# 1.基本结构，通过3种写法来解析

|  |
| --- |
| let fs = require('fs')  let path = require('path')  let filePath1 = path.join(\_\_dirname,"files/client.txt")  let filePath2 = path.join(\_\_dirname,"files/secret.txt")  let filePath3 = path.join(\_\_dirname,"files/info.txt")  function getPromise(filepath){    return new Promise((resolve,reject)=>{        fs.readFile(filepath,"utf8",(err,data)=>{        if(err){            reject(err)        }        resolve(data)      })    })  }  async function joinData(filePath1,filePath2,filePath3){       let str = ''       let data1 = await getPromise(filePath1)       let data2 = await getPromise(filePath2)       let data3 = await getPromise(filePath3)      // console.log(data1);      str = data1 + '\r\n' +data2 + '\r\n' + data3      console.log('joined data:\n',str);      fs.writeFile('./files/result2.txt',str,err=>{          if(err){              console.log(err);              return;          }          console.log('saved result in file with success...');      })  }  joinData(filePath1,filePath2,filePath3) |
| let fs = require('fs')  let path = require('path')  let util = require('util')  let filePath1 = path.join(\_\_dirname,"files/client.txt")  let filePath2 = path.join(\_\_dirname,"files/secret.txt")  let filePath3 = path.join(\_\_dirname,"files/info.txt")  const readFile = util.promisify(fs.readFile) //util.promisify可以把一个异步函数变为一个返回Promise的函数  async function joinData(filePath1,filePath2,filePath3){       let str = ''       let data1 = await readFile(filePath1)       let data2 = await readFile(filePath2)       let data3 = await readFile(filePath3)      // console.log(data1);      str = data1 + '\r\n' +data2 + '\r\n' + data3      console.log('joined data:\n',str);      fs.writeFile('./files/result3.txt',str,err=>{          if(err){              console.log(err);              return;          }          console.log('saved result in file with success...');      })  }  joinData(filePath1,filePath2,filePath3) |
| let fs = require('fs')  let path = require('path')  let util = require('util')  let filePath1 = path.join(\_\_dirname,"files/client.txt")  let filePath2 = path.join(\_\_dirname,"files/secret.txt")  let filePath3 = path.join(\_\_dirname,"files/info.txt")  const readFile = util.promisify(fs.readFile) //util.promisify可以把一个异步函数变为一个返回Promise的函数  const writeFile = util.promisify(fs.writeFile) //util.promisify可以把一个异步函数变为一个返回Promise的函数  async function joinData(filePath1,filePath2,filePath3){       let str = ''       let data1 = await readFile(filePath1)       let data2 = await readFile(filePath2)       let data3 = await readFile(filePath3)      // console.log(data1);      str = data1 + '\r\n' +data2 + '\r\n' + data3      console.log('joined data:\n',str);      writeFile('./files/result3.2.txt',str)               .then(()=>{                  console.log('saved result in file with success...');               })  }  joinData(filePath1,filePath2,filePath3) |

## 函数必须加async关键字，可以在里面使用await来获取数据

# 2.注意事项

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

## 错误处理，外部捕获异常

|  |
| --- |
| //async ... await的await是异步的  let fs = require('fs')  let path = require('path')  let util = require('util')  let filePath1 = path.join(\_\_dirname,"files/client.txt")  let filePath2 = path.join(\_\_dirname,"files/secret.txt")  // let filePath2 = path.join(\_\_dirname,"files/client.txt")  let filePath3 = path.join(\_\_dirname,"files/info.txt")  const readFile = util.promisify(fs.readFile) //util.promisify可以把一个异步函数变为一个返回Promise的函数  async function testGetData(filePath1,filePath2,filePath3){         let data1 = await readFile(filePath1,"utf-8")       console.log(data1);       let data2 = await readFile(filePath2,"utf-8")       console.log(data2);       let data3 = await readFile(filePath3,"utf-8")       console.log(data3);  }  外部捕获异常  testGetData(filePath1,filePath2,filePath3).catch(err=>{      console.log(err);  }).finally(()=>{      console.log("finally,done...");  }) |

## 效果：

|  |
| --- |
|  |

## 效果，我们特意把应该路径写错

|  |
| --- |
|  |

## 内部错误处理，内部捕获异常

|  |
| --- |
| //async ... await的await是异步的  let fs = require('fs')  let path = require('path')  let util = require('util')  let filePath1 = path.join(\_\_dirname,"files/client1.txt")  let filePath2 = path.join(\_\_dirname,"files/secret.txt")  // let filePath2 = path.join(\_\_dirname,"files/client.txt")  let filePath3 = path.join(\_\_dirname,"files/info.txt")  const readFile = util.promisify(fs.readFile) //util.promisify可以把一个异步函数变为一个返回Promise的函数  async function testGetData(filePath1,filePath2,filePath3){         let data1 = await readFile(filePath1,"utf-8")       console.log(data1);       let data2 = await readFile(filePath2,"utf-8")       console.log(data2);       let data3 = await readFile(filePath3,"utf-8")       console.log(data3);  }  //内部异常处理  async function testGetData2(filePath1,filePath2,filePath3){      try {          let data1 = await readFile(filePath1,"utf-8")          console.log(data1);          let data2 = await readFile(filePath2,"utf-8")          console.log(data2);          let data3 = await readFile(filePath3,"utf-8")          console.log(data3);      } catch (error) {          console.log(error);      }    }  testGetData2(filePath1,filePath2,filePath3) |

## 效果，我们特意把应该路径写错

|  |
| --- |
|  |

## 也可以加finally

|  |
| --- |
| //async ... await的await是异步的  let fs = require('fs')  let path = require('path')  let util = require('util')  let filePath1 = path.join(\_\_dirname,"files/client.txt")  let filePath2 = path.join(\_\_dirname,"files/secret.txt")  // let filePath2 = path.join(\_\_dirname,"files/client.txt")  let filePath3 = path.join(\_\_dirname,"files/info.txt")  const readFile = util.promisify(fs.readFile) //util.promisify可以把一个异步函数变为一个返回Promise的函数  async function testGetData(filePath1,filePath2,filePath3){         let data1 = await readFile(filePath1,"utf-8")       console.log(data1);       let data2 = await readFile(filePath2,"utf-8")       console.log(data2);       let data3 = await readFile(filePath3,"utf-8")       console.log(data3);  }  // 外部捕获异常  testGetData(filePath1,filePath2,filePath3).catch(err=>{      console.log(err);  }).finally(()=>{      console.log("finally,done...");  })  //内部异常处理  async function testGetData2(filePath1,filePath2,filePath3){      try {          let data1 = await readFile(filePath1,"utf-8")          console.log(data1);          let data2 = await readFile(filePath2,"utf-8")          console.log(data2);          let data3 = await readFile(filePath3,"utf-8")          console.log(data3);      } catch (error) {          console.log(error);      }finally{          console.log('finally...');      }    } |