



Construye formularios usando React

Jefferson A. Peña Torres

jefferson.amado.pena@correounivalle.edu.co

- Eventos
- Componentes controlados
- Componentes NO controlados



Construye formularios usando React

Repasemos, acerca del estado de un componente

1.Eventos

En React o cualquier *framework* basado en componentes utilizan el estado para lograr la interacción con el usuario y cada componente tiene su estado.

2. Componentes controlados

Mientras que las propiedades (*Props*) son inmutables y son pasadas de un componente padre a un componente hijo. El estado puede ser modificado o alterado de acuerdo al comportamiento que se programe.

3. Componentes NO controlados

Un hook es una característica que tiene los componentes para acceder a su estado sin escribir una clase o un método constructor.

Construye formularios usando React

Eventos

1

Los eventos son parte importante de las aplicaciones interactivas, las SPA que usan React incorporan eventos que tiene nombres similares a los propios del DOM.

2

3

`<body>`

- `onload`
- `onunload`

`<form>`

- `onblur`
- `onchange`
- `onfocus`
- `onreset`
- `onselect`
- `onsubmit`

Construye formularios usando React

Eventos

1

Los eventos son parte importante de las aplicaciones interactivas, las SPA que usan React incorporan eventos que tiene nombres similares a los propios del DOM.

2

3

Teclado

- onkeydown
- onkeypress
- onkeyup

Ratón

- onclick
- ondblclick
- onmousedown
- onmousemove
- onmouseout
- onmouseover
- onmouseup



El futuro digital
es de todos

MinTIC



Construye formularios usando React

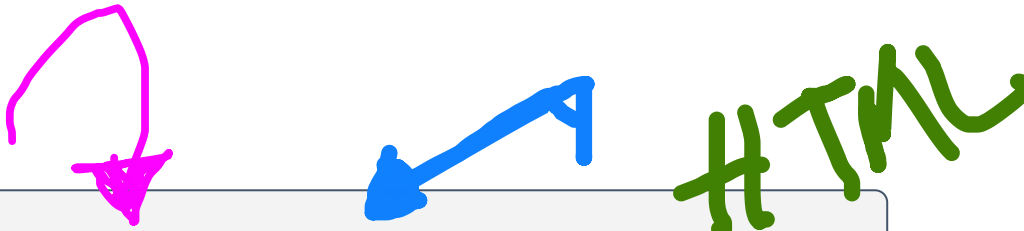
Eventos

1

Los eventos son parte importante de las aplicaciones interactivas, las SPA que usan React incorporan eventos que tiene nombres similares a los propios del DOM.

2

3



```
<button onclick="contar()">  
  Click Aquí  
</button>
```

Click Aquí

HTML



Construye formularios usando React

Eventos


1

Los eventos son parte importante de las aplicaciones interactivas, las SPA que usan React incorporan eventos que tiene nombres similares a los propios del DOM.

2

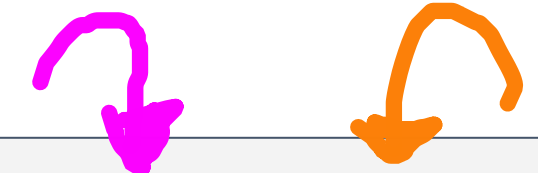
3

- En React los eventos se nombran usando camelCase
- Con JSX se utiliza la referencia a la función en lugar de una cadena



```
<button onclick="contar()">  
  Click Aquí  
</button>
```

Click Aquí



```
<button onClick={contar}>  
  Click Aquí  
</button>
```

Click Aquí



Construye formularios usando React

Formularios: Componentes controlados

1

Los componentes controlados consideran una propiedad llamada value. El valor de las entradas es controlable y es administrado por React. Con esta propiedad los cambios son reflejados en visualización

2

3

```
class Form extends React.Component {
```

```
  constructor(props) {
```

```
    super(props);
```

```
    this.onChange = this.onChange.bind(this);
```

```
    this.state = { name: " " };
```

```
  }
```

```
  onChange(e) {
```

```
    this.setState({ name: e.target.value });
```

```
  }
```

```
  render() {
```

```
    return (
```

```
      <div>
```

```
        <label for='name-input'>Name: </label>
```

```
        <input id='name-input'
```

```
          onChange={this.onChange}
```

```
          value={this.state.name} />
```

```
      </div>
```

```
    )
```

```
  }
```

```
}
```



Construye formularios usando React

Formularios: Componentes controlados

1

Los componentes controlados consideran una propiedad llamada value. El valor de las entradas es controlable y es administrado por React. Con esta propiedad los cambios son reflejados en visualización

2

3

En este ejemplo la propiedad *value* define el valor de la entrada (input) y el controlador de eventos. Cuando sucede un **onChange** se actualiza el estado del componente con al entrada del usuario.

```
class Form extends React.Component {  
  constructor(props) {  
    super(props);  
    this.onChange = this.onChange.bind(this);  
    this.state = { name: " " };  
  }  
  onChange(e) {  
    this.setState({ name: e.target.value });  
  }  
  render() {  
    return (  
      <div>  
        <label for='name-input'>Name: </label>  
        <input id='name-input'  
          onChange={this.onChange}  
          value={this.state.name} />  
      </div>  
    )  
  }  
}
```




Construye formularios usando React

Formularios: Componentes controlados

1

Los componentes controlados consideran una propiedad llamada `value`. El valor de las entradas es controlable y es administrado por React. Con esta propiedad los cambios son reflejados en visualización

2

3

En este ejemplo la propiedad `value` define el valor de la entrada (`input`) y el controlador de eventos. Cuando sucede un `onChange` se actualiza el estado del componente con al entrada del usuario.

Las entradas de los formularios deben definirse como componentes controlados. Esto garantiza que el estado del componente y las entradas del usuario estén sincronizadas

```
class Form extends React.Component {  
  constructor(props) {  
    super(props);  
    this.onChange = this.onChange.bind(this);  
    this.state = { name: " " };  
  }  
  onChange(e) {  
    this.setState({ name: e.target.value });  
  }  
  render() {  
    return (  
      <div>  
        <label for='name-input'>Name: </label>  
        <input id='name-input'  
          onChange={this.onChange}  
          value={this.state.name} />  
      </div>  
    )  
  }  
}
```



Construye formularios usando React

Formularios: Componentes NO controlados

Los componentes controlados **NO** consideran una propiedad llamada value. Para este tipo de componentes es responsabilidad de la aplicación mantener sincronizados el estado del componente y el valor que el usuario a ingresado.

```
class Form extends React.Component {  
  constructor(props) {  
    super(props);  
    this.onChange = this.onChange.bind(this);  
    this.state = { name: 'Jefferson' };  
  }  
  onChange(e) {  
    this.setState({ name: e.target.value });  
  }  
  render() {  
    return (  
      <div>  
        <label for='name-input'>Name: </label>  
        <input id='name-input'  
          onChange={this.onChange}  
          defaultValue={this.state.name} />  
      </div>  
    )  
  }  
}
```



Construye formularios usando React

Formularios: Componentes NO controlados

1

Los componentes controlados **NO** consideran una propiedad llamada value. Para este tipo de componentes es responsabilidad de la aplicación mantener sincronizados el estado del componente y el valor que el usuario a ingresado.

2

3

Al igual que los controlados ante un **onChange** se cambia el valor por defecto en lugar del valor inicial o por defecto. Éste solo tomará el valor del estado al renderizarse, pero luego los cambios en el estado no serán visualizados.

```
class Form extends React.Component {  
  constructor(props) {  
    super(props);  
    this.onChange = this.onChange.bind(this);  
    this.state = { name: 'Jefferson' };  
  }  
  onChange(e) {  
    this.setState({ name: e.target.value });  
  }  
  render() {  
    return (  
      <div>  
        <label for='name-input'>Name: </label>  
        <input id='name-input'  
          onChange={this.onChange}  
          defaultValue={this.state.name} />  
      </div>  
    )  
  }  
}
```

Construye formularios usando React

Resumen

1. Eventos

En React o en cualquier aplicación web es importante tener en **cuenta los eventos**, el DOM posee unos y **de manera similar** este *framework* provee métodos que capturan **eventos sintéticos** de acuerdo a las **especificaciones W3C**.

2. Componentes controlados

3. Componentes NO controlados

En cuanto a la construcción de componentes hay dos opciones **controlados** y no controlados. Los primeros permiten la sincronización entre **la entrada del usuario y el estado del componente**. Mientras que con la segunda opción, componentes no controlados, es responsabilidad **de la aplicación mantener la información**.



www.upb.edu.co/es/mision-tic
#MisiónTICSomosTodos