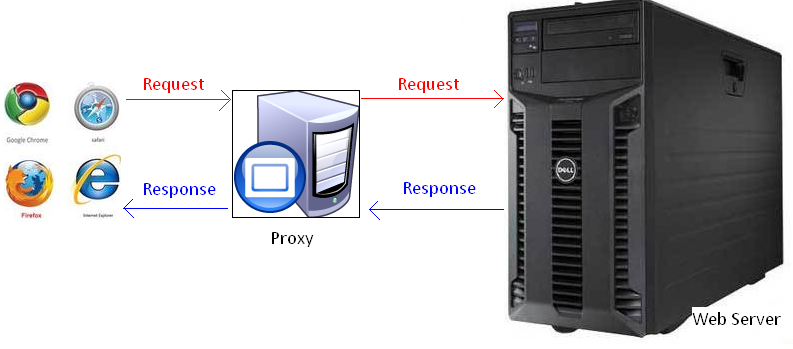
NodeJS 使用 CMD 模块系统，主模块作为程序入口点，所有模块在执行过程中只初始化一次。

http：[超文本传输协议](http://baike.baidu.com/view/468465.htm)（HTTP，HyperText Transfer Protocol)

用户代理（user agent）



Request 有可能是经过了代理服务器，最后才到达Web服务器的。

代理服务器就是网络信息的中转站，有什么功能呢？

　　1. 提高访问速度， 大多数的代理服务器都有缓存功能。

　　2. 突破限制， 也就是翻墙了

　　3. 隐藏身份。

**URL详解**

　　URL(Uniform Resource Locator) 地址用于描述一个网络上的资源，  基本格式如下

schema://host[:port#]/path/.../[;url-params][?query-string][#anchor]

　　scheme       指定低层使用的协议(例如：http, https, ftp)

　　host        HTTP服务器的IP地址或者域名

　　port#       HTTP服务器的默认端口是80，这种情况下端口号可以省略。如果使用了别的端口，必须指明，例如 http://www.cnblogs.com:8080/

　　path        访问资源的路径

　　url-params

　　query-string       发送给http服务器的数据

　　anchor-             锚

　　URL 的一个例子

http://www.mywebsite.com/sj/test;id=8079?name=sviergn&x=true#stuff

Schema: http

host: www.mywebsite.com

path: /sj/test

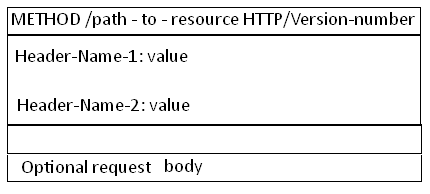
URL params: id=8079

Query String: name=sviergn&x=true

Anchor: stuff

# HTTP消息的结构

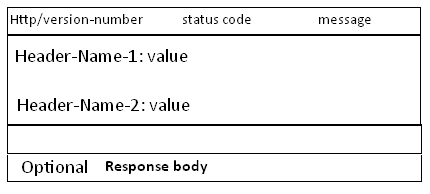
　　先看Request 消息的结构，   Request 消息分为3部分，第一部分叫请求行， 第二部分叫http header, 第三部分是body. header和body之间有个空行，



第一行中的Method表示请求方法，比如"POST"，"GET"，  Path-to-resoure表示请求的资源， Http/version-number 表示HTTP协议的版本号

当使用的是"GET" 方法的时候， body是为空的

我们再看Response消息的结构， 和Request消息的结构基本一样。 同样也分为三部分，第一部分叫request line, 第二部分叫request header，第三部分是body. header和body之间也有个空行，  结构如下图



# Get和Post方法的区别

　　Http协议定义了很多与服务器交互的方法，最基本的有4种，分别是GET,POST,PUT,DELETE. 一个URL地址用于描述一个网络上的资源，而HTTP中的GET, POST, PUT, DELETE就对应着对这个资源的查，改，增，删4个操作。 我们最常见的就是GET和POST了。GET一般用于获取/查询资源信息，而POST一般用于更新资源信息.

　　我们看看GET和POST的区别

　　1. GET提交的数据会放在URL之后，以?分割URL和传输数据，参数之间以&相连，如EditPosts.aspx?name=test1&id=123456.  POST方法是把提交的数据放在HTTP包的Body中.

　　2. GET提交的数据大小有限制（因为浏览器对URL的长度有限制），而POST方法提交的数据没有限制.

　　3. GET方式需要使用Request.QueryString来取得变量的值，而POST方式通过Request.Form来获取变量的值。

　　4. GET方式提交数据，会带来安全问题，比如一个登录页面，通过GET方式提交数据时，用户名和密码将出现在URL上，如果页面可以被缓存或者其他人可以访问这台机器，就可以从历史记录获得该用户的账号和密码.

# 状态码

　　Response 消息中的第一行叫做状态行，由HTTP协议版本号， 状态码， 状态消息 三部分组成。

　　状态码用来告诉HTTP客户端，HTTP服务器是否产生了预期的Response.

　　HTTP/1.1中定义了5类状态码， 状态码由三位数字组成，第一个数字定义了响应的类别

　　1XX  提示信息 - 表示请求已被成功接收，继续处理

　　2XX  成功 - 表示请求已被成功接收，理解，接受

　　3XX  重定向 - 要完成请求必须进行更进一步的处理

　　4XX  客户端错误 -  请求有语法错误或请求无法实现

　　5XX  服务器端错误 -   服务器未能实现合法的请求

页面出问题有可能是代码被人误覆盖，svn查看

寻找大玩家页面修复；搜索页面包屑；详情页样式；

width: 100px;

white-space: nowrap;

text-overflow: ellipsis;

overflow: hidden;

**15. Text-transform和Font Variant**

Text-transform用于将所有字母变成小写字母、大写字母或首字母大写：

　　p {text-transform: uppercase}   
　　p {text-transform: lowercase}   
　　p {text-transform: capitalize}

Font Variant用于将字体变成小型的大写字母（即与小写字母等高的大写字母）。

　　p {font-variant: small-caps}

**17. 用图片充当列表标志**

默认情况下，浏览器使用一个黑圆圈作为列表标志，可以用图片取代它：

　　ul {list-style: none}

　　ul li {   
　　　　background-image: url("path-to-your-image");   
　　　　background-repeat: none;   
　　　　background-position: 0 0.5em;   
　　}

**11. IE专用语句：方法二**

除了IE6以外，所有浏览器都不能识别属性前的下划线。而除了IE7之外，所有浏览器都不能识别属性前的\*号，因此可以写出只有这两个浏览器才能读到的语句：

　　.element {   
　　　　background: red; /\* modern browsers \*/   
　　　　\*background: green; /\* IE 7 and below \*/   
　　　　\_background: blue; /\* IE6 exclusively \*/   
　　}

**24. CSS提示框**

当鼠标移动到链接上方，会自动出现一个提示框。

　　<a class="tooltip" href="#">链接文字 <span>提示文字</span></a>

CSS这样写：

　　a.tooltip {position: relative}   
　　a.tooltip span {display:none; padding:5px; width:200px;}   
　　a:hover {background:#fff;} /\*background-color is a must for IE6\*/   
　　a.tooltip:hover span{display:inline; position:absolute;}

word-break: break-all;

《计算机文化（第8版）》（《New Perspectives on Computer Concepts》中文版，机械工业出版社）、《计算机科学导论》（《Foundations of Computer Science: From Data Manipulation to Theory of Computation》中文版，机械工业出版社）

《编程卓越之道（第一卷）：深入理解计算机》（《WRITE GREAT CODE: Volume 1:Understanding the Machine》

《深入理解计算机系统（修订版）》 （《Computer Systems A Programmer's Perspective》中文版，中国电力出版社）

最经典的当数C语言的设计者Kernighan与 Ritchie 合著的权威白皮书——《C程序设计语言（第2版）》（《The C Programming Language》中文版，机械工业出版社）。

《计算机程序的构造和解释（第2版）》 （《Structure and Interpretation of Computer Programs》中文版，机械工业出版社）

《编程珠玑（第2版）》（《Programming Pearls》中文版，中国电力出版社）、《程序设计实践》（《The Practice of Programming》中文版，机械工业出版社）、《代码阅读方法与实践》（《Code Reading: The Open Source Perspective》中文版，清华大学出版社）、《C专家编程》（《Expert C Programming》中文版，人民邮电出版社）。

《编程之道》（《The Tao of Programming》双语版，电子工业出版社）

我从《计算机组成原理》开始了解计算机相关的硬件和最基础的知识，比如主板、CPU、CPU时间片、中断等知识；而后从《计算机网络》学习网络通信过程， 比如TCP、UPD、路由规则等网络知识；而后从《计算机操作系统》学习进程调度，文件系统等系统环境知识；而后从《Linux操作系统》中学习 Linux操作与系统知识；而后从《编译原理》中学习编译器的相关知识，学习到编程语言的运行原理；而后从《C程序设计》学习C语言知识，从《C++程序 设计》中学习C++，学习C/C++语言是为了去学习一些用C/C++开发出的一些底层的软件，比如分析Nginx源码、PHP源码等；而后从《数据结 构》学习数据结构，如链表、堆栈、HashTable等；而后从《设计模式》中学习到那些以前不知所云的工厂模式、装饰器模式等。这些知识虽然不多，但是 这些方面的知识是整个计算机专业知识体系里面的最重要的几部分。当学过这些知识后，我发现我对整个计算机专业相关的知识都清晰了很多，工作中对一些底层的 东西也越来越清楚了。能够跟运维一起商量操作服务器的一些设置，因为我学过操作系统，能够跟PHP大牛去谈PHP源码的一些东西，因为我学过C/C++， 看过源码。

举例：认为 var a = {"id":1}; var b = {"id":1} 那么 a == b (正确答案： a != b)

**避免办法：**需要正确理解 JSON 对象的定义。 每次 {} 就相当于生成一个新的 JSON 对象。

这是一个刚使用 jQuery 时非常容易犯的错误。

**错误描述：**var a = $("#test"); if(!!a) {alert("ok!");} 运行这段代码你会发现，永远都会 alert，不管是否存在 id = "test" 的DOM

**避免办法：**jQuery 的 $ 方法查找对象，始终都会返回一个 jQuery 对象的，不管是否存在查找的结果。检查是否有结果请利用 length 属性，比如 if(a.length>0) {alert("ok!");}

Express cookies

var express = require('express');

var cookieParser = require('cookie-parser');

var app = express();

app.use(cookieParser());

app.get('/', function (req, res) {

// 检查 session 中的 isVisit 字段是否存在

// 如果存在则增加一次，否则为 session 设置 isVisit 字段，并初始化为 1。

if (req.cookie.isVisit) {

req.cookie.isVisit++;

res.send('<p>第 ' + req.cookie.isVisit + '次来此页面</p>');

} else {

req.cookie.isVisit = 1;

res.send("欢迎第一次来这里");

console.log("Cookies: ", req.cookies); //打印cookie

}

});

app.listen(80);

session是另一种记录客户状态的机制，不同的是Cookie保存在客户端浏览器中，而session保存在服务器上。

var express = require('express');

var session = require('session');

var app = express();

app.use(session({

secret: 'hubwiz app', //secret的值建议使用随机字符串

cookie: {maxAge: 60 \* 1000 \* 30} // 过期时间（毫秒）

}));

app.get('/', function (req, res) {

if (req.session.sign) {//检查用户是否已经登录

console.log(req.session);//打印session的值

res.send('welecome <strong>' + req.session.name + '</strong>, 欢迎你再次登录');

} else {//否则展示index页面

req.session.sign = true;

req.session.name = '汇智网'；

res.end('欢迎登陆！');

}

});

app.listen(80);

 name: 设置 cookie 中，保存 session 的字段名称，默认为 connect.sid 。

 store: session 的存储方式，默认存放在内存中，也可以使用 redis，mongodb 等。express 生态中都有相应模块的支持。

 secret: 通过设置的 secret 字符串，来计算 hash 值并放在 cookie 中，使产生的 signedCookie 防篡改。

 cookie: 设置存放 session id 的 cookie 的相关选项，默认为 (default: { path: ‘/’, httpOnly: true, secure: false, maxAge: null })

 genid: 产生一个新的 session\_id 时，所使用的函数， 默认使用 uid2 这个 npm 包。

 rolling: 每个请求都重新设置一个 cookie，默认为 false。

 resave: 即使 session 没有被修改，也保存 session 值，默认为 true

游记编辑页，删除，打印，layer报错

Node lessons

$ mkdir lesson1 && cd lesson1

# 这里没有从官方 npm 安装，而是使用了大淘宝的 npm 镜像

$ npm install express --registry=https://registry.npm.taobao.org

创建

$ touch app.js

我们来新建一个 lesson2 项目，并生成一份它的 package.json。

$ mkdir lesson2 && cd lesson2

$ npm init

$ npm install express utility --save

这次的安装命令与上节课的命令有两点不同，一是没有指定 registry，没有指定的情况下，默认从 npm 官方安装，上次我们是从淘宝的源安装的。二是多了个 --save 参数，这个参数的作用，就是会在你安装依赖的同时，自动把这些依赖写入 package.json。命令执行完成之后，查看 package.json，会发现多了一个 dependencies 字段，

* append() - 在被选元素的结尾插入内容
* prepend() - 在被选元素的开头插入内容
* after() - 在被选元素之后插入内容
* before() - 在被选元素之前插入内容

行程单显示；详情页编辑；

老版游记谷歌浏览器打开上传图片缓慢；modulesAdd；

老版游记修改

待做：新增图片UI；打开图片上传缓慢形成游记已改未发布；阻止重复提交；图片为上传结束不让点完成

**这是我原来的代码**

setTimeout(setTab('one',2,5),500);//setTab是要执行的方法，后来发现setTimeout的第一个参数必须是**需要编译的代码**或者是**一个函数方法**，而如果直接传入一行可执行代码，那么抱歉，这里会立即执行，没有延迟效果。   
**修改后的代码**

setTimeout(function(){setTab('one',2,5);},500);

$(this).attr("disabled", "disabled")

# CSS background-position 属性

|  |  |
| --- | --- |
| x% y% | 第一个值是水平位置，第二个值是垂直位置。  左上角是 0% 0%。右下角是 100% 100%。  如果您仅规定了一个值，另一个值将是 50%。 |

图片上传增加再传；重复提交；预览改成保存草稿；经典游记优化

目录滚动条；文字统计；编辑顺序传参；头条标签；

头条游记标签，列表页搜索页详情页；目录hover显示；分页插件；编辑页传参；

$(document).keydown(function(e){

    if(!e) var e = window.event;

    if(e.keyCode==32){

        alert("hha");//在这里写要改变的东西

    }

 });

keyCode 大写

**一、数组转字符串**

需要将数组元素用某个字符连接成字符串，示例代码如下：

[?](http://www.jb51.net/article/52038.htm)

|  |  |
| --- | --- |
| 1  2  3 | var a, b;  a = new Array(0,1,2,3,4);  b = a.join("-"); |

**二、字符串转数组**

实现方法为将字符串按某个字符切割成若干个字符串，并以数组形式返回，示例代码如下：

[?](http://www.jb51.net/article/52038.htm)

|  |  |
| --- | --- |
| 1  2 | var s = "abc,abcd,aaa";  ss = s.split(",");// 在每个逗号(,)处进行分解 |

Disabled input button有效 a无效

**TCP/IP 是用于因特网 (Internet) 的通信协议。**

TCP/IP 是供已连接因特网的计算机进行通信的通信协议。

TCP/IP 指传输控制协议/网际协议 (*T*ransmission *C*ontrol *P*rotocol / *I*nternet *P*rotocol)。

TCP/IP 定义了电子设备（比如计算机）如何连入因特网，以及数据如何在它们之间传输的标准。

TCP (传输控制协议) - 应用程序之间通信

IP (网际协议) - 计算机之间的通信

## 域名

12 个阿拉伯数字很难记忆。使用一个名称更容易。

用于 TCP/IP 地址的名字被称为域名。w3school.com.cn 就是一个域名。

当你键入一个像 http://www.w3school.com.cn 这样的域名，域名会被一种 DNS 程序翻译为数字。

在全世界，数量庞大的 DNS 服务器被连入因特网。DNS 服务器负责将域名翻译为 TCP/IP 地址，同时负责使用新的域名信息更新彼此的系统。

当一个新的域名连同其 TCP/IP 地址一同注册后，全世界的 DNS 服务器都会对此信息进行更新。

## 协议族

TCP/IP 是基于 TCP 和 IP 这两个最初的协议之上的不同的通信协议的大的集合。

## TCP - 传输控制协议

TCP 用于从应用程序到网络的数据传输控制。

TCP 负责在数据传送之前将它们分割为 IP 包，然后在它们到达的时候将它们重组。

## IP - 网际协议

IP 负责计算机之间的通信。

IP 负责在因特网上发送和接收数据包。

## HTTP - 超文本传输协议

HTTP 负责 web 服务器与 web 浏览器之间的通信。

HTTP 用于从 web 客户端（浏览器）向 web 服务器发送请求，并从 web 服务器向 web 客户端返回内容（网页）。

## HTTPS - 安全的 HTTP

HTTPS 负责在 web 服务器和 web 浏览器之间的安全通信。

作为有代表性的应用，HTTPS 会用于处理信用卡交易和其他的敏感数据。

## SSL - 安全套接字层

SSL 协议用于为安全数据传输加密数据。

window.onbeforeunload =function() {

if (checkSave()) {

return "您的修改内容还没有保存，您确定离开吗？"

}

}

var命令和function命令声明的全局变量，依旧是顶层对象的属性；另一方面规定，let命令、const命令、class命令声明的全局变量，不属于顶层对象的属性。也就是说，从ES6开始，全局变量将逐步与顶层对象的属性脱钩。

var a = 1;

// 如果在Node的REPL环境，可以写成global.a

// 或者采用通用方法，写成this.a

window.a // 1

let b = 1;

window.b // undefined

a标签点击执行时间在跳转：

一个可见a标签给用户点击，我href；另一个不可见a标签有href实现跳转；

可见a标签点击事件，包含一个延迟执行，模拟点击事件点击不可见a标签；

Gulp，combo，coffee

isNaN判断是否为数字

vagrant init tuniu初始化

vagrant status检查状态

D:\jobs\vboxs\tuniu

vagrant up tuniu开启

玩法点赞修复；

分页插件；

玩法样式修改

输入框超过字数后无法输入

if (len > 499) {

console.log($(this).val())

$(this).val($(this).val().substr(0,499))

}

node渲染500错误，模板里变量无效

let params = url.parse(req.url).search

得到？id=1&name=2

/landlord/manage/edit

前面没加/跳转请求不正确；

select标签选项值，$(‘.class’).val()；

请求地址在浏览器中能打开，而在node中失败，给请求url加url = encodeURI(url)

$.trim(value).length == 0；判断字符串为空，或只有空格

$.trim()删除首尾空格

### data

data(name) ⇒ value

data(name, value) ⇒ self

读取或写入dom的 data-\* 属性。行为有点像 [attr](http://www.bootcss.com/p/zeptojs/#attr) ，但是属性名称前面加上 data-。

$('li').eq(-1) //=> only the last list item

### toggle

toggle([setting]) ⇒ self

显示或隐藏匹配元素。如果 setting为true，相当于[show](http://www.bootcss.com/p/zeptojs/#show) 法。如果setting为false。相当于 [hide](http://www.bootcss.com/p/zeptojs/#hide)方法。

var input = $('input[type=text]')

$('#too\_long').toggle(input.val().length > 140)

nodeType 属性返回以数字值返回指定节点的节点类型。

如果节点是元素节点，则 nodeType 属性将返回 1。

如果节点是属性节点，则 nodeType 属性将返回 2。

It works because in JavaScript, true && expression always evaluates to expression, and false && expression always evaluates to false.

Var data=null;

data.id=123; wrong;

Var data={};right;

插件的方法函数，有回掉函数

types = $('input[name="valueContentType"]:checked')

types[i].val()报错改为types[i].value