2016/10/13学习笔记

知识点：Webpack、CommonJS、模块化、ES6模块化

一、Common JS简介

规范

希望JavaScript能够在任何地方运行

对于JavaScript自身而言，它还有以下缺陷：

1. 没有模块系统
2. 标准库较少。ECMAScript仅定义了部分核心库，对于文件系统，I/O流等常见需求却没有标准的API。
3. 没有标准接口。在JavaScript中，几乎没有定义过如Web服务器或者数据库之类的标准统一接口。
4. 缺乏包管理系统。这导致JavaScript应用中基本没有自动加载和安装依赖的能力。

CommonJS规范的提出，主要是为了弥补当前JavaScript没有标准的缺陷，以达到像Python、Ruby和Java具备开发大型应用的基础能力，而不是停留在小脚本程序的阶段。他们期望那些用CommonJS API写出的应用可以具备跨宿主环境执行的能力，这样不仅可以利用JavaScript开发富客户端应用，而且可以编写以下应用

1. 服务器端JavaScript应用程序
2. 命令行工具
3. 桌面图形界面应用程序
4. 混合应用

--commonJS相当于是对浏览器端的JavaScript的API扩展，且没有纳入W3C的规范之中，并且一部分内容已被node实现

node借鉴了CommonJS的Modules规范实现了模块系统

NPM对Packages规范实现包管理

1. CommonJS中的模块

CommonJS对模块的定义主要分为3个方面

1. 模块引用
2. 模块定义
3. 模块标识
4. 模块引用 require（）方法

var math = require(‘math’);

1. 模块定义

在模块中，上下文提供require（）方法来引入外部模块。对应引入的功能，上下文提供了exports 对象用于导出当前模块的方法或者变量，并且它是唯一导出的出口。在模块中，还存在一个module 对象，它代表模块自身，而exports是module的属性。在node中，一个文件就是一个模块，将方法挂载在exports对象上作为属性即可定义导出的方式

1. 模块标识

模块标识就是传给require（）方法的参数，符合驼峰命名的字符串，或者是以.、。。开头的相对路径或者绝对路径，没有文件名后缀如.js

1. ES6中模块化

ES6模块的设计思想，是尽量的静态化，使得编译时就能确定模块的依赖关系，以及输入和输出的变量，CommonJS和AMD模块，都只能在运行时确定这些东西

ES6模块不是对象，而是通过export命令显示制定输出的代码，输入时也采用静态命令的形式

语法格式：

import { stat,exists,readFile} from ‘fs’

上面代码是从fs模块加载3个方法，其他方法不加载

浏览器使用ES6模块的语法：

<script type=”module” src=”foo.js”></script>

node的默认模块格式是commonJS，所以只能通过babel这样的转码器，在node中使用ES6模块

模块功能主要由2个命令组成：

1. import：输入其他模块提供的功能
2. export：规定模块的对外接口

一个模块就是一个独立的文件，该文件内部的所有变量，外部都无法获取，如果要外部读取模块内部的某个变量，就必须使用export关键字输出

可以输出变量（使用花括号{ }）、函数和类（class）

使用export命令定义了模块的对外接口以后，其他js文件就可以通过import命令加载这个模块。Import命令接受一个对象（{ }），里面指定要从其他模块导入的变量名。如果想为输入的变量重新取个名字，可以用as关键字。

import命令具有提升效果

模块的整体加载：上述为加载某个输出值，还可以使用整体加载，使用星号\*

export default命令：为模块指定默认输出

这样使用import引入该模块的输出方法时，可以用任意名称指向，此时不要写{ }，其实只是输出一个叫做default的变量，所以它后面不能跟变量声明语句

一个模块只能有一个默认输出，因此export default命令只能使用一次

ES6模块化和CommonJS模块化不同处：

模块加载机制不同：

CommonJS模块输出的是一个值的拷贝，而ES6模块输出的是值的引用

因此，CommonJS输出一个值后，模块

2016/10/14学习笔记

知识点：npm常用命令 babel

Material ui 安装

npm install material-ui

1、需要react-tap-event-plugin

安装react-tap-event-plugin

$ npm i --save react-tap-event-plugin

用法：

var injectTapEventPlugin = require("react-tap-event-plugin");

injectTapEventPlugin();

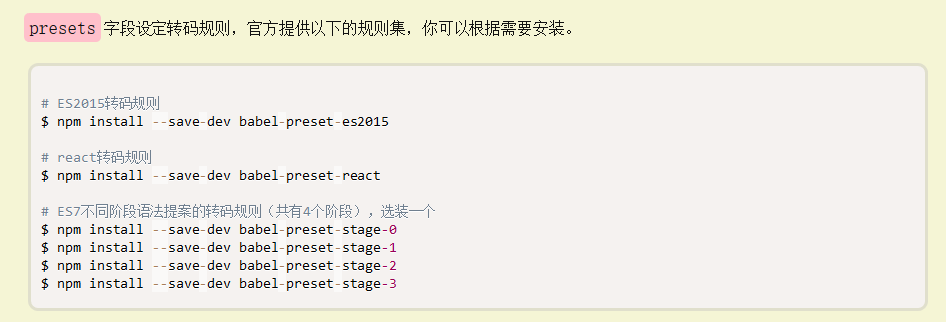
Or

import injectTapEventPlugin from 'react-tap-event-plugin';

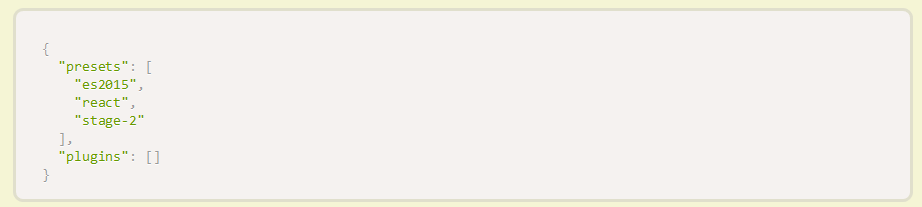
injectTapEventPlugin();

1. babel 安装stage-1

$ npm install --save-dev babel-preset-stage-1



<http://babeljs.io/docs/plugins/preset-stage-1/>



Webpack配置文件内写入：presets: ['es2015', 'react' ,'stage-1']（接着后面写）

1. Roboto字体

<link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">

font-family: 'Roboto', sans-serif;

2016/10/19学习笔记

Object.assign用于对象的合并

将源对象（source）的所有可枚举属性，复制到目标对象（target）

Object.assign（target，source1.source2）

如果目标对象与源对象有同名属性，或多个源对象有同名属性，则后面的属性会覆盖前面的属性。

Object.assign方法实行的是浅拷贝，而不是深拷贝。

如果源对象某个属性的值是对象，那么目标对象拷贝得到的是这个对象的引用。

2016/10/20学习笔记

关键字：JIRA node加载模块 理解回调函数

JIRA Concepts - Issues

Issue Types：

<https://williamoneil.atlassian.net/secure/ShowConstantsHelp.jspa?decorator=popup#IssueTypes>

Text Formatting Notation Help

<https://williamoneil.atlassian.net/secure/WikiRendererHelpAction.jspa?section=lists>

加载模块

通过require()可以加载模块，返回模块对象

模块的类型：核心模块，通过npm安装的第三方模块，本地模块

1. 加载核心模块

只能通过模块名引用。而不能通过文件路径引用，即使已经存在一个与其同名的第三方模块，也会优先加载核心模块

Var http = require(“http”);

1. 加载文件模块

提供绝对路径或者相对路径从文件系统加载非核心模块

var myModule = require(‘/home/pedo/my\_modules/my\_module’);

注意可以省略文件扩展名.js,如果没有找到这个文件，Node会在文件名后加上.js扩展名再次查找路径。

所以以下两句是等效的

var myModule = require(‘./my\_module’);

var myModule = require(‘./my\_module.js’);

1. 加载文件夹模块

var myModule = require(‘./myModuleDir’);

node会在指定的文件夹下查找模块

Node会假定该文件夹是一个包，并试图查找包定义。

如果文件夹中不存在包定义文件package.json,那么包的入口点会假定为默认值index.js

包定义包含在名为package.json的文件中

如果文件夹中存在package.json文件，那么node就会尝试解析该文件并查找main属性，将main属性当做入口点的相对路径。

{

“name”：“myModule”

“main”：“./lib/myModule.js”

}

1. 从node\_modules文件夹加载

本地目录node\_modules是npm安装模块的默认位置

1. 模块缓存

模块在首次加载时会被缓存起来

2016.10.21学习笔记

Arrow functions 箭头函数（ES6新特性，（ECMAScript2015））

Syntax语法

(param1,param2,..,paramN) => {statement}

(param1,param2,..,paramN) => expression 等价于=> {return expression}

当只有一个参数时，圆括号可以省略 theOnlyParameter => {statement}

没有参数时，圆括号必须有() => {statements}

Rest parameters

Function(a,b,...theArgs){

}

Arguments 和 the rest parameters的区别

1.Rest parameters are only the ones that haven’t been given a separate name,while the arguments object contains all arguments passed to the function;

2.The aurguments object is not a real array,while rest parameters are Array instances,meaning methods like sort,map,forEach or pop can be applied on it directly;

3.The arguments object has additional functionality specific to itself(like the callee property).

this

in strict mode, if this was not defined by the execution context, it remains undefined.

An arrow function does not create its own this context, so this has the original meaning from the enclosing context.

Arrow functions don't have their own arguments object.So in most cases rest parameters are a good alternative

Arrow functions cannot be used as constructors.

Hashchange

hashchange事件会在页面URL中的片段标识符(第一个#号开始到末尾的所有字符,包括#号)发生改变时触发.

规范：HTML5 接口：HashChangeEvent

React Router

import { Router, Route, Link } from 'react-router'

<Router>

<Route path="/" component={App}>

<Route path="about" component={About} />

<Route path="inbox" component={Inbox} />

</Route>

</Router>

路由配置

使用 IndexRoute 来设置一个默认页面

<IndexRoute component={Dashboard} />

跳转

<Redirect from="messages/:id" to="/messages/:id" />

Route 可以定义 onEnter 和 onLeave 两个 hook

onleave hook会在所有将离开的路由中触发，从最下层的子路由开始直到最外层父路由结束

onEnter hook会从最外层的父路由开始直到最下层子路由结束

History

常见的history有三种形式

createHashHistory

createBrowserHistory

creatMemoryHistory

import { browserHistory } from 'react-router'