

# JavaScript

# INDEX

---

- AJAX

**AJAX**

## AJAX란?

- Asynchronous JavaScript And XML (비동기식 JavaScript와 XML)
- 비동기 통신을 이용하면 화면 전체를 새로고침 하지 않아도 서버로 요청을 보내고, 데이터를 받아 화면의 일부분만 업데이트 가능
- 이러한 '비동기 통신 웹 개발 기술'을 AJAX라 함
- 비동기 웹 통신을 위한 라이브러리 중 하나가 Axios

## AJAX 특징

- 페이지 전체를 reload(새로 고침)를 하지 않고서도 수행되는 “비동기성”
  - 서버의 응답에 따라 전체 페이지가 아닌 일부분만을 업데이트 할 수 있음
1. 페이지 새로고침 없이 서버에 요청
  2. 서버로부터 응답(데이터)을 받아 작업을 수행

# 비동기 적용하기

# 비동기(Async) 적용하기

## | 사전 준비

- 마지막 Django 프로젝트 준비하기 (M:N까지 진행한 프로젝트)
- 가상 환경 생성 및 활성화, 패키지 설치

# 비동기(Async) 적용하기

## | 팔로우 (follow)

- 각각의 템플릿에서 script 코드를 작성하기 위한 block tag 영역 작성

```
<!-- base.html -->

<body>
    ...
    {% block script %}
    {% endblock script %}
</body>
</html>
```



# 비동기(Async) 적용하기

## | 팔로우 (follow)

- axios CDN 작성

```
<!-- accounts/profile.html -->
```

```
{% block script %}
```

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

```
<script>
```

```
</script>
```

```
{% endblock script %}
```

## | 팔로우 (follow)

- form 요소 선택을 위해 id 속성 지정 및 선택
- 불필요해진 action과 method 속성은 삭제 (요청은 axios로 대체되기 때문)

```
<!-- accounts/profile.html -->  
  
<form id="follow-form">  
  ...  
</form>
```

```
<!-- accounts/profile.html -->  
  
<script>  
  const form = document.querySelector('#follow-form')  
</script>
```

# 비동기(Async) 적용하기

## | 팔로우 (follow)

- form 요소에 이벤트 핸들러 작성 및 submit 이벤트 취소

```
<!-- accounts/profile.html -->

<script>
  const form = document.querySelector('#follow-form')
  form.addEventListener('submit', function (event) {
    event.preventDefault()
  })
</script>
```

# 비동기(Async) 적용하기

## | 팔로우 (follow)

- axios 요청 준비

```
<!-- accounts/profile.html -->

<script>
  const form = document.querySelector('#follow-form')
  form.addEventListener('submit', function (event) {
    event.preventDefault()
    axios({
      method: 'post ',
      url: `/accounts/${{???}}/follow/`,
    })
  })
</script>
```

## | 팔로우 (follow)

- 현재 axios로 POST 요청을 보내기 위해 필요한 것
  1. url에 작성할 user pk는 어떻게 작성해야 할까?
  2. csrftoken은 어떻게 보내야 할까?

# 비동기(Async) 적용하기

## | 팔로우 (follow)

- 현재 axios로 POST 요청을 보내기 위해 필요한 것
  1. url에 작성할 user pk는 어떻게 작성해야 할까?
  2. csrftoken은 어떻게 보내야 할까?

# 비동기(Async) 적용하기

## 팔로우 (follow)

- url에 작성할 user pk 가져오기 (HTML -> JavaScript)

```
<!-- accounts/profile.html -->  
  
<form id="follow-form" data-user-id="{{ person.pk }}">  
  ...  
</form>
```

```
<!-- accounts/profile.html -->  
  
<script>  
  const form = document.querySelector('#follow-form')  
  form.addEventListener('submit', function (event) {  
    event.preventDefault()  
  
    const userId = event.target.dataset.userId  
    ...  
  })  
</script>
```

# 비동기(Async) 적용하기

## | data-\* attributes (1/2)

- 사용자 지정 데이터 특성을 만들어 임의의 데이터를 HTML과 DOM사이에서 교환 할 수 있는 방법

- 사용 예시

```
<div data-my-id="my-data"></div>
<script>
  const myId = event.target.dataset.myId
</script>
```

- 모든 사용자 지정 데이터는 dataset 속성을 통해 사용할 수 있음

[https://developer.mozilla.org/ko/docs/Web/HTML/Global\\_attributes/data-\\*](https://developer.mozilla.org/ko/docs/Web/HTML/Global_attributes/data-*)



## | data-\* attributes (2/2)

- 예를 들어 **data-test-value** 라는 이름의 특성을 지정했다면  
JavaScript에서는 **element.dataset.testValue**로 접근할 수 있음
- 속성명 작성 시 주의사항
  - 대소문자 여부에 상관없이 xml로 시작하면 안 됨
  - 세미콜론을 포함해서는 안됨
  - 대문자를 포함해서는 안됨

# 비동기(Async) 적용하기

## | 팔로우 (follow)

- url 작성 마치기

```
<!-- accounts/profile.html -->

<script>
  const form = document.querySelector('#follow-form')
  form.addEventListener('submit', function (event) {
    event.preventDefault()
    const userId = event.target.dataset.userId
    axios({
      method: 'post',
      url: `/accounts/${userId}/follow/`,
    })
  })
</script>
```

## | 팔로우 (follow)

- 현재 axios로 POST 요청을 보내기 위해 필요한 것
  1. url에 작성할 user pk는 어떻게 작성해야 할까?
  2. csrftoken은 어떻게 보내야 할까?

# 비동기(Async) 적용하기

## 팔로우 (follow)

- 먼저 hidden 타입으로 숨겨져있는 csrf 값을 가진 input 태그를 선택해야 함

<https://docs.djangoproject.com/en/3.2/ref/csrf/>

The screenshot displays a web application interface on the left and its corresponding HTML structure in the browser's developer tools on the right.

**Web Page Content (Left):**

- Links: [Login](#) [Signup](#)
- Section: **test1님의 프로필**
- Text: 팔로워 : 1 / 팔로잉 : 0
- Button: 팔로우
- Section: **test1이 작성한 모든 게시글**
- Section: **test1이 작성한 모든 댓글**
- Section: **test1이 좋아요 한 모**

**Developer Tools - Elements Panel (Right):**

```
<!-- base.html -->
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div class="container">
      <a href="/accounts/login/">Login</a>
      <a href="/accounts/signup/">Signup</a>
      <hr>
      <h1>test1님의 프로필</h1>
      <div>...</div>
      <div>
        <form id="follow-form" data-user-id="1">
          <input type="hidden" name="csrfmiddlewaretoken" value="PwqWK20x1M8uRNUZ1omCjeDpYcgRY54YvJrE0ipkoc6mn7tgqRj0reEKFs2FkHk6"> == $0
          <input type="submit" value="팔로우">
        </form>
      </div>
    </div>
  </body>
</html>
```

A red rectangular box highlights the following HTML snippet in the developer tools:

```
<input type="hidden" name="csrfmiddlewaretoken" value="PwqWK20x1M8uRNUZ1omCjeDpYcgRY54YvJrE0ipkoc6mn7tgqRj0reEKFs2FkHk6"> == $0
```

# 비동기(Async) 적용하기

## | 팔로우 (follow)

- 먼저 hidden 타입으로 숨겨져있는 csrf 값을 가진 input 태그를 선택해야 함  
<https://docs.djangoproject.com/en/3.2/ref/csrf/>

```
<!-- accounts/profile.html -->

<script>
  const form = document.querySelector('#follow-form ' )
  const csrftoken = document.querySelector('[name=csrfmiddlewaretoken]').value
</script>
```

# 비동기(Async) 적용하기

## | 팔로우 (follow)

- AJAX로 csrftoken을 보내는 방법

<https://docs.djangoproject.com/en/3.2/ref/csrf/#setting-the-token-on-the-ajax-request>

```
<!-- accounts/profile.html -->

<script>
  const form = document.querySelector('#follow-form ')
  const csrftoken = document.querySelector('[name=csrfmiddlewaretoken]').value
  form.addEventListener('submit', function (event) {
    event.preventDefault()
    const userId = event.target.dataset.userId
    axios({
      method: 'post ',
      url: `/accounts/${userId}/follow/`,
      headers: {'X-CSRFToken': csrftoken,}
    })
  })
</script>
```

## | 팔로우 (follow)

- 팔로우 버튼을 토글하기 위해서는 현재 팔로우가 된 상태인지 여부 확인이 필요
- axios 요청을 통해 받는 response 객체를 활용해 view 함수를 통해서 팔로우 여부를 파악 할 수 있는 변수를 담아 JSON 타입으로 응답하기

# 비동기(Async) 적용하기

## 팔로우 (follow)

- 팔로우 여부를 확인하기 위한 is\_followed 변수 작성 및 JSON 응답

```
# accounts/views.py

from django.http import JsonResponse

@require_POST
def follow(request, user_pk):
    if request.user.is_authenticated:
        User = get_user_model()
        me = request.user
        you = User.objects.get(pk=user_pk)
        if me != you:
            if you.followers.filter(pk=me.pk).exists():
                you.followers.remove(me)
                is_followed = False
            else:
                you.followers.add(me)
                is_followed = True
            context = {
                'is_followed': is_followed,
            }
            return JsonResponse(context)
        return redirect('accounts:profile', you.username)
    return redirect('accounts:login')
```



## 팔로우 (follow)

- view 함수에서 응답한 is\_followed를 사용해 버튼 토글하기

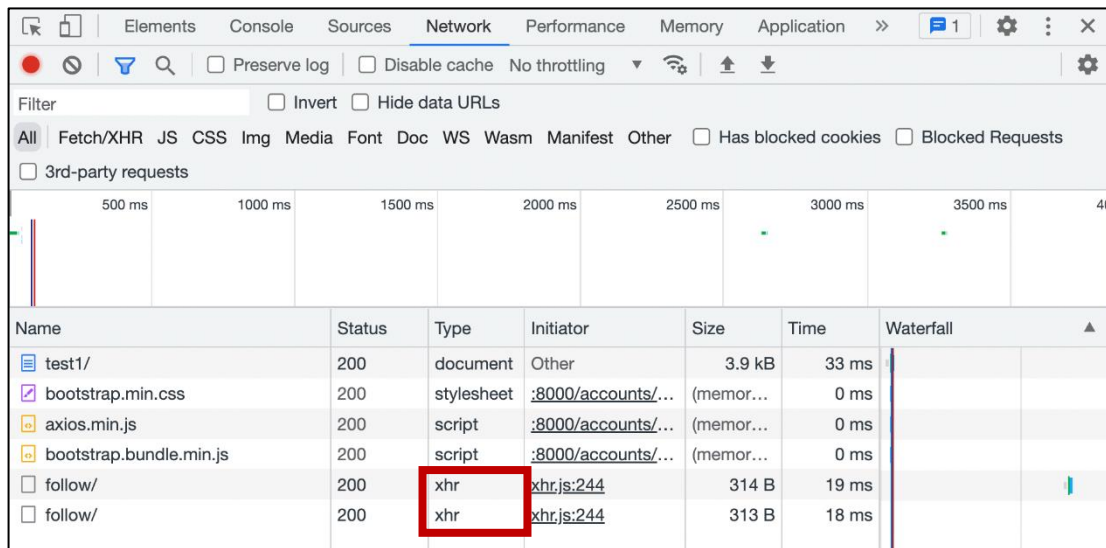
```
<!-- accounts/profile.html -->

<script>
  ...
  axios({
    method: 'post',
    url: `/accounts/${userId}/follow/`,
    headers: {'X-CSRFToken': csrftoken},
  })
  .then((response) => {
    const isFollowed = response.data.is_followed
    const followBtn = document.querySelector('#follow-form > input[type=submit]')
    if (isFollowed === true) {
      followBtn.value = '언팔로우'
    } else {
      followBtn.value = '팔로우'
    }
  })
</script>
```

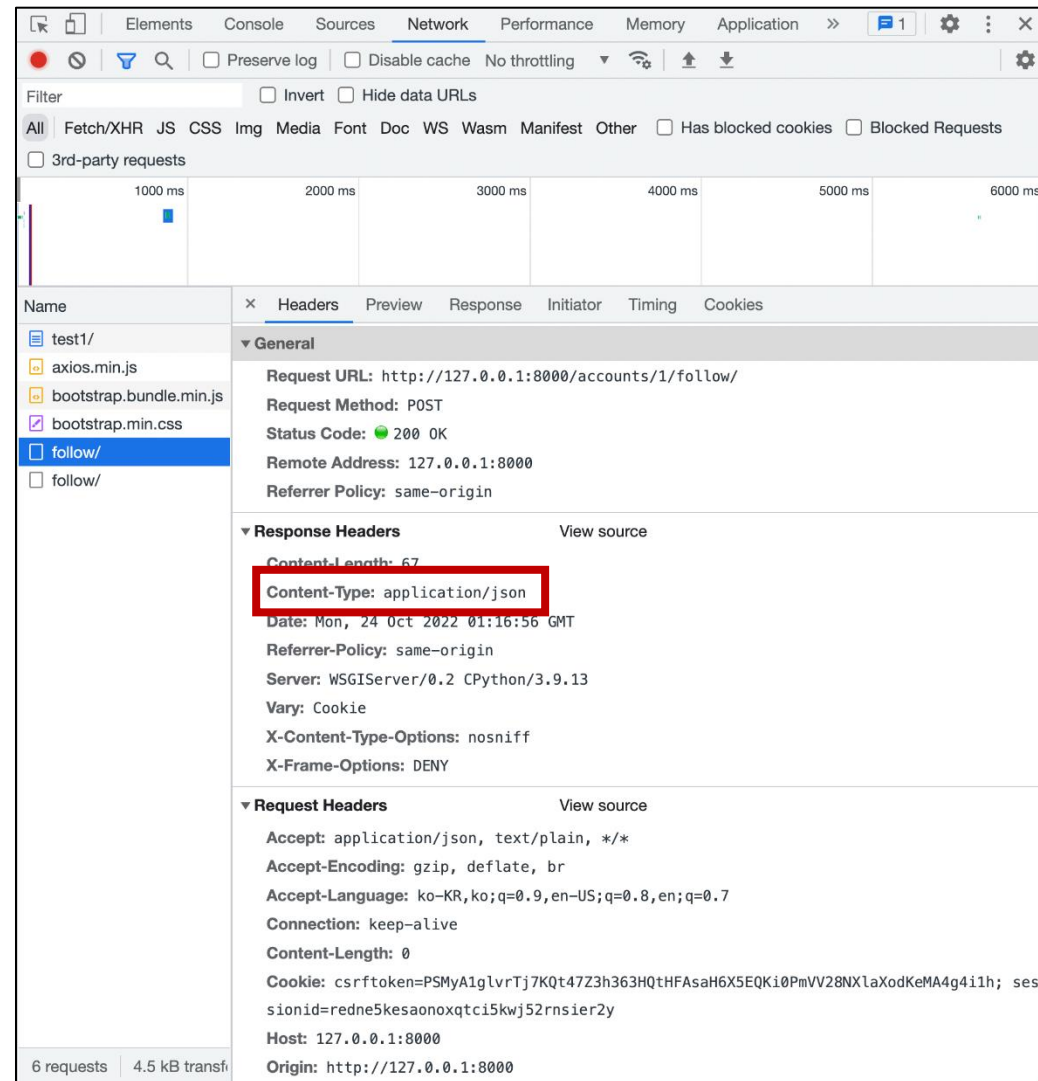
# 비동기(Async) 적용하기

## 팔로우 (follow)

- 결과 확인 (개발자 도구 - Network)



Name	Status	Type	Initiator	Size	Time	Waterfall
test1/	200	document	Other	3.9 kB	33 ms	
bootstrap.min.css	200	stylesheet	:8000/accounts/...	(memor...)	0 ms	
axios.min.js	200	script	:8000/accounts/...	(memor...)	0 ms	
bootstrap.bundle.min.js	200	script	:8000/accounts/...	(memor...)	0 ms	
follow/	200	xhr	xhr.js:244	314 B	19 ms	
follow/	200	xhr	xhr.js:244	313 B	18 ms	



Network tab details for the 'follow/' request:

- General:**
  - Request URL: http://127.0.0.1:8000/accounts/1/follow/
  - Request Method: POST
  - Status Code: 200 OK
  - Remote Address: 127.0.0.1:8000
  - Referrer Policy: same-origin
- Response Headers:**
  - Content-Length: 67
  - Content-Type: application/json**
  - Date: Mon, 24 Oct 2022 01:16:56 GMT
  - Referrer-Policy: same-origin
  - Server: WSGIServer/0.2 CPython/3.9.13
  - Vary: Cookie
  - X-Content-Type-Options: nosniff
  - X-Frame-Options: DENY
- Request Headers:**
  - Accept: application/json, text/plain, \*/\*
  - Accept-Encoding: gzip, deflate, br
  - Accept-Language: ko-KR, ko;q=0.9,en-US;q=0.8,en;q=0.7
  - Connection: keep-alive
  - Content-Length: 0
  - Cookie: csrftoken=PSMyA1glvrTj7KQt47Z3h363HQthFAsaH6X5EQKi0PmVV28NX1aXodKeMA4gi1h; sessionid=redne5kesaonoxqtc15kwj52rnsier2y
  - Host: 127.0.0.1:8000
  - Origin: http://127.0.0.1:8000

## [참고] XHR

- "XMLHttpRequest"
- AJAX 요청을 생성하는 JavaScript API
- XHR의 메서드로 브라우저와 서버 간 네트워크 요청을 전송할 수 있음
- Axios는 손쉽게 XHR을 보내고 응답 결과를 Promise 객체로 반환해주는 라이브러리

# 비동기(Async) 적용하기

## 팔로워 & 팔로잉 수 비동기 적용 (1/4)

- 해당 요소를 선택할 수 있도록 span 태그와 id 속성 작성

```
<!-- accounts/profile.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>{{ person.username }}님의 프로필</h1>
  <div>
    팔로워 : <span id="followers-count">{{ person.followers.all|length }}</span> /
    팔로잉 : <span id="followings-count">{{ person.followings.all|length }}</span>
  </div>
```

## 팔로워 & 팔로잉 수 비동기 적용 (2/4)

- 직전에 작성한 span 태그를 각각 선택

```
<!-- accounts/profile.html -->

<script>
  ...
  axios({
    method: 'post',
    url: `/accounts/${userId}/follow/`,
    headers: {'X-CSRFToken': csrftoken,}
  })
  .then((response) => {
    ...
    const followersCountTag = document.querySelector('#followers-count')
    const followingsCountTag = document.querySelector('#followings-count')
  })
</script>
```

# 비동기(Async) 적용하기

## 팔로워 & 팔로잉 수 비동기 적용 (3/4)

- 팔로워, 팔로잉 인원 수 연산은 view 함수에서 진행하여 결과를 응답으로 전달

```
# accounts/views.py

@require_POST
def follow(request, user_pk):
    ...
    context = {
        'is_followed': is_followed,
        'followers_count': you.followers.count(),
        'followings_count': you.followings.count(),
    }
    return JsonResponse(context)
    return redirect('accounts:profile', you.username)
    return redirect('accounts:login')
```

## 팔로워 & 팔로잉 수 비동기 적용 (4/4)

- view 함수에서 응답한 연산 결과를 사용해 각 태그의 인원 수 값 변경하기

```
<!-- accounts/profile.html -->

<script>
  ...
  axios({
    method: 'post',
    url: `/accounts/${userId}/follow/`,
    headers: {'X-CSRFToken': csrftoken,}
  })
  .then((response) => {
    ...
    const followersCount = response.data.followers_count
    const followingsCount = response.data.followings_count
    followersCountTag.innerText = followersCount
    followingsCountTag.innerText = followingsCount
  })
</script>
```

# 비동기(Async) 적용하기

## 최종 코드 (1/3)

- HTML 코드

```
<!-- accounts/profile.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>{{ person.username }}님의 프로필</h1>
  <div>
    팔로워 : <span id="followers-count">{{ person.followers.all|length }}</span> /
    팔로잉 : <span id="followings-count">{{ person.followings.all|length }}</span>
  </div>

  {% if request.user != person %}
  <div>
    <form id="follow-form" data-user-id="{{ person.pk }}">
      {% csrf_token %}
      {% if request.user in person.followers.all %}
        <input type="submit" value="언팔로우">
      {% else %}
        <input type="submit" value="팔로우">
      {% endif %}
    </form>
  </div>
  {% endif %}
...

```



# 비동기(Async) 적용하기

## 최종 코드 (2/3)

- Python 코드

```
# accounts/views.py

@require_POST
def follow(request, user_pk):
    if request.user.is_authenticated:
        User = get_user_model()
        me = request.user
        you = User.objects.get(pk=user_pk)
        if me != you:
            if you.followers.filter(pk=me.pk).exists():
                you.followers.remove(me)
                is_followed = False
            else:
                you.followers.add(me)
                is_followed = True
        context = {
            'is_followed': is_followed,
            'followers_count': you.followers.count(),
            'followings_count': you.followings.count(),
        }
        return JsonResponse(context)
    return redirect('accounts:profile', you.username)
return redirect('accounts:login')
```

# 비동기(Async) 적용하기

## 최종 코드 (3/3)

- JavaScript 코드

```
<!-- accounts/profile.html -->
const form = document.querySelector('#follow-form')
const csrftoken = document.querySelector('[name=csrfmiddlewaretoken]').value

form.addEventListener('submit', function (event) {
  event.preventDefault()
  const userId = event.target.dataset.userId

  axios({
    method: 'post',
    url: `/accounts/${userId}/follow/`,
    headers: {'X-CSRFToken': csrftoken,}
  })
  .then((response) => {
    const isFollowed = response.data.is_followed
    const followBtn = document.querySelector('#follow-form > input[type=submit] ')
    if (isFollowed === true) {
      followBtn.value = '언팔로우 '
    } else {
      followBtn.value = '팔로우 '
    }
    const followersCountTag = document.querySelector('#followers-count ')
    const followingsCountTag = document.querySelector('#followings-count ')
    const followersCount = response.data.followers_count
    const followingsCount = response.data.followings_count
    followersCountTag.innerText = followersCount
    followingsCountTag.innerText = followingsCount
  })
  .catch((error) => {
    console.log(error.response)
  })
})
```

# 비동기(Async) 적용하기

## | 좋아요 (like)

- 좋아요 비동기 적용은 “팔로우와 동일한 흐름 + `forEach()` & `querySelectorAll()`”
  - index 페이지 각 게시글에 좋아요 버튼이 있기 때문

# 비동기(Async) 적용하기

## 최종 코드 (1/3)

- HTML 코드

```
<!-- articles/index.html -->

{% extends 'base.html' %}

{% block content %}
<h1>Articles</h1>
{% if request.user.is_authenticated %}
  <a href="{% url 'articles:create' %}">CREATE</a>
{% endif %}
<hr>
{% for article in articles %}
<p>
  <b>작성자 : <a href="{% url 'accounts:profile' article.user %}">{{ article.user }}</a></b>
</p>
<p>글 번호 : {{ article.pk }}</p>
<p>제목 : {{ article.title }}</p>
<p>내용 : {{ article.content }}</p>
<div>
  <form class="like-forms" data-article-id="{{ article.pk }}">
    {% csrf_token %}
    {% if request.user in article.like_users.all %}
      <input type="submit" value="좋아요 취소" id="like-{{ article.pk }}">
    {% else %}
      <input type="submit" value="좋아요" id="like-{{ article.pk }}">
    {% endif %}
  </form>
</div>
<a href="{% url 'articles:detail' article.pk %}">상세 페이지</a>
<hr>
{% endfor %}
{% endblock content %}
```

# 비동기(Async) 적용하기

## 최종 코드 (2/3)

- Python 코드

```
# articles/views.py

from django.http import JsonResponse

@require_POST
def likes(request, article_pk):
    if request.user.is_authenticated:
        article = Article.objects.get(pk=article_pk)

        if article.like_users.filter(pk=request.user.pk).exists():
            article.like_users.remove(request.user)
            is_liked = False
        else:
            article.like_users.add(request.user)
            is_liked = True
        context = {
            'is_liked': is_liked,
        }
        return JsonResponse(context)
    return redirect('accounts:login')
```

# 비동기(Async) 적용하기

## 최종 코드 (3/3)

- JavaScript 코드

```
<!-- articles/index.html -->

const forms = document.querySelectorAll('.like-forms')
const csrftoken = document.querySelector('[name=csrfmiddlewaretoken]').value

forms.forEach((form) => {
  form.addEventListener('submit', function (event) {
    event.preventDefault()
    const articleId = event.target.dataset.articleId

    axios({
      method: 'post',
      url: `http://127.0.0.1:8000/articles/${articleId}/likes/`,
      headers: {'X-CSRFToken': csrftoken},
    })
    .then((response) => {
      const isLiked = response.data.is_liked
      const likeBtn = document.querySelector(`#like-${articleId}`)
      if (isLiked === true) {
        likeBtn.value = '좋아요 취소'
      } else {
        likeBtn.value = '좋아요'
      }
    })
    .catch((error) => {
      console.log(error.response)
    })
  })
})
```

# 이어서 ..

삼성 청년 SW 아카데미

# 마무리



## 왜 비동기(Asynchronous) 방식이 필요할까

- “human-centered design with UX”
  - “인간 중심으로 설계된 사용자 경험”
  - 실제 Ajax라는 용어를 처음 논문에서 사용한 Jesse James Garrett이 Ajax를 소개하며 강조한 한 마디



# 마무리

- 동기와 비동기
- JavaScript의 비동기 처리
  - Call Stack, Web API, Task Queue, Event Loop
- Axios 라이브러리
  - then & catch
- Async Callback과 Promise
- AJAX

# 다음 방송에서 만나요!

삼성 청년 SW 아카데미