Launch Darkly Project

Nabeel William Khashan

The first thing I tried was creating an SDK using Python. I confirmed through my terminal what version of Python I had and used the terminal to install Python Upgrade.

```
nkhashan@NKHASHAN-M-HPG2 ~ % python3 --version
Python 3.9.6
```

The next step was to follow the quickstart for creating a python3 SDK. I made a hello-python directory with the requirements.txt and the main.py.

```
nkhashan@NKHASHAN-M-HPG2 hello-python % ls
main.py                    requirements.txt
```

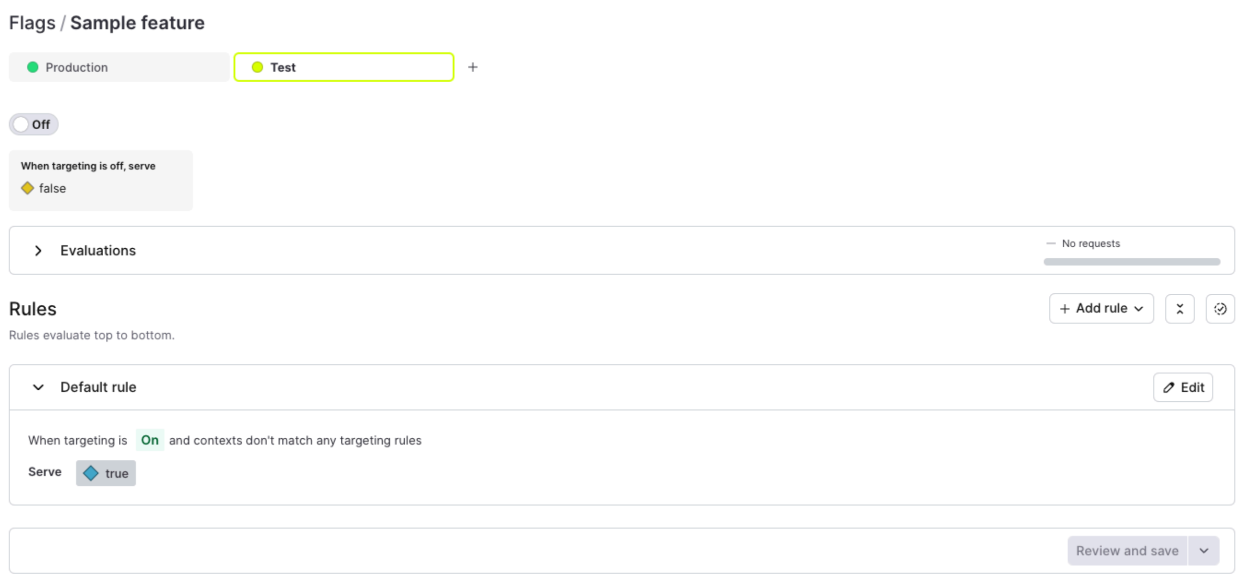
I added the code requested in the instructions to the main.py file I created.

```
nkhashan@NKHASHAN-M-HPG2 hello-python % vi main.py
import os
import ldclient
from ldclient import Context
from ldclient.config import Config
from threading import Lock, Event
# Set sdk_key to your LaunchDarkly SDK key.
sdk_key = os.getenv("LAUNCHDARKLY_SDK_KEY")
# Set feature_flag_key to the feature flag key you want to evaluate.
feature_flag_key = "flag"
def show_evaluation_result(key: str, value: bool):
    print()
    print(f"*** The {key} feature flag evaluates to {value}")
def show_banner():
    print()
    print("        ██       ")
    print("          ██     ")
    print("      ████████   ")
    print("         ███████ ")
    print("██ LAUNCHDARKLY █")
    print("         ███████ ")
    print("      ████████   ")
    print("          ██     ")
    print("        ██       ")
    print()
class FlagValueChangeListener:
    def __init__(self):
        self.__show_banner = True
        self.__lock = Lock()
    def flag_value_change_listener(self, flag_change):
        with self.__lock:
            if self.__show_banner and flag_change.new_value:
                show_banner()
                self.__show_banner = False
            show_evaluation_result(flag_change.key,
flag_change.new_value)
if __name__ == "__main__":
    if not sdk_key:
        print("*** Please set the LAUNCHDARKLY_SDK_KEY env first")
        exit()
    if not feature_flag_key:
        print("*** Please set the LAUNCHDARKLY_FLAG_KEY env first")
        exit()
    ldclient.set_config(Config(sdk_key))
    if not ldclient.get().is_initialized():
        print("*** SDK failed to initialize. Please check your internet
connection and SDK credential for any typo.")
        exit()
```

```
print("*** SDK successfully initialized")
# Set up the evaluation context. This context should appear on your
# LaunchDarkly contexts dashboard soon after you run the demo.
context = \
```

Once you run the script with the SDK key this is what you get.

```
LAUNCHDARKLY_SDK_KEY="sdk-3d2795da-b837-43cb-975b-af95377f8971" python3 main.py
```


Flags / Sample feature

● Production ● Test

Off

When targeting is off, serve
false

Evaluations — No requests

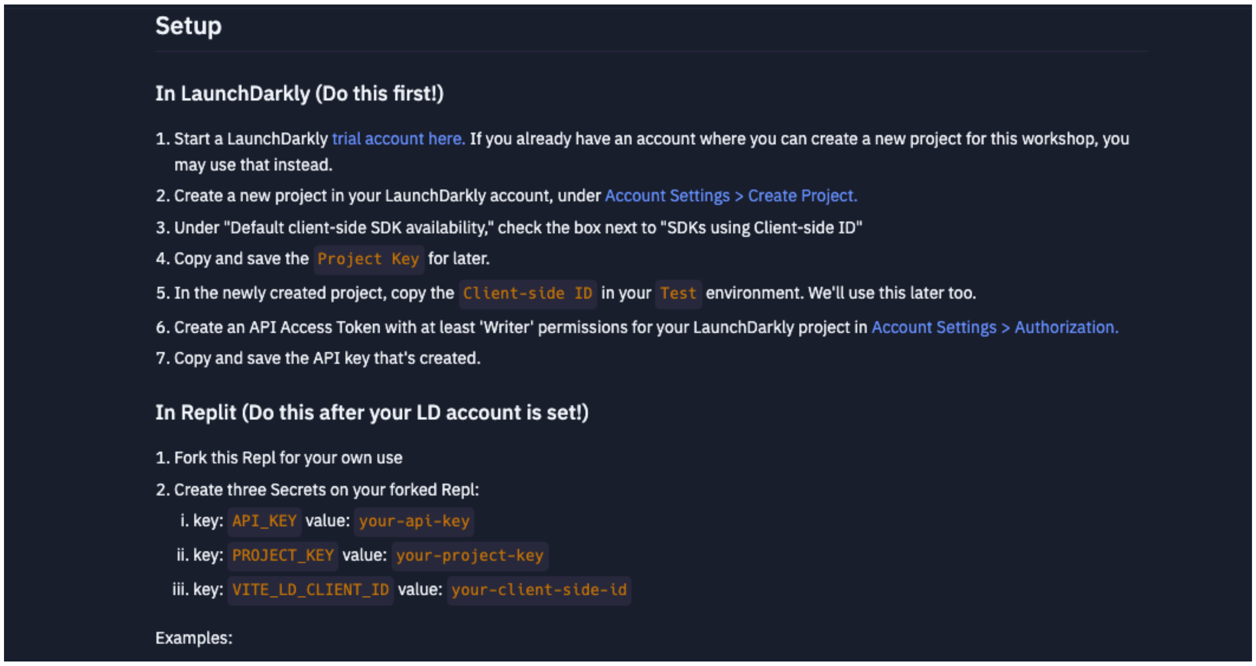
Rules
Rules evaluate top to bottom.

Default rule

When targeting is On and contexts don't match any targeting rules
Serve true

I wanted an application to show the functionality of the project. I watched a bunch of videos but the one I started paying attention to was Cody De Arkland. I noticed he had his demo application on a public site on Replit and I was told I could fork an application. I created a Replit account and I looked up the Launch

Darkly project. I followed the readme file. I had already started a Launch Darkly Trial Account.



Setup

In LaunchDarkly (Do this first!)

1. Start a LaunchDarkly trial account here. If you already have an account where you can create a new project for this workshop, you may use that instead.
2. Create a new project in your LaunchDarkly account, under Account Settings > Create Project.
3. Under "Default client-side SDK availability," check the box next to "SDKs using Client-side ID"
4. Copy and save the `Project Key` for later.
5. In the newly created project, copy the `Client-side ID` in your `Test` environment. We'll use this later too.
6. Create an API Access Token with at least 'Writer' permissions for your LaunchDarkly project in Account Settings > Authorization.
7. Copy and save the API key that's created.

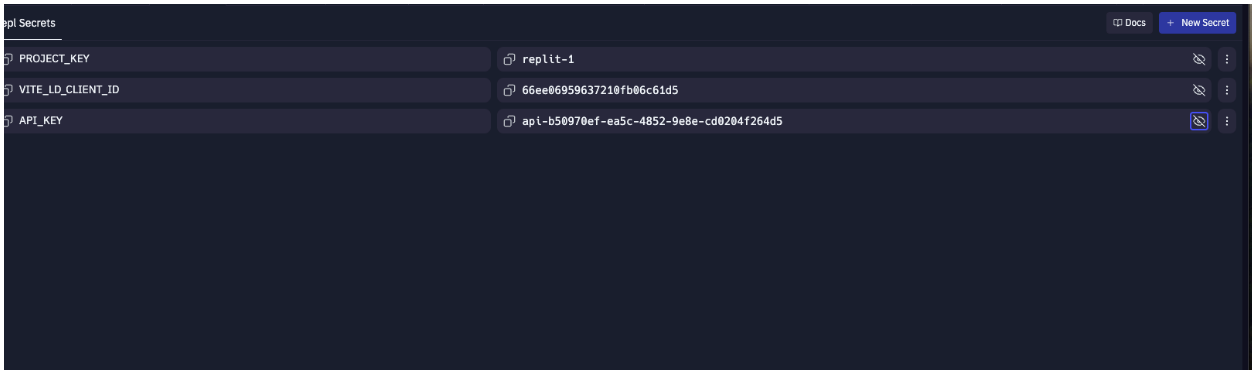
In Replit (Do this after your LD account is set!)

1. Fork this Repl for your own use
2. Create three Secrets on your forked Repl:
   i. key: `API_KEY` value: `your-api-key`
   ii. key: `PROJECT_KEY` value: `your-project-key`
   iii. key: `VITE_LD_CLIENT_ID` value: `your-client-side-id`

Examples:

Forking the Application from Replit

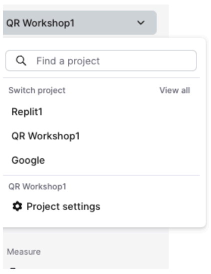
This was the first time I have ever done a fork from Replit. I have most of my past projects from Riverbed and AppDynamics in GitHub. I followed the instructions and used the Secret section to input the information needed to integrate with Launch Darkly.

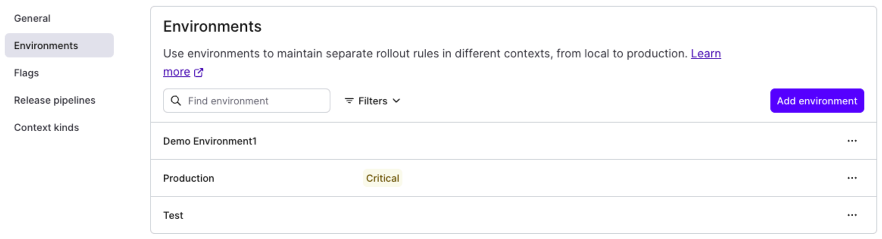


epl Secrets

PROJECT_KEY	replit-1
VITE_LD_CLIENT_ID	66ee06959637210fb06c61d5
API_KEY	api-b50970ef-ea5c-4852-9e8e-cd0204f264d5

Once I completed the Fork all the flags that were set up in the QRCode Application showed up in Launch Darkly under the project I created Replit-1. I was playing with the Launch Darkly a bit and I created a Project that I didn't end up using. The main project used for the assignment is Replit-1 just to reiterate. Google was a default and the QR Workshop is one I manually added.



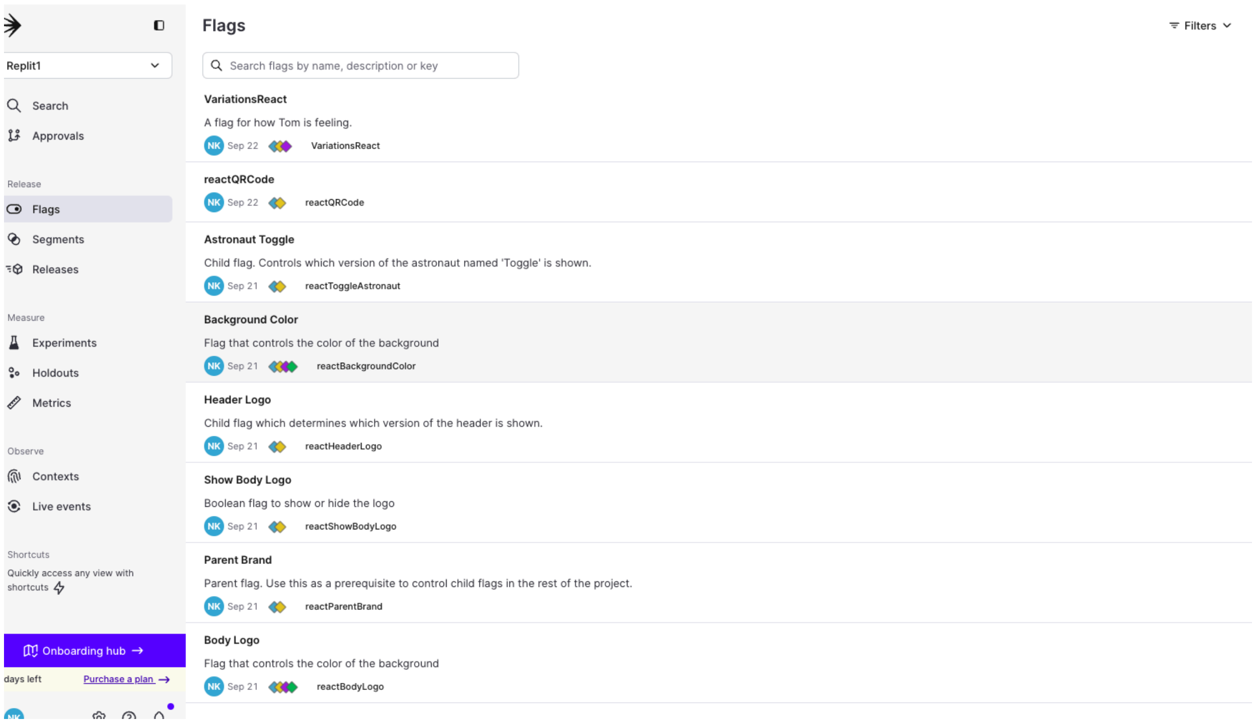


Environments

Use environments to maintain separate rollout rules in different contexts, from local to production. Learn more

Demo Environment1	
Production	Critical
Test	

The next screen shot will show the flags that were brought over from the Fork. One thing I wanted to note I manually created some Flags in other projects just to get an understanding of how to create Flags in the platform.



Replit1

Search
Approvals

Release
Flags
Segments
Releases

Measure
Experiments
Holdouts
Metrics

Observe
Contexts
Live events

Shortcuts
Quickly access any view with shortcuts

Onboarding hub →
days left Purchase a plan →

Flags

Filters

Search flags by name, description or key

VariationsReact
A flag for how Tom is feeling.
NK Sep 22 VariationsReact

reactQRCode
NK Sep 22 reactQRCode

Astronaut Toggle
Child flag. Controls which version of the astronaut named 'Toggle' is shown.
NK Sep 21 reactToggleAstronaut

Background Color
Flag that controls the color of the background
NK Sep 21 reactBackgroundColor

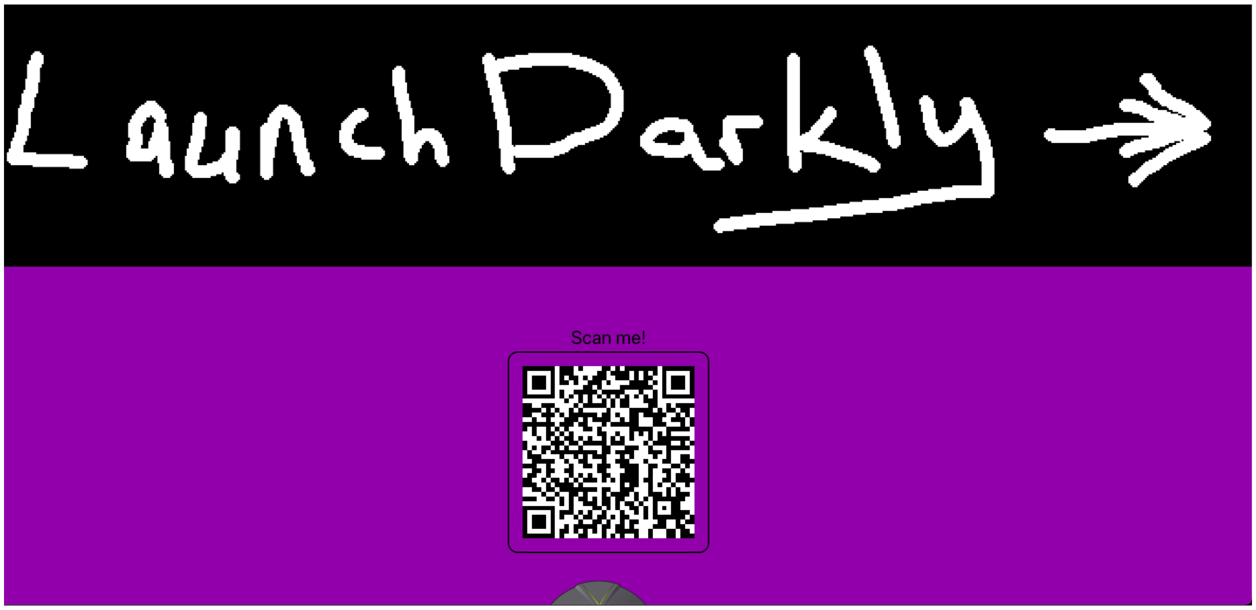
Header Logo
Child flag which determines which version of the header is shown.
NK Sep 21 reactHeaderLogo

Show Body Logo
Boolean flag to show or hide the logo
NK Sep 21 reactShowBodyLogo

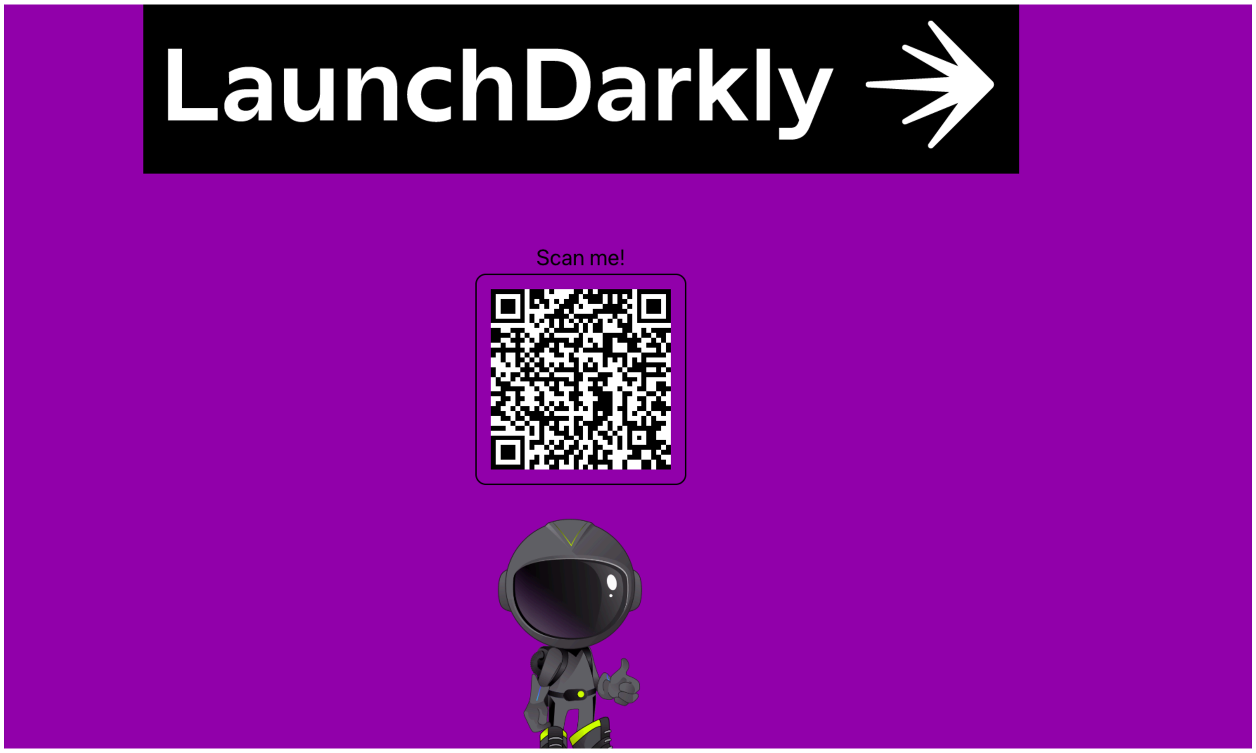
Parent Brand
Parent flag. Use this as a prerequisite to control child flags in the rest of the project.
NK Sep 21 reactParentBrand

Body Logo
Flag that controls the color of the background
NK Sep 21 reactBodyLogo

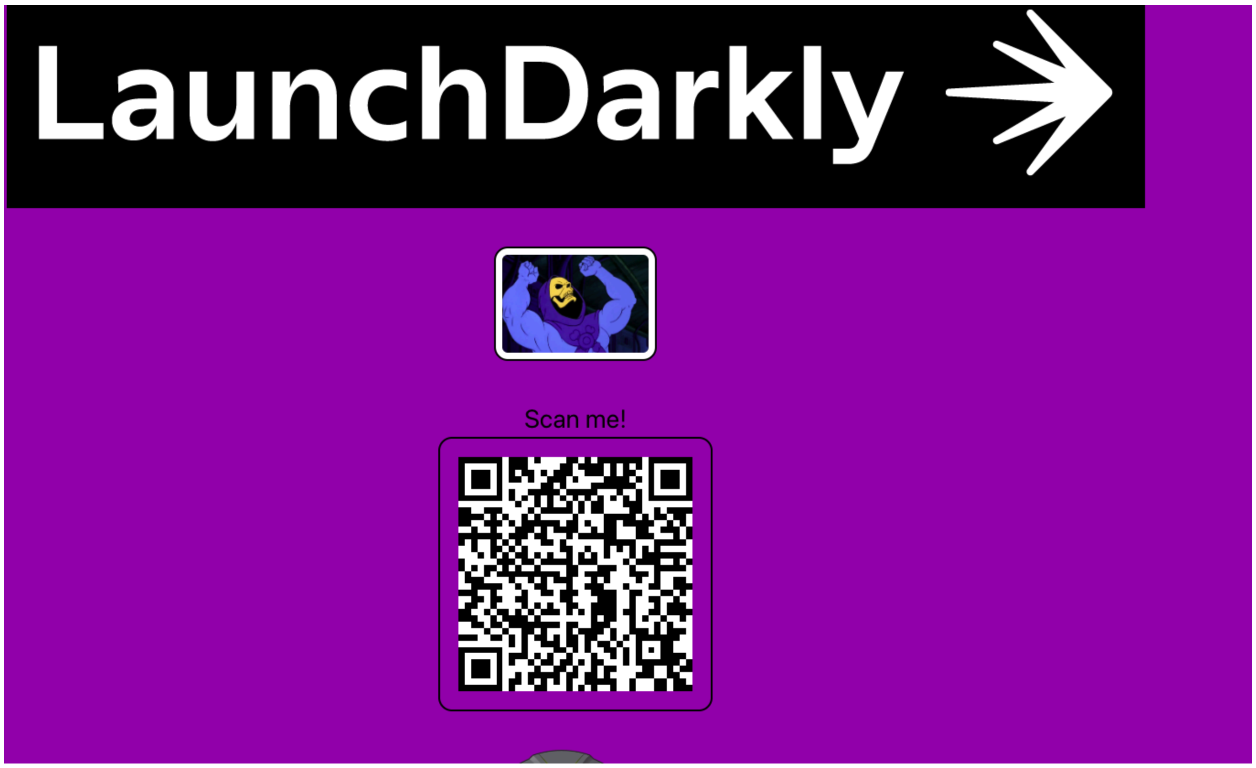
The next step was to perform the things mentioned in the project. I toggled the flags as requested. I used the QRCode to put the application on my mobile device and changed the colors. When I toggle the feature flag Header Logo on this is what I get.



When I toggle the feature off this is my result.

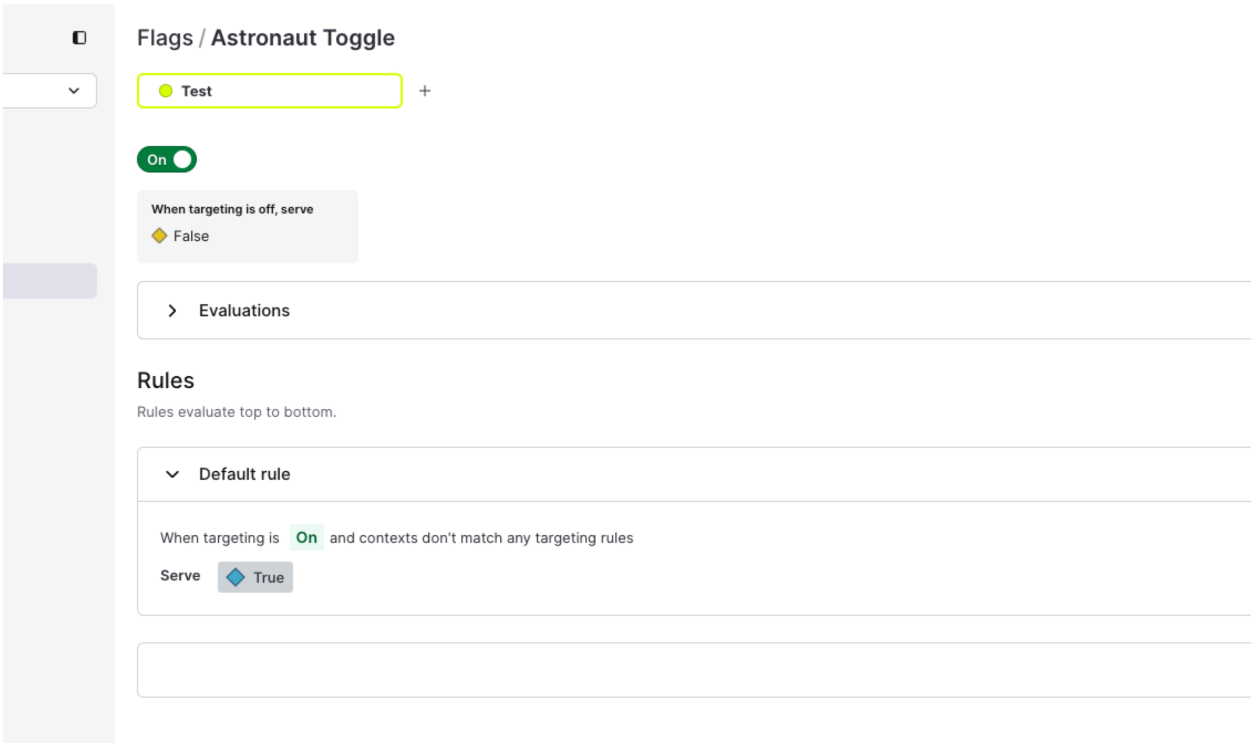


The next Feature Flag is the Show Body Logo. There should be a Skeletor Body Logo, I had to update the variation in the Body Logo to view Skeletor, otherwise, it would be a blank circle or blank depends on Variations chosen and Targeting.



I was able to Toggle the image and use the Variations in the Body Logo to determine what is shown and you use Targeting to show users, Regions, or users using certain devices. Coming from a testing background this is all amazing to do in production or test. The next Toggle we are going to look at is the AstronautToggle.

We saw the Astronaut in the image above. Let's turn the Astronaut Toggle on for the new image.



Flags / Astronaut Toggle

Test

On

When targeting is off, serve
False

Evaluations

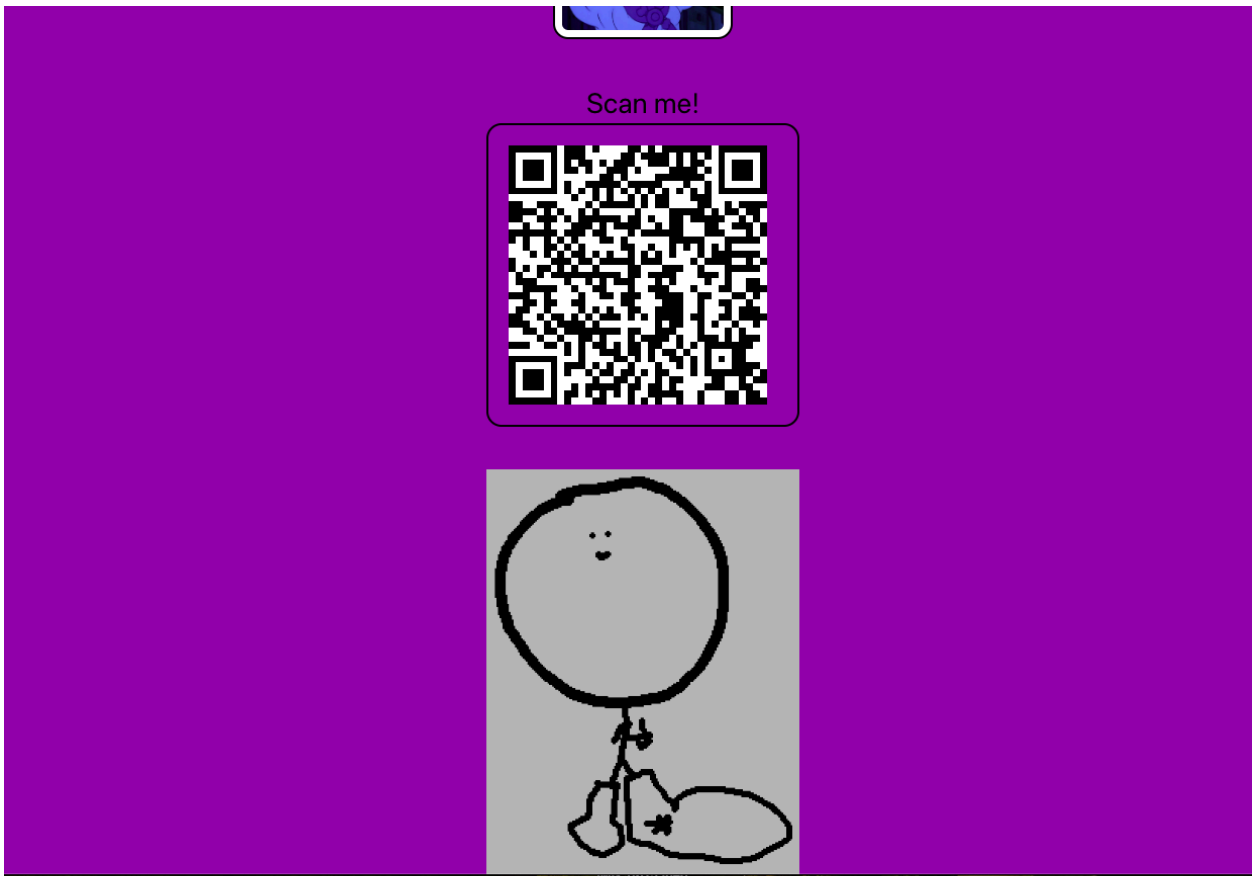
Rules

Rules evaluate top to bottom.

Default rule

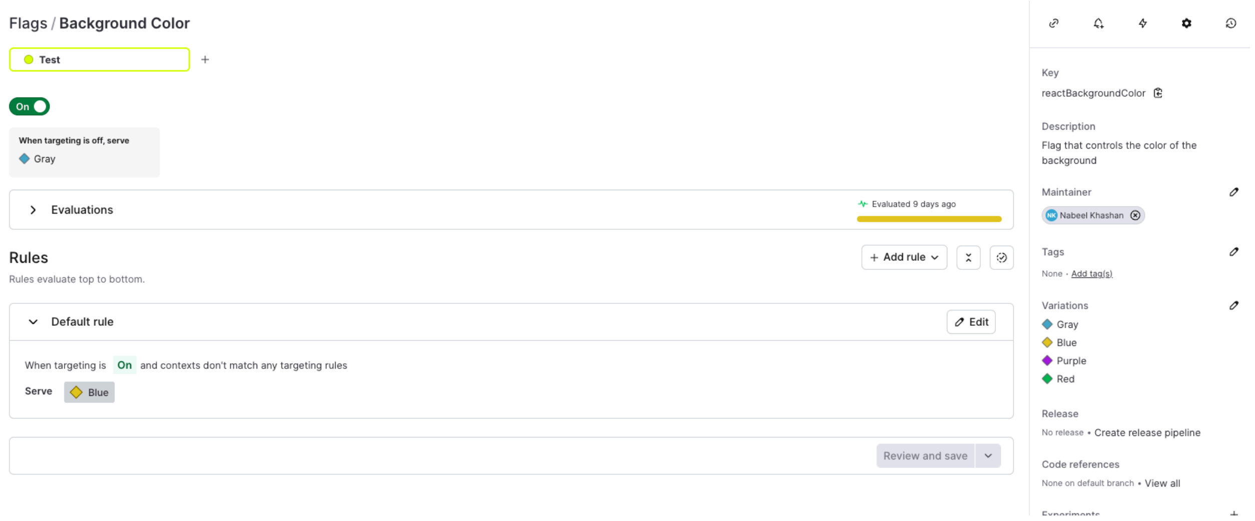
When targeting is On and contexts don't match any targeting rules

Serve True



This isn’t very professional for a Logo so I will turn this Feature Flag off.

The next one I will show is the background Color. Let's see research finds Purple to be unappealing for marking purposes. Let's change the color with a feature flag. The Background Color had four options Grey, Blue, Purple, and Red. In this case, we changed it to Blue.



Flags / Background Color

Test

On

When targeting is off, serve
Gray

Evaluations — Evaluated 9 days ago

Rules
Rules evaluate top to bottom.

Default rule

When targeting is On and contexts don't match any targeting rules
Serve Blue

Review and save

Key
reactBackgroundColor

Description
Flag that controls the color of the background

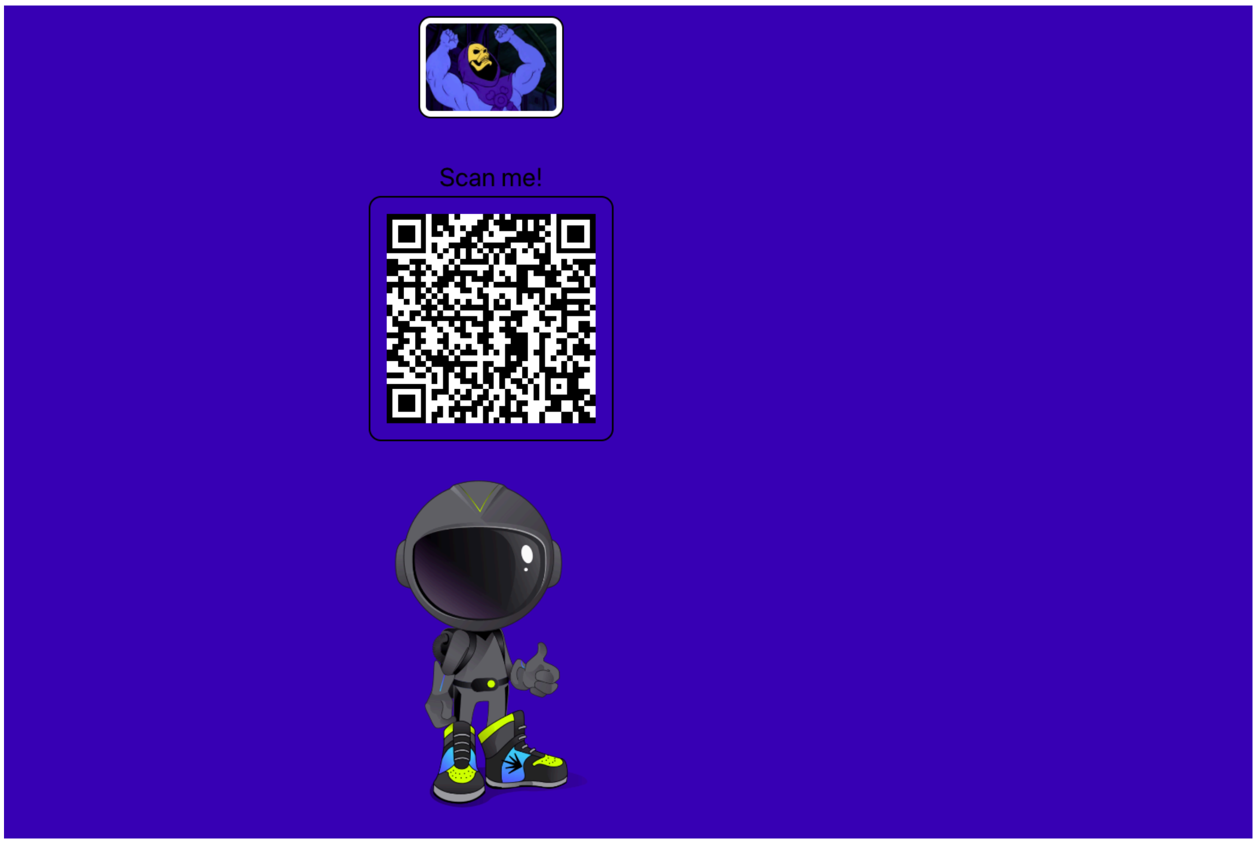
Maintainer
Nabeel Khashan

Tags
None · Add tag(s)

Variations
- Gray
- Blue
- Purple
- Red

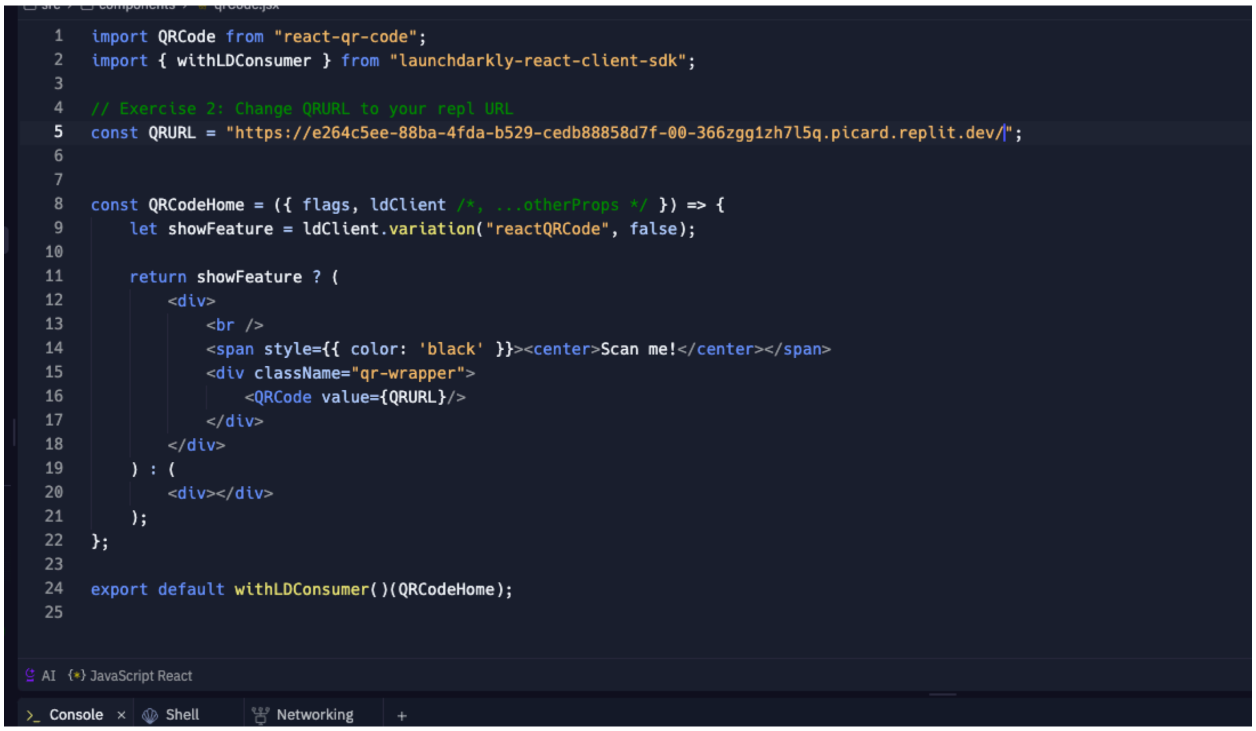
Release
No release • Create release pipeline

Code references
None on default branch • View all



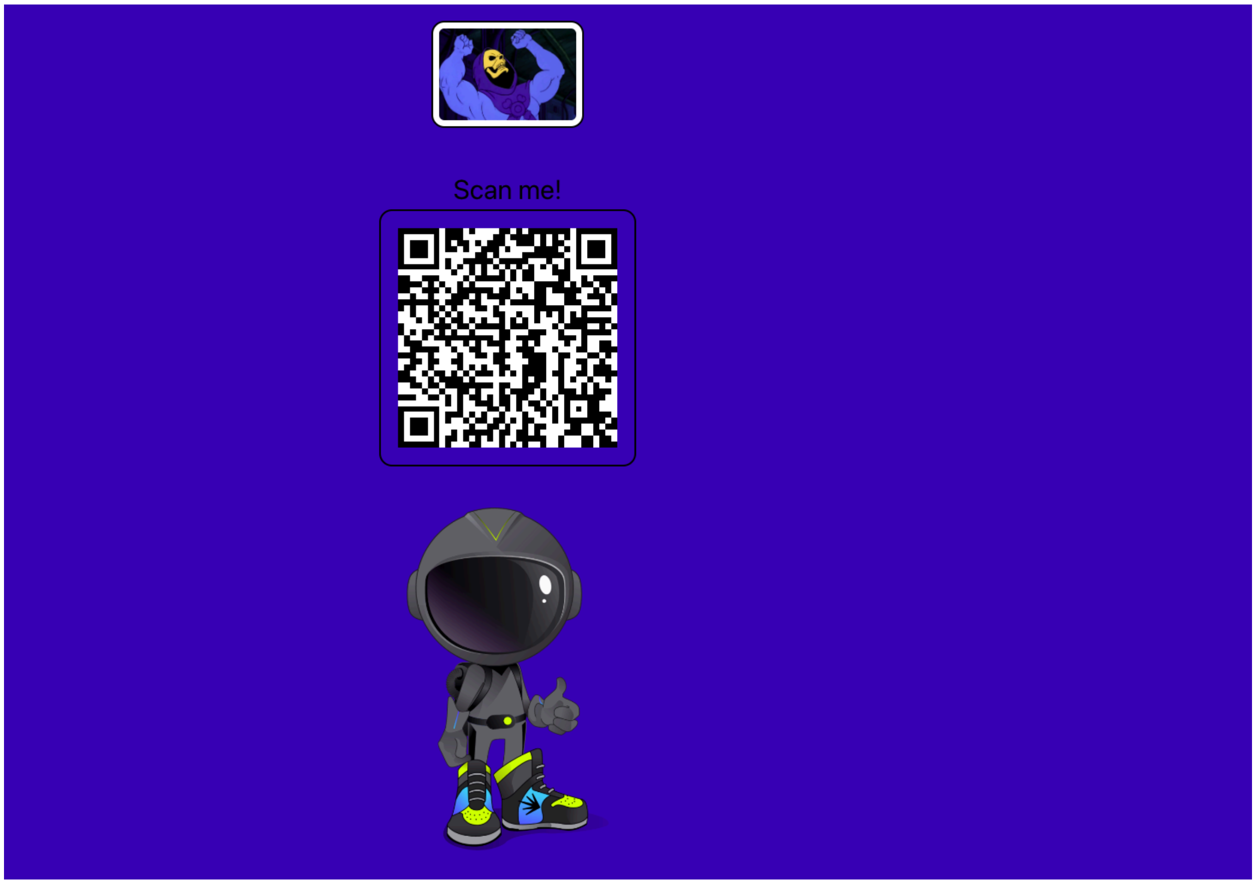
The QRCode required a bit of link manipulation. The original Link has this guy on it when you take a picture of your QRCode, singing his hit song from the 80's "Never Gonna Give You Up".

I changed the URL to Rick Astley to match the URL with the application I have been working on. Please see line 5 in the screenshot below shows the qrCode.jsx.

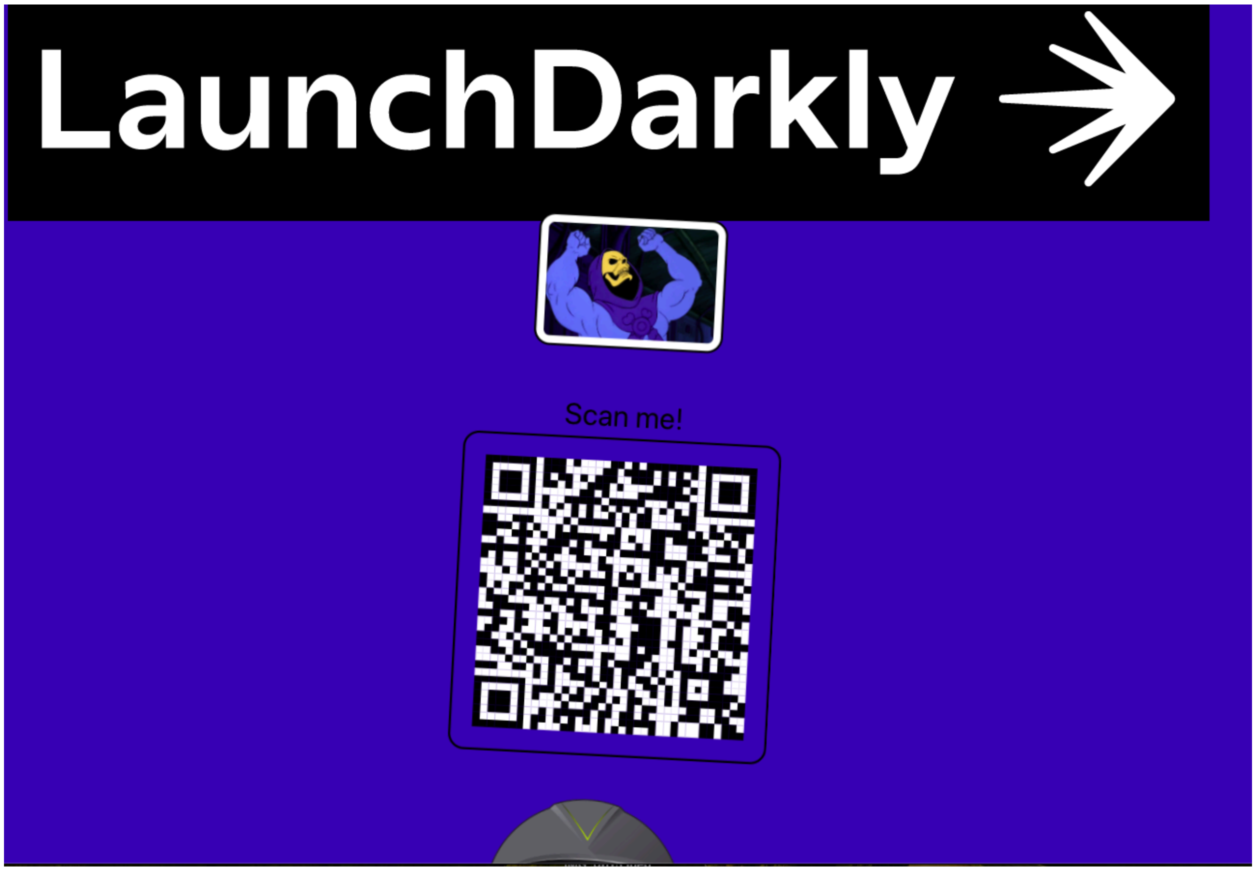


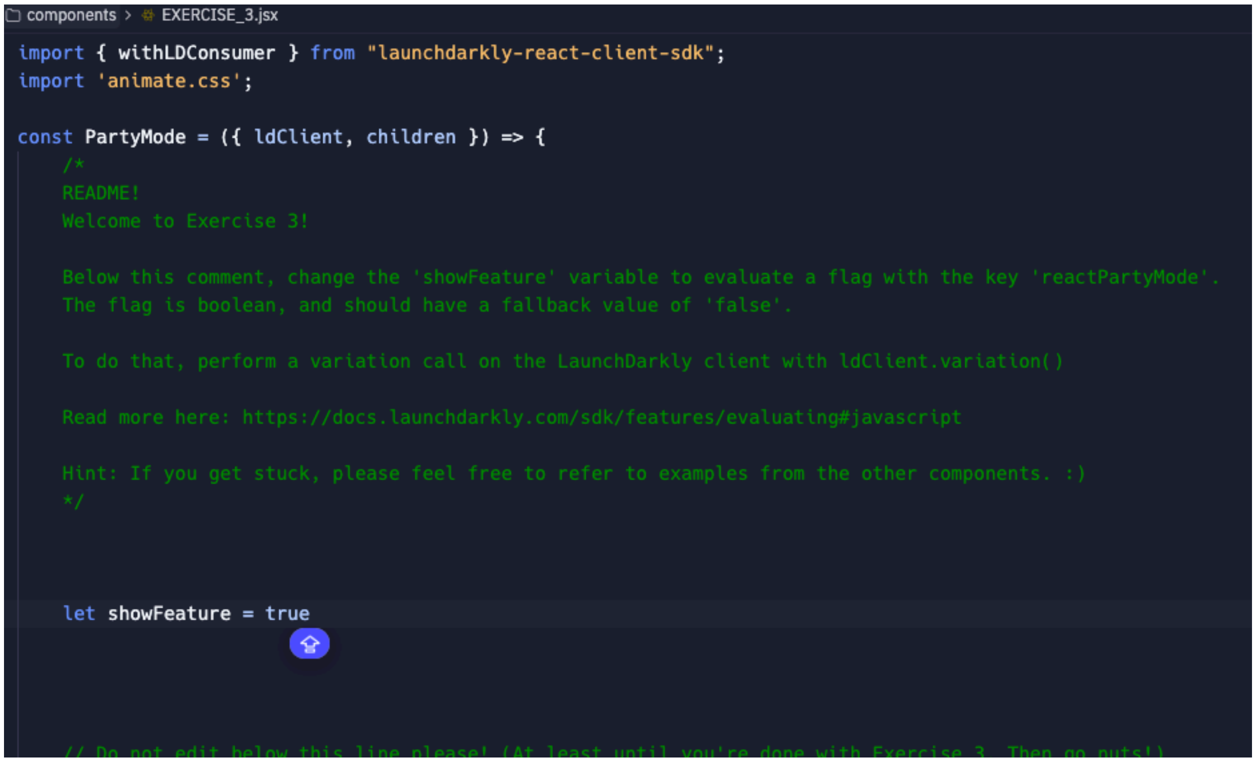
```
 1  import QRCode from "react-qr-code";
 2  import { withLDConsumer } from "launchdarkly-react-client-sdk";
 3
 4  // Exercise 2: Change QRURL to your repl URL
 5  const QRURL = "https://e264c5ee-88ba-4fda-b529-cedb88858d7f-00-366zgg1zh7l5q.picard.replit.dev/";
 6
 7
 8  const QRCodeHome = ({ flags, ldClient /*, ...otherProps */ }) => {
 9      let showFeature = ldClient.variation("reactQRCode", false);
10
11      return showFeature ? (
12          <div>
13              <br />
14              <span style={{ color: 'black' }}><center>Scan me!</center></span>
15              <div className="qr-wrapper">
16                  <QRCode value={QRURL}/>
17              </div>
18          </div>
19      ) : (
20          <div></div>
21      );
22  };
23
24  export default withLDConsumer()(QRCodeHome);
25
```

I was able to use my Iphone to take a picture of the code and this appeared on my cell phone.



The updated code in Exercise 3 for the Partymode flag. Below is the code for the Partymode Excercise. I Toggle the flag on in the Client and I changed the code below. The let showFeature = true now not false. Notice the qrCode and the Skeletor logo are dancing. These meet the requirements in the exercise for flags, variations, and Targets. The last piece I did was the curl script to Toggle the Astronaut off after turning it on with the unwanted illustration.

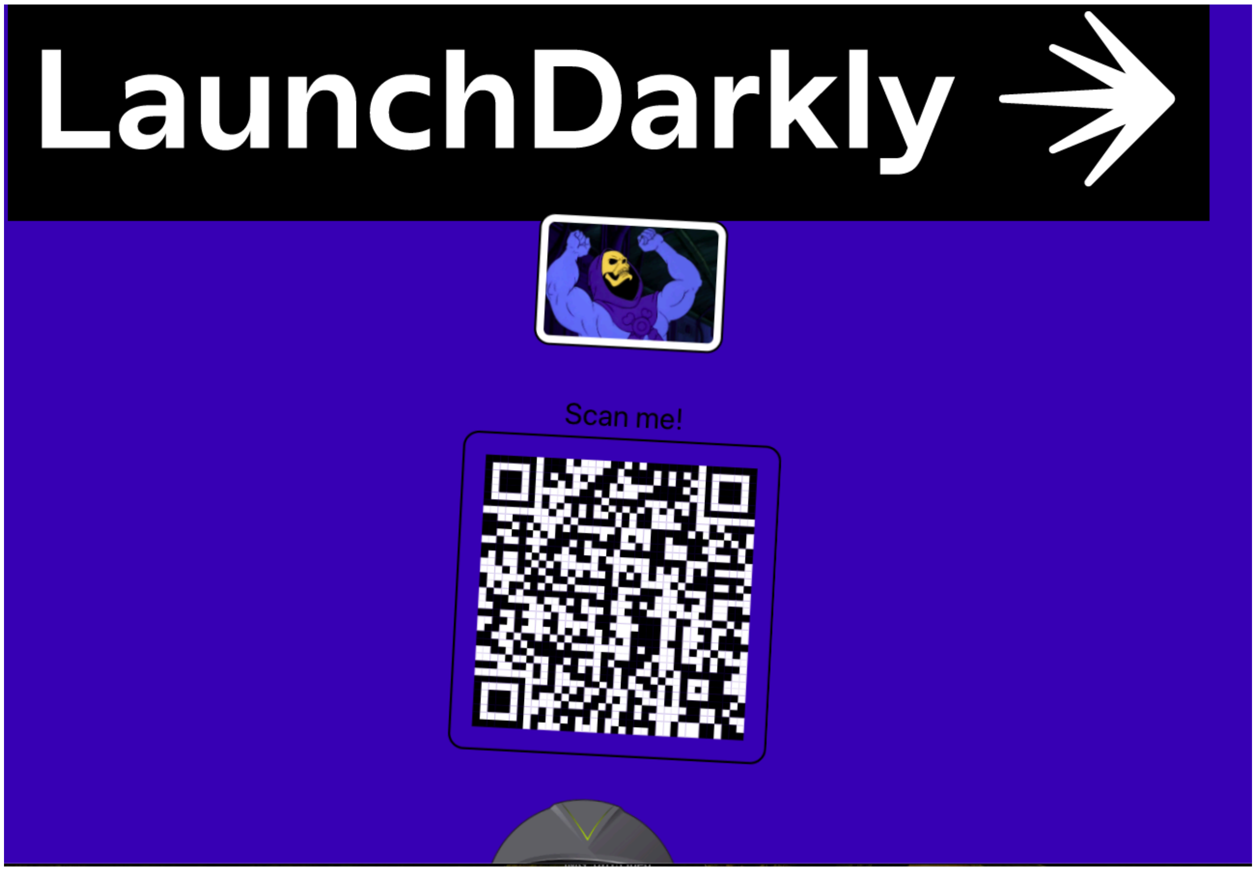



components > EXERCISE_3.jsx

```
import { withLDConsumer } from "launchdarkly-react-client-sdk";
import 'animate.css';

const PartyMode = ({ ldClient, children }) => {
    /*
    README!
    Welcome to Exercise 3!

    Below this comment, change the 'showFeature' variable to evaluate a flag with the key 'reactPartyMode'.
    The flag is boolean, and should have a fallback value of 'false'.

    To do that, perform a variation call on the LaunchDarkly client with ldClient.variation()

    Read more here: https://docs.launchdarkly.com/sdk/features/evaluating#javascript

    Hint: If you get stuck, please feel free to refer to examples from the other components. :)
    */



    let showFeature = true




    // Do not edit below this line please! (At least until you're done with Exercise 3. Then go nuts!)
```

The Curl script I wrote came from the Launch Darkly Documentation. I needed to install curl on my Mac using Homebrew. I used the project Key, the FeatureFlagKey, the environmentKey, and the API Key from my project to make this work. Unfortunately, I didn't get it to work exactly the way I wanted.

```
curl -i -X POST \

https://app.launchdarkly.com/api/v2/flags/replit-1/reactToggleAstr
onaut/triggers/test\
  -H Authorization: api-b50970ef-ea5c-4852-9e8e-cd0204f264d5 \
  -H Content-Type: application/json \
  -d '{
    "comment": “Turn Flag Off”,
    "instructions": [
      {
        "kind": "turnFlagOn”
      }
    ],
    "integrationKey": "generic-trigger"
  }'
```



LaunchDarkly

Scan me!