面试宝典

一、 静态页面

1、什么是 PC 端，什么是移动端 ？

移动端是指在手机、平板等移动设备上面使用客户端软件，PC 端一般是指在电脑上面使用的客户端软件。

2、 图片加载优化问题

先加载一张很小的缩略图，它可能只有 1K 左右，并且按所需尺寸模糊显示，等到大尺寸图片下载好再完整显示大图。这样给用户的感觉是先加载模糊的图片，然后再是变清晰的图片，图片加载过程流畅，极大的提高用户体验。Antimoderate.js 使用了 html5 的 canvans 将图片做了处理，优先加载缩略图，并做模糊化，当大图下载完成时加载大图显示，使用它极大的加快了网页打开速度。

3、 h5 跟 h4 的区别

H5 推出的理由及目标

1. web 浏览器之间的兼容性很低
2. 文档结构不够明确
3. web 应用程序功能受到了限制

H5 语法的改变

内容类型　文件扩展名 html 内容类型不变 依然为 text-html

doctype 声明 直接<!doctype html>

指定字符编码　<meta charset="UTF-8"/>

可以省略标记的元素

具有 boolean 值的属性

省略引号

HTML5 新增的元素和废除的

新增结构元素: section(文档结构) article(与上下文不相关的内容) aside (artice 内容相关的辅助信息)

header (取代 div 做页面头部) hgroup (与标题结合使用<h1>-<h6>) footer (做页面底部使用) nav (导航连接) figure (独立流内容 独立单元)

其他: video audio embed mark progress meter time ruby rt rp wbr canvas (画布,可以通过脚本将想绘制的放于画布.而达到效果) command details datalist

datagrid keygen output source menu

新增 input 类型元素 ：

email url number range(输入的数字范围值) Date Pickers (日历 日期)

废除的元素 ：

1. 能使用 css 代替的元素 ： basefont big center font s tt u
2. 不再使用 frame 框架
3. 只有部分浏览器支持的元素
4.其他被废除的元素

新增的属性

1. 表单相关的属性
2. 链接相关的属性
3.其他属性

废除的属性

全局属性

contentEditable 属性 允许用户编辑元素中的内容 boolean 类型 有继承性

designMode 属性 指定整个页面是否可编辑 (on 可编辑/off 不可编辑)

hidden 属性 不渲染元素 隐藏元素 boolean 类型

spellcheck 对用户输入的内容进行语法检查

tabindex 遍历 tab 元素 按 tabindex 访问元素 调整为-1 时 无法获取焦点 但在程序中可以获取到

4、 css3 的新特性

选择器、 RGBA 和透明度、多栏布局、 多背景图、 Word Wrap、文字阴影、@font-face 属性、圆角(边框半径)、边框图片、 盒阴影 、 盒子大小、 媒体查询、 语音

5、怎么给加载不出来的图片添加一个默认的图片，点击刷新该图片

1、在 img 标签中加上 onerror="this.src='error.png ' ";

<img src="Images/1.png " onerror="this.src='error.png ' "

>

2、不想在每个 img 中都定义 onerror 事件的话，就使用 jquery 试试

JavaScript code

```
$(window).load(function() {
    $('img').each(function() {
      if (!this.complete || typeof this.naturalWidth ==
```

```
"undefined" || this.naturalWidth == 0) {
          this.src = './image/logo.gif';
          }
       });
    });
```

3、但是都要考虑，重载的图片仍然错误，就会陷入死循环
下面给出一个带重试次数，并且延迟加载的实现，超过重试次数仍

不能正常显示的，显示缺省图片

```
<scripttype="text/javascript">

functionshowImgDelay(imgObj,imgSrc,maxErrorNum){
   showSpan.innerHTML += "--" +maxErrorNum;   //显示出加载

图片出错的次数

   if(maxErrorNum>0){
       imgObj.onerror=function(){
           showImgDelay(imgObj,imgSrc,maxErrorNum-1);
       };
       setTimeout(function(){
           imgObj.src=imgSrc;
       },500);
   }else{
       imgObj.onerror=null;
       imgObj.src="images/default.jpg";
   }
}

</script>
<img onerror="showImgDelay(this,'123456.png',2);"

src="666.png" width="200" height="200"/>
```

6、 垂直居中的方式有哪些?

http://www.cnblogs.com/haoyijing/p/5815394.html#css1

7、 一个图片如何让它垂直居中；

题目的难点在于两点：

1.垂直居中；

2.图片是个置换元素，有些特殊的特性。

至于如何解决，下面是一个权衡的相对结构干净，CSS 简单的解决方法：

```
.box {
 /*非 IE 的主流浏览器识别的垂直居中的方法*/
 display: table-cell;
 vertical-align:middle;
 /*设置水平居中*/
 text-align:center;
 /* 针对 IE 的 Hack */
 *display: block;
 *font-size: 175px;/*约为高度的 0.873，200*0.873 约为 175*/
 *font-family:Arial;/*防止非 utf-8 引起的 hack 失效问题，如 gbk 编码*/
 width:200px;
 height:200px;
 border: lpx solid #eee;
}
.box img {
 /*设置图片垂直居中*/
 vertical-align:middle;
}
″ />
```

8、 有没有使用过 less 或 sass；

用过 less.css 的预编译语言

9、 一般做动画效果，是使用 c3 还是 jq，animate 如何实现动画，需要传入哪些参数；

(主要是问两种动画的区别)；

CSS3 的动画

优点：

1.在性能上会稍微好一些，浏览器会对 CSS3 的动画做一些优化（比如专门新建一个图层用来跑动画）

2.代码相对简单

缺点：

1.在动画控制上不够灵活

2.兼容性不好

3.部分动画功能无法实现（如滚动动画，视差滚动等）

Jquery 的动画

优点：

1.控制能力很强，可以单帧的控制、变换

2.兼容性好，写得好完全可以兼容 IE6，且功能强大。

缺点：

计算没有 css 快，另外经常需要依赖其他的库。

结论

所以，不复杂的动画完全可以用 css 实现，复杂一些的，或者需要交互的时候，用 js 会靠谱一些~

```
$(".btn1").click(function(){
  $("#box").animate({height:"300px"});
});
```

可选。规定动画的额外选项。

可能的值：

speed - 设置动画的速度

easing - 规定要使用的 easing 函数

callback - 规定动画完成之后要执行的函数

step - 规定动画的每一步完成之后要执行的函数

queue - 布尔值。指示是否在效果队列中放置动画。如果为 false，则动画将立即开始

specialEasing - 来自 styles 参数的一个或多个 CSS 属性的映射，以及它们的对应 easing 函数

10、css 布局

11、写一个首页用百分比　头部两个盒子左边一个右边一个　下面盒子内容一样

12、盒子居中

```
mrgin:0  auto;
```

13、定位的几种方式

4 种不同类型的定位。
static：默认值，没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。
relative：生成相对定位的元素，相对于其正常位置进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。
fixed：元素框的表现类似于将 position 设置为 absolute，不过其包含块是视窗本身。
absolute：生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。
元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。

14、伸缩布局居中

```
<style type="text/css">
    .big{
        width: 100%;
        background-color:;
        height: 100%
    }
    .small{
        width: 30%;
        background: green;
        margin: 0 auto;
        height: 30px;
    }
    </style>
</head>
<body>
    <div class="big">
        <div class="small"></div>
```

```
    </div>
</body>
```

16、H5 增加了什么新属性

http://www.w3school.com.cn/html5/html_5_form_attributes.asp

18、会不会离线缓存想

http://www.cnblogs.com/wuzhiquan/p/4843740.html

19、兼容问题

20、性能优化

21、Html 和 XML 的区别

http://www.cnblogs.com/lkjson/p/4337754.html

22、移动端用的什么适配

适配的要求
1、在不同分辨率的手机上，页面看起来是自适应的。整体效果看起来比较和谐。不会说大屏幕上看起来特别小。小屏幕上看起来特别大
2、主要是关注字体，宽高，间距，图片大小等。
3、所提供的设计图一般是手机分辨率的两倍，才能方便做适配。
4、使用 rem 做单位，而不是传统的 px
http://www.jb51.net/article/102704.htm

23、用 less 是直接传到服务器上还是要编译 CCS

24、分页

pagination 分页插件：
http://www.zhangxinxu.com/jq/pagination_zh/

25、Ie 兼容

https://www.douban.com/note/163291324/

26、Less 中怎么设置全局变量

27、Less 怎么引用另一个 less

28、rem 怎么用

http://www.w3cplus.com/css3/define-font-size-with-css3-rem

31、简单 css，边距合并问题，盒模型，定

32、块级，行内

块级元素：div ，
p ， form， ul， li ， ol， dl， form， address， fieldset，
hr， menu， table
行内元素：
span， strong， em， br， img ， input， label， select， textarea，
cite，

33、本地存储和 cookies 的区别

1. cookie 在浏览器和服务器间来回传递。而 sessionStorage 和 localStorage 不会自动把

数据发给服务器，仅在本地保存。

2. cookie 数据还有路径（path）的概念，可以限制 cookie 只属于某个路径下。存储大小限制也不同，cookie 数据不能超过 4k，同时因为每次 http 请求都会携带 cookie，所以 cookie 只适合保存很小的数据，如会话标识。sessionStorage 和 localStorage 虽然也有存储大小的限制，但比 cookie 大得多，可以达到 5M 或更大。

3. 数据有效期不同，sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；cookie 只在设置的 cookie 过期时间之前一直有效，即使窗口或浏览器关闭。

4. 作用域不同，sessionStorage 不在不同的浏览器窗口中共享，即使是同一个页面；localStorage 在所有同源窗口中都是共享的；cookie 也是在所有同源窗口中都是共享的。Web Storage 支持事件通知机制，可以将数据更新的通知发送给监听者。Web Storage 的 api 接口使用更方便。

35、浮动、定位

36、 图片优化

1. 去掉无意义的修饰。
2. 不用图片。
3. 使用矢量图替代位图
4. 使用恰当的图片格式。
5. 使用 data url。
6. 按照 HTTP 协议设置合理的缓存。
7. 使用支持 SPDY 的服务器。
8. 资源的 lazyload 或 postpone。（lazyload：延迟到其他资源下载完成后再加载，postpone：延迟到元素可见再加载。）
9. 资源的 prefetch。可用 <link rel=prefetch>，见 http://www.whatwg.org/specs/web-apps/current-work/#link-type-prefetch。注意 prefetch 只是 hint，Firefox 会预取资源（如果网络空闲的话），而 IE 9 则是对该资源的 hostname 进行 DNS 预解析。如果你真的需要更强的控制，则得用脚本。注意：Chrome 支持与 prefetch 相近但更进一步的<link rel=prerender>

38、盒模型

CSS 盒模型本质上是一个盒子，封装周围的 HTML 元素，它包括：边距，边框，填充，和实

际内容。盒模型允许我们在其它元素和周围元素边框之间的空间放置元素。

39、 px 和 em 的区别

px 表示像素（计算机屏幕上的一个点：1px = 1/96in），是绝对单位 ，不会因为其他元素的尺寸变化而变化。
em 表示相对于父元素的字体大小。em 是相对单位 ，没有一个固定的度量值，而是由其他元素尺寸来决定的相对值。

40、html 兼容问题

41、css 兼容问题

42、pc 端和移动端兼容问题

http://blog.csdn.net/zhangmeng1020/article/details/50886185

43、清除浮动的原理

http://blog.csdn.net/clare504/article/details/39524215

44、bootstrap 经典布局是什么，有什么特点

45、display 和 visiable

45、css3 新效果用过哪些

45、Canvas 和 svg

46、行内元素和块级元素

47、对于语义化的理解

48、iframe 的缺点

优点：

1. iframe 能够原封不动的把嵌入的网页展现出来。

2. 如果有多个网页引用 iframe，那么你只需要修改 iframe 的内容，就可以实现调用的每一个页面内容的更改，方便快捷。

3. 网页如果为了统一风格，头部和版本都是一样的，就可以写成一个页面，用 iframe 来嵌套，可以增加代码的可重用。

4. 如果遇到加载缓慢的第三方内容如图标和广告，这些问题可以由 iframe 来解决。

5. 重载页面时不需要重载整个页面，只需要重载页面中的一个框架页(减少了数据的传输，增加了网页下载速度)

6. 方便制作导航栏

缺点：

1. 会产生很多页面，不容易管理

2. 不容易打印

3. 浏览器的后退按钮无效

4. 代码复杂，无法被一些搜索引擎索引到，这一点很关键，现在的搜索引擎爬虫还不能很好的处理 iframe 中的内容，所以使用
iframe 会不利于搜索引擎优化。

5. 多数小型的移动设备（PDA 手机）无法完全显示框架，设备兼容性差

6. 多框架的页面会增加服务器的 http 请求，对于大型网站是不可取的。

49、为什么要初始化 css

因为每个浏览器初始值是不一样的，要我们重新定义

50、css 定义的权重

51、 前端页面三层结构

最准确的网页设计思路是把网页分成三个层次，即：结构层、表示层、行为层。

网页的结构层（structural layer）由 HTML 或 XHTML 之类的标记语言负责创建。标签，也就是那些出现在尖括号里的单词，对网页内容的语义含义做出了描述，但这些标签不包含任何关于如何显示有关内容的信息。例如，P 标签表达了这样一种语义：“这是一个文本段。”

网页的表示层（presentation layer） 由 CSS 负责创建。 CSS 对“如何显示有关内容”的问题做出了回答。

网页的行为层（behavior layer）负责回答“内容应该如何对事件做出反应”这一问题。这是 Javascript 语言和 DOM 主宰的领域。

52、css 的基本语句构成机构

53、html 中的 img 标签设置 display：none。在打开页面的时候的 img 标签会不会被加载进来。

54、只有 css 文件没有 DOM 结构。css 文件会是什么效果。

55、如何把页面中的一个元素隐藏。只要不显示出来就可以。不用考虑会不会影响其他的页面元素。

56、如何实现页面的左侧区域固定右侧内容随页面大小动态改变。

57、列举 css 里曾用过的伪类和伪元素

伪类：用于向某些选择器添加特殊的效果

伪元素：用于将特殊的效果添加到某些选择器

- :first-child 选择某个元素的第一个子元素；
- :last-child 选择某个元素的最后一个子元素；
- :nth-child()选择某个元素的一个或多个特定的子元素；
- :nth-last-child()选择某个元素的一个或多个特定的子元素，从这个元素的最后一个子元素开始算；
- :nth-of-type()选择指定的元素；
- :nth-last-of-type()选择指定的元素，从元素的最后一个开始计算；
- :first-of-type 选择一个上级元素下的第一个同类子元素；
- :last-of-type 选择一个上级元素的最后一个同类子元素；
- :only-child 选择的元素是它的父元素的唯一一个了元素；
- :only-of-type 选择一个元素是它的上级元素的唯一一个相同类型的子元素；
- :empty 选择的元素里面没有任何内容。

二、 Javascript

1、 手写 js 时间函数

http://www.blogjava.net/parable-myth/archive/2008/01/12/174761.html

2、 对象怎么继承

1.原型链

基本思想：利用原型让一个引用类型继承另外一个引用类型的属性和方法。

构造函数，原型，实例之间的关系：每个构造函数都有一个原型对象，原型对象包含一个指向构造函数的指针，而实例都包含一个指向原型对象的内部指针。

原型链实现继承例子：

?

```
function SuperType() {
this.property = true;
}
SuperType.prototype.getSuperValue = function() {
return this.property;
}
function subType() {
this.property = false;
}
//继承了 SuperType
SubType.prototype = new SuperType();
SubType.prototype.getSubValue = function (){
return this.property;
}
var instance = new SubType();
console.log(instance.getSuperValue());//true
```

2.借用构造函数

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

例子：

```
function SuperType() {
this.colors = ["red","blue","green"];
}
function SubType() {
SuperType.call(this);//继承了 SuperType
}
var instance1 = new SubType();
instance1.colors.push("black");
console.log(instance1.colors);//"red","blue","green","black"
var instance2 = new SubType();
console.log(instance2.colors);//"red","blue","green"
```

3.组合继承

基本思想：将原型链和借用构造函数的技术组合在一块，从而发挥两者之长的一种继承模式。

例子：

```
function SuperType(name) {
this.name = name;
this.colors = ["red","blue","green"];
}
SuperType.prototype.sayName = function() {
console.log(this.name);
}
function SubType(name, age) {
SuperType.call(this,name);//继承属性
this.age = age;
}
//继承方法
SubType.prototype = new SuperType();
```

```
Subtype.prototype.constructor = Subtype;
Subtype.prototype.sayAge = function() {
console.log(this.age);
}
var instance1 = new SubType("EvanChen",18);
instance1.colors.push("black");
consol.log(instance1.colors);//"red","blue","green","black"
instance1.sayName();//"EvanChen"
instance1.sayAge();//18
var instance2 = new SubType("EvanChen666",20);
console.log(instance2.colors);//"red","blue","green"
instance2.sayName();//"EvanChen666"
instance2.sayAge();//20
```

4.原型式继承

基本想法：借助原型可以基于已有的对象创建新对象，同时还不必须因此创建自定义的类型。

原型式继承的思想可用以下函数来说明：

```
function object(o) {
function F(){}
F.prototype = o;
return new F();
}
```

例子：

```
var person = {
name:"EvanChen",
friends:["Shelby","Court","Van"];
};
var anotherPerson = object(person);
anotherPerson.name = "Greg";
anotherPerson.friends.push("Rob");
```

```
var yetAnotherPerson = object(person);
yetAnotherPerson.name = "Linda";
yetAnotherPerson.friends.push("Barbie");
console.log(person.friends);//"Shelby","Court","Van","Rob","Barbie"
```

ECMAScript5 通过新增 Object.create()方法规范化了原型式继承，这个方法接收两个参数：一个用作新对象原型的对象和一个作为新对象定义额外属性的对象。

```
var person = {
name:"EvanChen",
friends:["Shelby","Court","Van"];
};
var anotherPerson = Object.create(person);
anotherPerson.name = "Greg";
anotherPerson.friends.push("Rob");
var yetAnotherPerson = Object.create(person);
yetAnotherPerson.name = "Linda";
yetAnotherPerson.friends.push("Barbie");
console.log(person.friends);//"Shelby","Court","Van","Rob","Barbie"
```

5.寄生式继承

基本思想：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，最后再像真正是它做了所有工作一样返回对象。

例子：

```
function createAnother(original) {
var clone = object(original);
clone.sayHi = function () {
alert("hi");
};
return clone;
}
var person = {
name:"EvanChen",
friends:["Shelby","Court","Van"];
```

```
};
var anotherPerson = createAnother(person);
anotherPerson.sayHi();///"hi"
```

6.寄生组合式继承

基本思想：通过借用函数来继承属性，通过原型链的混成形式来继承方法

其基本模型如下所示：

```
function inheritProperty(subType, superType) {
var prototype = object(superType.prototype);//创建对象
prototype.constructor = subType;//增强对象
subType.prototype = prototype;//指定对象
}
```

例子：

```
function SuperType(name){
this.name = name;
this.colors = ["red","blue","green"];
}
SuperType.prototype.sayName = function (){
alert(this.name);
};
function SubType(name,age){
SuperType.call(this,name);
this.age = age;
}
inheritProperty(SubType,SuperType);
SubType.prototype.sayAge = function() {
alert(this.age);
}
```

3、 模拟百度搜索功能

基于 jquery 实现百度搜索功能

http://www.jb51.net/article/28075.htm

6、闭包

7、Window 对象

http://www.w3school.com.cn/jsref/dom_obj_window.asp

8、 JSON 是什么格式

http://www.ruanyifeng.com/blog/2009/05/data_types_and_json.html

10、 Jquery

Js 的库

11、 获取兄弟元素有哪些方法

jQuery.prev()，返回上一个兄弟节点，不是所有的兄弟节点

jQuery.prevAll()，返回所有之前的兄弟节点

jQuery.next(),返回下一个兄弟节点，不是所有的兄弟节点

jQuery.nextAll()，返回所有之后的兄弟节点

jQuery.siblings(),返回兄弟姐妹节点，不分前后

jQuery.find(expr),跟 jQuery.filter(expr)完全不一样。jQuery.filter()是从初始的 jQuery 对象集合中筛选出一部分，而 jQuery.find()

的返回结果，不会有初始集合中的内容，比如$("p"),find("span"),是从<p>元素开始找,等同于$("p span")

12、 Find 和 child 的区别

.children(selector) 方法是返回匹配元素集合中每个元素的所有子元素（仅儿子辈）。参数可选，添加参数表示通过选择器进行过滤，对元素进行筛选。

.find(selector)方法是返回匹配元素集合中每个元素的后代。参数是必选的，可以为选择器、jquery 对象可元素来对元素进行筛选。

13、 怎么让 Ajax,异步请求变成同步请求

Ajax 请求默认的都是异步的 如果想同步 async 设置为 false 就可以(默认是 true)

```
var html = $.ajax({
  url: "some.PHP",
  async: false
}).responseText;
```

或者在全局设置 Ajax 属性

```
$.ajaxSetup({
  async: false
  });
```

再用 post，get 就是同步的了

13、Ajax 加载完成以后怎么删除一个节点

14、用了哪些代码管理工具

github 官网：https://github.com/

svn

14、 返回顶部功能

```
<script src="http://ajax.microsoft.com/ajax/jQuery/jquery-1.7.2.min.js"></script>
<script>
$(function(){
        //当滚动条的位置处于距顶部 100 像素以下时，跳转链接出现，否则消失
        $(function () {
            $(window).scroll(function(){
                if ($(window).scrollTop()>100){
                    $("#back-to-top").fadeIn(1500);
                }
                else
                {
                    $("#back-to-top").fadeOut(1500);
                }
```

```
    });

    //当点击跳转链接后，回到页面顶部位置
    $("#back-to-top").click(function(){
        //$('body,html').animate({scrollTop:0},1000);
if ($('html').scrollTop()) {
        $('html').animate({ scrollTop: 0 }, 1000);
        return false;
```

20、 http 和 https 的区别

https://www.zhihu.com/question/19577317

22、问了常用插件轮播图

23、日期选择 datapicker

http://blog.csdn.net/cuihaiyang/article/details/6218928/

24、js 兼容问题

25、异步加载和同步加载

26、哪些地方用到了闭包，干什么用的，遇到内存泄漏没，怎么处理的

要解决循环引用带来的内存泄露问题，我们只需要把循环引用中的变量设为 null 即可。将变量设置为 null 意味着切断变量与它此前引用的值之间的联系。当垃圾收集器下次运行时，就会删除这些值并回收他们占用的内存。

27、数组 添加修改

28、绑定事件的方法

29、事件冒泡如何阻止

30、 原生 JS 获取元素的方法

31、 原生 JS 获取元素的方法数组去重

32、 异步 ajax 优缺点

33、 json 理解

34、js 作用域是怎样的

35、闭包解释 特性 对页面影响

36、js 添加 删除 替换 插入 节点

37、js 本地对象 内置对象 宿主对象

38、 load 和 ready 的区别

39、 == 和=== 的区别

40、js 同源策略

41、判断对象是否属于某个类

javascript 中检测对象的类型的运算符有：typeof、constructor、instanceof。

typeof：typeof 是一个一元运算符，返回结果是一个说明运算数类型的字符串。如："number"，"string"，"boolean"，"object"，"function"，"undefined"（可用于判断变量是否存在）。但 typeof 的能力有限，其对于 Date、RegExp、Array 类型返回的都是"object"。所以它只在区别对象和原始类型的时候才有用。要区一种对象类型和另一种对象类型，必须使用其他的方法。

```
> typeof new Date();
< "object"
> typeof new RegExp();
< "object"
> typeof {};
< "object"
> typeof [];
< "object"
```

instanceof 运算符：instanceof 运算符要求其左边的运算数是一个对象，右边的运算数是对象类的名字或构造函数。如果 object 是 class 或构造函数的实例，则 instanceof 运算符返回 true。如果 object 不是指定类或函数的实例，或者 object 为 null，则返回 false。instanceof 方法可以判断变量是否是数组类型，但是只限同一全局环境之内，在一个页面有多个 iframe 的情况下，instanceof 失效。

```
true
```

constructor 属性：JavaScript 中，每个对象都有一个 constructor 属性，它引用了初始化该对象的构造函数，常用于判断未知对象的类型。如给定一个求知的值 通过 typeof 运算符来判断它是原始的值还是对象。如果是对象，就可以使用 constructor 属性来判断其类型。

```
> var arr=[];
  arr.constructor();
< []
> var arr=new Function();
  arr.constructor();
< function anonymous() {

  }
> var arr=new RegExp();
  arr.constructor();
< /(?:)/
> var arr=new Date();
  arr.constructor();
< "Tue Jan 05 2016 17:49:23 GMT+0800 (中国标准时间)"
```

Object.prototype.toString.call()：该方法是目前为止发现的判断一个对象类型的最好的办法。

```
> var arr=[];
  Object.prototype.toString.call(arr);
< "[object Array]"
> var arr={};
  Object.prototype.toString.call(arr);
< "[object Object]"
> var arr=new Date();
  Object.prototype.toString.call(arr);
< "[object Date]"
> var arr=new Function();
  Object.prototype.toString.call(arr);
< "[object Function]"
> var arr=new RegExp();
  Object.prototype.toString.call(arr);
< "[object RegExp]"
```

42、说说对 function 的认识

我们都知道 function 是用来声明函数的操作符，javascript 有个特别的 Function 对象,俗称函数对象， 所有通过 funtion 声明出的对象都是 Function 对象，包括 Object，Array，Boolean，String 等几个基本类型对象，这几个也是 js 中最主要的构造函数，都是自己专属的属性和方法

所有函数在 function 的时候都会构建出 prototype 属性，指向 prototype 对象，这个 prototype 对象主要起到传宗接代的作用。它包括一个私有的[[proto]]指针，在 firefox，webkit 里公开为__proto__，为什么叫指针，因为它不是自身的属性，这个主要起到认祖归宗的作用，它永远指向其构造者的 prototype。

字面理解上就是构建者，它指向其构建者，也就是创建了自己的东东，不管是什么。所以理论上任何东西都是有 constructor 的。

43、说说冒泡排序和系统方法（sort）的实现

44、请写出 jquery 里 bind()和 gelegate()有什么区别

.bind()

```
$('a').bind('click', function() { alert("That tickles!") });
```

这是最简单的绑定方法了。JQuery 扫描文档找出所有的$(‘a’)元素，并把 alert 函数绑定到每个元素的 click 事件上。

.live()

```
$('a').live('click', function() { alert("That tickles!") });
```

JQuery 把 alert 函数绑定到$(document)元素上，并使用’click’和’a’作为参数。任何时候只要有事件冒泡到 document 节点上，它就查看该事件是否是一个 click 事件，以及该事件的目标元素与’a’这一 CSS 选择器是否匹配，如果都是的话，则执行函数。

live 方法还可以被绑定到具体的元素(或 context)而不是 document 上，像这样：

```
$('a', $('#container')[0]).live(...);
```

.delegate()

```
$('#container').delegate('a', 'click', function() { alert("That tickles!")
});
```

JQuery 扫描文档查找$(‘#container’)，并使用 click 事件和’a’这一 CSS 选择器作为参数把 alert 函数绑定到$(‘#container’)上。任何时候只要有事件冒泡到$(‘#container’)上，它就查看该事件是否是 click 事件，以及该事件的目标元素是否与 CCS 选择器相匹配。如果两种检查的结果都为真的话，它就执行函数。

可以注意到，这一过程与.live()类似，但是其把处理程序绑定到具体的元素而非 document 这一根上。精明的 JS’er 们可能会做出这样的结论，即$('a').live() == $(document).delegate('a')，是这样吗?嗯，不，不完全是。

45、如何用 html 标签来区分 IE 浏览器版本

以下是一个完整示例：

```
<script>
/*@cc_on
   @if (@_jscript_version >= 10)
      alert("你正在使用 IE10 浏览器！");
   @elif (@_jscript_version == 9)
      alert("你正在使用 IE9 浏览器！");
   @elif (@_jscript_version == 5.8)
      alert("你正在使用 IE8 浏览器！");
   @elif (@_jscript_version == 5.7 && window.XMLHttpRequest)
      alert("你正在使用 IE7 浏览器！");
      @elif (@_jscript_version == 5.6 || (@_jscript_version == 5.7
&& !window.XMLHttpRequest))
      alert("你正在使用 IE6 浏览器！");
   @elif (@_jscript_version == 5.5)
      alert("你正在使用 IE5.5 浏览器！");
   @else
      alert("你正在使用 IE5 或更远古的 IE 浏览器！");
   @end
@*/
</script>
```

46、写一段 javascript 判断页面是否有滚动条

```
function hasScrollbar() {
    return document.body.scrollHeight > (window.innerHeight ||
document.documentElement.clientHeight);
}
```

47、写一段 javascript 脚本,让一个图片跟随鼠标移动

48、写一段正则表达式,判断某个字符串里是否全是英文

正则：^[A-Za-z]+$

49、写一段 JS,让某个 div 每 5 秒换一次背景色

定时器

50、用 jquery,怎么找出一个 div 元素里所有的兄弟元素 a 标签

$('div a')

51、jsonp 跨域的原理

动态创建 script,通过 script 的 src 进行跨域

52、异步请求会出现什么问题

53、localstorage 有什么安全性问题

54、有没有做过屏幕翻转页面转换的

55、如果后台接口给的有问题，你有没有能力给他找出来

56、浅拷贝，深拷贝

可以使用 json.stringfy() 然后 json.parse()

57、因为我说了 sessionStorage 和 localStorage,他紧接着会问，他们两者的区别，在哪种情况下会使用，以及你在项目中使用过没有？

58、口述原生轮播图的原理

我当时说的是首尾增加图片的方式，然后他又说：使用增加图片的方式，无疑会增加服务器的负担，有没有不增加图片的方式去实现轮播图

59、问我移动端怎么适配

Rem 百分比，响应式

60、问我作用域和递归，奇葩的是问我用 jQuery 怎么写闭包。

Ajax

1、手写 js 实现异步请求 ？

```
<script type="text/javascript">
  window.onload = function(){
    var xmlrequest;
    //完成 XMLHttpRequest 初始化
    function creatXMLHttpRequest(){
      if(window.XMLHttpRequest){ //判定兼容性
        xmlrequest= new XMLHttpRequest(); //Dom2 浏览器
      }
      else if(window.ActiveXObject){  //IE 浏览器
        try{
          xmlrequest= new ActiveXObject("msxml2.XMLHTTP");
        }catch(e){
          try{
            xmlrequest= new ActiveXObject("Microsoft.XMLHTTP");
          }catch(e){}
        }
      }
    }
    function change(url){//接受请求响应的 URL
      creatXMLHttpRequest();
      xmlrequest.open("POET",url,true);//连接服务器

xmlrequest.setRequestHeader("content-Type","applicaion/x-www-form-url
encoded");
      //设置请求头编码方式
      xmlrequest.onreadystatechange = processResponse;
      xmlrequest.send(null);  //发送请求
    }
    //定义处理响应的回调函数
    function processResponse(){
      if(xmlrequest.status ==4){
        if(xmlrequest.status ==200){  //请求成功
          var headers = xmlrequest.getAllResponseHeaders();
          alert("响应头类型"+typeof headers+"\n"+headers);
```

```
                document.getElementById('odiv').innerHTML  =  headers;
    //请求成功后要做的事情
                }
                else{
                  window.alert("请求页面有异常");
                }
              }
            }
          }
        </script>
```

2、ajax 中 form 中的 content-type 代表的是？

http://blog.csdn.net/qlcql/article/details/51206972

3、ajax headers 中 accept 代表的是

http://blog.csdn.net/albertfly/article/details/52330919

4、 get 和 post 的区别

1. get 是从服务器上获取数据，post 是向服务器传送数据。

2. get 是把参数数据队列加到提交表单的 ACTION 属性所指的 URL 中，值和表单内各个字段一一对应，在 URL 中可以看到。post 是通过 HTTP post 机制，将表单内各个字段与其内容放置在 HTML HEADER 内一起传送到 ACTION 属性所指的 URL 地址。用户看不到这个过程。

3. 对于 get 方式，服务器端用 Request.QueryString 获取变量的值，对于 post 方式，服务器端用 Request.Form 获取提交的数据。

4. get 传送的数据量较小，不能大于 2KB。post 传送的数据量较大，一般被默认为不受限制。但理论上，IIS4 中最大量为 80KB，IIS5 中为 100KB。

5. get 安全性非常低，post 安全性较高。但是执行效率却比 Post 方法好。

建议：

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；

5、ajax 发送步骤

要完整实现一个 AJAX 异步调用和局部刷新,通常需要以下几个步骤:

(1)创建 XMLHttpRequest 对象,也就是创建一个异步调用对象.

(2)创建一个新的 HTTP 请求,并指定该 HTTP 请求的方法、URL 及验证信息.

(3)设置响应 HTTP 请求状态变化的函数.

(4)发送 HTTP 请求.

(5)获取异步调用返回的数据.

(6)使用 JavaScript 和 DOM 实现局部刷新.
http://www.cnblogs.com/kennyliu/p/3876729.html
具体的流程

5、 跨域问题

1、document.domain+iframe 的设置

2、动态创建 script

3、利用 iframe 和 location.hash

4、window.name 实现的跨域数据传输

5、使用 HTML5 postMessage

6、利用 flash

http://www.cnblogs.com/luoguixin/p/6124297.html

6、 ajax 发送大量数据怎么优化。

语法：

JSON.stringify(value [, replacer] [, space])

作用：这个函数的作用主要是为了系列化对象的。

可能有些人对系列化这个词过敏，我的理解很简单。就是说把原来是对象的类型转换成字符串类型（或者更确切的说是 json 类型的）。就这么简单。打个比方说，你有一个类，那么你可以通过这个方法转换成相应的 json 类型的。很简单吧。

接着看。

value：是必须要的字段。就是你输入的对象，比如数组啊，类啊等等。

replacer：这个是可选的。它又分为 2 种方式，一种是方法，第二种是数组。

var obj={

```
 webName:"脚本之家",
 url:"jb51.net",
 age:"2"
}
var str=JSON.stringify(obj)
console.log(str);
```

7、ajax 几种状态。

8、 AJAX 请求数据步骤。

9、 异步加载和同步加载。

10、 ajax，如果数据获取失败返回 500 怎么不让用户看到

用 error 接收 500，跳转到当前页

11、 原生的 ajax 的封装

12、 http 的有几次握手

HTTP 三次握手

HTTP（HyperText Transfer Protocol)超文本传输协议是互联网上应用最为广泛的一种网络协议。由于信息是明文传输，所以被认为是不安全的。而关于 HTTP 的三次握手，其实就是使用三次 TCP 握手确认建立一个 HTTP 连接。

如下图所示，SYN（synchronous）是 TCP/IP 建立连接时使用的握手信号、Sequence number（序列号）、Acknowledge number（确认号码），三个箭头指向就代表三次握手，完成三次握手，客户端与服务器开始传送数据。

第一次握手：客户端发送 syn 包(syn=j)到服务器，并进入 SYN_SEND 状态，等待服务器确认；

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 状态，完成三次握手。

13、 说说 ajax 的三级联动

14、 怎么判断一个页面的 ajax 请求全部成功了

添加一个计数，ajax 请求完成后计数+1，并在每次请求完成后都判断计数是否为所有 ajax 的数量

一种是使用 jquery 自带的 when

四、vue 问题

1、问：Vue 的生命周期，钩子函数？

答：beforecreated ：可以在这加个 loading 事件
created ：在这结束 loading，还做一些初始化，实现函数自执行
mounted ： 在这发起后端请求，拿回数据，配合路由钩子做一些事情
beforeDestroy： 你确认删除 XX 吗？ destroyed ：当前组件已被删除，清空相关内容

生命周期钩子函数

beforeCreated()：el,data 并未初始化时触发

created()：data 初始化完成，el 没有初始化时，触发

beforeMount()：el,data 初始化完成触发

mounted()：完成挂载，页面成功渲染触发

beforeUpdate()：视图层的数据改变前触发，即 data 数据变动，视图层还未改变时

Updated()：视图层数据改变后触发

beforeDestroy()：页面销毁前触发

destroyed()：页面销毁后触发

2、父子传值?

父传子：是在子组件标签内通过 v-bind 属性绑定父组件内的值，在子组件中，通过 props 属性获取子组件的值,props 的值是一个数组，里面的字符串元素就是绑定的属性名，在子组件内可以直接通过插值表达式获得父组件传过来的值

子传父：子组件内通过触发一个事件，通过函数内 this.$emit(‘传给父组件内的事件名’,值)的方法,在父组件的子组件标签内设置@传过来的事件名=“父组件内的方法”,父组件的方法函数中，形参会接收从子组件传过来的参数

兄弟组件：通过新建一个中间的 Vue 实例 newVue 对象，要传值的组件内注册一个事件，通过 newVue.$emit(‘传递的事件名’,值)方法，然后再需要接收兄弟组件值得组件里，在 mounted()钩子函数内设置 newVue.$on(‘传递过来的事件名’, data=>this.xx=data),获取值

3、vuex 设置全局的，有没有什么安全问题

有安全问题，用户这样可以在控制台中获取到 Vuex 中的数据(vue devTools 插件可以直接查看 vuex 的数据)，那么在控制台就可以通过 this.$vm.$store 对象来修改 store 里的值，获取一些权限

4、vue 中用什么发送异步请求

基于 Promise 语法的 axios.js 或者 vue-resourse.js

5、说一下 MVVM 思想

本质上是 MVC 的改进，

M-->Model 层，数据层，VUE 实例里的 data 对象

V-->View 层，就是视图层，就是页面

VM-->ViewModel 层

Model 层的数据改变，VM 层会感知到变化，是 View 层做出相应改变

View 层发生变换，VM 同样也会感知到，通知 Model 层发生变化

这也是双向数据绑定的概念

6、Vue 中 get，post，jsonp

通过 axios.js 可以使用 get，post 请求，没有 jsonp 请求

Vue-resourse.js 有 jsonp 请求

7、Vue 了解过吗？使用 vue 做过哪些项目？

肯定答了解过，项目需要自己找

8、vuex 设置全局的，有没有什么安全问题

有安全问题，用户这样可以在控制台中获取到 Vuex 中的数据(vue devTools 插件可以直接查看 vuex 的数据)，那么在控制台就可以通过 this.$vm.$store 对象来修改 store 里的值，获取一些权限

9、Vue 中 get，post，jsonp

通过 axios.js 可以使用 get, post 请求，没有 jsonp 请求

Vue-resourse.js 有 jsonp 请求

10、 VueX 了解过吗？

答：状态管理模式、集中式存储管理 一听就很高大上，蛮吓人的。在我看来 vuex 就是把需要共享的变量全部存储在一个对象里面，然后将这个对象放在顶层组件中供其他组件使用。这么说吧，将 vue 想作是一个 js 文件、组件是函数，那么 vuex 就是一个全局变量，只是这个“全局变量”包含了一些特定的规则而已。

在 vue 的组件化开发中，经常会遇到需要将当前组件的状态传递给其他组件。父子组件通信时，我们通常会采用 props + emit 这种方式。但当通信双方不是父子组件甚至压根不存在相关联系，或者一个状态需要共享给多个组件时，就会非常麻烦，数据也会相当难维护，这对我们开发来讲就很不友好。vuex 这个时候就很实用，不过在使用 vuex 之后也带来了更多的概念和框架，需慎重！

注意：并不是 Vuex 一定是好的，如果页面不复杂，可以不需要用 Vuex，简单的组件传值就可以了

11、 ajax 和 axios 的区别以及 axios 的好处

JQuery 整个项目太大，单纯使用 ajax 却要引入整个 JQuery 非常的不合理，基于原生 XHR 开发，XHR 本身架构不清晰。ajax 处理回调麻烦。

Axios 只是一个单独的 js 文件，文件小，模块化开发思想

Axios 基于 Promise 语法，提供了一些并发请求的接口，可以同时处理大量异步操作

Axios 自动转换 JSON 数据

Axios 在客户端支持防止 CSRF/XSRF

12、 v-if 和 v-show 的区别，vue 如何绑定数据

v-if 值是 false 是删除这个元素

v-show 值是 false 是设置 display:none

一般如果是异步数据进行渲染的时候用 v-if

操作大量 dom 的时候用 v-show

数据绑定： {{ }}，v-text, v-html, v-bind,v-cloak,v-once,v-for,v-on....

13、 路由传值

答：①通过路由配置中的 name 属性，在组件中{{$route.name}}

②通过路由带参数进行传值

两个组件 A 和 B,A 组件通过 query 把 orderId 传递给 B 组件（触发事件可以是点击事件、钩子函数等）

this.$router.push({ path: '/conponentsB', query: { orderId: 123 } }) // 跳转到 B

在 B 组件中获取 A 组件传递过来的参数

this.$route.query.orderId

③通过<router-link>标签中的 to 传参

<router-link :to="{name:'hi1',params:{username:xxx}}">跳转到 B 页面</router-link> //A 页面

{path:'/xxxxx',name:'hi1',component:组件名} //路由配置

$route.params.uesrname //B 页面中接收 A 页面传过来的值

④利用 url 传递

```
{
  path:'/params/:newsId/:newsTitle',
  component:Params
}
```

//路由配置 我们需要传递参数是新闻 ID（newsId）和新闻标题（newsTitle）.所以我们在路由配置文件里制定了这两个值

在跳转后的页面使用$route.params.newsId 和 $route.params.newsTitle 来获取通过 url 传递过来的值

14、 按需加载

就是模块化开的思想，以前我们要使用 JQ 中的 AJAX,需要引入整个 JQ 文件，按需加载就是把特定的功能封装成一个模块，在哪个模块中需要使用就导入哪个模块

语法： import ... from ‘路径’

15、 vue 项目最后打包是什么指令？Vue 打包完成的 dist 文件夹都有是么？

npm run build 这个是在 webpack 中配置好的，你也可以自己定义，package.json 中可以改名字

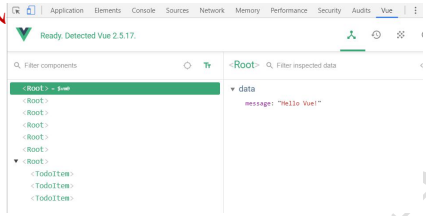
dist 文件夹是默认的，可以自己修改路径和名称，在项目 config 文件下的 index.js 的 build：{ } 这个属性里

16、 数据展示 v-bind v-model

v-bind 是绑定属性，比如绑定 src 和 class 属性
V-model 是双向数据绑定，只能出现在 input　textarea　select 标签中

17、 vue 代码如何调试

①通过 vue-devtools 这个插件



②移动端调试代码：
和电脑一样在同一个局域网下，输入地址就能看到效果了，iphone 可以用 safari 来检查元素；android 可以用 chrome 远程调试

18、 什么是单页面应用程序？双页面应用程序？

Vue 就是典型的单页面应用程序，系统只加载一次资源，即 CSS，JS 只加载一次，在全局中加载后，其他组件都可以使用。后面的操作都是通过路由，异步请求来实现的

优点：用户体验好，内容改变不需要重新加载整个页面，
减轻服务器压力
一套代码，适用于 PC，移动端

缺点：不利于 SEO
初始加载慢
导航不可以用，要通过路由实现

双（多）页面应用程序就是我们常见的 PC 端的页面

19、 如何创建自定义指令

全局： Vue.directive('指令名称', { 钩子函数(el, binding){} })
局部： 组件中 directives{ '指令名称' :{ 钩子函数(el, binding){} } }

20、 封装过 vue 中的 http 吗，二次封装呢（promise),以下就是 2 次封装

具体实现思路 --> 封装之前需要用 npm 安装并引入 axios,使用一个单独的 js 模块作为接口请输出对象，然后 export dafult 这个对象

1. 首先我们需要在 Vue 实例的原型 prototype 中扩展一个$http 的方法取代 axios=>Vue.prototype.$http=axios 这样子就可以在组件中通过 this.$http 方法引用 axios 的 get 等内置 API

2. 将方法写在一个对象中方便调用

```
const vm = new Vue();
const xhr = {
    httpRequest(options = {

    }) {
        vm.$http.get(options.url, { params: options.data }).then((res) => {
            if (res.data.code == 0) {
                //xhr.showmsg(res.data.msg);
            } else if (res.data.code == 401) {
                xhr.showmsg('登录失效，请重新登录');
                Vstorage.remove('loginData');
                return;
            } else {
                xhr.showmsg(res.data.msg);
            }
            options.success(res.data);
        }).catch((error) => {
            xhr.showmsg(error);
        });
    },
}
```

3. 类似这个就将该请求方法写在了 xhr 这个对象中，

```
export default {
    install(Vue) {
        Vue.prototype.$xhr = {
            xhr: xhr,
            apiUrl: apiUrl
        }
    }
}
export {
    xhr,
    apiUrl
}
```

4. 通过=>方法全局注册 Vue 原型方法$xhr 下的 xhr 对象，export 之后可以在组件中通过下面方法实现调用，类似于 jq 的 ajax 的请求以及回调格式

this.$xhr.xhr.httpRequest({//xhr 对象中封装的 ajaxrequest 方法

url:'url 地址',

data:{

//传参

},

success:=>(data){//此处 data 相当于封装的 ajaxrequest 方法中的 res.data

//回调处理逻辑

}

})

21、 Vue 中的 jsonp 怎么实现

有点复杂，可以参考这个网址 https://segmentfault.com/a/1190000008036838

1.生成 script 标签

2.注册 callback 函数

3.返回一个 Promise

4.移除 script 标签

5.移除 callback 函数

如果不会自己动手写，就说用 vue-resourse 或者 axios(有 jsonp 请求,需要手动安装一下，引用就行,npm i jsonp --save-dev 然后在页面中引用 import jsonp from 'jsonp'，使用方式也是非常简单的)

22、 路由怎么切换

①导入路由 import Router from 'vue-router'

②使用/注册：Vue.use(Router)

③配置路由

④把路由注入到根实例中(main.js 里)

⑤在 APP.vue 组件中定义<router-link to=”要跳转的路径”>A 页面</router-link>

<router-view></router-view> //对应的组件内容渲染到 router-view 中

五、框架

1. es6 的一些新特性？

http://www.cnblogs.com/Wayou/p/es6_new_features.html

2、bootstrap 怎么轻量化？

3、什么是 angular？

AngularJS 的官方文档是这样介绍它的。

1、完全使用 JavaScript 编写的客户端技术。同其他历史悠久的 Web 技术（HTML、CSS 和 JavaScript）配合使用，使 Web 应用开发比以往更简单、更快捷。

2、AngularJS 主要用于构建单页面 Web 应用。它通过增加开发人员和常见 Web 应用开发任务之间的抽象级别，使构建交互式的现代 Web 应用变得更加简单。

3、AngularJS 的开发团队将其描述为一种构建动态 Web 应用的结构化框架。

4、AngularJS 使开发 Web 应用变得非常简单，同时也降低了构建复杂应用的难度。它提供了开发者在现代 Web 应用中经常要用到的一系列高级功能，例如：

解耦应用逻辑、数据模型和视图；

Ajax 服务；

依赖注入；

浏览历史（使书签和前进、后退按钮能够像在普通 Web 应用中一样工作）；

4、上传一个 50MB 的文件怎么实现分段传输？

5、用过 angular 么，我们需要懂底层原理的？

6、用了哪些代码管理工具？

7、gitHub 怎么管理代码？

github 官网：https://github.com/

svn

8、各种模板的使用？

10、vue 和 angular 区别，为什么要用 vue vue 好处 为什么移动端青睐用 vue？

vue 仅仅是 mvvm 中的 view 层，只是一个如 jquery 般的工具库，而不是框架，而 angular 而是 mvvm 框架。

vue 的双向邦定是基于 ES5 中的 getter/setter 来实现的，而 angular 而是由自己实现一套模版编译规则，需要进行所谓的“脏”检查，vue 则不需要。因此，vue 在性能上更高效，但是代价是对于 ie9 以下的浏览器无法支持。

vue 需要提供一个 el 对象进行实例化，后续的所有作用范围也是在 el 对象之下，而 angular 而是整个 html 页面。一个页面，可以有多个 vue 实例，而 angular 好像不是这么玩的。

vue 真的很容易上手，学习成本相对低，不过可以参考的资料不是很丰富，官方文档比较简单，缺少全面的使用案例。高级的用法，需要自己去研究源码，至少目前是这样。

12、轮播图如何实现？

15、生命周期函数的理解？(react)

16、钩子函数的理解？

17、Node.js 中 Stream 如何理解?

在服务器端，文件体积一般很庞大，用这种方式效率低下，导致线程阻塞，

甚至会因为内存不足而崩溃。

这里要引用 Stream 流的概念。

nodejs 中 Stream 是 EventEmitter 的实现，你可以理解为在程序后台打开了一个文件(不占用主线程)，

20、Vue、Angular、React 的区别在哪里？哪些场景会考虑不使用 Angular，而是用 Vue？

一、数据流

数据绑定

Angular 使用双向绑定即：界面的操作能实时反映到数据，数据的变更能实时展现到界面。

实现原理：

$scope 变量中使用脏值检查来实现。像 ember.js 是基于 setter，getter 的观测机制，$scope.$watch 函数，监视一个变量的变化。函数有三参数，”要观察什么”，”在变化时要发生什么”,以及你要监视的是一个变量还是一个对象。

使用 ng-model 时，你可以使用双向数据绑定。

使用$scope.$watch（视图到模型）以及$scope.$apply（模型到视图），还有$scope.$digest 调用$scope.$watch 时只为它传递了一个参数，无论作用域中的什么东西发生了变化，这个函数都会被调用。在 ng-model 中，这个函数被用来检查模型和视图有没有同步，如果没有同步，它将会使用新值来更新模型数据。

双向绑定的三个重要方法：

?

```
1  $scope.$apply()
2
3  $scope.$digest()
4
5  $scope.$watch()
```

在 AngularJS 双向绑定中,有 2 个很重要的概念叫做 dirty check,digest loop,dirty check（脏检测）是用来检查绑定的 scope 中的对象的状态的，例如，在 js 里创建了一个对象，并且把这个对象绑定在 scope 下，这样这个对象就处于 digest loop 中，loop 通过遍历这些对象来发现他们是否改变，如果改变就会调用相应的处理方法来实现双向绑定

Vue 也支持双向绑定，默认为单向绑定，数据从父组件单向传给子组件。在大型应用中使用单向绑定让数据流易于理解。

脏检测的利弊

和 ember.js 等技术的 getter/setter 观测机制相比（优）：

getter/setter 当每次对 DOM 产生变更，它都要修改 DOM 树的结构，性能影响大，Angular 会把批量操作延时到一次更新，性能相对较好。

和 Vue 相比（劣）：

Vue.js 有更好的性能，并且非常非常容易优化，因为它不使用脏检查。Angular，当 watcher 越来越多时会变得越来越慢，因为作用域内的每一次变化，所有 watcher 都要重新计算。并且，如果一些 watcher 触发另一个更新，脏检查循环（digest cycle）可能要运行多次。Angular 用户常常要使用深奥的技术，以解决脏检查循环的问题。有时没有简单的办法来优化有大量 watcher 的作用域。Vue.js 则根本没有这个问题，因为它使用基于依赖追踪的观察系统并且异步列队更新，所有的数据变化都是独立地触发，除非它们之间有明确的依赖关系。唯一需要做的优化是在 v-for 上使用 track-by。

React-单向数据流

MVVM 流的 Angular 和 Vue，都是通过类似模板的语法，描述界面状态与数据的绑定关系，然后通过内部转换，把这个结构建立起来，当界面发生变化的时候，按照配置规则去更新相应的数据，然后，再根据配置好的规则去，从数据更新界面状态。

React 推崇的是函数式编程和单向数据流：给定原始界面（或数据），施加一个变化，就能推导出另外一个状态（界面或者数据的更新）。

React 和 Vue 都可以配合 Redux 来管理状态数据。

二、视图渲染

Angular1

AngularJS 的工作原理是:HTML 模板将会被浏览器解析到 DOM 中, DOM 结构成为 AngularJS 编译器的输入。AngularJS 将会遍历 DOM 模板, 来生成相应的 NG 指令,所有的指令都负责针对 view(即 HTML 中的 ng-model)来设置数据绑定。因此, NG 框架是在 DOM 加载完成之后, 才开始起作用的。

React

React 的渲染建立在 Virtual DOM 上——一种在内存中描述 DOM 树状态的数据结构。当状态发生变化时，React 重新渲染 Virtual DOM，比较计算之后给真实 DOM 打补丁。

Virtual DOM 提供了函数式的方法描述视图，它不使用数据观察机制，每次更新都会重新渲染整个应用，因此从定义上保证了视图与数据的同步。它也开辟了 JavaScript 同构应用的可能性。

在超大量数据的首屏渲染速度上，React 有一定优势，因为 Vue 的渲染机制启动时候要做的工作比较多，而且 React 支持服务端渲染。

React 的 Virtual DOM 也需要优化。复杂的应用里可以选择 1. 手动添加 shouldComponentUpdate 来避免不需要的 vdom re-render；2. Components 尽可能都用 pureRenderMixin，然后采用 Flux 结构 + Immutable.js。其实也不是那么简单的。相比之下，Vue 由于采用依赖追踪，默认就是优化状态：动了多少数据，就触发多少更新，不多也不少。

React 和 Angular 2 都有服务端渲染和原生渲染的功能。

Vue.js 不使用 Virtual DOM 而是使用真实 DOM 作为模板，数据绑定到真实节点。Vue.js 的应用环境必须提供 DOM。Vue.js 有时性能会比 React 好**，而且几乎不用手工优化。

三、性能与优化

性能方面，这几个主流框架都应该可以轻松应付大部分常见场景的性能需求，区别在于可优化性和优化对于开发体验的影响。Vue 的话需要加好 track-by 。React 需要 shouldComponentUpdate 或者全面 Immutable,Angular 2 需要手动指定 change detection strategy。从整体趋势上来说，浏览器和手机还会越变越快，框架本身的渲染性能在整个前端性能优化体系中，会渐渐淡化，更多的优化点还是在构建方式、缓存、图片加载、网络链路、HTTP/2 等方面。

四、模块化与组件化

Angular1 -> Angular2

Angular1 使用依赖注入来解决模块之间的依赖问题，模块几乎都依赖于注入容器以及其他相关功能。不是异步加载的，根据依赖列出第一次加载所需的所有依赖。

可以配合类似于 Require.js 来实现异步加载，懒加载（按需加载）则是借助于 ocLazyLoad 方式的解决方案，但是理想情况下应该是本地框架会更易懂。

Angular2 使用 ES6 的 module 来定义模块，也考虑了动态加载的需求。

Vue

Vue 中指令和组件分得更清晰。指令只封装 DOM 操作，而组件代表一个自给自足的独立单元 —— 有自己的视图和数据逻辑**。在 Angular1 中两者有不少相混的地方。
React
一个 React 应用就是构建在 React 组件之上的。
组件有两个核心概念：props,state。
一个组件就是通过这两个属性的值在 render 方法里面生成这个组件对应的 HTML 结构。
传统的 MVC 是将模板放在其他地方，比如 script 标签或者模板文件，再在 JS 中通过某种手段引用模板。按这种思路，想想多少次我们面对四处分散的模板片段不知所措？纠结模板引擎，纠结模板存放位置，纠结如何引用模板。
React 认为组件才是王道，而组件是和模板紧密关联的，组件模板和组件逻辑分离让问题复杂化了。所以就有了 JSX 这种语法，就是为了把 HTML 模板直接嵌入到 JS 代码里面，这样就做到了模板和组件关联，但是 JS 不支持这种包含 HTML 的语法，所以需要通过工具将 JSX 编译输出成 JS 代码才能使用（可以进行跨平台开发的依据，通过不同的解释器解释成不同平台上运行的代码，由此可以有 RN 和 React 开发桌面客户端）。
五、语法与代码风格
React,Vue,Angular2 都支持 ES6,Angular2 官方拥抱了 TypeScript 这种 JavaScript 风格。
React 以 JavaScript 为中心，Angular 2 依然保留以 HTML 为中心。Angular 2 将 “JS” 嵌入 HTML。React 将 “HTML” 嵌入 JS。Angular 2 沿用了 Angular 1 试图让 HTML 更强大的方式。
React 推荐的做法是 JSX + inline style，也就是把 HTML 和 CSS 全都整进 JavaScript 了。Vue 的默认 API 是以简单易上手为目标，但是进阶之后推荐的是使用 webpack + vue-loader 的单文件组件格式（template,script,style 写在一个 vue 文件里作为一个组件）

22、如何注册组件？

25、ES6 与 ES5 的区别？

25、对于 angularjs 你知道哪些指令？

25、原生开发和混合开发？

1、动画

动画有很多种，比如侧边栏菜单的滑入滑出、元素的响应动画、页面切换之间的过场等等，在 H5 之下的众多实现方法都没有办法达到纯原生的性能。一般这些的话有几种不同的选择：css3 动画、javascript 动画、原生动画。

css3 动画非常的消耗性能，如果某一个元素用到 css3 动画可能还看不出来，但大面积或过场使用 css3 动画会让 app 低端手机体验非常差。最好的选择一般是通过框架调用底层的动画，但不管怎么样等于在原来的代码上包上了一层，性能还是不可避免的受到影响。

比如在一个新页面的载入上，如果调用底层动画要考虑的问题有两个，一个是本身资源页面的渲染问题，另一个是远程数据的获取。即便是这些动画能够很快的响应，但大量的 css 页面会导致渲染卡顿，滑入时可能会有白屏/机器卡顿的现象。为了解决这些性能问题又必须要用到预加载或模拟动画。即便是这样，滑入滑出的动画在低端的安卓机器上还是有很多问题，如果获取服务端数据处理的方式不合适，卡顿白屏的现象会更严重。具体看下面的数据获取方式。

2、获取服务端数据

首先要接受的是，这里的数据获取都是在资源页面上异步完成的，因为只有这样才能让这些资源页面完成预加载或者渲染。但是异步拿到的数据在填入页面中时可能会涉及 DOM 操作，众所周知，DOM 操作非常消耗性能，如果页面小还好，页面稍大数据稍微复杂一点，频繁的 DOM 操作会导致明显的闪白。而且最重要的一点是，如果页面加载进来之后数据更新的速度太慢，也会让页面模板等待很长时间，对用户体验又不友好，总不能每次打开都像浏览器一样等待刷新是吧。

这个问题如果没有得到解决，H5APP 是很难承担大规模数据的页面，在它们之中频繁切换更是难上加难，那么肯定有人也会想到用 MVVM 的方式，其实我也写过一些基于 MVVM 的 H5APP，相对来说它们获取数据和更新数据的方式更敏捷更科学，但写的过程中又要注意很多 H5 独有的问题，这些问题在下面的页面切换里来讲。

3、页面切换

上面我们看到了几种不错的实现方式，比如预加载和模拟动画，甚至有批量的预加载，批量的截图模拟动画等等，虽然看起来很友好解决了不少问题，但事实上如果页面足够多就会引发另一个问题——页面的生存周期。

试想一下，如果引导页或者主页面缓存了 5 个子页面的资源，在跳转到响应的子页面时又会缓存这些子页面的下级页面资源，如此反复肯定会占据大量内存使 APP 的体验下降。那么怎么知道那些页面是需要的，最多缓存多少页面，什么时候结束哪些页面的生存周期呢？在我用过的很多 H5APP 的框架里都没有对这些问题有一个完美的解答，因此在页面较多内容较多的 APP 中可能会因这些资源分配的问题降低性能。

这时候我们回过头来再看看 MVVM 的数据加载问题，实际上不管哪个 MVVM 框架，写过的人都知道管理这种新型的前端代码最重要的问题是内存的问题，你既要保证代码写的足够优雅没有任何内存泄露问题，也要考虑到在页面生存周期结束时它们的控制器/页面资源是否得到释放，这对全局有没有什么影响，在多个请求时也要合理的分配资源，甚至是复用这些

父级页面传过来的缓存资源等等。较小的 APP 可能并不会有这些问题，如果你想用纯 H5 来开发大型 APP，这很可能会浪费你很多时间——而且结果还不会让你满意。

4、Android/iOS 的区别

很多人都说纯 H5APP 一次编写就能编译 Android/iOS 两种不同的 APP,大大降低了成本。实际上这个观点本身就是值得怀疑的，如果你写过这类 APP 就能明白我在说什么，它们既不省事，又存在很多 BUG，调试时尤其繁琐。举一个很简单的例子，Android 和 iOS 在返回上一页的处理方式上就有明显的区别，iOS 的顶部 bar 在全屏下怎样处理，Android 机器出现 smart bar 怎样处理页面的布局，调用底层硬件时怎样区分不同的场景等等，你需要写一个又一个机型和系统的判断,然后分别在 Android 和 iOS 下调试,最后你却发现这并没有卵用，累的要死却什么没学到，只有一堆不知道什么时候会过时的经验。

现在做 H5 混合 APP 开发的人很多，但是纯 H5 却很年轻，很多问题都没有很好的解决，这几个是我在做这些 APP 时考虑最多的问题。当然大家也不必担心，随着 ES6 的推行，硬件发展越来越快，纯 H5APP 未必没有一席之地。最后说一个很少人注意到的 H5 优势，大家大谈 H5APP 时都是快速开发、低成本、多平台等等，但我却觉得它和很多 APP 开发方式相比有一个不同之处——图文混合的排版。正是这些复杂多变的 CSS 样式消耗了性能，但是它带来了排版的多样性，能够细致到每一个字宽行高和风格的像素级处理，才是 H5 的优异之处。

六、Node.js

1、你用 node 实现过哪些东西

服务器，

2、以及 node 框架中你使用了哪些东西

3、因为面试的前端，后端他问的不多，都是需要你自己去扩散，把你懂得展示给他看。

4、对 node 的插件有什么了解以及 node 中路由有关的问题，对 express 的理解

七、项目

1、就项目而言,哪些让你影响深刻,为什么?

2、谈到学习能力,可否谈谈在工作中能深刻体现的例子?

3、你觉得你在上家公司离职的情形会在我们公司继续吗?

4、你觉得你加入我们的团队,有哪些优势?

5、公司有淘汰机制,无论试用阶段和正式上班都有,你能接受吗?你是怎么理解的?

6、你觉得进入公司后,你需要努力的方向和需要注意的有哪些?

7、模拟百度搜索功能?

8、三个项目处理了哪些兼容问题?

9、简单介绍一下做的项目?

10、打开项目链接，说一下你做的是哪些？

11、你们公司是怎么测试的？

12、平时会进行自我提升吗，会去哪些地方学，看了哪些书？

13、项目太少？

14、有做过性能优化吗，有思考过这方面的东西吗？

15、JSP 有用过吗？

16、你们项目所用的技术栈有那些？

17、你觉得你的 js 处于什么水平？

18、说说你的项目中经常用的那种页面的布局？

19、部门管理系统的部门管理页面，你是怎么布局的？

20、说说你的最大缺点？

21、说说你对加班看法？

22、说说你对薪资要求？

23、说说你的职业规划？

24、说说你朋友对你的评价？

25、如果你在这次考试中没有被录用，你怎么打算？

26、问我项目中都是用什么工具，我说 git 和 webpack，不会问你细节的

八、 前端代码工具

1. 问了我用什么代码管理工具，git 然后问了我要是代码冲突怎么办，工作中主要遇到哪些冲突

2. 常用的 git 命令？

http://www.ruanyifeng.com/blog/2015/12/git-cheat-sheet.html

3. webpack 与 gulp 有什么区别？要求能够回答深一点？

gulp

gulp 强调的是前端开发的工作流程，我们可以通过配置一系列的 task，定义 task 处理的事务（例如文件压缩合并、雪碧图、启动 server、版本控制等），然后定义执行顺序，来让 gulp 执行这些 task，从而构建项目的整个前端开发流程。

PS：简单说就一个 Task Runner

webpack

webpack 是一个前端模块化方案，更侧重模块打包，我们可以把开发中的所有资源（图片、js 文件、css 文件等）都看成模块，通过 loader（加载器）和 plugins（插件）对资源进行处理，打包成符合生产环境部署的前端资源。

PS：webpack is a module bundle

功能	gulp	webpack
文件合并与压缩（css）	使用 gulp-minify-css 模块 gulp.task('sass',function(){ gulp.src(cssFiles) .pipe(sass().on('error',sass.lo	样式合并一般用到 extract-text-webpack-plugin 插件， 压缩则使用 webpack.optimize.UglifyJsPlugin。

	gError)) .pipe(require('gulp-minify-css')()) .pipe(gulp.dest(distFolder)); });	
文件合并与压缩（js）	使用 gulp-uglify 和 gulp-concat 两个模块	js 合并在模块化开始就已经做， 压缩则使用 webpack.optimize.UglifyJsPlugin
sass/less 预编译	使用 gulp-sass/gulp-less 模块	sass-loader/less-loader 进行预处理

启动 server	使用 gulp-webserver 模块
	var webserver =require('gulp-webserver');
	gulp.task('webserver',function(){
	gulp.src('./')
	.pipe(webserver({
	host:'localhost',
	port:8080,
	livereload:true, //自动刷新
	directoryListing:{
	enable:true,
	path:'./'
	},
	}));
	});
		使用 webpack-dev-server 模块
		module.exports = {
		.....
		devServer: {
		contentBase: "build/",
		port:8080,
		inline: true //实时刷新
		}
		}

版本控制	使用 gulp-rev 和 gulp-rev-collector 两个模块	将生成文件加上 hash 值
		module.exports = {
		.....
		output: {
		.....
		filename: "[name].[hash:8].js"
		},
		plugins:[
		.....

		new ExtractTextPlugin(style.[hash].css")] }

相同功能

gulp 与 webpack 可以实现一些相同功能，如下（列举部分）：

两者区别

虽然都是前端自动化构建工具，但看他们的定位就知道不是对等的。

gulp 严格上讲，模块化不是他强调的东西，他旨在规范前端开发流程。

webpack 更是明显强调模块化开发，而那些文件压缩合并、预处理等功能，不过是他附带的功能。

总结

gulp 应该与 grunt 比较，而 webpack 应该与 browserify（网上太多资料就这么说，这么说是没有错，不过单单这样一句话并不能让人清晰明了）。

gulp 与 webpack 上是互补的,还是可替换的,取决于你项目的需求。如果只是个 vue 或 react 的单页应用，webpack 也就够用；如果 webpack 某些功能使用起来麻烦甚至没有（雪碧图就没有），那就可以结合 gulp 一起用

4. webpack 的配置知道吗？

.webpack 配置文件常用配置项介绍

1.webpack 有一个默认的配置文件 webpack.config.js，这个文件需要手动的创建，位于项目根目录中。可以为一个项目设置多个配置文件，已达到不同的配置文件完成不同的功能。怎么设置后面介绍。

2.webpack 的配置文件会暴露出一个对象，

3.常用配置项将要说明

entry:打包的入口文件，一个字符串或者一个对象

output:配置打包的结果，一个对象

fileName：定义输出文件名，一个字符串

path：定义输出文件路径，一个字符串

module:定义对模块的处理逻辑，一个对象

loaders：定义一系列的加载器，一个数组

```
[
  {
```

test:正则表达式，用于匹配到的文件

loader/loaders：字符串或者数组，处理匹配到的文件。如果只需要用到一个模块加载器则使用 loader：string，如果要使用多个模块加载器，则使用 loaders：array

include:字符串或者数组，指包含的文件夹

exclude:字符串或者数组，指排除的文件夹

}

]

resolve:影响对模块的解析，一个对象

extensions：自动补全识别后缀，是一个数组

plugins:定义插件，一个数组

4.entry 详细说明

（1）当 entry 是一个字符串时，这个字符串表示需要打包的模块的路径,如果只有一个要打包的模块，可以使用这种形式

（2）当 entry 是一个对象

a.是数组时,当需要将多个模块打包成一个模块，可以使用这个方式。如果这些模块之间不存在依赖，数组中值的顺序没有要求，如果存在依赖，则要将依赖性最高的模块放在最后面。

例如：entry:["./app/one.js",".app/two.js"]

b.是键值对形式的对象是，当需要分别打包成多个模块时，可以使用这种方式注:当 entry 是一个键值对形式的对象时，包名就是键名，output 的 filename 不能是一个固定的值，因为每个包的名字不能一样

5.output 详细说明

（1）output 是一个对象

（2）output.filename:指定输出文件名，一个字符串。当输出一个文件，output.filename 为一个确定的字符串

当输出多个文件，output.filename 不能为一个确定的字符串。为了让每个文件有一个唯一的名字，需要用到下面的变量

[name] is replaced by the name of the chunk.对应 entry 的键名

[hash] is replaced by the hash of the compilation.

[chunkhash] is replaced by the hash of the chunk.

如：

（3）output.path:指定输出文件的路径，相对路径，一个字符串

（4）output 中还有其他的一些值，不在这里说明,可以在 webpack 的官方网站中获得更多的详细信息

6.module.loaders 详细说明

（1）module 是一个对象，定义对模块的处理逻辑

（2）module.loaders 是一个数组，定义一系列加载器，这个数组中的每一项都是一个对象

（3）module.loaders:[

{

test:正则，用于匹配要处理的文件

loader/loaders: 字符串或者数组， 如果只需要用到一个模块加载器 ,则使用 loader：string，

如果要使用多个模块加载器，则使用 loaders：array

include:字符串或者数组，指包含的文件夹

exclude：字符串或者数组，指排除的文件夹

}

]

（4）module 除了可以配置 loaders 以外还能配置其他的值,在 webpack 的官网中获得更多的信息

7.resolve.extensions 详细说明

（1）resolve.extensions 并不是必须配置的，当不配置时，会使用默认值["", ".webpack.js", ".web.js", ".js"]，当手动为 resolve.extensions 设置值，它的默认值会被覆盖

（2）如果你想要每个模块都能够按照它们自己扩展名正确的被解析，要在数组中添加一个空字符串。

（3）如果你想请求一个 js 文件但是在请求时不带扩展（如：require('somecode')),那么就需要将'.js'添加到数组中。其他文件一样

（4）resolve 还有其他的配置项，在 webpack 的官网获得更多信息

8.补充

（1）当设置了配置文件后，在命令行中输入 webpack 就可按照默认配置文件中的配置项打包模块了。

（2）设置多个 webpack 配置文件。webpack 默认的配置文件是 webpack.config.js,当在命令行中输入 webpack 时默认找的是 webpack.config.js。通过在 package.json 的 scripts 中添加例如"start-html":"webpack --config webpack.html.config.js"在命令行中输入 npm run start-html 查找的就是

webpack.html.config.js，通过这种方式可以实现不同的配置文件有不同的用处，这样就不用反复修改同一个配置文件

9.下面是一个简单的配置文件

三.webpack-dev-server

1.webpack-dev-server 是一个轻量级的服务器，修改文件源码后，自动刷新页面将修改同步到页面上