COM 642 | Assignment 1 Technical Log

Harrison Lingren, Fall 2017

To view this document as a webpage, click here.

Overview

In this document I will outline the design and functionality implemented in my assignment 1 submission.

My design is a simple 3-view web application which allows users to:

1. Register and book accommodations for a trip
2. Search for and view existing bookings
3. Update or cancel their booking

To accomplish this, I used simple AJAX calls to fetch and send data, and manipulated the DOM with jQuery to display the information.

Persistent data via REST API

Although it is outside the scope of this assignment, it should be noted that I wrote a simple REST API to allow for persistent storage of booking records in a database. I will not detail the implementation of this API here, but I will describe its behavior when discussing the AJAX functions that were used.

Code for the REST API can be made available upon request.

Design & structure

HTML overview

I used a single-page design to create this application. All the markup code is contained within `jquery_form.html` in the root directory.

The `<head>` section links to my CSS stylesheet & external fonts that were used, and also to `jQuery`, `spin.js` for loading animations, and `jQuery UI` for a `tooltips` and`spinner` and`selectmenu` elements.

Within `<body>`, I use a container `<div>` to hold the primary app contents. Inside that, I separate the 3 primary application views into `<div>` elements with the following classes:

- `.search-form`
- `.booking-form`
- `.success-page`

Each view is separated into 3 sub-container classes: `.section-header`, `.section-wrapper`, and `.section-footer`.

Search form

The search view contains a search input & button. When a user enters a booking reference ID number and clicks the search button, a client side function validates the input and submits the query if the input is valid - otherwise, it alerts the user.

If the ID is found, it switches to the `.booking-form` view with the user's registered information.

Booking form

The booking form contains the following inputs:

- First & last name
- Email
- Number of adults & children
- Destination location

The submit button calls a client-side function which validates the information, and then submits to the API if valid or alerts the user if revision is needed.

If the registration is successful, the view changes to the `.success-page` and displays the data for review. If the user is viewing their existing registered information, a cancellation link is displayed that allows them to delete their booking. This link also links to the success page when clicked.

Success page

The success page simply lists the user's registered info once a booking has been created or updated. Additionally, it adapts as a cancellation page if an existing booking was cancelled.

This page includes a link at the bottom to start over, which returns the user to the `.booking-form`.

Style & CSS

Typography

For this assignment I used Google's Open Sans and Material Iconsfonts.

Design

I created a very simple design with muted solid colors, rounded corners, and wider spacing for this application.

The `.container` is centered horizontally in the document, with large margins on both sides. The background of the `<body>` is a light muted blue, and the container is white. `<input>` fields have

dark grey rounded borders and generous padding.

Functions & application logic

Below are descriptions of the client-side functions that were implemented as part of this assignment. These include the DOM manipulation methods, and AJAX / data handler methods.

DOM manipulation

There are a couple helper functions implemented, which are listed below. However, most of the DOM manipulation occurs in the `$(document).ready()` method.

showTooltip

Parameters:

- `target` - the element to display the tooltip next to
- `content` - the content to use in the tooltip

Description:

Creates a message using the jQuery UI widget, tooltip. Adds the tooltip next to the `target` element using `content`.

removeTooltip

Parameters:

- `target` - the element with the tooltip that is to be removed

Description:

Removes a previously created tooltip from the `target` element.

showSpinner

Parameters:

none

Description:

Shows the loading spinner element. This was generated using spin.js.

hideSpinner

Parameters:

none

Description:

Hides the loading spinner element.

$(document).ready

Most event handlers and DOM initialization is handled within this method. Below are listed the primary parts:

Initialize page view

```
234   // hide booking form and success page
235   $('.booking-form').hide();
236   $('.cancel-prompt').hide();
237   $('.success-page').hide();
238   $('.spinner').hide();
```

Hides all content except the `.search-form` view.

Event handlers

`.search-link` and header title:

```
241   $('.search-link, header>h2').click(() => {...})
```

Changes view to `.search-form`.

`.booking-link`:

```
248   $('.booking-link').click(() => {...})
```

Resets all values on `.booking-form` and switches the view to it.

`#submit-btn`:

```
276   let submitBtn = $('#submit-btn');
277   submitBtn.click(() => {...})
```

Validates input fields on `.booking-form`, and passes values into an object to send to `submitForm()` if they are valid. Otherwise, displays a tooltip to alert the user.

`#search-btn`:

```
308   let searchBtn = $('#search-btn');
```

```
309   searchBtn.click(() => {...})
```

Validates `.search-input`. If valid, passes value through `parseInt()` and sends to `findByRef()`. Otherwise, displays a tooltip to alert the user.

AJAX methods

There are two different uses of AJAX in this application. Below are descriptions of their behavior:

The first loads a set of location objects from the file `destination.json`, and uses this data to populate the destinations `<select>` menu element on the booking form.

The second interacts with the aforementioned REST API to create, receive, update, and delete booking records.

buildLocationMenu

Parameters:

- `target` - the menu element to construct

Loads the data from `destinations.json` and inserts it into the `target` menu. It uses jQuery's `$.getJSON()` to load the data, and then iterates on it to build the menu.

findByRef

Parameters:

- `id` - the booking reference ID to search for

Description:

Constructs the query URL using parameter `id`, then submits an `$.ajax()` call to `GET` booking data in JSON format. If successful, pases the data to `handleSearchSuccess()`. Otherwise, a tooltip is displayed to alert the user of the error.

If the booking was found, the ID is stored in a hidden input field for later use in `submitForm()`.

submitForm

Parameters:

- `booking` - the data from the form
- `oldId` (nullable) - the reference ID of the booking to update

Description:

Creates or updates a record based on the value of `updateFlag`. This function uses one of two `$.ajax` calls:

If the edit is a creation, it gathers the form data and sends a `POST`request to the API query URL.

If the edit is an update, it gathers the form data and sends a `PUT`request to the API query URL, using the ID value passed in with `oldId`.

In both cases, the successful response data from the AJAX call is passed into `handleEditSuccess()`. Otherwise, a tooltip is displayed to alert the user of the error.

cancelBooking

Parameters:

- `id` - the reference ID of the booking to be deleted

Description:

Deletes a booking record with reference ID = `id`. Sends a `DELETE`request using the `$.ajax` method to the API query URL. If successful, passes the response JSON data to `handleDeleteSuccess()`.

handleSearchSuccess

Parameters:

- `result` - xmlhttprequest object
- `status` - status of the AJAX call
- `resp` - response data

Description:

Using the JSON data in `resp`, this method populates the input fields on `.booking-form` so that the user can view their booking. The header and footer text is also adapted to fit the "Update" workflow so that the user knows they are editing (rather than creating) a booking.

handleEditSuccess

Parameters:

- `result` - xmlhttprequest object
- `status` - status of the AJAX call
- `resp` - response data

Description:

If the request was successful (`resp.status == 201`), this method displays the `.success-page` view with the user's registered booking information using the JSON data from `resp`.

If the request was not successful, it shows a tooltip alerting the user and does not display the `.success-page` view.

handleDeleteSuccess

Parameters:

- `result` - xmlhttprequest object
- `status` - status of the AJAX call
- `resp` - response data

Description:

Using the ID value that was previously stored in the DOM, this method displays the `.success-page` view and amends it to inform the user of their successful cancellation.