How does Bootstrap's test suite work?

Bootstrap uses [Jasmine](#). Each plugin has a file dedicated to its tests in `tests/unit/<plugin-name>.spec.js`.

- `visual/` contains "visual" tests which are run interactively in real browsers and require manual verification by humans.

To run the unit test suite via [Karma](#), run `npm run js-test`. To run the unit test suite via [Karma](#) and debug, run `npm run js-debug`.

How do I add a new unit test?

1. Locate and open the file dedicated to the plugin which you need to add tests to (`tests/unit/<plugin-name>.spec.js`).
2. Review the [Jasmine API Documentation](#) and use the existing tests as references for how to structure your new tests.
3. Write the necessary unit test(s) for the new or revised functionality.
4. Run `npm run js-test` to see the results of your newly-added test(s).

Note: Your new unit tests should fail before your changes are applied to the plugin, and should pass after your changes are applied to the plugin.

What should a unit test look like?

- Each test should have a unique name clearly stating what unit is being tested.
- Each test should be in the corresponding `describe`.
- Each test should test only one unit per test, although one test can include several assertions. Create multiple tests for multiple units of functionality.
- Each test should use `expect` to ensure something is expected.
- Each test should follow the project's [JavaScript Code Guidelines](#)

Code coverage

Currently we're aiming for at least 90% test coverage for our code. To ensure your changes meet or exceed this limit, run `npm run js-test-karma` and open the file in `js/coverage/lcov-report/index.html` to see the code coverage for each plugin. See more details when you select a plugin and ensure your change is fully covered by unit tests.

Example tests

```
// Synchronous test
describe('getInstance', () => {
  it('should return null if there is no instance', () => {
    // Make assertion
    expect(Tab.getInstance(fixtureEl)).toBeNull()
  })

  it('should return this instance', () => {
    fixtureEl.innerHTML = '<div></div>'

    const divEl = fixtureEl.querySelector('div')
    const tab = new Tab(divEl)
```

```javascript
    // Make assertion
    expect(Tab.getInstance(divEl)).toEqual(tab)
  })
})

// Asynchronous test
it('should show a tooltip without the animation', () => {
  return new Promise(resolve => {
    fixtureEl.innerHTML = '<a href="#" rel="tooltip" title="Another tooltip"></a>'

    const tooltipEl = fixtureEl.querySelector('a')
    const tooltip = new Tooltip(tooltipEl, {
      animation: false
    })

    tooltipEl.addEventListener('shown.bs.tooltip', () => {
      const tip = document.querySelector('.tooltip')

      expect(tip).not.toBeNull()
      expect(tip.classList.contains('fade')).toEqual(false)
      resolve()
    })

    tooltip.show()
  })
})
```