JavaScrip简介

2019年5月6日 星期一 9:00

1. javascript概述

   javascript是一门独立的语言。

   javascript是一门脚本语言。无需编译，直接解释执行。

   javascript是一门基于对象的语言。

   javascript是一门常用语浏览器端的语言，在后端也有技术的实现，Node. js

   特点：

   弱类型

   基于对象

   脚本语言

   特性：

   交互性

   安全性

   跨平台性

2. javascript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

   a. 引入方式：

      i. 将javascript书写在head标签当中，书写格式如下：

      ```
      <script type="text/javascript"></script>
      ```

      ii. 在head标签中书写一个script标签对，通过起始标签的src属性引入一个外部的js文件。

   b. 浏览器中内置一个js解析器，会对浏览器加载到的js语句逐行执行，如果当前js语句没有添加分号，则js解析器会自动拼接上一个分号。注意：在这里建议大家手动添加上分号，避免以后js语言和java语言相互切换使用的时候出现一些不必要的错误。

   c. 浏览器中内置的js解析器会逐行执行js代码，如果代码中书写有错误，则会导致js代码失效，可能全部的js代码效果无法正常执行。

   d. script标签必须要书写成一个标签对，不可以写成一个自闭标签。如果写成一个自闭标签则会导致标签中的js代码无法正常执行。

4. console.log使用：

   a. 在script标签中添加console.log()语句，在括号中间可以添加要输出的数据，这些数据会最终显示在浏览器控制台中。

5. js语法--数据类型：

   js当中分为基本数据类型和复杂数据类型

   a. 基本数据类型：共有5种

Number(数值型) String(字符串) Boolean(布尔类型) undefined Null

i. Number数值型：

1) Number不仅是一个数据类型，还是js的一个包装对象。

属性：

Number.MAX_VALUE 最大值

Number.MIN_VALUE 最小值

```
console.log(Number.POSITIVE_INFINITY);//Infinity
console.log(Number.NEGATIVE_INFINITY);//-Infinity
```

NaN属性：表示一个非数字。和任何值都不相等包括它本身。

isNaN()用来判断当前值是否为一个非数字的值。如果返回值为true，则表明当前值应该为一个非数字的值，如果返回值为false，则表明当前值不是一个非数字的值，即为一个数值类型的值。

函数：

查阅API文档

ii. String字符串类型：

String表示的数据必须使用双引号包含，这样才表示当前值为一个字符串类型。

String不仅是一个数据类型，还是js的一个包装对象。

属性：

length 当前字符串的长度

函数：

查阅API，会使用即可。

iii. Boolean布尔类型

Boolean类型只有两个值，分别是true和false。

Boolean不仅是一个数据类型，还是js 的一个包装对象。

||

&&

|

&

!

```
console.log(true || false)//true
console.log(true && false)//false
console.log(true | false);//1
console.log(true & false);//0
console.log(!true);//false
```

iv. undfined类型

Undefined是js 的一个基本类型，其中这个里只包含undefined这一个值，这个值表示一个未被定义的值。

```
var a;console.log(a);//undefined
```

v. Null类型

Null类型是js的一个基本数据类型，其中类型中只包含null一个值，这个值表示一个不存在值。常用作返回值使用。

6. js语法- 数据类型自动转换

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会转换为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

7. js语法-运算符

+	在加号中如果有字符串则执行拼接的操作，如果是纯数字则执行加法运算
-	只能执行减法运算
*	3/2*1000=1500
/	在结果中如果有浮点型数据则会以浮点型结果直接展示，不用关心数值是否为整形或浮点型。

三元表达式：2+3>5?"true":"false";

typeof()--判断当前值的类型：

在运算中如果结果可以转换成整形则会以整型的形式展示。

8. js语法-语句

for ---- 没有增强for循环

while

do..while

switch case

---和java中的使用方式相同

if语句特点：

```
var x = 5;
if(4 == x){
    console.log("正确");
}else{

    console.log("错误");
}
```

注意：

```
if(x = 4){
    console.log("正确");
}else{
    console.log("错误");
}
```

如果在if判断中只有一个等号，则，执行的操作为，先赋值，再运算的操作。操作过程为：先将x赋值为4，再判断if(x)，也就是if(4)，即if(true)，所以只会

打印为ture的部分代码。

9. js语法-定义变量
   a. 使用关键字var来定义变量。定义的变量为弱类型的变量。
      使用var定义的变量是没有数据类型的。

```
var a = 1;
a = "abc";
a = true;
a = new Object();
```

      注意：因为定义a的时候定义的类型为若类型，这个类型是没有数据类型的，所以可以为这个类型的变量指定任意数据类型的变量。
   b. 全局变量和局部变量：

```
    x = 4; ---全局变量
var x = 4; ---局部变量
```

10. 复杂数据类型
    函数 数组 对象
    a. 函数：
       i. 是一段可执行代码的合集，在需要执行的时候可以在方法名之后添加一对小括号执行方法。是一段可执行的字符串。
       ii. 在函数中有一个隐藏的属性arguments，其中保存的是用户输入的全部参数，可以通过arguments.length获取用户输入参数的长度。如果用户输入的参数数量大于函数现有的参数长度，多余的参数没有被抛弃，利用arguments依然可以获取到用户输入全部参数。如果用户的输入的参数数量小于函数现有的参数长度，则缺少的参数会使用undefined来赋值通过arguments也可以获取用户传入的参数。
       iii. 在js的函数中可以认为函数是一种特殊的变量，这个变量可以作为参数使用，可以作为方法使用。作为参数使用时，直接书写方法名即可，这时书写的方法名就是代表当前方法的变量。作为方法使用，则需要在方法名之后添加上一对小括号，这时这个函数就会执行函数中的函数体。
       iv. 函数的定义：
          1) 普通方法定义函数：

```
function mx(a,b){
    return a+b;
}
```

          执行函数：

```
mx(1,2);
```

          2) 动态函数定义：
          动态函数定义方式，参数列表中先书写全部参数，最后一个参数需要书写方法体。（动态函数最后一个参数位置是填写方法体的位置。）

```
var mx = new Function("a","b","return a + b");
mx(2,3);
```

          3) 匿名函数定义(直接量函数定义)：

```
var mx = function(a,b){
    return a+b;
}
```

mx(3,4);

b. 数组 -- Array对象

在js中，数组本质就是一个用中括号括起来，其中添加任意类型的元素，每个元素用逗号隔开的字符串。

```
var arr = new Array();
var arr = new Array(3);
var arr = new Array(1,ture,"a");
var arr = [1,"b",false,new Object()];
```

特点：

数组的长度是任意的。

数组存储的元素类型是任意的。

数组操作：

pop()

push()

shift()

for遍历数组。

c. 对象

内置对象和自定义对象。

i. 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。

ii. 自定义对象

js中对象的本质就是一个大括号，其中包含任意多个键值对，键值对直接使用逗号隔开，这种形式组成的字符串就是对象。所以对象的本质也是一个字符串。

第一种方式：无参构造函数创建对象

```
function Person() {

}
var p = new Person();
```

第二种方式：有参构造函数创建对象

```
function Person(name, age) {
    this.name = name;
    this.age = age;
}
var p = new Person("ls",18);
```

第三种方式：直接量方式定义对象

```
var p = {name:"js", age:18, addr:"bj"};
```

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"]},
    ]
}
```

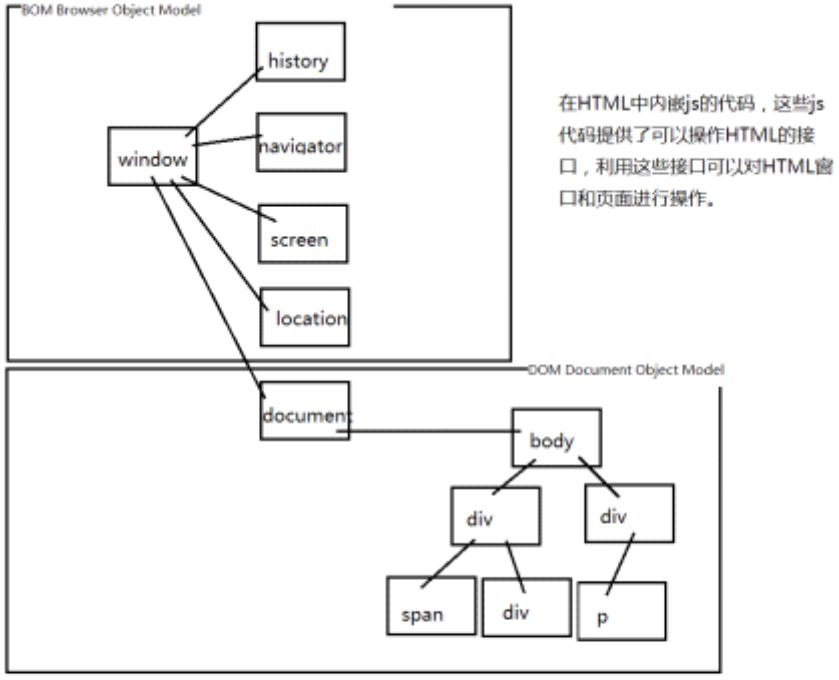
查看当前JSON中第二个wife的工作

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

DHTML

2019年5月6日 星期一 16:53

1. DHTML
   a. DynamicHTML
   b. 将整个页面转换成多个js对象来处理。
   c. 将页面中全部元素转换成js对象之后，页面就会产生了一棵js的文档树
   d. DHTML分为两部分BOM和DOM。
   e. 利用js操作HTML页面其实就是对js文档树上的节点进行操作。
   f. 可以对节点进行增删改的操作。



2. BOM Browser Object Model 浏览器对象模型
   - 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()

a.