Assignment 4

Contents

The previous assignment gave you practice writing functions and manipulating data in JavaScript. In lecture, we've been learning how to use jQuery to generate and insert new HTML elements directly into the DOM. Over the next two assignments, you will use these skills to build a dynamically generated, interactive web page that displays a Hero Database of Marvel Avengers heroes and offers users the ability to edit individual hero entries.

This assignment, a04, focuses on dynamically generating and inserting the HTML needed to display the hero entries. In this assignment we will also design a "hero edit form" which will be used later in a05 to allow the user to make updates to hero entries. The high-level goal of a04 is to (1) start with some predefined data stored in an array of JavaScript objects (2) render (aka convert) that data to HTML elements, and (3) insert the HTML into the DOM.

Some starter code has already been provided for this assignment to help you organize your JavaScript code in a clean, reusable way. For autograding to work correctly, you **must** use the code structure provided. However, you are free to design the HTML and CSS portions of the assignment however you want, so feel free to get creative when styling your site! If you feel like you need inspiration for the design of your Hero Database, see Section 3 for screenshots showing how you could design your heroes.

1. Getting started

1. Open Visual Studio Code and take a look at the a04 assignment directory.

2. You should see one HTML file, two linked JavaScript files, and a sub-folder of icons. Open these files and skim their contents. You'll notice that there are no CSS files included, and the HTML file is empty besides a single `<div id="root"></div>` element.

- **icons/** - contains eight different icons, one for each of the eight heroes in the database
- **data.js** - defines an array of hero data that will be used to generate content for index.html
- **index.html** - a basic HTML file with no content except a single empty `<div>`
- **render.js** - this is where you will write code to generate and insert content into index.html

4. Open a terminal in the a04 assignment directory and initialize a new npm project using `npm init`.

5. Use `npm install` to install packages `jquery` (required) and `bulma` (optional) to the project. Refer to a02 if you need to review installing packages with npm.

6. Link the downloaded jQuery library to the index.html file. To do this, find the jQuery `TODO` comment in index.html and add a `<script>` tag with `src` attribute set to `node_modules/jquery/dist/jquery.js`.

7. (Optional) If you installed bulma with npm, you'll also need to link bulma to index.html. Find the bulma `TODO` comment and add a `<link>` to Bulma's CSS code, using `href` path `node_modules/bulma/css/bulma.css`.

8. Run `browser-sync -sw` to start the Browsersync local development server. This should automatically open a new browser tab showing the initial, blank Hero Database page.

   Note: If you get a `command not found` error, this means you have not installed Browsersync as a global npm package. Go back to Section 5.4 in Assignment a02 to see how to install packages globally, and follow the instructions to install Browsersync as a global npm package.

2. Summary

Now that all dependencies have been added, downloaded, and linked to the html file, we're ready to start generating heroic content for the Hero Database. The majority of your code for this assignment will go in render.js, so switch to this file now and take a look at the `TODO` comments. You'll notice the file is roughly split into four "sections".

1. `renderHeroCard(hero)` - Given an object containing data about an Avengers comic hero, this function should render (aka generate) HTML to **display a single hero's information on the page**. The rendered HTML should be returned as a string, an HTMLElement object, or a jQuery object. See Section 4 for minimum requirements of this function.

2. `renderHeroEditForm(hero)` - Given an object containing data about an Avengers comic hero, this function should render HTML to **display a form with fields for editing a single hero's data**. The rendered HTML should be returned as a string, an HTMLElement object, or a jQuery object. See Section 4 for minimum requirements of this function.

3. `loadHeroesIntoDom(heroes)` - Given an array of hero objects, this function calls `renderHeroCard()` and `renderHeroEditForm()` to **render each hero as HTML and load it into the DOM**. See Section 4 for minimum requirements of this function.

4. `$(function() { ... })` - The code inside this anonymous function will automatically be executed by jQuery as soon as the page has loaded. This section does not need to be edited at all, as it essentially just calls the `loadHeroesIntoDom()` function.

   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 html rendering functions, each of which takes as input a simple object data, and produces as output a rendered HTML fragment. Then, a different function is in charge of *using* these functions to actually create HTML and insert it in the DOM. This is a good practice that enforces separation of concerns.

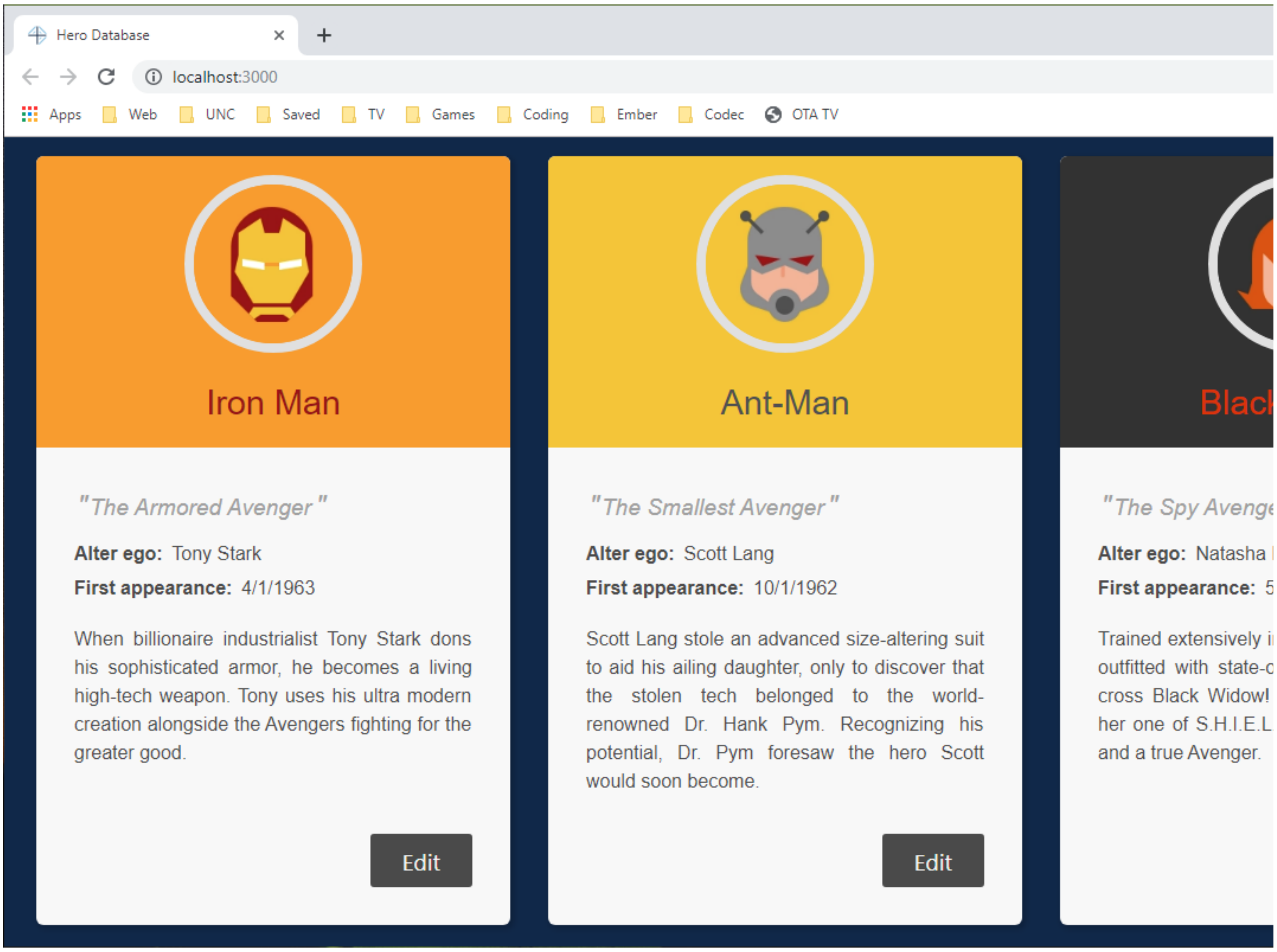
For this assignment, the `loadHeroesIntoDom()` function will be executed automatically as soon as the page loads. `loadHeroesIntoDom()` is responsible for looping through all the heroes in the database, calling `renderHeroCard()` on each hero object, and inserting the resulting HTML into the DOM.

Additionally, `loadHeroesIntoDom()` should also select a hero at random from the list, use `renderHeroEditForm()` to generate a "hero edit form" for that hero, and insert the resulting HTML into the DOM. The code to randomly select a hero from the list has already been written for you. **Note: this part of the `loadHeroesIntoDom()` function is temporary, and will be changed in a05**. It is only included in a04 so that you can see what your hero edit form looks like.

3. Screenshots

Here are some screenshots showing how you *might* choose to style your hero database.

Keep in mind that **these are only suggestions**! Feel free to get as creative as you want!



Iron Man

"The Armored Avenger"

Alter ego: Tony Stark

First appearance: 4/1/1963

When billionaire industrialist Tony Stark dons his sophisticated armor, he becomes a living high-tech weapon. Tony uses his ultra modern creation alongside the Avengers fighting for the greater good.

Ant-Man

"The Smallest Avenger"

Alter ego: Scott Lang

First appearance: 10/1/1962

Scott Lang stole an advanced size-altering suit to aid his ailing daughter, only to discover that the stolen tech belonged to the world-renowned Dr. Hank Pym. Recognizing his potential, Dr. Pym foresaw the hero Scott would soon become.

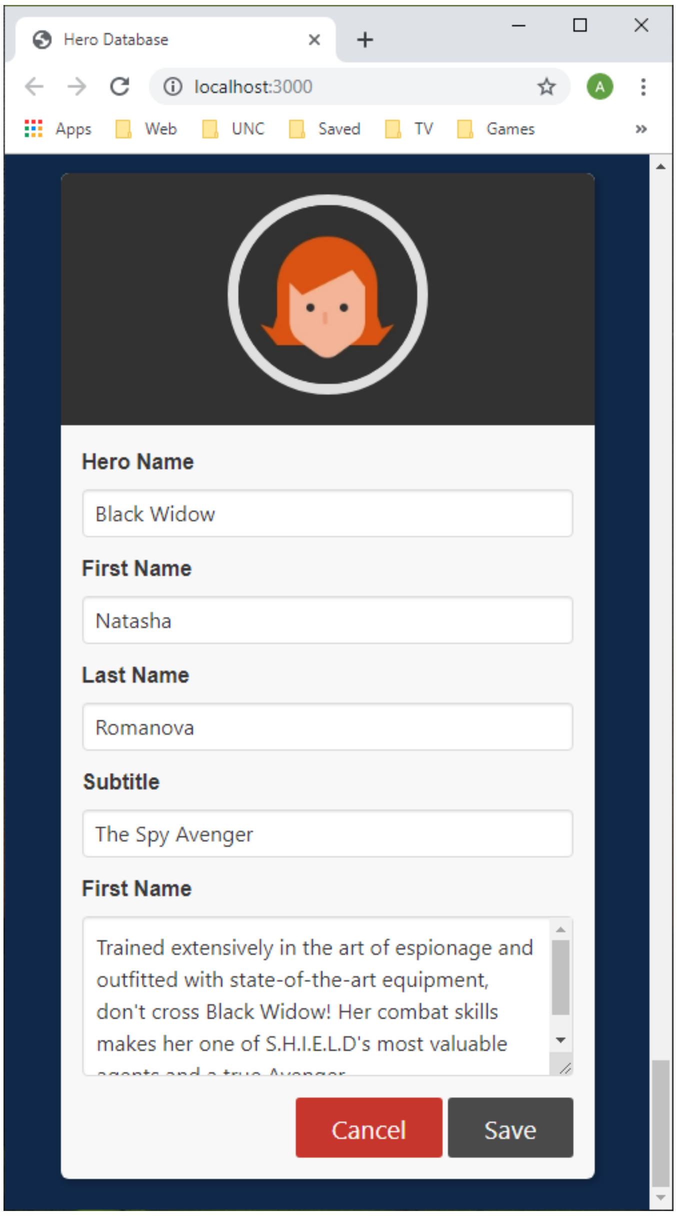
Black

"The Spy Avenge

Alter ego: Natasha

First appearance: 5

Trained extensively i outfitted with state-c cross Black Widow! her one of S.H.I.E.L and a true Avenger.



Hero Name

Black Widow

First Name

Natasha

Last Name

Romanova

Subtitle

The Spy Avenger

First Name

Trained extensively in the art of espionage and outfitted with state-of-the-art equipment, don't cross Black Widow! Her combat skills makes her one of S.H.I.E.L.D's most valuable

4. Requirements

`renderHeroCard()`

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

- The function must return valid HTML, either as a string, an HTMLElement object, or a jQuery object.
- The returned HTML must display *at minimum* the following properties from the hero object parameter: ***`hero.first`*** *- the first name of the hero* **`hero.last`** - the last name of the hero ***`hero.name`*** *- the superhero name of the hero* **`hero.img`** - the hero image url, which must be displayed on the page via an `<img>` HTML tag or `background-image` inline CSS property ***`hero.color`*** *- a foreground color matching the theme of the hero, which must be used as part of an inline CSS property* **`hero.backgroundColor`** - a background color matching the theme of the hero, which must be used as part of an inline CSS property ***`hero.description`*** *- A short paragraph description of the hero* **`hero.firstSeen`** - The month and year of the first comic book issue in which the hero appears, stored as a JavaScript Date object
- The returned HTML must include *at minimum* a `<div>`, a `<p>`, and a `<span>` tag.
- The returned HTML must additionally include an "edit" `<button>` element which will be used in a05 to edit the hero by showing the hero edit form (for now, the button should do nothing when clicked).
- The function must **not** modify the DOM in any way, shape, or form.

`renderHeroEditForm()`

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

- The function must return valid HTML, either as a string, an HTMLElement object, or a jQuery object.
- The returned HTML must include *at minimum* **form fields** (e.g. `<input>`, `<textarea>`, etc.) for editing the following hero properties: ***`hero.first`*** *- the first name of the hero* **`hero.last`** - the last name of the hero ***`hero.name`*** *- the superhero name of the hero* **`hero.description`** - A short paragraph description of the hero * **`hero.firstSeen`** - The month and year of the first comic book issue in which the hero appears, stored as a JavaScript Date object
- By default, the form fields should initially be pre-populated with the field values taken from the hero object parameter.
- The returned HTML must include *at minimum* a `<form>` tag containing *at least* three `<input>` tags and a `<textarea>` tag.
- The returned HTML must additionally include a "save" `<button>` element which will be used in a05 to save changes to the hero (for now, the button should do nothing when clicked).
- The returned HTML must also include a "cancel" `<button>` element which will be used in a05 to cancel making changes to the hero without saving (for now, the button should do nothing when clicked).
- The function must **not** modify the DOM in any way, shape, or form.

`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 `$root` jQuery object.
- The function must call `renderHeroEditForm()` exactly once on the randomly selected `randomHero` object.
- The function must append the returned HTML from `renderHeroEditForm()` to the `$root` jQuery object.

Hints

Here are some further details about the assignment:

- You are free to create any other files in your project you deem necessary, including `.css` files for adding custom styles.
- If you included bulma as a dependency, use bulma classes to make your page look even better!
- Feel free to add any supporting HTML in index.html and/or in the root element.
- In `renderHeroEditForm()`, you must provide a way to edit the `lastSeen` property, which is a date. There are lots of ways to edit dates in HTML, including the input date tag or select tags.