Control de eventos (práctica)

Tiempo estimado: 15min

El objeto de esta práctica es afianzar, reforzar y consolidar los conocimientos teóricos presentados en la lección.

Al finalizarla, el estudiante:

- Habrá definido controladores de eventos personalizados.
- Habrá cancelado la propagación de eventos.
- Habrá cancelado los controladores o acciones predeterminadas asociadas a los eventos.

Objetivos

El objetivo de la práctica es mostrar cómo trabajar con eventos en aplicaciones React.

Preparación del entorno

Para comenzar, crearemos un proyecto de aplicación React mediante el generador de Justo:

1. Abrir una consola.
2. Crear el directorio de la práctica e ir a él.
3. Invocar el generador de Justo:

   ```
   > justo -g react
   ```

   Responder a las preguntas realizadas por el generador. A la hora de responder:

   - Responda N cuando le pregunte si la aplicación usa React Router.
4. Instalar las dependencias del proyecto:

   ```
   > npm install
   ```
5. Invocar la tarea build del catálogo del proyecto para construir la aplicación:

   ```
   > justo build
   ```
6. Abrir el archivo dist/index.html con el navegador.

   Debe aparecer el mensaje Hello World!

Definición de controladores de evento

Una vez preparado el entorno, ya podemos comenzar a divertirnos. En primera instancia, vamos a definir dos controladores de eventos para varios elementos HTML.

1. Editar el archivo app/views/App.jsx.
2. Definir su método render() como sigue:

```
render() {
  return (
    <div onClick={() => alert("Div #1")} style={{backgroundColor: "orange", padding:
"5px"}}>
      Div #1
      <div onClick={() => alert("Div #2")} style={{backgroundColor: "maroon", padding:
"5px"}}>
        Div #2
        <div onClick={() => alert("Div #3")} style={{backgroundColor: "blue", padding:
"5px"}}>
          Div #3
        </div>
      </div>
```



```
    </div>
  );
}
```

Observe que se define los controladores sin el objeto evento. Recordemos, es opcional. Se define cuando vamos a usar, dentro del controlador, la información que nos proporciona.

3. Guardar cambios.
4. Reconstruir la aplicación:

```
> justo build
```

5. Abrir la aplicación.

   Debe aparecer algo como:

6. Hacer clic en cualquier punto de la sección Div #3.

   Observe que se ejecutan los tres controladores, de dentro afuera. Esto se debe a la propagación o *bubble*.

7. Hacer clic en la sección Div #2.

   Ahora, no se ejecuta el de Div #3.

8. Hacer clic en la sección Div #1.

   Ahora, sólo se ejecuta el de Div #1.

Cancelación de la propagación

En esta sección, vamos a cancelar la propagación de eventos:

1. Editar el archivo app/views/App.jsx.
2. Cancelar la propagación de eventos en el controlador de la sección Div #2:

```
render() {
  return (
    <div onClick={() => alert("Div #1")} style={{backgroundColor: "orange", padding:
"5px"}}>
      Div #1
      <div onClick={(evt) => { evt.stopPropagation(); alert("Div #2"); }}
style={{backgroundColor: "maroon", padding: "5px"}}>
        Div #2
        <div onClick={() => alert("Div #3")} style={{backgroundColor: "blue", padding:
"5px"}}>
          Div #3
        </div>
      </div>
    </div>
  );
}
```

3. Guardar cambios.
4. Reconstruir y reabrir la aplicación.
5. Hacer clic en la sección Div #3.

   Ahora, debido a la cancelación de la propagación en el controlador de la sección Div #2, no se ejecutará el controlador de Div #1 cuando se genere el evento en las secciones Div #2 y Div #3.

6. Hacer clic en la sección Div #2.
7. Hacer clic en la sección Div #1.

Cancelación de los controladores predeterminados

Recordemos que hay dos tipos de controladores, atendiendo a quién los define, los predeterminados y los personalizados. Los predeterminados los define el *framework* y/o navegador. Son inherentes a ellos. Mientras que los personalizados los definimos nosotros mismos. Primero se ejecutan los personalizados y al final los predeterminados.

1. Editar el archivo app/views/App.jsx.
2. Definir el método render() como sigue:

```
render() {
  return (
    <div>
      Texto: <input type="text" onKeyPress={(evt) => alert(evt.key)} autoFocus />
    </div>
  );
}
```

3. Guardar cambios.
4. Reconstruir y reabrir la aplicación.

   Debe aparecer algo como:

5. Escribir el texto hola y observar la ejecución del controlador del evento KeyPress.
6. Editar el archivo app/views/App.jsx.
7. Cancelar que se ejecute los controladores predeterminados:

```
render() {
  return (
    <div>
      Texto: <input type="text" onKeyPress={(evt) => { evt.preventDefault();
alert(evt.key); }} autoFocus />
    </div>
  );
}
```

8. Guardar cambios.
9. Reconstruir y reabrir la aplicación.
10. Escribir hola y observar que no se muestra los caracteres en el cuadro de texto.

    ¿¡Por qué!? El comportamiento predeterminado de este evento es mostrar el carácter en el cuadro de texto. Al cancelarlo, no lo hará.

