Chapter 2. Variable object

Data declaration
数据声明

如果**变量和执行上下文有关**，我们必须搞懂与他相关的数据存储在那以及如何获取它们，这种机制被称为 **变量对象**

变量对象（简称VO）是一种与执行上下文密切相关的特殊对象，它存储了：

- 变量(var 变量声明)
- 方法声明（方法声明简称FD)
- 和方法型参

VO = {};

VO是执行上下文的一个属性

```
activeExecutionContext = {
  VO: {
    // context data (var, FD, function arguments)
  }
};
```

Variable object in different execution contexts
不同执行上下文中的变量对象

GlobalContextVO

FunctionContextVO

Variable object in global context
全局上下文中变量对象

```
global = {
  Math: <...>,
  String: <...>
  ...
  ...
  window: global
};
```

当引用全局对象的属性时前缀可以省去，因为全局对象不能通过名字直接访问。然

而，通过全局对象中的this或其引用可以访问它本身，例如BOM中的window，因此简写如下：

```
String(10); // means global.String(10);

// with prefixes
window.a = 10; // === global.window.a = 10 === global.a = 10;
this.b = 20; // global.b = 20;
```

Variable object in function context
方法上下文变量对象

在方法的执行上下文中VO是不能被直接访问的，同时被称作为**活动对象**(简写为AO)

Phases of processing the context code
代码的执行阶段

以下内容将是我们文章的精华。代码执行过程被分为两个阶段：

1. **进入执行上下文**
2. **执行代码**

Entering the execution context
进入执行上下文

当进入执行上下文(在进入**代码执行之前**)，**VO已被以下属性填充**(他们已经在上文中所阐明过)

- 方法的每一个型参(如在方法执行上下文中)
- 每一个方法申明(方法申明FD);
- 每一个变量声明

Code execution
代码执行

这时，**AO/VO已经被属性填充**(并不是所有属性都被我们传入的值所赋值，其中很大一部分只是初始值undefined)

```
alert(x); // function

var x = 10;
alert(x); // 10

x = 20;

function x() {}

alert(x); // 20
```

About variables
关于变量

```
a = 10;
alert(window.a); // 10

alert(delete a); // true

alert(window.a); // undefined

var b = 20;
alert(window.b); // 20

alert(delete b); // false

alert(window.b); // still 20
```

Conclusion
总结

在本文中我们学习了与执行上下文相关的对象。我希望这些知识点有所用，能清除你之前的一些理解歧义。按规划，我嗯下移章节将介绍 作用域链、标示符解析、闭包。