



**¡Les damos la
bienvenida!**

¿Comenzamos?

Esta clase va a ser

- grabada

Clase 10. REACT JS

Eventos

Temario

09

Routing y Navegación

- ✓ Organicemos nuestra APP
- ✓ React Router

10

Eventos

- ✓ [Eventos](#)
- ✓ [Componentes basados en eventos](#)

11

Context

- ✓ Contexto
- ✓ Contexto dinámico
- ✓ Nodo proveedor
- ✓ Custom Provider

Objetivos de la clase



Entender el sistema de eventos de react y su implementación



Diseñar componentes orientados a eventos

CLASE N°9

Glosario

NavLink: es un link con un estilo, está siempre detectando la ruta actual, y si coincide con la suya nos activa la clase que le demos para que el user sepa qué ítem de la lista corresponde con la vista actual.

useParams: lo podemos utilizar para leer en js los parámetros de la ruta. En combinación con un useEffect, nos sirve para obtener actualizaciones sobre los parámetros.

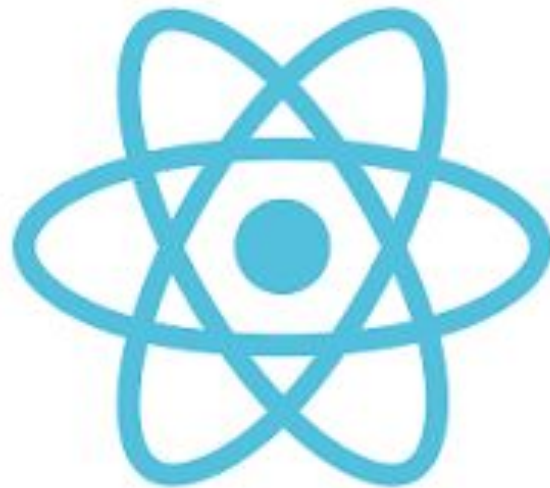
MAPA DE CONCEPTOS



Eventos

Eventos

Si bien existen muy variados tipos de aplicaciones, es raro encontrar alguna que pueda tener sentido sin **eventos**.



¿Qué es un evento en nuestro contexto?

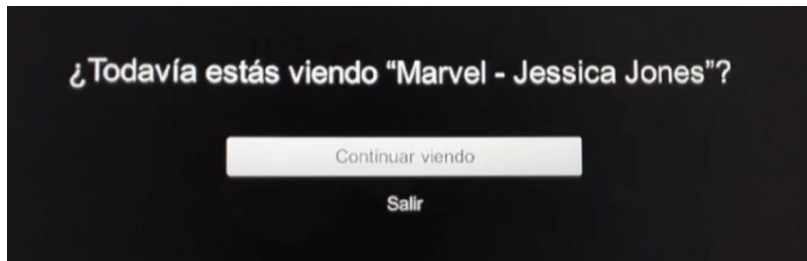
Es un **estímulo** programático, que puede ser provocado de manera **automática**, o ser el resultado de una **interacción** del usuario con la **UI**



Tipos de eventos

Eventos automáticos

Si estamos viendo **Netflix** por **mucho tiempo** sin tocar el control remoto, ocurre un **evento automático por inactividad** que nos pregunta...



Eventos manuales

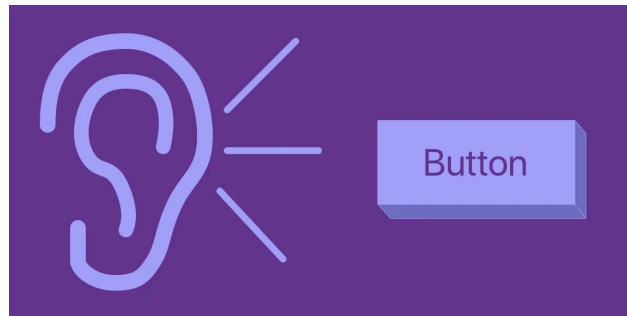
Son todas las **interacciones** del usuario que producen algún tipo de **respuesta o efecto** secundario.

Text Input

Submit



Please fill out this field.



DOM Events

DOM

El **DOM** tiene una serie de eventos [estándar](#), y se dividen en varias categorías:

- ✓ **Dispositivo/acción:** mouse, input, keyboard, wheel, focus, etcétera.
- ✓ **Custom events:** es posible definir eventos propios que disparen la información que queramos.

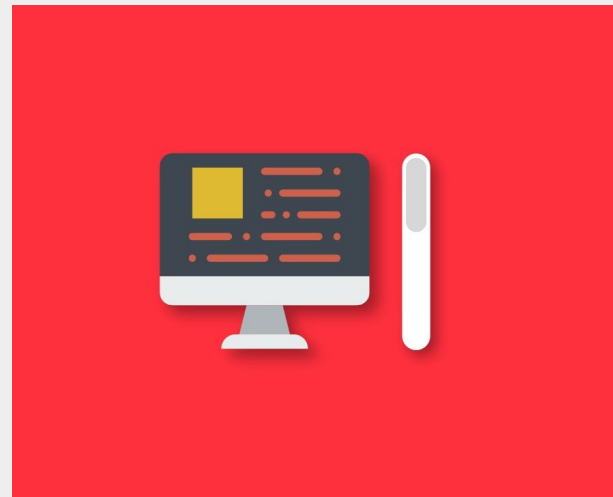


El evento de UI más conocido es el **click**

```
return (  
  <div>e  
    <button onClick={input}>Button</button>  
    <p>Pressed?</p>  
  </div>  
);
```



Aunque probablemente lo utilices casi tanto como el scroll
vía un **wheel event**



Event listener

Event listener

Un **Event Listener** es un patrón de diseño que sirve, como su nombre lo indica, para **escuchar cuando un algo ocurre en algún elemento, librería o API**, y poder realizar una acción en consecuencia.




Tip: ¡hay otros lenguajes que también implementan eventos!

Configurando event listeners

Agregando un event listener

```
window.addEventListener('resize', onResize);
```



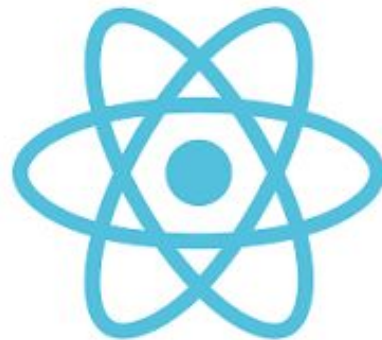
Nombre de evento que quiero escuchar

Referencia de la función a registrar

Nota: ¡guardar **referencia** para poder removerlo después!

Removiendo un event listener

```
return () => {  
  console.log('On dismount');  
  window.removeEventListener(onResize)  
}
```



Nota: invocar el **removeEventListener** en la función de limpieza de nuestros hooks en donde los hayamos registrado.

Removiendo eventos

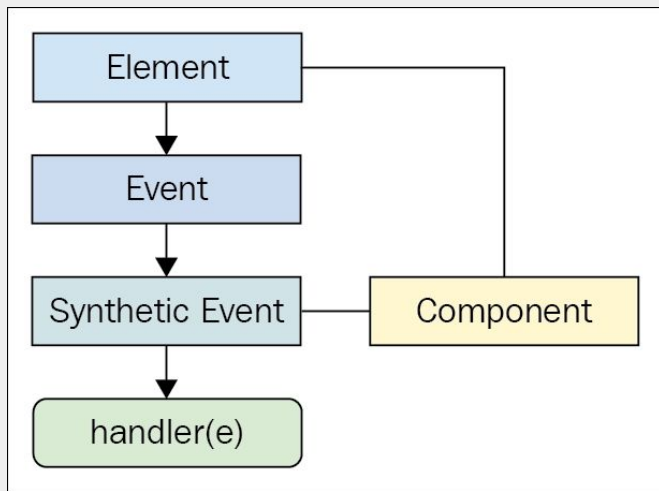
- ✓ Se **des-registran** con el nombre y la referencia a la función con que los registramos (no alcanza únicamente el nombre).
- ✓ Si registras **manualmente** un evento del **DOM** en tu componente de react hazlo dentro de un **effect** y asegúrate de de-registrarlo en la función de limpieza en el **return** del efecto.
- ✓ **Recordemos:** Si dejamos event listeners sin des-registrar corremos riesgos de crear leaks de memoria o registrar un evento más de una vez (se ejecutara una vez por cada register).

React y los eventos

Synthetic events

Los **distintos browsers** suelen tener algunas **variaciones** en el contenido de los eventos.

Esto haría difícil utilizarlos de manera uniforme en cada plataforma. React es consciente de esto, y nos ayuda proveyendo esta **abstracción**.



Synthetic events

- ✓ Sirven para **normalizar/estandarizar** eventos entre browsers.
- ✓ Siempre que registre un evento vía React/Jsx con **onClick**, no obtendré el evento nativo, sino uno sintético.
- ✓ **Se destruyen** al terminar la ejecución de la función vinculada (por performance).
- ✓ Puedo acceder al evento nativo via **evt.nativeEvent**

```
> evt.nativeEvent  
← MouseEvent {isTrusted: true, screenX: 1062, screenY: 256, clientX: 45, clientY: 19, ...}
```



Ejemplo en vivo

¡Vamos al código!

Declarando un evento

Declarando un evento

```
import React from "react";
import "./style.css";

export default function App() {
  function onClick(evt) {
    console.log('Clicked')
    // Al terminar esta función el evt se destruye
  }
  return (
    <div>
      <button onClick={onClick}>Click-me</button>
    </div>
  );
}
```

Click-me

Console



☒ Clear console on reload

Console was cleared

Clicked



Si necesito almacenar el valor del evento puedo guardarlo en un estado.

Declarando un evento

```
export default function App() {  
  function onInput(evt) {  
    evt.preventDefault();  
  }  
  return (  
    <div>  
      <label>Coder name </label>  
      <input onKeyDown={onInput}></input>  
    </div>  
  );  
}
```

Algunos eventos como
onKeyDown son
cancelables, por
ejemplo:

evt.preventDefault()



Declarando un evento

```
export default function App() {  
  function onClick(evt) {  
    evt.stopPropagation();  
  }  
  return (  
    <div>  
      <label>Coder name </label>  
      <input onClick={onClick}></input>  
    </div>  
  );  
}
```

Los eventos por default se ejecutan en el elemento, y en cada uno de sus ancestros. Si esto puede traer algún efecto secundario podemos cancelar la propagación (bubbling):



evt.stopPropagation()



Ejemplo en vivo

¡Vamos al código!



Crear una máscara de Input

Crea un input de texto que no permita el ingreso de vocales

Duración: **15 minutos**



ACTIVIDAD EN CLASE

Crear una máscara de input

Descripción de la actividad.

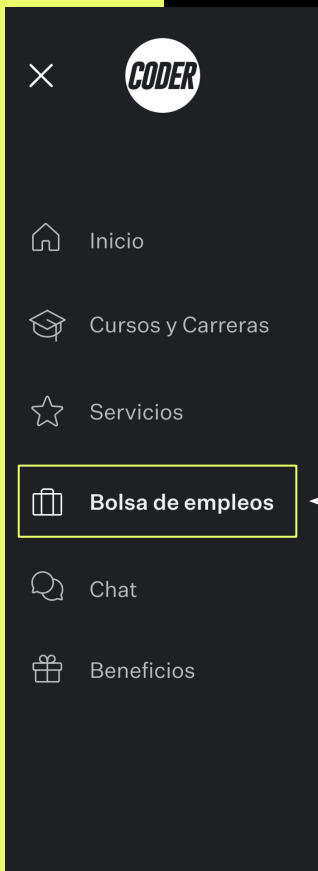
En [stackblitz](#) crea un input de texto que no permita el ingreso de vocales, cancelando su evento `onKeyDown` en los keys adecuados.

Pista: el synthetic event de `keydown` tiene varias propiedades, encuentra cuál te puede dar la información de la tecla ;)



Break

¡10 minutos y volvemos!



Nuevo

¡Lanzamos la Bolsa de Empleos!

Un espacio para seguir **potenciando tu carrera** y que tengas más **oportunidades de inserción laboral**.

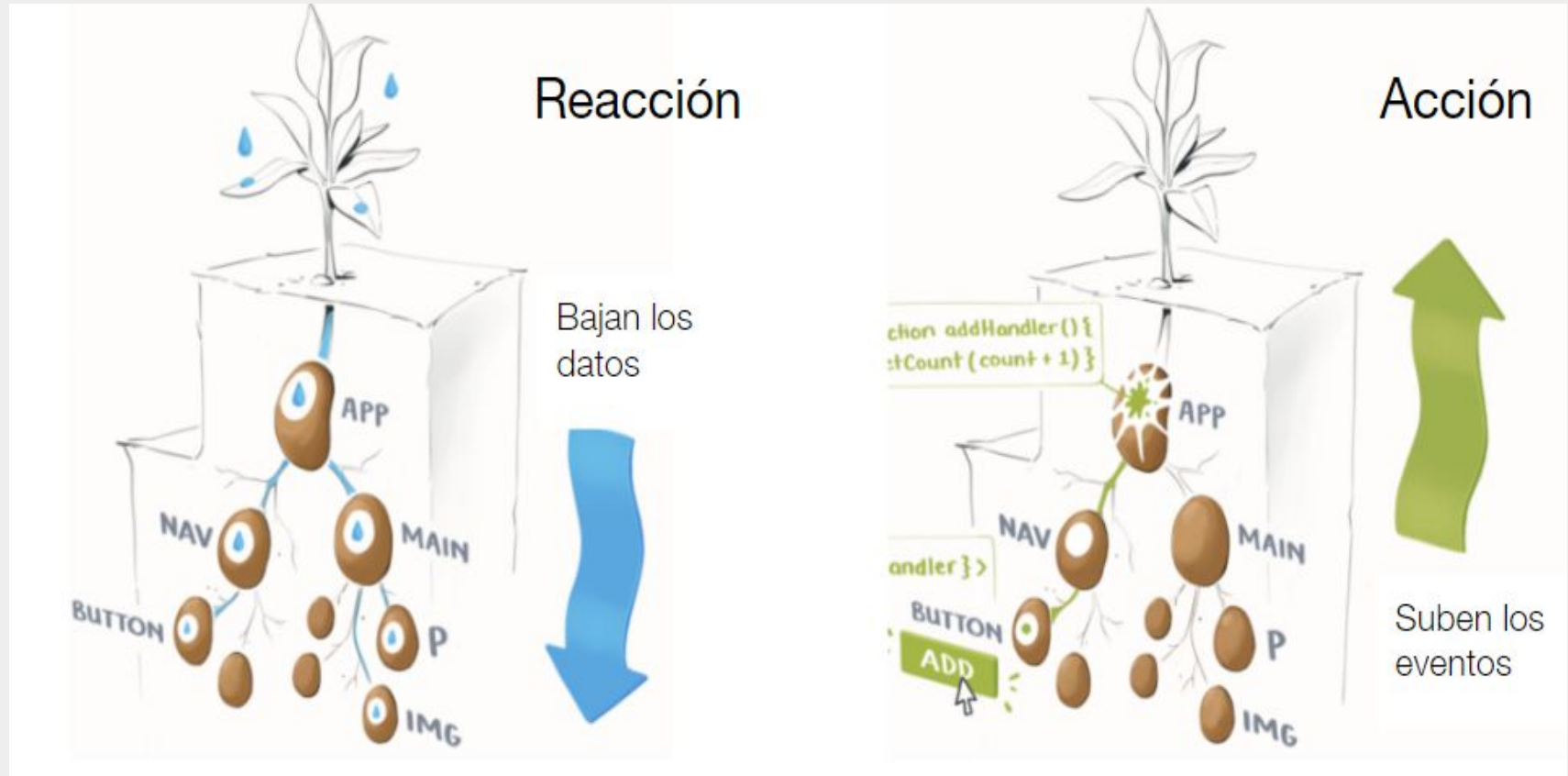
Podrás encontrar la **Bolsa de Empleos** en el menú izquierdo de la plataforma.

Te invitamos a conocerla y ¡postularte a tu futuro trabajo!

Conócela

Componentes basados en eventos

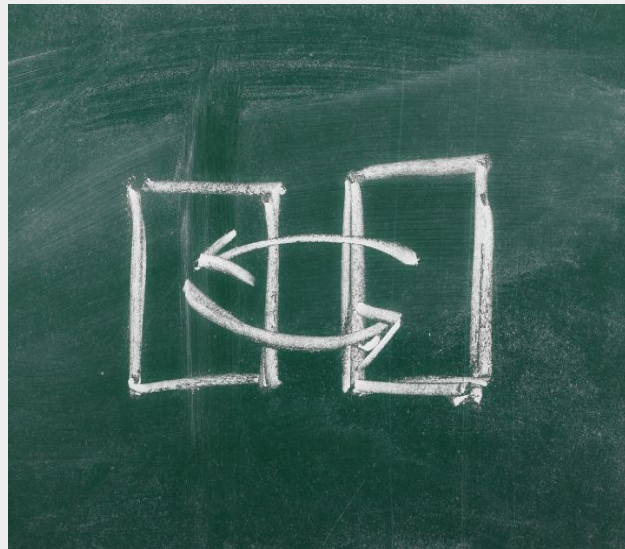
Unidirectional Symmetry



Intercambiabilidad/ Agnostic Behavior

Intercambiabilidad

Implementando componentes de manera eficiente, podremos generar intercambiabilidad, e **intercambiar funcionalidades** sin mucho esfuerzo.



Intercambiabilidad

Podemos generar **variaciones del mismo componente**, con distinto layout y el mismo comportamiento.

```
const InputCount = ({ onConfirm, maxQuantity }) => {  
  };  
const ButtonCount = ({ onConfirm, maxQuantity }) => {  
  };  
  
export default function ItemDetail({ item, inputType = 'input' }) {  
  const Count = inputType === 'button' ?  
    | ButtonCount : InputCount;  
  const itemMax = item.max;  
  const min = item.min;  
  
  function addToCart(quantity) {  
    if(item.inStock) {  
      console.log(`Agregar al cart el item: ${item.id}  
      | con cantidad: ${quantity}`);  
    }  
  }  
  
  return (  
    <div>  
      <label>Item description </label>  
      <Count onConfirm={addToCart} maxQuantity={item.max}></Count>  
    </div>  
  );  
}
```


Abstracción

```
function Select({ options, onSelect, defaultOption }) {  
  return <select  
    onChange={({ evt }) => onSelect(evt.target.value)}>  
    {options.map(o => <option value={o.value}>{o.text} </option>)}  
  </select>  
}  
  
export default function App() {  
  const [option, setOption] = useState(1);  
  const options = [{ value: 1, text: 'Azul' }, { value: 2, text: 'Rojo' }];  
  
  function optionSelected(value) {  
    setOption(value)  
  };  
  
  return <>  
    <Select  
      options={options}  
      onSelect={optionSelected}  
      defaultOption={1}>  
    </Select>  
  
    <p>Seleccionada: {option}</p>  
  </>  
}
```

Azul ▾

Seleccionada: 1

Caso 1

Sirve como estrategia
para **ocultar el
comportamiento
interno** de rendering e
implementación de
change events

Abstracción

```
function Select({ options, onSelect, defaultOption }) {  
  return options.map(o => (<  
    <input onChange={evt => {  
      onSelect(o.value)  
    }}  
    type="radio"  
    name="color"  
    checked={defaultOption === o.value}  
    id={o.value} />  
    <label for={o.value}>{o.text}</label>  
  </>));  
}  
  
export default function App() {  
  const [option, setOption] = useState(1);  
  const options = [{ value: 1, text: 'Azul' }, { value: 2, text: 'Rojo' }];  
  
  function optionSelected(value) {  
    setOption(value)  
  };  
  
  return <  
    <Select  
      options={options}  
      onSelect={optionSelected}  
      defaultOption={option}>  
    </Select>  
  >;  
}
```

☒ Azul ☐ Rojo

Seleccionada: 1

Caso 2

El consumer sigue sin cambiar la firma de consumo:

- **onSelect**
- **defaultOption**

Orientación a eventos



Permite mover la lógica compleja a componentes de menor orden



Si ambos se comportan igual, el parent no lo sabrá aunque sus implementaciones sean distintas.



Permite que el parent se encargue del resultado final sin darle esa responsabilidad a sus children.



Ejemplo en vivo

¡Vamos al código!



#Coderalert

Ingresa al manual de prácticas y realiza la cuarta actividad “Sincronizar Counter”. Ten en cuenta que el desarrollo de la misma será importante para la resolución del Proyecto Final.



Sincronizar Counter

Descripción de la actividad.

- ✓ Importa el ItemCount.js de la primera pre-entrega del PF en el counter ItemDetail.js, y configura el evento de compra, siguiendo los detalles de manual.

Recomendaciones

- ✓ Debes lograr separar la responsabilidad del count, del detalle del ítem, y esperar los eventos de agregado emitidos por el ItemCount
- ✓ Cuando ItemCount emita un evento onAdd almacenarás ese valor en un estado interno del ItemDetail para hacer desaparecer el ItemCount
- ✓ El botón de terminar mi compra debe poder navegar a un componente vacío por el momento en la ruta '/cart'.

¿Preguntas?

Resumen de la clase hoy

- ✓ DOM y Synthetic events.
- ✓ Diseño de eventos en componentes.

Opina y valora
esta clase

Muchas gracias.

#DemocratizandoLaEducación

Javascript: Funciones Útiles

Javascript

- ✓ **FILTER:** Filtrar elementos de un array según cierta condición. Devuelve un nuevo array con los elementos que cumplen con dicha condición.
- ✓ **MAP:** Transformar los elementos de un array. Crea un nuevo array con el mismo número de elementos que el array original, pero aplicando una función de transformación a cada elemento.
- ✓ **FIND:** Buscar el primer elemento en un array que cumpla con una condición específica. Retorna el valor del primer elemento encontrado que satisface la condición, o undefined si ningún elemento lo cumple.

