Test Plan

Kevin Niemiller

To perform unit testing with Meteor, there is a package which allows these unit tests to take place which is called tinytest. Tiny test, like typical unit testing treats the application as an open system and allows access to all of the internals of the web application. Tiny test allows the following types of test syntax:

- test.equal(actual, expected, message, not)
- test.notEqual(actual, expected, message)
- test.instanceOf(obj, klass)
- test.matches(actual, regexp, message)
- test.isTrue(actual, msg)
- test.isFalse(actual, msg)
- test.isNull(actual, msg)
- test.isNotNull(actual, msg)
- test.isUndefined(actual, msg)
- test.isNaN(actual, msg)
- test.isUndefined(actual, msg)
- test.length(obj, expected_length, msg)

An example tiny test / unit testing that I used on my application is shown below. This testing is to test the AccountsEntry class.

Below is the html part of the AccountsEntry:

```
<template name="test_helper_home">

</template>

<template name="test_helper_account_buttons">
  {{> accountButtons}}
</template>

<template name="test_helper_sign_in">
  {{> entrySignIn}}
</template>

<template name="test_helper_sign_up">
  {{> entrySignUp}}
</template>
```

Below is the coffee script for the tiny test:

```
renderToDiv = (comp) ->
  div = document.createElement("DIV")
  Blaze.render(comp).attach(div)
  div

Tinytest.add "Accounts Entry - {{accountButtons}} helper", (test) ->
  div = renderToDiv(Template.test_helper_account_buttons)
  html = canonicalizeHtml(div.innerHTML)
  test.include html, "Sign In"
  test.include html, "Register"

Tinytest.add "Accounts Entry - wrapLinks setting on should wrap links in li elements", (test) ->
  AccountsEntry.settings.wrapLinks = true
  div = renderToDiv(Template.test_helper_account_buttons)
  html = canonicalizeHtml(div.innerHTML)
  test.include html, '<li><a href="/sign-in">Sign In</a></li>'

Tinytest.add "Accounts Entry - wrapLinks setting on should not show 'or span'", (test) ->
  AccountsEntry.settings.wrapLinks = true
  div = renderToDiv(Template.test_helper_account_buttons)
  html = canonicalizeHtml(div.innerHTML)
  scan = html.indexOf('<span>or</span>')
  test.equal(scan, -1, "Html output includes or span but shouldn't")

Tinytest.add "Accounts Entry - wrapLinks setting off should not wrap links in li elements", (test) ->
  AccountsEntry.settings.wrapLinks = false
  div = renderToDiv(Template.test_helper_account_buttons)
  html = canonicalizeHtml(div.innerHTML)
  scan = html.indexOf('<li>')
  test.equal(scan, -1, "Html output shouldn't have li tags")

Tinytest.add "Accounts Entry - wrapLinks setting off should show 'or span'", (test) ->
  AccountsEntry.settings.wrapLinks = false
  div = renderToDiv(Template.test_helper_account_buttons)
  html = canonicalizeHtml(div.innerHTML)
  test.include html, '<span>or</span>'

Tinytest.add "Accounts Entry - forgot password link should not show up if username only is set", (test) ->
  AccountsEntry.settings.passwordSignupFields = "USERNAME_ONLY"
  div = renderToDiv(Template.test_helper_sign_in)
  html = canonicalizeHtml(div.innerHTML)
  scan = html.indexOf('<a href="/forgot-password">')
  test.equal(scan, -1, "Forgot password link should not show up if username only is set")
```

The above represents unit tests for the following Use Cases:

- Use Case 1: Sign Up
    - Tests that a user must enter an “@” and a “.” somewhere after the “@“ symbol.
        - Tests a username with no “@” and it fails
        - Tests a username with no “.” and it fails
        - Tests a username with a “.” before the “@” and it fails
        - Tests a username with a @ as 5th character and “.” as 8th character and it passes

    - Password must have at least 7 characters and 1 digit
        - Tests with 6 characters and the test failed
        - Test with 7 characters and none of them are a digit and fails
        - Tests with 7 characters, one of them being a digit and test passes
        - Tests with 8 characters and one of them being a digit and the test passes.
- Use Case 8: Sign In
    - Tests that a user must enter an "@" and a "." somewhere after the "@" symbol for the username.
    - Tests that the username\email exists in the system
    - Tests that the password correctly aligns with the username\email
- Use Case 9: Sign Out
    - Tests that a user must enter an "@" and a "." somewhere after the "@" symbol.

Other Use Case Tests:

- Use Case 14: Remove Travel Path
    - Tests that the travel path can be removed from the database.
- Use Case 10: Subscribe to Alerts for Travel Paths
    - Tests that by setting the Subscribe radio button that the entry in the Subscribe database collection is updated to true.
- Use Case 11: Unsubscribe to Alerts for Travel Paths
    - Tests that by setting the Unsubscribe radio button that the entry in the Subscribe database collection is updated to false.