紫云飞

订阅 随笔-180 评论-460 访问量-undefined

JavaScript:Object.prototype.toString方法的原理

在JavaScript中,想要判断某个对象值属于哪种内置类型,最靠谱的做法就是通过Object.prototype.toString方法.

```
var arr = [];
console.log(Object.prototype.toString.call(arr))  //"[object Array]"
```

本文要讲的就是,toString方法是如何做到这一点的,原理是什么.

ECMAScript 3

在ES3中,Object.prototype.toString方法的规范如下:

> 15.2.4.2 Object.prototype.toString()
>
> 在**toString**方法被调用时,会执行下面的操作步骤:
>
> 1. 获取this对象的[[Class]]属性的值.
>
> 2. 计算出三个字符串"**[object]**",第一步的操作结果Result(1),以及"**]**"连接后的新字符串.
>
> 3. 返回第二步的操作结果Result(2).

[[Class]]是一个内部属性,所有的对象(原生对象和宿主对象)都拥有该属性.在规范中,[[Class]]是这么定义的

内部属性	描述
[[Class]]	一个字符串值,表明了该对象的类型.

然后给了一段解释:

> 所有内置对象的[[Class]]属性的值是由本规范定义的.所有宿主对象的[[Class]]属性的值可以是任意值,甚至可以是内置对象使用过的[[Class]]属性的值.[[Class]]属性的值可以用来判断一个原生对象属于哪种内置类型.需要注意的是,除了通过**`Object.prototype.toString`**方法之外,本规范没有提供任何其他方式来让程序访问该属性的值(查看 15.2.4.2).

也就是说,把Object.prototype.toString方法返回的字符串,去掉前面固定的**`"[object "`**和后面固定的**`"]"`**,就是内部属性[[class]]的值,也就达到了判断对象类型的目的.jQuery中的工具方法$.type(),就是干这个的.

在ES3中,规范文档并没有总结出[[class]]内部属性一共有几种,不过我们可以自己统计一下,原生对象的[[class]]内部属性的值一共有10种.分别是:`"Array", "Boolean", "Date", "Error", "Function", "Math", "Number", "Object", "RegExp", "String".`

ECMAScript 5

在ES5.1中,除了规范写的更详细一些以外,Object.prototype.toString方法和[[class]]内部属性的定义上也有一些变化,Object.prototype.toString方法的规范如下:

> **15.2.4.2 Object.prototype.toString ( )**
>
> 在**`toString`**方法被调用时,会执行下面的操作步骤:
>
> 1. 如果**this**的值为**undefined**,则返回`"[object Undefined]"`.
> 2. 如果**this**的值为**null**,则返回`"[object Null]"`.
> 3. 让*O*成为调用ToObject(**this)**的结果.
> 4. 让*class*成为*O*的内部属性[[Class]]的值.
> 5. 返回三个字符串**`"[object "`**, *class*, 以及 **`"]"`**连接后的新字符串.

可以看出,比ES3多了1,2,3步.第1,2步属于新规则,比较特殊,因为`"Undefined"`和`"Null"`并不属于[[class]]属性的值,需要注意的是,这里和严格模式无关(大部分函数在严格模式下,this的值才会保持undefined或null,非严格模式下会自动成为全局对象).第3步并不算是新规则,因为在ES3的引擎中,也都会在这一步将三种原始值类型转换成对应的包装对象,只是规范中没写出来.ES5中,[[Class]]属性的解释更加详细:

> 所有内置对象的[[Class]]属性的值是由本规范定义的.所有宿主对象的[[Class]]属性的值可以是除了"Arguments", "Array", "Boolean", "Date", "Error", "Function", "JSON", "Math", "Number", "Object", "RegExp", "String"之外的的任何字符串.[[Class]]内部属性是引擎内部用来判断一个对象属于哪种类型的值的.需要注意的是,除了通过**`Object.prototype.toString`**方法之外,本规范没有提供任何其他方式来让程序访问该属性的值(查看 15.2.4.2).

和ES3对比一下,第一个差别就是[[class]]内部属性的值多了两种,成了12种,一种是arguments对象的[[class]]成了"Arguments",而不是以前的"Object",还有就是多个了全局对象JSON,它的[[class]]值为"JSON".第二个差别就是,宿主对象的[[class]]内部属性的值,不能和这12种值冲突,不过在支持ES3的浏览器中,貌似也没有发现哪些宿主对象故意使用那10个值.

ECMAScript 6

ES6目前还只是工作草案,但能够肯定的是,**[[class]]内部属性没有了**,取而代之的是另外一个内部属性[[NativeBrand]].[[NativeBrand]]属性是这么定义的:

内部属性	属性值	描述
[[NativeBrand]]	枚举NativeBrand的一个成员.	该属性的值对应一个标志值(tag value),可以用来区分原生对象的类型.

[[NativeBrand]]属性的解释:

[[NativeBrand]]内部属性用来识别某个原生对象是否为符合本规范的某一种特定类型的对象.[[NativeBrand]]内部属性的值为下面这些枚举类型的值中的一个:NativeFunction, NativeArray, StringWrapper, BooleanWrapper, NumberWrapper, NativeMath, NativeDate, NativeRegExp, NativeError, NativeJSON, NativeArguments, NativePrivateName.[[NativeBrand]]内部属性仅用来区分区分特定类型的ECMAScript原生对象.只有在表10中明确指出的对象类型才有[[NativeBrand]]内部属性.

表10 — [[NativeBrand]]内部属性的值

属性值	对应类型
NativeFunction	Function objects
NativeArray	Array objects
StringWrapper	String objects
BooleanWrapper	Boolean objects
NumberWrapper	Number objects

NativeMath	The Math object
NativeDate	Date objects
NativeRegExp	RegExp objects
NativeError	Error objects
NativeJSON	The JSON object
NativeArguments	Arguments objects
NativePrivateName	Private Name objects

可见,和[[class]]不同的是,并不是每个对象都拥有[[NativeBrand]].同时,Object.prototype.toString方法的规范也改成了下面这样:

15.2.4.2 Object.prototype.toString ( )

在**`toString`**方法被调用时,会执行下面的操作步骤:

1. 如果**this**的值为**undefined**,则返回`"[object Undefined]"`.
2. 如果**this**的值为**null**,则返回`"[object Null]"`.
3. 让*O*成为调用ToObject(**this)**的结果.
4. 如果*O*有[[NativeBrand]]内部属性,让*tag*成为表29中对应的值.
5. 否则
    1. 让*hasTag*成为调用*O*的[[HasProperty]]内部方法后的结果,参数为@@toStringTag.
    2. 如果*hasTag*为**false**,则让*tag*为`"Object"`.
    3. 否则,
        1. 让*tag*成为调用*O*的[[Get]]内部方法后的结果,参数为@@toStringTag.
        2. 如果*tag*是一个abrupt completion,则让*tag*成为NormalCompletion(`"???"`).
        3. 让*tag*成为*tag*.[[value]].
        4. 如果Type(*tag*)不是字符串,则让*tag成为*`"???"`.

5. 如果*tag*的值为"Arguments", "Array", "Boolean", "Date", "Error", "Function", "JSON", "Math", "Number", "Object", "RegExp",或者"String"中的任一个,则让*tag*成为字符串"~"和*tag*当前的值连接后的结果.
6. 返回三个字符串"[object ", tag, and "]"连接后的新字符串.

表29 — [[NativeBrand]] 标志值

[[NativeBrand]]值	标志值
NativeFunction	"Function"
NativeArray	"Array"
StringWrapper	"String"
BooleanWrapper	"Boolean"
NumberWrapper	"Number"
NativeMath	"Math"
NativeDate	"Date"
NativeRegExp	"RegExp"
NativeError	"Error"
NativeJSON	"JSON"
NativeArguments	"Arguments"

可以看到,在规范上有了很大的变化,不过对于普通用户来说,貌似感觉不到.

也许你发现了,ES6里的新类型Map,Set等,都没有在表29中.它们在执行toString方法的时候返回的是什么?

```
console.log(Object.prototype.toString.call(Map()))   //"[object Map]"

console.log(Object.prototype.toString.call(Set()))   //"[object Set]"
```

其中的字符串"Map"是怎么来的呢:

> **15.14.5.13** Map.prototype.@@toStringTag
>
> @@toStringTag 属性的初始值为字符串**"Map"**.

由于ES6的规范还在制定中,各种相关规定都有可能改变,所以如果想了解更多细节.看看下面这两个链接,现在只需要知道的是:[[class]]没了,使用了更复杂的机制.

http://stackoverflow.com/questions/13151643/access-nativebrand-class-in-es6-ecmascript-6

https://mail.mozilla.org/pipermail/es-discuss/2012-June/023676.html

posted @ 2012-11-05 10:55 阅读(22746) 评论(6)

评论列表

#1楼 2012-11-05 12:23 e.e.p

ECMA6 那要什么时候才用在浏览器中

#2楼[楼主] 2012-11-05 13:27 紫云飞

@ Coolicer

TC39的计划是,大概2013年1月完成所有的特性制定工作(feature-complete).2013年年底发布正式标准.

不过,虽然Firefox和chrome现在就已经开始了ES6的实现工作,可是ES6非常复杂和庞大.实现起来肯定会花不少时间,完全实现就不知道哪年了,至于IE,IE11估计能沾点边吧.

#3楼 2012-11-05 13:49 Lukexywang

现有了ie后有了javascript标准,并且ie占的市场份额相当大,指定标准的时候又不按照ie的现实标准指定,这是坑微软呀

#4楼 2014-06-17 14:26 luobotang

ES6会是一个步子迈得太大以至于扯到蛋的版本

#5楼 2015-07-28 10:05 氷氷深蓝

所有宿主对象的[[Class]]属性的值可以是除了"Arguments".....之外的的任何字符串，

那我自定义一个function ClassA(){}，然后new出来，执行Object.prototype.toString.call，结果还是 Object，为什么不是 ClassA?

#6楼 2016-09-23 17:24 颗粒君

对我来讲后面两大段比较高深啦，膜拜楼主。