|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **类型名** | **实例var obj=** | **Object.prototype.toString.call(obj)值** | **typeof(obj)值** | **obj.constructon.name** | **实例构造** | **备注** |
| number | 123 | "[object Number]" | "number" | "Number" | Number(objStr) |  |
| Number | new Number(123) | "[object Number]" | "object" | "Number" | new Number(objStr) |  |
| NaN | NaN | "[object Number]" | "number" | "Number" |  |  |
| string | “123” | "[object String]” | “string” | "String" |  |  |
| String | new String(“123”) | "[object String]” | "object" | "String" |  |  |
| boolean | false | "[object Boolean]” | "boolean" | "Boolean" |  |  |
| Boolean | new Boolean(false) | "[object Boolean]” | "object" | "Boolean" |  |  |
| Date | new Date() |  |  |  |  |  |
| Undefined | defined | "[object Undefined]” | "undefined" | 报错 |  |  |
| Null | null | "[object Null]" | "object" | 报错 |  |  |
| array | [“123”] | "[object Array]" | "object" | "Array" |  | 两者无法区分，其实是一样的 |
| Array | new Array([“123”]) | "[object Array]" | "object" | "Array" |  |
| function | function(){return null;} | "[object Function]" | "function" | "Function" |  | 两者无法区分，其实是一样的 |
| Function | new Function(“return null;”) | "[object Function]" | "function" | "Function" |  |
| object | {} | "[object Object]" | "object" | "Object" |  | 两者无法区分，其实是一样的 |
| Object | new Object() | "[object Object]" | "object" | "Object" |  |
| Person | new Person() | "[object Object]" | "object" | “Person” |  |  |
|  |  |  |  |  |  |  |

function Person(){} var obj=new Person();console.log(obj.constructor.name);还是要用constructor.name来判断才能判断最直接的父类类型

\*对于function类型，存字符串funcStr。 获取的时候可以var func=(new Function(‘return ’+funcStr))();