# 七、JS高级

## JQuery一个对象可以同时绑定多个事件，这是如何实现的？

jQuery可以给一个对象同时绑定多个事件，低层实现方式是使用addEventListner或attachEvent兼容不同的浏览器实现事件的绑定，这样可以给同一个对象注册多个事件。

## 知道什么是webkit么? 知道怎么用浏览器的各种工具来调试和debug代码么?

Webkit是浏览器引擎，包括html渲染和js解析功能，手机浏览器的主流内核，与之相对应的引擎有Gecko（Mozilla Firefox 等使用）和Trident（也称MSHTML，IE 使用）。

对于浏览器的调试工具要熟练使用，主要是页面结构分析，后台请求信息查看，js调试工具使用，熟练使用这些工具可以快速提高解决问题的效率

## 如何测试前端代码? 知道BDD, TDD, Unit Test么? 知道怎么测试你的前端工程么(mocha, sinon, jasmin, qUnit..)?

了解BDD行为驱动开发与TDD测试驱动开发已经单元测试相关概念，

## 4、前端templating(Mustache, underscore, handlebars)是干嘛的, 怎么用?

Web 模板引擎是为了使用户界面与业务数据（内容）分离而产生的，

Mustache 是一个 logic-less （轻逻辑）模板解析引擎，它的优势在于可以应用在 Javascript、PHP、Python、Perl 等多种编程语言中。

Underscore封装了常用的JavaScript对象操作方法，用于提高开发效率。

Handlebars 是 JavaScript 一个语义模板库，通过对view和data的分离来快速构建Web模板。

# 5、简述一下 Handlebars 的基本用法？

没有用过的话说出它是干什么的即可

官网：<http://handlebarsjs.com/>

参考：J:\代码,PPT,笔记,电子书\面试题\handlebarDemo

## 6、简述一下 Handlerbars 的对模板的基本处理流程， 如何编译的？如何缓存的？

学习技术不仅要会用，还有熟悉它的实现机制，这样在开发中遇到问题时才能更好的解决

## 7、用js实现千位分隔符?

原生js的熟练度，实践经验，实现思路

## 8、检测浏览器版本版本有哪些方式？

IE与标准浏览器判断，IE不同版本的判断，userAgent var ie = /\*@cc\_on !@\*/false;

## 9、我们给一个dom同时绑定两个点击事件，一个用捕获，一个用冒泡，你来说下会执行几次事件，然后会先执行冒泡还是捕获

对两种事件模型的理解

## 10、实现一个函数clone，可以对JavaScript中的5种主要的数据类型（包括Number、String、Object、Array、Boolean）进行值复制

* 考察点1：对于基本数据类型和引用数据类型在内存中存放的是值还是指针这一区别是否清楚
* 考察点2：是否知道如何判断一个变量是什么类型的
* 考察点3：递归算法的设计

|  |  |
| --- | --- |
|  | // 方法一：  Object.prototype.clone = function(){     var o = this.constructor === Array ? [] : {};     for(var e in this){       o[e] = typeof this[e] === "object" ? this[e].clone() : this[e];     }     return o;  }  //方法二：     /\*\*       \* 克隆一个对象       \* @param Obj       \* @returns       \*/      function clone(Obj) {          var buf;          if (Obj instanceof Array) {              buf = [];//创建一个空的数组              var i = Obj.length;              while (i--) {                  buf[i] = clone(Obj[i]);              }              return buf;          }else if (Obj instanceof Object){              buf = {};//创建一个空对象              for (var k in Obj) { //为这个对象添加新的属性                  buf[k] = clone(Obj[k]);              }              return buf;          }else{ //普通变量直接赋值              return Obj;          }      } |

## 11、如何消除一个数组里面重复的元素？

|  |  |
| --- | --- |
|  | var arr=[1,2,3,3,4,4,5,5,6,1,9,3,25,4];          function deRepeat(){              var newArr=[];              var obj={};              var index=0;              var l=arr.length;              for(var i=0;i<l;i++){                  if(obj[arr[i]]==undefined)                    {                      obj[arr[i]]=1;                      newArr[index++]=arr[i];                    }                  else if(obj[arr[i]]==1)                    continue;              }              return newArr;          }          var newArr2=deRepeat(arr);          alert(newArr2); //输出1,2,3,4,5,6,9,25 |

## 12、小贤是一条可爱的小狗(Dog)，它的叫声很好听(wow)，每次看到主人的时候就会乖乖叫一声(yelp)。从这段描述可以得到以下对象：

|  |  |
| --- | --- |
|  | function Dog() {        this.wow = function() {                 alert(’Wow’);        }        this.yelp = function() {                this.wow();        }  } |

小芒和小贤一样，原来也是一条可爱的小狗，可是突然有一天疯了(MadDog)，一看到人就会每隔半秒叫一声(wow)地不停叫唤(yelp)。请根据描述，按示例的形式用代码来实。（继承，原型，setInterval）

|  |  |
| --- | --- |
|  | function MadDog() {      this.yelp = function() {            var self = this;            setInterval(function() {                  self.wow();            }, 500);        }  }  MadDog.prototype = new Dog();  //for test  var dog = new Dog();  dog.yelp();  var madDog = new MadDog();  madDog.yelp(); |

## 13、下面这个ul，如何点击每一列的时候alert其index?（闭包）

|  |  |
| --- | --- |
|  | <ul id=”test”>  <li>这是第一条</li>  <li>这是第二条</li>  <li>这是第三条</li>  </ul> |
|  | // 方法一：  var lis=document.getElementById('2223').getElementsByTagName('li');  for(var i=0;i<3;i++)  {      lis[i].index=i;      lis[i].onclick=function(){          alert(this.index);      };  }  //方法二：  var lis=document.getElementById('2223').getElementsByTagName('li');  for(var i=0;i<3;i++){      lis[i].index=i;      lis[i].onclick=(function(a){          return function() {              alert(a);          }      })(i);  } |

## 14、编写一个JavaScript函数，输入指定类型的选择器(仅需支持id，class，tagName三种简单CSS选择器，无需兼容组合选择器)可以返回匹配的DOM节点，需考虑浏览器兼容性和性能。

/\*\*\* @param selector {String} 传入的CSS选择器。\* @return {Array}\*/

|  |  |
| --- | --- |
|  | var query = function(selector) {  var reg = /^(#)?(\.)?(\w+)$/img;  var regResult = reg.exec(selector);  var result = [];  //如果是id选择器  if(regResult[1]) {  if(regResult[3]) {  if(typeof document.querySelector === "function") {  result.push(document.querySelector(regResult[3]));      }else {        result.push(document.getElementById(regResult[3]));      }     }     }     //如果是class选择器     else if(regResult[2]) {      if(regResult[3]) {         if(typeof document.getElementsByClassName === 'function') {           var doms = document.getElementsByClassName(regResult[3]);           if(doms) {             result = converToArray(doms);           }         }       //如果不支持getElementsByClassName函数       else {         var allDoms = document.getElementsByTagName("\*") ;         for(var i = 0, len = allDoms.length; i < len; i++) {           if(allDoms[i].className.search(new RegExp(regResult[2])) > -1) {             result.push(allDoms[i]);           }         }       }    }  }    //如果是标签选择器    else if(regResult[3]) {      var doms = document.getElementsByTagName(regResult[3].toLowerCase());      if(doms) {        result = converToArray(doms);      }    }    return result;    }    function converToArray(nodes){      var array = null;      try{        array = Array.prototype.slice.call(nodes,0);//针对非IE浏览器      }catch(ex){       array = new Array();      for( var i = 0 ,len = nodes.length; i < len ; i++ ) {        array.push(nodes[i])      }    }    return array;  } |

## 15、请评价以下代码并给出改进意见。

|  |  |
| --- | --- |
|  | if(window.addEventListener){      var addListener = function(el,type,listener,useCapture){          el.addEventListener(type,listener,useCapture);    };  }  else if(document.all){      addListener = function(el,type,listener){          el.attachEvent("on"+type,function(){            listener.apply(el);        });     }  } |

* 不应该在if和else语句中声明addListener函数，应该先声明；
* 不需要使用window.addEventListener或document.all来进行检测浏览器，应该使用能力检测；
* 由于attachEvent在IE中有this指向问题，所以调用它时需要处理一下

改进如下：

|  |
| --- |
| function addEvent(elem, type, handler){  　　if(elem.addEventListener){  　　　　elem.addEventListener(type, handler, false);  　　}else if(elem.attachEvent){  　　　　elem['temp' + type + handler] = handler;  　　　　elem[type + handler] = function(){  　　　　elem['temp' + type + handler].apply(elem);  　　};  　　elem.attachEvent('on' + type, elem[type + handler]);    }else{  　　elem['on' + type] = handler;  　　}  } |

## 16、给String对象添加一个方法，传入一个string类型的参数，然后将string的每个字符间价格空格返回，例如：

addSpace(“hello world”) // -> ‘h e l l o  w o r l d’

|  |  |
| --- | --- |
|  | String.prototype.spacify = function(){        return this.split('').join(' ');      }; |

接着上述问题答案提问，1）直接在对象的原型上添加方法是否安全？尤其是在Object对象上。(这个我没能答出？希望知道的说一下。)　2）函数声明与函数表达式的区别？

答案：在js中，解析器在向执行环境中加载数据时，对函数声明和函数表达式并非是一视同仁的，解析器会率先读取函数声明，并使其在执行任何代码之前可用（可以访问），至于函数表达式，则必须等到解析器执行到它所在的代码行，才会真正被解析执行。

## 17、定义一个log方法，让它可以代理console.log的方法。

可行的方法一：

|  |
| --- |
| function log(msg)　{      console.log(msg);  }  log("hello world!") // hello world! |

如果要传入多个参数呢？显然上面的方法不能满足要求，所以更好的方法是：

|  |
| --- |
| function log(){      console.log.apply(console, arguments);  }; |

到此，追问apply和call方法的异同。

对于apply和call两者在作用上是相同的，即是调用一个对象的一个方法，以另一个对象替换当前对象。将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。

但两者在参数上有区别的。对于第一个参数意义都一样，但对第二个参数： apply传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而call则作为call的参数传入（从第二个参数开始）。 如 func.call(func1,var1,var2,var3)对应的apply写法为：func.apply(func1,[var1,var2,var3]) 。

## 18、在Javascript中什么是伪数组？如何将伪数组转化为标准数组？

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，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getElementsByTagName,document.childNodes之类的,它们都返回NodeList对象都属于伪数组。可以使用Array.prototype.slice.call(fakeArray)将数组转化为真正的Array对象。

假设接第八题题干，我们要给每个log方法添加一个”(app)”前缀，比如’hello world!’ ->’(app)hello world!’。方法如下：

|  |
| --- |
| function log(){        var args = Array.prototype.slice.call(arguments);  //为了使用unshift数组方法，将argument转化为真正的数组        args.unshift('(app)');        console.log.apply(console, args);      }; |

## 19、对作用域上下文和this的理解，看下列代码：

|  |
| --- |
| var User = {    count: 1,    getCount: function() {      return this.count;    }  };  console.log(User.getCount());  // what?  var func = User.getCount;  console.log(func());  // what? |

问两处console输出什么？为什么？

答案是1和undefined。

func是在winodw的上下文中被执行的，所以会访问不到count属性。

继续追问，那么如何确保Uesr总是能访问到func的上下文，即正确返回1。正确的方法是使用Function.prototype.bind。兼容各个浏览器完整代码如下：

|  |
| --- |
| Function.prototype.bind = Function.prototype.bind || function(context){     var self = this;     return function(){        return self.apply(context, arguments);     };  }  var func = User.getCount.bind(User);  console.log(func()); |

## 20、原生JS的window.onload与Jquery的$(document).ready(function(){})有什么不同？如何用原生JS实现Jq的ready方法？

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

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

|  |
| --- |
| /\*   \* 传递函数给whenReady()   \* 当文档解析完毕且为操作准备就绪时，函数作为document的方法调用   \*/  var whenReady = (function() {               //这个函数返回whenReady()函数      var funcs = [];             //当获得事件时，要运行的函数      var ready = false;          //当触发事件处理程序时,切换为true      //当文档就绪时,调用事件处理程序      function handler(e) {          if(ready) return;       //确保事件处理程序只完整运行一次          //如果发生onreadystatechange事件，但其状态不是complete的话,那么文档尚未准备好          if(e.type === 'onreadystatechange' && document.readyState !== 'complete') {              return;          }          //运行所有注册函数          //注意每次都要计算funcs.length          //以防这些函数的调用可能会导致注册更多的函数          for(var i=0; i<funcs.length; i++) {              funcs[i].call(document);          }          //事件处理函数完整执行,切换ready状态, 并移除所有函数          ready = true;          funcs = null;      }      //为接收到的任何事件注册处理程序      if(document.addEventListener) {          document.addEventListener('DOMContentLoaded', handler, false);          document.addEventListener('readystatechange', handler, false);            //IE9+          window.addEventListener('load', handler, false);      }else if(document.attachEvent) {          document.attachEvent('onreadystatechange', handler);          window.attachEvent('onload', handler);      }      //返回whenReady()函数      return function whenReady(fn) {          if(ready) { fn.call(document); }          else { funcs.push(fn); }      }  })(); |

如果上述代码十分难懂，下面这个简化版：

|  |
| --- |
| function ready(fn){      if(document.addEventListener) {//标准浏览器          document.addEventListener('DOMContentLoaded', function() {              //注销事件, 避免反复触发              document.removeEventListener('DOMContentLoaded',arguments.callee, false);              fn();//执行函数          }, false);      }else if(document.attachEvent) {//IE          document.attachEvent('onreadystatechange', function() {              if(document.readyState == 'complete') {                  document.detachEvent('onreadystatechange', arguments.callee);                  fn();//函数执行              }          });      }  }; |

## 21、（设计题）想实现一个对页面某个节点的拖曳？如何做？（使用原生JS）

回答出概念即可，下面是几个要点

1. 给需要拖拽的节点绑定mousedown, mousemove, mouseup事件
2. mousedown事件触发后，开始拖拽
3. mousemove时，需要通过event.clientX和clientY获取拖拽位置，并实时更新位置
4. mouseup时，拖拽结束
5. 需要注意浏览器边界的情况

## 22、请实现如下功能

[](http://jbcdn2.b0.upaiyun.com/2014/10/a2e3bf5eab2f56af1010d2d0153ed19d.png)

|  |
| --- |
| function setcookie(name,value,days){  //给cookie增加一个时间变量  　　var exp = new Date();  　　exp.setTime(exp.getTime() + days\*24\*60\*60\*1000); //设置过期时间为days天  　　document.cookie = name + "="+ escape (value) + ";expires=" + exp.toGMTString();  }  function getCookie(name){  　　var result = "";  　　var myCookie = ""+document.cookie+";";  　　var searchName = "+name+"=";  　　var startOfCookie = myCookie.indexOf(searchName);  　　var endOfCookie;  　　if(satrtOfCookie != -1){  　　　　startOfcookie += searchName.length;  　　　　endOfCookie = myCookie.indexOf(";",startOfCookie);  　　　　result = (myCookie.substring(startOfCookie,endOfCookie));  　　}  　　return result;  }  (function(){  　　var oTips = document.getElementById('tips');//假设tips的id为tips  　　var page = {  　　check: function(){//检查tips的cookie是否存在并且允许显示  　　　　var tips = getCookie('tips');  　　　　if(!tips || tips == 'show') return true;//tips的cookie不存在  　　　　if(tips == "never\_show\_again") return false;  　　},  　　hideTip: function(bNever){  　　　　if(bNever) setcookie('tips', 'never\_show\_again', 365);  　　　　oTips.style.display = "none";//隐藏  　　},  　　showTip: function(){  　　oTips.style.display = "inline";//显示，假设tips为行级元素  　　},  　　init: function(){  　　　　var \_this = this;  　　　　if(this.check()){  　　　　\_this.showTip();  　　　　setcookie('tips', 'show', 1);  　　}  　　oTips.onclick = function(){  　　　　\_this.hideTip(true);  　　};  　　}  　　};    page.init();  })(); |

## 23、说出以下函数的作用是？空白区域应该填写什么？

|  |
| --- |
| //define  (function(window){      function fn(str){          this.str=str;      }        fn.prototype.format = function(){          var arg = \_\_\_\_\_\_;          return this.str.replace(\_\_\_\_\_,function(a,b){               return arg[b]||"";        });      }      window.fn = fn;  })(window);    //use  (function(){      var t = new fn('<p><a href="{0}">{1}</a><span>{2}</span></p>');      console.log(t.format('http://www.alibaba.com','Alibaba','Welcome'));  })(); |

答案：访函数的作用是使用format函数将函数的参数替换掉{0}这样的内容，返回一个格式化后的结果：

第一个空是：arguments

第二个空是：/\{(\d+)\}/ig

## Javascript作用域链?

理解变量和函数的访问范围和生命周期，全局作用域与局部作用域的区别，JavaScript中没有块作用域，函数的嵌套形成不同层次的作用域，嵌套的层次形成链式形式，通过作用域链查找属性的规则需要深入理解。

## 谈谈this对象的理解。

理解不同形式的函数调用方式下的this指向，理解事件函数、定时函数中的this指向，函数的调用形式决定了this的指向。

## eval是做什么的？

它的功能是把对应的字符串解析成JS代码并运行；应该避免使用eval，不安全，非常耗性能（2个步骤，一次解析成js语句，一次执行）

## 关于事件，IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

[1].在IE中,事件对象是作为一个全局变量来保存和维护的.所有的浏览器事件,不管是用户触发的，还是其他事件,都会更新window.event对象.所以在代码中，只要调用window.event就可以获取事件对象， 再event.srcElement就可以取得触发事件的元素进行进一步处理.

[2].在FireFox中，事件对象却不是全局对象，一般情况下，是现场发生，现场使用，FireFox把事件对象自动传给事件处理程序.

关于事件的兼容性处理要熟练掌握，事件对象具体哪些属性存在兼容性问题，IE与标准事件模型事件冒泡与事件捕获的支持要理解

## 什么是闭包（closure），为什么要用它？

简单的理解是函数的嵌套形成闭包，闭包包括函数本身已经它的外部作用域

使用闭包可以形成独立的空间，延长变量的生命周期，报存中间状态值

## 29、javascript 代码中的"use strict";是什么意思 ? 使用它区别是什么？

意思是使用严格模式，使用严格模式，一些不规范的语法将不再支持

# 严格模式

链接：<http://www.ruanyifeng.com/blog/2013/01/javascript_strict_mode.html>

**全局变量显式声明**

**静态绑定**

**禁止使用with语句**

**eval中定义的变量都是局部变量**

**禁止this关键字指向全局对象**

**禁止在函数内部遍历调用栈**

严格模式下无法删除变量。只有configurable设置为true的对象属性，才能被删除

正常模式下，对一个对象的只读属性进行赋值，不会报错，只会默默地失败。严格模式下，将报错。

严格模式下，对一个使用getter方法读取的属性进行赋值，会报错。

严格模式下，对禁止扩展的对象添加新属性，会报错。

严格模式下，删除一个不可删除的属性，会报错。

正常模式下，如果对象有多个重名属性，最后赋值的那个属性会覆盖前面的值。严格模式下，这属于语法错误。

正常模式下，如果函数有多个重名的参数，可以用arguments[i]读取。严格模式下，这属于语法错误。

正常模式下，整数的第一位如果是0，表示这是八进制数，比如0100等于十进制的64。严格模式禁止这种表示法，整数第一位为0，将报错。

**不允许对arguments赋值**

**arguments不再追踪参数的变化**

**禁止使用arguments.callee**

严格模式只允许在全局作用域或函数作用域的顶层声明函数。也就是说，不允许在非函数的代码块内声明函数

严格模式新增了一些保留字：implements, interface, let, package, private, protected, public, static, yield。

## 30、如何判断一个对象是否属于某个类(严格来说在ES6之前，js没有类的概念)？

instanceof constructor

## 31、new操作符具体干了什么呢?

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 。

## 32、用原生JavaScript的实现过什么功能吗？

主要考察原生js的实践经验

## 33、Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？

HasOwnProperty

## 34、对JSON的了解？

轻量级数据交互格式，可以形成复杂的嵌套格式，解析非常方便

## 35、js延迟加载的方式有哪些？

方案一：<script>标签的async="async"属性（详细参见：script标签的async属性）

方案二：<script>标签的defer="defer"属性

方案三：动态创建<script>标签

方案四：AJAX eval（使用AJAX得到脚本内容，然后通过eval\_r(xmlhttp.responseText)来运行脚本）

方案五：iframe方式

## 36、模块化开发怎么做？

理解模块化开发模式：浏览器端requirejs，seajs；服务器端nodejs；ES6模块化；fis、webpack等前端整体模块化解决方案；grunt、gulp等前端工作流的使用

## 37、AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

理解这两种规范的差异，主要通过requirejs与seajs的对比，理解模块的定义与引用方式

的差异以及这两种规范的设计原则

参考链接1：<https://www.zhihu.com/question/20351507/answer/14859415>

参考链接2：<https://github.com/seajs/seajs/issues/277>

1、对于依赖的模块，AMD 是提前执行，CMD 是延迟执行。不过 RequireJS 从 2.0 开始，也改成可以延迟执行（根据写法不同，处理方式不同）。CMD 推崇 as lazy as possible.

2、CMD 推崇依赖就近，AMD 推崇依赖前置。

3. AMD 的 API 默认是一个当多个用，CMD 的 API 严格区分，推崇职责单一。比如 AMD 里，require 分全局 require 和局部 require，都叫 require。CMD 里，没有全局 require，而是根据模块系统的完备性，提供 seajs.use 来实现模块系统的加载启动。CMD 里，每个 API 都简单纯粹。

## 38、requireJS的核心原理是什么？（如何动态加载的？如何避免多次加载的？如何 缓存的？）

核心是js的加载模块，通过正则匹配模块以及模块的依赖关系，保证文件加载的先后顺序，根据文件的路径对加载过的文件做了缓存

## 39、让你自己设计实现一个requireJS，你会怎么做？

核心是实现js的加载模块，维护js的依赖关系，控制好文件加载的先后顺序

# 40、谈一谈你对ECMAScript6的了解？

ES6新的语法糖，类，模块化等新特性

关于ES6参考链接：<http://es6.ruanyifeng.com/>

1. [ECMAScript 6简介](http://es6.ruanyifeng.com/#docs/intro)
2. [let和const命令](http://es6.ruanyifeng.com/#docs/let)
3. [变量的解构赋值](http://es6.ruanyifeng.com/#docs/destructuring)
4. [字符串的扩展](http://es6.ruanyifeng.com/#docs/string)
5. [正则的扩展](http://es6.ruanyifeng.com/#docs/regex)
6. [数值的扩展](http://es6.ruanyifeng.com/#docs/number)
7. [数组的扩展](http://es6.ruanyifeng.com/#docs/array)
8. [函数的扩展](http://es6.ruanyifeng.com/#docs/function)
9. [对象的扩展](http://es6.ruanyifeng.com/#docs/object)
10. [Symbol](http://es6.ruanyifeng.com/#docs/symbol)
11. [Proxy和Reflect](http://es6.ruanyifeng.com/#docs/proxy)
12. [二进制数组](http://es6.ruanyifeng.com/#docs/arraybuffer)
13. [Set和Map数据结构](http://es6.ruanyifeng.com/#docs/set-map)
14. [Iterator和for...of循环](http://es6.ruanyifeng.com/#docs/iterator)
15. [Generator函数](http://es6.ruanyifeng.com/#docs/generator)
16. [Promise对象](http://es6.ruanyifeng.com/#docs/promise)
17. [异步操作和Async函数](http://es6.ruanyifeng.com/#docs/async)
18. [Class](http://es6.ruanyifeng.com/#docs/class)
19. [Decorator](http://es6.ruanyifeng.com/#docs/decorator)
20. [Module](http://es6.ruanyifeng.com/#docs/module)

## 41、ECMAScript6 怎么写class么，为什么会出现class这种东西?

class Point {

constructor(x, y) {

this.x = x;

this.y = y;

}

toString() {

return '('+this.x+', '+this.y+')';

}

}

## 42、异步加载的方式有哪些？

方案一：<script>标签的async="async"属性（详细参见：script标签的async属性）

方案二：<script>标签的defer="defer"属性

方案三：动态创建<script>标签

方案四：AJAX eval（使用AJAX得到脚本内容，然后通过eval\_r(xmlhttp.responseText)来运行脚本）

方案五：iframe方式

## 43、documen.write和 innerHTML的区别?

document.write是重写整个document, 写入内容是字符串的html

innerHTML是HTMLElement的属性，是一个元素的内部html内容

## 44、DOM操作——怎样添加、移除、移动、复制、创建和查找节点?

（1）创建新节点

createDocumentFragment() //创建一个DOM片段

createElement\_x() //创建一个具体的元素

createTextNode() //创建一个文本节点

（2）添加、移除、替换、插入

appendChild()

removeChild()

replaceChild()

insertBefore()

（3）查找

getElementsByTagName() //通过标签名称

getElementsByName() //通过元素的Name属性的值

getElementById() //通过元素Id，唯一性

## 45、call() 和 apply() 的含义和区别？

apply的参数是数组形式，call的参数是单个的值，除此之外在使用上没有差别，重点理解这两个函数调用的this改变

## 46、数组和对象有哪些原生方法，列举一下？

Array.concat( ) 连接数组

Array.join( ) 将数组元素连接起来以构建一个字符串

Array.length 数组的大小

Array.pop( ) 删除并返回数组的最后一个元素

Array.push( ) 给数组添加元素

Array.reverse( ) 颠倒数组中元素的顺序

Array.shift( ) 将元素移出数组

Array.slice( ) 返回数组的一部分

Array.sort( ) 对数组元素进行排序

Array.splice( ) 插入、删除或替换数组的元素

Array.toLocaleString( ) 把数组转换成局部字符串

Array.toString( ) 将数组转换成一个字符串

Array.unshift( ) 在数组头部插入一个元素

**Object对象的常用方法**

Object.hasOwnProperty( ) 检查属性是否被继承

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

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

Object.toLocaleString( ) 返回对象的本地字符串表示

Object.toString( ) 定义一个对象的字符串表示

Object.valueOf( ) 指定对象的原始值

## 47、JS 怎么实现一个类。怎么实例化这个类

严格来讲js中并没有类的概念，不过js中的函数可以作为构造函数来使用，通过new来实例化，其实函数本身也是一个对象。

# 48、JavaScript中的作用域与变量声明提升？

理解JavaScript的预解析机制，js的运行主要分两个阶段：js的预解析和运行，预解析阶段所有的变量声明和函数定义都会提前，但是变量的赋值不会提前

## 49、如何编写高性能的Javascript？

使用 DocumentFragment 优化多次 append

通过模板元素 clone ，替代 createElement

使用一次 innerHTML 赋值代替构建 dom 元素

使用 firstChild 和 nextSibling 代替 childNodes 遍历 dom 元素

使用 Array 做为 StringBuffer ，代替字符串拼接的操作

将循环控制量保存到局部变量

顺序无关的遍历时，用 while 替代 for

将条件分支，按可能性顺序从高到低排列

在同一条件子的多（ >2 ）条件分支时，使用 switch 优于 if

使用三目运算符替代条件分支

需要不断执行的时候，优先考虑使用 setInterval

## 50、那些操作会造成内存泄漏？

闭包，死循环

# 51、javascript对象的几种创建方式？

1. 工厂模式

2. 构造函数模式

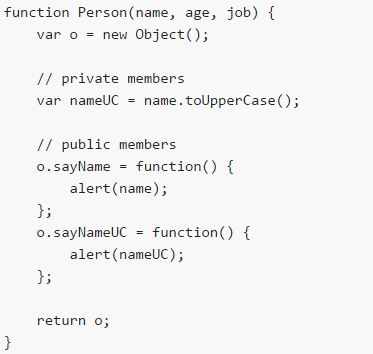
3. 原型模式

4. 混合构造函数和原型模式

5. 动态原型模式

6. 寄生构造函数模式

7. 稳妥构造函数模式



# 52、javascript继承的 6 种方法？

1. 原型链继承

2. 借用构造函数继承

3. 组合继承(原型+借用构造)

4. 原型式继承

5. 寄生式继承

6. 寄生组合式继承

## 53、eval是做什么的？

1. 它的功能是把对应的字符串解析成JS代码并运行

2. 应该避免使用eval，不安全，非常耗性能（2次，一次解析成js语句，一次执行）

# 54、JavaScript 原型，原型链 ? 有什么特点？

1. 原型对象也是普通的对象，是对象一个自带隐式的 \_\_proto\_\_ 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为 null 的话，我们就称之为原型链

2. 原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链

## 55、事件、IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为

2. 事件处理机制：IE是事件冒泡、firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件

3. ev.stopPropagation();

注意旧ie的方法：ev.cancelBubble = true;

## 56、简述一下Sass、Less，且说明区别？

他们是动态的样式语言，是CSS预处理器,CSS上的一种抽象层。他们是一种特殊的语法/语言而编译成CSS。

变量符不一样，less是@，而Sass是$;

Sass支持条件语句，可以使用if{}else{},for{}循环等等。而Less不支持;

Sass是基于Ruby的，是在服务端处理的，而Less是需要引入less.js来处理Less代码输出Css到浏览器

## 57、关于javascript中apply()和call()方法的区别？

相同点:两个方法产生的作用是完全一样的

不同点:方法传递的参数不同

Object.call(this,obj1,obj2,obj3)

Object.apply(this,arguments)

apply()接收两个参数，一个是函数运行的作用域(this)，另一个是参数数组。

call()方法第一个参数与apply()方法相同，但传递给函数的参数必须列举出来。

## 58、简述一下JS中的闭包？

闭包用的多的两个作用：读取函数内部的变量值；让这些变量值始终保存着(在内存中)。

同时需要注意的是：闭包慎用，不滥用，不乱用，由于函数内部的变量都被保存在内存中，会导致内存消耗大。

## 59、说说你对this的理解？

在JavaScript中，this通常指向的是我们正在执行的函数本身，或者是，指向该函数所属的对象。

全局的this → 指向的是Window

~~函数中的this → 指向的是函数所在的对象~~ 错误答案

对象中的this → 指向其本身

事件中this → 指向事件对象

## 60、分别阐述split(),slice(),splice(),join()？

join()用于把数组中的所有元素拼接起来放入一个字符串。所带的参数为分割字符串的分隔符，默认是以逗号分开。归属于Array

split()即把字符串分离开，以数组方式存储。归属于Stringstring

slice() 方法可从已有的数组中返回选定的元素。该方法并不会修改数组，而是返回一个子数组。如果想删除数组中的一段元素，应该使用方法 Array.splice()

splice() 方法向/从数组中添加/删除项目，然后返回被删除的项目。返回的是含有被删除的元素的数组。

## 61、事件委托是什么？

让利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行！

# 62、如何阻止事件冒泡和默认事件？

阻止浏览器的默认行为

window.event?window.event.returnValue=false:e.preventDefault();

停止事件冒泡

window.event?window.event.cancelBubble=true:e.stopPropagation();

原生JavaScript中，return false;只阻止默认行为，不阻止冒泡，jQuery中的return false;既阻止默认行为，又阻止冒泡

## 63、添加 删除 替换 插入到某个接点的方法？

obj.appendChidl()

obj.removeChild()

obj.replaceChild()

obj.innersetBefore()

## 64、你用过require.js吗？它有什么特性？

（1）实现js文件的异步加载，避免网页失去响应；

（2）管理模块之间的依赖性，便于代码的编写和维护。

## 65、谈一下JS中的递归函数，并且用递归简单实现阶乘？

递归即是程序在执行过程中不断调用自身的编程技巧，当然也必须要有一个明确的结束条件，不然就会陷入死循环。

## 66、请用正则表达式写一个简单的邮箱验证。

/^[a-zA-Z0-9\_-]+@[a-zA-Z0-9\_-]+(\.[a-zA-Z0-9\_-]+)+$/;

## 67、简述一下你对web性能优化的方案？

1、尽量减少 HTTP 请求

2、使用浏览器缓存

3、使用压缩组件

4、图片、JS的预载入

5、将脚本放在底部

6、将样式文件放在页面顶部

7、使用外部的JS和CSS

8、精简代码

## 68、在JS中有哪些会被隐式转换为false

Undefined、null、布尔值false、NaN、零、空字符串

## 69、定时器setInterval有一个有名函数fn1，setInterval（fn1,500）与setInterval（fn1(),500）有什么区别？

第一个是重复执行每500毫秒执行一次，后面一个只执行一次。

## 70、外部JS文件出现中文字符，会出现什么问题，怎么解决？

会出现乱码，加charset=”GB2312”;

另一种解决方式：网页文件和外部JS文件都是UTF8编码

## 71、谈谈浏览器的内核，并且说一下什么是内核？

Trident (['traɪd(ə)nt])--IE，Gecko (['gekəʊ])--Firefox, Presto (['prestəʊ])--opera,webkit—谷歌和Safari

浏览器内核又可以分成两部分：渲染引擎和 JS 引擎。它负责取得网页的内容（HTML、XML、图像等等）、整理讯息（例如加入 CSS 等），以及计算网页的显示方式，然后会输出至显示器或打印机。JS 引擎则是解析 Javascript 语言，执行 javascript 语言来实现网页的动态效果。

# 72、JavaScript原型，原型链 ? 有什么特点？

\* 原型对象也是普通的对象，是对象一个自带隐式的 \_\_proto\_\_ 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为null的话，我们就称之为原型链。

\* 原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链。

\* JavaScript的数据对象有那些属性值？

　　writable：这个属性的值是否可以改。

　　configurable：这个属性的配置是否可以删除，修改。

　　enumerable：这个属性是否能在for…in循环中遍历出来或在Object.keys中列举出来。

　　value：属性值。

\* 当我们需要一个属性的时，Javascript引擎会先看当前对象中是否有这个属性， 如果没有的话，就会查找他的Prototype对象是否有这个属性。

function clone(proto) {

　　function Dummy() { }

　　Dummy.prototype = proto;

　　Dummy.prototype.constructor = Dummy;

　　return new Dummy(); //等价于Object.create(Person);

}

function object(old) {

function F() {};

F.prototype = old;

return new F();

}

var newObj = object(oldObject);

# 73、写一个通用的事件侦听器函数

`// event(事件)工具集，来源：https://github.com/markyun

markyun.Event = {

// 页面加载完成后

readyEvent : function(fn) {

if (fn==null) {

fn=document;

}

var oldonload = window.onload;

if (typeof window.onload != 'function') {

window.onload = fn;

} else {

window.onload = function() {

oldonload();

fn();

};

}

},

// 视能力分别使用dom0||dom2||IE方式 来绑定事件

// 参数： 操作的元素,事件名称 ,事件处理程序

addEvent : function(element, type, handler) {

if (element.addEventListener) {

//事件类型、需要执行的函数、是否捕捉

element.addEventListener(type, handler, false);

} else if (element.attachEvent) {

element.attachEvent('on' + type, function() {

handler.call(element);

});

} else {

element['on' + type] = handler;

}

},

// 移除事件

removeEvent : function(element, type, handler) {

if (element.removeEnentListener) {

element.removeEnentListener(type, handler, false);

} else if (element.datachEvent) {

element.detachEvent('on' + type, handler);

} else {

element['on' + type] = null;

}

},

// 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)

stopPropagation : function(ev) {

if (ev.stopPropagation) {

ev.stopPropagation();

} else {

ev.cancelBubble = true;

}

},

// 取消事件的默认行为

preventDefault : function(event) {

if (event.preventDefault) {

event.preventDefault();

} else {

event.returnValue = false;

}

},

// 获取事件目标

getTarget : function(event) {

return event.target || event.srcElement;

},

// 获取event对象的引用，取到事件的所有信息，确保随时能使用event；

getEvent : function(e) {

var ev = e || window.event;

if (!ev) {

var c = this.getEvent.caller;

while (c) {

ev = c.arguments[0];

if (ev && Event == ev.constructor) {

break;

}

c = c.caller;

}

}

return ev;

}

};

## 74、事件、IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。

2. 事件处理机制：IE是事件冒泡、火狐是 事件捕获；

3. ev.stopPropagation();

## 75、什么是闭包（closure），为什么要用？

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

## 76、如何判断一个对象是否属于某个类？

使用instanceof （待完善）

if(a instanceof Person){

alert('yes');

}

## 77、new操作符具体干了什么呢?

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 。

var obj = {};

obj.\_\_proto\_\_ = Base.prototype;

Base.call(obj);

## 78、JSON 的了解

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小

{'age':'12', 'name':'back'}

## 79、js延迟加载的方式有哪些

defer和async、动态创建DOM方式（用得最多）、按需异步载入js

# 80、模块化怎么做？

立即执行函数,不暴露私有成员

1. 使用字面量实现命名空间(YUI)：

Itcast.common.dom={};

Itcast.common.css={};

Itcast.common.event={};

2、使用闭包

var module1 = (function(){

　　　　var \_count = 0;

　　　　var m1 = function(){

　　　　　　//...

　　　　};

　　　　var m2 = function(){

　　　　　　//...

　　　　};

　　　　return {

　　　　　　m1 : m1,

　　　　　　m2 : m2

　　　　};

　　})();

## 81、异步加载的方式

(1) defer，只支持IE

(2) async：

(3) 创建script，插入到DOM中，加载完毕后callBack

documen.write和 innerHTML的区别

document.write只能重绘整个页面

innerHTML可以重绘页面的一部分

## 82、告诉我答案是多少？

(function(x){

delete x;

alert(x);

})(1+5);

函数参数无法delete删除，delete只能删除通过for in访问的属性。

当然，删除失败也不会报错，所以代码运行会弹出“1”。

## 83、JS中的call()和apply()方法的区别？

例子中用 add 来替换 sub，add.call(sub,3,1) == add(3,1) ，所以运行结果为：alert(4);

注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

function add(a,b){

alert(a+b);

}

function sub(a,b){

alert(a-b);

}

add.call(sub,3,1);

## 84、Jquery与jQuery UI 有啥区别？

\*jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。

\*jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。

提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等

## 85、jquery 中如何将数组转化为json字符串，然后再转化回来？

jQuery中没有提供这个功能，所以你需要先编写两个jQuery的扩展：

$.fn.stringifyArray = function(array) {

return JSON.stringify(array)

}

$.fn.parseArray = function(array) {

return JSON.parse(array)

}

然后调用：

$("").stringifyArray(array)

## 86、JavaScript中的作用域与变量声明提升？

其他部分

（HTTP、正则、优化、重构、响应式、移动端、团队协作、SEO、UED、职业生涯）

\*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++) {}

# 87、前端开发的优化问题（看雅虎14条性能优化原则）。

参考资料：J:\代码,PPT,笔记,电子书\面试题\雅虎14条优化规则.docx

（1） 减少http请求次数：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。

（2） 前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数

（3） 用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。

（4） 当需要设置的样式很多时设置className而不是直接操作style。

（5） 少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。

（6） 避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)。

（7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

（8） 避免在页面的主体布局中使用table，table要等其中的内容完全下载之后才会显示出来，显示比div+css布局慢。

## 88、http状态码有那些？分别代表是什么意思？

100-199 用于指定客户端应相应的某些动作。

200-299 用于表示请求成功。

300-399 用于已经移动的文件并且常被包含在定位头信息中指定新的地址信息。

400-499 用于指出客户端的错误。

400 语义有误，当前请求无法被服务器理解。

401 当前请求需要用户验证

403 服务器已经理解请求，但是拒绝执行它。

500-599 用于支持服务器错误。

503 – 服务不可用

## 89、一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？（流程说的越详细越好）

要熟悉前后端的通信流程，最好把动态网站的背后细节也介绍一遍

第二部分：JS面试题

1. 获取一堆name名相同的div，点击弹出显示第几个div

（本题主要考察闭包的用法，是在面试中常被问的题目，不难但是比较容易上当的题目）

var boxes = document.getElementsByTagName("div");

for(var i = 0; i < boxes.length; i++) {

boxes[i].index = i+1;

boxes[i].onclick = function() {

alert(this.index);

}

}

答案二

var boxes = document.getElementsByTagName("div");

for(var i = 0; i < boxes.length; i++) {

(function(i){

boxes[i].onclick = function() {

alert(i+1);

}

})(i);

}

1. abcdefgsdds 获取里面字面最多的字母 （答案有很多）

var str = “abcdefgsdds”;

var obj = {};

var arr = [];

var letter;

for(var i = 0,len = str.length;i<len;i++){

letter = str[i];

if(!obj[letter]){

obj[letter] = 1;

}else{

obj[letter]++;

}

}

var max\_key,max\_num=0;

for(key in obj){

if(max\_num <obj[key]){

max\_num = obj[key];

max\_key = key;

}

}

document.write(“字母：”+max\_key+”　次数：”+max\_num);

1. abcdefghi 获取ghi怎么实现

答：”abcdefghi".substring(6)

1. ‘asdjhklzxc’ 截取’asd’

答：”asdjhklzxc".substring(3,-1)

1. 翻转字符串

答：”asd".split("").reverse().join("")

1. 删除数组第一个元素

答：arr.shift()

1. [1,2,1,3,2,3,5,6,5] 去除重复元素

var arr = [1,2,1,3,2,3,5,6,5] ;

var newArr = [];

for(var i = 0; i < arr.length; i++) {

if(newArr.indexOf(arr[i]) == -1) {

newArr.push(arr[i]);

}

}

alert(newArr)

1. JS中有几种数据类型分别是哪几种？

答：Number,String,Boolean,Object,Null,Undefined

1. 说一下拖拽的原理？怎么样实现多个DIV在拖拽时有相撞让它们改变属性与属性值和他们的位置？（部分程度不太好的同学看到拖拽会懵，项目期有这个情况）

mousedown,mousermove,mouseup事件

e.clientX,e.clientY 鼠标位置

碰撞检测

1. 用JS创建一个a标签，在body中显示，点击在新窗口打开

var alink = document.createElement("a");

alink.innerHTML = "google";

alink.href = "http://www.google.com";

alink.target = "\_blank";

document.body.appendChild(alink);

1. 鼠标滚轮动画

function mousewheel(element,up,down) {

element.onmousewheel = fn;

if(window.addEventListener) {

element.addEventListener("DOMMouseScroll",fn,false)

}

function fn(e) {

var e = e || window.event;

console.log(e);

if(e.wheelDelta > 0 || e.detail < 0) {

up();

}else {

down();

}

e.cancelBubble = true;

e.stopPropagation();

}

}

mousewheel(document,function(){},function(){})//函数调用

1. 定位右下角广告

答：使用css position:fixed

1. 鼠标经过A A变大 其他的位置不动

答：A绝对定位，脱离文档流即可

1. JS中避免命名冲突的三个方法（有些同学或许会想不到这些概念）

答：命名空间，闭包，匿名函数

1. JS在HTML中的引入方式有哪些？又有什么优缺点？

答：a外部引入 b内部嵌套 c行内代码 例：<button onclick=”alert(‘hello’)”>按钮</button> 对比css引入

1. 写出5条标识符的命名规范 （）

答：1.区分大小写；

2.第一个字符必须是一个字母、下划线（\_）或一个美元符号（$）

3..不能含有空格

4.不能以关键字或保留字命名,不能以数字开头

1. 从JS的角度，写出8条以上优化代码的方法？

答：参考 <http://www.haorooms.com/post/web_xnyh_jscss>

1. 按优先级，对这几种运算符进行排序：！ && \* = >

答：！ && \*> =

1. 在循环中，break和continue语句的功能有何不同

答：break 终止整个循环，continue终止当次循环

1. 解释事件冒泡、事件捕获的区别

答：事件流解释即可

1. 说说什么叫做事件委托

答：利用事件冒泡的原理，子元素的事件会冒泡到父元素，可以只给父元素添加事件，通过事件目标判断元素。优点：节省内存，动态添加的子元素也包含事件

1. 列举不少于6条的IE与FF脚本兼容性问题，需要写出命令（也许同学们缺乏总结，看着简单，实际很难答全）

(1) window.event：

表示当前的事件对象，IE有这个对象，FF没有，FF通过给事件处理函数传递事件对象

(2) 获取事件源

IE用srcElement获取事件源，而FF用target获取事件源

(3) 添加，去除事件

IE：element.attachEvent(“onclick”, function) element.detachEvent(“onclick”, function)

FF：element.addEventListener(“click”, function, true) element.removeEventListener(“click”, function, true)

(4) 获取标签的自定义属性

IE：div1.value或div1[“value”]

FF：可用div1.getAttribute(“value”)

(5) document.getElementByName()和document.all[name]

IE；document.getElementByName()和document.all[name]均不能获取div元素

FF：可以

(6) input.type的属性

IE：input.type只读

FF：input.type可读写

(7) innerText textContent outerHTML

IE：支持innerText, outerHTML

FF：支持textContent

(8) 是否可用id代替HTML元素

IE：可以用id来代替HTML元素

FF：不可以

1. 构造函数有何优缺点？

答：构造函数可以创建多个对象，

缺点：容易跟普通函数一样调用

1. 用自己的话描述一下“闭包“

答：函数内部又定义了一个函数，内部函数引用外部函数的变量，就构成了闭包

1. 用自己的话描述一下对“异步“和”同步“的理解

同步：按顺序执行，同一时刻只能执行一个事件

异步：不按顺序执行，同一时刻可以执行多个任务

1. 写倒计时 距离2018年还有 多少天多少小时多少分钟多少秒

var time = document.getElementById("time");

function clock() {

//设置目标日期

var targetDate = new Date(2018,0,1);

var currentDate = new Date();

var remainTime = targetDate - currentDate;

//天数

remainDay = parseInt(remainTime/1000/60/60/24);

//小时数

remainHours = parseInt(remainTime/1000/60/60%24);

//分钟

remainMinutes = parseInt(remainTime/1000/60%60);

//秒

remainSeconds = parseInt(remainTime/1000%60);

time.innerHTML = remainDay + "天"+remainHours+"小时"+remainMinutes+"分钟"+remainSeconds+"秒";

}

setInterval(clock,1000);

1. alert怎样换行

答：转义字符,换行符\n

alert("大家好\n大家好")

1. 左侧内容可上下滚动，右侧是滚动条，如何实现布局

答：固定高度，overflow-y:scroll

1. 向上滚动动画如何实现

document.body.scrollTop = "xxx";

document.documentElement.scrollTop = "xxx";

1. form表单提交

答：默认提交，form.onsubmit = function(){}可在表单数据提交之前验证

1. javaScript与ECMAScript的区别？

答：javascript是具体实现，ECMAScript是标准，大纲

1. ==与===的区别？

==等于，判断值是否相等，会有类型转换

===全等 ，判断类型跟值是否相等，类型不会转换

1. 什么是DOM？

答：文档对象模型

1. Undefined与null的区别？

答：Undefined无定义，null空

1. 有两个对象，var o = {num:1};var p = {num:2};要求书写一段代码，使得o+p===3为true。

Object.prototype.toString = function(){

return this.num;

}

1. 如果给你这么一个页面：顶部有logo什么的，下来是导航条，中上部有图片切换，右边是登录注册，左边有简单的信息展示，中下部是一些新闻，底部是一般的那些联系我们呀什么的传统页尾。这样一个布局加上所有的js效果和新闻链接包括所有的兼容，你大概能多久完成（新鸿儒）；
2. 讲解一下this （作用域）

this的指向：普通函数内的this指向全局变量

构造函数内部this指向新创建出来的对象

对象方法内的this指向的是调用该方法的对象

call,apply,bind可以改变this的指向

1. 构造函数

答：new关键字后紧接的函数即为构造函数，约定首字母大写

1. 原型，原型链的理解

答：ECMA5之前没有类的概念，js的继承是基于原型链的

1. 说说前端优化从哪方面着手，具体说明有哪些？

html 语义化，符合标准，减少嵌套层数，加快浏览器DOM解析速度

css 合并，压缩 少写冗余css代码，使用精灵图 ...

js 减少DOM操作，使用变量缓存数据，少用全局变量，做动画的元素都绝对定位...

1. 代码说明call和apply方法的区别 (咱们这方面讲解的少，这样的题有变式，需要举例讲解一下)

答：这两个都是函数的方法，可以改变this的指向，fn.call(obj,param1,param2,…)

call传入单个参数

fn.apply(obj,[param1,param2,...])

apply传入一个数组

1. function f(x){

alert(x);(1)

arguments[0]=null;

alert(x);(2)

}

f(5);

\_\_\_(2)\_\_\_

答案：5，null

1. var a=new Array(3); （实际上是闭包的用法）

a[0]=function(x){return x\*x;}

a[1]=20;

a[2]=a[0](a[1]);

a[2]的结果\_\_\_400\_\_\_

1. var fn = function(i){

return function(){

return i++;

}

}(10);

var a = fn();

var b = fn();

alert(a);(1)

alert(b);(2)

(1)\_10\_\_(2)\_\_11\_

1. 对于js语言，下列说法中（ C ）是正确的？

A.一种标记语言规范的版本名 B.前端编程中，新增特性的一种泛指和统称

C.一种脚本语言 D.可以实现跨平台开发的技术

1. 用两种方法，将第上题中的第一个p标签里的内容的颜色设置为红色

element.style.color = “red”

element.className = “red” .red{color:red}

1. 如何测算数组的长度？

答：array.length

1. 滚动类效果中，会涉及到的定位属性有哪些，同时写出每个命令的含义。

答：left,top,right,bottom,scrollLeft,scrollTop

1. 在放大镜效果中，获取鼠标位置的方法，同时说明鼠标位置是针对谁获取的？

答：e.clientX,e.clientY 相对于窗口坐标

1. 事件种类有哪些？分别举出不少于4个的例子？

答：鼠标事件，键盘事件，触屏事件

1. 访问到tween算法中的Bounce的easeInOut方法？

var startPos = 0;//确定起点位置

var endPos = 500;//确定终点位置

var everyPos = 20;//确定每步移动距离

var step = 0;//确定当前步数

var stepMax = 25;//确定总步数

var tween = {

Bounce:{

easeIn:xxxxx;

easeOut:xxxxx;

easeInOut:xxxx;

}

}

tween.Bounce.easeInOut(step,startPos,(endPos-startPos),stepMax)

1. Date对象中getDay()和getDate()分别是什么含义？取值范围是多少？

getDay()获取星期几 0~6

getDate() 获取日期 1~31

1. 设置当前时间为2004年12月12日的代码书写方法？（分写合写两种方式）

new Date(2004,11,12)

var d = new Date(); d.setFullYear(2004),d.setMonth(11);d.setDate(12);

1. 写出不少于5种Math对象的方法，分别说明含义以及语法

Math.random()随机数0~1之间

Math.abs()绝对值

Math.floor() 向下取整

Math.ceil()向上取整

Math.min();Math.max().....

1. 使用任意一种方法随机生成一个从100-300的整数。

function rnd(min, max) {return parseInt(Math.random() \* (max - min)) + min} //不包含300

rnd(100,300)

1. tab切换效果 代码填空

<style type="text/css">

.show{display:block;}

.select{background:#fcc;}

div {display:none}

</style>

<body>

<h2>

<span id='span1' class='select'>标题1</span>

<span id='span2'>标题2</span>

<span id='span3'>标题3</span>

</h2>

<div id='con1' class='show'>内容1</div>

<div id='con2'>内容2</div>

<div id='con3'>内容3</div>

</body>

<script type="text/javascript">

var cons = document.getElementsByTagName('div');

var spans = document.getElementsByTagName('span')

for (var i=0; i< spans.length ; i++) {

spans[i] .onclick=function () {

for (var j=0; j< cons.length ; j++) {

if ( this == spans[j] ) {

cons[j]. className = “show” ;

spans[j]. className = “select” ;

}

else{

cons[j]. className = “” ;

spans[j]. className = “” ;

}

}

}

}

</script>

1. 用JS截取标签中的abc

<body>

<p id='con'>abcdef</p>

</body>

<script type="text/javascript">

var con = document.getElementById('con');

var string = con. innerText ;

var result = string. substring(0,3) ;

alert(result);

</script>

1. 用js如何获取一个上传文件的扩展名

<input type="file" id="file";

var file = document.getElementById(“file”)

alert(file.files[0].name)

1. 用js实现类、继承（继承构造函数内部的属性方法及原型的属性和方法）、私有变量

function Car() {

//构造函数

this.name = "benz"; //属性

var price = "1000000"; //私有变量，外部不能访问，只能通过下面的getPrice方法调用

this.getPrice = function() {

return price;

}

}

Car.prototype.drive = function() {console.log("drive")}//给原型添加drive方法，创建出来的实例都有该方法

var c = new Car();//创建实例对象

1. 写一个trim函数，兼容IE firefox chrome（正则）（怕部分学生想不到正则）

if(typeof String.prototype.trim !== "function") {

String.prototype.trim = function() {

return this.replace(/^\s\* | \s\*$/g,"");

}

}

直接调用即可，例：var str = “ sdf ”;str.trim();

如何提升网页加载速度

前端角度：合并压缩css,js减少http请求数以及大小，使用精灵图，减少dom操作...

1. 列举事件绑定的方法（原生+JQ addEventlistener与attachEvent的区别）addEventlistener现代浏览器都支持，ie从ie9开始支持，可以绑定多个事件

attachEvent ie专有，可以绑定多个事件，但this指向全局对象

setInterval 与 setTimeout 的区别 (略)

split() join() 的区别

split()字符串方法，将字符串转换成数组

join()数组方法，将数组合并成字符串

1. addEventlistener和通过on这种方式绑定的事件（例：ele.onclick = funciton(){}）有什么区别

addEventlistener可以绑定多个事件

on这种方式只能绑定一次，之后绑定的事件会覆盖前面绑定的，实质是对象属性重新赋值

DOM事件流

事件捕获，事件目标，事件冒泡

1. ajax请求时，如何解析json数据

三种方式 1.JSON.parse(string)

new Function(“return ” + string)()

eval(“(”+string+”)”)

1. 写一个获取非行间样式的函数

//获取非行间样式，兼容现代浏览器跟低版本ie

function getStyle(element,property) {

if(window.getComputedStyle) {

return window.getComputedStyle(element,null)[property];

}else {

return element.currentStyle[property];

}

}

1. 闭包是什么，有什么特性，对页面有什么影响

答：闭包简单理解就是：函数内部有嵌套定义里一个函数，内部函数调用外部函数的局部变量的时候就构成闭包。能保存局部变量，函数执行完不销毁。消耗内存资源

1. 解释jsonp的原理，以及为什么不是真正的ajax （jsonp有些同学可能不知道）

原理：通过script标签的跨域功能，而非XMLHttpRequest

window load 和document ready的区别

document ready 实质是DOMContentLoaded事件

所有的节点渲染完毕，就执行

load事件要等到所有的资源加载完毕，如图片，视频，js脚本等

1. 如何使用原生 Javascript 代码深度克隆一个对象（注意区分对象类型）//对象的深复制

function deepCode(oldObj,newObj) {

var newObj = newObj || {};//如果存在newObj 就newObj,否则用空对象

for(var i in oldObj) {

if(oldObj.hasOwnProperty(i)) {

if(typeof oldObj[i] === "object") {//判断属性是否引用类型的

var c = (oldObj[i].constructor === Array)?[]:{};//如果判断值是否是数组

newObj[i] = deepCode(oldObj[i],c);

//递归

}else {

newObj[i] = oldObj[i];

}

}

}

return newObj;

}

1. 截取字符串abcdefg的efg

答：alert(‘abcdefg’.substring(4));

1. 判断一个字符串中出现次数最多的字符，统计这个次数

var str = 'asdfssaaasasasasaa';  
var json = {};  
  
for (var i = 0; i < str.length; i++) {  
        if(!json[str.charAt(i)]){  
                json[str.charAt(i)] = 1;  
        }else{  
                json[str.charAt(i)]++;  
        }  
};  
var iMax = 0;  
var iIndex = '';  
for(var i in json){  
        if(json[i]>iMax){  
                iMax = json[i];  
                iIndex = i;  
        }  
}  
alert(‘出现次数最多的是:'+iIndex+'出现'+iMax+'次');

参考答案：a出现9次

1. IE与FF脚本兼容性问题

obj.addEventListener(sEv, fn, false);

obj.attachEvent(‘on’+sEv,fn);

obj.getCurrentStyle[attr]

getComputedStyle(obj,false)[attr]

XMLHttpRequest

ActiveXObject(‘Mircorsoft.XMLHttp')

FF本地能设置读取cookie 其他不行

event  ev

事件源

srcElement||target

toElement||relatedTarget

obj.setCapture();只有ie认

obj.releaseCapture();

1. 规避javascript多人开发函数重名问题

命名空间

封闭空间

js模块化mvc（数据层、表现层、控制层）

seajs（如果了解的呢，可以说）

变量转换成对象的属性

对象化

1. javascript面向对象中继承实现

function Person(name){

        this.name = name;

}

Person.prototype.showName = function(){

        alert(this.name);

}

function Worker(name, job){

        Person.apply(this,arguments)

        this.job = job;

}

for(var i in Person.prototype){

        Worker.prototype = Person.prototype;

}

new Worker('easy', ‘coders').showName();

1. outerHTML和innerHTML区别

使用innerHTML的时候将找着元素的内容（不包含元素本身）

使用outerHTML的时候将找着元素的内容（包含元素本身）

注意:

W3C 只支持innerHTML. 其他都是微软的规定.(outerHTML,outerText,innerText只有微软的IE 好使, 其他浏览器不好用(firefox,

mozilla等),必须用其他方法实现)

innerHTML & outerHTML：获取/设置的内容是包含标签的

两者已经被HTML5纳入规范。

innerText & outerText：获取/设置的内容是文本内容

innerHTML在获取的时候：

IE和Opera会将获取到的标签全部转换为大写形式。而苹果、谷歌、火狐会将内容按照原来的格式返回HTML，包括空格和缩进。（这也就意味着返回后的不同内容的DOM树结构截然不同）

outerHTML：新建的DOM子树会替代掉“调用元素”

IE4+、Safari4+、Chrome、Opera8+支持，FF7-不支持

1. 编写一个方法 求一个字符串的字节长度（假设一个中文占两个字节）

var str = ‘22两是’;

alert(getStrlen(str))

function getStrlen(str){

        var json = {len:0};

        var re = /[\u4e00-\u9fa5]/;

        for (var i = 0; i < str.length; i++) {

                if(re.test(str.charAt(i))){

                        json[‘len']++;

                }

        };

        return json[‘len']+str.length;

}

1. 写出3个使用this的典型应用

答：事件： 如onclick  this->发生事件的对象

构造函数          this->new 出来的object

call/apply        改变this

1. 如何深度克隆

var arr = [1,2,43];

var json = {a:6,b:4,c:[1,2,3]};

var str = ‘sdfsdf';

var json2 = clone(json);

alert(json[‘c'])

function clone(obj){

        var oNew = new obj.constructor(obj.valueOf());

        if(obj.constructor == Object){

                for(var i in obj){

                        oNew[i] = obj[i];

                        if(typeof(oNew[i]) == ‘object'){

                                clone(oNew[i]);

                        }

                }

        }

        return oNew;

}

1. JavaScript中如何检测一个变量是一个String类型？请写出函数实现

typeof(obj) == ‘string'

obj.constructor == String;

1. 闭包

答：子函数能被外部调用到，则该作用连上的所有变量都会被保存下来。

1. 请解释什么是Javascript的模块模式，并举出实用实例。

js模块化mvc（数据层、表现层、控制层）

seajs

1. 你如何组织自己的代码？是使用模块模式，还是使用经典继承的方法

对内：模块模式

对外：继承

1. 你如何优化自己的代码？

代码重用

避免全局变量（命名空间，封闭空间，模块化mvc..）

拆分函数避免函数过于臃肿

加注释

1. 你能解释一下JavaScript中的继承是如何工作的吗？

子构造函数中执行父构造函数，并用call\apply改变this

克隆父构造函数原型上的方法

1. 请说明navigator 常用属性？

navigator.userAgent 检查用户代理（设备信息、平台）

mac下的FF浏览器返回的信息：

"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:47.0) Gecko/20100101 Firefox/47.0”

检查cooke是否可用：

navigator.cookieEnabled 返回true为可用，false不可用

更多信息使用浏览器调试工具查看，进行总结

1. 浏览器调试快捷键是什么？

F12 或者 Ctrl+Shift+J两种方法。

ctrl+shift+m 调用FF浏览器的响应视图

ctrl+shift+i 调用chrome浏览器的调试工具

1. 数组怎么去重？

先将原数组进行排序；检查原数组中的第i个元素 与 结果数组中的最后一个元素是否相同，因为已经排序，所以重复元素会在相邻位置；如果不相同，则将该元素存入结果数组中。

利用对象的原型方法，代码为：

Array.prototype.unique2 = function(){

this.sort(); //先排序

var res = [this[0]];

for(var i = 1; i < this.length; i++){

if(this[i] !== res[res.length - 1]){

res.push(this[i]);

}

}

return res;

}

var arr = [1, 'a', 'a', 'b', 'd', 'e', 'e', 1, 0]

alert(arr.unique2());

//结果：0,1,a,b,d,e

1. 数组排序。

数组排序方法还有冒泡排序：

注意：主要说清楚冒泡排序的执行顺序以及原理，编写代码其实是次要的

var arr = [3,7,9,1,5,10];

var temp = 0;

for (var i = 0; i < arr.length; i++){

for (var j = 0; j < arr.length - i; j++){

if (arr[j] > arr[j + 1]){

temp = arr[j + 1];

arr[j + 1] = arr[j];

arr[j] = temp;

}

}

}

alert(arr);

1. 常用变量声明规则？

匈牙利命名：开头字母用变量类型的缩写，其余部分用变量的英文或英文的缩写，要求单词第一个字母大写

小驼峰式：(little camel-case)：第一个单词首字母小写，后面其他单词首字母大写

大驼峰式：(big camel-case)：每个单词的第一个字母都大写;

1. call和apply的区别（动手验证）

call和apply，这两个方法基本上是一个意思，区别在于 call 的第二个参数可以是任意类型，而apply的第二个参数必须是数组，也可以是arguments。

1. 什么是闭包，在js中的作用是什么？

答：闭包就是讲函数内部生成的变量保存到内存中，进行下次调用；也可以说函数外不可以调用函数内部的变量；

当函数内部返回一个函数时，闭包搭建了方法内部与方法外部的桥梁，使得外部也可以任意的获取到方法内部的资源，但是闭包会造成变量在内存中持久占用，因此会有一定的性能问题，不要轻易使用，即便使用，也要在恰当的时机进行释放。

1. 请描述一下cookies、sessionStorage与localStorage的区别。(百度搜索)

答：共同点：都是保存在浏览器端，且同源的。

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

注意：（这里面考虑到安全因素，必要突出安全性方面的信息，比如：敏感信息如用户信息—银行账户、其它登录账户或密码）

1. JavaScript的数据类型都有什么？

基本数据类型：

字符串 String 数字 Number 布尔Boolean

复合数据类型：

数组 Array 对象 Object

特殊数据类型：

Null 空对象 Undefined 未定义

1. 本地存储和事务存储之间的区别是什么？

答：本地存储数据持续永久，但是会话在浏览器打开时有效知道浏览器关闭时会话变量重置。

1. 什么是事务存储？我们如何创建一个事务存储？

会话存储和本地存储类似，但是数据在会话中有效，简而言之数据在你关闭浏览器的时候就被删除了。

为了创建一个会话存储你需要使用“sessionStorage.variablename.”在以下的代码我们创建了一个名为”clickcount”的变量；

如果你刷新浏览器则数目增加，但是如果你关闭浏览器，“clickcount”变量又会从0开始。

实例：

if(sessionStorage.clickcount){

sessionStorage.clickcount=Number(sessionStorage.clickcount)+1;

}else{

sessionStorage.clickcount = 0;

}

1. 本地存储的生命周期是什么？

答：本地存储没有生命周期，它将保留知道用户从浏览器清除或者使用Javascript代码移除。

1. HTML5中的本地存储概念是什么？

本地存储是把网页上的某些数据按照键值对的方式存放在浏览器里面 一个域最多可以存5M数据 不区分离线还是在线 这些被存的数据都会一直在浏览器里面 除非手动清除数据 。

（小心有陷阱）面试官会问做过项目里面哪些地方用过？

1. 我们如何从本地存储中添加和移除数据？

数据添加到本地存储采用键值对，以下示例显示了城市数据”北京”添加了键”test”

localStorage.setItem("test","北京");

从本地存储中检索数据我们可以提供键名并使用”getItem”方法

localStorage.getItem(“test");

如果你想存储Json格式，你可以使用“JSON.stringify”方法，如下所示：

通过浏览器可以查看

var country = {};

country.name = "中国";

country.code = "I001";

//localStorage.setItem("I001", country);

//var country1 = localStorage.getItem("I001");

localStorage.setItem("I001",JSON.stringify(country));

localStorage.getItem(“I001");

移除本地存储：localStorage.removeItem(指定键)

localStorage.removeItem(“I001");

1. 如何实现浏览器内多个标签页之间的通信? (阿里)

调用localstorge、cookies等本地存储方式

1. 介绍JavaScript的基本数据类型。

基本数据类型5种：字符串（string）、数字（number）、布尔（Boolean）、空（Null）、未定义（Undefined）

复杂数据类型：数组、对象（跟本题没有多大关系，如果面试官换一种问法，就需要注意了，问JavaScript数据类型都有哪些呢？）

1. 说几条写JavaScript的基本规范？

（1）不要在同一行声明多个变量

（2）如果你不知道数组的长度，使用 push

（3）请使用 ===/!== 来比较 true/false 或者数值

（4）对字符串使用单引号 ''(因为大多时候我们的字符串。特别html会出现")

（5）使用对象字面量替代 new Array 这种形式

（6）绝对不要在一个非函数块里声明一个函数，把那个函数赋给一个变量。浏览器允许你这么做，但是它们解析不同

（7）不要使用全局函数

（8）总是使用 var 来声明变量，如果不这么做将导致产生全局变量，我们要避免污染全局命名空间

（9）Switch 语句必须带有 default 分支

（10）使用 /\*\*...\*/ 进行多行注释，包括描述，指定类型以及参数值和返回值

（11）函数不应该有时候有返回值，有时候没有返回值

（12）语句结束一定要加分号

（13）for 循环必须使用大括号

（14）if 语句必须使用大括号

（15）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

（16）避免单个字符名，让你的变量名有描述意义

（17）当命名对象、函数和实例时使用驼峰命名规则

（18）给对象原型分配方法，而不是用一个新的对象覆盖原型，覆盖原型会使继承出现问题

（19）当给事件附加数据时，传入一个哈希而不是原始值，这可以让后面的贡献者加入更多数据到事件数据里，而不用找出并更新那个事件的事件处理器

参考文档：http://www.jb51.net/article/24579.htm

1. JavaScript有几种类型值？（堆：原始值和 栈：引用值），你能画一下他们的内存图吗？

栈：原始数据类型（Undefined，Null，Boolean，Number，String）

堆：引用数据类型（对象、数组、函数）

两种类型的区别：存储位置不同

//

原始数据类型直接存储在栈(stack)中的简单数据段，占据空间小、大小固定，属于被频繁使用数据，所以放入栈中存储；

引用数据类型存储在堆(heap)中的对象,占据空间大、大小不固定,如果存储在栈中，将会影响程序运行的性能；引用数据类型在栈中存储了指针，该指针指向堆中该实体的起始地址。当解释器寻找引用值时，会首先检索其在栈中的地址，取得地址后从堆中获得实体。

1. Javascript如何实现继承？

(1)构造继承

(2)原型继承

(3)实例继承

(4)拷贝继承

//

原型prototype机制或apply和call方法去实现较简单，建议使用构造函数与原型混合方式。

function Parent() {

this.name = 'song';

}

function Child() {

this.age = 28;

}

Child.prototype = new Parent(); //通过原型,继承了Parent

//

var demo = new Child()l;

alert(demo.age);

alert(demo.name); //得到被继承的属性

1. 如何创建一个对象? （画出此对象的内存图）

javascript创建对象简单的说,无非就是使用内置对象或各种自定义对象，当然还可以用JSON；但写法有很多种，也能混合使用。

//

(1)对象字面量的方式

person={firstname:"Mark",lastname:"Yun",age:25,eyecolor:"black"};

(2)用function来模拟无参的构造函数

function Person(){}

var person = new Person(); //定义一个function，如果使用new"实例化",该function可以看作是一个Class

person.name = "Xiaosong";

person.age = "23";

person.work = function() {

alert("Hello " + person.name);

}

person.work();

(3)用function来模拟参构造函数来实现（用this关键字定义构造的上下文属性）

function Person(name,age,hobby) {

this.name = name; //this作用域：当前对象

this.age = age;

this.work = work;

this.info = function() {

alert("我叫" + this.name + "，今年" + this.age + "岁，是个" + this.work);

}

}

var Xiaosong = new Person("WooKong",23,"程序猿"); //实例化、创建对象

Xiaosong.info(); //调用info()方法

(4)用工厂方式来创建（内置对象）

var jsCreater = new Object();

jsCreater.name = "Brendan Eich"; //JavaScript的发明者

jsCreater.work = "JavaScript";

jsCreater.info = function() {

alert("我是"+this.work+"的发明者"+this.name);

}

jsCreater.info();

(5)用原型方式来创建

function Standard(){}

Standard.prototype.name = "ECMAScript";

Standard.prototype.event = function() {

alert(this.name+"是脚本语言标准规范");

}

var jiaoben = new Standard();

jiaoben.event();

(6)用混合方式来创建

function iPhone(name,event) {

this.name = name;

this.event = event;

}

iPhone.prototype.sell = function() {

alert("我是"+this.name+"，我是iPhone5s的"+this.event+"~ haha!");

}

var SE = new iPhone("iPhone SE","官方翻新机");

SE.sell();

1. 谈谈This对象的理解。

this总是指向函数的直接调用者（而非间接调用者）；

如果有new关键字，this指向new出来的那个对象；

在事件中，this指向触发这个事件的对象，特殊的是，IE中的attachEvent中的this总是指向全局对象Window；

1. eval是做什么的？

它的功能是把对应的字符串解析成JS代码并运行；

应该避免使用eval，因为不安全，非常耗性能（2次，一次解析成js语句，一次执行）。

（注意：在项目里面写js代码的时候，禁止使用的，因为有安全因素）

1. null，undefined的区别？

null 表示一个对象被定义了，值为“空值”；

undefined 表示不存在这个值。

//

typeof undefined

//"undefined"

undefined :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 undefined；

例如变量被声明了，但没有赋值时，就等于undefined。

//

typeof null

//"object"

null : 是一个对象(空对象, 没有任何属性和方法)；

例如作为函数的参数，表示该函数的参数不是对象；

//

注意：

在验证null时，一定要使用　=== ，因为 == 无法分别 null 和　undefined

1. ["1", "2", "3"].map(parseInt) 答案是多少？

[1,NaN,NaN]

因为 parseInt 需要两个参数(val,radix)，其中 radix 表示解析时用的基数。

map 传了3个(element,index,array)，对应的 radix 不合法导致解析失败。

parseInt(转换数字,转换进制2、8、10、16)

一般般使用parseInt(“010”,10) //10 parseInt(“010”,8) //8

1. 事件、IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

(1)我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。

(2)事件处理机制：IE是事件冒泡、Firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件；

(3)ev.stopPropagation();（旧ie的方法 ev.cancelBubble = true;）

1. "use strict";是什么意思 ? 使用它的有什么好处或坏处？

use strict是一种ECMAscript 5 添加的（严格）运行模式,这种模式使得 Javascript 在更严格的条件下运行,使JS编码更加规范化的模式,消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为。

默认支持的糟糕特性都会被禁用，比如不能用with，也不能在意外的情况下给全局变量赋值;

全局变量的显示声明,函数必须声明在顶层，不允许在非函数代码块内声明函数,arguments.callee也不允许使用；

消除代码运行的一些不安全之处，保证代码运行的安全,限制函数中的arguments修改，严格模式下的eval函数的行为和非严格模式的也不相同;

提高编译器效率，增加运行速度；

为未来新版本的Javascript标准化做铺垫。

1. new操作符具体干了什么呢?

(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 。

//

var obj = {};

obj.\_\_proto\_\_ = Base.prototype;

Base.call(obj);

1. Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？

hasOwnProperty

JavaScript 中 hasOwnProperty 函数方法是返回一个布尔值，指出一个对象是否具有指定名称的属性。此方法无法检查该对象的原型链中是否具有该属性；该属性必须是对象本身的一个成员。

使用方法：

object.hasOwnProperty(proName)

其中参数object是必选项，一个对象的实例。

proName是必选项，一个属性名称的字符串值。

如果 object 具有指定名称的属性，那么JavaScript中hasOwnProperty函数方法返回 true，反之则返回 false。

1. 对JSON的了解？

JSON(JavaScript Object Notation)是一种轻量级的数据交换格式。

它是基于JavaScript的一个子集。数据格式简单，易于读写，占用带宽小。

如：

{"age":"12", "name":"back"}

(注意：强调一下之前做的项目里面通过ajax请求，后端返回数据常用的格式)

1. 解释一下Math常用哪些方法？

//x为传入参数

Math.abs(x); //返回数的绝对值。

Math.ceil(x); //对数进行上舍入或向上取整

Math.floor(x); //对数进行下舍入或向下取整

Math.max(x,y) //返回 x 和 y 中的最大值。

Math.min(x,y) //返回 x 和 y 中的最小值。

Math.random() // 返回 0 ~ 1 之间的随机数。

Math.round(x) //把数四舍五入为最接近的整数。

Math.PI //返回圆周率（约等于3.14159）。

1. js延迟加载的方式有哪些？

Defer(只支持 IE)和async、动态创建DOM方式（用得最多）、按需异步载入js

1. 知道ECMAScript 6里怎么写class么?

ES6的class可以看作只是一个语法糖，它的绝大部分功能，ES5都可以做到，新的class写法只是让对象原型的写法更加清晰、更像面向对象编程的语法而已。

//定义类

class Point {

constructor(x,y) { //构造方法

this.x = x; //this关键字代表实例对象

this.y = y;

}

toString() {

return '(' + this.x + ',' + this.y + ')';

}

}

1. AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

AMD 规范在这里：[https://github.com/amdjs/amdjs-api/wiki/AMD](https://github.com/amdjs/amdjs-api/wiki/AMD%22%20%5Ct%20%22_blank)

CMD 规范在这里：[https://github.com/seajs/seajs/issues/242](https://github.com/seajs/seajs/issues/242%22%20%5Ct%20%22_blank)

1. call() 和 .apply() 的区别？

一、方法的定义

call方法:

语法：call(thisObj，Object)

定义：调用一个对象的一个方法，以另一个对象替换当前对象。

说明：

call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。

如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。

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。

1. JavaScript中的作用域与变量声明提升？

参考地址：http://blog.csdn.net/sunxing007/article/details/9034253

http://www.cnblogs.com/silentjesse/p/4024536.html

1. 如何编写高性能的Javascript？

参考地址：<http://developer.51cto.com/art/200906/131335.htm>

1. 那些操作会造成内存泄漏？

内存泄漏是指任何对象在您不再拥有或需要它之后任然存在。

垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量，如果一个对象的引用数量为0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。

//

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）

1. 知道什么是webkit么? 知道怎么用浏览器的各种工具来调试和debug代码么?

Chrome、safari浏览器的内核。

其中chrome的模板引擎使用的是Gecko

Gecko在互联网中是最流行的[排版引擎](http://baike.baidu.com/view/459307.htm%22%20%5Ct%20%22_blank)之一，其流行程度仅次于[Trident](http://baike.baidu.com/view/682014.htm%22%20%5Ct%20%22_blank)（用于Windows版的[Internet Explorer](http://baike.baidu.com/view/85144.htm%22%20%5Ct%20%22_blank)4.0版本开始），其后尚有[WebKit](http://baike.baidu.com/view/1510583.htm%22%20%5Ct%20%22_blank)（用于[Safari](http://baike.baidu.com/view/110484.htm%22%20%5Ct%20%22_blank)以及[Google Chrome](http://baike.baidu.com/view/1835854.htm%22%20%5Ct%20%22_blank)）以及[Presto](http://baike.baidu.com/view/682394.htm%22%20%5Ct%20%22_blank)（用于Opera）。

可以通过js检查当前所使用浏览器的内核以及模版引擎

alert(navigator.userAgent);

1. 如何测试前端代码么? 知道BDD, TDD, Unit Test么? 知道怎么测试你的前端工程么(mocha, sinon, jasmin, qUnit..)?

如何测试前端代码么?

PC端浏览器：市场份额：http://tongji.baidu.com/data/browser

1. 编写完前端静态html+css(注意不包括js)

2. 加上js代码以及逻辑处理（再进行测试）

在各个浏览器里面做兼容测试：1）.ietest—ie6-ie11（ie模拟浏览器）、真正的windows版本自带的IE浏览器F12可以看见下图,可以切换不同的浏览器模式，后面跟着把文档模式也改成浏览器模式一样

2).FF(火狐浏览器) 、chrome（谷歌浏览器）、opera（欧朋浏览器）、maxthon（遨游浏览器）、safari(苹果的浏览器)、猎豹安全浏览器

比较流氓的浏览器：（不知不觉电脑上就给安装上了）

360浏览器（可以切换浏览器模式）、搜狗高速浏览器、腾讯QQ浏览器、百度浏览器

还有其它用户量不大的浏览器就不用一一列举了。

http://liulanqixiazai2016.lofter.com/

移动端浏览器：移动份额

uc浏览器（知道为啥要第一个写吗？在移动端用户量最多）、腾讯qq浏览器、百度浏览器、360浏览器、猎豹浏览器、safari 、chrome、opera、maxthon、ff、搜狗浏览器、海豚浏览器

2016年网页浏览器发展现状分析

<http://www.chinabgao.com/k/liulanqi/22495.html>

移动设备品牌占比

http://tongji.baidu.com/data/mobile/brand

知道BDD, TDD, Unit Test么?

TDD：测试驱动开发（Test-Driven Development）

BDD：行为驱动开发（Behavior Driven Development）

ATDD：验收测试驱动开发（Acceptance Test Driven Development）

参考地址：<http://www.testwo.com/article/414>

知道怎么测试你的前端工程么(mocha, sinon, jasmin, qUnit..)?

参考地址：

<http://www.tuicool.com/articles/yuMvQz>

sinon 参考地址：<http://www.maobashi.com/1366.html>

参考地址：http://www.w3cmark.com/2016/446.html

1. 前端templating(Mustache, underscore, handlebars)是干嘛的, 怎么用?

前端js模板引擎

Mustache：

参考地址：http://www.open-open.com/lib/view/open1416792564461.html

http://www.cnblogs.com/iyangyuan/archive/2013/12/12/3471227.html

underscore

[Underscore](http://github.com/jashkenas/underscore/) 是一个 JavaScript 工具库，它提供了一整套函数式编程的实用功能，但是没有扩展任何 JavaScript 内置对象。 他解决了这个问题：“如果我面对一个空白的 HTML 页面，并希望立即开始工作，我需要什么？” 他弥补了 [jQuery](http://jquery.com) 没有实现的功能，同时又是 [Backbone](http://backbonejs.org) 必不可少的部分。

参考地址：http://www.bootcss.com/p/underscore/

简单做个了解就可以（真实用在仔细学习）

1. 简述一下 Handlebars 的基本用法？

参考地址：http://baike.baidu.com/link?url=OF4LwdRf8jbdYZt8w1Yig7Q7dSCRA2jvpfxRISOE91l6kQzl1wZwmHnJb1CEJZ4Ov7IE5wHVPvQbBam8B-O0E\_

<http://www.cnblogs.com/iyangyuan/archive/2013/12/12/3471227.html>

1. ie各版本和chrome可以并行下载多少个资源

IE6 两个并发，iE7升级之后的6个并发，之后版本也是6个

Firefox，chrome也是6个

1. 谈一谈你对 ECMAScript6 的了解？

ECMAScript 6 是JavaScript语言的下一代标准，已经在2015年6月正式发布了。它的目标，是使得JavaScript语言可以用来编写复杂的大型应用程序，成为企业级开发语言。

标准的制定者有计划，以后每年发布一次标准，使用年份作为标准的版本。因为当前版本的ES6是在2015年发布的，所以又称ECMAScript 2015。也就是说，ES6就是ES2015

1. 说说最近最流行的一些东西吧？常去哪些网站？

Node.js、Mongodb、npm、MVVM、MEAN、three.js,React 。

网站：w3cfuns,sf,hacknews,CSDN,慕课，博客园，InfoQ,w3cplus等

1. 前端安全问题？

sql注入原理

就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。

总的来说有以下几点：

1.永远不要信任用户的输入，要对用户的输入进行校验，可以通过正则表达式，或限制长度，对单引号和双"-"进行转换等。

2.永远不要使用动态拼装SQL，可以使用参数化的SQL或者直接使用存储过程进行数据查询存取。

3.永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。

4.不要把机密信息明文存放，请加密或者hash掉密码和敏感的信息。

XSS原理及防范

Xss(cross-site scripting)攻击指的是攻击者往Web页面里插入恶意html标签或者javascript代码。比如：攻击者在论坛中放一个 看似安全的链接，骗取用户点击后，窃取cookie中的用户私密信息；或者攻击者在论坛中加一个恶意表单， 当用户提交表单的时候，却把信息传送到攻击者的服务器中，而不是用户原本以为的信任站点。

XSS防范方法

1.代码里对用户输入的地方和变量都需要仔细检查长度和对”<”,”>”,”;”,”’”等字符做过滤；其次任何内容写到页面之前都必须加以encode，避免不小心把html tag 弄出来。这一个层面做好，至少可以堵住超过一半的XSS 攻击。

2.避免直接在cookie 中泄露用户隐私，例如email、密码等等。 3.通过使cookie 和系统ip 绑定来降低cookie 泄露后的危险。这样攻击者得到的cookie 没有实际价值，不可能拿来重放。

4.尽量采用POST 而非GET 提交表单

XSS与CSRF有什么区别吗？

XSS是获取信息，不需要提前知道其他用户页面的代码和数据包。CSRF是代替用户完成指定的动作，需要知道其他用户页面的代码和数据包。

要完成一次CSRF攻击，受害者必须依次完成两个步骤：

　　1.登录受信任网站A，并在本地生成Cookie。 　　2.在不登出A的情况下，访问危险网站B。

CSRF的防御

1. 服务端的CSRF方式方法很多样，但总的思想都是一致的，就是在客户端页面增加伪随机数。

2. 使用验证码

1. 如何判断当前脚本运行在浏览器还是node环境中？（阿里）

通过判断Global对象是否为window，如果不为window，当前脚本没有运行在浏览器中

1. 把 Script 标签 放在页面的最底部的body封闭之前 和封闭之后有什么区别？浏览器会如何解析它们？

参考资料：

https://segmentfault.com/a/1190000004292479?\_ea=560235

http://www.zhihu.com/question/20027966

1. 移动端的点击事件的有延迟，时间是多久，为什么会有？ 怎么解决这个延时？

参考资料：http://www.mamicode.com/info-detail-666685.html

1. Node.js的适用场景？

高并发、聊天、实时消息推送、服务器做前端资源压缩

## 10.JavaScript原型，原型链 ? 有什么特点？

原型对象也是普通的对象，是对象一个自带隐式的 \_\_proto\_\_ 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为null的话，我们就称之为原型链。

原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链。

JavaScript的数据对象有那些属性值？

　 writable：这个属性的值是否可以改。

configurable：这个属性的配置是否可以删除，修改。

enumerable：这个属性是否能在for…in循环中遍历出来或在Object.keys中列举出来。

value：属性值。

当我们需要一个属性的时，Javascript引擎会先看当前对象中是否有这个属性， 如果没有的话，就会查找他的Prototype对象是否有这个属性。

function clone(proto) {

　　 function Dummy() { }

　　 Dummy.prototype = proto;

　　 Dummy.prototype.constructor = Dummy;

　　 return new Dummy(); //等价于Object.create(Person);

}

function object(old) {

   function F() {};

    F.prototype = old;

    return new F();

}

var newObj = object(oldObject);

**写一个通用的事件侦听器函数**

// event(事件)工具集，来源：https://github.com/markyun

markyun.Event = {

// 页面加载完成后

readyEvent : function(fn) {

if (fn==null) {

fn=document;

}

var oldonload = window.onload;

if (typeof window.onload != 'function') {

window.onload = fn;

} else {

window.onload = function() {

oldonload();

fn();

};

}

},

// 视能力分别使用dom0||dom2||IE方式 来绑定事件

// 参数： 操作的元素,事件名称 ,事件处理程序

addEvent : function(element, type, handler) {

if (element.addEventListener) {

//事件类型、需要执行的函数、是否捕捉

element.addEventListener(type, handler, false);

} else if (element.attachEvent) {

element.attachEvent('on' + type, function() {

handler.call(element);

});

} else {

element['on' + type] = handler;

}

},

// 移除事件

removeEvent : function(element, type, handler) {

if (element.removeEnentListener) {

element.removeEnentListener(type, handler, false);

} else if (element.datachEvent) {

element.detachEvent('on' + type, handler);

} else {

element['on' + type] = null;

}

},

// 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)

stopPropagation : function(ev) {

if (ev.stopPropagation) {

ev.stopPropagation();

} else {

ev.cancelBubble = true;

}

},

// 取消事件的默认行为

preventDefault : function(event) {

if (event.preventDefault) {

event.preventDefault();

} else {

event.returnValue = false;

}

},

// 获取事件目标

getTarget : function(event) {

return event.target || event.srcElement;

},

// 获取event对象的引用，取到事件的所有信息，确保随时能使用event；

getEvent : function(e) {

var ev = e || window.event;

if (!ev) {

var c = this.getEvent.caller;

while (c) {

ev = c.arguments[0];

if (ev && Event == ev.constructor) {

break;

}

c = c.caller;

}

}

return ev;

}

};

# JavaScript部分

## 要点： 数据类型、面向对象、继承、闭包、插件、作用域、跨域、原型链、模块化、自定义事件、异步装载回调、模板引擎、Nodejs等。

js的几种数据类型：number,string,boolean,object,undefined

js的常见内置对象类：Date,Array,Math、Number、Boolean、String、Array、RegExp、Function...

通常可以做一些小练习来判断TA的水平，js 虽然很灵活，但是具体的代码和实现方式能体现出一个人的全局观，随着代码规模的增长，复杂度增加，如何合理划分模块实现功能和接口的能力比较重要。（下面例题）

["1", "2", "3"].map(parseInt)

[typeof null, null instanceof Object]

[ [3,2,1].reduce(Math.pow), [].reduce(Math.pow)] ]

var val = 'smtg';

console.log('Value is ' + (val === 'smtg') ? 'Something' : 'Nothing');

## 1.创建一个对象

function Person(name, age) {

this.name = name;

this.age = age;

this.sing = function() { alert(this.name) }

}

## 2.谈谈This对象的理解。

this是js的一个关键字，随着函数使用场合不同，this的值会发生变化。

但是总有一个原则，那就是this指的是调用函数的那个对象。

this一般情况下：是全局对象Global。 作为方法调用，那么this就是指这个对象

## 3.事件、IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。

2. 事件处理机制：IE是事件冒泡、火狐是 事件捕获；

3. ev.stopPropagation();

## 4.什么是闭包（closure），为什么要用？

待完善

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

## 5.如何判断一个对象是否属于某个类？使用instanceof

（待完善）

if(a instanceof Person){

alert('yes');

}

## 6.new操作符具体干了什么呢?

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 。

var obj = {};

obj.\_\_proto\_\_ = Base.prototype;

Base.call(obj);

## 7.JSON 的了解

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小

{'age':'12', 'name':'back'}

## 9.ajax 是什么?ajax 的交互模型?同步和异步的区别?如何解决跨域问题?

待完善

1. 通过异步模式，提升了用户体验

2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用

3. Ajax在客户端运行，承担了一部分本来由服务器承担的工作，减少了大用户量下的服务器负载。

### Ajax的最大的特点是什么。

Ajax可以实现动态不刷新（局部刷新）

readyState属性 状态 有5个可取值： 0=未初始化 ，1=正在加载 2=以加载，3=交互中，4=完成

ajax的缺点

1、ajax不支持浏览器back按钮。

2、安全问题 AJAX暴露了与服务器交互的细节。

3、对搜索引擎的支持比较弱。

4、破坏了程序的异常机制。

5、不容易调试。

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

## 10.模块化怎么做？

[立即执行函数](http://benalman.com/news/2010/11/immediately-invoked-function-expression/),不暴露私有成员

var module1 = (function(){

　　　　var \_count = 0;

　　　　var m1 = function(){

　　　　　　//...

　　　　};

　　　　var m2 = function(){

　　　　　　//...

　　　　};

　　　　return {

　　　　　　m1 : m1,

　　　　　　m2 : m2

　　　　};

　　})();

## 11.对Node的优点和缺点提出了自己的看法：

（优点）因为Node是基于事件驱动和无阻塞的，所以非常适合处理并发请求，

因此构建在Node上的代理服务器相比其他技术实现（如Ruby）的服务器表现要好得多。

此外，与Node代理服务器交互的客户端代码是由javascript语言编写的，因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。

\*（缺点）Node是一个相对新的开源项目，所以不太稳定，它总是一直在变，而且缺少足够多的第三方库支持。看起来，就像是Ruby/Rails当年的样子。

## 12.异步加载的方式

(1) defer，只支持IE

(2) async：

(3) 创建script，插入到DOM中，加载完毕后callBack

documen.write和 innerHTML的区别

document.write只能重绘整个页面

innerHTML可以重绘页面的一部分

## 13.告诉我答案是多少？

(function(x){

delete x;

alert(x);

})(1+5);

函数参数无法delete删除，delete只能删除通过for in访问的属性。

当然，删除失败也不会报错，所以代码运行会弹出“1”。

## 14.JS中的call()和apply()方法的区别？

例子中用 add 来替换 sub，add.call(sub,3,1) == add(3,1) ，所以运行结果为：alert(4);

注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

function add(a,b)

{

alert(a+b);

}

function sub(a,b)

{

alert(a-b);

}

add.call(sub,3,1);

## 1.JS数组去重

以下是展示三种方法：

Array.prototype.unique1 = function () {

  var n = []; //一个新的临时数组

for (var i = 0; i < this.length; i++) //遍历当前数组

  {

    //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(this[i]) == -1) n.push(this[i]);

  }

  return n;

}

NO.2

Array.prototype.unique2 = function(){   var n = {},r=[]; //n为hash表，r为临时数组  for(var i = 0; i < this.length; i++) //遍历当前数组{   if (!n[this[i]]) //如果hash表中没有当前项  { n[this[i]] = true; //存入hash表  r.push(this[i]); //把当前数组的当前项push到临时数组里面  }}return r;}

NO.3

Array.prototype.unique3 = function(){   var n = [this[0]]; //结果数组    for(var i = 1; i < this.length; i++) //从第二项开始遍历   {        //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 (this.indexOf(this[i]) == i) n.push(this[i]);    }    return n;}

## 2.js操作获取和设置cookie

//创建cookie

function setCookie(name, value, expires, path, domain, secure) {

    var cookieText = encodeURIComponent(name) + '=' + encodeURIComponent(value);

    if (expires instanceof Date) {

        cookieText += '; expires=' + expires;

    }

    if (path) {

        cookieText += '; expires=' + expires;

    }

    if (domain) {

        cookieText += '; domain=' + domain;

    }

    if (secure) {

        cookieText += '; secure';

    }

    document.cookie = cookieText;

}

//获取cookie

function getCookie(name) {

    var cookieName = encodeURIComponent(name) + '=';

    var cookieStart = document.cookie.indexOf(cookieName);

    var cookieValue = null;

    if (cookieStart > -1) {

        var cookieEnd = document.cookie.indexOf(';', cookieStart);

        if (cookieEnd == -1) {

            cookieEnd = document.cookie.length;

        }

        cookieValue = decodeURIComponent(document.cookie.substring(cookieStart + cookieName.length, cookieEnd));

    }

    return cookieValue;

}

//删除cookie

function unsetCookie(name) {

    document.cookie = name + "= ; expires=" + new Date(0);

}

## 8.javascript里面的继承怎么实现，如何避免原型链上面的对象共享

用构造函数和原型链的混合模式去实现继承，避免对象共享可以参考经典的extend()函数，很多前端框架都有封装的，就是用一个空函数当做中间变量

## 32.javascript对象的几种创建方式

　　1，工厂模式

　　2，构造函数模式

　　3，原型模式

　　4，混合构造函数和原型模式

　　5，动态原型模式

　　6，寄生构造函数模式

　　7，稳妥构造函数模式

## 33.javascript继承的6种方法

　　1，原型链继承

　　2，借用构造函数继承

　　3，组合继承(原型+借用构造)

　　4，原型式继承

　　5，寄生式继承

　　6，寄生组合式继承

　　详情：JavaScript继承方式详解