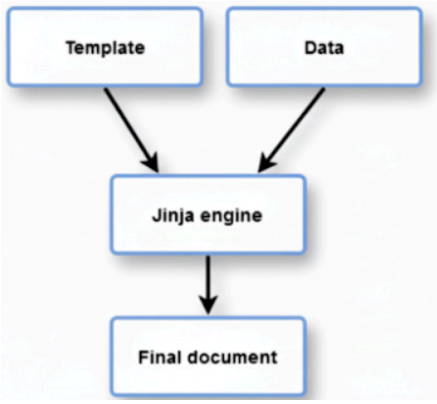
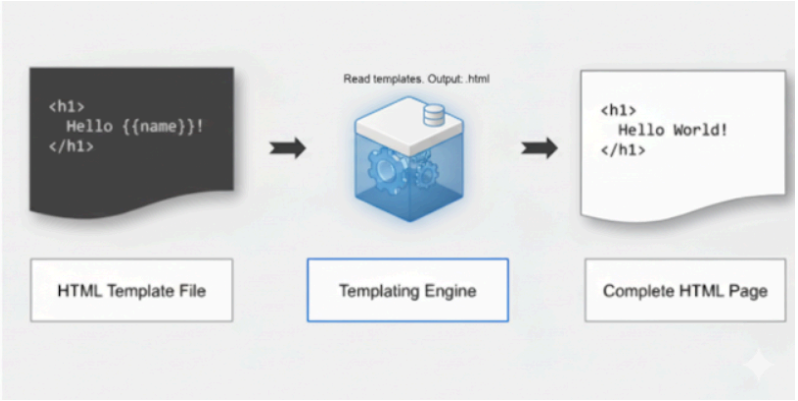
1. Jinja2Template 소개

태그 기반의 **HTML**과 같은 마크업은 파일이 길고 복잡해지게 되는데, 이러한 문제를 해결하기 위해 템플릿 엔진을 사용하여 표현**(Frontend)** 로직과 데이터/처리**(Backend)** 로직을 쉽게 분리

템플릿 엔진의 주요 장점:

- 역할 분리: 개발팀과 디자이너의 역할 분리
- 동적 콘텐츠 생성: 유연한 동적 콘텐츠 생성 지원
- 개발 효율성: 소스코드 모듈화 및 재사용성 증대





Python 기반의 대표적인 템플릿 엔진인 **Jinja**은 뛰어난 성능과 **Python**과 유사한 문법이 특징이며 **FastAPI**에 내장되어 제공

```python
from fastapi.templating import Jinja2Templates

templates = Jinja2Templates(directory="templates")
```

2. Jinja2Template Syntax

Jinja2는 Python 기반의 강력한 템플릿 엔진으로, HTML과 같은 정적 문서에 동적인 데이터와 로직을 삽입할 수 있으며 템플릿 파일 내에서 사용되는 주요 구문(Syntax)은 다음과 같이 세 가지 유형

2.1. 변수 **(Variable)** 및 표현식 출력 구문**: {{ ... }}**

이중 중괄호로 감싸진 부분은 Python에서 전달된 **변수(데이터)**의 값 또는 표현식의 결과를 템플릿에 출력할 때 사용

2.2. 제어 구조 **(Control Structures)** 구문**: {% ... %}**

중괄호와 퍼센트 기호로 감싸진 부분은 템플릿의 흐름을 제어하는 로직(조건문, 반복문 등)을 작성할 때 사용하며 출력되는 내용 없이 내부적으로 동작

2.3. 주석 **(Comments)** 구문**: {# ... #}**

중괄호와 샵 기호로 감싸진 부분은 템플릿 파일 내에서 개발자가 메모를 남기기 위해 사용하며, 최종 **HTML** 출력물에는 포함 안됨됨

구문	용도	예시 코드(HTML)
{{ ... }}	변수 또는 표현식의 값을 템플릿에 출력	Item id: {{ id }} query: {{ q_str }} {{ item }}
{% ... %}	if (조건문), for (반복문), macro, extends 등 로직 및 제어 구조를 정의	<body> {% if gubun == "admin" %} 이것은 관리용 item입니다. {% else %} 이것은 일반용 item입니다. {% endif %} {{item}} </body> <body> {% for item in all_items %} item name: {{ item.name }} item price: {{ item.price }} {% endfor %} </body>
{# ... #}	템플릿 내에 주석을 작성하며, 최종 출력 문서에서는 제거	{# 이 부분은 사용자에게 보이지 않는 주석입니다. #}

3. FastAPI에서 정적 파일(Static Files)을 처리하는 방법

CSS, JavaScript, 이미지**,** 정적 **HTML** 파일과 같이 내용이 동적으로 변하지 않는 파일들을 FastAPI 애플리케이션에서 어떻게 관리하는지 설명

3.1. 정적 파일의 특징 (첫 번째 불릿)

- 정적 파일은 (데이터를 동적으로 반환하는 API 엔드포인트와 달리) 내용이 변경되지 않고 그대로 제공되는 파일
- 예: CSS, JavaScript, 이미지(Image), 정적 HTML 파일 등

3.2. FastAPI의 처리 방식 (두 번째 불릿)

- FastAPI는 일반적인 엔드포인트(`@app.get("/")` 등)와 별도로 이 정적 파일들을 관리
- 정적 파일들을 위해 별도의 **ASGI** 서버를 생성하여 제공하며, 이를 위해 `StaticFiles` 클래스를 제공

3.3. `StaticFiles` 설정 코드 설명

제공된 코드는 `StaticFiles`를 사용하여 정적 파일을 서빙하는 방법

```
from fastapi.staticfiles import StaticFiles

app.mount("/static", StaticFiles(directory="static"), name="static")
```

코드 부분	의미
"/static"	URL Path, 사용자가 /static/style.css와 같은 경로로 접근할 때 사용되는 URL 접두사
StaticFiles(directory="static")	물리적인 Directory 명, FastAPI 서버를 실행하는 시스템의 파일 시스템에서 정적 파일들이 실제로 저장된 폴더 이름
name="static"	url_for()와 같은 함수를 사용하여 템플릿 등에서 이 정적 경로를 참조할 때 사용할 내부적인 이름

부록 : **JavaScript**가 '정적 파일'로 분류되는 이유

FastAPI나 다른 웹 프레임워크에서 **정적 파일(Static Files)**이라고 할 때, 여기서 '정적'의 의미는 파일 내용 그 자체가 서버 측에서 동적으로 생성되거나 변경되지 않는다는 것을 의미미

---

1. 파일 내용의 불변성 **(**서버 측 관점**)** 💾

- 정적**:** JavaScript 파일(`main.js` 등)의 코드 내용은 서버가 사용자에게 응답을 보낼 때마다 변하지 않고**,** 서버는 단순히 이미 저장된 파일 내용 전체를 읽어서 사용자 브라우저에 전송할 뿐
    - 예시*:* 서버는 `console.log("Hello")`라는 코드가 담긴 파일을 요청이 들어올 때마다 수정하지 않고, 항상 그 내용 그대로 전송
- 동적 **(API** 엔드포인트**):** 반면에, `app.get("/data")`와 같은 엔드포인트는 요청이 들어올 때마다 데이터베이스를 조회하거나 계산을 수행하여 **새로운 내용(JSON, HTML 등)**을 동적으로 생성해서 전송

---

2. 실행 시점과 역할의 분리 🚀

- **JavaScript (**클라이언트 사이드**):** JavaScript 코드는 서버에서 전송된 후, 사용자 브라우저(클라이언트)에서 실행되어 동적인 효과(DOM 조작, 애니메이션, 이벤트 처리)를 생성, 즉, 동작 자체는 동적이지만, 코드 파일 자체는 정적
- **FastAPI (**서버 사이드**):** FastAPI의 `StaticFiles`는 파일을 클라이언트에게 전달하는 역할만 담당하며, 그 파일 내부의 코드가 브라우저에서 어떤 동적인 행동을 할지는 관여하지 않음

결론**:** '정적'은 '서버가 생성하지 않는'다는 의미

따라서 CSS, 이미지 파일처럼 JavaScript 파일도 서버의 파일 시스템에 미리 저장되어 있으며, 서버는 요청 시 그 내용을 있는 그대로**(as-is)** 제공하기 때문에 정적 파일로 분류

파일 유형	서버 역할 (FastAPI 관점)
JS, CSS, Image	미리 저장된 파일을 읽어 사용자에게 그대로 전달 (정적)
API 엔드포인트	요청 시 새로운 데이터를 생성/계산하여 전달 (동적)