汪汪， 腾讯web前端工程师

41

这篇文章是对我大四秋招以来面试的总结，里面包含前端面试知识的方方面面，目前本人已经拿到腾讯offer，希望能对后面找工作的学习学妹们有所帮助。

腾讯面试对基础比较看重，然后需要你有两三个比较好的项目，一面重视面试者对前端基础的把握，还要手写代码，不过不难，二面部门的leader面，这一面比较难，官会对你的项目细节进行深挖，所以说项目要牛逼一点，最后还会有一道逻辑题（我没有答上来），三面是HR面，如果你想进大公司的话，下面这些技术是肯定要掌握的：html5，css3，JavaScript，略懂一点jQuery源码，Node.js，express，mongoose，数据库mongodb。大公司问的核心在于JavaScript。如果下面的知识点你都以打上来，恭喜你拿下bat不是问题--2016-11-11写

转载请注明出处，码这么多字不容易。

一、html+css部分、

（1）css盒模型，可能会要求手写一个布局，这个布局基本上用到的css是margin的负值，boxing-sizing：border-box，布局尽量往这方面想。浏览器布局的基本元素盒，在w3c的标准模式下，width=width，但是在怪异模式下，width=border*2+padding*2+width;其中后代元素的width：100%；参照的是右边的那个width，

（2）html5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frame，frameset等标签

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、webstorage，webstorage是本地存储，存储在客户端，包括localeStorage和sessionStorage，localeStorage是持久化存储在客户端，只要用户不主动删除，就消失，sessionStorage也是存储在客户端，但是他的存在时间是一个回话，一旦浏览器的关于该回话的页面关闭了，sessionStorage就消失了，

6、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
```

7、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。

（3）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。

（4）cookie，sessionStorage，localeStorage的区别

cookie是存储在浏览器端，并且随浏览器的请求一起发送到服务器端的，它有一定的过期时间，到了过期时间自动会消失。sessionStorage和localeStor是存储户端的，同属于web Storage，比cookie的存储大小要大有8m，cookie只有4kb，localeStorage是持久化的存储在客户端，如果用户不手动清除的话，不会自动消失

一直存在，sessionStorage也是存储在客户端，但是它的存活时间是在一个回话期间，只要浏览器的回话关闭了就会自动消失。

（5）多个页面之间如何进行通信

使用cookie，使用web worker，使用localeStorage 41 sionStorage

（6）浏览器的渲染过程

1、首先获取html，然后构建dom树

2、其次根据css构建render树，render树中不包含定位和几何信息

3、最后构建布局数，布局是含有元素的定位和几何信息

（7）重构、回流

浏览器的重构指的是改变每个元素外观时所触发的浏览器行为，比如颜色，背景等样式发生了改变而进行的重新构造新外观的过程。重构不会引发页面的重新布局，不伴随着回流，

回流指的是浏览器为了重新渲染页面的需要而进行的重新计算元素的几何大小和位置的，他的开销是非常大的，回流可以理解为渲染树需要重新进行计算，一般最好触素的重构，避免元素的回流；比如通过通过添加类来添加css样式，而不是直接在DOM上设置，当需要操作某一块元素时候，最好使其脱离文档流，这样就不会引起回了，比如设置position：absolute或者fixed，或者display：none，等操作结束后在显示。

二、JavaScript部分

（1）JavaScript的数据类型

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

复杂数据类型：Object，Array，Function，RegExp，Date，Error

全局数据类型：Math

（2）JavaScript的闭包

闭包简单的说就是一个函数能访问外部函数的变量，这就是闭包，比如说：

```
function a(x){
       var tem=3;
      function b(y){
          console.log(x+y+(++tem));
     }
}
```

a函数中的b函数就是闭包了，b函数可以使用a函数的局部变量，参数，最典型的闭包应该是下面这样，将定义在函数中的函数作为返回值

```
function a(x){
       var tem=3;
      function b(y){
          console.log(x+y+(++tem));
     }
return b;
}
```

闭包的另一种作用是隔离作用域，请看下面这段代码

```
for(var i=0;i<2;i++){
      setTimeout(function(){
              console.log(i);
        },0);
}
```

上面这段代码的执行结果是2,2而不是0,1，因为等for循环出来后，执行setTimeout中的函数时，i的值已经变成了2，这就是没有隔离作用域所造成的，请看下面代码

```
for(var i=0;i<2;i++){
      (function(i){
             setTimeout(function(){
              console.log(i);
        },0)
    })(i);
}
```

这样就会输出0,1，我们的立即执行函数创建了一个作用域，隔离了外界的作用域，闭包的缺点是，因为内部闭包函数可以访问外部函数的变量，所以外部函数的变量不被释放，如果闭包嵌套过多，会导致内存占用大，要合理使用闭包。

（3）new 操作符到底做了什么

41

首先，new操作符为我们创建一个新的空对象，然后i 量指向该对象，

其次，空对象的原型执行函数的原型，

最后，改变构造函数内部的this的指向

代码如下：

```
var obj={};
obj.__proto__=fn.prototype;
fn.call(obj);
```

（4）改变函数内部this指针的指向函数

call和apply，假设要改变fn函数内部的this的指向，指向obj，那么可以fn.call(obj);或者fn.apply(obj);那么问题来了，call和apply的区别是什么，其是call和apply的区在于参数，他们两个的第一个参数都是一样的，表示调用该函数的对象，apply的第二个参数是数组，是[arg1,arg2,arg3]这种形式，而call是arg1,arg2,arg3这样的形式还有一个bind函数，

var bar=fn.bind(obj);那么fn中的this就指向obj对象了，bind函数返回新的函数，这个函数内的this指针指向obj对象。

（5）JavaScript的作用域和作用域链

JavaScript的作用域指的是变量的作用范围，内部作用域由函数的形参，实参，局部变量，函数构成，内部作用域和外部的作用域一层层的链接起来形成作用域链，当在函数内部要访问一个变量的时候，首先查找自己的内部作用域有没有这个变量，如果没有就到这个对象的原型对象中去查找，还是没有的话，就到该作用域所在的作用域找，直到到window所在的作用域，每个函数在声明的时候就默认有一个外部作用域的存在了，比如：

```
var t=4;
function foo(){
       var tem=12;
      funciton bar(){
       var temo=34;
       console.log(t+" "+tem+" "+temo);
      }
}
```

bar找t变量的过程就是，先到自己的内部作用域中找，发现没有找到，然后到bar所在的最近的外部变量中找，也就是foo的内部作用域，还是没有找到，再到window用域中找，结果找到了

（6）JavaScript的继承

```
function A(name){  this.name=name; }
A.prototype.sayName=function(){ console.log(this.name); }
function B(age){ this.age=age; }
```

原型继承

```
B.prototype=new A("mbj");  //被B的实例共享
var foo=new B(18);
foo.age;    //18,age是本身携带的属性
foo.name;   //mbj，等价于foo.__proto__.name
foo.sayName(); //mbj,等价于foo.__proto__.proto__.sayName()
foo.toString();  //"[object Object]",等价于foo.__proto__.__proto__.__proto__.toString();
```

这样B通过原型继承了A，在new B的时候，foo中有个隐藏的属性__proto__指向构造函数的prototype对象，在这里是A对象实例，A对象里面也有一个隐藏的属性__proto__,指向A构造函数的prototype对象，这个对象里面又有一个__proto__指向Object的prototype

这种方式的缺第一个缺点是所有子类共享父类实例，如果某一个子类修改了父类，其他的子类在继承的时候，会造成意想不到的后果。第二个缺 构造子类实例的候，不能给父类传递参数。

构造函数继承

```
function B(age,name){  this.age=age;A.call(this,name); }
var foo=new B(18,"wmy");
```

```
foo.name;     //wmy
foo.age;      //18
foo.sayName();   //undefined
```

采用这种方式继承是把A中的属性加到this上面，这样相当于就是B的属性，sayName不在A的构造函数中，所以访问不到sayName。这种方法的缺点是父类的prototype中的函数不能复用。

41

原型继承+构造函数继承

```
function B(age,name){  this.age=age;A.call(this,name); }
B.prototype=new A("mbj");
var foo=new B(18,"wmy");
foo.name;     //wmy
foo.age;      //18
foo.sayName();   //wmy
```

这样就可以成功访问sayName函数了，结合了上述两种方式的优点，但是这种方式也有缺点，那就是占用的空间更大了。

（7）JavaScript变量提升

请看下面代码

```
var bar=1;
function test(){
  console.log(bar);     //undeifned
  var bar=2;
  console.log(bar);  //2
}
test();
```

为什么在test函数中会出现上述结果呢，这就是JavaScript的变量提升了，虽然变量bar的定义在后面，不过浏览器在解析的时候，会把变量的定义放到最前面，上面的函数相当于

```
function test(){
  var bar;
  console.log(bar);   //undefined
  bar=2;
  console.log(bar);   //2
}
```

再看

```
var foo=function(){  console.log(1); }
function foo(){  console.log(2); }
foo();  //结果为1
同样的，函数的定义也会到提升到最前面，上面的代码相当于
function foo(){  console.log(2); }
var foo;
foo=funciton(){ console.log(1); }
foo();   //1
```

（8）JavaScript事件模型

原始事件模型，捕获型事件模型，冒泡事件模型，

原始事件模型就是ele.onclick=function(){}这种类型的事件模型

冒泡事件模型是指事件从事件的发生地（目标元素），一直向上传递，直到document，

捕获型则恰好相反，事件是从document向下传递，直到事件的发生地（目标元素）

IE是只支持冒泡事件模型的，下面是兼容各个浏览器的事件监听代码

```
EventUtil={
  addListener:function(target,type,handler){
    if(target.addEventListener){
        target.addEventListener(type,handler);
    }else if(target.attachEvent){
        target.attach("on"+type,function(){
              handler.call(target);  //让handler中的this指向目标元素
        });
    }else{
```

```
        target["on"+type]=handler;
    }
  },
  removeListener:function(target,type,handler){
      if(target.removeEventListener){
        target.removeEventListener(type,handler)
    }else if(target.detachEvent){
        target.detachEvent("on"+type,handler);
    }else{
        target["on"+type]=null;
    }
  },
  getEvent:function(e){      //获取事件对象
    var evt=window.event||e;
    return evt;
  },
  getTarget:function(e){      //获得目标对象
    var evt=EventUtil.getEvent(e);
    var target;
    if(evt.target){ target=evt.target;}
    else {target=evt.srcElement;}
    return target;
  },
  stopPropagation:function(e){  //停止冒泡
    var evt=EventUtil.getEvent(e);
    if(evt.stopPropagation) {evt.stopPropagation();}
    else {evt.cancelBubble=true;}
  },
  preventDefault:function(e){   //阻值默认行为的发生
    var evt=EventUtil.getEvent(e);
    if(evt.preventDefault){ evt.preventDefault(); }
    else {e.returnValue=false;}
  }
 }
```

41

（9）内存泄漏

内存泄漏指的是浏览器不能正常的回收内存的现象

（10）浏览器的垃圾回收机制

垃圾收集器必须跟踪哪个变量有用哪个变量没用，对于不再有用的变量打上标记，以备将来收回其占用的内存，内存泄露和浏览器实现的垃圾回收机制息息相关， 而浏器实现标识无用变量的策略主要有下两个方法：

第一，引用计数法

跟踪记录每个值被引用的次数。当声明一个变量并将引用类型的值赋给该变量时，则这个值的引用次数就是1。如果同一个值又被赋给另一个变量，则该值的引用次 数相反，如果包含对这个值引用的变量又取得另外一个值，则这个值的引用次数减1.当这个值的引用次数变成0时，则说明没有办法访问这个值了，因此就 可以将其占用的存空间回收回来。

```
如：  var a = {};     //对象{}的引用计数为1
      b = a;          //对象{}的引用计数为 1+1
      a = null;       //对象{}的引用计数为2-1
```

所以这时对象{}不会被回收;

IE 6, 7 对DOM对象进行引用计数回收， 这样简单的垃圾回收机制，非常容易出现循环引用问题导致内存不能被回收， 进行导致内存泄露等问题，一般不用引用计数法

第二，标记清除法

到2008年为止，IE,Firefox,Opera,Chrome和Safari的javascript实现使用的都是标记清除式的垃圾收集策略（或类似的策略），只不过垃圾收集的时间间隔互有不同。

标记清除的算法分为两个阶段，标记(mark)和清除(sweep). 第一阶段从引用根节点开始标记所有被引用的对象，第二阶段遍历整个堆，把未标记的对象清除。

（11）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。

（12）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）、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)

（13）异步和同步

同步指下一个程序的执行需要等到上一个程序执行完毕，也就是得出结果后下一个才能执行，

异步指的是上一个程序指向后，下一个程序不用等到 41 程序出结果就能执行，等上一个出结果了调用回调函数处理结果就好。

（14）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。
```

（15）优化下面代码

```
var str="我喜欢我可爱的女朋友，";
str=str+"她叫喵喵，";
str=str+"她时而可爱，时而认真，";
str=str+"她那天真的笑声可以让人忘掉一切烦恼。";
console.log(str);
```

这里的优化主要是对加号操作符的优化，因为加号在JavaScript中非常耗时和耗内存，需要经过以下六步：

```
1、首先开辟一块临时空间，存储字符串，
2、然后在开辟一块空间
3、把str中的字符串复制到刚刚开辟的空间
4、在把需要连接的字符串复制到str后面
5、str指向这块空间
6、回收str原来的空间和临时空间
```

优化的方法是使用数组的push方法，数组是连续的存储空间，可以省下很多步

```
var res=[];
var str="我喜欢我可爱的女朋友，";
res.push(str);
res.push("她叫喵喵，");
res.push("她时而可爱，时而认真，");
res.push("她那天真的笑声可以让人忘掉一切烦恼。");
console.log(res.join(""));
```

（16）封装cookie的添加，删除，查询方法

cookie是存储在浏览器端的，可以用于存储sessionID，也可以用于自动登陆，记住密码等，但是在浏览器端并没有官方的操作cookie的方法，下面我们来封装一下：

```
CookieUtil={
    addCookie:function(key,value,options){
        var str=key+"="+escape(value);
        if(options.expires){
           var curr=new Date();   //options.expires的单位是小时
           curr.setTime(curr.getTime()+options.expires*3600*1000);
           options.expires=curr.toGMTString();
        }
        for(var k in options){   //有可能指定了cookie的path，cookie的domain
           str+=";"+k+"="+options[k];
        }
        document.cookie=str;
    },
    queryCookie:function(key){
      var cookies=document.cookie;
     //获得浏览器端存储的cookie,格式是key=value;key=value;key=value
      cookies+=";";
      var start=cookies.indexOf(key);
      if(start<=-1){ return null; }  //说明不存在该cookie
      var end=cookies.indexOf(";",start);
      var value=cookies.slice(start+key.length+1,end);
      return unescape(value);
    },
    deleteCookie:function(key){
      var value=CookieUtil.queryCookie(key);
      if(value===null){return false;}
```

```
        CookieUtil.addCookie(key,value,{expires:0});//把过期时间设置为0，浏览器会马上自动帮我们删除cookie
    }
}
```

（17）事件委托机制

41

事件委托指的是，不再事件的发生地设立监听函数，而是事件发生地的父元素或者祖先元素设置监听器函数，这样可以大大提高性能，因为可以减少绑定事件的元素如：

```
<ul>
 <li></li>
 <li></li>
 <li></li>
</ul>
```

要给li元素绑定click事件，使用事件委托机制的话，就只需要给ul绑定click事件就行了，这样就不需要给每个li'绑定click事件，减小内存占用，提高效率，有兴趣的童鞋以去看看jQuery的live，bind，on，delegate函数的区别，这几个函数就采用了事件委托机制。

三、其他部分

（1）http状态码

http状态码是表示服务器对请求的响应状态，主要分为以下几个部分

1**：这类响应是临时响应，只包含状态行和某些可选的响应头信息，并以空行结束

2**：表示请求成功，

3**：表示重定向

4**：表示客户端错误

5**：表示服务器端错误

100（continue），客户端应当继续发送请求。这个临时响应是用来通知客户端它的部分请求已经被服务器接收

200（OK），表示请求成功，请求所希望的响应头或数据体将随此响应返回。

202（Accepted），服务器已接受请求，但尚未处理。

204（No-Content），服务器成功处理了请求，但不需要返回任何实体内容

205（Reset-Content），服务器成功处理了请求，且没有返回任何内容。但是与204响应不同，返回此状态码的响应要求请求者重置文档视图。该响应主要是被用于接用户输入后，立即重置表单，以便用户能够轻松地开始另一次输入。

206（Partial-Content），服务器已经成功处理了部分 GET 请求。

301（Moved-Permanently），永久性重定向

302（Moved-Temporarily），暂时性重定向

304（Not-Modified），浏览器端缓存的资源依然有效

400（Bad-Reques），请求有误，当前请求无法被服务器理解。

401（Unauthorized），当前请求需要用户验证。

403（Forbidden），服务器已经理解请求，但是拒绝执行它。

404（Not-Found），请求的资源没有被找到

500（Interval Server Error），服务器内部错误

502（Bad GateWay），网关出错

503（Service Unavailable），由于临时的服务器维护或者过载，服务器当前无法处理请求。

504（Gateway Timeout），作为网关或者代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI标识出的服务器，例如HTTP、FTP、LDAP）或者辅助服务（例如DNS）收到响应。

（2）xss，csrf的概念以及防范方法

大公司如bat在面试的时候，web安全问题是必问的问题，所以一定要懂，要彻底理解xss和csrf的概念和防范方式，最好在项目中有用到对这两种攻击的防范这样会给的面试加很多分。由xss和csrf涉及的东西比较多，我就不具体给出了，详情请看XSS攻击及防御 ，CSRF攻击

（3）CommonJs，AMD，CMD规范

对于前端模块化来说，这三个规范是必须要了解的，详情请看我的这篇文章CommonJS，AMD，CMD

（4）谈谈对前端模块化的理解

41

前端模块话就是把复杂的文件分成一个个独立的模块，比如js文件，分成独立的模块之后有利于代码的重用和维护，但是这样又会引来模块与模块之间的依赖问题，所以有了CommonJS、AMD、CMD规范，最后出现了webpack，webpack就是前端模块话的一种解决方案，基本上大公司都会使用webpack，想要详细的学习webpack话请看webpack简明使用教程

（5）优雅降级和渐进增强

优雅降级指的是一开始就构建功能完好的网站，然后在慢慢兼容低版本的浏览器，使得各个浏览器之间的差异不要太大。

渐进增强是指在基本功能得到满足的情况下，对支持新特性的浏览器使用新特性，带给用户更换的体验。

优雅降级和渐进增强的出发点不同，前者是慢慢向下兼容，是向后看，后着是慢慢向上，增强功能，是向前看。

（6）前端优化（提高网页的加载速度）

1、使用css sprites，可以有效的减少http请求数

2、使用缓存

3、压缩js，css文件，减小文件体积

4、使用cdn，减小服务器负担

5、懒加载图片

6、预加载css，js文件

7、避免dom结构的深层次嵌套

8、给DOM元素添加样式时，把样式放到类中，直接给元素添加类，减少重构，回流

更多详细的前端优化请看前端优化：雅虎35条　或者前端性能优化----yahoo前端性能团队总结的35条黄金定律

四、前端学习文章推荐

知乎上面有人推荐了很多前端学习网站，具体信息请看

关于 Javascript 学习，有哪些好的博客或者网站推荐？

发布于 2016-11-17　13 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

蛋白喵， 码农一只

ie6 z-index bug，双倍margin bug

js变量作用域，原型链概念，对象概念

都是些比较经典的，不过ie6可能逐渐退出视线，也要看工作需要而出题

发布于 2011-09-15　1 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

托托， Don't have fucking excuses, just go.

github.com/hawx1993/Fro...

发布于 2016-04-16　添加评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

织田玲绪， 求职中的野生Lv1前端工程师

@陀振华 这个题目不错~

发布于 2013-06-03　添加评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

Tony

6 人赞同

1. 使用ajax需要考虑哪些方面？优缺点？

2. ==与===的区别

3. 闭包，应用场景