[Have app loaded]
Hi I'm Mark Packham. Our app uses the progressive JavaScript framework, Vue for its front end and for the backend the NoSQL database, MongoDB with routing carried out by the Node.js web application framework, Express. All the code is stored in Github along with setup documentation. The educational site takes in data from 4 apis, 3 from external sites and one created by ourselves that our backend framework handles. Cameron created a seeds file we can run whenever we need to repopulate our database during testing. The external apis are the country one we used during labs, REST Countries. One that displays maps, from the Open Street Map and one that provides information for the country quiz that Neil and Nathan worked on, from the Open Trivia Database api.

We're using fairly popular node packages on our backend, Nodemon to save us from constantly having to restart Express when dealing with updates, Body-Parser to handle the JSON, obviously a MongoDB driver to interact with our database and Cors so we can pass information between our locally hosted Vue and Express apps. The REST request testing app, Insomnia proved helpful for experimenting with our Express based CRUD requests before we got round to building the necessary web forms in Vue.

[Load the front page to show animations]
The front end loads with a lot of gimmicky CSS animations mainly to make it more memorable to an audience, a bit like a cheesy advert giving the app 90s aesthetic. With a simple color scheme of green and blue to represent land and sea along with the use of standard web safe fonts. A fair amount of html symbols used to give the users eyes a break from all the text data.

The bulk of components are held in the homepage App.vue, **[App homepage]** thankfully we're not dealing with vast amounts of data & Vue has the ability to minify its CSS and JavaScript in a script intended for production environments to improve performance. We keep CSS in CSS specific files that are imported into components rather than using inline styling. This is to stop components from getting too bloated and hard to debug containing, CSS, JavaScript and Html.

Countries are divided into specific regions with sub regions and users can click on a country in a sub region to receive more information about it. **[Demo France]**

There's also a modal made purely with CSS to provide the user with even greater amounts of information about a country. **[Modal Example - New Zealand]**
The countries' latitude and longitude data from the country api is used in order to populate the url of an open source map site which gives us the map location. We have to use string interpolation and Vue's v-bind: on the src file to achieve this. And place the map inside an iFrame.
[Map String Interpolation]

The site is capable of displaying random facts about countries by clicking a button to "Get a fact" **[Country Facts - Get]**. If you're not fond of that specific fact you can delete it or create your fact about a country in a text field. Obviously the "Delete the fact" button doesn't materialise till you "Get a fact" first. So these are typical Get, Post and Delete Requests that trigger the respective Read, Create and Delete actions in the database.

Now onto Nathan who will cover the country quiz.