1、什么是ajax？

异步的javascript and  xml

使用ajax能够实现页面无刷新（局部刷新）

用户体验好

响应数据的速度快

2、ajax如何实现客户端和服务器端数据交换

3、异步 和  同步

同步：顺序执行 （代码按照从上到下的顺序执行）   打电话

异步：同时执行  （生活中的同步）    发短信

4、计算机中    进程

程序从开始到结束这样一个过程 称为一个进程

进程 分为  单线程 和  多线程

多线程 异步 执行

5、ajax请求方式

get  （常用的方式）：通过url传值  ，传送的数据量较少  ，安全性较低

post  ：   传送的数据量较大  ， 安全性较高

6、ajax如何请求数据步骤

打电话流程：

1、需要一个电话

2、拨号

3、建立连接

4、说话

ajax请求数据步骤

1、创建ajax对象

2、通过  open方法 和服务器建立连接

    第一个参数： 请求方法

    第二个参数： 请求路径url

    第三个参数： true  表示异步   可以省略

3、通过 send 方法向服务器发送数据

4、 服务器响应客户端数据的请求  并将结果传递给客户端

     通过 onreadystatechange 事件  响应数据的请求

     判断状态值 4 和状态码 200    来 确定 是否请求数据成功  如果请求成功，才能接受服务器返回的结果

ajax状态值说明：

0 ：请求未初始化（还没有调用open）

     1 ：请求已经建立，但是还没有发送（还没有调用send()）

     2 : 请求已发送，正在处理中

     3 : 请求在处理中，通常响应中已有部分数据可以用了，但是服务器还没有完成响应的生成

     4 ：响应已完成 您可以获取并使用服务器的响应了

ajax状态码：

     200 请求数据成功

     404 没有找到文件

     500 服务器错误

     400 服务器语法错误

7、缓存问题

缓存：  cpu和内存之间的一块存储区   容量小  传送数据速度快

IE浏览器中 在运行ajax时 容易出现缓存问题，  不能及时刷新请求的数据

解决 ：　改变请求的url    在路径上 加一个随机数

         data.txt?id = new Date().getTime()

         data.txt?id = Math.random()

思考

data.txt ---  [23,1,5,6,7,8]     通过ajax方式请求这个文件  操作数组   找到最小值

8、JSON.parse 和 eval 区别

JSON.parse()   使用时 安全较高

eval()       安全性 低          eval(  "(" + str + ")"  );

1、同源策略（同一个来源）

所谓的同源 就是指 ajax在请求数据时， 保证  协议、域名、端口号 完全一致，才能够通过请求

[http://www.baidu.com:80](http://www.baidu.com/)

同源策略是浏览器的一个行业标准，有了同源策略保证用户数据的安全性。

所有的javascript都要遵循这个同源策略的

ajax不能实现跨域请求数据（受同源策略的影响不能实现跨域）

2、jsonp的跨域访问数据

什么是jsonp：

json  with padding，  是json的一种数据使用模式，可以通过jsonp实现数据的跨域访问。

jsonp跨域的原理： 通过动态创建script标签，然后设置src的路径为服务器提供的接口路径，通过一个回调函数callback将服务器信息返回。

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

src="http://php1711/xxx.php?**data=callback**";

document.body.appendChild(script);

通过jsonp方式得到的服务器数据 都是object类型

jsonp接口：

 https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?wd="+txt+"&cb=fn

 wd 参数值为用户搜索的数据值

 cb 为callback回调函数

3、cors的跨域访问

header("Access-Control-Allow-Origin:\*");--- 表示任何域名下都可以访问   该代码在服务器上设置

4、promise对象

为了  解除嵌套，让程序更清晰   为了解决javascript的异步，   我们使用 promise对象

startMove(obj,{},function(){  ---异步

})

ajax(url,callback,data)   --- 异步

Promise对象 ----  异步编程

promise 三种状态： pending --- 进行中     resolved --- 已完成      rejected  --- 失败

如果promise对象中执行了success()   ,表示成功，可以执行then方法了

then()方法 ，参数可以有两个匿名函数，第一个匿名函数会执行成功，第二个匿名函数会执行失败

如果promise对象中执行了failed()   ,表示失败，可以执行catch方法了

5、xss脚本攻击

xss ：Cross Site  Scripting  跨站脚本攻击

为了不和层叠样式表(Cascading Style Sheet)的缩写混淆，故将跨站脚本攻击缩写为xss

6、什么是脚本攻击

思考一个问题： 网站注册功能

脚本攻击：在表单中输入一些具有特殊含义代码的数据  比如：  <a href="#">xxx</a>

解决办法1：在表单做js验证

问题来了：

     网站使用可以通过右键单击浏览器查看源代码的方式，获取到网站的表单数据（包括后台路径）

     这时 网站的使用者会将表单数据粘贴到本地的另一个页面，运行这个页面，在表单中输入一些特殊含义的代码，也能够实现脚本攻击。

     所以前端的 js验证 并没有从根本上解决脚本攻击。

解决办法2：　在服务器上做验证

服务器如何验证-核心思想：去掉表单中输入的标签，保留标签的内容

什么是对象？

对象是一个整体，对外提供一些功能

一切具有本质特征和行为的物质

一切具有属性和方法的事物

汽车 ：

     本质特征：  品牌    颜色     价格     厂家

     行为 ：     行驶    启动

学生 ：

      本质特征 ：  性别  年龄  姓名     身高

      行为   ：  学习      吃饭     瞌睡

对象有两个特性：

           属性  和   方法

array  ：  属性---length    方法：   pop   push ....

所有的对象都具有特定的属性和方法。  使用时：   对象.属性    /   对象.方法

所有的基本数据类型都没有属性和方法

所有的对象数据类型都有属性和方法

方法和函数区别：  方法属于函数    函数包含方法

面向对象好处：   不会产生变量的污染

                         将每一个功能都独立出来，便于后期的管理维护

2、什么是类？（js中没有类的概念）

类是具有相同本质特征的一类事物的总称

类和对象的关系：

类是对象的抽象，对象是类的实例（类是对象的抽象化，对象是类的具象化）

Array  ---  构造函数 （类）

var arr = new Array();

var brr = new Array();

2、什么是面向对象（oop）  object   oriented   programing

面向对象编程本身是一种 编程思想

面向过程（C语言）：                面向过程体现的是数学逻辑

面向对象（c++、java、php、c#）：   面向对象体现的是生活逻辑

路：  属性--长度   500

人：  行走速度 ： 20/minute

3、面向对象的特性

封装：屏蔽内部细节，只给提供对外实现的一个接口

继承：子类继承父类中特有的属性和方法

多态（js中没有多态的概念）

4、对象创建

1、var obj = new Object();

2、var obj = {};

定义一个对象car    属性包含：名称 ： 福特汽车   颜色：白色     价格：15W

        方法   汽车描述方法----输出   这是一辆..颜色 价格为 ..  的   福特汽车

                 汽车行走方法--- 输出   汽车缓慢行驶在人行道上

5、工厂模式（设计模式）

优点：  工厂模式可以解决 同类对象创建时  重复代码  写多次的问题

缺点：  不能确定某个对象属于哪一个类

       一般一个对象是同new关键字构造出来的，而工厂模式只是一个普通函数的调用，不符合对象的创建规范

6、构造函数(类)

构造函数中的属性 叫作  实例属性

构造函数中的方法 叫作  实例方法

一般所有的对象都是由构造函数创建出来的

为了和普通函数进行区分，一般构造函数命名规范是 大驼峰  ---  ZhongGuo

构造函数中的this 指向的是 构造函数执行时创建出来的那个对象

优点：可以通过构造函数的方式 确定 某个对象属于 哪个类

缺点：同类对象创建时，相同方法会重复创建，空间不共享

7、原型

写在构造函数外面的属性叫 原型属性

写在构造函数外面的方法叫 原型方法

原型优点：

     原型方法能够解决相同类对象创建时，方法重建问题，达到空间共享

原型缺点：

     相同对象的属性值也相同

8、混合

把属性写成实例属性

把方法写成原型方法

原型

所有的构造函数都会有一个原型属性prototype

所有的构造函数创建的对象都会有一个原型属性   实现：

对象.\_\_proto\_\_   指向的是 prototype

原型链 ：　通过\_\_proto\_\_ 属性  串联起来的一个串，直到结果 null ，这个过程称为一个原型链

          p.\_\_proto\_\_.\_\_proto\_\_.\_\_proto\_\_ （null）    p是new出来的一个对象

原型练习：

在Array的原型上实现数组去重

在String的原型上实现字符串去空白

2、原型模式执行流程

1、首先查找构造函数中的实例属性或实例方法，如果有就返回，也就是说不在取查找原型属性了。

2、如果构造函数中没有实例属性和实例方法，就取原型中查找，如果有就返回，如果没有  结果为 undefined

3、原型中的几个关键字（扩展）

测试某个对象是否属于某个类的方法 isPrototypeOf()

语法： 构造函数.prototype.isPrototypeOf(对象):判断一个对象是否指向了该构造函数的原型对象，可以使用isPrototypeOf()方法来测试     结果为boolean类型  同  instanceof

delete ：删除实例属性

语法 ：  delete  对象.实例属性

hasOwnProperty() 是否包含该实例属性,包含返回true，否则返回false

语法： 实例对象.hasOwnProperty("实例属性")

in 是否存在该属性（原型或实例中）

语法： "属性" in  对象实例

4、继承

什么是继承：

     子类继承父类中所有的属性和方法。   这些属性和方法在子类中是不存在的，都是通过父类继承过来的。

继承种类：

     单继承： 一个子类只拥有一个父类

     多继承： 一个子类可以拥有多个父类

5、继承方式1  -   通过改变构造函数(父类)的执行环境

this.parent = Father

6、继承方式2  ---  通过  call方法

 Father.call(this,firstname);

7、继承方式3  --  通过  apply 方法

call 方法和apply方法继承区别：

call方法继承时 第二个参数个数不固定

apply方法继承时  第二个参数是一个数组

8、继承方式4  ---  原型继承

Son.prototype = new Father();

9、继承方式5 --- 混合继承

通过apply 继承实例属性

通过原型 方式 继承方法

1、什么是设计模式？

设计模式（Design pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。

2、设计模式之单例模式

单例：对象只能实例化一次就叫单例

单例模式：字面量的对象声明，其实在设计模式中可以看作是一种单例模式，所谓单例模式，就是永远保持对象的一个实例。

3、设计模式之代理模式

代理模式: 为其他对象提供一种代理，并以控制对这个对象的访问。

4、设计模式之适配器模式

适配器模式（Adapter Pattern）是作为两个不兼容的接口之间的桥梁。

将一个类的接口转换成客户希望的另外一个接口。适配器模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

5、设计模式之工厂模式

让对象的调用者和对象创建过程分离，当对象调用者需要对象时，直接向工厂请求即可。从而避免了对象的调用者与对象的实现类似编码方式耦合，以提高系统的可维护性、可扩展性。

高内聚：   函数参数之间的一个紧密联系性      内聚性越高     参数的联系性越好

低耦合：   模块之间的独立性     耦合性越低  独立性越好

6、闭包

一个函数中返回一个匿名函数，这个函数就成为闭包

闭包就是能够读取其它函数内部变量的函数。闭包的存在延长了局部变量的生命周期。闭包的形式多变，但本质都是一样的，即局部函数全局执行。

闭包可以将一个局部变量长期驻留在内存中。

闭包中的this指向的是window对象