Listas e Composição

React trabalha muito bem com os mesmos métodos de *Arrays* que vimos nos primeiros módulos de Javascript, e melhor que isso, por ser declarativo não precisamos nos preocupar se a lista inteira vai recarregar ou apenas um item, basta escrever o código que a biblioteca do React decidirá como e o que irá mudar.

```
render() {
  return (
    <ul>
      {list.map(item => (
        <li>{item.content}</li>
      ))}
    </ul>
  )
}
```

E não somente o `map` que facilita o uso e exibição de elementos com base em um *array*, `filter`, `reduce` e `sort` funcionam da mesma forma para que possamos renderizar elementos JSX em React utilizando métodos de *Arrays*.

Elementos e suas *keys*

Mas não é só desses métodos que vivemos, em React temos uma *Prop* especial, a `key`. Ela serve como uma indicação para que o React possa saber qual componente se refere a qual valor do array.

```
render() {
  return (
    <ul>
      {list.map((item, index) => (
        <li key={index}>{item.content}</li>
      ))}
    </ul>
  )
}
```

Dessa forma as atualizações de componentes são mais performáticas pois nada é alterado sem a necessidade.

MAS ATENÇÃO! Utilizar `index` como *key* de seus elementos é perigoso, tente encontrar valores **únicos** para não causar problemas com as iterações da sua

aplicação, IDs são bons valores para se usar nas *keys*.

Compondo componentes em Arrays

E depois disso tudo voltamos a compor componentes, mas aqui não tem grandes diferenças do que fizemos até agora.

```
render() {
  return (
    <ul>
      {list.map((item, index) => (
        <Item key={index} content={item.content} />
      ))}
    </ul>
  )
}
```

E o detalhe importante aqui é que a *key* deve ficar na **chamada do componente** que esta sendo iterado e não no elemento JSX desse componente. Segue um exemplo do que **não funciona**:

```
function Item(props) {
  return <li key={props.content}>{props.content}</li>
}
```

E indo alem, podemos criar componentes de classe que podem ter estados e enquanto esses estados mudam nossos componentes possuem a funcionalidade de renderizar apenas o elemento que teve essa alteração.

Possíveis Problemas

Vimos em aula como React pode ser travesso, principalmente se não utilizarmos as *keys* de forma correta. A documentação do React mostra mais sobre o assunto abordado na aula e também os links para os exemplos utilizados em aula.

https://pt-br.reactjs.org/docs/reconciliation.html