# 作用域

# 作用域

## 导入

在前面，我们学习了变量，我们知道了变量的声明的格式，可是变量的作用范围是无边无际的吗？显然不是

## 作用域

所谓变量的作用域就是变量可以起作用的范围

**全局变量**

|  |
| --- |
| 在任何地方都可以访问到的变量就是全局变量，对应**全局作用域** |

**局部变量**

|  |
| --- |
| 只在固定的代码片段内可访问到的变量，最常见的例如：函数内部。对应局部作用域(函数作用域) |

注意：不使用var声明的变量是全局变量，不推荐使用。

变量退出作用域之后会销毁，全局变量关闭网页或浏览器才会销毁

块级作用域

任何一对花括号({和})中的语句都属于一个块，在这之中定义的所有变量在代码块外都是不可见的，我们称为块级作用域。现阶段我们可以认为JavaScript没有块级作用域

词法作用域

变量的作用域是在定义时决定的而不是执行时决定，也就是说词法作用域取决于源码，通过静态分析就能确定，因此词法作用域也叫静态作用域。

在js中词法作用域规则：

函数允许访问函数外的数据

整个代码结构中只有函数可以限定作用域

作用域规则首先使用提升规则分析

如果当前作用规则中有名字了，就不考虑外面的名字

## 总结

在函数外面(js文件 script标签)定义的任何变量都是全局变量

在函数中定义的变量就是局部变量

全局作用域 就是全局变量的作用范围

局部作用域 就是局部变量的作用域范围 就是函数内部

块级作用域： 在大括号中声明的变量 不能被外面的语句访问到 现阶段js没有块级作用域

# 作用域链

## 导入

刚才我们说明了变量的作用域，变量的作用域说明了变量的作用范围，但是在多层函数嵌套的场景下，函数中的变量又是怎样的，只有通过学习作用域链才能解释

## 语法格式

只有函数可以制造作用域结构，那么只要是代码，就至少有一个作用域，即全局作用域。凡是代码中有函数，那么这个函数就构成另一个作用域。如果函数中还有函数， 那么在这个作用域中就可以诞生一个作用域。

将这样的所有的作用域列出来，可以有一个结构: 函数内指向函数外的链式结构。就称为作用域链

案例1：

|  |
| --- |
| function f1() {  function f2() {  }  }  var num = 456;  function f3() {  function f4() {  }  } |

案例2

|  |
| --- |
| function f1() {  var num = 123;  function f2() {  console.log( num );  }  f2();  }  var num = 456;  f1(); |



## 总结

函数内指向函数外的链式结构。就称为作用域链

# 预解析

## 导入

在我们写代码的过程中，有些地方会出现很有意思的事情，比如为什么函数声明在后，调用在前，函数居然没有报错，为了解释这样的事情，我们需要学习一下预解析

## 概念说明

所谓预解析，就是提前解析。JavaScript代码的执行是由JS解析器来执行的。JS解析器执行js代码的时候，分为两个过程: 预解析过程和代码执行过程。

预解析过程:

|  |
| --- |
| 1.把变量的声明提升到当前作用域的最前面，只会提升声明，不会提升赋值  2.把函数的声明提升到当前作用域的最前面，只会提升声明，不会提升调用  3.先提升声明var，再提升function |

## 总结

定义变量的时候，变量的声明会被提升到作用域的最上面，变量的赋值不会提升

使用函数的时候，函数的声明会被提升到整个作用域的最前面