React Native

acm.hack

LA HACKS 2023

Attendance!

Best attendance will receive a prize during the closing ceremony!

https://tinyurl.com/lahacks23attendance

Link to slides:

http://links.uclaacm.com/hoth9-react-native-slides

Link to README:

http://links.uclaacm.com/hoth9-react-native-readme

acm.hack

The Game Plan

- Intro to React Native
- Basic setup
- JSX
- Views and Components and Stylesheets
- Custom Components
- Functionality

DISCLAIMER

- We gonna be zoomin
- An understanding of javascript and HTML will greatly help.
  - Highly recommend checking out workshop on Javascript/HTML/CSS if you're unfamiliar with these already.

~~cross-platform~~
mobile development

React Native

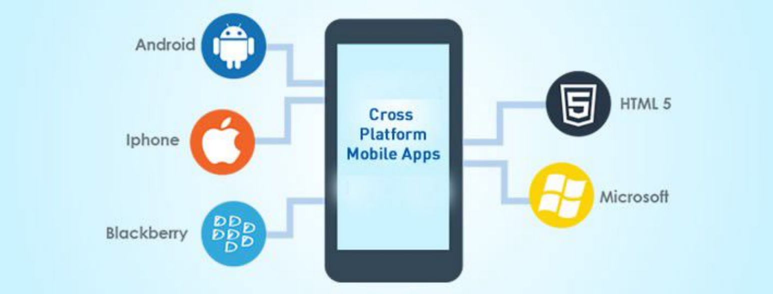
How does one natively react pls help

Native Mobile Development

- **Native apps**: apps installed directly on the mobile device
- (Previously) only able to be developed with *platform-specific languages/tools*
  - **iOS**: Objective-C, Swift
  - **Android**: Java, Kotlin

Cross-Platform Development

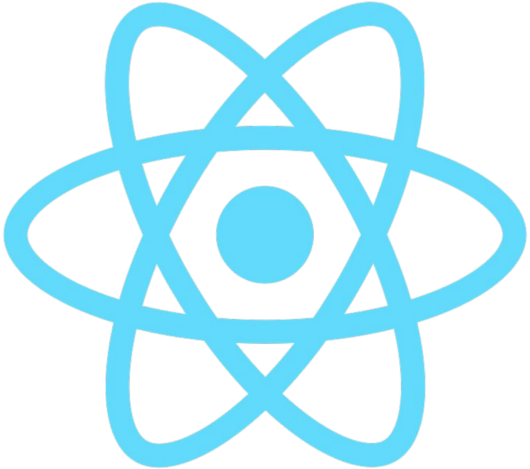
- Build apps across multiple platforms *using one codebase*
- Multiple solutions:
  - React Native
  - Ionic
  - Flutter
  - Xamarin



acm.hack

React Native

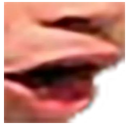
- JavaScript tool for building **native** mobile apps
- Cross-platform
    - JavaScript code is *translated* into each platform's native code
- Backed by Facebook
- Lots of community support
- https://reactnative.dev/

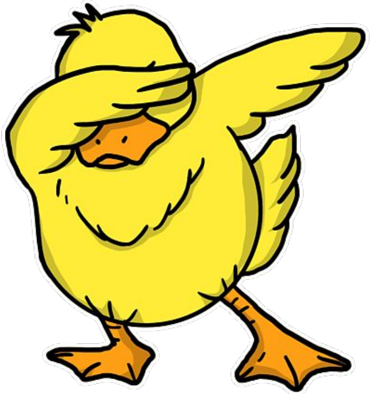


A fat TL;DR (Summary)

IOS and Android apps are made in different programming languages.

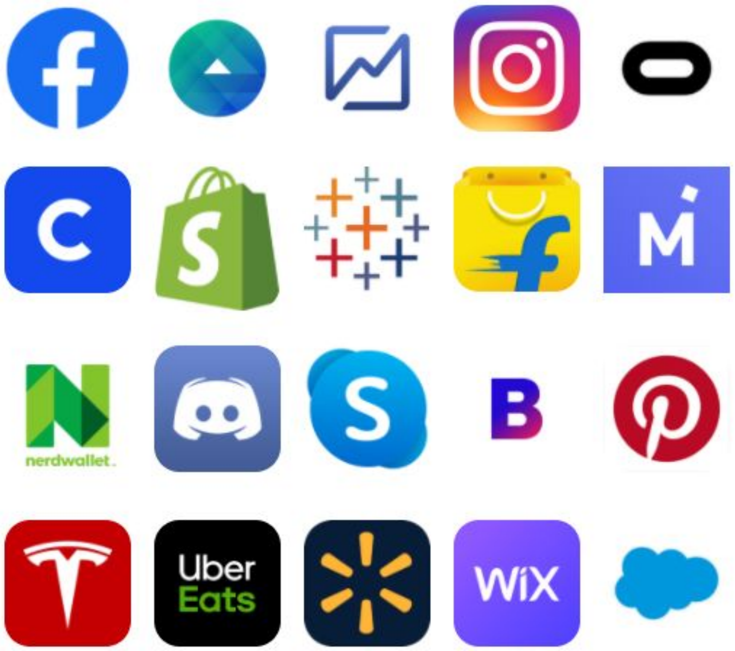
- Can we just learn one language/technology ~~a~~cross ~~all~~ platforms~~?~~





React Native is a cross-platform Javascript tool that lets us!

Apps using React Native



Basic Setup

STUFF TO INSTALL (sorry kind of a lot):

- VSCode (Text editor for code)
- Node.JS (DOWNLOAD LTS, NOT CURRENT)
- Expo Go (This is an app you'd download to your phone on the app store unless you have a simulator on computer)

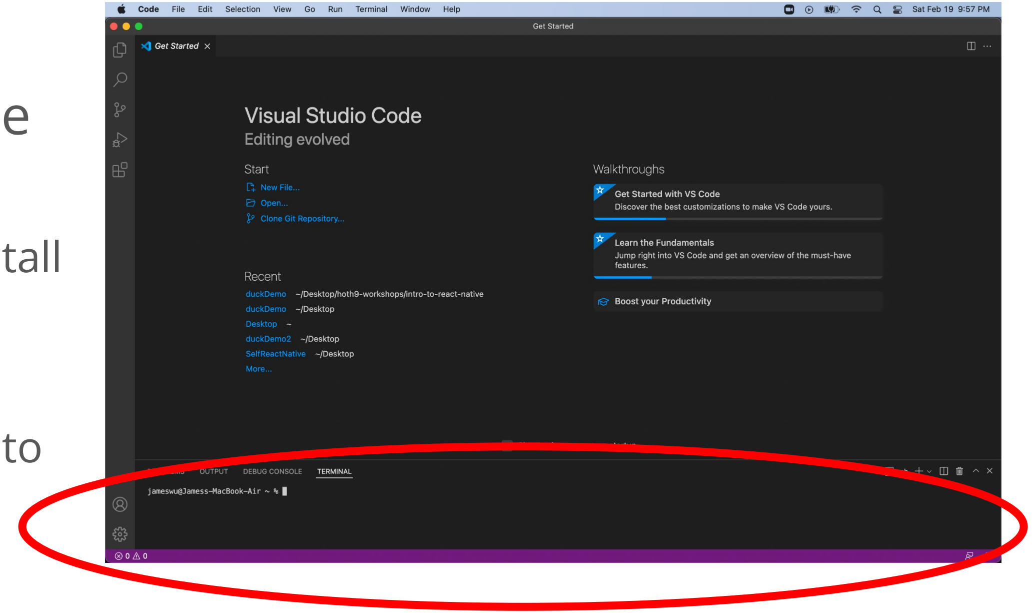
Feel free to pause and get these installed before continuing!

Create your first React Native app!

Open VSCode.

- Open the terminal in VSCode
- FOR WINDOWS:
  - Type “npm i -g expo-cli “ to install expo
- FOR MAC:
  - Type “sudo npm i -g expo-cli “ to install expo

You only need to do this once!



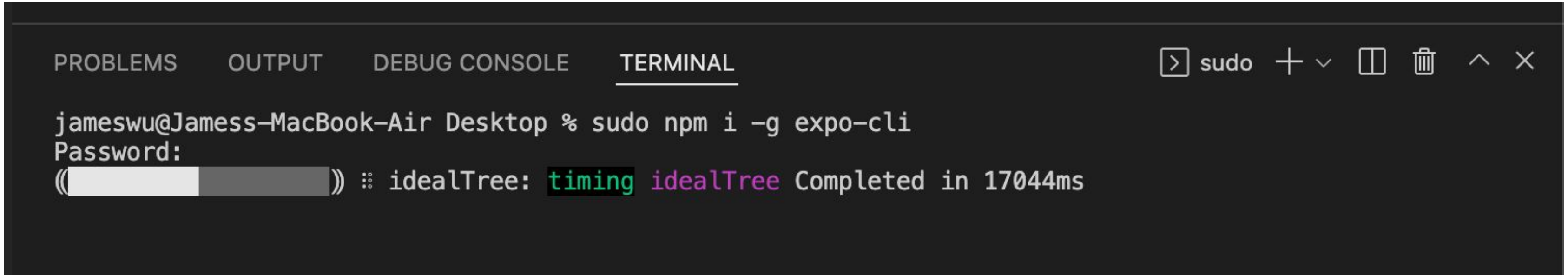
Visual Studio Code
Editing evolved

Start
New File...
Open...
Clone Git Repository...

Recent
duckDemo ~/Desktop/hoth9-workshops/intro-to-react-native
duckDemo ~/Desktop
Desktop ~
duckDemo2 ~/Desktop
SelfReactNative ~/Desktop
More...

Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Jump right into VS Code and get an overview of the must-have features.
Boost your Productivity

```
jameswu@Jamess-MacBook-Air ~ % █
```

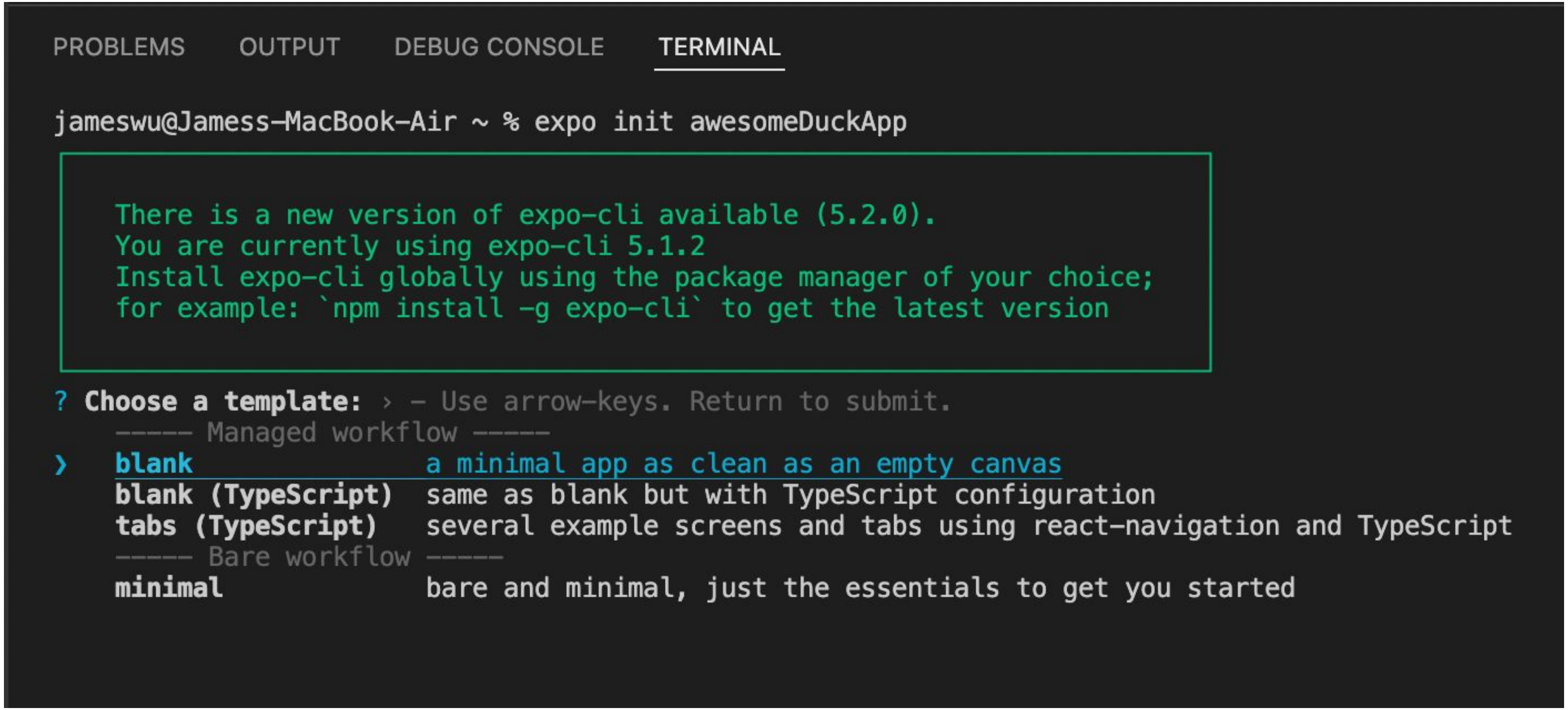


```
jameswu@Jamess-MacBook-Air Desktop % sudo npm i -g expo-cli
Password:
(██████████░░░░░░░░░) ⠏ idealTree: timing idealTree Completed in 17044ms
```

Create your first React Native app pt. 2

- Navigate to a folder on your computer you want to put your project in
- After this, type: “npx create-expo-app [name]”, where name is whatever you want to call your project.
- Choose a blank template.
- To run code, type: “Expo start”

You can use a simulator or scan the QR code with your phone to open your app on your phone!



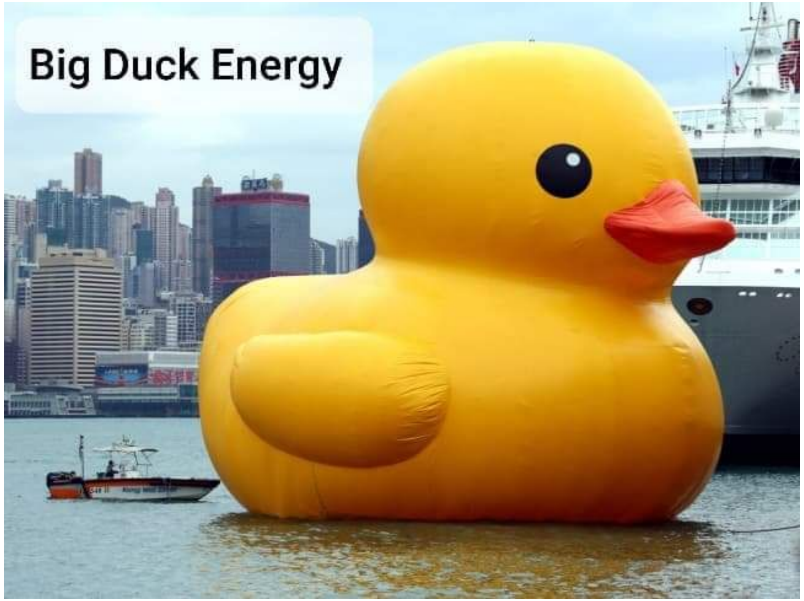
```
jameswu@Jamess-MacBook-Air ~ % expo init awesomeDuckApp

   There is a new version of expo-cli available (5.2.0).
   You are currently using expo-cli 5.1.2
   Install expo-cli globally using the package manager of your choice;
   for example: `npm install -g expo-cli` to get the latest version

? Choose a template: › - Use arrow-keys. Return to submit.
    ----- Managed workflow -----
❯   blank                 a minimal app as clean as an empty canvas
    blank (TypeScript)    same as blank but with TypeScript configuration
    tabs (TypeScript)     several example screens and tabs using react-navigation and TypeScript
    ----- Bare workflow -----
    minimal               bare and minimal, just the essentials to get you started
```

acm.hack

Time to fill this with stuff



Javascript and XML’s love child

Javascript – a programming language.

XML - a markup language. (Tells browsers how to format information. Very very similar to HTML.)

Javascript XML (JSX) - Their baby

JSX basically lets us write UI in Javascript.

JSX Cont.

Basically, JSX lets us store “markup” code into variables, and directly write it into JavaScript.

- Remember, “markup” is just a fancy term for “stuff that formats the screen.”
- Examples: View, Text, etc. (Demo)

Note: Save the file after typing code to instantly see your changes!

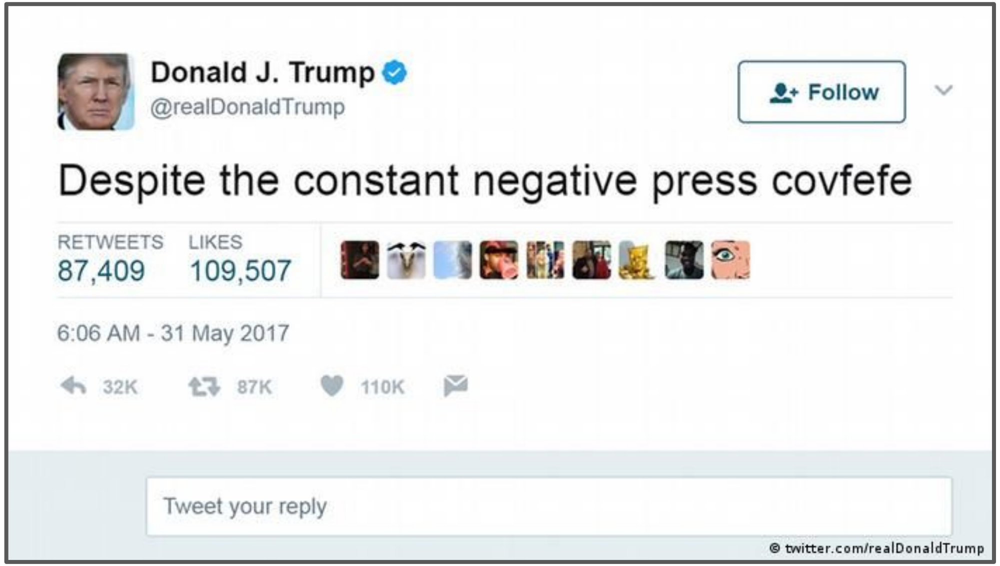
Components

“View” and “Text” are examples of components. Think of components like basic app elements provided to us!

Ex:

- Text
- Image
- Button
- ScrollView
- TextInput

...and TONS more.



Donald J. Trump
@realDonaldTrump

Despite the constant negative press covfefe

RETWEETS 87,409 LIKES 109,507

6:06 AM - 31 May 2017

Text and Image

Two bread and butter components that I’ll quickly demonstrate!

Text:

```
<Text> Ducks go quack </Text>
```

Image:

```
<Image source={require(‘./assets/duck.png’)} />
```

OR

```
<Image source={{uri: ‘https://someLinkToImage.png’,}}/>
```

Imports

Remember how React Native is cross-platform?

- Uses certain bits and pieces from different places
- To use these things, you must IMPORT them.
  Ex: To use an “Image” component, it has to be imported at the top of the code.
- React Native usually does this for us, but sometimes it misses something!

DEMO

Views

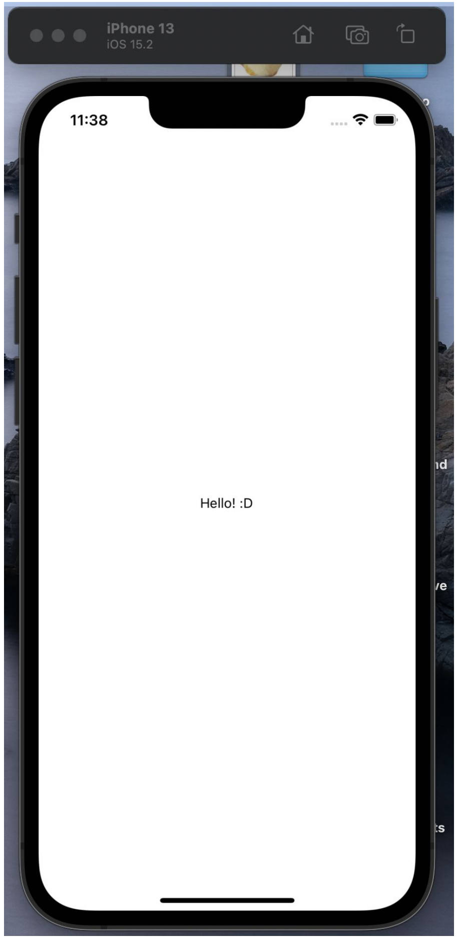
Views are containers that hold stuff!

- They make stuff show up on the screen
- Invisible component

```
<View>

<Text> Hello :D </Text>

</View>
```



Hello! :D

What is all this extra stuff

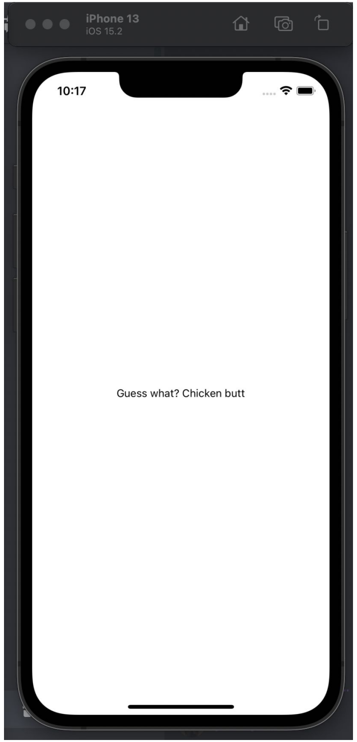
```
export default function App() {

    return(

    <View style={styles.container}>
    <Text> Guess what? Chicken butt </Text>
    </View>

    );

}
```



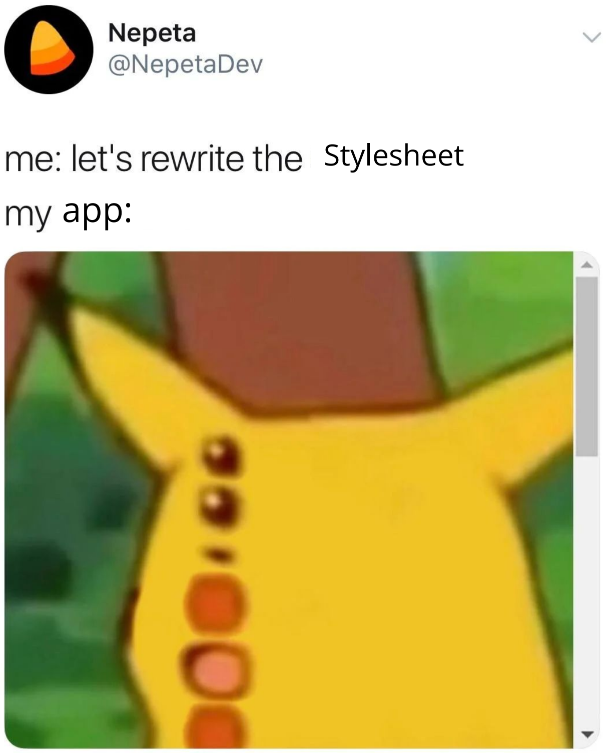
Guess what? Chicken butt

Let’s make it pretty!

Stylesheets

Easy (and most common) way to format views

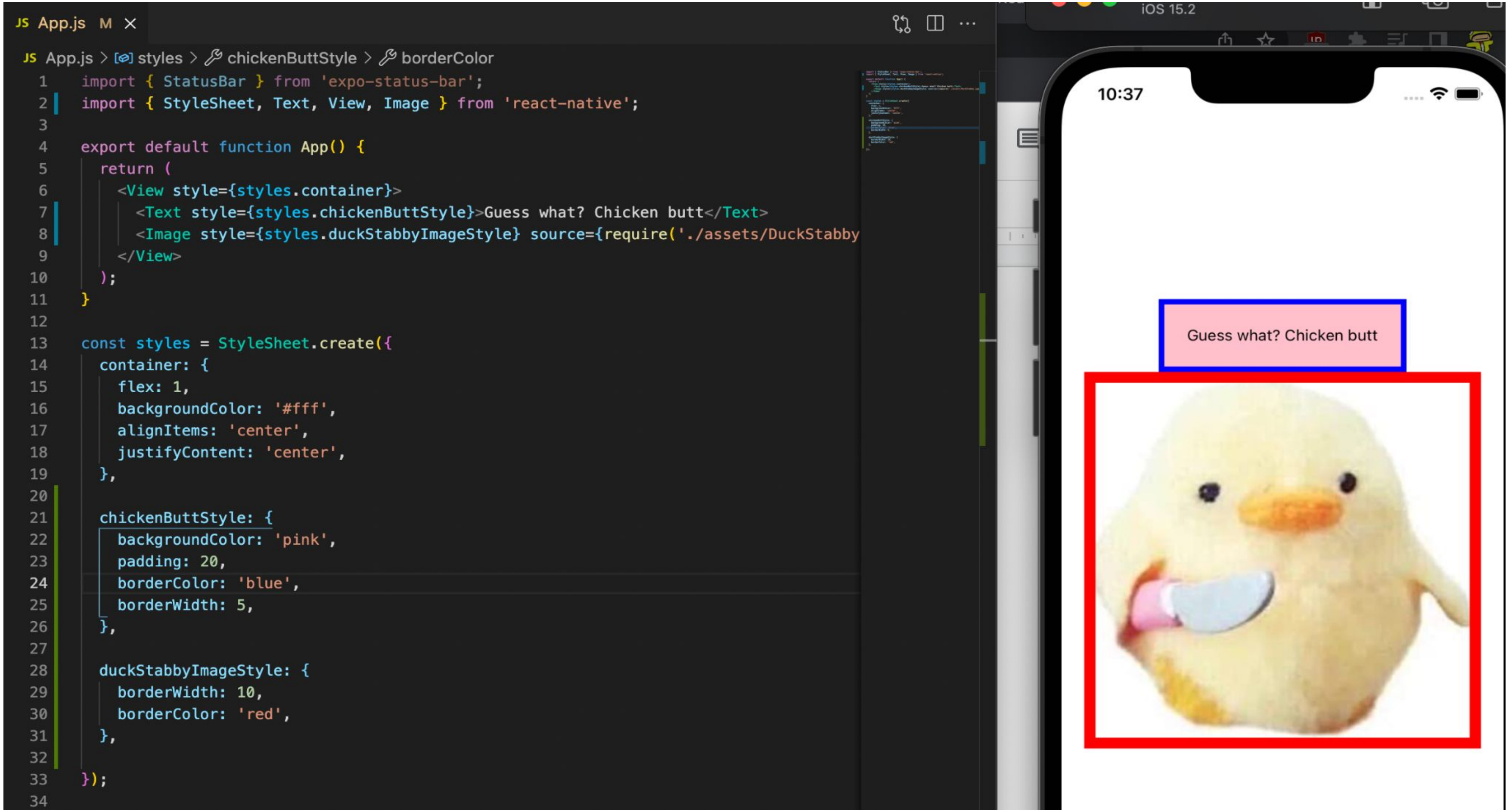
- Put all the formatting for a view under one “style” that you get to name
- You can have multiple styles in one style sheet
- Using JSX within the code (curly braces!)



Nepeta
@NepetaDev

me: let's rewrite the Stylesheet

my app:



```
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View, Image } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text style={styles.chickenButtStyle}>Guess what? Chicken butt</Text>
      <Image style={styles.duckStabbyImageStyle} source={require('./assets/DuckStabby
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },

  chickenButtStyle: {
    backgroundColor: 'pink',
    padding: 20,
    borderColor: 'blue',
    borderWidth: 5,
  },

  duckStabbyImageStyle: {
    borderWidth: 10,
    borderColor: 'red',
  },

});

```

Guess what? Chicken butt

Demo

Custom Components

What are they: functions **you** make, that you can insert in multiple places later in your code.

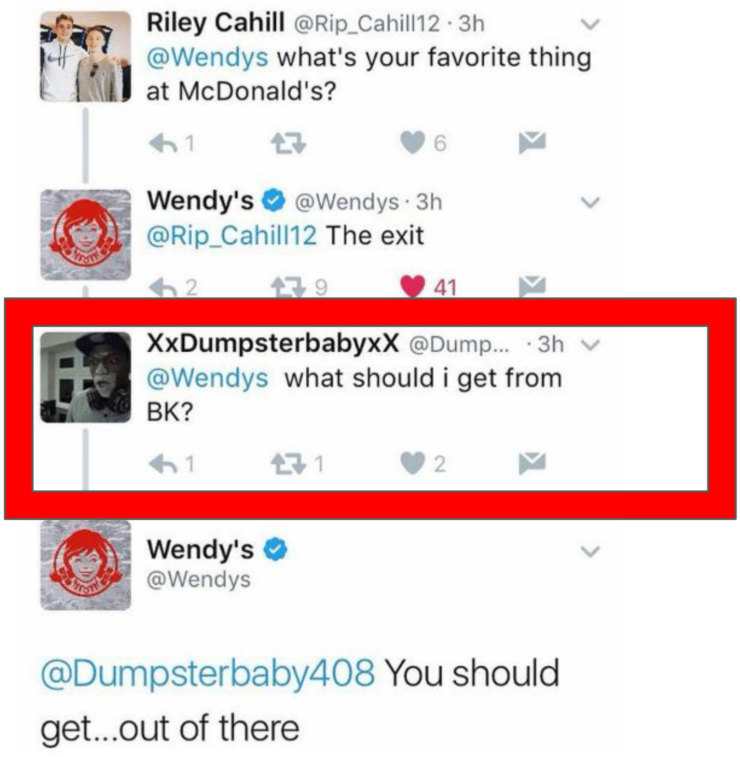
Why this matters: Lets you call one function to reference a whole bunch of code, instead of typing the same code in a bunch of areas.

Custom Components Continued -> alliteration!

May be useful to put multiple parts of the same UI element onto the screen.

(UI means user interface– just the stuff the user sees.)

COMPONENT



Riley Cahill @Rip_Cahill12 · 3h
@Wendys what's your favorite thing at McDonald's?

Wendy's @Wendys · 3h
@Rip_Cahill12 The exit

XxDumpsterbabyxX @Dump... · 3h
@Wendys what should i get from BK?

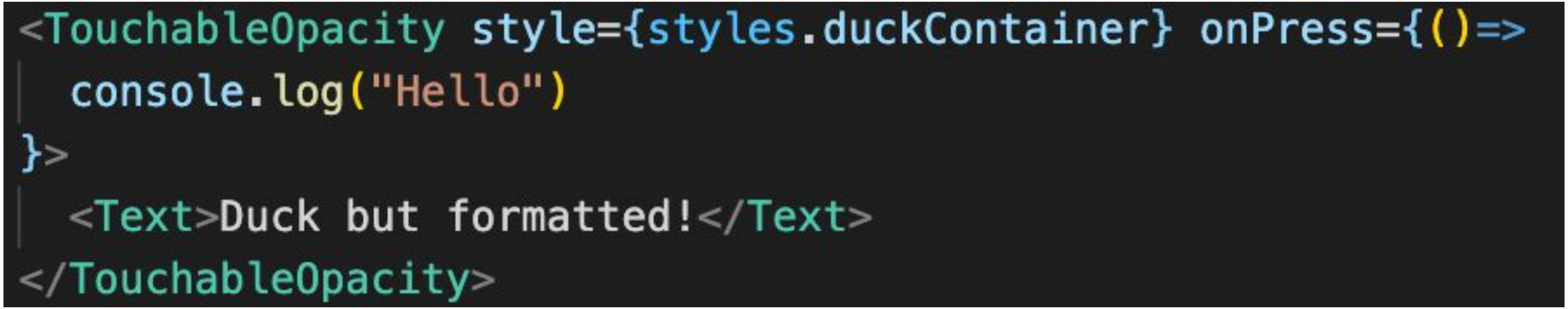
Wendy's
@Wendys

@Dumpsterbaby408 You should get...out of there

Now, let’s make our app do something.

TouchableOpacity

- Basically a button with a smidge of flair
- Component that user can touch
- Slightly changes color! Very cool.



```
<TouchableOpacity style={styles.duckContainer} onPress={()=>
  console.log("Hello")
}>
  <Text>Duck but formatted!</Text>
</TouchableOpacity>
```

DEMO

Screen changes

What if we wanted to have the screen change after the user presses a button?

- Right now, the screen won't do anything.
- We have a button, but no action for it.

useState()

useState() is a something called a “hook”.

- Basically, lets us update the screen after user interacts with stuff.
- When it is called, it re-renders the whole screen, so we can see new content that might’ve changed from the user’s actions!

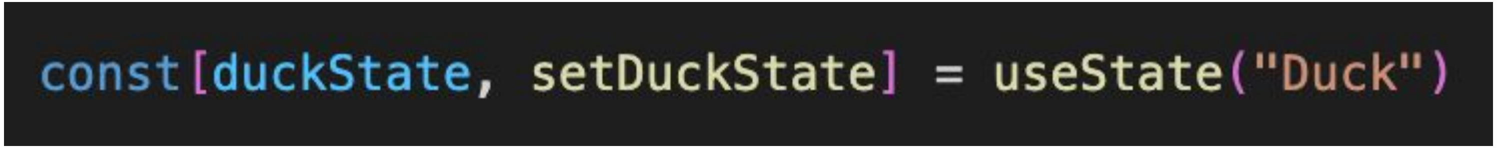
useState()

The initial value we set. This can be a string, integer, and so on. It's just whatever value you want to start with.

A function we'll call later in the body of our code to change the current state with.

The current state of whatever the element is.

?????????



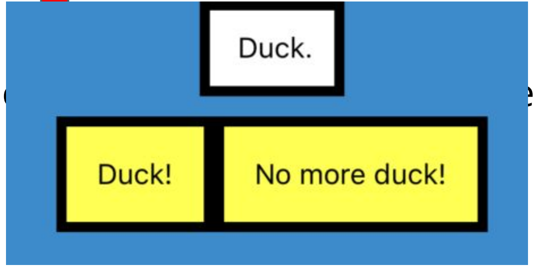
```
const[duckState, setDuckState] = useState("Duck")
```

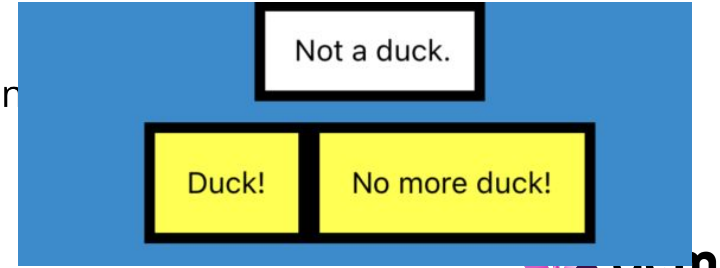
Ex: *user clicks “Duck!” button.*

User clicks “No more duck!” button.

De

e (in this case, an





How to use that current state value

So, we’ve defined our current state using the useState() function. How do we put it into our code?

- With curly brackets. { }
- Treat your current state like a variable, and just sandwich in between those two bad boys

```
<Text>{duckState}</Text>
```

- This should change the text to whatever the current state “duckState” is.

DEMO

Time to make some ducks

WOOOO we made our first app!

There are still tons of things beyond this workshop that we didn't cover. Even the stuff in this workshop, you could spend more than an hour on each!

- Organization
- Navigation
- Component documentation
- Props
- Etc.

Quick disclaimer for debugging

By now, whether you’ve been coding along or simply vibing to the information, it’s noteworthy to mention that bugs are common. YOU WILL MAKE ERRORS! (Unless you’re a coding god)

Common sources of headache (CHECK THESE!)

- Syntax (got all your {}? What about []? Maybe even a ()? Or a ,? Any typos?)
- Has everything been imported correctly?
- Only returning one thing?
- Google is your friend

Happy Hacking!