8. 데이터 포함시키기 (2)

Prof. Seunghyun Park (sp@hansung.ac.kr)

Division of Computer Engineering

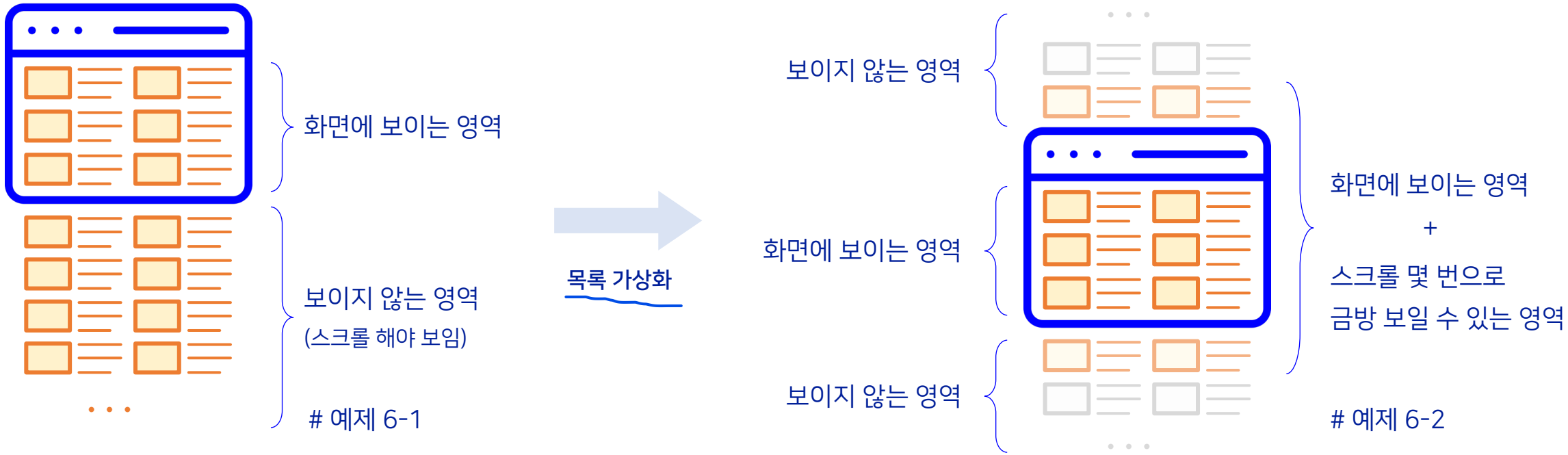
한성대학교 HANSUNG UNIVERSITY

학습 목표: 8장. 데이터 포함시키기

- 데이터 요청하기
  - `fetch()`: `promise`, `async/await`
  - `XMLHttpRequest()`, `axios.get()`
- React App에서의 데이터 요청
  - `useState()`, `useEffect()` 활용
- 웹 스토리지와 활용 예
- Promise 상태 처리
- 렌더 프롭
- 대량 데이터 처리
  - 가상화 리스트
- 데이터 요청 훅과 컴포넌트
  - `useFetch()` 훅
  - `<Fetch />` 컴포넌트
- 여러 요청 처리하기

대량 데이터 처리

5,000개의 요소를 가진 데이터를 로딩하는 경우, 화면에 보이지 않는 영역까지 미리 읽어야 할 것인가?



- 다량의 데이터를 화면에 표시하기 위해
  모든 데이터를 한 번에 읽어 렌더링할 경우
  성능 이슈 발생 가능 (메모리, 프로세스, 네트워크 등)

- 바로 화면에 보이는 영역과
  스크롤 몇 번으로 금방 보일 수 있는 영역의 데이터만 렌더링
- 나머지는 필요시 필요한 부분만 다시 렌더링

5,000명의 faker 데이터 렌더링

```
/* ch08/proj/06-1/src/data/bigList.js */
import { faker } from "@faker-js/faker";

const bigList = [...Array(5000)].map(() => ({
  name: faker.person.fullName(),
  email: faker.internet.email(),
  avatar: faker.internet.avatar()
}));

export default bigList;
```

더미 데이터 생성:
faker 패키지 설치

요소가 5000개인
빈 배열을 생성하여 반환

배열의 각 요소에 faker 데이터
{ name, email, 이미지 경로}를 객체로 입력

```
/* ch08/proj/06-1/src/component/List.js */

const List = ({ data = [], renderItem, renderEmpty }) => (
  !data.length ? renderEmpty : (
    <ul>
      {data.map((item, i) => (
        <li key={i}>{renderItem(item)}</li> ))}
    </ul> )
);

export default List;
```

요소가 5000개인 데이터를 bigList로부터 전달받아
로 반환

```
> npm install -D @faker-js/faker
```

https://fakerjs.dev/guide/

```
/* ch08/proj/06-1/src/App.js */
import bigList from "./data/bigList";
import List from "./component/List";

const App = () => {
  const renderItem = item => (
    <div style={{ display: "flex" }}>
      <img src={item.avatar} alt={item.name} width={50} />
      <p>{item.name} - {item.email}</p>
    </div>
  );
  return <List data={bigList} renderItem={renderItem} />;
};

export default App;
```

요소가 5000개인 데이터 렌더링

화면에 보이는 영역

보이지 않는 영역
(스크롤 해야 보임)

보이지 않는 영역까지 모두 렌더링

목록 가상화 구현: react-window의 FixedSizeList

```
/* ch08/proj/06-2/src/App.js */
import bigList from "./data/bigList";
import { FixedSizeList } from "react-window";

const App = () => {
 const renderRow = ({ index, style }) => (
  <div style={{ ...style, ...{display: "flex"} }}>
    <img src={bigList[index].avatar}
      alt={bigList[index].name} width={50} />
    <p>{bigList[index].name} - {bigList[index].email}</p>
  </div>
 );
 return (
  <FixedSizeList
    height={window.innerHeight-20}
    width={window.innerWidth-20}
    itemCount={bigList.length} itemSize={50}>
    {renderRow}
  </FixedSizeList>
 );
};
export default App;
```

react-window 패키지 설치하고, FixedSizeList 컴포넌트 활용

목록으로 렌더링할 item 정의

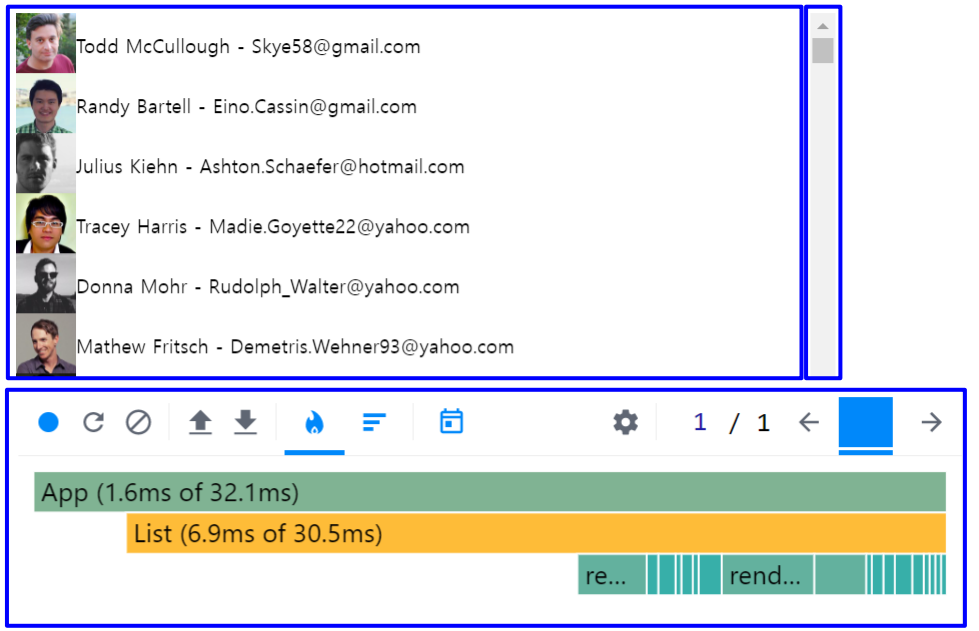
전체 목록의 폭과 높이, 전체 item의 개수, 각 item의 크기 확인

전체 목록의 크기를 계산하고, 화면에 표시할 영역에만 데이터를 렌더링

```
> npm install react-window
```

- react-window: 큰 목록의 데이터를 효율적으로 렌더링하도록 도와주는 라이브러리

https://react-window.vercel.app/#/examples/list/fixed-size



데이터 요청: useFetch() 커스텀 훅

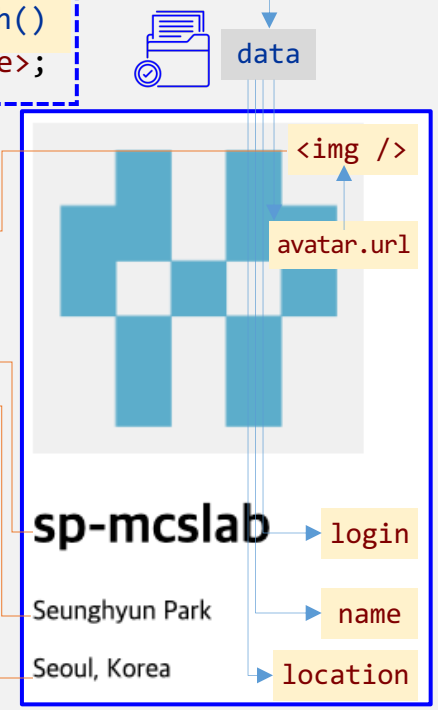
예제 4-1과 유사

```
/* ch08/proj/07-1/src/GithubUser.js */
import { useFetch } from "./hooks";

const GithubUser = ({ login }) => {
  const { loading, data, error } =
    useFetch(`https://api.github.com/users/${login}`);

  if (loading) return <h1>loading...</h1>;
  if (error)
    return <pre>{JSON.stringify(error, null, 2)}</pre>;

  return (
    <div className="githubUser">
      <img src={data.avatar_url}
        alt={data.login} style={{ width: 200 }} />
      <div>
        <h1>{data.login}</h1>
        {data.name && <p>{data.name}</p>}
        {data.location && <p>{data.location}</p>}
      </div>
    </div>
  );
}

export default GithubUser;
```

데이터의 요청 (로딩)과 렌더링을 분리

요청/읽기: useFetch()

렌더링을 위한 데이터 반환




```
/* ch08/proj/07-1/src/hooks.js */
import { useState, useEffect } from "react";

export const useFetch = uri => {
  const [data, setData] = useState();
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState();

  useEffect(() => {
    if (!uri) return;
    fetch(uri)
    .then(response => response.json())
    .then(setData)
    .then(() => setLoading(false))
    .catch(setError)
  }, [uri]);

  return { loading, data, error };
};
```

```
/* ch08/proj/07-1/src/App.js */
import React, {useState} from "react";
import GithubUser from "./GithubUser";

const App = () => {
  const [login] = useState("sp-mcslab");
  return <GithubUser login={login} />;
};
export default App;
```

데이터 요청: Fetch 컴포넌트로 구조 개선

예제 4-1과 유사

```
/* ch08/proj/07-2/src/components/GithubUser.js */

// import { useFetch } from "../hooks/hooks";
import Fetch from "./Fetch";

const GithubUser = ({ login }) => (
  <Fetch
    uri={`https://api.github.com/users/${login}`}
    renderSuccess={userDetails} />
);

const userDetails = ({ data }) => (
  <div className="githubUser">
    <img src={data.avatar_url}
      alt={data.login} style={{ width: 200 }} />
    <div>
      <h1>{data.login}</h1>
      {data.name && <p>{data.name}</p>}
      {data.location && <p>{data.location}</p>}
    </div>
  </div>
);

export default GithubUser;
```

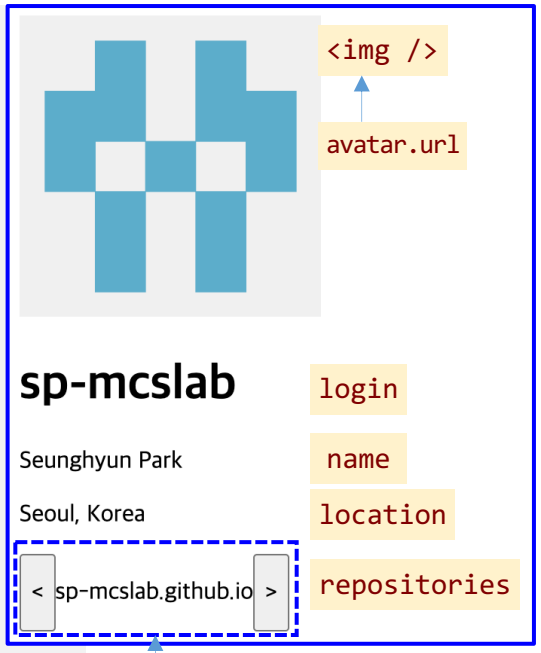
```
/* ch08/proj/07-2/src/components/Fetch.js */
import { useFetch } from '../hooks/hooks';

const Fetch = ({ uri, renderSuccess,
  loadingFallback = <p>loading...</p>,
  renderError = error => (
    <pre>{JSON.stringify(error, null, 2)}</pre>)
}) => {
  const { loading, data, error } = useFetch(uri);

  if (loading) return loadingFallback;
  if (error) return renderError(error);
  if (data) return renderSuccess({ data });
}
export default Fetch;
```

```
/* ch08/proj/07-2/src/hooks.js */
export const useFetch = uri => {
  const [data, setData] = useState();
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState();
  useEffect(() => {
    if (!uri) return;
    fetch(uri)
    .then(response => response.json())
    .then(setData)
    .then(() => setLoading(false))
    .catch(setError)
  }, [uri]);
  return { loading, data, error };
};
```

추가 데이터 요청: repository 목록

여러 URL에서 데이터 가져오기

```
/* ch08/proj/07-4/src/components/GithubUser.js */
import Fetch from "./Fetch";
import UserRepositories from "./UserRepositories";

const GithubUser = ({ login }) => (
  <Fetch
    uri={`https://api.github.com/users/${login}`}
    renderSuccess={userDetails}
  /> );

const userDetails = ({ data }) => (
  <div className="githubUser">
    <img src={data.avatar_url} ... />
    <div>
      <h1>{data.login}</h1>
      {data.name && <p>{data.name}</p>}
      {data.location && <p>{data.location}</p>}
    </div>
    <UserRepositories
      login={data.login}
      onSelect={repoName =>
        console.log(`${repoName} selected`)} />
  </div> );

export default GithubUser;
```




```
/* ch08/proj/07-4/src/components/Fetch.js */
import { useFetch } from '../hooks/hook';
const Fetch = ({
  uri, renderSuccess, loadingFallback, renderError
}) => {
  const { loading, data, error } = useFetch(uri);

  if (loading) return loadingFallback;
  if (error) return renderError(error);
  if (data) return renderSuccess({data});
}
export default Fetch;
```

추가 데이터 요청: useIterator() 훅으로 목록 탐색

```
/* ch08/proj/07-4/src/components/UserRepositories.js */
const UserRepositories = ({ login, selectRepo,
  onSelect = f => f }) => (
  <Fetch
    uri={`https://api.github.com/users/${login}/repos`}
    renderSuccess={ ({data}) => (
      <RepoMenu repositories={data}
        selectRepo={selectRepo} onSelect={onSelect} />
    )}
  /> );
export default UserRepositories;
```

```
/* ch08/proj/07-4/src/components/RepoMenu.js */
const RepoMenu = ( { repositories, onSelect = f => f }) => {
  const [ {name}, previous, next ] = useIterator(repositories);
  useEffect(() => {
    if (!name) return;
    onSelect(name);
  }, [name]);
  return (
    <div style={{ display: "flex "}}>
      <button onClick={previous}>&lt;</button><p>{name}</p>
      <button onClick={next}>&gt;</button>
    </div> );
}
export default RepoMenu;
```

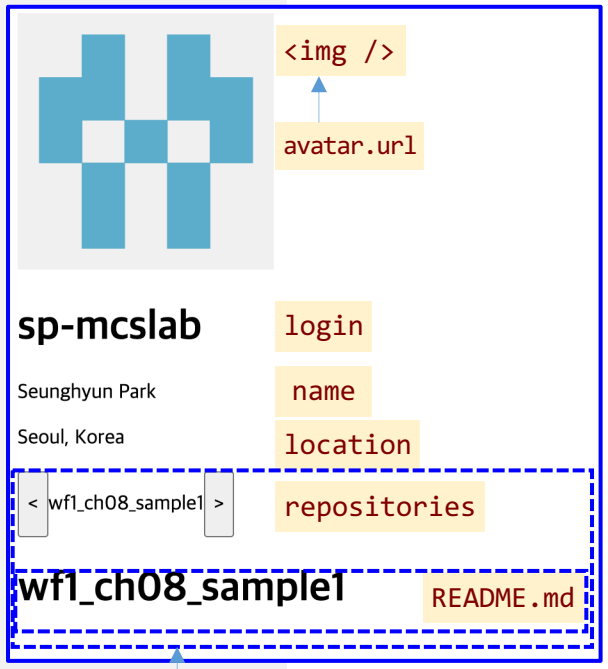
```
/* ch08/proj/07-4/src/hooks/hook.js */
```

```
export const useFetch = uri => {
  const [data, setData] = useState();
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState();
  useEffect(() => {
    if (!uri) return;
    fetch(uri)
    .then(response => response.json())
    .then(setData)
    .then(() => setLoading(false))
    .catch(setError)
  }, [uri]);
  return { loading, data, error };
};
```

```
export const useIterator = (items=[], initialIndex=0) => {
  const [i, setIndex] = useState(initialIndex);
  const prev = useCallback(() => {
    if (i === 0) return setIndex(items.length - 1);
    setIndex(i - 1);
  }, [i]);
  const next = useCallback(() => {
    if (i === items.length-1) return setIndex(0);
    setIndex(i + 1);
  }, [i]);
  const item = useMemo(() => items[i], [i]);
  return [item || items[0], prev, next];
}
```

추가 데이터 요청: README.md

```
/* ch08/proj/07-5/src/components/GithubUser.js */
import Fetch from "./Fetch";
import UserRepositories from "./UserRepositories";

const GithubUser = ({ login }) => (
  <Fetch
    uri={`https://api.github.com/users/${login}`}
    renderSuccess={userDetails} />
);

const userDetails = ({ data }) => (
  <div className="githubUser">
    <img src={data.avatar_url}
      alt={data.login} style={{ width: 200 }} />
    <div>
      <h1>{data.login}</h1>
      {data.name && <p>{data.name}</p>}
      {data.location && <p>{data.location}</p>}
    </div>
    <UserRepositories login={data.login} />
  </div>
);
export default GithubUser;
```



https://api.github.com/users/{login}

https://api.github.com/users/{login}/repos

https://api.github.com/repos/{login}/{repo}/readme

추가 데이터 요청: markdown

react-markdown: 마크다운 데이터를 html로 렌더링하는 라이브러리

https://github.com/remarkjs/react-markdown

```
/* ch08/proj/07-5/src/components/UserRepositories.js */
const UserRepositories = ({ login }) => (
 <Fetch
  uri={`https://api.github.com/users/${login}/repos`}
  renderSuccess={ ({data}) => (
   <RepoMenu repositories={data} login={login} /> )} />
);
export default UserRepositories;
```

```
/* ch08/proj/07-5/src/components/RepoMenu.js */
const RepoMenu = ( { repositories, login }) => {
 const [{name}, previous, next] = useIterator(repositories);
 return (
  <>
   <div style={{ display: "flex " }}>
    <button onClick={previous}>&lt;</button>
    <p>{name}</p>
    <button onClick={next}>&gt;</button>
   </div>
   <RepositoryReadme login={login} repo={name} />
  </> );
}
export default RepoMenu;
```

repo 목록

repo 별 readme 마크다운

```
/* ch08/proj/07-5/src/components/RepositoryReadme.js */
import ReactMarkdown from "react-markdown";
const RepositoryReadme = ({ repo, login }) => {
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState();
  const [markdown, setMarkdown] = useState("");

  const loadReadme = useCallback( async (login, repo) => {
    setLoading(true);
    const uri = `https://api.github.com/repos/${login}/${repo}/readme`;
    const { download_url } = await fetch(uri).then(res => res.json());
    const md = await fetch(download_url).then(res => res.text());
    setMarkdown(md);
    setLoading(false);
  }, []);

  useEffect(() => {
    if (!repo || !login) return;
    loadReadme(login, repo).catch(setError);
  }, [repo]);

  if (error) return <pre>{JSON.stringify(error, null, 2)}</pre>;
  if (loading) return <p>loading...</p>;
  return (
    <ReactMarkdown>{markdown}</ReactMarkdown>
  );
};
export default RepositoryReadme;
```

> npm i react-markdown

학습 정리: 8장. 데이터 포함시키기

- 데이터 요청하기
  - `fetch()`: `promise`, `async/await`
  - `XMLHttpRequest()`, `axios.get()`
- React App에서의 데이터 요청
  - `useState()`, `useEffect()` 활용
- 웹 스토리지와 활용 예
- Promise 상태 처리
- 렌더 프롭
- 대량 데이터 렌더링
  - 가상화 리스트
- 데이터 요청 훅과 컴포넌트
  - `useFetch()` 훅
  - `<Fetch />` 컴포넌트
- 여러 요청 처리하기

Appendix. github API: account 정보와 repository 정보

useFetch() 커스텀 훅
<Fetch /> 컴포넌트

https://api.github.com/users/{login}


```
{
  "login": "sp-mcslab",
  "id": 90921202,
  "node_id": "MDQ6VXNlcjkwOTIxMjAy",
  "avatar_url": "https://avatars.githubusercontent.com/u/90921202?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/sp-mcslab",
  "html_url": "https://github.com/sp-mcslab",
  "...": "...",
  "type": "User",
  "site_admin": false,
  "name": "Seunghyun Park",
  "company": "Hansung University",
  "blog": "http://cse.hansung.ac.kr",
  "location": "Seoul, Korea",
  "email": null,
  "hireable": null,
  "bio": "Assistant Professor in Hansung University",
  "twitter_username": null,
  "public_repos": 5,
  "public_gists": 0,
  "followers": 0,
  "following": 0,
  "created_at": "2021-09-17T15:23:54Z",
  "updated_at": "2023-10-17T12:32:54Z"
}
```


https://api.github.com/users/{login}/repos


```
[
  {
    "id": 601425202,
    "node_id": "R_kgDOIRwGXw",
    "name": "sp-mcslab.github.io",
    "full_name": "sp-mcslab/sp-mcslab.github.io",
    "private": false,
    "owner": { "..." : "..." },
    "html_url": "https://github.com/sp-mcslab",
    "description": null,
    "fork": false,
    "url": "https://api.github.com/users/sp-mcslab",
    "...": "..."
  },
  {
    "id": 555484937,
    "node_id": "R_kgDOIRwHCQ",
    "name": "wf1_ch08_sample2",
    "full_name": "sp-mcslab/wf1_ch08_sample2",
    "private": false,
    "owner": { "...": "..." },
    "html_url": "https://github.com/sp-mcslab/wf1_ch08_sample2",
    "description": null,
    "fork": false,
    "url": "https://api.github.com/repos/sp-mcslab/wf1_ch08_sample2",
    "...": "..."
  },
  { "...": "..." },
  { "...": "..." },
  { "...": "..." }
]
```

Appendix. github API: repository의 readme 정보

https://api.github.com/repos/{login}/{repository}/readme


```
{
  "name": "README.md",
  "path": "README.md",
  "sha": "1eb3de0c9275e31a3a6e0eae4133f677a1d3530e",
  "size": 29,
  "url": "https://api.github.com/repos/sp-mcslab/wf1_ch08_sample1/contents/README.md?ref=master",
  "html_url": "https://github.com/sp-mcslab/wf1_ch08_sample1/blob/master/README.md",
  "git_url": "https://api.github.com/repos/sp-mcslab/wf1_ch08_sample1/git/blobs/1eb3de0c9275e31a3a6e0eae4133f677a1d3530e",
  "download_url": "https://raw.githubusercontent.com/sp-mcslab/wf1_ch08_sample1/master/README.md",
  "type": "file",
  "content": "IyByZWFkbWUgYXQgd2YxX2NoMDhfc2FtcGxlMQo=\n",
  "encoding": "base64",
  "_links": {
    "self": "https://api.github.com/repos/sp-mcslab/wf1_ch08_sample1/contents/README.md?ref=master",
    "git": "https://api.github.com/repos/sp-mcslab/wf1_ch08_sample1/git/blobs/1eb3de0c9275e31a3a6e0eae4133f677a1d3530e",
    "html": "https://github.com/sp-mcslab/wf1_ch08_sample1/blob/master/README.md"
  }
}
```


```
# readme at wf1_ch08_sample1
```

https://raw.githubusercontent.com/{login}/{repository}/master/README.md