# 前端面试总结

**一、cookie，localstorage,sessionStorage**

**cookie**: 大小不能超过4k，不设置过期时间每次访问都会存在（不会因为关闭浏览器或窗口清除）、cookie的数据会自动传递到服务器，服务器也可以写cookie到客户端

sessionStorage：大小可以到5m以上、关闭浏览器窗口会删除数据、不会把数据传递到服务器，只会在本地保存

localStorage：除了手动删除，会一直在本地保存（不会因为浏览器关闭删除数据）、大小可以到5m以上

**二、数组的去重与排序**

Let arr = [1,2,3,4,5,6,7,7,7,2,5,5,4]

第一种：Function sort(arr){

let narr = arr.filter((val,ind,arr)=>{

return arr.indexOf(val) === ind;

})

return narr.sort((a,b)=>(a-b));

}

第二种：Function sort(arr){

let newArr = [];

arr.forEach((val,ind)=>{

if(newArr.indexOf(val) === -1){

newArr.push(val);

}

})

return newArr.sort((a,b)=>a-b);

}

第三种：function sort(arr){

let newArr = arr.concat(); // 数组深拷贝

// 或者 arr.slice(0)

For(let i=0;i<newArr.length;i++){

If(newArr.indexOf(newArr[i]) !== i){

newArr.slice(i,1);

i--

}

}

return newArr;

}

第四种：function sort(arr){

return Array.from(new Set(arr)).sort((a,b)=>a-b);

}

数组的深拷贝：

Arr.concat()、arr.slice(0)

对象数组的拷贝：

JSON.parse(JSON.stringify(json))

Json对象的拷贝：

可以使用Object.keys(json)或者Object.values(json)，配合for of进行拷贝

JSON.parse(JSON.stringify(json))这个依然可以使用

三、数组的冒泡排序

let arr = [7, 4, 3, 67, 23, 34, 15, 1, 8];

function sort(arr){

let len = arr.length-1;

let flag = true

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

for(let j=0;j<len-i;j++){

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

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

Flag = false;

}

}

If(flag) break;

}

}

**四、JS数据类型**

原始类型：

1. number类型：专门保存类型的类型，可用于数字的计算
2. string类型：字符串，用于存储字符串
3. Boolean：true 或 false
4. Null：不指向任何地址、手动赋值、清空内容等。。。
5. Undefined：表示未定义，这个值的变量不含有值
6. Symbol：表示独一无二的值（补充：symbol函数不能使用new命令，否则会报错、因为symbol是一个存在原始类型的值、不是对象symbol函数可以接受一个字符串的参数，表示对symbol实例的描述，主要是为了在控制台显示，或者为字符串时，比较容易区分）

引用类型

7、object : **array**(数组)、**object**(大多数类型之都是obj类型的实例)、**function**（每个函数都是function类型的实例）

**五、闭包**

当函数可以记住并访问所在的词法作用域时，就产生了闭包，即使函数是在当前词法作用域之外执行

无论通过何种手段将内部函数传递到所在的词法作用域外，它都会持有对原始定义作用域的引用，无论在何处执行这个函数都会使用闭包。

（简而言之，就是访问其他函数的局部变量，形成了闭包）

**闭包就是能够读取其它函数内部变量的函数**

例子1：function foo(){

Var a = 2;

Function baz(){

Console.log(a);

}

Return baz;

}

var bar = foo(); bar(); // bar在词法作用域访问到了baz，形成了闭包

例子2：function foo(){

Var a = 2;

Function baz(){

Console.log(a);

}

bar(baz);

}

Function bar(fn){

Fn(); // 这就是闭包

}

Foo(); //把内部函数baz传递给bar，当调用这个内部函数时（现在是fn），它涵盖的foo（）内部的作用域闭包可以观察到了，因为它能够访问a。

例子3：传递也可以是间接的。

var fn;

function foo(){

Var a = 2;

Function baz(){

Console.log(a);

}

fn = baz // 将baz分配给全局变量

}

Function bar(){

fn(); // 这就是闭包

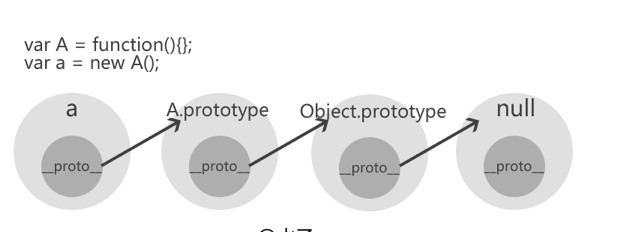
}

Foo(); bar(); //2

六、原型链：

由于\_\_proto\_\_是任何对象都有的属性，而js里万物街对象，所以会形成一条\_\_proto\_\_连起来的链条，递归访问\_\_proto\_\_必须要最终到头，并且值是null

当js引擎查找对象的属性时，县查找对象本身是否存在该属性，如果不存在，会在原型链上查找，但不会查找自身的prototype



var A = function(){};

var a = new A();

console.log(a.\_\_proto\_\_); //A {}（即构造器function A 的原型对象）

console.log(a.\_\_proto\_\_.\_\_proto\_\_); //Object {}（即构造器function Object 的原型对象）

console.log(a.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_); //null

HTML的新特性：

1、标签语义化，比如header，footer，nav，aside，article，section等，新增了很多表单元素，入email，url等，除去了center等样式标签，还有除去了有性能问题的frame，frameset等标签

**对html5的语义话的理解**

html5的语义化指的是用正确的标签包含正确的内容，比如nav标签，里面就应该包含导航条的内容，而不是用做其他的用途，标签语义化的好处就是结构良好，便于阅读，方便威化，也有利于爬虫的查找，提高搜索率。

2、音视频元素，video，audio的增加使得我们不需要在依赖外部的插件就可以往网页中加入音视频元素。

3、新增很多api，比如获取用户地理位置的window.navigator.geoloaction，

4、websocket

websocket是一种协议，可以让我们建立客户端到服务器端的全双工通信，这就意味着服务器端可以主动推送数据到客户端

5、缓存

html5允许我们自己控制哪些文件需要缓存，哪些不需要，具体的做法如下：

1、首先给html添加manifest属性，并赋值为cache.manifest

2、cache.manifest的内容为**:**

CACHE MANIFEST

#v1.2

CACHE **:** *//表示需要缓存的文件*

a.js

b.js

NETWORK**:** *//表示只在用户在线的时候才需要的文件，不会缓存*

c.js

FALLBACK

**/** /index.html /**/**表示如果找不到第一个资源就用第二个资源代替

6、web worker，web worker是运行在浏览器后台的js程序，他不影响主程序的运行，是另开的一个js线程，可以用这个线程执行复杂的数据操作，然后把操作结果通过postMessage传递给主线程，这样在进行复杂且耗时的操作时就不会阻塞主线程了。

**递归**

同源策略是浏览器有一个很

**同源策略**

同源策略是浏览器有一个很重要的概念。所谓同源是指，域名，协议，端口相同。不同源的客户端脚本(javascript、ActionScript)在没明确授权的情况下，不能读写对方的资源。简单的来说，浏览器允许包含在页面A的脚本访问第二个页面B的数据资源，这一切是建立在A和B页面是同源的基础上。

**跨域的几种方式**

jsonp（利用script标签的跨域能力）跨域、websocket（html5的新特性，是一种新协议）跨域、设置代理服务器（由服务器替我们向不同源的服务器请求数据）、CORS（跨源资源共享，cross origin resource sharing）、iframe跨域、postMessage(包含iframe的页面向iframe传递消息)，document.domain跨域（比如：在一个文件中设置了document.domain="[http://qq.com](http://link.zhihu.com/?target=http%3A//qq.com)",那么另一个设置了document.domain="[http://qq.com](http://link.zhihu.com/?target=http%3A//qq.com)"的，他们两个就是同源）

**JavaScript的值类型和引用类型**

JavaScript有两种类型的数据，值类型和引用类型，一般的数字，字符串，布尔值都是值类型，存放在栈中，而对象，函数，数组等是引用类型，存放在堆中，对引用类型的复制其实是引用复制，相当于复制着地址，对象并没有真正的复制。

**var** a**=**5;**var** b**=**a;a**=null**; *//那么b是5*

**var** a**=**{},**var** b**=**a;b.name**=**"mbj";

console.log(a.name); *//mbj，因为a，b指向同一个对象*

a**=null**;console.log(**typeof** b); *//object，a=null，只是a不再指向该对象，但是这个对象还是在堆中确确实实的存在，b依然指向它。*

[**Doctype作用，标准模式与兼容模式的区别**](https://www.cnblogs.com/sunhe/p/4994058.html)

<!DOCTYPE>声明位于位于HTML文档中的第一行，处于 <html> 标签之前。告知浏览器的解析器用什么文档标准解析这个文档。DOCTYPE不存在或格式不正确会导致文档以兼容模式呈现。

标准模式的排版 和JS运作模式都是以该浏览器支持的最高标准运行。在兼容模式中，页面以宽松的向后兼容的方式显示,模拟老式浏览器的行为以防止站点无法工作。  
  
HTML5 为什么只需要写 <!DOCTYPE HTML>？

HTML5 不基于 SGML，因此不需要对DTD进行引用，但是需要doctype来规范浏览器的行为（让浏览器按照它们应该的方式来运行）；

而HTML4.01基于SGML,所以需要对DTD进行引用，才能告知浏览器文档所使用的文档类型

JS 数据之间的互相转换

# 盒子模型

1、有两种盒子模型：IE盒子模型，W3C盒子模型

2、盒模型：内容（content），填充（padding），边框（border），外边距（margin）

3、区别：IE把content部分把padding与border计算在内

也就是box-sizing： border-box

**栈堆的概念**

说白了，堆和栈都是在内存里开辟出来的一块专门的区域。

堆，又称队列，是先入先出的结构，就像排队的先来后到一样（当然还有循环队列首尾相连，这个更常用），如果你去看数据结构方面的书籍，看到这里也很好理解。

栈，是后入先出的结构，数据进栈又称压栈。楼上几位的例子比较形象，我就不赘述了。

但需要注意的是，栈和堆原本只是普通的内存区，但是因为人为或者系统的定义设计，使得它们有了不同的性质，可以用在不同的场合。像递归，就是隐式利用了栈结构的例子