



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

**Centro de
e-Learning**

Desarrollo en React JS

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148

www.sceu.frba.utn.edu.ar/e-learning



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

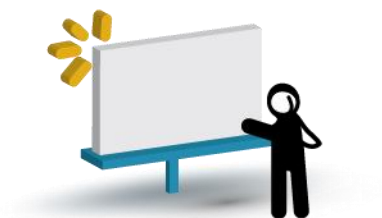
**Centro de
e-Learning**

p. 2

Módulo II

Componentes y Virtual DOM

Unidad 4: Eventos



Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Presentación:

En esta unidad aprenderemos como utilizar los eventos nativos del navegador con react. Veremos qué eventos están disponibles, como declararlos, que variantes tenemos para uso y cerraremos la unidad con un ejemplo.



Objetivos:

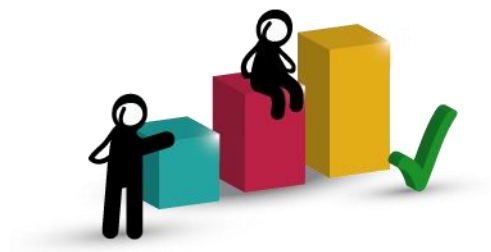
Que los participantes*:

- Comprendan que son los eventos
- Aprendan cómo programar basado en eventos en react
- Realicen un ejemplo con la utilicen de los mismos.



Bloques temáticos*:

- Eventos con react
- Cómo declarar los eventos
- Eventos disponibles en react
- Cómo recibir parámetros en los eventos
- Ejemplo aplicado a nuestro código
- Ejemplo aplicado a formularios
- React hook forms



Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



Tomen nota*

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

Eventos con react

React tiene sus propios eventos, que cuentan con la misma interfaz de los eventos nativos del navegador, con la ventaja que los eventos de React tienen un comportamiento compatible con la mayoría de los navegadores.

Estos eventos reciben una función, o manejadores de eventos. Lo que hacen estas funciones es definir el comportamiento de la aplicación si se corre X o Y evento.

Algo muy común en React, es declarar los manejadores de eventos como funciones dentro de una clase. Cuando hacemos esto, debemos tener cuidado, ya que this no se conecta por defecto a la clase. Para resolver esto, tenemos tres opciones: hacer la conexión en el render(), en el constructor(), o usar arrow functions.



Cómo declarar los eventos

En javascript la declaración de eventos la realizamos de la siguiente manera:

```
<div onclick="handleClick()">click me</div>

function handleClick() {
    alert('clicked');

    return false;
}
```

Como vemos utilizamos el onclick, en el cual hacemos referencia a una función declarada.



En react lo haremos de la siguiente manera:

```
<div onClick={handleClick}>click me</div>

function handleClick(event) {
  alert('clicked');

  event.preventDefault();
  event.stopPropagation();
}
```

Como vemos también utilizamos el evento onClick, pero a diferencia de lo realizado en javascript solo declaramos el nombre de la función.

La función declarada sólo recibe el argumento evento, sobre el cual podremos realizar la acción de stop propagation (detener propagación o comportamiento por defecto).

Consideraciones:

- El nombre del evento tiene que ser camelCase y no minúscula sostenida.
- Al evento se le pasa la función y no una cadena de texto.
- En react si quieres prevenir un comportamiento por defecto o la propagación de un evento debes hacerlo explícitamente llamando los métodos **preventDefault()** y **stopPropagation()** respectivamente.



Eventos sintéticos

En este caso event es un evento sintético de React, en React todos los manejadores de eventos son instancias de **SyntheticEvents**. Los eventos sintéticos son una envoltura de los eventos nativos del navegador, por lo que estos eventos cuentan con la misma interfaz de los eventos nativos, como por ejemplo **preventDefault()** y **stopPropagation()**, con la ventaja de que todos estos eventos funcionan idénticamente en la mayoría de los navegadores.

Eventos disponibles en react

React soporta los siguientes eventos:

- **Mouse:** onClick, onContextMenu, onDoubleClick, onMouseDown, onMouseEnter, onMouseLeave, onMouseMove, onMouseOut, onMouseOver, onMouseUp,
- **Drag & Drop:** onDrag, onDragEnd, onDragEnter, onDragExit, onDragLeave, onDragOver, onDragStart, onDrop.
- **Focos y formularios:** onFocus, onBlur, onChange, onInput, onSubmit.
- **Touches:** onTouchCancel, onTouchEnd, onTouchMove, onTouchStart.
- **Cortar y pegar:** onCopy, onCut, onPaste.
- **Scrolls:** onScroll, onWheel.



Cómo recibir parámetros en los eventos

Si queremos pasar un parámetro a la función llamada en la ejecución de un evento en react debemos hacerlo utilizando la función arrow de es6. Por ejemplo:

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

Aquí vemos que cuando se ejecuta el onClick, se llama a la función arrow que tiene como parámetro a **e**. Luego esta función llama en su ejecución a la función **deleteRow**, la cual debe estar declarada dentro del componente.



Ejemplo aplicado a nuestro código

Aplicaremos la utilización de eventos en nuestro componente **Login**, quedando de la siguiente manera:

```
class Login extends Component {  
  |  
  |  
  | ingresar(e){  
  |   alert("entre");  
  | }  
  | render() {  
  |   |  
  |   | return (  
  |   |   <div>  
  |   |     <form>  
  |   |       <div>  
  |   |         <label>Usuario</label>  
  |   |         <input type="text" placeholder="Introduzca su usuario" />  
  |   |       </div>  
  |   |       <div>  
  |   |         <label>Contraseña</label>  
  |   |         <input type="text" placeholder="Introduzca su contraseña" />  
  |   |       </div>  
  |   |       <button onClick={this.ingresar}>Ingresar</button>  
  |   |     </form>  
  |   |   </div>  
  |   | )  
  |   | }  
  | }  
}
```

Como vemos en el **onClick** debemos utilizar el **this** si el método lo tenemos declarado a nivel clase.



Ahora si declaramos la función ingresar dentro del render, lo hacemos sin él **this**.
Quedando de la siguiente manera:

```
render() {  
  ingresar(e){  
    alert("entre");  
  }  
  return (  
    <div>  
      <form>  
        <div>  
          <label>Usuario</label>  
          <input type="text" placeholder="Introduzca su usuario" />  
        </div>  
        <div>  
          <label>Contraseña</label>  
          <input type="text" placeholder="Introduzca su contraseña" />  
        </div>  
        <button onClick={ingresar}>Ingresar</button>  
      </form>  
    </div>  
  )  
}
```



En el caso de las funciones, el llamado se realiza sin this y la misma debe estar declarada en el componente:

JSX

```
<button onClick={handleClick}>Registrarse</button>
```

Función declarada en el componente

```
const handleClick = (event)=>{  
  console.log("handleSubmit", form)  
  event.preventDefault()  
}
```



Ejemplo aplicado a formularios

En el siguiente ejemplo veremos como aplicar el uso de eventos a un formulario en un componente tipo función.

Vamos a declarar un hook para almacenar los datos introducidos en el formulario:

```
const [form, setForm] = useState({nombre:'', apellido:'', email:'', password:''})
```

Por otro lado vamos a declarar una función que se ejecutara con el evento submit del formulario:

```
const handleSubmit = (event) => {  
  console.log("handleSubmit", form)  
  event.preventDefault()  
}
```

El console.log nos retornara los datos que el usuario introdujo en el formulario.

Vemos el formulario (jsx) desarrollado:

```
return(  
  <div>  
    <form onSubmit={handleSubmit}>  
      <div>  
        <label>Nombre</label>  
        <input type="text" name="nombre" value={form.nombre} onChange={handleChange}></input>  
      </div>  
      <div>  
        <label>Apellido</label>  
        <input type="text" name="apellido" value={form.apellido} onChange={handleChange}></input>  
      </div>  
      <div>  
        <label>Email</label>  
        <input type="email" name="email" value={form.email} onChange={handleChange}></input>  
      </div>  
      <div>  
        <label>Contraseña</label>  
        <input type="password" name="password" value={form.password} onChange={handleChange}></input>  
      </div>  
      <button type="submit">Registrarse</button>  
    </form>  
  </div>  
)
```




En react el flujo de datos es unidireccional, por lo cual debemos hacer una doble relación de los datos. En este caso asociamos el value del input a la propiedad del hook form correspondiente.

Ejemplo el input nombre queda asociado a form.nombre, esto implica que cuando se modifique dicha propiedad en el hook se modificara el value en el input.

Por otro lado capturamos el evento onchange y llamamos a la función handleChange, en la misma realizamos la modificación del hook form de la siguiente manera:

```
const handleChange = (event) => {  
  const name = event.target.name  
  const value = event.target.value  
  console.log("handleChange", name, value)  
  setForm({...form, [name]: value})  
}
```

Para que esto funcione el nombre de cada propiedad del hook form tiene que llamarse igual que el atributo name del elemento, mediante esta función identificamos el elemento que esta siendo modificado y actualizamos el valor del hook.



El componente completo queda de la siguiente manera:

```
import React, { useState } from "react"

function RegistroPage(){

  const [form,setForm] = useState({nombre:'',apellido:'',email:'',password:''})

  const handleSubmit = (event)=>{
    console.log("handleSubmit",form)
    event.preventDefault()
  }

  const handleChange = (event)=>{
    const name = event.target.name
    const value = event.target.value
    console.log("handleChange",name,value)
    setForm({...form,[name]:value})
  }

  return(
    <div>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Nombre</label>
          <input type="text" name="nombre" value={form.nombre} onChange={handleChange}></input>
        </div>
        <div>
          <label>Apellido</label>
          <input type="text" name="apellido" value={form.apellido} onChange={handleChange}></input>
        </div>
        <div>
          <label>Email</label>
          <input type="email" name="email" value={form.email} onChange={handleChange}></input>
        </div>
        <div>
          <label>Contraseña</label>
          <input type="password" name="password" value={form.password} onChange={handleChange}></input>
        </div>
        <button type="submit">Registrarse</button>
      </form>
    </div>
  )
}

export default RegistroPage;
```



React hook forms

Es una biblioteca para react, en la cual nos permite realizar construcción y validaciones de formularios de manera sencilla.

<https://react-hook-form.com/>

En el siguiente video se puede observar como aplicar la misma:

<https://www.youtube.com/watch?v=xjkzYyoz5k8&list=PLhA16ilUESxnCoaLz74beZ0fUDm1gemYk&index=14&t=14s>



Bibliografía utilizada y sugerida

Fedosejev, A. (2015). React.js Essentials (1 ed.). EEUU, Packt.

Amler, . (2016). ReactJS by Example (1 ed.). EEUU, Packt.

Stein, J. (2016). ReactJS Cookbook (1 ed.). EEUU, Packt.

<https://medium.com/@shmesa23/eventos-en-react-b53179ec9683>

<https://www.mmfilesi.com/blog/react-4-eventos-y-estados/>

<https://reactjs.org/docs/handling-events.html>

<http://blog.codeando.org/articulos/manejadores-de-eventos-con-react.html>

<https://react-hook-form.com/>

<https://es.reactjs.org/docs/handling-events.html>



Lo que vimos:

En esta unidad aprendimos que son y cómo trabajar con los eventos en react.



Lo que viene:

En la próxima unidad aprenderemos qué es y cómo funciona firebase.

