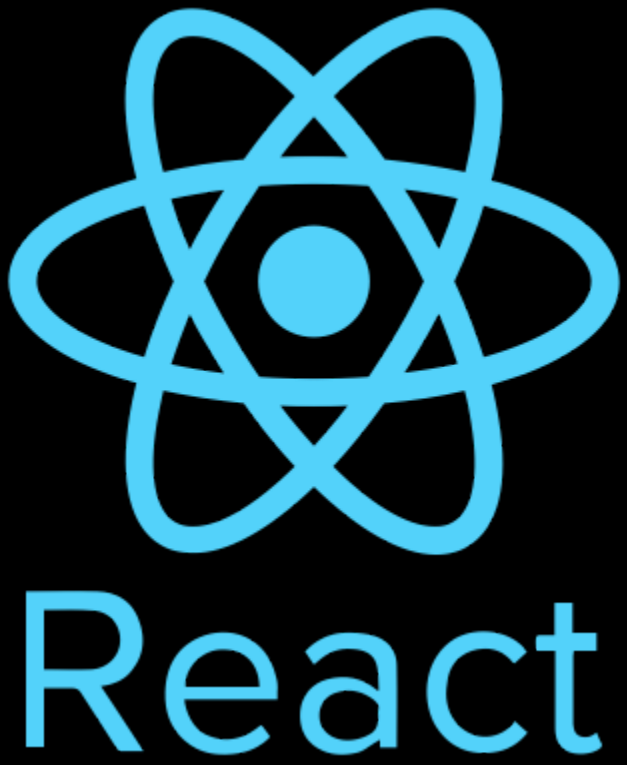
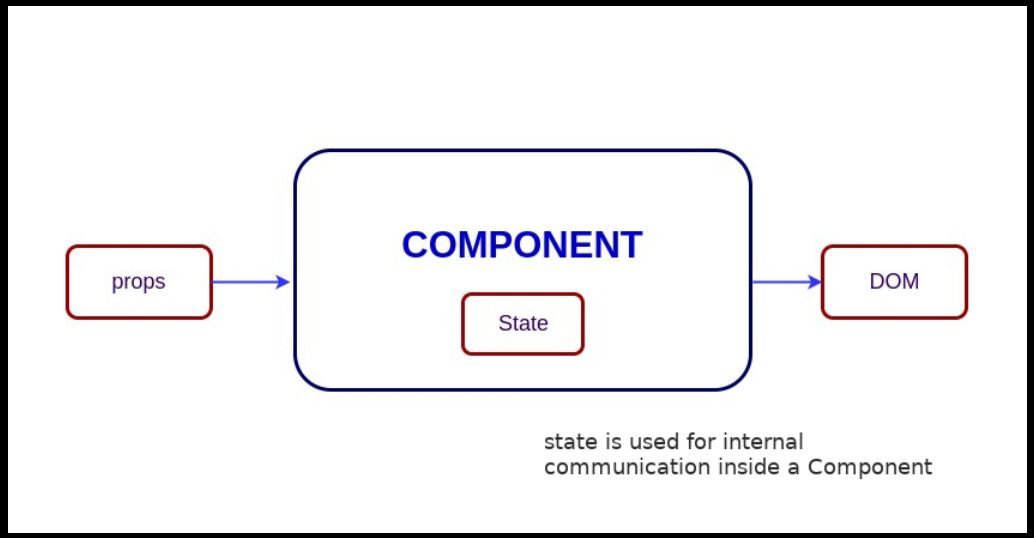

HENRY



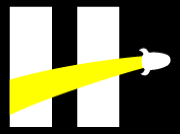
Estados y LifeCycles

Estados

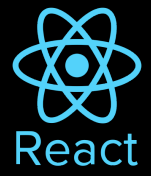


Las props son la **configuración inicial del componente**, son los datos con los que van a ser rendereados.

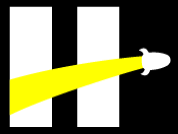
Pero la vida de un componente no termina ahi. De hecho, cada componente puede tener un **estado**. Podremos acceder al estado de cada componente a traves del objeto en this.state.



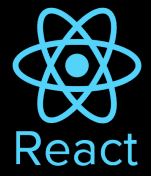
Estados




```
1
2  class Contador extends React.Component {
3   constructor(props) {
4     super(props);
5     this.state = {contador: this.props.contador}
6     this.onButtonClick = this.onButtonClick.bind(this);
7   }
8   onButtonClick(e){
9     e.preventDefault();
10    this.setState({
11    contador: this.state.contador + 1,
12    });
13  }
14  render(){
15    return (
16    <div>
17          <button onClick={this.onButtonClick}>Suma uno!</button>
18          <p>Hola, van {this.state.contador}!!</p>
19    </div>
20    )
21  }
22  };
```



Estados



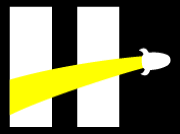
setState no siempre actualiza inmediatamente, por lo que habría que evitar leer this.state luego de haber utilizado this.setState

Posibles soluciones:

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

```
...

  handleChange(event) {
    this.setState({ username: event.target.username }, function() {
      this.validateUsername();
    });
  }
```



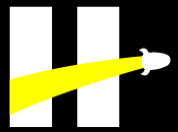
Estados

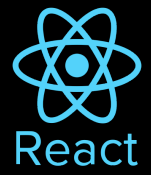
React

```
class Children extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      title: this.props.title
    };
  }

  render(){
    return (
      <div>
      <h2>{this.state.title}</h2>
      </div>
    )
  }
};
```

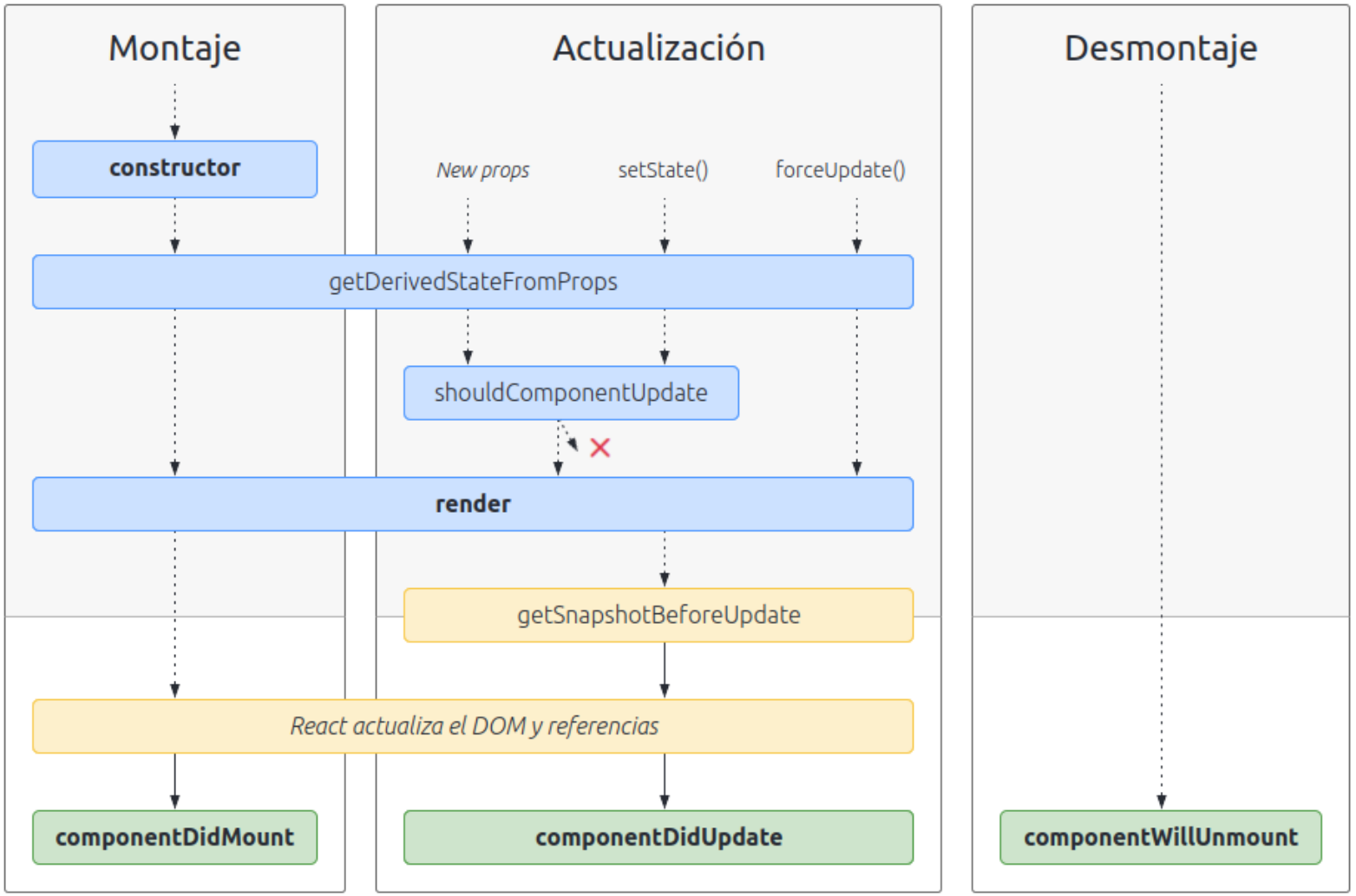
<Demo copyState/>





<Demo />

Ciclo de Vida



"Fase Render"

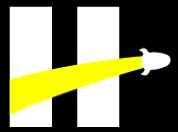
Pura y sin efectos co-laterales, puede ser pausada, abortada o reiniciada por React.

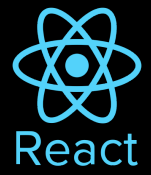
"Fase Pre-commit"

Puede leer el DOM.

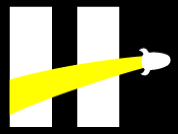
"Fase Commit"

Puede operar sobre el DOM, ejecutar side-effects, agendar ac-tualizaciones.





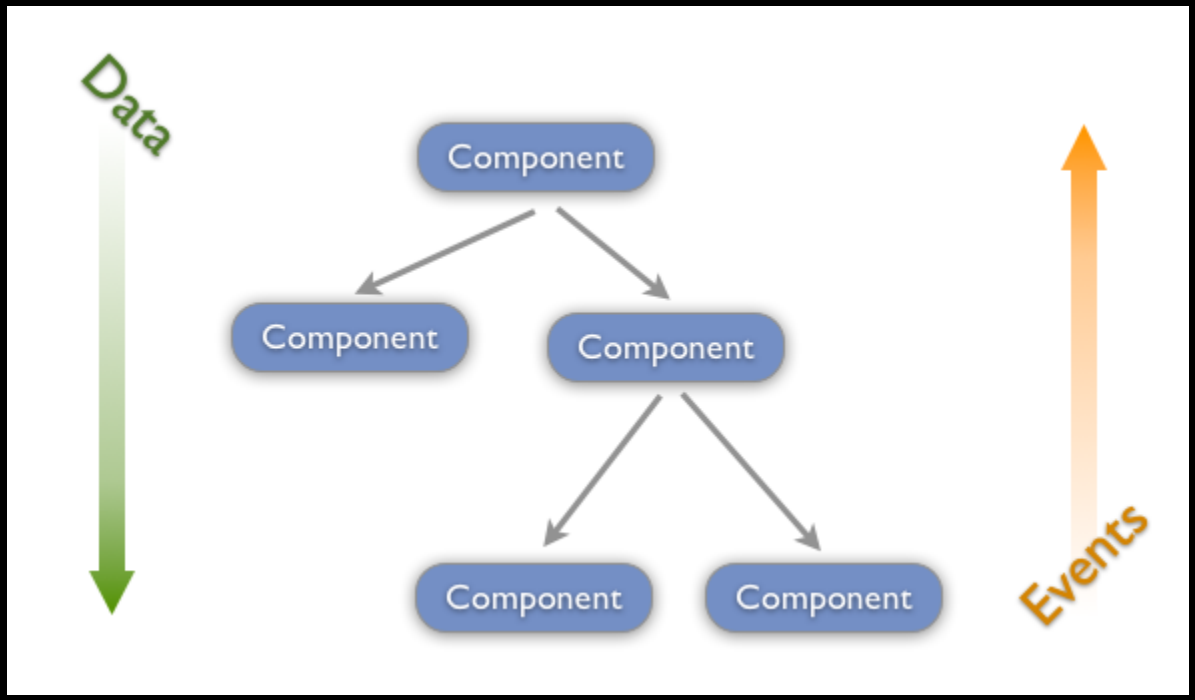
<Demo />

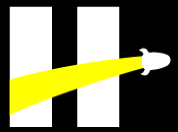


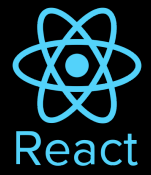
One Way Data Flow

Components y Containers

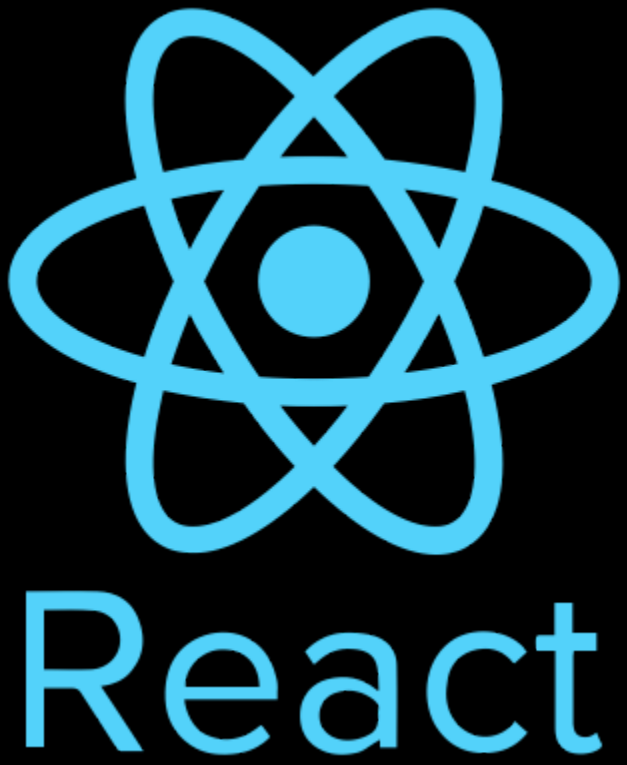
React



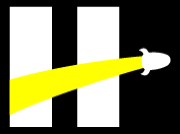




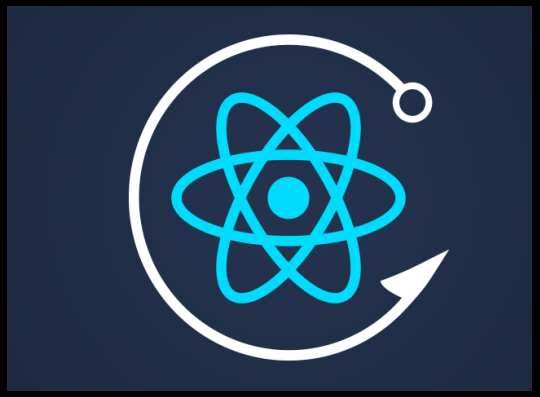
<Demo />



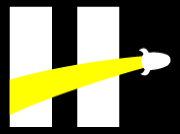
Hooks



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.

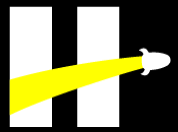


Hooks

useState

Devuelve un valor con estado y una función para actualizarlo.

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



Hooks

```
const { useState } = React;

function MuestraCuenta(props) {
  return (
    <p>Hola, van {props.contador}!!</p>
  );
}

function Contador(props) {
  const [contador, setContador] = useState(props.contador);

  const onButtonClick = () => {
    setContador(contador +1)
  }

  return (
    <div>
        <button onClick={onButtonClick}>Suma uno!</button>
                <MuestraCuenta contador={contador} />
        </div>
  );
};
```

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