HTML5 面试题

一、 Doctype 的作用？严格模式和混杂模式的区分，以及如何触发这 2 种模式?

<!DOCTYPE>声明位于文档中的最前面，处于<html>标签之前。告知浏览器的解析器，用什么文档类型 规范来解析这个文档。

DOCTYPE 不存在或格式不正确会导致文档以混杂模式呈现。

严格模式就是浏览器根据 web 标准去解析页面，是一种要求严格的 DTD(Document Type Definition)，不允许使用任何表现层的语法，

混杂模式是一种向后兼容的解析方法。

触发标准模式或者说严格模式很简单，就是 Html 前申明正确的 DTD，出发混杂模式可以在 html 文档开始不声明 DTD，或者在 DOCTYPE 前加入 XML 声明

二、 请写出至少 20 个 HTML5 标签

<article><aside><audio><canvas><datalist><command><details><embed><figcaption><figure><footer><header><hgroup><keygen><mark><nav><section><time><video><summary><meter><output><progress><source>

三、 语义化的理解？

1.html 语义化就是让页面的内容结构化，便于对浏览器、搜索引擎解析；

2.在没有样式 CSS 情况下也以一种文档格式显示，并且是容易阅读的。

3.搜索引擎的爬虫依赖于标记来确定上下文和各个关键字的权重，利于 SEO(Search Engine Optimization)。

4.使阅读源代码的人对网站更容易将网站分块，便于阅读维护理解。

四、 列举 5 种 IE haslayout 的属性及其值

haslayout 是 Windows Internet Explorer 渲染引擎的一个内部组成部分。在 Internet Explorer 中，一个元素要么自己对自身的内容进行计算大小和组织，要么依赖于父元素来计算尺寸和组织内容。为了调节这两个不同的概念，渲染引擎采用了 hasLayout 的属性，属性值可以为 true 或 false。当一个元素的 hasLayout 属性值为 true 时，我们说这个元素有一个布局（layout）

部分的 IE 显示的错误，都可以通过激发元素的 haslayout 属性来修正。可以通过设置 css 尺寸属性(width/height)等来激发元素的 haslayout，使其“拥有布局”。如下所示，通过设置以下 css 属性即可。

*** display: inline—block**
*** height: (任何值除了 auto)**
*** float: (left 或 right)**
*** position: absolute**
*** width: (任何值除了 auto)**
*** writing—mode: tb—rl; （实现文字可以垂直显示，具体意义可百度了解）**
*** zoom: (除 normal 外任意值)**

Internet Explorer 7 还有一些额外的属性(不完全列表):

*** min—height: (任意值)**
*** max—height: (除 none 外任意值)**
*** min—width: (任意值)**
*** max—width: (除 none 外任意值)**
*** overflow: (除 visible 外任意值)**
*** overflow—x: (除 visible 外任意值)是否对内容的左/右边缘进行裁剪。**
*** overflow—y: (除 visible 外任意值)是否对内容的上/下边缘进行裁剪。**
*** position: fixed**

五、 简述 jpg，gif，png-8，png-24 的区别，及其各自的使用场景

gif、jpg、png 格式的图片在网站制作中的区别

Gif 格式特点:

1.透明性,Gif 是一种布尔透明类型，它既可以是全透明，也可以是全不透明，但是它并没有半透明（alpha 透明）。

2.动画,Gif 这种格式支持动画。

3.无损耗性,Gif 是一种无损耗的图像格式，这也意味着你可以对 gif 图片做任何操作也不会使得图像质量产生损耗。

4.水平扫描,Gif 是使用了一种叫做 LZW 的算法进行压缩的，当压缩 gif 的过程中，像素是由上到下水平压缩的，这也意味着同等条件下，横向的 gif 图片比竖向的 gif 图片更加小。例如 500*10 的图片比 10*500 的图片更加小

5.间隔渐进显示,Gif 支持可选择性的间隔渐进显示

由以上特点看出只有 256 种颜色的 gif 图片不适合照片，但它适合对颜色要求不高的图形（比如说图标，图表等），它并不是最优的选择，我们会在后面中看到 png 是最优的选择。

Jpeg（jpg）格式特点:

1.透明性,它并不支持透明。

2.动画,它也不支持动画。

3.损耗性,除了一些旋转（仅仅是 90、180、270 度旋转），裁切，从标准类型到先进类型，编辑图片的原数据之外，所有其它操作对 jpeg 图像的处理都会使得它的质量损失。所以我们在编辑过程一般用 png 作为过渡格式。

4.隔行渐进显示,它支持隔行渐进显示（但是 ie 浏览器并不支持这个属性，但是 ie 会在整个图像信息完全到达的时候显示）。

由上可以看出 Jpeg 是最适 web 上面的摄影图片和数字照相机中。

Png 格式特点:

1.类型,Png 这种图片格式包括了许多子类，但是在实践中大致可以分为 256 色的 png 和全色的 png，你完成可以用 256 色的 png 代替 gif，用全色的 png 代替 jpeg

2.透明性,Png 是完全支持 alpha 透明的（透明，半透明，不透明），尽管有两个怪异的现象在 ie6（下面详细讨论）

3.动画,它不支持动画

PNG 图片格式现在包含三种类型：

1.PNG8 256 色 PNG 的别名

2.PNG 24 全色 PNG 的别名

3.PNG 32 全色 PNG 的别名

基本上 PNG32 就是 PNG24，但是附带了全 alpha 通道。就是说每个像素上不仅存储了 24 位真色彩信息还存储了 8 位的 alpha 通道信息，就如同 GIF 能存储透明和不透明信息一样。当我们把图片放到不太搭配的背景上的时候，透明 PNG 图片的边缘会显示得更加平滑。

当然，我也知道你的想法，“但是 Photoshop 也能生成带透明通道的 PNG 图片！”我也知道，它只是表面上这么说是 PNG24，让我也产生困惑了。

作为一个伤感的 Fireworks 倡导者，我只使用 PNG32 支持附带 alpha 通道的真色彩图片。不管怎样，如果你习惯使用 Photoshop，你就应该知道，Photoshop 在“存储为 WEB 格式”中只提供 PNG8 和 PNG24 两种 PNG 格式。

我敢肯定你经常会勾选“支持透明”选项，以获得带有透明度的 PNG 图片，但是这样你就获取了一张 PNG32 图片。——Photoshop 只是觉得把 PNG32 这个名称给隐藏掉了。。。。

对 png8 的误解

Png8 的在 ie 中的怪异表现：

半透明的 png8 在 ie6 以下的浏览器显示为全透明。

Alpha 透明的全色 PNG（png32）在 ie6 中会出现背景颜色（通常是灰色）。

由上面可以总结：

（a）全透明的 png8 可以在任一浏览器正常显示（就像 gif一样）。半透明的 png8 在除了 ie6 及其以下的浏览器下错误的显示成全透明，其它浏览器都能正常显示半透明。这个 bug 并不需要特殊对待，因为在不支持半透明的浏览器下只是显示为全透明，对用户体验影响不大，它反而是透明 gif 的加强版。

（b）第二个 bug 没有什么好的方法解决，只能通过影响性能的方法 AlphaImageLoader 与需要加特殊标签（VML）。

因此得出结论就是：请使用 PNG8。

Png8 的软件问题：

Photoshop 只能导出布尔透明的 PNG8。

Fireworks 既能导出布尔透明的 PNG8，也能导出 alpha 透明的 PNG8.

六、 能够设置文本加粗的样式属性是什么

字体加粗（font—weight）

功能：用于设置字体笔划的粗细。

属性值：正常度 — normal

相对度 — bold, bolder, light, lighter

渐变度 — 100, 200, 300, 400(相当于 normal), 500, 600, 700(相当于 bold、lighter、bolder、以及数值 100－900。

语法为：h1 {font—weight: 属性值}

七、 Html 和 xhtml 有什么区别?

html 是一种基本的 web 网页设计语言，xhtml 是一个基于 XML 的置标语言。

最主要的不同:

XHTML 元素必须正确的被嵌套，元素必须关闭，标签必须小写，必须有根元素。

八、 算法题：有一个长度为 n—1 的数组，包含 1—n 中不重复的乱序的数，求寻找范围内不在数组中的数，考虑空间占用，性能优化，溢出等情况，至少写两个算法

当 n 不太大时，可以考虑求和。先算出 1~n 的所有数的和，然后减去数组中出现的所有自然数的和。时间复杂度为 O(n)，空间复杂度 O(1)。这种方法的缺点是 n 不能太大，n 比较大时，求和容易溢出。

用位图。从头到尾的扫描整个数组，把出现的数相应的位设置为 1.然后再扫描位图，找出不为 1 的那一位，即为要找的数。这种方法的时间复杂度为 O(n),空间复杂度为 O(n)。

异或有个很巧妙的地方：同一变量和该变量与另一变量的异或值的异或等于这个变量自身。所以我们可以把 1~n 的所有数异或，再把数组中出现的所有数异或，然后再把这两个异或的结果异或，最后得到的值即为我们要找的值。这样时间复杂度为 O(n),空间复杂度为 O（1）。在空间上比第二种方法要好，而且不会出现第一种方法中所说的溢出问题。

九、 实现以下方法（与标准一致）

Element.prototype.getElementsByClassName：

```
Element.prototype.getElementsByClassName = function(searchClass,
node, tag) {
        if (document.getElementsByClassName) {
            var nodes = (node ||
document).getElementsByClassName(searchClass),
    result = [];
            for (var i = 0; node = nodes[i++]; ) {
                if (tag !== "*"&& node.tagName === tag.toUpperCase()) {
                    result.push(node);
                }
            }
            return result;
        } else {
            node = node || document;
            tag = tag || "*";
            var classes = searchClass.split(""),
    elements = (tag === "*"&& node.all) ? node.all :
node.getElementsByTagName(tag),
    patterns = [],
    current,
    match;
```

```
                var i = classes.length;
                while (——i >= 0) {
                    patterns.push(new RegExp("(^|\\s)" + classes[i] + "(\\s|$)"));
                }
                var j = elements.length;
                while (——j >= 0) {
                    current = elements[j];
                    match = false;
                    for (var k = 0, kl = patterns.length; k < kl; k++) {
                        match = patterns[k].test(current.className);
                        if (!match)
                            break;
                    }
                    if (match)
                        result.push(current);
                }
                return result;
            }
}
```

Function.prototype.bind：

```
        Function.prototype.bind = function (oThis) {
            if (typeof this !== "function") {
         throw new TypeError("bind function error");
            }
            var aArgs = Array.prototype.slice.call(arguments,1),
                fToBind = this,
                fBound = function () {
          return fToBind.apply(oThis ||
      window,aArgs.concat(Array.prototype.slice.call(arguments)));
                };
            return fBound;
};
```

十、 编写一个方法去掉一个数组的重复元素

1.遍历数组法

最简单的去重方法， 实现思路：新建一新数组，遍历传入数组，值不在新数组就加入

该新数组中；注意点：判断值是否在数组的方法“indexOf”是 ECMAScript5 方法，IE8 以下不支持，需多写一些兼容低版本浏览器代码，源码如下：

```
// 最简单数组去重法
function unique1(array){
  var n = []; //一个新的临时数组
  //遍历当前数组
  for(var i = 0; i < array.length; i++){
    //如果当前数组的第 i 已经保存进了临时数组，那么跳过，
    //否则把当前项 push 到临时数组里面
    if (n.indexOf(array[i]) == -1) n.push(array[i]);
  }
  return n;
}
// 判断浏览器是否支持 indexOf ，indexOf 为 ecmaScript5 新方法 IE8 以下（包括 IE8， IE8 只支持部分 ecma5）不支持
if (!Array.prototype.indexOf){
  // 新增 indexOf 方法
  Array.prototype.indexOf = function(item){
    var result = -1, a_item = null;
    if (this.length == 0){
      return result;
    }
    for(var i = 0, len = this.length; i < len; i++){
      a_item = this[i];
      if (a_item === item){
        result = i;
        break;
      }
    }
    return result;
  }
}
```

2.对象键值对法

该方法执行的速度比其他任何方法都快， 就是占用的内存大一些；实现思路：新建一js 对象以及新数组，遍历传入数组时，判断值是否为 js 对象的键，不是的话给对象新增该键并放入新数组。注意点： 判断是否为 js 对象键时，会自动对传入的键执行

“toString()”，不同的键可能会被误认为一样；例如： a[1]、a["1"] 。解决上述问题还是得调用“indexOf”。

```
        // 速度最快， 占空间最多（空间换时间）
        function unique2(array){
            var n = {}, r = [], len = array.length, val, type;
                for (var i = 0; i < array.length; i++) {
                        val = array[i];
                        type = typeof val;
                        if (!n[val]) {
                                n[val] = [type];
                                r.push(val);
                        } else if (n[val].indexOf(type) < 0) {
                                n[val].push(type);
                                r.push(val);
                        }
                }
                return r;
}
```

3.数组下标判断法

还是得调用“indexOf”性能跟方法 1 差不多，实现思路：如果当前数组的第 i 项在当前数组中第一次出现的位置不是 i，那么表示第 i 项是重复的，忽略掉。否则存入结果数组。

```
        function unique3(array){
            var n = [array[0]]; //结果数组
            //从第二项开始遍历
            for(var i = 1; i < array.length; i++) {
                //如果当前数组的第 i 项在当前数组中第一次出现的位置不是 i，
                //那么表示第 i 项是重复的，忽略掉。否则存入结果数组
                if (array.indexOf(array[i]) == i) n.push(array[i]);
            }
            return n;
}
```

4.排序后相邻去除法

虽然原生数组的”sort”方法排序结果不怎么靠谱，但在不注重顺序的去重里该缺点毫无影响。实现思路：给传入数组排序，排序后相同值相邻，然后遍历时新数组只加入不与前一值重复的值。

```
        // 将相同的值相邻，然后遍历去除重复值
        function unique4(array){
            array.sort();
            var re=[array[0]];
            for(var i = 1; i < array.length; i++){
                if( array[i] !== re[re.length—1])
                {
                    re.push(array[i]);
                }
            }
            return re;
}
```

5.优化遍历数组法

实现思路：获取没重复的最右一值放入新数组。（检测到有重复值时终止当前循环同时进入顶层循环的下一轮判断）

```
        // 思路：获取没重复的最右一值放入新数组
        function unique5(array){
            var r = [];
            for(var i = 0, l = array.length; i < l; i++) {
                for(var j = i + 1; j < l; j++)
                    if (array[i] === array[j]) j = ++i;
                r.push(array[i]);
            }
            return r;
}
```

十一、 请使用 javascript 写出数组快速排序代码

```
<script>
function quickSort(arr){
        var   left =[],right=[];
        if(arr.length<1){
            return   arr;
        }
        var   index = Math.floor(arr.length/2);
        var   point = arr.splice(index,1);
```

```
        for(var i=0,len=arr.length;i<len;i++){
            if(arr[i]<point){
        left.push(arr[i]);
}else{
    right.push(arr[i]);
}
        }
        return quickSort(left).concat(point,quickSort(right));
    }
</script>
```

十二、 编写一个布局，页面宽度自适应，最小宽度 300px，左边定宽 35%，右边定宽 65%

```
<div class="container">
        <div class="left"></div>
        <div class="right"></div>
    </div>
    <style>
        .container{
            height: 600px;
            _width: 300px;
            min-width: 300px;
        }
        .left{
            width: 35%;
            height: 100%;
            background: #ff0;
            float: left;
        }
        .right{
            overflow:hidden;
            width: 65%;
            height: 100%;
            background: #0f0;
        }
    </style>
```

十三、 谈谈对 html5 的了解

1.良好的移动性，以移动设备为主。

2.响应式设计，以适应自动变化的屏幕尺寸

3.支持离线缓存技术，webStorage 本地缓存

4.新增 canvas，video，audio 等新标签元素。新增特殊内容元素：article，footer，header，nav，section 等，新增表单控件：calendar，date，time，email，url，search。

5.地理定位...

6.新增 webSocket/webWork 技术

十四、 Js 面向对象的几种方式

1.对象的字面量 var obj = {}

2.创建实例对象 var obj = new Object();

3.构造函数模式 function fn(){} , new fn();

4.工厂模式：用一个函数，通过传递参数返回对象。function fn(params){var obj =new Object();obj.params = params; return obj;},fn(params);

5.原型模式：function clock(hour){} fn.prototype.hour = 0; new clock();

首先，每个函数都有一个 prototype(原型)属性，这个指针指向的就是 clock.prototype 对象。而这个原型对象在默认的时候有一个属性 constructor，指向 clock，这个属性可读可写。而当我们在实例化一个对象的时候，实例 newClock 除了具有构造函数定义的属性和方法外（注意，只是构造函数中的）,还有一个指向构造函数的原型的指针，ECMAScript 管他叫[[prototype]]，这样实例化对象的时候，原型对象的方法并没有在某个具体的实例中，因为原型没有被实例。

十五、 在 css 中哪个属性会影响 dom 读取文档流的顺序

1.direction, writing—mode

十六、 前端页面由哪三层构成，分别是什么，作用是什么

Html (结构)：超文本标记语言,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。”

Css (表现)：层叠样式表 ，由 css 负责创建。css 对“如何显示有关内容”的问题做出了回答。

Js (行为)：客户端脚本语言 ，内容应该如何对事件做出反应

十七、 Css 的基本语句构成是？

语法:

（自定义的样式名称）｛

样式内容（属性：属性值；）

｝

十八、 如何对网站的文件和资源进行优化

1．文件合并（目的是减少 http 请求）

2．文件压缩 （目的是直接减少文件下载的体积）

3．使用 cdn 托管资源

4．使用缓存

5．gizp 压缩需要的 js 和 css 文件

6．meta 标签优化（title,description,keywords）,heading 标签的优化,alt 优化

7．反向链接，网站外链接优化

十九、 Javascipt 的本地对象，内置对象和宿主对象

本地对象：Object、Function、Array、String、Boolean、Number、Date、RegExp、Error、EvalError、RangeError、ReferenceError、SyntaxError、TypeError、URIError, 简单来说，本地对象就是 ECMA—262 定义的类.

内置对象：ECMA—262 把内置对象（built—in object）定义为“由 ECMAScript 实现提供的、独立于宿主环境的所有对象，在 ECMAScript 程序开始执行时出现”。这意味着开发者不必明确实例化内置对象，它已被实例化了。

同样是“独立于宿主环境”。根据定义我们似乎很难分清“内置对象”与“本地对象”的区别。而 ECMA—262 只定义了**两个内置对象，即 Global 和 Math** （它们也是本地对象，根据定义，每个内置对象都是本地对象）。

如此就可以理解了。内置对象是本地对象的一种。而其包含的两种对象中，Math 对象我们经常用到，可这个 Global 对象是啥东西呢？

Global 对象是 ECMAScript 中最特别的对象，因为实际上它根本不存在，有点玩人的意思。大家要清楚，在 ECMAScript 中，不存在独立的函数，所有函数都必须是某个对象的方法。

类似于 isNaN()、parseInt()和 parseFloat()方法等，看起来都是函数，而实际上，它们都是 Global 对象的方法。而且 Global 对象的方法还不止这些.

宿主对象:ECMAScript 中的“宿主”就是我们网页的运行环境,即“操作系统”和“浏览器”。所有非本地对象都是宿主对象（host object），即由 ECMAScript 实现的宿主环境提供的对象。所有的 BOM 和 DOM 对象都是宿主对象。因为其对于不同的“宿主”环境所展示的内容不同。其实说白了就是，ECMAScript 官方未定义的对象都属于宿主对象，因为其未定义的对象大多数是自己通过 ECMAScript 程序创建的对象。自定义的对象也是宿主对象。

二十、 输入 url 后的加载过程

1) 查找域名对应 IP 地址

2) 建立连接(TCP 的三次握手)

3) 构建网页

4) 断开连接(TCP 的四次挥手)

二十一、 说说 TCP 传输的三次握手四次挥手策略

为了准确无误地把数据送达目标处，TCP 协议采用了**三次握手**策略。用 TCP 协议把数据包送出去后，TCP 不会对传送 后的情况置之不理，它一定会向对方确认是否成功送达。握手过程中使用了 TCP 的标志：SYN 和 ACK。

发送端首先发送一个带 SYN 标志的数据包给对方。接收端收到后，回传一个带有 SYN/ACK 标志的数据包以示传达确认信息。

最后，发送端再回传一个带 ACK 标志的数据包，代表“握手”结束。

若在握手过程中某个阶段莫名中断，TCP 协议会再次以相同的顺序发送相同的数据包。

断开一个 TCP 连接则需要“**四次挥手**”：

第一次挥手：主动关闭方发送一个 FIN，用来关闭主动方到被动关闭方的数据传送，也就是主动关闭方告诉被动关闭方：我已经不 会再给你发数据了(当然，在 fin 包之前发送出去的数据，如果没有收到对应的 ack 确认报文，主动关闭方依然会重发这些数据)，但是，此时主动关闭方还可 以接受数据。

第二次挥手：被动关闭方收到 FIN 包后，发送一个 ACK 给对方，确认序号为收到序号+1（与 SYN 相同，一个 FIN 占用一个序号）。

第三次挥手：被动关闭方发送一个 FIN，用来关闭被动关闭方到主动关闭方的数据传送，也就是告诉主动关闭方，我的数据也发送完了，不会再给你发数据了。

第四次挥手：主动关闭方收到 FIN 后，发送一个 ACK 给被动关闭方，确认序号为收到序号+1，至此，完成四次挥手。

二十二、 JQuery 中有几种类型的选择器

1.层叠选择器$(“form input”)

2.基本过滤选择器 :first :last :not()

3.内容过滤选择器:odd:eq():animated

4.可视化过滤选择器 :hidden :visible

5.属性过滤选择器：div[id]

6.子元素过滤选择器:first—child :last—child :only :child

7.表单元素过滤选择器 :enabled :disabled :checked :selected

8.id,类，类型,元素...

二十三、 jQuery 中的 Delegate()函数有什么作用

delegate() 方法为指定的元素（属于被选元素的子元素）添加一个或多个事件处理程序，并规定当这些事件发生时运行的函数。

使用 delegate() 方法的事件处理程序适用于当前或未来的元素（比如由脚本创建的新元素）。$("div").delegate("button","click",function(){

$("p").slideToggle();});

二十四、 行内元素有那些。块级元素有那些。空元素有哪些

首先：CSS 规范规定，每个元素都有 display 属性，确定该元素的类型，每个元素都有默认的 display 值，如 div 的 display 默认值为“block”，则为“块级”元素；span 默认

display 属性值为“inline”，是“行内”元素。

（1）行内元素有：a b span select strong（强调的语气）img input（内联元素）

（2）块级元素有：div ul ol li dl dt dd h1 h2 h3 h4…p

（3）常见的空元素：

<hr><img><input><link><meta>

鲜为人知的是：

<area><base><col><command><embed><keygen><param><source><track><wbr>

二十五、 说几条 javasprit 的基本规范

1.不要在同一行声明多个变量。

2.请使用 ===/!==来比较 true/false 或者数值

3.使用对象字面量替代 new Array 这种形式

4.不要使用全局函数。

5.Switch 语句必须带有 default 分支

6.函数不应该有时候有返回值，有时候没有返回值。

7.For 循环必须使用大括号

8.If 语句必须使用大括号

9.for—in 循环中的变量 应该使用 var 关键字明确限定作用域，从而避免作用域污染。

二十六、 介绍一下标准的 css 盒模型，低版本 ie 盒模型有什么不同

（1）盒模型有两种， IE 盒子模型、W3C 盒子模型；

（2）盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border)；

（3）区 别： IE 的 width 部分把 border 和 padding 计算了进去;

二十七、 说出三种减少页面加载的方法（加载时间指感知的时间或实际加载的时间）

CSS Sprites；

JS、CSS 源码压缩、图片大小控制合适；

网页 Gzip；

CDN 托管；

data 缓存 ；

图片服务器；

二十八、 用 js 代码简单的介绍下自己

```
<script>
    function Person(name,jingli,jineng) {
```

```
        this.name=name;
        this.jingli=jingli;
        this.jineng=jineng;
    }
    Person.prototype.show=function(){
        console.log("我是"+this.name+"；我有如下经历:"+this.jingli+"；我会如下技能："+this.jineng);
    }
    var myself=new Person("小田","小田工作室创办人，凤翔网络推广顾问","熟悉前端基本技能，熟悉网络营销思想有实战经验，掌握项目经理技能，可以编写文档，也可以使用 axure 进行原型设计，掌握自动化测试和性能测试技能")
    myself.show();
</script>
```

二十九、 Html5 中 datalist 是什么

<datalist> 标签定义选项列表，与 input 元素配合使用该元素，来定义 input 可能的值。

datalist 及其选项不会被显示出来，它仅仅是合法的输入值列表。

```
    <input id="myCar" list="cars" />
    <datalist id="cars">
    <option value="BMW">
    <option value="Ford">
    <option value="Volvo">
    </datalist>
```

三十、 Ajax 同步和异步的区别，如何解决跨域问题

同步的概念应该是来自于 OS 中关于同步的概念:不同进程为协同完成某项工作而在先后次序上调整(通过阻塞,唤醒等方式).同步强调的是顺序性.谁先谁后.异步则不存在这种顺序性.

同步：浏览器访问服务器请求，用户看得到页面刷新，重新发请求,等请求完，页面刷新，新内容出现，用户看到新内容,进行下一步操作。

异步：浏览器访问服务器请求，用户正常操作，浏览器后端进行请求。等请求完，页面不刷新，新内容也会出现，用户看到新内容。

jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面

三十一、 列举几种后端通讯的方法及其使用的场景，关于跨域的理解。

1.后端程序可以通过 session 来进行通讯，session 有过期时间，主要用于验证码的验

证，登录过期等的应用。

2.数据库，数据库支持多种语言的操作，那么通过数据库就可以通讯。

关于跨域：

跨域请求存在的原因：由于浏览器的同源策略，即属于不同域的页面之间不能相互访问各自的页面内容。

跨域的场景：

1.域名不同 www.yangwei.com 和 www.wuyu.com 即为不同的域名）

2.二级域名相同，子域名不同（www.wuhan.yangwei.com www.shenzheng.yangwei.com 为子域不同）

3.端口不同，协议不同 （ http://www.yangwei.com 和 https://www.yangwei.com 属于跨域 www.yangwei.con:8888 和 www.yangwei.con:8080)

跨域的方式：（内容较多，需掌握 CORS 和 jsonp，其他内容也要了解）

1.前端的方式: possMessage，window.name,document.domain,image.src(得不到数据返回)，jsonP(script.src 后台不配合得不到数据返回)，style.href（得不到数据返回)

一.image.src,script.src,style.href 不受同源策略的影响可以加载其他域的资源，可以用这个特性，向服务器发送数据。最常用的就是使用 image.src 向服务器发送前端的错误信息。image.src 和 style.href 是无法获取服务器的数据返回的，script.src 服务器端配合可以得到数据返回。

二 possMessage,window.name,document.domain 是两个窗口直接相互传递数据。

（1）possMessage 是 HTML5 中新增的，使用限制是 必须获得窗口的 window 引用。IE8+支持，firefox，chrome,safair,opera 支持

（2）window.name ，在一个页面中打开另一个页面时，window.name 是共享的，所以可以通过 window.name 来传递数据，window.name 的限制大小是 2M，这个所有浏览器都支持,且没有什么限制。

3） document.domain 将两个页面的 document.domain 设置成相同，document.domain 只能设置成父级域名，既可以访问，使用限制：这顶级域名必须相同

2.纯后端方式: CORS，服务器代理

CORS 是 w3c 标准的方式,通过在 web 服务器端设置：响应头 Access—Cntrol—Alow—Origin 来指定哪些域可以访问本域的数据，ie8&9(XDomainRequest),10+,chrom4 ，firefox3.5,safair4，opera12 支持这种方式。

服务器代理，同源策略只存在浏览器端，通过服务器转发请求可以达到跨域请求的目的，劣势：增加服务器的负担，且访问速度慢。

3.前后端结合:JsonP

script.src 不受同源策略的限制，所以可以动态的创建 script 标签，将要请求数据的域写在 src 中参数中附带回调的方法，服务器端返回回调函数的字符串，并带参数。

如 script.src="http://www.yangwei.com/?id=001&callback=getInfoCallback",服务器端返回 getInfoCallBack("name:yangwei;age:18") 这段代码会直接执行，在前面定义好 getInfoCallBack 函数，既可以获得数据并解析。 这种是最常见的方式。

4.webSocket（了解性拓展）

服务端推送 websocket 和 sse 场景及应用

应用场景

都可以进行服务端推送,并且都是使用长连接来进行.但两者的实现又有一点不同,sse 仍使用 http 协议,并且使用相同的链接发送正常的 http 协议报文.而 websocket 是使用 http 协议进行握手,然后再使用同一个链接进行 websocket 协议的通信.

websocket 可以进行双向的通信,即服务端可以往客户端发信息,客户端也可以向服务端发信息.而 sse 是单向的,只能由服务端往客户端发.

websocket 自带连接的保持,即通过 ping/pong 协议保证连接可以始终维持,sse 没有这个保证,不过可以参考 ping/pong 协议,自己周期性地发送信息来同样地进行处理.比如,5 秒往客户端发一个特别的信息(通过 type/name 进行区分).其次,因为是基于浏览器的使用,sse 有一个特性,就是浏览器发现一个连接断掉了,就会自动地进行重联,即重新发送请求.这样,服务端也不用担心连接被断开,不过需要处理新的请求必须和上一次请求的内容相连续,以及新的推送注册.

因为都是使用 http 协议进行起始处理,因此在签权上都可以使用到 http 协议本身的一些东西,比如 header/cookie 签权.在相应的握手阶段,通过读取 cookie(session)来保证相应的请求必须是经过授权的,也可以用于定位使用人.甚至可以通过这些信息保证单个用户只能有一个请求,避免重复请求

由于都是基于浏览器使用,因此建议的数据传输都是文本型.虽然 websocket 支持二进制 frame 传输,不过一些都不建议使用.sse 只能传输文本

不管是 websocket 还是 sse,在用于通信时,都建议只用于进行数据的推送,而不是进行完整的应用处理.这里可以理解为,常规的业务处理仍然交给后端的服务来处理.这样,即可以使用之前的业务开发的优势,又可以使用推送的优势.而不是走向另一个级端,即所有的信息都想通过推送来传递.

开发方式

websocket 开发首选 netty,因为 netty 对协议的封装已经做到了完全的支持.通过 HttpServerCodec 作为握手协议,WebSocketServerProtocolHandler 作为协议处理,然后再加一个自己的 handler,就完成了相应的业务处理.同时在性能上,netty 在一个 ws 的请求建立起来之后,会自动地去除 httpServerCodec 相关的 handler,这样保证后续的处理都是按照 ws 的协议来进行.

sse 开发首选 jersey,jersey—media—sse 提供了相应的 sse 支持,并且通过与 rest 相集成,开发一个 sse 就跟普通的业务开发相同.

ws 和 sse 在文本支持上都只支持 utf—8 编码,因此在处理上需要注册编码方式.同时在使用 sse 时,如果后端第一次进行响应时,相应的编码不对.chrome 会直接报错,包括 utf8 都会报错(这是之前后端开发的一个问题),可以修正或者增加相应的拦截器,保证后端 content—type 响应中的 charset=UTF—8.

ws 和 sse 都可以通过 nginx 进行代理转发.ws 的处理只需要设置 http 版本,以及重新转发前端的 Upgrade 和 Connection 头即可.而 sse,也可以通过禁用 buffer 来处理.参考 http://stackoverflow.com/questions/27898622/server—sent—events—stopped—work—after—enabling—ssl—on—proxy

特定实现

为保证在开发时推送类的和业务类的系统不会耦合在一起,或者同一个应用内有两种处理模式的功能存在.建议直接在系统层就开发 2 个不同的系统,一个专门用于推送,另一个用于相应的业务处理.然后业务处理后的数据,需要再交由推送处理,则可以在后端进行通过消息系统进行中转,如 kafka(持久保证)或 redis(内存订阅)等

因为二者在 ie 上的支持都很有限,因此不建议在 ie 上进行尝试

使用 sse 还是 websocket,取决于是否需要前台交互,还取决于对后端的支持技术的了解程序.比如,了解 jersey 多一点,还是 netty 多一点.由于最近 netty 进行微服务化底层通信支持越来越流行,个人更倾向于使用 websocket.但如果仅仅是一个简单的推送功能,又不希望修改代码,那也可以使用 jersey(毕竟之前的系统就是在上面进行开发的)

需要后端有的时候需要进行定向发送或者是群发,这种需求 ws 和 sse 的实现中都有相应的处理.如 ChannelGroup 和 SseBroadcaster,这样在后端获取到一个消息,需要进行路由时就可以从这里面拿相应的 channel 信息.不过,前提是对各个 channel 上进行了特定的消息绑定,这样就好区分具体的路由信息.具体路由策略可以在建立时绑定 session,后续通过 session 来路由.

三十二、 设计一个幻灯应用，需要列举选择的基础框架、项目的基础框架和代码管理、幻灯数据的存储和读取，部分特效的实现，可以只写思路，后续面聊。

本题无标准答案，同学们可以自己研究考虑一下，。

三十三、 Html5 中本地存储概念是什么，有什么优点，与 cookie 有什么区别？

html5 中的 Web Storage 包括了两种存储方式：sessionStorage 和 localStorage。

sessionStorage 用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此 sessionStorage 不是一种持久化的本地存储，仅仅是会话级别的存储。而 localStorage 用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的；

cookie 是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。

区别：

1、 cookie 数据始终在同源的 http 请求中携带（即使不需要），即 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。cookie 数据还有路径（path）的概念，可以限制 cookie 只属于某个路径下。

2、 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。

三十四、 说说你对作用域链的理解

作用域链的作用是保证执行环境里有权访问的变量和函数是有序的，作用域链的变量只能向上访问，变量访问到 window 对象即被终止，作用域链向下访问变量是不被允许的。

三十五、 什么是 ajax 和 json，它们的优缺点

ajax 的全称：Asynchronous Javascript And XML。

异步传输+js+xml。实现无刷新状态更新页面和异步提交

所谓异步，在这里简单地解释就是：向服务器发送请求的时候，我们不必等待结果，而是可以同时做其他的事情，等到有了结果它自己会根据设定进行后续操作，与此同时，页面是不会发生整页刷新的，提高了用户体验。

Ajax 实现过程：

(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

优点：

不需要插件支持

用户体验极佳

提升 Web 程序性能

减轻服务器和宽带的负担

缺点：

前进后退按钮被破坏

搜索引擎的支持不够

开发调试工具缺乏

JSON（JavaScript Object Notation）和 XML一样也是一种简单文本格式。是一种比较流行的标准格式，是数据的载体，相对于 XML，JSON 更加易读、更便于肉眼检查。在语法的层面上，JSON 与其他格式的区别是在于分隔数据的字符，JSON 中的分隔符限于单引号、小括号、中括号、大括号、冒号和逗号。

优点：

作为一种数据传输格式，JSON 与 XML 很相似，但是它更加灵巧。

JSON 不需要从服务器端发送含有特定内容类型的首部信息。

缺点：

语法过于严谨

代码不易读

eval 函数存在风险

三十六、 Html5 有那些新增的表单元素

表单控：color，calendar，date，datetime，datetime—local，time，mouth，week，email，url，search，range，tel

新的表单元素：datalist，keygen，output

三十七、 http 状态码有那些，分别代表什么意思

简单版:

100 Continue 继续，一般在发送 post 请求时，已发送了 http header 之后服务端将返回此信息，表示确认，之后发送具体参数信息

200 OK 正常返回信息

201 Created 请求成功并且服务器创建了新的资源

202 Accepted 服务器已接受请求，但尚未处理

301 Moved Permanently 请求的网页已永久移动到新位置。

302 Found 临时性重定向。

303 See Other 临时性重定向，且总是使用 GET 请求新的 URI。

304 Not Modified 自从上次请求后，请求的网页未修改过。

400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。

401 Unauthorized 请求未授权。

403 Forbidden 禁止访问。

404 Not Found 找不到如何与 URI 相匹配的资源。

500 Internal Server Error 最常见的服务器端错误。

503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。

完整版

1**(信息类)：表示接收到请求并且继续处理

100——客户必须继续发出请求

101——客户要求服务器根据请求转换 HTTP 协议版本

2**(响应成功)：表示动作被成功接收、理解和接受

200——表明该请求被成功地完成，所请求的资源发送回客户端

201——提示知道新文件的 URL

202——接受和处理、但处理未完成

203——返回信息不确定或不完整

204——请求收到，但返回信息为空

205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件

206——服务器已经完成了部分用户的 GET 请求

3**(重定向类)：为了完成指定的动作，必须接受进一步处理

300——请求的资源可在多处得到

301——本网页被永久性转移到另一个 URL

302——请求的网页被转移到一个新的地址，但客户访问仍继续通过原始 URL 地址，重定向，新的 URL 会在 response 中的 Location 中返回，浏览器将会使用新的 URL 发出新的 Request。

303——建议客户访问其他 URL 或访问方式

304——自从上次请求后，请求的网页未修改过，服务器返回此响应时，不会返回网页内容，代表上次的文档已经被缓存了，还可以继续使用

305——请求的资源必须从服务器指定的地址得到

306——前一版本 HTTP 中使用的代码，现行版本中不再使用

307——申明请求的资源临时性删除

4**(客户端错误类)：请求包含错误语法或不能正确执行

400——客户端请求有语法错误，不能被服务器所理解

401——请求未经授权，这个状态代码必须和 WWW一Authenticate 报头域一起使用

HTTP 401.1 — 未授权：登录失败

HTTP 401.2 — 未授权：服务器配置问题导致登录失败

HTTP 401.3 — ACL 禁止访问资源

HTTP 401.4 — 未授权：授权被筛选器拒绝

HTTP 401.5 — 未授权：ISAPI 或 CGI 授权失败

402——保留有效 ChargeTo 头响应

403——禁止访问，服务器收到请求，但是拒绝提供服务

HTTP 403.1 禁止访问：禁止可执行访问

HTTP 403.2 — 禁止访问：禁止读访问

HTTP 403.3 — 禁止访问：禁止写访问

HTTP 403.4 — 禁止访问：要求 SSL

HTTP 403.5 — 禁止访问：要求 SSL 128

HTTP 403.6 — 禁止访问：IP 地址被拒绝

HTTP 403.7 — 禁止访问：要求客户证书

HTTP 403.8 — 禁止访问：禁止站点访问

HTTP 403.9 — 禁止访问：连接的用户过多

HTTP 403.10 — 禁止访问：配置无效

HTTP 403.11 — 禁止访问：密码更改

HTTP 403.12 — 禁止访问：映射器拒绝访问

HTTP 403.13 — 禁止访问：客户证书已被吊销

HTTP 403.15 — 禁止访问：客户访问许可过多

HTTP 403.16 — 禁止访问：客户证书不可信或者无效

HTTP 403.17 — 禁止访问：客户证书已经到期或者尚未生效

404———个 404 错误表明可连接服务器，但服务器无法取得所请求的网页，请求资源不存在。eg：输入了错误的 URL

405——用户在 Request—Line 字段定义的方法不允许

406——根据用户发送的 Accept 拖，请求资源不可访问

407——类似 401，用户必须首先在代理服务器上得到授权

408——客户端没有在用户指定的饿时间内完成请求

409——对当前资源状态，请求不能完成

410——服务器上不再有此资源且无进一步的参考地址

411——服务器拒绝用户定义的 Content—Length 属性请求

412———个或多个请求头字段在当前请求中错误

413——请求的资源大于服务器允许的大小

414——请求的资源 URL 长于服务器允许的长度

415——请求资源不支持请求项目格式

416——请求中包含 Range 请求头字段，在当前请求资源范围内没有 range 指示值，请求也不包含 If—Range 请求头字段

417——服务器不满足请求 Expect 头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求长。

5**(服务端错误类)：服务器不能正确执行一个正确的请求

HTTP 500 — 服务器遇到错误，无法完成请求

HTTP 500.100 — 内部服务器错误 — ASP 错误

HTTP 500—11 服务器关闭

HTTP 500—12 应用程序重新启动

HTTP 500—13 — 服务器太忙

HTTP 500—14 — 应用程序无效

HTTP 500—15 — 不允许请求 global.asa

Error 501 — 未实现

HTTP 502 — 网关错误

HTTP 503：由于超载或停机维护，服务器目前无法使用，一段时间后可能恢复正常

三十八、 HTTP 的请求方法

HTTP（Hypertext Transfer Protocol）的八种请求方法：

方法	概述
❤GET	请求页面的详细信息，并返回实体主体。
❤POST	向指定资源提交数据进行数据请求（例如提交表单，或者上传文件）。数据被包含在请求体中。POST 请求可能会导致新的资源的建立和/或已有资源的修改。
PUT	从客户端向服务器传送的数据取代指定的文档内容。
DELETE	请服务器删除指定的页面。
HEAD	类似与 Get 请求，只不过返回的响应中没有具体的内容，用于获取报头
CONNECT	HTTP/1.1 协议中预留给能够将连接改为管道方式的代理服务器。
OPTIONS	允许客户端查看服务器的性能。
TRACE	回显服务器收到的请求，主要用于测试或诊断。

三十九、 什么是闭包（closure）为什么要用它

闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。

闭包的特性：

1.函数内再嵌套函数

2.内部函数可以引用外层的参数和变量

3.参数和变量不会被垃圾回收机制回收

例如：//li 节点的 onclick 事件都能正确的弹出当前被点击的 li 索引

```
<ul id="testUL">
<li> index = 0</li>
<li> index = 1</li>
<li> index = 2</li>
```

```
    <li> index = 3</li>
    </ul>
    <script type="text/javascript">
        var nodes = document.getElementsByTagName("li");
        for(i = 0;i<nodes.length;i+= 1){
            nodes[i].onclick = (function(i){
                    return function() {
                        console.log(i);
                    } //不用闭包的话，值每次都是 4
                })(i);
        }
    </script>
```

执行 say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在使得 Javascript 的垃圾回收机制 GC 不会收回 say667()所占用的资源

因为 say667()的内部函数的执行需要依赖 say667()中的变量

这是对闭包作用的非常直白的描述

```
function say667() {
    // Local variable that ends up within closure
    var num = 666;
    var sayAlert = function() {
        alert(num);
    }
    num++;
    return sayAlert;
}
  var sayAlert = say667();
  sayAlert()//执行结果应该弹出的 667
```

你知道哪些针对 jQuery 的优化方法

基于 Class 的选择性的性能相对于 Id 选择器开销很大，因为需遍历所有 DOM 元素。

频繁操作的 DOM，先缓存起来再操作。用 Jquery 的链式调用更好。

比如：var str=$("a").attr("href");

for (var i = size; i < arr.length; i++) {}

for 循环每一次循环都查找了数组 (arr) 的.length 属性，在开始循环的时候设置一

个变量来存储这个数字，可以让循环跑得更快：

for (var i = size, length = arr.length; i < length; i++) {}

四十、 用原型链继承的方式写一个类和子类

```
function Person(name,age){
        this.name=name;
        this.age=age;
    }
    Person.prototype.study=function(){
        return "学习"
    }
    /*var p1 =new Person("张三",20);*/
    /*p1.study();*/
    function Student(class_,name,age){
        this.class_=class_;
        this.name=name;
        this.age=age;
    }
    Student.prototype=new Person();
    var s1 =new Student("二班","李大人",16);
    console.log(s1.name,s1.age,s1.class_,s1.study());
```

四十一、 编写一个方法求一个字符串的字节长度，假设：一个英文字符占用一个字节，一个中文字符占用两个字节

```
function num(str) {
    var num1 = str.length;
    var num2 = 0;
    for (var i = 0; i < str.length; i++) {
        if (str.charCodeAt(i) >= 10000) {
            num2++;
        }
    }
    console.log(num1 + num2)
}
```

四十二、 简单概括浏览器事件模型，如何获得资源 dom 节点

在各种浏览器中存在三种事件模型:原始事件模型(original event model),DOM2 事件模型,IE 事件模型.其中原始的事件模型被所有浏览器所支持,而 DOM2 中所定义的事件模型目前被除了 IE 以外的所有主流浏览器支持。

浏览器事件模型分为三个阶段

1、捕获阶段

2、目标阶段

3、冒泡阶段

Dom 节点获取方法：

1．通过 id 属性获取 document.getElementById()

2．通过 name 属性获取 document.getElementsByName()

3．通过标签名获取 document.getElementsByTagName()

4．通过 class 属性获取 document.getElementsByClassName()

5．原生 js 中的 querySelector 和 querySelectorAll 方法也同样可以获取到相应的 dom 节点，相似于 jquery，但比 jq 更快

四十三、 写一段 ajax 提交的 js 代码

```
var xhr =xhr();
function xhr(){
        if(window.XMLHttpRequest){
            return   window. XMLHttpRequest();
        }else if(window.ActiveXObject){
            try {
                return   new ActiveXObject("Microsoft.XMLHTTP");
            }catch (e) {
                try {
                    return   new ActiveXObject("Msxml2.XMLHTTP");
                }catch (ex) { }
            }
        }
        }
    xhr.open("get","url","true");
    xhr.onreadystatechange=function(){
        if (xhr.readyState==4 && (xhr.status==200||xhr.status==304)){
            document.getElementById("myDiv").innerHTML=xhr.responseText;
        }
```

```
}
        xhr.send();
}
```

四十四、 判断字符串是否是这样组成的，第一个必须是字母，后面可以是字母和数字、下划线，总长度为 5—20（请使用正则表达式）

```
function if_fit(str){
        var reg=/^[A—Za—z]{1}\w{5,20}/g;
            var result=str.search(reg);
        return result;
    }
```

四十五、 截取字符串 abcdefg 的 efg

```
var str="abcdefg";
console.log(str.slice(4));
```

四十六、 css 引入的方式有哪些，link 和@import 的区别是什么

有四种形式:

1.链入外部样式表,就是把样式表保存为一个样式表文件,然后在页面中用<link rel="stylesheet" type="text/css" href="*.css">链接这个样式表文件.

2.内部样式表,就是把样式表放到页面的<head>区里. <style type="text/css">

div {height: 600px;}

</style>

3.导入外部样式表,用@import,在<head>与</head>之间，<style type="text/css">

<!——

@import "*.css"

——>

</style>

4.内嵌样式,就是在标签内写入 style="",比如:

设置 div 背景色为灰色.

区别：

1）. link 是 XHTML 标签，除了加载 CSS 外，还可以定义 RSS 等其他事务；@import 属于 CSS 范畴，只能加载 CSS。

2）. link 引用 CSS 时，在页面载入时同时加载；@import 需要页面网页完全载入以

后加载。

3）. link 是 XHTML 标签，无兼容问题；@import 是在 CSS2.1 提出的，低版本的浏览器不支持。

4）. link 支持使用 Javascript 控制 DOM 去改变样式；而@import 不支持。

四十七、 将字符串 helloChina 反转输出

```
var str = "helloChina";
方法 1：console.log( str.split("").reverse().join("") )；');
方法 2：for (var x = str.length—1; x >=0; x——) {
document.write(str.charAt(x));
}
方法 3：var a=str.split("");
var rs = new Array;
while(a.length){
    rs.push(a.pop());
}
alert(rs.join(""));
```

四十八、 为什么无法定义 1px 左右高度的容器

IE6 下这个问题是因为默认的行高造成的，解决的方法也有很多，例如：overflow:hidden | zoom:0.08 | line—height:1px

四十九、 FireFox 中标签的居中问题的解决办法

*{margin：0px auto；}

五十、 请写出 XHTML 和 css 如何注释

XHTML:<!—— 注释内容——>

css:/* 注释内容*/

五十一、 现在想调节—下父元素的透明度，但是又不影响子元素的透明度，怎么破？

方法 1：用 RGBA

方法 2：再加—层与父元素同级的 div 装载子元素 定位到子元素原位置

五十二、 简述 ECMASCRIPT6 的新特性

1.增加块作用域

2.增加 let const

3.解构赋值

4.函数参数扩展 （函数参数可以使用默认值、不定参数以及拓展参数）

5.增加 class 类的支持

6.增加箭头函数

7.增加模块和模块加载（ES6 中开始支持原生模块化啦）

8.math, number, string, array, object 增加新的 API

五十三、 Apply 和 call 方法的异同

相同点:两个方法产生的作用是完全一样的，都是改变 this 指向，第一个参数都是对象；

不同点:

call()方法参数将依次传递给借用的方法作参数，即 fn.call(thisobj, arg1,arg2,arg3...argn)，有 n 个参数

apply()方法第一个参数是对象，第二个参数是数组 fn.apply(thisobj,arg)，此处的 arg 是一个数组,只有两个参数

五十四、 在 javascript 中什么是伪数组，如何将伪数组转化为标准数组

这里把符合以下条件的对象称为伪数组：

1，具有 length 属性

2，按索引方式存储数据

3，不具有数组的 push,pop 等方法

伪数组（类数组）：无法直接调用数组方法或期望 length 属性有什么特殊的行为，不具有数组的 push,pop 等方法，但仍可以对真正数组遍历方法来遍历它们。典型的是函数的 argument 参数，还有像调用 document.getElementsByTagName, document.childNodes 之类的,它们返回的 NodeList 对象都属于伪数组。

可以使用以下函数将伪数组转化为真正的 Array 对象（兼容问题处理）。

```
function    makeArray(c) {
      try{
     return Array.prototype.slice.call(c);
      }catch(e){
          var ret = [],i, len = c.length;
           for(i = 0; i < len; i++) {
                ret[i] = (c[i]);
           }
           return ret;
     }
}
```

五十五、 Js 和 native 交互的方法与问题

实现 JS 和 Native 交互有两种方式:

第一种：shouldOverrideUrlLoading(WebView view, String url）

通过给 WebView 加一个事件监听对象（WebViewClient)并重写 shouldOverrideUrlLoading(WebView view, String url）方法。当按下某个连接时 WebViewClient 会调用这个方法，并传递参数 view 和 url

第二种：JS 和 Java 互调

WebView 开启 JavaScript 脚本执行

WebView 设置供 JavaScript 调用的交互接口

客户端和网页端编写调用对方的代码

JS 调用 JAVA

```
JS : window.jsInterfaceName.methodName(parameterValues)
native: webView.addJavascriptInterface(new JsInteration(), “androidNative”);
```

Java 调用 JS

```
webView 调用 js 的基本格式为：
    webView.loadUrl(“javascript:methodName(parameterValues)”)
调用 js 无参无返回值函数：
String call = “javascript:sayHello()；webView.loadUrl(call);
调用 js 有参无返回值函数：
String call = “javascript:alertMessage(\”” + “content” + “\”)”;
webView.loadUrl(call);
调用 js 有参数有返回值的函数
```

Android 在 4.4 之前并没有提供直接调用 js 函数并获取值的方法，所以在此之前，常用的思路是 java 调用 js 方法，js 方法执行完毕，再次调用 java 代码将值返回。Android 4.4 之后使用 evaluateJavascript 即可。

```
private void testEvaluateJavascript(WebView webView) {
  webView.evaluateJavascript("getGreetings()", new ValueCallback<String>() {
  @Override
  public void onReceiveValue(String value) {
       Log.i(LOGTAG, "onReceiveValue value=" + value);
  }});
}
```

注:

参数类型如果是简单的 int 或 String，可以直接传，对于复杂的数据类型，建议以字符

串形式的 json 返回。

evaluateJavascript 方法必须在 UI 线程（主线程）调用，因此 onReceiveValue 也执行在主线程。

当 native 与 js 交互时存 cookie 看到很多人遇到过这样一个问题,cookie 存不进去,网上有很多解释方案,但是很多没说到重点上,这里直接贴一下代码:

```
public static void synCookies(Context context, String url, String version) {
        CookieSyncManager.createInstance(context);
        CookieManager cookieManager = CookieManager.getInstance();
        cookieManager.setAcceptCookie(true);
        cookieManager.removeAllCookie();
        cookieManager.setCookie(url,
"sessionKey="+UserInfoShareprefrence.getInstance(context).getLocalSession
Key())
        cookieManager.setCookie(url, "productVersion=android—epocket—v"
+ version);
        CookieSyncManager.getInstance().sync();
    }
```

存不进去的很大一部分原因是你的 url 不对 ,这里的 url 就是显示的 url 的域名,这里顺便贴出取域名的方法,给出的是通过正则提取域名

```
 /**
     * 获得域名
     * @param url
     * @return
 */
public static String getDomain(String url) {
        Pattern p = Pattern.compile("[^//]*?\\.(com|cn|net|org|biz|info|cc|tv)",
Pattern.CASE_INSENSITIVE);
        Matcher matcher = p.matcher(url);
        matcher.find();
        return matcher.group();
    }
```

还有一点就是,如果你想传递多个值给 cookie 的话,可以多次使用 setCookie,不要擅自的自己拼值,因为你拼的字符串中可能存在分号,内部多分号做了特殊处理,截取分号之前的,之后的直接放弃!

五十六、 用 sass 的 minix 定义一些代码片段，且可传参数

```
/**
 * @module 功能
```

```
 * @description 生成全屏方法
 * @method fullscreen
 * @version 1.7.0
 * @param {Integer} $z—index 指定层叠级别 <1.7.0>
 * @param {Keywords} $position 指定定位方式，取除`static | relative`之外的值，默认值：absolute <1.8.5>
 */
@mixin fullscreen($z—index: null, $position: absolute) {
    position: $position;
    z—index: $z—index;
    top: 0;
    right: 0;
    bottom: 0;
    left: 0;
}
```

五十七、 移动端经常出现的兼容问题，谈谈移动端应用或者 wap 站的一些优化技巧和心得

1、 安卓浏览器看背景图片，有些设备会模糊。

因为手机分辨率太小，如果按照分辨率来显示网页，字会非常小，安卓手机 devicePixelRatio 比较乱，有 1.5 的，有 2 的也有 3 的。想让图片在手机里显示更为清晰，必须使用 2x 的背景图来代替 img 标签（一般情况都是用 2 倍），或者指定 background—size:contain;都可以

2、 防止手机中网页放大和缩小

<meta name="viewport" content="width=device—width,initial—scale=1.0,maximum—scale=1.0,user—scalable=0" />

3、 apple—mobile—web—app—capable 是设置 Web 应用是否以全屏模式运行。

<meta name="apple—mobile—web—app—capable" content="yes">如果 content 设置为 yes，Web 应用会以全屏模式运行，反之，则不会。content 的默认值是 no，表示正常显示;也可以通过只读属性 window.navigator.standalone 来确定网页是否以全屏模式显示。

6. format—detection 启动或禁用自动识别页面中的电话号码。

语法：<meta name="format—detection" content="telephone=no">

默认情况下，设备会自动识别任何可能是电话号码的字符串。设置 telephone=no 可以禁用这项功能。

7. html5 调用安卓或者 ios 的拨号功能

html5 提供了自动调用拨号的标签，只要在 a 标签的 href 中添加 tel:就可以了。

如下：400—810—6999 转 1034

拨打手机号 如下点击拨打 15677776767

8. 上下拉动滚动条时卡顿、慢

```
body {
—webkit—overflow—scrolling: touch;
  overflow—scrolling: touch;
}
```

Android3+和 iOS5+支持 CSS3 的新属性为 overflow—scrolling

9. 禁止复制、选中文本

```
Element {
—webkit—user—select: none;
—moz—user—select: none;
—khtml—user—select: none;
   user—select: none;
}
```

解决移动设备可选中页面文本(视产品需要而定)

10. 长时间按住页面出现闪退

```
element {
—webkit—touch—callout: none;
}
```

11. iphone 及 ipad 下输入框默认内阴影

```
Element{
—webkit—appearance: none;
}
```

12. ios 和 android 下触摸元素时出现半透明灰色遮罩

```
Element {
—webkit—tap—highlight—color:rgba(255,255,255,0)
}
```

设置 alpha 值为 0 就可以去除半透明灰色遮罩，备注：transparent 的属性值在 android 下无效。详细介绍参照（http://www.jb51.net/post/phone_web_ysk）

13. active 兼容处理 即 伪类 :active 失效

方法一：body 添加 ontouchstart

```
<body ontouchstart="">
```

方法二：js 给 document 绑定 touchstart 或 touchend 事件

```
<style>
```

```
a {
  color: #000;
}
a:active {
  color: #fff;
}
</style>
<a herf=foo >bar</a>
<script>
  document.addEventListener('touchstart',function(){},false);
</script>
```

14. 动画定义 3D 启用硬件加速

```
Element {
—webkit—transform:translate3d(0, 0, 0)
  transform: translate3d(0, 0, 0);
}
```

注意：3D 变形会消耗更多的内存与功耗

15. Retina 屏的 1px 边框

```
Element{
  border—width: thin;
}
```

16. webkit mask 兼容处理

某些低端手机不支持 css3 mask，可以选择性的降级处理。

比如可以使用 js 判断来引用不同 class：

```
if( 'WebkitMask' in document.documentElement.style){
  alert('支持 mask');
} else {
  alert('不支持 mask');
}
```

17. 旋转屏幕时，字体大小调整的问题

```
html, body, form, fieldset, p, div, h1, h2, h3, h4, h5, h6 {
—webkit—text—size—adjust:100%;
}
```

18. transition 闪屏

/设置内嵌的元素在 3D 空间如何呈现：保留 3D /

—webkit—transform—style: preserve—3d;

/ 设置进行转换的元素的背面在面对用户时是否可见：隐藏 /

—webkit—backface—visibility:hidden;

19. 圆角 bug

某些 Android 手机圆角失效 background—clip: padding—box;

20. 顶部状态栏背景色

<meta name="apple—mobile—web—app—status—bar—style" content="black" />

说明：

除非你先使用 apple—mobile—web—app—capable 指定全屏模式，否则这个 meta 标签不会起任何作用。

如果 content 设置为 default，则状态栏正常显示。如果设置为 blank，则状态栏会有一个黑色的背景。如果设置为 blank—translucent，则状态栏显示为黑色半透明。如果设置为 default 或 blank，则页面显示在状态栏的下方，即状态栏占据上方部分，页面占据下方部分，二者没有遮挡对方或被遮挡。如果设置为 blank—translucent，则页面会充满屏幕，其中页面顶部会被状态栏遮盖住（会覆盖页面 20px 高度，而 iphone4 和 itouch4 的 Retina 屏幕为 40px）。默认值是 default。

21. 设置缓存

<meta http—equiv="Cache—Control" content="no—cache" />

手机页面通常在第一次加载后会进行缓存，然后每次刷新会使用缓存而不是去重新向服务器发送请求。如果不希望使用缓存可以设置 no—cache。

22. 桌面图标

<link rel="apple—touch—icon" href="touch—icon—iphone.png" />

<link rel="apple—touch—icon" sizes="76x76" href="touch—icon—ipad.png" />

<link rel="apple—touch—icon" sizes="120x120" href="touch—icon—iphone—retina.png" />

<link rel="apple—touch—icon" sizes="152x152" href="touch—icon—ipad—retina.png" />

iOS 下针对不同设备定义不同的桌面图标。

<link rel="apple—touch—icon—precomposed" href="touch—icon—iphone.png" />

图片尺寸可以设定为 5757（px）或者 Retina 可以定为 114114（px），ipad 尺寸为 72*72（px)

23. 启动画面

<link rel="apple—touch—startup—image" href="start.png"/>

iOS 下页面启动加载时显示的画面图片，避免加载时的白屏。

可以通过 madia 来指定不同的大小：

<!——iPhone——>

<link href="apple—touch—startup—image—320x460.png" media="(device—width:

320px)" rel="apple—touch—startup—image" />

<!—— iPhone Retina ——>

<link href="apple—touch—startup—image—640x920.png" media="(device—width: 320px) and (—webkit—device—pixel—ratio: 2)" rel="apple—touch—startup—image" />

<!—— iPhone 5 ——>

<link rel="apple—touch—startup—image" media="(device—width: 320px) and (device—height: 568px) and (—webkit—device—pixel—ratio: 2)" href="apple—touch—startup—image—640x1096.png">

<!—— iPad portrait ——>

<link href="apple—touch—startup—image—768x1004.png" media="(device—width: 768px) and (orientation: portrait)" rel="apple—touch—startup—image" />

<!—— iPad landscape ——>

<link href="apple—touch—startup—image—748x1024.png" media="(device—width: 768px) and (orientation: landscape)" rel="apple—touch—startup—image" />

<!—— iPad Retina portrait ——>

<link href="apple—touch—startup—image—1536x2008.png" media="(device—width: 1536px) and (orientation: portrait) and (—webkit—device—pixel—ratio: 2)" rel="apple—touch—startup—image" />

<!—— iPad Retina landscape ——>

<link href="apple—touch—startup—image—1496x2048.png"media="(device—width: 1536px) and (orientation: landscape) and (—webkit—device—pixel—ratio: 2)"rel="apple—touch—startup—image" />

24.浏览器私有及其它 meta

QQ 浏览器私有

全屏模式

<meta name="x5—fullscreen" content="true">

强制竖屏

<meta name="x5—orientation" content="portrait">

强制横屏

<meta name="x5—orientation" content="landscape">

应用模式

<meta name="x5—page—mode" content="app">

UC 浏览器私有

全屏模式

<meta name="full—screen" content="yes">

强制竖屏

<meta name="screen—orientation" content="portrait">

强制横屏

<meta name="screen—orientation" content="landscape">

应用模式

<meta name="browsermode" content="application">

其它(针对手持设备优化，主要是针对一些老的不识别 viewport 的浏览器，比如黑莓)

<meta name="HandheldFriendly" content="true">

微软的老式浏览器

<meta name="MobileOptimized" content="320">

windows phone 点击无高光

<meta name="msapplication—tap—highlight" content="no">

25. IOS 中 input 键盘事件 keyup、keydown、keypress 支持不是很好

用 input search 做模糊搜索的时候，在键盘里面输入关键词，会通过 ajax 后台查询，然后返回数据，然后再对返回的数据进行关键词标红。用 input 监听键盘 keyup 事件，在安卓手机浏览器中是可以的，但是在 ios 手机浏览器中变红很慢，用输入法输入之后，并未立刻响应 keyup 事件，只有在通过删除之后才能响应！

解决办法：

可以用 html5 的 oninput 事件去代替 keyup

```
<input type="text" id="testInput">
<script type="text/javascript">
  document.getElementById('testInput').addEventListener('input', function(e){
    var value = e.target.value;
  });
</script>
```

然后就达到类似 keyup 的效果！

26. h5 网站 input 设置为 type=number 的问题

一般会产生三个问题，一个问题是 maxlength 属性不好用了。另外一个是 form 提交的时候，默认给取整了。三是部分安卓手机出现样式问题。

问题一解决，用 js 如下

```
<input type="number" oninput="checkTextLength(this ,10)">
function checkTextLength(obj, length) {
        if(obj.value.length > length)  {
          obj.value = obj.value.substr(0, length);
        }
}
```

问题二，是因为 form 提交默认做了表单验证，step 默认是 1,要设置 step 属性，假如保留 2 位小数，写法如下：

<input type="number" step="0.01" />

关于 step：input 中 type=number，一般会自动生成一个上下箭头，点击上箭头默认增加一个 step，点击下箭头默认会减少一个 step。number 中默认 step 是 1。也就是 step=0.01,可以允许输入 2 位小数，并且点击上下箭头分别增加 0.01 和减少 0.01。

假如 step 和 min一起使用，那么数值必须在 min 和 max 之间。

问题三，去除 input 默认样式

input[type=number] {

—moz—appearance:textfield;

}

input[type=number]::—webkit—inner—spin—button,

input[type=number]::—webkit—outer—spin—button {

—webkit—appearance: none;

margin: 0;

}

27. ios 设置 input 按钮样式会被默认样式覆盖

解决方式如下：

input,

textarea {

border: 0;

—webkit—appearance: none;

}

设置默认样式为 none

28. IOS 键盘字母输入，默认首字母大写

解决方案，设置如下属性

<input type="text" autocapitalize="off" />

29. select 下拉选择设置右对齐

设置如下：

select option {

direction: rtl;

}

30. 通过 transform 进行 skew 变形，rotate 旋转会造成出现锯齿现象

可以设置如下：

—webkit—transform: rotate(—4deg) skew(10deg) translateZ(0);

transform: rotate(—4deg) skew(10deg) translateZ(0);

outline: 1px solid rgba(255,255,255,0)

31.移动端点击 300ms 延迟

300ms 导致用户体验不是很好，解决这个问题，我们一般在移动端用 tap 事件来取代 click 事件。推荐两个 js，一个是 fastclick，一个是 tap.js

关于 300ms 延迟，具体请看：http://thx.github.io/mobile/300ms—click—delay/

32.移动端点透问题

案例如下：

```
<div id="haorooms">点头事件测试</div>
<a href="www.jb51.net">www.jb51.net</a>
```

div 是绝对定位的蒙层,并且 z—index 高于 a。而 a 标签是页面中的一个链接，我们给 div 绑定 tap 事件：

```
$('#haorooms').on('tap',function(){
    $('#haorooms').hide();
});
```

我们点击蒙层时 div 正常消失，但是当我们在 a 标签上点击蒙层时，发现 a 链接被触发，这就是所谓的点透事件。

原因：

touchstart 早于 touchend 早于 click。即 click 的触发是有延迟的，这个时间大概在 300ms 左右，也就是说我们 tap 触发之后蒙层隐藏， 此时 click 还没有触发，300ms 之后由于蒙层隐藏，我们的 click 触发到了下面的 a 链接上。

解决：

（1）尽量都使用 touch 事件来替换 click 事件。例如用 touchend 事件(推荐)。

（2）用 fastclick，参考：https://github.com/ftlabs/fastclick

（3）用 preventDefault 阻止 a 标签的 click

（4）延迟一定的时间(300ms+)来处理事件 （不推荐）

（5）以上一般都能解决，实在不行就换成 click 事件。

下面介绍一下 touchend 事件，如下：

```
$("#haorooms").on("touchend", function (event) {
    event.preventDefault();
 });
```

33.关于 iOS 与 OS X 端字体的优化(横竖屏会出现字体加粗不一致等)

iOS 浏览器横屏时会重置字体大小，设置 text—size—adjust 为 none 可以解决 iOS 上的问题，但桌面版 Safari 的字体缩放功能会失效，因此最佳方案是将 text—size—adjust 为 100% 。

—webkit—text—size—adjust: 100%;

—ms—text—size—adjust: 100%;

text—size—adjust: 100%;

34. 关于 iOS 系统中，中文输入法输入英文时，字母之间可能会出现一个六分之一空格 可以通过正则去掉 this.value = this.value.replace(/\u2006/g, '');

35. 移动端 HTML5 audio autoplay 失效问题

这个不是 BUG，由于自动播放网页中的音频或视频，会给用户带来一些困扰或者不必要的流量消耗，所以苹果系统和安卓系统通常都会禁止自动播放和使用 JS 的触发播放，必须由用户来触发才可以播放。

解决方法思路：先通过用户 touchstart 触碰，触发播放并暂停（音频开始加载，后面用 JS 再操作就没问题了）。

解决代码：

```
document.addEventListener('touchstart', function () {
    document.getElementsByTagName('audio')[0].play();
    document.getElementsByTagName('audio')[0].pause();
});
```

38. 移动端 HTML5 input date 不支持 placeholder 问题

复制代码 代码如下:

```
<input placeholder="Date" class="textbox—n" type="text" onfocus="(this.type='date')"  id="date">
```

有的浏览器可能要点击两遍！

39. 部分机型存在 type 为 search 的 input，自带 close 按钮样式修改方法

有些机型的搜索 input 控件会自带 close 按钮（一个伪元素），而通常为了兼容所有浏览器，我们会自己实现一个，此时去掉原生 close 按钮的方法为

```
#Search::—webkit—search—cancel—button{
    display: none;
}
```

如果想使用原生 close 按钮，又想使其符合设计风格，可以对这个伪元素的样式进行修改。

40. 唤起 select 的 option 展开

zepto 方式:

```
$(sltElement).trrgger("mousedown");
```

原生 js 方式:

```
function showDropdown(sltElement) {
    var event;
    event = document.createEvent('MouseEvents');
    event.initMouseEvent('mousedown', true, true, window);
    sltElement.dispatchEvent(event);
```

};

五十八、 H5 中新增的单位 rem 是什么意思，和 em 的关系，以及 rem 在自适应布局中的应用方法

Rem 为单位：

rem 是相对于根元素<html>的“font—size”为基准。比如说我们给 html 设置 font—size 为 100px，

那么我们要给 html 中的 p 标签设置 16px 的字体，font—size 设置.16rem 就可以，在这里 16px=.16rem。

Em 为单位：

这种技术需要一个参考点，一般都是以<body>的“font—size”为基准。比如说我们使用“1em”等于“10px”来改变默认值“1em=16px”，这样一来，我们设置字体大小相当于“14px”时，只需要将其值设置为“1.4em”。

这个单位与 em 有什么区别呢？

区别在于使用 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。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。对于不支持它的浏览器，应对方法也很简单，就是多写一个绝对单位的声明。这些浏览器会忽略用 rem 设定的字体大小。

五十九、 如何实现浏览器内多个标签页之间的通信？

通过 WebSocket 或 SharedWorker 把客户端和服务器端建立 socket 连接，从而实现通信；也可以调用 localstorge、cookies 等本地存储方法。

六十、 假设现在页面里有一个 id 是 con 的 div，现在需要编写 js 代码，在页面加载完成后 将 div 的高度设置成 100px，宽度设置成 60px，并设置成灰色的 1px 的边框，背景设置成浅黄色。

```
window.onload=function(){
  var oDiv=document.getElementById("con");
  oDiv.style.height="100px";
  oDiv.style.width="60px";
  oDiv.style.width="1px solid gray";
  oDiv.style.backgroundColor="yellow";
}
```

六十一、 对新技术有那些了解，常去的网站有那些

node.js、angular.js、vue.js，reactjs，react-native，微信小程序

掘金、简书、github、csdn，知乎等

六十二、 用程序找出数组中出现次数超过一半的数字

思路：

1、 一个数字在数组中出现次数超过了一半，则排序后，位于数组中间的数字一定就是该出现次数超过了长度一半的数字（可以用反证法证明），也即是说，这个数字就是统计学上的中位数。最容易想到的办法是用快速排序对数组排序号后，直接取出中间的那个数字，这样的时间复杂度为 O(nlogn)，空间复杂度为 O(1)。

2 、事实上可以不用对数组进行排序，或者说仅部分排序，受快速排序的 partition 函数的启发，我们可以利用反复调用 partition 函数来求的该数字。我们现在数组中随机选取一个数字，而后通过 Partition 函数返回该数字在数组中的索引 index，如果 index 刚好等于 n/2，则这个数字便是数组的中位数，也即是要求的数，如果 index 大于 n/2，则中位数肯定在 index 的左边，在左边继续寻找即可，反之在右边寻找。这样可以只在 index 的一边寻找，而不用两边都排序，减少了一半排序时间。这种情况的平均时间复杂度大致为：T(n) = n+n/2+n/4+n/8+....+1，很明显当 n 很大时，T(n)趋近于 2n，也就是说平均情况下时间复杂度为 O(n),但是这种情况下，最坏的时间复杂度依然为 O(n*n)，最坏情况下，index 总是位于数组的最左或最右边，这样时间复杂度为 T(n) = n+n—1+n—2+n—3+....+1 = n(n—1)/2，显然，时间复杂度为 O(n*n)，空间复杂度为 O(1)。

六十三、 请设计一套方案，用于确保页面中 js 加载完全，对于优化某网页的加载速度，有什么独到见解

js 方法：

```
<script type="text/javascript">
window.onload=function(){
    var userName="xiaoming";
    alert(userName);
}
</script>
```

jquery 方法：

```
<script type="text/javascript">
$(document).ready(function(){
    var userName="xiaoming";
    alert(userName);
});
</script>
或者简写：
$(function(){
    var userName="xiaoming";
    alert(userName);
});
```

如何确定一个 js 是否加载完全或者页面中的所有 js 加载完全，具体办法如下：

```
function loadScript（ url, callback） {
    var script = document.createElement("script");
    script.type = "text/javascript";
    if (script.readyState) {
        script.onreadystatechange = function() {
            if (script.readyState == "loaded" || script.readyState == "complete")
{
                script.onreadystatechange = null;
                callback();
            }
        }
    } else {
        script.onload = function() {
            callback();
        }
    }
    script.src = url;
    document.getElementsByName("head")[0].appendChild(script);
}
```

如何让脚本的执行顺序按照你设定的顺序执行，使用嵌套的方式：

```
    loadScript("file1.js", function() {
        loadScript("file2.js", function() {
            loadScript("file3.js", function() {
                alert("All files are loaded");
            });
        });
    });
```

网页加载速度优化：

1、减少请求

最大的性能漏洞就是一个页面需要发起几十个网络请求来获取诸如样式表、脚本或者图片这样的资源，这个在相对低带宽和高延迟的移动设备连接上来说影响更严重。

CDNs（内容分发网络）把资源放在离用户地理位置更近的地方对解决这个问题能起到很大作用，但是比起获取请求，大量的请求对页面加载时间的影响更为严重，而且最近的发现表明，CDNs 对移动端用户的性能影响越来越低。

2、整合资源

对开发者来说，将 Javascript 代码和 CSS 样式放到公共的文件中供多个页面共享是

一种标准的优化方法，这个方法能很简单的维护代码，并且提高客户端缓存的使用效率。

在 Javascript 文件中，要确保在一个页面中相同的脚本不会被加载多次，当大团队或者多个团队合作开发的时候，这种冗余的脚本就很容易出现，你可能会对它的发生频率并不低感到非常吃惊。

Sprites 是 css 中处理图片的一项技术，Sprites 就是将多张图片整合到一个线性的网状的大图片中，页面就可以将这个大图片一次性获取回来并且做为 css 的背景图，然后使用 css 的背景定位属性展示页面需要的图片部分，这种技术将多个请求整合成一个，能显著地改善性能。

平稳地改进但是需要对资源有控制权限，根据开发者的网站不同权限，一些资源并不需要被整合起来（例如，一些由 CMS 生成的资源），还有，对于一些外部域引用的资源，强行整合可能会导致问题，马海祥提醒大家需要注意的是，整合资源对手机浏览器来说是一把双刃剑，整合资源确实会在首次访问减少请求，但是大的资源文件可能会导致缓存失效，所以，需要小心地使用各种技术整合资源，以达到优化本地存储的目的。

3、使用浏览器缓存和本地缓存

现在所有的浏览器都会使用本地资源去缓存住那些被 Cache—Control 或者 Expires 头标记的资源，这些头能标记资源需要缓存的时间，另外，ETag（实体标签）和 Last—Modified 头来标识当资源过期后是否需要重新请求，浏览器为了减少不必要的服务器请求，尽可能地从本地缓存中获取资源，并且将那些已经过期的、或者当缓存空间减小的时候将那些很久不用的资源进行清理，浏览器缓存通常包括图片，CSS，Javascript 代码，这些缓存能合理地提高网站的性能（比如为了支持后退和前进的按钮，使用一个单独的缓存来保存整个渲染的页面）。

移动浏览器缓存，通常是比桌面 PC 小的多，这就导致了缓存的数据会很经常被清理，HTML5 的缓存基于浏览器缓存提供了一个很好的替换方案，Javascript 的 localStorage 已经在所有主流的桌面和移动端浏览器上都实现了，使用脚本代码能简便地支持 HTML5 的 localStorage 操作，可以读写键值数据，每个域名大概有 5MB 的容量，虽然不同的移动浏览器上读写速度相差很大，但是 localStorage 大容量的缓存使得它很适合作为客户端的缓存，从 localStorage 获取资源明显快于从服务器上获取资源，而且在大多数移动设备上也比依靠缓存头或者浏览器的本地缓存更灵活可靠，这是移动浏览器比桌面 PC 更有优势的一个地方，在桌面 PC 上，本地缓存仍然优先使用标准的浏览器缓存，导致桌面 PC 本地缓存的性能落后于移动浏览器。

在此，马海祥要提醒各位一下：虽然 localStorage 的机制易于实现，但是它的一些控制机制却是非常复杂的，你需要考虑到缓存带给你的所有问题，比如缓存失效（什么时候需要删除缓存？），缓存丢失（当你希望数据在缓存中的时候它并不在怎么办？），还有当缓存满的时候你怎么办？

4、首次使用的时候在 HTML 中嵌入资源

HTML 的标准是使用链接来加载外部资源，这使得更容易在服务器上（或者在 CDN 上）操作更新这些资源，而不是在每个页面上修改更新这些资源，根据上文讨论的，这种模式也使得浏览器能从本地缓存而不是服务器上获取资源。

但是对还没有缓存到浏览器 localStorage 的资源来说，这种模式对网站的性能有负面的影响，一般来说，一个页面需要几十个单独的请求来获取资源从而渲染页面。

所以说，从性能的角度来说，如果一个资源没有很高的被缓存的几率的话，最好把它嵌入到页面的 HTML 中（叫 inlining），而不是使用链接外部，脚本和样式是支持内嵌

到 HTML 中的，但是图片和其他的二进制资源其实也是可以通过内嵌包含 base64 编码的文本来嵌入到 HTML 中的。

内嵌的缺点是页面的大小会变得非常大，所以对于 Web 应用来说，关键的是能够跟踪分析这个资源什么时候需要从服务端获取，什么时候已经缓存到客户端了。

另外，在第一次请求资源后必须能够使用代码在客户端缓存资源，因此，在移动设备上，使用 HTML5 localStorage 能很好地做到内嵌。

由于不知道用户是否已经访问过这个页面了，所以需要网站有机制能生成不同版本的页面。

5、使用 HTML5 服务端发送事件

Web 应用已经使用了各种从服务器上轮询资源的方法来持续地更新页面，HTML5 的 EventSource 对象和 Server—Sent 事件能通过浏览器端的 JavaScript 代码打开一个服务端连接客户端的单向通道，服务端可以使用这个写通道来发送数据，这样能节省了 HTTP 创建多个轮询请求的消耗。

这种方式比 HTML 的 WebSocket 更高效，WebSocket 的使用场景是，当有许多客户端和服务端的交互的时候（比如消息或者游戏），在全双工连接上建立一个双向通道。

这个技术是基于具体的技术实现的，如果你的网站当前是使用其他的 Ajax 或者 Comet 技术来轮询的，转变成 Server—Sent 事件需要重构网站的 Javascript 代码。

6、消除重定向

当用户在一个移动设备上访问桌面 PC 网站的时候，Web 网站应用通常读取 HTTP 的 user—agent 头来判断这个用户是否是来自移动设备的，然后应用会发送带有空 HTTP body 和重定向 HTTP 地址头的 HTTP 301（或者 302）请求，把用户重定向到网站的移动版本上去，但是这个额外的客户端和服务端的交互通常在移动网络上会消耗几百毫秒，因此，在原先的请求上传递移动的 web 页会比传递一个重定向的信息并让客户端再请求移动页面更快。

对于那些想要在移动设备上看桌面 PC 网站的用户来说，你可以在移动 web 页面上提供一个链接入口，这样也能同时表示你的网站是并不提倡这种行为的。

虽然这个技术在理论上是简单的，但是实际上并不易于实施，由于有些 m.sites 是宿主在其他地方的，所以许多网站会选择重定向到一个不同的服务器上，有的网站则是会在重定向请求的时候种植上 Cookie 告诉 Web 应用这个用户是在使用移动设备，这种方法可能对 web 应用来说更容易控制。

7、减少资源负载

关于移动端页面的大小问题，渲染小页面更快，获取小资源也更快，减小每个请求的大小通常不如减少页面请求个数那么显著地提高性能。

但是有些技术在性能方面，特别是在需要对带宽和处理器性能精打细算的移动设备环境下，仍然是能带来很大利益的。

8、压缩文本和图像

诸如 gzip 这样的压缩技术，依靠增加服务端压缩和浏览器解压的步骤，来减少资源的负载，但是，一般来说，这些操作都是被高度优化过了，而且测试表明，压缩对网站还是起到优化性能的作用的，那些基于文本的响应，包括 HTML，XML，JSON（Javascript Object Notation），Javascript，和 CSS 可以减少大约 70%的大小。

浏览器在 Accept—Encoding 请求头中申明它的解压缩技术，并且当它们接收到服务端返回的 Content—Encoding 响应头标示的时候，就会按照这个响应头自动做解压操作。

马海祥觉得这种方法的优点就是易于实现，如果设置正确的话，现在所有的 Web 服务器都支持压缩响应，但是，也有一些桌面 PC 的安全工具会将请求头中的 Accept—Encoding 头去掉，这样即使浏览器支持解压缩，用户也无法获取到压缩后的响应。

9、代码简化

简化通常是使用在脚本和样式文件中，删除一些不必要的字符，比如空格，换行符，或者注释等，不需要暴露给外部的命名就可以被缩短为一个或者两个字符，比如变量名，合适的简化资源通常在客户端不需要做任何其他的处理，并且平均减少 20%的资源大小，内嵌在 HTML 中的脚本和样式文件也是可以精简的，有很多很好的库来做精简化的操作，这些库一般也同时会提供合并多个文件这样减少请求数的服务（具体可查看马海祥博客《手机网站制作的常用方法及优化技巧》的相关介绍）。

简化带来的好处并不局限于减少带宽和延迟，对于那些移动设备上缓存无法保存的过大资源来说，也是很有改善的，Gzip 在这个方面并没有任何帮助，因为资源是在被解压后才被缓存起来的。

Google 的 Closure Compiler 已经难以置信地完成了理解和简化 Javascript 的工作，但是 CSS 的简化则没有那么容易，因为对不同浏览器来说有不同的 CSS 技术能迷惑 CSS 简化工具，然后让 CSS 简化后无法正常工作，马海祥提醒大家必须要注意的是，已经有这样的案例了，即使只是删除了不必要的字符，简化工作也有可能破坏页面，所以当你应用简化技术之后，请做一下完整的功能测试工作。

10、调整图片大小

图片通常是占用了 Web 页面加载的大部分网络资源，也占用了页面缓存的主要空间，小屏幕的移动设备提供了通过调整图片大小来加速传输和渲染图片资源的机会，如果用户只是在小的移动浏览器窗口中看图片的话，高分辨率的图片就会浪费带宽、处理时间和缓存空间。

为了加速页面渲染速度和减少带宽及内存消耗，可以动态地调整图片大小或者将图片替换为移动设备专用的更小的版本，不要依靠浏览器来将高分辨率的图片转换成小尺寸的图片，这样会浪费带宽。

另外一个方法是先尽快加载一个低分辨率的图片来渲染页面，在 onload 或者用户已经开始和页面交互以后将这些低分辨率的图片替换成为高分辨率的图片。

特别应用在高度动态化的网站是有优势的。

11、使用 HTML5 和 CSS 3.0 来简化页面

HTML5 包括了一些新的结构元素，例如 header，nav，article 和 footer，使用这些语义化的元素比传统的使用 div 和 span 标签能使得页面更简单和更容易解析，一个简单的页面更小加载更快，并且简单的 DOM（Document Object Model）代表着更快的 JavaScript 执行效率，新的标签能很快地应用在包括移动端的新浏览器版本上，并且 HTML5 设计让那些不支持它的浏览器能平稳过渡使用新标签。

HTML5 的一些表单元素提供了许多新属性来完成原本需要 javascript 来完成的功能，例如，新的 placeholder 属性用于显示在用户输入进入输入框之前显示的介绍性文字，autofocus 属性用于标示哪个输入框应当被自动定位。

也有一些新的输入框元素能不用依靠 Javascript 就可以完成一些通用的需求，这些新的输入框类型包括像 e一mail，URL，数字，范围，日期和时间这样需要复杂的用户交互和输入验证的元素，在移动浏览器上，当需要输入文本的时候，弹出的键盘通常是由特定的输入框类型来做选择的，不支持指定的输入类型的浏览器就会只显示一个文本框。

另外，只要浏览器支持内建的层次，圆角，阴影，动画，过渡和其他的图片效果，CSS 3.0 就能帮助你创建轻便简易的页面了，而这些图片效果原先是需要加载图片才能完成的，这样，这些新特性就能加速页面渲染了。

人工地做这些改动是非常复杂和耗时的，如果你使用 CMS，它可以帮你生成许多你不需要控制的 HTML 和 CSS（具体可查看马海祥博客《制作移动端手机网站过程中的 SEO 优化方法技巧》的相关介绍）。

12、延迟渲染"BELOW一THE一FOLD"内容

可以确定的是如果我们将不可见区域的内容延迟加载，那么页面就会更快地展现在用户面前，这个区域叫做"below the fold"，为了减少页面加载后需要重新访问的内容，可以将图片替换为正确的高宽所标记的<img>标签。

一些好的 Javascript 库可以用来处理这些 below一the一fold 延迟加载的图像。

13、延迟读取和执行的脚本

在一些移动设备上，解析 Javascript 代码的速度能达到 100 毫秒每千字节，许多脚本的库直到页面被渲染以后都是不需要的加载的，下载和解析这些脚本可以很安全地被推迟到 onload 事件之后来做。

例如，一些需要用户交互的行为，比如托和拽，都不大可能在用户看到页面之前被调用，相同的逻辑也可以应用在脚本执行上面，尽量将脚本的执行延迟到 onload 事件之后，而不是在初始化页面中重要的可被用户看到的内容的时候执行。

这些延迟的脚本可能是你自己写的，更重要的是，也有可能是第三方的，对广告、社交媒体部件、或者分析的差劲的脚本优化会导致阻塞页面的渲染，会增加珍贵的加载时间，当然，你需要小心地评估诸如 jquery 这样为移动网站设计的大型脚本框架，特别当你仅仅只是使用这些框架中的一些对象的时候更要小心评估。

许多第三方的框架现在提供延迟加载的异步版本的 API，开发者只需要将原先的逻辑转化到这个异步版本，一些 JavaScript 要做延迟加载会有些复杂，因为在 onload 之后执行这些脚本需要注意很多注意事项（例如，你有个脚本需要绑定到 onload 事件上，你需要做什么？如果你将脚本延迟到 onload 事件之后，就一定就会失去很多执行的时机）。

14、使用 Ajax 来增强进程

Ajax（Asynchronous JavaScript and XML）是一项使用 XHR（XMLHttpRequest）对象来从 Web 服务器上获取数据的技术，它并不需要更新正在运行的页面，Ajax 能更新页面上的某个部分而不需要重新构建整个页面，它通常用来提交用户的交互相应，但是也可以用来先加载页面的框架部分，然后当用户准备好浏览网页的时候再填充详细的内容。

尽管是这个名字，但是 XMLHttpRequest 并不强制要求你只能使用 XML，你可以通过调用 overrideMineType 方法来制定"application/json"类型来使用 json 替换 XML，使用 JSON.parse 会比使用原生的 eval()函数快了几乎两倍，并且更为安全。

同时，切记 Ajax 的返回响应也会得益于那些应用在普通的返回响应的优化技术上面，确保对你的 Ajax 返回响应使用了缓存头，简化，gzip 压缩，资源合并等技术。

由于这个技术是根据具体应用不同而不同的，所以很难量化，或许由于跨域问题，你需要使用 XHR2，这个技术能使用外部域的资源，从而能进行跨域的 XHR 请求。

15、根据网络状况进行适配处理

由于使用更多带宽会使用更多移动网络的费用，所以只有能检测网络的类型才能使用针对特定网络的优化技术。

例如，预加载未来使用到的请求是非常聪明的做法，但是如果用户的带宽很稀有，并且加载的有些资源是永远不会用到的话，这个技术就是不合理的了。

在 Android 2.2+，navigator.connection.type 属性的返回值能让你区分 Wifi 和 2G/3G/4G 网络，在 Blackberry 上，blackberry.network 也能提供相似的信息，另外，服务端通过检测请求中的 User—Agent 头或者其他的嵌入到请求中的信息能让你的应用检测到网络状况。

检测网络信息的 API 最近已经有所变化了，接口现在不是直接定义 Wi—Fi，3G 等网络状况，而是给出了带宽信息和诸如“非常慢，慢，快和非常快”这样的建议，有个属性能给出估计的 MB/s 值和一个“meterd”的 Boolean 值来表示它的可信度，但是对浏览器来说，很难根据这个来判断环境，判断当前网络环境然后适配仍然是一种最好的方法（具体可查看马海祥博客《百度移动搜索开放适配服务的 3 种方法》的相关介绍），但是这种方法正在被考虑被替换。

16、对多线程来说尽量使用 HTML5 的 WEB WORKER 特性

HTML5 中的 Web Worker 是使用多个线程并发执行 Javascript 程序，另外，这种特别的多线程实现能减少困惑开发者多年的，在其他平台上遇到的问题，例如，当一个线程需要改变一个正在被其他线程使用的资源该如何处理，在 Web Worker 中，子线程不能修改主用户界面（UI）线程使用的资源。

对提高移动站点的性能来说，Web Worker 中的代码很适合用来预处理用户完成进一步操作所需要的资源的，特别是在用户的带宽资源不紧缺的情况下，在低处理器性能的移动设备上，过多的预加载可能会干扰当前页面的 UI 响应，使用多线程代码，让 Web Worker 对象（并且尽可能使用 localStorage 来缓存数据）在另外一个线程中操作预加载资源，这样就能不影响当前的 UI 表现了。

要特别说明的是，Web Worker 只在 Android 2.0 以上的版本实现，而且 iphone 上的 ios5 之前的版本也不支持，在桌面 PC 上，总是落后的 IE 只在 IE 10 才支持 Web Worker。

虽然这项技术并不是非常难实现，但是对 Web Workers 来说，有一些限制需要强制遵守，Web Workers 不能进入到页面的 DOM，也不能改变页面上的任何东西，Web Worker 很适合那种需要后台计算和处理的工作。

17、将 CLICK 事件替换成 TOUCH 事件

在触摸屏设备上，当一个用户触碰屏幕的时候，onclick 事件并没有立即触发，设备会使用大约半秒（大多数设备差不多都是 300 毫秒）来让用户确定是手势操作还是点击操作，这个延迟会很明显地影响用户期望的响应性能，要使用 touchend 事件来替换才能解决，当用户触碰屏幕的时候，这个事件会立即触发。

为了要确保不会产生用户不期望的行为，你应该也要使用 touchstart 和 touchmove 事件，例如，除非同时有个 touchstart 事件在 button 上，否则不要判断 touchend 事件在

button 上就意味着点击行为，因为用户有可能从其他地方触碰开始，然后拖拽到 button 上触碰结束的，你也可以在 touchstart 事件之后使用 touchmove 事件来避免将 touchend 事件误判为点击，当然前提是需要假设拖拽的手势并不是预期产生点击行为。

另外，你也需要去处理 onclick 事件来让浏览器改变 button 的外观从而标识为已点击的状态，同时你也需要处理那些不支持 touch 事件的浏览器，为了避免代码在 touchend 和 onclick 代码中重复执行，你需要在确保用户触碰事件已经在 touchend 执行了之后，在 click 事件中调用 preventDefault 和 stopPropagation 方法。

这种技术需要更多工作才能在一个页面中增加和维护链接，touch 事件的代码必须考虑其他手势，因为替换 click 的还有可能是缩放或者敲击动作。

18、支持 SPDY 协议

应用层 HTTP 和 HTTPS 协议导致的一些性能瓶颈，使得不论是桌面还是移动端的网站都非常难受，在 2009 年，谷歌开始研发一种叫做 SPDY（谐意是“speedy”）的协议来替换已有的协议，这种协议宣称能突破这些限制，这个协议的目标是让多种浏览器和多种 Web 服务都能支持，所以这个协议是开源的，但是初步地，只有 Google 的 Chrome 浏览器（在版本 10 及之后的）和 google 的站点支持，一旦一个 Web 服务支持 SPDY，那么它上面的所有站点都可以和支持这个协议的浏览器使用 SPDY 进行交互，将 SPDY 应用在 25 个 top100 的 Internet 网站上，Google 收集到的数据是网站的速度会改善 27%到 60%不等。

SPDY 自动使用 gzip 压缩所有内容，和 HTTP 不同的是，它连 header 的数据也使用 gzip 压缩，SPDY 使用多线程技术让多个请求流或者响应流能共用一个 TCP 连接，另外 SPDY 允许请求设置优先级，比如，页面中心的视频会比边框的广告拥有更高的优先级。

或许 SPDY 中最变革性的发明就是流是双向的，并且可以由客户端或者服务端发起，这样能使得信息能推送到客户端，而不用由客户端发起第一次请求，例如，当一个用户第一次浏览一个站点，还没有任何站点的缓存，这个时候服务端就可以在响应中推送所有的请求资源，而不用等候每个资源被再次独立请求了，作为替换协议，服务端可以发送暗示给客户端，提示页面需要哪些资源，同时也允许由客户端来初始化请求。即使是使用后一种这样的方式也比让客户端解析页面然后自己发现有哪些资源需要被请求来得快。

虽然 SPDY 并没有对移动端有什么特别的设置，但是移动端有限的带宽就使得如果支持 SPDY 的话，SPDY 在减少移动网站的延迟是非常有用的。

依据网站和服务的环境来进行平稳操作或进一步考虑，Google 有一个 SPDY 模块支持 Apache2.2 – mod_spdy – 这个模块是免费的；但是 mod_spy 有线程上的问题，并且和 mod_php 协作并不是很好，所以要求你使用这个技术的时候要确保你的网站的正常运行。

六十四、 请实现鼠标点击任意标签，alert 该标签的名称（注意兼容性）

事件的有捕获类型和冒泡类型

在这里我们可以利用冒泡解决该问题

```
var el = document.getElementsByTagName('body');

el[0].onclick=function(event){

evt=event||window.event;

var selected=evt.target||evt.srcElement;

alert(selected.tagName);
```

六十五、 对 string 对象进行扩展，使其具有删除前后空格的方法

```
String.prototype.trim = function() {
  return      this.replace(/(^\s*)|(\s*$)/g, "");
}
```

六十六、 常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？

1)bootstrap, easy UI, highcharts 和 echarts， jqueryUI , jquery、angular.js， vue.js, reactjs 等。

2)前端开发工具：gulp webpack

3)轮播插件，拖拽插件

六十七、 用一句话概述您的优点，用一句话概述您的缺点

自由发挥

六十八、 描述下你对 js 闭包。面向对象、继承的理解

1）闭包理解：

个人理解：闭包就是能够读取其他函数内部变量的函数；

使用闭包主要是为了设计私有的方法和变量。闭包的优点是可以避免全局变量的污染，缺点是闭包会常驻内存，会增大内存使用量，使用不当很容易造成内存泄露。在 js 中，函数即闭包，只有函数才会产生作用域的概念

闭包有三个特性：

1.函数嵌套函数

2.函数内部可以引用外部的参数和变量

3.参数和变量不会被垃圾回收机制回收

闭包常见用途：

创建特权方法用于访问控制

事件处理程序及回调

2） 面向对象：

面向对象编程，即 OOP，是一种编程范式，满足面向对象编程的语言，一般会提供类、封装、继承等语法和概念来辅助我们进行面向对象编程。

参考：

http://www.ruanyifeng.com/blog/2010/05/object_oriented_javascript_encapsulation.html

3）继承：

对象继承分两种情况，一种是构造函数的继承，一种是原型（prototype）的继承：

1. 构造函数的继承，比较简单，只需要在子对象中添加代码：parent.apply(this, arguments);

关于原型的继承最优化的方法，利用空对象作为中介

2. 拷贝继承

可参考：

https://segmentfault.com/a/1190000002440502

http://blog.csdn.net/james521314/article/details/8645815

六十九、 你做的页面在哪些浏览器测试过？这些浏览器的内核分别是什么？

IE 内核浏览器：360，傲游，搜狗，世界之窗，腾讯 TT。

非 IE 内核浏览器：firefox opera safari chrome 。

IE 浏览器的内核 Trident、Mozilla 的 Gecko、Chrome 的 Blink（WebKit 的分支）、Opera 内核原为 Presto，现为 Blink；

七十、 写出几种 IE6 bug 的解决方法

1）png24 位的图片在 iE6 浏览器上出现背景，解决方案是做成 PNG8.也可以引用一段脚本处理.

2）IE6 双倍边距 bug：在该元素中加入 display:inline 或 display:block

3）像素问题 使用多个 float 和注释引起的 使用 dislpay:inline 一3px

4）超链接 hover 点击后失效 使用正确的书写顺序 link visited hover active

5）z一index 问题 给父级添加 position:relative

6）Min一height 最小高度 ！ Important 解决’ 7.select 在 ie6 下遮盖 使用 iframe 嵌套

7）为什么没有办法定义 1px 左右的宽度容器（IE6 默认的行高造成的，使用 over:hidden,zoom:0.08 line一height:1px）

七十一、 清楚浮动的几种方法，各自的优缺点

1、 父级 div 定义伪类:after 和 zoom

```
<style type="text/css">
        .div1{background:#000080;border:1px solid red;}
       .div2{background:#800080;border:1px solid red;height:100px;margin—top:10px}
        .left{float:left;width:20%;height:200px;background:#DDD}
        .right{float:right;width:30%;height:80px;background:#DDD}
        /*清除浮动代码*/
        .clearfloat:after{display:block;clear:both;content:"";visibility:hidden;height:0}
        .clearfloat{zoom:1}
</style>
<div class="div1 clearfloat">
      <div class="left">Left</div>
      <div class="right">Right</div>
</div>
<div class="div2">
        div2
</div>
```

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 转有属性)可解决 ie6,ie7 浮动问题。

优点：浏览器支持好，不容易出现怪问题（目前：大型网站都有使用，如：腾迅，网易，新浪等等）。

缺点：代码多，不少初学者不理解原理，要两句代码结合使用，才能让主流浏览器都支持。

建议：推荐使用，建议定义公共类，以减少 CSS 代码。

2、 父级 div 定义 overflow:hidden

```
<style type="text/css">
    .div1{background:#000080;border:1px solid red;/*解决代码*/width:98%;overflow:hidden}
   .div2{background:#800080;border:1px solid red;height:100px;margin—top:10px;width:98%}
    .left{float:left;width:20%;height:200px;background:#DDD}
```

```
    .right{float:right;width:30%;height:80px;background:#DDD}
</style>
<div class="div1">
    <div class="left">Left</div>
    <div class="right">Right</div>
</div>
<div class="div2">
    div2
</div>
```

原理：必须定义 width 或 zoom:1，同时不能定义 height，使用 overflow:hidden 时，浏览器会自动检查浮动区域的高度。

优点：简单，代码少，浏览器支持好。

缺点：不能和 position 配合使用，因为超出的尺寸的会被隐藏。

建议：只推荐没有使用 position 或对 overflow:hidden 理解比较深的朋友使用。

3、 结尾处加空 div 标签 clear:both

```
<style type="text/css">
    .div1{background:#000080;border:1px solid red}
    .div2{background:#800080;border:1px solid red;height:100px;margin—top:10px}
    .left{float:left;width:20%;height:200px;background:#DDD}
    .right{float:right;width:30%;height:80px;background:#DDD}
    /*清除浮动代码*/
    .clearfloat{clear:both}
</style>
<div class="div1">
    <div class="left">Left</div>
    <div class="right">Right</div>
    <div class="clearfloat"></div>
</div>
<div class="div2">
    div2
</div>
```

原理：添加一个空 div，利用 css 提高的 clear:both 清除浮动，让父级 div 能自动获取到高度。

优点：简单，代码少，浏览器支持好，不容易出现怪问题。

缺点：不少初学者不理解原理；如果页面浮动布局多，就要增加很多空 div，让人感

觉很不爽。

建议：不推荐使用，但此方法是以前主要使用的一种清除浮动方法。

4、 父级 div 定义 height

```
<style type="text/css">
        .div1{background:#000080;border:1px solid red;/*解决代码*/height:200px;}
        .div2{background:#800080;border:1px solid red;height:100px;margin—top:10px}
        .left{float:left;width:20%;height:200px;background:#DDD}
        .right{float:right;width:30%;height:80px;background:#DDD}
</style>
<div class="div1">
    <div class="left">Left</div>
    <div class="right">Right</div>
</div>
<div class="div2">
    div2
</div>
```

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。

优点：简单，代码少，容易掌握。

缺点：只适合高度固定的布局，要给出精确的高度，如果高度和父级 div 不一样时，会产生问题。

建议：不推荐使用，只建议高度固定的布局时使用。

5、 父级 div 定义 overflow:auto

```
<style type="text/css">
        .div1{background:#000080;border:1px solid red;
          /*解决代码*/width:98%;overflow:auto}
    .div2{background:#800080; border:1px solid red; height:100px; margin-top:10px;width:98%}
        .left{float:left;width:20%;height:200px;background:#DDD}
        .right{float:right;width:30%;height:80px;background:#DDD}
</style>
<div class="div1">
    <div class="left">Left</div>
    <div class="right">Right</div>
```

```
</div>
<div class="div2">
    div2
</div>
```

原理：必须定义 width 或 zoom:1，同时不能定义 height，使用 overflow:auto 时，浏览器会自动检查浮动区域的高度。

优点：简单，代码少，浏览器支持好。

缺点：内部宽高超过父级 div 时，会出现滚动条。

建议：不推荐使用，如果你需要出现滚动条或者确保你的代码不会出现滚动条就使用吧。

七十二、 Javascript 的 typeof 返回哪些数据类型，列举 3 种强制类型转换和 2 中隐式类型转换

1）返回数据类型

undefined

string

boolean

number

symbol(ES6)

Object

Function

2）强制类型转换

Number(参数) 把任何类型转换成数值类型。

parseInt(参数 1，参数 2) 将字符串转换成整数

parseFloat()将字符串转换成浮点数字

string(参数)：可以将任何类型转换成字符串

Boolean() 可以将任何类型的值转换成布尔值。

3）隐式类型转换

1.四则运算

加法运算符+是双目运算符，只要其中一个是 String 类型，表达式的值便是一个 String。

对于其他的四则运算，只有其中一个是 Number 类型，表达式的值便是一个 Number。

对于非法字符的情况通常会返回 NaN：

'1' * 'a' // => NaN，这是因为 parseInt(a)值为 NaN，1 * NaN 还是 NaN

2.判断语句

判断语句中的判断条件需要是 Boolean 类型，所以条件表达式会被隐式转换为 Boolean。 其转换规则同 Boolean 的构造函数。比如：

```
var obj = {};if(obj){
    while(obj);}
```

3.Native 代码调用

JavaScript 宿主环境都会提供大量的对象，它们往往不少通过 JavaScript 来实现的。 JavaScript 给这些函数传入的参数也会进行隐式转换。例如 BOM 提供的 alert 方法接受 String 类型的参数：

```
alert({a: 1});       // => [object Object]
```

七十三、 写出 3 个使用 this 的典型应用

1.

```
function Thing() { }
 Thing.prototype.foo = "bar";
 Thing.prototype.logFoo = function () {
       console.log(this.foo);
 }
 Thing.prototype.setFoo = function (newFoo) {
       this.foo = newFoo;
  }
  var thing1 = new Thing();
  var thing2 = new Thing();
 thing1.logFoo(); //logs "bar"
 thing2.logFoo(); //logs "bar"
 thing1.setFoo("foo");
 thing1.logFoo(); //logs "foo";
 thing2.logFoo(); //logs "bar";
 thing2.foo = "foobar";
 thing1.logFoo(); //logs "foo";
 thing2.logFoo(); //logs "foobar";
```

2.

```
function Thing1() { }
  Thing1.prototype.foo = "bar";
  function Thing2() {
      this.foo = "foo";
```

```
}
  Thing2.prototype = new Thing1();
  function Thing3() {}
  Thing3.prototype = new Thing2();
  var thing = new Thing3();
  console.log(thing.foo); //logs "foo"
```

3.

```
function Thing() {}
  Thing.prototype.foo = "bar";
  Thing.prototype.logFoo = function () {
        function doIt() {
             onsole.log(this.foo);
         }
        doIt.apply(this);
  }
 function doItIndirectly(method) {
method();
 }
var thing = new Thing();
doItIndirectly(thing.logFoo.bind(thing)); //logs bar
```

七十四、 对前端界面工程师这个职位是怎么理解的？它的前景怎样？

前端工程师属于一个比较新兴的技术，各种技术层出不穷，随着客户体验的重要性前端需要掌握的技能也越来越多，对前端的要求也越来越多，而且我们前端是最贴近用户的程序员，主要负责实现界面交互，提升用户体验，而且有了 Node.js，前端可以实现服务端的一些事情，针对服务器的优化、拥抱最新前端技术，除了掌握必要的技能还要掌握用户的心理，善于沟通。

前景：前景无疑是值得肯定的，也需要我们时刻关注最新的技术，这会是一个时刻都在学习的道路

七十五、 Eval 函数的作用

eval 可以将字符串生成语句执行，一般执行动态的 js 语句。

eval 的使用场合：有时候我们预先不知道要执行什么语句，只有当条件和参数给时才知道执行什么语句，这时候 eval 就派上用场了。

七十六、 标签上 title 与 alt 属性的区别是什么

title 是鼠标放上去的额外信息

alt 是图片不能正常显示的时候，用文字代替

七十七、 对 WEB 标准以及 w3c 的理解与认识？

Web 标准就是将页面的解构、表现和行为各自独立实现，w3c 对标注提出了规范化的要求

1.对结构的要求：（标签规范可以提高搜索引擎对页面的抓取效率，对 SEO 很有帮助）

1）标签字母要小写；

2）标签要闭合；

3）标签不允许随意嵌套。

2.对 css 和 js 的要求：

1）尽量使用外联 css 样式表和 js 脚本，使结构、表现和行为分成三块，符合规范，同时提高页面渲染速度，提高用户体验；

2）样式尽量少用行间样式表，使结构与表现分离，标签的 id 和 class 命名要做到见文知义，标签越少，加载越快，用户体验更高，代码维护更简单，便于改版；

3）不需要变动页面内容，便可提供打印版本而不需要复制内容，提高网站易用性

七十八、 Css 选择符有哪些？哪些属性可以继承？优先级算法如何计算？

1.id 选择器（ # myid）

2.类选择器（.myclassname）

3.标签选择器（div, h1, p）

4.相邻选择器（h1 + p）

5.子选择器（ul < li）

6.后代选择器（li a）

7.通配符选择器（ * ）

8.属性选择器（a[rel = "external"]）

9.伪类选择器（a: hover, li: nth — child）

可继承： font—size font—family color,;

不可继承 ：border padding margin width height

优先级就近原则，样式定义最近者为准；载入样式以最后载入的定位为准;

优先级为:!important > id > class > tag

important 比 内联优先级高

七十九、 请戳出 ie6/7 下特有的 cssbug

一：li 边距“无故”增加

设置 ul 的显示形式为*display:inline—block;即可，前面加*是只 针对 IE6/IE7 有效

二：IE6 不支持 min—height 属性，但它却认为 height 就是最小高度

使用 ie6 不支持但其余浏览器支持的属性!important。

三:Overflow:

在 IE6/7 中，overflow 无法正确的隐藏有相对定位 position:relative;的子元素。解决方法就是给外包容器.wrap 加上 position:relative;。

四：border：none 在 IE6 不起作用： 写成 border：0 就可以了

五:100%高度

在 IE6 下，如果要给元素定义 100%高度，必须要明确定义它的父级元素的高度，如果你需要给元素定义满屏的高度，就得先给 html 和 body 定义 height:100%;。

六:双边距 Bug

当元素浮动时，IE6 会错误的的把浮动方式的 margin 值双倍计算，给 float 的元素添加一个 display：inline

七：躲猫猫 bug

定义了:hover 的链接，当鼠标移到那些链接上时，在 IE6 下就会触发躲猫猫。

1.在（那个未浮动的）内容之后添加一个<span style=”clear: both;”>

2.触发包含了这些链接的容器的 hasLayout，一个简单的方法就是给其定义 height:1%;

八:IE6 绝对定位的元素 1px 间距 bug

当绝对定位的父元素或宽度为奇数时，bottom 和 right 会多出现 1px，

解决方案，针对 IE6 进行 hack 处理

八十、 如何将一个元素 600 毫秒的速度缓慢向上滑动显示？

如果需要在父元素底部向上，可以利用 margin—top 把子元素，挤下去，同事父元素设置隐藏，然后改变 margintop 的值也可以利用定来做，把子元素定位最下边

```
(function(){
        var oDiv = document.createElement('div');
        oDiv.style.width = '100px';
        oDiv.style.height = '100px';
        oDiv.style.backgroundColor = 'red';
        oDiv.style.position = 'absolute';
        oDiv.style.marginTop = 100 + 'px';
        document.body.appendChild(oDiv);
        var timer = setInterval(function(){
            var m = parseInt(oDiv.style.marginTop);
            if (m == 0 ) {
                clearInterval(timer);
                return;
            }
```

```
            oDiv.style.marginTop = parseInt(oDiv.style.marginTop) — 1 + 'px';
        },600);
    })();
```

八十一、 写一个获取非行间样式的函数

```
Function getStyle(obj, attr){
    If(obj.currentStyle){
        return obj.currentStyle[attr];
    }else{
        return getComputedStyle(obj,false)[attr];
    }
}
```

八十二、 请用正则表达式验证数字

`/[0—9]*$/`

八十三、 为什么利用多个域名来提供网站资源会更有效？

1) 突破浏览器的并发限制（浏览器同一域名最大的并发请求数量为 6 个，ie6 为 2 个）
2) 节约 cookie 带宽
3) CDN 缓存更方便
4) 防止不必要的安全问题（尤其是 cookie 的隔离尤为重要）
5) 节约主机域名连接数，优化页面响应速度

八十四、 你如何从浏览器的 URL 中获取参数信息

浏览器宿主环境中，有一个 location 对象，同时这个对象也是 window 对象和 document 对象的属性。

location 对象中提供了与当前窗口加载的文档有关的的信息，即 URL 信息。

如 https://www.baidu.com/api/sousu?search=baidu&id=123#2

location.href: 完整 URL

location.protocol: 返回协议（https:）

location.host: 返回服务器名称和端口号（www.baidu.com）

location.hostname: 返回服务器名称（www.baidu.com）

location.port:返回服务器端口号（http 默认 80，https 默认 443）

location.pathname:返回 URL 中的目录和文件名（api/sousu）

location.search:返回查询字符串（?search=baidu&id=123#2）

location.hash:返回 hash 值（#2）

八十五、 手机端文字大小用什么单位

对于只需要适配少部分手机设备，且分辨率对页面影响不大的，使用 px 即可

对于需要适配各种移动设备，使用 rem，例如只需要适配 iPhone 和 iPad 等分辨率差别比较挺大的设备

八十六、 是否做过有上百图层的 psd 切图？ps 隐藏其他图层，只显示其中一个图层的快捷键

Alt + 当前图层前眼睛

八十七、 浏览器标准模式和怪异模式之间的区别是什么？

这是个历史遗留问题，W3C 标准推出前，旧的页面都是根据旧的渲染方式对页面进行渲染的，因此在 W3C 标准推出后为了保证旧页面的正常显示，保持浏览器的兼容性，这样浏览器上就产生了能够兼容 W3C 标准渲染的严格模式和保证旧页面显示的怪异模式的标准兼容模式。

具体表现：

1.在严格模式中 ：width 是内容宽度，元素真正的宽度 = margin—left + border—left —width + padding—left + width + padding—right + border—right— width + margin—right;

在怪异模式中 ：width 则是元素的实际宽度，内容宽度 = width — (padding—left + padding—right + border—left—width + border—right—width)

2）可以设置行内元素的高宽

在标准模式下，给 span 等行内元素设置 wdith 和 height 都不会生效，而在怪异模式下，则会生效。

3）可设置百分比的高度

在标准模式下，一个元素的高度是由其包含的内容来决定的，如果父元素没有设置高度，子元素设置一个百分比的高度是无效的。

4）用 margin:0 auto 设置水平居中在 IE 下会失效

使用 margin:0 auto 在标准模式下可以使元素水平居中，但在怪异模式下却会失效，怪异模式下的解决办法，用 text—align 属性:

body{text—align:center};#content{text—align:left}

5）怪异模式下设置图片的 padding 会失效

6）怪异模式下 Table 中的字体属性不能继承上层的设置

7）怪异模式下 white—space:pre 会失效

八十八、 Javascript 同源策略

同源策略是 Javascript 重要的安全度量标准。它最早出自 Netscape Navigator2.0，其目的是防止某个文档或脚本从多个不同源装载。所谓的同源就是同协议，同主机名，同端口号。

它的精髓很简单：它认为自任何站点装载的信赖内容是不安全的。当被浏览器半信半疑的脚本运行在沙箱时，它们应该只被允许访问来自同一站点的资源，而不是那些来自其它站点可能怀有恶意的资源。

八十九、 为什么要有同源限制？

我们举例说明：比如一个黑客程序，他利用 Iframe 把真正的银行登录页面嵌到他的页面上，当你使用真实的用户名，密码登录时，他的页面就可以通过 Javascript 读取到你的表单中 input 中的内容，这样用户名，密码就轻松到手了。

缺点：

现在网站的 JS 都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被 merge 后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节。

九十、 了解响应式布局吗？请大体说一说

响应式布局概念：Responsive design，意在实现不同屏幕分辨率的终端上浏览网页的不同展示方式。通过响应式设计能使网站在手机和平板电脑上有更好的浏览阅读体验。

设计步骤：

1．设置 meta 标签

2．根据媒体查询设置样式

3．设置多种视图宽度

4．注意点：

5．宽度使用百分比

6．处理图片缩放问题

九十一、 身为以为 web 前端工程师，你肯定知道现在最流行的前端技术吧，有那些？

Vuejs2.0/Angular2.0/React Native /es6//Nodejs

http2

gulp/webpack

九十二、 请简述为什么要使用数据库的事务

数据库事务(Database Transaction) ，是指作为单个逻辑工作单元执行的一系列操作，要么完全地执行，要么完全地不执行。

原子性（Atomic）（Atomicity)

事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。通常，与某个事务关联的操作具有共同的目标，并且是相互依赖的。如果系统只执行这些操作的一个子集，则可能会破坏事务的总体目标。原子性消除了系统处理操作子集的可能性。

一致性（Consistent）(Consistency)

事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如 B 树索引或双向链表）都必须是正确的。某些维护一致性的责任由应用程序开发人员承担，他们必须确保应用程序已强制所有已知的完整性约束。例如，当开发用于转帐的应用程序时，应避免在转帐过程中任意移动小数点。

隔离性（Insulation）(Isolation)

由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。事务查看数据时数据所处的状态，要么是另一并发事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看中间状态的数据。这称为隔离性，因为它能够重新装载起始数据，并且重播一系列事务，以使数据结束时的状态与原始事务执行的状态相同。当事务可序列化时将获得最高的隔离级别。在此级别上，从一组可并行执行的事务获得的结果与通过连续运行每个事务所获得的结果相同。由于高度隔离会限制可并行执行的事务数，所以一些应用程序降低隔离级别以换取更大的吞吐量。

持久性（Duration）(Durability）

事务完成之后，它对于系统的影响是永久性的。该修改即使出现致命的系统故障也将一直保持。

九十三、 聊一聊前端存储。

老朋友 cookie

短暂的 sessionStorage

简易强大的 localStorage

websql 与 indexeddb

详细参见：https://segmentfault.com/aZ1190000005927232

九十四、 BFC

w3c 规范中的 BFC 定义：

浮动元素和绝对定位元素，非块级盒子的块级容器（例如 inline—blocks, table—cells, 和 table—captions),以及 overflow 值不为“visiable”的块级盒子，都会为他们的内容创建新的 BFC(块级格式上下文）。

在 BFC 中，盒子从顶端开始垂直地一个接一个地排列，两个盒子之间的垂直的间隙是由他们 的 margin 值所决定的。在一个 BFC 中，两个相邻的块级盒子的垂直外边距会产生折叠。

在 BFC 中，每一个盒子的左外边缘（margin—left)会触碰到容器的左边缘(border—left)(对于 从右到左的格式来说，则触碰到右边缘）。

BFC 的通俗理解：

首先 BFC 是一个名词，是一个独立的布局环境，我们可以理解为一个箱子（实际上是看不见 摸不着的），箱子里面物品的摆放是不受外界的影响的。转换为 BFC 的理解则是：BFC 中的 元素的布局是不受外界的影响（我们往往利用这个特性来消除浮动元素对其非浮动的兄弟元 素和其子元素带来的影响。）并且在一个 BFC 中，块盒与行盒（行盒由一行中所有的内联元 素所组成）都会垂直的沿着其父元素的边框排列。

详细参见：

http://www.w3cplus.com/css/understanding—bfc—and—margin—collapse.html

https://www.zhihu.com/question/28433480

————————————前端工程化————————

九十五、 场景：你是第一天来公司上班的，项目代码托管在 GitLab，项目地址：git@lab.com:org/project.git，现在有一处代码需要你修改。

请完成此项任务中，与 git/gitlab 相关的操作步骤。

第一步：$> ssh—keygen —t rss —C zhangsan@abc.com

第二步：拷贝公钥到 gitlab

第三步：

$> git config —global user.name zhangsan

$> git config —global user.email zhangsan@abc.com

第四步：$> git clone git@lab.com:org/project.git

第五步：$> git checkout —b project—20170227—zhangsan—bugfix

第六步：修改代码

第七步：git status

第八步：git add .

第九步：git commit —am ‘bugfix’

第十步：

git push ——set—upstream origin project—20170227—zhangsan—bugfix

九十六、 CSS，JS 代码压缩，以及代码 CDN 托管，图片整合

CSSJS 代码压缩：

可以应用 gulp 的 gulp—uglify, gulp—minify—css 模块完成；可以应用 webpack 的 UglifyJsPlugin 压缩插件完成。

CDN:

内容分发网络(CDN)是一个经策略性部署的整体系统，包括分布式存储、负载均衡、网络请 求的重定向和内容管理 4 个要件。主要特点有：本地 Cache 加速，镜像服务，远程加速，带 宽优化。关键技术有：内容发布，内容路由，内容交换，性能管理。CDN 网站加速适合以 咨询为主的网站。CDN 是对域名加速不是对网站服务器加速。CDN 和镜像站比较不需要访 客手动选择要访问的镜像站。CDN 使用后网站无需任何修改即可使用 CDN 获得加速效果。

如果通过 CDN 后看到的网页还是旧网页，可以通过 URL 推送服务解决，新增的网页和图片 不需要 URL 推送。使用动态网页可以不缓存即时性要求很高的网页和图片。CDN 可以通过 gi 域 SVN 来管理。

图片整合

减少网站加载时间的最有效的方式之一就是减少网站的 HTTP 请求数。实现这一目标的一个 有效的方法就是通过 CSS Sprites ——将多个图片整合到一个图片中，然后再用 CSS 来定 位。缺点是可维护性差。可以使用百度的 fis/webpack 来自动化管理 sprite。

九十七、 如何利用 webpack 把代码上传服务器以及转码测试?

代码上传：

可以使用 sftp—webpack—plugin,但是会把子文件夹给提取出来，不优雅。可以使用 gulp +webpack 来实现。

转码测试

webpack 应用 babel 来对 ES6 转码，开启 devtool: “source—map"来进行浏览器测试。应用 karma 或 mocha 来做单元测试。

九十八、 项目上线流程是怎样的？

流程建议

—模拟线上的开发环境

本地反向代理线上真实环境开发即可。（apache, nginx, nodejs 均可实现）

—模拟线上的测试环境

模拟线上的测试环境，其实是需要—台有真实数据的测试机，建议没条件搭 daily 的，就直接 用线上数据测好了，只不过程序部分走你们的测试环境而已，有条件搭 daily 最好。

—可连调的测试环境

可连调的测试环境，分为 2 种。—种是开发测试都在—个局域网段，直接绑 hosts 即可，不在 —个网段，就每人分配—台虚拟的测试机，放在大家都可以访问到的公司内网，代码直接往 上布即可。

—自动化的上线系统

自动化的上线系统，可以采用 Jenkins。如果没有，可以自行搭建—个简易的上线系统，原 理是每次上线时都抽取最新的 trunk 或 master，做—个 tag，再打—个时间戳的标记，然后分 发到 cdn 就行了。界面里就 2 个功能，打 tag,回滚到某 tag,部署。

—适合前后端的开发流程

开发流程依据公司所用到的工具，构建，框架。原则就是分散独立开发，互相不干扰，连调 时有 hosts 可绑即可。

简单的可操作流程

—代码通过 git 管理，新需求创建新分支，分支开发，主干发布 —上线走简易上线系统，参见上—节

—通过 gulp+webpack 连到发布系统，—键集成，本地只关心原码开发

—本地环境通过 webpack 反向代理的 server

—搭建基于 linux 的本地测试机，自动完成 build+push 功能

九十九、 工程化怎么管理的？

前端工程化可以自动化处理—些繁复的工作，提高开发效率，减少低级错误。

目前前端构建工具很多，综合比较来看，gulp 相对来说更灵活，可以做更多的定制化任务，而 webpack 在模块化方面更完美—些

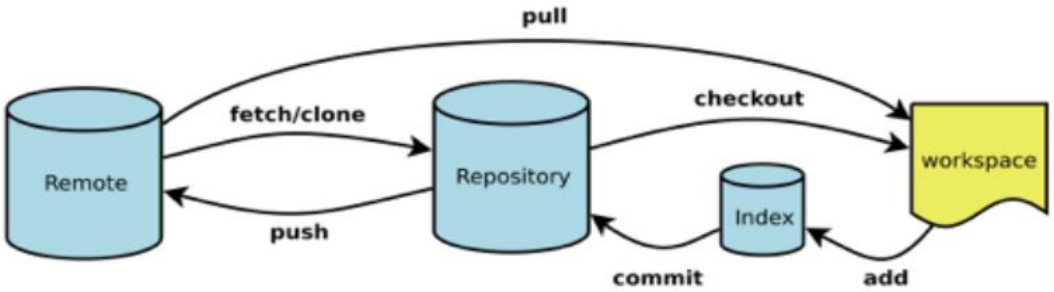
gulp 打造前端工程化方案，同时引入 webpack 来管理模块化代码，大致分工如下：

gulp：处理 html 压缩/预处理/条件编译，图片压缩，精灵图自动合并等任务

webpack：管理模块化，构建 js/css。

具体流程可参考： http://blog.csdn.net/java_goodstudy/article/details/52797322

一百、 git 常用命令



Workspace:工作区

Index/Stage:暂存区

Repository:仓库区（或本地仓库）

Remote:远程仓库

git init; (# 在当前目录新建一个 Git 代码库)

git add （添加当前目录的所有文件到暂存区）

git rm –cached [file] (# 停止追踪指定文件，但该文件会保留在工作区)

git commit [file1] [file2] -m [message] (# 提交暂存区的指定文件到仓库区)

git branch -a (列出所有本地分支和远程分支)

git checkout [branch-name] (# 切换到指定分支，并更新工作区)

git status(# 显示有变更的文件)

git push [remote] -tags (# 提交所有 tag)

详细参见：http://www.ruanyifeng.com/blog/2015/12/git—cheat—sheet.html

一百〇一、git 与 svn 的区别

git 是分布式的，svn 不是。

git 跟 svn一样有自己的集中式版本库或服务器。但 git 更倾向于被使用于分布式模式，克隆版本库后即使没有网络也能够 commit 文件，查看历史版本记录，创建项目分支等，等网络再次连接上 Push 到服务器端。

git 把内容按元数据方式存储，而 svn 是按文件。

所有的资源控制系统都是把文件的元信息隐藏在一个类似.svn,.cvs 等的文件夹里。

git 目录是处于你的机器上的一个克隆版的版本库，它拥有中心版本库上所有的东西，例如标签，分支，版本记录等。

git 没有一个全局的版本号，svn 有。

git 的内容完整性优于 svn。因为 git 的内容存储使用的是 SHA-1 哈希算法。

git 可以有无限个版本库，svn 只能有一个指定中央版本库。

当 svn 中央版本库有问题时，所有工作成员都一起瘫痪直到版本库维修完毕或者新的版本库设立完成。

每一个 git 都是一个版本库，区别是它们是否拥有活跃目录（Git Working Tree）。如果主要版本库（例如：置於 GitHub 的版本库）有问题，工作成员仍然可以在自己的本地版本库（local repository）提交，等待主要版本库恢复即可。工作成员也可以提交到其他的版本库！

一百〇二、webpack 和 gulp 对比

Gulp 就是为了规范前端开发流程，实现前后端分离、模块化开发、版本控制、文件合并与 压缩、mock 数据等功能的一个前端自动化构建工具。说的形象点，“Gulp 就像是一个产品的 流水线，整个产品从无到有，都要受流水线的控制，在流水线上我们可以对产品进行管 理。”另外，Gulp 是通过 task 对整个开发过程进行构建。

Webpack 是当下最热门的前端资源模块化管理和打包工具。它可以将许多松散的模块按照 依赖和规则打包成符合生产环境部署的前端资源。还可以将按需加载的模块进行代码分隔， 等到实际需要的时候再异步加载。通过 loader 的转换，任何形式的资源都可以视作模块，比 如 CommonJs 模块、AMD 模块、ES6 模块、CSS、图片、JSON、Coffeescript、LESS 等。

Gulp 和 Webpack 功能实现对比：从基本概念、启动本地 Server、sass/less 预编译、模块化 开发、文件合并与压缩、mock 数据、版本控制、组件控制八个方面对 Gulp 和 Webpack 进行对比。

详细参见：http://www.tuicool.com/articles/e632EbA

一百〇三、webpack 打包文件太大怎么办？

webpack 把我们所有的文件都打包成一个 JS 文件，这样即使你是小项目，打包后的文件也 会非常大。可以从去除不必要的插件，提取第三方库，代码压缩，代码分割，设置缓存几个 方面着手优化。

详细参见：http://www.jianshu.com/p/a64735eb0e2b

一百〇四、谈谈你对 webpack 的看法

WebPack 是一个模块打包工具，你可以使用 WebPack 管理你的模块依赖，并编绎输出模块们所需的静态文件。它能够很好地管理、打包 Web 开发中所用到的 HTML、JavaScript、CSS 以及各种静态文件（图片、字体等），让开发过程更加高效。对于不同类型的资源，webpack 有对应的模块加载器。webpack 模块打包器会分析模块间的依赖关系，最后 生成了优化且合并后的静态资源。

webpack 的两大特色：

1.code splitting（可以自动完成）

2.loader 可以处理各种类型的静态文件，并且支持串联操作

webpack 是以 commonJS 的形式来书写脚本滴，但对 AMD/CMD 的支持也很全面，方便旧项目进行代码迁移。

webpack 具有 requireJs 和 browserify 的功能，但仍有很多自己的新特性：

1. 对 CommonJS 、 AMD 、ES6 的语法做了兼容

2. 对 js、css、图片等资源文件都支持打包

3. 串联式模块加载器以及插件机制，让其具有更好的灵活性和扩展性，例如提供对 CoffeeScript、ES6 的支持

4. 有独立的配置文件 webpack.config.js

5. 可以将代码切割成不同的 chunk，实现按需加载，降低了初始化时间

6. 支持 SourceUrls 和 SourceMaps，易于调试

7. 具有强大的 Plugin 接口，大多是内部插件，使用起来比较灵活

8.webpack 使用异步 IO 并具有多级缓存。这使得 webpack 很快且在增量编译上更加快

一百〇五、说说你对 AMD 和 Commonjs 的理解

CommonJS 是服务器端模块的规范，Node.js 采用了这个规范。CommonJS 规范加载模块是同步的，也就是说，只有加载完成，才能执行后面的操作。AMD 规范则是非同步加载模块，允许指定回调函数。

AMD 推荐的风格通过返回一个对象做为模块对象，CommonJS 的风格通过对 module.exports 或 exports 的属性赋值来达到暴露模块对象的目的。

一百〇六、不想让别人盗用你的图片，访问你的服务器资源该怎么处理？

目前常用的防盗链方法主要有两种：

（1） 设置 Referer:适合不想写代码的用户，也适合喜欢开发的用户（Referer 是 HTTP 协议中的请求头，在跨页面访问的时候会带上。需要看看浏览器请求的 Referer 是 http:// 还是 https://，一般是 http://）

（2） 签名 URL:适合喜欢开发的用户

详细参见：https://yq.aliyun.com/articles/57931

一百〇七、精灵图和 base64 如何选择？

css 精灵，用于一些小的图标不是特别多，一个的体积也稍大，比如大于 10K (这个没有严 格的界定）。

base64,用于小图标体积较小（相对于 css 精灵），多少都无所谓。字体图标，用于一些别 人做好的图标库（也有少数自己去做的）用起来比较方便，他的图标只能用于单色，图标用 只能于一种颜色。

一百〇八、webpack 怎么引入第三方的库？

拿 jQuery 为例：

```
entry: {
  page: 'path/to/page.js',
  jquery: 'node—modules/jquery/dist/jquery.min.js'
}
```

```
new HtmlWebpaekPlugin({
 filename: 'index.html',
 template: 'index.html',
 inject: true,
 chunks: ['jquery','page'] //  按照先后顺序插入  script  标签
})
```

一百〇九、如果线上出现 buggit 怎么操作？

方法 **1**：在当前主分支修改 bug，暂存当前的改动的代码，目的是让工作空间和远程代码一致：

Git stash

修改完 bug 后提交修改：

git add .

git commit —m "fix bug 1"

git push

从暂存区把之前的修改恢复，这样就和之前改动一样了

git stash pop

这时可能会出现冲突，因为你之前修改的文件，可能和 bug 是同一个文件，如果有冲突会提示：

Auto—merging xxx.Java

CONFLICT (content): Merge conflict in xxx.java

前往 xxx.java 解决冲突

注意 stash pop 意思是从暂存区恢复到工作空间，同时删除此条暂存记录。

方式 **2**：拉一个新分支，老司机都推荐这样做，充分利用了 git 特性，先暂存一下工作空间改动：

git stash

新建一个分支，并且换到这个新分支

git branch fix_bug //新建分支

git checkout fix_bug //切换分支

这时候就可以安心的在这个 fix_bug 分支改 bug 了，改完之后：

git add .

git commit —m "fix a bug"

切换到 master 主分支

git checkout master

从 fix_bug 合并到 master 分支

git merge fix_bug

提交代码

git push

然后从暂存区恢复代码

git stash pop

此时如有冲突，需要解决冲突

一百一十、用过 Nginx 吗？都用过哪些？

nginx 是一个高性能的 HTTP 和反向代理服务器。

常使用场景：

(1) 反向代理

(2) 网站负载均衡

详细参见：http://www.cnblogs.com/hobinly/p/6023883.html

移动端布局与适配

一百一十一、 iscroll 安卓低版本卡顿，如何解决？

方案一：iScrollv5.1.3 设置 momentum: true

方案二：配置 probeType

方案三：开启硬价加速：给 scroll 元素增加 css 样式：—webkit—transform:translate3d(0,0,0);

方案四：判断手机版系统版本，应用原生 CSS: overflow:scroll—y

一百一十二、 移动布局自适应不同屏幕的几种方式

(1) 响应式布局

(2) 100%布局（弹性布局）

(3) 等比缩放布局（rem)

一百一十三、 请说下移动端常见的适配不同屏幕大小的方法？

响应式布局

简而言之，就是页面元素的位置随着屏幕尺寸的变化而变化，通常会用百分比来定位，而在设计上需要预留一些可被“压缩”的空间。

Cover 布局

就跟 background—size 的 cover 属性一样，保持页面的宽高比，取宽或高之中的较小者占满屏幕，超出的内容会被隐藏。此布局适用于主要内容集中在中部，边沿无重要内容的设计。

Contain 布局

同样，也跟 background—size 的 contain 属性那样，保持页面的宽高比，取宽或高之

中的较大者占满屏幕，不足的部分会用背景填充。个人比较推荐用这种方式，但在设计上需要背景为单色，或者是可平铺的背景。

一百一十四、 你们做移动端平时在什么浏览器上测试？

Chrome，Safari，微信 X5， UC，其他手机自带浏览器

一百一十五、 说说移动端是如何调试的？

移动端调试：

（1） 模拟手机调试

（2） 真机调试之 android 手机+Chrome

（3） 真机调试之 iphone+ safari (4)UC 浏览器

（1） 微信内置浏览器调试

（2） debuggap

（3） 抓包

详细参考：https://segmentfault.eom/a/1190000005964730

一百一十六、 ICONFONT 使用及其利与弊？

把一些零散的 icon 做成字体。我们调用文字的时候，渲染出来的就是 icon 图像，这样的显示就是 iconfont（字体图标）

好处：

1. iconfont 图像放大后，不会失真。
2. iconfont 节省流量
3. iconfont 在颜色变幻方面很简单

缺点：

1. iconfont 不能支持一个图像里面混入多重颜色
2. iconfont 的使用没有使用图片那么直接，简单。

详细参见：https://segmentfault.com/a/1190000005904616? ea=953657

一百一十七、 说说移动端 Web 分辨率

从以下几个方面做答：

（1） pc 到移动，渲染的变迁

（2） 可以更改的布局宽度

（3） 再次变迁的像素

（4） 又一次变迁

（5） 是时候说说安卓了

详细参见：https://segmentfault.com/a/1190000005884985

性能和效率

一百一十八、 你平时如何评测你写的前端代码的性能和效率。

ChromeDevTools 的 Timeline:是用来排查应用性能瓶颈的最佳工具。

ChromeDevTools 的 Audits:对页面性能进行检测，根据测试的结果进行优化。

第三方工具 Yslow。

详细参见：

http://www.cnblogs.com/—simon/p/5883336.html

http://blog.csdn.net/ivan0609/artide/details/45508365

http://www.wtoutiao.com/p/1305TZW.html

一百一十九、 如何优化页面，加快页面的加载速度(至少 5 条)

(1) 优化图片资源的格式和大小

(2) 开启网络压缩

(3) 使用浏览器缓存

(4) 减少重定向请求

(5) 使用 CDN 存储静态资源

(6) 减少 DNS 查询次数

(7) 压缩 css 和 js 内容

详细参见：http://www.mahaixiang.cn/wyzz/1589.html

一百二十、怎么保证多人开发进行内存泄漏的检查（内存分析 工具）

1) 使用 xcode 里面的 Analyze 进行静态分析

build setting ----》 automa ----》 mrc 环境

product ----》 analyze ----》command + R

2) 为避免不必要的麻烦，多人开发的时候尽量使用 ARC

内存泄露：

参考：http://blog.csdn.net/panda_bear/article/details/8009421

一百二十一、 前后端性能如何调优？

1. 减少 http 请求数
2. 使用内容分布式网络

3.给头部添加一个失效期或者 Cache—Control

4. Gzip 压缩组件
5. 把样式表放在前面
6. 把脚本放在最后

7. 不使用 CSS 表达式
8. 使用外部的 JavaScript 和 CSS
9. 减少 DNS 的查询
10. 缩小 JavaScript 和 CSS

参考：http://blog.csdn.net/sonta/article/details/44454787

一百二十二、 浏览器 http 请求过多怎么解决？

（1） 合并 JS、CSS 文件

（2） 合并图片 csssprite

（3） 使用 Imagemaps

（4） data 嵌入图片：如 base64

（5） 使用 CDN，减少 http 请求头

Web 安全

一百二十三、 你所了解到的 Web 攻击技术

（1） XSS 攻击

（2） CSRF 攻击

（3） 网络劫持攻击

（4） 控制台注入代码

（5） 钓鱼

详细参见：http://blog.csdn.net/fengyinchao/article/details/52303118

一百二十四、 如何防止 XSS 攻击？

（1） 将前端输出数据都进行转义

（2） 将输出的字符串中的\反斜杠进行转义

（3） 从 url 中获取的信息，防止方法是由后端获取，在前端转义后再行输出

（4） 使用 cookie 的 HttpOnly 属性，保护好 cookie

详细参见：http://blog.csdn.net/fengyinchao/article/details/52303118

一百二十五、 项目中有没有用过加密，哪种加密算法？

项目中没有用过，但我了解几个加密算法：

（1） RSA 加密

（2） MD5 加密

（3） SHA256 加密

一百二十六、 聊一聊网页的分段传输与渲染

从下面几个方面说：

（1） CHUNKED 编码

（2） BIGPIPE

（3） 分段传输与 bigpipe 适用场景

详细参见：https://segmentfault.com/a/1190000005989601？ea=984496

一百二十七、 百度移动端首页秒开是如何做到的？

从几个方面优化：

（1） 静态文件放置

（2） 缓存

（3） 外链

（4） 缓存 DOM

（5） 使用 iconfont

（6） 卡片的异步加载与缓存

（7） 不在首屏的就要异步化

（8） 少量静态文件的域名

详细参见：https://segmentfault.com/a/1190000005882953

一百二十八、 前端速度统计（性能统计）如何做？

回答下面的两个问题：

（1） 网站都有哪些指标？

（2） 如何统计自己网站的这些指标？

详细参见：https://segmentfault.eom/a/1190000005869953

架构

一百二十九、 如果让你来制作一个访问量很高的大型网站，你会如何来管理所有 CSS、JS 文件、图片？

（1） 遵循自定的一套 CSS，JS 和图片文件和文件夹命名规范

（2） 依托采用的前端工程化工具，依照工具脚手架规范（gulp, webpack, grunt, yeoman)

（3） 依据采用的框架规范（Vue, React, jQuery)

一百三十、如果没有框架、怎么搭建你的项目

应用原生 JS 自己尝试搭建一个 MVC 架构：

(1)基本模块

common:公共的一组件，下面的各模块都会用到

config:配置模块，解决框架的配置问题

startup:启动模块，解决框架和 Servlet 如何进行整合的问题

plugin:插件模块，插件机制的实现，提供 IPlugin 的抽象实现

routing:路由模块，解决请求路径的解析问题，提供了 IRoute 的抽象实现和基本实现

controller:控制器模块，解决的是如何产生控制器

model:视图模型模块，解决的是如何绑定方法的参数

action: action 模块，解决的是如何调用方法以及方法返回的结果，提供了 IActionResult 的 抽象实现和基本实现

view:视图模块，解决的是各种视图引擎和框架的适配

filter:过滤器模块，解决是执行 Action,返回 IActionResult 前后的 AOP 功能，提供了 IFilter 的抽象实现以及基本实现

扩展模块 filters：一些 IFilter 的实现

results:一些 IActionResult 的实现

routes:一些 IRoute 的实现

plugins:一些 IPlugin 的实现

一百三十一、 在选择框架的时候要从哪方面入手

影响团队技术选型有很多因素，如技术组成，新技术，新框架，语言及发布等。为了更好的 考量不同的因素，需要列出重要的象限，如开发效率、团队喜好，依次来决定哪个框架更适 合当前的团队和项目。上线时间影响框架选择，不要盲目替换现有框架。

（1） jQuery

项目功能比较简单。并不需要做成一个单页面应用，就不需要 MV*框架。项目是一个遗留 系统。与其使用其他框架来替换，不如留着以后重写项目。

（2） AngularJS

当我们在制作一个应用，它对性能要求不是很高的时候，那么我们应该选择开发速度更快的 技术栈 AngularJS，她拥有混合开发能力的 ionic 框架。对于复杂的前端应用来说，基于 Angular.js 应用的运行效率，仍然有大量地改进空间。Angular2 需要学习新的语言，需慎重 选择。

（3） React

选择 React 有两个原因，一是通过 VirtualDOM 提高运行效率，二是通过组件化提高开发效 率。大型项目首选。选择 React 还有_个原因是: ReactNative、ReactVR 等等，可以让 React 运行在不同的平台之上。我们还能通过 React 轻松编写出原生应用，还有 VR 应用。

令人遗憾的是 React 只是一个 View 层，它是为了优化 DOM 的操作而诞生的。为了

完成一 个完整的应用，我们还需要路由库、执行单向流库、webAP 丨调用库、测试库、依赖管理库 等等，为了完整搭建出一个完整的 React 项目，我们还需要做大量的额外工作。

（4） Vue.js

对于使用 Vue.js 的开发者来说，我们仍然可以使用熟悉的 HTML 和 CSS 来编写代码。并 且，Vue.js 也使用了 Virtual D〇M、Reactive 及组件化的思想，可以让我们集中精力于编写 应用，而不是应用的性能。

对于没有 Angular 和 React 经验的团队，并且规模不大的前端项目来说，Vue.js 是一个非常 好的选择。

详细参见：https://zhuanlan.zhihu.com/p/25194137

一百三十二、 聊一聊前端模板与渲染

（1） 页面级的渲染，后端模板

如 smarty，这种方式的特点是展示数据快，直接后端拼装好数据与模板，展现到用户面前， 对 SEO 友好。

（2） 异步的请求与新增模板，前端模板

如 Mustache, ArtTemplate,前端解析模板的引擎的语法，与后端解析模板引擎语法一致。 这样就达到了一份 HTML 前后端一起使用的效果。

详细参见：https://segmentfault.com/a/1190000005916423

混合开发

一百三十三、 UIWebView 和 JavaScript 之间是怎么交互的？

UlWebView 是 i〇SSDK 中渲染网面的控件，在显示网页的时候，我们可以 hack 网页然后显示想显示的内容。其中就要用至 JavaScript 的知识，而 UlWebView 与 javascript 交互的 方法就是 stringByEvaluatingJavaScriptFromString:

有了这个方法我们可以通过 objc 调用 javascript,可以注入 javascript。

UIWebView 是 iOSSDK 中渲染网面的控件，在显示网页的时候，我们可以 hack 网页然后显示想显示的内容。其中就要用至 JavaScript 的知识，而 UIWebView 与 javascript 交互的 方法就是 stringByEvaluatingJavaScriptFromString,有了这个方法我们可以通过 objc 调用 javascript,可以注入 javascript

Js 调用〇C 方法原理就是利用 UIWebView 重定向请求，传一些命令到我们的 UIWebView,在 UIWebView 的 delegate 的方法中接收这些命令，并根据命令执行相应的 objc 方法。这样就相当于在 javascript 中调用 objc 的方法。

在 android 中，我们有固有组件 webview,经过设置可以让它支持我们的 js 的渲染，然后在代码中设置（WebViewClient/WebChromeClient)让应用跳转页面时在本 webview 中跳 转，通过 webview.loadurl (String str)方法可以在需要的地方加载我们前端的页面或者调用 前端所定义的方法（wv.loadUrl("javascript:sendDataToAndroid('我是来自 js 的呦，你看到了 吗')");），我们再通过 JavascriptInterface 接口设置我们前端和 android

通讯的标识，

wv.addJavascriptInterface(newMJavascriptInterface(getApplicationContext()), "WebViewFunc");

这样前端就可以在页面上调用我们的方法了，funl 方法是在 android 中定义的 Window.WebViewFunc.fun1 ();

总之，前端和 android 或者 ios 进行结合开发，我们称之为混合开发，原理就是在原生 的开发语言中，我们提供了一个组件 webview，这个组件就是我们的原生语言的浏览器，但 是我们得自行设置让其能够完美支持我们的应用，需要设置对应的标识，然后连接起来，我 们称之为 JavascriptInterfac。

一百三十四、 混合开发桥接 api 是怎么调用的，需要引入类库嘛？ 调用的对象是什么？

Hybrid 框架结构

HyBridApp= H5 App+ Native 框架

H5App 用来实现功能逻辑和页面渲染 Native 框架提供 WebView 和设备接口供 H5 调用

方案一重混合应用，在开发原生应用的基础上，嵌入 WebView 但是整体的架构使用 原生应用提供，一般这样的开发由 Native 开发人员和 Web 前端开发人员组成。Native 开发人

员会写好基本的架构以及 API 让 Web 开发人员开发界面以及大部分的渲染。保证到交互设 计，以及开发都有一个比较折中的效果出来，优化得好也会有很棒的效果。

Hybrid App 技术发展的早期，Web 的运行性能成为主要瓶颈！

为解决性能问题 Hybrid App 走向“重混”。

通过多 WebView:实现流畅的多页加载和专场动画。

使用 Navtive UI 组件：框架、菜单、日期等。

“重混”的优缺点 优点：

一提升了运行性能 一增强了交互体验

缺点一

一Web 和 Native 技术交叉混杂 一需要同时掌握 Web 和 Native 技术，学习难度增加 一一个页面有 Web 组件也有 Native 组件，编程调试困难

需要引入各自需要的各种依赖工具

方案二：轻混合应用，使用 PhoneGap、AppCan 之类的中间件，以 WebView 作为用 户界面层，以 Javascript 作为基本逻辑，以及和中间件通讯，再由中间件访问底层 API 的方 式，进行应用开发。这种架构一般会非常依赖 WebView 层的性能。

随着时代的发展，手机硬件、浏览器技术、无线网络技术都得到了大幅的提升，H5 已经可 以支持复杂应用，并拥有良好的运行性能。使用轻混方案的 App 也越来越多。

目前我们要学习的 HybridApp 开发就是方案二，使用 H5+Js+Native 框架开发当前轻混合应用。

Phonegap 引入 phonegap.js 或者 cordova.js,对象为 navigator Dcloud 引入引入

mui.js 或者其他的 js 组件，对象为 plus apiloud 引入各种第三方插件，对象为 api

顺变提一下，2012 年 8 月，微信公众平台的上线，重新定义了移动应用：移动应用 =IphoneApp+ AndroidApp+ 微信 App

一百三十五、 说一下你对支付，推送（远程，本地）的理解

消息的推送主要有两种：

一种是本地推送，主要应用在系统的工具中，例如：闹钟，生日提醒等；实现本地推送需要以下三个步骤，

1. 实例化一个本地推送对象
2. 设置通知对象的各个属性
3. 添加本地推送对象

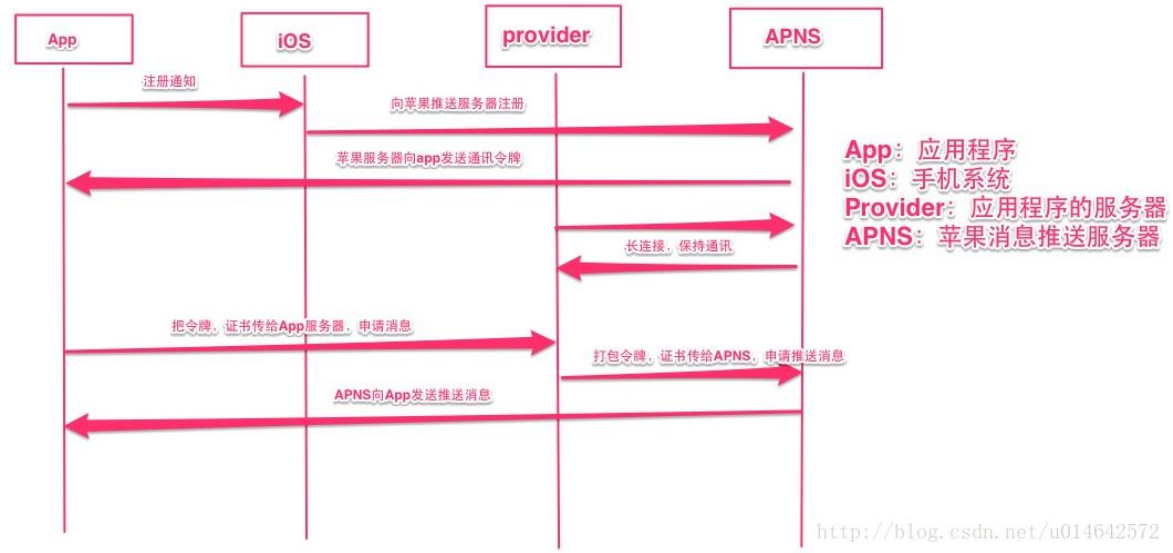
一种是远程消息推送，主要应用联网设备的信息推送，例如：邮件，各种软件的广告或优惠信息的推送。远程推送比较复杂，需要使用开发者账号进行申请证书，获得实

现推送功能的配置文件，所以想要实现远程推送功能，必须要有开发者账号并且生成配置文件

1. 完成证书的申请和 Xcode 的配置
2. 在 Demo 中注册远程服务对象，并设置其代理
3. 找一个简单的 App 服务器进行消息推送（推荐使用:PushMeBaby, gitup 网站上就有）
4. 运行 PushMeBaby

参考网址：http://blog.csdn.net/u014642572/article/details/26857717

远程推送流程图如下



一百三十六、 什么是代理和通知，写一下他们基本的实现方式

代理：“一对一”，对同一个协议，一个对象只能设置一个代理 delegate

六个步骤：

1．声明一个协议,定义代理方法

2．遵循协议

3．设置一个代理对象

4．调用代理方法

5．给代理赋值

6．实现代理方法

注意事项：

1．单例对象不能用代理；

2．代理执行协议方法时要使用 respondsToSelector 检查其代理是否符合

3．协议(检查对象能否响应指定的消息),以避免代理在回调时因为没有实现方法而造成程序崩溃

使用场景：

公共接口，方法较多也选择用 delegate 进行解耦 iOS 最常用 tableViewDelegate, textViewDelegate iOS 有很多例子比如常用的网络库 AFNetwork，ASIHTTP 库，UlAlertView 类。

通知：一对一一对多传值

四个步骤：

1．发送通知

2．创建监听者

3．接收通知

4．移除监听者

使用场景：

1一很多控制器都需要知道一个事件，应该用通知；

2 一相隔多层的两个控制器之间跳转

注意事项：

1．一旦接收消息的对象多了，就难以控制了，可能有你不希望的对象接收了消息并做了处理

2．创建了观察者，在 dealloc 里面一定要移除；

Block:Block 是 i〇S4.0+和 Mac OS X 10.6+引进的对 C 语言的扩展，用来实现匿名函数的特性。Blocks 语法块代码以闭包得形式将各种内容进行传递，可以是代码，可以是数组 无所不能。闭包就是能够读取其它函数内部变量的函数。就是在一段请求连续

代码中可以看 到调用参数（如发送请求）和响应结果。所以采用 Block 技术能够抽象出很多共用函数，提 高了代码的可读性，可维护性，封装性。

使用场景：

—:动画

二：数据请求回调

三：枚举回调

四：多线程 gcd

注意事项：Iblock 需要注意防止循环引用

参考网址：http://www.cnblogs.com/wenboliu/articles/5422033.html

一百三十七、 UIViewController 的生命周期

1．通过 alloc init 分配内存，初始化 controller.

2．loadView （loadView 方法默认实现[superloadView]

如果在初始化 controller 时指定了 xib 文件名，就会根据传入的 xib 文件名加载对应的 xib 文件，如果 没传 xib 文件名，默认会加载跟 controller 同名的 xib 文件，如果没找到相关联的 xib 文件，就会创建一个空白的 UIView,然后赋給 controller 的 view）

3．viewDidLoad(当 loadView 创建完 view 之后，此时 view 已经完成加载了，会调用 viewDidLoad 方法；一般我会在这里做界面上的初始化操作，比如添加按钮，子视图，等等.）

4．viewWillAppear(当 view 在 load 完之后，将要显示在屏幕之前会调用这个方法，在重写这些方法时候最好先调用一下系统的方法之后在做操作。）

5．viewDidAppear (当 view 已经在屏幕上显示出来之后，会调用这个方法，当一个视图被移除屏幕并且销毁的时候）

6．viewWillDisappear (当视图将要从屏幕上移除时候调用)

7．viewDidDisappear (当视图已经从屏幕上移除时候调用)

8．Dealloc (view 被销毁时候调用，如果是手动管理内存的话，需要释放掉之前在 init 和 viewDidLoad 中分配的内存（类似 alloc,new,copy) ; dealloc 方法不能由我们主动调用，必须等引用计数为 0 时候由系统调用.）

9．参考网址： http://www.cnblogs.eom/wujy/p/5822329.html

一百三十八、 rem 布局字体太大怎么处理？

一般情况下我们设置了 html 根节点的字体大小作为 rem 单位的一个基本标准，那么我 们可以紧接着在 body 标签内设置一个字体大小为该应用的基本字体大小

针对于一些机型如果一开始就显示的字体不正常，我们可以通过判断机型然后加载不同的样式

```
〈scriptlanguage="javascript">
        window.onload= function() {
            alert(“1”);
```

```
varu = navigator.userAgent;
if(u.index〇f('Android') >—1 || u.indexOf('Linux') >—1) { //安卓手机
    alert("安卓手机")；
} elseif(u.index〇f('iPhone') >—1) { //苹果手机
    alert("苹果手机")；
} elseif(u.index〇f('WindowsPhone') >—1) { //winphone 手机
alert("winphone 手机")；
}
}
〈/script>
```

一百三十九、 如何调用原生的接口？

首先你得选择一个合适的框架作为自己的基础，以 Dcloud 为例，页面中一定要存在一个事件，plusready, plusready 实际上是原生将桥接 js 注入到页面中的容器，进行任何方法 调用的时候都在 plusready 之后。所有 api 方法全部都托管在了一个 plus 对象中。使用语法 plus.模块名称.具体方法（参数，callback)

当我们需要打开系统相册的时候，可以这样做：

Gallery 模块管理系统相册，支持从相册中选择图片或视频文件、保存图片或视频文 件到相册等功能。通过 plus.gallery 获取相册管理对象。打开相册 plus.gallery.pick 进行打开，选取多个图片{multiple:true,maximum:9,system:false}

一百四十、微信支付怎么做？说说流程

1．申请微信公众号及支付功能申请：根据公众号申请流程申请即可。

2．获取商户支付配置信息及支付测试配置：

支付授权目录最多可以配置三个域名，测试授权目录只可以一个，这里需要 注意的是域名大小写必须要网站 URL一致，否则会无法通过授权，提示支付请求的 URL 不合法。另外，测试支付的微信号必须加到测试白名单，否则无法进行支付测 试。

3．H5 页面发起支付请求，请求生成支付订单，获取用户授权（获取用户的 openid)

4．调用统一下单 API，生成预付单

5．生成 JSAPI 页面调用的支付参数并签名，注意时间戳 timeStamp 是 32 位字符串

6．返回支付参数 prepay—id,paySign 参数的 html 文本给前端。

7．微信浏览器自动调起支付 JSAPI 接口支付，提示用户输入密码。

8．确认支付，输入密码，提交支付。

9．步通知商户支付结果，商户收到通知返回确认信息。

10．返回支付结果，并发微信消息提示。

11．展示支付信息给用户，跳转到支付结果页面。

一百四十一、 混合开发的注意点

增强 WebView:原生 WebView 基本是 PC 平台浏览器内核的移植，但对于移动场景并不完全适合，各种硬件 API 得不到 HTML5 原生支持。因此对于 WebView 的种种 Hack、增强应运而生，甚至出现了基于增强 WebView 提供第三方服务的。

路由：应用内跳转由于加入了 WebView 而变得复杂起来，同时由于组件化、模块化带来的问 题，路由也成为人们讨论的重点。

缓存：移动网络条件差，为了用户体验，必须要做资源缓存和预加载。

通信：即 HTML5 和 Native 之间的通信。利用系统提供的桥接 API 可以实现，不过在应用上还 有着一些坑点和安全问题。

一百四十二、 说说你对手机平台的安装包后缀的理解

Android：**apk

Ios：**ipa

Windows： wp7 wp8 的是 xap wp8.1 以后用 8.1 开发的是 appx

NodeJS

一百四十三、 谈谈你对 Socket 编程的理解及实现原理，Socket 之间是怎么通讯的

A、Socket 定义

Socket 是进程通讯的一种方式，即调用这个网络库的一些 API 函数实现分布在不同主机的相 关进程之间的数据交换。几个定义：IP 地址：即依照 TCP/IP 协议分配给本地主机的网络地 址，两个进程要通讯，任一进程首先要知道通讯对方的位置，即对方的 IP。端口号：用来辨 别本地通讯进程，一个本地的进程在通讯时均会占用一个端口号，不同的进程端口号不同， 因此在通讯前必须要分配一个没有被访问的端口号。连接：指两个进程间的通讯链路。

B、实现原理

在 TCP/IP 网络应用中，通信的两个进程间相互作用的主要模式是客户/服务器（Client/Server, C/S)模式，即客户向服务器发出服务请求，服务器接收到请求后，提供相应的服务。客户/服务器模式的建立基于以下两点：首先，建立网络的起因是网络中软硬件资源、运算能力和信息不均等，需要共享，从而造就拥有众多资源的主机提供服务，资源较少的客 户请求服务这一非对等作用。其次，网间进程通信完全是异步的，相互通信的进程间既不存 在父子关系，又不共享内存缓冲区，因此需要一种机制为希望通信的进程间建立联系，为二 者的数据交换提供同步，这就是基于客户/服务器模式的 TCP/IP。

C、通讯过程

服务器端：其过程是首先服务器方要先启动，并根据请求提供相应服务：（1)打开一通信 通道并告知本地主机，它愿意在某一公认地址上的某端□(如 FTP 的端口可能为 21)接收客 户请求；（2)等待客户请求到达该端口； （3)接收到客户端的服务请求时，处理该请求并 发送应答信号。接收到并发服务请求，要激活一新进程来处理这个客户请求（如 UNIX 系统 中用 fork、exec)。新进程处理此客户请求，并不需要对其它请求作出应答。服务完成后， 关闭此新进程与客户的通信链路，并终止。（4)返回第（2)步，等待另一客户请求。（5）关闭服务器客户端：（1)打开一通信通道，并连接到服务器所在主机的特定端口；(2)向服务器发服务请求报文，等待并接收应答；继续提出请求.....

（3)请求结束后关闭通信通道并终止。

从上面所描述过程可知：（1）客户与服务器进程的作用是非对称的，因 此代码不同。（2）服务器进程一般是先启动的。只要系统运行，该服务进程一直存在，直到正常或强迫终止。

详细参见：

https://www.zhihu.com/question/29637351

http://blog.csdn.net/panker2008/article/details/46502783?ref=myread

一百四十四、 WEB 应用从服务器主动推送 Data 到客户端有哪些方式？

一般的服务器 Push 技术包括：

1) 基于 AJAX 的长轮询（long—polling)方式，服务器 Hold—段时间后再返回信息；
2) HTTPStreaming,通过 iframe 和〈script〉#签完成数据的传输；
3) TCP 长连接
4) HTML5 新引入的 WebSocket，可以实现服务器主动发送数据至网页端，它和 HTTP—样，是一个基于 HTTP 的应用层协议，跑的是 TCP，所以本质上还是个长连接，双向通信， 意味着服务器端和客户端可以同时发送并响应请求，而不再像 HTTP 的请求和响应。
5) nodejs 的 http://socket.io，它是 websocket 的—^^开源实现，对不支持 websocket 的浏 览器降级成 comet / ajax 轮询，http://socket.io 的良好封装使代码编写非常容易。

上述的 1 和 2 统称为 comet 技术。comet 详细参考：
http://www.ibm.com/developerworks/cn/web/wa—lo—comet/

一百四十五、 简述 Node.js 的适用场景？

IIO 密集而非计算密集的情景；高并发微数据（比如账号系统）的情景。特别是高并发，Node.js 的性能随并发数量的提高而衰减的现象相比其他 server 都有很明显的优势。

Bad Use Cases

1．CPU heavy apps （高 CPU 消耗的 app）

2．Simple CRUD / HTML apps （简单的 CRUD / HTML apps）

3．NoSQL + Node.js + Buzzword Bullshit （NoSQL + Node.js + 各种扯淡的时髦词汇）

Good Use Cases

1．JSON API

2．Single page apps （单页面 app）

3．Shelling out to unix tools （对 unix 工具的脚本化调用）

4．Streaming data （流数据）

5．Soft Realtime Applications （软件实时程序）

一百四十六、 什么是 HTTPS，做什么用的呢？如何开启 HTTPS?

1) 什么是 HTTPS

https 是 http 的加密版本，是在 http 请求的基础上，采用 ssl 进行加密传输。

2) 做什么用

加密数据，反劫持，SEO

3) 如何开启

生成私钥与证书，配置 nginx,重启 nginx 看效果

详细参见：
https://segmentfault.com/a/1190000006199237?utm_source=tuicool&utm_medium=referral

一百四十七、 你们原来公司如何发送的新消息推送?

（参考：）一般的服务器 Push 技术包括：

1．基于 AJAX 的长轮询（long—polling）方式，服务器 Hold一段时间后再返回信息；

2．HTTP Streaming，通过 iframe 和<script>标签完成数据的传输；

3．TCP 长连接

4．HTML5 新引入的 WebSocket，可以实现服务器主动发送数据至网页端，它和 HTTP一样，是一个基于 HTTP 的应用层协议，跑的是 TCP，所以本质上还是个长连接，双向通信，意味着服务器端和客户端可以同时发送并响应请求，而不再像 HTTP 的请求和响应

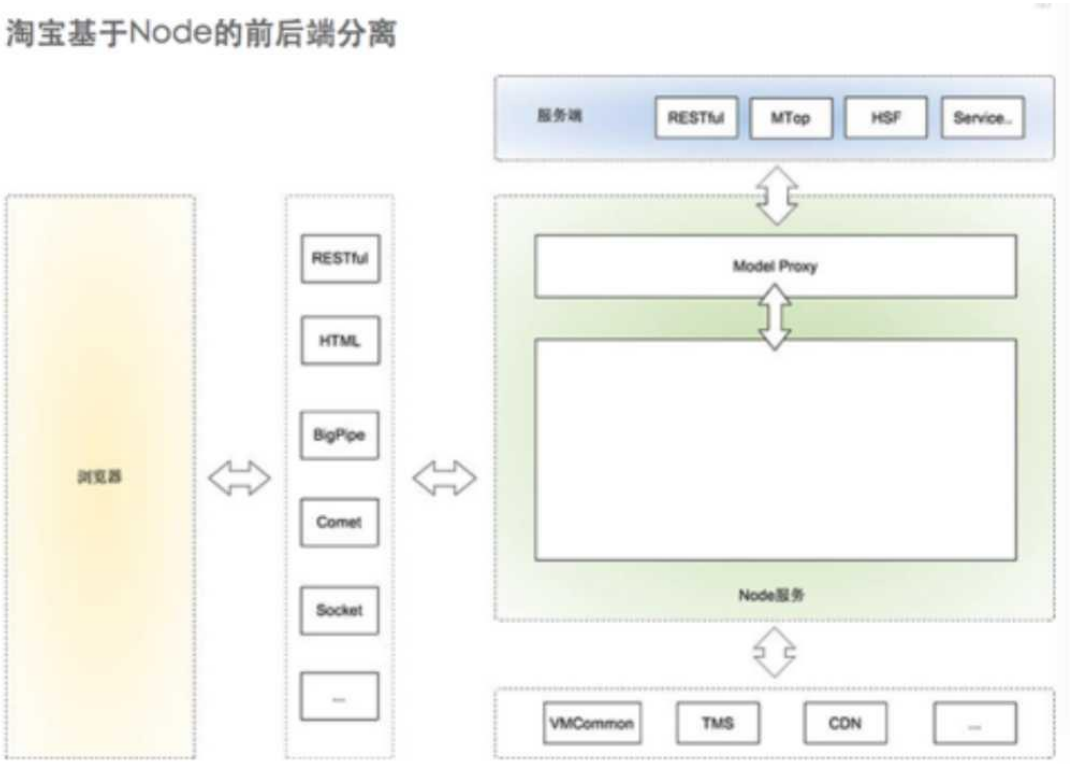
上述的 1 和 2 统称为 comet 技术，Comet：基于 HTTP 长连接的“服务器推”技术前些日子给项目网站加了后台通知的实时推送到前端显示，用的是 nodejs 的 http://socket.io，它是 websocket 的一个开源实现，对不支持 websocket 的浏览器降级成 comet / ajax 轮询，http://socket.io 的良好封装使代码编写非常容易。

一百四十八、 如何用 NodeJS 搭建中间层？

（如下图）最上端是服务端，就是我们常说的后端。后端对于我们来说，就是一个接口的集合，服务端提供各种各样的接口供我们使用。因为有 Node 层，也不用局限是什么形式的服务。对于后端开发来说，他们只用关心业务代码的接口实现。

服务端下面是 Node 应用。

Node 应用中有一层 ModelProxy 与服务端进行通讯。这一层主要目前是抹平我们对不同接口 的调用方式，封装一些 view 层需要的 Model。



Node 层还能轻松实现原来 vmcommon,tms (引用淘宝内容管理系统）等需求。

Node 层要使用什么框架由开发者自己决定。不过推荐使用 express+xTemplate 的组合， xTemplate 能做到前后端公用。

怎么用 Node 大家自己决定，但是令人兴奋的是，我们终于可以使用 Node 轻松实现我们想要的输出方式 JSON/JSONP/RESTful/HTML/BigPipe/Comet/Socket/同步、异步，想怎么整就怎么整，完全根据你的场景决定。

浏览器层在我们这个架构中没有变化，也不希望因为引入 Node 改变你以前在浏览器中开发的认知。

引入 Node，只是把本该就前端控制的部分交由前端掌控。

详细参见：http://blog.csdn.net/u011413061/article/details/50294263

组件设计

一百四十九、 设计一个弹框组件，组件宽度为屏幕高度的 50%， 宽度为屏幕宽度的 80%，水平垂直居中。弹窗组件有 header, body, footer 三部分，header 中有标题，可定制，body 区域，footer 区域有确定和取消按钮，可定制两个按钮的文字内容，组件外的内容有遮罩，点击遮罩和取消按钮时关闭弹框，参照下图。（类似于 layer 的弹出层插件）

使用面向对象封装插件较为合适

构造函数的参数有 header 的标题及 body 内容和按钮文字内容

封装的方法应该有 show, hide,在点击遮罩和取消按钮时调用 hide 方法

并且 hide 和 show 方法应该有返回值以供判断。

一百五十、实现一个手势滑动轮播图组件。效果参考：https://static.xiaohongchun.com/goods/4514(请在手机里打开)

详细参考：http://www.jb51.net/article/65177.htm

一百五十一、 设计基于观察者模式的事件绑定机制

观察者模式（发布-订阅模式）的定义：Observer 的意图是定义对象之间的一种一（被观察者）对多（观察者） 的关系，当一个对象的状态发生改变时，所有依赖它的对象得到通知，并且会自动更新自己

在 JavaScript 中，一般使用事件模型来替代传统的观察者模式。

好处：

（1）可广泛应用于异步编程中，是一种替代传递回调函数的方案。

（2）可取代对象之间硬编码的通知机制，一个对象不用再显示地调用另外一个对象的某个接口。两对象轻松解耦。

代码参考：http://blog.csdn.net/phker/article/details/6880371

http://www.cnblogs.com/LuckyWi nty/p/5796190.html

一百五十二、 jq 自己扩展过什么插件？

弹出层插件、pagination 插件、瀑布流插件、模态框插件等

参考：Jquery 插件库 jquery 之家

一百五十三、 侧滑菜单如何实现？

主要依靠两个大的容器来模拟侧滑菜单界面和主界面，把侧滑菜单放到页面右侧看不到的地方，在操作的同时，使用 css3 过渡、动画或者 jq 来使两个容器相对运动，实现侧滑菜单效果

参考：http://www.111cn.net/wy/js—ajax/99687.htm

一百五十四、 权限管理如何实现？

（1）前端控制：

前端的控制比较简单，从后台获取到用户的权限之后，可以存在 session 或者 cookie 中，然后在页面加载的时候，通过 session 或者 cookie 中存的权限来选择让该功能展现或者禁用。

前端实现代码详细参见：http://blog.csdn.net/liuweidagege/article/details/42497731

（2）后台控制：

仅仅依靠前端的控制是无法完美解决权限控制的问题，因为前端页面的加载过程是在浏览器中完成的，用户可以自行篡改页面；或者用户可以直接通过 URI 请求来获取非法权限功能。所以需要在后台实现权限控制。

后台的控制方法也很多，比如 filter、spring 的 AOP 等。在此选用 springMVC 的 interceptor 来控制。

（3）全局异常管理：

思路是在拦截器中权限校验失败时，抛出一个权限校验失败的异常，然后通过全局异常管理类来捕获并返回前端特定的格式。具体如下。

一百五十五、 一个大数组，可能存了 100 万个数字，要从其中取出 来第二大的数的下标，有什么快速的方法？

用两个变量 max，max2，其中 max 储存最大值，max2 储存第二大值；初始化的时候，将数组中的第一个元素中较大的存进 max 中，较小的存进 max2 中，然后从第三个元素(下标为 2)的元素开始，如果遇到的数比 max 大，就让 max2=max;max 等于遇到的数一直循环，直到数组尾部，最后输出 max2

单元测试

一百五十六、 单个组件怎么测试性能

React 组件测试框架用 mocha,测试库用官方的测试工具库，也可使用第三方库 Enzyme,建议使用第三方的。

详细参见：http://www.ruanyifeng.com/blog/2016/02/react—testing—tutorial.html

Vue 使用 Unit 和 e2e 测试工具：

详细参见：http://www.tuicool.com/articles/6vulNvR

React

一百五十七、 有了解过 React.js 吗？

React.js 只是—个视图库

（1）声明式设计

（2）高效：通过对 DOM 的模拟，最大限度的减少与 DOM 的交互。

（3）灵活：可以与已知的框架或库很好的配合。

（4）JSX：是 js 语法的扩展，不—定使用，但建议用。

（5）组件：构建组件，使代码更容易得到复用，能够很好地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，这也是解释了它为什么比传统数据绑定更简单。

一百五十八、 redux 中间件

中间件提供第三方插件的模式，自定义拦截 action—> reducer 的过程。变为 action —〉 middlewares —> reducer。这种机制可以让我们改变数据流，实现如异步 action，action 过滤，日志输出，异常报告等功能。

常见的中间件：

redux—logger:提供日志输出

redux—thunk:处理异步操作

redux—promise:处理异步操作，actionCreator 的返回值是 promise

一百五十九、 redux 有什么缺点

1．一个组件所需要的数据，必须由父组件传过来，而不能像 flux 中直接从 store 取。

2．当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新 render,可能会 有效率影响，或者需要写复杂的 shouldComponentUpdate 进行判断。

一百六十、react 组件的划分业务组件技术组件？

根据组件的职责通常把组件分为 UI 组件和容器组件。

UI 组件负责 UI 的呈现，容器组件负责管理数据和逻辑。

两者通过 React—Redux 提供 connect 方法联系起来。

具体使用可以参照如下链接：

http://www.ruanyifeng.com/blog/2016/09/redux_tutorial_part_three_react—redux.html

一百六十一、 react 生命周期函数

这个问题要考察的是组件的生命周期

一、 初始化阶段：

Constructor 初始化状态

componentWillMount:组件即将被装载、渲染到页面上

render:组件在这里生成虚拟的 DOM 节点

componentDidMount:组件真正在被装载之后

二、 运行中状态：

componentWillReceiveProps:组件将要接收到属性的时候调用

shouldComponentUpdate:组件接受到新属性或者新状态的时候（可以返回 false, 接收数据后不更新，阻止 render 调用，后面的函数不会被继续执行了）

componentWillUpdate:组件即将更新不能修改属性和状态

render:组件重新描绘

componentDidUpdate:组件已经更新

三、 销毁阶段：

componentWillUnmount:组件即将销毁

一百六十二、 react 性能优化是哪个周期函数？

shouldComponentUpdate 这个方法用来判断是否需要调用 render 方法重新描绘 dom。因为 dom 的描绘非常消耗性能，如果我们能在 shouldComponentUpdate 方法中能够写出更优化的 domdiff 算法，可以极大的提高性能。

详细参考：https://segmentfault.com/a/1190000006254212

一百六十三、 为什么虚拟 dom 会提高性能？

虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法避免了没有必要的 dom 操作，从而提高性能。

具体实现步骤如下：

1．用 JavaScript 对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档当中

2．当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异

3．把 2 所记录的差异应用到步骤 1 所构建的真正的 DOM 树上，视图就更新了。

参考链接：https://www.zhihu.com/question/29504639?sort=created

一百六十四、 diff 算法？

1) 把树形结构按照层级分解，只比较同级元素。

2) 给列表结构的每个单元添加唯_的 key 属性，方便比较。

3) React 只会匹配相同 class 的 component (这里面的 class 指的是组件的名字）

4) 合并操作，调用 component 的 setState 方法的时候,React 将其标记为 dirty.到每一个事件循环 结束,React 检查所有标记 dirty 的 component 重新绘制.

5) 选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能。

参考链接：https://segmentfault.com/a/1190000000606216

一百六十五、 react 性能优化方案

1) 重写 shouldComponentUpdate 来避免不必要的 dom 操作 0

2) 使用 production 版本的 react.js0

3) 使用 key 来帮助 React 识别列表中所有子组件的最小变化。

参考链接：

https://segmentfault.com/a/1190000006254212http://blog.csdn.net/limm33/article/details/50948869

一百六十六、 简述 flux 思想

Flux 的最大特点，就是数据的"单向流动"。

1. 用户访问 View
2. View 发出用户的 Action
3. Dispatcher 收到 Action,要求 Store 进行相应的更新
4. Store 更新后，发出一个"change"事件
5. View 收到"change"事件后，更新页面

参考链接：http://www.ruanyifeng.com/blog/2016/01/flux.html

一百六十七、 React 项目用过什么脚手架？ Mern？Yeoman？

Mern: MERN 是脚手架的工具，它可以很容易地使用 Mongo, Express, ReactandNodeJS 生成

同构 JS 应用。它最大限度地减少安装时间，并得到您使用的成熟技术来加速开发。

参考链接：http://www.open—open.com/lib/view/open1455953055292.html

Vue.js

一百六十八、 vue 与 react 的对比，如何选型？从性能，生态圈，数据量，数据的传递上，作比较

1) React 和 Vue 有许多相似之处，它们都有：

使用 VirtualDOM

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

将注意力集中保持在核心库，伴随于此，有配套的路由和负责处理全局状态管理的库。

2) 性能：

到目前为止，针对现实情况的测试中，Vue 的性能是优于 React 的。

3) 生态圈

Vue.js: ES6+Webpack+unit/e2e+Vue+vue—router+单文件组件+vuex+iVew

React: ES6+Webpack+Enzyme+React+React—router+Redux

4) 什么时候选择 Vue.js

如果你喜欢用（或希望能够用）模板搭建应用，请使用 Vue

如果你喜欢简单和”能用就行”的东西，请使用 Vue

如果你的应用需要尽可能的小和快，请使用 Vue

如果你计划构建一个大型应用程序，请使用 React

如果你想要一个同时适用于 Web 端和原生 App 的框架，请选择 React

如果你想要最大的生态圈，请使用 React

详细参考：

http://cn.vuejs.org/v2/guide/comparison.html#React

http://blog.csdn.net/yzh_2017/article/details/54909166

一百六十九、 vueslot 是做什么的？

简单来说，假如父组件需要在子组件内放一些 DOM，那么这些 DOM 是显示、不显示、在哪 个地方显示、如何显示，就是 slot 分发负责的活。

详细参考：http://cn.vueis.org/v2/guide/comp○nents.html#使用一Slot一分发内容

一百七十、vue 和 angular 的优缺点以及适用场合？

参见：《在选择框架的时候要从哪方面入手》一题

一百七十一、 vue 路由实现原理？

以官方仓库下 examples/basic 基础例子来一点点具体分析整个流程。

和流程相关的主要需要关注点的就是 components、history 目录以及 create—matcher.js、 create—route—map.js、index.js，install.js。

从入口，作为插件，实例化 VueRouter,实例化 History,实例化 Vue, defineReactive 定义_route, router—link 和 router—view 组件等几个方面展开分析

必须参见：http://www.tuicool.com/articles/jQRnlrF

一百七十二、 你们 vue 项目是打包了一个 js 文件，一个 css 文件，还是有多个文件？

根据 vue—cli 脚手架规范，一个 js 文件，一个 CSS 文件。

详细参见：

http://blog.csdn.net/lx376693576/article/details/54911340

https://zhuanlan.zhihu.com/p/24322005

一百七十三、 vue 遇到的坑，如何解决的？

Vue1.0 升级 2.0 有很多坑:生命周期；路由中引入静态 js，全局组件，全局变量，全局 function; v—for 循环的 key, value 值互换了位置，还有 track—by; filter 过滤器；遍历数组时， key 值不能做 model;父子通信等。

其他坑详见：

http://blog.csdn.net/lemon_zhao/article/details/55510589https://segmentfault.com/a/1190000008347498http://www.tuicool.com/articles/aUrmumV

一百七十四、 vue 的双向绑定的原理，和 angular 的对比

在不同的 MWM 框架中，实现双向数据绑定的技术有所不同。

AngulaJS 采用“脏值检测”的方式，数据发生变更后，对于所有的数据和视图的绑定关系进 行一次检测，识别是否有数据发生了改变，有变化进行处理，可能进一步引发其他数据的改 变，所以这个过程可能会循环几次，一直到不再有数据变化发生后，将变更的数据发送到视 图，更新页面展现。如果是手动对 ViewModel 的数据进行变更，为确保变更同步到视图， 需要手动触发一次“脏值检测”。

VueJS 则使用 ES5 提供的Object.defineProperty()方法，监控对数据的操作，从而可以自 动触发数据同步。并且，由于是在不同的数据上触发同步，可以精确的将变更发送给绑定的 视图，而不是对所有的数据都执行一次检测。

详细参见：http://www.jianshu.eom/p/d3a15a1f94a0

一百七十五、 vue—cli， 脚 手 架

安装：$ npminstaH—gvue—cli

使用：$ vueinit<template—name><project—name>

webpack 配置详解：https://zhuanlan.zhihu.eom/p/24322005

一百七十六、 Vue 里面 router—link 在电脑上有用，在安卓上没反应 怎么解决？

Vue 路由在 Android 机上有问题，babel 问题，安装 babelpolypill 插件解决。

框架底层

一百七十七、 jOuery 源码中值得借鉴的？

使用模块化思想，模块间保持独立，不会导致多个开发人员合作时产生的冲突。

1. 在设计程序时，要结构清晰，高内聚，低耦合。
2. 利用多态的方式，实现方法的重载，提高代码的复用率
3. jQuery 的链式调用以及回溯
4. jQuery.fn.extend 与;jQuery.extend 方法来实现扩展静态方法或实例方法

一百七十八、 $.ready 是怎么实现的？

原生 js 中 window.onload 事件是在页面所有的资源都加载完毕后触发的.如果页面上有大图片等资源响应缓慢，会导致 window.onload 事件迟迟无法触发.所以出现了 DOM Ready 事件.此事件在 D0M 文档结构准备完毕后触发，即在资源加载前触发.

jQuery 中的 ready 方法实现了当页面加载完成后才执行的效果，但他并不是 window.onload 或者 doucment.onload 的封装，而是使用标准 W3C 浏览器 DOM 隐藏 api 和旧浏览器缺陷来完成的。可以通过阅读 jq 源码来理解：

```
DOMContentLoaded = function(){
    //取消事件监听，执行 ready 方法
  if ( document.addEventListener ){
            document.removeEventListener( "DOMContentLoaded",
  DOMContentLoaded, false );
            jQuery.ready();
  }else if ( document.readyState === "complete" ) {
            document.detachEvent( "onreadystatechange", DOMContentLoaded );
            jQuery.ready();
  }
};
```

在 jQuery 中完整的代码如下所示。

```
jQuery.ready.promise = function( obj ) {
  if ( !readyList ) {
            readyList = jQuery.Deferred();
                    //表示页面已经加载完成，直接调用 ready 方法
            if ( document.readyState === "complete" ) {
                    //将 jQuery.ready 压入异步消息队列，设置延迟时间 1 毫秒（注意，有些浏览器延迟不能小于 4 毫秒）
                    setTimeout( jQuery.ready);
            }
            else if ( document.addEventListener ) //
            {
                     //监听 DOM 加载完成
                    document.addEventListener( "DOMContentLoaded",
DOMContentLoaded, false );
                     //这里是为了确保所有 ready 执行结束，如果 DOMContentLoaded 方法执行了，将有—个状态值 isReady 被设置为 true,因此，
```

```
                    //ready 方法一旦执行，那么将只执行一次，window.addEventListener
中的 ready 将被 return 中断
                    window.addEventListener( "load", jQuery.ready, false );
            } else {
                    //低版本的 IE 浏览器
                    document.attachEvent( "onreadystatechange",
DOMContentLoaded );
                    window.attachEvent( "onload", jQuery.ready );
                    var top = false;
                    try {
                            top = window.frameElement == null &&
document.documentElement;
                    } catch(e) {
                    }
                    if ( top && top.doScroll )   //剔除 iframe 的成分
                    {
                            (function doScrollCheck() {
                                    if ( !jQuery.isReady ) {

                                            try {
                                                    //根据 bug 来兼容低版本的 IE
http://javascript.nwbox.com/IEContentLoaded/
                                                    top.doScroll("left");
                                            } catch(e) {
                                                    //由于低版本的 IE 浏览器，
onreadystatechange 事件不可靠，因此需要根据各个 bug 来判断页面是否已加载完成
                                                    return
setTimeout( doScrollCheck, 50 );
                                            }
                                            jQuery.ready();
                                    }
```

```
                                    })();
                            }
                    }
            }

return readyList.promise( obj );
};
```

需要的时候，在我们调用 ready 函数的时候，才需要注册这些判断页面是否完全加载的处理，如下所示：

```
ready: function( wait ){
  if ( wait === true ? ——jQuery.readyWait : jQuery.isReady ) {
            //判断页面是否已完成加载并且是否已经执行 ready 方法
            return;
  }
  if ( !document.body ) {
            return setTimeout( jQuery.ready );
  }
  jQuery.isReady = true; //指示 ready 方法已被执行
  if ( wait !== true && ——jQuery.readyWait > 0 ) {
            return;
  }
  readyList.resolveWith( document, [ jQuery ] );
  if ( jQuery.fn.trigger ) {
            jQuery( document ).trigger("ready").off("ready");
  }
}
```

总结：

页面加载完成有两种事件，—是 ready，表示文档结构已经加载完成（不包含图片等非文字媒体文件），二是 onload，指示页 面包含图片等文件在内的所有元素都加载完成。(可以说：ready 在 onload 前加载！！！)
—般样式控制的，比如图片大小控制放在 onload 里面加载;
jS 事件触发的方法，可以在 ready 里面加载;

一百七十九、 懒加载的实现原理?

意义：懒加载的主要目的是作为服务器前端的优化，减少请求数或延迟请求数。

实现原理：先加载一部分数据，当触发某个条件时利用异步加载剩余的数据，新得到的数据 不会影响原有数据的显示，同时最大程度上减少服务器端的资源耗用。

实现方式：

1. 第一种是纯粹的延迟加载，使用 setTimeOut 或 setInterval 进行加载延迟.
2. 第二种是条件加载，符合某些条件，或触发了某些事件才开始异步下载。
3. 第三种是可视区加载，即仅加载用户可以看到的区域，这个主要由监控滚动条来实现， 一 般会在距用户看到某图片前一定距离便开始加载，这样能保证用户拉下时正好能看到图片。

一百八十、双向数据绑定和单向数据的区别?

1．单向数据流中，父组件给子组件传递数据，但反过来不可以传递，也就是说单向数据流是从最外层节点传递到子节点，他们只需从最外层节点获取 props 渲染即可，如果顶层组件的 某个 prop 改变了，React 会递归的向下便利整棵组件树，重新渲染所有使用这个属性的组件， React 组件内部还具有自己的状态，这些状态只能在组件内修改；双向数据绑定是数据与视图 双向绑定，数据发生改变时，视图也改变，视图发生改变时，数据也会发生改变。

2．双向数据绑定的各种数据相互依赖相互绑定，导致数据问题的源头难以被跟踪到；单向 数据流的数据流动方向可以跟踪，流动单一，追查问题的时候可以更快捷，缺点是写起来不太方便，要使视图发生改变就得创建各种 action 来维护 state。

3．双向绑定把数据变更的操作隐藏在框架内部，调用者并不会直接感知。而在践行单向数 据流的 flux 系的实现中，其实不过是在全局搞了一个单例的事件分发器（dispatcher)，开发者 必须显式地通过这个统一的事件机制做数据变更通知。

一百八十一、 怎么实现一个类似于 const 功能的方法?

es6 中 const 相当于声明常量不可更改，我们利用 defineProperty 可以模拟实现；我们把 writable 设置为 false 的时候，该属性就成了只读，也就满足了常量了性质，我们把常量封装 在 CONST 命名空间里面，但是因为我们依然可以通过修改属性 writable 为 true 修改属性值，所以 configurable 设置为 false,不能修改属性；

模拟：

如下代码 CONST.a 相当于 es6 中 cont a=2; CONST.a 是不可以更改的常量；

```
var CONST = { };
Object.defineProperty(CONST, 'a', {
        value: 2,
        writable: false,
        configurable: false,
        enumerable: true //可枚举
});
```

```
console.log(CONST.a);      //2
CONST.a = 3;
console.log(CONST.a);      //2
```

一百八十二、 使用原生 js 模拟一个 apply 方法

apply 方法：

语法：apply([thisObj[,argArray]])

定义：应用某一对象的一个方法，用另一个对象替换当前对象。

说明：

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。

```
Function.prototype.apply = function (context, arr) {
    var context = Object(context) || window;
    context.fn = this;
    var result;
    if (!arr) {
        result = context.fn();
    }
    else {
        var args = [];
        for (var i = 0, len = arr.length; i < len; i++) {
            args.push('arr[' + i + ']');
        }
        result = eval('context.fn(' + args + ')')
    }
    delete context.fn
    return result;
}
```

一百八十三、 使用原生 js 模拟一个 call 方法

call()方法在使用一个指定的 this 值和若干个指定的参数值的前提下调用某个函数或方法。

```
Function.prototype.call = function (context) {
```

```
    var context = context || window;
    context.fn = this;
    var args = [];
    for(var i = 1, len = arguments.length; i < len; i++) {
        args.push('arguments[' + i + ']');
    }
    var result = eval('context.fn(' + args +')');
    delete context.fn
    return result;
}
```

以上两个方法的具体实现原理可以参考：
https://juejin.im/post/5907eb99570c3500582ca23c

一百八十四、 object.create()和直接创建对象有什么区别？

Object.create()方法创建一个拥有指定原型和若干个指定属性的对象
//Object.create(proto,[propertiesObject])

该方法创建一个对象，其接受两个参数，第一个参数是这个对象的原型对象 proto,

第二个是一个可选参数，用以对对象的属性做进一步描述

如果 proto 参数不是 null 或一个对象值，则抛出一个 TypeError 异常

var objl = Object.create({

x: 1,

y: 2

}); //对象 obj1 继承了属性 x 和 y

varobj2 = Object.create(null);//对象 obj2 没有原型

对象字面量是创建对象最简单的一种形式，

目的是在于简化创建包含大量属性的对象的过程。

对象字面量由若干属性名(keys)和属性值(values)成对组成的映射表，

key 和 value 中间使用冒号(:)分隔，

每对 key/value 之间使用逗号(,)分隔，

整个映射表用花括号({})括起来。

在用字面量来创建对象的时候，对象中的 property 定义可以用单引号或双引号来包括，也可以忽略引号。不过，当 property 中出现空格、斜杠等特殊字符，或者使用的 property 与 JS 关键词冲突时，则必须使用引号。

var obj = {

property_1: value_1,// property—# 可能是一个标识符...

2: value_2, //或者是一个数字 "property n": value_n // 或是一个字符串

}

通过对象字面量创建的对象复用性较差，

使用 Object.create()创建对象时不需要定义一个构造函数就允许你在对象中选择其原型对象。

一百八十五、 使用 forin 遍历对象和使用 Object.keys 来遍历对象有什么区别？

1．for in 主要用于遍历对象的可枚举属性，包括自有属性、继承自原型的属性

2．Object.keys 返回一个数组，元素均为对象自有的可枚举属性

3．Object.getOwnProperty 用于返回对象的自有属性，包括可枚举的和不可枚举的

```
var obj = {
"name":"xiaosan",
"age":23
}
Object.defineProperty(obj,"height",{value:178,enumerable:false})
Object.prototype.prototypel = function(){
console.log('aaa')
}
Object.prototype.prototype2 = 'bbb';
//for in
for(var i in obj){
console.log(i); //name age prototypel prototype2
}
//Object.keys
console.log(Object.keys(obj)) //name age
//Object.getOwnProperty
console.log(Object.getOwnPropertyNames(obj)) //name age height
```

一百八十六、 深拷贝和浅拷贝以及应用场景

i.浅拷贝

//拷贝就是把父对象的属性，全部拷贝给子对象。

```
var Chinese = {
    nation:'中国’
}
Var Doctor={
    career:'医生'
```

```
}
function extendCopy(p) {
    var c = {};
    for (var i in p) {
        c[i] = p[i];
    }
    c.uber = p; return c;
}
//使用的时候，这样写：
Doctor = extendCopy(Chinese);
Doctor.career ='医生';
alert(Doctor.nation); //  中国
```

//但是，这样的拷贝有一个问题。那就是，如果父对象的属性等于数组或另一个对象，那么实际上，子对象获得的只是一个内存地址，而不是真正拷贝，因此存在父对象被篡改的可能。

//请看，现在给 Chinese 添加一个"出生地"属性，它的值是一个数组。

Chinese.birthPlaces=['北京','上海','香港'];

//通过 extendCopy()函数，Doctor 继承了 Chinese。

Doctor= extendCopy(Chinese);

//然后，我们为 Doctor 的"出生地"添加一个城市：

Doctor.birthPlaces.push('厦门');

//看一下输入结果

alert(Doctor.birthPlaces); //北京，上海，香港，厦门 alert(Chinese.birthPlaces); //北京，上海，香港，厦门

//结果是两个的出生地都被改了。

//所以，extendCopy()只是拷贝了基本类型的数据，我们把这种拷贝叫做"浅拷贝〃。

2.深拷贝

//所谓"深拷贝"，就是能够实现真正意义上的数组和对象的拷贝。它的实现并不难， 只要递归调用"浅拷贝"就行了。

```
var Chinese = {
    nation:'中国’
}
var Doctor = {
    career:'医生'
}
```

```
function deepCopy(p, c) {
    var c = c || {};
    for (var i in p) {
        if (typeof p[i] === 'object') {
            c[i] = (p[i].constructor === Array) ? [] : {}; deepCopy(p[i], c[i]);
        } else {
            c[i] = p[i];
        }
    }
    return c;
}
//看一下使用方法：
Doctor = deepCopy(Chinese);
```

//现在，给父对象加一个属性，值为数组。然后，在子对象上修改这个属性：

Chinese.birthPlaces=['北京','上海','香港'];

Doctor.birthPlaces.push('厦门')；

alert(Doctor.birthPlaces); //北京，上海，香港，厦门

alert(Chinese.birthPlaces); //北京，上海，香港

JavaScript 中的对象一般是可变的（Mutable),因为使用了引用赋值，新的对象简单的引用了原始对象，改变新的对象将影响到原始对象。如'foo={a: 1}; bar=foo; bar.a=2'你会发现此时 'foo.a'也被改成了 '2'。

虽然这样做可以节约内存，但当应用复杂后，这就造成了非常大的隐患，Mutable 带来的优点变得得不偿失。为了解决这个问题，一般的做法是使用 shallowCopy (浅拷贝）或 deepCopy (深拷贝）来避免被修改，但这样做造成了 CPU 和内存的浪费。

Immutable 可以很好地解决这些问题。

一百八十七、 原型链，闭包与继承

闭包的好处：

1．不会污染全局环境；

2．可以进行形参的记忆，减少形参的个数,延长形参生命周期；

```
functionadd(x) {
    returnfunction(y) {
        return (x+y);
    }
}
varsum = add(2);
sum(5);//结果为 7
```

3．方便进行模块化开发；

```
varmodule= (function() {
    varname= '123';
    functioninit() {
        console.log(name);
    }
    return {
        getname:init
    }
})()
module.getname();//结果为 123;
```

继承：一个构造函数继承另一个构造函数中的方法;可以省去大量的重复。

```
function Man(name,age) {
    this.name = name; this.age = age;
}
var person = new Man('tom',19);
function Woman(name,age) {
    this.sex = 'woman';
    Man.call(this,name,age);
}
Woman.prototype = Man.prototype;
var person1 = new Woman('july',20);
person1.name//结果为 july
person1.age //结果为 20
person1.sex //结果为 woman
```

原型链查找：进行方法调用的时候，会先在实例自身上找，如果没有就去该实例的原型上找。

```
function People() {
    this.name = 'a People';
}
People.prototype.say = function() {
    this.age = '10';
    console.log(this.name,this.age);
}
var person = new People();
```

```
person.say();
```

AngularJS

一百八十八、 说说你对 MVC 和 MVVM 的理解

mvc:

View 传送指令到 Controller

Controller 完成业务逻辑后，要求 Model 改变状态

Model 将新的数据发送到 View，用户得到反馈

所有通信都是单向的。

Angular 它采用双向绑定（data-binding）：View 的变动，自动反映在 ViewModel，反之亦然。

组成部分 Model、View、ViewModel

View：UI 界面

ViewModel：它是 View 的抽象，负责 View 与 Model 之间信息转换，将 View 的 Command 传送到 Model；

Model：数据访问层

一百八十九、 对 bootstrap 的掌握？为什么用 angular+bootstrap 搭建后台管理系统

bootstrap 是一个快速开发的响应式框架，主要是为了快速搭建 ui 界面，bootstrap 的 web 组件和 js 插件对 pc 端开发比较友好，尤其是栅格化系统可以良好兼容浏览器， 低版本浏览器可以使用 bootstrap—responsive 的插件兼容，js 插件有各种回调机制，可以满足自己的多样开发需求，而且 bootstrap 使用 css 属性来操作样式，免去了手写原生代码的痛苦，使用 angular 进行数据绑定，bootstrap 来搭建界面，提升开发效率

个人心得：

我在实际开发中使用 ace admin 这套基于 boostrap 的框架，可以更快速的开发，数据项通过 json 结构进行配置，几乎不用手写代码，提升开发效率

一百九十、angular 中 ng—if 和 ng-show/hide 有什么区别？

1．ng—if 在后面表达式为 true 的时候才创建这个 dom 节点，ng—show 是初始时就创建了，用 display:block 和 display:none 来控制显示和不显示。

2．ng—if 会（隐式地）产生新作用域，ng—switch .ng—include 等会动态创建一块界面的也是如此。

个人心得：

ng—if 添加删除节点，那么肯定回创建作用域，而 ng—show/hide 则不会

一百九十一、 Angular 中 ng—click 中写的表达式，可以用 js 原生上 的方法吗？为什么？

ng—click 和原生事件完成的功能是一样的，但是 ng—click 做了优化，而且 ng—click

里面可以写表达式，运算过程，click 则要单独处理，手写功能。

个人心得：

如果不在作用域里添加函数，可以配合 ng—init 初始化属性值，在 ng—click 里添加算法或者某一功能，虽然 ng—inK 不推荐使用，但是侧面说明 ng—click 的优势

一百九十二、 内置 filter 都有哪些？

ng 内置的 filter 有九种：

date（日期）

currency（货币）

limitTo（限制数组或字符串长度）

orderBy（排序）

lowercase（小写）

uppercase（大写）

number（格式化数字，加上千位分隔符，并接收参数限定小数点位数）

json（格式化 json 对象）

filter（处理一个数组，过滤出含有某个子串的元素）

filter 有两种使用方法，一种是直接在页面里：

{{now | date : 'yyyy—MM—dd'}}

另一种是在 js 里面用：

// $filter('过滤器名称')(需要过滤的对象, 参数 1, 参数 2,...)

$filter('date')(now, 'yyyy—MM—dd hh:mm:ss');

一百九十三、 如何自定义 filter?

在模块下挂在一个 filter()方法，第一个参数传入过滤器的名字，第二个参数是回调函数，处理过滤方法的详细内容，最后返回结果，这样外部就可以根据过滤器的名字调用了

例如

```
myAppModule.filter("reverse"，function(){
    return   function(input,uppercase){
        var out ="";
        for(var i=0 ; i<input.length; i++){
            out = input.charAt(i)+out;
        }
        if(uppercase){
            out = out.toUpperCase();
```

```
        }
        return out;
    }
});
```

使用：name | reverse 通过管道符调用

一百九十四、 factory、service 和 provider 是什么关系？

factory, service, provider 都是 angular 提供的服务

factory 就是原生 js 里的方法，一个简单的函数

service 类似原生里构造函数的过程，拥有一个构造器 constructor，也就是说有 new 的过程，追加属性和方法都是在 this 上追加的

provider 是服务商当 service 需要配置的时候，需要使用 provider 提供服务，例如当使用 angular 进行跨域访问，需要配置 jsonp 信息的时候，就可以使用 provider 进行 config 的配置，简单理解是 service 的高级版本，provider 提供一个$get 的属性来返回 $provider 的实例

他们都是单例模式，只实例化一次

个人理解 ：

provider> service> factory

factory 用来配置简单的服务

service 是在 factory 的基础之上加入了面向对象的思想，提供更多功能的服务

provider 是在 service 的基础上进一步改进配置信息

factory 与 service 在底层代码上都来源于 provider

例子介绍：

我可以在 factory 里写一个$http( )请求，不做任何配置，参数写死

我可以在 service 里写一个$http( )请求，传入请求的参数可以先配置在 this 的属性上传入方法

我可以在 provider 里写一个请求，然后在 config 上传入要配置的参数，URL，method，data 等信息，通过 config 来修改 provider 的参数，再将服务商提供的服务注入控制器 controller

注意事项：

config 里传入的参数是 nameProvider 而不是 name，也就是说你的叫做 myProvider, config 里传入的参数就是 myProviderProvider 而不是 myProvider

一百九十五、 angular 的数据绑定采用什么机制？详述原理

通过$watch 来监听每一次 dom 的变化，然后$digest 来遍历循环所有的$watch 队列，发现与原来不同的值，也就是脏值则进行修改，最后通知$apply, $apply 会进入 angularcontext 的执行环境，通知浏览器拿回控制权，修改相应的 dom 节点

个人心得：

每一个 ng 指令的触发都在内部触发了一个$Watch 的队列，加入一组标签

<ling—repeat="iteminitems">

{{ item }}

循环了 10 次，那么就触发了 10 个 item 与 1 个 ng—repeat 的 11 个$watch 的队列，

$digest 会遍历循环这些队列，比较值的变化，有变化的即为脏值过程叫做 dirty—checking，$digest 修改完对应的值就会通知$apply()准备进入 angularcontext 的执行阶段修改 dom，没有变化则不修改。也就是说我们在页面每次触发的操作，每次输入一个文字都会触发$watch，可见于 react 相比 angular 的劣势出现了

一百九十六、 两个平级界面块 a 和 b，如果 a 中触发一个事件，有哪些方式能让 b 知道？详述原理

1．通过 a 的子 controller 将事件使用$emit 传递给父 controller 再将事件用 $broadcast 传递给 bcontroller 实现数据传递

2．也可以通过 service 服务，将数据保存在 service 之内，然后在 b 中调用 service

个人心得：

像这种数据传递的方式其实有很多种，本质是不同作用于之间的数据传递，只要 掌握住这一点思想有很多方式解决，比如我可以尝试挂在$rootScope 之上进行共 享，也可以用本地存储来存储数据，实现数据共享。方法不重要，关键是如何解决的思路。

一百九十七、 一个 angular 应用应当如何良好地分层？

目录结构的划分

对于小型项目，可以按照文件类型组织，比如：

```
css
js
    controllers
    models
    services
    filters
templates
```

但是对于规模较大的项目，最好按业务模块划分，比如：

```
css
    modules
        account
        controllers
        models
        services
        filters
```

templates

disk

controllers

models

services

filters

templates

modules 下最好再有一个 common 目录来存放公共的东西

一百九十八、 angular 应用常用哪些路由库，各自的区别是什 么？

ng—router，ui—touter，ui—router 可以嵌套子视图

一百九十九、 如果通过 angular 的 directive 规划一套全组件化体系，可能遇到哪些挑战？

隔离作用域，ng—指令的作用域传递

二百、 分属不同团队进行开发的 angular 应用，如果要做整合，可能会遇到哪些问题，如何解决？

可能会遇到不同模块之间的冲突。比如一个团队所有的开发在 moduleA 下进行， 另一团队开发的代码在 moduleB 下

```
angular.module('myApp.moduleA'，[ ])
.factory('serviceA', function(){
.....
})
angular.module('myApp.moduleB'，[ ])
.factory('serviceA'，function(){
.....
})
angular.module('myApp'，['myApp.moduleA'，'myApp.moduleB'])
```

会导致两个 module 下面的 serviceA 发生了覆盖。

个人心得：没有太好的解决方案，只能约定命名规范

二百〇一、angular 的缺点有哪些？

不适合做交互过多的项目，因为没有选择器的存在，

导致学习成本较高，对前端不友好。但遵守 AngularJS 的约定时，生产力会很高，对 Java 程序员友好

因为所有内容都是动态获取并渲染生成的，搜索引擎没法爬取

二百〇二、如何看待 angularl.2 中引入的 controller as 语 法？

为 angular 添加 this 作用域链，使得 angular 更加像原声写法

二百〇三、详述 angular 的“依赖注入”。

依赖注入是一个在组件中给出的替代了硬的组件内的编码它们的依赖关系的软件设计模式。这减轻一个组成部分，从定位的依赖，依赖配置。这有助于使组件可重用，维护和测试。

AngularJS 提供了一个至高无上的依赖注入机制。它提供了一个可注入彼此依赖 constantvaluefactoryserviceprovide 核心组件。

二百〇四、当你简单的动态给页面插入 html 时，此时 html 带有 angular 的语法不会执行的，为什么？

通过$compile 进行处理，任何指令的生效都需要 compile，这一步在 app 启动的时候 angular 先帮你做了，但你插入的 html 是没有经过 compile 这个步骤的，所以你手动 compile 下即可。

二百〇五、使用 ng—repeat 出错：Error: [ngRepeat:dupes]，怎么回事？

Error: [ngRepeat:dupes]这个出错提示具体意思是指数组中有 2 个以上的相同数字。ngRepeat 不允许 collection 中存在两个相同 Id 的对象

对于数字对象来说，它的 id 就是它自身的值，因此，数组中是不允许存在两个相同的数字的。为了规避这个错误，需要定义自己的 track by 表达式。

例如：在 ng—repeat="itmeinitems"中加入 rackbyitem.id 或者 trackbyfnCustomld(item)。 嫌麻烦的话，直接拿循环的索引变量$index 来用 iteminitemstrackby $index——>ng—repeat="itmeinitemstrackby $index"

一句话总结：因为 angular 不允许数组中出现重复的值，所以会报错 dupes 错误，意思是重复的参数错误

二百〇六、Ng—repeat 迭代数组的时候，如果数组中有相同值，会有什么问题，如何解决？

见上题

二百〇七、使用第三方插件或者原生的 js 修改 angular 中的 model 或者 view 的值时，相应的 model 或者 view 的值 不会变化，也就是 angular 的双向数据绑定失效，怎么回事？

angular 有自己的一个上下文，所有与 angular 有关的代码执行（如双向数据绑定）都在这个上下文中进行，因此如果你用第三方插件或者原生的 js 进行操作时，此时代码是在 javascript 的上下文中执行，angular 无法知道你是否修改 model 或者 view 的值，自然也就无法进行双向数据绑定。

解决方案是在操作之后执行$scope.$apply( )或者将操作的代码放在 $scope.$apply(function( ){//操作的代码...})

二百〇八、angular 中注入方式有推断式注入、$inject 注入、内联式注入，当然这三种方式在 angular 中是等效的，但推断式注入对于压缩的 JavaScript 代码来说是不起作用的，为什么？

因为压缩过后的 JavaScript 代码重命名了函数的参数名。在压缩 js 代码的时候尽量不要用推断式注入，最佳是用内联式注入的方式。

二百〇九、如何看待 angular2?

相比 Angularl.x，Angular2 的改动很大，几乎算是一个全新的框架。

基于 TypeScript(可以使用 TypeScript 进行开发），在大型项目团队协作时，强语言类型更有利。

组件化，提升开发和维护的效率。

还有 module 支持动态加载，newrouter，promise 的原生支持等等。

迎合未来标准，吸纳其他框架的优点，值得期待，不过同时要学习的东西也更多了(ESnext、TS、Rx 等）

详细参考：

http://www.tuicool.com/articles/ymmq2mf

http://www.cnblogs.eom/laixiangran/p/4938732.html

综合问题

二百一十、请列举你知道的前端框架？常用的前端开发工具？ 开发过哪些应用和组件？

(1) 前端框架

bootstrap/jQuery/zepto/backbone/AngularJS/vue.js/React/

ReactNative/小程序

(2) 前端开发工具 gulp/webpack/git/svn/npm/linux

架构工具 ：bower、npm、yeoman、gulp、webpack

(3) 应用和组件

根据自己做的项目对答

二百一十一、 项目测试没问题。但是放到线上就有问题了，你是怎么分析解决的？

可能的原因：

(1) 后端原因：后端接口，后端服务器

(2) 域名、IP 和路径问题

(3) 网络环境问题

(4) 线上库、框架、工具的版本和本地不一致问题

(5) 线上和本地数据资源不一致问题

(6) 程序 bug

二百一十二、 ES6 里面你用过什么？

1) 块作用域—let

2) 衡量—const

3) 解构数组—ArrayDestructuring

4) 解构对象—ObjectDestructuring

5) 模板字符串—TemplateStrings

6) 展开操作符

7) 剩余操作符

8) 解构参数

9) 箭头函数

10) 对象表达式

11) 对象属性名

12) 对比两个值是否相等

13) 把对象的值复制到另一个对象里

14) 设置对象的 prototype

15) __proto__

16) supper

17) 迭代器

18) class 类

19) getset

20) 静态方法

21) 继承

22) 模块化

细节参见：http://es6.ruanyifeng.com/

二百一十三、 如何管理团队？

(1) 区分技术和管理角色在意识上的差异

(2) 时间管理

(3) 同时管理自己和其他人的代码

(4) 赢得团队的尊敬

详细参见：http://www.t262.com/read/187780.html

二百一十四、 你做过的你负责的最难的数据交互模块是?

根据自己做的项目对答。

二百一十五、 你平时写过什么业务逻辑?

根据自己做的项目对答。

二百一十六、 在项目开发过程中你负责的具体是什么模块?

根据自己做的项目对答。

二百一十七、 如果需要你加班，你会加吗，抵触吗?

其实你肯定抵触，但你肯定要回答如果项目需要肯定会加

二百一十八、 一个小项目让你自己负责搭建底层一些架构，你能胜任吗?

例：我肯定愿意尝试，并做到最优的选择方案出来

二百一十九、 如果项目拖太久，你情绪低落或者厌烦了怎么调节?

例：你结合自身挑着好听的说就行，就像聊天

二百二十、你建议自己造轮子，还是利用开源的轮子?

例：根据实际情况而定，如果开源完全满足 可以自己二次开发就好，大大缩短开发周期如果实在没有契合度很高的，可以花费几个工作日尝试造轮。

注：

开放性问题（一般面试中我们都避免不了），这些问题往往决定你是否最终被录用或者等到终轮面试，技术点回答错了不要紧，人脑不是机器，是可以恶补的。但如果你没有思想和独到的思路，基础挖的再深，可能也打动不了面试官，因为比你基础好的一大堆，但每个人的个性思想却是不同的，因此我们除了在学会知识点的基础上，还要懂得一些为人处事，注意我们平时的言行举止，不要因小失大；