1. **javascript的typeof返回哪些数据类型**

答案：string,boolean,number,undefined,function,object

1. **例举3种强制类型转换和2种隐式类型转换?**

答案：强制（parseInt,parseFlo

at,number）；隐式（==  ===）

1. **Split和join的区别**

答案：前者是将字符串切割成数组的形式，后者是将数组转换成字符串

1. **数组方法pop() push() unshift() shift()**

答案：push()尾部添加 pop()尾部删除

unshift()头部添加 shift()头部删除

1. **IE和标准下有哪些兼容性的写法**

答案：

var ev = ev || window.event

document.documentElement.clientWidth || document.body.clientWidth  
Var target = ev.srcElement||ev.target

1. **ajax请求的时候get 和post方式的区别**

答案：

一个在url后面 ，一个放在虚拟载体里面  
get有大小限制(只能提交少量参数)  
安全问题  
应用不同 ，请求数据和提交数据

1. **call和apply的区别**

答案：

Object.call(this,obj1,obj2,obj3)  
Object.apply(this,arguments)

**8. ajax请求时，如何解析json数据**

答案：使用JSON.parse

1. **什么是事件委托**

答案: 利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行！

1. **如何阻止事件冒泡**

答案：ie:阻止冒泡ev.cancelBubble = true;非IE ev.stopPropagation();

1. **如何阻止默认事件**

答案：(1)return false；(2) ev.preventDefault();

1. **添加 删除 替换 插入到某个节点的方法**

答案：

1）创建新节点

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() //添加

removeChild() //移除

replaceChild() //替换

insertBefore() //插入

3）查找

getElementsByTagName() //通过标签名称

getElementsByName() //通过元素的Name属性的值

getElementById() //通过元素Id，唯一性

1. **Javascript的事件流模型都有什么?**

答案：

“事件冒泡”：事件开始由最具体的元素接受，然后逐级向上传播

“事件捕捉”：事件由最不具体的节点先接收，然后逐级向下，一直到最具体的

“DOM事件流”：三个阶段：事件捕捉，目标阶段，事件冒泡

1. **null和undefined的区别？**

答案：

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。 null用来表示尚未存在的对象

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：

（1）变量被声明了，但没有赋值时，就等于undefined。

（2）调用函数时，应该提供的参数没有提供，该参数等于undefined。

（3）对象没有赋值的属性，该属性的值为undefined。

（4）函数没有返回值时，默认返回undefined。

null表示"没有对象"，即该处不应该有值。典型用法是：

（1） 作为函数的参数，表示该函数的参数不是对象。

（2） 作为对象原型链的终点。

1. **new操作符具体干了什么呢?**

答案：

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 。

1. **js延迟加载的方式有哪些？**

答案：

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

1. **如何获取javascript三个数中的最大值和最小值？**

答案：

Math.max(a,b,c);//最大值

Math.min(a,b,c)//最小值

1. **.form中的input可以设置为readonly和disable，请问2者有什么区别？**

答案：  
readonly不可编辑，但可以选择和复制；值可以传递到后台  
disabled不能编辑，不能复制，不能选择；值不可以传递到后台

1. **Ajax原理：**

答案：

(1)创建对象

var xhr = new XMLHttpRequest();

(2)打开请求

xhr.open('GET', 'example.txt', true);

(3)发送请求

xhr.send(); 发送请求到服务器

(4)接收响应

xhr.onreadystatechange =function(){}

(1)当readystate值从一个值变为另一个值时，都会触发readystatechange事件。

(2)当readystate==4时，表示已经接收到全部响应数据。

(3)当status ==200时，表示服务器成功返回页面和数据。

(4)如果(2)和(3)内容同时满足，则可以通过xhr.responseText，获得服务器返回的内容。

1. **解释什么是Json:**

答案：

(1)JSON 是一种轻量级的数据交换格式。

(2)JSON 独立于语言和平台，JSON 解析器和 JSON 库支持许多不同的编程语言。

(3)JSON的语法表示三种类型值，简单值(字符串，数值，布尔值，null),数组，对象

1. **浏览器的滚动距离：**

答案：

可视区域距离页面顶部的距离

scrollTop=document.documentElement.scrollTop||document.body.scrollTop

1. **可视区的大小：**

答案：

(1)innerXXX（不兼容ie）

window.innerHeight 可视区高度，包含滚动条宽度

window.innerWidth  可视区宽度，包含滚动条宽度

(2)document.documentElement.clientXXX(兼容ie)

document.documentElement.clientWidth可视区宽度，不包含滚动条宽度

document.documentElement.clientHeight可视区高度，不包含滚动条宽度

1. **节点的种类有几种，分别是什么？**

答案：

(1)元素节点：nodeType ===1;

(2)文本节点：nodeType ===3;

(3)属性节点：nodeType ===2;

1. **innerHTML和outerHTML的区别**

答案：

innerHTML(元素内包含的内容）

outerHTML(自己以及元素内的内容）

**25. offsetWidth offsetHeight和clientWidth clientHeight的区别**

答案：

(1)offsetWidth （content宽度+padding宽度+border宽度）

(2)offsetHeight（content高度+padding高度+border高度）

(3)clientWidth（content宽度+padding宽度）

(4)clientHeight（content高度+padding高度）

1. **闭包的好处**

答案：

(1)希望一个变量长期驻扎在内存当中(不被垃圾回收机制回收)

(2)避免全局变量的污染

(3)私有成员的存在

(4)安全性提高

1. **dom事件委托有什么原理，有什么优缺点？**

答案：

事件委托原理:事件冒泡机制

优点：

1.可以大量节省内存占用，减少事件注册。比如ul上代理所有li的click事件就很不错。  
2.可以实现当新增子对象时，无需再对其进行事件绑定，对于动态内容部分尤为合适

缺点：

事件代理的常用应用应该仅限于上述需求，如果把所有事件都用事件代理，可能会出现事件误判。即本不该被触发的事件被绑定上了事件。

1. **dom选择器优先级是什么，以及权重值计算**

答案：

1.行内样式 1000  
2.id 0100  
3.类选择器、伪类选择器、属性选择器[type="text"] 0010  
4.标签选择器、伪元素选择器(::first-line) 0001  
5.通配符\*、子选择器、相邻选择器 0000

1. **重排和重绘**

答案：

部分渲染树（或者整个渲染树）需要重新分析并且节点尺寸需要重新计算。这被称为重排。注意这里至少会有一次重排-初始化页面布局。  
由于节点的几何属性发生改变或者由于样式发生改变，例如改变元素背景色时，屏幕上的部分内容需要更新。这样的更新被称为重绘。

1. **什么情况会触发重排和重绘**

答案：

添加、删除、更新 DOM 节点  
通过 display: none 隐藏一个 DOM 节点-触发重排和重绘  
通过 visibility: hidden 隐藏一个 DOM 节点-只触发重绘，因为没有几何变化  
移动或者给页面中的 DOM 节点添加动画  
添加一个样式表，调整样式属性  
用户行为，例如调整窗口大小，改变字号，或者滚动

1. **什么是JavaScript？**

答案：

我们可以从几个方面去说JavaScript是什么：

<1基于对象

javaScript中内置了许多对象供我们使用【String、Date、Array】等等

javaScript也允许我们自己自定义对象

<2事件驱动

当用户触发执行某些动作的时候【鼠标单机、鼠标移动】，javaScript提供了监听这些事件的机制。当用户触发的时候，就执行我们自己写的代码。

<3解释性语言

[x] javaScript代码是由浏览器解析的，并不需要编译。

<4基于浏览器的动态交互技术

既然javaScript是由浏览器解析的，那么它肯定要基于浏览器。 javaScript让网页变得更加“灵活”“

<5弱类型

[x] 像java、c++等编译型语言，要先定义变量，后使用。javaScript能够直接使用，不需要先定义

1. **es6中的箭头函数和普通函数有什么区别？**

答案：

<1普通函数中的 this总是指向调用它的那个对象，

<2箭头函数没有自己的this,他的this永远指向其定义环境，任何方法都改变不了其指向，如call()、bind()、apply()。（正是因为它没有this，所以也就不能用作构造函数，也没有原型对象）

箭头函数不能当作构造函数，也就是说，不能使用new命令，否则会报错。

箭头函数没有原型属性。

箭头函数不可以使用yield命令，因此箭头函数不能用作Generator函数。

箭头函数不能使用arguments对象，该对象在函数体内不存在。如果要用，可以用rest参数代替。

变量提升：由于js的内存机制，function的级别最高，而用箭头函数定义函数的时候，需要var（let、const）关键字，而var所定义的变量不能得到变量提升。故箭头函数一定要定义于调用之前。

拓展：this的指向问题？

 1、普通函数中，this指向其函数的直接调用者；

2、箭头函数中，this指向其定义环境，任何方法都改变不了其指向，如call( ）、bind（）等；

3、构造函数中，如果不使用new，则this指向window，

     如果使用new创建了一个实例，则this指向该实例。

4、window内置函数中，如setInterval，setTimeout等，其内部的this指向Window。

5、匿名函数的this指向Window。

6、apply（）、call（）、bind（）可以改变this的指向

1. **请指出 JavaScript 宿主对象和原生对象的区别？**

答案：

宿主对象是指DOM和BOM。  
原生对象是Object、Function、Array、String、Boolean、Number、Date、RegExp、Error、Math等对象

1. **请尽可能详尽的解释 Ajax 的工作原理。以及使用 Ajax 都有哪些优劣？**

答案：

Ajax是无需刷新页面就能从服务器取得数据的一种方法。

Ajax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更新页面。

过程：

<1创建XMLHttpRequest对象。

<2设置响应HTTP请求的回调函数。

<3创建一个HTTP请求，指定相应的请求方法、url等。

<4发送HTTP请求。

<5获取服务器端返回的数据。

<6使用JavaScript操作DOM更新页面。

缺点:

<1对搜索引擎不友好

<2要实现Ajax下的前后退功能成本较大

<3跨域问题限制

1. **请解释变量声明提升。**

答案：

变量的声明前置就是把变量的声明提升到当前作用域的最前面。  
函数的声明前置就是把整个函数提升到当前作用域的最前面(位于前置的变量声明后面)。

案例：

//变量的声明前置：

console.log(num); // undefined

var num=1;

等价于：

//变量的声明前置

var num;

console.log(num); //undefined

num=1;

1. **请描述事件冒泡机制。**

答案：

事件冒泡,事件最开始时由触发的那个元素身上发生，然后沿着DOM树向上传播，直到document对象。如果想阻止事件起泡，可以使用e.stopPropagation()。

1. **请解释 JSONP 的工作原理，以及它为什么不是真正的 Ajax。**

答案：

JSONP（JSON with Padding）是一种非官方跨域数据交互协议，它允许在服务器端集成< script >标签返回至客户端，通过javascript回调的形式实现跨域访问。

因为同源策略的原因，我们不能使用XMLHttpRequest与外部服务器进行通信，但是< script >可以访问外部资源，所以通过JSON与< script >相结合的办法，可以绕过同源策略从外部服务器直接取得可执行的JavaScript函数。

原理：

客户端定义一个函数，比如jsonpCallback，然后创建< script >，src为url + ?jsonp=jsonpCallback这样的形式，之后服务器会生成一个和传递过来jsonpCallback一样名字的参数，并把需要传递的数据当做参数传入，比如jsonpCallback(json)，然后返回给客户端，此时客户端就执行了这个服务器端返回的jsonpCallback(json)回调。

通俗的说，就是客户端定义一个函数然后请求，服务器端返回的javascript内容就是调用这个函数，需要的数据都当做参数传入这个函数了。

优点：兼容性好，简单易用，支持浏览器与服务器双向通信  
缺点：只支持GET请求；存在脚本注入以及跨站请求伪造等安全问题

补充一点，JSONP不使用XMLHttpRequest对象加载资源，不属于真正意义上的AJAX。

1. **请举出一个匿名函数的典型用例？**

答案：

定义回调函数，立即执行函数，作为返回值的函数，使用方法var foo = function() {}定义的函数。

1. **描述以下变量的区别：null，undefined 或 undeclared？该如何检测它们？**

答案：

未定义的属性、定义未赋值的为undefined，JavaScript访问不会报错；null是一种特殊的object；NaN是一种特殊的number；undeclared 是未声明也未赋值的变量，JavaScript访问会报错

1. **请解释同步和异步函数的区别。**

答案：

同步调用，在发起一个函数或方法调用时，没有得到结果之前，该调用就不返回，直到返回结果；

异步调用的概念和同步相对，在一个异步调用发起后，被调用者立即返回给调用者，但调用者不能立刻得到结果，被调用者在实际处理这个调用的请求完成后，通过状态、通知或回调等方式来通知调用者请求处理的结果。

简单地说，同步就是发出一个请求后什么事都不做，一直等待请求返回后才会继续做事；异步就是发出请求后继续去做其他事，这个请求处理完成后会通知你，这时候就可以处理这个回应了

1. **你使用哪些工具和技术来调试 JavaScript 代码？**

答案：

1. javascript的debugger语句  
   需要调试js的时候，我们可以给需要调试的地方通过debugger打断点，代码执行到断点就会暂定，这时候通过单步调试等方式就可以调试js代码

if(waldo){

debugger;

}

这时候打开console面板，就可以调试了

2.DOM断点  
DOM断点是一个Firebug和chrome DevTools提供的功能，当js需要操作打了断点的DOM时，会自动暂停，类似debugger调试。  
使用DOM断点步骤：  
<1选择你要打断点的DOM节点  
<2右键选择Break on..  
<3选择断点类型

另外的调试方法例如alert, console.log，查看元素等

1. **使用 Promises 而非回调 (callbacks) 优缺点是什么？**

答案：

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。它由社区最早提出和实现，ES6将其写进了语言标准，统一了用法，原生提供了Promise对象。

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。

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）。

原生重点：

字符串的方法

数组的方法

面向对象

**闭包**

**作用域**

**作用域链 需要重点理解概念性的问题**

**原型链**

**原型**

**继承**

This指向的问题

24道基础算法题

高级算法（比如斐波那契数列，二叉树等等）