API 연동하기

준비하기

프로젝트 생성

- 백엔트 서버와 통신을 위한 신규 프로젝트 생성하기

$ npx create-react-app api-integrate

- axios 라는 라이브러리를 설치하기

$ cd api-integrate

$ yarn add axios

- axios를 사용해서 GET, PUT, POST, DELETE 등의 메서드로 API 요청할 수 있음
  - GET: 데이터 조회
  - POST: 데이터 등록
  - PUT: 데이터 수정
  - DELETE: 데이터 제거

REST API

Fetch

- FETCH API

```
const getPostsData = () => {
  fetch(‘https://jsonplaceholder.typicode.com/posts’)
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.log(error));
  }
}
 getPostsData();
```

REST API

Axios

- Axios

```
const getPostsData = () => {
  axios
  .get("https://jsonplaceholder.typicode.com/posts")
  .then(data => console.log(data.data))
  .catch(error => console.log(error));
 };
getPostsData();
```

REST API

Fetch vs. Axios

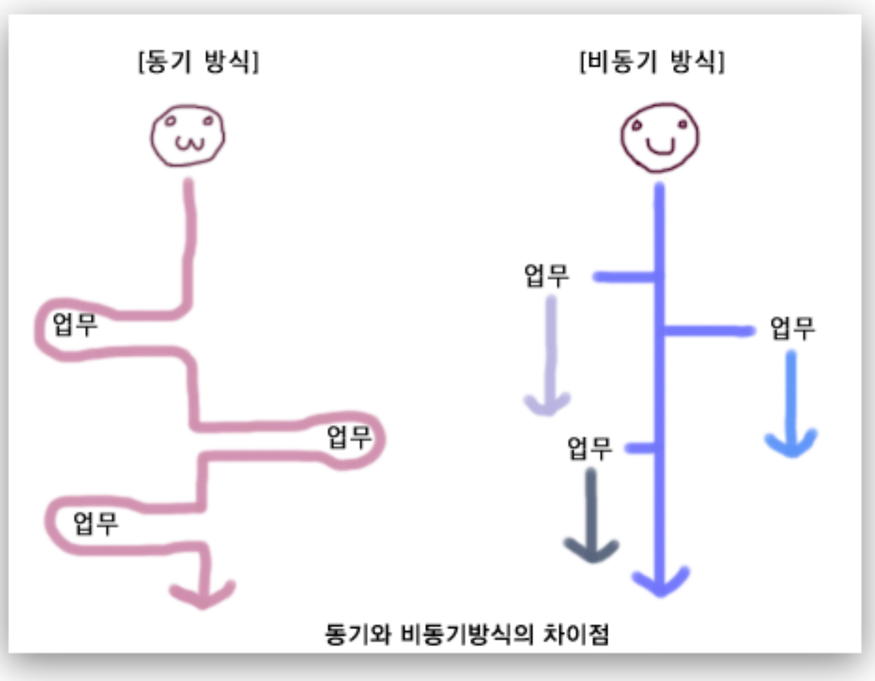
```
axios.get('http://localhost:4000/todos/')
    .then(res => {
        console.log("AXIOS RES", res)
        this.setState({ todos: res.data })
    })
    .catch(function (error) {
        console.log(error)
    })

fetch(`http://localhost:4000/todos`)
    .then(res => res.json())
    .then(res => {
        console.log("FETCH RES", res)
        this.setState({ todos: res })
    })
    .catch(function (error) {
        console.log(error)
    })
```

준비하기

Axios

- 리액트에서 많이 쓰이는 HTTPClient 라이브러리의 하나로서 REST 인터페이스 지원
- Promise 기반이고, async/await 코드를 쉽게 구현할 수 있게 해줌

```
const axios_get = () => {
        axios.get("http://localhost:8080/get")
            .then((response)=> {
                console.log(response);
        })
        .catch((error)=> {
                console.log(error);
      })
 }
```



준비하기

Axios

- 데이터를 요청하는 경우 axios.get()를 사용

```
import axios from 'axios';

axios.get(‘/users/1');
```

- 새로운 데이터를 등록하고 싶다면 axios.post() 를 사용

```
axios.post('/users', {
  username: 'blabla',
  name: 'blabla'
});
```

Axios 사용하기

- axios 의 get 메소드로 자료 가져오기

```
// photos, setPhotos 비구조화 할당
let [photos, setPhotos] = useState([]);

// 통신 메서드
function searchApi() {
    const url = "https://jsonplaceholder.typicode.com/photos";
    axios.get(url)
    .then(function(response) {
        setPhotos(response.data);
        console.log("성공");
    })
    .catch(function(error) {
        console.log("실패");
    })

}
```

Axios 사용하기

- 요청 응답결과 출력하기

```
// 조회 데이터 존재할 경우
if(photos.length > 0) {
    return (
        photos.map(photo => (
            (photo.id < 10) ? (
                <div key={photo.id}>
                    <img src={photo.thumbnailUrl} alt="img" />
                    <p>title : {photo.title}</p>
                </div>)
            : null
        ))
    );
} else { // 조회 데이터 존재하지 않을 경우
    return (
        <div>
            <button onClick={searchApi}> 불러오기 </button>
        </div>
    )
}
```

title : accusamus beatae ad facilis cum similique qui sunt

title : reprehenderit est deserunt velit ipsam

Axios 사용하기

(GET) useState 와 useEffect

App_Effect.js

```
import axios from "axios";
import React from “react";
const baseURL = “https://jsonplaceholder.typicode.com/posts/1";

export default function App() {
  const [post, setPost] = React.useState(null);
  React.useEffect(() => {
    axios.get(baseURL).then((response) => {
        console.log (response.data);
        setPost(response.data);
    });
  }, []);
```

Axios 사용하기
(GET) useState 와 useEffect

App_Effect.js

```
if (!post) return null;
 return (
    <div>
      <h1>{post.title}</h1>
      <p>{post.body}</p>
    </div>
  );
}
```

Axios 사용하기

(POST) useState 와 useEffect

App_effect_post.js

```
React.useEffect(() => {
    axios.get(`${baseURL}/1`).then((response) => {
      setPost(response.data);
    });
  }, []);

function createPost() {
    axios
      .post(baseURL, {
        title: "Hello World!",
        body: "This is a new post."
      })
      .then((response) => {
        setPost(response.data);
      });
  }
```

Axios 사용하기

(POST) useState 와 useEffect

App_effect_post.js

```
if (!post) return "No post!"

  return (
    <div>
      <h1>{post.title}</h1>
      <p>{post.body}</p>
      <button onClick={createPost}>Create Post</button>
    </div>
  );
```

Axios 사용하기

(GET) useState 와 useEffect

- 사용 API 주소 : https://jsonplaceholder.typicode.com/users


```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2,
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
      "street": "Victor Plains",
      "suite": "Suite 879",
      "city": "Wisokyburgh",
      "zipcode": "90566-7771",
      "geo": {
        "lat": "-43.9509",
        "lng": "-34.4618"
      }
    },
```

Axios 사용하기

(GET) useState 와 useEffect

- useState 를 사용하여 요청 상태를 관리하고, useEffect 를 사용하여 컴포넌트가 렌더링 되는 시점에 요청을 시작하는 작업을 수행
- 요청에 대한 3가지 상태를 관리함
  - 요청의 결과
  - 로딩 상태
  - 에러

Axios 사용하기

(GET) useState 와 useEffect

- useState 와 useEffect 로 데이터 로딩

```
const [users, setUsers] = useState(null);
const [loading, setLoading] = useState(false);
const [error, setError] = useState(null);
```

```
useEffect(() => {
  const fetchUsers = async () => {
    try {
      // 요청이 시작 할 때에는 error 와 users 를 초기화하고
      setError(null);
      setUsers(null);
      // loading 상태를 true 로 바꿉니다.
      setLoading(true);
      const response = await axios.get(
        'https://jsonplaceholder.typicode.com/users'
      );
      setUsers(response.data); // 데이터는 response.data 안에 들어있습니다.
    } catch (e) {
      setError(e);
    }
    setLoading(false);
  };

  fetchUsers();
}, []);
```

Axios 사용하기
(GET) useState 와 useEffect

- App.js

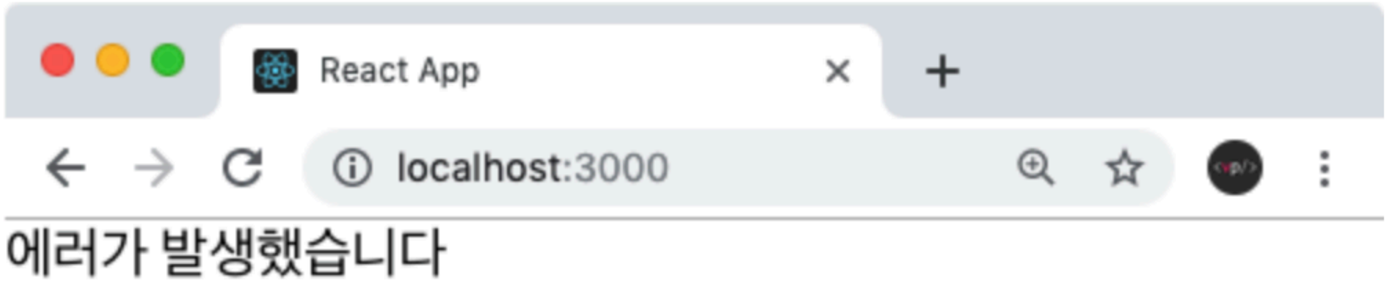
```
import React from 'react';
import Users from './Users';

function App() {
  return <Users />;
}

export default App;
```

Axios 사용하기

(GET) useState 와 useEffect

- 잘못된 주소의 경우 오류 발생처리하기

```
const response = await axios.get(
  'https://jsonplaceholder.typicode.com/users/showmeerror'
);
```



Axios 사용하기
(POST) useState 와 useEffect

- 다시읽기

```
  if (loading) return <div>로딩중..</div>;
  if (error) return <div>에러가 발생했습니다</div>;
  if (!users) return null;
  return (
    <>
      <ul>
        {users.map(user => (
          <li key={user.id}>
            {user.username} ({user.name})
          </li>
        ))}
      </ul>
      <button onClick={fetchUsers}>다시 불러오기</button>
    </>
  );
}

export default Users;
```

useReducer() 실습

Axios 사용하기

useReducer 로 요청 상태 관리

- LOADING, SUCCESS, ERROR 액션 처리

```
const [state, dispatch] = useReducer(reducer, {
  loading: false,
  data: null,
  error: null
});

const fetchUsers = async () => {
  dispatch({ type: 'LOADING' });
  try {
    const response = await axios.get(
      'https://jsonplaceholder.typicode.com/users'
    );
    dispatch({ type: 'SUCCESS', data: response.data });
  } catch (e) {
    dispatch({ type: 'ERROR', error: e });
  }
};
```

```
function reducer(state, action) {
  switch (action.type) {
    case 'LOADING':
      return {
        loading: true,
        data: null,
        error: null
      };
    case 'SUCCESS':
      return {
        loading: false,
        data: action.data,
        error: null
      };
    case 'ERROR':
      return {
        loading: false,
        data: null,
        error: action.error
      };
    default:
      throw new Error(`Unhandled action type: ${action.typ
  }
}
```

Axios 사용하기

useReducer 로 요청 상태 관리

- LOADING, SUCCESS, ERROR 액션 처리

```
function reducer(state, action) {
  switch (action.type) {
    case 'LOADING':
      return {
        loading: true,
        data: null,
        error: null
      };
    case 'SUCCESS':
      return {
        loading: false,
        data: action.data,
        error: null
      };
    case 'ERROR':
      return {
        loading: false,
        data: null,
        error: action.error
      };
    default:
      throw new Error(`Unhandled action type: ${action.type}`);
  }
}
```

Axios 사용하기

useReducer 로 요청 상태 관리

```
function Users() {
  const [state, dispatch] = useReducer(reducer, {
    loading: false,
    data: null,
    error: null
  });

  const fetchUsers = async () => {
    dispatch({ type: 'LOADING' });
    try {
      const response = await axios.get(
        'https://jsonplaceholder.typicode.com/users'
      );
      dispatch({ type: 'SUCCESS', data: response.data });
    } catch (e) {
      dispatch({ type: 'ERROR', error: e });
    }
  };

  useEffect(() => {
    fetchUsers();
  }, []);
```

```
  const { loading, data: users, error } = state; // state.

  if (loading) return <div>로딩중..</div>;
  if (error) return <div>에러가 발생했습니다</div>;
  if (!users) return null;
  return (
    <>
      <ul>
        {users.map(user => (
          <li key={user.id}>
            {user.username} ({user.name})
          </li>
        ))}
      </ul>
      <button onClick={fetchUsers}>다시 불러오기</button>
    </>
  );
}

export default Users;
```