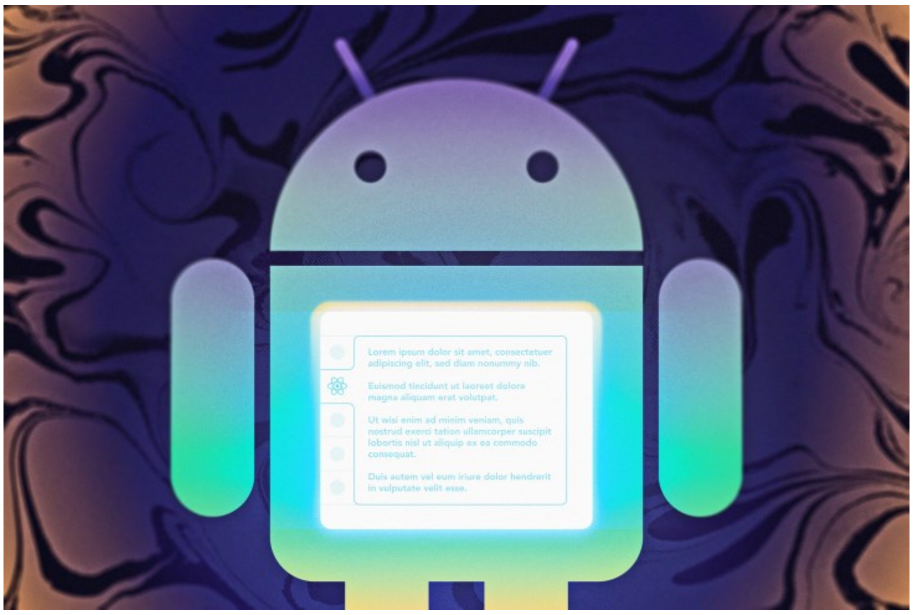
Kingsley Ubah Follow
21. Web Developer. Technical Writer. African in Tech.

Creating a vertical tab layout for React Native apps

January 4, 2023 · 5 min read



When developing a multilevel site or application containing lots of resources, you'd want to make it easy for users to navigate to these different levels from one location on the user interface. You can design a UI that accomplishes this goal using tab elements.

In this tutorial, we'll create a vertical tab layout from scratch in React Native, using CSS flexbox to ensure that the layout is responsive and will look similar on various devices. Then we'll discuss both horizontal and vertical tab layouts and when it's best to use one over the other. Jump ahead:

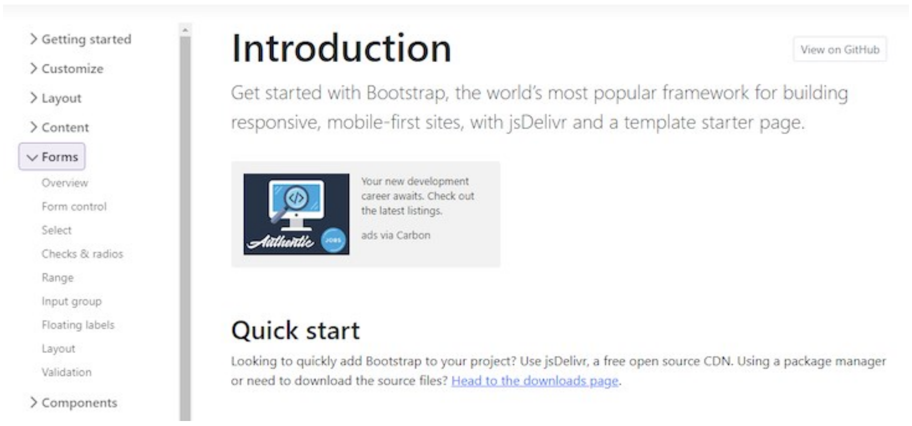
As a prerequisite for this tutorial, you should be familiar with React Native and CSS. You should also have a starter app on Expo Snack.

Why use vertical tabs for React Native apps?

A tabbed interface contains a set of clickable labeled cards or tabs. Typically, these tabs are either navigational or modular. Navigational tabs link to different pages, whereas modular tabs link to subsections of the same page.

The tabbed interface pattern is similar to tabbed dividers in a filing cabinet, which organizes the contents into clear categories and subcategories before choosing a file. Like in this real-world example, a user can go through the tab labels to find the content they want to access before clicking the tab to view it.

Tabs are great for organizing content. They let you tell the user which content falls under which category, making it easy for them to locate the content that aligns with their current aims and objectives. Here's an example from Bootstrap's site:



Introduction

Get started with Bootstrap, the world's most popular framework for building responsive, mobile-first sites, with jsDelivr and a template starter page.

Your new development career awaits. Check out the latest listings.

ads via Carbon

Quick start

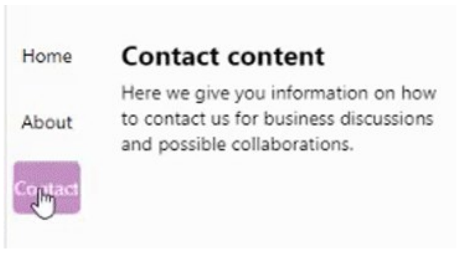
Looking to quickly add Bootstrap to your project? Use jsDelivr, a free open source CDN. Using a package manager or need to download the source files? Head to the downloads page.

Under the Forms category, you can see several subcategories linking to other parts of the page. Now, let's learn how to create our own simple vertical tab menu in React Native.

Previewing our vertical tab layout project

In a vertical tab layout, the tabs are arranged vertically on the left. Each tab label corresponds to a block of related content and only one tab can be active at a time.

Here's a preview of the final project:



Contact content

Here we give you information on how to contact us for business discussions and possible collaborations.

Before we move on to the next section, you should have your starter app at Expo Snack already set up. If you don't, simply go to snack.expo.dev in your web browser.

Importing project components

Start by deleting all the code in the `App.js` file. Then, in the same file, make the following imports:

```
import * as React from 'react';
import { StyleSheet, View, Image, TouchableOpacity, Text, Dimension, FlatList, Sa
import {useState} from 'react'
```

Here we imported the core React library, several components we'll use to build our UI, and the `useState` Hook to manage the application state.

Defining tab labels and content

We'll use `listTab` to define our tab labels and `data` to define the content each of them will link to. See how in the code below:

```
const listTab = [
  {
    status: 'Home'
  },
  {
    status: 'About'
  },
  {
    status: 'Contact'
  },
]

// Linking content
const data = [
  {
    name: "Home content",
    text: "This is my homepage. Here I welcome you to my website and try me bes
```

Each content object has a `status` property matching the status within `listTab`. That's how we'll be linking the tab labels to their corresponding data.

Setting up functionality for React Native tab elements

In the next code block, we will define the `App` component function:

```
export default function App() {
  // the following code goes here
}
```

Inside `App`, begin by setting the tab to `Home` by default. Make sure the `data` for `Home` is displayed as well:

```
const [status, setStatus] = useState('Home')
    const [dataList, setDataList] = useState([...data.filter(e => e.status === 'I
```

Now you need a way to activate a tab item once the user clicks on it. We'll be doing this using the `setStatusFilter` function:

```
const setStatusFilter = status => {
      if(status !== 'Home') {
        setDataList([...data.filter(e => e.status === status)])
      }else {
        setDataList([...data.filter(e => e.status === 'Home')])
      }

      setStatus(status)
    }
```

The above function is executed when a tab is clicked. Inside the function, we simply update the `dataList` to display content related to whichever tab the user clicks.

Next, we'll return a `SafeAreaView` element containing both the tab menu and content list. Inside the first `<View>` component, we'll loop through the tab items with the `map()` array function. For each item, we'll render a `TouchableOpacity` element with the tab label.

We'll also dynamically apply the `btnTabActive` class to the active tab and call the `setStatusFilter` method to update the active status when the user presses the element:

```
return (
      <SafeAreaView style={styles.container}>
        <View style={styles.listTab}>
          {
            listTab.map((e) => {
              return (
                <TouchableOpacity
                  style={[styles.btnTab, status === e.status && styles.btnTabAc
                  onPress={() => setStatusFilter(e.status)}
                >
                  <Text style={[styles.textTab, status === e.status && styles.te
                </TouchableOpacity>
              )
            })
          }
        </View>
```

Below the `<View>` component is the `<FlatList>` component. `<FlatList>` takes in the data related to the active tab and renders it by calling the `renderItem` function. Here's the function body:

```
const renderItem = ({item, index}) => {
      return (
        <View key={index} style={styles.itemContainer}>
          <Text style = {styles.itemName}>{item.name}</Text>
          <Text>{item.text}</Text>
        </View>
      )
 }
```

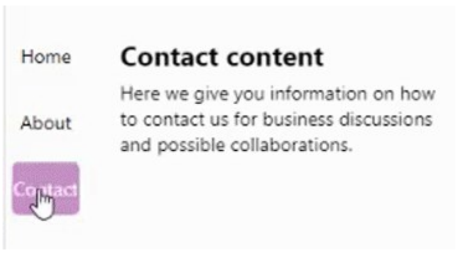
In the code above, we're rendering two `Text` components to display the title and text related to the active tab.

Styling our responsive React Native vertical tab layout

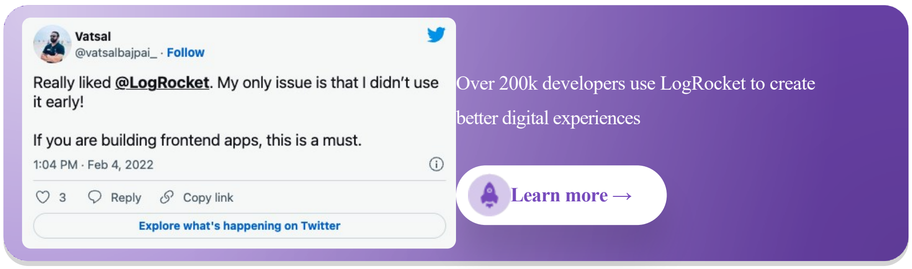
Finally, here is the stylesheet responsible for styling the elements. Note that we're using the CSS flexbox layout system to achieve a responsive vertical tab layout that will look similar between different devices:

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    flexDirection: 'row',
    paddingHorizontal: 10,
    justifyContent: 'center'
  },
  listTab: {
    backgroundColor: '#fff',
    flexDirection: 'column',
    justifyContent: 'flex-start',
    marginTop: 20,
    marginLeft: 5,
    height: 100
  },
  btnTab: {
    width: 50,
    flexDirection: 'row'
```

That's it! You now have a working navigation tab. When you click on a tab, its corresponding content will be displayed to the right, as you can see below:


Home
About
Contact
Contact content
Here we give you information on how to contact us for business discussions and possible collaborations.

We mentioned that there are two kinds of tab layouts — vertical and horizontal. Let's learn more about each of them.


Vatsal
@vatsalbajpai_ · Follow

Really liked @LogRocket. My only issue is that I didn't use it early!

If you are building frontend apps, this is a must.

1:04 PM · Feb 4, 2022

3 Reply Copy link

Explore what's happening on Twitter

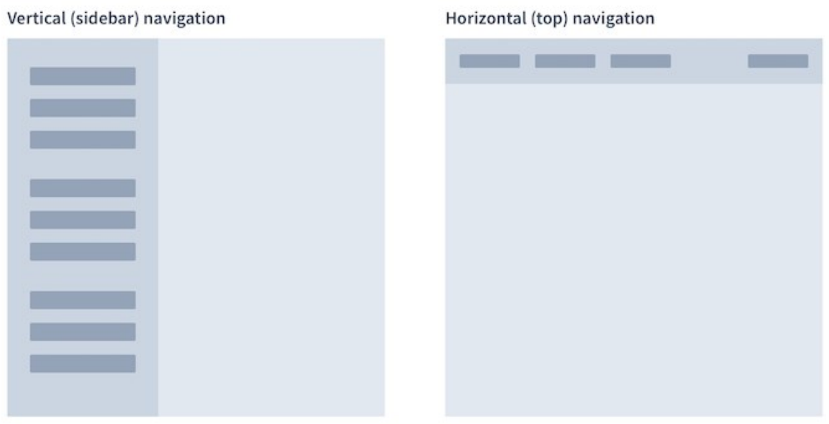
Over 200k developers use LogRocket to create better digital experiences

Learn more →

When to use vertical tabs vs. horizontal tabs

Vertical tabs, commonly known as sidebars, are a list of links placed on either side of the page and its main content. A horizontal tab layout, on the other hand, is placed at the top of the page. You can make it sticky so that it appears no matter where you navigate on the page.

Below, you can see the difference between the placement of vertical tab elements that appear on the left side of the screen and horizontal tab elements that appear at the top of the screen:



When building the user interface of your site or app, you'll first need to decide whether to use a vertical or horizontal tab layout for your site navigation. So when should you use one over the other?

It all comes down to available space. In a vertical tab layout, the tabs and content both share the same horizontal space, which means limited space for your content. Unless your intention is to grab user attention while they're on your content, vertical tabs might not be a good idea.

Vertical tabs are best used on large screens. For example, you can use media queries to apply a vertical tab layout when the screen is large enough to contain both the tab and main content, but then switch to a horizontal layout on smaller screens to free up more space for content.

Conclusion

I hope this tutorial helped you understand vertical and horizontal tabs better. You should now be able to build a navigation tab from scratch in a React Native app using the techniques we covered in this post.

If you have any questions regarding this topic, leave a comment and I'll respond as soon as I get it. Have a great week!

LogRocket: Instantly recreate issues in your React Native apps.

Share this:

Twitter | Reddit | LinkedIn | Facebook

#react native

« Conversion funnel analysis: How to optimize your customer journey

Build a React app using Grommet for the UI »

LogRocket
Frontend Analytics

Stop guessing about your digital experience with LogRocket

Get started for free

How to speed up incremental builds with Gatsby's Slice...

Peter Ekene Eze
Feb 15, 2023 · 5 min read

Getting started with Inertia.js, Vue.js, and AdonisJs

Emmanuel Odioko
Feb 8, 2023 · 11 min read

Building a carousel component in React using Hooks

Florian Rappl
Feb 1, 2023 · 9 min read

Leave a Reply

Enter your comment here...

Search