一、JAVASCRIPT

2018年11月1日 15:07

1. js概述

- 基于对象的一门语言。
- 一门独立的语言。
- 脚本语言，没有编译过程，解释运行。
- 主要应用在客户端，在服务器端也有所应用（Node.js）。
  - 特点：
    脚本语言没有编译过程
    基于对象
    弱类型
  - 特性：
    交互性
    安全性
    跨平台性

2. JavaScript不得不说的历史

和JAVA的关系：语法十分相似，但其实毫无关系。

1995年5月，Netscape，LiveScript

1995年12月，改名为JavaScript

1996年8月，微软，Jscript

1997-1999年，ECMA，ECMAScript，基于已有的JavaScript和Jscript，提出了标准的Script语法规则，JavaScript和Jscript都遵循这套标准。

3. js和html的结合

- 方式一：在html中的head标签中书写<script>标签，在标签内部写js代码。
- 方式二：引入外部文件中的js代码。在head标签中书写<script>标签，通过标签的src属性引入外部js文件中的js,script标签进行外部引入时，不能进行自闭，否则无效。

- js代码在浏览器加载到时直接运行。
- js是弱类型的语言，所以变量定义时不需要指定类型，变量可以指向任意类型的数据。
- js中的分号可以写也可以不写，js解析引擎自动会在需要的地方拼接分号，但是不要养成这种习惯，最好在必要的地方都加上分号。

4. js的语法-数据类型

JS中的数据类型分为基本数据类型和复杂数据类型。

- 基本数据类型

  数值类型(Number)、字符串(String)、布尔类型(Boolean)、undefined、null

  - Number

    代表数字的基本类型。

    JS不区分整形和浮点型，JS中的所有数值类型底层实现都是浮点型。

    - 数值类型中有如下几个特殊值
      - Infinity 无穷大

-Infinity 负无穷大

- NaN 非数字，非数字非常特殊，和任何值都不行等，包括本身，即NaN==NaN的值为false。可以通过isNaN()判断某值是否为非数字，true表示确实为非数字的值，false表示为数字的值。
- 数值类型是基本数据类型，但JS本身提供了对应的包装对象Number，具有和数值处理相关的属性和方法。
  - Number提供的属性：

    Number.MAX_VALUE 可表示的最大数字

    Number.MIN_VALUE 可表示的最小数字

    Number.NaN 非数字值

    Number.POSITIVE_INFINITY 正无穷大

    Number.NEGATIVE_INFINITY 负无穷大

String

- Js中的字符串是基本数据类型，字符串常量必须用双引号引起来。
- JS中提供了字符串的包装对象String，提供了字符串处理相关的属性和方法。
  - String对象提供的属性：

    length：字符串中字符数
  - String 对象提供的方法：

    参考JS文档

Boolean

- Boolean类型只有两个值，true、false。
  - 与或非

JS中提供了布尔类型的包装对象Boolean，提供了布尔类型处理的相关属性和方法。

参考JS文档

Undefined

Undefined类型只有一个值就是undefined，表示变量未定义，当一个变量未初始化时，值为undefined。

Null

Null只有一个值就是null。null用来表示尚未存在的对象，常用来表示函数企图返回一个不存在的对象。

复杂数据类型

对象、数组、函数

js中数据类型的自动转换

js在需要时会自动对类型进行转换，转换的规则：

数字	可以在需要时转换为对应的字符串形式。0会转换位false，其他数字会转换为true.在需要对象时可以自动转换位Number对象。
字符串	可以在需要时转换位对应的数值，需要注意当数值和字符串进行加法运算时，会处理为字符串的拼接，所以需要通过parseInt或parseFolat将字符串强制转换为数值类型参与运算。非空字符串转换为ture，空字符串转换为false。需要对象时自动转换位String对象

布尔类型	true转换位1，false转换位0.转换为字符串的"true"和"false".转换位Boolean对象
对象类型	如果为null则转成字符串"null",如果为null转换为false

5. 定义变量

JS中有数据类型，但是JS的引用是不区分类型的，所以称JS为弱类型，即，一个引用可以先指向数字类型，后再指向字符类型，还可以再指向对象类型。如下 的代码是没有问题的。

```
var x = 99;
x = "aa";
x = true;
x = new Object();
```

JS中定义对象的时候可以不使用var来定义，这样定义的变量将成为全局变量，在任何一个位置都可以使用。

```
function mx(){
    x = "123"
}
mx();  //mx()方法执行过后x才被定义出来。
console.log(x);
```

6. js语法--运算符

Javascript中的运算符和Java大致相同。

只是运算过程中需要注意几点：

```
var x = 3120/1000*1000; x = 3120;而不是3000。
var x = 2.4+3.6 ; x = 6;而不是6.0
var x = “12” + 1; x = “121”; x = “12” - 1 ; x = 11;
```

- 加号对于字符串是连接符
- && || 是逻辑运算符 & | 是位运算符。
- 也支持三元运算符 ？：
  - 2+3>5?console.log("aaa"):console.log("bbb")
- 特殊运算符 typeof：返回一个操作表达式的数据类型的字符串。

```
var x = 3;
var y = "123";
var z = false;
typeof(x); //number
typeof(y); //string
typeof(z); //boolean
```

7. js语法--语句

- if语句：

  var x = 3;

  情况1：if(x==4)//可以进行比较运算。

  情况2：if(x=4)//可以进行赋值运算，而且可以同样进行判断。不报错。

  原因：因为在JS中0或者null就是false，非0或者非null就是true。if(x=4)是先将x赋值为4，然后对值为4的x进行判断，4会被认为是true，所以结果是true；

可以通过if(4==y)来解决该问题。因为4=y不会进行判断，而是会报错。

- switch case --与java中使用方式一致
- while、dowhile、for -- 不支持增强for循环
  --与java中使用方式一致

8. js语法--函数
   - js中的函数是一堆可执行代码的合集。在需要的时候可以通过函数的名字调用其中的代码。函数可以理解为一种特殊的对象，其实本质上就是一段可执行的字符串。
   - js中函数调用时，实参的数量可以和形参的数量不一致。如果实参少于形参，则形参依次赋值，没有被赋值到的形参取值位undefined。如果实参多余形参，则依次赋值，对于没有被赋值到的实参也不会丢失可以在方法中通过arguments来获取。
   - js中的函数可以认为是一种特殊的对象，可以任意的赋值给不同的引用甚至通过方法来当做参数传递，唯一特殊的是，通过跟一对小括号可以执行函数中的代码。其实js是一门解释运行的语言，函数的本质就是一段js代码的字符串，来回赋值、来回传递都可以，一旦跟一对小括号，就执行这段js代码字符串。
   - js中的函数具有自己的包装对象Function,提供了一些重要的属性和方法。
     - 方法1:普通方法定义函数

```
function fun1(参数列表){
    函数体
}
在需要调用时通过 fun1(实参) 方式调用。
```

     - 方法2:动态函数

```
var fun2 = new Function("a","b","方法体");
fun2("x","y")
var fun2x = fun1();
fun2x("t","u");
```

     - 方法3:匿名函数定义

```
var fun3 = function(参数列表){
    方法体
}
fun3();
```

9. js语法 - 数组 - Array - js中的数组，本质上就是一个用中括号括起来用逗号分割内容的字符串。

```
var arr = new Array();//定义一个长度为0的空数组
var arr = new Array(3);//定义一个长度为3的数组
var arr = new Array(1,2,3,6,8);//定义具有指定初始值的数组
var arr = [2,3,5,7];//数组直接量定义数组
```

   - 特点:
     - 可以存储不同类型的数据
     - 长度可以变化。
   - 案例:
     - 添加元素 push()

- 删除最后一个元素 pop()
- 删除第一个元素 shift()
- 遍历打印数组

10. js语法 - 对象

○ js的内置对象：

String -- 基本数据类型 字符串类型 的包装对象

Boolean -- 基本数据类型 布尔类型 的包装对象

Nubmer -- 基本数据类型 数值类型 的包装对象

Array -- 数组类型 的包装对象

Function -- 函数类型 的包装对象

Math -- 数据对象，封装了很多数学常量和数学方法

Date -- 日期时间对象，封装了很多和日期实现相关的方法

Global -- 全局对象。js中有一些方法和属性经常使用，但归到哪个对象上都不合适，所以js中有一个Global对象整合了这些方法和属性。Global中定义的方法和属性特点是属于全局，可以直接使用。

思考：parseInt("123abc123");打印结果。

RegExp --正则对象，保存有关正则表达式模式匹配信息的固有全局对象。Partten

邮箱正则。

○ 自定义对象

构造函数模拟了java中类的功能。js中的对象可以动态增加/删除属性和函数。 -- js中对象的本质就是用大括号起来的键值的集合，本质上是一段字符串，有点类似于java中的map。

□ 方法一：构造函数创建对象1

```
function Person(){}
var p = new Person();
p.name = "zhangfei";
p.age = 19;
p.say = function(){alert(this.name+"say.....")};

alert(p.name);
alert(p["age"]);
p.say();
//删除一个属性
delete  p.name
console.log(p);
//删除一个函数
delete p.say
console.log(p);
```

□ 方法二：构造函数构造对象2

```
function Person(name,age){
    this.name = name;
    this.age = age;
    this.say = function(){alert(this.name+"say....")}
}
var p = new Person("guanyu",20);
alert(p.name);
alert(p["age"]);
p.say();
```

□ 方法三：对象直接量定义对象

```
var p = {name:"liubei",age:19,sleep:function()
```

```
{alert(this.name+"sleep.....")}};
alert(p.name);
alert(p["name"]);
p.sleep();
```

- 对象的操作:
  - with语句:with语句定义了某个对象的作用域，在该域中可以直接调用该对象的成员。

```
var p = {name:"liubei",age:19,sleep:function()
{alert(this.name+"sleep.....")}};
with(p){
    alert(name);
    alert(age);
    sleep();
}
```

  - for...in语句:用来遍历对象的所有属性的名称

```
var p = {name:"liubei",age:19,sleep:function()
{alert(this.name+"sleep.....")}};
for(var x in p){
    alert(x);
}
```

  - delete语句:删除对象的属性

```
var p = {name:"liubei",age:19}
p.addr = "peixian";
alert(p.addr);
delete p.addr;
alert(p.addr);
```

11. JSON

JSON本质上就是一段字符串，能够保存较复杂关系的数据，具有良好的数据保存格式，又极为轻量，加之多种代码平台都支持对字符串的处理，所以我们可以使用JSON字符串进行数据的传入，甚至跨平台传输。

```
data = {
    name:"zs",
    age:19,
    addr:["bj,sh,gz"],
    wife:[
        {name:"苏荃",age:40,job:["教主夫人","大大老婆"]},
        {name:"建宁",age:20,job:["公主","小老婆"]},
    ]
}
```

查看当前JSON中第二个wife的工作

```
data["wife"][1]["job"];
```

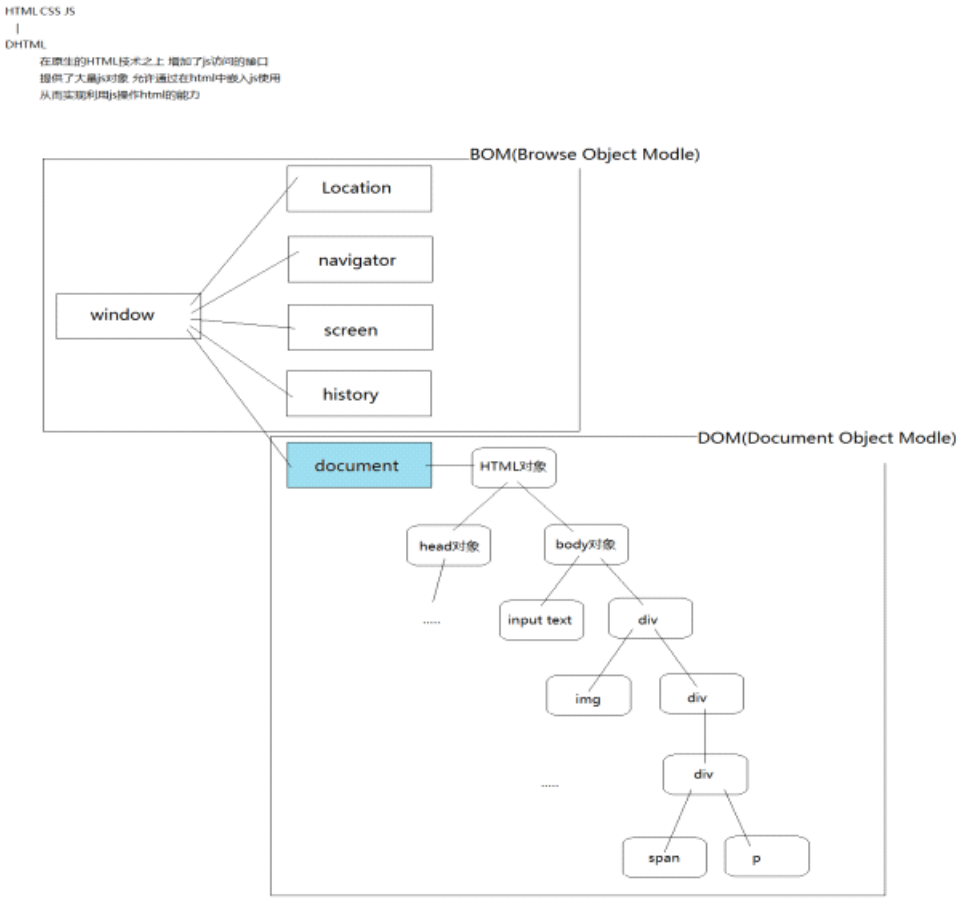
二、DHTML - 动态HTML

2018年11月4日 15:23

1. DHTML概述

- DHTML将浏览器加载html文档中的所有的内容都当做JS对象来处理。最终就组成了一颗由js对象组成的对象树。
- 通过操作代表html元素的js对象来操作html中的元素。
- 通过操作js对象组成树来操作html文档的结构。从而实现了html和js的结合，实现了可以通过js来操作html。
- DHTML可以分为由BOM(Browser Object Model)和DOM(Document Objecect Model)两部分组成。

2. BOM和DOM



3. BOM -- 浏览器对象模型

bom -- browser object model

(主要参考API文档，会使用其中的API即可)

- window:代表一个浏览器窗口的对象
  - 其中包含的方法：

    onblur:失去焦点

    onfocus:获得焦点

    !!!onload:当前浏览器页面装载完成后触发

    !!!alert

    !!!confirm

!!!prompt

!!!close (//仅限ie浏览器，chrom和firefox需要解决浏览器禁止js关闭非js创建页面的问题)

!!!setInterval

!!!setTimeout

setInterval 和 setTimeout的不同?

- 其中包含的对象：(我们可以通过window引出这些对象)

location

screen

history

navigator

document

- location

!!!href 获取或设置地址栏上的地址。通过此属性js可以控制浏览器访问一个新的地址。

- navigator
- history

length

back()

forward()

go()

4. DOM -- 文档对象模型

dom -- document object model

- 获取文档对象的方法

getElementById("id")	根据id获取一个元素
getElementsByName("name")	根据name获取一组元素
getElementsByTagName("tagname")	根据元素名称获取一组元素
innerHTML()	设置或获取位于对象起始和结束标签内的HTML
innerText()	设置或获取位于对象起始或结束标签内的文本

- 对文档对象进行增删改的操作

创建元素：

```
document.createElement("节点类型"); //为指定标签创建一个元素的实例。
```

挂载元素：

```
parentNode.appendChild(childNode);//在父元素最后位置添加子元素
parentNode.insertBefore(newNode,oldNode);//将元素作为父对象的子节点插入到文档层次结构中。
```

删除元素：

```
parentNode.removeChild(childNode)
```

修改元素：

```
parentNode.replaceChild(newNode,oldNode);
```

克隆节点：

```
div = div.cloneNode(boolean);//如果为false或者 不写(默认)，不复制克隆节点中的子节点，只复制指定的克隆节点。
                             //如果为true，复制当前节点及其子节点
```

调整样式：

通过修改元素的class属性，使元素使用不同的类来启用不同的样式。

```
div.className = "xxx";
```

通过元素的style属性来进行样式的修改。

```
div.style.backgroudColor="#f00";
```

通过修改元素display属性，调整节点展示方式：

```
div.style.display = "none"|"block";
```

扩展：nextSibling 获取对此对象的下一个兄弟对象的引用。

问题集

2018年11月5日　15:57

问题：dom解析时，在页面中写好一个function demo1(){}，适用button按钮调用这个方法，执行却发现报错，错误：ncaught ReferenceError: demo1 is not defined

（demo1未定义）

解决：

仔细检查<script> </script>中的function demo1(){}代码，少了大括号，会出现未定义的错误。