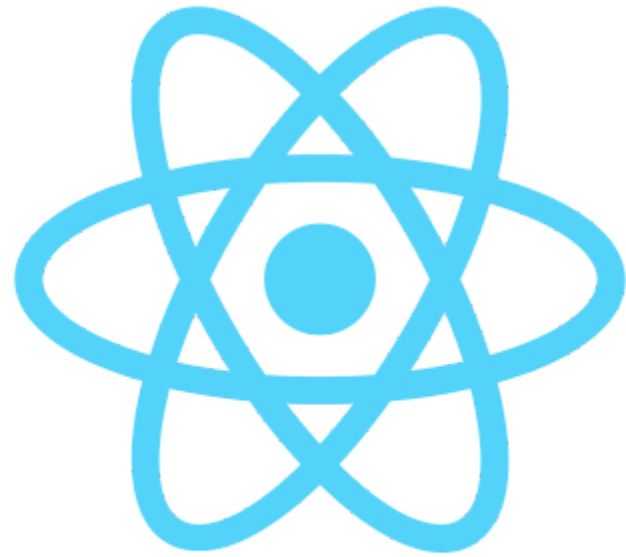


HENRY

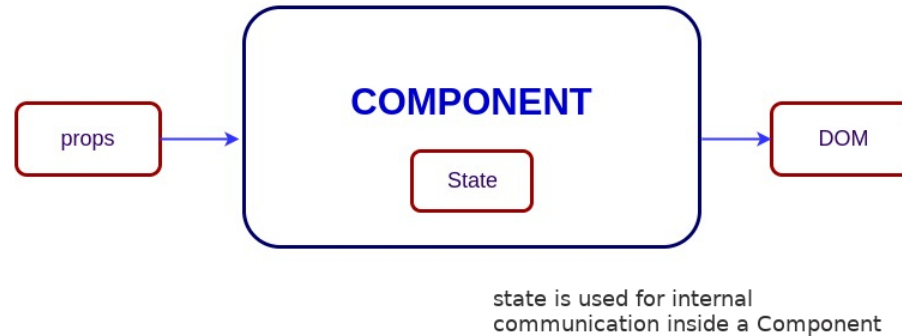
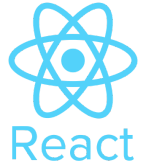
A bright yellow beam of light originates from the left edge of the frame and points towards the letter 'R' in the word 'HENRY'. The beam is wider on the left and tapers as it moves to the right. The word 'HENRY' is written in a bold, black, sans-serif font.



React

Estados y LifeCycles

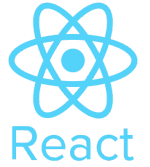
Estados



Las props son la **configuración inicial del componente**, son los datos con los que van a ser renderizados.

Pero la vida de un componente no termina ahí. De hecho, cada componente puede tener un **estado**. Podremos acceder al estado de cada componente a través del objeto en `this.state`.

Estados vs Variables



Los componentes solo van a volver a renderizarse cuando se produzcan cambios en el estado del componente o en las propiedades recibidas

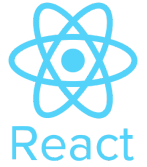


En componentes de clase es posible invocar al método `forceUpdate` pero NO se recomienda su uso

`<Demo indexState/>`

`<Demo indexStatevsVar />`

Estados



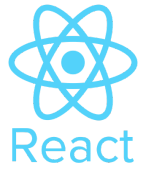
setState no siempre actualiza inmediatamente, por lo que habría que evitar leer this.state luego de haber utilizado this.setState

- Utilizar el lifecycle componentDidMount
- Agregar una función de callback al setState

```
1 ...  
2  
3 handleChange(event) {  
4   this.setState({ username: event.target.username }, function() {  
5     this.validateUsername();  
6   });  
7 }
```

<Demo setStateAsync/>

Estados



```
1 class Children extends React.Component {
2   constructor(props) {
3     super(props);
4     this.state = {
5       title: this.props.title
6     };
7   }
8
9   render(){
10    return (
11      <div>
12        <h2>{this.state.title}</h2>
13      </div>
14    )
15  }
16 };
```

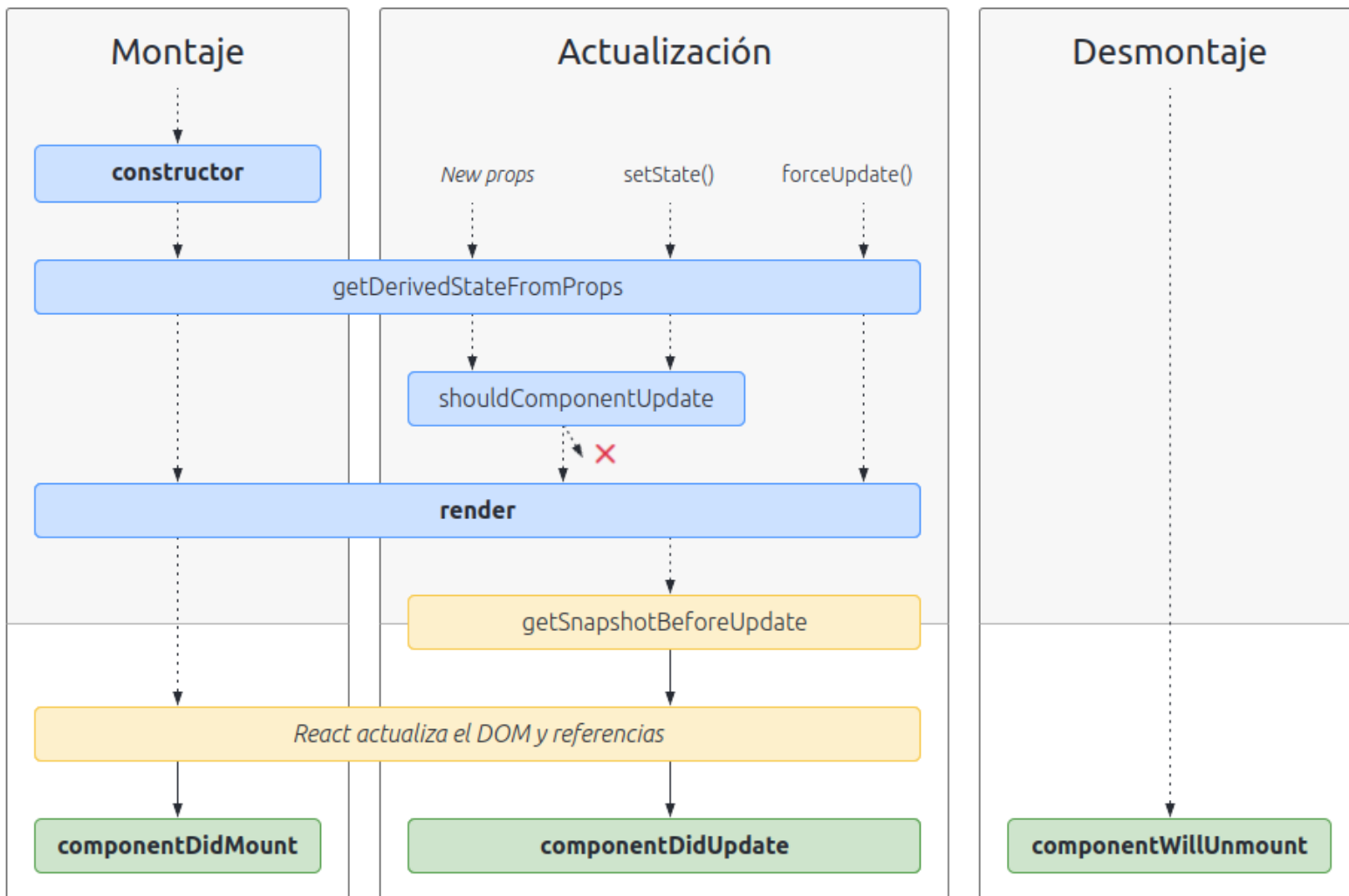


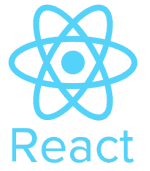
Bind con Arrow Function

```
1 class BindArrowFunction extends React.Component {
2   ...
3   this.state = {
4     mensaje: this.props.mensaje
5   }
6 }
7 handleChange = (e) => {
8   this.setState({
9     mensaje: e.target.value
10  })
11 }
12 render() {
13   return (
14     <div>
15       <input
16         type="text"
17         name="message"
18         placeholder="Ingresa mensaje"
19         onChange={this.handleChange}
20       />
21       ...
22     </div>
23   );
24 }
25 };
```

<Demo demoArrowFunctionInClass/>

Ciclo de Vida

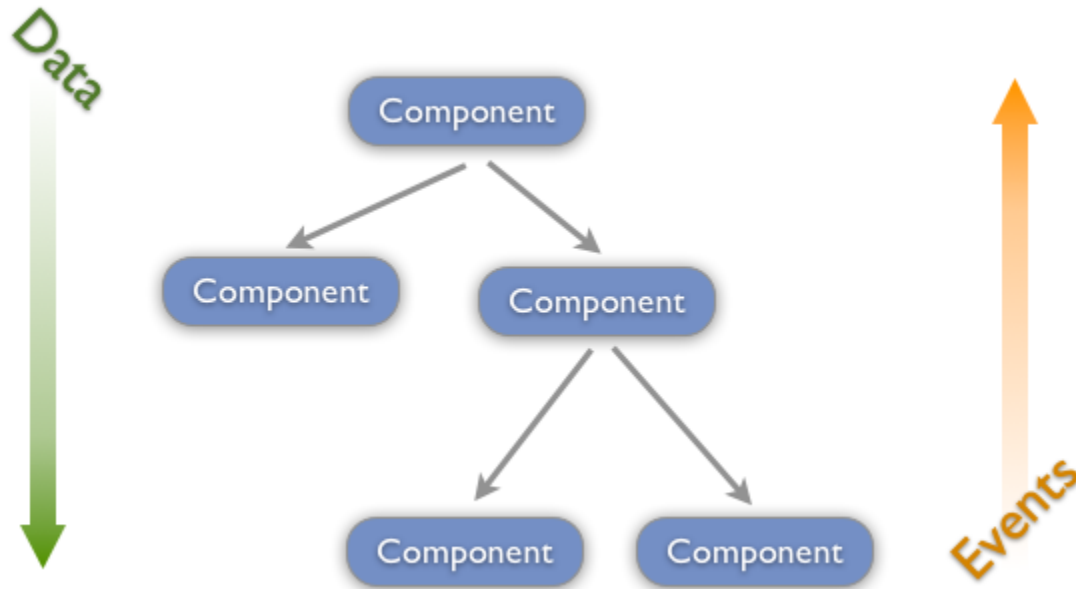


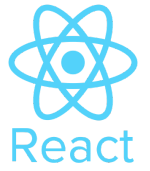


<Demo />

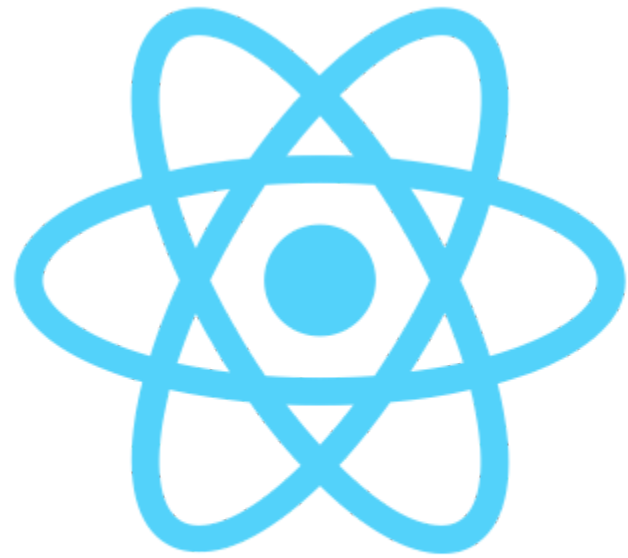
One Way Data Flow

Components y Containers





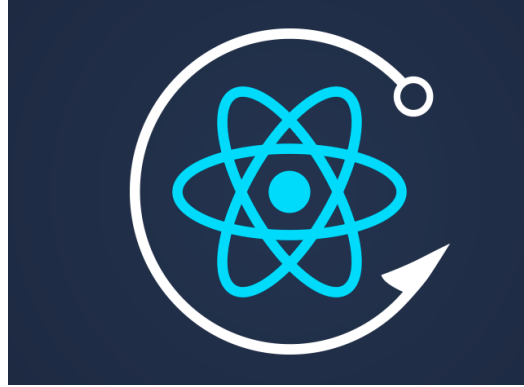
<Demo />



React

Hooks

Hooks



Los Hooks son una nueva API de la librería de React que nos permite tener estado, y otras características de React, en los componentes creados con una function. Esto, antes, no era posible y nos obligaba a crear un componente con class.



Hooks

useState

Devuelve un valor con estado y una función para actualizarlo.

```
1  
2  
3 const [state, setState] = useState(initialState);
```

Hooks

```
1  const { useState } = React;
2
3  function MuestraCuenta(props) {
4    return (
5      <p>Hola, van {props.contador}!!</p>
6    );
7  }
8
9  function Contador(props) {
10   const [contador, setContador] = useState(props.contador);
11
12   const onClick = () => {
13     setContador(contador + 1)
14   }
15
16   return (
17     <div>
18       <button onClick={onClick}>Suma uno!</button>
19       <MuestraCuenta contador={contador} />
20     </div>
21   );
22 };
```

<https://es.reactjs.org/docs/hooks-reference.html>