React

Lesson 4

Simple API server

- Install json-server via yarn:

```
yarn add json-server
```

- Create a db.json file with empty data for visits:

```
echo '{"visits": []}' > db.json
```

(not sure if works on Windows)

- Add db.json to .gitignore
- Run server on a custom port:

```
yarn json-server --watch db.json --port 3001
```

Sending requests

`fetch(url, {method, headers, ...})` - returns Promise that resolves to Response

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

Processing response

```
fetch(...)
.then(response => response.json())
.then(data => doSomething(data))
```

New hooks

`useEffect` https://reactjs.org/docs/hooks-effect.html

`useCallback` https://reactjs.org/docs/hooks-reference.html#usecallback

Loading initial data

```
useEffect(() => {
  loadData()
}, [loadData])

// memoized function
const loadData = useCallback(() => {
  // fetch data and add it to state
}, [])
```

Handling loading state

- Set loading state to true before doing request
- Set loading state to false after processing response

Error handling

```
fetch(...)
    .then(resp => resp.ok ?
        resp.json() :
        Promise.reject(“Error”)
    )
    .then(data => doSomethingWithData(data))
    .catch(err => doSomethingWithError(err))
    .finally(() => doSomethingCommon())
```

Optimistic update

- Update the state without waiting for response
- Revert the state in case of an error

Promise chaining vs async / await

```
funcA()
 .then(processData)
   .catch(processError)
 .finally(doSomethingCommon)

try {
  const data = await funcA()
  processData(data)
} catch (error) {
  processError(error)
} finally {
  doSomethingCommon()
}
```

Home task

Set up json-server (or any other server with database of your choice).

Implement API methods (it's ok to copy api.ts from demo after understanding what it does)

Fetch and edit data with API.

Display loading state while request is pending.

Indicate error in case of failed request.