Formulários e Componentes Controlados

Formulários em React

Vimos que **Formulários em React** não são tão diferentes de formulários feitos com HTML e Javascript. Na verdade são quase a mesma coisa. Porem vimos como podemos manipular de uma forma muito mais fácil os eventos em React e a partir disso guardar esses dados de *input's* em estados, mas “por que”?

Mantendo esses valores guardados em estados a gente mantem o código (literalmente o código) mais responsável pelas manipulações e envios dos dados, isso faz com que alteremos o layout da nossa aplicação de forma mais livre e personalizada do que usando `required` nas tags HTML's. Por exemplo:

1. Fazer com que estados sejam obrigatórios ou até mesmo com alguma condição:

```
// método onSubmit de um formulario qualquer
handleSubmit() {
  if (!this.state.username.length) return

  if (this.state.password.length < 6) return

  // Faz alguma coisa com username e password
}
```

O código acima vai validar se o `username` tem um tamanho **maior que 0** (pois 0 é um valor considerado falso) e ao mesmo tempo vai ver se `password` tem um tamanho **maior que 6**. E não se limite a só isso, podemos até criar um estado de erro para tratar qual campo esta com erro e a partir disso mostrar uma mensagem personalizada e bonita com CSS e *Renderização condicional.*

Manipulação de Eventos

A primeira aula ja falou bastante sobre isso, mas vamos detalhar um pouco mais sobre o `Event`**.**

Basicamente todo **evento** que criamos recebe a variavel `event` e com ela temos uma gama de dados referente ao evento que foi disparado. Se um **clique** ocorrer o

`target` do evento será o elemento que foi clicado. Se **alterarmos um input** o `target` será o elemento que foi alterado. Acho que não temos problemas com isso, mas tem uma implementação especifica que é problemática com o React, **Radio Buttons**:

```
render() {
  return (
    <div>
      <input type="radio" value="Masculino" name="gender" /> Masculino
      <input type="radio" value="Feminino" name="gender" /> Feminino
      <input type="radio" value="Outro" name="gender" /> Outro
    </div>
  );
}
```

Espero que você esteja se perguntando "aonde o valor esta sendo guardado?” e eu também me perguntei o mesmo quando fiz a primeira vez. Você não vai acreditar, é na **DIV!!!** Se colocarmos um `onChange` nela podemos ver o valor do `target` mudar!

Uma observação importante é que essa implementação só funciona da forma esperada com os *input's* do tipo `radio` por causa de algo chamado de **Event Bubbling**, pois é né achou que nunca mais ouviria isso? Pois eu te digo mais, os três modulos de JS são mais que essenciais então faça uso deles se isso não esta claro ainda.

Componentes Controlados

Essa aula é inteiramente o que chamei de **padrão de entendimento**. Vamos manter em mente que **sempre** iremos criar componentes/elementos controlados, isso traz uma organização muito melhor do nosso código e padrões.

Para seguir esse padrão não podemos nos limitar a *inputs* de texto, sempre que criarmos um **componente** que **recebe e altera um valor** devemos deixar o pai a cargo de fazer os métodos que manipulam esses dados, os componentes filhos devem fazer o trabalho duro de decidir onde, como e quando disparar os métodos que foram recebidos via *Props*.

De maneira resumida, **Componentes Controlados** nada mais, nada menos são **elementos ou componentes** que recebem **AMBOS** o valor do estado e um método que altera esse estado. Claro que acabamos escrevendo um pouco menos de código, mas essa é uma maneira efetiva de criar maneiras que no futuro iremos dar uma boa e fácil manutenção para nossa aplicação.

Validações e Erros

Essa é a melhor parte quando combinamos React, Formulários e Componentes Controlados, com esses três trabalhando juntos podemos criar uma forma de validar dados e erros muito mais fácil do que apenas com Javascript.

Lembre-se que React é **DECLARATIVO** logo você não precisa ficar manipulando os elementos da página, você precisa apenas fazer de **Renderização Condicional** ou até mesmo animações CSS para fazer sua mensagem aparecer. Pode ser apenas um "O nome de usuário é obrigatório” ou uma validação mais trabalhosa como CPF ou email, "O CPF não é válido”.

```
render() {
  return (
    <div>
      <label>CPF:</label>
      <input
        type="text"
        name="cpf"
        value={this.state.cpf.value}
        onChange={this.handleChange} />
      {this.state.cpf.value.length && (
        <p>O CPF é obrigatório</p>
      )}
      {this.state.cpf.error && (
        <p>O CPF digitado não é válido.</p>
      )}
    </div>
  );
}
```

Componentes NÃO Controlados

Esse tópico é importante. Não controlar um componente é uma decisão importante principalmente porque faz com que eles mantenham os dados no DOM, tornando integrações com código React e Javascript comum mais fáceis, mas essa não pode ser a única coisa a se considerar ao adotar essa implementação.

Vamos lembrar um pouco lá do começo de React quando falamos de estados, antes de **saber** alterar estados qual foi a primeira coisa que tentamos? Criar uma variável na classe do nosso componente e alterar ela:

```
render() {
  return (
    <>
```

```
      <label for="cpf">CPF</label>
      <input
        type="text"
        id="cpf"
        onChange={(event) => this._cpf = event.target.value}
      />
    </>
  )
}
```

Basicamente componentes não controlados funcionam assim, mas utilizando algo que chamamos de *ref,* ou melhor uma referencia do nosso elemento do JSX em nossa classe - exatamente como se fossemos obter um elemento do HTML via `getElementBy` ou `querySelector`.

```
handleSubmitClick() {
  const cpf = this._cpf.value;
};

render() {
  return (
    <div>
      <input type="text" ref={(input) => (this._cpf = input)} />
    </div>
  );
}
```

O atributo `ref` do JSX atribui o campo do *input* na variável da classe `this._cpf` isso libera o uso em qualquer método da nossa classe dos dados que foram colocados nesse *input* **mas impossibilita que manipulemos e alteremos esse dado em tempo real para o usuário.** Além disso, vamos nos atentar para alguns pontos importantes tambem:

	Não Controlado	Controlado
Obter os valores em qualquer método	✅	✅
Validação no envio do formulário	✅	✅
Validação instantânea no preenchimento do campo	❌	✅
Desabilitar condicionalmente o botão de enviar	❌	✅
Forçar a entrada de um formato no input	❌	✅
Vários inputs para um único dado	❌	✅
Inputs dinâmicos	❌	✅

https://pt-br.reactjs.org/docs/uncontrolled-components.html

https://goshacmd.com/controlled-vs-uncontrolled-inputs-react/