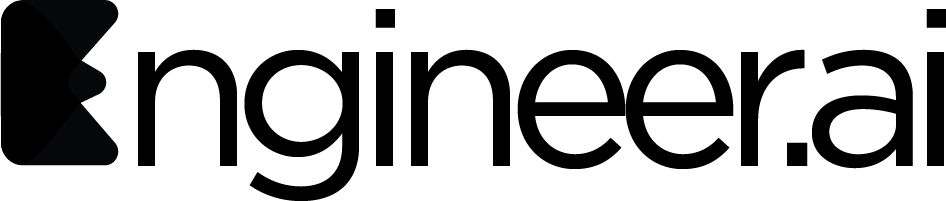
Capacity Partner Guide Building Blocks (ReactNative)

Executive Approver: Tooraj Helmi

Authors: @joseph.rifkin and @david.he

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **KICK OFF** | **DRAFT** | **COMPLETE** | **PRM** | **RELEASE** |



# 

# 

# Version Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Changes | Author | Date | Comments |
| 1 | Initial | @joseph.rifkin @david.he | 4/3/20 |  |
| 2 |  |  |  |  |

[**Version Control**](#_xv2jezl46w8b) **2**

[**Getting Started**](#_wqduy1x0ua4b) **5**

[**Prerequisites**](#_uho77ep4vm3a) **5**

[Homebrew](#_7su4fpnycbxg) 5

[Git](#_az2jk1g5t0lb) 5

[Node](#_j4a3g6dw93lp) 5

[Yarn](#_5otsk0b6t3r8) 5

[Watchman](#_vukgl3wnlmc) 5

[ReactNative](#_7rwevvmmcqji) 6

[JDK 13 & JDK 11](#_d5qftg60ykao) 6

[Android Studio](#_a58nmr1c5fcf) 6

[XCode](#_j5116mr2npb) 6

[Android SDK](#_glty6raeejtd) 6

[XCode Command Line Tools](#_cnfqqqtiea44) 6

[**Recommend Tools**](#_u3ir542s8w1a) **7**

[SourceTree](#_qhgeaa3fvkpb) 7

[Postman](#_hdnyfwcxmrbr) 7

[Visual Studio Code (or your IDE of choice)](#_hg2sjawtoc0i) 7

[**Building Blocks Process**](#_6lsu8ncgaewx) **7**

[**React Native Application Process**](#_renwdrfipmjx) **7**

[**Running React Native Build Blocks Application**](#_1jtkbg7gwg12) **8**

[Web](#_lcrq02ks6n42) 8

[iOS](#_c0iz74mbig95) 8

[Android](#_3rae9hc43bz9) 8

[**Run Release and Generate iOS and Android Apps**](#_1eg3eugvbzhw) **8**

[Web](#_j8sh9v1q3qlq) 8

[iOS](#_n7sjlwvdiwxl) 8

[Android](#_vwu7vvlu3m3h) 9

[**Adding node packages**](#_s0rkg8bqabz1) **10**

[**Development Files Rules**](#_s3oex6yl3kzw) **11**

[Project Structure](#_ioh91ybnfaq7) 12

[Block Code Structure View vs. Controller](#_3p99i5ul0pyk) 12

[Block Inheritance Hierarchy](#_efflrlw6fu2a) 13

[Git branch repo](#_o6lrn4i58cjg) 13

[React Native Routers File Extension Convention](#_ggsqhay5il1q) 14

[Building Blocks Framework Reference and Examples](#_bum6r674vhzg) 14

[React Native Mono Repo Architecture](#_yry0ejmuuvja) 14

[RunEngine](#_hzjqjhx0i654) 14

[Messages](#_qpozc4gqkd8l) 14

[Create RunEngine Message](#_d5jydiizgcyp) 14

[Get data from a RunEngine Message](#_whm4wzuadlw8) 15

[Add data to a RunEngine Message](#_bp4up26k23nu) 15

[BlockComponent](#_6t8bug8r8jzg) 15

[Register for a RunEngine Message](#_a3iipn8wtbbl) 15

[Receive a RunEngine Message](#_u7d805vw2ffy) 15

[Show Alert Modal](#_kgg5onbcyv9c) 15

[Helpers](#_afzrhbvav7pn) 16

[Rest Client Block](#_5sj7ghf8dy23) 16

[Example Api Call Request](#_yl9y1ngt0yh) 16

[Navigation Block](#_1uqjwm9pte6t) 19

[Adapters](#_dbls5zrh8rcx) 19

[Example Basic Adapter](#_5m87r3rk4iao) 21

[Example Adapter w/ Data Passed w/ Navigation](#_3lqcaqjqhsw2) 22

[Component vs Block](#_dfrhn67dgzmo) 25

[Mobile Route Map](#_774ahrrfstbt) 26

[Web Route Map and webpack](#_iahd0k8gof99) 27

[See EXAMPLE adding Component to webpack](#_s4vpmxtnu3vt) 29

[**Using existing packages**](#_u8jkgwbofyxo) **30**

[React native web](#_64f7fhdn2j4e) 30

[Unit and UI Testing](#_j7zg1xdqnb7c) 30

[React-native elements](#_ffe88wyzaluy) 30

[Material UI](#_t694a812ik0o) 31

[React-native-vector-icons](#_yerq3lrn90fw) 31

[**Coding Best Practices**](#_mnb7eqqnaf6s) **31**

[Using images and other assets](#_v0dvljppy037) 31

[Creating a custom component vs using existing one](#_tlze6x7x6dml) 32

[Putting string into config files](#_4ai2kqixc4t9) 32

[Using Stylesheet.Create vs external .css file](#_7b3ob93voqku) 32

[Styling blocks - Using flexbox](#_4h26wka7yrcg) 33

[Eslint](#_d2e797tzfttw) 34

[Typescript](#_1ny9e8f7bxvl) 34

[Prettier](#_b91ax3kbnndj) 35

[Javascript / Typescript general standards](#_6muw0ipasik3) 35

React Native Complete Block and Application

### Getting Started

The first step is to starting the block development is to make sure you have the following prerequisite tools.

### Prerequisites

#### Homebrew

|  |
| --- |
| $ **/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"** |

#### Git

|  |
| --- |
| $ brew install git |

#### Node

|  |
| --- |
| $ brew install node |

#### Yarn

|  |
| --- |
| $ brew install -g yarn |

### 

#### Watchman

|  |
| --- |
| $ brew install watchman |

### 

#### ReactNative

|  |
| --- |
| $ sudo npm install -g react-native-cli |

#### JDK 13 & JDK 11

|  |
| --- |
| $ brew tap homebrew/cask-versions $ brew cask install java $ brew cask install java11 $ brew cask install java13 |

#### Android Studio

<https://developer.android.com/studio/install.html>

#### XCode

<https://itunes.apple.com/us/app/xcode/id497799835?mt=12>

#### **Android** **SDK**

|  |
| --- |
| $ brew cask install android-sdk |

#### XCode Command Line Tools

|  |
| --- |
| $ xcode-select --install |

|  |
| --- |
| $ sudo gem install cocoapods |

### 

### Recommend Tools

#### SourceTree

<https://www.sourcetreeapp.com/>

#### Postman

<https://www.postman.com/>

#### Visual Studio Code (or your IDE of choice)

<https://code.visualstudio.com/>

### Building Blocks Process

Block Requirements will contain a [branch] for the Block Developer

|  |
| --- |
| $ git clone -b [branch] git@gitlab.builder.ai:builder/builder-bx/Bx/blocks/react-native/development/react-native-master-app/boiler-plate.git --recursive |

|  |
| --- |
| $ cd boiler-plate && yarn |

### React Native Application Process

Project Requirements will contain a [branch] and repo locations for Project

|  |
| --- |
| $ git clone -b [branch] PROJECT\_REPO --recursive |

|  |
| --- |
| $ cd PROJECT\_REPO\_NAME && yarn |

(Example of E2E Application From Blocks: <https://docs.google.com/document/d/1tBl1l2SIAC9pSPlG4jyvdVlcDEH-K6Mt7oYaRs0gTYo/edit> **)**

### Running React Native Build Blocks Application

#### Web

|  |
| --- |
| $ yarn workspace web start |

#### iOS

|  |
| --- |
| $ cd packages/mobile/ios && pod install && cd ../../../ && npx react-native bundle --entry-file ./packages/mobile/index.js --platform ios --dev true --bundle-output ./packages/mobile/ios/main.jsbundle --assets-dest ios && yarn ios |

#### Android

|  |
| --- |
| $ export JAVA\_HOME=`/usr/libexec/java\_home -v 11`; java -version; export ANDROID\_HOME=${HOME}/Library/Android/sdk; export PATH=${PATH}:${ANDROID\_HOME}/emulator; yarn android |

### Run Release and Generate iOS and Android Apps

#### Web

|  |
| --- |
| $ yarn workspace web build |

#### iOS

|  |
| --- |
| $ cd packages/mobile/ios && pod install && cd ../../../ && npx react-native bundle --entry-file ./packages/mobile/index.js --platform ios --dev true --bundle-output ./packages/mobile/ios/main.jsbundle --assets-dest ios && yarn xcode |

<https://developer.apple.com/documentation/xcode/preparing_your_app_for_distribution>

#### Android

|  |
| --- |
| $ export JAVA\_HOME=`/usr/libexec/java\_home -v 11`; java -version; export ANDROID\_HOME=${HOME}/Library/Android/sdk; export PATH=${PATH}:${ANDROID\_HOME}/emulator; cd packages/mobile/android; ./gradlew assembleRelease |

### 

### 

### Adding node packages

Navigate to platform specific package.json containing folder

Common - Node Packages

|  |
| --- |
| ./ |

Mobile - Node Packages Only Used for Mobile

|  |
| --- |
| ./packages/mobile |

Web - Node Packages Only Used for Web

|  |
| --- |
| ./packages/web |

|  |
| --- |
| $ npm install [package\_name] --save |

OR

|  |
| --- |
| $ npm install [package\_name] --save-dev |

Navigate to the project root

|  |
| --- |
| $ yarn |

|  |
| --- |
| $ cd packages/mobile/ios && pod install && cd ../../../ |

* Update packages/web/config-overdies.js(See Appendix)
* Follow Android Linking Instruction from package
* Perform any platform specific link required

### 

### Analytics

#### Web

Update values in packages/web/src/App.js

const firebaseAPI = firebase.initializeApp({

apiKey: "AIzaSyDgl9aTbKMdRZ9-ijSZRionh3V591gMJl4",

authDomain: "rnmasterapp-c11e9.firebaseapp.com",

databaseURL: "https://rnmasterapp-c11e9.firebaseio.com",

projectId: "rnmasterapp-c11e9",

storageBucket: "rnmasterapp-c11e9.appspot.com",

messagingSenderId: "649592030497",

appId: "1:649592030497:web:7728bee3f2baef208daa60",

measurementId: "G-FYBCF3Z2W3"

});

#### Mobile

See platform specific configurations (<https://firebase.google.com/docs/analytics>)

### Development Files Rules

* Start with common **Web & Mobile** Component Controller and Component View using react-native and js
* If there is a layout only issue use **Web & Mobile** Component Controller + **Mobile** and **Web** Component Views
* Last if there is a truly unique **Web** or **Mobile** only dependency (ie. Facebook/Google Log In) create **Mobile** and **Web** (ReactNative/JS/ReactJS) Component Controller + **Mobile** and **Web** (ReactNative/JS/ReactJS) Component Views
* If there is unique code other than styles needed for **iOS** vs **Android** then a Component will be created and placed under **./packages/component**
* **Styles** should be use react-native Platform.OS

#### Project Structure

|  |
| --- |
| Project Root ├── packages  ├── blocks  ├── components  ├── src  ├── framework  ├── package.json  ├── src  ├── Blocks  ├── Messages  ├── mobile  ├── package.json  ├── App.tsx  ├── index.js  ├── ios  ├── android  ├── web  ├── package.json  ├── config-overrides.js  ├── src  ├── App.ts  ├── inex.js |
|  |

#### Block Code Structure View vs. Controller

|  |
| --- |
| Block ├── src  ├── BlockScreenController.tsx  ├── BlockScreen.tsx  ├── config.tsx  ├── assets |

BlockScreenController

* Inherits from BlockComponent<Props,S,SS>
  + Contains Block Logic

BlockScreen

* Inherits from BlockScreenController
  + Implements Component Render Method

config.tsx

* Block Text
* Block End-points
* Block Asset Objects

#### Block Inheritance Hierarchy



#### Git branch repo

* dev
* stg
* prod
* features/[Ticket #]
* bugs/[Ticket #]
* tasks/[Ticket #]

#### React Native Routers File Extension Convention

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Web & Mobile Shared** | **Mobile** | **Web** | **iOS** | **Android** |
| Block.js | Block.js | Block.web.js | Component.ios.js | Component.android.js |
| Block.ts | Block.ts | Block.web.js | Component.ios.js | Component.android.js |
| Block.tsx | Block.tsx | Block.web.js | Component.ios.js | Component.android.js |

#### Building Blocks Framework Reference and Examples

##### React Native Mono Repo Architecture

<https://github.com/brunolemos/react-native-web-monorepo>

##### 

##### RunEngine

Debug Logs

|  |
| --- |
| runEngine.debugLog("Message", OBJECT); |

How to send RunEngine Message

|  |
| --- |
| const msg = new Message(getName(MessageEnum.CountryCodeMessage)); runEngine.sendMessage(getName(MessageEnum.CountryCodeMessage), msg); |

##### Messages

###### Create RunEngine Message

|  |
| --- |
| const msg: Message = new Message(getName(MessageEnum.NavigationNewPasswordMessage));  msg.addData(getName(MessageEnum.NavigationPropsMessage), this.props);  this.send(msg); |

###### Get data from a RunEngine Message

|  |
| --- |
| var navProps = message.getData(getName(MessageEnum.NavigationPropsMessage)); |

###### Add data to a RunEngine Message

|  |
| --- |
| requestMessage.addData(  getName(MessageEnum.RestAPIResponceEndPointMessage),  loginAPiEndPoint  ); |

##### BlockComponent



###### Register for a RunEngine Message

|  |
| --- |
| constructor(props: Props) {  super(props);  //onUnMount will unsubscribe from this.subScribedMessages  this.subScribedMessages = [  getName(MessageEnum.RestAPIResponceMessage),  getName(MessageEnum.NavigationPayLoadMessage)  ];  this.receive = this.receive.bind(this);  runEngine.attachBuildingBlock(this, this.subScribedMessages); } |

###### Receive a RunEngine Message

|  |
| --- |
| async receive(from: string, message: Message) { } |

###### Show Alert Modal

|  |
| --- |
| showAlert(error: string, btnPositiveText?: string, btnPositiveMessage?: Message, btnNegativeText?: string, btnNegativeMessage?: Message,btnNeutralText?: string, btnNeutralMessage?: Message) |

###### Helpers

|  |
| --- |
| isPlatformWeb() isPlatformiOS() isPlatformAndroid() |

##### 

##### Rest Client Block

* Listens for *RestAPIRequsetMessage*
* Preforms API calls
* Provides an automatic response with *RestAPIResponseDataMessage*

|  |
| --- |
| private constructor() {  super();  runEngine.attachBuildingBlock(this as IBlock, [  getName(MessageEnum.RestAPIRequestMessage)  ]);  }   async receive(from: string, message: Message) {  } |

#### 

###### Example Api Call Request

|  |
| --- |
| async doEmailLogIn() {   if (  this.state.email == null ||  this.state.email.length == 0 ||  !this.emailReg.test(this.state.email)  ) {  this.showAlert("Error", errorEmailNotValid);  return;  }   if (this.state.password == null || this.state.password.length == 0) {  this.showAlert("Error", errorPasswordNotValid);  return;  }   const header = {  "Content-Type": loginApiContentType  };   const attrs = {  email: this.state.email,  password: this.state.password  };   const data = {  type: "email\_account",  attributes: attrs  };   const httpBody = {  data: data  };   const requestMessage = new Message(getName(MessageEnum.RestAPIRequestMessage));   this.apiEmailLoginCallId = requestMessage.messageId;  requestMessage.addData(  getName(MessageEnum.RestAPIResponceEndPointMessage),  loginAPiEndPoint  );   requestMessage.addData(  getName(MessageEnum.RestAPIRequestHeaderMessage),  JSON.stringify(header)  );   requestMessage.addData(  getName(MessageEnum.RestAPIRequestBodyMessage),  JSON.stringify(httpBody)  );   requestMessage.addData(getName(MessageEnum.RestAPIRequestMethodMessage), loginAPiMethod);   runEngine.sendMessage(requestMessage.id, requestMessage);  } |

|  |
| --- |
| async makeApiPostCall(  uniqueApiCallId: string,  endpoint: string,  headers: string,  body: string  ) {  let apiResponseMessage = new Message(  getName(MessageEnum.RestAPIResponceMessage)  );  apiResponseMessage.addData(  getName(MessageEnum.RestAPIResponceDataMessage),  uniqueApiCallId  );   try {  const response = await fetch(endpoint, {  method: 'POST',  headers: JSON.parse(headers),  body  });   const responseJson = await response.json();   // setting Response  apiResponseMessage.addData(  getName(MessageEnum.RestAPIResponceSuccessMessage),  responseJson  );  } catch (error) {  runEngine.debugLog('RestApiClient Error', error);  apiResponseMessage.addData(  getName(MessageEnum.RestAPIResponceErrorMessage),  'An error has occuured. Please try again later.'  );  }   if (this.props) {  apiResponseMessage.addData(  getName(MessageEnum.NavigationPropsMessage),  this.props  );  }   this.send(apiResponseMessage);  } |

|  |
| --- |
| async receive(from: String, message: Message) {    if (  getName(MessageEnum.RestAPIResponceMessage) === message.id &&  this.anExampleCallID != null &&  this.anExampleCallID === message.getData(getName(MessageEnum.RestAPIResponceDataMessage))  ) {   var responseJson = message.getData(  getName(MessageEnum.RestAPIResponceSuccessMessage)  );    if ( responseJson !== undefined && responseJson?.errors === undefined) {  // you have response   } else {  var errorReponse = message.getData(  getName(MessageEnum.RestAPIResponceErrorMessage)  );    this.parseApiCatchErrorResponse(errorReponse);  }    } } |

##### Navigation Block

* Listens for *NavigationMessage* and *NavigationPropsMessage*
* Preforms Stack Navigation
* Provides method to send RunEngine Messages process on componentWillMount

|  |
| --- |
| **constructor(props: Props = { navigation: null }) {  super(props);  this.receive = this.receive.bind(this);  runEngine.attachBuildingBlock(this as IBlock, [  getName(MessageEnum.NavigationMessage),  getName(MessageEnum.NavigationPropsMessage)  ]);  }   async receive(from: string, message: Message) {  }** |

##### Adapters

* Bridge between Blocks
* Listens for messages of Type A and Converts to Message Type B

Example:

|  |
| --- |
| enum MessageEnum {  NavigationEmailLogInMessage, } |

|  |
| --- |
| import { v4 as uuidv4 } from 'uuid'; import { runEngine } from '../../../framework/src/RunEngine'; import { IBlock } from '../../../framework/src/IBlock'; import { Message } from '../../../framework/src/Message'; import MessageEnum, {  getName } from '../../../framework/src/Messages/MessageEnum';  export default class EmailAccountLogInAdapter {  send: (message: Message) => void;   constructor() {  const blockId = uuidv4();  this.send = message => runEngine.sendMessage(blockId, message);  runEngine.attachBuildingBlock(this as IBlock, [  getName(MessageEnum.NavigationEmailLogInMessage)  ]);  }   convert = (from: Message): Message => {  const to = new Message(getName(MessageEnum.NavigationMessage));   to.addData(  getName(MessageEnum.NavigationTargetMessage),  'EmailAccountLogin'  );   to.addData(  getName(MessageEnum.NavigationPropsMessage),  from.getData(getName(MessageEnum.NavigationPropsMessage))  );   return to;  };   receive(from: string, message: Message): void {  this.send(this.convert(message));  } } |

|  |
| --- |
| **import EmailAccountLogInAdapter from '../../blocks/adapters/src/EmailAccountLogInAdapter';** |

|  |
| --- |
| **const emailAccountLogInAdapter = new EmailAccountLogInAdapter();** |

|  |
| --- |
| **goToEmailLogin() {  const msg: Message = new Message(getName(MessageEnum.NavigationEmailLogInMessage));  msg.addData(  getName(MessageEnum.NavigationPropsMessage),  this.props  )  this.send(msg)  }** |

### 

###### Example Basic Adapter

|  |
| --- |
| import { v4 as uuidv4 } from 'uuid'; import { runEngine } from '../../../framework/src/RunEngine'; import { IBlock } from '../../../framework/src/IBlock'; import { Message } from '../../../framework/src/Message'; import MessageEnum, {  getName } from '../../../framework/src/Messages/MessageEnum';  export default class HomeScreenAdapter {  send: (message: Message) => void;   constructor() {  const blockId = uuidv4();  this.send = message => runEngine.sendMessage(blockId, message);  runEngine.attachBuildingBlock(this as IBlock, [  getName(MessageEnum.NavigationHomeScreenMessage)  ]);  }   convert = (from: Message): Message => {  const to = new Message(getName(MessageEnum.NavigationMessage));  to.addData(getName(MessageEnum.NavigationTargetMessage), 'Home');  to.addData(  getName(MessageEnum.NavigationPropsMessage),  from.getData(getName(MessageEnum.NavigationPropsMessage))  );   return to;  };   receive(from: string, message: Message): void {  this.send(this.convert(message));  } } |

###### Example Adapter w/ Data Passed w/ Navigation

|  |
| --- |
| import { v4 as uuidv4 } from 'uuid'; import { runEngine } from '../../../framework/src/RunEngine'; import { IBlock } from '../../../framework/src/IBlock'; import { Message } from '../../../framework/src/Message'; import MessageEnum, {  getName } from '../../../framework/src/Messages/MessageEnum';  export default class InfoPageAdapter {  send: (message: Message) => void;   constructor() {  const blockId = uuidv4();  this.send = message => runEngine.sendMessage(blockId, message);  runEngine.attachBuildingBlock(this as IBlock, [  getName(MessageEnum.NavigationInfoPageMessage),  getName(MessageEnum.AccoutResgistrationSuccess),  getName(MessageEnum.AccoutLoginSuccess)  ]);  }   convert = (from: Message): Message => {  let title = from.getData(getName(MessageEnum.InfoPageTitleMessage));  let body = from.getData(getName(MessageEnum.InfoPageBodyMessage));  let buttonText = from.getData(  getName(MessageEnum.InfoPageButtonTextMessage)  );  let navigationBarTitle = from.getData(  getName(MessageEnum.NavigationScreenNameMessage)  );  let buttonNavigationMessage = from.getData(  getName(MessageEnum.NavigationRaiseMessage)  );   if (from.id === getName(MessageEnum.AccoutResgistrationSuccess)) {  title = 'Account Creation';  body = 'Account was successfully created.';  buttonText = 'Ok';  navigationBarTitle = 'Account Creation';  buttonNavigationMessage = new Message(  getName(MessageEnum.NavigationHomeScreenMessage)  );  } else if (from.id === getName(MessageEnum.AccoutLoginSuccess)) {  title = 'Account Login';  body = 'Account was successfully logged in.';  buttonText = 'Ok';  navigationBarTitle = 'Account Login';  buttonNavigationMessage = new Message(  getName(MessageEnum.NavigationHomeScreenMessage)  );  }   const to = new Message(getName(MessageEnum.NavigationMessage));  to.addData(getName(MessageEnum.NavigationTargetMessage), 'InfoPage');   to.addData(  getName(MessageEnum.NavigationPropsMessage),  from.getData(getName(MessageEnum.NavigationPropsMessage))  );   to.addData(  getName(MessageEnum.NavigationScreenNameMessage),  navigationBarTitle  );   const raiseMessage: Message = new Message(  getName(MessageEnum.NavigationPayLoadMessage)  );  raiseMessage.addData(getName(MessageEnum.InfoPageTitleMessage), title);  raiseMessage.addData(getName(MessageEnum.InfoPageBodyMessage), body);  raiseMessage.addData(  getName(MessageEnum.InfoPageButtonTextMessage),  buttonText  );   buttonNavigationMessage.addData(  getName(MessageEnum.NavigationPropsMessage),  from.getData(getName(MessageEnum.NavigationPropsMessage))  );   raiseMessage.addData(  getName(MessageEnum.InfoPageNavigationScreenMessage),  buttonNavigationMessage  );   to.addData(getName(MessageEnum.NavigationRaiseMessage), raiseMessage);   return to;  };   receive(from: string, message: Message): void {  this.send(this.convert(message));  } } |

##### Component vs Block

|  |
| --- |
| import React, { Component } from "react"; import { StyleSheet, View } from "react-native"; import { SearchBar } from "react-native-elements";  interface Props {  containerStyle: any;  placeholder: any;  lightTheme: boolean;  round: boolean;  showLoading: boolean;  onChangeText: any;  autoCorrect: any;  autoFocus: any;  value: any; }  interface S {  containerStyle: any;  placeholder: any;  lightTheme: boolean;  round: boolean;  showLoading: boolean;  onChangeText: any;  autoCorrect: any;  autoFocus: any;  value: any; } export default class GenericSearch extends Component <Props, S> {  constructor(props: Props) {  super(props);  this.state = {  containerStyle: props.containerStyle,  placeholder: props.placeholder,  lightTheme: props.lightTheme,  round: props.round,  showLoading: props.showLoading,  onChangeText: props.onChangeText,  autoCorrect: props.autoCorrect,  autoFocus: props.autoFocus,  value: props.value  };  }  render() {  return(  <View style = {styles.container}>  <SearchBar  containerStyle={this.props.containerStyle}  placeholder={this.props.placeholder}  lightTheme={this.props.lightTheme}  round={this.props.round}  onChangeText={text => this.props.onChangeText(text) }  autoCorrect={this.props.autoCorrect}  autoFocus={this.props.autoFocus}  value={this.props.value}  showLoading={this.props.showLoading}  />  </View>  )  } } |

|  |
| --- |
| //@ts-ignore import GenericSearch from "../../../../components/src/GenericSearch" |

|  |
| --- |
| renderHeaderSearchBarWithCancelButton = () => {  return (  <View style={styles.searchBarViewStyle}>  <**GenericSearch**  containerStyle={styles.searchBarStyleWithCancelButton}  placeholder="Search a place"  lightTheme  round  onChangeText={(text: string) => this.makeGooglePlaceRequest(text)}  autoCorrect={false}  autoFocus={true}  value={this.state.searchBarTextValue}  showLoading={false}  />  <**TouchableOpacity** style={styles.cancelButtonStyle} onPress={() => this.cancelButtonTapped(this.state.isGenericSearchActive)} >  <**Text** style={styles.cancelLabelStyle}>Cancel</**Text**>  </**TouchableOpacity**>  </**View**>  );  }; |

#### Mobile Route Map

See EXAMPLE adding Component to Route Maps

|  |
| --- |
| // App.tsx - MOBILE  import React from 'react';  import {  createStackNavigator } from "react-navigation";  import HomeScreen from "../components/src/HomeScreen"; import InfoPage from '../blocks/info-page/src/InfoPageBlock'  ***import COMPONET\_EXPORTS from 'COMPONET\_PATH'***  const HomeStack = createStackNavigator({  Home: { screen: HomeScreen, navigationOptions: { header: null, title: "Home" } },  InfoPage: { screen: InfoPage, navigationOptions: { title: "Info" } },  ***RouteName: { screen: COMPONET\_EXPORT, navigationOptions: { title: SCREEN\_TITLE }***  });   export function App() {  return (  <**HomeStack** />  ); }; |

##### Web Route Map and webpack

See EXAMPLE adding Component to Route Maps

|  |
| --- |
| // App.js - WEB import React, { Component } from "react"; import { View } from "react-native"; import firebase from 'firebase' import { connect } from 'react-firebase'  import WebRoutesGenerator from "../../components/src/NativeWebRouteWrapper"; import { ModalContainer } from "react-router-modal"; import HomeScreen from "../../components/src/HomeScreen"; import TopNav from "../../components/src/TopNav";  import InfoPage from '../../blocks/info-page/src/InfoPageBlock' import AlertBlock from '../../blocks/alert/src/AlertBlock.web'  ***import COMPONET\_EXPORTS from 'COMPONET\_PATH***  const routeMap = {  Home: {  component: HomeScreen,  path: '/',  exact: true  },  InfoPage: {  component: InfoPage,  path: '/InfoPage'  },   AlertWeb: {  component: AlertBlock,  path: "\*/AlertWeb",  modal: true  },  AlertWeb: {  component: AlertBlock,  path: "\*/AlertWeb",  modal: true  },  ***RouteName: {***  *component****: COMPONET\_EXPORT,***  *path****: “URL”***  ***}***  };  const firebaseAPI = firebase.initializeApp({  apiKey: "AIzaSyDgl9aTbKMdRZ9-ijSZRionh3V591gMJl4",  authDomain: "rnmasterapp-c11e9.firebaseapp.com",  databaseURL: "https://rnmasterapp-c11e9.firebaseio.com",  projectId: "rnmasterapp-c11e9",  storageBucket: "rnmasterapp-c11e9.appspot.com",  messagingSenderId: "649592030497",  appId: "1:649592030497:web:7728bee3f2baef208daa60",  measurementId: "G-FYBCF3Z2W3" });  class App extends Component {    render() {   const defaultAnalytics = firebaseAPI.analytics();  defaultAnalytics.logEvent('APP\_Loaded');    return (  <**View** style={{ height: '100vh', width: '100vw' }}>  <**TopNav** />  {WebRoutesGenerator({ routeMap })}  <**ModalContainer** />  </**View**>  );  } }  export default App; |

##### See EXAMPLE adding Component to webpack

|  |
| --- |
| // App.js - WEB  const fs = require('fs') const path = require('path') const webpack = require('webpack')  const appDirectory = fs.realpathSync(process.cwd()) // config-overrides.js - WEB  const resolveApp = relativePath => path.resolve(appDirectory, relativePath)  //MARK::Add Web Blocks const appIncludes = [  resolveApp('src'),  resolveApp('../components/src'),  resolveApp('../framework/src'),  resolveApp('../../node\_modules/react-native-elements'),  resolveApp('../../node\_modules/react-native-vector-icons'),  resolveApp('../../node\_modules/react-native-ratings'),  resolveApp('../../node\_modules/react-native-image-picker'),  resolveApp('../../node\_modules/react-native-check-box'),  resolveApp('../blocks/restClient/src'),  resolveApp('../blocks/alert/src'),  resolveApp('../blocks/adapters/src'),  resolveApp('../blocks/info-page/src'),  ***resolveApp('COMPONET\_PATH')***  ]  module.exports = function override(config, env) {  // allow importing from outside of src folder  config.resolve.plugins = config.resolve.plugins.filter(  plugin => plugin.constructor.name !== 'ModuleScopePlugin'  )  config.module.rules[0].include = appIncludes  config.module.rules[1] = null  config.module.rules[2].oneOf[1].include = appIncludes  config.module.rules[2].oneOf[1].options.plugins = [  require.resolve('babel-plugin-react-native-web'),  ].concat(config.module.rules[2].oneOf[1].options.plugins)  config.module.rules = config.module.rules.filter(Boolean)  config.plugins.push(  new webpack.DefinePlugin({ \_\_DEV\_\_: env !== 'production' })  )  return config } |

### Using existing packages

#### React native web

<https://github.com/necolas/react-native-web>

#### Unit and UI Testing

<https://www.npmjs.com/package/jest-cucumber>

<https://jestjs.io/docs/en/tutorial-react>

#### React-native elements

React-native elements is a lightweight UI-framework library for react-native which is also compatible with react native web. It provides equivalent substitutes for many react-native components such as Buttons, Images, Inputs, and more. The full list of components can be viewed [here](https://react-native-elements.github.io/react-native-elements/docs/overview.html).

Here is an example of how to use a react-native-element button.

import { Button } from 'react-native-elements';

<Button

title="Outline button"

type="outline"

/>

#### Material UI

Material UI is another UI framework library. It is based on material design which was developed by Google. It’s a great tool to create consistent appearances across web and mobile applications. It comes with more advanced controls, such as DatePicker, AutoComplete, Pagination, and more. The full documentation is available [here](https://material-ui.com/).

import { Rating } from '@material-ui/lab';

<Rating name="half-rating" defaultValue={2.5} precision={0.5} />

#### React-native-vector-icons

React-native-vector-icons provides vector icons from popular css libraries such as font-awesome, foundation, material icons, and more. The advantage of using vector icons instead of images is that they are much smaller in size, have adjustable color, and can be scaled to any size without loss of quality. Here is an example of using the search icon from font awesome.

import Icon from "react-native-vector-icons/FontAwesome";

<Icon name="search" size={12} color="white"></Icon>

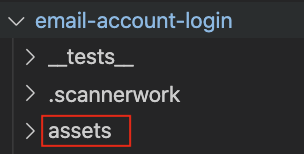
To use a different icon, you just replace the icon name with the icon you want. I use the following process to find the icon I want. I’m using font-awesome as an example but you use the same approach for any other css library.

1. Search for the icon I want on [font awesome’s](https://fontawesome.com/icons?d=gallery) home page.
2. Write down or take note of the icon name.
3. Verify that the same icon name is contained in [this](https://github.com/oblador/react-native-vector-icons/blob/master/glyphmaps/FontAwesome.json) file.

### Coding Best Practices

#### Using images and other assets

When using a local image, make sure the image is placed in the asset folder of the current feature / block.



Next, you want to import the image like this:

const imgPasswordVisible = require("./assets/ic\_password\_visible.png");

And here’s how to use the imported image

<Image source={imgPasswordVisible}/>

#### Creating a custom component vs using existing one

To avoid reinventing the wheel, and to keep our styles consistent, it’s best to first check if a component already exists before creating a new one. You may find what you need as either a block or a component.

If you don’t find what you need in the project, the next step is to look at our UI frameworks for a compatible component. For web, check out [react-native-elements](https://react-native-elements.github.io/react-native-elements/docs/overview.html) and [material-ui](https://material-ui.com/getting-started/installation/), and for mobile, check out [react-native](https://reactnative.dev/docs/components-and-apis#basic-components) components.

When you create the component, try to make it as generic as possible so that it can be used by other developers in the future. An example would be to pass in properties and event handlers via props instead of hard-coding them within the custom component.

#### Putting string into config files

When creating blocks, it’s a good idea to retrieve variable names, buttons, from config files instead of hard-coding them.

#### Using Stylesheet.Create vs external .css file

When building in React native components, the convention is to create a styles object using Stylesheet.create() and apply the style by passing it in through the style prop. Here is an example:

import { View, StyleSheet } from "react-native";

const styles = StyleSheet.create({

booking\_container: {

position: "absolute",

backgroundColor: "white",

width: "auto",

bottom: 24,

top: 10,

left: 10,

zIndex: 10

},

<View style={styles.booking\_container}>

If you try to use an html component such as a div with the stylesheet object, you will get errors. For example, the following code won’t work:

<div style={styles.booking\_container}></div>

You have two options:

1. Use an inline style such like this:

<div

style={{

position: "absolute",

}}

></div>

1. Import an external stylesheet and apply className attribute and use it as you would a regular css class.

import "./assets/SearchPanel.css";

<div className="booking\_container"></div>

#### Styling blocks - Using flexbox

Because blocks need to work on a variety of different screen sizes and platforms, the layout needs to be flexible and responsive. CSS Flexbox, introduced in 2008 still remains one of the best tools for solving this type of challenge.

In a nutshell, flexbox gives you unprecedented control in centering elements, aligning items, without writing complicated css, and is nearly universally supported in all modern browsers.

I will not go into detail about using flexbox in this guide but I will provide the following resources.

<https://cvan.io/flexboxin5/> is a sandbox where you can arrange elements as you would like to have them on your page and then copy the generated css to your block.

If you are using React-Native style css, you can coy and paste your code to this site to convert to the JSX version: <https://staxmanade.com/CssToReact/>

#### Eslint

Linting is the process of running a program that analyzes your code for programmatic and stylistic errors. A linting tool, or a linter, marks or flags any potential errors in your code such as syntax errors or incorrectly spelled variable names. This can save time and help you write better code. Eslint is one implementation of eslint and it’s one of the tools we use to ensure code quality.

You may run **yarn lint** at any point to see the lint errors and most should be simple to fix. However, there may be some cases where there is a good reason to keep the code with the lint error is actually preferred. In this case, you may tell eslint to ignore the error by adding this one-liner just above the offending code.

For example:

// eslint-disable-next-line react-native/no-inline-styles

The first part - **eslint-disable-next-line** tells eslint to ignore the next line and the second part - **react-native/no-inline-styles** is the rule that was broken. You can find the name of the broken rule in the terminal with the other eslint error messages.

#### Typescript

Typescript is another tool we use to help ensure that we are writing code that’s well-formed. Just like Eslint, you will get errors if you write code that violates the typescript’s rules. It is strongly suggested that you heed these errors. However, there may be a few situations where it’s impossible to satisfy Typescript’s requirements, such as when a third party package doesn’t have the typescript definition files.

In this case, it is permissible to override the error by adding //@ts-ignore like this:

//@ts-ignore

import Autocomplete from "react-google-autocomplete";

#### Prettier

Prettier is an opinionated JavaScript formatter inspired by refmt with advanced support for language features from ES2017, JSX, and Flow. It removes all original styling and ensures that all outputted JavaScript conforms to a consistent style. You may read more here: <https://github.com/madskristensen/JavaScriptPrettier>

Just as as important as writing code that’s syntactically correct, is the need to write code that’s stylistically consistent. Prettier helps us do this by automatically formatting your code each time you commit. The project you will be working with will already have Prettier configured. There is nothing extra you need to do but you may notice that your code structure may change after you commit.

#### Javascript / Typescript general standards

<https://docs.google.com/document/d/1PiZu0iulehaWr6uZUotnkLcI9q4SeFL2wBU3pF9-L6s/edit>