**JS模块如何暴露与导入**

**当项目功能越来越多，代码量便也会越来越多，后期的维护难度会增大，此时在JS方面就会考虑使用模块化规范去管理维护。**

稍微介绍一下常用的模块暴露方法

**随着ES6的普及以及babel的广泛应用，越来越多的喜欢使用ES6模块导入，在ES6之前，也有遵循COMMON.JS 使用require进行模块导入**

我们通常使用的暴露有三种方式：exports module.exports export default export 经常使用的引入方式有两种：import from 和 require

**有时候记太多概念并没什么效果 用就完事了**

**我们只需记住**

1. export || export default => import
2. 复制代码

**ES6通过export 以及export default 暴露模块 通过import 导入**

1. exports || module.exports =>require
2. 复制代码

**遵循COMMON.JS的规范 进行模块化开发**

**ES6模块导入**

导出: export，export default 可以导出变量，函数，对象，文件，模块 ###在测试的时候最好是在有babel的环境下，不然有可能报错，建议在vue-cli中的APP.VUE中测试 首先先新建一个js文件

1. function add() {
2. console.log("hello");
3. }
4. var a=9
5. var obj={
6. a:3,
7. b:4,
8. c:5
9. }
10. export {add,a,obj}
11. 复制代码

也有这种写法

1. export var a = 9
2. export function fn(){}
3. 复制代码

错误写法

1. *// 报错*
2. export 1;
3. *// 报错*
4. var m = 1;
5. export m;
6. 复制代码

模块导入 import \* from ' '

1. <script>
2. import {add,a,obj} from './es6.js' =>export 暴露 导出一定要加大括号 按需导入 需要什么导什么
3. export default {
4. name: 'App',
5. created(){
6. add() =>导入之后可直接使用
7. console.log(a)
8. console.log(obj,obj.a)
9. }
10. }
11. </script>
12. 复制代码

也可以将所有内容导入对象中，通过对象来调用

1. var a={
2. fn:function(){
3. console.log(9)
4. },
5. a:9,
6. obj:{
7. a:1,
8. b:2,
9. c:3
10. }
11. }
12. export {a}
13. 复制代码

还是使用{}improt 导入

1. <script>
2. import {a} from './es6.js'
3. export default {
4. name: 'App',
5. created(){
6. a.fn()
7. }
8. }
9. </script>
10. 复制代码

**export default**

1. var a={
2. fn:function(){
3. console.log(9)
4. },
6. a:9,
7. obj:{
9. a:1,
10. b:2,
11. c:3
12. }
13. }
15. export default a =>注意 这里使用了export default a并没有加{}
16. 复制代码

导入default

1. <script>
2. import a from './es6.js' =>其余不变 取消括号
3. export default {
4. name: 'App',
5. created(){
6. a.fn()
7. }
9. }
10. </script>
11. 复制代码

区别：export default 只能存在一个 导入不需要加{ } export可以存在多个 导入一定要加{ }

1. var a=1;
2. var b=2;
3. var c=3
5. export default a
6. export {b,c}
7. 复制代码

**两者同时存在的引入方法还是那样 default 不加{ } export 加{ }**

1. import a ,{b,c} from './es6.js'
2. export default {
3. name: 'App',
4. created(){
5. console.log(a,b,c)
6. }
8. }
9. </script>
10. 复制代码

**as关键字 这种用法比较少 了解一下**

1. var a=1;
2. var b=2;
3. var c=3
5. export {a,b,c}
6. 复制代码

在导入的时候 将数据存储到新的对象名中使用

1. <script>
2. import \* as obj from './es6.js' =>注意此处没有{} 在全部数据都导入obj的时候 也就是使用通配符（\*）的时候
3. export default {
4. name: 'App',
5. created(){
6. console.log(obj.a) =>相当于把所有的数据存到了obj中
7. }
9. }
10. </script>
11. 复制代码

引入局部的时候

1. <script>
2. import {a as obj} from './es6.js'
3. export default {
4. name: 'App',
5. created(){
6. console.log(obj) =>注意此时 使用obj调用 用a 反而报错
7. }
9. }
10. </script>
11. 复制代码

在暴露的时候同样可以使用as

1. var a=1;
2. var b=2;
3. var c=3
5. export {a as default} =>很容易看懂 就是将a变成了默认暴露
6. 复制代码

既然变成了默认就可以随意引入了

1. <script>
2. import aaaaa from './es6.js' =>一旦是默认暴露就可以自由命名
3. export default {
4. name: 'App',
5. created(){
6. console.log(aaaaa)
7. }
9. }
10. </script>
11. 复制代码

全部暴露

1. var a=1;
2. var b=2;
3. var c=3
5. export {a as default,b,c}
6. 复制代码

引入又变成跟以前一样了 其实个人感觉没什么用 了解一下

1. <script>
2. import aaaaa,{b,c} from './es6.js'
3. export default {
4. name: 'App',
5. created(){
6. console.log(aaaaa,b,c)
7. }
9. }
10. </script>
11. 复制代码

**require 导入**

require 导入是比较霸道的，直接require（‘path’） 一点不跟你含糊 全部导入 类似import \* as from ‘ ’

**exports --- moudel.exports**

moudel.exports的简单用法 在新建js中

1. var a={
2. name:'node',
3. length:4
4. }
6. module.exports=a 根据export的例子很容易就能看懂了
7. 复制代码

使用require 引入

1. <template>
2. <div id="app">
3. {{value}}
4. </div>
5. </template>
7. <script>
8. var a=require('./exports.js')
9. var that=this;
10. export default {
11. name: 'App',
12. data(){
13. return{
14. value:a.name
15. }
16. }
18. }
19. </script>
20. 复制代码

刚才例子比较好理解，来一套多个数据的

1. function fn1(){
2. return '666'
3. }
4. function fn2(){
5. return '888'
6. }
7. c=9
8. module.exports={
9. fn1:fn1, =>前面的fn1 类似于 export中的 as 别名 引入 通过fn1 调用 fn1 方法 哎呀是不是很拗口
10. fn2, =>键值相同可以简写的 相信小伙伴都知道
11. c:c
12. }
13. 复制代码

导入的方式依旧粗暴

1. <template>
2. <div id="app">
4. {{value}}
5. <hr>
6. {{value1}}
7. <hr>
8. {{value2}}
10. </div>
11. </template>
13. <script>
14. var a=require('./exports.js')
15. var that=this;
16. export default {
17. name: 'App',
18. data(){
19. return{
20. value:a.c,
21. value1:a.fn1(),
22. value2:a.fn2()
23. }
24. }
26. }
27. </script>
28. 复制代码

**exports用法**

1. function fn1(){
2. return '666'
3. }
4. function fn2(){
5. return '888'
6. }
7. c=9
9. exports.fn1=fn1
10. exports.fn2=fn2
11. */\*{ 这种写法虽然不报错 但是是无效滴*
12. *exports = {*
13. *fn1,*
14. *fn2*
15. *};*
16. *}\*/*
17. 复制代码

导入方法还是不变 暴力导入

1. <template>
2. <div id="app">
3. <hr>
4. {{value1}}
5. <hr>
6. {{value2}}
8. </div>
9. </template>
11. <script>
12. var a=require('./exports.js')
13. var that=this;
14. export default {
15. name: 'App',
16. data(){
17. return{
18. value1:a.fn1(),
19. value2:a.fn2()
20. }
21. }
23. }
24. </script>
25. 复制代码

**exports module.exports小结**

其实exports只是辅助器，到最后还是会转换成module.exports  
真正暴露的是module.exports 所以还是直接使用module.exports 更好哦