模块化开发

在前端开发中有很多编程思想是非常不错的，而且大家也使用过一些，比如模块化编程思想，在现在的模块化编程中在前端我们常用到的是Seajs、RequireJS无论我们使用哪种都是为了方便我们使用模块化的思想，简化我们的操作，但是模块化实现方式比较多，我们不知道怎么选择而且现在标准还不一样，那我们该怎么办呢？现在有办法了，我们使用ES6进行开发，他即是JavaScript新的标准也是后期JavaScript发展的方向，同时ES6里面还实现了模块化的编程思想，在ES6中还有很多非常实用的对象方法等。虽然ES6现在看还需要转译工具进行转译，但是ES6是JavaScript后期的发展方向。现在流行的三大框架的开发基本上都是用ES6编码来实现。因此为了我们后期有高收入还是要努力学习一下，接下来我们就来看一下在ES6中使用比较频繁而且比较重要的Module语法吧。

首先我们要注意的是ES6自动使用严格模式进行开发，严格模式主要有以下限制：

* 变量必须声明后再使用
* 函数的参数不能有同名属性，否则报错
* 不能使用with语句
* 不能对只读属性赋值，否则报错
* 不能使用前缀 0 表示八进制数，否则报错
* 不能删除不可删除的属性，否则报错
* 不能删除变量delete prop，会报错，只能删除属性delete global[prop]
* eval不会在它的外层作用域引入变量
* eval和arguments不能被重新赋值
* arguments不会自动反映函数参数的变化
* 不能使用arguments.callee
* 不能使用arguments.caller
* 禁止this指向全局对象
* 不能使用fn.caller和fn.arguments获取函数调用的堆栈
* 增加了保留字（比如protected、static和interface）

关于严格模式的定义和具体的信息，这里就不过多的阐述，如有不了解的可以先去网上查一下关于严格模式的一些内容再继续学习。

## export、import的用法

在使用ES6的模块化设计的时候最常遇到的问题就是如何导入和导出属性和方法，我们在创建或者使用对应的js文件的时候我们常用的方式是使用export命令导出对应的属性和方法使用import命令导入对应的属性和方法，这两种方式基础使用我们就不多说了，用法非常简单。

**export：**

// 一次导出单个变量

export let firstName = 'zhang';

export let lastName = 'miao';

// 一次导出多个变量

let firstName = 'zhang';

let lastName = 'miao';

export {firstName, lastName};

// 一次导出一个函数

export function fn() {};

// 一次导出多个函数

function fn() {}

function fun() {}

export {fn,fun};

**import：**

// 导入对应的变量和方法

import {firstName, lastName} from './user.js';

// 修改导入的变量或函数的名称

import { lastName as name } from './user.js';

// 导入一个JS文件

import 'lodash';

## export、import特殊用法

以上为export 、import这两个命令常用的方法，但是有些特殊情况在这里还是要说一下，比如 export default 命令，是为了给用户提供方便，不用阅读文档就能加载模块，就要用到export default命令，为模块指定默认输出。

Eg：

// user-default.js

export default function () {

console.log('fn');

}

// import.js

import fn from './user-default';

fn(); // 'fn'

我们在使用import的时候也是有一些特殊的用法的，比如：import \* as fn from "fn.js";这种使用方式是将这个js中所有的方法全部赋值给fn这个函数，调用的时候使用” fn.函数名”即可。

其实，在使用import的时候还有一种特殊情况是按需加载js文件，调用代码是：import('./user.js') 这样可以在需要引入js的时候引入对应的js内容和方法。

## Es6实现反射机制

我们在js开发工程中，有的时候会涉及到一些常量的调用或者方法的调用，在我们进行开发的时候为了方便项目管理，将一些常量放置到了一个公共的js文件中，但是我们在调用的时候需要先引入这个文件到我们需要的文件中，再调用这个变量，很不方便，比如公共请求及URL地址这块的处理，我们拿这个案例来说：

// http.js 文件

import request from '@/common/request'

import \* as url from "./url"

export function postRequest (params) {

return request('post', url[params.type], {...params.data})

}

// url.js 文件

// 注册页面请求地址

export const registerUrl = '/api/register/code';

// 登录页面请求地址

export const loginUrl = '/api/register/code';

// thrd.js

this.$http.postRequest({type:'registerUrl',data:{.....}})

我们在http.js文件中添加上面的代码，主要是使用url来获取到对应的常量信息,将这个方法类暴露为公共的请求方法，使用this.$http.fn 的方式来获取到http.js文件中的方法值。在url.js中去添加上所有数据请求的常量值。在我们需要调用的文件中只需要使用this.$http.postRequest({type:'registerUrl',data:{.....}})就可以返回一个数据对象给我们，我们在新添加数据请求时只需要在url.js增加一个常量即可，而且便于后期对代码的管理。