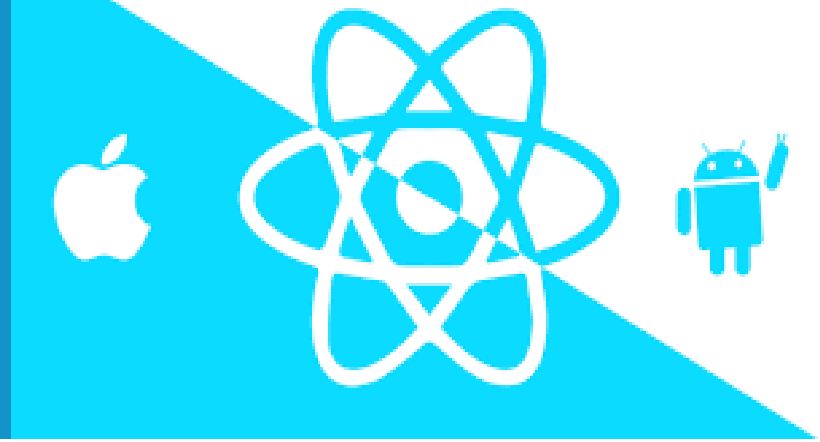
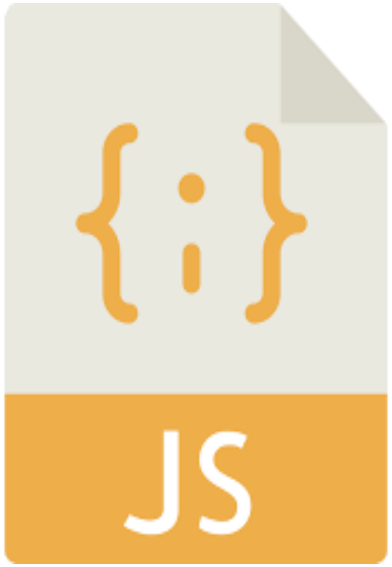
React Native

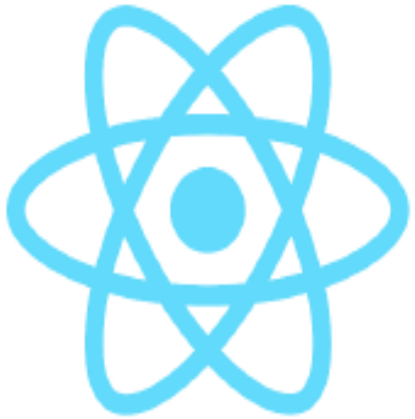


Karthik Raman

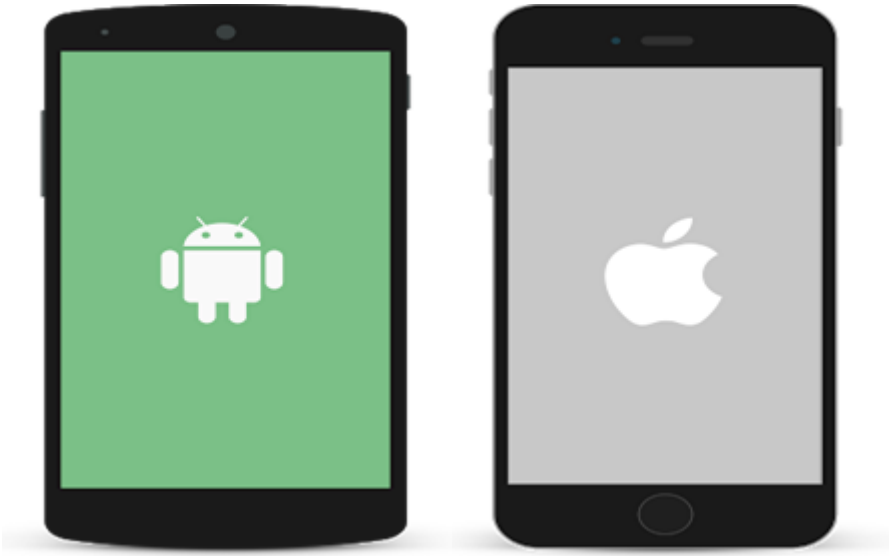
What is React Native?



JavaScript



React



Android iOS

Builds Native Apps

Builds Real Native Apps Cross-platform

iOS

Android

Real Native App is indistinguishable from an App built with Java/Kotlin for Android and Objective-C and Swift for iOS

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';

class WhyReactNativeIsSoGreat extends Component {
  render() {
    return (
      <View>
        <Text>
          React Native uses React library to build mobile apps.
        </Text>
        <Text>
          You just use native components like 'View' and 'Text',
          instead of web components like 'div' and 'span'.
        </Text>
      </View>
    );
  }
```

History

React is a JavaScript library for the web was open-sourced by Facebook in 2013

React Native a mobile solution for building native apps using React, and was open-sourced by Facebook in 2015

React Native is embraced by many large Fortune 500 companies and small companies in their mobile space

https://reactnative.dev/

https://github.com/facebook/react-native

Who Is Using React Native?

Facebook

Instagram

Uber Eats

Walmart Labs

Pinterest

Bloomberg Financial

Native, Hybrid and Native Hybrid Comparison

Approach	Platform	Logic	UI	UX
Native	Object C, Swift (iOS), JAVA,Kotlin (Android)	Native	Native	Smooth
HTML5 Hybrid	PhoneGap, Cordova, Ionic	JS	HTML / CSS3	Laggy
Native Hybrid	React Native, NativeScript	JS	Native	Smooth

Advantage of both the ecosystems (Native and Hybrid) has been adopted in React Native.

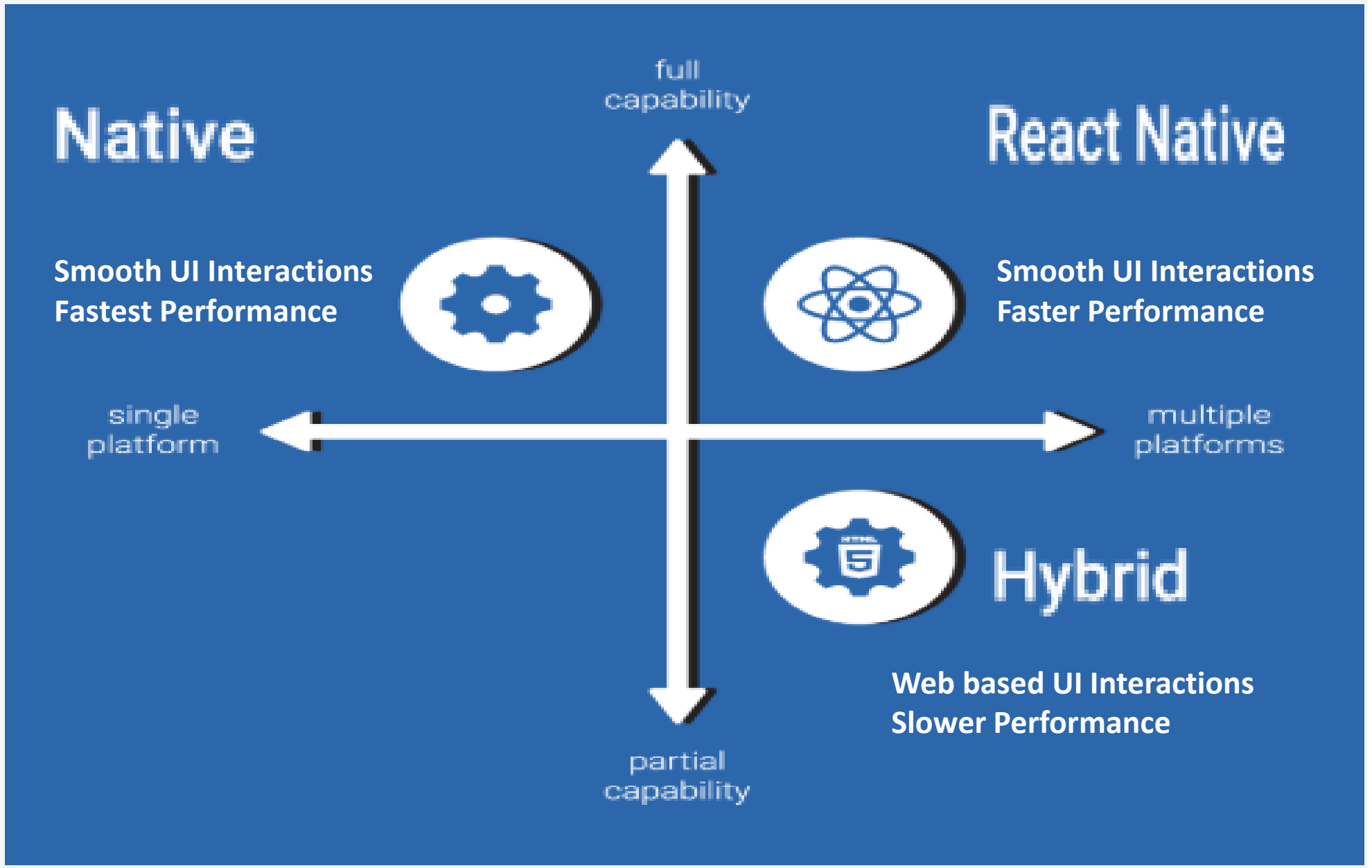
Specific comparison between React Native and Cordova (current mobile app platform) is given in the following slide.

React Native v/s Cordova Comparison

	React Native	Cordova
Philosophy	Learn once, write anywhere	Write once, run anywhere
Multi-Platform	Android and iOS	Andoid and iOS
Look and Feel	Native Look and Feel with Native Controls	Browser DOM Look and feel and Web Controls
Tech Stack	React and JSX with React Native packages	HTML5 CSS3 Javascript and Plugins
Styling	React Native FlexBox Styling	CSS3 Stying
Freedom of Framework Usage	Works only with React	Works on any Javascript framework
Behaviour	App can be configured to look and behave same or differently on different device	App looks standard on different devices
Performance	Faster as it works with multiple threads and UI rendering is faster. Runtime Performance is faster	Laggy as all the application code works as a single thread
Bundle Size	Relatively larger as it includes many native controls as built-in features	Small as it packages only the necessary plugins used by the application. Build time performance is faster
Debugging	Easier - Allows Hot and Live Reloading and Introspect the states in the application	Similar Debugging tools needs to setup manually
Extensibility	Native controls are built-in, and the third-party packages are evolving as the RN community is growing	Readily available third-party plugins and Custom plugins can be written easily

React Native v/s Flutter Comparison

	React Native	Flutter
Created by	Facebook	Google
Programming Language	Javascript (most popular language used worldwide)	Dart (created by Google which is less popular)
Framework	Flux and Native Bridge	Skia
Architecture	Only UI Elements in Native Code	Compiled into ARM C/C++ libraries
Framework Installation	Simple	Slightly complex
Learning Curve	Is simple as it uses Javascipt	Longer as it uses Dart
Developer Productivity	Greater support for most popular IDEs	Lack of support but slowly growing
Built-In Components	Ships with basic built-in components and usage of third party libs for advanced components	Ships with more built-in Widgets
Community Support	Good third party ecosystem	Slowly growing
Popularity	Most popular	Slowly growing
Performance	Better Performance (Java thread and UI thread) and uses Native bridge	Even better performance as it is compiled code
Testing Support	Tools such as Jest	In-built Widget testing features
Real World Usage	Facebook,Instagram,Skype,Airbnb,Walmart,Pinterest ,Tesla	Google Ads app,Alibaba,eBay,Groupon
Skill Availability	Market is flooded with React Native skillset	Resources are Scarce but slowly growing in numbers



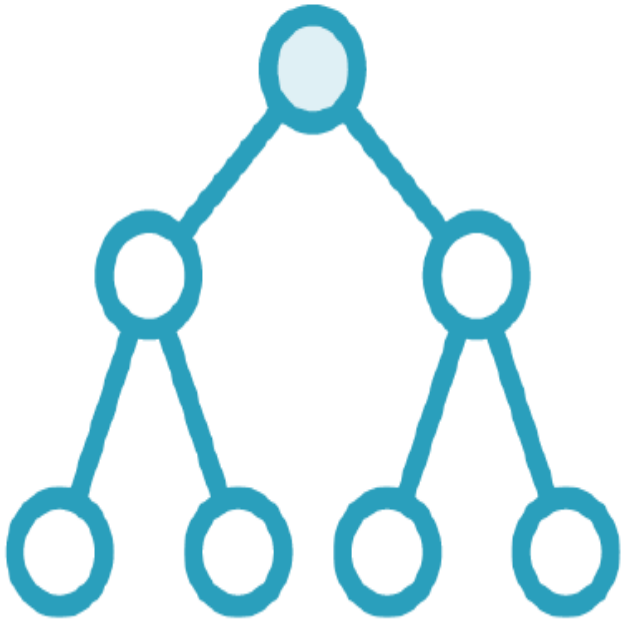
uses React Framework --- Learn Once Write Anywhere

Component-based architecture

Reuse of components

One directional data flow

Everything Is a Component



- Components
- JSX
- Props & States
- UniDirectional Flow
- Virtual DOM
  - Shadow Tree
- Hooks
- HOC
- Axios
- Redux

True Native App from Javascript

Code Sharing 

Faster Development and Testing

Hot Reloading

React Native Debugger
- React Inspector
- Redux DevTools

Performance on par with Native App

Over the Air (OTA) Update

Excellent Community

https://reactnative.dev/

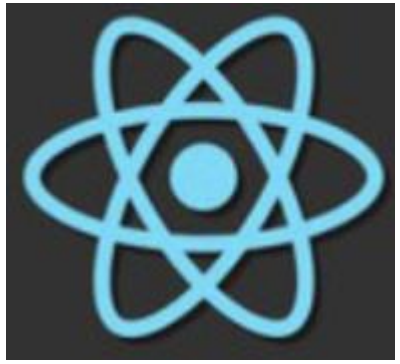
Able to Bridge to Native Code when needed

Expo

Cloud Build

React Native BoilerPlate

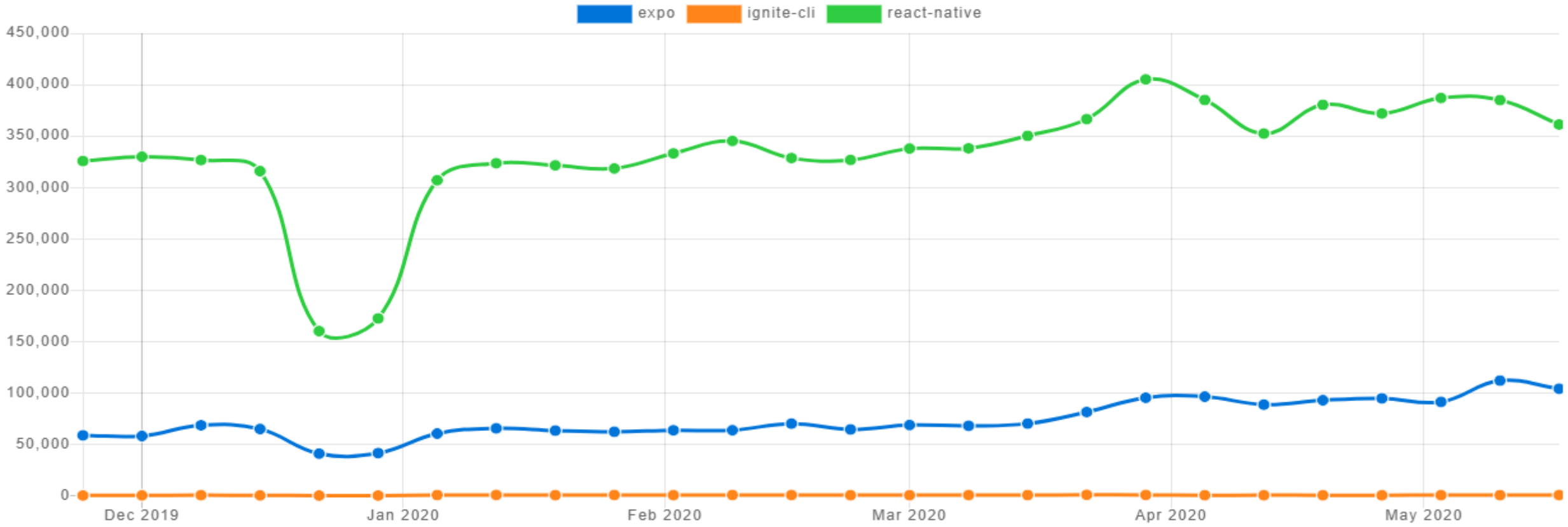
React Native CLI

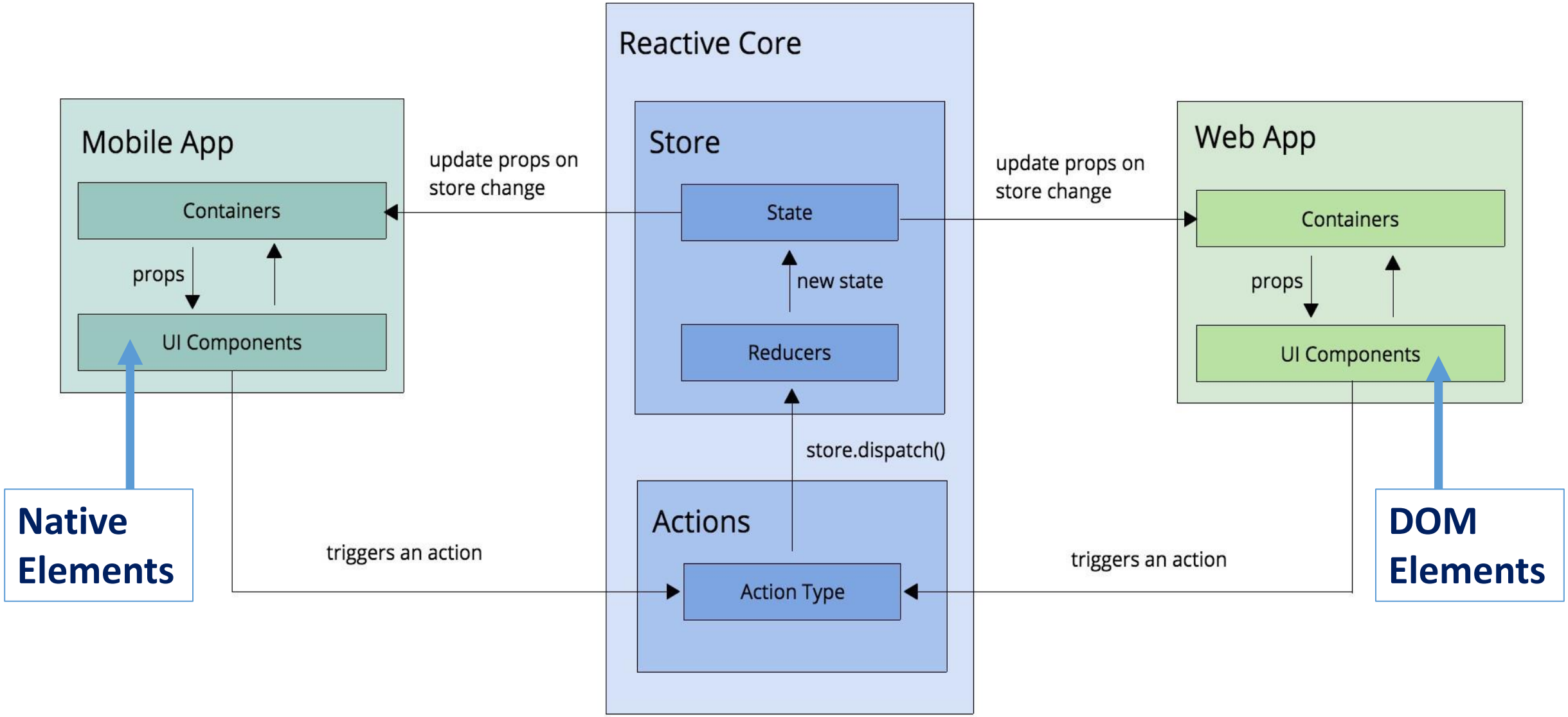


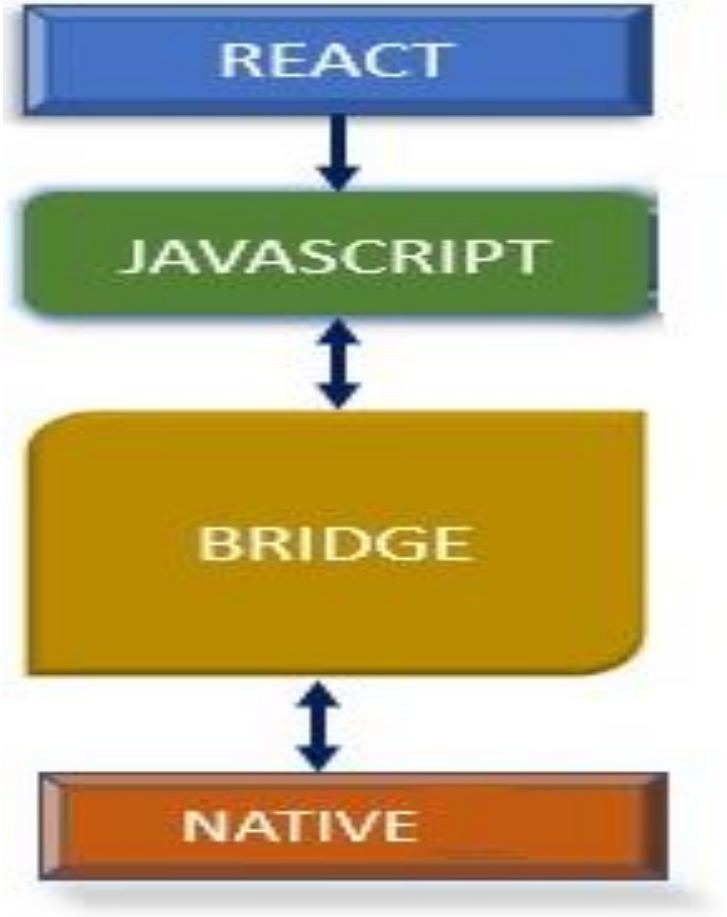
Local Build

expo × ignite-cli × react-native ×

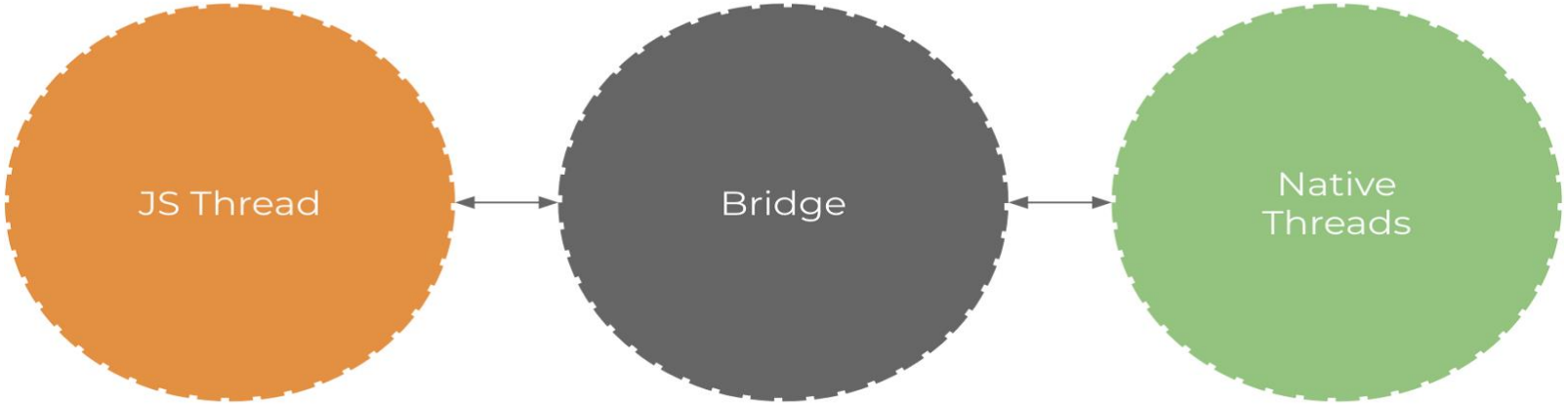
Downloads in past 6 Months







React Native runs on a separate thread from the Main UI rendering thread using JavaScript Asynchronous call via the bridge to invoke the host native platform underlying API and UI Elements



Advantages

- Setting up a project is easy and can be done very quickly
- Can build for both Android and iOS without having Mac Machine
- No need Android or iOS simulator for running the app
- Need Expo Account to publish the App to a permanent URL or to build for Android PlayStore (apk) and Apple AppStore (ipa)

DisAdvantages

- SDK and Expo APIs
- Cannot add Native Modules
- Cloud based build – Security issue
- File Size

https://docs.expo.io/introduction/why-not-expo/?redirected

- **Install NodeJS (https://nodejs.org/)**

- **install expo-cli globally with node package manager (npm)**

```
npm install -g expo-cli
```

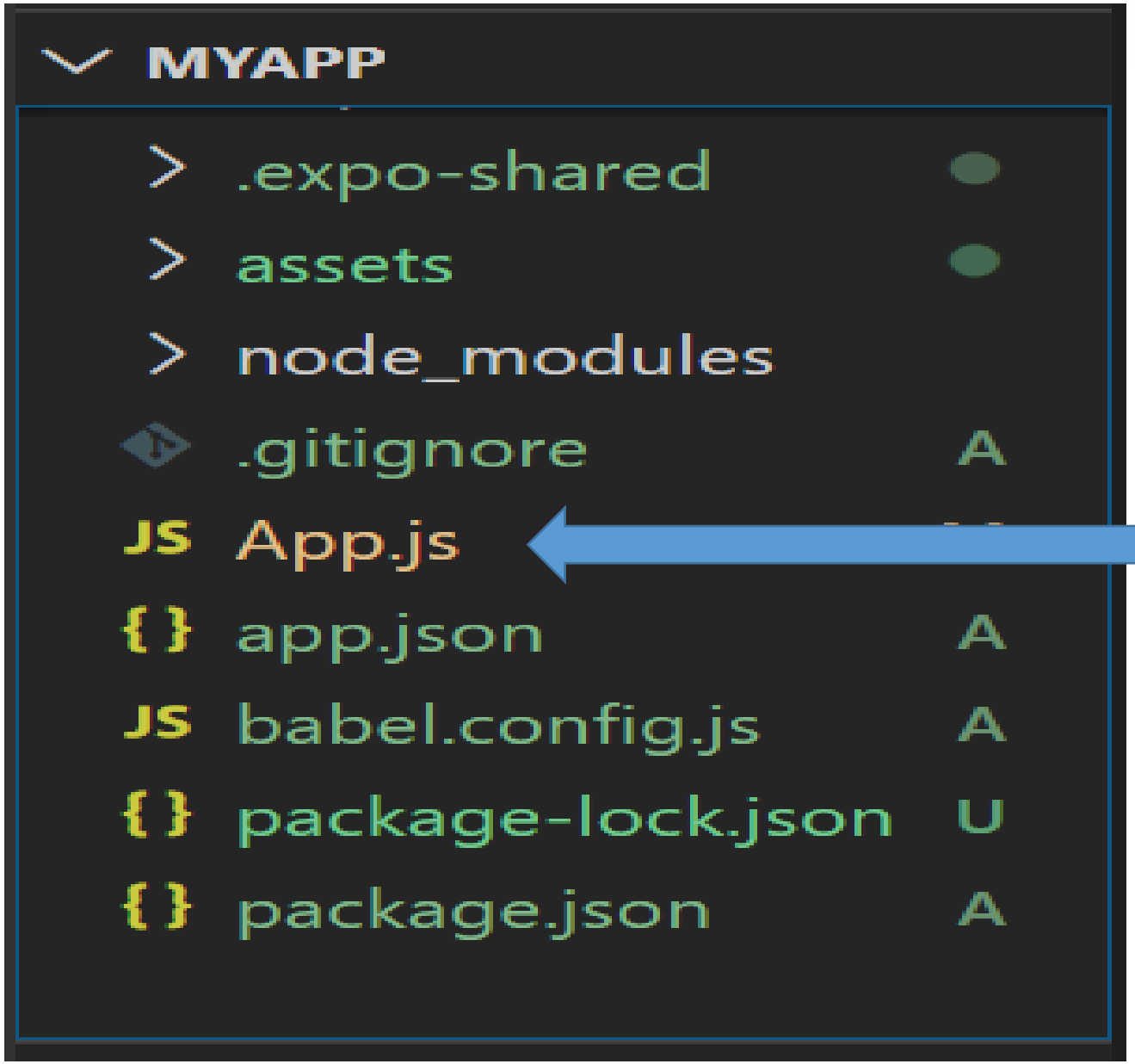
- **Run expo init in your chosen directory**

```
expo init myapp
```

- **Navigate to the newly created directory and run the start script**

```
cd my-app/
npm start
```

- **This will start the development server**



- Install the ExpoGo App on Android phone from PlayStore or iOS phone (AppStore)
- Connect to Phone the same wireless network as your computer
- On Android, use the Expo app to scan the QR code
- On iOS, use the built-in QR code scanner of the Camera app

Run on Android device/emulator

Run on iOS simulator

Run in web browser

Send link with email...

Publish or republish project...

PRODUCTION MODE

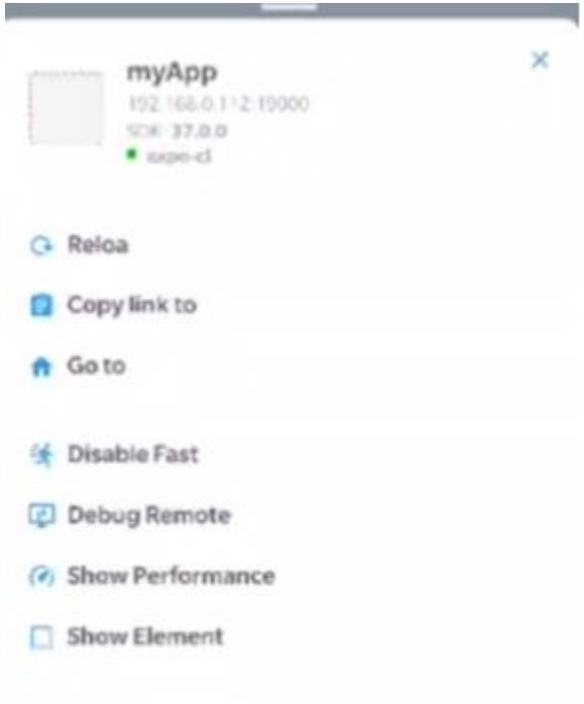
CONNECTION Tunnel LAN Local

exp://192.168.0.112:19000

QR Code in Browser

Scan QR Code using Expo Client

- **Shake the Device**



- **Inspect Element**

- **Remote Debug using React Developer Tools : react-devtools**

Android

Download and Install the following :

- Java Development Kit (JDK) https://www.java.com/en/download/
- Android Studio https://developer.android.com/studio/
- Open Android Studio and install SDK Components
- Tools -> AVD Manager -> Create Virtual Device

npm start

Press ‘a’ to run on Android Emulator

Press Ctrl M to get the developer option

Press R twice (Windows) or CMD R to reload the App

iOS

Download Xcode

npm start

Press ‘i’ to run on iOS Simulator

Android

Check the path of simulator installation (emulator.exe):

In windows , typically it may be at
C:\Users\{loggedIn Name}\AppData\Local\Android\Sdk\emulator\emulator.exe

Ex : C:\Users\{karth}\AppData\Local\Android\Sdk\emulator\emulator.exe

To List all the Android Virtual Devices

emulator -list-avds **Ex: output Pixel_3_API_29**

To Run the App on a Specific AVD

emulator -avd <avd_name>

emulator -avd Pixel_3_API_29

- **VS Code**

View -> Extensions
Install React Native Snippet - Jundat95

rnce{tab} - Snippet for Class based React Native Component with export

rnss{tab} – Snippet for Create StyleSheet

- **View**
- **Text**

HelloWorld.js

ViewStyleExample.js

ViewClickExample.js

React Native has the concept of style inheritance but limited to text subtrees.
inheritance pattern : styles.text will be applied to all Text components in the subtree

TextExample.js

- **Inline Style**

```
<Text style={{color:'red'}}>Red text</Text>
```

- **Styling using a Stylesheet**

```
const styles = StyleSheet.create({
red: {
color: 'red'
},
big: {
fontSize: 30
}
})

<View>
<Text style={{styles.red}}>Red</Text>
<Text style={[styles.red, styles.big]}>Big Red</Text>
</View>
```

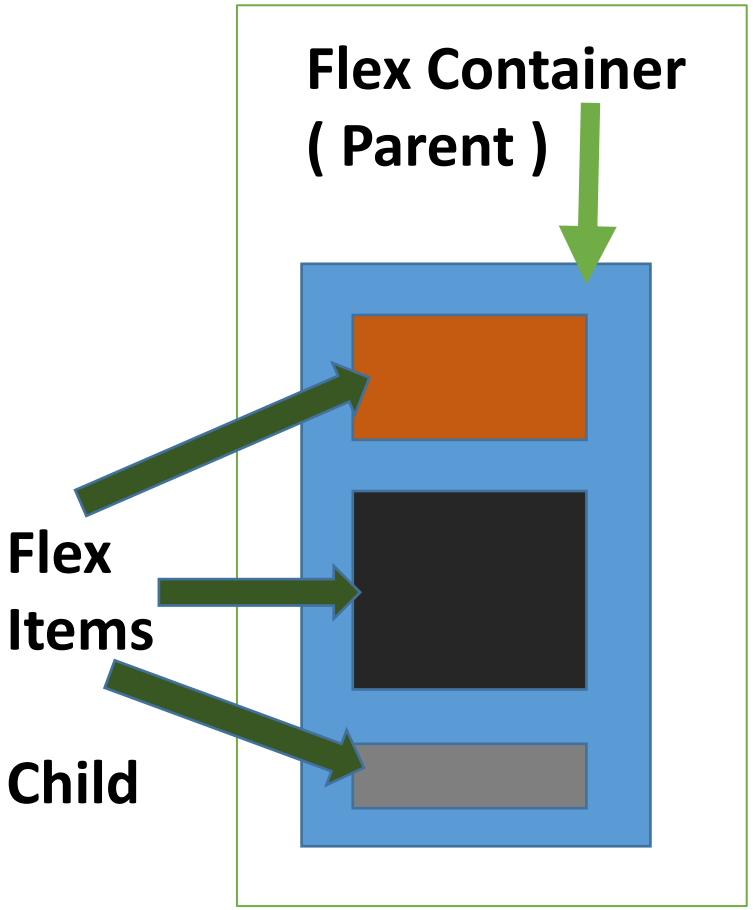
Adding multiple styles

If Conflict, the last one Wins

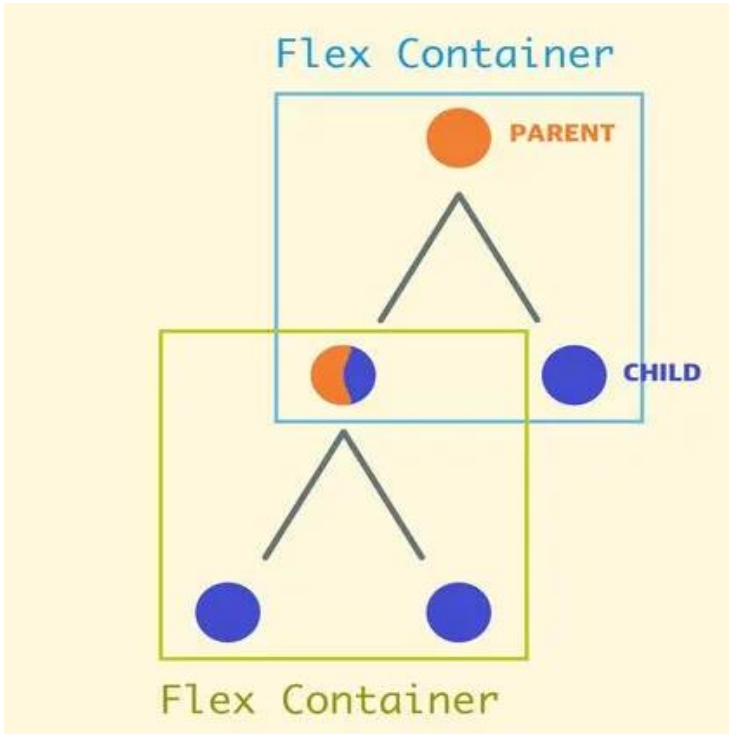
Conditional Styling

```
<View style={[(this.props.isTrue) ? styles.bgcolorBlack :
                                    styles.bgColorWhite]}>
```

- FlexBox for CSS on the Web (layout method for rendering items in rows and columns
- Dynamic Sizing and Vertical centering
- React Native works on the concept of FlexBox
- Designed to provide a consistent layout on different screen sizes



Flex affects the Immediate Child Level Only



FlexBox properties

DIRECTION	ALIGNMENT	SIZE
flexDirection	justifyContent	flex
flexWrap	alignItems	flexGrow
	alignContent	flexShrink
	alignSelf	flexBasis

Parent

Self

Fixed Dimensions

The general way to set the dimensions of a component is by adding a fixed width and height to style.
All dimensions in React Native are unitless, and represent density-independent pixels

FixedDimensions.js

Flex Dimensions

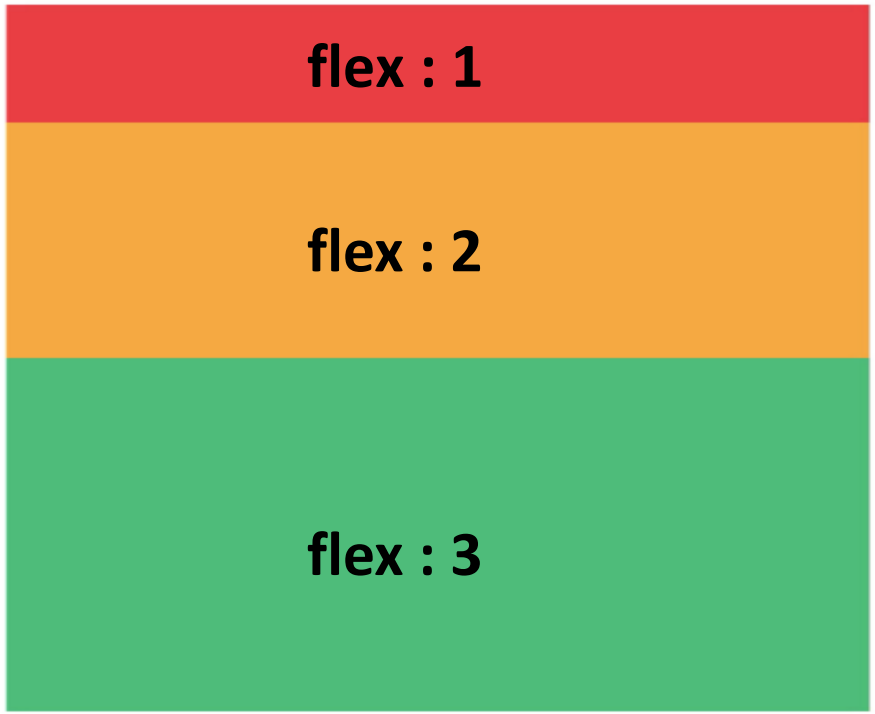
flex to expand and shrink dynamically based on available space
flex: 1, which tells a component to fill all available space

If a parent does not have either a fixed width and height or flex, the parent will have dimensions of 0 and the flex children will not be visible

flex

will define how items are going to “fight” over the available space along the main axis

Space will be divided according to each element flex property



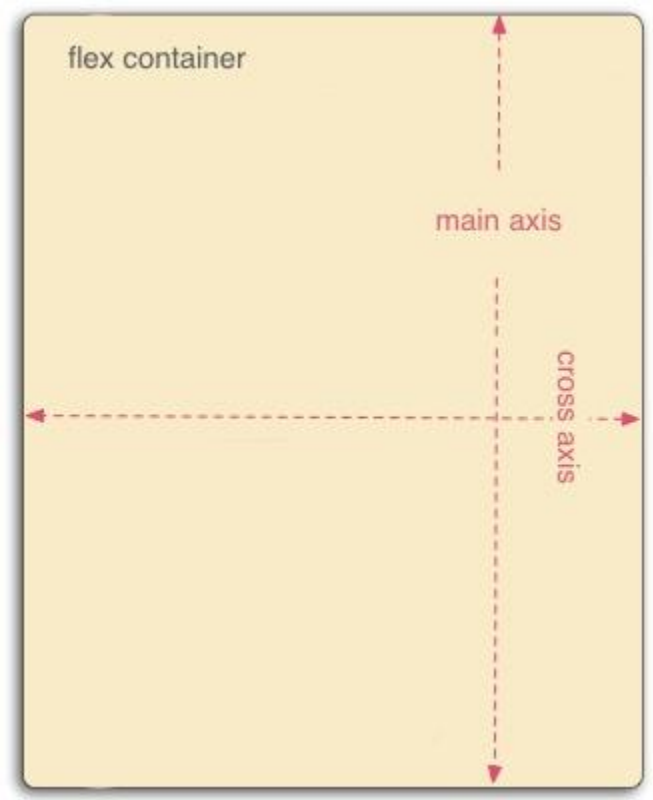
total flex = 1+2+3 = 6
red view will get 1/6 of the space
yellow 2/6 of the space
green 3/6 of the space

FlexTest.js

flexDirection

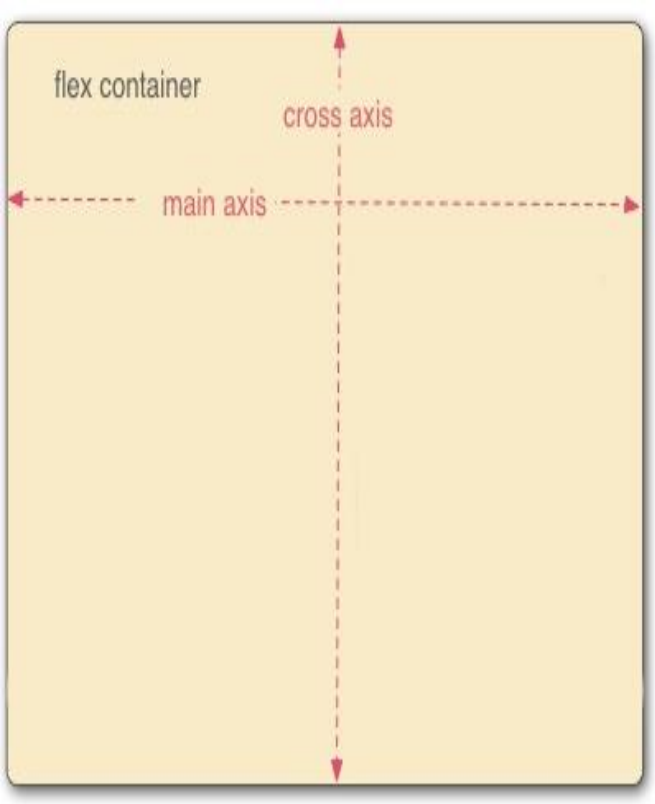
controls the direction in which the children of a node are laid out

flexDirection : column



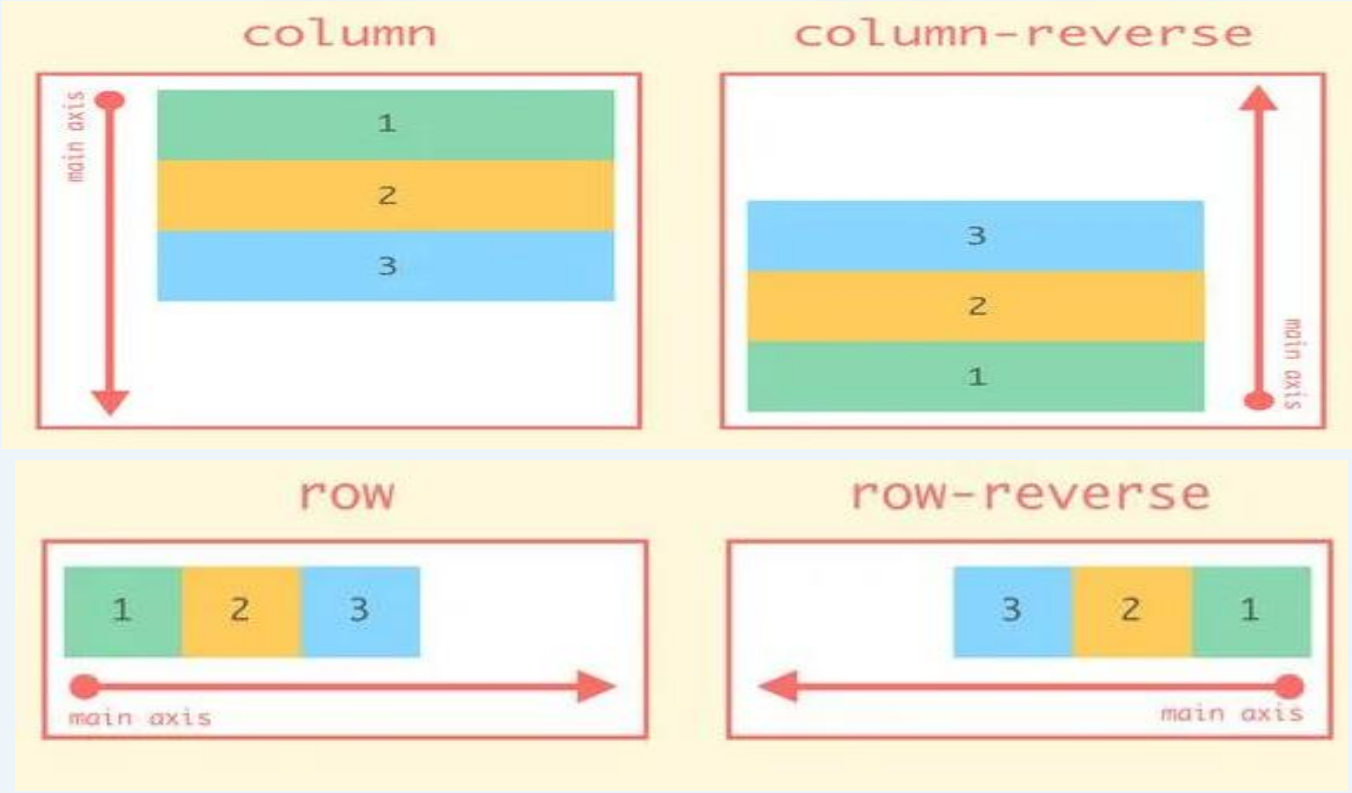
default

flexDirection : row



flexDirection

controls the direction in which the children of a node are laid out



Main Axis

Cross Axis

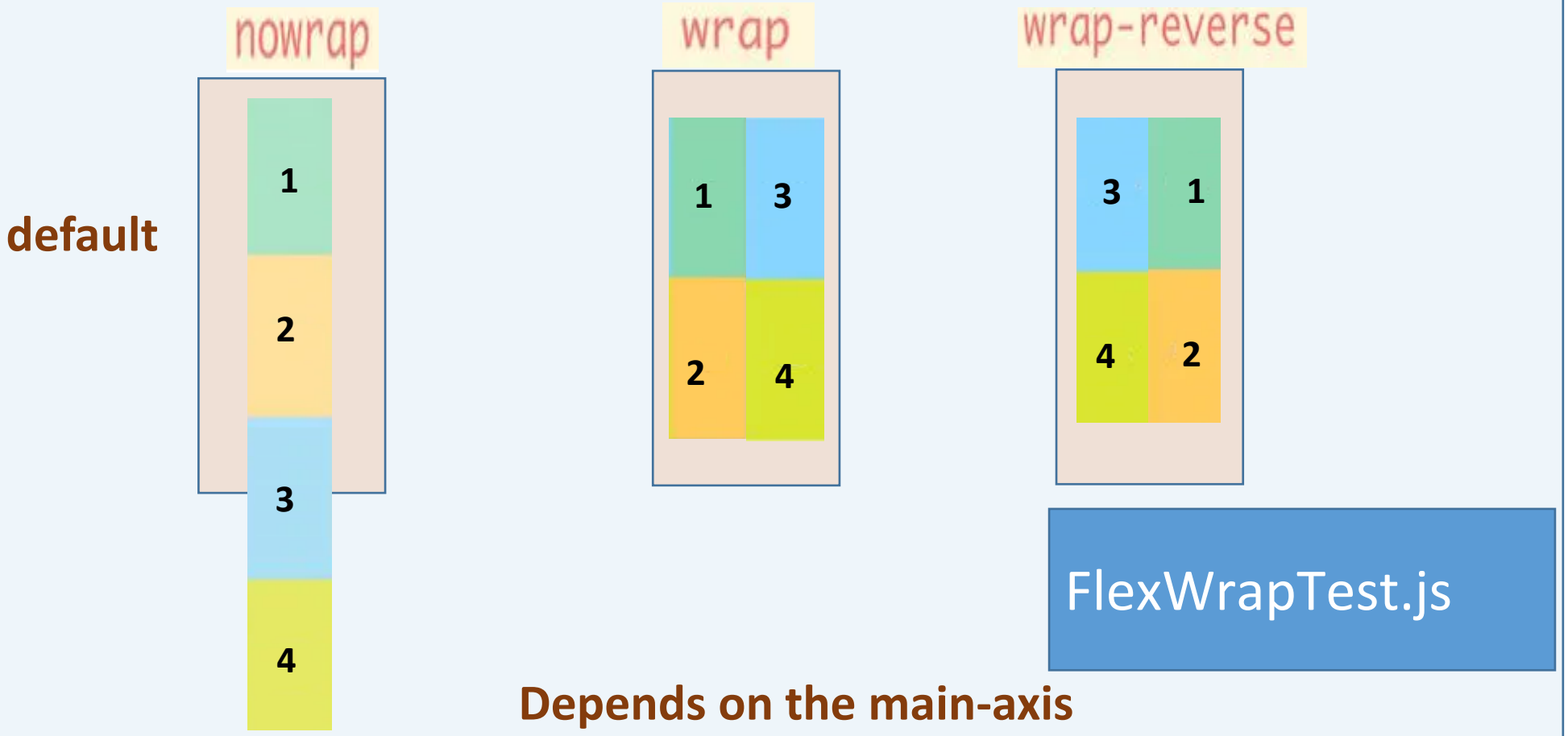
FlexDirectionTest.js

React Native – default direction is column

Web – default is row

flexWrap

controls what happens when children overflow the size of the container along the main axis

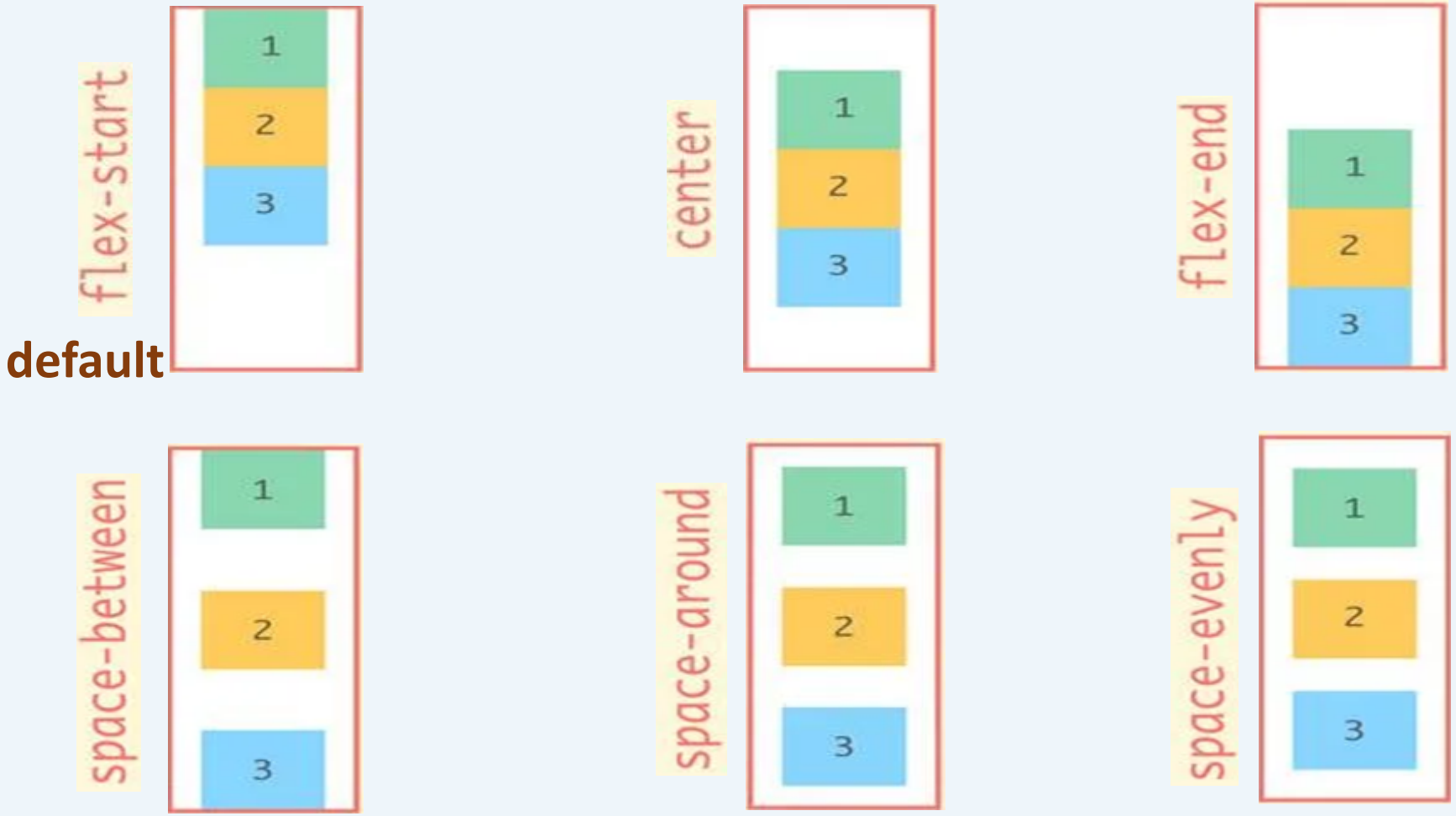


FlexWrapTest.js

Depends on the main-axis

justifyContent **flexDirection : column**

controls how to align children within the main axis



Sets alignment along the main-axis

justifyContent

flex-start(default value) : Align children of a container to the start of the container's main axis.

flex-end : Align children of a container to the end of the container's main axis.

center : Align children of a container in the center of the container's main axis.

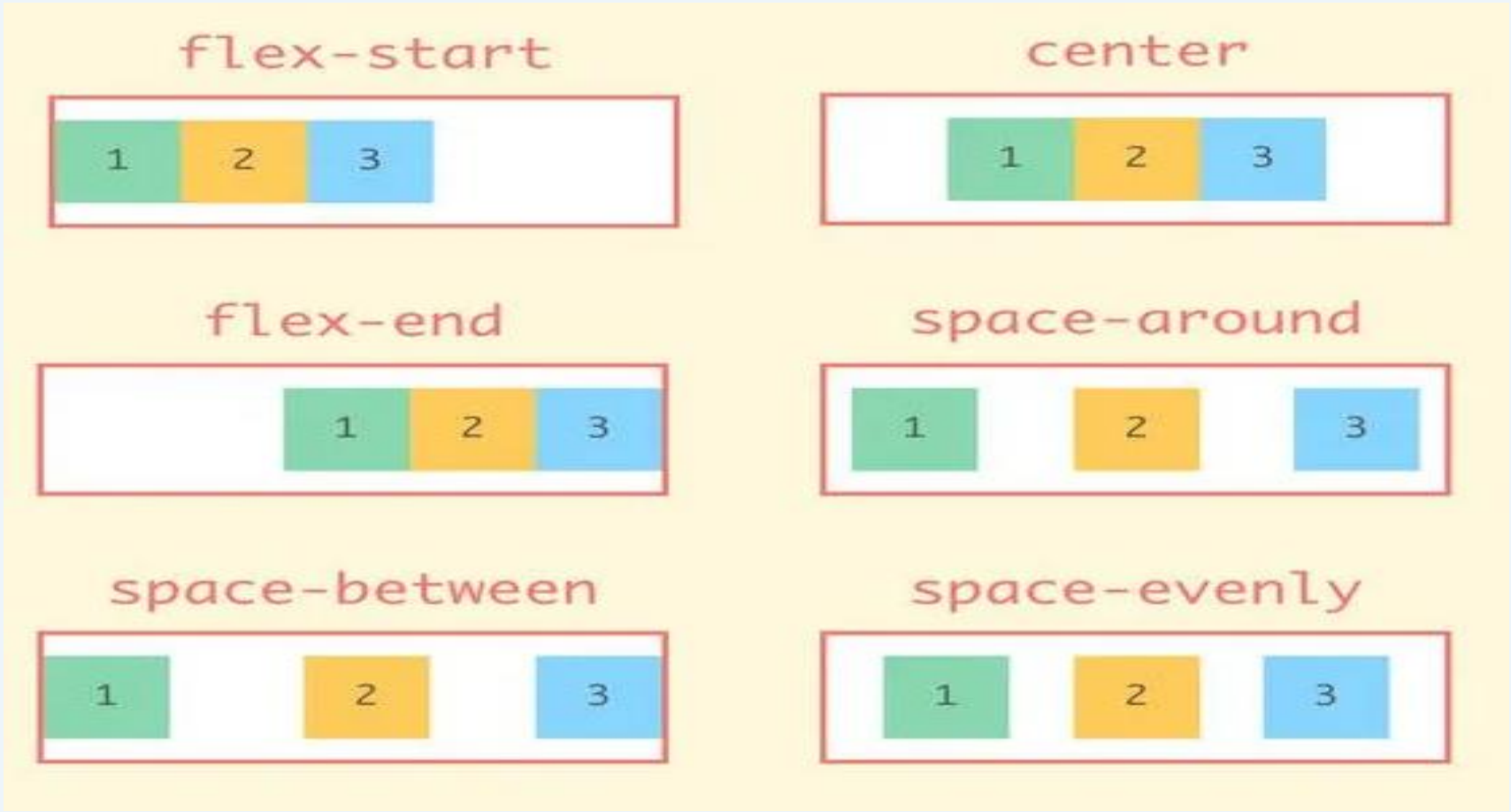
space-between : Evenly space off children across the container's main axis, distributing the remaining space between the children.

space-around : Evenly space off children across the container's main axis, distributing the remaining space around the children. Compared to space-between, using space-around will result in space being distributed to the beginning of the first child and end of the last child.

space-evenly : Evenly distribute children within the alignment container along the main axis. The spacing between each pair of adjacent items, the main-start edge and the first item, and the main-end edge and the last item, are all exactly the same.

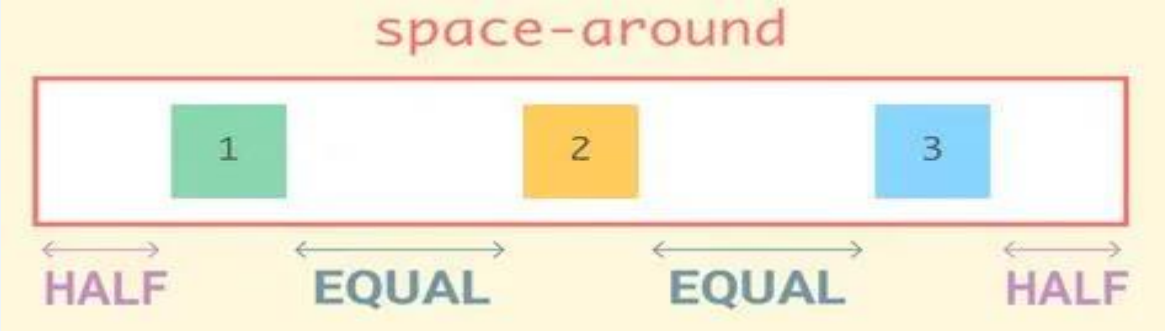
Sets alignment along the main-axis

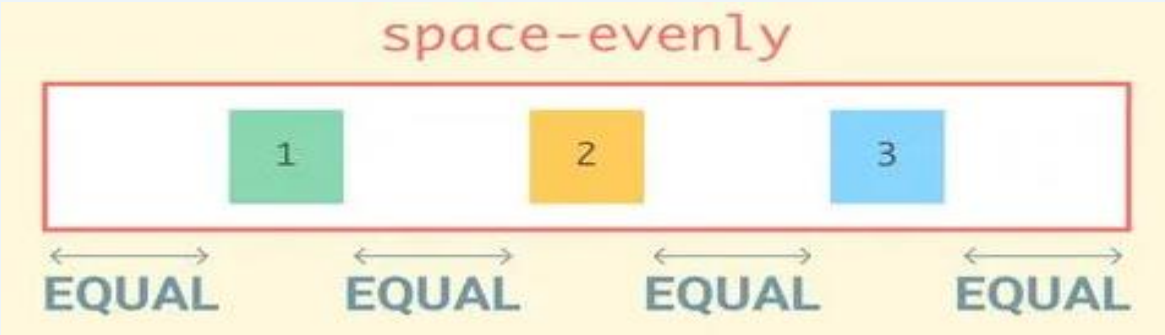
justifyContent **flexDirection : row**



Sets alignment along the main-axis

justifyContent **flexDirection : row**



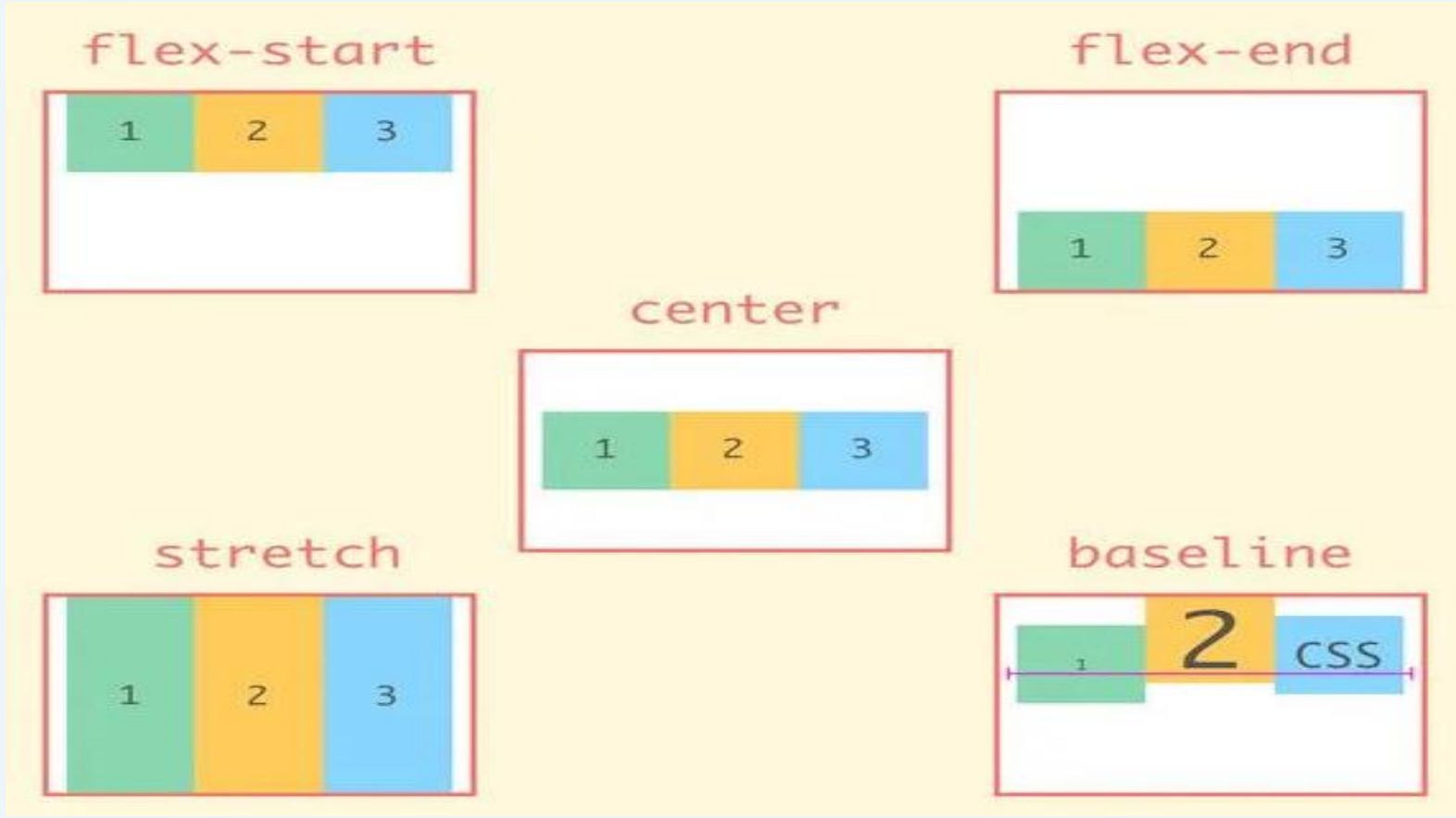


JustifyContentTest.js

Sets alignment along the main-axis

alignItems **flexDirection : row**

controls how to align children along the cross axis



default

Sets alignment along the cross-axis

alignItems

stretch (default value) : Stretch children of a container to match the height of the container's cross axis

flex-start : Align children of a container to the start of the container's cross axis

flex-end : Align children of a container to the end of the container's cross axis

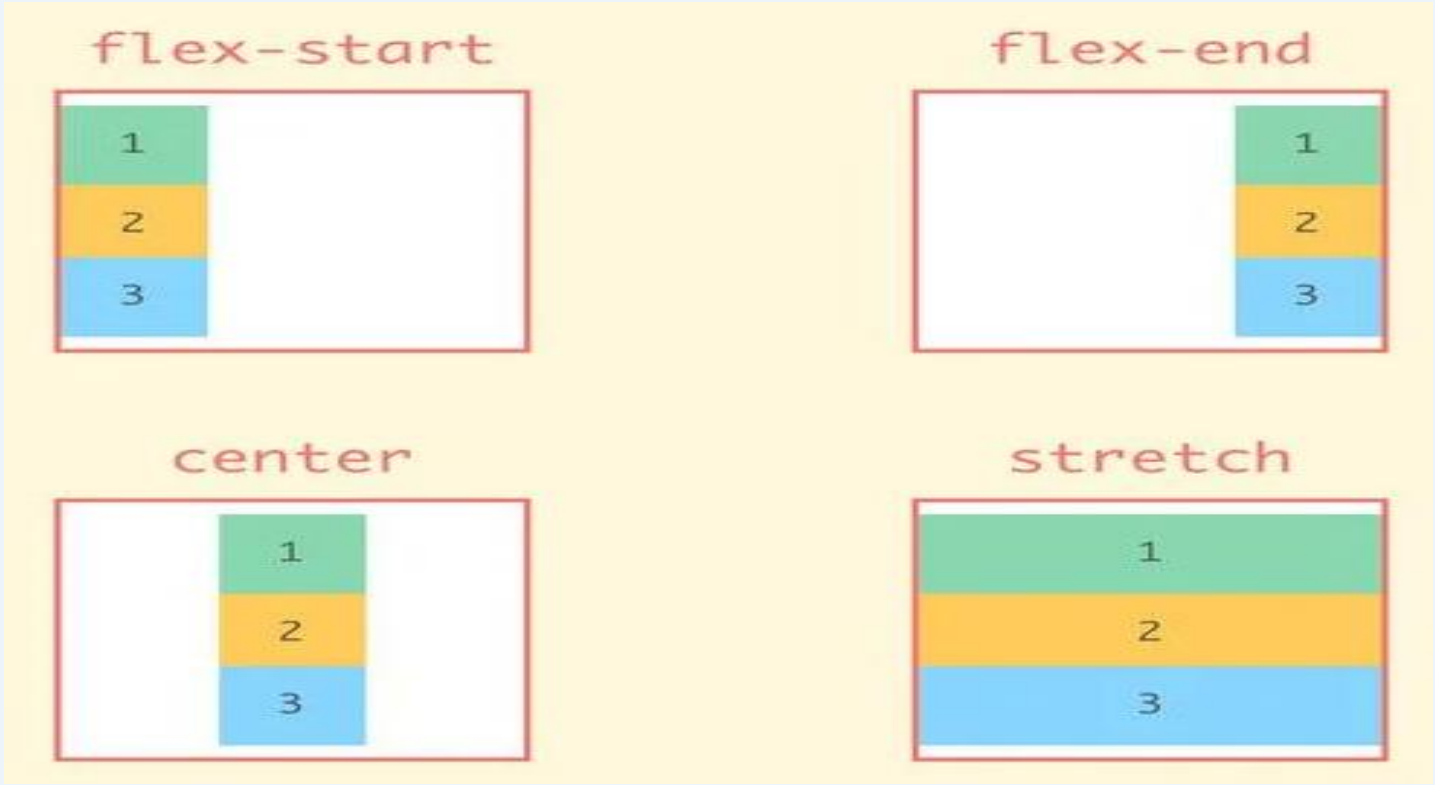
center : Align children of a container in the center of the container's cross axis

baseline : Align children of a container along a common baseline

Sets alignment along the cross-axis

alignItems

flexDirection : column



Sets alignment along the cross-axis

For stretch to have an effect, children must NOT have a fixed dimension along the cross axis

AlignItemsTest.js

flexDirection : column

justifyContent and alignItems

are the best friends

justifyContent - controls how to align children within the main axis

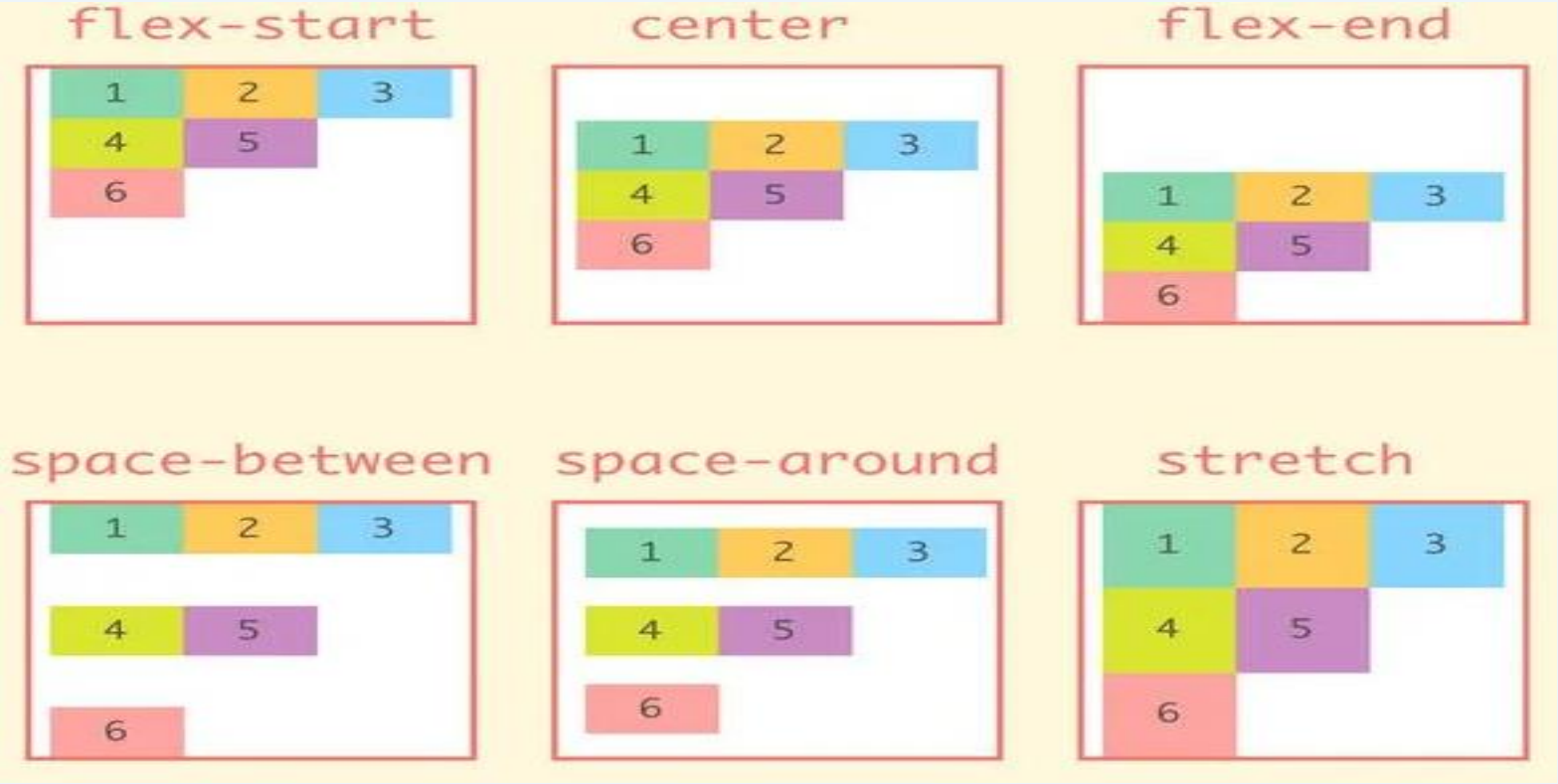
alignItems - controls how to align children within the cross axis

JustifyContentAlignItemsTest.js

alignContent

flexDirection : row
flexWrap : wrap

the distribution of lines along the cross-axis. This only has effect when items are wrapped to multiple lines using flexWrap

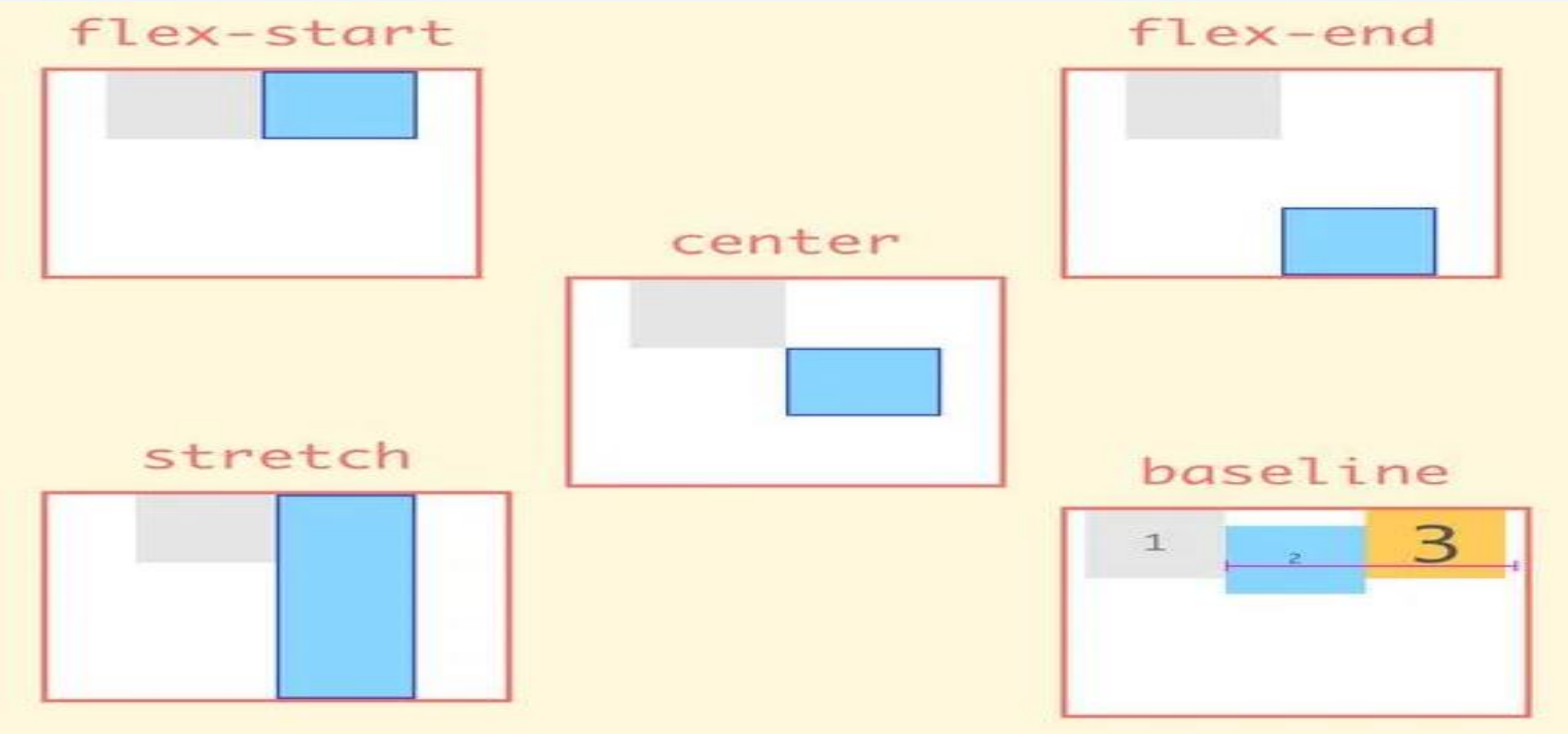


Sets distribution along the cross-axis

AlignContentTest.js

alignSelf

align an item along the cross axis overwriting his parent alignItem property



AlignSelfTest.js

flexGrow flexShrink flexBasis

flexGrow describes how any space within a container should be distributed among its children along the main axis. After laying out its children, a container will distribute any remaining space according to the flex grow values specified by its children

flexShrink describes how to shrink children along the main axis in the case in which the total size of the children overflows the size of the container on the main axis

flexBasis is an axis-independent way of providing the default size of an item along the main axis

https://yogalayout.com/docs/flex

width and height

auto (default) calculates the width/height for the element based on its content
pixels Defines the width/height in absolute pixels. this may or may not be the final dimension of the node
percentage Defines the width or height in percentage of its parent's width or height, respectively

position

relative (default) By default, an element is positioned relatively
absolute When positioned absolutely, an element doesn't take part in the normal layout flow
position is determined based on the top, right, bottom, and left values

PositionTest.js

React Native Styling References

https://reactnative.dev/docs/flexbox

https://medium.com/wix-engineering/the-full-react-native-layout-cheat-sheet-a4147802405c

https://yogalayout.com/docs/

Dimensions Module

get the application window's width and height

get the Screen width and height

Dimensions are available immediately, they may change (e.g due to device rotation) so any rendering logic or styles that depend on these constants, call this function on every render

DimensionsTest.js

- **Buttons**

ButtonExample.js

- **Touchables**

Touchable Opacity - will change the opacity of an element when touched

Touchable Highlight - will get darker and the underlying color will show through

Touchable Native Feedback - will simulate ink animation (Only For Android)

Touchable Without Feedback - touch event without any animation

TouchableExample.js

- **ScrollView**

ScrollViewExample1.js

ScrollView renders all its react child components at once

pagingEnabled
horizontal

ScrollViewExample2.js

- **TextInput**

TextInputExample1.js

TextInputExample2.js

- **Image**

Static Image Resources
Network Images
URI Data Images

ImageExample1.js

Image as Button

ImageExample2.js

- **resizeMode**

cover: (maintain Aspect Ratio)
Scale Image Width and Height ≥ Image View Dimension Width and Height

contain: (maintain Aspect Ratio)
Scale Image Width and Height ≤ Image View Dimension Width and Height

stretch: (will not maintain the aspect ratio)
Scale Image Width = Image View Dimension Width

ImageExample3.js

repeat: Repeat the image to cover the frame of the view.

center: Center the image in the view along both dimensions.
If the image is larger than the view, scale it down uniformly (for repeat and center)

https://mehrankhandev.medium.com/understanding-resizemode-in-react-native-dd0e455ce63

Device specific Images

use @2x and @3x suffixes to provide images for different screen densities

img

- **image.png**
- **image @2x.png**
- **image @3x.png**

- **ImageBackground**

ImageBackgroundExample.js

- **Picker**

PickerExample1.js

PickerExample2.js

- **Switch**

SwitchExample.js

- **ActivityIndicator**

ActivityIndicatorExample.js

- **Modal**

ModalExample.js

- **FlatList**

  similar to ScrollView

  works well for large lists of data

  shows only those renders elements which are currently displaying on the screen, not all the elements of the list at once

FlatListExample1.js

FlatListExample2.js

two required props: data and renderItem.

The data is the source of elements for the list, and renderItem takes one item from the source and returns a formatted component to render

Platform Module

Platform.OS

Platform.Version

Platform-specific extensions

MyComponent.android.js

MyComponent.ios.js

PlatformTest.js

Other Components

- **StatusBar**
- **KeyboardAvoidingView**
- **SafeAreaView**

OtherComponents.js

React Navigation

A React Navigation is a set of navigators that define the screen flow of the app

React Navigation 5 has a dynamic Component based API

Types of Navigation Patterns

- Stack Navigation
- Tab Navigation
- Drawer Navigation

React Navigation

Installation

```
npm install @react-navigation/native
```

Installing dependencies into an Expo managed project

```
expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context @react-native-community/masked-view
```

Installing the stack navigator library

```
npm install @react-navigation/stack
```

Installing the tab navigator library

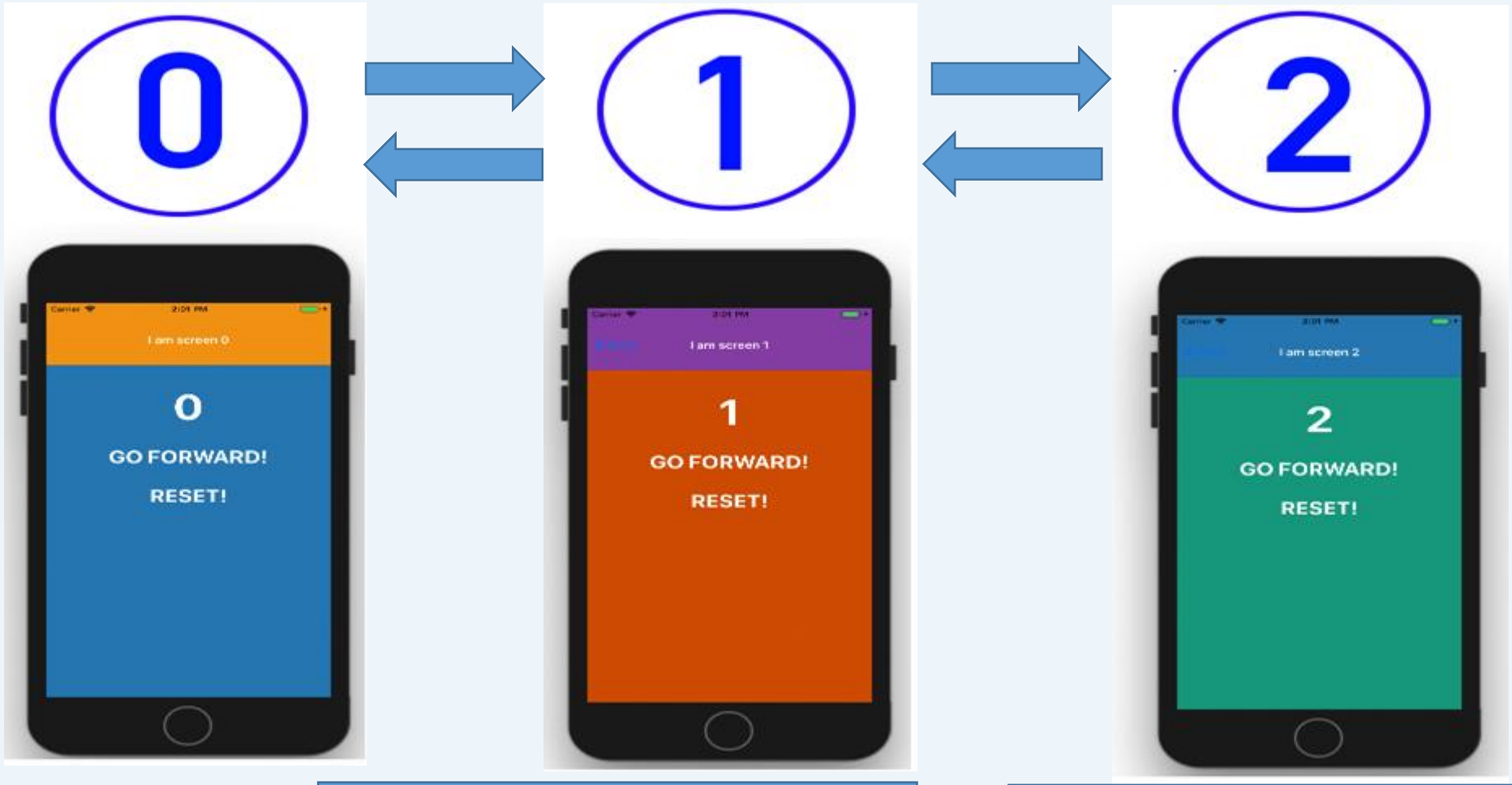
```
npm install @react-navigation/bottom-tabs
```

Installing the drawer navigator library

```
npm install @react-navigation/drawer
```

Stack Navigation

User taps a link, a new screen is put on top of old screens



StackNavigatorExample1.js

StackNavigatorExample2.js

Stack Navigation

Styling the Header
Passing Parameters between screen

Navigation Lifecycle

Consider a stack navigator with screens A and B

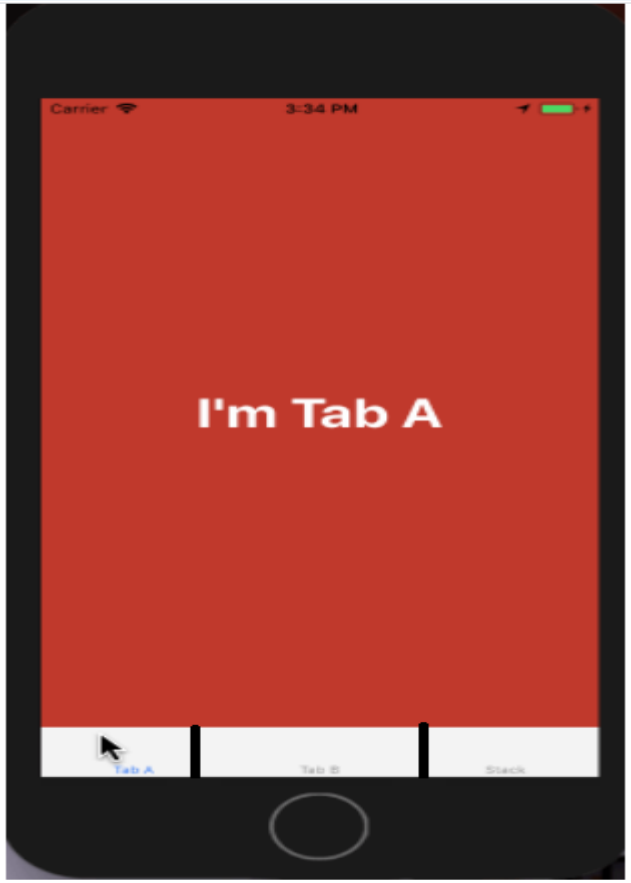
After navigating to A, its componentDidMount is called.

When pushing B, its componentDidMount is also called, but A remains mounted on the stack and its componentWillUnmount is therefore not called.

When going back from B to A, componentWillUnmount of B is called, but componentDidMount of A is not because A remained mounted the whole time.

Tab Navigation

User navigates to different screens by tapping on tabs along the top or bottom of the screen



TabNavigatorExample1.js

TabNavigatorExample2.js

Tab Navigation

Other types of Tab Navigation

MaterialBottomTabNavigator

```
npm install @react-navigation/material-bottom-tabs react-native-paper
```

TabNavigatorExample3.js

MaterialTopTabNavigator

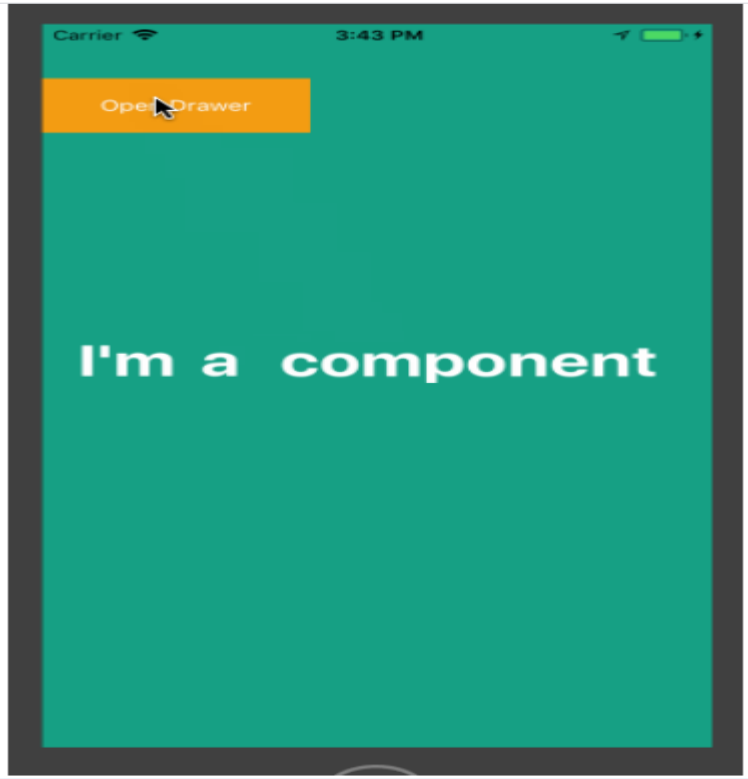
```
npm install @react-navigation/material-top-tabs react-native-tab-view
```

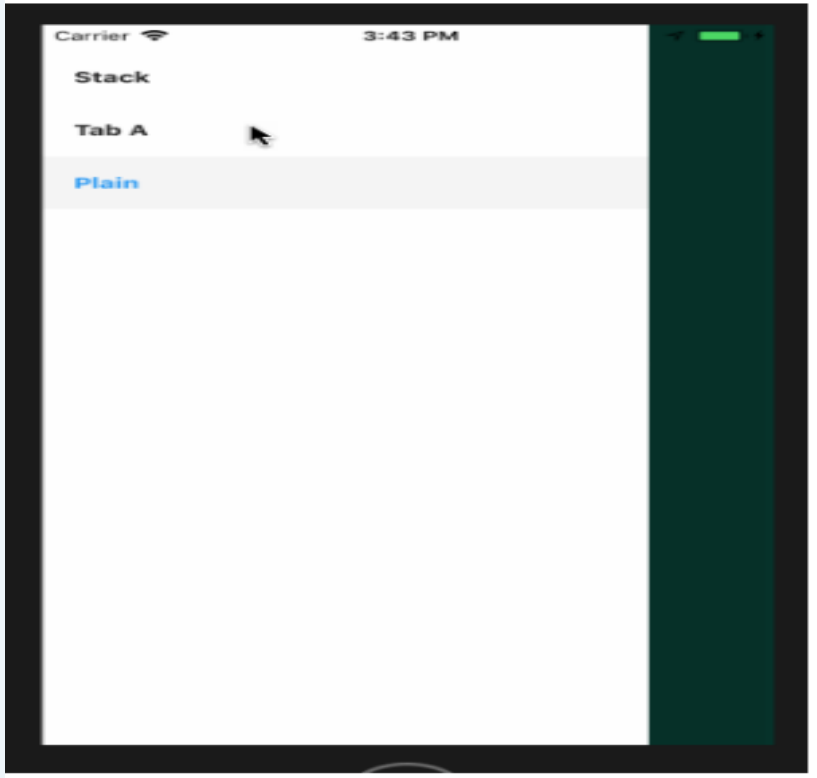
TabNavigatorExample4.js

```
npm install react-native-pager-view
```

Drawer Navigation

A slide-out 'drawer' contains links to different screens





DrawerNavigatorExample1.js

DrawerNavigatorExample3.js

DrawerNavigatorExample2.js

Nested Navigators

rendering a navigator inside a screen of another navigator

Parent navigator's UI is rendered on top of child navigator

- Stack navigators nested inside each screen of drawer navigator - The drawer appears over the header from the stack
- Tab navigator nested inside the initial screen of stack navigator - New screens cover the tab bar when you push them
- Stack navigators nested inside each screen of tab navigator - The tab bar is always visible

Expo Application

Every app in Expo can be made up of combination of React Native Core, Expo SDK, and third-party libraries

Expo SDK API Reference https://docs.expo.io/versions/latest/

React Native Third Party Libraries

React Native Directory is a searchable database of libraries built specifically for RN

https://reactnative.directory/

To look for Expo compatible Libraries, verify that it has a "✔ Expo client" tag or Filter by Expo Client

Other References https://www.awesome-react-native.com/

React Native Elements

https://react-native-elements.github.io/react-native-elements/

NativeBase https://nativebase.io/

UI Kitten https://akveo.github.io/react-native-ui-kitten/

React Native Paper https://reactnativepaper.com/

Galio https://galio.io/

Nachos https://avocode.com/nachos-ui