react-player-custom TS

1.11.14 • Public • Published 2 years ago

Readme Explore BETA 3 Dependencies 0 Dependents 15 Versions

ReactPlayer

npm v2.9.0 build passing dependencies repo or path not found or david internal error coverage 80% donate PayPal

A React component for playing a variety of URLs, including file paths, YouTube, Facebook, Twitch, SoundCloud, Streamable, Vimeo, Wistia, Mixcloud, and DailyMotion. Not using React? No problem.

Usage

```
npm install react-player --save
# or
yarn add react-player
```

```
import React, { Component } from 'react'
import ReactPlayer from 'react-player'

class App extends Component {
  render () {
    return <ReactPlayer url='https://www.youtube.com/watch?v=ysz5S6PUM-U' playing />
  }
}
```

Demo page: `https://cookpete.com/react-player`

The component parses a URL and loads in the appropriate markup and external SDKs to play media from various sources. Props can be passed in to control playback and react to events such as buffering or media ending. See the demo source for a full example.

For platforms like Meteor without direct use of `npm` modules, a minified version of `ReactPlayer` is located in `dist` after installing. To generate this file yourself, checkout the repo and run `npm run build:dist`.

Polyfills

- If you are using `npm` and need to support browsers without `Promise` you will need a `Promise` polyfill.
- To support IE11 you will need to use `babel-polyfill` or a similar ES2015+ polyfill.

Autoplay

As of Chrome 66, videos must be `muted` in order to play automatically. Some players, like Facebook, cannot be unmuted until the user interacts with the video, so you may want to enable `controls` to allow users to unmute videos themselves.

Props

Prop	Description	Default
`url`	The url of a video or song to play ◦ Can be an array or `MediaStream` object	
`playing`	Set to `true` or `false` to pause or play the media	`false`
`loop`	Set to `true` or `false` to loop the media	`false`
`controls`	Set to `true` or `false` to display native player controls ◦ Vimeo, Twitch and Wistia player will always display controls	`false`
`light`	Set to `true` to show just the video thumbnail, which loads the full player on click ◦ Pass in an image URL to override the preview image	`false`

Install

```
> npm i react-player-custom
```

Repository

github.com/CookPete/react-player

Homepage

github.com/CookPete/react-player

Weekly Downloads

63

Version	License
1.11.14	MIT

Unpacked Size	Total Files
1.45 MB	31

Issues	Pull Requests
115	16

Last publish

2 years ago

Collaborators

>_ Try on RunKit

Report malware

`volume`	Set the volume of the player, between `0` and `1` ◦ `null` uses default volume on all players #357	`null`
`muted`	Mutes the player ◦ Only works if `volume` is set	`false`
`playbackRate`	Set the playback rate of the player ◦ Only supported by YouTube, Wistia, and file paths	`1`
`width`	Set the width of the player	`640px`
`height`	Set the height of the player	`360px`
`style`	Add inline styles to the root element	`{}`
`progressInterval`	The time between `onProgress` callbacks, in milliseconds	`1000`
`playsinline`	Applies the `playsinline` attribute where supported	`false`
`pip`	Set to `true` or `false` to enable or disable picture-in-picture mode	`false`
`wrapper`	Element or component to use as the container element	`div`
`config`	Override options for the various players, see config prop	

Callback props

Callback props take a function that gets fired on various player events:

Prop	Description
`onReady`	Called when media is loaded and ready to play. If `playing` is set to `true`, media will play immediately
`onStart`	Called when media starts playing
`onPlay`	Called when media starts or resumes playing after pausing or buffering
`onProgress`	Callback containing `played` and `loaded` progress as a fraction, and `playedSeconds` and `loadedSeconds` in seconds ◦ eg `{ played: 0.12, playedSeconds: 11.3, loaded: 0.34, loadedSeconds: 16.7 }`
`onDuration`	Callback containing duration of the media, in seconds
`onPause`	Called when media is paused
`onBuffer`	Called when media starts buffering
`onBufferEnd`	Called when media has finished buffering ◦ Works for files, YouTube and Facebook
`onSeek`	Called when media seeks with `seconds` parameter
`onEnded`	Called when media finishes playing ◦ Does not fire when `loop` is set to `true`
`onError`	Called when an error occurs whilst attempting to play media
`onEnablePIP`	Called when picture-in-picture mode is enabled
`onDisablePIP`	Called when picture-in-picture mode is disabled

Config prop

As of version `0.24`, there is a single `config` prop to override the settings for the various players. If you are migrating from an earlier version, you must move all the old config props inside `config`:

```
<ReactPlayer
  url={url}
  config={{
    youtube: {
      playerVars: { showinfo: 1 }
    },
    facebook: {
      appId: '12345'
    }
```

```
  }}
/>
```

The old style **config props** still work but will produce a console warning:

```
<ReactPlayer
  url={url}
  youtubeConfig={{ playerVars: { showinfo: 1 } }}
  facebookConfig={{ appId: '12345' }}
/>
```

Settings for each player live under different keys:

Key	Options
`youtube`	`playerVars` : Override the default player vars `embedOptions` : Override the default embed options `preload` : Used for preloading
`facebook`	`appId` : Your own Facebook app ID
`soundcloud`	`options` : Override the default player options `preload` : Used for preloading
`vimeo`	`playerOptions` : Override the default params `preload` : Used for preloading
`wistia`	`options` : Override the default player options
`mixcloud`	`options` : Override the default player options
`dailymotion`	`params` : Override the default player vars `preload` : Used for preloading
`twitch`	`options` : Override the default player options

`file`	`attributes` : Apply element attributes
	`forceVideo` : Always render a `<video>` element
	`forceAudio` : Always render an `<audio>` element
	`forceHLS` : Use hls.js for HLS streams
	`forceDASH` : Always use dash.js for DASH streams
	`hlsOptions` : Override the default `hls.js` options
	`hlsVersion` : Override the `hls.js` version loaded from `cdnjs` , default: `0.10.1`
	`dashVersion` : Override the `dash.js` version loaded from `cdnjs` , default: `2.9.2`

Preloading

When `preload` is set to `true` for players that support it, a short, silent video is played in the background when `ReactPlayer` first mounts. This fixes a **bug** where videos would not play when loaded in a background browser tab.

Methods

Static Methods

Method	Description
`ReactPlayer.canPlay(url)`	Determine if a URL can be played. This does *not* detect media that is unplayable due to privacy settings, streaming permissions, etc. In that case, the `onError` prop will be invoked after attemping to play. Any URL that does not match any patterns will fall back to a native HTML5 media player.
`ReactPlayer.canEnablePiP(url)`	Determine if a URL can be played in picture-in-picture mode
`ReactPlayer.addCustomPlayer(CustomPlayer)`	Add a custom player. See Adding custom players
`ReactPlayer.removeCustomPlayers()`	Remove any players that have been added

	using `addCustomPlayer()`

Instance Methods

Use **`ref`** to call instance methods on the player. See **the demo app** for an example of this.

Method	Description
`seekTo(amount, type)`	Seek to the given number of seconds, or fraction if `amount` is between `0` and `1` ◦ `type` parameter lets you specify `'seconds'` or `'fraction'` to override default behaviour
`getCurrentTime()`	Returns the number of seconds that have been played ◦ Returns `null` if unavailable
`getSecondsLoaded()`	Returns the number of seconds that have been loaded ◦ Returns `null` if unavailable or unsupported
`getDuration()`	Returns the duration (in seconds) of the currently playing media ◦ Returns `null` if duration is unavailable
`getInternalPlayer()`	Returns the internal player of whatever is currently playing ◦ eg the YouTube player instance, or the `<video>` element when playing a video file ◦ Use `getInternalPlayer('hls')` to get the hls.js player ◦ Use `getInternalPlayer('dash')` to get the dash.js player ◦ Returns `null` if the internal player is unavailable

Advanced Usage

Light player

The `light` prop will render a video thumbnail with simple play icon, and only load the full player once a user has interacted with the image. **Noembed** is used to fetch thumbnails for a video URL. Note that automatic thumbnail fetching for Facebook, Wistia, Mixcloud and file URLs are not supported, and ongoing support for other URLs is not guaranteed.

If you want to pass in your own thumbnail to use, set `light` to the image URL rather than `true`.

The styles for the preview image and play icon can be overridden by targeting the CSS classes `react-player__preview`, `react-player__shadow` and `react-player__play-icon`.

Responsive player

Set `width` and `height` to `100%` and wrap the player in a **fixed aspect ratio box** to get a responsive player:

```
class ResponsivePlayer extends Component {
  render () {
    return (
      <div className='player-wrapper'>
        <ReactPlayer
          className='react-player'
          url='https://www.youtube.com/watch?v=ysz5S6PUM-U'
          width='100%'
          height='100%'
        />
      </div>
    )
  }
}
```

```
.player-wrapper {
  position: relative;
  padding-top: 56.25% /* Player ratio: 100 / (1280 / 720) */
}

.react-player {
  position: absolute;
  top: 0;
  left: 0;
}
```

See `jsFiddle` example

Single player imports

If you are only ever playing a single type of URL, you can import individual players to keep your bundle size down:

```
import YouTubePlayer from 'react-player/lib/players/YouTube'

<YouTubePlayer
  url='https://www.youtube.com/watch?v=d46Azg3Pm4c'
  playing
  controls
  // Other ReactPlayer props will work here
/>
```

See a list of available players here.

Standalone player

If you aren't using React, you can still render a player using the standalone library:

```
<script src='https://unpkg.com/react-player/dist/ReactPlayer.standalone.js'></script>
<script>
  const container = document.getElementById('container')
  const url = 'https://www.youtube.com/watch?v=d46Azg3Pm4c'

  renderReactPlayer(container, { url, playing: true })

  function pausePlayer () {
    renderReactPlayer(container, { url, playing: false })
  }
</script>
```

See `jsFiddle` example

Adding custom players

If you have your own player that is compatible with ReactPlayer's internal architecture, you can add it using `addCustomPlayer`:

```
import YourOwnPlayer from './somewhere';
ReactPlayer.addCustomPlayer(YourOwnPlayer);
```

Use `removeCustomPlayers` to clear all custom players:

```
ReactPlayer.removeCustomPlayers();
```

It is your responsibility to ensure that custom players keep up with any internal changes to ReactPlayer in later versions.

Using Bower

```
bower install react-player --save
```

```
<script src='bower_components/react/react.js'></script>
<script src='bower_components/react/react-dom.js'></script>
<script src='bower_components/react-player/dist/ReactPlayer.js'></script>
<script>
  ReactDOM.render(
    <ReactPlayer url='https://www.youtube.com/watch?v=d46Azg3Pm4c' playing />,
    document.getElementById('container')
  )
</script>
```

Mobile considerations

Due to various restrictions, `ReactPlayer` is not guaranteed to function properly on mobile devices. The **YouTube player documentation**, for example, explains that **certain mobile browsers require user interaction** before playing:

> The HTML5 `<video>` element, in certain mobile browsers (such as Chrome and Safari), only allows playback to take place if it's initiated by a user interaction (such as tapping on the player).

Multiple Sources and Tracks

When playing file paths, an array of sources can be passed to the `url` prop to render multiple `<source>` tags.

```
<ReactPlayer playing url={['foo.webm', 'foo.ogg']} />
```

You can also specify a `type` for each source by using objects with `src` and `type` properties.

```
<ReactPlayer
  playing
  url={[
    {src: 'foo.webm', type: 'video/webm'},
    {src: 'foo.ogg', type: 'video/ogg'}
  ]}
/>
```

`<track>` elements for subtitles can be added using `fileConfig`:

```
<ReactPlayer
  playing
  url='foo.webm'
  config={{ file: {
    tracks: [
      {kind: 'subtitles', src: 'subs/subtitles.en.vtt', srcLang: 'en', default: true},
      {kind: 'subtitles', src: 'subs/subtitles.ja.vtt', srcLang: 'ja'},
      {kind: 'subtitles', src: 'subs/subtitles.de.vtt', srcLang: 'de'}
    ]
  }}}
/>
```

Supported media

- YouTube videos use the YouTube iFrame Player API
- Facebook videos use the Facebook Embedded Video Player API
- SoundCloud tracks use the SoundCloud Widget API
- Streamable videos use `Player.js`
- Vidme videos are no longer supported
- Vimeo videos use the Vimeo Player API
- Wistia videos use the Wistia Player API
- Twitch videos use the Twitch Interactive Frames API
- DailyMotion videos use the DailyMotion Player API
- Supported file types are playing using `<video>` or `<audio>` elements
    - HLS streams are played using `hls.js`
    - DASH streams are played using `dash.js`

Contributing

See the **contribution guidelines** before creating a pull request.

Thanks

- Many thanks to Kostya Luchankin for help overhauling the player inheritance patterns.
- Thanks to anyone who has contributed.

Keywords

react **media** **player** **video** **audio** **youtube** **facebook** **twitch** **soundcloud** **streamable** **vimeo** **wistia** **dailymotion** **hls** **dash** **react-component**