single-spa-react

single-spa-react is a helper library that helps implement [single-spa registered application](https://single-spa.js.org/docs/configuration#registering-applications) [lifecycle functions](https://single-spa.js.org/docs/building-applications#registered-application-lifecycle) (bootstrap, mount and unmount) for use with [React](https://reactjs.org/). Check out the [single-spa-react github](https://github.com/single-spa/single-spa-react).

single-spa- React是一个帮助实现[single-spa注册应用程序生命周期功能](https://single-spa.js.org/docs/configuration#registering-applications)(初始化、挂载和卸载)的帮助库，可与[React](https://reactjs.org/)一起使用。请查看单[spa-react github](https://github.com/single-spa/single-spa-react)。

##### 安装说明

[#](https://zh-hans.single-spa.js.org/docs/ecosystem-react#installation)Installation

npm install --save single-spa-react

# Or

yarn add single-spa-react

Alternatively, you can use single-spa-react by adding

或者，您也可以通过添加来使用single-spa-react

<script src="https://unpkg.com/single-spa-react"></script>

 and accessing the singleSpaReact global variable.

并访问singleSpaReact全局变量。

##### 快速开始

[#](https://zh-hans.single-spa.js.org/docs/ecosystem-react#quickstart)Quickstart

Your bundler's "entry file" should look like this, which allows your application to be downloaded as an in-browser ES module.

bundler的“入口文件”应该是这样的，它允许以浏览器内的ES模块的形式下载应用程序。

import React from 'react';

import ReactDOM from 'react-dom';

import rootComponent from './path-to-root-component.js';

// Note that SingleSpaContext is a react@16.3 (if available) context that provides the singleSpa props

//注意，SingleSpaContext是一个react@16.3(如果可用)上下文，它提供了singleSpa 属性

import singleSpaReact, {SingleSpaContext} from 'single-spa-react';

const reactLifecycles = singleSpaReact({

React,

ReactDOM,

rootComponent,

});

export const bootstrap = reactLifecycles.bootstrap;

export const mount = reactLifecycles.mount;

export const unmount = reactLifecycles.unmount;

##### 参数选项

[#](https://zh-hans.single-spa.js.org/docs/ecosystem-react#options)Options

All options are passed to single-spa-react via the `opts` parameter when calling

`singleSpaReact(opts)`. The following options are available:

1、React: (required) The main React object, which is generally either exposed onto the window or is available via `require('react')` 、`import React from 'react'`.

2、ReactDOM: (required) The main ReactDOMbject, which is available via `require('react-dom')`、`import ReactDOM from 'react-dom'`.

3、rootComponent: (required) The top level React component which will be rendered. Can be omitted only if `loadRootComponent` is provided.

4、loadRootComponent : (optional) A loading function that takes [custom single-spa props](https://single-spa.js.org/docs/building-applications/#custom-props) and returns a promise that resolves with the parcel. This takes the place of the `rootComponent`  opt, when provided. It is intended to help people who want to lazy load the source code for their root component. The source code will be lazy loaded during the bootstrap lifecycle.

5、suppressComponentDidCatchWarning: (optional) A boolean that indicates if single-spa-react should warn when the rootComponent does not implement componentDidCatch. Defaults to false.

6、domElementGetter: (optional) A function that takes in no arguments and returns a DOMElement. This dom element is where the React application will be bootstrapped, mounted, and unmounted. Note that this opt can be omitted. When omitted, the `domElementGetter`or `domElement` [custom single-spa props](https://single-spa.js.org/docs/building-applications/#custom-props) are used. To use those, do

singleSpa.registerApplication(name, app, activityFn, {

domElementGetter: function() {...}

})

 or

singleSpa.registerApplication(name, app, activityFn, {

domElement: document.getElementById(...)

})

If no dom element can be found through any of those methods, then a container div will be created and appended to document.body, by default.

7、parcelCanUpdate: (optional) A boolean that controls whether an update lifecycle will be created for the returned parcel. Note that option does not impact single-spa applications, but only parcels. It is true by default.

8、renderType: (optional) ENUM of one of the following: [ 'render', 'hydrate', 'createRoot', 'createBlockingRoot' ]. Defaults to 'render'. Allows you to choose which ReactDOM render method you want to use for your application.

在调用singleSpaReact(opts)时，所有选项都通过opts参数传递给single-spa-react。以下是可供选择的方案:

React:(required)主React对象，它通常要么暴露在窗口上，要么通过require(‘React’)从‘React’导入React。

ReactDOM:(required)主ReactDOMbject，可通过require('react-dom')从'react-dom'导入ReactDOM。

rootComponent:(必选)将要呈现的顶级React组件。只有提供loadRootComponent时才能省略。

loadRootComponent:(可选)一个装载函数，它接受定制的单spa道具，并返回一个承诺，该承诺将与包裹一起解析。在提供根组件opt时，它将代替它。它的目的是帮助那些想要延迟加载根组件源代码的人。源代码将在引导生命周期中延迟加载。

(可选)一个布尔值，指示当rootComponent没有实现componentDidCatch时，单spa-react是否应该发出警告。默认值为false。

domElementGetter:(可选的)不接受参数并返回一个DOMElement的函数。这个dom元素将引导、装载和卸载React应用程序。注意，这个opt可以省略。省略时，将使用domElementGetter或domElementcustom单spa道具。要使用这些，使用singleSpa。registerApplication(name, app, activityFn， {domElementGetter: function(){…}})或singleSpa。registerApplication(name, app, activityFn， {domElement: document.getElementById(…)})如果通过任何这些方法都找不到dom元素，那么将创建一个容器div并将其附加到文档中。身体,默认情况下。

parcelCanUpdate:(可选)一个布尔值，用于控制是否为返回的包创建更新生命周期。注意，该选项不影响单spa应用程序，只影响包裹。默认情况下是正确的。

renderType:(可选)以下其中之一的ENUM: ['render'， 'hydrate'，'createRoot'， 'createBlockingRoot']。默认为“呈现”。允许您选择要为应用程序使用的ReactDOM呈现方法。

##### 说明

[#](https://zh-hans.single-spa.js.org/docs/ecosystem-react#notes)Notes

For react@>=16, it is best practice to have each single-spa application's root application implement componentDidCatch in order to avoid the entire application unmounting unexpectedly when an error occurs. single-spa-react will warn to the console if componentDidCatch is not implemented. See <https://reactjs.org/blog/2017/07/26/error-handling-in-react-16.html> for more details.

对于react@>=16，最佳实践是让每个单spa应用程序的根应用程序实现componentDidCatch，以避免在发生错误时意外卸载整个应用程序。如果componentDidCatch没有实现，single-spa-react将向控制台发出警告。详见[https://reactjs.org/blog/2017/07/26/err-handling-inreact-16.html](https://reactjs.org/blog/2017/07/26/error-handling-in-react-16.html)。

#### SingleSpaContext（上下文）

[#](https://zh-hans.single-spa.js.org/docs/ecosystem-react#singlespacontext)SingleSpaContext

##### 沙箱

[#](https://zh-hans.single-spa.js.org/docs/ecosystem-react#parcels)Parcels

single-spa-react can also be used to create a single-spa parcel (instead of a single-spa application). To do so, simply call singleSpaReact() the same as for an application, except without a domElementGetter (since those are provided by the code that will mount the parcel).

Additionally, single-spa-react provides a <Parcel> component to make using framework agnostic single-spa parcels easier. This allows you to put the parcel into your render method's jsx, instead of having to implement componentDidMount and componentWillUnmount. You can use the Parcel component either by npm installing the library and importing `single-spa-react/parcel` or by adding

<script src="https://unpkg.com/single-spa-react/parcel"></script>

 and then accessing the Parcel component with `window.Parcel.default`.

single-spa-react还可以用于创建single-spa 沙箱(而不是单spa应用程序)。为此，只需调用与应用程序相同的singleSpaReact()，除了没有domElementGetter(因为这些是由将挂载包的代码提供的)。

此外，single-spa-react提供了一个<Parcel>组件，使得使用框架无关的single-spa沙箱变得更加容易。这允许您将包放入呈现方法的jsx中，而不必实现componentDidMount和componentWillUnmount。您可以通过安装npm库并导入“single-spa-react/ Parcel”或添加来使用Parcel组件

< script src = " https://unpkg.com/single-spa-react/parcel " > < /script>

然后使用“window. package .default”访问Parcel组件。

##### 沙箱属性

[#](https://zh-hans.single-spa.js.org/docs/ecosystem-react#parcel-props)Parcel props

 config(required): Either a single-spa parcel config object, or a "loading function" that returns a Promise that resolves with the parcel config.

 wrapWith(optional): A string [tagName](https://developer.mozilla.org/en-US/docs/Web/API/Element/tagName).

 `<Parcel>` will create a dom node of that type for the parcel to be mounted into. Defaults to `div`

appendTo(optional): A dom element to append the parcel to. By default, this is not needed because the parcel will be mounted in the DOM that the `<Parcel>`  component was rendered into. Useful for appending parcels to document.body or other separate parts of the dom.

mountParcel (sometimes required, sometimes not): The `mountParcel` function provided by single-spa. In general, it is preferred to use an application's mountParcel function instead of the single-spa's root mountParcel function, so that single-spa can keep track of the parent-child relationship and automatically unmount the application's parcels when the application unmounts. Note that if the `<Parcel>` component is being rendered by a single-spa application that uses single-spa-react, it is unnecessary to pass in the prop, since `<Parcel>` can get the prop from [SingleSpaContext](https://zh-hans.single-spa.js.org/docs/ecosystem-react#singlespacontext)

handleError (optional): A function that will be called with errors thrown by the parcel. If not provided, errors will be thrown on the window, by default.

parcelDidMount (optional): A function that will be called when the parcel finishes loading and mounting.

config(必需的):一个single-spa 沙箱配置对象，或者一个“加载函数”，它返回一个使用沙箱配置解决的promise。

wrapWith(可选):一个字符串标记名。' <Parcel> '将创建一个该类型的dom节点，用于装载包裹。默认为“div”

appendTo(可选):将沙箱添加到的dom元素。默认情况下，这是不需要的，因为沙箱将挂载在“< parcel >”组件被呈现到的DOM中。用于将沙箱附加到文件中。dom的主体或其他独立部分。

mountParcel(有时需要，有时不需要):single-spa提供的“mountParcel”功能。一般来说，最好使用应用程序的mountParcel函数，而不是single-spa的root mountParcel函数，这样一来，single-spa可以跟踪父-子关系，并在应用程序卸载时自动卸载应用程序的沙箱。请注意，如果使用single-spa-react的single-spa应用程序呈现' <Parcel> '组件，则无需传递该属性，因为' <Parcel> '可以从SingleSpaContext获取属性

handleError(可选):一个函数，当包抛出错误时调用它。如果没有提供，默认情况下，将在window上抛出错误。

parcelDidMount(可选):在沙箱完成加载和挂载后调用的函数。

##### 示例

[#](https://zh-hans.single-spa.js.org/docs/ecosystem-react#examples)Examples

import Parcel from 'single-spa-react/parcel'

import \* as parcelConfig from './my-parcel.js'

// config is required. The parcel will be mounted inside of the

// of a div inside of the react component tree

<Parcel

config={parcelConfig}

wrapWith="div"

handleError={err => console.error(err)}

customProp1="customPropValue2"

customProp2="customPropValue2"

/>

// If you pass in an appendTo prop, the parcel will be mounted there instead of

// to a dom node inside of the current react component tree

<Parcel

config={parcelConfig}

wrapWith="div"

appendTo={document.body}

/>

// You can also pass in a "loading function" as the config.

// The loading function must return a promise that resolves with the parcel config.

// The parcel will be mounted once the promise resolves.

<Parcel

config={() => import('./my-parcel.js')}

wrapWith="div"

/>

// If you are rendering the Parcel component from a single-spa application, you do not need to pass a mountParcel prop.

// But if you have a separate react component tree that is not rendered by single-spa-react, you \*\*must\*\* pass in a mountParcel prop

// In general, it is preferred to use an application's mountParcel function instead of the single-spa's root mountParcel function,

// so that single-spa can keep track of the parent-child relationship and automatically unmount the application's parcels when the application

// unmounts

<Parcel

mountParcel={singleSpa.mountParcel}

config={parcelConfig}

wrapWith="div"

/>

##### 创建react应用

[#](https://zh-hans.single-spa.js.org/docs/ecosystem-react#create-react-app)Create React App

See [FAQ for CRA](https://single-spa.js.org/docs/faq.html#create-react-app)