学海泛舟系列之二

史上被骂最多的编程语言—JavaScript

金旭亮

注：

这是一个系列文章，访问以下链接查看全部文章

《学海泛舟系列文章开篇语》
（http://blog.csdn.NET/bitfan/article/details/10049513）

==========

一、世无英雄，遂使竖子成名

1 Web客户端编程语言事实上的王者

（1）JavaScript为何成功？

几乎所有的JavaScript书都会先介绍一番JavaScript的历史，比较有趣的是我在资料中看到这样一句话：

JavaScript的设计者Brendan Eich曾在一次采访中说，JavaScript“几天就设计出来了”。

Ruby的设计者—松本行弘为此感叹：

“这样的出身，得到这样的成功，还真让人出乎意料，……”，

“但由于开发周期短，确实也存在着不足……”。

确实，作为一种编程语言，JavaScript并不完美，在实际开发中，JavaScript同样也存在着严重的问题，其中最突出的就是早期没有完整的规范，导致浏览器厂商各行其是所带来的兼容性问题。

但奇怪的是，就是这样一个并不完美的东西，最终却成为了Web客户端编程语言事实上的王者，而它的种种缺陷，却给我们带来了丰富多彩的各种JavaScript库及框架，整个JavaScript社区则生机勃勃，活力四射，这真是一件很有趣的事。

在我看来，JavaScript的成功在于它“生逢其时”，赶上了互联网大发展的时代契机。而当时“时无英雄，使竖子成名”。

读者可能不服气，那建议你看看那本著名的、指出过“10000小时专业训练是成为天才的必要条件”的《异类》，在此书中，作者Malcolm Gladwell列举了诸多二十世纪在IT业呼风唤雨的大人物：微软比尔盖茨、保罗艾伦和史蒂夫鲍尔默，苹果的乔布斯，Google的埃里克施密特，Sun的比尔乔伊……，他们全部出生在1953~1956之间；

再看看当前中国IT大佬们—李彦宏（1968.11）、张朝阳(1964)、马化腾(1971)、丁磊(1971)，马云（1964），刘强东（1974）……，他们的出生日期都集中在上世纪60与70年代交替的时期，……

为什么会这样呢？请结合他们创业的时代大背景看看就明白了。

时势造英雄。JavaScript就是这样的幸运儿。生逢其时，伴随着互联网的发展，有不计

其数的Web应用是使用它编写的，JavaScript应用得越广，其社区就越活跃，优秀的程序员不断为JavaScript贡献大量的框架和库，还有Google和微软这样的大公司不断地想办法弥补或改进JavaScript的现有缺陷，推出新的脚本语言和优化其引擎……

一切都构成了一个完整的生态圈。以后的脚本语言，哪怕比JavaScript要好很多，也难以重现JavaScript的辉煌了，JavaScript已经成为Web客户端事实上的王者。

（2）为什么JavaScript招人骂？

有趣的是，尽管JavaScript用得如此广泛，但仍然承担了很多开发者的抱怨，我估计JavaScript可能是“到目前为止，计算机软件史上被人骂得最多的编程语言”。

为什么会这样呢？

我看原因有三。

首先是至今犹存的浏览器兼容性问题，请看以下代码：

```javascript
function  showInformation(info) {

        var h3 = document.getElementById("info");
        //适合于IE,Chrome, Opera

        if (h3.innerText!==undefined) {

           h3.innerText = info;

        }

        else {

            //适合于FireFox

            h3.textContent = info;

        }

```

上述代码只是简单地给一个<h3>元素赋值，你可以看到，代码必须针对不同的浏览器写不同的代码，更让人郁闷的是，还可能为同一种浏览器的不同版本写不同的代码，有些浏览器（比如IE6）生命周期之长让人无奈，这些都强迫Web开发者把大量时间花在**测试**兼容性这种毫无成就感的工作之上，实在颇有点浪费生命的感觉。

大家都觉得不爽的地方，就是创新点，于是有了以**jQuery**为代表的大量的JavaScript库和框架，其成功在于有意无意地迎合了人们的这种偷懒的心理：

“**把麻烦丢给别人，把方便留给自己**”。

JavaScript招人骂的第2个原因是它名字引起的误会。

JavaScript，名字中有个**Java**，初次听到几乎100%的人都以为它与Java有着密切的关系，而且很有可能就是Java语言的子集。

但事实上，叫“李富贵”的人可能在街上讨饭，叫“刘美美”的其实长得不怎么样，而取名“杨德有”的人其实是个小人，干了很多见不得人的事……

这世上名不副实的多了，其实，JavaScript与Java根本是两种语言，取这个名字纯粹是最早设计并实现JavaScript的那批人想攀高枝拍Java马屁的结果。

由于许多人把它当成自己所熟悉的Java语言去用，结果发现它根本不是自己所熟悉的那种

编程语言，加上下面要讲的第三个原因让人编起程来很不爽，许多人愤怒了："NND，给这丫骗了！"

JavaScript招人骂的第3个原因其实与语言本身无关，而与浏览器相关。

许多人最早使用JavaScript不是因为喜欢这种语言，而是工作中必须用它，在实际工作中用得最多的就是DOM，而DOM API的设计与各浏览器的具体实现 实在不怎么样，按照jQuery设计者John Resig的观点，它绝不可能获得任何"年度友好API（Friendliest API of the Year）"的奖项！

2 精华与糟粕的并存JavaScript

Douglas Crockford写了一本《JavaScript：The Good Parts》，在书中他这样写到：

JavaScript建立在一些非常好的想法和少数非常坏的想法之上。

那些非常好的想法包括函数、弱类型、动态对象和一个富有表现力的对象字面量表示法，而那些坏的想法包括基于全局变量的编程模型、缺乏块作用域、"保留"了一堆根本没用到的保留字，不支持真正的数组（它所提供的类数组对象性能不好）等等。

还有一些是"鸡肋"，比如with语句，原始类型的包装对象，new,void等等

JavaScript受到的主要批判有：

（1）无法应对复杂的互联网应用程序，不支持大家已普遍熟悉的以类为模板的面向对象编程方式

（2）运行速度慢，其对象内部采用散列表形式组织，相比数组和结构体，存取速度慢

（3）不支持多核CPU，JavaScript没有线程的概念，也缺乏必要的线程同步手段，使得它几乎无法编写能充分应用客户端多核CPU计算能力的代码

（4）浏览器兼容性问题是硬伤

（5）……

不少被批判的内容并不全是事实，或者说现在不少已经有很大改善，但JavaScript身上的这些骂名是洗不掉的了。

其实，JavaScript本身有很多精华，下面的内容就集中于JavaScript的这些亮点之上。

二、JavaScript技术导航

谈到JavaScript技术，其实应该区分以下三个概念：JavaScript语言、JavaScript库和JavaScript宿主。

JavaScript语言的学习主要是JavaScript语法学习，JavaScript宿主是指JavaScript程序的运行环境，通常是浏览器，浏览器提供了许多对象（比如window,document等），JavaScript代码可以直接调用它们，另外，浏览器还包容一个专门负责运行JavaScript代码的组件，我们把它称为**JavaScript引擎**，在实际学习过程中，一般不需要深入地了解JavaScript引擎的内部运行机理。JavaScript库通常是指由JavaScript社区所贡献出来的能完成特定功能的打包在一起的JavaScript代码。

在学习过程中，通常是把JavaScript语言与JavaScript运行环境所提供的对象和实现的功能"打包到一块"作为一个整体学习，因此，下面的介绍不再明确地区分哪些内容属于JavaScript语言特性，哪些功能实际上是由宿主环境提供的。至于JavaScript库，不在本文的介绍范围之内。读者可自行阅相关的技术书籍。

1 掌握JavaScript基础编程技能

（1）第一件事情，弄明白在哪儿写JavaScript代码

三种方式写JavaScript代码。

- **Inline JavaScript**：直接将简短的JavaScript代码嵌入到HTML元素声明中：

```html
<a  href="/about" onclick= " alert('this is the thing');">About</a>
```

- **Embedded JavaScript**：将javaScript放到<script>元素中

```javascript
<script  type="text/javascript">

//在此写JavaScript代码

```

- **链接外部JavaScript文件**：将JavaScript代码放到独立的.js文件中，然后在<script>元素引用它：

```html
<script  type="text/javascript"  src="js/external.js"></script>
```

（2）快速了解JavaScript语法基础

学习这部分内容，可以与C/Java/C#的基础语法相对照，重点关注其不同点就行了，以下是部分要点：

- JavaScript定义了四种基本数据类型：numbers, strings, Booleans, undefined和null，其余所有的都是对象。
- JavaScript所有的数都是64位浮点数，还有一个常量叫NaN（Not a Number），在编程中常用。
- JavaScript有一些比较独特的运算符，列举几个：

===（严格判等运算符）、!!（把后面跟着的表达式变成一个bool值），方括号运算符（可以用于访问对象属性）

- 变量的作用域：JavaScript没有块作用域，但有函数作用域。即：定义在函数中的参数和变量在函数外部不可见，并且在一个函数中的任何位置定义的变量在该函数中的任何地方都可见。这点与C和Java等编程语言都不一样。
- ……

（3）比较独特与有用的内部对象

JavaScript本身提供了一些内部对象，可以在代码中直接使用，列举几个：

- **数组**：JavaScript其实没有传统意义上的数组，因此，你应该把它看成是“另外一种东西”，需要花点时间去明白它的特性。
- **全局对象**：JavaScript中有一个全局的global对象，除了那些有明确对象归属的成员，其它所有的东西都成为它的成员，在浏览器环境中，window对象就是全局对象。
- **正则表达式**：正则表达式在处理字符串上功能强大，花时间在这上面是值得的。

- **timer对象**:可以用它实现定时调用
- ……

（4）JavaScript代码调试方法

有几种方法可以调试JavaScript代码，最土的一种是使用alert()输出信息，比较专业的是使用console.log和设置断点。

每个Web开发者都一定要至少掌握一种浏览器所提供的调试工具：

Firebug（Firefox）、IE Developer Tools（IE 8以后）、Opera Dragonfly（Opera 9.5以后）、WebKit Developer Tools（ Safari和Chrome）

大多数浏览器调试工具都可以使用F12这个热键呼叫出来，并且其提供的功能都很强大。

另外，一些IDE（比如Visualstudio），也支持对JavaScript代码的跟踪与调试。

2 把握JavaScript编程语言的精华

在学习JavaScript的过程中，我建议别把JavaScript看成是一种OO语言，应把它看成是一种函数式语言！

另外，重点搞掂**函数**、**对象**、**闭包**三个东东，则JavaScript精华尽在我手！

首先，我们先来摆函数的龙门阵。

（1）函数

JavaScript中函数是“一等公民”。理解JavaScript的函数是打开这门编程语言奥秘的钥匙，由它可以引申出N多重要的特性。

函数是一个对象

JavaScript使用function关键字定义函数：

```javascript
function add(x, y) {

            return x + y;

}
```

函数可看成是一个“函数”对象。函数名是指向这一“函数”对象的指针，可以有多个变量引用同一个函数对象：

```javascript
console.info(add(100,200));  //300

var other =add;  //other和add引用同一函数对象
```

函数中定义的变量是私有的，因此，JavaScript变量只有两种作用域：全局的和由函数所限制的局部作用域。这点非常重要。

函数可以没有名字，我们通常把这种“匿名”函数赋值给一个变量，通过变量来调用它：

```javascript
var square = function (x) { return x * x; }
```

牢记“函数是一个对象”，对看懂许多JavaScript代码至关重要。

返回函数的函数

由于函数是对象，因此，我们可以写出一个返回函数的函数，这是一种非常重要的编程技巧：

```javascript
function func(x,y) {

    var value=300;

    return function () {

        return value + x + y;

    };

}
```

被返回给外界的“内部”函数能够直接访问外部函数的变量，并且需要时它还可以再返回另一个函数，这样便可以构成一个排成“一字长蛇阵”的连续函数调用语句，这在许多JavaScript库中都能看到。

函数的参数

JavaScript对函数的要求极其地宽松。

- 定义函数时，不需要指定参数类型，对于参数值，JavaScript不会进行类型检查，任何类型的值都可以被传递给参数。
- 对于函数参数，如果过少，没得到值的参数为undefined，如果过多，多的会被忽略掉。

JavaScript将所有传给函数的参数放到一个**arguments**对象中（它类似于数组，但JavaScript中没有传统意义上的数组，只能说是类似于数组的对象），使用它可以写出灵活的代码，比如模拟实现OO语言中的方法重载（method overload）.

特别地，函数可以作为另一个函数的参数：

```javascript
var values = [ 213, 16, 2058, 54, 10, 1965, 57, 9 ];
```

如果有C#的delegate经验，看懂上述代码一点也不困难，反过来，理解了上述JavaScript代码，再学习C#的delegate和Lambda表达式也就没多少难度，这就是各种语言均有相通之处的一个例子。

this对象

this对象

JavaScript中的this与Java/C#中的不一样，在JavaScript中，每次对函数的调用都有一个**上下文对象**，this关键字引用它。如果函数归属于某个对象，则this关键字将引用此对象，它就是本次函数调用的上下文。

this引用的函数上下文对象是可以动态改变的，使用函数对象的call方法可以动态地指定它：

```javascript
window.color = "red";

var o = { color: "blue" };

function  sayColor(){

    alert(this.color);

}

sayColor(); //red

sayColor.call(this); //red

sayColor.call(window); //red
```

不少JavaScript库中，使用这个特性玩出了许多花样。

（2）闭包

“**闭包（closure）**”是函数式编程的重要特性，这也是在学习时最让人难以理解的技术关键点之一。请费点脑筋看看以下代码：

```javascript
function a() {

            var i = 0;

            return function b() {

                 console.info(++i);

            };

        };

        var c = a();

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

            c();

        }
```

上述代码中，内部函数b中的代码会访问外部函数a中的变量i，最值得注意的是：执行完c=a()一句后，函数a()已经执行完毕，但由于c引用a()返回的函数b，因此，当前的执行环境被完整保存，从而让i保存上次调用的值！

这就是闭包的神奇特性

这就是闭包的神奇特性。

闭包在JavaScript Library开发中被广泛使用。

有关对闭包内部机理的详细介绍，可以参看《Professional JavaScript For Web Developer》一书。

（3）对象与原型

JavaScript的面向对象特性非常地独特，学习这部分内容时，已非常熟悉C#/C++/Java的朋友一定要Undo已有的知识，才能真正理解它们。

对象的创建

在JavaScript中，对象并不是以类为模板创建出来的，它可以看成是一堆属性的集合，每个属性都有一个name（就是它的属性名字）和Value。

正式地说，JavaScript中的对象是可变的键控集合（keyedcollections），既然是一个集合，所以它支持foreach遍历，也支持动态地给对象添加（直接赋值即可）和删除成员（使用delete内部函数）。

JavaScript中可以使用四种方式定义对象，用起来非常地灵活：

方式一：定义一个空对象，再给它添加成员：

```javascript
var myObject = {};

myObject.name = “John”;

```

方式二：使用对象字面量直接创建对象：

```javascript
var myObject = {

            name:"John",

             age:40

};
```

可以看到，这种形式非常类似于Json字串，事实上，JavaScript提供了相应的机制直接由JSON字串创建JavaScript对象（或反之）

方式三：使用工厂函数构建

new一个空白对象，添加完成员之后，return给外界：

```javascript
function createPerson(name, age, job){

```

```
    var o = new Object();

   o.name = name;

  o.sayName = function(){

  alert(this.name);

};

return o;

```

方式四：通过对象构造函数创建

```
[javascript]

function Person(name, age, job){

this.name = name;

this.sayName = function(){

alert(this.name);

};

}

```

可以看到，这种方式使用this关键字给对象添加成员，使用new关键字调用并创建对象。

通常会把构造函数的首字母设置为大写的。

构造函数其实也是一个函数，不同之处在于调用它时必须要加一个“new”关键字，如果不加这个关键字，则对它的调用被认为是普通函数调用。

使用这种方法构造对象，每个对象都加了一个constructor属性：

```
[javascript]

```

JavaScript对象在运行时可以动态地创建和修改其成员，这就给编程带来了很强的灵活性，下面举一个例子，看看如何在函数内部构建一个数据缓冲区：

```
[javascript]

function getElements(name) {

if (!getElements.cache)

 getElements.cache = {};

```

```
return getElements.cache[name] =

getElements.cache[name] || document.getElementsByTagName(name);
```

上述函数在内部使用一个名为cache的空对象用于保存已访问过的页面元素，仅在首次访问时调用DOM API去获取节点对象，从而提升了性能。

对象原型（Pototype）

这是JavaScript语言中最有特色的地方。

```javascript
function MyObject(name) {

      this.name = name;

  };

  var obj1 = newMyObject("Object1");

  //向原型中添加新成员

  MyObject.prototype.value = 100;

  //新对象与老对象将同时拥有这个新的成员

  var obj2 = newMyObject("Object2");

  console.info(obj1.value);  //100
```

上述代码的背后，其实是以下文字描述的JavaScript内部机理：

每个对象都连接到一个原型对象（Prototype），如果我们添加一个新的属性到原型中，该属性会立即对所有基于该原型创建的对象可见。

各个对象的原型也是一个对象，它们可以“链接”起来，构成一个原型链。

当我们尝试去获取对象的某个属性值，且该对象没有此属性值，那么JavaScript会尝试从它直接关联的原型对象中去获取，如果那个原型对象也没有此属性值，那么会再从这一原型对象所关联的另一个原型中寻找，依次类推，直到该过程最后到达终点Object.prototype。如果想要的属性完全不存在于原型链中，那么结果就是undefined。

JavaScript引入原型，其主要目的之一就是为开发者提供经典的OOP编程风格：以类为模板创建对象和实现继承，其实现思路基于以下事实：

当你创建一个新对象时，你可以选择某个对象作为它的原型。

以下代码使用prototype模拟实现了面向对象编程中的继承特性。

```javascript
function Parent() {
```

```
2.
3.            this.baseFunc = function () {
4.
5.             console.info("基类方法");
6.
7.        };
8.
9.        }
10.
11.       function Child() {
12.
13.           this.childFunc = function () {
14.
15.                console.info("子类方法");
16.
17.           };
18.
19.       }
20.
21.        //形成继承关系
22.
23.       Child.prototype = new Parent();
24.
25.
26.
27.       var obj = new Child();
28.
29.       obj.childFunc();
30.
```

上述示例代码只是使用JavaScript语言特性实现OOP的方式之一，还有不少JavaScript书还介绍了其他实现继承的方法，五花八门，大多比较复杂，理解起来有一定难度，类似于“代码游戏”，这就引发了一个问题：

我们是否一定要使用JavaScript以经典的OOP方式编程？

要回答这样一个问题，先来思索一下另一个相关联的现实问题：

是否所有的Web应用都应该使用面向对象的方式开发？

其实要回答这个问题并不困难。

对于很小的很简单的网站，不用面向对象其实也没什么问题。想想看，整个网站只有几个十几个页面，这也要分成N个类，应用XXX设计模式，再加上能支持分布式缓存和负载均衡N层架构，是不是有点过份？

当然，对于复杂的网站情况就不一样了，这种Web系统后台通常会包容复杂的业务流程，并且很可能需要与多个其他的内部或外部系统进行信息交换，……。实践证明在这种情况下，面向对象是最有效最成熟的解决问题的方法。

JavaScript本身是一种主要用于Web前端的脚本语言，你想想，一个将要运行于客户端浏览器内的Web网页，需要它完成多复杂的事？并且就算它真能完成很多工作，也不能将业务逻辑前移到客户端实现，这会带来巨大的风险。

尽管现在有Node.js之类可以让JavaScript代码跑在服务端，但Node.js更多是基于模块来构建系统，很少使用它来建立一个拥有复杂的继承体系的用户类型系统，这些工作使用标准的OO语言如Java或C#完成更为合适。

另外，JavaSript本身从一开始就没打算设计成纯面向对象编程语言，只是支持OOP，前面也看到了，用原生的JavaScript实现继承都比较麻烦，如果还想实现多态，那就更费脑筋了。

除非你要写一个诸如jQuery之类的框架，那深入研究并把握JavaScript的OOP编程技巧才是必须的，大多数Web开发者直接用JavaScript“原生的”函数、闭包、对象、原型这些

语言特性就足够应付工作所需。

3 探索JavaScript应用技术领域

在介绍完了JavaScript编程语言本身的亮点之后，现在将目光转向JavaScript的应用领域。

（1）BOM

浏览器对象模型（BrowserObject Model，BOM）是由浏览器实现的，可供JavaScript程序调用的一组对象，通过它JavaScript代码可以完成与“控制”浏览器进程相关的许多工作。BOM由一系列相关的对象构成，主要有以下六个：

- window对象，前面也说过，它是JavaScript的最顶层对象，其它的BOM对象都是windows对象的属性。
- document对象表示浏览器中加载页面的文档对象；
- location对象包含了浏览器当前的URL信息；
- navigator对象包含了浏览器本身的信息；
- screen对象包含了客户端屏幕及渲染能力的信息；
- history对象包含了浏览器访问网页的历史信息。

当浏览器装载网页完毕之后，这几大对象就可用了。

基于BOM的编程很简单，主要就是使用JavaScript访问这些对象的相关属性和调用它们的相关方法，查查技术手册（诸如《JavaScript权威指南》那样厚达1000多页的书）就OK了。

（2）DOM

Web网页本身是一个纯文本形式的文档，是由许多嵌套的HTML元素所构成，如果直接按照字符串来处理文档，相当地不便。

浏览器装入HTML文档之后，依据文档中包容的内容，创建出一棵树，并把这棵树称为“DOM（Document Object Model，文档对象模型）”，程序员们通常直接称之为DOM树。

事实上，Web前端编程主要就是围绕着DOM树而展开的。使用JavaScript可以很方便地操控整个DOM树。其基本步骤可以简化为以下两步：

（1）从DOM树中选取要操作的节点（或节点集合），有多种方式完成这一工作，最常用的就是document对象所提供的getElementById( )：

```
[javascript]
```

其它的JavaScript库或框架，比如jQuery,提供了更多更方便的方式选择节点，但其最终目的都是一样的—选择要操控的DOM节点。

（2）针对选择的节点对象，设置其属性、调用其方法或响应其事件：

```
[javascript]
```

记住这两点之后，花些时间看看DOM文档，知道我们可以使用哪些方法、事件和属性，也就够了，在编程上并没有什么特殊之处。

这里特别地说说事件，DOM规范定义了两种事件：**冒泡型**和**捕获型**。这两种类型事件的在DOM树中传输方向正好相反，一个从触发事件的节点向上传播到DOM树的根节点，另一个则从DOM树的根节点向下传播到触发事件的节点。把握这两者的区别在实际开发中很重要，同时注意不同浏览器和不同版本浏览器的具体实现会有所区别。

（3）AJAX与JSON

AJAX其实它是一个很古老的技术，IE浏览器很早就支持它。后来，Google在其Google Map中应用AJAX实现了让人惊讶的Web体用体验，一下子让AJAX成为炙手可热的技术。

AJAX的思想其实很简单：

使用浏览器实现的XmlHttpRequest对象在“后台”向Web服务器发出请求，Web服务器收到请求之后，向浏览器发回数据，浏览器收到之后更新页面。

以下是简化过的完成某次AJAX数据交换功能的示例代码：

```javascript
var myRequest;

//1：依据浏览器的类别创建相应的对象

if (window.XMLHttpRequest) {

   //非IE浏览器

    myRequest = new  XMLHttpRequest();

} else if (window.ActiveXObject) {

     //IE浏览器

    myRequest = new  ActiveXObject("Microsoft.XMLHTTP");

}

// 2: 响应XMLHttpRequest对象的readystatechange事件，为其定义回调代码：

myRequest.onreadystatechange = function(){

    if (myRequest.readyState=== 4) {

       //收到的数据放在myRequest.responseText中，接着可使用DOM来更新页面

        ……

    }

};

// 发送请求

myRequest.open('GET', 'simple.json', true);
```

可以看到，数据取回以后，通常使用DOM来更新页面。

服务端返回什么样格式的数据完全可以由开发者自行决定，现在比较流行的是直接返回

JSON格式的数据，一是方便，JavaScript能直接解析它；二是数据紧凑，JSON比XML数据量要小得多，三是跨平台性好，目前许多手机应用都采用JSON来从服务器上提取数据。

（4）各种JavaScript框架

当前在互联网上可以找到N种JavaScript框架或库，各有各的用途，各有各的应用场景。有大量的书籍和资料介绍这些框架，在此就不废话了。如果你时间有限，并且是初学者，那我建议你只要好好学习jQuery就够了，这是一个当前应用极为广泛的成熟的框架，设计得非常出色，你可以举一反三，再学习其他框架也并不困难。

（5）服务端的JavaScript——Node.js

JavaScript最初是一种运行在浏览器环境中的脚本语言，但Google推出的了一个名为Node.js的JavaScript运行环境，使用其研发的JavaScript V8引擎，使得JavaScript代码可以运行在服务端。

Node.js采用事件驱动和异步I/O，高度模块化，性能表现相当优异，属于近几年的技术热点，有机会时我针对它另写篇文章向大家介绍。

（5）开发Win8与智能手机应用

当前主流的智能手机操作系统——iOS和Android，其浏览器都配备了JavaScript脚本引擎，并且对HTML5特性的支持比较好。因此，现在有许多使用JavaScript开发的移动Web应用，开拓了JavaScript应用的新天地。

顺便说一下，使用JavaScript也能在微软的Windows 8中开发新的Windows 8类型的应用并放到Windows商店上去卖，但目前其前景并不算明朗，感兴趣的朋友可以自行钻研一下。

有关移动Web开发技术，计划放在另外的文章中介绍。

三、我的JavaScript学习建议

不管你对JavaScript感觉如何，只要你从事互联网应用开发，JavaScript就是你绕不过去的。因此，本小节就针对初学者谈谈JavaScript的学习建议。

我总结了一下，JavaScript大致可以分为以下几个学习阶段：

（1）开始起步：这一阶段主要是学习与掌握基本语法

比如了解JavaScript有哪些关键字，有哪些数据类型，变量作用域是如何定义的，分支与循环语句如何编写等等，这没什么好说的，几乎是学习所有编程语言都要完成的工作。

只要你学过C/Java/C#/C++，掌握JavaScript基本语法用不了两小时。

（2）深入探索：这一阶段主要是把握JavaScript特性，学习典型编程技巧，理解相应内部机理

通常这一阶段需要阅读大量的JavaScript技术书籍，并花费相当的时间编写各种小的Demo，才能真正掌握好JavaScript编程语言，为下一阶段打下基础。需要重点把握的内容在前文已有介绍。

（3）学以致用：基于各种JavaScript库（或直接使用JavaScript）编写实际应用，并进一步地学习JavaScript相关的技术，比如Node.js或进一步学习开发智能手机Web应用等技术。

这就没什么好说的啦，依据你的工作与学习需求，选择相应的内容来学习。不打算成为JavaScript专家的话，大多数人应该都会长期停留在这一阶段。

（4）游刃有余：处于这一阶段的人，已经具备编写浏览器兼容性和可重用可扩展的库或框架的能力

这部分人精通JavaScript同时又有自己的想法，往往会致力于开发新JavaScript库及框

架，或者是进一步扩充JavaScript的具体应用领域。

（5）开疆拓土：设计全新的脚本编程语言和运行平台

这活个人英雄主义可能就行不通了，现在通常是由大公司或“大牛人”来做，比如Google推出了V8引擎，还设计了一种JavaScript的替代语言—Dart（https://www.dartlang.org/），意图弥补JavaScript的缺陷。

而C#的设计者—AndersHejlsberg，就参与了微软推出的TypeScript（http://www.typescriptlang.org/）脚本编程语言的设计与研发工作。与TypeScript类似的另一种知名的脚本编程语言是CoffeeScript（http://coffeescript.org/），它们都构建了一套“自认为”更合理的语法体系，但并不实现自己的脚本运行引擎，而是把程序代码“编译（Complile）”成标准的JavaScript代码，在现有的JavaScript引擎上运行。

小结：

JavaScript虽是一种有着近20年历史的编程语言，但其生命力却日见旺盛，伴随着互联网的发展，它已经成为21世纪最重要的编程语言之一。

JavaScript技术领域包容相当多的内容，并且还在扩展当中。这篇小文从诸多JavaScript技术书籍和教学资源中选出了一些我觉得比较重要的内容整理并介绍给大家，挂一漏万，且可能存在着错误，只希望能给初学者以一点引导与帮助，吾愿足矣！

==========

预告：

本系列的下一篇文章将介绍HTML 5技术领域。