Estilização e listas

FUTURE4

Sumário

O que vamos ver hoje?

- Estilizando com *styled-components*
- Renderizando listas dinamicamente no React
- Listas no estado

styled-components

FUTURE4

Motivação

- Já temos HTML e JS juntos, só o CSS permanece separado
- Trabalhoso manter nomes de classes e ids
- É chato (e às vezes impossível) garantir que nomes de classes e ids não sejam conflitantes entre componentes

Solução

- CSS de cada componente **totalmente restrito** àquele componente

- Temos algumas bibliotecas que fazem isso por nós

- Escolhemos o *styled-components*, por ser uma das soluções mais aceitas e difundidas no mercado

Instalando

```
$ npm install styled-components
```

Lembrar de instalar toda vez que começar um projeto novo

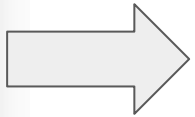
FUTURE4

Como usar

```
import styled from 'styled-components'

const RedTitle = styled.h1`
  color: red;
`

function App() {
  return (
    <div>
      <RedTitle>Meu Titulo</RedTitle>
    </div>
  )
}
```



Meu Titulo

FUTURE4

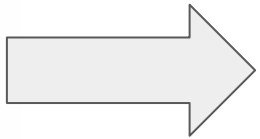
Como usar

```
import styled from 'styled-components'

const RedTitle = styled.h1`
  color: red;
`

const TitleContainer = styled.div`
  display: flex;
  align-items: center;
  height: 60px;
  background-color: blue;
  padding-left: 10px;
`

function App() {
  return (
    <TitleContainer>
      <RedTitle>Meu Titulo</RedTitle>
    </TitleContainer>
  )
}
```



FUTURE4

Adaptando o estilo via props

- Podemos querer controlar partes do nosso estilo com código JavaScript

- Podemos passar props para um styled-component, que podem ser utilizadas dentro do código CSS

Adaptando o estilo via props

```
import styled from 'styled-components'

const RedTitle = styled.h1`
  color: red;
`

const TitleContainer = styled.div`
  display: flex;
  align-items: center;
  height: ${props => props.size};
  background-color: ${props => props.color};
  padding-left: 10px;
`

function App() {
  const containerColor = 'blue'
  return (
    <TitleContainer color={containerColor} size={'60px'}>
      <RedTitle>Meu Titulo</RedTitle>
    </TitleContainer>
  )
}
```

FUTURE4

Adaptando o estilo via props

```
import styled from 'styled-components'

const RedTitle = styled.h1`
  color: red;
`

const TitleContainer = styled.div`
  display: flex;
  align-items: center;
  height: ${props => {
    if(props.size === 'big') {
      return '120px'
    } else if (props.size === 'small') {
      return '60px'
    }
  }};
  background-color: ${props => props.color};
  padding-left: 10px;
`

function App() {
  const containerColor = 'blue'
  return (
    <TitleContainer color={containerColor} size={'big'}>
      <RedTitle>Meu Titulo</RedTitle>
    </TitleContainer>
  )
}
```

FUTURE4

Pausa :)

Coding Together

FUTURE4

Adaptando nosso currículo para usar styled-components

[Clique aqui para abrir o código base]

FUTURE4

Listas

FUTURE4

Motivação

- Como já vimos, em React normalmente vamos ter nossos dados em objetos JavaScript, que servirão como entrada dos nossos componentes

- Frequentemente modelamos nossos dados como listas (arrays)

- Precisamos de uma forma simples de mostrar esses componentes na tela

Renderizando uma lista

- Se colocarmos um array de componentes no JSX, o React entende que deve renderizar cada componente

```
const listaDeComponentes = [
  <li>Item 1</li>,
  <li>Item 2</li>,
  <li>Item 3</li>
]

function App() {
  return (
    <div>
      <ul>{listaDeComponentes}</ul>
    </div>
  );
}
```

- Item 1
- Item 2
- Item 3

FUTURE4

Renderizando um array de dados

- Precisamos transformar um array de dados em um array de componentes

- Relembrando: podemos usar a função de array **map** quando queremos transformar um array em outro, mapeando cada posição

Renderizando um array de dados

```
const listaDeDados = [
  'Item 1',
  'Item 2',
  'Item 3'
]

const listaDeComponentes = listaDeDados.map((dado) => {
  return <li>{dado}</li>
})

function App() {
  return (
    <div>
      <ul>{listaDeComponentes}</ul>
    </div>
  );
}
```

FUTURE4

Keys

- Cada componente deve ter um identificador único
- Serve para o React otimizar o processamento, sabendo quais elementos mudaram
- Deve ser passada como a prop key do componente da lista

Keys

```
const listaDeDados = [
  'Item 1',
  'Item 2',
  'Item 3'
]

const listaDeComponentes = listaDeDados.map((dado) => {
  return <li key={dado}>{dado}</li>
})

function App() {
  return (
    <div>
      <ul>{listaDeComponentes}</ul>
    </div>
  );
}
```

FUTURE4

Exemplo

```
const videos = [{
  imagem: 'https://picsum.photos/200/100?a=1',
  titulo: 'Titulo do vídeo 1'
}, {
  imagem: 'https://picsum.photos/200/100?a=2',
  titulo: 'Titulo do vídeo 2'
}, {
  imagem: 'https://picsum.photos/200/100?a=3',
  titulo: 'Titulo do vídeo 3'
}, {
  imagem: 'https://picsum.photos/200/100?a=4',
  titulo: 'Titulo do vídeo 4'
}, {
  imagem: 'https://picsum.photos/200/100?a=5',
  titulo: 'Titulo do vídeo 5'
}, {
  imagem: 'https://picsum.photos/200/100?a=6',
  titulo: 'Titulo do vídeo 6'
}, {
  imagem: 'https://picsum.photos/200/100?a=7',
  titulo: 'Titulo do vídeo 7'
}, {
  imagem: 'https://picsum.photos/200/100?a=8',
  titulo: 'Titulo do vídeo 8'
}]
```

```
const listaDeVideos = videos.map(video => {
  return (
  <VideoCard
    key={video.titulo}
    imagem={video.imagem}
    titulo={video.titulo}
  />)
})

function App() {
  return (
    <div className="App">
      <Header />
      <main>
        {listaDeVideos}
      </main>
      <Footer />
    </div>
  );
}
```

Exemplo

```
const videos = [{
  imagem: 'https://picsum.photos/200/100?a=1',
  titulo: 'Titulo do vídeo 1'
}, {
  imagem: 'https://picsum.photos/200/100?a=2',
  titulo: 'Titulo do vídeo 2'
}, {
  imagem: 'https://picsum.photos/200/100?a=3',
  titulo: 'Titulo do vídeo 3'
}, {
  imagem: 'https://picsum.photos/200/100?a=4',
  titulo: 'Titulo do vídeo 4'
}, {
  imagem: 'https://picsum.photos/200/100?a=5',
  titulo: 'Titulo do vídeo 5'
}, {
  imagem: 'https://picsum.photos/200/100?a=6',
  titulo: 'Titulo do vídeo 6'
}, {
  imagem: 'https://picsum.photos/200/100?a=7',
  titulo: 'Titulo do vídeo 7'
}, {
  imagem: 'https://picsum.photos/200/100?a=8',
  titulo: 'Titulo do vídeo 8'
}]
```

```
function App() {
  return (
    <div className="App">
      <Header />
      <main>
        {videos.map(video => {
          return (
          <VideoCard
            key={video.titulo}
            imagem={video.imagem}
            titulo={video.titulo}
          />)
        })}
      </main>
      <Footer />
    </div>
  );
}
```

FUTURE4

Pausa :)

Listas no estado

- Comumente, nossas listas serão dinâmicas, ou seja, poderão receber novos itens, ou ter itens apagados

- Nesse caso, é interessante armazenar a lista no estado, pois queremos que nossa UI reflita o conteúdo da lista

- Precisamos tomar alguns cuidados

Listas no estado

- Lembrando:
  - Arrays são referências, e quando adicionamos um item a eles, a referência continua a mesma
  - O React só atualiza a UI quando detecta uma mudança no estado ou nas props

- Portanto, não podemos simplesmente adicionar ou remover um item de um array

- É preciso criar um **novo** array

Adicionando um elemento - push

```
class App extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      videos: videos
    }
  }

  addVideo = (newVideo) => {
    const videosCopy = [...this.state.videos]
    videosCopy.push(newVideo)
    this.setState({videos: videosCopy})
  }

  render() {...}
}
```

FUTURE4

Adicionando um elemento - spread

```
class App extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      videos: videos
    }
  }

  addVideo = (newVideo) => {
    const videosCopy = [...this.state.videos, newVideo]
    this.setState({videos: videosCopy})
  }

  render() {...}
}
```

FUTURE4

Removendo um elemento - splice

```
class App extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      videos: videos
    }
  }

  removeVideo = (index) => {
    const videosCopy = [...this.state.videos]
    videosCopy.splice(index, 1)
    this.setState({videos: videosCopy})
  }

  render() {...}
}
```

FUTURE4

Removendo um elemento - filter

```
class App extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      videos: videos
    }
  }

  removeVideo = (indexToRemove) => {
    const videosCopy = this.state.videos.filter((video, index) => {
      return index !== indexToRemove
    })
    this.setState({videos: videosCopy})
  }

  render() {...}
}
```

FUTURE4

Pausa :)

Coding Together

FUTURE4

Criando forms com lista de itens com seletor de cor

Criem um novo projeto no stackblitz: https://stackblitz.com/

Dúvidas?

FUTURE4

Obrigado!

FUTURE4