JavaScript 变量可以用来保存两种类型的值：基本类型值和引用类型值。基本类型的值源自以下 5 种基本数据类型：Undefined、Null、Boolean、Number 和 String。基本类型值和引用类型值具有以下特点：

- 基本类型值在内存中占据固定大小的空间，因此被保存在栈内存中；
- 从一个变量向另一个变量复制基本类型的值，会创建这个值的一个副本；
- 引用类型的值是对象，保存在堆内存中；
- 包含引用类型值的变量实际上包含的并不是对象本身，而是一个指向该对象的指针；
- 从一个变量向另一个变量复制引用类型的值，复制的其实是指针，因此两个变量最终都指向同一个对象；

- 确定一个值是哪种基本类型可以使用 typeof 操作符，而确定一个值是哪种引用类型可以使用 instanceof 操作符。

所有变量（包括基本类型和引用类型）都存在于一个执行环境（也称为作用域）当中，这个执行环境决定了变量的生命周期，以及哪一部分代码可以访问其中的变量。以下是关于执行环境的几点总结：

- 执行环境有全局执行环境（也称为全局环境）和函数执行环境之分；
- 每次进入一个新执行环境，都会创建一个用于搜索变量和函数的作用域链；
- 函数的局部环境不仅有权访问函数作用域中的变量，而且有权访问其包含（父）环境，乃至全局环境；
- 全局环境只能访问在全局环境中定义的变量和函数，而不能直接访问局部环境中的任何数据；
- 变量的执行环境有助于确定应该何时释放内存。

JavaScript 是一门具有自动垃圾收集机制的编程语言，开发人员不必关心内存分配和回收问题。可以对 JavaScript 的垃圾收集例程作如下总结。

- 离开作用域的值将被自动标记为可以回收，因此将在垃圾收集期间被删除。
- “标记清除”是目前主流的垃圾收集算法，这种算法的思想是给当前不使用的值加上标记，然后再回收其内存。
- 另一种垃圾收集算法是“引用计数”，这种算法的思想是跟踪记录所有值被引用的次数。JavaScript 引擎目前都不再使用这种算法；但在 IE 中访问非原生 JavaScript 对象（如 DOM 元素）时，这种算法仍然可能会导致问题。
- 当代码中存在循环引用现象时，“引用计数”算法就会导致问题。
- 解除变量的引用不仅有助于消除循环引用现象，而且对垃圾收集也有好处。为了确保有效地回收内存，应该及时解除不再使用的全局对象、全局对象属性以及循环引用变量的引用。