
# SimCube JavaScript

Version 1.4

编写：谭立方

日期：2015.6.17

# 目 录

# 1. 前言

本文档是 SimCube（仿立方）系列软件中专为 SimCube 定制的 JavaScript 编程接口使用指南。

在 SimCube 系列软件中，如 Automation 提供了 JavaScript 的脚本环境，用户可通过编写 JavaScript 脚本与软件交互。SimCube 中的 JavaScript 解释器除具备标准 ECMA-262（http://www.ecma-international.org/publications/standards/Ecma-262.htm）中的功能之外，为了满足自身软件应用的需求，还提供了更多的编程功能接口。本手册不提供 JavaScript 基础的介绍，用户可参考 JavaScript：The Definitive Guide（中文译本：JavaScript 权威指南 第 6 版）学习该语言，这里只介绍 SimCube 中非标准的 JavaScript 编程功能接口。

SimCube 的 JavaScript 解释器（下文中如无特别说明，提到的 JavaScript 解释器也即 SimCube 中的解释器）提供了丰富的全局函数、对象、类，以及相关方法与属性。按 JavaScript 语言的命名约定，全局函数、对象以及方法名称的首字母使用小写，类与属性的首字母使用大写。表 1 全局函数、对象与类的列表中提供了简要说明，下述各节将详细描述这些函数、对象与类的功能。

**表 1 全局函数、对象与类的列表**

| 名称 | 类型 | 说明 |
|---|---|---|
| fs | 对象 | 访问文件系统的功能 |
| office | 对象 | 访问微软 office 中的 docx 与 xlsx 文件 *版本 1.1* |
| os | 对象 | 获取操作系统相关信息的功能 |
| path | 对象 | 操作文件路径的功能 |
| process | 对象 | 当前应用的进程对象 |
| sql | 对象 | 访问数据库的功能 |
| ui | 对象 | 访问 GUI 的功能 *版本 1.1* |

| | | |
|---|---|---|
| log | 全局函数 | 用于把变量的值输出到控制台窗口 |
| __dirname | 字符串 | JavaScript 脚本文件中已初始化的变量，其值是当前文件所在的绝对路径，不包含文件名 |
| __filename | 字符串 | JavaScript 脚本文件中已初始化的变量，其值是当前文件所在的绝对路径，包含文件名 |
| require | 全局函数 | 提供模块功能，可动态加载 JavaScript 编写的模块 |
| Buffer | 类 | 二进制表示的字节缓冲区 |
| ChildProcess | 类 | 启动子进程，与子进程通信 |
| File | 类 | 与文件 IO 的处理 |
| DataStream | 类 | 数据流，与 File 或 Buffer 类联合起来使用 |
| TextStream | 类 | 文本流，与 File 或 Buffer 类联合起来使用 |
| DatabaseObject | 类 | 数据库类，访问数据库，执行查询、事务、回滚等 |
| SqlQuery | 类 | 访问数据库的数据查询类 |
| SqlRecord | 类 | 访问数据库的数据记录类 |
| SqlError | 类 | 访问数据库的数据错误类 |
| XmlStreamAttribute | 类 | 表示 XML 元素中一个属性的类 |
| XmlStreamReader | 类 | XML 流读取器 |

# 2. os 对象

使用对象 os 中的各种方法获取运行应用程序操作系统的各种信息。

## 2.1 os.tmpdir()

该方法用于获取操作系统中默认的用于存放临时文件的目录。

## 2.2 os.endianness()

该方法用于获取 CPU 的字节序，可能的值是“BE”或“LE”。

## 2.3 os.hostname()

该方法用于获取计算机名。

## 2.4 os.type()

该方法用于获取操作系统的类型。

## 2.5 os.arch()

该方法用于获取 CPU 架构，实际上是返回应用本身的 CPU 架构，在 Windows 系统上如果 32 位的应用程序运行在 64 位机器上，将返回“x86”，而不会返回“x64”。

## 2.6 os.uptime()

该方法用于获取操作系统的当前运行时间，单位为秒。

## 2.7 os.totalmem()

该方法返回系统的总内存量，单位为字节。

## 2.8 os.freemem()

该方法返回系统的空闲内存量，单位为字节。

## 2.9 os.cpus()

该方法返回机器的总 CPU 数量。

## 2.10 os.networkInterfaces()

该方法返回一个对象，其中存放了系统中的所有网络接口。

## 2.11 os.eol

该属性为操作系统中使用的换行符。一般在 Windows 系统中是“\r\n”，其它系统是“\n”。

# 3. path 对象

使用对象 path 中的各种方法处理和转换文件路径。

## 3.1 path.normalize(path)

该方法对路径 path 进行正规化，返回的路径中没有“..”和“.”部分。

## 3.2 path.join([path1], [path2], […])

该方法连接所有参数中的路径，并把最终的路径正规化后返回。

## 3.3 path.dirname(path)

该方法返回路径 path 的目录名称，不包含文件名。

## 3.4 path.basename(path)

该方法返回路径 path 的文件名称，不包括文件名后缀。

## 3.5 path.filename(path)

该方法返回路径 path 的文件名称。

## 3.6 path.extname(path)

该方法返回路径 path 的扩展名，从最后一个“.”开始的路径 path 后面部分。如果 path 中没有“.”则返回空字符串。

## 3.7 path.sep

该属性返回平台相关的文件分隔符，“\”或“/”，在 Windows 系统下返回前者，其它系统返回后者。

## 3.8 path.delimiter

该属性返回平台相关的路径分隔符，“;”或“:”，在 Windows 系统下返回前者，其它系统返回后者。

# 4. fs 对象

使用对象 fs 中的各种操作文件系统中的文件与目录。

## 4.1 fs.copy(fileName, newName)

该方法拷贝文件 fileName 至 newName。如果成功返回 true，否则返回 false。如果文件 newName 已经存在，那么拷贝将失败并返回 false。

## 4.2 fs.copyDirectory(srcFilePath, targetFilePath)

该方法递归地拷贝目录 srcFilePath 中的内容至目录 targetFilePath 中，若后者不存在则将创建。如果成功返回 true，否则返回 false。*版本 1.2 中实现。*

## 4.3 fs.exists(path)

该方法判断 path 是否是一个存在的文件或目录，如果是则返回 true，否则返回 false。

## 4.4 fs.link(srcPath, dstPath)

该方法创建文件 srcPath 的链接，新的链接是 dstPath，在 Windows 系统下是快捷方式，Unix 类系统下是符号链接。如果成功则返回 true，否则返回 false。

## 4.5 fs.mkdir(path)

该方法创建 path 中指定的目录，path 中所有的父目录都将创建。如果成功则返回 true，否则返回 false。

## 4.6 fs.openUrl(url)

该方法以操作系统默认的方式打开 url，此参数可以是一个本地文件，也可以是一个网址，例如 fs.openUrl('http://cn.bing.com/search?q=pera.simcube')，将打开浏览器并使用必应搜索 pera.simcube。另外也可以使用它打开关联的邮件客户端发送邮件，例如 fs.openUrl('mailto:SimCube@peraglobal.com?subject=SimCube 软件使用反馈&body=反

馈如下：')，打开邮件客户端发送邮件给 SimCube@peraglobal.com 反馈软件的使用情况。*版本 1.3 中实现。*

## 4.7 fs.readdir(path)

该方法返回 path 中指定目录中所有的文件名与子目录名的数组。如果路径 path 不存在则返回空数组。

## 4.8 removeDirectory(path)

该方法递归删除目录 path 及其子目录中的所有内容。如果成功返回 true，否则返回 false。*版本 1.2 中实现。*

## 4.9 fs.rename(oldPath, newPath)

该方法对文件或目录重命名，如果成功则返回 true，否则返回 false。

## 4.10 fs.rmdir(path)

该方法删除 path 中指定的目录，包括所有的父目录，目录必须为空才可删除。如果成功返回 true，否则返回 false。

## 4.11 fs.stat(path)

该方法以一个对象返回 path 中指定目录或文件的信息，如果 path 指定的文件或目录不存在，则返回一个空对象。对象中包含 6 个键值对。

| 键名称 | 说明 |
|---|---|
| isFile | 如果是文件则返回 true，否则返回 false |
| isDir | 如果是目录则返回 true，否则返回 false |
| size | 返回文件的大小，以字节为单位，若是目录则返回 0 |
| atime | 文件的访问时间 |
| ctime | 文件的创建时间 |
| mtime | 文件的修改时间 |

### 4.12 fs.truncate(path, len)

该方法对文件 path 进行截断操作，操作成功后的文件长度是 len，以字节为单位。如果成功返回 true，否则返回 false。

### 4.13 fs.unlink(path)

该方法删除文件 path。如果成功则返回 true，否则返回 false。

### 4.14 fs.unzip(zipFile, unzipDir)

该方法把以通用的 zip 格式文件 zipFile 解压缩到目录 unzipDir 中，如果该目录不存在则会先创建。如果操作成功返回 true，否则返回 false。*版本 1.2 中实现*。

### 4.15 fs.zip(zipFile, zipPath)

该方法把文件或目录 zipPath 中的所有内容以通用的 zip 压缩格式压缩为文件 zipFile，参数 zipPath 既可以是一个文件，也可以是一个目录。如果操作成功返回 true，否则返回 false。*版本 1.2 中实现*。

## 5. process 对象

SimCube 中带有 JavaScript 解释器的应用都有一个全局的 process 对象，通过该对象可访问应用进程的相关信息，以及与 JavaScript 解释器的交互操作。

### 5.1 process.appVersion

该属性用于返回应用程序的版本信息。

### 5.2 process.arguments

该属性返回执行应用程序的参数列表数组。

### 5.3 process.availableExtensions

该属性返回以字符串数组表示的可以加载的各种扩展，也即插件。

## 5.4 process.chdir(path)

该方法用于修改应用的当前工作目录为 path。参数 path 为一个字符串，用于指定当前工作目录，该目录可以为一个相对路径，也可以为一个绝对路径。如果成功修改则返回 true，否则返回 false。

```
log('当前目录: ', process.cwd());
process.chdir('../');
log('上层目录: ', process.cwd());
```

## 5.5 process.cwd()

该方法返回应用的当前工作目录。

## 5.6 process.env([key])

该方法返回一个对象，其中包含了运行应用的操作系统环境变量信息。可给定一个字符串参数，表示只返回该参数的环境变量，否则返回所有环境变量。

## 5.7 process.error([arg1], [arg2], […])

该方法可以往标准错误输出（如果是控制台程序）或者 JavaScript Shell 窗口中（非控制台程序）输出各个参数信息，错误信息与一般信息将有可能用不同的颜色表示。

## 5.8 process.execPath

该属性值返回应用程序的可执行文件的绝对路径。

## 5.9 process.exit(returnCode)

该方法可以使得应用的进程退出，并返回 returnCode 整数值，如果是成功退出，返回码可设置为 0，否则为非 0。在图形界面软件下请不要使用此方法，将导致软件异常。

## 5.10 process.importedExtensions

该属性返回以字符串数组表示的已经加载的各种扩展，也即插件。

## 5.11 process.importExtension([ext1], [ext2], […])

该方法加载扩展，参数是各个扩展的名称，这些名称可以通过 process.availableExtensions 属性获取。

## 5.12 process.load(file1, [file2], […])

该方法加载脚本文件 file1，或多个脚本文件，依次在解释器中执行。解释器只是简单的执行这些脚本，不会更新图形界面的状态，如在自动化应用中当使用此方法加载一个创建模型的脚本时，模型视图界面不会得到刷新，因此不适合使用此方法直接加载应用相关的脚本文件。

## 5.13 process.mainWindow

该属性返回当前应用程序的主窗口对象，如果当前应用不是图形界面程序，那么该属性未定义。

## 5.14 process.msleep(mseconds)

该方法使当前执行 JavaScript 代码的线程睡眠 mseconds 毫秒。

## 5.15 process.pid

该属性返回当前运行应用进程的 PID 值。

## 5.16 process.QtVersion

该属性值返回应用程序使用的 Qt 版本。

## 5.17 process.quit()

该方法可以使得应用的进程成功退出，相当于调用 process.exit(0)方法。

## 5.18 process.read([arg1], [arg2], […])

该方法可以从标准输入（如果是控制台程序）或者一个输入对话框中（非控制台程序）获取一个用户输入的字符串数据，方法的参数将作为提示信息显示，如果有多个参数，则使用空格连接。

## 5.19 process.sleep(seconds)

该方法使当前执行 JavaScript 代码的线程睡眠 seconds 秒。

## 5.20 process.uptime()

该方法用于获取应用程序的当前运行时间，单位为秒。

# 6. office 对象[1]

全局作用域中的 office 对象提供访问微软 Office 中的 docx、xlsx 两种文件类型。对象 office 中提供了 Docx 类，用于操作 docx 文件，如读入 docx 文件，与其交互，然后写出 docx 文件；提供的 Xlsx 类，用于操作 xlsx 文件，如读入 xlsx 文件，与其交互，然后写出 xlsx 文件。与这两种文件类型交互时，不要求计算机安装 Office 软件。

## 6.1 Xlsx 类

该类提供了读取、创建 xlsx 文件的各种方法，同时在该类的名字空间下有多个与 xlsx 文件操作相关的类，如 Cell 类，表示单元格；Format 类，表示单元格中的格式；Worksheet 类，表示工作表。

有三种方式创建一个 Xlsx 对象：

1. new office.Xlsx()
2. new office.Xlsx(filename)

[1] 在版本 1.1 中引入

3. new office.Xlsx(fileobj)

第一种方法创建的 Xlsx 对象没有与 xlsx 文件关联，可通过 saveAs 方法保存为 xlsx 文件；第二种方法打开字符串 filename 指定的 xlsx 文件创建的 Xlsx 对象；第三种方法的 fileobj 是从一个 File 类的对象中创建，File 类对象打开了一个 xlsx 文件。

### 6.1.1 常量

表示页面类型的常量：

| 常量 | 值 | 说明 |
|---|---|---|
| Xlsx.WorkSheet | 0 | 工作表 |
| Xlsx.ChartSheet | 1 | 图表，未实现 |
| Xlsx.DialogSheet | 2 | 对话框表，未实现 |
| Xlsx.MacroSheet | 3 | 宏表，未实现 |

### 6.1.2 方法

#### 6.1.2.1 addSheet([name], [type=WorkSheet])

该方法以 name 为名称（若不提供，则是默认的名称，如 Sheet1 等）添加一个工作表页面（默认，目前不支持其它类型页面）。若成功则返回 true，否则返回 false。

#### 6.1.2.2 cellAt(row, col), cellAt(cell)

该方法获得单元格的对象，也即 Cell 对象。指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。若成功获得了单元格对象，则返回该对象，否则返回 undefined 值。

#### 6.1.2.3 copySheet(srcName, distName)

该方法拷贝字符串 srcName 表示的页面为字符串 distName 表示的页面，若成功则返回 true，否则返回 false。

#### 6.1.2.4 deleteSheet(name)

该方法删除字符串 name 表示的页面，若成功则返回 true，否则返回 false。

#### 6.1.2.5 insertSheet(index, [name], [type = WordSheet])

该方法在 index 处插入名称为 name（若不提供则使用默认的名称，如 sheet1 等）工作表 type（默认为 WordSheet, 目前不支持其它类型）。若成功则返回 true，否则返回 false。

#### 6.1.2.6 renameSheet(oldName, newName)

该方法将名称为 oldname 的工作表重命名为 newName, 若成功则返回 true，否则返回 false。

#### 6.1.2.7 moveSheet(srcName, distIndex)

该方法把工作表srcName移动到distIndex位置，若成功则返回ture，否则返回false。

#### 6.1.2.8 documentProperty(key)

该方法得到文档属性 key 对应的内容。如得到文档的创建者。可使用的 key 如下表中所示：

| key | 描述 |
|---|---|
| title | 文档的标题。 |
| subject | 文档的主题。 |
| creator | 文档的创建者。 |
| company | 文档创建者的公司。 |
| category | 文档内容分类，值可以是 Resume, Letter, Financial Forecast, Proposal, Technical。 |
| keywords | 支持搜索和索引的关键字。 |

| | |
|---|---|
| description | 文档内容的描述。 |
| contentStatus | 文档的状态，值可以为 Draft, Reviewed, Final 等。 |

6.1.2.9 setDocumentProperty(key, property)

该方法设置文档属性 key 的内容，如设置文档创建者。Key 的取值参考 6.1.2.8 的表格内容。

6.1.2.10 documentPropertyNames()

该方法得到文档所有的属性的名称。

6.1.2.11 rowCount()

该方法返回当前工作表的有效行数。

6.1.2.12 columnCount()

该方法返回当前工作表的有效列数。

6.1.2.13 currentWorksheet()

该方法返回当前的工作表对象。

6.1.2.14 mergeCells(range, [format])

该方法合并指定的单元格范围 range，并设置其样式 format(如果不提供则使用文档默认样式)。若成功则返回 ture，否则返回 false。如合并 A3 到 C8 的单元格范围, mergeCells(‘A3:C8’)。

6.1.2.15 unmergeCells(range)

该方法取消合并单元格范围 range, 若成功则返回 ture，否则返回 false。如取消合并 A3 到 C8 的单元格范围，unmergeCells(‘A3:C8’)。

6.1.2.16 insertImage(row, col, imgPath|img)

该方法是在行 row 列 col 处插入图片，可以使用图片路径 imgPath，也可以直接插入图片对象 img(QImage)。若成功则返回 ture，否则返回 false。

6.1.2.17 setColumnWidth(width, colFirst, [colLast])

该方法设置列的宽度，只提供 colFirst，则设置单列 colFirst 的宽度。提供 colLast 值可以设置连续多列的宽度。若成功则返回 ture，否则返回 false。

6.1.2.18 setColumnFormat(format, colFirst, [colLast])

该方法设置列的字体样式，只提供 colFirst，则设置单列 colFirst 样式 ，提供 colLast 值可以设置连续多列的样式。若成功则返回 ture，否则返回 false。

6.1.2.19 setColumnHidden(hidden, colFirst, [colLast])

该方法设置是否隐藏列。只提供 colFirst，则设置单列。提供 colLast 值可以设置连续多列是否隐藏。若成功则返回 ture，否则返回 false。

6.1.2.20 columnWidth(column)

该方法返回列 column 的宽度。

6.1.2.21 columnFormat(column)

该方法返回列 column 的样式。

6.1.2.22 isColumnHidden(column)

该方法返回列 column 是否是隐藏状态。返回 true 为隐藏状态，false 为显示状态。

6.1.2.23 setRowHeight(height, rowFirst, [rowLast])

该方法设置行的高度，只提供 rowFirst，则设置单行的高度，提供 rowLast 值可以设置连续多行的高度。若成功则返回 ture，否则返回 false。

6.1.2.24 setRowFormat(format, rowFirst, [rowLast])

该方法设置行的字体样式，只提供 rowFirst，则设置单行的样式，提供 rowLast 可以设置连续多行的样式。若成功则返回 ture，否则返回 false。

6.1.2.25 setRowHidden( hidden, rowFirst, [rowLast])

该方法设置行是否隐藏，只提供 rowFirst，则设置单行是否隐藏，提供 rowLast 可以设置连续多行是否隐藏。若成功则返回 ture，否则返回 false。

6.1.2.26 rowHeight(row)

该方法返回行 row 的高度。

6.1.2.27 rowFormat(row)

该方法返回行 row 的样式。

6.1.2.28 isRowHidden(row)

该方法判断行 row 是否是隐藏状态。返回 true 为隐藏状态，false 为显示状态。

6.1.2.29 groupRows(rowFirst, rowLast, [collapsed])

该方法设置把行从 rowFirst 到 rowLast 分成一组，collapsed 表示这一组是否是合并状态，默认是 true。若成功则返回 ture，否则返回 false。

6.1.2.30 groupColumns(colFirst, colLast, [collapsed])

该方法设置列从 colFirst 到 colLast 分成一组,collapsed 表示这一组是否是合并状态，默认是 true。若成功则返回 ture，否则返回 false。

6.1.2.31 defineName(name, formula, [comment], [scope])

该方法用于创建一个列区间，公式的名称 name, formula 是列的区间，描述 comment（默认为空），scope 为工作表名称(默认为空，表示是全局区域)。若创建成功则返回

ture，否则返回 false。

如 xlsx.defineName("demo", "=Sheet1!$C$1:$C$10", "", "Sheet1");

xlsx.write(11, 1, "=SUM(demo)");

6.1.2.32 read(row, col), read(cell)

该方法读取单元格的值并返回单元格中的值，若此单元格无值或读取错误，则返回 undefined。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

6.1.2.33 save()

该方法保存当前文档至文件系统中，若未指定文档的名称，则以缺省名称“book1.xlsx”保存。若成功保存返回 true，否则返回 false。

6.1.2.34 saveAs(name)

该方法保存文档至 name 文件中。若成功保存返回 true，否则返回 false。

6.1.2.35 selectSheet(name)

该方法选择 name 表示的工作表为当前活动的工作表。若成功返回 true，否则返回 false。

6.1.2.36 sheetNames()

该方法返回当前文档中所有的工作表名称数组。

6.1.2.37 worksheet(sheetName)

该方法返回 sheetName 表示的工作表(Worksheet)对象，若无此工作表，则返回 null。

6.1.2.38 write(row, col, value, [format]), write(cell, value, [format])

该方法以格式 format 写入值 value 至指定的单元格中。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

## 6.2 Xlsx.Cell 类

该类也是在 office 全局对象之下，它没有提供构造函数，不可直接创建，而只能通过 Xlsx 的 cellAt 方法获得。

### 6.2.1 常量

单元格类型常量

| 常量 | 值 | 说明 |
| --- | --- | --- |
| Xlsx.Cell.BooleanType | 0 | 布尔值 |
| Xlsx.Cell.NumberType | 1 | 数值 |
| Xlsx.Cell.ErrorType | 2 | 错误类型 |
| Xlsx.Cell.SharedStringType | 3 | 共享的字符串类型 |
| Xlsx.Cell.StringType | 4 | 字符串类型 |
| Xlsx.Cell.InlineStringType | 5 | 内联字符串类型 |

### 6.2.2 属性

#### 6.2.2.1 cellType

该属性获取单元格对象的类型。

#### 6.2.2.2 dateTime

该属性获取日期时间类型单元格的日期时间值。

#### 6.2.2.3 format

该属性获取单元格的格式对象。

#### 6.2.2.4 hasFormula

该属性获取单元格是否是一个公式。

#### 6.2.2.5 isDateTime

该属性获取单元格是否上一个日期时间。

#### 6.2.2.6 isRichString

该属性获取单元格是否是一段富文本。

#### 6.2.2.7 value

该属性获取单元格的数据内容。

## 6.3 Xlsx.Format 类

该类也是在 office 全局对象之下，可以通过 new office.Xlsx.Format() 或 new office.Xlsx.Format(format)创建一个格式对象。

### 6.3.1 常量

字体类型常量

| 常量 | 值 | 说明 |
|---|---|---|
| Xlsx.Format.FontScriptNormal | 0 | 正常字体 |
| Xlsx.Format.FontScriptSuper | 1 | 上标 |
| Xlsx.Format.FontScriptSub | 2 | 下标 |

字体位置

| 常量 | 值 | 说明 |
|---|---|---|
| Xlsx.Format. AlignLeft | 1 | 居左 |

| | | |
|---|---|---|
| Xlsx.Format. AlignHCenter | 2 | 居中 |
| Xlsx.Format. AlignRight | 3 | 居右 |

### 6.3.2 属性

#### 6.3.2.1 fontSize

该属性设置或获取格式中的字体尺寸大小。

### 6.3.3 方法

#### 6.3.3.1 setHorizontalAlignment(align)

该方法设置文字的位置。align 使用字体位置的常量。

## 6.4 Xlsx.Worksheet 类

该类也是在 office 全局对象之下，它没有提供构造函数，不可直接创建，而只能通过 Xlsx 对象的 worksheet 方法获得。

### 6.4.1 方法

#### 6.4.1.1 cellAt(row, column), cellAt(cell)

该方法获取单元格对象。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

#### 6.4.1.2 read(row, column), read(cell)

该方法返回读取的单元格中的值。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

6.4.1.3 write(row, column, value, [format]), write(cell, value, [format])

该方法以格式 format 写入值 value 至指定的单元格中。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

6.4.1.4 writeBlank(row, column, [format]), writeBlank(cell, [format])

该方法以格式 format 写入空白至指定的单元格中。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

6.4.1.5 writeBool(row, column, value, [format]), writeBool(cell, value, [format])

该方法以格式 format 写入布尔值 value 至指定的单元格中。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

6.4.1.6 writeDateTime(row, column, value, [format]), writeDateTime (cell, value, [format])

该方法以格式 format 写入日期时间值 value 至指定的单元格中。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

6.4.1.7 writeInlineString(row, column, value, [format]), writeInlineString (cell, value, [format])

该方法以格式 format 写入内联字符串 value 至指定的单元格中。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

6.4.1.8 writeNumeric(row, column, value, [format]), writeNumeric (cell, value, [format])

该方法以格式 format 写入数值 value 至指定的单元格中。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

6.4.1.9 writeString(row, column, value, [format]), writeString (cell, value, [format])

该方法以格式 format 写入字符串 value 至指定的单元格中。

指定一个单元格有两种方法，一种是基于 1 开始的行列整数索引，另一种是使用字符串表示的，如“A2”，表示第一列，第二行的单元格。

6.4.1.10 rowCount()

该方法得到工作表的有效行数。

6.4.1.11 columnCount()

该方法得到工作表的有效列数。

6.4.1.12 mergeCells(range, [format])

该方法合并指定的单元格范围 range，并设置其样式 format(如果不提供则使用文档默认值)。若成功则返回 ture，否则返回 false。如合并 A3 到 C8 的单元格范围, mergeCells(‘A3:C8’)。

6.4.1.13 unmergeCells(range)

该方法取消合并单元格范围 range, 若成功则返回 ture，否则返回 false。如取消合并 A3 到 C8 单元格范围，unmergeCells(‘A3:C8’)。

6.4.1.14 insertImage (row, col, imgPath)

该方法是在行 row 列 col 处插入图片 imgPath(图片路径)。若成功则返回 ture，否

则返回 false。

6.4.1.15 setColumnWidth (width, colFirst, [colLast])

该方法设置列的宽度，只提供 colFirst，则设置单列 colFirst 的宽度。提供 colLast 值可以设置连续多列的宽度。若成功则返回 ture，否则返回 false。

6.4.1.16 setColumnFormat (format, colFirst, [colLast])

该方法设置列的字体样式 format，只提供 colFirst，则设置单列 colFirst 样式 ，提供 colLast 值可以设置连续多列的样式。若成功则返回 ture，否则返回 false。

6.4.1.17 setColumnHidden (hidden, colFirst, [colLast])

该方法设置是否隐藏列。只提供 colFirst，则设置单列。提供 colLast 值可以设置连续多列是否隐藏。若成功则返回 ture，否则返回 false。

6.4.1.18 columnWidth(column)

该方法返回列 column 的宽度。

6.4.1.19 columnFormat(column)

该方法返回列 column 的样式。

6.4.1.20 isColumnHidden(column)

该方法返回列 column 是否是隐藏状态。返回 true 为隐藏状态，false 为显示状态。

6.4.1.21 setRowHeight(height, rowFirst, [rowLast])

该方法设置行的高度，只提供 rowFirst，则设置单行的高度，提供 rowLast 值可以设置连续多行的高度。若成功则返回 ture，否则返回 false。

6.4.1.22 setRowFormat(format, rowFirst, [rowLast])

该方法设置行的字体样式，只提供 rowFirst，则设置单行的样式，提供 rowLast

可以设置连续多行的样式。若成功则返回 ture，否则返回 false。

6.4.1.23 setRowHidden(hidden, rowFirst, [rowLast])

该方法设置行是否隐藏，只提供 rowFirst，则设置单行是否隐藏，提供 rowLast 可以设置连续多行是否隐藏。若成功则返回 ture，否则返回 false。

6.4.1.24 rowHeight(row)

该方法返回行 row 的高度。

6.4.1.25 rowFormat(row)

该方法返回行 row 的样式。

6.4.1.26 isRowHidden(row)

该方法判断行 row 是否是隐藏状态。返回 true 为隐藏状态，false 为显示状态。

6.4.1.27 groupRows(rowFirst, rowLast, [collapsed])

该方法设置把行从 rowFirst 到 rowLast 分成一组 collapsed 表示这一组是否是合并状态，默认是 true。若成功则返回 ture，否则返回 false。

6.4.1.28 groupColumns(colFirst, colLast, [collapsed])

该方法设置列从 colFirst 到 colLast 分成一组,collapsed 表示这一组是否是合并状态，默认是 true。若成功则返回 ture，否则返回 false。

6.4.1.29 isWindowProtected()

该方法查看工作表是否是保护状态。返回 true 为保护状态，false 为非保护状态。

6.4.1.30 setWindowProtected(protect)

该方法设置工作表是否是保护状态。protect 为 true 表示保护，false 表示不保护。

6.4.1.31 isFormulasVisible()

该方法查看是否显示公式。返回 true 为显示状态，false 为隐藏状态。

6.4.1.32 setFormulasVisible(visible)

该方法设置是否显示公式。visible 为 true 表示显示，false 表示不显示。

6.4.1.33 isGridLinesVisible()

该方法查看是否显示工作表网格线。返回 true 为显示状态，false 为隐藏状态。

6.4.1.34 setGridLinesVisible(visible)

该方法设置是否显示工作表网格线。visible 为 true 表示显示，false 表示不显示。

6.4.1.35 isRowColumnHeadersVisible()

该方法查看是否显示行和列的标头。返回 true 为显示，false 为隐藏。

6.4.1.36 setRowColumnHeadersVisible(visible)

该方法设置是否显示行和列的标头。visible 为 true 表示显示，false 表示隐藏。

6.4.1.37 isRightToLeft()

该方法查看工作表是否是从右到左显示。返回 true 为从右到左显示状态，false 为正常显示状态。

6.4.1.38 setRightToLeft(enable)

该方法设置工作表是否从右到左显示。enable 为 true 表示从右到左显示，false 表示正常显示。

6.4.1.39 isZerosVisible()

该方法查看列如果为 0 时是否显示。返回 true 为显示，false 为隐藏。

6.4.1.40 setZerosVisible(visible)

该方法设置如果列为 0 时是否显示。visible 为 true 表示显示，false 表示隐藏。

6.4.1.41 isSelected()

该方法查看工作表的标签是否是选中状态。返回 true 为选中状态，false 为未选中状态。

6.4.1.42 setSelected(select)

该方法设置工作表的标签是否是选中状态。select 为 true 表示选中，false 表示不选中。

6.4.1.43 isRulerVisible()

该方法查看是否显示标尺。返回 true 为显示状态，false 为隐藏状态。

6.4.1.44 setRulerVisible(visible)

该方法设置是否显示标尺。visible 为 true 表示显示，false 表示隐藏。

6.4.1.45 isOutlineSymbolsVisible()

该方法查看是否显示分显示级符号。返回 true 为显示状态，false 为隐藏状态。

6.4.1.46 setOutlineSymbolsVisible(visible)

该方法设置是否显示分级显示符号。visible 为 true 表示显示，false 表示隐藏。

6.4.1.47 isWhiteSpaceVisible()

该方法查看是否显示空格。返回 true 为显示状态，false 为隐藏状态。

6.4.1.48 setWhiteSpaceVisible(visible)

该方法设置是否显示空格。visible 为 true 表示显示，false 表示隐藏。

## 6.5 Docx 类

待补充。

# 7. ui 对象[2]

全局作用域中的 ui 对象提供访问图形界面的功能，包含 ui.UiLoader 类，它能动态加载由界面设计器设计的 ui 文件。

## 7.1 ui.UiLoader 类

用于加载设计器创建的 ui 文件生成图形界面。

### 7.1.1 方法

#### 7.1.1.1 load(uifile, [parentWidget])

该方法加载 uifile 指定的 ui 文件，创建一个 QWidget 对象并返回它，可通过 parentWidget 指定该对象的父对象，若要指定为主窗口对象为父对象，则可传入 process.mainWindow 对象。

## 7.2 常量

行编辑的响应模式常量

| 常量 | 值 | 说明 |
|---|---|---|
| ui.Normal | 0 | 缺省模式，输入时显示字符串 |
| ui.NoEcho | 1 | 输入时不显示任何东西 |
| ui.Password | 2 | 输入时显示平台相关的密码掩码字符 |
| ui.PasswordEchoOnEdit | 3 | 在输入时显示字符，否则显示密码的掩码字符 |

文件对话框选项

[2] 在版本 1.1 中引入

| 常量 | 值 | 说明 |
| --- | --- | --- |
| ui. ShowDirsOnly | 1 | 文件对话框中只显示目录，缺省情况文件与目录都显示 |
| ui. DontResolveSymlinks | 2 | 文件对话框中不解析符号链接，缺省是解析 |
| ui. DontConfirmOverwrite | 4 | 若选择了一个已存在的文件不会弹出确认信息，缺省需要确认 |
| ui. DontUseNativeDialog | 16 | 不使用本地文件对话框。缺省使用本地文件对话框 |
| ui.ReadOnly | 32 | 表示只读 |
| ui. HideNameFilterDetails | 64 | 表示是否隐藏文件名称过滤器的细节 |
| ui.DontUseCustomDirectoryIcons | 128 | 不使用定制的目录图标，总是使用系统默认的 |

图片类型选项

| 常量 | 值 | 说明 |
| --- | --- | --- |
| ui.Area | 0 | 面积图 |
| ui.StackArea | 1 | 堆垒条状图 |
| ui.FillArea | 2 | 填充面积图 |
| ui.Column | 3 | 柱状图 |
| ui.ColumnPlusLabel | 4 | 柱状图+标签 |
| ui.Bar | 5 | 条状图 |
| ui.BarPlusLabel | 6 | 条状图+标签 |
| ui.StackColumn | 7 | 堆垒柱状图 |
| ui.StackBar | 8 | 堆垒条状图 |
| ui.FloatingColumn | 9 | 浮动柱状图 |
| ui.FloatingBar | 10 | 浮动条状图 |
| ui.Bubble | 11 | 气泡图 |
| ui.Line | 12 | 折线图 |
| ui.HorizontalStep | 13 | 水平阶梯图 |
| ui.VerticalStep | 14 | 垂直阶梯图 |

| | | |
|---|---|---|
| ui.Stick | 15 | 棒图 |
| ui.SplineConnected | 16 | 样条线图 |

数据数组物理意义

| 常量 | 值 | 说明 |
|---|---|---|
| ui. X | 0 | 代表 X 轴的数据 |
| ui. Y | 1 | 代表 Y 轴的数据 |
| ui. Z | 2 | 代表 Z 轴的数据 |
| ui. V | 3 | 代表标量类型的数据 |
| ui.VX | 4 | 代表矢量类型 X 分量的数据 |
| ui. VY | 5 | 代表矢量类型 Y 分量的数据 |
| ui.VZ | 6 | 代表矢量类型 Z 分量的数据 |
| ui.XError | 7 | 代表数据的 X 轴上的偏差 |
| ui.YError | 8 | 代表数据的 Y 轴上的偏差 |

## 7.3 方法

### 7.3.1 ui.critical(parent, title, text, [detailedText])

该方法弹出一个表示错误的消息框，parent 是消息框的父窗口，可以为 null，title 是消息框的标题，text 是消息框的文本提示，可选的 detailedText 是消息框的详细文本提示。

### 7.3.2 ui.getDouble(parent, title, label, [value=0], [min=-2147483647], [max=2147483647], [decimals=1])

该方法弹出一个对话框用于输入一个浮点数，parent 是对话框的父窗口，可以为 null，title 是对话框的标题，label 是对话框的输入文本提示，value 指定默认值，min 指定可接受的最小值，max 指定可接受的最大值，decimals 指定小数点的位数。

如果从对话框获取了一个值则返回该值，否则返回 undefined。

### 7.3.3 ui.getExistingDirectory([parent=null], [caption=’’], [dir=’’], [options=ui.ShowDirsOnly])

该方法弹出一个对话框获取用户选择的目录并返回。parent 是对话框的父窗口，默认为 null，caption 是对话框标题文本，默认为空字符串，dir 是对话框定位的初始目录，默认为空字符串，options 目录对话框的选项，默认是只显示目录常数。

### 7.3.4 ui.getInt(parent, title, label, [value=0], [min=-2147483647], [max=2147483647], [step=1]

该方法弹出一个对话框用于输入一个整数，parent 是对话框的父窗口，可以为 null，title 是对话框的标题，label 是对话框的输入文本提示，value 指定默认值，min 指定可接受的最小值，max 指定可接受的最大值，step 指定增量。

如果从对话框获取了一个值则返回该值，否则返回 undefined。

### 7.3.5 ui.getItem(parent, title, label, items, [current=0], [editable=true])

该方法弹出一个对话框让用户从一个字符串数组中选择一项，parent 是对话框的父窗口，可以为 null，title 是对话框的标题，label 是对话框的输入文本提示，items 是字符串数组，current 指定默认选择的字符串数组索引，editable 若为 true 表示用户可编辑字符串数组中的字符，否则不可编辑。

如果选择了一个列表项则返回该列表项的字符串，否则返回 undefined。

### 7.3.6 ui.getMultiLineText(parent, title, label, [text=’’])

该方法弹出一个对话框让用户输入多行文本，parent 是对话框的父窗口，可以为 null，title 是对话框的标题，label 是对话框的输入文本提示，text 是默认提供的文本。

如果从对话框获取了文本则返回该值，否则返回 undefined。

### 7.3.7 ui.getOpenFileName([parent=null], [caption=''], [dir=''], [filter=''], [options=0])

该方法弹出文件对话框用于选择一个已存在的文件，parent 是对话框的父窗口，可以为 null，caption 设置对话框的标题，dir 用于设置对话框的初始路径，如可设置为“E:/data”，filter 设置文件对话框的过滤器，如可设置为"Images (*.png *.xpm *.jpg);;Text files (*.txt);;XML files (*.xml)"，其中使用;;分割多个过滤器，options 是文件对话框的常数。

如果用户选择了一个文件并点击对话框的确定按钮则返回该文件的路径，否则返回 undefined。

### 7.3.8 ui.getOpenFileNames([parent=null], [caption=''], [dir=''], [filter=''], [options=0])

该方法弹出文件对话框用于选择一个或多个已存在的文件，parent 是对话框的父窗口，可以为 null，caption 设置对话框的标题，dir 用于设置对话框的初始路径，如可设置为“E:/data”，filter 设置文件对话框的过滤器，如可设置为"Images (*.png *.xpm *.jpg);;Text files (*.txt);;XML files (*.xml)"，其中使用;;分割多个过滤器，options 是文件对话框的常数。

如果用户选择了一个或多个文件并点击对话框的确定按钮则返回该这些文件路径的数组，否则返回 undefined。

### 7.3.9 ui.getSaveFileName([parent=null], [caption=''], [dir=''], [filter=''], [options=0])

该方法弹出文件对话框用于选择一个文件，该文件可以不存在，parent 是对话框的父窗口，可以为 null，caption 设置对话框的标题，dir 用于设置对话框的初始路径，如可设置为“E:/data”，filter 设置文件对话框的过滤器，如可设置为"Images (*.png *.xpm

*.jpg);;Text files (*.txt);;XML files (*.xml)"，其中使用;;分割多个过滤器，options 是文件对话框的常数。

如果用户选择了一个文件并点击对话框的确定按钮则返回该文件的路径，否则返回 undefined。

### 7.3.10 ui.getText(parent, title, label, [mode=ui.Normal], [text=’’])

该方法弹出文件对话框用于选择输入字符串，parent 是对话框的父窗口，可以为 null，title 设置对话框的标题，label 是对话框的输入文本提示，mode 是 ui 对象中的行编辑响应模式常量，text 是默认的字符串。

如果用户输入了字符串并点击对话框的确定按钮则返回该字符串，否则返回 undefined。

### 7.3.11 ui.information(parent, title, text, [detailedText])

该方法弹出一个表示信息的消息框，parent 是消息框的父窗口，可以为 null，title 是消息框的标题，text 是消息框的文本提示，可选的 detailedText 是消息框的详细文本提示。

### 7.3.12 ui.question(parent, title, text, [detailedText])

该方法弹出一个询问用户的对话框，parent 是对话框的父窗口，可以为 null，title 是对话框的标题，text 是对话框的文本提示，可选的 detailedText 是对话框的详细文本提示。

如果用户对问题选择是则返回 true，否则返回 false。

### 7.3.13 ui.saveGraph(data, path, [options])

该方法绘制 data 数值数值中的曲线图，然后保存到 path 表示的文件中。参数 data 可以是一维或二维数组，如一维数组[2,3,7]将绘制一条以 0,1,2…为 X 轴的曲线图，二

维数组[[2,3,6], [5,2,3],[23,2,22]]将绘制两条曲线图，默认将以第一个数组为 X 轴，参数 path 是文件的路径，如果没有提供文件后缀名，将自动设置为 png。方法将自动判断文件类型，可以生成 png，jpg，bmp，pdf，svg 等文件类型。

可选参数 options 对象可设置图片的标题，图片类型，图片尺寸，各条数据（如果是二维数组）的物理意义。描述如下表 2 options 对象说明。

表 2 options 对象说明

| 属性 | 键 | 类型 | 描述 |
|---|---|---|---|
| 图片标题 | title | 字符串 | 图片默认有一个标题，也可通过该键指定 |
| 图片类型 | type | 常量 | 按图片类型常量值给定，不提供则是折线图 |
| 图片尺寸 | size | 字符串 | 按图片的宽度与高度的像素数量给定，由于内部换算误差，会有一点出入，不提供则是 402*268。值的格式是：宽度方向像素*高度方向像素。如“1024*768”，表示宽度方向有 1024 像素，高度方向有 768 像素 |
| 数据的物理意义 | significance | 常量 | 一维数组无需提供该属性。按二维数组中的数据顺序给定各个数据的物理意义 |

### 7.3.14 ui.uiInteractive(str|function)

该方法用于在非主线程（GUI 线程）中利用 JavaScript 代码创建图形界面，并与之交互。方法参数可以是一段 JavaScript 代码，或是一个 JavaScript 匿名函数。

如果脚本代码的最后一行求值结果是 QWidget 类型，那么返回该对象。

### 7.3.15 ui.warning(parent, title, text, [detailedText])

该方法弹出一个表示警告的消息框，parent 是消息框的父窗口，可以为 null，title 是消息框的标题，text 是消息框的文本提示，可选的 detailedText 是消息框的详细文本提示。

# 8. sql 对象

全局作用域中的 sql 对象提供访问各种关系数据库系统的入口，通过该对象创建或删除数据库对象及连接。与数据库访问有关的其它四个类 Database、SqlQuery、SqlRecord、SqlError，它们都不可直接通过 new 操作符创建，而是间接地通过调用其它对象的方法创建，如 Database 对象是调用 sql 对象中的 addDatabase()方法创建的。

## 8.1 sql.addDatabase(type, connectionName)

该方法使用数据库驱动器类型 type 和连接名称 connectionName 添加一个数据库连接对象（DatabaseObject 类的对象）到数据库连接池中，并返回它。如果已经存在一个同名的数据库连接，且该连接是通过 sql 对象创建的，那么将先删除该连接，再重新创建一个新的连接。如果待创建的同名连接不是通过 sql 对象创建的，那么抛出异常提示并返回 null 值。当驱动器类型 type 不可用时，则抛出异常并返回 null 值。

在使用返回的数据库连接对象之前，必须对其初始化，设置数据库名称、用户名、密码、主机名称以及端口号等属性，当驱动器类型是 Sqlite 数据库时，只需设置数据库名称，也即数据库文件名。

## 8.2 sql.cloneDatabase(other, connectionName)

该方法使用数据库连接名称 connectionName 克隆已存在的数据库对象 other，并返回克隆的数据库连接对象。克隆的数据库对象都使用 other 的设置，如数据库名，主机名等。新建的数据库连接并没有打开，在使用该新连接之前请调用数据库连接对象的 open()方法打开它。

## 8.3 sql.contains(connectionName)

该方法测试连接 connectionName 是否通过 sql 对象创建了，是则返回 true，否则返回 false。

## 8.4 sql.globalContains(connectionName)

该方法测试当前应用程序范围内连接 connectionName 是否存在，是在返回 true，否则返回 false。连接可以不是通过 sql 对象创建。

## 8.5 sql.isDriverAvailable(name)

该方法测试驱动器 name 是否可用，是则返回 true，否则返回 false。可使用 drivers 属性获得所有可使用的驱动器名称列表。

## 8.6 sql.removeDatabase(connectionName)

该方法从数据库连接池中移除 connectionName 连接。

## 8.7 sql.database(connectionName)

该方法返回使用 sql 对象创建的已存在的 connectionName 连接，若不存在则返回 null 值。

## 8.8 sql.autoThrow

该属性是一个可读写的属性，可读取或设置 sql 对象在出现错误时是否抛出异常，如果设置为 true 表示抛出异常，设置为 false 则不抛出异常。

## 8.9 sql.connectionNames

该属性是一个只读属性，返回使用 sql 对象创建的所有数据库连接名称字符串数组。

## 8.10 sql.globalConnectionNames

该属性是一个只读属性，返回当前应用程序范围内所有数据库连接名称字符串数组。连接可以不是通过 sql 对象创建。

## 8.11 sql.drivers

该属性是一个只读属性，返回当前应用程序支持的所有驱动器类型字符串数组。

# 9. DatabaseObject 类

类 DatabaseObject 代表一个数据库连接，它通过与数据库的连接提供访问数据库的接口。不可使用 new 语句创建此类对象，必须调用 sql 对象中的 addDatabase()方法创建。

## 9.1 方法

### 9.1.1 close()

该方法关闭数据库连接，释放分配的连接资源，并使得与此连接相关的所有 SqlQuery 对象都无效。

### 9.1.2 commit()

该方法在数据库中提交一个事务，前提是驱动器支持事务并且已经使用 transaction()开启了一个事务。如果事务操作成功则返回 true，否则返回 false。

### 9.1.3 exec([query])

该方法在数据库上执行 query 中的 SQL 语句，并返回一个 SqlQuery 对象。如果 query 是空的 SQL 语句，则返回一个空的、无效的 SqlQuery 对象。

### 9.1.4 open()

该方法使用当前连接值打开数据库连接，如果成功返回 true，否则返回 false。

### 9.1.5 open(user, password)

该方法使用给定的用户名 user 与密码 password 打开数据库连接，如果成功返回

true，否则返回 false。该方法不保存给定的密码，打开数据库连接之后该密码即被舍弃了。

### 9.1.6 query([query])

该方法使用当前数据库连接以及 query 中的 SQL 语句创建一个 SqlQuery 查询对象，并返回它。如果 query 是空的 SQL 语句，则返回一个空的、无效的 SqlQuery 对象。

### 9.1.7 record(tablename)

该方法返回当前数据库连接中填充了 tablename 表或视图中所有字段名称的 SqlRecord 对象。在记录中出现的字段顺序是未定义的。如果 tablename 并不存在，那么返回一个空的对象。

### 9.1.8 rollback()

该方法回滚数据库的一个事务，前提是驱动器支持事务，并且已经使用 transaction() 开启了一个事务。如果操作成功返回 true，否则返回 false。

### 9.1.9 tables(type)

该方法根据类型 type 字符串参数返回数据库中的表、系统表和视图等的名称列表。

| Type 值 | 描述 |
|---|---|
| Tables | 所有对用户可见的表 |
| SystemTables | 被数据库使用的内部表 |
| Views | 所有对用户可见的视图 |
| AllTables | 上面所列的全部 |

### 9.1.10 transaction()

该方法开启数据库的一个事务，如果数据库驱动器支持事务。如果操作成功返回 true，否则返回 false。

## 9.2 属性

### 9.2.1 autoThrow

该属性是一个可读写的属性，可读取或设置 DatabaseObject 对象在出现错误时是否抛出异常，如果设置为 true 表示抛出异常，设置为 false 则不抛出异常。

### 9.2.2 connectionName

该属性是一个只读属性，字符串类型，返回数据库的连接名称，有可能为空。注意：连接名称不是数据库名称。

### 9.2.3 connectOptions

该属性是一个可读写属性，字符串类型，设置特定数据库的连接选项。必须在连接打开之前设置，否则无效。也可以使用 close()方法关闭当前连接，然后设置连接选项，再调用 open()方法重新打开连接。选项字符串是以分号分隔的选项名称或选项=值的值对。选项字符串依赖使用的数据库类型：

| ODBC | MySQL | PostgreSQL |
| --- | --- | --- |
| ▪ SQL_ATTR_ACCESS_MODE<br>▪ SQL_ATTR_LOGIN_TIMEOUT<br>▪ SQL_ATTR_CONNECTION_TIMEOUT<br>▪ SQL_ATTR_CURRENT_CATALOG<br>▪ SQL_ATTR_METADATA_ID<br>▪ SQL_ATTR_PACKET_SIZE<br>▪ SQL_ATTR_TRACEFILE<br>▪ SQL_ATTR_TRACE<br>▪ SQL_ATTR_CONNECTION_POOLING<br>▪ SQL_ATTR_ODBC_VERSION | ▪ CLIENT_COMPRESS<br>▪ CLIENT_FOUND_ROWS<br>▪ CLIENT_IGNORE_SPACE<br>▪ CLIENT_SSL<br>▪ CLIENT_ODBC<br>▪ CLIENT_NO_SCHEMA<br>▪ CLIENT_INTERACTIVE<br>▪ UNIX_SOCKET<br>▪ MYSQL_OPT_RECONNECT | ▪ connect_timeout<br>▪ options<br>▪ tty<br>▪ requiressl<br>▪ service |
| **DB2** | **OCI** | **TDS** |
| ▪ SQL_ATTR_ACCESS_MODE<br>▪ SQL_ATTR_LOGIN_TIMEOUT | ▪ OCI_ATTR_PREFETCH_ROWS<br>▪ OCI_ATTR_PREFETCH_MEMORY | *none* |
| **SQLite** | **Interbase** | |
| ▪ QSQLITE_BUSY_TIMEOUT<br>▪ QSQLITE_OPEN_READONLY<br>▪ QSQLITE_OPEN_URI<br>▪ QSQLITE_ENABLE_SHARED_CACHE | ▪ ISC_DPB_LC_CTYPE<br>▪ ISC_DPB_SQL_ROLE_NAME | |

例子：

// MySQL 连接，使用 SSL 连接到服务器

db.connectionOptions = “CLIENT_SSL=1;CLIENT_IGNORE_SPACE=1”;

if (!db.open())

db.connectionOptions = “”;

// PostgreSQL 连接 使用 SSL 连接到服务器

db.connectionOptions = “requiressl=1”;

if (!db.open())

db.connectionOptions = “”;

// ODBC 连接

db.connectionOptions =

“SQL_ATTR_ACCESS_MODE=SQL_MODE_READ_ONLY;SQL_ATTR_TRACE=SQL

_OPT_TRACE_ON”;

if (!db.open())

db.connectionOptions = “”;

关于更多的不同选项参考具体数据库的客户端库文档。

### 9.2.4 databaseName

该属性是一个可读写属性，字符串类型，设置连接的数据库名称。必须在连接打开前设置，否则无效。或者可以使用 close()方法关闭连接，设置数据库名称，然后调用 open()方法重新打开连接。注意：数据库名称不是连接名称。连接名称必须在调用 sql.addDatabase()方法时传入。

对于 Oracle 数据库，数据库名称是 TNS 服务名称。

对于 ODBC 驱动器的数据库，数据库名称可以是一个 DSN，一个 DSN 文件名（此时文件必须有一个.dsn 的扩展名），或一个连接字符串。

例子：Microsoft Access 用户可以用下面的连接字符串直接打开一个.mdb 文件，而不必在 ODBC 管理器中创建一个 DSN 入口。

var db = sql.addDatabase(“QODBC”, “abc”);

db.databaseName = “DRIVER={Microsoft Access Driver (*.mdb)};FIL={MS Access};DBQ=myaccessfile.mdb”;

if (db.open()) {

// success

}

### 9.2.5 driverName

该属性是一个只读属性，字符串类型，返回数据库的驱动器名称。

### 9.2.6 hostName

该属性是一个可读写属性，字符串类型，表示数据库连接的主机名称，可能为空。主机名必须在连接打开之前设置才有效。或者可以使用 close()方法关闭连接，设置主机名，然后再调用 open()方法打开。

### 9.2.7 isOpen

该属性是一个只读属性，返回 true 表示数据库连接当前已经打开，返回 false 表示未打开。

### 9.2.8 isOpenError

该属性是一个只读属性，返回 true 表示打开数据库连接是有错误，返回 false 表示无错误。使用 lastError 属性可以获取错误信息。

### 9.2.9 isValid

该属性是一个只读属性，如果数据库对象有一个有效的驱动器则返回 true，否则返回 false。

### 9.2.10 lastError

该属性是一个只读属性，返回一个 SqlError 对象，包含了数据库操作中最近一次发生的错误信息。

### 9.2.11 password

该属性是一个可读写属性，字符串类型，表示数据库连接的密码。必须在连接打开之前设置才有效，也可以使用 close()方法关闭连接，设置密码，然后调用 open()方法重新打开它。

### 9.2.12 port

该属性是一个可读写属性，整数类型，表示数据库连接的端口号。必须在连接打开之前设置才有效，也可以使用 close()方法关闭连接，设置端口号，然后调用 open()方法重新打开它。

### 9.2.13 userName

该属性是一个可读写属性，字符串类型，表示数据库连接的用户名。必须在连接打开之前设置才有效，也可以使用 close()方法关闭连接，设置用户名，然后调用 open()方法重新打开它。

# 10. SqlQuery 类

类 SqlQuery 提供了执行和操作 SQL 语句的工具，它通过 SQL 查询封装了在数据库上执行创建、遍历和获取数据的功能，它可用于执行 DML（数据操作语言）语句，例如 SELECT，INSERT，UPDATE 和 DELETE，以及 DDL（数据定义语言）语句，例如 CREATE TABLE，也可用于执行特定数据库的非 SQL 标准命令（如 PostgreSQL 数据库中的 SET DATASTYLE=ISO）。

使用 SqlQuery 查询对象成功执行 SQL 语句后将设置对象的状态是活动的，也即 isActive 属性的值为 true，否则它的状态是非活动的。无论哪一种情况，当执行一条新的 SQL 语句时，SqlQuery 对象将指向一条无效的记录。活动状态的查询对象在数据获取前必须被导航到一条有效的记录上（isValid 属性的值是 true）。

对有些数据库而言，如果活动查询是以执行 SELECT 语句而得，当在数据库对象上执行 commit()或 rollback()方法时将失败，细节参考 isActive 属性。

可以使用如下方法遍历各条记录：

- next()

- previous()
- first()
- last()
- seek()

上述方法可以在查询对象返回的记录中向前、向后或任意遍历。如果只需向前遍历所有记录（通过调用 next()），可通过设置属性 isForwardOnly = true 达成，这将在某些数据库中节省大量内存并改善性能。一旦活动查询对象指向了一个有效的记录，即可调用 value()方法获取数据。

例子：

```
var q = db.query("SELECT country FROM artist");
while (q.next()) {
    var country = q.value(0);
    log(country);
}
```

为了访问从查询中返回的数据，可以使用 value(int)。整数索引是使用 SELECT 语句中字段对应的位置，从 0 开始。这将不建议使用 SELECT *语句查询数据，因为将导致返回字段的顺序不确定。

SqlQuery 支持预先准备好的查询（使用 prepare 方法）以及可以把参数绑定到占位符上。有些数据库不支持这些特性，在 SqlQuery 中模仿了这些功能，使得与不同数据库的交互接口是一致的。

Oracle 数据库使用冒号语法，如:name，表示占位符，ODBC 简单的使用?字符，SqlQuery 支持这两种语法，但不要在一条 SQL 语句中混用这两种语法。可以调用 boundValues()方法获得所有的绑定字段。

下面是四个不同的绑定方法例子。

命名占位符的命名绑定

```
query.prepare("INSERT INTO person (id, forename, surname) "
              "VALUES (:id, :forename, :surname)");
query.bindValue(":id", 1001);
query.bindValue(":forename", "Bart");
query.bindValue(":surname", "Simpson");
query.exec();
```

命名占位符的位置绑定

```
query.prepare("INSERT INTO person (id, forename, surname) "
              "VALUES (:id, :forename, :surname)");
query.bindValue(0, 1001);
query.bindValue(1, "Bart");
query.bindValue(2, "Simpson");
query.exec();
```

位置占位符的绑定方法 **1**

```
query.prepare("INSERT INTO person (id, forename, surname) "
              "VALUES (?, ?, ?)");
query.bindValue(0, 1001);
query.bindValue(1, "Bart");
query.bindValue(2, "Simpson");
query.exec();
```

位置占位符的绑定方法 **2**

```
query.prepare("INSERT INTO person (id, forename, surname) "
              "VALUES (?, ?, ?)");
```

```
query.addBindValue(1001);
query.addBindValue("Bart");
query.addBindValue("Simpson");
query.exec();
```

## 10.1 方法

### 10.1.1 addBindValue(val)

该方法用于在位置占位符绑定时添加值 val 到各个值的列表中。该方法的调用顺序决定了该值绑定到哪一个占位符上。

### 10.1.2 bindValue(placeholder | pos, val)

该方法用于在字符串占位符或位置占位符上绑定 val 值。

### 10.1.3 boundValue(placeholder | pos)

该方法用于返回字符串占位符或位置占位符上绑定的值。

### 10.1.4 boundValues()

该方法以对象的方式返回所有绑定的值。

### 10.1.5 clear()

该方法用于清除查询对象的结果集，释放查询对象占用的任何资源。设置查询为不活动状态。

### 10.1.6 exec([query])

该方法在不带参数时执行先前使用 prepare()方法准备的 SQL 查询。如果查询成功执行则返回 true，否则返回 false。

如果指定了 query 参数，则执行 query 中的 SQL 语句，如果查询成功执行则返回

true 并设置查询对象状态为活动的，否则返回 false。

查询执行后将指向一个无效的记录，在获取数据之前必须先导航到一个有效的记录上，如调用 next()方法。

对于 SQLite 数据库，query 字符串中一次只能包含一条语句，如果超过一条，那么就返回 false。

### 10.1.7 execBatch(mode)

该方法用于以批处理的方式执行先前使用 prepare()方法准备的 SQL 查询。所有的绑定参数必须是数组。如果数据库不支持这种批处理执行，那么驱动器将使用 exec()方法模仿这种方式。如果查询成功执行返回 true，否则返回 false。

mode 只有两种值：0 或 1，表示被绑定的值如何解释。如果 mode=0，那么数组中每个值解释为表中新行的一个值。当 mode=1 时，是专为 Oracle 数据库准备的，数组中的值将被解释为 Oracle 中的 Table 类型的字段的值。

例子：

```
q.prepare("insert into myTable values (?, ?)");


var ints = [ 1, 2, 3, 4];
q.addBindValue(ints);
var names = ["Harald", "Boris", "Trond", null];
q.addBindValue(names);


if    (!q.execBatch())
      log(q.lastError());
```

上例插入了四行至表 myTable 中：

1 Harald

2 Boris

3 Trond

4 NULL

注意：每个绑定的数组应包含相同的元素个数，而且元素的类型必须一致，null 除外。

### 10.1.8 finish()

该方法用于提示数据库驱动器当前查询对象不再有更多的数据需要获取了，除非查询重新执行。一般不需要调用此方法，但为了释放资源（如锁和光标）同时又想在以后重用该查询对象，则是有帮助的。

调用此方法后，查询进入非活动状态，绑定值仍旧保留。

### 10.1.9 first()

该方法用于获取结果中的第一条记录，如果第一条记录有效，将指向该记录。在调用此方法之前数据集结果必须在活动状态并且 isSelect 的值是 true，否则该方法不做任何操作并返回 false。如果成功则返回 true，不成功则查询设置了一个无效的位置并返回 false。

### 10.1.10isNull(str | int)

该方法用于接收一个字符串（字段名称）或基于 0 为索引的整数（字段位置），当查询状态是非活动，指向一个无效的记录，无此字段，或字段值是 null 时返回 true，其它情况返回 false。

注意，对某些数据库，该方法将返回不是很精确的信息，除非对获得的数据执行某些操作。

### 10.1.11last()

该方法用于获取结果中的最后一条记录，如果最后一条记录有效，将指向该记录。在调用此方法之前数据集结果必须在活动状态并且 isSelect 的值是 true，否则该方法不做任何操作并返回 false。如果成功则返回 true，不成功则查询设置了一个无效的位置并返回 false。

### 10.1.12lastInsertId()

该方法用于返回数据库中最近插入行的对象 ID 值，前提是数据库必须支持这个特性。如果查询没有插入任何值或者数据库不报告该 ID 则返回 undefined，如果插入了多行，那么返回的结果不确定。

对于 MySQL 数据库，将返回行的自动递增字段。

注意：如果该方法要在 PostgreSQL 下工作，那么数据库中的表必须包含 OID，默认是不创建的，因此需确认 default_with_oids 配置参数。

### 10.1.13next()

该方法在结果中获取下一条记录，如果有，指向获取的该记录。调用此方法前查询必须处于活动状态并且 isSelect 返回 true，否则是无效操作并返回 false。

将应用以下三种规则：

- 如果当前定位在第一条记录之前，当查询刚刚执行完毕后处于此状态，那么调用此方法后将指向第一条记录。
- 如果当前定位在最后一条记录之后，那么不会发生什么改变，返回 false。
- 如果当前定位在结果的中间记录，那么将指向下一条记录。

如果不能获取记录，那么将指向最后一条记录之后并返回 false。如果成功获得了记录，那么返回 true。

### 10.1.14nextResult()

该方法用于丢弃当前结果集，导航到下一个结果集，如果有。某些数据库对于存储过程或 SQL 批处理（查询字符串包含多条语句）可以返回多个结果集。如果在执行查询后多条结果集可用，该方法可用于导航到下一个结果集。

如果新的结果集可用，该方法返回 true。查询将重新定位到新结果集的无效记录上，在获取数据之前必须导航到一个有效的记录上。如果新的结果集不可用，则返回 false，查询设置为非活动状态。在任何情况下老的结果集将被丢弃。

### 10.1.15prepare(query)

该方法用于为 SQL 语句 query 的执行做准备。如果 query 成功准备了则返回 true，否则返回 false。在 query 中可以包含用于值绑定的占位符。Oracle 中的分号样式（如:name）以及 ODBC 中的?样式都是支持的；但不能把这两种样式在一条 SQL 语句中混用。

移植问题：某些数据库选择当第一次执行语句时才对 SQL 语句做准备工作，在这种情况下，如果准备了一个语法错误的语句，那么即使这里返回了 true，在调用 exec() 方法执行时也将失败。

对于 SQLite，查询字符串每次只能包含一条语句，如果多于一条，那么返回 false。

例子：

```
query.prepare("INSERT INTO person (id, forename, surname) "
              "VALUES (:id, :forename, :surname)");
query.bindValue(":id", 1001);
query.bindValue(":forename", "Bart");
query.bindValue(":surname", "Simpson");
query.exec();
```

### 10.1.16previous()

该方法在结果中获取上一条记录，如果有，指向获取的该记录。调用此方法前查询必须处于活动状态并且 isSelect 返回 true，否则是无效操作并返回 false。

将应用以下三种规则：

- 如果当前定位在第一条记录之前，那么不会发生什么改变，返回 false。
- 如果当前定位在最后一条记录之后， 那么将指向最后一条记录。
- 如果当前定位在结果的中间记录，那么将指向上一条记录。

如果不能获取记录，那么将指向第一条记录之前并返回 false。如果成功获得了记录，那么返回 true。

### 10.1.17record()

该方法返回包含了当前查询字段信息的 SqlRecord 对象。如果查询指向一个有效的行（isValid 返回 true），那么 SqlRecord 对象中填充了该行的值。当查询处于非活动状态（isActive 返回 false）将返回一个空的记录 SqlRecord 对象。

从一个查询结果中获取值，应该调用基于索引的 value()方法，它的效率最高。

在下面的例子中，执行了一条 SELECT * FROM 语句，因为列的顺序没有定义，所以使用 SqlRecord.indexOf()获得列的索引。

```
q.exec("select * from employees");
var rec = q.record();
log("Number of columns: ", rec.count());

var nameCol = rec.indexOf("name"); // index of the field "name"
while (q.next())
    log(q.value(nameCol)); // output all names
```

### 10.1.18seek(i, relative)

该方法获取位置 i 处的记录，如果有，定位到这条记录上。第一条记录的位置是 0，调用此方法前查询必须处于活动状态并且 isSelect 返回 true。

如果 relative=false，应用下面的规则：

- 如果 i 是负数，那么将定位在第一条记录之前并返回 false。
- 否则，将尝试移动到位置 i 处。如果不可获得 i 处的记录，那么将移动到最后一条记录之后并返回 false。如果成功获得记录，则返回 true。

如果 relative=true，应用下面的规则：

- 如果当前定位在第一条记录或第一条记录之前，并且 i 是负数，那么不做改变，返回 false。
- 如果当前定位在最后一条记录之后，并且 i 是正数，那么不做改变，返回 false。
- 如果当前定位在中间某个位置，并且根据相对偏移量 i 移动后小于 0，那么将定位于第一条记录之前，返回 false。
- 否则，尝试往前移动 i 条记录（或者如果 i 是负数，往后移动 i 条记录）。如果在 i 处的记录无法获取，那么在 i>=0 时将指向最后一条记录之后，若是负数则指向第一条记录之前，返回 false。如果成功获得记录，返回 true。

### 10.1.19value(str | int)

该方法返回当前记录中字段名称（若参数是字符串），字段索引（若参数是整数）的值。

字段索引是根据 select 语句中的字段基于 0 开始的整数索引，当使用 select *时无法获得正确的索引值。若字段索引或名称不存在，查询处于非活动状态，或查询定位到一个无效的记录上将返回 undefined。

## 10.2 属性

### 10.2.1 at

该属性是只读属性，整数类型，返回查询对象当前内部位置。第一条记录是位置0。如果当前位置无效，属性值将是-1（表示在第一条记录之前），或-2（表示在第二条记录之后）。

### 10.2.2 autoThrow

该属性是一个可读写的属性，可读取或设置查询对象在出现错误时是否抛出异常，如果设置为 true 表示抛出异常，设置为 false 则不抛出异常。

### 10.2.3 executedQuery

该属性是一个只读属性，字符串类型，返回成功执行的最后一个查询字符串。

### 10.2.4 isActive

该属性是一个只读属性，如果查询对象是活动的返回 true，否则返回 false。一个活动的查询是指查询已经成功执行，但还没有结束。当结束了一个活动查询时，可以通过调用 finish()或 clear()函数使其置于非活动状态。

注意：对有些数据库而言，如果活动查询是以执行 SELECT 语句而得，当在数据库对象上执行 commit()或 rollback()方法时将失败，此时在执行 commit()或 rollback()方法之前，必须通过上面所述方法使其置于非活动状态。

### 10.2.5 isForwardOnly

该属性是一个可读写属性，获取或设置只能向前移动的模式，如果是 true 表示 next()和 seek()方法只能传入正值，向前遍历记录。

只能向前移动的模式对某些数据库能提高性能，因此此时结果不必缓存了。因为这个原因，在查询对象准备（prepare）或执行之前可设置该属性为 true。该属性默认

是 false。

注意：在查询执行之后设置该属性在好的情况下导致不可预期的结果，坏的情况下将崩溃。

### 10.2.6 isSelect

该属性是一个只读属性，如果当前查询是一个 SELECT 语句返回 true，否则返回 false。

### 10.2.7 isValid

该属性是一个只读属性，如果当前查询指向一个有效记录则返回 true，否则返回 false。

### 10.2.8 lastError

该属性是一个只读属性，返回当前查询中最后一个错误对象 SqlError，如果无错误，则返回 null 值。

### 10.2.9 lastQuery

该属性是一个只读属性，字符串类型，返回当前正使用的查询文本，如果没有查询文本则返回空字符串。

### 10.2.10numRowsAffected

该属性是一个只读属性，整数，返回 SQL 语句执行后影响的行的数量，如果不能确定则返回-1。如果查询是非活动的，那么返回-1。

注意：对于 SELECT 语句，该值未定义，使用 size 属性。

### 10.2.11size

该属性是一个只读属性，整数，返回结果（返回行的数量）的数量，如果不能确定或数据库不支持则返回-1。对于非 SELECT 查询（isSelect 值为 false）将返回-1。如

果查询是非活动的（isActive 值为 false）也返回-1。

获取非 SELECT 语句影响行的数量使用 numRowsAffected。

# 11. SqlRecord 类

类 SqlRecord 封装了一条数据库记录（通常是数据库中表或视图的一行）。SqlRecord 对象可通过调用 SqlQuery 对象的 record()方法获得。

## 11.1 方法

### 11.1.1 contains(name)

该方法判断 name 是否在记录中，若是则返回 true，否则返回 false。

### 11.1.2 fieldName(i)

该方法返回记录中位置 i 处的字段名称。如果此处无字段则返回空字符串。

### 11.1.3 indexOf(name)

该方法返回记录中名称是 name 的字段所在位置，如果不能找到则返回-1。字段名称忽略大小写，如果有多个字段匹配，则返回第一个字段的位置。

### 11.1.4 isNull(str | int)

该方法判断字段名称（字符串）或字段位置（整数）是否是 null，如果是则返回 true，否则返回 false。

### 11.1.5 value(str | int)

该方法返回字段名称（字符串）或字段位置（整数）表示的字段的值。如果不能返回有效的值，则在 autoThrow=true 时抛出一个异常对象，否则返回一个 undefined。

## 11.2 属性

### 11.2.1 autoThrow

该属性是一个可读写的属性，可读取或设置记录对象在出现错误时是否抛出异常，如果设置为 true 表示抛出异常，设置为 false 则不抛出异常。

### 11.2.2 count

该属性是一个只读属性，整数类型，返回记录中字段的数量。

### 11.2.3 isEmpty

该属性是一个只读属性，判断记录是否为空，若是则返回 true，否则返回 false。

# 12. SqlError 类

类 SqlError 提供了数据库的错误信息。

## 12.1 属性

### 12.1.1 databaseText

该属性是一个只读属性，字符串类型，返回由数据库报告的错误文本。

### 12.1.2 driverText

该属性是一个只读属性，字符串类型，返回由驱动器报告的错误文本。

### 12.1.3 text

该属性是一个只读属性，字符串类型，它把 databaseText 与 driverText 两个字符串连接起来返回。

### 12.1.4 type

该属性是一个只读属性，字符串类型，返回错误的类型，一共有五种类型，如下

表所示。

| 类型 | 描述 |
| --- | --- |
| NoError | 没有错误发生 |
| ConnectionError | 连接错误 |
| StatementError | SQL 语句语法错误 |
| TransactionError | 事务失败错误 |
| UnknownError | 未知错误 |

# 13. ChildProcess 类

类 ChildProcess 为 JavaScript 解释器提供了创建子进程并与之交互的能力，通过 JavaScript 的 new 运算符创建并初始化一个新对象：

```
var cp = new ChildProcess();
```

创建了子进程对象后，可使用该对象的方法，通过它调用其它应用程序，从而启动一个新的子进程，子进程启动后，可以通过标准输入、输出流与之通信。这个过程比较复杂，有可能子进程无法启动，或者产生某些错误，父进程（SimCube 中的应用程序）有可能关心子进程什么时候启动完毕，什么时候子进程退出了，什么时候可以读取子进程的输出流中的数据等等。ChildProcess 对象可以对外发射信号，通知外界状态的变化、是否启动了子进程，子进程是否关闭。如果用户关心某个信号，那么可以通过编写一个 JavaScript 函数连接到该信号上，当信号发射时，该函数即可得到调用。

## 13.1 常量

表示子进程状态（ProcessState）的常量

| 常量 | 值 | 说明 |
| --- | --- | --- |
| ChildProcess.NotRunning | 0 | 子进程不在运行状态 |

| | | |
|---|---|---|
| ChildProcess.Starting | 1 | 子进程正在启动，但是程序还没有被调用起来 |
| ChildProcess.Running | 2 | 子进程正在运行中，读写操作已经准备好了 |

表示子进程错误（ProcessError）的常量

| 属性 | 值 | 说明 |
|---|---|---|
| ChildProcess.FailedToStart | 0 | 子进程启动失败。可能是找不到被调用的程序，或者没有足够的权限运行该程序 |
| ChildProcess.Crashed | 1 | 子进程在成功启动后崩溃 |
| ChildProcess.Timedout | 2 | 子进程在 waitForStarted 或 waitForFinished 时超时了 |
| ChildProcess.WriteError | 3 | 往子进程写入时出错，可能是子进程没在运行中，或是关闭了它的输入通道 |
| ChildProcess.ReadError | 4 | 从子进程读取时出错，可能是子进程没在运行中 |
| ChildProcess.UnknownError | 5 | 不能识别的错误发生了，这是 error()方法的默认返回值 |

表示子进程退出状态（ExitStatus）的常量：

| 常量 | 值 | 说明 |
|---|---|---|
| ChildProcess.NormalExit | 0 | 子进程正常退出 |
| ChildProcess.CrashExit | 1 | 子进程崩溃退出 |

## 13.2 信号

ChildProcess 类共有 7 个信号可以发射。

| 信号 | 说明 |
|---|---|
| started() | 当子进程启动后发射，此时 state() 方法返回 ChildProcess.Running 属性值 |
| error(err) | 当子进程发生错误时发射，err 是 ProcessError 属性中的一个值 |

| | |
|---|---|
| finished(exitCode, exitStatus) | 当子进程退出时发射，exitCode 是子进程的退出码（只在正常退出时有效），exitStatus 是 ExitStatus 属性中的一个值。子进程退出后，它的缓冲区仍然是完好的，可以从中读取子进程退出前写入的数据 |
| childProcessFinished() | 当子进程创建的子进程退出时发射，它适用于子进程创建了它的子进程之后关闭了自己，只留下它的子进程在运行，此时可以通过该信号判断子进程的子进程关闭的时刻 |
| stateChanged(newState) | 当子进程的状态发生改变时发射，newState 是子进程的新状态，它是 ProcessState 属性中的一个值 |
| readyReadStandardError() | 当子进程的标准错误通道（stderr）中有新的数据可利用时发射 |
| readyReadStandardOutput() | 当子进程的标准输出通道（stdout）中有新的数据可利用时发射 |

ChildProcess 的对象发射信号后，根据情况可以响应也可不加理会，当要响应发射的信号后，则需编写一个函数连接到该信号，函数的参数列表需与信号的相同，以便接受随信号而来的参数值。如关心子进程状态的变化，那么可以如下实现信号与函数的连接：

```
var cp = new ChildProcess();
...
cp.stateChanged.connect(function(newState) {
    switch(newState) {
    case ChildProcess.NotRunning:
        log("子进程未运行");
        break;
    case ChildProcess.Starting:
        process.print("子进程正在启动中");
        break;
    case ChildProcess.Running:
        log("子进程正在运行");
```

```
        break;
        }
    });
```

### 13.2.1 connect(fun)方法

注意上面的 connect 方法，每个信号都带有该方法，它的参数是一个函数，用以当信号发射时调用它。这里直接传递一个匿名函数给 connect 的参数，匿名函数的参数将从信号中获得实际的值，当然它的参数接口必须与信号的参数接口相同。

## 13.3 方法

### 13.3.1 arguments()

该方法用于返回启动子进程的命令行参数。

### 13.3.2 errorId()

该方法用于返回子进程最后一次发生的错误标识，是进程错误 ProcessError 中的一个常数。

### 13.3.3 exitCode()

该方法返回子进程的退出码。

### 13.3.4 exitStatus()

该方法返回子进程的退出状态，是退出状态 ExitStatus 中的一个常数。

### 13.3.5 kill()

该方法杀死当前的子进程，使其立刻退出。在 Windows 系统下，使用系统的 TerminateProcess 函数，在 Unix 和 Mac OS X 下发送 SIGKILL 信号给子进程。

### 13.3.6 pid()

该方法在子进程正在运行时返回它的进程标识，如果子进程没有运行则返回 0。要注意的是，如果子进程已经退出了，那么该进程标识是无效的。

### 13.3.7 processEnvironment()

该方法返回子进程的环境变量，以一个对象返回。

### 13.3.8 readAllStandardOutput()

该方法返回子进程的所有可获取的标准输出，以 Buffer 对象返回。

### 13.3.9 readAllStandardError()

该方法返回子进程的所有可获取的标准错误，以 Buffer 对象返回。

### 13.3.10spawn(program, [args], [options])

该方法启动字符串参数 program 表示的子进程，可选参数 args 是子进程的命令行参数，它是一个字符串数组参数，第二个可选参数 options 是一个对象，可包含如下属性：

| options 属性 | 说明 |
| --- | --- |
| cwd | 字符串，设置启动子进程的工作路径 |
| env | 对象，设置启动子进程环境变量的对象 |
| detached | 布尔值，若为真，启动的子进程与当前进程无父子关系，默认为假，启动的子进程是当前进程的子进程 |
| hideWindow | 数值，隐藏子进程的窗口（包含子进程创建的子进程的窗口），子进程窗口句柄要晚于子进程本身句柄的创建，这里设定的数值是子进程启动后等待窗口句柄创建的时间，以毫秒为单位，如果晚于该时间创建了窗口句柄，则无法隐藏窗口 |
| waitForStarted | 数值，如果设置了此属性，表示将等待 ChildProcess 对象 |

| | |
|---|---|
| | 发出 started()信号再返回，或等待设置的毫秒数，-1 表示永久等待 |
| waitForFinished | 数值，如果设置了此属性，表示将等待 ChildProcess 对象发出 finished()信号再返回，或等待设置的毫秒数，-1 表示永久等待，当要等待子进程退出时，必须使用此属性 |
| waitForChildProcessFinished | 数值，如果设置了此属性，表示将等待子进程创建的子进程结束，或等待设置的毫秒数，-1 表示永久等待，当要等待子进程创建的子进程结束时，必须使用此属性。另外，当此属性与 waitForFinished 属性同时设置时，此属性将无效 |

该方法返回启动的子进程的进程标识，如果使用等待子进程结束的属性，那么返回的子进程标识当子进程退出时已经失效了。

### 13.3.11state()

该方法返回子进程的当前状态，是进程状态 ProcessState 中的一个常数。

### 13.3.12terminate()

该方法尝试退出子进程。调用此方法子进程可能不会马上退出（它给了用户保存子进程相关文档的机会）。在 Windows 系统下此方法发送了一个 WM_CLOSE 消息给子进程的顶层窗口，在 Unix 与 Mac OS X 系统下发送了 SIGTERM。Windows 的控制台程序没有事件循环，或者事件循环不处理 WM_CLOSE 消息，那么只能使用 kill()方法终止子进程。

### 13.3.13workingDirectory()

该方法返回子进程的工作目录。

### 13.3.14write(str | buf)

该方法往打开的子进程标准输入流中写入字节，写入的数据一般都在缓冲区中，

如需立即写入子进程的标准输入流中，可在字节后加上\n。接受一个字符串参数或一个 Buffer 对象，返回写入的字节数量。

# 14. Buffer 类

Buffer 类可以表示字节数组，用于存储原始的二进制字节数据。它在接口上也类似于数组 Array，如 length 属性获取数组的字节长度，可以使用操作符 buffer[i]获取或设置索引 i 处的字节，数组中每个字节在 0~255 之间。当使用操作符[]设置的索引超过现有数组的长度时，Buffer 对象将分配更大的空间；当使用操作符[]获取字节数据，但索引超出范围时，将返回 undefined。

## 14.1 方法

### 14.1.1 Buffer(buffer), Buffer(str), Buffer(len)

分别有三种方式创建一个 Buffer 对象，其一是从另一个 Buffer 对象创建；其二是传递一个字符串对象，以 UTF8 编码存入，其三是给定 Buffer 对象的字节长度。

### 14.1.2 appendBuffer(buffer)

往 Buffer 对象中追加另一个 buffer 对象。

### 14.1.3 appendString(str)

往 Buffer 对象中追加字符串。

### 14.1.4 chop(n)

从 Buffer 对象的后部移除 n 个字节。

### 14.1.5 clear()

清除 Buffer 对象所有字节。

### 14.1.6 containsBuffer(buffer)

判断 Buffer 对象是否包含另一个 buffer 对象，是则返回 true，否则返回 false。

### 14.1.7 containsString(str)

判断 Buffer 对象是否包含字符串 str，是则返回 true，否则返回 false。

### 14.1.8 countBuffer(buffer)

返回 Buffer 对象中包含另一个 buffer 的出现次数。

### 14.1.9 countString(str)

返回 Buffer 对象中包含字符串 str 的出现次数。

### 14.1.10equals(buffer)

判断 Buffer 对象与 buffer 所含数据是否相等，是则返回 true，否则返回 false。

### 14.1.11left(len)

返回 Buffer 对象左侧 len 个字节的子 Buffer 对象。

### 14.1.12toHexString()

返回 Buffer 对象的 16 进制字符串表示。

### 14.1.13toString()

返回 Buffer 对象的字符串表示。

### 14.1.14valueOf()

返回 Buffer 对象的内部值。

## 14.2 属性

### 14.2.1 length

获取或设置 Buffer 对象的字节长度。

# 15. File 类

类 File 为 JavaScript 解释器提供了与操作系统文件的输入输出（IO）能力，通过 JavaScript 的 new 运算符创建并初始化 File 新对象：

```
var file = new File(filename);
```

使用一个字符串参数表示文件路径创建文件对象后，可以使用各种模式（只读，只写，可读可写等）打开该文件。正常打开文件后，即可对文件进行读取或写入操作等。File 类与 Buffer、DataStream、TextStream 类在使用中紧密相关。

## 15.1 常量

表示文件打开模式（OpenModeFlag）的常量。

| 常量 | 值 | 说明 |
|---|---|---|
| File.NotOpen | 0x0 | 文件未打开 |
| File.ReadOnly | 0x1 | 文件以只读模式打开 |
| File.WriteOnly | 0x2 | 文件以只写模式打开，该模式隐含了截断模式 |
| File.ReadWrite | 0x3 | 文件以可读可写模式打开 |
| File.Append | 0x4 | 文件以追加模式打开，所有的数据都将写在文件末尾 |
| File.Truncate | 0x8 | 文件打开之前被截断，以前的所有内容被清除 |
| File.Text | 0x10 | 当读取文件时，行尾结束符转换为’\n’，当写文件时，行尾结束符转换为本地编码，如在 Windows 下为’\r\n’ |
| File.Unbuffered | 0x20 | 读写文件不使用缓冲区，Windows 下不支持 |

上述常量可以使用操作符“|”组合使用。

方法 error()可能返回的文件错误常量。

| 常量 | 值 | 说明 |
|---|---|---|
| File.NoError | 0 | 无错误发生 |
| File.ReadError | 1 | 发生了读取错误 |
| File.WriteError | 2 | 发生了写错误 |
| File.FatalError | 3 | 发生了致命错误 |
| File.ResourceError | 4 | 发生了资源错误，比如有太多打开的文件，内存不足等 |
| File.OpenError | 5 | 文件不能打开 |
| File.AbortError | 6 | 操作被终止 |
| File.TimeOutError | 7 | 发生了超时 |
| File.UnspecifiedError | 8 | 发生了一个未说明的错误 |
| File.RemoveError | 9 | 文件不能移除 |
| File.RenameError | 10 | 文件不能重命名 |
| File.PositionError | 11 | 文件的定位不能改变 |
| File.ResizeError | 12 | 文件不能改变尺寸 |
| File.PermissionsError | 13 | 文件不可访问 |
| File.CopyError | 14 | 文件不可拷贝 |

## 15.2 方法

### 15.2.1 File(filename)

使用 new 操作符新建一个文件对象，指定文件的名称。

### 15.2.2 atEnd()

如果文件当前的读写位置已在文件尾，则返回 true，否则返回 false。

### 15.2.3 close()

关闭文件并设置它的打开模式为 File.NotOpen。

### 15.2.4 error()

返回文件的错误状态，返回值是文件错误常量值。

### 15.2.5 flush()

把缓冲区中的数据写入文件中，如果成功返回 true，否则返回 false。

### 15.2.6 open(flags)

以输入的 flags 模式打开文件，flags 使用文件打开模式中的常量。如果文件打开成功则返回 true，否则返回 false。

### 15.2.7 openMode()

返回文件的打开模式，返回值是文件打开模式的常量值。

### 15.2.8 pos()

返回读写文件的当前位置值。

### 15.2.9 read(maxSize)

最多从文件中读取 maxSize 个字节，并以 Buffer 对象返回读取的数据。该方法无法报告错误，如果返回一个空的 Buffer 对象，则表示当前无数据可读，或有错误发生。

### 15.2.10readAll()

以 Buffer 对象返回所有可读取的数据。该方法无法报告错误，如果返回一个空的 Buffer 对象，则表示当前无数据可读，或有错误发生。

### 15.2.11readLine([maxSize=0])

从文件中读取一行，最大不超过 maxSize 个字符，如果 maxSize 为 0，读取的行可以是任意长度。默认值是 0。返回字符串对象，返回的字符串中已经没有行尾符号'\n'或'\r\n'。

### 15.2.12seek(pos)

设置读写文件的当前位置为 pos，如果成功返回 true，否则返回 false。

### 15.2.13size()

返回文件的大小。

### 15.2.14write(buffer)

把 Buffer 对象写入到文件中，返回实际写入的字节数量。如果发生错误则返回-1.

### 15.2.15writeString(str)

把字符串 str 以 UTF8 编码写入文件中。

# 16. DataStream 类

该类用于二进制数据流。可以从 Buffer、File 对象中读取或写入数据流。

## 16.1 常量

数据流状态常量

| 常量 | 值 | 说明 |
|---|---|---|
| DataStream.Ok | 0 | 流操作正常 |
| DataStream.ReadPastEnd | 1 | 流的读取操作超过了底层数据的尾部 |
| DataStream.ReadCorruptData | 2 | 流读取了坏数据 |
| DataStream.WriteFailed | 3 | 流不能把数据写入底层设施（文件或字节数组中） |

字节序常量，字节在内存中的顺序

| 常量 | 值 | 说明 |
|---|---|---|
| DataStream.BigEndian | 0 | 大端字节序，高字节存于内存低地址；低字节存于内存高地址 |
| DataStream.LittleEndian | 1 | 小端字节序，低字节存于内存低地址；高字节存于内存 |

| | | |
|---|---|---|
| | | 高地址 |

浮点数精度常量

| 常量 | 值 | 说明 |
|---|---|---|
| DataStream.SinglePrecision | 0 | 在数据流中的浮点数是 32 位精度 |
| DataStream.DoublePrecision | 1 | 在数据流中的浮点数是 64 位精度 |

数据序列化的版本常量

| 常量 | 值 | 说明 |
|---|---|---|
| DataStream.Qt_5_0 | 13 | 版本 13，Qt5.0 |
| DataStream.Qt_5_1 | 14 | 版本 14，Qt5.1 |
| DataStream.Qt_5_2 | 15 | 版本 15，Qt5.2 |
| DataStream.Qt_5_3 | 15 | 与 Qt5.2 一样，Qt5.3 |
| DataStream.Qt_5_4 | 16 | 版本 16，Qt5.4 |

## 16.2 方法

### 16.2.1 DataStream(file), DataStream(buffer, OpenMode)

两种方式创建一个 DataStream 对象，其一是关联一个打开的文件，随后的读取与写入操作都是针对该文件对象的；其二是关联一个 Buffer 对象，第二个参数指定打开方式，随后的读取与写入操作是针对该 Buffer 对象的。

### 16.2.2 atEnd()

如果到达了文件或流的尾部则返回 true，否则返回 false。

### 16.2.3 byteOrder()

返回当前的字节序设置，返回 File.BigEndian 或 File.LittleEndian。

### 16.2.4 setByteOrder(bo)

设置字节序为 File.BigEndian 或 File.LittleEndian。默认是 File.BigEndian，一般都

采用默认值，除非有特别需要。

### 16.2.5 floatingPointPrecision()

返回数据流中浮点数的精度，返回值是 File.SinglePrecision 或 File.DoublePrecision 值之一。

### 16.2.6 setFloatingPointPrecision(precision)

设置数据流中浮点数的精度为 precision。该值是 File.SinglePrecision 或 File.DoublePrecision 值之一。缺省是 File.DoublePrecision 值。

### 16.2.7 status()

返回数据流的状态，返回值是数据流状态值之一。

### 16.2.8 setStatus(status)

设置数据流的状态为 status。随后的该方法调用将被忽略，除非调用 resetStatus()。

### 16.2.9 resetStatus()

重置数据流的状态。

### 16.2.10version()

返回数据序列化的版本。返回值是各个版本号之一。

### 16.2.11setVersion(version)

设置数据序列化的版本号。一般无需设置版本号，除非使用自定义的二进制格式，或者打开以前版本写入的数据文件。

### 16.2.12skipRawData(len)

从底层设施（文件或字节数组中）忽略 len 个字节，返回实际忽略的字节数量，如果出错则返回-1。

### 16.2.13write*(data)与 read*系列方法

| 方法 | 描述 |
| --- | --- |
| readBool(), writeBool(data) | 读取和写入布尔数据，写数据返回当前的流对象，因此可以用级联的调用方式，下同 |
| readInt8(), writeInt8(data) | 读取和写入 8 位带符号整数 |
| readUInt8(), writeUInt8(data) | 读取和写入 8 位无符号整数 |
| readInt16(), writeInt16(data) | 读取和写入 16 位带符号整数 |
| readUInt16(), writeUInt16(data) | 读取和写入 16 位无符号整数 |
| readInt32(), writeInt32(data) | 读取和写入 32 位带符号整数 |
| readUInt32(), writeUInt32(data) | 读取和写入 32 位无符号整数 |
| readInt64(), writeInt64(data) | 读取和写入 64 位带符号整数 |
| readUInt64(), writeUInt64(data) | 读取和写入 64 位无符号整数 |
| readFloat(), writeFloat(data) | 读取和写入单精度浮点数 |
| readDouble(), writeDouble(data) | 读取和写入双精度浮点数 |
| readString(), writeString(data) | 读取和写入字符串 |

# 17. TextStream 类

该类用于文本数据流。可以从 Buffer、File 对象中读取或写入文本数据流。

## 17.1 常量

字段对齐常量

| 常量 | 值 | 说明 |
| --- | --- | --- |
| TextStream.AlignLeft | 0 | 在字段的右侧填充字符 |
| TextStream.AlignRight | 1 | 在字段的左侧填充字符 |
| TextStream.AlignCenter | 2 | 在字段的两侧填充字符 |
| TextStream.AlignAccountingStyle | 3 | 与 AlignRight 一样，除了数值的符号在左边 |

数值标志的常量，影响数值的输出形式（包括整数与浮点数）

| 常量 | 值 | 说明 |
| --- | --- | --- |
| TextStream.ShowBase | 0x1 | 显示进制作为数值的前缀，16（0x），8（0），或 2（0b） |
| TextStream.ForcePoint | 0x2 | 总是输出小数点，即使不是小数 |
| TextStream.ForceSign | 0x4 | 总是输出数值的符号，即使是正数 |
| TextStream.UppercaseBase | 0x8 | 使用大写输出进制的前缀（0X，0B） |
| TextStream.UppercaseDigits | 0x10 | 使用大写表示数字 10 至 35 |

数据流状态常量

| 常量 | 值 | 说明 |
| --- | --- | --- |
| TextStream.Ok | 0 | 流操作正常 |
| TextStream.ReadPastEnd | 1 | 流的读取操作超过了底层数据的尾部 |
| TextStream.ReadCorruptData | 2 | 流读取了坏数据 |
| TextStream.WriteFailed | 3 | 流不能把数据写入底层设施（文件或字节数组中） |

浮点数表示法常量

| 常量 | 值 | 说明 |
| --- | --- | --- |
| TextStream.SmartNotation | 0 | 科学或定点表示法，依赖于哪个表示法更紧凑 |
| TextStream.FixedNotation | 1 | 定点表示法 |
| TextStream.ScientificNotation | 2 | 科学表示法 |

## 17.2 方法

### 17.2.1 TextStream(file), TextStream(buffer, OpenMode)

两种方式创建一个 TextStream 对象，其一是关联一个打开的文件，随后的读取与写入操作都是针对该文件对象的；其二是关联一个 Buffer 对象，第二个参数指定打开方式，随后的读取与写入操作是针对该 Buffer 对象的。

### 17.2.2 atEnd()

如果到达了文件或流的尾部则返回 true，否则返回 false。

### 17.2.3 codecName()

返回当前文本流的编码字符串名称。

### 17.2.4 flush()

把缓冲区中的数据写入文件中，如果成功返回 true，否则返回 false。

### 17.2.5 pos()

返回读写文件的当前位置值。

### 17.2.6 seek(pos)

设置读写文件的当前位置为 pos，如果成功返回 true，否则返回 false。

### 17.2.7 read(maxSize)

最多从文件中读取 maxSize 个字符，并以字符串对象返回读取的数据。

### 17.2.8 readLine([maxSize=0])

从文件中读取一行，最大不超过 maxSize 个字符，默认值是 0，读取任意个字符，并以字符串对象返回读取的数据。该方法无法报告错误，如果返回一个空的 Buffer 对象，则表示当前无数据可读，或有错误发生。

### 17.2.9 readAll()

以字符串形式返回文本流中所有的内容。当读取大文件时避免这样调用，将消耗大量内存。如果对数据量大小未知，使用 readLine 方法为好。

### 17.2.10readNumber()与 writeNumber(num)

从文本流中读取一个数值并返回；写入一个数值 num 到文本流中。

### 17.2.11readInteger()与 writeInteger(int)

从文本流中读取一个整数并返回；写入一个整数 int 到文本流中。

### 17.2.12readString()与 writeString(str)

从文本流中读取一个字符串并返回；写入一个字符串到文本流中。

### 17.2.13setCodec(codecname)

设置文本流的编码名称，如“utf-8”，“utf-16”，“iso 8859-1”等等。

### 17.2.14skipWhiteSpace()

读取文本流并忽略空白字符，直到遇上非空白字符，或直到 atEnd()方法返回 true。

### 17.2.15setFieldWidth(width)

设置当前的字段宽度为 width，缺省情况下是 0。字段宽度等于文本流中生成的文本长度。调用后对接下来所有的操作都应用。

### 17.2.16setPadChar(char)

设置填充字符，char 是一个只包含一个字符的字符串。缺省值是 ASCII 码的空格字符。该字符用于在生成文本时填充字段的空间。

### 17.2.17setFieldAlignment(mode)

设置字段对齐模式为 mode，是字段对齐的常数值之一。

### 17.2.18setIntegerBase(base)

设置整数的进制，可以是 2（二进制），8（八进制），10（十进制）或 16（十六进制），如果是 0，那么文本流将试图检查流中的数据判断整数的进制。默认是使用十进制。

### 17.2.19setNumberFlags(flags)

设置数值标志为 flags，该值是数值标志位的组合。

### 17.2.20setRealNumberNotation(notation)

设置浮点数的表示法为 notation，该值是浮点数表示法常数之一。

### 17.2.21setRealNumberPrecision(precision)

设置浮点数的精度为 precision，不能是一个负数，缺省值是 6。

# 18. XmlStreamAttribute 类

## 18.1 方法

### 18.1.1 name()

返回属性的局部名称。

### 18.1.2 namespaceUri()

返回属性经过解析的名字空间统一资源标志符，如果属性没有定义的名字空间则返回空字符串。

### 18.1.3 prefix()

返回属性的名字空间前缀。

### 18.1.4 qualifiedName()

返回属性的限定名字。它包含名字空间的前缀，以及一个冒号，然后是属性的局部名称。

### 18.1.5 value()

返回属性的值。

# 19. XmlStreamReader 类

以流的方式解析 XML。

## 19.1 常量

错误标识常量

| 常量 | 值 | 说明 |
| --- | --- | --- |
| XmlStreamReader.NoError | 0 | 无错误发生 |
| XmlStreamReader. UnexpectedElementError | 1 | 解析器遇到了非期望的元素 |
| XmlStreamReader.CustomError | 2 | 发生了一个调用 raiseError()的定制错误 |
| XmlStreamReader.NotWellFormedError | 3 | XML 结构不良抛出错误 |
| XmlStreamReader.PrematureEndOfDocumentError | 4 | 读取流中所有 XML 文档后结构不良抛出的错误 |

读取元素文本的行为，指定了使用 readElementText()方法的不同行为。

| 常量 | 值 | 说明 |
| --- | --- | --- |
| XmlStreamReader. ErrorOnUnexpectedElement | 0 | 在读取到一个子元素时抛出 UnexpectedElementError 错误然后返回目前已读取的内容 |
| XmlStreamReader. IncludeChildElements | 1 | 递归地包含所有子元素的文本 |
| XmlStreamReader. SkipChildElements | 2 | 忽略所有子元素 |

表示 XML 文档中的各种类型记号常量

| 常量 | 值 | 说明 |
| --- | --- | --- |
| XmlStreamReader. NoToken | 0 | 读取器还未读入任何内容 |
| XmlStreamReader. Invalid | 1 | 发生了错误，可通过 error() 与 errorString()方法获取错误信息 |
| XmlStreamReader. StartDocument | 2 | 读取器可使用 documentVersion()方法 |

| | | |
|---|---|---|
| | | 获取 XML 版本号，documentEncoding()获取 XML 文档的编码。如果文档声明为 standalone，那么方法 isStandaloneDocument()返回 true，否则返回 false |
| XmlStreamReader. EndDocument | 3 | 已到文档尾部 |
| XmlStreamReader. StartElement | 4 | 已到元素的头部，可使用方法 namespaceUri()与 name()获取元素信息。如果是空元素，那么接下来将是 EndElement 记号。调用 readElementText()方法可方便地获得直到相应 EndElement 记号出现时所有的内容。调用 attributes()方法获得元素所有的属性数组。 |
| XmlStreamReader. EndElement | 5 | 已到元素的尾部 |
| XmlStreamReader. Characters | 6 | 可使用方法 text()获取字符。如果都是空白字符，则调用 isWhitespace()返回 true。如果字符来自于 CDATA 节，那么调用 isCDATA()返回 true。 |
| XmlStreamReader. Comment | 7 | 注释。可通过调用 text()获取。 |
| XmlStreamReader. DTD | 8 | DTD。可通过调用 text()获取。 |
| XmlStreamReader. EntityReference | 9 | 实体引用。引用的名字可调用 name()获取，引用的内容可调用 text()获取。 |
| XmlStreamReader. ProcessingInstruction | 10 | 处理指令。可通过调用方法 processingInstructionTarget()和 processingInstructionData()获取。 |

## 19.2 方法

### 19.2.1 XmlStreamReader(file)

创建一个读取 XML 文件的对象，file 是一个 File 类实例，它打开了一个可读文件。

### 19.2.2 atEnd()

读取器已读取到 XML 文档尾部或发生了一个错误导致异常退出返回 true，其它情况下返回 false。

### 19.2.3 attributes()

返回当前正读取元素的属性数组，数组中的元素是 XmlStreamAttribute 类型。

### 19.2.4 characterOffset()

返回当前字符偏移量，从 0 开始。

### 19.2.5 clear()

清除读取器中所有数据，重置内部状态为初始化状态。

### 19.2.6 documentEncoding()

如果 tokenType()是 StartDocument，那么此方法返回 XML 文档中声明的编码字符串，否则返回一个空字符串。

### 19.2.7 documentVersion()

如果 tokenType()是 StartDocument，那么此方法返回 XML 文档中声明的版本字符串，否则返回一个空字符串。

### 19.2.8 error()

返回当前错误的类型常数，如果返回 NoError 则表示无错误。

### 19.2.9 errorString()

返回使用 raiseError()设置的错误字符串。

### 19.2.10hasError()

若有错误发生返回 true，否则返回 false。

### 19.2.11isCDATA()

若读取器获得了源自于 CDATA 节的字符返回 true，否则返回 false。

### 19.2.12isCharacters()

若 tokenType()是 Characters 则返回 true，否则返回 false。

### 19.2.13isComment()

若 tokenType()是 Comment 则返回 true，否则返回 false。

### 19.2.14isDTD()

若 tokenType()是 DTD 则返回 true，否则返回 false。

### 19.2.15isEndDocument()

若 tokenType()是 EndDocument 则返回 true，否则返回 false。

### 19.2.16isEndElement()

若 tokenType()是 EndElement 则返回 true，否则返回 false。

### 19.2.17isEntityReference()

若 tokenType()是 EntityReference 则返回 true，否则返回 false。

### 19.2.18isProcessingInstruction()

若 tokenType()是 ProcessingInstruction 则返回 true，否则返回 false。

### 19.2.19isStandaloneDocument()

若 XML 文档声明了 standalone 则返回 true，否则返回 false。若没有 XML 声明可解析，也返回 false。

### 19.2.20isStartDocument()

若 tokenType()是 StartDocument 则返回 true，否则返回 false。

### 19.2.21isStartElement()

若 tokenType()是 StartElement 则返回 true，否则返回 false。

### 19.2.22isWhitespace()

若读取器读取的字符只有空白字符则返回 true，否则返回 false。

### 19.2.23name()

返回读取到 StartElement，EndElement 或 EntityReference 记号处的局部名称。

### 19.2.24namespaceUri()

返回读取到 StartElement 或 EndElement 的元素名字空间的统一资源标志符。

### 19.2.25prefix()

返回读取到 StartElement 或 EndElement 的元素前缀名。

### 19.2.26processingInstructionData()

返回读取到 ProcessingInstruction 记号的数据。

### 19.2.27processingInstructionTarget()

返回读取到 ProcessingInstruction 记号的目标。

### 19.2.28qualifiedName()

返回读取到 StartElement 或 EndElement 的元素限定名称。

### 19.2.29raiseError([message])

使用可选的 message 错误信息抛出定制的错误。

### 19.2.30readElementText([behaviour])

在读取到一个 StartElement 记号时的方便方法，利用它可读取到对应的 EndElement 记号之间的所有文本。如果没有发生错误，那么调用此方法后的当前记号是 EndElement。

方法将连接所有遇到的 Characters 或 EntityReference 记号，但忽略 ProcessingInstruction 和 Comment 记号中的文本并返回它。若当前记号不是 StartElement，那么将返回一个空字符串。

可选参数 behaviour（常量）用于定义在读取到 EndElement 之前的行为。它可以包含或者忽略子元素的文本，或者抛出一个 UnexpectedElementError 异常并返回当前已读取的文本（默认是此种行为）。

### 19.2.31readNext()

读取下一个记号并返回记号的类型。一个例外是，当发生了错误时将不再读取 XML 流，此时 atEnd()返回 true，hasError()返回 true，该方法返回 Invalid。

### 19.2.32readNextStartElement()

在当前元素中驱动读取器读取到下一个元素开始处。若到达了一个元素的开始处返回 true，当到达了元素的结束位置，或发生了错误则返回 false。

该方法适合于只对解析 XML 元素感兴趣的场景。

### 19.2.33skipCurrentElement()

驱动读取器读取到当前元素的尾部，忽略所有的子节点。该方法对忽略未知的元素很有用。

### 19.2.34text()

返回记号是 Characters，Comment，DTD 或 EntityReference 的文本。

### 19.2.35tokenString()

返回读取器的当前记号的字符串表示。

### 19.2.36tokenType()

返回当前记号的类型。

# 20. log(arg1, [arg2], […])函数

全局函数 log 可以往图形界面程序的 JavaScript Console 窗口中输出各个参数值。

# 21. require(filename)函数

SimCube 的 JavaScript 解释器除了使用 C++代码编写的功能接口扩展之外，还可以利用 JavaScript 语言编写扩展。为了能够方便地加载使用 JavaScript 实现的功能，SimCube 的 JavaScript 解释器实现了一个全局函数 require(filename)，利用它可以动态加载 JavaScript 文件（可以把这些文件称为模块，这些文件以 js 为扩展名），从而给解释器添加模块中实现的功能。利用 require 函数即可把待实现的相关功能切分到不同的模块文件中，从而在设计与实现上达到良好的封装与重用性。

## 21.1 模块加载路径

全局函数 require 加载模块时，只需指定模块的文件名，省略 js 扩展名称，同时可

自动在设定的路径中搜寻该模块文件。通过访问 require 函数的 paths 属性可以获得该路径，它是一个字符串数组对象，默认提供四个路径：

1. 应用程序可执行文件所在的路径或使用 require 加载模块文件语句所在文件的路径（也即在一个模块文件中使用 require 再加载一个子模块的情况下）；
2. 应用程序可执行文件所在路径下的/plugins/*appId* 路径，如自动化应用是/plugins/automation 目录；
3. 应用程序可执行文件所在路径下的/plugins 路径；
4. 应用程序可执行文件所在路径下的/ package 路径

第一个路径是动态生成的，因此在使用 require.paths 属性查看路径列表时看不到此项。require 函数在加载模块时首先在第一个路径下搜索，如果没有再搜索第二个路径，直到找到该模块文件为止，如果所有的路径搜索完毕后也不能找到该模块，则抛出异常。

通过修改 require.paths 属性（一个字符串数组）可以动态改变模块加载路径，如使用数组的 unshift()方法在数组头部插入元素，即可优先搜索这个路径。这里有个例外，在一个模块文件中使用 require 再加载一个子模块的情况下，第一个搜索的路径总是该模块文件所在的路径。

注意：当两个相同名称的模块分别放入搜索的不同路径下，那么只会加载第一个路径下找到的模块。一般可以通过改变模块的名称解决此问题，或者动态调整搜索路径的顺序。

## 21.2 模板规范

一个模块是一个文件扩展名为 js 的 JavaScript 文件，模块内部预置了三个变量可以使用：

1. __filename：任何模块文件内部，可以使用此变量获取当前模块文件的带有完

整绝对路径的文件名；

2. __dirname：任何模块文件内部，可以使用此变量获取当前模块文件所在目录的完整绝对路径；

3. exports：模块文件对外导出的一个对象，调用 require 函数后返回该对象。

在 JavaScript 解释器的全局空间定义了一个名为“$MODULES”的变量，它是一个对象，是以模块文件的绝对路径为键，以导出的对象为值。

## 21.3 模块示例

编写一个简单的 JavaScript 脚本文件 test.js。内容如下：

```
1  var data = 123;
2
3  function add(a, b) {
4      return a + b;
5  }
6
7  exports = { file: __filename,
8              dir: __dirname,
9              data: data,
10             add: add
11 };
```

第 1 行：定义了一个 data 变量，它的值是 123；

第 3-5 行：定义了一个 add 函数，返回两个参数的和；

第 7-11 行：把当前模块文件名称以及路径，该模块定义的 data 数据，add 函数放入模块已经定义的 exports 变量中，作为模块的外部导出对象。

把模块文件 test.js 放入默认定义的三个目录下，调用加载模块的 require 函数，获得模块中导出的对象，然后通过该对象即可访问模块中提供的导出变量与函数。如下所示：

```
var test = require('test');
log(test.file);
```

```
log(test.dir);
log(test.data);
log(test.add(1, 2));
```

如果不是把模块文件放入默认定义的三个目录下，如放入“e:\scripts”中，那么在调用 require 函数前，需把该路径加入模块搜索路径中：

```
require.paths.push('e:/scripts');
```

随后按上面所示调用 require 函数即可导入模块。