nodejs模块化开发

# 1.模块化的概念以及好处

|  |
| --- |
|  |

# 2.模块化的使用

|  |
| --- |
|  |
|  |

# 3.nodejs模块化导出数据的第二种方法

|  |
| --- |
|  |

## 注意事项

|  |
| --- |
|  |

## 说明：不能使用exports=xxx的发送来导出数据的主要原因是1.module.exports和exports默认都是指向一个空对象{},2.require()函数导入模块只会返回module.exports的值,也就是说，你用exports=xxx导出数据，这样子，exports和module.exports就不是指向同一个对象，因为你改变了exports的指向，此时渲染exports的值改变了，module.exports还是{}，所以这样子导出的数据在外面是无法获取的。必须使用exports.name=value的形式，因为此时你在修改对象的值，而不是指向别的对象，也就是说此时module.exports还是指向你修改的对象，所以这样子导出是有效的。为了避免混乱，建议你就使用module.exports=xxx的方式来导出数据

# 4.导入模块文件的注意事项

|  |
| --- |
|  |
|  |
|  |

## 小案例

### 新建一个文件夹export-import-test，在export-import-test里面新建一个main.js文件和一个data.json文件，新建一个mymodules文件夹，在它里面新一个index.js文件，里面写一个Person类

### 项目结构

|  |
| --- |
|  |

### data.json

|  |
| --- |
| {      "name":"Jackline",      "age":18,      "gender":"female"  } |

### ./mymodules/index.js

|  |
| --- |
| class Person{      constructor(name,age){          this.name = name          this.age = age      }      intro(){          console.log(`hi,my name is ${this.name},my age:${this.age}`);      }  }  module.exports = Person |

## main.js

|  |
| --- |
| let data = require('./data')  //导入一个文件夹他会自动导入package.json的main属性指定的文件或者如果没有package.json,它会自动导入index.js  let Person = require('./mymodules')  console.log(data);  new Person("Jackline",18).intro(); |

## 效果

|  |
| --- |
|  |

### 注意:如果一个文件夹没有package.json文件而且同时有index.js和index.json,它会优先导入index.js

# 5.导入自定义模块的基本流程，其实内置模块也是类似的

|  |
| --- |
|  |

## 课堂例子：require的伪代码实现

|  |
| --- |
|  |
|  |
|  |
|  |

# 6.CommonJS模块化规范

|  |
| --- |
|  |