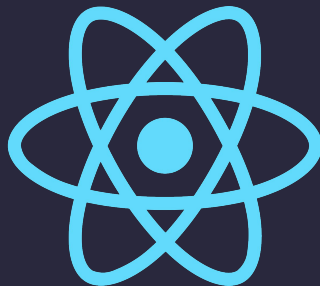


# React



Felipe Guizar  
Victor Servin



<https://github.com/vservin/react-laboratoria>

# React



## Declarativo

Permite crear UIs  
interactivas  
eficientes sin  
dolor



## Componentes

Construimos  
componentes  
encapsulados que  
manejan su propio  
estado



## JSX/TSX

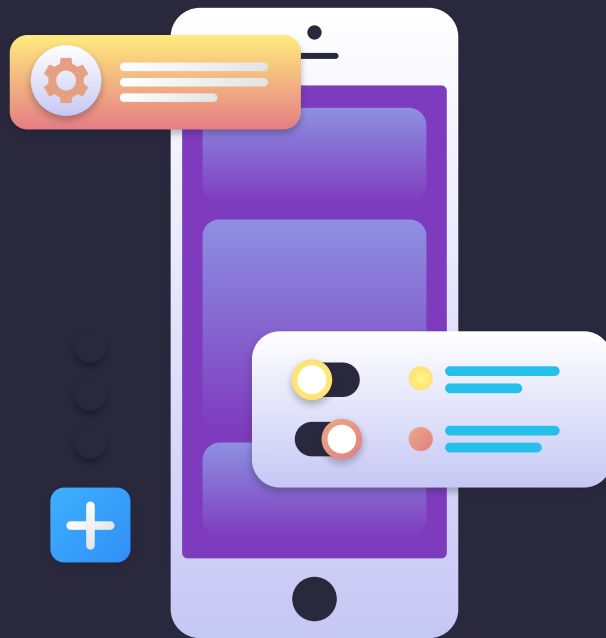
Introduce  
Javascript XML una  
extensión al  
sintaxis de JS



# /01

# JSX

<https://es.reactjs.org/docs/introducing-jsx.html>





```
const elemento = <h3>iHola mundo!</h3>;
```




```
1 const nombre = 'Samantha Reyes';  
2 const elemento = <h1>Hola, {nombre}</h1>
```



```
function convierteNombre(usuario) {  
  return `${usuario.nombre} ${usuario.apellido}`;  
}  
  
const usuario = {  
  nombre: 'Spencer',  
  apellido: 'Hastings'  
};  
  
const elemento = (  
  <h1>  
    Hola, extraño!  
  </h1>  
)  
);  
  
function traeSaludo(usuario) {  
  if (usuario) {  
    return <h1>Hola, {convierteNombre(usuario)}!</h1>;  
  }  
  return elemento;  
}
```



# Esencialmente, JSX representa Objetos



```
const elemento = (  
  <h1 className="saludo">  
    ¡Hola mundo!  
  </h1>  
);
```

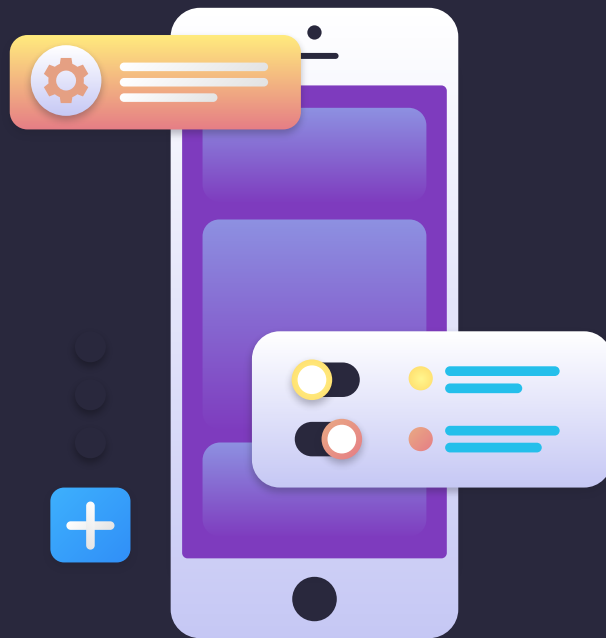
```
const elemento = React.createElement(  
  'h1',  
  {className: 'saludo'},  
  '¡Hola mundo!'  
);
```



/02

# Componentes

<https://es.reactjs.org/docs/components-and-props.html>

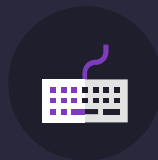


## Componentes Funcionales



```
function BienvenidaFuncional(props) {  
  return <h3>Hola, {props.nombre}!</h3>;  
}
```

## Componentes de clase



```
class BienvenidaClase extends React.Component {  
  render() {  
    return <h1>Hola, {this.props.nombre}</h1>;  
  }  
}
```

# Componentes



## Puros

Todos los componentes deben ser funciones puras



## Reusables

Buscan implementar patrón DRY



## Anidados

Un componente puede contener múltiples componentes



## A la medida

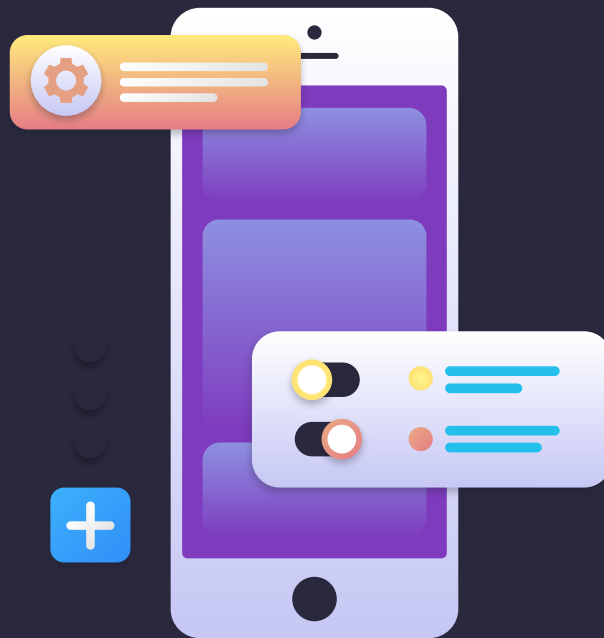
Los componentes que se cargan/descargan del DOM en tiempo de ejecución

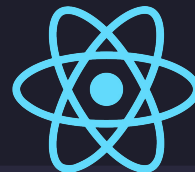


# /03

## Eventos

<https://es.reactjs.org/docs/handling-events.html>



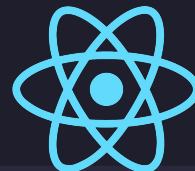


```
1 <button onclick="activarLasers()">
2   Activar láser
3 </button>
4
5 function activarLasers() {
6   alert("Láser activados!");
7 }
```



```
1 function ComponenteLasers() {
2   const activarLasers = () => {
3     alert("Láser activados!");
4   };
5   return (
6     <button onClick={activarLasers}>
7       Activar láser
8     </button>
9   );
10 }
11
```





```
1 <button onclick="activarLasers()">
2   Activar láser
3 </button>
4
5 function activarLasers() {
6   alert("Láser activado!");
7 }
```



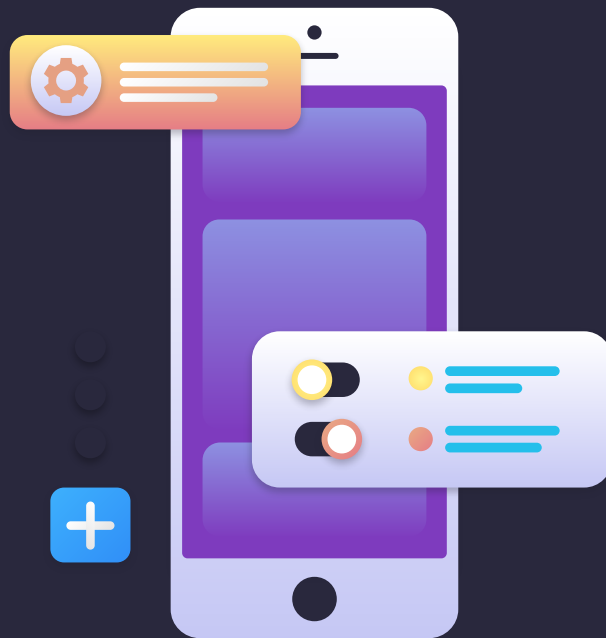
```
1 class ComponenteLasers extends React.Component {
2   activarLasers() {
3     alert("Láser activado!");
4   }
5
6   render() {
7     return (
8       <button onClick={() => this.activarLasers()}>
9         Activar láser
10      </button>
11    );
12  }
13 }
```



/04

# Listas y llaves

<https://es.reactjs.org/docs/lists-and-keys.html>

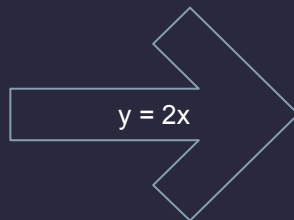




# Listas o Arreglos



```
1 const numeros = [1, 2, 3, 4, 5];  
2 const dobles = numeros.map((num) => num * 2);  
3 console.log(dobles);  
4 // [2, 4, 6, 8, 10]
```



# Listas o Arreglos

```
1 const listaDeSuper = [  
2   "Leche",  
3   "Jamón",  
4   "Vino tinto",  
5   "Queso suizo",  
6   "Croutones"  
7 ];  
8  
9 root.render(  
10   <div>  
11     <h2>Lo que tengo que comprar de super es:</h2>  
12     <ul>  
13       {listaDeSuper.map((item) => <li>{item}</li>)}  
14     </ul>  
15   </div>  
16 );
```



React element





# Listas o Arreglos



```
1 const listaDeSuper = [  
2   "Leche",  
3   "Jamón",  
4   "Vino tinto",  
5   "Queso suizo",  
6   "Croutones"  
7 ];  
8  
9 root.render(  
10   <div>  
11     <h2>Lo que tengo que comprar de super es:</h2>  
12     <ul>  
13       {listaDeSuper.map((item) => <li>{item}</li>)}  
14     </ul>  
15   </div>  
16 );
```

✖ ▶Warning: Each child in a list should have a unique "key" prop. [react.development.js:199](#)

Check the top-level render call using <ul>. See <https://react.js.org/link/warning-keys> for more information.  
at li





# Listas o Arreglos



```
1 function ListaDeSuper(props) {  
2   const { porComprar } = props  
3   const elementosDeLista = porComprar.map((item, index) =>  
4     <li key={`_${index}_${item}`} >{item}</li>  
5   );  
6   return <ul>{elementosDeLista}</ul>;  
7 }  
8  
9 const milista = ["Leche", "Jamón", "Vino tinto", "Queso suizo", "Croutones"];  
10  
11 const root = ReactDOM.createRoot(document.getElementById('root'));  
12 root.render(  
13   <div>  
14     <ListaDeSuper porComprar={milista} />  
15   </div>  
16 );  
17
```

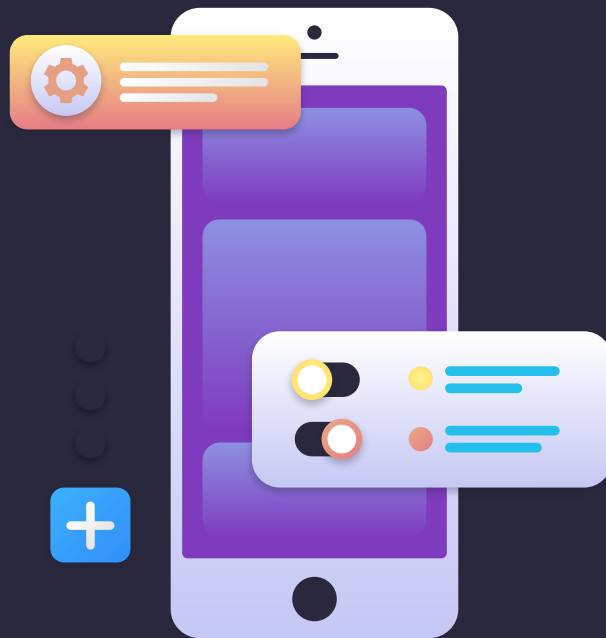




/05

# Renderizado condicional

<https://es.reactjs.org/docs/conditional-rendering.html>





```
1 function BienvenidaUsuario(props) {  
2   return <h1>¡Bienvenido de vuelta {props.nombre}!</h1>;  
3 }  
4 function BienvenidaInvitado() {  
5   return <h1>Por favor, inicia sesión.</h1>;  
6 }
```

¡Bienvenidx de vuelta Emily!

Por favor, inicia sesión.



```
1 function Bienvenida(props) {  
2   const { usuario } = props;  
3   if (usuario) {  
4     return <BienvenidaUsuario nombre={usuario.nombre} />;  
5   }  
6   return <BienvenidaInvitado />;  
7 }
```



```
1 root.render(<Bienvenida usuarix={{ nombre: 'Emily' }} />);  
2 root.render(<Bienvenida />);
```



```
1 function Correspondencia(props) {
2   const { mensajesSinLeer } = props;
3   return <div>
4     <h2>Bienvenidx!</h2>
5     { mensajesSinLeer.length > 0 &&
6       <p>¡Parece que tienes {mensajesSinLeer.length} mensajes sin leer!</p>
7     }
8   </div>;
9 }
```

Operador lógico “Y” o “AND”

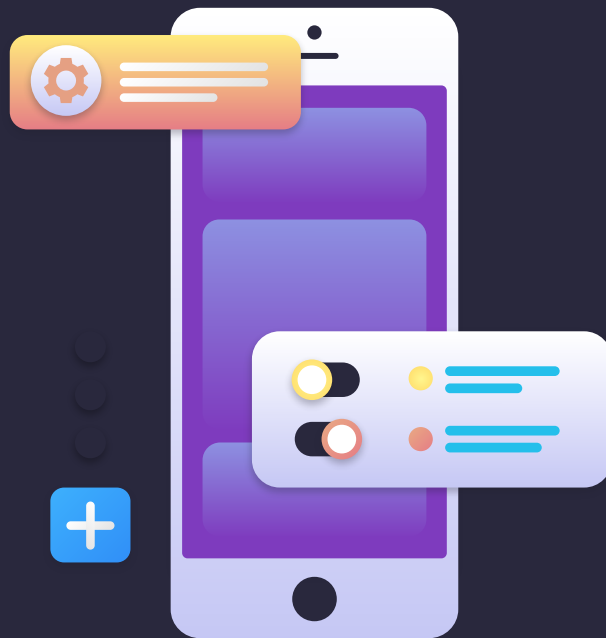
Operador ternario

```
1 function Correspondencia(props) {
2   const { mensajesSinLeer } = props;
3   return <div>
4     <h2>Bienvenidx!</h2>
5     { mensajesSinLeer.length > 0
6       ? <p>¡Parece que tienes {mensajesSinLeer.length} mensajes sin leer!</p>
7       : <p>No tienes ningún mensaje nuevo 🐱</p>
8     }
9   </div>;
10 }
```

/06

# Estado

<https://es.reactjs.org/docs/lifting-state-up.html>



## Inicializar y mostrar el estado

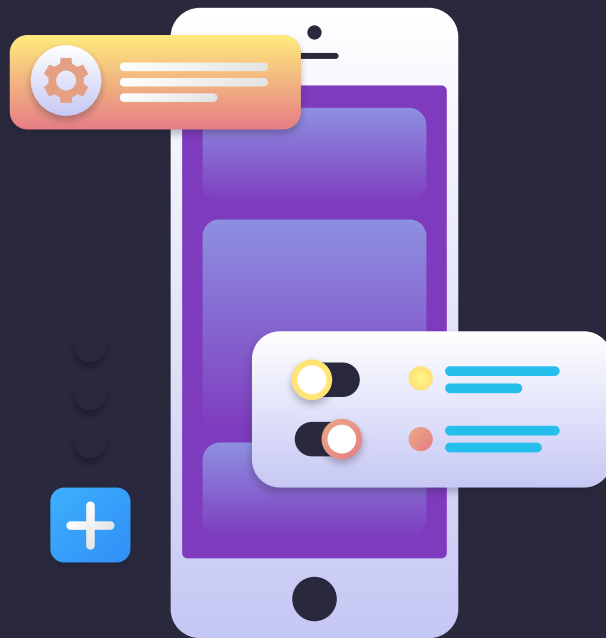
```
1 class Contador extends React.Component {  
2   constructor(props) {  
3     super(props);  
4     this.state = {  
5       count: 0,  
6     };  
7   }  
8  
9   render() {  
10    return <h1>{this.state.count}</h1>;  
11  }  
12 }  
13  
14 const root = ReactDOM.createRoot(document.getElementById('root'));  
15 root.render(<Contador />);  
16
```

```
1 class Contador extends React.Component {
2   constructor(props) {
3     super(props);
4     this.state = {
5       count: 0,
6     };
7   }
8
9   reiniciar() {
10    this.setState({ count: 0 });
11  }
12
13  agregarOQuitar(valor) {
14    this.setState((s) => ({ count: s.count + valor }));
15  }
16
17  render() {
18    return <div>
19      <h1>{this.state.count}</h1>
20      <button onClick={() => this.agregarOQuitar(-1)}>Quitar</button>
21      <button onClick={() => this.reiniciar()}>Reiniciar</button>
22      <button onClick={() => this.agregarOQuitar(1)}>Añadir</button>
23    </div>;
24  }
25 }
26
27 const root = ReactDOM.createRoot(document.getElementById('root'));
28 root.render(<Contador />);
29
```

/07

# Hooks

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



# useState

Permite usar estados “state” en componentes funcionales.

Este hook regresa el valor de estado, y un asignador “setter” que permite actualizar el estado.

¡Cuidado! El asignador siempre siempre vuelve a “renderizar” el componente



```
import { useState } from "react";

const FooBar = () => {
  const [value, setValue] = useState("foo");

  const onClick = () => {
    setValue("bar");
  };

  return (
    <div>
      <button onClick={onClick}>Presiona!</button>
      <p>{text}</p>
    </div>
  );
};
```



# useEffect

Ejecutar side-effects como suscribir eventos al DOM, Data fetching, logging, etc.

Los “efectos” se ejecutan una vez React ha actualizado el DOM y sus dependencias han cambiado.



```
import { useEffect } from "react";

useEffect(() => {
  // se ejecuta cada vez que el
  // componente se actualiza
  // Equivalente a componentDidUpdate
})

useEffect(() => {
  // se ejecuta la primera vez que el
  // componente actualizado el DOM.
  // Equivalente a componentDidMount
}, [])

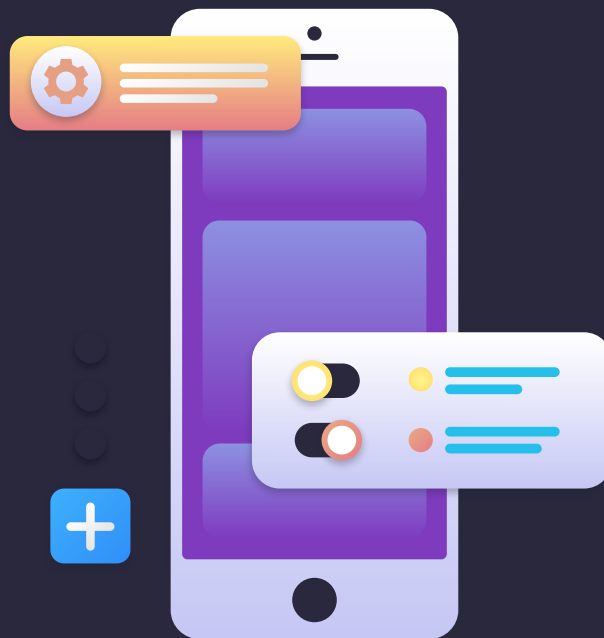
useEffect(() => {
  // se ejecuta cada vez que el
  // componente se actualiza
  // y las dependencias han cambiado
}, [dependency])
```



/08

# Módulos CSS

<https://create-react-app.dev/docs/adding-a-css-modules-stylesheet/>





npx create-react-app mi-primer-app

08-css-modules

node\_modules

public

src

# App.css

JS App.js

# App.module.css

JS App.test.js

# index.css

JS index.js

logo.svg

JS reportWebVitals.js

JS setupTests.js

.gitignore

package-lock.json

package.json

README.md

08-css-modules > src > JS index.js > ...

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6
7 const root = ReactDOM.createRoot(document.getElementById('root'));
8 root.render(
9   <React.StrictMode>
10     <App />
11   </React.StrictMode>
12 );
13
14 // If you want to start measuring performance in your app, pass a function
15 // to log results (for example: reportWebVitals(console.log))
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17 reportWebVitals();
18
```





# npx create-react-app mi-primer-app



```
1 // Importar hoja de estilos regular
2 import './App.css';
3 // Importar un módulo CSS y se asigne a una variable "estilos"
4 import estilos from './App.module.css';
5
6 function App() {
7   return (
8     <p className={estilos.pruebaTexto}>Hola desde mi componente App!!!</p>
9   );
10 }
11
12 export default App;
13
```

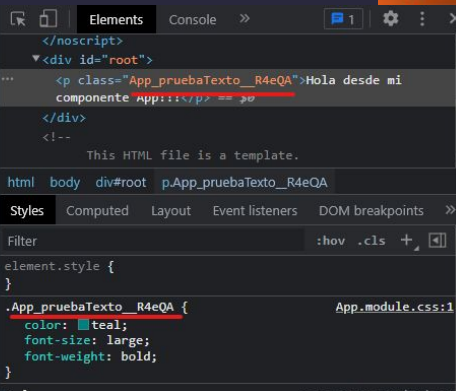


```
1 body {
2   background-color: aliceblue;
3 }
```



```
1 .pruebaTexto {
2   color: teal;
3   font-size: large;
4   font-weight: bold;
5 }
```

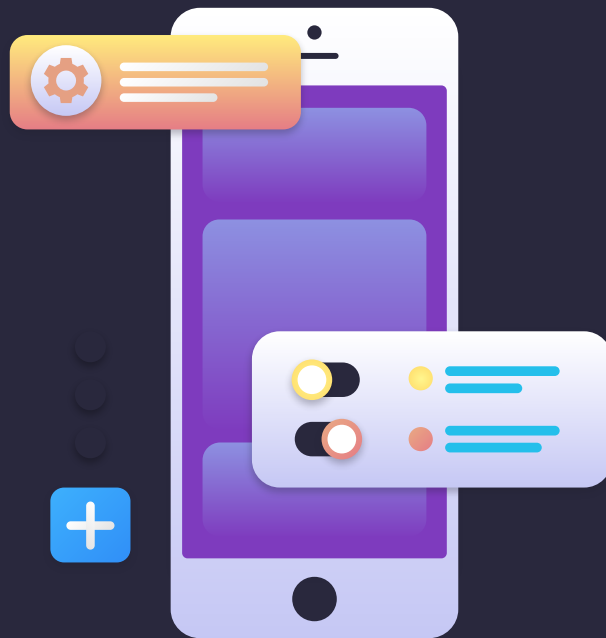
Hola desde mi componente App!!!



/09

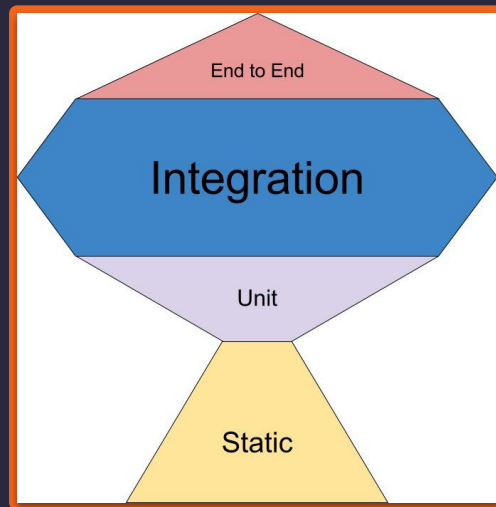
# Pruebas

<https://testing-library.com/docs/react-testing-library/intro>



# Pruebas

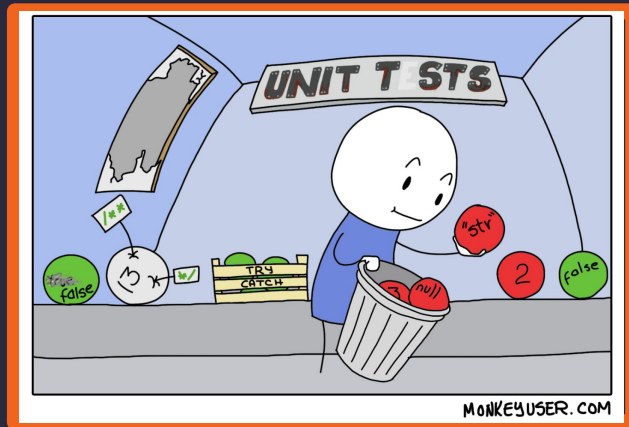
Unit testing es la metodología más simple y barata. Sin embargo, involucra detalles de implementación que son propensos a cambiar al modificar el código.



# Pruebas

En este caso, si el resultado esperado es el mismo y la implementación ha cambiado, se pueden generar falsos negativos.

En el peor caso, generar falsos positivos.



# Gracias!

## Tienes alguna pregunta?

<https://github.com/vservin/react-laboratoria>



CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

