01\_如何使用javascript

如何在页面中使用javascript脚本语言

\* \* 使用<script>标签

\* \* type：设置使用脚本语言

\* \* language：设置使用脚本语言

\* \* src：设置路径，引入外部独立的js文件

\*

\* javascript脚本语言：在页面中任意位置都可以使用

\* \* 一般情况：head中、head与body之间、body后

\* \* head中：一般情况都是用于引入外部js文件

\* \* head与body之间：

\* \* 不能获取页面元素

\* \* 定义函数

\* \* 函数内：获取页面元素。只定义不调用

\* \* body后：

\* \* 获取页面元素

\* \* 定义函数：既定义又调用

02\_ECMAScript入门

区分大小写：与java习惯一样

\* javascript中的变量，只有在初始化时，才知道类型

\* 定义不清楚变量的类型：弱类型

\* 在javascript中，使用变量时，按照java编写的习惯

//java中，每行最后没有分号：编译报错

//javascript中，每行最后可以没有分号。但是，依旧按照java的编写习惯

//ECMAScript：只说这些内容，但是所有内容还是比较多。大部分的内容与java保持一致。

01\_函数

//javascript定义函数的三种方式

//1 普通方式定义函数

/\*

\* function 方法名(参数){

\* 方法体

\* }

\*/

//function add(a,b){

// return a + b;

//}

//alert(add(1,1));

//2 构造函数方式定义函数（Function要大写）

/\*

\* var 变量名 = new Function(参数n,方法体);

\*

\* javascript中，Function类型

\*/

// var add = new Function('a','b','return a+b;');

// alert(add(1,2));

//3 直接量方式定义函数(没有new 关键字)

/\*

\* var 变量名 = function(参数){

\* 方法体

\* }

02\_window对象

//访问window对象下的属性和方法，是可以省略"window."

//window对象的属性

//1 特殊属性：window对象下的其他对象，可以作为入口对象，访问其它5大对象(window,navigator,history,location,screen,document)

//Window 对象的方法

//1 提示框

// window.alert("xxx");

//2 确认框

// var flag = window.confirm("你确认要退出吗?");

// //confirm()方法的返回值是boolean类型，true是确认，false是取消

// alert(flag);

//3 标准输入框：弃用！

//prompt()方法：第一个参数：提示信息；第二个参数：默认值

window.prompt("请输入你的用户名：","username");

//4 open()与close()方法：打开和关闭页面

//5 setTimeout()和clearTimeout()

// 定时器功能，设置多长时间后var id\_of\_settimeout = setTimeout(code,millisec)，执行什么操作，取消执行操作clearTimeout(id\_of\_settimeout)。

//6 setInterval()和clearInterval()

//定时器功能，设置每多长时间执行什么操作var id\_of\_setinterval = setInterval(code,millisec[,"lang"])，和取消clearInterval(id\_of\_setinterval)

Form对象

//1 获取页面中的第一个表单的action属性

//表单对象是数组对象，默认获取页面中所有表单内容

// //1 获取表单对象的方式：document.表单对象[索引值]

// var form1 = document.forms[0];

//2 获取页面中的第二个表单的action属性

//2 通过表单对象的name属性来获取：document.表单名

var form2 = document.form2;

01\_查找页面元素

//要求：获取页面中id="username"的元素

// var username = document.getElementById("username");

//要求：获取页面中name="username"的元素

// var usernames = document.getElementsByName("username");

//要求：获取页面中所有input标签

// var inputs = document.getElementsByTagName("input");

02\_创建节点

创建节点：

\* \* 创建元素节点：createElement(标签名)

\* \* 创建属性节点

\* \* 创建文本节点：createTextNode(文本内容)

获取某节点的属性

\* \* 节点名.属性名：除了id和name属性

\* \* 节点名.getAttribute(属性名)：所有属性都可以获取

\*

\* javascript的时候：很多时候，语法上是正确的，但是效果上是不对的。

\* \* 在语法上，有很多方式可以做到想要到的效果。但是，不是所有方式效果都对。

\* \* 在编写javascript时，是应该以最终的效果为准的。

\*

\* 浏览器解析javascript，主要是由浏览器内核来决定的：

\* \* IE：三个版本，1)IE6.0会让IE7.0；2)IE8.0和IE9.0；3)IE10和IE11

\* \* webkit内核为主：苹果的产品

\* \* 苹果浏览器

\* \* 谷歌浏览器

\* \* 火狐浏览器

\* \* 几乎所有第三方浏览器

\* \* 谷歌浏览器：即将推出自主内核

\* \* 火狐浏览器：即将推出自主内核

\* \* 众多的国产浏览器：

\* \* 百度浏览器：号称自主内核

\* \* 傲游浏览器：号称自主内核

\* \* 国产浏览器情况比较复杂，不做考虑。

\* this代表了北京节点

\* \* this的用法：必须要有上下文的情况下使用

\* \* this的用法：明天还会再研究

\* \* this的用法：到目前为止，没有能总结的特别好的。规律很难抓到。

\* \* 建议：会使用的话，最好。不会使用的话，尽量别用!

\* WEB1.0与WEB2.0的区别：

\* \* WEB1.0：垂直门户网站(以内容为主)

\* \* WEB2.0：社交网站(以人的关系为主)

\* \* WEB3.0：基于移动互联网的社交网站

Arguments 对象

\*

\* \* java中，函数里面：函数的重载

\* \* 节省的函数命名空间

\* \* 执行效率比较高，性能好

\*

\* \* javascript中，函数里面：能不能实现函数的重载？

\* \* 不能实现函数重载

\* \* 同名的函数中，最后一个函数会覆盖前面所有同名函数

\*

\* \* 想办法：模拟出java中函数的重载效果！

\* \* arguments.length：获取到是参数的个数

\* \* 利用Arguments 对象的特性，模拟实现了函数重载的效果

\* 变量的作用域

\* \* 变量a是全局变量

\* \* 定义函数fn()，全局函数

\* \* 定义变量b是局部变量，fn()函数域

\*

\* \* 在全局环境中，可以访问

\* \* 全局变量a

\* \* 全局函数fn()

\* \* fn()函数域的变量b是访问不到

\* \* 访问局部变量b，在fn()函数域中访问到

\* 定义局部变量b时，没有使用var修饰符

\* \* 第一次调用变量b，没有定被义

\* \* 第二次调用函数fn()

\* \* 在javascript中，不使用var修饰符时，环境会自动增加var修饰符

\* \* 而且，环境会将变量b，定义为一个全局变量

\* 全局变量与局部变量同名：局部变量的优先级高于全局变量

\* \* 当代码执行到不调用函数fn()

\* \* 定义一个全局变量a，并且初始化

\* \* 定义一个全局函数fn()

\* \* 定义了局部变量a，并没有初始化

\* \* 调用函数fn()时

\* \* 打印变量a，是局部变量a

\* \* 打印变量a，是局部变量a

\* \* 两次打印，都是打印局部变量a

几种特殊的函数

\* \* 匿名函数：没有名的函数

\* \* 可以将匿名函数作为参数传递给其他函数。这样，接收方函数就能利用所传递的函数来完成某些事情.

\* \* 可以定义某个匿名函数来执行某些一次性任务.

\* 回调函数：一个函数作为另一个函数的参数被调用，作为参数的函数就是回调函数

\* 定义函数fn()，接收的参数是一个函数

\* \* 调用函数fn()时，所传递的one和two都是一个函数

\* \* one和two函数是回调函数

\*

\* \* 改写后的内容：匿名回调函数

\*

\* 回调函数的优点：

\* \* 它可以在不做命名的情况下传递函数（这意味着可以节省全局变量）。

\* \* 可以将一个函数调用操作委托给另一个函数（这意味着可以节省一些代码编写工作）。

\* \* 回调函数也有助于提升性能。

\* 当页面加载完毕，进行必要的初始化工作

\* \* 定义一个函数来完成：body标签中，onload="函数名()"

\* \* 定义一个函数来完成：定义函数，调用

\*

\* 自调函数：定义及调用

\* \* 第一个小括号：封装了一个匿名函数

\* \* 第二个小括号：负责调用封装的匿名函数

\* \* 如果封装的匿名函数接收参数，在第二个小括号中可以传递参数

\*

\* 自调函数：

\* \* 不用手动去设置调用的内容

\* \* 也有助于提升性能。

\* 内部（私有）函数：java中内部类来学习

\* \* 有助于确保全局名字空间的纯净性（这意味着命名冲突的机会很小）。

\* \* 私有性 —— 只将一些必要的函数暴露给“外面世界”，并保留属于自己的函数，使它们不为该应用程序的其他部分所用。

//返回函数的函数

// function fn(){

// var a = "javascript";

//

// return function(){

// return a;

// }

// }

// alert(fn()());

作用域链???????????????????

闭包 函数可以使用函数之外定义的变量

//循环中的闭包

/\*

\* 定义全局函数f()

\* \* 定义了一个空的数组

\* \* 定义了一个局部变量i

\* \* 循环中，i的值：0,1,2,3；循环完毕后，i的值是3

\* \* 循环体中，i的值是0,1,2

\* \* 访问数组：a[0],a[1],a[2]

\* \* 数组中每一块内容是："function(){return i;}"作为一个字符串返回

\* \* 函数f()，返回值是数组a

定义对象的三种方式

//1 new Object：定义一个空的对象

//2 对象直接量

//3 定义函数对象：在javascript中，函数即对象，对象即函数

//以下哪个描述是错误的？D

// var a = {}; //定义一个空的对象

// var b = []; //定义一个空的数组

// var c = //; //定义一个空的正则表达式

// var d = (); //错误的