



第七课：数据类型转换

■ 主讲老师：万章



目录

数据类型判断

数字转换

字符转换

布尔转换



数据类型判断

数据类型判断

```
var a=1;
var b="你好";
var c=[0,1,2];
var d={x:"hello",y:"world"};
var e=null;
var f=false;
var g=function(){
    console.log("万章大帅比");
};
var h;

console.log(typeof a);
console.log(typeof b);
console.log(typeof c);
console.log(typeof d);
console.log(typeof e);
console.log(typeof f);
console.log(typeof g);
console.log(typeof h);
```

number	数据类型.html:22
string	数据类型.html:23
object	数据类型.html:24
object	数据类型.html:25
object	数据类型.html:26
boolean	数据类型.html:27
function	数据类型.html:28
undefined	数据类型.html:29
>	

typeof 操作符 可以判断一个值的数据类型

我们假设有一个变量a;
那么通过代码 `typeof a`;可以根据代码返回的值来判断变量a的数据类型

下方是返回的不同的值所代表的含义:

- "undefined"——如果这个值未定义;
- "boolean"——如果这个值是布尔值;
- "string"——如果这个值是字符串;
- "number"——如果这个值是数值;
- "object"——如果这个值是对象或null;
- "function"——如果这个值是函数;



数字类型转换

数字类型转换之Number()

Number(value)

通过这个函数，我们可以把其他的数据类型转换成数字类型

如果是Boolean 值，true 和false 将分别被转换为1 和0。

如果是数字值，只是简单的传入和返回。

如果是null 值，返回0。

如果是undefined，返回NaN。

```
> Number(true)
< 1
> Number(false)
< 0
> Number(3)
< 3
> Number(null)
< 0
> Number(undefined)
< NaN
>
```

数字类型转换之Number()

□ 如果是字符串，遵循下列规则：

如果字符串中只包含数字（包括前面带正号或负号的情况），则将其转换为十进制数值，即"1"会变成1，"123"会变成123，而"011"会变成11（注意：前导的零被忽略了）；

如果字符串中包含有效的浮点格式，如"1.1"，则将其转换为对应的浮点数值（同样，也会忽略前导零）；

如果字符串中包含有效的十六进制格式，例如"0xf"，则将其转换为相同大小的十进制整数值；

如果字符串是空的（不包含任何字符），则将其转换为0；

如果字符串中包含除上述格式之外的字符，则将其转换为NaN。

```
> Number("+123")
< 123
> Number("-123")
< -123
> Number("00123")
< 123
> Number("1.3")
< 1.3
> Number("0xb")
< 11
> Number("")
< 0
> Number("123h")
< NaN
> Number("hello123")
< NaN
> Number("[212]")
< NaN
> Number("!212")
< NaN
```

数字类型转换之parseInt()

Number()函数在转换字符串时比较复杂而且不够合理，因此在处理整数的时候更常用的是parseInt()函数。
parseInt()是Number()的一个分支, int是整数的意思

```
> parseInt(" 123")
< 123
> Number(" 212")
< 212
> parseInt("123hello")
< 123
> parseInt("12445 hello")
< 12445
>
```

```
top
> parseInt("h123")
< NaN
> parseInt("+123")
< 123
> parseInt("-123")
< -123
> parseInt("!123")
< NaN
>
```

```
Elements Console Sources
top
> parseInt("12.3")
< 12
> parseInt("12.9")
< 12
> parseInt("0.9")
< 0
> parseInt("1.9hello")
< 1
```

它会忽略字符串前面的空格，直至找到第一个非空格字符如果第一个字符是数字字符，parseInt()会继续解析第二个字符，直到解析完所有后续字符或者遇到了一个非数字字符

如果第一个字符不是数字字符或者负号，parseInt()就会返回NaN

如果待解析的值是小数或是字符串打头的是小数的话，那么parseInt()的结果只会截取整数部分，无论小数部分是多少

数字类型转换之parseFloat()

parseFloat与parseInt唯一的区别就是它可以分析出小数

```
> parseFloat("1.9hello")  
1.9  
> parseFloat("+1.9hello")  
1.9  
> parseFloat(" 1.9hello")  
1.9  
> parseFloat("23.32.32")  
23.32
```



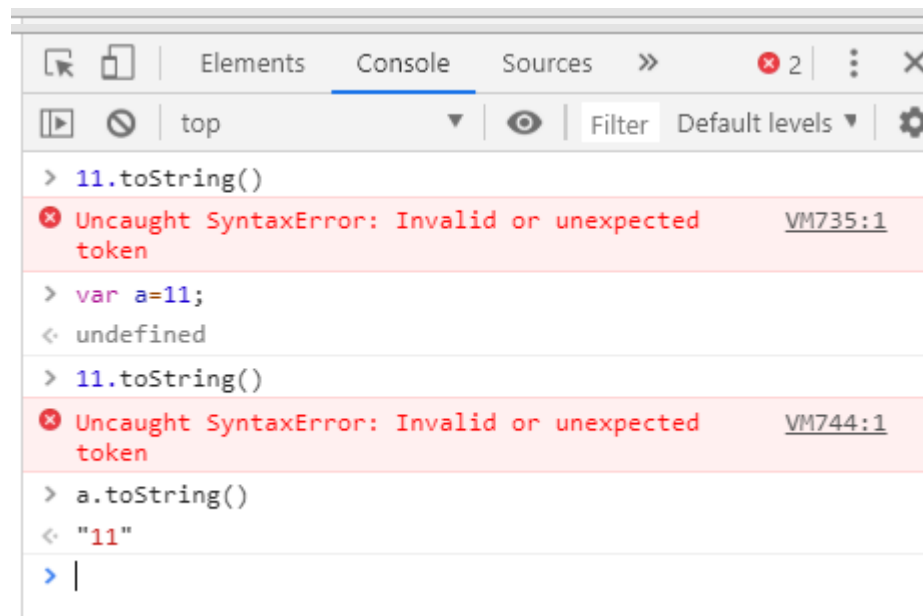
字符类型转换

字符类型转换toString()

万物皆对象!!!!!!

几乎每个值都有的toString()方法,但是调用的方法不同

数字转化为字符串不能直接用
数字.toString()来实现转化
需要把数字存在一个变量中间, 然后用
变量.toString()实现转化



```
> 11.toString()
Uncaught SyntaxError: Invalid or unexpected token VM735:1

> var a=11;
undefined

> 11.toString()
Uncaught SyntaxError: Invalid or unexpected token VM744:1

> a.toString()
"11"

> |
```

字符类型转换toString()

数组转为字符串的方法只需一个个拆分出来研究就好

[值1,值2,值3].toString();

1:值1->字符串1

2:值2->字符串2

3:值3->字符串3

然后用 , (英文逗号)号把几个字符串连接起来:

字符串1,字符串2,字符串3

如果是嵌套数组, 即数组里面有小数组, 那么先把小数组变成字符串, 然后再来解析大数组

[值0, [值1,值2,值3]].toString();

1:值1->字符串1

2:值2->字符串2

3:值3->字符串3

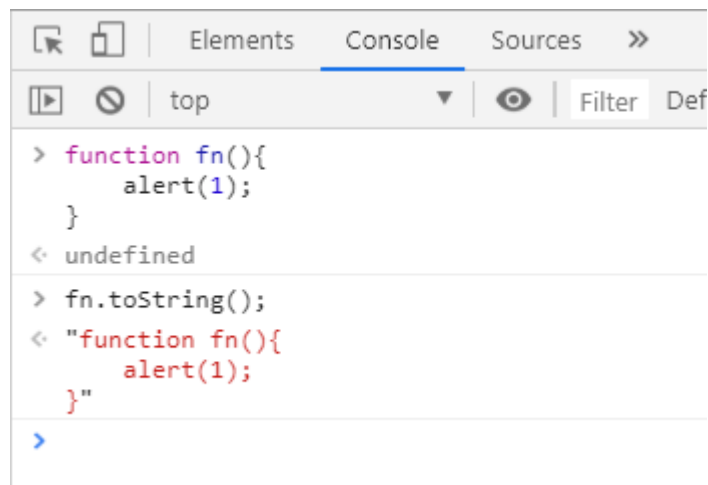
4:[值1,值2,值3] -> 字符串1,字符串2,字符串3

5:[值0,"字符串1,字符串2,字符串3"]

6:值0 -> 字符串0

```
> var a=["hello","world","1"];
< undefined
> a.toString();
< "hello,world,1"
> var b=["hello","world",["hello","world"]];
< undefined
> b.toString();
< "hello,world,hello,world"
> |
```

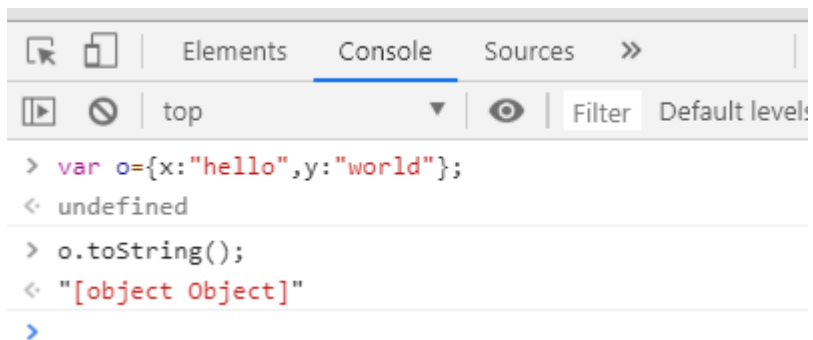
字符类型转换toString()



```
> function fn(){  
  alert(1);  
}  
← undefined  
  
> fn.toString();  
← "function fn(){  
  alert(1);  
}"  
  
>
```

函数转为字符串的结果默认情况下就是直接返回函数的完整代码

字符类型转换toString()



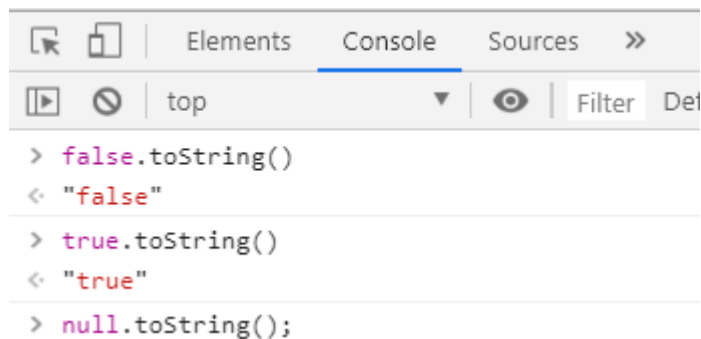
```
> var o={x:"hello",y:"world"};
< undefined

> o.toString();
< "[object Object]"

>
```

对象转为字符串的结果默认情况下就是直接一个标识符号
[object,object]

字符类型转换toString()



```
> false.toString()
< "false"

> true.toString()
< "true"

> null.toString();
```

布尔值转为字符串的结果就是返回相应的字符串

String()函数遵循下列转换规则：

如果值有toString()方法，则调用该方法并返回相应的结果；

如果值是null，则返回"null"；

如果值是undefined，则返回"undefined"。

字符类型转换toString()

```
Elements Console Sources >> 2
top Filter Default levels
> var a=null;
< undefined
> a.toString();
✖ ▶ Uncaught TypeError: Cannot read property 'toString' of null VM558:1
    at <anonymous>:1:3
> String(a)
< "null"
> var b=undefined;
< undefined
> String(b)
< "undefined"
> b.toString();
✖ ▶ Uncaught TypeError: Cannot read property 'toString' of undefined VM634:1
    at <anonymous>:1:3
>
```

String()函数遵循下列转换规则：

如果值有toString()方法，则调用该方法并返回相应的结果；

如果值是null，则返回"null"；

如果值是undefined，则返回"undefined"。

```
Elements Console Sources >>
top Filter Default levels
> String({x:"hello",y:"world"})
< "[object Object]"
> String([1,2,3])
< "1,2,3"
>
```


字符类型拓展之转义字符

字 面 量	含 义
<code>\n</code>	换行
<code>\t</code>	制表
<code>\b</code>	空格
<code>\r</code>	回车
<code>\f</code>	进纸
<code>\\</code>	斜杠
<code>\'</code>	单引号 (')，在用单引号表示的字符串中使用。例如: <code>'He said, \'hey.\\''</code>
<code>\"</code>	双引号 (")，在用双引号表示的字符串中使用。例如: <code>"He said, \"hey.\\\""</code>
<code>\xnn</code>	以十六进制代码 <code>nn</code> 表示的一个字符（其中 <code>n</code> 为0 ~ F）。例如， <code>\x41</code> 表示"A"
<code>\unnnn</code>	以十六进制代码 <code>nnnn</code> 表示的一个Unicode字符（其中 <code>n</code> 为0 ~ F）。例如， <code>\u03a3</code> 表示希腊字符Σ

```
> "hello\nworld"
< "hello
world"

> "hello\tworld"
< "hello  world"

> "hello\bworld"
< "hello□world"

> "hello\rworld"
< "helloworld"

> "hello\"w\"orld"
< "hello"w"orld"

>
```



布尔类型转换

布尔类型转换toString()

数据类型	转换为true的值	转换为false的值
Boolean	true	false
String	任何非空字符串	""（空字符串）
Number	任何非零数字值（包括无穷大）	0和NaN（参见本章后面有关NaN的内容）
Object	任何对象	null
Undefined	/	undefined