## 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

## Js 实现一个事件监听函数

//stopPropagation - calcelBubble = true

//preventDefault - returnValue = false

var Event = {

addEvent: function(element, type, handler) {

if (element.addEventListener) {

element.addEventListener(type, handler, false);

} else if (element.attachEvent) {

element.attachEvent('on' + type, function() {

handler.call(element);

});

} else {

element['on' + type] = handler;

}

},

removeEvent: function(element, type, handler) {

if (element.removeEventListener) {

element.removeElementListener(type, handler, false);

} else if (element.detachEvent) {

element.detachEvent('on' + type, function() {

handler.call(element);

})

} else {

element['on' + type] = null;

}

}

}

## [“1”,”2”,”3”].map(parseInt) 答案是多少

答案为[1,NaN,NaN]

Map是传递了3个参数，string-index-array

ParseInt只接受2个参数 string-radix

parseInt(“1”,”0”,[“1”,”2”,”3”])

parseInt(“2”,”1”,arr) => NaN

## js 中use strict

js的严格模式

未经定义的变量不能直接使用/禁止删除变量(delete)/

## 常见的Dom操作

创建操作: createDocumentFragment()/createElement()/createTextNode()

添加、移除、替换、插入: appendChild()/removeChild()/replaceChild()/insertBefore()

查找: getElementsByTagName()/getElementsByName()/getElementById()

## Script标签放置在body之前还是之后

H5之后，放在Body之后认为是不合法，但会自动修复解析到body之前。所以script标签放在Body之前比较合适

## jQuery怎么将数组转化为json字符串，在转化回来

/+++++++++++++++++++++++++++++++++++

jquery 中如何将数组转化为json字符串，然后再转化回来

把 Script 标签 放在页面的最底部的body封闭之前 和封闭之后有什么区别？浏览器会如何解析它们？

知道各种JS框架(Angular, Backbone, Ember, React, Meteor, Knockout...)么? 能讲出他们各自的优点和缺点么?

——————————————————————

页面重构怎么操作？

网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。

也就是说是在不改变UI的情况下，对网站进行优化，在扩展的同时保持一致的UI。

对于传统的网站来说重构通常是：

表格(table)布局改为DIV+CSS

使网站前端兼容于现代浏览器(针对于不合规范的CSS、如对IE6有效的)

对于移动平台的优化

针对于SEO进行优化

深层次的网站重构应该考虑的方面

减少代码间的耦合

让代码保持弹性

严格按规范编写代码

设计可扩展的API

代替旧有的框架、语言(如VB)

增强用户体验

通常来说对于速度的优化也包含在重构中

压缩JS、CSS、image等前端资源(通常是由服务器来解决)

程序的性能优化(如数据读写)

采用CDN来加速资源加载

对于JS DOM的优化

HTTP服务器的文件缓存

————————————————

js优化

（1） 减少http请求次数：CSS Sprites, JS、CSS源码压缩、图片大小控制合适；网页Gzip，CDN托管，data缓存 ，图片服务器。

（2） 前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数

（3） 用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。

（4） 当需要设置的样式很多时设置className而不是直接操作style。

（5） 少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。

（6） 避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)。

（7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

（8） 避免在页面的主体布局中使用table，table要等其中的内容完全下载之后才会显示出来，显示比div+css布局慢。

对普通的网站有一个统一的思路，就是尽量向前端优化、减少数据库操作、减少磁盘IO。向前端优化指的是，在不影响功能和体验的情况下，能在浏览器执行的不要在服务端执行，能在缓存服务器上直接返回的不要到应用服务器，程序能直接取得的结果不要到外部取得，本机内能取得的数据不要到远程取，内存能取到的不要到磁盘取，缓存中有的不要去数据库查询。减少数据库操作指减少更新次数、缓存结果减少查询次数、将数据库执行的操作尽可能的让你的程序完成（例如join查询），减少磁盘IO指尽量不使用文件系统作为缓存、减少读写文件次数等。程序优化永远要优化慢的部分，换语言是无法“优化”的。