重要性A

内核 IE -Trident safari - webkit chrome - webkit opear - Blink Firefox - Gecko

属性 arguments 获取的实参 并转换成伪数组 arguments.callee 当前函数体

GeiBoudingClientRect（）获取一个元素到视口的距离是一个对象 有l r tb 属性

Json

.onblur失去焦点；

数组常用方法 .Length .From 伪数组转换成数组 .concat 一个新数组合并 返回一个新数组 ，copythin 复制数组 (0,1,4) 开始0 1-4 复制 2参数忽略第一个下标对应的数值 然后返回一个与数组长度相等的数组；

.push 像数组后面添加元素 并获得新长度； unshifit 像头部添加并返会

Pop 删除最后一个元数组元素并返回删除的； shift 删除第一个返回第一个数组元素；

。Splice 左开右闭 一个参数删除第几个 数组对应下标 并返回

两个 参数删除第一个到第二个的下标 左开右闭

多个参数 从删除的那个开始顺序替换 可以额外添加 删除1个可以添加多个

。Filter()传一个回调 参数可以是数组 然后做判断 return arr>18;

Foreach 、Map 三个参数都是回调函数形参 a ，b,c 第一个是循环第一个值 第二个是下标

第三个是数组体；

搜索框改变里面的内容用value submit value 不提交



解析器 HTML 解析器 解析文档 ，css 解析器解析css样式（link进来的）；

link引入的css样式 1 由css解析器完成 2 阻塞渲染 3 阻塞js 语句执行

3阻塞js 执行 ；4 不阻塞Dom解析

Js阻塞 1 阻塞DOM解析 避免做无用功 js 可以操作DOM

2阻塞页面渲染 3阻塞后面js 可能 前者是后者的开发需求 dev ；

4js与css解析是互斥的 不能同时进行 ；

概念 重绘与 重排

重绘 ： 将浏览器元素的改动重新绘制

重排 ： 将布局 重新排列 (因为布局都改变了 元素位置肯定也改变了 所以必定触发重绘)；

Cdn加速 由服务器向cdn发送请求 cdn服务器 查询离他最近的 服务器并返回 还能阻止ddos攻击 如果没有缓存 将去DNS根服务器的一个区寻找然后递归查询或者返回；

Git指令最先进的分布式版控制

git init 变成可管理的仓库 想要管理首先要 成为一个库

Git add xx 把文件xx 放在暂存区

Git commit -m ‘提交的注释’将暂存区的文件提交

Git reset --hard HEAD^ 往上回退一个版本；

Git remote add origin 地址 关联一个远程库 起 名字 origin

Git push -u origin master将文件提交到origin仓库 分支是master 的文件 -u是第一次提交写

Git clone 地址 从远程库克隆 全部的

Git checkout -b dev 创建一个dev分支并切换到dev分支上 注意原分支是否有 文件不然原分支 会删掉

Git pull origin master 将文件从master分支上拉去下来

Git status 查看 状态 git 忽略文件 .gitignore 一个囊

Browser 主进程 Renderer 线进程

渲染过程 Html - Html解析器 添加到Dom解析

Css -css 解析器 css样式

合并成解析文档 - 布局 + 渲染 - 绘制 -显示

图层 1 为什么设置图层 ？ 为什么开启？

便于修改；

因为 绘制的单位是图层 开启图层减少 重绘的资源浪费

2开启图层的一些条件 canvas video css动画 translate 属性 csswill加速属性

优化 重排必定伴随重绘 所以减少重排 用重绘来代替 transform 代替posiiton left

离线修改 display none ； 少使用 重绘重排 ；

缓存机制 强缓存 弱缓存

强缓存不会像服务器请求 ；

协商缓存 像服务器 发送请求如果明知 304 返回读取 缓存的东西

共同点减少带宽浪费

模块化

Common js

暴露 module.exports || exports.xxx (同时存在听前者改变指针方向了)

引入用require （） 可选解构赋值；

Npm指令

Npm i jquery -g 全局 -g可选不然局部

Npm -v 版本查询 ctrl +c 结束指令

Npm init 初始化 添加package，json文件

Npm root -g 查看全局安装位置 npm i uniq @xxx版本 下载指定版本 ^锁定大版本一位数~ 两位数 不写就是版本指定为 此版本 --save 生产依赖 -dev 开发依赖

文件 - v查询版本 cls 请空指令界面

Es6模块

暴露 export 内容 {}

Export defalut 内容 默认暴露 只有一次

引入 import {} from ‘地址’；

翻译者 commonJs - browserify 全局安装 npm i browserify -g 局部安装 npm i browserify --save-dev

Es6翻译者 全局安装browserify 和babel npm babel-cli - g

局部安装browserify npm-babel-preser-es2015 --save-dev preser预调整

创建 一个babel文件夹. babelrc 设置解析格式 {“presers”：[“es2015”]};

Browserify -o babel -d

Io处理 计算客户需求 直白 瞬间或者提前 准备好的

Cpu 密集型 不断的传输 需求；

正则表达式的手机 跟邮箱

\*\*

var phoneReg = /^1[345789]\d{9}$/; 手机号正则格式  
var emailReg = /^\w+((-\w+)|(\.\w+))\*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+)\*\.[A-Za-z0-9]+$/; 邮箱正则格式

\*\*

Mongo菲关系性 不可分割性 原子性 insert 插入 collections 收集 查看9

NOdejs 实现原生 数据库

Let http = require‘http’；

Let sever = http.CreateServen((requset,respouse)=>{

Let parse = requst.url

Parst.split(?)[1]问号分隔

})

数组方法↑↑↑↑↑

状态码

**\*** 1xx : 服务器已经收到了本次请求，但是还需要进一步的处理才可以。  
**\*** 2xx : 服务器已经收到了本次请求，且已经分析、处理等........最终处理完毕！  
**\*** 3xx : 服务器已经接收到了请求，还需要其他的资源，或者重定向到其他位置，甚至交给其他服务器处理。  
**\*** 4xx ：一般指请求的参数或者地址有错误， 出现了服务器无法理解的请求（一般是前端的锅）。  
**\*** 5xx ：服务器内部错误（不是因为请求地址或者请求参数不当造成的），无法响应用户请求（一般是后端人员的锅）。

常见的 200成功状态 理想状态 301 重定向 旧的资源被永久移出了 不可以访问了，

将会跳转到一个新的资源 ， 将旧的网址替换为定向网址。

304 请求资源重定向 到缓存中（协商缓存命中）；

404 资源未找到 ，请求不到资源

500 服务器 收到了请求但是 内部产生错误

502 连接服务器失败了；

Cookie 是一个字符串 以 key value的形式储存 一般是4kb

Cookie 分为session cookie 和 持久性cookie

会话 cookie 在页面关闭后 消失 （他在 计算机浏览器运行内存上）

持久性cookie 储存在 计算机硬盘中 不会随着关闭浏览器而关闭；

用户第一次访问 服务器的时候 服务器会给 用户浏览器种下一个cookie

用户再次访问时 自动携带者cookie

服务器判断cookie是否合法 合法后解析 渲染

应用场景 http 无状态 7天免登录

前端也可以创建 cookie 但没什么意义

一半 cookie 配合着session一起使用

原生node 搭建服务器

Let http = require （‘http’） node 自带http；

Let server = http.creatServer((requist,response){

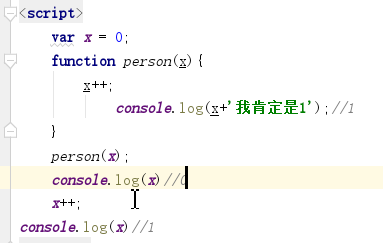
Response.end (‘<h1><h1> ’<=脚本路径不可以)

})

Server.listen()

<!你没资格放松自己 ，因为你 没赢过， 你失落过 ， 低谷过， 更没资格谈骄傲，

人外有人，I belive you >



什么是this ？

This 是个关键字 一个内置的引用变量 在函数中可以直接引用 ；

This在调用的时候才确定、绑定

This代表调用的函数的当前对象

在定义函数时this 没有确定 只有在执行时才确定绑定

如何确定this的值

Test （） //window

Obj.Test //obj new test () 空对象 test.bind(obj ) //obj

* **get方法** : 用来得到当前属性值的回调函数
* **set方法** : 用来监视当前属性值变化的回调函数
* **Array.prototype.forEach(function(item, index){}) : 遍历数组**
* **Array.prototype.map(function(item, index){}) : 遍历数组返回一个新的数组**
* **Array.prototype.filter(function(item, index){}) : 遍历过滤出一个子数组**