Serverless React SPA 스터디

Orientation

멘토 소개

박지원

park@jiwon.me

- 정보시스템학과 20학번
- (주)스케치소프트 SW 엔지니어
    - (겨울인턴 환영! 네이버/카카오만큼 챙겨줍니다)
- 페이스북 페이지 유사수학 탐지기
- 한국 TEX 학회

자기소개!

스터디의 목표

최대한 빠르게 MVP(Minimum Viable Product)를 만들어내자!

1. Typescript React 프로젝트

2. netlify / heroku / firebase를 이용해 배포하기

3. GitHub CI를 이용한 CI/CD 구축

진행 방식

매주 토요일 오후 2~4시 (2시간)

장소

사회적 거리두기 3,4단계: 온라인(Zoom)
사회적 거리두기 1,2단계: 오프라인(역삼 팁스타운, 변동가능)

방식

1. 각자 public GitHub repository를 생성하고,
    - 매주 자신이 공부해온 내용을 문서화 하고 스터디원들에게 설명하는 방식으로 진행
      ex) 공부하면서 나타난 문제점과 해결방안, 인터넷이나 책에서 찾은 레퍼런스 설명 등
    - 형식/양식은 자유 (ppt보다는 notion에 작성하는게 편함)
2. (필요시) 카카오톡 단체채팅방에 스터디원들에게 문제 해결 요청

스터디 일정 계획

주차	날짜	내용
1	9월 18일	Orientation
2	9월 25일	CRA로 React App 만들고 netlify에 바로 배포하기
3	10월 2일	firebase의 storage를 이용해 파일 저장하기
4	10월 9일	firebase로 oAuth 소셜 로그인 구현
5	10월 23일	jest를 이용한 unit test 작성
6	10월 30일	github CI 사용하기
7	11월 6일	사이드 프로젝트 (1)
8	11월 13일	사이드 프로젝트 (2)

왜 TypeScript인가

JS

```
 3  - 1  // -> 2
 3  + 1  // -> 4
'3' - 1  // -> 2
'3' + 1  // -> '31'

[1, 2, 3] + [4, 5, 6] // -> [1, 2, 34, 5, 6]
(1, 2, 3) + (4, 5, 6) // -> 9
```

이런 짓거리를 못하게 강제할 수 있다.

TypeScript

ECMAScript(aka JavaScript)의 super set으로,
JavaScript와 달리 강력한 type checking 기능을 제공한다.

Microsoft에서 개발함.

대충 어떤 것이 있나

```
// Javascript
const user = (name, age, bloodType) => {
    return ({
        name, age, bloodType,
    });
};
```

```
// Typescript
type BloodType = 'A' | 'B' | 'O' | 'AB';

interface User {
    name: string,
    age: number,
    bloodType: BloodType,
}

const user:User = (name: string, age: number, bloodType: BloodType):User => {
    return({
        name, age, bloodType,
    });
};
```

Javascript의 원시 타입(Primitive Types)

- `string` : 문자열
- `number` : 숫자
- `bigInt` : 되게 큰 정수값
- `null` : null
- `symbol` : 유일값
- `boolean` : 진리값
- `undefined` : 정의되지 않은 값
- `object` : 위에 3개를 제외한 모든 것들 (함수도 `object`)

Typescript의 원시 타입

Typescript는 추가적으로 다음의 원시타입을 제공합니다.

- `any` : 임의의 타입 (모든 타입을 허용)
- `unknown` : 어떤 타입이 올 지 모르지만 아무튼 단일 타입
- `never` : 절대 발생해서는 안되는 타입 (이 타입이 할당되면 에러)
- `void` : 리턴값이 없는 함수

타입추론 (Type by Inference)

```
let helloWorld = "Hello World"; // 누가봐도 string
```

이런 경우에는 자동으로 `helloWorld` 에 `string` 타입이 부여되며,
다른 타입을 넣으려고 하면 에러를 발생시킵니다.

타입 정의하기 (Defining Types)

여러 원시타입을 조합하여 새로운 타입을 만들 수 있습니다.

```
type BloodType = 'A' | 'B' | 'O' | 'AB';

interface User {
    name: string,
    age: number,
    bloodType: BloodType,
}

const user:User = (name: string, age: number, bloodType: BloodType):User => {
    return({
        name, age, bloodType,
    });
};
```

제네릭 (Generics)

C++ 등의 언어에서와 같이 overload를 위한 제너릭을 제공합니다.

```
type Array<T> = T[];

type StringArray = Array<string>;
type NumberArray = Array<number>;
type ObjectWithNameArray = Array<{ name: string }>;
```

추천 개발환경

Visual Studio Code를 사용하는 것을 추천드립니다.
같은 Microsoft에서 개발한거라 Extension 지원이 좋습니다.

개발환경설정

- Node.js
  - conda 또는 nvm를 이용하면 버전관리가 쉽습니다.
- VSCode 등의 IDE

Typescript 컴파일 및 실행

node.js나 브라우저는 `.ts` 파일을 실행할 수 없습니다.
`.ts`를 `.js`로 변환후 실행하기 위해 `tsc`를 사용합니다.

1. typescript 설치

```
npm install typescript
```

2. 작성한 `index.ts` 변환

```
tsc index.ts
```

3. 변환된 `index.js` 실행

```
node index.js
```

레퍼런스 확인 순서

1. Microsoft TypeScript 공식 레퍼런스
2. Mozilla MDN JavaScript 공식 레퍼런스
3. 사용하는 Library 공식 Document
    - React는 Facebook React 공식 레퍼런스

공식 레퍼런스로 해결이 안되면

- StackOverflow 검색
- 해당 Library GitHub Issue 확인
- 개발자 커뮤니티에 질문
- 옆 동료에게 물어보기

참고할만한 자료

1. 타입스크립트 공식 설명서
2. velog: Advanced Type
3. 타입스크립트 playground

과제

1. Typescript 실행 가능 환경 구성하기
2. github repository 생성 후 `index.ts` commit 남기기
3. Linting 개념 찾아보고 내 레포지토리에 ESLint 적용하기