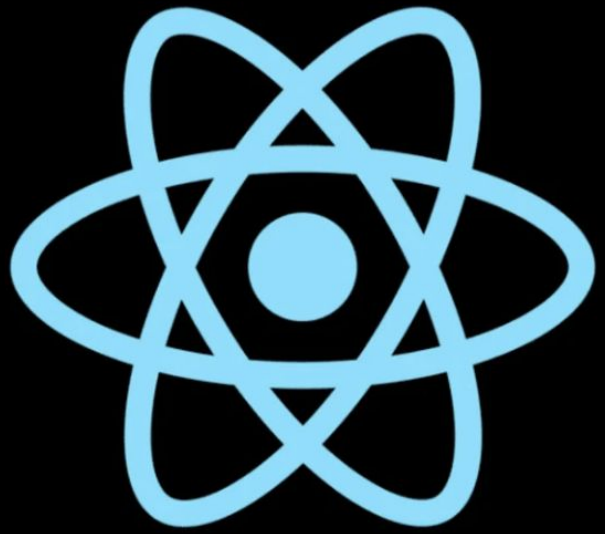
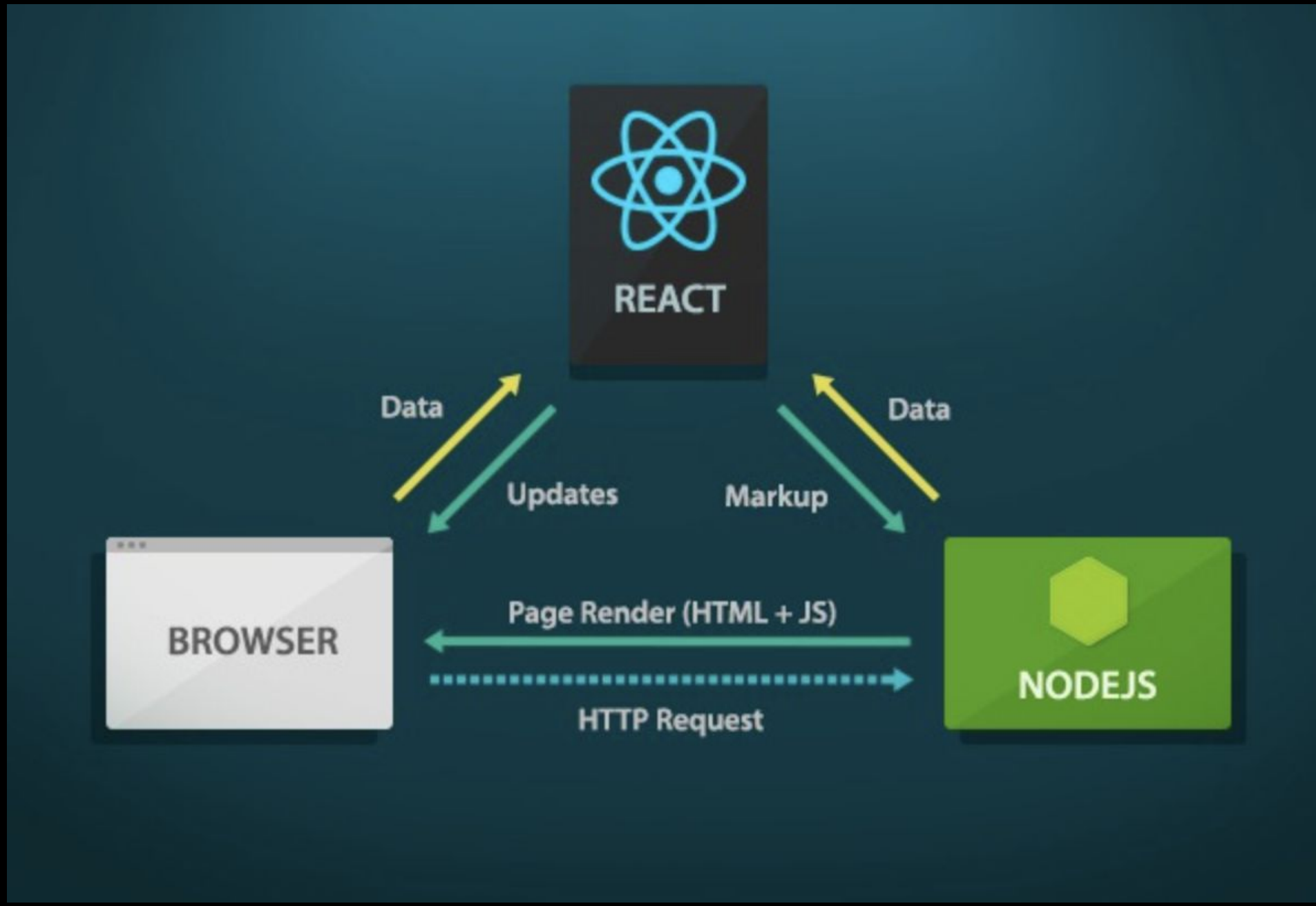
React



React JS


REACT
Data
Updates
Markup
Data
BROWSER
Page Render (HTML + JS)
HTTP Request
NODEJS

1000 likes — 100 talking about this — 10 were here

Company
Aliquam lectus orci, adipiscing et, sodales ac, feugiat non, lacus. Ut dictum velit nec est. Quisque posuere, purus sit

About

Photos | App 1 | App 2

Highlights

Your awesome company x1254 shared a link
55 minutes ago

Like – Comment – Share 55

79 people like this.

View all 15 comments

User Awesome Lorem ipsum dolor sit amet
May 17 at 1:26pm . I like

User Awesome Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis.
May 18 at 1:50pm . I like

30 friends like your company

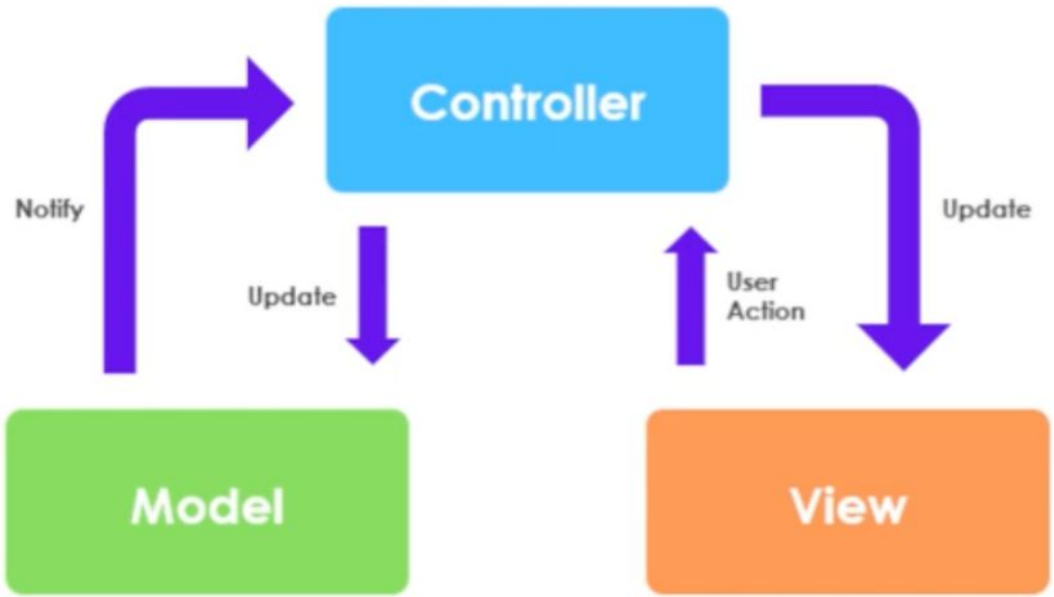
+21

Some great activity happenin' here
April 30

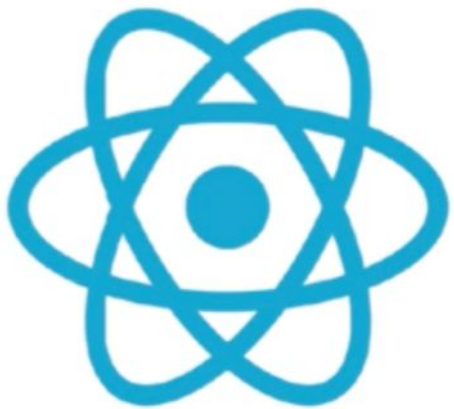
Like – Comment – Share 16 13 5

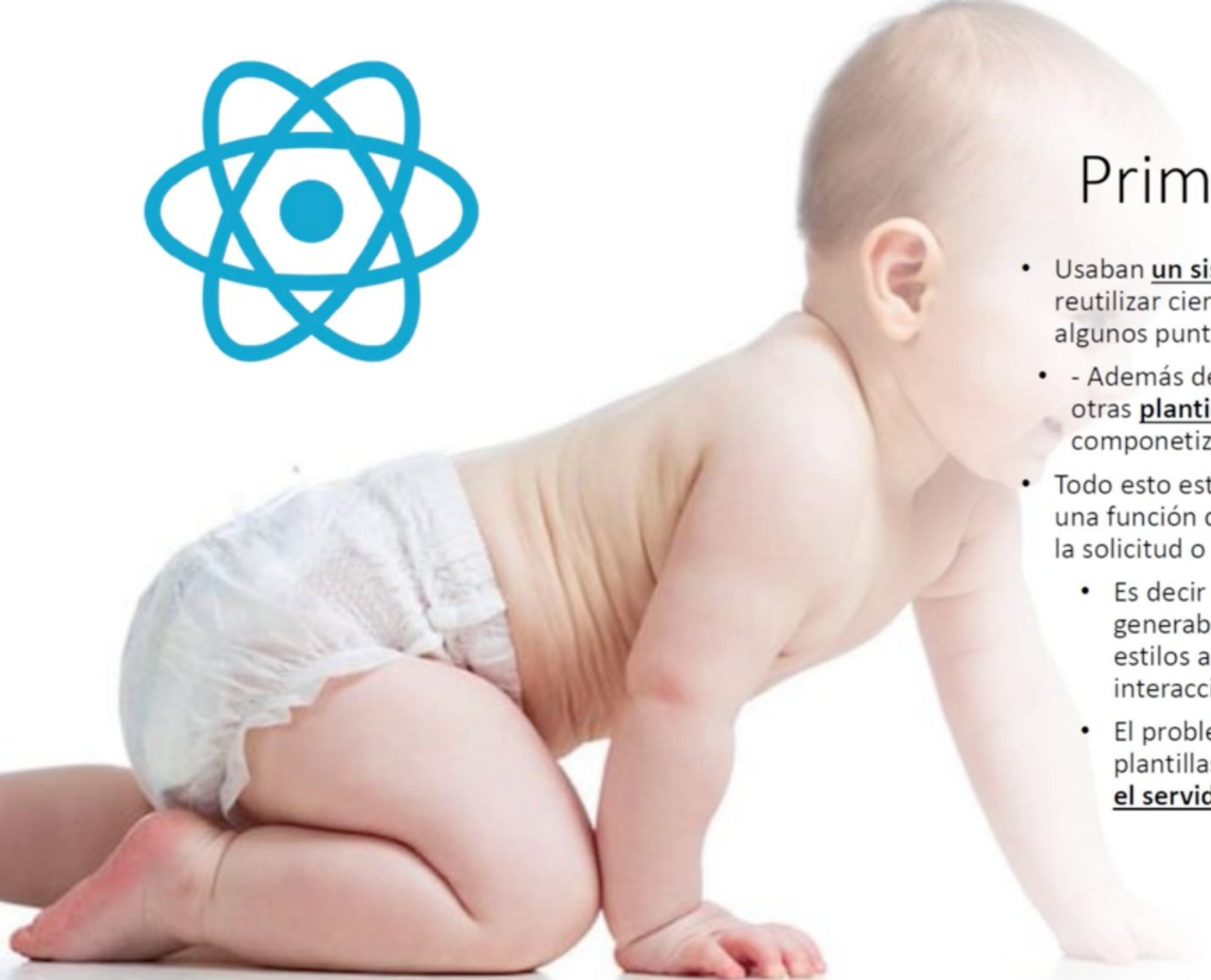
HISTORIA DE REACT

En el año 2011 los desarrolladores de Facebook tenía problemas con la arquitectura de sus aplicaciones → patrón MVC.



En esta arquitectura MVC el usuario interactúa con la aplicación a través de la **vista**, las acciones del usuario se gestionan en el **controlador** y este aplica los cambios necesarios sobre el **modelo. E**l **modelo** le notifica a la **vista** los nuevos cambios y le presenta al usuario dichos cambios. Se puede decir que cada acción del usuario provocaba una llamada al servidor.





Primeros pasos

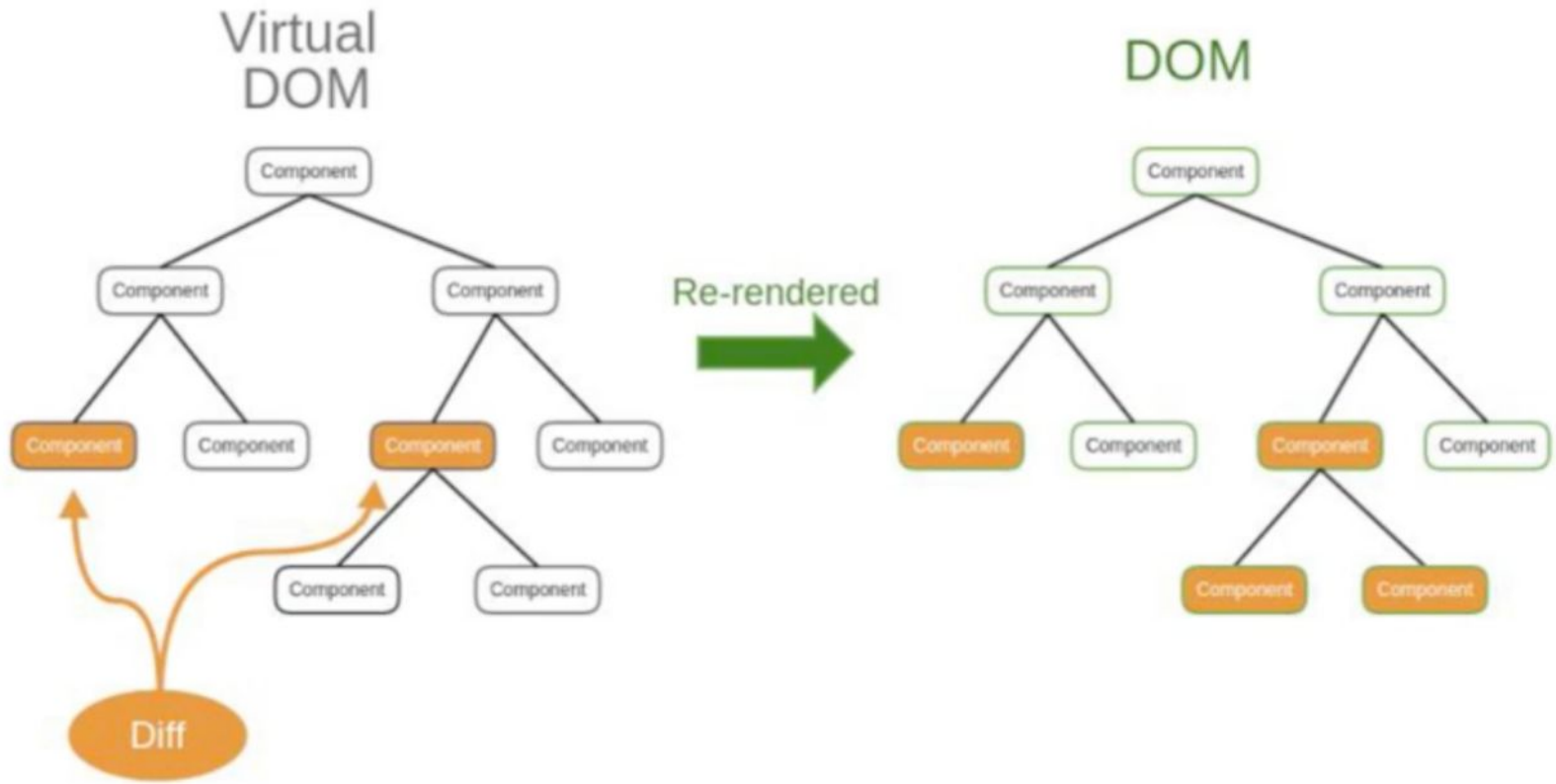
- Usaban **un sistema de plantillas** para poder reutilizar cierta funcionalidad repetida en algunos puntos de su aplicación.
  - - Además dentro de esta plantillas teníamos otras **plantillas más pequeñas** que la componetizaban.
- Todo esto estaba basado en un fichero php con una función que disparaba el contenido según la solicitud o información recibida.
  - Es decir a través de sus parámetros generaba un fichero HTML con unos estilos asociados y un JS para la interacción.
  - El problema era que este motor de plantillas era PHP y PHP **solo funciona en el servidor.**

Origen de React y JSX

- vieron que el volumen de peticiones al servidor no era escalable y fue entonces cuando decidieron que era el momento de dar entrada a nuestro querido Javascript
- Javascript introdujeron que toda la lógica de renderizado de template
- Dejaron de depender de PHP
- FaxJS o lo que conocemos hoy en día ReactJS:
  - Permitía la escritura de HTML con Javascript para la creación de pequeños templates dando vida a lo que conocemos como JSX → Lógica + vista en un mismo fichero

Imperativo vs Declarativo

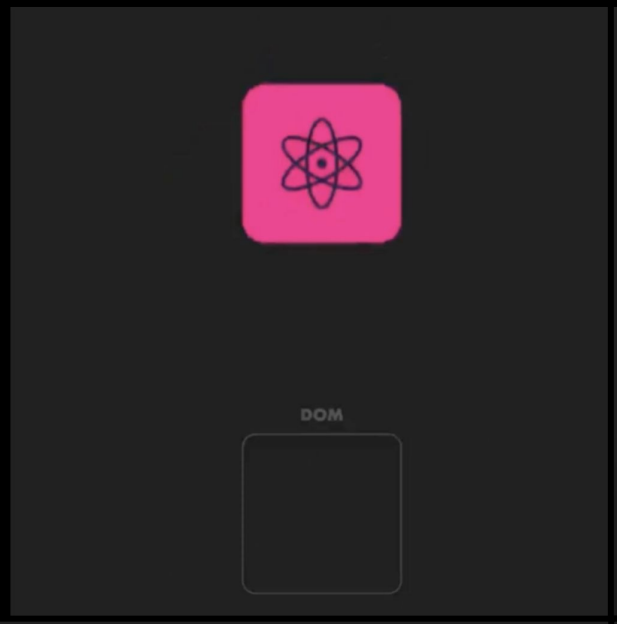
- React tiene un enfoque declarativo, es decir trabajaremos con estados, side effects … pero no tendremos que preocuparnos de cómo traer estos cambios al DOM.

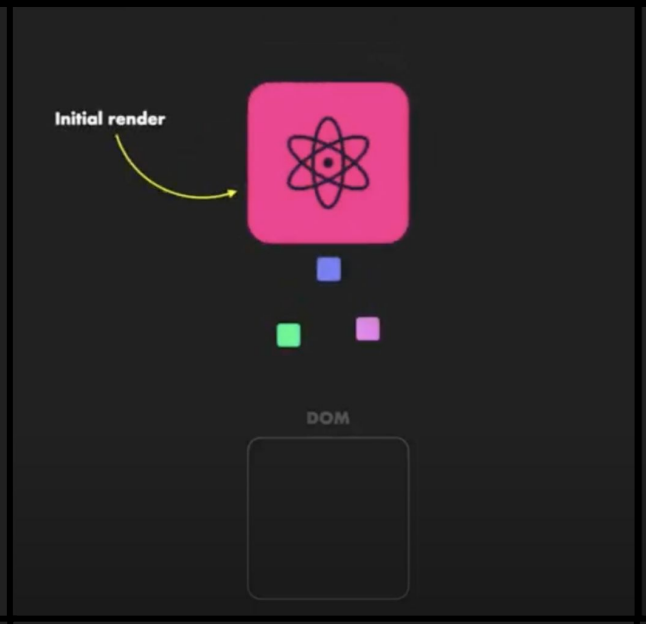


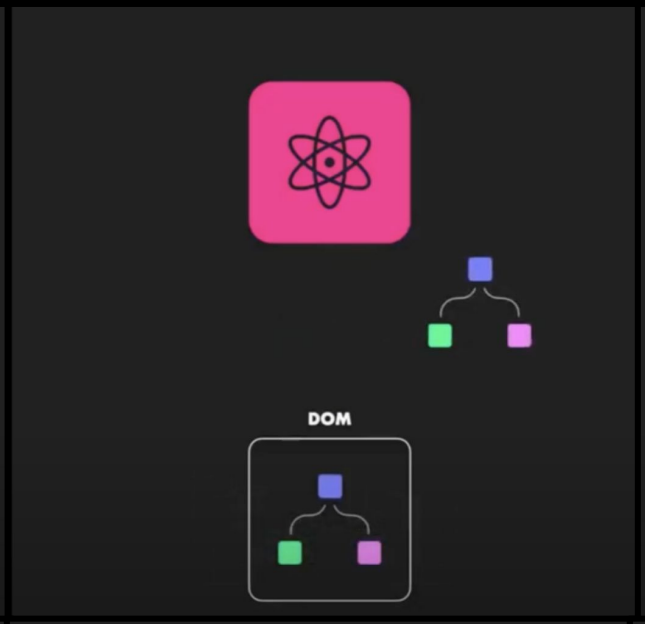
Cambios en cascada

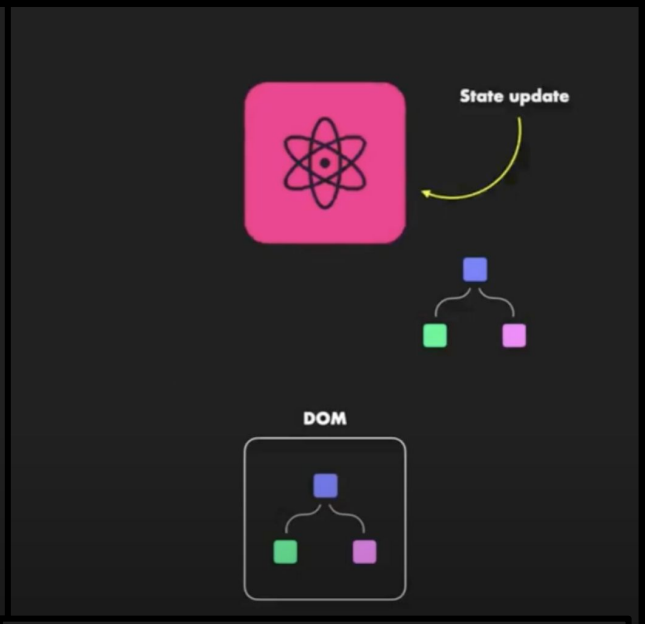
El ***Virtual DOM* es una representación en memoria del *DOM*** *de Javascript* que actúa de intermediario entre el estado de la aplicación y el DOM de la interfaz gráfica que está viendo el usuario

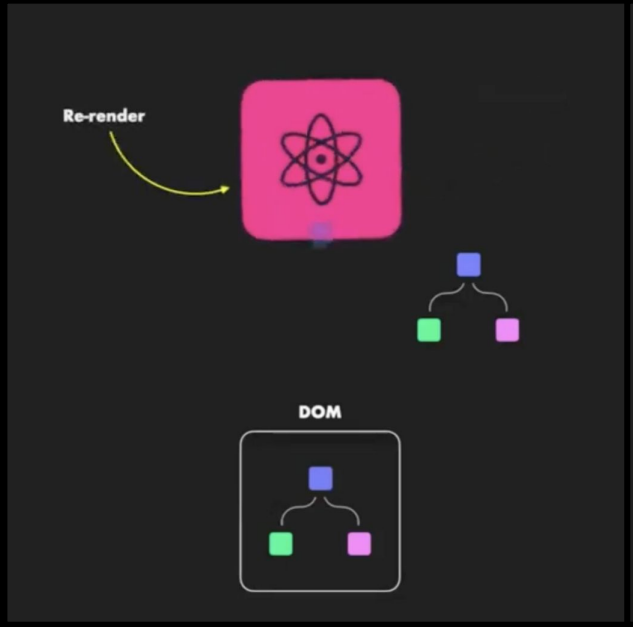
Las siglas **DOM** significan **Document Object Model**, o lo que es lo mismo, la estructura del documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina **árbol DOM** (*o simplemente DOM*).

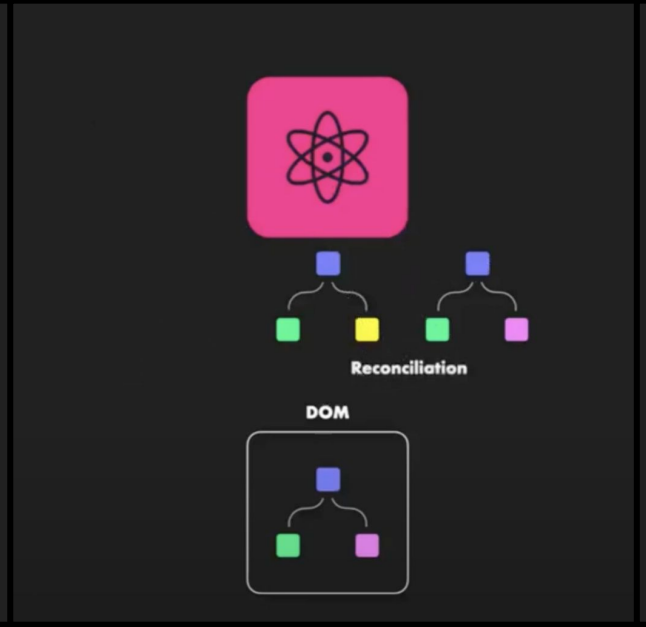

DOM

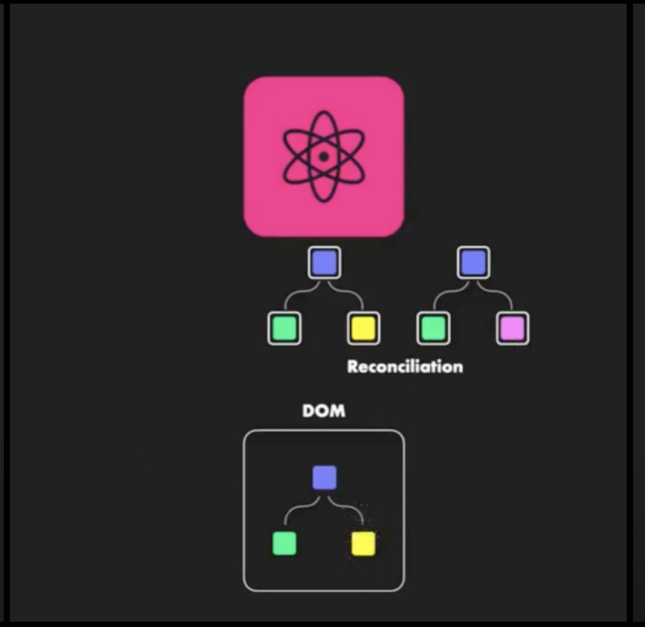

Initial render
DOM

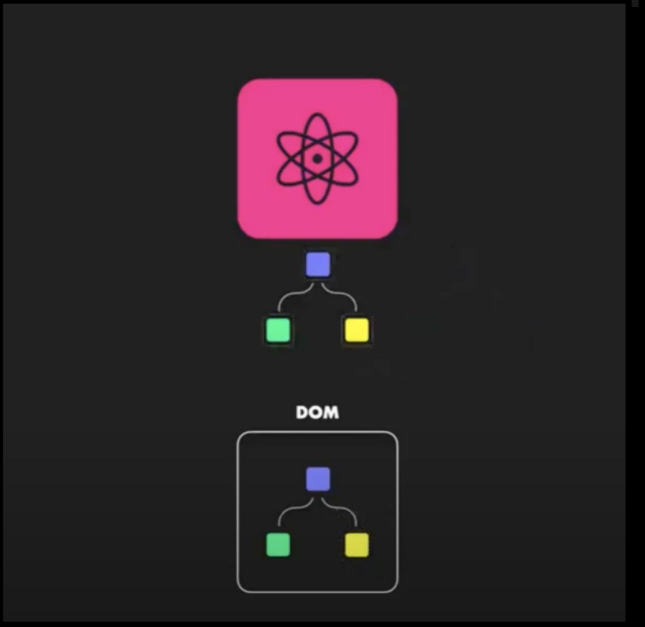

DOM


State update
DOM


Re-render
DOM


Reconciliation
DOM


Reconciliation
DOM


DOM

Que es React JS

Es una librería de Javascript para crear interfaces de usuario, como has oído React es una librería y no un framework dejando la libertad de usarla de la manera que mejor se adapte a nuestras necesidades. Para ello vamos a aclarar la diferencia entre Librería y Framework.

1.Control de flujo:

1. **Biblioteca**: Cuando utilizas una biblioteca, tú, como desarrollador, **estás al mando**. Tú controlas cuándo y cómo se llama a la biblioteca.
2. **Framework**: En cambio, cuando utilizas un framework, el **framework tiene el control**. Te proporciona una estructura y una plantilla para tu aplicación y "llama" a tu código en los momentos apropiados. Este es un patrón conocido como "Inversión de Control" (IoC). Un ejemplo común de un framework JavaScript es Angular, que proporciona una estructura detallada para tu aplicación y te dice exactamente dónde poner tu código y cómo se interactuará con él.

2.Complejidad y capacidad:

1. **Biblioteca**: Por lo general, **es menos compleja** y más fácil de aprender que un framework completo, pero también proporciona menos funcionalidad "fuera de la caja".
2. **Framework**: Contiene **múltiples bibliotecas** para diferentes tareas, así como código de infraestructura para pegarlas todas juntas y hacer que funcionen sin problemas. Es más poderoso y flexible que una sola biblioteca, pero también más complejo y tiene una curva de aprendizaje más empinada.

React actual

Los componentes de React se basan en funciones:

- Podemos tener estados en componentes funcionales vía Hooks

Las actualizaciones en React son asíncronas. Es decir si yo cambio un dato del estado de mi componente en la siguiente línea de código no veré ese cambio reflejado. Nosotros podemos solicitar el pintado o repintado de contenido pero no de manera directa sino como he indicado "solicitamos" y es el propio motor de React quién realiza la acción

El estado en React es inmutable:

- Nunca modificaremos un objeto, en caso de necesitar un cambio crearemos uno nuevo.
- Imagina que tenemos la ficha estudiante con los campos: id, nombre y apellidos. Si quisiese cambiar el campo de nombre lo que haré será crear una ficha de estudiante nueva en la que copiaré todos los campos y sobreescribo la propiedad de nombre.

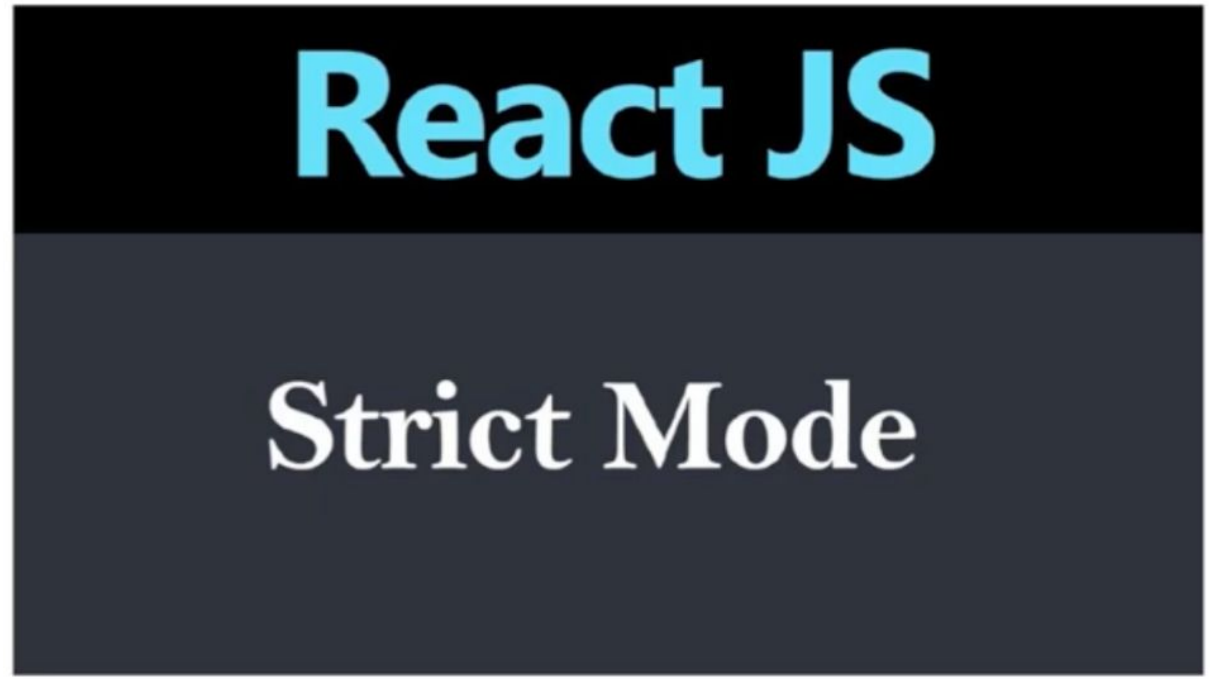
Flujo unidireccional:
React se basa en el flujo unidireccional de los datos, es decir tengo los datos de mi aplicación y estos fluyen del componente **padre a los hijos.**

React Strict Mode puede hacer un doble renderizado de los componentes para **ayudar a encontrar problemas durante la fase de desarrollo**. Esto se hace solo en el entorno de desarrollo y no en producción.

Los doble renderizados ayudan a detectar efectos secundarios no deseados en el renderizado de los componentes.

Al realizar un doble renderizado de los componentes, React Strict Mode puede alertar a los desarrolladores sobre las operaciones que se realizan más de una vez cuando no deberían serlo. Esto ayuda a **prevenir errores sutiles y difíciles de detectar** que pueden surgir debido a efectos secundarios no intencionados.

Es importante señalar que aunque React Strict Mode hace un doble renderizado en el entorno de desarrollo, **no se realizan doble renderizados en el entorno de producción** y, por lo tanto, no tiene ningún impacto en el rendimiento en producción.



Fin