# JavaScript学习笔记

## 1、JavaScript的原型和原型链：

（1）、原型对象的存储：但是，一旦定义了原型属性或原型方法，则所有通过该构造函数实例化出来的所有对象，都继承了这些原型属性和原型方法，这是通过内部的\_proto\_链来实现的。

　譬如：

　A.prototype.say=function(){alert("Hi")};

　那所有的A的对象都具有了say方法，这个原型对象的say方法是唯一的副本给大家共享的，而不是每一个对象都有关于say方法的一个副本。

（2）、原型对象的继承：

## 2、==和===的区别？

”==”与”===”是不同的,一个是判断值是否相等,一个是判断值及类型是否完全相等。  
下面的规则用于判定===运算符比较的两个值是否相等的判断条件  
•如果两个值的类型不同，它们就不相同。  
•如果两个值是数字，而且值相同，那么除非其中一个或两个都是NaN(这种情况它们不是等同的),否则它们是等同的。值NaN永远不会与其他任何值等同，包括它自身(奇怪的家伙),要检测一个值是否是NaN，可以使用全局函数isNaN()。  
•如果两个值都是字符串，而且在串中同一位置上的字符完全相同，那么它们就完全等同。如果字符串的长度或内容不同，它们就不是等同的。  
•如果两个值都是布尔型true，或者两个值都是布尔型false，那么它们等同。  
•如果两个值引用的是同一个对象、数组或函数，那么它们完全等同。如果它们引用的是不同的对象（数组或函数），它们就不完全等同，即使这两个对象具有完全相同的属性，或两个数组具有完全相同的元素。  
•如果两个值都是null或都是undefined，它们完全相同。  
  
下面的规则用于判定==运算符比较的两个值是否相等的判断条件  
•如果两个值具有相同的类型，那么就检测它们的等同性。如果这两个值完全相同，它们就相等。如果它们不完全相同，则它们不相等。  
•如果两个值的类型不同，它们仍然可能相等。用下面的规则和类型转换来检测它们的相等性 ◦如果一个值是null，另一个值是undefined，它们相等。  
◦如果一个值是数字，另一个值是字符串，把字符串转换为数字，再用转换后的值进行比较。  
◦如果一个值为true，将它转化为1，再进行比较。如果一个值为false，把它转化为0，再进行比较。  
◦如果一个值是对象，另一个值是数字或字符串，将对象转换成原始类型的值，再埋比较。可以使用对象的toString()方法或valueOf()方法把对象转化成原始类型的值。JavaScript核心语言的内部类通常先尝试valueOf()方法转换，再尝试toString()方法转换，但是对于Date类，则先执行toString()方法再执行valueOf()方法转换。不属于JavaScript核心语言的对象则可以采用JavaScript实现定义的方式把自身转换成原始数值。  
◦其他的数值组合是不相等的。

# 2.1、undefined与null的区别

undefined和null在if语句中，都会被自动转为false，相等运算符甚至直接报告两者相等。

其次，JavaScript的最初版本没有包括错误处理机制，发生数据类型不匹配时，往往是自动转换类型或者默默地失败。Brendan Eich觉得，如果null自动转为0，很不容易发现错误。

JavaScript的最初版本是这样区分的：**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。**

现在的用法：

**null表示"没有对象"，即该处不应该有值。**典型用法是：

（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。

Object.getPrototypeOf(Object.prototype)

// null

**undefined表示"缺少值"，就是此处应该有一个值，但是还没有定义。**典型用法是：

（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。

## 3、XHTML 规则概要

将传统的 HTML 转换为 XHTML 1.0 是快捷且无痛的，只要你遵守一些简单的规则和容易的方针。不管是否使用过 HTML，都不会妨碍你使用 XHTML。

* 使用恰当的文档类型声明和命名空间。
* 使用 meta 元素声明你的内容类型。
* 使用小写字母书写所有的元素和属性。
* 为所有的属性值加引号。
* 为所有的属性分配值。
* 关闭所有的标签。
* 使用空格和斜线关闭空标签。
* 不要在注释中写双下划线。
* 确保小于号及和号为 < 和 &

5、jquery 的on()方法绑定事件。

tip:如果你需要移除on()所绑定的方法，可以使用off()方法处理。

$(document).ready(function(){

　　$("p").on("click",function(){

　　　　$(this).css("background-color","pink");

　　});

　　$("button").click(function(){

　　　　$("p").off("click");

　　});

});

6、**async 与 defer 的差别 ——js文件的无阻塞加载。**  
[WebKit官方博客](http://webkit.org/blog/1395/running-scripts-in-webkit/" \t "_blank) 很好地解释了async 与 defer 的不同

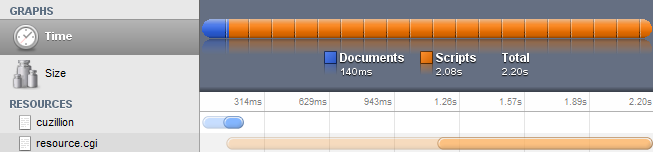
------------------------------------

正常情况下，当浏览器在解析HTML源文件时如果遇到外部的script，那么解析过程会暂停，并发送请求来下载script文件，只有script完全下载并执行后才会继续执行DOM解析。比如：  
<script src="myBlockingScript.js"></script>  
在下载过程中浏览器是被阻止做其他有用的工作的，包括 解析HTML，执行其他脚本，以及展示CSS布局。虽然Webkit预加载扫描程序可以探测性地在下载阶段进行多线程下载，但是某些页面仍然存在很大的网络延迟。  
当前有很多技术来提升页面显示速度,但都需要额外的代码以及针对特定浏览器的技巧。现在，script可以通过添加async或者defer属性来让脚本不必同步执行，示例如下：

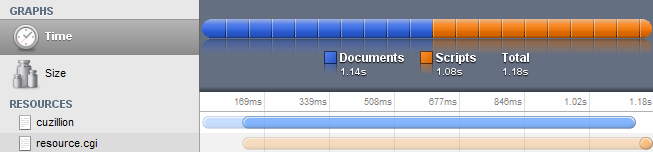
**[html]** [view plain](http://blog.csdn.net/renfufei/article/details/10210949" \o "view plain)[copy](http://blog.csdn.net/renfufei/article/details/10210949" \o "copy)[在CODE上查看代码片](https://code.csdn.net/snippets/326013)

1. <script async src="myAsyncScript.js" onload="myInit()"></script>
2. <script defer src="myDeferScript.js" onload="myInit()"></script>

async 和 defer 标注的 script 都不会暂停HTML解析就立刻被下载,两者都支持onload事件回调来解决需要该脚本来执行的初始化。  
**两者的区别在于执行时的不同**：  
async 脚本在script文件下载完成后会立即执行,并且其执行时间一定在 window的load事件触发之前。这意味着多个async脚本很可能不会按其在页面中的出现次序顺序执行。  
与此相对，浏览器确保多个 defer 脚本按其在HTML页面中的出现顺序依次执行,且执行时机为DOM解析完成后，document的DOMContentLoaded 事件触发之前。

下面展示的是一个需要1秒来下载，以及1秒来解析执行其他操作的例子，我们可以看到整个页面载入花了大约2秒钟。  


同样的例子，但这次我们指定了script的 defer 属性.因为当defer脚本下载的时候，其他操作可以并行执行，所以大概快了1倍。



------------------------------------

**哪些浏览器支持async 和 defer**  
同样是上面引用的文章中提到：

**[plain]** [view plain](http://blog.csdn.net/renfufei/article/details/10210949" \o "view plain)[copy](http://blog.csdn.net/renfufei/article/details/10210949" \o "copy)[在CODE上查看代码片](https://code.csdn.net/snippets/326013)

1. 除了基于Webkit的新版本浏览器,FireFox已经支持defer和onload属性很长时间了，而且从FF3.6开始添加了async属性。IE同样支持defer属性,但还不支持async属性，从IE9开始，onload属性也将被支持。

**aynsc 棒极了！**  
看到webkit实现async我开心得合不拢嘴了。对每个网站来说，阻塞都是一个巨大的性能瓶颈，而可以直接指定script文件异步加载无疑会加快web页面的速度.

## 7、全部变量和局部变量：

大家都知道，变量是需要用var关键字声明的（对函数也一样）。但是javascript中也可以隐式的使用变量，就是不用声明，直接使用。而且，千万注意，javascript把隐式声明的变量总是当成全局变量来使用的。

Var a = 10; // 局部变量。

b=2; //全局变量。

c=a+b; // c也是全局变量。

函数也一样：

testVar = function (){

console.log("这是一个全局函数。");

}

var testVar = function (){

console.log("这是一个局部函数。");

}

Function testVar (){

console.log("这是一个局部函数。");

}

**8、推荐使用正常的for循环来处理数组，使用for-in循环来处理对象。**

并且使用for-in循环时最好加一个过滤器来过滤掉对象的原型属性。如下:

For(var i in man){

If(man.hasOwnProperty(i)){ //filter

//todo

}

}

# 9、JavaScript中的分号插入机制详细介绍

**一般JavaScript代码在解析时会自动在换行处（就是每行的结尾）加一个分号（;）。但是也有时是不会自动加上一个分号的。就是以某些符号结束或开始一行时是会禁用自动插入分号的。**

**仅在}之前、一个或多个换行之后和程序输入的结尾被插入**

也就是说你只能在一行、一个代码块和一段程序结束的地方省略分号。

所以你必须得注意下一条语句的开始，从而确定你是否能够合法的省略分号。一般在（、[、+、-、和/  这五个字符开始的语句，那么最好前面不要省略分号。

所以如果你需要省略分号，必须检查接下来的一行开始标记是否为上述五个字符导致解析器会禁用自动插入分号，或者你也可以在（、[、+、-、和/  这五个字符前置一个分号

**总结一下哦**

1.仅在}标记之前、一行的结束和程序的结束处推导分号  
2.仅在紧接着的标记不能被解析的时候推导分号  
3.在以（、[、+、-、和/ 字符开头的语句前绝不能省略分号  
4.当脚本连接的时候，在脚本之间显式的插入分号  
5.在return、throw、break、continue、++或--的参数之前绝不能换行  
6.分号不能作为for循环的头部或空语句的分隔符而被推导出

10、javascript中函数的静态属性和静态函数。一个诡异的问题。

~~Javascript中函数的静态属性（一定要是存储字符串类型的静态属性）不能和函数内定义的属性同名，否则以静态属性的形式是访问不到这个属性的字符串值得。~~

~~控制台的输入输出如下：~~

~~Var parent = function ()~~*~~{name: "aoaooa", getname: function(){return this. name;}, age: 24;~~* ~~getage~~*~~:function(){return this. age };};~~*

~~undefined~~

~~parent.age=34;~~

~~34~~

~~parent.age~~

~~34~~

~~parent.name = "yyyyy";~~

~~"yyyyy"~~

~~parent.name~~

~~""~~

~~如上看出，其他的静态属性（包括字面量对象变量，函数方法）都可以和parent函数内定义的属性方法同名，也是可以访问到的（当然访问的是静态属性）。~~

~~如果再定义一个静态的getname和getage方法，则控制台输出也是name=””,age=34;~~

~~总之就是静态的字符串变量不能和函数内定义的字符串变量同名，不然静态的那个变量会总是空的。。~~

上面一段都是错的，，因为在js的函数中name默认内置是表示函数名的，而且是不能改变的。通过var parent = function (){ } 的方式定义的函数，函数名是空的。这是匿名函数形式定义的，所以最好是var parent = function parent(){ }方式定义

一、私有变量：就是直接在构造函数定义里用var tname=”yaoao”; 这种方式定义的变量，私有函数也是一样。要想在外部访问到私有变量，可以通过共有方法：（1）、直接在构造函数里定义的共有方法可以访问到私有变量。this.getName(){ } 。 （2）、通过实例对象后来添加上去的共有方法是访问不到私有变量的。

可以看到getname()是可以访问到私有变量ttname的。而通过实例对象\_p.getage = function (){} 添加的共有方法是访问不到私有变量age的。打印出\_p实例对象后，可以看到，构造函数中定义的所有变量和函数都是在原型对象中的。

而以后通过实例对象添加的共有方法是不在原型中的，所以访问不到原型中的私有变量。



二、共有变量和共有函数： 就是通过this，在构造函数中用this.name=”liuyaoao”;这种方式定义的变量。

三、静态变量：就是用函数名 parent.tname=”aoaoao”;定义的变量。也只能用函数名去访问。也可以通过实例对象：

\_p.constructor.tname访问到。

10、原生js的使用知识点：

在 JavaScript 中，document.write() 可用于直接向 HTML 输出流写内容。绝对不要在文档加载完成之后使用 document.write()。这会覆盖该文档。

## 改变 HTML 内容

修改 HTML 内容的最简单的方法时使用 innerHTML 属性。

如需改变 HTML 元素的内容，请使用这个语法：

document.getElementById(*id*).innerHTML=*新的 HTML*

## 改变 HTML 属性

如需改变 HTML 元素的属性，请使用这个语法：

document.getElementById(*id*).属性名*=新属性值*

## 改变 HTML 样式

如需改变 HTML 元素的样式，请使用这个语法：

document.getElementById(*id*).style.*property*=*新样式*

示例：

document.getElementById("p2").style.color="blue";

document.getElementById("p2").style.fontFamily="Arial";

document.getElementById("p2").style.fontSize="larger";

## 创建新的 HTML 元素

var para=document.createElement("p");  
var node=document.createTextNode("这是一个新段落。");  
para.appendChild(node);  
  
var element=document.getElementById("div1");  
element.appendChild(para);

|  |  |
| --- | --- |
| [document.querySelector()](http://www.runoob.com/jsref/met-document-queryselector.html) | 返回文档中匹配指定的CSS选择器的第一元素 |
| document.querySelectorAll() | document.querySelectorAll() 是 HTML5中引入的新方法，返回文档中匹配的CSS选择器的所有元素节点列表 |

var getWeekDayByDate=function(year,month,day){

var w=0,y=0,c=0,m=0,d=0;

if(month<3){

m = month + 12;

year = year-1;

}else{

m = month;

}

y = year%100;

c = year/100;

d = day;

w=y+(y/4)+(c/4)-2\*c+(26\*(m+1)/10)+d-1;

return w%7;

}