Node.js笔记

node.js是一个运行平台。

- 浏览器
- node.js从浏览器中将js剥离出来，让其可以去做后端编程。

特点:

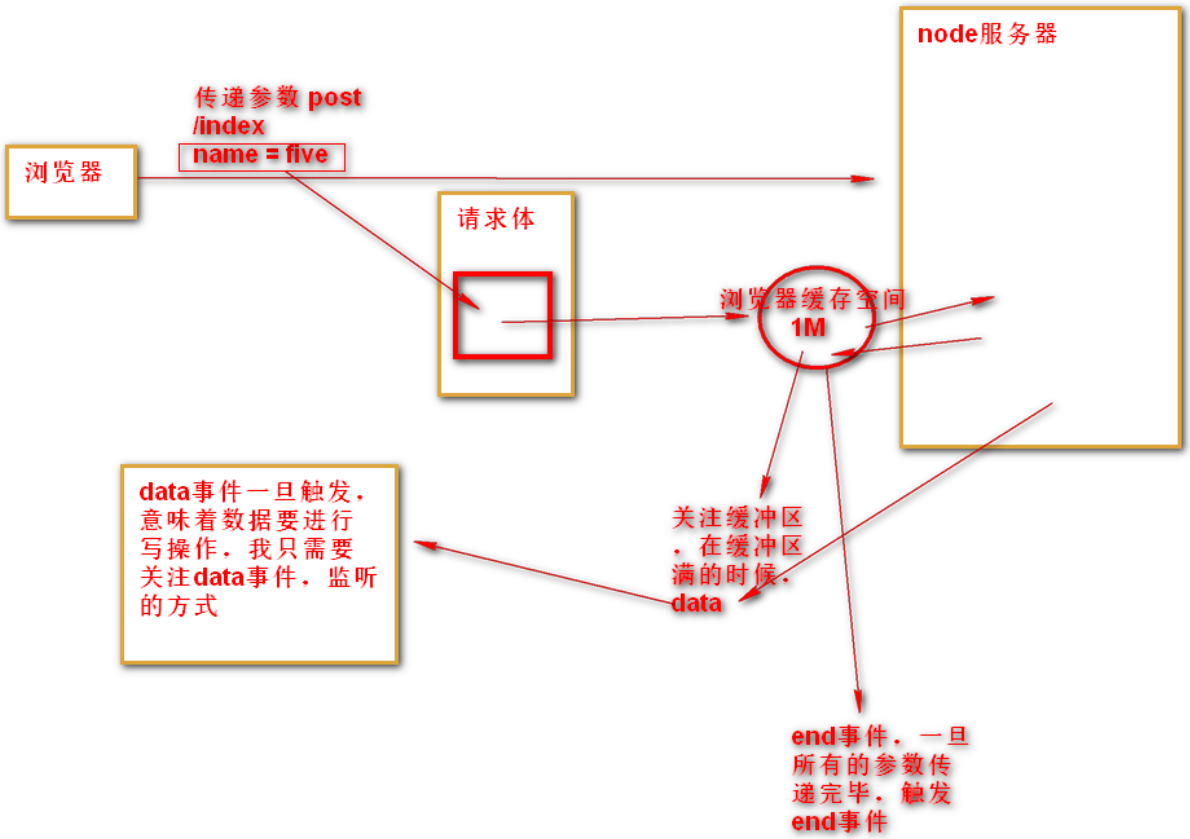
- 跨平台
- 运行js的速度快
    - V8引擎
        - 直接编译成机器码运行，极大的提高速度，但是机器码占用大量的内存空间
        - 半编译半解释，节省空间，折中的选择
- node.js提供了一个后台的运行环境
    - 服务器
        - 承载所有后台资源
        - 负责处理请求
- node服务器
    - node平台提供给js一个可以创建服务器的功能模块
- 第一个服务端代码
    - 服务器
    - 请求响应 处理逻辑

```
/**
 * Created by 12526 on 2018/8/23.
 */
//第一步 使用js语法 将node提供给js创建服务器的功能模块引入
进当前文件
```

```
//是node提供的http模块
const http = require("http");
//利用http模块语法创建服务 server是手动创建的服务

//创建处理请求的函数，从而做出响应
function start(req,res){
    console.log(req.url);
    console.log(req.method);
    console.log(req.headers);
    console.log("请求接收到")
    //如何响应请求    write是响应给请求所携带的信息
    let date = new Date();
    res.write("<h1>"+date+"</h1>");
    //结束标识符   结束本次响应处理
    res.end();
}
//req,res是服务器给予start
const server = http.createServer(start);
//手动定义端口号
server.listen(8181)
```

- /favicon.ico node.js默认每次请求都会访问当前的这个url来获取图标
- 如何接受get请求
    - node提供了相关模块，来提取get传递的参数
        - url
            - url.parse(req.url).query
        - querystring
            - querystring.parse(url.parse(req.url).query)
    - 直接使用url模块
        - url.parse(req.url,true).query
- 如何接受post请求



- node.js路由
  - 请求的种类各式各样，如何区分这些请求从而对应不同的请求做出不同的数据响应—-路由
  - 获取衣服数据访问 /yifu
  - 获取鞋子数据访问 /xiezi
- 关联静态文件
  - node提供的模块，来完成服务关联本地静态文件的功能
  - fs (filesystem)关联本地静态文件
    - fs.readFile 读操作 异步非阻塞
    - fs.readFileSync 读操作 同步阻塞 无回调
    - fs.writeFile 写操作 异步非阻塞 {flag:”a”}
    - fs.writeFileSync 写操作 同步阻塞
    - fs.createReadStream 预备读操作
    - fs.createWriteStream 预备写操作 没有立即执行 等待调用后在执行写操作
- 文件的分包

    - bin目录 服务文件
    - public目录 公开关联服务静态文件
    - routes目录 路由文件

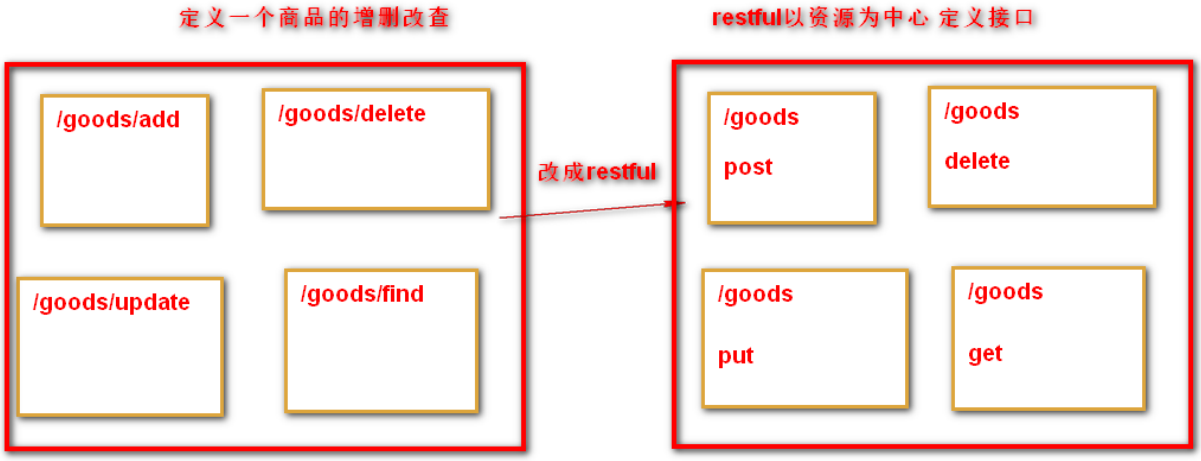
Express

是一个极简的web框架，历史悠久，比较成熟的一个框架。

- 使用express需要从npm下载 npm i express@4.15
- 第一步引入
- 第二步实例化

使用语法和属性：

- 路由
    - app.方法类型(“路由规则”，function(req,res){})
    - 在定义路由规则的时候，提供了几种写法
        - /fin*d *可以代表任何字符，中文也可以
        - /fin+d 代表+前面的字符可以有多个，不能一个没有
        - /fin?d 代表着n可有可无，也可以控制多个字符，使用括号
    - restful的支持
        - 以资源为中心的接口定义风格
    - restful接受参数
        - 传参/goods/five/18
        - 接参 req.params.key
        - 传参如果是? req.query接收



- 接收参数
  - get req.query可以将参数接收后直接转换成json对象
  - post 默认express是不携带接受post参数的语法

如何使本地静态文件关联到express框架中？？
中间件就是一些功能性组件，作用就是用来拓展express框架本身没有的功能

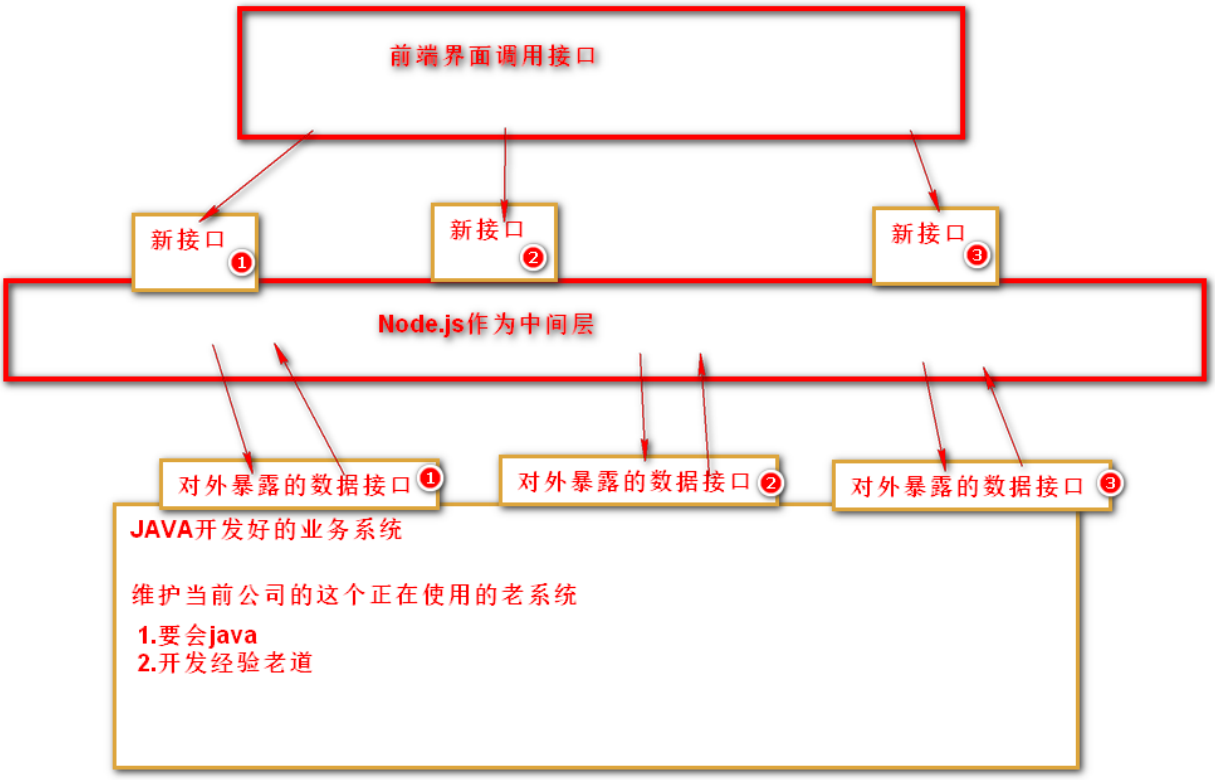
post接参语法,静态文件

- 使用中间件，配置到express中，使express拥有中间件赋予给它的功能
  - 使用中间件的步骤
    - 1.下载(内置不用下载)
    - 2.配置中间件生效
      - 需要将中间件调用出来 express.static(“关联文件夹路径”)
      - 调用实例化对象app.use() 将中间件功能关联到服务

- 关联静态文件–(static)`（唯一内置中间件）`大部分中间件需要下载，但是static不用去下载，已经内置在了express框架中
  - 配置属性 ctrl点击static，点击serve-static，点击send
    - extensions []可以配置静态文件的所有后缀，访问的时候不需要在加文件后缀
    - index配置默认跳转位置

        - etag，lastModified 配置浏览器304缓存

- 接收post请求传递的参数，简化post接参语法–(body-parser)
    - 使用
        - 下载 npm i body-parser
        - 使用app.use(bodyparser.urlencoded())来将中间件功能关联到当前app开启的服务
        - app.use(bodyparser.json()) 解决特殊场景的get传参问题
        - 特殊传参场景的传参场景
            - GET传参 restful ？ 都是在地址栏，但是有的GET请求传递参数并没有放在地址栏中，而放在了请求体中
            - 下载request模块，使用场景：异构系统之间做一个中间层，降低维护成本，增加原有系统的性能



- 日志中间件 morgan
    - 1.下载
    - 2.app.use(morgan()) 日志配置建议放在所有中间件的最上方
    - 自定义日志显示风格 源码中 format是定义显示风格，token是定义format显示风格中对应的显示:key
    - 持久化日志到本地文件，创建sream后交给morgan

    - 如何将日志保存到数据库中？自定义一个对象，这个对象当中一定有write方法，write对应的function我们可以自己控制，从而获得到了日志源数据，可以进行保存数据库操作

- cookie-parser 第三方
    - 1.下载 npm i cookie-parser
    - 2.使用 app.use(cookie())
        - res.cookie(“key”,”value”,{maxAge:1000});res.send()
        - req.cookies.key
        - res.clearCookie(“key”); res.send();

- express-session 第三方
    - 1.下载
    - 2.使用app.use()
        - req.session.key = {} 存储session
        - req.session.key 取session
        - req.session.destroy() 删除当前用户创建的所有session

```
app.use(session({
    saveUninitialized:false,//是否保存刚刚初始化的session对象到内存中
    resave:false,//是否定期更新保存在内存当中的session对象
    secret:"asd",//秘钥  加密session
    rolling:true,//每一次操作如果在存活周期时间内，是否重新重置session存活时间
    cookie:{maxAge:1000*10}//设定session存活时间
    /*store:"MemoryStore"*/ //指定session存储位置，以后需要将session剥离出服务器进行持久化操作
}))
```

自定义中间件：

- 对用户身份校验，合法身份才能访问对应的路径，否则就让他去登陆

    - 自定义一个功能性组件—拦截器，校验身份
    - app.use(function(req,res,next){}) next对象可以放行当前的请求，如果校验合法，调用next()放行，否则指定跳转路径
    - res.redirect 默认路径会拼接当前“/token”过滤的路径，可以写全域名+文件名解决上述问题，也可以使用../返回上一级直接过滤掉“/token”

- 什么是模块什么是中间件？？
    - 中间件：一旦在app.use配置，就可以生效当前中间件的功能。
    - 模块：什么时候使用，什么时候调。
- 上传下载(busboy)
    - 需要借助第三方模块来完成上传功能
        - 1.下载 npm i busboy
        - 2.引入
        - 3.赋予头信息
        - 4.赋予req中流信息 req.pipe()
        - 5.监听file和field事件，file事件将文件剥离出来，field事件将文本提取出来
        - 6.从file事件中的file对象获取原生流信息 保存到指定路，field提取出所有提交的文本信息后，赋予给定义的json对象
        - 7.监听finish事件，来获取所有file操作与field操作的完成
- 实战上传操作
    - 需要管理每一个用户上传的不同的文件，需要对每一个用户创建属于自己的文件夹，当前用户所有上传的文件都保存在其属于自己的文件夹中
    - 使用fs来判断和生成文件夹 fs.existsSync fs.mkdirSync
    - 路径问题 使用path模块
        - path屏蔽系统差异 window \ linux / ,一旦写死，可能系统之间会有路径不合法的情况，这时候path模块可以自动根据当前系统，来生成合法路径 path.join(参数1，参数2)
- EJS 就是用来动态的显示数据，服务端渲染，将数据如视图结合后响应给用户。
    - <% %>声明脚本，负责执行逻辑运算

- <%= %>输出脚本，负责显示数据
- 配置参数
  - debug 控制台可以显示详细的渲染源代码
  - cache+filename 为指定的模板添加缓存