Phase 1 SA-01 Code Challenge: Flatdango[Compulsory]

Start Assignment

Due No Due Date **Points** 15 **Submitting** a website url

Learning Goals

- Implement a 'mini' web application using JavaScript.

Introduction

For this assessment, you'll be working on Flatdango. Flatiron Movie Theater is open for business! You will be building out an application, Flatdango, that allows a user to purchase movie tickets from the theater.

The instructions below will walk you through the process of ideation and planning your app: deciding on your user interface, planning how the information will be laid out on the page, etc. You should work through all the planning steps before you start doing any coding.

Requirements

For this project, you must:

- Have a well-written README file.
- Fetch data from a local server running JSON DB server.

Pre-requisite Data

You can use this JSON file for your server DB.

db.json

Project Setup

Once you have the plan in place for the application you want to build take the following steps:

- Create a new project folder.
- Create a new GitHub repository (**NB: ENSURE IT IS PRIVATE**).
- Add your TM as a contributor to the project. (**This is only for grading purposes. We promise we won't steal your code**)
- Please make sure you regularly commit to the repository.

Project Guidelines

Your project should conform to the following set of guidelines:

Core Deliverables:

As a user, I can:

1. See the first movie's details, including its **poster, title, runtime, showtime, and available tickets** when the page loads. The number of available tickets will need to be derived by subtracting the number of `tickets_sold` from the theater's `capacity`. You will need to make a GET request to the following endpoint to retrieve the film data:

GET /films/1

Example Response:

```
{
  "id": "1",
  "title": "The Giant Gila Monster",
  "runtime": "108",
  "capacity": 30,
  "showtime": "04:00PM",
  "tickets_sold": 27,
  "description": "A giant lizard terrorizes a rural Texas community and a heroic teenager attempts to destroy the creature.",
  "poster": "https://www.gstatic.com/tv/thumb/v22vodart/2157/p2157_v_v8_ab.jpg"
}
```

2. See a menu of all movies on the left side of the page in the `ul#films` element when the page loads. (_optional_: you can style each film in the list by adding the classes `film item` to each `li` element.) There is a placeholder `li` in the `ul#films` element that is hardcoded in the HTML — feel free to remove that element by editing the HTML file directly, or use JavaScript to remove the placeholder element before populating the list. You will need to make a GET request to the following endpoint to retrieve the film data:

GET /films

Example response:

```
[
  {
    "id": "1",
    "title": "The Giant Gila Monster",
    "runtime": "108",
    "capacity": 30,
    "showtime": "04:00PM",
    "tickets_sold": 27,
    "description": "A giant lizard terrorizes a rural Texas community and a heroic teenager attempts to destroy the creature.",
    "poster": "https://www.gstatic.com/tv/thumb/v22vodart/2157/p2157_v_v8_ab.jpg"
  },
  {
    "id": "2",
      "title": "Manos: The Hands Of Fate",
      "runtime": "118",
      "capacity": 50,
      "showtime": "06:45PM",
      "tickets_sold": 44,
      "description": "A family gets lost on the road and stumbles upon a hidden, underground, devil-worshiping cult led by the fearsome Mast
      "poster": "https://www.gstatic.com/tv/thumb/v22vodart/47781/p47781_v_v8_ac.jpg"
    }
  ]
```

3. Buy a ticket for a movie. After clicking the "Buy Ticket" button, I should see the number of available tickets decreasing on the frontend. I should not be able to buy a ticket if the showing is sold out (if there are 0 tickets available). ****No persistence is needed for this feature****.

Bonus Deliverables

These bonus deliverables are here if you want an extra challenge and won't affect your score. ****Make sure to commit your work to save your progress before attempting the bonus deliverables!****

1. Click on a movie in the menu to replace the currently displayed movie's details with the new movie's details. Note that you may have to make an additional GET request to access the movie's details.

2. When a movie is sold out (when there are no available tickets remaining), indicate that the movie is sold out by changing the button text to "Sold Out". Also update the film item in the `ul#films` menu by adding a class of `sold-out` to the film. For reference, here's what the contents of the `ul#films` element should look like with a sold out film:

```
html
 <li class="film item">(Title of film)</li>
 <li class="sold-out film item">(Title of a sold-out film)</li>
 <li class="film item">(Title of film)</div>
```

Extra Bonus

These extra bonus deliverables involve using `fetch` to update data on the `json-server` backend by using `**POST**`, `**PATCH**`, and `**DELETE**` requests. These are meant for an extra, extra challenge and won't affect your grade. ****Make sure to commit your work to save your progress before attempting the extra bonus deliverables!****

1. When a ticket is purchased, persist the updated number of `tickets_sold` on the server. Remember, the frontend shows the number of available tickets based on the `tickets_sold` and the `capacity`, so only the `tickets_sold` should be updated on the backend when a ticket is purchased. You will need to make a request that follows this structure:

```
PATCH /films/:id
  Request Headers: {
    Content-Type: application/json
  }

  Request Body: {
    "tickets_sold": 28
  }

    Example Response:

{
    "id": "1",
    "title": "The Giant Gila Monster",
    "runtime": "108",
    "capacity": 30,
    "showtime": "04:00PM",
    "tickets_sold": 28,
    "description": "A giant lizard terrorizes a rural Texas community and a heroic teenager attempts to destroy the creature.",
    "poster": "https://www.gstatic.com/tv/thumb/v22vodart/2157/p2157_v_v8_ab.jpg"
  }
```

2.
```
Delete a film from the server. Add a delete button next to each film in the `ul#films` menu. When the button is clicked, remove the film 
```

```
and also delete the film on the server:
```

DELETE /films/:id

Example Response:

```
{}
```
```

**Phase 3 Rubric**

| **Criteria** | **Ratings** | | | | | **Pts** |
|---|---|---|---|---|---|---|
| DOM Manipulation | **5 pts**<br>**5**<br>All of the above, plus completed at least one Advanced Deliverable. | **4 pts**<br>**4**<br>Structured HTML creation code cleanly and in a reusable way, using a semantically correct HTML structure without any unnecessary elements. | **3 pts**<br>**3**<br>Successfully rendered and updated the DOM as described by the Core Deliverables. | **2 pts**<br>**2**<br>Rendered elements to the DOM, but with some errors. | **1 pts**<br>**1**<br>Did not properly render elements to the DOM. | 5 pts |
| Events | **5 pts**<br>**5**<br>All of the above, plus completed at least one Advanced | **4 pts**<br>**4**<br>Structured code in a clean and reusable way, splitting functions, using descriptive names and | **3 pts**<br>**3**<br>Successfully attached event listeners to handle DOM events and met all of the | **2 pts**<br>**2**<br>Attached event listeners, but incompletely or with | **1 pts**<br>**1**<br>Did not attach event listeners to respond to | 5 pts |
```

	Deliverable.	using target properties effectively.	Core Deliverables.	some errors.	events.	
Communication with the Server	**5 pts** **5** All of the above, plus completed at least one Advanced Deliverable.	**4 pts** **4** Code structured in a clean and reusable way, splitting into functions and reusing them where needed, with clear function and variable naming.	**3 pts** **3** Able to perform a GET and a non-GET request successfully. All Core Deliverables met.	**2 pts** **2** Partially able to communicate with the server, but incompletely or with some errors.	**1 pts** **1** Unable to communicate with the server.	5 pts
						Total Points: 15

◂ Previous   Next ▸

Help