vscode配置nodejs代码补全工具

<https://blog.csdn.net/liyijun4114/article/details/51658087>

安装typings

使用npm全局安装typings

npm install -g typings

安装语言包

npm install --save-dev @types/node

注意:不能使用node.js作为文件名

注意: 在node中没有DOM(document) 和 BOM(window)

执行nodejs文件需要在cmd中定位到当前目录 然后使用 node 文件名 命令执行

node中的JavaScript

EcmaScript

没有DOM和BOM

核心模块(核心模块为nodejs内置的,需要使用的话要引入)

在node中为JavaScript提供了很多服务器级别的API,这些Api绝大多数都被包装到一个具名核心模块中了

例如操作文件的 fs 模块,http模块,path路径操作模块,os操作系统信息模块

第三方模块

用户定义模块

注意:相对路径必须加 ./ 和 ../

node模块化编程 // node中没有全局作用域只有模块作用域

require 引入模块并执行里面的代码,拿到被加载文件模块到处的借口对象

exports 导出模块中的类,方法,变量,默认都是封闭的,这个对象是一个动态对象,所需要的导出的成员都要挂在到这个对象上面

function add(a, b) {

  return (a + b)

}

exports.add = add

## fs模块

var fs = require(“fs”)

fs是file-system的简写,就是文件系统的意思,在node中如果想要对文件进行操作,就必须引入fs这个模块

fs.readFile(”要读取的文件路径” , function(error,data){})

//读取文件 , 如果读取失败error是错误对象,data是null,如果读取成功data是读取的数据,error是null

// 这里读取到的是十六进制的数据,如果需要转换则需要调用String构造对象或者data.toString()

var fs = require("fs")

fs.readFile("./hellow", function (err, data) {

  if (!err) {

    // 两种方式都可以

    // console.log(data.toString())

    console.log(String(data))

  } else {

    console.log("读取失败", err)

  }

})

fs.writeFile(“路径/文件名”,”要写入的内容”,function(err){})

// 写入文件,参数1位路径(其中包括文件名),参数2为要写入的数据(可以使用”\n”进行换行),参数3位回调函数(写入方法中只接收错误)

var fs = require("fs")

fs.writeFile("./aa.txt", "", function (err) {

  if (err) {

    console.log(err)

  } else {

    fs.readFile("./aa.txt", function (err, data) {

      if (!err) {

        if (data) {

          console.log("空文件data不判断为空")

        }

        console.log(data.toString())

      } else {

        console.log(err)

      }

    })

  }

})

## http模块

一个简单的http模块 请求地址localhost:8080

端口号: 范围是从0 到65535,默认http端口号为80端口https端口号为443

此外还有一些其他应用程序需要占用端口号进行一些操作,具体查询百度

个人习惯使用的端口号为8080

var http = require("http") // 引入http模块

var server = http.createServer() // 创建一个web服务器

// request请求事件处理函数需要接收两个参数

// request 请求对象,承载请求中的所有参数

// response 响应对象，使用response返回响应

server.on("request", function (request, response) {

  // 当服务器接收到request请求触发回调函数,注意此处的request是固定的

  console.log(request)

  console.log("收到收到,over.请求路径为", request.url)

  response.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' })// 解决中文响应乱码,浏览器不知道服务器响应内容编码的情况下会以当前操作系统默认的编码去解析,中文操作系统默认为GBK

  response.write("黄鸡收到黄鸡收到")

  // 事实证明...\n没有起到任何作用

  switch (request.url) {

    case "/":

    case "/index":

      response.write("\n首页");

      break;

    case "/login":

      response.write("\n登录");

      break;

    case "/about":

      response.write("\n关于我");

      break

  }

  response.end()

})

// 绑定端口号,启动服务器,回调函数是服务启动成功进行调用

server.listen(8080, function () {

  console.log("服务器启动成功")

})

补充知识点:

Content-Type:

text/plain; charset=utf-8 普通文本

text/html; charset=utf-8 浏览器会渲染为HTML标签

其余的 <https://tool.oschina.net/commons>