作用域基础概念

作用域是什么

理解作用域

- 处理var a = 2;
  - 引擎、编译器、作用域
- 引擎、编译器、作用域
  - 引擎：从头到尾负责整个js程序的执行以及执行过程
  - 编译器：负责语法分析以及代码生成等
  - 作用域
    - 负责手机并维护由所有生命的标识符（变量）组成的一系列查询。
    - 实施严格的规则，确定当前执行的代码对这些表示的访问权限
- var a = 2;变量的赋值
  - 1.编译器会在当前作用域中声明一个变量（前提是之前没有声明过）
  - 2.运行时js引擎会在作用域中查找该变量，如果查找到就对他赋值
- 编译器的幻匿术语
  - LHS：变量出现在复制操作的做测试进行LHS查询
  - RHS：变量出现在右侧是进行RHS查询

作用域是一套规则，用于确定在何处以及如何查找变量标识符

传统编译语言

- 程序中一段源代码执行前会经历三个步骤，统称为编译
- 分词/词法分析：将字符组成的字符串分解成有意义的代码块，称之为词法单元(token)
- 解析/语法分析（parsing）：将词法单元流，转换成一个由元素逐级嵌套所组成的程序语法树（抽象语法树(AST)）
- 代码生成：将AST转换为可执行代码，将AST语法树转化为一组机器指令

对于js来说编译发生在代码执行前的几微秒的间内

作用域嵌套

一个块节奏或函数嵌套在另一个块或函数时就发生了作用域的嵌套

遍历作用域链，从当前执行的作用域开始查找变量，如果找不到，就像上一级继续查找。直至找到全局作用域。

```
function foo(a){
  console.log(a+b);
  b = a;
  console.log(b)
}
foo(2)
```

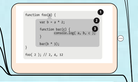
第一个console.log(a+b)抛出异常ReferenceError因为此时变量b未经声明就使用

第二个console.log(b)输出2，因为b=a的时候相当于自动的创建了一个全局变量b，并给他赋值

词法作用域

词法作用域就是定义在词法阶段的作用域

词法作用域是由你在写代码时将变量和块作用域写在哪里决定的。基本上在词法分析器处理代码时会保持作用域不变



作用域1，是全局作用域。其中只有一个标识符：foo

作用域2，是foo所创建的作用域。其中有三个标识符：a,bar,b

作用域3，是bar所创建的作用域。其中只有一个标识符c

作用域的结果和之间的位置关系，为引擎提供了足够的位置信息，引擎可以查找标识符的位置。

但是查找顺序是先从最内部的作用域开始找，依次往上级作用域查找

欺骗词法，修改词法作用域

eval

接收一个字符串为参数，并将其中的内容视为好像在书写时时候就存在于程序中这个位置的代码。

可以在你写的代码中用程序生成代码并运行，好像代码本身就是写在那个位置一样

eval()会调用"var b = 3"

var b = 3就会被当作本来就在哪里一样进行处理

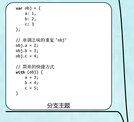
在foo函数内部声明并创建了一个变量b并赋值3

在函数foo内部使用时就直接使用这个b=3这个变量

在严格模式中evalyou自己的词法作用域。不能使用其来更改所在作用域

with

通常可以被看作重复引用同一个对象中的多个属性的快捷方式，可以不必重复引用对象本身



创建了o1、和o2两个对象

函数foo结束一个obj参数，该参数是一个对象的引用

对接收的obj对象引用执行了with（obj）{...}

在with内部将a并赋值2

当将o1传递进去，a=2找到了o1.a并将2赋值

而当 o2 传递进去，o2 并没有 a 属性，因此不会创建这个属性， o2.a 保持 undefined

实际上 a = 2 赋值操作创建了一个全局的变量 a

with 可以将一个没有或有多个属性的对象处理为一个完全隔离的词法作用域，因此这个对 象的属性也会被处理为定义在这个作用域中的词法标识符。

词法作用域意味着作用域是由书写代码时函数声明的位置来决定的

eval(..) 函数如果接受了含有一个或多个声明的代码，就会修改其所处的词法作用域，而 with 声明实际上是根据你传递给它的对象凭空创建了一个全新的词法作用域。