

프로젝트

≡ 작성자	채런 🐼
≡ 분류	설계
☀ 상태	시작안함

▼ 목차

목차

 Environment

 폴더구조

 .storybook

 main.ts

 preview.ts

 public

 assets


 assets/fonts

 assets/images

 assets/images/pwa

 assets/svg

 src

 assets

 components

 hooks

 pages

 atoms

 stories

 styles

 types

 Router.tsx

 index.tsx

 react-app-env.d.ts

 .env

 .env.development

 .env.production

 .gitignore

 .prettierrc.js

 package-lock.json

 package.json

 tsconfig.json

Environment

스택	툴 설명	버전
환경		
node.js	JavaScript 런타임	18.16.1
npm	node package manager	9.5.1
CSR, 타입 라이브러리		
React	자바스크립트 라이브러리	18.2.0
TypeScript	자바스크립트 슈퍼셋, 오픈소스 프로 그래밍 언어	4.9.5
전역 상태 관리		
Recoil	전역 상태 관리 라이브러리	0.7.7
recoil-persist	클라이언트 로컬에 저장해주는 라이 브러리	5.1.0

스택	툴 설명	버전
React query	비동기 데이터 관리 라이브러리	3.39.3
css 라이브러리		
emotion		11.11.1
emotion/styled		11.11.0
애니메이션 라이브러리		
Framer Motion		10.15.0
코드 포매팅		
Prettier		3.0.0
HTTP 비동기 통신 라이브러리		
axios	HTTP 비동기 통신 라이브러리	1.4.0
라우터		
react-router-dom	라우팅 라이브러리	6.14.2
테스팅 라이브러리		
storybook	컴포넌트 테스트 라이브러리	^7.4.1

폴더구조

.storybook

참고 <https://velog.io/@juno7803/Storybook-Storybook-200-활용하기>

main.ts

- storybook을 위한 config 설정들
- `npm sb init` 을 통해서 기본으로 설정되는 **stories**와 **addons** 세팅과 (stories에서 story 파일이 프로젝트 내 어디에 어떤 형식의 파일이 위치하는지를 명시해 주어야 storybook 실행 시 정상적으로 불러올 수 있습니다.)

```
import type { StorybookConfig } from '@storybook/react-webpack5';

const config: StorybookConfig = {
  stories: ['../src/**/*.mdx', '../src/**/*.stories.@(js|jsx|mjs|ts|tsx)'],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@storybook/preset-create-react-app',
    '@storybook/addon-onboarding',
    '@storybook/addon-interactions',
  ],
  framework: {
    name: '@storybook/react-webpack5',
    options: {},
  },
  docs: {
    autodocs: 'tag',
  },
  staticDirs: ['../public'],
};
export default config;
```

preview.ts

- 해당 프로젝트의 모든 Story에 global하게 적용될 포맷을 세팅하는 곳.

```
import type { Preview } from '@storybook/react';

const preview: Preview = {
  parameters: {
    actions: { argTypesRegex: '^on[A-Z].*' },
    controls: {
      matchers: {
        color: /(background|color)$/i,
```

```
    date: /Date$/,
  },
},
},
};

export default preview;
```

public

```
public
├── assets
│   ├── fonts
│   ├── images
│   │   └── pwa
│   │       ├── favicon.ico
│   │       ├── logo192.png
│   │       └── logo512.png
│   └── svgs
├── index.html
└── manifest.json
```

assets

- 코드 이외에 정적인 리소스를 관리한다(이미지파일, 폰트, svg파일들 등..)

assets/fonts

- 폰트 관련 자산 관리 폴더

assets/images

- 이미지 자산 관리폴더
- 추후 이미지를 추가한다면 관련있는 이미지들끼리 묶어서 관리하도록 함

assets/images/pwa

- pwa 관련 이미지 자산관리 폴더

assets/svgs

- svgs 관련 자산 관리폴더

src

```
src
├── assets
│   ├── images
│   └── svgs
├── components
│   ├── features
│   │   └── Quiz
│   │       ├── QuestionBox
│   │       │   ├── QuestionBox.styled.ts
│   │       └── QuestionBox.tsx
│   └── utils
├── hooks
├── pages
├── atoms
└── stories
```

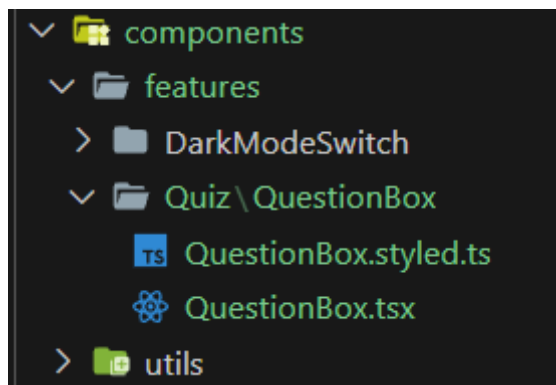
```
| └─ 📄 QuestionBox.stories.ts
├─ 📁 styles
├─ 📁 types
├─ 📄 Router.tsx
├─ 📄 index.tsx
└─ 📄 react-app-env.d.ts
```

📁 assets

- font, images, svgs 파일들 보관

📁 components

- 컴포넌트 파일들 보관
- 이름 규칙 : 파스칼케이스 (첫글자 대문자로 작성!)
- 📁 **utils** : 여러군데서 사용되는 컴포넌트 (예를들어 nav, button 등등..)
- 📁 **features** : 기능단위의 컴포넌트 (예를들어 camera 등등..)
- 컴포넌트 작성 예시



- 만들 컴포넌트 이름으로 폴더 생성
- 부모 자식 컴포넌트라고 생각된다면 한 폴더에 작성 or 폴더 내부에서 하위폴더 생성해서 관리

📁 hooks

- 커스텀 훅 보관 폴더
- 📁 **apis**
 - 서버와 통신할 때 관련된 커스텀 훅은 이 폴더에서 관리
- 📁 **custom**
 - 반복되는 함수를 커스텀 훅으로 만들때는 이 폴더에서 관리

📁 pages

- 컴포넌트들로 만들 페이지 관리
- 작은 컴포넌트 단위들 말고 페이지 단위도 생성한다(메인페이지, 게임페이지 등등..)

📁 atoms

- 전역 상태 관리 파일 보관

📁 stories

- 스토리 파일 보관
- 스토리 파일들도 컴포넌트 이름과 동일하게 작성

📁 styles

- 베이스 css 코드 보관 ex) reset.css, base.css

📁 types

- 타입 관리

Router.tsx

- 라우팅 관리

index.tsx

- 최상위 엔트리(진입점) 컴포넌트 렌더링 파일
 - Main컴포넌트 한개만 렌더링한다

react-app-env.d.ts

- 환경변수 타입 입력 파일

.env

- 환경변수 관리 파일
- 코드로 직접 노출되면 안되는 api 키 같은 값들을 관리한다
- npm run build, npm run start 했을 때 모두 실행된다

```
// 사용예시
REACT_APP_TOKEN=토큰번호
```

.env.development

- 개발용 환경변수 관리 파일
- 개발할때 사용되는 임시 api키 값들을 관리한다
- npm run start 했을 때 실행된다

.env.production

- 프로덕션용 환경변수 관리 파일
- 실제로 사용되는 api 키 값을 관리한다
- npm run build 했을 때 실행된다

.gitignore

- git hub에 안올려도 되는 파일들 관리
- node modules, env 파일 등등

.prettierrc.js

- 프리티어 설정파일

```
module.exports = {
  // 화살표 함수 식 매개변수 () 생략 여부 (ex: (a) => a)
  arrowParens: 'always',
  // 닫는 괄호(>) 위치 설정
  // ex: <div
  //       id="unique-id"
  //       class="contaienr"
  //     >
  htmlWhitespaceSensitivity: 'css',
  bracketSameLine: false,
  // 객체 표기 괄호 사이 공백 추가 여부 (ex: { foo: bar })
  bracketSpacing: true,
  // 행폭 설정 (줄 길이가 설정 값보다 길어지면 자동 개행)
```

```

printWidth: 80,
// 산문 래핑 설정
proseWrap: 'preserve',
// 객체 속성 key 값에 인용 부호 사용 여부 (ex: { 'key': 'xkieo-xxxx' })
quoteProps: 'as-needed',
// 세미콜론(;) 사용 여부
semi: true,
// 싱글 인용 부호(') 사용 여부
singleQuote: true,
// 탭 너비 설정
tabWidth: 2,
// 객체 마지막 속성 선언 뒤 부분에 콤마 추가 여부
trailingComma: 'es5',
// 탭 사용 여부
useTabs: false,
};

```

package-lock.json

- 의존성 버전 정보를 고정시키기 위해 필요한 것
- npm 패키지 매니저에서 node_modules 디렉토리에 설치된 패키지들의 의존성 트리를 기록하는 파일
- npm을 사용해서 node_modules 트리나 package.json 파일을 수정하게 되면 자동으로 생성되는데, 이 파일은 파일 생성 시점의 의존성 트리에 대한 정보를 가지게 된다. 따라서 package.json의 버전 범위에 따라 버전이 바뀌더라도, package-lock.json 파일이 작성된 시점의 의존성 트리가 다시 생성될 수 있게끔 보장할 수 있게 되는 것이다.

package.json

```

{
  "name": "endorfin",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@mui/icons-material": "^5.14.0",
    "@mui/material": "^5.14.2",
    "@testing-library/jest-dom": "5.17.0",
    "@testing-library/react": "13.4.0",
    "@testing-library/user-event": "13.5.0",
    "@types/jest": "27.5.2",
    "@types/node": "16.18.50",
    "@types/react": "18.2.21",
    "@types/react-dom": "18.2.7",
    "@types/react-router-dom": "^5.3.3",
    "emotion-normalize": "^11.0.1",
    "normalize.css": "^8.0.1",
    "axios": "1.4.0",
    "framer-motion": "10.15.0",
    "react": "18.2.0",
    "react-dom": "18.2.0",
    "react-query": "3.39.3",
    "react-router-dom": "6.14.2",
    "react-scripts": "5.0.1",
    "react-styled-toggle": "^1.1.0",
    "recoil": "0.7.7",
    "recoil-persist": "5.1.0",
    "typescript": "4.9.5",
    "web-vitals": "2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject",
    "storybook": "storybook dev -p 6006",
    "build-storybook": "storybook build"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest",
      "plugin:storybook/recommended"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
  },

```

```

    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
    "@emotion/react": "11.11.1",
    "@emotion/styled": "11.11.0",
    "@storybook/addon-essentials": "^7.4.1",
    "@storybook/addon-interactions": "^7.4.1",
    "@storybook/addon-links": "^7.4.1",
    "@storybook/addon-onboarding": "^1.0.8",
    "@storybook/blocks": "^7.4.1",
    "@storybook/preset-create-react-app": "^7.4.1",
    "@storybook/react": "^7.4.1",
    "@storybook/react-webpack5": "^7.4.1",
    "@storybook/testing-library": "^0.2.0",
    "babel-plugin-named-exports-order": "^0.0.2",
    "eslint-plugin-storybook": "^0.6.13",
    "prettier": "3.0.0",
    "prop-types": "15.8.1",
    "storybook": "^7.4.1",
    "webpack": "^5.88.2"
  }
}

```

tsconfig.json

```

{
  "compilerOptions": {
    "target": "es6",
    "lib": ["dom", "dom.iterable", "esnext", "es6"],
    "allowJs": true,
    "skipLibCheck": true,
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "noFallthroughCasesInSwitch": true,
    "module": "esnext",
    "moduleResolution": "node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react-jsx",
    "baseUrl": "src",
  },
  "include": ["src"]
}

```

