Scene It App

Part 2: Creating a Watchlist with localStorage

Your task :

If you've finished part 1, your Scene It app should be showing movies to the screen when you submit the search form. Your task for part 2 is to implement a feature that allows users to save movies to their "watch list".

When you're done, users will be able to press the "add" button on a selected movie, and then later click on "Go to my watchlist" to see the movies that they've saved.

Since we're not actually using the OMDB API until part 3, for now there's a file called *data.js* that contains some movie data. The search bar will "work", but it will always show the list of movies in *data.js,* no matter what you enter into the search bar.

Let's do it!

Step 1 - Set up a click listener on the movies

Now that there are movies showing up underneath the search bar, we can make them clickable. We'll have to attach a click listener to the "movies-container" div.

1) Revisit your **`renderMovies()`** function
2) Look in your template literal strings for wherever you're rendering an "add movie" button.
3) Give them a class of `add-button`
4) Also give these buttons a **`data-imdbid`** *attribute*.
    a) Inject the IMDB id of the movie as the parameter passed in to **`data-imdbid`** using ${ } notation
    b) Once it has been added to the DOM, it should look like this but with individual movie IDs:
    `data-imdbid="tt1345836"`

c) What is this doing? The ‘data-*’ attributes are a way to store data in your HTML/DOM that does not need to be visible to the user, but we as developers still need access to it. In this instance, we’re storing the IMDb ID, but the user doesn’t need to know that bit.

Next we need to add an event listener to each ‘Add’ button, but we have a minor problem. When the page first loads, none of the buttons are on the page yet, and we can’t attach an event to something that doesn’t exist. In this case, we know the button will *eventually* be on the page, but it isn’t yet. This means that we have to add the listener to something else, and then check if the target of the event is one of the buttons.

5) Inside of your “document ready” block, add a new click event listener to the document. It should look like this:

```
document.addEventListener('click', function(event) {
  // code for document click listener goes here
}
```

6) Next we need to only perform an action if the target of the event is a button with a class of '`add-button`'. Add a condition to check if the event.target.classList (an Array) contains the class ‘`add-button`’. (Make an if statement, and use the `.contains()` array method)
7) Inside this condition, we need to get the IMDb ID from the button and store it in a variable. Let’s call it `movieID`. Here is an example that gets a dog breed from a different button:

   HTML
   ```
   <button data-breed="chihuahua">Dog</button>
   ```

   JavaScript
   ```
   const breed = event.target.dataset.breed
   console.log(breed) // chihuahua
   ```

8) Now after you have the imdb id in a variable, you can pass it through to a new function saveToWatchlist. We haven’t written this function, but we’re about to. Pass it the imdb variable you created earlier as an argument.

   Outside of your “document ready” block, define this new function **`saveToWatchlist`** which should expect to receive a parameter **`movieID`**

9) You can test that each button works by adding a **`console.log(movieID)`** inside

of this function and then clicking each button. You should see the different **`imdbID`**s show up in your developer console.

Step 2- Implement saveToWatchlist

At this point, the user should be able to trigger our **`saveToWatchlist`** function by clicking on the different "Add movie" buttons on the page. For this next step, we'll be making use of the **`localStorage`** object so that we can save the list of movies that the user wants to watch. Later on in step 4, we'll create a new *watchlist.html* file where the user can see all the movies they've saved.

1) saveToWatchlist has a parameter called **`movieID`** which will tell us which movie the user clicked on. This is what we are already logging to the console. We'll use it to sift through movieData.js to find the relevant movie information.
2) In saveToWatchlist, create a variable called **`movie`** which will contain the rest of this movie's data.
3) We'll use the Array prototype method **`find()`** (MDN docs) to pull the rest of the movie data from *data.js*. The 'find' array method will look through an array until the function returns true. It will then return the item that it found.
    a) Create the variable **`movie`** with **`const movie =`**
    b) Set it equal to **`movieData.find();`**
    c) Give the **`find`** method an anonymous function that takes **`currentMovie`** as a parameter
    d) Have this anonymous function **`return currentMovie.imdbID == movieID;`**
    This is the condition that we are looking for. We're effectively telling the `find` method to stop looking once it finds a movie that has the same `imdbID` as the `movieID` that we are looking for.
    e) In the end, you'll have

```
const movie = movieData.find(function(currentMovie){
            return currentMovie.imdbID == movieID;
});
```

Whew! We have only a little bit more work to do! Now that the variable **movie** has the movie information that we want in our watchlist, we need to pull in the watchlist from localStorage, add it to the watchlist, and then save the watchlist *back* to localStorage.

4) In the next line (still inside saveToWatchlist), pull down the watchlist from local

storage

```
let watchlistJSON = localStorage.getItem('watchlist');
```

5) Parse the watchlist with JSON

```
let watchlist = JSON.parse(watchlistJSON);
```

6) Use an if-statement to check if the watchlist is **`null`**
    a) If it is null, set **`watchlist`** to an empty array
    b) Try this on your own! Call for attention if you're having trouble with this one.
7) Push **`movie`** into the watchlist

```
watchlist.push(movie);
```

8) Turn the watchlist back into JSON

```
watchlistJSON = JSON.stringify(watchlist);
```

9) Save the JSONified watchlist back into local storage

```
localStorage.setItem('watchlist', watchlistJSON);
```

And that's it! Now, when you click the add button for any given, you should see it's data saved into local storage under the key "watchlist"!

Step 3- Create the "My Watchlist" page

Now that we know what movies the user wants to watch, we can give them an interface to see what movies they've saved.

1) Somewhere in index.html, add a link that says "Go to my watchlist"
    a) Have that link navigate to "/watchlist.html"
2) Create a new `watchlist.html` file
3) Complete this html in a similar style to index.html (if you're feeling adventurous, try a different layout, like a table)
    a) Have the "Scene It" title at the top
    b) Don't include a search bar this time
    c) Have a "`#movies-container`" `div` to hold the list of saved movies
4) Create a `watchlist.js` file that does the following:
    a) When the page loads, pull the watchlist from localStorage

    i) Use `localStorage.getItem('watchlist');`
b) Renders each movie to "`movies-container`", just like in index.js
    i) You can have the movies render exactly like they did in index.js, or you can switch it up!

You'll find that the above code is almost identical to what you did in Part 1! The only difference is that instead of showing a list of movies from *data.js,* you'll be showing a list of movies from **`localStorage.getItem('watchlist');`**