FE 환경 세팅

환경 세팅 : 프로젝트 시작을 위한 node, 라이브러리 설치 등을 정리한 문서

NVM 없이 설치하기

https://nodejs.org/ko 내 설치파일 설치

NVM으로 Node.js 설치하기

https://velog.io/@februaar/Node.js-윈도우에서-nvm-설치하기

1. **nvm이란**

> 💡 **Node.js Version Manager**
> 여러 버전의 Node.js를 관리하고 전환할 수 있는 유틸리티

node.js는 버전이 빠르게 바뀌어서 동일한 서버 환경 내에서 여러 버전을 사용할 때 버전을 전환하면서 사용할 수 있어 유용하다

2. **설치 (Window)**

https://github.com/coreybutler/nvm-windows/releases

1. 위 링크로 들어가 ***nvm-setup.exe*** 다운로드
2. 약관 동의 및 Next를 통해 설치

명령어

- 특정 버전 node.js 버전 설치하기

```
nvm install v(버전) #nvm install v20.10.0

# nvm 설치 여부 및 버전 확인
nvm --version
```

- 특정 node.js 버전으로 스위칭하기

```
nvm use <버전> #nvm use 20.10.0

#default 버전 설정
nvm alias default <버전> #nvm alias default 20.10.0
```

- 기타 명령어

```
#설치된 node.js 목록 확인
nvm ls

#필요없는 node 버전 삭제
nvm uninstall <버전>
```

패키지 관리 툴 설치

NPM vs YARN

- 패키지 설치 측면에서 yarn이 npm보다 빠르다
- 보안 측면에서 yarn이 npm보다 더 안전하다

- 사용자 측면에서 npm이 yarn보다 더 많은 사용자를 보유하고 있다
- 둘 다 지속적으로 관리 및 업데이트 되고 있기 때문에 개인의 취향에 따라 선택 가능하다

npm

- Node.js 기본 패키지 관리자
- 노드를 다운로드하면 자동으로 설치된다
- 기본 명령어

```
#package.json 생성
npm init

#package.json 파일 내 종속성 모듈 설치
npm install

#설치한 패키지 업데이트
npm update

#패키지 삭제
npm uninstall
```

yarn

- 페이스북에서 만든 JS 패키지 매니저
- npm을 통해 설치
- npm의 단점인 **속도, 안정성, 보안성** 등을 향상시켰다

```
#yarn 설치
npm install yarn --global

#package.json 생성
yarn init

#package.json 파일 내 종속성 모듈 설치
yarn install

#설치한 패키지 업데이트
yarn upgrade

#패키지 삭제
yarn remove
```

React + Vite

프로젝트 설치 (프로젝트 시작할 때 하면 됩니다)

```
#1. 프로젝트 생성
npm create vite@latest --template react-ts
--
yarn create vite <프로젝트명> --template react-ts

#2. 해당 프로젝트로 이동
cd <프로젝트명>

#3. 종속 모듈 설치
npm install
--
yarn

#4. 실행
npm run dev
--
yarn dev
```

ESLint

- 코드의 오류를 찾고 수정하기 위해 사용
- JS 코드를 검사해서 잘못되거나 개선할 부분을 알려준다
- vite로 프로젝트 생성 시, 자동으로 설치된다

Style-Lint

- 필요하면 각자 추가

Prettier

https://issell.tistory.com/entry/VSCode-프로젝트에-prettier-사용하기

- 코드의 형식을 일관되게 하기 위해 사용

Prettier와 ESLint 설정

참고

https://tyoon9781.tistory.com/entry/vscode-React-Prettier-ESLint-setting

설치

```
npm install -D prettier eslint-plugin-prettier eslint-config-prettier
```

- eslint-config-prettier: 린트 위에 사용할 프리티어 플러그인
- eslint-plugin-prettier: 린트 설정과 중복되는 부분이 있으면 프리티어 룰에서 제외하는 플러그인

이후 과정은 맨 아래 **"프로젝트 생성 단계 > prettier & ESLint 설정"** 부분을 참고하여 차례대로 진행

이외 라이브러리 설치

axios

```
yarn add axios
```

react-router-dom

```
yarn add react-router-dom
```

@types/node

TypeScript에서 Node.js 모듈을 쓸 수 있도록 도와주는 환경 구축

```
yarn add -D @types/node
```

tailwind css

```
yarn add -D tailwindcss postcss autoprefixer
```

▼ tailwind 초기화

```
yarn tailwind init -p
```

- `tailwind.config.js` 내에 다음 내용

```
/** @type {import('tailwindcss').Config} */
export default {
  content: ['./index.html', './src/**/*.{js,jsx,ts,tsx}'],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

- `index.css` 내 내용 다 삭제 후, 다음 내용 추가

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

react query

```
yarn add @types/react-query
```

zustand

```
yarn add zustand
```

---

env 환경 파일

- 포트, DB 관련 정보, API_KEY 등 **git에 올라가면 안되는 값들을 저장**
- .gitignore에도 .env 반드시 포함 !!

설정

프로젝트 최상위 루트에 `.env` 파일을 생성

내부에 값 설정

```
#예시 : REACT_APP_변수명 = 값
REACT_APP_DATABASE = *******
```

사용법

```
const key = process.env.REACT_APP_변수명
```

작업 순서

1. 깃 클론

```
git clone <레포주소>
```

2. dev 브랜치 pull

```
git fetch
git pull origin dev
```

- 항상 작업하기 전에, dev 브랜치에 새로 업데이트된 내용 있는지 확인할 것

3. 로컬 브랜치 생성

```
git checkout -b <브랜치명>
```

- 브랜치명은 깃 브랜치 네이밍 규칙을 따를 것

4. 개발

- 본인의 로컬 브랜치에서 작업
- 항상 브랜치 확인하기

5. 개발 완료 시 커밋 생성

```
git commit

# 위 명령어 사용 시, 새로운 화면이 뜸
# i 를 입력해 입력모드로 전환 후 개발 내용 작성
# 예시
feat: 로그인 기능 생성
- 설명 1
- 설명 2
- ...
```

6. 커밋 생성 후 push

```
git push origin <작업한브랜치명>
```

7. git 사이트로 이동 후 PR (Pull Request) 생성

8. 코드 리뷰 및 수정 후 dev 브랜치에 merge

⭐ 주요 명령어 정리

프로젝트 이동

```
cd <프로젝트명>
```

프로젝트 생성

```
npm create vite@latest
```

프로젝트 패키지 설치

```
npm install
```

개발모드 실행

```
npm run dev
```

실행 중인 서버 종료

```
ctrl + c
```

개발된 프로젝트 빌드

```
npm rum build
```

⭐ 프로젝트 생성 단계 & 명령어 정리

프로젝트 생성

```
yarn create vite Frontend --template react-ts
cd Frontend
yarn set version berry
yarn
```

- `node_modules` 폴더 전체 제거
- `.yarnrc.yml` 내 `nodeLinker: node_modules` 제거

```
//이 코드만 남는다
yarnPath: .yarn/releases/yarn-4.6.0.cjs
nodeLinker: pnp
```

- 터미널에서 다시 `yarn` 입력

필요한 툴 설치

```
yarn add -D prettier eslint-plugin-react eslint-plugin-prettier eslint-config-pr
yarn add axios
yarn add react-router-dom
yarn add -D @types/node
yarn add @types/react-query
yarn add zustand
yarn add -D tailwindcss postcss autoprefixer
```

prettier & ESLint 설정

1. vscode extension에서 prettier 및 ESLint 설치
2. `root` 디렉토리에 `.prettierrc` 파일을 생성해준다
   - 이 파일이 없으면 기본값으로 세팅됨
   - 프로젝트에 맞게 내용 변경

```
{
  "arrowParens": "always",
  "bracketSameLine": false,
  "bracketSpacing": true,
```

```
  "embeddedLanguageFormatting": "auto",
  "htmlWhitespaceSensitivity": "css",
  "insertPragma": false,
  "jsxSingleQuote": true,
  "printWidth": 80,
  "proseWrap": "always",
  "quoteProps": "as-needed",
  "requirePragma": false,
  "semi": true,
  "singleAttributePerLine": false,
  "singleQuote": true,
  "tabWidth": 2,
  "trailingComma": "es5",
  "useTabs": false,
  "vueIndentScriptAndStyle": false
}
```

4. eslint 설정

https://tyoon9781.tistory.com/entry/vscode-React-Prettier-ESLint-setting

```
yarn eslint --init
```

명령어 입력 후 아래와 같이 설정

- problems : To check syntax and find problems
- esm : Javascript modules (import/export)
- 설치는 yarn으로

```
You can also run this command directly using 'npm init @eslint/config@latest'.
@eslint/create-config: v1.4.0

√ How would you like to use ESLint? · problems
√ What type of modules does your project use? · esm
√ Which framework does your project use? · react
√ Does your project use TypeScript? · typescript
√ Where does your code run? · browser
The config that you've selected requires the following dependencies:
```

5. `eslint.config.js` 를 다음과 같이 설정

```
import globals from 'globals';
import pluginJs from '@eslint/js';
```

```
import tseslint from '@typescript-eslint/eslint-plugin';
import tsParser from '@typescript-eslint/parser';
import pluginReact from 'eslint-plugin-react';
import pluginPrettier from 'eslint-plugin-prettier';
import configPrettier from 'eslint-config-prettier';

/** @type {import('eslint').Linter.Config[]} */
export default [
  {
    files: ['**/*.{js,mjs,cjs,ts,jsx,tsx}'], // 검사 대상 파일
    languageOptions: {
      globals: globals.browser,
      parser: tsParser,
      parserOptions: {
        ecmaVersion: 'latest',
        sourceType: 'module',
        ecmaFeatures: {
          jsx: true,
        },
      },
    },
    plugins: {
      '@typescript-eslint': tseslint,
      react: pluginReact,
      prettier: pluginPrettier,
    },
    rules: {
      ...pluginJs.configs.recommended.rules, // 기본 JS 규칙
      ...tseslint.configs.recommended.rules, // TypeScript 규칙
      ...pluginReact.configs.flat.recommended.rules, // React 규칙
      'prettier/prettier': [
        'error',
        {
          endOfLine: 'auto',
        },
      ], // Prettier 규칙을 ESLint에서 에러로 처리
      'react/react-in-jsx-scope': 'off', // React 17+에서 불필요한 규칙 비활성화
    },
  },
  {
    rules: {
      ...configPrettier.rules, // Prettier와 충돌하는 규칙 비활성화
    },
  },
  {
```

```
    ignores: ['node_modules', 'dist'], // 무시할 디렉토리
  },
];
```

6. yarn 과 연동

```
yarn dlx @yarnpkg/sdks vscode
```

- 설치 후, `Ctrl + Shift + P` 누른 후, `workspace version typescript` enter
- 그 후, `use workspace version` 선택

7. vscode 내에서 `Ctrl + ,` 누르고, 오른쪽 위 파일 부분(아래 이미지 참고)을 누른 다음 아래 json 코드 삽입

```
{
  "workbench.iconTheme": "material-icon-theme",
  "window.newWindowProfile": "Default",
  "terminal.integrated.defaultProfile.windows": "Command Prompt",
  "editor.formatOnSave": true,
  "[javascript]": {
    "editor.formatOnSave": true
  },
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": "explicit"
  },
  "[typescriptreact]": {
    "editor.defaultFormatter": "esbenp.prettier-vscode"
  },
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "typescript.tsdk": ".yarn/sdks/typescript/lib",
  "eslint.nodePath": ".yarn/sdks",
  "eslint.validate": ["javascript", "typescript", "typescriptreact", "json"]
}
```