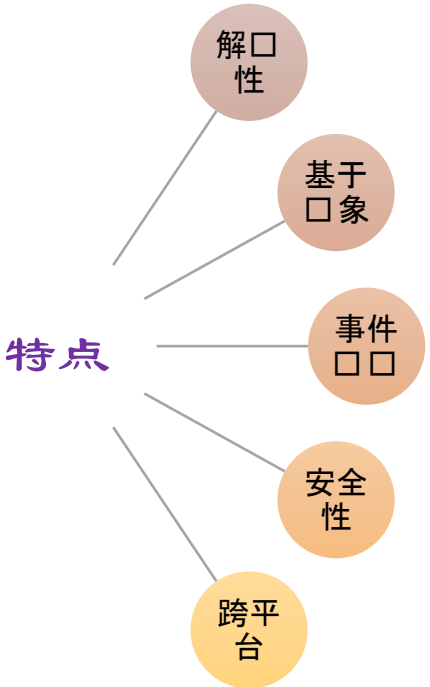
第4讲 JavaScript

1. JavaScript简介

- JavaScript是一种基于对象和事件驱动并具有安全性能的解释型脚本语言。
- 它不需要进行编译，而是直接嵌入在HTTP页面中，把静态页面转变成支持用户交互并响应应用事件的动态页面。

2. JavaScript的特点



3. JavaScript与Java语言

➢基于对象和面向对象

- JavaScript是一种基于对象和事件驱动的脚本语言，它本身提供了非常丰富的内部对象供设计人员使用。
- Java是一种真正的面向对象的语言，即使是开发简单的程序，也必须声明对象。

解释和编译

- JavaScript是一种解释性编程语言，其源代码在发往客户端执行之前不需经过编译，而是将文本格式的字符代码发送给客户端由浏览器解释执行
- Java的源代码在传递到客户端执行之前，必须经过编译才可以执行

弱变量和强变量

- JavaScript采用弱变量，即变量在使用前无须声明其具体类型，解释器在运行时将检查其数据类型
- Java则使用强类型变量检查，即所有变量在使用之前必须声明。

4. 使用JavaScript

- 页面中可以直接嵌入
  - 例4-1
- 链接外部的JavaScript文件

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

5. 基本语法

- 区分大小写
- 每行结尾的分号可有可无
- 变量是弱类型，使用var定义变量
- 使用花括号标记代码块
- 注释方法同C/C++语言注释方法

6. 数据类型

➢数值型

- ●整型：包括正整数、负整数和0，可以用十进制，8进制和16进制表示。如：729，071, 0x3B等
- ●浮点型：整数部分加上小数部分组成，只能采用十进制，可以使用科学计数法或者标准方式来表示。如：3.1415926，1.6E5等

- 字符型：用单引号或者双引号括起来的一个或者多个字符。如：‘a’，‘石家庄铁道大学’，“中华人民共和国”
- 布尔型：布尔型数据只有两个值，即true或false，主要用来说明或代表一种状态或标志。在JavaScript中，也可以使用整数0表示false，使用非0的整数表示true

- 空值：用于定义空的或者不存在的对象。其值为null。
- 未定义值：当使用了一个并未声明的变量，或者使用了一个已经声明但没有赋值的变量时，将返回未定义值(undefined)。

7. 变量的声明

用于指定变量名，该变量名必须遵守变量的命名规则

```
var  variable  ;
```

可以使用一个关键字var同时声明多个变量

```
var now , year , month , date ;
```

可以在声明变量的同时对其进行赋值，即初始化

```
var now="2009-05-12",year="2009", month="5",date="12";
```

8. 算术运算符

运 算 符	描　述	示　例
+	加运算符	4+6　//返回值为10
-	减运算符	7-2　//返回值为5
*	乘运算符	7*3　//返回值为21
/	除运算符	12/3　返回值为4
%	求模运算符	7%4　返回值为3
++	自增运算符。该运算符有两种情况：i++（在使用i之后，使i的值加1）；++i（在使用i之前，先使i的值加1）	i=1; j=i++　//j的值为1，i的值为2 i=1; j=++i　//j的值为2，i的值为2
--	自减运算符。该运算符有两种情况：i--（在使用i之后，使i的值减1）；--i（在使用i之前，先使i的值减1）	i=6; j=i--　//j的值为6，i的值为5 i=6; j=--i　//j的值为5，i的值为5

9. 赋值运算符

运算符	描　述	示　例
=	将右边表达式的值赋给左边的变量	userName="mr"
+=	将运算符左边的变量加上右边表达式的值赋给左边的变量	a+=b //相当于a=a+b
-=	将运算符左边的变量减去右边表达式的值赋给左边的变量	a-=b //相当于a=a-b
*=	将运算符左边的变量乘以右边表达式的值赋给左边的变量	a*=b //相当于a=a*b
/=	将运算符左边的变量除以右边表达式的值赋给左边的变量	a/=b //相当于a=a/b
%=	将运算符左边的变量用右边表达式的值求模，并将结果赋给左边的变量	a%=b //相当于a=a%b

10. 比较运算符

给定x=5

运算符	描述	例子
==	等于	x==8 为 false x==5为true x=='5'为true
===	全等（值和类型）	x===5 为 true x==="5" 为 false
!=	不等于	x!=8 为 true
!==	值不相等或者类型不相等	x!==5为false
>	大于	x>8 为 false
<	小于	x<8 为 true
>=	大于或等于	x>=8 为 false
<=	小于或等于	x<=8 为 true

11. 逻辑运算符

算符	描述	例子
&&	and	(x < 10 && y > 1) 为 true
||	or	(x==5 || y==5) 为 false
!	not	!(x==y) 为 true

12. 字符串运算符

算符	描述	例子
+	连接两个字符串	var a=“one” + “ world”
+=	连接两个字符串，并将结果赋给第一个字符串	var a = “one” a+=“ world”

13. 函数的定义

- 函数是由关键字function 、 函数名加一组参数以及置于大括号中需要执行的一段代码定义的。

```
function 函数名([参数1, 参数2,……]){
    语句;
    [return 返回值;]
}
```

14. 函数的调用

- 调用不带参数的函数，使用函数名加上括号
- 调用带参数的函数，在括号中加上需要传递的参数
- 如果包含多个参数，各参数之间用逗号分隔
- 如果函数有返回值，可将其赋给一个变量

15. 匿名函数

- 没有名字的函数
- 匿名函数可以赋给一个变量（Funtion对象），再通过该变量名去调用此函数，例如

```
var abc = function(x, y) {return x*y;};
var result = abc(2,3);
```

这样做可以动态修改调用的函数

- 匿名函数也可以直接使用，例如

```
(function(x, y) {return x*y;}) (2,3);
```

16. 事件和事件处理程序

- 事件(Event)是系统运行或者用户操作中产生的一系列时间节点
- 事件处理程序(Event Handler)是在某个事件发生时执行的程序，用来完成所需的操作任务
- JavaScript编程主要是对各种对象产生的各种事件编写事件处理程序，以完成所需的操作

17. JavaScript常用事件

事　件	何时触发
onabort	对象载入被中断时触发
onblur	元素或窗口本身失去焦点时触发
onchange	域的内容发生改变时触发
onclick	单击鼠标左键时触发
ondblclick	双击鼠标左键时触发
onerror	出现错误时触发
onfocus	任何元素或窗口本身获得焦点时触发
onkeydown	某个键盘按键被按下
onkeypress	某个键盘按键被按下并松开
onkeyup	某个键盘按键被松开
onload	一张页面或一幅图像完成加载

事　件	何时触发
onmousedown	鼠标按键被按下
onmousemove	鼠标被移动
onmouseout	鼠标从某元素移开
onmouseover	鼠标移到某元素之上
onmouseup	鼠标按键被松开
onreset	重置按钮被点击
onresize	窗口或框架被重新调整大小
onscroll	在任何带滚动条的元素或窗口上滚动时触发
onselect	文本被选中
onsubmit	确认按钮被点击
onunload	用户退出页面

18. 事件处理程序的绑定

- 将事件和用户编写的事件处理程序联系在一起称为事件处理程序的绑定。
- 两种方法：
  - HTML属性绑定
  - JavaScript代码绑定（推荐使用）

(1) HTML属性绑定举例

```
<script language="javascript">
    function calculate(){
        ……
    }
  </script>

<input type="button" id="jisuan" name="jisuan"
value="计算" onclick="calculate()" />
```

- 例4-2

(2) JavaScript代码绑定举例

```
<input name="jisuan" type="button" value="保存">
…
<script type="text/javascript">
    document.getElementById("jisuan").onclick = calculate;
</script>
```

- 例4-3, 4-4

19. JavaScript内置对象

- JavaScript语言自身内置的对象，具有自己的属性和方法
  - Array对象
  - Date对象
  - String对象
  - Math对象
  - Number对象

(1) Array对象

- Array对象用于在单个的变量中存储多个值
- 创建Array对象的语法：

  new Array();
  new Array(*size*);
  new Array(*element0*, *element1*, ..., *elementn*);
- 常用属性：
  - ✓ length: 设置或返回数组中元素的数目

- 常用方法：
  - ✓ concat(): 连接两个或者更多的数组，并返回结果
  - ✓ pop(): 删除并返回数组的最后一个元素
  - ✓ push(): 向数组的末尾添加一个或更多元素，并返回新的长度
  - ✓ sort()：对数组的元素进行排序

(2) Date对象

- 用于处理日期和时间
- 创建Date对象的语法

  ```
  var myDate = new Date();
  //当前日期和时间作为该日期对象初始值
  ```

- 常用方法
  - Date(): 返回当前的日期和时间
  - getDate(): 返回一个月中的某一天(1-31)
  - getDay()：返回一周中的某一天(0-6)
  - getMonth()：返回月份(0-11)

- 常用方法
  - getFullYear(): 返回四位数字的年份
  - getHours()：返回小时(0-23)
  - getMinutes()：返回分钟(0-59)
  - getSeconds(): 返回秒数(0-59)
  - getMilliseconds(): 返回毫秒数(0-9990)
  - toString(): 转化为字符串
  - toUTCString()：转换为世界时字符串
  - toLocaleString(): 转换为本地时间字符串
- 例4-1, 4-5

(3) String对象

- String对象用于处理文本（字符串）
- 创建String对象的语法

  var str = new String(s);

  //参数s是初值
- 常用属性
  - ✓ length: 字符串的长度

- 常用方法
  - ✓ charAt(): 返回指定位置的字符
  - ✓ concat(): 连接字符串
  - ✓ indexOf(): 检索字符串
  - ✓ replace(): 替换字符串
  - ✓ substr(): 返回子串
  - ✓ split(): 把字符串分割为字符串数组
- 例4-6

(4) Math对象

- Math对象用于执行数学操作
- Math对象无需创建，可以直接使用其属性和方法

```
var pi_value = Math.PI;
var sqrt_value = Math.sqrt(15);
```

- 常用属性
  - E：返回自然对数的底数
  - PI：返回圆周率

- 常用方法
  - ✓ abs(x): 返回x的绝对值
  - ✓ exp(x): 返回e的x次方
  - ✓ log(x): 返回x的自然对数
  - ✓ random(): 返回0和1之间的随机数
  - ✓ sqrt(x): 返回x的平方根
  - ✓ round(x): 返回x四舍五入后的值

(5) 全局对象

- 全局对象代表JavaScript运行环境
- 全局对象的属性和函数相当于全局符号常量和全局函数，可以单独使用
- 常用属性
  - Infinity：代表正的无穷大的数值
  - NaN：代表某个值不是数字值
  - undefined：表示未定义的值

- 常用全局函数
  - ✓ eval(): 计算JavaScript字符串，并把它作为脚本代码来执行
  - ✓ isFinite(): 检查某个值是否为有穷大的数
  - ✓ isNaN(x): 检查某个值是否是数字。
    Number(): 把对象的值转换为数字
  - ✓ parseFloat(): 解析一个字符串并返回一个浮点数
  - ✓ parseInt(): 解析一个字符串并返回一个整数
  - ✓ String(): 把对象的值转换为字符串

20. 浏览器对象

- 浏览器(Browser)在解释执行JavaScript代码时提供的对象，使得JavaScript通过这些对象的属性或方法对浏览器相关信息进行读取或者修改操作
  - window对象：表示浏览器打开的窗口
  - Navigator对象：包含有关浏览器的信息
  - Screen对象：有关客户端屏幕显示的信息
  - History对象：用户访问过的历史记录

Window对象

- Window 对象表示浏览器中打开的窗口。
- Window对象是全局对象，其属性和方法可以直接使用，而不必加上Window.
- 常用属性：
  - document对象：对Document对象的只读引用
  - innerHeight：返回窗口的文档显示区高度
  - innerWidth：返回窗口的文档显示区宽度
  - outerHeight：返回窗口的外部高度
  - outerWidth: 返回窗口的外部宽度

- 常用方法
  - alert()：显示带有一段消息和一个确认按钮的警告框
  - blur()：把键盘焦点从顶层窗口移开
  - close()：关闭浏览器窗口。例4-7
  - confirm()：显示带有一段消息以及确认按钮和取消按钮的对话框
  - focus()：把键盘焦点给予一个窗口
  - open()：打开一个新的浏览器窗口。例4-7
  - setInterval()：按照指定的周期（以毫秒计）来调用函数或计算表达式。例4-5
  - setTimeout()：在指定的毫秒数后调用函数或计算表达式。例4-8
  - clearInterval()：取消由 setInterval() 设置的 timeout。
  - clearTimeout()：取消由 setTimeout() 方法设置的 timeout。

21. HTML DOM

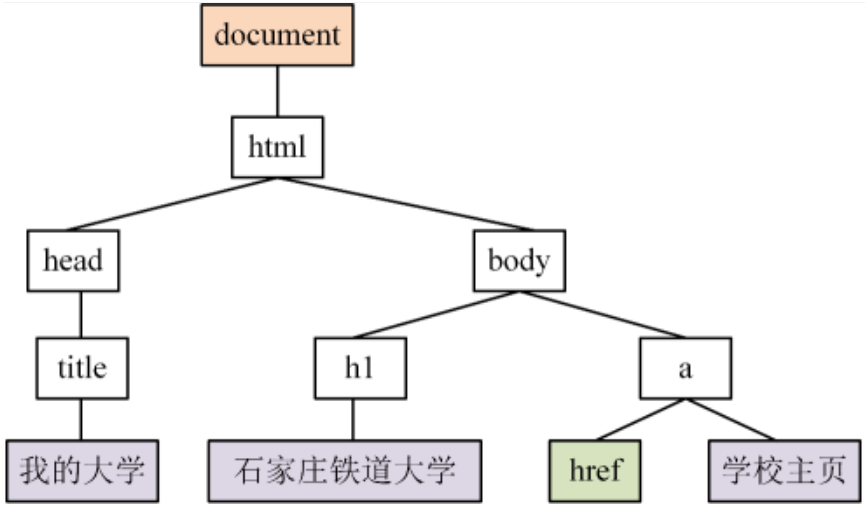
- DOM: 文档对象模型(Document Object Model)，意思是把一个结构化文档的各个组成元素视为一个个的对象，把整个文档视作一个结构化的多个对象构建的模型
- HTML DOM：将一个完整的HTML视作一个由多个HTML标记元素对象组成的模型，它是一个树结构

HTML DOM Tree

- HTML文档中的所有内容都是结点
- 整个文档是一个文档结点
- 每个 HTML 标记元素对象是元素结点
- 每个HTML标记元素内的文本是文本结点
- 每个HTML标记元素的属性是属性结点

HTML DOM Tree举例

```
<html>
   <head>
      <title>我的大学</title>
   </head>
   <body>
      <h1>石家庄铁道大学</h1>
      <a href="www.stdu.edu.cn">学校主页</a>
   </body>
<html>
```


document
html
head
body
title
h1
a
我的大学
石家庄铁道大学
href
学校主页

22. HTML DOM的编程接口

- JavaScript提供了操作HTML DOM树上各个结点的属性和方法
- 元素、属性、文本和文档结点对象的共有属性
  - nodeName：结点的名称——元素结点的名称同标签名，属性结点的名称同属性名，文本结点的名称是#text，文档结点的名称是#document)
  - nodeValue：结点的值——元素结点的值是undefined 或 null，文本结点的值是文本本身，属性结点的值是属性值
  - nodeType：结点的类型——元素结点为1,属性结点为2, 文本结点为3, 文档结点为9

元素结点对象通用的部分属性

- attributes：返回元素结点的属性结点集合
- childNodes：返回元素结点的子结点集合
- className：设置或返回元素的class属性
- firstChild：返回元素的首个子结点
- innerHTML：元素结点的文本值（即开始和结束标记之间的内容）
- nextSibing：返回位于相同结点树层级的下一个结点
- parentNode：元素结点的父结点
- style：设置或返回元素的style属性
- tagName：返回元素的标签名

元素结点对象通用的部分方法

- appendChild()：增加新的子结点
- cloneNode()：克隆元素
- removeChild()：删除子结点
- replaceChild()：替换子结点。
- insertBefore()：在指定的子结点前面插入新的子结点
- getAttribute()：返回指定的属性值
- setAttribute()：把指定属性设置或修改为指定的值

- 文档结点对象(document)独有的部分方法
  - getElementById()：返回带有指定ID的元素
  - getElementsByTagName()：返回带有指定标签名称的所有元素的结点列表
  - getElementsByClassName()：返回包含有指定类名的所有元素的结点列表
  - createElement()：创建元素结点
  - createAttribute()：创建一个属性结点
  - createTextNode()：创建文本结点

23. HTML DOM对象

- http://www.w3school.com.cn/jsref/index.asp
- 例4-9：修改HTML的内容和样式
- 例4-10：动态菜单
- 例4-11：跟随鼠标走的文字
- 例4-12：俄罗斯方块
- 推箱子：https://www.html5tricks.com/html5-canvas-box-game.html
- 植物大战僵尸：http://js.alixixi.com/pvz/

24. JavaScript的调试

- 常用的浏览器都支持开发者模式，用于程序员对JavaScript代码进行调试，推荐Google Chrome。
- 开发者模式的功能：
  - 查看当前HTML文件的实时代码和DOM树
  - 查看当前HTML文件中各个标签的样式、属性和绑定的事件驱动程序
  - 控制台可以查看当前Javascript代码运行相关信息
  - 可以查看当前网页的网络连接相关信息、
  - 可以对JavaScript代码设置断点，进行调试

25. Ajax技术

- Ajax是Asynchronous JavaScript and XML的缩写，意思是异步的JavaScript与XML。
- 使用Ajax技术的主要目的是客户端与服务器交换数据时，客户端页面不会完全重新加载，只是网页的局部进行更新
- 浏览器提供了XMLHttpRequest对象来完成与服务器交换数据的操作

Ajax的工作流程

- 发送请求：
  - 初始化XMLHttpRequest对象
  - 为XMLHttpRequest对象指定一个回调函数
  - 创建一个与服务器的链接
- 待服务器发送回响应后，处理服务器响应。响应的结果可能是数字、字符串、XML或JSON数据。
- 例4-13

25. JQuery技术

- jQuery是一套简洁、快速、灵活的JavaScript脚本库，它是由John Resig于2006年创建的，它帮助我们简化了JavaScript代码。
- 引入JQuery：

```
<script type="text/javascript" src="jquery-1.7.2.min.js"></script>
```

Jquery的通常编程方法

在Web文件加载完成后，选取所需的HTML对象，为其绑定所需的事件处理程序代码。

1. 选取对象使用jQuery的工厂函数实现：

   $(“…”)

2. 绑定方法：

   $(“…”).事件名(事件处理程序)

   其中，事件处理程序通常使用匿名函数function(){}实现。

3. 文件加载完成后需要编写的代码的格式如下：

   $(document).ready(function(){…….})

JQuery的工厂函数

从一个“$”符号和一对“()”开始，类似于CSS的选择器，是为HTML元素的选取编制的。

- 在参数中使用标记名

例：$("div")：用于获取文档中全部的 。

- 在参数中使用ID

例：$("#username")：用于获取文档中ID属性值为username的一个元素。

- 在参数中使用CSS类名

例：$(".btn_grey")：用于获取文档中使用CSS类名为btn_grey的所有元素

http://www.w3school.com.cn/jquery/jquery_ref_selectors.asp

JQuery中的常见事件举例

Event 函数	绑定函数至
$(document).ready(function)	将函数绑定到文档的就绪事件（当文档完成加载时）
$(selector).click(function)	触发或将函数绑定到被选元素的点击事件
$(selector).dblclick(function)	触发或将函数绑定到被选元素的双击事件
$(selector).focus(function)	触发或将函数绑定到被选元素的获得焦点事件
$(selector).mouseover(function)	触发或将函数绑定到被选元素的鼠标悬停事件

JQuery的操作函数

- jQuery效果：可以实现一些动画效果
- jQuery文档操作：可以在匹配的元素中插入、增加或者删除所需的内容或者结点。
- jQuery属性操作：可以修改匹配的元素的属性
- jQuery CSS操作：可以修改匹配的元素的样式
- jQuery Ajax技术：可以使用Ajax技术与服务器进行交互

JQuery举例

- 例4-14
- 例4-15

26. json

- json指的是 JavaScript 对象表示法（JavaScript Object Notation）
- json是轻量级的文本数据交换格式
- json独立于语言
- json具有自我描述性，更易理解

json语法规则

- 数据在名称/值（key/value）对中
- 数据由逗号分隔
- 花括号保存对象
- 方括号保存数组

json名称/值对

- 包括字段名称（在双引号中），后面写一个冒号，然后是值
- 例如："firstName" : "John"

json值类型

- 数字（整数或浮点数）
- 字符串（在双引号中）
- 逻辑值（true 或 false）
- 数组（在方括号中）
- 对象（在花括号中）
- null

json举例

- var1={ "firstName":"John", "lastName":"Doe" }
- var2=[
  { "firstName":"John" , "lastName":"Doe" },
  { "firstName":"Anna" , "lastName":"Smith" },
  { "firstName":"Peter" , "lastName":"Jones" }
  ]
- var3 = { "student" : { "firstName":"John" , "lastName":"Doe" }}
- var4 = { "students" : [
  { "firstName":"John" , "lastName":"Doe" },
  { "firstName":"Anna" , "lastName":"Smith" },
  { "firstName":"Peter" , "lastName":"Jones" }
  ]
  }