第六节 Nodejs 模块原理

```
const fs = require('fs')
const path = require('path')
const vm = require('vm')
function Module(id) {
  this.id = id
  this.exports = {}
}
Module.prototype.load = function() {
  let extname = path.extname(this.id)
  Module._extensions[extname]=(this)
}
Module._extensions = {
  '.js'(module){
    // 读取
    let content = fs.readFileSync(module.id, 'utf-8')
    // 包装
    content = Module.wrapper[0] + content + Module.wrapper[1]
    // compile to fn
    let compileFn = vm.runInThisContext(content)
    let exports = module.exports
    let dirname = path.dirname(module.id)
    let filename = module.id
    // 调用
    compileFn.call(exports, exports,myRequire,module,filename,dirname)
  },
  '.json'(module){
     module.exports = JSON.parse(fs.readFileSync(module.id, 'utf-8'))
  },
}
Module._cache = {}
Module.wrapper = ['(function(exports, require,module,__filename,__dirname){','})']
Module._resolveFileName = function(filename) {
```

```
  let absPath = path.resolve(__dirname, filename)
  if(fs.existsSync(absPath)) {
    return absPath
  } else {
    let suffix = Object.keys(Module._extensions)
    for (let i = 0; i < suffix.length; i++) {
      const mPath = absPath + suffix[i]
      if(fs.existsSync(mPath)) {
        return mPath
      }
    }
    throw new Error(`${filename} is not exists`)
  }
}
function myRequire(filename) {
  // 1.绝对路径
  let mPath = Module._resolveFileName(filename)
  console.log(mPath)
  // 2.缓存优先
  let cacheModule = Module._cache[mPath]
  if(cacheModule) {
    return cacheModule.exports
  }
  // 3.创建空对象加载目标对象
  let module = new Module(mPath)

  // 4.缓存已加载过的模块
  Module._cache[mPath] = module
  // 5.编译执行
  module.load()
  // 6.返回数据
  return module.exports
}
let obj = myRequire('./v')
console.log(obj)
```