

 [react-native-maps](#) / [react-native-maps](#)

React Native Mapview component for iOS + Android

 MIT License

☆ 12k stars    🔗 4.1k forks

☆ Star

 Notifications

< > Code

! Issues 60

🔗 Pull requests 4

▶ Actions

📁 Projects 1

🛡 Security

🔗 master ▾

Go to file



eyalyoli Better docs for custom markers (#3771) ...

✓ 4 days ago ⌚ 1,097

[View code](#)

# react-native-maps npm v0.27.1

React Native Map components for iOS + Android



## Maintainers Wanted maintainers wanted

We are in need of more people or companies willing to help. If you have enough time and knowledge, and want to become a maintainer, please let us know [here](#).

## Installation

See [Installation Instructions](#).

See [Setup Instructions for the Included Example Project](#).

## Compatibility

Due to the rapid changes being made in the React Native ecosystem, we are not officially going to support this module on anything but the latest version of React Native. With that said, we will do our best to stay compatible with older versions as much that is practical, and the peer dependency of this requirement is set to `"react-native": "*" explicitly for this reason. If you are using an older version of React Native with this module though, some features may be buggy.`

## Component API

---

`<MapView />` [Component API](#)

`<Marker />` [Component API](#)

`<Callout />` [Component API](#)

`<Polygon />` [Component API](#)

`<Polyline />` [Component API](#)

`<Circle />` [Component API](#)

`<Overlay />` [Component API](#)

`<Heatmap />` [Component API](#)

`<Geojson />` [Component API](#)

## General Usage

---

```
import MapView from 'react-native-maps';
```

or

```
var MapView = require('react-native-maps');
```

This MapView component is built so that features on the map (such as Markers, Polygons, etc.) are specified as children of the MapView itself. This provides an intuitive and react-like API for declaratively controlling features on the map.

## Rendering a Map with an initial region

# MapView

---

```
<MapView
  initialRegion={{
    latitude: 37.78825,
    longitude: -122.4324,
    latitudeDelta: 0.0922,
    longitudeDelta: 0.0421,
  }}
/>
```

## Using a MapView while controlling the region as state

```
getInitialState() {
  return {
    region: {
      latitude: 37.78825,
      longitude: -122.4324,
      latitudeDelta: 0.0922,
      longitudeDelta: 0.0421,
    },
  };
}

onRegionChange(region) {
  this.setState({ region });
}

render() {
  return (
    <MapView
      region={this.state.region}
      onRegionChange={this.onRegionChange}
    />
  );
}
```

## Rendering a list of markers on a map

```
import { Marker } from 'react-native-maps';

<MapView
  region={this.state.region}
  onRegionChange={this.onRegionChange}
>
```

```

{this.state.markers.map((marker, index) => (
  <Marker
    key={index}
    coordinate={marker.latlng}
    title={marker.title}
    description={marker.description}
  />
))}
</MapView>

```

## Rendering a Marker with a custom image

1. You need to generate an png image with various resolution (lets call them custom\_pin) - for more information go to [Android](#), [iOS](#)
2. put all images in Android drawables and iOS assets dir
3. Now you can use the following code:

```

<Marker
  coordinate={{ latitude : latitude , longitude : longitude }}
  image={{uri: 'custom_pin'}}
/>

```

Note: You can also pass the image binary data like `image={require('custom_pin.png')}`, but this will not scale good with the different screen sizes.

## Rendering a Marker with a custom view

Note: This has performance implications, if you wish for a simpler solution go with a custom image (save your self the head ache)

```

<Marker coordinate={{ latitude : latitude , longitude : longitude }}>
  <MyCustomMarkerView {...marker} />
</Marker>

```

## Rendering a custom Marker with a custom Callout

```

import { Callout } from 'react-native-maps';

<Marker coordinate={marker.latlng}>
  <MyCustomMarkerView {...marker} />
  <Callout>
    <MyCustomCalloutView {...marker} />
  </Callout>
</Marker>

```

```
</Callout>
</Marker>
```

## Draggable Markers

```
<MapView initialRegion={...}>
  <Marker draggable
    coordinate={this.state.x}
    onDragEnd={(e) => this.setState({ x: e.nativeEvent.coordinate })}
  />
</MapView>
```

## Using a custom Tile Overlay

### Tile Overlay using tile server

```
import { UrlTile } from 'react-native-maps';

<MapView
  region={this.state.region}
  onRegionChange={this.onRegionChange}
>
  <UrlTile
    /**
     * The url template of the tile server. The patterns {x} {y} {z} will be replace
     * For example, http://c.tile.openstreetmap.org/{z}/{x}/{y}.png
     */
    urlTemplate={this.state.urlTemplate}
    /**
     * The maximum zoom level for this tile overlay. Corresponds to the maximumZ set
     * MKTileOverlay. iOS only.
     */
    maximumZ={19}
    /**
     * flipY allows tiles with inverted y coordinates (origin at bottom left of map)
     * to be used. Its default value is false.
     */
    flipY={false}
  />
</MapView>
```

For Android: add the following line in your AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET" />
```

For IOS: configure [App Transport Security](#) in your app

## Tile Overlay using local tiles

Tiles can be stored locally within device using xyz tiling scheme and displayed as tile overlay as well. This is usefull especially for offline map usage when tiles are available for selected map region within device storage.

```
import { LocalTile } from 'react-native-maps';

<MapView
  region={this.state.region}
  onRegionChange={this.onRegionChange}
>
  <LocalTile
    /**
     * The path template of the locally stored tiles. The patterns {x} {y} {z} will b
     * For example, /storage/emulated/0/mytiles/{z}/{x}/{y}.png
     */
    pathTemplate={this.state.pathTemplate}
    /**
     * The size of provided local tiles (usually 256 or 512).
     */
    tileSize={256}
  />
</MapView>
```

For Android: LocalTile is still just overlay over original map tiles. It means that if device is online, underlying tiles will be still downloaded. If original tiles download/display is not desirable set mapType to 'none'. For example:

```
<MapView
  mapType={Platform.OS == "android" ? "none" : "standard"}
>
```

See [OSM Wiki](#) for how to download tiles for offline usage.

## Overlaying other components on the map

Place components you that wish to overlay MapView underneath the MapView closing tag. Absolutely position these elements.

```
render() {
  return (
    <MapView
      region={this.state.region}
    />
    <OverlayComponent
      style={{position: "absolute", bottom: 50}}
    />
  );
}
```

## Customizing the map style

Create the json object, or download a generated one from the [google style generator](#).

```
// The generated json object
mapStyle = [ ... ]

render() {
  return (
    <MapView
      region={this.state.region}
      onRegionChange={this.onRegionChange}
      customMapStyle={mapStyle}
    />
  );
}
```

For iOS, in addition to providing the `mapStyle` you will need to do the following

```
import MapView, { PROVIDER_GOOGLE } from 'react-native-maps'

// ...

<MapView
  provider={PROVIDER_GOOGLE}
  customMapStyle={MapStyle}
/>
```

Then add the AirGoogleMaps directory:

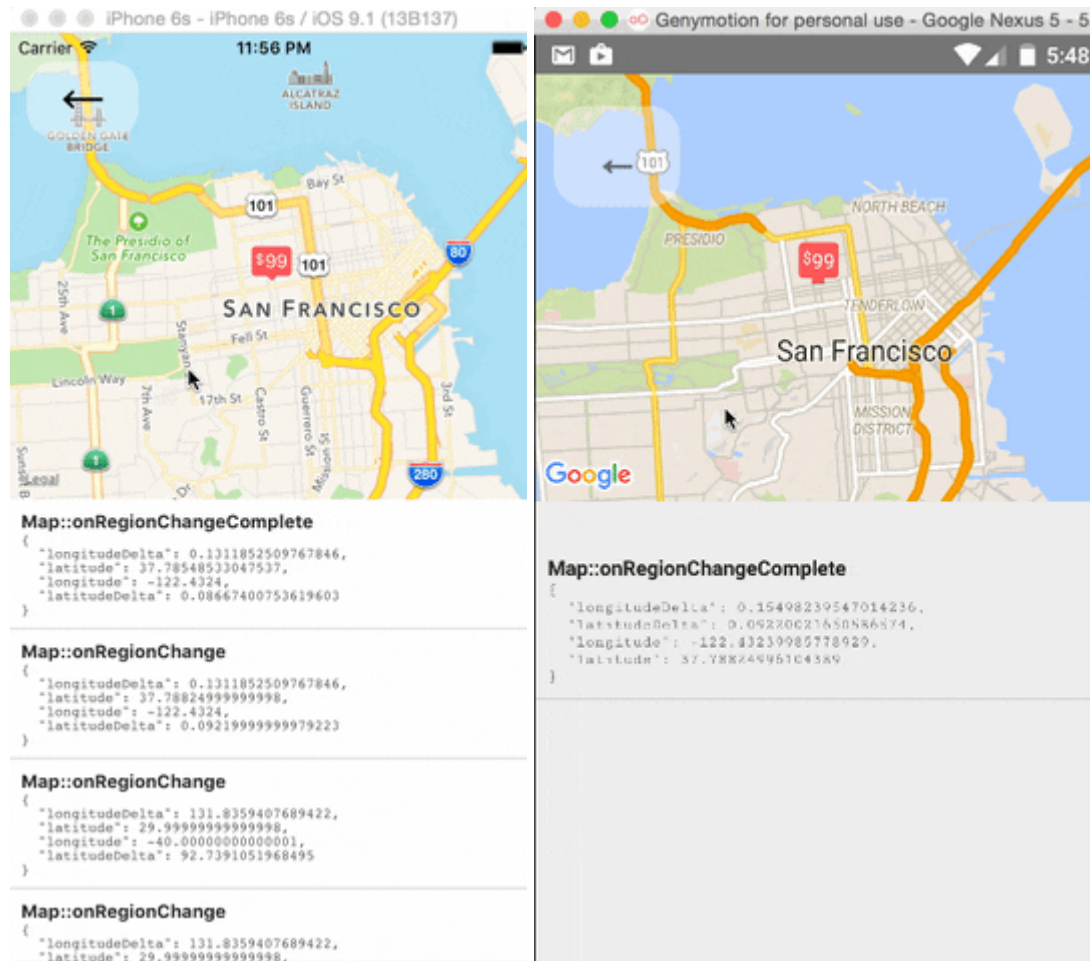
<https://github.com/react-native-maps/react-native-maps/blob/1e71a21f39e7b88554852951f773c731c94680c9/docs/installation.md#ios>

An unofficial step-by-step guide is also available at

<https://gist.github.com/heron2014/e60fa003e9b117ce80d56bb1d5bfe9e0>

## MapView Events

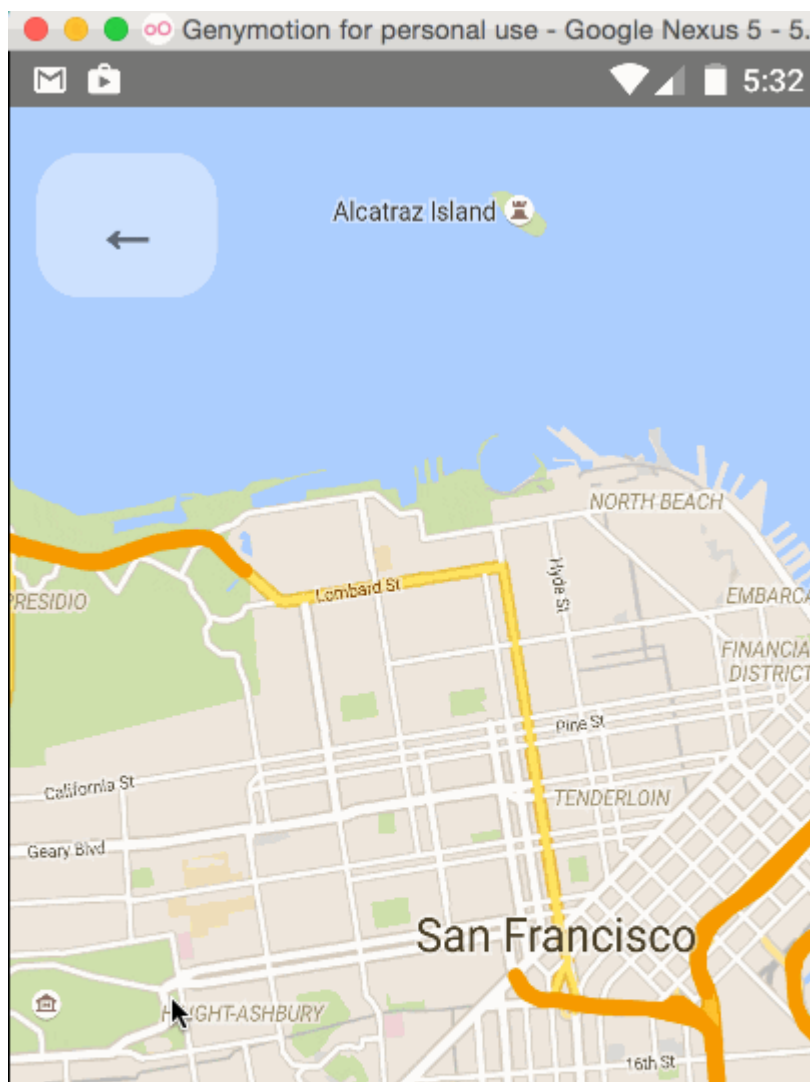
The `<MapView />` component and its child components have several events that you can subscribe to. This example displays some of them in a log as a demonstration.

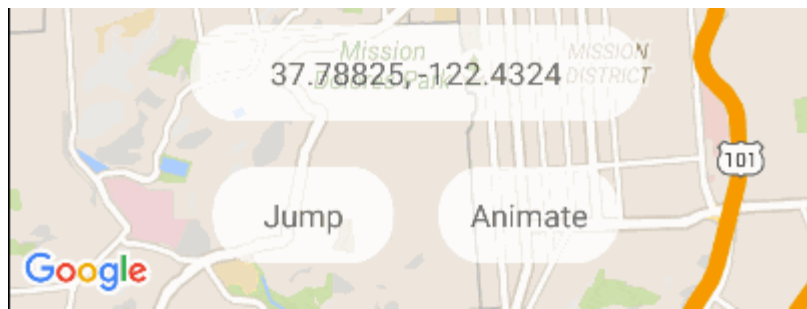


## Tracking Region / Location



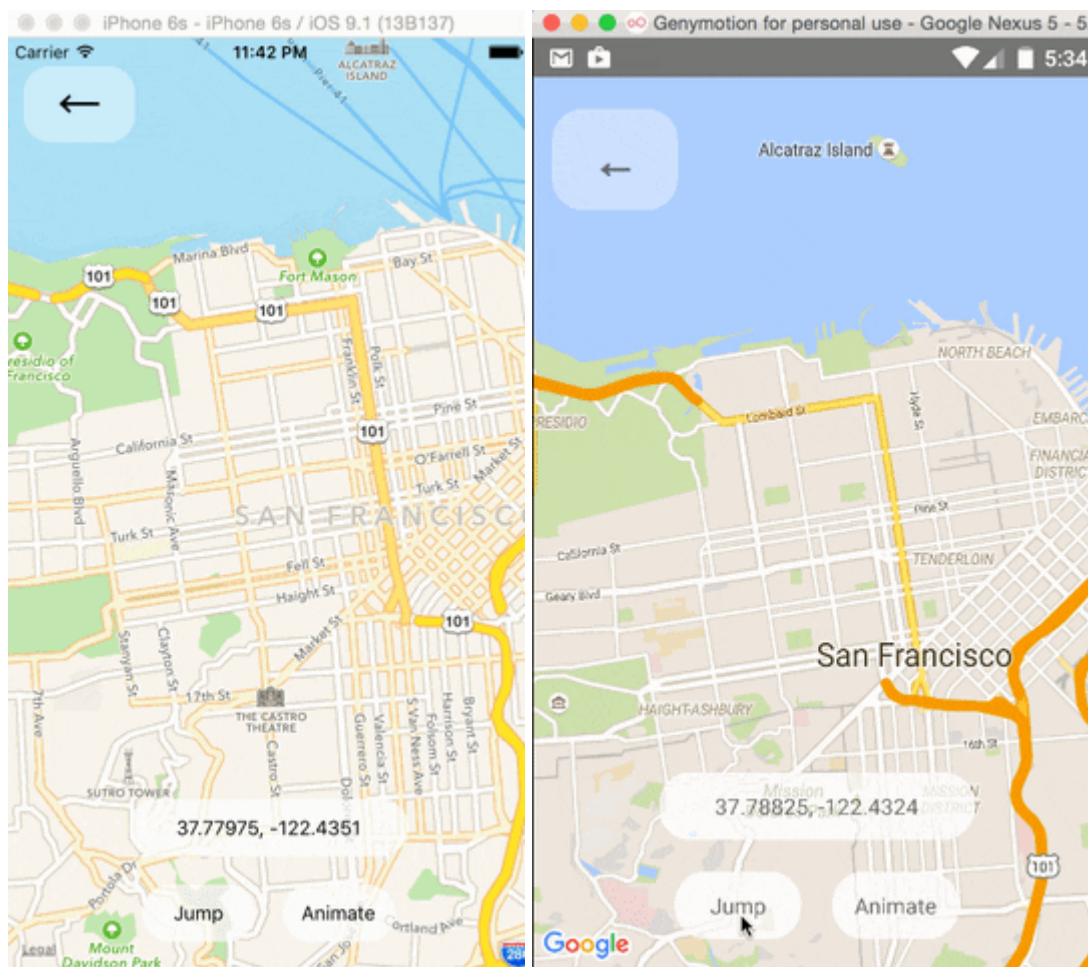




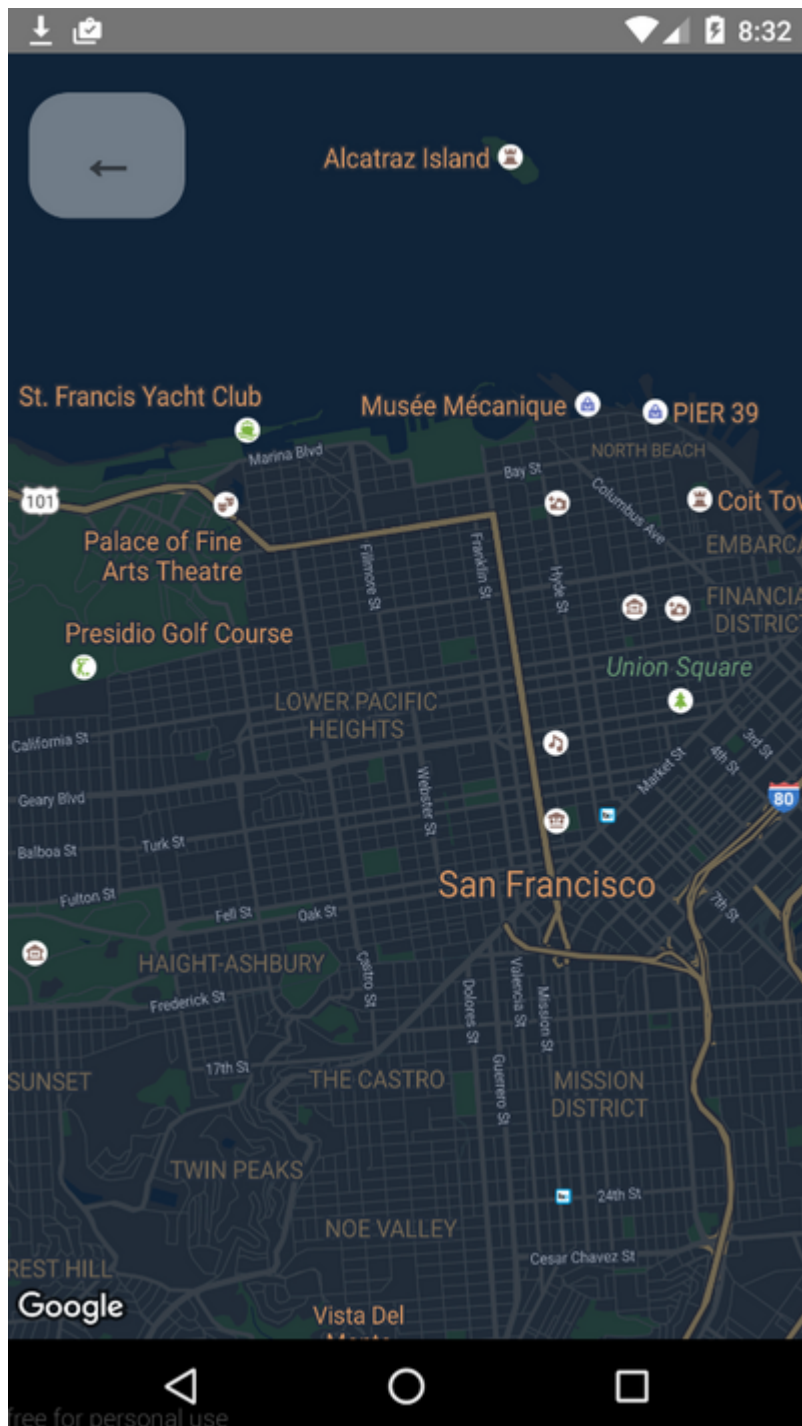


## Programmatically Changing Region

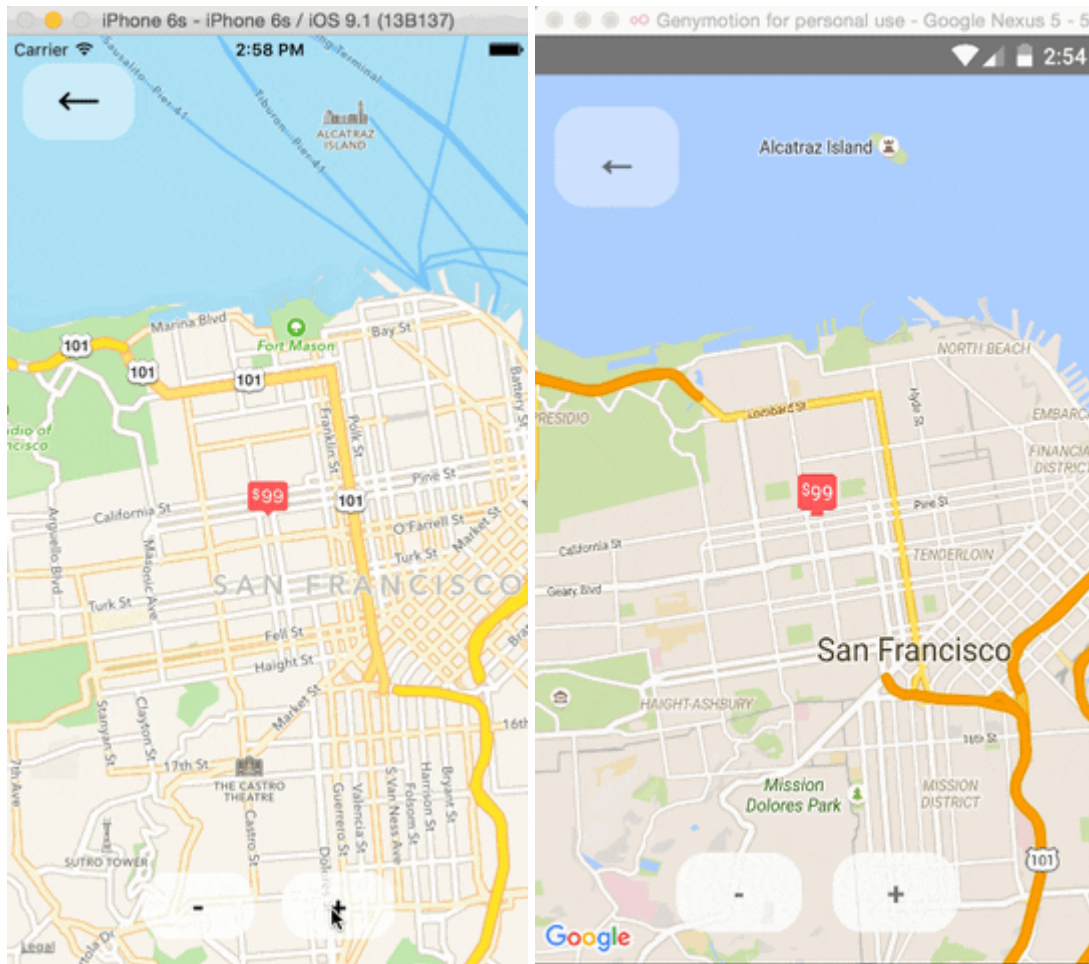
One can change the mapview's position using refs and component methods, or by passing in an updated `region` prop. The component methods will allow one to animate to a given position like the native API could.



## Changing the style of the map



## Arbitrary React Views as Markers

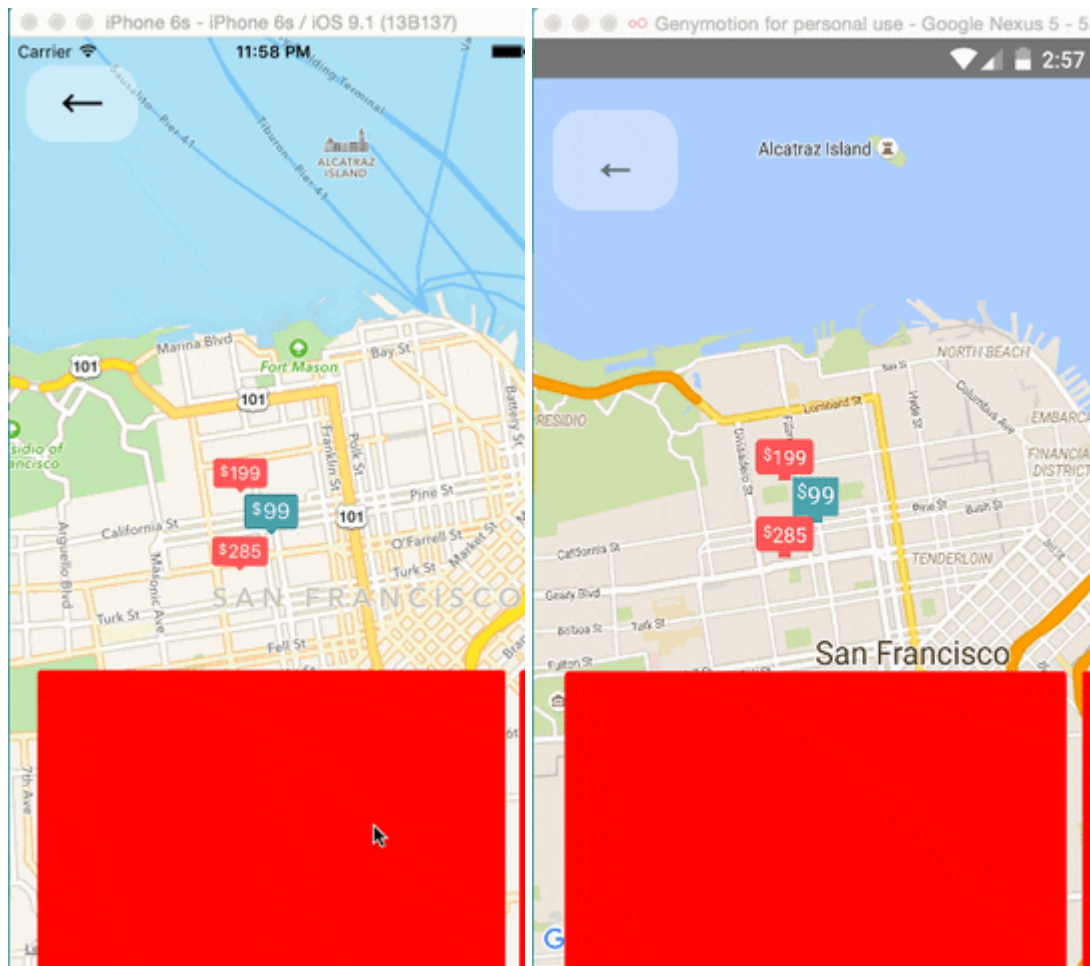


## Using the MapView with the Animated API

The `<MapView />` component can be made to work with the Animated API, having the entire `region` prop be declared as an animated value. This allows one to animate the zoom and position of the MapView along with other gestures, giving a nice feel.

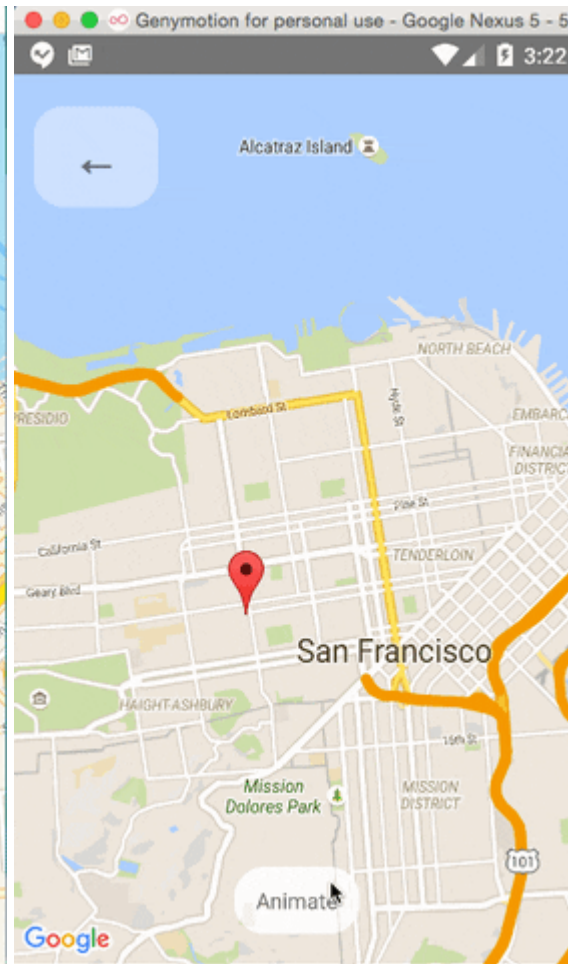
Further, Marker views can use the animated API to enhance the effect.



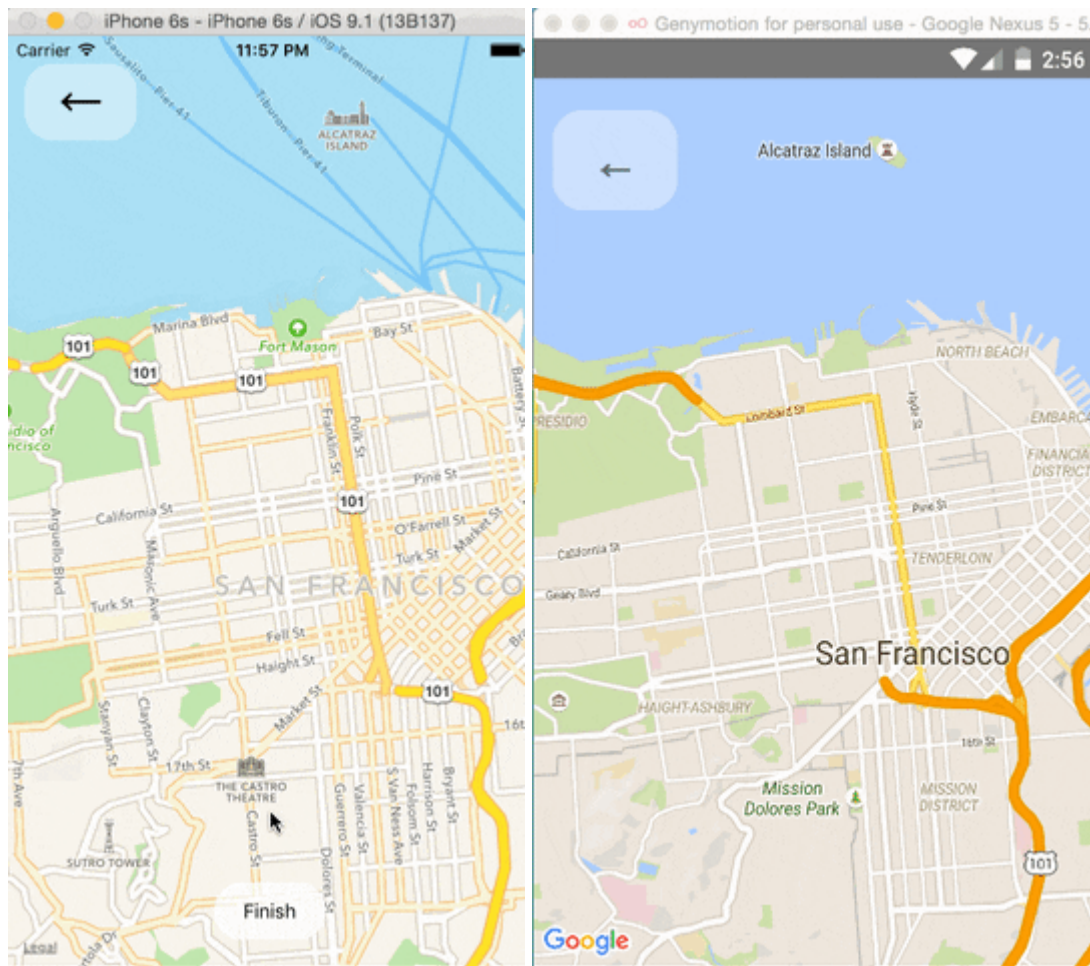


Issue: Since android needs to render its marker views as a bitmap, the animations APIs may not be compatible with the Marker views. Not sure if this can be worked around yet or not.

Markers' coordinates can also be animated, as shown in this example:



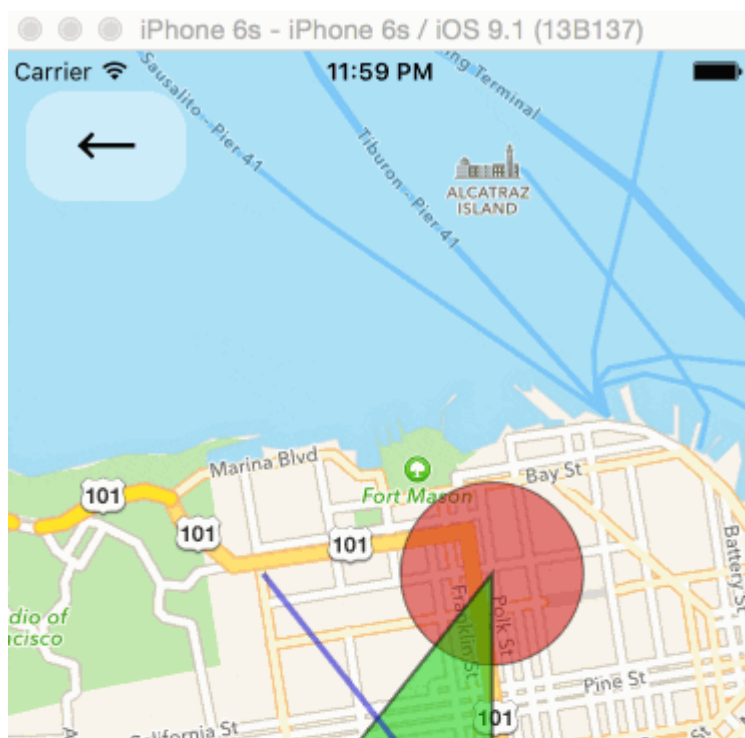
## Polygon Creator

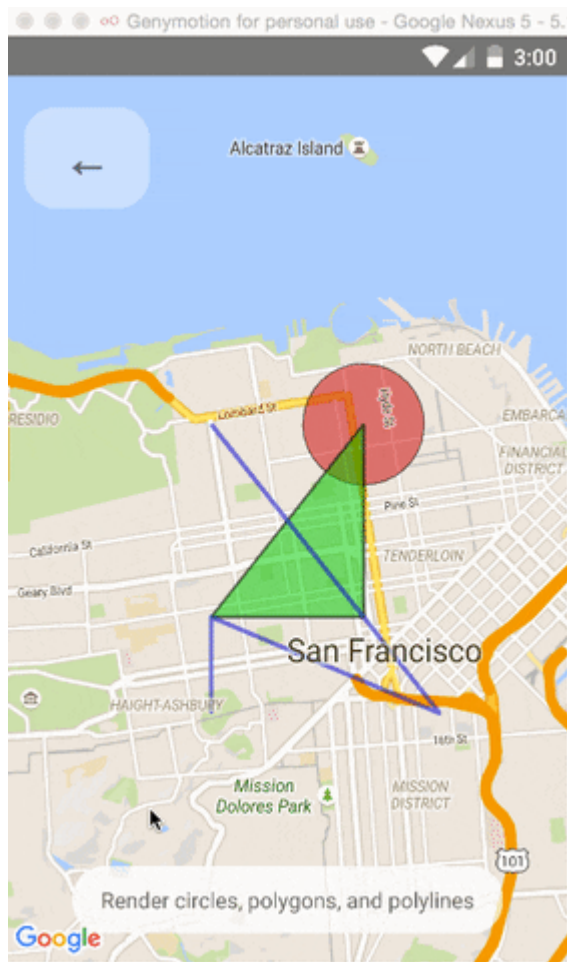
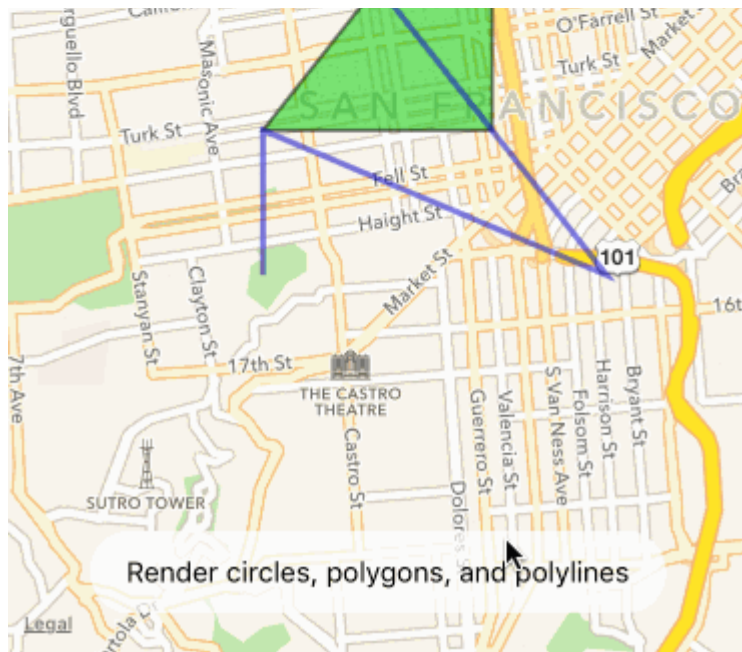


## Other Overlays

So far, `<Circle />`, `<Polygon />`, and `<Polyline />` are available to pass in as children to the `<MapView />` component.

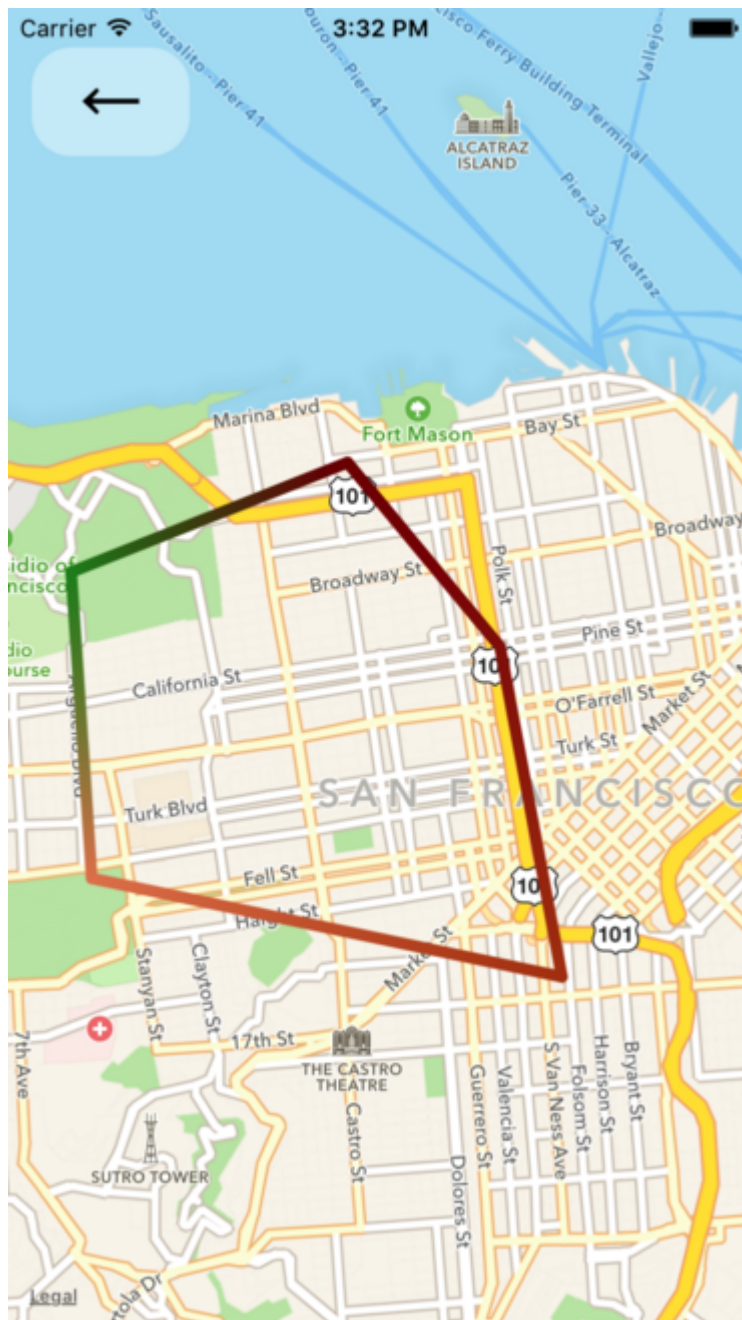






## Gradient Polylines (iOS MapKit only)

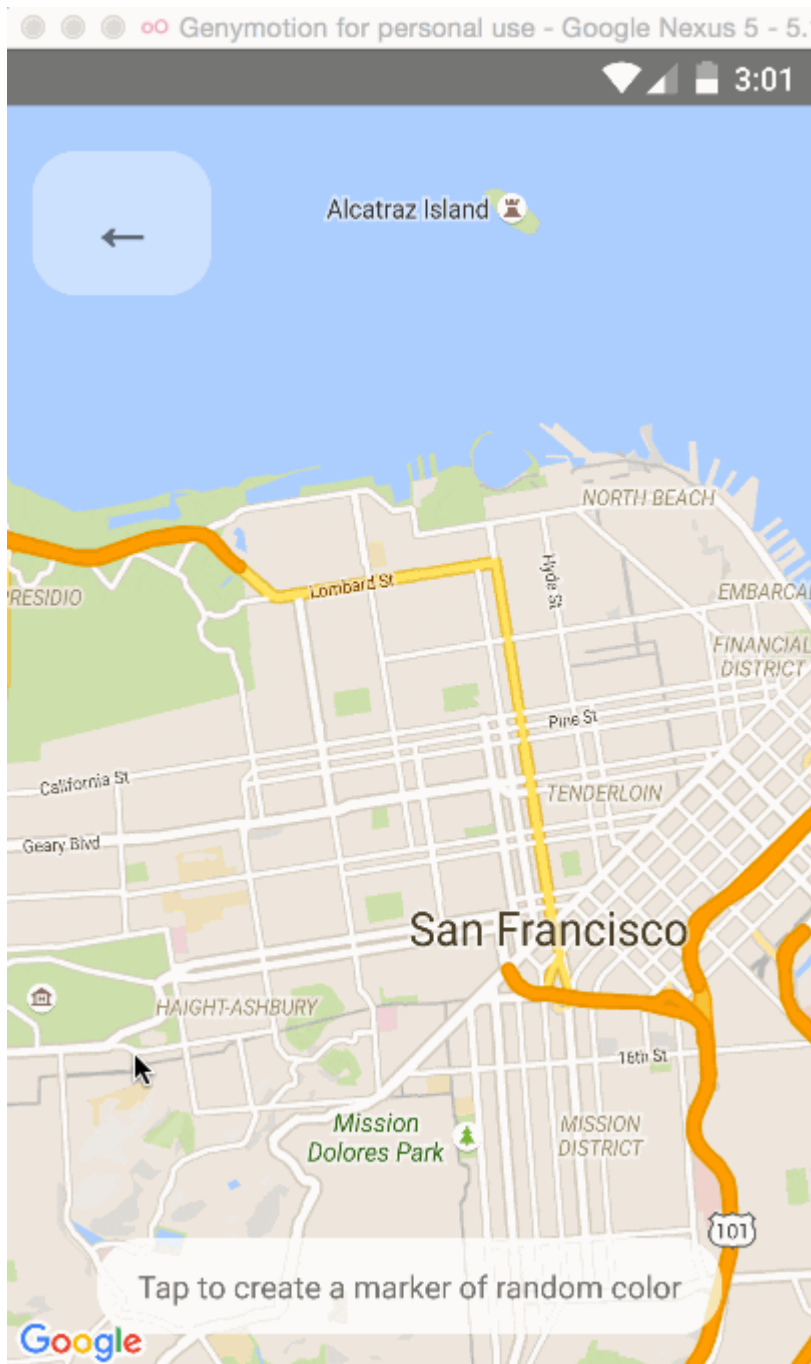
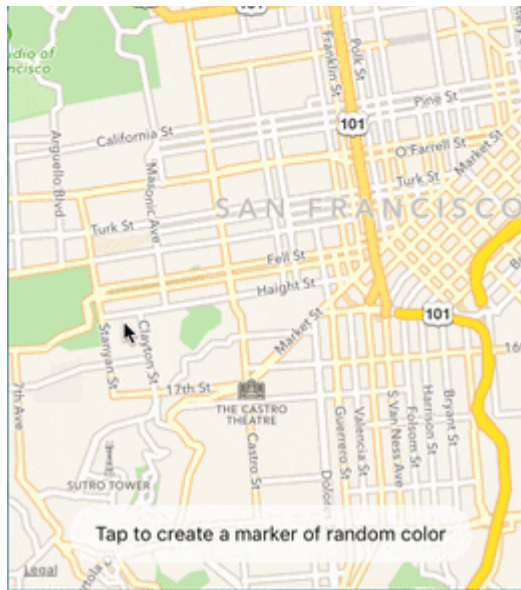
Gradient polylines can be created using the `strokeColors` prop of the `<Polyline>` component.



## Default Markers

Default markers will be rendered unless a custom marker is specified. One can optionally adjust the color of the default marker by using the `pinColor` prop.





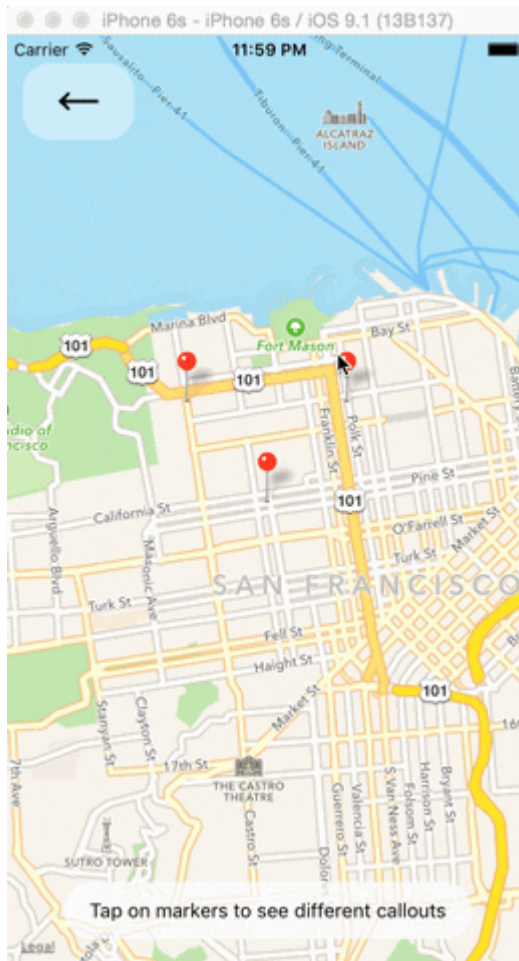
## Custom Callouts

Callouts to markers can be completely arbitrary react views, similar to markers. As a result, they can be interacted with like any other view.

Additionally, you can fall back to the standard behavior of just having a title/description through the `<Marker />`'s `title` and `description` props.

Custom callout views can be the entire tooltip bubble, or just the content inside of the system default bubble.

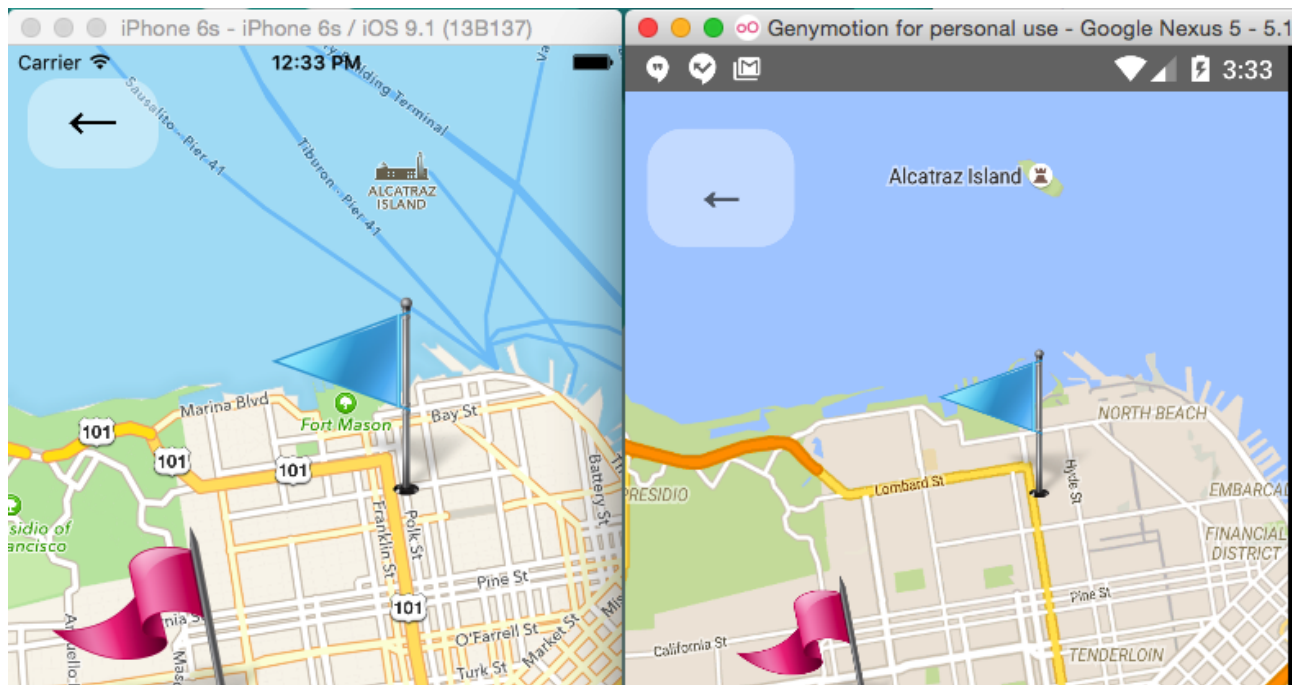
To handle press on specific subview of callout use `<CalloutSubview />` with `onPress`. See `Callouts.js` example.



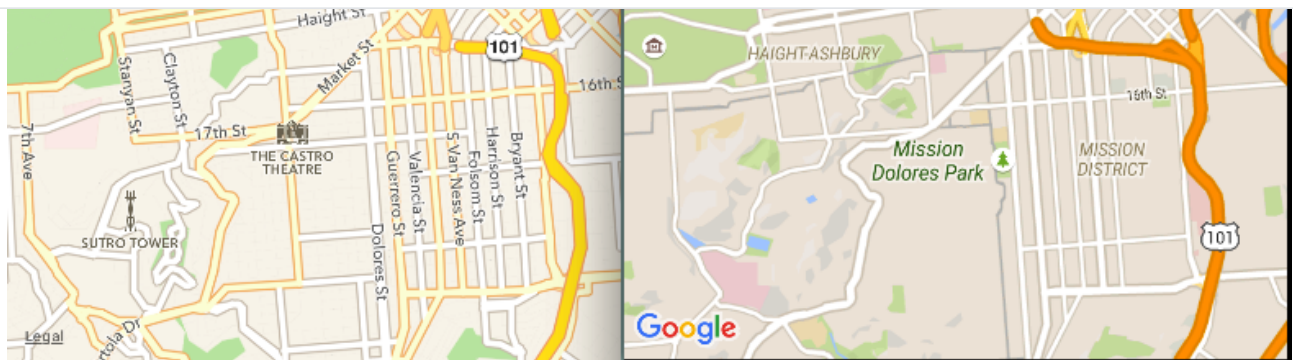
## Image-based Markers

Markers can be customized by just using images, and specified using the `image` prop.



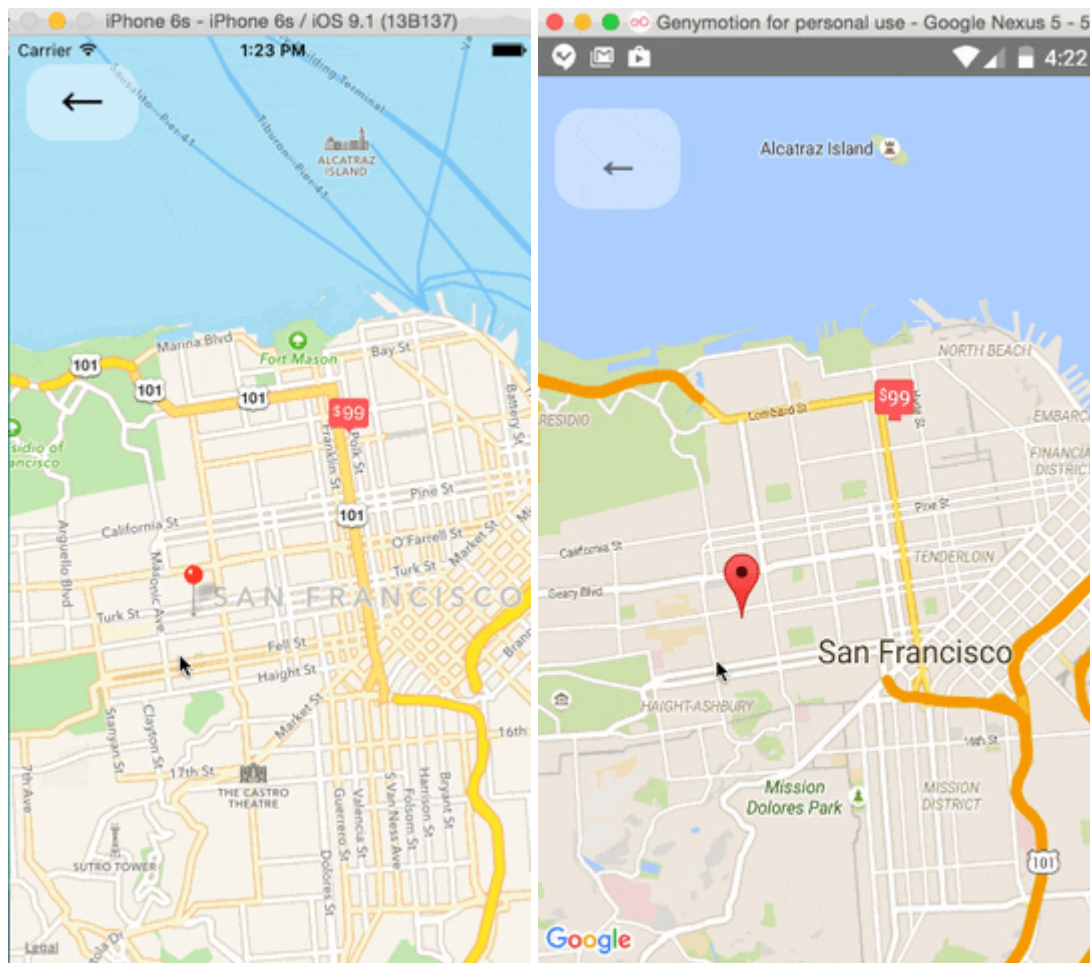


☰ README.md



## Draggable Markers

Markers are draggable, and emit continuous drag events to update other UI during drags.



## Lite Mode ( Android )

Enable lite mode on Android with `liteMode` prop. Ideal when having multiple maps in a View or ScrollView.

## On Poi Click (Google Maps Only)

Poi are clickable, you can catch the event to get its information (usually to get the full detail from Google Place using the `placeId`).

## Animated Region

The MapView can accept an `AnimatedRegion` value as its `region` prop. This allows you to utilize the Animated API to control the map's center and zoom.

```
import MapView, { AnimatedRegion, Animated } from 'react-native-maps';
```



```

getInitialState() {
  return {
    region: new AnimatedRegion({
      latitude: LATITUDE,
      longitude: LONGITUDE,
      latitudeDelta: LATITUDE_DELTA,
      longitudeDelta: LONGITUDE_DELTA,
    }),
  };
}

onRegionChange(region) {
  this.state.region.setValue(region);
}

render() {
  return (
    <Animated
      region={this.state.region}
      onRegionChange={this.onRegionChange}
    />
  );
}

```

## Animated Marker Position

Markers can also accept an `AnimatedRegion` value as a coordinate.

```

import Mapview, { AnimatedRegion, Marker } from 'react-native-maps';

getInitialState() {
  return {
    coordinate: new AnimatedRegion({
      latitude: LATITUDE,
      longitude: LONGITUDE,
    }),
  };
}

componentWillReceiveProps(nextProps) {
  const duration = 500

  if (this.props.coordinate !== nextProps.coordinate) {
    if (Platform.OS === 'android') {
      if (this.marker) {
        this.marker.animateMarkerToCoordinate(
          nextProps.coordinate,
          duration
        );
      }
    }
  }
}

```

```

    }
  } else {
    this.state.coordinate.timing({
      ...nextProps.coordinate,
      duration
    }).start();
  }
}
}

render() {
  return (
    <MapView initialRegion={...}>
      <MapView.Marker.Animated
        ref={marker => { this.marker = marker }}
        coordinate={this.state.coordinate}
      />
    </MapView>
  );
}

```

If you need a smoother animation to move the marker on Android, you can modify the previous example:

```

// ...

componentWillReceiveProps(nextProps) {
  const duration = 500

  if (this.props.coordinate !== nextProps.coordinate) {
    if (Platform.OS === 'android') {
      if (this.marker) {
        this.marker.animateMarkerToCoordinate(
          nextProps.coordinate,
          duration
        );
      }
    } else {
      this.state.coordinate.timing({
        ...nextProps.coordinate,
        duration
      }).start();
    }
  }
}

render() {
  return (
    <MapView initialRegion={...}>

```

```

      <Marker.Animated
        ref={marker => { this.marker = marker }}
        coordinate={this.state.coordinate}
      />
    </MapView>
  );
}

```

## Take Snapshot of map

```

import MapView, { Marker } from 'react-native-maps';

getInitialState() {
  return {
    coordinate: {
      latitude: LATITUDE,
      longitude: LONGITUDE,
    },
  };
}

takeSnapshot () {
  // 'takeSnapshot' takes a config object with the
  // following options
  const snapshot = this.map.takeSnapshot({
    width: 300,      // optional, when omitted the view-width is used
    height: 300,     // optional, when omitted the view-height is used
    region: {...},   // iOS only, optional region to render
    format: 'png',    // image formats: 'png', 'jpg' (default: 'png')
    quality: 0.8,     // image quality: 0..1 (only relevant for jpg, default: 1)
    result: 'file'    // result types: 'file', 'base64' (default: 'file')
  });
  snapshot.then((uri) => {
    this.setState({ mapSnapshot: uri });
  });
}

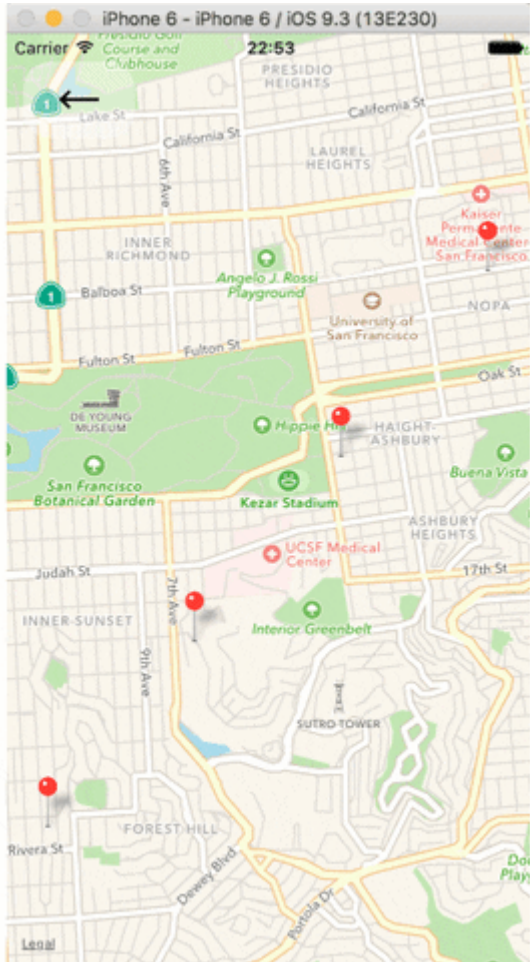
render() {
  return (
    <View>
      <MapView initialRegion={...} ref={map => { this.map = map }}>
        <Marker coordinate={this.state.coordinate} />
      </MapView>
      <Image source={{ uri: this.state.mapSnapshot.uri }} />
      <TouchableOpacity onPress={this.takeSnapshot}>
        Take Snapshot
      </TouchableOpacity>
    </View>
  );
}

```

```
);  
}
```

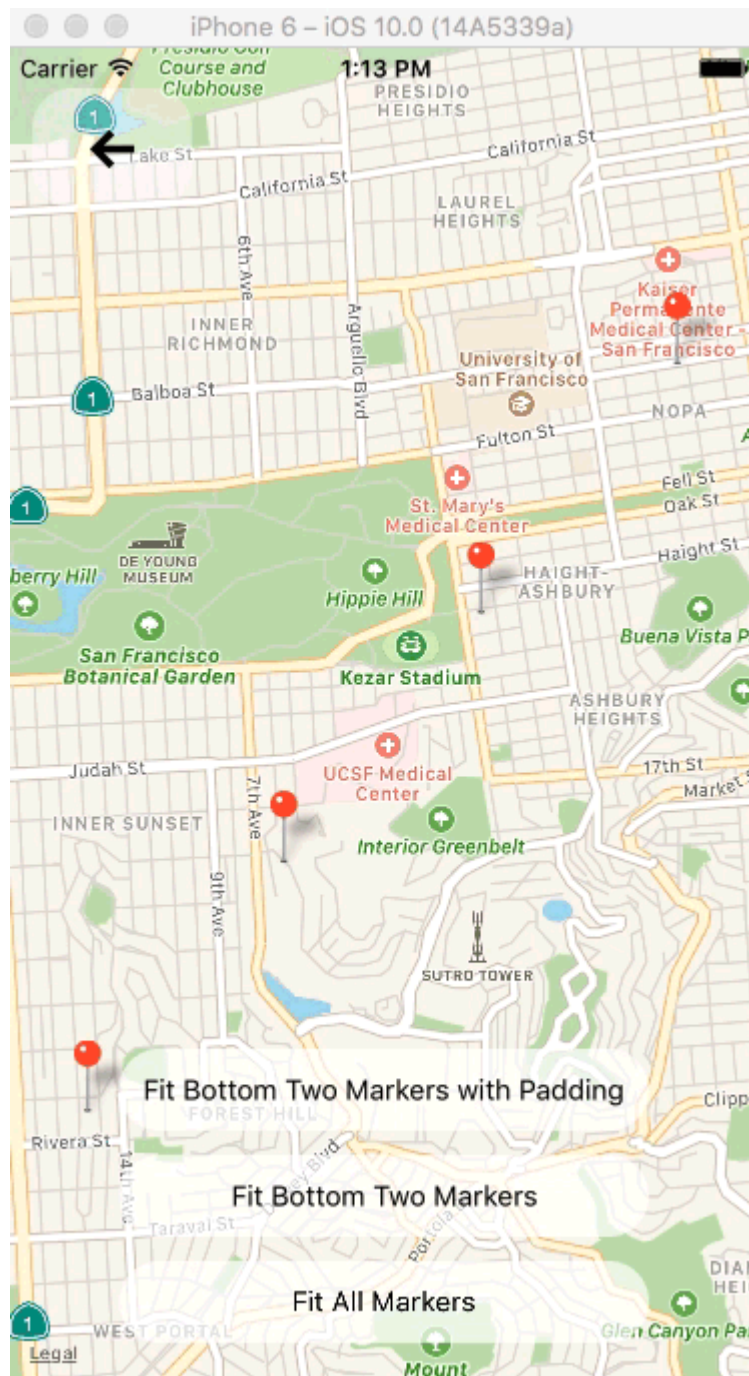
## Zoom to Specified Markers

Pass an array of marker identifiers to have the map re-focus.



## Zoom to Specified Coordinates

Pass an array of coordinates to focus a map region on said coordinates.



## Troubleshooting

### My map is blank

- Make sure that you have [properly installed](#) react-native-maps.
- Check in the logs if there is more informations about the issue.
- Try setting the style of the MapView to an absolute position with top, left, right and bottom values set.
- Make sure you have enabled Google Maps API in [Google developer console](#)

```
const styles = StyleSheet.create({
  map: {
    ...StyleSheet.absoluteFillObject,
  },
});
```

```
<MapView
  style={styles.map}
  // other props
/>
```

## Inputs don't focus

- When inputs don't focus or elements don't respond to tap, look at the order of the view hierarchy, sometimes the issue could be due to ordering of rendered components, prefer putting MapView as the first component.

Bad:

```
<View>
  <TextInput/>
  <MapView/>
</View>
```

Good:

```
<View>
  <MapView/>
  <TextInput/>
</View>
```

## Children Components Not Re-Rendering

Components that aren't declared by this library (Ex: Markers, Polyline) must not be children of the MapView component due to MapView's unique rendering methodology. Have your custom components / views outside the MapView component and position absolute to ensure they only re-render as needed. Example: Bad:

```
<View style={StyleSheet.absoluteFillObject}>
  <MapView style={StyleSheet.absoluteFillObject}>
    <View style={{ position: 'absolute', top: 100, left: 50 }}/>
```

```
</MapView>
</View>
```

Good:

```
<View style={StyleSheet.absoluteFillObject}>
  <MapView style={StyleSheet.absoluteFillObject} />
  <View style={{ position: 'absolute', top: 100, left: 50 }}/>
</View>
```

Source: <https://github.com/react-native-maps/react-native-maps/issues/1901>

## License

---

Copyright (c) 2017 Airbnb

Licensed under the The MIT License (MIT) (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<https://raw.githubusercontent.com/airbnb/react-native-maps/master/LICENSE>

Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an "AS IS" BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the License for the specific language governing permissions and  
limitations under the License.

## Releases 71

 **v0.27.1** Latest  
on Mar 19, 2020

[+ 70 releases](#)

## Packages

No packages published

---

Used by 114k



### Contributors 406



+ 395 contributors

### Languages







