

ACÀMICA

# Eventos

Practicaremos cómo manejar eventos solo con React, en nuestros componentes.

# Agenda

---

Daily

Referencia vs. invocación

El evento onClick

El evento onChange

El objeto evento



# Daily express *(10 mins)*



Daily



## Sincronizando...

### Toolbox



¿Cómo te ha ido?  
¿Obstáculos?  
¿Cómo seguimos?

### Challenge



¿Cómo te ha ido?  
¿Obstáculos?  
¿Cómo seguimos?

# Referencia vs. Invocación.



En Javascript, este tipo de función se conoce como función d\_\_\_\_\_a.

```
function manejarElClick() {  
  const n1 = 2;  
  const n2 = 30;  
  alert("La sumatoria de estos números es " + (n1 + n2));  
}
```

A su vez, el código anterior representa la d\_\_\_\_\_n de la función.

Mientras que este código:

```
manejarElClick()
```

Corresponde a la i\_\_\_\_\_n de la función.

En Javascript, este tipo de función se conoce como función **declarada**.

```
function manejarElClick() {  
  const n1 = 2;  
  const n2 = 30;  
  alert("La sumatoria de estos números es " + (n1 + n2));  
}
```

A su vez, el código anterior representa la **definición** de la función.

Mientras que este código:

```
manejarElClick()
```

Corresponde a la **invocación** de la función.



# Referencia vs. invocación.

En Javascript, esto es una **invocación**:

```
manejarElClick()
```

En otras palabras, la invocación consiste en llamar a una función para que se ejecute.

Ahora, esto, es una referencia a una función.

```
let una_referencia = manejarElClick;
```

En otras palabras, una referencia consiste en una variable que almacena el código de una función. En JS, todas las variables siempre referencian a un valor. En este caso, la variable "una\_referencia" guarda el código de la función.

# Referencia vs. invocación.

Considerando este código, ¿cuál será el resultado para las variables `opcion1` y `opcion2`?

Es decir, que se mostrará por consola, si hacemos un `console.log` a `"opcion1"` y `"opcion2"`?

```
const suma = (x,y) => {  
  return x + y  
};
```

```
let opcion1 = suma(4,5);  
let opcion2 = suma;
```

# Referencia vs. Invocación.

opcion1

9

opcion2

```
function suma(x,y) {  
  return x + y  
};
```

En Javascript, cuando **referenciamos** una función, estamos pasando solo las instrucciones de ejecución, mientras que una **invocación** ejecuta esas instrucciones.

Cuando manejamos eventos, tenemos que tener cuidado con lo que pasamos. Los eventos deben recibir REFERENCIAS y no INVOCACIONES.

```
<button onClick={manejarElClick} class="btn">CLICKEAME</button>
```

¿Qué pasaría si pasáramos la invocación del event handler directamente en el atributo del evento onClick?

```
<button onClick={manejarElClick()} className="btn">CLICKEAME</button>
```

Se ejecutará de inmediato, lo cual haría un efecto contrario a lo que queremos, que es que la función se ejecute cuando hagamos click en el botón,

Recuerden esto, JS **invocará** funciones donde vea dos paréntesis, siempre.

```
onClick={manejarElClick()}
```

Mientras que las referencias las ejecutará solo cuando un evento se dispare.

```
onClick={manejarElClick}
```

# El evento `onClick` en React



# Programamos

todas las personas, ¡atentas!



# Paso 1: Agregar un botón en <Filtros />

Para ejecutar estos pasos, empezaremos por el ejercicio pasado. [Acá hay un ejemplo terminado del encuentro anterior.](#)

Agrega un botón en el componente <Filtros />. Este botón será el responsable de resetear todos los filtros (esta funcionalidad la codearemos en clases futuras).

También, agrega el evento onClick y pasale una función (aún no creada) con un nombre como "manejarClick".

```
function Filtros() {  
  return (  
    <div className="filtros">  
      <h1>Filtros</h1>  
      <button onClick={manejarClick}>Reset filtros</button>  
    </div>  
  );  
}
```

¿Quién sigue al siguiente paso?



## Paso 2: Construir el manejador del evento

Ahora, tenemos que construir la función que se ejecutará cuando hagamos click sobre este botón.

Construye una función de flecha, cuyo nombre será igual a la **referencia** pasada en el evento onClick. Esta función, por ahora, solo mostrar un alert indicando que el hotel ha sido reservado.

```
function Filtros() {  
  const manejarClick = () => {  
    alert("el hotel ha sido reservado")  
  };  
  return (  
    <div className="filtros">  
      <h1>Filtros</h1>  
      <button onClick={manejarClick}>Reset filtros</button>  
    </div>  
  );  
}
```

Si todo salió bien, al hacer click sobre el botón verás un alert encima de la pantalla.

A close-up photograph of a white ceramic cup filled with a latte. The surface of the milk is decorated with intricate latte art, featuring a central heart shape surrounded by concentric, wavy lines. The cup is placed on a matching white saucer. In the background, a white napkin and a silver spoon are visible, though they are out of focus. The overall lighting is soft and even.

**¡BREAK!**

---

# El evento onChange en React



# Programamos

todas las personas, ¡atentas!



# Paso 1: Agregar un select en <Filtros />

Agrega un select en el componente <Filtros />. Este select será el responsable de mostrar los países disponibles, que actuará de filtro contra los hoteles.

Cada <option> del select tendrá el atributo "value" alineado con un país correspondiente. Además, agrega una <option> con el valor de "todos".

```
return (  
  <div className="filtros">  
    <h1>Filtros</h1>  
    <button onClick={manejarClick}>Reset filtros</button>  
    <select>  
      <option value="todos">Todos</option>  
      <option value="argentina">Argentina</option>  
      <option value="brasil">Brasil</option>  
      <option value="chile">Chile</option>  
      <option value="uruguay">Uruguay</option>  
    </select>  
  </div>  
) ;
```

¿Quién va al siguiente paso?

## Paso 2: Agregar el onChange en el <select>

Agrega el evento onChange en el select de países. Pasa una función de referencia (aún no construida) y dale un nombre como "manejarCambioPais". Por ahora, construye solo la función manejadora, pero no coloques nada adentro de ella aún.

```
function Filtros() {  
  · const manejarClick = () => {  
  ·   alert("el hotel ha sido reservado");  
  · };  
  · const manejarCambioPais = () => {  
  
  · }  
  · return (  
  ·   <div className="filtros">  
  ·     <h1>Filtros</h1>  
  ·     <button onClick={manejarClick}>Reset filtros</button>  
  ·     <select onChange={manejarCambioPais}>  
  ·       <option value="todos">Todos</option>  
  ·       <option value="argentina">Argentina</option>  
  ·       <option value="brasil">Brasil</option>  
  ·       <option value="chile">Chile</option>  
  ·       <option value="uruguay">Uruguay</option>  
  ·     </select>  
  ·   </div>  
  · );  
}
```

# El objeto evento



# El objeto evento

Todos los eventos devuelven un objeto que tiene información sobre el evento disparado. Ese objeto se puede capturar como argumento en el event handler.

```
function Button() {  
  const manejarElClick = (objetoEvento) => {  
    console.log(objetoEvento)  
  }  
  return (  
    <button onClick={manejarElClick}>CLICKEAME</button>  
  )  
}
```



## Paso 3: Construir el manejador

Agrega el parámetro “evento” en el manejador de cambios de país. Haz un `console.log` de ese parámetro y observemos que nos muestra.

```
const manejarCambioPais = (evento) => {  
  console.log(evento)  
};
```

Para activar este evento, deberemos cambiar el valor del select, escogiendo un país, o la opción de “todos”.

React arroja un “evento sintético”, que no es más que una palabra elegante para indicar un evento que podrá ser leído y procesado, por todos los navegadores.

Como podrán ver, este objeto evento tiene un montón de información sobre el evento que acaba de ocurrir, como por ejemplo, la hora en la que ocurrió, y que etiqueta HTML lo disparó.

```
▼ SyntheticBaseEvent {_reactName: "onChange", _targetInst: null, type: "change", nativeEvent: Event,  
  target: select, ...} ⓘ  
  bubbles: true  
  cancelable: false  
  currentTarget: null  
  defaultPrevented: false  
  eventPhase: 3  
  ▶ isDefaultPrevented: f functionThatReturnsFalse()  
  ▶ isPropagationStopped: f functionThatReturnsFalse()  
    isTrusted: true  
  ▶ nativeEvent: Event {isTrusted: true, type: "change", target: select, currentTarget: null, eventP...  
  ▶ target: select  
    timeStamp: 9740.300000011921  
    type: "change"  
    _reactName: "onChange"  
    _targetInst: null  
  ▶ [[Prototype]]: Object
```

¿Quién sigue al siguiente paso?

## Paso 4: Obteniendo info del evento

Lo anterior es importante por que a través del evento `onChange`, podemos saber qué país fue seleccionado, cuando cambiamos el valor del select.

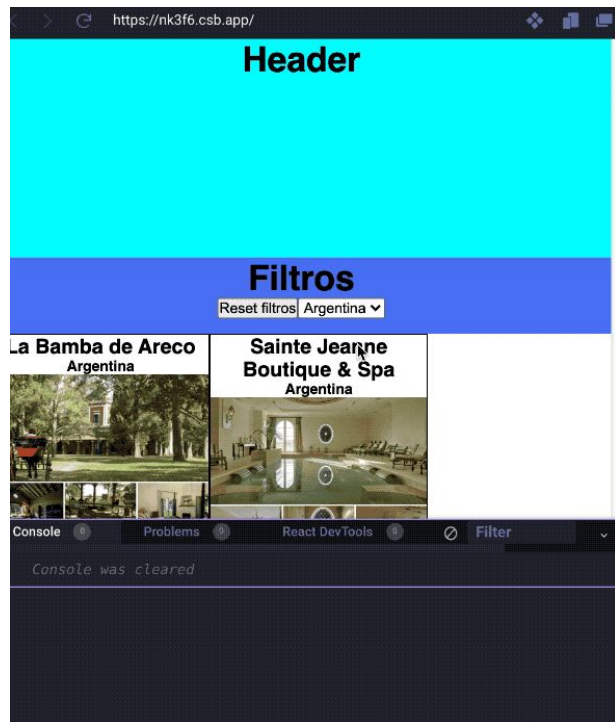
Este dato será importante cuando queramos filtrar los hoteles por algún valor, por ejemplo, el del país.

Modifica el `console.log` para que se muestre la propiedad `target.value` del objeto evento, es ahí donde se almacena el string del atributo “value” de cada `<option>`.

```
const manejarCambioPais = (evento) => {  
  console.log(evento.target.value);  
};
```

Si todo salió bien, al modificar el valor del select, veremos por consola el país seleccionado.

[Te dejamos el sandbox con el ejercicio de hoy finalizado](#), para que puedas comparar el código de lo que hiciste!



# Próximos pasos



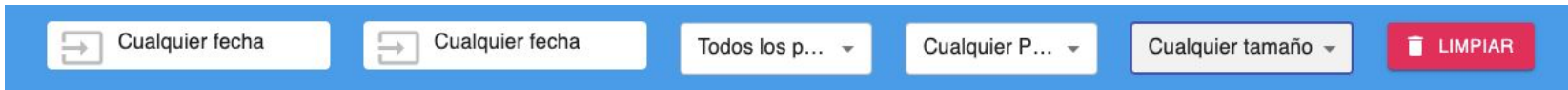
# Agrega otros filtros en <Filtros />

Dentro de <Filtros />, agrega selects para:

- precio, con los siguientes valores:
  - \$ (oferta), con el value "\$"
  - \$\$ (económico), con el value "\$\$"
  - \$\$\$ (moderado), con el value "\$\$\$"
  - \$\$\$\$ (lujoso), con el value "\$\$\$\$"
- tamaño, con los siguientes valores:
  - pequeño, con el value "pequeño"
  - mediano, con el value "mediano"
  - grande, con el value "grande"
- también agrega dos inputs de type "date".

Todos los selects tendrán que tener sus respectivos manejadores de cambio. Los inputs de fecha los trabajaremos más adelante.

Trabajen el CSS para que puedan tener algo como esto:



A horizontal filter bar with a blue background. It contains five white buttons with rounded corners. The first two buttons are labeled 'Cualquier fecha' and each has a small calendar icon to its left. The third button is labeled 'Todos los p...' and has a downward arrow icon. The fourth button is labeled 'Cualquier P...' and has a downward arrow icon. The fifth button is labeled 'Cualquier tamaño' and has a downward arrow icon. To the right of these buttons is a red button with rounded corners labeled 'LIMPIAR' with a trash can icon to its left.

# Para la próxima

---

- 1) Termina el ejercicio de la meeting de hoy.
- 2) Lee la toolbox 29.
- 3) Resuelve el challenge.

¡Te damos un descanso en la generación del sprint project de reservas! Si tienes checkpoints atrasados, aprovecha este tiempo para ponerte al día.

ACÀMICA