



## Fundamentos React - Projeto 3 Gerador de Meme

### O que vamos aprender

- Event Listener
- State
- Side Effects

### 1. Criando seu app

Crie uma pasta chamada `react`. Recomendamos que esta pasta seja criada dentro da pasta que foi criada para o curso.

Execute `npx create-react-app meme-generator`, dê o comando `cd` em seu projeto e abra-o. No componente App.js Você pode excluir tudo na instrução `return` de forma que vai retornar apenas um elemento div vazio. Você também pode excluir todos os templates create-react-app fornecidos e apenas deixar `index.js` e `App.js` no diretório src.

### Desafio 1 - Projeto :

- Crie um componente com o nome de Header.

Resultado desejado:



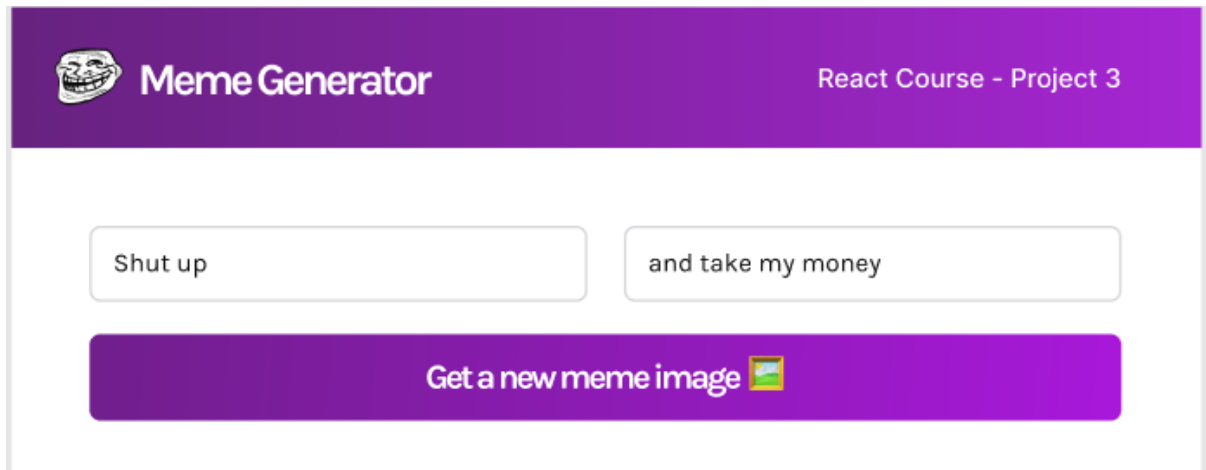
### Desafio 2 - Projeto :

- Crie um componente chamado Meme;
- Dentro do componente Meme, renderize um formulário estilizado com as 2 entradas (input) e o botão;



- Não se preocupe em adicionar nenhuma funcionalidade ainda.

Resultado desejado:



### Desafio 3 - Projeto - Event Listener:

- Obter uma imagem aleatória do array `memesData` \* quando o botão "Get a new meme image" é clicado;
- \*Registre a URL da imagem no console. (Não se preocupe \* sobre a exibição da imagem ainda).

Meme.js

```
import React from "react"

import memesData from "../memesData.js"

export default function Meme() {

  let url
```



```
function getMemeImage() {

    const memesArray = memesData.data.memes

    const randomNumber = Math.floor(Math.random() *
memesArray.length)

    url = memesArray[randomNumber].url

    console.log(url)

}

return (

    <main>

        <p>{url}</p>

        <div className="form">

            <input

                type="text"

                placeholder="Top text"

                className="form-input"

            />

            <input

                type="text"

                placeholder="Bottom text"

                className="form-input"

            />


            <button
```



```
        className="form-button"

        onClick={getMemeImage}

    >

        Get a new meme image 

    </button>

</div>

</main>

)

}
```

#### Desafio 4 - Labs - Event Listener:

- Mapeie o array thingsArray para gerar um elemento <p> para cada item e renderizá-los na página abaixo do botão.

index.js

```
import React from 'react';

import ReactDOM from 'react-dom';

function App() {

    const thingsArray = ["Thing 1", "Thing 2"]

    /*Coloque seu código aqui */

    return (
```



```
    <div>

        <button>Add Item</button>

        { /* insira as coisas aqui*/ }

    </div>

)

}

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

styles.css

```
* {

    box-sizing: border-box;

}

body {

    background-color: #70B85D;

    color: white;

    padding: 20px;

    font-family: 'Karla', sans-serif;

    font-size: 1.3rem;

}
```



```
button {

    width: 100%;

    background-color: transparent;

    padding: 1rem;

    color: white;

    border: 2px solid white;

    border-radius: 50px;

    cursor: pointer;

    font-family: 'Karla', sans-serif;

    margin-bottom: 20px;

}

button:hover {

    background-color: #FFFEF1;

    color: #2C5E2E;

}

button:focus {

    outline: 0;

}
```



### Desafio 5 - Labs - Event Listener - necessidade da atualização do state - useState:

- Adicione um event listener ao botão para que, quando for clicado, adiciona outra 'coisa' ao nosso array;

**Dica:** use o comprimento do array + 1 (array.length+1) para determinar o índice da "Coisa" que está sendo adicionada.

Além disso, veja como fica o conteúdo do array, utilizando console.log(thingsArray), depois de adicionar o novo item ao array.

**Spoiler:** a página não será atualizada quando novas coisas forem adicionadas a array!

index.js

```
import React from 'react';

import ReactDOM from 'react-dom';

function App() {

  const thingsArray = ["Thing 1", "Thing 2"]

  function addItem() {

    const newThingText = `Thing ${thingsArray.length + 1}`

    thingsArray.push(newThingText)

    console.log(thingsArray)

  }

}
```



```
const thingsElements = thingsArray.map(thing => <p
key={thing}>{thing}</p>)

return (

  <div>

    <button onClick={addItem}>Add Item</button>

    {thingsElements}

  </div>

)
}

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

## O que são props?

"Props" refere-se às propriedades que estão sendo passadas para um componente para que ele funcione corretamente, semelhante a como uma função recebe parâmetros: "de cima". Um componente que recebe props não tem permissão para modificar esses props. Eles são "imutáveis".

## O que é state?

"State" refere-se a valores que são gerenciados pelo componente, semelhante a variáveis declaradas dentro de uma função. Sempre que você alterar valores que devem ser salvos/exibidos, provavelmente estará usando state.

## Desafio 6 - Labs - Props vs State :

- Crie uma função chamada `handleClick` que execute `setIsImportant("Yes")`;





- Adicione um event listener de clique à classe *div.state--value* que executa *handleClick* quando a *div* é clicada.

App.js

```
import React from "react"

import {useState} from "react"

export default function App() {

  const [isImportant, setIsImportant] = useState("Yes")

  function handleClick() {

    setIsImportant("No")

  }

  return (

    <div className="state">

      <h1 className="state-title">Is state important to
know?</h1>

      <div className="state-value" onClick={handleClick}>

        <h1>{isImportant}</h1>

      </div>

    </div>

  )
```



styles.css

```
* {  
  
  box-sizing: border-box;  
  
}  
  
body {  
  
  margin: 0;  
  
  font-family: 'Inter', sans-serif;  
  
  background-color: #262626;  
  
  color: #D9D9D9;  
  
  padding: 20px;  
  
  height: 100vh;  
  
  display: flex;  
  
  justify-content: center;  
  
  align-items: center;  
  
}  
  
.state {  
  
  display: flex;  
  
  flex-direction: column;  
  
  align-items: center;
```



```
}  
  
.state-title {  
  text-align: center;  
}  
  
.state-value {  
  background-color: white;  
  height: 100px;  
  width: 100px;  
  border-radius: 50%;  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  color: #262626;  
}
```

### Desafio 6 - Labs - useState - prática contador :

- Configure o estado (state) para rastrear nossa contagem (o valor inicial é 0)

App.js

```
import React from "react"
```



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

export default function App() {

  const [count, setCount] = useState(0)

  return (

    <div className="counter">

      <button className="counter-minus">-</button>

      <div className="counter-count">

        <h1>{count}</h1>

      </div>

      <button className="counter-plus">+</button>

    </div>

  )
}
```

styles.css

```
* {
  box-sizing: border-box;
}
```



```
body {  
  
    margin: 0;  
  
    font-family: 'Inter', sans-serif;  
  
    background-color: #262626;  
  
    color: #D9D9D9;  
  
    padding: 20px;  
  
    height: 100vh;  
  
    display: flex;  
  
    justify-content: center;  
  
    align-items: center;  
  
}  
  
.counter {  
  
    display: flex;  
  
    align-items: flex-end;  
  
}  
  
.counter > button {  
  
    height: 50px;  
  
    width: 50px;  
  
    border-radius: 50%;  
  
    border: none;
```



```
cursor: pointer;

background-color: #737373;

color: #D9D9D9;

font-size: 1.5rem;
}

.counter > button:hover {

    background-color: #404040;

    color: #D9D9D9;
}

.counter > button:focus {

    outline: none;
}

.counter-count {

    background-color: white;

    height: 100px;

    width: 100px;

    border-radius: 50%;

    display: flex;

    justify-content: center;
```



```
align-items: center;

color: #262626;

}

.counter-plus {

margin-left: -20px;

}

.counter-minus {

margin-right: -20px;

z-index: 1;

}
```

### Desafio 7 - Labs - useState - prática contador :

- Crie uma função chamada `add` que é executada quando o botão + é clicado. (Pode apenas console.log("add") por enquanto)
- Crie uma função chamada `subtract` que é executada quando o botão - é clicado. (Pode apenas console.log("subtract") por enquanto)

App.js

```
import React from "react"

import { useState } from 'react';

export default function App() {
```



```
const [count, setCount] = useState(0)

function add() {
  setCount(count + 1)
}

function subtract() {
  setCount(count - 1)
}

return (
  <div className="counter">
    <button className="counter-minus"
onClick={subtract}>-</button>
    <div className="counter-count">
      <h1>{count}</h1>
    </div>
    <button className="counter-plus"
onClick={add}>+</button>
  </div>
)
```





**Observação:** se você precisar do valor antigo do state para ajudá-lo a determinar o novo valor de state, passe uma função de callback para sua função de configuração de estado em vez de usar state diretamente. Essa função de callback receberá o antigo valor de estado como seu parâmetro, que você pode usar para determinar seu novo valor de estado.

### Desafio 8 - Labs - mudando state com callback function :

- atualizar `add` e `subtract` para usar uma função de retorno de chamada (callback).

App.js

```
import React from "react"

import { useState } from 'react';

export default function App() {

  const [count, setCount] = useState(0)

  function add() {

    setCount(prevCount => prevCount + 1)

  }

  function subtract() {

    setCount(prevCount => prevCount - 1)
```



```
}

return (

  <div className="counter">

    <button className="counter-minus"
onClick={subtract}>-</button>

    <div className="counter-count">

      <h1>{count}</h1>

    </div>

    <button className="counter-plus"
onClick={add}>+</button>

  </div>

)

}
```

### Desafio 9 - Projeto - adicione imagens ao gerador de meme.

- Salve a URL do meme aleatório no estado (state);
- Crie um novo estado(state) chamado `memelImage` com uma string vazia como padrão;
- Quando a função getMemelImage é chamada, atualize o estado `memelImage` para ser a URL da imagem escolhida aleatoriamente;
- Abaixo do div.form, adicione um <img /> e defina o src para o novo estado `memelImage` que você criou.



Meme.js

```
import React from "react";

import {useState} from "react";

import memesData from "../memesData.js";

export default function Meme() {

  const [memeImage, setMemeImage] = useState("")

  function getMemeImage() {

    const memesArray = memesData.data.memes

    const randomNumber = Math.floor(Math.random() *
memesArray.length)

    setMemeImage(memesArray[randomNumber].url)

  }

  return (

    <main>

      <div className="form">

        <input

          type="text"

          placeholder="Top text"


```



```
        className="form-input"

    />

    <input

        type="text"

        placeholder="Bottom text"

        className="form-input"


    />

    <button

        className="form-button"

        onClick={getMemeImage}

    >

        Get a new meme image 

    </button>

</div>

<img src={memeImage} className="meme-image" />

</main>

)

}
```

styles.css [ acrescentar ]

```
.meme-image {

    max-width: 100%;
```



## Desafio 10 - Projeto - Refatorando state.

- Atualize nosso estado para salvar os dados relacionados ao meme como um objeto chamado `meme`. Deve ter as seguintes 3 propriedades: topText, bottomText, randomImage;
- Os 2 estados de texto podem ser padronizados para strings vazias por enquanto, e *randomImage* deve ser padronizado para ["http://i.imgflip.com/1bij.jpg"](http://i.imgflip.com/1bij.jpg);
- Em seguida, crie uma nova variável de estado chamada `allMemelImages`, cujo padrão será `memesData`, que importamos acima;
- Por fim, atualize a função `getMemelImage` e a marcação para refletir nosso objeto de estado recém-reformado e array da maneira correta.

Meme.js

```
import React from "react"

import {useState} from "react"

import memesData from "../memesData.js"

export default function Meme() {

  const [meme, setMeme] = useState({

    topText: "",

    bottomText: "",

    randomImage: "http://i.imgflip.com/1bij.jpg"

  })
```



```
const [allMemeImages, setAllMemeImages] =
useState(memesData)

function getMemeImage() {

  const memesArray = allMemeImages.data.memes

  const randomNumber = Math.floor(Math.random() *
memesArray.length)

  const url = memesArray[randomNumber].url

  setMeme(prevMeme => ({

    ...prevMeme,

    randomImage: url

  })))

}

return (

  <main>

    <div className="form">

      <input

        type="text"

        placeholder="Top text"

        className="form-input"

      />
```

```
<input
  type="text"
  placeholder="Bottom text"
  className="form-input"
/>

<button
  className="form-button"
  onClick={getMemeImage}
>
  Get a new meme image 
</button>

</div>

<img src={meme.randomImage} className="meme-image"
/>

</main>

)
}
```

### Desafio 11 - Projeto - Adicionando texto na imagem.

- Configure as entradas de texto para salvar as variáveis de estado `topText` e bottomText`;`
- Substitua o texto codificado na imagem pelo texto salvo no estado (state).

Meme.js



```
import React from "react";

import {useState} from "react";

import memesData from "../memesData.js";

export default function Meme() {

  const [meme, setMeme] = useState({

    topText: "",

    bottomText: "",

    randomImage: "http://i.imgflip.com/1bij.jpg"

  })

  const [allMemeImages, setAllMemeImages] =
React.useState(memesData)

  function getMemeImage() {

    const memesArray = allMemeImages.data.memes

    const randomNumber = Math.floor(Math.random() *
memesArray.length)

    const url = memesArray[randomNumber].url

    setMeme(prevMeme => ({

      ...prevMeme,

      randomImage: url

    })))

  }
```





```
}

function handleChange(event) {

  const {name, value} = event.target

  setMeme(prevMeme => ({

    ...prevMeme,

    [name]: value

  })))

}

return (

  <main>

    <div className="form">

      <input

        type="text"

        placeholder="Top text"

        className="form-input"

        name="topText"


        value={meme.topText}

        onChange={handleChange}

      />
```



```
<input
  type="text"
  placeholder="Bottom text"
  className="form-input"
  name="bottomText"
  value={meme.bottomText}
  onChange={handleChange}
/>

<button
  className="form-button"
  onClick={getMemeImage}
>
  Get a new meme image 
</button>

</div>

<div className="meme">
  <img src={meme.randomImage}
className="meme-image" />

  <h2 className="meme-text
top">{meme.topText}</h2>

  <h2 className="meme-text
bottom">{meme.bottomText}</h2>

</div>

</main>
```



```
)  
}
```

styles.css [completar]

```
.meme {  
  
    position: relative;  
  
    display: flex;  
  
    flex-direction: column;  
  
    align-items: center;  
}  
  
.meme-image {  
  
    max-width: 100%;  
  
    border-radius: 3px;  
}  
  
.meme-text {  
  
    position: absolute;  
  
    width: 80%;  
  
    text-align: center;  
  
    left: 50%;  
  
    transform: translateX(-50%);  
}
```



```
margin: 15px 0;

padding: 0 5px;

font-family: impact, sans-serif;

font-size: 2em;

text-transform: uppercase;

color: white;

letter-spacing: 1px;

text-shadow:

    2px 2px 0 #000,

    -2px -2px 0 #000,

    2px -2px 0 #000,

    -2px 2px 0 #000,

    0 2px 0 #000,

    2px 0 0 #000,

    0 -2px 0 #000,

    -2px 0 0 #000,

    2px 2px 5px #000;
}

.bottom {

    bottom: 0;
}
```



```
.top {  
  
  top: 0;  
  
}
```

## Desafio 12 - Projeto - Obtendo memes de API - useEffect.

- Assim que o componente Meme for carregado pela primeira vez, faça uma chamada de API para [https://api.imgflip.com/get\\_memes](https://api.imgflip.com/get_memes);
- Quando os dados chegarem, salve apenas a parte do array de memes desses dados no estado `allMemes`;
- Pense se há alguma dependência que, se mudada, você gostaria de fazer com que essa função fosse executada novamente;
- **Dica:** por enquanto, não tente usar uma função async/await. Em vez disso, use blocos `.then()` para resolver as promessas de usar `fetch`. Vamos aprender o porquê após este desafio.

Meme.js

```
import React from "react"  
  
import {useState, useEffect} from "react"  
  
export default function Meme() {  
  
  const [meme, setMeme] = useState({  
  
    topText: "",  
  
    bottomText: "",  
  
    randomImage: "http://i.imgflip.com/1bij.jpg"  
  
  })  
  
}
```



```
const [allMemes, setAllMemes] = useState([])

useEffect(() => {

  fetch("https://api.imgflip.com/get_memes")

    .then(res => res.json())

    .then(data => setAllMemes(data.data.memes))

}, [])

function getMemeImage() {

  const randomNumber = Math.floor(Math.random() *
allMemes.length)

  const url = allMemes[randomNumber].url

  setMeme(prevMeme => ({

    ...prevMeme,

    randomImage: url

  })))

}

function handleChange(event) {

  const {name, value} = event.target

  setMeme(prevMeme => ({

    ...prevMeme,
```



```
[name]: value

    )))

}

return (

    <main>

        <div className="form">

            <input

                type="text"

                placeholder="Top text"

                className="form-input"

                name="topText"

                value={meme.topText}

                onChange={handleChange}

            />

            <input

                type="text"

                placeholder="Bottom text"

                className="form-input"

                name="bottomText"

                value={meme.bottomText}

                onChange={handleChange}
```



```
        />

        <button

            className="form-button"

            onClick={getMemeImage}

        >

            Get a new meme image 

        </button>

    </div>

    <div className="meme">

        <img src={meme.randomImage}
className="meme-image" />

        <h2 className="meme-text
top">{meme.topText}</h2>

        <h2 className="meme-text
bottom">{meme.bottomText}</h2>

    </div>

</main>

)

}
```





Com Async/Await, faça a substituição por esse useEffect:

```
useEffect(() => {  
  
  async function getMemes() {  
  
    const res = await  
fetch("https://api.imgflip.com/get_memes")  
  
    const data = await res.json()  
  
    setAllMemes(data.data.memes)  
  
  }  
  
  getMemes()  
  
, [])
```