



Componentes Web

Prof. Victor Farias

v 3.0

Introdução



Problema

- Como reutilizar componentes?
 - Copia e cola códigos HTML, CSS e JS inteiros de terceiros
 - Linhas e linhas de código incompreensíveis sujando seu projeto

Modal

```
<!DOCTYPE html>
<html>
<head>
<style>
/* The Modal (background) */
.modal {
  display: none; /* Hidden by default */
  position: fixed; /* Stay in place */
  z-index: 1; /* Sit on top */
  left: 0;
  top: 0;
  width: 100%; /* Full width */
  height: 100%; /* Full height */
  overflow: auto; /* Enable scroll if needed */
  background-color: rgb(0,0,0); /* Fallback color */
  background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
  -webkit-animation-name: fadeIn; /* Fade in the background */
  -webkit-animation-duration: 0.4s;
  animation-name: fadeIn;
  animation-duration: 0.4s
}

/* Modal Content */
.modal-content {
  position: fixed;
  bottom: 0;
  background-color: #fefefe;
  width: 100%;
  -webkit-animation-name: slideIn;
  -webkit-animation-duration: 0.4s;
  animation-name: slideIn;
  animation-duration: 0.4s
}

/* The Close Button */
.close {
  color: white;
  float: right;
  font-size: 28px;
  font-weight: bold;
}
```

```
close:hover,
.close:focus {
  color: #000;
  text-decoration: none;
  cursor: pointer;
}

.modal-header {
  padding: 2px 16px;
  background-color: #5cb85c;
  color: white;
}

.modal-body {padding: 2px 16px;}

.modal-footer {
  padding: 2px 16px;
  background-color: #5cb85c;
  color: white;
}

/* Add Animation */
@-webkit-keyframes slideIn {
  from {bottom: -300px; opacity: 0}
  to {bottom: 0; opacity: 1}
}

@keyframes slideIn {
  from {bottom: -300px; opacity: 0}
  to {bottom: 0; opacity: 1}
}

@-webkit-keyframes fadeIn {
  from {opacity: 0}
  to {opacity: 1}
}

@keyframes fadeIn {
  from {opacity: 0}
  to {opacity: 1}
}

</style>
</head>
<body>
```

<h2>Bottom Modal</h2>

```
<!-- Trigger/Open The Modal -->
<button id="myBtn">Open Modal</button>
```

```
<!-- The Modal -->
<div id="myModal" class="modal">
```

```
  <!-- Modal content -->
  <div class="modal-content">
    <div class="modal-header">
      <span class="close">&times;</span>
      <h2>Modal Header</h2>
    </div>
    <div class="modal-body">
      <p>Some text in the Modal Body</p>
      <p>Some other text...</p>
    </div>
    <div class="modal-footer">
      <h3>Modal Footer</h3>
    </div>
  </div>
```

</div>

```
<script>
// Get the modal
var modal = document.getElementById('myModal');
```

```
// Get the button that opens the modal
var btn = document.getElementById("myBtn");
```

```
// Get the <span> element that closes the modal
var span = document.getElementsByClassName("close")[0];
```

```
// When the user clicks the button, open the modal
btn.onclick = function() {
  modal.style.display = "block";
}
```

```
// When the user clicks on <span> (x), close the modal
span.onclick = function() {
  modal.style.display = "none";
}
```

```
// When the user clicks anywhere outside of the modal, close it
window.onclick = function(event) {
  if (event.target === modal) {
    modal.style.display = "none";
  }
}
</script>
```

</body>

</html>

Problema

- Como reutilizar componentes?
 - Copia e cola códigos HTML, CSS e JS inteiros de terceiros
 - Linhas e linhas de código incompreensíveis sujando seu projeto
 - Usar framework de componente - Bootstrap, Materialize, Foundation ...
 - Modo próprio de importar componentes
 - Importação de componentes muda de um para outro

Modal

Bootstrap

```
<div class="modal fade">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Modal title</h5>
        <button type="button" class="close"
          data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <p>Modal body text goes here.</p>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-primary">
          Save changes</button>
        <button type="button" class="btn btn-secondary"
          data-dismiss="modal">Close</button>
      </div>
    </div>
  </div>
</div>
```

Foundation

```
<a href="#" data-reveal-id="myModal">Click Me
For A Modal</a>
```

```
<div id="myModal" class="reveal-modal"
data-reveal aria-labelledby="modalTitle"
aria-hidden="true" role="dialog">
  <h2 id="modalTitle">Awesome. I have it.</h2>
  <p class="lead">Your couch. It is mine.</p>
  <p>I'm a cool paragraph that lives inside of
an even cooler modal. Wins!</p>
  <a class="close-reveal-modal"
aria-label="Close">&#215;</a>
</div>
```

Solução?

Componentes
Web!





Componentes WEB

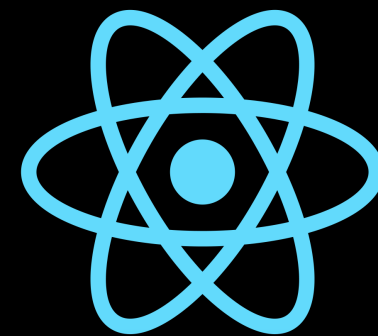
Componentes

```
<my-modal>  
  <h2>Título da modal</h2>  
  <p>Texto do corpo da modal</p>  
</my-modal>
```

- Reutilizável!
- Limpo!
- Semântico!

Como?

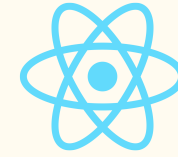
- Web Components 
 - Componentes nativos do *browser*
- Angular  React  Vue.js 
 - Frameworks que implementam componentes web segundo sua própria filosofia
 - Além disso, oferecem uma solução completa para aplicações web



React



React



- Framework para desenvolvimento de interfaces Web
- Mantida pelo Facebook
- Multiplataforma
 - Web
 - Móvel - React Native
- Linguagem JavaScript NextGen
- **JSX**
- Dispõe de ferramentas em linha de comando
- Suporta testes

Pré-requisito

1. Ter o node instalado

<https://nodejs.org/en/>

Novo projeto (npm 5.1 ou inferior)

1. Instalar ferramentas Create React App (p/ npm 5.1 ou menor)

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

2. Criar novo projeto

```
create-react-app my-app
```

3. Entrar na pasta

```
cd my-app
```

4. Rodar servidor

```
npm start
```

Novo projeto (npm 5.2 ou superior)

1. Criar novo projeto

```
npx create-react-app my-app
```

2. Entrar na pasta

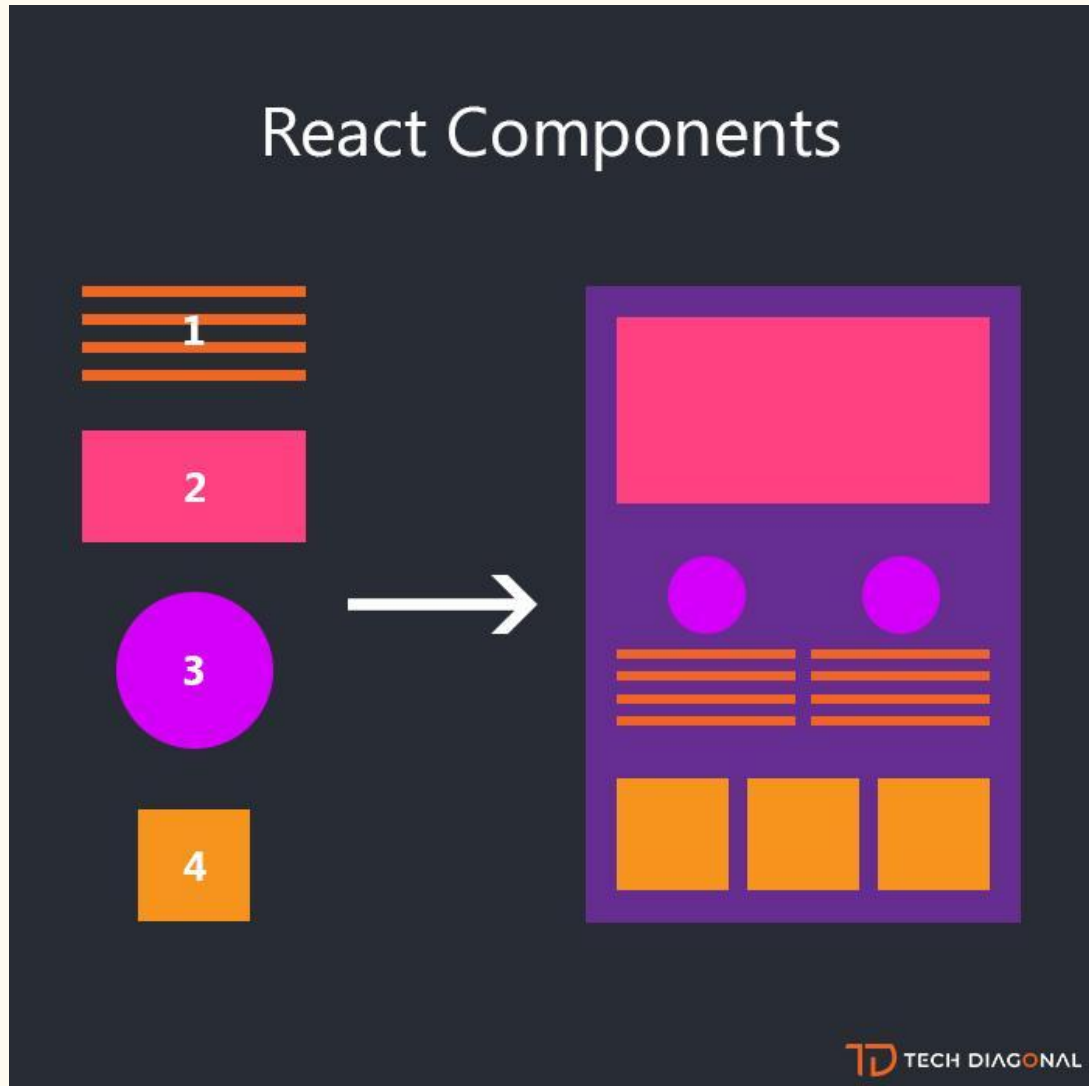
```
cd my-app
```

3. Rodar servidor

```
npm start
```

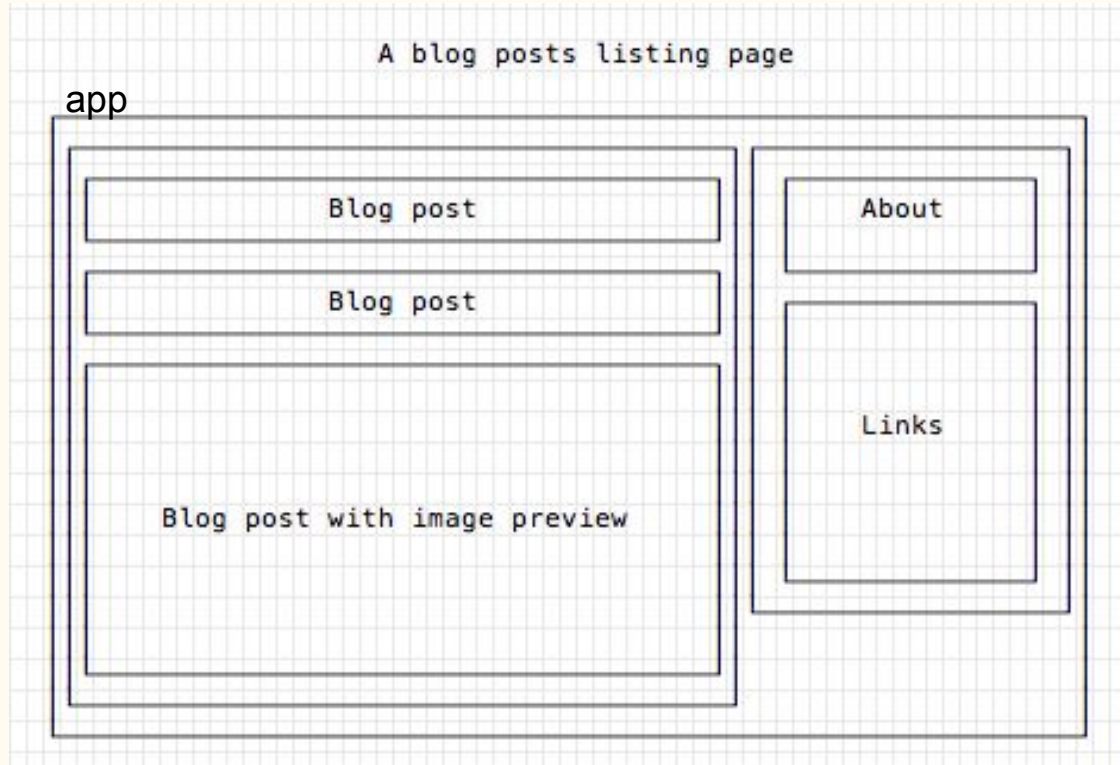
Componentes React

Componentes



- Cada componente produz um pedaço de código HTML
- A página inteira é construída pela composição dos componentes

Componentes



<https://flaviocopes.com/react-components/>

- Componentes podem ser reaproveitados em diversos lugares/páginas
- Componente raiz padrão: **App**
- Dois modos de fazer componentes
 - Componentes baseado em **função**
 - Componentes baseado em classe

Funcao.js

Componentes produzem código HTML

```
export function ComponenteFuncao(){  
  return <div> Componente funcao </div>;  
}
```

Retorna código HTML
Obs: Só pode ter um elemento pai!

Como usar esse componentes?

Componentes se comportam como novas tags!

app.js

```
import ComponenteFuncao from './Funcao'
...
function App() {
  return (
    <div className="App">
      <ComponenteFuncao></ ComponenteFuncao >
    </div>
  );
}
```

Estrutura do projeto

Estrutura do projeto

- src
 - components
 - commom
 - Navegador
 - Navegador.js
 - Navegador.css
 - pages
 - PaginaPrincipal
 - PaginalPrincipal.js
 - PaginaPrincipal.css

JSX



JSX

- Facilita a construção de componentes HTML usando JS
- Permite a escrita de código HTML diretamente no JS
- Trata código HTML como objeto
 - Pode-se manipular como qualquer objeto JS
 - Inserir em lista
 - Receber como parâmetro de função
 - Serve como valor de retorno de função
- Além disso, se pode colocar código JS dentro do código HTML
- **Código JS dentro HTML vai dentro {}**

JSX

```
export function Teste(){  
  return <div> Aqui vai uma soma: {1+2} </div>;  
}
```

JSX

```
export function Teste(){  
  let name = "João";  
  return <div> Meu nome é {name} </div>;  
}
```

JSX

```
export function Teste(){  
  let link = (<a href="http://globo.com">link para globo</a>);  
  return <div> Segura um link: {link} </div>;  
}
```

Props



Props

- Como enviar dados de entrada para compor um componente?
 - Passar o nome de um produto
 - Passar a mensagem de um post
- Podemos usar o **Props**!
- Props são passados pelo componente pai como atributo na tag HTML do componente filho
- Props são recebidos pelo componente filho via
 - parâmetro do construtor (componente classe)
 - parâmetro da função (componente função)

Componente pai

...

```
<Teste nome="teste 1"></Teste>
```

...

Componente Filho (ComponenteClasse.js)

```
export function Teste(props){  
    return <div> Sou um componente {props.nome}</div>;  
}
```

Componente Filho (ComponenteClasse.js)

```
export function Teste({nome}){  
    return <div> Sou um componente {nome}</div>;  
}
```


Lista de elementos



Lista de elementos

```
export function Conteudo(){  
  let disciplinas = [  
    {  
      nome: "LMS",  
      codigo: "QXD253",  
    },  
    {  
      nome: "PIW",  
      codigo: "QXD5435",  
    },  
    {  
      nome: "SOC",  
      codigo: "QXD2323",  
    }  
  ];...
```

Lista de elementos (continuação)

```
...
  let lis = disciplinas.map((disciplina) => (<li>{disciplina.nome} - {disciplina.codigo}</li>))
  return(
    <div>
      <ul>
        {lis}
      </ul>
    </div>
  )
}
```

Estilo



Teste.css

```
.fundo-azul{
```

```
    background-color: blue;
```

```
}
```

Teste.js

```
import './Teste.css';
```

```
export function Teste(){  
    return <div className="fundo-azul">Teste</div>  
}
```

Classes Dinâmicas

ClassesDinamicas.css

```
.azul{  
    background-color: blue;  
}
```

```
.botao{  
    width: 200px;  
    height: 200px;  
}
```


ClassesDinamicas.js

```
import './ClassesDinamicas.css';

export function ClassesDinamicas() {

    let classes_list = []
    if(props.azul == true){
        classes_list.push('azul');
    }
    classes_list.push('botao')

    let classes = classes_list.join(' ')
    ... (cont)
```

ClassesDinamicas.js (cont.)

...

```
    return (  
      <button className={classes}>  
        Me clique  
      </button> );  
  }
```

Biblioteca classnames

npm install classnames

Biblioteca classnames

```
classNames('foo', 'bar'); // => 'foo bar'
```

```
classNames('foo', { bar: true }); // => 'foo bar'
```

```
classNames({ 'foo-bar': true }); // => 'foo-bar'
```

```
classNames({ 'foo-bar': false }); // => ''
```

```
classNames({ foo: true }, { bar: true }); // => 'foo bar'
```

```
classNames({ foo: true, bar: true }); // => 'foo bar'
```

Eventos



Eventos

- Como capturar eventos?
 - Click
 - MouseDown
 - DoubleClick
- Pode-se escutar eventos usando atributos HTML do componente
- Para executar uma ação, passamos uma função de *callback* que executa a ação desejada

Click - Componente pai - MyButton.js

```
function ActionLink() {  
  function handleClick(e) {  
    e.preventDefault();  
    console.log('O link foi clicado.');  }  
  
  return (  
    <a href="#" onClick={handleClick}>  
      Clique Aqui  
    </a>  
  );  
}
```

Eventos de Mouse

- `onClick`
- `onDoubleClick`
- `onMouseDown`
- `onMouseEnter`
- `onMouseLeave`
- `onMouseMove`
- `onMouseOut`
- `onMouseOver`
- `onMouseUp`

Perguntas?

Prof. Victor Farias