**Git简介**

**Git**是一个分布式[版本控制](http://zh.wikipedia.org/wiki/%E7%89%88%E6%9C%AC%E6%8E%A7%E5%88%B6" \o "版本控制)／[软件配置管理](http://zh.wikipedia.org/wiki/%E8%BD%AF%E4%BB%B6%E9%85%8D%E7%BD%AE%E7%AE%A1%E7%90%86)软件，原来是linux内核开发者[林纳斯·托瓦兹](http://zh.wikipedia.org/wiki/%E6%9E%97%E7%BA%B3%E6%96%AF%C2%B7%E6%89%98%E7%93%A6%E5%85%B9" \o "林纳斯·托瓦兹)（Linus Torvalds）为了更好地管理[linux内核](http://zh.wikipedia.org/wiki/Linux%E5%86%85%E6%A0%B8" \o "Linux内核)开发而创立的.

**Git**是用于Linux内核开发的版本控制工具。与[CVS](http://zh.wikipedia.org/wiki/CVS" \o "CVS)、[Subversion](http://zh.wikipedia.org/wiki/Subversion" \o "Subversion)一类的集中式版本控制工具不同，它采用了分布式版本库的作法，不需要服务器端软件，就可以运作版本控制，使得源代码的发布和交流极其方便。Git的速度很快，这对于诸如Linux kernel这样的大项目来说自然很重要。Git最为出色的是它的合并追踪（merge tracing）能力。  
Git和其他[版本控制系统](http://zh.wikipedia.org/wiki/%E7%89%88%E6%9C%AC%E6%8E%A7%E5%88%B6%E7%B3%BB%E7%BB%9F" \o "版本控制系统)（如[CVS](http://zh.wikipedia.org/wiki/%E5%8D%94%E4%BD%9C%E7%89%88%E6%9C%AC%E7%B3%BB%E7%B5%B1" \o "协作版本系统)）有不少的差别，Git本身关心档案的整体性是否有改变，但多数的CVS，或[Subversion](http://zh.wikipedia.org/wiki/Subversion" \o "Subversion)系统则在乎档案内容的差异。因此Git更像一个档案系统，直接在本机上取得资料，不必连线到host端取资料回来。

由于svn需要有一个专门的代码托管服务器,而我做的是一个个人项目,没有svn需要的统一的svn服务器,但是为了不同机器之间代码的共享和快速迁移,所以选择一个网上有托管地址的方式. 因此选择了git作为代码版本控制工具,使用网上的代码托管地址 GitHub

-----------------维基百科

Node.js简介

**Node.js**是一个[事件驱动](http://zh.wikipedia.org/wiki/%E4%BA%8B%E4%BB%B6%E9%A9%85%E5%8B%95" \o "事件驱动)[I/O](http://zh.wikipedia.org/wiki/I/O)服务端[JavaScript](http://zh.wikipedia.org/wiki/JavaScript" \o "JavaScript)环境，基于[Google](http://zh.wikipedia.org/wiki/Google" \o "Google)的[V8引擎](http://zh.wikipedia.org/wiki/V8_(JavaScript%E5%BC%95%E6%93%8E)" \o "V8 (JavaScript引擎))。目的是为了提供撰写可扩充网络程序，如[Web服务](http://zh.wikipedia.org/wiki/Web%E6%9C%8D%E5%8B%99)。[[1]](http://zh.wikipedia.org/wiki/Node.js#cite_note-1)第一个版本由Ryan Dahl于2009年发布，后来，Joyent雇用了Dahl，并协助发展**Node.js**。[[2]](http://zh.wikipedia.org/wiki/Node.js" \l "cite_note-2) [[3]](http://zh.wikipedia.org/wiki/Node.js#cite_note-Node.js_pushes_JavaScript_to_the_server-side-3)

与一般[JavaScript](http://zh.wikipedia.org/wiki/JavaScript)不同的地方，**Node.js**并不是在[Web浏览器](http://zh.wikipedia.org/wiki/Web%E6%B5%8F%E8%A7%88%E5%99%A8)上运行，而是一种在服务器上运行的服务端[JavaScript](http://zh.wikipedia.org/wiki/JavaScript)。**Node.js**实现了部份[CommonJS](http://zh.wikipedia.org/w/index.php?title=CommonJS&action=edit&redlink=1" \o "CommonJS（页面不存在）)规格（Spec）。

**Node.js的优点**

nodejs作为一个新兴的后台语言，有很多吸引人的地方：

**RESTful API**

**单线程**

Node.js可以在不新增额外线程的情况下，依然可以对任务进行并行处理 —— Node.js是单线程的。它通过事件轮询（event loop）来实现并行操作.

**非阻塞IO**

Node采用一系列“非阻塞”库来支持事件循环的方式。本质上就是为文件系统、数据库之类的资源提供接口。向文件系统发送一个请求时，无需等待硬盘（[寻址](http://baike.baidu.com/view/1303626.htm)并检索文件），硬盘准备好的时候非阻塞接口会通知Node。

事件驱动事先定义一组操作,当服务器进行到某个操作时,会触发预先为这个操作定义好的事件处理函数,但是这个函数什么时候执行是不确定的.

**V8虚拟机**

Google的V8引擎本身使用了一些最新的[编译技术](http://baike.baidu.com/view/553519.htm" \t "_blank)。这使得用Javascript这类[脚本语言](http://baike.baidu.com/view/76320.htm" \t "_blank)编写出来的代码与用C这类高级语言写出来的代码性能相差无几，却节省了开发成本。对性能的苛求是Node的一个关键因素。 Javascript是一个[事件驱动](http://baike.baidu.com/view/536048.htm" \t "_blank)语言，Node利用了这个优点，编写出可扩展性高的服务器。Node采用了一个称为“事件循环(event loop）”的架构，使得编写可扩展性高的服务器变得既容易又安全

-------百度百科

**Express.js**  
Express.js是一个小而强大的node.js的web程序框架,它提供了一系列强大的功能来创建各种web应用程序. 在大量衫的HTTP工具方法和Connect蹭伯的帮助下,用户可以快速又方便的创建强健且用户友好的API. Express仅仅是给应用程序包装了一层轻量级的功能函数,不会影响任何node.js特性.

----------express.js官网

NoSQL MongoDB 简介

NoSQL，指的是非关系型的数据库。随着互联网[web2.0](http://baike.baidu.com/view/733.htm" \t "_blank)网站的兴起，传统的关系数据库在应付web2.0网站，特别是超大规模和高并发的SNS类型的web2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。

**NOSQL**

随着互联网[web2.0](http://baike.baidu.com/view/733.htm" \t "_blank)网站的兴起，非关系型的数据库成了一个极其热门的新领域，非关系数据库产品的发展非常迅速。而传统的关系数据库在应付[web2.0](http://baike.baidu.com/view/733.htm" \t "_blank)网站，特别是超大规模和高并发的SNS类型的web2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题，例如：

1、High performance - 对数据库高并发读写的需求

[web2.0](http://baike.baidu.com/view/733.htm)网站要根据用户个性化信息来实时生成[动态页面](http://baike.baidu.com/view/2065821.htm" \t "_blank)和提供动态信息，所以基本上无法使用[动态页面静态化](http://baike.baidu.com/view/2510440.htm)技术，因此数据库并发负载非常高，往往要达到每秒上万次读写请求。关系数据库应付上万次SQL查询还勉强顶得住，但是应付上万次SQL写数据请求，硬盘IO就已经无法承受了。其实对于普通的BBS网站，往往也存在对高并发写请求的需求。

2、Huge Storage - 对海量数据的高效率存储和访问的需求

对于大型的SNS网站，每天用户产生海量的用户动态，以国外的Friendfeed为例，一个月就达到了2.5亿条用户动态，对于关系数据库来说，在一张2.5亿条记录的表里面进行SQL查询，效率是极其低下乃至不可忍受的。再例如大型web网站的用户登录系统，例如腾讯，盛大，动辄数以亿计的帐号，关系数据库也很难应付。

3、High Scalability && High Availability- 对数据库的高可扩展性和高可用性的需求

[1]在基于web的架构当中，数据库是最难进行横向扩展的，当一个应用系统的[用户量](http://baike.baidu.com/view/6999815.htm)和访问量与日俱增的时候，你的数据库却没有办法像web server和app server那样简单的通过添加更多的硬件和服务节点来扩展性能和负载能力。对于很多需要提供24小时不间断服务的网站来说，对[数据库系统](http://baike.baidu.com/view/7809.htm)进行升级和扩展是非常痛苦的事情，往往需要停机维护和[数据迁移](http://baike.baidu.com/view/1342339.htm)，为什么数据库不能通过不断的添加[服务器节点](http://baike.baidu.com/view/4729549.htm)来实现扩展呢？

在上面提到的“三高”需求面前，关系数据库遇到了难以克服的障碍，而对于[web2.0](http://baike.baidu.com/view/733.htm)网站来说，关系数据库的很多主要特性却往往无用武之地，例如：

1、数据库事务一致性需求

很多web[实时系统](http://baike.baidu.com/view/344431.htm)并不要求严格的数据库事务，对读一致性的要求很低，有些场合对写一致性要求也不高。因此数据库事务管理成了数据库高负载下一个沉重的负担。

2、数据库的写实时性和读实时性需求

对关系数据库来说，插入一条数据之后立刻查询，是肯定可以读出来这条数据的，但是对于很多web应用来说，并不要求这么高的实时性。

3、对复杂的SQL查询，特别是多表关联查询的需求

任何[大数据](http://baike.baidu.com/view/6954399.htm)量的web系统，都非常忌讳多个大表的关联查询，以及复杂的数据分析类型的复杂SQL报表查询，特别是SNS类型的网站，从需求以及产品设计角度，就避免了这种情况的产生。往往更多的只是单表的主键查询，以及单表的简单条件[分页](http://baike.baidu.com/view/159980.htm)查询，SQL的功能被极大的弱化了。

因此，关系数据库在这些越来越多的应用场景下显得不那么合适了，为了解决这类问题的非关系数据库应运而生。

特点:

**它们可以处理超大量的数据**。

**它们运行在便宜的PC服务器集群上**。

PC集群扩充起来非常方便并且成本很低，避免了“sharding”操作的复杂性和成本。

**它们击碎了性能瓶颈**。

NoSQL的支持者称，通过NoSQL架构可以省去将Web或Java应用和数据转换成SQL友好格式的时间，执行速度变得更快。

“SQL并非适用于所有的程序代码，” 对于那些繁重的重复操作的数据，SQL值得花钱。但是当数据库结构非常简单时，SQL可能没有太大用处。

**没有过多的操作**。

虽然NoSQL的支持者也承认关系数据库提供了无可比拟的功能集合，而且在[数据完整性](http://baike.baidu.com/view/702953.htm" \t "_blank)上也发挥绝对稳定，他们同时也表示，企业的具体需求可能没有那么多。

## 优点

### 易扩展

### 大数据量，高性能

### 灵活的数据模型

### 高可用

**NoSQL与关系型数据库设计理念比较**

[关系型数据库](http://baike.baidu.com/view/495393.htm)中的表都是存储一些格式化的数据结构，每个元组字段的组成都一样，即使不是每个元组都需要所有的字段，但数据库会为每个元组分配所有的字段，这样的结构可以便于表与表之间进行连接等操作，但从另一个角度来说它也是关系型数据库性能瓶颈的一个因素。而非[关系型数据库](http://baike.baidu.com/view/495393.htm)以键值对[存储](http://baike.baidu.com/view/87682.htm)，它的结构不固定，每一个[元组](http://baike.baidu.com/view/176522.htm)可以有不一样的字段，每个元组可以根据需要增加一些自己的键值对，这样就不会局限于固定的结构，可以减少一些时间和空间的开销

MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。他支持的[数据结构](http://baike.baidu.com/view/9900.htm)非常松散，是类似json的bjson格式，因此可以存储比较复杂的[数据类型](http://baike.baidu.com/view/675645.htm)。Mongo最大的特点是他支持的查询语言非常强大，其语法有点类似于[面向对象](http://baike.baidu.com/view/125370.htm)的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。它的特点是高性能、易部署、易使用，存储数据非常方便。

**主要功能特性**：

◆ 面向集合存储，[易存储](http://baike.baidu.com/view/2712177.htm)对象类型的数据

◆ 模式自由

◆支持动态查询

◆支持完全索引，包含内部对象

◆支持查询

◆支持复制和故障恢复

◆使用高效的二进制数据存储，包括大型对象（如视频等）

◆自动处理碎片，以支持[云计算](http://baike.baidu.com/view/1316082.htm)层次的扩展性

◆支持RUBY，PYTHON，JAVA，C++，PHP等多种语言

◆文件存储格式为BSON（一种JSON的扩展）

◆可通过网络访问

------------------百度百科

MarkDown

**Markdown** 是一种[轻量级标记语言](http://zh.wikipedia.org/wiki/%E8%BD%BB%E9%87%8F%E7%BA%A7%E6%A0%87%E8%AE%B0%E8%AF%AD%E8%A8%80)，创始人为[约翰·格鲁伯](http://zh.wikipedia.org/w/index.php?title=%E7%B4%84%E7%BF%B0%C2%B7%E6%A0%BC%E9%AD%AF%E4%BC%AF&action=edit&redlink=1)（John Gruber）和[亚伦·斯沃茨](http://zh.wikipedia.org/wiki/%E4%BA%9A%E4%BC%A6%C2%B7%E6%96%AF%E6%B2%83%E8%8C%A8" \o "亚伦·斯沃茨)（Aaron Swartz）。它允许人们“使用易读易写的纯文本格式编写文档，然后转换成有效的[XHTML](http://zh.wikipedia.org/wiki/XHTML)(或者[HTML](http://zh.wikipedia.org/wiki/HTML))文档”。这种语言吸收了很多在[电子邮件](http://zh.wikipedia.org/wiki/%E7%94%B5%E5%AD%90%E9%82%AE%E4%BB%B6)中已有的纯文本标记的特性。使其成为可读性最大并可再发行的可输入输出的格式。

## Markdown优点

* 纯文本，所以兼容性极强，可以用所有文本编辑器编辑。
* 可以专注写作而不是排版。用Word写作的时候，经常浪费大量时间去思考排版，而用Markdown，写作完成后可以自己css定义样式或使用别人的css样式。
* Markdown 语法简单，很快就可以学会。
* Markdown 的标记语法有极好的可读性。
* 格式转换方便，Markdown 的文本你可以轻松可以通过各种工具（如<http://johnmacfarlane.net/pandoc/>）转换为html、pdf、epub、mobi等格式。
* 可以使用git、svn进行版本管理，协作也更方便。

当使用Html写作方式时,需要自己注意书写开始闭合标签.当写作的结构很复杂时,就需要很多标签来格式化文本,但是标签数目过多的时候,满满的文章中就会到处是开始闭合标签,而把真正需要写作的内容淹没,可读性非常差.但是使用markdown写作方式时,markdown本身的语法就非常具有可读性.

EJS

Ejs是一个express.js的前端框架,配合现在流行的MVC编程模式,将模板和数据分开,使得数据层和模板层耦合度大降低,提高开发效率,而且可读性很强.

**Ejs和jade对比**

### WEB模板jade、ejs、handlebars 万行代码解释效率比较，jade完败

在刚刚入门NodeJS，在考虑使用哪个模板时，简单的以1万行数据，进行解释效率比较：  
Jade 模板：

!!!

html

head

title #{title}

meta(charset="UTF-8")

body

div.description #{description}

ul

- each data **in** datas

li.item(id='item\_'+data.index)

span= data.time

a.art(href=data.url)= data.title

ejs 模板：

<!doctype html>

**<html>**

**<head>**

**<meta** **charset**="UTF-8"**>**

**<title>**<%=title%> - Page Test**</title>**

**</head>**

**<body>**

**<div** **class**="description"**>**<%=description%>**</div>**

**<ul>**

<% **function** data(data) { %>

<li class="item" id="item\_<%=data.index%>">**<span>**<%=data.time%>**</span>**<a href="<%=data.url%>" class="art"><%=data.title%>**</a></li>**

<% } %>

<% datas.map(data) %>

**</ul>**

**</body>**

**</html>**

handlebars 模板：

<!doctype html>

**<html>**

**<head>**

**<meta** **charset**="UTF-8"**>**

**<title>**{{title}} - Page Test**</title>**

**</head>**

**<body>**

**<div** **class**="description"**>**{{description}}**</div>**

**<ul>**

{{#datas}}

**<li** **class**="item" **id**="item\_{{index}}"**><span>**{{time}}**</span><a** **href**="{{url}}" **class**="art"**>**{{title}}**</a></li>**

{{/datas}}

**</ul>**

**</body>**

**</html>**

效率比较结果（平均消耗时间，约数）  
**jade 287ms** > **ejs 43ms** > **handlebars 28ms**

jade因为采用了类似zen code的语法，比较新奇，但效率极其低下。如果只保留

 部分的1万行数据解释，则约为245ms。  
综上所述，对jade，我个人不建议，除了效率，另外一个主要原因是可视化太弱，甚至可以说是毫无可视化可言，学习成本高，维护与团队合作成本高，语法过于晦涩、复杂。

----------------------<http://cnodejs.org/topic/50e70edfa7e6c6171a1d70fa>

Jquery简介

jQuery是一个兼容多浏览器的[javascript](http://baike.baidu.com/view/16168.htm" \t "_blank)库，核心理念是write less,do more(写的更少,做的更多)。jQuery在2006年1月由美国人[John Resig](http://baike.baidu.com/view/3141971.htm" \t "_blank)在纽约的barcamp发布，吸引了来自世界各地的众多JavaScript高手加入，现在由Dave Methvin率领团队进行开发。如今，jQuery已经成为最流行的[javascript](http://baike.baidu.com/view/16168.htm" \t "_blank)库，在世界前10000个访问最多的网站中，有超过55%在使用jQuery。

jQuery是免费、开源的，使用[MIT](http://baike.baidu.com/view/74918.htm" \t "_blank)许可协议。jQuery的[语法](http://baike.baidu.com/view/135635.htm" \t "_blank)设计可以使开发者更加便捷，例如操作[文档](http://baike.baidu.com/view/55621.htm)对象、选择[DOM](http://baike.baidu.com/view/14806.htm)元素、制作[动画](http://baike.baidu.com/view/7262.htm" \t "_blank)效果、事件处理、使用[Ajax](http://baike.baidu.com/view/1641.htm)以及其他功能。除此以外，jQuery提供API让开发者编写插件。其模块化的使用方式使开发者可以很轻松的开发出功能强大的静态或动态网页。

## 特点

jQuery包含以下特点：

[1]

1.动态特效

2.AJAX

3.通过插件来扩展

4.方便的工具 - 例如浏览器版本判断

5.渐进增强

6.链式调用

7.多浏览器支持，支持[Internet Explorer](http://baike.baidu.com/view/85144.htm" \t "_blank)6.0+、[Opera](http://baike.baidu.com/view/10019.htm" \t "_blank)9.0+、[Firefox](http://baike.baidu.com/view/3279.htm" \t "_blank)2+、[Safari](http://baike.baidu.com/view/110484.htm" \t "_blank)2.0+、[Chrome](http://baike.baidu.com/view/1835504.htm" \t "_blank)1.0+（在2.0.0中取消了对[Internet Explorer](http://baike.baidu.com/view/85144.htm" \t "_blank)6,7,8的支持）

-----------------百度百科

**为什么要进行前端优化**

随着WEB2.0时代来，给网络的带来了空前的发展。前端用户体验变得越来越显的重要，从而来弥补B/S结构的用户交互型差的一些弊端，可是这样会带来一个问题就是会增加客户端的压力，比如大量运用JS代码，大家都知道JS代码是运行在客户端的，会影响到整个网页的在浏览器的解析效率，这样也可能暗示着会增加客户端的流量，所以不管是从服务器负载角度还是站在用户的角度来看，对客户端的代码进行优化都显得尤为重要！

**为什么要从前端开始着手有三个主要原因**：

1. 这里有提升和改进的潜力。如果能减少一半的体积，就能减少40%的响应时间。
2. 改进前端比改进后端需要的时间和资源更少。（改进后端要重新设计应用程序规划，代码，寻找优化代码的方法，添加或改变硬件配置，分布式数据库，等等）
3. 我们的黄金规则是：首先优化前端表现，这些东西耗费了用户端响应时间中的80%。

常见的优化方法 :

**2.把样式表放到顶部**

我们发现把css放到文档头部会让网页加载得更快。因为这样可以让页面逐渐加载。  
把样式表放到接近底部的问题是它阻止了页面元素的逐渐显示。这样还会导致“flash of unstyled content” 即在样式表加载之前页面内容是以没有样式的形式显示出来的，待加载完样式后，页面重绘，内容一闪即改变了样式表现。

**把脚本放到底部**

把脚本放到尽可能底部的地方，一个原因是让页面逐渐渲染，另一个是实现更好的并行下载。

对于脚本，脚本以下的内容被阻止逐渐加载了，因为只有当下载完脚本以后才会下载下面的内容，第二个脚本引起的问题是阻止平行下载。 “http/1.1 specification”建议浏览器对一个域名， 同一时间下载数不超过2个（按：实际监测发现一般有超过2个），我曾经让ie并行下载100个图片。 当脚本正在下载的时候，浏览器不会开始下载任何东西

**.避免css expressions**

**让脚本和样式外延**

Javascript和CSS应该是外部调用还是内嵌呢？  
用外部调用文件的方式更快，因为他们是可以被缓存的，如果是内嵌在页面中他们就无法被缓存了！想想如果用户要在你的网站看很多很多的页面，如果都是使用同一个外部脚本和样式，那么他们一旦被缓存，就再也不需要下载了，这样会给你带来很大的潜在好处。

**减小脚本体积**

有两个比较流行的工具是用来减小脚本的体积的–JSMin和YUI Compressor

主要是对 js代码和CSS代码 还有图片 进行优化,

辅助进行的是对服务器发来的html代码进行压缩,去掉空格 回车等等

Js代码优化:

1压缩js代码为一个文件,,减少http请求次数,去掉空格回车符

2 服务器端Gzip压缩 减小传输体积

3 服务器端对js文件**加一个长时间过期的头部,便利客户端能够缓存js文件**

**4 运用cdn技术,提高js文件加载速度**

**Css代码优化**

1压缩css代码,去掉空格回车符

2 CSS代码简写。这里说的简写主要解决的问题是，网站代码冗余。如果可以用一句表达的语言，就不要多出几行代码，符合W3C标准。

3 同属性提出

如果2个样式带有同样的属性，可以把同属性单独提出处理。例如A样式和B样式的字体颜色、大小、边框都一样，那么就可以考虑提出。

图片优化

1压缩图片体积

2 css图片使用CSS Sprit技术减少服务器请求次数

3 尽量少使用大图片,防止网络原因造成效果显示缓慢,影响用户体验