[**javascript prototype介绍的文章**](http://blog.csdn.net/czh_friend/article/details/1566319)

2007-04-16 11:31 5869人阅读 [评论](http://blog.csdn.net/czh_friend/article/details/1566319#comments)(9) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/czh_friend/article/details/1566319#report)

[prototype](http://blog.csdn.net/tag/details.html?tag=prototype)[javascript](http://blog.csdn.net/tag/details.html?tag=javascript)[function](http://blog.csdn.net/tag/details.html?tag=function)[object](http://blog.csdn.net/tag/details.html?tag=object)[constructor](http://blog.csdn.net/tag/details.html?tag=constructor)[string](http://blog.csdn.net/tag/details.html?tag=string)

JavaScript是基于对象的，任何元素都可以看成对象。然而，类型和对象是不同的。本文中，我们除了讨论类型和对象的一些特点之外，更重要的是研究 如何写出好的并且利于重用的类型。毕竟，JavaScript这种流行的脚本语言如果能够进行良好的封装，并形成一个庞大的类型库，对于重用是非常有意义 的。

网上对于prototype的文章很多，一直没明白核心的思想。最后写了很多例子代码后才明白：prototype只能用在类型上。

以下是一些关于类型和对象的例子，大家看完例子后可能更容易理解类型和对象之间的联系：

|  |  |  |
| --- | --- | --- |
|  | 例子代码 | 说明 |
| 1 | Object.prototype.Property = 1; Object.prototype.Method = function ()  {      alert(1);  }    var obj = new Object();  alert(obj.Property);  obj.Method(); | 可以在类型上使用proptotype来为类型添加行为。这些行为只能在类型的实例上体现。  JS中允许的类型有Array, Boolean, Date, Enumerator, Error, Function, Number, Object, RegExp, String |
| 2 | var obj = new Object(); obj.prototype.Property = 1; //Error  //Error  obj.prototype.Method = function()  {      alert(1);  } | 在实例上不能使用prototype，否则发生编译错误 |
| 3 | Object.Property = 1;  Object.Method = function()  {      alert(1);  }    alert(Object.Property);  Object.Method(); | 可以为类型定义“静态”的属性和方法，直接在类型上调用即可 |
| 4 | Object.Property = 1;  Object.Method = function()  {      alert(1);  }  var obj = new Object();  alert(obj.Property); //Error  obj.Method(); //Error | 实例不能调用类型的静态属性或方法，否则发生对象未定义的错误。 |
| 5 | function Aclass()  {  this.Property = 1;  this.Method = function()  {      alert(1);  }  }  var obj = new Aclass();  alert(obj.Property);  obj.Method(); | 这个例子演示了通常的在JavaScript中定义一个类型的方法 |
| 6 | function Aclass()  {  this.Property = 1;  this.Method = function()  {      alert(1);  }  }  Aclass.prototype.Property2 = 2;  Aclass.prototype.Method2 = function  {      alert(2);  }  var obj = new Aclass();  alert(obj.Property2);  obj.Method2(); | 可以在外部使用prototype为自定义的类型添加属性和方法。 |
| 7 | function Aclass()  {  this.Property = 1;  this.Method = function()  {      alert(1);  }  }  Aclass.prototype.Property = 2;  Aclass.prototype.Method = function  {      alert(2);  }  var obj = new Aclass();  alert(obj.Property);  obj.Method(); | 在外部不能通过prototype改变自定义类型的属性或方法。  该例子可以看到：调用的属性和方法仍是最初定义的结果。 |
| 8 | function Aclass()  {  this.Property = 1;  this.Method = function()  {      alert(1);  }  }  var obj = new Aclass();  obj.Property = 2;  obj.Method = function()  {      alert(2);  }  alert(obj.Property);  obj.Method(); | 可以在对象上改变属性。（这个是肯定的）  也可以在对象上改变方法。（和普遍的面向对象的概念不同） |
| 9 | function Aclass()  {  this.Property = 1;  this.Method = function()  {      alert(1);  }  }  var obj = new Aclass();  obj.Property2 = 2;  obj.Method2 = function()  {      alert(2);  }  alert(obj.Property2);  obj.Method2(); | 可以在对象上增加属性或方法 |
| 10 | function AClass()  {         this.Property = 1;         this.Method = function()         {                alert(1);         }  }    function AClass2()  {         this.Property2 = 2;         this.Method2 = function()         {                alert(2);         }  }  AClass2.prototype = new AClass();    var obj = new AClass2();  alert(obj.Property);  obj.Method();  alert(obj.Property2);  obj.Method2(); | 这个例子说明了一个类型如何从另一个类型继承。 |
| 11 | function AClass()  {         this.Property = 1;         this.Method = function()         {                alert(1);         }  }    function AClass2()  {         this.Property2 = 2;         this.Method2 = function()         {                alert(2);         }  }  AClass2.prototype = new AClass();  AClass2.prototype.Property = 3;  AClass2.prototype.Method = function()  {         alert(4);  }  var obj = new AClass2();  alert(obj.Property);  obj.Method(); | 这个例子说明了子类如何重写父类的属性或方法。 |

   以上例子中，关于通过类型实现重用方面，重要的有：

·例子1：JavaScript中允许添加行为的类型

·例子2：prototype使用的限制

·例子3：如何定义类型上的静态成员

·例子7：prototype在重定义类型的成员上的限制

·例子10：如何让一个类型继承于另一个类型

·例子11：如何在子类中重新定义父类的成员

可见JavaScript能够实现的面向对象的特征有：

·公有属性(public field)

·公有方法(public Method)

·私有属性(private field)

·私有方法(private field)

·方法重载(method overload)

·构造函数(constructor)

·事件(event)

·单一继承(single inherit)

·子类重写父类的属性或方法(override)

·静态属性或方法(static member)