EVENTOS

EVENTOS

- **Que aprenderemos en este apartado?**

1. **SyntheticEvent**
2. **Eventos**
3. **Clipboard, keyboard, Focus, Form, Mouse**
4. **Touch, Image**

EVENTOS

- En React los eventos de los elementos se manejan de una forma muy similar a como se manejan los eventos de los elementos DOM, pero con algunas diferencias importantes:
  - Los eventos React se nombran usando camelCase, en lugar de minúsculas.
  - Con JSX se pasa una función como controlador de eventos: el objeto, no la invocación.
  - Se debe llamar a e.preventDefault() explícitamente, devolver false no evita el comportamiento predeterminado (propagación de eventos) en React.
- Sintaxis:
  - HTML: <button onclick="sube()">Sube</button>
  - JSX: <button onClick={this.sube}>Sube</button>

EVENTOS

- El controlador de evento se puede implementar como:
  - Método público de la clase:

```
baja(e) {
  e.preventDefault();
  this.setState((prev, props) => {
    return { contador: prev.contador - prev.delta }
  })
}
```

  - Campo público de la clase (experimental):

```
sube = (e) => {
  e.preventDefault();
  this.setState((prev, props) => {
    return { contador: prev.contador + prev.delta }
  })
};
```

  - Función flecha:

```
<button onClick={(e) => this.setState({ contador: 0 })}>Inicia</button>
```

EVENTOS

- Hay que tener cuidado con el significado del this en los callbacks de JSX. En JavaScript, los métodos de clase no están vinculados por defecto. Si no se vincula explícitamente, this estará como undefined cuando se invoque a la función.

```
constructor(props) {
  super(props);
  // ...
  this.baja = this.baja.bind(this);
}
```

- Esto no es un comportamiento específico de React, es cómo funcionan las métodos en JavaScript, si se refiere a un método sin () se debe vincular dicho método.
- No es necesario vincular los campos públicos ni las funciones flecha.
- Las funciones flecha tienen el problema de que se crea una devolución de llamada diferente cada vez que se procesa. En la mayoría de los casos, esto está bien. Sin embargo, si este callback se pasa como una propiedad a los componentes anidados, esos componentes podrían hacer un render adicional. En general, es recomendable enlazar en el constructor o usar la sintaxis de los campos de clase, para evitar este tipo de problema de rendimiento.

EVENTOS

- Por defecto el controlador de eventos recibe como argumento un objeto evento sintético (SyntheticEvent), un contenedor alrededor del evento nativo del navegador. React define estos eventos sintéticos de acuerdo con la especificación W3C, por lo que no hay que preocuparse por la compatibilidad entre navegadores.
- En React, generalmente no se necesita llamar addEventListener para agregar controladores al elemento DOM después de haber sido creado. Basta con proporcionar el controlador cuando el elemento se represente inicialmente.
- Para pasar argumentos a los controladores de eventos:

```
<button onClick={this.deleteRow.bind(this, id)}>Delete Row</button>
<button onClick={(e) => this.deleteRow(id, e)}>Delete Row</button>
```

- Son equivalentes pero el argumento e, que representa el evento React:
  - En el bind, se pasará automáticamente como último argumento.
  - En la función flecha, que ya ha recibido el valor del argumento e, hay que propagarlo explícitamente.

- El objeto SyntheticEvent se reutilizará y todas las propiedades se anularán después de que se haya invocado el controlador de eventos por lo que no se puede acceder al evento de manera asincrónica, hay que cachear su información.

Eventos soportados

- **Clipboard**

Eventos del Portapapeles

Nombres de Eventos:

```
onCopy onCut onPaste
```

Propiedades:

```
DOMDataTransfer clipboardData
```

Eventos soportados

- **keyboard**

Eventos del Teclado

Nombres de Eventos:

```
onKeyDown onKeyPress onKeyUp
```

Propiedades:

```
boolean altKey
number charCode
boolean ctrlKey
boolean getModifierState(key)
string key
number keyCode
string locale
number location
boolean metaKey
boolean repeat
boolean shiftKey
number which
```

Eventos soportados

- **Focus**

Eventos de Enfoque

Nombres de Eventos:

```
onFocus onBlur
```

Estos eventos de enfoque funcionan en todos los elementos en React DOM, no sólo en los elementos de formulario.

Propiedades:

```
DOMEventTarget relatedTarget
```

Eventos soportados

- **Form**

Eventos de Formulario

Nombres de Eventos:

```
onChange onInput onInvalid onReset onSubmit
```

Eventos soportados

- **Mouse**

Eventos del Ratón

Nombres de Eventos:

```
onClick onContextMenu onDoubleClick onDrag onDragEnd onDragEnter onDragExit
onDragLeave onDragOver onDragStart onDrop onMouseDown onMouseEnter onMouseLeave
onMouseMove onMouseOut onMouseOver onMouseUp
```

Eventos soportados

- **Mouse**

Propiedades:

```
boolean altKey
number button
number buttons
number clientX
number clientY
boolean ctrlKey
boolean getModifierState(key)
boolean metaKey
number pageX
number pageY
DOMEventTarget relatedTarget
number screenX
number screenY
boolean shiftKey
```

Eventos soportados

- **Touch**

Eventos Táctiles

Nombres de Eventos:

```
onTouchCancel onTouchEnd onTouchMove onTouchStart
```

Propiedades:

```
boolean altKey
DOMTouchList changedTouches
boolean ctrlKey
boolean getModifierState(key)
boolean metaKey
boolean shiftKey
DOMTouchList targetTouches
DOMTouchList touches
```

Eventos soportados

- **Image**

Eventos de Imagen

Nombres de Eventos:

```
onLoad onError
```

EVENTOS SOPORTADOS

- **Mas eventos ...**

- **https://es.reactjs.org/docs/events.html**