Fast visual testing for embedded content

Author: Daniel Kezerashvili

Claps: 39

Date: May 18, 2018

Embedding content on a customerâ€™s site is a privilege. You have the direct means to change every single visitorâ€™s experience. This should not be taken lightly.

Last month Klaviyo released an overhaul of our signup form feature set to make it easy to embed effective signup forms to our customersâ€™ websites. This was an exciting product to develop with a host of interesting challenges.

Most ecommerce websites have popup or flyout signup form modals that appear to the user and ask them to signup for future marketing. Since marketers typically create these, weâ€™ve built an easy to use tool where they can create signup forms in a drag and drop editor. Then, after installing a short code snippet to their site, we deliver these modals to visitors as they browse.

Our content needs to be robust so that the forms our customers see in the designer are the same as what they see on their site.

In this blog post Iâ€™ll cover the tools we use to make sure our embedded content is consistent with every release.

We have no control over what CSS or javascript will be on our customerâ€™s website. Weâ€™ve encountered custom overrides to methods on `Array` and `Object`, CSS like `div{ width: 100% }`, and more.

Even though we donâ€™t have total control, we want to be sure that regardless of the CSS changes a customer may make, or code that we ship, the content we show works and looks the same.

Enter visual regression testing

Visual regression tests are meant to catch drift in the visual representation of content. They are not meant to replace unit tests, or to act as â€œSelenium styleâ€ tests. Visual regression tests are not the tool to test your API.

Our Goals:

- Test suite should run in < 30 seconds
- Test suite results should be consistent across multiple runs.
- Tests should provide enough detail to start debugging
- Tests should be easy to write

What we use:

- graphcoolâ€™s Chromeless for running headless Chrome in AWS Lambda
- Amexâ€™s jest-image-snapshot for generating visual diffs
- Jenkins for Continuous Integration

By running headless Chrome in Lamba, we can spawn many tests asynchronously bringing down runtime. In fact, this is so fast (and cheap) that we can run these tests using jestâ€™s watcher function! We donâ€™t usually do this, but itâ€™s certainly helpful when trying to debug an issue.

Real Life Examples!

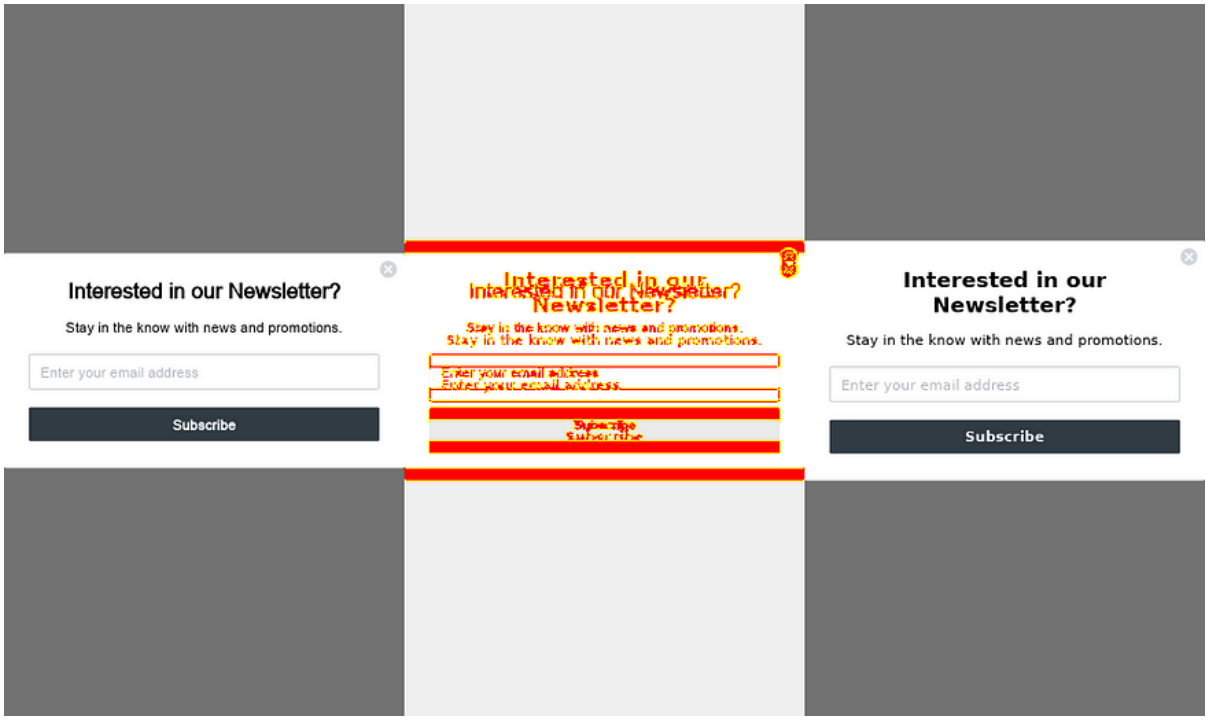
This has already helped us catch a few subtle bugs and confirm than any changes we do make, are what we expect.

> Embedding content on a customerâ€™s site is a privilege.

Example 1: Different OS Font Defaults

The headless chrome instance in Lambda may not have the fonts you expect. Itâ€™s good to explicitly include the font families you will need.

In the below image we show a diff between our correct fonts, and what our form renders as when those fonts are missing.



Example 2: Subtle Padding Change Due to CSS Selectivity

While developing a new feature, we accidentally made a very small change in padding to our labels by overriding some selectively. It would have been hard to tell the difference between the forms by eye but the regression system caught it quickly!