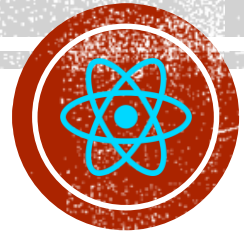
INTODUÇÃO AO REACTJS



Douglas Nassif Roma Junior

/douglasjunior

/in/douglasjunior

douglasjunior.me

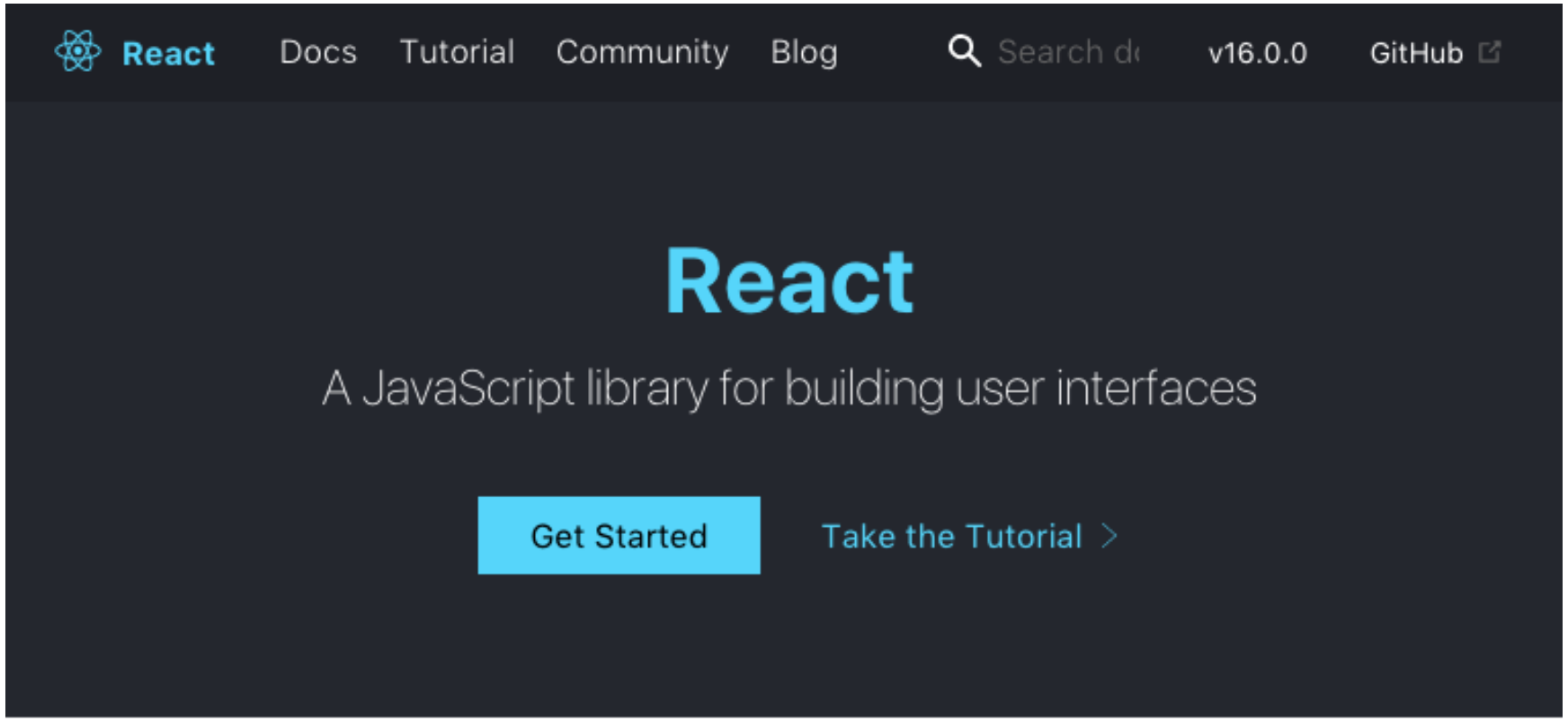
nassifrroma@gmail.com

Slides: https://git.io/vbU3N

AGENDA

- Introdução ao ReactJS
- Componentes
  - Componente funcional
  - Componente de classe
- ECMAScript 2015 e JSX
  - Create-React-App
- Estado e Clico de Vida
- Coleções de Componentes
- Referências

INTRODUÇÃO AO REACTJS



Declarative

Component-Based

Learn Once,
Write Anywhere

http://reactjs.org

INTRODUÇÃO AO REACTJS

- **Declarativo**

- React facilita a criação de UIs interativas. Crie *views* simples para cada estado em seu aplicativo e o React irá atualizar e renderizar eficientemente apenas os componentes certos quando seus dados forem alterados.

- Views declarativas tornam seu código mais previsível e mais fácil de depurar.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}
```

INTRODUÇÃO AO REACTJS

- **Baseado em Componentes**

- Crie componentes encapsulados que gerenciem seu próprio estado, e então, use-os para compor UIs complexas.

- Uma vez que a lógica dos componentes está escrita em JavaScript, você pode facilmente passar dados através do seu aplicativo e manter o estado fora do DOM.

```
class Timer extends React.Component {

  tick() {
    this.setState((prevState) => ({
      seconds: prevState.seconds + 1
    }));
  }

  componentDidMount() {
    setInterval(() => this.tick(), 1000);
  }

  render() {
    return (
      <div>
        Seconds: {this.state.seconds}
      </div>
    );
  }
}
```

INTRODUÇÃO AO REACTJS

- **Aprenda uma vez, use em qualquer lugar**

- Não exige que você altere o conjunto de tecnologias utilizados em sua stack, evitando reescrever o código de sua aplicação.

- React também pode renderizar no servidor usando **Node** ou alimentar aplicativos móveis usando o **React Native**.

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';

class WhyReactNativeIsSoGreat extends Component {
  render() {
    return (
      <View>
        <Text>
          Se você gosta do React na web, você
          vai gostar do React Native.
        </Text>
        <Text>
          Você apenas usa componentes nativos
          como 'View' e 'Text', em vez de um
          componentes web como 'div' e 'span'.
        </Text>
      </View>
    );
  }
}
```

INTRODUÇÃO AO REACTJS

- Para utilizar o ReactJS na web, você precisa apenas importar as bibliotecas React e React-DOM.

```
<script crossorigin src="https://unpkg.com/react@16.2.0/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16.2.0/umd/react-dom.development.js"></script>
```

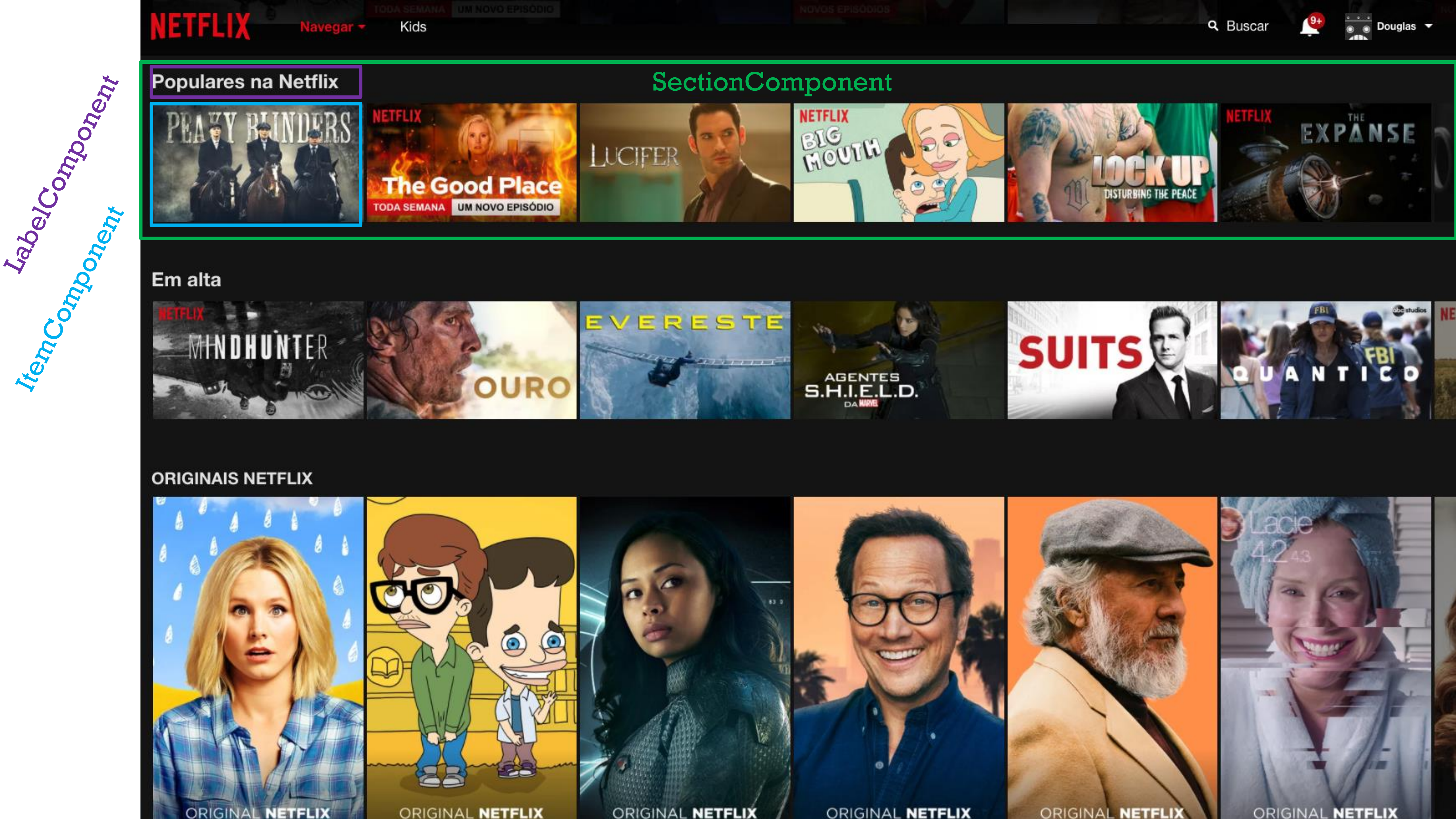
- E então você já pode renderizar seu primeiro componente.

```
<div id="root"></div>

<script>
  const MyDiv = React.createElement('div', null, 'Olá React JS');

  ReactDOM.render(MyDiv, document.getElementById('root'));
</script>
```

COMPONENTES

- Os componentes permitem que você divida a UI em partes independentes, reutilizáveis e pense em cada uma isoladamente.

- Conceitualmente, os componentes são como funções JavaScript. Eles aceitam entradas arbitrárias (chamados de "`props`") e retornam elementos descrevendo o que deve aparecer na tela.



NETFLIX
Navegar
Kids
Buscar
Douglas
LabelComponent
ItemComponent
Populares na Netflix
SectionComponent
PEAKY BLINDERS
NETFLIX
The Good Place
TODA SEMANA
UM NOVO EPISÓDIO
LUCIFER
NETFLIX
BIG MOUTH
LOCK UP
DISTURBING THE PEACE
NETFLIX
THE EXPANSE
Em alta
NETFLIX
MINDHUNTER
OURO
EVERESTE
AGENTES S.H.I.E.L.D. DA MARVEL
SUITS
FBI
QUANTICO
ORIGINAIS NETFLIX
ORIGINAL NETFLIX
ORIGINAL NETFLIX
ORIGINAL NETFLIX
ORIGINAL NETFLIX
ORIGINAL NETFLIX
ORIGINAL NETFLIX

COMPONENTES FUNCIONAIS

- A maneira mais simples de se criar um Componente em ReactJS, é declarando uma função JavaScript.

```
function WelcomeComponent(props) {
   return React.createElement('h1', null, 'Hello, ' + props.name);
}
```

- Esta função é um componente válido pois ela recebe um parâmetro único chamado "`props`" e retorna um elemento React.

- É chamado de componente "funcional" pois, literalmente, é uma função JavaScript.

COMPONENTES DE CLASSES

- Os componentes de classe também recebem valores através de "`props`" e podem renderizar um ou mais elementos React.

- Adicionalmente, os componentes de classe são capazes de gerenciar seu próprio estado.

- À partir da versão 16, o React moveu a função de criação de classes para uma biblioteca separa, então é preciso declarar:

```
<script crossorigin src="https://unpkg.com/create-react-class@15.6.2/create-react-class.js"></script>
```

COMPONENTES DE CLASSES

- Os componentes de classe podem ser declarados assim:

```
const CounterComponent = createReactClass({
  getInitialState: function () {
    return {
      count: 0,
    };
  },
  componentDidMount: function () {
    const self = this;
    setInterval(function () {
      self.setState({ count: self.state.count + 1 })
    }, 1000);
  },
  render: function () {
    return React.createElement('p', null, this.state.count);
  }
});
```

ECMASCRIPT 2015 E JSX

- Visando aproveitar todo o poder da versões mais recentes do JavaScript, o React dispõe de um plugin para o **Babel** que permite o uso do ECMAScript 2015 (ES6) e JSX.

```
function WelcomeComponent(props) {
  return React.createElement('h1', null, 'Hello, ' + props.name);
}

const WelcomeComponent = (props) => (
  <h1>Hello, {props.name}</h1>
)
```

- Para isso, é recomendado o uso de ferramentas como "`webpack`" ou "`Browserify`" para auxiliar na "transpilação" e empacotamento do código.

- Felizmente, existe uma ferramenta chamada "`create-react-app`" que faz todo o trabalho necessário com um único comando.

CREATE-REACT-APP

- Ferramenta de linha de comando para auxiliar na criação e configuração de um projeto "`webpack`" com ReactJS.

- Adicionalmente, `create-react-app` traz um conjunto de configurações e módulos auxiliares que facilitam o processo de desenvolvimento e implantação.

- Para instalar o `create-react-app` utilize:

```
$ npm install -g create-react-app
```

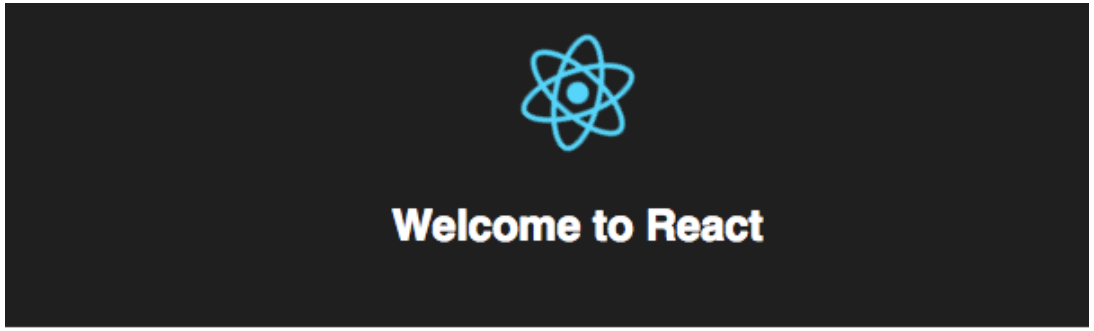
CREATE-REACT-APP

- **Para criar seu primeiro projeto, execute o comando:**

```
$ create-react-app meu-projeto-reactjs
```

- **Entre na pasta do projeto e execute e inicie o `webpack-dev-server` em modo de desenvolvimento:**

```
$ cd meu-projeto-reactjs
$ npm start
```



To get started, edit `src/App.js` and save to reload.

ESTADO E CICLO DE VIDA

- Estado é semelhante às propriedades, porém ele é privado e totalmente controlado pelo componente de classe.

- Para entender a diferença, vamos criar um componente Relógio que recebe o tempo via "`props`" e em seguida alterá-lo para controlar seu próprio estado.

https://reactjs.org/docs/state-and-lifecycle.html

ESTADO E CICLO DE VIDA

Recebendo data e hora via "`props`".

```
import React from 'react';
import ReactDOM from 'react-dom';

function Relogio(props) {
  return (
    <div>
      <h1>Olá React!</h1>
      <h2>Hora certa: {props.date.toLocaleTimeString()}.</h2>
    </div>
  );
}

function contar() {
  ReactDOM.render(<Relogio date={new Date()} />,
    document.getElementById('root'));
}

setInterval(contar, 1000);
```

ESTADO E CICLO DE VIDA

Controlando seu próprio estado.

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

class Relogio extends Component {

  state = {
    date: new Date()
  }

  componentDidMount() {
    setInterval(this.contar.bind(this), 1000);
  }

  contar() {
    this.setState({ date: new Date() });
  }

  // continua
```

```
  render() {
    const { date } = this.state;
    return (
      <div>
        <h1>Olá React!</h1>
        <h2>
          Hora certa:
          {date.toLocaleTimeString()}.
        </h2>
      </div>
    );
  }
}

ReactDOM.render(
  <Relogio />, document.getElementById('root')
);
```

ESTADO E CICLO DE VIDA

- Componentes de classe também possuem eventos que podem ser utilizados para detectar momentos do clico de vida dos componentes.

- **Montagem**: Eventos disparados quando o componente é criado e anexado ao DOM.
  - `constructor()`
  - `componentWillMount()`
  - `render()`
  - `componentDidMount()`

ESTADO E CICLO DE VIDA

- **Atualização**: Uma atualização pode ocorrer por uma mudança nas `props` ou no estado. Estes eventos são disparados quando o componente é re-renderizado.
  - `componentWillReceiveProps()`
  - `shouldComponentUpdate()`
  - `componentWillUpdate()`
  - `render()`
  - `componentDidUpdate()`

- **Desmontagem**: Evento chamado quando o componente será destruído.
  - `componentWillUnmount()`

COLEÇÕES DE COMPONENTES

- Com a popularização do ReactJS, é comum encontrar versões de bibliotecas, frameworks e conjuntos de componentes já prontos para o uso com React. Basta pesquisar no Google por: "React <palavra-chave>"

- Dois exemplos que devemos citar são: **React-Bootstrap** (Bootstrap 3) e **ReactStrap** (Bootstrap 4)

- Ambos trazem o framework CSS Bootstrap em forma de componentes React, prontos para uso.

COLEÇÕES DE COMPONENTES

- Para instalar o ReactStrap, basta executar:

```
$ npm install --save bootstrap@4.0.0-beta.3 reactstrap@next
```

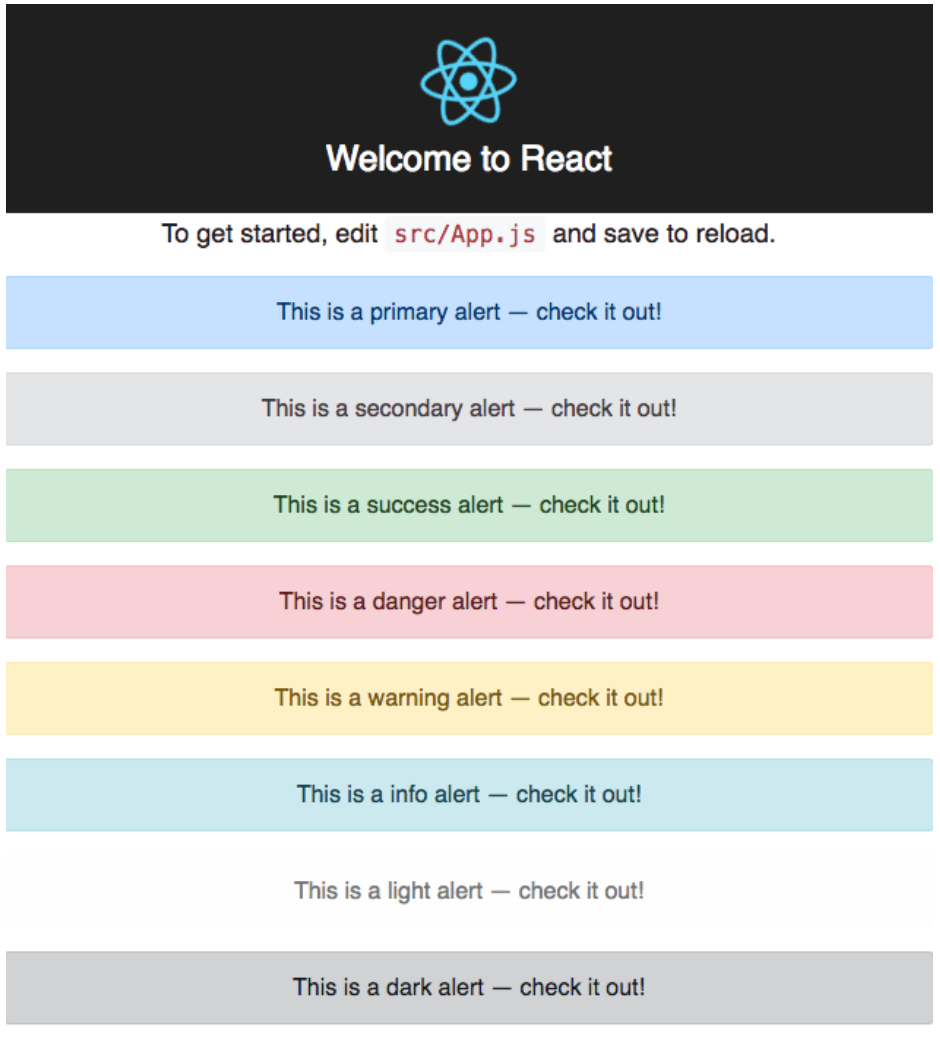
- E então, importar o Bootstrap 4 globalmente no `index.js`:

```
import 'bootstrap/dist/css/bootstrap.css';
```

https://reactstrap.github.io/

COLEÇÕES DE COMPONENTES

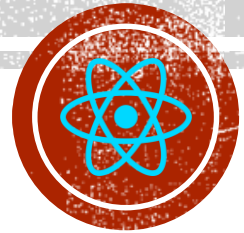
```
import { Alert } from 'reactstrap';
...
<div className="App">
  ...
  <Alert color="primary">
    This is a primary alert — check it out!
  </Alert>
  <Alert color="secondary">
    This is a secondary alert — check it out!
  </Alert>
  <Alert color="success">
    This is a success alert — check it out!
  </Alert>
  <Alert color="danger">
    This is a danger alert — check it out!
  </Alert>
  <Alert color="warning">
    This is a warning alert — check it out!
  </Alert>
  <Alert color="info">
    This is a info alert — check it out!
  </Alert>
  <Alert color="light">
    This is a light alert — check it out!
  </Alert>
  <Alert color="dark">
    This is a dark alert — check it out!
  </Alert>
</div>
```



REFERÊNCIAS

- React JS - http://reactjs.org
- JSX - https://reactjs.org/docs/introducing-jsx.html
- Documentação - https://reactjs.org/docs/installation.html
- Create-react-app - https://github.com/facebookincubator/create-react-app
- Podcast - https://hipsters.tech/react-o-framework-onipresente-hipsters-66/
- React-Bootstrap - https://react-bootstrap.github.io/
- ReactStrap - https://reactstrap.github.io/

DÚVIDAS?



Douglas Nassif Roma Junior

/douglasjunior

/in/douglasjunior

douglasjunior.me

nassifrroma@gmail.com

Slides: https://git.io/vbU3N