COMP 426

Assignment 5

CONTENTS

This is the second part of a two-part assignment in which we are building a dynamic Hero Database web page that displays a list of heroes from the Avengers comics and allows the user to make changes to the hero entries. In a04, you designed the HTML/CSS/JavaScript to dynamically generate, display, and style the list of heroes. You also created a dynamically generated HTML form for editing a given hero object.

In this assignment, we will be continuing where we left off from a04. **You must have already completed a04 before you can start this assignment.** The goal of this assignment is to write the JavaScript necessary to respond to user-initiated events such as clicking the "edit" button for a certain hero and saving an edited hero entry.

1. Getting started

1. Open Visual Studio Code and navigate to the a05 project directory.
2. Copy `a04/index.html` and `a04/package.json` from a04 into the a05 project directory. Also copy any css files or any other files you were using into a05. **DO NOT copy your `a04/render.js` file into `a05/` because this will overwrite `a05/render.js`.**

3. Open a terminal in the a05 project directory and run the command `npm install` to reinstall the project dependencies.
4. Open `a04/render.js` and `a05/render.js`.
5. In the a05 version of render.js, you should see `TODO` comments for `renderHeroCard()` and `renderHeroEditForm()`. Copy your code directly from `a04/render.js` for these functions.
6. In the a05 version of `loadHeroesIntoDOM()`, you should see five `TODO` comments. The first two tell you to generate the heroes using `renderHeroCard()` and append them to the `$root` element. Copy your code from `a04/render.js` to perform these actions. **DO NOT copy the code from a04 that generated a random hero and inserted an edit form in the DOM**. We're not using that code anymore.
7. Run `browser-sync -sw` to start the Browsersync local development server. This should automatically open a new browser tab showing your complete Hero Database web page from a04, with the exception of the edit form (which has now been removed).

> If the Hero Database page you see is not the same as it was for a04, this likely means you didn't copy all the required dependent files or code from a04. Double check both directories/files and look for differences.

2. Summary

Just like with a04, the majority of your code for this assignment will go in render.js. You'll notice there are three newly-defined event handler functions for you to implement:

1. `handleEditButtonPress(event)` - Handles a JavaScript event representing a user clicking on the "edit" button for a particular hero. The job of this function is to (1) identify which hero was clicked on to be edited, (2) use `renderHeroEditForm()` to render the edit form for that hero, and (3) replace the hero's display card in the DOM with the rendered hero edit form.
2. `handleCancelButtonPress(event)` - Handles a JavaScript event representing a user clicking on the "cancel" button while editing a particular hero. The job of this function is to (1) identify which hero was clicked on to cancel editing, (2) use `renderHeroCard()` to render the display card for that hero, and (3) replace the hero's edit form in the DOM with the rendered display card.
3. `handleEditFormSubmit(event)` - Handles a JavaScript event representing a user submitting the edit form for a particular hero. The job of this function is to (1) identify which

hero was clicked on to cancel editing, (2) use the values from the HTML form to update the hero object's values, (3) use `renderHeroCard()` to render the updated display card for that hero, and (4) replace the hero's edit form in the DOM with the rendered display card.

> Just like with a04, the structure of render.js was specifically designed as a good example of how you should try to organize your web JavaScript code. In particular, note the use of explicitly named event handler functions. This is a good practice that enforces separation of concerns.

After implementing these methods, the final change you will have to make is in the `loadHeroesIntoDom()` function. Recall from a04 that this function will be executed automatically as soon as the page loads. To complete this assignment, you need to use jQuery to register `handleEditButtonPress`, `handleCancelButtonPress`, and `handleEditFormSubmit` as event handlers on the DOM. They should be registered as listeners for the correct events (i.e. button press events and/or form submission events). See jQuery.on() for more info on how to register event listener functions with jQuery.

3. Requirements

`handleEditButtonPress()`

The following are **bare minimum** requirements for the `handleEditButtonPress()` function:

- The function must use the `event` parameter to determine which hero should be edited
- The function should call `renderHeroEditForm()` to render the edit form for that hero
- The function should **remove** the hero's display card from the DOM
- The function should **insert** the rendered edit form for the hero into the DOM

`handleCancelButtonPress()`

The following are **bare minimum** requirements for the `handleCancelButtonPress()` function:

- The function must use the `event` parameter to determine which hero should be canceled
- The function should call `renderHeroCard()` to render the display card for that hero
- The function should **remove** the hero's edit form card from the DOM
- The function should **insert** the rendered display card for the hero into the DOM

- No changes should be made to any of the hero's properties---even if the user had made changes to the properties in the form

`handleEditFormSubmit()`

The following are **bare minimum** requirements for the `handleEditFormSubmit()` function:

- The function must use the `event` parameter to determine which hero should be updated
- The function should use the field values from the hero's edit form to update the hero object's properties
- The function should call `renderHeroCard()` to render the display card for that hero using the hero's updated properties
- The function should **remove** the hero's edit form card from the DOM
- The function should **insert** the rendered display card for the updated hero into the DOM
- The hero object's properties should be updated in the `heroicData` array

`loadHeroesIntoDom()`

The following are **bare minimum** requirements for the `loadHeroesIntoDom()` function:

- The function must call `renderHeroCard()` one time for every hero passed in the `heroes` array parameter.
- The function must append the returned HTML from every call to `renderHeroCard()` to the `$root` jQuery object.
- The function must use jQuery to add `handleEditButtonPress` as an event handler for clicking the edit hero button
- The function must use jQuery to add `handleEditFormSubmit` as an event handler for submitting the edit hero form
- The function must use jQuery to add `handleCancelButtonPress` as an event handler for clicking the cancel editing button

4. Hints

Here are some further details and hints about the assignment:

- Make sure that you can edit a hero, save the changes, and then edit the hero again.

- Identifying which hero is being modified in a click handler is tricky. One option is to save the hero's `id` in an HTML data attribute, and access it through the `event.target` property.
- Another option is to pass the hero object as an extra parameter into the click handler function. This requires you to bind the click handler separately for every button or form element in the DOM.