#>/<>

HACK A BOSS

<CODE YOUR TALENT>

¿Quién soy?

Fernando Blanco

- Argalleiro e “fillo de taberneiro”
- Aprendiz de informático
- Senior Software Developer

#>/<>
HACK A BOSS
<CODE YOUR TALENT>

@ferblancodosil

Paso 2

#>/<>
HACK A BOSS
<CODE YOUR TALENT>

React

- Componentes
- Ciclo de vida
- Props y children
- Estado
- Fragments
- Listas
- Renderizado condicional
- Eventos
- Refs

@ferblancodosil

Paso 2

Componentes

@ferblancodosil

¿Qué es un componente?

Un componente es un elemento de un sistema que ofrece un servicio predefinido, y es capaz de comunicarse con otros componentes.

Un componente debe ser diseñado e implementado de tal forma que pueda ser reutilazado.

React: Componentes

- ~~Elemento~~ "Trozo" visual reusable
- Conjunto de elementos React
- Puede tener estado
- Por convenio, PascalCase

function Component

- en forma de funciones
- return es render()
- ~~sin estado~~ (desde hooks también tienen estado)

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Class component

- usando clases ES6
- extend React.Component

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

- pueden tener estado

React: Componentes

- *props* son "solo lectura"
- regla: All React components must act like pure functions with respect to their props.
- Componer componentes
- Extraer componentes

REACT

Componer componentes

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}
```

REACT

Extraer componentes

```
function Comment(props) {
  return (
    <div className="Comment">
      <div className="UserInfo">
        <img className="Avatar"
          src={props.author.avatarUrl}
          alt={props.author.name}
        />
        <div className="UserInfo-name">{props.author.name}</div>
      </div>
      <div className="Comment-text">{props.text}</div>
    </div>
  );
}
```

REACT

Extraer componentes

```
const Avatar = (props) => <img className="Avatar"
  src={props.user.avatarUrl}
  alt={props.user.name}
/>;
function Comment(props) {
  return (
    <div className="Comment">
      <div className="UserInfo">
        <Avatar user={props.author} />
        <div className="UserInfo-name">{props.user.name}</div>
      </div>
      <div className="Comment-text">{props.text}</div>
    </div>
  );
}
```

React: Estilos

- Dos formas “naturales” de establecer estilos
  - Estilos en línea con la prop `style`
  - CSS normal mediante la prop `className`
- Librerías avanzadas CSS-in-JS (emotion, styled-components...)
  - Css In Your JS (transparencias)

React: prop **style**

- En HTML se pasa un string de CSS

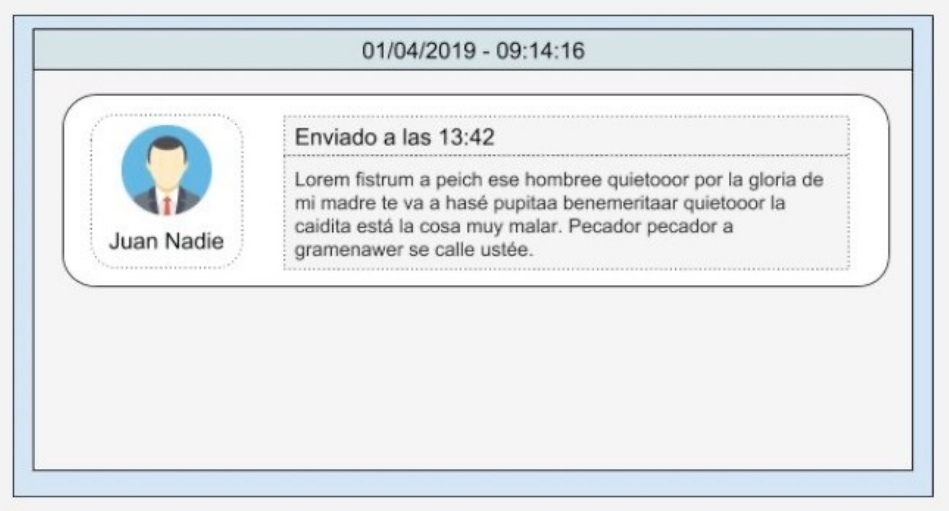
```
<div style="margin-top: 20px; background-color: blue;"></div>
```

- En React se pasa un objeto de CSS

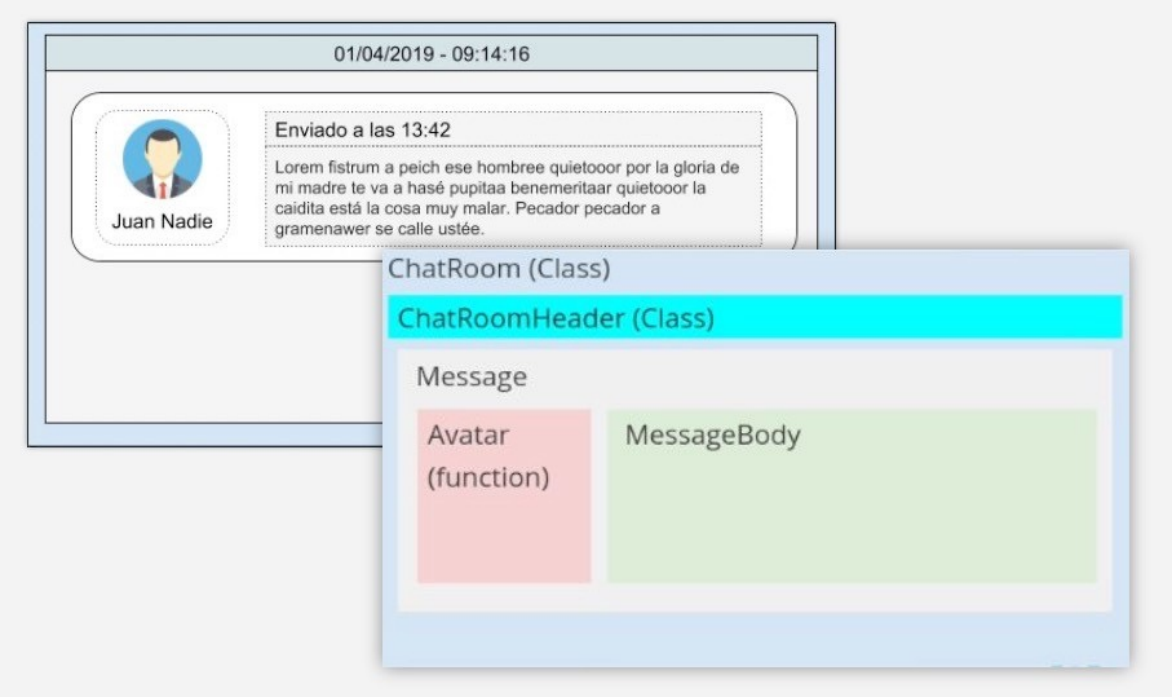
```
<div style={{marginTop: 20, backgroundColor: 'blue'}} />
```

- `{{...attr}}` →expresión que envuelve un objeto
- Las propiedades se pasan a `camelCase`

¿En qué componentes descompondrías esta vista?



¿En qué componentes descompondrías esta vista?



Componente creado como Componente VS Función

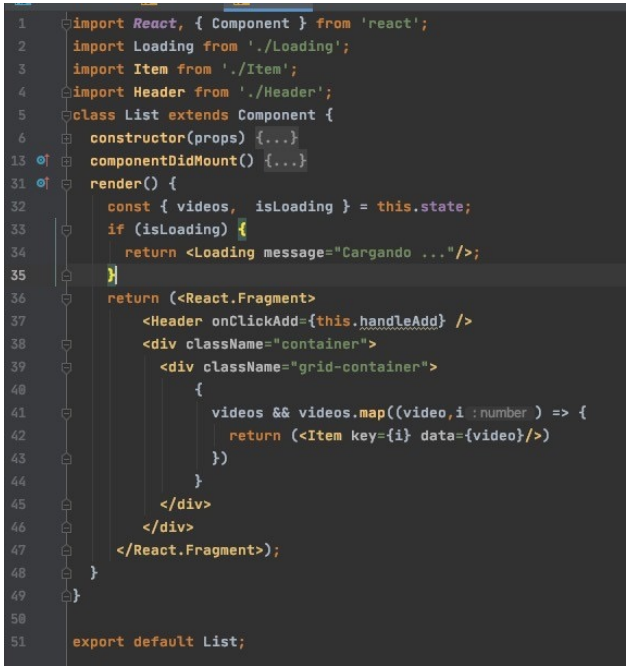
Función: estructura más primitiva.

Función: solo pueden recibir propiedades y no tienen estado.

Función: no tiene ”this” para utilizar propiedades.

Función: orientado a componentes muy simples y más fácil reusar.

Componente creado como Componente VS Función



```
import React, { Component } from 'react';
import Loading from './Loading';
import Item from './Item';
import Header from './Header';
class List extends Component {
  constructor(props) {...}
  componentDidMount() {...}
  render() {
    const { videos,  isLoading } = this.state;
    if (isLoading) {
      return <Loading message="Cargando ..."/>;
    }
    return (<React.Fragment>
        <Header onClickAdd={this.handleAdd} />
        <div className="container">
          <div className="grid-container">
              {
                videos && videos.map((video,i : number ) => {
                  return (<Item key={i} data={video}/>)
                })
              }
          </div>
        </div>
      </React.Fragment>);
  }
}

export default List;
```

#>/<>
HACK A BOSS
<CODE YOUR TALENT>

Componente creado como Componente VS Función

```jsx
import React from "react";
import PropTypes from 'prop-types';

const Item = ({ data }) => (
  <div className="grid-item" >
    <img className="preview-image" src={data.thumbnail} alt={data.title}/>
    <div className="preview-title">{data.title}</div>
  </div>
)

Item.propTypes = {
  data: PropTypes.object.isRequired
};

export default Item;
```

Paso 2

Ciclo de vida

#>/<>
HACK A BOSS
<CODE YOUR TALENT>

@ferblancodosil

Lifecycle: Mount

- **constructor()** asignar this.state directamente
- static getDerivedStateFromProps()
- **render()**
- **componentDidMount()**
- ~~UNSAFE_componentWillMount()~~

Lifecycle: Update

- static getDerivedStateFromProps()
- shouldComponentUpdate()
- **render()**
- getSnapshotBeforeUpdate()
- **componentDidUpdate()**
- ~~UNSAFE_componentWillUpdate()~~
- ~~UNSAFE_componentWillReceiveProps()~~

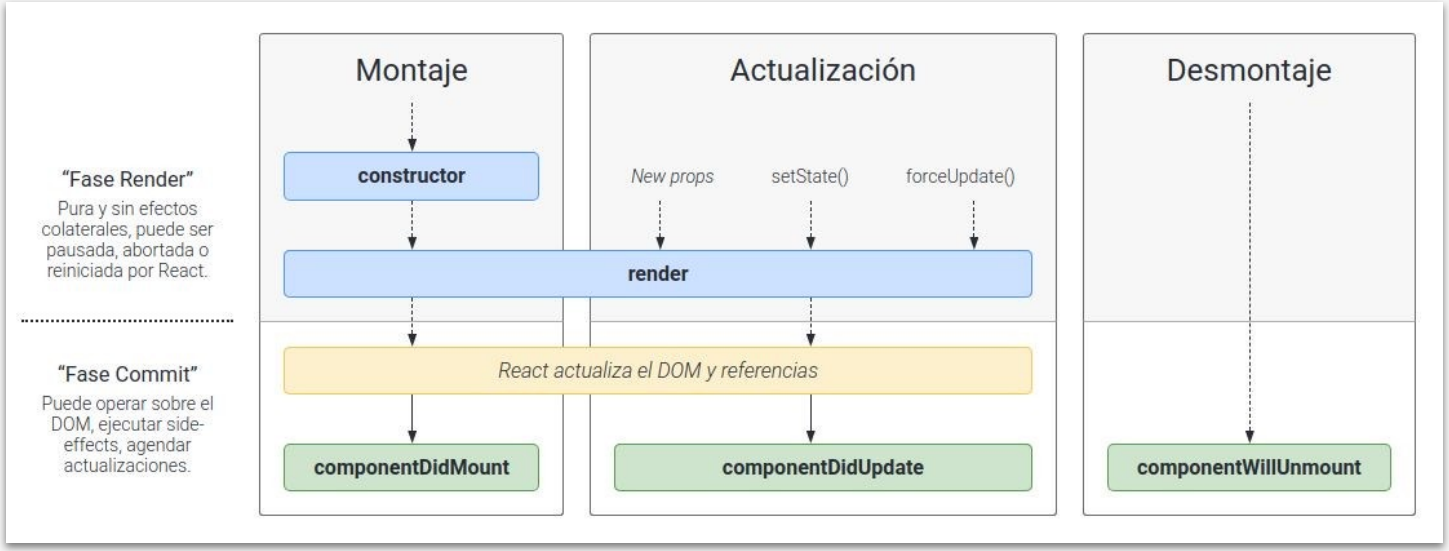
Lifecycle: Unmount

- **componentWillUnmount()**

Lifecycle: Error handling

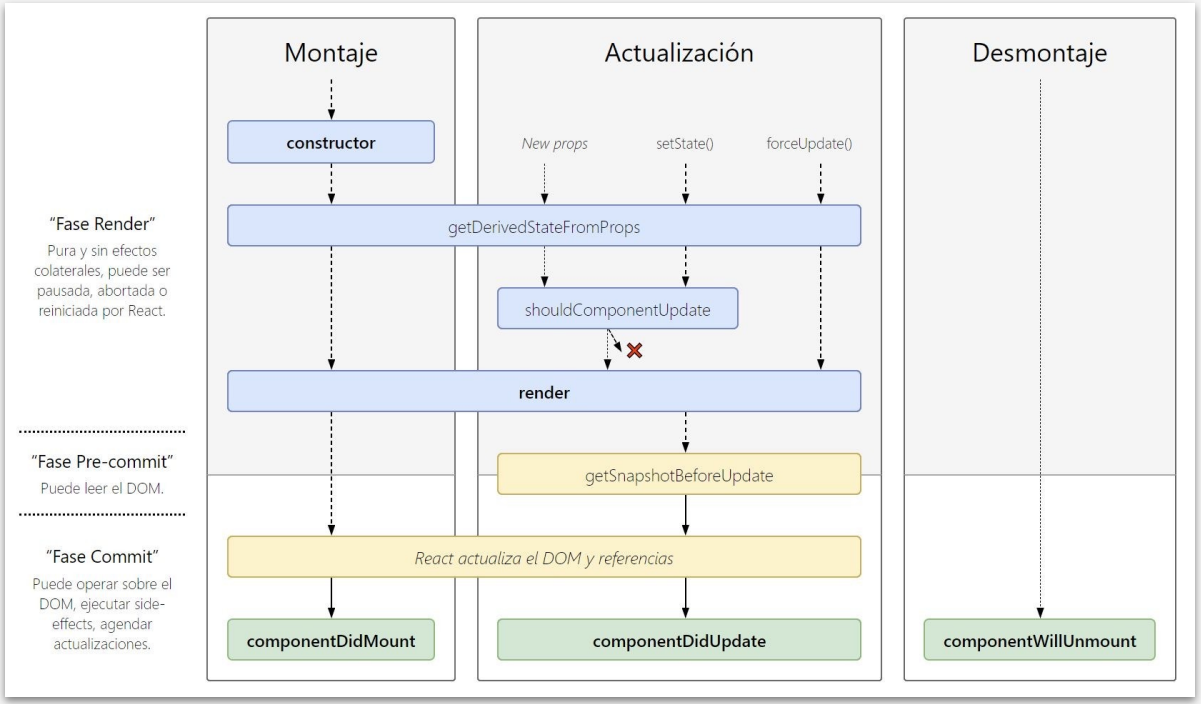
- static getDerivedStateFromError()
- componentDidCatch()

Lifecycle

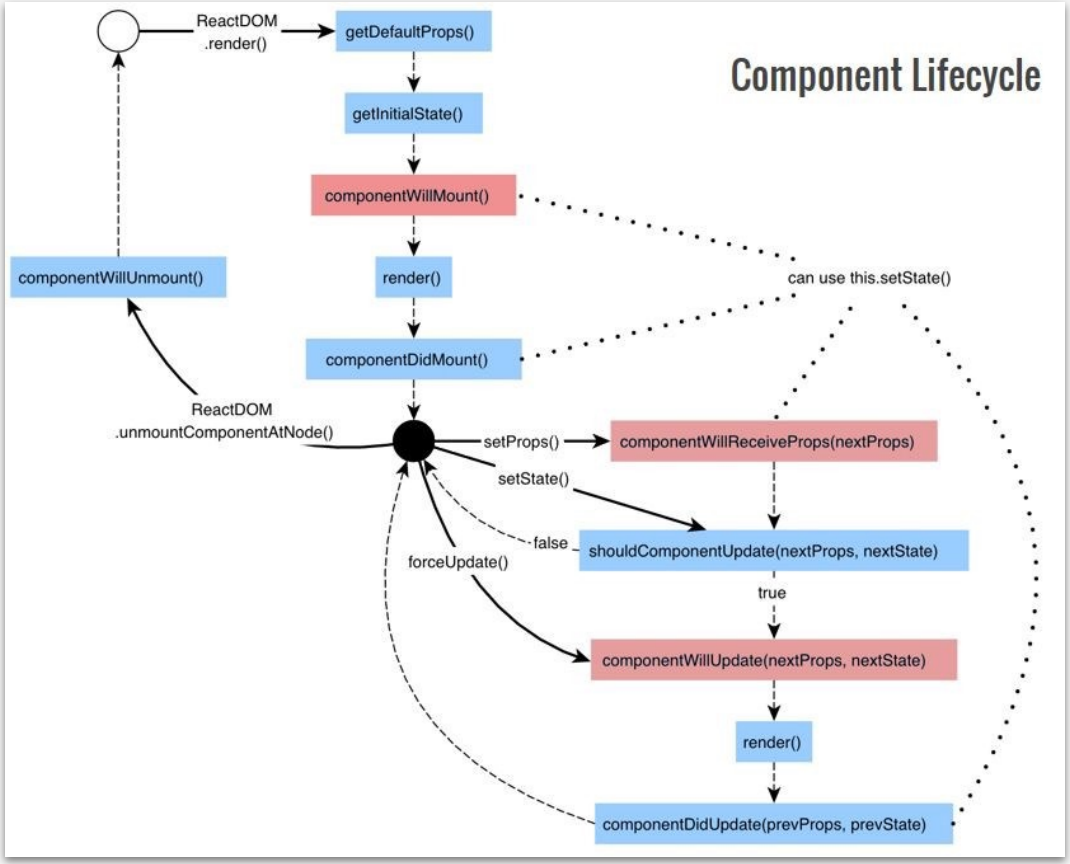


http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

Lifecycle


Montaje
Actualización
Desmontaje
constructor
New props
setState()
forceUpdate()
getDerivedStateFromProps
shouldComponentUpdate
render
getSnapshotBeforeUpdate
React actualiza el DOM y referencias
componentDidMount
componentDidUpdate
componentWillUnmount
"Fase Render"
Pura y sin efectos colaterales, 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 actualizaciones.

Lifecycle


Component Lifecycle
ReactDOM
.render()
getDefaultProps()
getInitialState()
componentWillMount()
render()
componentDidMount()
componentWillUnmount()
ReactDOM
.unmountComponentAtNode()
can use this.setState()
setProps()
componentWillReceiveProps(nextProps)
setState()
false
shouldComponentUpdate(nextProps, nextState)
forceUpdate()
true
componentWillUpdate(nextProps, nextState)
render()
componentDidUpdate(prevProps, prevState)

artista

Artista Favorito

Faxu

El artista del alambre

Amaral

Compositor Del Año

Joel Ney

Pareja Del Año

Sebastian Yatra

Paso 2

Props y children

#>/<>
HACK A BOSS
<CODE YOUR TALENT>

@ferblancodosil

React: props en function component

```
1  import React from "react";
2  import PropTypes from 'prop-types';
3
4  const Item = ({ data }) => (
5    <div className="grid-item" >
6      <img className="preview-image" src={data.thumbnail} alt={data.title}/>
7      <div className="preview-title">{data.title}</div>
8    </div>
9  )
10
11 Item.propTypes = {
12   data: PropTypes.object.isRequired
13 };
14
15 export default Item;
```

React: props en function component

```jsx
import React from "react";
import PropTypes from 'prop-types';

const Item = ({ data }) => (
  <div className="grid-item" >
    <img className="preview-image" src={data.thumbnail} alt={data.title}/>
    <div className="preview-title">{data.title}</div>
  </div>
)

Item.propTypes = {
  data: PropTypes.object.isRequired
};

export default Item;
```

```jsx
{
  videos && videos.map((video,i) => {
    return (<Item key={i} data={video}/>)
  })
}
```

React: props en class component

```
import React, {PureComponent} from "react";
import PropTypes from 'prop-types';

class Add extends PureComponent {
    constructor (props) {
        super(props);
    }
    handleSubmit(e){
        e.preventDefault();
        const { onClose } = this.props;
        const token = parseYoutubeUrl(this.state.url || '');
        if(this.validation(this.state) && token){
            this.setState({showSending:true})
            addVideo({
                title: this.state.title,
                description: this.state.description,
                url: this.state.url,
                thumbnail: `https://img.youtube.com/vi/${token}/maxresdefault.jpg`,
                embed: `https://www.youtube.com/embed/${token}`
            }).then(onClose(true));
        }else{
            this.setState({
                hasError:true
            });
        }
    }
}

Add.propTypes = {
    onClose: PropTypes.func.isRequired
};

export default Add;
```

```
return (<Add onClose="close"/>)
```

Paso 2

Estado

@ferblancodosil

React: Estado

- Local a cada componente
- **NUNCA** se cambia a mano
- Sólo class component (hasta que llegó hooks)

React: Estado

- constructor invoca super(props)
- Inicializar estado

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }
  render() {
    return (
      <div>
        <h2>
          {this.state.date.toLocaleTimeString()}
        </h2>
      </div>
    );
  }
}
```

React: Estado

- acceso: this.state.[...]
- modificación: this.setState()

MAL

```
this.state.carColor = 'red';
```

BIEN

```
this.setState({carColor: 'red'});
```

React: Estado

- modificación puede ser asíncrona
- cuando depende de props o

  state => this.setState(fn) MAL

```
this.setState({
  counter: this.state.counter + this.props.increment,
});
```

BIEN fn: (state, props) => ({nextState})

```
this.setState((state, props) => ({
  counter: state.counter + props.increment
}));
```

React: Estado

- this.setState({}) mezcla
  - superficial (shallow)

```
//ANTES
this.state = {
  posts: [],
  comments: ["comment1", "comment2"]
};
this.setState({
  comments: ["comment3"]
})
//DESPUÉS
this.state = {
  posts: [],
  comments: ["comment3"]
};
```

React: useState

- Para... usar el state
- Desestructuración al asignar
- Parámetro: valor inicial

```
const [count, setCount] = useState(0);
```

Atención: esto es un **hook**.

Se explicarán en detalle más adelante

REACT

React: useState

```
import React, { useState } from 'react';

function Example() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
       Click me
      </button>
    </div>
  );
}
```

Paso 2

Fragments

@ferblancodosil

Fragments

Un patrón común en React es que un componente devuelva múltiples elementos. Los fragmentos le permiten agrupar una lista de elementos secundarios sin agregar nodos adicionales al DOM.

```
class Table extends React.Component {
  render() {
    return (
      <table>
        <tr>
          <Columns />
        </tr>
      </table>
    );
  }
}
```

```
class Columns extends React.Component {
  render() {
    return (
      <div>
        <td>Hello</td>
        <td>World</td>
      </div>
    );
  }
}
```

Fragments

```
class Table extends React.Component {
  render() {
    return (
      <table>
        <tr>
          <Columns />
        </tr>
      </table>
    );
  }
}
```

```
class Columns extends React.Component {
  render() {
    return (
      <div>
        <td>Hello</td>
        <td>World</td>
      </div>
    );
  }
}
```

```
<table>
  <tr>
    <div>
      <td>Hello</td>
      <td>World</td>
    </div>
  </tr>
</table>
```

Fragments

```
class Table extends React.Component {
  render() {
    return (
      <table>
        <tr>
          <Columns />
        </tr>
      </table>
    );
  }
}
```

```
class Columns extends React.Component {
  render() {
    return (
      <React.Fragment>
        <td>Hello</td>
        <td>World</td>
      </React.Fragment>
    );
  }
}
```

```
class Columns extends React.Component {
  render() {
    return (
      <>
        <td>Hello</td>
        <td>World</td>
      </>
    );
  }
}
```

```
<table>
  <tr>
    <td>Hello</td>
    <td>World</td>
  </tr>
</table>
```

#>/<>
HACK A BOSS
<CODE YOUR TALENT>

Paso 2

Listas

@ferblancodosil

Listas

¿Cómo iteramos un array en Javascript?

Listas

¿Cómo iteramos un array en Javascript?

```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map((number) => number * 2);
console.log(doubled);
```

Listas

¿Cómo iteramos un array en Javascript?

```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map((number) => number * 2);
console.log(doubled);
```

Imagina devolver código HTML

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li>{number}</li>
);
```

Listas

¿Cómo iteramos un array en Javascript?

```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map((number) => number
console.log(doubled);
```

LLEVEMOSLO A UN COMPONENTE REACT

...ágina devolver código HTML

```
...ers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li>{number}</li>
);
```

HACK A BOSS
<CODE YOUR TALENT>

Listas

```
function NumberList(props) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
    <li>{number}</li>
  );
  return (
    <ul>{listItems}</ul>
  );
}

const numbers = [1, 2, 3, 4, 5];
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<NumberList numbers={numbers} />);
```

Listas

```
function NumberList(props) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
    <li>{number}</li>
  );
  return (
    <ul>{listItems}</ul>
  );
}

const numbers = [1, 2, 3, 4, 5];
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<NumberList numbers={numbers} />);
```

Este código nos dará un warning de React!

Listas

```
function NumberList(props) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
    <li key={number.toString()}>
      {number}
    </li>
  );
  return (
    <ul>{listItems}</ul>
  );
}

const numbers = [1, 2, 3, 4, 5];
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<NumberList numbers={numbers} />);
```

Listas

Cuando ejecute este código, recibirá una advertencia de que se debe proporcionar una clave para los elementos de la lista. Una "clave" es un atributo de cadena especial que debe incluir al crear listas de elementos.

Las “key” ayudan al virtual DOM de React a identificar qué elementos han cambiado, se agregaron o se eliminaron. Se deben dar claves a los elementos dentro de la matriz para darles una identidad estable.

- La mejor manera de elegir una clave es usar una cadena que identifique de forma única un elemento de la lista entre sus hermanos. La mayoría de las veces, usaría ID de sus datos como claves
- Cuando no tiene ID estables para los elementos representados, puede usar el índice de elementos como clave como último recurso:
- No se recomienda usar índices para claves si el orden de los elementos puede cambiar. Esto puede tener un impacto negativo en el rendimiento y puede causar problemas con el estado del componente.
- Las claves solo deben ser únicas entre hermanos

Listas: refactor para incrustar el map

```
function NumberList(props) {
  const numbers = props.numbers;
  const listItems = numbers.map((number) =>
    <li key={number.toString()}>
      {number}
    </li>
  );
  return (
    <ul>{listItems}</ul>
  );
}
```

```
function NumberList(props) {
  const numbers = props.numbers;
  return (
    <ul>
      {numbers.map((number) =>
        <ListItem key={number.toString()}
                  value={number} />
      )}
    </ul>
  );
}
```

#>/<>
HACK A BOSS
<CODE YOUR TALENT>

Paso 2

Renderizado condicional

#>/<>
HACK A BOSS
<CODE YOUR TALENT>

@ferblancodosil

React: renderizado condicional

```
function Mailbox(props) {
  const unreadMessages = props.unreadMessages;
  return (
    <div>
      <h1>Hello!</h1>
      {unreadMessages.length > 0 &&
        <h2>
          You have {unreadMessages.length} unread messages.
        </h2>
      }
    </div>
  );
}

const messages = ['React', 'Re: React', 'Re:Re: React'];

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Mailbox unreadMessages={messages} />);
```

Paso 2

Eventos

@ferblancodosil

React: Events

- SyntheticEvent
- En camelCase (onClick)
- Se pasan funciones, no texto

```html
<!-- HTML -->
<button onclick="activateLasers()">
  Activate Lasers
</button>
```

```jsx
//JSX
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

React: Events

- Se evalúa lo que esté entre llaves
- Se pasa el evento como parámetro

```
<button onClick={activateLasers}>
  Activate Lasers
</button>
//se va a ejecutar activateLasers(e)
//e va a ser el evento sintético click
```

Es decir: los "manejadores" de eventos son funciones que tienen una firma así:

```
function eventHandler(event){
    //something
}
//equivalente
const eventHandler = (event) =>
{
    //something
}
```

React: Events

- minimizar llamadas a addEventListener()
- invocar preventDefault() explícitamente

```
function ActionLink() {
  function handleClick(e) {
    e.preventDefault();
    console.log('The link was clicked.');
  }
  return (
    <a href="#" onClick={handleClick}>
      Click me
    </a>
  );
```

React: Events

- Cuidado con el binding de this

```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};
    // This binding is necessary to make `this` work in the callback
    this.handleClick = this.handleClick.bind(this);
  }
  handleClick() {
    this.setState(state => ({
      isToggleOn: !state.isToggleOn
    }));
  }
  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}
```

React: Events

- Cuidado con el binding de this

```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};
  }
  handleClick: () => { //experimental!!!
    this.setState(state => ({
      isToggleOn: !state.isToggleOn
    }));
  }
  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}
```

React: Events

- Cuidado con el binding de this

```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};
  }
  handleClick(){
    this.setState(state => ({
      isToggleOn: !state.isToggleOn
    }));
  }
  render() {
    return (
      <button onClick={(e) => this.handleClick(e)}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}
```

Eventos: parámetros

- Nueva función

```
<button onClick={(e) => this.deleteRow(id, e)}>Delete Row</button>
```

- bind this

```
<button onClick={this.deleteRow.bind(this, id)}>Delete Row</button>
```

Paso 2

Refs

@ferblancodosil

React: refs

Las referencias proporcionan una forma de acceder a los nodos DOM o a los elementos React creados en el método de representación.

- Guardan referencias a elementos del DOM
- Guardan referencias a variables
  - que persisten durante la vida del componente (como useState)
  - que **NO** notifican a React de sus cambios (a diferencia de useState)

React: refs

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.myRef = React.createRef();
  }
  render() {
    return <div ref={this.myRef} />;
  }
}
```

```
const node = this.myRef.current;
```

REACT

React: refs

```
class CustomTextInput extends React.Component {
  constructor(props) {
    super(props);

    this.textInput = null;

    this.setTextInputRef = element => {
      this.textInput = element;
    };

    this.focusTextInput = () => {
      // Focus the text input using the raw DOM API
      if (this.textInput) this.textInput.focus();
    };
  }

  componentDidMount() {
    // autofocus the input on mount
    this.focusTextInput();
  }

  render() {
    // Use the `ref` callback to store a reference to the text input DOM
    // element in an instance field (for example, this.textInput).
    return (
      <div>
        <input
          type="text"
          ref={this.setTextInputRef}
        />
        <input
          type="button"
          value="Focus the text input"
          onClick={this.focusTextInput}
        />
      </div>
    );
  }
}
```

React: refs

No abuses de las referencias
Su primera inclinación puede ser usar referencias para "hacer que las cosas sucedan" en su aplicación. Si este es el caso, tómese un momento y piense de manera más crítica sobre dónde debe ser propiedad del estado en la jerarquía de componentes.