

ChatGPT streaming 구현하기

신현지 (Align AI)

<https://www.linkedin.com/in/hyeonji-shin>

ChatGPT streaming



You

세종대왕 맥북 던짐 사건에 대해 알려줘.

< 2 / 2 > 



ChatGPT




"세종대왕 맥북 던짐 사건"은 실제 역사적 사실이 아닌, 인공지능 챗봇이 생성한 가상의 일화입니다. 이 사건은 인터넷에서 밈(meme)으로 유행하며, AI가 잘못된 정보를 제공할 수 있음을 보여주는 예로 인식되고 있습니다. AI의 발전과 함께 그 한계와 오류에 대한 인식도 커지고 있는데, 이런 오류는 AI 기 ●

ChatGPT streaming

민병철유폰, 대화형AI 학습서비스 ‘도와줘 유폰 GPT’ 출시

연지연 기자

입력 2023.07.03 19:38

URL  가  

1:1 영어회화 전문 브랜드 민병철유폰이 ‘유폰 GPT’를 출시했다고 3일 밝혔다. 이번에 선보이는 유폰 GPT는 민병철유폰만의 대화형 AI 학습 서비스다. 지난 11월 오픈AI가 공개한 챗지피티(ChatGPT) 기술이 탑재됐다.

앱 안에 인공지능 기능… 챗GPT 활용 서비스 쏟아진다

각종 프로그램에 인공지능 삽입 붐, 앱스토어처럼 생태계 형성

임경업 기자 김성민 기자

업데이트 2023.03.10. 11:44 ▼

국내 AI(인공지능) 스타트업 업스테이지가 만든 카카오톡 기반 챗봇 ‘아숙업(AskUp)’은 지난 5일 출시 이후 나흘 만에 이용자 수 4만명을 돌파했다. 최근 화제가 되고 있는 챗봇AI 챗GPT는 오픈AI의 영문 웹사이트에서만 사용 가능하지만, 아숙업은 이를 카카오톡으로 불러와서 쓸 수 있게 만들었다. 추가로 OCR(광학문자인식) 기술을 활용해 카카오톡에 문서 사진을 찍어 올리면 문서를 요약해 주는 기능도 추가했다. 업스테이지 창업자 김성훈 대표가 개발해 무료로 출시했다. 김 대표는 “오픈AI가 챗GPT의 API(응용프로그램 인터페이스)를 공개한 덕분”이라며 “이달 안에 챗GPT 기반 서비스가 수백 개 나올 것”이라고 말했다.

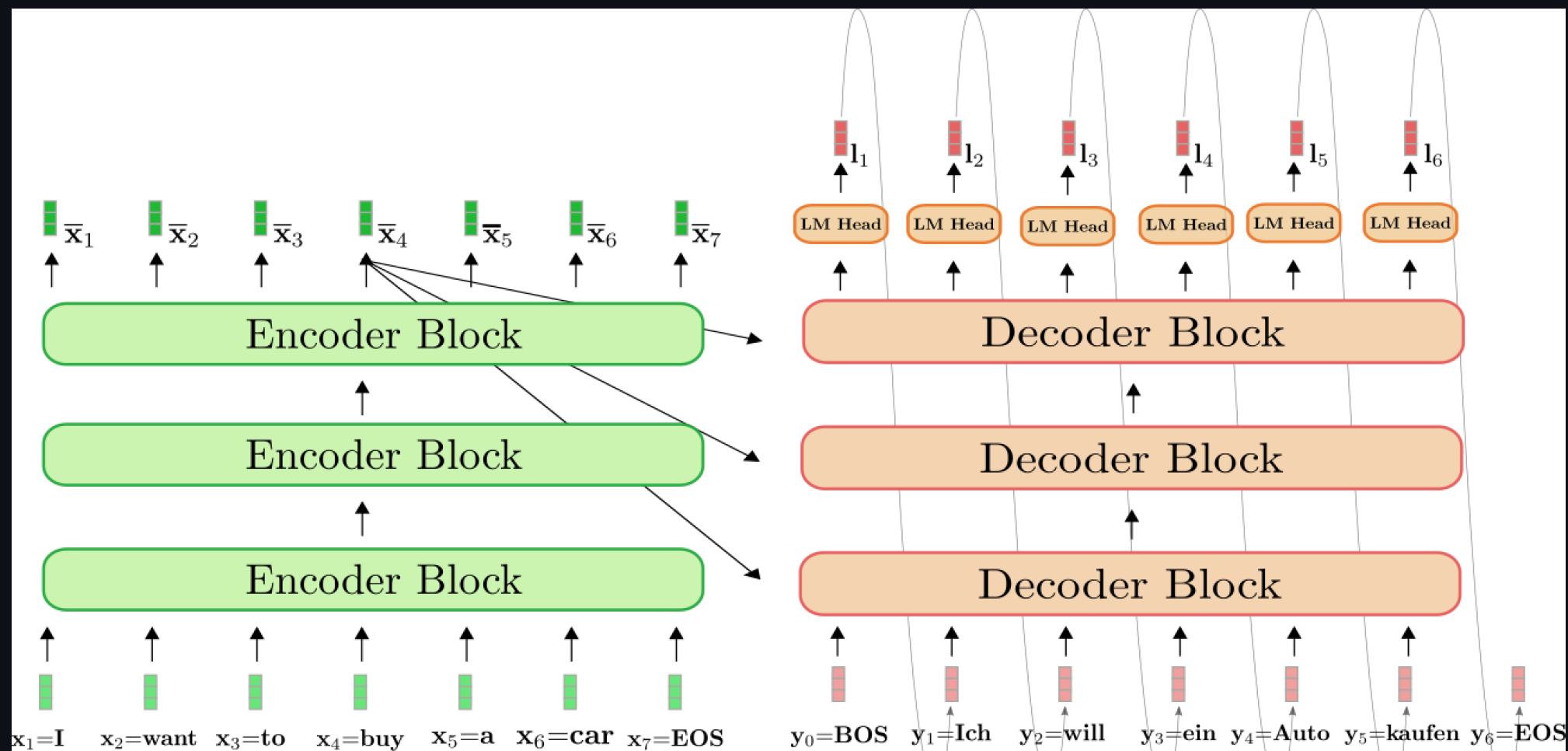
GPT는 어떻게 응답을 생성할까?

GPT-4 is a **Transformer-style model** pre-trained to predict the next token in a document, ...

- OpenAI, GPT-4 Technical Report (2023)

GPT는 어떻게 응답을 생성할까?

- Transformer



어떻게 응답을 즉시 전달할까?

HTTP 세션의 과정:

1. 클라이언트가 TCP 연결을 수립
 2. 클라이언트가 서버에 요청을 전송한 뒤 응답을 대기
 3. 서버는 요청을 처리한 뒤 응답을 전송
- HTTP/1.1부터는 3번 과정 이후 **TCP 연결을 유지**
 - 하지만 서버가 먼저 클라이언트로 데이터를 전송할 수 없음

서버가 클라이언트로 데이터를 전송하는 방법

- Polling
- Long Polling
- WebSocket
- SSE(Server-Sent Events)

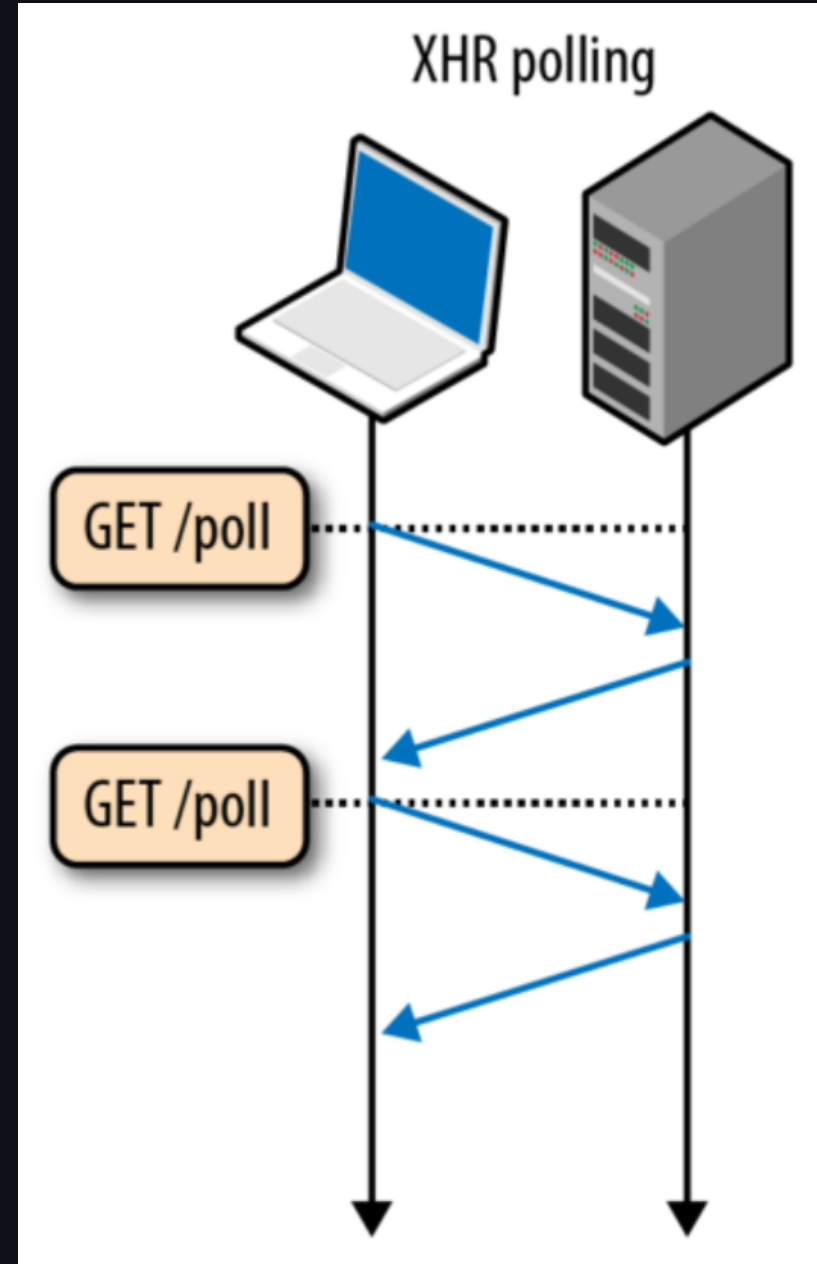
Polling

클라이언트가 서버에게 주기적으로 HTTP 요청을 보내 응답을 받음

장점 : 구현이 간단

단점 : 불필요한 트래픽 발생, 서버 부하

☞ 업데이트가 빈번하지 않거나 실시간 응답이 중요하지 않은 경우 적합



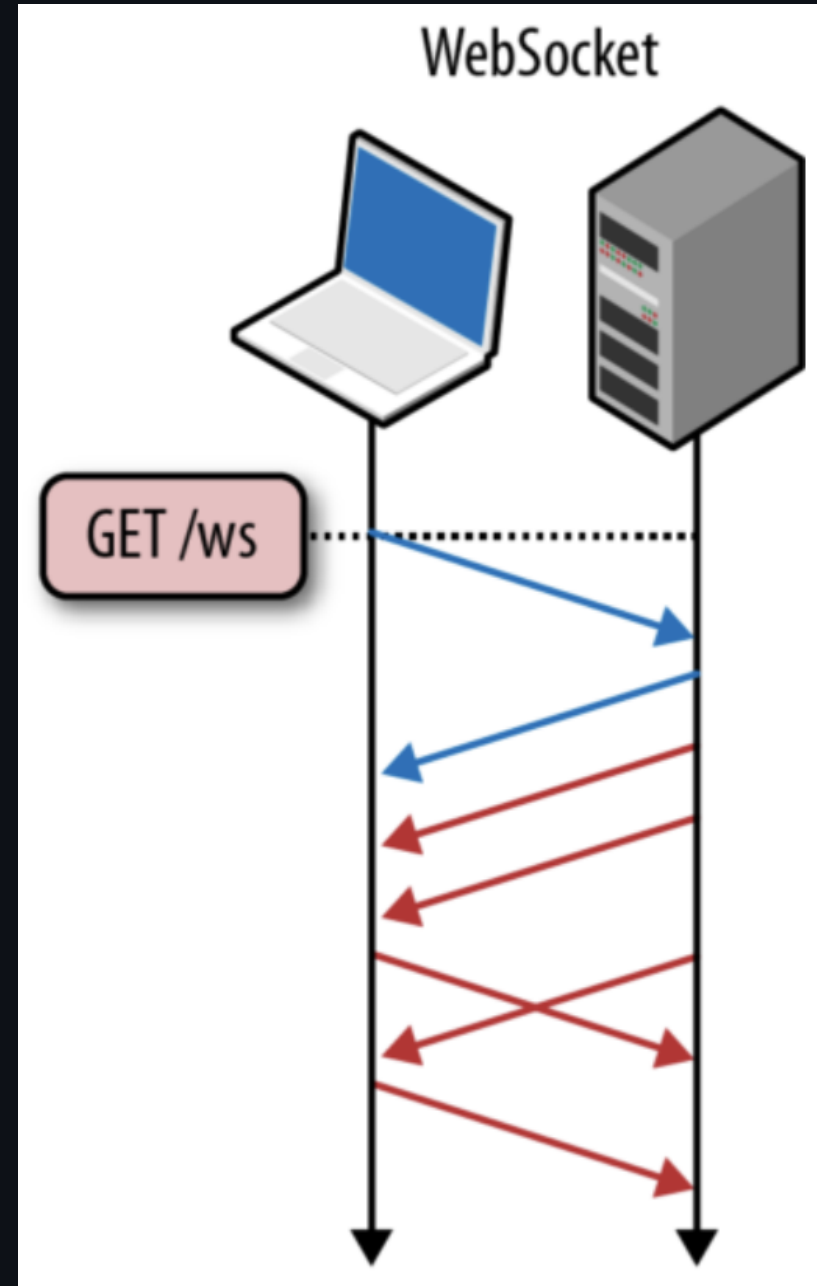
WebSocket

클라이언트와 서버 간 양방향 통신을 지원하는 프로토콜

장점 : 양방향 통신

단점 : HTTP가 아닌 TCP 레이어에서 동작
하므로 구현의 복잡도 증가

☞ 채팅, 협업 도구와 같이 지속적인 데이터 전송이 필요한 경우 적합



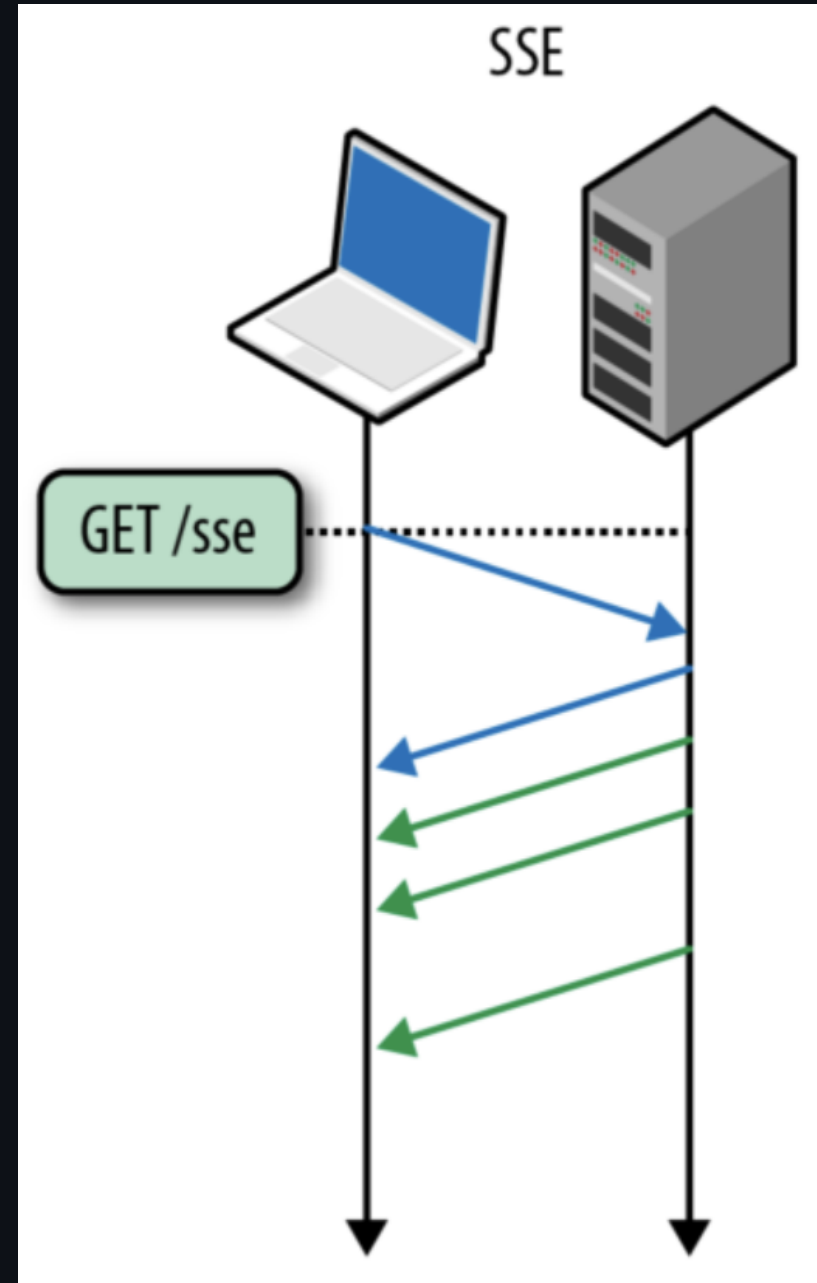
SSE

서버가 클라이언트로 데이터를 전송하는 단
방향 통신 방식

장점 : WebSocket의 복잡성 없이 실시간
업데이트 가능

단점 : 단방향 통신

☞ 뉴스 피드, 실시간 모니터링 대시보드와
같은 단방향 시나리오에 적합



ChatGPT streaming

- Polling: 매우 잦은 요청이 발생하므로 적합하지 않음
- WebSocket: 양방향 통신이 필요하지 않으므로 적합하지 않음
- SSE: 단방향 통신이므로 적합

ChatGPT streaming

 conversation



헤더

페이로드

EventStream

시작점

타이밍

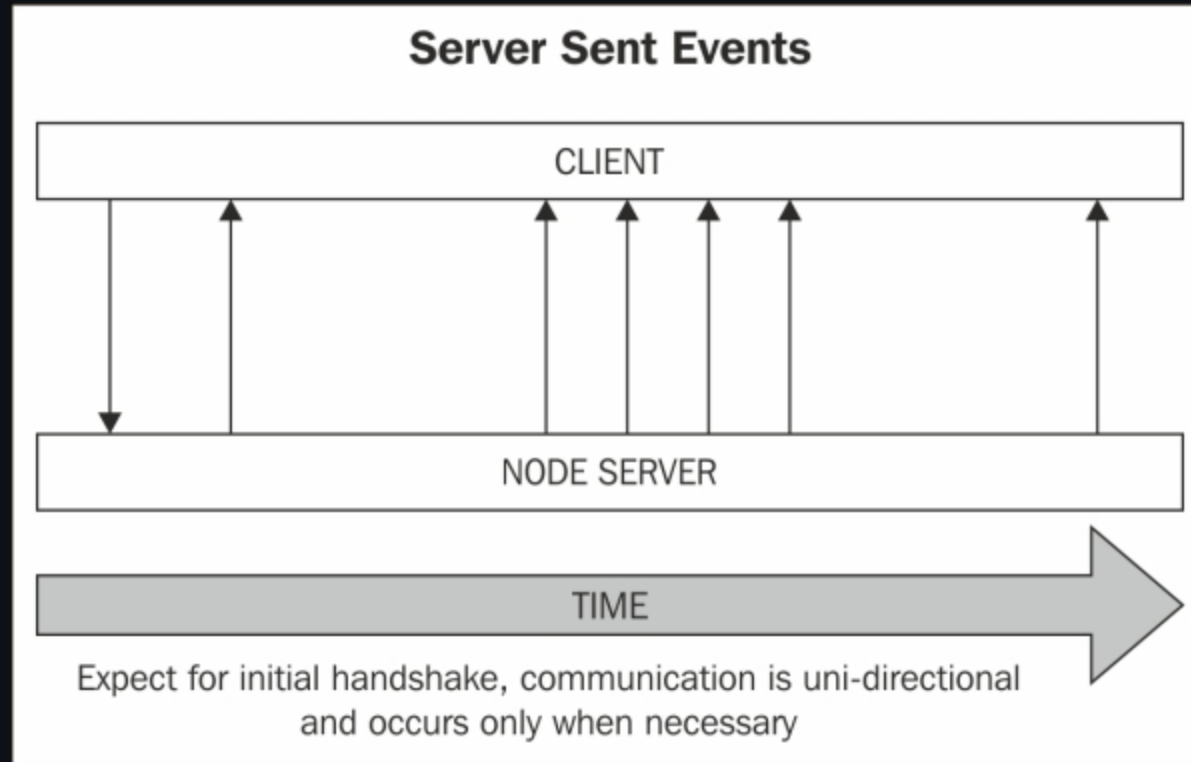
Content-Type:

text/event-stream; charset=utf-8

Cross-Origin-Opener-Policy:

Date:

SSE



- HTTP 프로토콜로 구현
- HTTP 연결이 유지되는 동안 서버가 클라이언트로 데이터를 즉시 전송 가능

SSE를 이용해 ChatGPT streaming 구현하기

What is ChatGPT?

Send



ChatGPT is a language model developed by OpenAI. It is powered by the GPT-3 architecture and aims to have more interactive and dynamic conversations 😊

FastAPI로 서버 구현하기

```
from openai import AsyncOpenAI
from fastapi.responses import StreamingResponse

aclient = AsyncOpenAI(
    api_key=OPENAI_API_KEY
)

async def stream_generator(query: str):
    completion = await aclient.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": query}],
        stream=True
    )

    async for response in completion:
        content = response.choices[0].delta.content
        if content is not None:
            yield content

@app.get("/completion")
async def stream(query: str):
    return StreamingResponse(stream_generator(query), media_type='text/event-stream')
```

FastAPI로 서버 구현하기

- `StreamingResponse` 와 `media_type='text/event-stream'` 를 이용해 SSE 구현
- 첫 번째 인자로 Async Generator를 받음

StreamingResponse 📄

Takes an async generator or a normal generator/iterator and streams the response body.

```
from fastapi import FastAPI
from fastapi.responses import StreamingResponse

app = FastAPI()

async def fake_video_streamer():
    for i in range(10):
        yield b"some fake video bytes"

@app.get("/")
async def main():
    return StreamingResponse(fake_video_streamer())
```


Generator

```
>>> def infinite_generator():  
...     count = 0  
...     while True:  
...         count+=1  
...         yield count  
...  
>>> gen = infinite_generator()  
>>> next(gen)  
1  
>>> next(gen)  
2  
>>> next(gen)  
3
```

FastAPI로 서버 구현하기

- `async` + `yield` 를 이용해 Async Generator 구현

```
async def stream_generator(query: str):
    completion = await aclient.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": query}],
        stream=True
    )

    async for response in completion:
        content = response.choices[0].delta.content
        if content is not None:
            yield content
```

React로 클라이언트 구현하기

```
const [data, setData] = useState<string[]>([]);

const fetchData = async () => {
  const response = await fetch(`http://localhost:8000/completion?query=${query}`); // FastAPI 서버의 스트리밍 URL
  const reader = response.body?.getReader();
  const decoder = new TextDecoder();

  if (!reader) return;

  while (true) {
    const { done, value } = await reader.read();
    if (done) break;

    const text = decoder.decode(value, { stream: true });
    setData((currentData) => [...currentData, text]);
  }
};
```

React로 클라이언트 구현하기

- HTTP 프로토콜을 사용하므로, `fetch` 를 이용해 서버로부터 데이터를 받음

```
const response = await fetch(`http://localhost:8000/completion?query=${query}`); // FastAPI 서버의 스트리밍 URL
```

React로 클라이언트 구현하기

- body는 `ReadableStream` 객체
- `getReader` 를 호출하면 reader를 얻을 수 있는데,
얻는 순간 stream이 lock되어 다른 reader를 얻을 수 없음

```
const reader = response.body?.getReader();
```

React로 클라이언트 구현하기

- `reader.read()` 를 호출하면 stream으로부터 하나의 data chunk를 읽음
- `done` : stream이 닫혔는지 여부

```
const { done, value } = await reader.read();
```

React로 클라이언트 구현하기

- `TextDecoder`를 이용해 data chunk를 디코딩

```
const decoder = new TextDecoder();  
const text = decoder.decode(value, { stream: true });
```

React로 클라이언트 구현하기

- 이 chunk를 state에 저장
- state를 이용해 화면에 렌더링

```
setData((currentData) => [...currentData, text]);

{
  data.map((item, index) => <span key={index}>{item}</span>);
}
```

```
▼ (3) ['Hi! How', ' can I assist', ' you today?'] ⓘ  
  0: "Hi! How"  
  1: " can I assist"  
  2: " you today?"
```


정리

- GPT는 Transformer-style model로, 답변을 순차적으로 생성
- 서버가 클라이언트로 데이터를 전송하는 방법: Polling, Long Polling, WebSocket, SSE
- ChatGPT streaming을 구현하기 위해 SSE를 사용
- FastAPI에서는 `StreamingResponse` 를 이용해 SSE를 구현
- React에서는 `fetch` 와 `ReadableStream` 을 이용해 SSE를 구현

감사합니다.