**HTML+JS前端大纲**

**整理学习笔记文档**

**整理人：焦康**

**班级：开发二班**

# 第7章 JAVASCRIPT

## 7.1 章节目标

## 7.2 为什么要学习JAVASCRIPT

## 7.3 什么是JAVASCRIPT

7.3.1 JavaScript的组成

ECMAScript(解释器),充当翻译官的角色, js的核心部分.

DOM=Document Object Model([文档对象模型](http://baike.baidu.com/item/%E6%96%87%E6%A1%A3%E5%AF%B9%E8%B1%A1%E6%A8%A1%E5%9E%8B)),赋予了js操作html的能力. BOM (浏览器对象模型).bom赋予了js操作浏览器的能力。

7.3.2 脚本的基本结构(演示)

<script>

Javascript语句

</script>

可以放到html的任何位置. alert('1');调试

7.3.3 在网页中引用JavaScript的方式

1.使用<script>标签

2.外部引用js文件,<script src=’xxx/sss/a.js’ ></script>

3.在html标签里直接使用.

第一种适用于代码量较少,并且页面适用位置不同,每个页面适用js代码均不相同的

情况.

第二种,若干个页面使用js相同,提高代码的复用性,可维护性

第三种,很简单的实现页面效果,比如一个按钮的弹出效果,或一个验证的事件绑定

## 7.4 JAVASCRIPT核心语法

Js是弱类型语言,基于面向对象的,没有明确的数据类型.不需要指定数据类型

7.4.1 变量的声明和赋值

// int var a = 12; a = 23;

// string var str; str = ‘abcd’;

// Boolean var falg; falg = true;

a = ‘abc’;

var 合法的变量名:

var a,b,c = 10;

特别注意:区分大小写,大小写不同的变量标识不同的变量.

Java也是如此

7.4.2 数据类型

Ja数据类型 ,常用的:

1. undefined(未定义类型)
2. number 数值类型
3. string 字符串类型
4. Boolean
5. Object 引用数据类型,比较对象,一些函数

7.4.3 运算符

7.4.4 逻辑控制语句

Switch 表达式 string 数值

Switch(表达式)

{

case 值1:

break;

case 值2

default:

}

循环结构

Java和js一样

while(条件){

Js语法

//一定要设置跳出条件,跳出本次循环continue,跳出整个循环break

}

for(num=1;条件;num++){

js语法

}

do(条件){

js语法

} while(表达式);

7.4.5 数组

Java:string[] a={“”,”,”};

String[] b = new String[](10);

B[0] = 10;b[1] =true;

Js: str = new Array(10);

Str[0] =10;

Str[1] = “zhangsan”;

Str[2] = true;

另一种:

Str1 = new Array(‘a’,12,true);

Str1[0]//”a”;

另一种:

Str2 = [‘a’,12,true]

Js和java里获取数组长度的属性一样,都用length

///注意: strlength = new Array(100); //定长

Strlength = newArray(); //根据需求不指定长度,那就是任意长

声明一个数组的时候,可以不指定长度,并且在以后改变数组长度的时候只需要给数组就行(也可以改变数组的属性)

存取数组元素: 下标0----length-1(长度)

Strs变量名 strs[下标]

Js里面没有trim方法.jquery里有trim方法

分离字符串:

Java:String str1 = <tel:123456:utl:234>;

Js: var str1 = “<tel:123456:utl:234>”; 实现方式:split(“标识”);返回一个数组

Js合并字符串 join(str); 合并之后使用逗号(,)隔开的

字符串数组的排序:

Sort();只适用于字符数组,不适用数值数组

例子:var arr1 = [“z”,”d”,”y”,”A”];

arr1.sort();

Docunment.write(arr1)

反向排序:reverse();

字符串数组的排序:number排序

Var num1 =[1.2.5.9.0];

数组的添加:

1. push(); 将一个或者多个元素添加带数组的结尾,返回数组的长度.
2. unshift();将一个或者多个元素添加到数组的开始,返回数组的长度,其他元素自动后移
3. splice();将一个或者多元素添加到数组的任意位置,添加成功之后返回空; 插入位置的元素后移

例子arr1.push(1,2,3,4);

Arr1.unshift(1,2,3); 如果长度超出了,数组长度自动扩展

Splice(index,howmany,param…)

Index:指定位置的下标

Howmany:删除多少元素

Param:插入的数组元素

数组的删除:

1. pop();将最后一个元素删除,返回该元素
2. shift();将第一个元素删除,返回该元素
3. splice();两个参数可以实现,要删除的下标,和该下标后几个元素

数组的合并和截取

Slice(参数1,参数2) 数组截取方法,参数1:从什么位置开始.参数2:结束的位置,可选,如果不指定结束位置,就从开始位置一直到最后

Concat() 合并数组,用于连接两个或者多个数组,但是并不改变原来的数组,只是返回连接数据的副本

Arr0=[0];

Arr1=[1,2,4];

Arr2=[3,6,7];

Narr = arr0.concat(arr1,arr2)

7.4.6 JavaScript语法规则(记住)

Var用来声明,尽量不省略

JS区分大小写,大写不能替换小写

比如:document Document,,for while永远都是小写

Math大写 new Date();内置对象大写 有些方法是区别大小写的 getElementById();

保留字:

保留字包括:javascript语言的组成部分,以及一些特殊的内置对象window

空格,

7.4.7 注释

/\*\*/ // <!-- -->

Js标记中出现<!-- -->,是为了更好的兼容浏览器

7.4.8 常用的输入、输出

输入:类似java中的scanner类的交互输入,prompt

输出 java syso js:alert(“提示信息”)

Prompt();例子 prompt(‘提示’,”请输入你喜欢的颜色”)

7.4.9 程序调试

## 7.5 JAVASCRIPT常用语法—函数

特点:

1. Js中完成特定功能并随时随地运行的语句
2. 2.不仅可以在一个或者多个html页面多次调用,而且可以在不同网站中应用,提高代码复用性
3. 划分成独立的部分,每一个部分编写一个函数.从而使得各部分相对独立.功能明确任务单一,代码易懂,易维护,易重用

fuction getName(x1,x2){

Js代码

return 返回类型

}

1. 事件驱动和函数绑定.
2. 可传值性,把值作为参数传递,或者返回特定结果(return)

7.5.1 常用系统函数

parseInt; parseFloat; isNaN();(判断是否是 非数字)

function parseInt(str){

var res = xxxx;

return res;

}

Function test(a,a){

A,bb

//事件绑定,或者效果展示

}

例子:

Var I = “123”;

Var a = ParseInt(i);

Var f = “12.12”;

Var fu = parseFloat(f);

IsNaN检测参数是否为数字.

IsNaN(“123”);flase

IsNaN(“12.6”);flase

IsNaN(“12.5s”);true

IsNaN(“45.8”);false

不是数字返回true,是数字返回flase

7.5.2 自定义函数

Js里的函数的语法格式:

Function 函数名 (参数1,参数2) {

// Js语句

}

参数1,和参数2就是传入的变量或者值

{}函数的开始和结束.

return用来规定,从函数中返回的值

如何调用自定义函数:

1. 直接调用
2. Html标签调用

总结:如果要执行一个自定义的函数,必须要先调用这个函数,调用这个函数时,不许指定函数名及其后面的参数.函数的调用一般和时间结合使用

格式: 时间名 = “函数名()”

7.5.3 上机练习3

7.5.4 变量的作用域

Js里全局变量和java中的成员变量(属性)类似,全局变量是在函数之外的脚本中声明的变量,作用范围是在该变量定义之后所有的语句.包含函数以及之后的script标记中的代码.

Js中的局部变量,是在函数内部声明的变量,作用域只在函数内部变量定义之后的语句

## 7.6 WINDOW对象

Window对象是浏览器中最顶层的对象.window对象也成浏览器对象,打开浏览器的时候,通常会生成一个window对象

Alert();

## 

7.6.1 常用的属性

有关客户访问过的url的信息

当前url的信息

7.6.2 常用的方法

Window提供的方法:

Alert(); prompt();

Confirm();简单交互 ,弹出框

Close();

Open();

SetTimeout();定时方法

SetIntervar();定时方法,按照周期,执行函数

例子: window.alert();

格式window.方法名();

1. confirm(“提示,是否删除对象”);弹出对话框,参数一个,返回true,flase

例子:var flag = window.confirm(“是否删除这条数据”);

Alert(flag);

If(flag)

1. open();在页面上弹出一个新的浏览器对象,也就是浏览器窗口,

例子: window.open(url,名称,特征);

Window.open([www.baidu.com](http://www.baidu.com),”弹出框”,”height=400”);

浏览器里有哪些我们可以设置的属性:

Height,width 高度和宽度 width =400;(像素)]

Toolbar 浏览器工具栏 yes/no

Location 是否显示地址栏 yes/no

Status 状态栏

Resizable 窗口是否可调节yes/no

Titlebar 是否显示标题栏 yes/no

把创建新的窗口写到一个函数中,然后在body里定义按钮激发

例子 window.open([www.baidu.com”,””,”toolbar=yes,location](http://www.baidu.com) = yes,status = yes,resizable = no”);

3.close

7.6.3 常用的事件

onloan 一个页面或一个图片加载完成

onclick()事件 鼠标单机某个对象时候调用的事件句柄

onmouseover 鼠标移动到某元素之上

onkeydown 键盘按下

onchanger 域内容改变的时候

1.<body onload=”js语句”></body>加载完之后,菜调用js语句

2.<script>window.onload = js语句><script>

(如果实现body加载完再调用js,可以把js语句放置到body体后面)

7.6.4 上机练习4

## 7.7 DATE对象

## Date对象也是内置对象, 如何创建一个date对象,需要new对象名().方式创一个实例,并且对象提供了一套操作时分秒的方法,或者有一系列的属性.

Math String Array

具体的方法:

getDate 从date里返回一个月的某一天

getDay 从一周中返回某一天

例子: var today = new Date();

today.getHouse(); today.getMinutes(); today.getSeconds(); //获取时分秒

如何计算当天是几号,周几,几时几分几秒

## 7.8 定时函数

7.8.1 常用的定时函数

setTime

## 7.9 课后总结

## 7.10 课后作业

# 第8章 正则表达式及表单验证

## 8.1 章节目标

## 8.2 表单验证的意义

## 

## 8.3 常用的表单验证方法

8.3.1 String对象方法

*var* str1 = "str1234589564";

*var* s=str1.indexOf("r");

*var* last = str1.lastIndexOf("5")

*var* c = str1.charAt(6)

*var* ss = str1.substring(3,6)

var array1=”a,c,b,d”

array1.

alert(s+"<br>"+last+" "+c+" "+ss)

8.3.2 数字验证方法

IsNaN() 如果是数字返回flase

练习:用户名\_\_\_\_\_\_(字母数字下划线)

密码\_\_\_\_\_(6-12字符组成)

性别\_\_\_\_\_(非空)

出生日期\_\_\_\_(年月日)

电子邮件\_\_\_\_(不能为空)

8.3.3 正则表达式

|  |  |  |
| --- | --- | --- |
| **符号** | **等价类** | **说明** |
| **.** | **[^\n\r]** | **匹配除换行以外的任意字符** |
| **\d** | **[0-9]** | **匹配数字字符** |
| **\D** | **[^0-9]** | **非数字字符** |
| **\s** | **[\t\n\xOB\f\r]** | **匹配任意空白符** |
| **\S** | **[^\t\n\xOB\f\r]** | **非空白自负** |
| **\w** | **[a-z A-Z\_0-9]** | **匹配字母或数字下划线** |
| **\W** | **[^a-z A-Z\_0-9]** | **非单词字符** |
| **\b** |  | **匹配单词的开始或者结束** |
| **^** |  | **匹配字符串的开始** |
| **$** |  | **匹配字符串的结束** |
| **\*** |  | **重复零次或更多次** |
| **+** |  | **重复一次或更多次** |
| **?** |  | **重复零次或一次** |
| **{n}** |  | **重复n次** |
| **{n,}** |  | **重复n次或更多次** |
| **{n,m}** |  | **重复n到m次** |

| 标识:左右两边表达式之间”或”的关系,匹配左边或者右边

() 表示: 匹配时括号中的内容可以被整体对待

转义符:\

有这样一个字符串” 135\*\*\*123 ”如果想表示 “\\*”

一个字符串”\user\sys\”; 表示”\\”;

正则表达式用法:

Var 字符串 = “ststem.out.prinpin ”; //是否包含两个点

Var 规则 = “”;

Var flag = 规则.test(字符串); flag就是一个结果,是真还是假

例子:

规则如何去写,首先想到的就是一个正则表达式对象RegExp;

Var = 规则 = new RegExp(“\\d+”); //通常我们不去这样写,用他的简写方式

Var 规则名 = /\d+/; 和上面是等价的

简写格式:

Var reg = /规则/;

Reg.test(要匹配饿的字符串);

If((Reg.test(要匹配的字符串)==true);

Alert(“真”)

)else{

Alert(“假”)

}

## 8.4 课后作业

8.4.1 网易邮箱注册页面及表单验证

# 第9章 DOM操作

## 9.1 章节目标

## 9.2 什么是DOM

## Dom:文档对象模型.

Google经典搜索算法: pagerank

## 9.3 获取元素的方法

9.3.1 getElementByID

根据id获取

9.3.2 getElementsByClassName

根据name属性获取元素

<input type = “text” value=”默认值” id=”” name=”服务器获取值的标识”/>

9.3.3 getElementsByTagName

根据标签获取 getElementsByTagName(“ul”)

9.3.4 getElementsByName

根据样式名字获取

## 9.4 获取和设置属性的方法

9.4.1 getAttribute方法

此方法只有一个参数,就是要查找的属性的名字.

只能在元素节点上调用此方法:element.getAttribute(“属性名字”);

9.4.2 setAttribute方法 两个参数(属性名字,值)

同上.

## 9.5 几个常用DOM属性

9.5.1 childNodes属性

childNodes可以获取元素的子元素 ,返回一个对象数组

9.5.2 nodeType属性

属性值 3#text 文本节点

属性值 2 属性节点

属性1 元素节点

9.5.3 nodeValue属性

文本节点的值

元素.nodeValue=” ”;

9.5.4 firstChild、lastChild属性

元素.firstChild 返回一个元素;

Body元素数组,调用firstchild ,返回的就是<h1>元素.

调用lastchild,返回的就是<a>标签

## 9.6 动态创建标记

9.6.1 传统方法

Document.write(olddom+“new node”) //整棵树,包括渲染的样式;

方便快捷的的把字符插入到文档中.

违背了单一职责原则.违背了行为和表现集成一体.

最好的方式就是把结构样式和行为分开.最最好就是用单独的js文件去控制页面行为

9.6.2 DOM方法

Dom动态添加方法,没有改变原来的文档树的物理内容

9.6.3 在已有元素前插入一个新元素

createElement(“新元素名”) ;

创建一个段落,并在段落里添加一些信息.

Var elemrent = Document.createlement(“p”);

新创建的元素不会自动的添加到dom树文档里.此新节点没有父节点nodeParent属性:

只存在于js语句中

1. createTextnode();创建新闻本节点,返回一个对象节点的引用指针
2. appendChild 把新节点插入到某文档节点,称为此文档的子节点

<body>

Body.appendChild(new Child);

通常结合createElement和CreateTextNode使用

9.6.4 在已有元素后插入一个新元素（了解）

9.6.5 删除节点

RemoveChild(node);

9.6.6 替换节点

9.6.7 innerText和innerHTML

## 9.7 课后作业

9.7.1 全选/全不选

9.7.2 添加新员工

9.7.3 省市区三级联动

# 第10章 JQUERY简介

## 10.1 章节目标

## 10.2 认识JQUERY

10.2.1 从JavaScript到jQuery

10.2.2 jQuery的优势

## 10.3 简单的JQUERY使用

10.3.1 jQuery环境配置

10.3.2 简单的jQuery代码

## 10.4 JQUERY对象和DOM对象

10.4.1 DOM 对象

10.4.2 jQuery 对象

10.4.3 jOurery 对象和DOM 对象的相互转换

## 10.5 本章作业

# 第11章 JQUERY选择器

## 11.1 章节目标

## 11.2 基本选择器

## 11.3 层次选择器

11.3.1 层次选择器

11.3.2 jQuery元素选择方法

## 11.4 过滤选择器

## 11.5 表单选择器

11.6 小结

# 第12章 DOM操作

## 12.1 章节目标

## 12.2 查找节点

12.2.1 查找元素节点

12.2.2 查找节点属性

## 12.3 创建节点

12.3.1 创建元素节点

## 12.4 插入节点

## 12.5 删除节点

## 12.6 复制节点

## 12.7 替换节点

## 12.8 包裹节点

## 12.9 属性操作

12.9.1 获取属性和设置属性

12.9.2 删除属性

## 12.10 样式操作

12.10.1 设置样式和获取样式

12.10.2 追加样式

12.10.3 移除样式

12.10.4 判断是否含有某个样式

## 12.11 设置和获取HTML、文本和值

## 12.12 CSS操作

## 12.13 小结

## 12.14 作业

# 第13章 事件和动画

## 13.1 章节目标

## 13.2 事件

13.2.1 ready事件

13.2.2 事件绑定

13.2.3 合成事件

13.2.4 事件移除

## 13.3 动画

13.3.1 显示和隐藏

13.3.2 淡入淡出

13.3.3 滑动

13.3.4 自定义动画

13.3.5 回调函数

13.3.6 其它动画方法

## 13.4 作业

# 第14章 表单表格操作

## 14.1 章节目标

## 14.2 表单应用

14.2.1 单行文本框

14.2.2 多行文本框

14.2.3 复选框

14.2.4 下拉框

14.2.5 表单验证

## 14.3 表格应用

14.3.1 表格变色

14.3.2 折叠效果

14.3.3 内容筛选

14.3.4 表头排序

# 第15章 JQUERY插件的使用和写法

## 15.1 概述

## 15.2 JQUERY表单验证插件——VALIDATION

15.2.1 Validation 简介

15.2.2 快速入门

15.2.3 增加中文验证的写法

15.2.4 去除class验证方式的写法

## 15.3 JQUERY表单插件-FORM

15.3.1 Form插件简介

15.3.2 快速上手

15.3.3 核心方法一ajaxForm()和ajaxSubmit()

15.3.4 ajaxForm()方法和ajaxSubmit()方法的参数

## 15.4 JQUERY UI插件

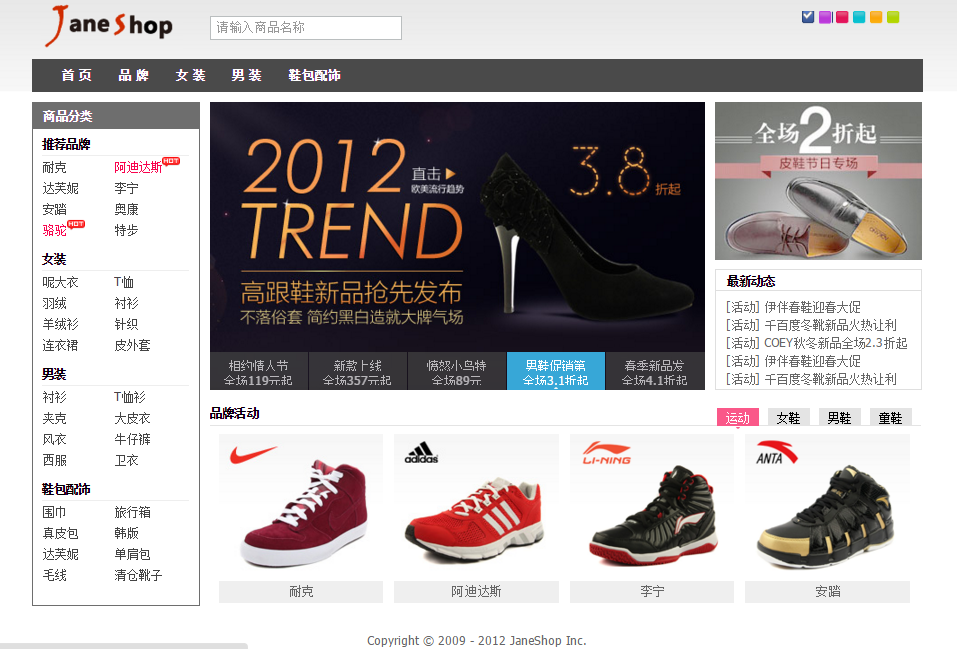
15.4.1 jquery UI 简介

15.4.2 快速上手

15.4.3 与单击事件的冲突

# 第16章 JQUERY打造个性网站

## 16.1 概述



## 16.2 网站结构

16.2.1 文件结构

16.2.2 网页结构

16.2.3 网站样式效果

## 16.3 网站脚本（JQUERY）

16.3.1 准备工作

16.3.2 首页（index.html）上的效果

16.3.3 详细页（detail.html）上的效果

## 16.4 总结