AFS

Agile Full Stack
Developer Bootcamp

Thoughtworks ©

React router

thoughtworks

React Router

thoughtworks

React Router keeps the UI consistent with the URL. When the route changed, it will render different components.

It provides API such as: **Route, BrowserRouter, Link** and so on. We can use these to define the URL path of our application.

Before we use router in our application, we need to install the **react-router-dom** package:

```
npm install react-router-dom
```

https://reactrouter.com/docs/en/v6/getting-started/tutorial

AFS
Agile Full Stack
Developer Bootcamp

Thoughtworks ©

React router Demo

thoughtworks

React Router Practice

1. Navigation Links
2. Homepage(TodoList)
3. Help Page
4. Done List Page
5. 404 Page

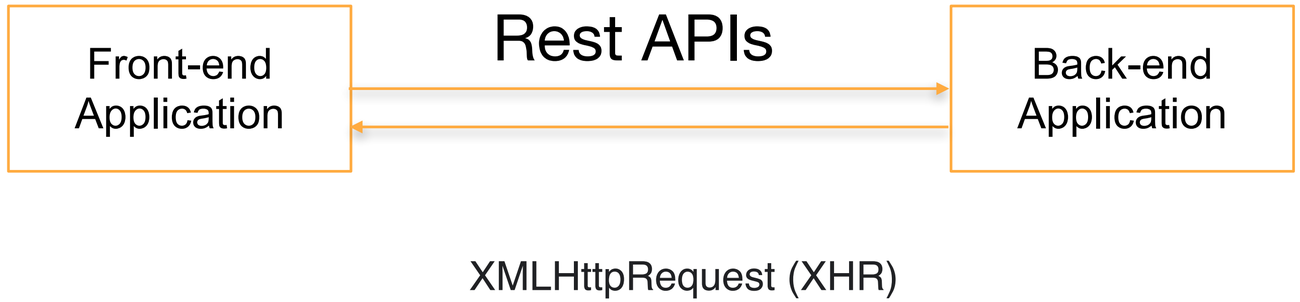
AFS
Agile Full Stack
Developer Bootcamp

Thoughtworks ©

Integrate Front-end and Back-end

thoughtworks

Integration between Front-end and Back-end



XMLHttpRequest (XHR)

```
(function () {
  const API_BASE_URL = 'https://pokeapi.co/api/v2';
  let pokemonInfo = null;
  let moveInfo = null;
  let machineInfo = null;
  let itemInfo = null;

  const pokemonXHR = new XMLHttpRequest();
  pokemonXHR.responseType = 'json';
  pokemonXHR.open('GET', `${API_BASE_URL}/pokemon/1`);
  pokemonXHR.send();

  pokemonXHR.onload = function () {
    pokemonInfo = this.response

    const moveXHR = new XMLHttpRequest();
    moveXHR.responseType = 'json';
    moveXHR.open('GET', pokemonInfo.moves[0].move.url);
    moveXHR.send();

    moveXHR.onload = function () {
      moveInfo = this.response;

      const machineXHR = new XMLHttpRequest();
      machineXHR.responseType = 'json';
      machineXHR.open('GET', moveInfo.machines[0].machine.url);
      machineXHR.send();

      machineXHR.onload = function () {
        machineInfo = this.response;

        const itemXHR = new XMLHttpRequest();
        itemXHR.responseType = 'json';
        itemXHR.open('GET', machineInfo.item.url);
        itemXHR.send();

        itemXHR.onload = function () {
          itemInfo = this.response;

          console.log('Pokemon', pokemonInfo);
          console.log('Move', moveInfo);
          console.log('Machine', machineInfo);
          console.log('Item', itemInfo);
        }
      }
    }
  }
})();
```

Examples

XMLHttpRequest

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function () {
  if (this.readyState === 4 && this.status === 200) {
    console.log(xhttp.response);
  }
};
xhttp.open("GET", "https://5d6f7991482b530014d2e376.mockapi.io/api/todos", true);
xhttp.send();
```

fetch

```
fetch('https://5d6f7991482b530014d2e376.mockapi.io/api/todos')
    .then(response => {
      return response.json()
    }).then(responseJson => {
      console.log(responseJson)
    })
```

axios

```
axios.get('https://5d6f7991482b530014d2e376.mockapi.io/api/todos')
    .then(response => {
      console.log(response)
    })
```

axios and Promise

axios

- axios is a Promise based HTTP client for the browser and Node.js

Promise

- The **`Promise`** object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

A Promise is in one of these states:

- *pending*: initial state, neither fulfilled nor rejected.
- *fulfilled* (resolved): meaning that the operation was completed successfully.
- *rejected*: meaning that the operation failed.

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

Promise

```
const promise = new Promise((resolve, reject) => {
    const count = 1+1;
    if (count == 2) {
        resolve('Success')
    } else {
        reject('Failed')
    }
})


promise.then((response) => {
    console.log('This is in the then: resolved' + response);
}).catch((response) => {
    console.log('This is in the catch: reject' + response)
})
```

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

How to use axios

Using npm:

```
$ npm install axios
```

```
const instance = axios.create({
  baseURL: 'https://5d6f7991482b530014d2e376.mockapi.io/api/',
});
```

```
instance.get('/users')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

```
instance.post('users', data)
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

https://github.com/axios/axios

Todo APIs (mock)

https://mockapi.io

GET https://5d6f7991482b530014d2e376.mockapi.io/api/todos

POST https://5d6f7991482b530014d2e376.mockapi.io/api/todos

```
{
    "text": "new todo item"
}
```

DELETE https://5d6f7991482b530014d2e376.mockapi.io/api/todos/1

PUT https://5d6f7991482b530014d2e376.mockapi.io/api/todos/1

```
{
    "done": true
}
```

Practice

Call the mock APIs to :

1. get all todos
2. add a new todo
3. delete a todo
4. toggle a todo’s status

GET https://5d6f7991482b530014d2e376.mockapi.io/api/todos

POST https://5d6f7991482b530014d2e376.mockapi.io/api/todos

```
{
    "text": "new todo item"
}
```

DELETE https://5d6f7991482b530014d2e376.mockapi.io/api/todos/1

PUT https://5d6f7991482b530014d2e376.mockapi.io/api/todos/1

```
{
    "done": true
}
```

UI component library

thoughtworks

UI libraries

- Ant Design
- Material UI
- React Bootstrap
- …

Ant Design

Install `antd` from yarn or npm:

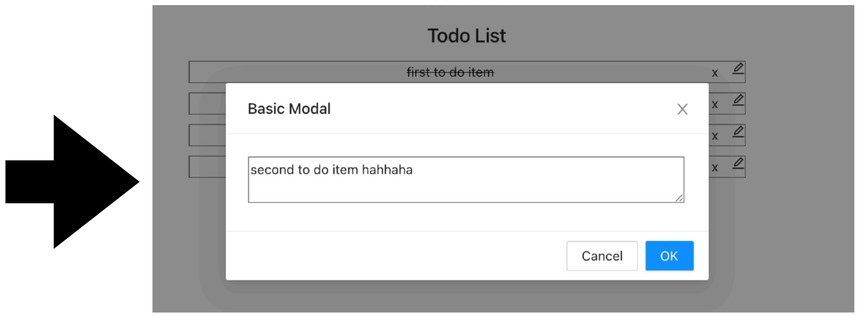
```
$ npm install antd
```

https://ant.design/docs/react/introduce

Homework

Deadline 22:00

- Finish the 4 APIs, and sync data to your store，with baby step commit.
  - get all todos
  - add a new todo
  - update a todo text
  - update a todo status
  - delete a todo



- Use Antd to make your page beautiful（beauty pageant tomorrow）
- Use Router to create at least 3 pages (TodoListPage, DonePage, 404Page)
- ORID Diary
- Submit the concept map on Agile Activity

Thank You !

thoughtworks