A Simple Way To Run JS in Background Thread on React Native

Takuya Matsuyama Follow
Mar 13, 2018 · 2 min read

CPU intensive tasks block UI, like indexing. Because, in React Native, JavaScript is executed on JavaScriptCore which means that you have only 1 thread. So you have to use a native module like react-native-workers which provides similar API as web workers. But it's kind of an overspec way if you just want to run a simple task in background. I don't feel like installing many native modules into my app because they would make the app more complicated and fragile. If you have an expo app, it can't use native modules.

I found that we already have background threads out of the box. That is, WebView. You can run JavaScript in it by calling `injectJavaScript` method. Inside webview, it is another instance of Safari(iOS)/Chrome(Android), so JS running in it won't block the app UI at all. I checked that on both platforms by running following code:

```
for (;;) { Math.random() * 9999 / 7 }
```

This is useful. You don't have to install native modules to run code in background thread!
Here is an example:

```
import React, { Component } from 'react'
import { WebView } from 'react-native'

export default class BackgroundTaskRunner extends Component {
  render() {
    return (
      <WebView
```

```
          ref={el => this.webView = el}
          source={{html: '<html><body></body></html>'}}
          onMessage={this.handleMessage}
        />
      )
    }
    runJSInBackground (code) {
      this.webView.injectJavaScript(code)
    }
    handleMessage = (e) => {
      const message = e.nativeEvent.data
      console.log('message from webview:', message)
    }
  }
```

To get a result of the code, you can specify `onMessage` prop to your webview.
A function that is invoked when the webview calls `window.postMessage`. Setting this property will inject a `postMessage` global into your webview, but will still call pre-existing values of `postMessage`.
`window.postMessage` accepts one argument, `data`, which will be available on the `event` object, `event.nativeEvent.data`. `data` must be a string.

Just call it on webview:

```
const message = { ok: 1 }
window.postMessage(message)
```

Then you get the message on the app:

```
message from webview:, { ok:1 }
```

That's it! 😄

React React Native JavaScript Programming