Intro a React

María Victoria Rubio Hijosa

String_


¡NO ENTIENDO NADA DOC! EL OTRO DÍA ME EXPLICARON LOS HOOKS PARA REACT, ¡¡¡PERO SI NI SIQUIERA SÉ LO QUE ES REACT!!!
NO TE PREOCUPES MARTY, VAMOS AL DELOREAN Y SOLUCIONEMOS ESTO


MONTH
DAY
YEAR
HOUR
MIN
APR
05
2019
AM
PM
11
00
DESTINATION TIME
MONTH
DAY
YEAR
HOUR
MIN
APR
26
2019
AM
PM
01
50
PRESENT TIME
MONTH
DAY
YEAR
HOUR
MIN
OCT
26
1985
AM
PM
01
20
LAST TIME DEPARTED


REACT
TO THE FUTURE

Empecemos por el principio...

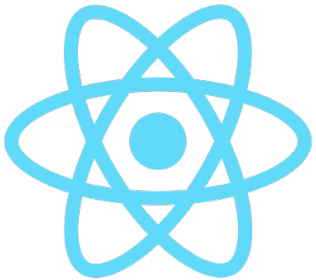
Una página web se compone de HTML, CSS y JavaScript.
Dentro de una página web tradicional podemos navegar, interactuar con ella, enviar y recibir datos...
Cada una de estas interacciones requiere una redirección del navegador.

Single Page Applications (SPA)

Son aplicaciones web que se cargan en una sola página para dar una experiencia más fluida a los usuarios. Cuando interactuamos con ellas no necesitan recargarse enteras ni navegar a otras páginas, sólo se actualizan las partes con las que interactuamos.

Este código de sincronización (manejo del DOM) puede complicarse mucho y ser muy propenso a errores.

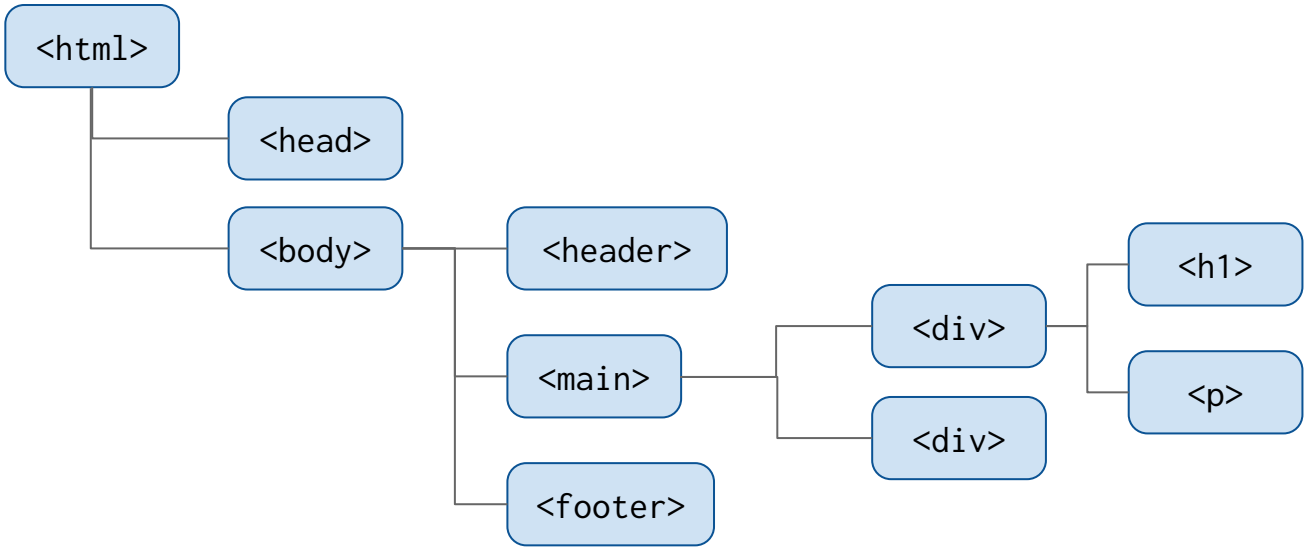
Por esta razón surgen los Frameworks JS



Su misión es hacer esta sincronización del DOM por nosotros y evitarnos muchos problemas. A cambio, vamos a tener que trabajar de una forma determinada para aprovechar las ventajas que los frameworks nos dan.

¿Pero qué es el DOM Doc?

Es un esquema virtual generado por el navegador al leer el contenido del archivo HTML (con su correspondiente CSS y JavaScript).
De esta forma representa los elementos de nuestra página como si fuese un árbol de objetos (con sus propiedades y métodos).
Cada uno de estos objetos será un nodo de ese esquema.



Entonces, ¿usamos React para manipular el DOM?

- No exactamente, Marty.
  Un framework o librería JavaScript como React nos soluciona uno de los principales problemas de la programación front-end: mantener la interfaz de usuario (UI) en sincronización con el estado nuestra aplicación.

- **Pero, ¿qué es el estado de una aplicación web?**

- Son los datos que necesitamos en cada momento para representarlos en el navegador.
  Una aplicación web se encarga de gestionar datos desde una interfaz.
  Los frameworks nos facilitan sincronizar el estado (los datos) con la interfaz (lo que se ve en la pantalla).

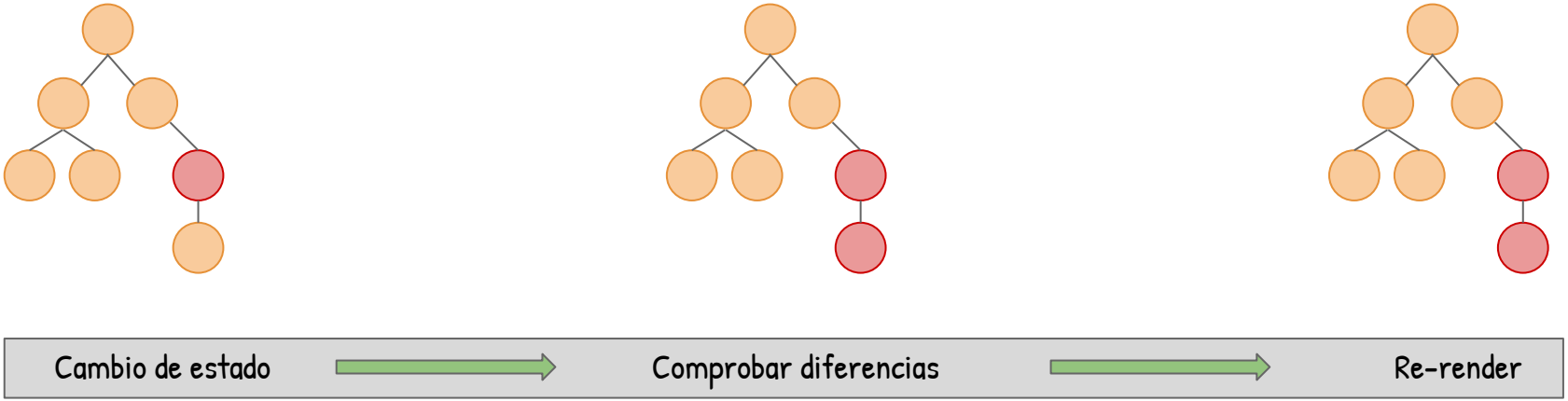
Virtual DOM

Para poder acceder al DOM del navegador, React usa el Virtual DOM que crear una copia en memoria del DOM del navegador.

Y entonces es cuando React hace su magia...

Cuando cambia el estado de alguno de los componentes, React actualiza primero el Virtual DOM y mediante un algoritmo, compara los cambios con el DOM del navegador y actualiza únicamente los nodos que han sufrido algún cambio.
De esta forma se agiliza mucho el proceso de renderizado, no teniendo que volver a cargar todo el contenido cada vez que se hace un cambio.

Virtual DOM



Browser DOM

Las ventajas de usar React

- Permite escribir vista y comportamiento juntos, en lo que llamamos componentes, que serán reutilizables.

- Estos componentes dan estructura a nuestra web y se pintan en el DOM a través del Virtual DOM como hemos visto anteriormente para que no tengamos que manejarlos directamente.

- Se pueden crear webs muy reactivas y rápidas ya que está pensado para que los componentes cambien ya sea a través del estado como de las props.

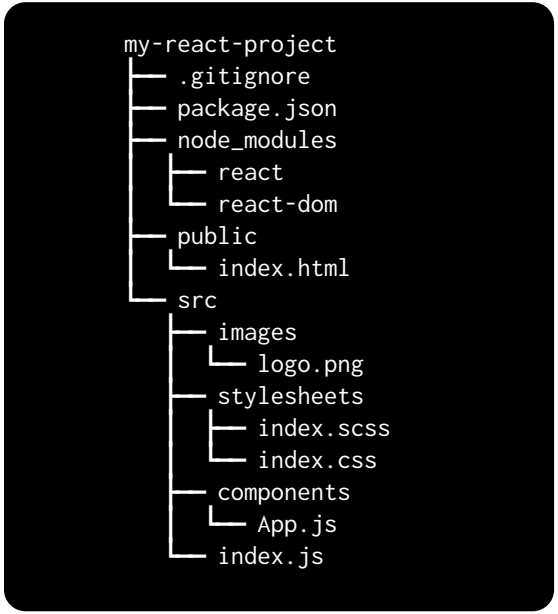
- Es una forma muy intuitiva de hacer webs porque todo son componentes que llaman a otros componentes. El flujo es unidireccional (de arriba abajo), así que es fácil entender y solucionar los errores que pueda haber: si el error no está en mi componente, está en quien ha llamado a mi componente y cómo.

Parece interesante Doc pero, entonces ¿cómo uso React?

Muy fácil Marty, hay varias formas de iniciar un proyecto react, pero la más sencilla es, teniendo node instalado introducir los siguientes comandos en nuestra terminal:

```
1.  npm install -g create-react-app
2.  create-react-app my-react-project
3.  cd my-react-project
4.  npm start
```

Esto nos crea una estructura de carpetas con la que podemos empezar a trabajar.



Ficheros Principales

public/index.html : Es el único fichero HTML que usaremos en nuestra aplicación. En el body se carga un `div` con id `root` que será donde se carga la aplicación de React.

src/index.js : Este será el fichero JS de entrada a nuestra aplicación React. Será el encargado de importar y pintar el componente principal de la aplicación, en este caso, llamado `App`.

src/App.js : Este fichero será nuestro primer componente React.

Conceptos básicos

Props : Son datos que pasamos a los componentes de React de la misma forma en la que pasamos atributos a los elementos del DOM. Podemos acceder a ellas a través de `this.props`, que es un objeto que contiene las claves y los valores que hemos consignado al declararlas.

State : El estado de un componente se define como una representación del mismo en un momento concreto, es decir, una instantánea del propio componente. Por defecto, el estado de un componente está vacío.

Ciclos de Vida : El ciclo de vida es una serie de momentos, definidos por métodos, por los cuales pasan los componentes con estado desde que se crea dicho componente hasta que se elimina.
Dentro de estos métodos el más importante es el `render` que se encarga de renderizar en el navegador el HTML correspondiente al componente.

Creando componentes

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

class App extends Component {
 render() {
   return (
     <h1>Hola, mundo</h1>
   );
 }
}

export default App;
```

```
import React from 'react';

function App (props) {
 return (
   <h1>Hola, mundo</h1>
 )
}

export default App;
```

Usando Props

```
import React, { Component } from 'react';
class Welcome extends Component {
 constructor(props) {
   super(props);
   this.state = {}
 }
 render() {
   return (
     <div>
       <h1>Hola, {this.props.name}</h1>
     </div>
   );
 }
}
export default Welcome;
```

```
import React, { Component } from 'react';
import Welcome from './Welcome';

class App extends Component {
 constructor(props) {
   super(props);
   this.state = {}
 }
 render() {
   return (
     <Welcome name="Mariví" />
   );
 }
}
export default App;
```

Cambiando el estado

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

class App extends Component {
 constructor(props) {
   super(props);
   this.state = {
     name: 'Mariví'
   }
 }

 _changeName() {
   this.setState({
     name: 'Juan'
   })
 }

 render() {
   return (
     <div>
       <h1>Hola, {this.state.name}</h1>
       <button onClick={() => this._changeName()}>Change</button>
     </div>
   );
 }
}

export default App;
```

Ciclos de Vida

"Render Phase"

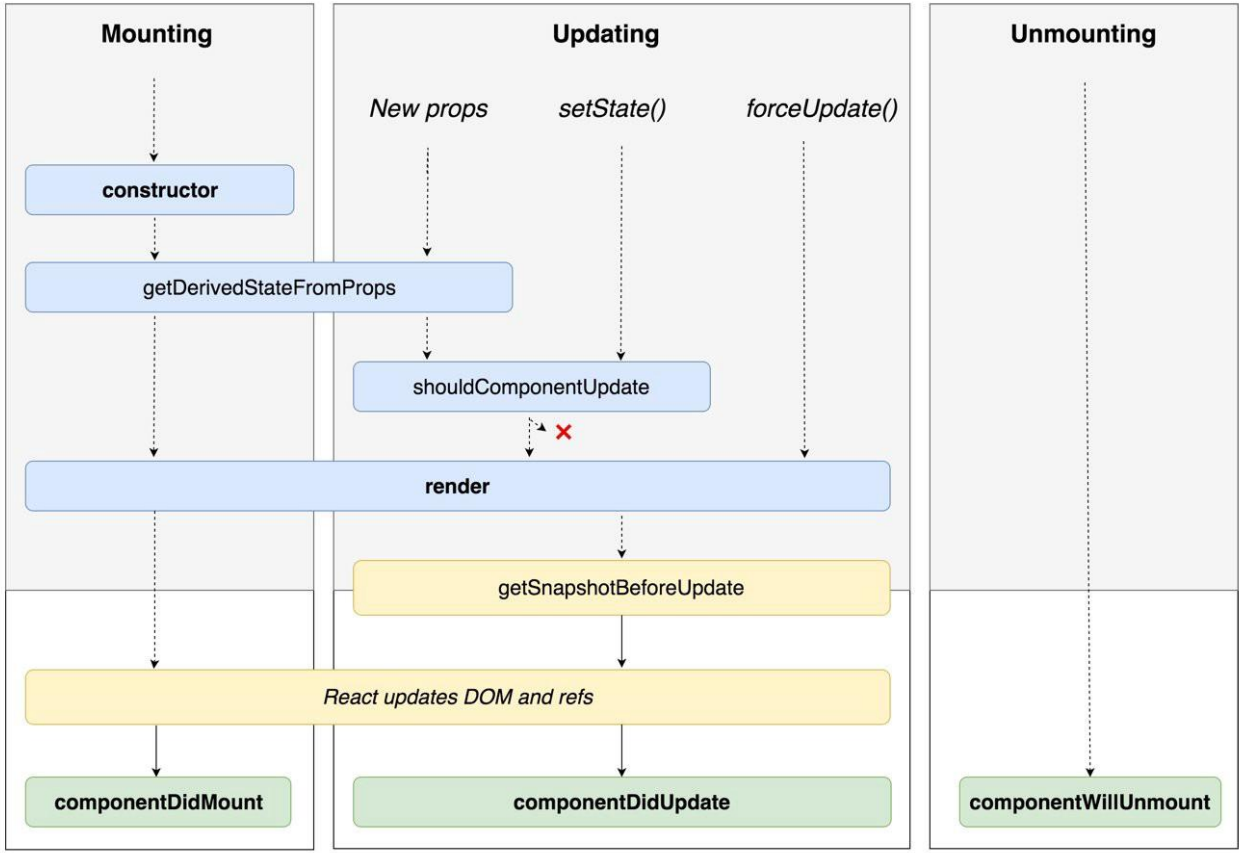
Pure and has no side effects. May be paused, aborted or restarted by React.

"Pre-Commit Phase"

Can read the DOM.

"Commit Phase"

Can work with DOM, run side effects, schedule updates.



Vale Doc, creo que lo he entendido todo, volvamos al futuro..., digo al presente...


¡OH NO! EL CONTROL DE TIEMPO DEL DELOREAN HA SUFRIDO UN CORTOCIRCUITO, ¡¡NO PODREMOS VOLVER!!
TRANQUILO MARTY, HE ESTADO TRABAJANDO EN UNA APLICACIÓN WEB QUE NOS PUEDE SACAR DE AQUÍ, VAMOS A PROBARLA


LET'S GO!