Javascript的熟练之路，小弟来了。



JavaScript简介：JavaScript一种[直译](http://baike.baidu.com/view/295412.htm)式[脚本语言](http://baike.baidu.com/view/76320.htm)，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的[解释器](http://baike.baidu.com/view/592974.htm)被称为JavaScript引擎，为[浏览器](http://baike.baidu.com/view/7718.htm)的一部分，广泛用于[客户端](http://baike.baidu.com/view/930.htm)的脚本语言，最早是在[HTML](http://baike.baidu.com/view/692.htm)（[标准通用标记语言](http://baike.baidu.com/view/5286041.htm)下的一个应用）网页上使用，用来给[HTML](http://baike.baidu.com/view/692.htm)网页增加动态功能。与1995年，由[Netscape](http://baike.baidu.com/view/153922.htm)公司的[Brendan Eich](http://baike.baidu.com/view/2135520.htm)设计实现，其相关标准为[ECMAScript](http://baike.baidu.com/view/810176.htm)，当前的版本为[ECMAScript](http://baike.baidu.com/view/810176.htm) 2016。

其组成包括3部分：**[ECMAScript](http://baike.baidu.com/view/810176.htm" \t "_blank)**，描述了该语言的语法和基本[对象](http://baike.baidu.com/subview/2387/10666278.htm)；[文档对象模型](http://baike.baidu.com/view/758570.htm)（**DOM**），描述处理网页内容的方法和接口；浏览器对象模型（[**BOM**](http://baike.baidu.com/subview/126558/5073177.htm)），描述与浏览器进行交互的[方法](http://baike.baidu.com/subview/169819/5086683.htm)和[接口](http://baike.baidu.com/view/159864.htm)。

Tip:常见BOM对象包括，document, location, history, navigator等。

基本数据类型，如下表所示。

|  |  |
| --- | --- |
| 数据类型 | 阐述 |
| 数值 | 整数: 55; 浮点数: 3.0,2.5e12; isNaN(3.14), NaN非数值;  isfinity(21), Infinity无穷大;parseFloat()浮点数, parseInt()整数; |
| 字符串 | 转义序列: \t制表符, \n新行  concat:连接字符串，返回字符串的一个拷贝  indexOf/lastIndexOf:返回指定值在字符串里出现的第一个位置,最后一个位置  replace:子串替换  split:把字符串分解为一系列子串，保存到数组，返回一个新数组  substr:从指定的开始位置，提取指定数量的字符组成字符串  toLowerCase/toUpperCase:转换为小写/大写字符 |
| 布尔值 | bool, 可以将如下值作为false: undefined, null, 0, NaN, ""，此外注意!操作符 |
| 数组 | var myArray = new Array(); var myArray = [];  concat:合并多个数组  join:把多个数组元素合并为一个字符串  toString:以字符串形式返回数组  index/lastIndexOf:在数组搜索指定元素  slice:根据指定的索引和长度返回一个新数组  sort:根据字母顺序或提供的函数对数组进行排序  spice:在数组指定索引添加或删除一个元素 |

控制结构：

|  |  |
| --- | --- |
| 控制结构 | 阐述 |
| 条件语句 | if(x==2){} else if(xxx){} else{} 需要注意的是==表示值相等，可能存在隐式转化，不够准确，比如2=='2',结果为true;而===表示值和类型都相等;常见比较符,!=, >, <, >=, <=  switch(xxx){case xx: xxx;break; .. default:xxx} |
| 循环结构 | while(xxx){}, do{}while(xxx), for(;;){}, for(i in days);break, continue |

函数：javascript中的函数形式非常简单，主要注意的是其相关的变量的作用域，当回调函数多了时，会涉及闭包等概念。

|  |
| --- |
| <script>  function addTax(subtotal, taxRate) {  var total = subtotal \* (1 + (taxRate / 100));  return total;  }  </script> |

常见**BOM**对象，如下表所示。

|  |  |
| --- | --- |
| 对象 | 阐述 |
| window | 提供如下方法  alert(), 消息提示窗（模态）  var boolValue = confirm('xxx'),  prompt(arg1,arg2)，参数二用于输入默认值 |
| document | .getElementById();此外可以通过innerHTML获取和设置<div>中html内容  .getElementsByTagName('div')获取特定的全部标签  .getElementsByClassName('div')返回具有特定的class属性值 |
| history | .forward(),.backward()相当于前进和后退，.next()获取下一个页面（向前或向后），  .go(-3)回退页面，.go('aaa.com')获取历史中第一个匹配的URL |
| location | .href完整url，.protocol协议名（包括:）, .hash为锚点,如'#list'  .host主机名加端口(可以分拆为hostname和port)  .pathname为项目完整路径，.search为查询字符串（包括?）,  页面导航的两种方式：location.href='www.xionger.com',location.replace('xx');  刷新页面：document.reload(true);true表示强制从服务器拉取，默认拉取本地 |
| navigator | 包含大量浏览器信息，如appName,appVersion,language,platform,cpuClass等 |

常见**内置对象**和函数，如下表所示。

|  |  |
| --- | --- |
| 对象 | 阐述 |
| Date | var mydate = new Date();new Date(milliseconds), new Date(dataString)  获取时间:.getFullYear(),.getMonth(),.getDate(),.getDay(),.getDay()  ,.getHours(),.getMinutes(),.getSeconds()  此外，还可以用对应的.setXxx()进行修改 |
| Math | .ceil/.floor向上/向下取整;.max(a,b,..)/min(..)最大最小值  .random()随机数;数学常数，如.PI,LN2,LN10,E(自然对数的底,2.718)等 |
| 关键字with | 类似.NET中的using,可以用于省略前缀对象，不常用 |

Javascript语法相对比较简单，但也需要养成良好的编程习惯，比较有实际意义的建议如下所示。

|  |  |
| --- | --- |
| 编程习惯 | 阐述 |
| 谨慎使用JS | 尽量使用常用的界面元素；样式依靠CSS而不是JS |
| 编写简单易读的代码 | 合理使用注释,/\*\*xxx\*/, //; 命名适当, 常量大写，变量骆驼命名法  尽量复用代码；不要假设，当函数内部运算失败时，返回false |
| 平稳退化 | 就是当用户的浏览器缺少某些让页面设计充分展示的功能，或关闭该功能时，仍然可以将站点的内容呈现给用户 |
| 渐进增强 | 首先建立一个大部分用户可以访问的网站，然后添加额外的功能层次，服务更高要求的用户（即开启相关功能的用户） |
| 妥善的错误处理 | try{xx}catch(err){xx} |

接下，通过一个表格简要介绍javascript有关面向对象编程的相关概念，javascript提供了面向对象3大特性中的两个，封装和继承。

|  |  |
| --- | --- |
| 概念 | 示例 |
| 对象创建 | 方式1,直接创建对象:  var obj = new Object();  obj.info = 'I like you';  myNewObject.showInfo=function(){altert(**this**.info);};//注意这儿为匿名方法  方式2，构造类，然后新建对象:  function MyObjectType(info) {  this.info = info;  this.showInfo = function() { alert(this.info) }  this.setInfo = function(newInfo) { this.info = newInfo; }  }  var obj = new MyObjectType(); |
| 继承 | 这儿会用到prototype关键字,首先介绍扩展方法，该方式也可以用于扩展js内置对象  MyObjectType.prototype.sayHello = function() { alert('Hello, Xionger');  继承示例  function Pet() {  this.name = '';  this.setName(newName) { this.name = newName; }  }  function Dog() {  this.breed = '';  this.setBreed = function(newBreed) { this.breed = newBreed; }  }  Dog.prototype = new Pet(); |
| 扩展 | 对于javascript来说，在构造函数内声明的变量只能在对象内部使用，对于外部是不可见的。如果想从外部访问这些变量和函数，需要在赋值时使用关键字this |

Tip：

可以使用如下方式判断一个javaScript函数是否存在, if(typeof document.getElementById == 'function'){}。此外type(xxx)的返回值还包括，'number'操作数为数值；'string'操作数字符串；'boolean'表示布尔类型；'object'表示对象；undefined/null表示未定义/null。

**JSON**是JavaScript对象的一种简单紧凑的标签，使用JSON时，对象可以简单的转换为字符串来进行存储和转换。

|  |  |
| --- | --- |
| 概念 | 示例 |
| 将JSON字符串转化为对象 | 方法1: var myObject = **eval**('(' + jsonObjectString + ')');  方法2(在浏览器支持时):**JSON.parse**(xxx); |
| 将对象序列化为JSON字符串 | var obj = new Object();obj.name = 'xionger';**JSON.stringif**(ibj); |
| JSON的数据类型 | 包括数值，字符串，对象等，但比如Date,Function等不属于JSON标准，需要编解码之后才能使用 |
| 模拟关联数组 | var myArr =[]; myArr[0]='Monday'; myArr[1]='Tuesday';  var myRefer = {'firstDay':'Monday', 'secondDay':'Tuesday'}  myRefer['secondDay'] = 'Tuesday' |
| 使用JSON创建对象 | var user = {  'name':'xionger',  'setName':function(newName){this.name=newName;}  },主要注意这种方式只能用于js环境，而不能用于数据交换 |
| 安全性 | Eval()函数可以执行任何js命令，不过存在潜在风险，推荐使用内置JSON解析器 |

**事件**及事件响应：对于主要致力于为页面添加交互性的脚本语言来说，事件功能必不可少，常见的事件处理其如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 事件处理器 | 示例 | 事件处理器 | 示例 |
| onBlur/Focus | 用户离开/进入字段 | onChange | 修改了值，正要离开 |
| onClick/DbClick | 点击/双击鼠标 | onKeydown/up/press | 在元素激活时，一个按键被按下/释放或者按下然后释放 |
| onLoad | 对象已加载 | onMousedown/up | 鼠标在一个对象上按下/释放 |
| onMousemove | 鼠标在其上方移动 | onMouseover/out | 鼠标移动到对象上/离开对象 |
| onReset | 用户重置表单 | onSelect | 用户选择了对象的一些内容 |
| onSubmit | 提交表单 | onUnload | 关闭浏览器窗口 |

事件相关概念

|  |  |
| --- | --- |
| 概念 | 示例 |
| 添加事件处理器 | document.getElementById('mybutton').onclick  =function(){alert('you click button');}; |
| 删除事件处理器 | document.getElementById('mybutton').onclick=null |
| 默认操作 | 比如用户点击一个链接时，我们可以通过修改src修改默认跳转的页面  document.getElementById('mybutton').onclick  =function(){this.href('www.baidu.com');};  **禁止默认行为**  可以通过在事件处理函数中添加return false;来阻止默认操作 |
| event对象 | 事件对象，包含事件相关参数。IE：window.event; W3C为事件处理函数的参数e  兼容写法function(e){if(!e) var e = window.event;}  通用属性:  type,事件类型;altKey,ctrlKey,shiftKey, alt/ctrl/shift键是否按下;  clientX,client, 浏览器窗口事件坐标;  screenX,screenY, 相对于屏幕的事件坐标  相异的属性:secElement(IE),target(W3C，接受事件的对象 |
| 添加事件处理器 | IE方式: element.attachEvent('onclick',myFunction);  element.detachEvent('onclick', myFunction);  W3C:第三个参数表示事件的处理顺序是**捕获还是冒泡**, W3C默认为捕获, IE为冒泡  捕获表示处理容器由外到内，而冒泡相反  element.addEventListener('click', myFunction,false);  element.removeEventListener('click', myFunction,false);  这部分兼容性问题完全可以通过jQuery库来避免(其内封装了兼容性的代码) |

**Cookie**:由于Http协议是一种无状态的协议，因此如果需要在会话中共享数据就需要用到cookie，每个域最多保存4KB的cookie。当然在H5标准中还提供LocalStorage和SessionStorage用于增强数据共享能力。

|  |  |
| --- | --- |
| 概念 | 示例 |
| cookie的组成 | cookieName:cookieValue;domain:example.com;path:/documents;  secure:true;expires:UTC时间，如果不设置默认，默认和当前浏览器会话一样长 |
| 编辑cookie | function createCookie(name, val, days) {  if (days) {  var date = new Date();  date.setTime(date.getTime() + (days \* 24 \* 60 \* 60 \* 1000));  var expires = date.toGMTString();  } else var expires = '';  cookieString = name + '=' + escape(val);  if (expires) { cookieString += '; expires=' + expires; }  document.cookie = cookieString;  }}这儿注意**escape**()和**unescape**()函数，他们将对字符串进行编码，使得分号、逗号及空白符号等特殊字符可以存储在cookie中，其方式是将任何ASCII字符都转换为相应的2位或位十六进制格式，如空格%20,&转化为%26 |
| 读取cookie | function getCookie(name) {  var nameEquals = name + '=';  var crumbs = document.cookie.split(';');  for (crumb in crumbs) {  if (crumb.indexOf(nameEquals) == 0) {  var str = crumb.substring(nameEquals.length, crumb.length);  return unescape(str);  }  }  } |
| 删除cookie | function deleteCookie(name){createCookie(name, '', -1);} |
| 多值cookie | var userdata='Xionger|28|SE',通过定界符组合字符串 |

**图形与动画**：

|  |  |
| --- | --- |
| 概念 | 示例 |
| 预加载图像 | 在用户请求之前加载页面可能需要的图片数据  window.onload = function ()  {  var img1 = new Image();  var img\_urls= new Array();  img\_urls[0] = 'http://www.xionger.com/image01.jpg';  img\_urls[1] = 'http://www.xionger.com/image02.jpg';  for(i=0;i<img\_urls.length;i++){  img1.src=img\_urls[0];  }  } |
| 页面元素的动画 | 将DOM元素按一定频率移动即可获得动画，在视频领域，这个频率被称为帧速率，单位为帧每秒fps(frame per second)  单次执行为:.setTimeout(action, delay);多次执行:setInterval(action,delay)  function hide(elementId) {  document.getElementById(elementId).style.display = 'none';  }  window.onload = function() {  var time1 = setTimeout("hide('id1')", 3000);  //clearTimeout(time1);//取消定时器  } |
| 渐变不透明度 | function setOpacity(opac) {  var elem = document.getElementById('id2');  elem.style.opacity = opac / 100;  elem.style.filter = 'alpha(opacity=' + opac + ')';  }  function fadeIn(opac) {  if (opac < 100) {  opac++;  setOpacity(opac);  time2 = window.setTimeout("fadeIn(" + opac + ")", 50);  } else {  clearTimeout(time2);  }  } |
| DOM移位 | function moveItRight() {  var el = document.getElementById('div1');  el.style.left = '50px';  if (parseInt(el.style.left) > (screen.width - 50)) {  el.style.left = 0;  }  el.style.left = parseInt(el.style.left) + 2 + 'px';  time3 = setTimeout(moveItRight, 25);  } |
| 优化性能 | 使用单个定时器；避免为DOM树深层次元素创建动画效果；尽可能使用低的帧速率 |

**DOM**：文档对象模型是一种父子关系组成的层次树形结构，构成当前Web页面的模型，最顶层的对象为window对象，document为其子对象。在document根结点，包含多种不同类型的结点，结点类型nodeType包含在节点元素的属性中，如下表所示。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| nodeType | 描述 | nodeType | 描述 | nodeType | 描述 |
| 1 | 元素 | 2 | 属性 | 3 | 文本(包含空白) |
| 4 | CDATA区域 | 5 | 实体引用 | 6 | 实体 |
| 7 | 执行指令 | 8 | HTML注释 | 9 | 文档 |
| 10 | 文档类型DTD | 11 | 文档片段 | 12 | 标签 |

常见DOM结点的选择和操作如下表所示：

|  |  |
| --- | --- |
| 概念 | 示例 |
| childNodes,parentNode | if(olElement.childNodes[i].nodeType == 1) count++;  rootElement = rootElement.childNodes[0].parentNode; |
| first/lastChild | rootElement.firstChild = rootElement.childNodes[0];  rootElement.lastChild =  rootElement.childNodes[rootElement.length - 1]; |
| previousSibling,nextSibling | 返回前一个/后一个兄弟节点 |
| 结点名称和节点值 | text = rootElement.nodeValue;  xxx = rootElement.nodeName;当节点为元素时，值为元素名称；当节点为属性，值为属性名；当为文本时，值为'#text' |
| 创建新节点 | var newDiv = document.createElement('div');  var newDiv = document.createTextNode('Xionger is a talent');  var newDiv = newDiv.cloneNode(true);参数表示是否为深拷贝 |
| 添加节点 | myDiv.appendChild(newDiv);  myDiv.insertBefore(newDiv, existDiv); |
| 编辑节点 | myDiv.replaceChild(newDiv, existDiv); |
| 删除节点 | myDiv.removeChild(newDiv); |
| 动态加载javascript | function loadScript() {  var scr = document.createElement('script');  scr.setAttribute('src', 'newScript.js');  document.head.appendChild(scr);  }  window.onload = loadScript;//常用于动态加载目录 |
| 读取元素属性 | element.getAttribute('name'); |
| 编辑元素属性 | element.setAttribute('name', 'xionger'); |

常见CSS相关操作如下所示

|  |  |
| --- | --- |
| 概念 | 示例 |
| 内容样式分离 | <link rel='styleSheets' type='text/css' href='style.css'/>  <style>p{color:black;}</style>  样式可复用；结构清晰易读；保证整个站点的总体一致性 |
| DOM的style属性 | myDiv.style.color='white'; |
| 使用className来访问类 | myDiv.className='classA'; |
| DOM中的styleSheets对象 | 页面上样式表的数量: document.styleSheets.length  启用和禁用样式表(可切换): document.styleSheets[0].disabled = true;  document.styleSheets[1].disabled = false;可以使用.styleSheets[0]的title属性来做切换。 |

参考资料：

1. Phil, Ballard. JavaScript入门经典（第五版）[M]. 北京:人民邮电出版社, 2013.