React: renderização condicional e de listas

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- Renderização condicional
- Renderização de listas
- Deploy

RENDERIZAÇÃO CONDICIONAL

Renderização condicional

Como vimos desde as primeiras aulas de JS uma
função só pode ter **um retorno**.
Isso não é diferente em componentes React,
já que são funções que retornam elementos JSX.

No entanto, há uma forma de **controlar** quando e quais
elementos renderizar: **renderização condicional**.

Assim como em outras funções JS, podemos usar técnicas de condições, como
blocos if, operadores ternários ou switch cases.

Renderização condicional

Por exemplo:

- se uma condição for **verdadeira** retornar determinado objeto
  - caso seja **falsa**, o JS atinge o retorno padrão da função

```
export default function App({ isLoading }) {

  if(isLoading) {
    return <p>Carregando...</p>
  }

  return (
    <>
      <h1>Aula 04</h1>
      <p>A aplicação carregou!</p>
    </>
  );
}
```

Renderização condicional

Também podemos usar renderização condicional **dentro do JSX**.
Basta abrir uma **interpolação de expressão JS**, determinando em um ponto do componente se um elemento deve ou não ser exibido.

```jsx
export default function Doces({ listaDeDoces = [] }) {
  return (
    <>
      <h2>Mamãe comprou doces?</h2>
      {listaDeDoces.length >= 1 ? (
        <p>Sim, ainda temos {listaDeDoces.length} doces em casa!</p>
      ) : (
        <>
          <p>Acabaram os doces!</p>
          <p>Urgente: pedir mais pra mamãe!</p>
        </>
      )}
    </>
  );
}
```

Exercício

Crie o seu próprio componente contendo **props** e **renderização condicional**.
Seja Criativo e procure utilizar tudo o que foi aprendido até o momento!

INTERVALO DE AULA

DEV!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20
Retorno: 20:40

RENDERIZAÇÃO DE LISTAS

Renderização de listas

Um componente React pode retornar muitos tipos de dados,
incluindo um **array** (o que costumamos chamar de listas).

Neste caso o React “concatena” cada um dos itens da lista e os exibe na tela.

Podemos usar essa funcionalidade para criar componentes a partir
de um array de dados com o método array**.map**

Renderização de listas

Por exemplo ao criar um componente **ListaDeDoces** que retorna um array de doces.

```jsx
export default function App() {

  const docesEmCasa = [
    { nome: 'bala de uva', id: 1 },
    { nome: 'babaloo de uva', id: 59 },
    { nome: 'chocolate', id: 3 },
    { nome: 'biscoitinho', id: 4 },
  ];

  return (
    <>
      <h1>Aula 04</h1>
      <ListaDeDoces listaDeDoces={docesEmCasa}/>
    </>
  );
}
```

```jsx
export default function ListaDeDoces({ listaDeDoces }) {
  return (
    <>
      <h3>Lista de Doces</h3>
      {listaDeDoces.map((doce) => (
        <div key={doce.id} id={doce.id}>
          <p>{doce.nome}</p>
        </div>
      ))}
    </>
  );
}
```

Renderização de listas

E deixando o código mais legível podemos criar um componente ***Doces*** e importá-lo na ***ListaDeDoces***.

```
export default function ListaDeDoces({ listaDeDoces }) {
  return (
    <>
      <h3>Lista de Doces</h3>
      {listaDeDoces.map((doce) => (
        <Doce doce={doce} />
      ))}
    </>
  );
}
```

```
export default function Doce({ doce }) {

  return (
    <div key={doce.id} id={doce.id} className="doce">
      <p>{doce.nome}</p>
    </div>
  );
}
```

Renderização de listas

ATENÇÃO: quando usamos renderização de listas **obrigatoriamente** precisamos passar uma **prop key** para cada um dos elementos.

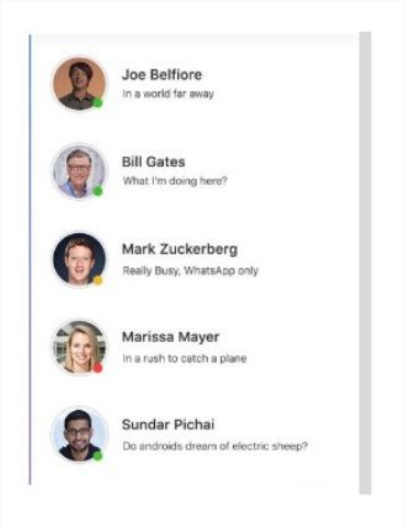
Esta é uma prop especial que o React usa para ***identificar corretamente cada um dos elementos*** no seu algoritmo de diffing (o algoritmo responsável por dizer ao React quando e o que atualizar na tela).

Não podemos usar o **index** ou algo aleatório como key. Porque?
A cada renderização esses valores vão mudar (por exemplo, se mudar de posição na lista), então o React não vai saber qual item era qual antes.

O React precisa saber disso para quando o item mudar de posição (for deletado ou acrescentado, por exemplo) saber exatamente o que precisa alterar na árvore de elementos.

Exercícios

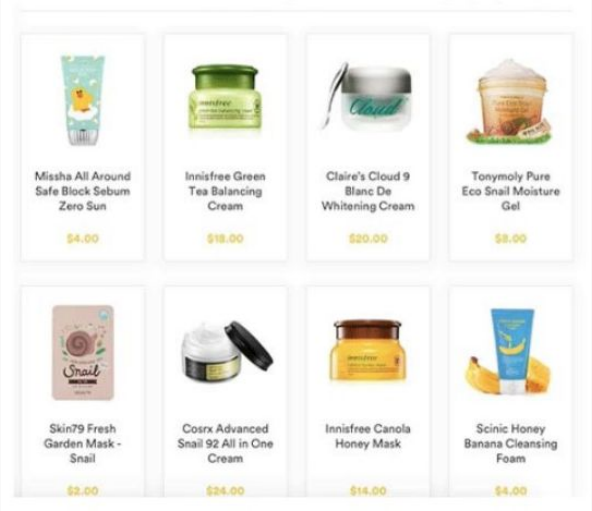
Crie um componente de **lista de contatos**.
Cada **contato** deve conter uma props photo, name e description.
Defina valores padrão para as props que o componente deve receber (defaultProps)



Exercícios

Crie um componente **lista de produtos**.
Cada **produto** contém as props photo, name e price.

Não se esqueça de controlar o tipo de props que o componente pode receber, além de definir propriedades padrão caso necessário.



DEPLOY

EXTRA: Deploy do projeto React com surge

Quando seu projeto React estiver pronto para "**ser visto pelo mundo**" uma maneira simples de fazer o deploy é usando a biblioteca surge

Primeiro certifique-se de já ter **instalado o surge globalmente** (no seu computador).

```
~/DEVinHouse » npm install --global surge
```

Em seguida, dentro do projeto, faça **build** (gere o executável da aplicação):

```
~/DEVinHouse » cd Modulo1_Rosana/REACT/semana-08/aula-02
~/DEVinHouse/Modulo1_Rosana/REACT/semana-08/aula-02 (main*) » npm run build

> aula-02@0.1.0 build /home/rosana/DEVinHouse/Modulo1_Rosana/REACT/semana-08/aula-02
> react-scripts build

Creating an optimized production build...
Compiled successfully.
```

E por fim faça o **deploy** com o surge, seguindo as instruções do terminal.

```
~/DEVinHouse/Modulo1_Rosana/REACT/semana-08/aula-02 (main*) » surge ./build

  Welcome to surge! (surge.sh)
  Login (or create surge account) by entering email & password.

        email: seu-email@gmail.com
     password:

  Running as seu-email@gmail.com (Student)

      project: ./build
       domain: seu-site.surge.sh
       upload: [====================] 100% eta: 0.0s (14 files, 562503 bytes)
          CDN: [====================] 100%
   encryption: *.surge.sh, surge.sh (274 days)
           IP: 138.197.235.123

  Success! - Published to seu-site.surge.sh
```

Material complementar

- Renderização condicional – React
- Listas e Chaves – React
- Array.prototype.map() - JavaScript | MDN

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>