Springboard

API Practice: TV Maze»

apis-tvmaze-starter-code.zip 2.5KB

In this exercise, you'll finish a partially-complete web app which is a front-end for the TVMaze API.

Step 1: Understand The API»

Explore the TVMaze API. You will need to make a get request to two endpoints:

http://api.tvmaze.com/search/shows?q=<search query>

and

http://api.tvmaze.com/shows/<show id>/episodes

Use a tool like curl or insomnia to make a HTTP request to both endpoints and get comfortable with the JSON that is returned. You will need to parse the JSON in order to get the data for the application.

Step 2: Understand Current Code

We've provided two files for you:

tvmaze.htmlAll the HTML for the application. You should be able to complete all of the pages for this exercise without having to change anything in this file.***tvmaze.js***Starter JavaScript for the application.

Right now, the application has this feature:

- You can type part of a TV show title into the search form and, on submission, it will return information about a hard coded show. It will show a series of cards with information on the show.

Note how we've structured this code:

- by having a separate function for *searchShows*, you can test the "search API for shows" without having to do deal with anything related to the DOM in tests. This also makes this function re-usable if we built a different version of this app with a different front-end (like in React); we could re-use *searchShows*.

  Play with this function in the console and get a sense for how it works.
- *populateShows* deals just with inserting the passed-in shows into the DOM. This makes this testable without having to have it tied to the code that gets data from the API.
- our *handleSearch* event handler ties the two together: it gets the search term, gets the shows using *searchShows*, and fills in the DOM with *populateShows*.

> Notice the data attribute!
>
> A particularly important thing to note in reading our code is how we used a "data attribute".
>
> Data attributes are very useful for when you want to associate some data (in this case, the show ID) in the DOM, so you can recall it later.
>
> Note in *populateShows* how we add *data-show-id* onto the outermost *.Show* div. Later, when we want to retrieve which show ID was clicked on, we'll be able to get that show ID.
>
> You may find it helpful to skim the MDN article linked above.

Step 3: Make AJAX request for Search

Remove the hard coded array from the *searchShows* function and make replace the code with an AJAX request to the search shows api from TVMaze. Make sure that the array of information you return from the function is formatted as described in the comments for the *searchShows* function.

Step 4: Add Show Images

The TVMaze API includes images for many shows, but we don't currently use these.

Explore the docs for the TVMaze API and find how we'd extract an image in the *searchShows* function. Add this image to the result object returned by this function.

Update *populateShows* to show the image. You can do this with the following snippet of HTML, inside the *.card* div:

```
<img class="card-img-top" src="/path/to/image">
```

Be careful how you implement this. Not all shows have images, and if you're not careful, this will break for shows without images. Make sure that you write this in a way where shows without missing images won't break your site.

For shows without an image, you can have it show this generic image instead: https://tinyurl.com/tv-missing

Step 5: Add Episode Lists

We want to add a feature where clicking an "Episodes" button at the bottom of a show card shows the episodes for this show at the bottom of the page.

- First, implement the *getEpisodes* function, which is given a show ID. It should return an array of objects with basic information on the episodes for that show, like:

```
[ {id: 1234, name: "Pilot", season: "1", number: "1"}, {id: 3434, name: "In the Beginning", season: "1", number: "2"}, /* and so on... */ ]
```

- To do this, you'll need to read how to get episode data from TVMaze API.
- Next, write a function, *populateEpisodes*, which is provided an array of episodes info, and populates that into the *#episodes-list* part of the DOM.

  The episodes list is a simple *<ul>*, and the individual episodes can just be basic *<li>* elements, like `<li>Pilot (season 1, number 1)</li>`.

  (Also, now that we have episodes, you'll need to reveal the *#episodes-area*, which is initially hidden!)
- Add an "Episodes" button at the bottom of each show card
- Add a click handler that listens for clicks on those buttons.
  - You'll need to make sure this eventlistener works even though the shows won't be present in the initial DOM
  - You'll need to get the show ID of the show for the button you clicked. To do this, you can read about getting data attributes with jQuery and also how to use jQuery to find something a few levels up in the DOM
  - Then, this should use your and functions.

    getEpisodes

    populateEpisodes

Make sure you put thought into good variable names and code style for these, and write comments!

Further Study

There are a lot of other things you could do here:

- **Write tests** for your functions. Practice writing software tests in a great way to get better at this critical developer skill!
- **Add other information/features from TVMaze**. There are other things you can get from TV Maze—you could list the actors in a show, or the genres of a show, or other things.
- **Make the episodes into a Bootstrap modal**. If you want to learn more about the components of Bootstrap, you could change you code so that it shows the list of episodes in a pop-up modal, rather than a list at the bottom of the page.

  If you wrote the functions in part 3 well, you should be able to this only by having to change the *populateEpisodes* function, but not other parts of your JavaScript — a nice reward for breaking your code thoughtfully into good functions!

Solution»

See our solution