## 第二部分：高级知识(JS高级 Jquery ES6语法 AJAX)

### new 操作符具体干了什么呢?

答:

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 。

### js 中深拷贝和浅拷贝的区别？

答:浅拷贝只复制指向某个对象的指针，而不复制对象本身，新旧对象还是共享同一块内存。 深拷贝会另外创造一个一模一样的对象，新对象跟原对象不共享内存，修改新对象不会改到原对象。

### 深拷贝有哪些方法：

1、递归拷贝

2、使用 Object.create()方法

3、jquery 有提供一个$.extend 也可以实现

4、函数库 lodash，也有提供 cloneDeep 用来实现

### 什么是回调地狱，怎么解决回调地狱？

答:由于回调函数是异步的，每一层的回调函数都需要依赖上一层的回调执行完，所以形成了 层层嵌套的关系最终形成了回调地狱。例如：定时器中再写定时器再写定时器，这种就形成了 通常所说的回调地狱。

解决:

1、避免函数的嵌套

2、模块化开发

3、使用 Promise 解决

### 对 ES6 中 Promise 的理解？

答:Promise 是异步编程的一种解决方案，比传统的解决方案——回调函数和事件——更合理 和更强大。所谓 Promise，简单说就是一个容器，里面保存着某个未来才会结束的事件（通常是 一个异步操作）的结果。从语法上说，Promise 是一个对象，从它可以获取异步操作的消息。 Promise 提供统一的 API，各种异步操作都可以用同样的方法进行处理。 特点：

1、自己身上有 all、reject、resolve 这几个方法

2、原型上有 then、catch 等方法

3、一旦建立，就无法取消，这是它的缺点

### 谈谈你对 this 理解？

答:

普通函数中：this->window

定时器中：this->window

构造函数中：this->当前实例化的对象

事件处理函数中：this->事件触发对象

在 js 中一般理解就是谁调用这个 this 就指向谁

### Js 的原型和原型链？原型链的应用？

答:每个对象都会在其内部初始化一个属性，就是 prototype(原型)，当我们访问一个对象的属性时， 如果这个对象内部不存在这个属性，那么他就会去 prototype 里找这个属性，这个 prototype 又会 有自己的 prototype，于是就这样一直找下去，也就是我们平时所说的原型链的概念。

应用：原型链是实现继承的主要方法。

### 对 json 的了解？

答:Json 指的是 js 对象表示法。

1.轻量级的数据交互格式

2.可以形成复杂的嵌套格式

3.解析非常方便

4.易于读写，占用带宽小

### jq 的在项目中的常用的方法？

答:

使用 JQuery 获取元素

JQ 获取元素的内容：$("xxxx").html()

JQ 中的事件操作

JQ 操作控件属性

JQ 操作 css ajax 操作

jq 函数级插件--$.fn.extend-添加成员函数/$.extend-添加静态方法

### $(this)和 this 的区别是什么？

答:$(this) 返回一个 jQuery 对象，你可以对它调用多个 jQuery 方法，比如用 text() 获取文本，用 val() 获取值等等。而 this 代表当前元素，它是 JavaScript 关键词中的一个，表示上下文中的当前 DOM 元素。

### jquery 对象和 dom对象是怎样转换的？

答:jquery 转 DOM 对象:jQuery 对象是一个数组对象，可以通过[index]的得到相应的 DOM 对象还可以 通过 get[index]去得到相应的 DOM 对象。DOM 对象转 jQuery 对象:$(DOM 对象)

### jq 和 zepto 区别是什么？

答:

相同点：

Zepto 最初是为移动端开发的库，是 jQuery 的轻量级替代品，因为它的 API 和 jQuery 相似，而文件更 小。Zepto 最大的优势是它的文件大小，只有 8k 多，是目前功能完备的库中最小的一个，尽管不大， Zepto 所提供的工具足以满足开发程序的需要。大多数在 jQuery 中·常用的 API 和方法 Zepto 都有， Zepto 中还有一些 jQuery 中没有的。另外，因为 Zepto 的 API 大部分都能和 jQuery 兼容，所以用起来极 其容易，如果熟悉 jQuery，就能很容易掌握 Zepto。你可用同样的方式重用 jQuery 中的很多方法，也可 以方面地把方法串在一起得到更简洁的代码，甚至不用看它的文档。

不同点:

针对移动端程序，Zepto 有一些基本的触摸事件可以用来做触摸屏交互（tap 事件、swipe 事件）， Zepto 是不支持 IE 浏览器的，这不是 Zepto 的开发者 Thomas Fucks 在跨浏览器问题上犯了迷糊，而是经 过了认真考虑后为了降低文件尺寸而做出的决定，就像 jQuery 的团队在 2.0 版中不再支持旧版的 IE（6 7 8）一样。因为 Zepto 使用 jQuery 句法，所以它在文档中建议把 jQuery 作为 IE 上的后备库。那样程序 仍能在 IE 中，而其他浏览器则能享受到 Zepto 在文件大小上的优势，然而它们两个的 API 不是完全兼 容的，所以使用这种方法时一定要小心，并要做充分的测试。

主要表现:

1、width()和 height()的区别：Zepto 由盒模型(box-sizing)决定，用.width()返回赋值的 width， 用.css('width')返回加 border 等的结果；jQuery 会忽略盒模型，始终返回内容区域的宽/高(不包含 padding、border)。

2、offset()的区别：Zepto 返回{top,left,width,height}；jQuery 返回{width,height}。

3、Zepto 无法获取隐藏元素宽高，jQuery 可以。

4、Zepto 中没有为原型定义 extend 方法而 jQuery 有。

### jQuery.fn 的 init 方法返回的 this 指的是什么对象？为什么要返回 this？

答:this 执行 init 构造函数自身，其实就是 jQuery 实例对象，返回 this 是为了实现 jQuery 的链式 操作

### jquery.extend 与 jquery.fn.extend 的区别？

答:jQuery 为开发插件提拱了两个方法，分别是：

1. jQuery.fn.extend();用来扩展 jQuery 实例

2. jQuery.extend();用来扩展 jQuery 对象本身

### Zepto 的点透问题如何解决？

答:点透主要是由于两个 div 重合，例如：一个 div 调用 show()，一个 div 调用 hide()；这个时候当点击 上面的 div 的时候就会影响到下面的那个 div；

解决办法主要有 2 种：

1.github 上有一个叫做 fastclick 的库，它也能规避移动设备上 click 事件的延迟响应， https://github.com/ftlabs/fastclick 将它用 script 标签引入页面（该库支持 AMD，于是你也可以按照 AMD 规范，用诸如 require.js 的 模块加载器引入），并且在 dom ready 时初始化在 body 上，

2.根据分析，如果不引入其它类库，也不想自己按照上述 fastclcik 的思路再开发一套东西，需要 1. 一个优先于下面的“divClickUnder”捕获的事件；2.并且通过这个事件阻止掉默认行为（下面的 “divClickUnder”对 click 事件的捕获，在 ios 的 safari，click 的捕获被认为和滚屏、点击输入框弹 起键盘等一样，是一种浏览器默认行为，即可以被 event.preventDefault()阻止的行为）。

### jQuery 中.bind() .live() .delegate() .on()的区别

答:

1、bind(type,[data],fn) 为每个匹配元素的特定事件绑定事件处理函数 $("a").bind("click",function(){alert("ok");});

2、live(type,[data],fn) 给所有匹配的元素附加一个事件处理函数，即使这个元素是以后再添加进来 的 $("a").live("click",function(){alert("ok");});

3、delegate(selector,[type],[data],fn) 指定的元素（属于被选元素的子元素）添加一个或多个事件处 理程序，并规定当这些事件发生时运行的函数 $("#container").delegate("a","click",function(){alert("ok");})

4、on(events,[selector],[data],fn) 在选择元素上绑定一个或多个事件的事件处理函数

差别：

.bind()是直接绑定在元素上 .live()则是通过冒泡的方式来绑定到元素上的。更适合列表类型的，绑定到 document DOM 节点上。 和.bind()的优势是支持动态数据。 .delegate()则是更精确的小范围使用事件代理，性能优于.live() .on()则是最新的 1.9 版本整合了之前的三种方式的新事件绑定机制

### json 字符串和 json 对象怎么相互转化？

答:

JSON.parse() 从一个字符串中解析出 json 对象

JSON.stringify() 从一个对象中解析出字符串 在前后数据交互中经常使用，必须要记住。

### http 和 https 的区别是什么？

答:http 和 https 使用的是完全不同的连接方式,用的端口也不一样,前者是 80,后者是 443。 HTTPS 是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版。 由于 https 要还密钥和确认加密算法的过程，所以更安全

### get 和 post 的区别？

答:

客户端对服务器的请求常用类型主要有四种：

GET（从服务器获取）

POST（向服务器发送请求数据）

PUT（更新）

POST 和 GET 区别 GET 的所有参数全部包装在 URL 中，明文显示，且服务器的访问日志会记录，非常不安全 POST 的 URL 中只有资源路径，不包含参数，参数封装在二进制的数据体中，服务器也不会记录参数，相对安全。所有涉及用户隐私的数据都要用 POST 传输。 GET：不同的浏览器和服务器不同，一般限制在 2~8K 之间，更加常见的是 1k 以内 POST 方法提交的数据比较大，大小靠服务器的设定值限制，PHP 默认是 2M

### jsonp 的原理，有什么优缺点？

答:ajax 请求受同源策略影响，不允许进行跨域请求，而 script 标签 src 属性中的链接却可以访问跨域的 js 脚本，利用这个特性，服务端不再返回 JSON 格式的数据，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。

Jsonp 优点:

完美解决在测试或者开发中获取不同域下的数据,用户传递一个 callback 参数给服务端，然后服务 端返回数据时会将这个 callback 参数作为函数名来包裹住 JSON 数据，这样客户端就可以随意定制自己 的函数来自动处理返回数据了。

Jsonp 缺点:

Jsonp 只支持 get 请求而不支持 post 请求,也即是说如果想传给后台一个 json 格式的数据,此时问题 就来了,浏览器会报一个 http 状态码 415 错误,告诉你请求格式不正确.

### 跨域的几种解决方式

答:

1.通过 jsonp 跨域

2. CORS 核心思想：在服务器端通过检查请求头部的 origin，从而决定请求应该成功还是失败。具体的方法 是在服务端设置 Response Header 响应头中的 Access-Control-Allow-Origin 为对应的域名，实现了 CORS（跨域资源共享），这里出于在安全性方面的考虑就是尽量不要用 \*，但对于一些不重要的 数据则随意。

3.通过反向代理

### 同源策略

答:

1. 同源，就是指两个页面具有相同的协议，主机（域名），端口，浏览器会对不同源的脚本或者文本的访问方式进行的限制

2. 页面中的链接，重定向以及表单提交不会受到同源策略的限制，允许跨域资源嵌入

### jq 中 ajax 请求的步骤？怎么解决跨域的？

答:

$.ajax({

Type:”get/post”,

Url:””,

dataType:”jsonp”, 利用 jsonp 进行跨域

Data:{},

async: false，--默认为 true 异步 false 同步

cache: false,--默认为 true 缓存 false 不缓存 beforeSend:fucntion(){},

Success:function(){},

Errorr:function(){}

})

### 简述 ajax 的交互原理

答:Ajax 的原理简单来说通过 XmlHttpRequest 对象来向服务器发异步请求，从服务器获得数据，然后 用 javascript 来操作 DOM 而更新页面。这其中最关键的一步就是从服务器获得请求数据

1、创建 XMLHttpRequest 对象

2、建立连接，设置为 GET 发送：xmlHttp.open("GET", "/ajax/getHint.jsp?q=" + txtValue, true);

3、发送数据：xmlHttp.send();

4、注册回调方法：xmlHttp.onreadystatechange = ajaxCallback;

5、执行回调： function ajaxCallback(){ //响应就绪时 if(xmlHttp.readyState == 4 && xmlHttp.status == 200){ searchInput.value = xmlHttp.responseText; } }

### 以及同步和异步的区别

答:

同步：同步任务指的是，在主线程上排队执行的任务，只有前一个任务执行完毕，才能执行后一个任务。

异步：不进入主线程、而进入"任务队列"（task queue）的任务，只有等主线程任务执行完毕，"任务队 列"开始通知主线程，请求执行任务，该任务才会进入主线程执行。

### ajax 出现错误怎么调试？

JQuery 使我们在开发 Ajax 应用程序的时候提高了效率，减少了许多兼容性问题，我们在 Ajax 项目 中，遇到 ajax 异步获取数据出错怎么办，我们可以通过捕捉 error 事件来获取出错的信息。 发送 error 可能有下面两张引起的，或者其他程序问题，需要我们认真仔细。

1、data:"{}", data 为空也一定要传"{}"；不然返回的是 xml 格式的。并提示 parsererror.

2、parsererror的异常和Header 类型也有关系。及编码header('Content-type: text/html; charset=utf8');

### ajax 的缺点（答出三点即可）

答:

1、ajax 不支持浏览器 back 按钮。

2、安全问题 AJAX 暴露了与服务器交互的细节。

3、对搜索引擎的支持比较弱。

4、破坏了程序的异常机制。

5、不容易调试。

### WEB 应用从服务器主动推送 Data 到客户端有那些方式？

答:html5 websoket, WebSocket 通过 Flash XHR 长时间连接 XHR Multipart Streaming 不可见的 Iframe <script>标签的长时间连接(可跨域)

### 请写出至少 5 种常见的 http 状态码以及代表的意义

答:5种常见的 http 状态码以及代表的意义如下：

1.200（OK）：请求已成功，请求所希望的响应头或数据体将随此响应返回。

2.303（See Other）：告知客户端使用另一个 URL 来获取资源。

3.400（Bad Request）：请求格式错误。1)语义有误，当前请求无法被服务器 理解。除非进行修改，否则客户端不应该重复提交这个请求；2)请求参数有误。

4.404（Not Found）：请求失败，请求所希望得到的资源未被在服务器上发现。

### 程序中捕获异常的方法？

window.error

try{}catch(){}finally{}

### JQuery一个对象可以同时绑定多个事件，这是如何实现的？

答:jQuery可以给一个对象同时绑定多个事件，低层实现方式是使用addEventListner或attachEvent兼容不同的浏览器实现事件的绑定，这样可以给同一个对象注册多个事件。

### 知道什么是webkit么? 知道怎么用浏览器的各种工具来调试和debug代码么?

答:Webkit是浏览器引擎，包括html渲染和js解析功能，手机浏览器的主流内核，与之相对应的引擎有Gecko（Mozilla Firefox 等使用）和Trident（也称MSHTML，IE 使用）。 对于浏览器的调试工具要熟练使用，主要是页面结构分析，后台请求信息查看，js调试工具使用，熟练使用这些工具可以快速提高解决问题的效率

### 在Javascript中什么是伪数组？如何将伪数组转化为标准数组？

答: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。

### eval是做什么的？

答:它的功能是把对应的字符串解析成JS代码并运行；应该避免使用eval，不安全，非常耗性能（2个步骤，一次解析成js语句，一次执行）

35.javascript 代码中的"use strict";是什么意思 ? 使用它区别是什么？

答:意思是使用严格模式，使用严格模式，一些不规范的语法将不再支持

### Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？

答:HasOwnProperty

### 怎么实现一个类。怎么实例化这个类

答:严格来讲js中并没有类的概念，不过js中的函数可以作为构造函数来使用，通过new来实例化，其实函数本身也是一个对象。

### JavaScript中的作用域与变量声明提升？

答:理解JavaScript的预解析机制，js的运行主要分两个阶段：js的预解析和运行，预解析阶段所有的变量声明和函数定义都会提前，但是变量的赋值不会提前

### 如何编写高性能的Javascript？

答:

1.使用 DocumentFragment 优化多次 append

2.通过模板元素 clone ，替代 createElement

3.使用一次 innerHTML 赋值代替构建 dom 元素

4.使用 firstChild 和 nextSibling 代替 childNodes 遍历 dom 元素

5.使用 Array 做为 StringBuffer ，代替字符串拼接的操作

6.将循环控制量保存到局部变量

7.顺序无关的遍历时，用 while 替代 for

8.将条件分支，按可能性顺序从高到低排列

9.在同一条件子的多（ >2 ）条件分支时，使用 switch 优于 if

10.使用三目运算符替代条件分支

11.需要不断执行的时候，优先考虑使用 setInterval

### jQuery.fn的init方法返回的this指的是什么对象？为什么要返回this？

答:this执行init构造函数自身，其实就是jQuery实例对象，返回this是为了实现jQuery的链式操作

### jquery中如何将数组转化为json字符串，然后再转化回来？

答$.parseJSON('{"name":"John"}');

### jQuery 的属性拷贝(extend)的实现原理是什么，如何实现深拷贝？

答:递归赋值

### jquery.extend 与 jquery.fn.extend的区别？

答:Jquery.extend用来扩展jQuery对象本身；jquery.fn.extend用来扩展jQuery实例

### Jquery与jQuery UI有啥区别？

答:jQuery是操作dom的框架，jQueryUI是基于jQuery做的一个UI组件库

### jQuery和Zepto的区别？各自的使用场景？

答:jQuery主要用于pc端，当然有对应的jQuerymobile用于移动端，zepto比jQuery更加小巧，主要用于移动端

### 针对 jQuery 的优化方法？

答:

1.优先使用ID选择器

2.在class前使用tag(标签名)

3.给选择器一个上下文

4.慎用 .live()方法（应该说尽量不要使用）

5.使用data()方法存储临时变量

### let const 的区别

答: let const 块级作用域 没有变量提升 const 不能修改

### 箭头函数

答: (形参) => {函数体} 箭头函数特点 没有this指向

### 使用展开运算符 合并数组和对象

答:

var arr1 = [11, 22, 33]

var arr2 = [44, 55, 66, ...arr1] // [44, 55, 66, 11, 22, 33]

var obj = {name: '宋公明'}

var obj1 = {age: 18}

var obj2 = {

desc: '及时雨',

...obj,

...obj1

}

console.log(obj2)

### 解构赋值

答:

对象解构赋值

var myobj = {name1: {a: 10},age: 18}

var name1 = myobj.name1

var {name1: ss, age} = myobj // 解构过程中通过: 可以修改解构的名字

console.log(ss, age)

数组解构赋值

var myarr = [11, 22, 33]

var [a, b, c, d] = myarr

var a,b,c,d;

console.log(a, b, c, d) // 11 22 33 undefined \*/

Set数据解构

var arr = [36, 99, 66, 99, 44, 99, 21, 36]

var result = new Set(arr)

console.log([...result])

console.log(Array.from(result)) // Array.from将类数组转为真正的数组 \*/

### 字符串操作的方法

答:

var str = `一点寒芒先到，随后枪出如龙`

1.获取字符串第一个字符

console.log(str[0])

2.判断字符串是否以 某个字符 开始或结束

console.log(str.startsWith('点')) // str是否以 点 是就返回 true 否就返回false

console.log(str.endsWith('龙')) // str是否以 点 是就返回 true 否就返回false

3.往字符串填充 padStart(填充完之后多少个字符, 要填充的字符)

console.log(str.padStart(20, 'abcd'))

console.log(str.padEnd(20, 'abcd'))

console.log(str)

str.includes('a') // str中是否包含a字符

4.包含

console.log(str.includes('芒'))

### Promise 使用Promise封装一个ajax

答:

function myAjax () {

return new Promise((resolve, reject) => {

// 这里开始封装ajax

// })

// }

// myAjax().then(成功回调).catch(失败回调)

// 10.async await

// 10.1 以前的

/\* function myTest () {

console.log(666666)

return '我是一块小小鸟'

}

console.log(myTest()) // 1. 执行函数体代码 2. 获取返回值 \*/

/\* async function myTest () {

console.log(666666)

return '我是一块小小鸟' // return 相当于resolve(return的值)

}

// console.log(myTest()) // romise {<resolved>: "我是一块小小鸟"}

myTest().then(result => {

console.log(result, 1111)

}) \*/

### JS如何检测基本数据类型和复杂数据类型

解答:

1. typeof 检测基本数据类型
2. Array.isArray(); 浏览器兼容性：IE9
3. 变量 instanceof object 检测复杂数据类型

### 例举3种强制类型转换和2种隐式类型转换?

解答:

强制类型转换：自己通过函数来进行数据类型转换  
举例：（parseInt,parseFloat,Number()）

隐式类型转换：JS引擎自动帮我们转换的  
举例：==、 console.log()、 alert() 、if() 、+-\*/

### split() join() 的区别

解答:

split()将字符串按照指定的字符分割成一个数组，并返回  
join()将数组用指定的字符连接成一个字符串，并返回

### 数组方法pop() push() unshift() shift()

解答：**栈方法：  
push()尾部添加，返回 数组长度  
pop()尾部删除，返回 被删除的元素**

队列方法：  
unshift()头部添加 ，返回 数组长度  
shift()头部删除，返回被删除的元素

### 事件绑定和普通事件有什么区别？

解答:

普通事件:

1. 如果说给同一个元素绑定了两次或者多次相同类型的事件，那么后面的绑定会覆盖前面的绑定  
   2、不支持DOM事件流 事件捕获阶段目标元素阶段=>事件冒泡阶段

事件绑定:

1. 如果说给同一个元素绑定了两次或者多次相同类型的事件，所有的绑定将会依次触发  
   2、支持DOM事件流的  
   3、进行事件绑定传参不需要on前端

### IE和DOM事件流的区别?

解答:

比较attachEvent和addEventListener：

1、attachEvent只支持事件冒泡 addEventListener既支持事件冒泡，也支持事件捕获

2、参数：attachEvent事件类型需要on前缀 addEventListener事件类型不需要on前缀

3、如果使用attachEvent对一个元素的目标阶段绑定了多次事件，那么会按照绑定顺序的相反顺序进行触发

如果使用addEventListener对一个元素的目标阶段绑定了多次事件，那么会按照绑定顺序进行触发

### call和apply，bind的区别

解答：call和apply的用法

call和apply和bind相同点：改变函数中this的指向

不同点：函数参数的传递形式  
call将函数参数依次传入  
apply将函数参数用一个数组的形式传入

Bind函数参数传入不同

### JavaScript this、闭包、作用域?

解答:

this：指向调用上下文

作用域：定义一个函数就开辟了一个局部作用域，整个js执行环境有一个全局作用域

闭包：一个函数可以访问其他函数中的变量（闭包是一个受保护的变量空间）

### 事件委托的原理是什么

解答：利用事件冒泡的原理，将事件绑定在父容器中，让父容器代为触发

**10.如何阻止事件冒泡和默认事件**

阻止事件冒泡：

IE9+ FF Chrome：e. stopPropagation();

window.event.cancelBubble=true;//ie9之前

默认行为：html标签所具有的默认行为，比如：

a、点击a标签，就会默认跳转到指定的页面

b、点击submit按钮，就会自动提交表单

适用场景：

1、异步操作

2、提交表单之前对表单进行一些基本的验证，比如邮箱是否合法，用户名是不是满足指定的格式

为了不让a点击之后跳转，我们就要给他的点击事件进行阻止

3、文本框获得焦点

阻止默认行为：

IE9之前：window.event.returnValue=false;

IE9+ FF Chrome： e.preventDefault();

### 添加 删除 替换 插入到某个节点的方法

解答:

obj.appendChild() //添加  
obj.insertBefore(newElement, referenceElement ) /插入

obj.replaceChild(newChild, oldChild)//替换

obj.removeChild(child)//删除

### javascript的本地对象，内置对象和宿主对象

解答:

本地对象为Array RegExp（正则表达式）等可以new实例化  
内置对象为global Math 等不可以实例化的  
宿主对象为浏览器自带的document,window 等

1. **javascript的同源策略**

解答:

一段脚本只能读取来自于同一来源的窗口和文档的属性，这里的同一来源指的是主机名、协议和端口号的组合

http,ftp:协议

关键词解释：

主机名：localhost、www.baidu.com

协议：http https ftp

端口：一个网站对应着一个端口， http协议的默认端口：80

https协议的默认端口是8083

同源策略带来的麻烦：ajax在不同域名下的请求无法实现，

如果说想要请求其他来源的js文件，或者json数据，那么可以通过jsonp来解决

### 数组去重的方法有那几种?

解答:

1. set数据类型，实例化进行去重
2. For循环结合indexOf进行去重
3. 利用数组原型上面的 filter 和 includes或者 forEach 和 includes方法可以进行数组去重

function unique7(arr) {

var newArr = []

array.forEach(item => {

return newArr.includes(item) ? '' : newArr.push(item)

});

return newArr

}

console.log(unique7([1, 1, 2, 3, 5, 3, 1, 5, 6, 7, 4]));

function unique6(arr) {

var newArr = []

newArr = arr.filter(function (item) {

return newArr.includes(item) ? '' : newArr.push(item)

})

return newArr

}

console.log(unique6([1, 1, 2, 3, 5, 3, 1, 5, 6, 7, 4]));

### JavaScript是一门什么样的语言，它有哪些特点？

解答：

弱类型、脚本语言、面向对象、

没有标准答案。

运行环境：JS引擎（v8(Chrome)/SpiderMonkey(FireFox)/JavaScriptCore(Safari)

/Chakra(IE)）

语言特性：

1、面向对象：原型继承、构造函数、原型链

2、动态语言：弱类型语言

//动态语言的特性 var num=10;//num是一个数字类型 num="jim";//此时num又变成一个字符串类型

//我们把一个变量用来保存不同数据类型的语言称之为一个动态语言 //静态语言：c# java c c++ OC //int a; //静态语言在声明一个变量就已经确定了这个变量的数据类型， // 而且在任何时候都不可以改变他的数据类型

### JavaScript的数据类型都有什么？

基本数据类型：number、string、boolean、undefined、null  
复杂数据类型：Object(Array,Date,RegExp,Function)

### 哪些操作会造成内存泄漏？

解答:

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。

1. setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。

2. 闭包

3. 控制台日志

### for in和for of的区别

解答:

1.推荐在循环对象属性的时候，使用for...in,在遍历数组的时候的时候使用for...of  
2.for...in循环出的是key，for...of循环出的是value  
3.注意，for...of是ES6新引入的特性。修复了ES5引入的for...in的不足  
4.for...of不能循环普通的对象，需要通过和Object.keys()搭配使用

### 什么是函数柯里化？

解答:

是把接受多个参数的函数变换成接受一个单一参数（最初函数的第一个参数）的函数，并且返回接受余下的参数而且返回结果的新函数的技术。

### 解释jsonp的原理，以及为什么不是真正的ajax?

解答:

动态创建script标签，回调函数  
Ajax是页面无刷新请求数据操作

### js延迟加载的方式有哪些？

defer和async、动态创建DOM方式（创建script，插入到DOM中，加载完毕后callBack）、按需异步载入js

### 请说出三种减低页面加载时间的方法?

解答:

1、压缩css、js文件  
2、合并js、css文件，减少http请求  
3、外部js、css文件放在最底下  
4、减少dom操作，尽可能用变量替代不必要的dom操作

### 什么是FOUC？你如何来避免FOUC？

解答:

由于css引入使用了@import 或者存在多个style标签以及css文件在页面底部引入使得css文件加载在html之后导致页面闪烁、花屏  
用link加载css文件，放在head标签里面

### http协议属于七层协议中的哪一层，下一层是什么

解答:

七层结构：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层  
tcp属于传输层；http属于应用层。表现层

### websocket长连接原理是什么?

解答：

Websocket是一个持久化的协议，相对于HTTP这种非持久的协议来说。

### UDP和TCP协议的概念？

解答:

TCP是事先为所发送的数据开辟出连接好的通道，然后再进行数据发送；而UDP则不为IP提供可靠性、流控或差错恢复功能。一般来说，TCP对应的是可靠性要求高的应用，而UDP对应的则是可靠性要求低、传输经济的应用。

### 全双工通讯协议的概念？

解答:

全双工是通讯传输的一个术语。通信允许数据在两个方向上同时传输，他在能力上相当于两个单工通信方式的结合。全双工指可以同时进行信号的双向传输。

全双工是：例如我们使用的手机就是全双工，在同一时刻两个用户可以同时给对方传送数据

半双工：例如我们使用的对讲机，当A方按住通话按钮才可以向B方传送数据，B方也是，在同一时刻只有一个用户能够传送数据（A/用户都可以传递信息，但是不能够同时传递）

单工：例如我们看电视时，我们只能接收对方发送的信息，不能够给对方传递信息；

### WebSocket和Socket的区别是什么？

解答:

Socket是应用层与TCP/IP协议通信的中间软件抽象层，它是一组接口。而WebSocket则不同，它是一个完整的应用层协议，包含一套标准的API。

### 26说一下什么是Http协议？

解答:对器客户端和 服务器端之间数据传输的格式规范，格式简称为“超文本传输协议”。

### 什么是Http协议无状态协议？怎么解决Http协议无状态协议？

解答:

无状态协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息

无状态协议解决办法： 通过1、Cookie 2、通过Session会话保存。

### Http协议有什么组成？

解答:

请求报文包含三部分：

请求行：包含请求方法、URI、HTTP版本信息

请求首部字段

请求内容实体

响应报文包含三部分：

状态行：包含HTTP版本、状态码、状态码的原因短语

响应首部字段

响应内容实体

### Http与Https优缺点？

解答:

通信使用明文不加密，内容可能被窃听，也就是被抓包分析。

不验证通信方身份，可能遭到伪装

无法验证报文完整性，可能被篡改

HTTPS就是HTTP加上加密处理（一般是SSL安全通信线路）+认证+完整性保护

### Http优化和http协议有哪些区别？

解答:

利用负载均衡优化和加速HTTP应用

利用HTTP Cache来优化网站

1. 支持客户/服务器模式；2、简单快速；3、灵活；4、无连接；5、无状态。

### var、let、const之间的区别?

解答:

var声明变量可以重复声明，而let不可以重复声明  
var是不受限于块级的，而let是受限于块级  
var会与window相映射（会挂一个属性），而let不与window相映射  
var可以在声明的上面访问变量，而let有暂存死区，在声明的上面访问变量会报错  
const声明之后必须赋值，否则会报错  
const定义不可变的量，改变了就会报错  
const和let一样不会与window相映射、支持块级作用域、在声明的上面访问变量会报错

### 使用箭头函数应注意什么？

解答:

1. 用了箭头函数，this就不是指向window，而是父级（指向是可变的）  
   （2）不能够使用arguments对象  
   （3）不能用作构造函数，这就是说不能够使用new命令，否则会抛出一个错误  
   （4）不可以使用yield命令，因此箭头函数不能用作 Generator 函数

### 介绍下 Set、Map的区别？

解答:应用场景Set用于数据重组，Map用于数据储存Set：　  
（1）成员不能重复  
（2）只有键值没有键名，类似数组  
（3）可以遍历，方法有add, delete,has  
Map:  
（1）本质上是健值对的集合，类似集合  
（2）可以遍历，可以跟各种数据格式转换

### Promise构造函数是同步执行还是异步执行，那么 then 方法呢？

解答:

promise构造函数是同步执行的，then方法是异步执行的

### 如何把es6,es7高级语法进行转换?

解答:

使用babel转换器，或者webpack，gulp进行转换

### 什么是AMD、CMD区别？

解答:

CMD 推崇依赖就近; AMD 推崇依赖前置  
CMD 是延迟执行; AMD 是提前执行  
CMD性能好,因为只有用户需要的时候才执行; AMD用户体验好,因为没有延迟,依赖模块提前执行了

### 请说出es6中常用的数组方法?

解答:

forEach() 循环整个数组相当于for循环，带索引和值

concat() //合并多个数组，返回合并后的新数组，原数组没有变化。

filter() //返回一个新数组，包含通过callback函数测试的所有元素。

map() //返回新数组

every()和some() //every()和some()方法是数组的逻辑判定

...扩展运算符

findIndex()返回第一个符合条件的数组成员的位置，如果所有成员都不符合条件，则返回-1

include()方法

判断数组是否包含给定的值，有返回true，没有返回false，第二个参数表示搜索的起始位置

### es6中常用的对象的方法?

解答:

1.Object.is() 为解决一些异常的比较问题js中新增了Object的is方法

Object.is(NaN, NaN); //true

Object.is(5, "5"); //false

2.Object.assign() 解决了两个对象间浅拷贝的问题：（深拷贝）

const target = { a: 1 };

const source1 = { b: 2 };

const source2 = { c: 3 };

Object.assign(target, source1, source2);

3.Object.values(obj) 获取obj的value名称并储存为一个数组，返回此数组

4.Object.keys(obj) 获取obj的key名称并储存为一个数组，返回此数组

### 箭头函数和普通函数的区别?

解答：

相比普通函数更简洁的语法

没有this

不能使用new

不绑定arguments，用rest参数...解决

使用call()和apply()调用

捕获其所在上下文的 this 值，作为自己的 this 值

箭头函数没有原型属性

不能简单返回对象字面量

箭头函数不能当做Generator函数,不能使用yield关键字

箭头函数不能换行

### Class和普通构造函数的区别？

解答:

1. Class 在语法上更加贴合面向对象的写法

2. Class在实现继承上更加易读、易理解

3. 更易于写java等后端语言

4.本质还是语法糖，使用prototype

### 什么是原型链？

解答:

Javascript是面向对象的，每个实例对象都有一个\_\_proto\_属性，该属性指向它原型对象，这个实例对象的构造函数有一个原型属性prototype，与实例的\_\_proto\_\_属性指向同一个对象。当一个对象在查找一个属性的时，自身没有就会根据\_\_proto\_\_ 向它的原型进行查找，如果都没有，则向它的原型的原型继续查找，直到查到Object.prototype.\_proto\_为nul，这样也就形成了原型链。

### 如何查找构造函数和原型中的属性？

解答:

构造函数.prototype  查看构造函数的原型属性

实例对象.\_\_proto\_\_ 查看实例对象的构造函数的原型

实例对象.\_\_proto\_\_.constructor 查看实例对象的构造函数

### 什么是闭包?

解答:

高级程序设计三中:闭包是指有权访问另外一个函数作用域中的变量的函数.可以理解为(能够读取其他函数内部变量的函数)

闭包的作用: 正常函数执行完毕后,里面声明的变量被垃圾回收处理掉,但是闭包可以让作用域里的 变量,在函数执行完之后依旧保持没有被垃圾回收处理掉

### 请说说你了解哪些算法?

解答:

1. 递归算法，斐波那契数列
2. 数组去重
3. 冒泡排序
4. 贪心算法
5. Diff算法
6. 排序算法

7.二叉树和二叉查找树

### 谈一谈你所理解的继承？

解答:

借用构造函数继承

function Parent0(){

this.name = "parent0";

this.colors = ["red","blue","yellow"];}function Child0(){

Parent0.call( this ); // 或apply

this.type = "child0";}

new Child0().name; // Parent0

new Child0().name; // Parent0new Child0().colors; // (3) ["red", "blue", "yellow"]

原型链式继承（借用原型链实现继承）

function Parent1(){

this.name = "parent1";

this.colors = ["red","blue","yellow"];

}

function Child1(){

this.name = "child1";

}

Child1.prototype = new Parent1();

Parent1.prototype.sex = "男";

Parent1.prototype.say = function() {

console.log(" Oh,My God! ");

}

new Child1().sex; // 男

new Child1().say(); // Oh,My God!

var s1 = new Child1();

s1.colors.push("black");

var s2 = new Child1();

s1.colors; // (4) ["red", "blue", "yellow", "balck"]

s2.colors; // (4) ["red", "blue", "yellow", "balck"]

ES6中classs继承

class Parent {

}

class Child1 extends Parent {

constructor(x, y, colors) {

super(x, y); // 调用父类的constructor(x, y)

this.colors = colors;

}

toString() {

return this.colors + ' ' + super.toString(); // 调用父类的toString()

}

}

constructor方法和toString方法之中，都出现了super关键字，它在这里表示父类的构造函数，用来新建父类的this对象。

子类必须在constructor方法中调用super方法，否则新建实例时会报错。如果子类没有定义constructor方法，这个方法会被默认添加，不管有没有显式定义，任何一个子类都有constructor方法。

ES5 的继承，实质是先创造子类的实例对象this，然后再将父类的方法添加到this上面（Parent.apply(this)）。ES6 的继承机制完全不同，实质是先创造父类的实例对象this（所以必须先调用super方法），然后再用子类的构造函数修改this