react-native-tvos /
react-native-tvos

Code 12 Issues Pulls Discussions Actions 2 Wiki 67 Releases

React Native repo with additions for Apple TV and Android TV support.
https://douglowder.github.io/react-native-apple-tv/

CC-BY-4.0 license

Code of conduct

657 stars 116 forks 28 watching Activity

Public repository

tvos-v0.72.4

Branches Tags

douglowder Bump version number (0.72.4-0) … 2 days ago 27,215

View code

react-native-tvos

Apple TV and Android TV support for React Native are maintained here and in the corresponding `react-native-tvos` NPM package, and not in the core repo. This is a full fork of the main repository, with only the changes needed to support Apple TV and Android TV.

Releases of `react-native-tvos` will be based on a public release of `react-native` ; e.g. the 0.72.4-0 release of this package will be derived from the 0.72.4 release of `react-native` . All releases of this repo will follow the 0.xx.x-y format, where x digits are from a specific RN core release, and y represents the additional versioning from this repo.

Releases will be published on npmjs.org and you may find the latest release version here: https://www.npmjs.com/package/react-native-tvos?activeTab=versions or use the tag `@latest`

You will find the relevant tvOS support and maintenance within the branches marked `tvos-v0.xx.x`;

To build your project for Apple TV, you should change your `package.json` imports to import `react-native` as follows, so that this package is used instead of the core react-native package.

```
"react-native": "npm:react-native-tvos@latest",
```

You cannot use this package and the core react-native package simultaneously in a project.

Hermes JS support

As of the 0.71 release, Hermes is fully working on both Apple TV and Android TV, and is enabled by default.

React Native new architecture (Fabric) support

- *Apple TV*: Modify your app's Podfile to set the `:fabric_enabled` value to `true` in both iOS and tvOS targets. After that, run `pod install` to pick up the additional pods needed for the new architecture. Some components (TVTextScrollView, TabBarIOS) have not been reimplemented in the new architecture so they will show up as an "unimplemented component".
- *Android TV*: To enable Fabric, modify `android/gradle.properties` in your app and set `newArchEnabled=true`, then rebuild your app.

Typescript

README.md

A minimal Typescript starter template can be used to start a new project using the community react-native CLI (see below for more information on the CLI).

```
react-native init TestApp --template=react-native-template-typescript-tv
```

General support for TV

TV device support has been implemented with the intention of making existing React Native applications "just work" on TV, with few or no changes needed in the JavaScript code for the applications.

The RNTester app supports Apple TV and Android TV. In this repo, `RNTester/Podfile` and `RNTester/RNTesterPods.xcodeproj` have been modified to work for tvOS. Run `pod install`, then open `RNTesterPods.xcworkspace` and build.

You should also install `yarn` globally, as it should be used instead of `npm` for working in React Native projects.

Build changes

- *Native layer for Apple TV*: React Native Xcode projects all now have Apple TV build targets, with names ending in the string '-tvOS'.
- *Maven artifacts for Android TV*: In 0.71, the React Native Android prebuilt archives are published to Maven instead of being included in the NPM. We are following the same model, except that the Maven artifacts will be in group `io.github.react-native-tvos` instead of `com.facebook.react`. The `@react-native/gradle-plugin` module has been upgraded so that the Android dependencies will be detected correctly during build.

New project creation

> *Pitfall:* Make sure you do not globally install `react-native` or `react-native-tvos`. You should only install `@react-native-community/cli` to use the commands below. If you have done this the wrong way, you may get error messages like `ld: library not found for -lPods-TestApp-tvOS`.

Creating a new project that uses this package is done using the react-native CLI. New projects created this way will automatically have properly configured Apple TV targets created in their XCode projects. To use this NPM package in a new project, you can reference it as in the following example using the React Native community CLI:

```
# Make sure you have the CLI installed globally (this only needs to be done on
npm install -g @react-native-community/cli
# Init an app called 'TestApp', note that you must not be in a node module (director
react-native init TestApp --template=react-native-tvos@latest
# Now start the app in the tvOS Simulator - this will only work on a macOS machine
```

```
cd TestApp && react-native run-ios  --simulator "Apple TV" --scheme "TestApp-tvOS"
# The Expo CLI may also be used -- see below for more details
```

Expo CLI support

To run Apple TV (and Android TV) targets from the command line, it is now possible to use the Expo CLI, using the following steps:

- In your app, install the required Expo module: `yarn add expo`
- Add a file `react-native.config.js` at the top level of your app directory, with these contents.
- Then an Apple TV target can be run: `npx expo run:ios --scheme MyApp-tvOS --device "Apple TV"`
- To run Android TV: `npx expo run:android`

The Expo dependency and `react-native.config.js` are included in the new app template.

See this document for more details on Expo CLI functionality. (Note that many of these features require that Expo SDK modules be built into your app, which is not yet supported on Apple TV.)

Code changes

- *JavaScript layer*: Support for TV has been added to the `Platform` React Native API.

```
var Platform = require('Platform');
var running_on_tv = Platform.isTV;

// If you want to be more specific and only detect devices running tvOS
// (but no Android TV devices) you can use:
var running_on_apple_tv = Platform.isTVOS;
```

- *Common codebase for iOS and tvOS*: Since tvOS and iOS share most Objective-C and JavaScript code in common, most documentation for iOS applies equally to tvOS. Apple TV specific changes in native code are all wrapped by the TARGET_OS_TV define. These include changes to suppress APIs that are not supported on tvOS (e.g. web views, sliders, switches, status bar, etc.), and changes to support user input from the TV remote or keyboard.

- *Common codebase for Android phone and Android TV*: Apps built for Android using this repo will run on both Android phone and Android TV. Most of the changes for TV are specific to handling focus-based navigation on a TV using the D-Pad on the remote control.
- *Access to touchable controls*: The `Touchable` mixin has code added to detect focus changes and use existing methods to style the components properly and initiate the proper actions when the view is selected using the TV remote, so `TouchableWithoutFeedback` , `TouchableHighlight` and `TouchableOpacity` will "just work" on both Apple TV and Android TV. In particular:
  - `onFocus` will be executed when the touchable view goes into focus
  - `onBlur` will be executed when the touchable view goes out of focus
  - `onPress` will be executed when the touchable view is actually selected by pressing the "select" button on the TV remote.
- *Pressable controls*: The `Pressable` API works with TV. Additional `onFocus` and `onBlur` props are provided to allow you to customize behavior when a Pressable enters or leaves focus. Similar to the `pressed` state that is true while a user is pressing the component on a touchscreen, the `focused` state will be true when it is focused on TV. `PressableExample` in RNTester has been modified appropriately.
- *TV remote/keyboard input*: Application code that needs to implement custom handling of TV remote events can create an instance of `TVEventHandler` and listen for these events. For a more convenient API, we provide `useTVEventHandler` .

```
import { TVEventHandler, useTVEventHandler } from 'react-native';

// Functional component

const TVEventHandlerView: () => React.Node = () => {
  const [lastEventType, setLastEventType] = React.useState('');

  const myTVEventHandler = evt => {
    setLastEventType(evt.eventType);
  };

  useTVEventHandler(myTVEventHandler);

  return (
    <View>
      <TouchableOpacity onPress={() => {}}>
        <Text>
          This example enables an instance of TVEventHandler to show the last
          event detected from the Apple TV Siri remote or from a keyboard.
        </Text>
      </TouchableOpacity>
      <Text style={{color: 'blue'}}>{lastEventType}</Text>
    </View>
  );

};

// Class based component

class Game2048 extends React.Component {
  _tvEventHandler: any;

  _enableTVEventHandler() {
    this._tvEventHandler = new TVEventHandler();
    this._tvEventHandler.enable(this, function(cmp, evt) {
      if (evt && evt.eventType === 'right') {
        cmp.setState({board: cmp.state.board.move(2)});
      } else if(evt && evt.eventType === 'up') {
        cmp.setState({board: cmp.state.board.move(1)});
      } else if(evt && evt.eventType === 'left') {
        cmp.setState({board: cmp.state.board.move(0)});
      } else if(evt && evt.eventType === 'down') {
        cmp.setState({board: cmp.state.board.move(3)});
      } else if(evt && evt.eventType === 'playPause') {
        cmp.restartGame();
      }
    });
  }

  _disableTVEventHandler() {
    if (this._tvEventHandler) {
      this._tvEventHandler.disable();
      delete this._tvEventHandler;
    }
  }

  componentDidMount() {
    this._enableTVEventHandler();
  }

  componentWillUnmount() {
    this._disableTVEventHandler();
  }
```

- *Flipper*: We do not support Flipper.

- *LogBox*: The new LogBox error/warning display (which replaced YellowBox in 0.63) is working as expected on TV platforms, after a few adjustments to make the controls accessible to the focus engine.
- *Dev Menu support*: On the Apple TV simulator, cmd-D will bring up the developer menu, just like on iOS. To bring it up on a real Apple TV device, make a long press on the play/pause button on the remote. (Please do not shake the Apple TV device, that will not work :)). Android TV dev menu behavior is the same as on Android phone.
- *TV remote animations on Apple TV*: `RCTTVView` native code implements Apple-recommended parallax animations to help guide the eye as the user navigates through views. The animations can be disabled or adjusted with new optional view properties.
- *Back navigation with the TV remote menu button*: The `BackHandler` component, originally written to support the Android back button, now also supports back navigation on the Apple TV using the menu button or '<' button on the Apple TV remote, and the back button as usual on Android TV remote.
- *TVEventControl for AppleTV*: (Formerly "TVMenuControl") This module provides methods to enable and disable features on the Apple TV Siri remote:
  - `enableTVMenuKey` / `disableTVMenuKey` : Method to enable and disable the menu key gesture recognizer, in order to fix an issue with Apple's guidelines for menu key navigation (see facebook/react-native#18930). The `RNTester` app uses these methods to implement correct menu key behavior for back navigation.
  - `enableTVPanGesture` / `disableTVPanGesture` : Methods to enable and disable detection of finger touches that pan across the touch surface of the Siri remote. See `TVEventHandlerExample` in the `RNTester` app for a demo.
  - `enableGestureHandlersCancelTouches` / `disableGestureHandlersCancelTouches` : Methods to turn on and turn off cancellation of touches by the gesture handlers in `RCTTVRemoteHandler` (see #366). Cancellation of touches is turned on (enabled) by default in 0.69 and earlier releases.
- *TVFocusGuideView*: This component provides support for Apple's `UIFocusGuide` API and is implemented in the same way for Android TV, to help ensure that focusable controls can be navigated to, even if they are not directly in line with other controls. An example is provided in `RNTester` that shows two different ways of using this component.

Prop	Value	Description
destinations	any[]?	Array of `Component` s to register as destinations of the FocusGuideView
autoFocus	boolean?	If true, `TVFocusGuide` will automatically manage focus for you. It will redirect the focus to the first focusable child on the first visit. It also remembers the last focused child and redirects the focus to it on the subsequent visits. `destinations` prop takes precedence over this prop when used together.
trapFocus* (Up, Down, Left, Right)	Prevents focus escaping from the container for the given directions.	

More information on the focus handling improvements above can be found in this article.

- *Next Focus Direction*: the props `nextFocus*` on `View` should work as expected on iOS too (previously android only). One caveat is that if there is no focusable in the `nextFocusable*` direction next to the starting view, iOS doesn't check if we want to override the destination.
- *TVTextScrollView*: On Apple TV, a ScrollView will not scroll unless there are focusable items inside it or above/below it. This component wraps ScrollView and uses tvOS-specific native code to allow scrolling using swipe gestures from the remote control.

Releases 67

0.72.4-0 Latest
2 days ago

Used by 51

+ 43

Contributors 2,437

● Java 28.4% ● C++ 26.0% ● JavaScript 14.0% ● Objective-C 12.3% ● Objective-C++ 9.3%
● Starlark 3.1% ● Other 6.9%

Report repository