## Dd

1. 如果只是声明变量而没有赋值，则该变量的值是undefined。undefined是一个特殊的值，表示“无定义”。
2. 如果使用var重新声明一个已经存在的变量，是无效的。

var x = 1;

var x;

x // 1

上面代码中，变量x声明了两次，第二次声明是无效的。

但是，如果第二次声明的时候还进行了赋值，则会覆盖掉前面的值。

1. 变量提升

JavaScript 引擎的工作方式是，先解析代码，获取所有被声明的变量，然后再一行一行地运行。这造成的结果，就是所有的变量的声明语句，都会被提升到代码的头部，这就叫做变量提升（hoisting）。

var a;

console.log(a);

a = 1;

最后的结果是显示undefined，表示变量a已声明，但还未赋值

1. 对于var命令来说，JavaScript的区块不构成单独的作用域

{

var a = 1;

}

a // 1

1. 六种基本数据类型
2. 数值（number）：整数和小数（比如1和3.14）
3. 字符串（string）：文本（比如Hello World）。
4. 布尔值（boolean）：表示真伪的两个特殊值，即true（真）和false（假）
5. undefined：表示“未定义”或不存在，即由于目前没有定义，所以此处暂时没有任何值
6. null：表示空值，即此处的值为空。
7. 对象（object）：各种值组成的集合。

数值、字符串、布尔值这三种类型，合称为原始类型（primitive type）的值，即它们是最基本的数据类型，不能再细分了。对象则称为合成类型（complex type）的值，因为一个对象往往是多个原始类型的值的合成，可以看作是一个存放各种值的容器。至于undefined和null，一般将它们看成两个特殊值

对象是最复杂的数据类型，又可以分成三个子类型。

* 狭义的对象（object）
* 数组（array）
* 函数（function）

1. typeof

* typeof null = object
* typeof undefined = undefined

1. 下面六个值被转为false，其他值都视为true。

* undefined
* null
* false
* 0
* NaN
* ""或''（空字符串）

1. JavaScript 内部，所有数字都是以64位浮点数形式储存，即使整数也是如此。所以，1与1.0是相同的，是同一个数
2. 由于浮点数不是精确的值，所以涉及小数的比较和运算要特别小心。

0.1 + 0.2 === 0.3

// false

0.3 / 0.1

// 2.9999999999999996

(0.3 - 0.2) === (0.2 - 0.1)

// false

1. 数值精度
2. JavaScript浮点数的64个二进制位，从左边开始，是这样组成的

## Dd