react-native-router-flux 使用详解（三）

作者：angels
原文链接：https://ld246.com/article/1496282665355
来源网站：链滴

在 上一章 http://cherylgood.cn/c/reac_native_router_flux_使用详解二
我们主要进一步介绍了 react-native-router-flux 的使用，接下来主要讲解 其主要配置参数和 ap，当前我主要是翻译官网的学习资料进行学习，我将在后面的章节中实际使用他，通关编写一个 rn 的微博 app React-Native 学习之制作 RN 版的微博 app
<h2 id="Available-imports">Available imports</h2>

<code>Router </code>
<code>Scene </code>
<code>Modal</code>
<code>TabBar</code>
<code>getInitialState</code>
<code>Reducer</code>
<code>DefaultRenderer</code>
<code>Switch</code>
<code>Actions</code>
<code>ActionConst</code>
<code>NavBar</code>

<h2 id="Router-">Router:</h2>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reducer</td>
<td><code>function</code></td>
<td></td>
<td>optional user-defined reducer for scenes, you may want to use it to intercept all actions nd put your custom logic 你可以为 scenes 定义 reducer，你可以通过使用 reducer 拦截所有的 actions 并执行你自定义的逻辑代码。</td>
</tr>
<tr>
<td>createReducer</td>
<td><code>function</code></td>
<td></td>
<td>function that returns a reducer function for {initialState, scenes} param, you may wrap Rducer(param) with your custom reducer, check Flux usage section below 该函数功能：createRducer({initialState, scenes})将返回一个 reducer，你可以用你自定义的 reduer 棒状一个 Reducer(pram)，可以参看下面章节中 Flux 的用例。</td>
</tr>
<tr>
<td>other props</td>
<td></td>
<td></td>
<td>all properties that will be passed to all your scenes 在 Router 中定义的所有属性都会传入的 scenes 组件中</td>

</tr>
<tr>
<td>children</td>
<td></td>
<td>required (if no scenes property passed)当没有 children 属性被传入时，router 必须有子节
</td>
<td>Scene root element 通过 children 属性定义我们的 root 元素</td>
</tr>
<tr>
<td>scenes</td>
<td><code>object</code></td>
<td>optional 可选</td>
<td>scenes for Router created with Actions.create. This will allow to create all actions BEFORE React processing. If you don't need it you may pass Scene root element as children 因为 scens 时 Router 被创建后通过 Actions.create 创建的。这将允许我们在 React 创建 scenes 之前创建好有的 actions。如果你不需要你可以使用 Scene 跟元素作为子节点</td>
</tr>
<tr>
<td>getSceneStyle</td>
<td><code>function</code></td>
<td>optional 可选</td>
<td>Optionally override the styles for NavigationCard's Animated.View rendering the scene. 据需要重写该样式去改变导航卡的动画。</td>
</tr>
<tr>
<td>backAndroidHandler</td>
<td><code>function</code></td>
<td>optional 可选</td>
<td>Optionally override the handler for <code>BackAndroid</code>, return <code>true</ode> to stay in the app or return <code>false</code> to exit the app. Default handler will p p a scene and exit the app at last when the back key is pressed on Android.可以重写该方法去制 android 设备的返回键，返回 true 时会停留在 app 内部，返回 false 时会直接退出 app，默认的作时重栈中出栈栈顶的 scene，如果该 scene 是最后一个，就会退出 app。（相信 android 程序员很熟悉）</td>
</tr>
<tr>
<td>onBackAndroid</td>
<td><code>function</code></td>
<td>optional 可选</td>
<td>Get called after back key is pressed and a scene is poped, won't affect the default behavor.在返回键被按下且 scene 弹出后将被调用，不会影响到默认的行为。可以通过该方法做一些弹出的操作。</td>
</tr>
<tr>
<td>onExitApp</td>
<td><code>function</code></td>
<td>optional</td>
<td>Optionally override the default action after back key is pressed on root scene. Return <cde>true</code> to stay, or return <code>false</code> to exit the app.可以重写该方法去定当处于 root scene 时，返回键被按下后的行为，返回 false 会退出该 app</td>
</tr>
</tbody>
</table>
<h2 id="Scene-"><a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.com%

Faksonov%2Freact-native-router-flux%2Fblob%2Fmaster%2Fdocs%2FAPI_CONFIGURATION.d%23scene" target="_blank" rel="nofollow ugc">Scene:</h2>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td><code>string</code></td>
<td>required 必须</td>
<td>Will be used to call screen transition, for example, <code>Actions.name(params)</code>. Must be unique.在切换屏幕的时候会使用该 key，例如 Actions.name(params).key 的值必须时一的。</td>
</tr>
<tr>
<td>component</td>
<td><code>React.Component</code></td>
<td>semi-required</td>
<td>The <code>Component</code> to be displayed. Not required when defining a nested code>Scene</code>, see example. If it is defined for 'container' scene, it will be used as custm container <code>renderer</code> 切换到该 scene 时，component 属性定义的组件将被展出来。当定义一个嵌套 scene 时不要求有。例如。如果他作为一个 scene 容器定义。他将被视作一自定义容器渲染者来使用。</td>
</tr>
<tr>
<td>initial</td>
<td><code>bool</code></td>
<td>false</td>
<td>Set to <code>true</code> if this is the initial scene 如果设置该属性为 true，该 scene 将为默认初始化 scene。你可以理解为进来后进一个看到的 scene</td>
</tr>
<tr>
<td>type</td>
<td><code>string</code></td>
<td><code>ActionConst.PUSH</code> or <code>ActionConst.JUMP</code></td>
<td>Defines how the new screen is added to the navigator stack. One of <code>ActionConstPUSH</code>, <code>ActionConst.JUMP</code>, <code>ActionConst.REPLACE</code>, <ode>ActionConst.RESET</code>. If parent container is tabbar (tabs=true), <code>ActionCont.JUMP</code> will be automatically set.定义如何去创建一个新的屏幕并放入导航栈中。可以是 AtionConst.PUSH,AtionConst.JUMP,ActionConst.REPLACK,AtionConst.RESET.如果父容器是一个 tbbar 且 tabs=true，将会自动设置为 ActionConst.JUMP.</td>
</tr>
<tr>
<td>clone</td>
<td><code>bool</code></td>
<td></td>
<td>Scenes marked with <code>clone</code> will be treated as templates and cloned into he current scene's parent when pushed. See example.在被 push 的时候，使用 clone 标识的 Sce

es 将被作为模版处理，并克隆到当前的 scene 的容器中。</td>
</tr>
<tr>
<td>passProps</td>
<td><code>bool</code></td>
<td>false</td>
<td>Pass all own props (except style, key, name, component, tabs) to children. Note that pas
Props is also passed to children.将自己所有的属性（except style，key，name，component，ta
s）传入到子节点。</td>
</tr>
</tbody>
</table>
<h2 id="ActionConst-"><a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.
om%2Faksonov%2Freact-native-router-flux%2Fblob%2Fmaster%2Fdocs%2FAPI_CONFIGURA
ION.md%23actionconst" target="_blank" rel="nofollow ugc">ActionConst:</h2>
We accept shorthand string literal when defining scene type or action params, like:
在定义 scene 类型活着 action 参数时，我们接受间断的字符串文字，例如：
<pre><code class="language-source-js highlight-chroma"><s
an class="highlight-cl">Actions.ROUTE_NAME({type: 'reset'});
<Scene key="m
scene" type="replace" >
</code></pre>
but it will be converted to const value when pass to reducer. RECOMMENDATION is to a
ways use const instead of string literal for consistency:
但是当传入 reducer 时，它将被转换成一个静态值，为来一致性，我们推荐总是使用静态的去替
字符串文字。
<pre><code class="language-source-js highlight-chroma"><s
an class="highlight-cl">Actions.ROUTE_NAME({type: ActionConst.RESET});
<Scene key="m
scene" type={ActionConst.REPLACE} >
</code></pre>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Shorthand</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionConst.JUMP</td>
<td><code>string</code></td>
<td>'REACT_NATIVE_ROUTER_FLUX_JUMP'</td>
<td>'jump'</td>
</tr>
<tr>
<td>ActionConst.PUSH</td>
<td><code>string</code></td>
<td>'REACT_NATIVE_ROUTER_FLUX_PUSH'</td>
<td>'push'</td>
</tr>
<tr>

<td>ActionConst.REPLACE</td>
<td><code>string</code></td>
<td>'REACT_NATIVE_ROUTER_FLUX_REPLACE'</td>
<td>'replace'</td>
</tr>
<tr>
<td>ActionConst.BACK</td>
<td><code>string</code></td>
<td>'REACT_NATIVE_ROUTER_FLUX_BACK'</td>
<td>'back'</td>
</tr>
<tr>
<td>ActionConst.BACK_ACTION</td>
<td><code>string</code></td>
<td>'REACT_NATIVE_ROUTER_FLUX_BACK_ACTION'</td>
<td>'BackAction'</td>
</tr>
<tr>
<td>ActionConst.POP_AND_REPLACE</td>
<td><code>string</code></td>
<td>'REACT_NATIVE_ROUTER_FLUX_POP_AND_REPLACE'</td>
<td>'popAndReplace'</td>
</tr>
<tr>
<td>ActionConst.POP_TO</td>
<td><code>string</code></td>
<td>'REACT_NATIVE_ROUTER_FLUX_POP_TO'</td>
<td>'popTo'</td>
</tr>
<tr>
<td>ActionConst.REFRESH</td>
<td><code>string</code></td>
<td>'REACT_NATIVE_ROUTER_FLUX_REFRESH'</td>
<td>'refresh'</td>
</tr>
<tr>
<td>ActionConst.RESET</td>
<td><code>string</code></td>
<td>'REACT_NATIVE_ROUTER_FLUX_RESET'</td>
<td>'reset'</td>
</tr>
<tr>
<td>ActionConst.FOCUS</td>
<td><code>string</code></td>
<td>'REACT_NATIVE_ROUTER_FLUX_FOCUS'</td>
<td>'focus'</td>
</tr>
</tbody>
</table>
<h3 id="ActionConst-and-Scene-type-explaination">ActionConst and Scene.type explaination</h3>
ActionConst

are just a bunch of constants represent real values of various actions/scene.type to avoid uture changes. you can treat it like redux action.
ActionConst 可以理解为就是一堆常量去表示真实的各种各样的 actions/scene.type 的值，这做可以避免后期的变化。你可以像 redux action 一样处理它。
These can be used directly, for example, Actions.pop() will dispatch correspond action wri ten in the source code, or, you can set those constants in scene type, when you do Actions.ma n(), it will dispatch action according to your scene type or the default one.
ActionConst 也可以直接使用，例如：Action.pop()将分派给符合的 action（你在代码中编写的 ction）,或者，你可以在一个 scene type 中设置她们的常量。当你执行 Action.main()时，它将根据 的 scene type 或者默认的一个去转发给合适的 action。
Not every ActionConst can be used the same way (use as an action or whether it can be et in scene type or not) that's why it's just a bunch of constants to mask the actual values.</p

不是每个 ActionConst 都可以使用相同的方式（作为一个动作使用，或者它是否可以在场景类设置或不能），这就是为什么它只是一堆常量来掩盖实际的值。（我的理解是在转换为常量值的时候会根据你定义的方式做分类，以此为依据进行处理，后续会阅读代码以确认 ~ ~ ）
Scene.type
Defines how the new screen is added to the navigator stack. One of push, modal, actionS eet, replace, switch, reset transitionToTop. Default is 'push'. And every <code>Scene.type</c de> string literal has a mapped constant in ActionConst, it is recommended to always use co stant.
定义如何去增加一个新的屏幕到导航栈中。可以是 push,modal,actionSheet，replace,switch,re eet transitionToTop 中的一个（相信前三个 ios 程序员将不会陌生，因为我现在主要是作为 android 程序员，但是也搞过 ios 开发，所有都懂那么一点点，理解起来好像挺有帮助的 ~ ~ ）.默认的是 pus 。每一个 Scene.type 都会在 ActionConst 中存在一个对应的常量，我们推荐总是使用常量来表示。 /p>
<code>replace</code>: tells navigator to replace current route with new route.
replace:告诉导航器使用一个新的 route 来替换当前的 route。
<code>actionSheet</code>: shows Action Sheet popup, you must pass callback as callb ck function.
actionSheet:以弹出的方式展示一个 Action Sheet，你必须传入一个回调作为回调方法。
<code>modal</code>: type inserts its 'component' into route stack after navigator com onent. It could be used for popup alerts as well for various needed processes before any navi ator transitions (like login auth process). it could be dismissed by using Actions.dismiss().
modal:在导航组件后的路由栈中插入该类型定义的组件。它可以被作为一个弹出的提示框使用 也可以在任何导航传输之前（像登录授权处理）做一些必须处理的操作进行使用。我们可以使用 Acti ns.dismiss()关闭它。（类似 android 原生种的 dialog，ios 中的模态视图）。
<code>switch</code>: is used for tab screens.
switch：在 tab 场景下使用。（一般是点击底部按钮切换不同的 scene）。
<code>reset</code>: is similar to <code>replace</code> except it unmounts the com onets in the navigator stack.
reset：类似 replace，但是它在导航栈中卸载了该组件。
<code>transitionToTop</code>: will reset router stack ['route.name'] and with animation if route has sceneConfig. eg <code><Route name="login" schema="modal" component={ ogin} type="transitionToTop" /></code>
transitionToTop:如果路由有 sceneConfig 配置，如： <code><Route name="login" sch ma="modal" component={Login} type="transitionToTop"/>，将根据name重置路由堆栈中 路由和动画。</code>
<h3 id="Animation">Animation</h3>
<table>
<thead>
<tr>

<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td><code>number</code></td>
<td></td>
<td>optional. acts as a shortcut to writing an <code>applyAnimation</code> function with code>Animated.timing</code> for a given duration (in ms).可选的。 充当在给定持续时间（以 ms 为单位）中使用 Animated.timing 编写 applyAnimation 函数的快捷方式。 </td>
</tr>
<tr>
<td>direction</td>
<td><code>string</code></td>
<td>'horizontal'</td>
<td>direction of animation horizontal/vertical/leftToRight ('horizontal' will be right to left)动的方向 水平 / 垂直 / 左到右 （水平即从右到左） </td>
</tr>
<tr>
<td>animation</td>
<td><code>string</code></td>
<td></td>
<td>animation options when transitioning: 'fade' currently only option 在转换时的动画选项，前只有 fade（渐变） </td>
</tr>
<tr>
<td>animationStyle</td>
<td><code>function</code></td>
<td></td>
<td>optional interpolation function for scene transitions: <code>animationStyle={interpolatinFunction}``用于场景转换的可选内插函数：animationStyle = {interpolationFunction}</code></d>
</tr>
<tr>
<td>applyAnimation</td>
<td><code>function</code></td>
<td></td>
<td>optional if provided overrides the default spring animation 可选，如果提供将覆盖默认的簧动画</td>
</tr>
</tbody>
</table>
<h3 id="Gestures-手势">Gestures 手势</h3>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>

<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>panHandlers</td>
<td><code>object</code></td>
<td></td>
<td>optional, provide null to disable swipe back gesture 可选的，置为 null 可以关闭滑动返回势。</td>
</tr>
<tr>
<td>getPanHandlers</td>
<td><code>function</code></td>
<td>optional</td>
<td>Optionally override the gesture handlers for scene 可选的去重写一个 scene 的手势操作</t
>
</tr>
</tbody>
</table>
<h3 id="Scene-styles-场景类型表"><a href="https://ld246.com/forward?goto=https%3A%2F
2Fgithub.com%2Faksonov%2Freact-native-router-flux%2Fblob%2Fmaster%2Fdocs%2FAPI_C
NFIGURATION.md%23scene-styles" target="_blank" rel="nofollow ugc">Scene styles
景类型表</h3>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sceneStyle</td>
<td><a href="https://ld246.com/forward?goto=https%3A%2F%2Ffacebook.github.io%2Freac
-native%2Fdocs%2Fview.html%23style" target="_blank" rel="nofollow ugc"><code>Viewstyl
</code></td>
<td>{ flex: 1 }</td>
<td>optional style override for the Scene's component 场景组件的可选样式覆盖</td>
</tr>
<tr>
<td>getSceneStyle</td>
<td><code>function</code></td>
<td>optional</td>
<td>Optionally override the styles for NavigationCard's Animated.View rendering the scene.
eceives first argument of <code>NavigationSceneRendererProps</code> and second argum
nt of <code>{hideNavBar,hideTabBar,isActive}</code> (see Example app).可以覆盖 Navigatio
Card 的 Animated.View 渲染场景的样式。 接收 NavigationSceneRendererProps 的第一个参数和{
ideNavBar，hideTabBar，isActive}的第二个参数。</td>
</tr>
</tbody>

</table>
<h3 id="Tabs">Tabs</h3>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tabs</td>
<td><code>bool</code></td>
<td>false</td>
<td>Defines 'TabBar' scene container, so child scenes will be displayed as 'tabs'. If no <code component</code> is defined, built-in <code>TabBar</code> is used as renderer. All child scenes are wrapped into own navbar.定义 TabBar 场景容器以便子场景可以作为 tabs 展示。如果没组件被定义，内置的 TabBar 将作为容器。所有的子场景都被包裹到自己的导航条中。</td>
</tr>
<tr>
<td>tabBarStyle</td>
<td><code>Viewstyl</code></td>
<td></td>
<td>optional style override for the Tabs component 可以选择重写去定义 Tabs 组件的样式</td

</tr>
<tr>
<td>tabBarBackgroundImage</td>
<td><code>Imaesource</code></td>
<td></td>
<td>optional background image for the Tabs component 可以选择重写去定义 Tabs 组件的背图片</td>
</tr>
<tr>
<td>tabBarIconContainerStyle</td>
<td><code>Viewstyl</code></td>
<td></td>
<td>optional style override for the View that contains each tab icon 可以选择重写去定义包含个 tab icon 的 vie 容器的样式</td>
</tr>
<tr>
<td>hideTabBar</td>
<td><code>bool</code></td>
<td>false</td>

<td>hides tab bar for this scene and any following scenes until explicitly reversed (if built-in abBar component is used as parent renderer)隐藏此场景的选项卡栏和任何后续场景，直到显式转（如果内置 TabBar 组件用作父渲染器）</td>
</tr>
<tr>
<td>hideOnChildTabs</td>
<td><code>bool</code></td>
<td>false</td>
<td>hides tab bar when another <code>tabs</code> scene is added to the navigation stack 当另一个选项卡场景添加到导航堆栈时，隐藏被添加到当行栏的场景的选项卡栏。</td>
</tr>
<tr>
<td>pressOpacity</td>
<td><code>number</code></td>
<td>0.2</td>
<td>the opacity when clicking on the tab 点击选项卡时的透明度，默认 0.2</td>
</tr>
</tbody>
</table>
<h3 id="Navigation-Bar">Navigation Bar</h3

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hideNavBar</td>
<td><code>bool</code></td>
<td>false</td>
<td>hides the navigation bar for this scene and any following scenes until explicitly reversed 隐藏当前 scene 的导航栏和后续 scene 直到明确的反转该值。</td>
</tr>
<tr>
<td>navigationBarStyle</td>
<td><code>View style</code></td>
<td></td>
<td>optional style override for the navigation bar 可以重写该属性去定义导航栏</td>
</tr>
<tr>
<td>navigationBarBackgroundImage</td>
<td><code>Image source</code></td>
<td></td>

<td>optional background image for the navigation bar 重写该属性去设置导航栏的背景图片</t
>
</tr>
<tr>
<td>navBar</td>
<td><code>React.Component</code></td>
<td></td>
<td>optional custom NavBar for the scene. Check built-in NavBar of the component for refer
nce 通过该属性设置自定义的导航栏。可以参考内置的导航栏组件</td>
</tr>
<tr>
<td>drawerImage</td>
<td><a href="https://ld246.com/forward?goto=https%3A%2F%2Ffacebook.github.io%2Freac
-native%2Fdocs%2Fimage.html%23source" target="_blank" rel="nofollow ugc"><code>Ima
e source</code></td>
<td><code>require('./menu_burger.png')</code></td>
<td>Simple way to override the drawerImage in the navBar</td>
</tr>
</tbody>
</table>
<h4 id="Navigation-Bar--Title-标题"><a href="https://ld246.com/forward?goto=https%3A%
F%2Fgithub.com%2Faksonov%2Freact-native-router-flux%2Fblob%2Fmaster%2Fdocs%2FAPI
CONFIGURATION.md%23navigation-bar-title" target="_blank" rel="nofollow ugc">Navigatio
Bar: Title 标题</h4>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td><code>string</code></td>
<td>null</td>
<td>The title to be displayed in the navigation bar 设置导航栏标题</td>
</tr>
<tr>
<td>getTitle</td>
<td><code>function</code></td>
<td>optional</td>
<td>Optionally closure to return a value of the title based on state 根据 state 返回标题</td>
</tr>
<tr>
<td>renderTitle</td>
<td><code>function</code></td>
<td>optional</td>
<td>Optionally closure to render the title</td>
</tr>
<tr>
<td>titleStyle</td>

<td><a href="https://ld246.com/forward?goto=https%3A%2F%2Ffacebook.github.io%2Freac
-native%2Fdocs%2Ftext.html%23style" target="_blank" rel="nofollow ugc"><code>Text styl
</code></td>
<td></td>
<td>optional style override for the title element 重写标题的样式</td>
</tr>
<tr>
<td>titleWrapperStyle</td>
<td><a href="https://ld246.com/forward?goto=https%3A%2F%2Ffacebook.github.io%2Freac
-native%2Fdocs%2Fview.html%23style" target="_blank" rel="nofollow ugc"><code>View sty
e</code></td>
<td></td>
<td>optional style override for the title wrapper 重写包裹标题的样式</td>
</tr>
<tr>
<td>titleOpacity</td>
<td><code>string</code></td>
<td>optional</td>
<td>Set opacity for the title in navigation bar 在导航栏中设置不透明的标题</td>
</tr>
<tr>
<td>titleProps</td>
<td><code>object</code></td>
<td>null</td>
<td>Any other properties to be set on the title component 任何其他的属性都会被设置到标题
件上</td>
</tr>
</tbody>
</table>
<h4 id="Navigation-Bar--Back-button-导航条的返回按钮"><a href="https://ld246.com/forwar
?goto=https%3A%2F%2Fgithub.com%2Faksonov%2Freact-native-router-flux%2Fblob%2Fmas
er%2Fdocs%2FAPI_CONFIGURATION.md%23navigation-bar-back-button" target="_blank" re
="nofollow ugc">Navigation Bar: Back button 导航条的返回按钮</h4>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backTitle</td>
<td><code>string</code></td>
<td></td>
<td>optional string to display with back button</td>
</tr>
<tr>
<td>renderBackButton</td>
<td><code>function</code></td>
<td></td>
<td>optional closure to render back text or button if this route happens to be the previous r

ute 如果该路由恰好是之前的路由，关闭重新渲染返回按钮文字或者按钮的行为</td>
</tr>
<tr>
<td>backButtonImage</td>
<td><code>Imaesource</code></td>
<td><code>require('./back_chevron.png')</code></td>
<td>Simple way to override the back button in the navBar</td>
</tr>
<tr>
<td>backButtonTextStyle</td>
<td><code>Textstyle/code></td>
<td></td>
<td>optional style override for the back title element</td>
</tr>
<tr>
<td>hideBackImage</td>
<td><code>boolean</code></td>
<td>false</td>
<td>no default back button image will be displayed 隐藏返回按钮图片</td>
</tr>
<tr>
<td>onBack</td>
<td><code>function</code></td>
<td>Actions.pop</td>
<td>actions for back button 点击返回按钮时的行为，默认是 Actions.pop</td>
</tr>
</tbody>
</table>
<h4 id="Navigation-Bar--Left-button-"><a href="https://ld246.com/forward?goto=https%3%2F%2Fgithub.com%2Faksonov%2Freact-native-router-flux%2Fblob%2Fmaster%2Fdocs%2FPI_CONFIGURATION.md%23navigation-bar-left-button" target="_blank" rel="nofollow ugc"Navigation Bar: Left button </h4>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>leftTitle</td>
<td><code>string</code></td>
<td></td>
<td>optional string to display on the left if the previous route does not provide <code>rendrBackButton</code> prop. <code>renderBackButton</code> > <code>leftTitle</code>gt;</td>
</tr>

<tr>
<td>getLeftTitle</td>
<td><code>function</code></td>
<td></td>
<td>optional closure to display on the left if the previous route does not provide <code>ren
erBackButton</code> prop. <code>renderBackButton</code> > <code>getLeftTitle</c
de> ></td>
</tr>
<tr>
<td>renderLeftButton</td>
<td><code>function</code></td>
<td></td>
<td>optional closure to render the left title / buttons element</td>
</tr>
<tr>
<td>onLeft</td>
<td><code>function</code></td>
<td></td>
<td>function will be called when left navBar button is pressed</td>
</tr>
<tr>
<td>leftButtonImage</td>
<td><a href="https://ld246.com/forward?goto=https%3A%2F%2Ffacebook.github.io%2Freac
-native%2Fdocs%2Fimage.html%23source" target="_blank" rel="nofollow ugc"><code>Ima
esource</code></td>
<td></td>
<td>Image for left button</td>
</tr>
<tr>
<td>leftButtonIconStyle</td>
<td><a href="https://ld246.com/forward?goto=https%3A%2F%2Ffacebook.github.io%2Freac
-native%2Fdocs%2Fview.html%23style" target="_blank" rel="nofollow ugc"><code>View sty
e</code></td>
<td></td>
<td>Image style for left button</td>
</tr>
<tr>
<td>leftButtonStyle</td>
<td><a href="https://ld246.com/forward?goto=https%3A%2F%2Ffacebook.github.io%2Freac
-native%2Fdocs%2Fview.html%23style" target="_blank" rel="nofollow ugc"><code>View sty
e</code></td>
<td></td>
<td>optional style override for the container of left title / buttons</td>
</tr>
<tr>
<td>leftButtonTextStyle</td>
<td><a href="https://ld246.com/forward?goto=https%3A%2F%2Ffacebook.github.io%2Freac
-native%2Fdocs%2Ftext.html%23style" target="_blank" rel="nofollow ugc"><code>Text styl
</code></td>
<td></td>
<td>optional style override for the left title element</td>
</tr>
</tbody>
</table>

<h4 id="Navigation-Bar--Right-button">Navigation Bar: Right button</h4>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rightTitle</td>
<td><code>string</code></td>
<td></td>
<td>optional string to display on the right. <code>onRight</code> must be provided for thi to appear.</td>
</tr>
<tr>
<td>getRightTitle</td>
<td><code>function</code></td>
<td></td>
<td>optional closure to display on the right. <code>onRight</code> must be provided for t is to appear.</td>
</tr>
<tr>
<td>renderRightButton</td>
<td><code>function</code></td>
<td></td>
<td>optional closure to render the right title / buttons element</td>
</tr>
<tr>
<td>onRight</td>
<td><code>function</code></td>
<td></td>
<td>function will be called when right navBar button is pressed</td>
</tr>
<tr>
<td>rightButtonImage</td>
<td><code>Ima e source</code></td>
<td></td>
<td>Image for right button</td>
</tr>
<tr>
<td>rightButtonIconStyle</td>
<td><code>View sty e</code></td>
<td></td>

<td>Image style for right button</td>
</tr>
<tr>
<td>rightButtonStyle</td>
<td><code>View style</code></td>
<td></td>
<td>optional style override for the container of right title / buttons</td>
</tr>
<tr>
<td>rightButtonTextStyle</td>
<td><code>Text style</code></td>
<td></td>
<td>optional style override for the right title element</td>
</tr>
<tr>
<td>...other props</td>
<td></td>
<td></td>
<td>all properties that will be passed to your component instance</td>
</tr>
</tbody>
</table>