前端面试宝典

第一部分

1.1 基础部分

1.1.1：技术起源理论题

1、什么是 HTML?

答、HTML 并不是真正的的程序语言，他是一种*标记语言*，用来结构化和含义化你想要放在 web 网站上的那些内容。它由一系列的**元素（elements）**所组成，这些元素可以用来封装你的内容中担任不同工作的各部分和各个角色。

2、什么是 CSS?

答、就像 HTML，CSS 也不是真正的编程语言。它是样式表语言，也就是说，它允许你有选择性的为 HTML 文档的元素添加样式。

3、什么是 JavaScript?

答、JavaScript（缩写：JS）是一门成熟的动态编程语言，当应用于 HTML 文档时，可以在网站上提供动态交互性。

4、其他了解知识！

答、JS 发明者（JS 之父、Brendan Eich 布兰登·艾奇）

前端权威论坛：MDN W3C

前端权威书籍：JS 高级程序设计 JS 权威指南

1.1.2 Html4/5 常见问题

1、HTML5 为什么只需要写 <!DOCTYPE HTML>?

答、（1）HTML5 不基于 SGML，因此不需要对 DTD 进行引用，但是需要 DOCTYPE 来规范浏览器的行为（让浏览器按照它们应该的方式来运行）；（2）HTML4.01 基于 SGML，所以需要对 DTD 进行引用，才能让浏览器知道该文档所使用的文档类型。

2、html5 有哪些新特性、HTML5 新标签的浏览器兼容问题如何处理?

答、html5 新增了以下的几大类元素：

内容元素：article、footer、header、nav、section。

表单控件：calendar、date、time、email、url、search。

控件元素：webworker, websockt, Geolocation。

移出的元素有下列这些：

显现层元素：basefont，big，center，font, s，strike，tt，u。

性能较差元素：frame，frameset，noframes。

HTML5 已形成了最终的标准，概括来讲，它主要是关于图像，位置，存储，多任务等功能的增加。

新增的元素有绘画 canvas ，用于媒介回放的 video 和 audio 元素，本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失，而 sessionStorage 的数据在浏览器关闭后自动删除。

新的技术：canvas,svg,webworker, websocket, Geolocation

3、简述一下你对 HTML 语义化的理解。

答：（1）HTML 语义化让页面的内容结构化，结构更清晰，便于对浏览器、搜索引擎解析；

（2）即使在没有样式 CSS 的情况下也能以一种文档格式显示，并且是容易阅读的；

（3）搜索引擎的爬虫也依赖于 HTML 标记来确定上下文和各个关键字的权重，有利于 SEO；

（4）使阅读源代码的人更容易将网站分块，便于阅读、维护和理解。

4、XHTML 与 HTML 有什么区别?

答、xhtml:1.所有的标记都必须要有一个相应的结束标记

2.所有标签的元素和属性的名字都必须使用小写

3.所有的 XML 标记都必须合理嵌套

4.所有的属性必须用引号""括起来

5.把所有<和&特殊符号用编码表示

6.给所有属性赋一个值=""

7.不要在注释内容中使“--”=""

8.图片必须有说明文字="" <="" code=""/>

5、Doctype 作用？严格模式与混杂模式如何区分？它们有何意义？

（1）<!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器以何种模式来渲染文档。

（2）严格模式的排版和 JS 运作模式是以该浏览器支持的最高标准运行。

（3）在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。

（4）DOCTYPE 不存在或格式不正确会导致文档以混杂模式呈现。

6、介绍一下你对浏览器内核的理解？常见的浏览器内核有哪些？

答、要或者说核心的部分是“Rendering Engine”，可大概译为“渲染引擎”，不过我们一般习惯将之称为“浏览器内核”。负责对网页语法的解释（如标准通用标记语言下的一个应用 HTML、JavaScript）并渲染（显示）网页。 所以，通常所谓的浏览器内核也就是浏览器所采用的渲染引擎，渲染引擎决定了浏览器如何显示网页的内容以及页面的格式信息。不同的浏览器内核对网页编写语法的解释也有不同，因此同一网页在不同的内核的浏

览器里的渲染（显示）效果也可能不同，这也是网页编写者需要在不同内核的浏览器中测试网页显示效果的原因。

常见浏览器内核

Trident 内核：IE,MaxThon,TT,The World,360,搜狗浏览器等。[又称 MSHTML]

Gecko 内核：Netscape6 及以上版本，FF,MozillaSuite/SeaMonkey 等。

Presto 内核：Opera7 及以上。 [Opera 内核原为：Presto，现为、Blink;]

Webkit 内核：Safari,Chrome 等。 [Chrome 的：Blink（WebKit 的分支）]

EdgeHTML 内核：Microsoft Edge。 [此内核其实是从 MSHTML fork 而来，删掉了几乎所有的 IE 私有特性]

7、对 WEB 标准以及 W3C 的理解与认识?

1. 标签闭合、标签小写、不乱嵌套、提高搜索机器人搜索几率、使用外链 css 和 js 脚本、结构行为表现的分离，
2. 文件下载与页面速度更快、内容能被更多的用户所访问、内容能被更广泛的设备所访问、更少的代码和组件，
3. 容易维护、改版方便，不需要变动页面内容、提供打印版本而不需要复制内容、提高网站易用性。

8、Iframe 的作用?

答、用法：

Iframe 是用来在网页中插入第三方页面，早期的页面使用 iframe 主要是用于导航栏这种很多页面都相同的部分，这样可以在切换页面的时候避免重复下载。

优点：便于修改，模块分离，像一些信息管理系统会用到。

但现在基本上不推荐使用。除非特殊需要，一般不推荐使用。

缺点：（1）iframe 的创建比一般的 DOM 元素慢了 1-2 个数量级

（2）iframe 标签会阻塞页面的加载，如果页面的 onload 事件不能及时触发，会让用户觉得网页加载很慢，用户体验不好.在 Safari 和 Chrome 中可以通过 js 动态设置 iframe 的 src 属性来避免阻塞.

（3）iframe 对于 SEO 不友好，替代方案一般就是动态语言的 Incude 机制和 ajax 动态填充内容等.

1.1.3 Css2/3

1、CSS3 新特性有哪些?

答、1. 颜色：新增 RGBA，HSLA 模式

2. 文字阴影（text-shadow）

3.边框： 圆角（border-radius） 边框阴影： box-shadow

4. 盒子模型：box-sizing

5.背景：background-size 设置背景图片的尺寸 background-origin 设置背景图片的原点 background-clip 设置背景图片的裁切区域，以”，”分隔可以设置多背景，用于自适应布局

6.渐变：linear-gradient、radial-gradient

7. 过渡：transition，可实现动画

8. 自定义动画

9. 在 CSS3 中唯一引入的伪元素是 ： selection.

10. 媒体查询，多栏布局

11. border-image

12.2D 转换：transform：translate(x，y) rotate(x，y) skew(x，y) scale(x，y)

13. 3D 转换

2、描述一下 CSS 中的渐进增强，优雅降级之间的区别?

答、优雅降级和渐进增强印象中是随着 css3 流出来的一个概念。由于低级浏览器不支持 css3，但 css3 的效果又太优秀不忍放弃，所以在高级浏览中使用 css3 而低级浏览器只保证最基本的功能。咋一看两个概念差不多，都是在关注不同浏览器下的不同体验，关键的区别是他们所侧重的内容，以及这种不同造成的工作流程的差异。

举个例子:

```
a{
    display：block；
    width：200px；
    height： 100px；
    background：aquamarine；
```

```
    /*我就是要用这个新 css 属性*/
     transition：all 1s ease 0s;
   /*可是发现了一些低版本浏览器不支持怎么吧*/
    /*往下兼容*/
    -webkit-transition：all 1s ease 0s;
    -moz-transition：all 1s ease 0s;
    -o-transition： all 1s ease 0s;
   /*那么通常这样考虑的和这样的侧重点出发的 css 就是优雅降级*/
}
a：hover{
    height：200px;
}
/  *那如果我们的产品要求我们要重低版本的浏览器兼容开始*/
  a{
 /*优先考虑低版本的*/
    -webkit-transition：all 1s ease 0s;
    -moz-transition：all 1s ease 0s;
    -o-transition： all 1s ease 0s;
    /*高版本的就肯定是渐进渐强*/
   transition：all 1s ease 0s;
```

}

“优雅降级”观点认为应该针对那些最高级、最完善的浏览器来设计网站。

“渐进增强”观点则认为应关注于内容本身。

3、对 BFC 规范的理解有哪些?

答、 1）定义：

BFC(Block formatting context)直译为"块级格式化上下文"。它是一个独立的渲染区域，只有 Block-level box 参与， 它规定了内部的 Block-level Box 如何布局，并且与这个区域外部毫不相干。

布局规则：A. 内部的 Box 会在垂直方向，一个接一个地放置。

B. Box 垂直方向的距离由 margin 决定。属于同一个 BFC 的两个相邻 Box 的 margin 会发生重叠。

C. 每个元素的 margin box 的左边， 与包含块 border box 的左边相接触(对于从左往右的格式化，否则相反)。即使存在浮动也是如此。

D. BFC 的区域不会与 float box 重叠。

E. BFC 就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。反之也如此。

F. 计算 BFC 的高度时，浮动元素也参与计算。

3）哪些元素会生成 BFC：

A. 根元素

B. float 属性不为 none

C. position 为 absolute 或 fixed

D. display 为 inline-block， table-cell， table-caption， flex， inline-flex

F. overflow 不为 visible

4、什么是 CSS reset 是什么？normalize.css 是什么？

Reset 重置浏览器的 css 默认属性，浏览器的品种不同，样式不同，然后重置，让他们统一。（暴力的，强制的）

- Normalize.css 是 css reset 的替代方案，保护有用的浏览器默认样式而不是完全去掉它们（温和的，根据浏览器特性的）
- 一般化的样式：为大部分 HTML 元素提供
- 修复浏览器自身的 bug 并保证各浏览器的一致性
- 优化 CSS 可用性：用一些小技巧
- 解释代码：用注释和详细的文档来。

5、你遇到过的兼容问题有哪些？如何解决的？

1、png24 位的图片在 iE6 浏览器上出现背景，解决方案是做成 PNG8.

2、浏览器默认的 margin 和 padding 不同。解决方案是加一个全局的{margin：0;padding：0;}来统一。

3、 IE6 双边距 bug：块属性标签 float 后，又有横行的 margin 情况下，在 ie6 显示 margin 比设置的大。浮动 ie 产生的双倍距离 #itcast{ float：left; width：10px; margin：0 0 0 100px;} 这种情况之下 IE 会产生 20px 的距离，解决方案是在 float 的标签样式控制中加入 _display：inline;将其转化为行内属性。(“_” 这个符号只有 ie6 会识别)

4、 IE 下,可以使用获取常规属性的方法来获取自定义属性,

也可以使用 getAttribute()获取自定义属性;

Firefox 下,只能使用 getAttribute()获取自定义属性。

解决方法：统一通过 getAttribute()获取自定义属性。

5、 IE 下,even 对象有 x,y 属性,但是没有 pageX,pageY 属性;

Firefox 下,event 对象有 pageX,pageY 属性,但是没有 x,y 属性。

解决方法：（条件注释）缺点是在 IE 浏览器下可能会增加额外的 HTTP 请求数。

6、 Chrome 中文界面下默认会将小于 12px 的文本强制按照 12px 显示,

可通过加入 CSS 属性 -webkit-text-size-adjust：none; 解决。

7、超链接访问过后 hover 样式就不出现了 被点击访问过的超链接样式不在具有 hover 和 active 了。解决方法是：改变 CSS 属性的排列顺序：L-V-H-A 、 a：link {} a：visited {} a：hover {} a：active {}

6、清除浮动有几种方式?

答：1、父级 div 定义 height

原理：父级 div 手动定义 height，就解决了父级 div 无法自动获取到高度的问题。 简单、代码少、容易掌握 ，但只适合高度固定的布局.

2、结尾处加空 div 标签 clear：both

原理：在浮动元素的后面添加一个空 div 兄弟元素，利用 css 提高的 clear：both 清除浮动，让父级 div 能自动获取到高度 ，如果页面浮动布局多，就要增加很多空 div，让人感觉很不好 .

3、父级 div 定义 伪类：after 和 zoom

```
/*清除浮动代码*/
.clearfix：after{
content："";
display：block;
visibility：hidden;
height：0;
line-height：0;
clear：both;
}
.clearfix{zoom：1}
```

原理：IE8 以上和非 IE 浏览器才支持：after，原理和方法 2 有点类似，zoom(IE 专有属性)可解决 ie6，ie7 浮动问题 ，推荐使用，建议定义公共类，以减少 CSS 代码。

4、父级 div 定义 overflow：hidden

超出盒子部分会被隐藏，不推荐使用.

5. 双伪元素法：

```
.clearfix：before，.clearfix：after {
                content：  "";
                display：  block;
                clear：both;
        }
        .clearfix {
                zoom：  1；}
```

7、要你出一套适应不同分辨率，不同终端的前端实现方案你有什么思路?

答、流式布局：

使用非固定像素来定义网页内容，也就是百分比布局，通过盒子的宽度设置成百分比来根据屏幕的宽度来进行伸缩，不受固定像素的限制，内容向两侧填充。这样的布局方式，就是移动 web 开发使用的常用布局方式。这样的布局可以适配移动端不同的分辨率设备。

响应式开发：

那么 Ethan Marcotte 在 2010 年 5 月份提出的一个概念，简而言之，就是一个网站能够兼容多个终端。越来越多的设计师也采用了这种设计。

CSS3 中的 Media Query（媒介查询），通过查询 screen 的宽度来指定某个宽度区间的网页布局。

超小屏幕（移动设备）　　768px 以下

小屏设备　　768px-992px

中等屏幕　　992px-1200px

宽屏设备　　1200px 以上

由于响应式开发显得繁琐些，一般使用第三方响应式框架来完成，比如 bootstrap 来完成一部分工作，当然也可以自己写响应式。

8、移动 WEB 和响应式有什么分别?

开发方式	移动web开发+PC开发	响应式开发
应用场景	一般在已经有PC端的网站，开发移动站的时候，只需单独开发移动端。	针对新建站的一些网站，现在要求适配移动端，所以就一套页面兼容各种终端，灵活。
开发	针对性强，开发效率高。	兼容各种终端，效率低，
适配	只适配 移动设备，pad上体验相对较差。	可以适配各种终端
效率	代码简洁，加载快。	代码相对复杂，加载慢。

9、px em rem 取用选择依据?

答、 1）px 像素（Pixel）。绝对单位。像素 px 是相对于显示器屏幕分辨率而言的，是一个虚拟长度单位，是计算机系统的数字化图像长度单位，如果 px 要换算成物理长度，需要指定精度 DPI。

2）em 是相对长度单位，相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。它会继承父级元素的字体大小，因此并不是一个固定的值。

3）rem 是 CSS3 新增的一个相对单位（root em，根 em），使用 rem 为元素设定字体大小时，仍然是相对大小，但相对的只是 HTML 根元素。

4）区别：IE 无法调整那些使用 px 作为单位的字体大小，而 em 和 rem 可以缩放，rem 相对的只是 HTML 根元素。这个单位可谓集相对大小和绝对大小的优点于一身，通过它既可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁反应。目前，除了 IE8 及更早版本外，所有浏览器均已支持 rem。

10、css 权重如何计算?

页面显示样式的优先级取决于其“特殊性”，特殊性越高，就显示最高的，当特殊性相等时，显示后者

特殊性表述为 4 个部分：0,0,0,0

一个选择器的特殊性如下:

- 对于选择器是#id 的属性值,特殊性值为：0,1,0,0
- 对于属性选择器，class 或伪类，特殊性值为：0,0,1,0
- 对于标签选择器或伪元素，特殊性值为：0,0,0,1
- 通配符 ‘*’ 特殊性值为：0,0,0,0
- 内联样式特殊性值为：1,0,0,0

11、水平垂直居中的方式有几种？（至少理解两种）

- 1.absolute + transform：绝对定位加+转换
-
- Demo
-
-
- <style>
- .parent {
- position：relative;
- }
- .child {
- position： absolute;
- left： 50%;
- top：50%;

- transform：translate(-50%, -50%);
- }
- </style>

2.inline-block + text-align + table-cell + vertical-align（单元格方式）

```
<div class="parent">
  <div class="child">Demo</div>
</div>
<style>
  .parent {
    text-align：center;
    display：table-cell;
    vertical-align：middle;
  }
  .child {
    display： inline-block;
  }
</style>
```

3.flex + justify-content + align-items（弹性模型）

```
<div class="parent">
```

```
  <div class="child">Demo</div>
</div>
<style>
  .parent {
    display:  flex;
    justify-content:  center; /*  水平居中  */
    align-items：center; /*垂直居中*/
  }
<style>
```

12、css sprites，如何使用?

答、Css 精灵图，把一堆小的图片整合到一张大的图片（png）上，减轻服务器对图片的请求数量。再利用 css 的“background-image”、“background-repeat”、“background-position”的组合进行背景定位

13、src 与 href 的区别

src（source）指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求 src 资源时会将其指向的资源下载并应用到文档中，如 js 脚本，img 图片和 iframe 等元素。

当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执

行完毕，类似于将所指向资源嵌入当前标签内。

href（hypertext reference/超文本引用）指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的链接，如果我们在文档中添加<link href="common.css" rel="stylesheet"/>那么浏览器会识别该文档为 css 文件，就会并行下载资源并且不会停止对当前文档的处理。

1.1.4 Js 部分

1、说说你对 this 的理解?

答、this 是一个关键字，它代表函数运行时，自动生成的一个内部对象，只能在函数内部使用。

1.作为纯粹的函数调用 this 指向全局对象

2.作为对象的方法调用 this 指向调用对象

3.作为构造函数被调用 this 指向新的对象（new 会改变 this 的指向）

4.apply 调用 this 指向 apply 方法的第一个参数

2、介绍一下 JS 有哪些内置对象。

Object 是 JavaScript 中所有对象的父对象

数据封装类对象：Object、Array、Boolean、Number、String

其他对象：Function、Argument、Math、Date、RegExp、Error

3、表单验证传输的什么数据？明文还是暗文==加密？如何加密？是每一次传输数据，都是加密之后才传输吗？

答、概述：

GET 是从服务器上请求数据，POST 是发送数据到服务器。事实上，GET 方法是把数据参数队列（query string）加到一个 URL 上，值和表单是一一对应的。比如说，name=John。在队列里，值和表单用一个&符号分开，空格用+号替换，特殊的符号转换成十六进制的代码。因为这一队列在 URL 里边，这样队列的参数就能看得到，可以被记录下来，或更改。通常 GET 方法还限制字符的大小（大概是 256 字节 ）。

事实上 POST 方法可以没有时间限制的传递数据到服务器，用户在浏览器端是看不到这一过程的，所以 POST 方法比较适合用于发送一个保密的（比如信用 卡号）或者比较大量的数据到服务器。

区别：

Post 是允许传输大量数据的方法，而 Get 方法会将所要传输的数据附在网址后面，然后一起送达服务器，因此传送的数据量就会受到限制，但是执行效率却比 Post 方法好。

总结：

1、get 方式的安全性较 Post 方式要差些，包含机密信息的话，建议用 Post 数据提交方式；

2、在做数据查询时，建议用 Get 方式；而在做数据添加、修改或删除时，建议用 Post 方式；

所以：

表达如果是向服务器传输数据(如帐号密码等)都是加密数据(post)， 如果只是单单想要从服务器获得数据或者传输的数据并不重要， 可以直接使用明文方式传输(get)

4、如何实现跨域？

解答：

URL	说明
http://www.a.com/b.js	同一域名下
http://www.a.com:8000/a.js	同一域名，不同端口
https://www.a.com/b.js	同一域名，不同协议
http://script.a.com/b.js	主域相同，子域不同
http://a.com/b.js	同一域名，不同二级域名
http://www.a.com/b.js	不同域名

对于端口和协议的不同，只能通过后台来解决。我们要解决的是域名不同的问题。

1.下面是用 php 进行的设置， “*”号表示允许任何域向我们的服务端提交请求、header{"Access-Control-Allow-Origin： *"}

2.JSONP(JSON with Padding 填充式 JSON 或参数式 JSON)

在 js 中，我们虽然不能直接用 XMLHttpRequest 请求不同域上的数据，但是在页面上引入不同域上的 js 脚本文件却是可以的，jsonp 正是利用这个特性来实现的。

JSONP 由两部分组成：回调函数和数据。回调函数是当响应到来时应该在页面中调用的函数，而数据就是传入回调函数中的 JSON 数据。

```
<script type="text/javascript">
    function dosomething(jsondata){
        //处理获得的json数据
    }
</script>
<script src="http://example.com/data.php?callback=dosomething"></script>
```

首先第一个 script 标签定义了一个处理数据的函数；

然后第二个 script 标签载入一个 js 文件，http：//example.com/data.php 是数据所在地址，但是因为是当做 js 来引入的，所以 http：//example.com/data.php 返回的必须是一个能执行的 js 文件；

最后 js 文件载入成功后会执行我们在 url 参数中指定的函数，并且会把我们需要的 json 数据作为参数传入。所以 php 应该是这样的：

```
<?php
$callback = $_GET['callback'];//得到回调函数名
$data = array('a','b','c');//要返回的数据
echo $callback.'('.json_encode($data).')';//输出
?>
```

JSONP 的优缺点

优点：

它的兼容性更好，在更加古老的浏览器中都可以运行，不需要 XMLHttpRequest 或 ActiveX 的支持；

能够直接访问响应文本，支持在浏览器与服务器之间双向通信

缺点：

JSONP 是从其他域中加载代码执行。如果其他域不安全，很可能会在响应中夹带一些恶意代码，而此时除了完全放弃 JSONP 调用之外，没有办法追究。因此在使用不是你自己运维的 Web 服务时，一定得保证它安全可靠。

它只支持 GET 请求而不支持 POST 等其它类型的 HTTP 请求；它只支持跨域 HTTP 请求这种情况，不能解决不同域的两个页面之间如何进行 JavaScript 调用的问题

5、什么是闭包?

答、简单的说，作用域是针对变量的，比如我们创建一个函数 a1，函数里面又包了一个子函数 a2。此时就存在三个作用域：

全局作用域、a1 作用域、a2 作用域；即全局作用域包含了 a1 的作用域，a2 的作用域包含了 a1 的作用域。

当 a1 在查找变量的时候会先从自身的作用域区查找，找不到再到上一级 a2 的作用域查找，如果还没找到就到全局作用域区查找，这样就形成了一个作用域链。

理解闭包首先要理解，js 垃圾回收机制，也就是当一个函数被执行完后，其作用域会被收回，如果形成了闭包，执行完后其作用域就不会被收回。

如果某个函数被他的父函数之外的一个变量引用，就会形成闭包。

闭包的作用，就是保存自己私有的变量，通过提供的接口（方法）给外部使用，但外部不能直接访问该变量。

6、什么是原型链?

答、Javascript 是面向对象的，每个实例对象都有一个__proto_属性，该属性指向它原型对象，这个实例对象的构造函数有一个原型属性 prototype，与实例的__proto__属性

指向同一个对象。当一个对象在查找一个属性的时，自身没有就会根据__proto__ 向它的原型进行查找，如果都没有，则向它的原型的原型继续查找，直到查到 Object.prototype._proto_为 null，这样也就形成了原型链。

7、实现继承的方法有什么?

答、

（1）借用构造函数。也叫伪造对象或经典继承。

思路：在子类构造函数的内部调用超类型构造函数。可以通过使用 apply()和 call()方法在新创建的对象上执行构造函数。

缺点：方法都在构造函数中定义，函数的复用就无从谈起。在超类型的原型中定义的方法，对子类而言也是不可见的，结果所有的类型都只能使用构造函数模式。

```
function SuperType(){
    this.colors = ["red", "blue", "green"];
}

function SubType(){
    //继承了 SuperType
    SuperType.call(this);
}

var instance1 = new SubType();
instance1.colors.push("black");
alert(instance1.colors);    //"red,blue,green,black"

var instance2 = new SubType();
alert(instance2.colors);    //"red,blue,green"
```

（2）组合继承。也叫伪经典继承。指的是将原型链和借用构造函数的技术组合到一起，从而发挥二者之长。

思路：使用原型链实现对原型属性属性和方法的继承，通过借用构造函数来实现实例属性的继承。

优点：既通过在原型上定义方法实现了函数复用，又能保证每一个实例都有它自己的数组。

组合继承避免了原型链和借用构造函数的缺陷，融合了他们的优点，成为 JavaScript 中常用的继承模式。

（3）原型链继承。

思路：借助原型可以基于已有的对象创建对象，同时还不必因此创建自定义类型。

在 object()函数内部，先创建一个临时的构造函数，然后将传入的对象作为这个构造函数的原型，最后返回了这个临时类型的一个新实例。

```
Function object(o){
Function  F(){};
F.prototype=o;
Return new F();
}
```

（4）寄生式继承。

思路：创建一个仅用于封装继承过程的函数，该函数在内部以某种方式来增强对象，最后再像真的是它做了所有的工作一样返回对象。

```
Function createAonter(original){
Var clone=object(original); //通过调用函数创建一个新对象
Clone.sayHi=function(){   //以某种方式来增强这个对象
Alert(“hi”);
}
Return clone;  //返回这个对象
}
```

缺点：使用寄生式继承来为对象添加函数，会由于不能做到函数复用二降低效率，这一点和构造函数模式类似。

（5）寄生组合式继承。是 JavaScript 最常用的继承模式。

思路：通过借用构造函数来继承属性，通过原型链的混成形式来继承方法。

本质上，就是使用寄生式继承来继承超类型的原型，然后再将结果指定给子类型的原型。

开发人员普遍认为寄生组合式继承时引用类型最理想的继承范式。

extend（）方法才用了这样的方式。

8、请列举字符串操作的方法。

charCodeAt 方法返回一个整数，代表指定位置字符的 Unicode 编码；

charAt 方法返回指定索引位置处的字符。如果超出有效范围的索引值返回空字符串；

slice 方法返回字符串的片段；

substring 方法返回位于 String 对象中指定位置的子字符串。

substr 方法返回一个从指定位置开始的指定长度的子字符串。

indexOf 方法返回 String 对象内第一次出现子字符串位置。如果没有找到子字符串，则返回-1；

lastIndexOf 方法返回 String 对象中字符串最后出现的位置。如果没有匹配到子字符串，则返回-1；

search 方法返回与正则表达式查找内容匹配的第一个字符串的位置。

concat 方法返回字符串值，该值包含了两个或多个提供的字符串的连接；

split 将一个字符串分割为子字符串，然后将结果作为字符串数组返回；

9、 AJAX 请求数据步骤是什么？传输的数据是用的 get 还是 post?

答：

```
var xhr;

xhr = new XMLHttpRequest();  //创建一个异步对象

xhr.open("Get",  "test.ashx",  true);//Get 方式括号中的三个参数分别为：1.发送请
```

求的方式 2.y 要请求的页面 3.是否异步

//xhr.open("post"，"test.ashx"，true)；

//xhr.setRequestHeader("Content-Type"， "application/x-www-form-urlencoded")； Post 方式发送数据

//这个回调函数主要用来检测服务器是否把数据返回给异步对象

xhr.setRequestHeader("If-Modified-Since"，"0")；//设置浏览器不使用缓存

xhr.onreadystatechange = function () {

if (xhr.readystate == 4) {

//readyState 属性指出了 XMLHttpRequest 对象在发送/接收数据过程中所处的几个状态。XMLHttpRequest 对象会经历 5 种不同的状态。

//0、未初始化。对象已经创建，但还未初始化，即还没调用 open 方法；

//1、已打开。对象已经创建并初始化，但还未调用 send 方法；

//2、已发送。已经调用 send 方法，但该对象正在等待状态码和头的返回；

//3、正在接收。已经接收了部分数据，但还不能使用该对象的属性和方法，因为状态和响应头不完整；

//4、已加载。所有数据接收完毕

if(xhr.status==200){ //检测服务器返回的响应报文的状态码是否为 200

alert(xhr.responseText)；//服务器返回的 Response 数据

//解析服务器返回的 json 格式的数据

```
            var s=xhr.responseText;
            var json=eval("("+s+")");
            alert(jason.data);
            }
};
};
```

xhr.send(null)；//异步对象发送请求

//xhr.send("txtName=roger&txtPwd=123")； 以 post 方式发送数据

ajax 中 get 和 post 方式请求数据都是明文的。

10、Javascript 内置的常用对象有哪些？并列举该对象常用的方法？（每种记忆三到五个）

Arguments 函数参数集合

arguments[ ] 函数参数的数组

Arguments 一个函数的参数和其他属性

Arguments.callee 当前正在运行的函数

Arguments.length 传递给函数的参数的个数

Array 数组

length 属性 动态获取数组长度

join() 将一个数组转成字符串。返回一个字符串。

reverse() 将数组中各元素颠倒顺序

delete 运算符 只能删除数组元素的值，而所占空间还在，总长度没变(arr.length)。

shift() 删除数组中第一个元素，返回删除的那个值，并将长度减 1。

pop() 删除数组中最后一个元素，返回删除的那个值，并将长度减 1。

unshift() 往数组前面添加一个或多个数组元素，长度要改变。

arrObj.unshift(“a” ， “b，“c”)

push() 往数组结尾添加一个或多个数组元素，长度要改变。arrObj.push(“a” ，“b” ， “c”)

concat( ) 连接数组

slice( ) 返回数组的一部分

sort( ) 对数组元素进行排序

splice( ) 插入、删除或替换数组的元素

toLocaleString( ) 把数组转换成局部字符串

toString( ) 将数组转换成一个字符串

Boolean 布尔对象

Boolean.toString( ) 将布尔值转换成字符串

Boolean.valueOf( ) Boolean 对象的布尔值

Date 日期时间

创建 Date 对象的方法

（1）创建当前(现在)日期对象的实例，不带任何参数

var today = new Date();

（2）创建指定时间戳的日期对象实例，参数是时间戳。

时间戳：是指某一个时间距离 1970 年 1 月 1 日 0 时 0 分 0 秒，过去了多少毫秒值(1 秒=1000 毫秒)。

var timer = new Date(10000)； //时间是 1970 年 1 月 1 日 0 时 0 分 10 秒

（3）指定一个字符串的日期时间信息，参数是一个日期时间字符串

var timer = new Date(“2015/5/25 10：00：00”)；

（4）指定多个数值参数

var timer = new Date(2015+100，4，25，10，20，0)； //顺序为：年、月、日、时、分、秒，年、月、日是必须的。

方法：

Date.getDate( ) 返回一个月中的某一天

Date.getDay( ) 返回一周中的某一天

Date.getFullYear( ) 返回 Date 对象的年份字段

Date.getHours( ) 返回 Date 对象的小时字段

Date.getMilliseconds( ) 返回 Date 对象的毫秒字段

Date.getMinutes( ) 返回 Date 对象的分钟字段

Date.getMonth( ) 返回 Date 对象的月份字段

Date.getSeconds( ) 返回 Date 对象的秒字段

Date.getTime( ) 返回 Date 对象的毫秒表示

Date.getTimezoneOffset( ) 判断与 GMT 的时间差

Date.getUTCDate( ) 返回该天是一个月的哪一天(世界时)

Date.getUTCDay( ) 返回该天是星期几(世界时)

Date.getUTCFullYear( ) 返回年份(世界时)

Date.getUTCHours( ) 返回 Date 对象的小时字段(世界时)

Date.getUTCMilliseconds( ) 返回 Date 对象的毫秒字段(世界时)

Date.getUTCMinutes( ) 返回 Date 对象的分钟字段(世界时)

Date.getUTCMonth( ) 返回 Date 对象的月份(世界时)

Date.getUTCSeconds( ) 返回 Date 对象的秒字段(世界时)

Date.getYear( ) 返回 Date 对象的年份字段(世界时)

Date.parse( ) 解析日期/时间字符串

Date.setDate( ) 设置一个月的某一天

Date.setFullYear( ) 设置年份，也可以设置月份和天

Date.setHours( ) 设置 Date 对象的小时字段、分钟字段、秒字段和毫秒字段

Date.setMilliseconds( ) 设置 Date 对象的毫秒字段

Date.setMinutes( ) 设置 Date 对象的分钟字段和秒字段

Date.setMonth( ) 设置 Date 对象的月份字段和天字段

Date.setSeconds( ) 设置 Date 对象的秒字段和毫秒字段

Date.setTime( ) 以毫秒设置 Date 对象

Date.setUTCDate( ) 设置一个月中的某一天(世界时)

Date.setUTCFullYear( ) 设置年份、月份和天(世界时)

Date.setUTCHours( ) 设置 Date 对象的小时字段、分钟字段、秒字段和毫秒字段(世界时)

Date.setUTCMilliseconds( ) 设置 Date 对象的毫秒字段(世界时)

Date.setUTCMinutes( ) 设置 Date 对象的分钟字段和秒字段(世界时)

Date.setUTCMonth( ) 设置 Date 对象的月份字段和天数字段(世界时)

Date.setUTCSeconds( ) 设置 Date 对象的秒字段和毫秒字段(世界时)

Date.setYear( ) 设置 Date 对象的年份字段

Date.toDateString( ) 返回 Date 对象日期部分作为字符串

Date.toGMTString( ) 将 Date 转换为世界时字符串

Date.toLocaleDateString( ) 回 Date 对象的日期部分作为本地已格式化的字符串

Date.toLocaleString( ) 将 Date 转换为本地已格式化的字符串

Date.toLocaleTimeString( ) 返回 Date 对象的时间部分作为本地已格式化的字符串

Date.toString( ) 将 Date 转换为字符串

Date.toTimeString( ) 返回 Date 对象日期部分作为字符串

Date.toUTCString( ) 将 Date 转换为字符串(世界时)

Date.UTC( ) 将 Date 规范转换成毫秒数

Date.valueOf( ) 将 Date 转换成毫秒表示

Error 异常对象

Error.message 可以读取的错误消息

Error.name 错误的类型

Error.toString( ) 把 Error 对象转换成字符串

EvalError 在不正确使用 eval()时抛出

SyntaxError 抛出该错误用来通知语法错误

RangeError 在数字超出合法范围时抛出

ReferenceError 在读取不存在的变量时抛出

TypeError 当一个值的类型错误时，抛出该异常

URIError 由 URI 的编码和解码方法抛出

Function 函数构造器

Function.apply( ) 将函数作为一个对象的方法调用

Function.arguments[] 传递给函数的参数

Function.call( ) 将函数作为对象的方法调用

Function.caller 调用当前函数的函数

Function.length 已声明的参数的个数

Function.prototype 对象类的原型

Function.toString( ) 把函数转换成字符串

Math 数学对象

Math 对象是一个静态对象

Math.PI 圆周率。

Math.abs() 绝对值。

Math.ceil() 向上取整(整数加 1，小数去掉)。

Math.floor() 向下取整(直接去掉小数)。

Math.round() 四舍五入。

Math.pow(x，y) 求 x 的 y 次方。

Math.sqrt() 求平方根。

Number 数值对象

Number.MAX_VALUE 最大数值

Number.MIN_VALUE 最小数值

Number.NaN 特殊的非数字值

Number.NEGATIVE_INFINITY 负无穷大

Number.POSITIVE_INFINITY 正无穷大

Number.toExponential( ) 用指数计数法格式化数字

Number.toFixed( ) 采用定点计数法格式化数字

Number.toLocaleString( ) 把数字转换成本地格式的字符串

Number.toPrecision( ) 格式化数字的有效位

Number.toString( ) 将一个数字转换成字符串

Number.valueOf( ) 返回原始数值

Object 基础对象

Object 含有所有 JavaScript 对象的特性的超类

Object.constructor 对象的构造函数

Object.hasOwnProperty( ) 检查属性是否被继承

Object.isPrototypeOf( ) 一个对象是否是另一个对象的原型

Object.propertyIsEnumerable( ) 是否可以通过 for/in 循环看到属性

Object.toLocaleString( ) 返回对象的本地字符串表示

Object.toString( ) 定义一个对象的字符串表示

Object.valueOf( ) 指定对象的原始值

RegExp 正则表达式对象

RegExp.exec( ) 通用的匹配模式

RegExp.global 正则表达式是否全局匹配

RegExp.ignoreCase 正则表达式是否区分大小写

RegExp.lastIndex 下次匹配的起始位置

RegExp.source 正则表达式的文本

RegExp.test( ) 检测一个字符串是否匹配某个模式

RegExp.toString( ) 把正则表达式转换成字符串

String 字符串对象

Length 获取字符串的长度。如：var len = strObj.length

toLowerCase() 将字符串中的字母转成全小写。如：strObj.toLowerCase()

toUpperCase() 将字符串中的字母转成全大写。如：strObj.toUpperCase()

charAt(index) 返回指定下标位置的一个字符。如果没有找到，则返回空字符串。

substr() 在原始字符串，返回一个子字符串

substring() 在原始字符串，返回一个子字符串。

区别：'''

“abcdefgh” .substring(2，3) = “c”

“abcdefgh” .substr(2，3) = “cde”

'''

split() 将一个字符串转成数组。

charCodeAt( ) 返回字符串中的第 n 个字符的代码

concat( ) 连接字符串

fromCharCode( ) 从字符编码创建一个字符串

indexOf( ) 返回一个子字符串在原始字符串中的索引值(查找顺序从左往右查找)。如果没有找到，则返回-1。

lastIndexOf( ) 从后向前检索一个字符串

localeCompare( ) 用本地特定的顺序来比较两个字符串

match( ) 找到一个或多个正则表达式的匹配

replace( ) 替换一个与正则表达式匹配的子串
search( ) 检索与正则表达式相匹配的子串
slice( ) 抽取一个子串
toLocaleLowerCase( ) 把字符串转换小写
toLocaleUpperCase( ) 将字符串转换成大写
toLowerCase( ) 将字符串转换成小写
toString( ) 返回字符串
toUpperCase( ) 将字符串转换成大写
valueOf( ) 返回字符串

11、JS 的数据类型有哪些?

简单数据类型：Undefined、Null、Boolean、Number 和 String。

复杂数据类型：Object

12、javaScript 的 Dom 节点操作创建、插入、删除、复制、查找

一、创建节点、追加节点

1、createElement（标签名）创建一个元素节点（具体的一个元素）。

2、createTextNode（节点文本内容）创建一个文本节点

3、createDocumentFragment()　//创建一个 DOM 片段

4、appendChild（节点）追加一个节点。

二、插入节点

1、appendChild（节点）也是一种插入节点的方式，还可以添加已经存在的元素，会将其元素从原来的位置移到新的位置。

2、insertBefore（a,b）是参照节点，意思是 a 节点会插入 b 节点的前面。

三、删除、移除节点

1、removeChild(节点) 删除一个节点，用于移除删除一个参数（节点）。其返回的被移除的节点，被移除的节点仍在文档中，只是文档中已没有其位置了。

四、复制节点

cloneNode() 方法，用于复制节点， 接受一个布尔值参数， true 表示深复制（复制节点及其所有子节点）， false 表示浅复制（复制节点本身，不复制子节点）

五、替换节点

1、replaceChild(插入的节点，被替换的节点），用于替换节点，接受两个参数，第一参数是要插入的节点，第二个是要被替换的节点。返回的是被替换的节点。

六、查找节点

1、getElementsByTagName() //通过标签名称

2、 getElementsByName() //通过元素的 Name 属性的值(IE 容错能力较强，会得到一个数组，其中包括 id 等于 name 值的)

3、getElementById() //通过元素 Id，唯一性

13、请说说事件委托机制？这样做有什么好处？

答、事件委托，就是某个事件本来该自己干的，但是自己不干，交给别人来干。就叫事件委托。打个比方：一个 button 对象，本来自己需要监控自身的点击事件，但是自己不来监控这个点击事件，让自己的父节点来监控自己的点击事件。

好处：

A，提高性能：列如，当有很多 li 同时需要注册事件的时候，如果使用传统方法来注册事件的话，需要给每一个 li 注册事件。然而如果使用委托事件的话，就只需要将事件委托给该一个元素即可。这样就能提高性能。

B，新添加的元素还会有之前的事件；

14、谈谈你对 jquery 的理解

JQuery 是继 prototype 之后又一个优秀的 Javascript 库。它是轻量级的 js 库 ，它兼容 CSS3，还兼容各种浏览器（IE 6.0+， FF1.5+， Safari 2.0+， Opera 9.0+），jQuery2.0 及后续版本将不再支持 IE6/7/8 浏览器。jQuery 使用户能更方便地处理 HTML（标准通用标记语言下的一个应用）、events、实现动画效果，并且方便地为网站提供 AJAX 交互。

jQuery 还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得很详细，同时还有许多成熟的插件可供选择。jQuery 能够使用户的 html 页面保持代码和 html 内容分离，也就是说，不用再在 html 里面插入一堆 js 来调用命令了，只需要定义 id 即可。

jQuery 是一个兼容多浏览器的 javascript 库，核心理念是 write less，do more(写得更少，做得更多)。 jQuery 是免费、开源的，使用 MIT 许可协议。jQuery 的语法设计可以使开发更加便捷，例如操作文档对象、选择 DOM 元素、制作动画效果、事件处理、使用 Ajax 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。

关于 jQuery 的内部封装原理

1、为了防止全局变量污染，把 jQuery 的代码写在一个自调函数中，

2、咱们平常使用的$实际上 jQuery 对外暴漏的一个工厂函数，

3、而构造函数在 jQuery 的内部叫 init，并且这个构造函数还被添加到了 jQuery 的原型中。当我们调用工厂函数的时候 返回的其实是一个构造函数的实例

4、jQuery 为了让第三方能够对其功能进行扩展，所以把工厂函数的原型与构造函数的原型保持了一致。这样子对外暴漏工厂函数，即可对原型进行扩展。

15、call 和 apply 的区别

答、 它们的共同之处：都“可以用来代替另一个对象调用一个方法，将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。”
它们的不同之处：
Apply：最多只能有两个参数——新 this 对象和一个数组 argArray。如果给该方法传递多个参数，则把参数都写进这个数组里面，当然，即使只有一个参数，也要写进数组里面。如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj， 并且无法被传递任何参数。
Call：则是直接的参数列表，主要用在 js 对象各方法互相调用的时候，使当前 this 实例指针保持一致，或在特殊情况下需要改变 this 指针。如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。

更简单地说，apply 和 call 功能一样，只是传入的参数列表形式不同如：func.call(func1，var1，var2，var3)对应的 apply 写法为：func.apply(func1，[var1，var2，var3])。

16、原生 JS 的 window.onload 与 Jquery 的 $(document).ready(function () {})，$(function () {})有什么不同?

答:

1.执行时间

window.onload 必须等到页面内包括图片的所有元素加载完毕后才能执行。

$(document).ready()是 DOM 结构绘制完毕后就执行，不必等到加载完毕。

2.编写个数不同

window.onload 不能同时编写多个，如果有多个 window.onload 方法，只会执行一个

$(document).ready()可以同时编写多个，并且都可以得到执行

3.简化写法

window.onload 没有简化写法

$(document).ready(function(){})可以简写成$(function(){});

17、在 JS 的计时器运行原理是怎样的，为什么可以触发计时效果？计时器是多线程吗?

答:

1. javascript 引擎只有一个线程，强迫异步事件排队等待被执行。

2. setTimeout 和 setInterval 本质上不同的地方是他们如何执行异步代码的。

3. 如果一个定时器正在执行的时候被阻塞了，那么它将会被推迟到下一个可能的执行点，这既是使得延迟时间有可能会超过声明定时器时设置的值。

4. Interval 如果有足够的时间来执行（大于制定的延迟），那么它将会无延迟的一个紧接着一个执行。

原理:

计时器通过设定一定的时间段（毫秒）来异步的执行一段代码。因为 Javascript 是一个单线程语言，计时器提供了一种绕过这种语言限制来执行代码的能力。

总结:

计时器是单线程的， 需要等待上一个执行完， 如果上一个没有执行完， 下一个需要延迟执行， 直到上一个执行完。

18、JavaScript 中的作用域、预解析与变量声明提升?

作用域:

就是变量的有效范围。

如何检测一个变量的作用域:在指定的区域内使用这个变量，如果不报错，说明这个变量的作用域包含此区域。

函数作用域:只有函数能够划分变量的作用域，这种作用域的规则就叫函数作用域。如果在函数内访问一个变量，优先找局部变量和形参,如果没有找到，去定义该函数的环境中查找，直到全局为止。

在 ES6 之前，只有函数可以划分变量的作用域，所以在函数的外面无法访问函数内的变量。在 ES6 之前，没有块级作用域的概念，所以在代码块的外面可以访问代码块内的变量

块级作用域：凡是代码块就可以划分变量的作用域，这种作用域的规则就叫块级作用域。

块级作用域 函数作用域 词法作用域

之间的区别：

1、块级作用域和函数作用域描述的是，什么东西可以划分变量的作用域

2、词法作用域描述的是，变量的查找规则。

之间的关系：

（1）、 块级作用域 包含 函数作用域。

（2）、 词法作用域 与 块级作用域、函数作用域之间没有任何交集，

他们从两个角度描述了作用域的规则。

ES6 之前 js 采用的是函数作用域+词法作用域，ES6 js 采用的是块级作用域+词法作用域 。

预解析:

在代码整体执行之前，先解析一部分。

预解析之后，代码才会从上往下依次整体执行，但是预解析执行过的代码不会

重复执行。

js 预解析干了什么事：js 中预解析会把声明部分的代码预先执行。

声明相关的代码可以分为两部分：

1、 变量声明

通过 var 关键字定义的变量。

2、函数声明

通过 function 关键字声明的函数

预解析时如果遇到重复的变量声明，那么忽略。

预解析时如果遇到重复的函数声明，保留后面的函数。

预解析时如果遇到变量与函数重名的情况，保留函数。

变量声明提升：

使用 var 关键字定义的变量，被称为变量声明；

函数声明提升的特点是，在函数声明的前面，可以调用这个函数。

19、javascript 的 typeof 返回哪些数据类型

typeof 一般判断基本数据类型。是一个操作符而不是函数，圆括号可有可无。

typeof 返回值有：string，number，boolean，undefined，object ，function，

基本数据类型：Boolean、Number、String、Undefined、Null

基本数据类型中数字，字符串，布尔类型返回其对类型

undefined 返回 undefined

九大内置构造函数及其他所有函数返回 function；

其他所有复杂类型对象和 null 返回 object

```
alert(typeof [1, 2]); //object
alert(typeof 'leipeng'); //string
var i = true;
alert(typeof i); //boolean
alert(typeof 1); //number
var a;
alert(typeof a); //undefined
```

```
function a(){;};
alert(typeof a) //function
```

20、简述列举文档对象模型 DOM 里 document 的常用的查找访问节点的方法并做简单说明

Document.getElementById 根据元素 id 查找元素
Document.getElementByName 根据元素 name 查找元素
Document.getElementTagName 根据指定的元素名查找元素

21、简述创建函数的几种方式

第一种（函数声明）：

```
function sum1(num1,num2){
   return num1+num2;
}
```

第二种（函数表达式）

```
var sum2 = function(num1,num2){
   return num1+num2;
}
```

第三种（函数对象方式）

```
var sum3 = new Function("num1","num2","return num1+num2");
```

22、Javascript 创建对象的几种方式?

1. 简单对象的创建 使用对象**字面量**的方式{}

创建一个对象（最简单，好理解，推荐使用）

代码如下

```
var Cat = {};//JSON
Cat.name="kity";//添加属性并赋值
Cat.age=2;
Cat.sayHello=function(){
 alert("hello "+Cat.name+",今年"+Cat["age"]+"岁了");//可以使用 “.” 的方式访问属性，也可以使用 HashMap 的方式访问
}
Cat.sayHello();//调用对象的（方法）函数
```

2.用 function(函数)来模拟 class

2.1 创建一个对象，相当于 new 一个类的实例(无参构造函数)

代码如下

```
function  Person(){

}
var  personOne=new  Person();//定义一个 function，如果有 new 关键字去"实例化",那么该 function 可以看作是一个类
personOne.name="dylan";
personOne.hobby="coding";
personOne.work=function(){
alert(personOne.name+"  is  coding  now...");
}

personOne.work();
```

2.2 可以使用有参构造函数来实现，这样定义更方便，扩展性更强（推荐使用）

代码如下

```
function  Pet(name,age,hobby){
   this.name=name;//this 作用域：当前对象
   this.age=age;
```

```
    this.hobby=hobby;
    this.eat=function(){
        alert("我叫"+this.name+",我喜欢"+this.hobby+",也是个吃货");
    }
}
var maidou =new Pet("麦兜",5,"睡觉");//实例化/创建对象

maidou.eat();//调用 eat 方法(函数)
```

3.使用工厂方式来创建（Object 关键字）

代码如下：

```
var wcDog =new Object();
wcDog.name="旺财";
wcDog.age=3;
wcDog.work=function(){
    alert("我是"+wcDog.name+",汪汪汪.....");
}

wcDog.work();
```

4.使用原型对象的方式　prototype 关键字

代码如下：

```
function  Dog(){

}
Dog.prototype.name="旺财";
Dog.prototype.eat=function(){
alert(this.name+"是个吃货");
}
var  wangcai  =new  Dog();
wangcai.eat();
```

5.混合模式(原型和构造函数)

代码如下：

```
function  Car(name,price){
  this.name=name;
  this.price=price;
```

```
}
Car.prototype.sell=function(){
   alert("我是"+this.name+"，我现在卖"+this.price+"万元");
  }

var  camry  =new  Car("凯美瑞",27);
camry.sell();
```

6.动态原型的方式(可以看作是混合模式的一种特例)

代码如下:

```

function  Car(name,price){
  this.name=name;
  this.price=price;
  if(typeof  Car.sell=="undefined"){
   Car.prototype.sell=function(){
    alert("我是"+this.name+"，我现在卖"+this.price+"万元");
   }
Car.sell=true;
```

```
  }
}

var camry =new Car("凯美瑞",27);
camry.sell();
```

以上几种，是 javascript 中最常用的创建对象的方式。

23、js 延迟加载的方式有哪些?

1. defer 和 async（异步加载）

   当浏览器碰到 script 脚本的时候:

   1. <script src="script.js"></script>

      没有 defer 或 async，浏览器会立即加载并执行指定的脚本，“立即”指的是在渲染该 script 标签之下的文档元素之前，也就是说不等待后续载入的文档元素，读到就加载并执行。

   2. <script async src="script.js"></script>

      有 async，加载和渲染后续文档元素的过程将和 script.js 的加载与执行并行进行（异步）。

   3. <script defer src="myscript.js"></script>

有 defer，加载后续文档元素的过程将和 script.js 的加载并行进行（异步），但是 script.js 的执行要在所有元素解析完成之后，DOMContentLoaded 事件触发之前完成。

从实用角度来说，**首先把所有脚本都丢到 </body> 之前是最佳实践，因为对于旧浏览器来说这是唯一的优化选择，此法可保证非脚本的其他一切元素能够以最快的速度得到加载和解析。**

补充：图示

蓝色线代表网络读取，红色线代表执行时间，这俩都是针对脚本的；绿色线代表 HTML 解析。

此图告诉我们以下几个要点：

1. *defer* 和 *async* 在网络读取（下载）这块儿是一样的，都是异步的（相较于 HTML 解析）
2. 它俩的差别在于脚本下载完之后何时执行，显然 *defer* 是最接近我们对于应用脚本加载和执行的要求的

3. 关于 ***defer***，此图未尽之处在于它是按照加载顺序执行脚本的，这一点要善加利用
4. ***async*** 则是一个乱序执行的主，反正对它来说脚本的加载和执行是紧紧挨着的，所以不管你声明的顺序如何，只要它加载完了就会立刻执行
5. 仔细想想，*async* 对于应用脚本的用处不大，因为它完全不考虑依赖（哪怕是最低级的顺序执行），不过它对于那些可以不依赖任何脚本或不被任何脚本依赖的脚本来说却是非常合适的，最典型的例子：Google Analytics

2. 动态创建 DOM 方式（创建 script，插入到 DOM 中，加载完毕后 callBack）

创建 script，插入到 DOM 中，加载完毕后 callBack，见代码

代码如下：（有兼容性封装）

```
function loadScript(url, callback){
var script = document.createElement_x("script")
script.type = "text/javascript";
if (script.readyState){ //IE
script.onreadystatechange = function(){
if (script.readyState == "loaded" ||
script.readyState == "complete"){
```

```
script.onreadystatechange = null;
callback();
}
};
} else { //Others:  Firefox, Safari, Chrome, and Opera
script.onload = function(){
callback();
};
}
script.src = url;
document.body.appendChild(script);
}
```

3. 按需异步载入 js

默认情况 javascript 是同步加载的，也就是 javascript 的加载时阻塞的，后面的元素要等待 javascript 加载完毕后才能进行再加载，对于一些意义不是很大的 javascript，如果放在页头会导致加载很慢的话，是会严重影响用户体验的。

细节补充：

(1) defer，只支持 IE

defer 属性的定义和用法（我摘自 w3school 网站）

defer 属性规定是否对脚本执行进行延迟，直到页面加载为止。

有的 javascript 脚本 document.write 方法来创建当前的文档内容，其他脚本就不一定是了。

如果您的脚本不会改变文档的内容，可将 defer 属性加入到 <script> 标签中，以便加快处理文档的速度。因为浏览器知道它将能够安全地读取文档的剩余部分而不用执行脚本，它将推迟对脚本的解释，直到文档已经显示给用户为止。

示例：

代码如下：

```
<script type="text/javascript" defer="defer">
alert(document.getElementById("p1").firstChild.nodeValue);
</script>
```

(2) async：

async 的定义和用法(是 HTML5 的属性)

async 属性规定一旦脚本可用，则会异步执行。

示例：

代码如下：

```
<script type="text/javascript" src="demo_async.js" async="async"></script>
```

注释：async 属性仅适用于外部脚本（只有在使用 src 属性时）。

注释：有多种执行外部脚本的方法：

•如果 async="async" 脚本相对于页面的其余部分异步地执行（当页面继续进行解析时，脚本将被执行）

•如果不使用 async 且 defer="defer" 脚本将在页面完成解析时执行

•如果既不使用 async 也不使用 defer 在浏览器继续解析页面之前，立即读取并执行脚本

24、ECMAScript 对象的继承结构

继承的规律：

1 对象继承的终点是 Object.prototype

2 所有函数默认的显示原型（即函数的 prototype） 都继承 Object.prototype

3 谁的实例，这个实例就继承谁的 prototype

3.1 所有的函数，都被看作是 Function 的实例，所以都继承 Function.prototype

3.2 所有的数组，都被看作是 Array 的实例，所以都继承 Array.prototype

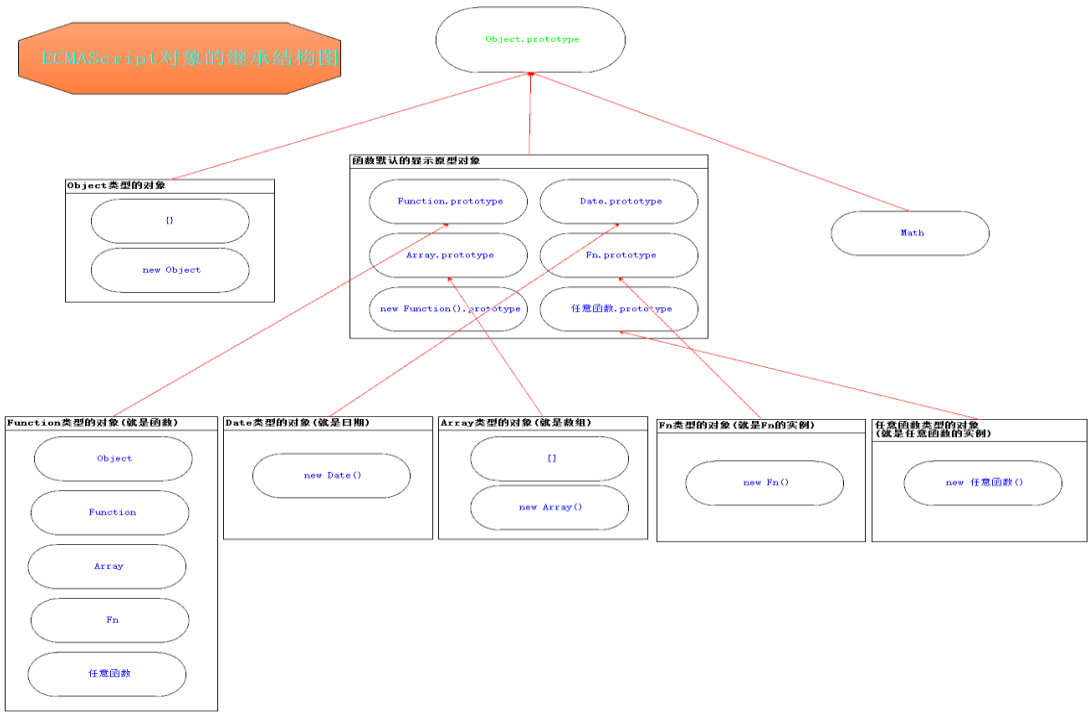
3.3 所有的正则，都被看作是 RegExp 的实例，所以都继承 RegExp.prototype

Eg:function fn() {}

1 fn 是 Function 的实例，所以继承 Function.prototype

2 fn.prototype 继承 Object.prototype

3 new fn() 是 fn 的实例，所以继承 fn.prototype



25、什么是事件冒泡/捕获?

答：

事件冒泡：子元素事件的触发会影响父元素事件；

开关事件冒泡：

A，开启事件冒泡：element.addEventListener(eventName，handler，false)；

B，关闭事件冒泡：假设传统方式事件的返回值为 e，就可以通过 e.stopPropagation()来关闭事件冒泡；

事件捕获：父元素的事件会影响子元素的事件；

开启事件捕获：element.addEventListener(eventName，handler，true)

26、如何阻止事件冒泡和默认事件?

举例说明什么是事件冒泡，何时阻止事件冒泡：

如果<p>是在 里面，那么呢，<P>有一个 onclick 事件， 也有 onclick 事件，为了触发<P>的点击事件时，不触发父元素的点击事件，那么就需要调用如下函数，阻止事件冒泡。

阻止事件冒泡函数：

```
function stopBubble(e){
  if(e&&e.stopPropagation){//非 IE
   e.stopPropagation();
  }
  else{//IE
```

```
    window.event.cancelBubble=true;
  }
}
```

阻止默认事件：

```
function stopDefault( e ) {
        //阻止默认浏览器动作(W3C)
        if ( e && e.preventDefault )
            e.preventDefault();
        //IE 中阻止函数器默认动作的方式
        else
            window.event.returnValue = false;
        return false;
    }
```

27、面向对象和类的区别?

答：简单的说类是对象的模版。

在 js 中没有类， 所以在 js 中所谓的 类 就是构造函数， 对象就是由构造函数创建出来的实例对象。面向对象就是使用面向对象的方式处理问题， 面向对象是对面向过程进行封装。

面向对象有三大特性

抽象性， 需要通过核心数据和特定环境才能描述对象的具体意义

封装性， 封装就是将数据和功能组合到一起， 在 js 中对象就是键值对的集合， 对象将属性和方法封装起来， 方法将过程封装起来

继承性， 将别人的属性和方法成为自己的， 传统继承基于模板(类)， js 中继承基于构造函数

28、简述 for in 循环的特点及使用场景?

答：for...in 语句用于对数组或者对象的属性进行循环操作。

```
for ... in 循环中的代码每执行一次，就会对数组的元素或者对象的属性进行一次操作。
for (变量 in 对象){
    在此执行代码
}
```

“变量”用来指定变量，指定的变量可以是数组元素，也可以是对象的属性。

注意：for in 循环不会按照属性的下标来排列输出。

对象的概念，面向对象编程的程序实际就是多个对象的集合，我们可以把所有的事物都抽象成对象，在程序设计中可以看作：对象=属性+方法。属性就是对象的数据，而方法就是对象的行为。

类的概念，类是对象的模版，而对象是类的实例化。举个例子，汽车设计图可以看作是类，而具体的汽车就是对象。再比如有一个类是表示人，然后可以通过人这个模版来实例化出张三、李四。

29、例举强制类型转换和隐式类型转换?

答：强制

转化成字符串 toString() String()

转换成数字 Number()、 parseInt()、 parseFloat()

转换成布尔类型 Boolean()

隐式

拼接字符串 例子 var str = "" + 18

- / % === ==

30、split() join() 的区别

Split()是把一串字符（根据某个分隔符）分成若干个元素存放在一个数组里

即切割成数组的形式;

join() 是把数组中的字符串连成一个长串，可以大体上认为是 Split()的逆操作

31、数组方法 pop() push() unshift() shift()

Push()尾部添加 pop() 删除并返回数组的最后一个元素

Unshift() 头部添加 shift() 头部删除

32、怎样判断一个 JavaScript 变量是 array 还是 obiect?

1、如果你只是用 typeof 来检查该变量，不论是 array 还是 object，都将返回 'object'。

此问题的一个可行的答案是检查该变量是不是 object，并且检查该变量是否有数字长度（当为空 array 时长度也可能为 0）。

然而，参数对象【arguments object】（传给制定函数的所有参数），也可能会适用于上述方法，技术上来说，参数对象并不是一个 array。

此外，当一个对象有 a.length 属性的时候，这个方法也不成立。

代码如下:

```
// Real array 正在的数组
var my_array = [];
// Imposter! 冒名顶替的!
var my_object = {};
my_object.length = 0;
// Potentially faulty 潜在的错误
function is_this_an_array(param) {
if (typeof param === 'object' && !isNaN(param.length)) {
console.log('Congrats, you have an array!');
}
else {
console.log('Bummer, not an array');
}
}
// Works 成功
is_this_an_array(my_array);
// Works, but is incorrect 成功了，但是不正确
is_this_an_array(my_object);
```

2、回答这个问题的另一个答案是用一个更加隐蔽的方法，调用 toString( )方法试着将该变量转化为代表其类型的 string。

该方法对于真正的 array 可行；参数对象转化为 string 时返回[object Arguments]会转化失败；此外，

对于含有数字长度属性的 object 类也会转化失败。

代码如下:

```
// Real array  真正的数组
var my_array = [];
// Imposter!  冒名顶替的!
var my_object = {};
my_object.length = 0;
// Rock solid  坚如磐石（检验函数）
function is_this_an_array(param) {
if (Object.prototype.toString.call(param) === '[object Array]') {
console.log('Congrats, you have an array!');
}
else {
console.log('Bummer, not an array');
```

```
}
}
// Works 成功了
is_this_an_array(my_array);
// Not an array, yay! 不是数组（array）！
is_this_an_array(my_object);
```

3、此外，可能在不可靠的多框架 DOM 环境中，instanceof 是个完美合适的操作。

扩展阅读："Instanceof Considered Harmful…"

http://perfectionkills.com/instanceof-considered-harmful-or-how-to-write-a-robust-isarray

代码如下:

```
var my_array = [];
if (my_array instanceof Array) {
console.log('Congrats, you have an array!');
}
```

4、对于 Javascript 1.8.5（ECMAScript 5），变量名字.isArray( )可以实现这个目的

https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Global_Objects/Array/isArray

代码如下:

```
var my_array = [];
if (Array.isArray(my_array)) {
console.log('Congrats, you have an array!');
}
```

33、解释 jsonp 的原理，以及为什么不是真正的 ajax

jsonp 的原理：动态创建 script 标签，回调函数

JSONP(JSON with Padding)是 JSON 的一种“使用模式”，可用于解决主流浏览器的跨域数据访问的问题。利用 <script> 元素的这个开放策略，网页可以得到从其他来源动态产生的 JSON 资料，而这种使用模式就是所谓的 JSONP。用 JSONP 抓到的资料并不是 JSON，而是任意的 JavaScript，用 JavaScript 直译器执行而不是用 JSON 解析器解析。

jsonp 为什么不是 ajax：

ajax 和 jsonp 这两种技术在调用方式上“看起来”很像，目的也一样，都是请求一个 url，然后把服务器返回的数据进行处理，因此 jquery 和 ext 等框架都把 jsonp 作为 ajax 的一种形式进行了封装；

2、但 ajax 和 jsonp 其实本质上是不同的东西。ajax 的核心是通过 XmlHttpRequest 获取非本页内容，而 jsonp 的核心则是动态添加<script>标签来调用服务器提供的 js 脚本。

3、所以说，其实 ajax 与 jsonp 的区别不在于是否跨域，ajax 通过服务端代理一样可以实现跨域，jsonp 本身也不排斥同域的数据的获取。

4、还有就是，jsonp 是一种方式或者说非强制性协议，如同 ajax 一样，它也不一定非要用 json 格式来传递数据，如果你愿意，字符串都行，只不过这样不利于用 jsonp 提供公开服务。

总而言之，jsonp 不是 ajax 的一个特例，哪怕 jquery 等巨头把 jsonp 封装进了 ajax，也不能改变着一点！

34、JavaScript 的事件流模型都有什么，以及怎么阻止他们?

答：1、原始事件模型

普通的事件绑定，比如事件赋值，按钮上绑定事件等

2、DOM 事件模型

addEventListener("eventType"，"handler"，"true!false");

removeEventListner("eventType"，"handler"，"true!false");

气泡模型（与 ie 有点区别），冒泡和捕获

3、IE 模型

气泡模型

attachEvent("eventType"，"handler")

detachEvent("eventType"，"handler")

与 dom2 不同的是 eventType 有 on 前缀

35、Javascript 中的垃圾回收机制

答：在 Javascript 中，如果一个对象不再被引用，那么这个对象就会被 GC 回收。如果两个对象互相引用，而不再被第 3 者所引用，那么这两个互相引用的对象也会被回收。因为函数 a 被 b 引用，b 又被 a 外的 c 引用，这就是为什么 函数 a 执行后不会被回收的原因。

36、请解释一下 javascript 的同源策略

为什么会有同源策略?

我们都知道 JavaScript 可以操作 web 文档的内容，试想，如果不对这一点加以限制，那么 JS 可以做的事情就太多了，危险性也太高，所以就针对它可以操作哪些文档的内容有了一个限制，这个限制就是同源策略。

同源策略在什么情况下会起作用呢?

当 web 页面使用多个<iframe>元素或者打开其他浏览器窗口的时候，这一策略就会起作用。

同源策略的含义：

脚本只能读取和所属文档来源相同的窗口和文档的属性。

这里就涉及到了一个浏览器如何判断两者是否同源以及如何判断脚本来源的问题。

注意一点：脚本本身的来源并不作为判断是否同源的依据，而是将脚本所属文档的来源作为判断依据。

1. 判断脚本来源

例如：文档 A 中通过 script 的 src 引用了一个外部脚本，这个脚本是 google 提供的，也是从 google 的主机上加载到文档 A 中的，那么这个脚本的所属文档是谁呢，答案是文档 A。

2. 判断是否同源

理解了脚本来源，接着理解怎么判断是否同源：如果两个文档在协议、主机以及载入文档的 URL 端口这三点中有一点不同，就认为他们不同源。

37、事件绑定和普通事件有什么区别

事件绑定是指把事件注册到具体的 DOM 元素之上，普通事件指的是可以用来注册的事件

事件绑定例子：

DOM 元素.addEventListener(事件类型，事件处理函数)

普通事件例子：

DOM 元素.事件类型 = function(){事件处理函数}

对于事件绑定来说，同一个 DOM 元素上面绑定同样的事件类型时，可以绑定多个事件处理函数，实际解决了多人开发情况下造成的时间处理覆盖的情况（污染），而如果使用普通事件来解决这个问题的时候会造成后写的代码将前写的代码进行了重写操作，造成环境污染。

实际开发的时候可以视情况而定，来选择绑定事件的方式。

38、怎么判断一个变量是否 null/undefined

Undefined:

```
if (typeof(reValue) == "undefined") {
    alert("undefined");
}
```

typeof 返回的是字符串，有六种可能："number"、"string"、"boolean"、"object"、"function"、 "undefined"

Null:

```
var exp = null;
if (!exp && typeof(exp)!=" undefined"  && exp!=0)
{
alert( "is null" );
}
```

我们在 DOM 应用中，一般只需要用（!exp）来判断就可以了，因为 DOM 应用中，可 能返回 null，可能返回 undefined，如果具体判断 null 还是 undefined 会使程序过于复杂。

39、”==”和“===”的不同

”==”判断值是否相等，会自动转换类型；

“===”是判断值及类型是否完全相等，不会自动转换类型。

40、Javascript 中 callee 和 caller 的作用?

caller 是返回一个对函数的引用，该函数调用了当前函数；

```
function a(){
    b();
};
function b(){
    alert(b.caller);
};
a(); //结果就是弹出函数 a 和内容
```

callee 是返回正在被执行的 function 函数，也就是所指定的 function 对象的正文。

先来看一个阶乘函数：

可以看到用到了递归函数

但是问题来了，要是改变了函数名，里面的函数名也要随着改变，这样很不方便所以我们用 callee 来试试

```
function Aaa(n){
    if(n<=1){
        return 1;
    }else{
        return n*Aaa(n-1);
    }
}
function Aaa(n){
    if(n<=1){
        return 1;
    }else{
        return n*arguments.callee(n-1)
    }
}
```

1.1.5 es6 部分

1、新增声明命令 let 和 const

在 es6 中通常用 let 和 const 来声明，let 表示变量、const 表示常量。

特点：

```
var str = `abc
def
gh`;
console.log(str);
let name = "小明";
function a() {
    return "ming";
}
console.log(`我的名字叫做${name}，年龄${17+2}岁，性别${'男'}，游戏ID: ${a()}`);
```

- let 和 const 都是块级作用域。以{}代码块作为作用域范围 只能在代码块里面使用。
- 不存在变量提升，只能先声明再使用，否则会报错。在代码块内，在声明变量之前，该变量都是不可用的。这在语法上，称为“暂时性死区”（temporal dead zone，简称 TDZ）。
- 在同一个代码块内，不允许重复声明。
- const 声明的是一个只读常量，在声明时就需要赋值。（如果 const 的是一个对象，对象所包含的值是可以被修改的。抽象一点儿说，就是对象所指向的地址不能改变，而变量成员是可以修改的。）

2、模板字符串（Template String）

用一对反引号(`)标识，它可以当作普通字符串使用，也可以用来定义多行字符串，也可以在字符串中嵌入变量，js 表达式或函数，变量、js 表达式或函数需要写在${ }中。

3、函数的扩展

- 函数的默认参数

  ES6 为参数提供了默认值。在定义函数时便初始化了这个参数，以便在参数没有被传递进去时使用。

```
function A(a,b=1){
    console.log(a+b);
}
A(1);    //2
A(2+3);  //5
```

- **箭头函数**

  在 es6 中，提供了一种简洁的函数写法，我们称作“箭头函数”。

  写法：函数名=(形参)=>{……}　当函数体中只有一个表达式时，{}和 return 可以省略，当函数体中形参只有一个时，()可以省略。

  特点：箭头函数中的 this 始终指向箭头函数定义时的离 this 最近的一个函数，如果没有最近的函数就指向 window。

```
//省略写法
var people = name => 'hello' + name;

var getFullName = (firstName, lastName) => {
    var fullName = firstName + lastName;
    return fullName;
}
```

4、对象的扩展

- 属性的简写。ES6 允许在对象之中，直接写变量。这时，属性名为变量名，属性值为变量的值。

```
var foo = 'bar';
var baz = {foo};  //等同于  var baz = {foo: foo};
```

- 方法的简写。省略冒号与 function 关键字。

```
var o = {
  method() {
    return "Hello!";
  }
};

// 等同于
var o = {
  method: function() {
    return "Hello!";
  }
};
```

- Object.keys()方法，获取对象的所有属性名或方法名（不包括原形的内容），返回一个数组。

```
var obj={name: "john", age: "21", getName: function () { alert(this.name)}};
console.log(Object.keys(obj));    // ["name", "age", "getName"]
console.log(Object.keys(obj).length);    //3

console.log(Object.keys(["aa", "bb", "cc"]));    //["0", "1", "2"]
console.log(Object.keys("abcdef"));    //["0", "1", "2", "3", "4", "5"]
```

- Object.assign ()，assign 方法将多个原对象的属性和方法都合并到了目标对象上面。可以接收多个参数，第一个参数是目标对象，后面的都是源对象。

```
var target  = {}; //目标对象
var source1 = {name : 'ming', age: '19'}; //源对象1
var source2 = {sex : '女'}; //源对象2
var source3 = {sex : '男'}; //源对象3，和source2中的对象有同名属性sex
Object.assign(target,source1,source2,source3);

console.log(target);    //{name : 'ming', age: '19', sex: '男'}
```

5、for...of 循环

是遍历所有数据结构的统一的方法。for...of 循环可以使用的范围包括数组、Set 和 Map 结构、某些类似数组的对象（比如 arguments 对象、DOM NodeList 对象）、Generator 对象，以及字符串。

```
var arr=["小林","小吴","小佳"];
for(var v of arr){
    console.log(v);
}
//小林
//小吴
//小佳
```

6、import 和 export

ES6 标准中，JavaScript 原生支持模块(module)了。这种将 JS 代码分割成不同功能的小块进行模块化，将不同功能的代码分别写在不同文件中，各模块只需导出公共接口部分，然后通过模块的导入的方式可以在其他地方使用。

export 用于对外输出本模块（一个文件可以理解为一个模块）变量的接口。

import 用于在一个模块中加载另一个含有 export 接口的模块。

import 和 export 命令只能在模块的顶部，不能在代码块之中。

```
//导入部分
//全部导入
import Person from './example'

//将整个模块所有导出内容当做单一对象，用as起别名
import * as example from "./example.js"
console.log(example.name)
console.log(example.getName())

//导入部分
import { name } from './example'

//导出部分
// 导出默认
export default App

// 部分导出
export class User extend Component {};
```

7、Promise 对象

- Promise 是异步编程的一种解决方案，将异步操作以同步操作的流程表达出来，避免了层层嵌套的回调函数。
- 它有三种状态，分别是 pending-进行中、resolved-已完成、rejected-已失败。

- Promise 构造函数包含一个参数和一个带有 resolve（解析）和 reject（拒绝）两个参数的回调。在回调中执行一些操作（例如异步），如果一切都正常，则调用 resolve，否则调用 reject。对于已经实例化过的 promise 对象可以调用 promise.then() 方法，传递 resolve 和 reject 方法作为回调。then()方法接收两个参数：onResolve 和 onReject，分别代表当前 promise 对象在成功或失败时。

```
var promise = new Promise((resolve, reject) => {
    var success = true;
    if (success) {
        resolve('成功');
    } else {
        reject('失败');
    }
}).then(
    (data) => { console.log(data)},
    (data) => { console.log(data)}
)
```

promise 的执行过程：

```
setTimeout(function() {
    console.log(0);
}, 0);
var promise = new Promise((resolve, reject) => {
    console.log(1);
    setTimeout(function () {
        var success = true;
        if (success) {
            resolve('成功');
        } else {
            reject('失败');
        }
    },2000);
}).then(
    (data) => { console.log(data)},
    (data) => { console.log(data)}
);
console.log(promise);   //<pending> 进行中
setTimeout(function () {
    console.log(promise);       //<resolved> 已完成
},2500);
console.log(2);

//1
//Promise {<pending>}
//2
//0
//成功
//Promise {<resolved>: undefined}
```

8、解构赋值

ES6 允许按照一定模式，从数组和对象中提取值，对变量进行赋值，这被称为解构（Destructuring）。

- 数组的解构赋值

数组中的值会自动被解析到对应接收该值的变量中，数组的解构赋值要一一对应如果有对应不上的就是 undefined

```
var [name, pwd, sex]=["小周", "123456", "男"];
console.log(name) //小周
console.log(pwd)//123456
console.log(sex)//男
```

- 对象的解构赋值

对象的解构与数组有一个重要的不同。数组的元素是按次序排列的，变量的取值由它的位置决定；而对象的属性没有次序，变量必须与属性同名，才能取到正确的值。

```
var obj={name:"小周", pwd:"123456", sex:"男"}
var {name, pwd, sex}=obj;
console.log(name) //小周
console.log(pwd)//123456
console.log(sex)//男

//如果想要变量名和属性名不同，要写成这样
let { foo: foz, bar: baz } = { foo: "aaa", bar: "bbb" };
console.log(foz) // "aaa"
console.log(foo) // error: foo is not defined
```

9、set 数据结构

Set 数据结构，类似数组。所有的数据都是唯一的，没有重复的值。它本身是一个构造函数。

属性和方法：

- size 数据的长度
- add() 添加某个值，返回 Set 结构本身。
- delete() 删除某个值，返回一个布尔值，表示删除是否成功。
- has() 查找某条数据，返回一个布尔值。
- clear() 清除所有成员，没有返回值。

应用：数组去重

```
var arr = [1,1,2,2,3];
var s = new Set(arr);
console.log(s); //{1, 2, 3}

console.log(s.size);    //3
console.log(s.add(4));  //{1, 2, 3, 4}
console.log(s.delete(4));       //true
console.log(s.has(4));  //false
s.clear();
```

10、Spread Operator 展开运算符(...)

- 将字符串转成数组

```
var str="abcd";
console.log([...str]) // ["a", "b", "c", "d"]
```

- 将集合转成数组

```
var sets=new Set([1,2,3,4,5])
console.log([...sets]) // [1, 2, 3, 4, 5]
```

- 两个数组的合并

```
var a1=[1,2,3];
var a2=[4,5,6];
console.log([...a1,...a2]); //[1, 2, 3, 4, 5, 6]
```

- 在函数中，用来代替 arguments 参数

  rest 参数 ...变量名称

  rest 参数是一个数组 ，它的后面不能再有参数，不然会报错

```
function func(...args){
    console.log(args);//[1, 2, 3, 4]
}
func(1, 2, 3, 4);

function f(x, ...y) {
    console.log(x);
    console.log(y);
}
f('a', 'b', 'c');     //a 和 ["b","c"]
f('a')                //a 和 []
f()                   //undefined 和 []
```

1.1.6 程序题（JS 程序）

1、写出代码对下列数组去重并从大到小排列{5,2,3,6,8,6,5,4,7,1,9}

```
function fn(arr){
```

```
        for (var i = 0; i < arr.length-1; i++) {
            for (var j = 0; j < arr.length-1-i; j++) {
                if(arr[j]<arr[j+1]){
                    var temp = arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=temp;
                }

            }

        }
        for (i = 0; i < arr.length; i++) {
            var c=arr[i];
            for (var s = i+1; s < arr.length; s++) {
                if(arr[s]==c)
                    arr.splice(s,1);
                    s--;
                }

            }
        }

        return arr;
    }
    console.log(fn([5,2,3,6,8,6,5,4,7,1,9]).toString());
```

2、用 JavaScript 实现冒泡排序。数据为 23、45、18、37、92、13、24

```
    //升序算法
    function sort(arr){
        for (var i = 0; i <arr.length; i++) {
            for (var j = 0; j <arr.length-i; j++) {
                if(arr[j]>arr[j+1]){
                    var c=arr[j];//交换两个变量的位置
                    arr[j]=arr[j+1];
                    arr[j+1]=c;
                }
            };
        };
        return arr.toString();
```

```
    }
    console.log(sort([23,45,18,37,92,13,24]));
```

3、用 js 实现随机选取 10–100 之间的 10 个数字，存入一个数组，并排序。

```
        function randomNub(aArray, len, min, max) {
                if (len >= (max - min)) {
                    return '超过' + min + '-' + max + '之间的个数范围' + (max - min - 1) + '个
的总数';
                }
                if (aArray.length >= len) {
                    aArray.sort(function(a, b) {
                        return a - b
                    });
                    return aArray;
                }
                var nowNub = parseInt(Math.random() * (max - min - 1)) + (min + 1);
                for (var j = 0; j < aArray.length; j++) {
                    if (nowNub == aArray[j]) {
                        randomNub(aArray, len, min, max);
                        return;
                    }
                }
                aArray.push(nowNub);
                randomNub(aArray, len, min, max);
                return aArray;
            }
    var arr=[];
    randomNub(arr,10,10,100);
```

4、已知数组

var stringArray = [“This”，“is”，“Baidu”，“Campus”]，Alert 出”This is Baidu Campus”。

```
    答案：var stringArray = ["This", "is", "Baidu", "Campus"]
```

```
    alert(stringArray.join(""))
```

5、已知有字符串 foo=”get-element-by-id”,写一个 function 将其转化成驼峰表示法”getElementById”。

```
    function combo(msg){
  var arr=msg.split("-");
  for(var i=1;i<arr.length;i++){
    arr[i]=arr[i].charAt(0).toUpperCase()+arr[i].substr(1,arr[i].length-1);
  }
  msg=arr.join("");
  return msg;
}
```

6、下面这段 JS 输出什么，并简述为什么?

```
function Foo() {

var i = 0;

return function () {

console.log(i++);

}

}

var f1 = Foo(),

f2 = Foo();

f1();

f1();

f2();
```

```
console.log(i);
0    //f1=Foo() 相当于 f1 赋值为函数 Foo()的返回值 f1=function（）{
console.log(i++)
}
1    //因为 f1=了一个 function 所以有了作用域，f2 和 f1 不同，不在一个内存中
0
报错   //i 为 Foo 内部的变量全局不可访问，全局中没有 i 变量所以会报错
```

7、有这样一个 URL：http://item.taobao.com/item.htm?a=1&b=2&c=&d=xxx&e，请写一段 JS 程序提取 URL 中的各个 GET 参数(参数名和参数个数不确定)，将其按 key-value 形式返回到一个 json 结构中，如{a:’1’, b:’2’, c:”, d:’xxx’, e:undefined}。

```
function serilizeUrl(url) {
    var urlObject = {};
    if (/\?/.test(url)) {
        var urlString = url.substring(url.indexOf("?") + 1);
        var urlArray = urlString.split("&");
        for (var i = 0, len = urlArray.length; i < len; i++) {
            var urlItem = urlArray[i];
            var item = urlItem.split("=");
            urlObject[item[0]] = item[1];
        }
        return urlObject;
    }
    return null;
}
```

8、var numberArray = [3,6,2,4,1,5]; （考察基础 API）

1) 实现对该数组的倒排，输出[5,1,4,2,6,3]

```
var numberArray = [3,6,2,4,1,5];
alert( numberArray .reverse())
```

2) 实现对该数组的降序排列，输出[6,5,4,3,2,1]

```
var numberArray = [3,6,2,4,1,5];
//  a-b 输出从小到大排序，b-a 输出从大到小排序。
 numberArray .sort(function(a,b){
      return b-a});
 console.log((numberArray .join()));
```

9、输出今天的日期，以 YYYY-MM-DD 的方式，比如今天是 2014 年 9 月 26 日，则输出 2014-09-26

```
var d = new Date();
// 获取年，getFullYear()返回 4 位的数字
var year = d.getFullYear();
// 获取月，月份比较特殊，0 是 1 月，11 是 12 月
var month = d.getMonth() + 1;
// 变成两位
month = month < 10 ? '0' + month : month;
// 获取日
var day = d.getDate();
day = day < 10 ? '0' + day : day;
alert(year + '-' + month + '-' + day);
```

10、将字符串”<tr><td>{$id}</td><td>{$name}</td></tr>”中的 {$id}替换成 10，{$nny （使用正则表达式）

```
答 案 ： ” <tr><td>{$id}</td><td>{$id}_{$name}</td></tr> ” .replace(/{\$id}/g, ’
```

```
10').replace(/{\$name}/g,  'Tony' );
```

11、请写出下面输出的值

Console.log(undefined || 1)；//值__**1**__

Console.log(null || NaN)；//值__**NaN**__

Console.log(0 && 1)；//值_**0**__

Console.log(0 && 1 || 0)；//值__**0**__

12、如何垂直居中一个浮动元素?

```
// 方法一：已知元素的高宽
#div1{
    background-color:#6699FF;
    width:200px;
    height:200px;
    position: absolute;        //父元素需要相对定位
    top: 50%;
    left: 50%;
    margin-top:-100px ;   //二分之一的 height，width
    margin-left: -100px;
    }

//方法二:未知元素的高宽

  #div1{
    width: 200px;
    height: 200px;
    background-color: #6699FF;

    margin:auto;
    position: absolute;        //父元素需要相对定位
    left: 0;
    top: 0;
    right: 0;
    bottom: 0;
```

```
        }
```

13、如何垂直居中一个 img?

```
#container     //<img>的容器设置如下
{
    display:table-cell;
    text-align:center;
    vertical-align:middle;
}
```

14、以下 js 的运行结果是什么，为什么?

```
var txt='hx';
    function hello(){
        var txt;
        fn();//world 函数名与变量名重复的时候，以函数名为主
        var fn=function(){alert('hello')}
        function fn(){alert('world');}
        alert(txt);//undefined 局部变量，只是声明，没有赋值
        fn();//hello 先进行声明，后赋值，执行 fn=function(){alert('hello')}
    }
    hello();
```

15、看下列代码，将会输出什么?(变量声明提升)

```
    var foo = 1;
    function(){
        console.log(foo);
        var foo = 2;
        console.log(foo);
    }
答案：输出 undefined 和 2。上面代码相当于：
    var foo = 1;
    function(){
        var foo;
        console.log(foo); //undefined
        foo = 2;
```

```
        console.log(foo); // 2;
    }
```

函数声明与变量声明会被 JavaScript 引擎隐式地提升到当前作用域的顶部，但是只提升名称不会提升赋值部分。

16、把两个数组合并，并删除第二个元素。

```
var array1 = ['a','b','c'];
var bArray = ['d','e','f'];
var cArray = array1.concat(bArray);
cArray.splice(1,1);
```

17、写一个 function，清除字符串前后的空格。（兼容所有浏览器）

```
//使用自带接口 trim()，考虑兼容性:
    if (!String.prototype.trim) {
     String.prototype.trim = function() {
     return this.replace(/^\s+/, "").replace(/\s+$/,"");
     }
    }
     // test the function
    var str = " \t\n test string ".trim();
    alert(str == "test string"); // alerts "true"
```

18、Javascript 中，以下哪条语句一定会产生运行错误？答案(B)

A、 var _变量=NaN;B、var 0bj = [];C、var obj = //;D、var obj = {};

19、以下两个变量 a 和 b，a+b 的哪个结果是 NaN? 答案(C)

A、var a=undefind; b=NaN

B、var a= ‘123’ ; b=NaN

C、var a =undefined , b =NaN

D、var a=NaN , b='undefined'

20、var a=10; b=20; c=4;　++b+c+a++ 以下哪个结果是正确的答案

(　B　)

A、　34　　B、35　C、36　D、37

21、写出程序运行的结果?

for(i=0, j=0; i<10, j<6; i++, j++){

k = i + j;}

结果：10

22、阅读以下代码，请分析出结果：

```
var arr = new Array(1 ,3 ,5);
arr[4]='z';
arr2 = arr.reverse();
arr3 = arr.concat(arr2);
alert(arr3);
```

弹出提示对话框：z,,5,3,1,z,,5,3,1

23、截取字符串 abcdefg 的 efg

alert('abcdefg'.substring(4));

24、判断一个字符串中出现次数最多的字符，统计这个次数

```
答：var str = 'asdfssaaasasasasaa';
var json = {};
for (var i = 0; i < str.length; i++) {
        if(!json[str.charAt(i)]){
                json[str.charAt(i)] = 1;
        }else{
```

```
                json[str.charAt(i)]++;
        }
};
var iMax = 0;
var iIndex = '';
for(var i in json){
        if(json[i]>iMax){
                iMax = json[i];
                iIndex = i;
        }
}
alert('出现次数最多的是:'+iIndex+'出现'+iMax+'次');
```

25、将数字 12345678 转化成 RMB 形式 如： 12,345,678

```
//个人方法;
//思路：先将数字转为字符， str= str + '' ;
//利用反转函数，每三位字符加一个 ','最后一位不加； re()是自定义的反转函数，最后再反转回去!
for(var i = 1; i <= re(str).length; i++){
   tmp += re(str)[i - 1];
   if(i % 3 == 0 && i != re(str).length){
      tmp += ',';
   }
}
```

26、加减运算

```
alert('5'+3); //53 string

alert('5'+'3'); //53 string

alert('5'-3); //2 number

alert('5'-'3'); //2 number
```

27、计算字符串字节数：

```
new function(s){
    if(!arguments.length||!s) return null;
    if(""==s) return 0;
    var l=0;
    for(var i=0;i<s.length;i++){
        if(s.charCodeAt(i)>255) l+=2; else l+=1;  //charCodeAt()得到的是 unCode 码
    }    //汉字的 unCode 码大于 255bit 就是两个字节
    alert(l);
}("hello world!");
```

28、看下列代码,输出什么?

```
var a = new Object();
a.value = 1;
b = a;
b.value = 2;
alert(a.value);
```

执行完后输出结果为 2

1.2 前端性能优化

1、你如何对网站的文件和资源进行优化?

文件合并：（目的是减少 http 请求）：Web 性能优化最佳实践中最重要的一条是减少 HTTP 请求，它也是 YSlow 中比重最大的一条规则。减少 HTTP 请求的方案主要有合并 JavaScript 和 CSS 文件、CSS Sprites、图像映射 （Image Map）和使用 Data URI 来编码图片。CSS Sprites 和图像映射现在已经随处可见了，但由于 IE6 和 IE7 不支持 Data URI 以及性能问题，这项技术尚未大量使用。目前大部分网页中的 JavaScript 和 CSS 文件数量和开发时一致，少量的网页会根据实际情况采取本地合并，这些合并中相当多的是有

选择的手动完成，每次新的合并都需要重新在本地完成并上传到服务器，比较的随意和繁琐，同样文件的压缩也有类似的情况。而利用服务端的合并和压缩，我们就可以按照开发的逻辑尽可能让文件的颗粒度变小，利用网页中 URL 的规则来自动实现文件的合并和压缩，这会相当的灵活和高效。参照 http://www.iamued.com/qianduan/1462.html

文件最小化/文件压缩：目的是直接减少文件下载的体积；常用的工具是 YUI Compressor。参考 http://www.cnblogs.com/Darren_code/archive/2011/12/31/property.html

使用 CDN 托管："其基本思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输的更快、更稳定。通过在网络各处放置节点服务器所构成的在现有的互联网基础之上的一层智能虚拟网络，CDN 系统能够实时的根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上。"　形象点说：古代打仗大家一定都知道，由于古代的交通很不发达，所以当外族进攻的时候往往不能及时的反击，等朝廷征完兵再把兵派往边境的时候那些侵略者却是早已不见了踪影，这个让古代的帝王很是郁闷。后来帝王们学聪明了，都将大量的兵员提前派往边境驻扎，让他们平时屯田，战时当兵，这样的策略起到了很显著的作用。

缓存的使用：（并且多个域名来提供缓存）

2、一个页面上有大量的图片（大型电商网站），加载很慢，你有哪些方法优化这些图片的加载，给用户更好的体验。

图片懒加载，在页面上的未可视区域可以添加一个滚动条事件，判断图片位置与浏览器顶端的距离与页面的距离，如果前者小于后者，优先加载。

如果为幻灯片、相册等，可以使用图片预加载技术，将当前展示图片的前一张和后一张优先下载。

如果图片为 css 图片，可以使用 CSSsprite，SVGsprite，Iconfont、Base64 等技术。

如果图片过大，可以使用特殊编码的图片，加载时会先加载一张压缩的特别厉害的缩略图，以提高用户体验。

如果图片展示区域小于图片的真实大小，则因在服务器端根据业务需要先行进行图片压缩，图片压缩后大小与展示一致。

3、图项目中图片处理相关的优化，项目中用到的优化方案，图片大小达到多少的时候选择处理?

答：1、首先了解在 web 开发中常见的图片有那些格式。JPG　通常使用的背景图片，照片图片，商品图片等等。这一类型的图片都属于大尺寸图片或较大尺寸图片一般使用的是这种格式。PNG　这种格式的又分为两种　一种 PNG-8，一种 PNG-24。PNG-8 格式不支持半透明，也是 IE6 兼容的图片存储方式。PNG-24 图片质量要求较高的半透明或全透明背景，保存成 PNG-24 更合适（为了兼容 IE 可以试用 js 插件 pngfix）一般是背景图标中使用的多。GIF 这种格式显而易见的是在需要 gif 动画的时候使用了。2．优化方案

样式代替图片例如：半透明、圆角、阴影、高光、渐变等。这些效果主流的浏览器都能够完美支持，而对于那些低端浏览器，我们并不会完全抛弃他们，“渐进增强”则是一个很好的解决方案。　精灵图 CSS Sprites，将同类型的图标或按钮等背景图合到一张大图中，减少页面请求。　字体图标 Icon Font，将图标做成字体文件。优点是图标支持多个尺寸，兼容所有浏览器，减少页面请求等。美中不足的是只支持纯色的 icon。SVG，对于绝大多数图案、图标等，矢量图更小，且可缩放而无需生成多套图。现在主流浏览器都支持 SVG 了，所以可放心使用！　Base64 将图片转化为 base64 编码格式，资源内嵌于 CSS 或 HTML 中，不必单独请求。Base64 格式 data：[][；charset=][；base64]，Base64 在 CSS 中的使用.demoImg{ background-image：　url("data：image/jpg；base64，/9j/4QMZRXhpZgAASUkqAAgAAAA L...."）；　}Base64 在 HTML 中的使用

　图片响应式通常图片加都是可以通过 lazy 加载的形式来的，那么可以在加载的时候来判断屏幕的尺寸来达到加载大图还是小图的目的来达到优化。像资源的优化方法

1.3 应用插件介绍

1、懒加载插件 Lazyload

介绍使用地址：https://vvo.github.io/lazyload/

也可以扫描二维码快速查看

2、响应式轮播插件

介绍使用地址：http://demo.lanrenzhijia.com/2013/pic1201/

3、PC 端特效插件合集

介绍使用地址：http://www.superslide2.com/

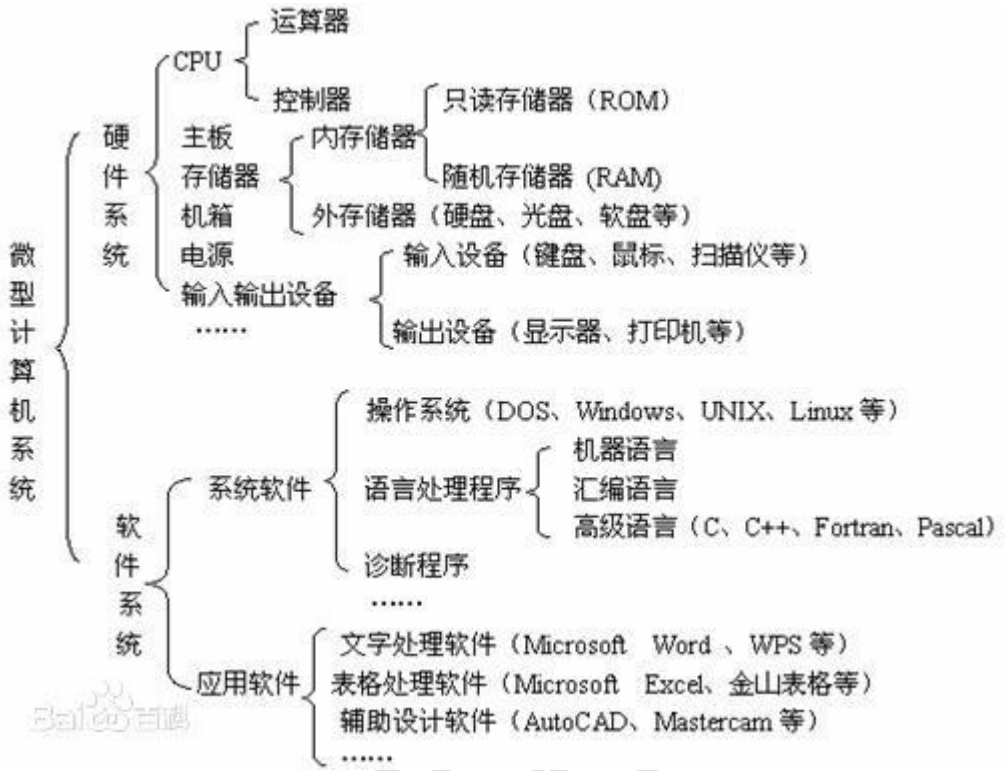
4、移动端特效插件合集

介绍使用地址 http://www.superslide2.com/

1.4 计算机相关术语

1、关于计算机相关术语的介绍

了解计算机相关术语的目的：作为一个计算机相关专业的学生来说，大学开设的课程有： 计算机基础、网页设计、计算机组成原理、数据结构、C 语言、C++、java、.net、计算机网络、高等数学、线性代数、离散数学、概率论、操作系统、软件测试、linux、汇编语言等。了解一定的计算机相关术语有助于更好的体现我们的专业性、技术性。



计算机组成图

2、http 超文本传输协议

超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。所有的 WWW 文件（即超文本文件（Hypertext），是指具有超链接功能的文件，它可以将文件中已经定义好的关键字（Keyword），经过鼠标的点取（Click），便可以得到该关键字的相关解释，这种方法使用户使用起来更感舒适。类似于早期使用的 WIN32 下的 HELP 文件。）都必须遵守这个标准。设计 HTTP 最初的目的是为了提供一种发布和接收 HTML 页面的方法。

工作原理：

一次 HTTP 操作称为一个事务，其工作过程可分为四步：

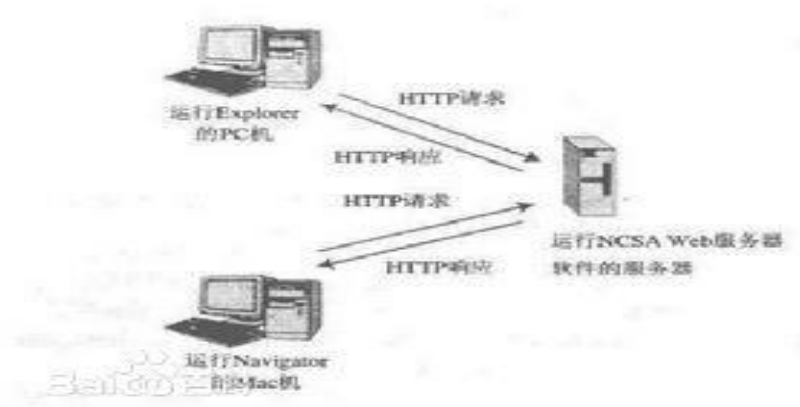
1、首先客户机与服务器需要建立连接。只要单击某个超级链接，HTTP 的工作就开始了。

2、建立连接后，客户机发送一个请求给服务器，请求方式的格式为：统一资源标识符（URL）、协议版本号，后边是 MIME 信息包括请求修饰符、客户机信息和可能的内容。

3、服务器接到请求后，给予相应的响应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后边是 MIME 信息包括服务器信息、实体信息和可能的内容。

4、客户端接收服务器所返回的信息通过浏览器显示在用户的显示屏上，然后客户机与服务器断开连接。

注意：如果在以上过程中的某一步出现错误，那么产生错误的信息将返回到客户端，由显示屏输出。对于用户来说，这些过程是由 HTTP 自己完成的，用户只要用鼠标点击，等待信息显示就可以了。



http 工作流程图

报文格式：

请求报文格式如下：

请求行 - 通用信息头 - 请求头 - 实体头 - 报文主体

应答报文格式如下：

状态行 - 通用信息头 - 响应头 - 实体头 - 报文主体

协议功能：

HTTP 协议（HyperText Transfer Protocol，超文本传输协议）是用于从 WWW 服务器传输超文本到本地浏览器的传输协议。它可以使浏览器更加高效，使网络传输减少。它不仅保证计算机正确快速地传输超文本文档，还确定传输文档中的哪一部分，以及哪部分内容首先显示(如文本先于图形)等。

HTTP 是客户端浏览器或其他程序与 Web 服务器之间的应用层通信协议。在 Internet 上的 Web 服务器上存放的都是超文本信息，客户机需要通过 HTTP 协议传输所要访问的超文本信息。HTTP 包含命令和传输信息，不仅可用于 Web 访问，也可以用于其他因特网/内联网应用系统之间的通信，从而实现各类应用资源超媒体访问的集成。

我们在浏览器的地址栏里输入的网站地址叫做 URL (Uniform Resource Locator，统一资源定位符)。就像每家每户都有一个门牌地址一样，每个网页也都有一个 Internet 地址。当你在

浏览器的地址框中输入一个 URL 或是单击一个超级链接时，URL 就确定了要浏览的地址。浏览器通过超文本传输协议(HTTP)，将 Web 服务器上站点的网页代码提取出来，并翻译成漂亮的网页。

3、TCP 协议

TCP 协议是网络协议中最基本的协议之一，TCP（Transmission Control Protocol 传输控制协议）是一种面向连接的、可靠的、基于字节流的传输层通信协议。

连接建立:

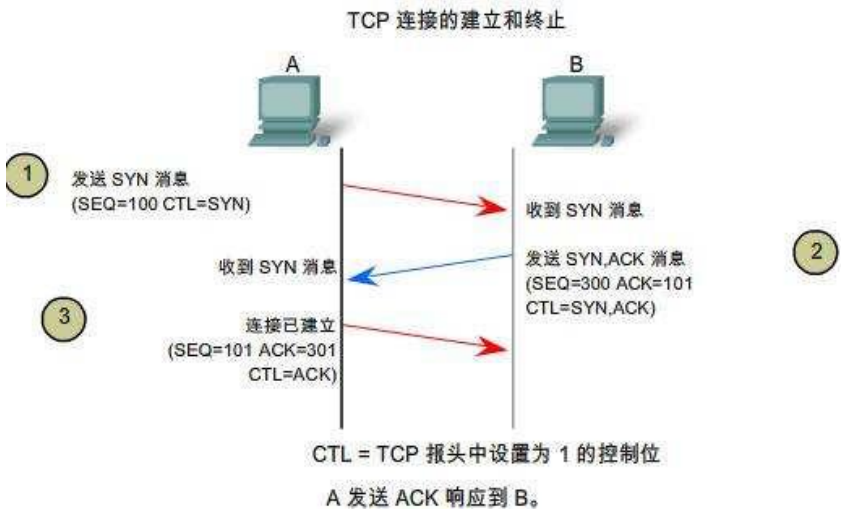
TCP 是因特网中的传输层协议，使用三次握手协议建立连接，完成三次握手，客户端与服务器开始传送数据。

TCP 三次握手:

第一次握手：建立连接时，客户端发送 syn 包（syn=j）到服务器，并进入 SYN_SENT 状态，等待服务器确认；SYN：同步序列编号（***Synchronize Sequence Numbers***）。

第二次握手：服务器收到 syn 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（syn=k），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态；

第三次握手：客户端收到服务器的 SYN+ACK 包，向服务器发送确认包 ACK(ack=k+1），此包发送完毕，客户端和服务器进入 ESTABLISHED（TCP 连接成功）状态，完成三次握手。



TCP 连接建立图

TCP 协议的优缺点：

优点：TCP 发送的包有序号，对方收到包后要给一个反馈，如果超过一定时间还没收到反馈就自动执行超时重发，因此 TCP 最大的优点是可靠。

缺点：很简单，就是麻烦，如果数据量比较小的话建立连接的过程反而占了大头，不断地重发也会造成网络延迟，因此比如视频聊天通常就使用 UDP，因为丢失一些包也没关系，速度流畅才是重要的。

4、计算机网络的分层体系结构

物理层：物理接口规范,传输比特流,网卡是工作在物理层的.

数据链路层：成帧,保证帧的无误传输,MAC 地址,形成 EHTHERNET 帧 数据链路层在不可靠的物理介质上提供可靠的传输。该层的作用包括：物理地址寻址、数据的成帧、流量控制、数据的检错、重发等。

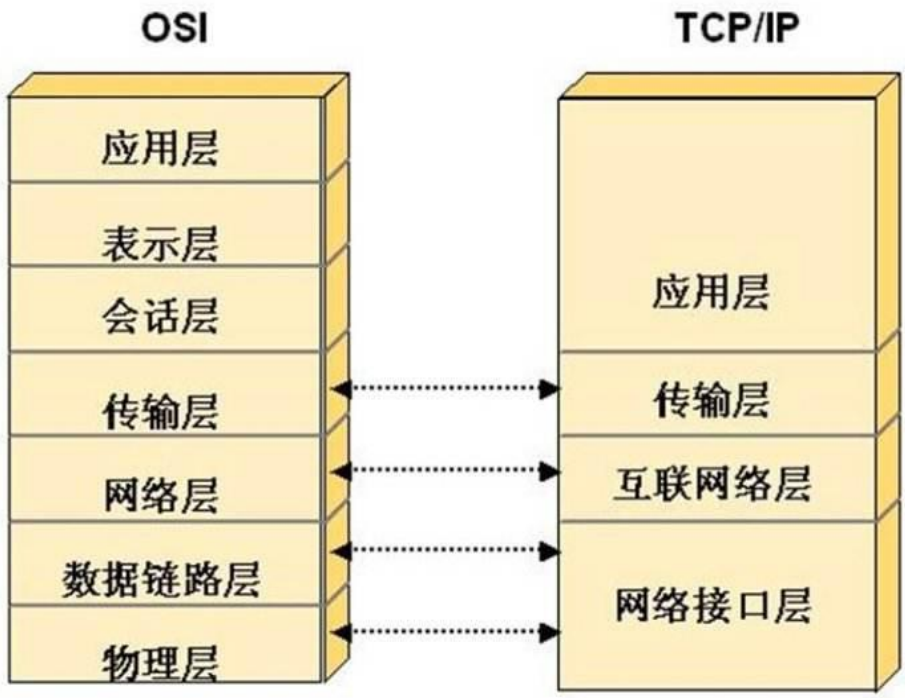
网络层：路由选择,流量控制,IP 地址,形成 IP 包

传输层：端口地址,如 HTTP 对应 80 端口.TCP 和 UDP 工作于该层,还有差错校验和流量控制.

会话层:组织两个会话进程之间的通信,并管理数据的交换使用 ETBIOS 和 WINSOCK 协议.QQ 等软件进行通讯因该是工作在会话层的.

表示层：使得不同操作系统之间通信成为可能.

应用层：对应于各个应用软件



计算机网络的分层体系结构图

5、计算机存储器相关知识

存储器：是计算机的重要组成部分. 它可分为：

计算机内部的存储器（简称内存）

计算机外部的存储器（简称外存）

内存储器从功能上可以分为：读写存储器 RAM、只读存储器 ROM 两大类

RAM 和 ROM 的区别：

RAM：（Ramdom Access Memory）易挥发性随机存取存储器，高速存取，读写时间相等，且与地址无关，如计算机内存等。RAM 表示的是读写存储器，可以与任一存储单元

进行读或写操作，计算机关闭电源后其内的信息将不在保存，再次开机需要重新装入，通常用来存放操作系统，各种正在运行的软件、输入和输出数据、中间结果及与外存交换信息等，我们常说的内存主要是指 RAM。

ROM：（Read Only Memory）只读存储器。断电后信息不丢失，如计算机启动用的 BIOS 芯片。存取速度很低，（较 RAM 而言）且不能改写。由于不能改写信息，不能升级，现已很少使用。ROM 表示的是只读存储器，即：它只能读出信息，不能写入信息，计算机关闭电源后其内的信息仍旧保存，一般用它存储固定的系统软件和字库等。

6、浏览器

一、浏览器相关知识介绍：

浏览器是指可以显示网页服务器或者文件系统的 HTML 文件（标准通用标记语言的一个应用）内容，并让用户与这些文件交互的一种软件。

它用来显示在万维网或局域网等内的文字、图像及其他信息。这些文字或图像，可以是连接其他网址的超链接，用户可迅速及轻易地浏览各种信息。大部分网页为 HTML 格式。

国内网民计算机上常见的网页浏览器有，QQ 浏览器、Internet Explorer、Firefox、Safari，Opera、Google Chrome、百度浏览器、搜狗浏览器、猎豹浏览器、360 浏览器、UC 浏览器、傲游浏览器、世界之窗浏览器等，浏览器是最经常使用到的客户端程序。

移动端产品有（移动端的浏览器）：百度、搜狗、UC、腾讯

内核：

IE 内核。包括 360 安全浏览器、IE、 Greenbrowser、 Maxthon2、世界之窗、刚开始的搜狗浏览器。

Chrome 内核，如 Chrome 浏览器。

双核(IE 和 chrome/webkit 内核)..双核的意思是一般网页用 chrome 内核(即 webkit 或高速模式)打开，网银等指定的网页用 IE 内核打开。 如 360 高速浏览器,搜狗高速浏览器，并不是 1 个网页同时用 2 个内核处理。

Firefox。

二、浏览器的主要构成（High Level Structure）

浏览器的主要组件包括：

1. 用户界面 - 包括地址栏、后退/前进按钮、书签目录等，也就是你所看到的除了用来显示你所请求页面的主窗口之外的其他部分。

2. 浏览器引擎 - 用来查询及操作渲染引擎的接口。

3. 渲染引擎 - 用来显示请求的内容，例如，如果请求内容为 html，它负责解析 html 及 css，并将解析后的结果显示出来。

4. 网络 - 用来完成网络调用，例如 http 请求，它具有平台无关的接口，可以在不同平台上工作。

5. UI 后端 - 用来绘制类似组合选择框及对话框等基本组件，具有不特定于某个平台的通用接口，底层使用操作系统的用户接口。

6. JS 解释器 - 用来解释执行 JS 代码。

7. 数据存储 - 属于持久层，浏览器需要在硬盘中保存类似 cookie 的各种数据，HTML5 定义了 web database 技术，这是一种轻量级完整的客户端存储技术

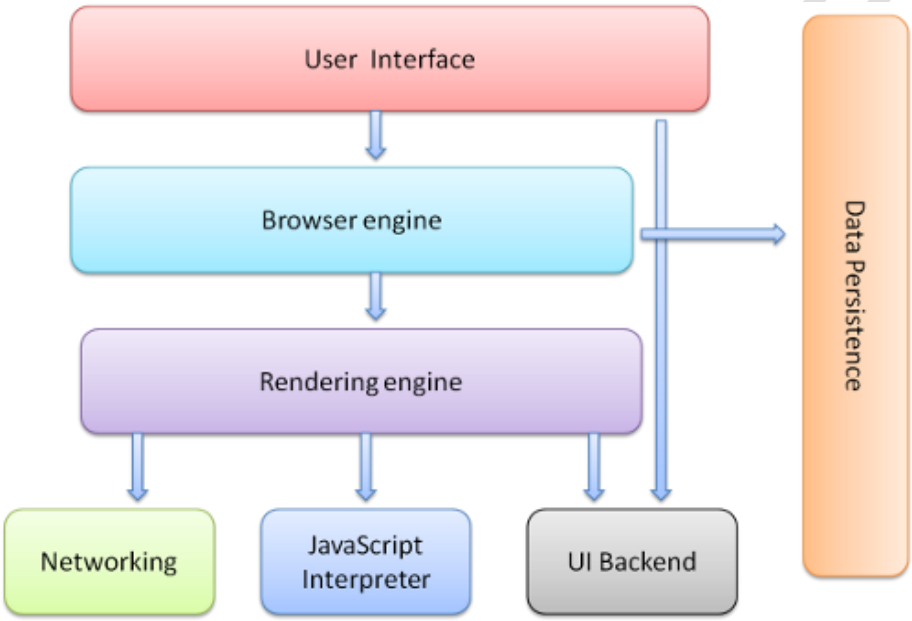


图 1：浏览器主要组件

需要注意的是，不同于大部分浏览器，Chrome 为每个 Tab 分配了各自的渲染引擎实例，每个 Tab 就是一个独立的进程。

对于构成浏览器的这些组件，后面会逐一详细讨论。

三、渲染引擎（The rendering engine）

渲染引擎的职责就是渲染，即在浏览器窗口中显示所请求的内容。

默认情况下，渲染引擎可以显示 html、xml 文档及图片，它也可以借助插件（一种浏览器扩展）显示其他类型数据，例如使用 PDF 阅读器插件，可以显示 PDF 格式，将由专门一章讲解插件及扩展，这里只讨论渲染引擎最主要的用途——显示应用了 CSS 之后的 html 及图片

详情参照：http://blog.csdn.net/finish_dream/article/details/52304276

7、服务器

一、介绍

服务器，也称伺服器，是提供计算服务的设备。由于服务器需要响应服务请求，并进行处理，因此一般来说服务器应具备承担服务并且保障服务的能力。

服务器的构成包括处理器、硬盘、内存、系统总线等，和通用的计算机架构类似，但是由于需要提供高可靠的服务，因此在处理能力、稳定性、可靠性、安全性、可扩展性、可管理性等方面要求较高。

在网络环境下，根据服务器提供的服务类型不同，分为文件服务器，数据库服务器，应用程序服务器，WEB 服务器等。

二、分类：

按照体系架构来区分，服务器主要分为两类：非 x86 服务器和 x86 服务器

按应用层次划分：**1、入门级服务器 2、工作组服务器 3、部门级服务器 4、企业级服务器 5、典型服务器应用（**办公 OA 服务器、ERP 服务器、WEB 服务器、数据库服务器、财务服务器、打印服务器、集群服务器、视频监控服务器、游戏服务器、论坛服务器）

三、特性：1、可扩展性 2、易使用性 3、可用性 4、易管理性

四、外形：1、机架式 2、刀片 3、塔式 4、机柜式

五、操作系统：服务器平台的操作系统。Unix 操作系统，由于是 Unix 的后代，大多都有较好的作服务器平台的功能。

8、线程与进程的区别

进程：

狭义定义：进程是正在运行的程序的实例。广义定义：进程是一个具有一定独立功能的程序关于某个数据集合的一次运行活动。它是操作系统动态执行的基本单元，在传统的操作系统中，进程既是基本的分配单元，也是基本的执行单元。进程是一个实体，进程是一个“执行中的程序”

进程的特点：并发性：任何进程都可以同其他进程一起并发执行

独立性：进程是一个能独立运行的基本单位，同时也是系统分配资源和调度的独立单位；

异步性：由于进程间的相互制约，使进程具有执行的间断性，即进程按各自独立的、不可预知的速度向前推进

结构特征：进程由程序、数据和进程控制块三部分组成。

多个不同的进程可以包含相同的程序：一个程序在不同的数据集里就构成不同的进程，能得到不同的结果；但是执行过程中，程序不能发生改变。

线程：

线程是进程中的实体，一个进程可以拥有多个线程，一个线程必须有一个父进程。线程不拥有系统资源，只有运行必须的一些数据结构；它与父进程的其它线程共享该进程所拥有的全部资源。线程可以创建和撤消线程，从而实现程序的并发执行。一般，线程具有就绪、阻塞和运行三种基本状态。

有时候，线程也称作轻量级进程。就象进程一样，线程在程序中是独立的、并发的执行路径，每个线程有它自己的堆栈、自己的程序计数器和自己的局部变量。但是，与分隔的进程相比，进程中的线程之间的隔离程度要小。它们共享内存、文件句柄和其它每个进程应有的状态。

线程的基本操作：

派生：线程在进程内派生出来，它即可由进程派生，也可由线程派生。

阻塞（Block）：如果一个线程在执行过程中需要等待某个事件发生，则被阻塞。

激活（unblock）：如果阻塞线程的事件发生，则该线程被激活并进入就绪队列。

调度（schedule）：选择一个就绪线程进入执行状态。

结束（Finish）：如果一个线程执行结束，它的寄存器上下文以及堆栈内容等将被释放。

二者的区别：

1）地址空间和其它资源（如打开文件）：进程间相互独立，同一进程的各线程间共享。某进程内的线程在其它进程不可见。

2）通信：进程间通信 IPC，线程间可以直接读写进程数据段（如全局变量）来进行通信——需要进程同步和互斥手段的辅助，以保证数据的一致性。

3）调度和切换：线程上下文切换比进程上下文切换要快得多。

4）在多线程 OS 中，进程不是一个可执行的实体。

补充了解：一个程序至少有一个进程,一个进程至少有一个线程. 线程的划分尺度小于进程，使得多线程程序的并发性高。另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。

线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。

从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。

进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动,进程是系统进行资源分配和调度的一个独立单位.线程是进程的一个实体,是 CPU 调度和分派的基本单位,它

是比进程更小的能独立运行的基本单位.线程自己基本上不拥有系统资源,只拥有一点在运行中必不可少的资源(如程序计数器,一组寄存器和栈),但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源.

一个线程可以创建和撤销另一个线程;同一个进程中的多个线程之间可以并发执行.进程和线程的主要差别在于它们是不同的操作系统资源管理方式。进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。但对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程。

9、经典编程算法

1、快速排序算法

2、堆排序算法

3、归并排序

4、二分查找算法

5、BFPRT（线性查找算法）

6、DFS（深度优先搜索）

7、BFS（广度优先搜索）

8、Floyd-Warshall all-pairs 最短路径算法

10、经典排序算法

1、插入排序—直接插入排序(Straight Insertion Sort)

2、插入排序—希尔排序（Shell`s Sort）

3、选择排序—简单选择排序（Simple Selection Sort）

4、选择排序—堆排序（Heap Sort）

5、交换排序—冒泡排序（Bubble Sort）

6、快速排序（Quick Sort）

7、归并排序（Merge Sort）

8、桶排序/基数排序(Radix Sort)

9、堆排序

10、计数排序

11、黑盒、白盒、灰盒测试

白盒测试：

白盒测试也称为结构测试或逻辑驱动测试，是针对被测单元内部是如何进行工作的测试。它根据程序的控制结构设计测试用例，主要用于软件或程序验证。白盒测试法检查程序内部逻辑结构，对所有的逻辑路径进行测试，是一种穷举路径的测试方法，但即使每条路径都测试过了，但仍然有可能存在错误。因为：穷举路径测试无法检查出程序本身是否

违反了设计规范，即程序是否是一个错误的程序；穷举路径测试不可能检查出程序因为遗漏路径而出错；穷举路径测试发现不了一些与数据相关的错误。

白盒测试需要遵循的原则有： 1. 保证一个模块中的所有独立路径至少被测试一次；2. 所有逻辑值均需要测试真（true）和假（false）；两种情况；3. 检查程序的内部数据结构，保证其结构的有效性；4. 在上下边界及可操作范围内运行所有循环。

白盒测试方法有：

静态测试、动态测试、单元测试、代码检查、同行评审、技术评审

黑盒测试：

黑盒测试又称为功能测试、数据驱动测试或基于规格说明书的测试，是一种从用户观点出发的测试。测试人员一般把被测程序当作一个黑盒子。

黑盒测试主要测到的错误类型有：不正确或遗漏的功能；接口、界面错误；性能错误；数据结构或外部数据访问错误；初始化或终止条件错误等等。

常用的黑盒测试方法有：等价类划分法；边界值分析法；因果图法；场景法；正交实验设计法；判定表驱动分析法；错误推测法；功能图分析法。

两者之间的区别：

黑盒测试着重测试软件功能。

黑盒测试并不能取代白盒测试，它是与白盒测试互补的测试方法，它很可能发现白盒测试不易发现的其他类型错误。

灰盒测试（Gray-Box Testing）

灰盒测试更像是白盒测试和黑盒测试的混合测试，现阶段对灰盒测试没有更明确的定义，但更多的时候，我们的测试做的就是灰盒测试，即既会做黑盒测试又会做白盒测试。

12、二叉排序树

定义：二叉排序树或者是一棵空树，或者是具有下列性质的二叉树：

（1）若左子树不空，则左子树上所有结点的值均小于或等于它的根结点的值；

（2）若右子树不空，则右子树上所有结点的值均大于或等于它的根结点的值；

（3）左、右子树也分别为二叉排序树；

详见：http://m.blog.csdn.net/yxb_yingu/article/details/51336197

1.5 git 使用

1、关于 git（简介）

Git 是 Linux 之父 Linus 的第二个伟大的作品，它最早是在 Linux 上开发的，被用来管理 Linux 核心的源代码。后来慢慢地有人将其移植到了 Unix、Windows、Max OS 等操作系统中。

Git 是一个分布式的版本控制系统，与集中式的版本控制系统不同的是，每个人都工作在通过克隆建立的本地版本库中。也就是说每个人都拥有一个完整的版本库，查看提交日志、提交、创建里程碑和分支、合并分支、回退等所有操作都直接在本地完成而不需要网络连接。

对于 Git 仓库来说，每个人都有一个独立完整的仓库，所谓的远程仓库或是服务器仓库其实也是一个仓库，只不过这台主机 24 小时运行，它是一个稳定的仓库，供他人克隆、推送，也从服务器仓库中拉取别人的提交。

Git 是目前世界上最先进的分布式版本控制系统，没有之一，对，没有之一!

2、git 常用命令总结

git init # 在本地新建一个仓库

git add # 将工作区的修改提交到暂存区

git commit # 将暂存区的修改提交到当前分支

git reset # 回退到某一个版本

git stash # 保存某次修改

git pull # 从远程更新代码

git push # 将本地代码更新到远程分支上

git reflog # 查看历史命令

git status # 查看当前仓库的状态

git diff # 查看修改

git log # 查看提交历史

git revert # 回退某个修改

3、git 命令详解请参考以下链接

https://www.cnblogs.com/my--sunshine/p/7093412.html

微信扫描二维码快速查看！

第二部分

2.1、流行框架前端热潮知识

2.1.1、前端框架与类库的区别。

11、总括前端框架与类库的区别

框架：(1)、框架是一种特殊的、已经实现了的 WEB 应用，你只需要对它填充具体的业务逻辑。这里框架是起主导作用的，由它来根据具体的应用逻辑来调用你的代码。

(2)封装好的代码，不过会功能更加多样，一个框架会包含多个类库，并且框架面向的顶层的开发，而类库更多的是面向底层的开发。

(3)、注重整体，框架帮助你解决“代码如何组织”的问题，面对中/大型项目，特别是需要多人共同参与的项目，选择一个合适的框架有助于写出规范的。

(4)、目前常见的前端框架：Bootstrap、angular.js、vue.js、react.js、node.js、amaze UI

类库：(1)、类库是一些函数的集合，通过调用开放出来的 API 获取相应的功能

它能帮助你写 WEB 应用。起主导作用的是你的代码，由你来决定何时使用类库。

封装好的代码方法

(2)、注重细节，类库帮助你解决“如何把代码写得更少/巧/强壮”的问题

对于小型应用来说，因为业务逻辑简单，代码总量不会太大，组织不会是太大得问题，所以用类库就够了。

(3)、常见的类库：jquery、zepto

12、jquery 和 javascript 的区别

jQuery 是 JavaScript 的一个封装集合。封装了很多 JavaScript 的方法，也就是说，

jQuery 里面的内容都是 JavaScript 语句。只是封装起来让我们学习和使用的简单一些.

JavaScript 的优点和缺点：

优点： 1、性能：由于 JavaScript 运行在客户端，节省了 web 服务器的请求时间和带宽 2、轻量级的脚本语言，比较容易学习 3、运行在用户机器上，运行结果和处理相对比较快。4、可以使用第三方附加组件来检查代码片段。

缺点： 1、安全问题：由于 JavaScript 在客户端运行，可能被用于黑客目的。2 渲染问题：在不同浏览器中的处理结果可能不同。

jQuery 的优点和缺点：

优点:1、使用 jQuery 最大的好处是少量的代码做更多的事情。与 JavaScript 相比，jQuery 的语法更加简单。通过 jQuery，可以很容易地浏览文档、选择元素、处理事件以及添加效果等，同时还允许开发者定制插件。 2、jQuery 消除了 JavaScript 跨平台兼容问题。相比其他 JavaScript 和 JavaScript 库，jQuery 更容易使用。jQuery 有一个庞大的库/函数。jQuery 有良好的文档和帮助手册。jQuery 支持 AJAX。

缺点： 由于不是原生 JavaScript 语言，理解起来可能会受到限制。项目中需要包含 jQuery 库文件。如果包含多个版本的 jQuery 库，可能会发生冲突。

2.1.2、VUE

1、VUE 的适用场景

对浏览器兼容要求不高，vuejs 是到 IE9；

对 MVVM 有一定的经验；

加载速度要求高；

对性能要求比较高；

需要组件化开发；

喜欢对原生 js 对象操作；

SPA

使用 vue 的项目有：苏宁易购触屏版的购物车结算页面、有桌面端（比如 饿了么安全应急响应中心）也有移动端（比如 饿了么招聘）

2、VUE 是什么

vue.js（读音 /vju:/，类似于 view） 是一套构建用户界面的渐进式框架

与其他重量级框架不同的是，Vue 采用自底向上增量开发的设计。Vue 的核心库只关注视图层，它不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与单文件组件和 Vue 生态系统支持的库结合使用时，Vue 也完全能够为复杂的单页应用程序提供驱动。

3、Vue 父子组件间传值

具体操作流程：http://www.cnblogs.com/daiwenru/p/6694530.html

总结：子组件中需要以某种方式例如点击事件的方法来触发一个自定义事件

将需要传的值作为其自定义事件的第二个参数，该值将作为实参传给响应自定义事件的方法

在父组件中注册子组件并在子组件标签上绑定对自定义事件的监听

4、React 和 Vue 的区别

React 和 Vue 的相似之处：

（1）、使用 Virtual DOM

（2）、提供了响应式（Reactive）和组件化（Composable）的视图组件。

（3）、将注意力集中保持在核心库，而将其他功能如路由和全局状态管理交给相关的库。

Vue 与 Angular、React 的对比：

1、vue.js 更轻量，gzip 后只有 20K+，angular:56K ,react:44K

2、vue.js 更易上手，学习曲线平稳

3、吸收两家之长，有 angular 的指令和 react 组件化思想

优缺点：

优点：体积小。接口灵活。侵入性好，可用于页面的一部分，而不是整个页面。扩展性好。源码规范简洁。代码较为活跃，作者是中国人，可在官方论坛中文提问。github9000+。基于组件化的开发。它是一个轻量级 mvvm 框架、数据驱动+组件化的前端开发、社区完善

缺点：社区不大，如果有问题可以读源码。功能仅限于 view 层，Ajax 等功能需要额外的库。对开发人员要求较高。开发的话，需要 webpack，不然很难用，最好配合 es6。

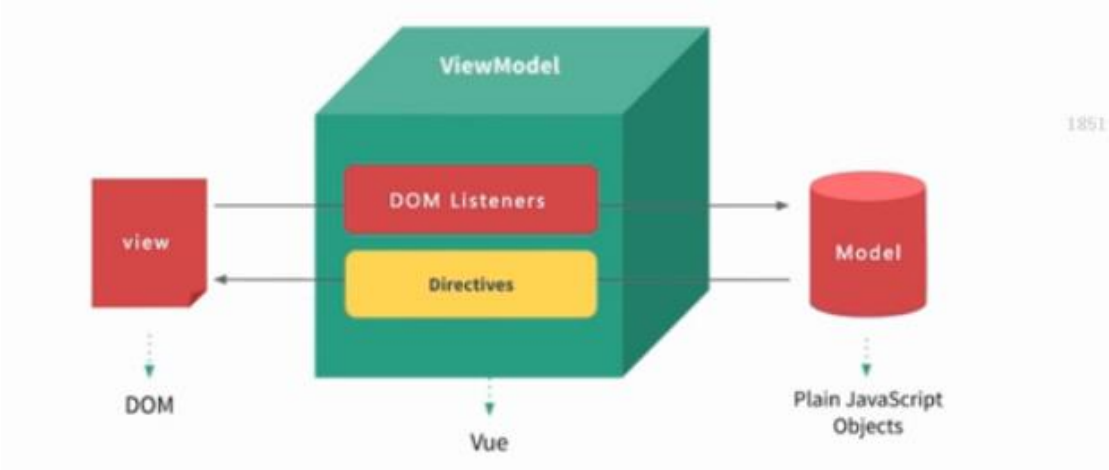
5、Vue.js 核心思想：

1、数据驱动（基于操作 dom 的方式）

（1）、Dom 是数据的一种自然映射（双向数据绑定）手动改变 DOM 非常麻烦使用 vue.js 之后，只需要改变数据，通过改变 Directives 指令,当数据发生变化，会通过数据指令去修改对应的 DOM

（2）、Vue.js 还会对事件进行一定的监听，当我们改变视图（view）的时候通过 DOM Listeners 来改变数据

通过以上两点就实现了数据的双向绑定



2、组件化

Vue-cli:(它是 vue 的脚手架工具)

作用：帮助我们完成基础的代码（包括：目录结构、本地调试、代码部署、热加载、单元测试）

组件的设计原则：

页面上每一个独立的可视/可交互区域视为一个组件

每个组件对应一个工程目录，组件所需要的各种资源在这个目录下就近维护

展示面不过是组件的容器，组件可以嵌套自由组合形成完整的页面

6、什么是 MVVM：

MVVM 是 Model-View-ViewModel 的缩写。MVVM 是一种设计思想。Model 层代表数据模型，也可以在 Model 中定义数据修改和操作的业务逻辑；View 代表 UI 组件，它负责将数据模型转化成 UI 展现出来，ViewModel 是一个同步 View 和 Model 的对象。

在 MVVM 架构下，View 和 Model 之间并没有直接的联系，而是通过 ViewModel 进行交互，Model 和 ViewModel 之间的交互是双向的， 因此 View 数据的变化会同步到 Model 中，而 Model 数据的变化也会立即反应到 View 上。

ViewModel 通过双向数据绑定把 View 层和 Model 层连接了起来，而 View 和 Model 之间的同步工作完全是自动的，无需人为干涉，因此开发者只需关注业务逻辑，不需要手动操作 DOM，不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。

7、MVVM 和 MVC 的区别：

mvc 和 mvvm 其实区别并不大。都是一种设计思想。主要就是 mvc 中 Controller 演变成 mvvm 中的 viewModel。mvvm 主要解决了 mvc 中大量的 DOM 操作使页面渲染性能降低，加载速度变慢，影响用户体验。

8、MVVM 和 jquery 的区别：

- vue 数据驱动，通过数据来显示视图层而不是节点操作。
- jQuery 是一个快速、简洁的 JavaScript 框架 ，jQuery 是基于事件驱动

9、Vue 的优点是什么：

- 低耦合。视图（View）可以独立于 Model 变化和修改，一个 ViewModel 可以绑定到不同的"View"上，当 View 变化的时候 Model 可以不变，当 Model 变化的时候 View 也可以不变。
- 可重用性。你可以把一些视图逻辑放在一个 ViewModel 里面，让很多 view 重用这段视图逻辑。
- 独立开发。开发人员可以专注于业务逻辑和数据的开发（ViewModel），设计人员可以专注于页面设计。
- 可测试。界面素来是比较难于测试的，而现在测试可以针对 ViewModel 来写。

10、Vue 组件之间的传值：

- 父组件往子组件传值 props
- 子组件往父组件传值，通过 emit 事件
- 不同组件之间传值，通过 eventBus（小项目少页面用 eventBus，大项目多页面使用 vuex）

11、Vue 路由之间跳转有哪些：

- 声明式（标签跳转）
- 编程式（ js 跳转）

12、vue-cli 怎样使用自定义组件？遇到过哪些问题：

第一步：在 components 目录新建你的组件文件（indexPage.vue），script 一定要 export default {}

第二步：在需要用的页面（组件）中导入：import indexPage from '@/components/indexPage.vue'

第三步：注入到 vue 的子组件的 components 属性上面,components:{indexPage}

第四步：在 template 视图 view 中使用，

例如有 indexPage 命名，使用的时候则 index-page

13、vue 如何实现按需加载配合 webpack 设置：

webpack 中提供了 require.ensure()来实现按需加载。以前引入路由是通过 import 这样的方式引入，改为 const 定义的方式进行引入。

不进行页面按需加载引入方式 import home from ../../common/home.vue'

进行页面按需加载的引入方式：const home = r => require.ensure([], () => r (require('../../common/home.vue')))

14、vuex 是什么？怎么使用？哪种功能场景使用它？

vuex 是 vue 框架中状态管理。

使用方式：在 main.js 引入 store，注入。新建一个目录 store，….. export 。

场景有：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

15、vuex 有哪几种属性？

有五种，分别是 State、 Getter、Mutation 、Action、 Module

- vuex 的 State 特性
    - A、Vuex 就是一个仓库，仓库里面放了很多对象。其中 state 就是数据源存放地，对应于一般 Vue 对象里面的 data
    - B、state 里面存放的数据是响应式的，Vue 组件从 store 中读取数据，若是 store 中的数据发生改变，依赖这个数据的组件也会发生更新

    - C、它通过 mapState 把全局的 state 和 getters 映射到当前组件的 computed 计算属性中
- vuex 的 Getter 特性
    - A、getters 可以对 State 进行计算操作，它就是 Store 的计算属性
    - B、 虽然在组件内也可以做计算属性，但是 getters 可以在多组件之间复用
    - C、 如果一个状态只在一个组件内使用，是可以不用 getters
- vuex 的 Mutation 特性
    - Action 类似于 mutation，不同在于：Action 提交的是 mutation，而不是直接变更状态；Action 可以包含任意异步操作。
- vuex 的 Action 特性
    - 包含任意异步操作，通过提交 mutation 间接更变状态
- vuex 的 Module 特性
    - 将 store 分割成模块，每个模块都具有 state、mutation、action、getter、甚至是嵌套子模块

16、不用 Vuex 会带来什么问题?

- 可维护性会下降，想修改数据要维护三个地方；
- 可读性会下降，因为一个组件里的数据，根本就看不出来是从哪来的；

- 增加耦合，大量的上传派发，会让耦合性大大增加，本来 Vue 用 Component 就是为了减少耦合，现在这么用，和组件化的初衷相背。

17、 v-show 和 v-if 指令的共同点和不同点?

- v-show 指令是通过修改元素的 display 的 CSS 属性让其显示或者隐藏
- v-if 指令是直接销毁和重建 DOM 达到让元素显示和隐藏的效果

18、 如何让 CSS 只在当前组件中起作用?

将当前组件的<style>修改为<style scoped>

19、 <keep-alive></keep-alive>的作用是什么?

<keep-alive></keep-alive> 包裹动态组件时，会缓存不活动的组件实例，主要用于保留组件状态或避免重新渲染。

20、Vue 中引入组件的步骤?

1) 采用 ES6 的 import ... from ...语法或 CommonJS 的 require()方法引入组件

2) 对组件进行注册

```
vue.component('My-component',{
template:"<div>my name is template!</div>"
```

});

3) 使用组件<my-component></my-component>

21、指令 v-el 的作用是什么?

提供一个在页面上已存在的 DOM 元素作为 Vue 实例的挂载目标.可以是 CSS 选择器，也可以是一个 HTMLElement 实例，vue2.0 当中改为 ref 属性

22、在 Vue 中使用插件的步骤?

1) 采用 ES6 的 import ... from ...语法或 CommonJSd 的 require()方法引入插件

2} 使用全局方法 Vue.use(plugin)使用插件,可以传入一个选项对象 Vue.use(MyPlugin, { someOption: true })

23、请列举出 3 个 Vue 中常用的生命周期钩子函数?（举两例）

- created: 实例已经创建完成之后调用,在这一步,实例已经完成数据观测，属性和方法的运算, watch/event 事件回调. 然而，挂载阶段还没有开始, $el 属性目前还不可见
- mounted: el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用该钩子。如果 root 实例挂载了一个文档内元素，当 mounted 被调用时 vm.$el 也在文档内。

24、active-class 是哪个组件的属性?

vue-router 模块的 router-link 组件。

25、怎么定义 vue-router 的动态路由以及如何获取传过来的动态参数?

- 在 router 目录下的 index.js 文件中，对 path 属性加上/:id。
- 使用 router 对象的 params.id。

26、vue-router 有哪几种导航钩子?

1) 全局导航钩子：router.beforeEach(to,from,next)

作用：跳转前进行判断拦。

2) 组件内的钩子；

3) 单独路由独享组件

27、vue 生命周期?

Vue 生命周期总共分为 8 个阶段：创建前/后，载入前/后，更新前/后，销毁前/后。

官网链接：

https://cn.vuejs.org/v2/guide/instance.html#生命周期图示

- 创建前/后： 在 beforeCreate 阶段，vue 实例的挂载元素 el 和数据对象 data 都为 undefined，还未初始化。在 created 阶段，vue 实例的数据对象 data 有了，el 还没有。
- 载入前/后：在 beforeMount 阶段，vue 实例的$el 和 data 都初始化了，但还是挂载之前为虚拟的 dom 节点，data.message 还未替换。在 mounted 阶段，vue 实例挂载完成，data.message 成功渲染。
- 更新前/后：当 data 变化时，会触发 beforeUpdate 和 updated 方法。
- 销毁前/后：在执行 destroy 方法后，对 data 的改变不会再触发周期函数，说明此时 vue 实例已经解除了事件监听以及和 dom 的绑定，但是 dom 结构依然存在

28、什么是 vue 生命周期?

Vue 实例从创建到销毁的过程，就是生命周期。也就是从开始创建、初始化数据、编译模板、挂载 Dom→渲染、更新→渲染、卸载等一系列过程，我们称这是 Vue 的生命周期。

29、vue 生命周期的作用是什么?

它的生命周期中有多个事件钩子，让我们在控制整个 Vue 实例的过程时更容易形成好的逻辑。

30、第一次页面加载会触发哪几个钩子?

第一次页面加载时会触发 beforeCreate, created, beforeMount, mounted 这几个钩子。

31、DOM 渲染在 哪个周期中就已经完成?

DOM 渲染在 mounted 中就已经完成了。

32、简单描述每个周期具体适合哪些场景?

- beforecreate：可以在这加个 loading 事件，在加载实例时触发
- created：初始化完成时的事件写在这里，如在这结束 loading 事件，异步请求也适宜在这里调用
- mounted：挂载元素，获取到 DOM 节点
- updated：如果对数据统一处理，在这里写上相应函数
- beforeDestroy：可以做一个确认停止事件的确认框

33、说出至少 4 种 vue 当中的指令和它的用法?

- v-if：判断是否隐藏；
- v-for：数据循环；
- v-bind:class：绑定一个属性；
- v-model：实现双向绑定

34、vue-loader 是什么？使用它的用途有哪些？

- vue-loader 是解析.vue 文件的一个加载器。
- 用途：js 可以写 es6、style 样式可以 scss 或 less、template 可以加 jade 等。

35、scss 是什么？

css 的预编译

36、scss 在 vue-cli 中的安装使用步骤是？

- 第一步：先装 css-loader、node-loader、sass-loader 等加载器模块
- 第二步：在 build 目录找到 webpack.base.config.js，在那个 extends 属性中加一个拓展.scss
- 第三步：在同一个文件，配置一个 module 属性
- 第四步：然后在组件的 style 标签加上 lang 属性 ，例如：lang=”scss”

37、scss 有哪几大特性？

- 可以用变量，例如（$变量名称=值）；
- 可以用混合器，例如（）
- 可以嵌套

38、v-for 为什么使用 key?

因为 vue 在更新渲染 dom 的时候是根据新旧 dom 数进行对比的，使用 key 来给每个节点做一个唯一标识，Diff 算法就可以正确的识别此节点，找到正确的位置区插入新的节点。

39、为什么避免 v-if 和 v-for 用在一起?

当 Vue 处理指令时，v-for 比 v-if 具有更高的优先级，通过 v-if 移动到容器元素，不会再重复遍历列表中的每个值。取而代之的是，我们只检查它一次，且不会在 v-if 为否的时候运算 v-for。

40、VNode 是什么?

Vue 在 页面上渲染的节点，及其子节点称为“虚拟节点 (Virtual Node)”，简写为“VNode”。

41、虚拟 DOM 是什么?

“虚拟 DOM”是由 Vue 组件树建立起来的整个 VNode 树的称呼。

42、vue 中利用索引修改数组的时候，页面会跟着同步吗?

利用索引修改数组的时候，页面不会进行同步，此时应该利用 vue.set 的方法进行设置数据

43、vue 首屏加载过慢如何解决?

- 路由懒加载，会将原来打包一个 app.js 的文件打包成多个文件
- 异步组件，按需加载
- 组件异步加载，将子组件
- webpack 开启 gzip 压缩
- 如果图片过多，开启图片懒加载
- 使用 cdn 资源
- 如果首页是登录页，做多入口

44、路由中如何去除 url 上的‘#’?

路由有两种模式，一种为 hash 模式，另一种为 history 模式，开启 history 模式后自动去除#，开启 history 模式需要后台配合

官网地址：https://router.vuejs.org/zh/guide/essentials/history-mode.html

45、vue 中的单项数据流和双向数据绑定是什意思?

- 单项数据流是指数据是单向的，父组件的数据传递给子组件，只能单项绑定，不可以在子组件修改父组件的数据
- 双向数据绑定：是指数据和页面进行双向绑定，相互影响

46、vue 中双向数据绑定的原理是什么?

vue 双向数据绑定的原理主要通过数据劫持 Object.defineProperty 和发布订阅模式实现的，通过 Object.defineProperty 监听数据发生变化然后通知订阅者（watcher），订阅者触发响应的回调

47、为什么 vue 组件中的 data 必须是函数?

因为如果默认为 data 是对象的话，对象为引用类型，这样的话，所有复用的组件都是引用的同一个数据，但是如果是函数的话，每次函数都会先创建一个新的数据，从而使每个组件的数据独立

48、你知道 webpack 中 babel、plugin、loader 都有什么作用吗?

- babel 用来出来 es6 转 es5
- plugin 配置 webpack 的一些插件
- loader 用来配置解析处理第三方文件的

49、keep-alive 的作用?

<keep-alive></keep-alive> 包裹动态组件时，会缓存不活动的组件实例，主要用于保留组件状态或避免重新渲染。

50、$route 和 router 的区别?

- $route：包括 path，params，hash，query，fullPath，matched，name 等路由信息参数
- $router: 是路由的跳转方法，钩子函数等

2.1.3、angularJS

1、angularJS 的适用场景

AngularJS 是为了克服 HTML 在构建应用上的不足而设计的。HTML 是一门很好的为静态文本展示设计的声明式语言，但要构建 WEB 应用的话它就显得乏力了，所以说 angular 适合做单页面应用程序

2、angularJS 是什么

AngularJS 是一个 JavaScript 框架。它可通过 <script> 标签添加到 HTML 页面。

AngularJS 通过 指令 扩展了 HTML，且通过 表达式 绑定数据到 HTML

3、angularJS 中的 MVVM 模式

采用这种方式为合理组织代码提供了方便，降低了代码间的耦合度，功能结构清晰可见。

Model：一般用来处理数据，包括读取和设置数据，一般指的是操作数据库。

View：一般用来展示数据，就是放数据，比如通过 HTML 来展示。

Controller：因为一个模块里面可能有多个视图和模型，控制器就起到了连接模型和视图的作用。

VM--> ViewModel（视图模型） 也就是$scope。

MVC 模式在 AngularJS 中的体现：

一、首先定义应用，采用 ng-app 属性指定一个应用，表示此标签内所包裹的内容都属于 APP 的一部分。

```
<html ng-app="App" lang="en">
```

二、定义模块

AngularJS 提供了一个全局对象 angular，在此全局对象下提供了很多属性和方法，其中的 angular.module()方法用来定义一个模块

```
//第一参数表示模块的名称，第二个参数表示此模块依赖的其它模块 var app = angular.module('app',[]);
```

三、定义控制器

控制器作为模型和视图的桥梁，只要我们定义好了控制器，就相当于定义好了模型和视图。

```
app.controller('StudentController',['$scope',function($scope){//定义模型,这个$scope 对象就相当于用来获取数据的。
    $scope.students = [
      {name:'周杰伦',sex:'男',age:39}
    ];
    }]);
```

控制器定义好了，但是要讲模型上的数据展示到视图上，就需要将控制器关联到视图上，通过为 HTML 标签添加 ng-controller 属性并赋值相应的控制器名称，就确定了关联关系。

```
    <table ng-controller='StudentController'>
      <tr><th> 姓 名 </th><th> 性 别 </th><th> 年 龄 </th></tr><tr  ng-repeat="student  in
students">
        <td>{{student.name}}</td>
        <td>{{student.sex}}</td>
        <td>{{student.age}}</td></tr></table>
```

4、模块化与依赖注入

AngularJS 模块：

AngularJS 模块是一种容器，把代码隔离并组织成简洁，整齐，可复用的块。

模块本身不提供直接的功能：包含其他提供功能的对象的实例：控制器，过滤器，服务，动画。可通过定义其提供的对象构建模块。通过依赖注入将模块连接在一起，构建一个完整的应用程序。

AngularJS 建立在模块原理上。大部分 AngularJS 提供的功能都内置到 ng-*模块中。

创建模块：

你可以通过 AngularJS 的 angular.module 函数来创建模块：

```
    <div ng-app="myApp">...</div>

<script>

var app = angular.module("myApp", []);
    </script>
```

依赖注入：（Dependency Injection，简称 DI）是一种软件设计模式，在这种模式下，一个或更多的依赖（或服务）被注入（或者通过引用传递）到一个独立的对象（或客户端）中，然后成为了该客户端状态的一部分。

该模式分离了客户端依赖本身行为的创建，这使得程序设计变得松耦合，并遵循了依赖反转和单一职责原则。与服务定位器模式形成直接对比的是，它允许客户端了解客户端如何使用该系统找到依赖

一句话 ，没事你不要来找我，有事我会去找你。

AngularJS 提供很好的依赖注入机制。以下 5 个核心组件用来作为依赖注入：value、factory、service、provider、constant

5、看过 Angular 的源码吗，它是怎么实现双向数据绑定的?

答：angular 对常用的 dom 事件，xhr 事件等做了封装， 在里面触发进入 angular 的 digest 流程。
在 digest 流程里面， 会从 rootscope 开始遍历， 检查所有的 watcher。

6、为什么 angular 不推荐使用 dom 操作?

答：Angular 倡导以测试驱动开发，在的 service 或者 controller 中出现了 DOM 操作，那么也就意味着的测试是无法通过的
使用 Angular 的其中一个好处是啥，那就是双向数据绑定，这样就能专注于处理业务逻辑，无需关系一堆堆的 DOM 操作。如果在 Angular 的代码中还到处充斥着各种 DOM 操作，那为什么不直接使用 jquery 去开发呢。

7、指令

AngularJS 通过被称为 指令 的新属性来扩展 HTML。AngularJS 通过内置的指令来为应用添加功能。AngularJS 允许你自定义指令。

AngularJS 指令:

AngularJS 指令是扩展的 HTML 属性，带有前缀 ng-。
ng-app 指令初始化一个 AngularJS 应用程序。
ng-init 指令初始化应用程序数据。
ng-model 指令把元素值（比如输入域的值）绑定到应用程序。

自定义的指令:

除了 AngularJS 内置的指令外，我们还可以创建自定义指令。

你可以使用 .directive 函数来添加自定义的指令。

要调用自定义指令，HTML 元素上需要添加自定义指令名。

使用驼峰法来命名一个指令， runoobDirective，但在使用它时需要以 - 分割, runoob-directive:

```
    <body ng-app="myApp">

<runoob-directive></runoob-directive>

<script>
var app = angular.module("myApp", []);
app.directive("runoobDirective", function() {
    return {
        template : "<h1>自定义指令!</h1>"
    };
});
</script>

</body>
```

自定义指令的类型：restrict 值可以是以下几种:

E 作为元素名使用

A 作为属性使用

C 作为类名使用

M 作为注释使用

restrict 默认值为 EA，即可以通过元素名和属性名来调用指令。

8、路由

angularjs 是特别适合单页面应用，为了通过单页面完成复杂的业务功能，势必需要能够从一个视图跳转到另外一个视图，也就是需要在单个页面里边加载不同的模板。为了完成这个功能 angularjs 为我们提供了路由服务($routeProvider)。

实例解析：

1、载入了实现路由的 js 文件：angular-route.js。

2、包含了 ngRoute 模块作为主应用模块的依赖模块。

```
    angular.module('routingDemoApp',['ngRoute'])
```

3、使用 ngView 指令。

```
<div ng-view></div>
```

该 div 内的 HTML 内容会根据路由的变化而变化。

4、配置 $routeProvider，AngularJS $routeProvider 用来定义路由规则。

```
module.config(['$routeProvider', function($routeProvider){
    $routeProvider
        .when('/',{template:'这是首页页面'})
        .when('/computers',{template:'这是电脑分类页面'})
        .when('/printers',{template:'这是打印机页面'})
        .otherwise({redirectTo:'/'});}]);
```

2.1.4、ReactJS

1、ReactJS 的适用场景

1、一些后台界面，或者是和后台数据比较多，又或者和用户交互比较多，dom 操作频繁的都可以用 react。因为 react 的大特点就是虚拟 DOM 技术，这样可以提高渲染的性能。个人的体验就是，用 react 可以让整个页面的速度提高很多。

2、另外这样做的话，和后台的数据通过接口来进行前后端分离，也挺好挺方便的。

2、ReactJS 是什么

React 是一个用于构建用户界面的 JAVASCRIPT 库。

React 主要用于构建 UI，很多人认为 React 是 MVC 中的 V（视图）。

React 起源于 Facebook 的内部项目，用来架设 Instagram 的网站，并于 2013 年 5 月开源。

React 拥有较高的性能，代码逻辑非常简单，越来越多的人已开始关注和使用它。

3、ReactJs 的特点

1.声明式设计 –React 采用声明范式，可以轻松描述应用。

2.高效 –React 通过对 DOM 的模拟，最大限度地减少与 DOM 的交互。

3.灵活 –React 可以与已知的库或框架很好地配合。

4.JSX – JSX 是 JavaScript 语法的扩展。React 开发不一定使用 JSX ，但我们建议使用它。

5.组件 – 通过 React 构建组件，使得代码更加容易得到复用，能够很好的应用在大项目的开发中。

6.单向响应的数据流 – React 实现了单向响应的数据流，从而减少了重复代码，这也是它为什么比传统数据绑定更简单。

使用 react 项目的有：

1、去哪儿网给航空公司用的收益辅助系统。采用的是 react+fluxxor

2、目前阿里内部一些系统在用，支付宝新一代的框架基于 React，可以关注下

3、美团很多内部系统都在用

4、react Native 比起标准 Web 开发或原生开发能够带来的三大好处：

（1）、手势识别：基于 Web 技术（HTML5/JavaScript）构建的移动应用经常被抱怨缺乏及时响应。而基于原生 UI 的 React Native 能避免这些问题从而实现实时响应。

（2）、原生组件：使用 HTML5/JavaScript 实现的组件比起原生组件总是让人感觉差一截，而 React Native 由于采用了原生 UI 组件自然没有此问题。

（3）、样式和布局：iOS、Android 和基于 Web 的应用各自有不同的样式和布局机制。

React Native 通过一个基于 FlexBox 的布局引擎在所有移动平台上实现了一致的跨平台样式和布局方案。

2.1.5、NodeJS

1、NodeJS 的适用场景

1. RESTful API

这是 NodeJS 最理想的应用场景，可以处理数万条连接，本身没有太多的逻辑，只需要请求 API，组织数据进行返回即可。它本质上只是从某个数据库中查找一些值并将它们组成一个响应。由于响应是少量文本，入站请求也是少量的文本，因此流量不高，一台机器甚至也可以处理最繁忙的公司的 API 需求。

2. 统一 Web 应用的 UI 层

目前 MVC 的架构，在某种意义上来说，Web 开发有两个 UI 层，一个是在浏览器里面我们最终看到的，另一个在 server 端，负责生成和拼接页面。

不讨论这种架构是好是坏，但是有另外一种实践，面向服务的架构，更好的做前后端的依赖分离。如果所有的关键业务逻辑都封装成 REST 调用，就意味着在上层只需要考虑如何用这些 REST 接口构建具体的应用。那些后端程序员们根本不操心具体数据是如何从一个页面传递到另一个页面的，他们也不用管用户数据更新是通过 Ajax 异步获取的还是通过刷新页面。

3. 大量 Ajax 请求的应用

例如个性化应用，每个用户看到的页面都不一样，缓存失效，需要在页面加载的时候发起 Ajax 请求，NodeJS 能响应大量的并发请求。 总而言之，NodeJS 适合运用在高并发、I/O 密集、少量业务逻辑的场景。

2、NodeJS 是什么

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。

Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。

Node.js 的包管理器 npm，是全球最大的开源库生态系统。

NodeJS 的特点

其特点为：

1. 它是一个 Javascript 运行环境
2. 依赖于 Chrome V8 引擎进行代码解释
3. 事件驱动
4. 非阻塞 I/O
5. 轻量、可伸缩，适于实时数据交互应用
6. 单进程，单线程

NodeJS 的优缺点：

优点：1. 高并发（最重要的优点）2. 适合 I/O 密集型应用

缺点：1. 不适合 CPU 密集型应用；CPU 密集型应用给 Node 带来的挑战主要是：由于 JavaScript 单线程的原因，如果有长时间运行的计算（比如大循环），将会导致 CPU 时间片不能释放，使得后续 I/O 无法发起；

2. 只支持单核 CPU，不能充分利用 CPU

3. 可靠性低，一旦代码某个环节崩溃，整个系统都崩溃

4. 开源组件库质量参差不齐，更新快，向下不兼容

5. Debug 不方便，错误没有 stack trace

原因：单进程，单线程

解决方案：（1）分解大型运算任务为多个小任务，使得运算能够适时释放，不阻塞 I/O 调用的发起；（2）Nnigx 反向代理，负载均衡，开多个进程，绑定多个端口；

（3）开多个进程监听同一个端口，使用 cluster 模块；

3、NodeJS 非阻塞 I/O 模型执行流程

主线程：

1.执行 node 的代码，把代码放入队列

2.事件循环程序（主线程）把队列里面的同步代码都先执行了，

3.同步代码执行完成，执行异步代码

4.异步代码分 2 种状况，

(1)、异步非 io setTimeout() setInterval()

判断是否可执行，如果可以执行就执行，不可以跳过。

(2)、异步 io 文件操作

会从线程池当中去取一条线程，帮助主线程去执行。

5 主线程会一直轮训，队列中没有代码了，主线程就会退出。

子线程：被放在线程池里面的线程，用来执行异步 io 操作

1.在线程池里休息

2.异步 io 的操作来了，执行异步 io 操作。

3.子线程会把异步 io 操作的 callback 函数，扔回给队列

4.子线程会回到线程池了去休息。

callback，在异步 io 代码执行完成的时候被扔回主线程。

4、node 核心内置类库(事件，流，文件，网络等)

1.node 的构架是什么样子的?

答案: 主要分为三层，应用 app >> V8 及 node 内置架构 >> 操作系统. V8 是 node 运行的环境，可以理解为 node 虚拟机．node 内置架构又可分为三层: 核心模块 (javascript 实现) >> c++绑定 >> libuv + CAes + http.

2. node 有哪些核心模块?

答案: EventEmitter, Stream, FS, Net 和全局对象

3. node 有哪些全局对象?

答案: process, console, Buffer 和 exports

4. process 有哪些常用方法?

答案: process.stdin, process.stdout, process.stderr, process.on, process.env, process.argv, process.arch, process.platform, process.exit

5. console 有哪些常用方法?

答案: console.log/console.info, console.error/console.warning, console.time/console.timeEnd, console.trace, console.table

6. node 有哪些定时功能?

答案: setTimeout/clearTimeout, setInterval/clearInterval, setImmediate/clearImmediate, process.nextTick

7. 什么是 EventEmitter?

参考答案: EventEmitter 是 node 中一个实现观察者模式的类，主要功能是监听和发射消息，用于处理多模块交互问题.

8. EventEmitter 有哪些典型应用?

参考答案: 1) 模块间传递消息 2) 回调函数内外传递消息 3) 处理流数据，因为流是在 EventEmitter 基础上实现的. 4) 观察者模式发射触发机制相关应用

9. 怎么捕获 EventEmitter 的错误事件?

参考答案: 监听 error 事件即可．如果有多个 EventEmitter,也可以用 domain 来统一处理错误事件.

5、nodejs 中流(stream)的理解

1. 什么是 Stream?

答案: stream 是基于事件 EventEmitter 的数据管理模式．由各种不同的抽象接口组成，主要包括可写，可读，可读写，可转换等几种类型.

2. Stream 有什么好处?

答案: 非阻塞式数据处理提升效率，片断处理节省内存，管道处理方便可扩展等.

3. Stream 有哪些典型应用?

答案: 文件，网络，数据转换，音频视频等.

4. 怎么捕获 Stream 的错误事件?

答案: 监听 error 事件，方法同 EventEmitter.

5. 有哪些常用 Stream,分别什么时候使用?

答案: Readable 为可被读流，在作为输入数据源时使用；Writable 为可被写流,在作为输出源时使用；Duplex 为可读写流,它作为输出源接受被写入，同时又作为输入源被后面的流读出．Transform 机制和 Duplex 一样，都是双向流，区别时 Transfrom 只需要实现一个函数_transfrom(chunk, encoding, callback);而 Duplex 需要分别实现_read(size)函数和_write(chunk, encoding, callback)函数.

6. 实现一个 Writable Stream?

答案: 三步走:1)构造函数 call Writable 2) 继承 Writable 3) 实现_write(chunk, encoding, callback)函数

NodeJs 面试题大全详见:

http://www.cnblogs.com/meteorcn/p/node_mianshiti_interview_question.html

2.1.6、less/sass 预处理

1、为什么要使用 less/sass 预处理器（优点、缺点）

(1)、 CSS 无法递归式定义

(2)、CSS 的 mixin 式复用性支持不够

(3)、预编译可缓解多浏览器兼容造成的冗余(4)css 并不能算是一们真正意义上的“编程”语言，它本身无法未完成像其它编程语言一样的嵌套、继承、设置变量等工作。为了解决 css 的不足，开发者们想到了编写一种对 css 进行预处理的“中间语言”，可以实现一些“编程”语言才有的功能，然后自动编译成 css 供浏览识别，这样既一定程度上弥补了 css 的不足，也无需一种新的语言来代替 css 以供浏览器识别。于是 css 预处理语言就应运而生了。

(4)、CSS 预处理语言，它扩展了 CSS 语言，增加了变量、Mixin、函数等特性，使 CSS 更易维护和扩展。

预处理器的缺点：(1)、CSS 的好处在于简便、随时随地被使用和调试。预编译 CSS 步骤的加入，让我们开发工作流中多了一个环节，调试也变得更麻烦了。(2)、预编译很容易造成后代选择器的滥用因此，实际项目中衡量预编译方案时，还是得想想，比起带来的额外维护开销，预编译有没有解决更大的麻烦。降低了自己对最终代码的控制力。(3)、更致命的是提高了门槛，首先是上手门槛，其次是维护门槛，再来是团队整体水平和规范的门槛。这也造成了初学学习成本的昂贵。

2、less/sass 区别

(1)sass 是基于 Ruby 的，然后是在服务器端处理的。很多开发者不会选择 LESS 因为 JavaScript 引擎需要额外的时间来处理代码然后输出修改过的 CSS 到浏览器。

(2)关于变量在 LESS 和 Sass 中的唯一区别就是，LESS 用@，Sass 用$

2.1.7、App 相关技术介绍

1、原生 app,WEBAPP,混合 app 的差异

什么叫做原生 App?

原生 App 是专门针对某一类移动设备而生的，它们都是被直接安装到设备里，而用户一般也是通过网络商店或者卖场来获取

原生应用是特别为某种操作系统开发的，比如 iOS、Android、黑莓等等，它们是在各自的移动设备上运行的。

优点:

可访问手机所有功能（GPS、摄像头）；

速度更快、性能高、整体用户体验不错;

可线下使用（因为是在跟 Web 相对地平台上使用的）；

支持大量图形和动画;

容易发现（在 App Store 里面）和重新发现（应用图标会一直在主页上）；

应用下载能创造盈利（当然 App Store 抽取 20-30% 的营收）。

缺点:

开发成本高;

支持设备非常有限（一般是哪个系统就在哪个平台专属设备上用）；

上线时间不确定（App Store 审核过程不一）；

内容限制（App Store 限制）；

获得新版本时需重新下载应用更新。

什么叫做移动 Web App?

一般说来，移动 Web App 都是需要用到网络的，它们利用设备上的浏览器(比如 iPhone 的 Safari)来运行，而且它们不需要在设备上下载后安装。

Web 应用本质上是为移动浏览器设计的基于 Web 的应用，它们是用普通 Web 开发语言开发的，可以在各种智能手机浏览器上运行。

优点:

支持设备广泛;

较低的开发成本;

可即时上线;

无内容限制

用户可以直接使用最新版本（自动更新，不需用户手动更新）。

缺点:

表现略差（对联网的要求比较大）;

用户体验没那么炫;

图片和动画支持性不高;

没法在 App Store 中下载、无法通过应用下载获得盈利机会；

要求联网；

对手机特点有限制（摄像头、GPS 等）。

对于这些缺点，如果能把 HTML 5 的优点用到 Web 上的话就会得到很大改善，尽管技术在提高，目前它还不能做原生应用可以做的每件事。

有些公司，比如金融时报继原生应用后也开发了 Web 应用，用户通过浏览器来访问他们的应用，因为他们的应用要采取应用内购买，而 App Store 是不允许应用内购买的，不然它们就无法拿到那 20%-30% 的抽成了。

什么是混合 app?

Hybrid App 是指介于 web-app、native-app 这两者之间的 app,它虽然看上去是一个 Native App，但只有一个 UI WebView，里面访问的是一个 Web App，比如街旁网最开始的应用就是包了个客户端的壳，其实里面是 HTML5 的网页，后来才推出真正的原生应用。再彻底一点的，如掌上百度和淘宝客户端 Android 版，走的也是 Hybrid App 的路线，不过掌上百度里面封装的不是 WebView，而是自己的浏览内核，所以体验上更像客户端，更高效。

Hybrid App（混合模式移动应用）兼具“Native App 良好用户交互体验的优势”和“Web App 跨平台开发的优势”。很多人不知道市场上一些主流移动应用都是基于 Hybrid App 的方式开发，比如国外有 Facebook、国内有百度搜索等。

综合一下就是：“Hybrid App 同时使用网页语言与程序语言开发，通过应用商店区分移动操作系统分发，用户需要安装使用的移动应用”。总体特性更接近 Native App 但是和 Web App 区别较大。只是因为同时使用了网页语言编码，所以开发成本和难度比 Native App 要小很多。因此说，Hybrid App 兼具了 Native App 的所有优势，也兼具了 Web App 使用 HTML5 跨平台开发低成本的优势

混合应用大家都知道是原生应用和 Web 应用的结合体，采用了原生应用的一部分、Web 应用的一部分，所以必须，部分在设备上运行、部分在 Web 上运行。

不过混合应用中比例很自由，比如 Web 占 90%，原生占 10%；或者各占 50%。

优点：

兼容多平台；

顺利访问手机的多种功能；

App Store 中可下载（Web 应用套用原生应用的外壳）；

可线下使用。

缺点：

不确定上线时间；

用户体验不如本地应用；

性能稍慢（需要连接网络）；

技术还不是很成熟。

比如 Facebook 现在的应用属于混合应用它可以在许多 App Store 畅通无阻，但是掺杂了大量 Web 特性，所以它运行速度比较慢，而现在为了提高性能 FB 又决定采用原生应用。

Web App、Hybrid App、Native APP对比

	Web App（网页应用）	Hybrid App（混合应用）	Native App（原生应用）
开发成本	低	中	高
维护更新	简单	简单	复杂
体验	差	中	优
Store或market认可	不认可	认可	认可
安装	不需要	需要	需要
跨平台	优	优	差

以下是列举的一些重点差异：

- 开发难度。移动 web 和混合 App 开发难度对于 web 开发者来说相对较低，而且可以充分利用现有的 web 开发工具和工作流程
- 发布渠道和更新方式。混合 App 可以在应用商店 App Store 发布，但可以自主更新，而原生 App 的更新必须通过应用商店 App Store。
- 移动设备本地 API 访问。混合 App 可以通过 JavaScript API 访问到移动设备的摄像头、GPS；而原生 App 可以通过原生编程语言访问设备所有功能。

● 跨平台和可移植性。基于浏览器的移动 web 最好的可移植性和跨平台表现；混合 App 也能节省跨平台的时间和成本，只需编写一次核心代码就可部署到多个平台，而原生 App 的跨平台性能最差。

● 搜索引擎友好。只有移动 web 对搜索引擎友好，可与在线营销无缝整合。

● 货币化。混合 App 除广告外，还支持付费下载及程序内购买；原生 App 的程序内购买金额 2012 年首次超过下载收费。

● 消息推送。只有混合 App 和原生 App 支持消息推送，这能增加用户忠诚度。

结论总结：选择合适的应用需要考虑些什么，比如：1、是否需要使用某些设备的特殊功能如：摄像头、摄像头闪光灯或者重力加速器 2、你的开发预算 3、你的应用是否一定需要网络 3 你的应用的目标硬件设备是所有的移动设备还是仅仅只是一部分而已 4、你自己已经熟悉的开发语言 5、这个应用对于性能要求是否苛刻 6、如何靠这个应用赢利

（详见：http://www.cnblogs.com/huanghundaxia/p/5734209.html）

2、ionic 流行开发框架（混合 APP）

Ionic 介绍：

1、ionic 是一个用来开发混合手机应用的，开源的，免费的代码库。可以优化 html、css 和 js 的性能，构建高效的应用程序，而且还可以用于构建 Sass 和 AngularJS 的优化。ionic 会是一个可以信赖的框架。

2、ionic 是一个专注于用 WEB 开发技术，基于 HTML5 创建类似于手机平台原生应用的一个开发框架。绑定了 AngularJS 和 Sass。这个框架的目的是从 web 的角度开发手机应用，基于 PhoneGap 的编译平台，可以实现编译成各个平台的应用程序。

ionic 的开发添加 android 和 ios 环境。

ionic 提供很多 css 组件 和 javascript UI 库。

ionic 可以支持定制 android 和 ios 的插件，也支持服务端 REST 的敏捷开发。

ionic 特点：

1.ionic 基于 Angular 语法，简单易学。

2.ionic 是一个轻量级框架。

3.ionic 完美的融合下一代移动框架，支持 Angularjs 的特性， MVVM ，代码易维护。

4.ionic 提供了漂亮的设计，通过 SASS 构建应用程序，它提供了很多 UI 组件来帮助开发者开发强大的应用。

5.ionic 专注原生，让你看不出混合应用和原生的区别

6.ionic 提供了强大的命令行工具。

7.ionic 性能优越，运行速度快。

Ionic 中文教程连接：http://www.runoob.com/ionic/ionic-tutorial.html

2.1.8、前端工具介绍或使用方法

1、前端工程化工具

发展历程：grunt -->gulp -->webpack

gulp 较之 grunt 的优势：

1、易用， Gulp 相比 Grunt 更简洁，而且遵循代码优于配置策略，维护 Gulp 更像是写代码。

2、高效， Gulp 相比 Grunt 更有设计感，核心设计基于 Unix 流的概念，通过管道连接，不需要写中间文件。

3、高质量， Gulp 的每个插件只完成一个功能，这也是 Unix 的设计原则之一，各个功能通过流进行整合并完成复杂的任务。例如：Grunt 的 imagemin 插件不仅压缩图片，同时还包括缓存功能。他表示，在 Gulp 中，缓存是另一个插件，可以被别的插件使用，这样就促进了插件的可重用性。目前官方列出的有 673 个插件。

4、易学， Gulp 的核心 API 只有 5 个，掌握了 5 个 API 就学会了 Gulp，之后便可以通过管道流组合自己想要的任务。

5、流， 使用 Grunt 的 I/O 过程中会产生一些中间态的临时文件，一些任务生成临时文件，其它任务可能会基于临时文件再做处理并生成最终的构建后文件。而使用 Gulp 的优势就是利用流的方式进行文件的处理，通过管道将多个任务和操作连接起来，因此只有一次 I/O 的过程，流程更清晰，更纯粹。

6、代码优于配置， 维护 Gulp 更像是写代码，而且 Gulp 遵循 CommonJS 规范，因此跟写 Node 程序没有差别。

2、Webpack

1、什么是 Webpack

WebPack 可以看做是模块打包机：它做的事情是，分析你的项目结构，找到 JavaScript 模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript 等），并将其打包为合适的格式以供浏览器使用。

2、为什要使用 WebPack

今的很多网页其实可以看做是功能丰富的应用，它们拥有着复杂的 JavaScript 代码和一大堆依赖包。为了简化开发的复杂度，前端社区涌现出了很多好的实践方法

a:模块化，让我们可以把复杂的程序细化为小的文件;

b:类似于 TypeScript 这种在 JavaScript 基础上拓展的开发语言：使我们能够实现目前版本的 JavaScript 不能直接使用的特性，并且之后还能能装换为 JavaScript 文件使浏览器可以识别;

c:scss，less 等 CSS 预处理器

……

这些改进确实大大的提高了我们的开发效率，但是利用它们开发的文件往往需要进行额外的处理才能让浏览器识别,而手动处理又是非常反锁的，这就为 WebPack 类的工具的出现提供了需求。

3、WebPack 和 Grunt 以及 Gulp 相比有什么特性

其实 Webpack 和另外两个并没有太多的可比性，Gulp/Grunt 是一种能够优化前端的开发流程的工具，而 WebPack 是一种模块化的解决方案，不过 Webpack 的优点使得 Webpack 可以替代 Gulp/Grunt 类的工具。

Grunt 和 Gulp 的工作方式是：在一个配置文件中，指明对某些文件进行类似编译，组合，压缩等任务的具体步骤，这个工具之后可以自动替你完成这些任务。

Webpack 的工作方式是：把你的项目当做一个整体，通过一个给定的主文件（如：index.js），Webpack 将从这个文件开始找到你的项目的所有依赖文件，使用 loaders 处理它们，最后打包为一个浏览器可识别的 JavaScript 文件。

优点：模块化

在 webpack 看来一切都是模块！这就是它不可不说的优点，包括你的 JavaScript 代码，也包括 CSS 和 fonts 以及图片等等，只要通过合适的 loaders，它们都可以被当做模块被处理。

Webpack：https://segmentfault.com/a/1190000006178770#articleHeader8

3、svn 和 git 的区别

1、GIT 是分布式的，SVN 是集中式的

2、git 是每个历史版本都存储完整的文件,便于恢复,svn 是存储差异文件,历史版本不可恢复。(核心)

3、git 可离线完成大部分操作,svn 则不能。

4、git 有着更优雅的分支和合并实现。

5、git 有着更强的撤销修改和修改历史版本的能力

6、Git 下载下来后，在本地不必联网就可以看到所有的 log，很方便学习，SVN 却需要联网；git 速度更快,效率更高。

7、Git 没有一个全局的版本号，而 SVN 有。

Svn 本地管理详见：http://www.cnblogs.com/pingwen/p/5152684.html

Git 版本管理详见：http://www.cnblogs.com/grimm/p/5368759.html

4、Git/GitHub

Git 是一个开源的分布式版本控制系统，用以有效、高速的处理各种规模的项目版本管理， 它是 Linux Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件，后来得到广泛的使用。

GitHub 是一个面向开源及私有软件项目的托管平台，因为只支持 Git 作为唯一的版本库格式进行托管，故名 GitHub。除了 Git 代码仓库托管及基本的 Web 管理界面以外，还提供了订阅、讨论组、文本渲染、在线文件编辑器、协作图谱（报表）、代码片段分享（Gist）等功能。目前，其注册用户已经超过 350 万，托管版本数量也是非常之多，其中不乏知名开源项目 Ruby on Rails、jQuery、python 等。

2.2、如何表述项目

2.2.1、项目描述方式

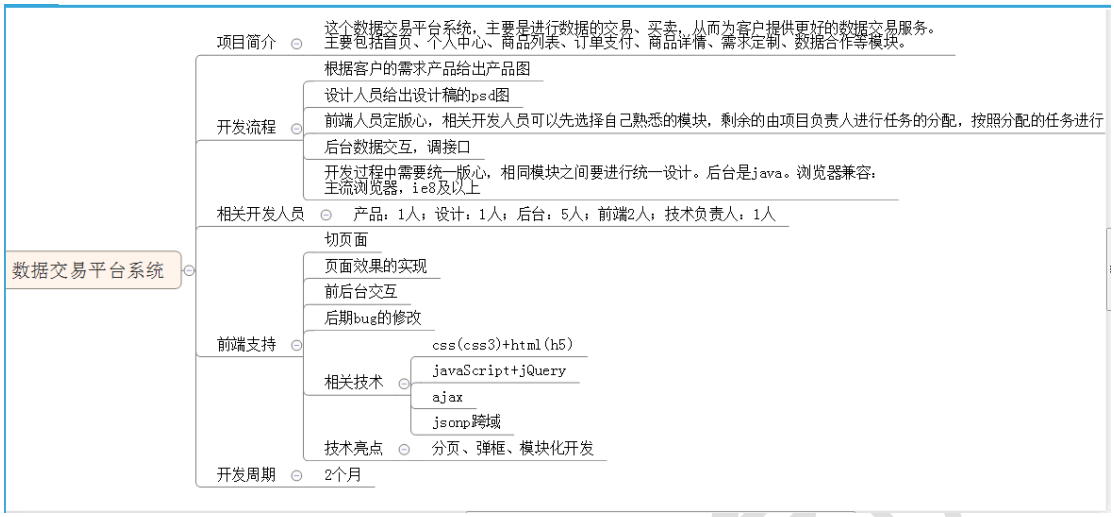
1、项目功能模块分析

项目功能模块分析结合自己的知识技术水平以及面试公司会用到的相关技术点阐述我们的项目，吸引面试官，让他感觉你跟他们公司的气质很适合。

2、项目如何开展分配

从项目的需求分析、原型定制、人员分配、技术定性、技术讨论、注意事项等展开我们开发这个项目的一个流程分析，可以加上开发过程中遇到的问题，怎么解决的，或者加上自己的见解等。

2.2.2、项目口述、面述话术模板



1、基于项目功能模块分析

1、介绍项目主要是干嘛的，例如：数据交易平台系统主要是主要是进行数据的交易、买卖，从而为客户提供更好的数据交易服务。主要包括首页、个人中心、商品列表、订单支付、商品详情、需求定制、数据合作等模块。

2、介绍自己负责的部分和使用的技术，例如：在这个项目中我负责的主要是个人中心这个模块的开发，包括静态页面和数据交互。开发过程中主要用到的技术有 H5+css3、ajax、插件（分页、弹框、模块化开发）。

3、然后结合自己的项目介绍，说什么技术实现了什么功能（如果产品上线可以给面试官看）4、此时你基本上介绍完毕，面试官如果对你介绍的感兴趣，他会问你的，如果他没有问，你可以问下该公司主要业务是什么，用什么技术开发，寻找共同点，展开话题

2、从项目如何开展分配分析

1、首先介绍项目主要是干嘛的，例如：数据交易平台系统主要是进行数据的交易、买卖，从而为客户提供更好的数据交易服务。主要包括首页、个人中心、商品列表、订单支付、商品详情、需求定制、数据合作等模块。

2、该项目结合用户提出的需求进行制定，技术人员讨论技术点的实现，结合考虑多方面的内容，定制产品模型。

3、产品模型出来之后，ui 设计师，设计出来 psd 图，同时前端人员可以做些准备工作（比如定版心，定框架、浏览器兼容版本等）

4、前端人员任务分配：可以先自行选择，然后项目经理进行 2 次分配。

5、确定时间节点，开发周期等比如：静态页面完成时间、与后台交互时间、测试时间等。

6、可以说一下在开发过程中遇到的一些问题，比如双边距问题，输入框默认值问题，讲一下怎么解决的，总结自己的感想。

7、介绍基本完成后可以适当与面试官进行交流，问下该公司主要业务是什么，用什么技术开发，寻找共同点，展开下一步的话题，可以占据主动性。