# react-native-safe-area-context 用法

马六甲的笔记 关注

0.101 2021.10.10 17:22:30 字数 838 阅读 2,879

react-native-safe-area-context 主要用于处理异形屏的适配，`React Navigation` 的适配就是使用该组件进行处理的。`React Navigation` V5 版是通过 `safeAreaInsets` 属性进行设置的，新版将该功能提取出来，通过 组件、Hook 方式处理，这样做的好处是：无需自行监听屏幕旋转，会自动更新同步渲染界面，处理起来也更加灵活。

## 安装

```
yarn add react-native-safe-area-context
npx pod-install
```

## 使用

### SafeAreaProvider

这是一个提供者，本身不会对布局产生任何影响，但只有在该组件包裹下的子组件才能使用 `react-native-safe-area-context` 提供的功能，通常，可以直接包裹在根组件上。`React Navigation` 本身已经使用该组件作了包裹，所以在配合 `React Navigation` 使用时，无需再进行包裹了，直接使用即可。独立使用时，可使用类似如下的代码：

```
import { SafeAreaProvider } from 'react-native-safe-area-context';

function App() {
  // 通常可以在 APP 最外层使用，也可以在深层使用，但只有子组件才能API
  // 注意：不要把该组件放到有动画或滚动的组件下级，比如 Animated 或 ScrollView
  // 支持 View 的所有属性，并支持额外的一个 initialMetrics 属性
  return <SafeAreaProvider initialMetrics={null}>...</SafeAreaProvider>;
}
```

### initialWindowMetrics

上面 `SafeAreaProvider` 的 `initialMetrics` 属性需要提供一个 Object 值，提供相关的尺寸位置信息，默认为自动获取，无需提供。默认提供的信息可以通过该 Hook 获取

```
import { initialWindowMetrics } from 'react-native-safe-area-context';

function HookComponent() {
  // 数据格式
  // {
  //    frame: { x: number, y: number, width: number, height: number },
  //    insets: { top: number, left: number, right: number, bottom: number },
  // }
  const insets = initialWindowMetrics();

  ...
}
```

### useSafeAreaFrame / SafeAreaFrameContext

获取离当前组件最近的 `SafeAreaProvider` 尺寸信息

```
import {
   useSafeAreaFrame,
   SafeAreaFrameContext
} from 'react-native-safe-area-context';

// 函数式组件
function HookComponent() {
  // 获取 SafeAreaProvider 的宽高、偏移 x,y
```

```
  const {x, y, width, height} = useSafeAreaFrame();

  // ...
}

// class 组件
class ClassComponent extends React.Component {
  render() {
    return (
      <SafeAreaFrameContext.Consumer>
        {(frame) => <View ... />}
      </SafeAreaFrameContext.Consumer>
    );
  }
}
```

## useSafeAreaInsets / SafeAreaInsetsContext

获取当前屏幕异形部分的尺寸

```
import {
  useSafeAreaInsets,
  SafeAreaInsetsContext
} from 'react-native-safe-area-context';

// 函数式组件
function HookComponent() {
  // 该 Hook 返回屏幕四个方向上异形的尺寸
  // 若屏幕旋转，该值也会自动更新，促使组件同步更新
  const {left, right, top, bottom} = useSafeAreaInsets();

  return <View style={{ paddingBottom: Math.max(bottom, 16) }} />;
}

// class 组件
class ClassComponent extends React.Component {
  render() {
    return (
      <SafeAreaInsetsContext.Consumer>
        {(insets) => <View style={{ paddingTop: insets.top }} />}
      </SafeAreaInsetsContext.Consumer>
    );
  }
}
```

## withSafeAreaInsets

上面两组分别是使用 Hook / Context 方式获取相关尺寸数值应用到 `Function`、`Class` 组件，对于 AreaInsets 还可使用 `withSafeAreaInsets` 应用到高阶组件。

```
class MyConmpoent extends React.Component {
  render() {
    const {insets} = this.props;
    return <View style={{ paddingTop: insets.top }} />
  }
}

export default withSafeAreaInsets(MyConmpoent);
```

## SafeAreaView

以上都是比较灵活的方式，自行获取数值进行处理。还有另外一种较为方便的方式，在 `SafeAreaProvider` 任何层级内都可以使用 `SafeAreaView`，该组件与 `View` 相同，默认添加了 `padding` 属性，在 `View` 四周添加了空白用以避开屏幕异形的部分，该组件会在屏幕旋转、或数值发生变动时自动更新。

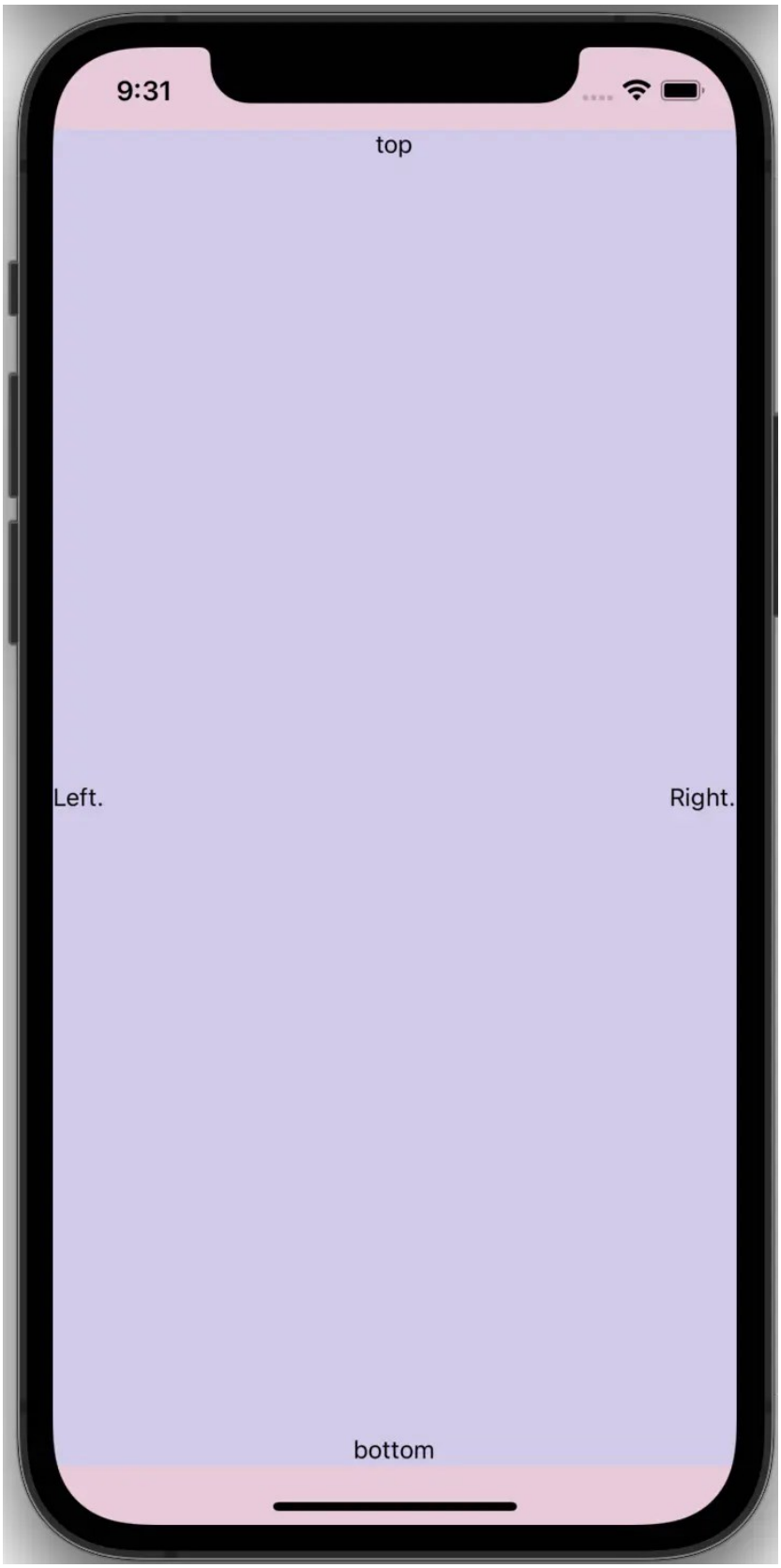
```
import { SafeAreaView } from 'react-native-safe-area-context';

function SomeComponent() {
  // 支持 edges / mode 两个属性
  // edges: 设置要添加空白的方向，数组形式
  // mode:添加空白的方式，支持 padding(默认) / margin
  return (
    <SafeAreaView
        style={{ flex: 1, backgroundColor: 'red' }}
```

```
        edges={['top', 'bottom', 'right', 'left']}
        mode="padding"
      >
        <View style={{ flex: 1, backgroundColor: 'blue' }} />
      </SafeAreaView>
    );
  }
```

## 图解

`react-native-safe-area-context` 返回屏幕异形尺寸，针对的是“非矩形”的部分，而不是“刘海”、“针孔”。如下图，倒脚部分都算作异形，在竖屏时通常没啥问题，但在横屏时可能不符合设计预期，需注意。




页面演示

以 `React Navigation` 举例，该组件已内置 `react-native-safe-area-context`，并在

`Header`、`BottomTab` 组件中处理了 `top` / `bottom` 方向的异形屏。在使用 `React Navigation` 时，如果是一般情况，在页面组件中无需刻意处理，如下图

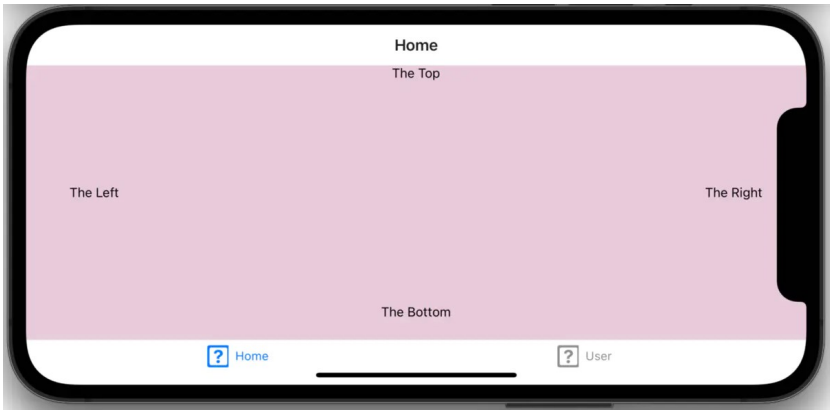

竖屏演示

但碰到以下两种情况时，仍需手动处理

- 页面组件未使用 `Header` 或 `BottomTab`，这就与上面的普通组件没什么区别了，很好理解
- 页面支持横屏，在横屏时，如未做任何处理，效果如下图


未对横屏进行处理

如果将页面组件包裹在 `SafeAreaView` 组件内，`top` / `bottom` 方向异形尺寸为 `0`，`left` / `right` 方向会自动应用 `padding`，效果如下图，但很有可能，下图也不一定符合设计预期，比如 `left` 方向，圆角异形导致了 `padding`，但同时也浪费了不少可视区域，可通过 `SafeAreaView` 的 `edges` 属性进行设置。




页面组件使用 SafeAreaView