1：可以将JavaScript代码放在html文件中任何位置，但是我们一般放在网页的head或者body部分。

2：Js代码必须写在<script type="text/javascript"> </script>之间 或者用<scriptsrc="script.js"></script>引入js文件。

3： 变量和变量赋值var mychar;

mychar="javascript";或者var mynum = 8;

函数定义：

function函数名()

{

函数代码;

}

函数定义，调用，以及html的调用。可以直接在html里调用。



function a()

{

函数代码;

return ?;(返回函数值)

}

var b = a();(将函数返回值保存在b中);

4：向网页输出语句: document.write("I");

Js输出空格问题：解决方法：1使用html标签&nbsp来解决：document.write("&nbsp;&nbsp;"+"1"+"&nbsp;&nbsp;&nbsp;&nbsp;"+"23"); 2使用css式样来解决“white-space:pre;”，该式样标识”空白会被浏览器保留”：document.write("<span style='white-space:pre;'>"+"  1     2   3 "+"</span>");:

5：js警告：alert(字符串或变量); var a=alert(“bbbbb”);点击确定则a=“true”。

6：js确认：confirm(str); 弹出含str的对话框，点确定返回true，点取消返回false；

<script type="text/javascript">

var mymessage=confirm("你喜欢JavaScript吗?");

if(mymessage==true)

{ document.write("很好,加油!"); }

else

{ document.write("JS功能强大，要学习噢!"); }

</script>

6：js提问：**prompt(str1, str2)，**弹出消息对话框,通常用于询问一些需要与用户交互的信息**;** str1: 要显示在消息对话框中的文本，不可修改；str2：文本框中的内容，可以修改； 点击确定按钮，文本框中的内容将作为函数返回值，当点击取消按钮，将返回**null；**

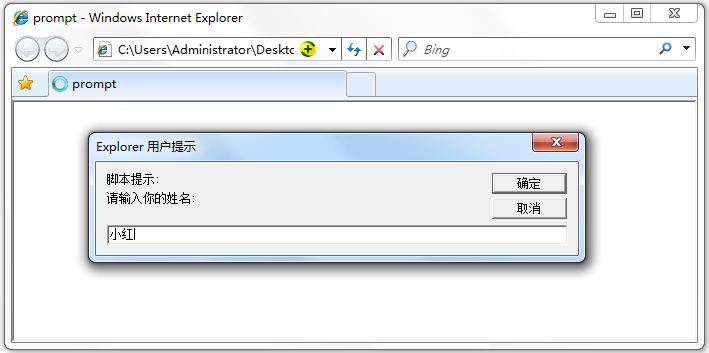
var myname=prompt("请输入你的姓名:");

if(myname!=null)

{ alert("你好"+myname); }

else

{ alert("你好 my friend."); }



7：js打开新窗户：window.open([URL], [窗口名称], [参数字符串])；**URL：**可选参数，在窗口中要显示网页的网址或路径。如果省略这个参数，或者它的值是空字符串，那么窗口就不显示任何文档。

**窗口名称：**可选参数，被打开窗口的名称。

1.该名称由字母、数字和下划线字符组成。

2."\_top"、"\_blank"、"\_self"具有特殊意义的名称。

  \_blank：在新窗口显示目标网页

  \_self：在当前窗口显示目标网页

  \_top：框架网页中在上部窗口中显示目标网页

3.相同 name 的窗口只能创建一个，要想创建多个窗口则 name 不能相同。

  4.name 不能包含有空格。

**参数字符串：**可选参数，设置窗口参数，各参数用逗号隔开。



<script type="text/javascript"> window.open('http://www.imooc.com','\_blank','width=300,height=200,menubar=no,toolbar=no, status=no,scrollbars=yes')；

</script>

8：js关闭窗口：window.close(); //关闭本窗口

或者<窗口对象>.close(); //关闭指定的窗口

<script type="text/javascript">

var mywin=window.open('http://www.imooc.com'); //将新打的窗口对象，存储在变量mywin中

mywin.close();

</script> 打开新窗口时就关闭了，故看不到新窗口。

9：DOM：文档对象模型DOM定义访问和处理HTML文档的标准方法。DOM 将HTML文档呈现为带有元素、属性和文本的树结构（节点树）。

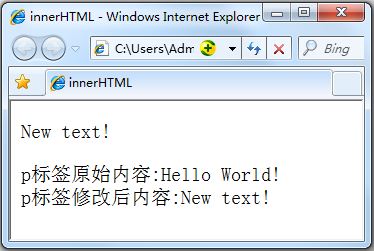
将HTML代码分解为DOM节点层次图:



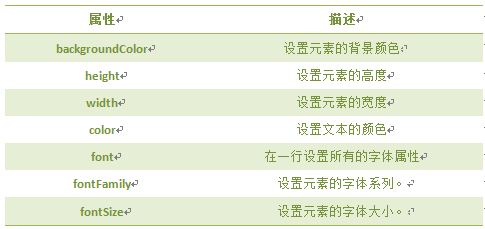
10：通过id获取元素：document.getElementById(“id”)

11：innerHTML 属性：用于获取或替换 HTML 元素的内容：**Object.innerHTML：**Object是获取的元素对象，如通过document.getElementById("ID")获取的元素。



结果

12：改变HTML样式：HTML DOM 允许 JavaScript 改变 HTML 元素的样式：Object.style.property=new style; Object是获取的元素对象，如通过document.getElementById("id")获取的元素。



<p id="pcon">Hello World!</p>

<script>

var mychar = document.getElementById("pcon");

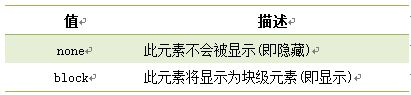
mychar.style.color="red";

mychar.style.fontSize="20";

mychar.style.backgroundColor ="blue";

</script>

13：display属性：显示和隐藏：Object.style.display = value；Object依旧为取到的对象；





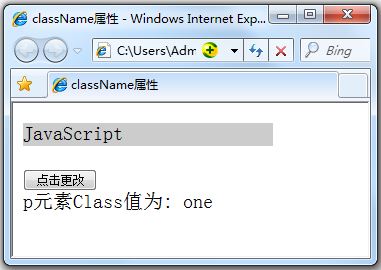
14：className 属性：设置或返回元素的class 属性：object.className = classname；

作用:

1.获取元素的class 属性

2. 为网页内的某个元素指定一个css样式来更改该元素的外观



点击“点击修改”就会修改显示的式样。

15：创建数组：var myarray= new Array(8);如果数组为空则输出：undefined。

二维数组 myarr[0][1] ,0 表示表的行，1表示表的列。定义：var arr=[[1,2,3],[4,5,6]];

16：事件：JavaScript 创建动态页面。事件是可以被 JavaScript 侦测到的行为



17:鼠标单击（onclick）常与按钮一起使用。

18：鼠标经过事件(onmouseover)：<input name="password" type="password" onmouseover="message()">当鼠标经过password时弹出消息框并调用message函数。

19：鼠标移开事件（onmouseout）；当鼠标移开时弹出消息框并调用message函数。

20：光标聚焦事件（onfocus）：<input name="password" type="password" onfocus ="message()">：当光标移至文本框时，即焦点在文本框内触发该事件。

21：失焦事件（onblur）：当光标移开当前聚焦对象时触发；

22：内容选中事件（onselect）：当文本框或文本域中的文字被选中时触发,同时调用message：<textarea name="summary" cols="60" rows="5" onselect =”message”>请写入个人简介</textarea>

23：文本框内容改变事件（onchange）: 通过改变文本框的内容来触发onchange事件，同时执行被调用的程序。

24：加载事件（onload）：事件会在页面加载完成后，立即发生，同时执行被调用的程序。  
注意：1. 加载页面时，触发onload事件，事件写在<body>标签内。

<body onload="message()">

欢迎学习JavaScript。

</body>

25：卸载事件（onUnload）：当用户退出页面时（页面关闭、页面刷新等），触发onUnload事件，同时执行被调用的程序。注意：不同浏览器对onunload事件支持不同。

<script type="text/javascript">

window.onunload = onunload\_message;

function onunload\_message(){

alert("您确定离开该网页吗？");

}

</script>

26： Date对象中处理时间和日期的常用方法：



自定义初始值：var d = new Date(2012, 10, 1);或var d = new Date('Oct 1, 2012');

getDay返回星期，0到6；0为星期日。

27：charAt（）方法返回指定位置的字符，如果越界则返回空字符串。

28：indexOf（）方法返回某个指定的字符串在字符串中首次出现的位置：stringObject.indexOf(substring, startpos)；substring为指定的字符串，startpos开始的位置，如果没有则从头开始。如果要检索的字符串没有出现则返回-1。

29：window对象：

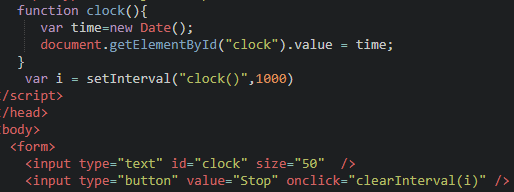


30：js计时器：

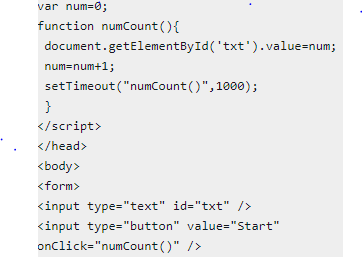


31：setInterval(代码,交互时间); 在执行时,从载入页面后每隔指定的时间执行代码。毫秒级

setInterval("clock()",1000)或setInterval(clock,1000)；

32: clearInterval() 方法可取消由 setInterval() 设置的交互时间。clearInterval(id),id为setInterval（）返回的id值。代码：

33：setTimeout()计时器，在载入后延迟指定时间后,去执行一次表达式,仅执行一次。setTimeout(代码,延迟时间);毫秒为单位。

当按钮被点击后，输入域便从0开始计数。

34：clearTimeout()和setTimeout()一起使用，停止计时器。clearTimeout(id)，id：由 setTimeout() 返回的 ID 值。该值标识要取消的延迟执行代码块。



点击Start按钮后，开始计数, 点击Stop按钮后，停止计数.

35: history对象记录了用户曾经浏览过的页面(URL)，并可以实现浏览器前进与后退相似导航的功能。注意:从窗口被打开的那一刻开始记录，每个浏览器窗口、每个标签页乃至每个框架，都有自己的history对象与特定的window对象关联。window.history.[属性|方法],

Window可以省略。属性：length，返回历史列表中的URL数量。

方法：back（）加载history列表中的上一个URL，语法window.history.back()，相当于go(-1);

forward（）加载history列表中的下一个URL, 相当于go(1)；go（）加载history列表中的某个具体的页面。

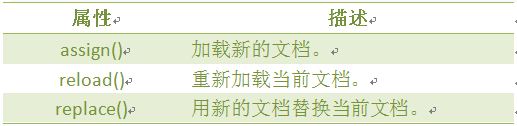
36: location用于获取或设置窗体的URL，并且可以用于解析URL。语法：location.[属性|方法]，location对象属性图示:



对象属性：

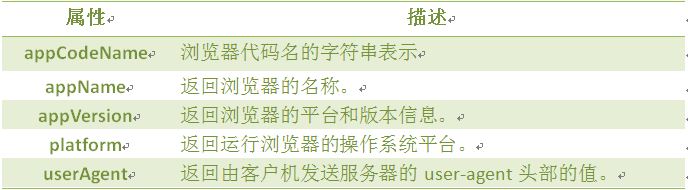


对象方法：

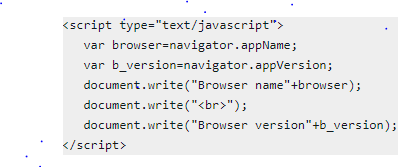


显示当前文档URL：document.write(window.location.href);

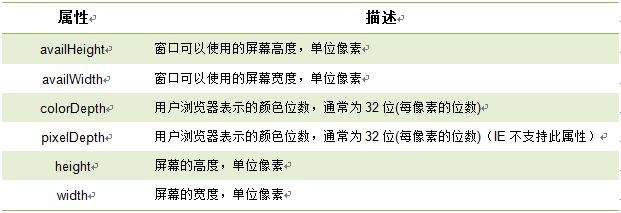
**3**7：Navigator 对象包含有关浏览器的信息，通常用于检测浏览器与操作系统的版本。



查看浏览器的名称和版本：



screen对象用于获取用户的屏幕信息，语法：window.screen.属性

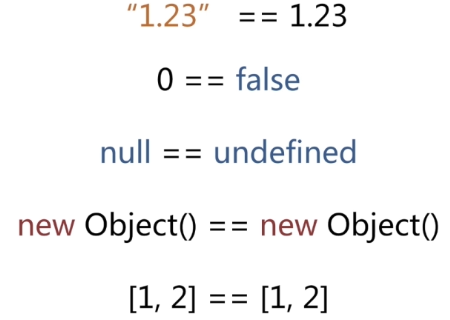


基本语法：

1：Var x = ‘37’ + 7 //377

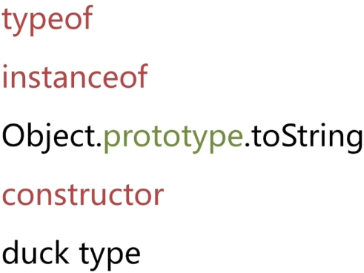
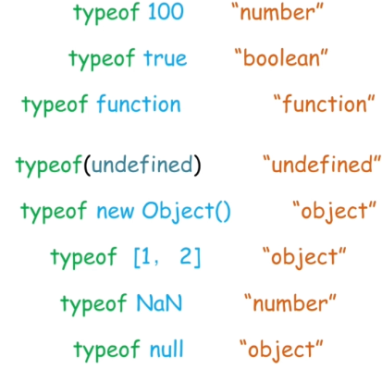
Var x = ‘37’ - 7 //30

所以字符串和数字间的类型转换：num-0转换为数字，num+‘’转换为字符串。a =+a；这句的目的是为了将a转换为数字。

2：等于：当类型不同时会尝试类型装换在比较；字符串和数字比较是会尝试吧字符串转化为数字进行比较。

严格等于===，先进行类型比较。NaN和任何比较都不相等，包括自己。对象的比较和java类似

3：基本类型具有它对应的包装类型的属性，当调用完属性后，基本类型创建的包装类型就销毁了。故var a = “string”；a.t = 3; //创建属性t赋值；a.t;//取值，此时为undefined，因为a的包装对象已经销毁了。

4：类型判断 ”

Instanceof用于对象判断，typeof多用于原始类型或者函数对象

5：函数：函数和变量本质上是一样的，可以把一个函数当作参数传入到另一个函数中：

square = function (a) {

return a \* a;

}

applyOperation = function (f, a) {

return f(a);

}

applyOperation (square, 10); // 返回100

可以再函数声明之前调用函数，叫做函数前置；但是函数表达式不行。

函数声明function a(){};函数表达式var a = function (){};

函数如果没有return就会在所有代码执行完后返回一个undefined；

6：js和java一样，支持三元运算符foo ? function1() : function2();

7：js对象：

myProfile = {

name: "Jare Guo",

email: "blabla@gmail.com",

'zip code': 12345,

isInvited: true

}

有一组用逗号相隔的键值对。每一对都包括一个 key（字符串类型，有时候会用双引号包裹）和一个 value（可以是任何类型：包括 string，number，boolean，变量名，数组，对象甚至是函数）。我们管这样的一对键值叫做对象的属性（property），key 是属性名，value 是属性值。可以在 value 中嵌套其他对象，或者由一组对象组成的数组：

myProfile = {

name: "Jare Guo",

email: "blabla@gmail.com",

city: "Xiamen",

points: 1234,

isInvited: true,

friends: [

{

name: "Johnny",

email: "blablabla@gmail.com"

},

{

name: "Nantas",

email: "piapiapia@gmail.com"

}

]

}

对象访问：

myProfile.name; // Jare Guo

myProfile.friends[1].name; // Nantas

对象{x:1，y:2}相当于var o = new Object(); o.x=1,o.y=2;

属性无序。

8：匿名函数：在参数位置直接写一个函数体。

9：链式语法：链条中的每个环节都会接到一个初始值，调用一个函数，然后把函数执行结果传递到下一环节：

var n = 5;

n.double().square(); //100

10：代码风格：使用驼峰命名法；在每一行结束时写一个; 在每个关键字前后都加上空格。

11：逗号运算符：var a = (1,2,3);//a = 3; 会将每个都运算一遍取最右边的。

12：delete obj.x 删除对象obj上的x属性。删除不存在的属性时也会返回true；

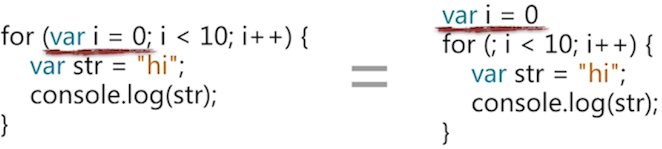
var定义的全局变量和定义的全局函数是无法被delete的，但是没有var的隐式定义的变量是可以的。

13：‘x’ in obj；//true或false 用in来判断obj对象中是否有x属性。

14：void 0或者void（0） 无论void后面是多少结果都是undefined

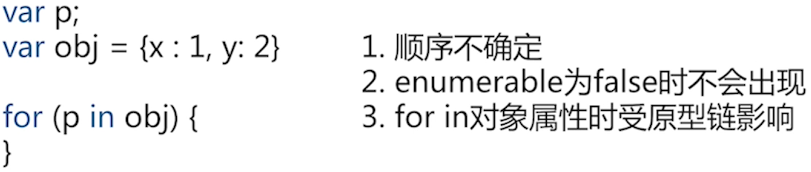
15：block块语句：常用于组合多个语句{语句1；语句2；…},花括号开头则被认为是块。

{a:1,b:2}报错，var a = { a:1,b:2}正确。 没有块级的作用域：



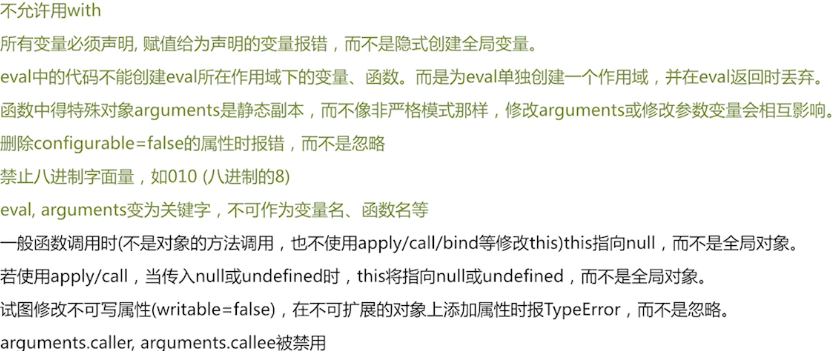
16：var a=b=1,这时，b将会是一个全局变量。

17：for…in语句：



18：with语句：修改当前的作用域，不建议使用。用定义变量来替代

19：严格模式：修补部分语言上的不足，提供更强的错误检查，增强安全性，语句或者文件开头加‘use strict’进入。



20：原型链：每个对象都有一个prototype的对象属性。New出来一个对象，这个对象就会指向它的prototype对象，它的prototype对象就会指向Object对象的prototype对象，这就是为什么对象没有声明toString属性却有toString属性。原型链的继承来实现属性的继承。 var obj = Object.create(mull);这样定义的对象没有原型链。

21：get和set：当在原型链中有set和get方法时，当对对象属性（它的原型中同属性名有get和set方法）赋值是不成功的，只能用Object.defineProperty(obj,’z’{value:100})这种方式来为对象obj添加属性z并赋值100 。

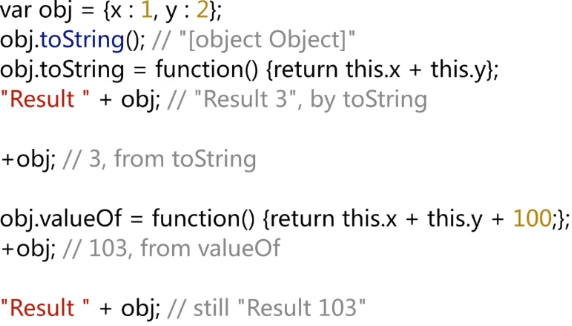
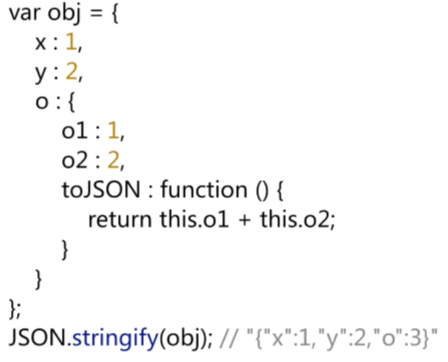
22：属性标签：enumerable（是否可枚举）,writable（是否可修改）,configurable（是否可删除）.

23：对象标签：原型标签：[[proto]];[[class]];对象是否可扩展（是否可添加）[[extensible]]：一般默认为true，调用Object.preventExtensions（obj）；则将obj对象的标签改为false。

Object.seal(obj);将obj对象上的全部属性的congigurable标签改为false；Objec.isSealed来判断是否seal了，通过Object.freeze(obj);来将全部属性的wriable和configurable改为false；不会修改原型链。

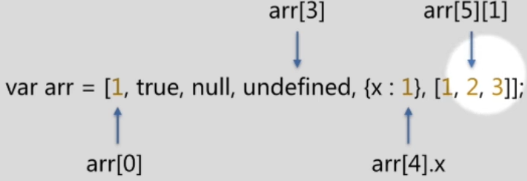
24：序列化：JSON.stingify(obj);当对象的属性值为undefined时就不会出现在序列化中。

反序列化：obj =JSON.parse(‘{“x”:1}’);将序列化的转化为对象。 对象的序列化是可以自定义的：



25：其他对象方法

26：数组，是弱类型的，数组中可以含有不同类型的元素，甚至是对象或者其他数组。

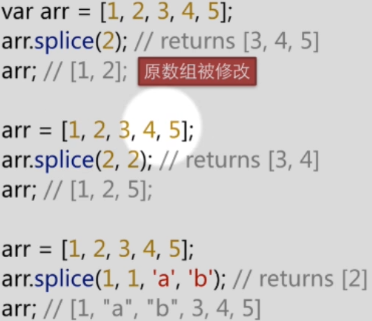
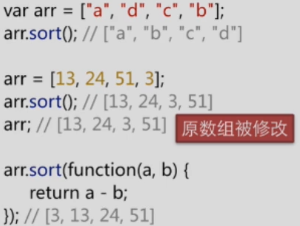
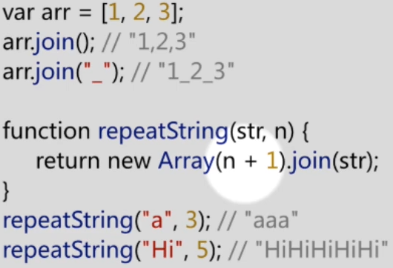


Arr.unsift(0)；在arr数组的首部加一个0. Delete arr[2];将索引为2的元素置为undefined。

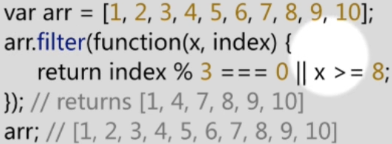
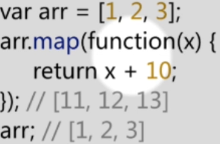
Arr.length -=1;删除尾部的元素。Arr.pop();删除尾部的元素。Arr.shift();头部删除元素。

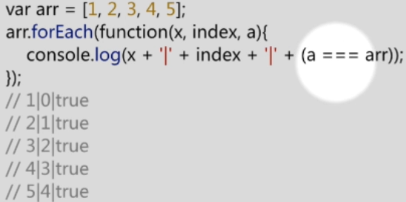
稀疏数组：并不含有从0开始的连续索引，一般length属性值比实际元素个数大。不常见。

27： 数组原型方法：join：将数组转为字符串。reverse：将数组逆序（原数组被修改）。concat：合并数组（原数组并没有修改）；slice返回部分数组slice（1，-1）从索引1到最后一个，左闭右开区间。splice：数组拼接；

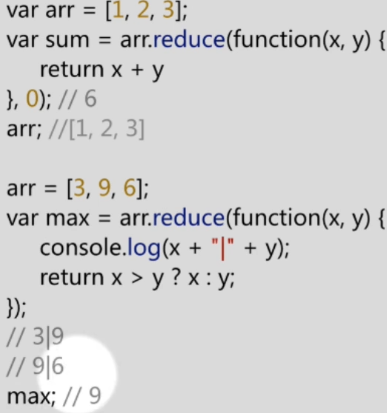
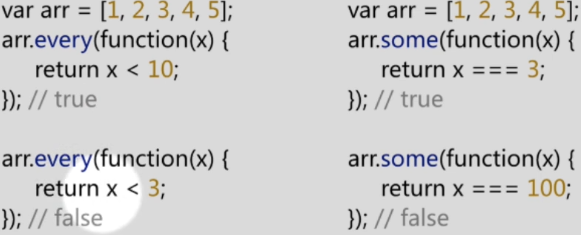


Es5新特性：forEach:数组遍历；map：数组映射;filter:数组过滤；

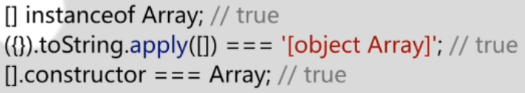
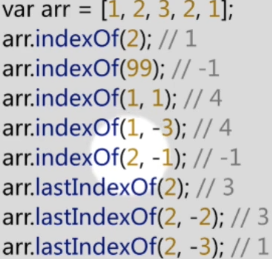


function参数：x值；index索引；a数组。

every和some：数组判断；reduce和reduceRight；后者是从右边开始遍历。

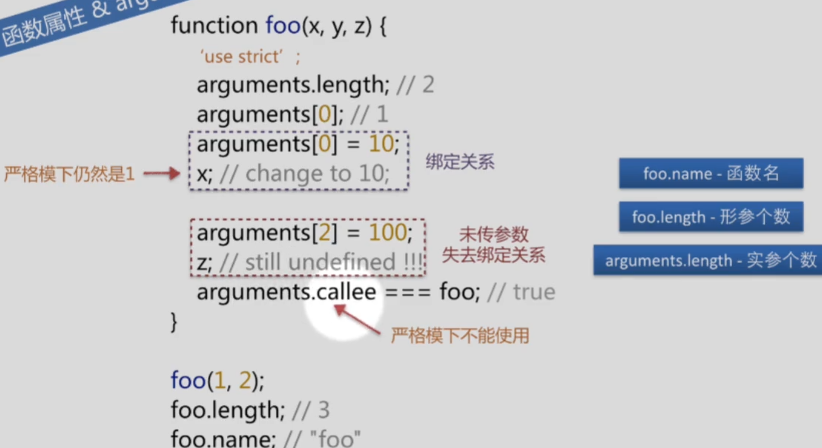


其中reduce的function之后有0则会多加一个0的参数。都是上一轮的结果作为下一轮的参数。indexOf和lastIndexOf：数组检索；后者是从右向左检索。Array.isArray():数组判断



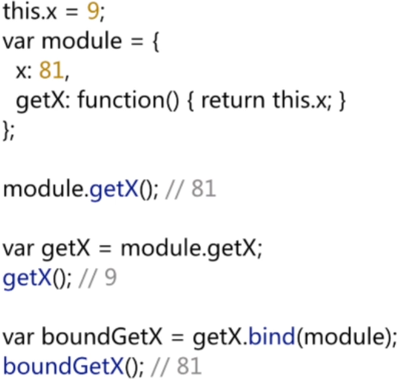
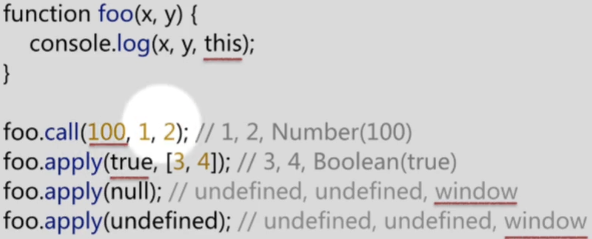
28：this:全局的this指向全局对象，浏览器下就是window；一般函数的this也相当于全局对象；作为对象方法的函数的this，调用时指向对象； 对象原型链上的this，调用时可以指向对象。

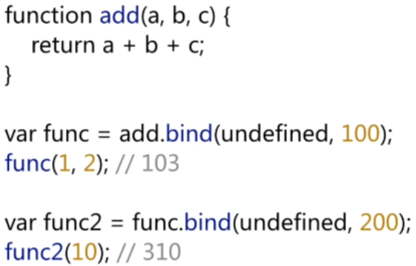
29：函数属性：arguments：实参个数

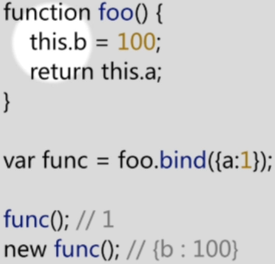


Call和apply：下面foo.call(100,1,2)其中100是作为this的对象。

Bind：es5才提供；修改this的指向；下面的例子中变量getX（）调用的module中的this.x指全局的this，最后的bind指定了this的指向。



最后的a是100，b是200，c是10.



30：闭包：不同于一般的函数，他允许一个函数在立即词法作用域调用时任可以访问非本地变量。使用灵活方便，便于封装，但也造成空间浪费，内存泄漏，性能消耗：



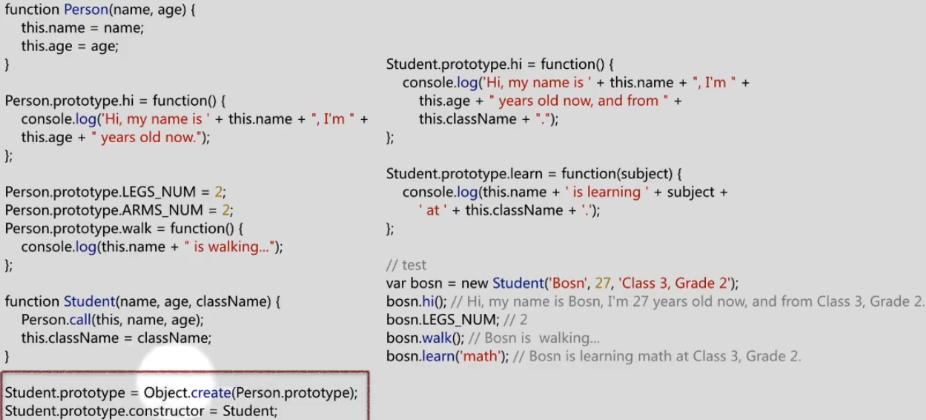
31：利用函数作用域封装，在函数最外层或者函数前面加！或者+来将函数声明改为函数表达式，来防止全局变量。

32：函数声明 若发生命名冲突会覆盖，变量声明发生命名冲突会忽略。

33：OOP：继承；封装；多态；抽象。

34：要分清原型和prototype属性。原型通常都是构造器的prototype属性。

35：案例：

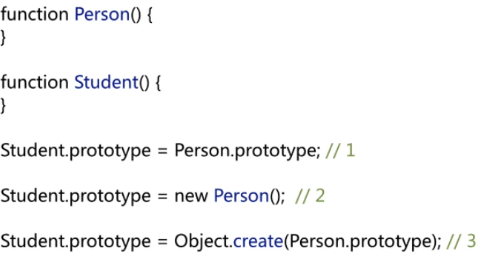


其中Object.create如果省略，则为student添加属性时也会为其父类person也会创建。constructor 属性是专门为 function 而设计的，它存在于每一个 function 的prototype 属性中。这个 constructor 保存了指向 function 的一个引用。

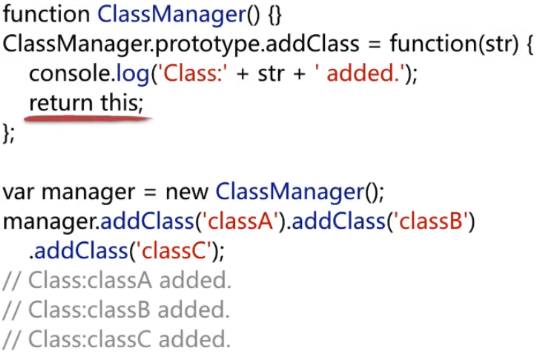
36：添加对象的prototype的属性是没有问题的，如果修改prototype对象那么对于之前的对象是不起作用的，只对新的对象产生作用。

37：instanceOf：其右边必须是函数；左边一般是函数，如果是基本类型就返回false；

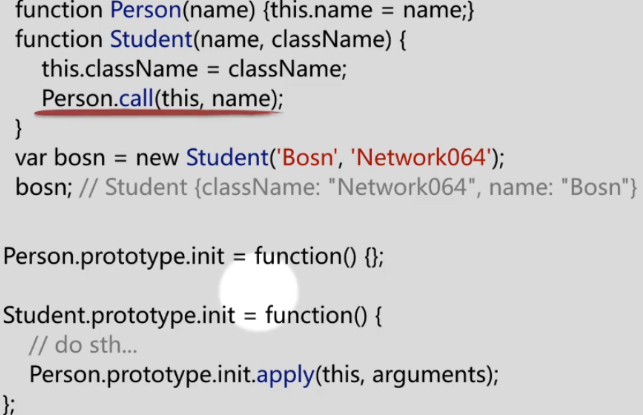
38：下面的继承方法，1是错的，修改student时会将person也会修改。



39：链式调用：return this；始终返回classManager的引用，过可以实现脸是调用。



40：调用子类方法是同时调用父类方法：call和apply实现父类调用。

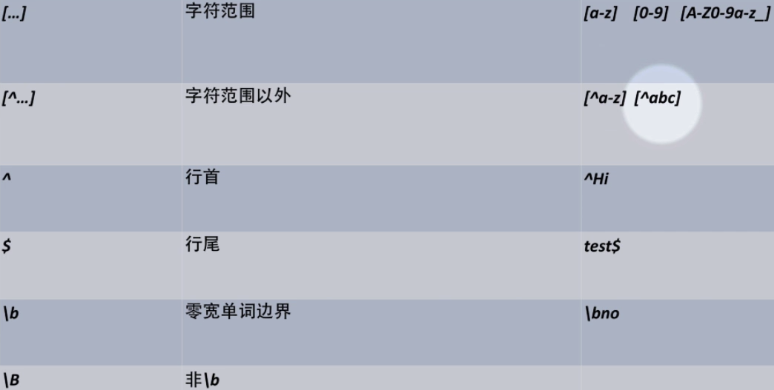


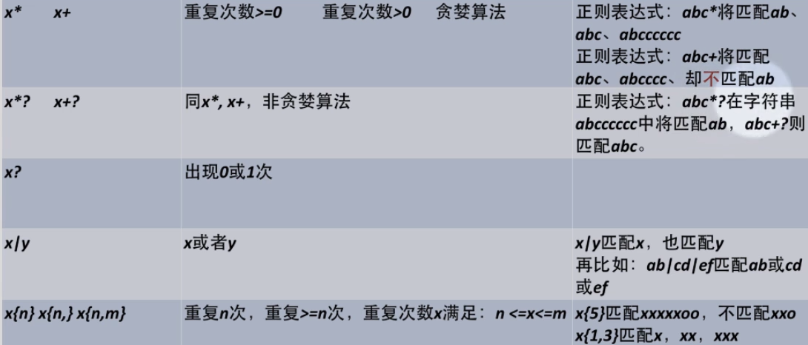
41：正则表达式：转义使用\

[a-z]{4}:四个字母；\d\d\d:三个数字；

js的例子：/\d\d\d/.test(“123”); //true; new（可略）RegExp(“Bosn”).test(“hi,Bosn”);

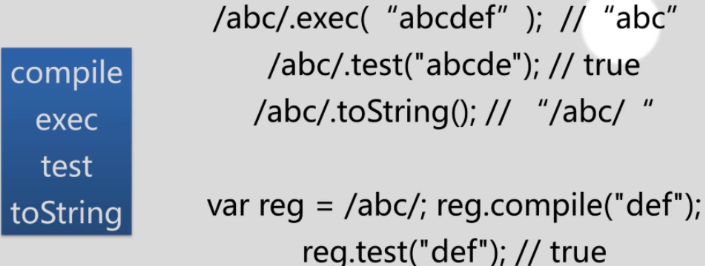
大小写敏感，





三个标识：global：匹配全部；ignoreCase：不区分大小写；multiline：跨行匹配或替换

RegExp对象方法。Compile改变正则的对象。



Canvas：

1：添加到html5标签中

<**canvas id="canvas "width="1024" height="700" style="border**: 1**px solid #aaa**;**display**: **block**;**margin**: 50**px auto**;**"**></**canvas**>

2：Js中引用：

var canvas =document.getElementById("canvas");把canvas拿出来

var context = canvas.getContext("2d")；得到绘图的上下文环境使用context进行绘制

3:绘图：基于状态的；先设置一个状态，之后调用具体的函数进行具体的绘制。

context.moveTo();和context.lineTo();来设置划线的路径；

context.beginPath();和context.closePath();来区分不同路径；close时会将不封闭曲线封闭；如不想封闭就只调用begin。

context.lineWidth = 5;来设置直线的宽度；

context.strokeStyle = "red";来设置直线的style，大多是颜色；

context.fillStyle = "black";和 context.fill();来设置填充的颜色块。

context.arc(300,300,200,0,1.5\*Math.PI):画弧线；坐标（300,300），半径200，起始弧度0PI，终止弧度1.5PI，默认顺时针。后面加true则为逆时针；

context.clearRect(0,0,50,50);对矩形进行刷新；

3：动画基础：setInterval(function，50) 方法可按照指定的周期（以毫秒计）来调用函数或计算表达式