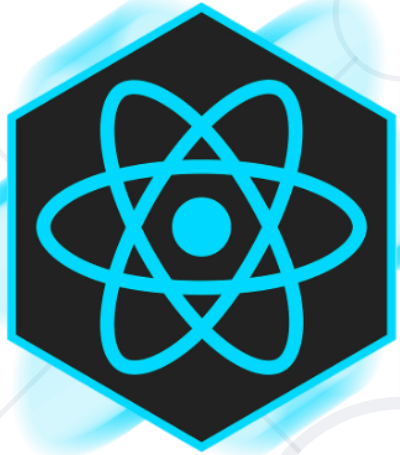


React – Routing

Single Page Applications, Blueprint for SPA



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://softuni.bg>

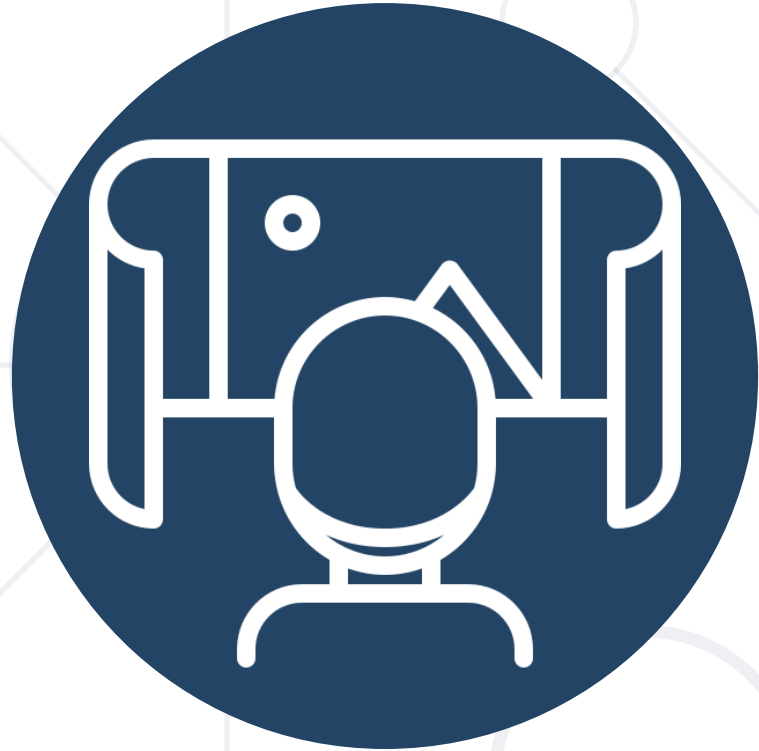
Table of Contents

1. Virtual DOM
2. Routing Overview
3. React Router
 - Installation, Links, Redirects and etc.
4. React Lazy & Suspense



sli.do

#js-web



Virtual DOM

Virtual DOM

- The virtual DOM (**VDOM**)
 - Virtual representation of a UI is kept in the **memory**
 - Synced the real DOM by a library such as **ReactDOM**
 - The term **Virtual DOM** is usually associated with React elements
 - They are the objects representing the UI



- React keeps track of all elements in a **virtual DOM**
 - On change, a **diffing algorithm** is applied
 - Only the **needed** parts are updated in the browser
- React syntax is **declarative**
 - You only describe the desired result
 - ReactDOM takes care of the **order** of operations



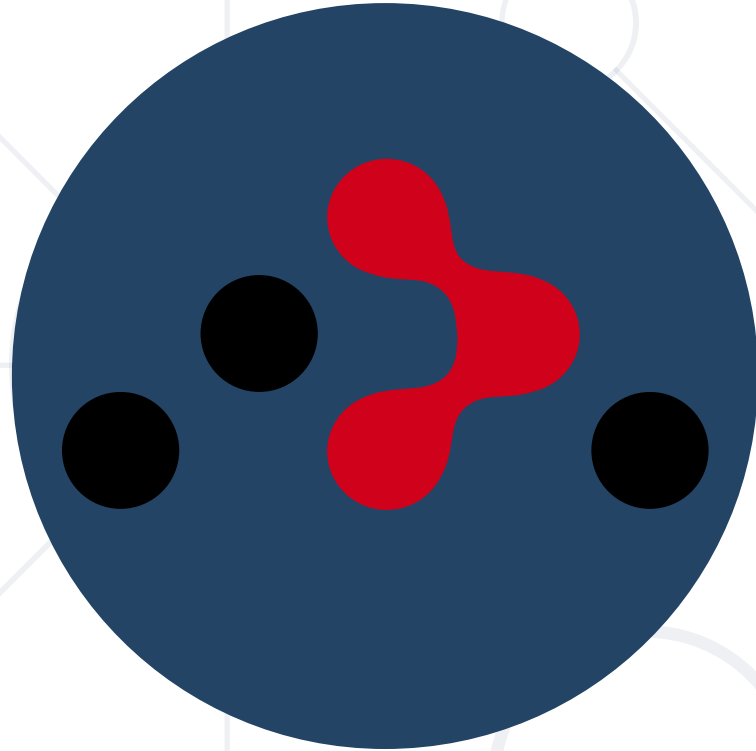
Routing Overview

Navigation for Single Page Apps

What is Client-side Routing?

- **Client-side routing** is internal handling of a route
- It's a pivotal element of writing **SPA's**
- Allows navigation, **without a full reloading of** the page
- Loading only the initial **HTML, CSS** and **JS**
- Gives better UX

- A **Router** loads the correct content when the **location changes**
- Change in content is reflected in the address bar
- Benefits
 - Load all scripts only once
 - Maintain state across multiple pages
 - Browser history can be used
 - Build User Interfaces that react quickly



React-Router

Routing Library Tailored for React

- **React Router** is an **API** for React applications
- Uses component structure

```
const App =(  
  <Routes>  
    <Route path="/catalog" element={<Catalog />} />  
    <Route path="/about" element={<About />} />  
  </Routes>);  
  
ReactDOM.render((  
  <BrowserRouter>  
    <App />  
  </BrowserRouter>  
>, document.getElementById('root'))
```

- Install using npm from the terminal

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

- **Route, Link, BrowserRouter, Routes** components helps to implement the routing

```
import {  
  BrowserRouter as Router,  
  Routes,  
  Route,  
  Link,  
} from 'react-router-dom';
```

- React components can be wrapped in a **Route** and bound to a specific path

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<App />} />
    <Route path="/catalog" element={<Catalog />} />
    <Route path="/about" element={<About />} />
  </Routes>
</BrowserRouter>
```

- Route path is exact by default
- Use * as a wildcard

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/about" element={<About />} />
  <Route path="/user/:id" element={<User />} />
  <Route path="*" element={<NotFound />} />
</Routes>
```

Default route

- **Link** replaces **<a>** and automatically prevents **page reload**

```
class App extends Component {  
  render() {  
    return (  
      <div>  
        <h1>React Router Tutorial</h1>  
        <Link to="/catalog">Catalog</Link>  
        <Link to="/about">About</Link>  
      </div>  
    );  
  }  
}
```

- Parameters are dynamic parts of the URL

```
/catalog/electronics/XYZ5538
```

- Configure the Route to work with params

```
<Route path="/catalog/:category/:userId"  
component={Catalog}/>
```

- Access from the component

```
const {category, userId} = useParams();
```


- The **location** object represents
 - Where the app is now
 - Where you want it to go
 - Where it was
- A **location** object is never mutated

- You can redirect the user by rendering a **Navigate** component
 - you can replace route state instead of push with replace attribute
- You can redirect with useNavigate hook

```
...  
if (condition) {  
  return <Navigate to="/home" />  
}
```

Determine the component to render at run-time

- **NavLink** knows when it's currently active
 - We can style them with **style**, **className** or **children**

```
<NavLink to="/catalog"
  style={({isActive}) =>
    isActive ? { color: 'red' } : {}>
  Catalog
</NavLink >
```

```
<NavLink to="/catalog"
  className={({isActive}) =>
    isActive ? activeStyle : undefined>
  Catalog
</NavLink >
```

- You can dynamically nest routes

```
const About = () => (  
  <div>  
    <h1>About Page</h1>  
    <Route  
      path="contact"  
      element={<Contact />}  
    />  
  </div>  
)
```



Lazy Loading

Code-Splitting, Bundling, React.lazy


Code-Splitting – Bundling

- Most React apps will have their files "**bundled**" using tools like **Webpack** or **Browserify**
- Bundling is the process of
 - Following **imported** files
 - Merging them into a **single file** (bundle)



Code-Splitting Bundling

- The bundle can be included on a webpage to **load** an entire app **at once**



```
export function add(a, b) {  
  return a + b;  
}
```

```
import { add } from './math.js';  
console.log(add(16, 26));
```

```
function add(a, b) {  
  return a + b;  
}  
console.log(add(16, 26));
```

- The best way to introduce **code-splitting** into your app is through the dynamic **import()** syntax

```
import { add } from './math';  
console.log(add(16, 26));
```



```
import("./math").then(math => {  
  console.log(math.add(16, 26));  
});
```


- The **React.lazy** function lets you render a dynamic import as a **regular component**

```
const OtherComponent = React.lazy(() =>
import(' ./OtherComponent '));

function MyComponent() {
  return (
    <div>
      <OtherComponent />
    </div>
  );
}
```

Suspense – Showing Indicators

- The **Suspense** component shows **fallback content** while we're waiting for another component to load

```
function MyComponent() {  
  return (  
    <div>  
      <Suspense fallback={<div>Loading...</div>}>  
        <OtherComponent />  
      </Suspense>  
    </div>  
  );  
}
```

Accepts any React element

- An example of how to setup route-based code splitting

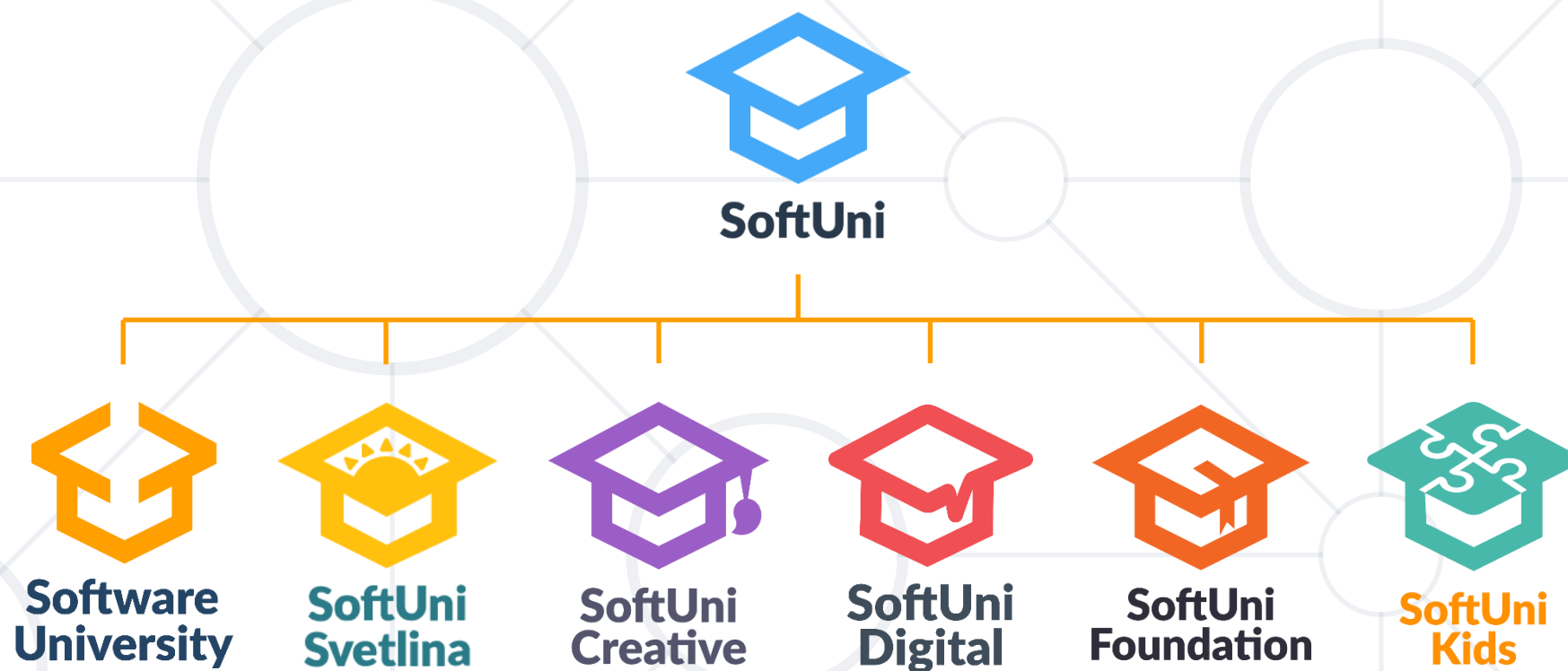
```
const Home = lazy(() => import('./routes/Home'));
const About = lazy(() => import('./routes/About'));

const App = () => (
  <Router>
    <Suspense fallback={<div>Loading...</div>}>
      <Routes>
        <Route path="/" component={<Home />} />
        <Route path="/about" component={<About />} />
      </Routes>
    </Suspense>
  </Router>
);
```

- Virtual Dom
 - The virtual DOM - **VDOM**
- Routing Overview
 - Internal handling of a route - **Client-side routing**
 - Single Page Applications - **Router**
- React Router
- React Lazy & Suspense



Questions?



SoftUni Diamond Partners



SCHWARZ





VIRTUAL RACING SCHOOL



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity

- Software University Forums

- forum.softuni.bg



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

