智能社：JavaScript教程-从入门到精通 笔记

# 第一章——基础入门

## 课时1、2

·JavaScript就是修改样式

### 写JS流程：

1.在写JS之前，要保证有一个稳定的布局（HTML+css），这个布局本身不能有任何兼容问题

2.确定要修改哪些属性

3.事件：确定用户要做哪些操作（产品设计）

4.编写JS：在事件中，用JS来修改页面元素的样式

### 分析效果实现原理：

1.样式：Div的display

2.事件：onmouseover、onmouseout

3.动手编写效果（例1\_1）

### 零散知识点：

·【document.getElementById()】通过id获取元素，在任何浏览器下均可使用（解决浏览器兼容性问题，如火狐）;

·网站的“开灯”“关灯”效果（网页换肤）

·任何标签都可以加ID，包括link

·任何标签的任何属性，也都可以修改

·HTML里面怎么写，js里就怎么写，仅className除外；（例1\_6；给div1运用class box时，在js里面不能写成【oDiv.class='box';】而要写成【oDiv.className='box';】）

#### 注意：

·可用【var xxx=xxxxxxx】去定义一串东西，就是给那串东西加个简短一点的名字;

·<a>的href里面一般不放【#】，因为一点，会弹到页面最上，而放【javascript:;】不会，可以在a里放js代码，也能执行，但是一般不会这么做，可以把那空着，写【javascript:;】（如<a href="javascirpt:;">123</a>）;

·【document.getElementById('div1').style.height='200px';】中的“.”，可以理解为“的”;

#### 扩展：

·为a链接添加JS（【<a href="javascript:;"> </a>】）

·className的使用

## 课时3、4、5

·函数传参：参数就是占位符

·改变div的任意样式

——操作属性的第二种方式

·什么时候用：要修改的属性不固定

·字符串123【'123'】和变量

——将属性名作为参数传递（例1\_11）

### 两种操作属性的方法：

1.【oTxt.value='123456789'】就是用点，点后加属性名字

2.【oTxt[name]='123456798'】用[]，此处的name是函数的一个参数，在调用函数的时候，给它一个参数，'value'、'title' etc...（例1\_11）

### style与className

——元素.style.属性=xxx是修改行间样式，style只能操作行间样式

——之后再修改className不会有效果的

原因：样式优先级\*<标签<class<ID<行间（例1\_12）

【建议：对于同一个元素，如同一个div，要么从头到尾操作它的class，要么就操作它的style，别混着来】

### 提取事件

——为元素添加事件

·事件和其他属性一样，可以用Js添加（例1\_13，但js一般不放在body，见例1\_14）

·window.onload的意义：加载完整个页面时再执行其中代码，防止出错（例1\_14）

·行为（Js）、样式（CSS）、结构（HTML）三者分离【即叠加行间样式，叠加行间Js事件】

### 获取一组元素

——getElementsByTagName：直接获取一组元素，如所有的div，像数组（例1\_15）

### 循环

循环的构成：初始化、条件、语句、自增

·while循环

·for循环（例1\_16、1\_17）

### 选项卡（例1\_18）

·按钮的实现

——添加事件

·this的使用

——先清空所有，再选择当前

·内容的实现（div 的显示）

——先隐藏所有Div，再显示当前Div

·索引值的使用

——什么时候用索引值：需要知道“第几个”的时候

·html中添加index会被FF过滤

·要用JS添加index

### JS简易日历（例1\_20）

·程序的实现

——类似于选项卡，只是下面只有一个div

——innerHTML的使用（例1\_19）

·数组的使用

·字符串连接

#### 注意：

·变量和参数就不用带【' '】单引号，即，你不带引号，系统都会把它识别为变量，字符串就要带单引号

·例1\_14中，JS都是一行一行读取的，如果不用window.onload，当JS读取到btn1处时，按钮都还没加载到页面中，因此就算点了按钮也没有反应，但是用了window.onload之后，当页面加载完了再执行里面的东西，就没有问题了

·index不是标准的HTML里具备的属性，如果直接加载HTML里，在标准的浏览器中会被过滤掉而出错，但是可以用js给HTML里的元素加index

# 第二章——基础应用

## 课时6

### JavaScript组成：

·ECMAScript：解释器、翻译（几乎没有兼容性问题）【把人可读的代码变成0和1】

·DOM：Document Object Model（有些操作不兼容）【是Js用来操作HTML的】

·BOM：Browser Object Model（没有兼容问题，因为完全不兼容）【用来操作浏览器的】

### 数据类型

·类型显示：typeof运算符（例2\_1）

常见类型：number、string、boolean（布尔），undefined、object、function

变量的类型取决于你在变量里面放的东西......

·一个变量 应该/最好 只存放一种类型的数据

### 数据类型转换

·例子：计算两个文本框内的数字和（例2\_2）

·显式类型转换（强制类型转换）

·parseInt()、parseFloat()

·NaN（例2\_3）

·隐式类型转换（例2\_4）

·==、===

·减法

·parseInt()转换整数：从字符串里扫描数字，是从左往右，碰到非数字立刻终止【如12px34，转换之后是12，若是abc，转换后得出的是NaN】

·parseFloat()转换小数

·NaN：Not a Number

·NaN不等于NaN、isNaN()的应用（例2\_3）

### 变量作用域和闭包

·局部变量：只能在定义它的函数里面使用

·全局变量：在任何地方都能用，定义在所有函数之外

·什么是闭包？（例2\_5）

·闭包就是能够读取其他函数内部变量的函数。【百度百科】

·子函数可以使用父函数的局部变量

### 命名规范

·命名规范及必要性

·可读性——能看懂

·规范性——符合规则

·匈牙利命名法

·类型前缀：给变量取名字的时候才用，给函数取名字时不需要

·首字母大写（oDivUserLogin）

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **前缀** | **类型** | **实例** |
| 数组 | a | Array | aItems |
| 布尔值 | b | Boolean | bIsComplete |
| 浮点数 | f | Float | fPrice |
| 整数 | i | Integer | iItemCount |
| 函数 | fn | Function | fnHandler |
| 对象 | o | Object | oDiv1 |
| 正则表达式 | re | RegExp | reEmailCheck |
| 字符串 | s | String | sUserName |
| 变体变量 | v | Variant | vAnything |

## 课时7

### 运算符

·算术：+加、-减、\*乘、/除、%取模

·实例：隔行变色（例2\_6）、秒数转时间（例2\_7）

·赋值：=、+=、-=、\*=、/=、%=【i+=1即i=i+1】

·关系：<、>、<=、>=、==、===、!=、!==

·逻辑：&&与、||或、!否

·运算符优先级：括号

注：

**【!=**】在表达式两边的数据类型不一致时,会隐式转换为相同数据类型,然后对值进行比较.

**【!==**】不会进行类型转换,在比较时除了对值进行比较以外,还比较两边的数据类型, 它是恒等运算符===的非形式.

**【!**】让真变假、假变真

### 程序流程控制

·判断：if、switch、?:

·循环：while、for

·跳出：break、continue（例2\_9）

·真：**true**、**非零**数字、**非空**字符串、**非空**对象

·假：**false**、数字**零**、**空**字符串、**空**对象（null）、**undefined**

【?:】三目运算符：（条件?语句1:语句2）条件成立执行语句1，否则执行语句2

·Json

·是什么？【Js中一种用来存数据的方法】

·Json和数组

·Json和for in

## 课时8

### 函数返回值

·什么是函数返回值：函数的执行结果

·可以没有return

·一个函数应该只返回一种类型的值

### 函数传参

·可变参（不定参）：arguments

·参数的个数可变，arguments是一个数组，存的是传给本函数的参数

·例子1：求所有参数的和（例2\_11）

·例子2：CSS函数（例2\_12、例2\_12-2）

判断arguments.length

给参数取名，增加可读性（例2\_12-2）

·取非行间样式（不能用来设置）：

·obj. currentStyle[attr] （attribute属性）可用于获取非行间样式，仅能用于IE

·getComputedStyle(obj,false)[attr] 不能用于IE，适用于360、FF、Chrome（例2\_13）

### 数组基础

·数组的使用

·定义

·var arr=[12,5,8,9];

·var arr=new Array(12,5,8,9);

·没有任何差别，[]的性能略高，因为代码短

·数组的属性

·length

·既可以获取，又可以设置（例2\_14）

·例子：快速清空数组（让length=0）

·数组使用原则：数组中应该只存一种类型的变量

### 数组添加、删除元素

·添加

·push(元素)，从尾部添加

·unshift(元素)，从头部添加

·删除

·pop()，从尾部弹出

·shift()，从头部弹出

·尾部：push——pop

·头部：unshift——shift

·push用的相对较多，其他较少（例2\_15）

### 数组插入、删除元素

·splice（例2\_16）

·删除【splice(起点，长度)】

·插入【splice(起点，0，元素...)】先删除，后插入

·替换（插入的特殊用法）

### 数组排序、转换

·排序

·sort([比较函数])，排序一个数组

·排序一个字符串数组（例2\_17）

·排序一个数字数组（例2\_18）

·【数组1.concat(数组2) 】连接两个数组（例2\_17）

·join(分隔符)

·用分隔符，组合数组中的元素，生成字符串（例2\_17）

·字符串split

#### 扩展：

·jQuery中：

·css(oDiv, 'width') 获取样式

·css(oDiv, 'width', '200px') 设置样式

·getComputedStyle(obj,false)[attr] 中，【false】位置是第二个参数，这个参数是个“垃圾”，往里面放什么都行，将【false】换成【null】、换成【12】，换成【document】，换成【'abcd'】.....都可以

·例子2\_13-2 是2\_13的简化，将解决兼容部分的内容封装成函数，直接调用函数；以后也会遇到很多这个情况，封装成的函数可以弄成外部的js文件，这种封装函数多了之后，大家可以注意总结，最后弄一套自己的库出来，用得很方便

·js里面，代码越短，加载得越快（功能一样的前提下）

·复合样式：background、border

·单一样式/简单样式：width、height、position

·obj. currentStyle[attr] 、getComputedStyle(obj,false)[attr]不能取复合样式，如果想取背景颜色，不能只打【background】，而要打【backgroundColor】

## 课时9、10

### 定时器的使用

·开启定时器

·setInterval 间隔型（每隔xx毫秒就执行一次）

·setTimeout 延时型（间隔xx毫秒只执行一次）

·停止定时器

·clearInterval() 关闭定时器的时候，要指明是关闭哪个定时器

·clearTimeout()

（例2\_19）

### 数码时钟

·获取系统时间（获取出来的数据类型都是数字，不是字符串）

·Date对象

·getHours、getMinutes、getSeconds

·获取年：getFullYear()

·获取月：getMonth()+1，获取到的数字会比实际月份少1，所以要加1

·获取日：getDate()

·获取星期：getDay()，会获取到0-6的数字，0代表周日，类推

·显示系统时间

·字符串连接

·空位补零

·设置图片路径

·charAt() 替代[] 解决兼容性问题

（例2\_20）

### 延时提示框

·像QQ，指着头像时显示QQ名片，鼠标移走后过一会再小时

·移入显示，移出延时隐藏（例2\_21）

·简化代码

·合并两个相同的mouseover和mouseout

·js中可以【a=b=c=1】，所以俩事件也可以连等一个函数（例2\_22）

### 无缝滚动

·物体运动基础

·让Div里的ul移动

·offsetLeft作用（可以综合考虑所有影响物体位置的因素，最后得出物体左边距）

·用定时器让物体连续移动

·复制li

·innerHTML和+=

·修改ul的width

·滚动过界后，重设位置

·判断过界

（例2\_23）

### 无缝滚动——扩展

·改变滚动方向

·修改speed

·修改判断条件

·鼠标移入暂停

·移入关闭定时器

·移出重新开启定时器

（例2\_24）

#### 扩展：

·offsetLeft/offsetTop

·offsetWidth / offsetHeight

·css中的\*是通配符，作用于所有元素。【\*{margin:0px;padding:0px}】就是整个页面里的所有元素，全部margin和padding都设为0，这是css reset的一种写法。不推荐。过于鲁莽，效率也不高。【https://segmentfault.com/q/1010000005362370】

# 第三章——DOM应用

## 课时11

### DOM基础 - 1

·什么是DOM——即document

·浏览器支持情况：IE（10%）、Chrome（60%）、FF（99%）

·DOM节点

·childNodes nodeType节点类型 （例3\_01）

·获取子节点

·children（例3\_02）//这个可以兼容，可以代替childNodes

·parentNode父节点（例3\_03）

·例子：点击链接，隐藏整个li（例3\_04）

·offsetParent

·作用：可以获取一个元素 它用来定位的父级（例3\_05）

#### 注意：

·childNodes是个数组，存的是某元素下的所有子节点，有length属性

·**子节点只算第一层的**，不会往里算（例3\_02中<li></li>中的<span></span>不会算进去）

·**绝对定位的元素，是根据有定位的父级定位的**，如果它直接的父级没定位，它就一直一级一级往上找有定位的，如果都没有，就一直找到body（例3\_05，如果DIV1的position删掉，div2就是根据body定位的）

### DOM基础 - 2

·DOM节点(2)

·首尾子节点

·有兼容性问题

·firstChild、firstElementChild（获取某元素下第一个子节点）

·lastChild、lastElementChild

·兄弟节点

·有兼容性问题

·nextSibling、nextElementSibling

·previousSibling、previousElementSibling

（例3\_06是firstChild、firstElementChild的对比，其他三个类推）

### 操纵元素属性

·元素属性操作方法

·第一种：oDiv.style.display="block";

·第二种：oDiv.style["display"]="block";

·第三种：Dom方式

·DOM方式操作元素属性

·获取：getAttribute(属性名称)

·设置：setAttribute(名称,值)

·删除：removeAttribute(名称)

（例3\_07）

#### 注意：

·能用getAttribute()和setAttribute()的地方都能用[]的方法

·但是平常一般不用getAttribute()

### DOM元素灵活查找

·用className选择元素

·选出所有元素

·通过className条件筛选

·封装成函数

## 课时12

### 创建、插入和删除元素

·创建DOM元素

·createElement(标签名) 创建一个节点（先创建）

·appendChild(节点) **追加**一个节点（再把创建好的在后面放进去）

·例子：为ul插入li（例3\_10）

·插入元素

·insertBefore(节点,原有节点) 在已有元素**前**插入

·例子：倒序插入li（例3\_11）

·删除DOM元素

·removeChild(节点) 删除一个节点

·例子：删除li（例3\_12）

### 文档碎片（基本不怎么用了，没什么太大用处，知道就行）

·文档碎片可以提高DOM操作性能（理论上）

·document.createDocumentFragment()

·例3\_13是插入一次渲染刷新一次, 3\_14则是插入完了再一次性添加，只用刷新一次

·在高级浏览器中用3\_13还是3\_14的方法都区别不大，甚至后者可能降低性能，不过在IE低版本浏览器中用后者怎能稍微提高性能

## 课时13

### 表格应用 - 1

·获取（例3\_15）

·tBodies、tHead、tFoot、rows、cells

·隔行变色

·

注意：

·<tbody></tbody>按道理可以不写，但是不写的话浏览器也会自动给你补上，写上的话程序不容易乱

·一个表格可以有很多个tbody，但只能有一个thead和一个tfoot，所以tBodies是一个数组，tHead和tFoot是一个元素

### 表单应用 - 1