

```
children: [  
  con(icon, color: color  
  container(  
    margin: const EdgeIns  
  child:  
    label  
    style
```

# Web 개발 기초 #0

이진용  
WEB Core

# 세션의 목표

- 웹 개발을 위한 최소한의 지식 전달
- 웹 서비스에 대한 이해
- 디테일한 기술보단 전체적인 **overview**를 이해

# Index

1. 웹이란?
2. 웹 서비스의 작동 방식
3. 웹 개발자들의 역할



Google Developer Groups

On Campus • Sogang University

# What is Web?

```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

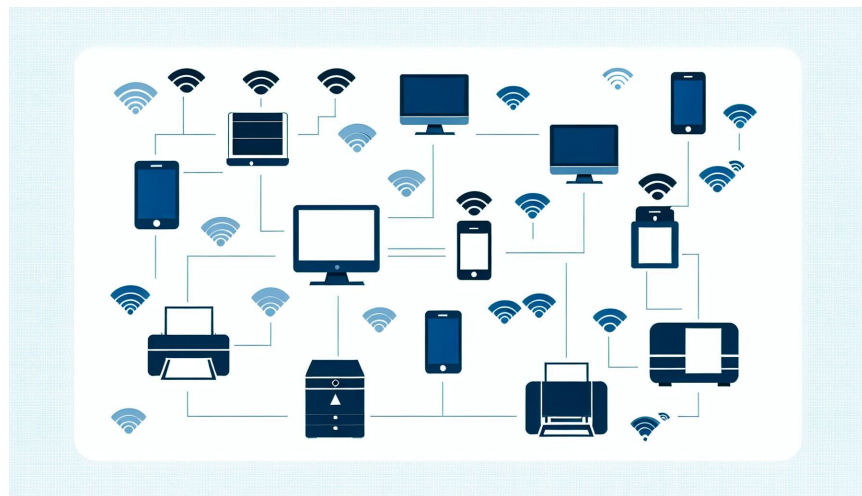
# Internet

전 세계에 걸쳐 원거리 접속이나 **파일 전송**, 전자 메일 등의 데이터 통신 서비스를 받을 수 있는, 컴퓨터 네트워크의 시스템.



# Internet

- 무선, 유선에 상관없이 인터넷에 연결되어 있다면 세계 모든 디바이스와 통신 가능
- PC, Mobile, Iot, printer ... 다양한 디바이스 통신 가능



# Web

## World Wide Web

- Internet 환경 안에서  
브라우저를 통해 파일을 공유
- 게시판, 검색, 동영상 스트리밍  
등 다양한 서비스 존재



# Web Service

- Web 환경에서 제공하는 서비스들
- IT 서비스 중, 일반 사용자에게 가장 친숙한 서비스
- Youtube, ChatGPT, Naver..
- 모바일도 웹 서비스로 제공하는 경우가 많음







Google Developer Groups

On Campus • Sogang University

# How Web working?

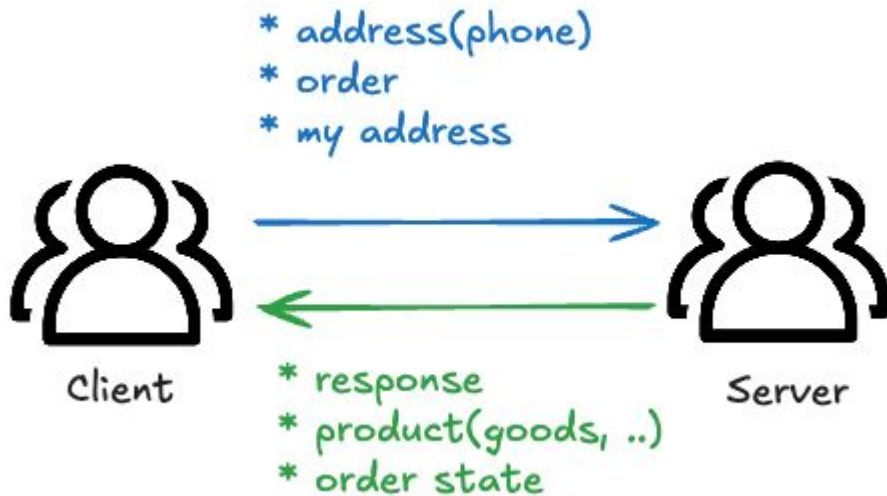
```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
lookup.StaticV  
_buckets=5)
```

# Service

**Client** : 소비자

**Server** : 서비스 제공자

- client가 상품을 주문하면 서비스 제공자가 이에 응답
- server의 입장에선 client의 요청을 기다리고 있음

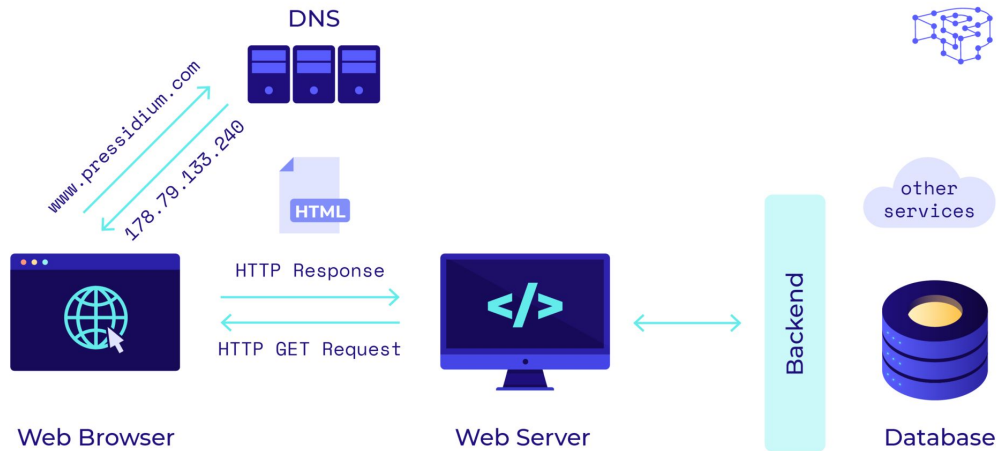


# Web Service

**Client**(개인 PC) : 브라우저

**Server**(서버 PC) : 웹 서버

- Client가 주문(**Request**)  
Server가 응답(**Response**)
- 제공하는 것은 HTML과 같이  
웹사이트를 그릴 수 있는  
파일



# URL

서버에 요청을 위한 링크

상품의 내용 및 서버 주소 정보

- **Protocol** : 웹 표준 통신 규약
- **Domain name** : 서버의 주소
- **Extension(path)** : 리소스의 위치

Protocol      Domain name      Extension

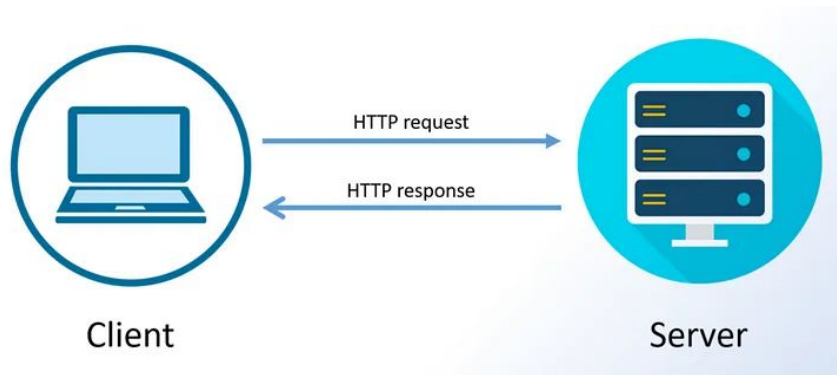
**http://www.example.com/blog/what-is-a-url**

URL

# HTTP

## Hypertext Transfer Protocol

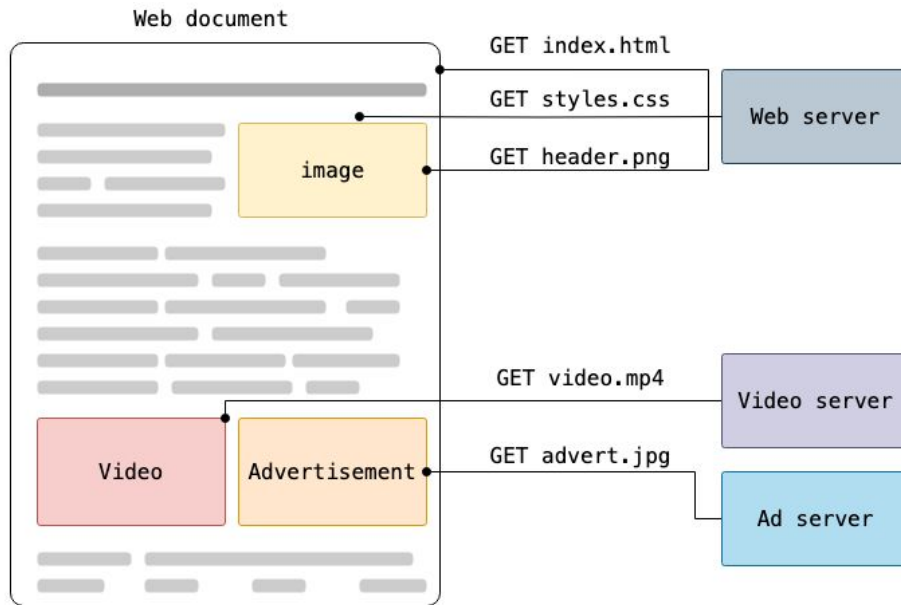
- Web에서 HTML 및 파일들을 전송하기 위한 **통신 규약**
- 브라우저와 서버간 통신은 해당 규약에 따라 이루어짐



# HTTP

## Hypertext Transfer Protocol

- Method (GET, POST ..)를 지정하여 서버와 통신
- 이미지, 비디오 등 다양한 파일들을 HTTP 통신을 통해 가져옴

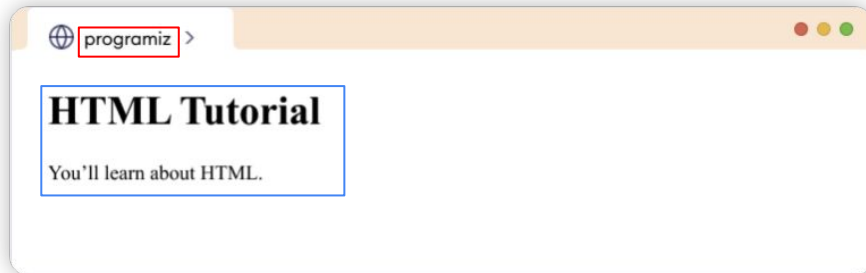


# HTML

## Hypertext Markup Language

- 브라우저가 읽어 화면을 그림
- 형식화된 포맷(태그)를 통해 웹 사이트 구성

```
<!DOCTYPE html>
<html>
  <head>
    <title>programiz</title>
  </head>
  <body>
    <h1>HTML Tutorial</h1>
    <p>You'll learn about HTML.</p>
  </body>
</html>
```



# Web browser (client)

## 주요 기능

- 주소(URL)을 통해 파일을 서버에게 요청
- HTML 파일을 통해 화면을 그림 (Word 프로그램과 동일)



**Safari**

Apple

MacOS, iOS



**Firefox**

Mozilla

MacOS, MS Windows, Linux OS,  
Android OS



**Chrome**

Google

MacOS, MS Windows, Linux OS,  
Android OS, Chrome OS



**Edge** new

Microsoft

MS Windows, MacOS, iOS,  
Android OS



**Opera**

Opera Software

MacOS, MS Windows, Linux OS,  
Android OS



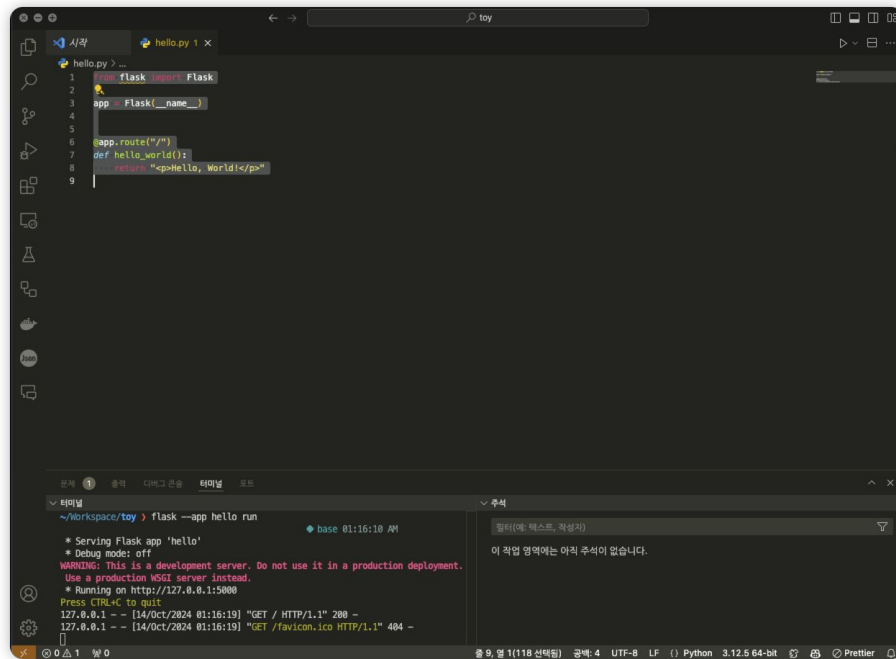


# Pop quiz

그러면 인터넷이 없어도 HTML파일만으로 브라우저는  
웹사이트를 그릴 수 있을까?

# Server (Web Server)

- 24시간 구동하며, 사용자의 요청을 기다림
- 서버도 하나의 컴퓨터
- 요청한 파일을 Client(browser)에게 전달
- 데이터를 저장하여, 여러 응답 처리



The image shows a VS Code editor window with a file named 'hello.py'. The code is a simple Flask application:

```
1 flask --help Flask
2
3 app = Flask(__name__)
4
5 @app.route("/")
6 def hello_world():
7     return "<ep>Hello, World!</ep>"
8
9
```

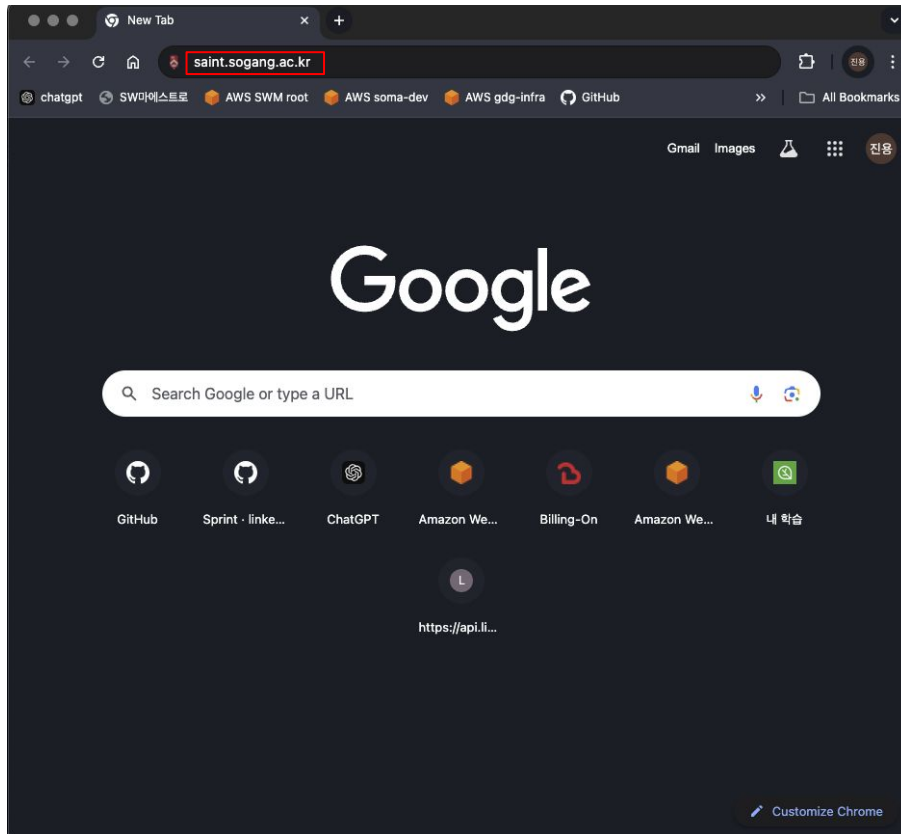
Below the editor is a terminal window showing the command to run the application and its output:

```
~/Workspace/toy > flask --app hello run
* Serving Flask app 'hello'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL-C to quit
127.0.0.1 -- [14/Oct/2024 01:16:19] "GET / HTTP/1.1" 200 -
127.0.0.1 -- [14/Oct/2024 01:16:19] "GET /favicon.ico HTTP/1.1" 404 -
```

간단한 파이썬 프로그램으로 서버 구동

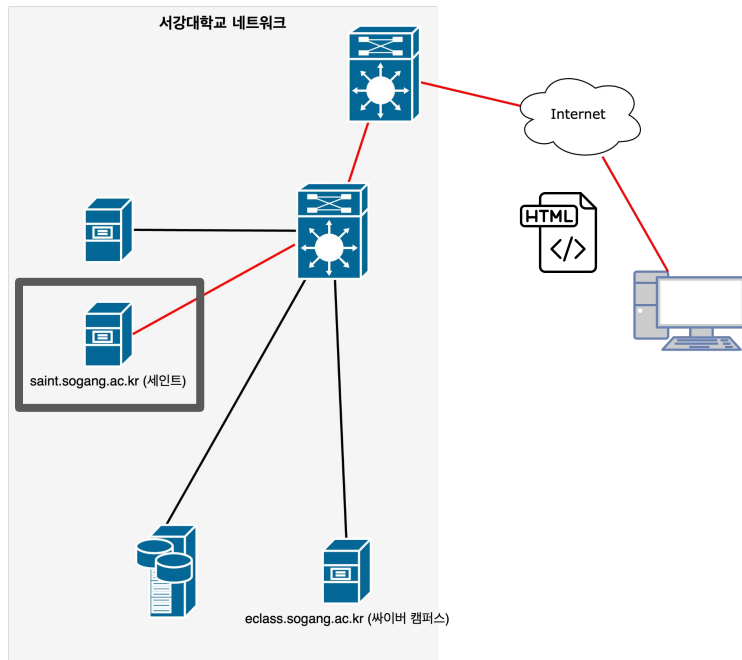
# Example (saint website)

1. 서버의 주소(URL)을 브라우저에 입력
2. HTTP 요청을 해당 주소의 서버에 전송



# Example (saint website)

1. PC(브라우저)에서 인터넷을 거쳐 학교 네트워크 전달
2. saint 웹 서비스를 운영중인 서버에게 요청
3. 웹 서버는 saint 사이트를 그릴 수 있는 HTML을 내 PC에 전송



## 전달 받은 HTML을 통해 화면 구성





Google Developer Groups

On Campus • Sogang University

# Who is Web Developer?

```
lookup.KeyValue  
f.constant(['em  
=tf.constant([G  
.lookup.StaticV  
_buckets=5)
```

# Web developer

- 사용자 요구사항이나 비즈니스 목표를 기술로 구현
- 브라우저에서 보이는 **화면(UI)**을 구현하고, 그 동작을 처리하는 **로직 개발**
- 데이터를 안전하게 저장하고, 요청에 따라 데이터를 제공하는 로직 개발
- 서버를 인터넷에 배포(실제 운영 환경에 올리기)

# Frontend vs. Backend

하나의 웹서비스를 만드는데, 많은 태스크 존재

- 디자인 퍼블리싱
- UI/UX 로직 구현
- 브라우저 로직 작성
- 사용자, 콘텐츠 데이터 저장
- 네트워크 설정
- 서버 관리
- ...



# Frontend vs. Backend

- 디자인 퍼블리싱
- UI/UX 로직 구현
- 브라우저 로직 작성
- 사용자, 콘텐츠 데이터 저장
- 네트워크 설정
- 서버 관리
- ...



## FE (사용자가 사용하는 **client** 개발)

- 디자인 퍼블리싱
- UI/UX 로직 구현
- 브라우저 로직 작성

## BE (사용자의 요청을 처리하는 **server** 개발)

- 사용자, 콘텐츠 데이터 저장
- 네트워크 설정
- 서버 관리

# FE (front-end)

- 웹 페이지 디자인 구현 : HTML, CSS 등을 사용해 시각적인 요소를 구현
- **\*UI/UX 개발** : 사용자 경험(UX)을 고려한 인터페이스(UI) 설계 및 개발
- 브라우저에서 동작하는 로직 개발 : \*JavaScript 등을 사용하여 웹 페이지의 동적 동작을 구현
- 사용자와의 상호작용 처리 : 버튼 클릭, 폼 제출 등 사용자의 입력 처리

\*UI/UX : User Interface, User Experience 사용자가 보는 화면 및 발생 이벤트

\*Javascript : 브라우저에서 실행되는 프로그램 언어로, 동적으로 데이터 통신 및 화면 구성에 사용

# BE (back-end)

- **\*데이터베이스 관리** : 데이터를 저장하고 불러오는 로직 구현
- **서버 로직 개발** : 클라이언트 요청에 맞게 데이터를 처리하고 응답하는 로직 개발
- **\*API 개발 및 관리** : 프론트엔드와 통신할 수 있는 API 제공
- **서버 운영 및 배포** : 서버를 설정하고 운영하며, 이를 안정적으로 유지 및 관리

\*데이터베이스 : 주문 정보, 사용자 정보등 별도의 저장이 필요한 데이터를 저장 및 관리하는 프로그램

parameter, url)을 미리 정의

# QnA

```
children: [
  Icon(icon, color: color),
  Container(
    margin: const EdgeInsets,
    child: Text(
      label,
      style: TextStyle,
```

감사합니다 :)

