**gif：**

支持动画

只有全透明和不透明两种模式

只有256种颜色

**jpg：**

1. 采用有损压缩算法
2. 体积较小
3. 不支持透明
4. 不支持动画

**png：**

1. 采用无损压缩算法
2. 体积也相对较小
3. 支持背景透明
4. 不支持动画

**svg：**

1. 无论怎样放大都不会失真
2. 不太应用照片
3. 保存颜色和形状相对简单的图片

### 

**NaN：**(Not a number：非数值)

**isNaN：**判断一个变量是否为非数字的类型，返回true或false

### JavaScript中的基本数据类型

1. **number**(数值类型)(最大值：number.MAX\_VALUE)
2. **string**(字符串类型)
3. **boolean**(布尔类型)
4. **null**(空类型)
5. **undefined**(未定义类型)
6. **Object**

**BOM模型：**浏览器对象模型(Browser Object Model)

**DOM：**文档对象模型(Document Object Model)

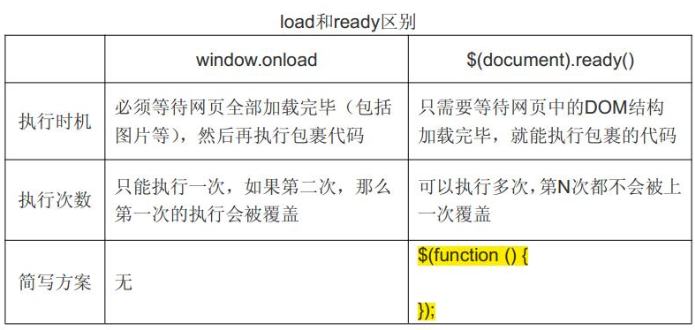
当浏览器把一个html文件加载到内存中之后，这个html文件，就是一个Document对象

# Window.onload和$(document).ready()

**$(document).ready:**是DOM结构绘制完毕后就执行，不必等到加 载完毕。 意思就是DOM树加载完毕，就执行，不必等到页面中图 片或其他外部文件都加载完毕。并且可以写多个.ready。

**window.onload:**是页面所有元素都加载完毕，包括图片等所有元 素。只能执行一次。

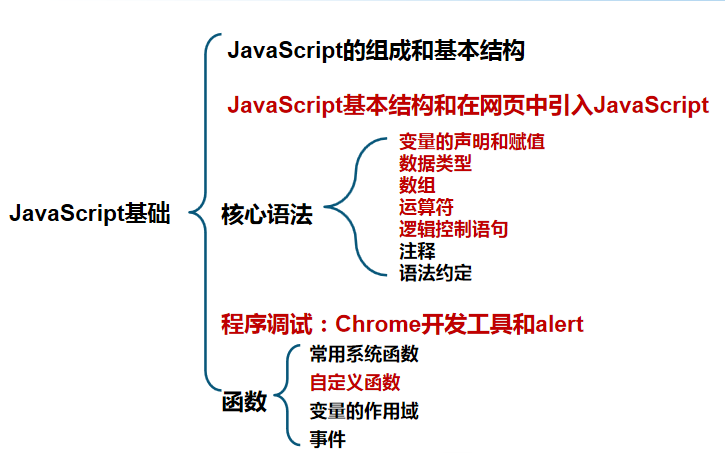
$(document).ready的执行时间要早于window.onload



# 访问页面元素

1. **getElementById()：**通过id得
2. **getElementsByTagName()：**根据标签名得
3. **getElementsByName()：**通过名称(name)来得

# 1.JavaScript基础



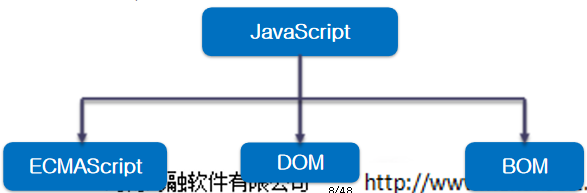
JS是一种基于对象和事件驱动的、并具有安全性能的脚本语言

## 特点：

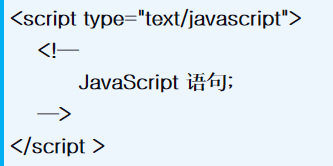
1.向HTML页面中添加交互行为

1. 脚本语言，语法和Java类似
2. 解释性语言，边执行边解释

## JS组成：ECMAScript+DOM+BOM

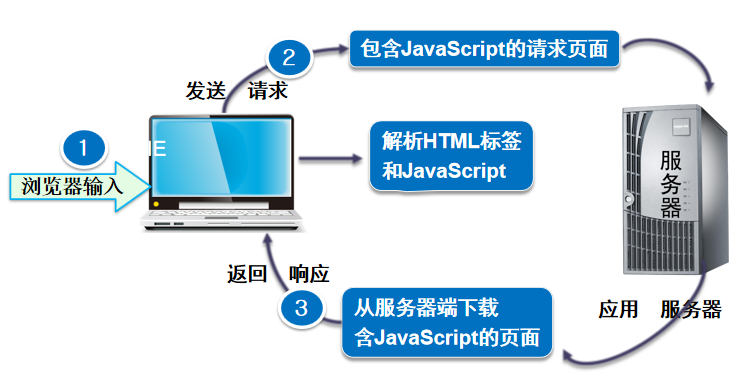


**基本结构**



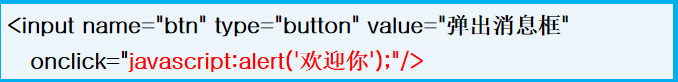
**<script>…</script>可以包含在文档中的任何地方，只要保证这些代码在被使用前已读取并加载到内存即可**

## 执行原理



## 引入方式：

1. 使用**<script>**
2. 外部JS文件<script src=”**a.js**” type=”text/javascript”></script>
3. 直接在HTML标签中



# JS核心语法



## 变量

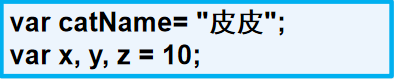
**变量可以不经声明而直接使用，但这种方法很容易出错，也很难查找排错，不推荐使用**

### 先声明变量在赋值

**var：**用于声明变量的关键字



### 通知声明和赋值变量



### 不声明直接赋值



## 数据类型

1. **Undefined**



1. **Null**



1. **Number**



1. **Boolean**



1. **String**



## typeof运算符：检测变量的返回值

1. undefined：变量被声明后，但未被赋值
2. string：用单引号或双引号来声明的字符串
3. boolean：true或false
4. number：整数或浮点数
5. object：javascript中的对象、数组和null

## String对象

属性：1.字符串对象.length

1. 方法
2. 字符串对象.方法名();



### 字符串转义符

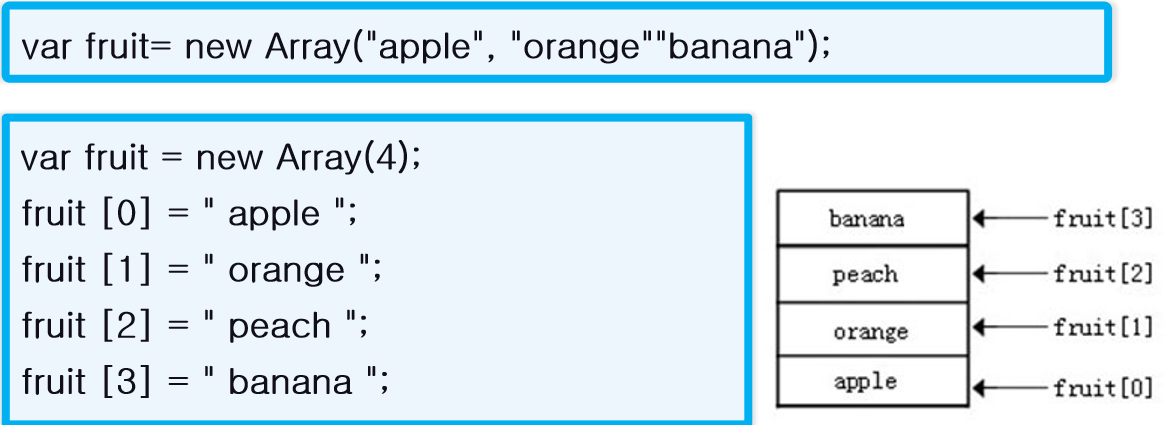


## 数组

创建数组：var 数组名称 = new Array(size)

size：表示数组中可以存放的元素总数

### 为数组元素赋值

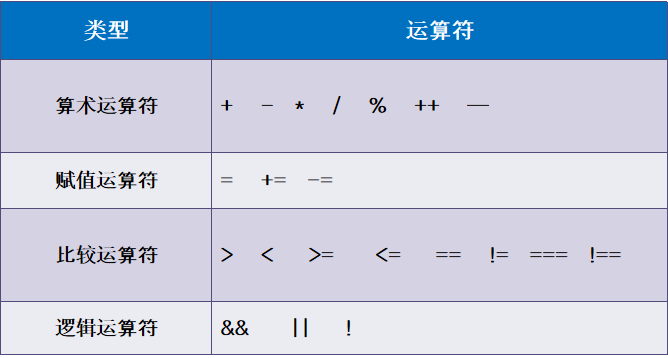


### 访问数组：数组名【下标】

### 常用属性和方法

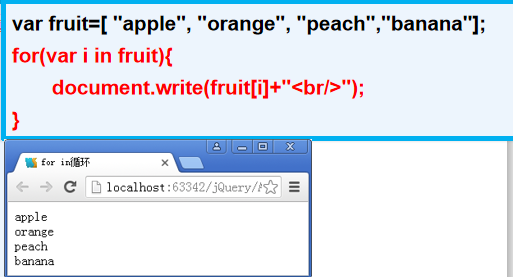


## 运算符号



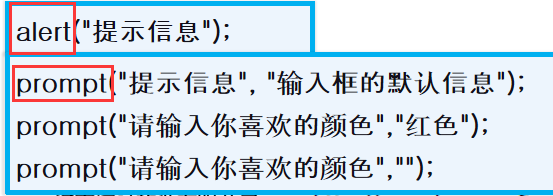
## 逻辑控制语句

1. if条件语句
2. Switch多分支语句
3. for、while循环语句
4. for-in



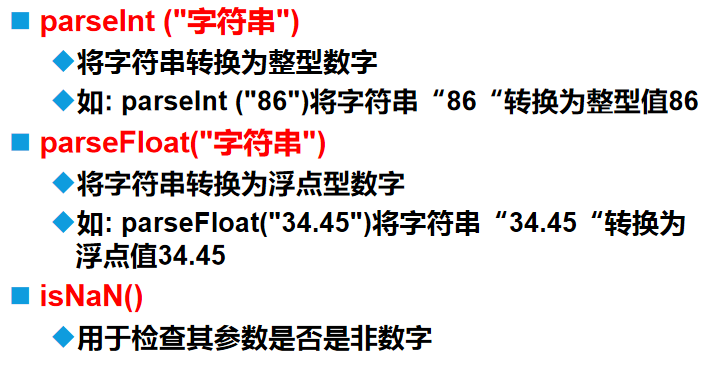
## 循环中断：break、continue

## 输入输出：alert()、prompt()

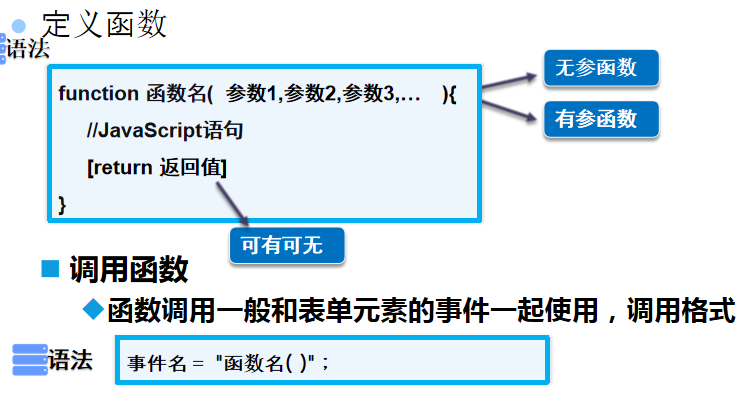


# 函数

## 常用系统函数



## 自定义函数



### 有参、无参调用



## 事件

1. onload：一个页面或一幅图像完成加载
2. onlick：鼠标单击某个对象
3. onmouseover：鼠标移动到某元素上
4. onkeydown：某个键盘按钮被按下
5. onchange：域的内容被改变

# 2.JS操作DOM对象

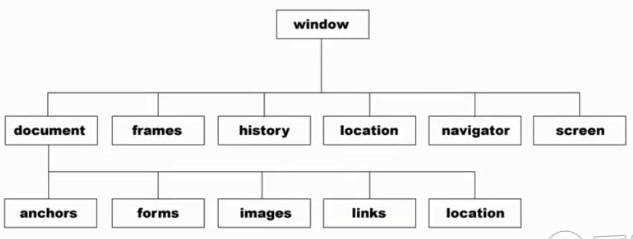


## BOM模型：浏览器对象模型(Browser Object Model)

可实现功能：1.弹出新的浏览器窗口

1. 移动、关闭浏览器窗口以及调整窗口的大小
2. 页面的前进、后退

## window对象



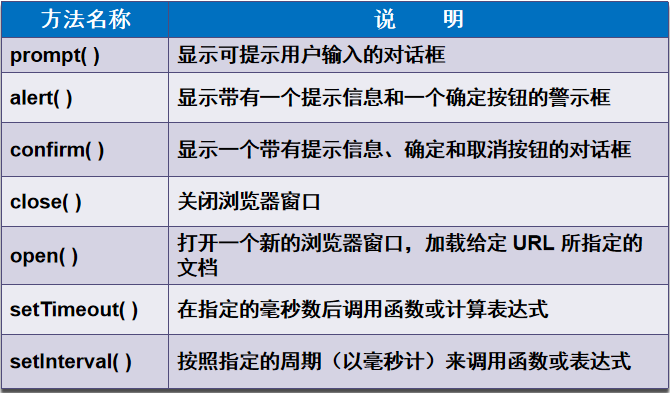
### 常用属性

1. **history：**有关客户访问过得URL的信息
   1. back()：前往浏览器历史条目**前一个URL**，类似后退
   2. forward()：前往浏览器历史条目**下一个URL**，类似前进
   3. go(num)：浏览器在history对象中向前或向后
2. **location：**有关当前URL的信息

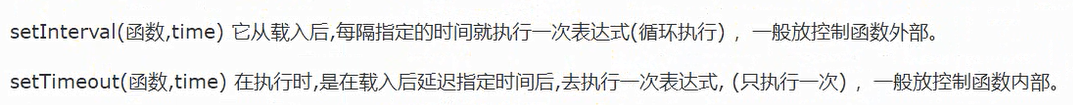
### 语法：window.属性=“属性值”

Eg：跳转到百度首页：window.location=”http://www.baidu.com”;

### 常用方法



### 定时函数：setTimeout(函数，时间)、setInterval(函数，时间)



如果要多次调用，使用setInterval()或者让disptime()自身再次调用setTimeout()

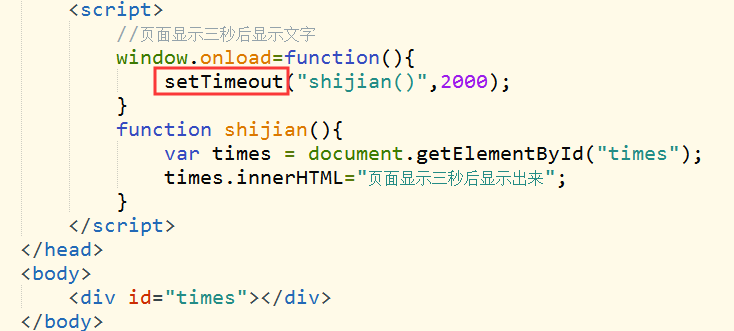
setInterval()：**频率函数**

语法：**setTimeout**(“调用的函数”，**等待**的毫秒数)

在指定的毫秒数后调用函数或计算表达式

**setInterval**(“调用的函数”，**间隔**的毫秒数)

按照指定的周期（以毫秒计）来调用函数或计算表达式





### 清除函数：clearTimeout()、clearInterval ()

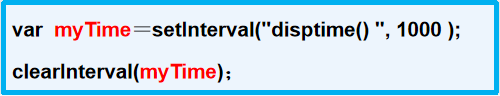
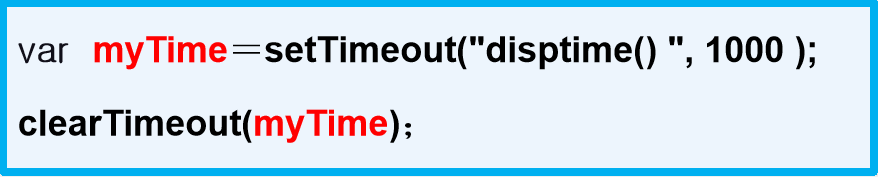
语法：**clearTimeout**(setTimeOut()返回的ID值)

取消由 **setTimeout()** 方法设置的 timeout --(之前定义接收的值)

**clearInterval**(setInterval()返回的ID值)

取消由 **setInterval()** 设置的 timeout--(之前定义接收的值)

Eg:



## confirm()方法：将弹出一个确认对话框



**confirm()与alert ()、 prompt()区别**

1. **alert( )：**一个参数，仅显示警告对话框的消息，无返回值，不能对 脚本产生任何改变
2. **prompt( )：**两个参数，输入对话框，用来提示用户输入一些信息， 单击“取消”按钮则返回null，单击“确定”按钮则返回用户输入的值， 常用于收集用户关于特定问题而反馈的信息
3. **confirm( )**：一个参数，确认对话框，显示提示对话框的消息、“确定”按钮和“取消”按钮，单击“确定”按钮返回true，单击“取消”按钮返回false，因此与if-else语句搭配使用

## open()方法

**window.open(“弹出窗口的url”，”窗口名称”，”窗口特征”)**

第二个参数：给打开的新窗口起一个名字，若以后在打开，再去加载url，就在这个已经起好名字的目标窗口加载url

打开当前窗口的父窗口的window对象

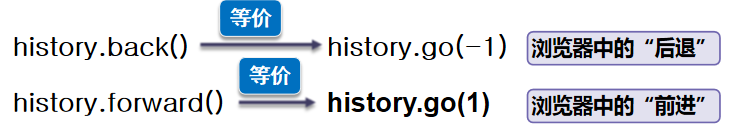
窗口特征：



## history对象

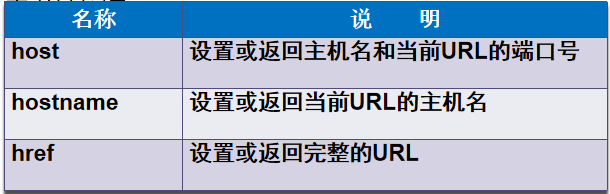
### 常用方法

1. **back()：**加载history对象列表中的前一个URL
2. **forward()：**加载history对象列表中的下一个URL
3. **go()：**加载history对象列表中的某个具体URL

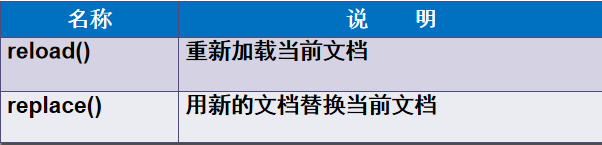


## location对象

### 常用属性



### 常用方法



## Document对象

### 常用属性

1. **referrer：**返回载入当前文档的URL
2. **URL：**返回当前文档的URL

### 语法：doucument.referrer/URL

### 应用：

1. 判断页面是否是链接进入
2. 自动跳转到登录页面

### 常用方法



## Document对象访问页面元素

1. 动态改变层、标签中的内容
2. 访问相同name的元素
3. 访问相同标签的元素

## JS内置对象

**Array：**用于在单独的变量名中存储一系列的值

**String：**用于支持对字符串的处理

**Math：**用于执行常用的数学任务，它包含了若干个数字常量和函数

**Date：**用于操作日期和时间

**Var 日期对象=new Date(参数)**(参数格式：MM DD,YYYY,hh:mm:ss)

### 常用方法



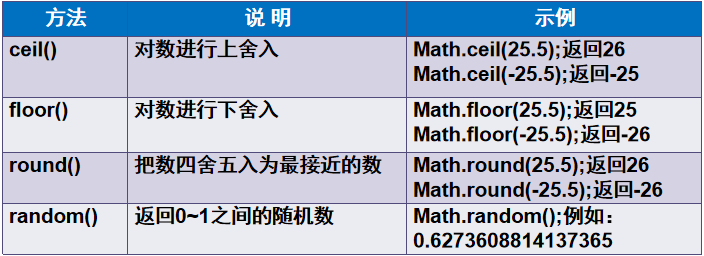
**制作时钟特效**





## Math对象

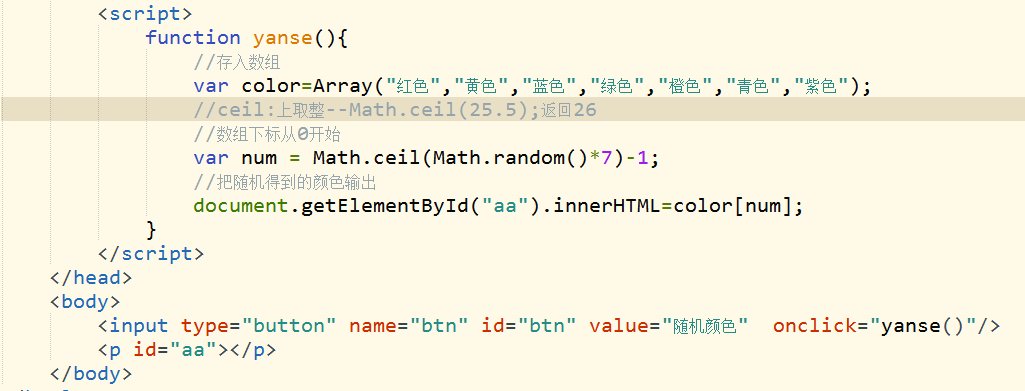
### 常用方法

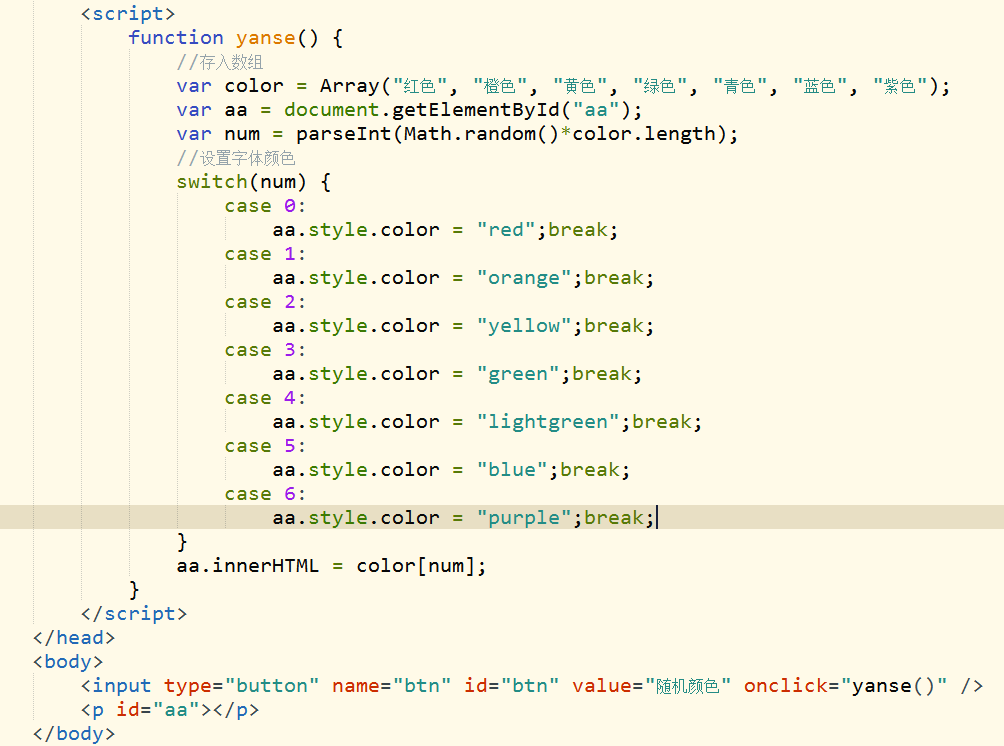


如何实现返回的整数范围为2~99？



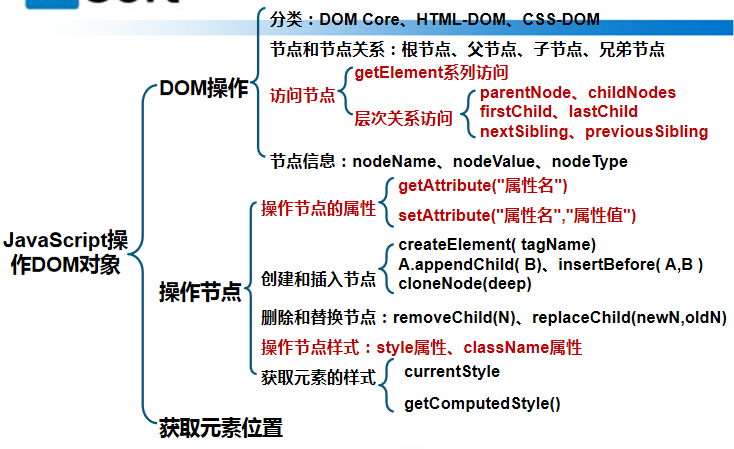
随机颜色：





## Date对象

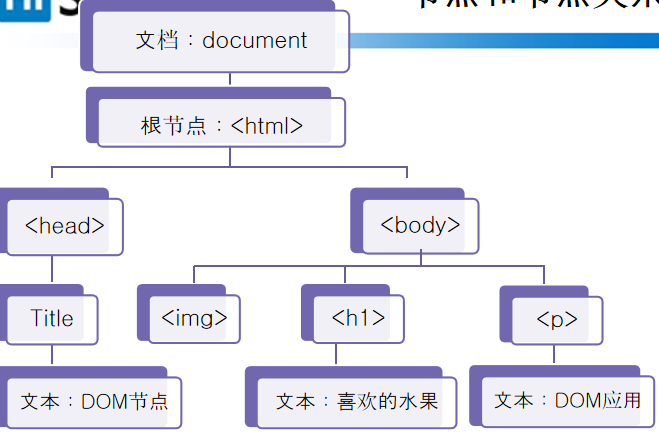
# 3.JavaScript操作DOM对象



## DOM：文档对象模型(Document Object Model)

1. DOM Core
2. HTML-DOM
3. CSS-DOM

## 节点和节点的关系



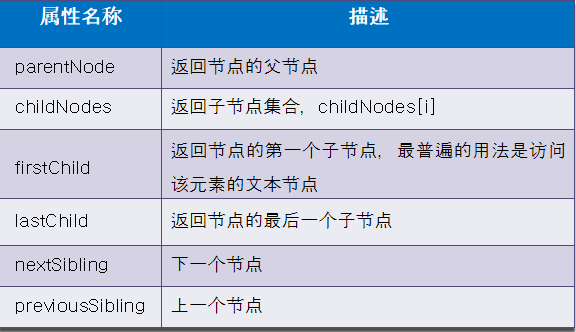
## 访问节点

### 使用getElement系列方法访问指定节点

* 1. getElementById()：通过id得
  2. getElementsByTagName()：根据标签名得
  3. getElementsByName()：通过名称(name)来得

### 根据层次关系访问节点

### 节点属性



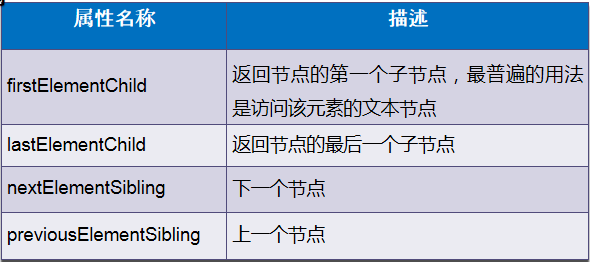
### \*\*\*element属性\*\*\*

**firstElementChild：返回节点的第一个子节点**

**lastElementChild：返回最后一个子节点**

**nextElementSibling：下一节点**

**previousElementSibling：上一节点**



## 节点信息

1. nodeName：节点名称
2. nodeValue：节点值
3. nodeType：节点类型

## 操作节点

### 操作节点的属性

* 1. **getAttribute("属性名")**
  2. **setAttribute("属性名","属性值")**

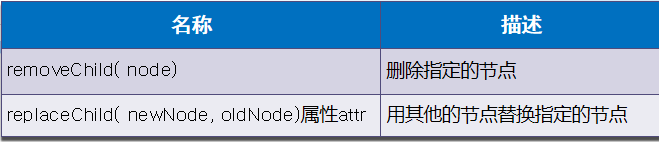
### 创建和插入节点

* 1. **createElement(tagName)：**创建tagName节点
  2. **A.appendChild(B)：**B节点追加到A节点末尾
  3. **insertBefore(A,B)**：把A节点插入到B节点之前
  4. **cloneNode(deep)**：复制某个指定的节点



### 删除和替换节点

* 1. removeChild(node)：**删除**指定的节点
  2. replaceChild(newNode，oldNode)属性attr：用其他节点**替换**指定节点



## 操作节点样式

改变样式的属性：style、className

### Style属性

语法：HTML元素.style.样式属性=”值”

Eg：document.getElementById("titles").style.color="#ff0000";



### className属性

**语法：HTML元素.className=”样式名称”**

Eg：document.getElementById("cart").className="cartOver";

### 获取元素的样式

语法：HTML元素.style.样式属性；

alert(document.getElementById("cartList").display);

语法：document.defaultView.getComputedStyle(元素,null).属性;

var cartList=document.getElementById("cartList");

alert(document.defaultView.getComputedStyle(cartList,null).display);

语法：HTML元素. currentStyle.样式属性;

alert(document.getElementById("cartList").currentStyle.display);**兼容IE浏览器**

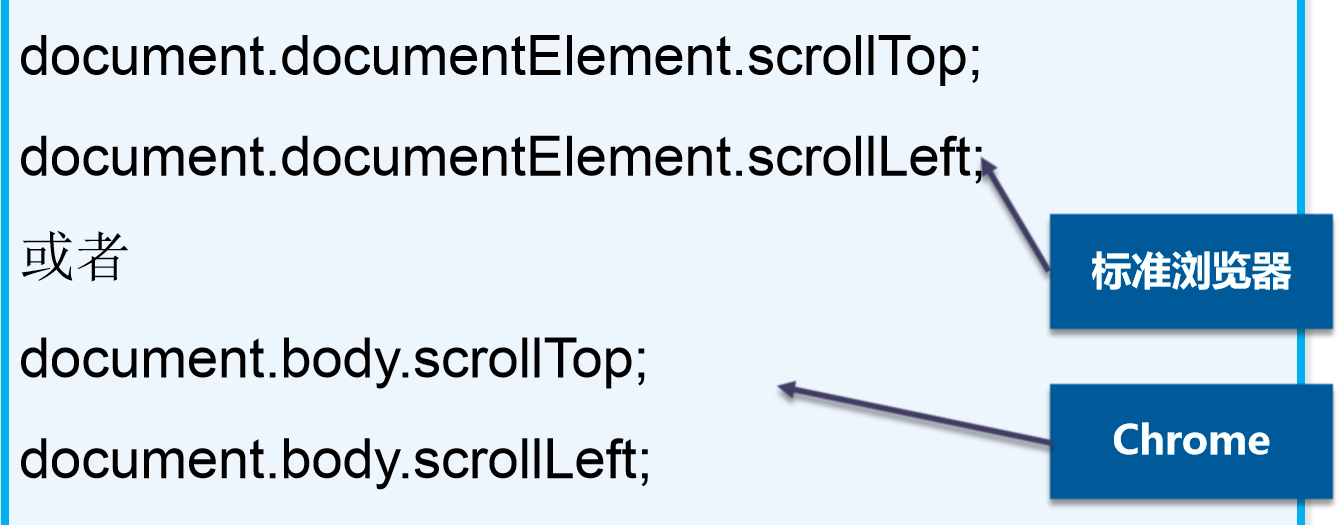
## 获取元素位置

HTML中元素属性

### 元素属性应用



元素属性应用



# 4.JavaScript对象及初识面向对象

## JavaScript中的基本数据类型

1. number(数值类型)
2. string(字符串类型)
3. boolean(布尔类型)
4. null(空类型)
5. undefined(未定义类型)
6. Object

### 转换为字符串：

1. toString()：变量.toString()
2. String()强制转换：String(变量)
3. **加号拼接字符串（隐式转换）**

### 转换为数字型

1. praseInt(String)函数
2. parseFloat(string)函数
3. Number()强制转换函数
4. 隐式转换（- \* /：减 乘 除）

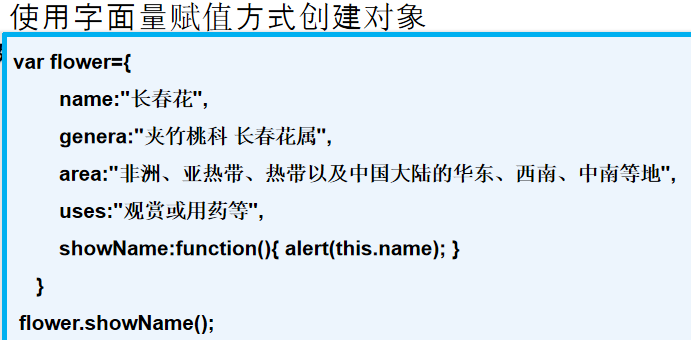
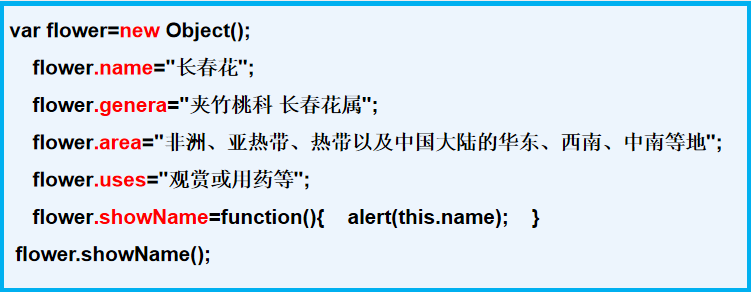
## 创建对象

1. 自定义对象
2. 内置对象

### 自定义对象

基于Object对象的方式创建对象 **通过** “.” **添加属性个方法**

语法：var 对象名称 = new Object();



### 内置对象

常见内置对象：

1. String(字符串)对象：
   1. length属性
   2. indexOf()方法、replace()方法
2. Date(日期)对象
   1. getXXX：**获取**年、月、日、时、分、秒等等
   2. setXXX：**设置**年、月、日、时、分、秒等等
3. Array(数组)对象
   1. length属性
   2. Sort()、concat()、join()方法
4. Boolean(逻辑)对象
   1. True或false
   2. toString()方法
5. Math(算数)对象
6. RegExp对象

## 构造函数和原型对象

构造函数始终都应以一个大写字母开头

### 调用构造函数的4个步骤

1. 通过new运算符，函数内部自动创建一个新对象
2. 将this绑定这个空对象
3. 执行构造函数中的代码
4. 函数自动return返回新对象

## constructor属性指向创建了该对象函数

## instanceof操作符

使用instanceof操作符检测对象类型