JS 비동기 세션

13번째 세션

NEXT X LIKELION 김민규

Intro

- HTTP
- Ajax
- 동기 / 비동기
- Fetch or Axios
- JSON

HTTP
HTTP란?

- **HTTP(Hyper Text Transfer Protocol)**는 웹 브라우저와 웹 서버가 웹 페이지(ex. html)나 동영상, 음성 파일 등을 주고받기 위한 프로토콜이다.

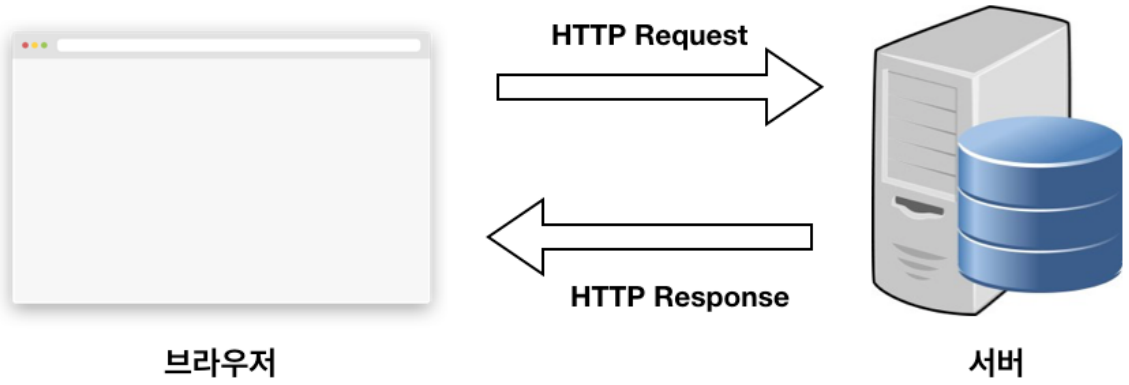
  ***프로토콜**은 컴퓨터 내부에서, 또는 컴퓨터 사이에서 데이터의 교환 방식을 정의하는 규칙 체계이다. 기기 간 통신은 교환되는 데이터의 형식에 대해 상호 합의를 요구한다. 이런 형식을 정의하는 규칙의 집합을 프로토콜이라고 한다.

- Django runserver 후 터미널 창에서도 HTTP를 확인할 수 있음.

```
"GET / HTTP/1.1" 200 573
"GET /static/base.css?v=0.1 HTTP/1.1" 200 385
"GET / HTTP/1.1" 200 573
"GET /static/base.css?v=0.1 HTTP/1.1" 304 0
```

HTTP
HTTP 통신

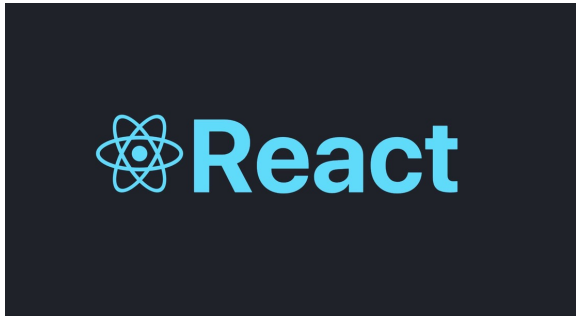
- **HTTP**에서는 클라이언트가 서버에 **요청(request)**을 보내면, 이에 대해 서버가 **응답(response)**을 반환함.



- **만약 클라이언트가 요청이 없다면?**

아니 클라이언트가 뭐고 서버가 뭐죠...?
난 Django 밖에 모르는데요...

vs.



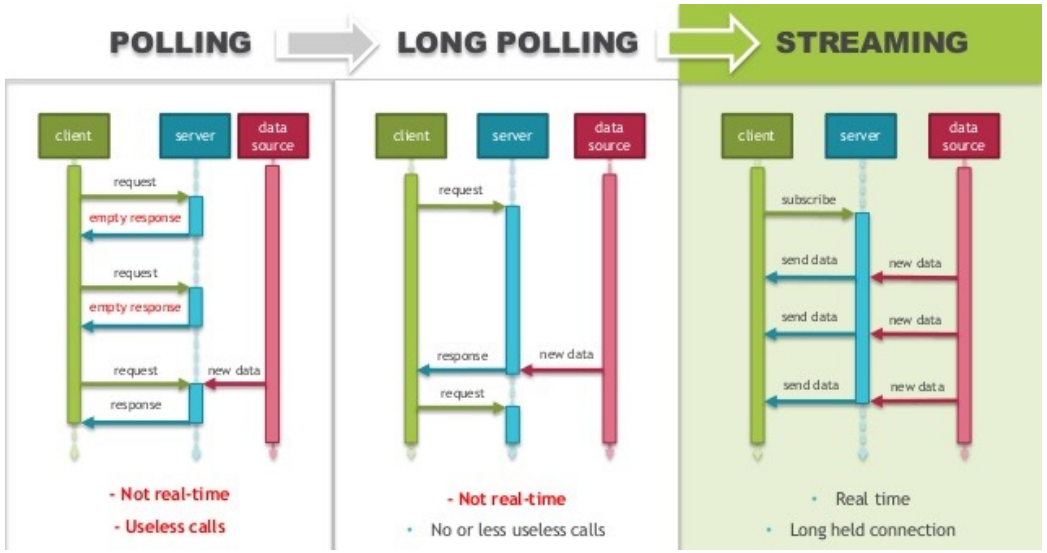
심화 내용(궁금하면 검색)

- **HTTP**에서는 전송 계층 프로토콜로 TCP를 사용하고 네트워크 계층 프로토콜로 IP를 사용하는 것이 일반적. 통칭 TCP/IP라고 부름.

- TCP/IP에서는 IP 주소를 통해 통신할 대상(컴퓨터)을 결정함.
  또한 포트 번호를 통해 해당 컴퓨터의 어떤 프로그램과 통신할지 결정.

심화 내용(궁금하면 검색)

- Polling, Websocket



HTTP

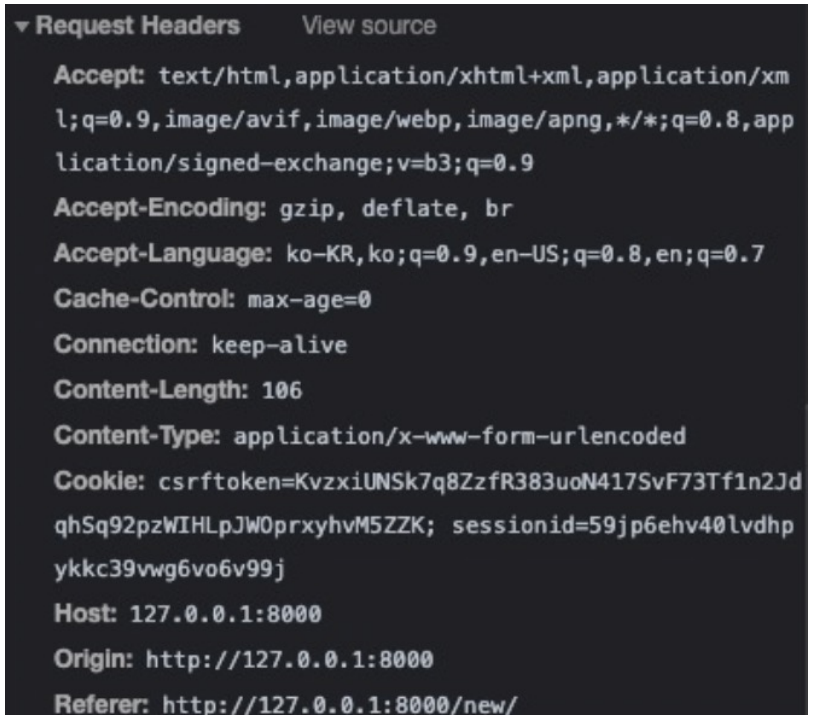
HTTP Request, Response

- HTTP 요청 메세지는 요청 행, 요청 헤더, 메세지 본문으로 구성.
- HTTP 응답 메세지는 응답 행, 요청 헤더, 메세지 본문으로 구성.

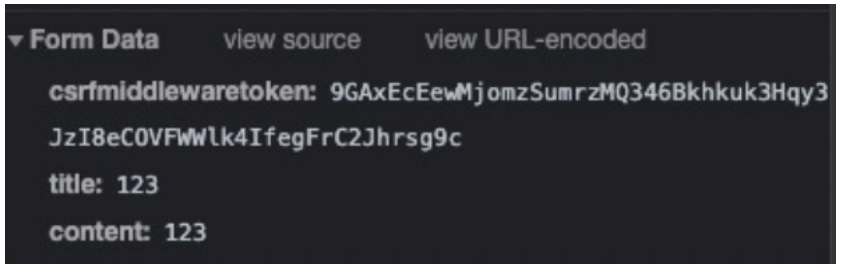
요청 메세지 예시

- Django에서 글 생성 시,

요청 헤더



요청 메세지 본문

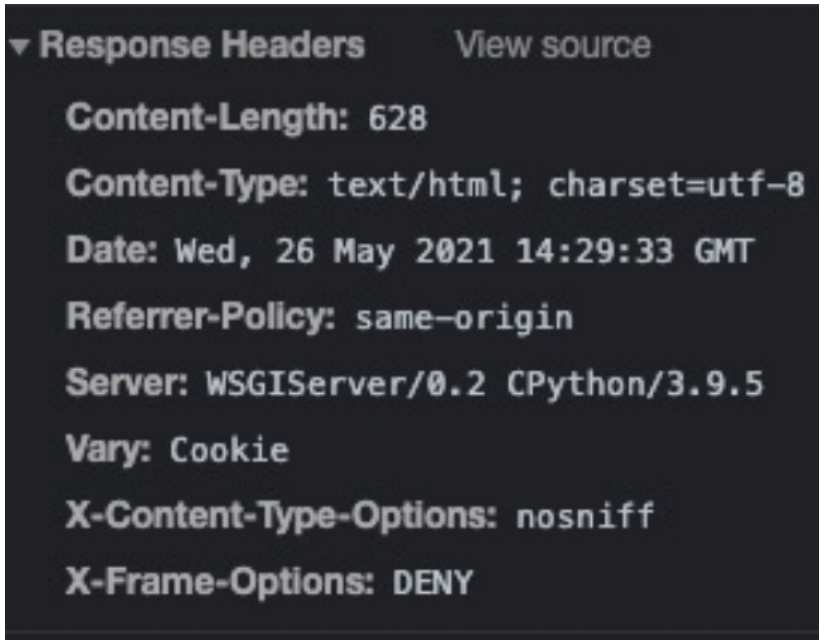


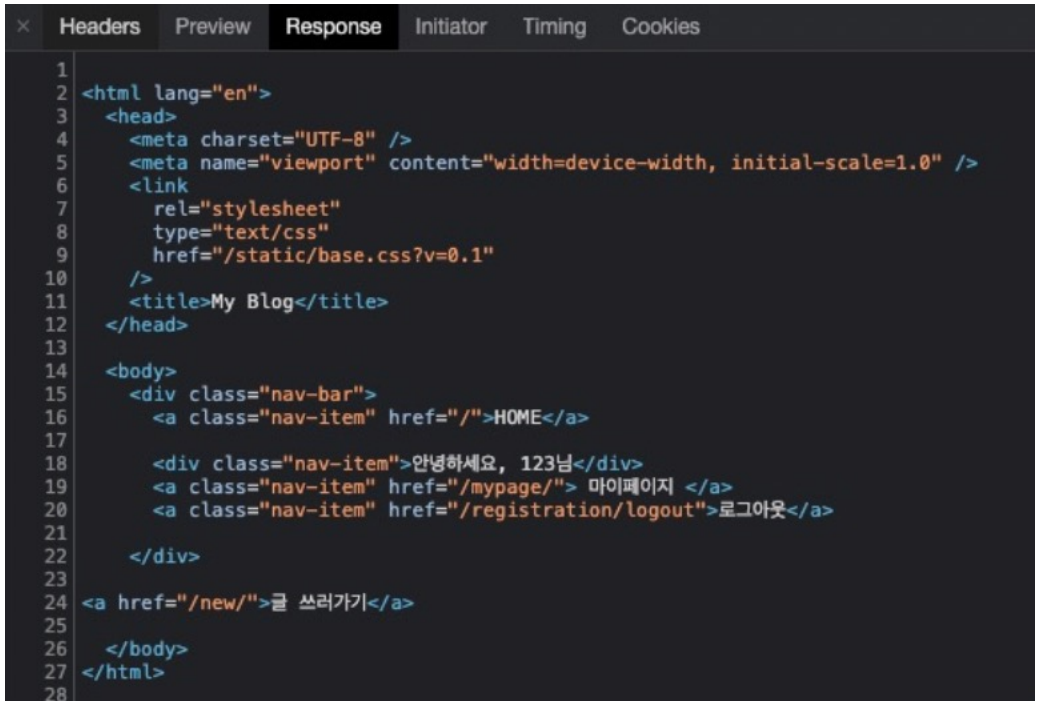
응답 메세지 예시

- Django에서 특정 페이지 진입 시,

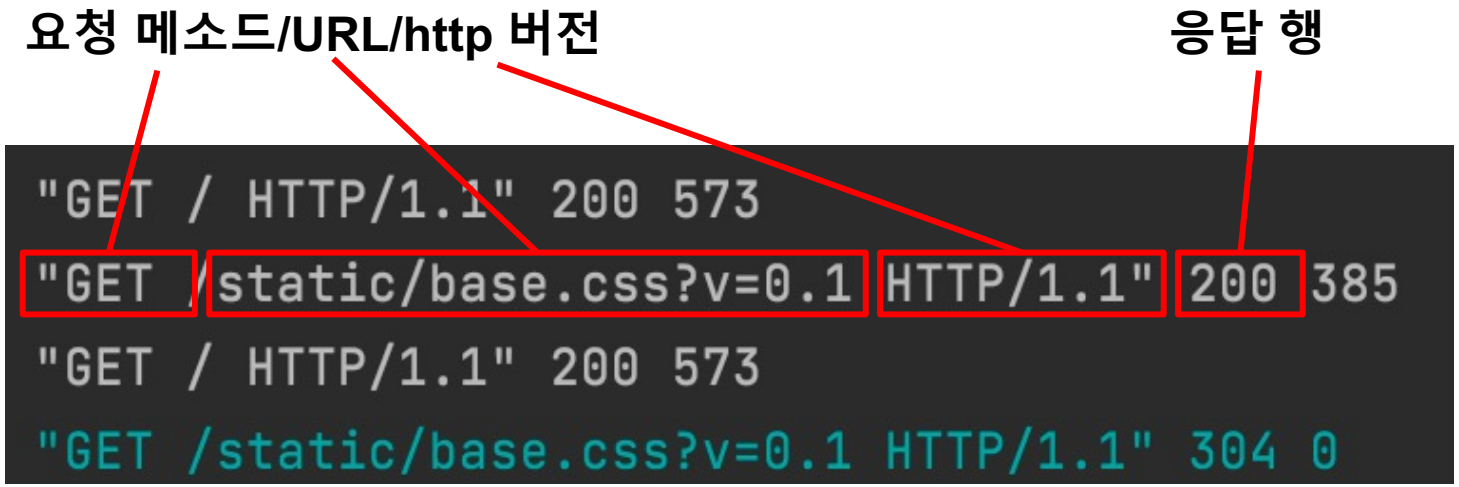
응답 헤더

응답 메세지





요청&응답 메세지 예시



http 상태 코드가 궁금하다면,
https://developer.mozilla.org/ko/docs/Web/HTTP/Status

Ajax
(Asynchronous JavaScript + XML)

- Ajax는 XMLHttpRequest라는 자바스크립트 객체를 활용해 웹 서버와 비동기로 통신하고 DOM을 이용하여 웹 페이지를 동적으로 갱신하는 프로그래밍 기법

- 다시 말해, 웹 페이지가 로딩된 후 일부 데이터를 서버로부터 요청할 수 있도록 함.

- Ajax를 위해서는 XMLHttpRequest 객체를 만들어야 하지만, 편의상 fetch API와 axios 라이브러리를 사용해 데이터를 송수신함.

Ajax
XML, JSON

XML
- Extensible Markup Language의 약자로 W3C에서 정의한 마크업 언어
- Ajax에서 사용하는 데이터 형식을 뜻하는 단어였으나,
  최근에는 XML보다는 JSON과 텍스트 데이터를 사용.

XML vs. JSON
- XML과 JSON은 모두 Markup language로, 태그 등을 이용해 문서나
  데이터 구조를 명기하는 언어.

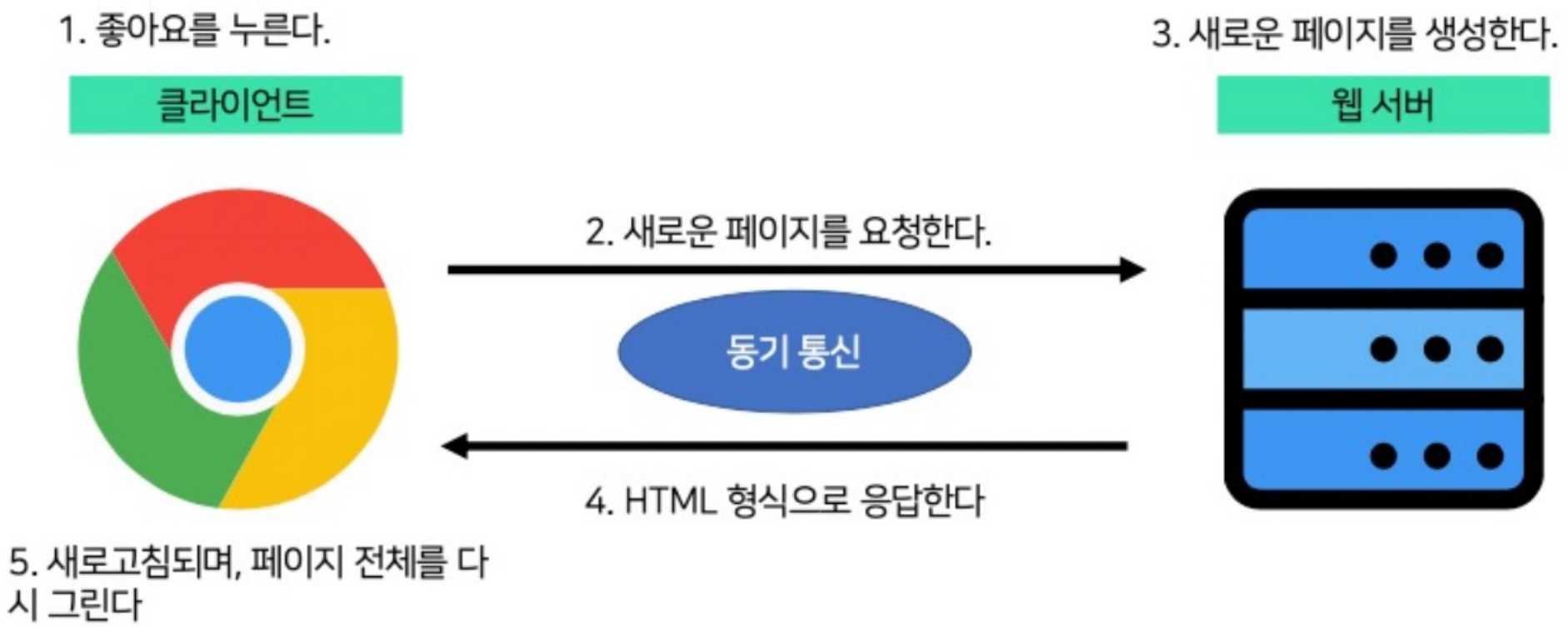
XML

```
<person>
    <age>26</age>
    <name>minkyu</name>
    <job>dev</job>
</person>>
```

JSON

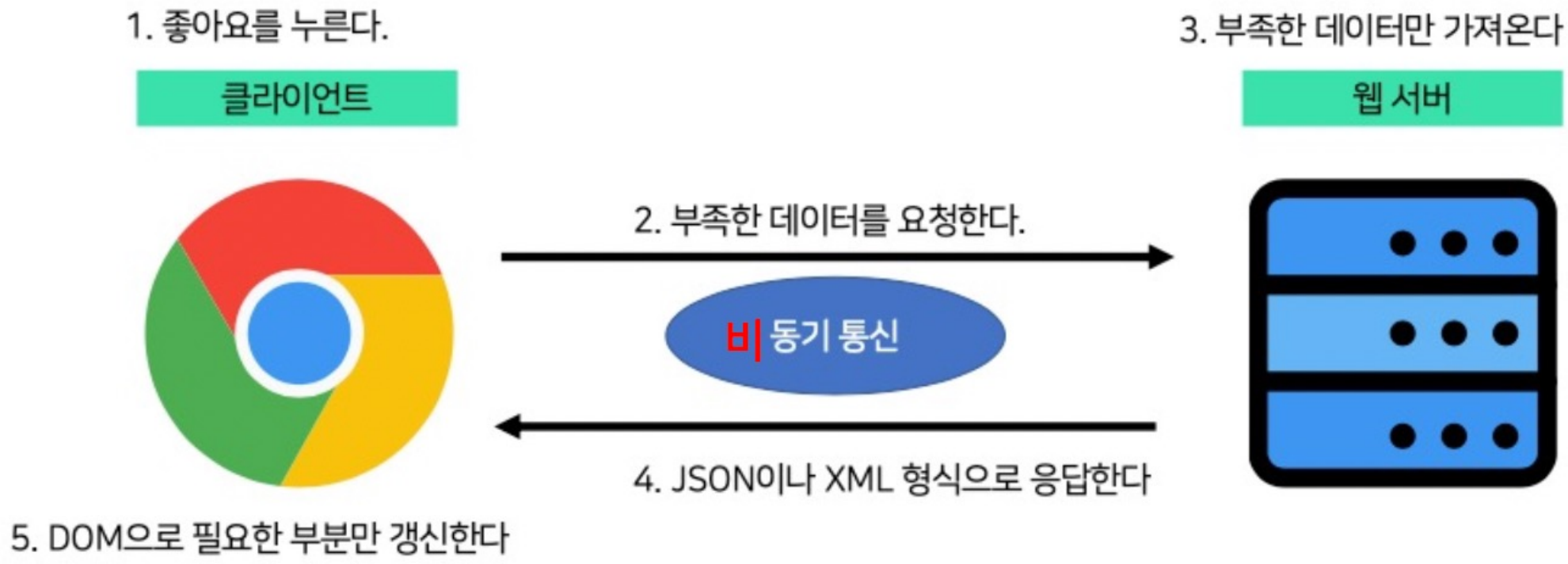
```
{
    'Person' : {
        'age' : 26,
        'name' : 'minkyu',
        'job' : 'dev',
    }
}
```

동기/비동기
동기 통신



여러분이 하고 계신 Django 프로젝트, 고파스 등이 이러한 SSR(Server Side Rendering)로 동작합니다.

동기/비동기
비동기 통신



인스타그램에서 좋아요를 누르면 페이지가 새로 로딩되지 않고 하트가 증가하는 것이 예시입니다.

동기/비동기

비동기 통신의 장점

- **필수 데이터로만 통신**하기 때문에 처리 속도가 빠르고 서버 부하와 통신 트래픽 부하가 적다.
- 비동기로 통신하기 때문에 각 영역에서 다른 작업을 처리할 수 있음.
- 페이지를 전환하는 것이 아닌 페이지 일부만을 변경하기 때문에 빠른 렌더링이 가능함.

실습: 좋아요 구현

- git clone https://github.com/mingrii/Next_Session13.git

- cd djangolike
- pipenv shell
- pipenv install Django
- cd djangolike
- code .

- python manage.py makemigrations
- python manage.py migrate
- python manage.py createsuperuser
- python manage.py runserver

실습: 좋아요 구현

Like Model 추가

```
class Like(models.Model):
    post = models.ForeignKey(
        Post, on_delete=models.CASCADE, related_name="likes"
    )
    user = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name="likes"
    )
```

- python manage.py makemigrations
- python manage.py migrate

실습: 좋아요 구현

Admin.py에 Model 등록

```python
from django.contrib import admin
from .models import Post, Comment, Like
# Register your models here.


admin.site.register(Post)
admin.site.register(Comment)
admin.site.register(Like)
```

실습: 좋아요 구현

동기 통신으로 먼저 구현합니다.

실습: 좋아요 구현

Detail.html에 like 전용 form 추가

```
  {% if user.is_authenticated %}
  <form method="POST">
    {% csrf_token %}
    <input type="text" name="content" placeholder="댓글을 입력하세요" />
    <button type="submit">댓글 쓰기</button>
  </form>

  <form method="POST" action="{% url 'like' %}">
    {% csrf_token %}
    <input type= "hidden" name="post_pk" value="{{ post.pk }}">
    <button type="submit">좋아요</button>
  </form>
  {% endif %}
  <div>좋아요 {{post.likes.count}}개</div>
</div>
{% endblock content %}
```

실습: 좋아요 구현

Url.py

```
urlpatterns = [
    path('admin/', admin.site.urls),

    # auth
    path('registration/signup', views.signup, name="signup"),
    path('registration/login', views.login, name="login"),
    path('registration/logout', views.logout, name="logout"),



    path('', views.home, name="home"),
    path('new/', views.new, name="new"),
    path('mypage/', views.mypage, name="mypage"),
    path('detail/<int:post_pk>', views.detail, name="detail"),
    path('edit/<int:post_pk>', views.edit, name="edit"),
    path('delete/<int:post_pk>', views.delete, name="delete"),
    path('delete_comment/<int:post_pk>/<int:comment_pk>',
         views.delete_comment, name="delete_comment"),

    #like
    path('like', views.like, name="like")
]
```

실습: 좋아요 구현

View.py에 like 올려주는 view 작성

```python
from django.shortcuts import render, redirect
from .models import Post, Comment, Like
from django.contrib.auth.models import User
from django.contrib import auth
from django.contrib.auth.decorators import login_required
```

먼저 import

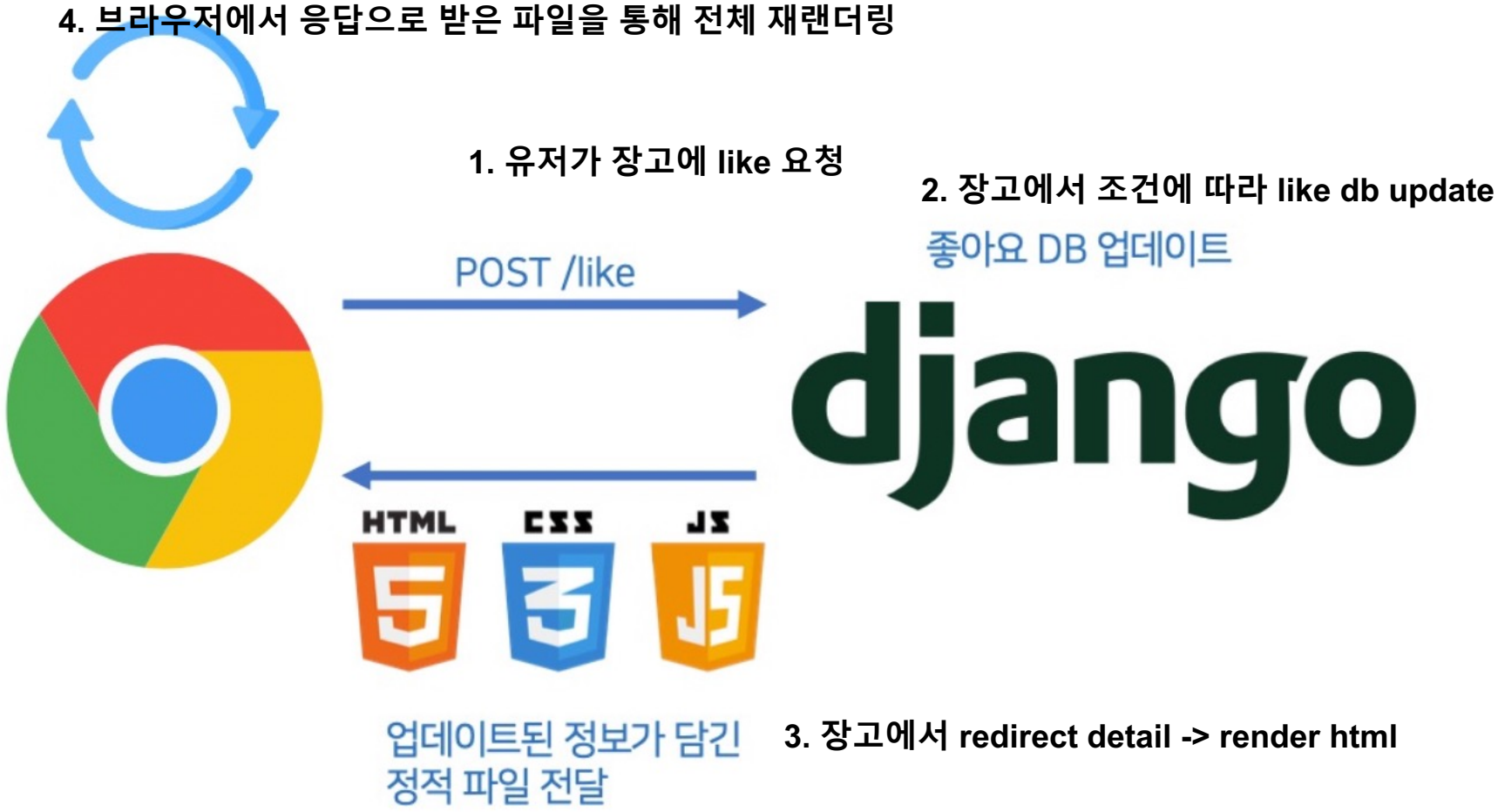
실습 1. 기존 글쓰기 CRUD와 같이 views.py를 작성합니다.

조건

1. 좋아요를 누르면 해당 글의 좋아요를 증가시킵니다.
2. 이미 좋아요를 누른 글이라면 좋아요를 취소합니다.
3. 좋아요를 누르면, 해당 글의 detail 페이지로 redirect 해줍니다.

실습: 좋아요 구현

동기 통신의 동작 방식



동기 통신의 단점

- 서버가 데이터를 보내기 전까지 클라이언트는 아무 것도 할 수 없음
- 웹 페이지 전체를 주고받기 때문에 불필요한 작업으로 인한 소요가 크다.

이러한 단점을 해결하기 위해 비동기 통신이 등장함.

실습: 좋아요 구현

이번엔 비동기 통신으로 구현해봅시다.

알아야 할 사전 개념

1. Django와 브라우저에서 JSON 다루기
2. 저번 세션에서 다루었던 DOM Control

실습: 좋아요 구현

JSON

- JSON.stringify() : 자바스크립트 객체를 JSON 문자열로 변환
- JSON.parse() : JSON 문자열을 자바스크립트 객체로 환원

- json.dumps() : dict 형식을 JSON 문자열로 변환
- json.loads() : JSON 문자열을 dict 형식으로 환원

실습: 좋아요 구현

비동기 통신의 동작 방식

실습: 좋아요 구현

Detail.html에 좋아요 버튼 추가(form 아님)

```
{% if user.is_authenticated %}
<form method="POST">
  {% csrf_token %}
  <input type="text" name="content" placeholder="댓글을 입력하세요" />
  <button type="submit">댓글 쓰기</button>
</form>

<button id="like-button">좋아요</button>
{% endif %}
<div id="like-count">좋아요 {{post.likes.count}}개</div>
```

실습: 좋아요 구현

그래서 button을 click 했을 때 무슨 일이 일어나는데요...?

사실 지금은 아무 일도 안 일어납니다.
onClick 이벤트를 추가해서
무슨 일이 일어나게 해봅시다.

onClick 이벤트 추가 방법

1. 버튼에 직접 attribute 추가하기

```
{% if user.is_authenticated %}
<form method="POST">
  {% csrf_token %}
  <input type="text" name="content" placeholder="댓글을 입력하세요" />
  <button type="submit">댓글 쓰기</button>
</form>

<button id="like-button" onclick="alert('좋아요 버튼이 눌렸어요')">좋아요</button>
{% endif %}
<div id="like-count">좋아요 {{post.likes.count}}개</div>
</div>
```

onClick 이벤트 추가 방법

2. script 이용. DOM으로 onClick event를 추가할 element를 지정하고, onClick시 실행할 함수를 직접 선언한다.

```
  {% if user.is_authenticated %}
  <form method="POST">
    {% csrf_token %}
    <input type="text" name="content" placeholder="댓글을 입력하세요" />
    <button type="submit">댓글 쓰기</button>
  </form>

  <button id="like-button">좋아요</button>
  {% endif %}
  <div id="like-count">좋아요 {{post.likes.count}}개</div>
</div>

<script>
  const likeBtn = document.getElementById('like-button')

  likeBtn.onclick = () => {
    alert('좋아요 버튼이 눌렸어요')
  }
</script>
```

onClick 이벤트 추가 방법

3. Script에서 함수 선언 후, html단에서 가져와 씀.

```
  <button id="like-button" onclick="alertLikeBtnClicked()">좋아요</button>
  {% endif %}
  <div id="like-count">좋아요 {{post.likes.count}}개</div>
</div>

<script>

  const alertLikeBtnClicked = () => {
    alert('좋아요 버튼이 눌렸어요')
  }
</script>
```

Fetch API

- **https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch**

```
const data = { username: 'example' };

fetch('https://example.com/profile', {
  method: 'POST', // or 'PUT'
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify(data),
})
.then(response => response.json())
.then(data => {
  console.log('Success:', data);
})
.catch((error) => {
  console.error('Error:', error);
});
```

Fetch API

- **Fetch API를 적용하여 요청을 보내는 함수를 작성해봅시다.**

```
<script>

  const like = () => {
    fetch('/like', {
      method: "POST",
      body: JSON.stringify({post_pk : "{{post.pk}}"})
    })
    .then(response => response.json())
    .then(res => document.getElementById("like-count").innerHTML = '좋아요' + res.like_count + '개')
    .catch(err => console.error(err))
  }

</script>
```

비동기 좋아요 구현

Django에서 브라우저(html)에서 보낸 요청 받기

먼저 Views.py에 필요한 것 import

```
from django.views.decorators.csrf import csrf_exempt
from django.http import HttpResponse
import json
```

```
@csrf_exempt
def like(request):
    if request.method == "POST":
        request_body = json.loads(request.body)
        post_pk = request_body["post_pk"]
```

비동기 좋아요 구현

Views.py에서 request body 받기

```python
@csrf_exempt
def like(request):
    if request.method == "POST":
        request_body = json.loads(request.body)
        post_pk = request_body["post_pk"]

        existing_like = Like.objects.filter(
            post = POST.objects.get(pk=post_pk),
            user = request.user
        )

        #좋아요 취소
        if existing_like.count() > 0 :
            existing_like.delete()

        #좋아요 생성
        else:
            Like.objects.create(
                post = POST.objects.get(pk=post_pk),
                user = request.user
            )
```

비동기 좋아요 구현

Views.py에서 response 보내기

```
    #좋아요 취소
    if existing_like.count() > 0 :
        existing_like.delete()

    #좋아요 생성
    else:
        Like.objects.create(
            post = POST.objects.get(pk=post_pk),
            user = request.user
        )

post_likes = Like.objects.filter(
    post = Post.objects.get(pk=post_pk)
)

response = {
    'like_count' : post_likes.count()
}

return HttpResponse(json.dumps(response))
```

비동기 좋아요 구현

왜 HttpResponse를 사용해야하나요?

```
response = {
    'like_count' : post_likes.count()
}

return HttpResponse(json.dumps(response))
```

http request에 대한 응답은
http response 형식(응답 행, 응답 헤더, 메세지)에 맞아야 합니다.

```
response = {
    'like_count' : post_likes.count()
}

return (json.dumps(response))
```

아래의 경우는 메세지에 해당하는 데이터만 존재하기 때문에, 형식에 맞는 정보를 자동으로 반환하는 httpResponse라는 모듈을 사용해야 합니다.

좋아요를 비동기 통신으로 구현

비동기 좋아요 구현

Fetch 대신 axios?

- https://github.com/axios/axios

- Axios의 장점
    1. JSON data를 자동으로 변환해줌
    2. Nodejs에서도 사용할 수 있음
    3. 문법이 간결함

비동기 좋아요 구현

Fetch 대신 axios?

- https://github.com/axios/axios

- Axios의 장점
    1. JSON data를 자동으로 변환해줌
    2. Nodejs에서도 사용할 수 있음
    3. 문법이 간결함

비동기 좋아요 구현

Axios 활용하기

스크립트 추가

<script src="https://unpkg.com/axios/dist/axios.min.js"></script>

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
<script>
  const like = () => {
    axios('/like', {
      method: "POST",
      data: {post_pk : "{{post.pk}}"}
    })
    .then(res => document.getElementById("like-count").innerHTML = "좋아요" + res.data.like_count + "개")
  }

</script>
```

비동기 좋아요 구현

Axios 더 간결하게 쓰기

Axios.[method](url, data, headers)

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
<script>
  const like = () => {
    axios.post('/like', {post_pk : "{{post.pk}}"})
    .then(res => document.getElementById("like-count").innerHTML = "좋아요" + res.data.like_count + "개")
  }
</script>
```

비동기 좋아요 구현

Promise.then 대신 async/await

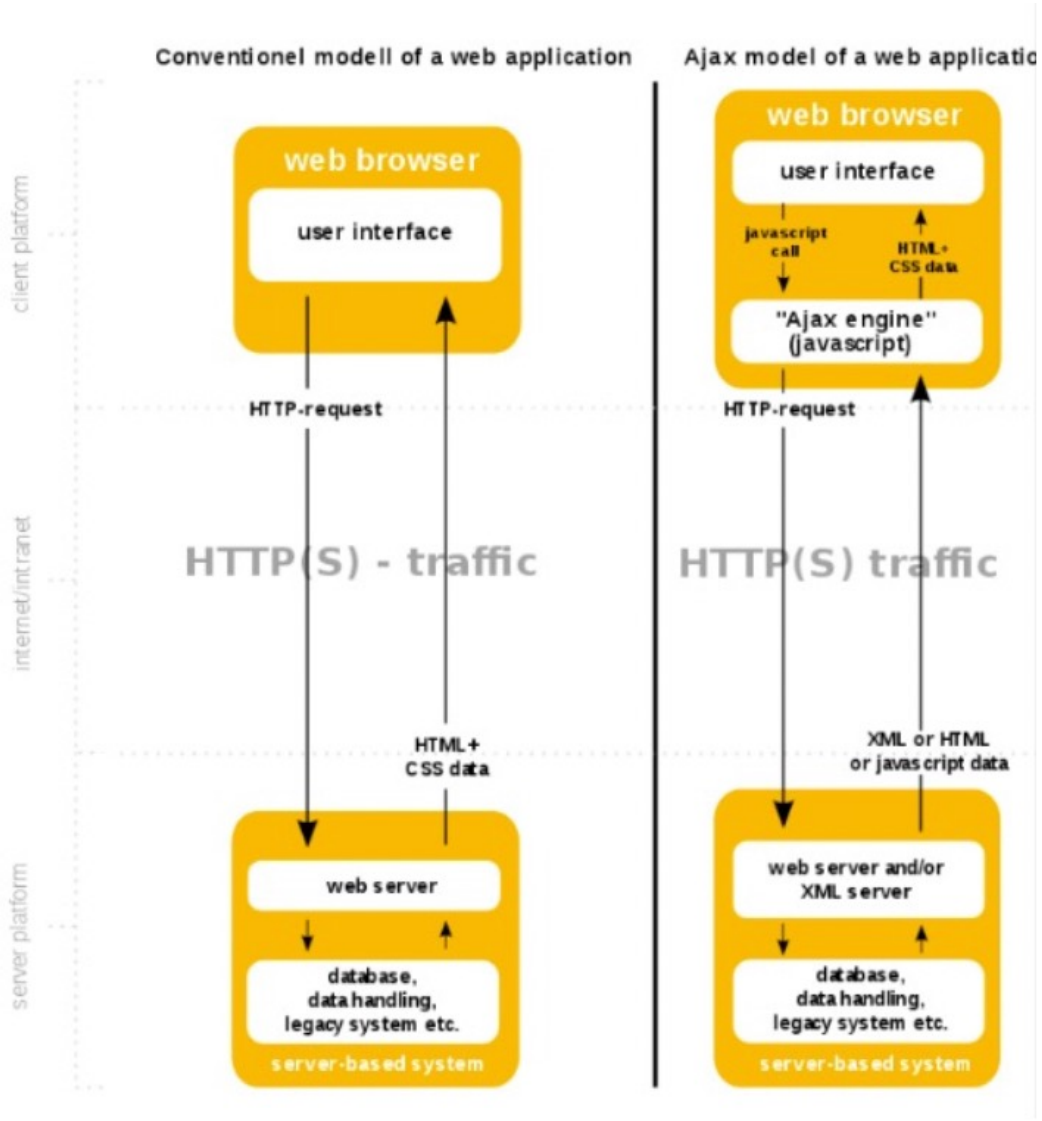
```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
<script>
  const like = async () => {
    try {
      const response = await axios.post('/like', {post_pk : "{{post.pk}}"})
      document.getElementById("like-count").innerHTML = "좋아요" + response.data.like_count + "개"
    }
    catch (e) {
      console.error(e)
    }

  }
</script>
```

총정리

동기 통신
1. Html 정적 파일 수신
2. 새로운 요청시 매번 새로 고침

비동기 통신
1. JSON 송수신
2. 필요한 부분만 DOM 수정



비동기에 대해

사실 비동기는 어렵고 심오합니다. 지금한 건 걸음마...

콜백지옥,

Promise,

이벤트 루프를 추가적으로 공부해보시길 바랍니다.

과제

1. 게시글에 접속했을 때, 유저가 좋아요를 누른 게시글이면 좋아요가 빨간색으로 표시되도록 해주세요. 최초 접속 시에도 가능해야 합니다.

2. 스크랩 기능을 만들어주세요. 스크랩한 게시물은 마이페이지에서 확인 가능하며, 총 스크랩된 횟수는 해당 게시글 작성자만 확인할 수 있도록 해주세요.

3. 마이페이지를 만들고, 좋아요 누른 게시글과 스크랩한 게시물의 제목만 모아볼 수 있게 만들어주세요.