Hi there and thank you for taking the time to complete a code challenge with dott.
Good luck and we hope you will find it interesting!

Your task is to develop a single page application using typescript where users can upload a picture of a dog and see a gallery of pictures of dogs of the same breed. More specifically, the application should allow users to upload a picture and see it in a preview. The app should classify a dog present in the image based on their breed and display the result. Further, the application should showcase pictures of dogs of the same breed in a gallery below. The gallery should take all the available screen space and consist of lazy loaded images. The images that don't fit the current screen should be accessible with an "infinite scroll"-approach.

For image classification we suggest you use the pre-trained tensorflow.js-model available at:
https://github.com/tensorflow/tfjs-models/tree/master/mobilenet

For finding pictures of dogs based on their breed we suggest you refer to the following API: https://dog.ceo/dog-api/

We have set up a boilerplate repository that you can use as a template if you want help getting started. It contains some basic configuration to get you started and can help you get an idea of what we are expecting. The repo can be found here:
https://github.com/ridedott/frontend-assignment-boilerplate

We will judge the assignment mainly based on completion of scope and code quality with an emphasis on the latter. Focus on supporting the latest browsers (we will test it in the latest version of chrome). An aesthetic design can help differentiate you from other candidates, but it is not a requirement.

Our current tech stacks are based on either React or WebComponents which is why we expect the assignment to be delivered using either of the two technologies. If you choose to do it using WebComponents, feel free to use a small library to help you write the components (we use lit-element ourselves).

INTERNAL PURPOSES ONLY:

Estimated time: ~ 4 hours

Checklist:
Usability:
The app works: interface is usable, it doesn't freeze or throw any unhandled errors.
The requirements of the task are met.
There's a loading indicator while asynchronous operations are running.
There's an empty state in case the uploaded image was not processed correctly and there is no gallery to display.
Errors are handled: api call failure or breed not found.

Code quality:
There's a descriptive README file guiding how to launch the app and explaining the design choices.
API responses are typed.
At least one unit test is present.
Code is separated into distinct sections such that each section addresses a separate concern.

Bonus points:
Providing a link to solution hosted on a free hosting like heroku
Responsive layout (works for mobile and desktop)