Instalación de React:

```
PS C:\Users\delmi\desktop> npx create-react-app readme
```

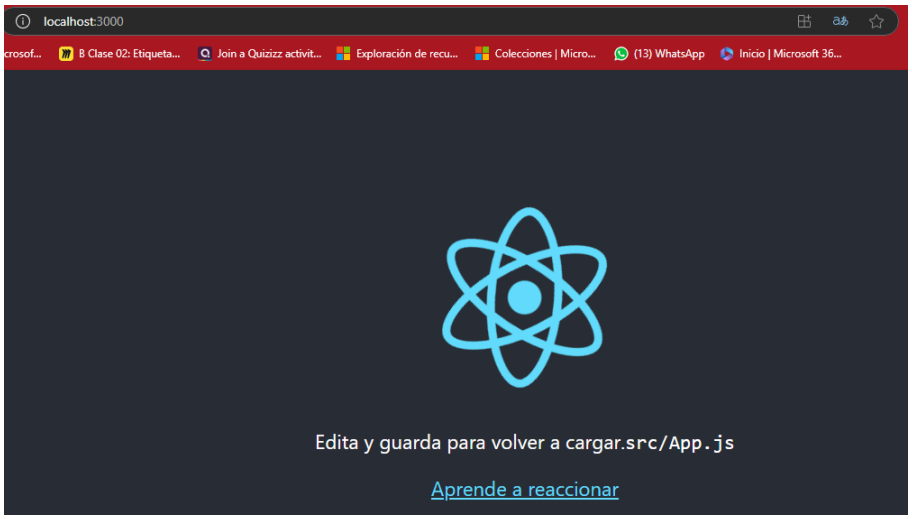
Una vez creado, navegamos en el directorio del proyecto

```
PS C:\Users\delmi\desktop> cd readme
```

Y ejecutamos con el comando start

```
PS C:\Users\delmi\desktop\readme> npm start
```

Esto abrirá nuestro navegador predeterminado y cargará la app



Creación de Componentes:

En tu carpeta src, abre el archivo App.js en cualquier editor de texto.

Edita el contenido para que se vea algo así:

```
JS App.js
C: > Users > delmi > Desktop > readme > src > JS App.js > App
1   import React from 'react';
2
3   function Header() {
4     return <h1>Hola, Bienvenido a mi App</h1>;
5   }
6
7   function App() {
8     return (
9       <div>
10        <Header />
11        <p>Este es mi primer componente en React</p>
12      </div>
13    );
14  }
15
16  export default App;
17
```

Uso de JSX

JSX es la sintaxis que parece HTML pero funciona dentro de JavaScript. En este ejemplo, la función App está retornando un div con HTML integrado, pero realmente es JSX. La diferencia clave entre JSX y HTML es que JSX permite incluir lógica de JavaScript directamente dentro del marcado.

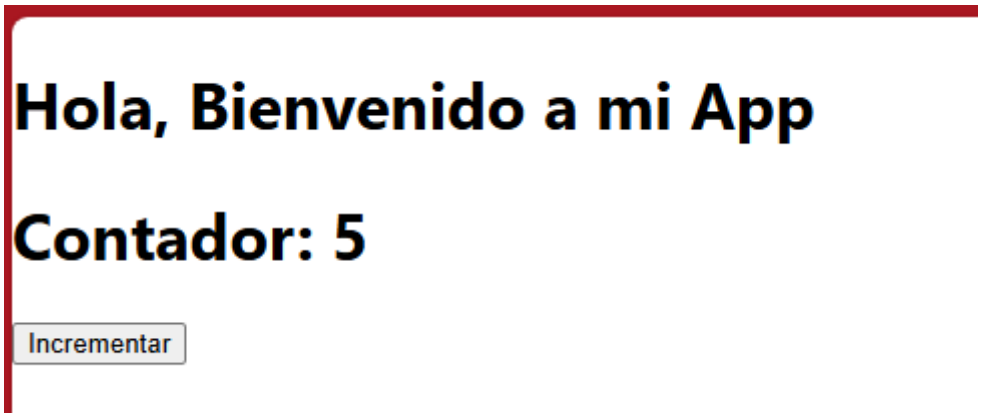
Manejo de Estado

Para agregar interactividad (como un contador), usa useState:

Edita tu App.js para incluir un contador simple:

```jsx
function App() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <Header />
      <h1>Contador: {count}</h1>
      <button onClick={() => setCount(count + 1)}>Incrementar</button>
    </div>
  );
}
```

Se vería asi:



Props en React:

1. Crea un segundo componente: Para usar props, primero necesitarás tener dos componentes: un componente "padre" y un componente "hijo". El componente padre será el que pasa los props al componente hijo.

C: > Users > delmi > Desktop > readme > src > JS Greeting.js.js > ...

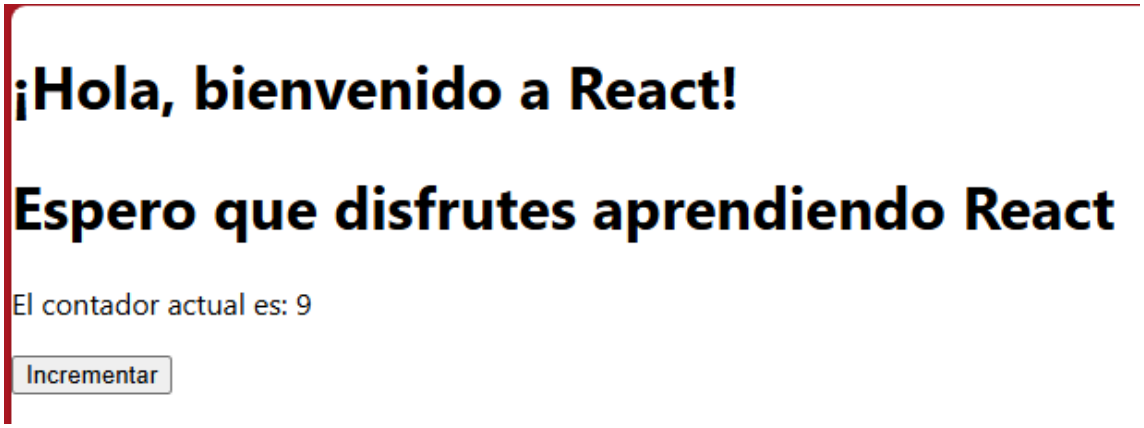
```jsx
import React from 'react';

// Componente que recibe props
function Greeting(props) {
  return <h1>{props.message}</h1>;
}

export default Greeting;

```

Modificar el componente App.js

Users > delmi > Desktop > readme > src > JS App.js > default

```jsx
import React, { useState } from 'react';
import Greeting from './Greeting'; // Importar el componente hijo

// Componente para mostrar el contador
function DisplayCounter(props) {
  return <p>El contador actual es: {props.count}</p>;
}

function App() {
  // Definir el estado del contador
  const [count, setCount] = useState(0);

  // Función para manejar el evento de clic
  const handleButtonClick = () => {
    setCount(count + 1);  // Incrementa el contador en 1 cada vez que se hace clic
  };

  // Definir un mensaje de saludo
  const greetingMessage = "¡Hola, bienvenido a React!";

  return (
    <div>
      {/* Usar el componente Greeting y pasarle el prop 'message' */}
      <Greeting message={greetingMessage} />
      <Greeting message="Espero que disfrutes aprendiendo React" />

      {/* Componente DisplayCounter para mostrar el contador */}
      <DisplayCounter count={count} />
```

```
        {/* Botón para incrementar el contador */}
        <button onClick={handleButtonClick}>Incrementar</button>
      </div>
    );
  }

  export default App;
```



Flujo de datos en React

En React, el flujo de datos es unidireccional, lo que significa que los datos fluyen en una sola dirección: del componente padre hacia el componente hijo. Los componentes padres pasan información a sus hijos a través de props, pero los hijos no pueden modificar esos datos directamente; en lugar de eso, los hijos pueden desencadenar eventos que luego afectan al estado del padre.

Vamos a agregar un componente adicional llamado ResetButton que le permitirá al usuario restablecer el contador a cero. Esto te permitirá ver cómo el flujo de datos unidireccional se implementa en React.

> Users > delmi > Desktop > readme > src > JS ResetButton.js > ...

```
1   import React from 'react';
2
3   // Componente que recibe una función de reset como prop
4   function ResetButton(props) {
5     return <button onClick={props.onReset}>Resetear Contador</button>;
6   }
7
8   export default ResetButton;
9
10
11
```

Manejo de Eventos:

Añadan un botón u otro elemento interactivo que capture eventos de usuario (por ejemplo, un **onClick**) y que modifique el estado del componente.

```
      {/* Campo de entrada y botón para establecer un valor personalizado */}
      <input
        type="number"
        value={inputValue}
        onChange={handleInputChange}
        placeholder="Introduce un valor"
      />
      <button onClick={handleSetCount}>Establecer Contador</button>
    </div>
  );
}
```

Añadimos un poco de estilo

C: > Users > delmi > Desktop > readme > src > styl.css > ...

```
1
2   div {
3       text-align: center;
4       padding: 20px;
5       font-family: Arial, sans-serif;
6     }
7
8     input {
9       padding: 10px;
10      margin-right: 10px;
11      border-radius: 5px;
12      border: 1px solid #ccc;
13      font-size: 16px;
14      width: 200px;
15    }
16
17    button {
18      padding: 10px 20px;
19      background-color: #e6328c;
20      color: white;
21      border: none;
22      border-radius: 5px;
23      cursor: pointer;
24      font-size: 16px;
25    }
26
27    button:hover {
28      background-color: #45a049;
29    }
```

Resultado final:


localhost:3000
Cuenta de Microsof...
B Clase 02: Etiqueta...
Join a Quizizz activit...
Exploración de recu...
Colecciones | Micro...
(13) WhatsApp
Inicio | Microsoft 36...
¡Hola, bienvenido a React!
Espero que disfrutes aprendiendo React
El contador es: 10
Incrementar
Resetear Contador
10
Establecer Contador