API ⌨️

웹 API는 웹 서버와 웹 브라우저 간의 애플리케이션 처리 인터페이스를 의미한다.

모든 웹 서비스는 API 이지만 모든 API가 웹 서비스는 아닌데
이는 API가 월드 와이드 웹 이전에 만들어졌기 때문에 JAVA API, 서비스 API 등 API 에 대한 다양한 용어가 존재한다.

API는 정의 자체로도 추상적이고, 동시에 다양한 해석이 가능해 처음 API를 접한다면 이해가 어려운데 간단히 표현하자면
API는 요청과 응답의 연속으로 데이터나 기능을 제공한다.

그렇다면 클라이언트는 어떤 방법으로 API를 사용하고 API는 어떻게 웹 서버에 요청을 전달 할 수 있을까?

정답은 정의된 코드의 실행이다.

API와 UI 관계 ⌨️

누구나 웹 브라우저의 주소창을 통해 API 요청을 작성하고
전달해 코드를 실행할 수 있지만 개발자가 아닌 사용자의 관점에서 빈 화면에 주소창만 있는 환경은 전혀 매력적이지 않고
접근하기 어렵기 때문에 웹 개발자는 GUI를 통해 접근성을 높이고 누구나 쉽게 코드를 실행해 API를 사용할 수 있게 유도한다.

웹툰을 보고 싶다면 브라우저에 그려진 이미지를 클릭해서 들어갈 수 있고,
로그인 버튼을 눌러 사이트에 접속 할 수 있는 것도 모두 코드의 실행을 UI를 통해 대신해주고 있는 것이다.

내용을 간략하게 정리하면 다음과 같다.

- API는 클라이언트와 서버의 중간에 위치한다.
- API는 서버에게 코드 실행을 요청한다.
- 이미지, 버튼 등 GUI를 통해 코드 실행의 접근성을 높인다

위 내용을 통해 API가 무엇인지 알게 되었고 웹 사이트를 이용한다면 누구나 코드를 실행하고 있다는 사실을 알 수 있게 되었지만,
아직 API가 어떻게 서버로 전달되는지 알 수 없다.

API와 서버 ⌨️

API 요청은 클라이언트에서 서버로 HTTP 프로토콜을 사용해 전송되는데 GET, POST, PUT, DELETE와 같은 HTTP 메시지로 전송되며
서버는 이 메시지를 바탕으로 요청에 맞는 데이터를 HTTP 응답으로 반환하게 된다.
(네트워크와 관련된 자세한 사항은 다음 포스트를 통해 정리하고 이번 포스트에선 API에 집중한다.)

다시 본론으로 돌아와 메시지 전송에 대해 알아보자

사용자의 접근성을 위해 GUI로 API의 실행을 대체하고, HTTP 메시지를 통해 서버로 전송된다.

이 문장으론 아직 완벽하게 API의 실행 방식을 설명하지 못한다.
만약 사용자 코드의 실행을 대체 할 GUI가 없다면 우리는 어떻게 API를 통해 코드를 실행할 수 있을까?

API 규칙 ⌨️

앞서 언급했듯 우리는 브라우저를 사용하고, 브라우저 주소창은 API를 작성하는 공간이다.
쉽게 말하자면 나는 구글 검색창에 네이버를 입력하고 돌아온 리턴 값을 클릭해 네이버라는 도메인을 가진 사이트에 접속할 수도 있지만
주소창에 www.naver.com을 직접 입력해 API를 호출하고 서버의 응답을 통해 네이버에 접속할 수도 있다.

이 결과는 과정의 차이만 있을 뿐 결과적으론 완벽히 같은데 이는 API가 가져야 할 내용이 정해져 있기 때문이다.

- API는 요청 방식을 가져야 한다(GET, POST ...).
- API는 어떤 데이터를 요청할 건지 알아야 한다(Endpoint)
- API는 자료 요청에 필요한 추가 정보를 가질 수 있다(Prameter)

일반적으로 뉴스 기사를 보거나, 웹툰을 보는 것은 모두 GET 요청을 통해 서버로부터 데이터를 전송 받는 것이고
회원 가입, 온라인 쇼핑은 POST 요청으로 새로운 데이터를 서버로 전송하는 과정이다.

이 때 naver와 같이 명확한 목적지가 있어야 하는데 이는 Endpoint로
각 Endpoint는 특정 데이터나 작업을 제공하거나 수행할 수 있는 URL을 뜻한다.

당장 유튜브에 접속해 영상을 클릭하게 되면 다음과 같은 URL을 볼 수 있다.

여기서 youtube.com/watch가 Endpoint에 속하고 ?로 구분된 부분이 파라미터로
다수의 동영상 중 클라이언트가 클릭한 특정 영상을 구분 지어주는 역할을 한다.
(위 URL로 들어가면 chatGPT는 엄준식 프로그래밍 언어도 잘할까 라는 영상이 나온다..)

이제 잘못된 문장을 고쳐보자

API는 정해진 규칙을 가지고 있다.

API를 통해 서버에서 원하는 데이터를 받고 싶다면 주소창을 통해 API 요청을 작성하면 되지만
이는 너무 까다롭고 누구나 알기 어렵기 때문에 개발자는 UI를 통해 누구나 쉽게 API 요청을 작

성할 수 있게
API 요청을 대체 하고 API는 HTTP 메시지를 통해 서버로 메시지를 전달한다.

API 흐름

1. 클라이언트가 API Endpoint URL을 호출한다.
2. API 서버는 클라이언트의 요청을 받는다.
3. 서버는 해당 요청에 맞는 데이터를 찾아서 응답을 만든다.
4. 서버는 응답을 HTTP 프로토콜을 사용해 클라이언트에게 전송한다.
5. 클라이언트는 API 응답을 받아 해당 데이터를 확인 할 수 있다.

```
[Client] <-- HTTP Request --> [API Server] <-- HTTP Response --> [Client]
```

API 제한 ⌨️

자바 언어에서 접근에 제한을 두듯 API 역시 비슷한 개념이 존재한다.
크게 세 가지로 Public Private Partner로 나뉘는데 OpenAPI, 공공 API가
Public에 속하고 누구나 자유롭게 사용할 수 있다.
Private API는 비공개 API로 회사 내에서만 사용되는 API가 존재할 경우 이에 해당하며
미리 정해둔 HOST만 사용할 수 있는 API는 Partner API이다.

참고 ⌨️

최신 웹 API는 REST API로 유연한 특성을 가져 현재 가장 많이 사용되는 API 이다.
주된 특징은 '무상태'로 서버가 요청 간에 클라이언트 데이터를 저장하지 않음을 의미한다.
서버에 대한 클라이언트 요청은 브라우저에 입력하는 URL과 유사하지만 서버 응답은 렌더링이 없는 일반 데이터이다.

마무리 ⌨️

모든 프로그램은 API를 가질 수 있다. Google, KAKAO, PAPAGO 등
대부분의 웹 사이트뿐만 아니라 프로그램은 API를 제공하고 API를 형식에 맞춰 사용한다면 누구나 쉽게 사용할 수 있다.
이는 내가 원하는 프로그램을 직접 개발하는 수고를 덜어주고 검증된 결과를 반환받을 수 있으므로 사용하지 않을 이유를 찾는 게 더 어렵다.

API라는 추상적인 개념은 결국 응답과 요청의 반복일 뿐이므로 너무 어렵게 생각할 필요는 없다.