第二版：JavaScript 96 道

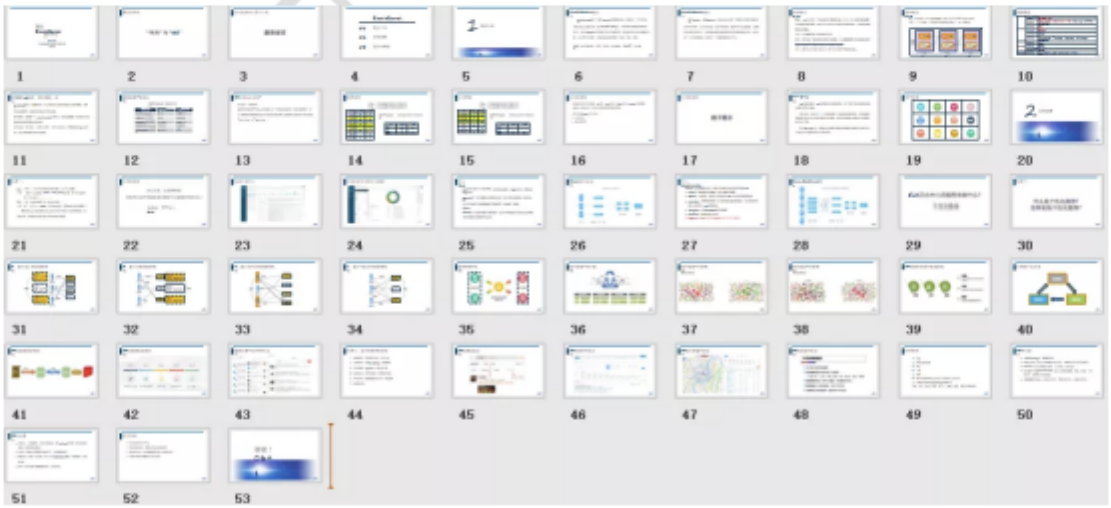
目录

我们的网站：https://tech.souyunku.com

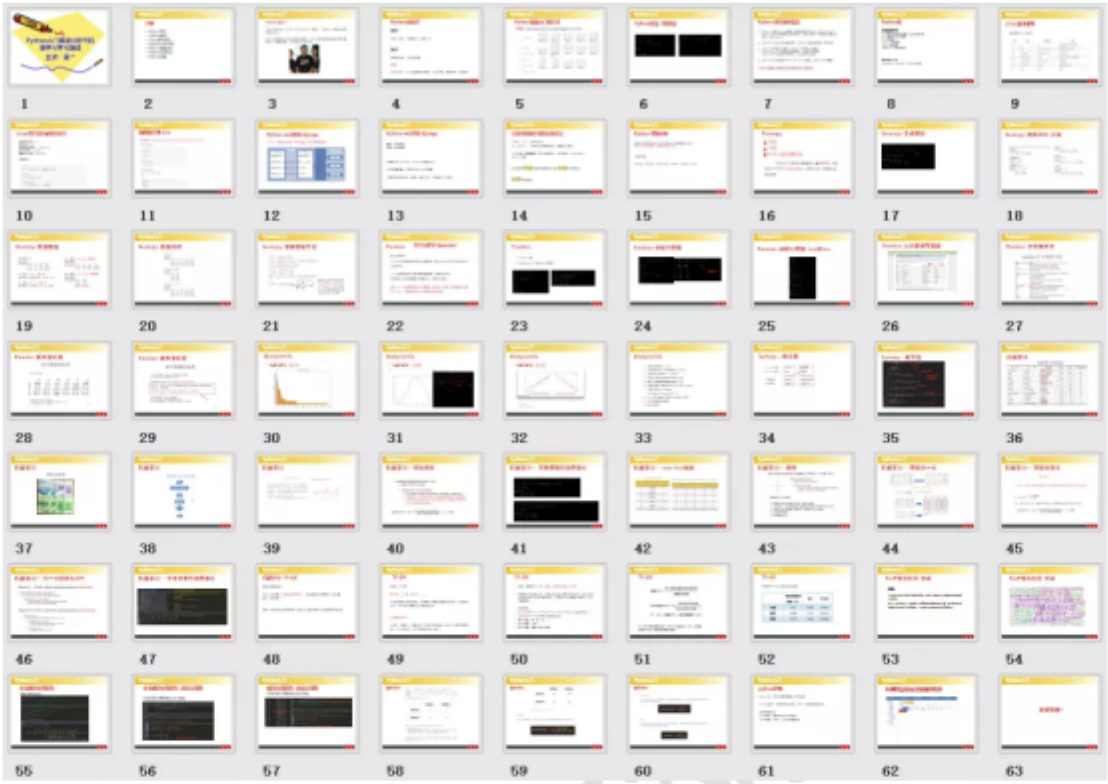
关注我们的公众号：搜云库技术团队，回复以下关键字

46 页《JVM 深度调优，演讲 PPT》

53 页《Elasticsearch 调优演讲 PPT》



63 页《Python 数据分析入门 PPT》



1、几种基本数据类型?复杂数据类型?值类型和引用数据类型?堆栈数据结构?###

基本数据类型：Undefined、Null、Boolean、Number、String
值类型：数值、布尔值、null、undefined。
引用类型：对象、数组、函数。
堆栈数据结构：是一种支持后进先出(LIFO)的集合,即后被插入的数据,先被取出!
js 数组中提供了以下几个方法可以让我们很方便实现堆栈：
shift:从数组中把第一个元素删除，并返回这个元素的值。
unshift: 在数组的开头添加一个或更多元素，并返回新的长度
push:在数组的中末尾添加元素，并返回新的长度
pop:从数组中把最后一个元素删除，并返回这个元素的值。

2、声明函数作用提升?声明变量和声明函数的提升有什么区别?###

(1) 变量声明提升：变量申明在进入执行上下文就完成了。
只要变量在代码中进行了声明，无论它在哪个位置上进行声明， js 引擎都会将它的声明放在范围作用域的顶部；

(2) 函数声明提升：执行代码之前会先读取函数声明，意味着可以把函数申明放在调用它的语句后面。
只要函数在代码中进行了声明，无论它在哪个位置上进行声明， js 引擎都会将它的声明放在范围作用域的顶部；

(3) 变量 or 函数声明：函数声明会覆盖变量声明，但不会覆盖变量赋值。

同一个名称标识 a，即有变量声明 var a，又有函数声明 function a() {}，不管二者声明的顺序，函数声明会覆盖变量声明，也就是说，此时 a 的值是声明的函数 function a() {}。注意：如果在变量声明的同时初始化 a，或是之后对 a 进行赋值，此时 a 的值变量的值。eg: var a; var c = 1; a = 1; function a() { return true; } console.log(a);

3、判断数据类型?###

typeof 返回的类型都是字符串形式，可以判断 function 的类型；在判断除 Object 类型的对象时比较方便。
判断已知对象类型的方法： instanceof，后面一定要是对象类型，并且大小写不能错，该方法适合一些条件选择或分支。

4、异步编程?

方法 1：回调函数，优点是简单、容易理解和部署，缺点是不利于代码的阅读和维护，各个部分之间高度耦合（Coupling），流程会很混乱，而且每个任务只能指定一个回调函数。

方法 2：时间监听，可以绑定多个事件，每个事件可以指定多个回调函数，而且可以“去耦合”（Decoupling），有利于实现模块化。缺点是整个程序都要变成事件驱动型，运行流程会变得很不清晰。

方法 3：发布/订阅，性质与“事件监听”类似，但是明显优于后者。

方法 4：Promises 对象，是 CommonJS 工作组提出的一种规范，目的是为异步编程提供统一接口。
简单说，它的思想是，每一个异步任务返回一个 Promise 对象，该对象有一个 then 方法，允许指定回调函数。

5、事件流?事件捕获？事件冒泡?

事件流：从页面中接收事件的顺序。也就是说当一个事件产生时，这个事件的传播过程，就是事件流。

IE 中的事件流叫事件冒泡；事件冒泡：事件开始时由最具体的元素接收，然后逐级向上传播到较为不具体的节点（文档）。对于 html 来说，就是当一个元素产生了一个事件，它会把这个事件传递给它的父元素，父元素接收到了之后，还要继续传递给它的上一级元素，就这样一直传播到 document 对象（亲测现在的浏览器到 window 对象，只有 IE8 及下不这样

事件捕获是不太具体的元素应该更早接受到事件，而最具体的节点应该最后接收到事件。他们的用意是在事件到达目标之前就捕获它；也就是跟冒泡的过程正好相反,以 html 的 click 事件为例,document 对象(DOM 级规范要求从 document 开始传播，但是现在的浏览器是从 window 对象开始的）最先接收到 click 事件的然后事件沿着 DOM 树依次向下传播，一直传播到事件的实际目标；

6、如何清除一个定时器?###

```
window.clearInterval();
window.clearTimeout();
```

7、如何添加一个 dom 对象到 body 中?innerHTML 和 innerText 区别?###

```
body.appendChild(dom 元素);
```

innerHTML:从对象的起始位置到终止位置的全部内容,包括 Html 标签。

innerText:从起始位置到终止位置的内容，但它去除 Html 标签

分别简述五个 window 对象、属性

成员对象

window.event window.document window.history

window.screen window.navigator window.external

Window 对象的属性如下:

window //窗户自身

window.self //引用本窗户 window=window.self

window.name //为窗户命名

window.defaultStatus //设定窗户状态栏信息

window.location //URL 地址,配备布置这个属性可以打开新的页面

8、数据持久化技术(ajax)?简述 ajax 流程###

1)客户端产生 js 的事件

2)创建 XMLHttpRequest 对象

3)对 XMLHttpRequest 进行配置

4)通过 AJAX 引擎发送异步请求

5)服务器端接收请求并且处理请求，返回 html 或者 xml 内容

6)XML 调用一个 callback()处理响应回来的内容

7)页面局部刷新

9、回调函数?###

回调函数就是一个通过函数指针调用的函数。如果你把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，我们就说这

是回调函数。回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应。

10.什么是闭包?* 堆栈溢出有什么区别? 内存泄漏? 那些操作会造成内存泄漏? 怎么样防止内存泄漏?

闭包：就是能够读取其他函数内部变量的函数。
堆栈溢出：就是不顾堆栈中分配的局部数据块大小，向该数据块写入了过多的数据，导致数据越界，结果覆盖了别的数据。经常会在递归中发生。
内存泄露是指：用动态存储分配函数内存空间，在使用完毕后未释放，导致一直占据该内存单元。直到程序结束。指任何对象在您不再拥有或需要它之后仍然存在。

造成内存泄漏：
setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。
闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）
防止内存泄露：
1、不要动态绑定事件；
2、不要在动态添加，或者会被动态移除的 dom 上绑事件，用事件冒泡在父容器监听事件；
3、如果要违反上面的原则，必须提供 destroy 方法，保证移除 dom 后事件也被移除，这点可以参考 Backbone 的源代码，做的比较好；
4、单例化，少创建 dom，少绑事件。

11.平时工作中怎么样进行数据交互?如果后台没有提供数据怎么样进行开发?mock 数据与后台返回的格式不同意怎么办?###

由后台编写接口文档、提供数据接口实、前台通过 ajax 访问实现数据交互；
在没有数据的情况下寻找后台提供静态数据或者自己定义 mock 数据；
返回数据不统一时编写映射文件 对数据进行映射。

12 简述 ajax 执行流程###

基本步骤:

```
var xhr =null;//创建对象
if(window.XMLHttpRequest){
        xhr = new XMLHttpRequest();
}else{
        xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
xhr.open(“方式”,”地址”,”标志位”);//初始化请求
    xhr.setRequestHeader(“”,”“);//设置 http 头信息
xhr.onreadystatechange =function(){}//指定回调函数
xhr.send();//发送请求
```

13.自执行函数?用于什么场景? 好处?###

自执行函数:1、声明一个匿名函数 2、马上调用这个匿名函数。
作用：创建一个独立的作用域。

好处：防止变量弥散到全局，以免各种 js 库冲突。隔离作用域避免污染，或者截断作用域链，避免闭包造成引用变量无法释放。利用立即执行特性，返回需要的业务函数或对象，避免每次通过条件判断来处理

场景：一般用于框架、插件等场景

14.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 的标记语言。

1、XHTML 元素必须被正确地嵌套。

2、XHTML 元素必须被关闭。

3、标签名必须用小写字母。

4、空标签也必须被关闭。

5、XHTML 文档必须拥有根元素。

15. 什么是构造函数？与普通函数有什么区别?###

构造函数：是一种特殊的方法、主要用来创建对象时初始化对象，总与 new 运算符一起使用，创建对象的语句中构造函数的函数名必须与类名完全相同。

与普通函数相比只能由 new 关键字调用，构造函数是类的标示

16. 通过 new 创建一个对象的时候，函数内部有哪些改变###

```
function Person(){}
Person.prototype.friend = [];
Person.prototype.name = '';
// var a = new Person();
```

```
// a.friend[0] = '王琦';
// a.name = '程娇';
// var b = new Person();
// b.friend?
// b.name?
```

1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。
2、属性和方法被加入到 this 引用的对象中。
3、新创建的对象由 this 所引用，并且最后隐式的返回 this 。

17.事件委托？有什么好处?###

（1）利用冒泡的原理，把事件加到父级上，触发执行效果
（2）好处：新添加的元素还会有之前的事件；提高性能。

18.window.onload ==? DOMContentLoaded ?###

一般情况下，DOMContentLoaded 事件要在 window.onload 之前执行，当 DOM 树构建完成的时候就会执行 DOMContentLoaded 事件，而 window.onload 是在页面载入完成的时候，才执行，这其中包括图片等元素。大多数时候我们只是想在 DOM 树构建完成后，绑定事件到元素，我们并不需要图片元素，加上有时候加载外域图片的速度非常缓慢。

19.节点类型?判断当前节点类型?###

1、 元素节点
2、 属性节点
3、 文本节点

8、 注释节点

9、 文档节点

通过 nodeObject.nodeType 判断节点类型：其中，nodeObject 为 DOM 节点（节点对象）。该属性返回以数字表示的节点类型，例如，元素节点返回 1，属性节点返回 2 。

20.如何合并两个数组？数组删除一个元素?###

```
//三种方法。
  (1) var arr1=[1,2,3];
                var arr2=[4,5,6];
                arr1 = arr1.concat(arr2);
                console.log(arr1);
  (2) var arr1=[1,2,3];
                var arr2=[4,5,6];
                Array.prototype.push.apply(arr1,arr2);
                console.log(arr1);
  (3) var arr1=[1,2,3];
                var arr2=[4,5,6];
                for (var i=0; i < arr2.length; i++) {
                      arr1.push( arr2[i] );
                }
                console.log(arr1);
```

21.强制转换 显式转换 隐式转换?###

```
//强制类型转换:
Boolean(0)                          // =false - 零
            Boolean(new object())     // =true - 对象
```

```
                Number(undefined)          // =   NaN
                Number(null)               // =0
                String(null)               // ="null"
parseInt( )
parseFloat( )
JSON.parse( )
JSON.stringify ( )
```

隐式类型转换：

在使用算术运算符时，运算符两边的数据类型可以是任意的，比如，一个字符串可以和数字相加。之所以不同的数据类型之间可以做运算，是因为 JavaScript 引擎在运算之前会悄悄的把他们进行了隐式类型转换的

（例如：x+"" //等价于 String(x)

+x //等价于 Number(x)

x-0 //同上

!!x //等价于 Boolean(x),是双叹号)

显式转换：

如果程序要求一定要将某一类型的数据转换为另一种类型，则可以利用强制类型转换运算符进行转换，这种强制转换过程称为显示转换。

显示转换是你定义让这个值类型转换成你要用的值类型，是底到高的转换。例 int 到 float 就可以直接转，int i=5,想把他转换成 char 类型，就用显式转换（char）i

22. Jq 中如何实现多库并存?###

Noconfict 多库共存就是“$ ”符号的冲突。

方法一： 利用 jQuery 的实用函数$.noConflict();这个函数归还$的名称控制权给另一个库，因此可以在页面上使用其他库。这时，我们可以用"jQuery "这个名称调用 jQuery 的功能。 $.noConflict();

```
jQuery('#id').hide();
.....
//或者给 jQuery 一个别名
var $j=jQuery
$j('#id').hide();
.....
```

方法二： (function($){})(jQuery)

方法三： jQuery(function($){})

通过传递一个函数作为 jQuery 的参数，因此把这个函数声明为就绪函数。 我们声明$为就绪函数的参数，因为 jQuery 总是吧 jQuery 对象的引用作为第一个参数传递，所以就保证了函数的执行。

23.Jq 中 get 和 eq 有什么区别?

get() :取得其中一个匹配的元素。num 表示取得第几个匹配的元素，get 多针对集合元素，返回的是 DOM 对象组成的数组 eq():获取第 N 个元素，下标都是从 0 开始，返回的是一个 JQuery 对象

24.如何通过原生 js 判断一个元素当前是显示还是隐藏状态?###

```
if( document.getElementById("div").css("display")==='none')
if( document.getElementById("div").css("display")==='block')
```

```
$("#div").is(":hidden"); // 判断是否隐藏
$("#div").is(":visible")
```

25.Jq 如何判断元素显示隐藏?

```
//第一种：使用 CSS 属性
var display =$('#id').css('display');
if(display == 'none'){      alert("我是隐藏的！"); }
//第二种：使用 jquery 内置选择器
<div id="test"<p>仅仅是测试所用</p></div>
if($("#test").is(":hidden")){            $("#test").show();      //如果元素为隐藏,则将它显现 }else{          $("#test").hide();       //如果元素为显现,则将其隐藏 }
//第三种：jQuery 判断元素是否显示 是否隐藏
var node=$('#id');
if(node.is(':hidden')){      //如果 node 是隐藏的则显示 node 元素，否则隐藏
    node.show();
}else{
    node.hide();
}
```

26.移动端上什么是点击穿透?###

点击穿透现象有 3 种：

点击穿透问题：点击蒙层（mask）上的关闭按钮，蒙层消失后发现触发了按钮下面元素的 click 事件跨页面点击穿透问题：如果按钮下面恰好是一个有 href 属性的 a 标签，那么页面就会发生跳转另一种跨页面点击穿透问题：这次没有 mask 了，直接点击页内按钮跳转至新页，然后发现新页面中对应位置元素的 click 事件被触发了

解决方案：

1、只用 touch

最简单的解决方案，完美解决点击穿透问题

把页面内所有 click 全部换成 touch 事件（ touchstart 、' touchend' 、' tap' ）

2、只用 click

下下策，因为会带来 300ms 延迟，页面内任何一个自定义交互都将增加 300 毫秒延迟

3、tap 后延迟 350ms 再隐藏 mask

改动最小，缺点是隐藏 mask 变慢了，350ms 还是能感觉到慢的

4、pointer-events

比较麻烦且有缺陷， 不建议使用 mask 隐藏后，给按钮下面元素添上 pointer-events: none; 样式，让 click 穿过去，350ms 后去掉这个样式，恢复响应缺陷是 mask 消失后的的 350ms 内，用户可以看到按钮下面的元素点着没反应，如果用户手速很快的话一定会发现

27.Jq 绑定事件的几种方式？ on bind ?###

jQuery 中提供了四种事件监听方式，分别是 bind、live、delegate、on，对应的解除监听的函数分别是 unbind、die、undelegate、off

Bind( )是使用频率较高的一种，作用就是在选择到的元素上绑定特定事件类型的监听函数;

Live( )可以对后生成的元素也可以绑定相应的事件,处理机制就是把事件绑定在 DOM 树的根节点上，而不是直接绑定在某个元素上;

Delegate( )采用了事件委托的概念，不是直接为子元素绑定事件，而是为其父元素（或祖先元素也可）绑定事件，当在 div 内任意元素上点击时，事件会一层层从 event target 向上冒泡，直至到达你为其绑定事件的元素；

on( )方法可以绑定动态添加到页面元素的事件，on()方法绑定事件可以提升效率；

28.Jq 中如何将一个 jq 对象转化为 dom 对象?

方法一：
jQuery 对象是一个数据对象，可以通过[index]的方法，来得到相应的 DOM 对象。
如： var $v =$("#v") ; //jQuery 对象
var v=$v[0]; //DOM 对象
alert(v.checked) //检测这个 checkbox 是否被选中

方法二：
jQuery 本身提供，通过.get(index)方法，得到相应的 DOM 对象
如： var $v=$("#v"); //jQuery 对象
var v=$v.get(0); //DOM 对象
alert(v.checked) //检测这个 checkbox 是否被选中

29.Jq 中有几种选择器?分别是什么?###

层叠选择器、基本过滤选择器、内容过滤选择器、可视化过滤选择器、属性过滤选择器、子元素过滤选择器、表单元素选择器、表单元素过滤选择器

30.Jq 中怎么样编写插件?###

```
//第一种是类级别的插件开发:
```

```
//1.1 添加一个新的全局函数 添加一个全局函数，我们只需如下定义：
jQuery.foo = function() {
      alert('This is a test. This is only a test.');   };

//1.2 增加多个全局函数 添加多个全局函数，可采用如下定义：
jQuery.foo = function() {
         alert('This is a test. This is only a test.');   };
jQuery.bar = function(param) {
        alert('This function takes a parameter, which is "' + param + '".');   };
调用时和一个函数的一样的:jQuery.foo();jQuery.bar();或者
$.foo();$.bar('bar');
//1.3 使用 jQuery.extend(object);
jQuery.extend({
        foo: function() {
             alert('This is a test. This is only a test.');
          },
        bar: function(param) {
             alert('This function takes a parameter, which is "' + param
+'".');
          }
      });
//1.4 使用命名空间
// 虽然在 jQuery 命名空间中，我们禁止使用了大量的 javaScript 函数名和变
量名。
// 但是仍然不可避免某些函数或变量名将于其他 jQuery 插件冲突，因此我们习
惯将一些方法
// 封装到另一个自定义的命名空间。
jQuery.myPlugin = {
foo:function() {
  alert('This is a test. This is only a test.');
 },
```

```
 bar:function(param) {
  alert('This function takes a parameter, which is "' + param + '".');
 }
};
//采用命名空间的函数仍然是全局函数，调用时采用的方法：
$.myPlugin.foo();
$.myPlugin.bar('baz');
//通过这个技巧（使用独立的插件名），我们可以避免命名空间内函数的冲突。

//第二种是对象级别的插件开发
//形式 1：
(function($){
   $.fn.extend({
    pluginName:function(opt,callback){
                    // Our plugin implementation code goes here.
    }
   })
})(jQuery);

//形式 2：
(function($) {
    $.fn.pluginName = function() {
            // Our plugin implementation code goes here.
    };
})(jQuery);
//形参是$，函数定义完成之后,把 jQuery 这个实参传递进去.立即调用执行。
//这样的好处是,我们在写 jQuery 插件时,也可以使用$这个别名,而不会与
prototype 引起冲突
```

31.$('div+.ab')和$('.ab+div') 哪个效率高?

$('div+.ab')效率高

32.$.map 和$.each 有什么区别###

map()方法主要用来遍历操作数组和对象，会返回一个新的数组。$.map()方法适用于将数组或对象每个项目新阵列映射到一个新数组的函数；
each()主要用于遍历 jquery 对象，返回的是原来的数组，并不会新创建一个数组。

33.编写一个 getElementsByClassName 封装函数?###

```
<body
<input type="submit" id = "sub" class="ss confirm btn" value="提交"/
<scriptwindow.onload = function(){
//方法一
    var Opt = document.getElementById('sub');
    var getClass = function(className,tagName){
        if(document.getElementsByTagName){
            var Inp = document.getElementsByTagName(tagName);
            for(var i=0; i<Inp.length; i++){
                if((new RegExp('(\\s|^)' +className
+'(\\s|$)')).test(Inp[i].className)){
                    return Inp[i];
                }
            }
        }else if(document.getElementsByClassName){
            return
document.getElementsByClassName(className);
        }
```

```
    }
//方法二
    var aa = getClass("confirm", "input");
        function getClass(className, targetName){
            var ele = [];
            var all = document.getElementsByTagName(targetName ||
"*");
            for(var i=0; i<all.length; i++){
                if(all[i].className.match(new
RegExp('(\\s|^)'+confirm+'(\\s|$)'))){
                    ele[ele.length] = all[i];
                }
            }
            return ele;
        }
//方法三
    function getObjsByClass(tagName, className){
            if(document.getElementsByClassName){
                alert("document.getElementsByClassName");
                return document.getElementsByClassName(className);
            }else{
                var el = [];
                var _el = document.getElementsByTagName(tagName);
                for(var i=0; i<_el.length; i++){
                    if(_el[i].className.indexOf(className) -1){
                        alert(_el[i]);
                        el[_el.length] = _el[i];
                    }
                }
                alert(el);
                return el;
```

```
            }
        }
    }
</script>
</body>
```

34.简述下工作流程###

我在之前的公司工作流程大概是这样的：公司定稿会结束以后，会进行简单的技术研讨，然后我们前端会进行先期的技术准备。前端切图人员会进行 psd 设计稿切图，并且将 css 文件进行整合。我们主要编写 JS 部分，其中包括搭建前端框架（大项目），编写 js 业务和数据持久化操作，我们也会编写 js 插件并且进行封装方便使用，还有就是编写 JS 前端组建和 JS 测试单元，最后将完成的 JS 部分与切图人员提供的 HTML 页面进行整合。最后对完成的页面进行功能测试、页面兼容、产品还原。然后对产品进行封存，提交测试。如果出现 BUG 会返回给我们开发人员进行修改，再提交测试，最后测试成功，进行版本封存。等到程序全部上线的时候进行线上测试。

35.一般使用什么版本控制工具?svn 如何对文件加锁###

svn 加锁目的：为了避免多个人同一时间对同一个文件改动的相互覆盖，版本控制系统就必须有一套冲突处理机制。

svn 加锁两种策略：乐观加锁：所有签出的文件都是可读写的，对文件的修改不必获得文件的锁，当你修改完文件签入时，会首先要求你更新本地文件，版本控制系统不会覆盖你的本地修改，而是会让你自己合并冲突后签入。

严格加锁：所有签出的文件都是只读的，任何对文件的修改必须要获得文件的锁，如果其他人没有拥有该文件的锁，那么版本控制系统就会授权给你文件的锁，并将文件设置为可编辑的。

svn 两种加锁步骤：乐观加锁：选择你想要获取锁定的文件，然后右键菜单点击 TortoiseSVN 选取获取锁定。

严格加锁：在想要采取严格加锁的文件或目录上点击右键，使用 TortoiseSVN 属性菜单，点击新建属性，选择需要锁定。

36. git 和 svn 的区别?###

SVN 是集中式版本控制系统，版本库是集中放在中央服务器的，而干活的时候，用的都是自己的电脑，所以首先要从中央服务器哪里得到最新的版本，然后干活，干完后，需要把自己做完的活推送到中央服务器。集中式版本控制系统是必须联网才能工作，如果在局域网还可以，带宽够大，速度够快，如果在互联网下，如果网速慢的话，就纳闷了。

Git 是分布式版本控制系统，那么它就没有中央服务器的，每个人的电脑就是一个完整的版本库，这样，工作的时候就不需要联网了，因为版本都是在自己的电脑上。既然每个人的电脑都有一个完整的版本库，那多个人如何协作呢？比如说自己在电脑上改了文件 A，其他人也在电脑上改了文件 A，这时，你们两之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。

37. jquery 和 zepto 有什么区别?###

1、针对移动端程序，Zepto 有一些基本的触摸事件可以用来做触摸屏交互（tap 事件、swipe 事件），Zepto 是不支持 IE 浏览器的，这不是 Zepto 的开发者 Thomas Fucks 在跨浏览器问题上犯了迷糊，而是经过了认真考虑后为了降低文件尺寸而做

出的决定，就像 jQuery 的团队在 2.0 版中不再支持旧版的 IE（6 7 8）一样。因为 Zepto 使用 jQuery 句法，所以它在文档中建议把 jQuery 作为 IE 上的后备库。那样程序仍能在 IE 中，而其他浏览器则能享受到 Zepto 在文件大小上的优势，然而它们两个的 API 不是完全兼容的，所以使用这种方法时一定要小心，并要做充分的测试。

2、Dom 操作的区别：添加 id 时 jQuery 不会生效而 Zepto 会生效。

3、zepto 主要用在移动设备上，只支持较新的浏览器，好处是代码量比较小，性能也较好。
jquery 主要是兼容性好，可以跑在各种 pc，移动上，好处是兼容各种浏览器，缺点是代码量大，同时考虑兼容，性能也不够好。

38. $(function(){})和 window.onload 和 $(document).ready(function(){})###

window.onload:用于当页面的所有元素，包括外部引用文件，图片等都加载完毕时运行函数内的函数。load 方法只能执行一次，如果在 js 文件里写了多个，只能执行最后一个。

$(document).ready(function(){})和$(function(){})都是用于当页面的标准 DOM 元素被解析成 DOM 树后就执行内部函数。这个函数是可以在 js 文件里多次编写的，对于多人共同编写的 js 就有很大的优势，因为所有行为函数都会执行到。而且$(document).ready()函数在 HMTL 结构加载完后就可以执行，不需要等大型文件加载或者不存在的连接等耗时工作完成才执行，效率高。

39. Jq 中 attr 和 prop 有什么区别###

对于 HTML 元素本身就带有的固有属性，在处理时，使用 prop 方法。

对于 HTML 元素我们自己自定义的 DOM 属性，在处理时，使用 attr 方法。

40. 简述下 this 和定义属性和方法的时候有什么区别?Prototype?

this 表示当前对象，如果在全局作用范围内使用 this，则指代当前页面对象 window； 如果在函数中使用 this，则 this 指代什么是根据运行时此函数在什么对象上被调用。 我们还可以使用 apply 和 call 两个全局方法来改变函数中 this 的具体指向。

prototype 本质上还是一个 JavaScript 对象。 并且每个函数都有一个默认的 prototype 属性。

在 prototype 上定义的属性方法为所有实例共享，所有实例皆引用到同一个对象，单一实例对原型上的属性进行修改，也会影响到所有其他实例。

41. 什么是预编译语音|预编译处理器?###

Sass 是一种 CSS 预处理器语言，通过编程方式生成 CSS 代码。因为可编程，所以操控灵活性自由度高，方便实现一些直接编写 CSS 代码较困难的代码。

同时，因为 Sass 是生成 CSS 的语言，所以写出来的 Sass 文件是不能直接用的，必须经过编译器编译成 CSS 文件才能使用。

CSS 预处理器是一种语言用来为 CSS 增加一些编程的的特性，无需考虑浏览器的兼容性问题，例如你可以在 CSS 中使用变量、简单的程序逻辑、函数等等在编

程语言中的一些基本技巧，可以让你的 CSS 更见简洁，适应性更强，代码更直观等诸多好处。最常用的 css 预处理器有 sass、less css、 stylus。

42.ajax 和 jsonp ？

ajax 和 jsonp 的区别：

相同点：都是请求一个 url

不同点：ajax 的核心是通过 xmlHttpRequest 获取内容

jsonp 的核心则是动态添加