JavaScript

2020年2月18日

14:31

学习内容

1. JavaScript语法
2. JavaScript内部对象
3. JavaScript的DOM
4. BOM
5. 自定义函数
6. 全局函数
7. 事件处理

基本语法

2020年2月18日

15:17

引入方式

* 1. <script>…</script>
  2. <script src="address.js"></script>

简单数据类型

1.数值类型 number

2.字符串类型 string

3.布尔值 boolean

4.数组 var arr = new Array(); var arr = [];

5.JS对象 var person={

firstname:"wang",

lastname:"hao",

id：666

}；

6.undefined null

基本语句：

If(){}else{}

JavaScript内部对象

2020年2月18日

15:20

String 方法

|  |  |
| --- | --- |
| **方法** | **说明** |
| .length #不加括号的是属性 | 返回长度 |
| .trim() #得到一个新值 | 移除空白 |
| .trimLeft() | 移除左边的空白 |
| .trimRight() | 移除右边的空白 |
| .concat(value, ...) #s1='hello';s.concat('xx');得到helloxx | 拼接 |
| .charAt(n) #n类似索引，从0开始，超过最大值返回''空字符串 | 返回第n个字符 |
| .indexOf(substring, start) #这个start是从索引几开始找，没有返回-1 | 子序列位置 |
| .substring(from, to) #不支持负数，所以一般都不用它 | 根据索引获取子序列 |
| .slice(start, end) #var s1='helloworld';s1.slice(0,-5) | 切片 |
| .toLowerCase() #全部变小写 | 小写 |
| .toUpperCase() #全部变大写 | 大写 |
| .split(delimiter, limit)#分隔,s1.splite(' '),后面还可以加参数s1.split(' '，2),返回切割后的元素个数 | 分割 |

数组

数组的创建

var colors = ['red','green','yellow'];

var colors = new Array();

|  |  |
| --- | --- |
| **方法** | **说明** |
| .length | 数组的大小 |
| .push(ele) | 尾部追加元素 |
| .pop() | 弹出并获取尾部的元素 |
| .unshift(ele) | 头部插入元素 |
| .shift() | 弹出并获取头部元素 |
| .slice(start, end) | 切片 |
| .reverse() #在原数组上改的 | 反转 |
| .join(seq)#a1.join('+')，seq是连接符 | 将数组元素连接成字符串 |
| .concat(val, ...) #连个数组合并,得到一个新数组，原数组不变 | 连接数组 |
| .sort() | 排序 |
| .forEach() | 将数组的每个元素传递给回调函数 |
| .splice() #参数：1.从哪删(索引), 2.删几个 3.删除位置替换的新元素(可多个元素) | 删除元素，并向数组添加新元素。 |
| .map() | 返回一个数组元素调用函数处理后的值的新数组 |

日期

Var myDate= new Date();

|  |  |
| --- | --- |
| **语法** | **含义** |
| getDate() | 返回指定日期对象的月份中的第几天(1-31) |
| Date() | 返回当前时间 |
| getMonth() | 返回指定日期对象的月份(0-11) |
| getFullYear() | 返回指定日期对象的年份 |
| getDay() | 返回指定日期对象的星期中的第几天(0-6,从周日开始算) |
| getHours() | 返回指定日期对象的小时(0-23) |
| getMinutes() | 返回指定日期对象的分钟(0-59) |
| getSeconds() | 返回指定日期对象的秒数(0-59) |

RegExp对象

//正则规则内不要包含不想要的空格

var reg = RegExp('正则表达式') //注意,写在字符串中所有带\的元字符都会被转义,应该写作\\

var reg2 = /正则表达式/ //内部的元字符就不会转义了

reg.test('待检测的字符串') //如果字符串中含有符合表达式规则的内容就返回true,否则返回false

Math

|  |  |
| --- | --- |
| **方法** | **含义** |
| Math.floor() | 向下取整 |
| Math.ceil() | 向上取整 |
| Math.max(a,b) | 求a和b中的最大值 |
| Math.min(a,b) | 求a和b中的最小值 |
| Math.random() | 0-1之间的随机数 |
| Math.abs(x) | 返回x的绝对值 |
| Math.round(x) | 返回四舍五入的整数 |

数据类型之间的转换

字符串转数字 parseInt()

字符串转小数 parseFloat()

转字符串 String(num)和num.toString()

DOM

2020年2月18日

15:23

DOM(Document Object Model 文档对象模型）就是对HTML标签属性内容的操作

* HTML DOM 允许 JavaScript 改变 HTML 元素的内容
  + getElementByID
  + getElementsByTagName
  + getElementsByClassName
  + getElementsByName
* HTML DOM 允许 JavaScript 改变 HTML 元素的样式
  + document.getElementById(id).style.property=新样式
* getAttribute()
* setAttribute(attribute,value)

DOM常见属性

1. nodeName
2. nodeValue
3. nodeType
4. firstChild、lastChild
5. parentNode

DOM节点操作

* 创建一个元素节点 createElement("标签名")
* 创建一个文本节点 createTextNode("文本值")
* 为元素节点添加子节点 appendChild() 例：var reference = element.appendChild(newChild)；
* 插入节点 insertBefore() 例：var reference = element.insertBefore(newNode,targetNode);
* 删除节点 var reference = element.removeChild(node)

DOM常用事件

onclick 
ondbclick 
on mousedown 
onmouseup 
onmousemove 
onmouseover 
onmouseout 
onkeypress 
onkeydown 
onkeyup 

页 面 窗 囗 事 件 
onabort 
onbeforeunload 
Onerror 
onload 
On 《 Ove 
OnreSlZe 
onscroll 
onstop 
说 明 
图 片 在 下 载 帔 用 户 中 断 
当 前 页 面 的 内 容 将 要 帔 改 变 触 友 此 事 件 
出 现 满 误 触 友 此 事 件 
页 面 内 容 完 成 触 友 此 事 件 
览 器 的 窗 囗 帔 移 动 触 友 此 事 件 
当 览 器 的 窗 囗 大 小 帔 改 变 触 友 此 事 件 
览 器 的 滚 动 条 位 罟 友 生 变 化 触 友 此 事 件 
览 器 的 停 止 帻 帔 帻 下 触 友 此 事 件 或 者 止 在 下 载 的 文 件 帔 中 断 
当 前 页 面 将 帔 改 变 触 友 此 事 件 

Onblur 
onchange 
onfocus 
on reset 
onsubmit 

BOM

2020年2月18日

15:42

BOM浏览器对象模型

bom结构图

Browser Objects (80M) 
window 
frames[ ] 
DOM 
history 
location 
applets[ ] 
images[ ] 
navigator 
areas[ ] 
layers[ ] 
screen 
embeds[ ] 
anchors[ ] 
document 
forms[ ] 

DOM是BOM的一部分

间歇调用和超时调用

* + 超时调用

setTimeout(); 只执行一次，返回一个数值id

参数：1.匿名函数内放要执行的代码；2.时间，单位毫秒

使用clearTimeout(id);通过id值取消指定的超时调用

* + 间歇调用

setInterval(); 重复执行，返回一个数值id

参数：1.匿名函数内放要执行的代码；2.时间，单位毫秒

使用clearInterval(id);通过id值取消指定的超时调用

弹出系统对话框

alert(); //不同浏览器中的外观是不一样的  
confirm(); //兼容不好  
prompt(); // 不推荐使用

打开关闭窗口

window.open(url, target);

url：要打开的地址。

target：新窗口的位置。可以是：\_blank, \_self, \_parent父框架。

window.close();

location对象

location对象的属性

href：跳转

hash：返回url中#后面的内容，包括#

host：主机名，包括端口

hostname：主机名

pathname：url中的路径部分

protocol：协议 一般是http、https

search：查询字符串

window.location.reload();

特别对象

**navigator对象**

window.navigator的一些属性可以获取客户端的一些信息。

* + userAgent：系统-浏览器
  + platform：浏览器支持的系统， win/mac/linux

console.log(navigator.userAgent);  
console.log(navigator.platform);

**history对象**

1.后退

* + history.back()
  + history.go(-1)      0是刷新

2.前进

* + history.forward()
  + history.go(1)

HTML5 存储技术 localStorage sessionStorage

sessionStorage(临时存储)：为每一个数据源维持一个存储区域，在浏览器打开期间存在，包括页面重新加载。

localStorage(长期存储)：与sessionStorage一样，但是浏览器关闭后，数据依然会一直存在

示例：

1.保存数据到本地

let info = {  
 name: 'Lee',  
 age: 20,  
 id: '001'  
};

sessionStorage.setItem('key', JSON.stringify(info));   
localStorage.setItem('key', JSON.stringify(info));

2.从本地存储获取数据

var data1 = JSON.parse(sessionStorage.getItem('key'));  
var data2 = JSON.parse(localStorage.getItem('key'));

3.本地存储中删除某个保存的数据

sessionStorage.removeItem('key');  
localStorage.removeItem('key');

4.删除所有保存的数据

sessionStorage.clear();  
localStorage.clear();

5.监听本地存储的变化

Storage 发生变化（增加、更新、删除）时的 触发，同一个页面发生的改变不会触发，只会监听同一域名下其他页面改变 Storage

window.addEventListener('storage', function (e) {  
 console.log('key', e.key);  
 console.log('oldValue', e.oldValue);  
 console.log('newValue', e.newValue);  
 console.log('url', e.url);  
 })

**浏览器查看方法**

* + 进入开发者工具
  + 选择 Application
  + 在左侧 Storage 下 查看 Local Storage 和 Session Storage

全局属性与全局函数

2020年2月18日

15:42

|  |  |
| --- | --- |
| **属性** | **描述** |
| Infinity | 表示正/负无穷大的数值 |
| NaN | "Not-a-Number" 值 |
| undefined | 表示尚未为变量分配值 |

|  |  |
| --- | --- |
| **函数** | **描述** |
| decodeURI() | 解码URI |
| decodeURIComponent() | 解码URI组件 |
| encodeURI() | 对URI进行编码 |
| encodeURIComponent() | 对URI组件进行编码 |
| escape() | 在1.5版中已弃用。 请改用encodeURI()或encodeURIComponent() |
| eval() | 计算字符串并执行它，就好像它是脚本代码一样 |
| isFinite() | 确定值是否为有限的合法数字 |
| isNaN() | 确定值是否为非法数字 |
| Number() | 将对象的值转换为数字 |
| parseFloat() | 解析字符串并返回浮点数 |
| parseInt() | 解析一个字符串并返回一个整数 |
| String() | 将对象的值转换为字符串 |
| unescape() | 在1.5版中已弃用。 请改用decodeURI()或decodeURIComponent() |

事件处理

2020年2月18日

15:42

见DOM内容

js弹窗

2020年5月12日

21:47

1. 警告框

alert（“”）

1. 确认框

confirm（“”）

1. 提示框

prompt（“文本”，“默认值”）

js对象

2020年5月12日

21:58

如何创建对象

// 创建js对象

person = new Object();

person.firstname = "Bill";

person.lastname = "Gates";

person.age = 56;

person.eyecolor = "blue";

// 或者

person = {firstname:"Bill",lastname:"Gates",age:56,eyecolor:"blue"};

// 使用对象构造器

function Person(name,age,gender){

this.name = name;

this.age = age;

this.gender = gender;

this.say=function(){

console.log(this.name+this.age+this.gender);

}

}

var person = new Person("zenghao",25,"nan");

person.say();

console.log(person);