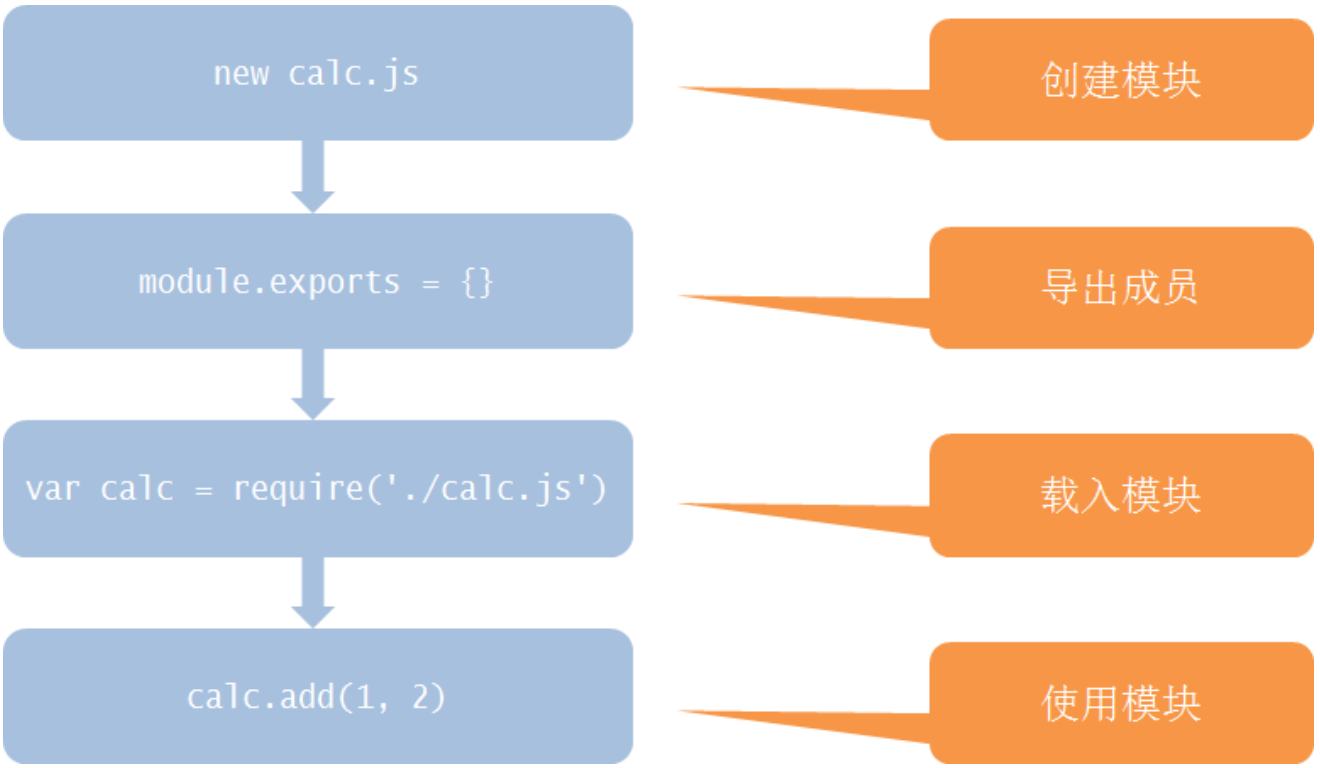
1- Node的模块化：commonJS

模块化开发

模块化开发过程：



Node.js遵循CommonJS规范

CommonJs：

- 服务端的模块化规范
- 代码运行在模块作用域，不会污染全局作用域
- 可多次加载，首次加载运行，之后加载运行缓存结果（可清除缓存）
- 加载顺序按照其在代码中出现的顺序

1-1模块的定义：

- 一个Js文件就是一个模块
- 模块内部是一个独立(封闭)的作用域
- 一个合格的模块是有导出成员，否则失去了模块的价值
- 模块之间必须通过导出或导入的方式协同

+ 导出方式
    - exports.name = value;
    - module.exprots = {name: value}
    - exports是指向module.exports的别名
    - var exprots = module.exprots;
    - module.exports被赋值会切断两者之间的联系

- 最终模块导出成员以module.exports为准

1-2载入模块：

- require：读入并执行一个js文件，返回该模块的exports对象

+ 扩展名
    - require('./module');　　执行JS文件
    - require('./module.js');　　执行JSON文件
    - require('./module.json');　　执行C++ 模块
    - require('./module.node');　　执行package.json 中main指向的文件
    - require('./module/default.js');　　执行module 中的index.js

+ 加载文件
    - 通过 ./ 或 ../ 开头：相对路径
    - 通过 / 开头：以系统根目录开始寻找模块

+ 加载目录
    - 目录路径：自动查看该目录的package.json文件，加载main属性指定的文件
    - 没有main属性或没有package.json文件，默认找目录下的index.js载入模块

+ 模块优先级
    - node核心模块
    - node_modules模块
    - 自定义模块

1-3require的实现机制

- 将传入的模块 ID 通过加载规则找到对应的模块文件
- 读取这个文件里面的代码
- 通过拼接的方式为该段代码构建私有空间
- 执行该代码
- 拿到 module.exports 返回

模块的缓存：

- 第一次加载某个模块时，Node会缓存模块，再次加载时直接从缓存取出该模块的module.exprots属性（不再执行该模块）
- 如果需要多次执行模块中的代码，一般可以让模块暴露行为（函数）
- 模块的缓存可以通过*require.cache*拿到，同样也可以删除

1-4模块内全局环境

- 操作文件需要使用绝对路径
- __dirname：获取当前文件的完整目录
- __filename：获取当前文件的完整路径
- 在REPL环境无效

+ module：模块对象

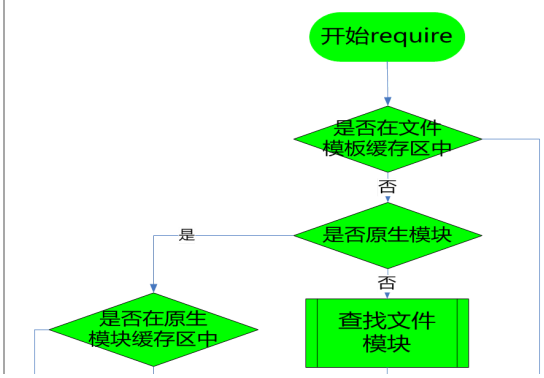
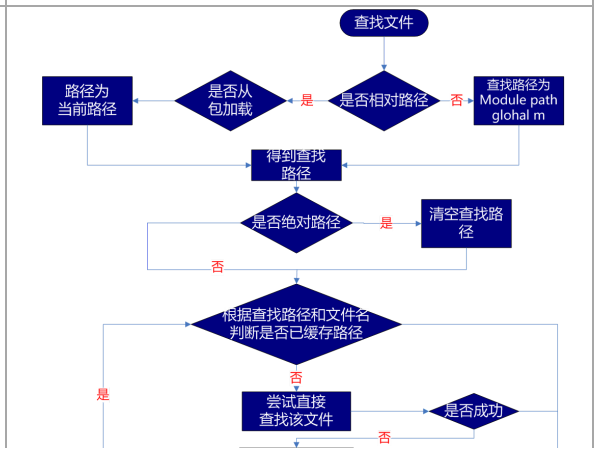
    - Node内部提供一个Module构造函数
    - 所有模块都是Module的实例
    - module.id：模块的识别符，带有绝对路径的文件名
    - module.filename：模块定义文件的绝对路径
    - module.loaded：返回一个布尔值，模块是否加载完成
    - module.parent：返回一个调度该模块的对象
    - module.children：返回一个数组，表示该模块要用到的其他模块
- exprots：映射到module.exports的别名
+ require（）
    - require.cache　//模块缓存
    - require.extensions　//如何处理文件名后缀
    - require.main　　//主模块
    - require.resolve() //查找模块位置,返回解析后的文件名(其实是完整路径)

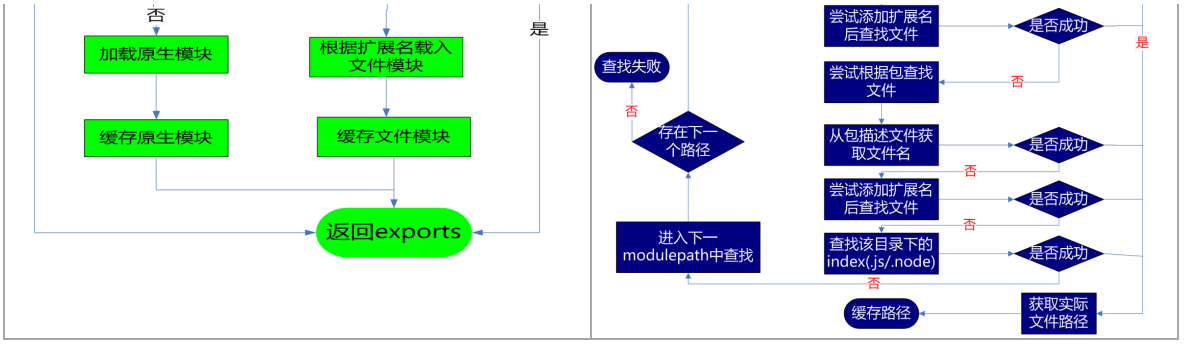
1-5常用内置模块

- path：处理文件路径。
- fs：操作（CRUD）文件系统。
- child_process：新建子进程。
- util：提供一系列实用小工具。
- http：提供 HTTP 服务器功能。
- url：用于解析 URL。
- querystring：解析 URL 中的查询字符串。
- crypto：提供加密和解密功能。

```
path.dirname(path)　//返回path目录名
path.join([..path])　//拼接路径
```

```
fs.readFile(path[, options], callback)　//异步地读取一个文件的全部内容
fs.readFileSync(path[, options])　//以同步的方式读取文件
```

查找模块	查找文件
	



2- 核心模块

- 如果只是服务器运行JavaScript代码，意义不大，因为无法实现任何功能（读写文件，访问网络）
- node本身提供了一系列的功能模块，用于与操作系统互动，这些模块为内置模块

内置模块	功能
path	处理文件路径
fs	操作（CRUD）文件系统
child_process	新建子进程
util	提供一系列实用小工具
http	提供 HTTP 服务器功能
url	用于解析 URL
querystring	解析 URL 中的查询字符串
crypto	提供加密和解密功能