Javascript 变量可以用来保存两种类型的值：基本类型和引用类型。基本类型的值源自以下5种基本数据结构：Undefined、Null、Boolean、Number、和String。基本类型和引用类型具有以下特点：

1. 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；
2. 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；
3. 引用类型的值是对象，保存在堆内存中；
4. 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；
5. 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

对非嵌套对象进行深拷贝：

```
Object.assign({},{a:nick});
```

对嵌套对象进行深拷贝：

```
let res = JSON.parse(JSON.stringify(from))
```

6. 确定一个值是那种基本类型可以使用 typeof 操作符，而确定一个值是那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（也成为作用域）当中，这个执行环境决定了变量的生命周期，以及那一部分代码可以访问其中的变量：

1. 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；
2. 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

作用域链的用途：

保证对执行环境有权访问的所有变量和函数的有序访问

结构：

前端：当前执行代码所在环境的变量对象=>父环境=>祖先环境=>...=>全局执行环境

3. 函数的局部环境不仅有权访问函数作用域中的变量，还有权访问其包含（父）环境，乃至全局环境
4. 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变量中的任何数据

5. 变量的执行环境有助于确定应该何时释放内存

JavaScript的垃圾收集

1. 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

2. “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

3. 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（DOM元素）时，这种算法仍然可能导致问题

4. 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

循环引用：

对象A中包含指向对象B的引用，对象B中包含指向对象A的引用，

```
function problem(){
    var objectA = new Object();
    var objectB = new Object();

    objectA.someOtherObject = objectB;
    objectB.anotherObject = objectA;
}
```

5. 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。为了确保有效的回收内存，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

```
function createPerson(name){
    var localPerson = new Object();
    localPerson.name = name;
    return localPerson;
}

var globalPerson = createPerson("Nicholas");

// 手工解除 globalPerson 的引用

globalPerson = null;
```