Module 11 Challenge

Start Assignment

Due Sunday by 11:59pm **Points** 100 **Submitting** a text entry box or a website url

Background

Dana's webpage and dynamic table are working as intended, but she'd like to provide a more in-depth analysis of UFO sightings by allowing users to filter for multiple criteria at the same time. In addition to the date, you'll add table filters for the city, state, country, and shape.

What You're Creating

This new assignment consists of one technical analysis deliverable and a written report. You will submit the following deliverables:

- Deliverable 1: Filter UFO sightings on multiple criteria
- Deliverable 2: A written report on the UFO analysis (README.md)

Files

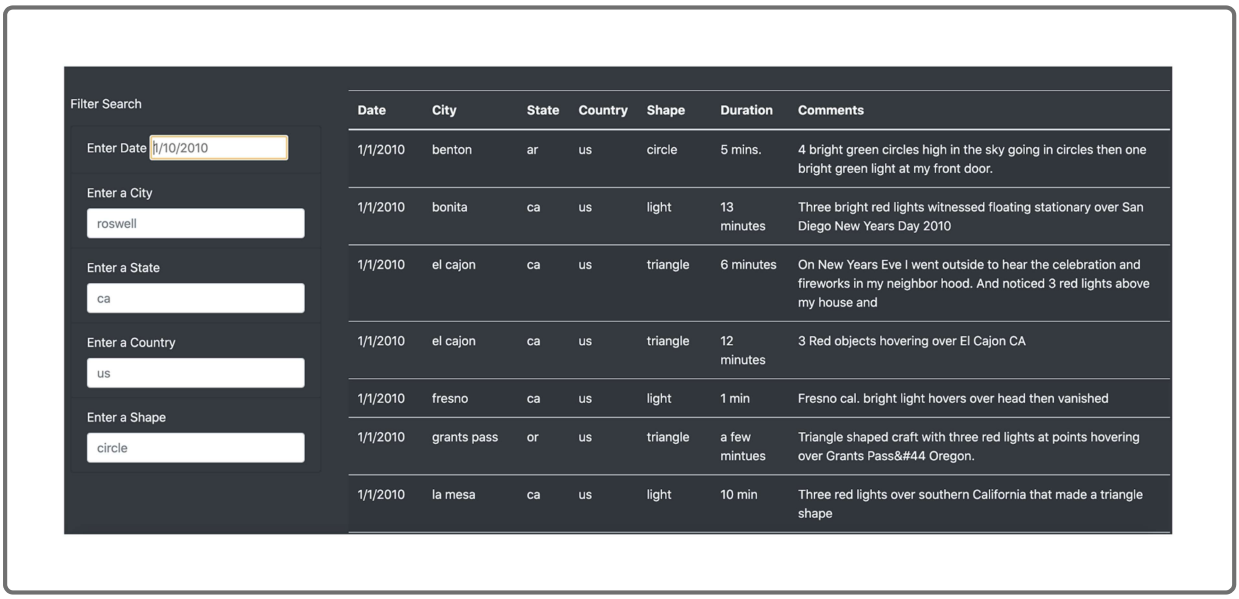
Use the following links to download the Challenge starter code.

Download the starter code. (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_11/ufo_starterCode.js)

Deliverable 1: Filter UFO sightings on multiple criteria (80 points)

Deliverable 1 Instructions

Using JavaScript and HTML, you'll modify the code in your `index.html` file to create more table filters. In addition to the date filter you created in this module, you'll add filters for the city, state, country, and shape, as shown in the following image:



Using JavaScript, you'll replace the `handleClick()` function in your `app.js` file with a new function that saves the element, value, and id of the filter that was changed. Then, you'll create a new function to loop through the dataset and keep only the results that match the search criteria. The webpage will be updated with the search criteria after pressing "Enter".

REWIND

For this deliverable, you've already done the following in this module:

- **Lesson 11.1.3:** Use `console.log()` to debug code
- **Lesson 11.2.2:** Create variables with `var` and `let`
- **Lesson 11.2.4:** Use `d3.select()`
- **Lesson 11.3.1:** Use JavaScript functions
- **Lesson 11.3.3:** Use arrow functions
- **Lesson 11.5.2:** Use `forEach()` with JavaScript objects
- **Lesson 11.5.3:** Use filters with `d3.select()`
- **Lesson 11.5.4:** Write `if-else` statements
- **Lesson 11.6.1:** Add `list-group-item class`, `label`, and `input` tags to add filtered data to an `index.html` file
- **Lesson 11.6.2:** Add the script tags for your code to an `index.html` file

Follow the instructions below and the numbered comments in the starter code to complete Deliverable 1.

1. Download the `ufo_starterCode.js`, rename it `app.js`, and place it in the js folder of your UFOs GitHub repository. The starter code includes the code used to populate the table from this module.

NOTE

> Before you rename the `ufo_starterCode.js` file, we suggest that you rename the `app.js` you created in this module as `app_1.js` or something similar to avoid using the wrong file for the Challenge.

2. In the `index.html` file, remove the list (`<li></li>`) element that creates the button.
3. Create four more list elements: city, state, country, and shape. These will be similar to the "Enter Date" list element. Each element should have the same "id" as the object properties in the `data.js` file.
4. In Step 1 of the `app.js` file, create an empty `filters` variable to keep track of all the elements that change when a search is entered. This variable will be used in Step 5 to store the property “id” and the value that was entered from user input.
5. Next, you will need to write code for two functions whose names we’ve provided in the starter code, `updateFilters()` and `filterTable()`.
    - The `updateFilters()` function will replace your `handleClick()` function and update the `filters` variable you created in Step 1.
    - The `filterTable()` function will filter the table data by the value that is entered for the "id" that has changed.
6. For Step 2, located before the `buildTable(tableData)` function at the end of the starter code, modify the event listener from this module so that it detects a "change" on each input element and calls the `updateFilters()` function.
7. In Step 3, we’ve provided the function, `updateFilters()`. Inside this function, you’ll write code in Steps 4-5 to update the filters based on user input.

8. In Step 4a, create a variable that saves the element that was changed using `d3.select(this)`.
9. In Step 4b, create a variable that saves the value of the changed element's property.
10. In Step 4c, create a variable that saves the attribute of the changed element's id.
11. In Step 5, write an `if-else` statement that checks if a value was changed by the user (variable from Step 4b). If a value was changed, add the element's id (variable from Step 4c) as the property and the value that was changed to the `filters` variable you created in Step 1. If a value was not entered, then clear the element id from the `filters` variable.

If you'd like a hint on how to update the filters based on user input, that's totally okay. If not, that's great too. You can always revisit this later if you change your mind.

SHOW HINT

12. In Step 6, inside the `updateFilters()` function, call the `filterTable()` function that will be used in Step 7.
13. In the `filterTable()` function in Step 7, write code to filter the table based on the user input that is stored in the `filters` variable.
14. In Step 8, create a variable for the filtered data that is equal to the data that builds the table. This variable will hold the updated table data based on the user input.

15. In Step 9, loop through the filters object and store the data that matches the filter values in the variable created in Step 8.
16. In Step 10, rebuild the table with the filtered data by passing the variable created in Step 8.
17. Deploy the web app on your GitHub pages.

Deliverable 1 Requirements

You will earn a perfect score for Deliverable 1 by completing all requirements below:

- The list element that creates the button is removed, and there are five list elements for filtering in the `index.html` file. **(20 pt)**
- The event listener is modified to detect changes to each filter in the `app.js` file. **(10 pt)**
- The `updateFilters()` function saves the element, value, and the id of the filter that was changed. **(20 pt)**
- The `filterTable()` function loops through all of the filters and keeps any data that matches the filter values. **(20 pt)**
- The webpage filters the table correctly based on user input. **(20 pt)**

Deliverable 2: A written report on the UFO analysis (README.md) (20 points)

Initialize your repository with a README, and write your analysis of Deliverable 1.

Deliverable 2 Instructions

For your written analysis, be sure to use complete and coherent sentences. Your written analysis should contain three sections, which cover the following:

1. **Overview of Project:** Explain the purpose of this analysis.
2. **Results:** Describe to Dana how someone might use the new webpage by walking her through the process of using the search criteria. Use images of your webpage during the filtering process to support your explanation.
3. **Summary:** In a summary statement, describe one drawback of this new design and two recommendations for further development.

Deliverable 2 Requirements

Structure, Organization, and Formatting (8 points)

The written analysis has the following structure, organization, and formatting:

- There is a title, and there are multiple paragraphs. **(2 pt)**
- Each paragraph has a heading. **(2 pt)**
- There are subheadings to break up text. **(2 pt)**
- Images are formatted and displayed where appropriate. **(2 pt)**

Analysis (12 points)

The written analysis has the following:

1. Overview of the analysis:
    - The purpose is well defined **(2 pt)**
2. Results:

- There is a description of how to perform a search, with images. **(4 pt)**

3. Summary:

    - The summary addresses one drawback of this webpage **(2 pt)**
    - The summary addresses two additional recommendations for further development **(4 pt)**

Submission

Once you're ready to submit, make sure to check your work against the rubric to ensure you are meeting the requirements for this Challenge one final time. It's easy to overlook items when you're in the zone!

As a reminder, the deliverables for this Challenge are as follows:

- Deliverable 1: Filter UFO sightings on multiple criteria
- Deliverable 2: A written report on the UFO analysis (README.md)

Upload the following to your UFOs GitHub repository:

1. The updated `app.js` file with the code for the challenge.
2. The updated `index.html` file.
3. The `data.js` file.
4. An updated README.md that has your written analysis

To submit your challenge assignment for grading in Bootcamp Spot, click Start Assignment, click the Website URL tab, then provide the URL of your UFOs GitHub repository, and then click Submit. Comments are disabled for graded submissions in BootCampSpot. If you have questions about your

feedback, please notify your instructional staff or the Student Success Manager. If you would like to resubmit your work for an improved grade, you can use the **Re-Submit Assignment** button to upload new links. You may resubmit up to 3 times for a total of 4 submissions.

IMPORTANT

> Once you receive feedback on your Challenge, make any suggested updates or adjustments to your work. Then, add this week's Challenge to your professional portfolio.

NOTE

> You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Next, and move on to the next Module.

Module-11 Rubric

Criteria	Ratings					Pts
Deliverable 1: Filter UFO Sightings on Multiple Criteria	**80 to >72.0 pts Demonstrating Proficiency** The index.html file has the following: ✓The list element for the button is removed. ✓ALL FIVE list elements for filtering are created. The app.js file has the following: ✓The event listener is modified to detect changes to ALL FIVE filters. ✓The updateFilters() function saves the element, value, and the id of the filter that was changed. ✓The filterTable() function loops through all of the filters and keeps any data that matches the filter values. ✓The webpage filters the table based on user input.	**72 to >66.0 pts Approaching Proficiency** The index.html file has the following: ✓ The list element for the button is removed. ✓ FOUR of FIVE list elements for filtering are created. The app.js file has the following: ✓The event listener is modified to detect changes to FOUR of FIVE filters. ✓The updateFilters() function saves the element, value, and the id of the filter that was changed, with one minor error. ✓The filterTable() function loops through all of the filters and keeps any data that matches the filter values, with two minor errors. ✓The webpage filters the table based on user input.	**66 to >60.0 pts Developing Proficiency** The index.html file has the following: ✓The list element for the button is removed. ✓THREE of FIVE list elements for filtering are created. The app.js file has the following: ✓The event listener is modified to detect changes to THREE of FIVE filters ✓The updateFilters() function saves the element, value, and the id of the filter that was changed, with two minor errors. ✓The filterTable() function loops through all of the filters and keeps any data that matches the filter values, with two minor errors. ✓The webpage filters the table based on user input.	**60 to >0.0 pts Emerging** The index.html file has the following: ✓The list element for the button is removed. ✓TWO of FIVE list elements for filtering are created. The app.js file has the following: ✓The event listener is modified to detect changes to TWO of FIVE filters. ✓The updateFilters() function saves the element, value, and the id of the filter that was changed, with three or more minor errors. ✓The filterTable() function loops through all of the filters and keeps any data that matches the filter values, with three or more minor errors. ✓The webpage filters the table based on user input.	**0 pts Incomplete**	80 pts

Criteria	Ratings					Pts
Deliverable 2: Structure, Organization, and Formatting	**8 to >7.0 pts Demonstrating Proficiency** The written analysis has ALL of the following: ✓There is a title, and there are multiple paragraphs. ✓Each paragraph has a heading. ✓There are subheadings. ✓Images are correct and displayed where appropriate.	**7 to >6.0 pts Approaching Proficiency** The written analysis has ALL of the following: ✓There is a title, and there are multiple paragraphs. ✓Each paragraph has a heading. ✓There are subheadings. ✓Images are correct and displayed where appropriate, but with one or two minor errors.	**6 to >4.0 pts Developing Proficiency** The written analysis has ALL of the following: ✓There is a title, and there are multiple paragraphs. ✓Each paragraph has a heading. AND ONE of the following: ✓There are subheadings. ✓Images are correct and displayed where appropriate.	**4 to >0.0 pts Emerging** The written analysis has ALL of the following: ✓There is a title, and there are multiple paragraphs. AND ONE of the following: ✓Each paragraph has a heading. ✓There are subheadings. ✓Images are correct and displayed where appropriate.	**0 pts Incomplete**	8 pts
Deliverable 2: Analysis	**12 to >10.0 pts Demonstrating Proficiency** The Deliverable Fulfills "Emerging" Required Criteria: AND has the following: ✓The summary addresses a drawback of the design. ✓The summary has TWO recommendations for further development.	**10 to >8.0 pts Approaching Proficiency** The Deliverable Fulfills "Emerging" Required Criteria: AND has at least ONE of the following: ✓The summary addresses a drawback of the design. ✓The summary has TWO recommendations for further development.	**8 to >6.0 pts Developing Proficiency** The Deliverable Fulfills "Emerging" Required Criteria: AND has at least ONE of the following: ✓The summary addresses a drawback of the design. ✓The summary has ONE of TWO recommendations for further development.	**6 to >0.0 pts Emerging** REQUIRED: ✓The purpose is well defined. ✓The description of how the search criteria works is well described with images.	**0 pts Incomplete**	12 pts

Criteria	Ratings	Pts
		Total Points: 100