## Js 继承实现方式

Prototype实现继承

构造函数实现继承

Call和Apply实现继承

//prototype 实现继承

function myConstructor(name, age) {

this.name = name;

this.age = age;

}

myConstructor.prototype.say = function() {

alert(this.name + this.age);

}

//构造函数 实现继承

function myConstructor(name) {

this.name = name;

this.sayHello = function() {

}

}

//call 和 apply实现继承

function you() {

myConstructor.call(this, "name", "age");

myConstructor.apply(this, ["name", "age"]);

}

## 多人js开发重名问题

将函数、变量封装到json对象中

闭包

Seajs等模块化工具

## new

创建了一块内存区域

将构造函数的属性和值注入到这块区域

添加\_\_proto\_\_属性，构造函数的prototype == 实例对象的\_\_proto\_\_

## 网站登录都需要验证码，网站安全的关系

防止恶意的注册、暴力的破解、频繁的进行服务端的请求

## XMLHttpRequest

var xmlhttp;

function loadXMLDoc(url) {

xmlhttp = null;

if (window.XMLHttpRequest) { //code for all new browsers

xmlhttp = new XMLHttpRequest();

} else if (window.ActiveXObject) { //code for IE5 and IE6

xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

}

if (xmlhttp != null) {

xmlhttp.onreadystatechange = state\_Change;

xmlhttp.open("GET", url, true);

xmlhttp.send(null);

} else {

alert("Your browser does not support XMLHTTP.");

}

}

function state\_Change() {

if (xmlhttp.readyState == 4) { //4 = "loaded"

if (xmlhttp.status == 200) { //200 = OK

//...our code here...

} else {

alert("Problem retrieving XML data");

}

}

}

## Ajax工作原理

Ajax XMLHttpRequest对象向服务端发起异步请求，获取数据后，更新Dom。

onReadyStateChange 每次状态改变触发响应事件

status 服务器返回的数字代码（404、200、500）

readyState 对象状态值

0(未初始化)，对象已建立，但是尚未初始化(尚未调用open)

1(初始化)，对象已建立，尚未调用Send方法

2(发送数据) send方法已调用，但是当前的状态及http头未知

3(数据传送中)已接收部分数据

4(完成)，数据接收完毕，可通过responseXml和responseText获取完整回应数据

## @import 和 link区别

## Cookies,sessionStorage,localStorage区别

Cookie在http请求中会被携带，即使不需要

两个storage不会自动把数据发给服务器，仅在本地保存

大小:

Cookie大小不能超过4k

Storage大小很大，可以达到5M甚至更多

有效时间:

Cookie可以设置过期时间

Localstorage不会主动删除

Sessionstorage在当前浏览器窗口关闭后自动删除

## Iframe缺点

Iframe会阻塞主页面的Onload事件

搜索引擎的检索程序无法解读这种页面，不利于seo

Iframe和主页面共享程序池,而浏览器对相同域的连接有限制，所以会影响页面的并行加载

使用iframe最好javascript动态给iframe添加src属性值

## 如何实现浏览器多个标签页之间的通信

Localstorage、cookie等本地存储方式

## Js基本数据类型和内置对象

基本数据类型: undefined、null、Boolean、Number、String

内置对象:

数据封装类对象: Object、Array、Boolean、Number和String

其他对象: Function、Arguments、Math、Date、RegExp、Error

## Js按值传递和按引用传递

按值传递: 函数的形参是实参的副本，修改形参并不会影响实参

按引用传递: 函数的形参是实参的引用，两者指向同一块区域，同步做更改

值的传递是拷贝，对象的传递是引用的传递

基本类型是不可变的，对象是可变的

## Js有几种类型的值

堆:原始数据类型(undefined,null,Boolean,Number,String)

栈:引用数据类型(对象、数组和函数)

两种类型区别:存储位置不同.

原始数据类型占据空间小、大小固定、频繁被使用，所以放入栈中存储

引用数据存储在堆中，占空间大、大小不固定，如果存栈会影响性能，所以只是将指针存储在栈中，指针指向堆中的起始地址。

## Js创建对象的方式

1. 字面量创建
2. 无参function实例化
3. 有参函数实例化
4. 工厂方式创建
5. 原型方式

## Js null和undefined对象

首先用==是判断不出null和undefined

Null，定义了对象，但是对象为空

Undefined，是根本就没定义对象