

Aula 2 – Introdução ao Front-End

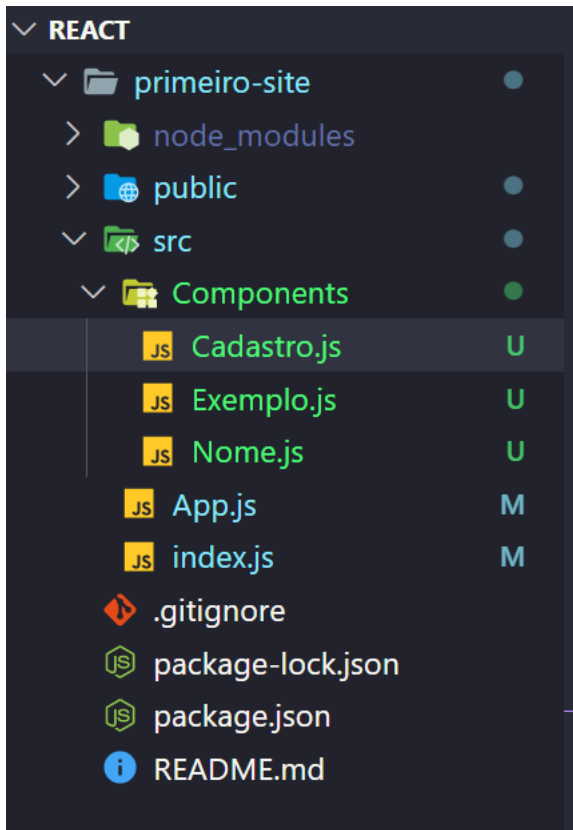
Objetivo da aula:

- Correção da atividade anterior
- Manipular lista por meio de arrays
- React Hooks
- useEffects
- localStorage
- Exercícios (para entregar)

Resposta do exercício 1:

Agora trabalhar com formulários em React para praticar: **Estados, Componentes e Propriedades.**

1 – Crie um novo componente na pasta Componentes chamado Cadastro.js.



2 – Nesse componente crie um formulário que deverá solicitar ao usuário seu nome, e-mail e idade.

```
import { useState } from 'react';
```

```
function Cadastro() {  
  const [nome, setNome] = useState("");  
  const [email, setEmail] = useState("");  
  const [idade, setIdade] = useState("");
```

```
  const [user, setUser] = useState({});
```

```
function handleRegistro(e){
  e.preventDefault();
  setUser({
    nome: nome,
    idade: idade,
    email: email,
  })
}

return (
  <div>
    <form onSubmit={handleRegistro}>
      <label>Nome: </label><br />
      <input placeholder='Digite seu Nome'
        value={nome}
        onChange={ (e) => setNome(e.target.value)}
      /><br />

      <label>Email: </label><br />
      <input placeholder='Digite seu Email'
        value={email}
        onChange={ (e) => setEmail(e.target.value)}
      /><br />

      <label>Idade: </label><br />
      <input placeholder='Digite sua Idade'
        value={idade}
        onChange={ (e) => setIdade(e.target.value)}
      /><br />
    </form>
  </div>
)
```

```

    <button type='submit'>Registro</button>

  </form>

  <br /><br />

  <div>

    <span>Bem vindo, {user.nome}</span><br />

    <span>Idade: {user.idade}</span><br />

    <span>Email: {user.email}</span><br />

  </div>

</div>

);
}

export default Cadastro;

```

3 - Após clicar em registrar as mesmas informações deverão aparecer na tela.

The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page content is divided into two sections. The top section contains a registration form with three input fields labeled 'Nome:', 'Email:', and 'Idade:', each with a placeholder text 'Digite seu Nome', 'Digite seu Email', and 'Digite sua Idade' respectively. Below these fields is a button labeled 'Registro'. The bottom section displays the output of the form, showing 'Bem vindo,', 'Idade:', and 'Email:' followed by empty lines, indicating that the user's information has been successfully registered and is now being displayed back to them.

4 - Deverá existir uma função chamada `handleRegistro` que irá gerenciar o estado das propriedades desse componente criado.

```
primeiro-site > src > Components > JS Cadastro.js > Cadastro
1  import { useState } from 'react';
2
3  function Cadastro() {
4    const [nome, setNome] = useState('');
5    const [email, setEmail] = useState('');
6    const [idade, setIdade] = useState('');
7
8    const [user, setUser] = useState({});
9
10   function handleRegistro(e){
11     e.preventDefault();
12     setUser({
13       nome: nome,
14       idade: idade,
15       email: email,
16     })
17   }
18 }
```

```
19
20   return (
21     <div>
22       <form onSubmit={handleRegistro}>
23         <label>Nome: </label><br />
24         <input placeholder='Digite seu Nome'
25           value={nome}
26           onChange={ (e) => setNome(e.target.value)}
27         /><br />
28
29         <label>Email: </label><br />
30         <input placeholder='Digite seu Email'
31           value={email}
32           onChange={ (e) => setEmail(e.target.value)}
33         /><br />
34
35         <label>Idade: </label><br />
36         <input placeholder='Digite sua Idade'
37           value={idade}
38           onChange={ (e) => setIdade(e.target.value)}
39         /><br />
40
41         <button type='submit'>Registro</button>
42       </form>
43     <br /><br />

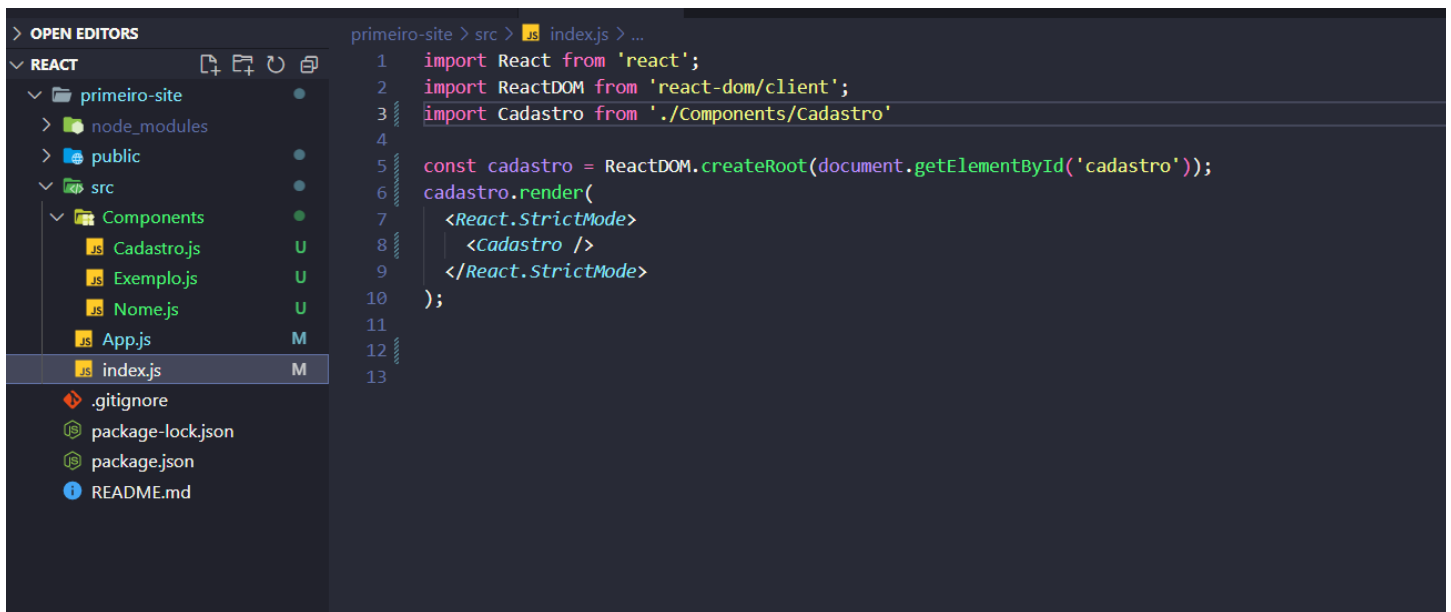
```

```

41     <button type='submit'>Registro</button>
42   </form>
43   <br /><br />
44
45   <div>
46     <span>Bem vindo, {user.nome}</span><br />
47     <span>Idade: {user.idade}</span><br />
48     <span>Email: {user.email}</span><br />
49   </div>
50
51 </div>
52 );
53 }
54 export default Cadastro;
55
56

```

5 – No arquivo index.js crie uma nova renderização para o Componente Cadastro.

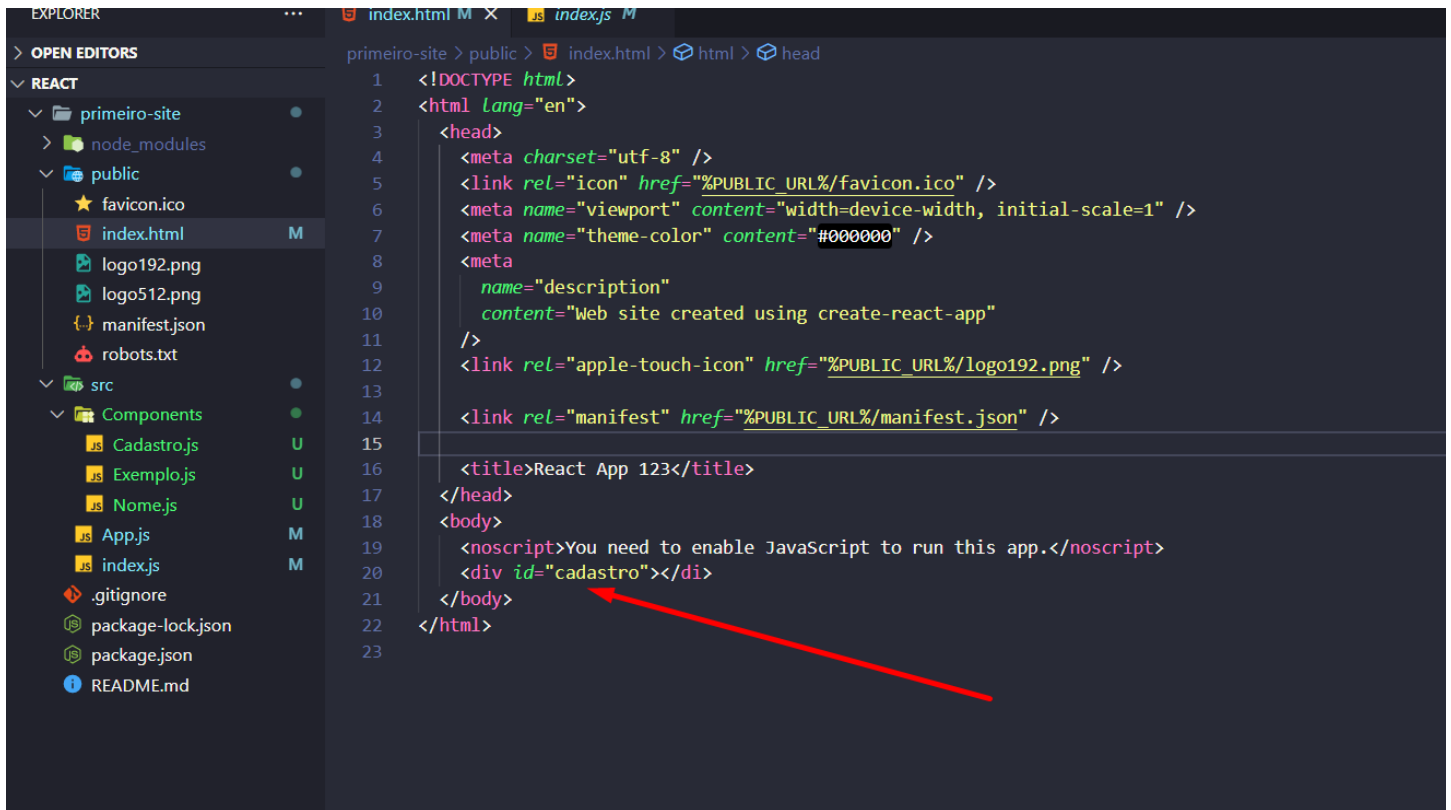


```

primeiro-site > src > index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import Cadastro from './Components/Cadastro'
4
5  const cadastro = ReactDOM.createRoot(document.getElementById('cadastro'));
6  cadastro.render(
7    <React.StrictMode>
8      <Cadastro />
9    </React.StrictMode>
10 );
11
12
13

```

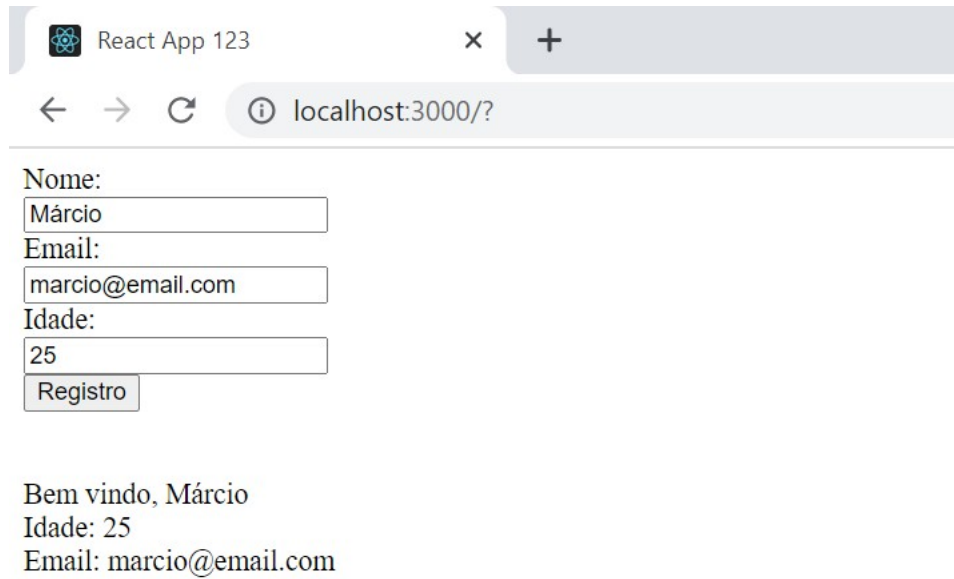
6 – No arquivo index.html chame o cadastro por meio de div



The image shows a VS Code editor interface with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'public' folder containing 'index.html' and 'index.js'. The code editor displays the content of 'index.html', which is an HTML document. A red arrow points to the line containing the opening tag for a div with the id 'cadastro'.

```
1 <!DOCTYPE html>
2 <html Lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6     <meta name="viewport" content="width=device-width, initial-scale=1" />
7     <meta name="theme-color" content="#000000" />
8     <meta
9       name="description"
10      content="Web site created using create-react-app"
11    />
12    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
13
14    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
15
16    <title>React App 123</title>
17  </head>
18  <body>
19    <noscript>You need to enable JavaScript to run this app.</noscript>
20    <div id="cadastro"></div>
21  </body>
22 </html>
23
```

7 – Esse deverá ser o resultado final



React App 123 × +

← → ↻ ⓘ localhost:3000/?

Nome:

Email:

Idade:

Bem vindo, Márcio
Idade: 25
Email: marcio@email.com

Tópico 1 – Trabalhando com Arrays

1 – Vamos alterar nosso arquivo Cadastro.js. Retirando email e idade e agora transformando esse arquivo na possibilidade de cadastrar uma tarefa. Crie um novo cons chamado tarefas que receberá um Array como parametro “useState({})”

```
primeiro-site > src > Components > js Cadastro.js > Cadastro
1  import { useState } from 'react';
2
3  function Cadastro() {
4
5    const [tarefas, setTarefas] = useState([]);
6
7    function handleRegistro(e){
8      e.preventDefault();
9    }
10
11    return (
12      <div>
13        <form onSubmit={handleRegistro}>
14          <label>Nome da tarefa: </label><br />
15          <input placeholder='Digite uma tarefa'
16            value={nome}
17            onChange={ (e) => setNome(e.target.value)}
18          /><br />
19
20          <button type='submit'>Registro</button>
21        </form>
22        <br /><br />
23
24        <div>
25          <span>Bem vindo, {user.nome}</span><br />
26          <span>Idade: {user.idade}</span><br />
27          <span>Email: {user.email}</span><br />
28        </div>
29      </div>
30    );
31  }
32
33  export default Cadastro;
```

2 – Vamos adicionar um novo input e utiliza-lo para dentro do formulário. Faça mais algumas alteração como value={input} e etc conforme documento abaixo:

```
primeiro-site > src > Components > JS Cadastro.js > Cadastro
1  import { useState } from 'react';
2
3  function Cadastro() {
4    const [input, setInput] = useState('');
5    const [tarefas, setTarefas] = useState({});
6
7    function handleRegistro(e){
8      e.preventDefault();
9    }
10
11    return (
12      <div>
13        <form onSubmit={handleRegistro}>
14          <label>Nome da tarefa: </label><br />
15          <input placeholder='Digite uma tarefa'
16            value={input}
17            onChange={ (e) => setInput(e.target.value)}
18          /><br />
19
20          <button type='submit'>Registro</button>
21        </form>
22      <br /><br />
23
24      <div>
25
26      </div>
27
28    </div>
29  );
30 }
31 export default Cadastro;
32
```

3 – Vamos adicionar uma título ao formulário usando <h1> chamando de Cadastro de Tarefas

```
primeiro-site > src > Components > js Cadastro.js > Cadastro
1  import { useState } from 'react';
2
3  function Cadastro() {
4    const [input, setInput] = useState('');
5    const [tarefas, setTarefas] = useState({});
6
7    function handleRegistro(e){
8      e.preventDefault();
9    }
10
11   return (
12     <div>
13       <h1>Cadastro de Tarefas</h1>
14
15       <form onSubmit={handleRegistro}>
16         <label>Nome da tarefa: </label><br />
17         <input placeholder='Digite uma tarefa'
18           value={input}
19           onChange={ (e) => setInput(e.target.value)}
20         /><br />
21
22         <button type='submit'>Registro</button>
23       </form>
24       <br /><br />
25     </div>
26   );
27 }
28
29 export default Cadastro;
```


4 – Rode o projeto, sua página deverá estar assim agora:



The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page title is 'Cadastro de Tarefas'. Below the title, there is a form with the label 'Nome da tarefa:' followed by a text input field containing the placeholder text 'Digite uma tarefa'. Below the input field is a button labeled 'Registro'.

5 – Vamos agora criar uma lista de tarefas que o usuário poderá escolher.
Altere o useState para receber um array []

```
primeiro-site > src > Components > JS Cadastro.js > Cadastro
1  import { useState } from 'react';
2
3  function Cadastro() {
4    const [input, setInput] = useState('');
5    const [tarefas, setTarefas] = useState([]);
6
7    function handleRegistro(e){
8      e.preventDefault();
9    }
10
11   return (
12     <div>
13       <h1>Cadastro de Tarefas</h1>
14
15       <form onSubmit={handleRegistro}>
16         <label>Nome da tarefa: </label><br />
17         <input placeholder='Digite uma tarefa'
18           value={input}
19           onChange={ (e) => setInput(e.target.value)}
20       </form>
21     </div>
22   );
23 }
```



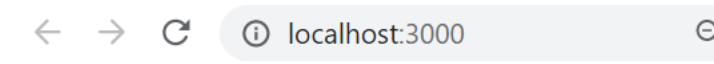
6 – Desse modo estamos preenchendo esse array:

```
primeiro-site > src > Components > JS Cadastro.js > Cadastro
1  import { useState } from 'react';
2
3  function Cadastro() {
4    const [input, setInput] = useState('');
5    const [tarefas, setTarefas] = useState([
6      "Pagar a conta de luz",
7      "Estudar Programação",
8      "Enviar a tarefa"
9    ]);
10
11   function handleRegistro(e){
12     e.preventDefault();
13   }
14 }
```

7 – Nós temos um array chamado tarefas (linha 5) que agora podemos exibir dentro da página do site. Vamos criar uma lista não ordenada, utilize para isso a tag e dentro dela vamos chamar nosso array tarefas e por meio da função 'map' podemos percorrer todo o array tarefas. Dentro da função map criamos o argumento tarefa que ao mapear um novo valor dentro do array irá gerar uma lista com a tag

```
24
25     <button type='submit'>Registro</button>
26   </form>
27   <br /><br />
28
29   <ul>
30     {tarefas.map( tarefa => (
31       <li>{tarefa}</li>
32     ) )}
33   </ul>
34
35 </div>
36 );
37 }
38 export default Cadastro;
39
40
```

8 – Sua página deverá exibir a lista mapeada:



Cadastro de Tarefas

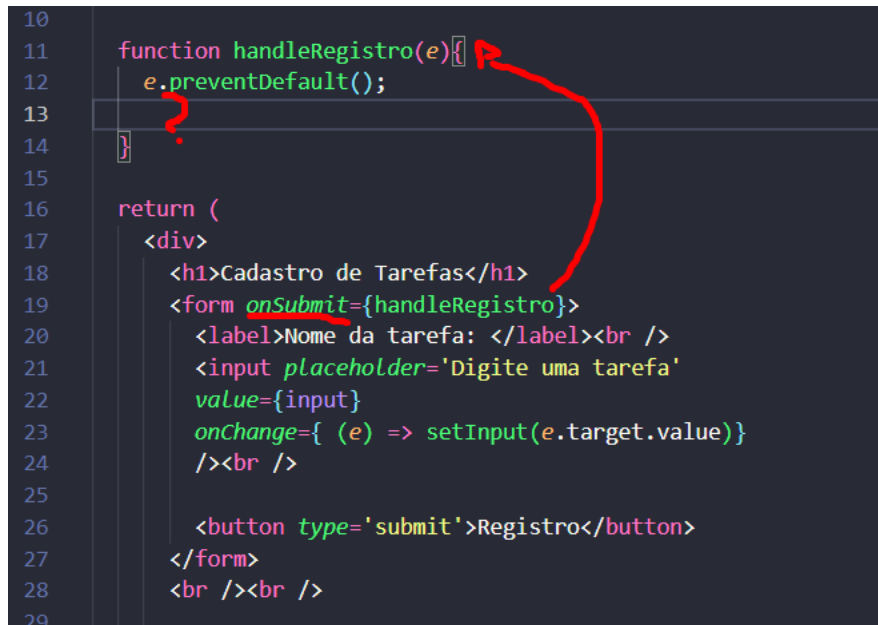
Nome da tarefa:

- Pagar a conta de luz
- Estudar Programação
- Enviar a tarefa

9 – Bom, queremos agora poder registrar novas tarefas dentro do array criado, certo?

Mas onde fazemos isso, ao olhar nosso código perceba que ao clicar no botão Registro ele chama a função `handleRegister`, então é ali que precisamos adicionar novos elementos ao meu array.

```
10
11  function handleRegistro(e){
12    e.preventDefault();
13  }
14
15
16  return (
17    <div>
18      <h1>Cadastro de Tarefas</h1>
19      <form onSubmit={handleRegistro}>
20        <label>Nome da tarefa: </label><br />
21        <input placeholder='Digite uma tarefa'
22          value={input}
23          onChange={ (e) => setInput(e.target.value)}
24        /><br />
25
26        <button type='submit'>Registro</button>
27      </form>
28      <br /><br />
29    </div>
  )
}
```

A red arrow originates from the 'Registro' button in the JSX (line 26) and points to the 'handleRegistro' function definition (line 11). A red question mark is placed near the arrow's start, highlighting the connection between the UI element and the logic function.

10 – Vamos chamar nosso array tarefas, definido anteriormente, pela sua função de setTarefas (linha 14). Como já nós já temos 3 objetos dentro do meu array usamos o **spread operator**, aqui são os 3 pontinhos, esse spread faz uma cópia dos objetos já existentes dentro do array e assim o próximo objeto não irá substituir o conteúdo do array.

```
primeiro-site > src > Components > JS Cadastro.js > Cadastro > handleRegistro
1  import { useState } from 'react';
2
3  function Cadastro() {
4    const [input, setInput] = useState('');
5    const [tarefas, setTarefas] = useState([
6      "Pagar a conta de luz",
7      "Estudar Programação",
8      "Enviar a tarefa"
9    ]);
10
11    function handleRegistro(e){
12      e.preventDefault();
13
14      setTarefas([...tarefas, input])
15    }
16
17    return (
18      <div>
19        <h1>Cadastro de Tarefas</h1>
20        <form onSubmit={handleRegistro}>
```

11 – Veja como ficará nossa aplicação sem utilizar o spread:

←

→

↺

📄 localhost:3000

🔍

🔗

☆

NEW

Cadastro de Tarefas

Nome da tarefa:

dormir

Registro

Pagar a conta de luz

Estudar Programação

Enviar a tarefa

dormir

- Pagar a conta de luz
- Estudar Programação
- Enviar a tarefa
- dormir

12 – Agora sim, se o código estiver igual ao item 10 você será capaz de adicionar novos objetos dentro do array tarefas:



A screenshot of a web browser window at localhost:3000. The page title is "Cadastro de Tarefas". Below the title, there is a label "Nome da tarefa:" followed by a text input field containing the word "dormir". Below the input field is a button labeled "Registro". Underneath the button, there is a bulleted list of tasks: "Pagar a conta de luz", "Estudar Programação", "Enviar a tarefa", and "dormir".

13 – Perceba que no exemplo anterior não sumiu de dentro do campo, após ser adicionado. Para corrigir isso basta adicionar um setInput vazio dentro da função handleRegistro.

```
9      ],  
10  
11      function handleRegistro(e){  
12          e.preventDefault();  
13  
14          setTarefas([...tarefas, input]);  
15          setInput('');  
16      }  
17  
18      return (
```



14 – Opa, temos um problema na nossa lista. Vá no seu navegador, aperte F12 e vá até o item Console. Veja o erro abaixo:

```
✖ ▶Warning: Each child in a list should have a unique "key" prop.  
    Check the render method of `Cadastro`. See https://reactjs.org/link/warning-keys for more information.  
    at li  
    at Cadastro (http://localhost:3000/static/js/bundle.js:27:76)
```

Aqui o React diz que sempre que utilizamos uma lista (array) precisamos usar a propriedade key para que cada objeto da lista seja única.

15 – Para corrigir o erro adicione a prop key desse modo:

```
27  
28     <button type='submit'>Registro</button>  
29   </form>  
30   <br /><br />  
31  
32   <ul>  
33     {tarefas.map( tarefa => (  
34       <li key={tarefa}>{tarefa}</li>  
35     ))}  
36   </ul>  
37  
38 </div>  
39 );
```



Tópico 2 – React Hooks

Anteriormente usamos o `useState` para definir uma mudança de estado de um elemento dentro do React.

Dentro do react existem diversos modos de alterar os estados de alguma aplicação Até a versão 16.7 do React, algumas **funcionalidades** só eram possíveis de ser acessadas através de classes (como, por exemplo, o *lifecycle*). Na versão 16.8 do React, foram lançado os **hooks**, que permitem o uso de vários recursos de forma simples através de funções. Eles também ajudam a **organizar a lógica** utilizada dentro dos **componentes**.

Os Hooks resolvem uma grande variedade de problemas encontrados durante o desenvolvimento de componentes. Por exemplo:

- Reutilização de lógica de estado entre components;
- Wrapper Hell (HOC, Render props — React DevTools);
- Componentes complexos e difíceis de se compreender;
- Componentes contendo grandes responsabilidades;
- Confusão ao utilizar classes (`this`, classes).

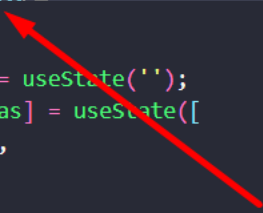
O `useState`, utilizado agora pouco é um exemplo de Hooks dentro do React. Existem outros, veja alguns exemplos:

<https://reactjs.org/docs/hooks-reference.html>

Bora testar outro Hook? Vamos agora testar o useEffects.

1 – Importe a useEffects para utilizamos em nossa aplicação.

```
primeiro-site > src > Components > JS Cadastro.js > ...  
1 import { useState, useEffect } from 'react';  
2  
3 function Cadastro() {  
4   const [input, setInput] = useState('');  
5   const [tarefas, setTarefas] = useState([  
6     "Pagar a conta de luz",  
7     "Estudar Programação",  
8     "Enviar a tarefa"  
9   ]);  
10  
11   function handleRegistro(e){  
12     e.preventDefault();  
13
```

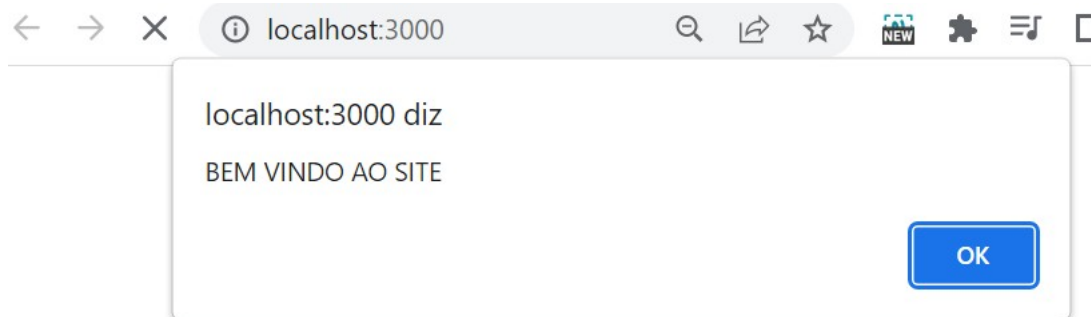


2 – chame a useEffects do modo abaixo. Como ainda não sabemos como iremos usar, utilize o recurso =>{} para chamar uma função anônima (que será definida depois) e um array vazio [].

```
6     "Pagar a conta de luz",  
7     "Estudar Programação",  
8     "Enviar a tarefa"  
9   ]);  
10  
11   useEffect(() =>{}, []);  
12  
13   function handleRegistro(e){  
14     e.preventDefault();  
15  
16     setTarefas([...tarefas, input]);
```

3 – Toda vez que a página renderiza nossa função App para o usuário esse Hook irá ser chamado. Vamos testar, adicione um alert, dê um F5 na página e veja que o alert irá aparecer.

```
6   "Pagar a conta de luz",  
7   "Estudar Programação",  
8   "Enviar a tarefa"  
9   ]);  
10  
11   useEffect(() =>{  
12     alert('BEM VINDO AO SITE')  
13   }, []);  
14  
15   function handleRegistro(e){  
16     e.preventDefault();  
17  
18     setTarefas([...tarefas, input]);  
19     setInput('');
```



4 – Ok, mas pra que a `useEffects` serve? Bom, toda vez que o estado de algum componente mudar, podemos dizer para o React ter uma nova ação. Ao chamar nosso array 'tarefas' para dentro desse `useEffect` toda vez que um novo elemento for adicionado a nossa lista o que estiver na linha 12 (dentro do `useEffect`) vai ser também executado.

```
7      "Estudar Programação",  
8      "Enviar a tarefa"  
9    ]);  
10  
11    useEffect( () =>{  
12      |  
13    }, [tarefas]);  
14  
15    function handleRegistro(e){  
16      e.preventDefault();  
17
```

5 – Vamos testar, faça com que toda vez que alguém adicionar um novo elemento a lista (array) um alert apareça na tela.

```
6    "Pagar a conta de luz",  
7    "Estudar Programação",  
8    "Enviar a tarefa"  
9  ]);  
10  
11  useEffect( () =>{  
12    alert('Tarefa adicionada com sucesso')  
13  }, [tarefas]);  
14  
15  function handleRegistro(e){  
16    e.preventDefault();  
17  
18    setTarefas([...tarefas, input]);  
19    setInput('');  
20  }
```

6 – Mas pra que isso é útil?

Imagine que um usuário está inserindo várias tarefas em uma lista e de repente ele ficou sem internet e tudo que ele fez foi perdido e precisará recomeçar pois o banco de dados não recebeu essas novas tarefas :(

Ou ainda, ele fechou o navegador ou a página foi atualizada e tudo foi perdido. Já aconteceu isso com você?

Para evitar isso podemos usar localStorage :D

O Local Storage é um espaço reservado pelos browsers para o salvamento de informações localmente.

Essa é uma alternativa interessante pra quando você precisa salvar informações, mas não precisa que isso fique salvo em uma API.

O LocalStorage é uma forma de salvar dados no computador do cliente. Ele permite que salvemos pares de chaves e valores no web browser sem uma data de expiração.

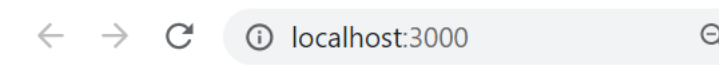
Essa forma de armazenamento só pode ser acessada via JavaScript e HTML5, mas é importante saber que o usuário pode limpar os dados/cache do browser se quiser.

7 – Bora testar, dentro do useEffects vamos então fazer com que toda vez que um novo valor seja adicionado a lista ele será salvo localmente no navegador do cliente para não se perder caso a página seja fechada.

```
8     "Enviar a tarefa"
9   ]);
10
11   useEffect( () =>{
12     localStorage.setItem('@tarefa', JSON.stringify(tarefas));
13   }, [tarefas]);
14
15   function handleRegistro(e){
16     e.preventDefault();
17   }
```

O @tarefa será a key de identificação desse localStorage. O localStorage só aceita JSON como informação, portanto, usando o JSON.stringify podemos converter um array tarefas para JSON.

8 – Será que ele salvou? Como verificar isso. Adicione mais um elemento dentro da lista para o useEffects ser chamado e assim executar o localStorage.

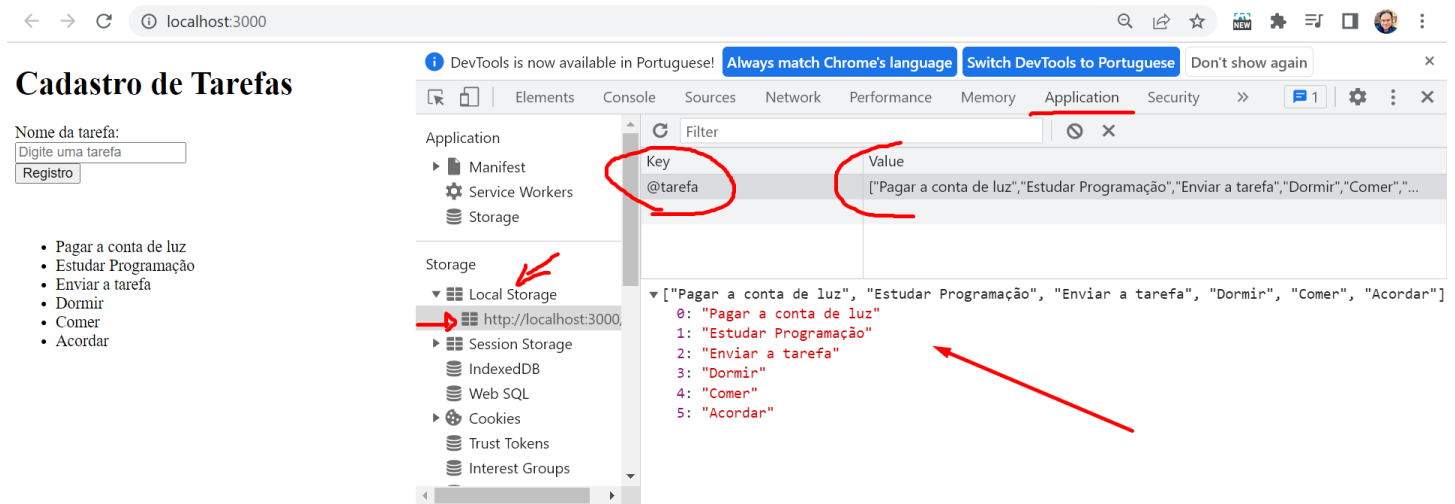


Cadastro de Tarefas

Nome da tarefa:

- Pagar a conta de luz
- Estudar Programação
- Enviar a tarefa
- Dormir
- Comer
- Acordar

9 – Aperte F12, vá em Application, Local Storage, <http://localhost:3000>. E olha lá quem está salvo, um JSON contendo todo o array, demais!



10 – Porém, veja que se você atualizar a página ou fechar e abrir novamente o site o que adicionamos a mais na lista sumiu :(

Para salvar no localStorage usamos o `localStorage.setItem('@tarefa', JSON.stringify(tarefas));`

Agora, vamos usar fazer pegar o que está no localStore usando `localStorage.getItem` criando uma variável chamada `tarefasStorage`:

```
primeiro-site > src > Components > JS Cadastro.js > Cadastro
5   const [tarefas, setTarefas] = useState([
6     "Pagar a conta de luz",
7     "Estudar Programação",
8     "Enviar a tarefa"
9   ]);
10
11   const tarefasStorage = localStorage.getItem('@tarefa');
12
13   useEffect( () =>{
14     localStorage.setItem('@tarefa', JSON.stringify(tarefas));
15   }, [tarefas]);
16
17   function handleRegistro(e){
18     e.preventDefault();
19
20     setTarefas([...tarefas, input]);
21     setInput('');
22   }
23
24
25
26
```


11 – Porém, apenas isso não basta pois nós já temos itens na lista. Portanto, vamos usar novamente o `useEffect` para que quando a página seja carregada ele verifique se já existem coisas no `localStorage`, e se sim, ele use o que está lá dentro da página:

```
primeiro-site > src > Components > js Cadastro.js > Cadastro
5   const [tarefas, setTarefas] = useState([
6     "Pagar a conta de luz",
7     "Estudar Programação",
8     "Enviar a tarefa"
9   ]);
10
11   const tarefasStorage = localStorage.getItem('@tarefa');
12
13   useEffect(() =>{
14     if(tarefasStorage){
15       setTarefas(JSON.parse(tarefasStorage));
16     }
17   }, []);
18
19   useEffect( () =>{
20
21     localStorage.setItem('@tarefa', JSON.stringify(tarefas));
22
23
24
```

O `if` verifica se existe informações no `localStorage`, no nosso caso sim, então ele usa a função `setTarefas` para que nossa variável `tarefas` receba o que estiver lá e exiba na tela.

12 – Pronto, agora você pode atualizar a página ou fechar a aba e abrir novamente o endereço <http://localhost:3000/> que a lista não irá sumir.



The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page title is 'Cadastro de Tarefas'. Below the title, there is a form with the label 'Nome da tarefa:' followed by a text input field containing the placeholder 'Digite uma tarefa' and a 'Registro' button. Below the form, there is a bulleted list of tasks: 'Pagar a conta de luz', 'Estudar Programação', 'Enviar a tarefa', and 'Dormir'.

← → ↻ ⓘ localhost:3000 🔍 ↗

Cadastro de Tarefas

Nome da tarefa:

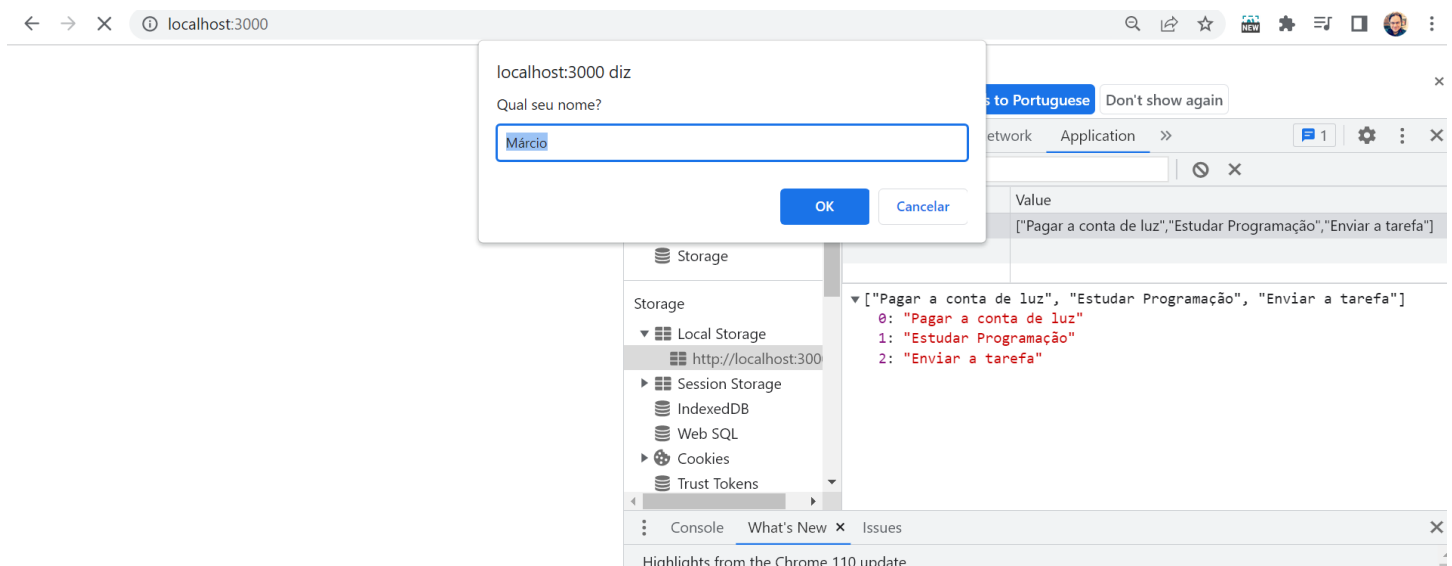
Registro

- Pagar a conta de luz
- Estudar Programação
- Enviar a tarefa
- Dormir

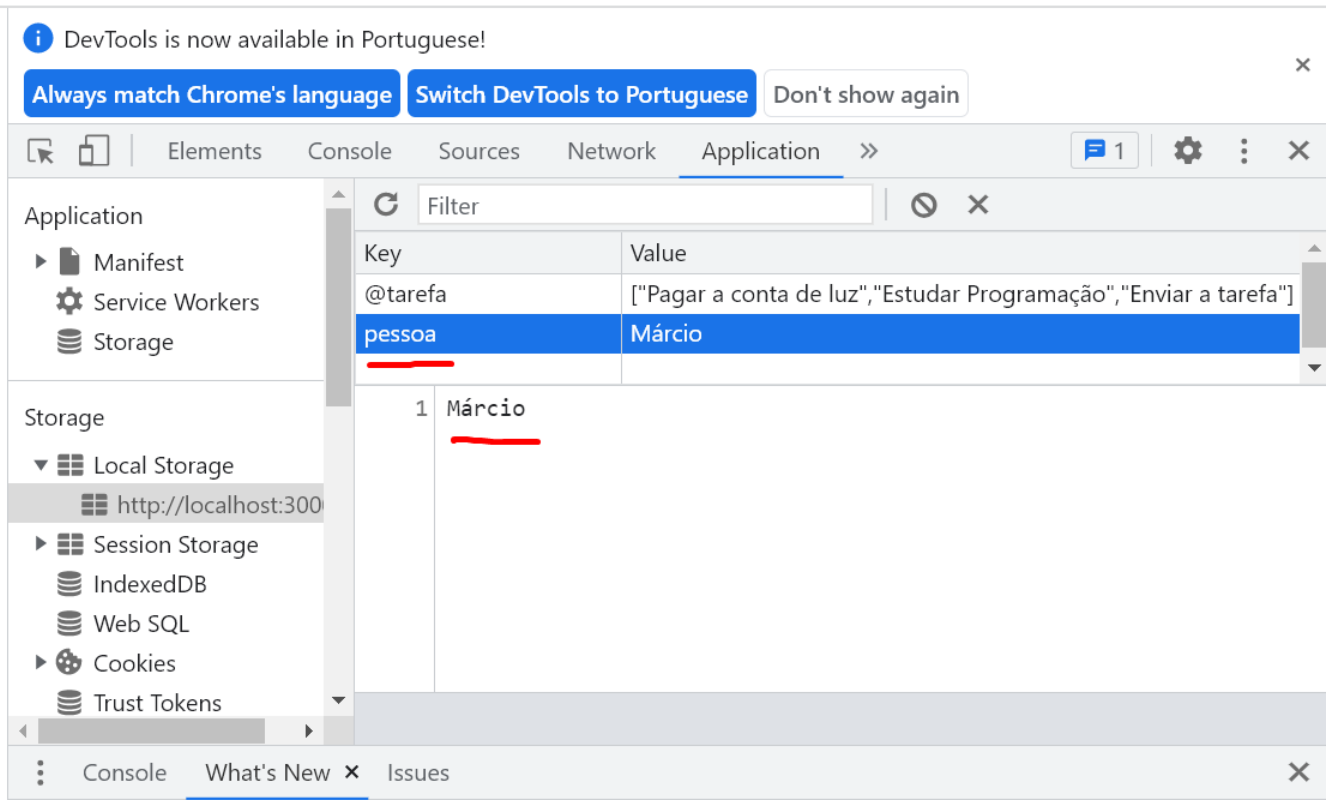
Exercício

Utilizando o mesmo projeto, faça as seguintes mudanças:

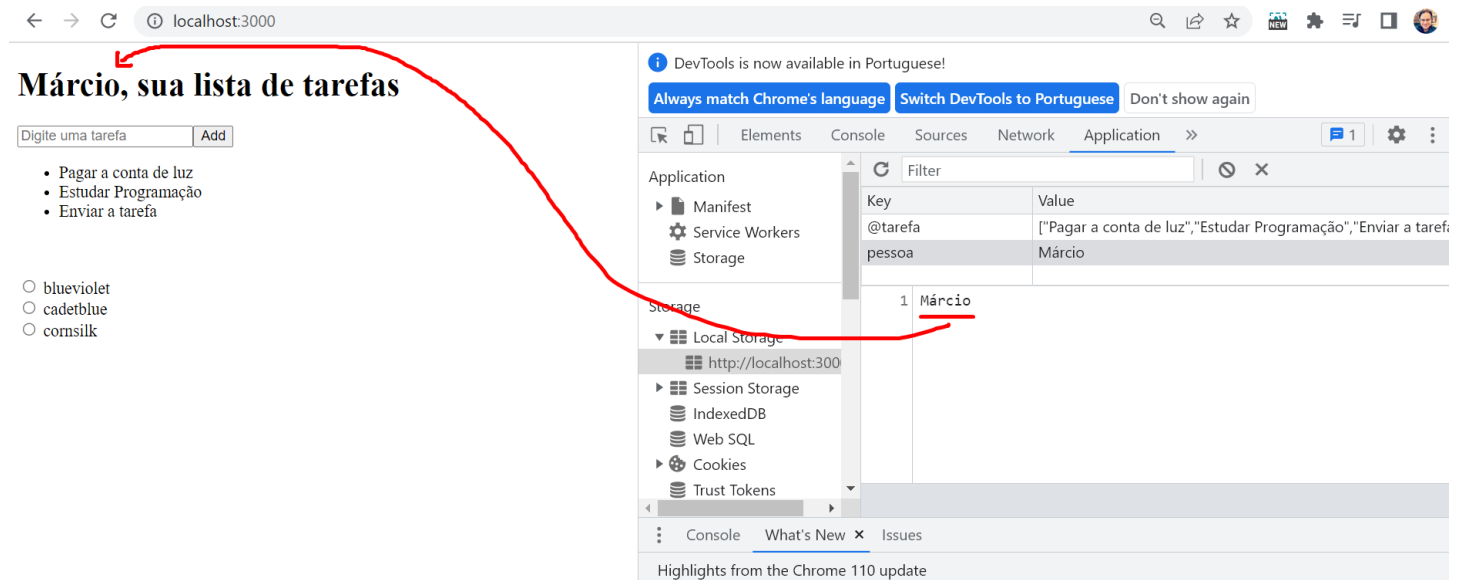
A – Utilizando Hooks e localStorage, ao usuário entrar na página um alert deve aparecer perguntando ao usuário seu nome.



B – Ao usuário informar seu nome o valor deve ser salvo no localStorage



C – Usando Hooks, faça a leitura do valor salvo no localState e insira na tela o nome do usuário e depois a frase “, sua lista de tarefas”



D – Crie alguns Radio Buttons com nome de cores. Utilize a documentação do React: <https://reactnative.dev/docs/colors>



E – Ao usuário escolher uma cor o fundo da página deverá receber a mesma cor

