## 高级前端工程师面试题

## 一、HTML

**1. html5 有哪些新特性、移除了那些元素？如何处理 HTML5 新标签的浏览器兼容问题？ 如何区分 HTML 和 HTML5？**

(Q1) HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加。

(1)绘画 canvas;

(2)用于媒介回放的 video 和 audio 元素;

(3)本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失;

(4)sessionStorage 的数据在浏览器关闭后自动删除;

(5)语意化更好的内容元素，比如 article、footer、header、nav、section; (6)表单控件，calendar、date、time、email、url、search;

(7)新的技术webworker, websocket, Geolocation;

(Q2) IE8/IE7/IE6支持通过document.createElement方法产生的标签， 可以利用这一特性让这些浏览器支持HTML5新标签， 浏览器支持新标签后，还需要添加标签默认的样式。 也可以直接使用成熟的框架、比如html5shim，

**2.**请描述一下 cookies，sessionStorage 和 localStorage 的区别？

cookie是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。

cookie数据始终在同源的http请求中携带（即使不需要），记会在浏览器和服务器间来回传递。

sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。

存储大小：

cookie数据大小不能超过4k。

sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。

有期时间：

localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动删除数据；

sessionStorage 数据在当前浏览器窗口关闭后自动删除。

cookie 设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭

**3.HTML5 为什么只需要写 <!DOCTYPE HTML>？**

HTML5 不基于 SGML，因此不需要对DTD进行引用，但是需要doctype来规范浏览器的行为（让浏览器按照它们应该的方式来运行）；

而HTML4.01基于SGML,所以需要对DTD进行引用，才能告知浏览器文档所使用的文档类型。

**4. 实现不使用 border 画出1px高的线，在不同浏览器的标准模式与怪异模式下都能保持一致的效果。**  
<div style="height:1px;overflow:hidden;background:red"></div>

**5.** 网页验证码是干嘛的，是为了解决什么安全问题。

区分用户是计算机还是人的公共全自动程序。可以防止恶意破解密码、刷票、论坛灌水；

有效防止黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试。

## 二、CSS

1.CSS3新增伪类举例：

　　p:first-of-type 选择属于其父元素的首个元素的每个元素。

　　p:last-of-type  选择属于其父元素的最后元素的每个元素。

　　p:only-of-type  选择属于其父元素唯一的元素的每个元素。

　　p:only-child    选择属于其父元素的唯一子元素的每个元素。

　　p:nth-child(2)  选择属于其父元素的第二个子元素的每个元素。

　　:enabled  :disabled 控制表单控件的禁用态。

　　:checked单选框或复选框被选中。

**2.** CSS3有哪些新特性？

新增各种CSS选择器 （: not(.input)：所有 class 不是“input”的节点）

圆角 （border-radius:8px）

多列布局 （multi-column layout）

阴影和反射 （Shadow\Reflect）

文字特效 （text-shadow、）

文字渲染 （Text-decoration）

线性渐变 （gradient）

旋转 （transform）

缩放,定位,倾斜,动画,多背景

例如:transform:\scale(0.85,0.90)\ translate(0px,-30px)\ skew(-9deg,0deg)\Animation:

3. 用纯CSS创建一个三角形的原理是什么？

把上、左、右三条边隐藏掉（颜色设为 transparent）

#demo {

width: 0;

height: 0;

border-width: 20px;

border-style: solid;

border-color: transparent transparent red transparent;

}

4. canvas 在标签上设置宽高 和在 style 中设置宽高有什么区别？

canvas 标签的 width 和 height 是画布实际宽度和高度，绘制的图形都是在这个上面。而 style的 width 和 height 是 canvas 在浏览器中被渲染的高度和宽度。如果 canvas 的 width 和 height没指定或值不正确，就被设置成默认值 。

5. canvas 如何绘制一个三角形|正方形

moveto 是移动到某个坐标， lineto 是从当前坐标连线到某个坐标。这两个函数加起来就是 画一条直线。 画线要用“笔”，那么 MoveTo()把笔要画的起始位置固定了（x,y）然后要固定 终止位置要用到 LineTo 函数确定终止位置（xend,yend）,这样一条线就画出来了。 每次与 前面一个坐标相连 。

stroke() 方法会实际地绘制出通过 moveTo() 和 lineTo() 方法定义的路径。默认颜色是黑 色。

<!DOCTYPE HTML5> <html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <title>画布</title>

</head>

<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;"> Your browser does not support the canvas element.

</canvas>

<script type="text/javascript">

var c=document.getElementById("myCanvas"); //三角形

var cxt=c.getContext("2d"); cxt.moveTo(10,10); cxt.lineTo(50,50); cxt.lineTo(10,50); cxt.lineTo(10,10); cxt.stroke();

//正方形

var cxt2=c.getContext("2d"); cxt2.moveTo(60,10); cxt2.lineTo(100,10); cxt2.lineTo(100,50); cxt2.lineTo(60,50); cxt2.lineTo(60,10); cxt2.stroke();

</script>

</body>

</html>

6. 什么是CSS 预处理器 / 后处理器？

预处理器例如：LESS、Sass、Stylus，用来预编译Sass或less，增强了css代码的复用性，还有层级、mixin、变量、循环、函数等，具有很方便的UI组件模块化开发能力，极大的提高工作效率。

后处理器例如：PostCSS，通常被视为在完成的样式表中根据CSS规范处理CSS，让其更有效；目前最常做的是给CSS属性添加浏览器私有前缀，实现跨浏览器兼容性的问题。

7. sass 和 less 有什么区别?

a.编译环境不一样

Sass 的安装需要 Ruby 环境，是在服务端处理的，而 Less 是需要引入 less.js 来处理 Less 代码输出 css 到浏览器，也可以在开发环节使用 Less，然后编译成 css 文件，直接放到项 目中。

b.变量符不一相，less 是@，而 scss 是$,而且它们的作用域也不一样，less 是块级作用域

c.输出设置，Less 没有输出设置，sass 提供 4 种输出选项，nested,compact,compressed 和 expanded

nested：嵌套缩进的 css 代码(默认)

expanded：展开的多行 css 代码

compact：简洁格式的 css 代码

compressed：压缩后的 css 代码

8.什么是面向对象的CSS(OOCSS)？

面向对象的CSS是一种容易重用的一种CSS规则，也是OOP的概念，从而降低了页面的加载时间，提高了网面的性能。这里关键的一点就是如何在页面中识别，创建和模块化可重用的对象，并在页面中任何需要的地方重用，并扩展其附加功能。

## 三、Javascript

**1．JavaScript原型，原型链 ? 有什么特点？**

每个对象都会在其内部初始化一个属性，就是prototype(原型)，当我们访问一个对象的属性时，

如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，

于是就这样一直找下去，也就是我们平时所说的原型链的概念。

关系：instance.constructor.prototype = instance.\_\_proto\_\_

特点：

JavaScript对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。当我们修改原型时，与之相关的对象也会继承这一改变。

当我们需要一个属性的时，Javascript引擎会先看当前对象中是否有这个属性， 如果没有的话，

就会查找他的Prototype对象是否有这个属性，如此递推下去，一直检索到 Object 内建对象。

function Func(){}

Func.prototype.name = "Sean";

Func.prototype.getInfo = function() {

return this.name;

}

var person = new Func();//现在可以参考var person = Object.create(oldObject);

console.log(person.getInfo());//它拥有了Func的属性和方法

//"Sean"

console.log(Func.prototype);

// Func { name="Sean", getInfo=function()}

2. 什么是闭包（closure），为什么要用它？

闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。

闭包的特性：

1.函数内再嵌套函数

2.内部函数可以引用外层的参数和变量

3.参数和变量不会被垃圾回收机制回收

//li节点的onclick事件都能正确的弹出当前被点击的li索引

<ul id="testUL">

<li> index = 0</li>

<li> index = 1</li>

<li> index = 2</li>

<li> index = 3</li>

</ul>

<script type="text/javascript">

var nodes = document.getElementsByTagName("li");

for(i = 0;i<nodes.length;i+= 1){

nodes[i].onclick = (function(i){

return function() {

console.log(i);

} //不用闭包的话，值每次都是4

})(i);

}

</script>

执行say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在

使得Javascript的垃圾回收机制GC不会收回say667()所占用的资源

因为say667()的内部函数的执行需要依赖say667()中的变量

这是对闭包作用的非常直白的描述

function say667() {

// Local variable that ends up within closure

var num = 666;

var sayAlert = function() {

alert(num);

}

num++;

return sayAlert;

}

var sayAlert = say667();

sayAlert()//执行结果应该弹出的667

3. javascript 代码中的"use strict";是什么意思 ? 使用它区别是什么？

use strict是一种ECMAscript 5 添加的（严格）运行模式,这种模式使得 Javascript 在更严格的条件下运行,

使JS编码更加规范化的模式,消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为。

默认支持的糟糕特性都会被禁用，比如不能用with，也不能在意外的情况下给全局变量赋值;

全局变量的显示声明,函数必须声明在顶层，不允许在非函数代码块内声明函数,arguments.callee也不允许使用；

消除代码运行的一些不安全之处，保证代码运行的安全,限制函数中的arguments修改，严格模式下的eval函数的行为和非严格模式的也不相同;

提高编译器效率，增加运行速度；

为未来新版本的Javascript标准化做铺垫。

4. new操作符具体干了什么呢?

a、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。

b、属性和方法被加入到 this 引用的对象中。

c、新创建的对象由 this 所引用，并且最后隐式的返回 this 。

var obj = {};

obj.\_\_proto\_\_ = Base.prototype;

Base.call(obj);

5. Ajax 是什么? 如何创建一个Ajax？

ajax的全称：Asynchronous Javascript And XML。

异步传输+js+xml。

所谓异步，在这里简单地解释就是：向服务器发送请求的时候，我们不必等待结果，而是可以同时做其他的事情，等到有了结果它自己会根据设定进行后续操作，与此同时，页面是不会发生整页刷新的，提高了用户体验。

(1)创建XMLHttpRequest对象,也就是创建一个异步调用对象

(2)创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息

(3)设置响应HTTP请求状态变化的函数

(4)发送HTTP请求

(5)获取异步调用返回的数据

(6)使用JavaScript和DOM实现局部刷新

6. 模块化开发怎么做？

立即执行函数不暴露私有成员

var module1 = (function(){

　　　　var \_count = 0;

　　　　var m1 = function(){

　　　　　　//...

　　　　};

　　　　var m2 = function(){

　　　　　　//...

　　　　};

　　　　return {

　　　　　　m1 : m1,

　　　　　　m2 : m2

　　　　};

　　})();

7. AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

Asynchronous Module Definition，异步模块定义，所有的模块将被异步加载，模块加载不影响后面语句运行。所有依赖某些模块的语句均放置在回调函数中。

区别：

1. 对于依赖的模块，AMD 是提前执行，CMD 是延迟执行。不过 RequireJS 从 2.0 开始，也改成可以延迟执行（根据写法不同，处理方式不同）。CMD 推崇 as lazy as possible.

2. CMD 推崇依赖就近，AMD 推崇依赖前置。看代码：

// CMD

define(function(require, exports, module) {

var a = require('./a')

a.doSomething()

// 此处略去 100 行

var b = require('./b') // 依赖可以就近书写

b.doSomething()

// ...

})

// AMD 默认推荐

define(['./a', './b'], function(a, b) { // 依赖必须一开始就写好

a.doSomething()

// 此处略去 100 行

b.doSomething()

// ...

})

8. .call() 和 .apply() 的区别？

例子中用 add 来替换 sub，add.call(sub,3,1) == add(3,1) ，所以运行结果为：alert(4);

注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

function add(a,b) {

alert(a+b);

function sub(a,b)

{

alert(a-b);

}

add.call(sub,3,1);

call 和 apply 都是改变 this 指向的方法，区别在于 call 可以写多个参数，而 apply 只能写 两个参数，第二个参数是一个数组，用于存放要传的参数。

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

Object.apply(this,arguments)

**9. 你有用过哪些前端性能优化的方法？**

（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。

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。

10. http状态码有那些？分别代表是什么意思？

100 Continue 继续，一般在发送post请求时，已发送了http header之后服务端将返回此信息，表示确认，之后发送具体参数信息

200 OK 正常返回信息

201 Created 请求成功并且服务器创建了新的资源

202 Accepted 服务器已接受请求，但尚未处理

301 Moved Permanently 请求的网页已永久移动到新位置。

302 Found 临时性重定向。

303 See Other 临时性重定向，且总是使用 GET 请求新的 URI。

304 Not Modified 自从上次请求后，请求的网页未修改过。

400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。

401 Unauthorized 请求未授权。

403 Forbidden 禁止访问。

404 Not Found 找不到如何与 URI 相匹配的资源。

500 Internal Server Error 最常见的服务器端错误。

503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。

11. 一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？

浏览器根据请求的URL交给DNS域名解析，找到真实IP，向服务器发起请求；

服务器交给后台处理完成后返回数据，浏览器接收文件（HTML、JS、CSS、图象等）；

浏览器对加载到的资源（HTML、JS、CSS等）进行语法解析，建立相应的内部数据结构（如HTML的DOM）；

载入解析到的资源文件，渲染页面，完成。

12. 事件流?事件捕获？事件冒泡？

事件流：从页面中接收事件的顺序。也就是说当一个事件产生时，这个事件的传播过程，就 是事件流。

IE 中的事件流叫事件冒泡；事件冒泡：事件开始时由最具体的元素接收，然后逐级向上传 播到较为不具体的节点（文档）。对于 html 来说，就是当一个元素产生了一个事件，它会 把这个事件传递给它的父元素，父元素接收到了之后，还要继续传递给它的上一级元素，就 这样一直传播到 document 对象（亲测现在的浏览器到 window 对象，只有 IE8 及下不这样）； 事件捕获是不太具体的元素应该更早接受到事件，而最具体的节点应该最后接收到事件。他 们的用意是在事件到达目标之前就捕获它；也就是跟冒泡的过程正好相反，以 html 的 click 事件为例，document 对象（DOM 级规范要求从 document 开始传播，但是现在的浏览器 是从 window 对象开始的）最先接收到 click 事件的然后事件沿着 DOM 树依次向下传播， 一直传播到事件的实际目标；

13. 如何清除一个定时器?

window.clearInterval();

window.clearTimeout();

14. 回调函数?

回调函数就是一个通过函数指针调用的函数。如果你把函数的指针（地址）作为参数传

递给另一个函数，当这个指针被用来调用其所指向的函数时，我们就说这是回调函数。回调 函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用 的，用于对该事件或条件进行响应。

15. 那些操作会造成内存泄漏？怎么样防 止内存泄漏？

内存泄露是指：用动态存储分配函数内存空间，在使用完毕后未释放，导致一直占据该内存 单元。直到程序结束。指任何对象在您不再拥有或需要它之后仍然存在。

造成内存泄漏：

setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。 闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环） 防止内存泄露：

a、不要动态绑定事件；

b、不要在动态添加，或者会被动态移除的 dom 上绑事件，用事件冒泡在父容器监听事件；

c、如果要违反上面的原则，必须提供 destroy 方法，保证移除 dom 后事件也被移除，这点 可以参考 Backbone 的源代码，做的比较好；

d、单例化，少创建 dom，少绑事件。

16. 什么是构造函数？与普通函数有什么区别?

造函数：是一种特殊的方法、主要用来创建对象时初始化对象，总与 new 运算符一起使 用，创建对象的语句中构造函数的函数名必须与类名完全相同。

与普通函数相比只能由 new 关键字调用，构造函数是类的标示

17. git 和 svn 的区别?

SVN 是集中式版本控制系统，版本库是集中放在中央服务器的，而干活的时候，用的都是 自己的电脑，所以首先要从中央服务器哪里得到最新的版本，然后干活，干完后，需要把自 己做完的活推送到中央服务器。集中式版本控制系统是必须联网才能工作，如果在局域网还 可以，带宽够大，速度够快，如果在互联网下，如果网速慢的话，就纳闷了。

Git 是分布式版本控制系统，那么它就没有中央服务器的，每个人的电脑就是一个完整的版 本库，这样，工作的时候就不需要联网了，因为版本都是在自己的电脑上。既然每个人的电 脑都有一个完整的版本库，那多个人如何协作呢？比如说自己在电脑上改了文件 A，其他人 也在电脑上改了文件 A，这时，你们两之间只需把各自的修改推送给对方，就可以互相看到 对方的修改了。

18. 简述下 this 和 定义属性和方法的时候有什么区别?Prototype？

this 表示当前对象，如果在全局作用范围内使用 this，则指代当前页面对象 window； 如果 在函数中使用 this，则 this 指代什么是根据运行时此函数在什么对象上被调用。 我们还可 以使用 apply 和 call 两个全局方法来改变函数中 this 的具体指向。

prototype 本质上还是一个 JavaScript 对象。并且每个函数都有一个默认的 prototype 属性。 在 prototype 上定义的属性方法为所有实例共享，所有实例皆引用到同一个对象，单一实例 对原型上的属性进行修改，也会影响到所有其他实例。

19.this指向

obj.foo() == obj //方法调用模式,this指向obj

　　foo() == window; //函数调用模式,this指向window

　　new obj.foo() == obj //构造器调用模式, this指向新建立对象

　　foo.call(obj) == obj;//APPLY调用模式,this指向obj

20. ajax 中 get 和 post 有什么区别?

get 和 post 都是数据提交的方式。

get 的数据是通过网址问号后边拼接的字符串进行传递的。post 是通过一个 HTTP 包体进行 传递数据的。

get 的传输量是有限制的，post 是没有限制的。

get 的安全性可能没有 post 高，所以我们一般用 get 来获取数据，post 一般用来修改数据。

21. Gc 机制是什么？为什么闭包不会被回收变量和函数？

a、Gc 垃圾回收机制;

b、外部变量没释放，所以指向的大函数内的小函数也释放不了

22. 简述下你理解的面向对象？

万物皆对象，把一个对象抽象成类,具体上就是把一个对象的静态特征和动态特征抽象成

属性和方法,也就是把一类事物的算法和数据结构封装在一个类之中,程序就是多个对象和 互相之间的通信组成的.

面向对象具有封装性,继承性,多态性。 封装:隐蔽了对象内部不需要暴露的细节,使得内部细节的变动跟外界脱离,只依靠接口进行

通信.封装性降低了编程的复杂性. 通过继承,使得新建一个类变得容易,一个类从派生类那 里获得其非私有的方法和公用属性的繁琐工作交给了编译器. 而 继承和实现接口和运行时 的类型绑定机制 所产生的多态,使得不同的类所产生的对象能够对相同的消息作出不同的 反应,极大地提高了代码的通用性.

总之,面向对象的特性提高了大型程序的重用性和可维护性.

23. slice() splice()?

slice() 方法可从已有的数组中返回选定的元素。

splice() 方法向/从数组中添加/删除项目，然后返回被删除的项目。

24. JavaScript 提供了哪几种“异步模式 ”？

a、回调函数（callbacks）

b、事件监听

c、Promise对象

25. 变量作用域?

变量作用域：一个变量的作用域是程序源代码中定义这个变量的区域。全局变量拥有全局作 用域，在 js 代码中任何地方都是有定义的。在函数内声明的变量只在函数体内有定义，它 们是局部变量，作用域是局部性的。函数参数也是局部变量，它们只在函数体内有定义。 var a = ""

window.b=''”; function(e) {

var c= ""; d=""; e="";

}

function go() { console.info(this);//window return function() {

console.info(this); // window return {

b:function(){

console.info(this); //b 的父对象

}

}

}

}

go()().b();

26. 开发时如何对项目进行管理?gulp?

gulp 是新一代的前端项目构建工具，你可以使用 gulp 及其插件对你的项目代码（less,sass） 进行编译，还可以压缩你的 js 和 css 代码，甚至压缩你的图片，能够合并文件，压缩文件， 语法检查，监听文件变化，测试，转换二进制，转换图片等一系列功能。gulp 仅有少量的 API，所以非常容易学习。实现良好的项目管理。

27. ajax 请求方式有几种（8 种）？

1）$.get(url,[data],[callback])

2）$.getJSON(url,[data],[callback])

3）$.post(url,[data],[callback],[type])

4）$.ajax(opiton)

5）$.getScript( url, [callback] )

6）jquery 对象.load( url, [data], [callback] )

7）serialize() 与 serializeArray()

28.IE和标准下有哪些兼容性的写法

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

**29**.javascript的本地对象，内置对象和宿主对象

本地对象为array obj regexp等可以new实例化  
内置对象为gload Math 等不可以实例化的  
宿主为浏览器自带的document,window 等

30.”==”和“===”的不同

前者会自动转换类型  
后者不会

**31.写一个function，清除字符串前后的空格。（兼容所有浏览**器）

function trim(str) {

if (str && typeof str === "string") {

return str.replace(/(^\s\*)|(\s\*)$/g,""); //去除前后空白符

}

}

32.使用正则表达式验证邮箱格式

var reg = /^(\w)+(\.\w+)\*@(\w)+((\.\w{2,3}){1,3})$/;

var email = "example@qq.com";

console.log(reg.test(email)); // true

33、线程与进程的区别

　　一个程序至少有一个进程,一个进程至少有一个线程。

　　线程的划分尺度小于进程，使得多线程程序的并发性高。

　　另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。

　　线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。

　　从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。

34.javascript常用的设计模式有哪些？

**a. 单例模式**

单例模式的定义是产生一个类的唯一实例，但js本身是一种“无类”语言。很多讲js设计模式的文章把{}当成一个单例来使用也勉强说得通。因为js生成对象的方式有很多种，我们来看下另一种更有意义的单例。

**b. 简单工厂模式**

简单工厂模式是由一个方法来决定到底要创建哪个类的实例, 而这些实例经常都拥有相同的接口.

var request1 = Request('cgi.xx.com/xxx' , ''get' );

request1.start();

request1.done( fn );

var request2 = Request('cgi.xx.com/xxx' , ''jsonp' );

request2.start();

request2.done( fn );

Request实际上就是一个工厂方法, 至于到底是产生xhr的实例, 还是jsonp的实例. 是由后来的代码决定的。

实际上在js里面，所谓的构造函数也是一个简单工厂。只是批了一件new的衣服. 我们扒掉这件衣服看看里面。

通过这段代码, 在firefox, chrome等浏览器里，可以完美模拟new.

**c.观察者模式**

var adultTv = Event();

adultTv .listen( ''play', function( data ){

alert ( "今天是谁的电影" + data.name );

});

//发布者

adultTv .trigger( ''play', { 'name': '麻生希' } )

35.跨域请求原理

a、通过jsonp跨域

利用在页面中创建<script>节点的方法向不同域提交HTTP请求的方法称为JSONP，这项技术可以解决跨域提交Ajax请求的问题。

js跨域是指通过js在不同的域之间进行数据传输或通信，比如用ajax向一个不同的域请求数据，或者通过js获取页面中不同域的框架中(iframe)的数据。只要协议、域名、端口有任何一个不同，都被当作是不同的域。

要解决跨域的问题，我们可以使用以下几种方法：

jsonp的原理就很清楚了，通过script标签引入一个js文件，这个js文件载入成功后会执行我们在url参数中指定的函数，并且会把我们需要的json数据作为参数传入。所以jsonp是需要服务器端的页面进行相应的配合的。

jquery的$.getJSON方法会自动判断是否跨域，不跨域的话，就调用普通的ajax方法；跨域的话，则会以异步加载js文件的形式来调用jsonp的回调函数。

b、通过修改document.domain来跨子域

修改document.domain的方法只适用于不同子域的框架间的交互。如果你想通过ajax的方法去与不同子域的页面交互，除了使用jsonp的方法外，还可以用一个隐藏的iframe来做一个代理。原理就是让这个iframe载入一个与你想要通过ajax获取数据的目标页面处在相同的域的页面

c、使用window.name来进行跨域

window对象有个name属性，该属性有个特征：即在一个窗口(window)的生命周期内,窗口载入的所有的页面都是共享一个window.name的，每个页面对window.name都有读写的权限，window.name是持久存在一个窗口载入过的所有页面中的，并不会因新页面的载入而进行重置。

window.name的值只能是字符串的形式，这个字符串的大小最大能允许2M左右甚至更大的一个容量，具体取决于不同的浏览器，但一般是够用了。

能兼容到几乎所有浏览器。

d、使用HTML5中新引进的window.postMessage方法来跨域传送数据

使用postMessage来跨域传送数据还是比较直观和方便的，但是缺点是IE6、IE7不支持，所以用不用还得根据实际需要来决定。

除了以上几种方法外，还有flash、在服务器上设置代理页面等跨域方式。