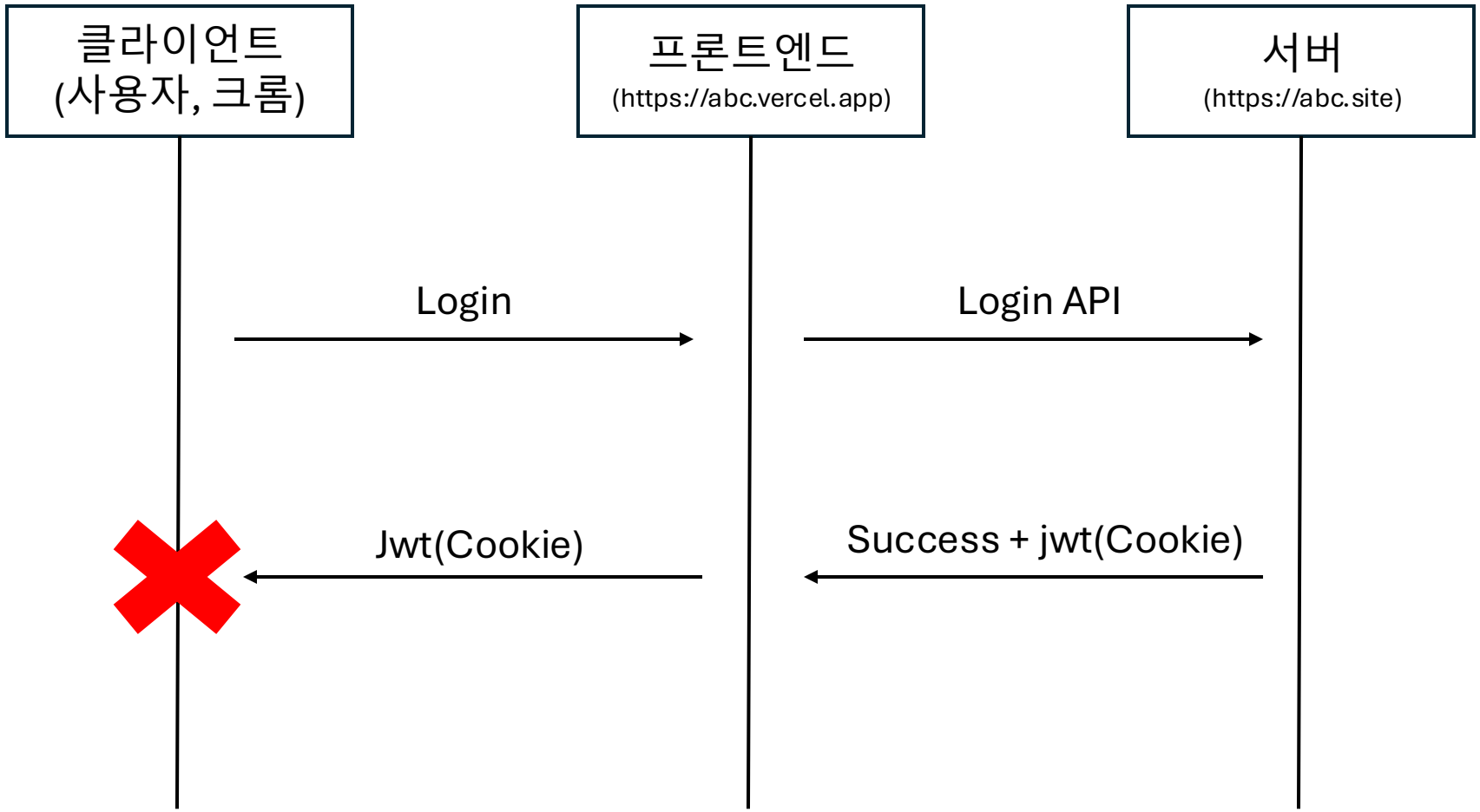
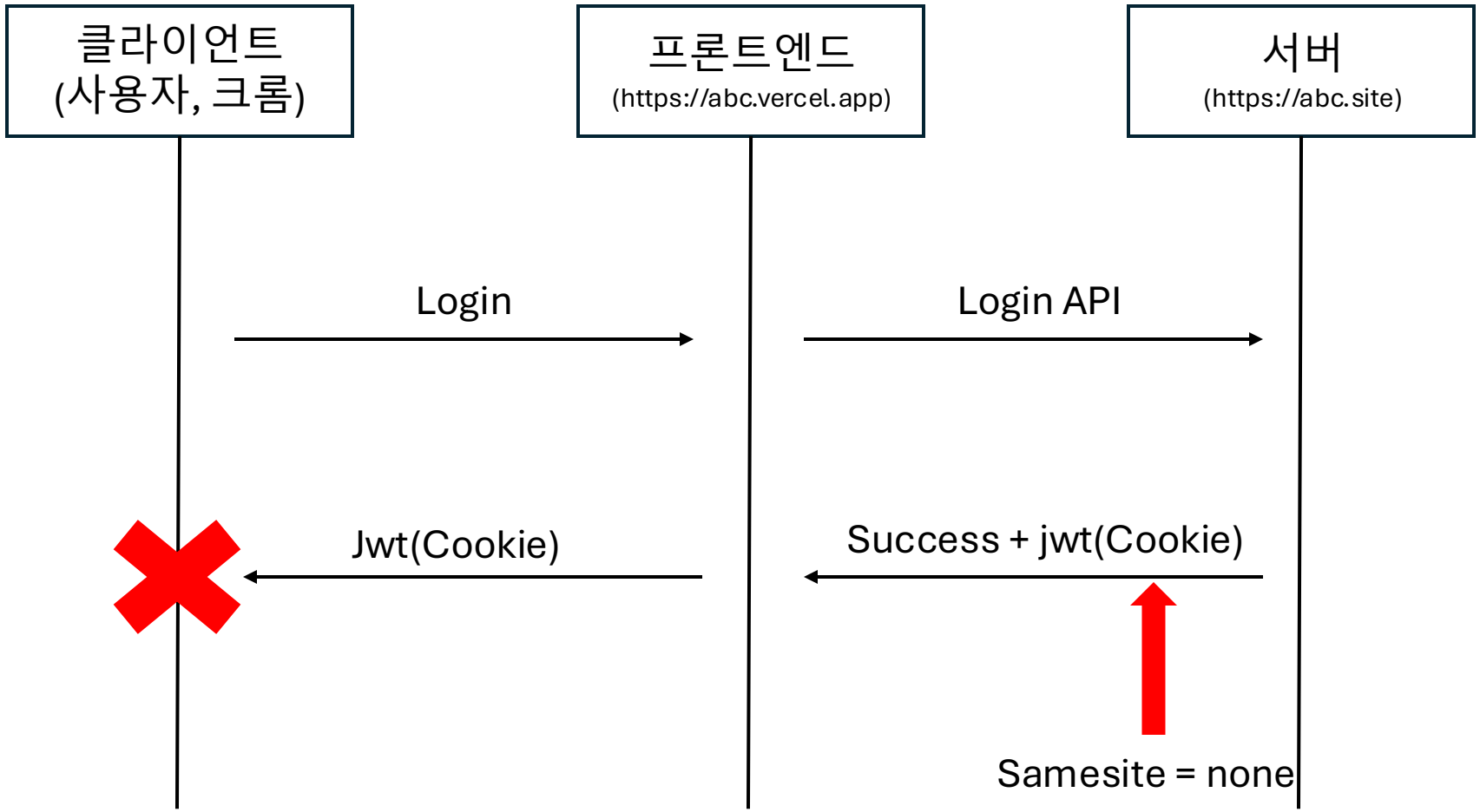
쿠키의 정책(with SameSite)

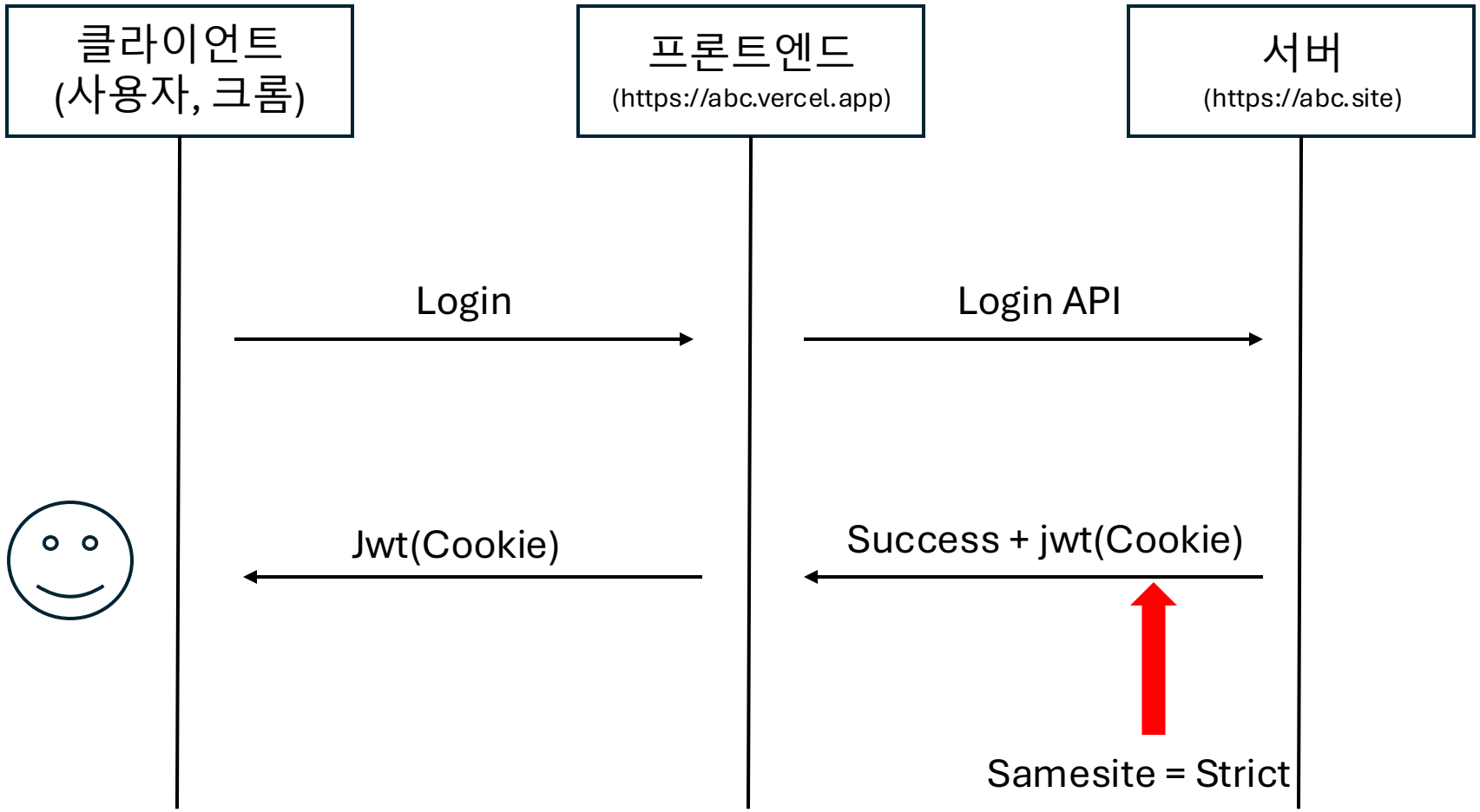
문제상황(1)


클라이언트
(사용자, 크롬)
프론트엔드
(https://abc.vercel.app)
서버
(https://abc.site)
Login
Login API
Jwt(Cookie)
Success + jwt(Cookie)

문제상황(2)


클라이언트
(사용자, 크롬)
프론트엔드
(https://abc.vercel.app)
서버
(https://abc.site)
Login
Login API
Jwt(Cookie)
Success + jwt(Cookie)
Samesite = none

해결방안


클라이언트
(사용자, 크롬)
프론트엔드
(https://abc.vercel.app)
서버
(https://abc.site)
Login
Login API
Jwt(Cookie)
Success + jwt(Cookie)
Samesite = Strict

SameSite

SameSite

1.SOP와 CORS

2.samesite

3.출처

4.동일 출처와 교차 출처

5.공개 접미사와 eTLD+1

6.동일 사이트와 크로스 사이트

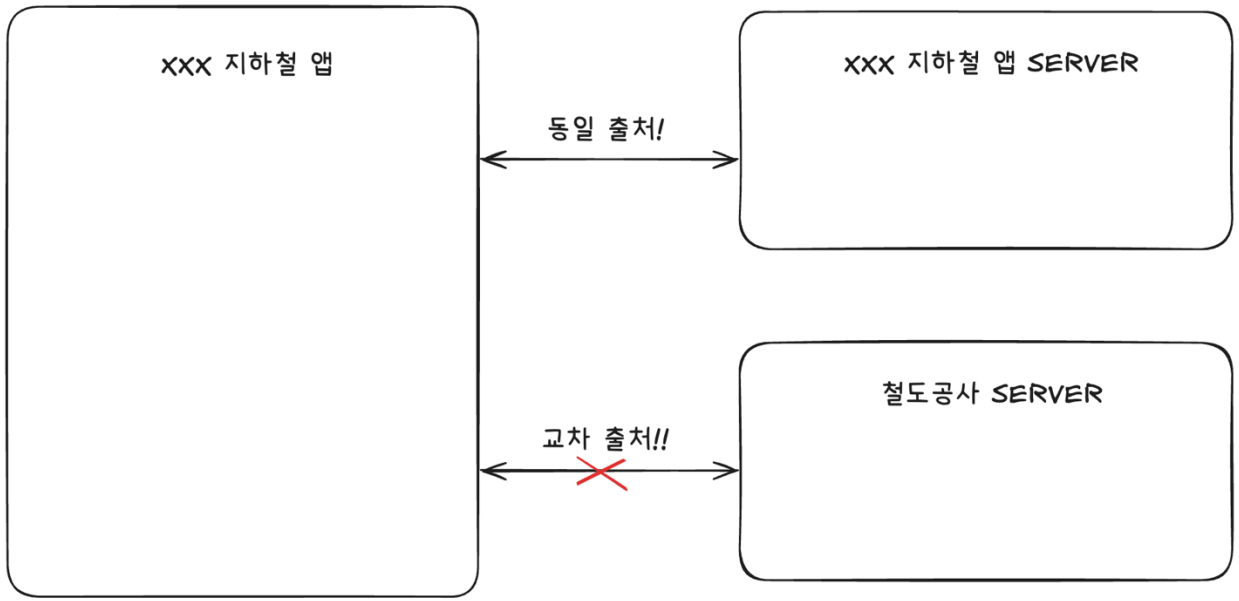
7.스키마 없는 동일 사이트

SOP와 CORS

· SOP(Same Origin policy) : 해석 그대로 **동일(같은) 출처 정책**.
동일 출처에 한해서는 리소스를 얼마든지 주고받을 수 있지만, 다른 출처와는 리소스를 주고 받을 수 없음.
- 정책 탄생 배경 : 동일 출처가 아닌 다른 출처와 리소스 공유를 열어두면 제 3자 가 XSS, CSRF와 같은 여러 수단을 통해 악의적인 행동 가능
이러한 문제를 사전에 방지하고자 동일 출처에 한해서만 리소스 공유가 가능하도록 하는 SOP 정책 탄생

· CORS(Cross-Origin-Resource-Sharing)는 **교차 출처의 리소스를 공유**하는 정책.
SOP를 위반하는 정책이라고 생각할 수 있지만, 만약 SOP의 정책을 필연적으로 지킨다면 보안적으로는 최선의 선택이 될 것이다.

SOP와 CORS



SOP를 최우선적으로 지킨다면 외부 API를 연동하는 서비스는 만들어지지 못한다..

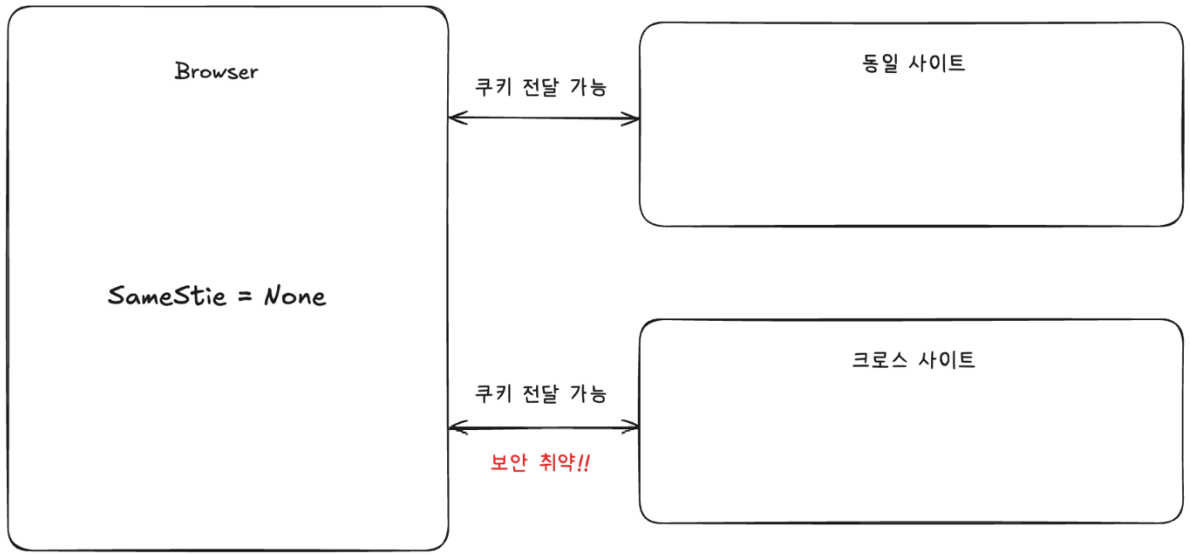
SameSite

· SameSite : 쿠키 정책 중 하나로 정책 설정에 따라 주고 받을 수 있는 쿠키의 범위(first cookie~third cookie)를 결정.

- None, Lax, Strict 3가지 옵션 존재

SameSite – None

· None : 어느 사이트 에서 쿠키를 주고 받을 수 있는 제일 느슨한 옵션.
CSRF 공격에 취약하며 권장되지 않고, 최근 구글 크롬에서는 None 정책의 쿠키를 Lax로 변경되도록 시정 조치시킴.

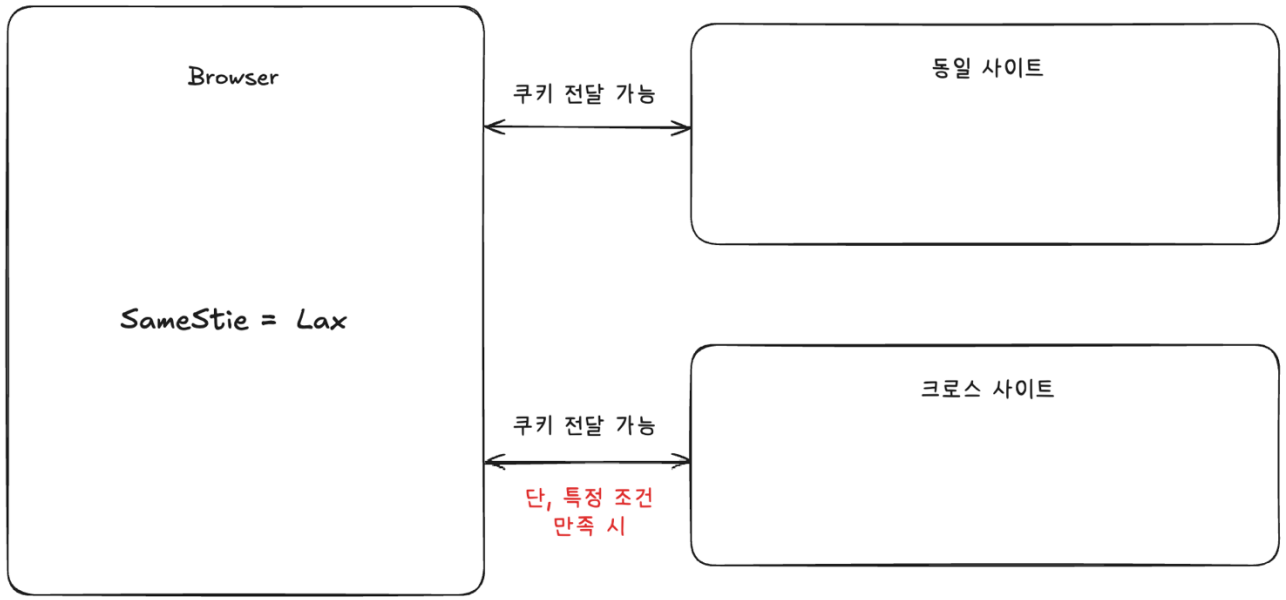


SameSite – Lax

· Lax : None과 비슷하지만 특정 조건이 만족한 상황에서만 쿠키 전달 가능한 옵션.
즉, 제한적인 쿠키 이동 허용 옵션

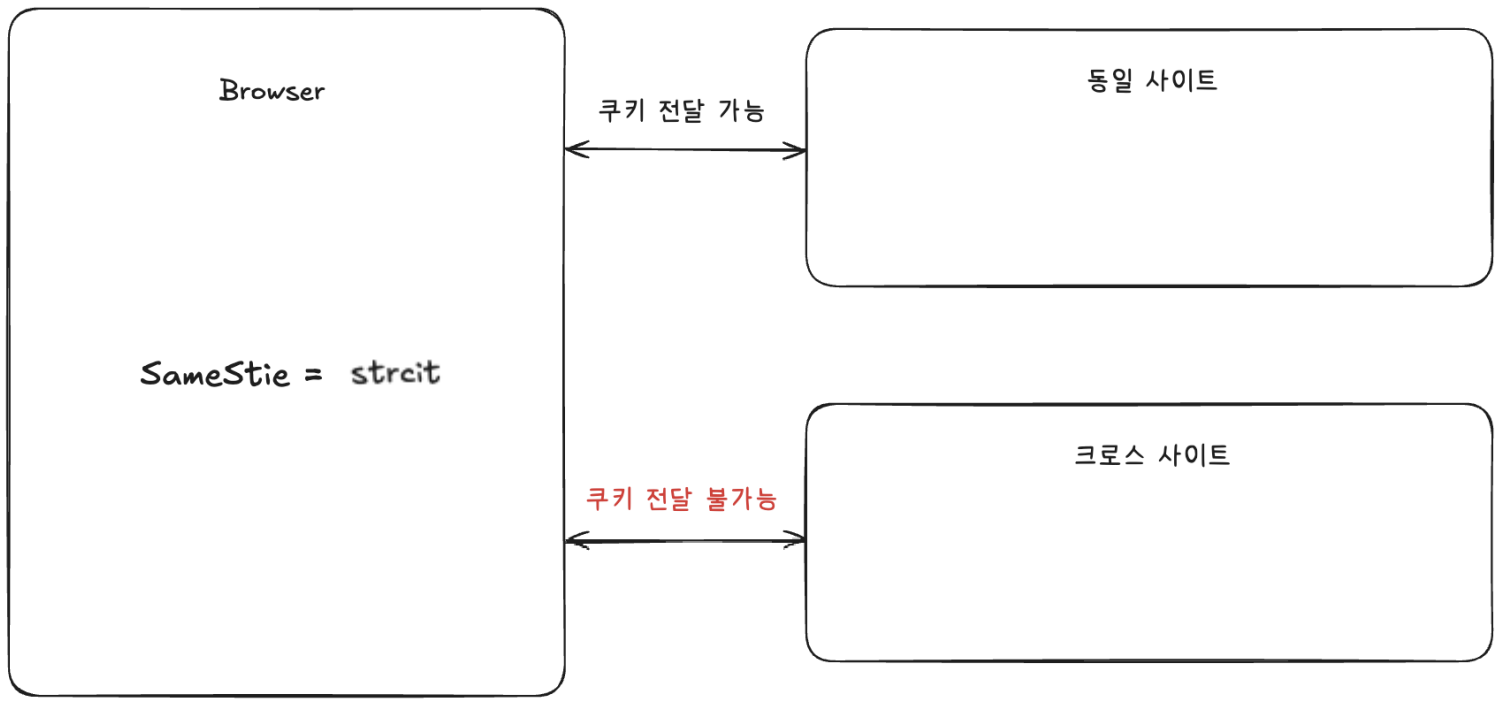
- 특정한 조건
1) 동일 사이트
2) 안전한 HTTP 메서드 요청(GET)
3) 단순한 웹 페이지 이동 * 302(redirect) 또는 <a> 태그

- 전달되지 않는 조건
1) iframe 내에서 이동
2) Img가 문서에 삽입된 HTTP 메서드 요청
3) 안전하지 않은 HTTP 메서드 요청(POST, DELETE)



SameSite – Strict

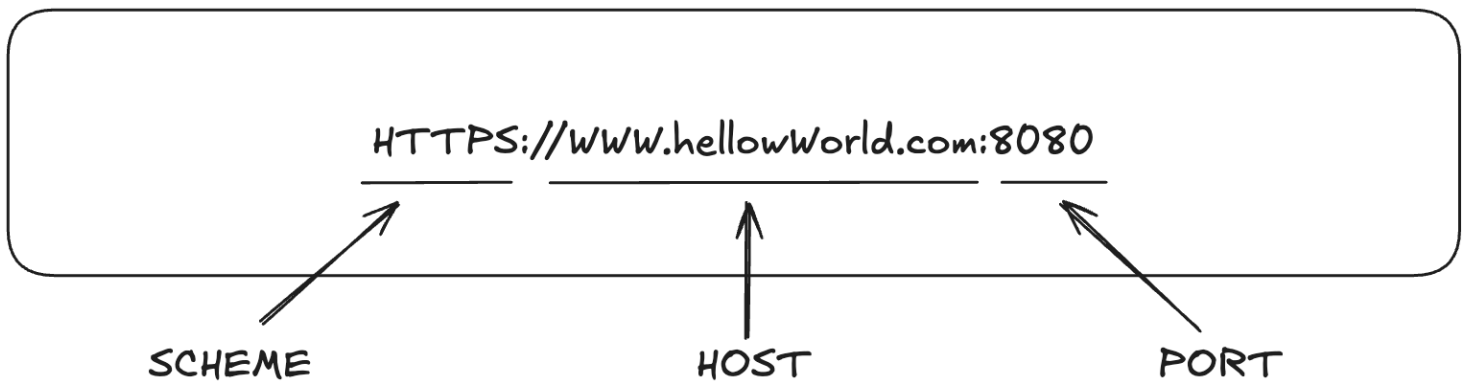
· Strict : 동일 사이트 요청에서만 쿠키가 전달되는 가장 보수적인 옵션.
즉, 크로스 사이트 간의 쿠키 전달은 절대 일어나지 않음.



출처

출처(Origin) : 스키마+호스트+포트의 조합.
- 스키마 : 주로 HTTP 또는 HTTPS로 구성
- 포트 : 지정된 포트번호가 있을 시에 함께 구성 * 명시되어 있지 않으면 기본 포트 80 / 443

즉, http://www.hellowWorld2.com라면 출처는 스키마+호스트 조합으로 구성되어 있다.



동일 출처와 교차 출처

출처 A	출처 B	동일/교차 사이트
https://www.hello.com:443	http://www.hello.com:443	교차 사이트: 다른 스키마
	https://www.bye.com:443	교차 사이트: 다른 호스트
	http://www.bye.com:443	교차 사이트: 다른 스키마,호스트
	https://www.hello.com:80	교차 사이트: 다른 포트
	https://www.hello.com	동일 사이트: 포트(443) 암시적
	https://www.hello.com:443	동일 사이트

공개 접미사와 eTLD+1

· eTLD(effective Top-Level-Domain) : 유효 최상위 도메인

https://www.helloWorld.vercel.app:8080/api/info

eTLD

공개 접미사와 eTLD+1

https://www.helloWorld.vercel.app:8080/api/info

동일사이트??

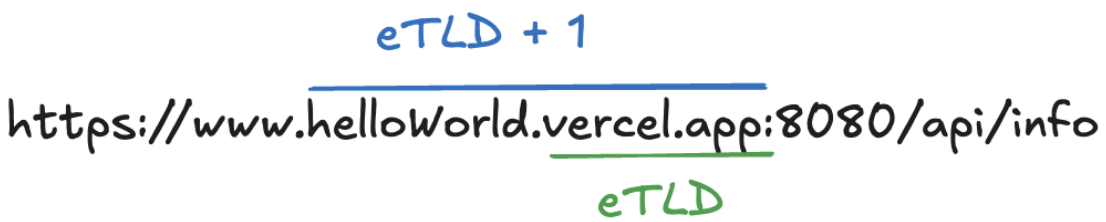
https://www.byeWorld.vercel.app:8080/api/info

과거 eTLD를 통해 동일 사이트인지 확인할 수 있었지만, 수 많은 도메인이 생기며 동일 사이트인지 구분하기 어려워짐..

공개 접미사와 eTLD+1

· eTLD+1 : eTLD보다 한칸 더 한 eTLD.

이를 통해 동일 사이트인지 확인 가능



공개 접미사와 eTLD+1

eTLD+1

https://www.helloWorld.vercel.app:8080/api/info

≠

https://www.byeWorld.vercel.app:8080/api/info

eTLD+1

동일 사이트와 크로스 사이트

A 사이트	B사이트	비교 결과
https://www.hello.vercle.app:8080	http://www.hello.vercle.app:8080	크로스 사이트: 스키마 불일치
	https://www.bye.vercle.app:8080	크로스 사이트: eTLD+1 불일치
	https://www.hello.vercle.app:9090	동일 사이트: Port는 중요하지 않음
	https://www.hello.vercle.app	동일 사이트: Port는 중요하지 않음
	https://www.hello.vercle.app:8080	동일 사이트: 스키마, eTLD+1 전부 일치
	https://dev.hello.vercle.app:8080	동일 사이트: 서브 도메인 불일치 영향X(=동일 사이트로 판단)

스키마가 다르거나, eTLD+1이 다르다면 크로스 사이트!!

스키마가 없는 동일 사이트

· SameSite 정책은 "사이트 신뢰 경계"를 설정하는 정책이므로, 스키마의 차이를 고려할 필요가 없어짐.
=> 단순하게 eTLD+1로만 동일 사이트 여부를 판단하면 됨!

A 사이트	B사이트	비교 결과
https://www.hello.vercle.app:8080	http://www.hello.vercle.app:8080	동일 사이트: 스키마 중요하지 않음
	https://www.bye.vercle.app:8080	크로스 사이트: eTLD+1 불일치
	https://www.hello.vercle.app:9090	동일 사이트: Port는 중요하지 않음
	https://www.hello.vercle.app	동일 사이트: Port는 중요하지 않음
	https://www.hello.vercle.app:8080	동일 사이트: 스키마, eTLD+1 전부 일치
	https://dev.hello.vercle.app:8080	동일 사이트: 서브 도메인 불일치 영향X(=동일 사이트로 판단)