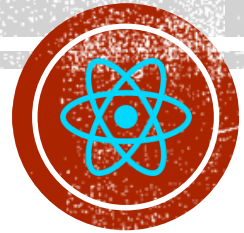
REACT ROUTER



Douglas Nassif Roma Junior

/douglasjunior

/in/douglasjunior

douglasjunior.me

nassifrroma@gmail.com

Slides: https://git.io/vbU3N

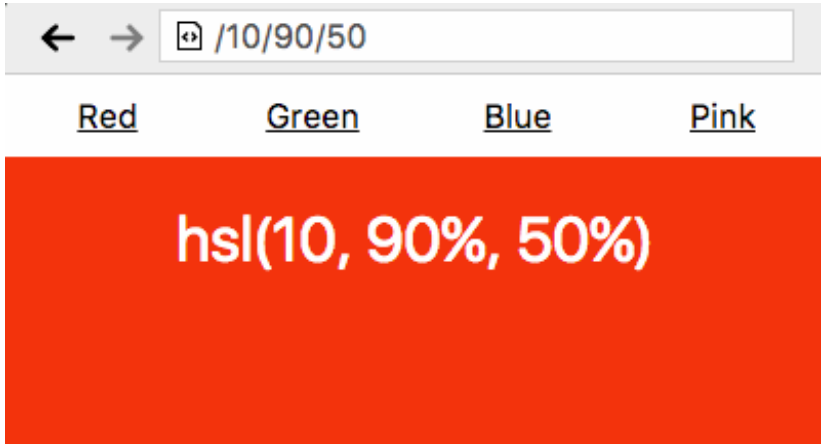
AGENDA

- Introdução ao React Router
- Parâmetros de URL
- Rotas privadas
- Links customizados
- Prevenindo transições
- Rotas desconhecidas (404)
- Rotas recursivas
- Transições animadas
- Referências

INTRODUÇÃO AO REACT ROUTER

- React Router é uma coleção de componentes de navegação que compõem declarativamente com sua aplicação.

- Se você quer ter URLs navegáveis para seu aplicativo Web ou uma maneira componentizada para navegar no React Native, o React Router funciona onde quer que o React JS esteja renderizando.



https://reacttraining.com/react-router/

INTRODUÇÃO AO REACT ROUTER

- Para instalar o React Router para Web, basta executar:

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

- Uso básico:

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


import Home from './pages/Home';
import Tasks from './pages/Tasks';
import About from './pages/About';
```

```
<Router>
  <ul>
    <li><Link to="/">Home</Link></li>
    <li><Link to="/tasks">Tarefas</Link></li>
    <li><Link to="/about">Sobre</Link></li>
  </ul>

  <Route exact path="/" component={Home} />
  <Route path="/tasks" component={Tasks} />
  <Route path="/about" component={About} />
</Router>
```

PARÂMETROS DE URL

- React Router permite que parâmetros sejam passados entre as rotas, usando parâmetros de URL.
- Para usar parâmetros de URL, basta usar o caractere “:” seguido do nome do parâmetro.

```
<Router>
  <div>
    <h2>Marvel</h2>
    <ul>
      <li><Link to="/ironman">Iron Man</Link></li>
      <li><Link to="/strange">Strange</Link></li>
      <li><Link to="/captain">Captain</Link></li>
      <li><Link to="/spide-man">Spider Man</Link></li>
    </ul>

    <Route path="/:heroId" component={Hero} />
  </div>
</Router>
```

```
const Hero = ({ match }) => (
  <div>
    <h3>Hero: {match.params.heroId}</h3>
  </div>
)
```

ROTAS PRIVADAS

- Usando o componente `Redirect` é possível redirecionar o usuário para uma rota determinada.
- Esta abordagem é útil, por exemplo, quando alguns dos componentes exigem autenticação do usuário.

```
const PrivateRoute = ({ component: Component, ...others }) => {
  return (
    <Route {...others} render={props => (
      fakeAuth.isAuthenticated
      ? <Component {...props} />
      : <Redirect to={{
          pathname: '/login',
          state: { from: props.location }
        }} />
    )} />
  )
}
```

```
<PrivateRoute path="/protected"
      component={Tasks} />
```

LINKS CUSTOMIZADOS

- Com a propriedade `children` do componente `Route` é possível renderizar um conteúdo que sofra alteração de acordo com a URL da página.
- Um exemplo desta aplicação é a criação de links customizados:

```
const CustomLink = ({ label, to, exact }) => (
  <Route path={to} exact={exact} children={({ match }) => (
    <div className={match ? 'active' : ''}>
      {match ? '> ' : ''}<Link to={to}>{label}</Link>
    </div>
  )} />
)
```

```
<CustomLink exact={true} to="/" label="Home" />
<CustomLink to="/tasks" label="Tarefas" />
```

PREVENINDO TRANSIÇÕES

- Em alguns casos, como preenchimento de formulários, pode ser interessante prevenir que o usuário navegue para outra rota da aplicação, sem antes salvar o trabalha em progresso.
- Exemplo de uso do componente `Prompt`:

```
<Prompt
    when={isBlocking}
    message={location => (
      `Tem certeza que deseja nevagar para ${location.pathname}?`
    )}
/>
```

ROTAS DESCONHECIDAS (404)

- Em uma aplicação Web pode ser interessante exibir uma mensagem amigável para um usuário. Com React Router, basta utilizar o componente `Switch`.
- Com o `Switch`, o React Router vai renderizar a primeira rota que combinar com a URL.

```
<Switch>
  <Route path="/" exact component={Home}/>
  <Redirect from="/old-match" to="/will-match"/>
  <Route path="/will-match" component={WillMatch}/>
  <Route component={NoMatch}/> {/* página 404 */}
</Switch>
```

ROTAS RECURSIVAS

- React Router também permite a renderização recursiva de componentes. Basta concatenar a URL atual com o Link das rotas que devem ficar aninhadas.

```
const Person = ({ match }) => {
  const person = find(match.params.id);
  return (
    <div>
      <h3>Amigos do {person.name}</h3>
      <ul>
        {person.friends.map(id => (
          <li key={id}>
            <Link to={`${match.url}/${id}`}>
              {find(id).name}
            </Link>
          </li>
        ))}
      </ul>
      <Route path={`${match.url}/:id`} component={Person} />
    </div>
  )
}
```

TRANSIÇÕES ANIMADAS

- Usando um módulo adicional ao React, é possível criar animações de transição com o React Router. Basicamente, este módulo adicionado irá adicionar/remover classes CSS em seus componentes, criando os efeitos desejados.

```
$ npm install --save react-transition-group
```

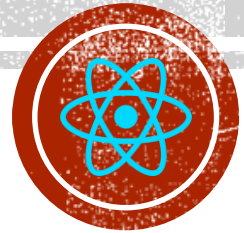
```
import { TransitionGroup, CSSTransition } from 'react-transition-group'
import './AnimationExample.css';

<TransitionGroup>
  <CSSTransition
   key={location.key}
   classNames="fade"
   timeout={{ exit: 300, enter: 300}}
  >
    <Route
     location={location}
     key={location.key}
     path="/sua/url/aqui"
     component={Comp}
    />
  </CSSTransition>
</TransitionGroup>
```

```
.fade-enter {
  opacity: 0;
  z-index: 1;
}

.fade-enter.fade-enter-active {
  opacity: 1;
  transition: opacity 300ms ease-in;
}

.fade-exit {
  opacity: 1;
}

.fade-exit.fade-exit-active {
  opacity: 0;
  transition: opacity 300ms ease-in;
}
```

REFERÊNCIAS

- React Router - https://reacttraining.com/react-router/
- React Router DOM - https://reacttraining.com/react-router/web/guides/philosophy
- React Transition Group - https://github.com/reactjs/react-transition-group

DÚVIDAS?



Douglas Nassif Roma Junior

/douglasjunior

/in/douglasjunior

douglasjunior.me

nassifrroma@gmail.com

Slides: https://git.io/vbU3N