# JavaScript面试题集锦

## 1.js基础

## 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，不安全，非常耗性能（2次，一次解析成js语句，一次执行）。

由JSON字符串转换为JSON对象的时候可以用eval，var obj =eval('('+ str +')');

## 2.什么是window对象? 什么是document对象?

window对象是指浏览器打开的窗口。

document对象是Document对象（HTML 文档对象）的一个只读引用，window对象的一个属性。

## 3.null，undefined 的区别？

null 表示一个对象是“没有值”的值，也就是值为“空”；

undefined 表示一个变量声明了没有初始化(赋值)；

undefined不是一个有效的JSON，而null是；

undefined的类型(typeof)是undefined；

null的类型(typeof)是object；

Javascript将未赋值的变量默认值设为undefined；

Javascript从来不会将变量设为null。它是用来让程序员表明某个用var声明的变量时没有值的。

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

null == undefined // true

null === undefined // false

再来一个例子：

null

Q：有张三这个人么？

A：有！

Q：张三有房子么？

A：没有！

undefined

Q：有张三这个人么？

A：有！

Q: 张三有多少岁？

A: 不知道（没有被告诉）

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

parseInt() 函数能解析一个字符串，并返回一个整数，需要两个参数 (val, radix)，

其中 radix 表示要解析的数字的基数。【该值介于 2 ~ 36 之间，并且字符串中的数字不能大于radix才能正确返回数字结果值】;

但此处 map 传了 3 个 (element, index, array),我们重写parseInt函数测试一下是否符合上面的规则。

function parseInt(str, radix) {

return str+'-'+radix;

};

var a=["1", "2", "3"];

a.map(parseInt); // ["1-0", "2-1", "3-2"] 不能大于radix

因为二进制里面，没有数字3,导致出现超范围的radix赋值和不合法的进制解析，才会返回NaN

所以["1", "2", "3"].map(parseInt) 答案也就是：[1, NaN, NaN]

## 5.事件是？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：I.E（IE8以下）.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;）

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

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。

## 7.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。

## 8.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"}

JSON字符串转换为JSON对象:

var obj =eval('('+ str +')');

var obj = str.parseJSON();

var obj = JSON.parse(str);

JSON对象转换为JSON字符串：

var last=obj.toJSONString();

var last=JSON.stringify(obj);

## 9.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

## 10.如何判断当前脚本运行在浏览器还是node环境中？（阿里）

this === window ? 'browser' : 'node';

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

## 11.javascript的typeof返回哪些数据类型

alert(typeof [1, 2]); //object

alert(typeof 'leipeng'); //string

var i = true;

alert(typeof i); //boolean

alert(typeof 1); //number

var a;

alert(typeof a); //undefined

function a(){;};

alert(typeof a) //function

## 12.例举3种强制类型转换和2种隐式类型转换?

强制（parseInt(),parseFloat(),Number()）

隐式（== ,!=）

## 13.split() 、join() 的区别

前者是切割成数组的形式，后者是将数组转换成字符串

## 14.数组方法pop() push() unshift() shift()

1. push()尾部添加 pop()尾部删除
2. unshift()头部添加 shift()头部删除
3. map() : 遍历数组中的元素, 返回一个新数组(包含回调函数返回的数据)
4. filter():遍历数组中的元素, 返回一个新数组(包含回调函数返回true的元素)

## 15.事件绑定和普通事件有什么区别

1) 普通添加事件的方法：

var btn = document.getElementById("hello");

btn.onclick = function(){

alert(1);

}

btn.onclick = function(){

alert(2);

}

执行上面的代码只会alert 2

1. 事件绑定方式添加事件：

var btn = document.getElementById("hello");

btn.addEventListener("click",function(){

alert(1);

},false);

btn.addEventListener("click",function(){

alert(2);

},false);

执行上面的代码会先alert 1 再 alert 2

1. 普通添加事件的方法不支持添加多个事件，最下面的事件会覆盖上面的，而事件绑定 （addEventListener）方式添加事件可以添加多个。
2. addEventListener不兼容低版本IE（IE8及以下）
3. 普通事件无法取消
4. addEventLisntener还支持事件冒泡+事件捕获
5. onclick不可以多个 下面会覆盖上面 不支持捕获阶段获取

## IE和DOM事件流的区别

1.执行顺序不一样、

2.参数不一样

3.事件加不加on

4.this指向问题

## 17.IE和标准下有哪些兼容性的写法

var ev = ev || window.event

document.documentElement.clientWidth || document.body.clientWidth

var target = ev.srcElement||ev.target

## 18.如何阻止事件冒泡和事件默认行为

//阻止事件冒泡

if(typeof ev.stopPropagation=='function') { //标准的

ev.stopPropagation();

} else { //非标准IE

window.event.cancelBubble = true;

}

//阻止事件默认行为

return false

## window.onload 和document ready的区别

window.onload 是在dom文档树加载完和所有文件加载完之后执行一个函数 document.ready原生中没有这个方法，jquery中有 $().ready(function),在dom文档树加 载完之后执行一个函数（注意，这里面的文档树加载完不代表全部文件加载完）。

$(document).ready要比window.onload先执行

window.onload只能出来一次，$(document).ready可以出现多次

## 20.”==”和“===”的不同

前者会自动转换类型

后者不会

## 21.JavaScript是一门什么样的语言，它有哪些特点？

javaScript一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML网页上使用，用来给HTML网页增加动态功能。JavaScript兼容于ECMA标准，因此也称为ECMAScript。

基本特点

1. 是一种解释性脚本语言（代码不进行预编译）。

2. 主要用来向HTML（标准通用标记语言下的一个应用）页面添加交互行为。

3. 可以直接嵌入HTML页面，但写成单独的js文件有利于结构和行为的分离。

跨平台特性，在绝大多数浏览器的支持下，可以在多种平台下运行（如Windows、Linux、Mac、Android、iOS等）。

## 22.JavaScript的数据类型都有什么？

基本数据类型：String,boolean,Number,Undefined, Null

引用数据类型：Object, Array, Function

那么问题来了，如何判断某变量是否为数组数据类型？

方法一.判断其是否具有“数组性质”，如slice()方法。可自己给该变量定义slice方法， 故有时会失效

方法二.obj instanceof Array 在某些IE版本中不正确

方法三.方法一二皆有漏洞，在ECMA Script5中定义了新方法Array.isArray(), 保证其兼容 性，最好的方法如下：

if(typeof Array.isArray==="undefined"){

Array.isArray = function(arg){

return Object.prototype.toString.call(arg)==="[object Array]"

};

}

## 23.当一个DOM节点被点击时候，我们希望能够执行一个函数，应该怎么做？

直接在DOM里绑定事件：<div onclick=”test()”></div>

在JS里通过onclick绑定：xxx.onclick = test

通过事件添加进行绑定：addEventListener(xxx, ‘click’, test)

那么问题来了，Javascript的事件流模型都有什么？

“事件冒泡”：事件开始由最具体的元素接受，然后逐级向上传播

“事件捕捉”：事件由最不具体的节点先接收，然后逐级向下，一直到最具体的

“DOM事件流”：三个阶段：事件捕捉，目标阶段，事件冒泡

## 24.看下列代码输出为何？解释原因。

var a;

alert(typeof a); // undefined

alert(b); // 报错

解释：Undefined是一个只有一个值的数据类型，这个值就是“undefined”，在使用var 声明变量但并未对其赋值进行初始化时，这个变量的值就是undefined。而b由于未声 明将报错。注意未申明的变量和声明了未赋值的是不一样的。

## 25.看下列代码,输出什么？解释原因。

var undefined;

undefined == null; // true

1 == true; // true

2 == true; // false

0 == false; // true

0 == ''; // true

NaN == NaN; // false

[] == false; // true

[] == ![]; // true

undefined与null相等，但不恒等（===）

一个是number一个是string时，会尝试将string转换为number

尝试将boolean转换为number，0或1

尝试将Object转换成number或string，取决于另外一个对比量的类型

所以，对于0、空字符串的判断，建议使用 “===” 。“===”会先判断两边的值类型， 类型不匹配时为false。

那么问题来了，看下面的代码，输出什么，foo的值为什么？

var foo = "11"+2-"1";

console.log(foo);

console.log(typeof foo);

执行完后foo的值为111，foo的类型为String。

## 已知数组 var stringArr = [“This”,“ is”， “Baidu”, “Campus” ]， Alert出 “This is Baidu Campus”;

alert(stringArray.join(“”))

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

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

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

|  |
| --- |
| var d = new Date();  // 获取年，getFullYear()返回4位的数字  var year = d.getFullYear();  // 获取月，月份比较特殊，0是1月，11是12月  var month = d.getMonth() + 1;  var d = new Date();  // 获取年，getFullYear()返回4位的数字  var year = d.getFullYear();  // 获取月，月份比较特殊，0是1月，11是12月  var month = d.getMonth() + 1; |

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

|  |
| --- |
| "<tr><td>{$id}</td><td>{$id}\_{$name}</td></tr>".replace(/{\$id}/g, '10').replace(/{\$name}/g, 'Tony'); |

## **为了保证页面输出安全，我们经常需要对一些特殊的字符进行转义，请写一个函数escapeHtml，将<, >, &, “进行转义**

|  |
| --- |
| function escapeHtml(str) {  return str.replace(/[<>”&]/g, function(match) {  switch (match) {  case “<”:  return “&lt;”;  case “>”:  return “&gt;”;  case “&”:  return “&amp;”;  case “\””:  return “&quot;”;  }  });  } |

## **foo = foo||bar ，这行代码是什么意思？为什么要这样写？**

如果foo存在，值不变，否则把bar的值赋给foo。

短路表达式：作为”&&”和”||”操作符的操作数表达式，这些表达式在进行求值时， 只要最终的结果已经可以确定是真或假，求值过程便告终止，这称之为短路求值。

## 看下列代码，将会输出什么?(变量声明提升)

|  |
| --- |
| var foo = 1;  (function(){  console.log(foo);  var foo = 2;  console.log(foo);  })()  答案：输出undefined 和 2。上面代码相当于：  var foo = 1;  (function(){  var foo;  console.log(foo); //undefined  foo = 2;  console.log(foo); // 2;  })()  函数声明与变量声明会被JavaScript引擎隐式地提升到当前作用域的顶部，但是只提升名称不会提升赋值部分。 |

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

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

## 34.把两个数组合并，并删除第二个元素。

var array1 = ['a','b','c'];

var bArray = ['d','e','f'];

var cArray = array1.concat(bArray);

cArray.splice(1,1);

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

function serilizeUrl(url) {

var urlObject = {};

if (/\?/.test(url)) {

var urlString = url.substring(url.indexOf("?") + 1);

var urlArray = urlString.split("&");

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

var urlItem = urlArray[i];

var item = urlItem.split("=");

urlObject[item[0]] = item[1];

}

return urlObject;

}

return null;

}

## 36.正则表达式构造函数var reg=new RegExp(“xxx”)与正则表达字面量var reg=//有什么不同？匹配邮箱的正则表达式？

当使用RegExp()构造函数的时候，不仅需要转义引号（即\”表示”），并且还需要双反斜杠（即\\表示一个\）。使用正则表达字面量的效率更高。

邮箱的正则匹配：

var regMail = /^([a-zA-Z0-9\_-])+@([a-zA-Z0-9\_-])+((.[a-zA-Z0-9\_-]{2,3}){1,2})$/;

## 37.写一个function，清除字符串前后的空格。

if (!String.prototype.trim) {

String.prototype.trim = function() {

return this.replace(/^\s+/, "").replace(/\s+$/,"");

}

}

//测试

var str = " \t\n test string ".trim();

alert(str == "test string"); // alerts "true"

## 38.以下两个变量a和b，a+b的哪个结果是NaN？

A、var a=undefined; b=NaN

B、var a= ‘123’; b=NaN

C、var a =undefined , b =NaN

D、var a=NaN , b='undefined'

## 39.下面的JavaScript语句中，（ ）实现检索当前页面中的表单元素中的所有文本框，并将它们全部清空

A. for(vari=0;i< form1.elements.length;i++) {

if(form1.elements.type==”text”)

form1.elements.value=”";}

B. for(vari=0;i<document.forms.length;i++) {

if(forms[0].elements.type==”text”)

forms[0].elements.value=”";

}

C. if(document.form.elements.type==”text”)

form.elements.value=”";

D. for(vari=0;i<document.forms.length; i++){

for(var j=0;j<document.forms.elements.length; j++){

if(document.forms.elements[j].type==”text”)

document.forms.elements[j].value=”";

}

}

## 40.typeof运算符返回值中有一个跟javascript数据类型不一致，它是？如何判断是不是数组？

Array，Array.isArray(data)

## 41.写出函数DateDemo的返回结果，系统时间假定为今天

function DateDemo(){

var d, s="今天日期是：";

d = new Date();

s += d.getMonth() +1+ "/";

s += d.getDate() + "/";

s += d.getFullYear();

return s;

}

结果：今天日期是：7/21/2016

## 42.写出简单描述html标签（不带属性的开始标签和结束标签）的正则表达式，并将以下字符串中的html标签去除掉

var str = “<div>这里是div<p>里面的段落</p></div>”;

<script type=”text/javascript”>

var reg = /<\/?\w+\/?>/gi;

var str = “<div>这里是div<p>里面的段落</p></div>”;

alert(str.replace(reg,”"));

</script>

## 43.截取字符串abcdefg的efg

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

## 44.简述创建函数的几种方式

第一种（函数声明）：

function sum1(num1,num2){

return num1+num2;

}

第二种（函数表达式）：

var sum2 = function(num1,num2){

return num1+num2;

}

第三种（函数对象方式）：

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

## 45.Javascript如何实现继承？

1.构造继承法

2.原型继承法

3.实例继承法

## 46.Javascript创建对象的几种方式？

1、var obj = {};（使用json创建对象）

如：obj.name = '张三';​

obj.action = function ()

{

alert('吃饭');

}​;

2、var obj = new Object();（使用Object创建对象）

如：obj.name = '张三';​

obj.action = function ()

{

alert('吃饭');

}​;

3、​通过构造函数创建对象。

(1)、使用this关键字​

如：var obj = function (){

this.name ='张三';

this.age = 19;

this.action = function ()

{

alert('吃饭');

}​;

}​

(2)、使用prototype关键字

如：function obj (){}

​ obj.prototype.name ='张三';

​obj.prototype.action=function ()

{

alert('吃饭');

}​;

4、使用内置对象创建对象。

如：var str = new String("实例初始化String");

var str1 = "直接赋值的String";

var func = new Function("x","alert(x)");//示例初始化func

var obj = new Object();//示例初始化一个Object

## 47.js延迟加载的方式有哪些？

1. defer和async

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

3. 按需异步载入js

## 48.哪些操作会造成内存泄漏？

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

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。

1. 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。

2. 闭包

3. 控制台日志

4. 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）

## 49.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');

## 50.写一个获取非行间样式的函数

function getStyle(obj,attr,value)

{

if(!value)

{

if(obj.currentStyle)

{

return obj.currentStyle(attr);

}

else{

obj.getComputedStyle(attr,false);

}

}

else

{

obj.style[attr] = value;

}

}

## 51.字符串反转，如将 '12345678' 变成 '87654321'

//思路：先将字符串转换为数组 split()，利用数组的反序函数 reverse()颠倒数组，再利 用 jion() 转换为字符串

var str = '12345678';

str = str.split('').reverse().join('');

## 52.将数字 12345678 转化成 RMB形式 如： 12,345,678

//思路：先将数字转为字符， str= str + '' ;

//利用反转函数，每三位字符加一个 ','最后一位不加； re()是自定义的反转函数，最后再反转回去！

function re(str) {

str += '';

return str.split("").reverse().join("");

}

function toRMB(num) {

var tmp='';

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

tmp += re(num)[i - 1];

if (i % 3 == 0 && i != re(num).length) {

tmp += ',';

}

}

return re(tmp);

}

## 53.生成5个不同的随机数；

//思路：5个不同的数，每生成一次就和前面的所有数字相比较，如果有相同的，则放 弃当前生成的数字！

var num1 = [];

for(var i = 0; i < 5; i++){

num1[i] = Math.floor(Math.random()\*10) + 1; //范围是 [1, 10]

for(var j = 0; j < i; j++){

if(num1[i] == num1[j]){

i--;

}

}

}

## 54.去掉数组中重复的数字

方法一:

//思路：每遍历一次就和之前的所有做比较，不相等则放入新的数组中！

//这里用的原型 个人做法；

Array.prototype.unique = function(){

var len = this.length,

newArr = [],

flag = 1;

for(var i = 0; i < len; i++, flag = 1){

for(var j = 0; j < i; j++){

if(this[i] == this[j]){

flag = 0; //找到相同的数字后，不执行添加数据

}

}

flag ? newArr.push(this[i]) : '';

}

return newArr;

}

方法二：

var arr=[1,2,3,3,4,4,5,5,6,1,9,3,25,4];

Array.prototype.unique2 = 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;

}

var newArr2=arr.unique2(arr);

alert(newArr2); //输出1,2,3,4,5,6,9,25

## 55.阶乘函数；

//原型方法

Number.prototype.N = function(){

var re = 1;

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

re \*= i;

}

return re;

}

var num = 5;

alert(num.N());

## 56.window.location.search() 返回的是什么？

http://localhost:8080/xxx?ver=1.0&id=123

返回值：?ver=1.0&id=timlq 也就是问号后面的部分

## 57.window.location.reload() 作用？

刷新当前页面。

## 58.javascript 中的垃圾回收机制？

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

## 59.精度问题: JS 精度不能精确到 0.1 所以  。。。。同时存在于值和差值中

var n = 0.3,m = 0.2, i = 0.2, j = 0.1;

alert((n - m) == (i - j)); //false

alert((n-m) == 0.1); //false

alert((i-j)==0.1); //true

## 60.计算字符串字节数：

new function(s){

if(!arguments.length||!s) return null;

if(""==s) return 0;

var l=0;

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

if(s.charCodeAt(i)>255) l+=2; else l+=1; //charCodeAt()得到的是unCode码

} //汉字的unCode码大于 255bit 就是两个字节

alert(l);

}("hello world!");

## 61.匹配输入的字符：第一个必须是字母或下划线开头，长度5-20

var reg = /^[a-zA-Z\_][a-zA-Z0-9\_]{5,20}/,

name1 = 'leipeng',

name2 = '0leipeng',

name3 = '你好leipeng',

name4 = 'hi';

alert(reg.test(name1));

alert(reg.test(name2));

alert(reg.test(name3));

alert(reg.test(name4));

## 62.如何在HTML中添加事件，几种方法？

1、标签之中直接添加 onclick="fun()";

2、JS添加 Eobj.onclick = method;

3、绑定事件 IE： obj.attachEvent('onclick', method)；

FF: obj.addEventListener('click', method, false);

## 63.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

注释

## 64.使用js实现这样的效果：在文本域里输入文字时，当按下enter键时不换行，而是替换成“{{enter}}”,(只需要考虑在行尾按下enter键的情况).

<html>

<head>

<script>

function back(ele,event){

event = event || window.event;

if(event.keyCode==13){

event.returnValue = false;

ele.value+="{{enter}}"

return false;

}

}

</script>

</head>

<body>

<textarea rows="3" cols="40" id="te" onkeypress="back(this,event);"></textarea>

</body>

</html>

## 65.简述readonly与disabled的区别

ReadOnly和Disabled的作用是使用户不能够更改表单域中的内容.

但是二者还是有着一些区别的：

1、Readonly只针对input(text/password)和textarea有效，而disabled对于所有的表单 元素有效，包括select,radio,checkbox,button等。

2、在表单元素使用了disabled后，我们将表单以POST或者GET的方式提交的话，这 个元素的值不会被传递出去，而readonly会将该值传递出去

## 66.为什么扩展javascript内置对象不是好的做法？

因为你不知道哪一天浏览器或javascript本身就会实现这个方法，而且和你扩展的实现有不一致的表现。到时候你的javascript代码可能已经在无数个页面中执行了数年，而浏览器的实现导致所有使用扩展原型的代码都崩溃了。。。

## 67.什么是三元表达式？“三元”表示什么意思？

三元运算符:

三元如名字表示的三元运算符需要三个操作数。

语法是 条件 ? 结果1 : 结果2;. 这里你把条件写在问号(?)的前面后面跟着用冒号(:)分 隔的结果1和结果2。满足条件时结果1否则结果2。

## 68.变量的命名规范以及命名推荐

变量，函数，方法：小写开头，以后的每个单词首字母大写 （驼峰）

构造函数，class：每个单词大写开头

基于实际情况，以动词，名词，谓词来命名。尽量言简意骇，以命名代替注释

## 69.三种弹窗的单词以及三种弹窗的功能

1.alert

//弹出对话框并输出一段提示信息

function ale() {

//弹出一个对话框

alert("提示信息！");

}

2.confirm

//弹出一个询问框，有确定和取消按钮

function firm() {

//利用对话框返回的值 （true 或者 false）

if (confirm("你确定提交吗？")) {

alert("点击了确定");

}

else {

alert("点击了取消");

}

}

3.prompt

//弹出一个输入框，输入一段文字，可以提交

function prom() {

var name = prompt("请输入您的名字", ""); //将输入的内容赋给变量 name ，

//这里需要注意的是，prompt有两个参数，前面是提示的话，后面是当对话框出来后，在对话框里的默认值

if (name)//如果返回的有内容

{

alert("欢迎您：" + name)

}

}

## 70.主流浏览器内核

IE trident 火狐gecko 谷歌苹果webkit Opera：Presto

## 71.JavaScript的循环语句有哪些？

for,for..in,while,do...while

## 72.闭包：下面这个ul，如何点击每一列的时候alert其index？

<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>

//js

window.onload = function() {

var lis = document.getElementById('test').children;

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

lis[i].onclick = (function(i) {

return function() {

alert(i)

};

})(i);

};

}

## 73.列出3条以上ff和IE的脚本兼容问题

(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”)

## 74.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。

## 75.请写出一个程序，在页面加载完成后动态创建一个form表单，并在里面添加一个input对象并给它任意赋值后义post方式提交到：http://127.0.0.1/save.php

window.onload=function(){

var form=document.createElement("form");

form.setAttribute("method", "post");

form.setAttribute("action", "http://127.0.0.1/save.php");

var input=document.createElement("input");

form.appendChild(input);

document.body.appendChild(form);

input.value="cxc";

form.submit();//提交表单

}

## 76.用JavaScript实现升序排序。数据为23、45、18、37、92、13、24

//升序算法

function sort(arr){

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

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

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

var c=arr[j];//交换两个变量的位置

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

arr[j+1]=c;

}

};

};

return arr.toString();

}

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

## 请说出下列输出的结果

var User = {

count = 1，

getCount：function（）{

return this.count;

}

}

console.log(User.getCount());

var func = User.getCount;

console.log(func());

1 undefined（因为是window对象执行了func函数）;

## 78.用程序实现找到html中id名相同的元素？

<body>

<form id='form1'>

<div id='div1'></div>

<div id='div2'></div>

<div id='div3'></div>

<div id='div4'></div>

<div id='div5'></div>

<div id='div3'>id名重复的元素</div>

</form>

</body>

var nodes=document.querySelectorAll("#form1>\*");

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

var attr=nodes[i].getAttribute("id");

var s=1;

for(var j=i+1;j<len;j++){

if(nodes[j].getAttribute("id")==attr){

s++;

alert("id为："+attr+"的元素出现"+s+"次");

}

}

}

## 79.程序中捕获异常的方法？ js基础提及

window.error

try{}catch(){}finally{}

## 下列控制台都输出什么

|  |
| --- |
| 第1题：  function setName(){  name="张三";  }  setName();  console.log(name);  答案："张三"  第2题：  //考点：1、变量声明提升 2、变量搜索机制  var a=1;  function test(){  console.log(a);  var a=1;  }  test();  答案：undefined  第3题：  var b=2;  function test2(){  window.b=3;  console.log(b);  }  test2();  答案：3  第4题：  c=5;//声明一个全局变量c  function test3(){  window.c=3;  console.log(c); //答案：undefined，原因：由于此时的c是一个局部变量c，并且没有被赋值  var c;  console.log(window.c);//答案：3，原因：这里的c就是一个全局变量c  }  test3();  第5题：  var arr = [];  arr[0] = 'a';  arr[1] = 'b';  arr[10] = 'c';  alert(arr.length); //答案：11  console.log(arr[5]); //答案：undefined  第6题：  var a=1;  console.log(a++); //答案：1  console.log(++a); //答案：3  第7题：  console.log(null==undefined); //答案：true  console.log("1"==1); //答案：true，因为会将数字1先转换为字符串1  console.log("1"===1); //答案：false，因为数据类型不一致  第8题：  typeof 1; "number"  typeof "hello"; "string"  typeof /[0-9]/; "object"  typeof {}; "object"  typeof null; "object"  typeof undefined; "undefined"  typeof [1,2,3]; "object"  typeof function(){}; //"function"  第9题：  parseInt(3.14); //3  parseFloat("3asdf"); //3  parseInt("1.23abc456");  parseInt(true);//"true" NaN  第10题：  //考点：函数声明提前  function bar() {  return foo;  foo = 10;  function foo() {}  //var foo = 11;  }  alert(typeof bar());//"function"  第11题：  //考点：函数声明提前  var foo = 1;  function bar() {  foo = 10;  return;  function foo() {}  }  bar();  alert(foo);//答案：1  第12题：  console.log(a);//是一个函数  var a = 3;  function a(){}  console.log(a);////3  第13题：  //考点：对arguments的操作  function foo(a) {  arguments[0] = 2;  alert(a);//答案：2，因为：a、arguments是对实参的访问，b、通过arguments[i]可以修改指定实参的值  }  foo(1);  第14题：  function foo(a) {  alert(arguments.length);//答案：3，因为arguments是对实参的访问  }  foo(1, 2, 3);  第15题  bar();//报错  var foo = function bar(name) {  console.log("hello"+name);  console.log(bar);  };  //alert(typeof bar);  foo("world");//"hello"  console.log(bar);//undefined  console.log(foo.toString());  bar();//报错  第16题：  function test(){  console.log("test函数");  }  setTimeout(function(){  console.log("定时器回调函数");  }, 0)  test();  结果：  test函数  定时器回调函数 |

## 81.请说出三种减低页面加载时间的方法

1、压缩css、js文件

2、合并js、css文件，减少http请求

3、外部js、css文件放在最底下

4、减少dom操作，尽可能用变量替代不必要的dom操作

## 82.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

|  |
| --- |
| // 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;  }  } |

## 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 |

## 84.定义一个log方法，让它可以代理console.log的方法。

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]) 。

## 85.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  }  });  }  }; |

## （设计题）想实现一个对页面某个节点(比如：对某个div)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

给需要拖拽的节点绑定mousedown, mousemove, mouseup事件

mousedown事件触发后，开始拖拽

mousemove时，需要通过event.clientX和clientY获取拖拽位置，并实时更新位置

mouseup时，拖拽结束

需要注意浏览器边界的情况

## 87. 数组和对象有哪些原生方法，列举一下？

1. Array.concat( ) 连接数组
2. Array.join( ) 将数组元素连接起来以构建一个字符串
3. Array.length 数组的大小
4. Array.pop( ) 删除并返回数组的最后一个元素
5. Array.push( ) 给数组添加元素
6. Array.reverse( ) 颠倒数组中元素的顺序
7. Array.shift( ) 将元素移出数组
8. Array.slice( ) 返回数组的一部分
9. Array.sort( ) 对数组元素进行排序
10. Array.splice( ) 插入、删除或替换数组的元素
11. Array.toLocaleString( ) 把数组转换成局部字符串
12. Array.toString( ) 将数组转换成一个字符串
13. Array.unshift( ) 在数组头部插入一个元素
15. Object.hasOwnProperty( ) 检查属性是否被继承
16. Object.isPrototypeOf( ) 一个对象是否是另一个对象的原型
17. Object.propertyIsEnumerable( ) 是否可以通过for/in循环看到属性
18. Object.toLocaleString( ) 返回对象的本地字符串表示
19. Object.toString( ) 定义一个对象的字符串表示
20. Object.valueOf( ) 指定对象的原始值

## 88.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。

## 89.在JS中有哪些会被隐式转换为false

Undefined、null、关键字false、NaN、零、空字符串

## 90.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=”utf-8”;

## 91.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;  }  }; |

## 92.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'}

# 2.DOM相关

## 1.target与currentTarget区别

target在事件流的目标阶段；currentTarget在事件流的捕获，目标及冒泡阶段。只有当事件流处在目标阶段的时候，两个的指向才是一样的， 而当处于捕获和冒泡阶段的时候，target指向被单击的对象而currentTarget指向当前事件活动的对象（一般为父级）。

## 2. DOM的增删改查操作？

（1）创建新节点

createDocumentFragment() //创建一个DOM片段

createElement() //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() //在已有的子节点前插入一个新的子节点

（3）查找

getElementsByTagName() //通过标签名称

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

getElementById() //通过元素Id，唯一性

## 请描述DOM事件流的过程？

DOM事件流包括三个阶段：事件捕获阶段、处于目标阶段、事件冒泡阶段。首先发生的事件捕获，为截获事件提供机会。然后是实际的目标接受事件。最后一个阶段是时间冒泡阶段，可以在这个阶段对事件做出响应。

## 事件委托的原理，作用，和触发事件的对象是谁？

因为事件具有冒泡机制，因此我们可以利用冒泡的原理，把事件加到父级上，触发执行效果。这样做的好处当然就是提高性能了

最重要的是通过event.target判断触发事件的对象是谁

## 5.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

# 3.BOM相关

# 4.JS高级

## 1.函数高级

### 1.闭包

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

闭包的缺点就是常驻内存，会增大内存使用量，使用不当很容易造成内存泄露。

为什么要使用闭包：

为了设计私有方法和变量，避免全局变量污染 希望一个变量长期驻扎在内存中

view detail: https://segmentfault.com/a/1190000000652891

### 2.原型与原型链

当从一个对象那里调取属性或方法时，如果该对象自身不存在这样的属性或方法，就会去自己关联的prototype对象那里寻找，如果prototype没有，就会去prototype关联的前辈prototype那里寻找，如果再没有则继续查找Prototype.Prototype引用的对象，依次类推，直到Prototype.….Prototype为undefined（Object的Prototype就是undefined）从而形成了所谓的“原型链”。

其中foo是Function对象的实例。而Function的原型对象同时又是Object的实例。这样就构成了一条原型链。

instanceof 确定原型和实例之间的关系

用来判断某个构造函数的prototype属性是否存在另外一个要检测对象的原型链上

对象的\_\_proto\_\_指向自己构造函数的prototype。obj.\_\_proto\_\_.\_\_proto\_\_...的原型链由此产生，包括我们的操作符instanceof正是通过探测obj.\_\_proto\_\_.\_\_proto\_\_... === Constructor.prototype来验证obj是否是Constructor的实例。

function C(){}

var o = new C(){}

//true 因为Object.getPrototypeOf(o) === C.prototype

o instanceof C

instanceof只能用来判断对象和函数，不能用来判断字符串和数字

isPrototypeOf

用于测试一个对象是否存在于另一个对象的原型链上。

判断父级对象 可检查整个原型链

### 3.作用域与作用域链

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。

全局函数无法查看局部函数的内部细节，但局部函数可以查看其上层的函数细节，直至全局细节。

当需要从局部函数查找某一属性或方法时，如果当前作用域没有找到，就会上溯到上层作用域查找，

直至全局函数，这种组织形式就是作用域链。

### 4.apply, call和bind有什么区别?

参考答案：三者都可以把一个函数应用到其他对象上，call、apply是修改函数的作用域（修改this指向），并且立即执行，而bind是返回了一个新的函数，不是立即执行．apply和call的区别是apply接受数组作为参数，而call是接受逗号分隔的无限多个参数列表，

Array.prototype.slice.call(null, args)

function getMax(arr){

return Math.max.apply(null, arr);

}

//call

function foo() {

console.log(this);//{id: 42}

}

foo.call({ id: 42 });

如果该方法是非严格模式代码中的函数，则null和undefined将替换为全局对象，并且原始值将被包装。 当你调用apply传递给它null时，就像是调用函数而不提供任何对象

### 5.谈谈对this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；

### 6.那些操作会造成内存泄漏？

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

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）

### 7.深入贯彻闭包思想，全面理解JS闭包形成过程

https://segmentfault.com/a/1190000009886713

### 8.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);

}

## 2.对象高级

### 1.js继承方式及其优缺点

原型链继承的缺点

一 是字面量重写原型会中断关系，使用引用类型的原型，并且子类型还无法给超类型传递参数。

借用构造函数（类式继承）

借用构造函数虽然解决了刚才两种问题，但没有原型，则复用无从谈起。所以我们需要原型链+借用构造函数的模式，这种模式称为组合继承

组合式继承

组合式继承是比较常用的一种继承方法，其背后的思路是 使用原型链实现对原型属性和方法的继承，而通过借用构造函数来实现对实例属性的继承。这样，既通过在原型上定义方法实现了函数复用，又保证每个实例都有它自己的属性。

### 2.上下文环境对象

### 3.new操作符具体做了什么

1、创建一个空对象，并且this变量引用该对象，同时继承了该函数的原型（实例对象通过\_\_proto\_\_属性指向原型对象；obj.\_\_proto\_\_ = Base.prototype;） 2、属性和方法被加入到 this 引用的对象中。

function Animal(name) {

this.name = name;

}

Animal.prototype.run = function() {

console.log(this.name + 'can run...');

}

var cat = new Animal('cat');

//模拟过程

new Animal('cat')=function(){

let obj={}; //创建一个空对象

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

//把该对象的原型指向构造函数的原型对象，就建立起原型了：obj->Animal.prototype->Object.prototype->null

return Animal.call(obj,'cat');// 绑定this到实例化的对象上

}

### 4.创建对象的几种方式

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="Mark";

person.age="25";

person.work=function(){

alert(person.name+" hello...");

}

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 Pet(name,age,hobby){

this.name=name;//this作用域：当前对象

this.age=age;

this.hobby=hobby;

this.eat=function(){

alert("我叫"+this.name+",我喜欢"+this.hobby+",是个程序员");

}

}

var maidou =new Pet("麦兜",25,"coding");//实例化、创建对象

maidou.eat();//调用eat方法

4、用工厂方式来创建（内置对象）

var wcDog =new Object();

wcDog.name="旺财";

wcDog.age=3;

wcDog.work=function(){

alert("我是"+wcDog.name+",汪汪汪......");

}

wcDog.work();

5、用原型方式来创建

function Dog(){

}

Dog.prototype.name="旺财";

Dog.prototype.eat=function(){

alert(this.name+"是个吃货");

}

var wangcai =new Dog();

wangcai.eat();

5、用混合方式来创建

function Car(name,price){

this.name=name;

this.price=price;

}

Car.prototype.sell=function(){

alert("我是"+this.name+"，我现在卖"+this.price+"万元");

}

var camry =new Car("凯美瑞",27);

camry.sell();

## 3.线程机制

### 1.同步和异步的区别?

同步：阻塞的

-张三叫李四去吃饭，李四一直忙得不停，张三一直等着，直到李四忙完两个人一块去吃饭

=浏览器向服务器请求数据，服务器比较忙，浏览器一直等着（页面白屏），直到服务器返回数据，浏览器才能显示页面

异步：非阻塞的

-张三叫李四去吃饭，李四在忙，张三说了一声然后自己就去吃饭了，李四忙完后自己去吃

=浏览器向服务器请求数据，服务器比较忙，浏览器可以自如的干原来的事情（显示页面），服务器返回数据的时候通知浏览器一声，浏览器把返回的数据再渲染到页面，局部更新

# 5.ES6相关

## 1.谈一谈let与var和const的区别？

let为ES6新添加申明变量的命令，它类似于var，但是有以下不同：

let命令不存在变量提升，如果在let前使用，会导致报错

暂时性死区的本质，其实还是块级作用域必须“先声明后使用”的性质。

let，const和class声明的全局变量不是全局对象的属性。

const声明的变量与let声明的变量类似，它们的不同之处在于，const声明的变量只可以在声明时赋值，不可随意修改，否则会导致SyntaxError（语法错误）。

const只是保证变量名指向的地址不变，并不保证该地址的数据不变。const可以在多个模块间共享 let 暂时性死区的原因：var 会变量提升，let 不会。

## 2.箭头函数

箭头函数不属于普通的 function，所以没有独立的上下文。箭头函数体内的this对象，就是定义时所在的对象，而不是使用时所在的对象。 由于箭头函数没有自己的this，函数对象中的call、apply、bind三个方法，无法"覆盖"箭头函数中的this值。 箭头函数没有原本(传统)的函数有的隐藏arguments对象。 箭头函数不能当作generators使用，使用yield会产生错误。

在以下场景中不要使用箭头函数去定义：

1. 定义对象方法、定义原型方法、定义构造函数、定义事件回调函数。

2. 箭头函数里不但没有 this，也没有 arguments, super ……

## 3. Map和Set

Symbol，Map和Set

Map 对象保存键值对。一个对象的键只能是字符串或者 Symbols，但一个 Map 的键可以是任意值。 Set 对象允许你存储任何类型的唯一值，Set对象是值的集合，Set中的元素只会出现一次 Symbol 是一种特殊的、不可变的数据类型，可以作为对象属性的标识符使用(Symbol([description]) )

let mySet = new Set()

mySet.add(1)

mySet.add('hello')

mySet.add('hello')

console.log(mySet.size);//2

console.log(mySet);//Set {1,'hello'}

//Map保存键值对也不能有重复的

let myMap = new Map();

let key1 = 'China',key2 = 'America';

myMap.set(key1,'welcome')

myMap.set(key2,'gold bless you')

console.log(myMap);//Map { 'China' => 'welcome', 'America' => 'gold bless you' }

console.log(myMap.get(key1));//welcome

console.log(myMap.get(key2));//gold bless you

let mySymbol = Symbol('symbol1');

let mySymbol2 = Symbol('symbol1');

console.log(mySymbol == mySymbol2);//false

//Symbols 在 for...in 迭代中不可枚举。

let obj = {}

obj['c'] = 'c'

obj.d ='d'

obj[Symbol('a')] = 'a'

obj[Symbol.for('b')] = 'b'

for(let k in obj){

console.log(k);//logs 'c' and 'd'

}

for...of可以用来遍历数组，类数组对象，argument，字符串，Map和Set，for...in用来遍历对象

## 4.Promise实现原理

现在回顾下Promise的实现过程，其主要使用了设计模式中的观察者模式：

通过Promise.prototype.then和Promise.prototype.catch方法将观察者方法注册到被观察者Promise对象中，同时返回一个新的Promise对象，以便可以链式调用。

被观察者管理内部pending、fulfilled和rejected的状态转变，同时通过构造函数中传递的resolve和reject方法以主动触发状态转变和通知观察者。

Promise.then()是异步调用的，这也是Promise设计上规定的，其原因在于同步调用和异步调用同时存在会导致混乱。

为了暂停当前的 promise，或者要它等待另一个 promise 完成，只需要简单地在 then() 函数中返回另一个 promise。

Promise 也有一些缺点。首先，无法取消 Promise，一旦新建它就会立即执行，无法中途取消。其次，如果不设置回调函数，Promise 内部抛出的错误，不会反应到外部。第三，当处于 Pending 状态时，无法得知目前进展到哪一个阶段（刚刚开始还是即将完成）。

一般来说，不要在then方法里面定义Reject状态的回调函数（即then的第二个参数），总是使用catch方法，理由是更接近同步的写法。 then的第二个函数参数和catch等价

Promise.all和Promise.race的区别？

Promise.all 把多个promise实例当成一个promise实例,当这些实例的状态都发生改变时才会返回一个新的promise实例，才会执行then方法。 Promise.race 只要该数组中的 Promise 对象的状态发生变化（无论是resolve还是reject）该方法都会返回。

## 5.Object.is() 与原来的比较操作符“ ===”、“ ==”的区别？

两等号判等，会在比较时进行类型转换；

三等号判等(判断严格)，比较时不进行隐式类型转换,（类型不同则会返回false）；

Object.is 在三等号判等的基础上特别处理了 NaN 、-0 和 +0 ，保证 -0 和 +0 不再相同，

但 Object.is(NaN, NaN) 会返回 true.

Object.is 应被认为有其特殊的用途，而不能用它认为它比其它的相等对比更宽松或严格。

## 6.前端面试之ES6篇

https://juejin.im/post/59c8aec0f265da065c5e965e

# 6.跨域

## 1.JSONP

回调函数+数据就是 JSON With Padding，简单、易部署。（做法：动态插入script标签，设置其src属性指向提供JSONP服务的URL地址，查询字符串中加入 callback 指定回调函数，返回的 JSON 被包裹在回调函数中以字符串的形式被返回，需将script标签插入body底部）。缺点是只支持GET，不支持POST（原因是通过地址栏传参所以只能使用GET）

## 2.document.domain 跨子域

document.domain 跨子域 （ 例如a.qq.com嵌套一个b.qq.com的iframe ，如果a.qq.com设置document.domain为qq.com 。b.qq.com设置document.domain为qq.com， 那么他俩就能互相通信了，不受跨域限制了。 注意：只能跨子域）

## 3. iframe

window.name + iframe ==> http://www.tuicool.com/articles/viMFbqV，支持跨主域。不支持POST

## 4.postMessage()

HTML5的postMessage()方法允许来自不同源的脚本采用异步方式进行有限的通信，可以实现跨文本档、多窗口、跨域消息传递。适用于不同窗口iframe之间的跨域

## 5.CORS

CORS（Cross Origin Resource Share）对方服务端设置响应头

设置相应头：”Access-Control-Allow-Origin“

CORS请求默认不发送Cookie和HTTP认证信息。如果要把Cookie发到服务器，一方面要服务器同意，指定Access-Control-Allow-Credentials字段。

## 6.服务端代理

服务端代理 在浏览器客户端不能跨域访问，而服务器端调用HTTP接口只是使用HTTP协议，不会执行JS脚本，不需要同源策略，也就没有跨越问题。简单地说，就是浏览器不能跨域，后台服务器可以跨域。（一种是通过http-proxy-middleware插件设置后端代理；另一种是通过使用http模块发出请求）

## 7.Ajax

### 1.ajax请求和原理

var xhr = new XMLHTTPRequest();

// 请求 method 和 URI

xhr.open('GET', url);

// 请求内容

xhr.send();

// 响应状态

xhr.status

// xhr 对象的事件响应

xhr.onreadystatechange = function() {}

xhr.readyState

// 响应内容

xhr.responseText

AJAX的工作原理

Ajax的工作原理相当于在用户和服务器之间加了—个中间层(AJAX引擎),使用户操作与服务器响应异步化。　Ajax的原理简单来说通过XmlHttpRequest对象来向服务器发异步请求，从服务器获得数据，然后用javascript来操作DOM而更新页面。

ajax优缺点

优点：无刷新更新数据 异步与服务器通信 前后端负载均衡

缺点：

1）ajax干掉了Back和history功能，对浏览器机制的破坏 2）对搜索引擎支持较弱 3）违背了URI和资源定位的初衷

### 2.fetch和Ajax有什么不同

XMLHttpRequest 是一个设计粗糙的 API，不符合关注分离（Separation of Concerns）的原则，配置和调用方式非常混乱，而且基于事件的异步模型写起来也没有现代的 Promise，generator/yield，async/await 友好。

fetch 是浏览器提供的一个新的 web API，它用来代替 Ajax（XMLHttpRequest），其提供了更优雅的接口，更灵活强大的功能。 Fetch 优点主要有：

语法简洁，更加语义化

基于标准 Promise 实现，支持 async/await

fetch(url).then(response => response.json())

.then(data => console.log(data))

.catch(e => console.log("Oops, error", e))

### 3.GET和POST的区别

GET和POST的区别

GET使用URL或Cookie传参，而POST将数据放在BODY中，这个是因为HTTP协议用法的约定。并非它们的本身区别。

GET方式提交的数据有长度限制，则POST的数据则可以非常大，这个是因为它们使用的操作系统和浏览器设置的不同引起的区别。也不是GET和POST本身的区别。

POST比GET安全，因为数据在地址栏上不可见，这个说法没毛病，但依然不是GET和POST本身的区别。

GET和POST最大的区别主要是GET请求是幂等性的，POST请求不是。（幂等性：对同一URL的多个请求应该返回同样的结果。）因为get请求是幂等的，在网络不好的隧道中会尝试重试。如果用get请求增数据，会有重复操作的风险，而这种重复操作可能会导致副作用

### 4.GET,POST,PUT,Delete

1. GET请求会向数据库获取信息，只是用来查询数据，不会修改，增加数据。使用URL传递参数，对所发送的数量有限制，一般在2000字符

2. POST向服务器发送数据，会改变数据的种类等资源，就像insert操作一样，会创建新的内容，大小一般没有限制，POST安全性高，POST不会被缓存

3. PUT请求就像数据库的update操作一样，用来修改数据内容，不会增加数据种类

4. Delete用来删除操作

### 5.缓存，存储相关（cookie，web storage和session）

cookie优点： 1.可以解决HTTP无状态的问题，与服务器进行交互 缺点： 1.数量和长度限制，每个域名最多20条，每个cookie长度不能超过4kb 2.安全性问题。容易被人拦截 3.浪费带宽，每次请求新页面，cookie都会被发送过去

cookie和session区别

1.cookie数据存放在客户的浏览器上，session数据放在服务器上。 2.session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能。考虑到减轻服务器性能方面，应当使用COOKIE。

sessionStorage是当前对话的缓存，浏览器窗口关闭即消失，localStorage持久存在，除非清除浏览器缓存。

页面缓存原理

页面缓存状态是由http header决定的，一个浏览器请求信息，一个是服务器响应信息。主要包括Pragma: no-cache、Cache-Control、 Expires、 Last-Modified、If-Modified-Since。

### 6.什么是Cookie 隔离？（或者说：请求资源的时候不要让它带cookie怎么做）

如果静态文件都放在主域名下，那静态文件请求的时候都带有的cookie的数据提交给server的，非常浪费流量，

所以不如隔离开。

因为cookie有域的限制，因此不能跨域提交请求，故使用非主要域名的时候，请求头中就不会带有cookie数据，

这样可以降低请求头的大小，降低请求时间，从而达到降低整体请求延时的目的。

同时这种方式不会将cookie传入Web Server，也减少了Web Server对cookie的处理分析环节，

提高了webserver的http请求的解析速度。

### 7. Ajax 解决浏览器缓存问题？

1、在ajax发送请求前加上 anyAjaxObj.setRequestHeader("If-Modified-Since","0")。

2、在ajax发送请求前加上 anyAjaxObj.setRequestHeader("Cache-Control","no-cache")。

3、在URL后面加上一个随机数： "fresh=" + Math.random();。

4、在URL后面加上时间戳："nowtime=" + new Date().getTime();。

5、如果是使用jQuery，直接这样就可以了 $.ajaxSetup({cache:false})。这样页面的所有ajax都会执行这条语句就是不需要保存缓存记录。