封装方法 smalldemo

1. 闭包的作用：

累加器：function add(){

var num=0;

function b(){

num++;

document.write(num);

}

return b;

}

var testAdd=Add();

testAdd();

testAdd();

testAdd();

缓存：function eater(){

var fruit="";

return{

eat:function(){

if(fruit!=""){

document.write('I am eating'+fruit);

fruit="";

}else{

document.write('There is nothing!');

}

},

put:function(myFruit){

fruit=myFruit;

}

}

}

var eater1=eater();

eater1.eat();

eater1.put('apple');

eater.eat();

1. 十个十：

function retB(){

var arr=[];

for(var i=0;i<10;i++){

arr[i]=function(){

document.write(i);

}

}

return arr;

}

var testArr=retB();

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

testArr[i]();

}

改正0-9

function retB(){

var arr=[];

for(var i=0;i<10;i++){

(function(n){

arr[n]=function(){

document.write(n);

}

}(i));

}

return arr;

}

var testArr=retB();

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

testArr[j]();

}

0123456789

1. 圣杯模式：

p.prototype

F.prototype=p.prototype

c.prototype=new F();

p c

**圣杯模式**

function inherit(C,P){

function F(){}

F.prototype=P.prototype;

C.prototype=new F();

C.prototype.constructor=C;

C.prototype.uber=P;

}

function Child(){}

function Parent(){}

Parent.prototype.name='sunny';

inherit(Child,Parent);

document.write(new Child().name);//sunny

document.write(new Child().constructor);//child

<==>

\*var inherit=(function(){

var F=function(){};

return function(C,P){

F.prototype=P.prototype;

C.prototype=new F();

C.prototype.constructor=C;

C.prototype.uber=P.Prototype;

}

}());

function Child(){}

function Parent(){}

Parent.prototype.name='sunny';

inherit(Child,Parent);

document.write(new Child().name);//sunny

document.write(new Child().constructor);//child

//F充当功能没具体意义 F为私有化变量 在作用链中，不在函数体中 闭包

**笔试原题**

**var name='222';**

**var a={**

**name:"111",**

**say:function(){**

**document.write(this.name);**

**}**

**}**

**var fun=a.say;**

**fun();//222**

**a.say();//111**

**var b={**

**name:"333",**

**say:function(fun){**

**fun()**

**}**

**}**

**b.say(a.say);//222**

**b.say=a.say;**

**b.say;//333**

**分析**

**（1）将方法拿过来，this指向全局即222**

**（2）a调用say**

**（3）传参为a.say将方法体传过来执行fun()自执行，没有调用与1类似**

**（4）b say的方法体换成a的方法体**

**arguments.callee函数自身自调用**

**6.浅层克隆**

function extend(parent,child){

var child =child||{};

for(var prop in parent){

if(parent.hasOwnProperty(prop)){

child[prop]=parent[prop];

}

}

}

var obj={

name:'sunny',

age:123,

money:[100,200,300,400]

}

var obj2={

}

extend(obj,obj2);

obj2.name='cherry';

document.write(obj.name);//sunny

obj2.money.push(500);

document.write(obj.money);//100 200 300 400 500

1. 按钮

<html>

<head lang="en,zh-cnm">

<title>pratice</title>

<meta charset="utf-8">

<style type="text/css">

.wrapper{

}

button{

background-color:#ffc;

}

.wrapper div{

border:2px solid red;

height: 100px;

width: 100px;

display: none;

}

.active{

background-color: orange;

}

</style>

</head>

<body>

<div class="wrapper">

<button class="active">111</button>

<button>222</button>

<button>333</button>

<div style="display:block">111</div>

<div>222</div>

<div>333</div>

</div>

<script type="text/javascript" >

var gBtn=document.getElementsByTagName('button'),

oWrapper=document.getElementsByClassName('wrapper')[0],

oEle=oWrapper.getElementsByTagName('div');

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

(function(i){

gBtn[i].onclick=function(){

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

gBtn[j].className='';

oEle[j].style.display='none';

}

oEle[i].style.display='block';

this.className='active';

}

}(i))

}

</script>

</body>

</html>

9.

<div class="example">

<p class="slogan">姬成，你最帅!</p>

</div>

提示 dom.className 可以读写class

代码<script type="text/javascript">

var p=document.createElement('p');

p.setAttribute('class','slogan');

var text=document.createTextNode('handsome');

p.appendChild(text);

var div=document.createElement('div');

div.setAttribute('class','example');

div.appendChild(p);

document.body.appendChild(div);

</script>

1. .遍历元素节点树，要求不能用children属性

解1：

<body>

<div id="only">

<p>

<span>

<em></em>

</span>

</p>

<div>

<a href="3"></a>

</div>

</div>

<script type="text/javascript">

function retChild(node){

var child=node.childNodes,

len=child.length;

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

if(child[i].nodeType===1){

console.log(child[i]);

child[i].hasChildNodes()&&retChild(child[i]);

}

}

}

</script>

</body>

retChild(document.body);

practice.html:27 <div id=​"only">​…​</div>​

practice.html:27 <p>​…​</p>​

practice.html:27 <span>​…​</span>​

practice.html:27 <em>​</em>​

practice.html:27 <div>​…​</div>​

practice.html:27 <a href=​"3">​</a>​

practice.html:27 <script type=​"text/​javascript">​…​</script>​

Undefined

12封装函数，返回元素e的第n层祖先元素

解2：<body>

<div id="only">

<div>

<p>

<a href="#"></a>

</p>

</div>

</div>

<script type="text/javascript">

var eleA=document.getElementsByTagName('a')[0];

function retParent(e,n){

if(n==0){

return e;

}

var n=n||1;

for(var i=0;i<n&&e;i++){

e=e.parentNode;

}

return e;

}

</script>

retParent(eleA,3);

<div id=​"only">​…​</div>​

retParent(eleA,0);

<a href=​"#">​</a>

13封装函数，返回元素e的第n个兄弟节点，n为正，返回后面的兄弟节点，n为负，返回前面的，n为0，返回自己。

<body>

<div>123</div>

<p id="demo">234</p>

<span>234</span>

<a href="#" id="only"></a>

<script type="text/javascript">

var eleP=document.getElementById('demo');

function retSibling(e,n){

if(n===0||n===undefined){

return e;

}

while(n!==0&&e){

if(n>0){

if(e.nextElementSibling){

e=e.nextElementSibling;

}else{

for(e=e.nextSibling;e&&e.nodeType!==1;e=e.nextSibling){}

}

n--;

}else{

if(e.previousElementSibling){

e=e.previousElementSibling;

}else{

for(e=e.previousSibling;e&&e.nodeType!==1;e=e.previousSibling){}

}

n++;

}

}

return e;

}

</script>

</body>

retSibling(eleP,2);

<a href=​"#" id=​"only">​</a>​

retSibling(eleP,4);

null

retSibling(eleP,-5);

Null

14编辑函数，封装children功能，解决以前部分浏览器的兼容性问题

<body>

<div>

<span>1</span>

<p>2</p>

</div>

<script type="text/javascript">

Element.prototype.retChildren=function(){

var child =this.childNodes,

len = child.length,

obj={

"length":0,

"push":Array.prototype.push

}

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

if(child[i].nodeType==1){

obj.push(child[i]);

}

}

return obj;

}

var div=document.getElementsByTagName('div')[0];

</script>

</body>

div

<div>​…​</div>​

div.retChildren()

Object {0: span, 1: p, length: 2}

15.自己封装hasChildren()方法，不可用children属性

解5：

<body>

<div>

<span>1</span>

<p>2</p>

</div>

<script type="text/javascript">

Element.prototype.hasChildren=function(){

var child=this.childNodes,

len = child.length;

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

if(child[i].nodeType==1){

return true;

}

}

return false;

}

var div=document.getElementsByTagName('div')[0];

</script>

</body>

div

<div>​…​</div>​

div.hasChildren()

True

16.字符串长度：

代码：

String.prototype.length=function(){

var len=0;

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

if(this.charCodeAt(i)<=255)

{len+=1;}

else{len+=2;}

}

return len;

}

var string="1fd2";

document.write(string.length);

或

String.prototype.retCodeLen=function(){

var len=this.length,

bytes=len;

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

if(this.charCodeAt(i)>255){

bytes++;

}

}

return bytes;

}

var str="abc,成哥你好";

document.write(str.retCodeLen());

17.数组去重

var arr=['a','b','c','d','a','b','c','d'];

Array.prototype.unique=function(){

var len=this.length,

obj={},

arr=[];

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

if(!obj[this[i]]){

obj[this[i]]=0];

arr.push(this[i]);

}

}

return arr;

}

document.write(arr.unique());

18.11深层克隆

var arr=[1,2,3,4];

document.write(Object.prototype.toString.call(arr));

//[object Array]

var arr={};

document.write(Object.prototype.toString.call(arr));

//[object Object]

代码：function cloneDeep(parent,child){

var toStr=Object.prototype.toString,

arrStr='[object Array]',

child=child||{};

for(var prop in parent){

if(parent.hasOwnProperty([prop])){

if(typeof(parent[prop])=='object'){

child[prop]=(toStr.call(parent[prop])==arrStr)?[]:{};

cloneDeep(parent[prop],child[prop]);

}else{

child[prop]=parent[prop];

}

}

}

}

var obj={

name:'sunny',

wife:{

name:'abc'

},

money:[100,100]

}

var obj1={

}

cloneDeep(obj,obj1);

obj1.wife.name="abcd";

obj1.money.push(100);

document.write(obj.wife.name);

document.write(obj.money);

Abc 100 100

19

封装函数，打印当前是何年何月何日何时，几分几秒：

function retTime(){

var date =new Date();

document.write(date.getFullYear()+"年"+(date.getMonth()+1)+"月"+date.getDate()"+”日”，星期"+date.getDay()+"。"+date.getHours()+"时"+date.getMinutes()+"分"+date.getSeconds()+"秒");

}

retTime()

Undefined

2016年3月6日，星期0。21时52分58秒

20 封装函数insertAfter()；功能类似insertBefore();

解1：<body>

<div>

<p>1</p>

<strong>2</strong>

</div>

<script type="text/javascript">

Element.prototype.inserAfter= function(targetNode,afterNode){

var nextSib=afterNode.nextElementSibling;

if(this.children.length<=1||!nextSib){

this.appendChild(targetNode);

}else{

this.inserBefore(targetNode,nextSib);

}

}

var div=document.getElementsByTagName('div')[0];

var strong=document.getElementsByTagName('strong')[0];

var p=document.getElementsByTagName('p')[0];

var span=document.createElement('span');

</script>

</body>

div.inserAfter(span,strong)

Undefined

21..封装remove(); 使得child.remove()直接可以销毁自身

<body>

<div>

<p>1</p>

<strong>2</strong>

</div>

<script type="text/javascript">

var div=document.getElementsByTagName('div')[0];

//var strong=document.getElementsByTagName('strong')[0];

var p=document.getElementsByTagName('p')[0];

var span=document.createElement('span');

Element.prototype.remove=function(){

var definedParent=this.parentNode;

definedParent.removeChild(this);

}

</script>

</body>

p.remove()

Undefined

22.将目标节点内部的节点顺序逆序。

eg:<div> <a></a> <em></em></div>

<div><em></em><a></a></div>

Element.prototype.reversed=function(){

var child=this.childNodes;

var len=child.length;

for(var i=len-1;i>=0;i--){

div.inserBefore(child[i],child[len]);

}

}

23.计时器，到一分钟停止

<style type="text/css">

input{

width: 50px;

height: 20px;

}

</style>

</head>

<body>

<input type="text" value="0">分

<input type="text" value="0">秒

<script type="text/javascript">

var minute=document.getElementsByTagName('input')[0],

second=document.getElementsByTagName('input')[1],

minuteCount=0,

secondCount=0,

timer;

timer=window.setInterval(function(){

secondCount++;

if(secondCount==60){

secondCount=0;

minuteCount++;

minute.value=minuteCount;

}

if(secondCount==0&&minuteCount==3){

clearInterval(timer);}

second.value=secondCount;

},100);

</script>

</body>

24封装兼容方法，求滚动论滚动距离getScrollOffset()

function getScrollOffset(){

if(window.pageXOffset){

return {

w:window.pageXOffset,

h:window.pageYOffset

}

}

var dis={

w:document.body.scrollLeft+document.documentElement.scrollLeft,

h:document.body.scrollTop+document.documentElement.scrollTop }

return dis；

}

getScrollOffset()

39

25.#封装兼容方法，返回浏览器视口尺寸

function getViewportOffset(){

if(window.innerWidth){

return{

w:window.innerWidth,

h:window.innerHeight

}

}

if(document.compatMode=="CSS1Compat"){

return{

w:document.documentElement.clientWidth,

h:document.documentElement.clientHeigth

}

}

}

26.封装元素宽和高

Element.prototype.getElementOffset=function(){

var objData=this.getBoundingClientRect();

if(objData.width){

return{

w:objData.width,

h:objData.height

}

}else{

return{

w:objData.right-objData.left,

h:objData.bottom-objData.top

}

}

}

27.#封装函数：元素相对于文档坐标getElementosition

Element.prototype.getElementPosition=function(){

var x=0,

y=0,

ele=this;

while(ele!=document.body){

x+=ele.offsetLeft;

y+=ele.offsetTop;

ele=ele.offsetparent;

}

return {

disX:x,

disX:y

}

}

28.封装兼容性方法getStyle(obj,prop);

function getStyle(obj,prop,null){

if(obj.currentStyle){

return obj.currentStyle[prop];

}else{

return window.getComputedStyle(obj,null)[prop];

}

}

getStyle(div,”width”) 100px;

29.拖拽

<style type="text/css">

.demo{

height: 100px;

width: 100px;

background-color: red;

position: absolute;

left:0;

top:0;

}

</style>

<script type="text/javascript" src="tools2.js"></script>

</head>

<body>

<div class="demo"></div>

<script type="text/javascript">

function getStyle(obj,prop){

if(obj.currentStyle){

return obj.currentStyle[prop];

}else{

return window.getComputedStyle(obj,null)[prop];

}

}

var div=document.getElementsByClassName('demo')[0],

disX=0,

disY=0;

div.addEventListener('mousedown',function(e){

disX=e.clientX-parseInt(getStyle(this,'left'));console.log("a");

disY=e.clientY-parseInt(getStyle(this,'top'));

document.addEventListener('mousemove',mouseMove,false);

div.addEventListener('mouseup',function(e){

document.removeEventListener('mousemove',mouseMove,false);

})

},false);

function mouseMove(e){

div.style.left=e.clientX-disX+"px";

div.style.top=e.clientY-disY+"px";

}

</script>

30：使用原生js.addEventListener,给每个li元素绑定一个click事件，输出它们的顺序

<ul>

<li></li>

<li></li>

<li></li>

</ul>

<ul>

<li>a</li>

<li>a</li>

<li>a</li>

<li>a</li>

</ul>

<script type="text/javascript">

var oLi=document.getElementsByTagName('li'),

len=oLi.length;

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

(function(j){

oLi[j].addEventListener('click',function(){

console.log(j);

},false);

}(i))

}

31.**绑定事件封装综合方法：**

function addEvent(elem,type,handle){

if(elem.addEventListener){

elem.addEventListener(type,handle,false);

}else if(elem.attachEvent){

elem['temp'+type+handle]=handle;

elem[type+handle]=function(){

elem['temp'+type + handle].call(elem);

}

elem.attachEvent('on' + type,elem[type+handle]);

}else{

elem['on'+type]=handle;

}

}

32.封装解除绑定函数

function removeEvent(elem,type,handle){

if(elem.removeEventListener){

elem.removeEventListener(type,handle,false);

}else if(elem.detachEvent){

elem.detachEvent('on'+type,handle);

}else{

elem['on'+type]=null;

}

}

33.封装取消冒泡的函数

function stopBubble(event){

if(event.stopPropagation){

event.stopPropagation();

}else{

event.cancelable=true;

}

}

34取消右键出菜单事件

document.oncontextmenu=function(){

return false;

}

35封装组织默认事件 cancelHandler

function cancelHandler(event){

if(event.preventDefault){

event.preventDefault();

}else if(event.returnValue){

event.returnValue=false;

}

}

36事件兼容性写法

document.addEventListener('click',function(e){

var event=e||window.event;

var target=event.target||event.srcElement;

},false);

37.拖拽木块

.wrapper{

height: 100px;

width: 100px;

background-color: red;

position: absolute;

left: 0;

top: 0;

}

</style>

<script type="text/javascript" src="tools2.js"></script>

</head>

<body>

<div class="wrapper">

</div>

<script type="text/javascript">

var div=document.getElementsByClassName('wrapper')[0];

function drag(elem){

var disX,

disY;

addEvent(elem,'mousedown',function(e){

var event=e||window.event;

disX=event.clientX-parseInt(getStyle(elem,'left'));

disY=event.clientY-parseInt(getStyle(elem,'top'));

addEvent(document,'mousemove',mouseMove);

addEvent(document,'mouseup',mouseUp);

stopBubble(e);

cancelHandler(e);

})

function mouseMove(e){

var event=e||window.event;

elem.style.left=event.clientX-disX+'px';

elem.style.top=event.clientY-disY+'px';

}

function mouseUp(e){

removeEvent(document,'mousemove',mouseMove);

removeEvent(document,'mouseup',arguments.callee);

}

}

drag(div);

</script>

38输入框

.fontColor{

color:#424242;

}

.fontNormalColor{

color:#999;

}

</style>

<script type="text/javascript" src="tools2.js"></script>

</head>

<body>

<input type="text" style="border:2px solid black;" class="fontNormalColor" value="请输入用户名" onfocus="if(this.value=='请输入用户名'){this.value='';this.className='fontColor'}"onblur="if(this.value==''){this.value='请输入用户名';this.className='fontNormalColor'}">

39.原题

请将形如aaaabbbb的字符串调换成bbbbaaaa的形式（提示：用正则做可给满分，其他方法酌情给分）

var str="aaaabbbb";

var reg=/(\w{4})(\w{4})/g;

console.log(str.replace(reg,"$2$1"));

Bbbbaaaa

40\*全部变化

var reg=/ab/g;

var str="abcdbcdbcd";

console.log(str.replace(/bc/g,"aa"));

aaadaadaad

41.ab-->ba

var reg=/(\w)(\w)/g;

var str="ab";

console.log(str.replace(reg,"$2$1"));

42,请将如下变量名转换为小驼峰式。

my-first-name-->myFirstName(提示：用正则做可得满分，其他方法)

toUpperCase()将当前字符转换成大写

var str="my-first-name";

var reg=/-(\w)/g;

str.replace(reg,function($,$1){

return $1.toUpperCase();

})

43.一串连续的数字，请实现打点功能，如100000000-->100.000.000

var str="100000000";

var reg=/(?=(\B)(\d{3})+$)/g;

console.log(str.match(reg));

console.log(str.replace(reg,"."));

44.匹配固定模式：

var reg=/(\w)(\w)\2\1/g;//反向引用

var str="abbacdda";

console.log(str.match(reg));

["abba"]

45封装classname

<body>

<div class=" demo "> </div>

<div class=" demo"> </div>

<div class="demo"> </div>

<script>

Document.prototype.getClassName = function(className){

var allEle = document.getElementsByTagName('\*'),

length=allEle.length,

retArr = [],

reg=/^\s+|\s+$/g;

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

if(allEle[i].className.replace(reg,"")== className){

retArr.push(allEle[i]);

}

}

return retArr;

}

document.getClassName('demo');

[<div class=​" demo ">​ ​</div>​, <div class=​" demo">​ ​</div>​, <div class=​"demo">​ ​</div>

46打乱数组：var arr=[1,3,5,6,8,9];

arr.sort(function (){

var num=Math.random();

return num-0.5;

});

document.write(arr);

47数组去重：哈希

//var arr=["a",3,4,"b",5,6,undefined];

var obj={

"a":"1",

"3":"1",

"4":"1",

"b":"1",

"5":"1",

"6":"1",

"undefined":"1"}

代码：var arr=["a","a",3,4,"b",5,6,"b",6,6,6,6,undefined,undefined];

Array.prototype.unique=function(){

var len=this.length,

arr=[],

obj={};

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

if(!obj[this[i]]){

obj[this[i]]="1";

arr.push(this[i]);

}

}

return arr;

}

document.write(arr.unique());

​48数组去重

Array.prototype.single=function(){

this.sort();

var a=[this[0]];

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

{

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

a.push(this[i]);

}

}

return a;

}

var arr=[1,'a',"a","b",1,0,"a","b"];

document.write(arr.single());