1.前端介绍

1.1什么是前端和前端开发

广义上，针对于互联网应用，凡是运行在客户端（PC、平板、手机以及其他连接互联网的设备）的程序或代码统称为前端.相反运行于服务器端的统称为后端.

狭义上，前端指的是使用Web技术(Html、css、js)开发Web应用的统称。

一个开发者可以通过学习 HTML,CSS,JavaScript进入前端开发领域, 这些代码可以运行在 Web浏览器, 无壳浏览器, Web视图之中, 或用于编译本地运行环境的输入。下面来说说一下四个运行场景

1.2前端和的运行环境

1. Web浏览器是用于检索, 呈现和遍历万维网(WWW)信息的软件. 一般而言, 浏览器可以运行在台式机, 笔记本电脑, 平板电脑或手机. 但是近来, 几乎在任何事物上都能够发现浏览器(如: 冰箱上, 汽车里等).常见的浏览器有Chrome、IE、Firefox、Safari等
2. 无壳浏览器是指没有图形用户界面的Web浏览器, 可以通过命令行接口控制达到网页自动化的目的(如: 功能测试, 单元测试等). 把无壳浏览器当做可以从命令行运行的浏览器, 它依然可以检索和遍历网页.最普遍的无壳浏览器有PhantomJS、slimerjs、trifleJS

1. Webviews 被本地 OS 用来运行网页. 把Web视图当做Web浏览器中的iframe或者单个的Tab, 其嵌入于运行在设备上的本地应用程序中(如：iOS, android, windows)。也就是我们常说的内嵌浏览器。 Web视图开发最普遍的解决方案如下:
    - Cordova (用于本地手机/平板应用)
    - NW.js (即 Node-Webkit, 用于桌面应用)
    - Electron (用于桌面应用)
2. 最后, 前端开发者从 Web 浏览器开发环境中学到的东西也可以用于不受浏览器引擎驱动的环境下. 目前, 脱离 Web 引擎，使用 Web 技术(如: CSS 和 JavaScript)去创建真正的本地应用的开发环境正在出现。这类的环境目前代表性的有React Native。

本节参考自：https://dwqs.gitbooks.io/frontenddevhandbook/content/

2.Web发展史

2.1 史前时代：Internet和www的构建

Internet(因特网)以相互交流信息资源为目的，基于一些共同的协议，并通过许多路由器和公共互联网而成，它是一个信息资源和资源共享的集合。

1969年，美国国防部高级研究计划局建立一个命名为ARPANET的网络。

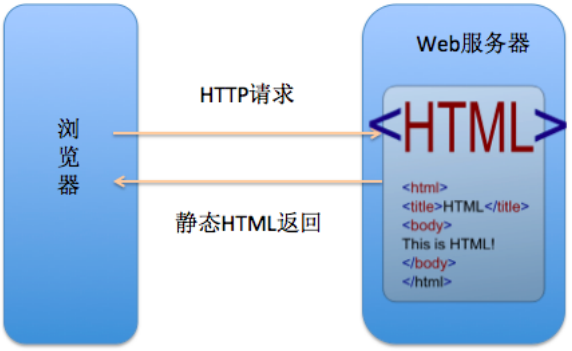
1985年，美国国家科学基金会在开始建立计算机网络NSFNET。NSFNET成为Internet上主要用于科研和教育的主干部分，代替了ARPANET的骨干地位

1989年MILNET（由ARPANET分离出来）实现和NSFNET连接后，开始采用Internet这个名称。自此以后，其他计算机网络相继并入Internet。

20世纪90年代初，商业机构开始进入Internet,使Internet开始了商业化进程，并发展为连接全世界的网络。

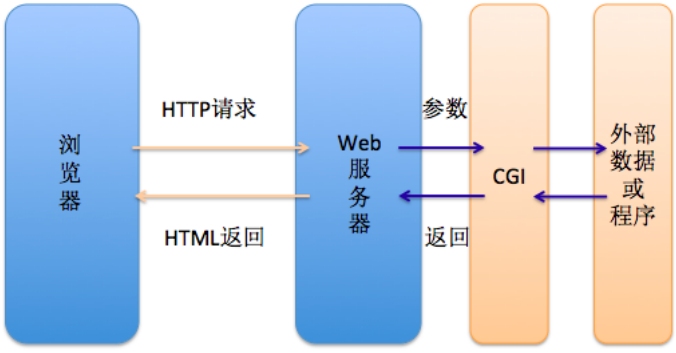
1991年8月6日，Tim Berners Lee在alt.hypertext新闻组贴出了一份关于World Wide Web的简单摘要，标志了Web页面在Internet上的首次登场。

1993年，万维网的发明者Berners Lee建立了万维网联盟（World Wide Web Consortium，W3C），负责Web相关标准的制定。浏览器的普及和W3C的推动，使得Web上可以访问的资源逐渐丰富起来。



这个时候Web的主要功能就是浏览器向服务器请求静态HTML信息。

最初在浏览器中主要展现的是静态的文本或图像信息，1993年CGI（Common Gateway Interface）出现了，Web上的动态信息服务（也就是后端）开始蓬勃兴起。CGI定义了Web服务器与外部应用程序之间的通信接口标准，因此Web服务器可以通过CGI执行外部程序，让外部程序根据Web请求内容生成动态的内容。

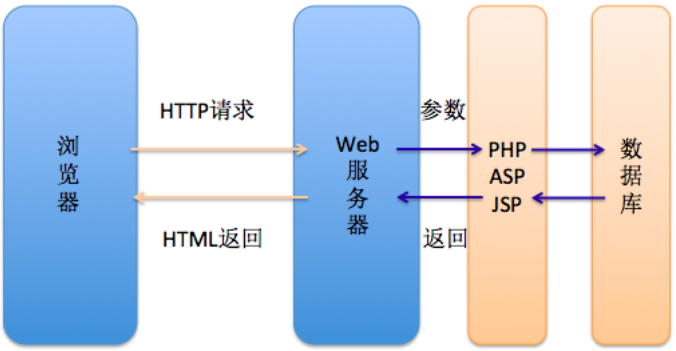


此时Web可以根据动态请求在因特网上分享信息资源

2.2 石器时代：后端脚本（PHP、ASP、JSP）、css的出现

上面提到的CGI采用Perl和C语言处理输入并输出整个HTML页面，当页面大到一定程度的时候，性能、可读性和维护性越来越差。为了处理复杂应用，一种方法是把输出HTML中的固定部分先存起来（称之为模板），把动态的部分标记出来，Web请求处理时，先用程序生成动态的内容，再把模板读进来、把动态内容填进去形成最终返回。这样的程序做起来太繁琐，于是1994年PHP诞生了，PHP可以把程序（动态内容）嵌入到HTML（模版）中去执行，不仅能更好的组织Web应用的内容，而且执行效率比CGI还更高。之后96年出现的ASP和98年出现的JSP本质上也都可以看成是一种支持某种脚本语言编程（分别是VB和Java）的模版引擎。

1996年W3C发布了CSS1.0规范。CSS允许开发者用外联的样式表来取代难以维护的内嵌样式，而不需要逐个去修改HTML元素，这让HTML页面更加容易创建和维护。

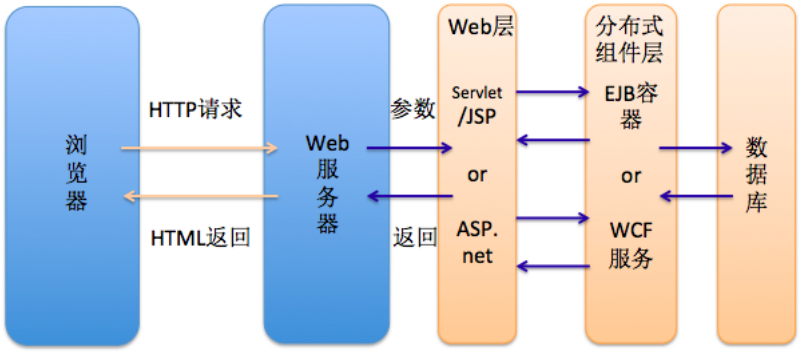



```
<html> <body>
<p>Hello world!</p> <p>
<%response.write("Hello world from server!")%> </p> </body> </html>
```

2.3青铜时代：J2EE/.net的出现以及后端框架横飞的年代

Web开始广泛用于构建大型应用时，在分布式、安全性、事务性等方面的要求催生了J2EE(现在已更名为Java EE)平台在1999年的诞生，从那时开始为企业应用提供支撑平台的各种应用服务器也开始大行其道。

2000年随之而来的.net平台，其ASP.net构件化的Web开发方式以及Visual Stidio.net开发环境的强大支持，大大降低了开发企业应用的复杂度。ASP.Net第一次让程序员可以像拖拽组件来创建Windows Form程序那样来组件化地创建Web页面。

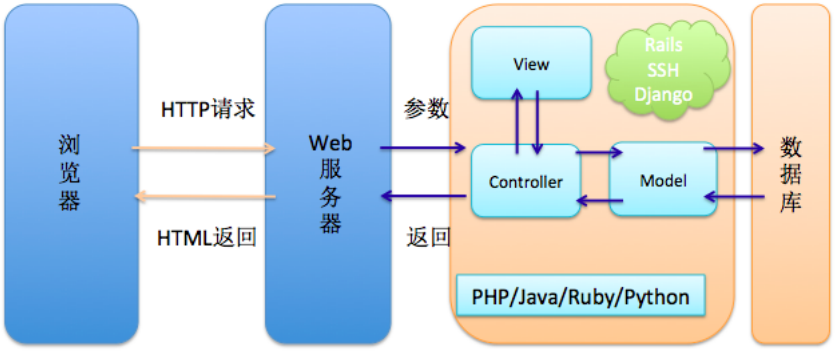


两大平台诞生之后，组件化编程技术盛极一时，Web技术的发展开始了一段框架横飞的年代，各种辅助Web开发的技术框架层出不穷。

虽然脚本语言大大提高了应用开发效率，但是试想一个复杂的大型Web应用，访问各种功能的URL地址纷繁复杂，涉及到的Web页面多种多样，同时还管理着大量的后台数据，因此我们需要在架构层面上解决维护性和扩展性等问题。这个时候，MVC的概念被引入到Web开发中来了。2004年出现的Struts就是当时非常流行的Java Web开发的MVC框架。此外，数据访问也逐渐通过面向对象的方式来替代直接的SQL访问，出现了ORM（Object Relation Mapping）的概念，2001年出现的Hibernate就是其中的佼佼者，已经成为Java持久层的规范JPA的主要参考和实现。

更多的全栈框架开始出现，比如2003年出现的Java开发框架Spring，同时更多的动态语言也被加入到Web编程语言的阵营中，2004年出现的Ruby开发框架Rails，2005出现的Python开发框架Django，都提供了全栈开发框架，或者自身提供Web开发的各种组件，或者可以方便的集成各种组件。比如Spring基于IoC和AOP思想可以方便得整合出全套Web开发组件，SSH（Struts+Spring+Hibernate）一度成为Java Web开发的标配。

后端框架的出现使开发业务复杂的Web应用更加便捷。



本节参考自：https://www.tianmaying.com/tutorial/web-history

3.前端发展史

3.1 设计师的时代

2005年之前并没有前端的概念，只有设计师和程序员，这时的Web属于后端MVC时代。设计师只负责做页面，后端在View层套用设计师开发好的页面，用户和应用的交互属于后端的范畴（form表单的提交、页面的跳转等）。

这时候的前端指的就是网页设计，也正是网页设计三剑客（Dreamweaver、Fireworks、Flash）的时代，也就是切图、做动画、拼页面。

2003年的新浪网：

早期静态页面代码

```
<body>
    <div id="container">
    <form action="" method="post">
    <table id="register">
     <tr>
        <td class="register_left">名字：</td>
        <td><input id="realname" type="text"  class="reg_text"/></td>
        <td rowspan="12" class="register_right">
            <h4><img src="images/read.gif" alt="alt" />阅读贵美网服务协议 </h4>
            <textarea cols="30" rows="15" name="agreement">欢迎阅读服务条款协议，本协议阐述之条款和条件适用于您使用Gmgw.com网站的各种工具和服务。
          本服务协议双方为本站与本站用户，本服务协议具有合同效力。
          本站的权利和义务：
            ......
      </textarea></td>
      </tr>
      <tr>
        <td class="register_left">姓氏：</td>
        <td><input id="xing" type="text"  class="reg_text"/></td>
      </tr>
      <tr>
        <td class="register_left">登录名：</td>
        <td><input id="username" type="text"  class="reg_text"/>（可包含 a-z、0-9 和下划线） </td>
      </tr>
      <tr>
        <td class="register_left">密码：</td>
        <td><input id="pwd" type="text"  class="reg_text"/>（至少包含 6 个字符）</td>
      </tr>
      <tr>
        <td class="register_left">再次输入密码：</td>
        <td><input id="repwd" type="text"  class="reg_text"/></td>
      </tr>
```

```
    <tr>
      <td class="register_left">电子邮箱：</td>
      <td><input id="email" type="text"  class="reg_text"/>（必须包含 @ 字符） </td>
    </tr>
    <tr>
      <td class="register_left">性别：</td>
      <td><input name="sex" type="radio" id="male"/><label for="male">男</label>
          <input name="sex" type="radio" id="female"/><label for="female">女</label></td>
    </tr>
    <tr>
      <td class="register_left">头像：</td>
      <td> <input type="file" /></td>
    </tr>
    <tr>
      <td class="register_left">爱好：</td>
      <td><input name="hobby" type="checkbox" id="run"/><label for="run">运动</label>
         <input name="hobby" type="checkbox" id="chat"/><label for="chat">聊天</label>
         <input name="hobby" type="checkbox" id="play"/><label for="play">玩游戏</label> </td>
    </tr>
    <tr>
      <td class="register_left">出生日期：</td>
      <td><input class="reg_small" id="nYear" value="yyyy" />
       年 
                <select id="nMonth" name="nMonth">
                  <option value="" selected="selected">[选择月份]</option>
                  <option value="0">一月</option>
                  <option value="1">二月</option>
                  <option value="2">三月</option>
                  <option value="3">四月</option>
                  <option value="4">五月</option>
                  <option value="5">六月</option>
                  <option value="6">七月</option>
                  <option value="7">八月</option>
                  <option value="8">九月</option>
                  <option value="9">十月</option>
                  <option value="10">十一月</option>
                  <option value="11">十二月</option>
                </select> 月 
                <input id="nDay" class="reg_small" value="dd"/>日
          </td>
    </tr>
    <tr>
      <td> </td>
      <td><input type="submit" class="submit" value=" " />
         <input type="reset" value=" " class="reset" />
         </td>
    </tr>
    <tr>
      <td> </td>
      <td> </td>
    </tr>
  </table></form></div>
  </body>
```

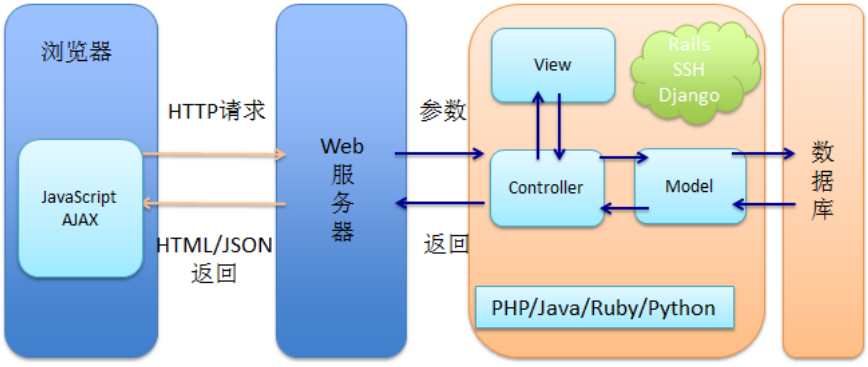
这个时期，界面由后端View层根据设计师模板渲染。Web的变更通常是后端程序员在View中更改一些功能，功能相应的界面排版和样式就会乱这时需要设计师重新布局排版；Web的改版基本等于重做，设计师要重新设计新的界面，后端也需要重新考虑功能和跳转等。

3.2Ajax和SPA应用的出现

1995年NetScape公司设计的JavaScript被用作浏览器上运行脚本语言为网页增加动态性。微软随后推出类似JScript，但是缺乏统一的语言规范，使得浏览器兼容性成为一个程序员的梦魇。JavaScript最终被提交到欧洲计算机制造商协会(ECMA)，做为中立的ECMA开始了标准化脚本语言之路，并将其命名为ECMAScript。JavaScript可以响应浏览器端的用户事件，检测表单的正确性，动态修改HTML页面结构DOM，因此可以减少与服务器端的通信开销，并且做出很酷的页面动态效果。

2005年出现的AJAX这个概念使得JavaScript再次大放异彩。AJAX即“Asynchronous JavaScript and XML”（异步的JavaScript与XML技术），指的是一套综合了多项技术的浏览器端网页开发技术，可以基于JavaScript的XmlHttpRequest的用于创建交互性更强的Web应用。

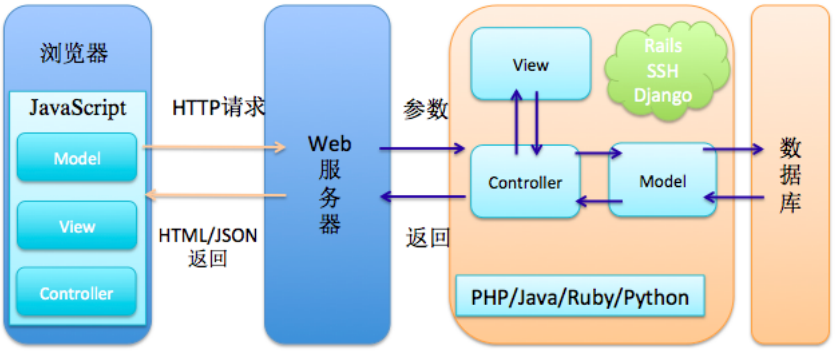
Google在地图和Gmail等产品中对这项技术的深入应用，以及AJAX这个吸引眼球的名字的提出，使其正式站在了聚光灯下，开始吸引无数人的目光。我们知道Web应用中用户提交表单时就向Web服务器发送一个请求，服务器接收并处理传来的表单，并返回一个新的网页。而前后两个页面中的往往大部分HTML代码是一样的，每次都返回整个页面内容是一种带宽资源的浪费。而AJAX应用仅向服务器发送并取回必须的数据，并在客户端采用JavaScript处理来自服务器响应，更新页面的局部信息。这样不仅浏览器和服务器的数据交换大大减少，而且客户端也可以更加快速地响应用户操作。如果你用Gmail就应该知道，Gmail从来都不刷新页面，所有的请求都是通过AJAX获取数据进行局部更新。AJAX的出现，以及诸如EXTJS、DOJO等一些前端开发框架的出现，也使得单页应用（Single Page Application）在这个时候流行起来。



3.3前端MVC的兴起

在Ajax和SPA的模式下,前后端的分工非常清晰，前后端的关键协作点是 Ajax 接口，规定好交互接口后，前后端工程师就可以根据约定，分头开工，开发环境中通过Mock等方式进行测试，同时在特定时间节点进行前后端集成测试。但是，随着业务功能的愈发复杂（看看现在的Gmail），这种模式本质上和JSP时代的Web开发并无本质区别，只不过是将复杂的业务逻

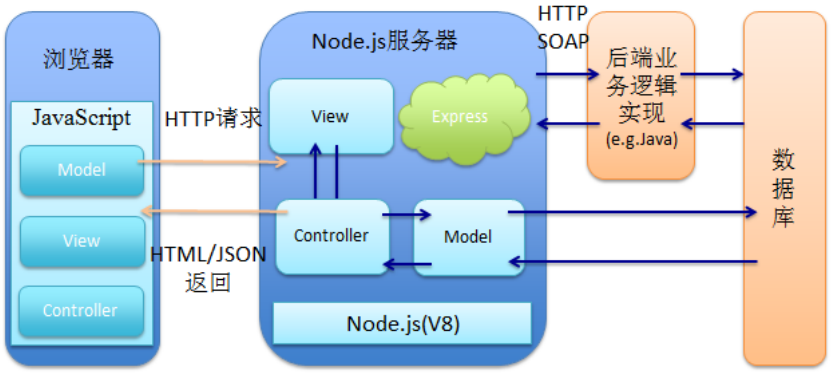
辑从JSP文件转移到了JavaScript文件中而已。现在，对于一个前端功能、交互复杂的SPA，JavaScript代码很容易膨胀（超过10万行）。很自然地，像服务端从JSP向MVC框架转换的过程一样，前端开发也出现了大量的MVC框架，比较典型的包括BackboneJS, AngularJS, EmberJS, KnockoutJS。总的来说，MV*框架的提出是为了解决前端开发的复杂度，提供一套规则组织代码、分层（MVC），通过合理的组织和分层，前端的代码职责明确、清晰，便于开发与测试。



3.4JavaScript在服务器端的逆袭：Node

各大浏览器的竞争，使其引擎的性能不断提升，至今Google V8引擎的性能已经足以运行大型Javascript程序。在V8之上加以网络、文件系统等内置模块，2009年形成了Node.js。

随着Node.js的出现，JavaScript开始拥有在服务端运行的能力，它的异步本质使得Node.js在处理I/O密集型业务中优势凸显，而大多Web业务中I/O性能都是瓶颈。eBay、Yahoo、甚至Microsoft Azure纷纷引入Node.js以提升性能。Node.js的package每天都有几千万的下载量。这对前端工程师来说可是一个好消息，精通JavaScript的他们也能够做服务端开发了！虽然现实中并不是这样美好（服务端开发涉及的不仅仅是语言层面），但一种新的开发模式也因此兴起：浏览器端处理展现层逻辑、而服务端Controller这一层以及相关的模板渲染、路由、数据接口以及Session/Cookie先关处理实际上交给了Nodejs来做。通过Nodejs, 意味着前后端很多代码可以复用（例如数据验证逻辑），在需要SEO的场景下也可以选择服务端模板渲染。



4生态圈爆炸：百家争鸣的时代

4.1前端技术栈

上面这个是@jayli 提出的前端知识结构图，github上有一个开源项目总结了前言的前端所需要的知识：https://github.com/JacksonTian/fks

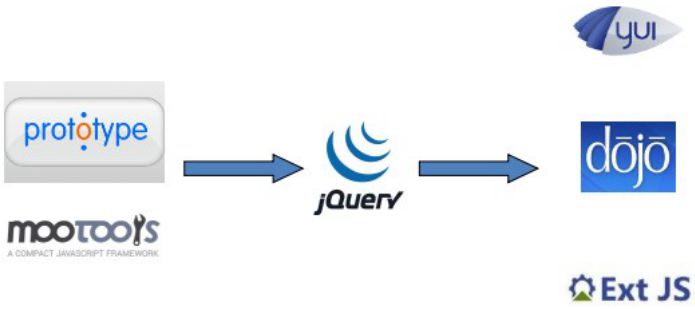
4.2前端生态的原由

花样百出的JS库、CSS库以及不断迭代的W3C标准还有ECMAScript的更新都是为了解决Web开发过程中的几个问题。

4.2.1浏览器兼容性问题

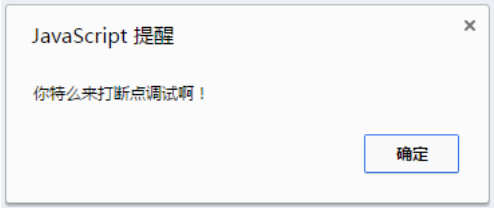
桌面端浏览器大战的硝烟尚未散尽，移动端纷争又起。大厂神仙打架，码农苦不堪言。各种CSS、JS不兼容，坑得码农尽白头。在兼容性这个问题上，IE浏览器最是臭名昭著。

早期出现的各种“JS库”，例如远古的prototype、中古的mootools，到近代的jQuery，再到大规模、紧封装的YUI、Extjs，很大的一个目标就是为了填“兼容性”这个大坑。



4.2.2没有完善的开发、调试、测试、部署工具链

各种后端语言都有自己完善的IDE支持，有的还不止一种IDE，以Java开发为例，市面上常见的就有来自IBM的Eclipse、开源免费的NetBeans、以及来自JetBrains的IntelliJ这些。整个的开发、调试、测试全部在IDE环境里面完成，十分完备。 相比之下，对于前端开发来说，在FireBug、chrome出现之前，前端调试基本上都是靠alert来支持。

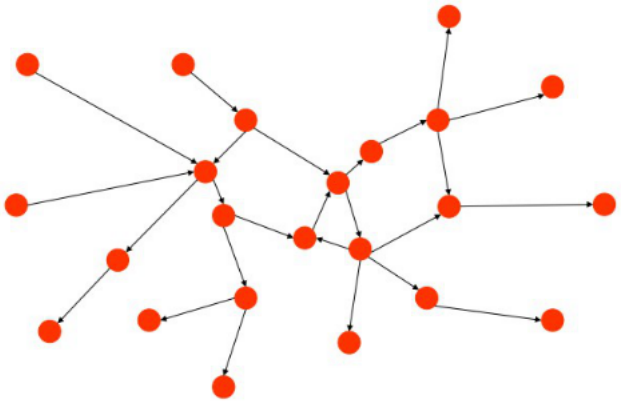


更加坑的是，整个前端的代码压缩、自动化测试、线上部署等等，根本没有神马特别好的工具支持。直到2009年，终于出现了NodeJS这个神器。

终于等到冤大头来填第二个巨坑了，好欣慰啊！ 围绕着NodeJS，出现了大量的前端开发脚手架，例如用来编译CSS的LESS和SASS，用来压缩和混淆的grunt、gulp、webpack，用来把ES6编译成ES5的Babel等等。

因此，从3大坑的角度看，NodeJS的出现，标志着前端开发正式进入工业化时代，刀耕火种的日子一去不复返了！

4.2.3JavaScript缺乏语言级的模块化和组件化机制



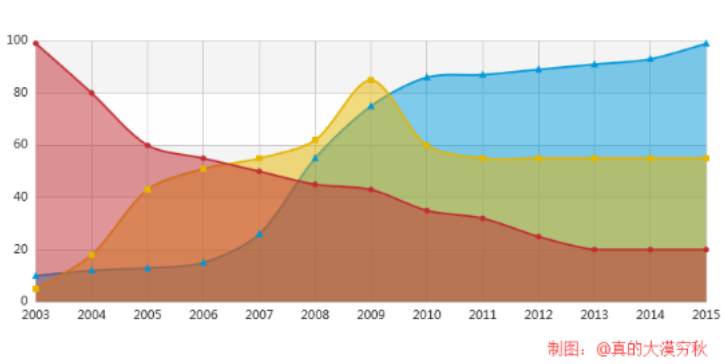
以Java为例，各个类之间的依赖关系最终会形成以上这个“图”数据结构（有向有环）。类之间的依赖关系：

对于前端开发来说，各个业务模块之间的依赖关系同样会形成这样一副“图”。然而很可惜的是，JS这门语言本身并没有对模块化提供有力的支持。JS（ECMA5之前）并没有import机制，对于大规模的application来说，处理模块之间的依赖关系绝对无比蛋疼。直到出现了CMD和AMD这样的理论和思想，这种动态依赖加载的问题才有了比较好的解决方案。

RequireJS:用来填没有动态import机制的坑

目前市面上比较成熟一点的前端框架，都会实现自己的“模块化”机制。

无论是哪一个框架，模块化最底层的机制都是一样的：用JS脚本动态创建script标签，从而诱使浏览器动态加载一段内容（动态加载的内容可以是JS脚本，也可以是普通的字符串，也可以是CSS）。



本节选自：http://damoqiongqiu.iteye.com/blog/2255046

4.3 如今前端生态圈的主要构成

4.3.1 技术基础

HTML

- HTML普通元素，目前HTML5新加了不少新元素
- Canvas 画布
- SVG 矢量图
- MathML 数学公式标记语言
- 多媒体类型 Video、Audio

编程语言（不一定是**JavaScript**）

- JavaScript： JavaScript (JS) 是一种轻量级，解释型的，有着 一等函数 (First-class Function) 的编程语言。一种基于原型、多范式的动态脚本语言，并且支持面向对象、命令式和声明式（如：函数式编程）编程风格。学习材料：https://developer.mozilla.org/zh-CN/docs/Web/JavaScript
- ECMAScript： ECMAScript 是 JavaScript 的标准。截至 2012 年，所有的现代浏览器都完整了支持 ECMAScript 5.1，旧式的浏览器至少支持 ECMAScript 3 标准。在2015年6月17日，ECMA国际组织发布了 ECMAScript 的第六版，该版本正式名称为ECMAScript 2015，但通常被称为 ECMAScript 6 或者ES6。自此，ECMAScript每年发布一次新标准。ES6学习材料：http://es6.ruanyifeng.com/
- TypeScript: TypeScript是一种由微软开发的自由和开源的编程语言。它是JavaScript的一个超集，而且本质上向这个语言添加了可选的静态类型和基于类的面向对象编程。
- CoffeeScript： CoffeeScript是一套JavaScript的转译语言，创建者 Jeremy Ashkenas 戏称它是- JavaScript 的不那么铺张的小兄弟。因为 CoffeeScript 会将类似 Ruby 语法的代码编译成 JavaScript，而且大部分结构都相似，但不同的是 CoffeeScript 拥有更严格的语法。

  编程语言上，ECMAScript是行业标准也是未来趋势。随着ECMASCript6的制定和发布，今后JavaScript会越来越标准化，各大浏览器厂商相继也会支持新标准，以前的浏览器不兼容的情况会渐渐消失。

  #### WebAPI

  Web 提供了各种各样的 API 来完成各种的任务。这些 API 可以用 JavaScript 来访问，令你可以做很多事儿，小到对任意 window 或者 element做小幅调整，大到使用诸如 WebGL 和 Web Audio 的 API 来生成复杂的图形和音效。

- DOM：文档对象模型，构成页面元素的文档对象，通常一个标签就是一个对象，可以使用JavaScript对标签对象进行操作。
- BOM：浏览器对象模型，BOM提供了独立于内容而与浏览器窗口进行交互的对象，核心对象是window。需要注意的是BOM缺乏标准化，JavaScript语法的标准化组织是ECMA，DOM的标准化组织是W3C，所以很多兼容性问题都是出自于浏览器间BOM的标准不一致，比如获取当前窗口的大小，IE、火狐、Chrome的API是不一样的。
- 数据管理API，比如WebStorage、IndexedDB等
- 通信API，比如WebSocket
- 设备API，一般使用在移动设备中

CSS

层叠样式表 (Cascading Style Sheets)，常缩写为 CSS， 是一门 样式表 (stylesheet) 语言，用来描述 HTML、XML（包括各种 XML 语言如 SVG、XHTML）文档的呈现。CSS 描述了在屏幕、电子纸、音频或其它媒体上如何渲染元素。

CSS语法与基础知识包括：权重 (specificity) 与 继承 (inheritance)，盒模型 (box model) 与 外边距合并 (margin collapsing)，堆叠 (stacking) 与 块格式化上下文 (block-formattin contexts)，初始值 (initial)、 计算值 (computed) 与 应用值 (used) 与 实际值 (actual values)，还有 速记（简写）属性 (shorthand properties) 等概念。 学习材料：http://www.runoob.com/css/css-tutorial.html、https://developer.mozilla.org/zh-CN/docs/Web/CSS

CSS预处理

CSS 预处理器是一种语言用来为 CSS 增加一些编程的的特性，无需考虑浏览器的兼容性问题，例如你可以在 CSS 中使用变量、简单的程序逻辑、函数等等在编程语言中的一些基本技巧，可以让你的 CSS 更见简洁，适应性更强，代码更直观等诸多好处。

目前常见的CSS预处理器有：Sass、LESS 和 Stylus

less学习材料：http://lesscss.cn/

4.3.2 工具链

构建工具

构建工具是用来让我们不再做机械重复的事情，解放我们的双手的。对于需要反复重复的任务，例如压缩（minification）、编译、单元测试、linting等，自动化工具可以减轻你的劳动，简化你的工作，比如Java有Maven和Ant来构建项目。

静态语言一般都把构建工具集成到了IDE比如微软的VS基本上涵盖了软件开发生命周期的所有工具。而前端在缺少标准化的很长一段时间里是没有一个强大的组织来做好这件事，所以目前前端的构建工具非常多，甚至很多大公司开发自己的构建工具，比如百度的FIS。

目前比较常用的构建工具和打包工具有这么几种：Grunt、Gulp、Webpack、browserify 这些工具都是用来自动化处理一些任务，其中比较重要的有：编译（非javaScript语言编译成javaScript语言或高版本javascript语言编译成低版本javascript语言；CSS预处理编译成普通CSS；HTML模板编译成HTML；图片等资源文件的编译）、压缩（代码压缩、资源文件的压缩等）、打包（有依赖关系的细碎文件打包成一个或几个文件）、发布（将打包好的项目文件推送至服务器）；

构建工具一般在配置文件中配置好需要执行的任务，开发阶段启动项目或发布版本的时候就可以执行相应的任务组合，比如开发阶段可以执行代码的热更新更改代码后自动编译刷新浏览器看到效果，发布阶段可以执行项目文件推送将新版本代码推送至服务器。

包管理器和**npm script**

npm 是 Node 的模块管理器，功能极其强大。它是 Node 获得成功的重要原因之一。

正因为有了npm，我们只要一行命令，就能安装别人写好的模块 。

```
$ npm install
```

上面说到的构建工具也是需要使用npm安装才能使用的，npm包管理器是前端生态不可或缺的重要工具。

npm script 也是构建工具的一种，可以在项目的package.json中配置一些script命令，并使用npm run+标签的方式来执行配置好的命令已达到项目构建的目的。

IDE和调试工具

IDE

对前端开发整个生命周期集成的最好的工具是jetbrains的 webstorm，jetbrains的IDE系列做的都非常好，除了js IDE webstorm之外，java 的IDEA也很受欢迎。但webstorm是收费产品只有30天试用期，另外webstorm打开速度也不是太快比起普通的文本编辑器还是慢了些。

比较流行的开发方式是文本编辑器+插件，目前最受欢迎的文本编辑器是Sublime和Atom；微软也退出了一款比较不错的编辑器叫做VS Code 使用起来也挺顺手。

调试工具

Chrome、Fire Debug等，目前基本所有的浏览器都带有debug工具，只要按下F12就可以找到并进入调试模式。

4.3.3 流行趋势

前面两节讲的是目前前端开发过程中不可或缺的基础和工具，下面来说说当前前端发展的趋势。 这个是github排行：https://github-ranking.com/ 从机构和项目来看，涉及前端的组织机构和项目占了一大半，不难看出开源软件的趋势越来越向前端靠拢。

其中前十的开源项目中，bootstrap、html5-boilerplate、font-awesome是前端UI相关；jquery、node是代表着前端两个时代的重要SDK性质的库；angular.js、vue.js是代表前端MV*时代的SPA开发框架；d3是前端最流行的可视化库。

关于流行趋势有个小段子：https://getpocket.com/a/read/1439676677

前端发展太快日新月异又处在百家争鸣的时代，各种语言或框架都有可能成为未来的标准，没有必要各各都精通。但各种语言或框架都是为了解决前端发展过程中的那几个问题，让开发变的简单便捷。缕清他们之间的关系就可以知道需要的到底是什么，从纷繁的框架或库中选一些搭配使用。