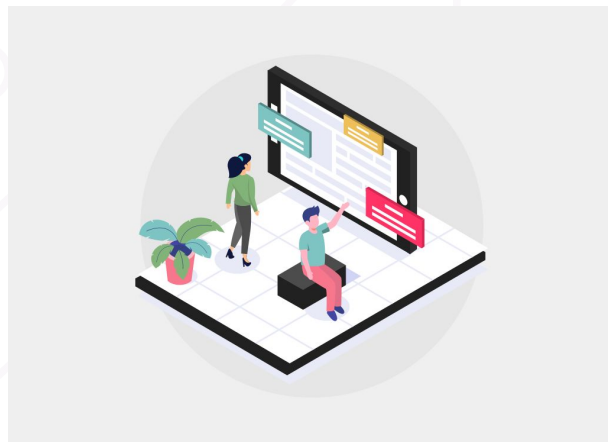


# SPA React JS

# ¿Qué son las Single-page application (SPA)? El desarrollo elegido por Gmail y LinkedIn



# ¿Qué es una Single-Page Application?

- 1) Una **Single-Page Application (SPA)** es un tipo de aplicación web que **ejecuta todo su contenido en una sola página**.
- 2) Funciona cargando el contenido **HTML**, **CSS** y **JavaScript** por completo al abrir la web.
- 3) Al ir pasando de una sección a otra, solo necesita cargar el contenido nuevo de forma dinámica si este lo requiere, pero no hace falta cargar la página por completo. **Esto mejora los tiempos de respuesta y agiliza mucho la navegación, favoreciendo así a la experiencia de usuario.**

# SPA (Single Page Application)

La velocidad de carga de una web es un factor determinante en la satisfacción del usuario e influye tanto en la tasa de conversión como en el abandono de la página.

# SPA (Single Page Application)

páginas o redes tan famosas como Gmail, Google Maps, LinkedIn, Trello o Twitter la carga entre vistas es extremadamente rápida. Esto se debe a que son **ejemplos de single-page application**.



juan.fulanez@gmail.com ▾

Gmail ▾



Más ▾

1-3 de 3



Es ▾



REDACTAR

Recibidos (3)

Destacados

Enviados

Borradores

Más ▾



Principal



Social



Promociones



Equipo de Gmail

Organízate mejor con la bandeja de entrada de Gmail - Hola, c

11:53



Equipo de Gmail

Lo mejor de Gmail estés donde estés - Hola, Juan: Descárgate la

11:53



Equipo de Gmail

Tres consejos para sacarle el máximo partido a Gmail - Hola, J

11:53

0 GB (0%) ocupados de 15 GB

[Administrar](#)

©2015 Google - [Condiciones](#) - [Privacidad](#)

Última actividad de la cuenta: hace 15 minutos

[Información detallada](#)



Juan ▾



No hay chats recientes

[Iniciar uno nuevo](#)



NavBar

Ramen

New York, NY

Q

For Businesses

Write a Review

Log In

Sign Up

Restaurants


Home Services

Auto Services

More

ResultList

ResultItem



1. Ramen Danbo

★★★★☆

246

\$\$

•

Ramen, Noodles

✓ Delivery

✓ Takeout


✓ Outdoor Seating

"I'm constantly on the hunt for a vegan **ramen** that will stand up to my absolute favorite spot in the city (RIP Shinobi). This place is the closest I've found! They actually surpassed my expectations with an EXTENSIVE vegan menu" more

Start Order

Offers takeout and delivery

ResultItem



2. Naruto Ramen

★★★★☆

252

\$\$

•

Ramen

✓ Delivery

✓ Takeout

✗ Outdoor Seating


"Literally the best **ramen** in the uws period and they follow order instructions! Definitely my go to spot" more

Start Order

Offers takeout and delivery

Map

Search as map moves



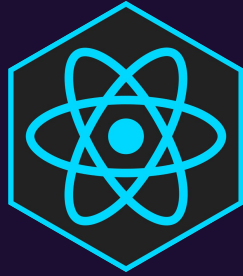
DEV.F

## ¿Cuándo conviene desarrollar una SPA?

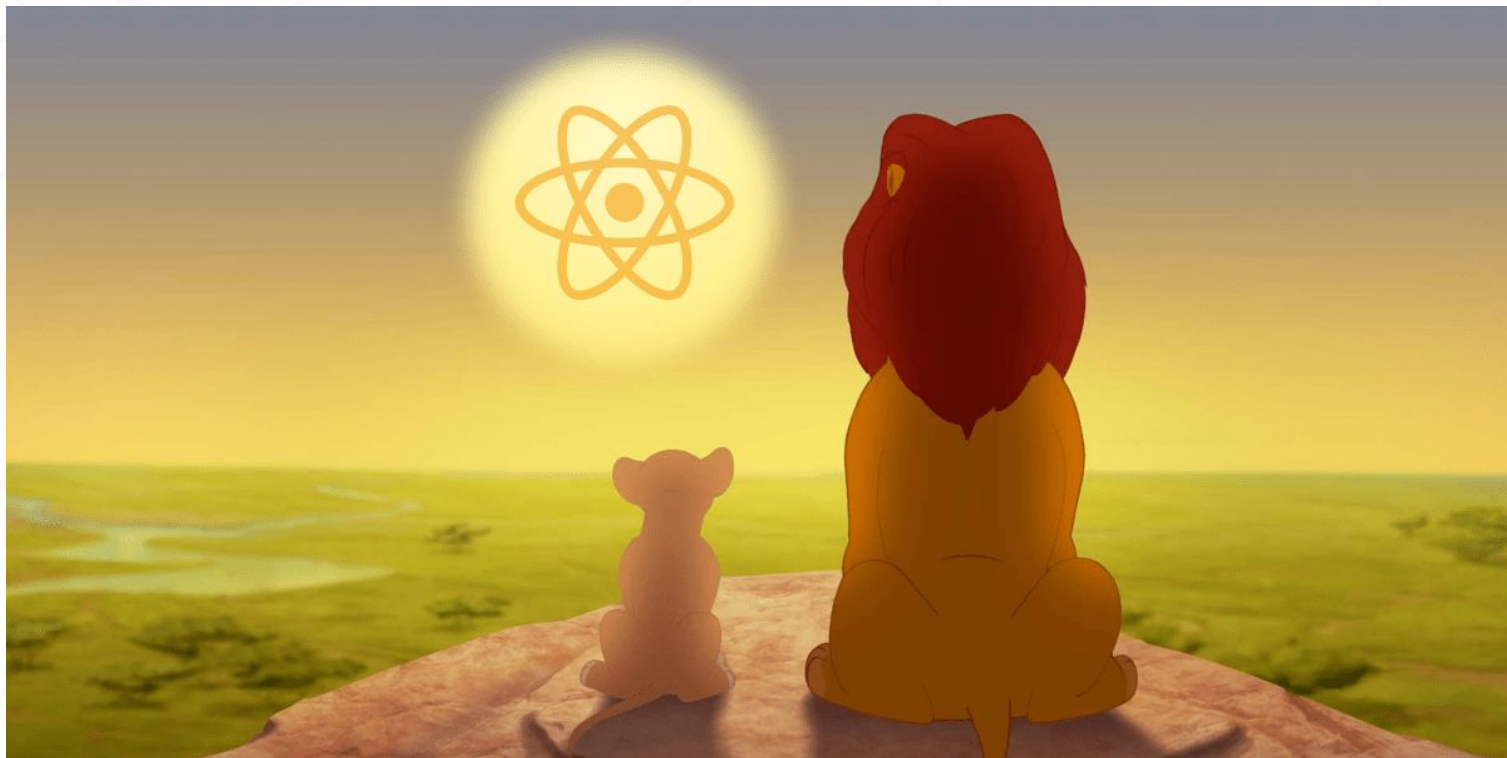
Para pymes o startups que buscan un sitio funcional y bien diseñado para **iniciar su presencia online**. Sin embargo, si el sitio es más completo, la decisión se complica un poco más. Cuando hay muchas secciones diferenciadas y una **jerarquía más compleja**, limitarlas a simples vistas puede resultar contraproducente.







# Ciclos de Vida



¡Hablemos del círculo vital del componente React!

# Ciclos de vida

En [React.js](#) los componentes que no sean puros (*todos los que se crean mediante clases o `React.createClass`*) poseen algo conocido como el **ciclo de vida**. Este ciclo de vida son una serie de funciones que se ejecutan en distintos momentos de la vida del componente y nos permiten realizar distintas acciones en estos momentos.

En las aplicaciones de React, **utilizamos componentes** para dividir y aislar diferentes partes de la interfaz de usuario web en partes individuales.



Estas piezas actúan de forma independiente y utilizan una función de renderizado para devolver elementos React en JSX. Estos elementos describen cómo debe mostrarse esa sección al usuario.



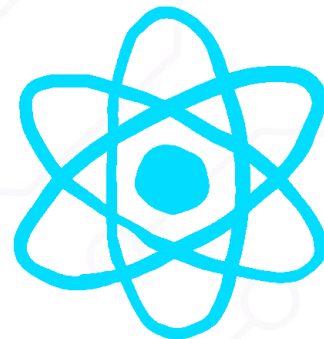
# Fases del ciclo de vida

Cada componente pasa por **tres** fases: **montaje(Mounting)** , **actualización(Updation)** y **desmontaje(Unmounting)** . También puede pensar en ello como nuestro ciclo de vida natural: nacemos, crecemos (adolescencia y edad adulta) y, finalmente, morimos.

**Tip:** Cada método tiene un prefijo **will** o **did** dependiendo de si ocurren antes o después de cierta acción.

# Fases del ciclo de vida

Los componentes de React **se crean al montarlos en el DOM**, cambian o crecen a través de actualizaciones y, finalmente, se pueden eliminar o desmontar del DOM. Estos tres hitos se conocen como el **ciclo de vida del componente React**.



**Tip:** Cada método tiene un prefijo **will** o **did** dependiendo de si ocurren antes o después de cierta acción.

# Life cycle

## Initialization

setup props and state

## Mounting

componentWillMount

render

componentDidMount

## Updation

### props

componentWillReceiveProps

shouldComponentUpdate

true

false

componentWillUpdate

render

componentDidUpdate

### states

shouldComponentUpdate

true

false

componentWillUpdate

render

componentDidUpdate

## Unmounting

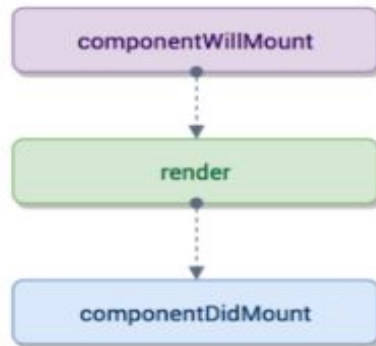
componentWillUnmount



# Fase de Montaje o Montado

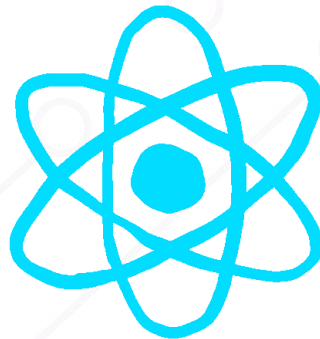
La primera fase ocurre solo una vez por componente cuando este se crea y monta en la UI. Esta fase se divide en 3 funciones.

## Mounting



# Constructor( )

Antes del inicio de la fase de montaje, es posible que necesitemos inicializar nuestro componente usando un `constructor( )` método. Esto se usa cuando necesitamos inicializar el estado y vincular métodos a nuestro componente. Este es el único lugar donde `this.state` se asigna explícitamente.





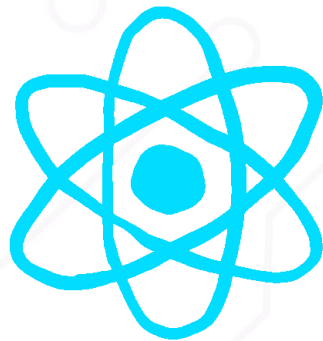
Nuevamente, suceden dos cosas importantes en este paso:

1. Se asigna un objeto a `this.state`
2. Los métodos se pasan / enlazan a la instancia de la clase  
a través de accesorios

# Render ( )

El `render()` método es el único método que es el componente *requerido* tener. Siempre se llamará y su trabajo es montar el componente en el DOM. El `render()` nos devuelve:

- **Elementos de reacción** : Nuestros JSX, estos son el objeto simple que describe lo que queremos que vea el usuario.



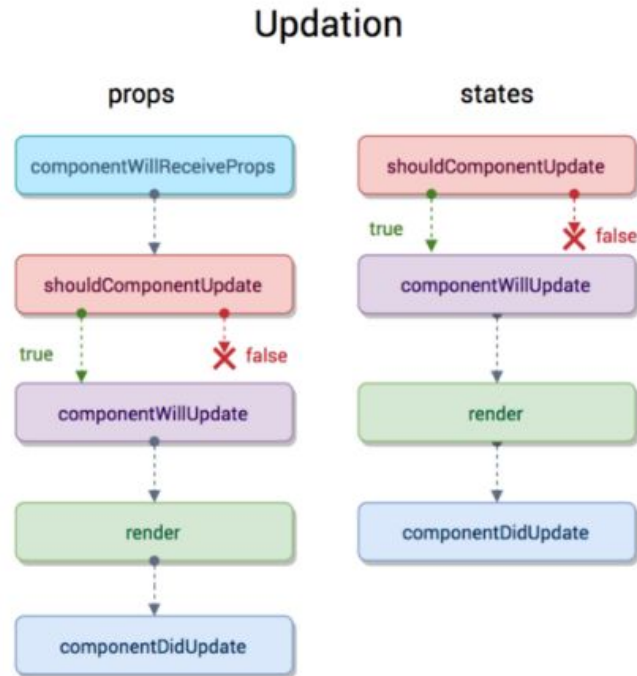
# componentDidMount( )

La última función en esta fase es `componentDidMount()`. Este método se invocará inmediatamente después de que se ejecute nuestro renderizado.

Podemos realizar una llamada a la API y actualizar el estado de los componentes en función de su respuesta. Podemos completar el contenido a partir de los datos que cargamos desde otro punto final. La llamada `setState()` debe usarse aquí,

# Actualización

Esa fase puede ocurrir múltiples veces (o incluso ninguna), sucede cuando algún dato del componente (**ya sea una propiedad, un estado o el contexto**) se modifica y por lo tanto requiere que la UI se vuelva a generar para representar ese cambio de datos.



# Desmontado

Esta última fase consiste en un solo método que se ejecuta antes de que un componente se **elimine (desmonte)** de la UI de nuestra aplicación.

## Unmounting

`componentWillUnmount`



# componentWillUnmount( )

Este método se ejecuta justo antes de desmontar el componente del DOM.

Puede pensar en este método como una forma de limpiar todo lo que sea necesario eliminar antes de que se destruya el componente.

# Conclusión

Estas son básicamente todas las fases del ciclo de vida con sus métodos y que hacen. Esa funcionalidad es una de las que hacen más poderoso a nuestros componentes de React ya que nos permite tener un control total sobre **qué ocurre en nuestra aplicación en todo momento. HAPPY HACKING!**



# React