前端 JavaScript

阅读目录

js 是一种轻量级的脚本语言，它可以部署在多种环境，最常见的部署环境就是浏览器。

所谓的脚本语言，指的是它不具备开发操作系统的能力，而是只用来编写控制其他大型应用程序的操作方法

一个完整的 JavaScript 实现是由以下 3 个不同部分组成的：

- 核心（ECMAScript）
- 文档对象模型（DOM） Document object model (整合 js，css，html)
- 浏览器对象模型（BOM） Broswer object model（整合 js 和浏览器）

JavaScript 引入的两种方式:

1.直接在 style 中 的 script 标签中写

```
<script>
  // 在这里写你的 JS 代码
</script>
```

2.导入外部 JavaScript 的文件

```
<script src="myscript.js"></script>
```

JavaScript 的语言规范:

1.注释:

```
/ 这是单行注释

/*
这是
多行注释
*/
```

2.JavaScript 中的语句要以;号结束

3.变量声明: 在 Javascrip 中使用新变量前要进行声明.

1. JavaScript 的变量名可以使用_，数字，字母，$组成，不能以数字开头。

2. 声明变量使用 var 变量名; 的格式来进行声明

```
var name = "Alex";
var age = 18;
```

还可以这么写:

```
<script>
console.log('你好');
    var name;
    name='xiaohong';
    console.log(name)
</script>
```

注意:

变量名是区分大小写的。

推荐使用驼峰式命名规则。 比如 passWord

JavaScript 中的数据类型

查看类型的方法为 typeof

```
  var name;
    name='xiaohong';
    console.log(typeof name)
```

JavaScript 拥有动态类型

1.数字类型 number

JavaScript 不区分整型和浮点型，就只有一种数字类型。number

常用方法:

parseInt 将字符串中的数字转换为整数，如果字符串不是以数字开头的，将返回 NAN，注意将小数部分取整。

```
var a1 = parseInt('123er56');
    console.log(a1)
//结果为
123
```

```
parseInt("123")  // 返回 123  把字符串的数字转换为数字
parseInt("ABC")  // 返回 NaN,NaN 属性是代表非数字值的特殊值。该属性用于指示某个值不是数字。
parseFloat("123.456")  // 返回 123.456
```

隐式转换：就是系统默认的不需要加声明就可以自动转换

```
 console.log(1+true);
    console.log(1+false);
    console.log(1+null);
    console.log(1+undefined);
//结果为
2
1
1
NaN
```

由此说明： true 等于 1. false=0 null 是一个空，不能单纯作为 0 处理 undefined 也等于 NAN

2.字符串:string

注意不区分空字符，只要用引号，引起来，就是字符串

字符串拼接的方法 推荐使用'+'

obj.length 注意 length 后面没有括号:返回字符串的长度.

obj.trim() 移除字符串左右两边的空格,但是不能移除字符串中间的空格

obj.trimLeft() 移除字符串左边的空格

obj.trimRight() 移除字符串右边的空格

obj.charAt(index) 通过索引查找对应的元素. 不可以用负值,负值会返回空字符串

obj.concat(value, ...): 字符串的拼接,不推荐使用这个用+号

obj.indexOf(字符串中的元素, startindex: 通过元素找元素的索引,如果有多个只显示最左边的那个. 如果找不到则返回-1, indexstart 可以省略,如果省略表示从头开始查找

如果不省略则从 indexstart 处开始查找,但是得到的结果还是和从开头找的结果一样,如果 startindex 是负数，则 startindex 被当作零。如果它比最大的字符位置索引还大，结果为-1 找不到。

obj.substring(from, to) :和 python 中的字符串切片一样前包后不包, 但是**不能用负数来表示**, to 可以不写

```
name;
"sticker, china"
var name11=name.substring(2,5);
undefined
name11;
"ick"
```

obj.slice(start, end): 返回一个新的字符串。包括字符串 stringObject 从 start 开始（包括 start）到 end 结束（不包括 end）为止的所有字符。可以**接受负数,to 可以不写.**

obj.toLowerCase(): 字符串变为小写.

obj.toUpperCase():字符串变为大写.

obj.split(separator,howmany):用于把一个字符串分割成一个列表(数组)。separator 字符串或正则表达式,从 separator 处切开,但是不包含 separator 自身,howmany 用于指定你需要列表中有几个符合条件的字符,当不填时会全部显示显示.

⊞我以空字符来分割的

结果:

```
(11) ["h", "e", "l", "l", "o", " ", "w", "o", "r", "l", "d"]
```

范例二:

```
var n2=n.split('',2);
```

结果

```
["h", "e"]
```

3.布尔类型 boolean

注意:区别于 Python，true 和 false 都是小写。

```
var a = true;
var b = false;
```

注意: 返回 false 的情况:

1. " " 即:空字符串
2. 0
3. -0
4. undefined
5. false
6. null
7. NaN

其他的的都返回 True ,包括空数组和对象

比较运算符

比较运算符在逻辑语句中使用，以测定变量或值是否相等。

注意: '=' 是赋值

== 是等于 仅判断值是否相等,不管类型的事

===是强等于 比较值和类型是否都相等.

给定 x=5，下面的表格解释了比较运算符：

运算符	描述	例子
==	等于	x==8 为 false
===	全等（值和类型）	x===5 为 true；x==="5" 为 false
!=	不等于	x!=8 为 true
>	大于	x>8 为 false
<	小于	x<8 为 true
>=	大于或等于	x>=8 为 false
<=	小于或等于	x<=8 为 true

逻辑运算符

逻辑运算符用于测定变量或值之间的逻辑。

给定 x=6 以及 y=3，下表解释了逻辑运算符：

运算符	描述	例子
&&	and	(x < 10 && y > 1) 为 true
||	or	(x==5 || y==5) 为 false
!	not	!(x==y) 为 true

++递增

有两种 a++ 和++a

当 a++或++a,单独成行的时候，a 在原来的基础上相当于加了一，

```
var a = 10;
    a++;
    console.log(a)
//结果等于
11
```

a++ 先运算再加 1 区别是 a++必须遇到+号才加一（单独一行不算）

++a 先加 1 在运算 ++a,不需要遇到+号就能加一

```
var a = 10;
var b = 10;

var result = a++ + a ; //结果等于 21，思路： a++本身等于 10，当它经过+后变为 11，然后加上 a 就是 21
var response = b++ + ++b; //结果等于 22，思路分析，b++本身等于 10 经过+后，变为 11，而++b 不需要经过+号就能变为 11
```

三元表达式

JavaScript 还包含了基于某些条件对变量进行赋值的条件运算符。

语法

var name=(condition)?value1:value2

当 condition 为 true 时，执行 value1 的结果，否则执行 value2 的结果。

例子:

```
var a = 22;
var age = a > 5 ? 'a 大于 5':'a 小于 5';
alert(age)
```

高阶的多个数的比较

```
 var a = 22;
    var b = 12;
    var c = 52;
    var maxNum = a > b ? a > c ? a : c : b > c ? b : c;
    alert(maxNum)
//书写顺序
```

```
//第一 a>b?a:b    先比较 ab
//第二 a>b?a>c?a:c :b //然后 c 再和 a 比
//第三步 a>b?a>c?a:c:b>c?b:c //然后 c 再和 b 比
//更好的理解的写法，他俩都和第三个数比
a > b ? (a > c ? a : c ): (b > c ? b : c)
```

4.数组

类似于 Python 中的列表。

obj.length 返回数组中的元素个数

obj.push(ele):数组尾部添加一个或多个元素

obj.unshift(ele):向数组的开头添加一个或更多元素，并返回新的长度。

```
var a=[1,2,5,6,8];
  a.unshift('1');
  console.log(a)
```

结果:

```
["1", 1, 2, 5, 6, 8]
```

obj.pop(): 删除数组的最后一个元素。

obj.shift()删除数组的第一个元素。

obj.slice()返回一个新的数组，包含从 start 到 end （不包括该元素）的 arrayObject 中的元素。

obj.reverse()反转

obj.join(seq):将数组元素连接成字符串

obj.concat(val, ...):连接数组

遍历数组中的元素:

```
var a=[1,2,5,6,8];
  for (var i=0;i<a.length;i++){
      console.log(a[i]);
  }
```

5.对象

对象是大括号定义的**无序**的数据集合，有键值构成，键值之间用冒号隔开，大括号末尾要使用分号结束

相当于 python 中的字典

注意:

1. 属性可以不加引号，但是一般都加

2. 最后一个键值对后边不加逗号

对象取属性值的方法有两种：第一种点运算符，第二种，中括号

```
var a = {"name": "Alex", "age": 18};
console.log(a.name); //第一种
console.log(a["age"]);//第二种
```

点运算符和中括号运算符的区别?

1. 点运算符不可以点一个数字，[] 可以。
2. 点运算符不可以通过字符串变量，访问一个对象的属性。
3. 点运算符可以将 js 中的关键字，作为对象的属性名添加，而中括号不可以。

```
var obj = {
        'hero':'钢铁侠'
    };
    var str = 'hero';
    console.log(obj.str);
    console.log(obj[str])
    //第一个结果为 undefined
    // 第二个结果为钢铁侠
```

遍历对象中的内容：

```
var a = {"name": "Alex", "age": 18};
for (var i in a){

  console.log(i, a[i]);
}
```

6.null 和 undefined

- undefined 表示的是当声明的变量未初始化时，该变量的默认值是 undefined。还有就是函数无明确的返回值时，返回的也是 undefined。
- undefined 是全局对象的一个属性。也就是说，它是全局作用域的一个变量。undefined 的最初值就是原始数据类型 undefined。一个没有被赋值的变量的类型是 undefined。如果方法或者是语句中操作的变量没有被赋值，则会返回 undefined,一个函数如果没有返回值，就会返回一个 undefined 值。
- null 表示值不存在

undefined 表示没有声明和赋值变量，就使用该变量。

null 声明了变量并且变量值是 null。相当于 python 中的 None

```
var a;
```

```
    a = null;
    console.log(typeof a)
//输入的结果为 object
```

出现 underfine 的情况有两种:

1、变量没赋初始值

2. 变量没有声明。

3、函数没有返回值

第一种情况：

```
<script>
   var a;
   alert(a);//这里没有初始化值，会出现 undefined
</script>
```

第二种情况

```
console.log(typeof a1) //没有声明和赋值
```

第三种情况：

```
<script>
   var a='1';
   function k(){
   }
   a=k();//这里函数没有返回值，会出现 undefined
</script>
```

类型查询

```
typeof "abc"  // "string"
typeof null  // "object"
typeof true  // "boolean"
typeof 123 // "number"
```

流程控制

if

注意当 if 后边的括号内是负性内容(表示否定的词）的时候，不可以进入分支内容负性内容包括：null,""(空的字符串），undefined，false,0,NAN

```
if (undefined){
        console.log('nihoa')
```

```
    }
```

if-else

```
 var a = parseInt(prompt('请输入您的年龄'));
    if (a > 18) {
        alert('你可以去网吧了')
    } else {
        alert('你不可以去网吧，赶紧回家写作业')
    }
```

if-else if-else

```
var a = 10;
if (a > 5){
  console.log("a > 5");
}else if (a < 5) {
  console.log("a < 5");
}else {
  console.log("a = 5");
}
```

switch

语句用于基于不同条件执行不同动作，效率比 if else 要高点额很多

```
var x;
    var d = new Date().getDay();
    switch (d) {
        case 0 :
            x = '今天休息';
            break;
        case 1 :
            x = '今天休息';
            break;
            if (d > 1) {
                x = '今天要上班';
                break;
```

```
            }
    }
    alert(x)
```

循环

任何循环语句：都有 4 个条件

1. 初始化变量。例如 var i =0
2. 循环条件，例如 i<10
3. 循环体，例如 console.log(i)
4. 迭代条件。i++

for

```
for (var i=0;i<10;i++) {
  console.log(i);
}
```

for in

```
var person = [1, 3, 4, 5, 6];
    for (var k in person) {
        console.log(k)
    }
```

while

```
var i = 0;
while (i < 10) {
  console.log(i);
  i++;
}
```

break 和 continue

它们的用法和 python 的一样

内置对象和方法

JavaScript 中的所有事物都是对象：字符串、数字、数组、日期，等等。在 JavaScript 中，对象是拥有属性和方法的数据。

对象只是带有属性和方法的特殊数据类型。

创建对象:

通过 JavaScript，您能够定义并创建自己的对象。

```
var person=new Object();  // 创建一个 person 对象
person.name="Alex";  // person 对象的 name 属性
person.age=18;  // person 对象的 age 属性
```

Date 对象

创建 Date 对象

```
//方法 1：不指定参数
var d1 = new Date();
console.log(d1.toLocaleString());
//方法 2：参数为日期字符串
var d2 = new Date("2004/3/20 11:12");
console.log(d2.toLocaleString());
var d3 = new Date("04/03/20 11:12");
console.log(d3.toLocaleString());
//方法 3：参数为毫秒数
var d3 = new Date(5000);
console.log(d3.toLocaleString());
console.log(d3.toUTCString());
//方法 4：参数为年月日小时分钟秒毫秒
var d4 = new Date(2004,2,20,11,12,0,300);
console.log(d4.toLocaleString());  //毫秒并不直接显示
```

Date 对象的方法：

```
var d = new Date();
//getDate()                获取日
//getDay ()                获取星期几 从星期天开始
//getMonth ()              获取月（0-11）
//getFullYear ()           获取完整年份
//getYear ()               获取年
//getHours ()              获取小时
//getMinutes ()            获取分钟
//getSeconds ()            获取秒
//getMilliseconds ()       获取毫秒
//getTime ()               返回累计毫秒数(从 1970/1/1 午夜)
```

json 对象

var str1 = '{"name": "Alex", "age": 18}'; var obj1 = {"name": "Alex", "age": 18};

// 对象转换成 JSON 字符串 var str = JSON.stringify(obj1); 注意 Json 需要大写

// JSON 字符串转换成对象 var obj = JSON.parse(str1);

regexp 对象

RegExp 是正则表达式的缩写。

```
// 创建 RegExp 对象方式
var reg1 = new RegExp("pattern", [flags]);
pattern:正则表达式的文本
[flags]: 匹配模式,这个参数可选 该标志有以下值的任意组合
g :全局匹配;找到所有匹配，而不是在第一个匹配后停止
i :忽略大小写
m :多行; 将开始和结束字符（^和$）视为在多行上工作（也就是，分别匹配每一行的开始和结束（由 \n 或 \r 分割），而不只是只匹配整个输入字符串的最开始和最末尾处。
u: Unicode; 将模式视为 Unicode 序列点的序列.
y 粘性匹配; 仅匹配目标字符串中此正则表达式的 lastIndex 属性指示的索引(并且不尝试从任何后续的索引匹配)。
```

RegExp 对象的方法

RegExp 对象有 3 个方法:

test():检索字符串中的指定值。返回值是 true 或 false。

```
1 var part2=new RegExp('e');
2 undefined
3 part2.test("The best things in life are free");
4 结果 true
```

exec() :检索字符串中的指定值。返回值是被找到的值。如果没有发现匹配，则返回 null,检索字符串中指定的值。返回找到的值，并确定其位置。

```
var part2=new RegExp('e');
part2.exec("The best things in life are free");
结果;["e", index: 2, input: "The best things in life are free"]
```

RegExpObject.compile(regexp,modifier):用于在脚本执行过程中编译正则表达式,也可用于改变和重新编译正则表达式。regexp 正则表达式。modifier 规定匹配的类型。"g" 用于全局匹配，"i" 用于区分大小写，"gi" 用于全局区分大小写的匹配。

String 对象与正则结合的 4 个方法

```
ar s2 = "hello world";
s2.match(/o/g);         // ["o", "o"]             查找字符串中 符合正则 的内容
s2.search(/h/g);        // 0                      查找字符串中符合正则表达式的内容位置
s2.split(/o/g);         // ["hell", " w", "rld"]  按照正则表达式对字符串进行切割
s2.replace(/o/g, "s");  // "hells wsrld"          对字符串按照正则进行替换
```

其中

s2.match(/o/g)下边式子的缩写,这样就不用新定义的对象了

```
var s2="hello, world";
var n = new RegExp("o","g");
console.log(s2.match(n ));
//RegExp 对象
//创建正则对象方式 1
// 参数 1 正则表达式
// 参数 2 验证模式：g(global)和 i(忽略大小写)
// 用户名只能是英文字母、数字和_，并且首字母必须是英文字母。长度最短不能少于 6 位 最长不能超过 12 位。
// 创建 RegExp 对象方式
var reg1 = new RegExp("^[a-zA-Z][a-zA-Z0-9_]{5,11}$", "g");
// 匹配响应的字符串
var s1 = "bc123";
//RegExp 对象的 test 方法，测试一个字符串是否符合对应的正则规则，返回值是 true 或 false。
reg1.test(s1);  // true
// 创建方式 2
// /填写正则表达式/匹配模式
var reg2 = /^[a-zA-Z][a-zA-Z0-9_]{5,11}$/g;
reg2.test(s1);  // true
// String 对象与正则结合的 4 个方法
var s2 = "hello world";
s2.match(/o/g);         // ["o", "o"]             查找字符串中 符合正则 的内容
s2.search(/h/g);        // 0                      查找字符串中符合正则表达式的内容位置
s2.split(/o/g);         // ["hell", " w", "rld"]  按照正则表达式对字符串进行切割
s2.replace(/o/g, "s");  // "hells wsrld"          对字符串按照正则进行替换
```

```
// 关于匹配模式：g 和 i 的简单示例
var s1 = "name:Alex age:18";
s1.replace(/a/, "哈哈哈");      // "n 哈哈哈 me:Alex age:18"
s1.replace(/a/g, "哈哈哈");     // "n 哈哈哈 me:Alex 哈哈哈 ge:18"      全局匹配
s1.replace(/a/gi, "哈哈哈");    // "n 哈哈哈 me:哈哈哈 lex 哈哈哈 ge:18"  不区分大小写
```

RegExp 对象

实例

在字符串中全局搜索 "man"，并用 "person" 替换。然后通过 compile() 方法，改变正则表达式，用 "person" 替换 "man"

```
var str='Every man in the world! Every woman on earth!';
patt=/man/g;
str2=str.replace(patt,"person");
console.log(str2);
```

结果:

```
"Every person in the world! Every woperson on earth!"
```

match 对象

abs(x)	返回数的绝对值。
exp(x)	返回 e 的指数。
floor(x)	对数进行下舍入。
log(x)	返回数的自然对数（底为 e）。
max(x,y)	返回 x 和 y 中的最高值。
min(x,y)	返回 x 和 y 中的最低值。
pow(x,y)	返回 x 的 y 次幂。
random()	返回 0 ~ 1 之间的随机数。
round(x)	把数四舍五入为最接近的整数。
sin(x)	返回数的正弦。
sqrt(x)	返回数的平方根。
tan(x)	返回角的正切。