



REACT - EVENTOS



Los eventos en React siguen las mismas normas que los `addEventListener` en JavaScript. Dentro del elemento pondremos `on` seguido del nombre del evento que queramos activar, `onClick`, `onChange`, `onInput`...

```
button.addEventListener('click', handleClick)
```

```
const Button = () => {  
  return <button onClick={handleClick}>Activar</button>;  
};  
  
const handleClick = () => {  
  console.log('Click');  
};
```



REACT - EVENTOS



Cuando generamos un evento tenemos bastante información asociada a ese evento.

```
const handleClick = event => {  
  console.log(event);  
};
```

En JavaScript tenemos la información del evento en el DOM

```
const handleClick = event => {  
  console.log(event);  
};
```

En React tenemos un evento sintético, el SyntheticEvent

```
ReactDOM.render(
  <div>
    <button onClick={handleClick}>Click me</button>
  </div>,
  document.getElementById('root')
);

// SyntheticEvent {reactName: 'onClick', _targetInst: null, type: 'click', nativeEvent:
//   PointerEvent, target: button, ...}
//   altKey: false
//   bubbles: true
//   button: 0
//   buttons: 0
//   cancelable: true
//   clientX: 29
//   clientY: 13
//   ctrlKey: false
//   currentTarget: null
//   defaultPrevented: false
//   detail: 1
//   eventPhase: 3
//   getModifierState: f modifierStateGetter(keyArg)
//   isDefaultPrevented: f functionThatReturnsFalse()
//   isPropagationStopped: f functionThatReturnsFalse()
//   isTrusted: true
//   metaKey: false
//   movementX: 0
//   movementY: 0
//   nativeEvent: PointerEvent {isTrusted: true, pointerId: 1, width: 1, height: 1, pressure:
//     pageX: 29
//     pageY: 13
//     relatedTarget: null
//     screenX: 1979
//     screenY: 130
//     shiftKey: false
//     target: button
//     timeStamp: 2929140
//     type: "click"
//   view: Window {window: Window, self: Window, document: document, name: "", location: Loca
//     _reactName: "onClick"
//     _targetInst: null
```



REACT - EVENTOS



Es MUY IMPORTANTE RECORDAR que el evento lo tiene que tener el elemento final, no el componente.

ESTO NO FUNCIONA

```
import Title from './components/title/Title';

const datoImportante = 'Hola';
const App = () => {
  return <Title onClick={() => console.log(datoImportante)} />;
};
```

```
1 const Title = () => {
2   return <h1>Title</h1>;
3 };
4
5 export default Title;
6
```

ESTO SI FUNCIONA

```
import Title from './components/title/Title';

const datoImportante = 'Hola';
const App = () => {
  return <Title elDato={datoImportante} />;
};
```

```
1 const Title = ({ elDato }) => {
2   return <h1 onClick={() => console.log(elDato)}>Title</h1>;
3 };
4
5 export default Title;
6
```



REACT - SYNTHETIC EVENT



El evento sintético que tenemos en React sirve para maximizar la compatibilidad entre navegadores, dentro de éste `syntheticEvent` SÓLO están las propiedades que tienen todos los navegadores y que son las que marca el estándar ECMA.

Si por los requisitos de tu aplicación necesitas acceder al evento real del DOM, dentro de éste `syntheticEvent` existe la propiedad `nativeEvent` que muestra el evento que se disparó en el DOM y que tendrá todas las propiedades propias del navegador donde lo ejecutes.



REACT - EVENT TARGET



Si queremos acceder a las propiedades del elemento que disparó el evento, tendremos toda esa información, el `event.target` SIEMPRE hará referencia al elemento que dispara el evento.

```
<input type='text' onChange={e => console.log(e.target.value)} />
```

h	App.jsx?t=1678738722938:23
ho	App.jsx?t=1678738722938:23
hol	App.jsx?t=1678738722938:23
hola	App.jsx?t=1678738722938:23