11월 19일(금)

1. 엘리스 SW async await , fetch 실습

- 유저 정보 요청하여 변환하기

유저 정보 요청하여 변환하기

`https://randomuser.me` API를 이용해, 여러 유저의 정보를 받아와 가공하는 함수를 만듭니다.

API 정보를 가공하여 활용하는 법을 연습해봅니다. 아래의 지시사항에 따라 `User.js` 의 `requestUsers` 함수를 만들어 봅시다.

지시사항

1. 유저 정보를 변환합니다.

```
- user.email -> email
- user.name.first, user.name.last -> name -
- user.picture.large -> pictureUrl
- user.login.username -> username
- user.location.country, user.location.stat
- user.dob.age -> user.age
```

- 예시


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Template</title>
  </head>
  <body>
    <div id="root">
      <main>
        <h2>requestUser App</h2>

        <div class="app">
          <button id="request-user-button">데이터 요청하기</button>

          <ul id="result"></ul>
        </div>
      </main>
    </div>

      <script src="./index.js"></script>
```

```
  </body>
</html>
```

```
import "./app.css";
import requestUser from "./User";

function createUserListItem(user) {
  return `<li style="list-style:none"><pre><code>${JSON.stringify(
    user,
    null,
    2
  )}</code></pre></li>`;
}

function renderUsers(users) {
  return users.map((user) => createUserListItem(user).trim()).join("");
}

const App = () => {
  const button = document.getElementById("request-user-button");
  const result = document.getElementById("result");

  button.addEventListener("click", () => {
    requestUser().then((users) => {
      result.innerHTML = renderUsers(users);
    });
  });
};

export default App;
```

```
import App from "./App";
import "./index.css";

const run = () => {
  window.addEventListener("DOMContentLoaded", () => {
    App();
  });
};

run();
```

```
const fetchUsers = () => {
  return fetch("https://randomuser.me/api/?results=10")
    .then((response) => response.json())
    .then((data) => data.results);
};

export default { fetchUsers };
```

```
import API from "./api";

function transformUser(user) {
      const email = user.email;
      const name = `${user.name.first} ${user.name.last}`;
      const pictureUrl = user.picture.large;
      const username = user.login.username;
      const location = `${user.location.country}, ${user.location.state}, ${user.location.city}`;
      const age = user.dob.age;

      return {
        email,
        name,
        pictureUrl,
        username,
        location,
        age,
      };
```

```
}



const requestUsers = () => {
  return API.fetchUsers().then((users) => {
    // 유저 정보를 변화하고, 필터링하는 코드를 작성해보세요.
    return users.map(user => transformUser(user))
                .filter(user => user.age > 40)
  });
};

export default requestUsers;
```

- **Posts 정보 조합하기**

Posts 정보 조합하기

유저 정보, post 정보, 커멘트 정보를 응답하는 API를 활용하여, 하나의 포스트 목록을 만들어보세요.

`https://jsonplaceholder.typicode.com` 의 API를 이용하여 데이터를 조합해봅니다.

`Posts.js` 의 `requestPosts` 메서드를 만듭니다. `requestPosts` 는 posts 정보를 아래 지시사항에 따라 조합하여 리턴하는 비동기 함수입니다.

지시사항

1. `api.js` 에 제공된 `fetchPosts` , `fetchUsers` , `fetchComments` API를 이용하여 포스트 정보, 유저 정보, 커멘트 정보를 각각 요청하세요.

2. 얻어온 데이터를 활용하여 정보를 조합하세요.

   - `fetchPosts` API를 이용하여 `posts` 목록을 가져옵니다.
   - 하나의 `post` 객체에는 `userId` 가 들어있습니다. `userId` 필드를 제거하고 `userId` 에 해당하는 유저 정보를 `post` 정보에 합칩니다. 유저 정보는 `fetchUsers` 에서 얻은 정보를 활용합니다.
   - 하나의 `post` 객체에는 `id` 가 들어있습니다. 이 post id를 이용하여 comments를 요청하고, 새롭게 `comments` 필드를 추가하여 응답받은 커멘트 목록을 `post` 합칩니다. 커멘트 정보는 `fetchComments` API를 활용합니다.
   - 하나의 `post` 정보에 `user` 필드를 추가하고, 그 필드에 `post.userId` 에 매칭되는 유저 정보를 넣습니다.
   - 하나의 `post` 정보에 `comments` 필드를 추가하고, 그 필드에 `comment.postId` 에 매칭되는 커멘트들의 목록을 넣습니다.
   - 최종적으로 하나의 `post` 정보는 모든 post 정보에 추가로 `user` , `comments` 필드를 갖게 됩니다.
   - `fetchPosts` 가 이렇게 정보가 합쳐진 `post` 들의 배열을 반환하도록 합니다.

3. 테스트 앱을 활용하여 결과가 제대로 만들어지는지 확인해보세요.

Tips!

- `fetchPosts` 는 포스트 목록을 반환합니다.
- `fetchUsers` 는 유저 목록을 반환합니다.
- `fetchComments` 는 postId를 파라미터로 받아 해당하는 커멘트 목록을 반환합니다.

```
import API from "./api";

const requestPosts = () => {
    return Promise.all([
        API.fetchPosts(),
        API.fetchUsers()
    ])

        .then(([ posts, users ]) => {
            return fetchCommentsByPosts(posts)
                    .then(comments => [ posts, users, comments ])

        })
        .then(([ posts, users, comments ]) => {
            const userMap = createUserMap(users);
            const commentMap = createCommentMap(comments);

            return transformPosts(posts, userMap, commentMap);

        })

};


export default requestPosts;



function transformPosts(posts, userMap, commentMap) {
    return posts.map(({ id, userId, ...rest })=> {
        return {
            ...rest,
            user: userMap[userId],
            comments: commentMap[id]
        }

    })

}



function createUserMap(users) {
    return users.reduce((map, user) => {
        map[user.id] = user
        return map
    }, {})
}


function fetchCommentsByPosts(posts) {
    return Promise.all(
        posts.map(post => API.fetchComments(post.id))
    )
    .then(commentArrays => commentArrays.flatMap(array => array))  // flatMap 은 이중 배열을 모두 풀어 배열을 원소로 만든다.
}


function createCommentMap(comments) {
    return comments.reduce((map, comment) => {
        const array = map[comment.postId] ? map[comment.postId] : [];
        array.push(comment)
        map[comment.postId] = array
        return map
    },{})
}
```

```
import "./app.css";
import requestPosts from "./Posts";

function Comments(comments) {
  return `
  <ul style="padding:0">
    ${comments
```

```
      .map(
        (comment) =>
          `<li style="list-style:none"><strong>${comment.email}</strong> - ${comment.body}</li>`
      )
      .join("")}
  </ul>
  `;
}

function Post(post) {
  return `<div style="width:350px;">
    <h2>제목 - ${post.title}</h2>
    <p>${post.body}</p>
    <div><strong>By ${post.user.name}</strong></div>
    <div>
      <h4>comments</h4>
      ${Comments(post.comments)}
    </div>
    </div>`;
}

function PostList(posts) {
  return `
  <ul style="padding:0">
    ${posts.map(Post).join("")}
  </ul>`;
}

const App = () => {
  const button = document.getElementById("request-user-button");
  const result = document.getElementById("result");

  button.addEventListener("click", () => {
    requestPosts().then((posts) => {
      console.log(posts);
      result.innerHTML = PostList(posts);
    });
  });
};

export default App;
```

```
const API_URL = "https://jsonplaceholder.typicode.com";

const fetchPosts = () =>
  fetch(`${API_URL}/posts`).then((response) => response.json());

const fetchUsers = () =>
  fetch(`${API_URL}/users`).then((response) => response.json());

const fetchComments = (postId) =>
  fetch(`${API_URL}/posts/${postId}/comments`).then((response) =>
    response.json()
  );

export default { fetchPosts, fetchUsers, fetchComments };
```

```
const testPostData = [
  {
    userId: 1,
    id: 1,
    title:
      "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    body:
      "quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est au
  },
  {
    userId: 2,
    id: 11,
    title: "et ea vero quia laudantium autem",
    body:
      "delectus reiciendis molestiae occaecati non minima eveniet qui voluptatibus accusamus in eum beatae sit vel qui neque voluptate
  },
```

```
  {
    userId: 3,
    id: 28,
    title: "delectus ullam et corporis nulla voluptas sequi",
    body:
      "non et quaerat ex quae ad maiores maiores recusandae totam aut blanditiis mollitia quas illo ut voluptatibus voluptatem similiq
  },
];

const testUserData = [
  {
    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",
      },
    },
    phone: "010-692-6593 x09125",
    website: "anastasia.net",
    company: {
      name: "Deckow-Crist",
      catchPhrase: "Proactive didactic contingency",
      bs: "synergize scalable supply-chains",
    },
  },
  {
    id: 3,
    name: "Clementine Bauch",
    username: "Samantha",
    email: "Nathan@yesenia.net",
    address: {
      street: "Douglas Extension",
      suite: "Suite 847",
      city: "McKenziehaven",
      zipcode: "59590-4157",
      geo: {
        lat: "-68.6102",
        lng: "-47.0653",
      },
    },
    phone: "1-463-123-4447",
    website: "ramiro.info",
    company: {
      name: "Romaguera-Jacobson",
      catchPhrase: "Face to face bifurcated interface",
      bs: "e-enable strategic applications",
    },
  },
];

const testCommentData = [
  {
    postId: 1,
    id: 1,
    name: "id labore ex et quam laborum",
    email: "Eliseo@gardner.biz",
```

```
    body:
      "laudantium enim quasi est quidem magnam voluptate ipsam eos tempora quo necessitatibus dolor quam autem quasi reiciendis et nam
  },
  {
    postId: 1,
    id: 2,
    name: "quo vero reiciendis velit similique earum",
    email: "Jayne_Kuhic@sydney.com",
    body:
      "est natus enim nihil est dolore omnis voluptatem numquam et omnis occaecati quod ullam at voluptatem error expedita pariatur ni
  },

  {
    postId: 11,
    id: 51,
    name: "molestias et odio ut commodi omnis ex",
    email: "Laurie@lincoln.us",
    body:
      "perferendis omnis esse voluptate sit mollitia sed perferendis nemo nostrum qui vel quis nisi doloribus animi odio id quas",
  },
  {
    postId: 11,
    id: 52,
    name: "esse autem dolorum",
    email: "Abigail.OConnell@june.org",
    body:
      "et enim voluptatem totam laudantium impedit nam labore repellendus enim earum aut consectetur mollitia fugit qui repellat exped
  },

  {
    postId: 28,
    id: 136,
    name: "et expedita est odit",
    email: "Rosanna_Kunze@guy.net",
    body:
      "cum natus qui dolorem dolorum nihil ut nam tempore modi nesciunt ipsum hic rem sunt possimus earum magnam similique aspernatur
  },
  {
    postId: 28,
    id: 137,
    name: "saepe dolore qui tempore nihil perspiciatis omnis omnis magni",
    email: "Ressie.Boehm@flossie.com",
    body:
      "reiciendis maiores id voluptas sapiente deserunt itaque ut omnis sunt necessitatibus quibusdam dolorem voluptatem harum error",
  },
];

const fetchUsers = () => new Promise((resolve) => resolve(testUserData));
const fetchPosts = () => new Promise((resolve) => resolve(testPostData));
const fetchComments = (postId) =>
  new Promise((resolve) => {
    resolve(testCommentData.filter((comment) => comment.postId === postId));
  });

const exportModule = { fetchUsers, fetchPosts, fetchComments };

module.exports = exportModule;
```

```
import requestPosts from "./Posts";

jest.mock("./api", () => require("./apiMock"));

const testResult = [
  {
    title:
      "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    body:
      "quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est au
    user: {
      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",
      },
    },
    comments: [
      {
        postId: 1,
        id: 1,
        name: "id labore ex et quam laborum",
        email: "Eliseo@gardner.biz",
        body:
          "laudantium enim quasi est quidem magnam voluptate ipsam eos tempora quo necessitatibus dolor quam autem quasi reiciendis et
      },
      {
        postId: 1,
        id: 2,
        name: "quo vero reiciendis velit similique earum",
        email: "Jayne_Kuhic@sydney.com",
        body:
          "est natus enim nihil est dolore omnis voluptatem numquam et omnis occaecati quod ullam at voluptatem error expedita pariatu
      },
    ],
  },
  {
    title: "et ea vero quia laudantium autem",
    body:
      "delectus reiciendis molestiae occaecati non minima eveniet qui voluptatibus accusamus in eum beatae sit vel qui neque voluptate
    user: {
      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",
        },
      },
      phone: "010-692-6593 x09125",
      website: "anastasia.net",
      company: {
        name: "Deckow-Crist",
        catchPhrase: "Proactive didactic contingency",
        bs: "synergize scalable supply-chains",
      },
    },
    comments: [
      {
        postId: 11,
        id: 51,
        name: "molestias et odio ut commodi omnis ex",
        email: "Laurie@lincoln.us",
        body:
          "perferendis omnis esse voluptate sit mollitia sed perferendis nemo nostrum qui vel quis nisi doloribus animi odio id quas",
      },
      {
        postId: 11,
        id: 52,
        name: "esse autem dolorum",
        email: "Abigail.OConnell@june.org",
        body:
          "et enim voluptatem totam laudantium impedit nam labore repellendus enim earum aut consectetur mollitia fugit qui repellat e
      },
    ],
  },
  {
    title: "delectus ullam et corporis nulla voluptas sequi",
    body:
      "non et quaerat ex quae ad maiores maiores recusandae totam aut blanditiis mollitia quas illo ut voluptatibus voluptatem similiq
    user: {
      id: 3,
      name: "Clementine Bauch",
      username: "Samantha",
      email: "Nathan@yesenia.net",
      address: {
        street: "Douglas Extension",
```

```
        suite: "Suite 847",
        city: "McKenziehaven",
        zipcode: "59590-4157",
        geo: {
          lat: "-68.6102",
          lng: "-47.0653",
        },
      },
      phone: "1-463-123-4447",
      website: "ramiro.info",
      company: {
        name: "Romaguera-Jacobson",
        catchPhrase: "Face to face bifurcated interface",
        bs: "e-enable strategic applications",
      },
    },
    comments: [
      {
        postId: 28,
        id: 136,
        name: "et expedita est odit",
        email: "Rosanna_Kunze@guy.net",
        body:
          "cum natus qui dolorem dolorum nihil ut nam tempore modi nesciunt ipsum hic rem sunt possimus earum magnam similique asperna
      },
      {
        postId: 28,
        id: 137,
        name: "saepe dolore qui tempore nihil perspiciatis omnis omnis magni",
        email: "Ressie.Boehm@flossie.com",
        body:
          "reiciendis maiores id voluptas sapiente deserunt itaque ut omnis sunt necessitatibus quibusdam dolorem voluptatem harum err
      },
    ],
  },
];

describe("Test requestPosts", () => {
  test("포스트 정보를 리턴합니다.", () => {
    return requestPosts().then((result) => {
     console.log('result : ', result)
     console.log('test result : ', testResult)
      expect(result).toEqual(testResult);
    });
  });
});
```