



## #3 / Understanding "same-site" and "same-origin"

"same-site" and "same-origin" are frequently cited but often misunderstood terms.

"same-site" 와 "same-origin" 은 자주 인용되지만 종종 오해를 받는 용어들이다.

For example, they are mentioned in the context of page transitions, `fetch()` requests, cookies, opening popups, embedded resources, and iframes.

예를 들어, 그들은 페이지간의 이동, `fetch()` 요청, 쿠키, 팝업창 띄우기, 임베디드 리소스, 아이프레임의 맵락안에서 잘못된 의미로 사용된다.

## Origin

## 출처

https://www.example.com:443

scheme	host name	port
--------	-----------	------

"Origin" is a combination of a scheme  
(also known as the protocol, for example HTTP or HTTPS),  
hostname, and port (if specified).

"Origin" 이란 scheme (HTTP 나 HTTPS 같은 protocol),  
hostname, 과 port (명시된 경우) 의 조합을 가리킨다.

For example, given a URL of

`https://www.example.com:443/fo`,

the "origin" is

`https://www.example.com:443`.

예를 들어,

`https://www.example.com:443/fo` 라는 URL 이 주어졌다면

여기서 "origin" 은

`https://www.example.com:443` 이다.

---

## "same-origin" and "cross-origin"

"같은 출처" 와 "서로 다른 출처"

Websites that have the combination of  
the same scheme, hostname, and port  
are considered "same-origin".

같은 scheme, hostname, port 를 가진 웹사이트들은  
"same-origin" 으로 간주된다.

Everything else is considered "cross-origin".

그 외의 경우에는 모두 "cross-origin" 으로 간주된다.

Origin A	Origin B	Explanation of whether Origin A and B are "same-origin" or "cross-origin"
	<a href="https://www.evil.com:443">https://www.evil.com:443</a>	cross-origin: different domains
	<a href="https://example.com:443">https://example.com:443</a>	cross-origin: different subdomains
	<a href="https://login.example.com:443">https://login.example.com:443</a>	cross-origin: different subdomains
<a href="https://www.example.com:443">https://www.example.com:443</a>	<a href="http://www.example.com:443">http://www.example.com:443</a>	cross-origin: different schemes
	<a href="https://www.example.com:80">https://www.example.com:80</a>	cross-origin: different ports
	<a href="https://www.example.com:443">https://www.example.com:443</a>	same-origin: exact match
	<a href="https://www.example.com">https://www.example.com</a>	same-origin: implicit port number (443) matches

## Site

사이트

The diagram illustrates the structure of the URL <https://www.example.com:443>. It is divided into three main parts: eTLD (top-level domain), www.example.com (domain name), and eTLD+1 (subdomains and port).

Top-level domains (TLDs) such as `.com` and `.org` are listed in the Root Zone Database.

`.com`이나 `.org`처럼 Top-level domains (TLDs)는 Root Zone Database에 나열되어 있다.

In the example above,  
"site" is the combination of the TLD and the part of the domain just before it.

위의 예제에서 "site" 는,  
TLD 와 TLD 바로 이전에 있는 domain 의 조합이다.

For example, given a URL of  
`https://www.example.com:443/fo` ,  
the "site" is  
`example.com` .

예를 들어,  
`https://www.example.com:443/fo` 라는 URL 이 주어졌다면,  
여기서 "site" 는  
`example.com` 이다.

---

However, for domains such as `.co.jp` or `.github.io` ,  
just using the TLD of `.jp` or `.io` is not granular enough  
to identify the "site".

그러나 `.co.jp` 나 `.github.io` 와 같은 도메인에서  
TLD 로 `.jp` 나 `.io` 로만 사용하기에는  
"site" 를 확인하는데 있어 정보가 충분치 않다.

And there is no way to algorithmically determine  
the level of registrable domains for particular TLD.

그리고 현재 특정 TLD 에서 자동으로 domain 이 기록하기에 적절한지 정도를  
결정하는 알고리즘은 존재하지 않는다.

That's why a list of "effective TLDs"(eTLDs) was created.  
이것이 바로 "effective TLDs"(eTLDs) 목록이 만들어진 이유이다.

These are defined in the Public Suffix List.  
이들은 Public Suffix List 에서 정의되어 있다.

The list of the eTLDs is maintained at [publicsuffix.org/list](http://publicsuffix.org/list).  
eTLDs 목록은 [publicsuffix.org/list](http://publicsuffix.org/list) 에서 관리되고 있다.

---

The whole site name is known as the eTLD+1.  
전체 site 이름은 eTLD+1 로 확인한다.

For example, given a URL of

`https://my-project.github.io`,

the eTLD is

`.github.io`

and the eTLD+1 is

`my-project.github.io`, which is considered a "site".

예를 들어, `https://my-project.github.io` 라는 URL 이 주어졌다면,

여기서 eTLD 는

`.github.io` 이고

eTLD+1 은

`my-project.github.io` 이며, "site" 로 간주된다.

In other words, the eTLD+1 is the effective TLD  
and the part of the domain just before it.

요약하자면, eTLD+1 은 effective TLD 로

TLD 와 TLD 바로 이전 domain 의 조합을 가리킨다.



## "same-site" and "cross-site"

### "같은 사이트" 와 "서로 다른 사이트"

Websites that have the same eTLD+1 are considered "same-site".

같은 eTLD+1 을 가진 웹사이트들은 "same-site" 로 간주된다.

Websites that have a different eTLD+1 are "cross-site".

다른 eTLD+1 을 가진 웹사이트들은 "cross-site" 로 간주된다.

Origin A	Origin B	Explanation of whether Origin A and B are "same-site" or "cross-site"
<a href="https://www.example.com:443">https://www.example.com:443</a>	<a href="https://www.evil.com:443">https://www.evil.com:443</a>	cross-site: different domains
	<a href="https://login.example.com:443">https://login.example.com:443</a>	same-site: different subdomains don't matter
	<a href="http://www.example.com:443">http://www.example.com:443</a>	same-site: different schemes don't matter
	<a href="https://www.example.com:80">https://www.example.com:80</a>	same-site: different ports don't matter
	<a href="https://www.example.com:443">https://www.example.com:443</a>	same-site: exact match
	<a href="https://www.example.com">https://www.example.com</a>	same-site: ports don't matter

### "schemeful same-site"

"scheme 을 고려한 같은 사이트"

https://www.example.com:443

scheme                            eTLD+1

Though "same-site" ignores schemes ("schemeless same-site"), there are cases that must strictly distinguish between schemes in order to prevent HTTP being used as a weak channel.

"same-site" 는 schemes 를 무시하지만 ("scheme 을 고려하지 않은 same-site") weak channel(?) 에 HTTP 가 사용되는 것을 막기 위해서 엄격하게 구분해야 되는 경우가 있다.

In those cases, some documents refer to "same-site" more explicitly as "schemeful same-site".

그러한 경우에서 몇몇 공식문서가 언급하는 "same-site" 는 정확히 말하자면 "schemeful same-site" 를 언급하고 있는 것이다.

In that case,

`http://www.example.com` and `https://www.example.com`

are considered cross-site because the schemes don't match.

이 경우, `http://www.example.com` 와 `https://www.example.com` 는

schemes 가 서로 다르기 때문에 cross-site 로 간주된다.

Origin A	Origin B	Explanation of whether Origin A and B are "schemeful same-site"
	<code>https://www.evil.com:443</code>	cross-site: different domains
	<code>https://login.example.com:443</code>	schemeful same-site: different subdomains don't matter
	<code>http://www.example.com:443</code>	cross-site: different schemes
<code>https://www.example.com:443</code>	<code>https://www.example.com:80</code>	schemeful same-site: different ports don't matter
	<code>https://www.example.com:443</code>	schemeful same-site: exact match
	<code>https://www.example.com</code>	schemeful same-site: ports don't matter

## How to check if a request is "same-site", "same-origin", or "cross-site"

요청이 "same-site", "same-origin" 또는 "cross-site" 인지 어떻게 확인할 수 있는가

Chrome sends requests along with a `Sec-Fetch-Site` HTTP header.

크롬은 HTTP header 에 `Sec-Fetch-Site` 를 포함시켜서 요청을 보낸다

No other browsers support `Sec-Fetch-Site` as of April 2020.

2020년 4월까지 다른 브라우저들은 `Sec-Fetch-Site` 를 지원하지 않는다.

This is part of a larger Fetch Metadata Request Headers proposal.

이는 Fetch Metadata Request Headers 라는 큰 제안의 일부분이다.

The header will have one of the following values:

Header 는 다음의 값들 중 하나를 가진다

- `cross-site`
- `same-site`
- `same-origin`
- `none`

By examining the value of `Sec-Fetch-Site`,  
you can determine if the request is  
"same-site", "same-origin", or "cross-site"  
("schemeful-same-site" is not captured in  
`Sec-Fetch-Site`).

`Sec-Fetch-Site` 값을 확인함으로써,  
요청이 "same-site", "same-origin" 또는 "cross-site" 인지  
알 수 있다.  
("scheme 을 고려한 same-site" 인지는  
`Sec-Fetch-Site` 로 알 수 없다).