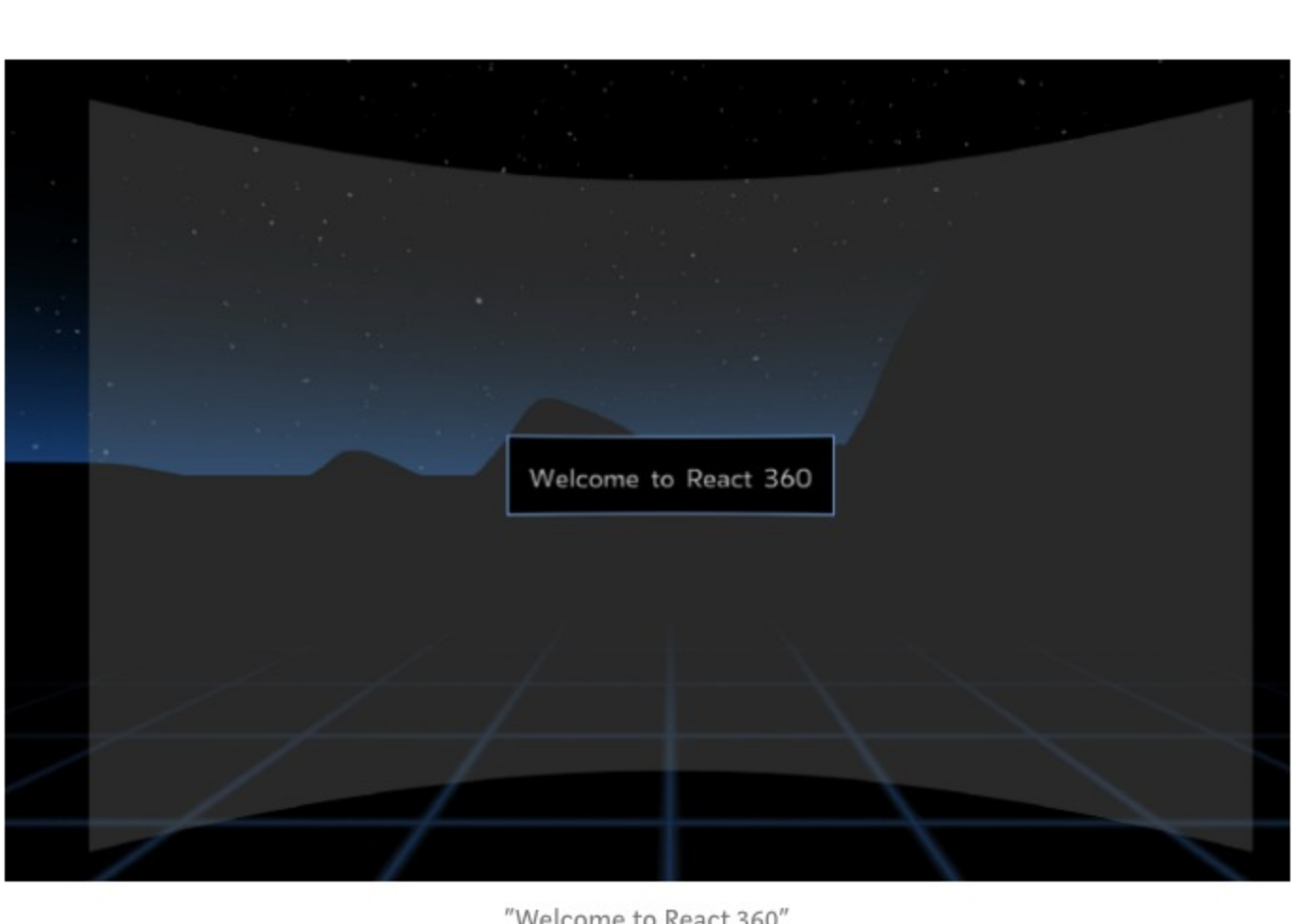
Exploring React 360 VR Library

React 360 is a framework for the creation of 3D and VR user interfaces

Indrek Lasn in Better Programming Follow
Jul 11 · 4 min read ★



"Welcome to React 360"

React 360 is a library built on top of React. It is designed to simplify the creation of complex UI. React 360 allows you to use familiar tools and concepts to create immersive 360 content on the web.

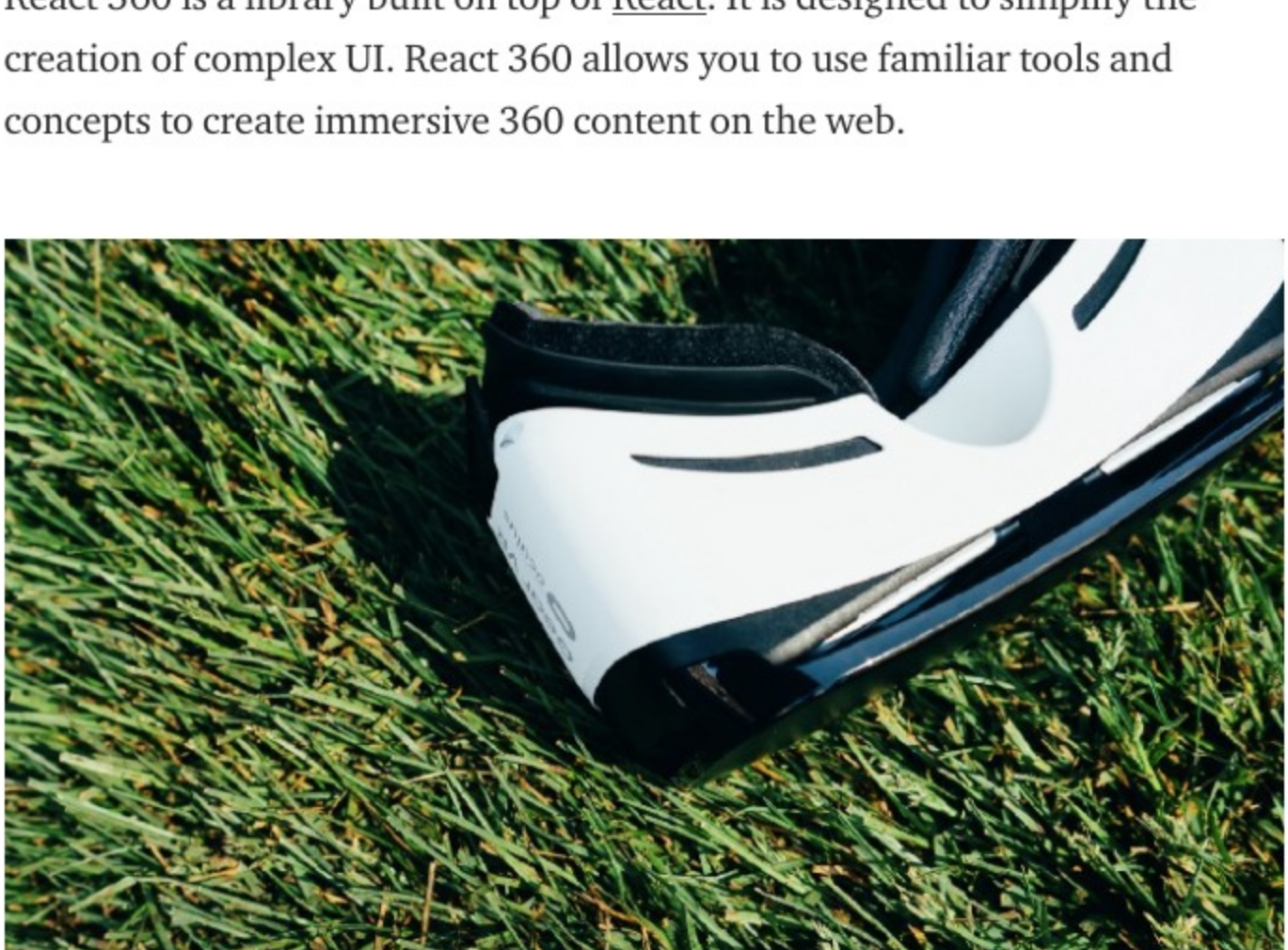


Photo by Scott Webb on Unsplash

· · ·

How Is React 360 Different From React VR?

In early 2017, Facebook and Oculus released a JavaScript library called React VR, which was designed for the creation of 3D and VR experiences in the web browser. At the same time, Oculus began using a native C++ version of the framework to build its own first-party apps.

Over time, the APIs of the two projects diverged as we addressed the different needs of the frameworks. To avoid confusion between the two systems, the open source framework has been renamed React 360. This better reflects its use case: the creation of immersive 360 experiences that can be consumed across PC, Mobile, and VR devices.

If you previously used React VR, you should find React 360 to be very similar. We've simplified a lot of the more common workflows, such as placing 2D UI elements in 3D space, and optimized performance for these use cases. See full comparison here.

We only need a browser and a code editor to get VR apps to work. Cool!

· · ·

Getting Started

You'll need to install Node.js. While React 360 runs in your web browser, the build pipeline that bundles up your code relies on Node.

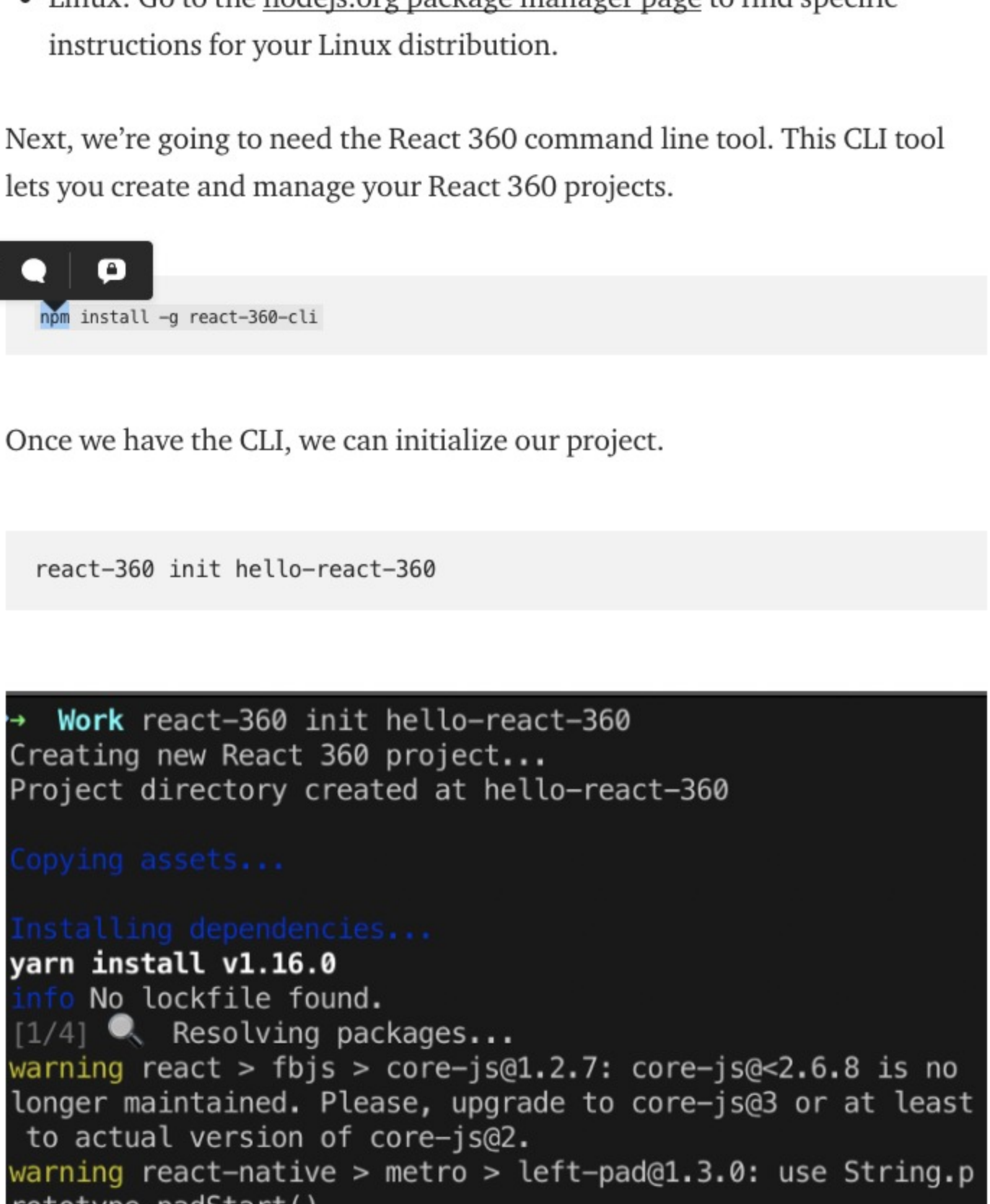
- Mac: On Mac, we recommend installing Node.js through Homebrew.
- Windows: Get the Windows installer from the nodejs.org download page.
- Linux: Go to the nodejs.org package manager page to find specific instructions for your Linux distribution.

Next, we're going to need the React 360 command line tool. This CLI tool lets you create and manage your React 360 projects.

```
npm install -g react-360-cli
```

Once we have the CLI, we can initialize our project.

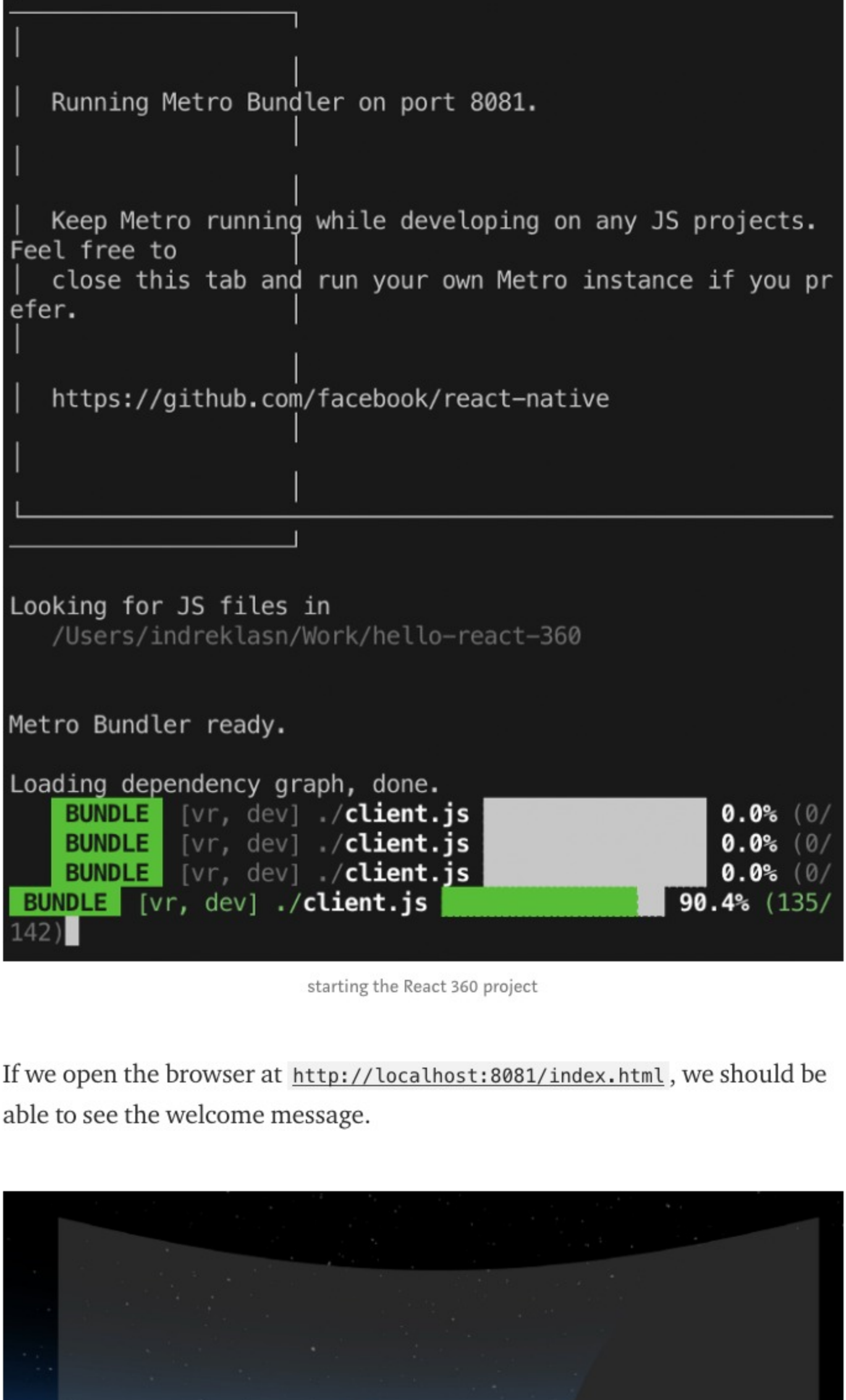
```
react-360 init hello-react-360
```



initializing the React 360 project

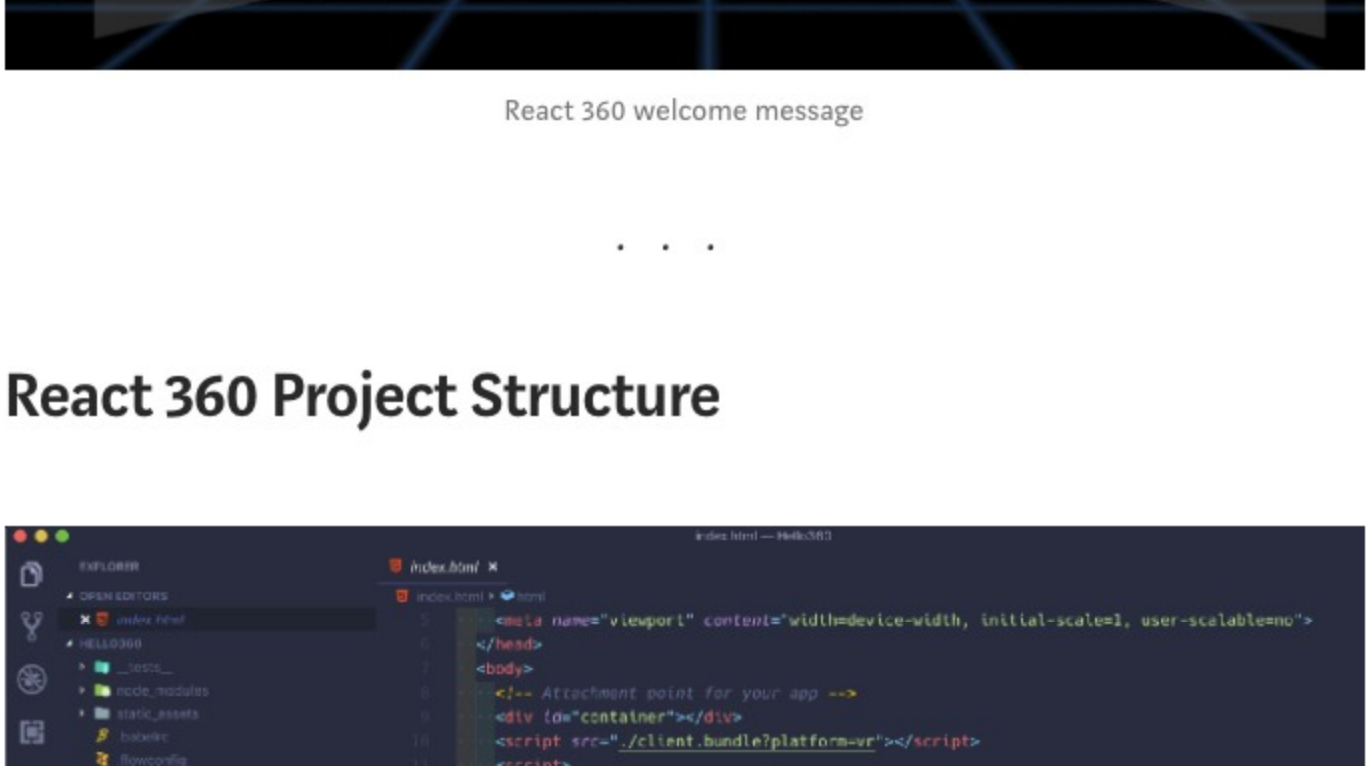
You can start your project now by running `npm start`

```
npm start
```



starting the React 360 project

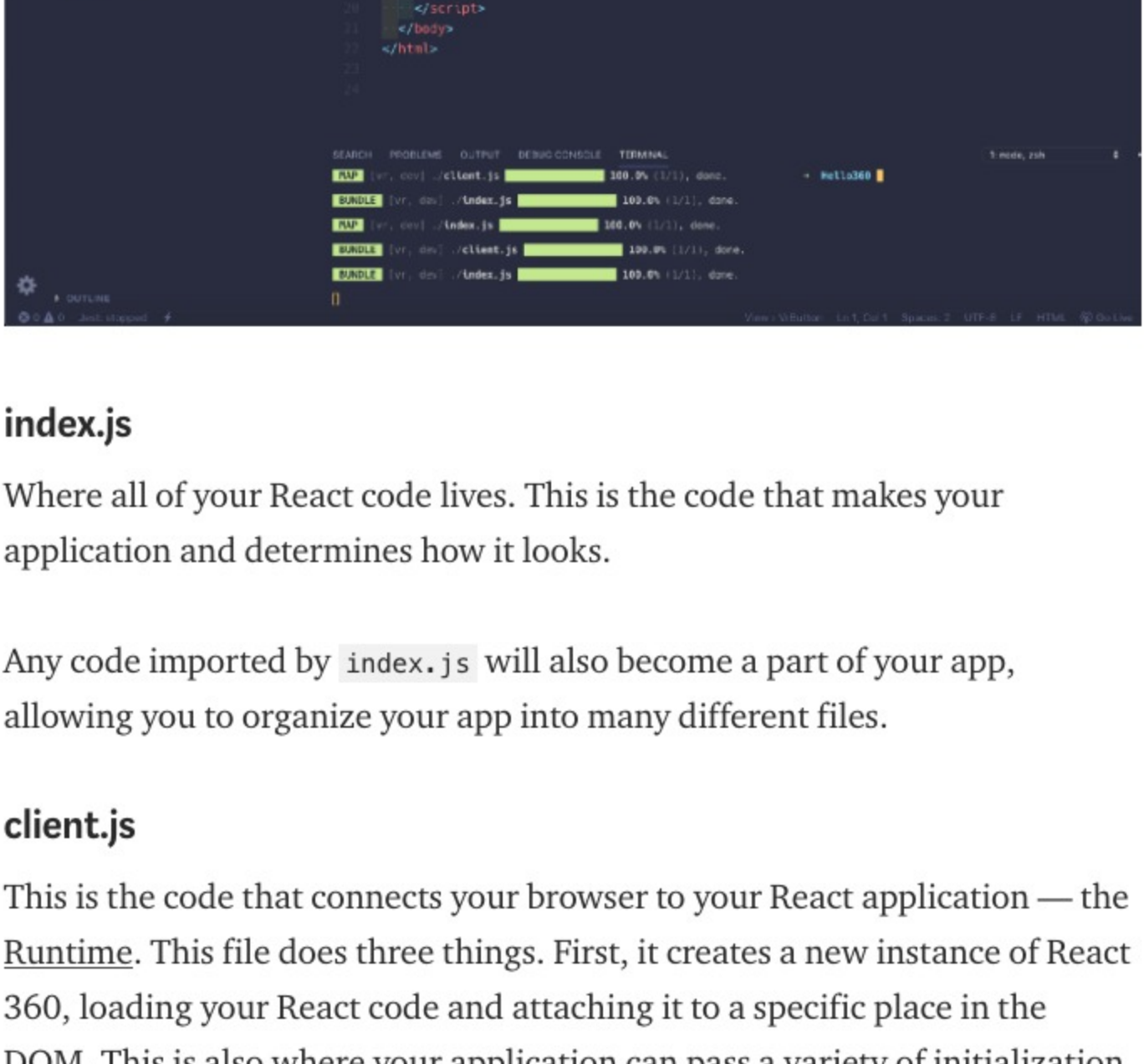
If we open the browser at http://localhost:8081/index.html, we should be able to see the welcome message.



React 360 welcome message

· · ·

React 360 Project Structure



index.js

Where all of your React code lives. This is the code that makes your application and determines how it looks.

Any code imported by `index.js` will also become a part of your app, allowing you to organize your app into many different files.

client.js

This is the code that connects your browser to your React application — the Runtime. This file does three things. First, it creates a new instance of React 360, loading your React code and attaching it to a specific place in the DOM. This is also where your application can pass a variety of initialization options.

`index.html`

This is the web page you load when viewing your application. All it does is provide a point to mount your JavaScript code. This is intentional. Most functionality is left out of HTML, so that you can easily integrate your React 360 application into server-render pages or existing web apps.

· · ·

Creating a Counter

For the short and sweet demo, let's create a counter. The React 360 library uses similar concepts as React Native. If you're interested in React Native concepts, check out this tutorial.

First, we need to import the `View`, `Text`, and `VrButton` elements from the react-360 library.

```
import React from 'react';
import {
  AppRegistry,
  StyleSheet,
  Text,
  View,
  VrButton
} from 'react-360';
```

index.js hosted with ❤ by GitHub view raw

Second, let's declare our initial `state` and create the `decrement` and `increment` functions.

```
export default class Hello360 extends React.Component {

        state = {
                count: 0,
        };

        // This method increments our count, triggering a re-render
        incrementCount = () => {
                this.setState({count: this.state.count + 1});
        };

        // This method decrements our count, triggering a re-render
        decrementCount = () => {
                this.setState({count: this.state.count - 1});
        };


        render() {
                return ();
        }
};
```

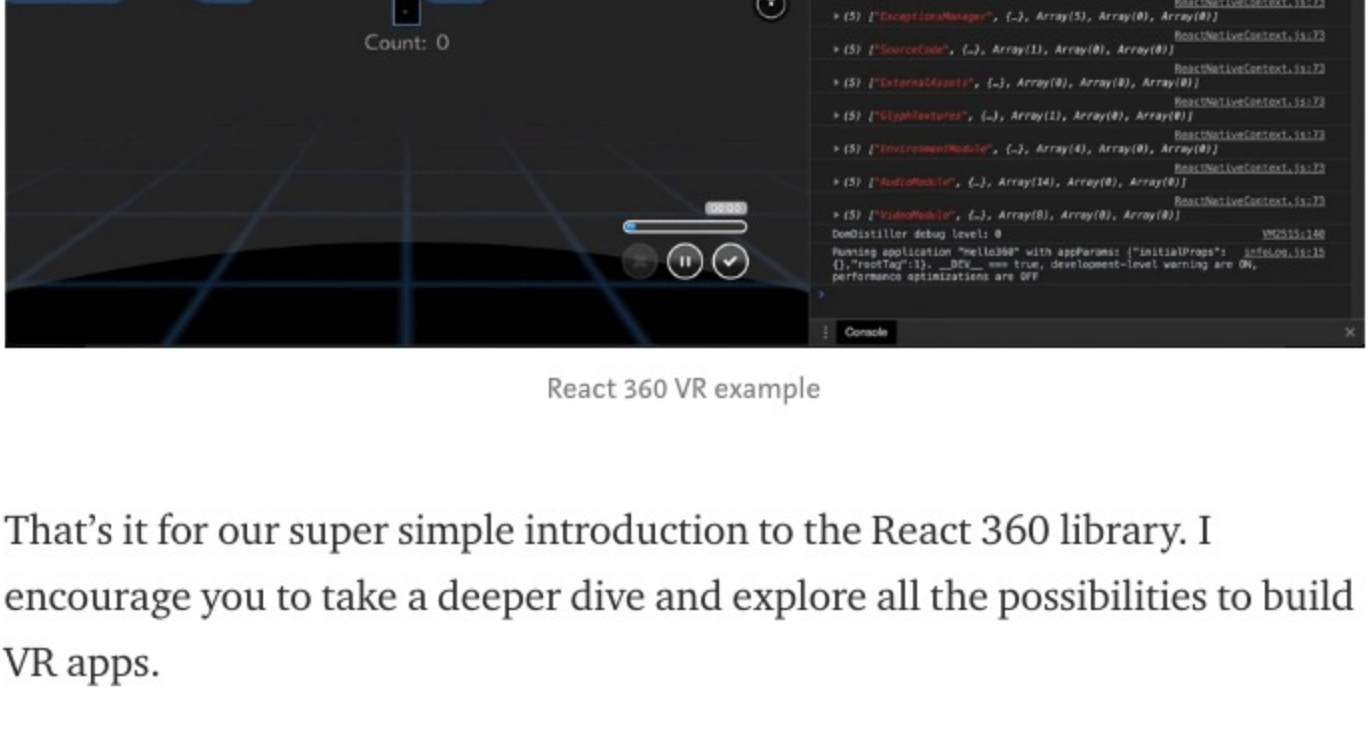
index.js hosted with ❤ by GitHub view raw

Pretty standard stuff. Next up, we're going to use the React 360 elements we imported earlier.

```
render() {
  return (
    <View style={styles.panel}>
    <VrButton
      onClick={this.incrementCount}
      style={styles.greetingBox}
    >
      <Text>+</Text>
    </VrButton>
    <VrButton
      onClick={this.decrementCount}
      style={styles.greetingBox}
    >
      <Text>-</Text>
    </VrButton>
    <Text style={styles.greeting}>
      {`Count: ${this.state.count}`}
    </Text>
  </View>
  );
}
```

index.js hosted with ❤ by GitHub view raw

Once we have the markup, we should be able to interact with our 360 counter.



React 360 VR example

That's it for our super simple introduction to the React 360 library. I encourage you to take a deeper dive and explore all the possibilities to build VR apps.

· · ·

Where to Go From Here

I highly recommend reading through the official React 360 documentation for an in-depth dive.

React 360 · Create amazing 360 experiences
Create amazing 360 experiences
Create amazing 360 experiencesfacebook.github.io

WebVR API
WebVR provides support for exposing virtual reality devices - for example head-mounted displays like the Oculus Rift or...
developer.mozilla.org

VR Applications using React 360
If you've used React, you know how it can provide smart solutions to complex problems. And how exciting that can be...
egghead.io

Introduction to React 360
React 360 (or React VR) is a library used to create virtual reality experiences with React and uses Three.js for...
alligator.io

Thanks for reading! ❤