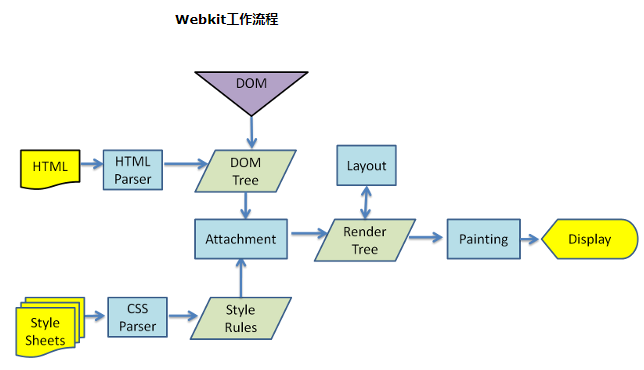
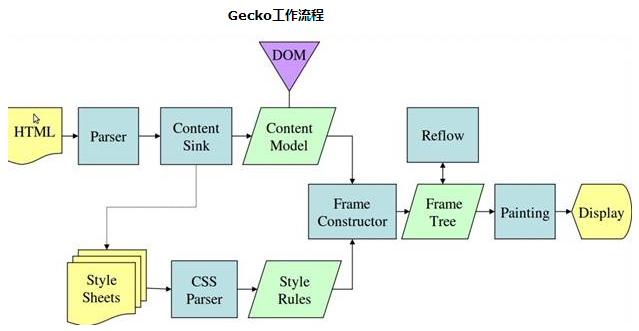
渲染引擎工作原理





浏览器访问网络的过程

用户在浏览器的地址栏输入地址，浏览器根据这个输入的地址，构建一个http请求报文，

构建后，会发起一个DNS地址解析的请求，浏览器把这个请求报文发送给服务器，服务器接收报文请求并解析，处理用户的请求，并将处理结果封装成http响应报文，然后发给把http响应报文发给浏览器，浏览器接收服务器的响应http报文，并解析。当浏览器在解析html页面的时候，发现遇到一些新的资源，会再次发起请求，最终浏览器展示出我们看到的页面。

web开发的本质

|  |  |
| --- | --- |
| 请求 | 客户端发起请求 |
| 处理 | 服务器处理请求 |
| 响应 | 服务器将处理发送给客户端 |

B\S与C\S网站架构

|  |  |
| --- | --- |
| B\S | Browser\Server 浏览器/服务器 |
| C\S | Client\Server 客户端/服务器 |
|  |  |

node卸载步骤

|  |  |
| --- | --- |
| 1 | 控制面板程序卸载 |
| 2 | Program Files(x86)\Node.js  Program Files\Node.js  Users\Administrator\AppData\Roaming\npm  Users\Administrator\AppData\Roaming\npm-cache |
| 3 | 删除文件环境变量 |
| 4 | 重启 |

node安装

|  |  |
| --- | --- |
| 1 | 检查是否安装，卸载 |
| 2 | 官网下载对应系统位数的window或其他版本 |
| 3 | 双击应用安装，都是默认next, finishud，默认的会自动把路劲添加到系统环境Path |
| 4 | cmd命令检查安装版本，node -v有版本安装成功，没有可能卸载不完全 |
| 版本 | LTS稳定版，Current最新版本 |

npm安装卸载

|  |  |
| --- | --- |
| 1 | 删除node\_modules及其package-lock.json文件 |
| 2 | 清除缓存 npm cache clean --force |
| 3 | 重新安装模块 npm install xxx |

技术栈

|  |  |
| --- | --- |
| Wamp | windows |
| lamp (php开发) | Linux Apache Mysql PHP |
| MEAN (node全栈工程) | MongoDB Express Angular Node.js |
| node论坛 | cnodejs.org专业的中文社区 |

nvm的window版本

|  |  |
| --- | --- |
| 1 | 用来维护多个版本的node，解决项目对node不同版本的需求 |
| 2 | 安装 [https://github.com/coreybutler/nvm-windows/releases](https://links.jianshu.com/go?to=https://github.com/coreybutler/nvm-windows/releases" \t "https://www.jianshu.com/p/_blank) |
| 3 | 可以直接通过nvm来进行node安装 |
| 4 |  |

node与传统，php,java,.net的区别

|  |  |
| --- | --- |
| 传统 | 依赖web容器，IIS,Apache |
| node | 不需要依赖web容器，本身就是一个web容器 |

npm命令

|  |  |  |
| --- | --- | --- |
| 安装 | npm install | npm install express (-g 全局安装) |
| 查看 | npm list |  |
| 卸载 | npm uninstall | npm uninstall express卸载查看npm ls |
| 搜索 | npm search | npm search express |
| 初始化 | npm init | 填一些信息 |
| 注册用户 | npm adduser | 邮箱注册 |
| 发布模块 | npm publish |  |
| 淘宝镜像 | npm install -g cnpm --registry=https://registry.npm.taobao.org | |
| cnpm安装 | cnpm install [name] | |

node特征

|  |
| --- |
| 事件驱动，非阻塞式I/O的模型，轻量高效 |
| 文件读取，写入都是异步的，同时进行，写入读取速度不同 |
| template引擎显示东西不能是json数组，对象问题 |
| readFile读取文件，设置的编码为了防止Buffer编码错误 |
| http页面响应编码设置在头部，能节省更多的代码 |

node模块 全局安装 ( -g )

|  |  |  |
| --- | --- | --- |
| nodemon | 避免每次重启服务,全局安装效果最佳 | nodemon jk.js  npm install nodemon -g (全局) |
| fs | 文件 |  |
| http | http |  |
| art-template | 引擎 |  |
| mime | 类型 | mime.getType(filename); |

http web服务

|  |  |  |
| --- | --- | --- |
| createServer() |  | http.createServer(function(req,res){}) |
| listen() |  | http.listen("8888","localhost",function(){}) |
| 响应编码 | 页面编码 | res.setHeader("Content-type","text/html;charset=utf-8"); |
| 结束响应 |  | res.end(date) |
| 重定向 |  | res.writeHead(302,'Found',{"Location":"/"}) |

fs 文件系统

|  |  |  |
| --- | --- | --- |
| readFile() | 读取文件 | readFile("music.txt",function(err,date){}) |
| writeFile() | 写入文件 | writeFile("music.txt",”内容”,function(err,date){}) |
| rename() | 移动文件 | fs.rename('旧文件.txt', '新文件.txt', (err) => {  if (err) throw err;  console.log('重命名完成');}); |

path与mime

|  |  |  |
| --- | --- | --- |
| 路由 | 为路由提供请求的 URL 和其他需要的 GET 及 POST 参数，随后路由需要根据这些数据来执行相应的代码 | |
| 自动获取类型 | let filename=path.join(\_\_dirname,req.url);  let page=mime.getType(filename); |  |
| 路劲 | path.join(\_\_dirname,"html/index.html"), "utf8", function(err, data) { }) | |

渲染引擎 art-template

|  |  |  |
| --- | --- | --- |
| 安装加载 | npm install art-template | var template=require(“art-template”) |
| template.render() | 第二个参数把数据渲染到第一个参数里 | template.render(date,list); |
|  |  |  |

弹窗模块

|  |  |
| --- | --- |
| 安装 | npm install -S alert-node |
| 加载模块 | var alert=require(“alert-node”) |
| 使用模块 | alert(“13254”) |
|  |  |

文件 增删查改 模块

|  |  |  |
| --- | --- | --- |
| express | 集成了http,url,express-art-template |  |
| bodyparser | 用来读取post提交的数据 |  |
| path | 路径匹配模块 |  |
| express-art-template | 安装之前必须安装了art-template |  |
| nodemon | 自动监听保存的内容  避免每次重启nodejsweb服务 | nodemon index.js |
| express.engine | 后缀名文件处理 | app.set("views",path.join(\_\_dirname,"message"))  app.set("views",path.join(\_\_dirname,"views")) |
| uuid | id唯一 | UUID.v1(); UUID.v4() |

其他

|  |  |  |
| --- | --- | --- |
| res.redirect() | 重定向 |  |
| res.render() | 渲染 | app.engine("html",require("express-art-template")); app.set("views",path.join(\_\_dirname,"/"));  res.render("index.html", list); |
|  |  |  |
|  |  |  |

mongoose模块 启动了才能用

mongod --dbpath "C:\Users\Administrator\Desktop\test\data\db" --logpath "C:\Users\Administrator\Desktop\test\data\log\mongodb.log" --logappend

|  |  |
| --- | --- |
| 安装 | npm install mongonse |
| 使用 | var mongoose=require(“mongoose”) |
| 连接,没有就创建数据库 | mongoose.connect("mongodb://localhost/user") |
| 设置数据结构 | var Schema=mongoose.Schema;  var People=new Schema({  name:{  type:String,//设置数据类型  required:true//设置必填选项  },  age:Number,  birthday:Date,  img:String,  friend:[String,String],  alive:Boolean  }) |
| 数据结构映射集合 | var Man=mongoose.model("student",People); |
| 集合增添数据 | var one=new Man({  name:"张三"+i,  age:18+i,  friend:["李四","王五"]  })  one.save(); |
| 集合删除数据 | Man.deleteOne({name:"张三29"}).then(function(){  console.log("删除成功")  })  Man.deleteMany({age:{$lte:30}}).then(function(){  console.log("删除成功")  }) |
| 集合修改数据 | // 修改一条数据(更改一整条)  Man.updateOne({name:"张三25"},{name:"李四",age:15},function(){  console.log("修改成功")  })  //修改一条数据(只能更改对应项目)  Man.findOneAndUpdate({name:"张三26"},{$set:{name:"王五"}},function(){  console.log("修改成功")  })  //修改多条数据  Man.updateMany({age:{$lte:38}},{name:"赵六",age:0},function(){  console.log("修改成功")  }) |
| 集合显示数据 | Man.find({age:{$lte:30}}).then(function(val){  console.log(val);  })  user.find({},function(err,all){  res.send(all);  }); |
| 重定向到首页 | res.redirect(“/”) |
| 传参id格式 | href={{'delete?id='+$value.\_id}} |

内存溢出的情形

|  |  |
| --- | --- |
| if写多了 | 解决方式 |
| 一般出现在递归函数中 | var str=JSON.stringify(data[0]);  var kjj=JSON.parse(str);  res.render("update.html",kjj); |
| 由于得到的数据是数组 |
| 要确保这个数组是字符串类型 |

multer 文件上传 npm install multer

|  |  |
| --- | --- |
| form格式表单格式 | <form action="/add" method="post" enctype="multipart/form-data"> |
| 定义文件存储位置，  文件目录不存在，则会自动创建，不建议用 | var upload = multer({ dest: 'uploads/' }) |
| 自定义文件目录与格式，目录得先写好，不然会报错  建议使用 | var storage = multer.diskStorage({  destination: function (req, file, cb) {  cb(null, path.join(\_\_dirname,'/html/files'))//不存在则报错  },  filename: function (req, file, cb) {  cb(null, file.fieldname + '-' + Date.now()+'-'+file.originalname)  }  })  var upload = multer({ storage: storage }) |
| 单个上传 | app.post("/add",upload.single('pic',2),function(req,res){  console.log(req.files[0].filename);  console.log(req.files[1].filename);  console.log(req.body);  res.send("666")  }) |
| 多个上传 | app.post("/add",upload.array('pic',2),function(req,res){  console.log(req.files[0].filename);  console.log(req.files[1].filename);  console.log(req.body);  res.send("666")  }) |
| req.files是对象  String-->Array | varvv=upload.fields([{name:'picture',maxCount:1},{name:'music',maxCount:2}])  app.post(‘/add’,uploads,function(req,res,next){ } |

express的优点

|  |
| --- |
| 不用设置中文响应编码 |
| 渲染转出为字符串在响应到页面，直接渲染到页面 |
| 可以直接设置请求的文件类型 |
| 可以设置不同提交方式的请求与响应 |
| 渲染后不用结束 |
|  |

数据库增删查改

|  |
| --- |
| <a href={{'delete?id='+$value.\_id}}><button class="btn btn-success" >删除</button></a> |
| app.use("/image",express.static(path.join(\_\_dirname,"img"))); |
| app.engine("html",require("express-art-template"));  app.set("views",path.join(\_\_dirname,"/")); |
| fs.unlinkSync(path.join(\_\_dirname,all[i].picture));  fs.unlinkSync(path.join(\_\_dirname,all[i].music)); |
| var vv=upload.fields([{name:'picture',maxCount:1},{name:'music',maxCount:1}]) |
| fs.existsSync(path.join(\_\_dirname,all[i].picture)) |

Unexpected field

|  |
| --- |
| 由于 HTML页面的name与  var cpUpload = upload.fields([{ name: 'pic', maxCount: 1 }, { name: 'mic', maxCount: 1 }])  里面的name 不一致导致 |

面试题

1. nodejs的优缺点

使用v8引擎速度非常快

不用担心多线程，锁，并行计算的问题

更新快，可能会出现版本问题

nodejs不像其他的服务器，对于不同的连接，不支持进程和线程操作

1. 什么是错误优先的回调函数

错误优先的回调函数,用于同时返回数据跟错误，第一个参数返回错误，并且验证是否出错，其他参数返回数据，例如

fs.readFile(filePath,function(err,data){ if(err){ }else{ } }

1. 如何用node监听80端口

在类Unix系统中你不应该去监听80端口，因为这需要超级用户权限。因此不推荐让你的应用直接监听这个端口。

目前，如果你一定要让你的应用80端口的话，你可以有通过在Node应用的前方再添加一层反向代理（例如nginx）来实现，如下图。否则，建议你直接监听大于1024的端口

反向代理指的是以代理服务器来接收Internet上的连接请求，然后将请求转发给内部网络上的服务器， 并且将服务器返回的结果发送给客户端。

1. 运算错误与程序员错误的区别

运算错误并不是bug,这是和系统相关的的问题，例如请求超时或者硬件故障。

而程序员错误就是所谓的bug。