**DAY01 基本使用**

* 什么是javascript
* 发展历史
* javascript组成 ECMAScript+BOM+DOM
  + BOM的思想（重点）
  + DOM的思想（重点）
* 使用<script></script>标签
* document.write方法
  + 如何输出换行
  + 原样输出标签
* 注释
  + 单行注释 //
  + 多行注释 /\* \*/
* 使用引入js文件的方式
  + <script type="text/javascript" src="demo1.js" />;不可以使用单标签
  + <scriptsrc="demo1.js">alert('xxxx')</script>;不可以在标签中写代码
* <noscript>标签的使用
  + 不支持JS时，显示
* <script>标签的属性
  + **language**已废弃。原来用于代码使用的脚本语言。由于大多数浏览器忽略它，所以不要用了。
  + **src** 表示要引入的外部文件
  + **type** 表示脚本语言的类型
* 原样输出标签的内容
  + &lt; 代表<&gt; 代表>
* 常量（字面量）
* 变量的定义 var （弱类型）
  + 数字类型、字符类型、未定义类型
  + 变量类型在赋值时才能确定
  + 可以一次定义多个变量 var name="yintao", age=18，weight=108;
* 变量命名规则(变量名应做到见名知意)
* alert方法的介绍
* 变量的类型
  + String、Number、Array、Object、
  + Boolean a==b
  + 特殊类型 Null、Undefined
  + 如果alert(abc)一个不存在的变量，会直接报错，而不是undefined，可见undefined和真正的未定义还是有些区别的。因为它是一种特殊类型。
  + null也是一种值，但逻辑含义是没有值，它是对象Object类型
  + typeof关键字
* 变量的计算
  + 加减乘除运算

|  |
| --- |
| **var** a=5+4; **var** b=a\*2; **var** c=b/4; **var** d=b-c;  **var** e=-d;  **var** f=e%4; **var** g=18.4;  **var** h=g%4;  **var** i=3; **var** j=i++;  **var** k=++i; |

* + 字符串和变量的拼接。

|  |
| --- |
| document.write("姓名:"+name+","+"age:"+age); |

* + 所有的语句切记要加；结束，虽然不写语法上正确，但这是非常不好的编程习惯
* 关键字

|  |  |  |  |
| --- | --- | --- | --- |
| Break | else | New | **var** |
| Case | finally | Return | void |
| Catch | for | Switch | while |
| Continue | function | This | with |
| Default | If | Throw |  |
| Delete | In | Try |  |
| Do | instanceof | **Typeof** |  |

* 保留字

|  |  |  |  |
| --- | --- | --- | --- |
| Abstract | enum | Int | short |
| Boolean | export | Interface | static |
| Byte | extends | Long | super |
| Char | final | Native | synchronized |
| Class | float | Package | throws |
| Const | goto | Private | transient |
| Debugger | implements | Protected | volatile |
| Double | import | Public |  |

* 类型的转换

|  |
| --- |
| var a=3; var b="10"; var c=true;  alert(a+b); alert(b+c); alert(a+c); |

* + 例子：计算两个文本框的和
  + 字符串转换数字类型：parseInt()、parseFloat()
  + 四舍五入Math.round(78.266)--->78
* 赋值操作
  + =、+=、-=、\*=、/=、%=
* 关系运算
  + <、>、<=、>=、==、===、!=
* 位运算(扩展)
  + var num= 25 & 3; //1
  + var num= 25 | 3; //27
  + var num = 2<< 3; //16
  + var num = 4^3; //
* 代码规范问题
* 将加法的案例改为 可以做加减乘除、求余五种运算
* 作业
  + 今天课堂所有的例子代码，照敲两遍
  + 为抵抗洪水，战士连续作战89小时，编程计算共多少天零多少小时？
  + 小明要到美国旅游，可是那里的温度是以华氏度为单位记录的。它需要一个程序将华氏温度（80度）转换为摄氏度，并以华氏度和摄氏度为单位分别显示该温度。  
    提示：摄氏度与芈氏度的转换公式为：摄氏度 = 5/9.0\*(华氏度-32)
  + 保留3位小数

|  |
| --- |
| var k=0;  alert(++k + k++ + ++k + k); |

**DAY02 基本使用**

* 逻辑运算
  + && 与、|| 或、! 否
* 自增、自减
  + ++a, a++
  + --a, a--
* 十进制、八进制、十六进制(扩展)
  + var num = 100;
  + var num= 070;
  + var num= 079; //非八进制
  + var num= 0xA;var box = 0x1f;
  + var num= 0.8;
  + var num = .8; //不好的习惯，不推荐此写法，虽然有不少人这样写
  + var num = 12.0; //这样写会自动转为整数12，为了节省内存
  + var num = 4.12e9; //科学计数法
* NaN
  + 当数学计算无法得到数字结果是，该变量的值为NaN

|  |
| --- |
| var num1 = 100-"abc";  alert(num1); |

* + 注意：因为NaN代表非数字，它不等于任何值，也不能做运算，  
    即使alert(NaN == NaN); 结果也是false
  + isNaN(num)函数，该函数判断num变量的值是否是NaN
* Number()函数，可以将任意类型尝试转换为数字

|  |
| --- |
| alert(Number(true)); //1，Boolean 类型的 true 和 false 分别转换成 1 和 0  alert(Number(25)); //25，数值型直接返回  alert(Number(null)); //0，空对象返回 0  alert(Number(undefined)); //NaN，undefined 返回 NaN |
| **如果是对象，则在转换数字失败后，调用toString方法获得返回值**  var box = {  toString : function () {  return '123'; //可以改成 return 'abc'查看效果  }  };  alert(Number(box)); //123 |

* Object类型(简单了解)
  + 创建一个对象，使用new关键字。 var obj = **new**Object();

|  |
| --- |
| var str = new String("hehe");  alert(typeof str); //结果也是object |

* 程序的三大结构
  + 顺序结构
  + 选择结构
  + 循环结构
* if判断语句
  + 注意大括号{} 有和没有的区别
  + else的作用
  + **计算某一年是否是闰年、与或非的运用**
  + else if的作用
  + 成绩判定
* switch语句
  + 案例：成绩判定、显示星期几
  + 注意switch的应用场景
  + 注意case穿透，要加break语句
* 作业：
  + 判断一个整数，属于哪个范围：大于0；小于0；等于0
  + 判断一个整数是偶数还是奇数，并输出判断结果
  + 开发一款软件，根据公式（身高-108）\*2=体重，可以有10斤左右的浮动。来观察测试者体重是否合适
  + 求出1-1/2+1/3-1/4…..1/100的和 var i=1; i\*-1\*-1
  + 打印出1900-2000年中所有的闰年，并以每行四个数的形式输出

**DAY03 语句**

* do-while循环
  + 案例 输出100个数字
  + break终止循环
  + continue跳过某次循环
  + 案例 打印100以内 7的倍数
  + 案例 打印100以内的奇数
  + 案例 打印100以内所有偶数的和
* while循环
  + 打印图形

|  |
| --- |
| \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\* |
| \*  \*\*  \*\*\*  \*\*\*\* \*\*\*\*\* |
| \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\* |

* for循环
  + 一个新入职，月工资为2000元的员工，每年涨工资20%，到退休时的月工资是多少？
  + 打印100–200之间所有能被3或者7整除的数
  + 计算100的阶乘
  + 计算1+3+5+...+99的和
  + 99乘法表
  + 求100-999之间的水仙花数。abc =++
* 死循环
* 作业：
  + 打印出1-100里所有的偶数
  + 打印出1-100里所有的奇数
  + 求出1-1/2+1/3-1/4…..1/100的和 var i=1; i\*-1\*-1
  + 输出20-80之间能被3整除的整数，每5个一行
  + 打印出1000-2000年中所有的闰年，并以每行四个数的形式输出
  + 输出九九乘法表

**DAY04 函数**

* 函数
  + 函数的概念，及作用
  + function关键字
  + 如何执行函数（）

|  |
| --- |
| function m1(){  //xxxxxxxxxxxxx  }  for(var i=0;i<10;i++){  m1();  } |

* + **事件驱动的概念 重要！！！！！**
    - onclick、ondbclick、onfocus、onblur
  + 函数的参数

|  |
| --- |
| function m1(v1, v2, v3...){  //xxxxxxxxxxxxx  } |

* + 年月日分别为自定义函数的参数，判断是否为正确的日期
  + return关键字，带返回值的函数
  + 参数可以传递多个，与函数的定义无关

|  |
| --- |
| function test(paramX,paramY){  }  test("a","b","c"); |

* + 使用arguments对象可以判断参数的个数

|  |
| --- |
| function test(paramX,paramY){  alert(arguments.length);  }  test("a","b","c"); |

* + 案例：要实现一个加法运算，将所有传进来的数字累加，而数字的个数可以任意
  + **作用域的问题**

|  |
| --- |
| var a = 10;  function m1(){  var a = 5;  alert(a);  } |

* + **形参与实参的重要区别**

|  |
| --- |
| var a = 10;  function test(x){  x += 10;  }  alert(a); |

* + **递归的用法**
* with语句
  + 可以省略对象名前缀

|  |
| --- |
| with(document){  write("abc");  getElementById("div\_bar");  } |

**DAY05 数组**

* for-in（结合数组或对象）
  + 数组的概念，一组相同类型的数据
  + 下标（索引）的概念
  + 创建数组

|  |
| --- |
| var arr= newArray(10); //创建一个包含 10 个元素的数组  var arr2= new Array('尹涛',30 ,'教师','安阳');//使用构造函数创建  var arr3 = [1,2,3,4,5]; |

* + 静态赋值/动态赋值

|  |
| --- |
| arr[0] = 5;  arr[1] = 12;  arr[2] = 4; |
| for(var i=0; i<5; i++){  arr[i] = Math.random();  } |

* + 数组的length属性 ，代表长度
  + **遍历访问数组 普通for循环和 for-each循环**
  + push/pop、unshift/shift方法

|  |
| --- |
| arr.push('BJ'); //数组末尾添加一个元素  arr.unshift('TJ','HN'); //数组开头添加两个元素  var str = arr.pop(); //移除末尾一个元素，并返回该元素  var str = arr.shift(); //移除开头一个元素，并返回该元素 |
| 数据结构： 栈结构、队列结构 |

* + 练习：找出所有1-100之间7的倍数和包含7的数字
  + 翻转reverse方法
  + 从小到大排序sort方法 (数字的排序，对象的排序)
  + 冒泡排序算法
  + 选择排序算法
  + 随机点名程序
* 作业
  + 冒泡排序(13K)
  + 随机生成一个五位以内的数，然后输出该数共有多少位，每位分别是什么
  + 输出斐波那契额数列数列的前20项(递归)(11K)
  + 山上有一口缸可以装50升水，现在有15升水。老和尚叫小和尚下山挑水，每次可以挑5升。问：小和尚要挑几次水才可以把水缸挑满？通过编程解决这个问题。
  + 有一个**从小到大排好序**的数组。现输入一个数，要求按原来的规律将它插入数组中。(11K)
  + 开发一个标题为“FlipFlop”的游戏应用程序。它从1计数到100，遇到3的倍数就替换为单词“Flip”，5的倍数就替换为单词“Flop”，既为3的倍数又为5的倍数则替换为单词“FlipFlop”。(8K)
  + 生成13位条形码(6K)  
    Ean-13码规则：第十三位数字是前十二位数字经过计算得到的校验码。  
    例如：690123456789  
    第shi三位计算其校验码的过程为：  
    @前十二位的奇数位和6+0+2+4+6+8=26  
    @前十二位的偶数位和9+1+3+5+7+9=34  
    @将奇数和与偶数和的三倍相加26+34\*3=128  
    @取结果的个位数：128的个位数为8  
    @用10减去这个个位数10-8=2  
    所以校验码为2（注：如果取结果的个位数为0，那么校验码不是（10-0=10），而是0）实现方法ean13（）计算验证码，输入12位条码，返回带验证码的条码。  
    例如：输入：692223361219输出：6922233612192
  + 自定义一个对象，该对象存储一个学生的信息，该对象包含学号、身份证、年龄、性别、所学专业等属性信息，同时该对象包含一个自我介绍的方法，用来输出该对象的所有信息。(8K)

**DAY06 String对象**

* 字符串的定义var str='fsdsdfsdf'；
  + 也可使用 var str = new String("abc"); 注意此刻str为object类型
* 字符串的属性length
* 字符串的获取方法
  + charAt(3) //获取下标为3的字符
  + charCodeAt(3) //获取下标为3的字符的编码
  + fromCharCode(94) //编码转换成字符
* 字符串的查找方法
  + indexOf("abc") 查找字符串第一次出现的位置
  + lastIndexOf("abc") 查找字符串最后一次出现的位置 如果没找到 返回-1
  + search() 正则匹配 (返回出现的位置)

|  |
| --- |
| **var** str="Visit W3School!"  document.write(str.search(/w3school/i)) |

* + matchstringObj.match(rgExp)
  + replace替换字符串 stringObj.replace("tmd", "\*");
  + 这里的替换只能执行一次，不能够进行全局匹配，如果需要全局匹配，则应使用正则表达式

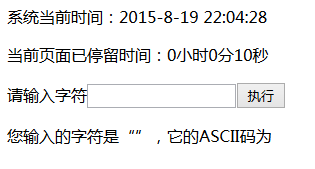
|  |
| --- |
| str.replace(/tmd/gi,"\*"); |
| g表示进行全局匹配，i表示匹配的时候忽略大小写 |

* 字符串比较方法
  + ==和===
  + str=new string('abc') 与var str='abc';
* 字符串的截取方法
  + substring(3)
* 字符串分割
  + split——根据分隔符、拆分成数组
  + 案例：内容查找替换功能
  + toLowerCase、toUpperCase
* 任意给定的一串字母，统计字符串里面的大写字母和小写字母的个数（6K）
  + ascii码 的由来
  + GBK及Unicode由来
* Math对象
  + Math.round(3.6) //四舍五入
  + random() //返回0-1之间的随机数
  + max(num1, num2) //返回较大的数
  + min(num1, num2) //返回较小的数
  + abs(num) //绝对值
  + ceil(19.3) //12 向上取整
  + floor(11.8) //11 向下取整
  + pow(x,y) //x的y次方
  + sqrt(num) //开平方
  + 案例：随机验证码
* 案例库：简易计算器、查找替换、获取验证码
* 作业
  + 课堂案例，照敲两遍以上
  + 注册页面的简单数据校验，（12K）
    - 名字只能包含数字、字母和下划线，数字不可以开头，长度不低于6，不高于20
    - 密码必须一致
    - 输入框中要有默认提示
    - 昵称一栏要将敏感词或特殊符号自动过滤，将输入的非法字符自动清除（**~~onkeyPress\onkeyDown~~**\onkeyUp）
    - 所有输入框不能为空，并数据合法，点击提交才可以正确提交表单数据
  + 已知字符串 “a,a,b,c,c,d” 编程实现以下功能。(13K)
    - 统计每个字符出现的次数，结果显示 a 2、b 1、c 2、d1
    - 去掉重复的字符，使结果显示 abcd
  + 案例库：计算器（11K）

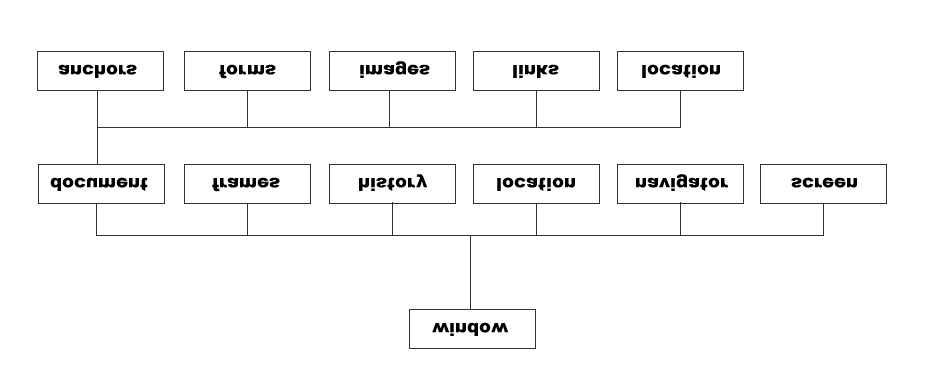
**DAY07 Date对象**

* 了解Date对象的作用
  + 代表日期
* 如何创建对象
  + var d = new Date();
  + var d = new Date("2015/08/22");
* 获取年月日
  + getFullYear()/setFullYear(2014)
  + getMonth()/setMonth(8)
    - 注意：获取月份是从0开始的
  + getDate()/setDate(25)
* 获取星期几
  + [getDay()](mk:@MSITStore:D:\学习资料\千锋\尹涛备课资料\W3School离线手册(2014.3.24).chm::/www.w3school.com.cn/jsref/jsref_getDay.asp) 0-6
* 获取时分秒
  + getHours()
  + getMinutes()
  + getSeconds()
  + 案例：显示当前时间\停留时间
* 日期处理
  + Date.parse("2015-08-24")
    - 转换格式默认支持2015-08-24或2015/08/24
    - 如果需要支持其他格式，只能手动处理
  + new Date(time)
    - 创建一个日期对象，并指定时间 可以指定毫秒数
    - 或者修改time属性， var d = new Date(); d.setTime(56521211021);
  + getTime()/setTime()
    - getTime获取某个日期自1970年以来的毫秒数
    - setTime修改日期的毫秒数，对应的日期时间也会修改
* 封装日期处理的函数库(重要！)
  + 判断某年份是否为闰年
  + 将日期格式化输出 “2015-08-24”
  + 获得某个月份的天数
  + 将字符串转换为日期
    - join（sep）方法的介绍，将数组按照指定的分隔符转为字符串
    - 例子：

|  |
| --- |
| var arr = ['y','i','n'];  var dstr = arr.join("-");  alert(dstr);//结果 y-i-n |

* + 判断两个日期相差的天数
  + 获得N天以后的日期
* 作业
  + 页面上完成这几项功能

**DAY08 BOM**(Browser Object Model)

* BOM的概念
  + 
* 与document的关系
* location对象
  + href属性 控制浏览器地址栏的内容
  + reload()方法 刷新页面
  + reload(true) 刷新页面，不使用缓存
  + toSource()方法 获取某个对象的源码
* navigator对象
  + appName 浏览器名称
  + appVersion 浏览器版本
  + platform操作系统
  + 最新的浏览器已经全面放弃以上这些属性
  + userAgent 用户代理信息，通过该属性可以获取浏览器及操作系统信息
* close() 关闭浏览器 FF不支持
* closed属性
  + 表示窗口是否要关闭
* defaultStatus 窗口状态栏默认信息
* status属性 状态栏信息
* alert(text) 弹出窗
* blur() 窗口失去焦点
* confirm()方法
  + confirm("对话框的提示文字")
  + 该方法有返回值，点击确定返回true，按取消返回false
* 输入框：prompt()，返回字符串或null

|  |
| --- |
| var name= window.prompt("请输入姓名");  alert(name); |

* open()方法: 弹出一个新窗口
  + 第二个参数：窗口命名

|  |
| --- |
| open('http://www.baidu.com','baidu','width=400,height=400,top=200,left=200,toolbar=yes');  如果两次弹出窗口的命名一样，将不会打开新弹窗，而再之前的弹窗中加载新页面 |

* + 子窗口操作父窗口

|  |
| --- |
| document.onclick = function() {  opener.document.title='子窗口让我输出的！';  } |

* showModalDialog()

|  |
| --- |
| **parent.htm** <script>     var obj = new Object();     obj.name="51js";     showModalDialog("modal.htm",obj,"dialogWidth=200px;dialogHeight=100px"); </script> **modal.htm** <script>     var obj = window.dialogArguments     alert("您传递的参数为：" + obj.name) </script> |

* setInterval() 定时执行任务 (每隔指定的时间，执行一次指定的函数)

|  |
| --- |
| setInterval(fun1,500);//每隔500毫秒执行一次fun1函数 |

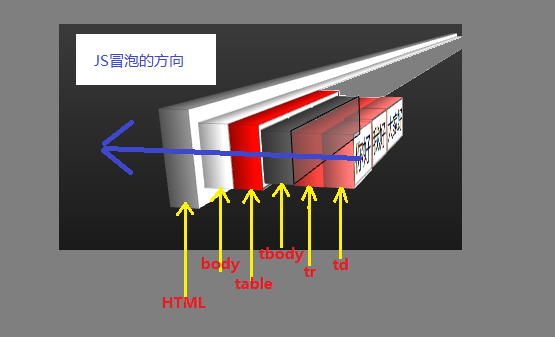
* + 显示日期时间
  + 案例：进度条\广告弹出窗
  + clearInterval(interval) 清除定时器
* setTimeout定时执行任务 （指定的时间以后，执行一次函数）
* 三目运算符
  + 逻辑表达式 ? 值1 : 值2;
  + 整个运算的最终结果是： 如果逻辑表达式的值为真，则最终结果为值1，否则为值2；
* 常见事件
  + onload 加载事件网页加载完毕后执行
  + **onscroll 滚动事件**
    - 案例：回到顶部
    - document.documentElement.scrollTop 代表垂直的滚动条，向下滚动的距离
    - document.body.scrollTop //IE 代表垂直的滚动条，向下滚动的距离
* 作业
  + 字母验证码
  + 进度条
  + 广告弹出窗
  + 回到顶部

**DAY09DOM(上)**

* DOM（文档对象模型）的概念
* 节点树种节点之间的关系，父子、兄弟
* 节点种类：元素节点、文本节点、属性节点。
* getElementById() 获取特定 ID 元素的节点
  + window.onload = function () { //预加载 html 后执行
* getElementsByTagName() 获取相同元素的节点列表
  + 返回数组，使用[0]来获取
* getElementsByName() 获取相同名称的节点列表
  + 不是所有标签都有name属性
  + 某些低版本浏览器会有兼容性问题
* getElementsByClassName() 获取相同class属性的节点列表
  + IE8及以下不支持
* getAttribute() 获取特定元素节点属性的值
  + 获取自定义属性时，**oDiv.xxx** ， IE最新版本不支持，其他浏览器不支持，因此要采用这个方法来访问
* setAttribute() 设置特定元素节点属性的值
  + IE低版本浏览器不支持这个方法
* removeAttribute() 移除特定元素节点属性
  + 某些低版本浏览器不支持
* 访问元素节点的属性
  + tagName 获取元素节点的标签名
  + innerHTML 获取元素节点里的内容
  + outerHTML/innerText，非 W3C DOM 规范
  + id 元素节点的 id 名称
  + title 元素节点的 title 属性值（鼠标悬停时的提示）
  + style CSS内联样式属性值
  + 获取外部样式的属性，必须要使用currentStyle（IE特有）

|  |
| --- |
| //var style = window.getComputedStyle("元素", "伪类");  var dom = document.getElementById("test");  style = window.getComputedStyle(dom , ":after");  //个别浏览器对第二个伪类不支持 |

* + **className**CSS元素的类 (不可以使用class)
* JS冒泡



* 案例：《JS简易年历》、《QQ延迟提示框》、《换肤》
* 作业：
  + **表格的即时编辑**
  + **输入框默认提示**

**DAY10 DOM(下)**

* node 节点属性
  + 节点可以分为元素节点、属性节点和文本节点，而这些节点又有三个非常有用的属性 ，分别为：nodeName、nodeType 和 nodeValue。
  + 节点属性nodeName、 nodeType、 nodeValue
  + 元素1 属性 2 文本 3
    - 在IE9之前， nodeType 的表示范围不符合标准
  + 节点的属性列表：
  + **childNodes** 获取当前元素节点的所有子节点
    - 这里面包含空白节点，在IE9之前，IE浏览器会自动忽略空白节点
  + **firstChild** 获取当前元素节点的第一个子节点
  + **lastChild** 获取当前元素节点的最后一个子节点
  + **ownerDocument** 获取该节点的文档根节点，相当与 document
  + **parentNode** 获取当前节点的父节点
  + **previousSibling** 获取当前节点的前一个同级节点
  + **nextSibling** 获取当前节点的后一个同级节点
  + **attributes** 获取当前元素节点的所有属性节点集合
* childNodes 属性
  + 返回元素节点和文本节点两种
  + 使用nodeName来获取元素的标签名称
  + 使用nodeValue来获取文本内容
  + 使用nodeType来判断到底是哪个节点（元素1 属性 2 文本 3）
* innerHTML 和 nodeValue
  + box.childNodes[0].nodeValue = '<strong>abc</strong>';
  + box.innerHTML = '<strong>abc</strong>';
  + 使用nodeValue，标签会被转义，然后在页面上原样输出
* attributes 属性
  + document.getElementById('box').attributes //NamedNodeMap
  + document.getElementById('box').attributes.length;//返回属性节点个数
  + document.getElementById('box').attributes[0]; //返回第一个属性节点
  + document.getElementById('box').attributes[0].nodeType; //2，属性
  + document.getElementById('box').attributes[0].nodeValue; //属性值
  + document.getElementById('box').attributes['id']; //返回属性为 id 的节点
  + document.getElementById('box').attributes.getNamedItem('id'); //
* 忽略空白文本节点
  + body.childNodes.length
  + 在低版本IE中，会自动忽略掉空白的文本节点，其他浏览器不会忽略
  + 可以编写一个手工过滤空白节点的方法
  + 或者编写一个删除空白节点的方法
  + 匹配空白节点可以使用正则：/^s+$/.test(box.nodeValue)
* 节点操作
  + DOM 不单单可以查找节点，也可以创建节点、复制节点、插入节点、删除节点和替换节点
  + write()
  + createElement() //创建一个元素节点
  + createTextNode() //创建一个文本节点
  + box.appendChild(node) //
  + box.insertBefore(newNode, existNode) //
  + box.removeChild(node)
* offsetParent（参照物父元素）
  + 当某个元素的父元素或以上元素都未进行CSS定位时，则返回body元素，也就是说元素的偏移量（offsetTop、offsetLeft）等属性是以body为参照物的
  + 当某个元素的父元素进行了CSS定位时（absolute或者relative），则返回父元素，也就是说元素的偏移量是以父元素为参照物的
  + 当某个元素及其父元素都未进行CSS定位时，则返回距离最近的使用了CSS定位的元素。
* 案例：点击单元格换图片
* 封装getElementByName方法，获取单个元素
* 作业
  + 表格的自动添加、删除、即时编辑
  + 隔行变色，鼠标划入变色

**DAY11 事件**

* 了解什么是event对象
  + 它是事件绑定中的一个隐藏的参数，可以通过arguments[0]来获取

|  |
| --- |
| btn.onclick = function () {  alert(arguments[0]); //Event，事件对象  }; |

* + 也可以叫事件源对象，这里面包含了关于这次单击事件的相关信息

|  |
| --- |
| alert(event.button);  //如果当前event是鼠标事件，则会有一个button属性，它是一个数字  0代表鼠标按下了左键 || 1代表按下了滚轮 || 2代表按下了右键 |

* + 不过老版本的**IE**并没有遵守W3C的规范，它的button属性含义如下

|  |
| --- |
| 1鼠标左键 2鼠标右键 3左右同时按 4滚轮 5左键加滚轮 6右键加滚轮 7三个同时 |

* + 目前IE11.0版本，无兼容性问题
* 掌握event对象的获取
  + 在W3C标准中，直接在函数中声明该参数即可

|  |
| --- |
| btn.onclick = function(event) {  //这里就可以使用了  }; |

* + 兼容性写法，支持老版本的IE

|  |
| --- |
| var evt = event ? event : window.event;  var evt = event || window.event; |

* 掌握事件冒泡的概念，如何阻止事件冒泡
  + 事件从子节点向上依次传递给父节点
  + 阻止事件冒泡的方法

|  |
| --- |
| event.stopPropagation();  event.cancelBubble=true;//IE的方式 |

* 鼠标事件中获取鼠标的位置

|  |
| --- |
| event.offsetX; //相对于事件源对象的偏移量,也就是元素坐标，相对坐标  event.clientX; //在可视区域的位置，也就是浏览器坐标  event.screenX; //整个屏幕坐标 |

* 获取鼠标在页面的绝对位置

|  |
| --- |
| event.clientX + document.documentElement.scrollLeft || document.body.scrollLeft;  event.clientY+ document.documentElement.scrollTop || document.body.scrollTop; |

* 案例：**显示光标位置/ 鼠标跟随效果 / 贴吧**
* 键盘事件获取用户按键方法 keyCode
  + 在keydown事件中，我们可以通过键码来判断用户按下了哪个键

|  |
| --- |
| event.keyCode || event.which  alert(evt.which); //老IE不支持 |

* + 使用编码值还原字符

|  |
| --- |
| String.fromCharCode(code) |

* 键盘的其他属性 ctrlKey\shiftKey\altKey
  + 判断ctrl、shift、alt键是否被按下

|  |
| --- |
| if(code == 13 &&event.ctrlKey) {  //发送消息  } |

* + 在输入框中使用ctrl+enter键模拟消息的发送
  + 在textarea中，获取或者更改标签内容，应使用value属性，除IE浏览器外，当对textarea标签的value赋值以后，会导致innerHTML失效.
* 作业：
  + 模拟select下拉菜单
  + 鼠标跟随一串效果

=====================================================

**DAY12拖拽效果**

* 阻止右键菜单

|  |
| --- |
| document.oncontextmenu = function(){  alert("右键被按下");  return false;  } |

* + 自定义右键菜单
* 阻止超链接的默认行为

|  |
| --- |
| evt.preventDefault(); |

* 拖拽的原理讲解
* 三个事件 (onmousedown\onmousemove\onmouseup)
  + 拖拽思路

|  |
| --- |
| 1. 给目标元素添加**onmousedown**事件，拖拽的前提是在目标元素按下鼠标左键 2. 当**onmousedown**发生以后，此刻给**document**添加**onmousemove**事件，意味着此刻鼠标在网页的移动都将改变目标元素的位置 3. 在**onmousemove**事件中，设定目标元素的left和top， 公式 目标元素的left = 鼠标的clientX – （鼠标和元素的横坐标差，即offsetX） 目标元素的top = 鼠标的clientY– （鼠标和元素的纵坐标差，即offsetY） 4. 当**onmousedown**发生以后，此刻给**document**添加**onmouseup**事件，意味着此刻鼠标在网页的任意位置松开鼠标，都会放弃拖拽的效果 5. 在**onmouseup**事件中，取消**document**的**onmousemove**事件即可。 |

* 事件监听器target.addEventListener("click", fun, false);
  + 可以绑定多个函数在一个对象上
  + 执行顺序按照绑定的顺序来
  + 第三个参数是否使用捕捉(反向冒泡)，默认false，为冒泡
* IE的事件监听器
  + target.attachEvent("onclick",fun);
  + 执行顺序按照绑定的反序
* 编写函数封装，解决addEventListener和attachEvent

=====================================================

**DAY13Cookie存储（会话跟踪技术）**

* Http是一个无状态的协议
* 如果浏览器完全禁止cookie功能的话，大多数网站的基本功能无法正常使用
* cookie的由来
  + 服务器端的会话技术(session)
    - 咖啡厅会员卡的例子
  + 保存会话信息
  + 文本形式存在
    - 文件存在的地方

|  |  |
| --- | --- |
| execl表格和会员卡都是会话跟踪技术，都是用来记录保存客户和咖啡厅之间交互来往时需要用到的一些信息。 | session和cookie都是会话跟踪技术，都是用来记录保存客户端跟服务器之间通信时需要用到的一些数据。 |
| excel表格存放在咖啡厅电脑上 | session对象存放在服务器内存里 |
| 会员卡存放在用户手里 | cookie存放在客户端本地硬盘上 |
| 用户每次去咖啡厅，都会携带该咖啡厅发放的会员卡 | 客户端每次访问服务器都会携带该服务器发送到本地的cookie文件 |
| 当使用excel表格保存用户信息时，会员卡上不再需要做大量记录。只需要记录一个会员编号 | 当使用session保存客户端信息时，cookie文件不再需要保存大量客户端信息，只需要简单记录一个sesssion的ID号 |
| 当会员卡丢失，咖啡厅便无法辨别你的身份，优惠政策你便不能享用了，你消费记录只能从零开始 | 当cookie文件丢失，服务器便无法辨别客户端的身份，只能假定客户端是第一次访问服务器，所以会要求客户端进行登陆操作。 |
| 咖啡厅规定，当你1年不再光顾咖啡厅，你的消费记录有可能会被清空，下次再来，会员卡也就作废了。只能重新办理会员卡，并重新为你建立excel表格。 | 服务器默认设定，当客户端30分不再与服务器产生交互，便认为会话过期，session对象会被销毁。此时再请求服务器，cookie中记录的sessionID号已经毫无意义。 session对象只能重新建立，用户也必须重新进行登陆。 |
| 当你拒绝使用咖啡厅发放的会员卡，咖啡厅便无法确认你的会员身份，你就不能享受优惠 | 当浏览器禁用掉cookie功能，服务器便彻底无法辨别客户端的身份，session功能就会失效，你就无法正常的访问网站。 |

* 添加一个cookie

|  |
| --- |
| document.cookie = "username=123"; //一次写入一个键值对  一次写入多个，只有第一个生效，或者采用&符号链接  user=尹涛&password=123&name=xxxx; |

* + 保存形式：key=value
  + 取出显示cookie的类型，取出cookie的值

|  |
| --- |
| alert(typeof document.cookie);  alert(document.cookie); |

* + 中文编码问题，使用encodeURIComponent('xxxxx')编码，再使用decodeURIComponent(document.cookie)解码，能解决中文乱码问题
* cookie失效时间
  + 如果不加这个参数，默认情况下为临时cookie，也就是关闭浏览器则cookie失效
  + expires=date，指定在是时间，cookie被自动清理
  + 如果手动调整系统的时间，让cookie消失，可以进行恢复，此时为假删除
  + 当覆盖某个键值对，并设置失效时间过期时，则cookie被删除，不能恢复
* cookie的完整存储形式
  + 键=值;expires=失效时间;domain=域名访问;
* cookie的路径问题
  + 在同一路径下的网页可以共享cookie，路径不同时，不能访问
* cookie的域名问题
  + 必须在绑定域名的服务器上才可以设置域名
  + 并且只能设置绑定的域名，也就是说，不同服务器间的cookie文件不共享
* cookie的限制
  + Chrome和Safari没有对cookie的个数做限制，一般浏览器限制同一域名数量为50个
  + cookie文件的总大小一般为4KB（同一个域名）
* 封装cookie的增删改查
  + 添加一个cookie
  + 获取一个cookie
  + 删除cookie
* 案例：
  + 拖拽的效果保存
  + 用户密码的保存
  + 购物车

**DAY14正则表达式**

* 什么是正则表达式
  + 正则表达式(regular expression)是一个描述字符规则的对象。
* 为什么要使用正则表达式
  + 前端往往有大量的表单数据校验的工作，采用正则表达式会使得数据校验的工作量大大减轻
* 如何创建正则表达式
  + 使用RegExp这个对象，第一个参数就是我们的模式字符串

|  |
| --- |
| var reg= new RegExp('study'); |

* + 第二个参数可选，模式修饰符

|  |
| --- |
| var reg = new RegExp('study', 'ig'); |

* + i 表示忽略大小写
  + g 表示全局匹配
* 还可以用常量方式直接声明
  + - var reg = /study/;
* 测试正则表达式 test方法
  + reg.test("Study") false
  + reg.test("study") true
  + 除了匹配相等以外，还可以匹配更复杂的规则
  + “good good study，day day up”判断是否包含符合该正则的内容
* 测试正则表达式exec方法
  + reg.exec(str) 如果没有匹配到，则返回null
* 使用字符串的正则表达式方法
  + 使用 match 方法获取获取匹配数组

|  |
| --- |
| var pattern = /good/ig; //全局搜索  var str = 'good good study!，day day up!';  alert(str.match(pattern)); //匹配到两个 good,good  alert(str.match(pattern).length); //获取数组的长度 |

* + 使用 search 来查找匹配数据

|  |
| --- |
| var pattern = /good/ig;  var str = 'good good study!，day day up!';  alert(str.search(pattern)); //查找到返回位置，否则返回-1 |

* + 使用 replace 替换匹配到的数据

|  |
| --- |
| var pattern = /Good/ig;  var str = 'good good study!，day day up!';  alert(str.replace(pattern, 'hard')); //将Good替换成了hard |

* + 使用 split 拆分成字符串数组

|  |
| --- |
| var pattern = / /ig;  var str = 'good good study!，day day up!';  alert(str.split(pattern)); //将空格拆开分组成数组 |

* 元字符（详情参见正则手册）
  + 使用点元字符

|  |
| --- |
| var pattern = /g..gle/; //.匹配一个任意字符  var str = 'google';  alert(pattern.test(str)); |

* + 重复多次匹配

|  |
| --- |
| var pattern = /g.\*gle/; //.匹配 0 到多个  var str = 'google'; //\*,?,+,{n,m}  alert(pattern.test(str)); |

* + 使用字符类匹配

|  |
| --- |
| var pattern = /g[a-zA-Z\_]\*gle/; //[a-z]\*表示任意个 a-z 中的字符  var str = 'google';  alert(pattern.test(str));  var pattern = /g[^0-9]\*gle/; //[^0-9]\*表示任意个非 0-9 的字符  var str = 'google';  alert(pattern.test(str));  var pattern = /[a-z][A-Z]+/; //[A-Z]+表示 A-Z 一次或多次  var str = 'gOOGLE';  alert(pattern.test(str)); |

* + 使用元符号匹配

|  |
| --- |
| var pattern = /g\w\*gle/; //\w\*匹配任意多个所有字母数字\_  var str = 'google';  alert(pattern.test(str));  var pattern = /google\d\*/; //\d\*匹配任意多个数字  var str = 'google444';  alert(pattern.test(str));  var pattern = /\D{7,}/; //\D{7,}匹配至少 7 个非数字  var str = 'google8';  alert(pattern.test(str)); |

* + 使用锚元字符匹配

|  |
| --- |
| var pattern = /^google$/; //^从开头匹配，$从结尾开始匹配  var str = 'google';  alert(pattern.test(str));  var pattern = /goo\sgle/; //\s 可以匹配到空格  var str = 'goo gle';  alert(pattern.test(str));  var pattern = /google\b/; //\b 可以匹配是否到了边界  var str = 'google';  var str2= 'googleaa googlexx google dsdddd';  alert(pattern.test(str)); |

* + 使用或模式匹配

|  |
| --- |
| var pattern = /google|baidu|bing/; //匹配三种其中一种字符串  var str = 'google';  alert(pattern.test(str)); |

* + 使用分组模式匹配

|  |
| --- |
| var pattern = /(google){4,8}/; //匹配分组里的字符串 4-8 次  var str = 'googlegoogle';  alert(pattern.test(str));  var pattern = /8(.\*)8/; //获取 8..8 之间的任意字符  var str = 'This is 8google8, dd 8ggg8';  alert(str.match(pattern));  alert($1); //得到第一个分组里的字符串内容  var pattern = /8(.\*)8/;  var str = 'This is 8google8';  var result = str.replace(pattern,'<strong>$1</strong>'); //得到替换的字符串输出  document.write(result);  var pattern = /(.\*)\s(.\*)/;  var str = 'google baidu';  var result = str.replace(pattern, '$2 $1'); //将两个分组的值替换输出  document.write(result); |

* 使用 exec 返回数组

|  |
| --- |
| var pattern = /^[a-z]+\s[0-9]{4}$/i;  var str = 'google 2012';  alert(pattern.exec(str)); //返回整个字符串    var pattern = /^[a-z]+/i; //只匹配字母  var str = 'google 2012';  alert(pattern.exec(str)); //返回 google  var pattern = /^([a-z]+)\s([0-9]{4})$/i; //使用分组  var str = 'google 2012';  alert(pattern.exec(str)[0]); //google 2012  alert(pattern.exec(str)[1]); //google  alert(pattern.exec(str)[2]); //2012 |

* + 捕获性分组和非捕获性分组

|  |
| --- |
| var pattern = /(\d+)([a-z])/; //捕获性分组  var str = '123abc';  alert(pattern.exec(str));  var pattern = /(\d+)(?:[a-z])/; //非捕获性分组  var str = '123abc';  alert(pattern.exec(str)); |

* + 使用分组嵌套

|  |
| --- |
| var pattern = /(A?(B?(C?)))/; //从外往内获取  var str = 'ABC';  alert(pattern.exec(str)); |

* + 使用特殊字符匹配

|  |
| --- |
| var pattern = /\.\[\/b\]/; //特殊字符，用\符号转义即可  var str = '.[/b]';  alert(pattern.test(str)); |

* + 使用换行模式

|  |
| --- |
| var pattern = /^\d+/mg; //启用了换行模式  var str = '1.baidu\n2.google\n3.bing';  var result = str.replace(pattern, '#');  alert(result); |

* 常见正则
  + 检查邮政编码//共 6 位数字，第一位不能为 0
    - /^[1-9]\d{5}$/
  + 检查文件压缩包 //xxx.zip\xxx.gz\xxx.rar
    - /^\w+\.(zip|gz|rar)$/ ~~war3.zipgz~~
  + 删除多余空格 //
    - str.replace(/\s+/,'');
  + 删除首尾空格
    - str.replace(/^\s+/,'');
    - str.replace(/\s+$/,'');s
  + 电子邮件( xxxxx @ xxxx(.xxxx)+)
    - /^\w+@\w+(\.\w+)+$/
  + 手机号
    - /^[1-3]\d{10}$/
  + 身份证
    - /^\d{17}(\d|X)$/
  + 日期 (xxxx-xx-xx| xxxx/xx/xx | xxxx.xx.xx)
    - /^\d{2}|\d{4}[-\/\.]\d{2}[-\/\.]\d{2}$/
  + 只能输入中文
    - str.replace(/[^\u4e00-\u9fa5]/g,'');
  + 账户名只能使用数字字母下划线，且数字不能开头，长度在6-15之间
    - /^[a-zA-Z\_]\w{5,14}$/
  + 验证IP
    - (xxx.)xxx.xxx.xxx|
    - 254.245.255.255
    - 240.196.19.5
    - /^((25[0-5]|2[0-4]\d|1\d{2}|[1-9]?\d)\.){3}(25[0-5]|2[0-4]\d|1\d{2}|[1-9]?\d)$/
  + 将所有的bag、beg、big和bog改为bug
    - str.replace(/(bag|beg|big|bog)/g,'bug');
  + 将所有方法foo(a,b,c)的实例改为foo(b,a,c)
    - str.replace(/foo\(([^,]+),([^,]+),([^,]+)\)/g,'foo($2,$1,$3)');
  + 假设有一个多字符的片断重复出现，例如：

|  |
| --- |
| Billy tried really hard  Sally tried really really hard  Timmy tried really really really hard  Johnny tried really really really really hard  而你想把"really"、"really really"，以及任意数量连续出现的"really"字符串换成一个简单的"very" |

* + - str.replace(/(really\s)+/mgi,'very');

**DAY15 表单综合案例**

**DAY16Ajax（上）**

* 什么是Ajax
  + **A**synchronous **J**avascript **A**nd **X**ml
  + 什么是异步？
    - 同时执行，也叫并发
  + 什么是同步？
    - 按步骤顺序执行
* 进程的概念
  + 进程**≠**程序
    - 程序从开始到结束的一次执行过程叫做进程
    - 一个进程当中，程序同时运行的多个分支，叫做线程
  + 多线程异步执行，可以提高程序的效率
* ajax的起源
  + google的下拉提示
* 局部刷新的应用场景
  + 分页、用户名及时验证、聊天
* 请求GET/POST请求数据
  + GET和POST请求的区别
* AJAX编写步骤
  + 创建请求对象

|  |
| --- |
| var request = new XMLHttpRequest(); |

* + 设置请求参数

|  |
| --- |
| request.open("get", "http://10.0.152.17/AJAX/ajaxtest", true); |

* + 设置回调函数

|  |
| --- |
| request.onreadystatechange = function(){  if(request.readyState == 4) {  alert(request.responseText);  }  } |

* + 发送请求

|  |
| --- |
| request.send(); |

* 浏览器兼容问题
  + IE

|  |
| --- |
| **new** ActiveXObject("Msxml2.XMLHTTP") |

* + 非IE

|  |
| --- |
| **new**XMLHttpRequest(); |

* JSON数据的应用
  + webservice技术的介绍

|  |
| --- |
| <person>  <id>4564523626256562</id>  <name>张三</name>  </person>  {  id : 21465461461461,  name: "张三"  } |

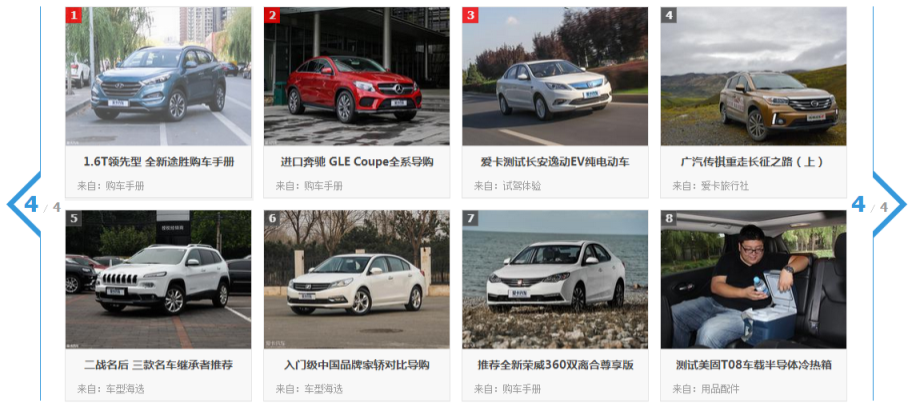
* eval方法的使用

|  |
| --- |
| //eval("("+json字符串+")");  它的作用是，将一个普通的json格式的字符串，转换成Json格式的对象  举例：  var list = eval("("+request.responseText+")"); |

**DAY17 Ajax（下）**

* 动态加载数据
  + 加载更多按钮（分页）
  + 多人聊天
  + 瀑布流

**DAY18 运动（上）**

* 更改元素的坐标left、top
* 使用定时器持续更改坐标
* 运动边界的处理
* 方向的改变
* 摩擦力的增加
* 五颗星评分
* 选项卡切换
* 导航下拉菜单
* 点击表格图片切换
* 数码时钟
* 作业：
  + 亚马逊导航菜单
  + <http://www.xcar.com.cn/> 主页面左右分页
  + 播放列表收缩展开
  + 竖排导航

**DAY19 运动（下）**

* 播放列表收缩展开
* 竖排导航
* 更多产品
* 顶部悬浮
* 展开菜单
* 透明度过渡函数

**DAY20 运动(综合案例)**

* 图片轮播
* 透明度切换
* 分享到

**DAY21闭包及面向对象**

* 什么是匿名函数
  + 自运行
  + 赋给变量
* 什么是闭包：闭包是指在函数外部访问函数作用域中变量的函数
  + 局部变量的累加
  + 没有自我执行的匿名函数保存循环变量
  + 匿名函数自运行后，保存循环变量
  + 增加语句块作用域
    - 循环嵌套时，打印99乘法表
* 作业：
  + 瀑布流、图片轮播、顶部悬浮、回到顶部、更多产品、亚马逊菜单、仿select下拉菜单

**DAY22 构造函数**

* 使用new Object()创建对象
* 引用类型的赋值person2 = person1
* 使用函数 createObject 来代替重复的对象创建
  + 减少重复代码
  + 但无法区分所属类型（角色）
* 使用构造函数
  + 构造函数的调用 new关键字
  + 普通方式调用Person()
* 对象属性的比较
* 对象函数的比较
  + new Function()
  + 使用外部函数调用
  + this关键字的指向
* prototype原型
  + 所有通过同一个构造函数创建的实例对象，都会共享同一个prototype
  + Person.prototype访问
  + person1.prototype访问
  + person1.\_\_proto\_\_访问
  + 默认自动生成
  + 用于共享属性方法
  + Person.prototype.isPrototypeOf检测 该原型是否属于某对象
  + delete person.name删除
  + person. hasOwnProperty('name') 是否包含该实例属性
  + 'name' in person 是否存在该属性（原型或实例中）
* 继承的三种方式
  + prototype
  + call
  + apply

**DAY23设计模式**

* 单例模式的概念
  + 每个类只允许创建一个实例对象
* 单例模式的应用
  + 在多个作用域中，使用同一个非全局的对象
* 单例模式的编写
  + 使用私有变量instance保存单例
  + 在使用时初始化，减少资源浪费
  + 返回闭包访问单例
* 方式1
  + 私有变量

|  |
| --- |
| var Singleton = (function(){  function Person(args){  //.......  }  var instance = null;  var staticObj = {  getInstance : function(args){  if(instance == null) {  instance = new Person(args);  }  return instance;  }  }  return staticObj;  })(); |

* 方式2
  + prototype

|  |
| --- |
| var Singleton = function(){  if(!this.instance) {  Singleton.prototype.instance = this;  }  //属性........  //方法........  return this.instance;  } |

* 方式3
  + 静态变量

|  |
| --- |
| var MyObject = function(){  if(!MyObject.instance) { //静态变量  MyObject.instance = {  prop:"abc",  fun:function(){  }  };  }  return MyObject.instance;  } |

* 工厂模式的概念
  + 利用工厂生产对象
  + 封装对象的创建过程，适用于创建过程比较繁琐的对象
* 实现代码

|  |
| --- |
| var DomFactory = (function(){  this.Link = function(){  this.appendTo = function(target){  var ele = document.createElement("link");  ele.url = this.url;  target.appendChild(ele);  }  }  this.Img = function(){  this.appendTo = function(target){  var ele = document.createElement("img");  ele.src = this.src;  target.appendChild(ele);  }  }  this.Div = function(){  this.appendTo = function(target){  var ele = document.createElement("div");  ele.style = "position:absolute;color:red";  ele.innerHTML = this.text;  target.appendChild(ele);  }  }  this.Input = function(){  this.appendTo = function(target){  var ele = document.createElement("input");  ele.type = this.type || "text";  ele.value = this.value;  target.appendChild(ele);  }  }  this.create = function(type){  return new this[type];  }  return this;  })();  window.onload = function(){  var \_input = DomFactory.create("Input");  \_input.value = "hello";  \_input.appendTo(document.body);    var \_div = DomFactory.create("Div");  \_div.text = "hehehe";  \_div.appendTo(document.body);  } |

* 适配器模式的概念
  + 用于适配接口不同，用于包装参数，或转换参数
* 实现方式1

|  |
| --- |
| var myObject = {  prop1 : "aaa",  prop2 : "bbb",  prop3 : "ccc"  };  function test(p1, p2, p3){  alert(p1+p2+p3);  }  //原始调用  test(myObject.prop1,myObject.prop2,myObject.prop3);  //适配器  function testAdapter(obj){  test(obj.prop1,obj.prop2,obj.prop3);  }  //使用适配器调用  testAdapter(myObject); |

* 实现方式2

|  |
| --- |
| var ipad = {  playgame: function(){  console.log("play game!")  }  };  var mobileAdapter = function(target){  this.\_target = target;  this.playgame = function(){  this.\_target.playgame();  }  this.callphone = function(){  if(!this.\_target.callphone){  console.log("not support function [callphone]!");  } else {  this.\_target.callphone();  }  }  }  function test(mobile){  mobile.playgame();  mobile.callphone();  }  //原始调用  test(ipad); //错误 mobile.callphone is not a function  //适配器调用  test(new mobileAdapter(ipad)); |