prentJsonpfunction的作用

在 main.js 中处理 webpackJsonp 过程中还有一段特殊的逻辑：

jsonpArray = window["webpackJsonp"] = window["webpackJsonp"] || [];

var oldJsonpFunction = jsonpArray.push.bind(jsonpArray);

...

jsonpArray = jsonpArray.slice();

for(var i = 0; i < jsonpArray.length; i++) webpackJsonpCallback(jsonpArray[i]);

var parentJsonpFunction = oldJsonpFunction;

复制代码

也就是说如果之前已经存在全局的 window["webpackJsonp"] 那么在替换其 push 函数之前会将原有的 push 方法保存为 oldJsonpFunction，同时将已存在于 window["webpackJsonp"] 中的内容，一一执行 webpackJsonpCallback。并且在 webpackJsonpCallback 中也将异步加载的内容也会在 parentJsonpFunction 中同样执行一次

if(parentJsonpFunction) parentJsonpFunction(data);

复制代码

这样的同步意义何在？试想下面的场景，webpack 中多入口情况下，例如如下配置

{

entry: {

bundle1: 'bundle1.js',

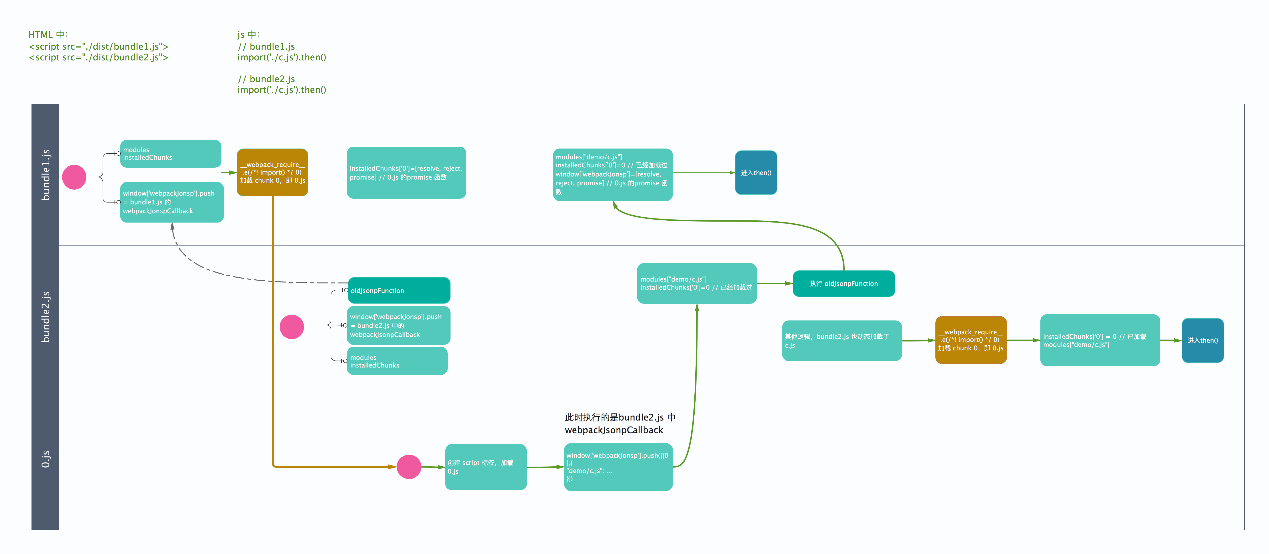
bundle2: 'bundle2.js'

}

}

复制代码

并且 bundle1 和 bundle2 中都用到了异步加载了 0.js。而且在同一个页面中同时加载了 bundle1 和 bundle2。那么由于上面的逻辑，执行的流程如下图：



通过上图可以看到，这样设计对于多入口的地方，可以将 bundle1.js 和 bundle2.js 中异步模块进行同步，这样不仅保证了 0.js 可以同时在两个文件中被引用，而且不会重复加载。