目 录

[第1章JavaScript简介](#_Toc17059_WPSOffice_Level1) [3](#_Toc17059_WPSOffice_Level1)

[1.1 JavaScript介绍](#_Toc19680_WPSOffice_Level2) [3](#_Toc19680_WPSOffice_Level2)

[1.2 JavaScript特点](#_Toc22481_WPSOffice_Level2) [4](#_Toc22481_WPSOffice_Level2)

[1.3 JavaScript核心](#_Toc17201_WPSOffice_Level2) [4](#_Toc17201_WPSOffice_Level2)

[1.4 JavaScript作用](#_Toc27374_WPSOffice_Level2) [4](#_Toc27374_WPSOffice_Level2)

[第2章JavaScript基础](#_Toc19680_WPSOffice_Level1) [4](#_Toc19680_WPSOffice_Level1)

[2.1 HTML中使用方式](#_Toc19150_WPSOffice_Level2) [4](#_Toc19150_WPSOffice_Level2)

[2.1.1 script标签](#_Toc19680_WPSOffice_Level3) [4](#_Toc19680_WPSOffice_Level3)

[2.1.2外部导入方式 (推荐)](#_Toc22481_WPSOffice_Level3) [4](#_Toc22481_WPSOffice_Level3)

[2.2 JavaScript语法](#_Toc10197_WPSOffice_Level2) [5](#_Toc10197_WPSOffice_Level2)

[2.2.1 变量](#_Toc17201_WPSOffice_Level3) [5](#_Toc17201_WPSOffice_Level3)

[2.2.2 注释和分隔符](#_Toc27374_WPSOffice_Level3) [5](#_Toc27374_WPSOffice_Level3)

[2.2.3 单双引号](#_Toc19150_WPSOffice_Level3) [5](#_Toc19150_WPSOffice_Level3)

[2.2.4 JavaScript数据类型](#_Toc10197_WPSOffice_Level3) [5](#_Toc10197_WPSOffice_Level3)

[2.2.5 数据类型转换](#_Toc21750_WPSOffice_Level3) [6](#_Toc21750_WPSOffice_Level3)

[2.2.6 运算符](#_Toc4707_WPSOffice_Level3) [7](#_Toc4707_WPSOffice_Level3)

[2.2.7 流程控制](#_Toc5982_WPSOffice_Level3) [8](#_Toc5982_WPSOffice_Level3)

[2.2.8 循环结构](#_Toc28053_WPSOffice_Level3) [8](#_Toc28053_WPSOffice_Level3)

[2.2.9 函数](#_Toc9383_WPSOffice_Level3) [8](#_Toc9383_WPSOffice_Level3)

[2.2.10对象](#_Toc22207_WPSOffice_Level3) [9](#_Toc22207_WPSOffice_Level3)

[2.3 JavaScript内置对象](#_Toc21750_WPSOffice_Level2) [9](#_Toc21750_WPSOffice_Level2)

[2.3.1 String](#_Toc27969_WPSOffice_Level3) [9](#_Toc27969_WPSOffice_Level3)

[2.3.2 Array](#_Toc19279_WPSOffice_Level3) [10](#_Toc19279_WPSOffice_Level3)

[2.3.3 RegExp](#_Toc5270_WPSOffice_Level3) [11](#_Toc5270_WPSOffice_Level3)

[2.3.4 Date](#_Toc30168_WPSOffice_Level3) [11](#_Toc30168_WPSOffice_Level3)

[2.3.5 Math](#_Toc9344_WPSOffice_Level3) [12](#_Toc9344_WPSOffice_Level3)

[2.4 事件](#_Toc4707_WPSOffice_Level2) [13](#_Toc4707_WPSOffice_Level2)

[2.4.1 定义与要素](#_Toc17460_WPSOffice_Level3) [13](#_Toc17460_WPSOffice_Level3)

[2.4.2 事件前](#_Toc14660_WPSOffice_Level3) [13](#_Toc14660_WPSOffice_Level3)

[2.4.3事件分类](#_Toc6144_WPSOffice_Level3) [13](#_Toc6144_WPSOffice_Level3)

[2.4.4 其他事件](#_Toc21173_WPSOffice_Level3) [15](#_Toc21173_WPSOffice_Level3)

[2.5 BOM](#_Toc5982_WPSOffice_Level2) [16](#_Toc5982_WPSOffice_Level2)

[2.5.1 简介](#_Toc11208_WPSOffice_Level3) [16](#_Toc11208_WPSOffice_Level3)

[2.5.2 BOM的子对象](#_Toc6958_WPSOffice_Level3) [16](#_Toc6958_WPSOffice_Level3)

[2.5.3 BOM基本方法](#_Toc23179_WPSOffice_Level3) [17](#_Toc23179_WPSOffice_Level3)

[2.6 DOM](#_Toc28053_WPSOffice_Level2) [19](#_Toc28053_WPSOffice_Level2)

[2.6.1简介](#_Toc15226_WPSOffice_Level3) [19](#_Toc15226_WPSOffice_Level3)

[2.6.2 查找元素](#_Toc9355_WPSOffice_Level3) [19](#_Toc9355_WPSOffice_Level3)

[2.6.3 操作元素](#_Toc19177_WPSOffice_Level3) [20](#_Toc19177_WPSOffice_Level3)

[2.6.4 XML](#_Toc25483_WPSOffice_Level3) [22](#_Toc25483_WPSOffice_Level3)

[2.7 AJAX](#_Toc9383_WPSOffice_Level2) [23](#_Toc9383_WPSOffice_Level2)

[2.7.1 简介](#_Toc9506_WPSOffice_Level3) [23](#_Toc9506_WPSOffice_Level3)

[2.7.2 流程](#_Toc29145_WPSOffice_Level3) [24](#_Toc29145_WPSOffice_Level3)

[第3章 jQuery](#_Toc22481_WPSOffice_Level1) [25](#_Toc22481_WPSOffice_Level1)

[3.1 简介](#_Toc22207_WPSOffice_Level2) [25](#_Toc22207_WPSOffice_Level2)

[3.1.1什么是jQuery](#_Toc31601_WPSOffice_Level3) [25](#_Toc31601_WPSOffice_Level3)

[3.1.2 jQuery的作用](#_Toc25310_WPSOffice_Level3) [25](#_Toc25310_WPSOffice_Level3)

[3.1.3配置](#_Toc26023_WPSOffice_Level3) [25](#_Toc26023_WPSOffice_Level3)

[3.2 应用](#_Toc27969_WPSOffice_Level2) [26](#_Toc27969_WPSOffice_Level2)

[3.2.1查找元素](#_Toc8463_WPSOffice_Level3) [26](#_Toc8463_WPSOffice_Level3)

[3.2.2元素操作](#_Toc5823_WPSOffice_Level3) [27](#_Toc5823_WPSOffice_Level3)

[3.2.3节点操作](#_Toc9323_WPSOffice_Level3) [27](#_Toc9323_WPSOffice_Level3)

[3.2.3事件](#_Toc15332_WPSOffice_Level3) [28](#_Toc15332_WPSOffice_Level3)

[3.2.4 Ajax](#_Toc19951_WPSOffice_Level3) [29](#_Toc19951_WPSOffice_Level3)

[3.2.5效果](#_Toc153_WPSOffice_Level3) [29](#_Toc153_WPSOffice_Level3)

[第4章 Bootstrap](#_Toc17201_WPSOffice_Level1) [30](#_Toc17201_WPSOffice_Level1)

[4.1简介](#_Toc19279_WPSOffice_Level2) [30](#_Toc19279_WPSOffice_Level2)

[4.2配置](#_Toc5270_WPSOffice_Level2) [30](#_Toc5270_WPSOffice_Level2)

[4.2.1 远程访问](#_Toc6056_WPSOffice_Level3) [30](#_Toc6056_WPSOffice_Level3)

[4.2.2 本地访问](#_Toc18692_WPSOffice_Level3) [30](#_Toc18692_WPSOffice_Level3)

[4.3使用](#_Toc30168_WPSOffice_Level2) [31](#_Toc30168_WPSOffice_Level2)

[4.3.1 样式](#_Toc1606_WPSOffice_Level3) [31](#_Toc1606_WPSOffice_Level3)

# 

# 第1章JavaScript简介

## 1.1 JavaScript介绍

* javascript 是一种客户端脚本语言(脚本语言是一种轻量级的编程语言)
* JavaScript通常被直接嵌入HTML页面
* JavaScript是一种解释性语言(代码执行不进行预编译)

## 1.2 JavaScript特点

* 跨平台
* 安全性好
* 兼容性
* JavaScript是基于对象的脚本语言

## 1.3 JavaScript核心

JavaScript语言中包含三个核心：ECMAScript基本语言、DOM、BOM

## 1.4 JavaScript作用

* 特效
* 表单验证
* Ajax

# 第2章JavaScript基础

## 2.1 HTML中使用方式

### 2.1.1 script标签

* <script></script>
* 【属性】

charset(可选)：字符集设置

defer(可选)：

src(可选)：使用外部的JS脚本文件路径

type：text/javascript (html4时必须加上，html5可不加)

* 【例】

<script>

alert(‘标签使用’);

</script>

### 2.1.2外部导入方式 (推荐)

* <script type=”text/javascript” src=”1.js”></script>

## 2.2 JavaScript语法

### 2.2.1 变量

* 【声明】
* 字母数字下划线
* 首字母不能为数字
* 严格区分大小写
* 不能使用关键字
* 【定义】

使用var关键字来声明

### 2.2.2 注释和分隔符

* 单行注释 //
* 多行注释 /\*\*/
* 分隔符 ;

### 2.2.3 单双引号

与Php单双引号的使用不相同。Js中单双引号可以互相嵌套使用，而Php与之不同。

* 单引号不能嵌套单引号
* 双引号不能嵌套双引号
* 单引号可以嵌套双引号
* 双引号也可以嵌套单引号

【解析能力】

两者都可以解析转义字符

两者都没有解析变量的能力

【注意】

1. 在+号连接符下除外。(Js中+号即使运算符也是连接符)
2. 在js中使用字符串的时候，不管是单双引号都不能随意换行。如果非要换行要不加上转义字符，要不使用反引号``来解析该字符串。

### 2.2.4 JavaScript数据类型

在Js中分为6种数据类型：

* undefined 类型
* function 类型
* boolean类型
* string类型
* number类型
* object类型(3类)

1.new Object 2.new Array 3.null

* 【注意】

1. NaN(not a number)属于number类型

1.任何一个数值跟NaN进行运算都是NaN

2.任何数值跟NaN进行比较的时候返回的结果都是false

3.检测一个值是否为NaN的时候只能使用isNaN函数

1. boolean 类型的值必须要小写
2. 在Js中boolean类型有7个值：0 ,‘’,0.0,false,null,NaN,undefined

而Php中布尔类型：0,’’,’0’,0.0,false,null,array

### 2.2.5 数据类型转换

* 【自动】Js中在计算和if判断的时候系统会自动转换数据类型

Number：在字符串与number之间用加号(+)则是连接符。

* 【举例】

var a = '100' + 20; //string  
var a = 10 + 10 + '30'; //2030  
var a = '10' + 10 +20; //101020  
var a = '300a' + 1; //300a1  
var a ='100'-50; //50  
var a = '100'/2; //50  
var a = '100' \*2; //200

* 【强行】

parseFloat与parseInt相似：

1.如果是纯数字的话，则直接返回数字类型的值

2如果首字母是数字，则截取到第一个不是数字之前的数字返回

3.如果首字母不是数字，返回NaN

Number:

1.纯数字转化成数字，如果中间有非数字则直接返回NaN

String：转出成字符串

Boolean：转换布尔值(0 0.0 ‘’ false null undefined NaN)

* 【举例】

1. parseInt

var s = 'anb30'; //NaN  
var s = '100.2ac';//100  
var s = '100abc'+20; //100  
var a = parseInt(s);

1. parseFloat

var f = 'asda3.14';//NaN  
var f = '200abc'  
var a = parseFloat(f);

1. Number

var n = '500zbc'; //NaN  
var n = '202.2aaa'; //NaN

var a = Number(n);

1. Boolean

var b = new Array; //true  
var b = function(){} //true  
var a = Boolean(b);

* 【检测数据类型】

typeof() ->在Js中检测数据类型

### 2.2.6 运算符

* 【算术运算符】

+ - \* / ++ --

* 【字符串链接】

+：在js中+号即使连接符又是运算符。

字串连接和数值求和，就是加号”+“两侧都是数值则求和，否则做字串连接

* 【赋值运算】

= += -= %=

* 【比较运算符】

< > >= <= == === != !==

* 【逻辑运算符】

逻辑 NOT ! 运算符 非

逻辑 AND && 运算符 与

逻辑 OR || 运算符 或

* 【位运算】

^ & | << >>

* 【其他运算符】

1. ? : 三元运算符
2. delete：用于删除对象中属性的

如：delete o.name; //删除o对象中的name属性

1. void：void 运算符对任何值返回 undefined

没有返回值的函数真正返回的都是 undefined

4.逗号运算符

用逗号运算符可以在一条语句中执行多个运算。

var iNum1=1, iNum2=2, iNum3=3;

### 2.2.7 流程控制

流程控制用于基于不同的条件来执行不同的动作。

1.if(){} else if(){}else{} 注意：else和if中间必须要用空格

2.switch-> case brack default

### 2.2.8 循环结构

程序中进行有规律的重复性操作，需要用到循环语句。

while

do...while

for

for in 语句是严格迭代语句，用于列举对象的属性

break 和 continue 语句对循环中的代码执行提供了更严格的控制。

with语句用于设置代码在特定对象中的作用域

* 【举例】

1.for -in

var a = [10,20,30,40,50];

//迭代的是数组的下标。

for(i in a){

document.write(a[i]);

}

//输出： 1020304050

2.with简化操作  
with(document){

wtith(‘test1’);

wtith(‘test2’);

wtith(‘test3’);

wtith(‘test4’);

}

### 2.2.9 函数

* 【声明】Js中函数的声明有以下三种

1.普通定义：function demo(){}

2.变量名定义：var demo = function(){}

3.var 函数名 = new Function(“参数1”,”参数2”,”函数体”);

var demo = new Function(‘x’,’y’,’alert(x+y)’);

* 【调用】Js中函数的调用也分为三种

在代码中直接调用

事件触发

赋值给一个变量

* 【参数】Js中函数的参数需要注意以下几点

1.在声明函数的时候不要给参数设置初始值(IE浏览器有问题)

2.调用函数的参数个数多于函数声明的参数个数，多于的参数被忽略

3.调用函数的参数个数小于函数声明时的参数个数，那么没有传递的参数默认是undefined

4.在函数代码中，使用特殊对象arguments[] 就能直接访问参数

* 【返回值】Js函数的返回值和Php中不一样

1.如果不使用return返回一个值的话那么函数的默认返回值是undefined

2.可以返回一个函数

* 【作用域】Js中关于变量和函数的作用域需要注意

1.函数外面定义的变量是全局变量，函数内可以直接使用。

2.在函数内部没有使用var定义的变量则为全局变量

### 2.2.10对象

* 【声明】Js中对象的三种声明方法

1.使用原始的方式创建内置对象 var myobject = new Object()

2.直接创建自定义对象 var 对象名 = {}

3.使用自定义构造函数创建对象 function函数名(参数){成员属性，成员方法} var pen = new 函数名();

* 【this】关键字

this单词本身就是 这个 的意思

在对象的方法中使用,代表着当前这个对象

意味着当对象调用这个方法时,方法中的this就代表着这个对象

* 【遍历】 使用for in 遍历或使用with简单操作

## 2.3 JavaScript内置对象

### 2.3.1 String

* 【属性】

String属性->length

* 【方法】

trim() 清除两侧的空白

charAt() 获取字符串指定位置上的字符 (索引从0开始)

indexOf() 获取指定字符或者字符串首次出现的位置

lastindexOf() 获取指定字符或者字符串最后一次出现的位置

concat() 连接两个或者多个字符串

slice() 截取字符串 (从下标0开始到结束 要前不要后)

split() 按照指定的字符或者字符串来拆分成字符串数组

substr() 截取字符串(从下标0开始到结束)

search() 检索与正则表达式相匹配的值

match() 找到一个或多个正则表达式的匹配

toLowerCase() 将字符串转化成小写

toUpperCase() 将字符串转化成大写

replace() 替换一个与正则表达式匹配的字符串

### 2.3.2 Array

* 【声明】Array 声明方式二种

1.new Array()

2.[]

* 【属性】

Array常用属性 length

* 【遍历】

for：只能遍历数组里面的元素

for - in：既可以遍历数组里面的元素也可以遍历数组对象的属性

* 【注意】

1.不支持[空]形式的添加成员

2.不支持关联数组，只支持索引数组

3.二维数组调用时需要先声明再调用

4.数组的索引是连续的

* 【举例】例子中有正确的也有常见错误。

Array定义：

var array = new Array(3); // 不是一个元素3 而是3个元素 元素值为undefined  
var array = new Array(3,4,5); //代表 当前数组里面有3个元素 分别是3,4,5

Array改：

var array = [null,undefined,function(){},true,'255',10,20,50];  
array[0] = 100; //将null替换掉

Array增：

array[] = 200; //这种弄定义时错误的 不能以[]往数组里面添加值 不写下标不行

array['name'] = 'www'; // 这种方式定义的不是数组的元素，而是数组对象的属性

Array 删：

delete(array[1]);  
array.pop(); //从数组的尾部删除元素

* 【方法】Array常用方法

concat() 连接一个或者多个数组

join() 将数组元素连接起来转化成一个字符串

reverse() 翻转数组元素

slice() 返回数组的一部分

sort() 对数组元素进行排序

splice() 插入、删除或替换数组的元素

push() 从数组的尾部添加一个元素

pop() 将数组尾部的元素删除

shift() 将数组的头部删除一个元素

unshift() 从数组的头部添加一个元素

### 2.3.3 RegExp

RegExp 正则表达式使用单个字符串来描述，匹配一系类符合某个句法规则的字符串，正则表达式通常被用来检索、替换那些符合某个模式的文本。

(该文本值做Js正则的整理其他为整理的转Php功能笔记)

* 【声明】Js中正则的声明方式有两种

1. var reg = new RegExp();
2. / /

* 【分隔符】

/ / (PHP的分割符一般是 ’//’ )

* 【修饰符】

JavaScript中只有四个模式修饰符：g、i、m、y。

这里，i标识也是忽略大小写的作用。

* 【方法】

test() 检测 返回的是布尔值

exec() 执行 返回的是数组

### 2.3.4 Date

* 【声明】Js中声明时间的方法分声明当前时间和声明指定时间

当前时间：var date = new Date();

指定时间：var date = new Date(‘2018-8-25 21:10:00’);

* 【方法】

getFullYear() 获取年份

getMouth() 获取月份 注意:获取的当前月份要+1，因为系统从0开始计算

getDate() 获取天日

getHours() 获取小时

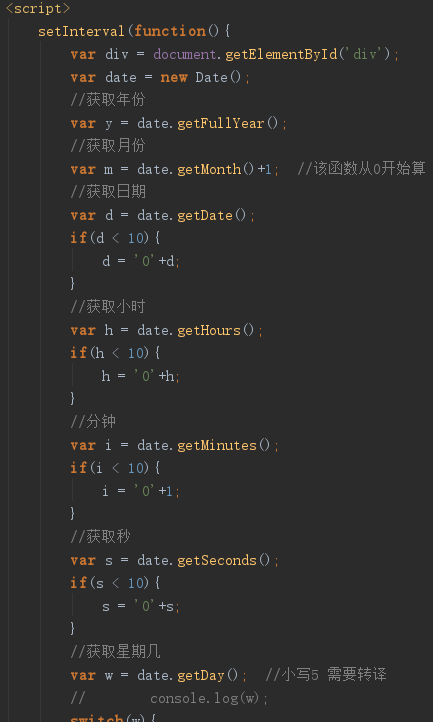
getMinutes() 获取分钟

getSeconds() 获取秒

getDay() 获取星期几 0-6 对应的是星期日到星期六

* 【举例】

//动态时间

### 2.3.5 Math

Math 不用声明定义，直接调用。

* 【方法】

round() 浮点数四舍五入

max() 获取最大值

min() 获取最小值

abs() 绝对值

floor() 舍一法

ceil() 进一法

pow() 获取N次幂

sqrt() 求平方根

random() 获取0-1之间的随机数

* 【random的算法】

//获取16-25 之间的随机数

function rand(m,n){  
 ///10+15  
 return Math.ceil(Math.random() \* (n-m+1)+(m-1));  
}  
console.log(rand(16,25));

## 2.4 事件

### 2.4.1 定义与要素

* 【定义】

当对文档中的元素进行操作的时候或者文档的状态发生改变的时候就会产生一个事件

* 【要素】

1. 事件源 2. 事件 3. 事件处理程序

### 2.4.2 事件前

* 【style】

1. 只能对行内样式进行绑定
2. 有横杠的css样式要使用驼峰法命名

* 【元素信息】

offsetTop 当前元素距离顶部的偏移量

offsetLeft 当前元素距离左部的偏移量

offsetWidth 获取当前元素的宽

offsetHeight 获取当前元素的高

* 【绑定事件】

1. 在html标签元素中 属性名=’属性值’

<div onclick=’func()’></div>

1. 使用元素对象的方式

lis.onclick = function(){

alert(‘JavaScript’);

}

1. addEventListener(非IE浏览器 或者在IE9以上使用)

lis.addEventListener(‘click’,function(){ });

### 2.4.3事件分类

* 【文档】

onload 文档事件，事件会在页面或者图像加载完成后立即生效。

onbeforeunload 退出浏览器前的提示（防止误操作）

* 【鼠标】

onclick 单击事件 阻止默认行为 return false (a标签中先阻止默认在改变)

ondblclick 双击事件

oncontextmenu 右键事件

onmouseover 鼠标放上去的事件

onmouseout 鼠标离开的事件

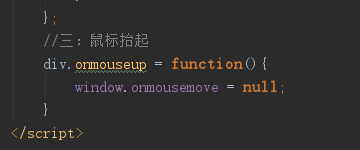
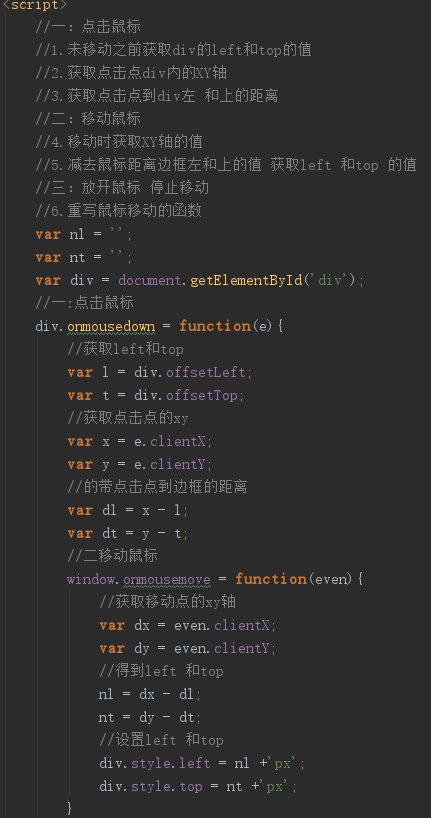
onmousemove 鼠标移动的事件

onmousedown 鼠标按下的事件

onmouseup 鼠标抬起事件

* 【鼠标实例】

//Js 操作鼠标移动div



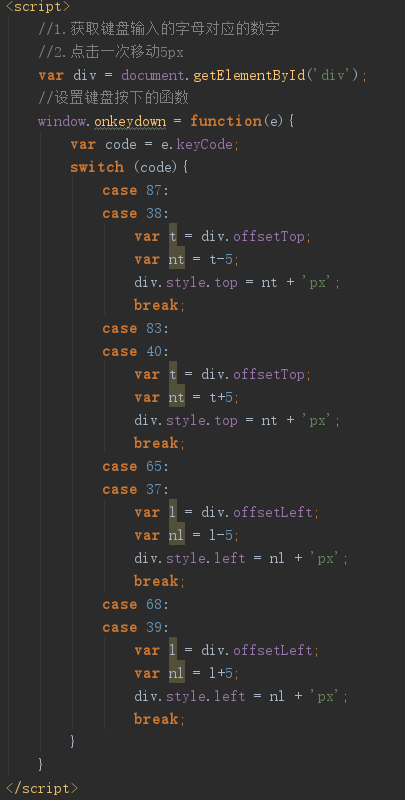
* 【键盘】

onkeydown 键盘按下

onkeyup 键盘抬起

* 【键盘实例】

//Js 操作键盘移动div

注意：参数e有好多可用的对象属性

* 【表单】

onfocus 获取表单焦点

onblur 丧失表单焦点

onchange 值在发生改变的时候触发 如:select 默认选择更改时

onselect 只能选中input输入框或者textarea文本域内容的时候触发

onsubmit 提交按钮事件

onreset 重置按钮事件

### 2.4.4 其他事件

* 【阻止冒泡】终止事件在传播过程的捕获、目标处理或起泡阶段进一步传播。调用该方法后，该节点上处理该时间的处理程序将被调用，时间不在被分派到其他节点

stopPropagation() 使用e(参数)对象里面的方法进行阻止

* 【实例】

<div id="dvs">  
 <div id="ine"></div>  
 </div>  
 <script>  
 var div = document.getElementsByTagName('div');  
 for (var i=0;i<div.length;i++){  
 div[i].onclick = function(e){  
 this.style.background = 'red';  
 //阻止点击 里面的div 外面的div也跟着触发 该行为叫阻止冒泡行为  
 e.stopPropagation();  
 }  
 }  
 </script>

* 【扩展】window.getComputedStyle()

getComputedStyle是一个可以获取当前元素所有最终使用的css属性值。返回的一个css样式声明对象

* 【实例】

var w = window.getComputedStyle(div[0]);  
console.log(w.width);

## 2.5 BOM

### 2.5.1 简介

* BOM是browser object model的缩写，简称浏览器对象模型
* BOM提供了独立于内容而与浏览器窗口进行交互的对象
* 由于BOM主要用于管理窗口与窗口之间的通讯，因此其核心对象是window
* BOM由一系列相关的对象构成，并且每个对象都提供了很多方法与属性
* BOM缺乏标准，JavaScript语法的标准化组织是ECMA，DOM的标准化组织是W3C
* BOM最初是Netscape浏览器标准的一部分

### 2.5.2 BOM的子对象

* location对象
* 【属性】

hash 设置或返回从井号 (#) 开始的

URL（锚）

host 设置或返回主机名和当前 URL 的端口号

hostname 设置或返回当前 URL 的主机名

href 设置或返回完整的 URL

pathname 设置或返回当前 URL 的路径部分

port 设置或返回当前 URL 的端口号

protocol 设置或返回当前 URL 的协议

search 设置或返回从问号 (?) 开始的 URL（查询部分）

* 【方法】

页面跳转:

location.href = ‘http://www.baidu.com’

location.replace(‘http://www.baidu.com’);

页面刷新：

location.reload(); 普通刷新

location.reload(true); 强行刷新

* screen对象
* 【属性】

screen.width 获取屏幕的宽度

screen.height 获取屏幕的高度

screen.availWidth 获取屏幕的宽度(不包含任务栏)

screen.availHeight 获取屏幕的高度(不包含任务栏)

* 【页面尺寸】

document.documentElement.clientWidth 获取浏览器可视区域的宽度

document.documentElement.clientHeight 获取浏览器可视区域的高度

document.documentElement.scrollWidth 获取浏览器文档的全文宽

document.documentElement.scrollHeight 获取浏览器文档的全文高

document.documentElement.scrollTop 获取滚动条距离顶部的偏移量(隐藏域)

document.documentElement.scrollLeft 获取滚动条距离左侧的偏移量(隐藏域)

* history对象
* 【属性】 length
* 【方法】

history.back() 后退

history.forward() 前进

history.go() 后退/前进 (-1/1，也可其他数字)

### 2.5.3 BOM基本方法

* 【方法框】
* alert() 提示框
* confirm() 确定框 返回值布尔值
* prompt() 获取用户输入的信息
* 【方法窗口】
* open() 打开新窗口
* 【参数】

1.fullscreen:yes或者no，表示浏览器窗口是否最大化，仅限IE

2.height：数值，新窗口的高度，不能小于100。

3.left：数值，新窗口的左坐标，不能是负值。

4.location：yes或no，是否在浏览器中显示地址栏，如果设置为no，浏览器有可能会隐藏地址栏或禁用地址栏

5.menubar：yes或no，是否在浏览器窗口中显示菜单栏，默认为no

6.resizable：yes或no，表示是否通过拖动浏览器边框改变其大小，默认为no（IE中有效）。

7.scrollbars：yes或no，表示如果内容在视口中显示不下，是否允许滚动。默认为no

8.status：yes或no，表示是否在浏览器窗口中显示状态栏，默认为no

9.toolbar：yes或no，表示是否在浏览器窗口中显示工具栏。默认为no

10.top：数值，表示新窗口的上坐标。不能是负值。

11.width：数值，表示新窗口的宽度。不能小于100。

* close() 关闭新窗口
* 【定时器】
* setTimeout() 按照指定的毫秒数后调用，只调用一次
* 【实例】

var buts = document.getElementById('buts');  
buts.onclick = function(){  
 into = setTimeout(function(){  
 alert('开始');  
 },3000);  
};

* setInterval() 按照指定的周期调用 定时循环
* 【注意】

1. clearTimeout() 停止setTimeout的定时器
2. clearInterval() 停止setInterval 的定时器
3. 不管那种定时器声明的方法都是有两种的。参数1既可以直接方法函数也可以方法函数名
4. 定时器中1000是1秒

* 【实例】

//制作一个定时器20秒后可以操作 显示秒数  
var bs = document.getElementById('bs');  
function foo(){  
 //获取倒计时的文本  
 var i = bs.innerHTML;  
 i--;  
 bs.innerHTML = i;  
 if(i == 0){  
 bs.style.cursor = 'pointer';  
 //停止定时器  
 clearInterval(info);  
 bs.innerHTML = '关闭';  
 bs.disabled = false;  
 }  
}  
var info = setInterval(foo,1000);

* 【IE】以下方法只在IE浏览器中使用
* moveBy() 浏览安装指定的像素移动坐标位置
* moveTo() 把窗口的左上角移动到一个指定的位置
* resizeTo() 把窗口的大小调到指定的宽和高
* resizeBy() 按照指定的像素调整窗口的大小
* 【滚动条】
* scrollTo() 把文档内容滚动到指定的坐标
* scrollBy() 按照指定的像素值来滚动内容
* 【事件】
* onresize 事件会在框架被调整大小的时候触发
* onscroll 元素滚动时执行JavaScript
* 【补充】

多数的网站都不是一次性的将网页文档页面的内容全部呈现到页面上，而是用户在观看的浏览器的可视区域底部时在提取一部分，然后循环渐进的来提取文档。这样大大减轻了服务器的压力。

## 2.6 DOM

### 2.6.1简介

HTML文档中的每个元素都是一个节点 , 无论是标签、标签属性、文本、注释甚至空白都算是一个节点。整个文档是一个文档节点，每个 HTML 元素是元素节点，HTML 元素内的文本是文本节点，每个 HTML 属性是属性节点，注释是注释节点。

### 2.6.2 查找元素

* 【通用方法】
* document.getElementById() 通过id属性来查找元素
* document.getElementsByTagName() 通过标签名来查找元素
* document.getElementsByClassName() 通过类名来查找元素
* document.getElementsByName() 通过name属性查找元素
* 【注意】

1. 通过class、name、tagname类名获取的元素，返回的结果是数组对象
2. 遍历这些数组对象时要注意 元素[] 和this的用法，分清this此时的代表

* 【实例】

//通过class 类名获取 返回的结果是数组对象  
var lis = document.getElementsByClassName('w');  
for(var i=0;i<lis.length;i++){  
 lis[i].onclick = function(){  
 this.style.background = 'orange'  
 }  
}  
//通过name值 返回的也是数组  
var uname = document.getElementsByName('y');  
for(var i=0;i<uname.length;i++){  
 uname[i].onmouseover = function(){  
 this.style.background = 'green';  
 }  
}

* 【父子关系节点】 前加上获取的元素名
* childNodes 获取所有的子节点(包括文本点和注释节点)
* children 获取所有的子元素节点
* firstChild 第一个节点
* firstElementChild 获取第一个元素节点
* lastChild 获取最后一个节点
* lastElementChild 获取最后一个元素节点
* previousSibling 获取上一个节点
* previousElementSibling 获取上一个元素节点
* nextSibling 获取下一个节点
* nextElementSibling 获取下一个元素节点
* parentNode 获取父级节点
* parentElement 获取父级元素节点
* 【特殊节点】
* document.doctype 获取文档头
* document.documentElement 获取html标签
* document.head 获取head标签元素
* document.title 获取页面的标题
* docuemnt.body 获取body元素
* 【节点集合】
* document.forms 获取所有的表单元素
* elements 获取input元素
* name 属性值 获取元素
* document.links 获取所有的a标签
* document.images 获取所有的img标签元素

### 2.6.3 操作元素

* 【修改元素】
* 修改节点属性

setAttribute() 设置属性名和属性值

getAttribute() 获取属性值

removeAttribute() 移除属性值

* 修改元素文本

innerHTML 设置和获取标签中间的文本

textContent 设置和获取标签中的纯文本

* 俩者的区别：

在设置时innerHTML有解析标签的能力，而textContent直接回把标签当文本显示。

在获取的时innerHTML 获取之间所有的包括标签，而textContent只获取文本信息。

* 修改元素样式

style

className 设置类名

* 设置form控件的值

form各种标签的获取和设置。这里那有下拉框来举例

//下拉框

var city = document.getElementsByName('city[]')[0];  
console.log(city.value);  
var ops = city.children; //获取子类  
for(var i =0;i<ops.length;i++){  
 var op = ops[i].selected;  
 if(op){  
 console.log(ops[i].textContent)  
 }  
}

* 节点操作
* 创建节点 document.createElement()
* 插入节点 appendChild() 从尾部插入 | insertBefore() 从 头部插入
* 删除节点 removeChild() 删除节点
* 替换节点 replaceChild() 节点替换
* 克隆节点 cloneNode(true) 克隆复制节点
* 【实例】

### 2.6.4 XML

* 【简介】

Extensible Markup Language 可扩展的文本标记语言。

1.整个文档是一个文档节点

2.每个 XML 标签是一个元素节点

3.包含在 XML 元素中的文本是文本节点

4.每一个 XML 属性是一个属性节点

5.注释属于注释节点

* 【作用/注意】

XML的主要作用是为了信息传递

XML 文档必须有根元素

所有 XML 元素都须有关闭标签

XML 标签对大小写敏感

XML 必须正确地嵌套

XML 的属性值须加引号

* 【js解析】
* 文件：

//IE

var xmlDoc=new ActiveXObject("Microsoft.XMLDOM");

xmlDoc.async=false;

xmlDoc.load("note.xml");

//非IE

var xmlDoc=document.implementation.createDocument("","",null);

xmlDoc.async=false;

xmlDoc.load("note.xml");

* 字符串：

if (window.DOMParser) { //非IE

var parser=new DOMParser();

var xmlDoc=parser.parseFromString(txt,"text/xml");

} else { //IE

var xmlDoc=new ActiveXObject("Microsoft.XMLDOM");

xmlDoc.async=false;

xmlDoc.loadXML(txt);

}

* 【实例】

<?xml version='1.0' encoding='utf-8'?>

<movies>

<movie>

<name>战狼</name>

<country>中国</country>

</movie>

<movie>

<name>速度与激情8</name>

<country>美国</country>

</movie>

<movie>

<name>传奇</name>

<country>美国</country>

</movie>

</movies>

## 2.7 AJAX

### 2.7.1 简介

* 【定义】

Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）

AJAX 不是新的编程语言，而是一种使用现有标准的新方法。就是页面无刷新请求服务器。

* 【作用】

AJAX是在不重新加载整个页眉的情况下与服务器交换数据并更新部分网页的技术(实现浏览器与服务器之间数据交互，实现页面的无刷新请求服务器，提高用户体验)实现局部刷新，但不是真的局部刷新。

{在不重新载入整个页面的情况下。Ajax请求web服务器并实现数据交互，通过JavaScript节点操作体验页面一部分的无刷新浏览。}

### 2.7.2 流程

* 【创建对象】
* 所有现代浏览器(IE7+、Firefox、Chrome、Safari以及Opera)均用以下方式

var ajax = new XMLHttpRequest();

* 老版本IE5和IE6使用以下方式

var ajax = new ActiveXobject(“Microsoft.XMLHTTP”);

* 【创建事件】创建事件函数(处理服务器返回的结果)

ajax.onreadystatechange = function(){}

* readyState ajax对象的状态码

0: 请求未初始化

1: 服务器连接已建立

2: 请求已接收

3: 请求处理中

4: 请求已完成，且响应已就绪

* status ajax的响应码

200: "OK"

404: 未找到页面

403: 权限不足

* responseText 接受服务器返回的数据
* 【初始化请求】 ajax.open()
* 三个参数：参数1：请求方式(GET/POST) 参数2：url地址(框架中是路由)参数3：同步异步 (true 异步(默认)，false同步)
* 【注意】

如果采用post请求方式，则需要使用setRequestHeader() 来添加一个HTTP头。然后在send() 方法中规定要发送的数据

ajax.setRequestHeader("Content-type","application/x-www-form-urlencoded");

* 【发送请求】ajax.send()
* 【实例】

<button id="buts">ajax的基本使用</button>  
<script>  
 var buts = document.getElementById('buts');  
 buts.onclick = function(){  
 //发送ajax  
 //1.创建ajax对象  
 var ajax = new XMLHttpRequest;  
 //2.创建事件 状态发生改变的事件  
 ajax.onreadystatechange = function(){  
 if(ajax.readyState == 4 && ajax.status ==200){  
 //接受服务器返回来的结果  
 console.log(ajax.responseText);  
 }  
 };  
 //3.初始化  
 //参数1.请求方式(method) 2.url 地址 5.1框架(路由) 3.同步异步 true 异步 false同步  
 ajax.open('GET','/home/ajax',true);  
 //4.ajax 发送  
 ajax.send();  
 };  
</script>

* 【注意】

1. 服务器返回给控制器的是数组，而在返回给ajax的时候需要返回字符串，此时就需要用到json\_encode($array){框架中用json}转译成字符串，在返回给ajax后还需要用eval(‘var array =’+ajax.responseText)字符串在转译成数组。
2. 关联数组使用evel转译后为 对象{name:’11’,age:’20’...}
3. 索引数组使用evel转译后是 数组对象 [‘apple’,’lemon’...]
4. 二维数组使用eval转译后是 外面是数组，里面是对象 [{},{}...]
5. 同源策略：当前页面要和服务器的协议地址和端口都要相同
6. Ajax中接收不到return返回回来的数据(框架中可以)

# 第3章 jQuery

## 简介

### 3.1.1什么是jQuery

是一个免费、开源的javascript库。

### 3.1.2 jQuery的作用

方便的完成web全端的相关操作，例如节点操作，元素操作，时事件绑定，ajax操作，且解决了大多数的兼容性问题。

### 3.1.3配置

* 【本地配置】

1. 下载jquery文件(版本)
2. <script src="jquery-3.2.1.min.js"></script> (min.js为纯净版)

* 【远程访问】

1. <script src="http://www.(有jquery的网站).com/jquery.js"></script>

* 【检测配置】

1. alert($) $是jQuery对象的简写
2. alery(jQuery)

* 【注意】

1.jQuery 中的Q严格大写。

2.jQuery 中变量函数有两种方法(1.$().each() 2.$.each();)

3.jQuery 中each()来遍历数组

## 应用

### 3.2.1查找元素

* 【基本获取】

1. 通过id查找元素 $(‘#id’)
2. 通过class属性获取 $(‘.class’)
3. 通过标签名或者元素 $(‘div’)
4. 逗号并列获取 $(‘#id,.class’)
5. 空格层级获取 $(‘#id .class’)

* 【过滤获取】

1. 获取第一个元素 $(‘li:first’)
2. 获取最后一个元素 $(‘li:last’)
3. 获取指定索引的元素 $(‘li:eq(5)’) / $(‘li’):eq(5)
4. 获取包含指定文本的元素$(‘li:contains(“国”)’)
5. 通过属性查找元素 单个：$(‘input[name=uname]’)

多个：$(‘input[type=text][name=uname]’)

1. 通过奇偶(索引)数来查找元素 偶数:$(‘ul li:exen’) / 奇数:$(‘ul li:odd’)
2. lt 从0(索引)开始匹配所有小于参数的值(不包括参数) $(‘ul li:lt(5)’)
3. gt从0(索引)开始匹配所有大于参数的值(不包括参数) $(‘ul li:gt(5)’)

* 【父子获取】

1. 获取所有的子元素 $(‘ul’).children()
2. 获取第一个子元素 $(‘ul li:first-child’)
3. 获取最后一个子元素 $(‘ul li:last-child’)
4. 获取指定顺序(1开始，不是索引)第几个元素 $(‘ul li:nth-child(1)’)
5. 获取下一个元素 $(‘#id’).next()
6. 获取上一个元素 $(‘#id’).prev()
7. 获取父级元素 $(‘#id’).parent()
8. 获取先辈元素 $(‘#id’).parents(‘tr’)
9. 获取同辈元素 $(‘#id’).siblings()
10. 在父级元素中查找元素 $(‘ul’).find(‘.class’)

* 【其他】

1. 表单类型 :text :radio :checkbox :submit :image :file 等
2. 表单对象属性 :checked :selected :

* 【注意】

1. jquery的获取和css 获取一致
2. 使用JavaScript基本步骤就是1.查找2.操作
3. 获取元素的方法还有很多，该文档不可能全部罗列完整，所有还是要常看手册

### 3.2.2元素操作

* 【属性操作】

1. attr() 两个参数：添加属性 一个参数：获取属性值
2. removeAttr() 移出该属性

* 【样式操作】

1. css() 单个添加属性名和属性值之间用逗号隔开 多个添加采用对象{}属性名和属性值之间用冒号分开，参数与参数之间用哪个逗号分开。
2. addclass() 添加类
3. removeClass()移出类

* 【文本操作】

1. text() 纯文本
2. html() 解析标签

* 【form操作】

1. 获取 $(‘input[name=uname]’).val()
2. 设置 $(‘input[name=uname]’).val(‘设置值’)

* 【其他操作】

1. 获取相对于文档的偏移量 offset()
2. 获取元素相对于父级的偏移量 position()
3. 获取可视区域的宽度 $(window).width()
4. 获取整个文档的宽度 $(‘document’).width()
5. 获取元素本身的宽度 $(‘#id’).width()
6. 获取可是区域的高度 $(window).height()
7. 获取整个文档的高度 $(‘document’).height()
8. 获取元素本身的高度 $(‘#id’).height()
9. 返回相应的索引值 index()

### 3.2.3节点操作

* 【创建】

1. $(‘<div></div>’)
2. $(‘<div class=’cur’><span>hello world</span></div>’)

* 【插入】

1. 内部尾部添加 $(‘#div’).append($(‘<div></div>’));

$(‘<div></div>’).appendTo(‘#div’);

1. 内部头部添加 $(‘#div’).preend($(‘<div></div>’));

$(‘<div></div>’).preendTo(‘#div’);

1. 元素外部之后添加 $(‘#div’).after($(‘<div></div>’));

$(‘<div></div>’).insetAfter(‘#div’);

1. 元素外部之前添加 $(‘#div’).before($(‘<div></div>’));

$(‘<div></div’).insetBefore(‘#div’);

* 【删除】

1. 删除当前节点 $(‘#div’).remove()
2. 清空节点 $(‘#div’).empty()

* 【替换】

$(‘#div’).replaceWith($(‘<div></div>’)); 替换节点

* 【克隆】

$(‘#div’).clone(true)

* 【注意】

jQuery 中也能的克隆能不把被克隆对象的事件也克隆过来，原生中的克隆不能

* 【包裹】

1. 包裹当前元素 $(‘#div’).wrap(‘<div></div>’)
2. 解除包裹 $(‘#div’).unwrap()

### 3.2.3事件

* 【绑定】大多基本绑定方法都可在手册中查询

1. $(‘#div’).bind(‘click’,function(){}); 通过bind 绑定
2. $(‘#div’).unbind() 解除绑定
3. $(‘#div’).on(‘click’,function(){}); 通过on来绑定

* 【触发】

$(‘button’).trigger(‘click’) 自动触发button按钮的点击功能。

* 【阻止事件默认/冒泡】

return false

* 【坐标】

e.clientX e.clientY / e.pageX e.pageY

两者区别：client只能浏览器上到这个距离，一旦页面有下拉则不会求出隐藏的部分，而page不同，也能求出隐藏部分

* 【注意】事件的加载

事件的加载 一般js的编写都在前面head，或者外文件引入。因此在写js代码的时候都需要加载js改变一下事件的顺序。第一种方法：$(function(){}),第二种方法：$(document).ready(function(){})

### 3.2.4 Ajax

* 【GET】

$.get(‘1.php’,{a:100&b:200},function(data){},’json’) data 服务器返回过来的数据

参数4:json 可不写

* 【POST】

$.post(‘1.php’,{num:10,name:’www’},function(data){}) data服务器返回过来的数据

* 【AJAX】

$.ajax({

url:’1.php’, //url 请求执行的脚本

data:’id=100&name=www’, //data发送的数据

dataType:’json’, //dataType数据的类型

type:’GET’, //type请求的类型 get/post

success:function(){}, //error失败的执行代码

error:function(){}, //error失败的执行代码

timeout:3000, //timeout超过的时间

async:true //async是否异步

});

### 3.2.5效果

* 【快速】

1. 快速显示 $(‘#div’).show() //可加参数
2. 快速隐藏 $(‘#div’).hide() /可加参数

* 【滚动】

1. 滑动显示 $(‘#div’).slideDown(2000); //可加参数
2. 滑动隐藏 $(‘#div’).slideUp(); //可加参数

* 【逐渐】

1. 渐显 $(‘#div’).fadeIn(2000); //可加  
   2. 渐隐 $(‘#div’).fadeOut(2000); //可加

* 【动画】

$(‘#div’).animate(){ }

如果动画指定中需要改变颜时需要在引入一个jquery.color.js 的文件

* 【停止】

$(‘#div’).stop();

* 【延时】

$(‘#div’).delay(1000)

$(‘#div’).hide(2000).delay.fadeIn(2000) //例子：div2秒隐藏后延时1秒后渐显

# Bootstrap

## 4.1简介

Bootstrap是一个HTML、CSS和JS框架。

大多所需的样式框架手册中有能复制的到，因此只总结那些需要注意的点。其他需要的样式可去手册中查找。最主要还是查看手册

## 4.2配置

### 4.2.1 远程访问

1.新 Bootstrap 核心 CSS 文件 <link rel=”stylesheet” href=””>

href="//cdn.bootcss.com/bootstrap/3.3.5/css/bootstrap.min.css"

1. 可选的bootstrap主题文件(一般不用引入)<link rel=”stylesheet” href=””>

href="//cdn.bootcss.com/bootstrap/3.3.5/css/bootstrap-theme.min.css"

1. 引入jQuery。务必在bootstrap之前引入

<script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></script>

1. 引入最新的bootstrap核心JavaScript文件

<script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>

### 4.2.2 本地访问

<link rel="stylesheet" href="/bootstrap.min.css">

<link rel="stylesheet" href="/bootstrap-theme.min.css">

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

一定要在bootstrap.min.js之前引入

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

## 4.3使用

### 4.3.1 样式

* 【代码容器】

.container //居中两边各有间隙

.container-fluid //从左到右

* 【实例】

<div class=’container’ >

<div class=”row”>

//.row表示这一行

</div>

</div>

* 【栅格系统】

1. 将一行分割为12列，可以使用col-md-\*的方式来声明需要占用的宽度
2. 根据屏幕的尺寸分成四种情况col-lg-\* col-md-\* col-sm-\* col-xs-\*
3. 列的偏移:col-md-offset-\*

* 【排版】

1. 标题：.h1~.h6
2. 段落颜色设置分背景色和前景色,背景色类名bg-颜色。前景色text-颜色(颜色有:primary,success,info,warning,danger)
3. 对齐方式 text-left(center,right)
4. 字符串大小写text-lowercase(uppercase、capitalize)
5. 引用:blockqute & footer(blockquote-reverse & footer)
6. 列表:无样式列表(list-unstyled)，行内列表(list-inline)
7. 浮动:清楚浮动(clearfix)
8. 显示和隐藏:show、hide

* 【表单】

1. 校验元素:has-warning、has-error、has-success
2. 元素大小:高度(input-lg、input-sm、input-md)，宽度(col-md-\*)

* 【表单】

1. 斑马文理 .table-striped
2. 边框.table-bordered
3. 悬浮变色.table-hover
4. 请景颜色

* 【按钮】

1. 类名：.btn
2. 样式：.btn-\*(default、primary、success、info、warning、danger)
3. 尺寸：.btn-\*(lg、md、sm、xs)

* 【图片】

1. 响应式：.img-responsive
2. 图片形状：.img-thumbnail

* 【响应式类】

1. visible-\*(lg、md、sm、xs)
2. hidden-\*(lg、md、sm、xs)