데이터 분석 기반
AI 시스템 개발자 양성 과정
02.AI기본프로그래밍(Backend)

조창제

강의자료

2025.08.

목차

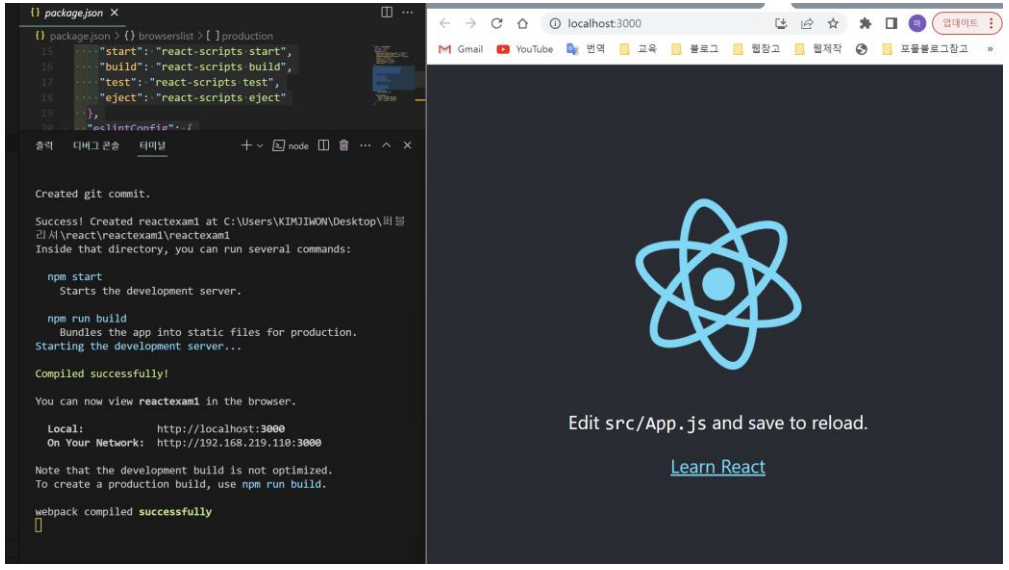
CONTENTS

목차

CONTENTS

1.런타임 환경

1) 정의: 소프트웨어나 프로그램이 실행될 수 있도록 지원하는 시스템 환경
2) 목적: 특정 프로그래밍 언어를 실행하는데 필요한 기능을 제공
3) 종류: Nodejs

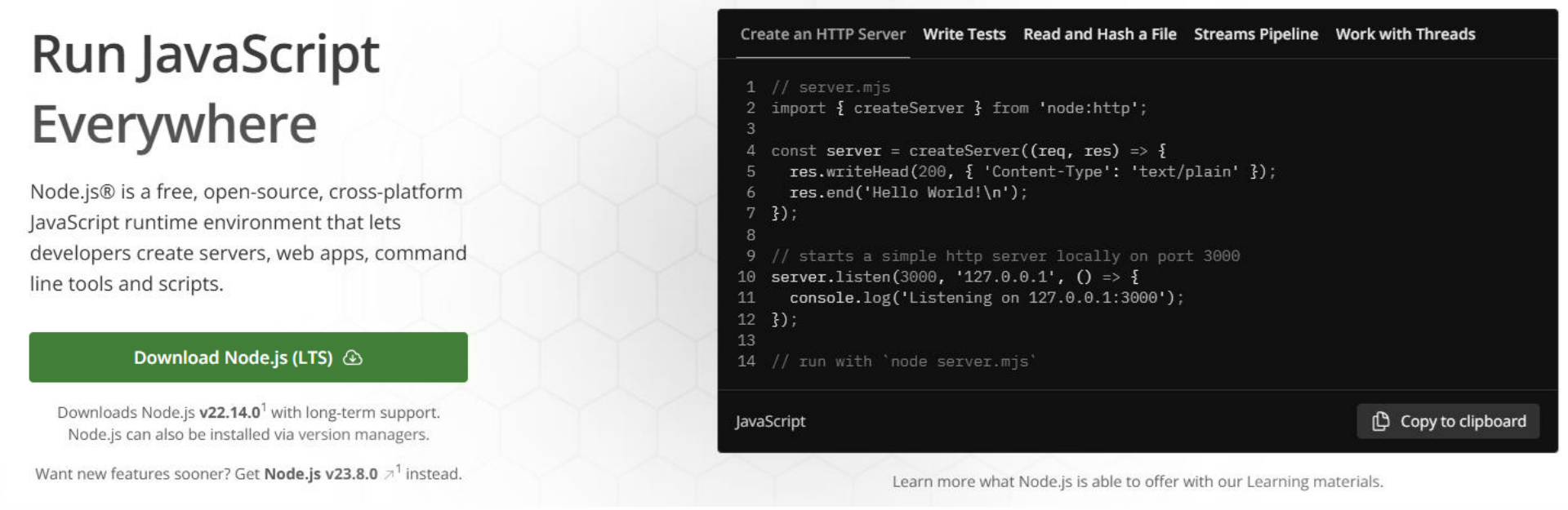


Nodejs 사용 예시

02. 소프트웨어 설치

1.Nodejs 설치

1) 설치링크: Nodejs 링크를 클릭
2) 별다른 설정 없이 설치 진행



01. 개요

1.백엔드

① 웹사이트나 앱에서 **사용자가 직접 보지 않는 서버 쪽**의 작업과 기능들을 처리하는 부분
② 프론트에서 특정 이벤트를 발생시켰을 때, **특정 처리를 수행하는 함수를 구현**하는 곳

1. 고객이 사이트에 접속
2. 사이트에서 HTML 정보를 고객에게 반환
3. ID/PW 입력 후 로그인 버튼 클릭
4. 백엔드 서버로 ID/PW데이터 전송
5. 백엔드 서버는 DB에 데이터 확인
6. 백엔드에서 로그인 처리 후 프론트에게 반환
7. 반환 값을 바탕으로 프론트 갱신
8. 갱신된 HTML 정보 고객에게 전달

02. 구성요소

1.로직 관련

① 로직 구현(CRUD): 데이터를 쓰기/읽기/수정/삭제하는 로직(함수)
② 요청 파라미터: 클라이언트가 데이터를 전송하는 방법(Path, Query, Body)
③ HTTP 메서드: 클라이언트가 요구하는 동작에 대한 표현(POST, GET, PUT, DELETE, PATCH)
④ 오류 처리

2.라우팅

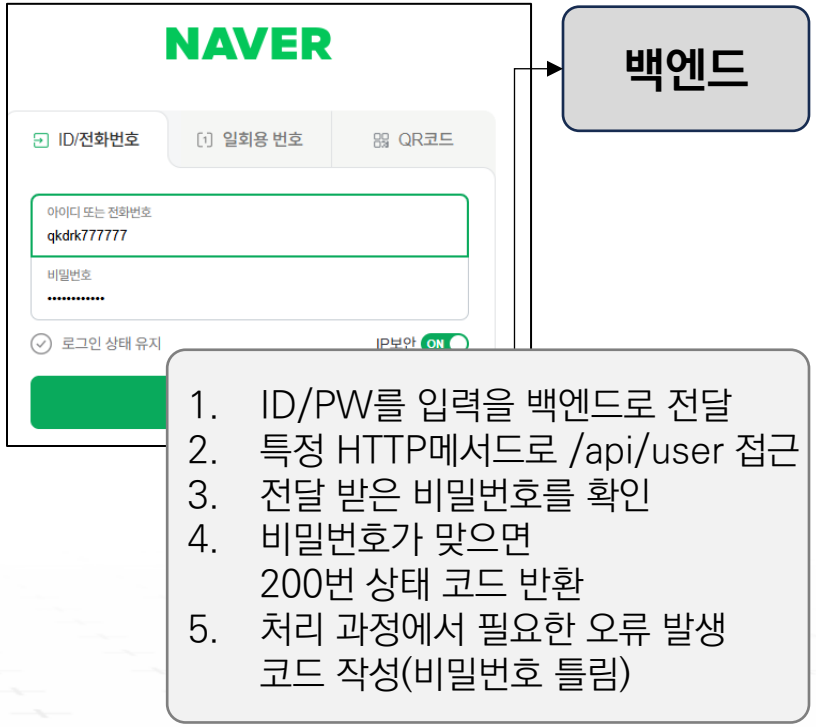
① 클라이언트 요청을 어떤 로직이 처리할지 **서로 연결**해주는 행위
- GET 방식 /api/user 요청이 오면 user 목록을 반환하는 로직과 매칭

3.미들웨어

① 정의: 요청 처리 전/후에 **공통**으로 거치는 로직
- 로그인 체크, 로깅, JSON 파싱, CORS 설정
  - 로깅: 프로그램 실행 중 발생하는 이벤트나 데이터를 기록하는 행위
  - 파싱: 데이터를 구조화된 형태로 가져오는 것

4.인증/인가 관련

① 쿠키, 세션, 토큰



1.요청 파라미터

① 클라이언트가 서버에 **데이터를 전달**하는 다양한 **방법**, 관례상 아래와 같이 사용
② param: 특정 리소스를 가져올 때 주로 활용(GET, PUT, DELETE)
③ query: 검색 필터, 정렬 등의 옵션/조건을 넘길 때 사용(GET)
④ body: POST나 PUT 같은 메서드로 데이터를 HTML/XML 등의 형태로 직접 전송(POST, PUT, PATCH)
- 파일 형태로 주고 받으면, **로그나 히스토리에 잘 남지 않아서 보안상 유리**

이 주소로 보내
(param)

이 주소로
등기(옵션) 보내
(query)

이 주소로
상품(HTML/XML) 보내
(body)

메소드	예시
param	GET /users/123
query	GET /users?age=30&sort=name
body	POST /users Content-Type:application/json

1.HTTP 메서드

① 서버에 **어떤 작업을 요청**하는지 의미하는 기법
② GET: 데이터 조회
③ POST: 데이터 생성
④ PUT: 데이터 전체 수정
⑤ PATCH: 데이터 일부 수정
⑥ DELETE: 데이터 삭제

메소드	예시
POST	진료 접수
GET	진료 기록 조회
PUT	진료 시간 및 담당의 변경
PATCH	진료시간 변경

1.API

① 상태코드

- 클라이언트가 보낸 요청에 서버가 응답한 결과를 숫자로 나타낸 것

상태코드	설명
100번대	정보응답
200번대	201 OK 201 Created 204 No Content
300번대	리다이랙션
400번대	클라이언트오류 400 Bad Request(잘못된 요청) 401 인증필요 403 금지 404 알수없음
500번대	서버오류

링크: 상세 보기

메소드	예시
200	접수 내역 조회
201	방문 접수
204	접수취소

메소드	예시
400	나이 300살, 잘못된 요청
401	신분증 안 챙겨 옴, 인증필요
403	면허 정지 의사, 권한이 없음
404	신입이 실수로 예약을 등록 안함, 알수없음

메소드	예시
500	병원 시스템 고장
503	병원 휴진

04. 더 알아보기

1.RESTful API

1) REST(Representational State Transfer)

① 자원을 이름으로 구분하여 자원의 상태를 주고 받는 것을 의미
- /patients/123

② HTTP 메서드를 통해 행위를 표현 – 명사 중심
- /getPatientList – 안 좋은 예, GET방식으로 /patients

③ 무상태성
- 요청마다 필요한 모든 정보를 담아 서버가 상태를 저장하지 않음 (세션 방식은 RESTful하지 않음)

④ 클라이언트와 서버가 구조적으로 분리
- 클라이언트가 바뀌어도 서버는 사용이 가능, 서로 독립

⑤ 데이터 형식을 Content-Type으로 기재

01. 인증과 인가

1.인증(Authentication)

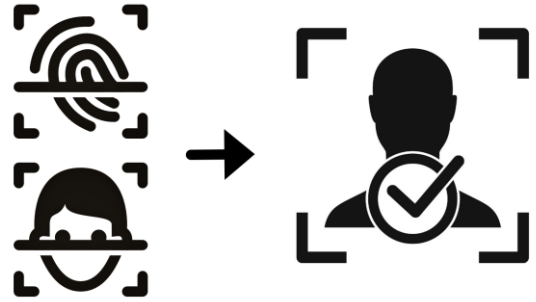
① 사용자의 **신원을 확인**하고 정당한 접근자인지 검증하는 과정

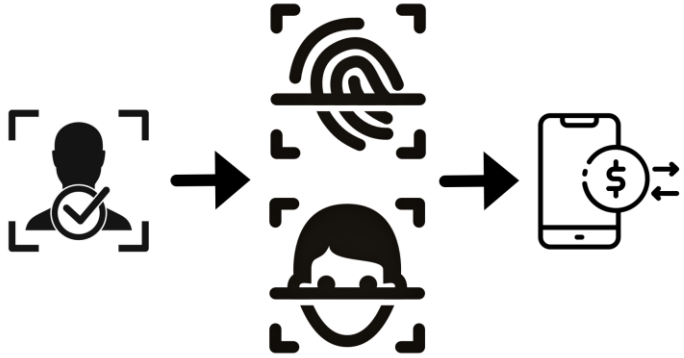
② 인증의 종류

- 비밀번호
- 스마트카드, 신분증, OTP, 공인인증서
- 지문, 홍채, 망막, 얼굴, 손모양, 서명, 목소리

③ 인증의 구현

- JWT(Jason Web Token), SMP/IMAP
- Oauth(다른 인증 제공자에게 위임) – Naver, Kakao, Google
- FIDO(Fast Identity Online) – 지문, Windows Hello 등
  - PASS KEY(FIDO2의 사례)





2.인가(Authorization)

① **인증된 사용자**에게 어떤 리소스나 기능에 접근할 수 있는 **권한이 있는지 결정**하는 보안 절차

[포스트맨 인증 API 사용법]

[포스트맨 파일전송 API 사용법]

1.인증 구현

1) 쿠키(Cookie)

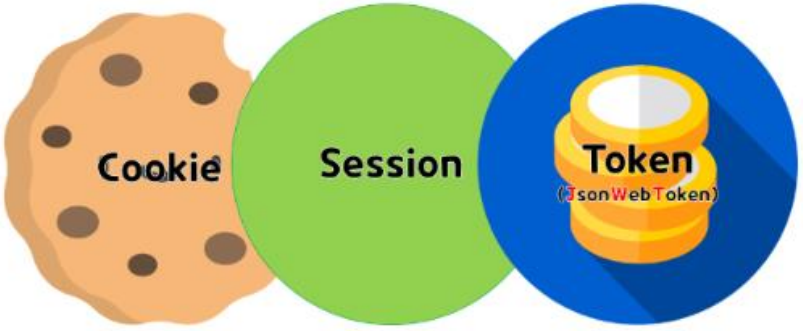
- 클라이언트(브라우저)에 저장되는 작은 데이터 조각

2) 세션(Session)

- 서버에서 사용자 상태를 기억하는 저장소

3) 토큰(Token)

- 클라이언트가 인증 받았음을 증명하는 문자열



02. 더 알아보기

1.해킹 기법

① 물리적 보안
- 케이블 스니핑, USB, 외장하드 등

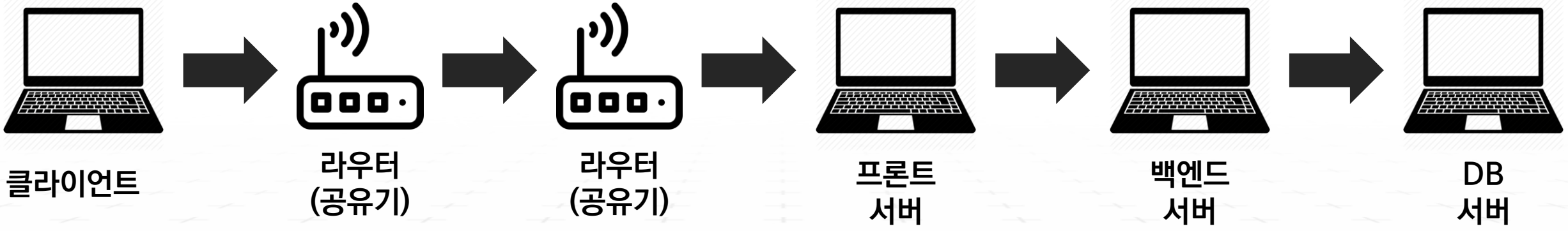
② 네트워크 보안
- 스니핑, 스푸핑, 하이재킹, DoS/DDoS, 랜섬웨어, 트로이목마

③ 애플리케이션 보안
- SQL 인젝션, XSS 방지

2.대응방안

① 망분리, 코드 수정, SSL, CORS 등 다양한 방지 대책 존재

기법	설명
스니핑	데이터(패킷)을 몰래 훔쳐보는 행위
스푸핑	자신을 다른 사람인 척 가장하는 행위
DoS/DDoS	서버에 과도한 부하를 발생
하이재킹	세션을 탈취
케이블 스니핑	랜선을 물리적으로 분해해서 스니핑
SQL 인젝션	사용자가 입력한 값을 검증하지 않고, 그대로 SQL 쿼리에 삽입해서 발생
XSS	악성 스크립트를 통해 쿠키 탈취
백도어	해커가 접근하기 쉽게 접속경로를 만드는 행위



01. 개요

1.비교

1) python 기반

① Django, Flask, FastAPI 등이 존재

2) JavaScript 기반

① Node.js 존재

기준	Django	flask	FastAPI	Nodejs
프레임워크유형	풀스텍 프레임워크	프레임워크	프레임워크	런타임
성능	중간	빠름	매우빠름	매우빠름
비동기지원	제한적	미지원	기본지원	기본지원
문서화	내장된 관리 시스템 제공	작업필요	자동생성	추가 라이브러리 필요
학습난이도	높음	쉬움	보통	쉬움

1.문법

1) MTV디자인 패턴 형태

```
conda create -n backend_39 python==3.9 -y
pip install Django djangorestframework
pip install flask
pip install uvicorn fastapi
```

가상환경 설정

```
django-admin startproject myproject
cd myproject
django-admin startapp myapp
```

환경설정

```python
# myapp/views.py
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
import json

# 기본 엔드포인트 (GET)
def home(request):
    return JsonResponse(
        {"message": "Hello, Django!"})

# GET 요청으로 데이터 제공
def get_data(request):
    return JsonResponse(
        {"message": "Hello, Django!", "status": "success"})

# URL 매개변수 사용 (GET)
def hello(request, name):
    return JsonResponse(
        {"message": f"Hello, {name}!"})

# POST 요청 처리 (데이터 받기)
@csrf_exempt  # CSRF 인증을 제외하기 위해
def handle_post(request):
    if request.method == 'POST':
        data = json.loads(request.body)
        return JsonResponse({"received": data})
```

```python
# myapp/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('api/data', views.get_data, name='get_data'),
    path('hello/<str:name>/',views.hello, name='hello'),
    path('api/post',
         views.handle_post, name='handle_post'),
]
```

```python
# myproject/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.urls')),
    # myapp의 URL 설정 포함
]
```

```
python manage.py runserver 8080
```

03. Flask

1.문법

```python
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, Flask!"

@app.route('/api/data')
def get_data():
    data = {"message": "Hello, Flask!", "status": "success"}
    return jsonify(data)

@app.route('/hello/<name>')
def hello(name):
    return f"Hello, {name}!"

@app.route('/api/post', methods=['POST'])
def handle_post():
    data = request.get_json()  # JSON 데이터 받기
    return jsonify({"received": data})

if __name__ == '__main__':
    app.run(debug=True, host="0.0.0.0", port=8080)
```

04. FastAPI

1.문법

```python
from fastapi import FastAPI, Request
from pydantic import BaseModel
from typing import Dict

app = FastAPI()

@app.get("/")
def home():
    return "Hello, FastAPI!"

@app.get("/api/data")
def get_data():
    return {"message": "Hello, FastAPI!", "status": "success"}

@app.get("/hello/{name}")
def hello(name: str):
    return {"message": f"Hello, {name}!"}

class PostData(BaseModel):
    data: Dict

@app.post("/api/post")
async def handle_post(request: Request):
    data = await request.json()
    return {"received": data}

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8080)
```

04. Node.js

1.문법

1) npm install express body-parser
2) node app.js

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const port = 8080;

// JSON 파싱을 위한 미들웨어 설정
app.use(bodyParser.json());

// 기본 엔드포인트
app.get('/', (req, res) => {
  res.send('Hello, Node.js!');
});

// GET 요청으로 데이터 제공
app.get('/api/data', (req, res) => {
  res.json({ message: 'Hello, Node.js!', status: 'success' });
});
```

```
// URL 매개변수 사용 (예: /hello/{name})
app.get('/hello/:name', (req, res) => {
  const { name } = req.params;
  res.json({ message: `Hello, ${name}!` });
});

// POST 요청 처리
app.post('/api/post', (req, res) => {
  const data = req.body;  // 요청 본문에서 JSON 데이터 추출
  res.json({ received: data });
});

// 서버 실행
app.listen(port, () => {
  console.log(`Server running at http://0.0.0.0:${port}`);
});
```

Thank You

Email: qkdrk777777@naver.com