**目录**

1、谈谈你对Ajax的理解？(概念、特点、作用)   
2、说说你对延迟对象deferred的理解?   
3、什么是跨域，如何实现跨域访问?   
4、为什么要使用模板引擎？   
5、JavaScript是一门什么样的语言，它有哪些特点？   
6、JavaScript的数据类型有哪些？   
7、已知ID的Input输入框，如何获取这个输入框的输入值？(不使用第三方框架)   
8、根据你的理解,请简述JavaScript脚本的执行原理?   
9、DOM操作怎样添加、移除、移动、复制、创建和查找节点?   
10、说说你对json的理解?   
11、ionic和angularjs的区别?   
12、谈谈你对闭包的理解?   
13、谈谈你This对象的理解?   
14、JavaScript对象的几种创建方式?   
15、get和post的区别,何时使用post   
16、null和undefined的区别？   
17、请写出js内存泄漏的问题?   
18、哪些地方会出现css阻塞，哪些地方会出现js阻塞？   
19、对比Flash与ajax哪个好？   
20、请你解释一下事件冒泡机制   
21、请你说说split()与join() 函数的区别?   
22、说说你对Promise的理解?   
23、谈谈你对Javascript垃圾回收机制的理解？   
24、说说你对原型（prototype）理解?   
25、typeof与instanceof的区别是什么？   
26、说说你对node.js的理解   
27、NPM(包管理器)作用是什么?

**1、谈谈你对Ajax的理解？(概念、特点、作用)**

AJAX全称为“Asynchronous JavaScript And XML”（异步JavaScript和XML） 是指一种创建交互式网页应用的开发技术、改善用户体验，实现无刷新效果。

**优点**

a、不需要插件支持   
b、优秀的用户体验   
c、提高Web程序的性能   
d、减轻服务器和带宽的负担

**缺点**

a、破坏浏览器“前进”、“后退”按钮的正常功能，可以通过简单的插件弥补   
b、对搜索引擎的支持不足

**2、说说你对延迟对象deferred的理解?**

deferred对象是从jQuery 1.5.0版本开始引入的一个新功能。

**a、什么是deferred对象**

开发网站的过程中，我们经常遇到某些耗时很长的javascript操作。其中，既有异步的操作（比如ajax读取服务器数据），也有同步的操作（比如遍历一个大型数组），它们都不是立即能得到结果的。   
通常的做法是，为它们指定回调函数（callback）。即事先规定，一旦它们运行结束，应该调用哪些函数。   
但是，在回调函数方面，jQuery的功能非常弱。为了改变这一点，jQuery开发团队就设计了deferred对象。   
简单说，deferred对象就是jQuery的回调函数解决方案。在英语中，defer的意思是”延迟”，所以deferred对象的含义就是”延迟”到未来某个点再执行。   
它解决了如何处理耗时操作的问题，对那些操作提供了更好的控制，以及统一的编程接口。

**b、它的主要功能，可以归结为四点：**

(1)、实现链式操作   
(2)、指定同一操作的多个回调函数   
(3)、为多个操作指定回调函数   
(4)、普通操作的回调函数接口

**3、什么是跨域，如何实现跨域访问?**

跨域是指不同域名之间相互访问。   
JavaScript同源策略的限制，A域名下的JavaScript无法操作B或是C域名下的对象

**实现：**

(1)、JSONP跨域：利用script脚本允许引用不同域下的js实现的，将回调方法带入服务器，返回结果时回调。   
(2)、跨域资源共享（CORS）   
跨域资源共享（CORS）是一种网络浏览器的技术规范，它为Web服务器定义了一种方式，允许网页从不同的域访问其资源。

**CORS与JSONP相比:**

a、 JSONP只能实现GET请求，而CORS支持所有类型的HTTP请求。   
b、 使用CORS，开发者可以使用普通的XMLHttpRequest发起请求和获得数据，比起JSONP有更好的错误处理。   
c、 JSONP主要被老的浏览器支持，它们往往不支持CORS，而绝大多数现代浏览器都已经支持了CORS。

**4、为什么要使用模板引擎？**

a、模板引擎（这里特指用于Web开发的模板引擎）是为了使用户界面与业务数据（内容）分离而产生的，它可以生成特定格式的文档，用于网站的模板引擎就会生成一个标准的HTML文档。   
b、在一些示例中javascript有大量的html字符串，html中有一些像onclick样的javascript，这样javascript中有html，html中有javascript，代码的偶合度很高，不便于修改与维护，使用模板引擎可以解决问题。

**5、JavaScript是一门什么样的语言，它有哪些特点？**

JavaScript 是一种脚本语言，官方名称为 ECMAScript（因定义语言的标准为 ECMA-262）。   
JS 的主要特点：   
a、语法类似于常见的高级语言，如 C 和 Java；   
b、脚本语言，不需要编译就可以由解释器直接运行；   
c、 变量松散定义，属于弱类型语言；   
d、面向对象的。   
JS 最初是为网页设计而开发的，现在也是Web 开发的重要语言。它支持对浏览器（浏览器对象模型，BOM）和HTML 文档（文档对象模型，DOM）进行操作而使网页呈现动态的交互特性。   
严格的说，JS只是ECMAScript 的一种实现，是ECMAScript和BOM、DOM组成的一种Web 开发技术。

**6、JavaScript的数据类型有哪些？**

基本数据类型：字符串 String、数字 Number、布尔Boolean   
复合数据类型：数组 Array、对象 Object   
特殊数据类型：Null 空对象、Undefined 未定义

**7、已知ID的Input输入框，如何获取这个输入框的输入值？(不使用第三方框架)**

document.getElementById("ID").value

**8、根据你的理解,请简述JavaScript脚本的执行原理?**

JavaScript是一种动态、弱类型、基于原型的语言，通过浏览器可以直接执行。   
当浏览器遇到\<script> 标记的时候，浏览器会执行之间的javascript代码。嵌入的js代码是顺序执行的，每个脚本定义的全局变量和函数，都可以被后面执行的脚本所调用。 变量的调用，必须是前面已经声明，否则获取的变量值是undefined。

**9、DOM操作怎样添加、移除、移动、复制、创建和查找节点?**

**（1）创建新节点**

createDocumentFragment() //创建一个DOM片段

createElement() //创建一个具体的元素

createTextNode() //创建一个文本节点

**（2）添加、移除、替换、插入**

appendChild()

removeChild()

replaceChild()

insertBefore() //在已有的子节点前插入一个新的子节点

**（3）查找**

getElementsByTagName() //通过标签名称

getElementsByName() //通过元素的Name属性的值(IE容错能力较强，会得到一个数组，其中包括id等于name值的)

getElementById() //通过元素Id，唯一性

**10、说说你对json的理解?**

**回答一：**

a、JSON对象：以“{”开始，以“}”结束，里面则是一系列的键（key）值（value）对，键和值用“:”分开，每对键值对之间用“,”分开。参考以下语法结构：{key1:value1,key2:value2,key3:value3…}其中键（key）是字符串，而值（value）可以是字符串，数值，true,false,null,对象或数组，也就是说某个键（key）的值（value）可以是一个数组，数组里面又是一些JSON对象，这种表示稍微复杂一些，但是参照这些理解可以很容易分辨出来。   
b、JSON数组：以”[”开始，”]”结束，如同程序语言一样，例如C#，Button[] btnArray,则BtnArray是一个Button类型的数组，里面就存放Button类型的对象，那么JSON数组也一样，里面存放的也是JSON对象.

**回答二：**

a、JSON 指的是 JavaScript 对象表示法（JavaScript Object Notation）   
b、JSON 是轻量级的文本数据交换格式，并不是编程语言   
c、JSON 独立于语言存在   
d、JSON 具有自我描述性，更易理解   
e、JSON 可以将 JavaScript 对象中表示的一组数据转换为字符串，然后就可以在函数之间轻松地传递这个字符串，或者在异步应用程序中将字符串从 Web 客户机传递给服务器端程序。这个字符串看起来有点儿古怪，但是JavaScript很容易解释它，而且 JSON 可以表示比”名称 / 值对”更复杂的结构。例如，可以表示数组和复杂的对象，而不仅仅是键和值的简单列表

**回答三：**

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。   
它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小.   
json简单说就是javascript中的对象和数组，所以这两种结构就是对象和数组两种结构，通过这两种结构可以表示各种复杂的结构。   
(1)、对象：对象在js中表示为“{}”括起来的内容，数据结构为 {key：value,key：value,…}的键值对的结构，在面向对象的语言中，key为对象的属性，value为对应的属性值，所以很容易理解，取值方法为 对象.key 获取属性值，这个属性值的类型可以是 数字、字符串、数组、对象几种。   
(1)、数组：数组在js中是中括号“[]”括起来的内容，数据结构为 [“java”,”javascript”,”vb”,…]，取值方式和所有语言中一样，使用索引获取，字段值的类型可以是数字、字符串、数组、对象几种。   
经过对象、数组2种结构就可以组合成复杂的数据结构了。

**11、ionic和angularjs的区别?**

a、ionic是一个用来开发混合手机应用的，开源的，免费的代码库。可以优化html、css和js的性能，构建高效的应用程序，而且还可以用于构建Sass和AngularJS的优化。   
b、AngularJS通过新的属性和表达式扩展了HTML。AngularJS可以构建一个单一页面应用程序（SPAs：Single Page Applications）。   
c、Ionic是一个混合APP开发工具，它以AngularJS为中间脚本工具(称为库，似乎又不恰当)，所以，你如果要使用Ionic开发APP，就必须了解AngularJS。

**12、谈谈你对闭包的理解?**

(1)、使用闭包主要是为了设计私有的方法和变量。闭包的优点是可以避免全局变量的污染，缺点是闭包会常驻内存，会增大内存使用量，使用不当很容易造成内存泄露。   
(2)、闭包有三个特性：   
a、函数嵌套函数   
b、函数内部可以引用外部的参数和变量   
c、参数和变量不会被垃圾回收机制回收

**13、谈谈你This对象的理解?**

**回答一：**

(1)、js的this指向是不确定的，也就是说是可以动态改变的。call/apply 就是用于改变this指向的函数，这样设计可以让代码更加灵活，复用性更高   
(2)、this 一般情况下，都是指向函数的拥有者。   
(3)、在函数自执行里，this 指向的是 window 对象。   
扩展：关于this，还有一个地方比较让人模糊的是在dom事件里，通常有如下3种情况：   
a、使用标签属性注册事件，此时this指向的是window对象。   
b、对与a，要让this指向input，可以将this作为参数传递。   
c、使用addEventListener等注册事件。此时this也是指向 input。

**回答二：**

(1)、处于全局作用域下的this：

this;/\*window\*/

var a = {name: this}/\*window\*/

var b = [this];/\*window\*/

在全局作用域下，this默认指向window对象。   
(2)、处在函数中的this，又分为以下几种情况：   
a、一般定义的函数，然后一般的执行：

var a = function(){

console.log(this);

}

a();/\*window\*/

this还是默认指向window。   
b、一般定义，用new调用执行：

var a = function(){

console.log(this);

}

new a();/\*新建的空对象\*/

这时候让this指向新建的空对象，我们才可以给空对象初始化自有变量   
c、作为对象属性的函数，调用时：

var a = {

f:function(){

console.log(this)

}

}

a.f();/\*a对象\*/

这时候this指向调用f函数的a对象。   
(3)、通过call()和apply()来改变this的默认引用：

var b = {id: 'b'};

var a = {

f:function(){

console.log(this)

　}

}

a.f.call(b);/\*window\*/

所有函数对象都有的call方法和apply方法，它们的用法大体相似，f.call(b);的意思 是，执行f函数，并将f函数执行期活动对象里的this指向b对象，这样标示符解析时，this就会是b对象了。不过调用函数是要传参的。所以，f.call(b, x, y); f.apply(b, [x, y]);好吧，以上就是用call方法执行f函数，与用apply方法执行f函数时传参方式，它们之间的差异，大家一目了然：apply通过数组的方式传递参数，call通过一个个的形参传递参数。   
(4)、一些函数特殊执行情况this的指向问题：   
a、setTimeout()和setInverval():

var a = function(){

console.log(this);

}

setTimeout(a,0);/\*window\*/

setInterval()类似。   
b、dom模型中触发事件的回调方法执行中活动对象里的this指向该dom对象。

**14、JavaScript对象的几种创建方式?**

**(1) 工厂模式**

function Parent(){

var Child = new Object();

Child.name="欲泪成雪";

Child.age="20";

return Child;

};

var x = Parent();

引用该对象的时候，这里使用的是 var x = Parent()而不是 var x = new Parent();因为后者会可能出现很多问题（前者也成为工厂经典方式,后者称之为混合工厂方式），不推荐使用new的方式使用该对象

**(2)构造函数方式**

function Parent(){

　　this.name="欲泪成雪";

　　this.age="20";

};

var x =new Parent();

**(3) 原型模式**

function Parent(){

};

Parent.prototype.name="欲泪成雪";

Parent.prototype.age="20";

var x =new Parent();

**(4)混合的构造函数，原型方式（推荐）**

function Parent(){

　　this.name="欲泪成雪";

　　this.age=22;

};

Parent.prototype.lev=function(){

　　return this.name;

};

var x =new Parent();

**(5)动态原型方式**

function Parent(){

　　this.name="欲泪成雪";

　　this.age=22;

;

if(typeof Parent.\_lev=="undefined"){

Parent.prototype.lev=function(){

　　return this.name;

}

Parent.\_lev=true;

}

};

var x =new Parent();

**15、get和post的区别,何时使用post**

(1)、get 是从服务器上获取数据，post 是向服务器传送数据。 get 请求返回 request - URI 所指出的任意信息。   
Post 请求用来发送电子邮件、新闻或发送能由交互用户填写的表格。这是唯一需要在请求中发送body的请求。使用Post请求时需要在报文首部 Content - Length 字段中指出body的长度。   
(2)、get 是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址，用户看不到这个过程。   
(3)、对于 get 方式，服务器端用Request.QueryString获取变量的值，对于 post 方式，服务器端用Request.Form获取提交的数据。   
(4)、get 传送的数据量较小，不能大于2KB。post 传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。 用IIS过滤器的只接受get参数，所以一般大型搜索引擎都是用get方式。   
(5)get安全性非常低，post 安全性相对较高。如果这些数据是中文数据而且是   
非敏感数据，那么使用get；如果用户输入的数据不是中文字符而且包含敏感数据，那么还是使用 post 为好。

**16、null和undefined的区别？**

(1)、null是一个表示”无”的对象，转为数值时为0；undefined是一个表示”无”的原始值，转为数值时为NaN。当声明的变量还未被初始化时，变量的默认值为undefined。   
(2)、null用来表示尚未存在的对象，常用来表示函数企图返回一个不存在的对象。   
(3)、undefined表示”缺少值”，就是此处应该有一个值，但是还没有定义。典型用法是：   
a、变量被声明了，但没有赋值时，就等于undefined。   
b、调用函数时，应该提供的参数没有提供，该参数等于undefined。   
c、对象没有赋值的属性，该属性的值为undefined。   
d、函数没有返回值时，默认返回undefined。   
(4)、null表示”没有对象”，即该处不应该有值。典型用法是：   
a、作为函数的参数，表示该函数的参数不是对象。   
b、作为对象原型链的终点。

**17、请写出js内存泄漏的问题?**

**回答一：**

(1)、IE7/8 DOM对象或者ActiveX对象循环引用导致内存泄漏   
a、多个对象循环引用   
b、循环引用自己   
(2)、基础的DOM泄漏   
当原有的DOM被移除时，子结点引用没有被移除则无法回收。   
(3)、timer定时器泄漏   
这个时候你无法回收buggyObject,解决办法，先停止timer然后再回收

**回答二：**

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。   
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。   
闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）也会引发内存泄漏问题。

**18、哪些地方会出现css阻塞，哪些地方会出现js阻塞？**

**js的阻塞特性：**

所有浏览器在下载JS的时候，会阻止一切其他活动，比如其他资源的下载，内容的呈现等等。直到JS下载、解析、执行完毕后才开始继续并行下载其他资源并呈现内容。为了提高用户体验，新一代浏览器都支持并行下载JS，但是JS下载仍然会阻塞其它资源的下载（例如.图片，css文件等）。   
由于浏览器为了防止出现JS修改DOM树，需要重新构建DOM树的情况，所以就会阻塞其他的下载和呈现。   
嵌入JS会阻塞所有内容的呈现，而外部JS只会阻塞其后内容的显示，2种方式都会阻塞其后资源的下载。也就是说外部样式不会阻塞外部脚本的加载，但会阻塞外部脚本的执行。

**CSS怎么会阻塞加载？**

CSS本来是可以并行下载的，在什么情况下会出现阻塞加载了(在测试观察中，IE6下CSS都是阻塞加载）   
当CSS后面跟着嵌入的JS的时候，该CSS就会出现阻塞后面资源下载的情况。而当把嵌入JS放到CSS前面，就不会出现阻塞的情况了。   
根本原因：因为浏览器会维持html中css和js的顺序，样式表必须在嵌入的JS执行前先加载、解析完。而嵌入的JS会阻塞后面的资源加载，所以就会出现上面CSS阻塞下载的情况。

**JS应该放在什么位置？**

(1)、放在底部，虽然放在底部照样会阻塞所有呈现，但不会阻塞资源下载。   
(2)、如果嵌入JS放在head中，请把嵌入JS放在CSS头部。   
(3)、使用defer（只支持IE）   
(4)、不要在嵌入的JS中调用运行时间较长的函数，如果一定要用，可以用setTimeout来调用

**Javascript无阻塞加载具体方式**

将脚本放在底部。\还是放在head中，用以保证在js加载前，能加载出正常显示的页面。\<script>标签放在\前。   
成组脚本：由于每个\<script>标签下载时阻塞页面解析过程，所以限制页面的\<script>总数也可以改善性能。适用于内联脚本和外部脚本。   
非阻塞脚本：等页面完成加载后，再加载js代码。也就是，在window.onload事件发出后开始下载代码。   
（1）defer属性：支持IE4和fierfox3.5更高版本浏览器   
（2）动态脚本元素：文档对象模型（DOM）允许你使用js动态创建HTML的几乎全部文档内容。代码如下：

<script>

var script=document.createElement("script");

script.type="text/javascript";

script.src="file.js";

document.getElementsByTagName("head")[0].appendChild(script);

</script>

此技术的重点在于：无论在何处启动下载，文件额下载和运行都不会阻塞其他页面处理过程。即使在head里（除了用于下载文件的http链接）。

**19、对比Flash与ajax哪个好？**

**Ajax的优势：**

(1)、可搜索性   
普通的文本网页会更有利于SEO。文本内容是搜索引擎容易检索的，而繁琐的swf字节码却是搜索引擎不愿触及的。虽然Google等一些大型的搜索引擎可以检索SWF内部的内容，但是仍然有很多麻烦存在。   
(2)、开放性   
Flash常年以来被Macromedia看的很死。包括Flex、FMS等辅佐技术一直都需要昂贵的安装、维护费用。而JS则没有这样的麻烦。没有人愿意承担法律和版权的风险。   
费用   
Flash开发是很昂贵的，因为FlashIDE等环境都是要收费的．而Ajax则不同．虽然有一些便宜的生成swf的工具，但是他们的工能实在无法满足复杂需求。   
(3)、易用性   
Ajax程序有更好的易用性。由于中间有一层Flashplayer代理层，因此许多辅助功能无法被Flash灵活利用。而且Flash在一些方面有着不好的口碑。比如弹出广告、比如恶意代码。   
（awflasher.com个人认为这八成是乱上xx网站造成的）   
(4)、易于开发   
人们开发复杂的Ajax和Flash应用程序时，都会借助一些高级的开发工具。普遍来说，Ajax的开发包比Flash简便、容易。

**Flash的优势：**

(1)、多媒体处理   
Flash在音频、视频等多媒体领域相比HTML有绝对的优势。现在几乎所有的网站都包含有Flash内容。   
(2)、兼容性   
兼容性好：由于通过了唯一的FlashPlayer“代理”。人们不必像调试JS那样，在不同的浏览器中调试程序。   
(3)、矢量图型   
这是Flash最大的优势，同样处在这一领域的SVG、Canvas element以及Direct完全不能与Flash相比。   
(4)、客户端资源调度   
Flash能够更容易的调用浏览器以外的外部资源。比如摄像头、麦克风等。然而这是普通的HTML无法完成的。但是这也许是一个缺点（为什么呢？）

**Ajax的劣势：**

(1)、它可能破坏浏览器的后退功能   
(2)、使用动态页面更新使得用户难于将某个特定的状态保存到收藏夹中 ，不过这些都有相关方法解决。

**Flash的劣势：**

(1)、二进制格式   
(2)、格式私有   
(3)、flash 文件经常会很大，用户第一次使用的时候需要忍耐较长的等待时间   
(4)/性能问题   
ajax与flash各有利弊，到底哪个好，这取决于你的需求

**20、请你解释一下事件冒泡机制**

a、在一个对象上触发某类事件（比如单击onclick事件），如果此对象定义了此事件的处理程序，那么此事件就会调用这个处理程序，如果没有定义此事件处理程序或者事件返回true，那么这个事件会向这个对象的父级对象传播，从里到外，直至它被处理（父级对象所有同类事件都将被激活），或者它到达了对象层次的最顶层，即document对象（有些浏览器是window）。   
b、冒泡型事件：事件按照从最特定的事件目标到最不特定的事件目标(document对象)的顺序触发   
c、js冒泡机制是指如果某元素定义了事件A，如click事件，如果触发了事件之后，没有阻止冒泡事件，那么事件将向父级元素传播，触发父类的click函数。

//阻止冒泡时间方法，兼容ie(e.cancleBubble)和ff(e.stopProgation)

function stopBubble(e){

var evt = e||window.event;

evt.stopPropagation?evt.stopPropagation():(evt.cancelBubble=true);//阻止冒泡

evt.preventDefault

**21、请你说说split()与join() 函数的区别?**

前者是切割成数组的形式，后者是将数组转换成字符串Join 函数获取一批字符串，然后用分隔符字符串将它们联接起来，从而返回一个字符串。Split 函数获取一个字符串，然后在分隔符处将其断开，从而返回一批字符串。但是，这两个函数之间的主要区别在于 Join 可以使用任何分隔符字符串将多个字符串连接起来，而 Split 只能使用一个字符分隔符将字符串断开。   
简单地说，如果你用split，是把一串字符（根据某个分隔符）分成若干个元素存放在一个数组里。而Join是把数组中的字符串连成一个长串，可以大体上认为是split的逆操作。

**22、说说你对Promise的理解?**

ES6 原生提供了 Promise 对象。   
所谓 Promise，就是一个对象，用来传递异步操作的消息。它代表了某个未来才会知道结果的事件（通常是一个异步操作），并且这个事件提供统一的 API，可供进一步处理。   
Promise 对象有以下两个特点。   
(1)、对象的状态不受外界影响。Promise 对象代表一个异步操作，有三种状态：Pending（进行中）、Resolved（已完成，又称 Fulfilled）和 Rejected（已失败）。只有异步操作的结果，可以决定当前是哪一种状态，任何其他操作都无法改变这个状态。这也是 Promise 这个名字的由来，它的英语意思就是「承诺」，表示其他手段无法改变。   
(2)、一旦状态改变，就不会再变，任何时候都可以得到这个结果。Promise 对象的状态改变，只有两种可能：从 Pending 变为 Resolved 和从 Pending 变为 Rejected。只要这两种情况发生，状态就凝固了，不会再变了，会一直保持这个结果。就算改变已经发生了，你再对 Promise 对象添加回调函数，也会立即得到这个结果。这与事件（Event）完全不同，事件的特点是，如果你错过了它，再去监听，是得不到结果的。

有了 Promise 对象，就可以将异步操作以同步操作的流程表达出来，避免了层层嵌套的回调函数。此外，Promise 对象提供统一的接口，使得控制异步操作更加容易。

Promise 也有一些缺点。首先，无法取消 Promise，一旦新建它就会立即执行，无法中途取消。其次，如果不设置回调函数，Promise 内部抛出的错误，不会反应到外部。第三，当处于 Pending 状态时，无法得知目前进展到哪一个阶段（刚刚开始还是即将完成）。

**23、谈谈你对Javascript垃圾回收机制的理解？**

**(1)、标记清除（mark and sweep）**

这是JavaScript最常见的垃圾回收方式，当变量进入执行环境的时候，比如函数中声明一个变量，垃圾回收器将其标记为“进入环境”，当变量离开环境的时候（函数执行结束）将其标记为“离开环境”。   
垃圾回收器会在运行的时候给存储在内存中的所有变量加上标记，然后去掉环境中的变量以及被环境中变量所引用的变量（闭包），在这些完成之后仍存在标记的就是要删除的变量了

**(2)、引用计数(reference counting)**

在低版本IE中经常会出现内存泄露，很多时候就是因为其采用引用计数方式进行垃圾回收。引用计数的策略是跟踪记录每个值被使用的次数，当声明了一个 变量并将一个引用类型赋值给该变量的时候这个值的引用次数就加1，如果该变量的值变成了另外一个，则这个值得引用次数减1，当这个值的引用次数变为0的时 候，说明没有变量在使用，这个值没法被访问了，因此可以将其占用的空间回收，这样垃圾回收器会在运行的时候清理掉引用次数为0的值占用的空间。   
在IE中虽然JavaScript对象通过标记清除的方式进行垃圾回收，但BOM与DOM对象却是通过引用计数回收垃圾的，也就是说只要涉及BOM及DOM就会出现循环引用问题。

**24、说说你对原型（prototype）理解?**

JavaScript是一种通过原型实现继承的语言与别的高级语言是有区别的，像java，C#是通过类型决定继承关系的，JavaScript是的动态的弱类型语言，总之可以认为JavaScript中所有都是对象，在JavaScript中，原型也是一个对象，通过原型可以实现对象的属性继承，JavaScript的对象中都包含了一个” prototype”内部属性，这个属性所对应的就是该对象的原型。

“prototype”作为对象的内部属性，是不能被直接访问的。所以为了方便查看一个对象的原型，Firefox和Chrome内核的JavaScript引擎中提供了”**proto**“这个非标准的访问器（ECMA新标准中引入了标准对象原型访问器”Object.getPrototype(object)”）。

原型的主要作用就是为了实现继承与扩展对象。

**25、typeof与instanceof的区别是什么？**

**在 JavaScript 中，判断一个变量的类型可以用typeof**

(1)、数字类型， typeof 返回的值是 number。比如说：typeof(1)，返回值是number   
(2)、字符串类型， typeof 返回的值是 string。比如typeof(“123”)返回值是string。   
(3)、布尔类型， typeof 返回的值是 boolean 。比如typeof(true)返回值是boolean。   
(4)、对象、数组、null 返回的值是 object 。比如typeof(window)，typeof(document)，typeof(null)返回的值都是object。   
(5)、函数类型，返回的值是 function。比如：typeof(eval)，typeof(Date)返回的值都是function。   
(6)、不存在的变量、函数或者undefined，将返回undefined。比如：typeof(abc)、typeof(undefined)都返回undefined。

**在 JavaScript 中,instanceof用于判断某个对象是否被另一个函数构造。**

使用 typeof 运算符时采用引用类型存储值会出现一个问题，无论引用的是什么类型的对象，它都返回 “object”。ECMAScript 引入了另一个 Java 运算符 instanceof 来解决这个问题。instanceof 运算符与 typeof 运算符相似，用于识别正在处理的对象的类型。与 typeof 方法不同的是，instanceof 方法要求开发者明确地确认对象为某特定类型。

**26、说说你对node.js的理解**

a、Node.js 是一个基于Google Chrome V8 引擎的 JavaScript 运行环境。Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。Node.js 的包管理器 npm，是全球最大的开源库生态系统。   
b、能方便地搭建响应速度快、易于扩展的网络应用，Node.js 使用事件驱动， 非阻塞I/O 模型而得以轻量和高效，非常适合在分布式设备上运行的数据密集型的实时应用。   
c、简单说Node.js就是运行在服务器端的JavaScript，是现在流行的语言中能同时运行在前端与后台的程序语言

**27、NPM(包管理器)作用是什么?**

NPM是随同NodeJS一起安装的包管理工具，能解决NodeJS代码部署上的很多问题，常见的使用场景有以下几种：   
a、允许用户从NPM服务器下载别人编写的第三方包到本地使用。   
b、允许用户从NPM服务器下载并安装别人编写的命令行程序到本地使用。   
c、允许用户将自己编写的包或命令行程序上传到NPM服务器供别人使用。