WHAT I LEARNED DURING THE PAST MONTH

AGENDA

- Overview of React
- React-Redux
- React Loadable
- Caching using Service Worker
- Server Side Rendering in React
- React-Router

REACT OVERVIEW

- React → makes creating interactive UI painless, by building components that have states and props

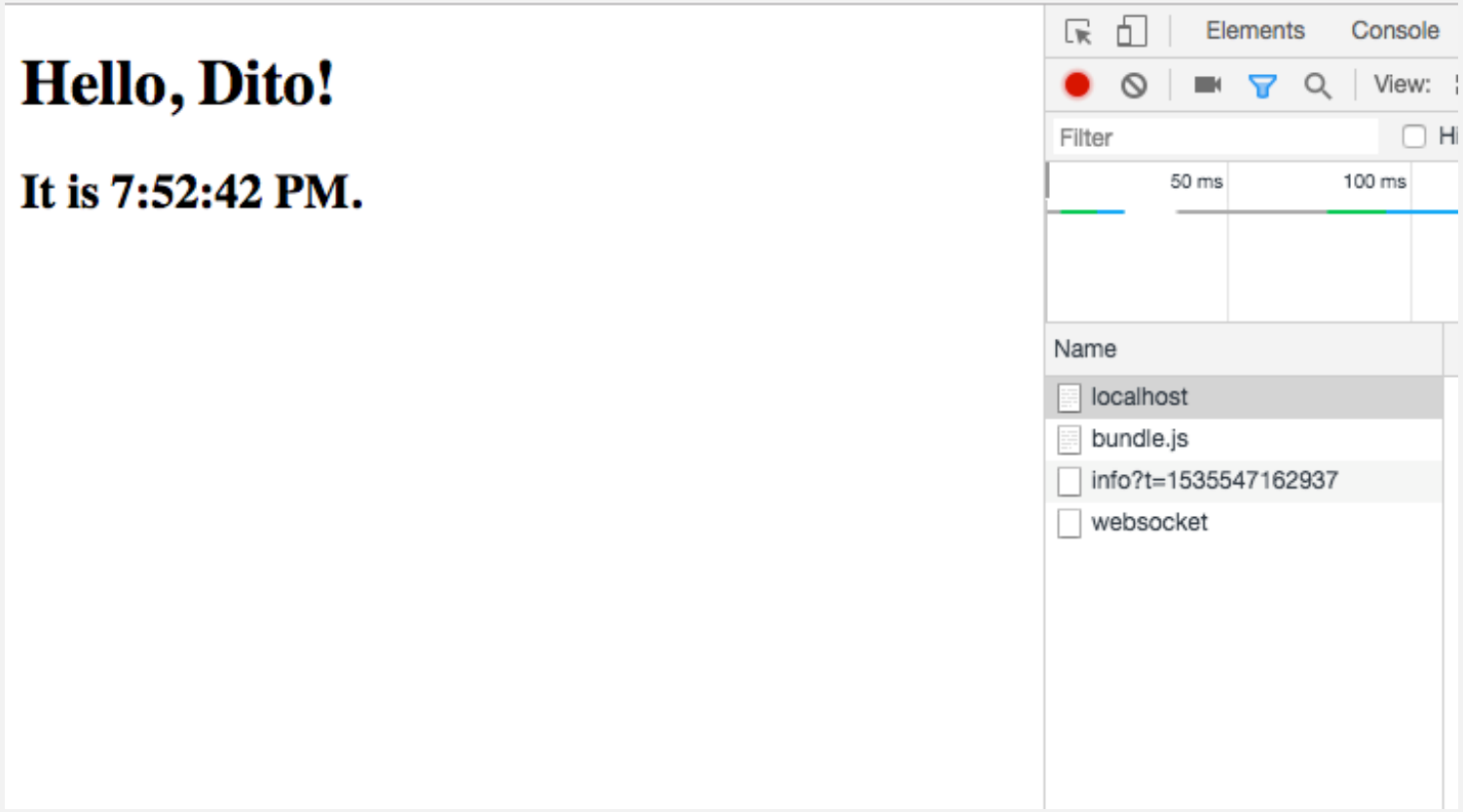
```
import React from 'react';
import ReactDOM from 'react-dom';
import Clock from './Clock';

ReactDOM.render(
    <Clock name="Dito"/>,
    document.getElementById('root')
);
```

index.js

```
import React, { Component } from 'react';

class Clock extends Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  render() {
    return (
      <div>
        <h1>Hello, {this.props.name}!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}

export default Clock;
```

Clock.js

For further reference: https://reactjs.org/

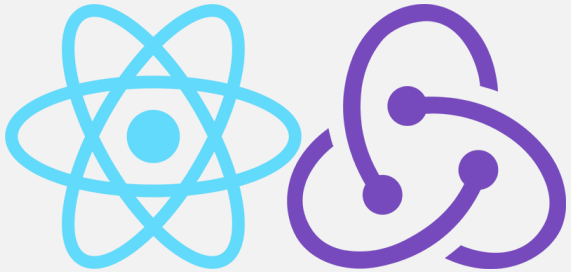


This example can be found on:

https://github.com/albertusandito-olx/simple-react-app

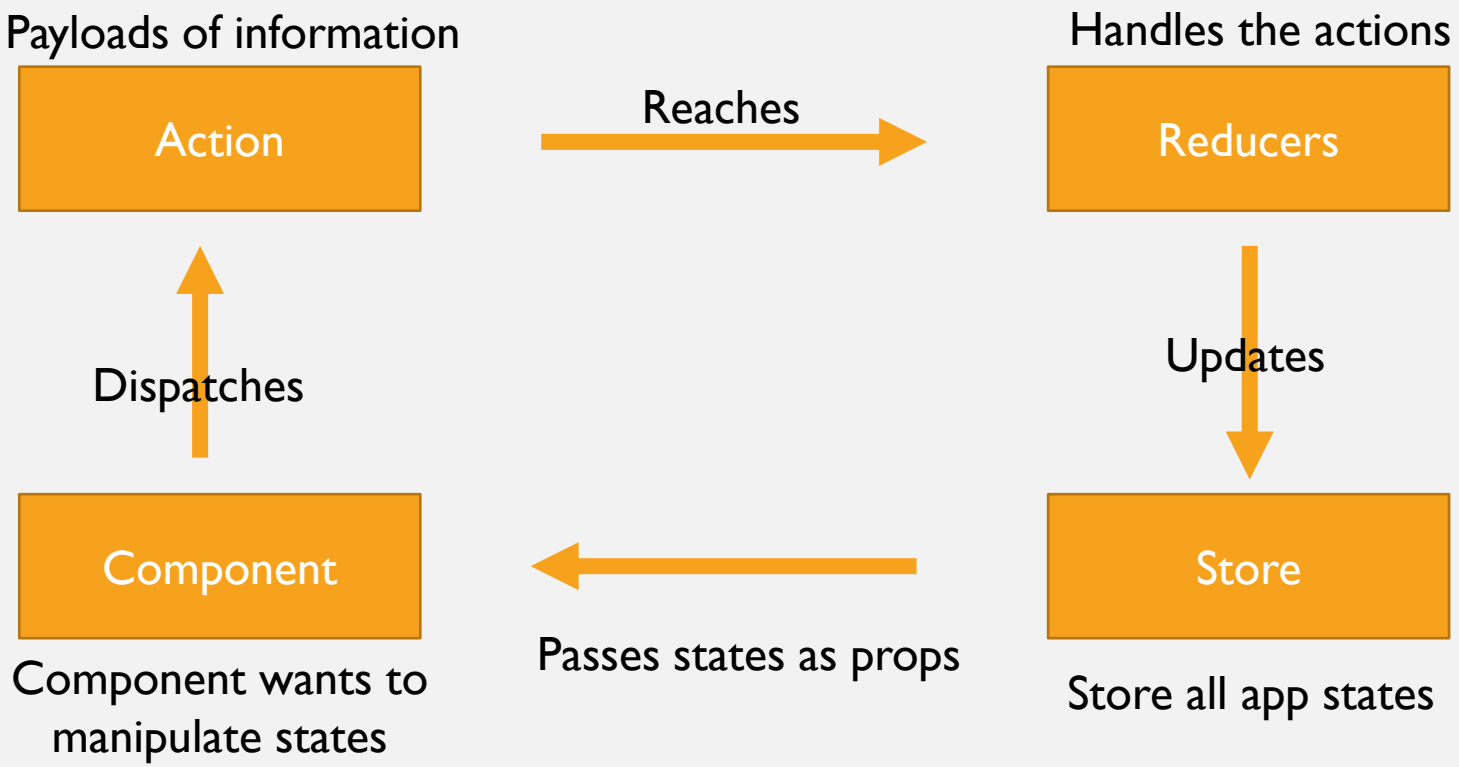
REACT-REDUX

- React app contains many states → **messy and difficult to manage**
- + With Redux, **state management is easy**!
- Only need one STORE to keep all of the states
- Redux uses ACTIONS to update the states, and REDUCER to handle the actions and connect to the store



For further reference: https://redux.js.org/basics

REDUX FLOWCHART



REACT LOADABLE

- React app's bundle is often so big, especially *Single Page Application* (SPA) → slow things down
- React-Loadable code-splits the app into several chunks.
- By using dynamic import, it only loads the components when needed.
- + Advantage? Faster! Lighten up the initial load!

For further reference: https://github.com/jamiebuilds/react-loadable

CACHING USING SERVICE WORKER

- Caching → Providing offline support!
- Caching is all done in the Service Worker
- Files are cached in the Cache Storage in Browser
- What to cache? App Shell!
- Multiple strategies to cache: I used network with cache fallback

For further reference: https://developers.google.com/web/ilt/pwa/caching-files-with-service-worker

REACT SERVER-SIDE RENDERING

- Normal React: send blank HTML to client → render on client
- React SSR: render on server → send full HTML to client
- Using Node and Express
- Pros:
  - + Could improve the SEO
  - + Displaying content to user faster
- Cons:
  - - More work for the server
  - - Can't be used with other languages other than Node (with Babel transpiler)

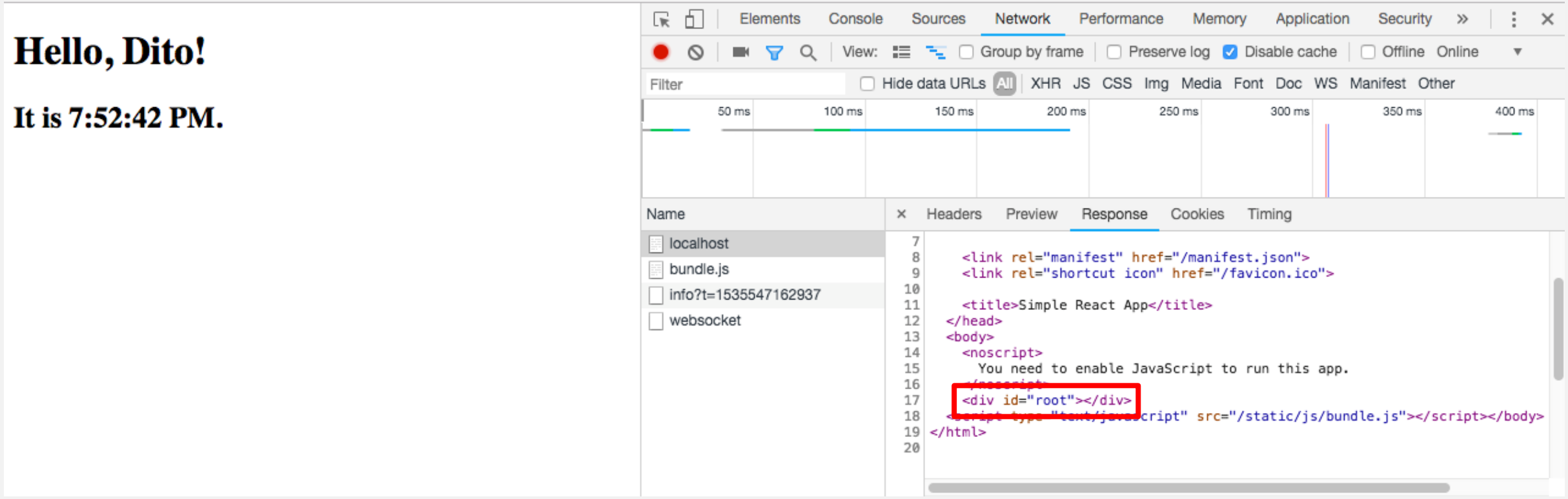
For further reference: https://naspers.udemy.com/server-side-rendering-with-react-and-redux/learn/v4/content

GITHUB REPO LINK

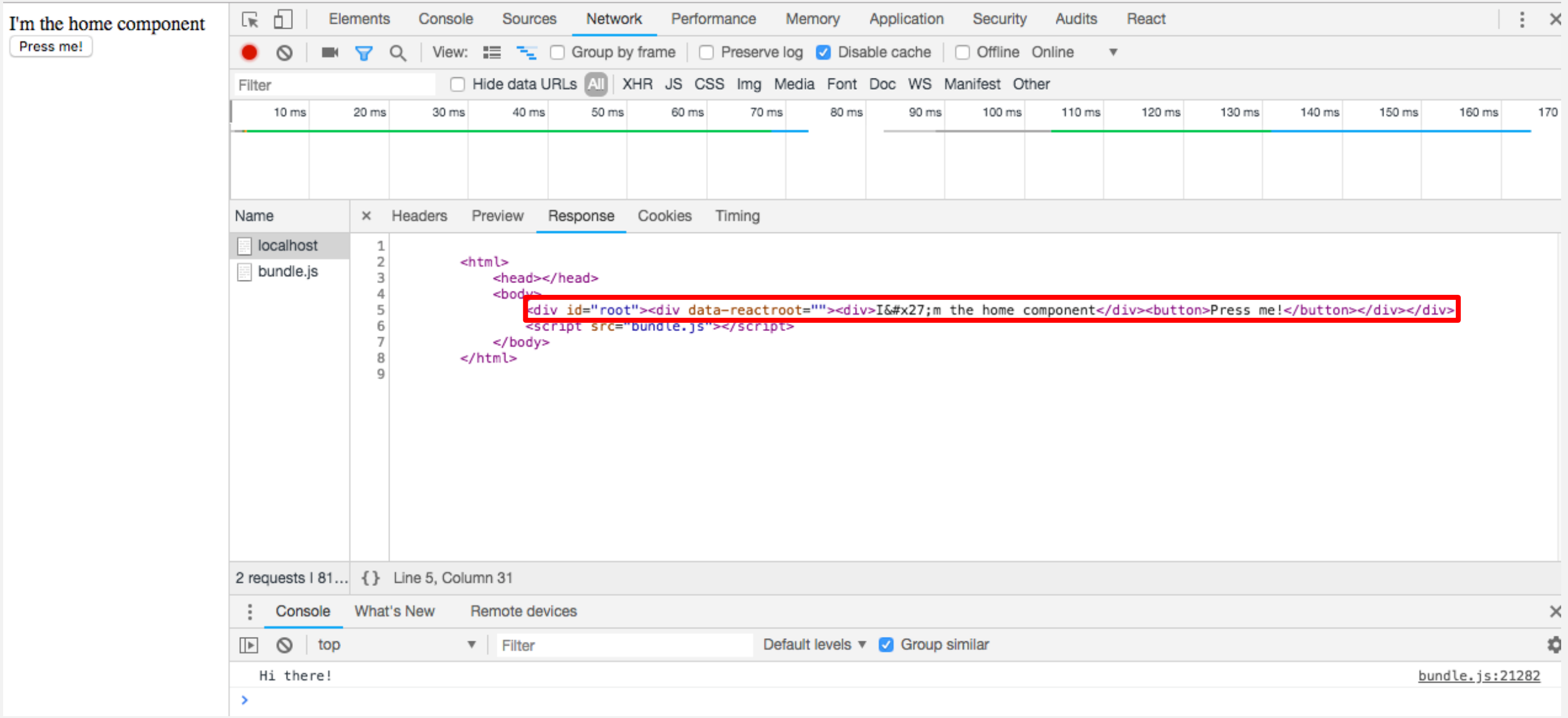
The following example can be found on:

https://github.com/albertusandito-olx/react-ssr-example

NORMAL REACT



REACT SERVER-SIDE RENDERING



SERVER – index.js

```
import path from 'path';
import fs from 'fs';

import express from 'express';
import React from 'react';
import { renderToString } from 'react-dom/server';
import Home from './client/components/Home';

const app = express();

app.use(express.static('public'));

app.get('/', (req, res) => {
    const content = renderToString(<Home />);

    const indexFile = path.resolve('./public/index.html');
    fs.readFile(indexFile, 'utf8', (data) => {
        return res.send(
            data.replace('<div id="root"></div>', `<div id="root">${content}</div>`)
        );
    });
});

app.listen(3000, () => {
    console.log('Listening on port 3000.');
});
```

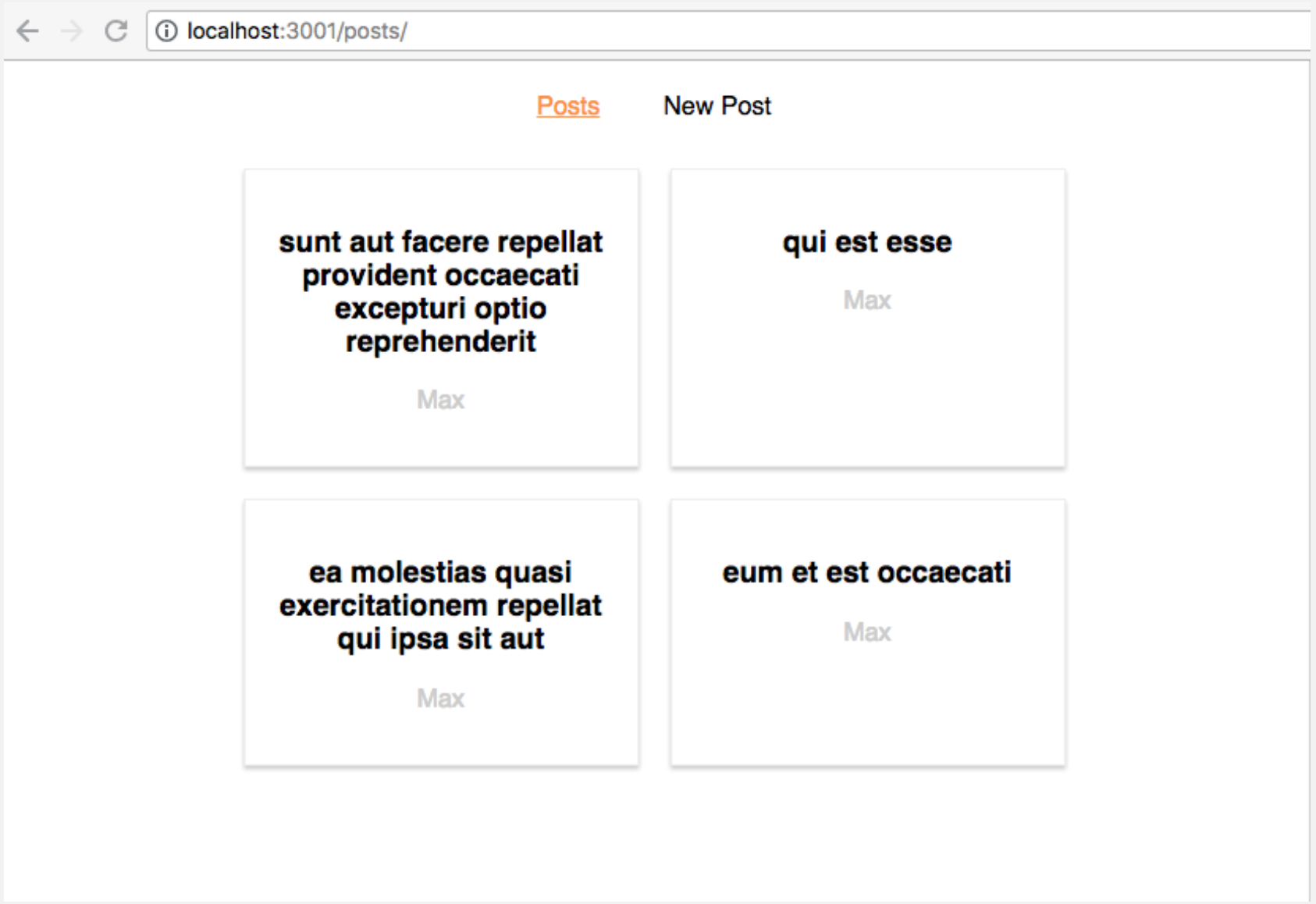
CLIENT – client.js

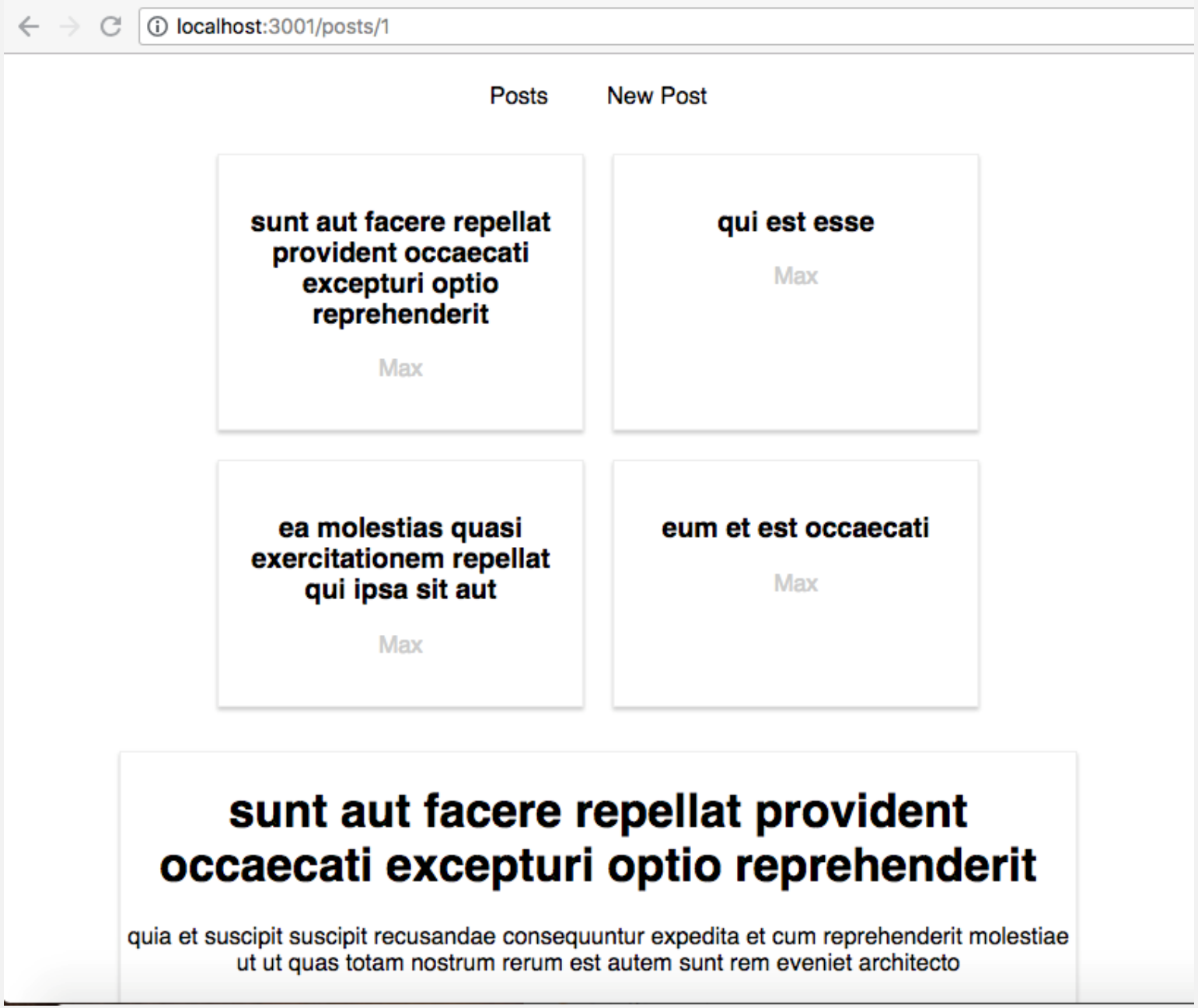
```
// Startup point for the client side application

import React from 'react';
import ReactDOM from 'react-dom';
import Home from './components/Home';

ReactDOM.hydrate(<Home />, document.querySelector('#root'));
```

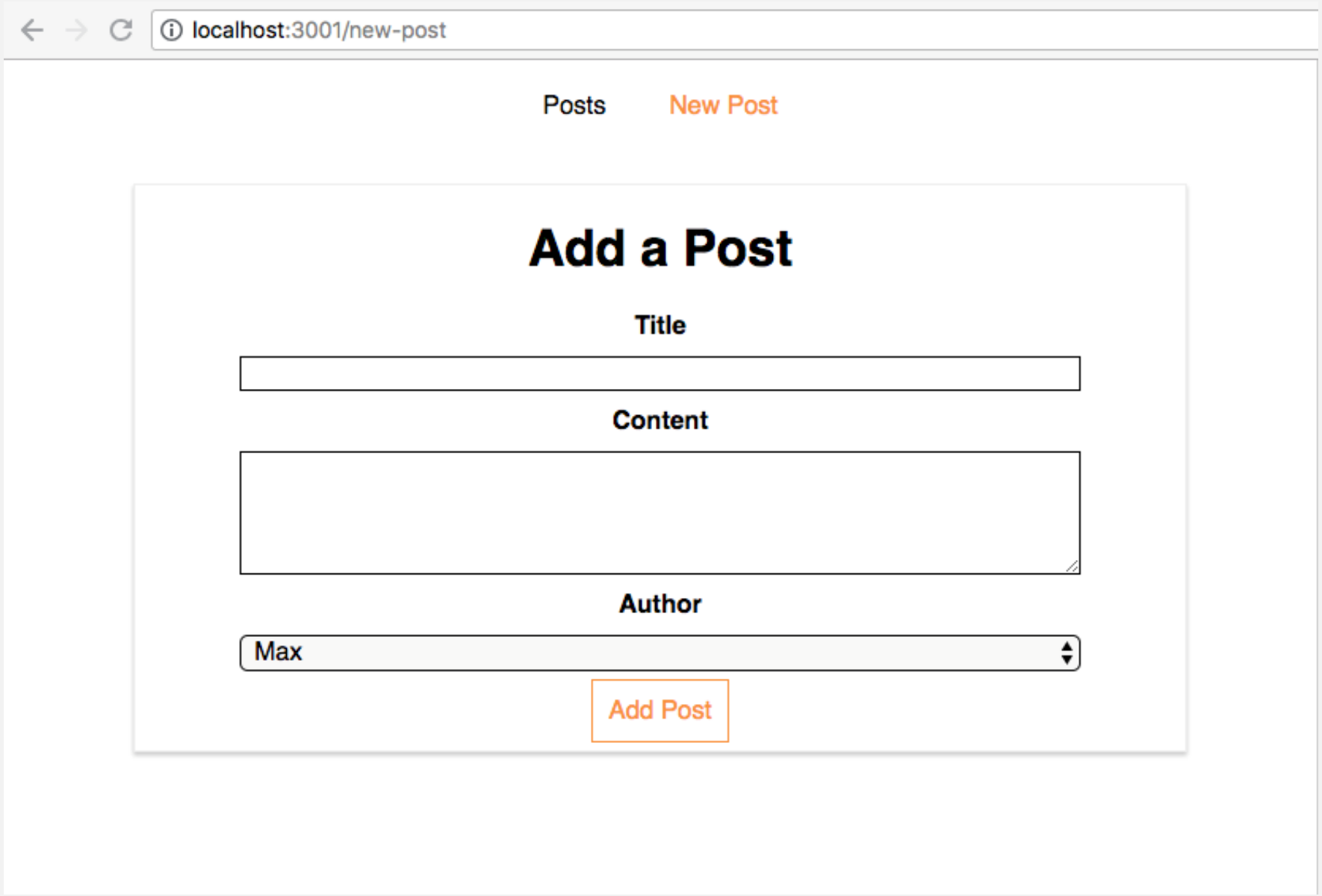
REACT-ROUTER

- Beneficial for SPA → gives Multi-Page-Feeling
- Navigating in SPA ≠ go to new page → the views load within the same page
- + the URL can be updated, reflecting the views
- + user can use the browser's back and forward buttons
- + easy to handle nested views because it's component based

For further reference: https://reacttraining.com/react-router/


localhost:3001/posts/
Posts
New Post
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
Max
qui est esse
Max
ea molestias quasi exercitationem repellat qui ipsa sit aut
Max
eum et est occaecati
Max


localhost:3001/posts/1
Posts
New Post
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
Max
qui est esse
Max
ea molestias quasi exercitationem repellat qui ipsa sit aut
Max
eum et est occaecati
Max
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit suscipit recusandae consequuntur expedita et cum reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto



The page isn't reloaded!!!

App.js

```
import React, { Component } from 'react';
import { BrowserRouter } from 'react-router-dom';

import Blog from './containers/Blog/Blog';

class App extends Component {
  render() {
    return (
        <BrowserRouter>
          <div className="App">
            <Blog />
          </div>
        </BrowserRouter>
    );
  }
}

export default App;
```

Blog.js

```
import React, { Component } from 'react';
import { Route, NavLink, Switch, Redirect } from 'react-router-dom';

import './Blog.css';
import Posts from './Posts/Posts';
import NewPost from './NewPost/NewPost';

class Blog extends Component {
    state = {
        auth: true
    }

    render () {
        return (
            <div className="Blog">
                <header>
                    <nav>
                        <ul>
                            <li><NavLink
                                to="/posts/"
                                exact
                                activeClassName="my-active"
                                activeStyle={{
                                    color: '#fa923f',
                                    textDecoration: 'underline'
                                }}>Posts</NavLink></li>
                            <li><NavLink to="/new-post">New Post</NavLink></li>
                        </ul>
                    </nav>
                </header>
```

Blog.js

```
                    <Switch>
                        {this.state.auth ? <Route path="/new-post" component={NewPost} /> : null}
                        <Route path="/posts" component={Posts} />
                        <Redirect from="/" to="/posts" />
                    </Switch>
                </div>
            );
        }
}

export default Blog;
```

Posts.js

```
render () {
    let posts = <p style={{textAlign: 'center'}}>Something went wrong!</p>;
    if (!this.state.error) {
        posts = this.state.posts.map(post => {
            return (
                <Link to={'/posts/' + post.id} key={post.id} >
                    <Post
                    key={post.id}
                    title={post.title}
                    author={post.author}
                    />
                </Link>
            );
        });
    }

    return (
        <div>
            <section className="Posts">
                    {posts}
            </section>
            <Route path={this.props.match.url + '/:id'} exact component={FullPost} />
        </div>

    );
};
```

THANK YOU.
ANY QUESTIONS?