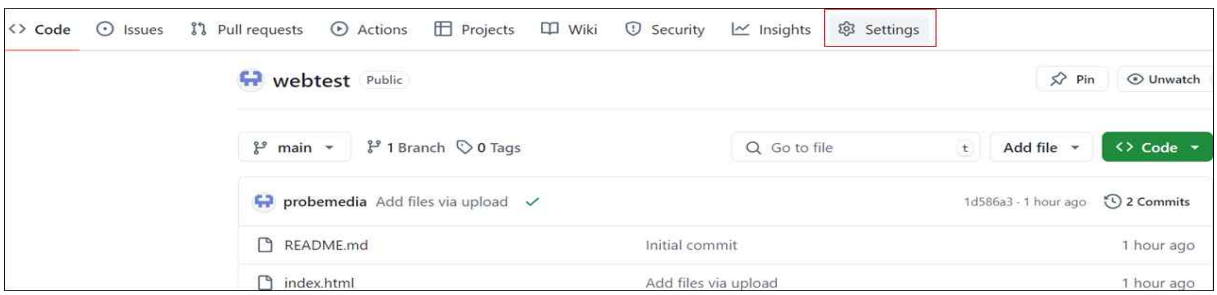
❶ github Repository에서 [New]를 눌러 새로운 레포지터리 생성



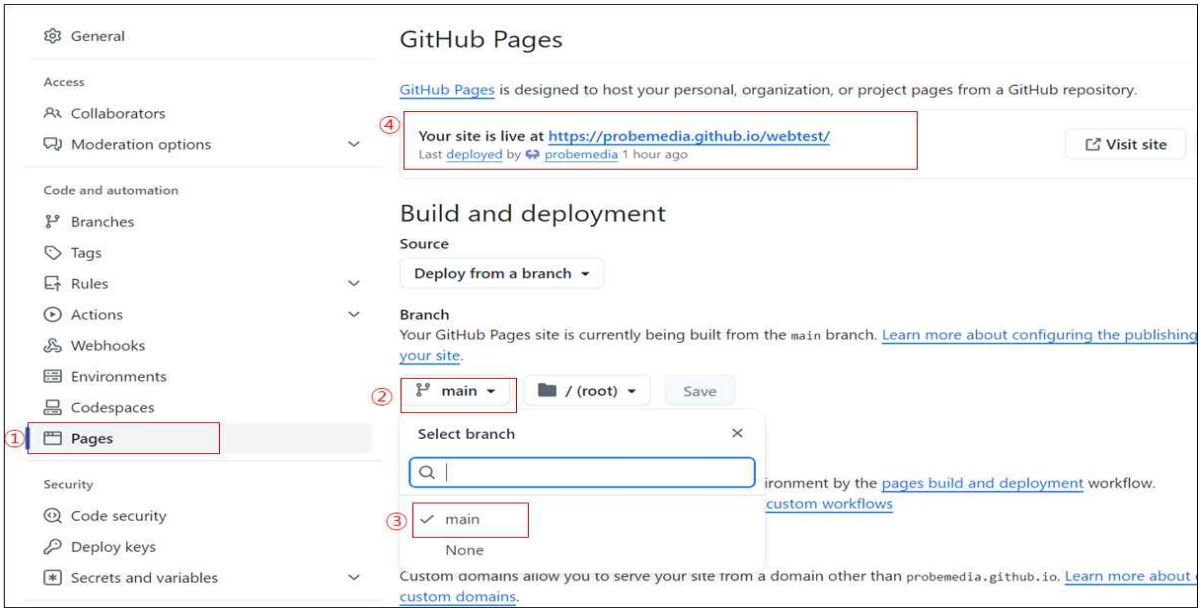
❷ index.html 업로드 또는 생성



❸ [settings]클릭



❹ 왼쪽의 [Pages]항목 선택, [Brench]항목의 [None]을 클릭해서 [main]선택



참고: 깃허브 가입

❶ https://github.com/ 에서 [Sign in] 클릭.

- [Sign in] : 로그인. 로그인시 아이디, 패스워드 입력 후 [Sign in] 버튼 클릭
- [Create account] : 회원가입. 회원가입시 [Create account]링크 클릭

❷ 화면이 전환되면 이메일 주소 입력 후 나머지 정보 입력하고 [Create account]버튼 클릭

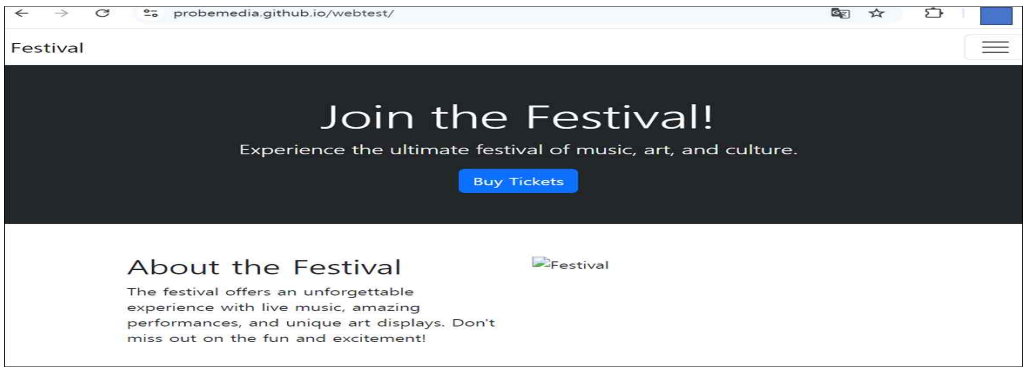
- Enter your email : 보안코드를 전송받을 이메일 입력.
- Create a password : password입력, 이메일주소의 비밀번호와 다른 비밀번호 사용 권장.
- Enter a username : username은 아이디. 깃허브에서 사용할 아이디명을 입력.
- Email preferences : github 업데이트 및 공지 등을 메일로 수신할지 여부. 선택안함

❸ 회원가입시 입력한 이메일로 받은 인증코드를 입력

❹ Welcom to Github 에서 깃허브를 어떤 목적, 사용인원 수 체크 후 [Continue]버튼 클릭, 무료/유료사용 선택화면에서 무료사용인 [Continue for free]버튼 클릭

❺ 저장소 생성시 [Repository]클릭, [New]클릭

❺ 잠시후 [GitHub Pages]에 웹페이지 배포 주소가 표시됨. 해당링크로 이동하면 웹페이지가 서비스되어 표시됨



4) 실행환경 테스트

- w3schools.com 또는 jsbin.com에서 다음코드 입력. test.html에서 제공

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0"/>
<title>JavaScript test</title>
</head>
<body>
  <script>
    var var1 = 1;
    document.write(var1);
  </script>
</body>
</html>
```

- w3schools.com에서는 입력 후 [Run]버튼 클릭, jsbin.com에서 코드 입력하면 자동 실행

w3schools.com에서 실행

jsbin.com에서 실행

jsbin.com의 소소코드 다운로드

- 코드작성
- [File]-[Save snapshot] : CTRL+S
- [Download]

다. PC에 프로그램 설치

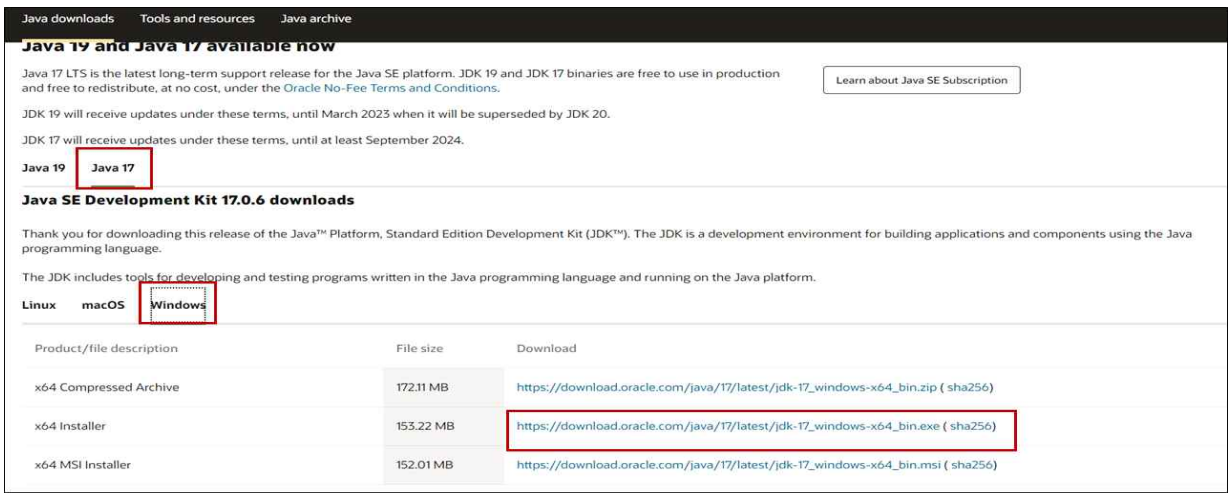
1) JDK 다운로드, 설치

- Java 17 LTS(이하 JDK 17) 버전은 자바(Java SE) 플랫폼의 최신 LTS(long-term support release)으로, LTS버전은 최소 8년 동안 성능 및 안정성 그라고 보안 등의 업데이트가 예정되어 있는 중요한 개발버전.(참고 https://blogs.oracle.com/javakr/post/jdk-17)
- JDK 17부터 Oracle No-Fee Terms and Conditions (NFTC) 라이선스를 적용해서 상업 및 프로덕션 용도를 포함하여, 모든 사용자들에게 무료로 사용 및 배포가 허용. 개발 툴로 권장되는 JDK버전은 8, 11, 17이 있으며 여기서는 최신인 17을 사용.

가) JDK 다운로드

- 자바기반의 프로젝트(자바프로젝트, JSP웹프로젝트, 안드로이드프로젝트)작성에 필요.
- 빅데이터에서 한글텍스트마이닝을 지원하는 KoNLP패키지 사용에 필요.
- 이클립스 사용에 필요

- https://www.oracle.com/java/technologies/downloads/사이트
- [Java downloads]항목이 표시되면 [Java 17]탭 클릭, [Windows]탭을 클릭 후 [x64 Installer]항목의 다운로드 링크를 클릭



나) JDK 설치

- 다운로드된 jdk-17_windows-x64_bin.exe 파일이 더블클릭해서 설치
- 기본값 그대로 사용해서 설치 : 라이센스에 동의, [Next]버튼을 누르다가 대화상자의 타이틀 바가 설치 완료를 표시하는 [Java(TM) SE Development Kit 17.~ - Complete]로 변경되면 [Close]버튼을 눌러 설치를 끝

3) JDK 환경변수 설정

- 자바명령어의 위치와 다른 프로그램[DBMS,톰캣,빅데이터처리 등등]과의 연동에 필요한 자바위치를 시스템에 알림

- [제어판]-[시스템] 또는 바탕화면의 [내 PC]-[속성]클릭
- 왼쪽메뉴에서 [고급시스템 설정]항목에서 클릭
- [시스템속성]대화상자에서 [고급]탭에서 [환경변수]버튼 클릭
- 표시되는 창에서 [시스템변수]항목에 PATH, JAVA_HOME항목 설정

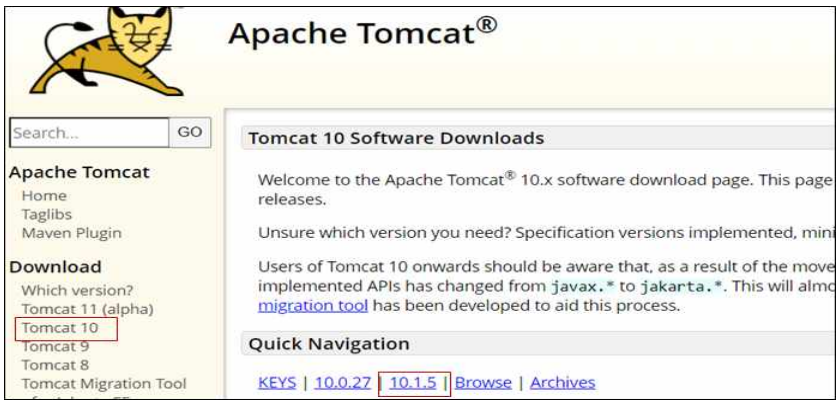
환경변수 명	환경변수 값
PATH	C:\Program Files\Java\jdk-17\bin
JAVA_HOME	C:\Program Files\Java\jdk-17

4) 톰캣 다운로드 및 설치

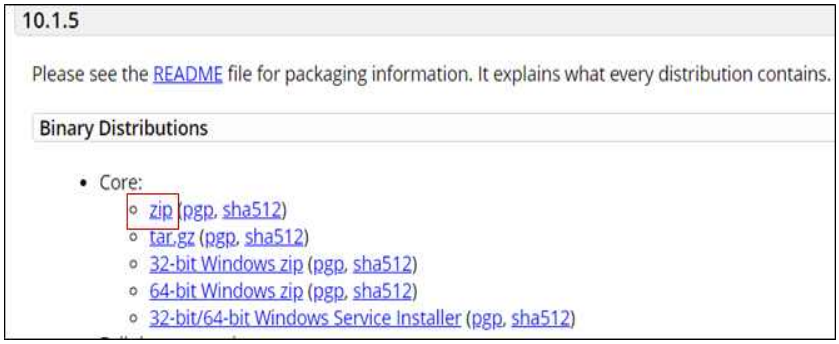
- 톰캣 : 웹컨테이너. 웹서버+웹컨테이너+웹애플리케이션 서버(WAS) 기능을 제공

가) 톰캣 다운로드

- http://tomcat.apache.org



- [Download] 에서 [Tomcat 9]또는 [Tomcat 10] 클릭
- [10.1.5]-[Binary Distributions]-[core]의 [zip]하이퍼링크 클릭 apache-tomcat-버전번호.zip 다운로드



나) 톰캣 설치

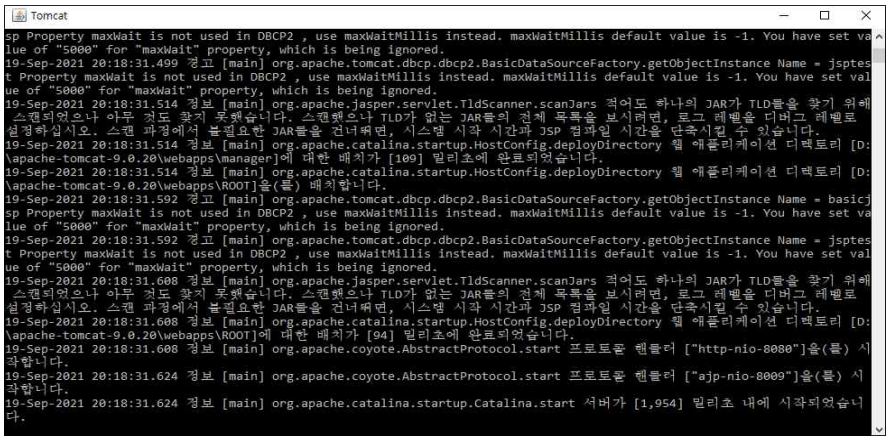
- 다운받은 apache-tomcat-버전번호.zip(예: apache-tomcat-10.1.5.zip)파일의 압축을 해제

다) 톰캣 환경변수 설정

- 변수명 : CATALINA_HOME
  변수값 : C:\apache-tomcat-10.1.5

라) 톰캣서버 실행 - 서버올리기/서버내리기

- 톰캣 홈(드라이브명:\apache-tomcat-톰캣 버전)의 하위 폴더인 [bin] 폴더에 있는 [startup.bat] 파일을 더블클릭. 예) 톰캣 홈(CATALINA_HOME) : C:\apache-tomcat-10.1.5



참고: 톰캣 서버가 올라오다가 내려간 경우 - 서비스 안됨

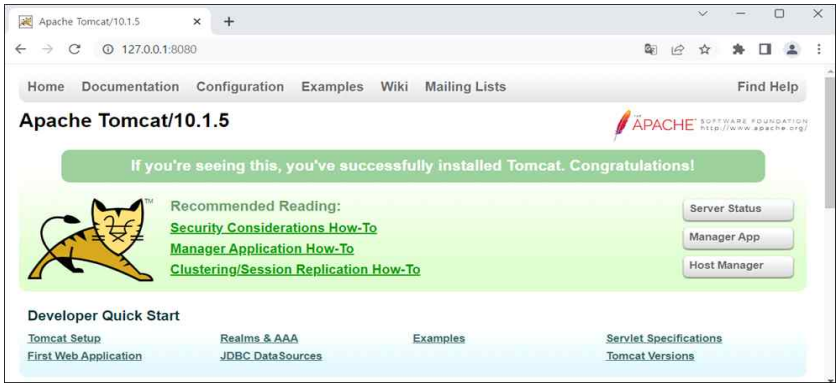
- 이유 : JAVA_HOME환경변수가 없거나 잘못된 경우
- 해결법: JAVA_HOME환경변수 경로를 맞게 수정

참고 : 톰캣 서버는 올라가나 서비스 안됨 - port충돌

- 이유: 오라클9i, 오라클클라이언트, icloud 서비스 등과 포트 점유 충돌
- 포트 충돌 확인 : netstat -nao 명령어로 cmd창에서 포트 점유확인. 서비스에서 pid옵션으로 충돌 프로그램 확인 후 서비스 중단

마) 톰캣 테스트 : 웹브라우저에 다음의 주소입력 후 엔터

- http://127.0.0.1:8080



마) 톰캣 서버 내림 - 서버내리기

- [bin] 폴더에 있는 [shutdown.bat] 파일을 더블클릭

4) 이클립스 다운로드 및 설치 : 구형버전

가) 이클립스 다운로드 : 통합개발환경(IDE)

- 통합개발환경 : 빌드자동화
  - 빌드 : 컴파일, 파일 배치를 수행해서 프로젝트를 프로그램으로 생성해 줌

빌드
프로젝트 ----------> 프로그램

- 톰캣, 오라클도 끌어다 사용 : 웹서버, DBMS제어
- [Eclipse IDE for Java EE Developers] 버전 설치 : 자바프로젝트, 웹 프로젝트, 모바일 웹 프로젝트, 하이브리드 앱 프로젝트 등의 다목적 사용이 가능.
- 여기서는 eclipse-jee-2022-12-R-win32-x86_64.zip 버전 사용

참고: 이클립스 버전별 설치파일 위치

https://www.eclipse.org/downloads/packages/release

나) 이클립스 설치

- 다운로드 받은 eclipse-jee-2022-12-R-win32-x86_64.zip 의 압축 해제

5) 이클립스 실행, 워크스페이스, 작업환경 설정

가) 이클립스 실행

- 탐색기에서 [eclipse] 폴더의 eclipse.exe 파일 또는 단축아이콘 더블클릭

나) 워크스페이스 지정

- c:\project
- 워크스페이스 : 프로젝트를 관리하는 폴더.
- 워크스페이스내의 모든 프로젝트의 공통 환경 설정이 저장됨.

- 프로젝트는 탐색기에서 폴더로 생성되며, 프로젝트 1개당 폴더가 1개 생성됨

다) 작업환경 설정 :
- [Window]-[Preferences] 메뉴
- 모든 프로젝트에 공통으로 적용되는 환경설정
- 작업환경은 워크스페이스에 저장
- 만일, 워크스페이스가 변경되면 작업환경을 다시 설정

(1) 한글 인코딩 지정
- [General]-[Workspace] 항목 선택
- 오른쪽에 표시되는 내용에서 [Text file encoding] 항목의 [Other]를 선택한 후 [UTF-8] 선택 후 [Apply] 버튼 클릭

(2) 글꼴의 크기 : 11~14사이 지정
- [General]-[Appearence]-[Colors and Fonts] 항목 선택
- [basic]-[text font]선택 후 [Edit]버튼 클릭
- 글꼴크키 : 11~14사이 지정 후 [OK] 버튼 클릭

6) 웹 애플리케이션 작성 :
- 서버 설정 후 동적 웹 프로젝트 작성
  - 동적 웹 프로젝트는 웹 사이트

가) 서버설정 : 톰캣을 이클립스에서 제어 - 톰캣을 이클립스로 끌어오기
- [File]- [New]-[Other] 선택
- [New]창에서 [Server]-[Server] 선택
- [Apache]-[Tomcat 10.1 Server] 선택

나) 동적 웹프로젝트 작성 : 웹사이트 만들기
- 동적 웹 프로젝트 1개당 웹사이트가 1개
- [File]-[New]-[Other] 선택
- [New]창에서 [Web]-[Dynamic Web project] 선택
- [Dynamic Web project] 창에서
- [Project name] : studyhtml5
- [Web Module]에서 [Generate web.xml deployment descriptor]항목 체크 후 [Finish]

참고) 이클립스에서 jeus연동 - 구글에서 "jeus 이클립스 연동"으로 검색어 입력, 티멕스사에서 제공 "FILE-20150618-000005_150618161418_1.pdf"참고

7) 동적 웹프로젝트를 서버에 등록

- [Servers]뷰에서 서버를 선택 후 마우스 버튼을 눌러[Add and Remove..]메뉴선택
- 등록할 프로젝트 선택후 [Add]버튼 클릭. [Finish]버튼 클릭

8) 서버실행 및 중단

- 서버실행: [Servers]뷰에 있는 [Tomcat v10.1~]항목을 선택 후 [start the server]아이콘 클릭
- 서버중단: [Servers]뷰에 있는 [Tomcat v10.1~]항목을 선택 후 [stop the server]아이콘 클릭

*중요: 로직수정([java resources]-[src]) 또는 DB수정한 경우 반드시 서버를 내렸다가 다시 올려야함.

9) HTML페이지, CSS, JSP페이지의 인코딩 변경: utf-8

- [Window]-[Preferences] 메뉴
- [Web]-[CSS]와 [Web]-[HTML], [Web]-[JSP]에서 [Ecoding]항목의 값을 [ISO 10646/Unicode(UTF-8)]로 지정 후 [Apply]버튼 클릭

참고 : 웹 프로젝트에서 파일 위치
- 웹프로젝트 : 로직소스파일, 뷰파일, 환경설정파일, 라이브러리

- 로직소스 파일 위치: [프로젝트]-[src]-[main]-[java]
  - 자바클래스파일, 서블릿(클래스파일) : .java
- 뷰파일(화면파일,웹페이지)과 기타필요파일 :
  - 뷰파일(화면파일,웹페이지) - html, jsp, css, js
  - 기타필요파일 - 이미지, 동영상파일, 음악...
  - 위치: [프로젝트]-[src]-[main]-[webapp]
- 환경설정 파일: 프로젝트의 환경설정 - 에러페이지제어, 커넥션풀제어..
  - web.xml : [프로젝트]-[src]-[main]-[webapp]-[WEB-INF]
- 라이브러리 :
  - 시스템제공 : jre, tomcat라이브러리
    [프로젝트]-[Java Resources]-[Libraries]
  - 추가로 넣어서 사용 : jdbc커넥터, 커넥션풀, 업로드, JSTL 라이브러리...
    [프로젝트]-[src]-[main]-[webapp]-[WEB-INF]-[lib]

10) 웹 페이지 템플릿 설정

- [Window]-[Preferences]메뉴를 선택한다.
- [Preferences]창이 표시되면 스크롤바를 내린 후 [Web]항목을 펼친 후 하위 항목인 [HTML Files]항목의 하위 항목인 [Editor]항목의 하위 항목인 [Templates]항목을 선택하면 오른쪽에 [Templates]의 내

용이 표시된다. 이때 표시되는 템플릿들 중에서 [New HTML file (5)]항목 선택 후 [Edit]버튼을 클릭.

- 4번째 줄의 <meta>태그 다음에 다음의 내용 추가
  - 제공되는 [source]폴더의 viewport.txt 파일 참고. 전자정부 권고안 뷰포트

```
---viewport.txt 내용---
<meta name="viewport"
      content="width=device-width,
               user-scalable=no,
               initial-scale=1.0,
               minimum-scale=1.0,
               maximum-scale=1.0"/>
```

11) 웹 페이지 작성 및 실행

- 웹페이지 작성위치: [프로젝트]-[src]-[main]-[webapp] 폴더
- 기본적으로 웹 페이지 1개가 화면 1개

가) html웹페이지 작성 : index.html

- 작성방법: [프로젝트]-[src]-[main]-[webapp]폴더에서 마우스 오른쪽 버튼을 클릭하고 [New]-[HTML File] 메뉴를 사용해서 index.html 작성
- 실행방법: 웹 페이지는 [Project Explorer]뷰에서 해당 웹 페이지에서 마우스 오른쪽 버튼을 클릭하고 [Run As]-[Run on Server] 메뉴 선택해서 실행

참고:

- 파일 또는 페이지를 실행 - 자바, jsp
- 프로젝트 단위로 실행 - 앱 : 안드로이드 앱, 하이브리드 앱

참고: 웹페이지의 기본 구성 [+ 서버사이드(jsp)]

- HTML : 문서의 구조와 내용 표시 <-태그 사용
- CSS : html태그를 표현 <-태그가 화면에 표시되는 모양을 지정
- js(JavaScript) : html태그를 클릭하면 어떤 작업 처리. 이벤트작업처리, 서버에 페이지 요청...

- html태그가 작업의 중심 : 태그에 디자인을 입히고, 태그에서 동작처리

12) 웹사이트 배포 - WAR

- 웹사이트 공유(기관서버로 올림)
- 자바기반의 프로젝트 배포
  - 자바프로젝트 : JAR file
  - 웹프로젝트 : WAR file
  - 안드로이드프로젝트 : APK file

가) 웹프로젝트를 WAR로 배포 : 웹 사이트
- WAR 파일의 목적: 실제 환경에서 서비스
- WAR 파일의 내용: 작성한 웹사이트전체를 가진 파일
- 주의사항 : 톰캣서버를 내린 후 한다.

(1) 작성방법

- 배포할 프로젝트에서 마우스오른쪽 버튼 클릭해서 [Export]-[WAR file]메뉴
- [WAR]창에서 [destination]항목에 경로(예:C:\apache-tomcat-10.1.5\webapps)를 지정하고, [Overwrite exist file]항목을 체크하고 [Finish]버튼 클릭

(2) 실환경에서 실행

- 톰캣홈\bin의 startup.bat실행
- 웹 브라우저에서 다음 페이지 실행. http://127.0.0.1:8080/studyhtml5

나) 이클립스 가상환경과 실제서비스 환경
- 프로젝트를 이클립스 가상환경에서 테스트한 후 개발자의 실제서비스 환경에 테스트한다. 이 테스트가 성공하면 기관의 서버로 올림(FTP사용해서 업로드).

(1) 이클립스 가상환경 : 이클립스에서 웹서비스환경을 제공
- [Project Explorer]뷰의 [Server]항목

(가) web.xml

- 웹서버가 관리는 모든 웹사이트에 공통적인 사항을 세팅. 예)특정확장자 파일을 열 것인지 다운할 것인지 결정
- cf)[프로젝트]-[src]-[main]-[webapp]-[WEB-INF]: web.xml
- 특정(1개)프로젝트의 환경설정 : 에러페이지 제어, 커넥션풀 사용, 커스텀태그사용 등등

(나) context.xml : 커넥션 풀 설정

- 커넥션 풀: 웹 페이지의 로딩 속도를 빠르게 함
- 탐색기에서는 워크스페이스(c:\project)의 [.meta]폴더
- 이클립스에서 프로젝트환경에 문제가 발생시- 워크스페이스를 새로 만들고 기존 프로젝트를 새 워크스페이스로 가져옴

(2) 실제서비스 환경

- 가상환경에서 만든 사이트를 실환경에서 어떻게 실행 : WAR파일

(가) 개발자 PC의 톰캣홈

- 탐색기에서 폴더로 존재(C:\apache-tomcat-10.1.5)
- 톰캣홈\conf폴더: web.xml, context.xml
- 웹서버가 관리는 모든 웹사이트에 공통적인 사항을 세팅
- [Project Explorer]뷰의 [Server]항목에서 하는 일과 같음

(나) 우리기관의 실제 서버의 홈

- 기관 서버의 홈

*중요: 이클립스 프로젝트의 실행에 문제가 발생하거나 잘되던 프로젝트에 갑자기 에러가 표시되는 이유.

- 원인 : 빌드시(컴파일하고 배치될 때) 오류 발생
- 해결 : 방법1 - [project]-[clean project]메뉴 사용
  방법2 - 이클립스 재기동

참고: HTML, CSS, JavaScript, SQL, Java, python의 문법과 라이브러리 참조 https://www.w3schools.com

3. HTML

가. HTML, CSS, JS(자바스크립트) 개요

1) HTML, CSS, JS로 분리된 이유

- HTML, CSS, JS는 분리해서 작업하면 유지보수가 좋음
- 내용과 디자인을 분리하지 않고 작성한 경우, 예) <p font="">
  - 내용(<p>)과 디자인(font="")이 혼재해서 유지보수가 불편. 이런 문제를 해결하고자 HTML5에는 내용과 디자인 분리

2) 웹 화면 작성순서

- HTML(화면내용구성) : HTML태그
- CSS(화면디자인) : CSS사용. 화면의 내용이 최적으로 보이도록
- JS(화면동작 기능): 자바스크립트. 사용자의 요구(요청)을 받기위해서
- 사용자 요청을 받고 처리 : 서버 영역 - 로직처리, DB연동

3) 웹페이지에서 HTML, CSS, JS가 하는 일

- HTML : 문서의 내용과 구조를 태그를 사용해서 기술
  - 스마트 기기 등을 지원, JS기능을 대체하는 HTML5태그 추가
- CSS : HTML태그를 표현(디자인).
  - 이미지 대체하는 CSS3 추가
- JS: HTML문서에서 발생하는 각종 처리.
  - HTML태그에서 발생한 이벤트(작업요청)처리
  - 서버에 요청(Ajax)
  - 플래시 대체 (캔버스(<canvas>) + JS)

참고: Windows운영체제에서 자신(서버)의 ip주소 확인

- [시작]-[프로그램 및 파일 검색]
- cmd 입력 후 엔터
- ipconfig 입력 후 Enter키
- 자신의 아이피 주소 확인

나. 주요html태그

- <HEAD>태그: 문서의 정보를 기술
- <BODY>태그: 문서의 내용을 기술

1) <meta> 태그

- html 문서에 대한 메타데이터를 정의하며 주로 페이지 설명, 키워드, 인코딩, 뷰포트, 검색 로봇 차단 등을 설정할 때 사용

가) 인코딩 설정

- <meta charset="인코딩">과 같이 charset속성에 인코딩을 지정. 인코딩은 구제표준 "UTF-8"권장. 예) <meta charset="UTF-8">

나) 뷰포트 설정

- 뷰포트 : 브라우저에 보이는 영역
- 작은 화면에서 컨텐츠가 자동 축소되는 것을 방지를 위한 뷰포트 설정이 필요.
- 뷰포트를 설정하면 PC브라우저와 모바일브라우저에서 내용이 제대로(원래크기로) 표시됨.
- 전자정부 권고안 뷰포트. viewport.txt파일에서 제공

```
<meta name="viewport"
      content="width=device-width,
                    user-scalable=no,
                    initial-scale=1.0,
                    minimum-scale=1.0,
                    maximum-scale=1.0"/>
```

2) <script> 태그

- 웹 페이지의 처리를 담당하는 코드인 자바스크립트를 정의.
- <script>와</script> 태그 사이에 자바스크립트 코드를 기술

예)

```
<head>
<script>
  function test(){ //자바스크립트 코드
      alert("test")
  }
</script>
</head>
```

- <script src="외부스크립트파일"></script>와 같이 외부스크립트 파일을 src속성으로 사용해서 html페이지로 끌어옴.
  - 외부스크립트 파일의 확장자는 .js
  - 예) <script src="main.js"></script>
- 실무에서는 </body>태그 바로 위에 작성 : 사이트의 성능고려

```
<html>
<head></head>
<body>
...내용들
<script src="main.js"></script>
</body>
</html>
```

3) <style> 태그

- 웹 페이지를 위한 스타일 정보를 정의하며 웹 브라우저에서 HTML 태그를 표현하는 방법을 기술.
- <style>과</style> 태그 사이에 스타일 시트 코드 기술

```
예)
<head>
<style>
    body { background-color : blue; } /*<body>태그의 배경색-파랑*/
</style>
</head>
```

- <link rel="stylesheet" href="외부스타일시트파일'>외부스타일시트를 href속성을 사용해서 html페이지로 끌어옴.
  - 외부스타일시트 파일의 확장자는 .css 예) <link rel="stylesheet" href="main.css">

```
<head>
  <link rel="stylesheet" href="main.css">
</head>
```

4) <body> 태그 : ch03/htmltags01~04.html

- html 문서의 본문의 시작과 종료를 알려주며, 웹 브라우저의 화면에 보이는 내용은 이곳에서 작성

가) 태그

- 문단과 문단 사이를 구분할 때 사용. 예) 태그 연습

나)
 태그

- 줄 바꿈을 하는 태그. 예) 리스트1

다) 글꼴 속성 태그 , : 가급적이면 CSS에서 지정

- , 태그는 글자를 강조할 때 사용: 태그 글꼴을 기울임, 태그 글꼴 진하게 표시. 예) 사물인터넷(IoT)

라) <img>태그

- 웹 페이지에 이미지를 삽입할 때 사용. 예) <img src="picture1.png" >그림1

마) 리스트태그 : ch03/htmltags05.html

- : 순서 없는 리스트. 헤더에서 네비게이션바(메뉴)를 작성할 때 사용.
  - : 순서 없는 리스트의 영역
  - : 리스트 목록

   낮잠 TV시청 	웹 브라우저에서 보이는 모양 ●낮잠 ●TV시청

바) <a> 태그

- 하이퍼링크(또는 링크)를 사용해서 이동할 페이지 또는 이동할 위치를 지정할 때 사용
- 다른 페이지로 이동: b.html로 이동
- 같은 페이지의 특정위치로 이동. 이때 id속성 값을 사용

```
<a href="#cnt1">cnt1로 이동</a>
….
<div id="cnt1">…</div>
```

- 순서 없는 리스트 + <a> : 메뉴(네비게이션) 작성시 사용

예)

```
<div id="menu">
  <ul>
   <li><a href="#">낮잠</a>
   <li><a href="#">TV시청</a>
  </ul>
</div>
```

웹 브라우저에서 보이는 모양

●낮잠

●TV시청

사) 테이블 관련 태그 : ch03/htmltags04.html

- PC브라우저 전용 웹페이지를 만들 때만 사용
  - <table> 태그 : 표를 작성하는 태그
  - <tr> 태그 : 표의 행을 정의하는 태그
  - <td> 태그 : 표의 열(셀)을 정의하는 태그

아) 태그

- html 문서에서 영역을 정의.
- 태그의 id 속성을 사용해서 영역 구분, id속성의 유일한 값은 각각의 영역을 유일하게 정의 해줌. 예)
- 웹 페이지의 문서 구역을 나누는 예시

```
<div id="header"> <!--문서의 헤더 영역-->
  <div id="nav"></div> <!--문서의 네비게이션 영역-->
</div>
<div id="content"></div> <!--문서의 내용 영역-->
<div id="footer"></div> <!--문서의 푸터 영역-->
```

- 문서 구역의 네비게이션 구역과 메뉴를 설정하는 예시

```
<div id="nav">
  <ul>
   <li><a href="#">낮잠</a>
   <li><a href="#">TV시청</a>
```

```
  </ul>
</div>
```

참고: ActiveX대체 기술 가이드라인 발췌
- 파일 처리 기술 및 그래픽/차트 표현 및 멀티미디어 재생 기술은 웹표준 기술로 구현이 가능하므로 HTML, CSS, 자바스크립트, DOM 등 웹 표준 기술을 이용하여 구현하기를 권고.
  - DOM(Document Object Model): 문서의 내용을 객체모델로 표현
- 구현예시
  - 다중 파일 업로드 : <input id="files-upload" type="file" multiple>
  - 그래픽/차트기술 : <svg>, <canvas>태그사용

5) HTML5문서구조 태그
- 문서의 내용과 구조를 기술하는 곳으로 문서 구조는 <header>, <nav>, <section>, <footer> 태그 등을 사용. 단, ie11미만 웹브라우저 사용자가 많은 경우, 문서구역을 나눌 때 태그 사용.

```
예)문서구역 - html5 문서구조태그
<header> <!--문서의 헤더 영역-->
   <nav></nav> <!--문서의 네비게이션 영역-->
</header>
<section></section> <!--문서의 내용 영역-->
<footer></footer> <!--문서의 푸터영역-->
```

6) 시멘틱 태그
가) 문서구조 태그 : documentArea.html
- <header> 태그 : 사이트의 소개나 내비게이션 등을 표시
- <nav> 태그 : 사이트의 내비게이션(메뉴) 항목을 표시
- <article > 태그 : 문서의 내용(글)을 표시하는 태그
- <section> 태그 : 문서의 내용을 표시하는 태그
- <aside> 태그 : 본문의 내용과는 독립적인 내용인 사이드바 콘텐츠를 표시
- <footer> 태그 : 사이트의 제작자, 주소, 저작권 정보 등을 주로 표시

7) 그 밖의 시멘틱태그 : htmltags_sy01~07.html
- <figure> 태그 : 설명글을 붙일 대상을 정의
- <figcaption> 태그 : <figure>태그로 정의한 대상에 설명글을 붙임. <figure>와 같이 사용

예) htmltags_sy03.html

```
<figure>
 <img src="picture1.png">
 <figcaption>나름 추상화</figcaption>
</figure>
```

참고: html주석 - <!--주석내용-->
-화면에 표시되지 않음. 단, html주석만 실행됨
참고: 어노테이션(annotation) - 일종의 주석
● 주석: 대상 - 사람. 어노테이션 : 대상 - 사람, 시스템. 자바계열: @

8) 웹 폼 태그 : htmltags_wf01~31.html
- 폼: 사용자에게 입력받는 화면
- 웹 폼 : 사용자에게 입력받는 화면을 가진 웹 페이지

가) HTML5에서 웹 폼을 구성하는 방법
- 모든 폼 태그는 원칙적으로 <form>과</form> 사이에 위치.
- Ajax 기능을 사용하는 경우, 프로그램을 코딩하는 방법에 따라서 <form> 태그를 생략해도 됨.
- 레이블이 없는 <input>, <select>와 같은 폼 태그에는 <label> 태그를 붙여서 작성
예)

```
<label for="name">이름</label>
<input type="text" id="name"/>
```

- 그룹화가 필요한 체크박스나 라디오 버튼 등은 <fieldset> 태그를 사용해서 그룹화
예)

```
<fieldset>
  <legend>취미</legend>
  <input type="checkbox" name="h1" id="h1"/>
  <label for="h1" >낮잠</label>
  <input type="checkbox" name="h2" id="h2"/>
  <label for="h2" >TV시청</label>
</fieldset>
```

나) 웹 폼 태그
- 사용자의 입력 등을 받을 때 사용하는 웹 폼을 구성할 때 사용.
● <input type="text"> : 텍스트 필드

```
<label for="id">아이디</label>
<input type="text" id="id" name="id" autofocus required
   placeholder="honhkd@kingdora.com"/><br>
<label for="name">이름</label>
<input type="text" id="name" name="name" required
   placeholder="홍길동"/>
```

- <input type="password"> : 패스워드 필드

```
<label for="id">아이디</label>
<input type="text" id="id" name="id" maxlength="20" size="30"
      pattern="[0-9A-Za-z@.]{6,20}" autofocus required/><br>
<label for="pass">비밀번호</label>
<input type="password" id="pass" name="pass" maxlength="12" size="30"
      pattern="[0-9A-Za-z#@$%^*]{6,12}" required/><br>
<input type="submit" value="전송">
```

- <input type="checkbox"> : 체크박스

```
<legend>취미</legend>
<input type="checkbox" name="h1" id="h1" value="a1" checked/>
<label for="h1" >낮잠</label>
<input type="checkbox" name="h1" id="h2" value="a2"/>
<label for="h2" >TV시청</label>
<input type="checkbox" name="h1" id="h3" value="a3"/>
<label for="h3" >만화보기</label><br>
<input type="submit" value="전송">
```

- <input type="radio"> : 라디오 버튼

```
<fieldset>
  <legend>성별</legend>
  <input type="radio" name="g" id="g" value="m" checked/>
  <label for="g">남</label>
  <input type="radio" name="g" id="g" value="f"/>
  <label for="g">여</label>
  <input type="submit" value="전송">
</fieldset>
```

- <input type="file"> : 파일 선택기

```
<form name="form1" action="#" method="post" enctype="multipart/form-data">
  <label for="id">파일선택:</label>
  <input type="file" id="file1" name="file1"/>
</form>
```

- <input type="submit"> : submit 버튼

```
<form action="submitPro.jsp">
  <label for="id">아이디</label>
  <input type="text" id="id" name="id" required/><br>
  <input type="submit" value="전송">
</form>
```

- <input type="reset"> : [reset] 버튼

```
<form>
  <label for="id">아이디</label>
  <input type="text" id="id" name="id" required/><br>
  <label for="pass">비밀번호</label>
  <input type="password" id="pass" name="pass" required/><br>
  <label for="name">이름</label>
  <input type="text" id="name" name="name" required/><br>
  <input type="reset" value="취소">
</form>
```

- <input type="button"> : [button] 버튼

```
<form>
  <input type="button" value="작업처리" onclick="alert('작업처리')">
</form>
```

- <input type="email"> : 이메일 필드

```
<form>
  <label for="email">이메일</label>
  <input type="email" id="email" name="email"/>
</form>
```

- <input type="url"> : URL 필드

```
<form>
  <label for="url">사이트 주소</label>
```

```
  <input type="url" id="url" name="url"/>
</form>
```

- <input type="tel"> : 전화번호 필드

```
<form>
  <label for="tel">전화번호</label>
  <input type="tel" id="tel" name="tel"/>
</form>
```

- <input type="number"> : 숫자 필드, ▲/▼ 눌러 선택

```
<form>
  <label for="num">나이</label>
  <input type="number" id="num" name="num" min="1" max="99" value="20"/>
</form>
```

- <input type="range"> : 숫자 필드, 슬라이드 막대를 사용해 숫자 선택

```
<label for="temp">온도</label>
  <input type="range" id="temp" name="temp" min="-20" max="70" value="20" step="1"/>
</form>
```

- <input type="date"> : 날짜 선택 필드

```
<form>
  <label for="day">오늘날짜</label>
  <input type="date" id="day" name="day"
       min="2020-01-01" max="2025-12-31" value="2021-12-30"/>
</form>
```

- <input type="time"> : 시간 선택 필드

```
<form>
  <label for="now">현재시간</label>
  <input type="time" id="now" name="now" value="09:00"/>
</form>
```

- <input type="week"> : 특정 연도의 주 선택 필드

```
<form>
  <label for="week">이번주</label>
  <input type="week" id="week" name="week"/>
</form>
```

- <input type="search"> : 검색 필드

```
<form>
  <label for="search">검색어</label>
  <input type="search" id="search" name="search" placeholder="검색어입력"/>
</form>
```

- <input type="color"> : 색상 선택 필드

```
<form>
  <label for="color">색상 선택</label>
  <input type="color" id="color" name="color"/>
</form>
```

다) select

- 여러 항목 중 1개를 선택할 때 사용하는 콤보상자를 만들 때 사용
- <select name="네임명" id="아이디명"><option value="항목값">화면에 표시되는 항목값</option>과 같이 <select> 태그는 name과 id속성을 가지며, <option>항목을 선택하면 value속성의 값은 name과 id속성의 값인 변수에 저장됨

예)

```
<select name="lang" id="lang">
    <option value="ko">한국어
    <option value="en" selected>영어
    <option value="ch">중국어
</select>
```

라) textarea

- 여러 줄을 입력하는 텍스트상자를 만들 때 사용
- 화면에 보이는 행의 수는 rows에 열 수는 cols에 저장. 지정한 행수나 열수를 넘으면 자동으로 스크롤바가 표시됨

예) <textarea name="lang" id="lang" rows="5" cols="30"></textarea>

마) button

- 버튼 생성 태그. 이벤트를 제어할 때 주로 사용. id속성의 값 필요

예) <button id="btn1">버튼1</button>

cf) <input type="button" id="btn1" value="버튼1">

9) 미디어 및 그래픽관련 태그 : htmltags_md01~06.html

가) 미디어 태그-멀티미디어 재생

- <audio> 태그 - 오디오 파일 재생. html5에서 제공
  - controls속성: 오디오 제어기

```
<audio controls>
  <source src = "play4.mp3" type ="audio/mpeg"/>
</audio>
```

- <video> 태그 - 비디오 재생. html5에서 제공
  - 화면을 갖기 때문에 width, height속성을 가지며, 비디오제어기 표시를 위해 controls 속성 사용

```
<video width="320" height="240" controls>
  <source src = "sea.mp4" type = "video/mp4">
</video>
```

- <embed> 태그 - 플러그인 콘텐츠 표시

```
<embed src="page_s.jpg" width="150" height="100">
<embed src="view.html" width="100" height="50">
<embed src="sea.mp4" width="200" height="150">
```

참고: 공공기관에서는 <iframe>대신 <embed>태그를 사용해서 유투브 동영상 연결
예) <embed src="//www.youtube.com/embed/StTqXEQ2l-Y" width="200" height="150">
  - 이유: <iframe>태그 안에 있는 if단어 때문.

- <object> 태그 - 내장 객체를 사용한 콘텐츠 표시
  - <object data="page_s.jpg" width="150" height="100"></object>
  - <object data="view.html" width="100" height="50"></object>
  - <object data="sea.mp4" width="200" height="150"></object>
  - <object data="initialization_parameter.pdf"></object>
- <iframe> 태그 - 유튜브 동영상 연결
  - <iframe width="560" height="315" src="https://www.youtube.com/embed/StTqXEQ2l-Y" frameborder="0" allowfullscreen></iframe>

나) 그래픽 태그 - 그래픽 영역 제공

- <canvas> 태그 : 게임을 만들 때 사용. 이벤트에 반응하는 그래픽영역
  - <canvas id = "canvas1" width="320" height="240"/>
- <svg> 태그 : 지도 표시 등에 사용
  - <svg width="300" height="100"><rect width="300" height="100" stroke="black" stroke-width="4" fill="gray"/></svg>

4. CSS

가. CSS의 기본 개요

- CSS는 캐스캐이딩 스타일시트(Cascading Style Sheets)의 약자로 HTML 태그를 화면상에 어떻게 표현할지를 설정

1) CSS파일 작성순서 :ch04/css_01~04

- 큰 범위에서 작은 범위

- 모든 태그에 적용되는 일반적인 규칙 정의 - 모든 화면크기
- 일부의 태그에만 적용되는 규칙 정의 - 모든 화면크기
  - 일반규칙에 추가할 규칙, 일부 변경할 규칙
- 미디어쿼리 규칙 정의 - 특정 화면크기에만 적용되는 규칙

2) CSS를 사용하는 이유

- CSS는 디자인, 레이아웃, 다양한 기기 및 화면크기에 따른 디스플레이의 다양성 문제를 포함한 웹페이지의 스타일을 정의할 때 사용

- CSS는 화면의 크기가 다양하더라도 최적의 화면을 제공
- 디자인을 별도의 파일로 작성해서 체계적인 개발
  - 내용과 디자인의 분리 : 유지보수가 편함

3) CSS의 작성 위치 :

가) 방법1 : 스타일 시트를 태그의 style 속성에 기술.

- ** 연습 **와 같이 태그안에 깃술
- 문제점: 같은 페이지에서 스타일을 재사용 못하고 필요한 태그에서 같은 코드를 또 기술

나) 방법2 : 스타일시트를 HTML 문서 내에 포함해서 작성

- **<style>div{width : 300px;}</style>**과 같이 <style>태그 안에 기술
- 문제점: 같은 페이지에서 스타일이 재사용 가능하나, 다른 페이지에 재사용이 안됨. 1개의 웹사이트는 수백개의 페이지로 이루어지기 때문에 웹사이트가 스타일을 유지하기 위해서는 매 페이지에서 재사용이 가능해야 함.

다) 방법3 : 스타일 시트를 별도의 파일로 작성

- 스타일시트를 여러 웹 페이지에서 재사용할 목적으로 별도의 파일로 저장해서 사용
- 스타일시트를 별도의 파일로 작성할 때 확장자는 .css
- CSS파일은 가급적이면 1개로 작성

예) **style.css**	예) html페이지에서 사용

@CHARSET "UTF-8"; body { /*body태그에 적용*/ background-color : #CAE2F7; } p { /*p태그에 적용*/ font-size: 16px; }	<link rel="stylesheet" href="style.css"/>

- 권장되는 글꼴의 치수 : rem
  - 상속관계에서 상대적 크기
  - 기본적으로 1rem= 1em = 16px = 100%
  - 1rem : 부모태그에서 지정한 크기와 같은 크기

예) CSS	예) HTML
div{ font-size : 1rem; /*16px*/} .d{font-size : 1rem; /*16px*/}	a b

4) CSS 작성방법

- 스타일시트를 구성하는 규칙 집합(rule set or rule)은 실렉터(selector)와 선언 블록({ })으로 구성
- 실렉터에는 태그명, id 속성 값, class 속성 값이 올 수 있음
  - * : 모든 태그
- 문법: 실렉터{규칙집합} 예) p{font-size:16px;}
- 선언 블록({ })안에는 선언들을 기술.
- 선언들은 속성과 값의 쌍으로 이루짐. 예) font-size:16px;
- 속성과 값은 콜론(:)으로 구분하며, 0개 이상의 선언을 나열할 수 있음.
- 선언과 선언 사이는 세미콜론(;)으로 구분

```
예) p{
        font-size: 16px;
        color: blue;
    }
```

- 두개이상의 실렉터에 같은 선언을 기술할 경우에는 실렉터를 콤마(,)를 사용해서 나열
  - div, p{font-size:16px;}
- /**/은 CSS의 주석. 예)/*주석입니다.*/

5) CSS실렉터

- 실렉터는 디자인을 적용할 태그를 선택하는 방법

- 실렉터에는 태그명 ,id속성 값, class속성 값, 의사엘리먼트, 조건 등을 사용할 수 있음

가) 태그명을 실렉터로 사용
- css가 적용되는 페이지의 같은 태그 모두에 적용
  예) 모든 태그에 적용

```
<div id="id1"></div>
<div id="id2" class="a"></div>
<div id="id3" class="a"></div>
```

```
div{/*적용: 모든 div태그*/
  color : red; /*글자색 빨강*/
}
```

나) class 속성 값을 실렉터로 사용
- css가 적용되는 페이지의 같은 class속성 값을 가진 모두에 적용
- 정의방법: .class속성값{}

```
예) class속성 값이 a인 모든 태그에 적용
  .a{ /*class속성값이 a인 태그. class="a"*/
     color : red; /*글자색 빨강*/
   }
```

다) id 속성 값을 실렉터로 사용
- css가 적용되는 페이지의 지정한 id 속성 값가진 1개의 태그에 적용
- 정의방법 : #id속성값

```
예)id속성값이 id3인 태그에만 적용
  #id3{ /*id속성 값이 id3인 태그. id="id3"*/
    color : red; /*글자색 빨강*/
  }
```

라) 의사 엘리먼트를 실렉터로 사용
- :hover 는 해당 태그에서 마우스 포인터가 위치하는 경우 적용되는 디자인
- 정의방법| 실렉터:hover{규칙집합}

```
예) CSS에서 작성 : main.css(ch04/css_main.html)
  div{
```

```
        width: 200px;
        height: 100px;
        background-color: blue;
    }

    div:hover{/*<div>태그에 마우스포인터가 들어오면*/
        box-shadow: 10px 10px 5px #888888;/*영역에 그림자 지정*/
    }
```

```
예) html
  <link rel="stylesheet" href="css/main.css">
  <div></div>
```

마) 조건을 실렉터로 사용

- 태그명[조건]과 같이 대괄호를 사용해서 표시. p[id]{ color:blue;}

예제) 다음의 표시1>로 표시되는 것을 표시2>로 표현 : css_menu.html

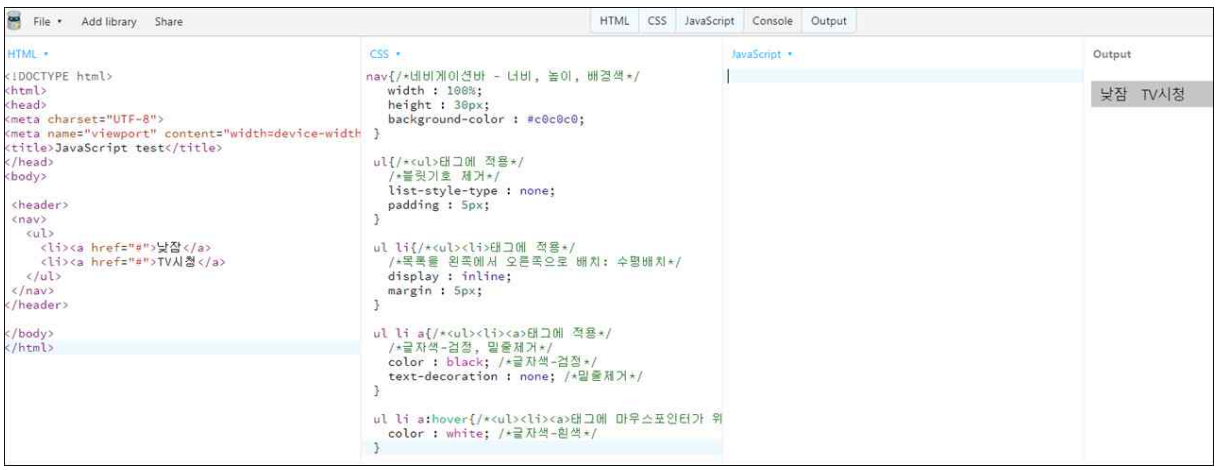
- HTML태그와 결과화면

<header> <nav> 낮잠 TV시청 </nav> </header>	표시1> 웹 브라우저에서 보이는 모양 ●낮잠 ●TV시청	표시2> 웹 브라우저에서 보이는 모양 낮잠 TV시청

- CSS코드

```
 nav{/*네비게이션바 - 너비, 높이, 배경색*/
    width : 100%;
    height : 30px;
    background-color : #c0c0c0;
 }

 ul{/*<ul>태그에 적용*/
    /*블릿기호 제거*/
```

```
    list-style-type : none;
    padding : 5px;
}

ul li{/*<ul><li>태그에 적용*/
    /*목록을 왼쪽에서 오른쪽으로 배치: 수평배치*/
    display : inline;
    margin : 5px;
}

ul li a{/*<ul><li><a>태그에 적용*/
    /*글자색-검정, 밑줄제거*/
    color : black; /*글자색-검정*/
    text-decoration : none; /*밑줄제거*/
}

ul li a:hover{/*<ul><li><a>태그에 마우스포인터가 위치하면*/
    color : white; /*글자색-흰색*/
}
```



css기본사용법: ch04/css_p01~10

나. 구역배치

- float, clear속성을 사용해서 각 구역을 화면크기에 맞게 자연스럽게 배치

1) float 속성 :css_p11

- 태그를 유동적(float)으로 배치
- 값 : left(왼쪽->오른쪽 배치) | right (오른쪽 ->왼쪽 배치)
예) float : left;

2) clear 속성: css_p12
- float 속성 해제
- clear : 해제 방향. 예) clear : both; 권장
  - 값 : left | right | both(양방향 해제)
  - 예)clear : both;

3) 반응형 웹 구현에 필수요소 - float 사용시
- float:left; 화면크기에 따른 각각의 구역배치
- @media screen~ 미디어쿼리: 각 화면 크기별 각 구역의 최적모양을 지정
- %, em, rem 유동치수:각 기준화면 사이의 간극을 처리

다. 반응형 웹
- 화면의 크기에 맞춰 컨텐츠를 자동으로 재배치
1) 미디어 쿼리 :css_m01
- 화면 크기를 인식. 디자인을 적용할 화면의 기준크기를 지정
- 기준크기에 따른 최적의 형태로 표현되도록 스타일 적용

가) 개요
- 목적: 최적의 형태로 표현
- 구현방법: 특정 디바이스의 능력이나 상황에 따라 특정한 CSS스타일을 적용할 수 있게 해 줌

나) 작성방법
- 실무에서 사용하는 기본형 :
  - @media screen and (조건){/*지정할 스타일*/}
- 조건: 화면사이즈 <-주로 지정하는 조건
  - ~이상 : min-width
  - ~이하 : max-width

  예)화면사이즈가 320px이상에서만 스타일 적용
  @media screen and (min-width : 320px){}
  @media screen and (min-width: 20rem) {}/*20rem=320px*/

다) 미디어 쿼리 구문 작성 위치
- 기존 CSS파일(.css)에 같이 기술
- 화면 크기를 모두 커버하는 CSS를 1개의 파일로 만들 경우 사용
- 로딩속도를 고려해서 가급적이면 CSS파일은 1개로 만듦. 권장
  - 예: @media screen and (min-width: 320px){}
- HTML페이지의 <link>태그에 기술
- 화면 크기별로 별도의 CSS파일을 만들어 사용할 경우 : 개별 파일을 각각 로딩해야해서 속도가 떨어짐.
  - 예) <link rel="stylesheet" type="text/css" href="theme.css" media="screen and (min-width: 320px)">
- 뷰포트의 너비 지정시 범위로 사용
  - min-width : ~이상. max-width : ~이하

(1) 브라우저 창의 크기를 줄이거나 늘릴 때 창의 크기화면에 표시

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $(window).resize(function(){
    var sizes = "<p>"+ window.innerWidth + "," + window.innerHeight+"</p>"
    $("#display").html(sizes);
  });
});
</script>
```

라) 미디어쿼리 레벨4
- 스크립팅, 포인터, 호버, 광도를 표현
(1) 스크립팅 기본 값 처리
- 스크립팅 기본 값은 스크립트가 동작되지 않게 하고, 자바스크립트가 페이지에서 실행되면 값이 대체됨.

기본 값 자바스크립트 실행 대체 값 <html class="no-js"> ----------> <html class="js"> 자바스크립트가 실행되는 코드로 자동 대체

- html주석문. 화면에는 표시되지 않지만 실행됨

예) <!-- [if IE 8]> <html class="no-js lt-ie9"> <![endif]-->

- 미디어쿼리에서 스크립팅	- 자바스크립트 실행 후
@media (scripting : none){} /*기본 값 : no-js*/	@media (scripting : enabled){} /*대체값 : js*/

(2) 포인터
- 입력형태에 따른 디자인
- pointer: none, coarse, fine
  - coarse: 손가락으로 터치하는 것과 같이 세밀하지 않은 제어
  - fine:마우스, 스타일러스 펜으로 터치하는 것과 같은 세밀한 제어

예) @media (pointer : coarse){}
@media (pointer : fine){}

(3) 호버 : 호버 사용 유무에 따른 디자인
- hover: none, on-demand, hover

예) @media (hover:none){}
@media (hover:on-demand){}
@media (hover){}

(4) 환경미디어 기능 - 주변광에 따른 디자인
- light-level : normal, dim, washed

예)@media (light-level:normal){}

2) 유동형 레이아웃
- 기준크기(브레이크 포인트)와 다음 기준크기사이의 간극을 처리해 줌
  - %, em, rem : 유동치수

마. 부트스트랩(Bootstrap)
- 부트스트랩(Bootstrap)은 HTML, CSS, JavaScript 반응형 웹사이트, 모바일우선 웹사이트를 쉽게 작성할 수 있도록 해주는 무료 프레임워크.
- 현재 최신버전은 부트스트랩5로 여기서는 이것을 사용해서 그리드기반의 디자인, 캐러셀, 모달 등을 쉽게 구현하는 방법을 제공.
- 상세한 사용방법은 https://www.w3schools.com/bootstrap5/index.php 을 참고.

1) 사용법
- 부트스트랩을 사용하려면 getbootstrap.com에서 부트스트랩을 다운로드 받아서 웹 애플리케이션 프로젝트에 추가하는 방법과 부트스트랩 CDN(Content Delivery Network)을 HTML페이지에 추가하는 방법이 있음. 여기선 jsDelivr가 제공하는 Bootstrap의 CSS와 JavaScript CDN을 사용한다. 아래의 CDN은 현재시점을 기준으로 한 최신버전으로 사용하는 시점에 따라 버전이 다를 수 있음.
- 부트스트랩5가 포함된 웹 페이지는 엣지나 크롬브라우저를 사용해서 실행.

Bootstrap CSS JavaScript 최신버전 CDN
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet"> <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>

- 부트스트랩을 웹페이지에 사용하는 방법.

① HTML5 기반의 문서 타입(<!DOCTYPE html>)을 사용모바일 우선 디자인(<meta name="viewport" content="width=device-width,initial-scale=1.0"/>)을 사용한다.

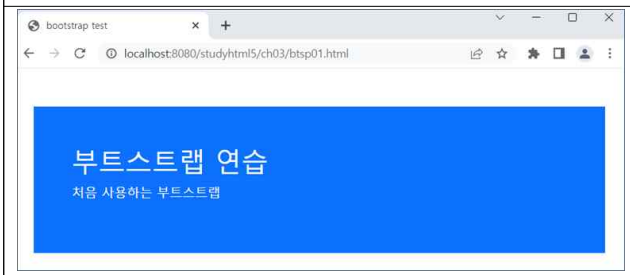
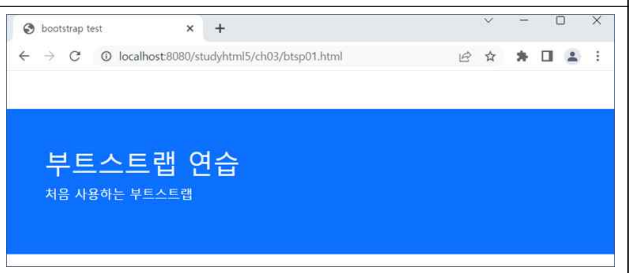
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0"/>
```

② Bootstrap의 CDN을 추가.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0"/>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
<title>bootstrap test</title>
</head>
```

③ 부트스트랩은 컨테이너의 사용해서 내용을 작성

- 사이트의 내용을 담는 컨테이너를 작성하고, 그 안에 웹페이지의 내용으로 기술될 태그들을 작성 컨테이너의 작성방법은 일반적으로 와 같이 태그에 class속성 값으로 "container", 또는 "container-fluid"를 사용.

 <img130.png>	 <img131.png>

"container"와 "container-fluid"의 차이점은 "container-fluid"는 항상 width(너비)를 100%로 사용하나 "container"은 화면의 너비에 따라 컨테이너의 max-width(최

대 너비)가 다르다.

	576px미만	576px이상	768px이상	992px이상	1200px이상	1400px이상
max-width	100%	540px	720px	960px	1140px	1320px

<"container"속성 값에서 화면 너비에 따른 컨테이너의 최대 너비>

참고: 웹 개발 라이브러리 제공 사이트
- 구글 개발자 사이트 : 구글에서 "구글개발자사이트"를 검색해서 찾음
  - 사이트- https://developers.google.com/
  - 블로그- https://developers-kr.googleblog.com/
- 네이버개발자 사이트
  - 사이트 - developers.naver.com

5. JavaScript

가. 자바스크립트의 개요

- 자바스크립트는 웹 페이지의 처리능력을 향상시킬 목적으로 개발한 스크립트 언어
- 사용분야: 웹사이트, 전자기기 제어, 데이터분석, 인공지능 프로그래밍

1) 자바스크립트 사용 방법

가) HTML페이지 내에 작성하는 방법

- <script>와</script>사이에 자바스크립트 코드 기술
- 문법:

```
<script>
  //자바스크립트 코드 기술
</script>
```

나) 별도의 자바스크립트 파일 작성 후 필요한 페이지에서 가져다 쓰는 방법(권장)

- 외부 자바스크립트 파일의 확장자는 js를 사용
- 문법: <script src="경로를 포함한 자바스크립트 파일명"></script>
- <script>태그의 위치 : <head>, <body>태그안
  - 사이트의 성능 향상: <body>태그의 마지막위치에 <script>태그 씀. 가급적이면 1개의 파일로 작성.

예) 실무에서 자바스크립트 사용 예

```
<html>
  <head></head>
  <body>
    <!--여기에 해당페이지의 내용 기술-->
```

```
    <script src="라이브러리 자바스크립트파일"></script>
    <script src="내가 만든 자바스크립트파일"></script>
  </body>
</html>
```

2) 자바스크립트의 문장 구조
- 유니코드(Unicode) 기반의 프로그래밍 언어
- 대소문자 구분
- 문장의 끝은 세미콜론(;)으로 구분
- 주석은 //, /**/을 사용

가) 자바스크립트의 리터럴(데이터의 값)
- 숫자 타입은 그냥 기술
- 문자열 타입은 큰따옴표("")또는 작은따옴표('')로 둘러싸서 표현
- 함수, 객체, 배열도 리터럴로 취급
  - 1, 'ㅁ', "a", function(){}, {id: "aaaa"}, [1, 2, 3]

나) 변수 선언
- var키워드를 사용해서 선언
- 선언방법 : var 변수명 = 리터럴;
  - 예) var var1 = 1; document.write(var1);

다) 식별자
- 식별자 : 변수명, 함수명, 클래스명(생성자 함수, 프로토타입), 메소드명, 프로퍼티(속성명) 이름
- 식별자의 첫 글자는 영문자, _, $를 사용할 수 있으며, 나머지 글자는 숫자도 사용.
- 변수명, 함수명 표기법: 첫글자 소문자, 단어가 바뀔 경우 다음 단어 첫 글자 대문자
- 클래스명, 생성자 함수명 표기법: 첫 글자를 대문자 사용, 단어가 바뀔 경우 다음 단어 첫 글자 대문자

  예) 변수명 : var1 = 5;
  함수명 : function testFunc(){}
  생성자 함수명 : function TestClass(){}

나. 데이터타입
- 프로그래밍은 데이터를 처리해서 원하는 결과를 얻어내는 과정.
- 데이터의 종류에 따라 처리하는 방법이 다름.
- 자바스크립트의 데이터타입에는 숫자, 문자, 부울, 객체, undefined, null, empty등이 있음

- 기본데이터 타입과 객체(참조, 레퍼런스)타입으로 나뉨
  - 기본데이터 타입 : 값 저장. 타입 종류 : 숫자, 문자, 부울, undefined

  예) var v1 = 5; //v1은 숫자 5를 저장하는 숫자변수
  - 객체(레퍼런스)타입 : 객체 주소 저장. 타입 종류 : 객체, 배열, 함수

  예) var v2 = {id:"aaaa", pass:"passwd"}; //v2는 객체의 주소를 저장하는 객체변수
- undefined, null, empty 값의 차이
  - undefined : 변수 등 타입이 정의되지 않음(값이 할당되지 않음)
  - null : 객체 레퍼런스변수가 가리키는 객체가 없음. 객체 초기화 등에 사용
  - empty : 문자열변수 등의 값이 비어있음(공백 값).

1) 기본 데이터 타입

가) 숫자 : (산술)연산

- 0~9, 부호(-), 소수점으로만 이루어짐. 예) 1000

나) 문자열

- 숫자, 문자로 이루진 데이터로 " "나 ' ' 로 둘러싸서 표현

예) "1000", "ㅁㅁㅁ", '1000', 'aaa', "she's gone"

다) 불린 : 부울 값

- true 또는 false 값을 가짐

2) 참조 데이터 타입 : 객체 타입

- 배열, 함수, 정규표현식도 객체로 간주됨

가) 자바스크립트 객체 : {}

- 문법 : var 객체변수명 = {};
- {}안에 프로퍼티(키:값의 쌍), 메소드 등을 기술.
- ,(쉼표)를 사용해서 나열

예) var obj1 = {id: "aaaa", pass: "pass"};

```
var obj1 = {id: "aaaa", pass: "pass"};
document.write(obj1.id);
document.write("<br>");
document.write(obj1.pass);
```

나) 배열 : []
- 문법 : var 배열변수명 = [];
- []안에 배열의 원소를 ,(쉼표)를 사용해서 나열
예) var arr1=[1, 2, 3];
var arr2 = new Array(3);

다) 함수 : function(){}
- 문법 : var 함수변수명 = function(인수리스트){//처리코드};
예) var f1 = function(){return 1;};

3) 변수 : 값 저장소
- 프로그램에서 필요한 값을 일시로 저장 : 메모리에 저장됨
- 참고로 파일, DB는 디스크에 영구저장 됨
가) 변수 선언
- var 키워드를 사용.
- 문법: var 변수명 = 리터럴 값;
  - 리터럴 값: 문자/숫자데이터, 객체(객체, 함수, 배열) 등이 옴
  - 숫자변수 선언 예: var v1 = 3
  - 문자변수 선언 예: var v2 = "aa"
  - 객체변수(객체 정의) 선언 예: var v3 = {id: "aaaa"};
  - 배열변수(배열정의, 권장) 선언 예: var v4 = ["a", "b"];
  - 빈 배열변수 선언(권장X) 예 : var v4 = new Array(2);
  - 원소 값을 갖는 배열변수 선언(권장X) 예 : var v4 = new Array("a", "b");
  - 함수변수(함수정의) 선언 예: var v5 = function(){return 1;}; function v5(){ return 1; }와 같음

```
//  js_05_05.html
var v1 = 3
var v2 ="aa"
var v3 = {id: "aaaa"};
var v4 = ["a", "b"];
var v4 = new Array("a", "b");
var v5 = function(){return 1;};
```

4) 객체 생성 방법
가) Object()생성자 함수 사용 : 권장X
- 예: var obj1 = new Object();

나) 객체 리터럴 사용 : 권장
- 문법 : var obj1 = {};
- 예) js_05_06.html
```
var obj1 = {id: 1234};
```
다) 생성자 함수 사용 :
- 객체 확장(상속)이 필요할 때 사용
- 문법: var obj1 = new 생성자함수();
예) js_05_07.html
```
var Person = function (name) { this.name = name; }; //생성자함수
var p1 = new Person("kingdora"); //객체생성 - p1객체 생성
```

5) 객체 사용(접근) : 프로퍼티나 메소드를 사용해서 접근
가) 프로퍼티를 사용한 객체 값에 접근
- 문법: 객체명.프로퍼티명 또는 객체명['프로퍼티명']
- 프로퍼티값 세팅 : 객체명.프로퍼티명 = 새값;
- 프로퍼티값 겟팅 : 객체명.프로퍼티명
- 새프로퍼티 생성 : 객체명.새프로퍼티명 = 값;

예) js_05_08.html
```
var objStudent ={name: "test", major: "computer science",
                 method1: function(){}};
document.write(objStudent.name);
objStudent.major = "electronics engineering";
objStudent.age = 22;
```

나) 메소드를 사용한 객체의 동작에 접근
- 객체명.메소드명()

예) document.write(objStudent.method1());

6) 객체내의 모든 프로퍼티를 반복해서 접근
- 방법: for(var 변수 in 객체)

예) js_05_09.html
```
for (var v1 in objStudent) {
    document.write(v1, " : ", objStudent[v1]);
    document.write("<br>");
}
```

7) 객체 프로퍼티 삭제 : delete연산자

- 방법: delete 객체.프로퍼티명

예) var objName = {name: "김왕쌍", subName : "kingdora"};

delete objName.subName;

```
// js_05_10.html
var objName = {name: "김왕쌍", subName : "kingdora"};
for (var v1 in objName) {
	document.write(v1, " : ", objName[v1]);
	document.write("<br>");
}

delete objName.subName;
document.write("<br>subName프로퍼티 제거<br>");
for (var v1 in objName) {
	document.write(v1, " : ", objName[v1]);
	document.write("<br>");
}
```

8) 객체비교

- 객체1과 객체2가 같은 객체를 가리키는 예 : js_05_11.html

```
var objName = {name: "김왕쌍", subName : "kingdora"};
var objName2 = objName
if(objName  == objName2){document.write("같은 객체");}
```

- 객체1과 객체2가 다른 객체를 가리키는 예 : js_05_12.html

```
var obj1 = { name: "yeontan"};
var obj2 = { name: "yeontan"};
if(obj1 == obj2){document.write("같은 객체");}
```

참고:Object객체 설명 및 메소드 설명 참조 사이트

-검색시: 구글에서 object method in javascript 로 검색

https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Object

다. 제어문 - 조건문, 반복문, 기타제어문

- 제어문: 프로그램의 실행 순서를 제어

- 조건문: 조건을 사용해서 프로그램 실행 순서 제어.
  - if문, switch문
- 반복문: 프로그램 실행을 반복문을 사용해서 제어.
  - for문, while문, do-while문
- 기타제어문 : 그 밖의 프로그램 실행 순서 제어.
  - 반복문을 탈출 또는 일시 탈출 : break문, continue문
  - 객체 생략 - with문, 예외처리 : try-catch-finally문

1) 조건문

가) if문

- 주어진 값에 따라 결과가 2개로 분기 시 사용
- 조건, 결과로 구성됨
  - 조건: ~이면
  - 결과가 2개: 참, 거짓
- 복잡한 조건식 구현: and, or연산자를 사용
  - 자바스크립트의 and, or연산자 : && - and연산자, || - or연산자
  - &&연산자 사용법 : 조건1 && 조건2 && ...
  - or연산자 사용법 : 조건1 || 조건2 || ....

참고: 엑셀 if()함수 =if(조건, 참, 거짓) and(조건1,조건2,...), or(조건1,조건2,...): 복잡한 조건식 표현 중첩if(): 결과의 수가 3개 이상일 때 사용, =if(조건, 참, if())

(1) 기본 if문

- 주어진 값에 따라 결과가 2개일 때 사용
- 기본형 : 조건을 만족할(참) 때와 조건을 만족하지 않을(거짓) 때 처리

```
if(조건){  // 조건을 만족할(참) 때
    // 참
}else{  // 조건을 만족하지 않을(거짓) 때
   // 거짓
}
```

```
//js_05_13.html
var i = 5;
var s = 10;

if(i < 10){
   s += i;
}else{
   s -= i
}

document.write("s변수의 값 : " + s +"<br/>");
```

- 조건을 만족할 때만 처리가 있는 형: else문을 사용하지 않음

```
if(조건){
   // 참
}
```

```
//js_05_14.html
var i = 5;
var s = 10;

if(i < 10){
   s += i;
}

document.write("s변수의 값 : " + s +"<br/>");
```

(2) 중첩if문

- 주어진 값에 따라 결과가 3개 이상일 때 사용

```
if(조건1){
   // 참1
 }else if(조건2){
   // 참2
 }else{
   // 거짓
}
```

```
//js_05_15.html
var x = 10;
var str1 = "";

if(x > 0){
   str1 = "양수";
}else if(x === 0){
   str1 = "0";
}else{
   str1 = "음수";
}

document.write(str1);
```

나) switch문 : 권장안함.

- switch문을 권장하지 않는 이유: C언어 계열(C/C++, 자바, 자바스크립트)의 문법을 사용하는 프로그래밍언어에서는 switch문은 조건식(표현식) 구현에 제한이 있음
- 조건식(표현식)의 결과가 정수만 가능

```
switch(표현식){
    case 값1:
      //처리
     break;
    case 값2:
      //처리
     break;
    default:
      //처리
     break;
}
```

다) 조건연산자

- 간단한 조건문 구현에 사용
- ?:연산자를 사용하며 문법은 (조건)? 참 : 거짓; 과 같이 구성됨

예) a값이 5보다 크면 v1값에 5를 넣고, 그렇지 않으면 v1에 a값 저장

```
var v1 = (a > 5)? 5 : a;
```

```
//js_05_16.html
var a = 10;

var v1 = (a > 5)? 5 : a;

document.write(v1);
```

2) 반복문
- 반복 작업 처리. 여러 건의 작업처리에 사용
- 횟수제어 반복문 구성에 필수요소 : 카운터 변수 선언과 초기화, 반복횟수제어 조건, 처리할 문장, 카운터 변수 누적
    - 카운터 변수 : 횟수 제어 변수

가) for문 : 횟수 반복
- 지정된 횟수만큼 반복수행
- 문법: for(var 카운터변수 선언과 초기화; 반복횟수제어 조건; 카운터변수 누적){//문장}

예) js_05_17.html
```
 for(var x=0; x<10; x++){document.write(x);}
```

나) for-in문 : 객체내의 프로퍼티반복
- 객체내의 프로퍼티 수만큼 반복 수행.
- 문법: for(var 변수 in 객체명){ //처리할 문장}
    - for in문 : 객체의 프로퍼티 반복
    - 객체반복 : 객체의 프로퍼티를 반복해서 처리

예) objName객체의 프로퍼티 출력 : js_05_18.html
```
var objName = {name: "김왕쌍", subName : "kingdora"};
for (var v1 in objName ) {
     document.write(v1, " : ", objName[v1]);
     document.write("<br>");
}
```

다) for-of문 : 객체 반복
    - 컬렉션반복 : 컬렉션내의 객체를 반복해서 처리. 컬렉션: 객체 모임
- 문법: for(let 변수 of 컬렉션){ //처리할 문장}

- for of문 : 객체의 반복
- 객체반복 : 컬렉션내의 객체를 반복해서 처리
- let문: 객체저장 변수에 값 할당

예) objName컬렉션의 객체 반복처리 : js_05_19.html, 로컬에서 실행 시 크롬브라우저 사용

```
var objName = [{name: "김태형", subName : "taetae"},
               {name: "김연탄", subName : "tan"}];
for(let obj1 of objName ) {
  for(var v1 in obj1){
    document.write(v1, " : ", obj1[v1]);
    document.write("<br>");
  }
}
```

라) while문

- while(조건)문은 조건을 만족하는 동안 반복수행
- 반복횟수를 알 수 없는 경우 선호했으나, for보다 수행속도 느림
- 문법:

```
카운터변수 선언과 초기화;
while(반복횟수제어 조건){
  //처리할 문장
  카운터 변수 누적;
}
```

마) do-while문

- 일단 1번 실행 후 조건비교 반복
- 문법:

```
카운터변수 선언과 초기화;
do{
  //처리할 문장
```

```
    카운터 변수 누적;
 }while(반복횟수제어 조건)
```

3)루프 탈출문

가) break문 : 반복문 완전탈출

- 무한루프 중단시 주로 사용
- 무한루프에서 break문 사용방법

```
 while(true){//무한루프
    if(조건) break;
 }
```

```
//js_05_20.html
var i;
var str = "";

for(i=0; i<10; i++){
   if(i==5) break;
   str += i + " ";
}

document.write("결과 : " + str);
```

나) continue문

- 일시적으로 탈출했다가 다시 반복문 수행

```
//js_05_21.html
var i;
var str = "";

for(i=0; i<10; i++){
   if(i==5) continue;
   str += i + " ";
}

document.write("결과 : " + str);
```

라. 함수

- 특정 로직 처리에 사용. 재사용 가능

- 프로그램 코드는 함수에 넣어서 사용
- 함수에는 시스템이 제공하는 함수와 사용자가 만드는 함수가 있음
  - 시스템이 제공하는 함수 : 내장함수
    공통적인 작업처리: 수학함수, 문자형 숫자를 숫자로 변환 등등..
  - 사용자가 만드는 함수 : 사용자정의 함수
    업무 로직 처리 : 급여계산, 게시판관리 등등

1) 함수원리
- 함수가 하는일 : 함수명
- 함수 인수의 개수와 타입 : 함수명(인수리스트)=> 인수리스트
- 함수의 실행결과 타입(리턴타입) : return문

예)function add(x, y) { return x + y;}
  - 함수가 하는 일 : 함수명 - add
    더하기, 합 계산
  - 함수 인수의 개수와 타입 : 인수리스트- 2개, 인수타입 - 둘 다 숫자
  - 함수의 실행결과 타입(리턴타입) : return x + y;
    숫자합의 결과타입: 숫자

2) 내장함수
- 시스템이 제공하는 함수
예) 메시지 상자 호출
alert(메시지내용);

3) 사용자 정의함수 작성방법
- 사용자 정의함수(업무처리 함수) 작성
가) 일반함수
- 기본적인 형태, 일반적인 사용의 사용자 정의 함수
- function문을 사용해서 함수 정의
- 일반함수는 함수 선언문으로 정의하는 방식과 함수표현식으로 정의하는 방식이 있음
  - 권장: 함수표현식으로 정의
(1) 함수선언문으로 함수정의
- function 함수명(매개변수리스트){} 와 같은 방식으로 선언
예) function add(x, y) { return x + y;}

```
//js_05_22.html
function add(x, y) { return x + y;}
document.write(add(5, 3));
```

(2) 함수표현식으로 함수정의
- var 함수명 = function(매개변수리스트){}; 과 같이 변수 선언문과 같은 방식으로 변수 선언. 권장방식
예) var add = function(x, y) { return x + y;};

```
//js_05_23.html
var add = function(x, y){ return x + y;};
document.write(add(5, 3));
```

- 권장 이유: 체계적인 프로그램 작성 - 후방참조 불가

(3) Function()생성자 함수를 사용해서 함수정의
- var 함수명 = new Function("매개변수리스트", "내용"); 와 같은 방식으로 함수 선언. 권장되지 않는 방식.
- new Function()은 쓰지 말 것이 권고 사항
예) var add = new Function("x", "y", "return x+y");

나) 익명함수 : 이름 없는 함수
- 사용이유 : 이름을 굳이 줄 필요가 없는 경우. 즉, 특정 경우에만 사용하고 제거
- 주 사용처 : 이벤트처리(콜백함수), 즉시실행 함수, 클로저, 함수표현식
- 작성방법 : function (){}
  - 이름이 없기 때문에 변수에 넣어서 사용하거나 함수의 인수로 사용

예) 익명함수를 만들고 v1변수에 저장
```
var v1 = function (){};
```
예) v1변수에 저장된 익명함수 실행
```
v1();
```

```
//js_05_24.html
var v1 = function(){return 10;};
document.write(v1());
```

(1) 콜백함수
- 이벤트에 반응하는 함수
- 이벤트처리 예시 : 자동 실행

```
image.onload = function(){  // 이미지가 로딩되면 자동실행 - 콜백함수
        ctx.drawImage(image,0,0);  // 캔버스의 (0,0)좌표에 이미지를 그린다.
}
현재 페이지의 로드가 끝나면 자동실행
$(document).ready(function(){ // 처리할작업});
```

(2) 즉시 실행함수(Self-Invoking Functions)

- 함수 정의와 동시에 실행. 호출 없이 자동으로 실행됨
- 선언방법 : (function() {//함수내용})();

예) (function() { var x = "Hello!!"; })();

- 함수가 선언 되자마자 실행되는 즉시 실행함수는 같은 함수를 다시 실행할 수 없음.
- 최초의 1번만 실행해야하는 초기화 등에 사용됨

```
//js_05_25.html
(function() {
  var x = "Hello!!";
  document.write(x);
})();
```

(3) 클로저

- 함수의 캡슐화, 런타임 실행시 인자 값 줘서 수행할 때 사용
  - 런타임 실행시 인자 값 줘서 수행하는 작업: 초/분/시 단위로 자동 동작되는 작업
- 함수 안에 리턴값으로 정의된 함수
- 보통 익명함수로 기술
- 사용이유: 변수의 스코프문제(보안문제)와 카운터 딜레마 문제 해결

예)
```
var func1 = function(){
    var x = 5;
    return function(){
      return x+=1;
    };
  }()
```

```
//js_05_26.html
var func1 = (function(){
  var x = 5;
```

```
        return function(){
          return x+=1;
        };
})();
document.write(func1());
```

2) 함수 사용(호출)법 :
- 함수명(인수리스트);와 같이 사용
예) func1()

마. 클로저
- 즉시실행 내부함수로 함수의 캡슐화, 런타임 실행시 인자값을 넘겨서 자동 동작되는 작업에 사용
- 참고:https://www.w3schools.com/js/js_function_closures.asp

1) 개요
- 함수 안에 리턴값으로 정의되는 익명 함수
예) var func1 = function(){ return function(){ return 1;};}();

2) 전역변수와 지역변수 스코프문제:
- 전역변수는 웹페이지내에서 긴 스코프를 가짐(해당 페이지가 서비스되는 동안 사용가능
```
var a = 4;
function myFunction() {
  return a * a; //전역변수
}
```
- 지역변수는 해당함수의 사용이 끝나면 사라짐
```
function myFunction() {
    var a = 4; //지역변수
    return a * a;
}
```
3) 카운터 딜레마
가) 카운터변수를 전역변수로 선언
- 카운트변수 처리가 쉬움
- 문제점: 전역변수에 접근 가능함. 보안문제 발생
- 해결방법 : 카운터 변수를 지역변수로 작성해야 함.

▪ 아래의 예시에서 add1()함수 안에 counter변수를 선언해서 지역변수로 사용해야함.

```
var counter = 0;
function add1() { return counter += 1;}
add1(); // counter = 1
add1(); // counter = 2
add1(); // counter = 3
```

```
//js_05_27.html
var counter = 0;

function add1(){ return counter += 1;}

document.write(add1(), "<br>");
document.write(add1(), "<br>");
document.write(add1(), "<br>");
document.write(counter)
```

나) 카운터변수를 지역변수로 선언

- 카운터가 동작 안 됨
- 문제점: 함수를 호출할 때마다 카운터가 0으로 초기화되어, 카운팅이 안됨.
  ▪ 아래의 예시와 같이 add1()함수를 호출할 때마다 var counter = 0; 이 실행되어 counter값이 누적되지 않음.
- 해결방법: 즉시실행함수와 내부함수 사용. 클로저를 사용해야 함.

```
function add1() {
    var counter = 0;
   return counter += 1;
}

add1(); // counter = 1
add1(); // counter = 1
add1(); // counter = 1
```

```
//js_05_28.html
function add1(){
  var counter = 0;
  return counter += 1;
}
```

```
document.write(add1(), "<br>");
document.write(add1(), "<br>");
document.write(add1(), "<br>");
```

다) 클로저를 사용해서 카운팅 딜레마 해결

- 즉시실행함수와 내부함수 결합
- 원리: 즉시실행함수는 단 1번만 실행되고 자식인 익명내부함수가 부모함수의 스코프에 접근.
  - 아래의 예시에서 add()1함수는 단 1번만 실행됨. var counter = 0;도 1번만 실행됨
  - add()1함수가 실행되고 있는 상태에서 호출시 익명내부함수가 리턴됨.
  - 익명내부함수가 부모함수인 add()1함수 안에 있는 counter변수에 접근해서 값 누적가능
  - 즉, counter변수는 익명함수에 의해 보호되고 add1()함수를 사용해서만 값을 변경할 수 있게 됨.

```
var add1 = (function () {
  var counter = 0;
  return function () {counter += 1; return counter}
})();

add1();// counter = 1
add1();// counter = 2
add1();// counter = 3
```

```
//js_05_29.html
var add1 = (function(){
  var counter = 0;
  return function(){counter += 1; return counter;}
})();

document.write(add1(), "<br>");
document.write(add1(), "<br>");
document.write(add1(), "<br>");
document.write(counter);
```

바. 객체

- 실무데이터 처리를 위해 사용
- 데이터를 객체로 저장하면 후 데이터를 원하는 위치까지 전달 가능

1) 객체 개요

- 객체 : 실무데이터 1건
- 속성(프로퍼티, 멤버필드) : 객체가 가진 값
- 메소드 : 객체가 하는 일

2) 자바스크립트에서 객체 생성 방법

(1) 객체생성

- var 객체변수명 = {속성: 값, 속성: 값, ...}; 로 객체 리터럴을 사용해서 생성

예)
```
var object1 = {
        number1: 1230,
        string1: "yeontan",
        boolean1: true,
        array1: [52, 273, 103, 32],
        method1: function(){}
  };
```

```
//js_05_30.html
var object1 = {
   number1: 1230,
   string1: "yeontan",
   boolean1: true,
   array1: [52, 273, 103, 32],
   method1: function(){return 1;}
};

for(var x1 in object1){
   document.write(x1, ":", object1[x1], "<br>");
}
document.write(object1.method1());
```

(2) 객체의 속성과 메소드 사용(접근)

- 객체의 속성 값 얻어냄: 객체명.속성명
  - object1.string1
- 객체의 속성 값 변경: 객체명.속성명 = 값
  - object1.number1 = "19951230"
- 객체명.메소드명()
  - object1.method1()

(3) 객체의 속성 반복 : for in

- 문법: for (var 속성저장변수 in 객체명){}
  - 객체명에서 속성1개를 뽑아서 속성저장변수에 넣음
  - var 속성저장변수 : 속성저장변수를 선언. 변수의 영역을 좁게 줘서 다 쓴 후에 메모리해제가 빨리됨.

예) for(var x1 in object1){}

3) 생성자 함수

- 생성자 함수는 객체 생성 틀
- 생성자가 하는일 : 프로퍼티값 겟팅/세팅

예) this.name = name;

- 자바스크립트에서 캡슐화 구현에 사용
- 프로퍼티과 메소드 정의시 this키워드 사용

예) this.name = name;

  - 프로퍼티 = 파라미터변수
  - this.name:프로퍼티, name:파라미터변수, 넘어오는 값

- 클래스이자 생성자 역할. 즉, 프로토타입 역할

예) 생성자 함수 선언

```
var PlayerCharacter = function(name){
    this.name = name;
};
```

```
//js_05_31.html
var PlayerCharacter = function(name){
    this.name = name;
};

var pc = new PlayerCharacter("warrior");
document.write(pc.name);
```

(1) 프로퍼티 정의 : this.프로퍼티명 = 값;

(2) 메소드 정의 : this.메소드명 = function(){}

예) this.getName = function(){return name;}

```
//js_05_32.html
var PlayerCharacter = function(name){
  this.name = name;
```

```
  this.getName = function(){return name;}
};

var pc = new PlayerCharacter("basic");
document.write(pc.getName(),"<br>");
```

(3) 프로토타입 밖에서 메소드 정의

- 방법: 프로토타입명.prototype.함수명= function(){ };

예) Student.prototype.getSum = function(){
    return this.korean + this.math;
  };

```
//js_05_33.html
function Student(name, korean, math) {
  this.name = name;
  this.korean = korean;
  this.math = math;
}
Student.prototype.getSum = function(){
  return this.korean + this.math;
};

var s1 = new Student("김연탄", 100, 90)
document.write(s1.name, " : ", s1.getSum());
```

사. 자바스크립트 프로그램 작성시 권고사항 - JS Best Practices

- 참고 https://www.w3schools.com/js/js_best_practices.asp

1) 전역변수는 가급적 사용 자제(쓰지 말것): Avoid Global Variables

2) 지역변수도 선언하고 사용: Always Declare Local Variables

- 선언: var a = 0;
- 사용: a = 5;

3) 먼저 선언하고 사용: Declarations on Top

- 선언: var a =5;
- 선언: var b;
- 사용: b = a + 1;

4) 변수는 사용 전에 초기화 - Initialize Variables

- 초기화: var a = 0;
- 초기화: var b ="";

5) 숫자, 문자, 불린 값은 객체형태로 쓰지 말 것: Never Declare Number, String, or Boolean Objects

- 성능 떨어짐.

6) new Object();문장은 쓰지 말것 - Don't Use new Object()

- new Object()대신 {} 사용 - Use {} instead of new Object()
  - var x1 = {}; // new object
- new String()대신 "" 사용 - Use "" instead of new String()
  - var x2 = ""; // new primitive string
- new Number()대신 0(숫자0) 사용 - Use 0 instead of new Number()
  - var x3 = 0; // new primitive number
- new Boolean()대신 false 사용 - Use false instead of new Boolean()
  - var x4 = false; // new primitive boolean
- new Array()대신 [] 사용 - Use [] instead of new Array()
  - var x5 = []; // new array object
- new RegExp()대신 /()/ 사용 - Use /()/ instead of new RegExp()
  - var x6 = /()/; // new regexp object
- new Function()대신 function(){}사용 -Use function (){} instead of new Function()
  - var x7 = function(){}; // new function object

7) 자동 형변환에 주의

- var x = "Hello"; //string. 문자타입
- x = 5; //number . 원래 문자타입을 잃고 숫자타입이 됨
- var a = "5" + 7 // string. 자동형 변환

아. 내장객체

1) 기본객체

-숫자(Number), 문자열(String), 배열(Array), 날짜(Date) 객체 제공

2) 문서 객체 모델(DOM)

- Attributes, Console, Document, Elements, Events, Event Objects, History, Location ,Navigator, Screen, Style, Window 등의 객체를 제공

- Window: 열린 브라우저 창에 대한 정보를 가진 객체
- Document : 브라우저에 로드된 문서에 대한 정보를 가진 객체

- Screen : 사용자의 스크린정보를 가진 객체
- History : 방문한 URL에 대한 정보를 가진 객체
- Location : 현재 URL에 대한 정보를 가진 객체
- Navigator: 브라우저에 대한 정보를 가진 객체

- 문서 객체에 대한 정보는 document객체의 메소드를 사용해서 얻어냄. 태그 접근에 사용.

가) 태그(엘리먼트)에 접근하는 메소드
- 리턴 값으로 DOMElement객체나 DOMElement객체타입의 배열을 리턴

(1) document.getElementById("id속성값")
- 주어진 "id속성값"을 가진 태그에 접근.
- 주로 이벤트처리에서 이벤트를 발생시킨 태그에 접근할 때 사용
  예)document.getElementById("id속성값")

(2) document.getElementsByTagName("태그명")
- 주어진 "태그명"을 가진 모든 태그 객체에 접근. 복수 개 태그를 배열로 리턴
- 주로 결과문서에 같은 이름의 태그들을 얻어낼 때 사용

나) 태그의 내용에 접근
- 태그객체.innerHTML속성 사용
- 방법: document.getElementById("id속성값").innerHTML

(1) 태그의 내용 얻어내기
- var v1 = document.getElementById("id속성값").innerHTML;

예) var v1 = document.getElementById("result2").innerHTML

(2) 태그에 내용 넣기
- document.getElementById("id속성값").innerHTML = "내용";

예) document.getElementById("result").innerHTML = " 연습 "

```
//js_05_34.html
<body>
  <div id="result"></div>
  <div id="result2">1111</div>

  <script>
    document.getElementById("result").innerHTML = "<p>연습</p>"
    var v1 = document.getElementById("result2").innerHTML
```

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

자. 예외처리

- 프로그램을 안정적으로 구현하기 위해서 사용
- 방법

```
try{
  //예외를 유발하는 코드
  …
}catch(exception){ // 예외발생시에만 실행됨
  // 예외발생시 처리할 코드
}finally{
  // 예외발생과 무관하게 반드시 실행할 코드 : 리소스해제
}
```

6. Ajax와 jQuery

- 자바스크립트를 사용한 비동기적 요청인 Ajax를 쉽게 사용할 수 있게 제공된 jQuery 라이브러리

가. Ajax의 개요 및 구성요소

1) Ajax의 개요

- Ajax라는 용어: 2005년 2월 18일 제임스 제시 개럿(Jesse James Garrett)이 자신의 블로그에 "Ajax : a new approach to web application"라는 제목의 글을 싣게 되면서 붙여지게 된 용어
- 구글(Google)의 실험적인 사이트인 구글 랩(Google Lab)에서 구글 서제스트(Google Suggest, 검색어 자동 완성기능으로 현재는 Autocomplete라 불림)나 구글 맵(Google Map, 구글 지도)과 같은 서비스를 제공
  - 구글 서제스트: 검색어 자동완성(Autocomplete). 요즘의 검색엔진들이 모두 갖추고 있는 기술인 동적인 검색방법
  - 찾는 단어의 일부를 입력하면 팝업상자로 관련단어들이 표시되는 기술

- 플래시(Flash)를 사용한 플러그인, 자바애플릿, 액티비브(ActiveX)등의 웹 브라우저 자체의 기능이 자바스크립트, DOM, 스타일시트 등의 웹 브라우저가 기본적으로 가지고 있는 기능들을 사용해도 가능.
- 이런 기술을 한마디로 정의할 용어: "Asynchronous JavaScript+XML"의 약어인 "Ajax(에이잭스)"

2) Ajax를 이루는 구성요소

- Ajax의 기능을 정의하기 위해서는 기본적인 구성요소
  - XHTML(현재는 HTML5)과 CSS를 사용한 표준기술기반의 웹 페이지
  - DOM을 사용한 동적인 화면표시와 상호작용
  - XML과 XSLT등을 사용한 데이터의 변경과 조작
  - XMLHttpRequest를 사용한 비동기적인 데이터 전송
  - 그리고 이것들을 결합해서 사용하는 자바스크립트

가) XHTML(HTML5)과 CSS를 사용한 표준기술기반의 웹 페이지

- 다양한 운영체제 및 웹 브라우저의 환경 하에서 Ajax가 같은 동작을 수행할 수 있기 위해서는 일정한 기준이 필요.
- Ajax에서는 W3C에서 규정한 HTML, CSS, Javascript 등의 표준기술을 기반으로 사이트를 작성하도록 권고.
- 표준기술을 준수한 Ajax로 작성한 사이트는 표준기술을 지원할 수 있는 웹 브라우저를 사용하면 같은 결과를 얻을 수 있음.

나) DOM을 사용한 동적인 화면표시와 상호작용

- 사용자의 동작에 대응해서 동적으로 웹 브라우저에 표시되는 결과를 변경할 수 있음.
- 웹 브라우저에 표시되는 내용의 변경: 자바 스크립트를 사용해서 DOM(Document Object Model, 문서 객체 모델)의 구조를 변경해 표현.
- DOM을 자바스크립트를 사용해서 웹 브라우저에 표시되는 내용을 변경한다는 것은 DynamicHTML란 용어로 불림.
  - Ajax에서 웹브라우저에 표시되는 내용을 변경한다는 것은 DynamicHTML을 사용한다는 의미.
- 웹 브라우저 측으로 보낼 수 있는 데이터: 표준화 데이터인 XML(Extensible Markup Language)또는 XML문서를 변경하는데 사용하는 XSLT(Extensible Stylesheet Language Transformation)를 사용.

다) XML과 XSLT등을 사용한 데이터의 변경과 조작

- HTML이 미리 정의한 특정 태그를 사용만 사용가능.
- XML은 사용자가 태그를 정의할 수 있음.
- 사용자가 태그를 정의하기 때문에 HTML보다 XML의 문법 준수규칙이 강함.

- HTML과 XML은 서로 보완적인 입장으로 사용 목적자체가 다름.
  - XML은 문서의 구조를 표현하는 것이 목적
  - HTML은 문서의 내용을 표시하는 것이 목적

라) XMLHttpRequest를 사용한 비동기적인 데이터 전송

- XMLHttpRequest객체는 HTTP(Hyper Text Transfer Protocol)를 사용한 서버와의 통신을 수행하기 위한 객체
- Ajax에서 데이터의 전송은 XMLHttpRequest객체를 사용해서 함.
- HTTP는 사용자의 요청을 HTTP는 인터넷상에서 HTML등을 송수신하기 위한 통신 프로토콜(절차).
- DOM을 사용한 웹 브라우저의 내용은 객체화되어 있음. 이것을 이용해서 비동기적으로 서버와의 통신을 수행할 수 있음.
- XMLHttpRequest객체는 원래 XML데이터의 전송을 목적으로 사용되는 객체이나 HTML이나 Text형식의 데이터도 사용가능.

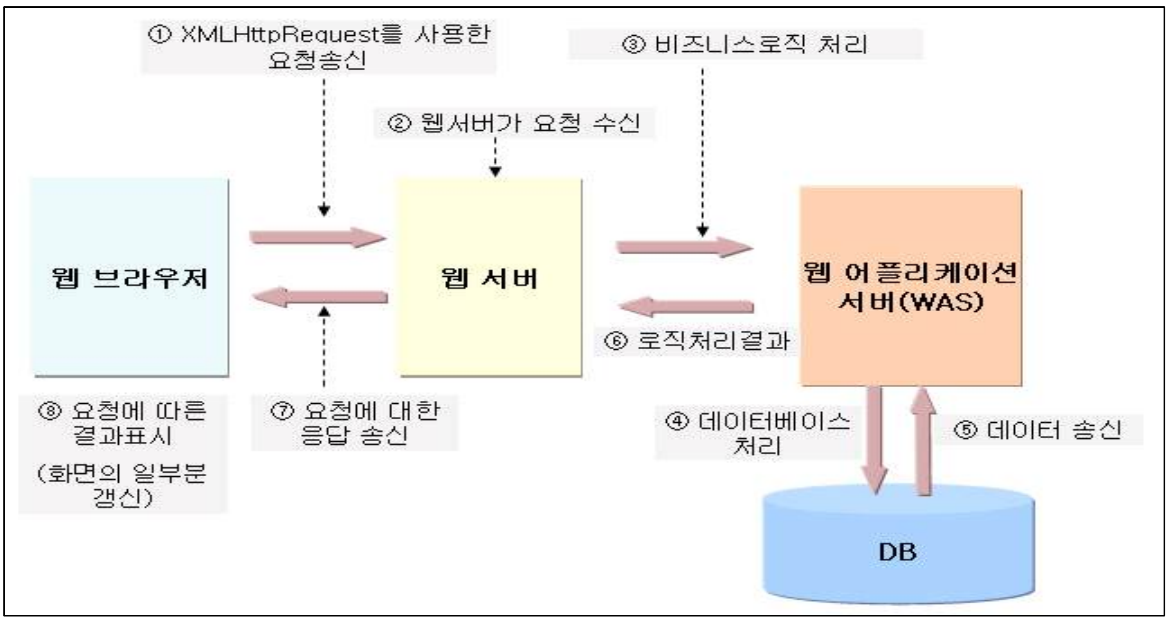
마) 그리고 이것들을 결합해서 사용하는 자바스크립트

- 자바스크립트는 Ajax를 사용한 사이트를 만들 때 기반이 되는 페이지
- 위의 구성요소들 사이의 동작을 제어 및 서버에 비동기적 요청을 하는 가장 중요한 부분을 담당.

3) Ajax의 비동기적 통신의 구조(데이터의 흐름)

- Ajax의 비동기적 통신의 특징
  - 웹 브라우저는 요청을 송신하면 응답을 기다리지 않음.
  - 서버는 필요한 데이터만을 응답.

- 동작방식



① 웹 브라우저에서 사용자의 동작에 의해 서버 측에 요청이 송신

- 이때 요청은 XMLHttpRequest객체에 의해 송신

② 일단 요청이 송신되면 웹 브라우저가 서버 측의 응답을 기다릴 필요 없이, 웹 브라우저 안의 작업을 처리할 수 있음.
③ 그 사이 서버 측에서는 송신 받은 요청을 처리
④⑤데이터베이스와 연동하는 작업 수행.
⑥ 처리가 끝나면 결과를 웹서버로 보냄
⑦ 웹서버는 클라이언트인 웹 브라우저로 처리 결과를 응답.
⑧ 이때 응답되는 페이지 전체를 새롭게 만들어 페이지 전체를 웹 브라우저에게 보내는 것이 아니라 필요한 데이터만을 보냄.

4) Ajax를 사용한 웹 페이지 작성
가) XMLHttpRequest 객체의 개요 및 생성
(1) XMLHttpRequest 객체의 개요
- XMLHttpRequest객체는 XHR객체로 지칭
- Ajax에서 서버와 웹 브라우저 간에 통신을 담당하는 가장 중요한 객체
- 웹 브라우저의 종류에 상관없이 XMLHttpRequest객체를 생성하도록 프로그래밍.

(2) XMLHttpRequest 객체 생성
- XMLHttpRequest객체는 서버와 클라이언트 간에 요청과 응답을 처리하기 위해서 필요한 객체. 이 작업을 처리하기 전에 반드시 생성
- 자바스크립트의 내장객체로 생성:
  - const xHttp= new XMLHttpRequest();
  - const 키워드: 이 키워드와 같이 선언된 변수는 재 정의되지 않고 재할당되지 않으며 블록스코프를 가짐

나) XMLHttpRequest객체의 메소드 및 프로퍼티
(1) XMLHttpRequest객체의 메소드
(가) abort()메소드 : void abort()
- 현재의 요청을 중단하는 메소드
  - XMLHttpRequest객체.abort();와 같은 방식으로 사용됨.

(나) getAllResponseHeaders()메소드 : String getAllResponseHeaders()
- HTTP요청에 대한 모든 응답헤더들을 키와 값의 쌍인 문자열로 리턴.
  - XMLHttpRequest객체.getAllResponseHeaders();와 같은 방식으로 사용.

(다) getResponseHeader() 메소드: String getResponseHeader(String headerName)
- 매개변수로 주어진 headerName에 해당하는 헤더의 값을 문자열로 리턴.
  - XMLHttpRequest객체.getResponseHeader();와 같은 방식으로 사용.

(라) open()메소드 : void open(String method, String url, boolean async, String userName, String password)

- 사용자의 요청을 설정하는 메소드.
- open(method, url, async, user, psw): method, url는 필수
  - method매개변수는 "GET", "POST", "HEAD", "PUT", "DELETE"중 하나의 값을 가짐
  - url매개변수는 상대 또는 절대경로를 포함한 파일명이 주어
  - async매개변수는 비동기로 전송시 "true", 동기적으로 전송할 경우 "false"지정. async를 생략하면 기본 값 "true".
  - userName매개변수는 사용자의 사용자명
  - password매개변수는 비밀번호 기술

예) XMLHttpRequest객체.open("GET", "test.xml", true);

(마) send()메소드 : void send(Object content)

- 사용자의 요청을 서버로 보냄
- 요청을 비동기적으로 설정하면 이 메소드는 바로 리턴되며, 동기적으로 설정한 경우에는 응답을 받을 때까지 기다림.
- 매개변수 content는 사용자 요청 설정
  - GET방식을 사용한 경우에는 null로 설정
  - xmlHttpObject.send(null);
  - POST방식으로 설정 시에는 DOM객체, 입력스트림, 문자열 등을 지정할 수 있음.
  - xmlHttpObject.send(str);

(바) setRequestHeader()메소드 : void setRequestHeader(String header, String value)

- 헤더의 값을 설정
  - header매개변수엔 값을 설정할 헤더명, value매개변수에는 설정할 헤더의 값 지정
  - xmlHttpObject.setRequestHeader("Content-Type", "application/x-www-form-urlencoded;");

(2) XMLHttpRequest객체의 프로퍼티

(가) onreadystatechange프로퍼티

- 상태의 변경이 발생했을 때, 이벤트를 처리하기 위한 이벤트 핸들러를 기술.
  - 이벤트 핸들러는 자바스크립트의 함수로 기술.
- 예) XMLHttpRequest객체.onreadystatechange = chageState;
  function chageState(){...}

(나) readyState프로퍼티

- 요청 객체의 상태를 리턴

리턴 값	상태	설명
0	uninitialized	open()메소드가 아직 호출되지 않은 상태

1	open	send()메소드가 아직 호출되지 않은 상태로 로드중
2	sent	send()메소드의 호출이 끝나서, 헤더와 status의 사용이 가능한 상태이며 로드완료 (IE에서는 아직 헤더와 status를 사용할 수 없다.)
3	receiving	일부데이터를 받을 수 있는 상태로, 처리중
4	loaded	모든 데이터를 받을 수 있는 상태로 완료

- XMLHttpRequest객체.readyState프로퍼티의 값이 4인가를 확인해서, 작업 처리

예) if(XMLHttpRequest객체.readyState == 4) { ....}

(다) responseText프로퍼티

- 문자열로 이루어진 서버의 응답을 받음.

(라) responseXML프로퍼티

- XML로 이루어진 서버의 응답을 받음.
  - 이 프로퍼티는 파싱이 가능한 DOM 노드 트리구조의 메소드와 프로퍼티를 XML 문서 객체로 리턴 받음.

(마) responseBody프로퍼티

- 이진코드문자열로 서버의 응답을 받음.
  - 이 프로퍼티는 XMLHttpRequest객체의 본래의 프로퍼티가 아니며, ActiveX 컴포넌트를 사용해서 생성한 XMLHttpRequest객체에서만 사용.

(바) status프로퍼티

- 서버로부터 응답받는 HTTP상태코드. 숫자로 리턴.
  - 200 : 정상(OK), 404: 요청한 페이지를 찾을 수 없음(Page Not Found), 500 : 서버에러(Internal Server Error)
  - 2xx번호는 성공을 4xx번호는 클라이언트의 오류, 5xx는 서버의 오류를 나타냄. 자세한 사항은 http://en.wikipedia.org/wiki/List_of_HTTP_status_codes 를 참조.

- xmlHttpObject.readyState프로퍼티의 값이 4이고, status프로퍼티 값이 200이면 정상적인 응답으로 간주해서 작업 처리

예) if(xmlHttpObject.readyState == 4){ if(xmlHttpObject.status == 200){} }

(사) statusText프로퍼티

- 서버로부터 HTTP상태를 문자열로 리턴.
  - "Not Found", "OK" 와 같은 문자열 리턴.

```
//xhrTest01.js
var xhrObject;

Event.observe( window, "load", function(){ startMethod();});

function createXHR(){ //XMLHttpRequest객체를 생성하는 메소드,
    if(window.XMLHttpRequest){
      xhrObject = new XMLHttpRequest();
```

```
    }else if(window.ActiveXObject){
        xhrObject = new ActiveXObject("Microsoft.XMLHTTP");
    }
}

function startMethod(){
        createXHR();
        xhrObject.onreadystatechange = resultProcess;
        xhrObject.open("GET", "xhrTest01.xml", "true");
        xhrObject.send(null);
}

function resultProcess(){
  if(xhrObject.readyState == 4){
     if(xhrObject.status == 200){
       $("resultArea").innerHTML = "서버로 부터 응답된 내용 : " +
xhrObject.responseText;
     }
  }
}
```

나. jQuery

- 자바스크립트 라이브러리 : 자바스크립트를 쉽고 간단하게 사용하는 방법을 제공
- 예) 태그를 자바스크립트에서 접근
  - 방법1 기본문법. document.getElementById("myCanvas");
  - 방법2 jQuery문법. $("#myCanvas");

1) 개요

- 빠르고 가볍고 다양한 기능을 가진 자바스크립트 라이브러리
- 제이쿼리 라이브러리가 제공하는 기능
  - HTML/DOM 작업
  - CSS 작업
  - HTML 이벤트 처리
  - 각종 효과 및 애니메이션
  - Ajax

- 각종 유틸리티 등

2) 사용방법
가) 방법1:
- http://jquery.com/ 사이트에서 최신 라이브러리를 다운로드 후
- [프로젝트]-[WebContent]-[js]에 라이브러리 배치
- 웹 페이지에 <script>코드 추가
  예)<script src="../js/jquery-3.1.0.js"></script>

나) 방법2:
- http://jquery.com/ 사이트에서 최신 CDN경로 복사
- 웹 페이지에 <script>코드 추가
예) <script src="https://code.jquery.com/jquery-3.1.0.js"></script>

3) 제이쿼리 기본 사용법
가) 기본 문법
- HTML 엘리먼트(태그)를 사용해서 어떤 동작을 수행
- 태그를 제어하기 위해서는 어떤 태그에(①) 어떤 동작이 발생했을 때(②) 어떻게 제어할지를 지정(③)
  - ①어떤 태그에 : 작업대상 태그
  - ②어떤 동작이 발생 : 이벤트 연결
  - ③어떻게 제어할지를 지정 : 이벤트 핸들러에 처리할 작업을 기술
- 문법: $(작업대상태그).어떤동작(function(){//어떻게 제어할지를 지정});
예)[로그인]버튼을 클릭하면 제어내용 처리 $("#login").click(function(){//제어내용});

(1) 작업 대상(태그, 객체) :
- 동작이 발생되는 태그 - 실렉터(selector)
- $(selector)
- 문법: $(selector).action()
  - $: 제이쿼리에서 태그, 객체 접근에 사용
  - (selector) :
  - selector - HTML 엘리먼트(태그명, class속성값, id속성값), 객체명(document, this)
(가) 태그명: $("태그명")
  예) js - $("p"), html- <p>
(나) class속성값 : $(".class속성값")
  예) js - $(".round"), html-

(다) id속성값: $("#id속성값")

예) js - $("#p1"). html- <p id="p1">

(라) 객체명: $(객체명)

예) $(document)

- document : 현재 html문서 자체를 의미하는 객체

(2) action() : 작업 대상에서 발생한 동작(이벤트) 처리()

예) hide()메소드는 작업대상을 숨김

- $(this).hide()
- $("p").hide()
- $(".test").hide()
- $("#test").hide()
- $(document).ready();

```
// ajax_06_01.html
$(document).ready(function(){
        $("#b1").click(function(){//[id="pic1"인 이미지 숨기기]버튼 클릭
                $("#pic1").hide();
        });

        $("#b2").click(function(){//[class="c1"인 이미지 숨기기]버튼 클릭
                $(".c1").hide();
        });

    $("#b3").click(function(){//[모든 이미지 숨기기]버튼 클릭
        $("img").hide();
        });

    $("#b4").click(function(){//[숨기기 취소]버튼 클릭
        $("img").show();
        });
})
```

나) 실렉터(selector)

- HTML 엘리먼트에 접근
- 실렉터는 $로 시작해 $()와 같은 형태로 사용
- ()안에는 엘리먼트명, 엘리먼트의 id 속성 값, class 속성 값, 객체명을 사용

- $(엘리먼트명/#엘리먼트의 id 속성 값/.엘리먼트의 class 속성 값)

다) 코드작성 방법

- 자바스크립트 코드는 $(document).ready(function(){//코드});안에 작성
  - 현재페이지(html)의 로딩이 끝나면 자동 실행

예) 현재페이지의 로드가 모두 끝나면 작업처리 :

```
$(document).ready(function(){ //현재페이지의 로드가 모두 끝나면
    $("button").click(function(){ //작업처리
        $("#displayArea").html("<img src='p2_s.png'/>");
    });
});
```

4) jQuery기술방법

- 반드시 $(document).ready(function(){}); 안에 제이쿼리 프로그램을 코딩

$(document).ready() : 현재 페이지의 로딩이 끝나서 사용가능한 상태

- 문법: $(document).ready(function(){// jQuery 메소드를 기술 });

ajax_06_02.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,initial-scale=1.0"/>
<title>jquery test</title>
<style>
  div#displayArea{
    width  : 200px;
    height : 200px;
    border : 5px double #6699FF;
  }
</style>

</head>
<body>
  <div id="displayArea">이곳의 내용 변경 </div>
  <button>표시</button>
```

```
  <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
  <script>
  $(document).ready(function(){
        $("button").click(function(){
          $("#displayArea").html("<img src='https://github.com/pdeveda/webbook/raw/master/images/2.jpg?raw=true' border='0' />");
        });
  });
  </script>
</body>
</html>
```

5) HTML 엘리먼트 내용에 접근하기 - get/set

- 엘리먼트의 내용에 접근하려면 text(), html(), val() 메소드 중 하나를 사용
- text() : 선택한 엘리먼트의 내용을 텍스트형태로 지정하거나 얻어냄.
  예) $("p").text(); , $("p").text("작업");
- html() : 선택한 엘리먼트의 내용을 HTML태그를 포함하여 지정하거나 얻어냄.
  예) $("p").html(); , $("p").html("1");
- val() : 폼 필드의 값을 지정하거나 얻어냄. 예) $("#id").val();

가) 엘리먼트의 내용을 얻어낼 때

- [엘리먼트.메소드()]와 같은 형태로 text(), html(), val() 메소드 사용

예) 테스트

```
$("p").text();
```

ajax_06_03.html

```
<style>
  div#result{
    width  : 200px;
    height : 100px;
    border : 5px double #6699FF;
  }
</style>

</head>
<body>
```

```
  <p>1010</p>
  <button id="process">처리</button>
  <div id="result"></div>
  <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
  <script>
    $(document).ready(function(){
        $("#process").click(function(){
                $("#result").text($("p").text());
        });
    });
  </script>
</body>
```

나) 엘리먼트의 내용을 변경할 때

- [엘리먼트.메소드("변경할 내용");]과 같은 형태로 text(), html(), val() 메소드 사용

예) 테스트

```
$("p").text("작업");
```

ajax_06_04.html

```
<style>
  p{
    width  : 200px;
    height : 50px;
    background-color : yellow;
    text-decoration-line: overline underline;
    text-decoration-style: wavy;
  }
</style>

<body>
  <p>마우스 포인터를 여기에!!!</p>
  <button>더블클릭하시구려.</button>
  <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
  <script>
    $(document).ready(function(){
                $("p").mouseenter(function(){
```

```
                $(this).text("왔구려, 마우스포인터!!!");
            });

            $("p").mouseleave(function(){
                $(this).text("돌아와 마우스포인터!!!");
              });

            $("button").dblclick(function(){
                $(this).css("background-color","purple");
            });
        });
  </script>
</body>
```

- 폼필드에서 값 얻어내기: var query = {id : $("#id").val(), passwd:$("#passwd").val()};
- 폼필드에서 값 수정 : $("#passwd").val("");

ajax_06_05.html

```
<style>
  div{
    width  : 400px;
    height : 200px;
    border : 5px double #6699FF;
  }
  ul li{
    list-style:none;
  }
</style>

<body>
  <div id="status">
     <ul>
        <li><label for="id">아이디</label>
            <input id="id" name="id" type="email" size="20"
              maxlength="50" placeholder="example@kings.com">
        <li><label for="passwd">비밀번호</label>
```

```
            <input id="passwd" name="passwd" type="password"
              size="20" placeholder="6~16자 숫자/문자" maxlength="16">
        <li class="label2">
            <button id="login">로그인</button>
    </ul>
</div>
<div id="result"></div>

<script src="https://code.jquery.com/jquery-3.1.0.js"></script>
<script>
  $(document).ready(function(){
            $("#login").click(function(){
                var query = {id : $("#id").val(),
                                passwd:$("#passwd").val()};
                $("#result").text(query.id + ", " + query.passwd);
                $("#id").val("");
                passwd:$("#passwd").val("");
            });
  });

</script>
```

```
$(document).ready(function(){
        $("#checkId").click(function(){
          if($("#id").val()){
                var query = {id:$("#id").val()};
            $.ajax({
                type:"post",//요청방식
                url:"confirmId.jsp",//요청페이지
                data:query,//파라미터
                success:function(data){//요청페이지 처리에 성공시
                        if(data == 1){//사용할 수 없는 아이디
                                alert("사용할 수 없는 아이디");
                        $("#id").val("");
                    }else if(data == -1)//사용할 수 있는 아이디
```

```
                        alert("사용할 수 있는 아이디");
                    }
                });
            }else{//아이디를 입력하지 않고 [ID중복확인]버튼을 클릭한 경우
                alert("사용할 아이디를 입력");
                $("#id").focus();
            }
        });
        $("#process").click(function(){
            checkIt(); //입력폼에 입력한 상황 체크

            if(status){
                var query = {id:$("#id").val(),
                             passwd:$("#passwd").val(),
                             name:$("#name").val(),
                             address:$("#address").val(),
                             tel:$("#tel").val()};
                $.ajax({
                    type:"post",
                    url:"registerPro.jsp",
                    data:query,
                    success:function(data){
                        //window.location.href("main.jsp");
                        $("#main_auth").load("loginForm.jsp");
                        }
                });
            }
        });
        //[취소]버튼을 클릭하면 자동실행
        $("#cancle").click(function(){
                //window.location.href("main.jsp");
                $("#main_auth").load("loginForm.jsp");
        });
});
```

6) 제이쿼리 이벤트 연결방법
- 참고: https://www.w3schools.com/jquery/jquery_ref_events.asp
가) 방법1:
- 이벤트발생태그.이벤트종류(function () {});
  예)$('h1').click(function () {});
나) 방법2:
- 이벤트발생태그.on(이벤트종류,function () {});
  예)$('h1').on('click', function () {});
다) 방법3:
- 이벤트발생태그.on(이벤트종류:function () {}; 이벤트종류:function () {};...);
  예)$('h1').on({ //다양한 이벤트 연결이 가능
        click : function () {}; //click이벤트
        mousedown : function () {}; //mousedown이벤트
    });
라) on(), off()메소드를 사용한 이벤트 연결과 제거
(1) on()메소드
- 이벤트 연결 . 3.0버전부터는 bind()대신 on()사용
  - 아직 생성되지 않은 자식엘리먼트(child element)에 이벤트를 연결할 수 있음.
  - 1개의 엘리먼트에 다양한 이벤트 연결 가능
- 문법:$(selector).on(event,childSelector,data,function,map)
  예)<p>태그에서 click이벤트가 발생하면 자동 실행
    $("p").on("click", function(){
      alert("The paragraph was clicked.");
    });
(2) off()메소드
- 이벤트 제거
- 문법:$(selector).off()
  예) $("p").off("click");

참고: 마우스 이벤트에서 왼쪽/오른쪽 버튼 구분
- jQuery에 event.which를 사용해서 구분 : 값이 1이면 왼쪽버튼, 3이면 오른쪽버튼
```
$("#element").mousedown(function(event) {
    switch (event.which) {
        case 1:
            alert('왼쪽마우스버튼 클릭.');
```

```
            break;
        case 2:
            alert(가운데마우스버튼 클릭');
            break;
        case 3:
            alert('오른쪽마우스버튼 클릭.');
            break;
        default:
            alert('해당 값 없음!');
    }
});
```

7) jQuery에서 서버사이드 요청

가) 제이쿼리 Ajax

- jQuery에서는 Ajax기능을 구현한 메소드들을 제공
- 서버로부터 TEXT, HTML, XML 또는 JSON 형태의 파일을 요청하고 응답받을 수 있는 기능을 제공
- 간단한 코드만을 사용해서 Ajax 기능 구현

나) Ajax 관련 메소드 중 서버 요청과 관련된 메소드

(1) .load()

- 서버로 데이터를 요청하고 HTML 엘리먼트에 응답받은 결과를 로드함(넣음).

예)
```
$("#div1").load(function()(responseText, textStatus, XMLHttpRequest){
    if(textStatus="success") { //성공시 처리할 작업}
    if(textStatus="error") { //실패시 처리할 작업}});
```

(2) $.get() : 서버로 HTTP get 방식의 요청을 함.

(3) $.post() : 서버로 HTTP post방식의 요청을 함.

(4) $.getJSON() : HTTP get 방식을 사용해서 JSON데이터를 요청함.

(5) $.ajax() : 비동기 Ajax요청을 수행함. get, post 방식을 지정해서 사용

예)
```
var query = {//프로퍼티}; //JS객체
$.ajax({
    type: "POST", //전송방식
    url: "process.jsp", //요청페이지
    data: query, //전송데이터
    success: function(data){ //요청페이지를 실행한 결과
        $("#result").html(data);
```

```
    }
});
```

다) 주로 사용하는 Ajax서버 요청 메소드

(1) .load()

- 서버로 데이터를 요청하고 HTML엘리먼트에 응답받은 결과를 로드.
- 응답 받은 결과를 화면에 표시해야 하는 로그인폼, 회원가입 폼, 글 목록 등을 실행할 때 주로 사용

예) $("#result").load("ajax/test.html");

(2) $.ajax()

- 비동기 Ajax요청을 수행함. get, post방식을 지정해서 사용.
- 주로 로그인 처리, 회원가입처리, 글쓰기처리등과 같이 DB와 연동 후 처리 결과만을 반환하는 경우에 사용

예)
```
$.ajax({
        type: "POST",
        url: "loginPro.jsp",
        data: query,
        success: function(data){}
    });
```

7. 그 밖의 주요 라이브러리 및 프레임워크

가. 라이브러리

1) node.js개요

가) 특징

- 푸시서버개발에 주로 사용됨.
- cpu를 많이 사용하지 않는 가벼운 작업 위주로 개발하는 것이 좋음.
- 자바스크립트가 다른 언어에 비해 가독성이 떨어지는 문제점으로 유지보수가 어려울 수 있음. 콜백 중첩
- V8엔진의 문제점 : 가베지 컬렉션기반의 메모리 관리로 CPU사용률이 정점을 치면 서버가 순간적으로 멈출 수 있음.

나) 장단점

(1) 장점 :

- 서버사이드의 진입장벽이 낮음
- socket.io를 사용한 서버와 클라이언트 양방향 통신이 쉬움. push구조 구현에 좋음

(2) 단점 :

- 단일 쓰레드 모델이기 때문에 하나의 작업에 시간이 많이 걸리면 전체 시스템의 성능이 떨어짐
- 데이터베이스 트랜잭션이 많은 애플리케이션의 경우, 기존의 WAS시스템보다 성능이 떨어짐

2) node.js설치
- nodejs.org에서 권장(recommanded)버전 다운로드받아 설치

3) node.js를 이클립스에서 세팅 :
가) nodeclipse플러그인 설치
- [Help]-[Eclipse Marketplace]
- node를 입력하고 검색 후 nodeclipse 찾음
- nodeclipse에서 [install]버튼 클릭
- 라이선스에 동의하고 [Next]버튼,[restart] 버튼 등을 눌러 설치

나) node 프로젝트 작성
- node 퍼스팩티브로 전환
- [File]-[New]-[node project]

나. 프레임워크
1) AngularJS
- "Angular.js" 또는 "AngularJS 1.X"는 자바스크립트 기반의 오픈 소스 프론트엔드 웹 애플리케이션 프레임워크
- 리치 인터넷 애플리케이션에 공통적으로 사용되는 구성 요소들과 더불어 클라이언트 사이드의 모델-뷰-컨트롤러(MVC)와 모델-뷰-뷰모델(MVVM) 구조를 위한 프레임워크를 제공
    - 애플리케이션들의 개발 및 테스트를 단순화하는 것이 목적

2) React와 Flux : 페이스북에서 개발
- React : 프런트엔드 라이브러리
- Flux : 애플리케이션 아키텍쳐
- React의 등장으로 고도화된 애플리케이션을 jQuery로 만들 필요 없이 쉽게 구조화할 수 있음.

3) Vue.js
- 참조: https://kr.vuejs.org/v2/guide/index.html
- 사용자 인터페이스를 만들기 위한 진보적인 프레임워크.
- 다른 단일형 프레임워크와 달리 Vue는 필요한 시점에서 필요한 라이브러리를 추가해서 기능 조합가능(점진적으로 채택할 수 있도록 설계).
- 핵심인 본체 라이브러리는 뷰 레이어만 초점을 맞추어 다른 라이브러리나 기존 프로젝트와의 통합이 매우 쉬움

참고: 서버사이드

1. JSP기본문법

가.JSP페이지의 구성요소 - 디렉티브, 선언문, 스크립트릿, 표현식

- jsp웹페이지를 구성하는 가장 기본적인 구성요소.

1) 디렉티브

- 클라이언트가 요청한 JSP 페이지가 실행될 때, 필요한 정보를 지정.
- 필요한 정보를 JSP 컨테이너에게 알려서 어떻게 처리되도록 하는 지시자.
- 디렉티브는 태그 안에서 @로 시작하며 page, include, taglib 등의 3가지 종류가 있음.

(1) page디렉티브: <%@page>

- JSP페이지에 대한 속성 설정
- 서버에 요청한 결과를 응답 받을 때 생성되는 페이지의 타입, 스크립트 언어, import할 클래스, 세션 및 버퍼의 사용여부 및 버퍼의 크기 등의 JSP페이지에서 필요한 설정 정보를 지정.

```
<%@ page 속성%>
```

(2) include디렉티브: <%@include>

- 조각코드를 넣을 때 사용
- 조각코드: 완전한 jsp페이지가 아니나, 웹페이지 반복적으로 사용되는 코드를 별도의 파일로 저장
- 공통적으로 포함될 내용을 가진 파일을 해당 JSP 페이지 내에 삽입하는 기능을 제공

```
<%@ include file="포함될 파일의 url"%>
```

- jsp페이지 템플릿은 <jsp:include>태그 사용

(3) taglib디렉티브: <%@taglib>

- 표현 언어, JSTL, 커스텀 태그를 JSP페이지 내에 사용할 때 씀.

```
<%@ taglib prefix="프리픽스" uri="url" %>
```

- prefix : uri대신 사용되는 네임스페이스, uri : 태그의 설정 정보

2) JSP 페이지의 스크립트요소

- JSP 페이지의 스크립트요소에는 선언문(Declaration), 스크립트릿(Scriptlet), 표

현식(Expression)이 있음

(1) 선언문 : <%! %>

- 멤버필드 선언과 메소드 선언

```
<%!
    String id = "Kingdora";

    public String getId() {
       return id;
    }
%>
```

(2) 스크립트릿 : <% %>

- 프로그래밍(JSP로직)코드 기술
- 스크립트릿에 쓸 수 없는 코드 : html태그, <%=%>

```
<%
  if(i>5){
       out.println("aa");
  }
%>
```

(3) 표현식: <%= %>

- 화면에 내용 출력. out.prinln(); 과 같음

```
id변수   : <%=id %><br>
```

(4) 주석(Comment) :

- JSP페이지에서 사용할 수 있는 주석: <!-- -->, <%-- --%>, //, /* */, html 주석, jsp주석, 자바주석을 사용할 수 있음
- html주석: <!-- --> , 화면에 표시되지 않으나 실행됨
- jsp주석: <%-- --%>
- 자바주석: //, /* */, jsp주석과 자바주석은 화면에 표시되지 않고 실행되지 않음.

나. 한글처리

- 웹서버로 전송하는 파라미터의 값과 응답되는 결과에 대한 한글처리 필요. 요청과 응답에서 한글처리

1) 웹서버로 전송하는 파라미터의 값에 대한 한글처리

```
<%request.setCharacterEncoding("utf-8");%>
```

2) 응답되는 결과에 대한 한글처리 - contentType="text/html; charset=UTF-8"

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
```

2. JSP페이지의 내장객체와 영역

- JSP페이지의 내장객체에는 request, response, out, pageContext, session, application, config, page, exception 가 있음
- request, session, application, pageContext 내장 객체는 영역(공유되는 범위)을 가지며, 속성 값을 저장하고 읽을 수 있는 메소드인 setAttribute()메소드와 getAttribute()메소드를 제공.

가. request : 요청정보 저장객체

- 웹 브라우저의 요청 정보를 저장하고 있는 객체
- 입력폼에 입력한 사용자의 요구사항을 얻어낼 수 있도록 요청 메소드를 제공
- 요청메소드:

▪ 1파라미터의 변수가 1개의 값을 가진 경우

```
String id = request.getParameter("id");
```

▪ 1파라미터의 변수가 여러 개의 값을 가진 경우

```
String[] h = request.getParameterValues("h");
```

- 웹 브라우저와 웹 서버의 정보도 가져올 수 있음

1) 요청객체 정보 설정

- request.setAttribute("s1","aaa");

2) 요청객체 정보 얻어내기

- String id = request.getAttribute("s1");

나. response: 응답정보 저장객체

- 웹 브라우저의 요청에 대한 응답 정보를 저장하고 있는 객체
- 응답 정보와 관련하여 주로 헤더 정보 입력, 리다이렉트 등의 기능을 제공

```
response.sendRedirect("이동할페이지");
```

다. out: 출력정보 저장객체

- JSP페이지의 출력할 내용을 가지도 있는 출력 스트림 객체.
- 표현식(<%=문장%>) 과 같음. out.println();

라. pageContext : JSP페이지 대한 정보 저장 객체

- JSP페이지 대한 정보를 저장하고 있는 객체.
- 다른 내장객체를 구하거나, 페이지의 흐름제어, 에러데이터를 얻어낼 때 사용.

마. session :웹브라우저의 정보 저장 객체

- 하나의 웹 브라우저 내에서 정보를 유지하기 위한 세션 정보를 저장하고 있는 객체
- 웹 브라우저(클라이언트)당 1개가 할당. 주로 회원관리 시스템에서 사용자 인증에 관련된 작업을 수행할 때 사용.

- 세션정보 설정 : session.setAttribute("id","aaa");
- 세션정보 얻어내기: String id = session.getAttribute("id");
- 세션 무효화: session.invalidate();

바. application :웹 어플리케이션 Context 정보 저장 객체

- 웹 어플리케이션 Context의 정보를 저장하고 있는 객체.
- 서버의 설정 정보, 자원에 대한 정보, 어플리케이션이 실행되는 동안에 발생할 수 있는 이벤트 로그 정보 등을 제공.
- 웹 어플리케이션 당 1개의 객체가 생성. 주로 방문자 카운트와 같은 하나의 웹 어플리케이션에서 공유하는 변수에 사용

```
application.log("로그 기록 : ");
String realPath = application.getRealPath(realFolder);
String realPath = context.getRealPath(realFolder);
```

사. config - JSP페이지 대한 설정정보를 저장 객체

- JSP페이지 대한 설정정보를 저장하고 있는 객체.
- 서블릿이 초기화되는 동안 참조해야 할 정보를 전달.
- 컨테이너당 1개의 객체가 생성.

```
String props = config.getInitParameter("propertyConfig");
ServletContext context = config.getServletContext();
```

아. page

- JSP 페이지를 구현한 자바 클래스 객체. 거의 사용 안함

자. exception

- JSP 페이지에서 예외가 발생한 경우에 사용되는 객체

3. JSP페이지의 액션태그 - include/forward 액션태그

가. include 액션태그 : <jsp:include>

- 다른 페이지의 실행결과를 현재의 페이지에 포함시킬 때 사용
- 기본적인 사용법

<jsp:include page="포함될페이지" flush="false"/>

예) <jsp:include page="b.jsp" flush="false"/>

1) include 액션태그에서 포함되는 페이지에 값 전달.

- include액션 태그의 바디(body) 안에 param 액션 태그(<jsp:param>)를 사용

```
<jsp:include page="포함되는 페이지" flush="false">
    <jsp:param name="paramName1" value="var1"/>
    <jsp:param name="paramName2" value="var2"/>
</jsp:include>
```

나. forward 액션태그 : <jsp:forward>

- 웹 페이지 간의 제어를 이동시킬 때 사용
- forward 액션태그를 만나게되면, 그 전까지 출력버퍼에 저장되어 있던 내용을 제거한 후 forward 액션태그가 지정하는 페이지로 이동.
- forward 액션태그의 기본적인 사용법

<jsp:forward page="이동할 페이지명"/>

예)<jsp:forward page="b.jsp">

1) forward 액션태그로 이동되는 페이지에 값 전달.

```
<jsp:forward page="<%=selectedPage%>">
     <jsp:param name="selectedColor" value="<%=selectedColor%>"/>
     <jsp:param name="name" value="<%=name%>"/>
</jsp:forward>
```

4. 에러처리 - 에러코드별 처리

- JSP에서의 에러는 하나의 코드에서 에러가 발생하더라도 웹 브라우저의 전체 화

면에 에러 메시지가 표시됨
- 에러가 어떠한 경로로 발생하게 되었는지 스택을 뒤집어서 그 경로를 추적해서 표시.
사이트의 사용자들이기 보기에는 부적합해서 완곡한 에러메시지의 표시가 필요.

가. JSP에서 404에러와 500에러 처리방법

① web.xml에 <error-page>엘리먼트를 사용해서 에러 제어
[프로젝트]-[WebContent]-[WEB-INF]의 web.xml에 404와 500에러를 제어할
 <error-page>엘리먼트를 각각 작성
② 에러가 발생시 표시할 페이지를 작성
[프로젝트]-[WebContent]-[error] 폴더에 404code.jsp와 500code.jsp페이지 작성
③ 404code.jsp 와 500code.jsp의 소스코드에
현재 페이지가 정상적으로 응답되는 페이지임을 지정하는 코드를 기술
<%response.setStatus(HttpServletResponse.SC_OK);%> 추가. 이 코드를 생략시,
  웹 브라우저는 자체적으로 제공하는 에러페이지를 표시

5. Ajax

- jQuery에서는 Ajax기능을 구현한 메소드들을 제공
- 서버로부터 TEXT, HTML, XML 또는 JSON 형태의 파일을 요청하고 응답받을 수 있는 기능을 제공
- 간단한 코드만을 사용해서 Ajax 기능 구현

가. Ajax 관련 메소드 중 서버 요청과 관련된 메소드

1) .load() : 서버로 데이터를 요청하고 HTML 엘리먼트에 응답받은 결과를 로드함
(넣음).

```
$("#div1").load(function()(responseText, textStatus, XMLHttpRequest){
 if(textStatus="success")
    //성공시 처리할 작업
 if(textStatus="error")
    //실패시 처리할 작업
});
```

2) $.get() : 서버로 HTTP get 방식의 요청을 함.
3) $.post() : 서버로 HTTP post방식의 요청을 함.
4) $.getJSON() : HTTP get 방식을 사용해서 JSON데이터를 요청함.
5) $.ajax() : 비동기 Ajax요청을 수행함. get, post 방식을 지정해서 사용

```
var query = {//프로퍼티};

$.ajax({
    type: "POST", //전송방식
    url: "process.jsp", //요청페이지
    data: query, //전송데이터
    success: function(data){ //요청페이지를 실행한 결과
       $("#result").html(data);
    }
});
```

나. 주로 사용하는 Ajax서버 요청 메소드

1) .load() : 서버로 데이터를 요청하고 HTML엘리먼트에 응답받은 결과를 로드.

- 응답 받은 결과를 화면에 표시해야 하는 로그인폼, 회원가입 폼, 글 목록 등을 실행할 때 주로 사용

예) $("#result").load("ajax/test.html");

2) $.ajax() : 비동기 Ajax요청을 수행함. get, post방식을 지정해서 사용.

- 주로 로그인 처리, 회원가입처리, 글쓰기처리등과 같이 DB와 연동 후 처리 결과만을 반환하는 경우에 사용

예)
```
$.ajax({
        type: "POST",
        url: "loginPro.jsp",
        data: query,
        success: function(data){}
    });
```

6. 자바빈

- 자바빈 개요/작성, useBean/setProperty/getProperty 액션태그
- 자바빈은 DB와의 연동을 담당하는 로직

가. 자바빈 작성

1) 자바빈(JavaBean) 클래스 작성

자바빈 클래스선언과 메소드 선언 - 접근 제어자 public

멤버변수(프로퍼티) - 접근제어자 private

예)

```
package bean.logon;
public class DbDataLogin{
  private String id;
  public void setId(String id){
    this.id = id.trim();
  }
  public String getId(){
    return id;
  }
}
```

2) 자바빈 사용

(1) 자바빈 객체 생성

- 방법1 : useBean 액션 태그(<jsp:useBean>)

<jsp:useBean id = "빈 이름" class = "자바빈 클래스 이름" scope = "범위" />

예<jsp:useBean id= "testBean" class="ch08.bean.TestBean" scope="page" />

- 방법2 : new를 사용

자바빈클래스이름 빈이름 = new 자바빈클래스이름();

예)ch08.bean.TestBean testBean = new ch08.bean.TestBean();

(2) 자바빈의 프로퍼티값 저장

-방법1: setProperty 액션 태그(<jsp:setProperty>)

<jsp:setProperty name= "빈 이름" property="프로퍼티 이름" value="프로퍼티에 저장할 값" />

```
예)<jsp:useBean id= "testBean" class="ch08.bean.TestBean" scope="page" >
     <jsp:setProperty name="testBean" property="id"/>
   </jsp:useBean>
```

-방법2: 레퍼런스.setXxx()

```
예)ch08.bean.TestBean testBean = new ch08.bean.TestBean();
   testBean.setId(id);
```

(3)자바빈의 프로퍼티값 얻기

- 방법1: getProperty 액션 태그(<jsp:getProperty>)
  <jsp:getProperty name= "빈 이름" property="프로퍼티 이름"/>
  예)<jsp:useBean id= "testBean" class="ch08.bean.TestBean" scope="page" />
    <jsp:getProperty name="testBean" property="id"/>

- 방법2: 레퍼런스.getXxx()
  예)ch08.bean.TestBean testBean = new ch08.bean.TestBean();
   testBean.getId();

7. 데이터베이스와 JSP연동

- 이클립스에서 DBMS제어, SQL쿼리문, JDBC를 사용한 DB연동, 커넥션 풀

가. DBMS다운로드 및 설치

- Oracle 다운 및 설치 - www.oracle.com
- Mysql 다운 및 설치 - dev.mysql.com
 예)다운받은 mysql-5.5.45-winx64.msi 파일을 더블클릭해서 설치
- JDBC커넥터 복사

1) JDBC 커넥터 복사 : 프로그램에서 오라클DB를 연동해서 사용
 -오라클10g : Windows7의 Oracle10g에서 C:\oracle\product\10.2.0\db_1\jdbc\lib안에 있는 ojdbc14.jar 파일을 복사해서 C:\Program Files\Java\jdk1.8.0_181\lib에 붙여넣기
.
 -오라클11g : Windows7/Windows10에서 Oracle11g를 기준으로 C:\app\계정명\product\11.2.0\dbhome_1\jdbc\lib안에 있는 ojdbc6.jar(12c는 ojdbc7.jar파일) 파일을 복사해 C:\Program Files\Java\jdk1.8.0_181\lib에 붙여넣기
 -mysql : mysql-connector-java-5.1.36-bin.jar 파일을 복사해 C:\Program Files\Java\jdk1.8.0_181\lib에 붙여넣기

나. 이클립스에서 드라이버 연결

1)이클립스 프로젝트에서 DBMS를 사용하기 위해 드라이버연결.
(1)웹 프로젝트에 JDBC드라이버 위치:
- 오라클: [프로젝트]-[WebContent]-[WEB-INF]-[lib]폴더에 C:\Program Files\Java\1.8.0_181\lib폴더 내에 있는 ojdbc14.jar 또는 ojdbc6.jar 또는 ojdbc7.jar파일을 복사.
- MySQL: [프로젝트]-[WebContent]-[WEB-INF]-[lib]폴더에 C:\Program

Files\Java\1.8.0_181\lib폴더 내에 있는 mysql-connector-java-5.1.36-bin.jar파일을 복사.

(2)톰캣컨테이너의 공용라이브러리에도 JDBC드라이버 위치:
- C:\Program Files\Java\1.8.0_181\lib폴더 내에 있는 ojdbc14.jar 또는 ojdbc6.jar 또는 ojdbc7.jar파일을 복사.또는 mysql-connector-java-5.1.36-bin.jar 파일을 복사해서 [톰캣홈]-[lib]폴더에 넣음.
- [톰캣홈]-[lib]폴더 예) C:\apache-tomcat-10.0.27\lib

2) 데이터베이스 생성 및 계정추가 - mysql의 경우
(1) MySQL에 데이터베이스 추가 - mysqladmin을 사용
 방법:
- C:\Program Files\MySQL\MySQL Server 5.5\bin>mysqladmin -u root -p create jsptest 입력 후 [enter]
- root계정 비밀번호(1234) 입력 후 [enter]

(2)생성된 데이터베이스에 사용자계정 추가 및 권한설정
- 방법:
▪ MySQL에 루트계정으로 데이터베이스에 접속
```
mysql -u root -p jsptest
```

▪ 사용자계정 추가 및 권한설정: root 권한으로 작성
- jsptest 데이터베이스에 로컬호스트(localhost)에 접근할 수 있는 사용자 계정 추가 및 권한을 설정
```
grant select, insert, update, delete, create, drop, alter
on jsptest.* to 'jspid'@'localhost'
identified by 'jsppass';
```

- jsptest 데이터베이스에 모든 서버(%)에 접근할 수 있는 사용자 계정 추가 및 권한을 설정
```
grant select, insert, update, delete, create, drop, alter
on jsptest.* to 'jspid'@'localhost'
identified by 'jsppass';
```

3) 이클립스에서 DBMS제어
- 이클립스에서 [Data Source Explorer]뷰를 사용한 데이터베이스 직접 제어

(1) Oracle(mysql)을 예시로 한 DB 커넥션 설정 : 데이터베이스 커넥션을 설정하기 위해 이클립스창의 아래에 위치한 [Data Source Explorer]뷰를 선택해서, [Database Connections]항목을 선택 후 마우스 오른쪽 버튼을 눌러 [New...]메뉴를 선택해서 한다.

① 데이터베이스 커넥션을 설정하기 위해 이클립스창의 아래에 위치한 [Data Source Explorer]뷰를 선택.
② [Data Source Explorer]뷰의 내용이 표시되면 [Database Connections]항목을 선택 후 마우스 오른쪽 버튼을 눌러 [New...]메뉴를 선택.
③ [New Connection Profile]창이 표시되면, [Connection Profile Type]항목에서 [Oracle] 또는 [Mysql]을 선택하고 [Name]항목에 "myOracle"을 입력 후 [Next]버튼을 클릭.
④ [Specify a Driver and Connection Details]화면이 표시되면, [Drivers]항목의 [New Driver Definition]버튼을 클릭한다.
⑤ [Name/Type]탭: Oracle의 JDBC드라이버가 표시. oracle 11g Thin driver선택, mysql의 JDBC드라이버가 표시. mysql 5.1선택
-[JAR List]탭: [Driver files]항목에서 기존의 예시로 표시된 드라이버를 선택하고 [Remove JAR/Zip]버튼을 클릭해서 제거한 후, 실제로 사용할 JDBC드라이버를 추가하기 위해 [Add JAR/Zip...]버튼을 클릭한 후 JDBC드라이브를 선택해 추가.
- 오라클의 경우는 C:\Program Files\Java\jdk1.8.0_172\lib 위치에 복사해 놓은 ojdbc14.jar 또는 ojdbc6.jar을 사용.
- mysql의 경우는 C:\Program Files\Java\jdk1.8.0_172\lib 위치에 복사해 놓은 mysql-connector-java-5.1.36-bin.jar 을 사용.

-[Properties]탭: 아래와 같이 입력 후 [OK]버튼을 클릭한다.
[Connection URL]항목- jdbc:oracle:thin:@localhost:1521:orcl
[Database Name]항목- orcl
[Password]항목- ict3488
[User ID]항목- system
- mysql
[Connection URL]항목- jdbc:mysql://localhost:3306/jsptest
[Database Name]항목- jsptest
[Password]항목- jsppass
[User ID]항목- jspid

⑥[Specify a Driver and Connection Details]화면의 [Properties]항목이 내용을 가진 화면이 표시된다. 이때 [Test Connection]버튼을 클릭. [Success]대화상자가 표시되면 [OK]버튼을 클릭.
⑦ [Specify a Driver and Connection Details]화면의 [Finish]버튼을 클릭하면, 모든 설정이 끝난다.
⑧ MySQL또는 오라클을 직접 제어하는 커넥션이 연결된 것을 확인할 수 있다.
커넥션을 연결할 때: [Connection]메뉴를 사용
커넥션을 해제할 때: [Disconnection]메뉴를 사용.

(2) [Data Source Explorer]뷰에서 설정된 데이터베이스 커넥션을 사용한 데이터베이스 제어
-쿼리문을 작성할 때는 스크랩북(scrapbook)을 생성해서 한다.
-스크랩북에서 쿼리 실행시 : 쿼리문을 입력하고 드래그해 블록을 지정한 후 Alt+X키
① 스크랩북을 작성하기 위해서는 [Data Source Explorer]뷰에서 설정된 데이터베이스 커넥션선택 후 마우스 오른쪽 버튼을 클릭해 [Open SQL Scrapbook]메뉴를 선택해서 한다.
② 새 스크랩 북이 편집기 뷰에 표시되면 [Connection profile]항목에 설정을 한다.
[Type]항목의 콤보상자를 사용해 [Oracle xxg]을 선택한 후, [Name]항목에서 [myOracle]을 선택하고 [Database]항목에서 [orcl]를 선택한다.

③ 스크랩북이 에디터뷰에 표시되면 쿼리문을 입력하고 드래그해 블록을 지정한 후 Alt+X키를 쿼리를 실행한다.
(3) 테이블 생성 : [studyjsp]프로젝트의 mysqlconn.sql파일 참조

```
create table member(
   id varchar2(12) not null primary key,
   passwd varchar2(12) not null,
   name varchar2(10) not null,
   reg_date date not null
);
```

다. SQL쿼리문

- SQL(Structured Query Language): 데이터베이스 생성부터 레코드 검색 등의 작업을 수행할 때 사용

1) SQL문의 종류
 -데이터 정의문(DDL)- CREATE, ALTER, DROP
 -데이터 제어문(DCL) - GRANT, REVOKE
 -데이터 조작문(DML) - UPDATE, INSERT, DELETE,
 -쿼리(Query) - SELECT
 -트랜잭션(Transaction) 처리 - COMMIT, ROLLBACK
2) 데이터데이터 타입(Data Type)
 - MySQL
  숫자: int, 문자: varchar, 날짜: datetime
 - Oracle
  숫자: number, 문자: varchar2, 날짜: date

3) 테이블 작업 관련 쿼리문
(1) 테이블 생성: CREATE문
작성 예
```
create table member(
  id varchar(50) not null primary key,
  passwd varchar(16) not null,
  name varchar(10) not null,
  reg_date datetime not null
);
```

(2) 테이블 수정: ALTER문
- 테이블의 필드를 추가하는 ALTER문의 일반형
```
ALTER TABLE table_name
  ADD (add_col_name1 type [DEFAULT] [NOT NULL/NULL],
        ....
        add_col_name3 type );
```

- 작성 예
```
alter table member
  add (address varchar(100) not null,
   tel varchar(20) not null);
```

(3) 테이블 제거: DROP문

- 문법 : DROP TABLE 삭제할테이블명;
- 작성 예 : drop table test;

4) 레코드 작업 관련 쿼리문
(1) 레코드 추가 : INSERT문
- 레코드를 추가하는 쿼리의 일반형
INSERT INTO table_name (col_name1,col_name2...)
    VALUES (col_value1, col_value2...)
- 사용 예
insert into member(id, passwd, name, reg_date)
  values('kingdora@dragon.com','1234','김개동', now());

(2) 레코드 검색 : SELECT문
- 레코드를 검색하는 쿼리의 일반형
SELECT col_name1,col_name2 FROM table_name;
- 사용 예
select * from member;

(3) 레코드 수정 : UPDATE문
- 레코드를 수정하는 쿼리의 일반형
UPDATE table_name SET col_name = value,... WHERE condition;
- 사용 예
update member set passwd='3579' where id='abc';

(4) 레코드 삭제 : DELETE문
- 테이블에 저장되어 있는 레코드를 삭제하는 쿼리의 일반형
DELETE FROM table_name WHERE condition;
- 사용 예
delete from member where id='abc';

라. JDBC를 사용한 DB연동: JSP와 DB사이의 연동
- 자바 프로그램(JSP포함)과 관계형 테이터 원본(데이터베이스, 테이블...)을 연결하는 인터페이스.
- JDBC 라이브러리는 'java.sql'패키지에서 제공하며 SQL문을 실행시키기 위한 인터페이스로 설계.
- JDBC프로그램은 JDBC 드라이버 로드, Connection 객체 생성, 쿼리 실행 객체 생성, 쿼리 수행의 필수 4단계에 의해 프로그램 됨

1) JDBC 드라이버 로드
- Class 클래스의 forName() 메소드를 사용해서 드라이버를 로드
예)
Class.forName("com.mysql.jdbc.Driver");//mysql
Class.forName("oracle.jdbc.driver.OracleDriver");//oracle

2) Connection 객체 생성
- DriverManager에 등록된 각 드라이버들을 getConnection(String url)메소드를 사용해서 식별
예)
//mysql
Connection conn=
DriverManager.getConnection("jdbc:mysql://localhost:3306/jsptest","jspid","jsppass");
//oracle
Connection conn=
DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:orcl", "scott", "tiger");

- 오라클 클라이언트 사용시
Connection conn=
DriverManager.getConnection ("jdbc:oracle:thin:@오라클서버IP:1521:orcl", "scott", "tiger");

3) 쿼리 실행 객체 생성
- sql쿼리를 생성하며, 반환된 결과를 가져오게 할 작업영역을 제공
예)
Statement stmt = conn.createStatement();
PrepardStatement pstmt = conn.prepareStatement(sql);
CallableStatement cstmt = conn.prepareCall(); //스토어드 프로시저

4) 쿼리 수행
- Statement/PrepardStatement/CallableStatement객체의 executeQuery()메소드나 executeUpdate()메소드를 사용해서 쿼리를 실행.

- stmt.executeQuery() :recordSet 반환. Select 문에서 사용
ResultSet rs = stmt.executeQuery ("select * from 소속기관");
- stmt.executeUpdate(): 성공한row 수 반환. Insert 문, Update 문, Delete 문에서 사용.
String sql="update member set passwd='3579' where id='abc' ";
stmt.executeUpdate(sql);

5) 쿼리결과 처리 : select문을 사용한 쿼리의 경우
- ResultSet 객체로부터 원하는 데이터를 추출하는 과정
- ResultSet 객체에서 한 행씩 이동(rs.next())하면서 getXxx()를 이용해서 원하는 필드 값을 추출. Xxx는 필드의 데이터 타입

마. 자카르타 DBCP API를 이용한 커넥션 풀 사용 : 커넥션풀 기반의 DB연동
1) DBCP API관련 jar파일 설치
- 톰캣홈\lib폴더에 있는 tomcat-dbcp.jar를 복사해서 이클립스의 [프로젝트]-[WebContent]-[WEB-INF]-[lib]에 붙여넣기

2) DBCP에 관한 정보 설정 - context.xml
(1) 이클립스 가상환경세팅 : 이클립스의 [Servers]-[Tomcat9.0~]에 있는 context.xml열기
- context.xml파일의 </Context>위에 제공하는 JNDIserver세팅.txt파일에서 아래의 내용 복사 후 붙여넣기
```
 ---붙여 넣을 내용 : mysql용 ----
  <Resource name="jdbc/jsptest"
            auth="Container"
            type="javax.sql.DataSource"
            driverClassName="com.mysql.jdbc.Driver"

            loginTimeout="10"
            maxWait="5000"
            username="jspid"
            password="jsppass"
            testOnBorrow="true"
            url="jdbc:mysql://localhost:3306/jsptest"
/>
```

---붙여 넣을 내용 : 오라클용 ----

```
<Resource name="jdbc/jsptesto"
          auth="Container"
          type="javax.sql.DataSource"
          driverClassName="oracle.jdbc.driver.OracleDriver"

          loginTimeout="10"
          maxWait="5000"
          username="system"
          password="ict3488"
          testOnBorrow="true"
          url="jdbc:oracle:thin:@localhost:1521:orcl"
 />
```

(2) 실제 환경세팅: [톰캣홈]-[conf]에 있는 context.xml열기

- context.xml파일의 </Context>위에 제공하는 JNDIserver세팅.txt파일에서 아래의 내용 복사 후 붙여넣기

---붙여 넣을 내용 : mysql용 ----

```
<Resource name="jdbc/jsptest"
          auth="Container"
          type="javax.sql.DataSource"
          driverClassName="com.mysql.jdbc.Driver"

          loginTimeout="10"
          maxWait="5000"
          username="jspid"
          password="jsppass"
          testOnBorrow="true"
          url="jdbc:mysql://localhost:3306/jsptest"
/>
```

---붙여 넣을 내용 : 오라클용 ----

```
<Resource name="jdbc/jsptesto"
          auth="Container"
          type="javax.sql.DataSource"
          driverClassName="oracle.jdbc.driver.OracleDriver"
```

```
            loginTimeout="10"
            maxWait="5000"
            username="system"
            password="ict3488"
            testOnBorrow="true"
            url="jdbc:oracle:thin:@localhost:1521:orcl"
 />
```

3) JNDI 리소스 사용 설정 - web.xml
- 이클립스의 [프로젝트]-[WebContent]-[WEB-INF]에 있는 web.xml열기
- web.xml파일의 </web-app>위에 제공하는 JNDIserver세팅.txt파일에서 아래의 내용 복사 후 붙여넣기

---붙여 넣을 내용 : mysql용 ----

```
<resource-ref>
    <description>jsptest db</description>
    <res-ref-name>jdbc/jsptest</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>
```

---붙여 넣을 내용 : 오라클용 ----

```
  <resource-ref>
      <description>jsptest oracledb</description>
      <res-ref-name>jdbc/jsptesto</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
  </resource-ref>
```

4) JSP페이지나 자바빈에서 커넥션 풀 사용 : 자바빈에 작성

```
 Context initCtx = new InitialContext();
 Context envCtx = (Context) initCtx.lookup("java:comp/env");
 DataSource ds = (DataSource)envCtx.lookup("jdbc/jsptesto");
 Connection conn = ds.getConnection();//conn객체를 사용해서 DB연동
```

바. 트랜잭션 처리

- 트랜잭션은 여러 단계의 작업을 하나로 처리하는 것

- 하나로 인식된 작업이 모두 성공적으로 끝나면 commit이 되고, 하나라도 문제가 발생하면 rollback되어서 작업을 수행하기 전 단계로 모든 과정이 회수
- JSP에서 제공하는 트랜잭션 처리에 메소드
- commit(): 트랜잭션의 commit을 수행
- rollback() : 트랜잭션의 rollback을 수행
- JSP는 오토커밋(Autocommit): 트랜잭션을 처리할 때는 오토커밋을 해제
- setAutoCommit(false);
- 트랜잭션 1단위 작성순서
- 오토커밋해제: setAutoCommit(false); // 트랜잭션 시작
- 1개의 작업으로 인식할 쿼리들을 나열...
- conn.commit();
- 오토커밋설정: setAutoCommit(true); // 트랜잭션 끝

사. 데이터베이스 암호화
- 암호화된 정보를 다시 복호화 할 수 없는 정보 : 해시함수
- 해시함수를 사용하는 것 : 비밀번호
- 암호화된 정보를 다시 복호화 할 수 있는 정보 : 블록암호
- 블록암호를 사용하는 것 : 바이오정보, 주민등록번호, 신용카드번호, 계좌번호, 여권번호, 운전면허번호, 외국인등록번호, 웹에서의 정보전달(웹사이트주소)

8. 쿠키와 세션
- 웹 페이지 간에 정보 유지

가. 쿠키
- 정보를 웹 브라우저에 저장 <-정보유지
- 이름, 값, 유효기간, 도메인, 경로 등으로 이루어짐

1) 쿠키 생성
- Cookie cookie = new Cookie(String name, String value);

2) 쿠키의 사용
- 쿠키를 response객체에 추가 : response.addCookie(cookie);
- request객체에 실려 온 쿠키를 읽어 올 때: Cookie[] cookies = request.getCookies();

나. 세션
- 웹 서버 쪽의 웹 컨테이너에 상태를 유지하기 위한 정보를 저장. 웹 브라우저 당 1개씩 생성되어 웹 컨테이너에 저장
- 웹 브라우저와 웹 서버의 상태유지가 훨씬 안정적이고 보안상의 문제도 해결

1) 세션 속성의 설정

- session객체의 setAttribute()메소드를 사용
  session.setAttribute("id","aaaa@king.com");
2) 설정된 세션의 속성을 사용해 정보를 유지
- session객체의 getAttribute()메소드를 사용
  String id = (String)session.getAttribute("id");
3) 세션의 속성 삭제
- session객체의 removeAttribute()메소드를 사용. 예) session.removeAttribute("id");
4)세션 무효화 - 모든 세션 속성 제거
- session객체의 invalidate()메소드. session.invalidate();

9. 회원관리/게시판 시스템 유기적 구조

가. 회원관리 시스템

1)회원관리 시스템의 개요

2)회원관리 필요 테이블 작성 : [member]테이블 작성

```
--오라클
create table member3(
  id varchar2(50) not null primary key,
  passwd varchar2(16) not null,
  name varchar2(10) not null,
  reg_date date not null
);
alter table member3
       add(address varchar2(100) not null,
            tel varchar2(20) not null);
alter table member3
       modify(passwd varchar2(60));

--mysql
create table member(
  id varchar(50) not null primary key,
  passwd varchar(16) not null,
  name varchar(10) not null,
  reg_date datetime not null
);
```

```
alter table member
      add (address varchar(100) not null,
           tel varchar(20) not null);
alter table member modify passwd varchar(60) not null;
```

3) 회원관리 자바빈 작성

- 데이터 저장빈: LogonDataBean.java
- DB 처리빈 : LogonDBBean.java

4) 회원관리 JSP페이지 작성

- jsp에서 메인페이지로 복귀 원할하지 않은 경우: window.location.href="main.jsp";
- 메인 페이지 작성
- 회원 가입 페이지 작성
- 회원 인증 페이지 작성
- 회원 정보 수정 및 탈퇴 페이지 작성

나. 게시판 시스템

1) 게시판 시스템의 개요

2) 게시판 필요 테이블 작성:[board]테이블

-mysql

```
create table board(
  num int not null primary key auto_increment,
  writer varchar(50) not null,
  subject varchar(50) not null,
  content text not null,
  passwd varchar(60) not null,
  reg_date datetime not null,
  ip varchar(30) not null,
  readcount int default 0,
  ref int not null,
  re_step smallint not null,
  re_level smallint not null
);
```

-oracle

```
create table board(
  num number not null primary key,
  writer varchar2(50) not null,
```

```
  subject varchar2(50) not null,
  passwd varchar2(60) not null,
  reg_date date not null,
  readcount number default 0,
  ref number not null,
  re_step number not null,
  re_level number not null,
  content clob not null,
  ip varchar2(30) not null
);
```

3) 게시판 자바빈 작성
- 데이터 저장빈 - BoardDataBean.java
- DB처리 빈 - BoardDBBean.java
4) 게시판 JSP페이지 작성
- 메인 페이지 작성
- 글쓰기 페이지 작성
- 글목록 보기 페이지 작성 - 글내용 보기 포함
- 글 수정 및 삭제 페이지 작성

10. 모델2기반의 MVC패턴

- 모델1과 모델2비교, MVC패턴, 컨트롤러 서블릿에 사용자 요청을 명령어로 전달

가. 모델2와 MVC패턴의 개요

1) 모델1(Model 1) VS 모델2(Model 2)
(1)모델1(Model 1)
- 웹 브라우저의 요청(request)을 받아들이고, 웹 브라우저에 응답(response)하는 것을 JSP페이지가 단독으로 처리하는 구조
(2)모델2(Model 2)
- 요청(request)처리, 데이터접근(data access), 비즈니스 로직(business logic)을 포함하고 있는 컨트롤러와 뷰는 엄격히 구분

나. MVC패턴(Model-View-Controller pattern)

- 전통적인 GUI(Graphic User interface) 기반의 어플리케이션을 구현하기 위한 디자인 패턴.
- MVC 구조는 사용자의 입력을 받아서 입력 에 대한 처리를 하고, 그 결과를 다시 사용자에게 표시하기 위한 최적화된 설계를 제시.

1) MVC패턴의 요소

-모델(Model) : 로직을 가지는 부분

- DB와의 연동을 통해서 데이터를 가져와 어떤 작업을 처리
- 처리한 작업의 결과를 데이터로서 DB에 저장하는 일을 처리
- 자바빈(JavaBean), 처리로직인 자바클래스(Java class)가 이에 해당

-뷰(View) : 화면에 내용을 표시

- 정보를 보여주는 역할만을 담당
- JSP페이지가 이에 해당

-컨트롤러(Controller) : 어플리케이션의 흐름을 제어

- 사용자의 요청을 받아서 모델(Model)에 넘겨줌
- 모델(Model)이 처리한 작업의 결과를 뷰(View)로 보냄
- 서블릿(Servlet)이 이에 해당

2) 컨트롤러(Controller)의 작업 처리과정

① 웹 브라우저의 요청을 받음: 웹브라우저의 요청은 서블릿의 서비스 메소드인 doGet() 또는 doPost()메소드가 받음

② 웹 브라우저가 요구하는 작업을 분석: 사용자가 요구한 작업에 맡는 로직이 실행되도록, 웹브라우저의 요구 작업을 분석

③ 모델을 사용해서 요청한 작업을 처리: 요청한 작업에 해당하는 로직을 처리

④ 로직 처리결과를 request객체의 속성에 저장: request객체의 속성에 처리결과를 저장. 처리결과는 같은 request객체 영역에서 공유.

⑤ 적당한 뷰(JSP페이지)를 선택 후 해당 뷰로 포워딩(forwarding): 처리결과를 저장한 request객체를 뷰로 전달.

나. 컨트롤러(Controller)인 서블릿에 사용자의 요청을 명령어로 전달 - 커맨드 패턴

- 사용자가 어떤 요청을 했는지 판단하기 위한 가장 일반적인 방법이 명령어로서 사용자의 요청을 전달
- 명령어와 로직을 연결하는 properties매핑파일이 필요.

1) 요청 파라미터로 명령어를 전달하는 방법

- 컨트롤러인 서블릿에 요청 파라미터를 정보를 덧붙여서 사용

- http://localhost:8080/studyjsp/MessageContoller?message=aaa

- 간편하긴 하나 명령어가 파라미터로 전달되게 되면 정보가 웹 브라우저를 통해 노출

2) 요청 URI자체를 명령어로 사용하는 방법

- 사용자가 요청한 URI자체를 명령어로 사용하는 방법

- http://127.0.0.1:8080/studyjsp/ch17/test.do

- 요청되는 URI가 실제 페이지가 아니고 명령어이므로 악의적인 명령어로부터 사이트가 보호되며, 요청되는 URL이 좀 더 자연스러워 진다는 장점

11. 시큐어코딩

가. 예외처리 메시지

- e.printStrackTrace();를 사용하지 않음 : 서비스할 때
- 디버깅후에는 꼭 제거
- 예외처리를 한 번에 Exception으로 하지 않고 각각의 코드에 대응해서 작성
- 각각의 예외에 대해서 메시지 출력.

```
System.out.println("메시지"); //서비스할 때
```

나. 캡슐화된 배열

- private으로 선언된 배열이나 컬렉션은 get/ set요청시(setter/getter사용시) 복사된 데이터를 반환하도록 코딩 : 책189p문제 해결

예)

```
public class Role{
  private String[] adminRoles={"admin","root","manager"};
  public String[] getAdminRole(){//배열을 얻어냄
    String[] newRoles = new String[adminRoles.length];
    System.arraycopy(adminRoles,0,newRoles,0,adminRoles.length)
    return newRoles;//복제된 배열반환
  }
  //adminRoles: 웹에서 받아온 배열
  public void setAdminRoles(String[] adminRoles){
    String[] newRoles = new String[adminRoles.length];
    System.arraycopy(adminRoles,0,newRoles,0,adminRoles.length)
    this.adminRoles = newRoles;
  }
}
```

참고문헌

- https://www.w3schools.com/
- http://jsbin.com/
- HTML&CSS그리고JavaScript, 김은옥, 삼양미디어
- Ajax대응 JavaScript 실무 프로그래밍, 김은옥, 삼양미디어
- JSP2.3 웹프로그래밍, 김은옥, 삼양미디어