This repository | Search   Pull requests Issues Marketplace Gist

wix / **react-native-calendars**

Add Repo | Watch 257 | Star 1,088 | Fork 146

Code | Issues 23 | Pull requests 3 | Insights

React Native Calendar Components

react-native android ios calendar ui-components

327 commits | **4** branches | **34** releases | **17** contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

tautvilas committed on **GitHub** Update README.md — Latest commit d1ae754 10 days ago

demo	doc: update agenda gif	3 months ago
example	fix package version	a month ago
spec	add support for babel in tests in non-intrusive manner	3 months ago
src	Implementing minDate and maxDate Attributes in Agenda View	10 days ago
.eslintrc.js	fix(lint)	3 months ago
.gitignore	chore: migrate to RN 44	3 months ago
.npmignore	build: update .npmignore	3 months ago
.nvmrc	Use node 6.9	9 months ago
.travis.yml	build: update travis.yml	3 months ago
ISSUE_TEMPLATE.md	add issue template	10 days ago
LICENSE	chore: add license	3 months ago
README.md	Update README.md	10 days ago
package.json	1.5.6	11 days ago
pom.xml	build: add package info	9 months ago

README.md

React Native Calendars

npm v1.5.6 | build passing

This module includes various customizable react native calendar components.

The package is both **Android** and **iOS** compatible.

Try it out

You can run example module by performing these steps:

```
$ git clone git@github.com:wix/react-native-calendars.git
$ cd react-native-calendars/example
$ npm install
$ react-native run-ios
```

You can check example screens source code in example module screens

This project is compatible with Expo/CRNA (without ejecting), and the examples have been published on Expo

Installation

```
$ npm install --save react-native-calendars
```

The solution is implemented in JavaScript so no native module linking is required.

Usage

`import { Calendar, CalendarList, Agenda } from 'react-native-calendars';`

All parameters for components are optional. By default the month of current local date will be displayed.

Event handler callbacks are called with `calendar objects` like this:

```
{
  day: 1,     // day of month (1-31)
  month: 1,   // month of year (1-12)
  year: 2017, // year
  timestamp,   // UTC timestamp representing 00:00 AM of this date
  dateString: '2016-05-13' // date formatted as 'YYYY-MM-DD' string
}
```

Parameters that require date types accept YYYY-MM-DD formated datestrings, JavaScript date objects, `calendar objects` and UTC timestamps.

Calendars can be localized by adding custom locales to `LocaleConfig` object:

```
import {LocaleConfig} from 'react-native-calendars';

LocaleConfig.locales['fr'] = {
  monthNames: ['Janvier','Février','Mars','Avril','Mai','Juin','Juillet','Août','Septembre','Octobre','Nove
  monthNamesShort: ['Janv.','Févr.','Mars','Avril','Mai','Juin','Juil.','Août','Sept.','Oct.','Nov.','Déc.'
  dayNames: ['Dimanche','Lundi','Mardi','Mercredi','Jeudi','Vendredi','Samedi'],
  dayNamesShort: ['Dim.','Lun.','Mar.','Mer.','Jeu.','Ven.','Sam.']
};

LocaleConfig.defaultLocale = 'fr';
```

Calendar

Basic parameters

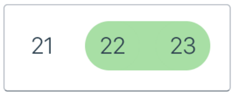
```
<Calendar
  // Initially visible month. Default = Date()
  current={'2012-03-01'}
  // Minimum date that can be selected, dates before minDate will be grayed out. Default = undefined
  minDate={'2012-05-10'}
  // Maximum date that can be selected, dates after maxDate will be grayed out. Default = undefined
  maxDate={'2012-05-30'}
  // Handler which gets executed on day press. Default = undefined
  onDayPress={(day) => {console.log('selected day', day)}}
  // Month format in calendar title. Formatting values: http://arshaw.com/xdate/#Formatting
  monthFormat={'yyyy MM'}
  // Handler which gets executed when visible month changes in calendar. Default = undefined
  onMonthChange={(month) => {console.log('month changed', month)}}
  // Hide month navigation arrows. Default = false
  hideArrows={true}
  // Replace default arrows with custom ones (direction can be 'left' or 'right')
  renderArrow={(direction) => (<Arrow />)}
  // Do not show days of other months in month page. Default = false
  hideExtraDays={true}
  // If hideArrows=false and hideExtraDays=false do not swich month when tapping on greyed out
  // day from another month that is visible in calendar page. Default = false
  disableMonthChange={true}
  // If firstDay=1 week starts from Monday. Note that dayNames and dayNamesShort should still start from Su
  firstDay={1}
/>
```

Date marking

Dot marking

```
<Calendar
  // Collection of dates that have to be marked. Default = {}
  markedDates={{
    '2012-05-16': {selected: true, marked: true},
    '2012-05-17': {marked: true},
    '2012-05-18': {disabled: true}
  }}
/>
```

Interval marking



```
<Calendar
  // Collection of dates that have to be colored in a special way. Default = {}
   markedDates={
    {'2012-05-20': [{textColor: 'green'}],
     '2012-05-22': [{startingDay: true, color: 'green'}],
     '2012-05-23': [{endingDay: true, color: 'green', textColor: 'gray'}],
     '2012-05-04': [{startingDay: true, color: 'green'}, {endingDay: true, color: 'green'}]
    }}
  // Date marking style [simple/interactive]. Default = 'simple'
  markingType={'interactive'}
/>
```

Keep in mind that different marking types are not compatible. You can use just one marking style for calendar.

Displaying data loading indicator

The loading indicator next to month name will be displayed if `<Calendar />` has `displayLoadingIndicator` property and `markedDays` collection does not have a value for every day of the month in question. When you load data for days, just set `[]` or special marking value to all days in `markedDates` collection.

Customizing look & feel

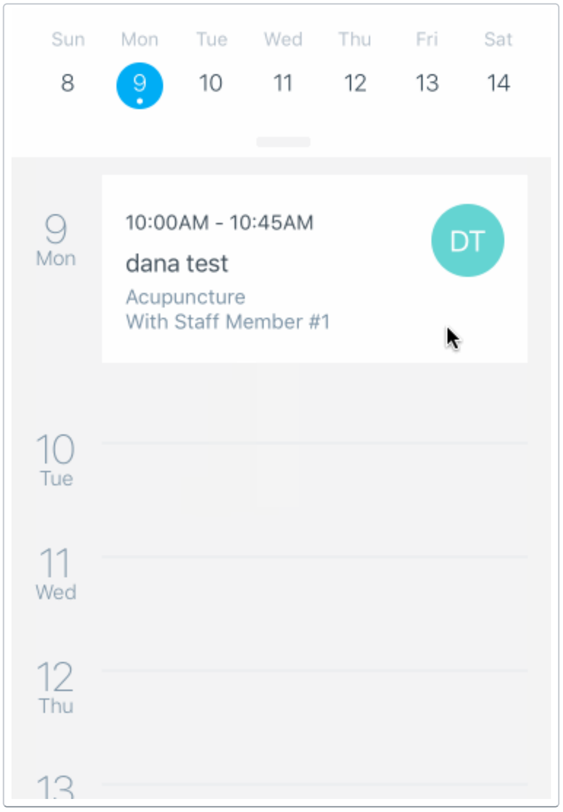
```
<Calendar
  // Specify style for calendar container element. Default = {}
  style={{
    borderWidth: 1,
    borderColor: 'gray',
    height: 350
  }}
  // Specify theme properties to override specific styles for calendar parts. Default = {}
  theme={{
    calendarBackground: '#ffffff',
    textSectionTitleColor: '#b6c1cd',
    selectedDayBackgroundColor: '#00adf5',
    selectedDayTextColor: '#ffffff',
    todayTextColor: '#00adf5',
    dayTextColor: '#2d4150',
    textDisabledColor: '#d9e1e8',
    dotColor: '#00adf5',
    selectedDotColor: '#ffffff',
    arrowColor: 'orange',
    monthTextColor: 'blue',
    textDayFontFamily: 'monospace',
    textMonthFontFamily: 'monospace',
    textDayHeaderFontFamily: 'monospace',
    textDayFontSize: 16,
    textMonthFontSize: 16,
    textDayHeaderFontSize: 16
  }}
/>
```

CalendarList

`<CalendarList />` is scrollable semi-infinite calendar composed of `<Calendar />` components. Currently it is possible to scroll 4 years back and 4 years to the future. All paramters that are available for `<Calendar />` are also available for this component. There are also some additional params that can be used:

```
<CalendarList
  // Callback which gets executed when visible months change in scroll view. Default = undefined
  onVisibleMonthsChange={(months) => {console.log('now these months are visible', months);}}
  // Max amount of months allowed to scroll to the past. Default = 50
  pastScrollRange={50}
  // Max amount of months allowed to scroll to the future. Default = 50
  futureScrollRange={50}
  // Enable or disable scrolling of calendar list
  scrollEnabled={true}
  ...calendarParams
/>
```

Agenda



An advanced agenda component that can display interactive listings for calendar day items.

```
<Agenda
  // the list of items that have to be displayed in agenda. If you want to render item as empty date
  // the value of date key kas to be an empty array []. If there exists no value for date key it is
  // considered that the date in question is not yet loaded
  items={
    {'2012-05-22': [{text: 'item 1 - any js object'}],
     '2012-05-23': [{text: 'item 2 - any js object'}],
     '2012-05-24': [],
     '2012-05-25': [{text: 'item 3 - any js object'},{text: 'any js object'}],
    }}
  // callback that gets called when items for a certain month should be loaded (month became visible)
  loadItemsForMonth={(month) => {console.log('trigger items loading')}}
  // callback that gets called on day press
  onDayPress={(day)=>{console.log('day pressed')}}
  // callback that gets called when day changes while scrolling agenda list
  onDayChange={(day)=>{console.log('day changed')}}
  // initially selected day
  selected={'2012-05-16'}
  // Minimum date that can be selected, dates before minDate will be grayed out. Default = undefined
  minDate={'2012-05-10'}
  // Maximum date that can be selected, dates after maxDate will be grayed out. Default = undefined
  maxDate={'2012-05-30'}
  // specify how each item should be rendered in agenda
  renderItem={(item, firstItemInDay) => {return (<View />);}}
  // specify how each date should be rendered. day can be undefined if the item is not first in that day.
  renderDay={(day, item) => {return (<View />);}}
  // specify how empty date content with no items should be rendered
  renderEmptyDate={() => {return (<View />);}}
  // specify your item comparison function for increased performance
  rowHasChanged={(r1, r2) => {return r1.text !== r2.text}}
  // Hide knob button. Default = false
  hideKnob={true}
  // agenda theme
  theme={{
    ...calendarTheme,
    agendaDayTextColor: 'yellow',
    agendaDayNumColor: 'green',
    agendaTodayColor: 'red'
  }}
```

```
    // agenda container style
    style={{}}
  />
```

Authors

- Tautvilas Mecinskas - Initial code - @tautvilas
- Katrin Zotchev - Initial design - @katrin_zot

See also the list of contributors who participated in this project.

Contributing

Pull requests are welcome. `npm run test` and `npm run lint` before push.