# React - 비동기 통신

## 목표

React App 에서 비동기 통신을 이용할 수 있어야 합니다. !

## 강의시간

4시간

## 전체 소스코드

<https://github.com/egoingsb/react-component-tutorial/tree/ajax>

## 선행지식

### AJAX & fetch API

<https://seomal.com/map/1/13>

### Create React App Proxy

<https://seomal.com/map/1/80>

### Promise

<https://seomal.com/map/1/185>

### async & await

<https://seomal.com/map/1/188>

### rest api & json-server

<https://seomal.com/map/1/104>

온라인 수업

### React & Ajax

<https://seomal.com/map/1/73>

## 데이터베이스 셋팅

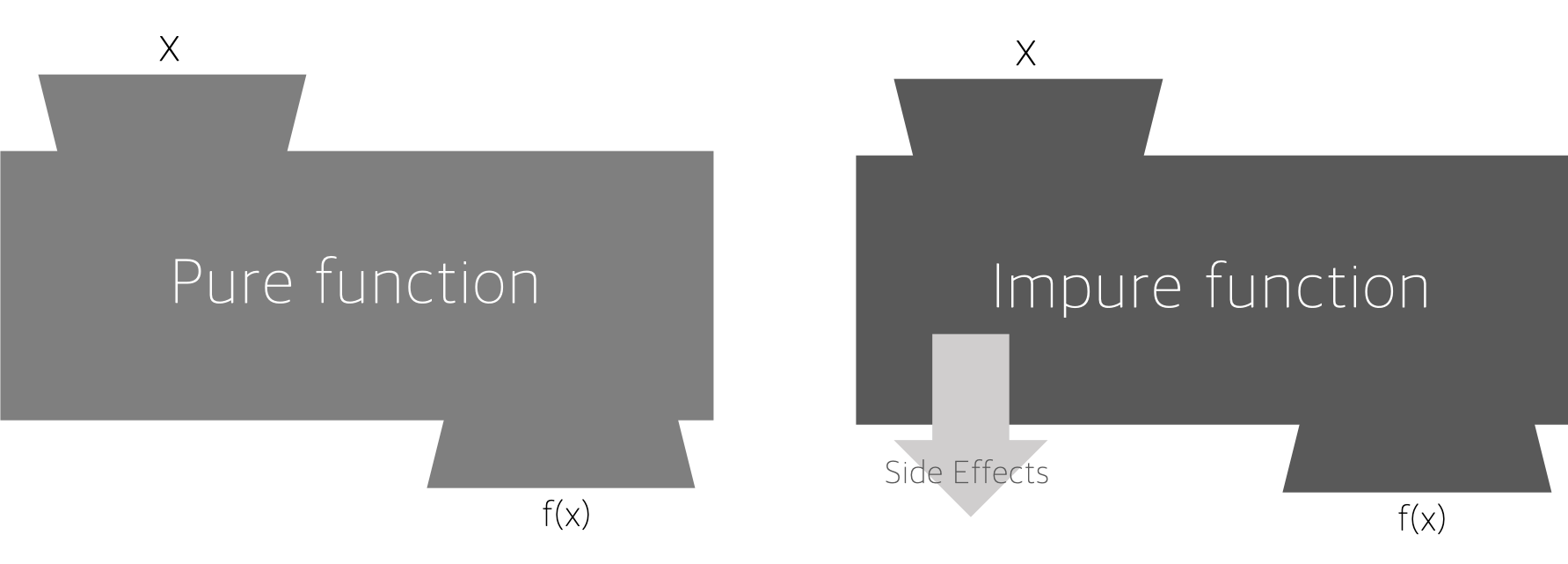
npx json-server –watch –port 3001 db.json

### db.json 기본내용

| {  "topics": [  {  "id": 1,  "title": "html",  "body": "html is ..."  },  {  "id": 2,  "title": "css",  "body": "css is ..."  },  {  "id": 3,  "title": "js",  "body": "js is ..."  }  ] } |
| --- |

## useEffect

사이드 이팩트를 처리하기 위한 훅, 순수하지 않은 함수를 처리합니다.



### 순수함수

언제나 같은 결과나 나오는 함수

| function add(a,b){  return a+b; } console.log(add(1,2)); |
| --- |

비순수함수

언제가 같은 결과를 나타내지 않습니다. b에 의존하고 있습니다.

| b = 2; function add(a){  return a+b; } console.log(add(1)) b=3; console.log(add(1)) |
| --- |

순수하지 않은 함수를 컴포넌트에가 실행할 때 useEffect를 사용합니다.

### useEffect 의 응용

#### 글목록 기능 구현

useEffect와 fetch 를 이용해서 nav에서 사용할 데이터를 비동기 방식으로 로딩하고 있습니다.

<https://github.com/egoingsb/react-component-tutorial/commit/a718de8d6597c7cd90697b730889f48f2dc669f0>

| function App() {  let [topics, setTopics] = useState([]);  let navigate = useNavigate();  let [nextId, setNextId] = useState(4);  useEffect(()=>{  let callback = async ()=>{  const request = await fetch('/topics');  const response = await request.json();  setTopics(response);  }  callback();  }, []);  function createHandler(data){  let newTopics = [...topics];  newTopics.push({id:nextId, title:data.title, body:data.body});  setTopics(newTopics);  navigate('/read/'+nextId);  setNextId(nextId+1);  }  function updateHandler(data){  let newTopics = [...topics];  for(let i=0; i<newTopics.length; i++){  let t = newTopics[i];  if(t.id === Number(data.id)){  t.title = data.title;  t.body = data.body;  break;  }  }  setTopics(newTopics);  navigate('/read/'+data.id);  }  function deleteHandler(id){  let newTopics = [];  for(let i=0; i<topics.length; i++){  if(topics[i].id !== Number(id)){  newTopics.push(topics[i]);  }  }  setTopics(newTopics);  navigate('/');  }  return (  <>   <**Header**></**Header**>  <**Nav** topics={topics}></**Nav**>  <**Routes**>  <**Route** path="/" element={<Article title="Welcome" body="Hello, WEB"></**Article**>}></**Route**>  <**Route** path="/read/:id" element={<Read topics={topics}></**Read**>}></**Route**>  <**Route** path="/create" element={<Create onCreate={createHandler}></**Create**>}></**Route**>  <**Route** path="/update/:id" element={<Update topics={topics} onUpdate={updateHandler}></**Update**>}></**Route**>  </**Routes**>  <**Routes**>  <**Route** path="/" element={<Control></**Control**>}></**Route**>  <**Route** path="/read/:id" element={<Control onDelete={deleteHandler}></**Control**>}></**Route**>  </**Routes**>  </>  ); } |
| --- |

#### 글 읽기 기능 구현

<https://github.com/egoingsb/react-component-tutorial/commit/807f098d16a41a0e6b8c272f9ce25b731b7f46ef>

function Read(props){

let params = useParams();

let [title, setTitle] = useState(null);

let [body, setBody] = useState(null);

useEffect(()=>{

let callback = async ()=>{

const request = await fetch('/topics/'+params.id);

const response = await request.json();

setTitle(response.title);

setBody(response.body);

}

callback();

}, [params.id]);

return <Article title={title} body={body}></Article>

}

#### 글쓰기 기능 구현

<https://github.com/egoingsb/react-component-tutorial/commit/15edf8f83517aeaeb0539e66b1b5b644db4f2dc5>

| function Create(props){  return <article>  <form onSubmit={async event=>{  event.preventDefault();  let t = event.target.title.value;  let b = event.target.body.value;  let request = await fetch('/topics', {  method:'POST',   headers:{  'Content-Type':'application/json'  },   body:JSON.stringify({title:t, body:b})  })  let response = await request.json();  props.onCreate({id: response.id, title:t, body:b});  }}>  <p><input type="text" name="title" placeholder="Title" /></p>  <p><textarea name="body" placeholder="Body"></textarea></p>  <p><input type="submit" value="Create" /></p>  </form>  </article> } |
| --- |

| let refreshTopics = async ()=>{  const request = await fetch('/topics');  const response = await request.json();  setTopics(response); } function createHandler(data){  navigate('/read/'+data.id);  refreshTopics();  } |
| --- |

#### update 기능구현

<https://github.com/egoingsb/react-component-tutorial/commit/1af51e3e56c2784db90f9078456fb08ad4f1dfcb>

| function Update(props){  let params = useParams();  let \_title, \_body = '';  for(let i=0; i<props.topics.length; i++){  let t = props.topics[i];  if(t.id === Number(params.id)){  \_title = t.title;  \_body = t.body;  break;  }  }  let [title, setTitle] = useState(\_title);  let [body, setBody] = useState(\_body);  return <article>  <form onSubmit={async event=>{  event.preventDefault();  let t = event.target.title.value;  let b = event.target.body.value;  let request = await fetch('/topics/'+params.id, {  method:'PUT',   headers:{  'Content-Type':'application/json'  },   body:JSON.stringify({title:t, body:b})  })  let response = await request.json();  props.onUpdate({id:params.id, title:t, body:b});  }}>  <p><input type="text" name="title" placeholder="Title" value={title} onChange={e=>{setTitle(e.target.value)}} /></p>  <p><textarea name="body" placeholder="Body" value={body} onChange={e=>{setBody(e.target.value)}}></textarea></p>  <p><input type="submit" value="Create" /></p>  </form>  </article> } |
| --- |

| function updateHandler(data){  navigate('/read/'+data.id);  refreshTopics();  } |
| --- |

#### 삭제 기능 구현

| async function deleteHandler(id){  let request = await fetch('/topics/'+id, {  method:'DELETE'  })  let response = await request.json();  navigate('/');  refreshTopics();  } |
| --- |