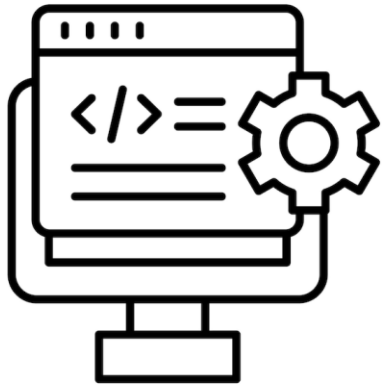
INT 161

Basic Backend Development
(การพัฒนาแบ็กเอนด์พื้นฐาน)



RESTful API Development

Course Description

ความรู้เบื้องต้นเกี่ยวกับ RESTful API การเชื่อมต่อกับฐานข้อมูล API ระบุตัวตน และความปลอดภัย การเรียงลำดับ การแบ่งข้อมูลออกเป็นหน้า และการกรองข้อมูล

Basic RESTful API, Connection with Database, API Authentication and Security, Sorting, Pagination and Filtering

PLO: 2C-I; 2D-E; 2F-E; 5BE-I; 5FS-I

2C ออกแบบองค์ประกอบทางด้านเทคโนโลยีสารสนเทศที่สนองความต้องการทางธุรกิจได้

2D พัฒนาองค์ประกอบทางด้านเทคโนโลยีสารสนเทศตามที่ได้กำหนดไว้ได้

2F ทดสอบองค์ประกอบและการเชื่อมต่อขององค์ประกอบทางด้านเทคโนโลยีสารสนเทศได้อย่างครบถ้วนบนข้อจำกัดที่มีได้

5BE ออกแบบและพัฒนาเว็บแอปพลิเคชันส่วนแบ็กเอนด์ที่ใช้ในระดับองค์กรได้

5FS ออกแบบและพัฒนาเว็บแอปพลิเคชันที่ใช้ในระดับองค์กร ทั้งในส่วนฟรอนเอนด์ และแบ็กเอนด์ได้

PLO แบ่งเป็น 5 ระดับ คือ

1. Introduce (I) – ระดับแนะนำ
2. Expand (E) – ต่อยอดจากองค์ความรู้/ทักษะ เดิมที่ได้พัฒนามาก่อนหน้า (INT141, INT150)
3. Strengthen (S) – มีความชำนาญมากขึ้น
4. Competence Level 1 (C1) – สมรรถนะในเชิงกว้างที่เป็นที่คาดหวังของบัณฑิตของหลักสูตร
5. Competence Level 2 (C2) – ผู้เรียนแสดงสมรรถนะในเชิงลึกที่เฉพาะเจาะจง ในสายอาชีพที่ผู้เรียนเลือก

Evaluation

Assignment 20%

2nd Exam. (Theory) 30%

- Multiple choices, single choose
- Describe

3rd Exam. (Theory) 20%

- Multiple choices, single choose

3rd Exam. (Practice) 30%

- Write a program to solve the problem.

Resources:

- https://nodejs.org/en/learn/getting-started/introduction-to-nodejs
- https://restfulapi.net/
- https://restful-api.dev/

Course outline - G2 (Monday)

No.	Date	Topic
1	4 Aug 2025	Introduction
2	11 Aug 2025	Bridge Public Holiday ***
3	18 Aug 2025	Introduction to RESTful API
4	25 Aug 2025	MVC / Express framework
5	1 Sep 2025	Connection to Database/ORM
	8 -12 Sep 2025	1st Examination
6	15 Sep 2025	Relationship Modeling with Prisma
7	22 Sep 2025	CRUD with Prisma + Express
8	29 Sep 2025	Validation & Error Handling
9	6 Oct 2025	Data Transfer Object (DTO)
10	13 Oct 2025	Filtering Sorting & Pagination
	20-27 Oct 2025	2nd Examination

No.	Date	Topic
11	3 Nov 2025	JWT : Jason Web Tokens
12	10 Nov 2025	Authentication/Authorization
13	17 Nov 2025	API Documentation - OpenAPI/Swagger
14	24 Nov 2025	Integrated Support Topic
	1-12 Dec 2025	3rd Examination

Course outline - G1 (Friday)

No.	Date	Topic
1	8 Aug 2025	Introduction
2	15 Aug 2025	Introduction to RESTful API
3	22 Aug 2025	MVC / Express framework
4	29 Aug 2025	Connection to Database/ORM
5	5 Sep 2025	Relationship Modeling with Prisma
	8 -12 Sep 2025	1st Examination
6	19 Sep 2025	CRUD with Prisma + Express
7	26 Sep 2025	Validation & Error Handling
8	3 Oct 2025	Data Transfer Object (DTO)
9	10 Oct 2025	Filtering Sorting & Pagination
10	17 Oct 2025	JWT : Jason Web Tokens
	20-27 Oct 2025	2nd Examination

No.	Date	Topic
11	31 Oct 2025	Authentication/Authorization
12	7 Nov 2025	API Documentation - OpenAPI/Swagger
13	14 Nov 2025	Integrated Support Topic
14	21 Nov 2025	Integrated Support Topic
	1-12 Dec 2025	3rd Examination

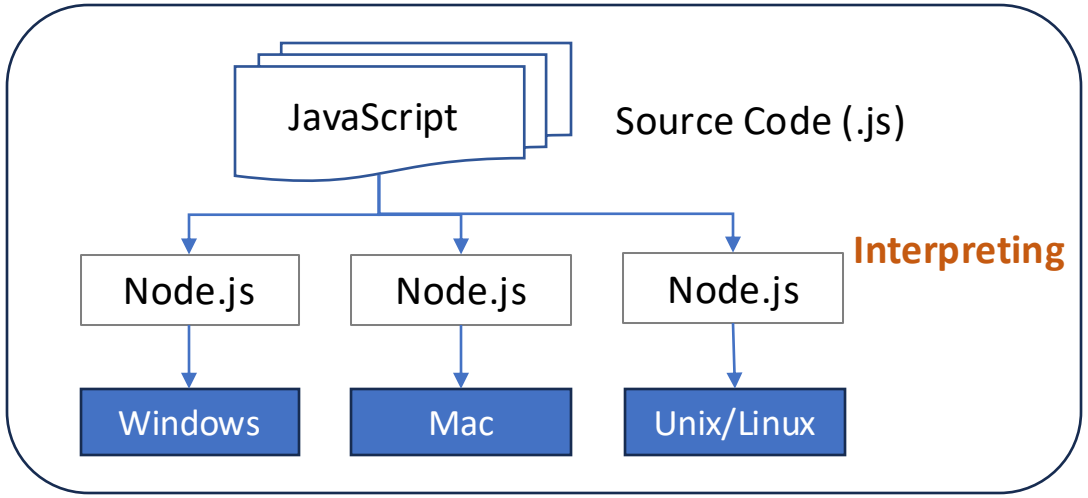
INT 161

Introduction to Node.js

Understanding the Basics of node.js

What is Node.js?

- Node.js is not a programming language
- Node.js is a runtime that converts JavaScript code into machine code.
- Node.js is an open source server environment
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS, etc.)
- Node.js uses **JavaScript** on the server



Why Learn Node.js?

Node.js can be used to full fill multiple purpose like server-side programming, build APIs, etc.

- Node.js is used for server-side programming with JavaScript. Hence, you can use a single programming language (JavaScript) for both front-end and back-end development.
- Node.js **implements asynchronous execution** of tasks in a single thread with async and await technique. This makes Node.js application significantly faster than multi-threaded applications.
- Node.js is being used to build command line applications, web applications, real-time chat applications, REST APIs etc.

How to Install Node.js?

```
# Download and install nvm:
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.2/install.sh | bash

# in lieu of restarting the shell
\. "$HOME/.nvm/nvm.sh"

# Download and install Node.js:
nvm install 22

# Verify the Node.js version:
node -v                # Should print "v22.14.0"
nvm current            # Should print "v22.14.0"

# Verify npm version:
npm -v                 # Should print "10.9.2"
```

The official Node.js website has installation instructions for Node.js: https://nodejs.org

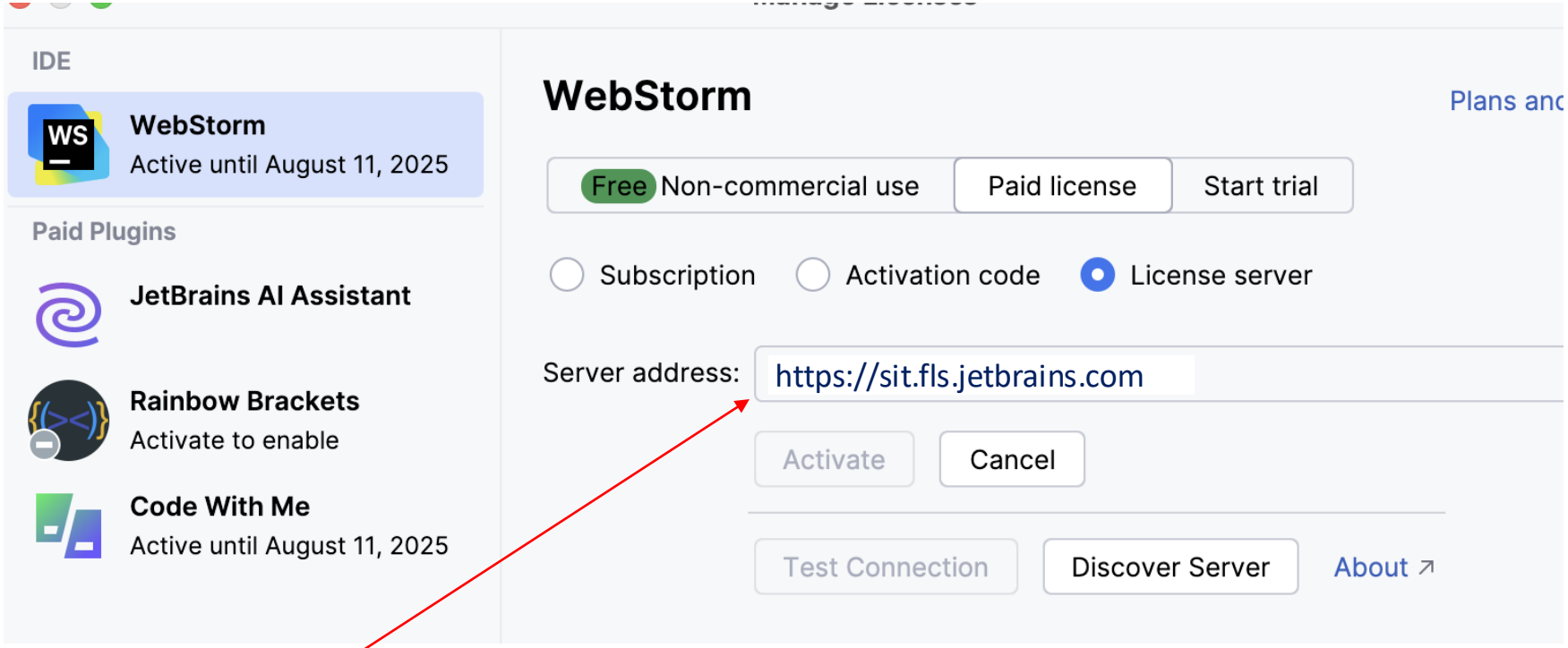
Tools

- WebStorm
  - The JavaScript and TypeScript IDE
- Express
  - MVC Framework for node.js
- Prisma
  - ORM framework for node.js

- Postman
  - Postman is an all-in-one API Platform for building and working with APIs

- Thunder Client:
  - Lightweight Rest API Client Extension for VS Code

JetBrains License

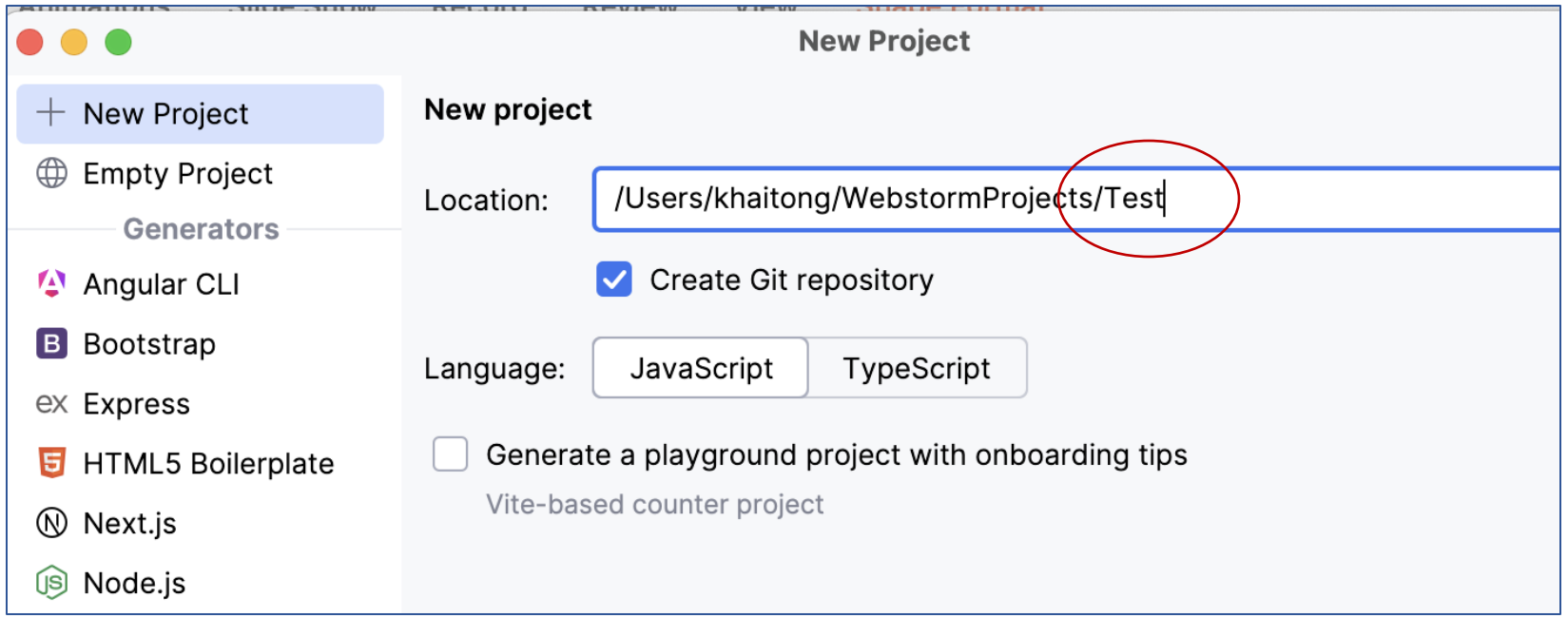


https://sit.fls.jetbrains.com

*** Register Jetbrains account with SIT email first ! ***

Test WebStorm

- Create New Project, Test



Create new JavaScript, MyWebApp

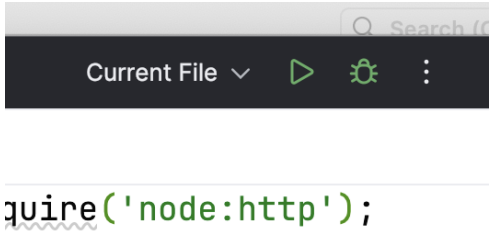
```
const { createServer } = require('node:http');

const hostname = '127.0.0.1';
const port = 3000;

const server = createServer((req, res) => {
   var x = 100;
   res.statusCode = 200;
   res.setHeader('Content-Type', 'text/plain');
   res.write ( 'x = ' + x + '\n');
   x = "Hello World";
   res.write(x);
   res.end();
});

server.listen(port, hostname, () => {
   console.log(`Server running at http://${hostname}:${port}/`);
});
```

1) Right Click on Test

2) Select New -> JavaScript File

3) Enter name: MyWebApp

4) Run



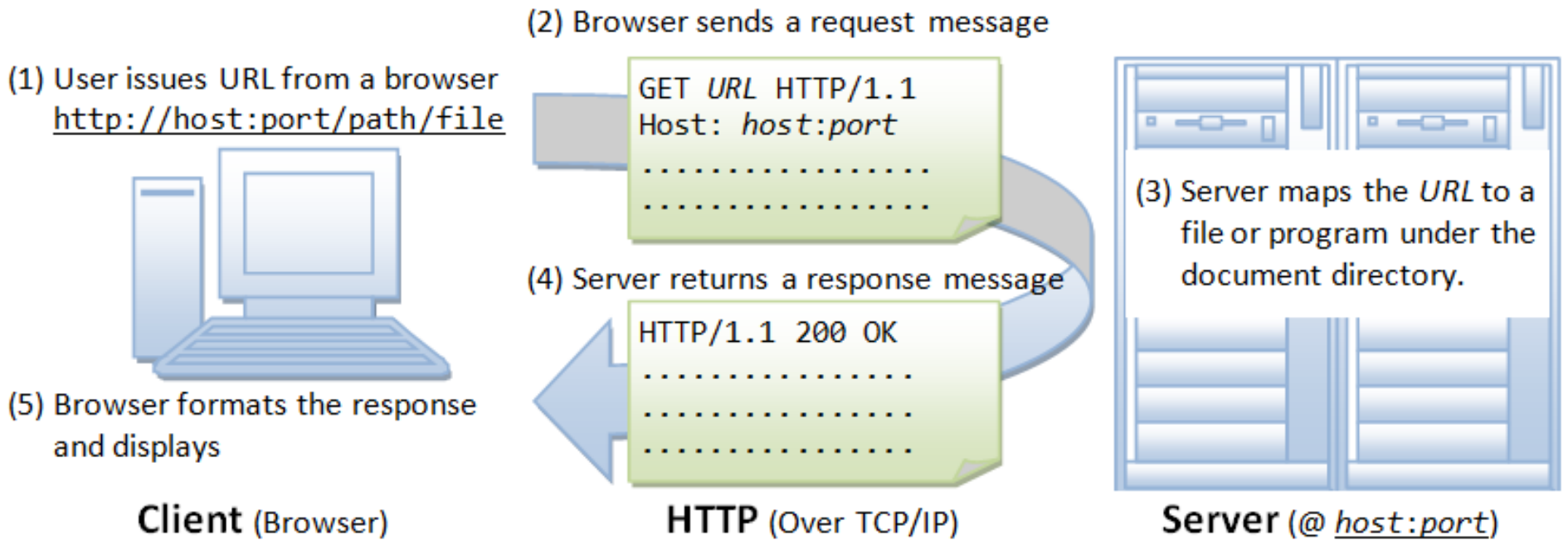
Modify, MyWebApp

```
const { createServer } = require('node:http');
const hostname = '127.0.0.1';
const port = 3000;

const server = createServer((req, res) => {
   var products =   {
      "id": "1",
      "name": "Google Pixel 6 Pro",
      "data": {
         "color": "Cloudy White",
         "capacity": "128 GB"
      }
   }
   res.statusCode = 200;
   res.setHeader('Content-Type', 'application/json; charset=utf-8');
   res.end(JSON.stringify(products));
});

server.listen(port, hostname, () => {
   console.log(`Server running at http://${hostname}:${port}/`);
});
```

World Wide Web Communication

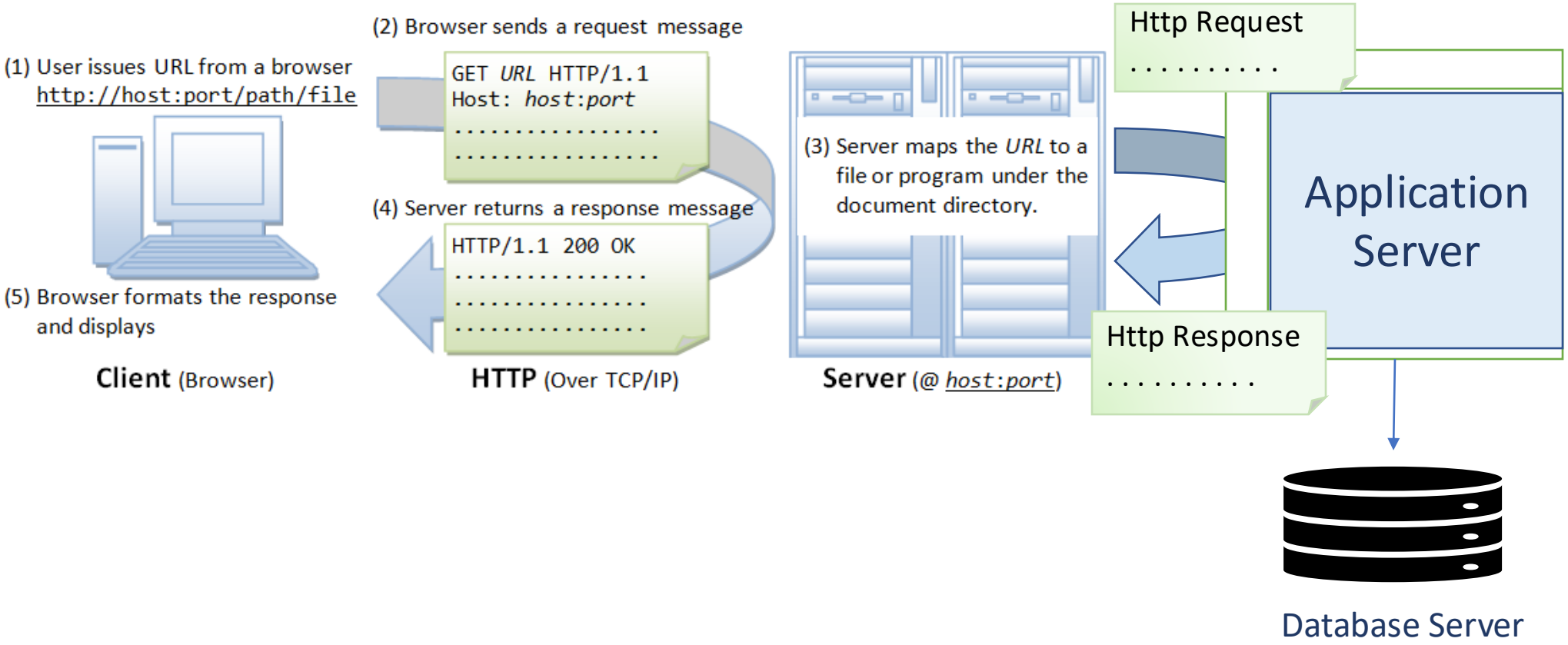
- **HTTP** stands for **H**yper **T**ext **T**ransfer **P**rotocol
- **WWW** is about communication between web **clients** and **servers**
- Communication between client computers and web servers is done by sending **HTTP Requests** and receiving **HTTP Responses**
- **Clients** are often browsers (Chrome, Edge, Safari), but they can be any type of program or device.
- **Servers** are most often computers in the cloud.

HTTP Protocol



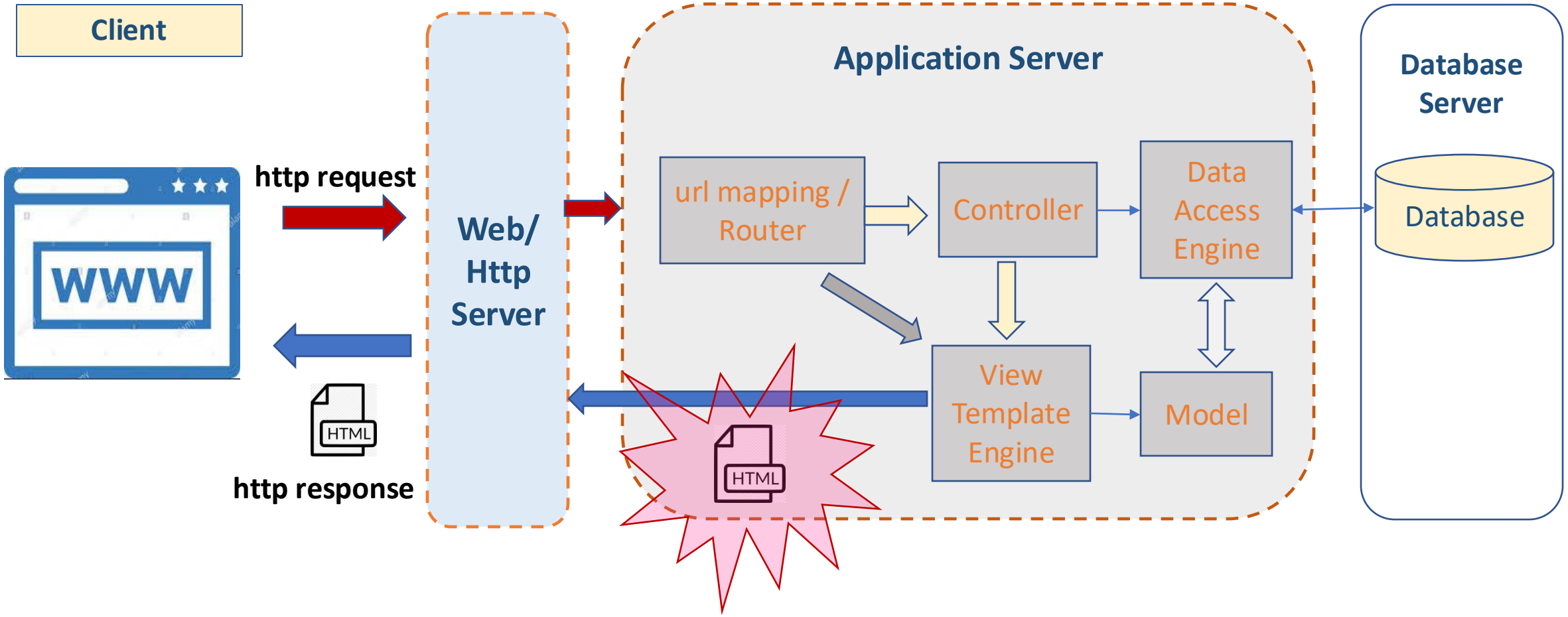
https://www.w3schools.com/whatis/whatis_http.asp

Web Application

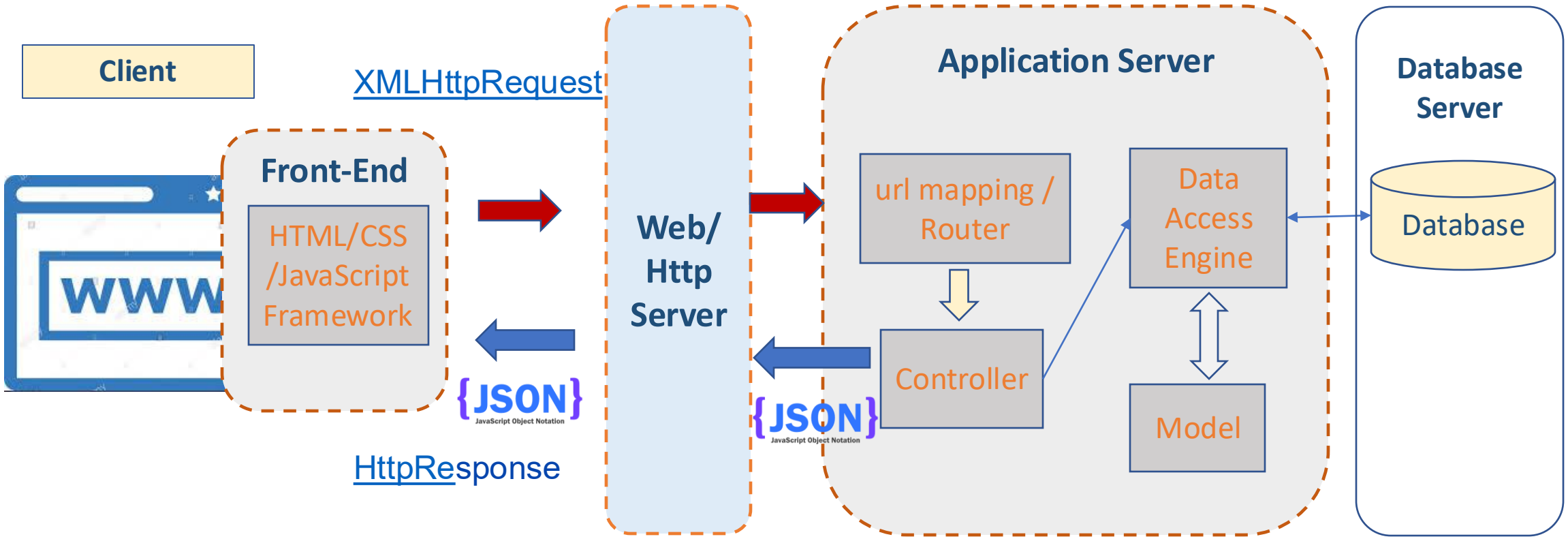


A Multi-Page Application (MPA) Web Application

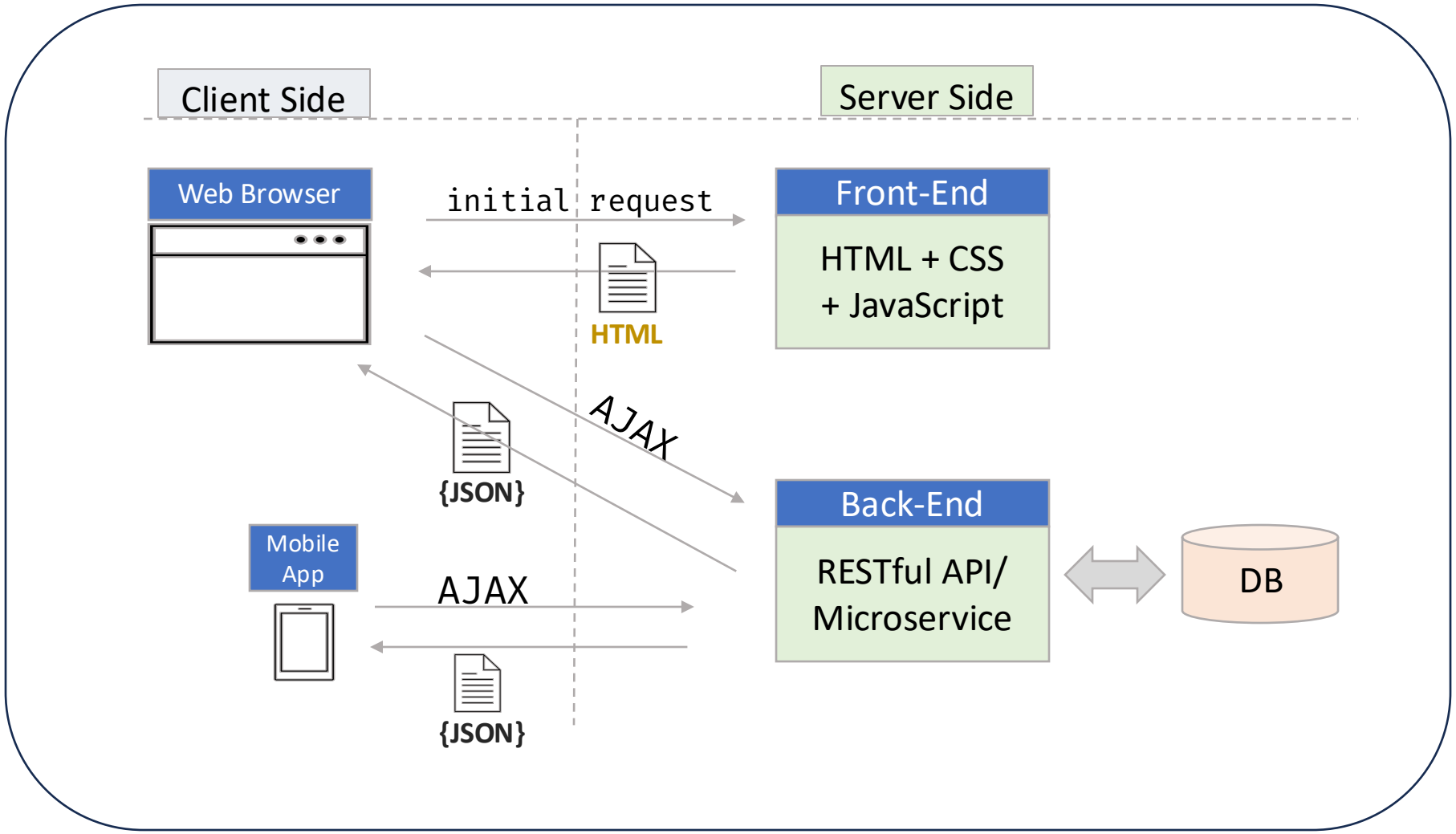
is a traditional web application architecture where each new page or view is loaded from the server with a full page refresh.



MVC Web Application Architectures (modern/spa)



SPA – Single Page Application architectures



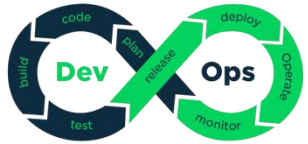
JSON : JavaSrcipt Object Notation

- A lightweight data-interchange format.
- Easy for humans to read and write.
- Easy for machines to parse and generate.
- Based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999.
- JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.

https://www.json.org/json-en.html

https://education.launchcode.org/js-independent-track/chapters/fetch-json/index.html

Full Stack Developer



Client Side

Server Side

{ REST }

NoSQL

SQL

Flutter

React Native

xPlatform
Mobile App.

Classic

WebApp
(Classic)

Microservice

Web
API

Database

Vue.js

React

SPA WebApp.
(JS Framework)

Database

EIS: Enterprise
Info. System

HTML 5

CSS 3

JS

php

laravel

node

ADONIS

python

django

Full Stack Web Developer

- A full stack web developer is a person who can develop both client and server software.
  - Program a browser (like using JavaScript, jQuery, Angular, or Vue)
  - Program a server (like using JavaEE, PHP, ASP, Python, or NodeJS)
  - Program a database (like using SQL, SQLite, or MongoDB)

https://www.w3schools.com/whatis/whatis_fullstack.asp

Web Development Roadmaps (1)

- Front-End Roadmap
  - Every Web Developer must have a basic understanding of HTML, CSS, and JavaScript.
  - Responsive Web Design is used in all types of modern web development.
  - ECMAScript 5 (JavaScript 5) is supported in all modern browsers.
  - On the CSS side you should choose a framework for responsive web design:
    - Bootstrap / Material Design / W3.CSS / Tailwind
  - On the JavaScript side you should learn at least one modern framework:
    - React.js / Angular.js / **Vue.js** / W3.JS

https://www.w3schools.com/whatis/default.asp

Web Development Roadmaps (2)

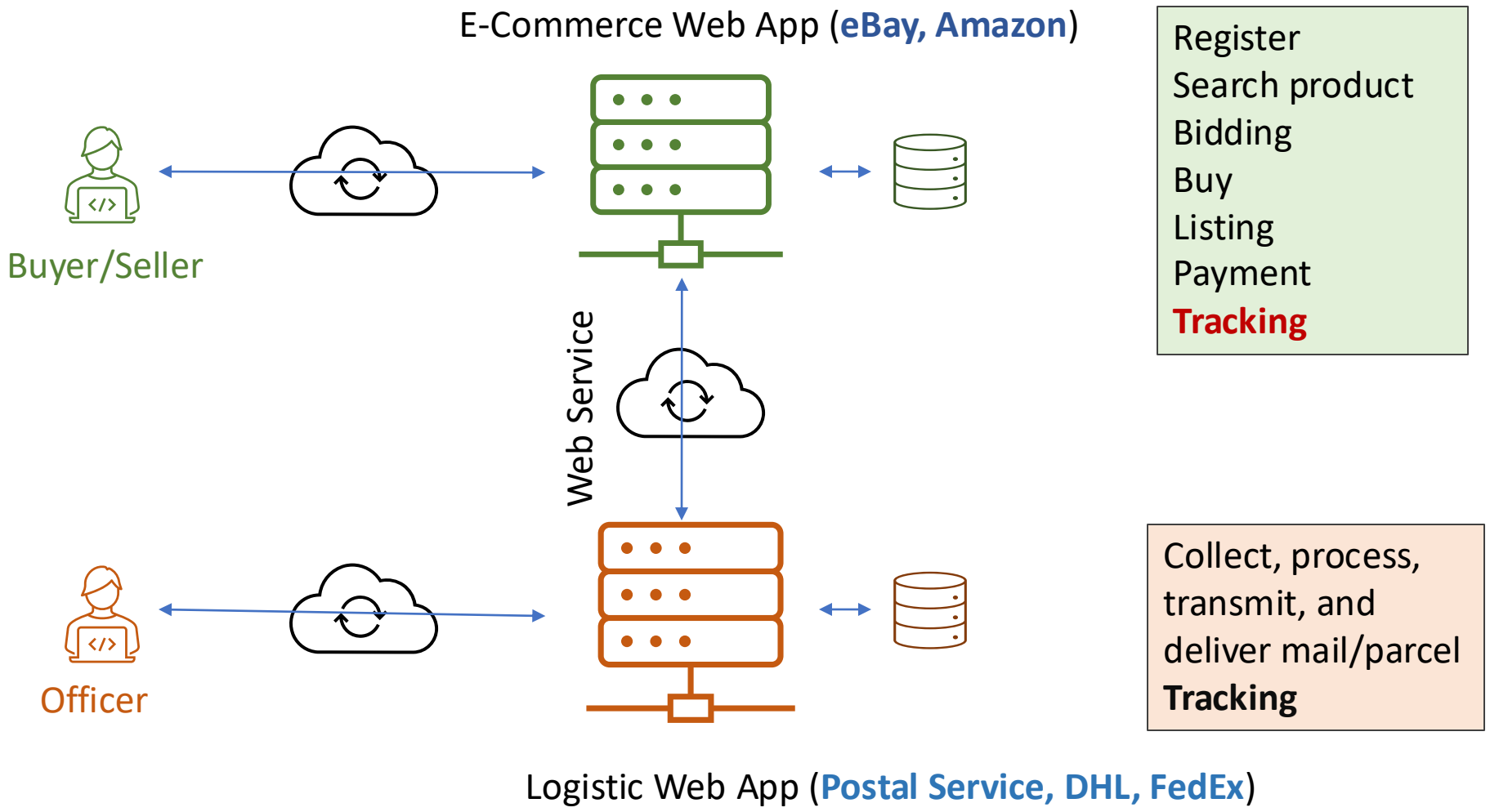
- Back-End Roadmaps
  - SQL / NoSQL
  - Language: Java / PHP / ASP, C# / Python / Node.js
  - Frame work: Spring Boot / Laravel / .NET / Django / Adonis, Express+Prisma
  - Web Service
  - Microservice

https://www.w3schools.com/whatis/whatis_fullstack.asp

Web Services

- Types of Web services are classified into two types
- SOAP Web Services
  - SOAP is an abbreviation for **Simple Object Access Protocol**. It establishes the standard XML format. It also outlines how web services are built. We utilize Web Service Definition Language (WSDL) to describe the request and response XML formats
- RESTful Web Services
  - It was created by Roy Thomas Fielding, who also created HTTP. The primary purpose of RESTful web services is to improve the efficiency of online services. RESTful web services attempt to describe services by utilizing the many principles currently existing in HTTP. REST is a design approach, not a protocol.

Web Service example



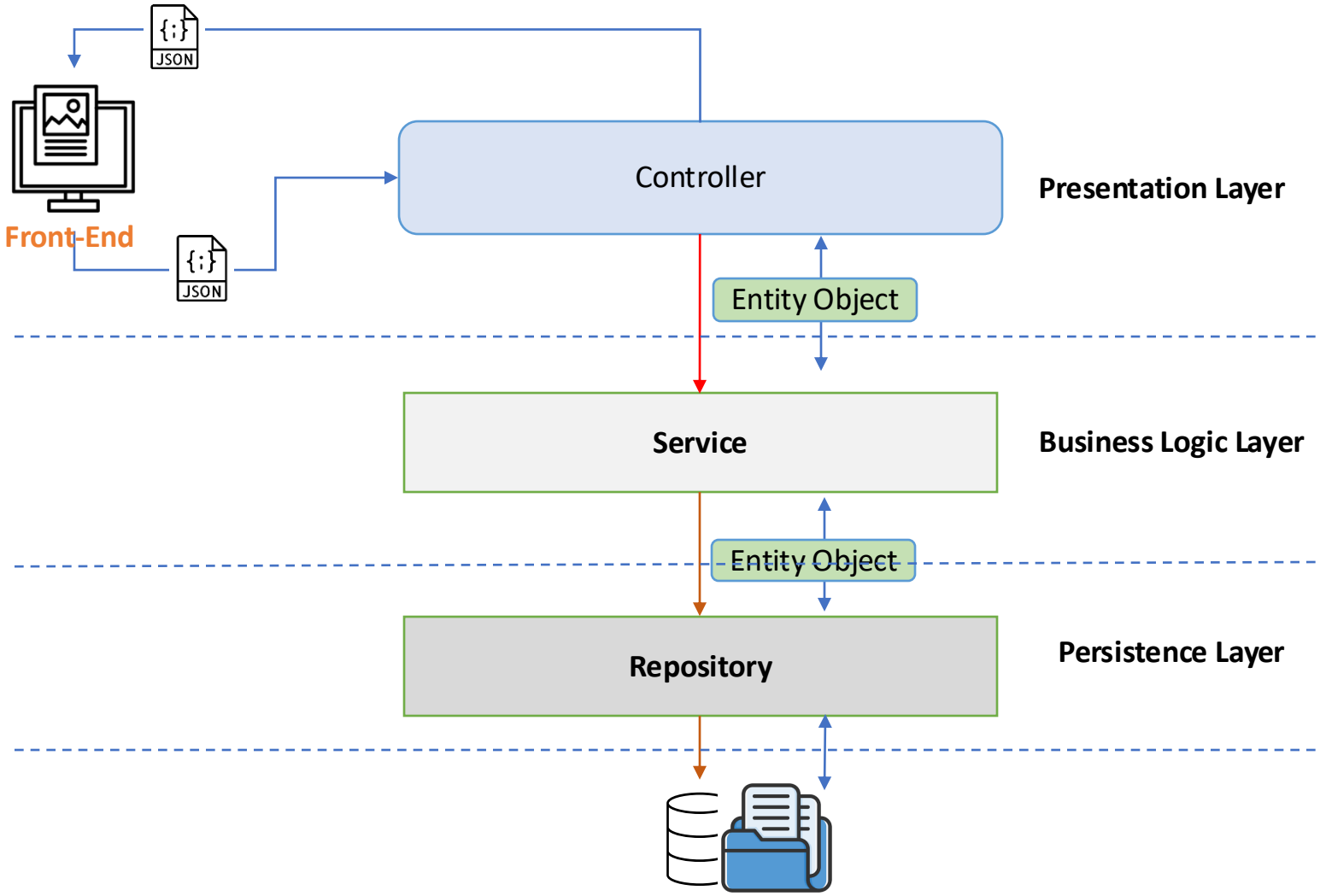
What is REST API

- REST stands for **Re**presentational **S**tate **T**ransfer.
- API: **A**pplication **P**rogramming **I**nterface.
- A REST API allows communication between systems using HTTP requests.
- Key operations: GET, POST, PUT, DELETE.

Core Concepts of REST

- **Statelessness:** No client context is stored on the server.
- **Resource Representation:** Data is represented in formats like JSON or XML.
- **Uniform Interface:** Standardized HTTP methods.
- **Layered System:** Architecture is modular and layered.

Layered System



HTTP Methods

- **Method Purpose Example**
  - GET Retrieve data /users
  - POST Create new resource /users (with data)
  - PUT Update existing data /users/{id} (with data)
  - DELETE Remove a resource /users/{id}

RESTful API Characteristics

- Stateless Communication
- Scalability
- Cacheability
- Client-Server Separation
- Uniform Interface

Example REST API Request

- HTTP Request:

```
GET /api/v1/users/123 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

- Response:

```
{
   "id": 123,
   "name": "John Doe",
   "email": "johndoe@example.com"
}
```

RESTful Resource Uniform Interface

URI	HTTP Verb	Description
api/offices	GET	Get all office
api/offices/1	GET	Get an office with id = 1
api/offices/1/employees	GET	Get all employee for office id = 1
api/offices	POST	Add new office
api/offices/1	PUT	Update an office with id = 1
api/offices/1	DELETE	Delete an office with id = 1

https://restful-api.dev/