Mapa conceptual de React

**1. Introducción a React**

**1.1. ¿Qué es React?**

**1.2. Historia y evolución de React**

**2. Principios fundamentales de React**

**2.1. Componentes**

**2.2. Estado y propiedades**

**2.3. Renderizado condicional**

**3. Conceptos avanzados de React**

**3.1. Ciclo de vida de un componente**

**3.2. Hooks**

**3.3. Context API**

**4. Manejo de eventos en React**

**5. Integración de React con otras tecnologías**

**5.1. React y Redux**

**5.2. React y GraphQL**

**6. Optimización y rendimiento en React**

**7. Testing en React**

**8. Aplicaciones prácticas de React**

**8.1. Desarrollo web**

**8.2. Desarrollo de aplicaciones móviles con React Native**

**9. Conclusiones y perspectivas futuras**

Mapa conceptual de React

# 1. Introducción a React

Por defecto si creamos un “HTML” normal de los de toda la vida, al hacer clic sobre él, llamar a un método “OnClick()” del elemento “Button”, nos cargaríamos TODO el “HTML” haciendo por tanto una lenta y costosa re-estructuración desde 0. Por otro lado, cualquier “Component” que sea el encargado de realizar la modificación del “DOM” virtual, tendrá que comparar su estructura con el “DOM” virtual actual (lo que será rápido, ya que el Virtual-DOM es una “varios miles de veces más rápido”, aproximadamente 1.300 veces más rápido, que el “Real-DOM”) para construir así el “New-DOM” o Nueva Versión del Árbol de Elementos cómo tendría que quedarse finalmente.

Vamos a ver cada concepto básico de React con el siguiente Mapa conceptual en el que va a quedar todo muy claro; dicho por algunos de nuestros alumnos les parece “impresionante y alucinante cuando se da uno cuenta de cómo funciona React internamente y el potencial que tiene”, traté de evitar que esto fuera demasiado impresionante. Universo JS me rompí el cerebro para tratar de simplificarlo lo máximo posible y que no fuera impresionante… (o por lo menos menos impresionante). Mapa conceptual React-Virtual-Dom. Virtual-DOM. Todo comienza aquí, cada vez que Modificamos el Estado de un “Component”, este es Say Hello to React JS Actualizaremos el Virtual-DOM Post con este sencillo Ejemplo de código todos y cada uno de los “Component” que corresponda realizarán alguna modificación solo en el Virtual-DOM.

## 1.1. ¿Qué es React?

React permite reutilizar el código, cada parte de la interfaz gráfica se compone de componentes. Estos componentes pueden tener hijos anidados mas componentes que a su vez podrán tener mas hijos en forma de arbol. Un padre no debe enviar información a los hijos directamente, el padre puede enviar a los hijos una función que ellos ejecutaran enviando la información a sus propios hijos. Del mismo modo un padre puede tener hijos con nodos de información que pueden ser editables, en ese caso los hijos pueden enviar información al padre y este la reparte entre los demás hijos. A una aplicación se le llama SPA cuando solo se recarga cuando el usuario navega entre paginas, el resto de llamadas de los usuarios se resuelven llamando a un servidor de datos que devuelve la información a React. No te equivoques de enfoque, si quiero guardar una venta de productos en una base de datos, se la pido a un servidor y este te dice si se ha guardado o no, no lo asesores desde React.

React es una biblioteca que está orientada a la creación de interfaces para el usuario y fue creado por los ingenieros de Facebook, quieren facilitar el trabajo con datos que cambian en el tiempo. React se encarga de renderizar la vista para nosotros y nos proporciona una serie de herramientas para hacer mas fácil el paso de estados entre las diferentes vistas. Recibir datos de un servidor, gestionar el login de los usuarios y en definitiva todas las cosas que necesiten del servidor de datos para funcionar no son parte de React. React trabaja en el entorno del navegador (frontend). Otras herramientas deben de trabajar de trasfondo grabando en bases de datos, enviando y recogiendo información necesaria del servidor para hacer posible el trabajo de React.