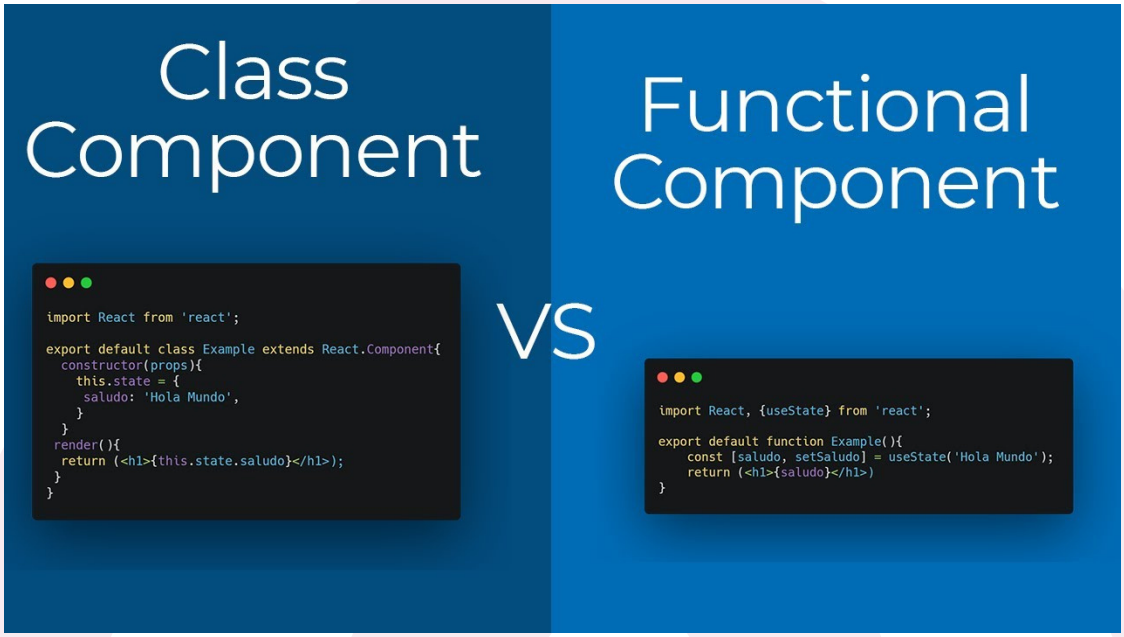
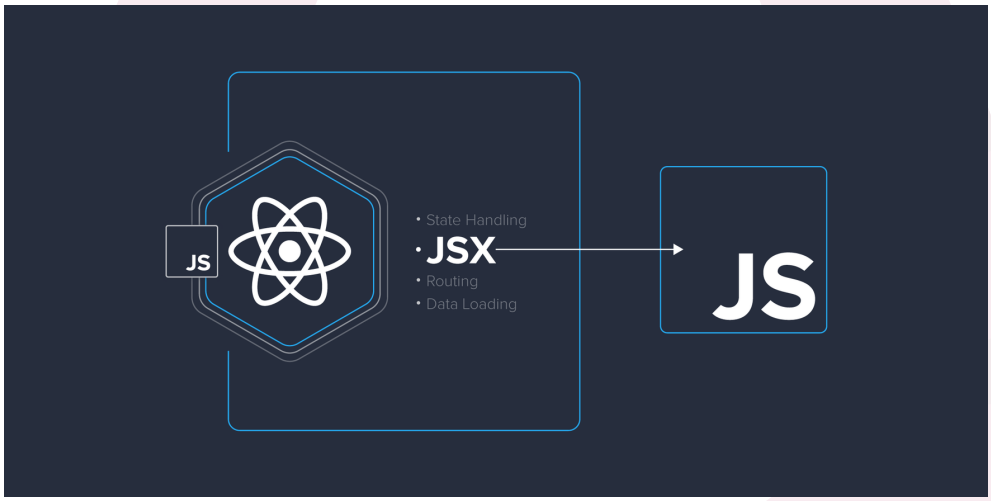
# React con Funciones o con clases

**A partir de React v16 con la implementación de los hooks, los componentes de tipo clase quedaron anticuados.**




# JSX

**JSX (JavaScript XML) es una extensión de JavaScript que se utiliza comúnmente en el desarrollo de aplicaciones web con React, una biblioteca de JavaScript para la creación de interfaces de usuario. Aquí tienes algunas de las reglas fundamentales de JSX:**




- Sintaxis XML-Like: En JSX, se utiliza una sintaxis similar a XML para definir elementos y componentes de interfaz de usuario. Los elementos suelen estar envueltos en etiquetas, y las etiquetas deben cerrarse correctamente. Por ejemplo:

```
<div>
 <p>Este es un párrafo.</p>
</div>
```

- Expresiones en Llaves: Puedes incluir expresiones JavaScript dentro de llaves {} dentro de elementos JSX. Esto te permite insertar dinámicamente valores, variables y expresiones en tu JSX. Por ejemplo:

```
<p>El resultado de 2 + 2 es {2 + 2}.</p>
```

- Componentes: Puedes definir tus propios componentes personalizados en JSX. Los componentes son funciones o clases que devuelven elementos JSX. Por ejemplo:

```
function MiComponente() {
  return <p>Este es un componente personalizado.</p>;
}
```

- Atributos de HTML: Puedes utilizar atributos HTML en elementos JSX, como class, id, href, src, etc. Sin embargo, debes usar className en lugar de class para la clase CSS, debido a conflictos de nombres en JavaScript. Por ejemplo:

```
<a href="https://www.ejemplo.com" className="enlace">Enlace de ejemplo</a>
```

- Eventos: Puedes adjuntar controladores de eventos a elementos JSX utilizando atributos especiales que comienzan con "on", como onClick, onChange, onSubmit, etc. Por ejemplo:

```
<button onClick={handleClick}>Haz clic</button>
```

- Renderizado de Listas: Puedes utilizar un mapeo para renderizar listas de elementos JSX. Por ejemplo:

```
const nombres = ["Alice", "Bob", "Charlie"];
const listaNombres = nombres.map((nombre, index) => <li key={index}>{nombre}</li>);
```

- Uso de Componentes de React: Puedes utilizar componentes de React dentro de otros componentes de React para construir jerarquías de componentes. Por ejemplo:

```
function App() {
  return (
    <div><Header /><MainContent /><Footer /></div>
  );
}
```

**Estas son algunas de las reglas básicas de JSX. JSX es una parte fundamental de React y se utiliza para crear interfaces de usuario declarativas y componentizadas en aplicaciones web.**

# Props

**Una definicion Basica de Props en React:** son como pequeñas piezas de información que puedes pasar a tus componentes. Imagina que estás ensamblando un rompecabezas, y cada pieza del rompecabezas es un componente. Las "props" son como las instrucciones que le das a cada pieza para que encajen correctamente en el rompecabezas. Puedes usar "props" para enviar datos o configuraciones desde un componente padre a un componente hijo. Esto permite que los componentes se comuniquen y compartan información entre sí, lo que hace que tu aplicación sea más dinámica y reutilizable. Las "props" son una forma de personalizar y adaptar el comportamiento y la apariencia de tus componentes en función de lo que necesitas en cada momento.

**Una definicion Tecnica de Props en React:** Son argumentos que se pasan a componentes de React para configurar su comportamiento y contenido. Son objetos JavaScript que contienen datos y se utilizan para la comunicación entre componentes, permitiendo que la información fluya de un componente padre a un componente hijo.