- 6种简单数据类型（原始类型 | 非引用类型）
  - Undefined
    - 一个变量被声明之后未初始化。
      这个变量的值是undefined
    - typeof一个从未声明的变量得到的值是undefined，typeof一个声明过但未初始化的变量得到的值也是undefined。
  - Null
    - null == undefined 为true
  - Boolean
    - 将其他值转换成布尔值：
      1、使用!!
      2、使用Boolean(其他值)
    - ""、0、NaN、null、undefined都可以转成false
  - Number
    - IEEE 754格式
    - 输入进制数：
      八进制：0开头
      十六进制：0x开头
      输出进制数：
      xx.toString(进制)

```
a
11.7
a.toString(8)
'13.5463146314631463'
a.toString(10)
'11.7'
a.toString(16)
'b.b33333333333'
```

    - 科学计数法：
      是一个数值（整数或浮点数）后跟一个大 写或小写的字母 e，再加上一个要乘的 10的多少次：3.125e7
    - 浮点值的精确度高可达 17位小数，在算术计算中远不如整数精确
    - 0.1 + 0.2不等于 0.3；
      但是0.15 + 0.15 等于0.3；
      永远不要测试某个特定的浮点
      之所以存在这种舍入错误，是因为使用了IEEE 754数值
    - 最小数值： Number.MIN_VALUE，在浏览器中值为：5e-324 ，也是Infinity
    - 最大数值：Number.MAX_VALUE ，在浏览器中值为： 1.797 693 134 862 315 7e+308；也是-Infinity；
      如果某个计算得到的 数值结果超出了 JavaScript 可以表示的范围，那么这个数值会被自动转换为一个特殊的 Infinity；
    - 确定一个值是不是有限大： isFinite()
    - NaN：
      0/0或者-0/+0得NaN；
      5/0和5/-0得Infinity和-Infinity；
      isNaN()可判断是否NaN。
    - 数值转换：
      Number():
      true => 1
      false => 0
      null =>0
      undefined => NaN
      "2" => 2
      "2.5" => 2.5
      "078" => 78
      "0x88" => 0x88
      "" => 0
      "xxxx" => NaN
      Object=> Date.valueOf() || Number().toString()
      parseInt():
      parseInt()与Number()不一样的地方： parseInt("")为NaN
      但是Number("")为0；
      第二个参数为进制：
      parseInt("AF",16)
  - 16位 Unicode字符序列
    - 可以使用双引号（"）、 单引号（'）或反引号（`）标示
    - 转换为字符串:toString（null 和 undefined 值没有 toString()方法），String(""+值

- 数据类型（7种）
  - String
    - String，“+值
    - 模板字面量标签函数：
      A`this is a ${"girl"},not a ${"boy"}!`
      function A(arrays,a,b){
      // array的值是由${}插值表达式分割出来的块组合
      // a是第一个${}的值，b是第二个${}的值
      }
    - String.raw可以获取原始字符串，而不是转义之后的字符串
  - Symbol
    - 独自Symbol出来的两个值永远不相等。
      Symbol("a") === Symbol("a") => false
    - 一般用于防止覆盖的对象属性。
      a = {};
      param = Symbol("param")
      a[param] = "xx";
    - 不能与new 关键字一起作为构造函数使用
    - Symbol.for("foo");根据传入字符串，判断是否已经存在，决定是创建还是重用。
    - Symbol.for("foo") 不等于 Symbol("foo")
    - Symbol.keyFor(Symbol实例)，传入Symbol实例（通过Symbol.for()创建），查找该实例的字符串键
    - 凡是可以使用字符串或者数值作为属性的地方，都可以使用符号作为属性。
      s1 = Symbol("foo");
      a = {s1:"xxx"}
      b = [];b[S1] = 3;
    - Object.getOwnPropertyNames()返回对象实例的常规属性数组
    - Object.getOwnPropertySymbols())返回对象实例的符号属性数组
    - Object.getOwnProperty- Descriptors()会返回同时包含常规和符号属性描述符的对象。。Reflect.ownKeys()会返回两种类型 的键
    - 内置符号　　内置符号属性都是不可写、不可枚举、不可配置
    - Symbol.asyncIterator：
      一般与for await of一起用
    - Symbol.hasInstance：
      每个构造函数自带静态方法
      static [Symbol.hasInstance]（obj）{}
      使用：
      Foo[Symbol.hasInstance](obj) 等于 obj instanceof Foo
    - 。。。不想列举了
- 复杂数据类型
  - Object
    - let o = new Object // 合法，但不推荐
    - 实例都有如下属性和方法
      - constructor
      - hasOwnProperty
      - isPrototypeOf(object)
      - propertyIsEnumerable(propertyName)
      - toLocaleString()
      - toString()
      - valueOf()
      - BOM 和 DOM 对象可能会也可能不会继承 Object
  - 为什么没有Array?
    因为Array也是Object类型
  - 为什么没有null?
    因为null也是Object类型
  - 为什么没有Function?
    因为函数虽然被认为是一种对象，但它不是数据类型。

posted @ 2022-05-11 17:42 胡姐姐 阅读(6) 评论(0) 编辑(未发布)