Assignment #5: Launch Checklist Form

Using our knowledge of forms, the DOM, and HTTP, the commanders of our favorite space shuttle program asked us to create a quick launch checklist. We have four fields that need to be filled out with vital information: the pilot's name, the co-pilot's name, the fuel levels, and the mass of the cargo.

Our pilot, Chris, and the co-pilot, Blake, have been hard at work securing the cargo and filling the shuttle tank. All we need to do is use validation to ensure that we have all of the info for the space shuttle program and make sure no one prematurely launches the shuttle.

Requirements

Create a Launch Checklist Form for astronauts to fill out in preparation for launch. The form should do the following:

1. Validate the user responses with `preventDefault()` to ensure the following:
   - The user entered something for every field.
   - The user entered text for names and numbers for fuel and cargo levels.
2. With validation, update a list of what is currently ready or notready for the shuttle launch.
3. Indicate what is good or bad about the shuttle and whether it is ready for launch by using the DOM to update CSS styles.
4. Fetch some planetary JSON to update the mission destination with vital facts and figures about where the shuttle is headed.

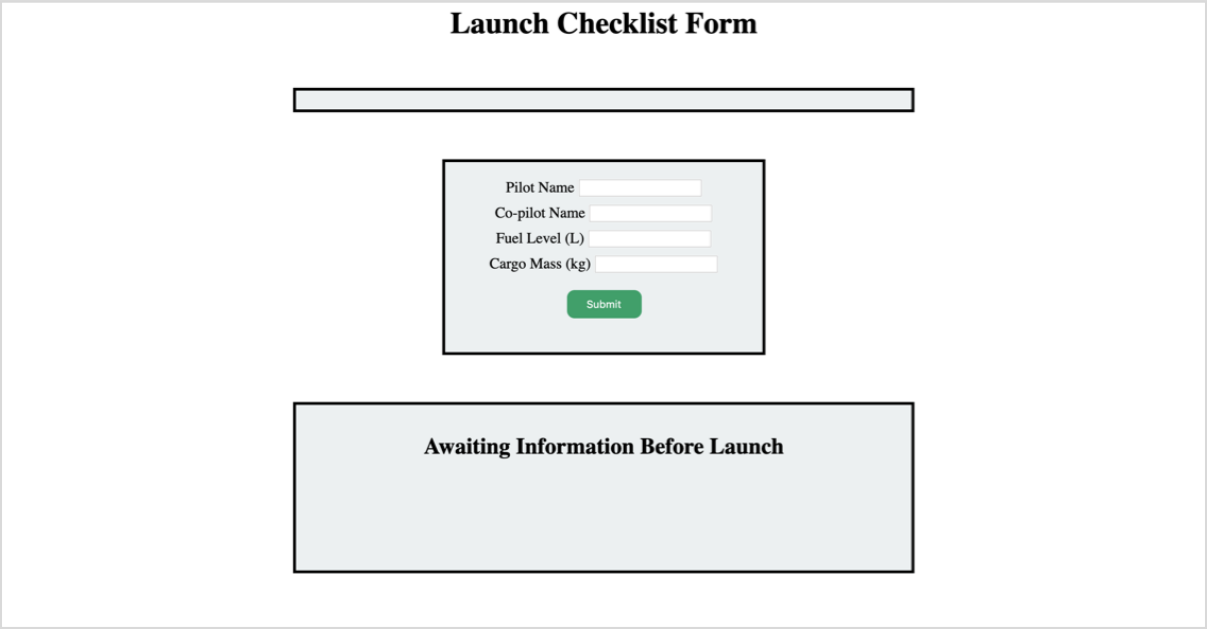
Note: After each new feature is complete, you should commit your code to your local repository. We'll remind you when it's time to commit with the following prompt: ✅ Commit now!

Setting Up Your Project Repository

Fork the repository with the starter code to your personal GitHub profile and clone the repository to the directory where you are keeping your assignments for the class.

To get started, navigate to the directory with your copy of the starter code. Open `index.html` with Firefox to verify that your starter code is working.

When you open `index.html`, you should see the Launch Checklist form with a rectangle above it for the mission destination and a rectangle below it that simply says "Awaiting Information Before Launch".



Adding Validation

Adding Alerts

First, let's add validation to notify the user if they forgot to enter a value for any one of the fields. Fetch each of the four text input nodes and read their values. If any one is empty, alert the user with `window.alert`. (*Hint:* you may want to add `id` attributes to the inputs.)

This process is going to look similar to the validation section in the chapter on forms. Make sure to use `preventDefault()` to cancel form submission.

✅ Commit now!

You also want to make sure that the user entered valid info for each of the fields. Valid information for the fields means that the user submits a value that is easily converted to the correct data type for our fellow engineers. The pilot and co-pilot names should be strings and the fuel level and cargo mass should be numbers. If any of these conditions is not met, alert the user.

Note: If you want to check if something is `NaN`, you cannot use `==` or *`===`. Instead, JavaScript has a built-in method called `isNaN(value)`* *that returns `true` if `value` is `NaN` and `false` if `value` is not* *`NaN`.*

✅ Commit now!

Updating Shuttle Requirements

Once initial validation is complete, we will update the detailed status report. These details are in the `ul` with the id `itemStatus`. Note that this list is initially hidden via the CSS rule `visibility: hidden` in `styles.css`.

After initial validation, make this list visible by changing its visibility to `visible`. Then, using template literals, update the `li` elements `pilotStatus` and `copilotStatus` to include the pilot's name and the co-pilot's name.

If the user submits a fuel level that is too low (less than 10,000 liters), change the fuel status to reflect that there is not enough fuel for the journey. The text of the `h2` element, `launchStatus`, should also change to "Shuttle not ready for launch" and its `color` should change to red. For an acceptable fuel level, report that “Fuel level check passed” in the `itemStatus` list.

If the user submits a cargo mass that is too large (more than 10,000 kilograms), change the cargo status to reflect that that there is too much mass for the shuttle to take off. The text of `launchStatus` should also change to "Shuttle not ready for launch" and the `color` should change to red. For an acceptable cargo mass, report that “Cargo mass check passed” in the `itemStatus` list.

If the shuttle is ready to launch, change the text of `launchStatus` to green and display "Shuttle is ready for launch".

✅ Commit now!

Fetching Planetary Data

Finally, we need some JSON to fill in the crew on the mission destination. Our planetary data can be found in JSON format.

On page load, use `fetch` to retrieve this data. Then, randomly select one of the planets to display.

In `script.js`, we have a block of code commented out at the top. Use this format within a template literal to display the chosen planet’s data in the `missionTarget` div.

✅ Commit now!

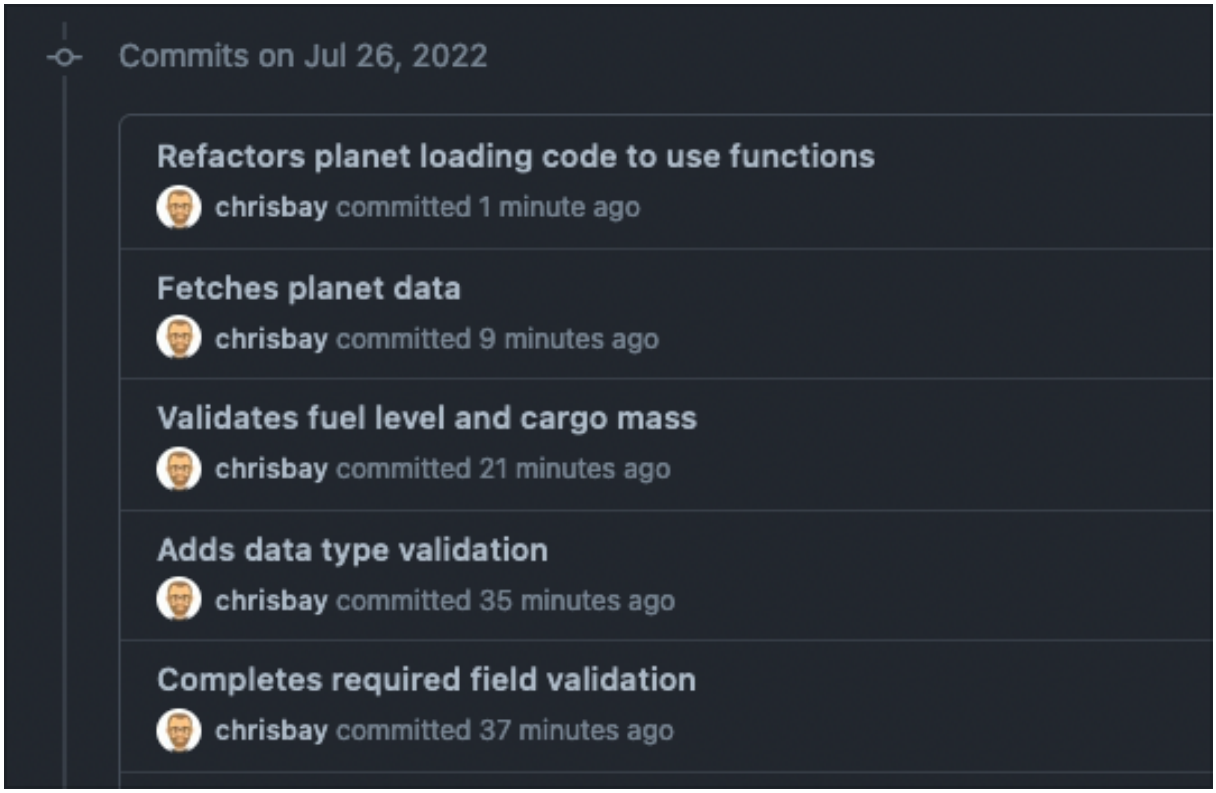
Refactor

Now that you've completed all requirements, spend some time refacotring your code. Look out for:

- any unused code or comments left over from debugging
- opportunities to DRY out your code
- opportunities to improve variable, function, and parameter names
- opportunities to use functions to better organize your code and make it more readable.

✅ Commit now!

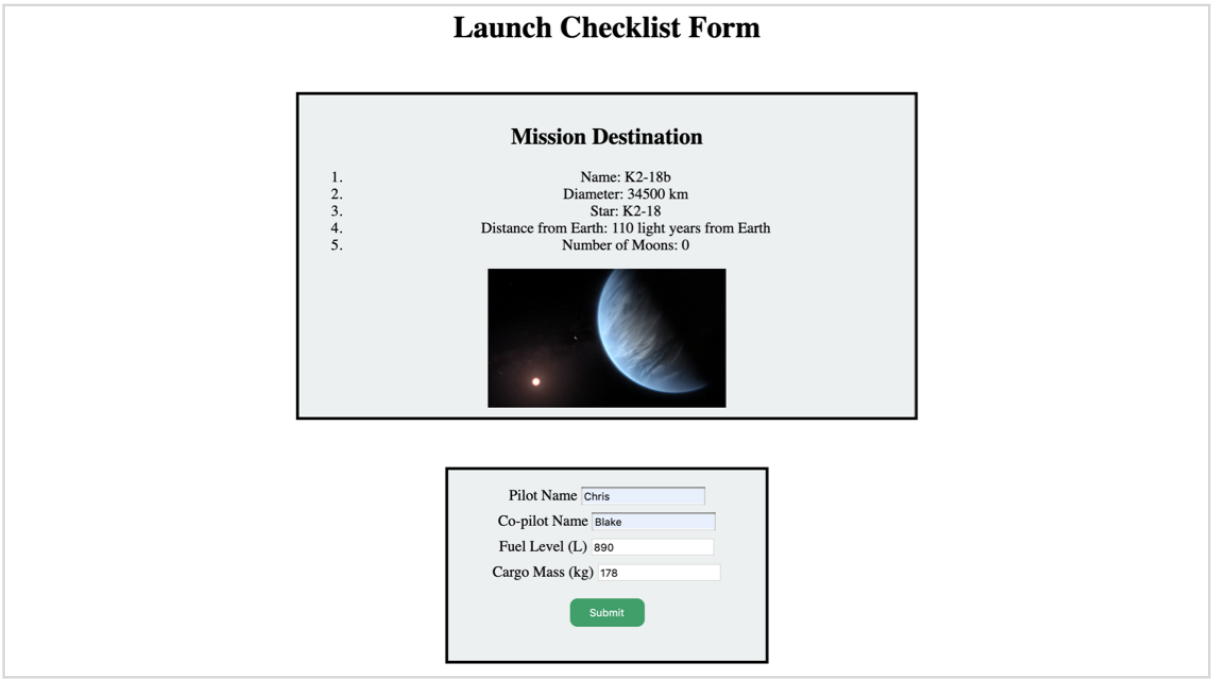
If you've been regularly committing, and using descriptive commit messages, your commit history on GitHub will be clean, readable, and professional!



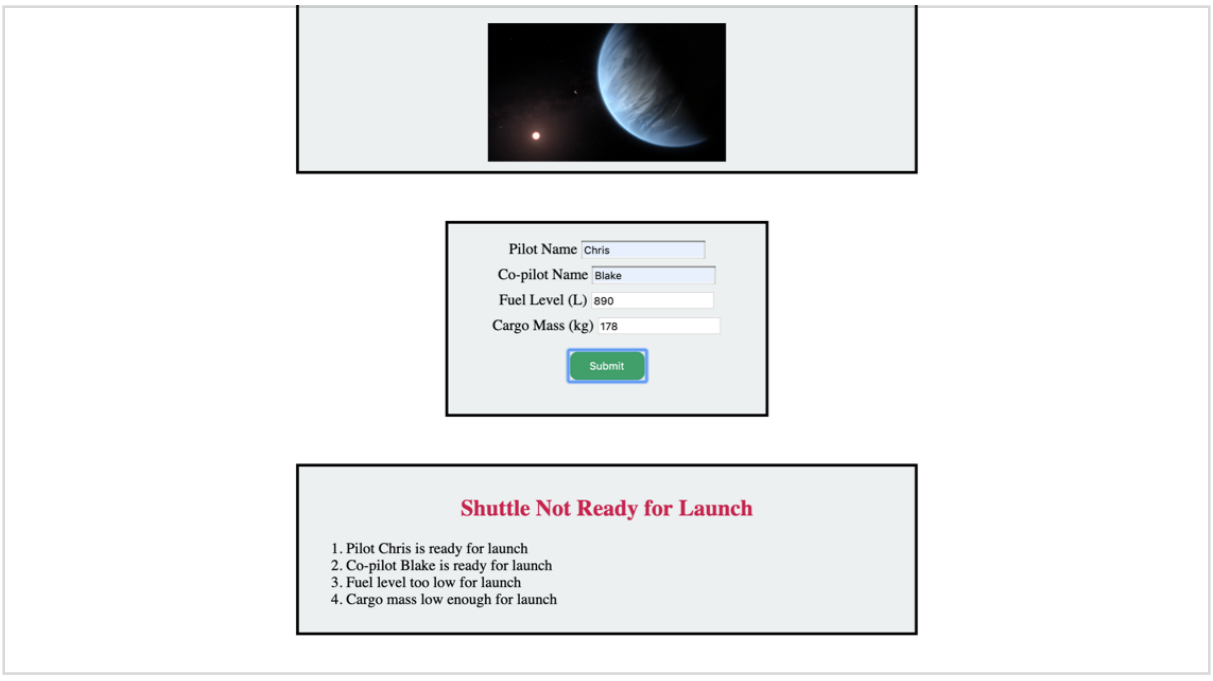
The End Result

After you implement everything, the following form submission would result in the proper updates to `launchStatus` and the `itemStatus`

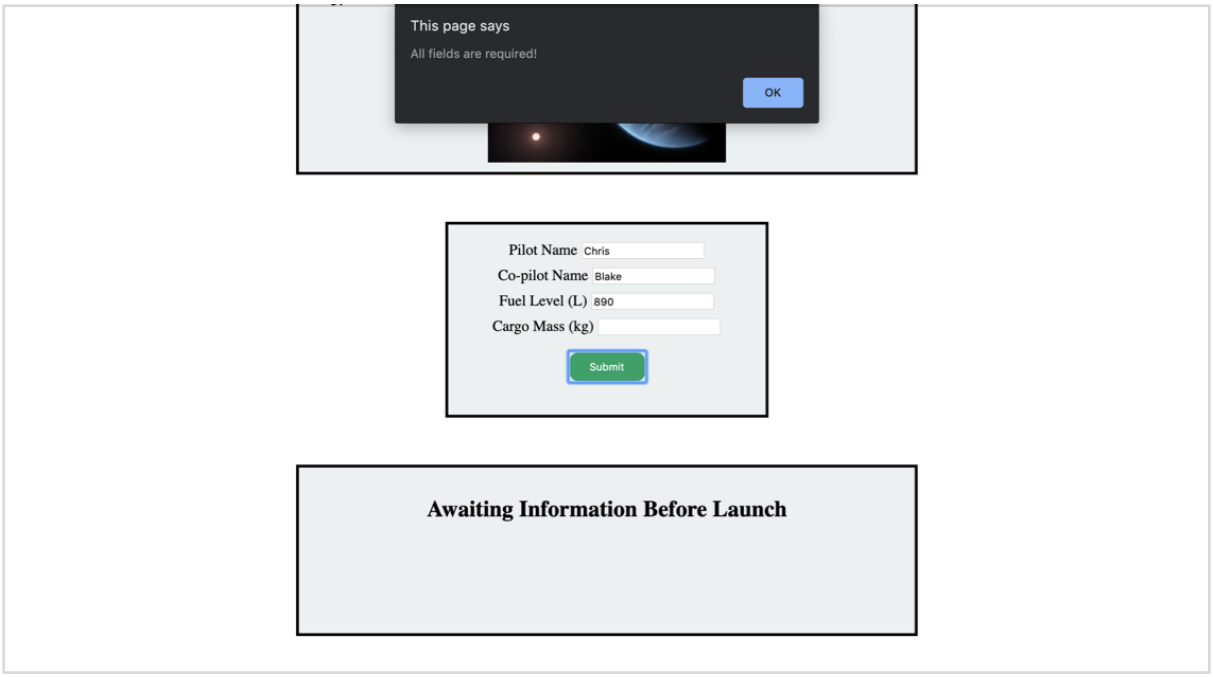
list.



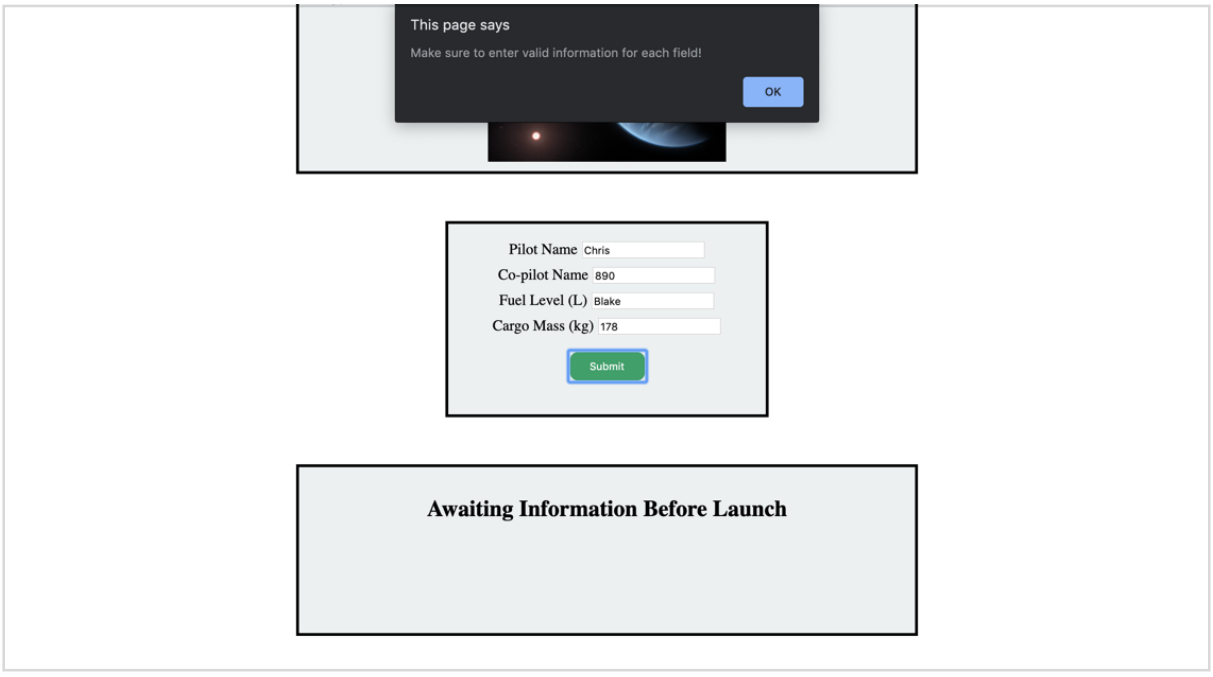
With only 890 liters of fuel, the status of the launch becomes red and states that the shuttle is not ready. The list has also updated to indicate that that is not enough fuel for the shuttle to launch.



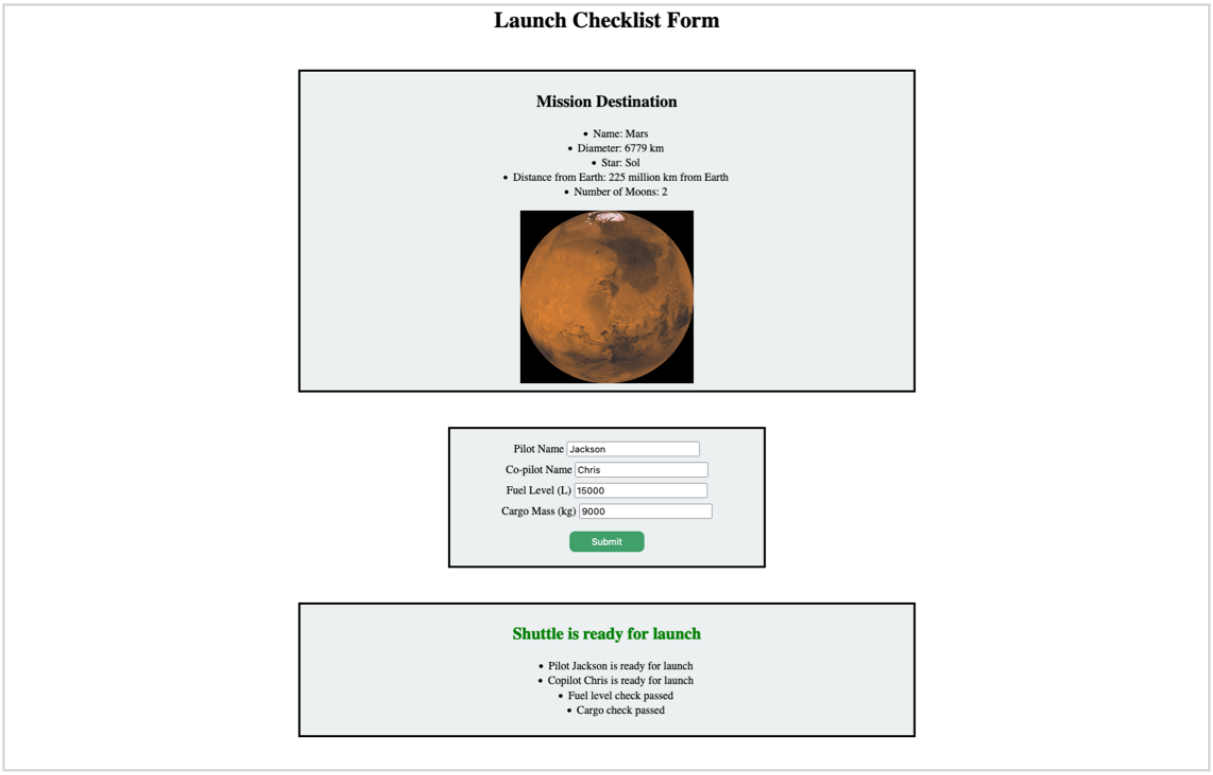
If the user forgets to enter the cargo mass, then an alert pops up letting the user know that all fields are required.



If the user switches up the information that needs to go in the fields, then an alert pops up letting the user know that they have tried to enter invalid information.



When all validation checks pass, we'll be ready for launch!



Bonus Missions

Your code chooses one of the planets at random from the JSON that it retrieves. Add a *Refresh Destination* button to the top section of the page that chooses another planet at random and updates the planetary data display accordingly.