**类型检测--typeof--instanceof--值类型与对象**

[类型检测--typeof--instanceof--值类型与对象](#KRC8-1620288223492)

[typeof](#IXWj-1620288258738)

[instanceof](#njNW-1620288663373)

[值类型与对象](#LKsz-1620289002373)

**typeof**

typeof 用于返回以下原始类型：

基本类型：number/string/boolean

function

object

undefined

可以使用typeof用于判断数据的类型

let a = 1;
console.log(typeof a); //number
let b = "1";
console.log(typeof b); //string
//未赋值或不存在的变量返回undefined
var hd;
console.log(typeof hd);
function run() {}
console.log(typeof run); //function
let c = [1, 2, 3];
console.log(typeof c); //object
let d = { name: "houdunren.com" };
console.log(typeof d); //object

**instanceof**

发音  [ˈɪnstənsof]

instanceof 运算符用于检测构造函数的 prototype 属性是否出现在某个实例对象的原型链上

也可以理解为是否为某个对象的实例， typeof 不能区分数组，但是 instanceof 则可以

    
let vue1 = []
let vue2 = {}
console.log(typeof (vue1))  //Object
console.log(typeof (vue2))  //Object
console.log(vue1 instanceof Array)  //true
console.log(vue2 instanceof Array)  //false
console.log(vue2 instanceof Object) //true
    
let c = [1, 2, 3];
console.log(c instanceof Array); //true
    
let d = { name: "javaScript" };
console.log(d instanceof Object); //true
    
function User() { }
let e = new User();
console.log(e instanceof User); //true

**值类型与对象**

下面是使用字面量与对象方法创建字符串，返回的是不同类型。

let a = "javaScript";
let b = new String("vue");
console.log(typeof a, typeof b); //string object

只有对象才有方法使用，但在JS中也可以使用值类型调用方法，因为它会在执行时将值类型转为对象。

let a = "javaScript";
let b = new String("vue");
console.log(a.length); //10
console.log(b.length); //3