

El concepto de Virtual DOM es un concepto que cada día esta más de moda. Vamos a intentar explicarlo de una forma gráfica. Imaginemonos que disponemos de una tabla con datos en una página HTML.

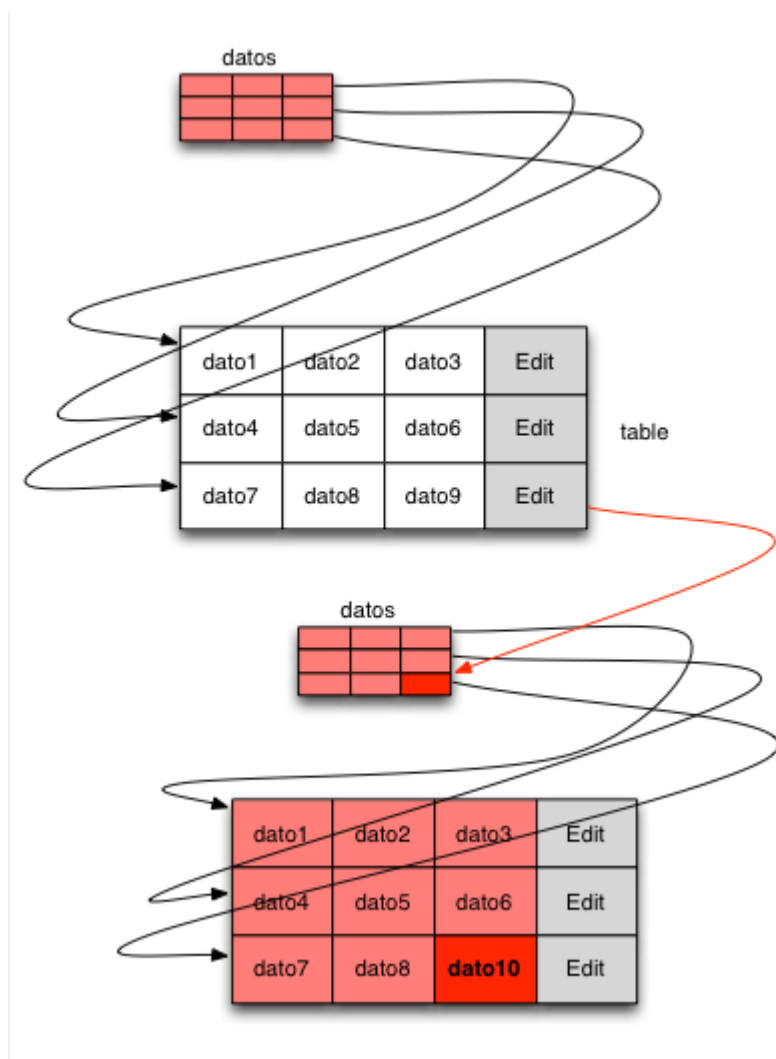
dato1	dato2	dato3	Edit
dato4	dato5	dato6	Edit
dato7	dato8	dato9	Edit

Esta tabla dispone de en cada fila de un botón de edición. Este botón nos permite cambiar de forma rápida los datos de la tabla. Por ejemplo podemos pulsar en la última fila y cambiar el dato9 por dato10.

dato1	dato2	dato3	Edit
dato4	dato5	dato6	Edit
dato7	dato8	dato10	Edit

Cambios sin VirtualDOM

¿Como realizamos este operación? . En la mayor parte de las situaciones los desarrolladores optan por una solución similar a lo siguiente. La tabla se crea a partir de un array de objetos , este array de objetos es actualizado cuando editamos. Finalizada la edición la tabla se vuelve a generar.

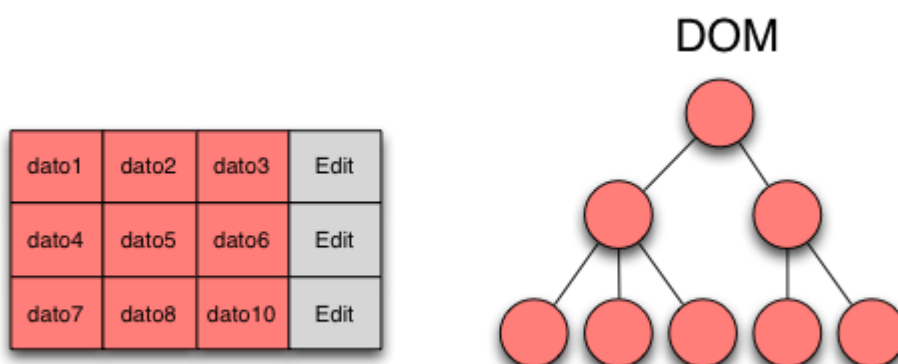


Nos encontramos ante un código que funciona correctamente pero que tiene un problema importante. ¿Era necesario volver a generar la tabla entera?. Se trata de una pregunta interesante ya que solo hemos cambiado un dato puntual. La respuesta es NO. Acabamos de forzar al navegador a redibujar todo el bloque cuando solo un mínimo de información cambiaba.

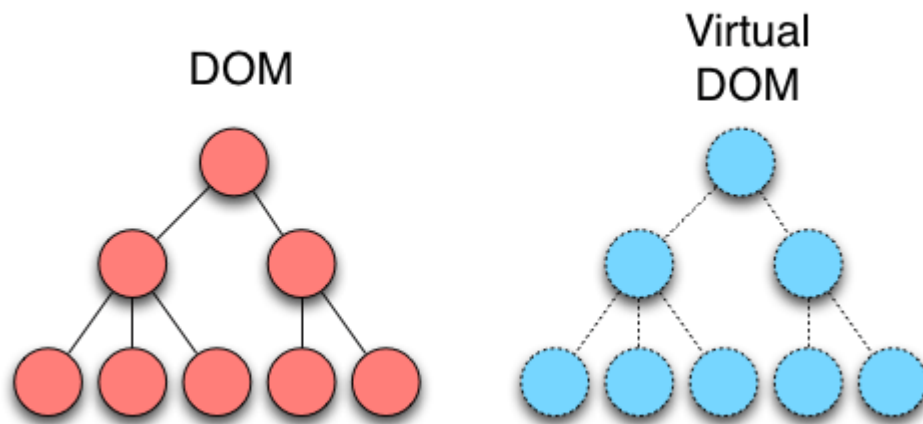
El concepto de Virtual DOM

Cambiar los nodos de un árbol DOM tiene un coste, cuantos más nodos cambiemos mayor será el coste para el navegador. En situaciones sencillas los problemas de rendimiento no se

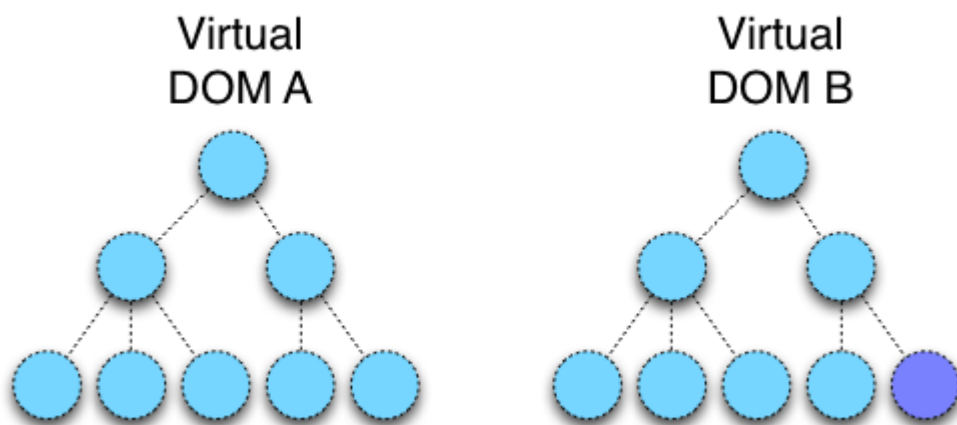
notan pero cuanto más compleja y grande sea nuestra web estas cosas serán importantes. En este caso nos encontramos con una tabla de datos que define una parte del árbol DOM



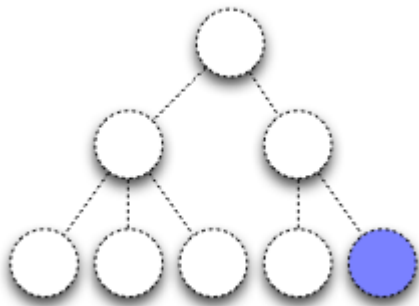
Lamentablemente cuando volvemos a dibujar la tabla estamos cambiando todos los nodos del árbol DOM y volviéndola a renderizar por completo. ¿Cómo funciona un enfoque de virtual DOM? . En este caso el framework o librería que corresponda se encarga de tener una copia de nuestro árbol DOM simplificado en memoria a la cual se denomina VirtualDOM.



Una vez que dispone de esta versión del árbol el framework se encarga de estar pendiente de los cambios que se generan sobre él. Siendo capaz de ver las diferencias entre un estado A y un estado B cuando nosotros actualizamos información.

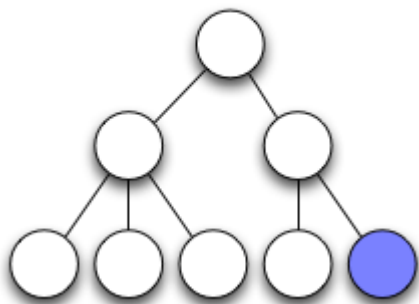


En este caso solo hemos cambiado un nodo del árbol:



Con estos cambios mínimos el framework o librería se encargará de actualizar el árbol DOM original sin tenerlo que redibujar completo.

DOM



Con este tipo de técnicas se reduce al mínimo las renderizados a nivel de navegador. El rendimiento se incrementará de forma considerable.

Otros artículos relacionados: [Sublime y React](#) , [10 Atom Plugins](#) , [JavaScript Memorization](#)