# Title: Stub and Mock for Ember test cases

When writing test cases be it Unit for Utils, Mixins, Routes, etc or Integration for Components, this article should help you to get writing tests easier.

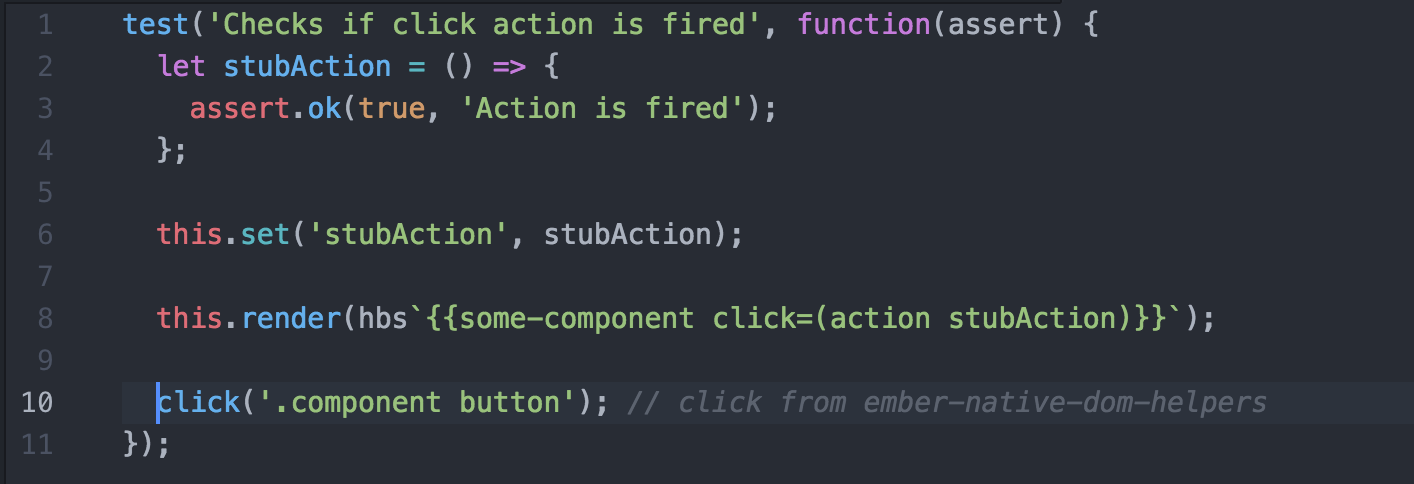
## Step-by-step guide

1. Before starting of to writing good unit/integration test cases, it is always important to know what code to cover and how to know that the tests you write will cover it? Here's an [article](https://confluence.freshworks.com/pages/viewpage.action?pageId=2525800) to know about code coverage for frontend.
2. After you have run your command in terminal (**npm run test-server**), you should see that the test starts in a browser (Chrome for now).
3. Choose the component/util/mixin/route/anything from the top right dropdown so that you can concentrate on writing test case for your file.
4. Once done, the test suite will re-run and supposedly if you are writing the test case for the first time for the file, you should mostly see a success (*Uncheck 'hide passed' check box*)
5. To ensure you have covered most part of your code, navigate to *freshsales/frontend/coverage* folder and open *index.html.*This file should show the progress you make with your test cases.

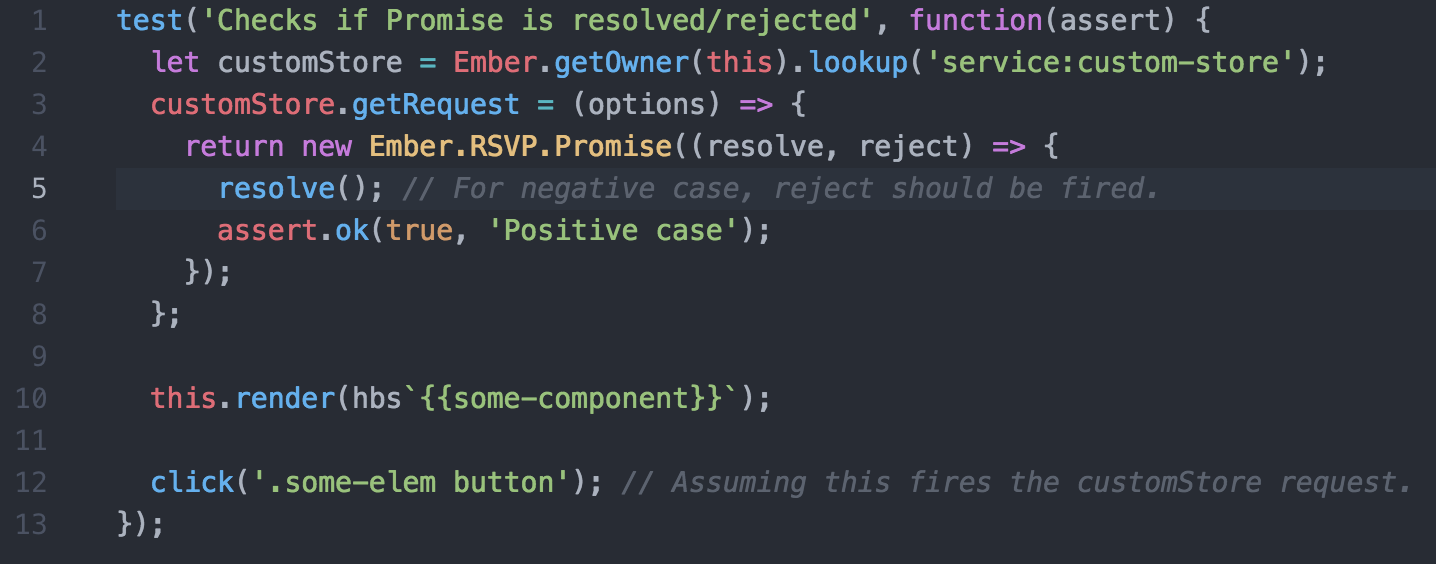
## Stubbing a method/action:

Stub refers to writing a dummy method that will be called when there is an action performed. For example, if your component calls an action on click of a button, you have to stub that action in your test suite like below so that you can ensure that button action will work when clicked.

**Reference: frontend/tests/components/bulk-update/component.js**



The most important use case for stubbing an action is for RSVP Promises. Your component might have a method like *get(this, 'customStore').getRequest({});*How do you validate if the customStore method that was fired, works fine for your component's use case for both positive and negative use case?



Once you try this, refresh the coverage page and you should see both the positive and negative cases for  you component in green!

**Reference: frontend/tests/components/bulk-update/component.js**

## Mocking data for consumption in unit/integration:

Mocking data required for a component is something that's dummy data you would like to add for the component to function. You can write a simple mock data using a variable and setting it up to the component or as big as seeding data into mirage db and using that up for your components when a query to the API is fired.

### Setting up dummy data within the test case

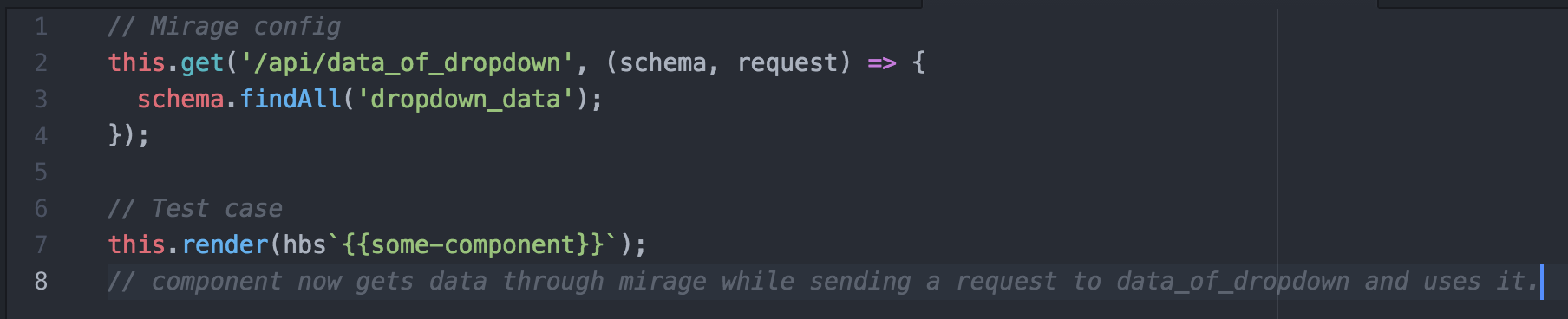
**Reference: frontend/tests/components/bulk-update/component.js**



### When do you need to use a Mirage API endpoint?

Let's assume you have a component that requires data from a generic API endpoint like */api/data\_of\_dropdown.json*

This can be easily achieved using mirage but placing a fixtures file in mirage. Refer this for more inputs: <http://www.ember-cli-mirage.com/docs/v0.3.x/fixtures/>



**Reference: frontend/tests/components/app-navigation-search/component.js**

## Related articles

<https://emberway.io/maintainable-user-interface-testing-with-ember-40f1b58040b6>

<https://github.com/rwjblue/rfcs/blob/42/text/0000-grand-testing-unification.md>

[http://www.ember-cli-mirage.com/](http://www.ember-cli-mirage.com/docs/v0.3.x/fixtures/)

* Page:
* [How to setup FS Bot catalog and write script for it](/display/freshsales/How+to+setup+FS+Bot+catalog+and+write+script+for+it)
* Page:
* [Fix for the 'browser disconnect' errors - testem](/display/freshsales/Fix+for+the+%27browser+disconnect%27+errors+-+testem)
* Page:
* [Stub and Mock for Ember test cases](/display/freshsales/Stub+and+Mock+for+Ember+test+cases)