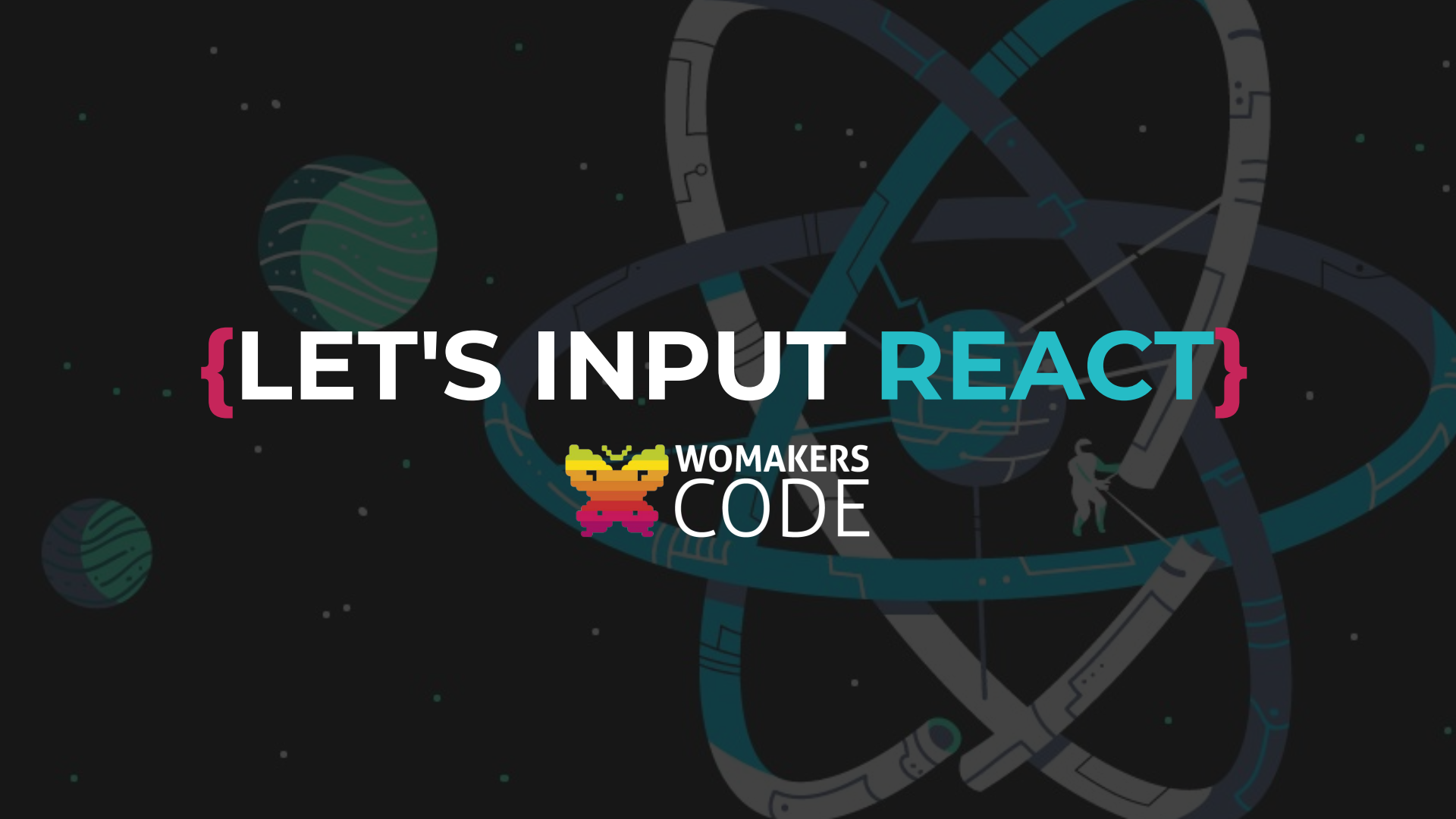

{LET'S INPUT REACT}
WOMAKERS
CODE

{ thinking React }

https://reactjs.org/docs/thinking-in-react.html

{ divida a UI em uma hierarquia de componentes }

{ construa uma versão estática em React }

{ Identificar o mínimo de estado na UI }

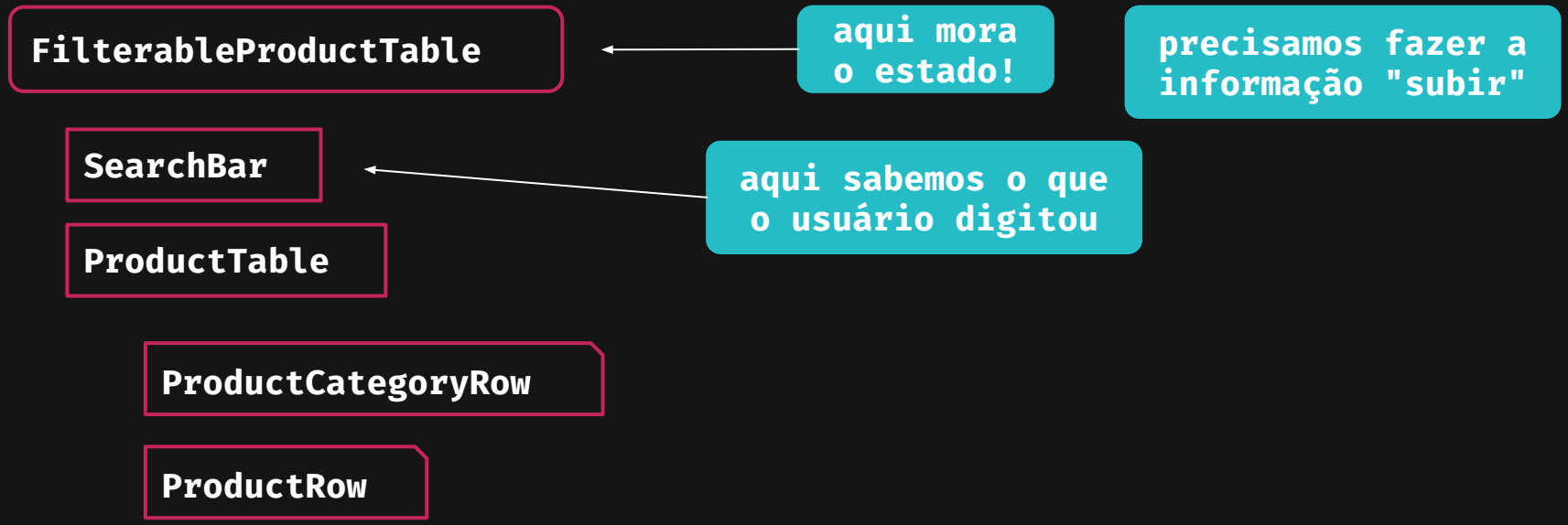
{ Identificar onde o seu estado deve ficar }

{ Inverter o fluxo de dados }

porque o React ignora o texto que estamos digitando na busca?

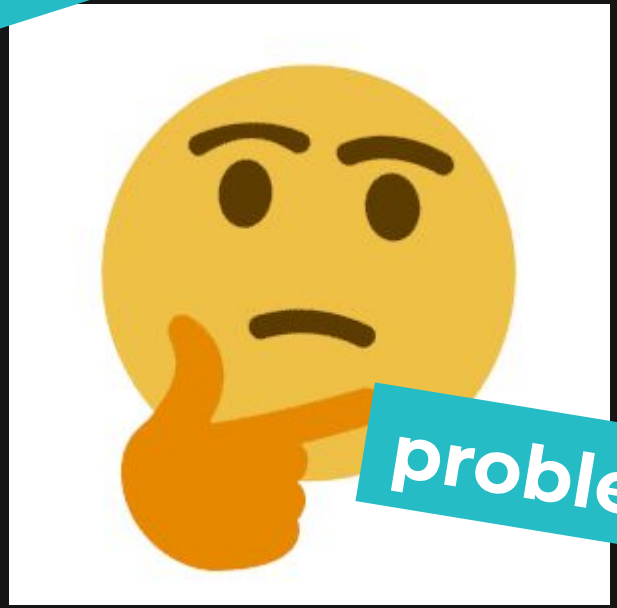
Para fazer com que a informação chegue do input na **FilterableProductTable** e altere seu estado precisamos usar callbacks

lembrando da hierarquia



mas e se o estado mor eto na SearchBar?

não ia precisar de callback nenhum!



problema resolvido

Cadastro de alunas

nome completo

cidade

email

email@email.com

cpf

000.000.000-00

telefone

(xx) xxxxx-xxxx

Increver

além disso temos a regra 4 do thinking React

o estado deve ficar acima dos componentes que usam a informação que ele guarda na hierarquia

{ live coding com callbacks }

e esse bind?

o bind garante que o this dentro da função é o mesmo de fora da função

a arrow function já tem o próprio bind e por isso não precisamos bindar novamente

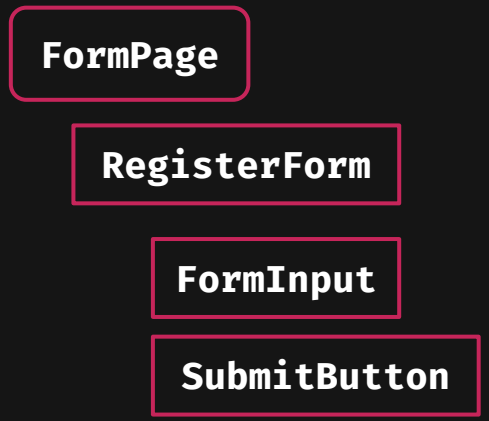
{ formulários }

imagem do formulário pronto

passo 1 - dividir a UI em componentes

15 minutos para pensar em como vocês dividiram a interface!

hierarquia de componentes



passo 2 - construir uma versão estática em React

`demonstração de como usar className`

passo 2 - construir uma versão estática em React

30 minutos para pensar em como vocês dividiram a interface!

Dicas:
Utilizem o input + label do HTML para criar o FormInput
O formulário deve estar envolvido em um <form></form>
Fazer a estilização usando classNames!

passo 3 - identificar o mínimo de estado da UI

10 minutos para pensar em como o state ficaria

Dicas:
Quais variáveis do formulário preciso guardar?

passo 4 - onde o estado deve ficar?

passo 4 - adicione o estado na aplicação

15 minutos para transformar o componente RegisterForm em um ClassComponent e criar o estado inicial dele

5 minutos para passar o valor guardado no estado para os inputs que precisam receber um estado inicial

Dicas:
a propriedade do input que recebe o valor inicial é o value
teste o recebimento de informações preenchendo o estado inicial com suas próprias informações

passo 5 - adicionar o fluxo inverso de dados

Agora vamos entender como funcionam eventos em React!
Depois a gente volta para implementar as funções de
handleChange no **onChange** formulário

{ eventos }

JSX vs HTML

```html
<button onclick="activateLasers()">
  Activate Lasers
</button>
```

```jsx
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

como fica no componente?

```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};

    // This binding is necessary to make `this` work in the callback
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(state => ({
      isToggleOn: !state.isToggleOn
    }));
  }

  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}
```

```
handleClick() {
  this.setState(state => ({
    isToggleOn: !state.isToggleOn
  }));
}
```

```
<button onClick={this.handleClick}>
```

como passar uma função para o componente

```
render() {
  // Wrong: handleClick is called instead of passed as a reference!
  return <button onClick={this.handleClick()}>Click Me</button>
}
```

```
render() {
  // Correct: handleClick is passed as a reference!
  return <button onClick={this.handleClick}>Click Me</button>
}
```

passo 5 - inverter o fluxo de dados

vamos adicionar juntas o fluxo inverso de informações na aplicação

20 minutos para adicionar o fluxo inverso de informações nos outros inputs

Dicas:
para cada cada campo você vai precisar de um `handleChange` (callback) diferente e uma função que será executada no `onChange` do componente

{ validação }

desafio:

não deixar o usuário submeter o formulário se algum campo estiver vazio