常用的single-spa配置

Using single-spa-config

The single spa root config consists of the following:

1. The root HTML file that is shared by all single-spa applications.

2. the javascript that calls `singleSpa.registerApplication()`.

Your root config exists only to start up the single-spa applications.

single-spa根配置包含以下内容:

1. 所有single-spa应用程序共享一个根HTML文件。
2. 用javascript调用' singleSpa.registerApplication() '。

根配置仅用于启动single-spa应用程序。

##### index.html文件

[#](https://zh-hans.single-spa.js.org/docs/configuration#indexhtml-file)Index.html file

See [this example root config](http://single-spa-playground.org/playground/html-file) for what a root HTML file looks like. Notice how it does not have any divs or buttons, but just calls `registerApplication()`.

You do not have to use SystemJS when using single-spa, but many examples and tutorials will encourage you to do so because it allows you to [independently deploy](https://zh-hans.single-spa.js.org/docs/separating-applications.html) your applications.

查看这个[根配置示例](http://single-spa-playground.org/playground/html-file)，了解根HTML文件是什么样子的。注意，它没有任何div或按钮，只是调用' registerApplication() '。

在使用single-spa时，您不必使用SystemJS，但是许多示例和教程将鼓励您这样做，因为它支持您[独立部署](https://zh-hans.single-spa.js.org/docs/separating-applications.html)应用程序。

##### 注册应用程序

[#](https://zh-hans.single-spa.js.org/docs/configuration#registering-applications)Registering applications

You must register [applications](https://zh-hans.single-spa.js.org/docs/building-applications) with single-spa so it knows how and when to initiate, load, mount, and unmount. Registration most commonly occurs inside of the single spa config, but does not have to. Note that if an application is registered from within another application, that no hierarchy will be maintained between the applications. Instead, the applications will be siblings and will be mounted and unmounted according to their own activity functions.

In order to register an application, call the `registerApplication(name, howToLoad, activityFunction)` api. Example:

// single-spa-config.js

import { registerApplication, start } from 'single-spa';

registerApplication("applicationName", loadingFunction, activityFunction);

start();

function loadingFunction() {

return import("src/app1/main.js");

}

function activityFunction(location) {

return location.pathname.indexOf("/app1/") === 0;

}

您必须向single-spa注册[应用程序](https://zh-hans.single-spa.js.org/docs/building-applications)，以便它知道如何以及何时启动、加载、挂载和卸载。注册最常发生在single-spa配置内部，但不一定要这样做，也有其他方式。请注意，如果一个应用程序是从另一个应用程序中注册的，则不会在应用程序之间维护层次结构。相反，应用程序将是同级的，并且将根据它们自己的激活函数进行挂载和卸载。

要注册一个应用程序，调用“registerApplication(name, howToLoad, activityFunction)”的API。例子:

// single-spa-config.js

import { registerApplication, start } from 'single-spa';

registerApplication("applicationName", loadingFunction, activityFunction);

start();

function loadingFunction() {

return import("src/app1/main.js");

}

function activityFunction(location) {

return location.pathname.indexOf("/app1/") === 0;

}

###### 应用程序命名

[#](https://zh-hans.single-spa.js.org/docs/configuration#application-name)Application name

The first argument to `registerApplication` must be a string name.

registerApplication的第一个参数必须是字符串名

###### 加载函数或应用

[#](https://zh-hans.single-spa.js.org/docs/configuration#loading-function-or-application)Loading Function or Application

The second argument to `registerApplication`  must be either a function that returns a promise [loading function](https://zh-hans.single-spa.js.org/docs/configuration#loading-function) or the resolved Application.

' registerApplication '的第二个参数必须是一个函数，该函数返回一个promise[加载函数](https://zh-hans.single-spa.js.org/docs/configuration#loading-function)或已解析的应用程序。

###### 应用程序的第二次参数

[#](https://zh-hans.single-spa.js.org/docs/configuration#application-as-second-argument)Application as second argument

Optionally for the second argument you can use the resolved Application, consisting of an object with the lifecycle methods. This allows you import the Application from another file or define applications inline in your single-spa-config

const application = {

bootstrap: () => Promise.resolve(), //bootstrap function

mount: () => Promise.resolve(), //mount function

unmount: () => Promise.resolve(), //unmount function

}

registerApplication('applicatonName', application, activityFunction)

对于第二个参数，您可以选择使用已解析的应用程序，该应用程序由一个对象和生命周期方法组成。这允许您从另一个文件导入应用程序，或者在single-spa-config中内联定义应用程序

###### 加载函数

[#](https://zh-hans.single-spa.js.org/docs/configuration#loading-function)Loading function

The second argument to `registerApplication` must be a function that returns a promise (or an ["async function"](https://ponyfoo.com/articles/understanding-javascript-async-await)). The function will be called with no arguments when it's time to load the application for the first time. The returned promise must be resolved with the application. The most common implementation of a loading function is an import call: `() => import('/path/to/application.js')`

“registerApplication”的第二个参数必须是一个返回promise(或“异步函数”)的函数。当第一次加载应用程序时，该函数将在没有参数的情况下被调用。返回的promise必须与应用程序一起解析。加载函数最常见的实现是导入调用:'()=>导入('/path/to/application.js') '

###### 激活函数

[#](https://zh-hans.single-spa.js.org/docs/configuration#activity-function)Activity function

The third argument to  `registerApplication` must be a pure function, the function is provided  `window.location` as the first argument, and returns a truthy value whenever the application should be active. Most commonly, the activity function determines if an application is active by looking at `window.location` /the first param.

Another way of looking at this is that single-spa is a top-level router that has a lot of applications that have their own sub-router.

single-spa will call each application's activity function under the following scenarios:

* `hashchange`  or `popstate ` event
* `pushState` or  `replaceState` is called
* `triggerAppChange` api is called on single-spa
* Whenever the `checkActivityFunctions`  method is called

registerApplication的第三个参数必须是一个纯函数，该函数提供了`window.location`作为第一个参数，并在应用程序应该处于激活状态时返回一个truthy值。最常见的情况是，激活函数通过查看`window.location`的地址路径的第一个参数，来匹配确定应用程序是否处于活动状态。

从另一个角度来看，single-spa是一个顶级路由器，它有很多应用程序，它们都有自己的子路由器。

 single-spa会在以下场景中调用每个应用的激活函数:

● “hashchange”或“popstate”事件

● “pushState”或“replaceState”被调用时

● 在single-spa上调用“triggerAppChange”API

● 每当调用“checkActivityFunctions”方法时

## singleSpa.start()的调用时机

[#](https://zh-hans.single-spa.js.org/docs/configuration#calling-singlespastart)Calling singleSpa.start()

The `start()`[api](https://zh-hans.single-spa.js.org/docs/api#start) must be called by your single spa config in order for applications to actually be mounted. Before `start` is called, applications will be loaded, but not bootstrapped/mounted/unmounted. The reason for `start`  is to give you control over performance. For example, you may want to register applications immediately (to start downloading the code for the active ones), but not actually mount the applications until an initial AJAX request (maybe to get information about the logged in user) has been completed. In that case, the best performance is achieved by calling `registerApplication`  immediately, but calling `start` after the AJAX request is completed.

//single-spa-config.js

import { start } from 'single-spa';

/\* Calling start before registering apps means that single-spa can immediately mount apps, without

\* waiting for any initial setup of the single page app.

\*/

start();

// Register applications....

必须通过single-spa配置调用' start() ' [API](https://zh-hans.single-spa.js.org/docs/api#start)，以便真正的挂载应用程序。在调用“start”之前，将加载应用程序，但不初始化/挂载/卸载。这样使用`start`的目的是为了让你控制自己的性能。例如，您可能希望立即注册应用程序(以便开始下载活动应用程序的代码)，但在完成初始AJAX请求(可能是获取有关登录用户的信息)之前并不真正的挂载应用程序。在这种情况下，最好的性能是通过立即调用‘registerApplication’，而在AJAX请求完成后调用‘start’来实现的。

## 两个注册应用程序可以并存吗？

[#](https://zh-hans.single-spa.js.org/docs/configuration#two-registered-applications-simultaneously)Two registered applications simultaneously??

Yep, it's possible. And it's actually not that scary if you do it right. And once you do, it's really really powerful. One approach to do this is to create a `<div id="app-name"></div>`

 for each app, so that they never try to modify the same DOM at the same time.

是的,这是可能的。如果你做对了，其实也没那么可怕。一旦你这么做了，它真的非常非常强大。一种方法是创建一个' <div id="app-name"></div> '对于每个应用程序，这样它们就不会试图在同一时间修改相同的DOM。