

# 6주차 강의

## 자바웹프로그래밍(1)

강사 : 최도현

# 트렌드 분석

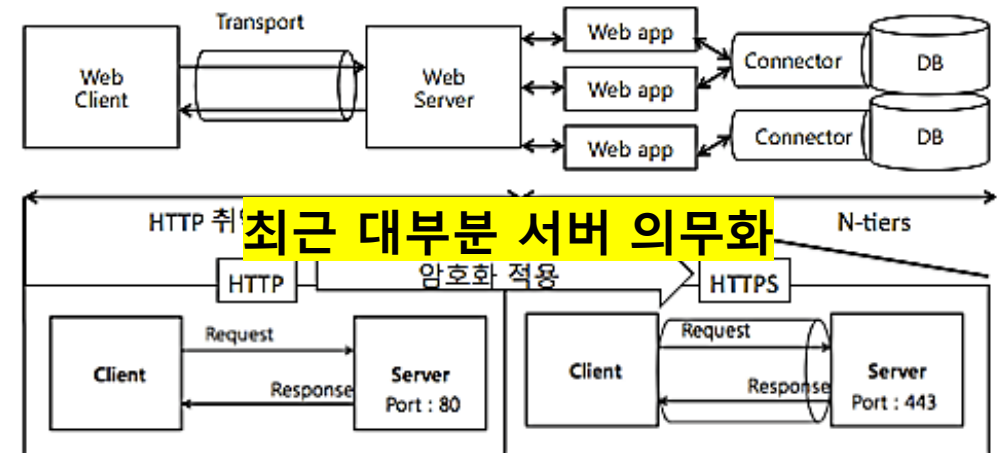
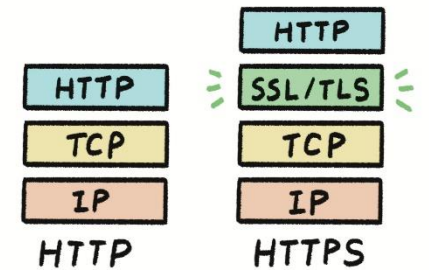
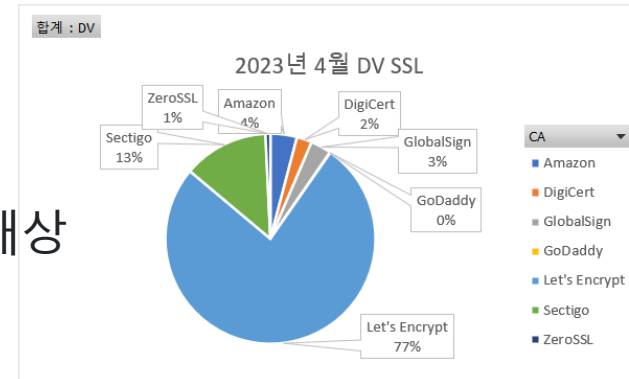
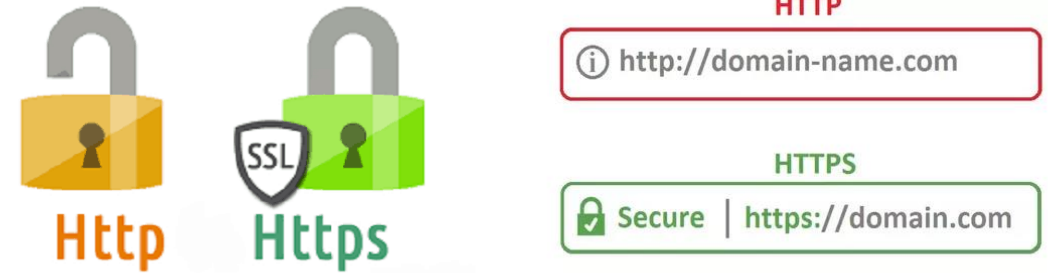
## •웹 서비스 및 개발 트렌드

기술 트렌드 - 통신 프로토콜과 보안



# HTTP, HTTPS

- HTTP : 웹 표준 통신 프로토콜
  - 컨텐츠(텍스트, 이미지 등) 전송
    - 문제점 : 설계 단계에서 이미 보안 X
      - 공개된 80포트 웹 서버는 항상 해커 공격 대상
- HTTPS : TLS/SSL 보안 기술 적용
  - 암호화 및 안전한 통신 과정 제공
    - 신뢰된 웹사이트(서버 SSL 인증서)
    - 고정 443포트에서 동작
- 서버에서 직접 보안설정
  - 보안 서버 구축
    - 국내 : 한국전자인증 SSL인증서
      - 대부분 LET 점유(무료)

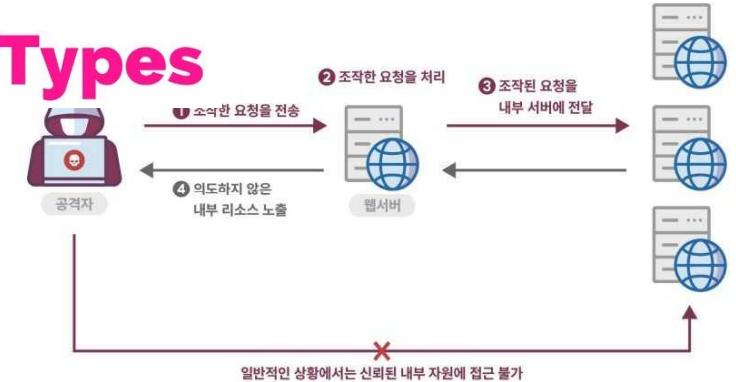


# 중요 데이터 - 보안

- HTML 폼 전송과 보안
  - 중요한 데이터를 전송하는 FORM?
    - 모든 정보는 조작 가능하다.
  - 보안을 위한 해결책 : 대부분 백엔드
    - HTTPS 프로토콜 적용 : 서버
    - 보안 서버 XSS 설정 : 서버
    - CSRF 토큰 구현 : 백엔드
    - 입력 데이터 검증 : 기본 JS(프론트 일부)
- 자바스크립트 필터링?
  - OWASP : 대표 웹 보안 분석 기관
    - 전용 필터링 라이브러리 제공
      - <https://cheatsheetseries.owasp.org/>

☐ Radio Button
 
☐ Checkbox

## HTML Input Types



OWASP TOP 10 (2017)	OWASP TOP 10 (2021)
A01:2017-Injection (인젝션)	A01:2021-Broken Access Control (취약한 접근통제)
A02:2017-Broken Authentication (취약한 인증)	A02:2021-Cryptographic Failures (암호화 오류)
A03:2017-Sensitive Data Exposure (민감한 데이터 노출)	A03:2021-Injection (인젝션)
A04:2017-XML External Entities(XXE) (XML 외부 객체(XXE))	A04:2021-Insecure Design (안전하지 않은 설계) (신규)
A05:2017-Security Misconfiguration (잘못된 보안 구성)	A05:2021-Identification and Authentication Failures (사용자 식별 및 인증 오류)
A06:2017-Session Management (세션 관리)	A06:2021-Software and Data Integrity Failures (소프트웨어와 데이터 무결성 오류) (신규)
A07:2017-Cross-Site Scripting(XSS) (크로스 사이트 스크립팅)	A07:2021-Server-Side Request Forgery(SSRF) (서버측 요청 위조) (신규)
A08:2017-Insecure Deserialization (안전하지 않은 역직렬화)	
A09:2017-Using Components With Known Vulnerabilities (알려진 취약점이 있는 구성요소 사용)	
A10:2017-Insufficient Logging & Monitoring (불충분한 로깅 및 모니터링)	

## OWASP TOP10 취약점 준수

# PART1

## • FORM을 통한 데이터 전송

자바스크립트 – 함수, 화살표 함수

팝업창(확장)

로그인 폼

로그인 폼(필터링)

### HTML Form

First Name :

Last Name :

Date of Birth :

Email id :

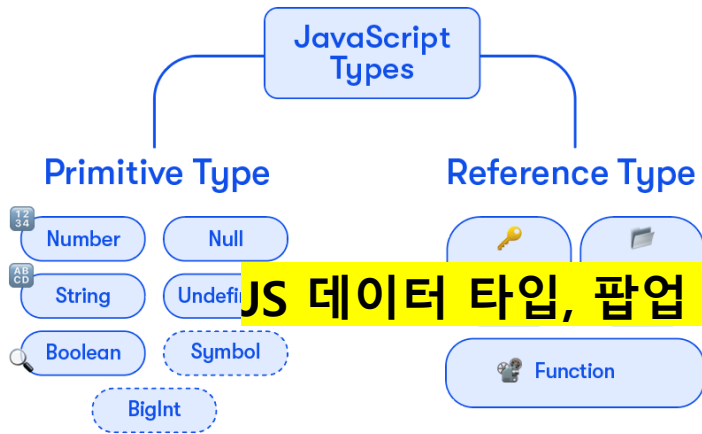
Mobile Number :

SUBMIT

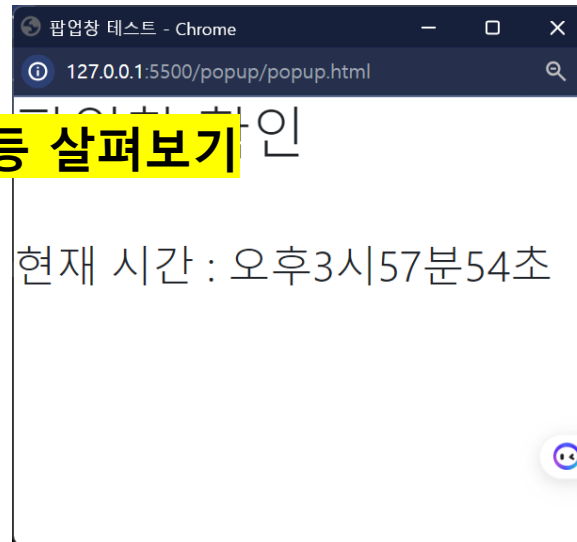
RESET

# 지난주 내용 살펴보기

- 자바스크립트 데이터 타입
  - 검색창(확장) 및 팝업 창 기능 확인



JS 데이터 타입, 팝업 창 등 살펴보기



항목	태그 이름/설명
JS의 참조 데이터 타입은? 2개 이상	
Undefined 데이터 타입의 특징은?	
문자열과 변수를 조합하여 출력하는 방법을 무엇이라 하는가?	
Get / post 2가지 전송 방식의 핵심 차이점은?	
페이지 요청에 Utf-8 인코딩 처리 전용 함수는?	
페이지 로딩과 동시에 js를 호출하는 속성은?	
상대경로를 지정하는 경우 상위 폴더를 지정하는 방법은?	
자바스크립트의 핵심 타이머 함수는?	

# 자바스크립트 - 함수, 화살표 함수

- 자바스크립트 함수
  - 기존 함수는 사용법 동일
    - 키워드 function, 상대적 복잡?
  - ES 5 → ES 6버전 이후
    - 화살표 함수(arrow function) 적용
      - 간결하고 명확한 표현, 괄호 생략 O
      - 단점 : NEW 연산자 불가능(객체 생성 X)
- 다양한 소스코드에서
  - 기존 VS 화살표
    - 일단 둘 다 사용하니 READING은 해두자!



구분	ES5 함수	ES6 화살표 함수
기능	함수 정의	함수 정의
문법	function 키워드 사용	화살표 (=>) 사용
간결성	상대적으로 복잡	훨씬 간결
this 바인딩	함수 내부에서 this는 자신을 가리킴	상위 스코프의 this를 유지
lexical scoping	블록 스코프 내 변수에 접근 불가능	블록 스코프 내 변수에 접근 가능
생성자 함수	사용 가능	사용 불가능
prototype 프로퍼티	생성된 함수 객체에 존재	생성된 함수 객체에 없음
yield 키워드	사용 가능	사용 불가능

# 자바스크립트 - 함수, 화살표 함수

- 화살표 함수를 사용해보자.
  - 기존 js 폴더 안에 popup.js를 수정한다.
    - 기존 소스코드 주석처리
    - over 함수를 화살표 함수로 변경한다.
  - search\_message도 화살표 함수로 변경한다.
- 정상 동작 확인
  - 마우스 호버, 검색 메시지 등
    - F12 개발자 모드에서 확인, 에러가 있다?
- 호이스팅(Hoisting)
  - 변수, 함수, 클래스 등 선언의 이동 처리
    - 기존 es5 표준 함수와 var 변수 지원
    - es6 이후 TDZ로 처리됨, let, const 추천!

```
const over = (obj) => {  
  obj.src = "image/LOGO.png";  
};
```

```
const search_message = () => {  
  const c = '검색을 수행합니다';  
  alert(c);  
};
```

호이스팅(이자리로 변수를 끌어올립니다)

```
console.log(a);  
var a = 100; // undefined
```

**선언은 상단을 권장!**

	선언단계	ReferenceError
	일시적 사각지대(TDZ)	
let foo;	초기화 단계	foo === undefined
foo = 1;	할당 단계	foo === 1



# LOL 웹 사이트 - 팝업창(확장)

- 팝업창 자동 닫기 기능을 구현한다.
  - 10초 정도 유지되고 자동으로 닫힌다.
- Popup 폴더의 popup.html을 수정한다.
  - 카운트 다운 출력을 최하단 추가한다.

```
<h1 class="display-4"><div id="Time" class="clock"></div></h1>
```

- Js 폴더에 popup\_close.js를 작성한다.
  - pop\_up.html의 head에 파일을 <script> 태그로 연동
- 팝업창의 카운트 다운 확인
  - 화면에 값을 삽입(innerText)하는 방식
  - F12 개발자 모드에서 에러가 없는지 확인

```
var close_time; // 시간 정보  
var close_time2 = 10; // 10초 설정
```

```
clearTimeout(close_time); // 재호출 정지  
close_time= setTimeout("close_window()", 10000);  
// 1/1000 초 지정, 바로 시작  
show_time(); // 실시간 시간 보여주기
```

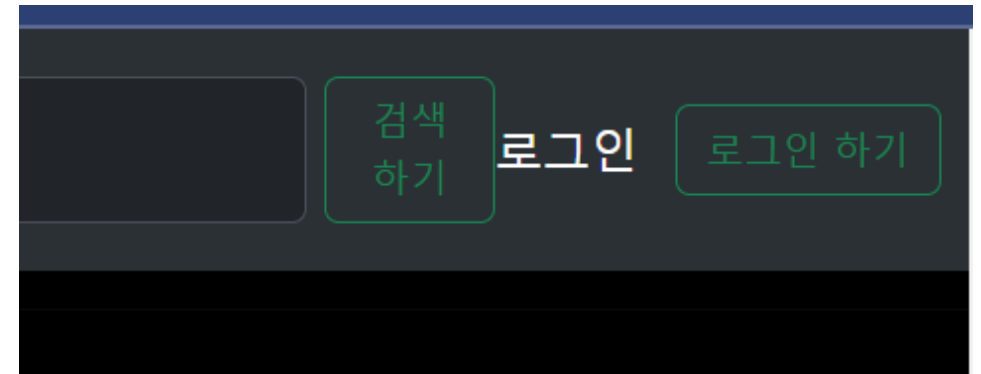
```
function show_time(){  
    let divClock = document.getElementById('Time');  
    divClock.innerText = close_time2; // 10초 삽입 시작  
    close_time2--; // 1초씩 감소  
    setTimeout(show_time, 1000); //1초마다 갱신  
}
```

```
function close_window() { // 함수 정의  
    window.close(); // 윈도우 닫기  
}
```



# LOL 웹 사이트 - 로그인 폼

- 메인화면에 표시될 로그인 버튼을 구현한다.
  - 부트스트랩 디자인 버튼을 사용
  - 위치는 검색 버튼 우측에 놓자.
- Index.html을 수정한다.
  - 검색 버튼의 위치를 파악한다. 아래 링크 추가



```
<a href="/login/login.html" class="btn btn-outline-success" id="login_btn">로그인 하기</a>
```

- 참고 : 검색 버튼과 다르다?
  - 부트스트랩 적용으로 같은 class 디자인 적용
  - 실제로는 버튼이 아니고 하이퍼 링크다.

Cannot GET /login/login.html

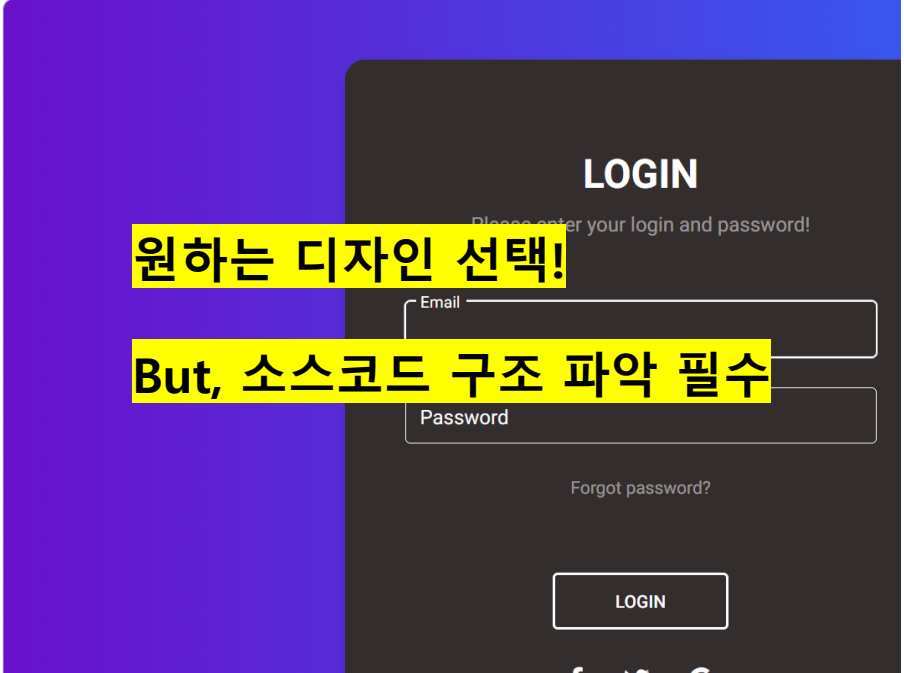
**로그인 페이지 필요**

# LOL 웹 사이트 - 로그인 폼

- 로그인 페이지 화면을 구현한다.
  - 아이디, 비밀번호 입력 등 로그인 기능 구현
- login 폴더를 생성하고 login.html을 작성한다.
  - 기존 index.html 에서 head 부분을 가져온다.
    - 부트스트랩, body와 footer는 남겨둔다.
  - 디자인 템플릿 선택(외부 무료 웹 사이트)
    - <https://mdbootstrap.com/docs/standard/extended/login/>
    - 부트스트랩 5 호환
  - 화면 하단의 SHOW CODE를 클릭
    - 웹 에서 HTML 소스를 login.html에 붙여넣기

## Login Modal

This example of a login card would work great as a [popup](#) on lighter backgrounds. Check out our [moc](#) styles and behavior.



**원하는 디자인 선택!**

**But, 소스코드 구조 파악 필수**

HTML CSS

# LOL 웹 사이트 - 로그인 폼

- 디자인 관련 css 소스 코드
  - 샘플 로그인 전용 디자인 css 파일 추가
- css 폴더를 생성하고 login.css 파일을 작성한다.
  - 웹에서 css 소스를 login.css에 붙여넣기

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

- 스타일 시트를 파일 형태로 저장한다.
- 로그인 화면 및 css가 잘 적용되었는지 확인
  - 현재 로그인 화면 내용 수정
  - 가독성을 위해 영어 → 한글로 직접 수정

검정 화면에 배경 파란색

로그인 메인화면

이메일 형식의 아이디와 패스워드를 입력해주세요!

이메일

패스워드

[패스워드 찾기](#)

로그인

# LOL 웹 사이트 - 로그인 폼

- 아이디, 비밀번호 입력 후 동작을 추가 구현
  - 로그인 버튼을 클릭 → 로그인 된 페이지로 전환된다.
- login.html에 form 태그를 추가한다.
  - 화면 영역을 기준으로 submit 된다.
  - Form 태그를 적절하게 위아래 추가

```
<form method="get" action="/login/index_login.html">
```

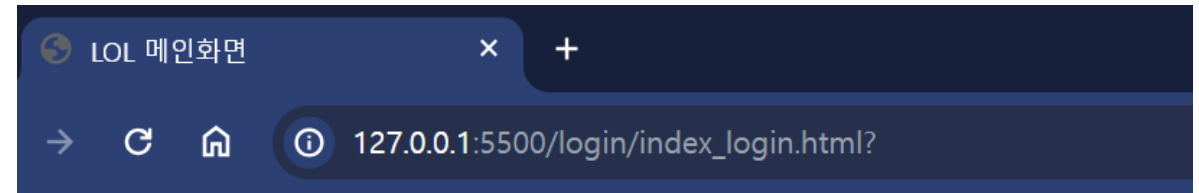
로그인 화면

```
</form>
```

- 로그인을 수행해본다.
  - Get 방식으로 전달, url 주소를 확인
  - 입력한 정보가 주소에 노출된다.

GET	POST
1. URL 상에 정보를 담는다	1. BODY에 정보를 담는다
2. 2048자 까지 담을 수 있다	2. 1MB~2GB 담을 수 있다
3. 정보가 눈에 보인다	3. 정보가 눈에 보이지 않는다
4. 검색, 페이지 상태 저장 유리	4. 로그인에 유리
5. 일반적인 페이지 접속	5. 정보를 저장할 때 이용

form으로 묶여야 하는 영역



# LOL 웹 사이트 - 로그인 폼

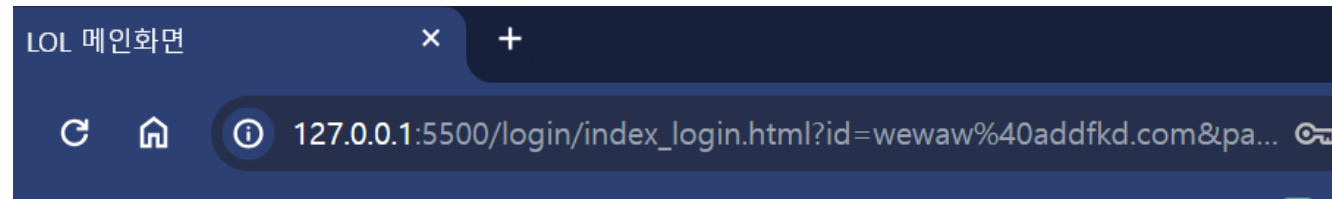
- 데이터 전송을 위해 name 속성 추가
  - name 속성이 없으면 파라미터 전달 x

```
<input type="email" id="typeEmailX" class="form-control form-control-lg" name="id" />  
<label class="form-label" for="typeEmailX">이메일</label>  
</div>
```

```
<div data-mdb-input-init class="form-outline form-white mb-4">  
<input type="password" id="typePasswordX" class="form-control form-control-lg" name="password" />  
<label class="form-label" for="typePasswordX">패스워드</label>  
</div>
```

- 이메일, 패스워드 name 속성을 추가함
  - 표를 참고하여 차이점을 잘 구분
- 
- 로그인 후 다시 확인
    - Url에 입력한 정보가 노출됨

id	유일한 고유 식별자	오직 1개, CSS, JavaScript에서 사용
name	폼 요소의 입력값 식별	폼 내에서 여러 개 사용 가능, 폼 전송 데이터 식별, 레이블 요소와 입력 요소 연결
class	비슷한 특징/스타일 가진 요소 그룹화	하나의 요소에 여러 개 동시에 지정 가능, CSS에서 요소 그룹 스타일 일괄 적용
label	폼 요소의 이름 정의	자동 대상 선택, for 속성으로 대상 연결, 이외 디자인, 가독성 등



# LOL 웹 사이트 - 로그인 폼(공백 체크)

- 이메일, 비밀번호 입력 값의 공백만 체크
  - 공백을 체크 후 로그인 수행 판단
- js 폴더에 login.js 파일을 작성한다.
  - <head> 태그에 <script> 태그로 삽입
  - 화살표 함수 check\_input 구현
    - 폼, 버튼, 이메일, 비밀번호 식별
    - 이메일, 비밀번호의 공백 제거
    - 공백이면 함수 중단
    - 마지막으로 폼을 submit
- 기존 검색 함수와 비교해보자.
  - 화살표 함수 : 하단으로 위치 변경(호이스팅)
  - 하나의 form도 고유 식별자를 사용 가능

```
const check_input = () => {  
  const loginForm = document.getElementById('login_form');  
  const loginBtn = document.getElementById('login_btn');  
  const emailInput = document.getElementById('typeEmailX');  
  const passwordInput = document.getElementById('typePasswordX');  
  
  const c = '아이디, 패스워드를 체크합니다';  
  alert(c);  
  
  const emailValue = emailInput.value.trim();  
  const passwordValue = passwordInput.value.trim();  
  
  if (emailValue === '') {  
    alert('이메일을 입력하세요.');    return false;  
  }  
  
  if (passwordValue === '') {  
    alert('비밀번호를 입력하세요.');    return false;  
  }  
  
  console.log('이메일:', emailValue);  
  console.log('비밀번호:', passwordValue);  
  loginForm.submit();  
};  
  
document.getElementById("login_btn").addEventListener('click', check_input);
```

# LOL 웹 사이트 - 로그인 폼(공백 체크)

- 로그인 화면의 공백 체크를 위한 작업
  - 공백을 순서대로 체크 후 로그인 수행 판단
- login.html 파일을 수정한다.
  - form과 버튼의 id 지정 및 타입 변경

```
<form method="get" action="/login/index_login.html" id="login_form">  
  <div data-mdb-input-init class="form-outline form-white mb-4">  
    <input type="email" id="typeEmailX" class="form-control form-control-lg" name="id" />  
    <label class="form-label" for="typeEmailX">이메일</label>  
  </div>
```

```
  <div data-mdb-input-init class="form-outline form-white mb-4">  
    <input type="password" id="typePasswordX" class="form-control form-control-lg" name="password" />  
    <label class="form-label" for="typePasswordX">패스워드</label>  
  </div>
```

```
  <p class="small mb-5 pb-lg-2"><a class="text-white-50" href="#!">패스워드 찾기</a></p>  
  <button data-mdb-button-init data-mdb-ripple-init class="btn btn-outline-light btn-lg px-5" id="login_btn" type="button">로그인</button>  
</form>
```

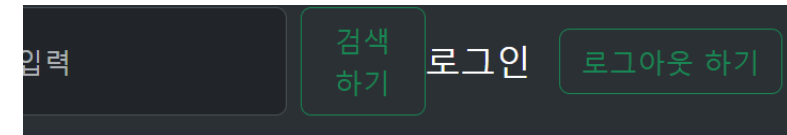
- 작성 후 공백 체크 확인



# LOL 웹 사이트 - 로그인 폼

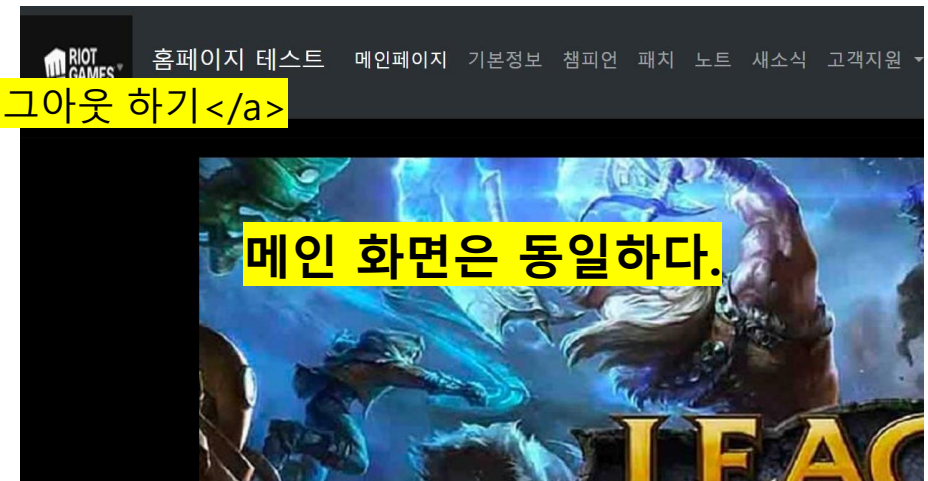
- 로그아웃 버튼과 화면을 구현
  - 로그인 후 화면 및 로그아웃 처리
- login 폴더에 index\_login.html 파일을 작성한다.
  - 기존 index.html 전체 소스코드를 가져온다.
  - Head 및 body부터 footer 등 기본 소스 코드를 유지한다.

## 로그인 후 페이지의 버튼



```
<a href="login/logout.html" class="btn btn-outline-success" id="logout_btn">로그아웃 하기</a>
```

- 버튼을 수정하고, 로그 아웃 페이지를 연결



# 응용 문제 풀기 – NOW!!!!

- login 폴더에 logout.html 파일을 작성한다.
  - 기존 index\_login.html 전체 소스코드를 가져온다.
    - Head 및 body부터 footer 등 기본 소스 코드를 유지한다.
  - 그림과 같은 로그아웃 화면으로 수정한다.
- 로그아웃 버튼 기능
  - 클릭하면 index.html로 연결한다.
  - 원래 사이트 메인으로 돌아온다.



## 로그아웃 메인화면

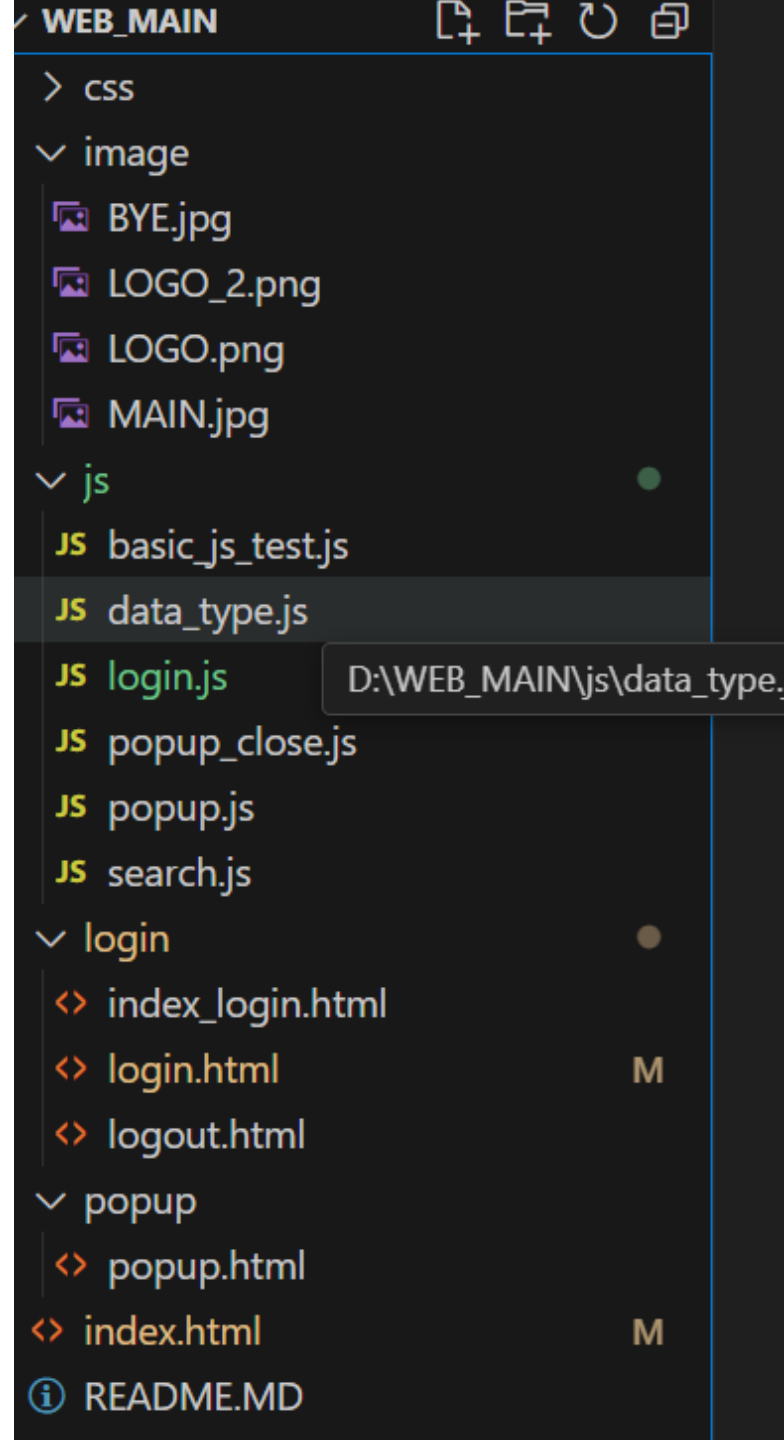
로그아웃

- 6주차 연습문제

Js 및 html 정리

# JS 및 HTML 정리하기

- 로그인, 로그아웃 등 팝업창 확인
  - Head 부분의 팝업창 **.js** 연동 부분을 모두 제거한다.
    - Index.html 이외 화면은 팝업이 필요 x
- 여러개의 .html 화면이 추가됐다.
  - 폴더 및 파일 체크
    - 기본 부트스트랩, head, footer 등 소스코드 일관성 확인
  - 소스 코드 정렬 확인
    - Vs의 들여쓰기 정렬 확인
    - 정렬 도우미
      - <https://prettydiff.com/tool.xhtml>
- 실습 결과 확인 – Q / A



# Q & A

- 다음주 할일
  - LOL 웹 사이트 구현(JS)
    - 로그인 창 : 입력 필터링(확장)

- 무엇이든 물어보세요!

