函数声明与函数表达式

本节到目前为止，我们一直没有对函数声明和函数表达式加以区别。而实际上，解析器在向执行环境中加载数据时，对函数声明和函数表达式并非一视同仁。解析器会率先读取函数声明，并使其在执行任何代码之前可用（可以访问）；至于函数表达式，则必须等到解析器执行到它所在的代码行，才会真正被解释执行。请看下面的例子。

```
alert(sum(10,10));
function sum(num1, num2){
    return num1 + num2;
}
```

FunctionDeclarationExample01.htm

以上代码完全可以正常运行。因为在代码开始执行之前，解析器就已经通过一个名为函数声明提升（function declaration hoisting）的过程，读取并将函数声明添加到执行环境中。对代码求值时，JavaScript 引擎在第一遍会声明函数并将它们放到源代码树的顶部。所以，即使声明函数的代码在调用它的代码后面，JavaScript 引擎也能把函数声明提升到顶部。如果像下面例子所示的，把上面的函数声明改为等价的函数表达式，就会在执行期间导致错误。

以上代码完全可以正常运行。因为在代码开始执行之前，解析器就已经通过一个名为函数声明提
（function declaration hoisting）的过程，读取并将函数声明添加到执行环境中。对代码求值时，JavaScr
引擎在第一遍会声明函数并将它们放到源代码树的顶部。所以，即使声明函数的代码在调用它的代码
面，JavaScript 引擎也能把函数声明提升到顶部。如果像下面例子所示的，把上面的函数声明改为等
的函数表达式，就会在执行期间导致错误。

```
alert(sum(10,10));
var sum = function(num1, num2){
    return num1 + num2;
};
```

FunctionInitializationExample01.htm

除了什么时候可以通过变量访问函数这一点区别之外，函数声明与函数表达式的语法其实是等价的。