1. 作用域问题：
   1. 作用域提升：所有的**函数**和**变量**的声明都会被提升到最前面（var），赋值在声明之后（=）。
   2. 函数声明：
      1. 声明式（function \*\*(){}）：自动将声明放在前面，并执行赋值过程
      2. 变量式（var \*\* = funcction(){}）：将声明提升，然后到赋值处再执行赋值
2. 变量对象：
   1. 基本类型（保存在内存的栈中，大小固定，复制变量时会创建这个值的副本）：
      1. Number：表示整数和浮点数值
         1. 所有以八进制和十六进制表示的数值最终都将被转换成十进制数值：
            1. 八进制数：0后面跟0~7
            2. 十六进制：0x后面跟0~9、a~f（大小写均可）
         2. 浮点数：
            1. .1（适用不推荐）
            2. 1. 和 10.0 会被转化为整数
      2. String：
         1. toString()：
            1. null 和 undefined 值没有这个方法。
            2. 调用数字的toString() 可带一个进制参数，返回数值对应的进制数值字符串.
         2. String()：能将任何类型的值转化为字符串（String(null)：”null” ; String(undefined)：”undefined”）
      3. Boolean：Boolean()转型函数，if语句自动执行该函数：
         1. Boolean(true/非空字符串/非0数字/对象) 返回true;
         2. Boolean(false/空字符串/数字0/null/undefined) 返回false;
      4. Null：var \*\* = null;
      5. Undefined：var \*\*; var \*\* = undefined;

\*（undefined == null //true）

(NaN == NaN //false)

* 1. 引用类型（值是对象,保存在堆内存中）：object

1. typeof返回值：
   1. String
   2. Number
   3. Boolean
   4. Undefined：声明了未初始化 或 未声明
   5. Object：typeof null；
   6. Function
2. isNaN()函数：确定这个参数是否“不是数值;
   1. isNaN（数字/数字字符串/boolean值）返回false;
   2. isNaN（NAN/非数字字符串）返回true;

\*（只有 0 除以 0 才会返回 NaN）

1. 数值转化：
   1. Number()：
      1. Boolean 值，true 和 false 将分别被转换为 1 和 0.
      2. null 值，返回 0.
      3. 是 undefined，返回 NaN.
      4. 字符串：
         1. 数字字符串：转换为数字，前导的零被忽略.
         2. 有效的进制字符串：转换为相同大小的十进制整数值.
         3. 空字符串：转换为0.
         4. 其他：NAN.
      5. 对象：
   2. parseInt()：忽略字符串前面的空格，直至找到第一个非空格字符
      1. 第一个字符不是数字字符或者负号，会返回 NaN.（空字符串会返回 NaN）
      2. 继续解析第二个字符，直到解析完所有后续字符或者遇到了一个非数字字符.
      3. 有效进制字符串会转换为十进制.

（ parseInt(\*\*，\*\*) 中的第二个参数为进制，第一个参数可不带进制前缀）

* 1. parseFloat()：字符串中的第一个小数点是有效的，而第二个小数点就是无效的了：
     1. 始终都会忽略前导的零，不能解析其他进制.
     2. 整数字符串会转换为整数.

1. Undefined 和 null 的区别：
   1. 相同之处：完全不可变，没有属性和方法
   2. 不同之处：
      1. Undefined：没想到没有值
         1. 访问未初始化的声明变量
         2. 访问一个不存在的对象属性或数组项
         3. 声明变量的函数没有返回语句
         4. 访问未定义参数函数的参数时
      2. Null：预想到没有值
2. 函数声明和函数表达式：函数声明会在任何表达式被解析和求值之前先被解析和求值，即使你的声明在代码的最后一行，它也会在同作用域内第一个表达式之前被解析/求值
   1. 函数声明：函数声明必须带有标示符（函数名），只能出现在程序或函数体内
      1. function \*\*( ) { };
      2. ( function ( ) {

function \*\*( ) { };

}

)( );

* 1. 函数表达式：
     1. var \*\* = function \*\*\*(){};
     2. new function \*\*(){};
     3. ( function \*\*(){

}

)

1. 立即执行函数：
   1. ( function \*\* ( ) { } ) ( );
   2. ( function \*\* ( ) { } ( ) );
2. v