

Place Keeper – ES6 & HTML5

General

Our app consists of three HTML pages:

- **index.html** – the App's home page with navigation links to the two other pages.
- **user-prefs.html** – displays a `<form>` for collecting user preferences. These preferences determine how various parts of the app are displayed.
- **map.html** – displays a list of places saved by the user and a map.

Try to incorporate the new HTML 5 and ES6 features you learned today

- semantic HTML 5 elements such as `<section>`, `<nav>`, `<main>` etc.
- The new JavaScript ES6 features – destructuring, arrow functions, default parameter values, `let`, `const`, etc.

Use the MVC pattern to keep your code organized and its components clearly defined. You should have the following services:

- **utilService** – *general utility functions.*
- **storageService** – *handles reading from and writing to local storage*
- **userService** – *manages saving and reading the user's preferences*
- **placeService** – *manages the place entity CRUDL*

This exercise involves self-learning and handling new documentation that we haven't met yet. Self-learning new technologies is a big part of becoming a professional programmer!

index.html

This is a simple home page:

- Use the stored user preferences to greet the user by his or her name.
- Use the stored user preferences to set the font and background colors.
- Add navigation links to the two other pages.

user-settings.html

The image shows a user settings form with the following elements:

- First Name:** A text input field.
- Background Color:** A color picker input field.
- Text Color:** A color picker input field.
- Zoom Factor:** A slider input field with a value of 16 displayed.
- Map Start Location:** A text input field.
- Save:** A blue button at the bottom of the form.

This page contains a `<form>` for collecting user preferences and saving them to `localStorage`. Save the preferences only when the form is submitted.

TIP: you will need to call `event.preventDefault()` in the `onsubmit` event handler.

The form should collect the following data:

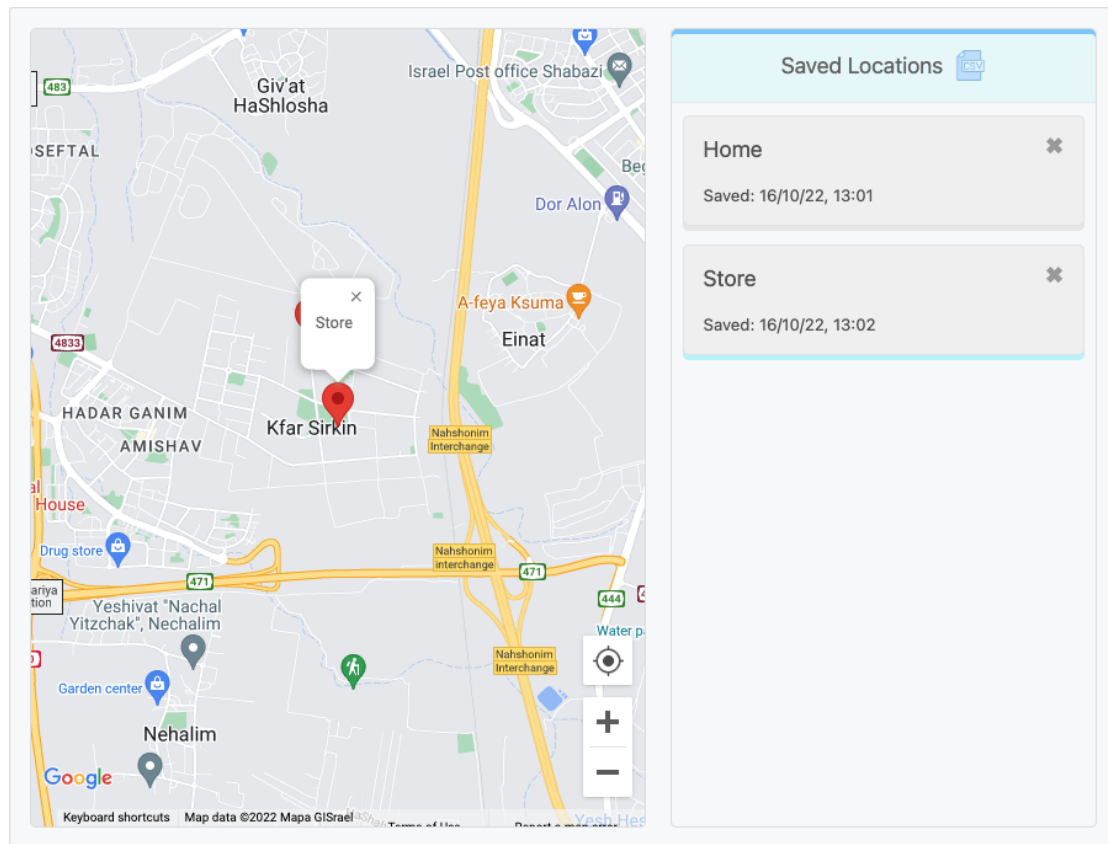
- Name – a regular `<input>`. Required.
- Background color – a color picker implemented with `<input type="color">`
- Font color – another color picker
- Map Initial zoom factor – a slider with a range of 1 – 21
- Map starting location – an `<input>` where the user can enter comma separated lat & lng values.

This page should also:

- Use the stored user preferences to set the page colors.
- Display a navigation link back to the home page.

map.html

This page has two components – a list of saved places and a map. It looks something like this:




To be able to use Google Maps, you will need to create an API key. Detailed instructions on how to do this are provided [here](#).

Here are some steps you may wish to follow in implementing this page:

- Obtain an API key according to the provided instructions.
- Use the examples from this morning's lesson and the Google Maps documentation to get a basic map to render on the page.
- Start controlling how the map is displayed –
 - Set an initial hard coded location by passing lat & lng values to it. Something like: `map.setCenter(new google.maps.LatLng(45,19))`
 - Set an initial hard coded value of 16 for the zoom factor.
- Connect an event handler for handling click events on the map (this isn't a regular DOM event – find out more in the Google Maps documentation).

- Make the click handler prompt the user for a location name and create an object with the name and coordinates. Use the `placeService` to add and save it. The object might look something like this:


```
{
  id: 123,
  lat: 32.1416,
  lng: 34.831213,
  name: 'Home'
}
```
- Start implementing the second component – the list of saved places. Populate the list using the `placeService` and support deleting places from the list.
- Start connecting the map with the list – when the user clicks a place in the list, center the map on that place’s coordinates.
- Add a  button (*search for "my location.png" in google*) and place it on the map. When the user clicks it, retrieve his current location, and center the map on it. *Note that the location information obtained by your browser is not always accurate.*
- Use the `userService` to retrieve the initial zoom factor and map location and use these values to initialize the map.
- Add markers to the map – each of the saved places should be shown on the map by a marker. Keep the list of markers synchronized with the list of places through all of the CRUD operations. Note that for technical reasons, Google Maps markers can not be saved to local storage, so you will need to manage them separately.
- As with the other two pages – use the stored user preferences to set the page colors.

Bonuses

- Implement the map starting location (in **`user-settings.html`**) using a `<datalist>` instead of a simple `<input>`. The datalist should include the following options:
 - o Current location
 - o First saved location
 - o Random saved location
 - o Last selected saved location
 - o The user can still enter comma separated lat & lng values into the datalist
- Replace the prompt with a nice modal
- Let the user download a CSV of the places
- Add a custom validation In the user preferences form – when the user enters coordinates in the map starting location `<datalist>`, make sure they are two valid coordinates separated by a comma, between 0 – 90 each.
- Create more pages and try out some HTML5 features we have covered