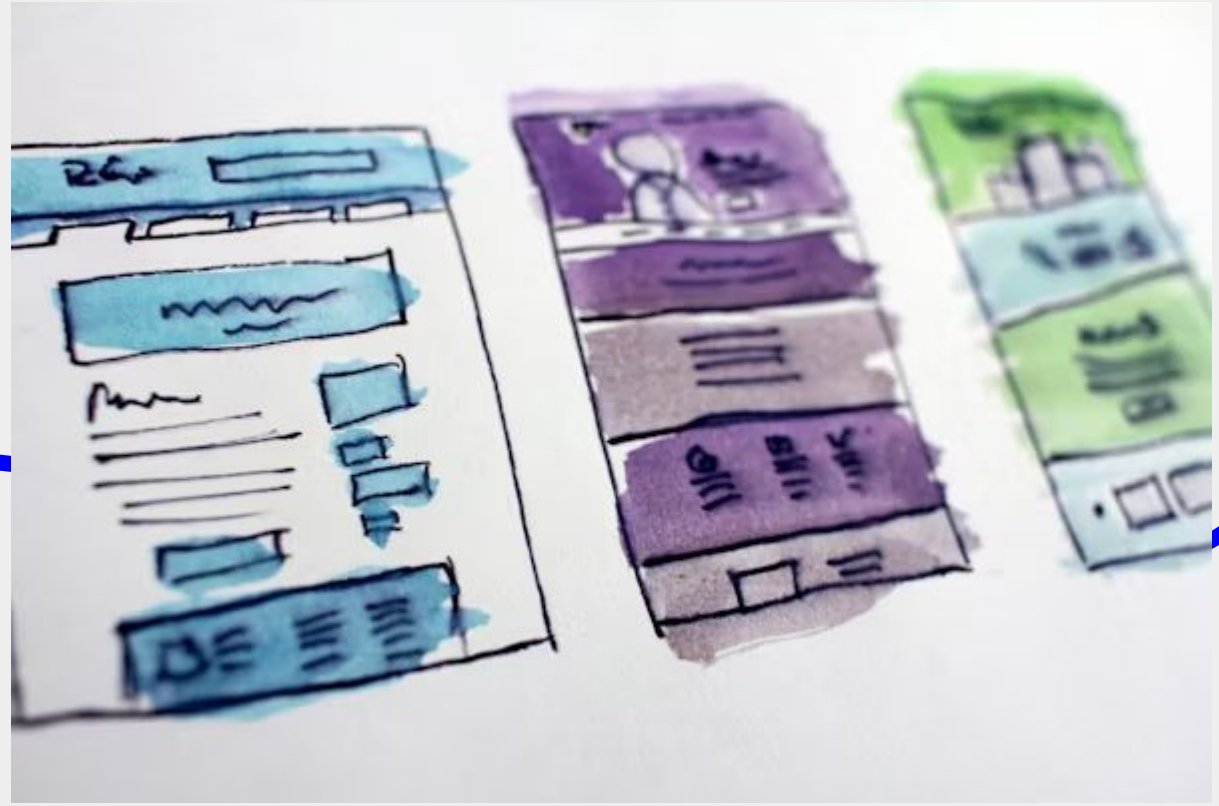
FastAPI Color Generator

Workflow & Code Analysis

Jinja2 ?

Data

Template



Display

System Overview

We will explore how Python, HTML, CSS, and JavaScript work together to create a dynamic web application.

Description of the workflow

[FastAPI 랜덤 색상 생성기 작동 흐름]

1. 사용자 요청 (User Request)

- **접속:** 사용자가 웹 브라우저를 통해 `http://localhost:8000/` URL에 접근
- **전송:** 브라우저가 서버(FastAPI) 측에 메인 페이지 호출(HTTP GET) 요청

2. 서버 로직 처리 (Server Logic - `main.py`)

- **수신:** `main.py` 내 `@app.get("/")` 라우터가 요청을 접수
- **생성:** 16진수 문자열(`0123456789abcdef`) 중 6자리를 무작위 추출하여 색상 코드 조합 (예: `#A1B2C3`)
- **준비:** 생성된 색상 데이터를 템플릿에 전달할 컨텍스트(`{"color": hex_color}`)로 구성

3. 템플릿 렌더링 (Template Rendering)

- **매핑:** Jinja2 템플릿 엔진이 `color.html` 파일 로드
- **주입:** HTML 내 `{{ color }}` 변수 위치에 생성된 16진수 색상 코드 삽입
- **결합:** 공통 레이아웃인 `base.html` 과 병합하여 최종 HTML 문서 완성

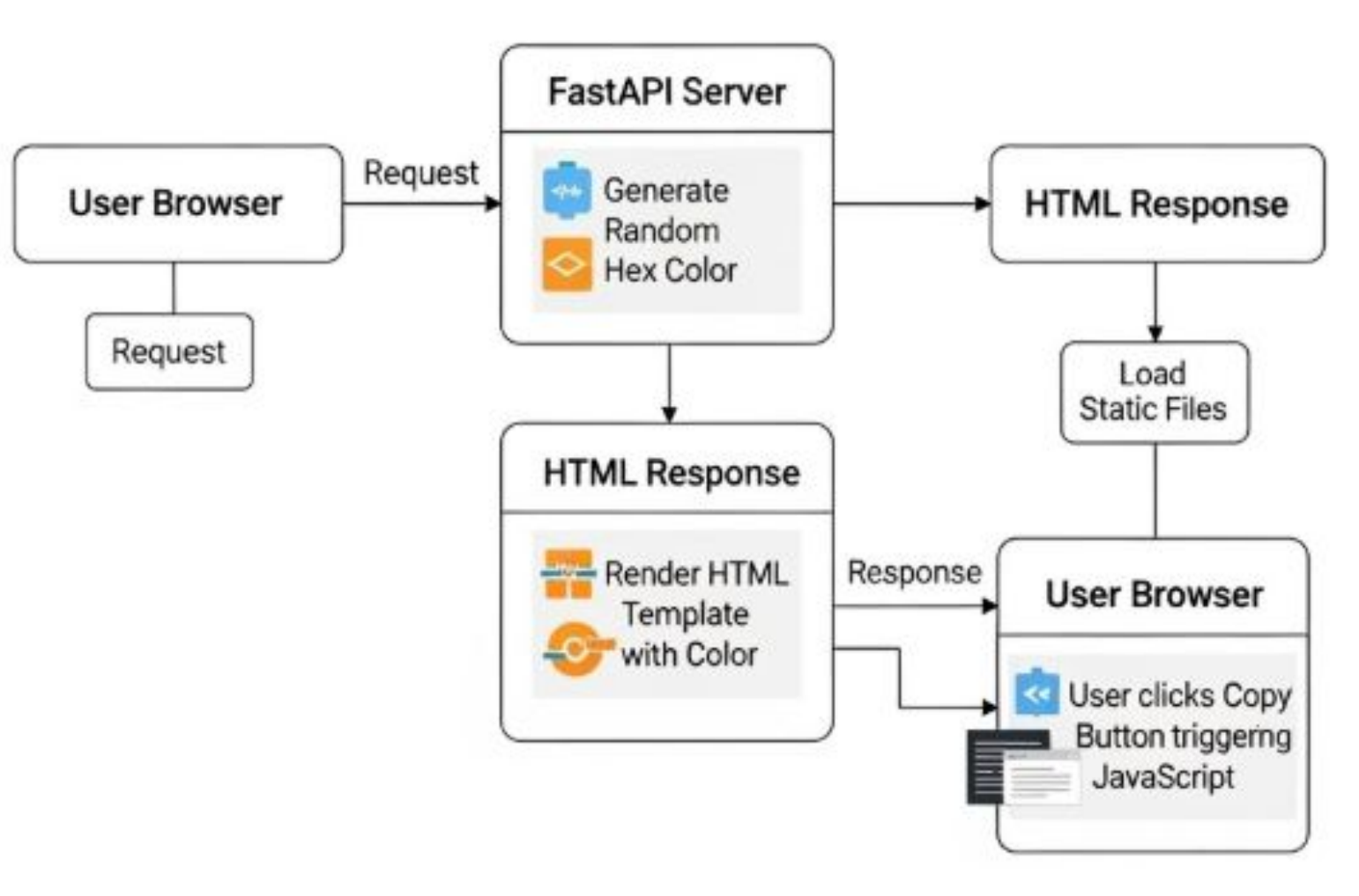
4. 응답 및 정적 파일 로드 (Response & Static Files)

- **반환:** 완성된 HTML 코드를 클라이언트(브라우저)로 응답 전송
- **해석:** 브라우저가 HTML 파싱 중 CSS 및 JS 파일 링크 식별
- **재요청:** `/static` 경로를 통해 `style.css` , `script.js` 파일 추가 요청
- **제공:** `app.mount` 설정을 통해 서버가 `static` 폴더 내의 해당 파일 제공
- **완료:** CSS 적용(배경색 변경) 및 화면 표시 완료

5. 사용자 상호작용 (Interaction - `script.js`)

- **행동:** 사용자가 화면 중앙의 **'Copy Hex Code'** 버튼 클릭
- **감지:** `script.js` 내 이벤트 리스너가 클릭 동작 감지
- **실행:** `navigator.clipboard` API를 호출하여 화면의 색상 코드를 사용자 클립보드에 복사

Architecture



Application Workflow

1. Request

User visits the site. main.py receives the GET request.

2. Logic

Server generates a random HEX code (e.g., #A1B2C3).

3. Response

Server injects color into HTML and sends it to browser.

4. Interaction

Browser loads CSS/JS. User clicks to copy the code.

The Engine: main.py

Core Logic

Routing: Uses @app.get("/") to handle homepage requests.

Hex Generation: Randomly selects 6 characters from '0123456789abcdef' to create a color string.

Template Response: Passes the generated color variable to the HTML template, connecting the backend to the frontend.

Understanding app.mount

The Purpose

This function connects a specific URL path to a physical folder on your server. It allows the browser to access files like CSS and JS directly.

```
app.mount("/static", ...)
```

The Anatomy

URL Path: "/static" (How the browser asks for it)

Directory: directory="static" (Where it really lives on your disk)

Name: name="static" (Internal nickname for FastAPI)

The Interface: Templates

base.html

The skeleton of the site. It holds common elements like and imports CSS/JS. It defines blocks for other files to fill.

color.html

The content. It extends base.html and fills the content block. It receives the {{ color }} variable to set the background.

Jinja2

The bridge. The syntax {{ color }} allows Python data to be dynamically inserted into the raw HTML before sending.

Static Files: CSS & JS

style.css (Design)

Centers content using Flexbox.

Sets font to Monospace for code clarity.

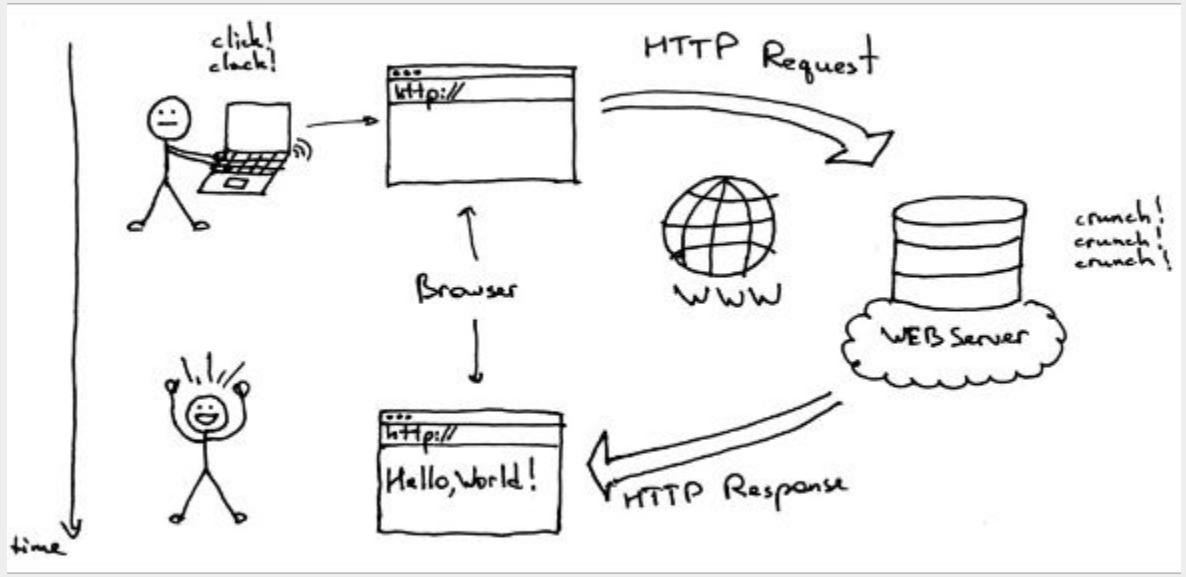
Ensures full height (100vh).

script.js (Action)

Listens for button clicks.

Uses navigator.clipboard API.

Copies the text content to the user's clipboard.



Key Takeaways

FastAPI

Handles Logic

Templates

Structure content

JavaScript

Adds Interactivity

Questions?

Thank you for exploring the Random Color Generator!

Image Sources

https://www.careersingovernment.com/tools/wp-content/uploads/et_temp/2-7-scaled-258785_1080x675.jpg

Source: www.careersingovernment.com

https://hackernoon.imgix.net/images/jot3yv6.jpg

Source: hackernoon.com

https://img.freepik.com/premium-vector/oops-404-error-with-broken-robot-concept-illustration_114360-1932.jpg

Source: www.freepik.com

https://static.vecteezy.com/system/resources/previews/069/429/377/non_2x/3d-colorful-abstract-background-overlap-layer-on-dark-space-with-glowing-circles-effect-decoration-modern-graphic-design-element-cutout-style-concept-for-web-art-flyer-card-or-brochure-cover-vector.jpg

Source: www.vecteezy.com