# 第三阶段面试题

## 一、JavaScript高级

### 1. 判断以下程序的输出结果:

var age=100;

function test(){

this.age=50;

return function(){

return this.age;

}

}

var m=new test();

alert(m());

var n=test();

alert(n());

**答案：**

100 50

1. 构造函数一旦返回一个对象，就不再创建新对象
2. m获得的是function(){ return this.age; }
3. n=test()，this指向window。先将全局变量age变为50,又返回一个函数function(){ return this.age; }保存在变量n中
4. 调用n时，this指向window。

### 2. 判断以下程序的输出结果:

var name="The Window";

var obj={

name:"My obj",

getName:function(){

return function(){

return this.name;

}

}

};

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

**答案：**

the window

1. obj.getName() 返回一个函数对象function(){ return this.name; }
2. (function(){ return this.name; }()) 相当于匿名函数自调，this指向window

### 3. 判断以下程序的输出结果:

var length=10;

function fn(){

console.log(this.length);

}

var obj={

length:5,

method:function(fn){

fn();

arguments[0]();

}

};

obj.method(fn,1)

**答案：**

10 2

1. fn() this指向window，所以输出10
2. arguments[0]() 属于特殊情况，this->arguments,相当于arguments.0(), 所以，this指向arguments。所以length输出的是obj.method()的参数个数，为2.

### 4. 统计一个字符串中出现次数最多的字符是? 共出现多少次

**答案：**

var dict={};

var c="", max=1;

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

var char=str[i];

if(dict[char]===undefined)

dict[char]=1;

else{

dict[char]+=1;

if(dict[char]>max){

max=dict[char];

c=char;

}

}

}

console.log(c,max);

### 5. 判断以下程序的输出结果:

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

setTimeout(function(){

console.log(i);

},0)

}

console.log(i);

**答案：**

5 5 5 5 5

1. 函数定义时，函数内容是不执行的，所以i还是i，不会变成0，1，2，3，4
2. 定时器中的回调函数只能再主程序执行完才能开始执行
3. 当主程序执行完，循环变量i，已经被改为5了。

### 6. 判断以下程序的输出结果:

window.color="red";

let color="green";

let obj={

color:"blue"

};

let sayColor=()=>{

return this.color;

}

console.log(sayColor.apply(obj));

let a=10;

console.log(window.a);

**答案：**

red undefined

1. let相当于匿名函数自调，所以，let声明的变量，不会自动加入到window
2. 剪头函数内外this通用，所以apply也无法替换sayColor函数内的this，所以this指向window，所以输出red

### 7. 判断以下程序的输出结果:

var c=1;

function c(c){

console.log(c);

var c=3;

}

c(2);

**答案：**

报错: TypeError: c不是一个函数

1. function c(c){} 整体被声明提前，后又被c=1代替。所以，c最后不是一个函数，而是数字1

### 8. 判断以下程序的输出结果:

function change(){

alert(typeof fn)

function fn(){ alert('hello') }

var fn;

}

change();

**答案：**

function

1. function fn(){…}被整体声明提前了
2. var fn发现已经有fn变量了，就不再重复创建，所以，var fn没作用。

### 9. 判断以下程序的输出结果:

a=3

a.prop=4; prop是从属性对象中取值

alert(a+a.prop)

**答案：**

NaN

1. a.prop=4，等效于new Number(a).prop=4, 但是new Number(a)，使用后自动释放，4也不存在了
2. 再次使用a.prop，又等效于新的new Number(a)，所以没有prop属性，值为undefined。
3. 数字+undefined, undefined隐式转为数字NaN，导致计算结果为NaN

### 10. 判断以下程序的输出结果:

var o={

a:10,

b:{

a:12,

fn:function(){

var a=13;

console.log(this.a);

}

}

}

o.b.fn();

**答案：**

12

1. this指.前的o.b对象，所以a为12

### 11. HTML事件的绑定方式有哪几种，如何绑定？

**答案：**

1. 在DOM元素中直接绑定,

例如，鼠标单击事件 onclick ，鼠标双击事件 ondouble，鼠标移入事件 onmouseover，鼠标移出事件 onmouseout 等

1. 在JavaScript代码中绑定

例如，elementObject.onXXX=function(){  
    // 事件处理代码  
}

1. 绑定事件监听函数

例如，onreadystatechange=function(){

//事件处理代码

}

### 12. 实现数组去重（元素js）？

**答案:**

实现思路：双层循环，外循环表示从0到arr.length，内循环表示从i+1到arr.length

将没重复的右边值放入新数组。（检测到有重复值时终止当前循环同时进入外层循环的下一轮判断）

function unique4(arr){

var hash=[];

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

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

if(arr[i]===arr[j]){ ++i;

}

}

hash.push(arr[i]);

}

return hash;

}

### 13.平时用过ES6哪些特性，体验如何，与ES5有什么不同？

**答案**：Let和const关键字，变量的解构赋值，字符串、数值的扩展，数组、对象的扩展，函数的扩展，for...of

与ES5的区别：

1、定义变量

ES5中用：var / function

ES6中用：let / const / class / import ...

没有变量提升

同一个作用域中不可重复声明

不会给window增加全局属性

会形成块级作用域

const设置的变量值是不可修改的（理解为常量）

2、解构赋值

构建一个和变量值相同结构的结构，快速获取对象或者数组中的某一部分内容

3、箭头函数

4、对象和数组中新增加一些属性和方法（正则和字符串中也新增很多）

dir(Array);//=>Array.xxx()

dir(Array.prototype);//=>给实例用的 [].xxx()

dir(Object);//=>Object.xxx()

dir(Object.prototype);//=>({}).xxx()

5、模板字符串 `xxx${JS CODE}...`

6、class / extends

### 14. 闭包是什么，有什么特性，对页面有什么影响？

**答案：**

闭包：既重用变量，又保护变量不被污染的一种机制。

特性：闭包是用外层函数包裹受保护的变量和内层函数对象，外层函数将内层函数对象返回到外部，使用者调用外层函数，获得返回的内层函数

影响:由于闭包时，变量的值都保存到内存中，会导致页面加载时内存消耗很大

### 15.js的原型链和继承？

**答案：**

**原型链: 由多级父元素，逐级继承，形成的链式结构。**

**继承：父对象的属性和方法，子对象无需重复创建就可直接使用。**

### 16.使用javascript打印出1-10000之间的所有对称数（例如121,1331等）

**答案：**

**function isSymmetryNum(start,end){**

**for(var i=start;i<end+1;i++){**

**var iNumber=+(i.toString().split("").reverse().join(""));**

**if(iNumber===i&&i>10){**

**console.log(i);**

**}**

**}**

**}**

**isSymmetryNum(1,10000);**

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

**答案：**

combo("get-element-by-id");

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);

}

console.log(arr);

msg=arr.join("");

console.log(msg);

}

### 18.解释 jsonp 的原理，以及为什么不是真正的 ajax

**答案：**

动态创建script标签,利用script标签的src属性可以获取任何域下的js脚本,通过这个特性,服务器端不在返回json格式,而是返回一段调用，某个函数的js代码，在src中进行了调用，这样实现了跨域。

Ajax与JSONP这两种技术看起来很像，目的也一样，都是请求一个url，然后把服务器返回的数据进行处理，实际上Ajax与JSONP有着本质上的不同。Ajax的核心是通过XMLHttpRequest获取数据，而JSONP的核心则是动态添加<script>标签来调用服务器提供的js文件。jsonp只支持get请求，ajax支持get和post请求。

### 19. 以下代码输出结果是什么？请说出给你出此答案的原因。

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

setTimeout(function(){

console.log(i),0})

}

**答案：**输出10次10

解析：

### 20．使用js代码为页面动态添加6个按钮，每个按钮上的文本为“button1”…”button6”,单击每个按钮时，分别弹出数字1、2……6，请按要求编写代码。

**答案：window.onload = function () {**

**var str = '';**

**for(var i =0 ;i<6;i++)**

**{**

**str+='<button>button'+(i+1)+'</button>';**

**}**

**document.body.innerHTML=str;**

**var btns=**

**document.getElementsByTagName("button");**

**//console.log(btns);**

**for(let i=0;i<btns.length;i++){**

**btns[i].onclick=function(){**

**alert(i);**

**}**

**}**

**}**

### 21. var str=" xiao ming "; var str2=str.trim();请写出console.log(str2);的结果？

**答案：**xiao ming（trim()会去掉字符串前后的空格）

**22.javascript 的本地对象，内置对象和宿主对象**

**答案**JavaScript的应用环境由宿主环境和运行期环境构成。宿主环境主要是指外壳程序（shell）和[Web浏览器](https://www.baidu.com/s?wd=Web%E6%B5%8F%E8%A7%88%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)等，运行期环境由JavaScript引擎内建的。

本地对象有哪些：

       Object、Function、Array、String、Number、Date、RegExp、Boolean、Error、EvalError、RangeError、ReferenceError、SyntaxError、TypeError、URIError 。

内置对象：由ECMAScript提供实现的、独立于宿主环境的所有对象，在ECMAScript程序开始执行时出现。

这意味着内置对象都是已经实例化好的，不需要我们再进行实例化了， ECMA-262定义的内置对象只有两个：Global和Math。

 宿主对象：由ECMAScript实现的宿主环境提供的对象。

可能这样不是很好理解，上面已经说过了宿主环境包括Web浏览器，所以我们可以这样理解，浏览器提供的对象都是宿主对象。

       也可以这样理解，因为本地对象是非宿主环境的对象，那么非本地对象就是宿主对象，即所有的BOM对象和DOM对象都是宿主对象。

       那么还有一种对象，那就是我们自己定义的对象，也是宿主对象。 最简单的理解：ECMAScript官方未定义的对象都属于宿主对象。

**23. 函数的几种定义方法**

**答案：**

1. **直接声明**

**function 函数名(参数列表){ 函数体; return 返回值}**

**问题: 会被声明提前！**

**解决:用赋值的方式就不会被声明提前。**

**2. 赋值方式**

**var函数名=function (参数列表){ 函数体; return 返回值}**

**优势: 不会被声明提前**

**3. 用new**

**var 函数名=new Function("参数1","参数2",...,"函数体;...")**

**24. this 关键字的指向**

**答案：**

obj.foo() == obj //方法调用模式,this 指向 obj

foo() == window; //函数调用模式,this 指向 window

new obj.foo() == obj //构造器调用模式, this 指向新建立对象

foo.call(obj) == obj;//APPLY 调用模式,this 指向 obj

**25.在 Javascript 中什么是伪数组？如何将伪数组转化为标准数组？**

**答案：**

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。

**26.什么是”use strict”？使用它的好处和坏处分别是什么？**

答案：严格模式，这种模式使得Javascript在更严格的条件下运行。

好处：

1. 消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;

2. 消除代码运行的一些不安全之处，保证代码运行的安全；

3. 提高编译器效率，增加运行速度；

4. 为未来新版本的Javascript做好铺垫。（注：经过测试 IE6,7,8,9 均不支持严格模式）

坏处：现在网站的 JS 都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被合并后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节。

## 二、DOM

### 1. 利用冒泡和不利用冒泡的差别

**答案：**

1. 绑定位置不同: 不利用冒泡绑定在目标元素上，利用冒泡绑定在父元素上
2. 监听对象的个数不同: 不利用冒泡会反复创建多个监听，利用冒泡始终只有一个监听
3. 动态生成的元素: 不利用冒泡无法自动获得事件处理函数，必须反复绑定

利用冒泡可让动态添加的子元素自动获得父元素的处理函数，无需反复绑定

### 2. 按HTML查找和按选择器查找的差别

**答案：**

1. 返回值不同: 按HTML查找返回动态集合，按选择器查找返回非动态集合
2. 效率不同: 按HTML查找效率高，按选择器查找效率低
3. 易用性不同: 当条件复杂时，按HTML查找繁琐，而按选择器查找简单

### 3. 列举DOM中常用优化

**答案：**

1. 查找时，如果之用一个条件就可查询出结果时，优先选择按HTML查找。如果查找条件复杂，则优先选择易用的按选择器查找
2. 添加时，尽量减少操作DOM树的次数，减少重排重绘。如果同时添加父元素和子元素，应先将子元素添加到到父元素，最后再将父元素添加到DOM树。如果添加多个平级子元素，则应先将子元素添加到文档片段，最后，再将文档片段添加到DOM树
3. 修改时，尽量减少重排重绘。如果同时修改多个元素的内容或样式，应尽量使用innerHTML和cssText方式修改元素的内容和样式。应使用class批量修改样式
4. 事件绑定时，应尽量利用冒泡减少事件监听的个数。

**4 .document load 和 document ready 的区别**

答案：

1.load是当页面所有资源全部加载完成后（包括DOM文档树，css文件，js文件，图片资源等），执行一个函数，问题：如果图片资源较多，加载时间较长，onload后等待执行的函数需要等待较长时间，所以一些效果可能受到影响。

2.$(document).ready()是当DOM文档树加载完成后执行一个函数 （不包含图片，css等）所以会比load较快执行，在原生的jS中不包括ready()这个方法，只有load方法就是onload事件。

## 三、jQuery

### 1. $的原理

**答案：**

1. $(“选择器”) 是先查找DOM元素，再将DOM元素放入jQuery对象中

其中自带优化:

如果选择器是#id，则自动调用getElementById

如果选择器是.class，则自动调用getElementsByClassName

如果选择器是标签名，则自动调用getElementsByTagName

否则，其它选择器，都自动调用querySelectorAll()

1. $(DOM元素) 是直接将DOM元素放入jQuery对象中
2. $(“HTML片段”) 是创建一个新元素
3. $(function(){}) 是绑定事件，在DOM内容加载后就提前触发。

### 2. 实现动画有几种方式，哪种好？

**答案：**

1. CSS: transition, animateion

优点: 由专门的排版引擎解析，效率高

缺点: 无法随意控制交互行为

1. JS: 定时器， $().animate()

优点: 可随意控制交互行为

缺点: 效率不如css动画

1. requestAnimationFrame()

优点: 可根据浏览器的刷新频率自动优化动画效果

缺点: 新API，有兼容性问题

### 3. 实现跨域访问有几种方式

**答案：**

主要有三种

1. JSONP:

客户端: 客户端动态添加script元素，用script发送请求，代替ajax请求，并携带客户端一个函数名到服务端

服务端: 接收客户端发来的函数名，将函数名和要返回的数据拼接为一条可执行的js语句，返回

1. CORS: Cross-Origin Resources Sharing

客户端正常发送ajax请求，服务端定义响应头，允许指定来源的请求跨域访问:

res.writeHead(200,{

…，

“Access-Control-Allow-Origin”:”允许的请求来源域名”

})  
 3. http代理方式跨域: (http-proxy)

纯客户端跨域。在客户端配置一个代理服务器程序，代替ajax实际发送请求，获取数据。因为不是ajax请求，所以，不受浏览器同源策略限制，可以随意发送跨域请求。  
 vue.config.js中:  
 module.exports={  
 … : … ;

devServer:{

proxy:{

“/”:{

target:”http://服务器端公共基础路径”,

changeOrigin:true

}

}

}  
 }

## 四、Vue

### 1. Vue的双向数据绑定原理是什么？

**答案：**

vue.js 是采用数据劫持结合发布者-订阅者模式的方式，通过Object.defineProperty()来劫持各个属性的setter，getter，在数据变动时发布消息给订阅者，触发相应的监听回调。

具体步骤：

第一步：需要observe的数据对象进行递归遍历，包括子属性对象的属性，都加上 setter和getter

这样的话，给这个对象的某个值赋值，就会触发setter，那么就能监听到了数据变化

第二步：compile解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图

第三步：Watcher订阅者是Observer和Compile之间通信的桥梁，主要做的事情是:

1、在自身实例化时往属性订阅器(dep)里面添加自己

2、自身必须有一个update()方法

3、待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调，则功成身退。

第四步：MVVM作为数据绑定的入口，整合Observer、Compile和Watcher三者，通过Observer来监听自己的model数据变化，通过Compile来解析编译模板指令，最终利用Watcher搭起Observer和Compile之间的通信桥梁，达到数据变化 -> 视图更新；视图交互变化(input) -> 数据model变更的双向绑定效果。

### 2. 请详细说下你对vue生命周期的理解

**答案：**

总共分为8个阶段创建前/后，载入前/后，更新前/后，销毁前/后。

创建前/后： 在beforeCreated阶段，vue实例的挂载元素$el和数据对象data都为undefined，还未初始化。在created阶段，vue实例的数据对象data有了，$el还没有。

载入前/后：在beforeMount阶段，vue实例的$el和data都初始化了，但还是挂载之前为虚拟的dom节点，data.message还未替换。在mounted阶段，vue实例挂载完成，data.message成功渲染。

更新前/后：当data变化时，会触发beforeUpdate和updated方法。

销毁前/后：在执行destroy方法后，对data的改变不会再触发周期函数，说明此时vue实例已经解除了事件监听以及和dom的绑定，但是dom结构依然存在

3. vue3中ref和toRefs的差别：

答案:

ref，专门用于将非响应式的数据，包装为响应式的数据。

toRefs，专门用于将ref包裹的一个大对象中，每个成员都转化为响应式的。

4. v-for为什么必须要加:key：  
 答案:

为每个html副本添加唯一标识，避免更新数组元素时重建整个列表，提高修改的效率。

5. 列举TypeScript中访问修饰符及其意义：  
 答案:

public: 公有属性，当前class、子类型class以及类外部都可访问

protected: 受保护的属性，当前class，子类型class可访问。类外部不能用.访问

private: 私有属性，仅当前class内可用。子类型class和类外部都不能访问。

6. 简述vuex的作用，及其组成部分:

答案: 只要多个组件都要共用一批数据时，就可用vuex。

其中包括:

state对象，专门保存多个组件共用的数据属性

mutations对象，专门保存修改state中变量的方法。

actions对象，专门保存发送异步ajax请求的方法。

7. 使用vue3实现点按钮搜索，一边输入一边搜索，按回车搜索:   
 答案:

<template>

<div class="home">

<input v-my-focus v-model="str" @keyup.enter="search"><button @click="search">百度一下</button>

</div>

</template>

<script lang="ts">

import { defineComponent, ref, toRefs, watch } from 'vue';

export default defineComponent({

directives:{

"my-focus":{

mounted(elem){

elem.focus();

}

}

},

setup(){ //创建阶段create

//1. 页面中所需的所有变量，放在一个ref()包裹的大对象中

var data=ref({

str:""

});

//2. 使用前，使用toRefs()从大的对象中解构出每个小的变量单独使用

//toRefs():只有toRefs解构出的变量才是受监控的！

let { str }=toRefs(data.value);

//从此无论有多少变量，ref和toRefs只需要写一次即可！

//3.所有方法都集中保存在methods对象中，不用ref包裹！

const methods={

search(){

if(str.value!=="")

console.log(`搜索${str.value}相关的内容`)

}

}

watch(str,(newVal,oldVal)=>{

console.log(`str的变量值从${oldVal}被改成${newVal}`);

methods.search();

})

//4. 今后，return永远这么写！

return {

//无论将来有多少变量

//这里永远这么写！

...toRefs(data.value),

//将来无论有多少函数，这里永远不变！

...methods

}

}

});

</script>