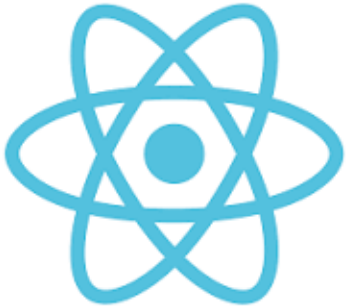
<codoa codo/>

Agencia de Aprendizaje a lo largo de la vida

# ESPECIALIZACIÓN REACT



BA

<codoacodo/>


# **Clase 11.** REACT JS

## *Renderizado Condicional*

BA

# RENDERIZADO CONDICIONAL

El renderizado condicional es la forma de mostrar un componente u otro dependiendo de una condición.

Para hacer renderizado condicional en React usamos el operador ternario:


En este caso, si la prop text existe, se renderiza el botón. Si no existe, no se renderiza nada.

En este caso, si la prop text existe, se renderiza el botón. Si no existe, no se renderiza nada.

Es común encontrar implementaciones del renderizado condicional con el operador &&, del tipo:

```
function List({ listArray }) {
    return listArray?.length && listArray.map(item=>item)
}
```

Parece que tiene sentido... si el length es positivo (mayor a cero) pintamos el map. !Pues no!
❌ Cuidado, si tiene length de cero ya que se pintará en el navegador un 0.

Es preferible utilizar el operador ternario. Kent C. Dodds tiene un artículo interesante hablando del tema. Use ternaries rather than && in JSX

# ESTILOS

Para aplicar clases CSS a un componente en React usamos la prop className:

```
function Button({ text }) {
    return (
        <button className="button">
            {text}
        </button>
    )
}
```

La razón por la que se llama className es porque class es una palabra reservada en JavaScript. Por eso, en JSX, tenemos que usar className para aplicar clases CSS.

Para aplicar estilos CSS en línea a un componente en React usamos la prop style. La diferencia de cómo lo haríamos con HTML, es que en React los estilos se pasan como un objeto y no como una cadena de texto (esto puede verse más claro con los dobles corchetes, los primeros para indicar que es una expresión JavaScript, y los segundos para crear el objeto):

```
function Button({ text }) {
    return (
        <button style={{ color: 'red', borderRadius: '2px' }}>
            {text}
        </button>
    )
}
```

Fíjate que, además, los nombres de las propiedades CSS están en camelCase

Podes aplicar estilos de forma condicional a un componente en React usando la prop style y un operador ternario:

```
function Button({ text, primary }) {
    return (
        <button style={{ color: primary ? 'red' : 'blue' }}>
            {text}
        </button>
    )
}
```

En el caso anterior, si la prop primary es true, el botón tendrá el color rojo. Si no, tendrá el color azul.

También puedes seguir la misma mecánica usando clases. En este caso, usamos el operador ternario para decidir si añadir o no la clase:

```
function Button({ text, primary }) {
    return (
        <button className={primary ? 'button-primary' : ''}>
            {text}
        </button>
    )
}
```