```
Object.prototype.toStringCustom = function () {
    /**
     * toString 的实现大致经历了三个阶段
     * 以下为 toString 的规范定义
     * https://262.ecma-international.org/6.0/#sec-object.prototype.tostring
     * 在 ecmascript 1.0 中: null 和 undefined 作为参数将会报错
     * 在 ecmascript 5.1 中: null 和 undefined 作为参数可以得到正确的结果
     * 在 ecmascript 6   中: 该方法在保证兼容的同时，使用 [@@toStringTag] 作为该方法的最终结果
     * 经过测试，[@@toStringTag] 的优先级是最高的，哪怕是原生类型，只要赋予 toStringTag 属性，那么便以该属性为准
     *
     * 至于 ecmascript 6 规范前的实现，应该是使用了一些内部工具方法和内部标记来完成的，
     * 经测试，改变一个原生对象的原型或者构造函数，无法改变 toString 的结果
     */

    /**
     * let array = [];
     * array.__proto__ = String.prototype;
     * Object.prototype.toString.call(array); // [object Array]
     * 说明内部实现与原型无关，估计使用了内部的不可见标记
     */

    /**
     * let array = [];
     * array[Symbol.toStringTag] = "真不错";
     * Array.prototype[Symbol.toStringTag] = "是真不错呀";
     * Object.prototype.toString.call(array); // [object 真不错]
     * 原型链上的 [@@toStringTag] 标记将会作为 toString 结果的最高优先级的参考值
     * [@@toStringTag] 的值必须为 string 类型，否则将会失效
     * ecmascript 6+ 规范中的新数据类型和数据结构都内置了 [@@toStringTag] 标识 (DOM 对象也有该标识)
     */

    if (this === null) {
        return '[object Null]';
    }

    if (this === undefined) {
        return '[object Undefined]';
    }

    if (typeof this !== 'object' || typeof this !== 'function') {
        // 如果参数不是对象，则使用 Object 方法将其转为包装对象
        this = Object(this);
    }

    if (typeof this[Symbol.toStringTag] === 'string') {
        return this[Symbol.toStringTag];
    }

    // 因为有部分规范是底层实现，所以非 [@@toStringTag] 部分使用套娃方式来表明其意
    return Object.prototype.toString.call(this);
}
```