AngularJS ngStorage

Learning Objectives

- Students will learn to save data (e.g., app state) to local storage and session storage.
- Students can explain the difference between local storage and session storage, and when to use each.

Resources

https://github.com/gsklee/ngStorage
https://blog.nraboy.com/2014/12/use-ngstorage-angularjs-local-storage-needs/

ngStorage

- ngStorage is a third party library. It exposes two services: **$localStorage** and **$sessionStorage**. These services wrap two pieces of native browser functionality that derive from the **Web Storage** specification:
  - **Local storage**: lasts until you or the user deletes it. Appropriate for long-term use (but not as persistent as server-side storage). Can communicate between tabs.
  - **Session storage**: attached to a specific window (tab in Chrome and Firefox). If a site is reloaded in the same window, the data persists. As soon as that window or tab is closed, the data is cleared. Less persistent than localStorage. Cannot communicate between tabs.

- Local and session storage are scoped to the document origin. E.g., example.com will not share data with google.com.
- For more details, see http://stackoverflow.com/questions/19867599/what-is-the-difference-between-localstorage-sessionstorage-session-and-cookies

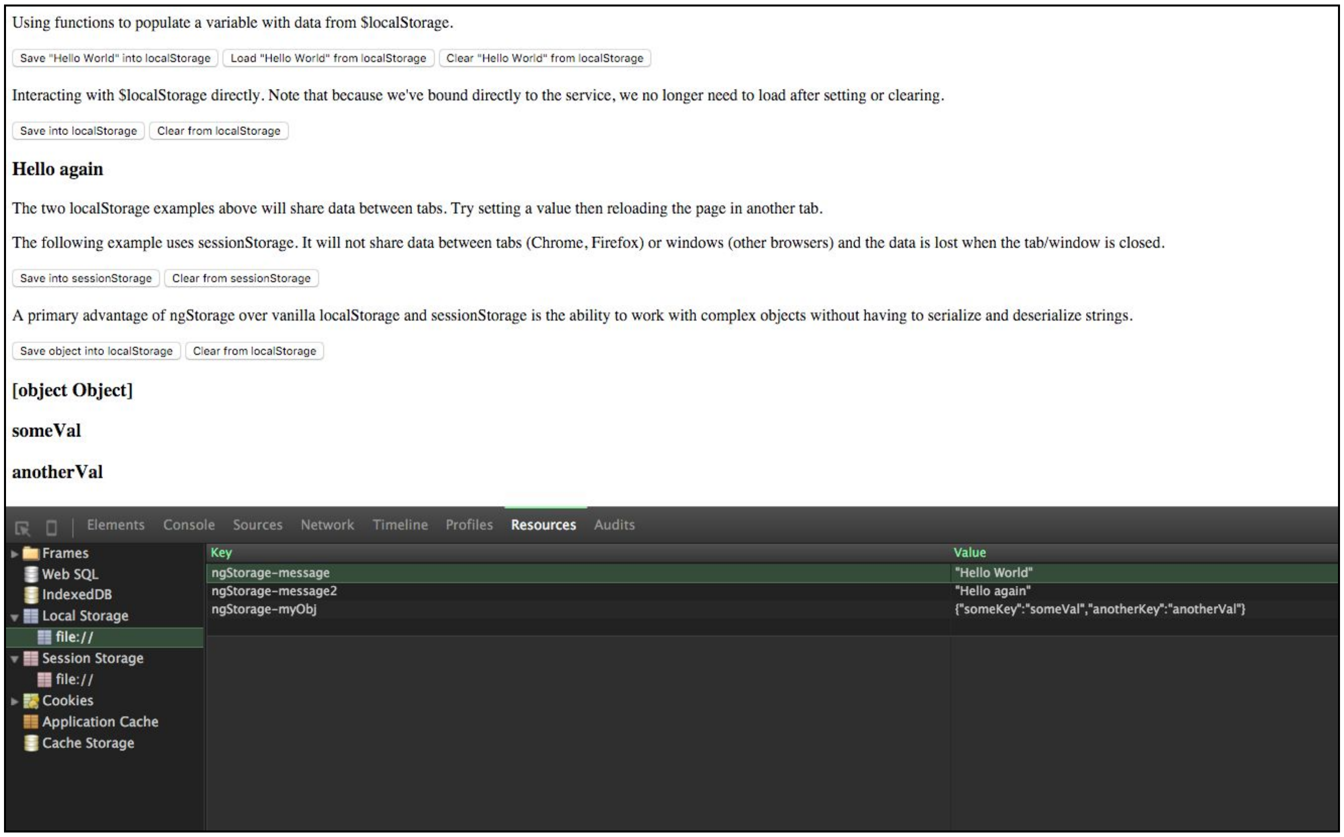
- Follow the installation instructions on the project's README here:
  1. Install using Bower, npm, or use CDN. E.g.,
     1. `bower install ngstorage --save`
  2. Include `ngStorage.js` in your `index.html` after the `angular.js` reference.
  3. Add module dependency:
     1. `angular.module('app', ['ngStorage'])`

Complete example: https://github.com/MountainlandWEB/ngStorage/blob/master/index.html

Hosted here: http://mountainlandweb.github.io/ngStorage/

- Note that this example shows using $localStorage to populate a scope variable, as well as interacting directly with the service from the markup. Exemplify use of localStorage by reloading the page or opening the page in multiple tabs. Exemplify use of sessionStorage by saving localStorage and sessionStorage values, closing tab, loading file in another tab, and noting that the localStorage items populate whereas the sessionStorage item does not.

- Demonstrate by viewing key:value pairs in dev tools:



Using functions to populate a variable with data from $localStorage.

Save "Hello World" into localStorage
Load "Hello World" from localStorage
Clear "Hello World" from localStorage

Interacting with $localStorage directly. Note that because we've bound directly to the service, we no longer need to load after setting or clearing.

Save into localStorage
Clear from localStorage

Hello again

The two localStorage examples above will share data between tabs. Try setting a value then reloading the page in another tab.

The following example uses sessionStorage. It will not share data between tabs (Chrome, Firefox) or windows (other browsers) and the data is lost when the tab/window is closed.

A primary advantage of ngStorage over vanilla localStorage and sessionStorage is the ability to work with complex objects without having to serialize and deserialize strings.

[object Object]

someVal

anotherVal

Key	Value
ngStorage-message	"Hello World"
ngStorage-message2	"Hello again"
ngStorage-myObj	{"someKey":"someVal","anotherKey":"anotherVal"}

- Additional example here: https://blog.nraboy.com/2014/12/use-ngstorage-angularjs-local-storage-needs/

- Contrast with persisting an object with vanilla localStorage:
  - localStorage can only handle key/value pairs where both the keys and the values are strings. ngStorage is more convenient because this serialization happens automatically.
  - Setting a value:
    - var someObject = { key1: "value1", key2: "value2" };
    - window.localStorage.set("myItem", JSON.stringify(someObject));
  - Getting a value:
    - var someObject = JSON.parse(window.localStorage.get("myItem"));

- Another very popular AngularJS localstorage library is angular-local-storage: https://github.com/grevory/angular-local-storage