**中高级JavaScript易错面试题-6**

<https://www.bbsmax.com/A/kjdwqm1wJN/>

1、写一段脚本，实现观察者模式

**2、js输出当前访问页面的设备的类型**

function checkEquipment() {

var output = {};

if (navigator.userAgent.match(/(iPhone|iPod|iPad);?/i)) {

output['ios'] = true;

} else if (

navigator.userAgent.match(/android/i)) {

output['android'] = true;

} else if (navigator.userAgent.match(/windows/i)) {

output['windows'] = true;

} return output;

};

console.log(checkEquipment())//对应输出其中一个：{android: true} {ios: true} {windows: true}

3、写一段脚本，以你认为最优的方式向<body>中添加10个<p>Hello World</p>然后将所有的p标签删除，

**4、以您认为合理的方式，给下面HTML元素绑定click事件,使绑定事件之后添加的li依然可以触发事修的处理函数**

<ul><li>1</li><li>2</li><li>3</li><li>4</li><li>5</li></ul>

**错误方式：55555**

var lis = document.getElementsByTagName('li')

var len = lis.length;

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

$(lis[i]).click(function () {

alert(i);

//555 })

})

}

**正确方式一(闭包)01234**

var lis = document.getElementsByTagName('li');

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

(function (index) {

lis [i].onclick = function () {

alert(index);

}

})(i);

};

**正确方式二（利用jq中的each） 01234**

var lis = $("ul>li");

$.each(lis, function (index, element) {

$(element).click(function () {

alert(index);//索引

})

})

**5、编写一段脚本，去除以下数组中重复元素**

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

console.log(arr)

方法一：

Array.prototype.unique = function () {

var res = [];

for (var i = 0, newArr = []; i < this.length; i++) {

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

res.push(arr[i]);

newArr[this[i]] = 1;

}

}

return res;

}

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

console.log(arr.unique());//[1, 2, 3, 4]

方法二：

let arr = [1,2,3,'3',4,2]

let set = new Set(arr);

let newArr = Array.from(set);

console.log(newArr); // [1, 2, 3, "3", 4]

方法三：

Array.prototype.unique=function(){

var newArr=[];

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

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

newArr.push(this[i]);

}

}

return newArr;

}

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

console.log(arr.unique());//[1, 2, 3, "3", 4]

**6、请问下述代码最终结果是什么？** 打印结果：55555

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

setTimeout(function(){

console.log(i)//55555

},1000)

//console.log('i',i)//01234 这里能正常打印

}

原因分析：setTimeout()是一个异步处理函数，它会等待所有的主线程任务处理完，才开始执行自己的内部的任务，每隔1s往任务队列中添加一个任务【闭包函数，setTimeout()中的函数，现在还没执行】，当主线程执行完时，这时i=5，

才开始执行任务队列中的任务【闭包函数，setTimeout()中的函数开始执行，执行5次】。

for循环括号内的就是主线程，执行完时i是5，所以会打印出5次5；

如果想打印出0,1,2，3，4

解决方案：

1. 把var改为let，let是块极作用域，每次for循环都会把对应的i绑定到添加的任务【闭包函数，setTimeout()中的函数】中,所以当主线程执行完时，也不会影响到每个任务中i。所以可以打印出0   1   2 3 4
2. 把定时器放在一个自执行函数中用i当做参数

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

(function(i){

setTimeout(function(){

console.log(i);//01234

},1000)

})(i)

}

**7、填充/TODO：处代码，以达成克隆目标对象的目的**

var sourceObject ={

id:0,

name:'Alan',

scores:[1,2,3],

favorite:{

tag1:'chinese',

tag2:'math',

}

}

var destObject=clone(sourceObject);

function clone(obj) {

var type = typeof (obj);

if (type === "string" || type === "boolean" || type === "number") {

var newone = obj;

} else if (type === "object") {

if (obj === null) {

var newone = null;

} else if (Object.prototype.toString.call(obj).slice(8, -1) === "Array") {

var newone = [];

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

newone.push(clone(obj[i]));

}

} else if (Object.prototype.toString.call(obj).slice(8, -1) === "Object") {

var newone = {};

for (var i in obj) {

newone[i] = clone(obj[i]);

}

}

}

return newone;

}

其他原型复制方法通用

Object.prototype.clone = function () {

var o = (this.constructor === Array ? [] : {});//{}

for (var key in this) { //this ->{name: "johnny"}

o[key]=typeof this[key]==="object"?this[key].clone() : this[key];

}

return o;

}

var destObject = sourceObject.clone();//使用

**7、编写一段脚本，实现Function.bind函数相同的功能**

Function.prototype.bind = function (context) {

var self = this; // 保存原函数

return function () { // 返回一个新函数

return self.apply(context, arguments); // 执行新函数时，将传入的上下文context作为新函数的this

}

}

8、请用纯编写一个下拉菜单

**10、通过继承的方式达到输出结果123**

var AClass = function(){

this.id=123;

}

var BClass=function(id){

AClass.call(this,id)

}

var bInstance = new BClass;

BClass.prototype.sayId=function(){

console.log(this.id)

}

bInstance.sayId();

**11、为div 设置类a与b，应编写HTML代码?** <div class=”a b”>\_</div>\_\_

**12、设置CSS属性clear的值为( none)可清除左右两边浮动**

**13、css属性(margin)可为元素设置外补丁**

**14、设置文字不换行，超出部分用... 代替样式为：**

overflow: hidden; text-overflow: ellipsis; white-space: nowrap; width: 100px;

**15、定义内联（非块状）元素可以定义宽度和高度？**

block:将行级别元素转为块级别元素

附：

inline-block:不能自动换行,但是可以设置宽高

inline:将块级别元素转为行级别元素

**16、如何使得英文单词不发生词内断行？**

word-break: break-word;

**17、写出定义标题的方法，br标签在XHTML中语义是？** 换行

**18、不换行必须设置哪些？** white-space: nowrap;

**19、在使用table表现数据时，有时候表现出来的会比自己实际设置的宽度要宽，为些需要设置哪些属性值？**

cellpadding=”0″，cellspacing=”0″

20、display:none与visbility:hidden的区别是什么？

1.display:none是彻底消失，不在文档流中占位，浏览器也不会解析该元素；visibility:hidden是视觉上消失了，可以理解为透明度为0的效果，在文档流中占位，浏览器会解析该元素；

2.使用visibility:hidden比display:none性能上要好，display:none切换显示时visibility，页面产生回流（当页面中的一部分元素需要改变规模尺寸、布局、显示隐藏等，页面重新构建，此时就是回流。所有页面第一次加载时需要产生一次回流），而visibility切换是否显示时则不会引起回流。

**21：结果？ƒ a(){}**

(function (a) {

console.log(a); //ƒ a(){}

var a = 10;

function a() { }

})(100)

解释： a可以声明提前，赋值不提前，函数可以随时调取，这里a不是参数是变量了

函数在预解析的时候会被提到最前面，这个函数名和变量名重名，会显示函数

把 function a(){}改成 function b(){}就能显示100了

100没起作用是**局部变量优先规则**

(function (a) {

var a;

console.log(a)

a = 10;

function a() { }

})(100)

局部变量优先原则，原理同下：

var a = 5;

function fn() {

var a = 10;

console.log(a)　　// 10，局部变量优先，在局部找到a后，不会再向外查找

}

---------------------------------------------

结果：undefined

var a = 10;

(function () {

console.log(a);//undefined

var a = 100;

})();

解释：

声明会提升到function最开头，但赋值发生在最后上面的代码等价于：打印a的时候，a并没有在function内赋值，所以是undefined

va

(function () {

var a;

console.log(a);

a = 100;

})();

---------------------------------------------

**22、编写一个脚本，0到10完成10随机整数的排序**

function sortNumber(a, b) {

return a - b;//升序

//return b-a;//降序

}

var iArray = [];

function getRandom(iStart, iEnd) {

var iChoice = iStart - iEnd;

return Math.abs(Math.ceil(Math.random() \* iChoice)) + iStart; //ceil() 方法可对一个数进行上舍入。

}

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

iArray.push(getRandom(1, 10))

}

iArray.sort(sortNumber);

console.log(iArray)//随机生成  [1, 1, 4, 5, 6, 6, 6, 6, 7, 9]

**23、写出ul ol dl三种列表的html结构**

前两个都是li

|  |
| --- |
| <dl><dt>标题</dt><dd>内容1</dd></dl> |

**24、css代码优化**

margin-top:20px;marginright:5px;margin-left:5px;margin-bottom:20px;font-style:italic;

font-weight:bold;font-size:1em;line-height:140%;font-family:'lucida Grande',sans-serif;

color:#333333;

参考：

margin: 20px5px;

font: italicsmall-capsbold1em/140%'Lucida Grande', sans-serif;

color: #369;

**25、针对不同浏览器写出box-shadow的格式**

-webkit-box-shadow: 3px 3px 3px;

 -moz-box-shadow: 3px 3px 3px;

box-shadow: 3px 3px 3px;

**26、HTML有哪些新的页面元素？列给5个以上；**

Article、 header footer figure（一组媒体及标签文字） mark video section embed外部插件或对象

**27、你认为良好的CSS架构目标是什么？**

**28、AJAX是什么，有什么优点和缺点？**

优点：

页面局部刷新，提高用户体验度；

使用异步方式与服务器通信，具有更加迅速的响应能力；

减轻服务器负担；

基于标准化的并被广泛支持的技术，不需要下载插件或者小程序。

缺点：不支持浏览器 back 按钮；安全问题；对搜索引擎的支持比较弱。

AJAX干掉了Back和History功能，即对浏览器机制的破坏。无法回到前一个页面状态

Ajax就如同对企业数据建立了一个直接通道。使开发者不经意间暴露比以前更多的数据和服务器逻辑

对搜索引擎的支持比较弱。如果使用不当，AJAX会增大网络数据的流量，从而降低整个系统的性能。

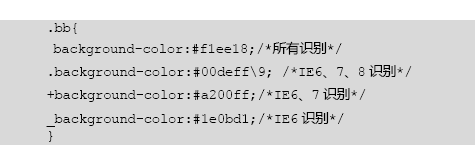
**29、模块加载器有何优点，以及常模块加载器有哪些**

require 加载模块。

module.exports 对外暴露接口。

一个文件就是一个模块，具备独立的作用域，不会污染全局。

1. **介绍你所知道的CSS hask技巧**



**31、列举常用WEB的性能优化方法**

网页内容：

减少HTTP请求次数、缓存AJAX、延迟加载、减少DOM元素数量、减少iframe数量、减少cookie大小

Css:将样式表置顶将、避免css表达式、用link代替@import、避免使用filters、css文件合并与压缩

Javascript:

将脚本置底（将脚本内容在页面信息内容加载后再加载）

使用外部javascript和css文件

去除重复脚本，避免重复的资源请求

减少DOM访问（修改和访问DOM元素会造成页面的重绘和重排，循环对DOM操作更是减慢页面加载速度）

js文件合并与压缩

图片方面：优化图片压缩、优化css Sprite

**32浏览器中数据存储有哪些方式？分别有什么区别？**

之前Cookie 实现，localStorage：适用于长期存储数据，浏览器关闭后数据不丢失, 数据只能由用户删除；sessionStorage：存储的数据在浏览器关闭后自动删除。

**33、多个页面之间如果进行通信，组件之间有哪些传值的方法？**

1、父组件往子组件传值props

2、子组件往父组件传值，通过emit事件

**3、vuex进行传值**

vuex主要是是做数据交互，父子组件传值可以很容易办到，但是兄弟组件间传值（兄弟组件下又有父子组件），或者大型spa单页面框架项目，页面多并且一层嵌套一层的传值，异常麻烦，用vuex来维护共有的状态或数据会显得得心应手。

4、provide/inject：**主要解决了跨级组件间的通信问题，不过它的使用场景，主要是子组件获取上级组件的状态，跨级组件间建立了一种主动提供与依赖注入的关系。**

5、$attrs/$listeners

$attrs：包含了父作用域中不被 prop 所识别 (且获取) 的特性绑定 (class 和 style 除外)。当一个组件没有声明任何 prop 时，这里会包含所有父作用域的绑定 (class 和 style 除外)，并且可以通过 v-bind="$attrs" 传入内部组件。通常配合 interitAttrs 选项一起使用。

$listeners：包含了父作用域中的 (不含 .native 修饰器的) v-on 事件监听器。它可以通过 v-on="$listeners" 传入内部组件

**34、为什么会出现跨域的问题以及解决方式？**

**35、写出以下执行结果，谈谈上下文和作用域的区别？**

结果：

~undefined

this:1

return1:2

return2:3

setTimeout:3

var a = 1;

function test() {

console.log('~' + a); //声明提前

var a = 2;

console.log('this:' + this.a) //全局

setTimeout(function () {

console.log('setTimeout:' + 3)

}, 0)

return function () {

console.log('return1:' + a)

a = 3

console.log('return2:' + a)

}

}

test()();

-----------------------------------------

**36、写下结果：说说Event Loop Task Queue? 23541**

Promise定义之后便会立即执行,其次主任务，其后的.then() 然后setTimeout

Promise比setTimeout()先执行

执行顺序 Promise-> 全局 ->then->setTimeout

setTimeout(function () {

console.log(1)

}, 0)

new Promise(function executor(resolve) {

console.log(2)

for (var i = 0; i < 10000; i++) {

i == 9999 && resolve();

}

console.log(3)

}).then(function () {

console.log(4)

})

console.log(5)

**37、Vue中给data中的对象属性添加一个新的属性时会发生什么，如何解决？**

解：点击button会发现， obj.b 已经成功添加，但是视图并未刷新：

因为vue底层的东西本身就不支持对数组某一个特定元素的监控，push，pop这些都能监控到。

对json或数组的添加和删除是监听不到的

不是VUE的bug，是VUE依赖底层库的问题，

vue依赖机制observe --数据监听

并不是所有的数据操作都能被监听到：

这时就需要使用Vue的全局api—— $set()：

vue实例.$set(数据，key，val)

Vue.set(数据，key，val) // 直接在Vue类上

addObjB() {

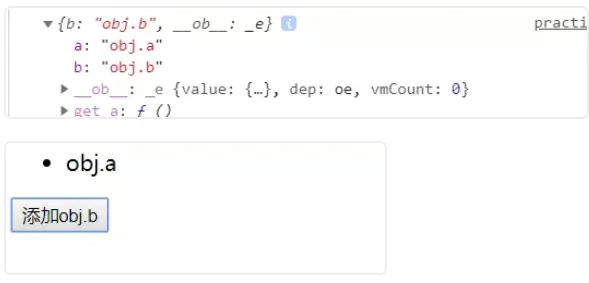
// this.obj.b = 'obj.b' 下面两种写法

this.$set(this.obj, 'b', 'obj.b')

Vue.set(this.obj, 'b', 'obj.b')

console.log(this.obj)

}



<div id='app'>

<ul><li v-for="value in obj" :key="value">{{value}}</li> </ul>

<button @click="addObjB">添加obj.b</button>

</div>

<script src='vue2.js'></script>

<script type="text/javascript">

var vm = new Vue({

el: '#app',

data() {

obj: {

a: 'obj.a'

}

},

methods: {

addObjB() {

this.obj.b = 'obj.b'

console.log(this.obj)

}

}

})

</script>

---------------------------

**38、Vue路由的钩子函数有哪些，并列举实际应用**

守卫作用：过来新的地址要先经过守卫同意了才能进去 也就是next()

分类：全局和局部

路由钩子函数有三种：

  1：全局钩子： beforeEach、afterEach

  2：单个路由里面的钩子：beforeEnter、beforeLeave

  3：局部组件路由：beforeRouteEnter、beforeRouteUpdate、beforeRouteLeave

**39、call apply bind函数存在的区别**

**相似之处**

1. 都是用来改变函数的this对象的指向的
2. 第一个参数都是this要指向的对象
3. 都可以利用后续参数传参

**区别**

1. **call和apply都是对函数的直接调用（也叫直接执行函数）**，而**bind方法返回的仍然是一个函数**，因此后面还需要()来进行调用才可以（将上下文绑定到bind()括号中的参数上，然后将它返回）。所以，bind后函数不会执行，而只是**返回一个改变了上下文的函数副本**。
2. call和apply都可以传参数。call后面的参数与fn方法中是一一对应的，而**apply的第二个参数是一个数组**，数组中的元素是和fn方法中一一对应的，这就是两者最大的区别。
3. bind是ES5中的方法，可以向call一样传参，也可以在调用的时候再进行传参。

**总结**

1. 当我们使用一个函数需要改变this指向的时候才会用到call apply bind
2. 如果你要传递的参数不多，则可以使用fn.call(thisObj, arg1, arg2 ...)
3. 如果你要传递的参数很多，则可以用数组将参数整理好调用fn.apply(thisObj, [arg1, arg2 ...])
4. 如果你想**生成一个新的函数长期绑定某个函数给某个对象使用**，则可以使用const newFn = fn.bind(thisObj); newFn(arg1, arg2...)

**40、判断真假:**

结果 false false true false

因为：typeof null ->Object

typeof NaN ->number

null instanceof object null === undefined null==undefined NaN==NaN

**41、foo对象有att属性，那么获取att怪属性的值，以下哪些做法对**

getAttribute() 和 attr() 都是获取元素属性的方法前者js写法后者jquery 写法

这里不能用

正确写法：

foo.att、foo['att']

错误写法：

foo('att')、foo{'att'}、foo['a','t','t']、foo.getAttribute('att')、foo.attr('att')

**42、以下哪个方法用于Array中删除元素，并向数组添加新元素**－＞splice

splice push shift slice

**43、解释一下为什么需要清除浮动？清除浮动的方式**

浮动定位是指将元素排除在普通之外，并且将它放置在包含框的左边或右边，但是依旧位于包含框之内，也就是说，浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。

由于浮动框不在文档的普通流中，所以元素浮动之后，其原有位置不在保留，其他元素的位置会受到影响。

如果需要清除左侧或右侧浮动元素带来的影响，刚可以使用clear属性来设置。另外，包含框内的子元素浮动后，如果包含框没有设置具体高度，则其高度会发生变化，此时可以使用overflow 属性来清除子玩素浮动后带来的影响。

**44、编写程序，请用javascript编写获取url中的参数值,要求至少写出两种实现方式**

//方法1

url =location.href;// var url = 'http://baidu.com?a=1&b=55';

var theRequest = {};

if (url.indexOf("?") != -1) {

    var str = url.substr(url.indexOf("?")+1);//a=1&b=55

    strs = str.split("&");

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

        theRequest[strs[i].split("=")[0]] = strs[i].split("=")[1];

    }

    console.log(theRequest) //{a: "1", b: "55"}

}

//方法2

function GetRequest() {

    var url = location.search; // search 取"?"符后的字串 ?a=2&b=33

    var theRequest = {};

    if (url.indexOf("?") != -1) {

        var str = url.substr(1);//a=2&b=33

        strs = str.split("&");//["a=2", "b=33"]

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

            theRequest[strs[i].split("=")[0]] = strs[i].split("=")[1];

        }

    }

    return theRequest;

}

Request = GetRequest();

console.log(Request)//{a: "2", b: "33"}

//方法3

// url?a=1&b=2

function getQueryString(name) {

var reg = new RegExp('(^|&)' + name + '=([^&]\*)(&|$)', 'i');

var r = location.search.substr(1).match(reg);

    if (r != null) {

         return unescape(r[2]);

    }

    return null;

 }

console.log(getQueryString('a'))

**45、ＥＳ５的继承和ＥＳ６的继承有什么区别？**

ES5的继承时通过prototype或构造函数机制来实现。ES5的继承实质上是先创建子类的实例对象，然后再将父类的方法添加到this上（Parent.apply(this)）。

ES6的继承机制完全不同，实质上是先创建父类的实例对象this（所以必须先调用父类的super()方法），然后再用子类的构造函数修改this。

具体的：ES6通过class关键字定义类，里面有构造方法，类之间通过extends关键字实现继承。子类必须在constructor方法中调用super方法，否则新建实例报错。因为子类没有自己的this对象，而是继承了父类的this对象，然后对其进行加工。如果不调用super方法，子类得不到this对象。

**46、在金融应用产品中，数值常常使用千分位分隔，请使用js实现一个具有此功能的简单常数函数**

function toNum(value) {

if (!value) return ' '

var intPart = Number(value).toFixed(0) // 获取整数部分

var intPartFormat = intPart.toString().replace(/(\d)(?=(?:\d{3})+$)/g, '$1,') // 将整数部分逢三一断

return intPartFormat;

}

var n = 12001

console.log( toNum(n))//12,001 12001.6->12,002

47、webpack用来解决什么问题

前端本身不支持像后端那样文件引用，使用webpack就可以实现这种功能。另外打包还会对代码做检查和压缩，达到优化的目的。

使用webpack过程：

1. 生成Packjson.json 执行命令 npm init –y；

文件作用：是node的项目描述文件，如项目依赖谁有哪些scripts 常用

1. 手动创建webpack.config.js

48、ｎｕｌｌ和ｕｎｄｅｆｉｎｅｄ的区别

null 是javascript关键字，空值，typeof返回object,可以表示数字，字符串和对象“无值”

undefined是预定义的全局变量，为“未定义”，变量一种取值，表示没有初始化。

当查询对象属性，数组元素值时，返回undefined时表示属性或元素不存在；

如果函数没返回值也返回undefined

**49、跟后端进行异步请求时，如何避免代码嵌套太深**

现在出现了一种比较好的写法， 就是用Promise.js来简单来写

var asny = function (text) {

var promise = new Promise(function (resolve, reject) {

setTimeout(function () {

console.log(text);

resolve();

}, 1000)

});

return promise;

}

asny("1").then(function () {

return asny("2");

}).then(function () {

return asny("3");

}).then(function () {

console.log("done");

});

**50、http协议中３０２状态是什么含义**

该[资源](https://baike.baidu.com/item/%E8%B5%84%E6%BA%90/9089683" \t "_blank)原本确实存在，但已经被**临时**改变了位置,换而言之，就是请求的资源暂时驻留在不同的URI下

**51、new操作符具体是干什么的**

（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 。

**52、如何准确的判断这个变量是否为数组**

**只有instanceof才能判断一个对象是否是真正的数组**

**53、输出结果 :NaN undefinedhello**

var a = 10

a.pro = 10

console.log(a.pro + a)//NaN

var s = 'hello';

s.pro = 'world';

console.log(s.pro + s)//undefined hello

解释：

变量a与s都是基本类型，无法给他们添加属性，所以a.pro和s.pro都是undefined。

undefined + 10 得到NaN（not a number）。

undefined + 'hello' 得到undefinedhello，其中undefined被转化为字符串类型。

-------------------------------------------

**54、同步和异步的区别**

异步操作：同时进行多个操作，用户体验好（如用户名检查，输入完就检查）. 异步缺点： 好用但写起来麻烦。

同步操作：一次只能进行一次操作，用户体验不好，按顺序执行,优点：清晰，

**55、浏览器输入URL到页面在用户面前，这上过程发生了什么**

1. 在浏览器中输入网址；

2. 发送至 DNS 服务器并获得域名对应的 WEB 服务器的 IP 地址；

3. 与 WEB 服务器建立 TCP 连接；

4. 浏览器向 WEB 服务器的 IP 地址发送相应的 HTTP 请求；

5. WEB 服务器响应请求并返回指定 URL 的数据，或错误信息，如果设定重定向，则重

定向到新的 URL 地址。

6. 浏览器下载数据后解析 HTML 源文件，解析的过程中实现对页面的排版，解析完成

后在浏览器中显示基础页面。

7. 分析页面中的超链接并显示在当前页面，重复以上过程直至无超链接需要发送，完

成全部显示。

**57、返回结果**

Typeof 100 返回number

Typeof(undefined)返回undefined

Typeof NaN返回number

Typeof(null)返回object

**58、类型检测方法typeof和 instanceof有什么区别**

typeof: 返回值是一个字符串， 用来说明变量的数据类型。 一般只能返回如下几个结果： number, boolean, string, function, object, undefined。

instanceof: 返回值为布尔值;用于判断一个变量是否属于某个对象的实例。

instanceof **运算符**用于检测构造函数的 prototype 属性是否出现在某个实例对象的原型链上。

**59、为什么要对URL进行编码**

是因为Url中有些字符会引起歧义，Url的编码格式采用的是ASCII码

encodeURIComponent编码的字符范围要比encodeURI的大，encodeURIComponent对特殊字符编码解决

用ASCII吗表示为:**3D:= 26:&**现在有这样一个问题，如果我的参数值中就包含=或&这种特殊字符的时候该怎么办。比如说“name1=value1”,其中value1的值是“va&lu=e1”二手字符串，那么实际在传输过程中就会变成这样“name1=va&lu=e1”。我们的本意是就只有一个键值对，但是服务端会解析成两个键值对，这样就产生了奇异。URL编码只是简单的在特殊字符的各个字节前加上%，例如，我们对上述会产生奇异的字符进行URL编码后结果：“name1=va%26lu%3D”，这样服务端会把紧跟在“%”后的字节当成普通的字节，就是不会把它当成各个参数或键值对的分隔符。

console.log(encodeURIComponent("va&lu=e1"))// va%26lu%3De1

**60、vue生命周期：**

beforeCreate 组件实例刚刚被创建,属性都没有

created 实例已经创建完成，属性已经绑定

beforeMount 模板编译之前（准备）

mounted 模板编译之后，代替之前ready \*

beforeUpdate 组件更新之前 data数据变了 (用在$.watch('a',function){})

updated 组件更新完毕 \* (用在$.watch('a',function){})

beforeDestroy 组件销毁前

destroyed 组件销毁后

Vue钩子函数：created mounted updated Destroy

**61、jsonp原理是什么**

1. 其背后原理就是利用了script标签不受同源策略的限制，在页面中动态插入了script，script标签的src属性就是后端api接口的地址,并且以get的方式将前端回调处理函数名称告诉后端，后端在响应请求时会将回调返还，并且将数据以参数的形式传递回去。

**62. JavaScript有哪几种数据类型**

参考答案：

简单：Number，Boolean，String，Null，Undefined

引用：Object，Array，Function

**65、css选择器优先级？**

！important ID选择器 class选择器 标签选择器 伪类选择器

**63. documen.write和 innerHTML的区别**

参考答案：

document.write只能重绘整个页面

innerHTML可以重绘页面的一部分

**64、谈谈以前端角度出发做好SEO需要考虑什么。**

参考答案：

1、了解搜索引擎如何抓取网页和如何索引网页

你需要知道一些搜索引擎的基本工作原理，各个搜索引擎之间的区别，搜索机器人（SE robot 或叫 web crawler）如何进行工作，搜索引擎如何

对搜索结果进行排序等等。

2、Meta标签优化

主要包括主题（Title)，网站描述(Description)，和关键词（Keywords）。还有一些其它的隐藏文字比如Author（作者），Category（目录），

Language（编码语种）等。

3、如何选取关键词并在网页中放置关键词

搜索就得用关键词。关键词分析和选择是SEO最重要的工作之一。首先要给网站确定主关键词（一般在5个上下），然后针对这些关键词进行优

化，包括关键词密度（Density），相关度（Relavancy），突出性（Prominency）等等。

4、了解主要的搜索引擎

虽然搜索引擎有很多，但是对网站流量起决定作用的就那么几个。比如英文的主要有Google，Yahoo，Bing等；中文的有百度，搜狗，有道等。

不同的搜索引擎对页面的抓取和索引、排序的规则都不一样。还要了解各搜索门户和搜索引擎之间的关系，比如AOL网页搜索用的是Google的搜索技

术，MSN用的是Bing的技术。

5、主要的互联网目录

Open Directory自身不是搜索引擎，而是一个大型的网站目录，他和搜索引擎的主要区别是网站内容的收集方式不同。目录是人工编辑的，主要

收录网站主页；搜索引擎是自动收集的，除了主页外还抓取大量的内容页面。

6、按点击付费的搜索引擎

搜索引擎也需要生存，随着互联网商务的越来越成熟，收费的搜索引擎也开始大行其道。最典型的有Overture和百度，当然也包括Google的广告

项目Google Adwords。越来越多的人通过搜索引擎的点击广告来定位商业网站，这里面也大有优化和排名的学问，你得学会用最少的广告投入获得最

多的点击。

7、搜索引擎登录

网站做完了以后，别躺在那里等着客人从天而降。要让别人找到你，最简单的办法就是将网站提交（submit）到搜索引擎。如果你的是商业网

站，主要的搜索引擎和目录都会要求你付费来获得收录（比如Yahoo要299美元），但是好消息是（至少到目前为止）最大的搜索引擎Google目前还

是免费，而且它主宰着60％以上的搜索市场。

8、链接交换和链接广泛度（Link Popularity）

网页内容都是以超文本（Hypertext）的方式来互相链接的，网站之间也是如此。除了搜索引擎以外，人们也每天通过不同网站之间的链接来

Surfing（“冲浪”）。其它网站到你的网站的链接越多，你也就会获得更多的访问量。更重要的是，你的网站的外部链接数越多，会被搜索引擎认为

它的重要性越大，从而给你更高的排名。

9、标签的合理使用

--------------------------------

**65、行内元素有哪些？块级元素有哪些？**

参考答案：

行内元素主要有：<span>、<a>、<b>、<img>、<br>、<button>、<strong>、<textarea>、<select>

块级元素主要有：<div>、<ul>、<li>、<p>、<fieldset>、<form>、<h1>、<h2>、<h3>、<h4>、<h5>、<h6>、<hr>、

<iframe>、<ol>、<pre>、<table>、<tr>、<td>

行内元素可以通过display:block转为块级元素。另外块级元素的margin和padding都正常，行内元素左右maring和左右padding正常，上下不识别，

也就是说不能通过margin-top和padding-top来改变行高

**66. 如何规避javascript多人开发函数重名问题**

参考答案：

(1) 可以开发前规定命名规范，根据不同开发人员开发的功能在函数前加前缀

(2) 将每个开发人员的函数封装到类中，调用的时候就调用类的函数，即使函数重名只要类名不重复就ok

**67、javascript面向对象中继承实现**

参考答案：

javascript面向对象中的继承实现一般都使用到了构造函数和Prototype原型链，简单的代码如下：

function Animal(name) {

this.name = name;

}

Animal.prototype.getName = function () {

alert(this.name)

}

function Dog() { };

Dog.prototype = new Animal("Buddy");

Dog.prototype.constructor = Dog;

var dog = new Dog();

dog.getName();//Buddy

**68、js的基础对象有那些, window和document的常用的方法和属性列出来**

参考答案：

String,Number,Boolean,Date,Math,Array,RegExp

Window:

方法：setInterval,setTimeout,clearInterval,clearTimeout,alert,confirm,open

属性：name,parent,screenLeft,screenTop,self,top,status

Document：

方法：createElement,execCommand,getElementById,getElementsByName,getElementByTagName,write,writeln

属性：cookie,doctype,domain,documentElement,readyState,URL

**69、 js中如何定义class,如何扩展prototype?**

参考答案：

Ele.className = “\*\*\*”; //\*\*\*在css中定义，形式如下：.\*\*\* {…}

A.prototype.B = C;

A是某个构造函数的名字

B是这个构造函数的属性

C是想要定义的属性的值

如何添加html元素的事件,有几种方法.

（1） 为HTML元素的事件属性赋值

（2） 在JS中使用ele.on\*\*\* = function() {…}

（3） 使用DOM2的添加事件的方法 addEventListener或attachEvent

**70、写一段js，判断是否是这样组成的：第一个必须是字母，后面可以是字母、数字、下划线，总长度为5-20**

参考答案：

var reg = /^[a-zA-Z][a-zA-Z\_0-9]{4,19}$/;

reg.test("a1a\_\_a1a\_\_a1a\_\_a1a\_\_");

**71、如何保持浮屠水平垂直居中**

（1）用 position和负边距：父节点相对定位。子节position:absolute;top:50%;margin-身高度的一半;（2）多行文本居中： vertical-align: middle;display: table-cell;（3）文本居中line-height

**Display:flex弹性布局”, 用来为盒子模型提供最大的灵活性;**

CSS水平垂直居中常见方法总结

1、文本水平居中

line-height,text-align:center(文字)

元素水平居中 margin:0 auo

方案1：position 元素已知宽度

父元素设置为：position: relative;

子元素设置为：

position: absolute; left: 50%;top: 50%;margin: -50px 0 0 -50px;

margin各减去上下距离的一半

方案2：position transform 元素未知宽度

子元素：margin: -50px 0 0 -50px;替换为：transform: translate(-50%,-50%);

方案3：flex布局

父元素加：

display: flex; //flex布局

justify-content: center; //使子项目水平居中

align-items: center; //使子项目垂直居中

------------------------------------------

运行结果：

1、结果：x值是3,到case1后 x是2在执行 case2

var x=0;

switch(++x){

case 0:

++x;

case 1:

++x;

case 2:

++x;

}

-------------------------------

2、结果： test is not a function

var test=1

function test(index){

console.log(index)

index=3

}

test(2) //test is not a function

解释：var 和 function同名并不冲突，但在部分浏览器中，var会遮蔽同时的function导致function失效，只是test被赋值就会not a function，如var test只声明就会显示2

结果：1201

此题的关键在于明白var result = test() 与var result2 = test()此处分别产生了两个闭包，它们之间是独立，互不干扰的

闭包的产生：**在A函数中定义了B函数，B函数中引用了A函数中的变量，此时会产生闭包**

function addCount() {

var n = 0

function add() {

n++

console.log(n)

}

return { n: n, add: add }

}

var newObj = addCount();

var newObj2 = addCount();

newObj.add()

newObj.add()

console.log(newOj.n)

newObj2.add()

---------------------

3、运行test()和new test()的结果是什么

var a = 5;

a = 0;

alert(a);

alert(this.a);

var a;

alert(a);

function test() {}

参考答案：

test(): 0 5 0

new test(): 0 undefined 0

---------------------

4、参考答案：

alert(typeof(null)) -> object

alert(typeof(undefined)) -> undefined

alert(typeof(NaN)) -> number

alert(NaN==undefined) -> false

alert(NaN==NaN) -> false

var str="123abc";alert(typeof(str++)) -> number

alert(str) -> NaN

---------------------

**5、结果：10**

if('a' in window) {

var a = 10;

}

console.log(a);　　// 10

解释：**变量提升、window的变量**

首先，if(){}的花括号并不像function(){}的花括号一样，具有自己的块级作用域，if的花括号还是全局的环境。根据JavaScript的变量提升机制，var a会被js引擎解释到第一行，如下：

var a;

if ('a' in window) {

a = 10;

}}

接着有个知识点，**全局变量是window对象的属性**，所以'a'  in  window会返回true，答案就很直白了。

这道题我在做的时候踩了个坑，我在代码编辑器里写了如下代码：

window.onload = function(){

if('a' in window){

　　　　var a = 10;

}

console.log(a)　　// undefined

这时候，a这个变量是定义在匿名函数function(){}里的，属于该函数的局部变量，所以a不再是window的对象。大家一定要注意细节。}

---------------------

6、结果：hello 666666 888888 world!

async function sayHello() {

console.log('Hello')

await sleep(1000)

console.log('world!')

}

function sleep(ms) {

　return new Promise(resolve => {

　　console.log("666666");

　　setTimeout(resolve, ms);

　　console.log("888888")})

}

sayHello()　　// hello 666666 888888 world!

解释：

async =true ; 就表示异步的

async 表示这是一个async函数，await只能用在这个函数里面。

await 表示在这里等待promise返回结果了，再继续执行。

首先打出hello，到了await，会等待promise的返回，所以“world”不会立刻打出，接着进入sleep函数，打出666，接着开了一个1秒的定时器，虽然js是单线程的，但setTimeout是异步的，在浏览器中，异步操作都是被加入到一个称为“events loop”队列的地方，浏览器只会在所有同步代码执行完成之后采取循环读取的方式执行这里面的代码，所以resolve被加入任务队列，先打印了888，一秒后执行了resolve，表示promise成功返回，打出了world。

---------------------

7、上方的函数作用域中a被重新赋值，未被重新声明，且位于console之下，所以输出全局作用域中的a。

var a = 1;

function test3() {

console.log(a); // 1

a = 2;

}

test3();

---------------------

8、上方函数作用域中使用了ES6的let重新声明了变量b，而let不同于var其不存在变量提升的功能，所以输出报错“b is not defined”。

let b = 1;

function test4() {

console.log(b); // b is not defined

let b = 2;//不加这就是1

}

test4();

---------------------

9、上方的函数作用域中用let声明了a为1，并在块级作用域中声明了a为2，因为console并不在函数内的块级作用域中，所以输出1。

function test5() {

let a = 1;

{

let a = 2;

}

console.log(a); // 1

}

test5();

---------------------

10、上方利用typeof比较数组和对象，因为typeof获取NULL、数组、对象的类型都为object，所以console为true。

var arr = [],

arr2 = {};

console.log(typeof (arr) === typeof (arr2)); // true

1. 上方利用instanceof判断一个变量是否属于某个对象的实例，因为在JavaScript中数组也是对象的一种，所以两个console都为true。

var arr = [];

console.log(arr instanceof Object); // true

console.log(arr instanceof Array); // true

---------------------

### **this指向**

12、This作用对象方法被调用指向对象

var obj = {

name: 'xiaoming',

getName: function () {

return this.name

}

};

console.log(obj.getName()); //xiaoming

nameFn()作为函数调用，obj.getName()这是作用方法调用

var obj = {

myName: 'xiaoming',

getName: function () {

console.log(this)

return this.myName

}

};

var nameFn = obj.getName;

console.log(nameFn()); // undefined

nameFn()作为函数调用，加apply后this指向obj2对象

var obj = {

myName: 'xiaoming',

getName: function () {

return this.myName

}

};

var obj2 = {

myName: 'xiaohua'

};

var nameFn = obj.getName;

console.log(nameFn.apply(obj2)); // 'xiaohua'

### **函数参数**

**Arr.slice(start,end)下标开始，结束（不包含该元素）**

var arr = new Array(6)

arr[0] = "George"

arr[1] = "John"

arr[2] = "Thomas"

arr[3] = "James"

arr[4] = "Adrew"

arr[5] = "Martin"

document.write(arr.slice(2, 4)) //Thomas,James

**5. 闭包问题**

var elem = document.getElementsByTagName('li'); // 如果页面上有5个li

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

elem[i].onclick = function () {

alert(i); // 总是5

};

}

// 上方是一个很常见闭包问题， 点击任何div弹出的值总是5， 因为当你触发点击事件的时候i的值早已是5，

可以用下面方式解决：在绑定点击事件外部封装一个立即执行函数， 并将i传入该函数即可。

var elem = document.getElementsByTagName('li'); // 如果页面上有5个li

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

(function (w) {

elem[w].onclick = function () {

alert(w); // 依次为0,1,2,3,4

}

})(i);

}

**Object.assign()方法进行对象的深拷贝可不改变源对象，返回目标对象**

Object.assign方法用于对象的合并，将源对象（source）的所有可枚举属性，复制到目标对象（target）。

Object.assign方法的第一个参数是目标对象，后面的参数都是源对象。

var obj2 = {

name: 'xiaoming',

age: 23

};

var newObj2 = Object.assign({}, obj2, {

color: 'blue'

});

newObj2.name = 'xiaohua';

console.log(obj2.name); // 'xiaoming'

console.log(newObj2.name); // 'xiaohua'

console.log(newObj2.color); // 'blue'

console.log(newObj2) // {name: "xiaohua", age: 23, color: "blue"}

console.log(obj2) //{name: "xiaoming", age: 23}

我们也可以使用Object.create()方法进行对象的拷贝，Object.create()方法可以创建一个具有指定原型对象和属性的新对象。

var obj3 = {

name: 'xiaoming',

age: 23

};

var newObj3 = Object.create(obj3);

newObj3.name = 'xiaohua';

console.log(obj3.name); // 'xiaoming'

console.log(newObj3.name); // 'xiaohua'

**简要介绍 Vuex 原理**

Vuex 实现了一个单向数据流，在全局拥有一个 State 存放数据，当组件要更改 State 中的数据时，必须通过 Mutation 进行，Mutation 同时提供了订阅者模式供外部插件调用获取 State 数据的更新。而当所有异步操作(常见于调用后端接口异步获取更新数据)或批量的同步操作需要走 Action，但 Action 也是无法直接修改 State 的，还是需要通过 Mutation 来修改 State 的数据。最后，根据 State 的变化，渲染到视图上。

3、js对url进行编码和解码的方法有哪些

encodeURI decodeURI