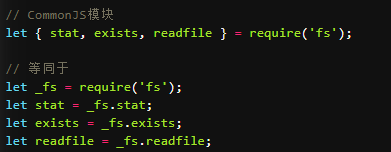
参考摘抄于：<https://es6.ruanyifeng.com/#docs/module>

**1、基本概念**

在ES6之前，社区制定了一些模块加载方案，主要有CommonJS和AMD两种。前者用于服务器（Node），后者用于浏览器。ES6模块可以在编译时就确定了各个模块之间的依赖关系以及输入输出的变量。CommonJS和AMD模块，只能在运行时确定这些东西。比如：CommonJS模块就是对象，输入时必须查找对象属性。



ES6模块不是对象，而是通过export显示输出指定的代码，再通过import命令引入。

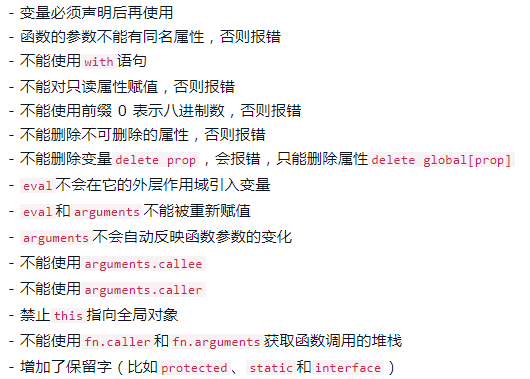


上面代码的实质是从fs模块加载了三个方法，其他方法不加载。这种加载叫做编译时加载或者静态加载，即ES6模块可以在编译时就完成模块加载，效率要比CommonJS模块高。当然这也导致了无法引用ES6模块本身，因为他并不是一个对象。

**2、严格模式**

ES6的模块自动采用严格模式，无论你是否在模块头部添加了’use strict’;

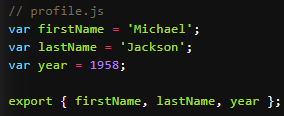
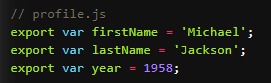
严格模式下，有以下限制：



尤其需要注意this的限制，在ES6模块中，顶层的this指向undefined，即不应该在顶层代码中使用this。

**3、export命令**

模块功能主要由两个命令构成：export和import。export命令用来规定模块的对外接口，import命令用于输入其他模块提供的功能。

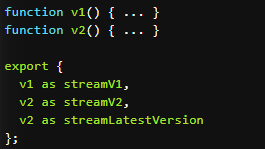


上面两种都是export的写法，二者是等价的。但是更推荐使用第二种写法，因为可以在代码尾部清楚的看到输出了哪些变量。

export不仅可以输出变量，也可以输出方法和类。

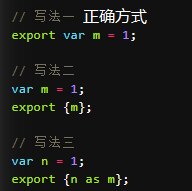


通常情况下，export输出的变量就是原本的名字，但是可以使用as关键字进行重命名。



上面代码使用as关键字重命名了v1/v2的对外接口，重命名后，v2可以使用不同的名字输出两次。

需要特别注意的是，export命令规定的是对外的接口，必须与模块中的变量一一对应。而export命令输出的必须是一个接口，如果直接输出一个值会报错。

同样function和class的输出也必须遵循这样的写法。

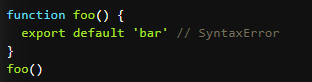


另外，export语句的输出的接口与其对应的值是动态绑定的关系，即通过该接口，可以实时获取到输出该接口的模块的内部的值。



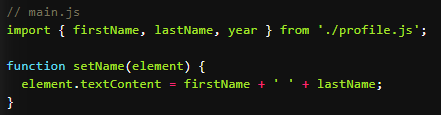
上面代码输出了变量foo，值为’bar’；500毫秒之后会变成’baz’。这一点与CommonJS不同，CommonJS输出的值的缓存，不存在动态更新。

最后，export命令可以出现在模块的任何位置，只要位于模块顶层就可以。如果处于块级作用域就会报错，import命令亦是如此。这是因为处于条件代码块中，就没法做静态优化了，违背了ES6模块的初衷。



**4、import命令**

使用export命令定义了模块的外部接口后，其他文件可以通过import命令加载这个模块。



上面代码的import命令，用于加载profile.js文件，并从中输入变量。import命令接受一对大括号，里面指定要从其他模块导入的变量名。大括号里面的变量名，必须与被导入模块（profile.js）对外接口的名称相同。使用as关键字可以为import命令导入的变量重命名。



import命令输入的变量都是只读的，因为它的本质就是输入接口。也就是说不允许在加载模块中改写接口。



但是如果输入的接口是一个对象，改写对象的属性是被允许的。但是由于这种写法很难查错，建议凡是输入的变量，都当做只读变量，不要轻易改写。

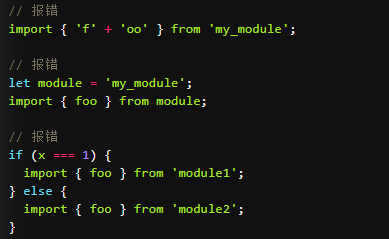


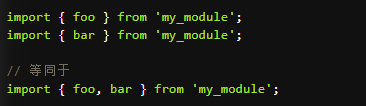
import后面的from指定模块文件的位置，可以是相对路径，也可以是绝对路径。如果不带有路径，只是一个模块名，那么必须有配置文件，告诉 JavaScript 引擎该模块的位置。



注意，import命令有提升效果，会提升到整个模块的头部首先执行。这种行为的本质在于import是在代码编译时执行的。

由于import是静态执行，所以不能使用表达式和变量，这些只有在运行时才能得到结果的语法结构。

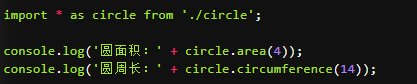


最后，import命令会执行所加载的模块；该代码会执行lodash而不输入任何值。如果多次执行同一句import命令，只会执行一次；该代码说明如果从一个模块中引入两个变量，尽管import两次，也只会合并到一次执行。

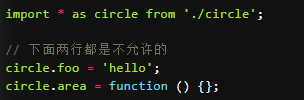
目前阶段，通过 Babel 转码，CommonJS 模块的require命令和 ES6 模块的import命令，可以写在同一个模块里面，但是最好不要这样做。因为import在静态解析阶段执行，所以它是一个模块之中最早执行的。有可能得不到想要的效果。

**5、模块的整体加载**

除了指定加载某个输出值，还可以使用整体加载，即用星号（\*）指定一个对象，所有输出值都加载在这个对象上面。

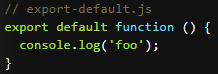


注意，模块整体加载所在的那个对象（上例是circle），应该是可以静态分析的，所以不允许运行时改变。

即在该对象上添加或修改属性都是不被允许的。

**6、export default 命令**

从前面的例子可以看出，使用import命令的时候，用户需要知道所要加载的变量名或函数名，否则无法加载。可以使用export default命令给模块设置默认输出。



上面代码的import命令，可以用任意名称指向export-default.js输出的方法，这时就不需要知道原模块输出的函数名。需要注意的是，这时import命令后面，不使用大括号。

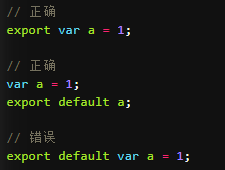
export default命令用在非匿名函数前，也是可以的。



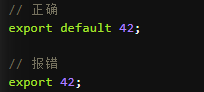
上面代码中，foo函数的函数名foo，在模块外部是无效的。加载的时候，视同匿名函数加载，即可使用import \* as XX from ‘./export-default’语句引入该模块。本质上，export default就是输出一个叫做default的变量或方法，然后系统允许你为它取任意名字。

export default命令用于指定模块的默认输出。显然，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。所以，import命令后面才可以不用加大括号，因为只可能唯一对应export default命令。

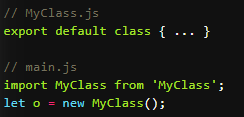
正是因为export default命令其实只是输出一个叫做default的变量，所以它后面不能跟变量声明语句。



同样地，因为export default命令的本质是将后面的值，赋给default变量，所以可以直接将一个值写在export default之后。

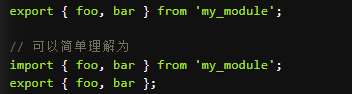


export default也可以用来输出类。



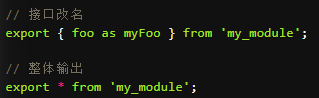
**7、export 与 import 的复合写法**

如果在一个模块之中，先输入后输出同一个模块，import语句可以与export语句写在一起。



但需要注意的是，写成一行以后，foo和bar实际上并没有被导入当前模块，只是相当于对外转发了这两个接口，导致当前模块不能直接使用foo和bar。

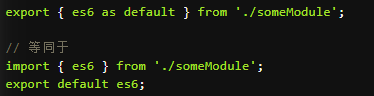
模块的接口改名和整体输出，也可以采用这种写法。



默认接口的写法如下。



具名接口改为默认接口的写法如下。



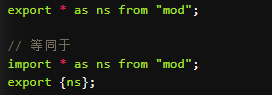
同样地，默认接口也可以改名为具名接口。



ES2020 之前，有一种import语句，没有对应的复合写法。

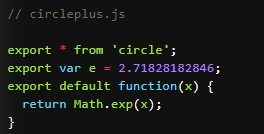


ES2020补上了这个写法。



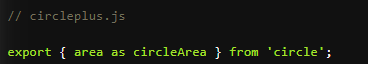
**8、模块的继承**

模块之间也可以继承。假设有一个circleplus模块，继承了circle模块。



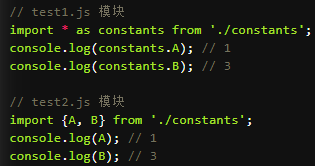
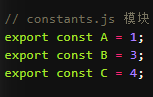
上面代码中的export \*，表示再输出circle模块的所有属性和方法。注意，export \*命令会忽略circle模块的default方法。然后，上面代码又输出了自定义的e变量和默认方法。

这时，也可以将circle的属性或方法，改名后再输出。



**9、跨模块常量**

const声明的常量只在当前代码块有效。如果想设置跨模块的常量（即跨多个文件），或者说一个值要被多个模块共享，可以采用下面的写法。



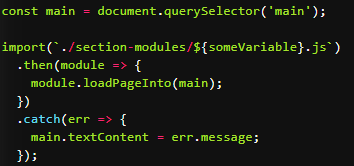
**10、import()**

import命令由于是在编译时就已经执行了，导致无法使用在if语句或函数体中，这样会导致无法在运行时动态加载模块。如果import命令要取代 Node 的require方法，这就形成了一个障碍。因为require是运行时加载模块，import命令无法取代require的动态加载功能。



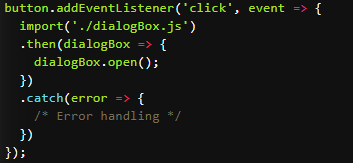
ES6提出import()函数，用于动态加载模块。

import函数的参数specifier，指定所要加载的模块的位置。import命令能够接受什么参数，import()函数就能接受什么参数，两者区别主要是后者为动态加载。import()返回一个Promise对象。

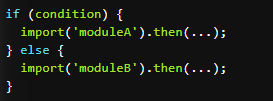


适用场景：

（1）按需加载。import()可以在需要的时候，再加载某个模块。



（2）条件加载。import()可以放在if代码块，根据不同的情况，加载不同的模块。

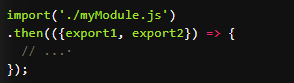


（3）动态的模块路径

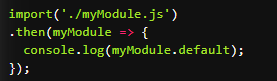


注意点：

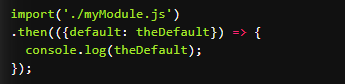
import()加载模块成功以后，这个模块会作为一个对象，当作then方法的参数。因此，可以使用对象解构赋值的语法，获取输出接口。



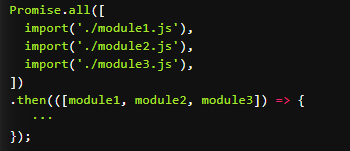
如果模块有default输出接口，可以用参数直接获得。



上面的代码也可以使用具名输入的形式。



如果想同时加载多个模块，可以采用下面的写法。



import()也可以用在 async 函数之中。

