Waffle Studio Frontend Seminar - 4

김기완

Contents

- 배열 Rendering
- API 연동
- CORS
- session/cookie

배열 Rendering

3가지 타입

1. without Key Props
2. Key = index
3. Key = id 등의 unique 한 값


list.map((item, index) => {
return (
<Item />
)
})

Without Key Props

Each child in a list should have a unique “key” prop.

```
▶Warning: Each child in a list should have a unique "key" prop.

Check the render method of `ShoppingList`. See https://fb.me/react-warning-keys for more information.
    in li (at ShoppingListNoKeys.js:9)
    in ShoppingList (at App.js:9)
    in div (at App.js:8)
    in App (at src/index.js:9)
    in StrictMode (at src/index.js:8)
```

Why?

Virtual DOM이 “key” 값을 사용해서 비교

있는 걸 사용할까? 다시 그릴까?

list에 아이템을 추가/삭제 하면 어떻게 될까?

1. 중간에 ‘z’ 추가
2. ‘a’ 삭제

Without Key!

a

b

c

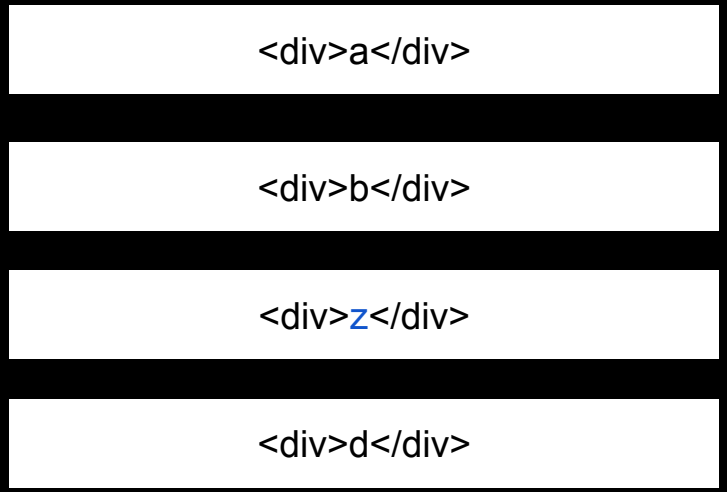
d

[‘a’, ‘b’, ‘c’, ‘d’]

Without Key!

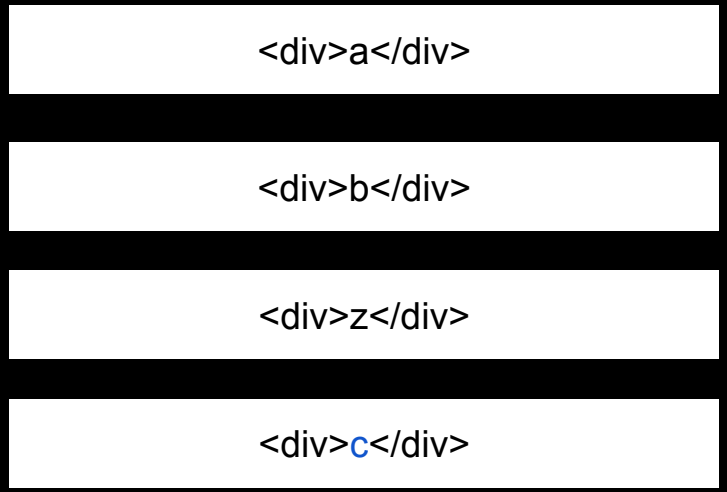
['a', 'b', 'z', 'c', 'd']

Without Key!



['a', 'b', 'z', 'c', 'd']

Without Key!



['a', 'b', 'z', 'c', 'd']

Without Key!

['a', 'b', 'z', 'c', 'd']

Without Key!

['~~a~~', 'b', 'z', 'c', 'd']

Without Key!

['a', 'b', 'z', 'c', 'd']

Without Key!

['a', 'b', 'z', 'c', 'd']

Without Key!

['a', 'b', 'z', 'c', 'd']

Without Key!

b

z

c

d

['~~a~~', 'b', 'z', 'c', 'd']

With Key!

```
<div key={0}>a</div>

<div key={1}>b</div>

<div key={2}>c</div>

<div key={3}>d</div>
```

```
[
    { id: 0, text: 'a' },
    { id: 1, text: 'b' },
    { id: 5, text: 'z' },
    { id: 2, text: 'c' },
    { id: 3, text: 'd' },
]
```

With Key!

```
<div key={0}>a</div>

<div key={1}>b</div>

<div key={5}>z</div>

<div key={2}>c</div>

<div key={3}>d</div>
```

```
[
    { id: 0, text: 'a' },
    { id: 1, text: 'b' },
    { id: 5, text: 'z' },
    { id: 2, text: 'c' },
    { id: 3, text: 'd' },
]
```

With Key!

```
<div key={1}>b</div>

<div key={5}>z</div>

<div key={2}>c</div>

<div key={3}>d</div>
```

```
[
    { id: 0, text: 'a' },
    { id: 1, text: 'b' },
    { id: 5, text: 'z' },
    { id: 2, text: 'c' },
    { id: 3, text: 'd' },
]
```


list.map((item, index) => {
return (
<Item key={index}/>
)
})

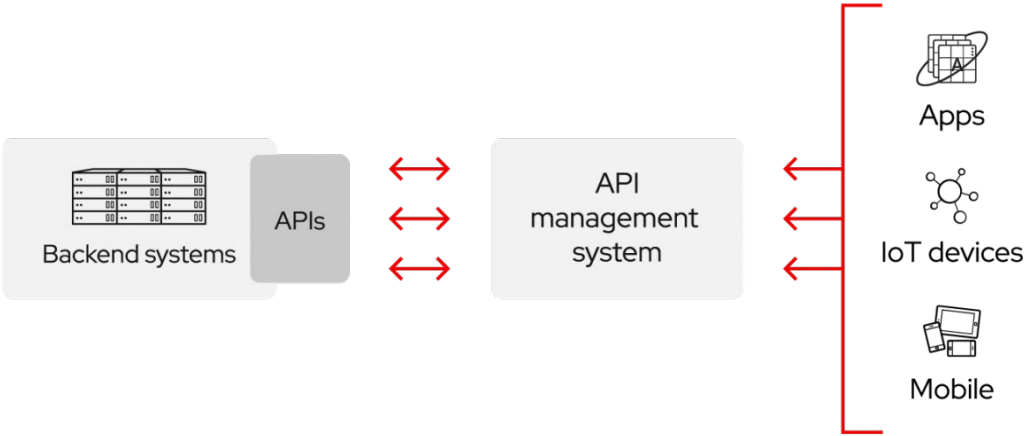

list.map((item, index) => {
return (
<Item key={item.id}/>
)
})

이상한 예시를 봅시다.

API 연동

API를 사용해서 서버와 정보를

주고 받음



REST API?

클라이언트/서버가 소통하는 방식

HTTP 메서드를 사용해서 방식을 구분

- GET: 조회
- POST: 등록
- PUT: 수정
- DELETE: 삭제

https://meetup.toast.com/posts/92

Axios

REST API 요청 시 프로미스 기반으로 처리하는 라이브러리

```
yarn add axios
```

데이터 로딩 시

3가지 상태

1. 결과
2. 로딩 상태
3. 에러

Axios Code Example

JSON Server

연습할 때 백엔드를 짜기엔? 어렵고 귀찮다

대체할 수 있는 연습용 REST API 서버를 제공

```
npx json-server ./data.json --port 4000
```

Context + API 연동

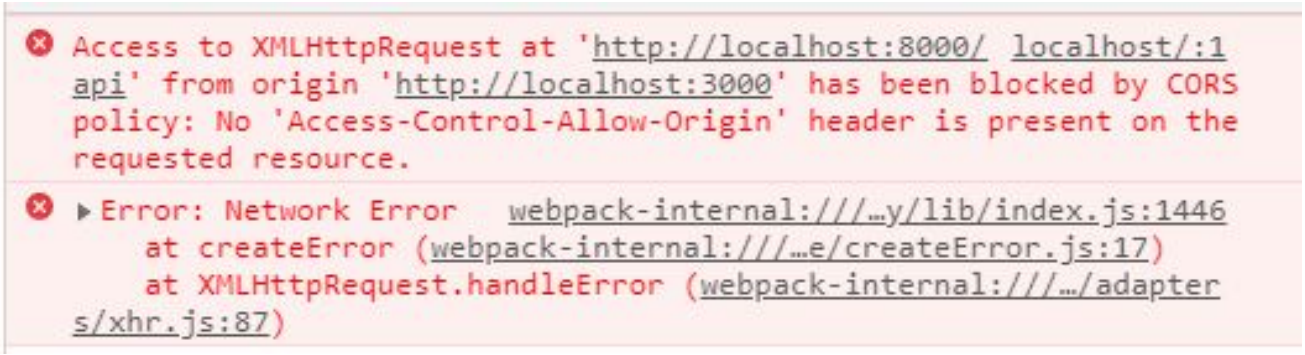
전역에서 필요한 값 (ex. 로그인 정보)

- Context에서 API를 연동

Context + Axios Code Example

CORS

API 연동하다보면...



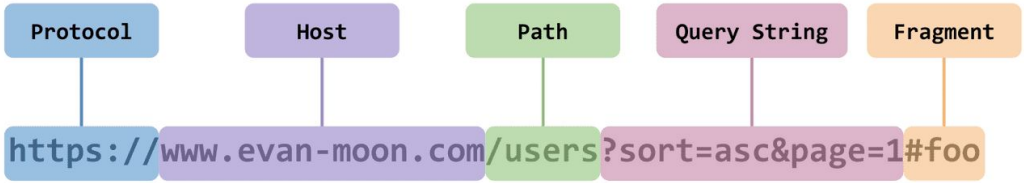
SOP + CORS

Same Origin Policy

Cross-Origin Resource Sharing

Origin?

- Protocol + Host + Port
- location.origin



SOP

Same-Origin Policy - 같은 Origin 에서만 리소스 공유 가능

다른 Origin을 무작정 막기엔,,,?

- 다른 Origin의 리소스 사용은 흔하디 흔한 일
- 몇 가지 예외 조항을 지키면 허용 -> CORS 정책을 지킨 리소스 요청

다른 Origin로 리소스 요청 + CORS 정책 위반 -> 다른 Origin의 리소스 사용 불가

왜 이렇게 귀찮게 할까,,,?

클라이언트(브라우저)의 보안은???

- DOM
- 난독화 된 자바스크립트 코드
- <script> 태그

모두 개발자 도구에서 확인 가능

왜 이렇게 귀찮게 할까,,,?

제약이 없으면?

- XSS(Cross-Site-Scripting)
- CSRF(Cross-Site Request Forgery)

웹에서 코드가 실행된 것처럼 속일 수 있다

개발자가 신경쓸 게 많아져요,,,

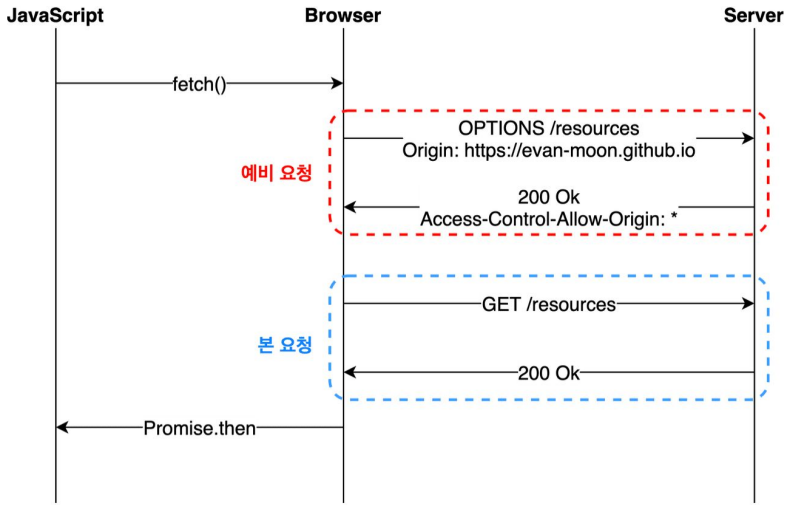
어떻게 동작하나요?

Preflight Request

- OPTIONS 메소드
- 유효성 검사

유효성 검사

- 응답 헤더
- Allow-Control-Allow-Origin



해결법은?

Access-Control-Allow-Origin 세팅하기

- 서버에서 해줍니다
- 에러나면 서버 개발자한테 뭐라하면 됩니다.

HTTP의 특징

Connectionless

- Response가 오면 연결을 차단

Stateless

- 클라이언트의 정보를 모름

로그인을 유지하고 싶으면?

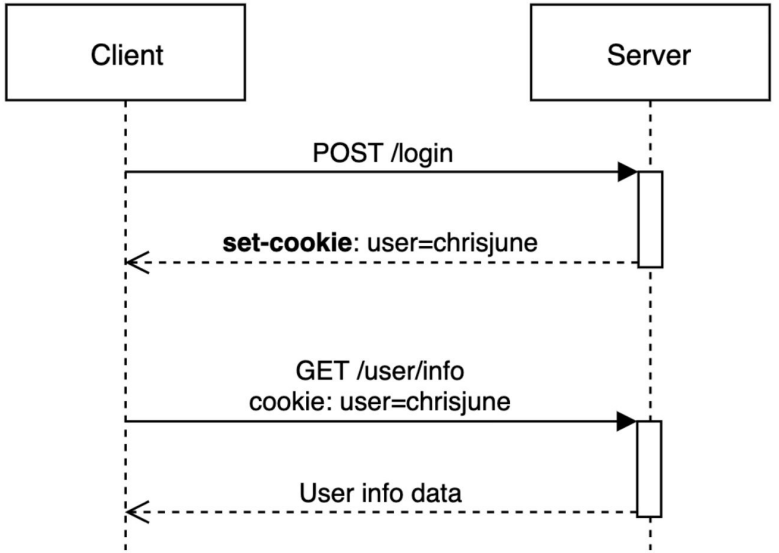
- Session or Cookie

Cookie

클라이언트 로컬에 저장

(Key, value) 의 작은 데이터파일

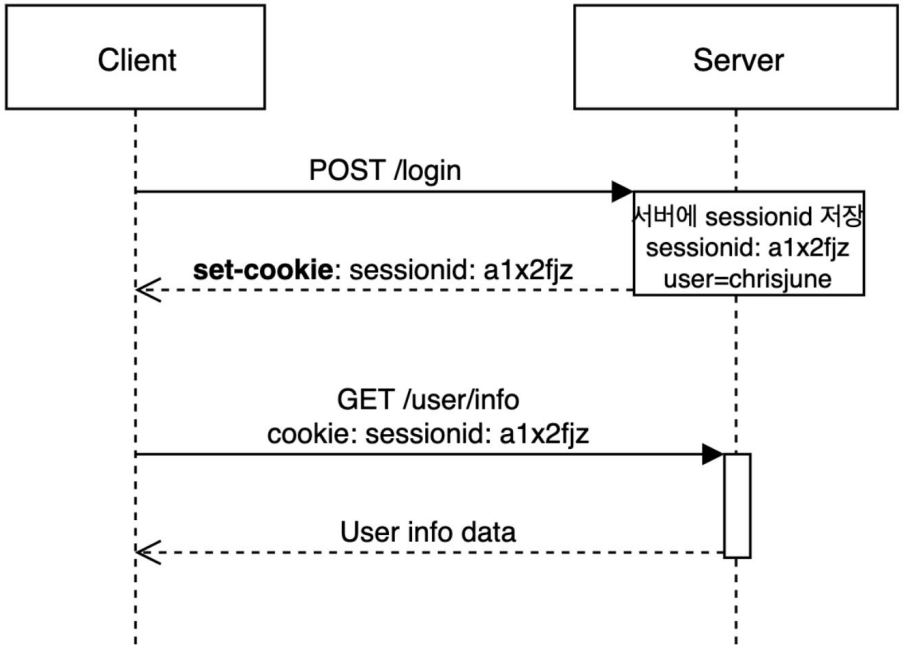
Session Cookie vs Persistent Cookie



Session

서버에 클라이언트의 상태를 저장

클라이언트는 세션 ID를 쿠키로 저장



Session vs Cookie

저장 위치

- 서버 vs 클라이언트

보안

- 비교적 안전 vs 탈취/변조 위험

Session vs Cookie

라이프 사이클

- 서버 측 관리 vs 브라우저에서 관리

속도

- 서버에서 다시 접근 vs 헤더를 바로 참조

과제

이번에도 저번 과제에 몇가지 추가/수정사항이 있을 예정입니다.

내일 밤까지 올려드리겠습니다.