Clase 10 - Storage & JSON

CLASE N°9 - Glosario

Evento: es la manera que tenemos en Javascript de controlar las acciones de los usuarios, y definir un comportamiento de la página o aplicación cuando se produzcan. Hay distintos tipos de eventos:

* Eventos de mouse
* Eventos de teclado
* Evento change
* Evento submit

Storage

Storage o almacenamiento

El objeto Storage (API de almacenamiento web) permite almacenar datos de manera local en el navegador sin necesidad de realizar ninguna conexión con el servidor.

De esta manera, cada cliente puede preservar información de la aplicación.

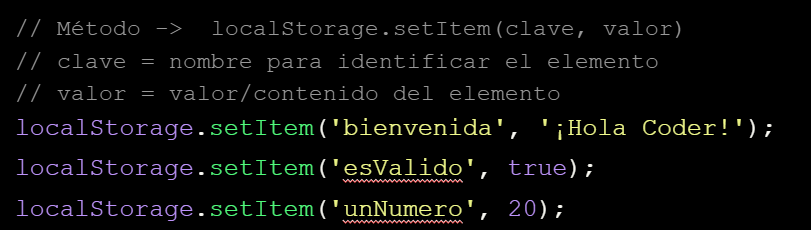
El navegador nos ofrece dos tipos de storage: localStorage y sessionStorage.

Localstorage: Setitem

Los datos almacenados en localStorage (variable global preexistente) se almacenan en el navegador de forma indefinida (o hasta que se borren los datos de navegación del browser):

La información persiste reinicio de navegador y hasta del sistema operativo.

Para almacenar información se utiliza setItem:

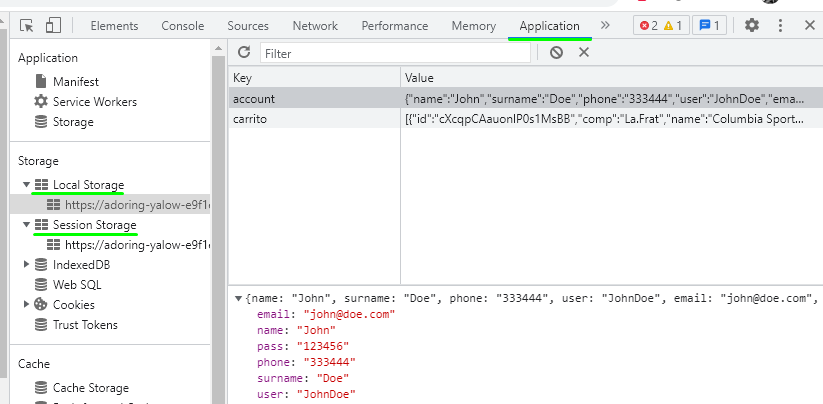


Clave-valor

La información almacenada en el Storage se guarda en la forma de clave-valor.

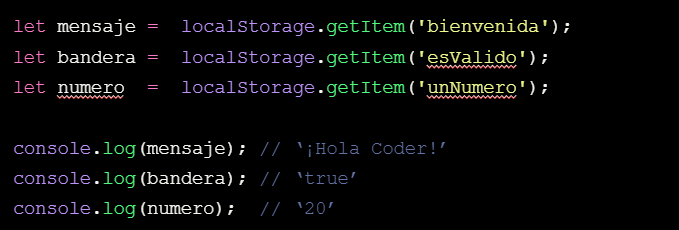
Similar al tratamiento de objetos, definimos claves en el storage donde almacenamos valores.

Podemos ver el Storage en el navegador a través de la pestaña de application:



Localstorage: getitem

Podemos acceder a la información almacenada en localStorage utilizando getItem. Las claves y valores de Storage se guardan en formato de cadena de caracteres (DOMString).



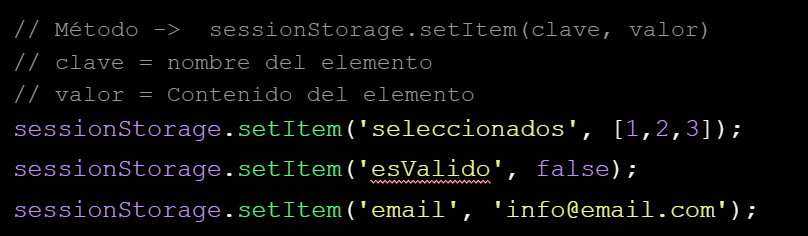
Localstorage: Setitem

La información almacenada en sessionStorage (variable global preexistente) se almacena en el navegador hasta que el usuario cierra la ventana.

Solo existe dentro de la pestaña actual del navegador. Otra pestaña con la misma página tendrá otro sessionStorage distinto, pero se comparte entre iframes en la pestaña (asumiendo que tengan el mismo origen).

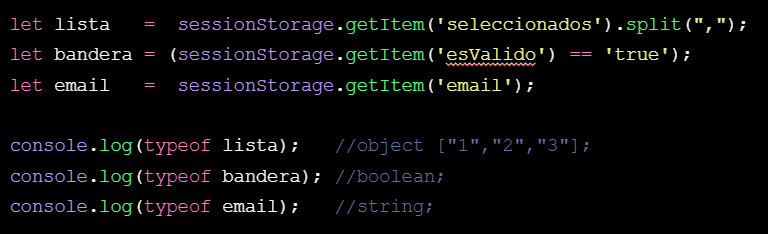
Sessionstorage: getitem

El tratamiento es similar al localStorage:



Sessionstorage: getitem

Podemos acceder a la información almacenada en sessionStorage utilizando getItem. Las claves y valores de Storage se guardan siempre en formato de cadena de caracteres.

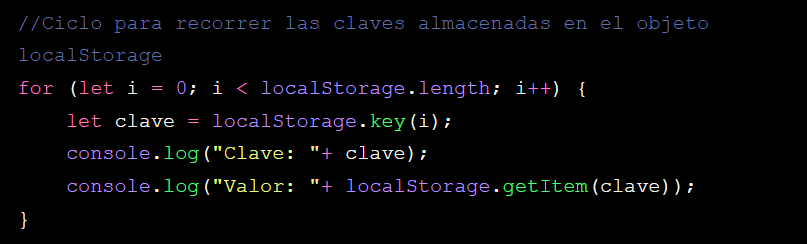


Recorriendo el storage

Es posible obtener todos los valores almacenados en localStorage o sessionStorage con un bucle.

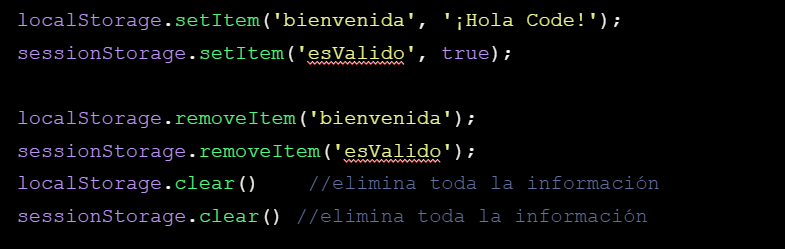
Pero no podemos usar for...of porque no son objetos iterables, ni for...in porque obtenemos otras propiedades del objeto que no son valores almacenados.

El bucle a emplear es for con el método key:



Eliminar datos del storage

Podemos eliminar la información almacenada en sessionStorage o localStorage usando el método removeItem o clear:



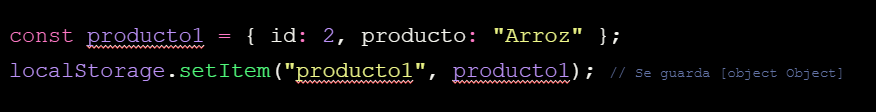
JSON

Almacenar objetos en storage

Si queremos almacenar la información de un objeto en un storage, hay que tener en cuenta que tanto la clave como el valor se almacenan en strings.

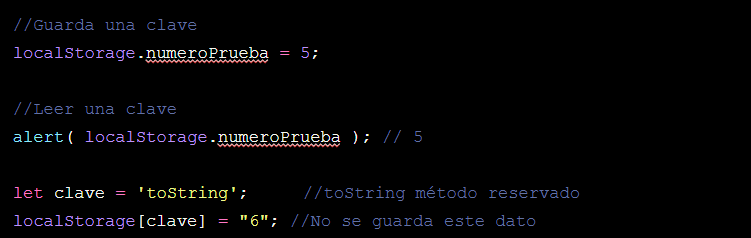
Ante cualquier otro tipo a guardar, como un número o un objeto, se convierte a cadena de texto automáticamente.

Entonces, al buscar almacenar un objeto sin una transformación previa, guardamos [object Object], la conversión por defecto de objeto a string. Para guardar la información correctamente hay que transformar el objeto a JSON.



Acceso tipo objeto

Dado que localStorage y sessionStorage son objetos globales, es posible crear y acceder a las claves como si fueran propiedades. Pero esto no es recomendable, porque hay eventos asociados a la modificación del storage cuando se emplea getItem o setItem.



¿Qué es JSON?

JavaScript Object Notation (JSON) es un formato basado en texto plano, para representar datos estructurados con la sintaxis de objetos de JavaScript. Es comúnmente utilizado para enviar y almacenar datos en aplicaciones web.

Aunque es muy parecido (casi similar) a la sintaxis de JavaScript, puede ser utilizado independientemente de JavaScript, y muchos entornos de programación poseen la capacidad de leer (convertir; parsear) y generar JSON.

JSON es un string con un formato específico.

Conversión de objetos y almacenamiento

Conversiones de/hacia JSON

Cuando sea necesario enviar un objeto Javascript al servidor o almacenarlo en storage, será necesario convertirlo a un JSON (una cadena) antes de ser enviado.

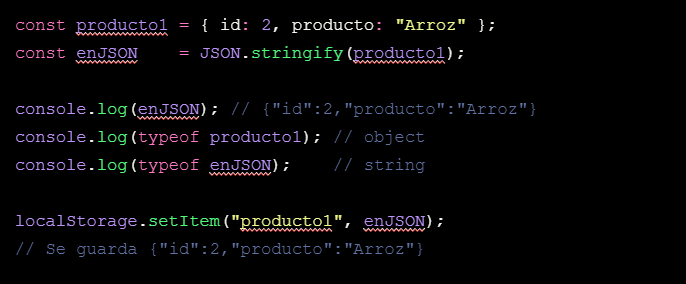
Cuando sea necesario enviar un objeto Javascript al servidor o almacenarlo en storage, será necesario convertirlo a un JSON (una cadena) antes de ser enviado.

Para eso usamos los siguientes métodos:

* stringify: acepta un objeto como parámetro, y devuelve la forma de texto JSON equivalente.
* parse: recibe un texto JSON como parámetro, y devuelve el objeto JavaScript correspondiente.

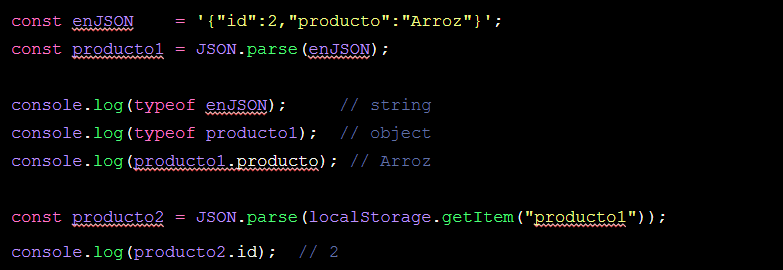
Stringify

Con JSON.stringify podemos transformar un objeto JavaScript a un string en formato JSON.



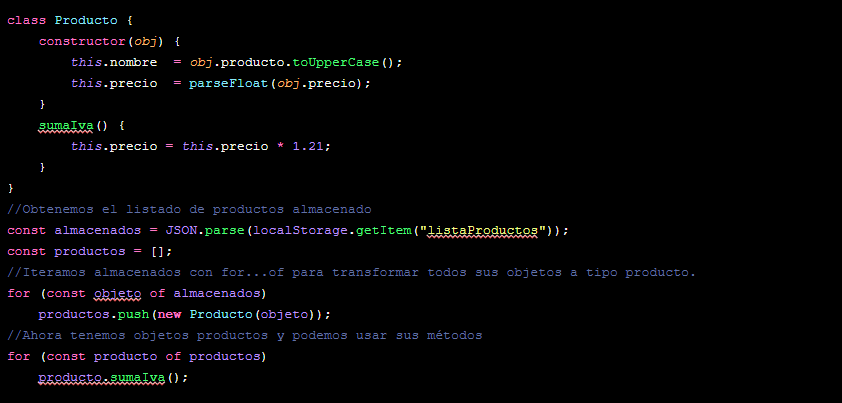
Parse

Con JSON.parse podemos transformar string en formato JSON a objeto JavaScript.



Ejemplo aplicado:

almacenar array de objetos

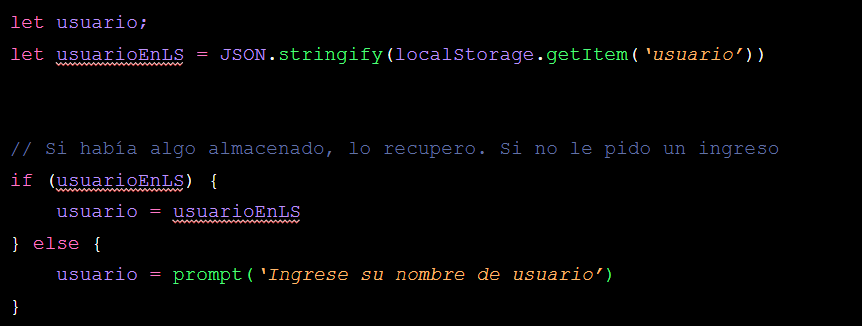


Recuperar datos

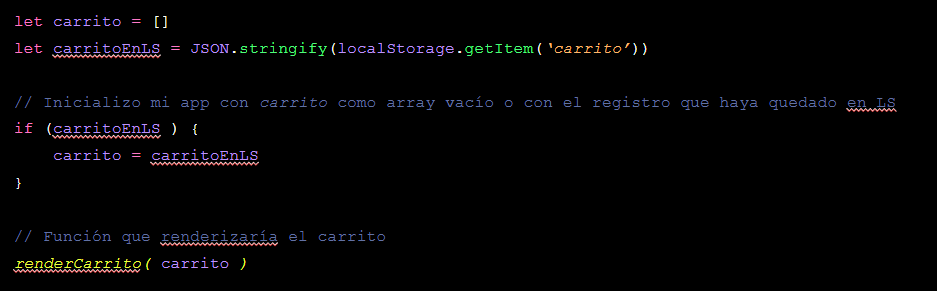
Muchas veces usamos el Storage para recuperar datos relacionados a la última navegación del usuario. Por ejemplo, su última sesión de login o el último estado de su carrito de compras.

Para esto, pensamos en inicializar las variables de la app consultando el Storage en el momento de inicio.

Ejemplo aplicado: recuperar estados previos



Ejemplo aplicado: almacenar array de objetos



MATERIAL AMPLIADO

**LocalStorage, sessionStorage**

* [Javascript.info](https://es.javascript.info/localstorage)

**JSON**

* [GitBooks. El formato JSON](https://josh1982.gitbooks.io/programacion-web-en-cliente/content/el_formato_json.html)
* [JSON Formatter](https://jsonformatter.curiousconcept.com/)
* [Generador JSON](https://www.mockaroo.com/)

**Documentación**

* [Documentación localStorage](https://developer.mozilla.org/es/docs/Web/API/Window/localStorage)
* [Documentación sessionStorage](https://developer.mozilla.org/es/docs/Web/API/Window/sessionStorage)
* [Documentación JSON](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/JSON)