Javascript常见面试题

作者:Alley

状态:连载中

博客园

微信公众号

001、浅谈堆和栈的理解?

> js变量存储有栈存储和堆存储，基本数据类型的变量存储在栈中，引用数据类型的变量存储在堆中引用类型数据的地址也存在栈中
>
> 当访问基础类型变量时，直接从栈中取值。当访问引用类型变量时，先从栈中读取地址，在根据地址到堆中取出数据

002、js中的数据类型分为哪几类

> 基本数据类型：number string undefined null boolean symbol
>
> 引用数据类型：Object

003、js中的强制类型转换与隐式类型转换

> 隐式类型转换：== != - * / % *= /= -= %=
>
> 强制类型转换：Number parseInt():取整转换 parseFloat()

004、undefined 和null的区别

> ull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。
>
> undefined：
>
> （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：
>
> （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。

005、谈谈你对NaN的理解

> - Not a Number 不是一个数字
> - NaN仅仅代表不是一个数字，自身和自身都不相等（NaN != NaN）
> - 如何判断NaN isNaN===>如果是NaN,返回true,否则返回false Number.isNaN( )
> - NaN的数据类型是number

006、javascript转换成false的值有哪些?

```
0  ''  null  undefined  false
```

007、null 、 true false转换成数字的值都是多少

> - false如果转换成一个数字的话是0
> - true如果转换成一个数字的话就是1
> - null如果转换成一个数字的话就是0

008、do-while循环的使用及while的区别?

> do-while() 无论条件是否成立至少执行一次，和while规则一样，唯一不同的是do{}while会先执行一次(先执行后判断

009、break和continue return的区别

> - continue:continue只是中止本次循环，接着开始下一次循环 ,只能出现在循环中
> - break:break用于完全结束一个循环，跳出循环体 不在执行break下面的代码,只能出现在选择或者循环中

> - return:返回函数的值，不在执行return下面的代码，只能出现函数中

010、什么是作用域链?

> 简单说就是作用域集合 当前作用域 -> 父级作用域 -> ... -> 全局作用域 形成的作用域链条
>
> - 全局作用域的变量和方法都可以进行调用
> - 局部的变量和方法只能局部进行调用(除闭包外)
> - 局部可以访问全局的变量和方法

011、请说一下js的预编译

> 预编译:
>
> - 把var 和 function 定义的变量提升到script的最上方
> - 赋值语句不会被提升，哪怕等号后面是一个function

012、简单的阐述一下js的变量声明提升

> 变量声明和函数声明从他们代码中出现的位置被移动到执行环境的顶部，这个过程就叫做提升 只有声明操作会被提升，赋值和逻辑操作会被留在原地等待执行
>
> Js编译器会把变量声明看成两个部分分别是声明操作(var a)和赋值操作(a=2)

013、如果判断一个对象是不是另一个对象创建出来的

> [].instanceof Array

014、如何将数组转换为字符串？如何将字符串转换为数组

> var str = 数组.join('')
>
> var arr = 字符串.split('')

015、简单谈一谈关于值传递与引用传递

> 值传递：传递的是基本数据类型的数据 （数据不会发生改变）
>
> 引用传递：传递的是引用数据类型的地址 (数据会发生改变)

016、数组排序的方法至少3种

017、什么是ES5?js中的'use strict'是什么？目的是什么?

> ECMA Script5：ECMA的第五次改版 时间：2009年
>
> 'use strict':严格模式
>
> 目的：
>
> 增加更多报错的场合，消除代码运行的一些不安全之处，保证代码 运行的安全。
>
> 提高编译器效率，增加运行速度。
>
> 为未来新版本的JavaScript做好铺垫

018、ES5中新增的数组的方法有哪些?

> ndexOf()：返回第一次出现的下标
>
> lastIndexOf()：返回最后一次出现的下标
>
> forEach()：循环
>
> map():映射
>
> filter():过滤
>
> reduce() 累加器

019、如何将一个字符转换成ascii码？如何将一个数字转换成对应的字符

> charCodeAt()：将字符转换成ascii码
>
> String.fromCharCode()：将数字转换成对应的字符

020、url的组成部分有哪些?

022、什么是浏览器缓存?

> 浏览器缓存（Browser Caching）是为了加速浏览，浏览器在用户磁盘上对请求过的文档进行存储，当访问者再次请求这个页面时，浏览器就可以从本地磁盘显示文档，这样就可以加速页面的阅览

023、如何打印当前浏览器的版本等信息

navigator.userAgent

返回包含浏览器版本等信息的字符串，常用于判断浏览器版本及使用设备（PC或者移动端)

024、在浏览器地址栏输入www.baidu.com后按下回车键会发生什么

1.域名解析

把域名解析ip地址 DNS域名解析系统

2.把ip发送到网络供应商 去找那个对应的主机服务器

3.TCP的三次握手 建立连接

4.开始发送请求 取回入口文件index.html

5.开始解析入口文件，并且取回需要的资源sources

6.进行

025、浅谈关于文档碎片的理解

1、js操作dom时发生了什么?

每次对dom的操作都会触发"重排"，这严重影响到能耗，一般通常采取的做法是尽可能的减少dom操作来减少"重排"

2、什么是文档碎片?

document.createDocumentFragment()一个容器，用于暂时存放创建的dom元素

3、文档碎片有什么用?

```
将需要添加的大量元素　先添加到文档碎片　中，再将文档碎片添加到需要插入的位置，大大减少dom操作，提高性能
```

026、什么是回流和重绘

当渲染树中的一部分或者全部因为元素的尺寸、布局、隐藏等改变而需要重新构建的时候，这时候就会发生回流。

每个页面都至少发生一次回流，也就是页面第一次加载的时候。

在回流的时候，浏览器会使渲染树中受到影响的元素部分失效，并重新绘制这个部分的渲染树，完成回流以后，浏览器会重新绘制受到影响的部分元素到屏幕中，这个过程就是重绘

什么时候会发生回流?

1、添加或者删除可见的DOM元素的时候

2、元素的位置发生改变

3、元素尺寸改变

4、内容改变

5、页面第一次渲染的时候

027、关于onkeydown和onkeypress的区别以及如何获取按下键盘的键盘码

onkeydown:

1、所有的英文字符都是大写

2、功能键也会被识别

onkeypress

1、所有英文字符大小写都支持

2、所有功能键都不会被识别

3、组合键ctrl+回车的code值是10

e.keyCode || e.which

028、什么是事件流？ 什么是事件冒泡？ 什么是事件捕获？

什么是事件流?

当一个HTML元素产生一个事件时,该事件会在元素节点与根节点之间的路径传播，路径所经过的节点都会收到该事件，这个传播的过程叫做DOM事件流

元素触发事件时，事件的传播过程称为事件流，过程分为捕获和冒泡两种

冒泡事件：微软提出的 事件由子元素传递到父元素的过程，叫做冒泡

捕获事件：网景提出的 事件由父元素到子元素传递的过程，叫做事件捕获

029、谈谈你对事件监听的理解

1、事件分为DOM 0级事件和Dom 2级事件，DOM2级事件也叫做事件监听。DOM 0级事件的缺点是如果事件相同 后者的事件会覆盖前者的事件，DOM2级事件可以解决这个问题

2、DOM2级事件的方法是

addEventListener()

参数1：事件类型 不需要加on

参数2：回调函数

参数3：布尔值 true代表捕获 false代表冒泡

解绑事件方法：removeEventListener()

IE浏览器下用：attachEvent()

参数1：事件类型 需要加on

参数2：回调函数

解绑事件方法：detachEvent()

3、事件流、事件冒泡、事件捕获

当一个HTML元素产生一个事件时,该事件会在元素节点与根节点之间的路径传播，路径所经过的节点都会收到该事件，这个传播的过程叫做DOM事件流

元素触发事件时，事件的传播过程称为事件流，过程分为捕获和冒泡两种

冒泡事件：微软提出的 事件由子元素传递到父元素的过程，叫做冒泡

捕获事件：网景提出的 事件由父元素到子元素传递的过程，叫做事件捕获

4、IE与火狐的事件机制有什么区别?

事件处理机制：IE是事件冒泡、火狐是 事件捕获;

5、事件代理/事件委托

利用冒泡机制，将子元素的事件委托给父元素去监听(给父元素添加事件)，当子元素触发事件时，事件冒泡到父级如果希望指定的子元素才能触发事件，可以通过事件对象(event)获得事件源(target)，然后通过 条件判断是不是期望的子元素，如果是的话，执行事件，否则不执行

6、事件委托的好处

- 实现对未来元素事件的绑定
- 减少事件绑定，提高性能

7、如何找到事件源

var target = e.target||e.srcElement

tagName能找到事件源的元素名

030、什么是cookie?

会话跟踪技术

> 特点：
>
> 1、大小限制(不能超过4K)
>
> 2、每个域下cookie不能超过50个
>
> 3、有效期（和设定时间有关），过了有效cookie会自动删除
>
> 4、cookie读取（只能访问同一个域名下的cookie）
>
> 5、cookie只能是字符串（文本文件）

031、什么是高内聚、低耦合

> 耦合性：指各个模块之间的联系程度，模块之间的联系越紧密那么耦合性就越高，在面向对象中模块是相对独立的，因此耦合度越低那么越好
>
> 内聚性：内聚指的是模块内部高内聚，模块内部相对独立，独立性越强，内聚度越高。既一个模块内部各个元素彼此之间的紧密联系,联系越紧密内聚度越高
>
> 两者之间并不矛盾，就相当于严以律己、宽以待人

032、ES6的Symbol的作用是什么?

> ES6引入了一种新的原始数据类型Symbol，表示独一无二的值 。一般用来做对象的key值

033、面向对象的好处

> 1、开发时间短，效率高，可靠性高，所开发的程序更强壮。由于面向对象编程的可重用性，可以在应用程序中大量采用成熟的类库，从而缩短了开发时间。
>
> 2、应用程序更易于维护、更新和升级。继承和封装使得应用程序的修改带来的影响更加局部化。

034、prototype的作用

> 1、节约内存
>
> 2、扩展属性和方法
>
> 3、可以实现类的继承

035、在执行new的过程中js执行了哪些操作?

> 1、在内存中开辟了一块空间
>
> 2、把this指向了当前对象

036、get与post的区别

> 1、post是通过HTTP post机制，用户看不到这个过程 。如果想要看到数据可以从控制台NetWork中的form Data中进行查看。
>
> 2、get进行数据请求的时候会将传递的参数信息通过url进行传递。在地址的？后面按照key=val进行传递如果需要传递多个数据的时候用&符进行分隔
>
> 3、 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。
>
> 4、 get安全性非常低，post安全性相比较get来说较高。但是执行效率却比Post方法好。get与post的安全性取决于http协议或者https协议

037、ajax请求数据的流程

> 1、创建通信对象
>
> - IE7及其以上版本中支持原生的 XHR 对象，因此可以直接使用
>
> var xhr = new XMLHttpRequest()
>
> - IE6及其之前版本中,XHR对象是通过MSXML库中的一个ActiveX对象实现的
>
> var xhr = new ActiveXObject("Microsoft,XMLHTTP");
>
> 2、链接和发送
>
> - open() 函数参数有三个：请求方式，请求地址，是否异步请求（同步请求的情况特别少）
>
> xhr.open('get','http://www.baidu.com',true)
>
> - GET 请求方式是通过URL参数将数据提交到服务器的，POST则是通过将数据作为 send的参数传递
> - xhr.send() 发送请求
>
> 3、监听服务器是否返回数据
>
> - 使用onreadystatechange事件监听服务器返回状态
>
> xhr.onreadystatechange = function(){}

038、什么是同源策略?

> 同源指的是域名、协议、端口号相同
>
> 同源策略规定了js代码的访问权限，只能访问和自己同源的页面。
>
> 同源策略是一种约定，它是浏览器最核心也最基本的安全功能

039、什么是跨域？如何解决跨域？ jsonp的原理？

> 1、由于浏览器的同源策略，即属于不同域的页面之间不能相互访问各自的页面内容。
>
> 2、跨域方式
>
> - CORS
> - 服务器代理
> - nginx
> - jsonp
>
> 3、jsonp原理
>
> 利用浏览器的"漏洞" src不受同源策略的影响，可以请求任何链接 。动态创建script标签，将事先写好的函数名传给服务器，供服务器使用

040、什么是jsonp? jsonp跨域的流程?

> jsonp是一种非正式传输协议，用于解决跨域问题
>
> 流程：
>
> 1、创建一个全局函数
>
> 2、创建一个script标签
>
> 3、给script添加src
>
> 4、给src添加回调函数test(callback=test) callback是传给后端的一个参数
>
> 5、将script放到页面上
>
> 6、script请求完成，将自己从页面上删除

041、简术你对promise的理解

> 1、什么是promise?
>
> 异步操作的同步代码
>
> 2、promise的基本使用
>
> 通过new promise创建一个promise对象，里面有一个参数，参数是一个回调函数，回调函数中有2个参数，resolve，reject resolve()当异步执行成功的时候调用的方法，reject()当异步失败的时候调用的方法。

除此之外promise有一个then方法，当成功的时候执行第一个回调函数，当失败的时候执行第二个回调函数。第二个回调函数也可以通过promise对象.catch调用

3、Promise.all():当所有的异步代码都执行完毕以后才会执行.then中的操作

4、Promise.race():只要有一个promise执行完毕后就会执行.then操作

042、请简述prototype、proto constructor三者的关系

1、prototype：

每一个函数都有一个prototype这个属性，而这个属性指向一个对象，这个对象我们叫做原型对象

作用：

- 节约内存
- 扩展属性和方法
- 可以实现类之间的继承

2、__ proto __

- 每一个对象都有一个**proto**属性
- `__proto__`指向创建自己的那个构造函数的原型对象
- 对象可以直接访问`__proto__`里面的属性和方法

3、constructor：

指向创建自己的那个构造函数

总结：

当我们创建一个构造函数的时候这个构造函数自带了一个prototype属性，而这个属性指向一个对象，也就是原型对象。

这个原型对象里面有一个constructor构造器，它的作用是指向创建自己的构造函数。除此之外prototype还可以存放公共的属性和方法。

当我们实例化一个对象的时候，这个对象自带了一个 __ proto __ 属性，这个 __ proto __ 指向创建自己的构造函数的原型对象。可以使用这个原型对象里面的属性和方法

043、请写出方法继承的方式

1、call 、apply:不建议使用浪费内存

> 2、原型对象继承
>
> 3、原型拷贝继承
>
> 4、原型链继承
>
> 5、混合继承
>
> 6、继承继承
>
> 7、ES6 类继承

044、请说出call、apply、bind的区别

> bind:bind绑定完this的指向后会返回一个新的函数体，不会被立即调用
>
> call&apply:绑定完this的指向后会立即调用
>
> call与apply的区别：
>
> call:第一个参数是this的指向，第二个以及后面的所有参数需要一个个进行传递
>
> apply:第一个参数是this的指向，第二个参数是一个数组

045、请解释一下什么叫同源策略，以及为什么浏览器会有同源策略

> 同源策略是浏览器的一个安全功能，不同源的客户端在没有授权的情况下，不能读取对方资源
>
> 为了保障数据的安全，即非同源网页不可请求

046、什么是闭包？用途？注意的地方？

> 1、闭包:打破了作用域链的规则
>
> 2、用途：a:可以读取函数内部的局部变量 b:让这些变量始终保持在内存当中
>
> 3、注意：由于闭包会使得函数中的变量都被保存在内存当中，内存会消耗很大，所以不能够滥用闭包，否则会造成网页性能的问题

047、Jquery中attr与prop的区别

> attr：是通过`setAttribute`和`getAttribute`来设置的使用的是DOM属性节点
>
> prop: 是通过`document.getElementById()[name] = value`来实现的,通常用来设置`checked selected`

048、window.onload、window.onresize、window.onscroll、$(document).ready(function(){})四者的区别

> 1、window.onload:当文档加载完毕以后 包括html,js,img,css

> 2、window.onresize：当窗口发送改变的时候,高频率出发事件
>
> 3、window.onscroll:当滚动条滚动的时候，高频率出发事件
>
> 4、$(document).ready(function()({}) 当页面加载完毕以后 不包括img

049、jquery中 width()、innerWidth、outerWidth的区别

> width():只会获取content内容区的宽度
>
> innerWidth():会获取content+padding的宽度
>
> outerWidth():会获取content+padding+border的宽度

050、jquery中offset()、position()、scrollTop()的作用

> offset():获取当前元素距离页面之间的偏移量
>
> position():获取当前元素距离以定位的父元素的偏移量
>
> scrollTop():获取滚动条滚动的距离

051、$(document).height()$与(window).height()的区别

> $(documemt).height()：获取整个页面的高度 类似于原生js里面的document.body.clientHeight
>
> $(window).height():获取可视区的高度 类似于原生js里面的 document.documentElement.clientHeight

052、jquery中事件绑定、委托中 bind() live() delegate()、on()之间的区别

> bind:为每个元素绑定事件处理函数
>
> 解绑事件：unbind()
>
> 缺点：
>
> 无法对未来元素实现事件绑定
>
> live:为所有匹配的元素添加事件处理函数，未来元素也可以绑定事件处理函数
>
> 解绑事件：die()
>
> 缺点：
>
> - 1.7版本后不再支持该方法
> - 阻止事件冒泡不管用

> live:是通过冒泡的方式来绑定到元素上的。更适合列表类型的，绑定到document DOM节点上。一旦事件冒泡到document上，jQuery将会查找selector/event metadata,然后决定那个handler应该被调用
>
> delegate:为指定的元素(子元素)添加一个或多个事件处理函数
>
> 解绑事件：undelegate()
>
> 在某些浏览器下是有兼容性问题
>
> 更精确的小范围使用事件代理，性能优于.live()。它不会把所有的event全部绑定到document,而是由你决定把它放在哪儿
>
> on: 为匹配的元素绑定一个活多个事件处理函数
>
> 解绑事件：off()
>
> 缺点：
>
> on不能取代live
>
> 整合了之前的三种方式的新事件绑定机制。.bind(), .live(), .delegate()都是通过.on()来实现的，.unbind(), .die(), .undelegate(),也是一样的都是通过.off()来实现的。

053、$(this) 和 this 关键字在 jQuery 中有何不同

> $(this) 返回一个 jQuery 对象，你可以对它调用多个 jQuery 方法
>
> 而 this 代表当前元素，它是 JavaScript 关键词中的一个，表示上下文中的当前 DOM 元素。你不能对它调用 jQuery 方法

054、jQuery 中的方法链是什么？使用方法链有什么好处

> 方法链是对一个方法返回的结果调用另一个方法，这使得代码简洁明了，同时由于只对 DOM 进行了一轮查找，性能方面更加出色。

055、哪种方式更高效：document.getElementbyId("myId") 还是 $("#myId")?

> 第一种，因为它直接调用了 JavaScript 引擎

056、JQ中find()、has()和filter()区别?

> filter()方法，条件作用于自身
>
> has()方法条件是作用于它的后代元素中
>
> find():当前选中元素的上下文中找到符合条件的后代，返回的是子元素

057、模块化开发的优点

> 1、解决文件之间的依赖关系
>
> 2、避免命名冲突 、解决全局变量级全局函数泛用的现象
>
> 3、解决代码的复杂性
>
> 4、按需加载

058、常用的模块化的方案有哪些?

> 1、服务端 (commonJS) 例如 Node.js
>
> 2、客户端(AMD CMD) 例如 require sea.js
>
> 3、ES6：module(export import);

059、什么是require.js作用是什么？什么是AMD?

> RequireJS是一个JavaScript文件或者**模块的加载器**。它可以提高JavaScript文件的加载速度，避免不必要的堵塞。
>
> requireJS的作用：
>
> 1、实现js的异步加载
>
> 2、管理模块之间的依赖关系，便于代码的编写和维护
>
> AMD是一种**定义和加载模块的规则**，使模块和它的依赖可以被异步的加载，但又按照正确的顺序

060、AMD与CMD的区别

> AMD 推崇**依赖前置**
>
> CMD 推崇**就近**

061、什么是css预处理器？优点是什么?

> css预处理器用一种专门的编程语言，进行web页面样式设计，然后在编译成正常的css文件，以供项目使用
>
> 在css中使用 变量、简单的逻辑程序、函数。可以让你的css更加简洁、适应性更强、可读性更佳、更易于代码的维护

062、请描述一下 cookies，sessionStorage 和 localStorage 的区别?

> localStorage 长期存储数据，浏览器关闭后数据不丢失;
>
> sessionStorage 数据在浏览器关闭后自动删除。
>
> cookie在浏览器和服务器间来回传递。 sessionStorage和localStorage不会
>
> sessionStorage和localStorage的存储空间更大;
>
> sessionStorage和localStorage有更多丰富易用的接口;
>
> sessionStorage和localStorage各自独立的存储空间

063、eval是做什么的?

> 它的功能是把对应的字符串解析成JS代码并运行;
>
> 应该避免使用eval，不安全，非常耗性能(2次，一次解析成js语句，一次执行)

064、谈谈This对象的理解

> this是js的一个关键字，随着函数使用场合不同，this的值会发生变化。
>
> 但是有一个总原则，那就是this指的是调用函数的那个对象。
>
> this一般情况下：是全局对象。 作为方法调用，那么this就是指这个对象

065、IE与火狐的事件机制有什么区别?

> 事件处理机制：IE是事件冒泡、火狐是 事件捕获;

066、new操作符具体干了什么呢

> 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 。

067、你有哪些性能优化的方法

> (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。

068、对前端界面工程师这个职位是怎么样理解的?它的前景会怎么样?

前端是最贴近用户的程序员，比后端、数据库、产品经理、运营、安全都近。

1、实现界面交互

2、提升用户体验

3、有了Node.js，前端可以实现服务端的一些事情

4、前端是最贴近用户的程序员，前端的能力就是能让产品从 90分进化到 100 分，甚至更好，

5、参与项目，快速高质量完成实现效果图，精确到1px;

6、与团队成员，UI设计，产品经理的沟通;

7、做好的页面结构，页面重构和用户体验;

8、处理hack，兼容、写出优美的代码格式;

9、针对服务器的优化、拥抱最新前端技术。

069、http的原理

1、工作原理： 客户机与服务器建立连接后，发送一个请求给服务器，请求格式为：统一资源标识符、协议版本号。服务器收到请求的信息（包括请求行，请求头，请求体）。服务器接收到请求后，给予相应的响应信息，格式为一个状态行（包括响应行，响应头，响应体）。

2、基于HTTP协议的客户/服务器模式的信息交换过程，分为四个过程：建立连接、发送请求信息、发送响应信息、关闭连接。

3、 服务器可能同时接受多个请求，这时就会产生多个sessoin，每个session分别处理各自的请求。

070、谈谈你对缓存的理解

1.定义：缓存就相当于是对资源的一种副本实现，不管是在客户端还是在服务端存储着，用相同的URL进行请求，直接从副本中请求资源而不再访问源服务器。

2.为什么使用缓存

> a.提高访问速度：缓存相对服务端离用户更近，所以在请求过程中从缓存中取内容比在源服务器上取的内容用的时间更少，加快了用户体验。
>
> b.降低网络传输：副本被重复使用，大大降低了用户的带宽使用，其实也是一种变相的省钱（如果流量要付费的话），同时保证了带宽请求在一个低水平上，更容易维护了。
>
> 3.缓存的种类：缓存种类很多，像是浏览器缓存，cdn缓存等都是我们比较熟悉的，当然还有代理服务器缓存，网关缓存等

071、实现延迟加载的方式

> 1、直接将script节点放置在前后，这样js脚本就会在页面显示出来之后再加载。
>
> 2、使用script标签的defer和async属性，defer属性为延迟加载，是在页面渲染完成之后再进行加载的，而async属性则是和文档并行加载，这两种解决方案都不完美，原因在于不是所有浏览器都支持。
>
> 3、通过监听window.onload事件，动态添加script节点：
>
> 4、通过ajax下载js脚本，动态添加script节点，但是ajax有一个缺点，就是无法引用使用CDN方式提供的js文件
>
> 5、使用setTimeout延迟加载

072、jquery.extend 与 jquery.fn.extend的区别

> Jquery.extend用来扩展jQuery对象本身；jquery.fn.extend用来扩展jQuery实例

073、JQuery一个对象可以同时绑定多个事件，这是如何实现的

> 可以同时绑定多个事件，底层实现原理是使用addEventListner与attachEvent兼容处理做事件注册

未完待续-----持续更新

Javascript常见兼容性问题

001、获取滚动条滚动的距离

```
var sTop = document.documentElement.scrollTop || document.body.scrollTop
```

002获取非行间样式

```
//IE：  currentStyle[attr]
//标准：  getComputedStyle[attr]

function getStyle(obj,attr){
    if(obj.currentStyle){
        return obj.currentStyle[attr]
    }else{
        return getComputedStyle(obj,false)[attr];
    }
}
```

003获取事件对象

```
var e = e || event;
```

004获取键盘信息

```
e.keyCode || e.which
```

005阻止浏览器的默认行为

```
function prevent(e){
    if(e.preventDefault){
        e.preventDefault();
    }else{
        e.returnValue = false;
    }
}
```

006阻止事件冒泡

```
e.stopPropagation?e.stopPropagation():e.cancelBubble = true;
```

007事件监听

```
addEventListener()
attachEvent()
```

008事件解绑

```
removeEventListener()
detachEvent()
```

009获取事件源

```
e.target || e.srcElement;
```

010ajax兼容

```
var xhr = new XMLHttpRequest() || new ActiveXObject("Microsoft,XMLHTTP");
```