

Que es ?

Wednesday, 14 October 2020

10:19 PM

React Es una biblioteca Para Crear interfaces de Usuario.

Un framework Es Estructo En la forma de acomodar El Código y react no
React es declarativo.

```
// index.js
<Layout>
  <Header />
  <Blogpost post={data} />
  <Footer />
</Layout>
```

Decimos que hacer Pero no Como hacerlo.
JSX Es Html dentro de mi Javascript.

N Se importa Como
Biblioteca.

React dom.render

Wednesday, 14 October 2020 3:53 PM

una forma de tener un aprendizaje más activo:

Ejemplo donde React.js es JavaScript

React es análogo a createElement

ReactDOM es análogo a appendChild

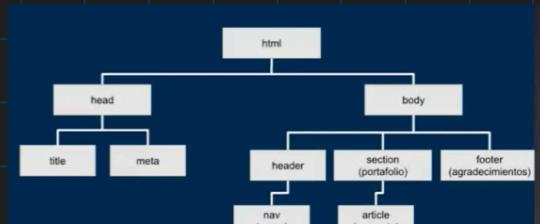
ReactDOM.render(_qué_, _dónde_);

Hecho por @jdreyespaez

ReactDom.render toma dos Elementos. Que, Dónde.

```
1 // const element = document.createElement('h1');
2 // element.innerText = 'Hello, Platzi Badges!';
3 // const container = document.getElementById('app');
4 // container.appendChild(element);
5
6 import React from 'react';
7 import ReactDOM from 'react-dom';
8
9 const element = <h1>Hello, Platzi Badges from React!</h1>
10
11 const container = document.getElementById('app');
12
13 // ReactDOM.render(_qué_, _dónde_);
14 ReactDOM.render(element, container)
```

“
Modelo de documento que se carga en el navegador web y que representa el documento como un árbol de nodos
”



Npx

Thursday, 15 October 2020

10:06 AM

La versión 5.2 de npm trae consigo npx, una herramienta que me parece que ha pasado algo desapercibida a pesar de estar a punto de cumplir un año y que creo que merece la pena conocer. Npx es una herramienta de cli que nos permite ejecutar paquetes de npm de forma mucho más sencilla. Vamos a ver varios ejemplos.

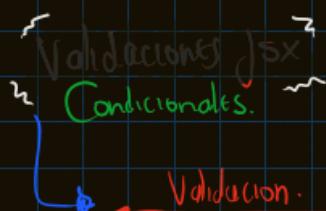
Si, por ejemplo, tenemos en nuestro proyecto ESLint, antes teníamos que buscar el binario en node_modules.

```
1 ./node_modules/.bin/eslint yourfile.js
```

Ahora npx lo buscará por nosotros.

```
1 npx eslint yourfile.js
```

Pero npx no solo busca los binarios en nuestro proyecto, también es capaz de ejecutar el paquete indicado sin tenerlo previamente instalado. Por ejemplo, `create-react-app` es un proyecto que no usamos muy a menudo y del que casi siempre queremos ejecutar la última versión.



JSX es una alternativa para React createElement

```
{videos myList.length > 0 &&
  <Categories title='My List'>
    <Carousel>
      <CarouselItem/>
    </Carousel>
  </Categories>
}
```

Condition => no hay videos
El componente no se muestra!

```
const App = () => {
  const [ videos, setVideos ] = useState([ ])
  useEffect(() => {
    fetch('http://localhost:3000/initialState')
      .then(response => response.json())
      .then(data => setVideos(data))
  }, [])
}
```

aplic & esta dividiendo la variable videos.
Con subvideos

Iterar en un Componente

```
<Categories title='Favorites'>
  <Carousel>
    {videos.trends.map(item =>
      <CarouselItem key={item.id} {...item} />
    )}
  </Carousel>
</Categories>
```

Con map itero en el Componente y se van a generar tantos Carousel como hayan en la API.

Destructuración

Props.

```
{videos.trends?.map(item =>
```



→ Condicionar que trends exista
→ If(videos.trends) ? map.

```
<Categories title='Tendencias'>
  <Carousel>
    {videos.trends ? undefined : videos.trends.map(item =>
      <CarouselItem key={item.id} {...item}></CarouselItem>
    )}
  </Carousel>
</Categories>
```

También se puede Condicionar en app.js

```
{error && (
```

```
  <p className="text-danger">{this.props.error.message}</p>
)
```

Siempre va a ser Verdadero
No es false.

→ Si error no es falsy retorna el componente.

También se puede usar el operador ternario dentro del return del render, por si alguna vez lo necesitan.

```
<button
  onClick={this.handleClick}
  className="btn btn-primary">Guardar</button>

{condicion
  ? <Elemento />
  : <OtroElemento />
}

</Form>
```

Webpack

Thursday, 15 October 2020

10:08 AM

Webpack es **la gestión de esos módulos** y de sus dependencias, pero también puede usarse para cuestiones como **concatenación de código, minimización y ofuscación**, verificación de **buenas prácticas (linting)**, **carga bajo demanda** de módulos, etc...

Una de las muchas cosas interesantes de Webpack es que **no solo el código JavaScript se considera un módulo**. Las hojas de estilo, las páginas HTML e incluso las imágenes se pueden utilizar también como tales, lo cual da un extra de potencia muy interesante.

Webpack es una herramienta de compilación (una *build tool* en la jerga) que coloca en un grafo de dependencias a **todos** los elementos que forman parte de tu proyecto de desarrollo: código JavaScript, HTML, CSS, plantillas, imágenes, fuentes... Esta idea central es la que lo convierte en una herramienta tan poderosa.

Webpack se puede considerar como un *Task Runner* muy especializado en el procesamiento de unos archivos de entrada para convertirlos en otros archivos de salida, para lo cual utiliza unos componentes que se denominan **loaders**.

'Componente' vs 'Elemento'

Un elemento es a un objeto como un componente es a una clase.

Si el elemento fuera una casa, el componente sería los planos para hacer esa casa.

Características

Elementos

NavBar → Componente

Ciclos de vida del componente

Wednesday, 14 October 2020 2:02 PM

ComponentDidMount() → todo Esto lo Reemplaza

ComponentWillUnmount() UseEffect.

Este Se activa Cuando el Usuario Cambia de Componente.

Render() → Método que renderiza el Componente.

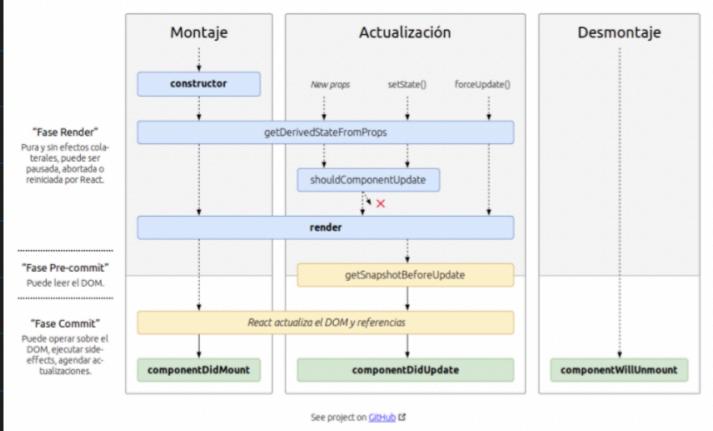
Los Hooks Propios Empiezan Por (use) y Son funciones.

Propio es añadir Dificultad.

Por que hooks?

- Era Difícil Compartir Logica de Estado entre Componentes. → Es opción usar Redux ó renderProps.
- Cada hook tiene logica Separada) una función Es mas facil que una clase
- Igualmente se Confundia Con this)

→ Usando Clases.



→ Usando Hooks.

Fase de Vida de los Componentes:

Montaje

- Representa el momento donde se inserta el código del componente en el DOM.
- Se llaman tres métodos: constructor, render, componentDidMount.

Actualización

- Ocurre cuando los props o el estado del componente cambia.
- Se llaman dos métodos: render, componentDidUpdate.

→ Recibe como argumento
↳ Props:
prevProps, prevState.

→ Cuando se monta llama 3 Métodos

Desmontaje

- Nos da la oportunidad de hacer limpieza de nuestro componente.
- Se llama un método: `componentWillUnmount`.

→ Momento para limpiar memoria.

3. Creación y diseño de componentes
Qué es y cómo funciona un componente en React.js

En esta clase aprenderás acerca del ciclo de vida de los componentes en React para crear aplicaciones dinámicas. Desde la importancia del montaje cuando los usuarios llegan por primera vez a nuestra aplicación, hasta la actualización y desaparición de los componentes.

Los componentes en React tienen vida; nacen, crecen y desaparecen .

El ciclo de vida de los componentes tiene 3 fases :

1. **El Montaje** es cuando los usuarios llegan a nuestra aplicación, cuando tienen su 1era interacción con él. Es cuando aparecen los componentes
2. **La Actualización**, cuando unos componentes cambian y otros no. los que si cambian decidimos que pasan por una actualización, cuando los componentes se actualizan se ejecuta el `render()`, generando el nuevo DOM, es cuando React manda una señal de confirmación `componentDidUpdate()` Es un buen momento para reaccionar a cambios. Como por ejemplo añadir una sugerencia a una bebida.
3. **Eliminación de los componentes**, al entrar a otra página, varios componentes que se veían ya no estarán en ella, React manda la señal `componentWillUnmount()`, seguido de la eliminación del código en el DOM.

Los componentes en React tienen vida; nacen, crecen y desaparecen. Por lo cual tienen tres fases El montaje, La actualización y la eliminación de los componentes.



```
constructor(props) {
  super(props);
  console.log('1. constructor()');

  this.state = {
    data: [
      {
        id: '2de30c42-9deb-40fc-a41f-05e62b5939a7',
        firstName: 'Freda',
        lastName: 'Grady',
      }
    ]
}

componentDidUpdate(prevProps, prevState) {
  console.log('5. componentDidUpdate()');
  console.log({
    prevProps: prevProps,
    prevState: prevState,
  });

  console.log({
    props: this.props,
    state: this.state,
  });
}
```

or Se inicializan los datos puro con
this

Puedo no usar el Constructor.

```
class BadgeEdit  
  extends Component {  
  
  state = {  
    loading: true,  
    error: null,  
    form: {  
      firstName: '',  
      lastName: '',  
      email: '',  
      jobTitle: '',  
      twitter: ''  
    }  
  }  
  
componentDidMount() {  
  this.fetchData()  
}  
}
```

Comparando los props en Component did update

```
(prevProps: {}, prevState: {}) =>
  history: {length: 8, action: "POP", location: {}, createRef: f, push: f, ...}
  location: {pathname: "/badges", search: "", hash: "", state: undefined, key: "..."}
  match: {path: "/badges", url: "/badges", isExact: true, params: {}}
  staticContext: undefined
  __proto__: Object
+prevState:
  data: []
  __proto__: Object
__proto__: Object

*(props: {}, state: {}) =>
  *props:
    history: {length: 8, action: "POP", location: {}, createRef: f, push: f, ...}
    location: {pathname: "/badges", search: "", hash: "", state: undefined, key: "..."}
    match: {path: "/badges", url: "/badges", isExact: true, params: {}}
    staticContext: undefined
    __proto__: Object
  +state:
    data: (3) [{}, {}, {}]
    __proto__: Object
  __proto__: state.data.t
```

Cambio el state porque se cargan datos de una API.

Class props

Thursday, 29 October 2020 7:50 PM

```
class Badge extends React.Component {  
  render() {  
    const {  
      firstName,  
      lastName,  
      avatarUrl,  
      jobTitle,  
      twitter  
    } = this.props;  
    return (  
      <div className="Badge">  
        <div className="Badge__header">  
          <img src={confLogo} alt="Logo de la  
conferencia" />  
        </div>  
  
        <div className="Badge__section-name">  
          <img  
            className="Badge__avatar"  
            alt="Avatar de la persona" />
```

→ así puedo desestructurar
el objeto Props.

Contar Página

Buenas prácticas

Sunday, 8 November 2020 11:52 AM

Hablando de buenas prácticas de programación también se debe considerar el orden de tu código. Un ejemplo de ésto es en los *imports*, a mí me gusta dividirlos en: *Dependencias*, *Componentes* y *Assets*. Así lo llevo a cabo:

```
import React, {Component} from 'react'
import {Link} from 'react-router-dom'

import api from '../api'
import BadgeList from '../components/BadgeList'
import Loading from '../components>Loading'
import MiniLoading from
'../components/MiniLoading'
import Error from '../components/Error'
import DeleteBadgeModal from
'../components/DeleteBadgeModal'

import './styles/Badges.css'
import logo from '../res/badge-header.svg'
```

Events & state

Saturday, 31 October 2020 8:52 PM

Manejo de eventos: onClick

Forma 1 para prevenir el envío: type="button"

```
handleClick = e => {
  console.log('Button was clicked')
}

<button
  type="button"
  onClick={this.handleClick}
  className="btn btn-primary">
  Save
</button>
```

Forma 2 para prevenir el envío: handleSubmit()

```
handleSubmit = e => {
  e.preventDefault()
  console.log('Form was submitted')
}

<form onSubmit={this.handleSubmit}>
  <div className="form-group">
    <label>First Name</label>
    <input
      className="form-control"
      onChange={this.handleChange}
      type="text"
      name="firstName"
    />
  </div>
  <button
    type="submit">

    <span>Submit</span>
  </button>
</form>
```

Hecho por @dreyespaez

Evento onChange

```
export class BadgeForm extends Component {

  handleChange = (e) => {
    console.log([value: e.target.value])
  }

  render() {
    return (
      <div>
        <h1>New Attendant</h1>

        <form>
          <div className="form-group">
            <label>First Name</label>
            <input onChange={this.handleChange} className='form-control' type="text" name='firstName'/>
          </div>
        </form>
      </div>
    )
  }
}
```

→ Para acceder al valor
e.target.value

e.target.name

↳ Asigno la función al evento
onChange

En los input Siempre se obtiene el evento
onChange → Es común y buena práctica nombrarlo
handleChange

```
class BadgeForm extends React.Component {
  handleChange = e => {
    console.log({
      name: e.target.name,
      value: e.target.value,
    });
  };
}

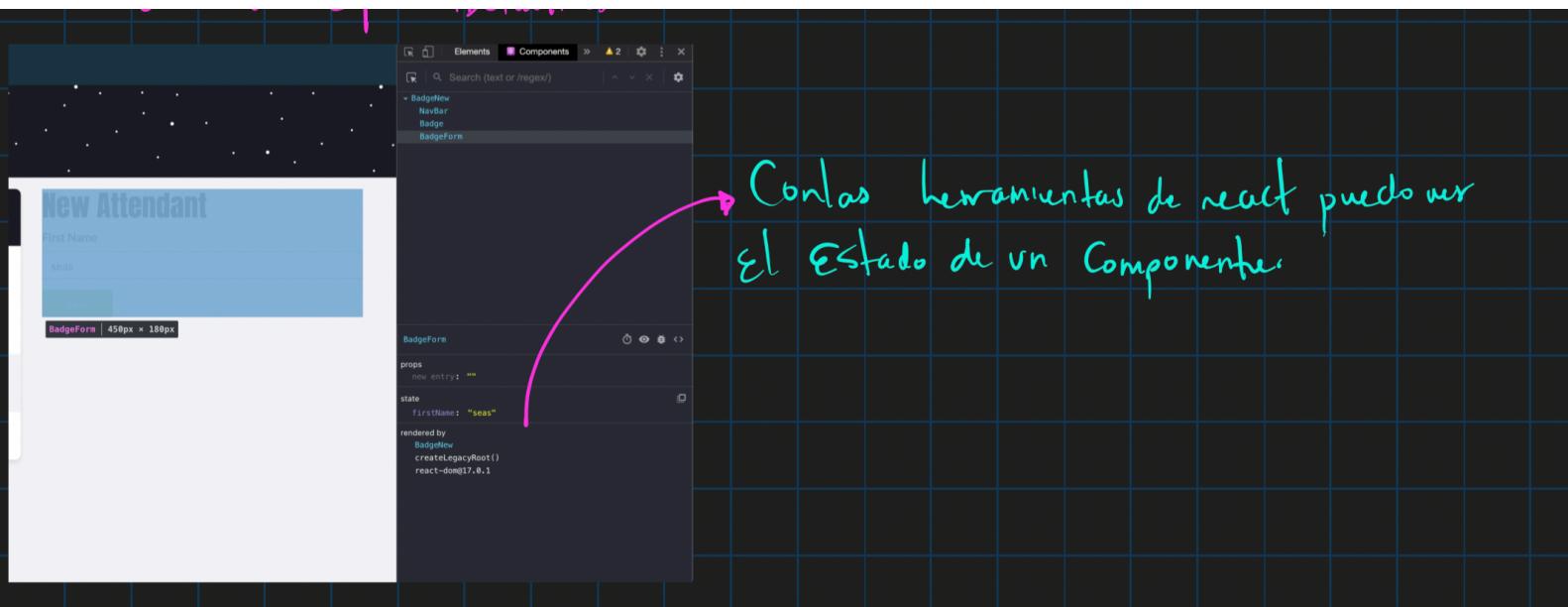
<button type="button" onClick={this.handleClick} className="btn btn-primary">Save</button>
```

→ Me da el name del input.

→ Value del input.

↳ Cuando hay un botón dentro de un form, éste se convierte de tipo submit.

↳ Para prevenir ese comportamiento se le pone al botón de type="button".
o Se usa e.preventDefault()



```
handleChange = ({ target }) => {
  // console.log(target.value)
  this.setState({
    [target.name]: target.value
  })
}
```

```
state
Email: "asd"
Twitter: "www"
firstName: "asd"
jobTitle: "fff"
lastName: "asd"
```

asi Creo una lista Con los atributos del formulario.

```
<form onSubmit={this.handleSubmit}>
  <div className="form-group">
    <label>First Name</label>
    <input onChange={this.handleChange}
      className='form-control'
      type="text"
      name='firstName'
      value={this.state.firstName}
    />
  </div>
```

Para Controlar los Campos de un formulario
Seteamos el value Para que Solo haya una fuente
de la verdad
Tenemos que definir el objeto Estado. ojo

PROPS - MATCH - HISTORY

Saturday, 7 November 2020 11:30 AM

match

```
<Route exact path='/teacher_profile/:id' component={ TeacherProfile } />
...
const teacherId = props.match.params.id
```

```
''
<Link to={`/teacher_profile/${id}`}>
<IconButton aria-label='see the profile'>
  <AddCircleIcon color='primary' />
</IconButton>
</Link>
```

así recibo el parámetro x la ruta.

lo saco con match.params.id

Así se lo mando a la wta.

History

```
try {  
  await api.badges.create(this.state.form);  
  this.setState({ loading: false });  
  
  this.props.history.push('/badges');
```

asi hago redirección a un sidebar
similar a (window.assign)
la web donde queremos
Redirigir.

User experience

Friday, 6 November 2020 5:13 PM

<https://www.npmjs.com/package/react-loading-skeleton>

react-loading-skeleton

2.1.1 • Public • Published 5 months ago

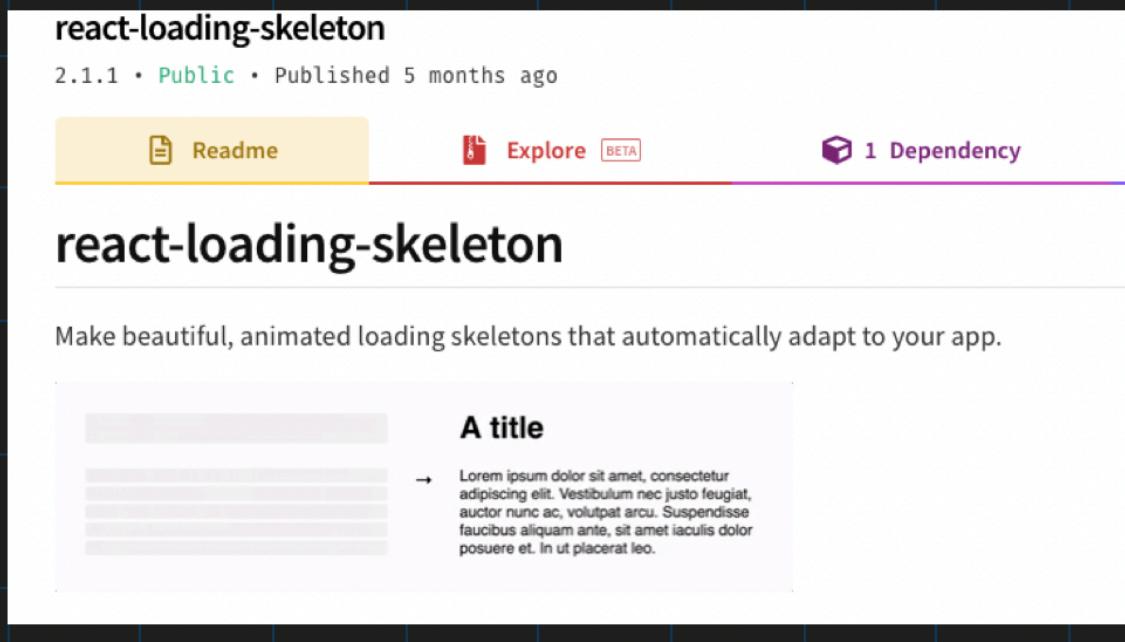
[Readme](#) [Explore BETA](#) [1 Dependency](#)

react-loading-skeleton

Make beautiful, animated loading skeletons that automatically adapt to your app.

A title

→ Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum nec justo feugiat, auctor nunc ac, volutpat arcu. Suspendisse faucibus aliquam ante, sit amet iaculis dolor posuere et. In ut placerat leo.



Gravatar

Friday, 6 November 2020 5:14 PM

<https://www.gravatar.com/avatar/eae35c458f8f5d9a1c6567db0d4f6b39?d=identicon>

Levantar estado

Saturday, 31 October 2020 11:39 PM

```
handleChange = e => {
  this.setState({
    form: [
      ...this.state.form,
      {e.target.name} : e.target.value
    ]
  })
}
```

Se agregan mas Datos al Estado para que no se sobre escriban

En los objetos no puede haber \equiv

```
state = { form: {} }
```

→ Es un objeto dentro de objeto
State.form → Para acceder

```
<div className="col-6">
  <BadgeForm onChange={this.handleChange} {...this.state.form}/>
</div>
```

Envia el objeto desestructurado.

```
render() {
  const { firstName, lastName, Twitter, jobTitle, Email } = this.props
```

→ Recibo el objeto desestructurado
Adentro de Rend Siempre

```
Warning: A component is changing an uncontrolled input of type text. This
component may be controlled. Input elements should not switch from uncontrolled to controlled
for user experience reasons. Instead, consider using a controlled or uncontrolled input element for
the lifetime of the component. More info: https://fb.me/react-controlled-components
in input (at BadgeForm.js:30)
in div (at BadgeForm.js:28)
in form (at BadgeForm.js:27)
in div (at BadgeForm.js:26)
in BadgeForm (at BadgeNew.js:42)
in div (at BadgeNew.js:41)
in div (at BadgeNew.js:30)
in div (at BadgeNew.js:29)
in div (at BadgeNew.js:23)
in BadgeNew (at src/index.js:18)
```

→ Para Solucionar Este warning
Se inicializan los valores del formulario.

```
state = { form: {
  firstName: '',
  lastName: '',
  firstName: '',
  firstName: '',
  firstName: ''
}; }
```

→ Solucion

Que es el virtual DOM

Wednesday, 14 October 2020

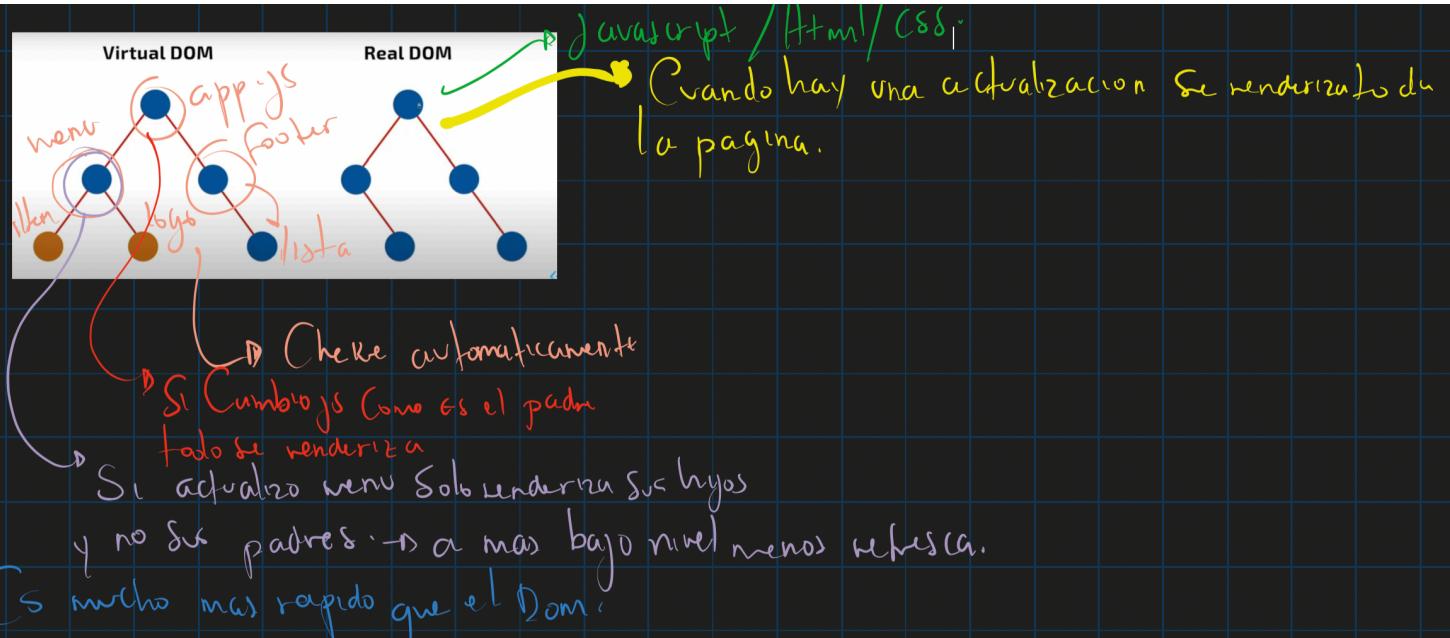
11:10 PM

Reconciliacion: Es el proceso de Comparar los Componentes Con el dom y Comparar Diferencias!

Renderizacion: Renderiza las diferencias que encontro } lo hace react or Server Side R.
En la reconciliacion y Pinta en el DOM. } DOM or react native

Virtual Dom Es la virtualizacion de lo que quiero renderizar .

Y Se actualiza automaticamente .



REACT

Public → Están los archivos públicos q' se van a producción

Src → la carpeta mas importante

↳ app.js → Componente visualizandose

Service Worker = Escucha si la aplicación Cambia o no

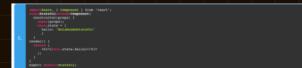


↳ import React {Component} from 'react'

Componente Presentacional

```
import React from 'react';
export class ComponentX extends React.Component {
  render() {
    return <h1>Hola Mundo!</h1>;
  }
}
```

Componente tipo clase



Tipos de Componentes

stateful y stateless

```
class Stateful extends Component {
  constructor(props) {
    super(props);
  }
  render() {
    return (
      <h1>Hola mundo!</h1>
    );
  }
}
export default Stateful;
```

↳ Stateful Component
↳ Constructor(props){}
↳ Super(props)
↳ this.state = {
 hello: 'hola'
}
↳ Objeto State

Componente stateless → sin estado.

→ no dependen de Estado ni Ciclo de Vida. Solo presenta logica

Código

```
import React from 'react';
const Stateless = () => {
  return (
    <h1>Hola mundo!</h1>
  );
}
export default Stateless;
```

Show Navigation

Redux

Wednesday, 7 October 2020

6:18 PM

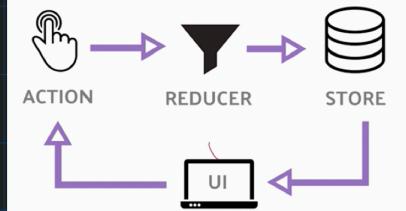
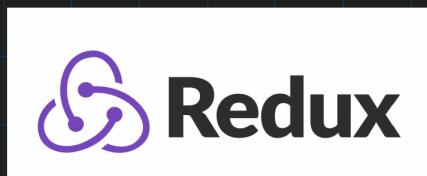
Es una librería escrita en Javascript
Basada en la arquitectura **flux**

Para manejar el flujo de la información de la aplicación.

Tiene 3 Principios fundamentales.

- ① Una sola fuente de la verdad.
- ② El estado es de solo lectura
- ③ Solo se pueden utilizar funciones Puras.

REDUX



Router!

Saturday, 3 October 2020 11:47 AM

9:30:09 > npm install react-router-dom --save

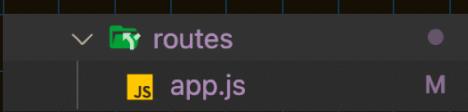
Instalacion
-- save Para guardarlo
Como dependencia.

BrowserRouter

↳ Encapsula todos los elementos

Para trabajar con la navegación.

↳ Se importa de react-router-dom



? Para manejar rutas Crea la
Carpeta routes y el archivo app.js.
Importo las dependencias.

```
import React from 'react'  
import { BrowserRouter, Switch, Route } from 'react-router-dom'  
import Home from '../container/Home'  
import Login from '../container/Login'  
import Register from '../container/Register'  
  
const App = () => (  
  <BrowserRouter>  
    <Switch>  
      <Route exact path='/' component={Home} />  
      <Route exact path='/login' component={Login} />  
      <Route exact path='/register' component={Register} />  
    </Switch>  
  </BrowserRouter>  
)  
  
export default App
```

index.js

```
1 import React from 'react'  
2 import ReactDOM from 'react-dom'  
3 import App from './routes/App'  
4  
5 ReactDOM.render(<App> , document.getElementById('app'))
```

Switch ya que info
de browser
Son los Conteniders de la
aplicacion

aquí se hace el ruteo de
importo el Componente segun
la ruta:

Switch solo permite que se
importe el primer Componente

ahora el index.js importa mi app
y lo muestra en el dom

AryRosvall Estudiante - elinfojapanda
La diferencia entre poner el switch y no ponerlo es que cuando tienes el mismo path para todos solamente toma el primero y lo renderiza.

404 Not found.

Para usar 404 se debe agregar la ruta así →

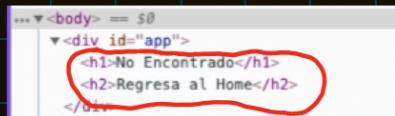
```
const App = () => (
  <BrowserRouter>
    <Switch>
      <Route exact path='/' component={Home} />
      <Route exact path='/login' component={Login} />
      <Route exact path='/register' component={Register} />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
)
```

→ no pongo ninguna ruta y el detecta que en cualquier ruta no definida va a renderizar el componente NotFound

fragments

```
2
3 const NotFound = () => (
4   <>
5     <h1> No Encontrado </h1>
6   </>
7 )
8 export default NotFound
```

Con el uso de fragments no agrego nodos innecesarios a mi aplicación, el dom lo renderiza sin nodos adicionales, así →



React Router ayuda a Crear Single Page Apps.

React Router (v4)

- Nos da las herramientas para poder hacer SPA fácilmente.
- Usaremos 4 componentes:
 - BrowserRouter
 - Route
 - Switch
 - Link

→ todo lo que este dentro de Comporta como SPA!

<Route />

Ejemplo: <Route path="/" component={Home} />

- Cuando hay un match con el path, se hace render del component.
- El component va a recibir tres props: match, history, location.

Proporcionan información sobre la dirección de internet.

`<Switch />`

- Dentro de Switch solamente van elementos de Route.
- Switch se asegura que solamente un Route se rendería.

`<Link />`

- Toma el lugar del elemento `<a>`.
- Evita que se recargue la página completamente.
- Actualiza la URL.

Link ⚡

Wednesday, 7 October 2020

4:34 PM

Navegación fluida de la aplicación.

```
<Link to="/">  
| <img className="header__img" src={logo} alt="Platzi Video" />  
</Link>
```

```
<a href="/">  
| <img className="header__img" src={logo} alt="Platzi Video" />  
</a>
```

Son lo mismo pero con la etiqueta link no se va a recargar toda la página. Con el <a> si se recarga toda la página, no da aspecto de SPA.

Layout

Wednesday, 7 October 2020

4:16 PM

→ planteja la Persistencia de los Componentes.

Para que la aplicación maneje el header y el footer cuando se cambie de página y sean persistentes.

Solo tener render del componente que se necesita.

```
import React from 'react'  
import Footer from './Footer'  
import Header from './Header'  
  
const Layout = ({children}) => (  
  <div className='App'>  
    <Header />  
    {children}  
    <Footer />  
  </div>  
)  
export default Layout
```

Recibe un children pero ya tiene el header y footer de la aplicación.

```
const App = () => (  
  <BrowserRouter>  
    <Layout>  
      <Switch>  
        <Route exact path='/' component={Home} />  
        <Route exact path='/login' component={Login} />  
        <Route exact path='/register' component={Register} />  
        <Route component={NotFound} />  
      </Switch>  
    </Layout>  
  </BrowserRouter>  
)
```

→ ahora el layout encierra Switch que es el rteador y header y footer
Son persistentes.

Hooks



Ofrece estado y Ciclo de Vida a los Componentes de tipo función o Stateless.

Se crearon x que la gente no entendía las claves y Dificultades Usar los Constructores

Era más complejo transmitir las Propiedades de los elementos y se formaban Cascadas.

Con react hook se pueden hacer llamados para manejar estados o Ciclo de Vida. Desde un Componente lejano. Sin transmitir información entre Componentes.

useState → Para manejar el Estado

useEffect → Para hacer Peticiones

○ Transferir Eventos entre Componentes

○ Escuchar Cambios.



const App = () => {
 return
 
 Explicado
)
 No está usando las llaves.

```
containers > App.jsx > ...
1 | import React, { useState, useEffect } from 'react';
```

Hooks

Permite a los componentes funcionales tener características que solo las clases tiene:

useState

Para manejo de estado

useEffect

Para suscribir el componente a su ciclo de vida.

useReducer

Ejecutar un efecto basado en una acción

useEffect

recibe dos Parámetros.
luego de Escuchar
una propiedad Cambie
se genera loop infinito.

Los hooks tienen que cumplir dos reglas → Custom Hooks

- ① Siempre deben Empezar x la palabra use
- ② Los hooks nunca se pueden usar En Condicionales.

Custom Hooks

Thursday, 1 October 2020 11:38 PM

Puedo Crear Custom Hooks para Separar la lógica de los Componentes a una función que se puede Usar en Cualquier Componente

```
const [ videos, setVideos ] = useState([]);
useEffect(() => {
  fetch('http://localhost:3000/initialState')
    .then(response => response.json())
    .then(data => setVideos(data));
}, []);
```

Este es un hook

↳ voy a agregar a una carpeta

en



useInitialstate.js

importo los estados.

la función va a manejar
el Estado y Ciclo de Vida
de los Componentes.

```
1 import {useState, useEffect} from 'react'
2
3 const useInitialState = () => {
4   const [ videos, setVideos ] = useState([])
5   useEffect(() => {
6     fetch('http://localhost:3000/initialState')
7       .then(response => response.json())
8       .then(data => setVideos(data))
9   }, [])
10  return videos
11 }
12
13 export default useInitialState
```

UseMemo

- Recibe una función y argumentos.
Cuando tenga los mismos argumentos ya sabe el resultado.

```
// Custom hook
function useSearchBadges(badges) {
  const [query, setQuery] = useState('')
  const [filteredBadges, setFilteredResults] = useState(badges)

  useMemo(() => {
    const result = badges.filter(badge => {
      return `${badge.firstName} ${badge.lastName}`
        .toLowerCase()
        .includes(query.toLowerCase())
    })
    setFilteredResults(result)
  }, [badges, query])
  return { query, setQuery, filteredBadges }
}
```

Hacer este cálculo sin usar memo
Se pude volver costoso.

Usar lowerCase para Normalizar

→ Use Memo recibe dos parámetros
que son los que van a volver a
Calcular el Resultado cuando
Esoz valores cambien.

Se usa el estado para setear filteredBadges

Modal o portales

Sunday, 8 November 2020 12:08 PM

7. Mejorando la UI (interfaz de usuario) Portales

Hay momentos en los que queremos renderizar una ventana (modal), un tooltip, etc. Esto puede volverse algo complicado ya sea por la presencia de un `z-index` o un `overflow hidden`. De otro componente ancestro

En estos casos lo ideal será renderizar en un nodo completamente aparte (en nuestra app el nodo principal es el de la app el `<div>` y allí renderizamos todo) y para renderizar fuera del nodo de app, React tiene una herramienta llamada Portales que funcionan parecido a `ReactDOM.render`; se les dice qué se desea renderizar y dónde, con la diferencia de que ese dónde puede ser fuera de la aplicación. (es decir fuera del nodo el `<div> app`)

En `BadgeDetails.js` al botón de eliminar le decimos cuando le hagamos click llame a un Modal, pero todavía no lo hacemos

- Importamos `import ReactDOM from 'react-dom'`; //esto para crear un portal
- Le decimos :
 - `{ReactDOM.createPortal(`
 `<h1>Hola, este es el portal</h1>,`
 `document.getElementById('modal')`
 `)}`
 - Tenemos que crear el nuevo `<div>` en el `index.html` en la carpeta `public`, esto para crear otro nodo donde podemos renderizar cosas | `<div id="modal"></div>`

Lo que hicimos fue renderizar desde un componente (`BadgeDetails`) que pertenece al nodo `<div id="app"></div>`, simplemente crear un portal que renderiza el `h1` desde otro nodo `<div id="modal"></div>`, lo logramos gracias a `{ReactDOM.createPortal(que, donde)}` recibe que vamos a renderizar y donde.(en este caso un `h1` en otro nodo)

PropTypes

Friday, 2 October 2020 12:08 AM

PropTypes ofrece una manera dinámica de verificar las propiedades que se le pasan a los Componentes. Para verificar el tipo de dato.

↳ npm install prop-types

→ Instalación

Se importan

```
import PropTypes from 'prop-types'
```

```
CarouselItem.propTypes = {
  cover: PropTypes.string,
  title: PropTypes.string,
  year: PropTypes.number,
  contentRating: PropTypes.string,
  duration: PropTypes.number
}
```

→ Se definen como se reciben los tipos de datos de las propiedades del componente.

```
Component.propTypes = {
  name: PropTypes.string.isRequired, // obligatorio
  lastName: PropTypes.string.isRequired, // obligatorio
  age: PropTypes.number, // opcional
  list: PropTypes.array, // opcional
};
```

API CALLS

Wednesday, 4 November 2020

7:30 PM

Introducción llamadas a un API

Las llamadas a una API siguen un patrón similar siempre que las hacemos, cada llamada consta de tres estados:

- **Loading:** cuando la petición se envía y estamos esperando.
- **Error:** se debe dejar un mensaje para el usuario para arreglar el error o volver a intentarlo.
- **Data:** los datos nos pueden llegar de dos formas, o en error o con los datos requeridos.

useState

Wednesday, 14 October 2020

2:28 PM

Permite almacenar el estado de un componente
React solo sabe dibujar en el DOM.

```
1
5 class App extends React.Component {
6   state = {
7     contador: 0,
8   }
9
10  incrementar = () => {
11    this.setState({ contador: this.state.contador + 1 })
12}
13
14  render() {
15    const { contador } = this.state
16    return (
17      <div className="App">
18        <header className="App-header">
19          <img src={logo} className="App-logo" alt="logo" />
20          <p>
21            {contador}
22          </p>
23          <button onClick={this.incrementar}>Incrementar</button>
24        </header>
25      </div>
26    );
27  }
28}
29
30 export default App;
```

```
> App.js > [o] default
1 | import React, {useState} from 'react';
2 | import logo from './Logo.svg';
3 | import './App.css';
4 |
5 | const App = () => {
6 |   const [counter , setCounter] = useState(0)
7 |   return (
8 |     <div>
9 |       La cuenta es: {counter}
10 |       <button onClick={() => {setCounter(counter + 1)}}>Add</button>
11 |       <button onClick={() => setCounter(counter - 1)}>Subtract</button>
12 |       <button onClick={() => setCounter(counter - counter)}>Reset</button>
13 |     </div>
14 |   )
15 | }
16
17 | export default App;
```

Ahora ya no es necesario escribir...

```
this.setState({  
    error: error  
})
```

Si las variables tienen el mismo nombre basta con escribir lo siguiente 😊

```
this.setState({  
    error  
})
```

Polling

Saturday, 7 November 2020

4:41 PM

6. Llamadas a un API Actualizaciones automáticas

Polling consiste en que cada cierto tiempo que es definido por nosotros se buscan los datos y se actualizan automáticamente. Esto se hará constantemente hasta que el usuario se vaya de la página.

Es en badges si hay un cambio se despliega solo aquí, vamos usar el **polling** cuando el componente se monta

- `setInterval(this.fetchData, 5000)` //1er parametro que va a ejecutar, 2do cada cuanto tiempo llamará a la función | esto lo que hacer es llamar a los datos cada 5segundos
- Cuando se monta el componente hace la primera llamada a la API GET y después vas a dejar un intervalo que Cada 5 segundos llama a `fetchData` coloca el `loading` true es decir actualiza el estado, hace la petición GET obtiene la lista de los badges y va a guardar esos datos en el estado vuelve a actualizar el estado

PERO CADA VEZ QUE PASAN LOS 5 SEGUNDOS SE MUESTRA EL LOADER Y IMPIDE AL USUARIO VER LO QUE ESTA EN PANTALLA

- Lo que vamos a hacer es que el `loading` de pagina completa sea solo al inicio, solo cuando en el estado no hay datos
- `if (this.state.loading === true && this.state.data === undefined) { renderiza el loader} o !this.state.data`

AÑADIMOS UN INDICADOR – lo añadimos debajo de la lista

- `{ this.state.loading && "Loading .." }` si en el estado loading es true muestra esto
- Vamos a copiar y a pegar el loader y vamos a hacer una variación llamada miniloader

AUNQUE LA PAGINA SE VAYA EL `setInterval` VAN A SEGUIR TRABAJANDO Y VA A CAUSAR CAMBIOS DE ESTADO EN UN COMPONENTE QUE YA NO ESTA

- `componentWillUnmount() { clearInterval(this.intervalId) } //justo antes que se desmonte el componente borramos el intervalo`

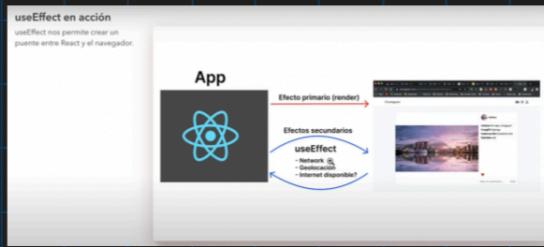
Lo que hicimos fue que cuando se monte el componente dejamos un `setInterval` que llama a `fetchData` cada 5segundos, es decir cada 5segundos hace la petición de tipo GET y guardamos esa info en el estado.

UseEffect

Wednesday, 14 October 2020

2:31 PM

UseEffect hace un Efecto Secundario Como una llamada HTTP. (hace algo distinto de solo pintar en el Dom)



→ te permite usar las apis del navegador

Todos los Hooks son funciones.

```
useEffect(() => {
  function onMouseMove(evento) {
    if (evento.clientX < window.innerWidth / 2) {
      setColor("orange");
    } else {
      setColor("blue");
    }
  }

  window.addEventListener("mousemove", onMouseMove);
  console.log("EJECUTANDO");

  return () => {
    console.log("LIMPIANDO");
    window.removeEventListener("mousemove", onMouseMove);
  };
});
```

Se ejecuta cuando el Componente Se desmonta.

↳ ComponentWillUnmount

```
1 import React, { useState, useEffect } from 'react'
2 import Loading from './Loading'
3
4 const ListProducts = () => {
5   const [isLoading, setIsLoading] = useState(true)
6
7   useEffect(() => {
8     setIsLoading(false)
9   })
10
11   return (
12     isLoading
13     ? <Loading />
14     : 'Mostrar listado'
15   )
16 }
17
18 export default ListProducts
19
```

```
useEffect(() => {
  console.log('only once time')
}, [ ])
```

Si le pongo el array solo se va a Ejecutar Una Sola Vez
o Cuando Cambie la Variable definida adentro.