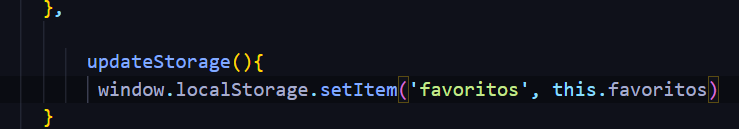
**Persistencia para que se vea la información después de refrescar**

Para esto se usa el almacenamiento local del navegador. Ver <https://developer.mozilla.org/es/docs/Web/API/Window/localStorage>

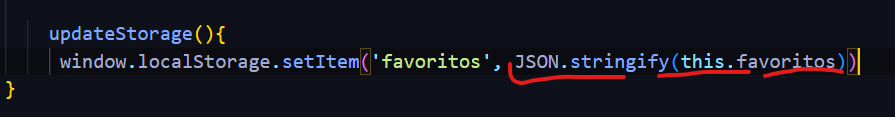
Aunque solo se pueden almacenar string, lo solucionaremos más adelante.

1. Se crea un nuevo método en la instancia Vue (en methods, debajo de RemoverFavorito).



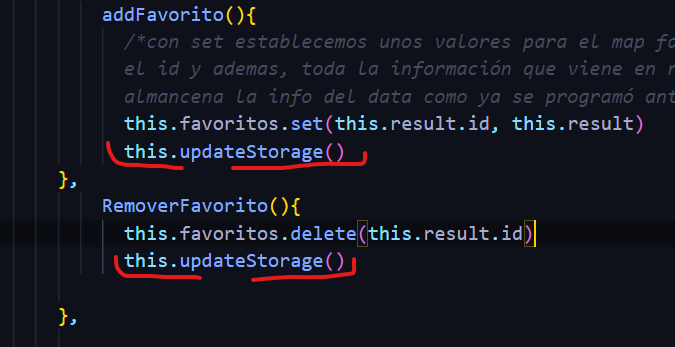
Pero esto no funcionaría ya que no podemos almacenar el map favoritos aquí, solo se pueden string.

1. Entonces usaremos un método de json para convertir el map nativo en un string json.

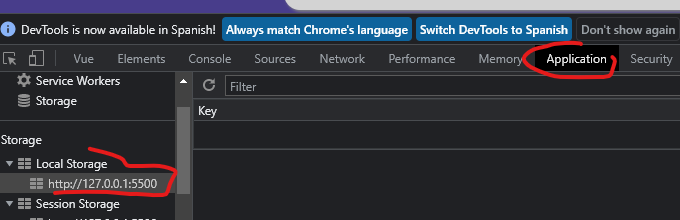


Luego para capturar el valor y traerlo de vuelta, habrá que hacer lo contrario, traer este string de tipo json y convertirlo otra vez en map.

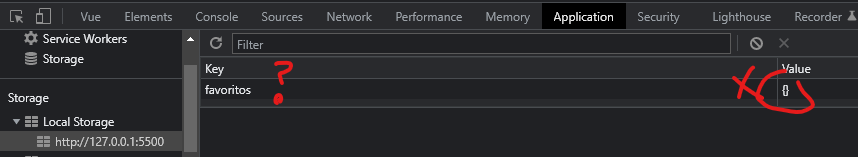
1. Lo anterior se trabaja cuando añado un favorito o cuando elimino un favorito. Por eso añadimos las siguientes líneas en los métodos respectivos:



1. Verificamos en Inspeccionar, en la opción de Application, en la sección de Local Storage:



1. Cuando añadimos un favorito, allí vemos información:



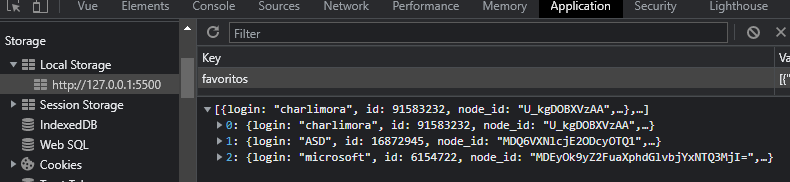
Pero notamos que devuelve un array vacío, lo cual es un error de codificación.

|  |
| --- |
| ***Es posible que adicional, hallemos otro error. En este ejemplo, se cometió el error de usar la palabra favorite en vez de favorito, al llamar el atrito name y quererlo poner en el alt de la imagn del div favorites donde se muestran los favoritos:*** |

1. Realmente no es favoritos lo que se debe usar sino TodosFavoritos(), donde traemos toda la información.



1. Volvemos a verificar en Local Storage:

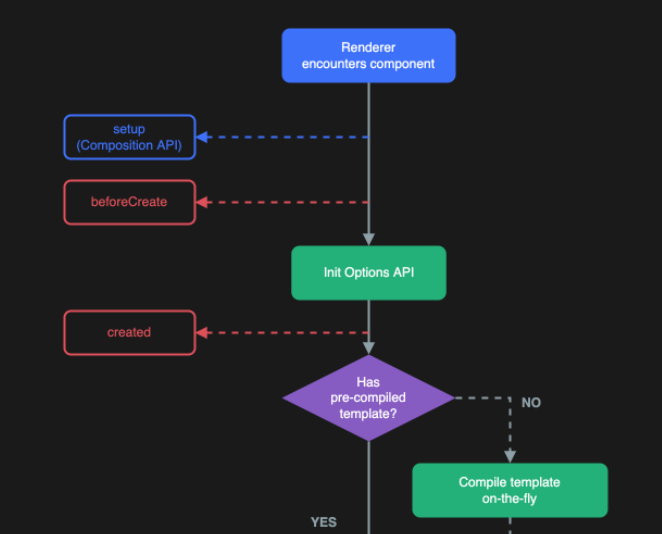


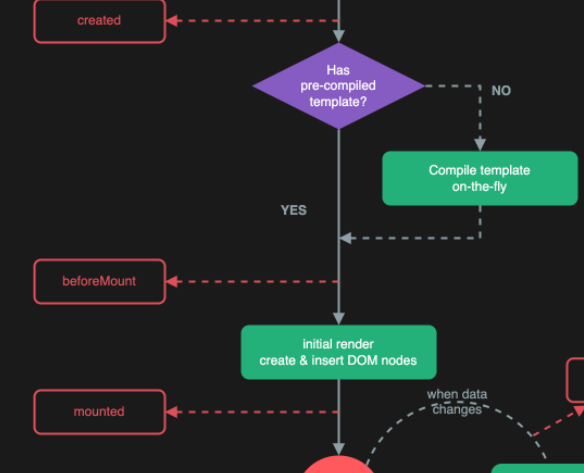


1. Si recargamos, la información no se verá más en pantalla, pero sí en Inspeccionar, en la sección de Local Storage.

**Recuperamos la información si existiera**

**Primero vamos a consultar los ciclos de vida de Vue en** [**https://vuejs.org/guide/essentials/lifecycle.html#registering-lifecycle-hooks**](https://vuejs.org/guide/essentials/lifecycle.html#registering-lifecycle-hooks)

****

****

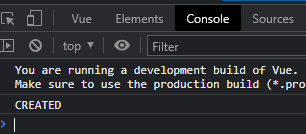
Es bueno tener en cuenta las diferencias entre created y mounted. Es de tener en cuenta que es mounted el que tiene acceso a la plantilla, al DOM. Created se ejecuta unos cuantos segundos antes que el mounted pero no tiene acceso al DOM.

En nuestro caso solo necesitamos acceder al almacenamiento local, no al DOM. En este caso el hook adecuado (enganche) es el created.

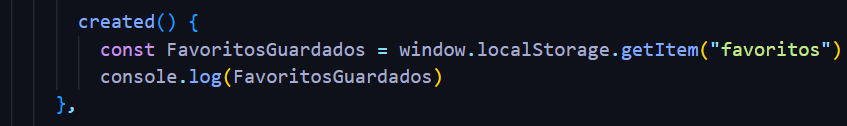
1. Iniciamos un created después del data:



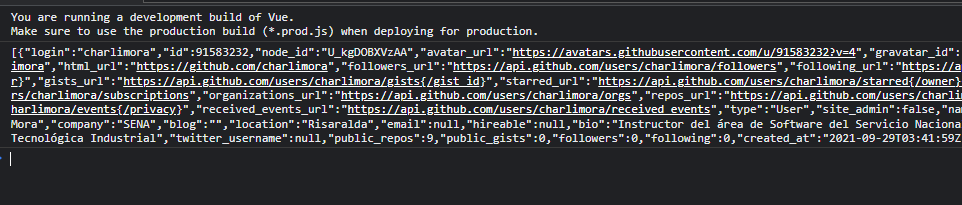
1. Lo podemos ver en consola:



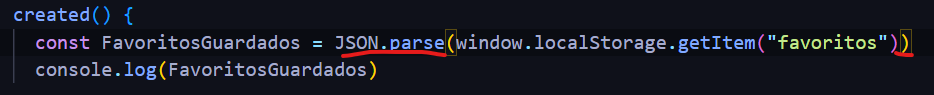
1. Queremos consultar el almacenamiento local para ver si hay favoritos:



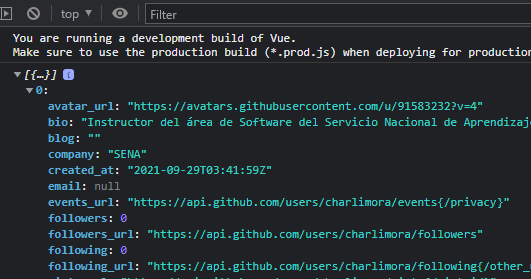
Pero la respuesta es un string:



1. Para reconvertirlo, debemos usar JSON.parse().



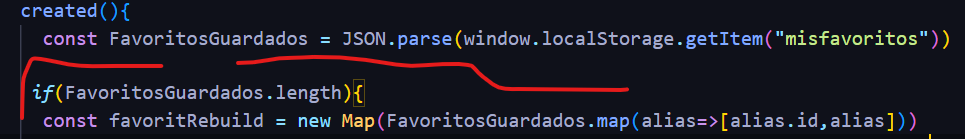
1. Ahora la información sale organizada dentro de un array



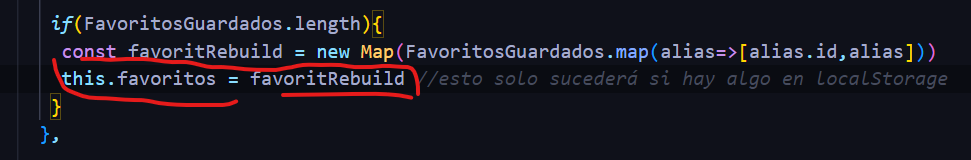
1. El reto ahora es almacenar esta información en el mapa de Favoritos. Para lo cual eliminamos el console log y creamos una nueva constante llamada favoritos (no confundir con el favoritos que ya existe en las propiedades en data).

\*\*\*\*\*

|  |
| --- |
| *con Map() creamos un nuevo objeto Map, y se le puede asignar su {clave, valor}. Es decir, Map(key,value). Pero ¿cómo podemos obtener el id del usuario que ya lo venimos usando como key y el resto de su información que lo hemos usado ya como value? En este caso podemos usar el método map de los arrays de Javascritp. Con este método, usando una especie de alias (si se me permite el uso del término), podemos acceder a una posición específica de un array de Javascript. Se usa el operador flecha => y a través de ese mismo alias accedemos a id y al resto de información, ya que alias en sí es toda la info del array* |



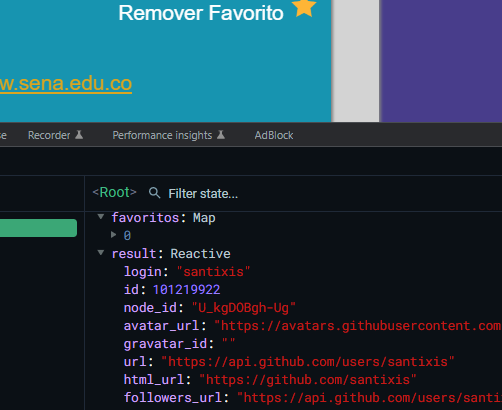
1. Finalmente, hacemos que el favoritos que tenemos en las propiedades de la instancia sea igual a la constante favoritos que acabamos de trabajar en el created():



\*\*\* se debe tener cuidado en que esta última línea no quede fuera del if, ya que podría generar un error donde no se muestre nada en pantalla.

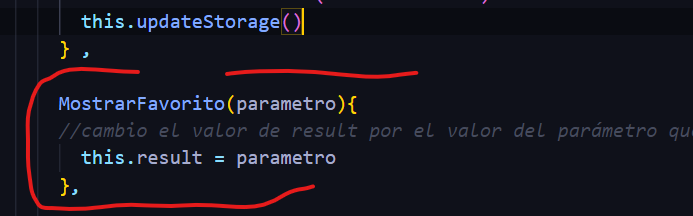
***Mostrando los favoritos al dar clic en la foto del usuario Favorito***

Para lograr este resultado, es de tener en cuenta que el atributo de Vue, que está almacenando toda la información del usuario y que posteriormente permite mostrarla en pantalla, es result. Lo podemos verificar nuevamente al buscar un usuario y utilizar Vue Devtools:



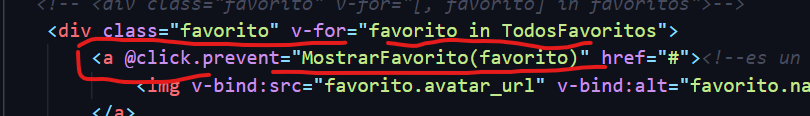
Ahora bien, recordar que para mostrar cada usuario seleccionado como favorito usamos un v-for, y que tuvimos que convertir el Map favoritos en un array el cual recorremos con un alias al que llamamos favorito o de cualquier otra forma. Lo que queremos es valernos de las propiedades de reactividad en ambos sentidos de Vue, haciendo que result finalmente tenga la información de cada alias (o favorito) que se seleccione en un caso dado.

1. Creamos una nueva función en methods:



Como vemos, esta función estará esperando un parámetro. Cuando se entregue el parámetro, él hará que result equivalga a ese parámetro. Y ¿cuál será ese parámetro? Va a ser el alias usado en el v-for de result para cada favorito.

1. Vamos al html, ubicamos el div de favoritos donde está el v-for e insertamos un v-on: (abreviado como @), específicamente en el link(a) que conectará con el método MostrarFavorito. El parámetro que se le pasará será el ya mencionado:



1. Guardamos, recargamos, y nos percatamos que al dar clic en cada favorito aparece en pantalla sin tener que buscarlo otra vez por el input.

Hasta aquí este interesante ejercicio.