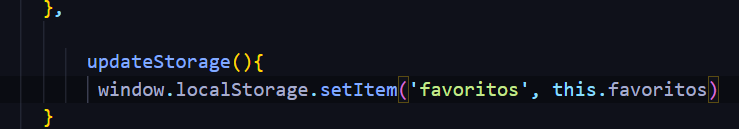
**Persistencia para que se vea la información después de refrescar**

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

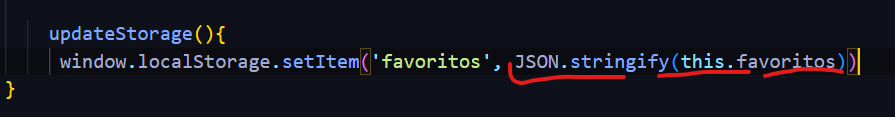
Aunque solo se pueden almacenar string, lo solucionaremos más adelante.

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



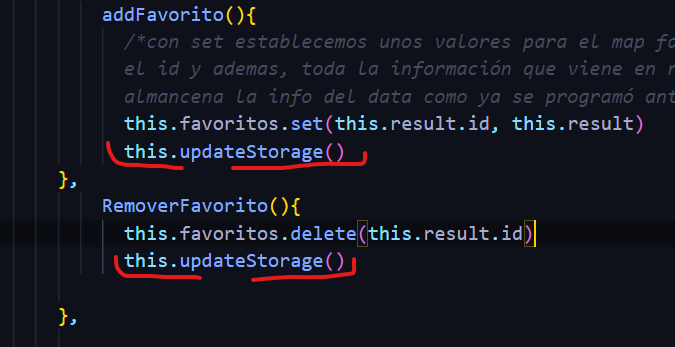
Pero esto no funcionaría ya que no podemos almacenar el map favoritos aquí, solo se pueden string.

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

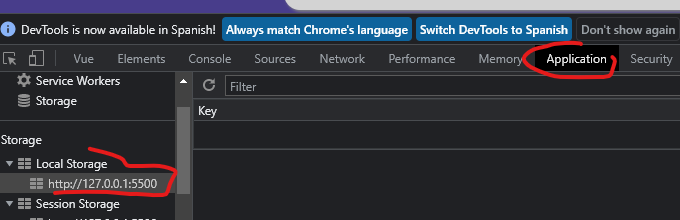


Luego para capturar el valor y traerlo de vuelta, habrá que hacer lo contrario, traer este string de tipo json y convertirlo otra vez en map.

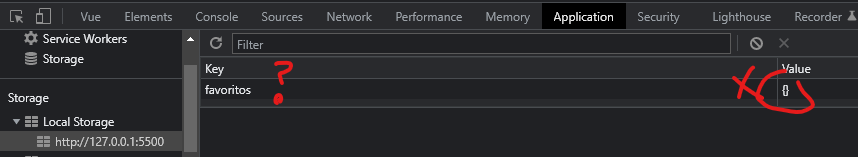
1. Lo anterior se trabaja cuando añado un favorito o cuando elimino un favorito. Por eso añadimos las siguientes líneas en los métodos respectivos:



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



1. Cuando añadimos un favorito, allí vemos información:



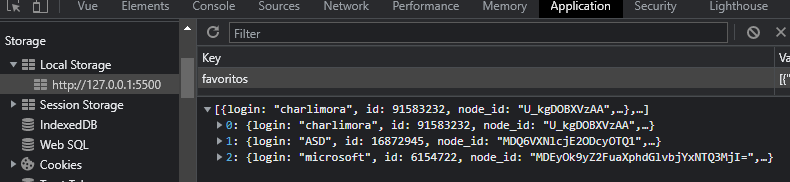
Pero notamos que devuelve un array vacío, lo cual es un error de codificación.

|  |
| --- |
| ***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:

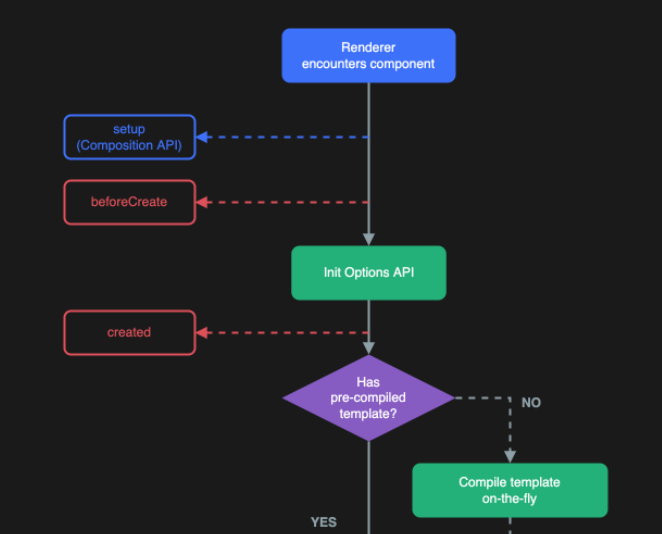


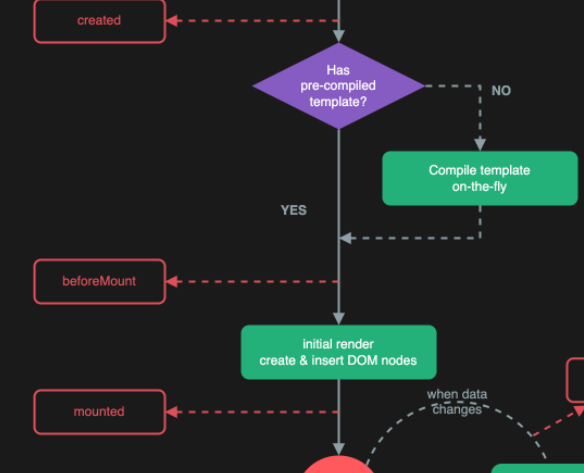


1. Si recargamos, la información no se verá más en pantalla, pero sí en Inspeccionar, en la sección de Local Storage.

**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)

****

****

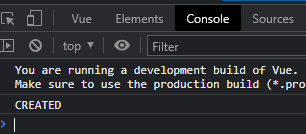
Es bueno tener en cuenta las diferencias entre created y mounted. Es de tener en cuenta que es mounted el que tiene acceso a la plantilla, al DOM. Created se ejecuta unos cuantos segundos antes que el mounted pero no tiene acceso al DOM.

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

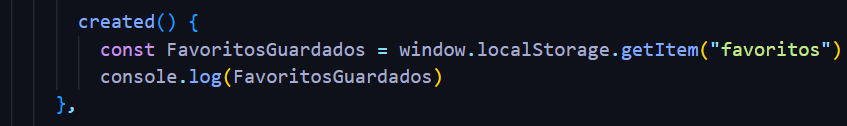
1. Iniciamos un created después del data:



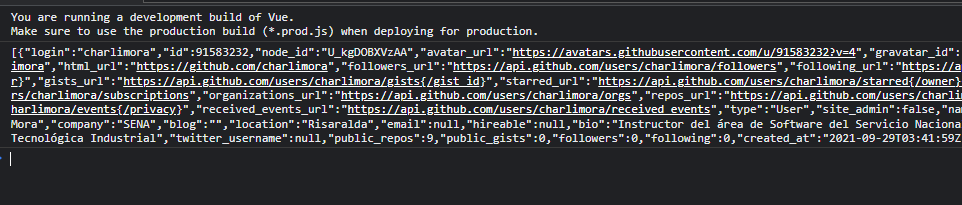
1. Lo podemos ver en consola:



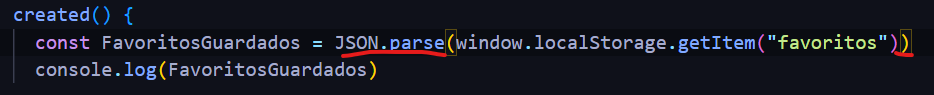
1. Queremos consultar el almacenamiento local para ver si hay favoritos:



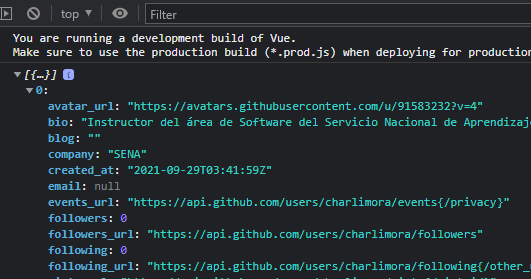
Pero la respuesta es un string:



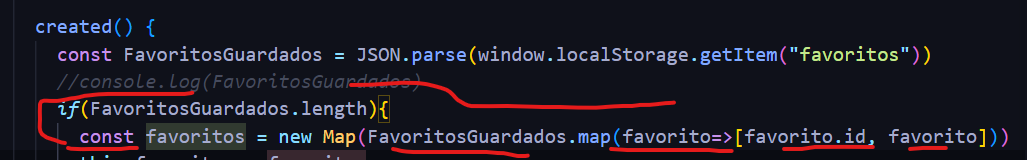
1. Para reconvertirlo, debemos usar JSON.parse().



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).



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():

