Chapter 5. Functions.

Introduction
介绍

Type of functions
方法的类型

在ECMAScript中有三种不同特征的方法。

Function Declaration（简称FD）
方法定义(或方法申明)

方法的定义是：

- 必须拥有一个**方法名**
- 它是定位于源代码中的: 要么在**程序级别**中，要么在**其他方法体中**
- 在**进入上下文阶段被创造**
- **影响变量对象**

这种类型的方法有个显著的特征：**仅影响变量对象**（在上下文中，他们被存放在VO中）

注意：ECMAScript中方法仅有可能在两个位置中申明(要不是在**表达式中**，要不在**代码块中**)

Function Expression（简称FE）
方法表达式

方法表达式是：

- 在源代码中在**表达式位置被定义**
- **方法名是可选**的
- 他的定义**不会对变量对象产生影响**
- 在**代码执行阶段被创建**

下例中我们可以看到各种ECMAScript表达式中的FE

```
// in parentheses (grouping operator) can be only an expression
(function foo() {});

// in the array initialiser – also only expressions
[function bar() {}];

// comma also operates with expressions
1, function baz() {};
```

这些定义再次确定**FE创建于代码执行阶段**而且**不影响VO**.见例

```
alert(foo); // "foo" is not defined

(function foo() {});

alert(foo);  // "foo" is not defined
```

我们到底需要这种类型的方法做什么？答案很简单：仅在**表达式中使用他们**，**不污染VO**。这点可以通过将方法作为参数的形式进行示范。

Question “about surrounding parentheses”
关于括号的问题

"当我们需要在方法定义后立即调用它时，为什么要用括号将它括住"。这个问题的答案：**表达式语句规则**

根据规范：
1. 表达式不能以大括号开始｛，避免与代码块混淆，
2. 表达式语句不能以function关键词开始，避免和方法申明混淆。

关于括号的所有完整解释如下:
1. 当**方法不在表达式位置**而我们又想在他创建后立即调用它那么**括号是必须的**，这样我们会手动将方法变为FE
2. 如果解析器将他处理为FE，即**方法本身就在表达式位置**，那么**括号不是必需的**

Implementations extension: Function Statement
实现扩展：方法申明

```
if (true) {

  function foo() {
    alert(0);
  }

} else {

  function foo() {
    alert(1);
  }

}

foo(); // 1 or 0 ? test in different implementations
```

如果严格按照规范这样的语法通常是错误的，因为我们了解，方法申明不能出现在代码块中(这里if else语句有代码块)。之前也提过，FD只能在两个位置出现(程序级别或别的方法中);

所以上例是错误的，因为代码块不能只包含申明。只有在方法的代码块中能包含这样的申明 - 表达式申明。但根据规范，他不能大括号或function关键词开头(避免同FD混淆);

然而在错误类型标准允许实现对语法错误扩展。这种扩展后方法可以出现在块中。目前所有实现中都能执行且没有一个报错,但是都是按照他自己的一套执行。

Feature of Named Function Expression (NFE)
有名方法表达式(NFE)特性(区别于无名)

Functions created via Function constructor
通过Function构造函数创建的方法

这中类型的方法对象和FD FE都不同，但是他有自己的特性。最重要的特性就是他的[[Scope]]属性只包含全局对象。

```
var x = 10;

function foo() {

  var x = 20;
  var y = 30;

  var bar = new Function('alert(x); alert(y);');

  bar(); // 10, "y" is not defined

}
```

Algorithm of function creation

方法创建法则