Experiment Condition _____

Welcome to the JavaScript Library Experiment!

Programming Experience

(*These answers will be stored with your task progress and used in our analysis)*

Approximately how many programming languages have you learned (that is, written at least one program in)? (Circle one)

1 2-4 5-9 10+

Approximately how many programming libraries have you learned (that is, have used at least once)? (Circle one)

1 2-4 5-9 10+

How confident do you feel debugging JavaScript in Chrome? (Circle one)

Very Uncomfortable *Very Comfortable*

1 2 3 4 5 6 7

Demographic Information

(These answers will only be used for reporting summary characteristics of all participants and will be stored separately from all other data)

Age:

Gender:

Year or stage in school:

Email Updates

(Email addresses will be stored separately and will only be used emailing updates and will be stored separately from all other data)

If you would like to receive updates on the research, including full solutions to the tasks once the research is over, put your email below:

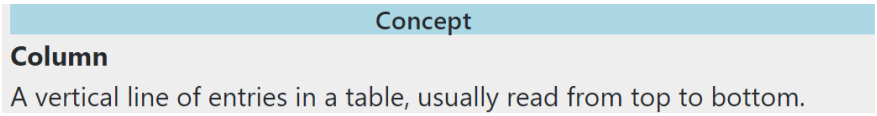
(Optional) Email:

Instructions

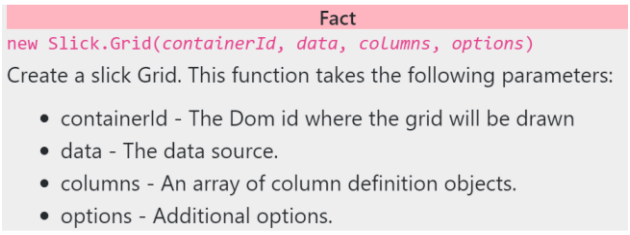
You will work on tasks in four different libraries. You will have 15 minutes to work on each API. Your goal while working on the tasks is not just to solve the them, but also understand how that solution works. We've designed the tasks to be very difficult without the expectation of anyone finishing them all or even getting close, so just complete as many as you can in order. When you complete a task, circle it with a pen. We'd also like a snapshot of your solution to each task, so when you complete a task, right click on the js file and select duplicate to make a copy and continue editing the original (see further details on the next page).

We have developed learning resources for these libraries in the form of code annotations. Depending on the task you will see the following three types of annotations:

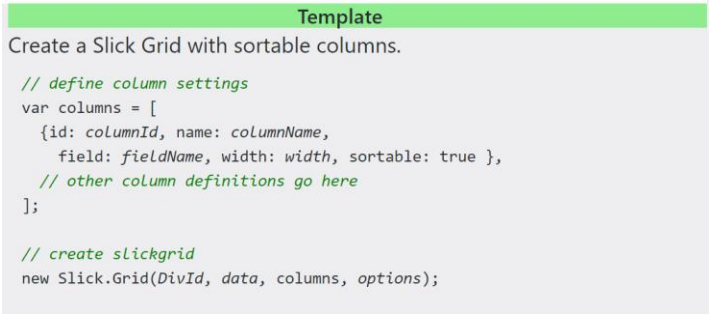
- Concept Definitions (**Concepts**): General concepts used by a code library.



- Execution and Usage Facts (**Facts**): Facts about how the library will run and how to make calls to the library.



- Code Templates (**Templates**): Multiple lines of code that are used together to achieve some output with the library.



We want to see how useful these resources are, so you can't use any other resources like Google or Stack-Overflow. Also, please try not to read from other participants' screens.

For each task, you will be given access to:

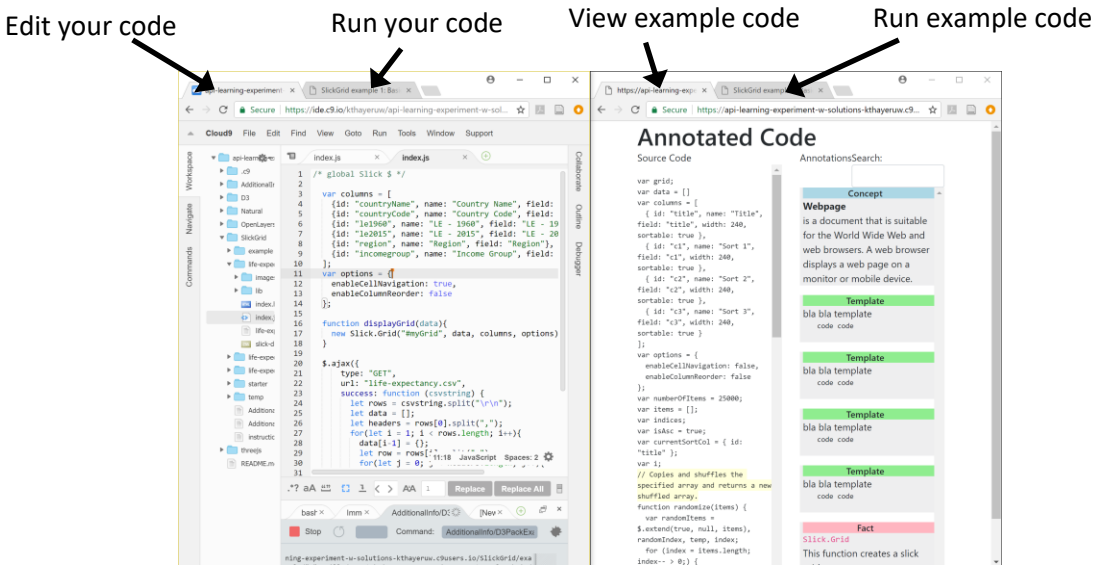
- The starting source code, which you will modify
- Example code, possible with some or many annotations. You may copy from this code.
- The running version of the example code

After working on each API, we will ask you how well you think you know each line of code in the example code, and ask what each line of code does. Then, when it is time to start the next task, you will minimize the left and right browser windows and begin.

After you have finished the study, do not talk about the tasks or Libraries with potential future participants until we have completed the study for everyone.

Practice API

You will see two browser windows open on your screen.



Left Browser window:

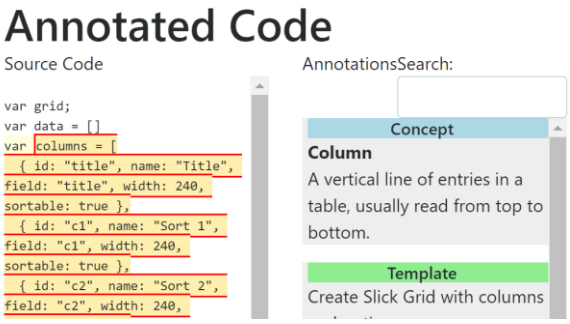
- The first tab has a JavaScript file open in cloud9. This is where you will edit files. Make sure to save them (ctrl-s) when you want to see your changes.
- The second tab has the running version of the code you are editing. On this tab use ctrl + refresh button to reload the page after you have finished editing.

Right Browser window:

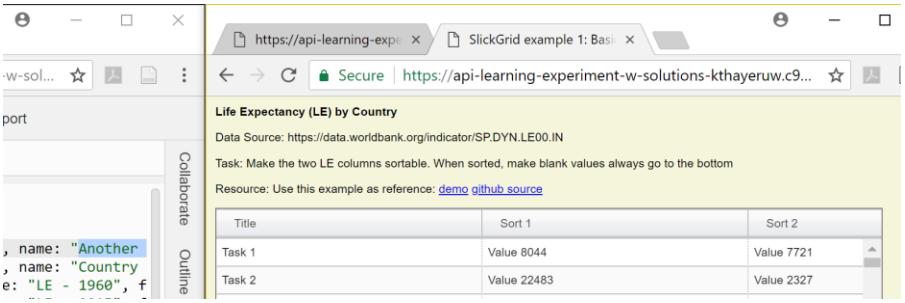
- The first tab has annotated example code. You can click on highlighted code to view annotations for that code.
- The second tab has the running example code. Look at this to gain insights on what the example code does. Depending on the task you may have few or no annotations.

Practice Tasks

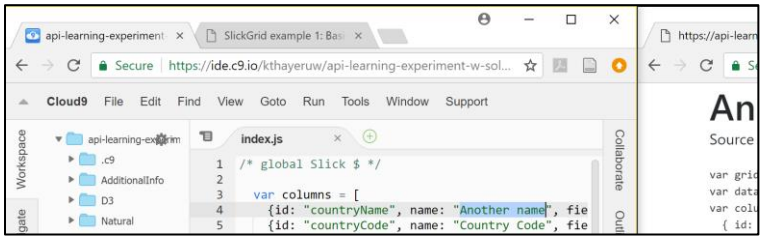
1) Click on the annotation for setting the columns variable. Scroll up and down the annotation column to see all the annotations. Click elsewhere to un-highlight the annotations.

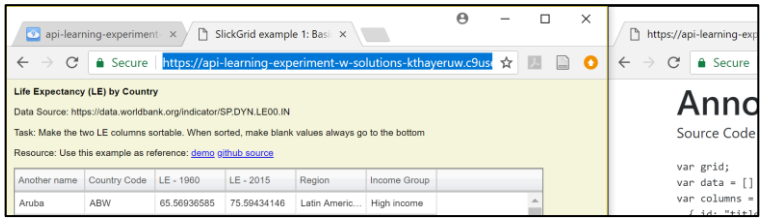


2) Open the second tab on the right browser to see the working version of the annotation code.

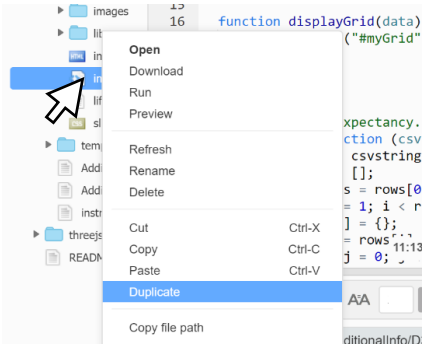


3) In the task.io code, on line 4 change `name: "Country Name"` to be `name: "Another Name"`. Save the change (ctrl-s) and refresh the second tab to see the change take effect.





4) Right click on the index.js file and select duplicate to make a snapshot copy of the file.



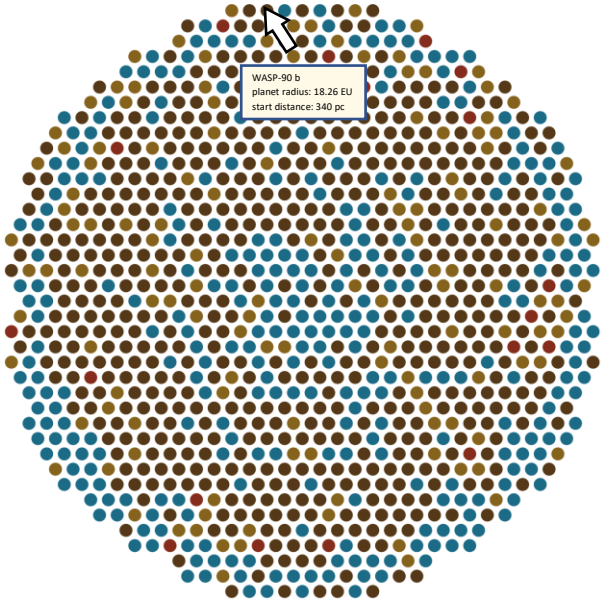
5) Tell the researcher that you are ready.

6) You can try editing the file more or reading the annotations if you want. Don't open any other files. Otherwise, just wait until everyone is ready.

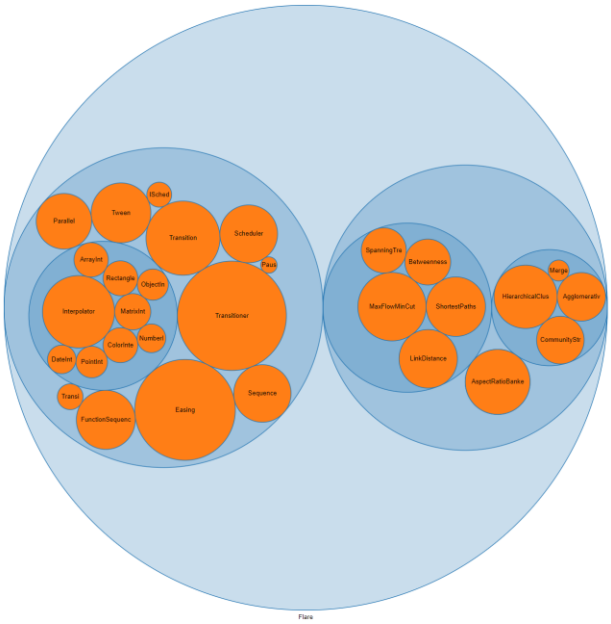
D3 Task

D3

You are working on a visualization of the known exoplanets. Exoplanets are planets that do not orbit around our sun. So far you have managed to get all the exoplanets to display (one dot per planet). The planets are colored according to their size and when you hover over one, you can see information about it. You want to improve the visualization by adding a title, making the dots size according to the planets size, include our solar system, and organize the planets by distance from our sun.



You have found an example visualization of some files in a programming library that may help you in your task.



Tasks: (Do these in order. When you've finished a task, circle the number and duplicate the js file. Continue working on the original js file.)

1) Add title to the top-left that says "Exoplanets" (see Final Graph below)

2) Make the circles have an area proportional with the square of the planet radius. Make sure the circles stay close together (reduce big spaces).

3) Add the planets of our solar system to the graph (the Array.concat function may help). It is difficult to tell if this worked, so if you think you have it, go to the next task.

4) Put planets of our solar system in a circle, separated from the rest of the planets. Make the circle surrounding the planets of our solar system not be filled in (you can use the provided "hollow-circle" class in the css).

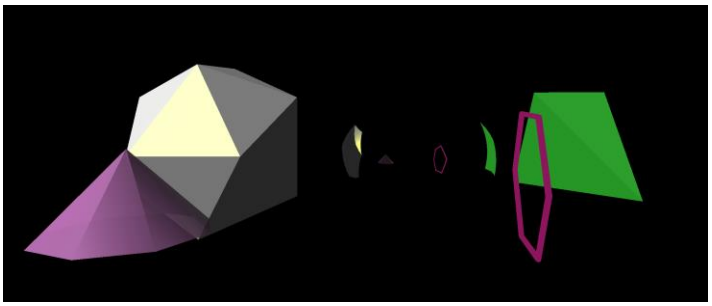
5) Sort planets by distance to the sun. That is, planets in the middle are close to the sun, planets on the outside are far from the sun.

three.js Task

three.js

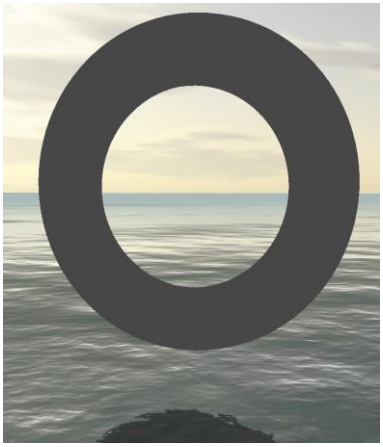
You want to make a cool loading screen for your website. You have already managed to create this 3D scene of an ocean:

You would like to add a spinning circular shape to indicate loading. You have found some example code that you think will help you do this. The example code creates the following scene:

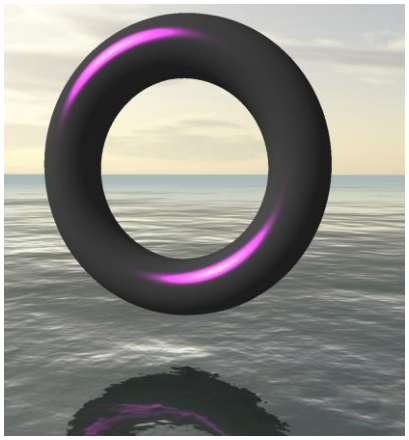


Tasks: (Do these in order. When you've finished a task, circle the number and duplicate the js file. Continue working on the original js file.)

1) Add this shape. Make it sized with an overall diameter of 40, the tube part with a diameter of about 10, and position the bottom 10 above the water. It should look smooth and circular. See the next page for clearer views of the shape.

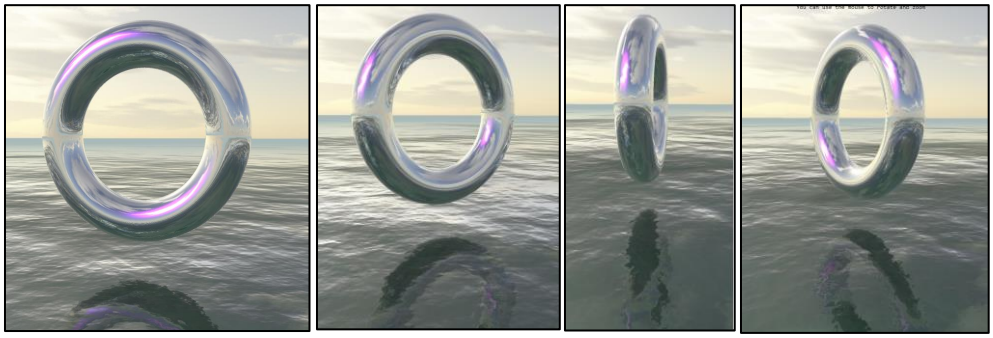


2) Make the shape be gray (hex color: 444444) and have a purple shine (hex color: 991199).



3) Make the shape appear perfectly reflective (note: the reflections can be hard to see if your shape is too dark, so make it white [hex color: ffffff] and uniformly shaded). Make sure this reflection work for both the sky and ocean.

4) Make the doughnut spin around it's vertical axis to indicate that the website is loading.



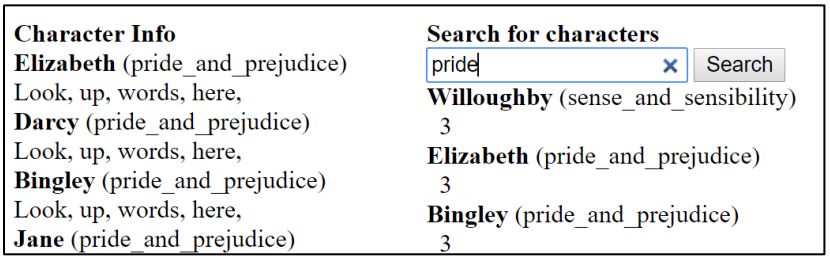
Natural Task

Natural

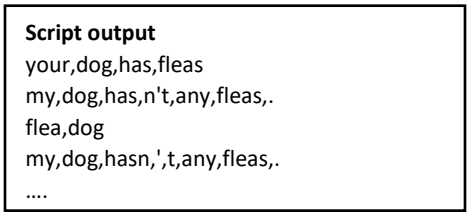
You want to analyze three Jane Austen novels (Pride and Prejudice, Sense and Sensibility, and Emma). Specifically, you want to see what words are associated with each character name, and search for characters based on words (see images for task 2 and 3). What's been done so far:

- Loaded the "Natural" JavaScript library
- Written code to load the text of the three novels
- Created a data structure with information on characters from those novels
- Created code that gets words that appear near character names
- Created filler code to display words associated with characters
- Created filler code for searching for characters that best match terms

Your page at the start has a temporary display of words that go with characters (these words are hard coded to start with), and a temporary search (how well characters match is currently just hard coded to a value of 3):



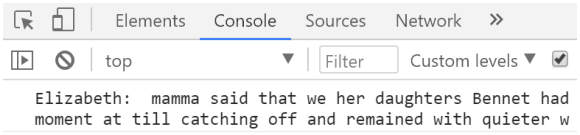
You also have found some code with examples of how to use the Natural Javascript Library. ?



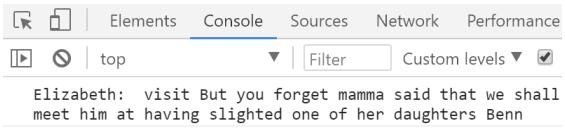
Tasks: (Do these in order. When you've finished a task, circle the number and duplicate the js file. Continue working on the original js file.)

1) `getWordsNextToCharacterName` currently returns words that occur up to two before and two after a character name, excluding the name itself (i.e., "word1 word2 characterName word1 word2"). You want to modify it to find words that occur up to six words before or six words after a character name. Check the results in the debugging console.

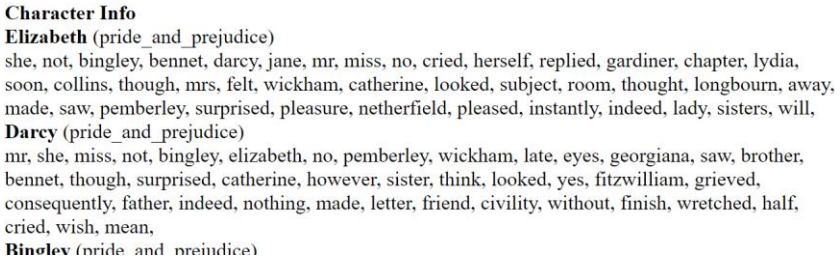
Before:



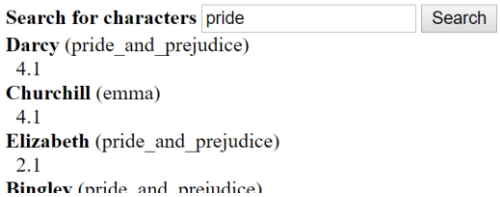
After:



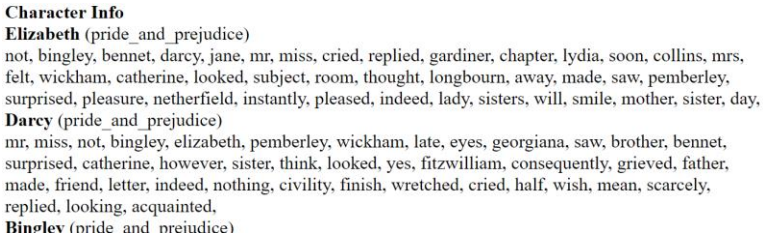
2) You want to find out which words commonly appear near a character's name (relative to words near other characters' names). For example: "she" often appears near the name Elizabeth, and more so than near other character names. Replace the code in `findWordsForCharacters` that sets `["Look", "up", "words", "here"]` with code that finds those commonly appearing words (you may have to modify other code as well). To do this, use the Natural library to group the strings returned by `getWordsNextToCharacterName` and find those words.



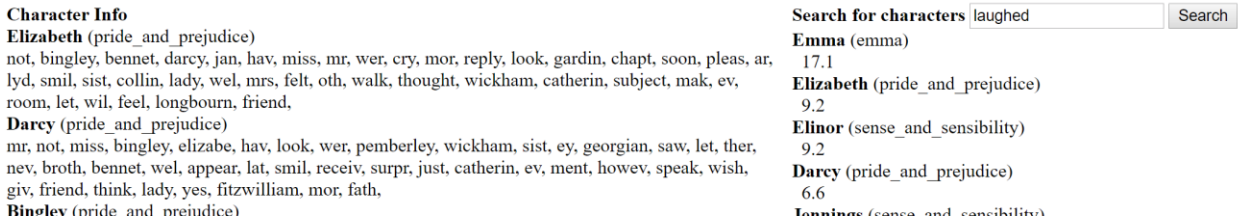
3) You want to be able to search for characters that best fit words. Use your work from task 2 and the Natural library to find how commonly the search words appear near a character's name (relative to other characters). Fill in the match value in the `search` function. Then search should work like this:



4) You notice some words found aren't ones you are interested in (like "she" and "not"). Use the Natural library to determine the type of word found based on the context of the group of seven words you found it with. Then ignore the following types of words:
   - Coordinating Conjunctions, like "nor"
   - Determiners, like "the"
   - Preposition or subordinating conjunctions, like "towards"
   - Personal pronouns, like "himself"
   - Possessive pronouns, like "our"



5) You notice that searching for "laugh" and "laughed" brings up different results, but you think those should be combined. Use the Natural library to combine related words like "laugh" and "laughed". Make sure search works with this.

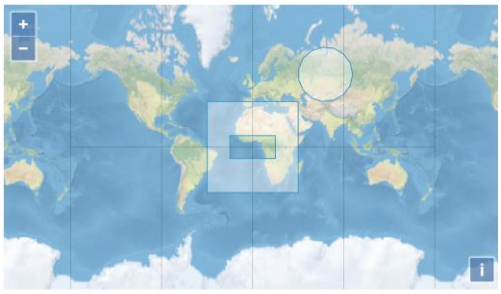


Open Layers Task

Open Layers

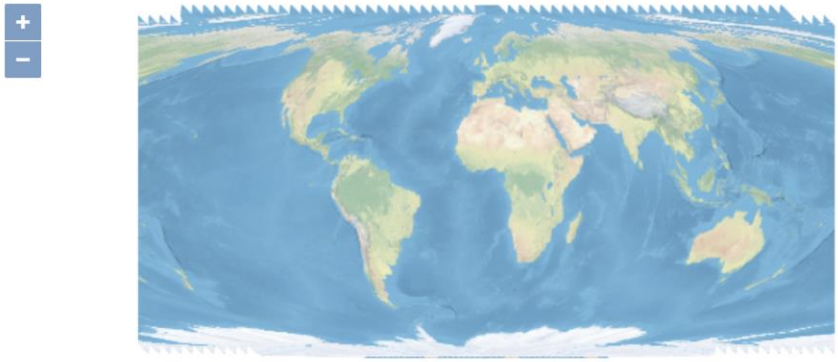
You are creating an interactive map on a webpage. To test if this mapping library will work for you project, you want to create a nice-looking interactive map that people can draw shapes on. You start with a map of the world:

You have found example code that creates the following two maps:

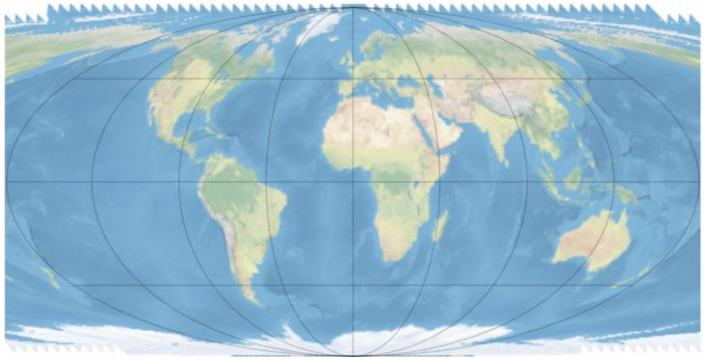


Tasks: (Do these in order. When you've finished a task, circle the number and duplicate the js file. Continue working on the original js file.)

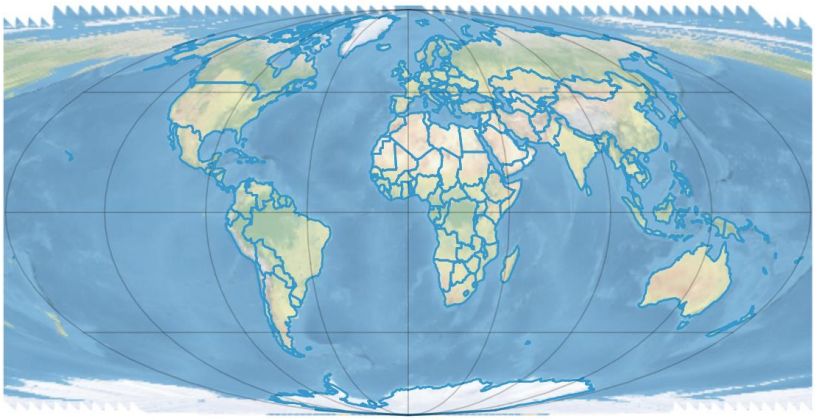
1) Change your map to be a rounded map using the sphereMollweideProjection variable already defined. If it works, you should see ridges along the top of the map.



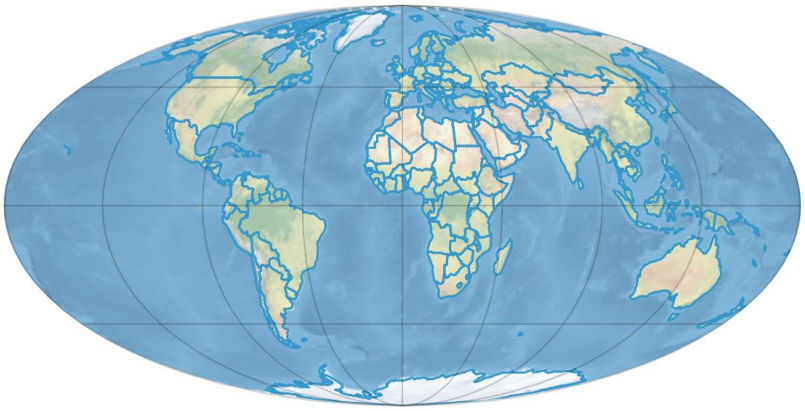
2) Add lines representing latitude and longitude to the map.



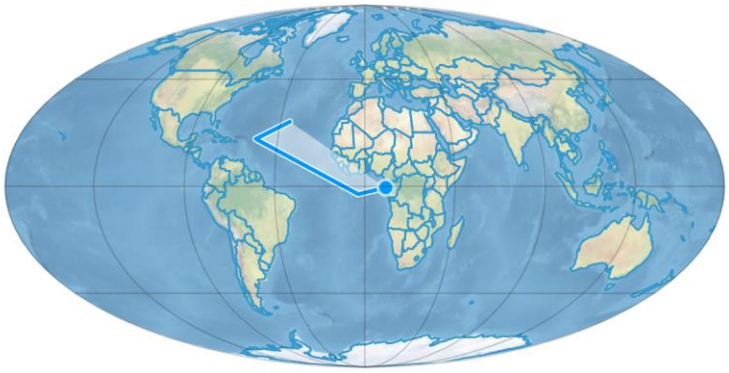
3) Add outlines to the countries from: https://openlayers.org/en/v4.6.4/examples/data/geojson/countries-110m.geojson (url in example code). Make sure you don't cover the countries with any color, but just have the outlines. Those outlines should be color #3399CC and have a width of 1.5.



4) Cover the areas outside the main globe (marked with latitude/longitude lines) with white so that you just see the main oval of the earth. Don't worry about covering the outside area perfectly, just get it close.



5) Allow users to draw shapes and modify them. Don't let them modify the country borders or any other feature. (Use OpenLayers default colors for the drawn shapes).



6) To aide with drawing, make it so the map makes it easier to draw points along the edges and corners of countries and previously drawn shapes.