

实例：

Var a = 1&&0 //0

Var b = 1&&8 //8

Var c = 1&&3&&5 //5

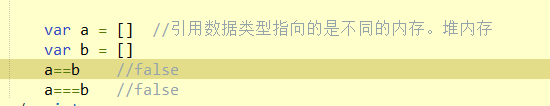
Var d = 0&&2&&5 //0

Var e = 0 || 1 //1

var a = 0 && 1 || 2 //2

parseInt(true)输出什么？ // NaN

return 后面如果不直接写 { 解析器自动给它后面加一个；



**javaScript的3种主要数据类型，2种复合数据类型和2种特殊数据类型。**

主要数据类型：string, boolean, number

复合数据类型：function, object

特殊类型：undefined，null

**一、documen.write和 innerHTML的区别？**

document.write只能重绘整个页面

innerHTML可以重绘页面的一部分

**二、请用5分钟介绍闭包,以及相关应用**

闭包就是能够读取其他函数内部变量的函数。由于在Javascript语言中，只有函数内部的子函数才能读取局部变量，因此可以把闭包简单理解成“定义在一个函数内部的函数”。

所以，在本质上，闭包就是将函数内部和函数外部连接起来的一座桥梁。闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。

使用闭包的注意点：由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。闭包会在父函数外部，改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public?Method），把内部变量当作它的私有属性（private?value），这时一定要小心，不要随便改变父函数内部变量的值。

**三、平时如何管理你的项目？**

先期团队必须确定好全局样式（globe.css），编码模式(utf-8) 等；

编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；

标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；

页面进行标注（例如 页面 模块 开始和结束）；

CSS跟HTML 分文件夹并行存放，命名都得统一（例如style.css）；

JS 分文件夹存放 命名以该JS功能为准的英文翻译。

图片采用整合的 images.png png8 格式文件使用 尽量整合在一起使用方便将来的管理

**四、几条写JavaScript的基本规范？**

1.不要在同一行声明多个变量。

2.请使用 ===/!==来比较true/false或者数值

3.使用对象字面量替代new Array这种形式

4.不要使用全局函数。

5.Switch语句必须带有default分支

6.函数不应该有时候有返回值，有时候没有返回值。

7.For循环必须使用大括号

8.If语句必须使用大括号

9.for-in循环中的变量 应该使用var关键字明确限定作用域，从而避免作用域污染。

**五、什么是伪数组？**

伪数组（类数组）：

无法直接调用数组方法或期望length属性有什么特殊的行为，不具有数组的push,pop等方法，但仍可以对真正数组遍历方法来遍历它们。典型的是函数的argument参数，还有像调用getElementsByTagName,document.childNodes之类的,它们都返回NodeList对象都属于伪数组。可以使用Array.prototype.slice.call(fakeArray)将数组转化为真正的Array对象。

这里把符合以下条件的对象称为伪数组：

1. 具有length属性  
   2，按索引方式存储数据  
   3，不具有数组的push,pop等方法

**>>>**转换为真正的数组的方法：

通过遍历将伪数组里元素放入到一个新的数组里

通过call改变数组slice方法里的this指向

现在数组有一个方法slice，这个方法每次都会返回一个新数组，如果不传参的话，返回的新数组的元素和原数组的元素是一模一样的

如果伪元素也能执行这个方法的话，那么是不是就返回一个真正的数组，并且元素一样，但是不能直接执行

所以我们使用偷梁换柱的方法，让一个真正的数组，或者直接从Array.prototype上执行slice方法，但是在执行的时候通过call来将里面的this换成伪数组，这样的话，就会返回一个元素和伪数组元素一样的真正数组了

let arr = [].slice.call(arg) //Array.prototype.slice.call(arg)

**六、Let的特点**

1、let不存在变量提升------>对应例1

2、用let声明过后,不能再次声明----->对应例2

3、3let不存在顶级作用域------>对应例3

4、用if判断或者for()循环中,代表的是父级作用域

5、暂时性死区,在let声明变量之前,该变量是不可用的

例子1:

let a=1

function show(){

console.log(a) //TDZ：暂时性死区

let a=2

//let不存在变量提升

}

例子2:

let a=2;

let a=3;

//会报错,标识符“a”已声明

例子3:

//let不存在顶级作用域,所以this指向window的情况下,不再是顶层对象的属性

例子4:

for(let i=0;i<10;i++){

let i=15

console.log(i)

}

在if判断或者for循环中（）

代表的是父级作用域块{}

**七、实现三栏布局的六种方式**

六种布局方式总结：圣杯布局、双飞翼布局、Flex布局、绝对定位布局、表格布局、网格布局。

1. 绝对定位布局：position + margin
2. 浮动布局： float + margin
3. flex布局
4. table布局
5. Grid网格布局
6. 圣杯布局

是指布局从上到下分为header、container、footer，然后container部分定为三栏布局

1. 双飞翼布局
2. 对比圣杯布局和双飞翼布局：
3. 都是左右栏定宽，中间栏自适应的三栏布局，中间栏都放到文档流前面，保证先行渲染。
4. 解决方案基本相似：都是三栏全部设置左浮动float：left,然后分别结局中间栏内容被覆盖的问题。
5. 解决中间栏内容被覆盖问题时，圣杯布局设置父元素的padding,双飞翼布局在中间栏嵌套一个div,内容放到新的div中，并设置margin，实际上，双飞翼布局就是圣杯布局的改进方案。

**八、解释下js中this是怎么工作的?**

this 在 JavaScript 中主要由以下五种使用场景：

1、作为函数调用，this 绑定全局对象，浏览器环境全局对象为 window 。

2、内部函数内部函数的 this 也绑定全局对象，应该绑定到其外层函数对应的对象上，这是 JavaScript的缺陷，用that替换。

3、作为构造函数使用，this 绑定到新创建的对象。

4、作为对象方法使用，this 绑定到该对象。

5、使用apply或call调用 this 将会被显式设置为函数调用的第一个参数。

**九、Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?**

1）、<!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器以何种模式来渲染文档。

2）、严格模式的排版和 JS 运作模式是 以该浏览器支持的最高标准运行。

3）、在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。

4）、DOCTYPE不存在或格式不正确会导致文档以混杂模式呈现。

**十、为什么要初始化CSS样式。**

因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对CSS初始化往往会出现浏览器之间的页面显示差异。

**十一、JS继承的实现方式**

1、原型链继承

核心： 将父类的实例作为子类的原型

特点：

1. 非常纯粹的继承关系，实例是子类的实例，也是父类的实例
2. 父类新增原型方法/原型属性，子类都能访问到
3. 简单，易于实现

缺点：

1. 要想为子类新增属性和方法，必须要在new Animal()这样的语句之后执行，不能放到构造器中
2. 无法实现多继承
3. 来自原型对象的所有属性被所有实例共享
4. 创建子类实例时，无法向父类构造函数传参

2、构造继承

核心：使用父类的构造函数来增强子类实例，等于是复制父类的实例属性给子类（没用到原型）

特点：

1. 解决了1中，子类实例共享父类引用属性的问题
2. 创建子类实例时，可以向父类传递参数
3. 可以实现多继承（call多个父类对象）

缺点：

1. 实例并不是父类的实例，只是子类的实例
2. 只能继承父类的实例属性和方法，不能继承原型属性/方法
3. 无法实现函数复用，每个子类都有父类实例函数的副本，影响性能

3、实例继承

核心：为父类实例添加新特性，作为子类实例返回

特点：

1. 不限制调用方式，不管是new 子类()还是子类(),返回的对象具有相同的效果

缺点：

1. 实例是父类的实例，不是子类的实例
2. 不支持多继承

4、拷贝继承

特点：

1. 支持多继承

缺点：

1. 效率较低，内存占用高（因为要拷贝父类的属性）
2. 无法获取父类不可枚举的方法（不可枚举方法，不能使用for in 访问到）

5、组合继承

核心：通过调用父类构造，继承父类的属性并保留传参的优点，然后通过将父类实例作为子类原型，实现函数复用

特点：

1. 弥补了方式2的缺陷，可以继承实例属性/方法，也可以继承原型属性/方法
2. 既是子类的实例，也是父类的实例
3. 不存在引用属性共享问题
4. 可传参
5. 函数可复用

缺点：

1. 调用了两次父类构造函数，生成了两份实例（子类实例将子类原型上的那份屏蔽了）

**十二、typeof和instanceof的区别？**

typeof 是一个一元运算，放在一个运算数之前，运算数可以是任意类型。

它返回值是一个字符串，该字符串说明运算数的类型。（typeof 运算符返回一个用来表示表达式的数据类型的字符串）

typeof其实就是判断参数是什么类型的实例，就一个参数

typeof 一般只能返回如下几个结果："number"、"string"、"boolean"、"object"、"function" 和 "undefined"。

instanceof 运算符用来测试一个对象在其原型链中是否存在一个构造函数的 prototype 属性。

参数：object（要检测的对象.）constructor（某个构造函数）

a instanceof b?alert("true"):alert("false"); //a是b的实例？真:假

instanceof 用于判断一个变量是否某个对象的实例，

在 JavaScript 中，判断一个变量的类型尝尝会用 typeof 运算符，在使用 typeof 运算符时采用引用类型存储值会出现一个问题，无论引用的是什么类型的对象，它都返回 “object”。这就需要用到instanceof来检测某个对象是不是另一个对象的实例。

**十三、link和@import的区别？**

1》link属于XHTML标签，而@import是css提供的；

2》页面被加载时，LINK会同时被加载，而@import引入的css会等到页面被加载完后再加载

3》@import只在IE5以上才能识别，而link是XHTML标签，无兼容 问题；

4》link方式的样式的权重，高于@import的权重

**十四、Js添加、创建、复制、查找节点？**

一、创建节点、追加节点

1、createElement（标签名）创建一个元素节点（具体的一个元素）。

2、appendChild（节点）添加一个节点。 document.body.appendChild(创建节点的名字)

3、createTextNode（节点文本内容）创建一个文本节点

二、删除、移除节点

1、removeChild(节点) 删除一个节点，用于移除删除一个参数（节点）。

其返回的被移除的节点，被移除的节点仍在文档中，只是文档中已没有其位置了。

三、替换节点

1、replaceChild(插入的节点，被替换的节点) ，用于替换节点，接受两个参数，

第一参数是要插入的节点，第二个是要被替换的节点。返回的是被替换的节点。

四、查找节点

1、childNodes 包含文本节点和元素节点的子节点。

2、children也可以获取子节点，而且兼容各种浏览器。包括IE6-8

3、parentNode：获取父节点

4、firstChild ; firstElementChild查找第一个子节点。此存在浏览器兼容问题：firstChild是IE兼容，firstElementChild是非IE兼容。

5、lastChild ; lastElementChild查找最后一个子节点。此存在浏览器兼容问题：lastChild 是IE兼容，lastElementChild是非IE兼容。

6、nextSibling ; nextElementSibling查找下一个兄弟节点。也是存在兼容性问题。

7、previousSibling ; previousElementSibling查找上一个兄弟节点。也是存在兼容性问题。

五、克隆节点

cloneNode 克隆节点

**十五、如何阻止默认行为？**

**event.preventDefault()** 阻止默认事件

**event.stopPropagation() 阻止冒泡事件 ||** cancelbubble =true

**return false**这个方法比较暴力，他会同事阻止事件冒泡也会阻止默认事件

**十六、JS对象的几种创建方式**

工厂模式、构造函数模式、原型模式、组合使用构造函数模式和原型模式

**十七、null和undenfined区别？**

undefined是访问一个未初始化的变量时返回的值，

而null是访问一个尚未存在的对象时所返回的值。

因此，可以把undefined看作是空的变量，而null看作是空的对象。

typeof(null) //object

typeof(undefined) // undefined

**十七、事件冒泡**

用户单击了一个元素,该元素拥有一个click事件,那么同样的事件也将会被它的祖先触发,这个事件从该元素开始一直冒泡到DOM树的最上层,这一过程称为事件冒泡

**十八、arguments（不定参）**

arguments对象是所有（非箭头）函数中都可用的局部变量

当我们在js中在调用一个函数的时候，我们经常会给这个函数传递一些参数，js把传入到这个函数的全部参数存储在一个叫做arguments的东西里面，

在js中万物皆对象，甚至数组字符串函数都是对象。所以这个叫做arguments的东西也是个对象，而且是一个特殊的对象，它的属性名是按照传入参数的序列来的，第1个参数的属性名是’0’，第2个参数的属性名是’1’，以此类推，并且它还有length属性，存储的是当前传入函数参数的个数，很多时候我们把这种对象叫做类数组对象。类数组对象和数组都是对象这个妈生的，但是数组是大哥比类数组对象多了很多其他的玩具(方法)，类数组对象只是长得很像数组的弟弟而已

十九、媒体传查询

**二十、”==”和“===”的不同**

前者会自动转换类型,再判断是否相等

后者不会自动类型转换，直接去比较

**二十一、函数声明与函数表达式的区别？**

在Javscript中，解析器在向执行环境中加载数据时，对函数声明和函数表达式并非是一视同仁的，解析器会率先读取函数声明，并使其在执行任何代码之前可用（可以访问），至于函数表达式，则必须等到解析器执行到它所在的代码行，才会真正被解析执行。

**二十二、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 。

**>>>This**

.1. 在一般函数方法中使用 this 指代全局对象

1. 作为对象方法调用，this 指代上级对象
2. 作为构造函数调用，this 指代new 出的对象
3. apply 调用 ，apply方法作用是改变函数的调用对象，此方法的第一个参数为改变后调用这个函数的对象，this指代第一个参数

**二十三、prototype中的属性[**

所有对象的**proto**（对象隐式原型）均指向自身构造函数的prototype（函数显示原型）  
1 constructor 属性指向构造函数本身Object（）

2 hasOwnProperty方法检测对象自身是否包含某属性（传入属性名字符串）  
本方法区别于in运算符， 只能检测对象自有属性（in运算符可以检测到继承属性）

3 isPrototypeOf方法检测对象自身是否处于另一个对象（参数对象）的原型链上  
本方法区别于instanceOf运算符 （instanceOf运算符检测对象是否是一个类的实例）

4 propertyIsEnumerable方法检测对象的某属性是否可以被枚举到（for in），此方  
法 只检测自有属性，继承属性统一返回false。事实上部分预定义继承属性和自定  
义的 继承属性如没有使用属性描述符（ES5）限制其可枚举性，它们都是可以被  
for in 循环枚 举到的。

**二十四、正则表达式量词分别是：贪婪的、惰性的、支配性的。**

贪婪:? \* + {n} {n,m} {n,}

惰性:??  \*?  +?  {n}?  {n,m}?  {n,}?

支配:?+  \*+  ++  {n}+  {n,m}+  {n,}+

分别的意思是:

零次或一次出现、零次或多次出现、一次或多次出现、恰好N次出现、至少N次最多M次出现、至少N次出现。

贪婪量词：先看整个字符串是否匹配，如果不匹配就把最后一个字符去掉在进行匹配，不匹配继续去掉最后一个字符，指导找到一个匹配或者不剩任何字符才停止。

惰性量词：先看第一个字符串是否匹配，如果第一个不匹配就在加入第二个字符串依此类推，指导找到一个匹配或者不剩任何字符才停止，贪婪量词与贪婪量词的方法正好相反.

浏览器对量词的支持还不完善，IE和OPERA都不支持量词，MOZILLA把支配量词看作是贪婪的

二十五、

scrollWidth 为实际内容的宽度。

clientWidth 是内容可视区的宽度。

offsetWidth 是元素的实际宽度

**二十六、b和strong的区别**

盲人朋友使用阅读设备阅读网络时：<strong>会重读，<b>不会

两者虽然在网页中显示效果一样，但实际目的不同。

<b>这个标签对应 bold，即文本加粗，其目的仅仅是为了加粗显示文本，是一种样式／风格需求；

<strong>这个标签意思是加强字符的语气，表示该文本比较重要，提醒读者／终端注意。为了达到这个目的，浏览器等终端将其加粗显示；

总结：<b>为了加粗而加粗，<strong>为了标明重点而加粗，也可以用其它方式来强调，比如下划线，比如字体加大，比如红色，等等，可以通过css来改变strong的具体表现。

**二十七、使用for in 也可以遍历数组，但是会存在以下问题**：

1.index索引为字符串型数字，不能直接进行几何运算

2.遍历顺序有可能不是按照实际数组的内部顺序

3.使用for in会遍历数组所有的可枚举属性，包括原型。例如上栗的原型方法method和name属性

所以for in更适合遍历对象，不要使用for in遍历数组。

**二十八、nul和undefined的区别？**

**Null：空值**

null这个值专门用来表示一个为空的对象

使用typeof检查一个null值时，会返回object

**Undefined：未定义**

当声明一个变量，但是并不给变量赋值时，它的值就是undefined

使用typeof检查一个undefined时也会返回undefined

1、null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；

2、当声明的变量还未被初始化时，变量的默认值为undefined

null用来表示尚未存在的对象，常用来表示函数企图返回一个不存在的对象。

3、undefined表示”缺少值”，就是此处应该有一个值，但是还没有定义

4、

**二十九、webStorage的优势：**

1. 相比cookie存储容量增加
2. 可以将请求的表单数据存于本地，减少http请求，节约带宽
3. webStorage拥有易用的API

webStorage的局限性：

1. 不同浏览器webStorage 和LocalStorage的大小不统一。
2. 在浏览器的隐私模式下面是不可读取的
3. 本质上是对字符串的读取，因此存储内容过多时页面会变卡
4. 不能被爬虫抓取到

**三十、WebStorage**

对浏览器来说，使用 WebStorage 存储键值对比存储 Cookie 方式更直观，它是H5新增的属性，低版本浏览器不兼容，而且容量更大，

它包含两种：localStorage 和 sessionStorage

1. sessionStorage（临时存储） ：为每一个数据源维持一个存储区域，在浏览器打开期间存在，包括页面重新加载
2. localStorage（长期存储） ：与 sessionStorage 一样，但是浏览器关闭后，数据依然会一直存在。可以储存数据的大小是5M。浏览器会把localstorage的内容识别为字符串。

**三十一、GET和POS（**GET/POST都是TCP链接**）**

GET和POST是HTTP协议中的两种发送请求的方法。

* 1. GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。
  2. GET在浏览器回退时是无害的，而POST会再次提交请求
  3. GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息
  4. GET请求只能进行url编码，而POST支持多种编码方式。
  5. GET请求在URL中传送的参数是有长度限制的，而POST么有。
  6. 对参数的数据类型，GET只接受ASCII字符，而POST没有限制。
  7. GET参数通过URL传递，POST放在Request body中。
  8. GET产生一个TCP数据包；POST产生两个TCP数据包。

三十二、编写Ajax

**>**创建Ajax对象

> XMLHttpRequest（）

> ActiveXObject("Microsoft.XMLHTTP")

**>**连接服务器

>Open(请求方法、文件名、异步传输)

>同步和异步

**>**发送请求

>send()

**>**接受返回值

* + - onreadystatechange事件
    - readyState属性：请求状态
    - 0 （未初始化）还没有调用open（）方法
    - 1 （载入）已调用send（）方法，正在发送请求
    - 2 （载入完成）send（）方法完成，已收到全部响应内容
    - 3 （解析）正在解析响应内容
    - 4 （完成）响应内容解析完成，可以在客户端调用
    - Status属性：请求结果
    - responseText：responseText 服务器端返回的json数据就是上述的内容在responseText中

**jsonp：解决跨域**

首先在客户端注册一个callback（名字任意），然后动态创建script标签通过src引入服务器端的php文件（类似引入js文件的方式），同时将客户端注册的callback的名字传给服务器，php文件载入成功后，服务器先生成我们需要的 json 数据，然后将其作为参数传入我们在url参数中指定的函数，所以jsonp是需要服务器端的页面进行相应的配合的

**同步和异步**

同步需要等待返回结果才能继续，异步不必等待，一般需要监听异步的结果  
同步是在一条直线上的队列，异步不在一个队列上 各走各的

**jsonp**

* 定义：一种可以实现跨域发送http请求的数据通信格式，可以嵌在ajax中使用。
* 原理：利用script标签可以跨域链接资源的特性。

**h5新增标签及css3新增属性**

**- h5新增的标签**

Video 表示一段视频并提供播放的用户界面

Audio 表示音频

Canvas 表示位图区域

Svg 定义矢量图

Time 日期和时间值

Keygen 秘钥对生成器控件

Progress 进度条

Menu 菜单

Section nav aside article footer header

**- css3**

背景和边框   
border-radius、box-shadow、border-image、   
background-size：规定背景图片的尺寸   
background-origin：规定背景图片的定位区域   
background-clip：规定背景的绘制区域

文本效果（常用）

text-shadow：设置文字阴影

word-wrap：强制换行

word-break

css3提出@font-face规则，规则中定义了font-family、font-weight、font-style、font-stretch、src、unicode-range

2/3D转换   
transform：向元素应用2/3D转换   
transition：过渡

@keyframes规则：   
animation、animation-name、animation-duration等

css3新增伪类

：nth-child()

：nth-last-child()

：only-child

：last-child

：nth-of-type()

：only-of-type()

：empty

：target 这个伪类允许我们选择基于URL的元素，如果这个元素有一个识别器（比如跟着一个#），那么:target会对使用这个ID识别器的元素增加样式。

：enabled

：disabled

：checked

：not

**ER6新添加的属性**

1、Str.includes（“aa”,6）

在字符串中查找,如果满足，则返回ture,如不满足，返回false

6 表示从字符串中第六位开始查找

2、Str.startsWith（“aa”,6）

在字符串的开头开始查找，如果满足，则返回ture,如不满足，返回false

6 表示从字符串中第六位开始查找

3、Str.endsWith（“aa”,6）

6 查找第六个之前的

4、str.repeat(2)

字符串重复

2 表示重复的次数

5、str.padStart(4,'aa')

补全，如需补全的数字比原字符串小，原字符串不变

否则，在原字符串前补全

6、str.padEnd(4,'aa')

与上一个相同，但它在原字符串后补全

**Let,const,三个点，箭头函数**

**JQ为什么可以链式调用？**

在对象的方法之后 return this，再把对象折返回来

**数组去重？**

1. 最基本的去重方法

思路：定义一个新数组，并存放原数组的第一个元素，然后将元素组一一和新数组的元素对比，若不同则存放在新数组中。

2. 先排序在去重

思路：先将原数组排序，在与相邻的进行比较，如果不同则存入新数组

3 .利用对象的属性去重（推荐）

思路：每次取出原数组的元素，然后在对象中访问这个属性，如果存在就说明重复

4. 利用下标查询

**什么是事件委托？**

事件委托——给父元素绑定事件，用来监听子元素的冒泡事件，并找到是哪个子元素的事件。

**事件委托的好处**：

事件委托技术可以避免对每个子元素添加事件监听器，减少操作DOM节点的次数，从而减少浏览器的重绘和重排，提高代码的性能。

使用事件委托，只有父元素与DOM存在交互，其他的操作都是在JS虚拟内存中完成的，这样就大大提高了性能。

用on方法。。用delegate（）方法。。live()

**HTML和XHTML的区别？**

HTML是一种基本的WEB网页设计语言，XHTML是一个基于XML的置标语言，看起来与HTML有些相象，只有一些小的但重要的[区别](http://www.so.com/s?q=%E5%8C%BA%E5%88%AB&ie=utf-8&src=internal_wenda_recommend_textn)，XHTML就是一个扮演着类似HTML的角色的XML，所以，[本质](http://www.so.com/s?q=%E6%9C%AC%E8%B4%A8&ie=utf-8&src=internal_wenda_recommend_textn)上说，XHTML是一个过渡技术，结合了XML(有几分)的强大功能及HTML(大多数)的简单[特性](http://www.so.com/s?q=%E7%89%B9%E6%80%A7&ie=utf-8&src=internal_wenda_recommend_textn)。

**本地储存localStorage与cookie的区别?**

共同点：都是保存在浏览器端，且都是同源的

区别：

1）、cookie在浏览器和服务器间来回传递

   sessionStroge和localStrige不会自动把数据发送给服务器，仅用于本地保存。

   cookie还有路径（path）的概念，可以限制cookie只属于某个路径下。

2）、存储大小限制不同

   cookie数据不能超过4k

sessionStroge和localStrige虽然也有限制，但可以达到5M以上。

3）、数据有效期不同

   cookie只在设置的cookie过期之前有效，即使窗口或浏览器关闭

   sessionStroge只在当前浏览器关闭之前有效

   localStroge不管浏览器关闭与否，数据永久保存。

4）、作用域不同

   cookie数据在所有同源窗口中都是共享的

sessionStroge在不同浏览器窗口不共享，即使是同一个页面

**要访问同一个localStroge对象页面必须来自同一个域名 （子域名无效）同一协议，同一端口**

**sessionStroge和localStrige无法存入数组或者对象，就算存入了也会被转化为字符串**

**使用 POST 请求的时候？**

* 无法使用缓存文件（更新服务器上的文件或数据库）
* 向服务器发送大量数据（POST 没有数据量限制）
* 发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

**对BFC的理解**？

Formatting context(格式化上下文) 是 W3C CSS2.1 规范中的一个概念。它是页面中的一块渲染区域，并且有一套渲染规则，它决定了其子元素将如何定位，以及和其他元素的关系和相互作用。

**具有 BFC 特性的元素可以看作是隔离了的独立容器，容器里面的元素不会在布局上影响到外面的元素，并且 BFC 具有普通容器所没有的一些特性。**

BFC，块级格式化上下文，一个创建了新的BFC的盒子是独立布局的，盒子里面的子元素的样式不会影响到外面的元素。在同一个BFC中的两个相邻的块级盒子在垂直方向（和布局有关系）的magin会发生折叠

**JSON?**

JSON：js的对象表示法 是轻量级的文本数据交换格式

是存储和交换文本信息的语法，类似XML

JSON比XML更小、更快、更易解析

JSON & XML

可扩展性：差不多

数据体积：JSON体积更小，传输速度更快

数据交互：JSON和JS的交互更加方便，更容易解析处理，更好的数据交互

传输速度：JSON的速度远远快于XML

基于哈希算法在信息安全中主要应用在1、文件校验 2、数字签名 3、鉴权协议

**对象字面量vs构造函数创建对象对比？**

1. 它的代码量更少，更易读；

2. 它可以强调对象就是一个简单的可变的散列表，而不必一定派生自某个类；

3. 对象字面量运行速度更快，因为它们可以在解析的时候被优化：它们不需要"作用域解析(scope resolution)"；因为存在我们创建了一个同名的构造函数Object()的可能，当我们调用Object()的时候，解析器需要顺着作用域链从当前作用域开始查找，如果在当前作用域找到了名为Object()的函数就执行，如果没找到，就继续顺着作用域链往上照，直到找到全局Object()构造函数为止

4. Object()构造函数可以接收参数，通过这个参数可以把对象实例的创建过程委托给另一个内置构造函数，并返回另外一个对象实例，而这往往不是你想要的

**事件监听兼容**

**function myEvent(obj,ev,fn){**

**If(obj . addEventListener){ 兼容IE9以上的浏览器、火狐、谷歌等浏览器**

**obj . addEventListener(ev,fn,false) 默认false 冒泡传递 true 捕获传递**

**}**

**Else{**

**Obj.attachEvent(“on”+ev,fn) 兼容IE8以下的浏览器**

**}**

**}**

**什么是回调函数？**

回调函数就是一个参数，将这个函数作为参数传到另一个函数里面，当那个函数执行完之后，再执行传进去的这个函数。这个过程就叫做回调。

**伪类与伪元素的特性及其区别？**

1. 伪类本质上是为了弥补常规CSS选择器的不足，以便获取到更多信息；
2. 伪元素本质上是创建了一个有内容的虚拟容器；
3. CSS3中伪类和伪元素的语法不同；
4. 可以同时使用多个伪类，而只能同时使用一个伪元素；

**关于使用let与const规则:**

* 使用let声明的变量可以重新赋值,但是不能在同一作用域内重新声明
* 使用const声明的变量必须赋值初始化,但是不能在同一作用域类重新声明也无法重新赋值.

**进程与线程的区别？**

进程和线程的主要差别在于它们是不同的操作系统资源管理方式。进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，**一个线程死掉就等于整个进程死掉**，**所以多进程的程序要比多线程的程序健壮**，但在进程切换时，耗费资源较大，效率要差一些。但对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程。

1) 简而言之,一个程序至少有一个进程,一个进程至少有一个线程.

2) 线程的划分尺度小于进程，使得多线程程序的并发性高。

3) 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。

4) 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。

5) 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。

线程和进程在使用上各有优缺点：线程执行开销小，但不利于资源的管理和保护；而进程正相反。

**面向对象和面向过程的区别？**

面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了。

面向对象是把构成问题事务分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描叙某个事物在整个解决问题的步骤中的行为。

**caller和callee的区别？**

**callee**

callee放回正在执行的函数本身的引用，它是arguments的一个属性

使用callee时要注意:

1 这个属性只有在函数执行时才有效

2 它有一个length属性，可以用来获得形参的个数，因此可以用来比较形参和实参个数是否一致，即比较arguments.length是否等于arguments.callee.length

3 它可以用来递归匿名函数。

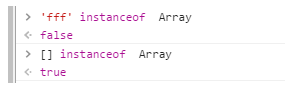
**caller**  
caller返回一个函数的引用，这个函数调用了当前的函数。  
使用这个属性要注意:  
1 这个属性只有当函数在执行时才有用  
2 如果在javascript程序中，函数是由顶层调用的，则返回null

**如何判断是否数组？**

**1、从原型入手，Array.prototype.isPrototypeOf(obj);**

利用isPrototypeOf()方法，判定Array是不是在obj的原型链中，如果是，则返回true,否则false。

**2、也可以从构造函数入手，obj instanceof Array**



**3、根据对象的class属性(类属性)，跨原型链调用toString()方法。**

**4、Array.isArray()方法。**

**[ ]==![ ]结果为true？**

**[]==[] //false**

因为在两边都是相同数据类型时，==是直接比较两边数据，结果为false，因为[]属于引用类型，在两个[ ]分别指向不同的堆内存

**[]==![]//true**

！的优先级要大于==的，所以先运算右边，![ ]---->结果为false，

这样[ ]==![ ]就等价于于：[ ]==false

再来看，这时候==两边数据类型就不相等辣，我们知道，当==两边数据类型不一致时，要先做数据类型转换，再判断，

== 类型转换规则如下:

值类型 == 值类型 //相当于 Number(值类型) == Number(值类型)

引用类型 == 值类型  // 对象转化成原始类型的值，再进行比较

**比较规则：数组与数值进行比较，会先转成数值，再进行比较；与字符串进行比较，会先转成字符串，再进行比较；与布尔值进行比较，两个运算子都会先转成数值，然后再进行比较。**



**拖拽元素原理？**

拖拽功能主要是用在让用户做一些自定义的动作，比如拖动排序，弹出框拖动移动等等

三大事件：onmousedown（选择元素） -> onmousemove（移动元素） -> onmouseup（释放元素）

拖拽其实是通过获取鼠标移动的距离来实现的，即计算移动前的位置的坐标（x，y）与移动中的位置的坐标（x，y）差值。  
当鼠标按下或鼠标移动时，都可以获取到当前鼠标的位置，即移动前的位置与移动中的位置。

被拖拽的元素的样式要设置成绝对或相对位置才有效果。

**JQ绑定与解绑？**

bind() //是直接绑定在元素上

live() // 优势是支持动态数据,像ajax或pjax和通过append出来的元素

delegate() // 则是更精确的小范围使用事件代理，性能优于live()

on() //是最新的1.7版本整合了之前的三种方式的新事件绑定机制

one() //每个元素只能运行一次事件处理器函数。自动解绑

unbind() //解除通过bind绑定的事件

die() //解除通过live绑定的事件

undelegate() // 解除通过delegate绑定的事件

off() //解除通过on绑定的事件

自 jQuery 版本 1.7 起，on()和 off()方法是添加和移除元素上的事件处理程序的首选方法。

**JS绑定及解绑？**

* + 1. 普通绑定

Function fn1(){aleat (1)}

解绑：document.onclick = null // 取消事件绑定

* + 1. addEventListener

解绑：. removeEventListener ("click", f1, false)

* + 1. attachEvent

解绑：detachEvent("onclick", f1)

#### 面向对象的三大特性：

1、封装  
隐藏对象的属性和实现细节，仅对外提供公共访问方式，将变化隔离，便于使用，提高复用性和安全性。  
2、继承  
提高代码复用性；继承是多态的前提。  
3、多态  
父类或接口定义的引用变量可以指向子类或具体实现类的实例对象。提高了程序的拓展性。

1、抽象会使复杂的问题更加简单化。  
2、从以前面向过程的执行者，变成了张张嘴的指挥者。  
3、面向对象更符合人类的思维，面向过程则是机器的思想

**HTTP?**

HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议

HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。

HTTP协议工作于客户端-服务端架构为上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。Web服务器根据接收到的请求后，向客户端发送响应信息。

**HTTP工作原理**

1、客户端连接到Web服务器

一个HTTP客户端，通常是浏览器，与Web服务器的HTTP端口（默认为80）建立一个TCP套接字连接。例如，http://www.oakcms.cn。

2、发送HTTP请求

通过TCP套接字，客户端向Web服务器发送一个文本的请求报文，一个请求报文由请求行、请求头部、空行和请求数据4部分组成。

3、服务器接受请求并返回HTTP响应

Web服务器解析请求，定位请求资源。服务器将资源复本写到TCP套接字，由客户端读取。一个响应由状态行、响应头部、空行和响应数据4部分组成。

4、释放连接TCP连接

若connection 模式为close，则服务器主动关闭TCP连接，客户端被动关闭连接，释放TCP连接;若connection 模式为keepalive，则该连接会保持一段时间，在该时间内可以继续接收请求;

5、客户端浏览器解析HTML内容

客户端浏览器首先解析状态行，查看表明请求是否成功的状态代码。然后解析每一个响应头，响应头告知以下为若干字节的HTML文档和文档的字符集。客户端浏览器读取响应数据HTML，根据HTML的语法对其进行格式化，并在浏览器窗口中显示。

例如：在浏览器地址栏键入URL，按下回车之后会经历以下流程：

1、浏览器向 DNS 服务器请求解析该 URL 中的域名所对应的 IP 地址;

2、解析出 IP 地址后，根据该 IP 地址和默认端口 80，和服务器建立TCP连接;

3、浏览器发出读取文件(URL 中域名后面部分对应的文件)的HTTP 请求，该请求报文作为 TCP 三次握手的第三个报文的数据发送给服务器;

4、服务器对浏览器请求作出响应，并把对应的 html 文本发送给浏览器;

5、释放 TCP连接;

6、浏览器将该 html 文本并显示内容;

**ie6以下margin 0 auto的解决办法**

1.在块外边添加div,并对此div应用 text-align:center

2

