Html 处理页面的结构

CSS 处理页面的样式

JavaScript 实现与用户的交互

# 一 JavaScript简介

在1995 年 Netscape 一位名为 Brendan Eich 的工程师创造了 JavaScript，随后在 1996 年初，JavaScript 首先被应用于 Netscape 2 浏览器上。最初的 JavaScript 名为 LiveScript，后来因为 Sun 公司 的 Java 语言的兴起和广泛使用，Netscape 出于宣传和推广的考虑，将它的名字从最初的 LiveScript 更改为 JavaScript——尽管两者之间并没有什么共同点。

几个月后，Microsoft 随着 IE 3 推出了一个与之基本兼容的语言 JScript。又几个月后，Netscape 将 JavaScript 提交至 Ecma International（一个欧洲标准化组织）， ECMAScript 标准第一版便在 1997 年诞生了，随后在 1999 年以 ECMAScript 第三版的形式进行了更新，从那之后这个标准没有发生过大的改动。由于委员会在语言特性的讨论上发生分歧，ECMAScript 第四版尚未推出便被废除，但随后于 2009 年 12 月发布的 ECMAScript 第五版引入了第四版草案加入的许多特性。第六版标准已经于2015年六月发布。

与大多数编程语言不同，JavaScript是一个在宿主环境（host environment）下运行的脚本语言，任何与外界沟通的机制都是由宿主环境提供的。浏览器是最常见的宿主环境

## 1 JavaScript的特点

JavaScript主要用来向html页面中添加交互行为

JavaScript是一种脚本语言，语法和c语言系列语言的语法类似，属弱语言类型。

JavaScript一般用来编写客户端脚本。

JavaScript是一种解释型语言，边执行边解释无需另外编译。

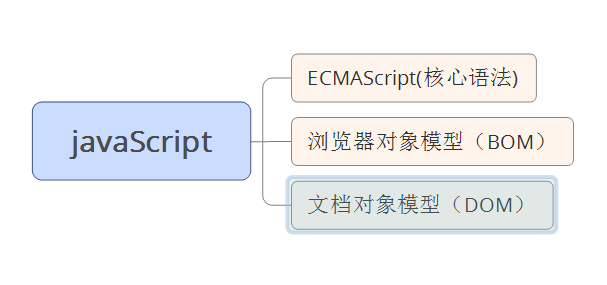
## 2 JavaScript的用途

JavaScript解决页面交互和数据交互，最终目的是丰富客户端效果以及数据的有效传递。

实现页面交互，提升用户体验实现页面特效。即通过js进行html的dom操作或样式操作。

在客户端实现表单验证。即在数据送达服务端之前进行用户提交信息即时有效地验证，减轻服务器压力。

## 3 主要组成



## 4 JavaScript引入方式

1)在html标签中引入（主要用于事件）

<input type="button" name="" onclick="alert('ok！');">

2)通过<script>标签在页面嵌入

<script type="text/javascript">

var a = 'hello';

alert(a);

</script>

3)引入外部js文件

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

# 二 JavaScript语法基础

## 1 类型

JavaScript 是一种弱类型语言，javascript的变量类型由它的值来决定。 定义变量需要用关键字 var

变量命名规范

1）区分大小写，由字母、数字、下划线、美元符号组成

2）第一个字符必须是字母、下划线（\_）或者美元符号（$）

1)基本类型 string、number、boolean、null 、undifiened

<script type="text/javascript">

//定义变量，并赋值

var num = 1;

//弹出消息提示框

//alert(num)//1

//获取变量的类型

//alert(typeof num);//number

var f = 12.3;

//alert(typeof f);//number

f = "javascript";

//alert(typeof f);//string

//定义变量时，var可以不写

age = 10;

//alert(age);

// 字符串内容使用 "" 或 '' 括起来

var str = "hello";

//alert(typeof str);//string

var str1 = 'world';

//alert(str1);

var flag = true;

//alert(typeof flag);//boolean

</script>

undefined 未定义的，表示变量不含有值。

可以通过将变量的值设置为 null 来清空变量

<script type="text/javascript">

var bike;

//alert(bike);//undefined

//alert(typeof bike);//undefined

var car = null;

//alert(car);//null

//alert(typeof car);//object

</script>

3)引用类型（复合类型） 数组、对象

本质上都是object类型

<script type="text/javascript">

//[] 括起来的内容 表示数组

var arr = [1, 2, 3, 4, 5];

//alert(arr);

//alert(typeof arr);//object

//alert(arr[0]);

// {}括起来的数据表示对象， 对象中数据为键值对

var obj = {name:"zhangsan", age:12};

//alert(obj)//[object Object]

//alert(typeof obj);//object

//根据key值获取value

//alert(obj.name);

//alert(obj["name"]);

//alert(obj[name]);//undifined

// key值是不是使用""括起来都有效

var obj1 = {"uname":"lisi"};

//alert(obj1.uname);

//alert(obj1["uname"]);

</script>

## 2 运算符

算术运算符： +(加)、 -(减)、 \*(乘)、 /(除)、 %(求余)

自增自减运算符：++ 、--

赋值运算符：=、 +=、 -=、 \*=、 /=、 %=

条件运算符：==、===、>、>=、<、<=、!=

逻辑运算符：&&、||、!

三目元算符：?:

## 3 分支与循环结构

1）分支结构

if…else if…else...

switch…case

<script type="text/javascript">

var num = 10;

var str = "10";

var str1 = "10";

// == 判断内容是否相同,不考虑类型

if(num == str){

//alert("y");

}else{

//alert("n");

}

if(str == str1){

//alert("y");

}else{

//alert("n");

}

// === 既判断内容，也判断类型是否相同

if(num === str){

//alert("y");

}else{

//alert("n");

}

// 支持数字、字符串类型

switch(num){

case 10:

//alert(10)

break;

case 11:

alert(11)

break;

default:

break;

}

2）循环结构

while

do...while

for循环

for...in...

var arr = [2, 4, 1, 6];

//length 获取数组的长度

for(var i = 0; i < arr.length; i++){

//alert(arr[i]);

}

//for-in,针对数组，i表示数组的下标

for(var i in arr){

//alert(arr[i]);

}

var obj = {name:"zhangsan", age:12, weight:100};

// 针对对象，k表示对象中的key值

for(var k in obj){

//alert(k);

//alert(obj.k);//undifined

alert(obj[k]);//获取key对应的value值

}

//while

//do-while

</script>

## 4 javascript对象

<script type="text/javascript">

//包装类型

var str = new String("hello");

//alert(str);

var num = new Number(100);

//alert(num);

//alert(typeof num);//object

//下标不连续、元素类型不一样，语法上都没问题

var arr = new Array();

arr[0] = "zhagnsan";

arr[1] = "lisi";

arr[3] = "wangwu";

arr[4] = 123;

//alert(arr);

//动态添加键值对

var obj = new Object();

obj.name = "haha";

obj.age = 12;

//alert(obj);

//alert(obj.name);

//alert(obj["name"]);

//不推荐如下的用法

arr.num = 100;

alert(arr.num);

str.nn = "heihei";

alert(str.nn);

num.tt = 34;

alert(num.tt);

</script>

（1）数值类型操作

toFixed()：把数字转换为字符串，可以指定小数的位数

toString()：数字转为字符串

Number.MAX\_VALUE 获取js中可以表示的最大数

Number.MIN\_VALUE 获取可以表示的最小数

NaN：表示非数字，isNaN()函数用来判断是否不是数字，如果参数可以转为数字，返回false

（2） 字符串操作

length 获取字符串的长度

concat()字符串合并,也可以通过“ + ”  
parseInt() 将数字字符串转化为整数  
parseFloat() 将数字字符串转化为小数

toString() 数字转为字符串  
split() 把一个字符串分隔成字符串组成的数组  
charAt() 获取字符串中的某一个字符，也可以通过下标方式，如str[0]

charCodeAt() 获取指定索引处字符的编码  
indexOf() 查找字符串是否含有指定内容  
substring() 截取字符串 用法： substring(start,end)（不包括end）

substr() 截取字符串，第二个参数表示截取的长度

replace() 字符串内容的替换  
toUpperCase() 字符串转大写  
toLowerCase() 字符串转小写

<script type="text/javascript">

//可以访问其他script中的变量

//alert(flag);

var str = "helloworld";

//alert(str);

//获取指定索引的字符

//alert(str.charAt(0));

//alert(str[2]);//通过小标获取字符

//获取指定索引处字符的编码

//alert(str.charCodeAt(0));

//获取字符串的长度

//alert(str.length);

//字符串截取，两个参数都表示索引，截取的内容不包括第二个索引处的字符

//alert(str.substring(2, 5));

//alert(str);//不会改变原来得字符串内容

//截取指定索引后面所有的字符

//alert(str.substring(2));

//第二个参数代表截取的字符串的长度

//alert(str.substr(2, 5));

//alert(str.substr(2));

//字符串内容替换，支持正则表达式的写法，只会替换第一个匹配的字符串

//alert(str.replace("l", "L"));

var ss = "Hello";

//比较字符串大小，相同返回0，前面比后面大返回1，比后面小返回-1

//alert(str.localeCompare(ss));

//字符串拼接

//alert(str.concat("hahaha"));

//通过+实现字符串拼接

//alert("hello" + " world");

//判断是否包含字符串,返回-1,说明不包含

var ret = str.indexOf("Hello");

//alert(ret);

var n = "123";

//整数字符串转换为数字

var n1 = parseInt(n);//parseFloat()

//alert(typeof n1);//number

//数字转换为字符串

var n2 = n1.toString();

//alert(typeof n2);

</script>

（3） 数组操作

length获取数组的长度

var aList = [1,2,3,4];

alert(aList.length); // 弹出4

join() 将数组成员通过一个分隔符合并成字符串

var aList = [1,2,3,4];

alert(aList.join('-')); // 弹出 1-2-3-4

push() 和 pop() 从数组最后增加成员或删除成员

var aList = [1,2,3,4];

aList.push(5);

alert(aList); //弹出1,2,3,4,5

aList.pop();

alert(aList); // 弹出1,2,3,4

unshift()和 shift() 从数组前面增加成员或删除成员

var aList = [1,2,3,4];

aList.unshift(5);

alert(aList); //弹出5,1,2,3,4

aList.shift();

alert(aList); // 弹出1,2,3,4

reverse() 将数组反转

var aList = [1,2,3,4];

aList.reverse();

alert(aList); // 弹出4,3,2,1

indexOf() 返回数组中元素第一次出现的索引值

var aList = [1,2,3,4,1,3,4];

alert(aList.indexOf(1));

splice() 在数组中指定索引处增加或删除成员

var aList = [1,2,3,4];

aList.splice(2,1,7,8,9); //从第2个元素开始，删除1个元素，然后在此位置增加'7,8,9'三个元素

alert(aList); //弹出 1,2,7,8,9,4

concat() 数组合并

练习：如何实现字符串反转？

var str = 'asdfj12jlsdkf098';

var str2 = str.split('').reverse().join('');

## 5 函数

函数就是重复执行的代码块

(1) 函数的定义

函数的基本语法：

function 函数名(参数列表){}

无参无返回值的函数：

function printInfo(){

alert(123);

}

//函数调用

//printInfo();

带参待返回值的函数：

//参数不用使用var，如果有返回值，直接return

function add(a, b){

return a + b;

}

var v1 = add(10, 20);

//alert(v1);

//alert(add(12.3, 34));

//alert(add("10", "40"));//字符串的拼接

//alert(add(10, "40"));//字符串的拼接

//调用js文件中的函数

//alert(sub(20, 12));

注意：js的函数，语法上，不用任何情况下都有返回值

function add1(a, b){

if(a > 10){

return a + b;

}

}

//alert(add1(2, 30));//undefined

(2) 匿名函数

没有名称的函数叫做匿名函数，可以将匿名函数直接赋值给元素绑定的事件来完成匿名函数的调用。

//匿名函数，没有函数名，不能直接调用

//所以使用变量来获取函数，变量相当于函数名

var func = function(a, b){

return a - b;

}

var ret = func(12, 3);

//alert(ret);

var arr = [56, 12, 1, 34, 3];

//排序，默认升序

//arr.sort();

//alert(arr);//1,12,3,34,56 将数字作为字符串处理，进行排序

//sort的参数，为一个提供比较方法的函数

// arr.sort(function(a, b){

// return a - b;

// });

// arr.sort(func);

function comp(a, b){

return a - b;

}

arr.sort(comp);

alert(arr);//1,3,12,34,56

（3）变量的作用域

变量作用域指的是变量的作用范围，

全局变量：在函数之外定义的变量，为整个页面公用，函数内部外部都可以访问。

局部变量：在函数内部定义的变量，只能在定义该变量的函数内部访问，外部无法访问

## 6 创建自定义对象 （理解）

(1)对象初始化器方式

var user = {

name : 'Tom',

age : 18,

showName : function(){

alert('我的名字叫'+this.name);

},

showAge : function(){

alert('我今年'+this.age+'岁');

}

}

（2）构造函数方式

<script type="text/javascript">

function Person(name,age,job){

this.name = name;

this.age = age;

this.job = job;

this.showName = function(){

alert('我的名字叫'+this.name);

};

this.showAge = function(){

alert('我今年'+this.age+'岁');

};

this.showJob = function(){

alert('我的工作是'+this.job);

};

}

var tom = new Person('tom',18,'程序员');

var jack = new Person('jack',19,'销售');

</script>

# 三 DOM 文档对象模型

Document Object Model

getElementById() 根据id获得元素对象

getElementsByTagName() 根据标签名获得元素对象

getElementsByClassName()根据class值获得元素对象

previousElementSibling：上一个兄弟元素

nextElementSibling：下一个兄弟元素

parentNode：获取父级元素

firstElementChild：第一个子元素

lastElementChild：最后一个子元素

innerText获得或设置元素的文本内容

innerHTML获得或设置元素的html格式的内容

value 获取或设置输入控件的值

## 1 改变HTML

<script type="text/javascript">

//根据id获得元素对象

var ele = document.getElementById("p1");

//获得元素的文本内容

//alert(ele.innerText);//hello world

//获得元素的html格式的内容

//alert(ele.innerHTML);//<b>hello world</b>

//根据标签名获得元素对象

var tagEles = document.getElementsByTagName("p");

//alert(tagEles);

//alert(tagEles.length);

//alert(tagEles[1].innerHTML);

//根据class值获得元素对象

var cEles = document.getElementsByClassName("pclass");

//alert(cEles[1].innerHTML)

var nameEle = document.getElementById("name");

//获得表单控件中输入的值

//alert(nameEle.value);

//赋值

//ele.innerText = "<b>aaaa</b>";

ele.innerHTML = "<b>aaaaa</b>"//通过innerHTML赋值，内容中的html标签会被浏览器解析

nameEle.value = "hello";

</script>

<script type="text/javascript">

//创建标签

var trEle = document.createElement("tr");

var tdEle1 = document.createElement("td");

tdEle1.innerHTML = "zhangsan";

var tdEle2 = document.createElement("td");

tdEle2.innerHTML = "20";

//追加子元素

trEle.appendChild(tdEle1);

trEle.appendChild(tdEle2);

var tEle = document.getElementById("t");

tEle.appendChild(trEle);

</script>

## 2 改变CSS

<script type="text/javascript">

var pEle = document.getElementById("p1");

//设置元素的css样式

pEle.style.background = "gray";

pEle.style.color = "red";

//设置、删除属性

pEle.setAttribute("style","color:red; background-color: gray;");

pEle.setAttribute("class", "ccc");

pEle.removeAttribute("class");

pEle.removeAttribute("style");

</script>

## 3 事件

事件是可以被 JavaScript 侦测到的行为。 网页中的每个元素都可以产生某个可以触发[JavaScript](https://baike.baidu.com/item/JavaScript" \t "_blank)函数的事件。比如，用户点击按钮时产生一个 onclick 事件来触发一个指定的操作

<!-- onmouseout="alert(123);" -->

<!-- 给元素绑定一个点击事件 -->

<!-- this 代表操作的元素对象 -->

<div id="div1" onclick="divClick()" onmouseover="divMouseOver(this)" onmouseout="divMouseOut()">

click me!

</div>

<script type="text/javascript">

//触发的事件对应的操作

function divClick(){

var divEle = document.getElementById("div1");

divEle.style.backgroundColor = "red";

}

function divMouseOver(divEle){

divEle.innerHTML = "over";

}

function divMouseOut(){

var divEle = document.getElementById("div1");

divEle.innerHTML = "Click me!";

}

</script>

<input id="in1" type="text" />+<input id="in2" type="text"/>

<button id="btn">求和</button>

<span id="vspan"></span>

<script type="text/javascript">

var btnEle = document.getElementById("btn");

//动态添加事件

btnEle.onclick = function(){

var in1Ele = document.getElementById("in1");

var in2Ele = document.getElementById("in2");

var v1 = parseInt(in1Ele.value);

var v2 = parseInt(in2Ele.value);

var spanEle = document.getElementById("vspan")

var v = v1 + v2;

spanEle.innerHTML = "<b>" + v + "</b>";

}

</script>

Onload事件：

页面完成加载触发的事件

<script type="text/javascript">

//web页面加载完成后执行的事件

window.onload = function(){

var pEle = document.getElementById("p1");

pEle.innerHTML = "onload";

}

</script>

# 四 正则表达式

## 1什么是正则表达式

正则表达式(regular expression)描述了一种字符串匹配的模式（pattern），可以用来检查一个串是否含有某种子串、将匹配的子串替换或者从某个串中取出符合某个条件的子串等

## 2正则表达式的基本语法

/正则表达式主体/修饰符(可选)

Js中可以通过如下方式创建正则表达式的对象：

var re=new RegExp('规则', '可选参数');

var re=/规则/参数;

## 3 正则表达式规则

1）普通字符匹配：

如：/a/ 匹配字符 ‘a’，/a,b/ 匹配字符 ‘a,b’

2）转义字符匹配：

\d 匹配一个数字，即0-9

\D 匹配一个非数字，即除了0-9

\w 匹配一个单词字符（字母、数字、下划线）

\W 匹配任何非单词字符。等价于[^A-Za-z0-9\_]

\s 匹配一个空白符

\S 匹配一个非空白符

. 匹配一个任意字符，除了换行符‘\n’和回车符‘\r’

\1 重复子项

3）量词：对左边的匹配字符定义个数

? 出现零次或一次（最多出现一次）

+ 出现一次或多次（至少出现一次）

\* 出现零次或多次（任意次）

{n} 出现n次

{n,m} 出现n到m次

{n,} 至少出现n次

4）任意一个或者范围

[abc123] : 匹配‘abc123’中的任意一个字符

[a-z0-9] : 匹配a到z或者0到9中的任意一个字符

5）限制开头结尾

^ 以紧挨的元素开头

$ 以紧挨的元素结尾

6）修饰参数：

g： global，全文搜索，默认搜索到第一个结果接停止

i： ingore case，忽略大小写，默认大小写敏感

m： multiple lines，多行搜索（更改^ 和$的含义，使它们分别在任意一行对待行首和行尾匹配，而不仅仅在整个字符串的开头和结尾匹配）

## 4常用函数

test

用法：正则.test(字符串) 匹配成功，就返回真，否则就返回假

search

用法：字符串.search(正则) 匹配成功，返回位置，否则返回-1

match

用法： 字符串.match(正则)；匹配成功，就返回匹配成功的数组，否则就返回null

replace

用法：字符串.replace(正则，新的字符串) 匹配成功的字符去替换新的字符

五 表单验证

JavaScript 可用来在数据被送往服务器前对 HTML 表单中的这些输入数据进行验证。

表单数据经常需要使用 JavaScript 来验证其正确性，比如，可以验证：

验证表单数据是否为空

验证输入是否是一个正确的email地址

验证表单输入内容是否为数字型

等等

<!-- 点击submit按钮，会触发onsubmit事件，如果返回false，不提交数据 -->

<form action="index.html" method="get" onsubmit="return formTest();">

<!-- onfocus 获得焦点触发的事件

onblur 失去焦点时触发的事件

-->

用户名<input id="nid" type="text" name="name" onfocus="textfocus()" /><span id="t1"></span><br/>

电话<input id="tid" type="text" name="tel" /><span id="t2"></span><br/>

<input type="submit" value="submit"/>

</form>

<script type="text/javascript">

function formTest(){

var nameEle = document.getElementById("nid");

var telEle = document.getElementById("tid");

var name = nameEle.value;

if(name == null || name == ""){

var t1Ele = document.getElementById("t1");

t1Ele.innerHTML = "<font color='red'>用户名必须输入</font>";

return false;

}

var tel = telEle.value;

if(tel == null || tel == ""){

var t1Ele = document.getElementById("t2");

t1Ele.innerHTML = "<font color='red'>电话必须输入</font>";

return false;

}

//js中的正则表达式写在 “/正则表达式 /“之间

var reg = /^1[3578]\d{9}$/;

//test函数，验证是否符合正则表达式

if(!reg.test(tel)){

var t1Ele = document.getElementById("t2");

t1Ele.innerHTML = "<font color='red'>电话格式不正确</font>";

return false;

}

}

function textfocus(){

var t1Ele = document.getElementById("t1");

t1Ele.innerHTML = "";

}

</script>

# 六 BOM 浏览器对象模型

Browser Object Model

所有浏览器都支持 window 对象。它表示浏览器窗口

## 1 弹出窗体

可以在 JavaScript 中创建三种消息框：警告框、确认框、提示框

<script type="text/javascript">

function alertClick(){

window.alert("普通提示框");

}

function confirmClick(){

//提供选择是、否按钮的对话框

var ret = window.confirm("是否取钱");

if(ret){

alert("取钱");

}else{

alert("不取钱");

}

}

function promptClick(){

//自带文本输入框的对话框

var money = window.prompt("请输入金额", 0);

if(money){

//定位到某个页面

window.location.href = "09返回.html";

}

}

</script>

## 2 其他

Location对象

History对象

<button onclick="window.history.back();">返回上一级</button>

# 七 其他内置对象

## 1 日期类

var date = new Date();//获得当前时间

//alert(date);

//获得年

alert(date.getFullYear());

//月

alert(date.getMonth() + 1);//0-11

//日

alert(date.getDate());

//星期

alert(date.getDay());

//时

alert(date.getHours());

//分

alert(date.getMinutes());

//秒

alert(date.getSeconds());

//时间戳，从1970年1月1日0时0分0秒到date表示时间的毫秒数

alert(date.getTime());

## 2 Math对象

//向上取整

//alert(Math.ceil(13.4))

//向下取整

//alert(Math.floor(13.6))

//获取0-1的随机数

//alert(Math.random())

//四舍五入为整数

//alert(Math.round(12.345))

//alert(Math.round(12.345 \* 100) / 100)

//求两值中的最大值

//alert(Math.max(12, 34))

//求x的y次幂

//alert(Math.pow(12, 2))

//获取绝对值

//alert(Math.abs(-12.3))

# 八 定时器

setInterval()：按照指定的周期（以毫秒计）来调用函数或计算表达式。

setTimeout()：在指定的毫秒数后调用函数或计算表达式。

<script type="text/javascript">

function showTime(){

var date = new Date();

var h = date.getHours();

var m = date.getMinutes();

var s = date.getSeconds();

var timeEle = document.getElementById("timeSpan");

timeSpan.innerText = h + ":" + m + ":" + s;

}

//定时执行一个操作，第二个参数表示时间间隔，单位毫秒

setInterval(showTime, 1000);

</script>