Instruction

Project Rubric

Your project will be evaluated by a Udacity code reviewer according to the **Feed Reader Testing project rubric**. Please review for detailed project requirements.

Get the Starter Code

As you just saw, you will learn how to use Jasmine to write a number of tests against a pre-existing application. Go ahead and grab the starter code for that application from the repo below:

- **Feed Reader Testing starter code**

Jasmine Testing

Great! You now have the starter code. If you still need a quick refresher before beginning, we recommend reviewing the content from **JavaScript Testing**. You'll be applying lots of the skills and tools that you've learned, so make sure you're comfortable with the content. Ask yourself:

- How does an **IIFE** work? Feel free to look back at **Object-Oriented JavaScript** if you need a review
- How do you retrieve a particular element from the `Document`? Feel free to look back at **JavaScript and the DOM** if you need a review
- What about accessing an element's list of classes? How is that performed?
- How do you `describe` an object by what `it` does?
- How does the `expect()` function work? What do you pass into it?
- How are **matchers** used? Recall some of the ones you've seen (e.g., `toBe()`, `toBeDefined()`, etc.)
- What is `done`'s role in testing asynchronous processes?

Development Strategy

1. Familiarize yourself with the starter code
- Open up `index.html` and review the functionality of the application within your browser
- What is all the code in `app.js` doing? Be sure to read all code comments
- Check out `style.css`. How is styling applied to the application?
2. Explore the Jasmine spec file in `feedreader.js`
- This is the file in which you'll be writing your tests
- Make sure to read all code comments here as well
- Review the **Jasmine documentation** if needed
3. Edit the `allFeeds` variable in `app.js` to make the provided test fail
- See how Jasmine visualizes this failure in your application

- Return the `allFeeds` variable to a passing state after reviewing the failed test

4. Write a test that loops through each feed in the `allFeeds` object and ensures it has a URL defined *and* that the URL is not empty

- For example, how would you use a `for...of` loop in this test?

5. Write a test that loops through each feed in the `allFeeds` object and ensures it has a name defined and that the name is not empty

- Think about how you wrote the previous test. What are you testing for this time?

6. Write a new test suite named `"The menu"`

- What are you `describe`-ing in this test suite?

7. Write a test that ensures the menu element is hidden by default

- You'll have to analyze the HTML and the CSS to determine how the hiding/showing of the menu element is implemented
- What code in `app.js` is directly involved with toggling the menu on and off?

8. Write a test that ensures the menu changes visibility when the menu icon is clicked. This test should have two expectations: does the menu display itself when clicked, and does it hide when clicked again?

- Think about how you wrote the previous test. What is different this time around?
- Which clickable element are you checking for?
- How do you "simulate" a mouse click that element without actually clicking it?

9. Write a test suite named `"Initial Entries"`

- What are you `describe`-ing in this test suite?

10. Write a test that ensures when the `loadFeed` function is called and completes its work, there is at least a single `.entry` element within the `.feed` container

- How does Jasmine's `beforeEach()`function work?
- How does the `loadFeed()` function in `app.js` work? Is it synchronous or asynchronous?

11. Write a test suite named `"New Feed Selection"`

- What are you `describe`-ing in this test suite?

12. Write a test that ensures when a new feed is loaded by the `loadFeed` function that the content actually changes

- How is this test different from the previous test?

Additionally, note that:

- No test should be dependent on the results of another
- Callbacks should be used to ensure that feeds are loaded before they are tested
- Error handling should be implemented for undefined variables and out-of-bound array access
- When complete, all of your tests should pass

When you're all finished, write a `README` file detailing all steps required to successfully run the application. If you have added additional tests, provide documentation for what these future features are and what the tests are checking for.

Additional Resources

Feel free to check out **this post in Knowledge** for additional resources, curated by your fellow students!

You may find this Jasmine **cheatsheet** handy as you write your tests as well.