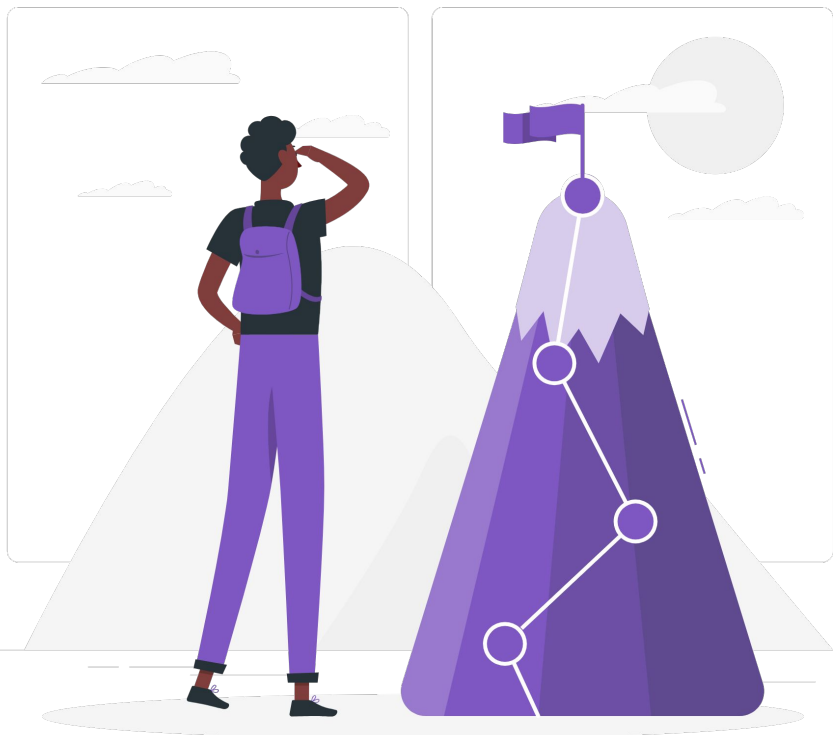




# Formularios en React

**DEV.FX**  
DESARROLLAMOS(PERSONAS);



# Objetivos

- Relevancia de los formularios
- Comprender la diferencia entre formularios tradicionales con HTML (*no controlados*) vs los formularios en React (*controlados*)
- Aprender a evitar que un formulario recargue la página (*prevent.default*)
- Aprender a usar formularios básicos en React.
- Implementar un custom Hook para ayudarnos a controlar los múltiples estados en formularios.

# Relevancia de Formularios

Los formularios son uno de los elementos más usados hoy en día en cualquier sistema, incluyendo por supuesto las aplicaciones web.

**Nos permiten poder introducir información al sistema** para que éste lo almacene o realice acciones, por ejemplo:

- Hacer inicio de sesión
- Registrar una cuenta
- Dar de alta algún ítem a la base de datos
- Visualizar la información capturada y modificarla
- Formularios de contacto, entre otros

## Sign Up

Name \*

Mary

Email \*

mymail@g

Incorrect e-mail. Please try again.

 We don't send spam to our users.

Password \*

☐ Show password

.....

 Password must be at least 6 characters long

\* Required fields

Sign Up

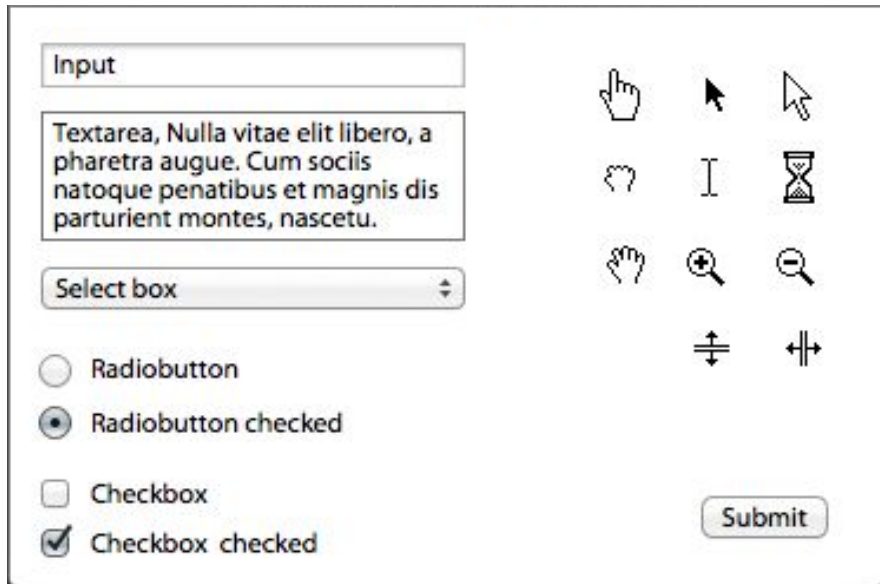
Already have an account? [Log in to your account](#)

# ¿Cómo funciona un Formulario? (1)

Cada elemento de un formulario contiene un valor.

Un valor puede ser escrito (**input**, **textarea**) o puede ser seleccionado (**checkbox**, **select**, **radiobutton**, etc.) por el usuario en el navegador.

Cuando el valor de un elemento del formulario es cambiado (por el hecho de escribir o seleccionar), su valor es actualizado.



The image displays a variety of HTML form controls and their associated mouse cursors. On the left, the controls include: an  field labeled 'Input'; a 

Textarea, Nulla vitae elit libero, a pharetra augue. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur.

; a 

Select box

; a ☐ Radiobutton; a ☒ Radiobutton checked; a ☐ Checkbox; and a ☒ Checkbox checked. On the right, a grid of mouse cursor icons is shown: a hand cursor (for text inputs), an arrow cursor (for buttons and links), a hand cursor (for checkboxes and radio buttons), a magnifying glass with a plus sign (for zooming in), a magnifying glass with a minus sign (for zooming out), and a double-headed arrow cursor (for range inputs). A 'Submit' button is also present at the bottom right.

## ¿Cómo funciona un Formulario? (2)

Podemos obtener el valor de cada elemento del formulario mediante la propiedad **.value**

Así mismo podemos usar **.value** para asignar un nuevo valor a ese elemento.

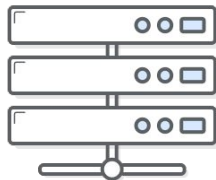
```
> b = document.querySelector('#vue-datatable-34444 > tbody > tr:nth-child(2) > td:nth-child(4) > input')
< <input data-family="precovarejo" value="11.00" type="text" class="animated" data-coord-x="1" data-coord-y="-1" data-last-val="11.00" data-produto="25fdd2d1-619f-42d7-aa52-90df57c67972" maxlength="7">

> b.value
< "11,00"
```



## FORM ELEMENTS

(FRONTEND HTML & CSS)



## FORM PROCESSING

(BACKEND SERVER)



```
<form onSubmit="/myaction.php">
  <input type="email" name="email" id="email">
  <input type="password" name="password" id="password">
  <input type="submit" value="Submit">
</form>
```

# ¿Cómo funciona un Formulario? (3)

Para indicarle a un formulario la acción que debe realizar, se utiliza el atributo **onSubmit** dentro de la etiqueta form de HTML.

Generalmente, un botón es el que desencadena el envío del formulario, este debe tener el atributo **type="submit"**.

Normalmente, esta acción es procesada por otra página, por lo que ocurre una recarga de la página. Sin embargo, en React no debemos permitir que esta recarga ocurra (**preventDefault()**)

# Formularios Controlled vs Uncontrolled

**DEV.FX**  
DESARROLLAMOS(PERSONAS);

dev



En React, existen al menos 2 formas de manejar la información en nuestros componentes de formulario:

1. Componentes no controlados
2. Componentes controlados





```
function App() {  
  function onSubmit() {  
    console.log("Valor de Email: " + window.email.value);  
    console.log("Valor de Password: " + window.password.value);  
  }  
  return (  
    <form onSubmit={onSubmit}>  
      <input type="email" name="email" id="email" required />  
      <input type="password" name="password" id="password" required />  
      <input type="submit" value="Submit" />  
    </form>  
  );  
}
```

# Componentes No Controlados (uncontrolled components)

Un componente no controlado es aquel cuya información del formulario es manejada por el propio DOM.

Es decir, los valores del formulario son tomados directamente del valor almacenado en el DOM.

Se dice qué es “no controlado” por el hecho de que esos componentes del formulario no son controlados por un estado de React, por lo tanto React no tiene control sobre el valor de los mismos.



```
const SimpleForm = () => {
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')

  const handleSubmit = (event) => {
    event.preventDefault()
    const submittedData = JSON.stringify({ email, password })
    console.log(submittedData)
  }

  return (
    <form className='form'>
      <label htmlFor='email'>Email</label>
      <input type='text' name='email'
        onChange={(event)=> setEmail(event.target.value)}
      />

      <label htmlFor='password'>Password</label>
      <input type='password' name='password'
        onChange={(event)=> setPassword(event.target.value)}
      />

      <button onClick={handleSubmit}> Login </button>
    </form>
  )
}
```

# Componentes Controlados (controlled components)

Un componente controlado en React es aquel cuya información (data) es manejada por un estado de React (state).

La primera consiste en usar un estado en el componente que maneja la información del formulario. Esto se le conoce como controlled component.

# Diferencias entre ambos

Algunas diferencias clave entre los componentes no controlados y los controlados son los siguientes:

1. Los componentes controlados son predecibles, por qué el estado de los elementos del formulario son manejados por el componente.
2. Los componentes no controlados no son predecibles, debido a que durante su ciclo de vida, los elementos pueden perder su referencia o ser cambiados o modificados por otras fuentes/entes.
3. En React, los componentes controlados posibilitan un mejor control de validaciones del formulario. No importa si algo afecta a los elementos del formulario, los valores siempre estarán seguros en un estado local, y ahí podremos realizar las validaciones.
4. En componentes controlados, se tiene el control de los valores del formulario, por lo que podemos decidir que información si puede o no puede ser capturada.

# ¡ Vamos al Código !

