## 1.有赞电面

### Q.let.const?const能修改对象里面的属性么?作用域都有哪些？

A：const对象的属性可以被修改。

块作用域，函数作用域

### Q.js继承

A：一个对象的prototype = 父对象的示例。

(super拷贝父类的属性，副本)

### Q.定位都有哪些？相对定位是相对哪里？

A相对定位，绝对定位，固定定位）。

ANSWER：absolute 生成绝对定位的元素，相对于 static 定位以外的第一个祖先元素进行定位。 relative 生成相对定位的元素，相对于其在普通流中的位置进行定位。

### Q.cookie怎么实现的？

ANSWER：Cookie实际上是一小段的文本信息。客户端请求服务器，如果服务器需要记录该用户状态，就使用response向客户端浏览器颁发一个Cookie。客户端浏览器会把Cookie保存起来。当浏览器再请求该网站时，浏览器把请求的网址连同该Cookie一同提交给服务器。服务器检查该Cookie，以此来辨认用户状态。服务器还可以根据需要修改Cookie的内容。

Cookie具有不可跨域性。

由于浏览器每次请求服务器都会携带Cookie，因此Cookie内容不宜过多，否则影响速度。Cookie的内容应该少而精。

### Q.从页面输入url到展示页面发生了什么？

ANSWER. 总体来说分为以下几个过程:

1. DNS解析
2. TCP连接
3. 发送HTTP请求
4. 服务器处理请求并返回HTTP报文
5. 浏览器解析渲染页面
6. 连接结束

### Q.浏览器缓存的页面是什么？DOM结构？整个页面？<http://www.cnblogs.com/skynet/archive/2012/11/28/2792503.html>（讲的很好~~）

Cache-control：max-age|max-scale，expires

A：？？

### Q.跨域？怎么解决？

（<http://www.cnblogs.com/2050/p/3191744.html>）

A. 跨域，指的是浏览器不能执行其他网站的脚本。它是由浏览器的同源策略造成的，是浏览器对javascript施加的安全限制。

方法：1.jsonp, 要注意JSONP只支持GET请求，不支持POST请求。。（comet忘了）

**2、代理：**

例如www.123.com/index.html需要调用www.456.com/server.php，可以写一个接口www.123.com/server.php，由这个接口在后端去调用www.456.com/server.php并拿到返回值，然后再返回给index.html，这就是一个代理的模式。相当于绕过了浏览器端，自然就不存在跨域问题。

**3、PHP端修改header（XHR2方式）**

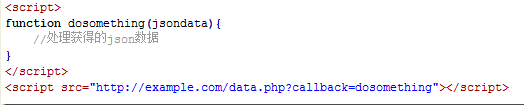
在php接口脚本中加入以下两句即可：

header('Access-Control-Allow-Origin:\*');//允许所有来源访问

header('Access-Control-Allow-Method:POST,GET');//允许访问的方式

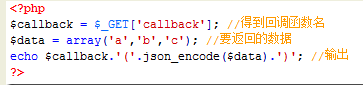
### Q.jsonp怎么实现的？

A：在js中，我们直接用XMLHttpRequest请求不同域上的数据时，是不可以的。但是，在页面上引入不同域上的js脚本文件却是可以的，jsonp正是利用这个特性来实现的。

比如，有个a.html页面，它里面的代码需要利用ajax获取一个不同域上的json数据，假设这个json数据地址是<http://example.com/data.php>,那么a.html中的代码就可以这样：

我们看到获取数据的地址后面还有一个callback参数，按惯例是用这个参数名，但是你用其他的也一样。当然如果获取数据的jsonp地址页面不是你自己能控制的，就得按照提供数据的那一方的规定格式来操作了。

因为是当做一个js文件来引入的，所以<http://example.com/data.php>返回的必须是一个能执行的js文件，所以这个页面的php代码可能是这样的:

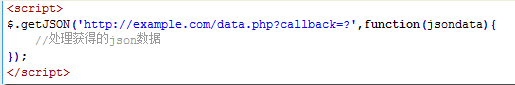


最终那个页面输出的结果是: 

所以通过<http://example.com/data.php?callback=dosomething>得到的js文件，就是我们之前定义的dosomething函数,并且它的参数就是我们需要的json数据，这样我们就跨域获得了我们需要的数据。

这样jsonp的原理就很清楚了，通过script标签引入一个js文件，这个js文件载入成功后会执行我们在url参数中指定的函数，并且会把我们需要的json数据作为参数传入。所以jsonp是需要服务器端的页面进行相应的配合的。

知道jsonp跨域的原理后我们就可以用js动态生成script标签来进行跨域操作了，而不用特意的手动的书写那些script标签。如果你的页面使用jquery，那么通过它封装的方法就能很方便的来进行jsonp操作了。



原理是一样的，只不过我们不需要手动的插入script标签以及定义回掉函数。jquery会自动生成一个全局函数来替换callback=?中的问号，之后获取到数据后又会自动销毁，实际上就是起一个临时代理函数的作用。$.getJSON方法会自动判断是否跨域，不跨域的话，就调用普通的ajax方法；跨域的话，则会以异步加载js文件的形式来调用jsonp的回调函数。

### Q.请求数据，除了ajax还有什么办法？

<https://www.zhihu.com/question/33561004/answer/56980806>

A．查数据库。从数据库中取值。

Q．接上，还有什么方法？

A．1. HTML赋值  
2. JS赋值  
3. script填充JSON

填充 JSON 数据到 `<script>` 标签中，前端通过 DOM 获取 JSON字符串并解析成对象。

<script type="text/template" id="data">{"username":"nimojs","userid":1}</script>

<script>

var data = JSON.parse($('#data').html());

//{username:"nimojs",userid:1}

</script>

4. AJAX获取JSON

使用 AJAX 获取JSON数据

<span id="showdata">查看资料</span>

<div style="display:none;" id="box">

<h2>用户信息</h2>

<p id="info"><img src="loading.gif" /></p>

</div>

$('#showdata').on('click',function(){

$('#box').show();

$.getJSON('/ajax/userdata/',function(oData){

// oData = {"username":"nimojs","userid":1}

$('#info').html('用户名：' + oData.username + '<br>用户ID：' + oData.userid);

})

})

5. WebSocket实时传输数据

(<http://www.runoob.com/html/html5-websocket.html> )

### Q.实现异步的方法有哪些？JS引擎是怎么实现异步的？

<http://blog.csdn.net/sdfujichao/article/details/52160028>

http://www.cnblogs.com/sprying/archive/2013/05/26/3100639.html

A．1.回调(这是异步编程最基本的方法。)

2.事件监听。

三、发布/订阅

四、Promises对象

### Q.布局都有哪些？flex知道么？弹性布局

Flex 布局，可以简便、完整、响应式地实现各种页面布局。目前，它已经得到了所有浏览器的支持，这意味着，现在就能很安全地使用这项功能。

<http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html>

### Q.与缓存有关的HTTP头？

Cache-control，expires，last-modified,If-Modified-Since,ETag,If-None-Match

1. http://www.cnblogs.com/yjf512/p/3244882.html

## 2.美团电面

### Q.深拷贝、浅拷贝？

Answer：用递归实现深拷贝。Object.assign()实现浅拷贝。

function deepClone ( obj ) {

if( typeof obj != "object" ){

return obj ;

}

var newObj = obj.constructor === Array ? [] : {}; //开辟一块新的内存空间

for( var i in obj ) {

newObj [ i ] = deepClone ( obj [ i ] ); //通过递归实现深层的复制

}  
  
　　return newObj;

}

### Q.react：父组件与子组件传值？

1.父组件向子组件传值：<Child value={15}>，子组件this.props.value;

2.子组件向父组件传值：

### Q.react生命周期

实例化

首次实例化

getDefaultProps

getInitialState

componentWillMount

render

componentDidMount

实例化完成后的更新

getInitialState

componentWillMount

render

componentDidMount

存在期

组件已存在时的状态改变

componentWillReceiveProps

shouldComponentUpdate

componentWillUpdate

render

componentDidUpdate

销毁&清理期

componentWillUnmount

说明

生命周期共提供了10个不同的API。

1.getDefaultProps

作用于组件类，只调用一次，返回对象用于设置默认的props，对于引用值，会在实例中共享。

2.getInitialState

作用于组件的实例，在实例创建时调用一次，用于初始化每个实例的state，此时可以访问this.props。

3.componentWillMount

在完成首次渲染之前调用，此时仍可以修改组件的state。

4.render

必选的方法，创建虚拟DOM，该方法具有特殊的规则：

只能通过this.props和this.state访问数据

可以返回null、false或任何React组件

只能出现一个顶级组件（不能返回数组）

不能改变组件的状态

不能修改DOM的输出

5.componentDidMount

真实的DOM被渲染出来后调用，在该方法中可通过this.getDOMNode()访问到真实的DOM元素。此时已可以使用其他类库来操作这个DOM。

在服务端中，该方法不会被调用。

6.componentWillReceiveProps

组件接收到新的props时调用，并将其作为参数nextProps使用，此时可以更改组件props及state。

componentWillReceiveProps: function(nextProps) {

if (nextProps.bool) {

this.setState({

bool: true

});

}

}

7.shouldComponentUpdate

组件是否应当渲染新的props或state，返回false表示跳过后续的生命周期方法，通常不需要使用以避免出现bug。在出现应用的瓶颈时，可通过该方法进行适当的优化。

在首次渲染期间或者调用了*forceUpdate*方法后，该方法不会被调用

8.componentWillUpdate

接收到新的props或者state后，进行渲染之前调用，此时不允许更新props或state。

9.componentDidUpdate

完成渲染新的props或者state后调用，此时可以访问到新的DOM元素。

10.componentWillUnmount

组件被移除之前被调用，可以用于做一些清理工作，在componentDidMount方法中添加的所有任务都需要在该方法中撤销，比如创建的定时器或添加的事件监听器。

### Q.react render函数什么时候会调用？

1. 首次加载

2. setState改变组件内部state。

注意： 此处是说通过setState方法改变。

3. 接受到新的props

### Q.Vue传值。Module。。Bind –》@

### Q.css选择器权重：

内联样式，id，class，标签选择器和属性选择器一样。

### Q.call，apply，bind的区别？

Call，apply返回的是函数执行的结果，bind返回的是这个函数。

三个的使用区别：

都是用来改变函数的this对象的指向的；

第一个参数都是this要指向的对象；

都可以利用后续参数传参；

bind是返回对应函数，便于稍后调用，apply、call是立即调用；

### Q.移动端适配怎么实现？

媒体查询，rem？

https://segmentfault.com/a/1190000008767416

### Q.箭头函数与传统的js函数有什么差异？？

1.没有this，super，arguments，new.target绑定。箭头函数中的this、super、arguments及new.target这些值由外围最近一层非箭头函数决定。

2.不能通过new关键字调用。箭头函数没有[[Construct]]方法，所以不能被用作构造函数，如果通过new关键字调用箭头函数，程序会抛出错误。

3.没有原型。由于不可以通过new关键字调用箭头函数，因而没有构建原型的需求，所以箭头函数也不存在prototype这个属性。

4.不可以改变this的绑定。函数内部的this值不可以被改变，在函数的生命周期内必须保持一致。

5.不支持arguments对象。箭头函数没有arguments绑定，所以你必须通过声明参数和不定参数这两种形式访问函数的参数。

6.不支持重复的命名参数。无论在严格还是非严格模式下，箭头函数都不支持重复的命名参数；而在传统的规定中，只有在严格模式下才不能有重复的命名参数。

### Q.闭包，工作中用到没？

### Q.Js实现继承用原型链，但是对于父类中的对象继承 就会还是和父类公用一个？

Answer:在子类的构造方法中，先调用父类的构造方法，就会拷贝一份父类的属性。就不会和父类共用一个对象属性了。

**Q.垂直居中：**

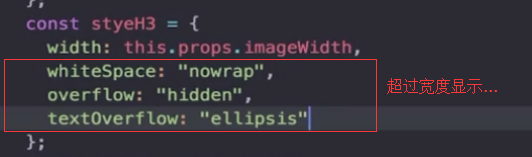
height=line-height；padding-top=padding-bottom；

Flex实现比较好。

display:flex;

align-items::center

### Q.文字超过宽度显示省略号：



### Q.识别数组的方法？

Object.prototype.toString.call(o)=='[object Array]'; 最好的

typeof arr ;是object，但arr instanceOf Array 为true。

判断arr.constructor===Array.

但上述两种方法 在框架frame传值时候会有问题。