

**<https://areyouhere.today/>**

**Passcode**



**HJLT**

# FastAPI Seminar

**Week 0. Why FastAPI?**

# Table of Contents

**1. FastAPI 소개**

**2. 배경 지식**

# 1. FastAPI 소개

# FastAPI 가 뭔가요?

## FastAPI 의 특징

- 빠르고
- 빠르게 개발할 수 있고
- 버그가 적고
- 직관적이고
- 배우기 쉽고
- 간결하고
- 단단하고
- 표준을 준수하는

- **Fast:** Very high performance, on par with **NodeJS** and **Go** (thanks to Starlette and Pydantic).  
[One of the fastest Python frameworks available.](#)
- **Fast to code:** Increase the speed to develop features by about 200% to 300%. \*
- **Fewer bugs:** Reduce about 40% of human (developer) induced errors. \*
- **Intuitive:** Great editor support. [Completion everywhere](#). Less time debugging.
- **Easy:** Designed to be easy to use and learn. Less time reading docs.
- **Short:** Minimize code duplication. Multiple features from each parameter declaration. Fewer bugs.
- **Robust:** Get production-ready code. With automatic interactive documentation.
- **Standards-based:** Based on (and fully compatible with) the open standards for APIs: [OpenAPI \[↗\]](#) (previously known as Swagger) and [JSON Schema \[↗\]](#).

# FastAPI 가 뭔가요?

## FastAPI 의 특징 - 정적 타입과 Pydantic

```
from fastapi import FastAPI
from pydantic import BaseModel

class Item(BaseModel):
    name: str
    description: str | None = None
    price: float
    tax: float | None = None

app = FastAPI()

@app.post("/items/")
async def create_item(item: Item):
    return item
```

# FastAPI 가 뭔가요?

## FastAPI 의 특징 - 의존성 주입

```
from typing import Annotated

from fastapi import Cookie, Depends, FastAPI

app = FastAPI()

def query_extractor(q: str | None = None):
    return q

def query_or_cookie_extractor(
    q: Annotated[str, Depends(query_extractor)],
    last_query: Annotated[str | None, Cookie()] = None,
):
    if not q:
        return last_query
    return q

@app.get("/items/")
async def read_query(
    query_or_default: Annotated[str, Depends(query_or_cookie_extractor)],
):
    return {"q_or_cookie": query_or_default}
```

# FastAPI 가 뭔가요?

## FastAPI 의 특징 - 의존성 주입

```
from typing import Annotated

from fastapi import Cookie, Depends, FastAPI

app = FastAPI()

def query_extractor(q: str | None = None):
    return q

def query_or_cookie_extractor(
    q: Annotated[str, Depends(query_extractor)],
    last_query: Annotated[str | None, Cookie()] = None,
):
    if not q:
        return last_query
    return q

@app.get("/items/")
async def read_query(
    query_or_default: Annotated[str, Depends(query_or_cookie_extractor)],
):
    return {"q_or_cookie": query_or_default}
```



# FastAPI 가 뭔가요?

## FastAPI 의 특징 - 간결한 구조적인 문법과 비동기 지원

```
from fastapi import FastAPI
from pydantic import BaseModel

class Item(BaseModel):
    name: str
    description: str | None = None
    price: float
    tax: float | None = None

app = FastAPI()

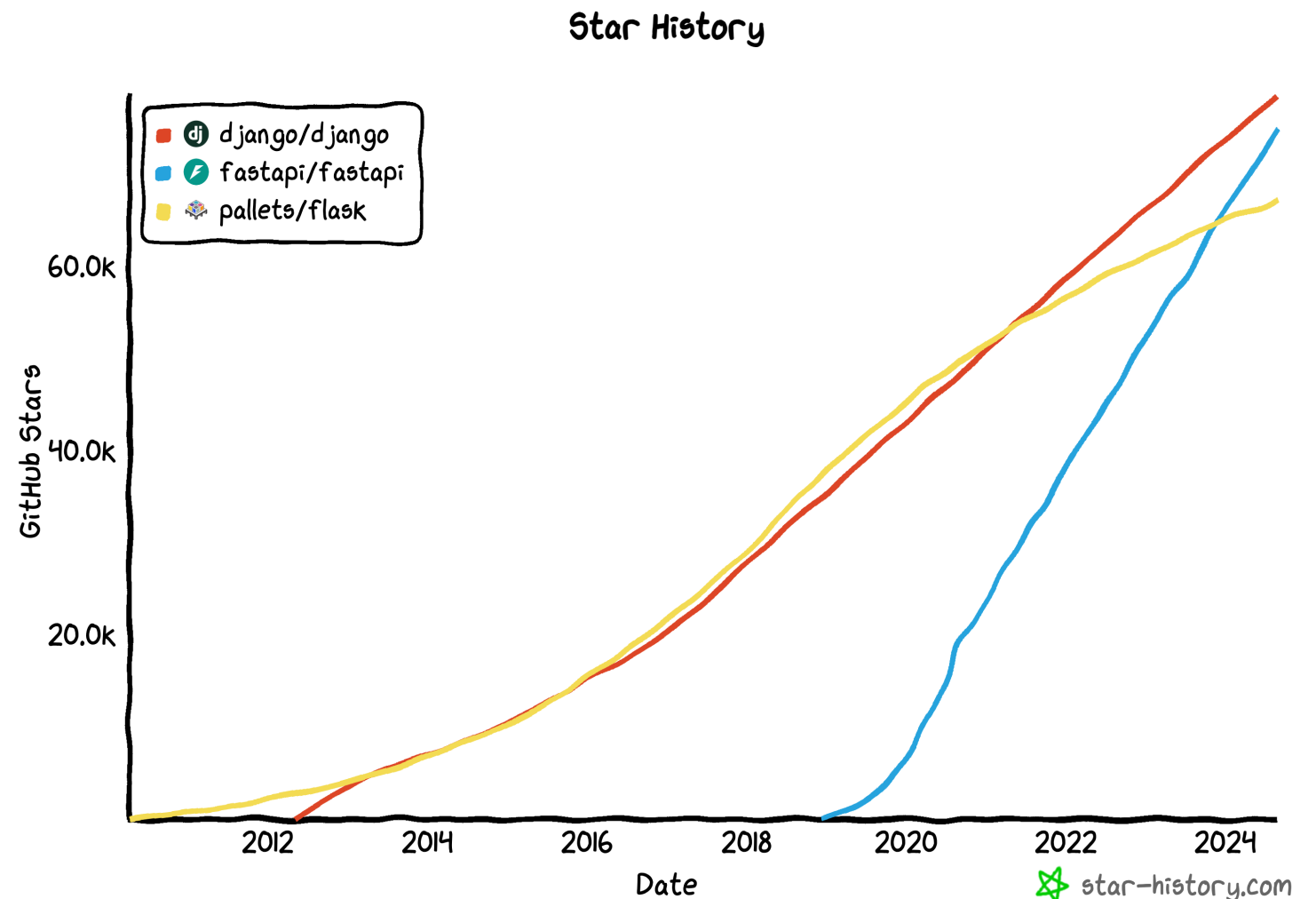
@app.post("/items/")
async def create_item(item: Item):
    return item
```

# FastAPI 가 뭔가요?

## FastAPI vs Other Frameworks










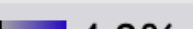


django




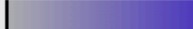

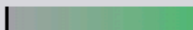




# FastAPI 가 뭔가요?

## FastAPI vs Other Frameworks

### 2021

Rnk	Framework	JSON	1-query	20-query	Fortunes	Updates	Plaintext	Weighted score
70	 nestjs	322,836	85,697	7,817	67,333	4,929	593,286	<b>1,349</b>    13.9%
72	 spring	150,259	102,803	15,979	23,401	7,329	183,737	<b>1,302</b>    13.4%
73	 fastapi	170,487	65,981	12,774	52,095	5,871	158,804	<b>1,202</b>    12.4%
105	 flask	62,895	34,169	6,542	23,573	1,367	83,223	<b>467</b>    4.8%
111	 django	73,024	20,179	1,643	15,508	882	79,266	<b>280</b>    2.9%

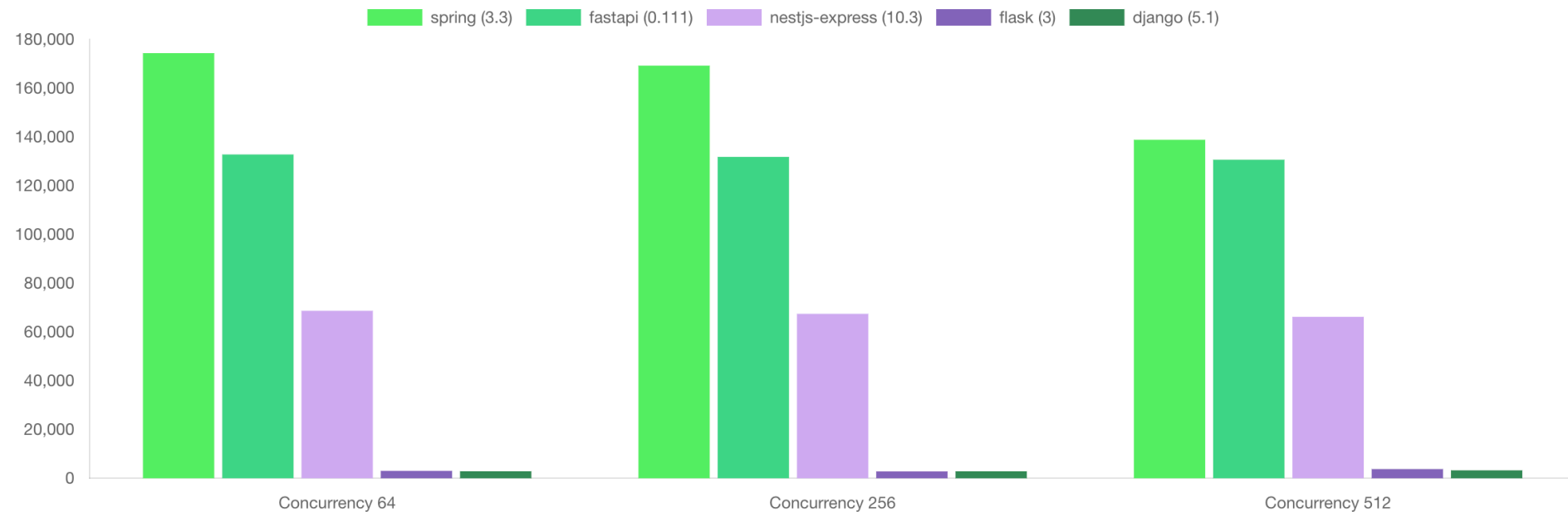
### 2023

Rnk	Framework	JSON	1-query	20-query	Fortunes	Updates	Plaintext	Weighted score
86	 fastapi	366,204	72,724	11,373	46,896	7,610	448,130	<b>1,524</b>    18.1%
88	 spring	236,259	147,907	15,932	24,082	7,131	506,087	<b>1,507</b>    18.0%
106	 nestjs	270,076	76,938	5,975	61,081	3,641	419,035	<b>1,099</b>    13.6%
143	 django	177,099	19,032	1,623	14,707	871	300,170	<b>413</b>    5.1%

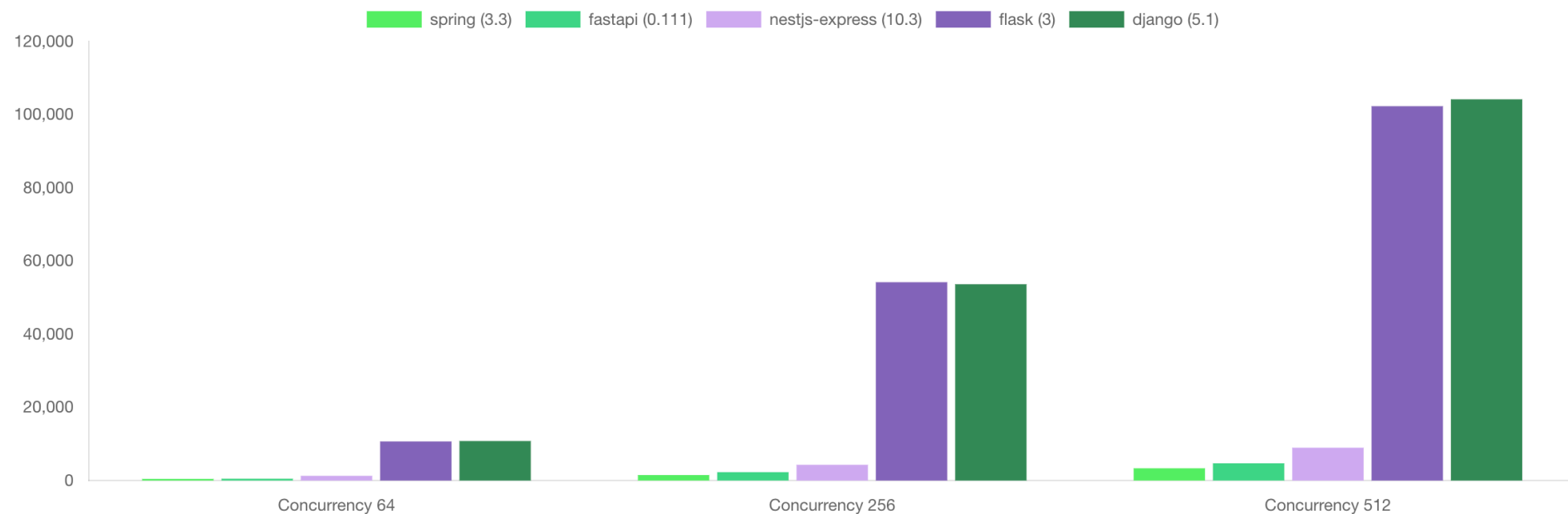
# FastAPI 가 뭔가요?

## FastAPI vs Other Frameworks

Total Requests per Second



Average Latency (ms)



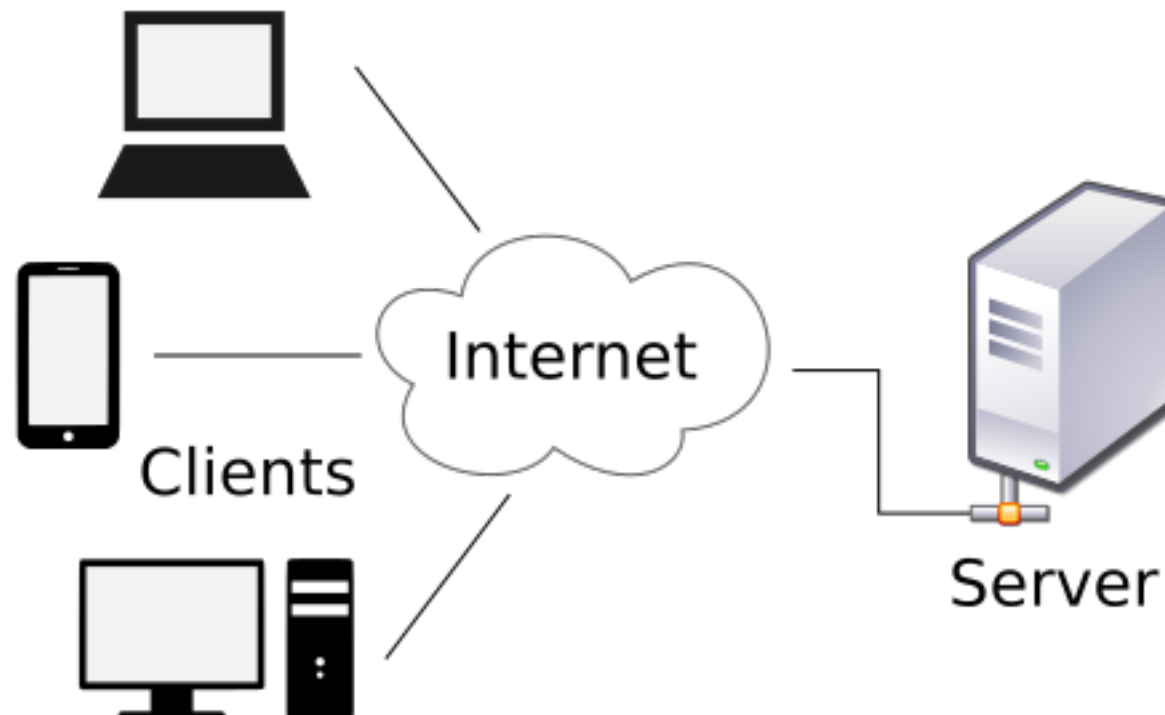
출처: <https://web-frameworks-benchmark.netlify.app/>

## 2. 배경 지식

# 서버 기초 개념

## 클라이언트-서버 모델

- 정의: 서비스 제공자와 요청자를 분리하는 일종의 동시성 모델
- 인터넷을 통해서 통신할 수도 있고, 같은 하드웨어 내에 존재할 수도 있음
- 상대적인 개념이므로 여러분의 서버도, 누군가의 클라이언트일 수 있음

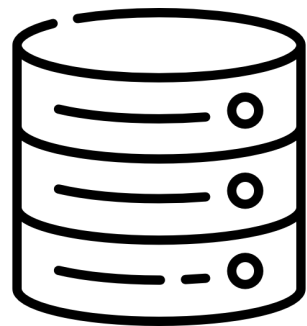
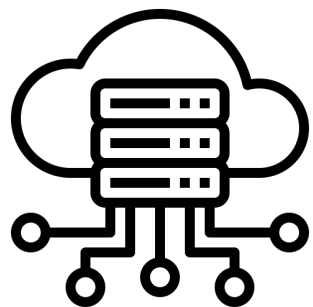


# 서버 기초 개념

## 클라이언트-서버 모델

### 서버

- 요청받은 서비스를 제공하는 소프트웨어
- e.g. 웹 서버, 데이터베이스



### 클라이언트

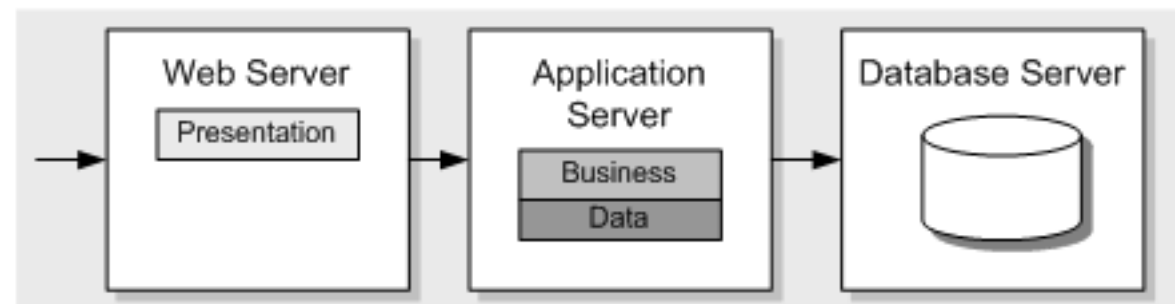
- 서버에 서비스를 요청하는 소프트웨어
- e.g. 웹 브라우저, 모바일 애플리케이션 등



# 서버 기초 개념

## 웹 서버, 애플리케이션 서버, 데이터베이스 서버

- 웹 서버는 서버에 존재하는 자원을 호스팅
- 애플리케이션 서버는 동적 콘텐츠 생성을 할 수 있도록 웹 서버를 확장
- 데이터베이스 서버는 데이터베이스 애플리케이션을 실행하는 서버



- 여러분은 앞으로...

"웹 프레임워크"를 사용해서 "애플리케이션 서버"를 통해 "웹 서버"로 전달되는 동적 콘텐츠를 생성하는 "웹 애플리케이션"을 개발



# 서버 기초 개념

## CGI, WSGI, ASGI

- **CGI (Common Gateway Interface)**
  - : 웹 서버가 외부 프로그램을 실행할 수 있도록 하는 공통 인터페이스
  - \* 외부 프로그램이란 스크립트 또는 컴파일된 프로그램을 말함
- **WSGI (Web Server Gateway Interface)**
  - : 파이썬으로 작성된 애플리케이션을 실행할 수 있도록 하는 인터페이스
- **ASGI (Asynchronous Server Gateway Interface)**
  - : WSGI의 슈퍼셋으로, 비동기 기능을 갖춘 파이썬 애플리케이션을 실행

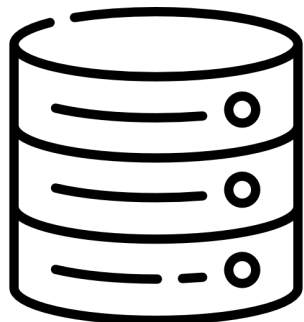
```
def application(environ, start_response):  
    start_response('200 OK', [('Content-Type',  
                                'text/plain')])  
    yield b'Hello, World!\n'
```

```
async def application(scope, receive, send):  
    event = await receive()  
    ...  
    await send({"type": "websocket.send",  
    ...})
```

# 서버 기초 개념

## HTTP

- HyperText Transfer Protocol
- 프로토콜이란?  
서로 다른 소프트웨어, 하드웨어에서 정보 전달 방법과 형태에 대한 약속



HTTP 요청

"/index.html" 이라는 이름의 문서를 내놔!



HTTP 응답

옳다! HTML 포맷이고 길이는 2181 자다!



# 서버 기초 개념

## HTTP 리소스

- 리소스 -> 콘텐츠의 원천
- 파일 시스템 상에 존재하는 정적 콘텐츠
- 애플리케이션 서버를 통해 만들어지는 동적 콘텐츠
- HTML, 워드 파일, JPEG, AVI, ...  
바이트로 표현해서 전달할 수 있는 모든 것들

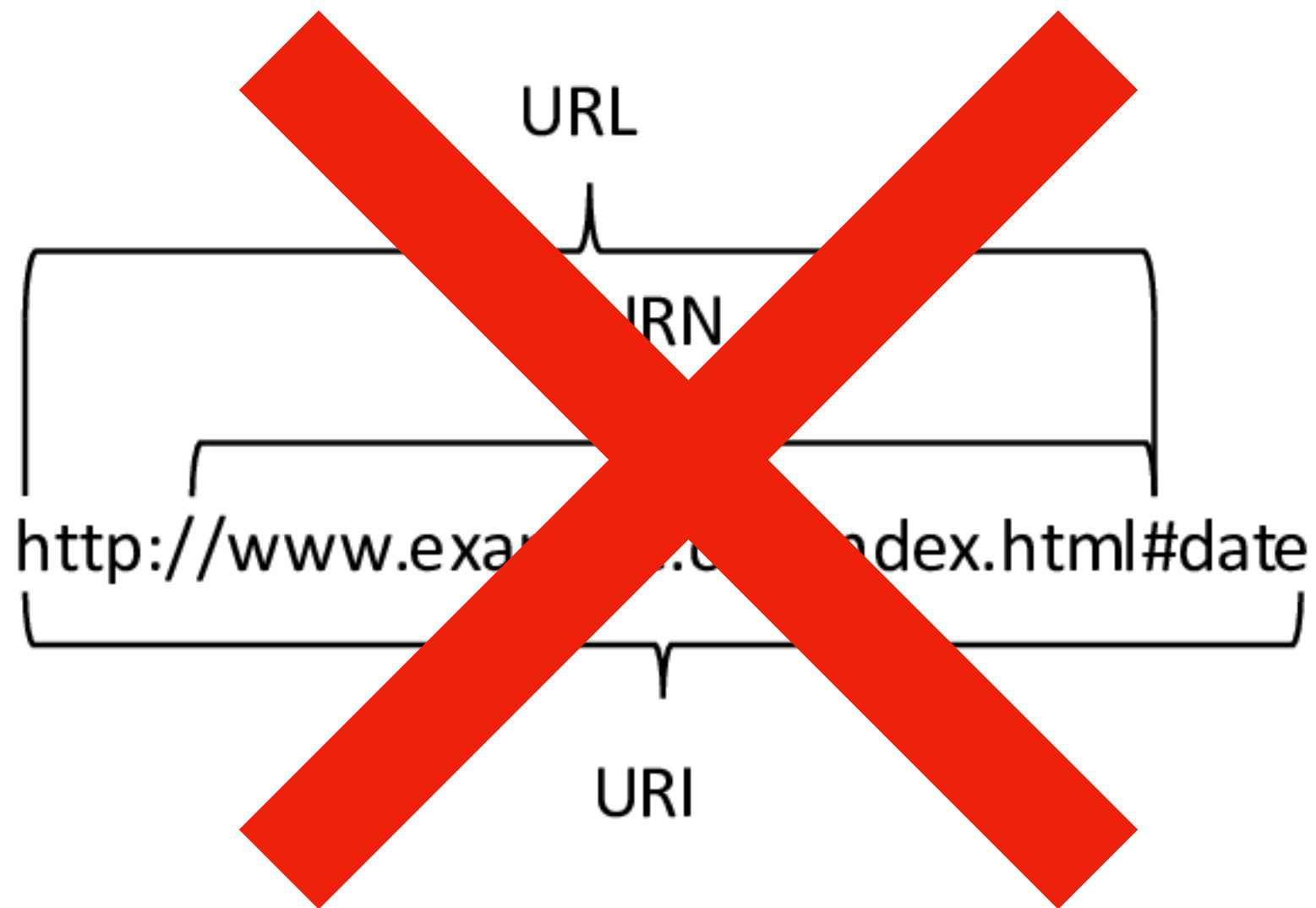
# 서버 기초 개념

## HTTP 리소스 - 미디어 타입

- 미디어 타입이란?  
HTTP 바디를 통해 전달되는 데이터의 포맷
- HTTP 를 통해 전달되는 모든 객체에는 MIME (다목적 이메일 확장) 이라는 타입 정보가 주어진다.
- e.g.  
Content-Type: text/html; utf-8  
Content-Length: 3199

# 서버 기초 개념

## HTTP 리소스 - URI, URL, URN



# 서버 기초 개념

## HTTP 리소스 - URI, URL, URN

- **URI (Unified Resource Identifier)**
- **URL (Unified Resource Locator)**
- **URN (Unified Resource Name)**

# 서버 기초 개념

## HTTP 리소스 - URI, URL, URN

### URL

Identify and locate resources

Authority  
http://www.mysite.com:80/path/to/myhome.html?product=camera#SomewhereinDoc  
Scheme Domain name port path to file query anchor



### URI

Identify resources

mailto://sathvik@mysite.com  
Scheme path

idap://2001:db8::7/c=GB?objectClass?one  
scheme authority path query



A URL is a subtype of URI

### URN

Identify resources

urn:isbn:123456  
Scheme NameSpace NameSpace specific string



A URN is a subtype of URI

# 서버 기초 개념

## 지식의 출처

- 블로그를 멀리 하세요!  
적어도, 어떤 개념의 정의나 근본을 추구할 때에는 말이죠
- 추천하는 지식의 출처
  - RFC (Requests for Comments)  
e.g. RFC-3986
  - PEP (Python Enhancement Proposal)  
e.g. PEP-20
  - 그리고 각종 공식문서들  
e.g. SQLAlchemy (Aysnchrohous I/O)



# 서버 기초 개념

## REST (Representational State Transfer)

- URL 은 단일 리소스를 식별한다.  
http://waffle-sns.com/get-post-3 (X)  
http://waffle-sns.com/post/3.html (O)
- Stateless  
상태가 전혀 존재하지 않는다는 뜻이 아님  
클라이언트와 서버 각각 상태를 저장하고 있으며, 클라이언트가 어떤 상태인지 서버는 신경쓰지 않는다는 의미
- 표준화된 메서드  
GET http://waffle-sns.com/post/3.html : 3번 포스트를 가져와!  
PUT http://waffle-sns.com/post/3.html : 3번 포스트로 저장해!
- RESTful 서비스: REST 원칙을 지키는 서비스

**Q&A**