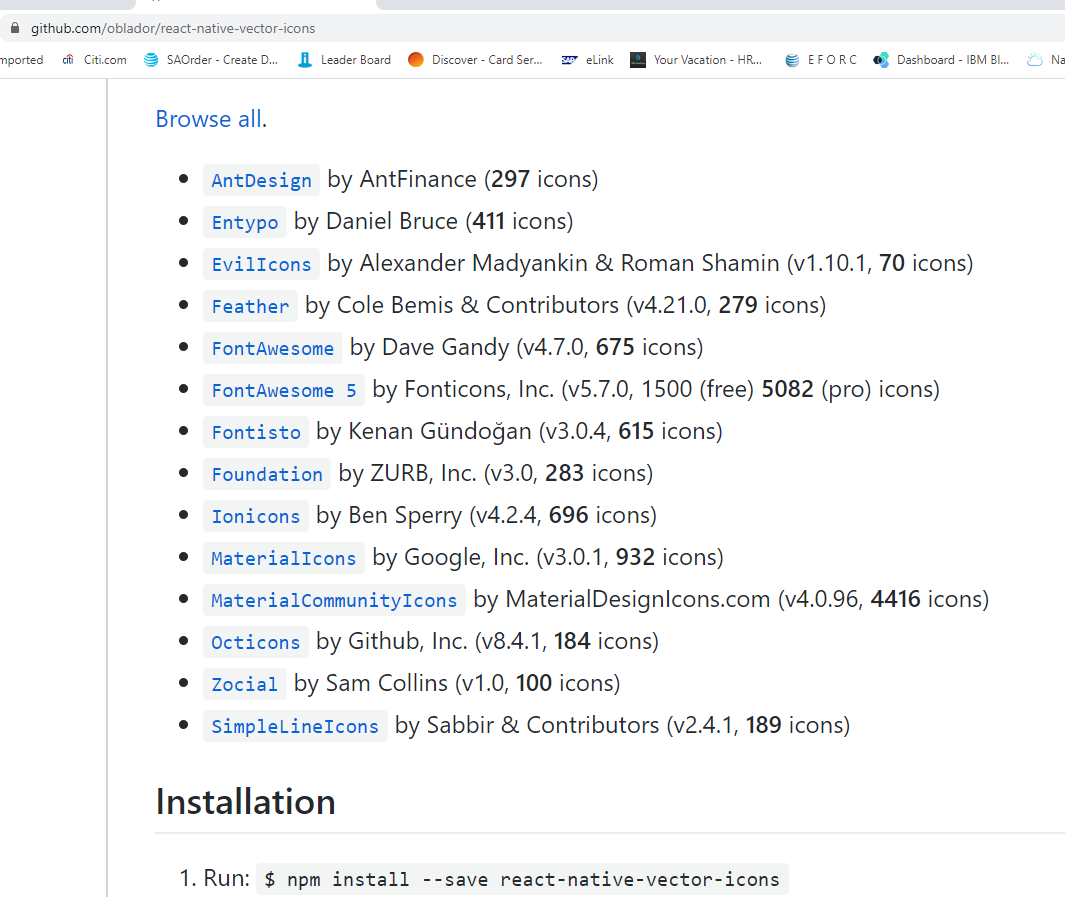
<https://github.com/oblador/react-native-vector-icons>



*getDecks: return all of the decks along with their titles, questions, and answers.  
getDeck: take in a single id argument and return the deck associated with that id.  
saveDeckTitle: take in a single title argument and add it to the decks.  
addCardToDeck: take in two arguments, title and card, and will add the card to the list of questions for the deck with the associated title.*

xx = {

React: {

title: 'React',

cards: [

{ cardid: '5ni6ok3ym7mf1p33lnez',

question: 'What is React?',

answer: 'A library for managing user interfaces',

quiz:’W’

},

{ cardid: '6ni6ok3ym7mf1p33lnez',

question: 'Where do you make Ajax requests in React?',

answer: 'The componentDidMount lifecycle event',

quiz:’C’

}

]

},

JavaScript: {

title: 'JavaScript',

cards: [

{ cardid: '4ni6ok3ym7mf1p33lnez',

question: 'What is a closure?',

answer: 'The combination of a function and the lexical environment within which that function was declared.',

quiz:null

}

]

}

}

    console.log(xx['React'].title)

    console.log(xx['React'].cards[1].answer)

    console.log(xx['JavaScript']['cards'][0].question)

In order to use Redux Store in Expo

import { registerRootComponent } from 'expo';

import {createStore} from 'redux'

import middleware from './middleware'

import {Provider} from 'react-redux'

import reducer from './reducers'

import App from './App';

// registerRootComponent calls AppRegistry.registerComponent('main', () => App);

// It also ensures that whether you load the app in the Expo client or in a native build,

// the environment is set up appropriately

class rootNode extends Component {

render(){

return (

<Provider store={createStore(reducer, middleware)}>

<App/>

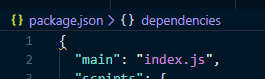
</Provider>

)}

}

registerRootComponent(rootNode);

Error: Could not find "store" in the context of "Connect(App)". Either wrap the root component in a <Provider>, or pass a custom React context provider to <Provider> and the corresponding React context consumer to Connect(App) in connect options.



Since we want to know if the flashcard is quiz correct or wrong, we need store it as well.

If answer correctly, quiz:’C’, if wrong, quiz: ‘W’ if not quizzed yet, quiz:null

There is something very strange or I did not fully understand the Redux about combineReducer.

The UI did get update if my reducer (index.js) is as:

import { combineReducers } from 'redux'

import decks from './decks'

export default combineReducers({

    decks,

})

But if I put the content of deck.js inside index.js  (reducer) directly…UI does not update after load the data in ComponentDidMount.

import {LOAD\_DATA} from '../actions/decks'

export default function decks (state = [], action) {

    switch(action.type) {

      case LOAD\_DATA :

        return {

            ...action.decks

        }

      default :

        return state

    }

  }

We have to use combineReducers

In order to know what deck is active when “Add Card” is pressed, we have to store “ActiveDeck” in state/store.

So, other than Decks in store/action, I am going to add ActiveDeck in action/reducer/state.

The basic elements in RN ( can not use bootstrap components)

ScrollView,Button,Picker,Switch

TextInput,Text,View,Image,StyleSheet

StatusBar,Alert,ActivityIndicator,Linking, Animated

Note…do not invoke the function…just pass the function

 <TouchableOpacity onPress={ this.saveCard} >

 <Text style={styles.submitButtonText}> Submit </Text> <TouchableOpacity>

<TextInput placeholder="Answer" onChangeText={this.handleAnswer} />

saveCard = (decktitle)=>{your code here }

 onPress={this.saveCard(title)} > will invike everytime when it type in input…NOT GOOD!

Since I have only a few decks….scrollview will do…no need to use FlatList (for better performace)

KeyboardAvoidingView to avoid keyboard on top of input.