Control de eventos

Tiempo estimado: 10min

La gestión de eventos en React es muy similar a la de HTML, están completamente relacionadas. Pero React lleva a cabo su propio tratamiento sintético. Por lo que hay que atender este tema detenida y separadamente.

La lección comienza introduciendo el concepto de evento y su procesamiento por parte del motor de eventos de la aplicación. A continuación, se presenta los controladores de eventos y cómo registrarlos en la aplicación, así como la cancelación de los controladores predeterminados definidos por el motor de eventos. Después, se muestra varios tipos de eventos: de ratón, de teclado y de foco. No siendo éstos todos los posibles, pero sí los más comunes. Finalmente, describimos la propagación o *bubbling* de eventos y cómo se puede cancelar.

Al finalizar la lección, el estudiante sabrá:

- Qué es un evento.
- Qué es el objeto evento.
- Qué es un controlador de eventos.
- Cómo definir controladores de evento personalizados.
- Cómo cancelar la ejecución de los controladores predefinidos.
- Qué es la propagación de eventos.
- Cómo cancelar la propagación de eventos.

Introducción

Un evento (*event*) representa un acontecimiento, suceso o algo que se ha producido en la aplicación y que deseamos atender explícitamente como, por ejemplo, el clic del usuario en un determinado botón, la entrada en un cuadro de texto, etc. En las aplicaciones React, al igual que en HTML, podemos especificar controladores de evento (*event handlers*), funciones que deben invocarse cuando el evento se produce. Estas funciones controladoras tienen como objeto atender el evento y realizar cualquier operación adicional que sea necesario como, por ejemplo, el envío de los datos al servidor, la aplicación de un filtro en una tabla y cosas por el estilo.

Los eventos se clasifican en dos tipos: nativos y sintéticos.

Un evento nativo (*native event*) es aquel que genera el navegador. Mientras que un evento sintético (*synthetic event*) es una envoltura de un evento nativo generada por el *framework*.

Cada vez que se produce un evento, se genera un objeto evento (*event object*), la instancia particular del evento generado. A través del cual consultar la información del evento y con el que interactuar con el motor de eventos.

React hace un control de eventos particular, tal como hacen otros *frameworks*. Actúa como puente entre el navegador y nuestra aplicación. No hace más que capturar el evento, envolverlo en un objeto con una interfaz común y, a continuación, invocar nuestro controlador de eventos. Este objeto envoltura se conoce como sintético porque reproduce el evento nativo al que representa, pero no es nativo.

En React, los objetos de eventos son sintéticos e instancias de la clase SyntheticEvent. Para acceder al

objeto evento nativo, podemos usar su propiedad nativeEvent.

Proceso de evento

El proceso de evento (*event process*) es la operación mediante la cual se trata o atiende un evento:

1. El evento se produce, por ejemplo, por la interactuación del usuario con la aplicación.
2. El motor de eventos captura el evento y genera el objeto evento.
3. El motor de eventos invoca los controladores de evento definidos por la aplicación. Desde el elemento o componente origen hasta la raíz. Se produce lo que se conoce formalmente como propagación de evento o *event bubbling*.
4. El motor de eventos invoca los controladores predeterminados asociados al evento y definidos por el *framework* y/o el navegador.

En React, el procesamiento de un evento se debe hacer siempre síncronamente. Si no se hace así, algunas propiedades del objeto evento podrían encontrarse a null.

Controladores de Evento

Ya sabemos que un evento no es más que un suceso o algo que se produce en la aplicación como, por ejemplo, el clic del ratón, la pulsación de una tecla, el movimiento del cursor del ratón, la carga del documento, etc.

Las aplicaciones webs pueden capturar eventos, pudiendo así reaccionar ante ellos. ¿Cómo lo hacen? Definiendo lo que se conoce formalmente como controladores de eventos (*event handlers*), funciones JavaScript que debe invocar el motor de eventos cada vez que se produzca el evento. Estos controladores nos dan la oportunidad de reaccionar ante el evento, por ejemplo, para realizar una determinada operación, tarea o trabajo.

En React, los controladores de evento deben presentar la siguiente signatura:

```
controlador(evt ?: SyntheticEvent)
```

Parámetro	Tipo de datos	Descripción
evt	SyntheticEvent	Objeto evento.

He aquí un ejemplo ilustrativo de registro de controlador de eventos:

```
onKeyPress={(evt) => this.handleKeyPress(evt)}
```

Por convenio y buenas prácticas, los controladores de eventos se suelen definir como métodos del componente con un nombre cuyo formato es handle seguido del nombre del evento.

Registro de controladores de evento

Estos controladores se pueden asociar tanto a elementos HTML como a componentes React. Primero, hay que tener claro que los eventos tienen un nombre que los identifica de los demás como, por ejemplo, Click, DoubleClick, MouseDown, KeyPress, etc. Por otro lado, hay que saber cómo indicar un controlador para un determinado evento. Para ello, se utiliza una propiedad del elemento o componente con la siguiente sintaxis:

```
onEvento={controlador}
```

Así pues, para indicar el controlador del evento KeyPress utilizaremos la propiedad onKeyPress.

Los controladores se especifican mediante expresiones que devuelven la función controladora a invocar. Generalmente, se suele hacer de dos maneras:

```
onEvento={this.handleEvento.bind(this)}
onEvento={(evt) => this.handleEvento(evt)}
```

Desde el equipo de React, se tiene predilección por la segunda forma. No recomiendan el uso del método bind(). Aunque ambas formas acabarán con la ejecución del método handleEvento().

Es importante tener claro que la expresión debe devolver una función. Esto no hará lo esperado, porque la expresión no devuelve una función:

```
onClick={alert("has hecho clic!")}
```


Lo anterior generalmente se resuelve como sigue, usando una función anónima:

```
onClick={() => alert("has hecho clic!")}
```

Controladores predeterminados

Los *frameworks* y/o navegadores definen en fábrica sus propios controladores, los cuales contienen las acciones que deben ejecutar una vez ejecutados los controladores de usuario, o sea, los definidos por nosotros mismos. Por ejemplo, cada vez que el usuario hace clic en un botón para enviar un formulario de datos al servidor, el *framework* y/o navegador tiene como acción predeterminada enviar los datos al URL indicado en el formulario.

Es importante diferenciar entre los controladores de evento personalizados, los que definimos nosotros mismos, y los predeterminados, aquellos que realizan la acción final asociada al evento. Los controladores predeterminados se ejecutan siempre después de los controladores personalizados.

En algunas ocasiones, es necesario que la aplicación React cancele las acciones predeterminadas. Los objetos de evento disponen del método preventDefault() con el que podemos indicarle al motor de eventos que no invoque los controladores predeterminados asociados al evento. Su signatura es:

```
preventDefault()
```

Así, por ejemplo, si necesitamos cancelar que se envíe, por la razón que sea, el contenido de un formulario HTML al servidor, bastará con definir el controlador del evento Submit del formulario, en el cual cancelar los controladores predeterminados. Veamos un ejemplo:

```
//en la clase componente cuya representación contiene el formulario
handleSubmit(evt) {
  evt.preventDefault();
}

//en el elemento formulario
onSubmit={(evt) => this.handleSubmit(evt)}
```

¿No es posible hacerlo sin invocar al controlador personalizado handleSubmit()? Sí, veamos cómo:

```
onSubmit={(evt) => evt.preventDefault()}
```

Tipos de evento

El tipo de evento (*event type*) no es más que el nombre del evento como, por ejemplo, click, drag, load, etc. Para conocer el tipo de un determinado evento, basta con consultar la propiedad type del objeto evento.

A continuación, se presenta los eventos más comunes. La lista completa de eventos se puede consultar en facebook.github.io/react/docs/events.html. Tómese su tiempo y échele un vistazo.

Eventos de ratón

Un evento de ratón (*mouse event*) es aquel que se genera debido a algo relacionado con el ratón como, por ejemplo, hacer clic o moverlo. Disponemos de los siguientes controladores de evento:

Evento	Descripción
Click	El usuario ha hecho clic en un elemento de la página.
DoubleClick	El usuario ha hecho doble clic en un elemento de la página.
MouseDown	El usuario ha pulsado el botón del ratón. Acción de pulsar.
MouseUp	El usuario ha pulsado el botón del ratón. Acción de liberar.
MouseOver	El usuario ha introducido el cursor del ratón en un elemento.
MouseMove	El usuario tiene el cursor del ratón dentro de un elemento y lo está moviendo.
MouseOut	El usuario ha sacado el cursor del ratón fuera de un elemento.
DragStart	El usuario ha comenzado a arrastrar un elemento.
Drag	El usuario ha arrastrado un elemento.
DragEnter	El usuario está arrastrando un elemento y ha entrado en otro.

DragLeave	El usuario está arrastrando un elemento, ha entrado en otro y está saliendo de él.
DragOver	El usuario está arrastrando un elemento y se encuentra encima de otro.
Drop	El usuario ha dejado un elemento arrastrado.
DragEnd	El usuario ha finalizado de arrastrar un elemento.
ContextMenu	El usuario ha solicitado el menú contextual.

El objeto evento dispone de las siguientes propiedades: altKey (boolean), button (number), buttons (number), clientX (number), clientY (number), ctrlKey (boolean), metaKey (boolean), pageX (number), pageY (number), relatedTarget (DOMEventTarget), screenX (number), screenY (number) y shiftKey (boolean).

Eventos de teclado

Un evento de teclado (*keyboard event*) es aquel que está relacionado con el teclado del usuario como, por ejemplo, la pulsación de una tecla.

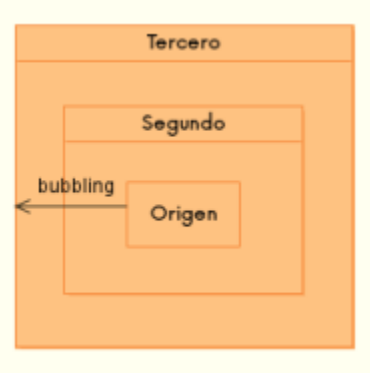
Evento	Descripción
KeyPress	El usuario ha pulsado una tecla.
KeyDown	El usuario ha pulsado una tecla. Una vez está pulsada, pero sin liberar el teclado.
KeyUp	El usuario ha pulsado una tecla. Una vez se ha liberado.

El objeto evento pasado al controlador dispone de las siguientes propiedades: altKey (boolean), charCode (number), ctrlKey (boolean), key (string), keyCode (number), locale (string), location (number), metaKey (boolean), repeat (boolean), shiftKey (boolean) y which (number).

Propagación de evento

Los eventos se generan contra un determinado elemento o componente conocido formalmente como objetivo (*target*). Para conocerlo, basta con consultar la propiedad target del objeto evento.

Cada vez que se genera un evento, el motor no ejecuta únicamente los controladores asociados directamente al componente o elemento objetivo. También invoca los controladores de evento personalizados de los componentes o elementos que contienen al objetivo. Se produce una propagación (*bubbling*), reproducción del evento desde el objetivo hasta el elemento raíz del documento. Como se puede observar, la propagación es de dentro afuera o de abajo arriba, según se quiera ver. La cuestión es que comienza en el elemento o componente donde se produce y se va reproduciendo a lo largo del árbol DOM hasta su raíz. Pasando por todos aquellos componentes o elementos dentro de los cuales se ha definido directa o indirectamente el objetivo del evento.



Si lo deseamos, podemos cancelar esta propagación en cualquier momento. Basta con invocar el método stopPropagation() del objeto evento en cualquiera de los controladores personalizados invocados por el motor de eventos. Su signatura es como sigue:

```
stopPropagation()
```

