One Page of Code: A Javascript Feature Flag Demo, Costing $0

What you'll need:

- A free LaunchDarkly trial account.
- Basic Javascript templating experience, and a blasphemous sense of disrespect.
- A free Heroku account.
- A hacker's will to just get something working.

Code on github: https://github.com/LizCira/ldarkly_onepage
Live on Heroku: https://launchdarkly-test.herokuapp.com/

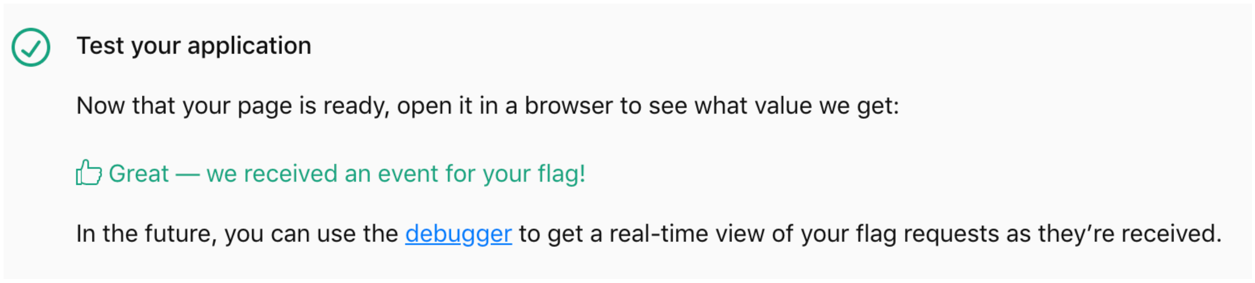
I'm lazy, I admit it. I'm also curious. Feature flags are--for very good reason-- shaking up the software development industry right now. I'd heard a lot about them, and I decided to see how quickly I could put together a demo. In that spirit, I opted for an off-the-shelf solution, vs. completely writing my own—there are much longer posts weighing the pros and cons of roll-your-own vs. SAAS feature flagging, but here is not the place. My goal was to get a web page that appeared differently to anonymous web users .

I signed up for LaunchDarkly, and jumped right into their Quickstart documentation. The page allows you to select your language of choice. I didn't have an existing project, and I was really looking for something visual, so I chose Javascript. Select a name for your new feature flag, and the Quickstart wizard generates a code snippet for you.

```
<script>
  var user = {
    firstName: 'Bob',
    lastName: 'Loblaw',
    key: 'UNIQUE IDENTIFIER',
    custom: {
      groups: 'beta_testers'
    }
  };

  var div = document.createElement('div');
  document.body.appendChild(div);

  div.appendChild(document.createTextNode('Initializing...'));

  var ldclient = LDClient.initialize('5cfd56c1a2f8290765e10047', user);

  function render() {
    var shouldShow = ldclient.variation('retro', false);
    var label = (shouldShow ? 'Showing' : 'Not showing') + ' your feature to
' + user.key;
    div.replaceChild(document.createTextNode(label), div.firstChild);
  }

  ldclient.on('ready', render);
  ldclient.on('change', render);
</script>
```

Paste the generated snippet in an index.html file, and open this in your local browser. You'll experience the satisfaction of seeing "Not showing your feature to Bob Loblow" rendered in plain text. You'll also see that your feature flag is reporting back to your LaunchDarkly dashboard, so the API itself is now active.

You'll see a client side ID has been generated in your code snippet, where the LaunchDarkly client is instantiated—this is how the API calls are made.



Test your application

Now that your page is ready, open it in a browser to see what value we get:

Great — we received an event for your flag!

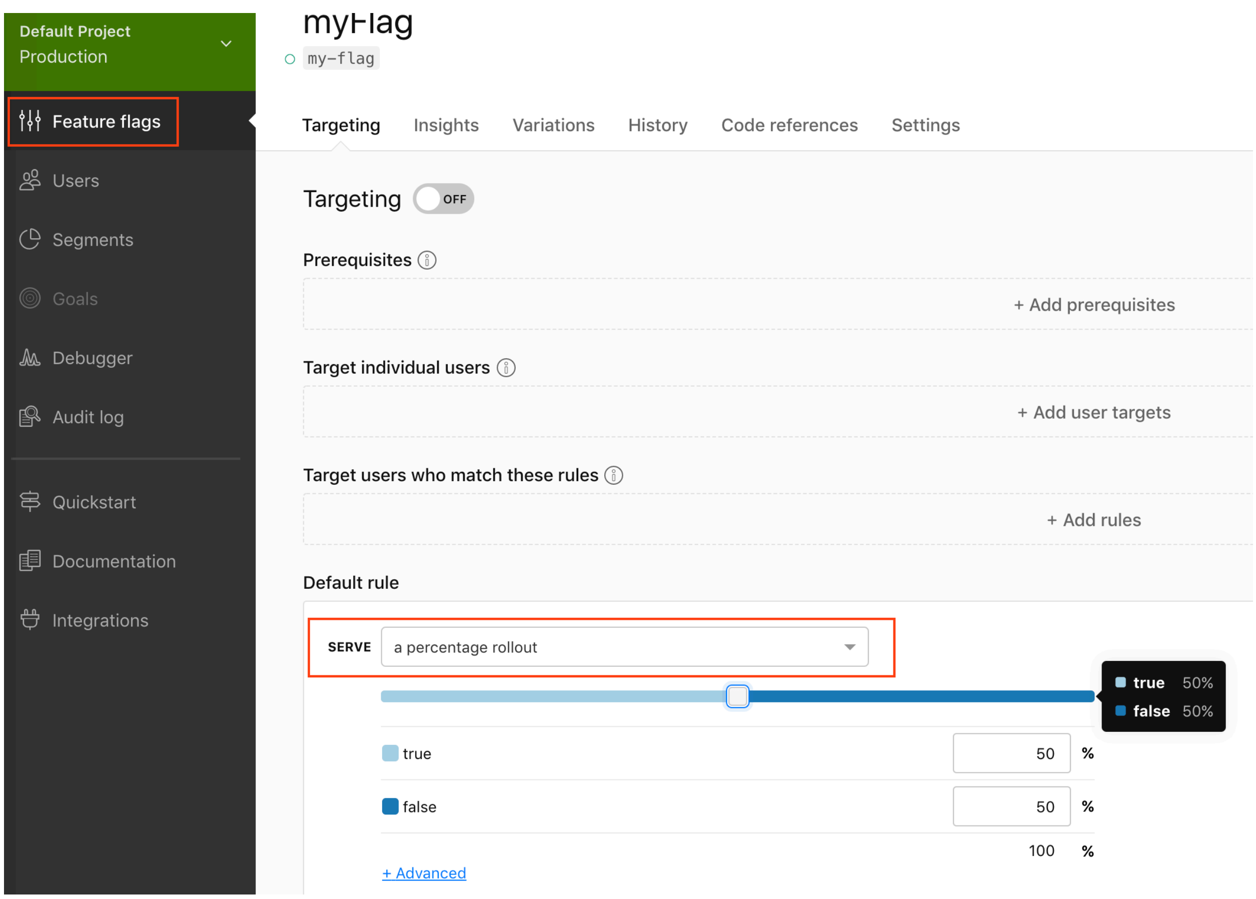
In the future, you can use the debugger to get a real-time view of your flag requests as they're received.

The ID string used by the client is not a secret, and can be safely checked into version control. You'll also see "Bob Loblow" has been added to the "Users" section of your LaunchDarkly dashboard.

Technically, this was it—I had a working feature flag. But I wanted something a little more visually exciting. It's been five years since I did much with Javascript, but if not now, then when?

First, I was a bit confused by the hardcoded user data in the snippet example. This piece of the setup wizard assumes you'll already have a user database of some sort to work with, and that you'll be using your flags to segment these users. I had no users. I wasted a bit of time creating a random function to generate new users, and looking into passing some form data to dynamically generate users and assign them to different groups; I still like this idea, but it wasn't the shortest road to where I wanted to go, so I'll revisit it later.

Instead, I created a new feature flag **without using the wizard**. After naming your flag, you can choose "a percentage rollout" under "Default rule" to serve the new flag to an assigned percentage of users, regardless of user data.


myFlag

my-flag

Targeting Insights Variations History Code references Settings

Targeting OFF

Prerequisites

+ Add prerequisites

Target individual users

+ Add user targets

Target users who match these rules

+ Add rules

Default rule

SERVE a percentage rollout

true	50	%
false	50	%
	100	%

+ Advanced

My initial thought was to use one of Javascript's most satisfying basic features—the prompt—and serve it to 50% of my users. However, the Javascript prompt didn't play well with LaunchDarkly, so I went with different colored backgrounds instead. I picked the 50/50 split option on the LaunchDarkly dashboard, and added an auto-refresh script so I could see my flag being served.

After that, it was a simple matter of changing the background and font colors within the feature flag code. **This is not best practice**, but it meant I didn't need to create a .css style sheet. My app still consists exclusively of index.html!

This was the full logic for my A/B test

```
function prompt(){
                    var shouldShow = ldclient.variation('name-pop', false);
                    console.log(shouldShow)

                    if (shouldShow) {
                    var html = 'Hello, friend'
                    document.body.style.backgroundColor = "red";
                    var elem = document.getElementById("div1");
                    elem.style.color = "Black";
```

```
                elem.style.fontSize = "120px";
} else {
        var html = 'Hello, world!'
        document.body.style.backgroundColor = "blue";
        var elem = document.getElementById("div1");
        elem.style.color = "White";
                elem.style.fontSize = "120px";
}
```

But only I could see my glorious page in my local browser. My free github page is already in-use, so to publish, I chose Heroku. Unfortunately, publishing only an index.html to Heroku is a known issue, and I had to create a couple more files to spoof a php page. I'm choosing not to count these extra pages, since using a different host would mean they weren't needed, but I'm including this info for anyone trying to exactly replicate this stack.

And that's it! I can now watch my page refresh with a 50% change-rate.

TO DO:

- Set up analytics, using LaunchDarkly's "Goals" feature in the dashboard (not covered with a free account. Alternatively, look into using Google Analytics for same functionality.
- Figure out why the Javascript pop-up wasn't working.
- Look into hosting Javascript on EC2 t2 free tier.