Users DB

UI de gestión de usuarios y roles y primera implementación de vDOM

Gerard Rodes Vidal

Contenido

Introducción

En la prueba definida en el documento “frontend test.docx”, se pedía realizar una interfaz para una aplicación web, capaz de administrar una base de datos en la que hay guardados una serie de registros de usuarios y roles.

Uno de los condicionantes del test era que no podía utilizar ningún framework ni librería. En un primer momento pensé en realizar una interfaz básica y definir la funcionalidad de la aplicación a partir de event listeners y controladores alrededor de los inputs de la interfaz, pero, ya que la finalidad de la prueba era comprobar mis conocimientos de JavaScript, decidí afrontar el reto de desarrollar, desde cero, un framework de web components y mi propia implementación del Virtual DOM, consiguiendo así una funcionalidad semejante a la que ofrecen Angular, React, Vue, Polymer o Ember, entre otros.

Mis conocimientos de JavaScript son limitados, mi primer contacto con este lenguaje de programación ocurrió hace unos 2 años, aun así, se ha convertido en uno de los lenguajes con los que más disfruto trabajando. Nunca antes había desarrollado un framework de estas condiciones, y aunque ha requerido mucha investigación y prueba y error, he terminado más que satisfecho con el resultado.

Obviamente mi framework no ofrece todas las utilidades que puede ofrecer cualquier framework de las anteriormente nombrados, ya que he ido desarrollándolas a medida que iba necesitando utilizarlas.

A través del código sigo una serie de convenciones a la hora de nombrar atributos, algunas de las más importantes son:

- Prefijo “_”: un atributo prefijado con “_” será reconocido como un atributo de privado, el cual no terminará nunca teniendo un impacto directo en el DOM y que se utiliza para manjar datos privados o metadatos.
- Prefijo “$”: un elemento es prefijado con “$” para simbolizar que está presente en el DOM, es decir, que es un elemento el cual está montado en la estructura HTML de nuestra aplicación.
- Prefijo “v”: un objecto/diccionario es prefijado con “v” cuando es una representación de un vNode, es decir, la representación del vDOM de un elemento del DOM.

Instalación

1. Clonar o descargar el repositorio https://github.com/GerardRodes/Users-DB
2. Dentro de la carpeta descargada, asegurándote de tener Node.js y npm instalados, ejecutar el comando: “npm install”, se instalarán todas las dependencias necesarias.
3. Introduciendo el comando “npm start” la aplicación se iniciara en http://localhost:9000
    a. Si al introducir “npm start” salta algún error del tipo “couldn’t resolve” instalar el paquete que se menciona con “npm install nombreDelPaquete”, o poneros en contacto conmigo para poder resolverlo.
    b. “npm start” inicia tanto el servidor json-server como la aplicación, al estar un rato funcionando el servidor puede dejar de response, creo que se puede deber a estar ejecutando ambos comandos desde la misma instancia de la consola, se soluciona parando el proceso con “Ctrl+C” y volviendo a ejecutar “npm start”. Si el error persiste, probar a ejecutar primero el servidor con “npm run server”, y luego, en otra instancia de la consola aparte, iniciar la aplicación con “npm run dev”
4. Para ejecutar los tests, ejecutar el comando “npm test”.

Virtual DOM

¿Qué es?

El Virtual DOM, o vDOM, es una representación virtual (en este caso, en lenguaje de JS), de la estructura de elementos, presentes en el DOM (Document Object Model).

Para realizar tal representación se entiende cada elemento del DOM como un diccionario (u objeto, mencionado en el código como vNode), el cual está compuesto de tres atributos básicos:

- El nombre del tag que lo representa (div, span, ul, li, …)
- Las propiedades que lo conforman (desde propiedades del dom como id, class, etc hasta propiedades para el funcionamiento interno del vDOM)
- Sus hijos, es decir, los elementos los cuales están alojados dentro del elemento, en relación directa.

¿Cómo funciona?

Cada uno de los componentes presentes en la web está formado de una función “render”, la cual retorna una serie de objetos (mencionados como vNodes), en estructura de árbol. Al estar en estructura de árbol se obtiene un solo objeto, formado por múltiples vNodes, el cual representa toda la estructura de la aplicación. Luego estos objetos se “parsean” para generar los objetos que representan y mantenerse enlazados y actualizados.

JSX

Para generar un vNode puede hacerse escribiendo un objeto tal cual, con los atributos necesarios, pero hemos de representar todos nuestros elementos HTML con esta sintaxis, por lo que puede hacerse pesado y engorroso. Para facilitar el trabajo podríamos programar una función, a la cual, pasándole unos cuantos parámetros nos generara el vNode correspondiente, por convención, esta función es llamada “h” (HyperScript), mi implementación de “h” se encuentra en “src/vdom/h.js”.

Aunque “h” ha facilitado en cierta medida la sintaxis, existe algo más de “sintactic sugar”, llamado JSX. JSX nos permite escribir nuestros elementos HTML con una sintaxis XML que nos resultara más familiar, luego, el módulo “babel-plugin-transform-react-jsx” se encarga de transformar este XML en múltiples llamadas a nuestra función “h”. La asignación de mi propia función “h” está configurada en “.babelrc”.

Mounting

Todos estos vNodes generados por la función “h”, que conforman nuestro vDOM, necesitan ser “parseados” para generar a partir de ellos un DOM real. Para ello tienen que montarse.

Mounting es el proceso el cual a partir de un vNode se genera su elemento correspondiente y se añade al DOM, los procesos de mounting de mi implementación del vDOM se efectúan en “src/vdom/mounting.js”. Cuando un objeto es montado se vinculan, de manera circular, todas sus representaciones, es decir, el vNode, la instancia (si es un componente) y el elemento real en el DOM.

Updating

Una vez representado nuestro vDOM en el DOM real, nuestros componentes comenzaran a mutar de estados y propiedades, por lo que la finalidad del proceso de updating, es la de mantener toda esta información actualizada y representada correctamente, tanto en el DOM como en el vDOM.

Los procesos de updating son gestionados desde “src/vdom/updating.js”.

Tipos de vNode

Como he dicho, para representar los elementos del DOM en mi vDOM, lo hago a partir de vNodes, aun así, no todos los elementos son iguales, por lo que se requiere por lo menos, de 3 tipos distintos de vNodes:

vText

vNode que representa un text node, el cual es creado con un “document.createTextNode”. Este vNode está conformado únicamente por una string, o número, los cuales son transformados a texto.

vElement

Este vNode representa un elemento normal del DOM como podría ser un div, input, form, o cualquier otro. Este vNode está conformado por el nombre del tag, sus propiedades, y sus hijos.

vComponent

Este vNode representa un web component, el cual contendrá la lógica de nuestra aplicación. Está formado por la clase para generar instancias suyas y todos sus métodos y atributos (como la función “render” o el atributo “state”). En una primera instancia un vComponent no contiene hijos, ya que estos son definidos una vez se ejecuta la función “render” que determina la estructura del vComponent.

Para generar un vComponent cuento con la clase Component definida en “src/vdom/Component.js”, la cual otorga métodos como setState, update o life cicle events.

App

La app desarrollada cuenta con dos secciones principales: Users y Roles.

Roles

Muestra un listado de los roles disponibles y a su derecha los permisos que otorgan cada uno.

Users

Presenta un listado de los usuarios existentes, y a su derecha los roles asignados a cada uno, en la parte superior del listado hay un input text el cual puede ser utilizado para filtrar los usuarios por el nombre.

Si clicamos sobre los roles de un usuario se nos permitirá asignar a desasignar roles, pero siempre teniendo en cuenta que debe tener como mínimo un rol asignado.

Si clicamos sobre el nombre de un usuario este quedara seleccionado, una vez seleccionado, si clicamos sobre “Delete user”, el usuario será eliminado de la base de datos. Podemos deseleccionar el usuario volviendo a clicar sobre él.

Clicando sobre el botón de “Add user +” se nos mostrara la opción de crear un nuevo usuario, asignando un nombre y un mínimo de un rol.

Referencias

https://en.wikipedia.org/wiki/Hypertext

https://medium.com/@deathmood/how-to-write-your-own-virtual-dom-ee74acc13060

https://medium.com/@calebmer/write-your-own-react-js-776dbef98b8

https://swennemans.gitbooks.io/building-your-own-react-js/content/01-basics/01-virtual-dom.html

https://blog.javascripting.com/2016/10/05/building-your-own-react-clone-in-five-easy-steps/

https://medium.com/@rajaraodv/the-inner-workings-of-virtual-dom-666ee7ad47cf