

9주차 강의

자바웹프로그래밍(1)

강사 : 최도현

트렌드 분석

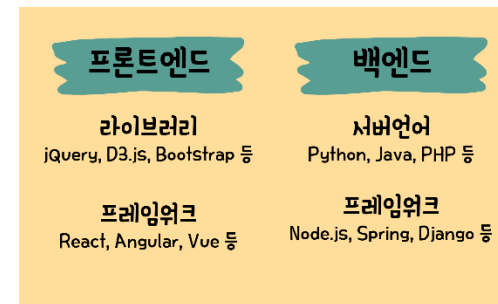
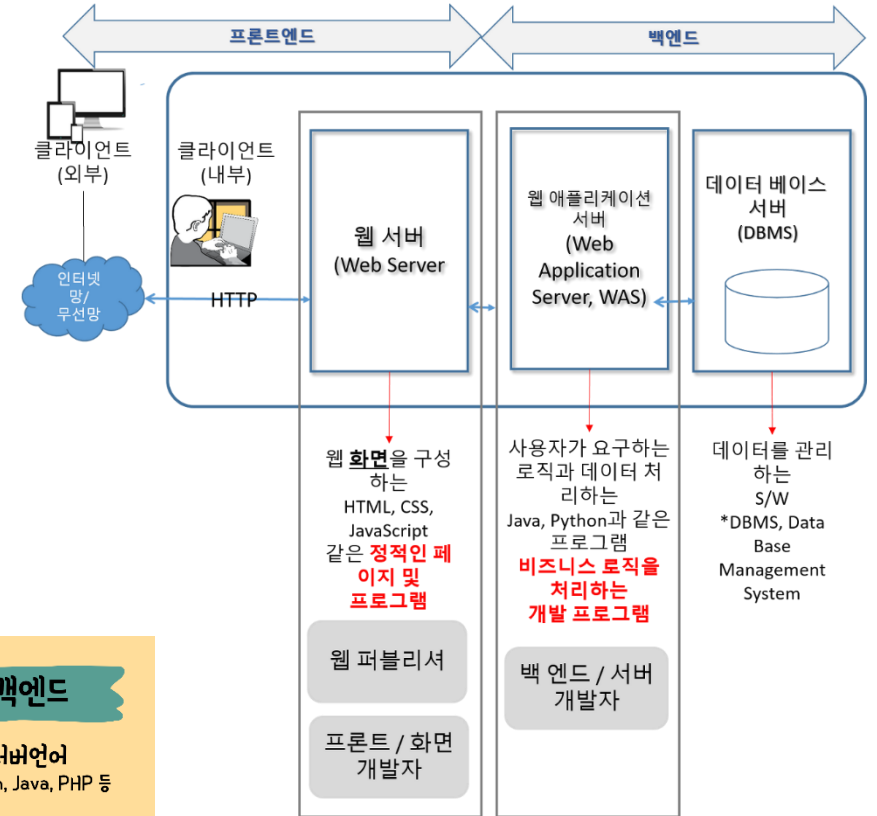
•웹 서비스 및 개발 트렌드

기술 트렌드 - 개발 및 보안



풀스택 개발자

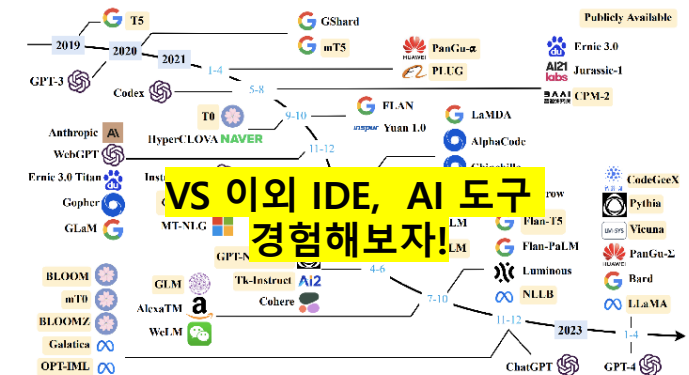
- 중간고사 앞서 테크 트리 체크
 - 프론트 중심 : HTML, JAVASCRIPT 등
 - 이후 : REACT, FLUTTER
 - 기초 상식 및 개발환경 경험
 - 목적 : 화면 개발
- 2학기 이후(백엔드) ← **선택의 시기**
 - 백엔드 : PHP, FLASK, JAVA(SPRING) 등
 - 데이터베이스 제어 : SQL 쿼리, 함수 조합
 - 서버 기술 : 서버 운영체제 구축/운영
 - 목적 : 핵심 기능 개발
- 기타 필수
 - AI 프로그래밍 활용
 - 깃 허브 등 개발자 필수 도구



웹, VS PC앱 VS 모바일(앱/웹)

무엇을 개발할 것인가?

개발 환경 조합 + 프레임워크 선택



입력 값 필수 보안

- 대표 웹 해킹 공격 2가지

- XSS: 악성 코드를 페이지에 삽입하여 웹 브라우저를 공격**

- 사용자 정보 탈취 : 개인정보, 로그인 정보 등
 - 해결책 : 입력 데이터 검증, 서버 보안 설정

- CSRF: 사용자 인증 정보를 악용하여 비정상 요청을 웹 서버에 전송**

- 인증 정보 탈취 : 로그인된 상태 정보 등
 - 해결책 : REFERER 검증, CSRF 토큰 구현, 서버 보안 설정

- 공격의 위험성

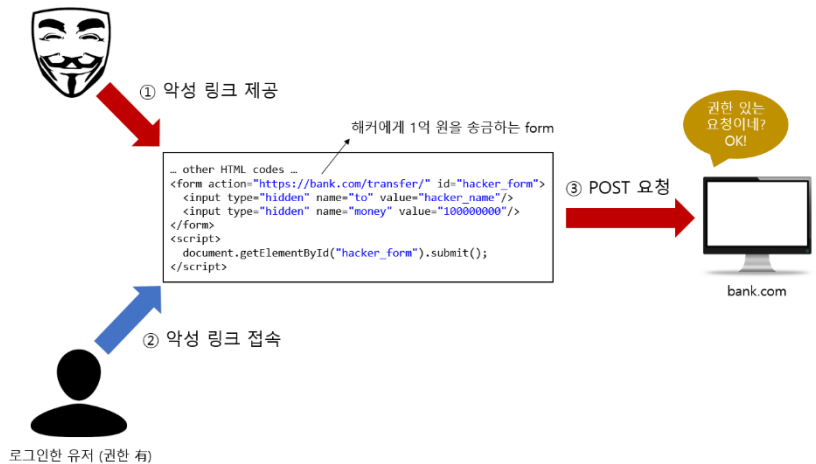
- XSS : 웹 브라우저 < CSRF : 웹 서버**

- 잘 알려진 DDOS의 원인이 될 수 있음

- DDOS : Distributed Denial of Service**

- 웹 서비스 중단 목적

	XSS	CSRF
공통점	악성 스크립트 사용 (단, CSRF는 악성 스크립트 없이도 가능)	
차이점	세션 탈취가 목적 사용자를 주로 공격 스크립트 실행이 목적	사이트간 요청 위조 서버를 주로 공격 특정 행동을 유도



PART1

• 자바스크립트 - 입력 필터링

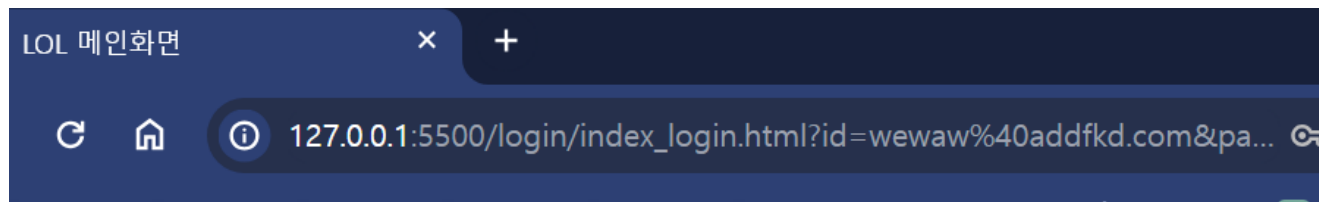
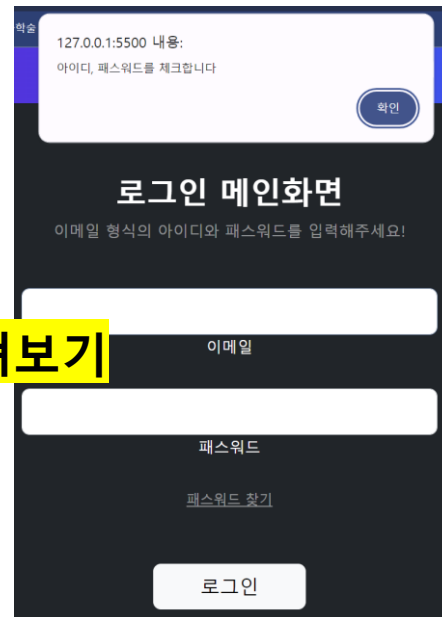
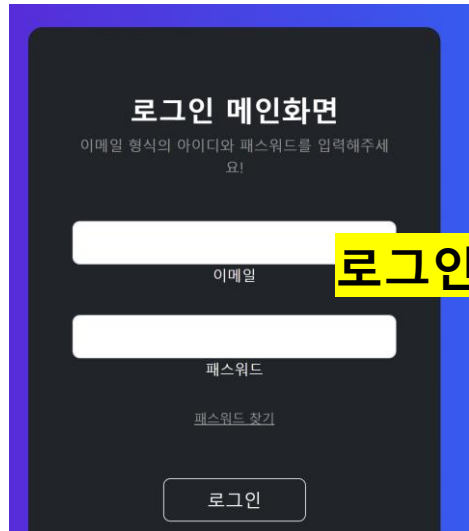
로그인 폼(길이 제한과 특수문자)

로그인 폼(XSS 방지)



지난주 내용 살펴보기

- 로그인 폼 및 JS 추가 기능
 - 기본 form 및 입력 필터링



항목	태그 이름/설명
Es6 버전 화살표 함수에서 사용 제한된 함수는?	
기존 function이나 var 등 선언 위치에 상관없이 인식하는 기능은?	
Form 전송 시에 get 방식으로 url에 파라미터를 전송하기 위한 추가 속성은?	
Label 태그의 주요 기능은 무엇인가?	
입력 값의 공백을 제거하는 함수 이름은?	
Form도 이름을 지정하여 하나의 식별자로 정의할 수 있는가?	
자바스크립트 내부에서 submit 하는데 html에서 버튼의 type은?	
값 비교에 ===는 무엇을 더 검사하는가?	

로그인 폼(입력 길이와 특수문자)

- 이메일, 비밀번호 입력 값의 길이를 체크
 - 기존 입력 필터링 조건을 확장
 - 자바의 정규표현식(regular expression) 활용
- js 폴더에 login.js 파일의 check_input 함수를 수정한다.
 - if문으로 조건의 순차 처리
 - 문자열의 길이 length 함수 처리
 - 특수 문자 및 대/소문자 체크
 - match 함수는 없으면 null 리턴
 - / / 범위의 문자열 찾는 정규표현식

```
if (emailValue.length < 5) {  
    alert('아이디는 최소 5글자 이상 입력해야 합니다.');
```

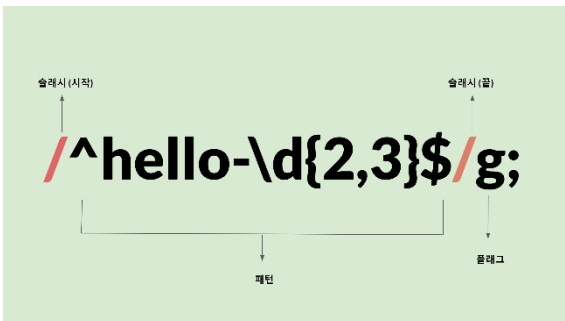
```
}  
  
if (passwordValue.length < 12) {  
    alert('비밀번호는 반드시 12글자 이상 입력해야 합니다.');
```

```
    return false;  
}  
  
const hasSpecialChar = passwordValue.match(/[!@#$%^&*()_+~\-=\[\]{};':"\\|,.<>\/?]+/) !== null;  
if (!hasSpecialChar) {  
    alert('패스워드는 특수문자를 1개 이상 포함해야 합니다.');
```

```
    return false;  
}  
  
const hasUpperCase = passwordValue.match(/[A-Z]+/) !== null;  
const hasLowerCase = passwordValue.match(/[a-z]+/) !== null;  
if (!hasUpperCase || !hasLowerCase) {  
    alert('패스워드는 대소문자를 1개 이상 포함해야 합니다.');
```

```
    return false;  
}  
  
127.0.0.1:5500 내용:  
패스워드는 대소문자를 1개 이상 포함해야 합니다.
```

확인



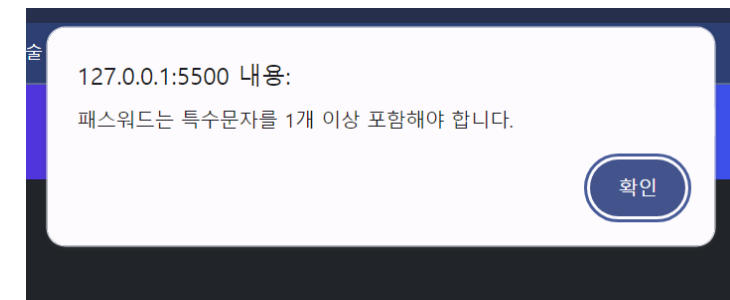
기호	설명
[]	한 개의 문자
[abc]	a,b,c 중 하나의 문자
[^abc]	a,b,c 이외 하나의 문자
[a-zA-Z]	a~z, A~Z 중 하나의 문자
\d	한 개의 숫자, [0-9]와 동일
\s	공백
\w	한 개의 알파벳 또는 한 개의 숫자 [a-zA-Z_0-9]와 동일
?	없음 또는 한 개
*	없음 또는 한 개 이상
+	한 개 이상
{n}	정확히 n개
{n,}	최소한 n개
{n, m}	n개에서부터 m개까지
()	그룹핑

로그인 폼(특수 문자)

- 어떤 특수 문자까지 필터링 해야 할까?
 - Why : 서버 내부 시스템 명령을 악의적으로 수행 가능
- 특수 문자 목록 예)
 - 백슬래시 (\): 쉘 스크립팅과 같은 특정 명령어 해석
 - 백틱 (`): 쉘 명령 실행을 위한 특수 문자
 - 쌍따옴표 ("): 문자열을 나타내는 데 사용되는 문자
 - 홑따옴표 ('): 쌍따옴표와 동일
 - 퍼센트 (%): URL 인코딩 및 쉘 스크립팅에서 사용
 - 물음표 (?): URL 검색 매개변수를 나타내는 데 사용
 - 앰퍼샌드 (&): 논리 연산자로 쉘 스크립팅에서 사용
 - 파이프 (|): 쉘 스크립팅에서 명령 실행 순서를 나타냄
 - 쉼표 (,): CSV 파일 등 데이터 구분 등
 - 세미콜론 (;): 쉘 스크립팅에서 명령 구분
- JS 구현 상의 어려움은?
 - 복잡한 특수문자 조합을 모두 인식하지 못한다.




다양한 특수문자 존재

No	특수기호	영어	영어 읽기	한글
1	~	TILDE	틸드	물결표
2	!	EXCLAMATION POINT	익스클레메이션 포인트	느낌표
3	?	QUESTION MARK	퀘스천 마크	물음표
4	@	AT SIGN	앳 사인 or 앳	앳,골뱅이
5	#	SHARP	샵	우물표시,우물정
6	\$	DOLLAR SIGN	달러 사인	달러
7	%	PERCENT SIGN	퍼센트 사인	백분표
8	^	CARET, CIRCUMFLEX	캐럿, 써큘플렉스	삿갓,모자,윗귀쇠
9	&	AMPERSAND	앰퍼샌드	앰드
10	*	ASTERISK	아스테리스크	별표,눈표,곱하기
11	_	UNDERSCORE	언더스코어	밑줄
12	+	PLUS SIGN	플러스 사인	더하기, 십자
13	-	HYPHEN, MINUS SIGN	하이픈, 마이너스	불임표, 빼기
14	=	EQUAL SIGN	이퀄 사인	같음표
15	,	COMMA	콤마	쉼표,반점
16	.	PERIOD OR DOT	피리어드 또는 닷	마침표,온점
17	(LEFT PARENTHESIS	레프트 퍼엔터시스	소괄호 열기
18)	RIGHT PARENTHESIS	라이트 퍼엔터시스	소괄호 닫기
19	{	BRACES	브레이스	중괄호



로그인 폼(XSS 방지)

- 이메일, 비밀번호 입력 값의 스크립트 체크
 - 기본 : 보안이 문제가 되는 특수문자
 - 추가 : 입력 가능한 모든 경우의 스크립트
- 기존 login.html 파일을 수정한다.
 - DOMPurify 라이브러리 활용
 - <https://cdnjs.com/libraries/dompurify>
 - <head> 태그에 <script> 태그로 삽입
- js 폴더에 login.js 파일의 check_xss 함수를 추가한다.
 - Check_input 함수 앞에 추가한다.
 - DOMPurify를 라이브러리 로드 후 초기화
 - Sanitized 함수
 - 특수문자를 안전한 문자형태로 변환
 - <script> <iframe> <svg> 태그 제거 등
 - 스타일 속성, url 형식 등 다양한 정보 제거

<https://cdnjs.cloudflare.com/ajax/libs/dompurify/3.2.5/purify.min.js>   

```
const check_xss = (input) => {  
  // DOMPurify 라이브러리 로드 (CDN 사용)  
  const DOMPurify = window.DOMPurify;  
  
  // 입력 값을 DOMPurify로 sanitize  
  const sanitizedInput = DOMPurify.sanitize(input);  
  
  // Sanitized된 값과 원본 입력 값 비교  
  if (sanitizedInput !== input) {  
    // XSS 공격 가능성 발견 시 에러 처리  
    alert('XSS 공격 가능성이 있는 입력값을 발견했습니다.');    return false;  
  }  
  
  // Sanitized된 값 반환  
  return sanitizedInput;  
};
```



안전한 HTML 형태로 변환

로그인 폼(XSS 방지)

- js 폴더에 login.js 파일의 check_input 함수 내부에 추가한다.
 - 앞서 작성된 check_xss 함수를 활용
 - 이메일, 패스워드 모두 xss 체크
- 직접 xss 악성 스크립트 코드 1개를 입력해보자.
 - ``

XSS 공격을 위한 임시 테스트 입력 값 (주의: 실제 환경에서는 사용 금지!)

1. 일반적인 XSS 공격: ``

2. 특수 문자를 활용한 XSS 공격: `javascript:alert('XSS 공격 발생!');/*`

`\"><script>alert('XSS 공격 발생!');</script>`

`<svg onload="alert('XSS 공격 발생!');"></svg>`

3. DOM 기반 XSS 공격: `<div onclick="alert('XSS 공격 발생!');">클릭하세요</div>`

`<button onclick="javascript:alert('XSS 공격 발생!');">버튼 클릭</button>`

`링크 방문`

4. HTML5 이벤트를 활용한 XSS 공격: `<input type="text" onfocus="alert('XSS 공격 발생!');">`

`<input type="password" onmouseover="alert('XSS 공격 발생!');">`

`<input type="file" onchange="alert('XSS 공격 발생!');">`

5. CSS를 활용한 XSS 공격: `<style>@import url("javascript:alert('XSS 공격 발생!');"");</style>`

`<style>body { background-image: url("javascript:alert('XSS 공격 발생!');""); }</style>`

6. 데이터 URI를 활용한 XSS 공격: `<img`

`src="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAQAAAC1HAWCAAAAC0I`

`EQVR42mNkYAAAAAAYAAjCB0C9AAAAASUVORK5CYII=" onerror="alert('XSS 공격 발생!');">`

`<a href="data:text/html;charset=utf-8,<script>alert('XSS 공격 발생!');">링크 방문`

```
const sanitizedPassword = check_xss(passwordInput);
```

```
// check_xss 함수로 비밀번호 Sanitize
```

```
const sanitizedEmail = check_xss(emailInput);
```

```
// check_xss 함수로 비밀번호 Sanitize
```

```
if (!sanitizedEmail) {
```

```
// Sanitize된 비밀번호 사용
```

```
return false;
```

```
}
```

```
if (!sanitizedPassword) {
```

```
// Sanitize된 비밀번호 사용
```

```
return false;
```

```
}
```

로그인 메인화면

이메일 형식의 아이디와 패스워드를 입력해주세요!

``

이메일

패스워드

패스워드 찾기

로그인

응용 문제 풀기 – NOW!!!!

- js 폴더에 login.js 파일의 check_input 함수를 수정한다.
 - 로그인 입력 길이 제한
 - 이메일 10글자 이하, 비밀번호 15글자 이하 수정
 - 로그인 입력 제한(패턴식 활용)
 - 3글자 이상 반복 입력 x
 - 예) 아이디는아이디, 123123
 - 연속되는 숫자 2개 이상 반복 입력 x
 - 예) 12아이디12



트렌드 분석

•웹 서비스 및 개발 트렌드

기술 트렌드 – 개발 및 보안

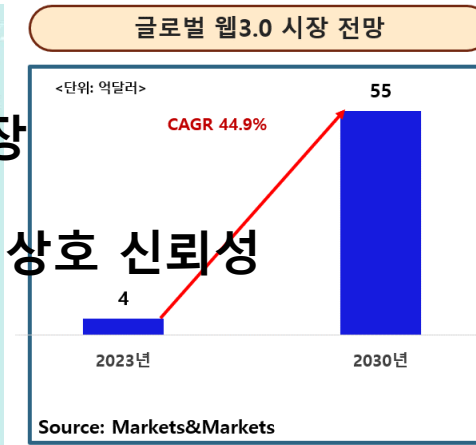


WEB 3.0과 데이터 저장

- 웹의 현재와 미래 - 데이터 저장 관점
 - 전통적(웹 2.0) : 서버 측 파일 시스템, DB(RDB, 관계형)
 - 최근 비정형 데이터베이스 NOSQL(대규모)
 - 발전 → 분산된 서버에 저장(클라우드)
 - 결국, 중앙집중형 구조의 한계, 저장 플랫폼 의존(구글 등)
- 블록체인 클라우드 기술 → 탈중앙화
 - 개인화된 저장 공간 = 분산된 네트워크 저장
 - 주체가 플랫폼이 아닌 네트워크 즉, 다수의 개인
 - 결제 관련 기능을 대체할 것으로 예상, HOW?

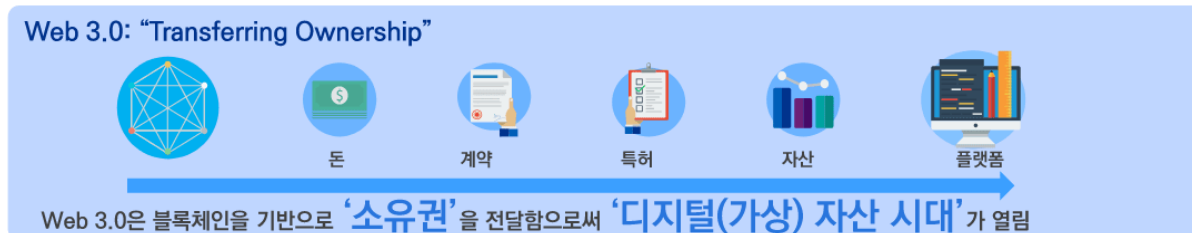
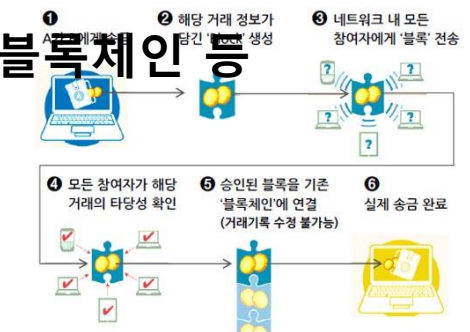
웹 3.0 개념 비교

	Web 1.0	Web 2.0	Web 3.0
데이터를 분산 저장			
사용 특징	읽기	읽기·쓰기	읽기·쓰기·소유
사용 방식	접속 및 검색	콘텐츠 제작 및 공유	데이터 소유 및 거래
인프라	개인 PC	클라우드	블록체인 네트워크
주요 사례	포털 사이트	SNS	NFT(대체불가능토큰), 디파이(탈중앙화 금융), DAO(탈중앙화 자율조직)
대표 기업	구글, 야후	페이스북, 유튜브	대퍼랩스, 골드핀치, 메타팩토리



주요 게임사 블록체인 사업 현황

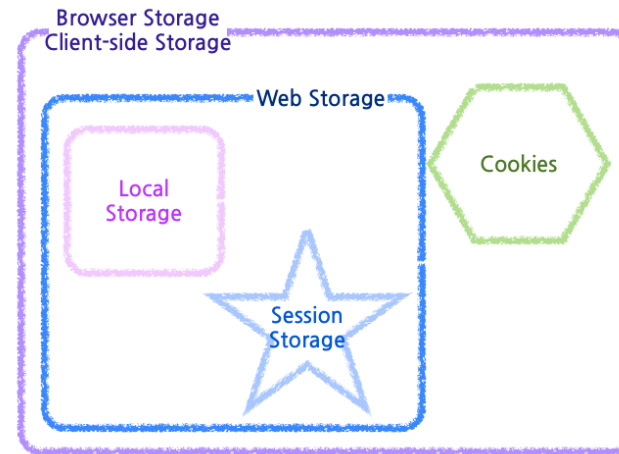
기업	블록체인 플랫폼	주력 게임
넥슨	-	메이플스토리 유니버스 (준비 중)
위메이드	메이플스토리	메이플스토리, 메이플스토리 2, 메이플스토리 3
컴투스홀딩스	엑스플라	붕어방타이콘, 미니게임천국, 워킹데드
넷마블	마브렉스	마브렉스, 마브렉스 A3: 스틸얼라이브
카카오게임즈	블록	아카원도, 버디산
네오위즈	인텔라X	준비 중
크래프톤	-	준비 중 (메타버스 오버더어)
NC	-	연구 단계
NHN	-	준비 중 (미스틴랩스 파트너십)



HTML5의 데이터 저장

• 현재 WEB 2.0 기준 웹 환경의 데이터 저장?

- 8주차 실습까지 내용을 생각해보자.
 - 응용 프로그램 실행 중에 저장 공간 = 메모리
 - 사실 중요한 대부분 정보는 서버에 저장
- 클라 - 브라우저 저장소에 저장할 수 있다.
 - 웹 스토리지 내부 - 로컬, 세션
 - 일반 로컬(내 c:/) - 쿠키
- 특징 및 차이점
 - 웹 스토리지 html5 부터 도입
 - 기존에는 언어 별 따로 구현 했다. 매우 불편
 - 위치의 차이 : 쿠키는 로컬, 세션은 서버측
 - 쿠키 → 로컬, BUT 항상 서버로 전송됨
- 서버 내부, 보안설정에 따라 구분하여 구현
 - 오늘 실습에서 쿠키를 구현해보자.



로컬, 세션 등 저장소 활용 가능

	쿠키 Cookie	로컬 스토리지 Local Storage	세션 스토리지 Session Storage
매번 요청마다 서버로 전송?	O	X	X
용량제한	4KB	모바일: 2.5MB 데스크탑: 5MB ~ 10MB	모바일: 2.5MB 데스크탑: 5MB ~ 10MB
어떻게 얻나요	document.cookie	window.localStorage	window.sessionStorage
영구적인가?	유효기간이 있음	사용자가 지우지 않는 한 영구적	윈도우나 브라우저 탭을 닫으면 제거
저장방식	Key-value	Key-value	Key-value

쿠키 : 보안 X, 일반 정보에 활용

참고: <https://www.zerocho.com/category/HTML&DOM/post/5918515b1ed39f00182d3048>

PART1

- 데이터 저장(쿠키)

데이터 저장 - 쿠키(팝업창)



Cookie

지난주 내용 살펴보기

- 로그인 폼 및 JS 추가 기능
 - 기본 form 및 입력 필터링(xss)

127.0.0.1:5500 내용:
패스워드는 대소문자를 1개 이상 포함해야 합니다.

확인

로그인 폼과 js 소스코드 살펴보기

127.0.0.1:5500 내용:
패스워드는 특수문자를 1개 이상 포함해야 합니다.

확인

로그인 메인화면
이메일 형식의 아이디와 패스워드를 입력해주세요!

이메일

패스워드

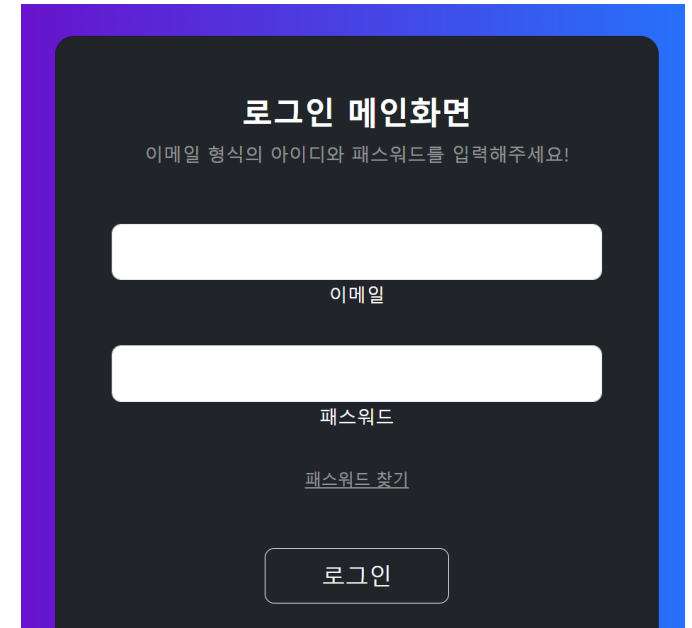
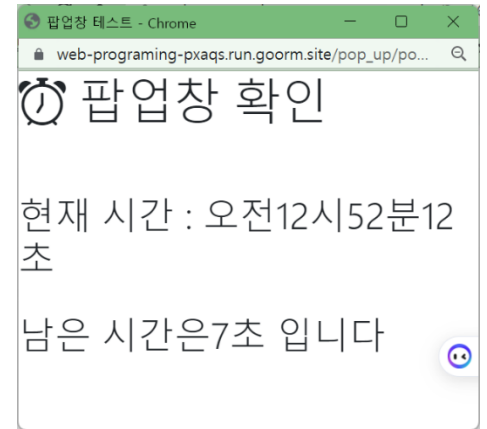
패스워드 찾기

로그인

항목	태그 이름/설명
정규표현식의 문자열을 검사하는 함수는?	
웹 스크립팅에서 명령 실행 순서를 나타내는 특수 문자는?	
정규표현식에서 한 개 이상을 나타내는 기호 표현은?	
특수문자를 필터링 해야 하는 이유는?	
DOMPurify 라이브러리의 핵심 검사 함수는?	
웹 스크립팅에서 명령 구분 명령을 구분하는 특수 문자는?	

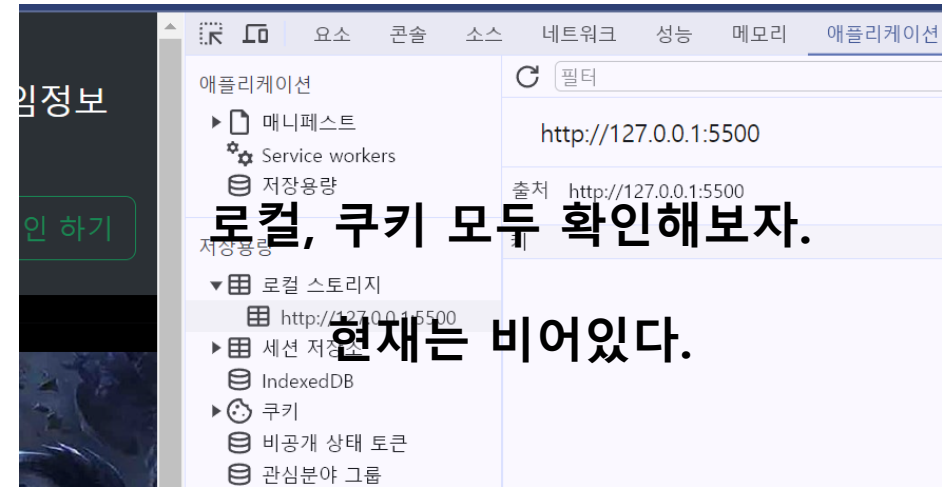
내 홈페이지 확인

- 생각해보자. 데이터를 저장에 필요한 곳은?
 - 특정 정보를 일정기간 동안 저장
 - 즉, 지속하여 유지해야 하는 경우
- 일반적인 웹 사이트 기능 예)
 - **팝업 페이지(오늘 구현 해보자)**
 - X 일 동안 팝업을 보이지 않기
 - 로그인 id
 - 이전에 입력한 id를 기억한다.
 - 로그인 유지 기능(세션)
 - 메인화면을 열어도 로그인 상태로 열림
 - 일정 시간 이후 자동 로그아웃



내 홈페이지 확인

- 내 홈페이지에서 확인(개발자 모드 F12)하기
 - 애플리케이션 탭 정보 확인
 - Local/session storage
 - Cookie
- 구글 또는 유튜브 예) Google.com
 - 페이지마다 다르게 정보를 저장
 - 3가지 저장소를 모두 활용 가능
 - 키, 값으로 이루어져 있음
- 저장소 역할
 - 일반적인 정보들을 저장



데이터 저장 - 쿠키(팝업창 - X일 보지 않기)

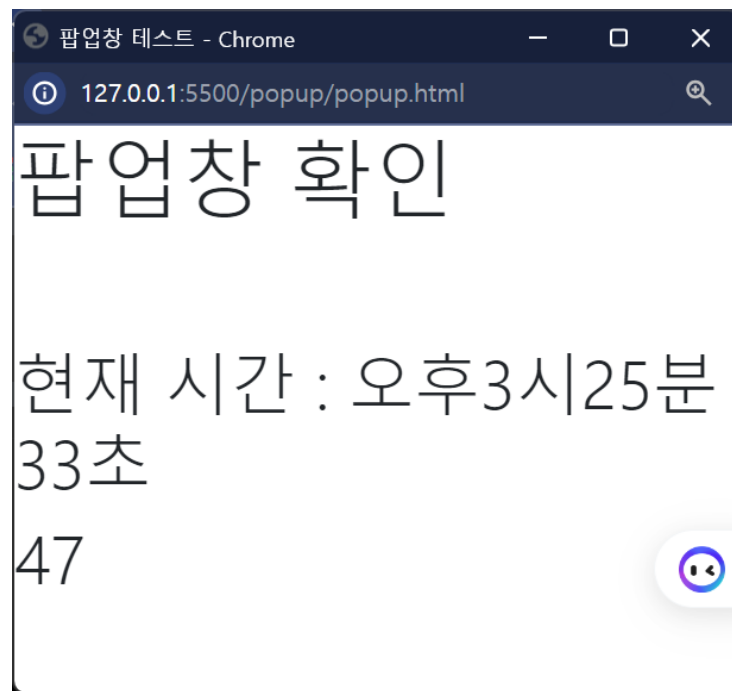
- js 폴더의 popup_close.js를 수정한다.
 - 팝업창 확인을 위해 시간을 조정한다.
 - 기존 페이지 닫기 시간 10초를 50초로 변경

```
var close_time; // 시간 정보
var close_time2 = 50; // 10초 설정

clearTimeout(close_time); // 재호출 정지
close_time= setTimeout("close_window()", 50000);
// 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
}
```

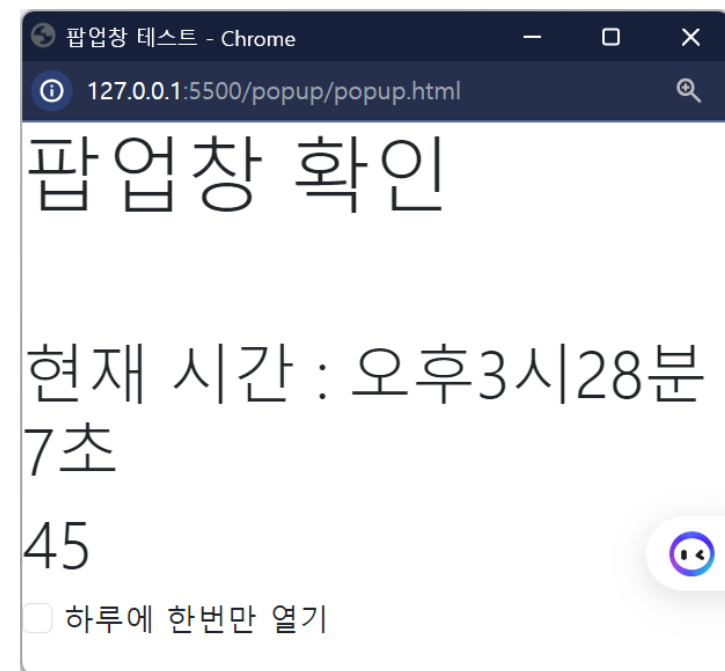


데이터 저장 - 쿠키(팝업창 - X일 보지 않기)

- popup 폴더의 pop_up.html 을 수정한다.
 - Pop_up.html에 체크 박스를 추가한다.
 - 클릭하면 함수 호출, 쿠키를 생성한다.

```
<input class="form-check-input" type="checkbox" id="check_popup" onclick="closePopup();" >  
<label class="form-check-label" for="flexCheckChecked">하루에 한번만 열기</label>
```

- 메인 페이지 새로고침, 팝업을 열어보자.
 - 최하단 체크 박스 확인
- 추가적으로 구현해야 할 것들?
 - 체크 박스, 클릭하면 쿠키 생성 구현
 - 자동 창이 닫히는 함수 구현

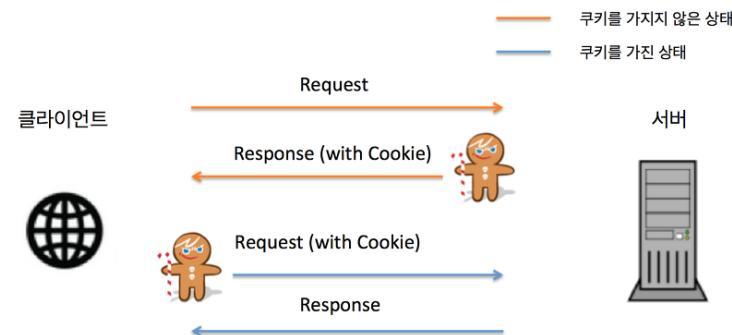


데이터 저장 - 쿠키(팝업창 - X일 보지 않기)

- js 폴더의 pop_up.js를 수정한다.
 - 메인페이지는 이제 쿠키의 유무를 체크 한다.

```
function pop_up() {  
    var cookieCheck = getCookie("popupYN");  
    if (cookieCheck != "N"){  
        window.open("../popup/popup.html", "팝업테스트", "width=400, height=300, top=10, left=10");  
    }  
}
```

- 쿠키에 값이 N이 없는 경우
 - 팝업 시에는 쿠키가 없다.
 - 무조건 윈도우를 새로 연다.
- 수정 후? 팝업이 열리지 않는다.
 - 쿠키의 저장을 추가 구현해보자.



```
Cookies. ( 'name', 'value');
```

```
Cookies. ( 'name', 'value', { expires: 7 });
```

JS는 SET / GET 메소드 지원

```
Cookies. ( 'name'); // => 'value'
```

```
Cookies. (); // => { name: 'value' }
```

데이터 저장 - 쿠키(팝업창 - X일 보지 않기)

- Pop_up.js에 추가 기능을 구현한다.

- 쿠키를 SET 하는 함수

- 현재 시간 기준 - date
 - setDate 함수는 시간을 설정
 - Expiredays 시간 설정 추가
 - getDate 함수는 UtC 표준 날짜 리턴

- 쿠키를 GET 하는 함수

- 쿠키를 얻는다. 존재하면
 - 배열 반복하여 내부에 =을 제외한
 - popupYN 을 찾아 값을 리턴
 - 참고 : 쿠키는 키, 값 으로 이루어짐
 - 즉, 값은 인덱스 [1]이 된다.

```
function setCookie(name, value, expiredays) {  
    var date = new Date();  
    date.setDate(date.getDate() + expiredays);  
    document.cookie = escape(name) + "=" + escape(value) + ";  
    expires=" + date.toUTCString() + "; path=/";  
}
```

```
function getCookie(name) {  
    var cookie = document.cookie;  
    console.log("쿠키를 요청합니다.");  
    if (cookie != "") {  
        var cookie_array = cookie.split("; ");  
        for ( var index in cookie_array) {  
            var cookie_name = cookie_array[index].split("=");
```

```
            if (cookie_name[0] == "popupYN") {  
                return cookie_name[1];  
            }  
        }  
    }  
    return ;  
}
```

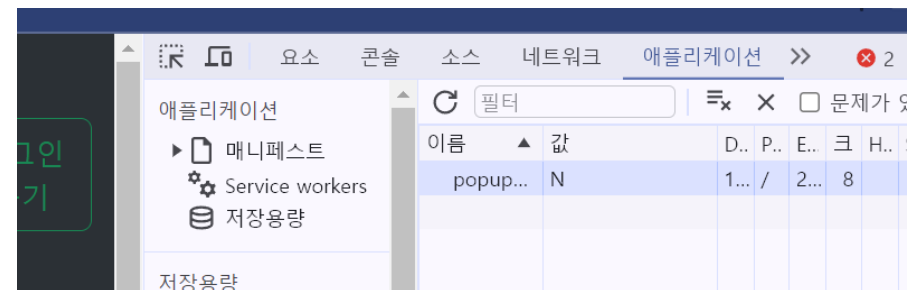
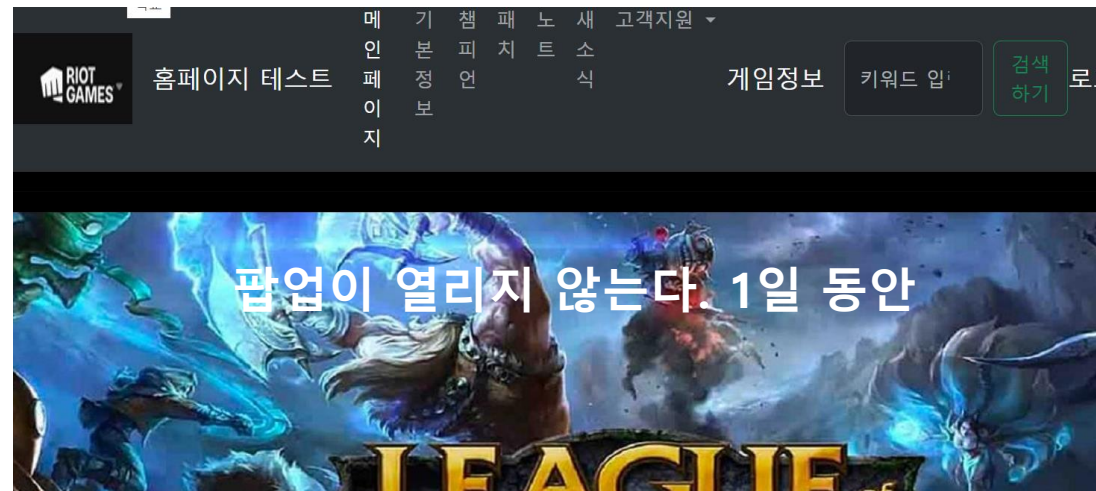
데이터 저장 - 쿠키(팝업창 - X일 보지 않기)

- Pop_up.js에 추가 기능을 구현한다.
 - 체크 박스 클릭 시 윈도우 닫는 함수
 - Id 값이 존재하면(처음에는 없다)
 - 클릭 이후에 쿠키를 set

```
function closePopup() {  
    if (document.getElementById('check_popup').value) {  
        setCookie("popupYN", "N", 1);  
        console.log("쿠키를 설정합니다.");  
        self.close();  
    }  
}
```

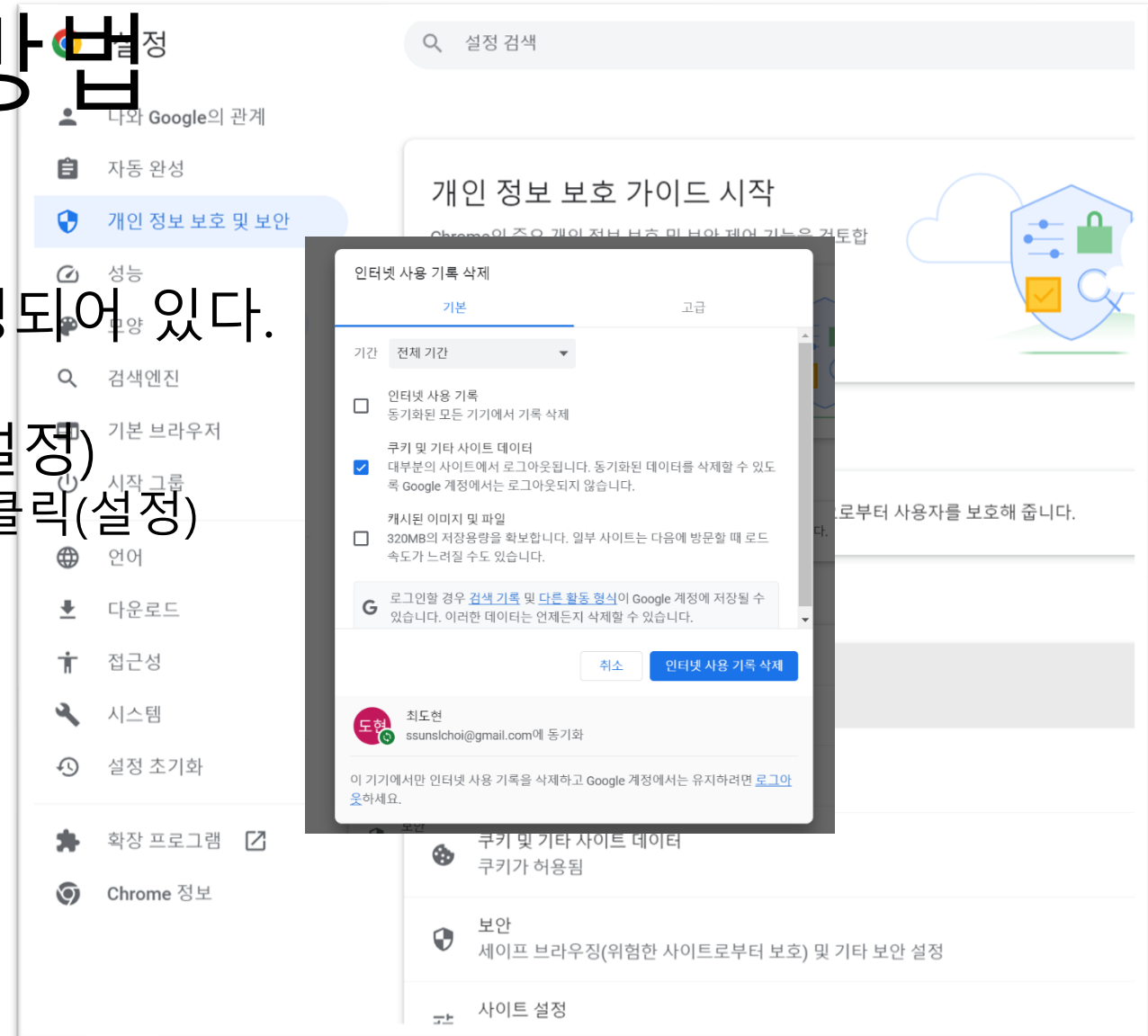
- Self.close()로 창을 닫는다.
 - 현재 보고 있는 창 자신

- F12 개발자 모드에서 각 필드의 값을 확인한다.



잠깐! 쿠키값 삭제 방법

- 쿠키 기능 구현중 초기화 방법
 - 쿠키는 지정된 시간동안 계속 생성되어 있다.
- 익스플로러, 크롬 등 웹브라우저(설정)
 - 크롬 기준 : 오른쪽 상단 계정 옆 클릭(설정)
 - 개인정보 보호 및 보안 탭
 - 쿠키 및 기타 사이트 데이터 체크!
 - 인터넷 사용 기록 삭제 클릭
- 쿠키가 삭제됨
 - 메인 화면으로 돌아가 확인해보자.



데이터 저장 - 쿠키(팝업창 - X일 보지 않기)

- 구글 크롬 기준

- 2020년 1월 이후 80버전 이후 기준 보안 설정
 - 쿠키에 대한 보안 정책과 동작 방식 관련 변경
 - 보안 개선 방안 - same site 속성 사용

- 크로스 사이트 취약점 보안 설정 추가

- 쿠키를 SET 하는 함수를 수정한다.

```
document.cookie = escape(name) + "=" + escape(value) + "; expires=" + date.toUTCString() + ";"  
path="/" + ";SameSite=None; Secure";
```

- 기존 쿠키를 삭제하고 다시 진행

- Secure가 체크된다.

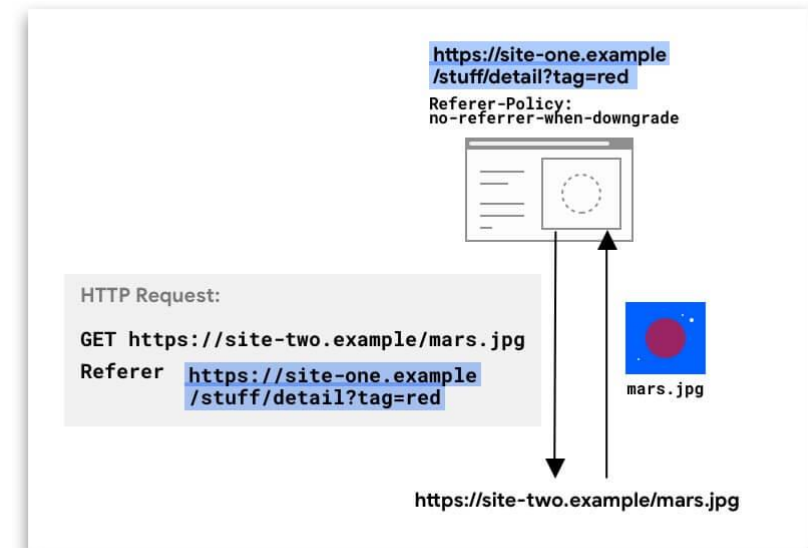
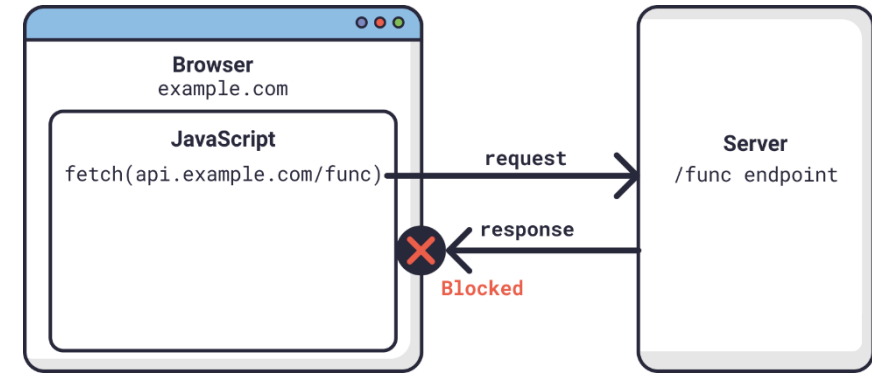
쿠키의 사용 범위를 제한하기 위한 쿠키 옵션. (외부 Access방지, java의 private public같은 접근제한자 역할)

종류	(접근정책 유연>강력 순)
None	도메인 검증하지 않음.(타사에서 접근가능) secure옵션(https만 접근가능)을 설정해야한다. <small>우 None으로 처리됨.</small>
Lax	자사도메인이 아니더라도 일부케이스(링크를 클릭해 이동하는 경우 등)에서는 접근을 허용하겠다. (상태를 변경하는 요청인 POST요청시 접근 불가)
Strict	자사 도메인으로만 쿠키전송 가능 (현재 브라우저의 URL과 쿠키의 도메인 일치하는경우)

이름	값	Domain	Path	Expires / M...	크기	HttpOnly	Secure	SameSite	Partition Key
popupYN	N	127.0.0.1	/	2024-05-0...	8		✓	None	

잠깐! 쿠키 : CORS 정책과 보안

- 최근 분산된 서버에 데이터 처리
 - 특정 URL에서 사용중인 자원을 다른 서버에서 접근 가능
 - 실제 접근 권한을 임시로 부여
- 쿠키와 어떤 관계가?
 - 주소 : `http://127.0.0.1:5500/` 요청에 대해서만 쿠키를 처리
 - 즉, 쿠키도 같은 경로에서만 통신해야 정책을 준수
- 크롬, 익스 등은 최신 업데이트에서 이를 차단
 - CSRF, XSS 해킹 공격에 취약하기 때문이다.
 - 쿠키, 세션 ID 획득 가능, 악성코드 실행 및 페이지 조작 가능
 - 항상 원본 도메인 주소를 검사한다. (Referer CHECK)
- 참고
 - 실제 서버를 운영할 때는 검사 기능 구현 필요
 - CORS 해제 방법이 있다. (웹 서버 직접 설정)



- 9주차 연습문제 / Q & A

전체 문제 체크

개인 문제 풀기 / Q & A

- **개인 문제 진행도 확인(오늘 체크)**
 - 3주차 - 부트스트랩 디자인 추가 수정
 - 4주차 - 자바스크립트(식별자 수정)
 - 5주차 - 자바스크립트(검색창 공백/비속어 검사)
 - 6주차 - 로그아웃 화면(메인화면으로 이동), 소스코드 정리
- **깃 허브 업로드 내역 확인**
 - Readme.md도 한번 정리하자.
- **다음주 할 일**
 - LOL 웹 사이트 구현(JS)
 - 로그인 창 : 쿠키 및 세션
- **무엇이든 물어보세요!**

