Speak React Native

2

React Native course by U+_

Overview

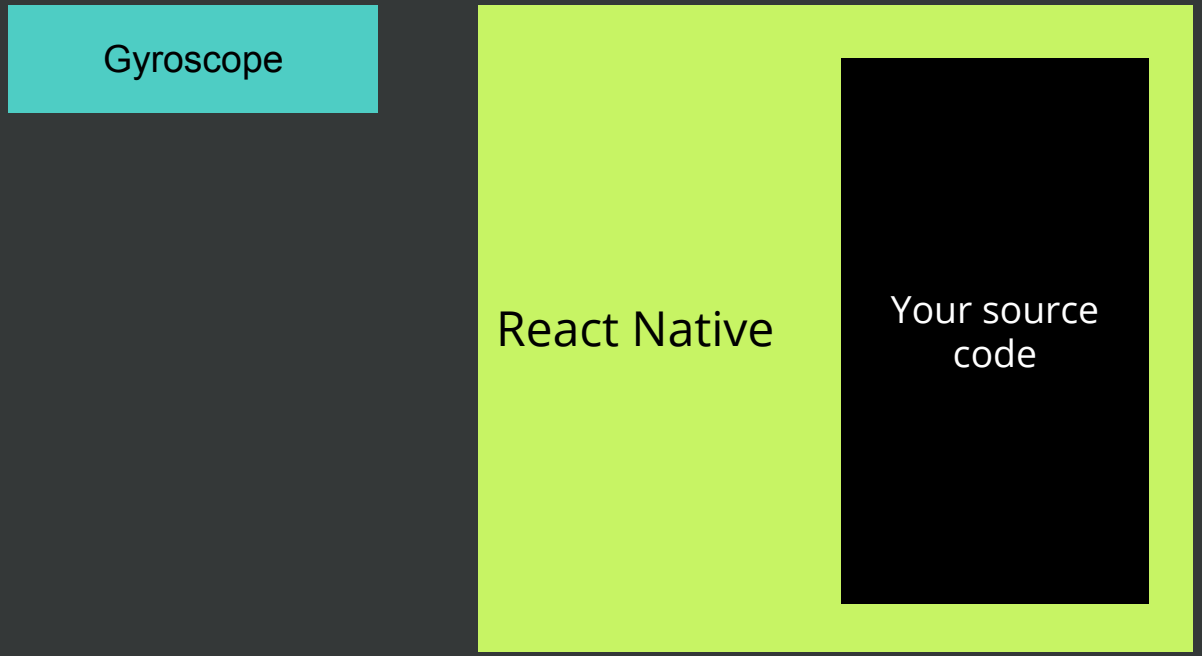
- Why are we using Expo?
- Javascript data types
- Git basics
- Basic React Native components + props
- Displaying datasets
- Implementation

Why are we using Expo?

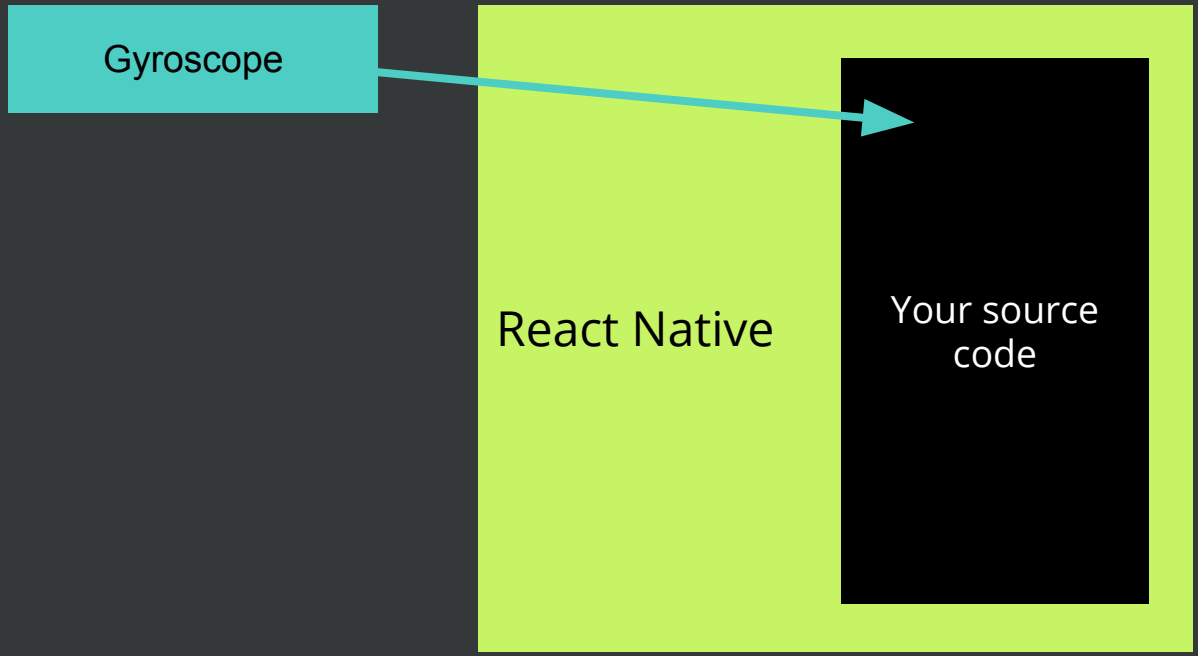
So, you want add some plugins...

(Development without Expo)

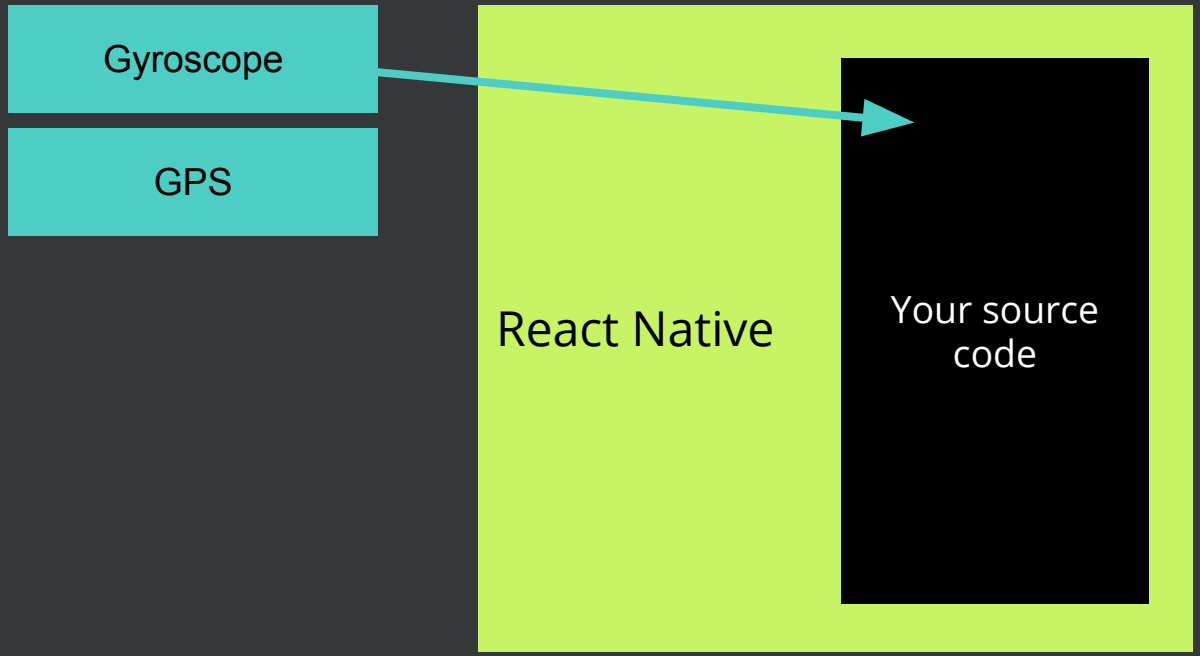
Install gyroscope plugin



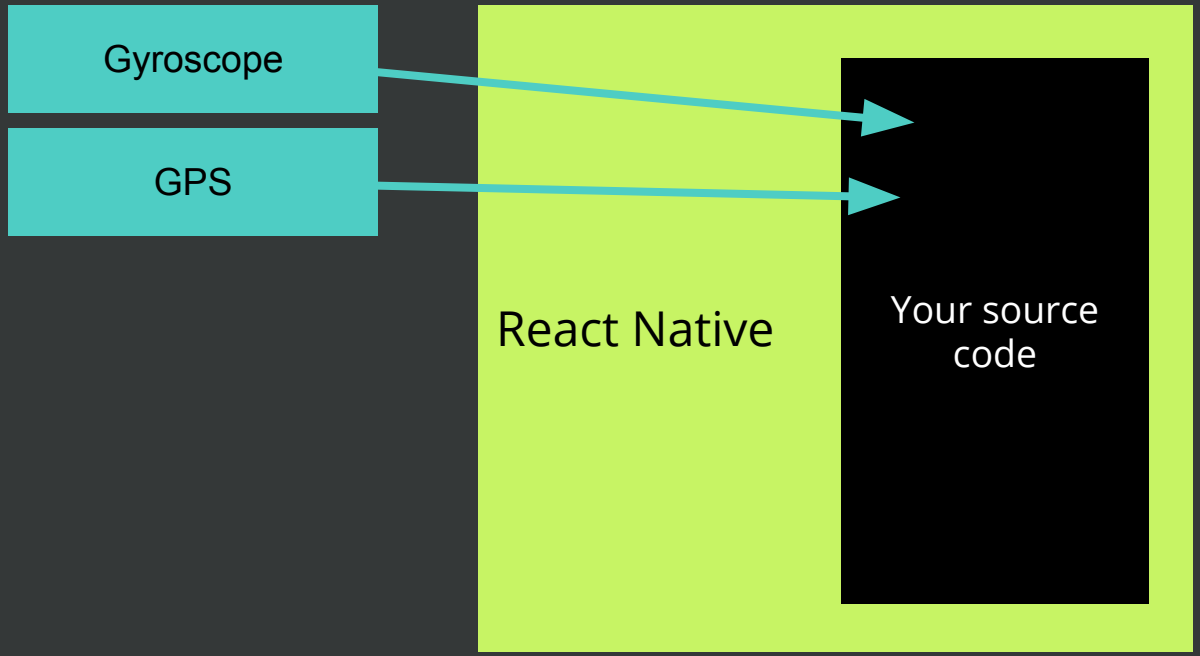
Link gyroscope to your app



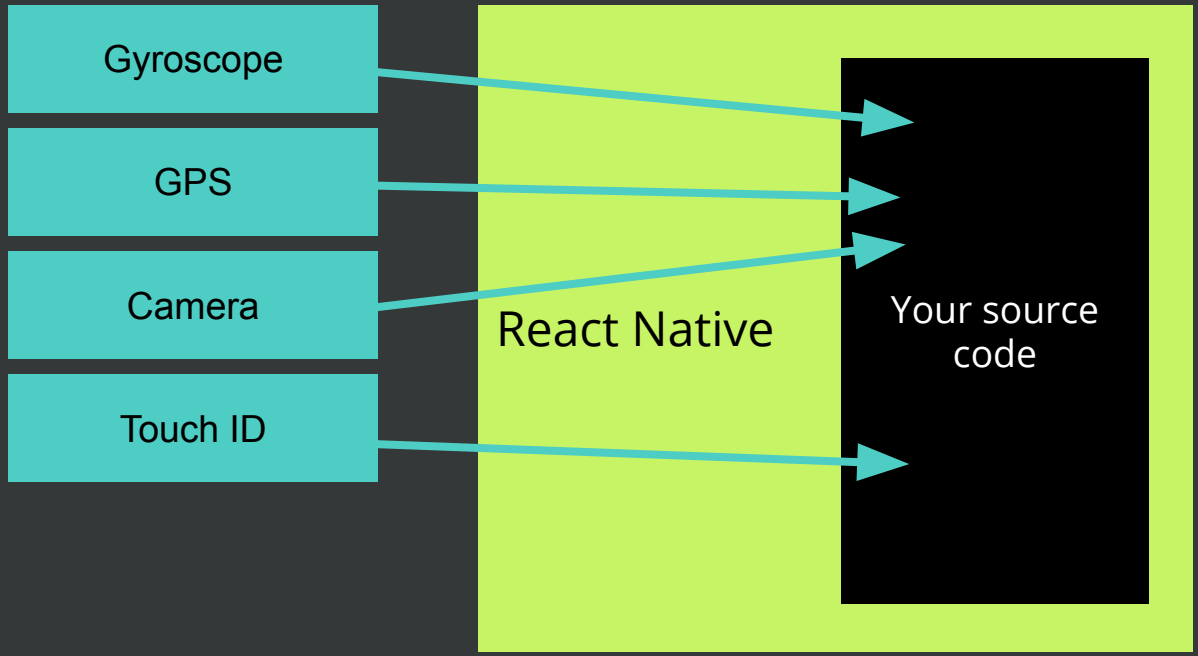
Install GPS plugin



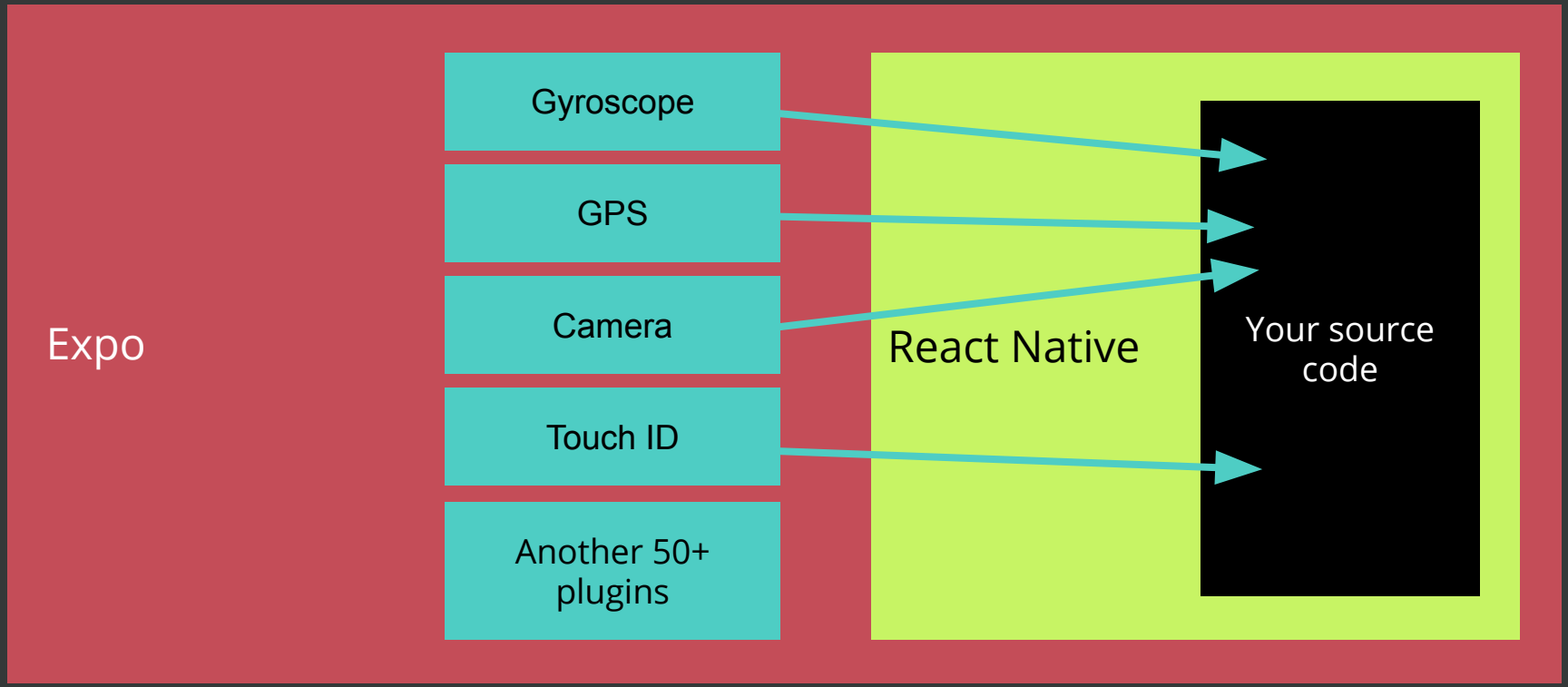
Link GPS to your app



And install & link more plugins...



With Expo it’s installed and linked automatically



Other reasons for Expo

- Verified set of native plugins
- Avoid working with Xcode or manual linking
- Easy to deploy and share your app (scan QR and run)
- Easier React Native updates

Javascript data types

Javascript data types

- Number `123`
- String `"Hello"`
- Boolean `true`
- Object `{}`
- Array `[]`

Object

- Key-value data structure
- Keys: strings, numbers
- Values: any type, can also be another object

```
const object = {
  key: "value",
}
```

Object

- Key-value data structure
- Keys: strings, numbers
- Values: any type, can also be another object

```
const beer = {
  name: "Pilsner Urquell",
  country: "Czechia",
  inProduction: true,
  alcohol: 4.4,
  brewery: {
    name: "Pilsen brewery",
    location: "Pilsen"
  }
}
```

Array

- “List” with some items
- Defined without keys, but uses number indexes
- Special Object type
- First item of array has **index 0**

```
const beers = [
  "Svijany",
  "Pilsner Urquell",
  "Hendrych",
]
```

Array

```
const beers = [
  "Svijany",
  "Bernard",
  "Hendrych",
]
```

Array

```
const beers = [
  "Svijany",
  "Bernard",
  "Hendrych",
]
```

How to use it

```
beers[1] // returns second (!) item:
"Bernard"
beers.length // returns array length: 3
beers[beers.length - 1] // returns last
item: "Hendrych"
```

Array

```
const beers = [
  "Svijany",
  "Bernard",
  "Hendrych",
]
```

Object Array

```
const beers = [
  {
    name: "Svijanský máz 11 %",
    alcohol: 4.8,
  },
  {
    name: "Pilsner Urquell",
    alcohol: 4.4,
  }
]
```

Object Array

```
const beers = [
  {
    name: "Svijanský máz 11 %",
    alcohol: 4.8,
  },
  {
    name: "Pilsner Urquell",
    alcohol: 4.4,
  }
]
```

Object Array

```
const beers = [
  {
    name: "Svijanský máz 11 %",
    alcohol: 4.8,
  },
  {
    name: "Pilsner Urquell",
    alcohol: 4.4,
  }
]
```

How to use it

```
beers[0] // returns first item: {name...}
beers[0].alcohol // returns 4.8
```

Git basics

Git basics

- System for version control (not only for code!)
- Fork, Sourcetree – graphic UIs for Git
  - Good for beginners, not good for real understanding of Git
- Recommended ebook (free): Pro git

Git basics (local repository)

```
$ git init                  # init new repository
$ git add <path>            # stage file(s) for commit
$ git status                # show changed files
$ git diff                  # show changed lines
$ git commit                # commit staged changes
```

Git basics (remote repository)

```
# add remote repository to our git
$ git remote add origin [repository-url]

# “get latest version”
$ git pull origin master

# “send last version”
$ git push origin master
```

Clone repository

```
$ git clone https://github.com/[username]/[reponame]

$ git clone https://github.com/jvaclavik/speak-react-native-skeleton
```

Git

- Tool
- Version control system, independent on GitHub

GitHub

- Service
- Remote repository hosting service which uses Git for version control

Fork repository

- Get your own copy of the repository

Fork repository

- Get your own copy of the repository

Basic React Native components

Basic React Native components

- View, ScrollView
- Text
- Image
- Touchables
- Stylesheet

View

- Basic container
- Imagine as `<div></div>`
- Overflowing content can be visible outside of view or hidden

ScrollView

- Scrollable container
- Overflowing content can be scrolled
- Supports scrolling events & more

Text

- May contain string or another nested Text component
- Should not contain any other components

```
<Text>
  Hello.
</Text>
```

Text

- May contain string or another nested Text component
- Should not contain any other components

```
<Text>
  Hello.
  <Text
    style={{
      fontWeight: "bold",
    }}
  >
    And a nested bold text.
  </Text>
</Text>
```

Image

- Does not contain any children
- Supports both local (using require) and remote images (uri)

```
// local image
<Image source={require("/path/to/local/img")} />

// OR remote
// note the double braces - source is an object
<Image source={{ uri: "http://remote.img" }} />
```

Touchables

- Something like button
  - Components for handling user interactions (taps)
- More types, same usage, differ by feedback given

Touchables

```
const doSomething = () => {
  alert("(Im)pressed")
}
...
<TouchableHighlight onPress={doSomething}>
  <Text>Do something</Text>
</TouchableHighlight>

// The same goes for TouchableOpacity, TouchableWithoutFeedback,...
```

Touchables (pass argument)

```
const doSomething = (something) => {
  alert(`(Im)pressed by ${something}!`)
}
...
<TouchableHighlight onPress={() => doSomething("React Native")}>
  <Text>Do something</Text>
</TouchableHighlight>

// The same goes for TouchableOpacity, TouchableWithoutFeedback,...
```

Styling

- Very similar to CSS
- Use StyleSheet class for better performance

```
import { StyleSheet } from "react-native"

const styles = StyleSheet.create({
 button: {
   backgroundColor: "blue",
 },
})

// Use like this
<TouchableOpacity style={styles.button}>
  <Text>Button</Text>
</TouchableOpacity>
```

Component props

Props

- Parameters (inputs) of the component
- Accessible through `this.props` or function parameters

```
<Component
  title="Title"
  count={123}
/>

// Access inside the component
<View>
  <Text>{this.props.title}</Text>
  <Text>{this.props.count}</Text>
</View>
```

Component Children prop

- Anything “inside” the component - other components, text,...
- Accessible through special `children` prop

```
<Component>
  <Text>Child</Text>
</Component>

// Access inside the component
<View>
  {this.props.children}
</View>
```

Render function

- Must return single component

```
// Good
render () {
  return (
    <SomeComponent>
      <Text>React</Text>
      <Text>Native</Text>
    </SomeComponent>
  )
}
```

```
// Bad
render () {
  return (
    <Text>React</Text>
    <Text>Native</Text>
  )
}
```

Pass objects to props

- {} indicates some Javascript inside
- {{}} indicates some object inside of Javascript

```
<BeerList
 data={["Svijany", "Bernard", "Hendrych"]}
 config={{
   showNonAlcoholic: true,
   maxLength: 10,
 }}
/>
```

Pass objects to props

- {} indicates some Javascript inside
- {{}} indicates some object inside of Javascript

```
<BeerList
 data={["Svijany", "Bernard", "Hendrych"]}
 config={{
   showNonAlcoholic: true,
   maxLength: 10,
 }}
/>
```

Implementation

Implementation

- Create a button using `TouchableOpacity` which displays alert when pressed
- Do some basic styling for the button

Implementation

- Use a `ScrollView` and map function to display a list of items
- Move `View` for a single item to separate component

Homework

- Create an account at http://github.com
- Create new repository `srn-homework1`
- Fork `https://github.com/jvaclavik/speak-react-native-skeleton`
- Create components for
  - Title
  - Image
  - Button
- Commit changes and push to your repository

Questions?

?