永洪脚本 Z-Script

用户使用手册

版权声明

免责声明

通讯方式

北京永洪商智科技有限公司

北京市朝阳区光华路 9 号光华路 SOHO 二期 C 座 9 层（100020）

电话：(86-10)-58430919

邮箱：public@yonghongtech.com

网站：http://www.yonghongtech.com

目录

第 1 章：JavaScript 编程简介

许多仪表盘环境需要动态定制仪表盘和实现自定义业务逻辑。这些需求从创建查询字段，到建立脚本查询；从修改文本颜色，到控制用户交互行为；从添加计算器，到控制表格渲染，等等。Yonghong Z-Suite 有一套完整的脚本环境体系，以支持用户的各种动态的需求。脚本体系是 Yonghong Z-Suite 产品中使用面较广的功能，让用户可以自己定制化一些高级需求，是决定产品是否强大的一个重要模块。

脚本系统是通过 JavaScript 的语言标准来支持的。脚本环境可以完全访问组件的绑定，属性，以及数据的输入；可以在仪表盘初始化的时候执行任务；可以处理用户交互行为。

本文从定制脚本的应用场景来分别产生脚本的使用，包括查询的自定义字段，脚本查询，计算器，动态计算器，表格渲染，组件级别的脚本和仪表盘级别的脚本等方面。

JavaScript 简介

通过介绍 JavaScript 编程过程，来了解如何将 JavaScript 脚本嵌入到仪表盘中。

面向对象的概念

JavaScript 是面向对象的编程语言 (OOP)，OOP 语言使我们有能力定义自己的对象和变量类型。它提供多种对象和方法，以及用户能够自定义方法。可以认为 JavaScript 里面所有东西几乎都是对象。

- 对象。对象只是一种特殊的数据。对象拥有属性和方法。
- 属性。属性是与对象有关的特征值，如名称，长度等。
- 方法。方法指对象可以执行的行为（或者可以完成的功能）。

脚本语言的基础

JavaScript 语言与通常的 C++， Java 等面向对象语言的语法类似，先创建类，类包含方法，然后再实例化对象来创建对象。

在 JavaScript 中，当我们用 function 关键字来创建一个函数时，实际上在 JavaScript 中，是按照对象来进行管理的，且我们可以动态的设置该对象的属性及方法等。一下是一个典型的例子：

```
// 定义构造函数，并设定一个属性
function Person(name){
    this.name = name;
}
// new 关键字实例化一个对象
var Tom = new Person("Tom");
```

注释和命名

JavaScript 使用双斜线 "//" 开始单行注释，以 "/**/" 来添加一个多行注释。

```
// 单行注释
/* 多行
注释 */
```

分号 ";" 作为语句分隔符。

```
var a_1 = 0;
```

JavaScript 变量用于保存值或表达式。可以给变量起一个简短名称 , 比如 x, 或者更有描述性的名称 , 比如 length。 JavaScript 变量也可以保存文本值 , 比如 carname="Volvo"。

JavaScript 变量名称的规则：

- 变量对大小写敏感（ y 和 Y 是两个不同的变量 ）
- 变量必须以字母或下划线开始
- 变量允许使用的字符包括：字母，数字，下划线

声明和赋值

Javascript 是一种弱类型的语言。这并不意味着它没有数据类型 , 只是变量或者 Javascript 对象属性不需要一个特定类型的值分配给它或者它始终使用相同的值 , 一个变量可以被分配任何值。该变量的类型是由当前赋值的类型来决定。因此 , 一个局部变量在使用之前 , 需要声明。使用关键字 "var" 来申明一个变量。

```
var a_1 = " 你好 ";
```

如果您所赋值的变量还未进行过声明 , 该变量会自动声明。

```
x=5;
carname="Volvo";
```

对象的类型和作用域

JavaScript 是基于对象的。这意味着在 JavaScript 中的每一个值都是一个对象。

```
// 下面的语句是等价的
var c = a.concat(b);
c = a + b;
```

一般来说 , 可以创建并使用的对象有三种：本地对象、内置对象和宿主对象。

JavaScript 把本地对象 （ native object ）定义为 “ 独立于宿主环境的 ECMAScript 实现提供的对象 ”。简单来说，本地对象就是 JavaScript 定义的类 （ 引用类型 ）。它们包括：Array， String， Number 等。

内置对象 （ built-in object ）定义为 “ 由 ECMAScript 实现提供的、独立于宿主环境的所有对象，在 ECMAScript 程序开始执行时出现 ”。这意味着开发者不必明确实例化内置对象，它已被实例化了。JavaScript 只定义了两个内置对象，即 Global 和 Math。

```
a = parseInt(1234);// parseInt 就是全局对象。
b = max(10, 12);// max 就是 Math 对象。
```

所有非本地对象都是宿主对象（ host object ）, 即由 ECMAScript 实现的宿主环境提供的对象。所有 BOM 和 DOM 对象都是宿主对象。

作用域指的是变量的适用范围。JavaScript 中只存在一种作用域 - 公用作用域。所有属性和方法默认都是公用的。

Number 对象

Number 对象是原始数值的包装对象。

```
var n = new Number(value);
var n = Number(value);
```

JavaScript 会自动地把原始数值转化成 Number 对象，调用 Number 方法的既可以是 Number 对象，也可以是原始数字值。

```
var n = 123;
```

数字可以是十进制，八进制，十六进制格式，默认是十进制。如果它以 "0x" 开头就是十六进制格式，如果以 "0" 开始，就是八进制格式。

```
var n1 = 80;// 默认为十进制格式
var n2 = 0xff;// 十六进制格式
var n3 = 012;// 八进制格式
```

数字类型可以使用四则运算符 +, *, /, -。也可以使用递增递减运算符 ++, --。

Boolean 对象

Boolean 对象表示两个值："true" 或 "false"。当一个值用在条件语句中，为定义的值是被当做 "false" 的结果返回。如果要判断一个值是否定义过，就可以用如下语句：

```
if(value) {
// 执行语句
}
```

String 对象

String 对象用于处理文本（字符串）。字符串通常用单引号或双引号括起来。

```
var s1 = " 你好 ";
var s2 = " 你好 ";
```

可以用加运算符的把字符串串联起来：

```
var s3 = s1 + " 吗 ?";
```

字符串还有很多常见的方法，包括 substring()，toLowerCase()， toUpperCase()， indexOf() 等等。

```
var s1 = "abc";
s1 = s1.toUpperCase(); // 转成 ABC
var idx = s1.indexOf("B"); // 返回 1
s1 = s1.substring(1, 2); // 返回 "B"
```

字符串内置了对正则表达的支持。match() 方法可在字符串内检索指定的值，或找到一个或多个正则表达式的匹配。 search() 方法用于检索字符串中指定的子字符串，或检索与正则表达式相匹配的子字符串。

```
var s2 = "abcdefg";
var reg = /bc/; // 创建正则表达式
var idx = s2.search(reg); // 返回 "bc" 的位置
```

Date 对象

Date 对象用于处理日期和时间。 Date 对象会自动把当前日期和时间保存为其初始值。

```
var d1 = new Date();
```

采用全局函数 formatDate() 可以将一个日期对象转化为一个字符串：

```
var s1 = formatDate(d1, "yyyy-MM-dd"); // 2010-02-21
```

Array 对象

Array 对象用于在单个的变量中存储多个值。列表中的每个项目作为一个数组元素，并包含在方括号（[]）中。当创建一个数组时，它被初始化使用指定的值作为其数组元素。其长度被设置为指定的元素的数目。

```
var arr1 = ["beijing", "shanghai", "tianjin"];
```

多维数组是一个数组的数组。

```
var arr2 = [[1, 2], [3, 4], [5, 6]];
```

条件语句

JavaScript 中的条件语句用于完成不同条件下的行为。在您编写代码时，经常需要根据不同的条件完成不同的行为。可以在代码中使用条件语句来完成这个任务。

- If 语句
- If...else 语句
- If...else if...else 语句

```
if(x > 0) {
  x = x + 1;
}
else {
  x = x - 1;
}
```

for 循环语句

在编写代码时，你常常希望反复执行同一段代码。我们可以使用循环来完成这个功能，这样就用不着重复地写若干行相同的代码。 JavaScript 中的循环用来将同一段代码执行指定的次数（或者当指定的条件为 true 时）。在脚本的运行次数已确定的情况下使用 for 循环。

```
for ( 变量 = 开始值 ; 变量 <= 结束值 ; 变量 = 变量 + 步进值 ) {
   需执行的代码
}
```

while 循环语句

JavaScript 中的循环用来将同一段代码执行指定的次数（或者当指定的条件为 true 时）。while 循环用于在指定条件为 true 时循环执行代码。

```
while ( 变量 <= 结束值 ) {
   需执行的代码
}
```

有两种特殊的语句可用在循环内部 : break 和 continue。break 命令可以终止循环的运行，然后继续执行循环之后的代码（如果循环之后有代码的话）。 continue 命令会终止当前的循环，然后从下一个值继续运行。

switch-case 语句

JavaScript 中的条件语句用于完成基于不同条件的行为。如果希望选择执行若干代码块中的一个，你可以使用 switch 语句：

```
var theDay = new Date().getDay();
switch (theDay) {
  case 5:
```

```
    s = "Finally Friday";
    break
  case 6:
    s = "Super Saturday";
    break
  case 0:
    s = "Sleepy Sunday";
    break
  default:
    s = "Work day";
}
```

switch 后面的 (n) 可以是表达式，也可以（并通常）是变量。然后表达式中的值会与 case 中的数字作比较，如果与某个 case 相匹配，那么其后的代码就会被执行。 break 的作用是防止代码自动执行到下一行。

JavaScript 函数

函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。将脚本编写为函数，就可以避免页面载入时执行该脚本。

函数包含着一些代码，这些代码只能被事件激活，或者在函数被调用时才会执行。

你可以在页面中的任何位置调用脚本（如果函数嵌入一个外部的 .js 文件，那么甚至可以从其他的页面中调用）。

```
function prod(a,b) {
  if(a > 0 && b > 0) {
    var x = a * b;
    return x;
  }
  else {
    return 0;
  }
}
```

JavaScript 变量的生存期

当您在函数内声明了一个变量后，就只能在该函数中访问该变量。当退出该函数后，这个变量会被撤销。这种变量称为本地变量。您可以在不同的函数中使用名称相同的本地变量，这是因为只有声明过变量的函数能够识别其中的每个变量。

如果您在函数之外声明了一个变量，则页面上的所有函数都可以访问该变量。这些变量的生存期从声明它们之后开始，在页面关闭时结束。

第 2 章：Script 查询

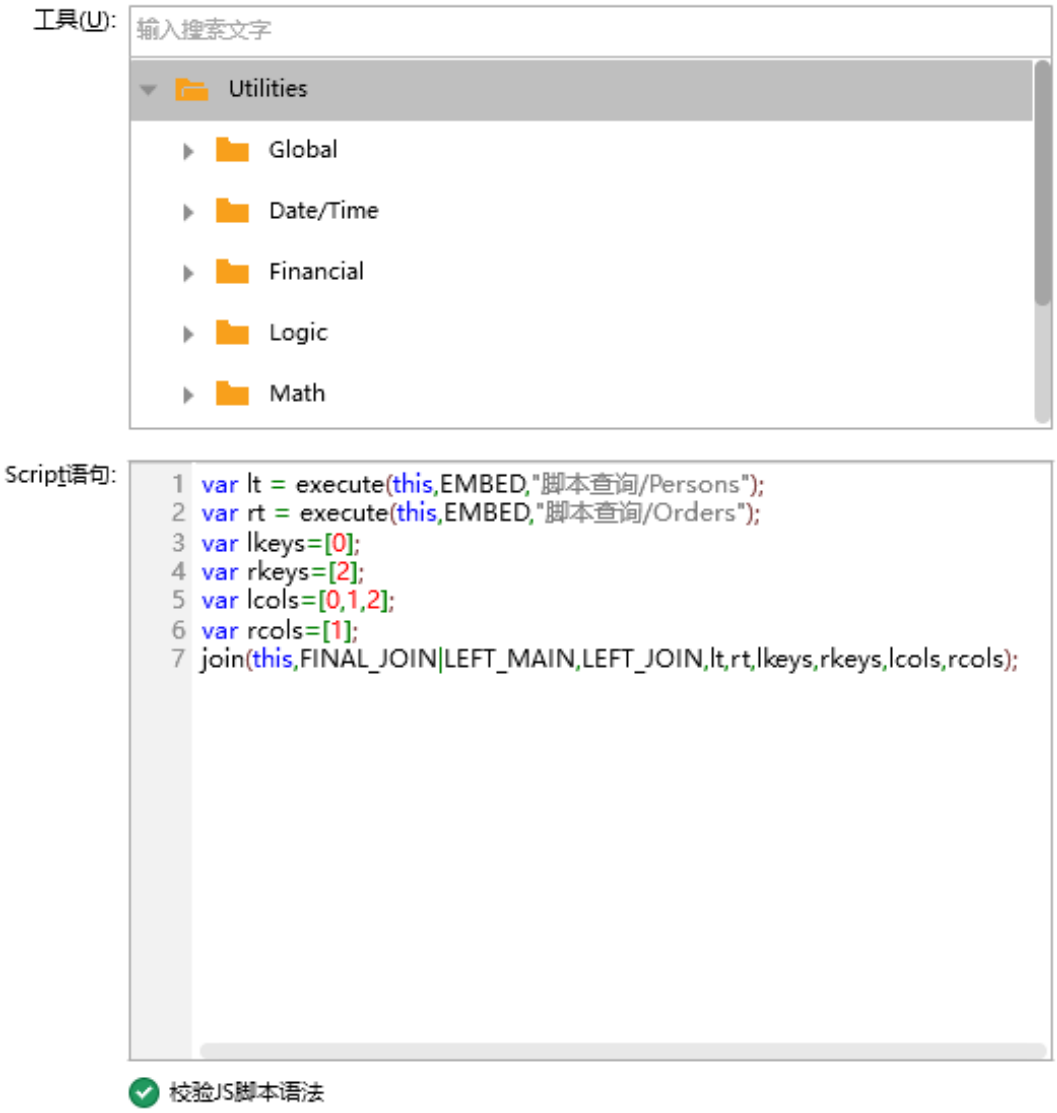
Script 查询提供脚本语言和数据接口的方式，实现 SQL 查询所提供的功能。但这种方式比较于 SQL 查询本身设计上的局限，又能使查询操作更加灵活自由，为专业人士提供了便利。通过在工具库里提供了一些常用 JavaScript 函数接口， Script 查询支持客户通过编程的方式进行查询。虽然这些接口被限制在设定功能范围内，但是已经能够满足现有日常的工作。

此外 Script 查询提供了 SQL 查询不支持的联合查询功能。其类似于 SQL 中的联合查询，根据客户需求，将来自不同数据表单上的数据，通过脚本 JOIN 查询方法展现在一张表单上。但 Script 查询提供了比 SQL 的联合查询提供了更为强大的功能。 SQL 联合查询的前提是所有的数据表单都来自于同一个数据库，而 Script 查询可以连接不同数据库的数据表单，例如某个公司的经营数据保存在 DB2，而其管理数据保存在 ORACLE，客户通过 Script 查询可以把他们联合在一起并组成一张新的查询表用来分析，实现了不同数据源之间的合并。

用户可以建立一个脚本查询，以便把不同数据源的数据混搭到一张查询表上。例如一个是 access 数据库的 Sql 表，一个 embed 类型的查询表，先把数据执行出来，再 Join 到一起，形成新表。

编辑界面

先启动编辑 Script 查询的界面，（如何启动，请参考永洪连接数据手册）。在打开的界面中，用户可编写脚本语句，来实现多表的连接。



界面中有三块区域，最上面是提供工具函数的树状结构，中间是一个供输入脚本的编辑区域；最下面是一个信息校验反馈栏。点击树节点时，会自动将点中节点的文本内容插入到脚本编辑区域，而且是光标所在的位置。当前编辑区域的内容会在每隔一段时间被自动校验，校验的结果显示在反馈栏里。

此编辑区间的目的是需要返回一个查询结果。中间可以有其他运行结果，但最后必须返回是一个查询结果。Script 查询的主要用途是把不同数据源的数据混搭到一张查询表上。那就需要把其他的查询先执行出来，再通过一些 Join 函数，把各表连接起来。

Script 查询可实现不同类型查询的链接，各查询的类型与对应标识显示如下图所示：

查询	对应标识
SQL 查询	SQL
Excel 查询	EXCEL
Script 查询	SCRIPT
Mongo 查询	MONGO
定制查询	CUSTOM
数据集市查询	CLOUD
内嵌数据查询	EMBED
组合查询	COMPOSITE
自服务数据查询	DATA_FLOW

执行查询的函数

一个已经定义的查询，可以通过调用执行函数，返回查询的结果。

execute(Scriptable script, int type, String path, Object allCols)

参数：

script 是指运行此脚本的作用域，通常用 this，表示当前作用域。

type 是指查询的类型，支持的 9 种类型 :SQL, EXCEL, SCRIPT, CUSTOM, CLOUD, EMBED, COMPOSITE， MONGO， DATA_FLOW。

path 是指查询的路径及名称。如果有目录就用 '/' 连接起来。

allCols 是指当为 true 时，查询的表达式列可见，当为 false 时，查询的表达式列不可见。

返回：

一个查询的结果

举例：

execute(this, EMBED, "folder1/query1", true);

Join 函数

将两个任意的查询执行后的查询结果，做联接，并返回新的查询结果。要求做映射的关键字段的数据类型要匹配。

join(Scriptable scope, int jhint, int jop, Object jleft, Object jright, Object jlkeys, Object jrkeys, Object jlcols, Object jrcols)

参数：

scope 是指运行此脚本的作用域，通常用 this，表示当前作用域。

jhint 是 join 的提示。 LEFT_MAIN 左表为主， RIGHT_MAIN 右表为主， FINAL_JOIN 是最终表，TEMP_JOIN 是个临时表，会被回收的表。

jop 是连接操作符。 JOIN 是内连接； LEFT_JOIN 左连接； RIGHT_JOIN 右连接； FULL_JOIN 全连接。

jleft 左表

jright 右表

jlkeys 左表的连接的字段

jrkeys 右表的连接的字段

jlcols 左表的留下哪些字段

jrcols 右表的留下哪些字段

返回：

一个查询的结果

举例：

join(this, FINAL_JOIN | LEFT_MAIN, LEFT_JOIN, query1, query2, [1], [3], [1,2], [3,1]);// 左表 query1 的第 1 列和右表 query2 的第 3 列连接，留下 query1 的 1 和 2 列， query2 的 3 和 1 列。

Union 函数

将两个任意的查询执行后的查询结果进行合并，并返回新的查询结果。两个查询的字段个数是匹配的，而且数据类型也是匹配的。

union(Scriptable scope, Object ugrids)

参数：

script 是指运行此脚本的作用域，通常用 this，表示当前作用域。

ugrids 哪些表求交集。如果列不匹配会出错。此参数是个数组。

返回：

一个查询的结果

举例：

union(this, [a, b]);// 将 a 和 b 进行合并。

取列函数

将某些列的数据从一个查询结果里提取出来，并返回一个新的查询结果。

columns(Scriptable scope, Object cgrid, Object ccols)

参数：

script 是指运行此脚本的作用域，通常用 this，表示当前作用域。

cgrid 是指哪个查询结果被提取。

ccols 是指哪些列被提取。

返回：

一个查询的结果

举例：

columns(this, a, [3,1]);// 将 a 的第 3 列和第 1 列提取出现。

排序函数

将查询结果中的某些列做排序，并返回一个新的查询结果。

sort(Scriptable scope, Object sgrid, Object scols, Object sascs)

参数：

scope 是指运行此脚本的作用域，通常用 this，表示当前作用域。

sgrid 是指哪个查询结果被排序。

scols 哪些字段被排序。

sascs 各字段用什么排序类型：true(升序)， false （ 降序)。

返回：

一个查询的结果

举例：

sort(this, a, [3,1], [false, true]);// 先将 a 的第 3 列降序，再将 a 的第 1 列升序。

例如，希望把一个 sybase 数据库的查询结果和一个 excel 文件作为数据来源的查询结果做一个联接。假设这两个查询的关键字段的数据类型是匹配的。

```
var lt = execute(this, SQL, "sybase/customer"); // 左表
var rt = execute(this, SQL, "Excel_Coffee_chain"); // 右表
var lkeys = [0];// 左表的第 0 列做 join
var rkeys = [1];// 右表的第 1 列做 join
var lcols = [0, 1];// 左表保留第 0 和 1 列
var rcols = [0, 2];// 右表保留第 0 和 2 列
join(this, FINAL_JOIN, LEFT_JOIN, lt, rt, lkeys, rkeys, lcols, rcols); // 做左联接，此结果是最终结果，返回新查询结果。
```

第 3 章：自定义字段的脚本

用户自定义的字段是指在查询的编辑界面，创建新的字段，作用域是当前查询，所有使用该查询的仪表盘都能使用此字段。在元数据区域界面上，右键选择增加表达式，并输入字段名称，选择数据类型，加入脚本语言来返回字段内容，脚本中不能使用聚合函数，创建后，可以通过拖拽，放到维度（Dimension）节点下或者度量（Measure）节点下。

创建方式

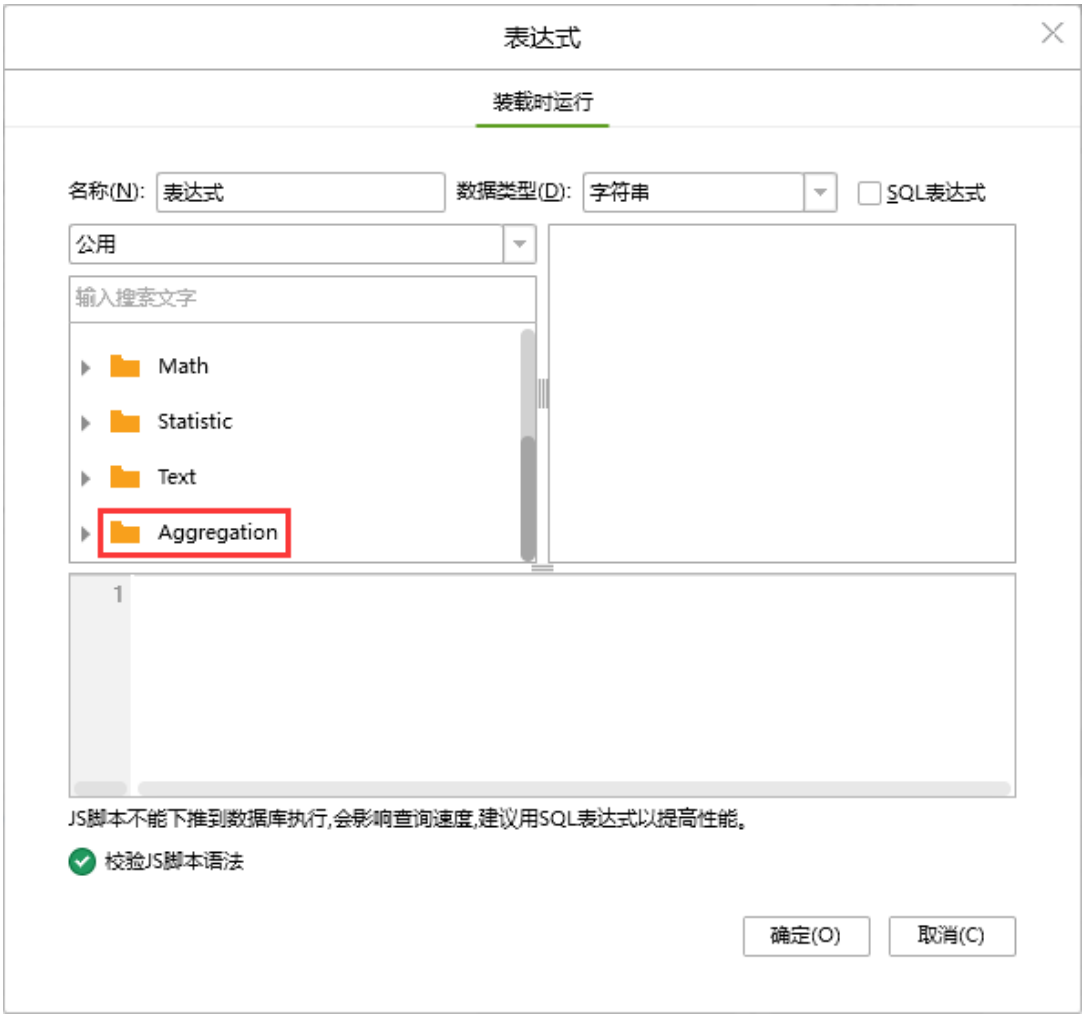
在元数据区域（下图中的红色区域中）用户可创建自己的数据段，该数据段只在当前查询中存在，在其他同类型的查询中不存在。在元数据区域右键选择增加表达式，即可打开表达式对话框。

详细介绍

在打开的对话框中，用户可在下图的脚本区域输入脚本来实现数据段的创建。在查询中创建数据段时，可调用 Utilities 文件夹下提供的各种函数，但不能使用 Aggregation 文件夹下提供的聚合函数。关于函数的详细介绍见顶级作用域的函数介绍。

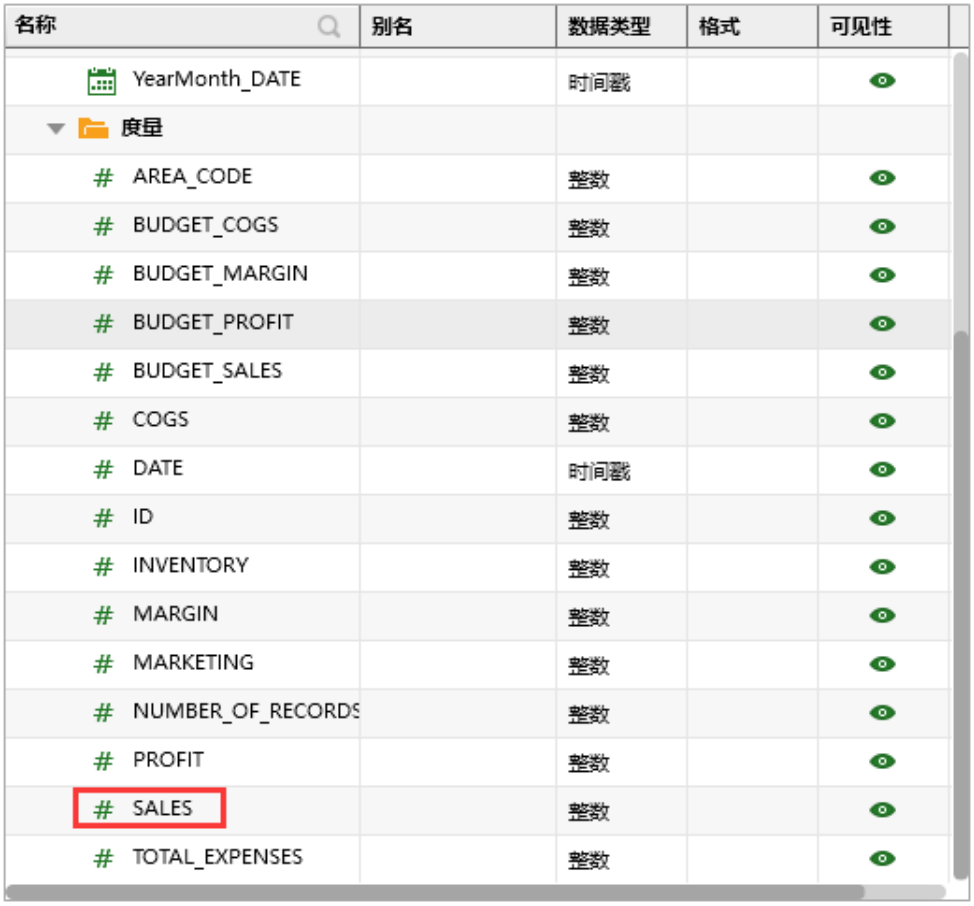
【文本】设定数据段的名称。

【数据类型】设定数据段的类型。

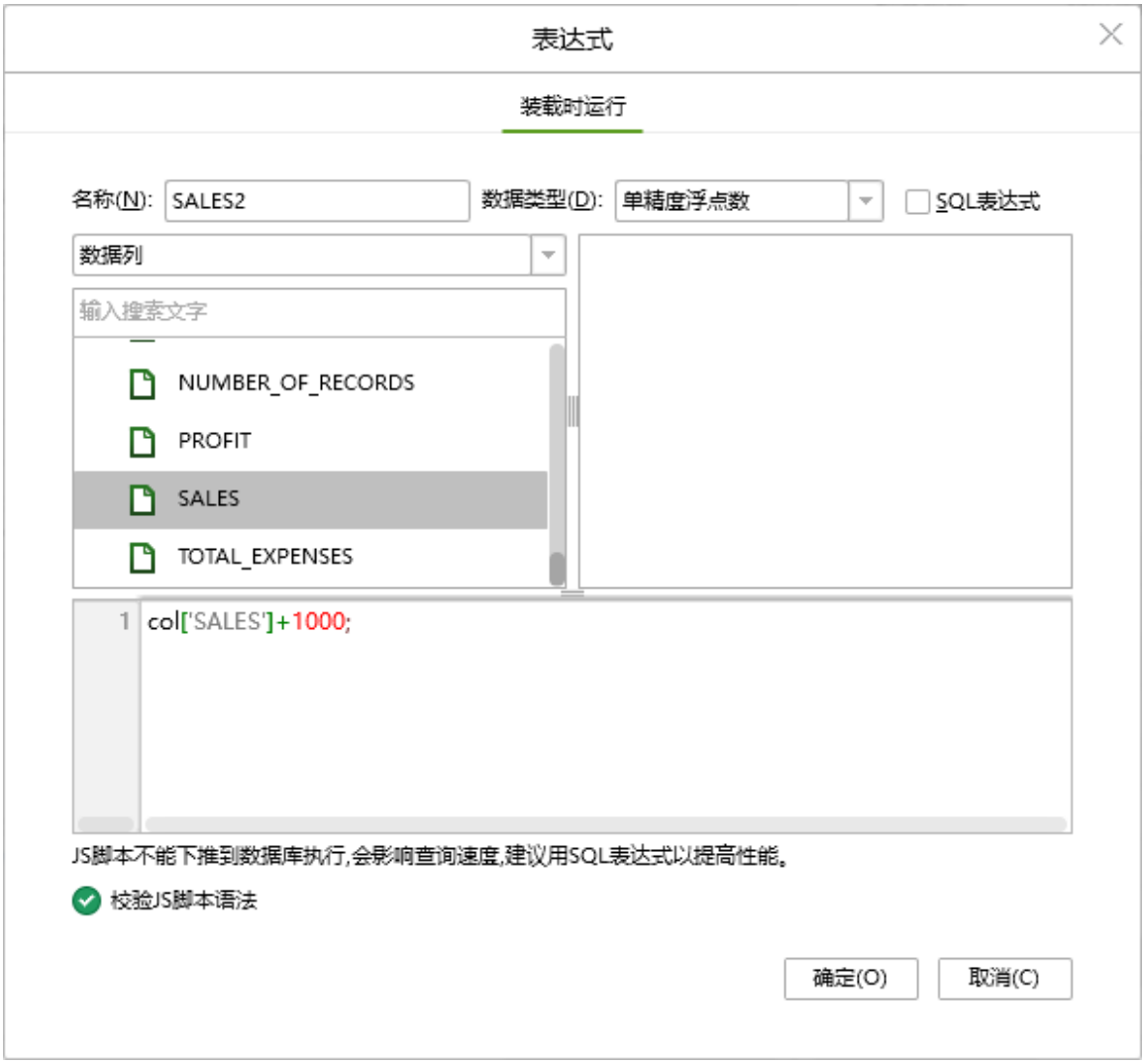


举例说明

如在元数据区中存在 SALES 数据段，如下图所示。



当用户需要给 SALES 字段中的每条数据增加 1000 时，可通过脚本来创建新的字段，假设名称为 SALES2，只需点击 SALES 列，然后在后面添加上 +1000，脚本内容为 col['SALES']+1000; 如下图所示。



则在度量节点下生成 SALES2 字段，如下图所示。

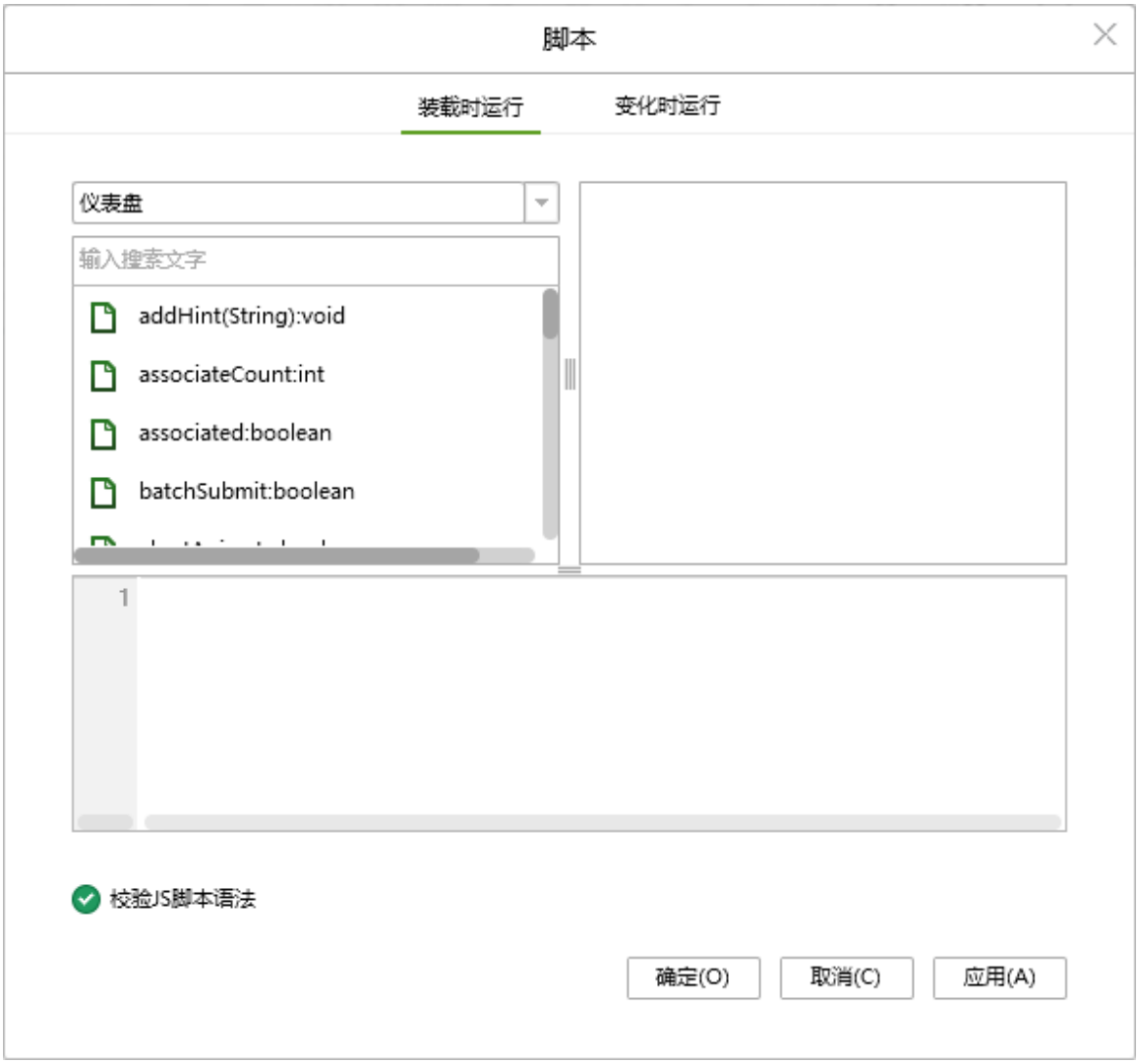
第 4 章：仪表盘的脚本

用户可通过脚本来实现对整个仪表盘的控制，例如通过脚本来实现对仪表盘定时刷新。通过仪表盘的脚本也可实现对各个组件的整体控制。

仪表盘的脚本分为装载时运行的脚本和变化时运行的脚本，两个类型的脚本的执行条件不同，装载时运行的脚本是在仪表盘打开时运行，而变化时运行的脚本是在组件有变化时运行。

打开方式

在仪表盘的空白处右键选择脚本，即可打开脚本对话框。



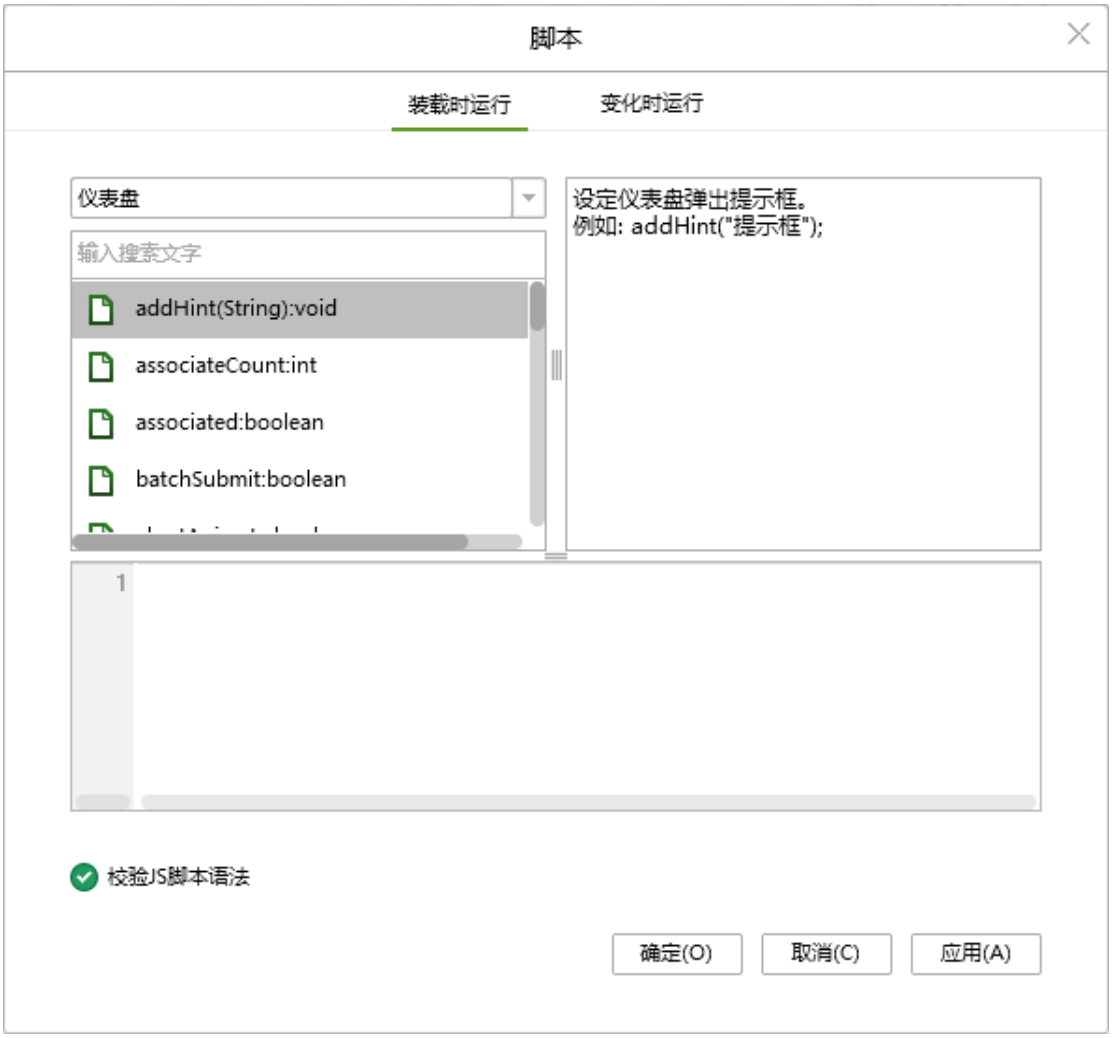
【装载时运行】即在新打开仪表盘时，或预览仪表盘时运行脚本内容。

【变化时运行】即在对组件变化时运行脚本内容。

脚本函数的详细介绍见顶级作用域的函数介绍以及仪表盘级别的函数。

仪表盘脚本函数描述

在仪表盘脚本函数上点击一下，右边框中即出现对应的描述。

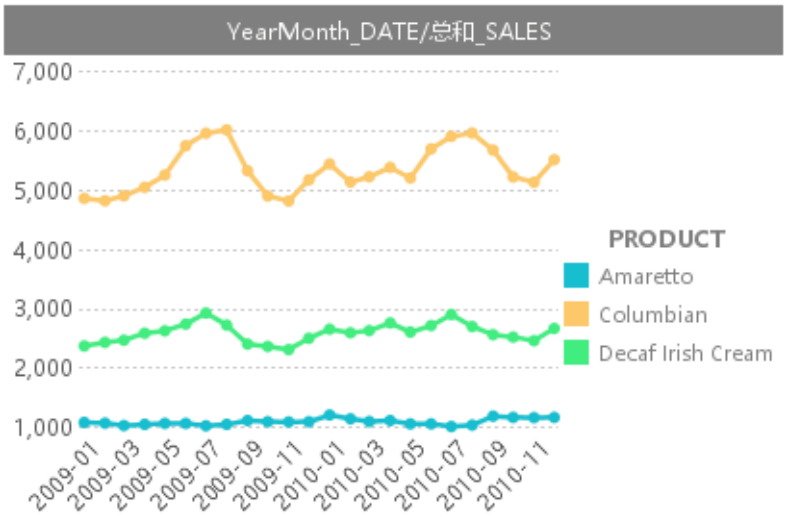


仪表盘组件脚本、顶级作用域脚本、以及动态计算器脚本都有相应的脚本函数描述。

举例说明

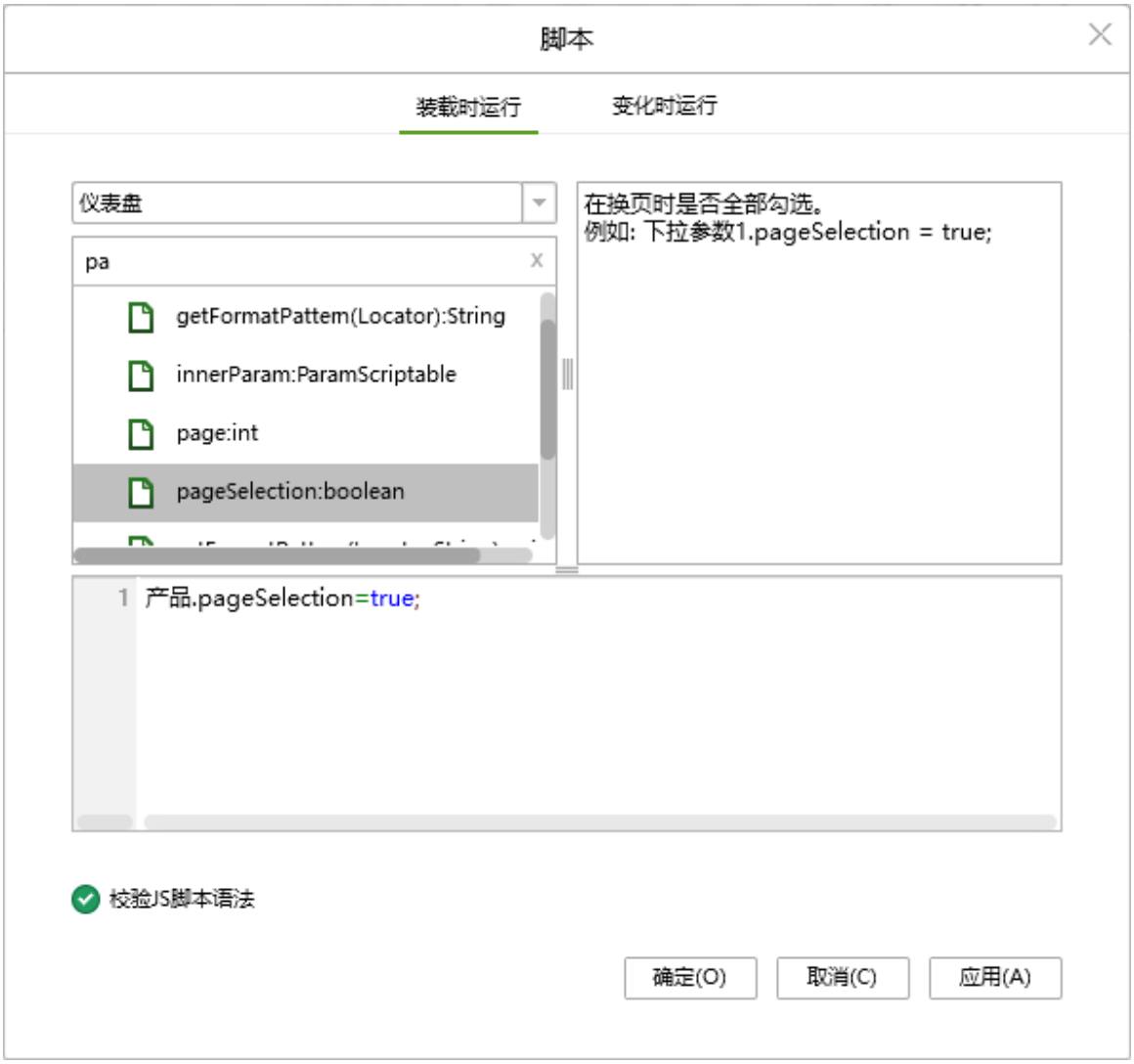
应用举例 1

如用户需要在查看报告中或预览模式下实现当选择一个产品类型时，该类型的所有产品均被勾选，并在图表上显示出来。即一个参数内容改变时，另一个参数内容自动全选。

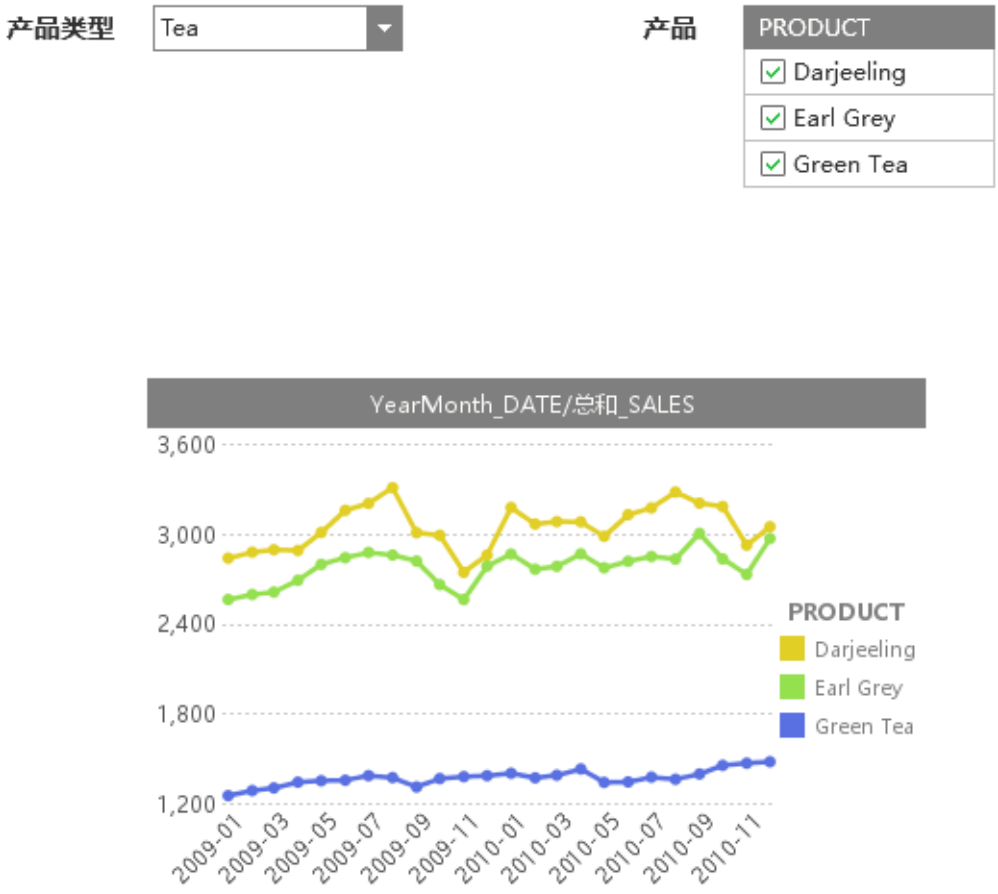


用户需要在图表上设定过滤器，过滤条件为产品 是 其中一个 ?{ 产品 }，其中复选框的名称为产品，作为参数传递给图表。同时需要在列表参数上设定过滤器，过滤条件为产品类型 =?{ 产品类型 }，其中下拉列表的名称为产品类型，作为参数传递到列表参数。用户需要在该仪表盘的空白处右键，选择脚本，选择装载时运行，脚本内容为：

产品 .pageSelection=true;



则在查看报告中或在预览模式中打开该仪表盘时，切换产品类型参数中的选项，则产品参数中的内容全选，如下图所示。



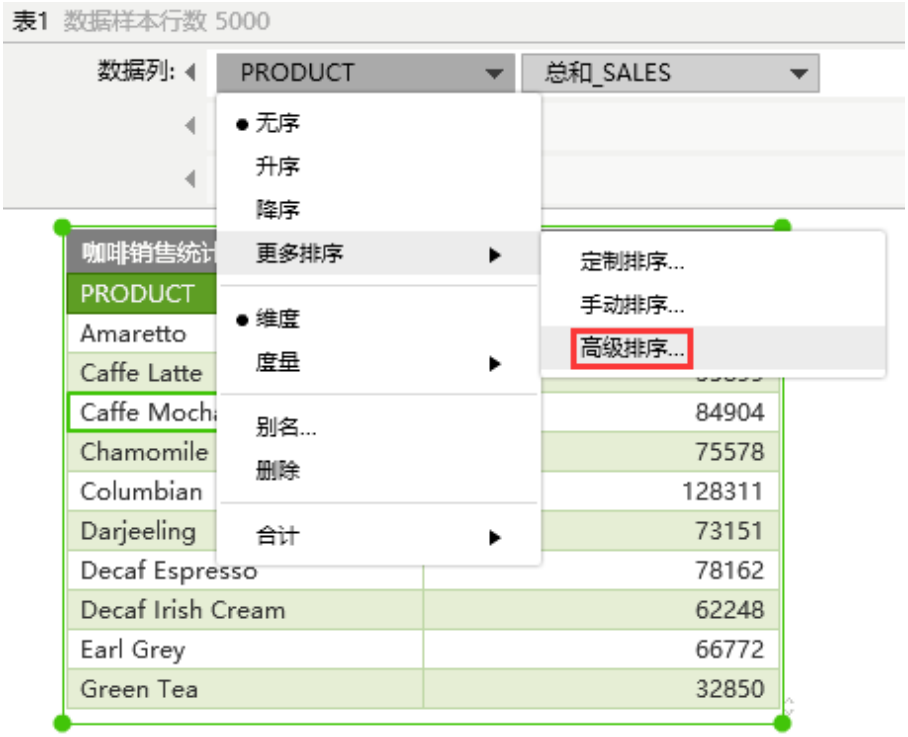
应用举例 2

1. 假设存在一张表，如下图所示。

咖啡销售统计	
PRODUCT	总和_SALES
Amaretto	26269
Caffe Latte	35899
Caffe Mocha	84904
Chamomile	75578
Columbian	128311
Darjeeling	73151
Decaf Espresso	78162
Decaf Irish Cream	62248
Earl Grey	66772
Green Tea	32850

2. 为了实现对总和 _SALES 的排名统计，需要使用高级排序。使用高级排序的前提是表组件处于聚合状态。

高级排序适用于聚合状态的表、交叉表以及处于聚合状态的图表。只有维度数据段才具有高级排序属性，度量类型的数据段不具有此属性，高级排序可实现对聚合的度量字段的排序。

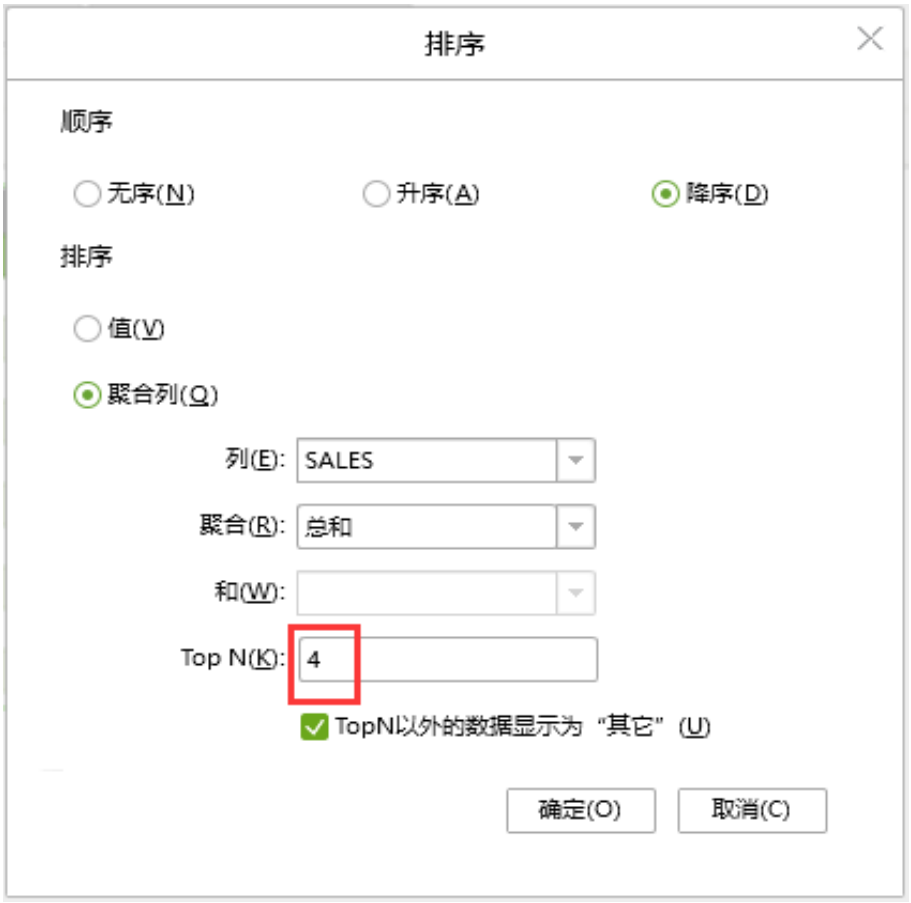


3. 如下图实现了表组件中的数据按照 SALES 字段的总和信息进行降序排列，即按照总和 _SALES 字段降序排列。

4. 排序结果如下图所示。

咖啡销售统计	
PRODUCT	总和_SALES
Columbian	128311
Lemon	95926
Caffe Mocha	84904
Decaf Espresso	78162
Chamomile	75578
Darjeeling	73151
Earl Grey	66772
Decaf Irish Cream	62248
Caffe Latte	35899
Mint	35710

5. 当在高级排序对话框中的保留选项选择 4 时，则表中的数据将被过滤，只保留总和 _SALES 的前四个数据，勾选 TopN 外数据显示为其它时，除保留的 TopN 数据，剩余数据以其它显示。



6. 保留结果如下图所示。

咖啡销售统计	
PRODUCT	总和_SALES
Columbian	128311
Lemon	95926
Caffe Mocha	84904
Decaf Espresso	78162
其它	432508

7. 当用户需要实时更改保留个数时，可通过脚本来实现。

- 首先创建一个列表参数并修改为单选，绑定任一查询中的数字类型数据，如下图所示。

咖啡销售统计	
PRODUCT	总和_SALES
Columbian	128311
Lemon	95926
Caffe Mocha	84904
Decaf Espresso	78162
其它	432508

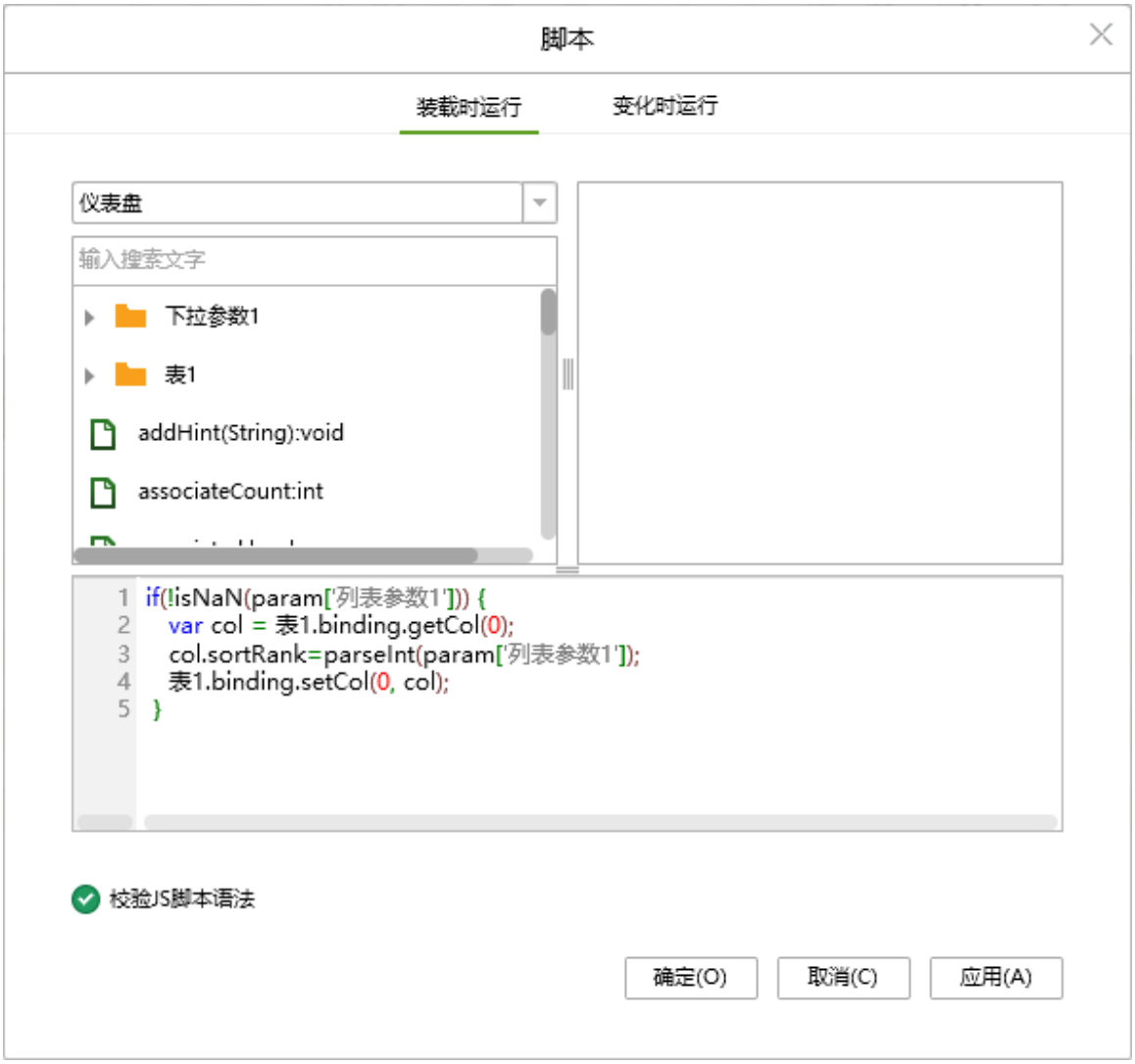
ID
1
2
3
4
5
6

• 在编辑区的空白处右键选择脚本。

• 在脚本对话框中的变化时运行选项卡中输入脚本。

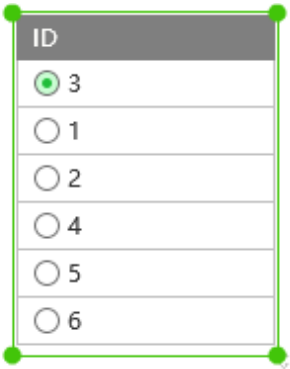
```
if(!isNaN(param[' 列表参数 1'])) {
  var col = 表 1.binding.getCol(0);
  col.sortRank=parseInt(param[' 列表参数 1']);
  表 1.binding.setCol(0, col);
}
```

实现将列表参数 1 中的参数值传递给高级排序对话框中的保留选项。



• 在参数列表 ID 中选择任一选项实现对表组件的动态筛选，即按照总和 _Sales 字段的降序顺序进行筛选。

咖啡销售统计	
PRODUCT	总和_SALES
Columbian	128311
Lemon	95926
Caffe Mocha	84904
其它	510670

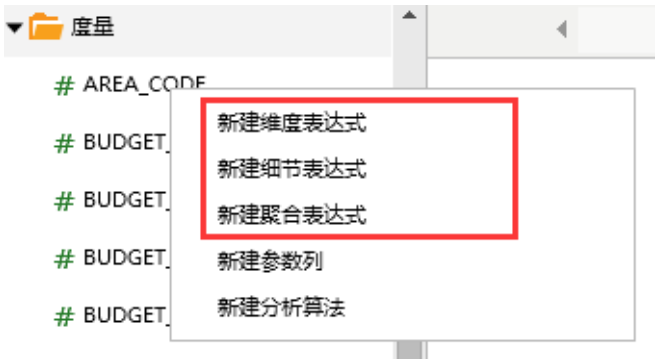


第 5 章：计算器的脚本

在打开的仪表盘中，用户可通过脚本来创建数据段，此数据段只适用于当前仪表盘，不适用于其他仪表盘。用户数据段包括新建维度表达式、新建细节表达式、新建聚合表达式。

创建方式

打开任何一个组件的绑定界面，即打开查询列表，在查询列表中右键即可选择需要创建的数据段类型，如下图所示。



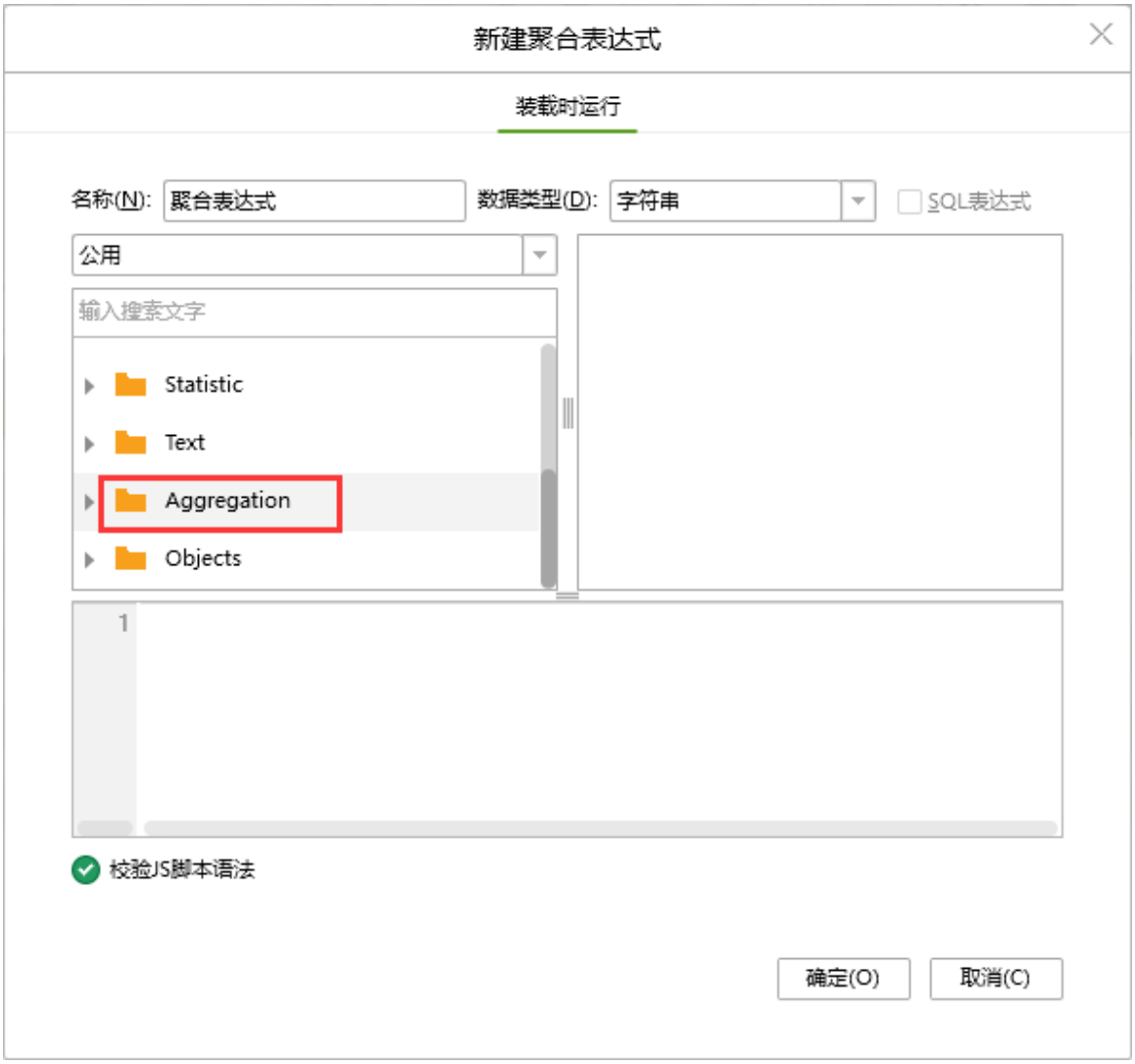
当用户选择其中一个表达式时，打开对话框，如下图所示。

【名称】设定数据段的名称。

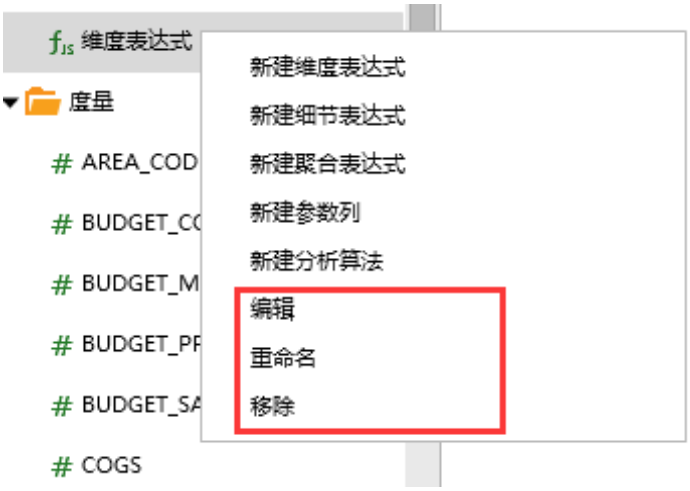
【数据类型】设定数据段的类型。

【SQL 表达式】当用户勾选此项时，支持 SQL 语句，新建聚合表达式时，灰化显示，不能勾选，如下图所示。

用户可使用 Utilities 文件夹中提供的函数，各个函数的用法将在顶级作用域的函数介绍详细介绍。注意维度表达式和细节表达式不能使用 Aggregation 文件夹中提供的聚合函数，只有聚合表达式才能够使用。



当用户创建好数据段后需要再次进行编辑、重命名、移除时，用户可选中此数据段，然后右键选择相应的选项卡即可。

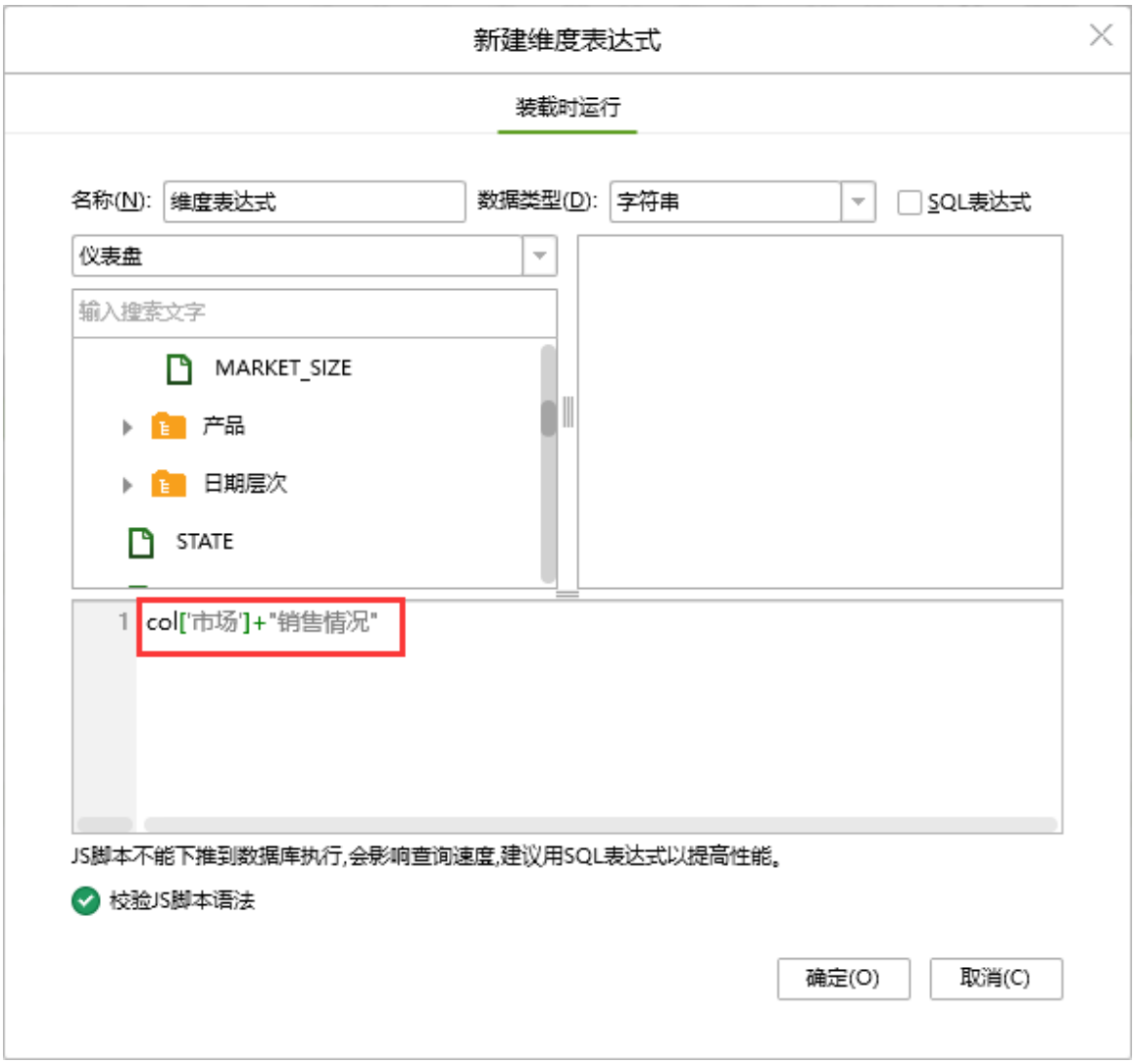


维度表达式

当用户需要创建作为维度来进行统计的数据段时，可选择新建维度表达式。维度表达式默认存放在维度目录下，不能被拖拽到度量目录下。用户可在打开的维度指标的计算器对话框中设定该数据段的名称、该数据段的数据类型、编写脚本，来实现维度数据段的创建。数据类型最好是字符串或字符等适合作为维度的类型，脚本中不能使用聚合函数。

举例说明：

假设存在一个维度数据段为市场，当用户需要给每个市场添加后缀“销售情况”时，可通过维度指标的计算器来实现，只需点击相应的列，如：市场，然后在其后加上”+ 销售情况”，如下图所示。



如下表所示，其中的市场销售情况数据段是通过新建维度表达式实现的。

3

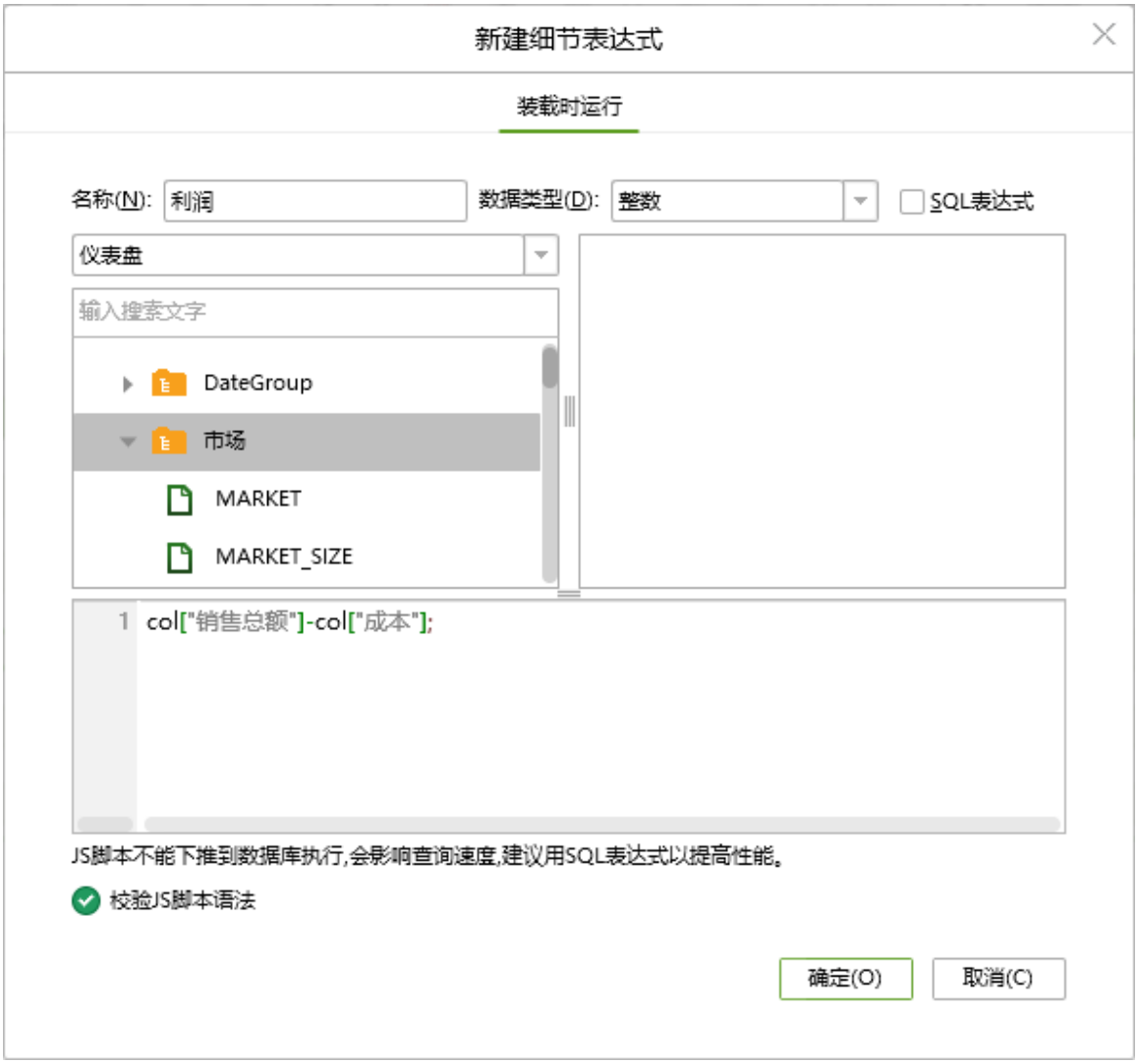
市场销售情况	总和_SALES
Central销售情况	265045
East销售情况	178576
South销售情况	103926
West销售情况	272264

细节表达式

当用户需要对某个数据段中的数据进行修改，则可创建细节表达式。细节表达式默认存放在度量目录下，不能被拖拽到维度目录下。用户可在打开的细节表达式对话框中设定该数据段的名称、该数据段的数据类型、编写脚本，来实现度量数据段的创建。数据类型最好是数值类型，适合进行度量，脚本中不能使用聚合函数。

举例说明：

假设存在两个数据段，一个是月销售总额，一个是月花销成本，当用户想计算出每月的利润时，可通过细节指标的计算器来实现，脚本内容为两个数据段之间的差值，新建细节表达式如下图所示。



如下表所示，其中的利润数据段是通过新建细节表达式实现的。

3

产品	销售总额	成本	利润
Amaretto	219	94	125
Columbian	190	68	122
Decaf Irish C	234	101	133
Green Tea	100	30	70
Caffe Mocha	134	54	80
Decaf Espres	180	53	127
Chamomile	341	99	242
Lemon	150		
Mint	140	33	107
Darjeeling	130	17	113

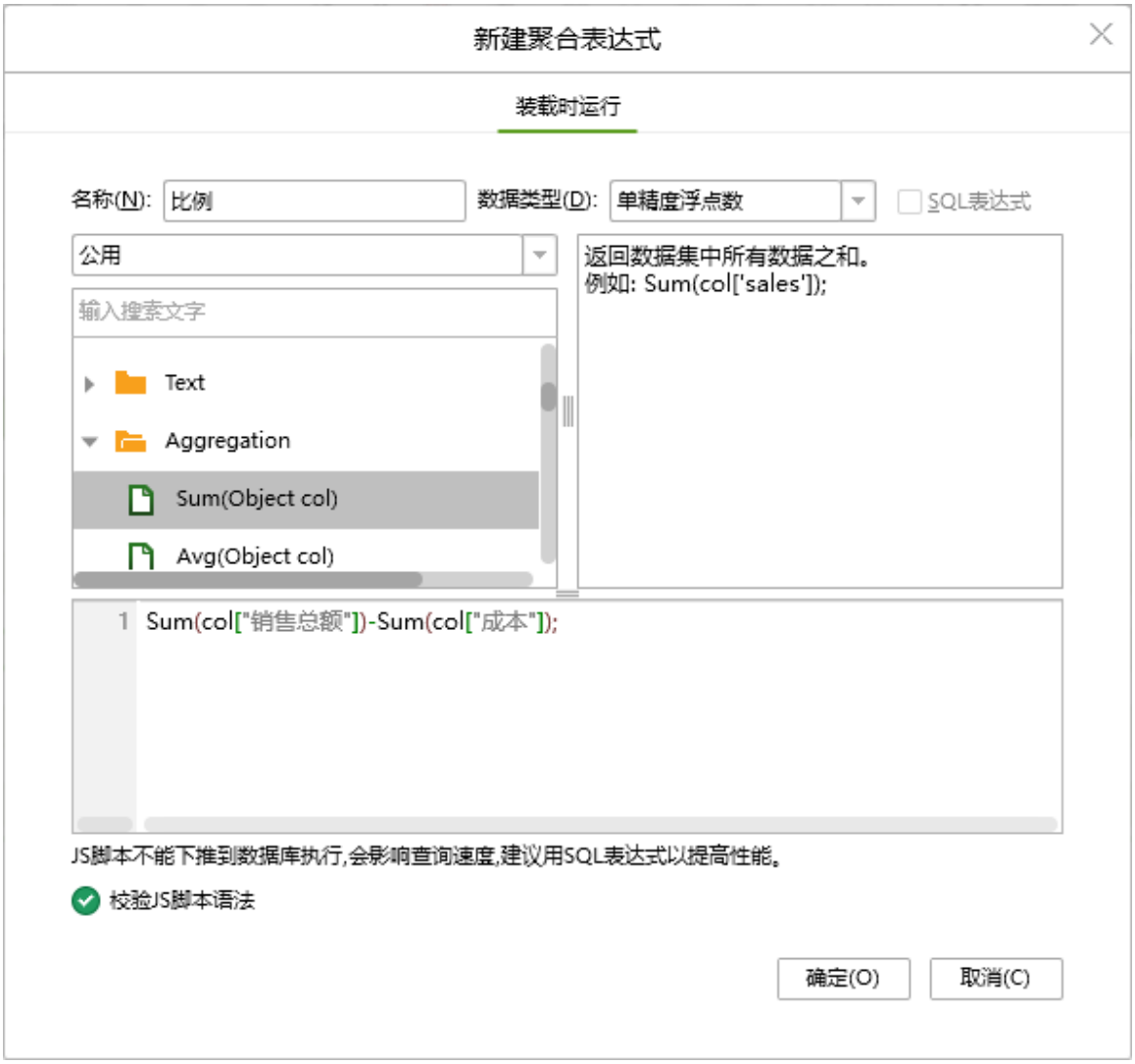
聚合表达式

当用户需要对某个数据段中的数据进行汇总时，则可新建聚合表达式。聚合表达式默认存放在度量目录下，不能被拖拽到维度目录下。用户可在打开的聚合表达式对话框中设定该数据段的名称、该数据段的数据类型、编写脚本，来实现度量数据段的创建。数据类型最好是数值类型，适合进行度量，脚本中可使用聚合函数。

聚合表达式只有在输出类型的组件（文本、表、交叉表、图表、仪表）中才能被绑定，并且组件处于聚合状态。

举例说明：

假设存在三个数据段，月销售总额、月花销成本、市场。当用户想计算出每个地区的销售总额与花销成本的比例时，可通过聚合表达式来实现，脚本内容为两个数据段求和之后的比例，如下图所示。



如下表所示，其中的比例数据段是通过新建聚合表达式实现的。

3

市场		总和_销售总额	总和_成本		比例
Central	+	265045	93852		2.824
East	+	178576	59217		3.016
South	+	103926	32478		3.2
West	+	272264	73996		3.679

第 6 章：组件的脚本

很多时候用户希望通过脚本去动态修改组件的属性，绑定数据来源。脚本会在组件被执行前运行，通过改变某个属性，就可以把某个部分隐藏。

给组件定义脚本，当组件被刷新的时候，会运行该脚本．组件的脚本和仪表盘上的脚本一样，都有顶级作用域和仪表盘作用域上支持的函数，对象，常量。

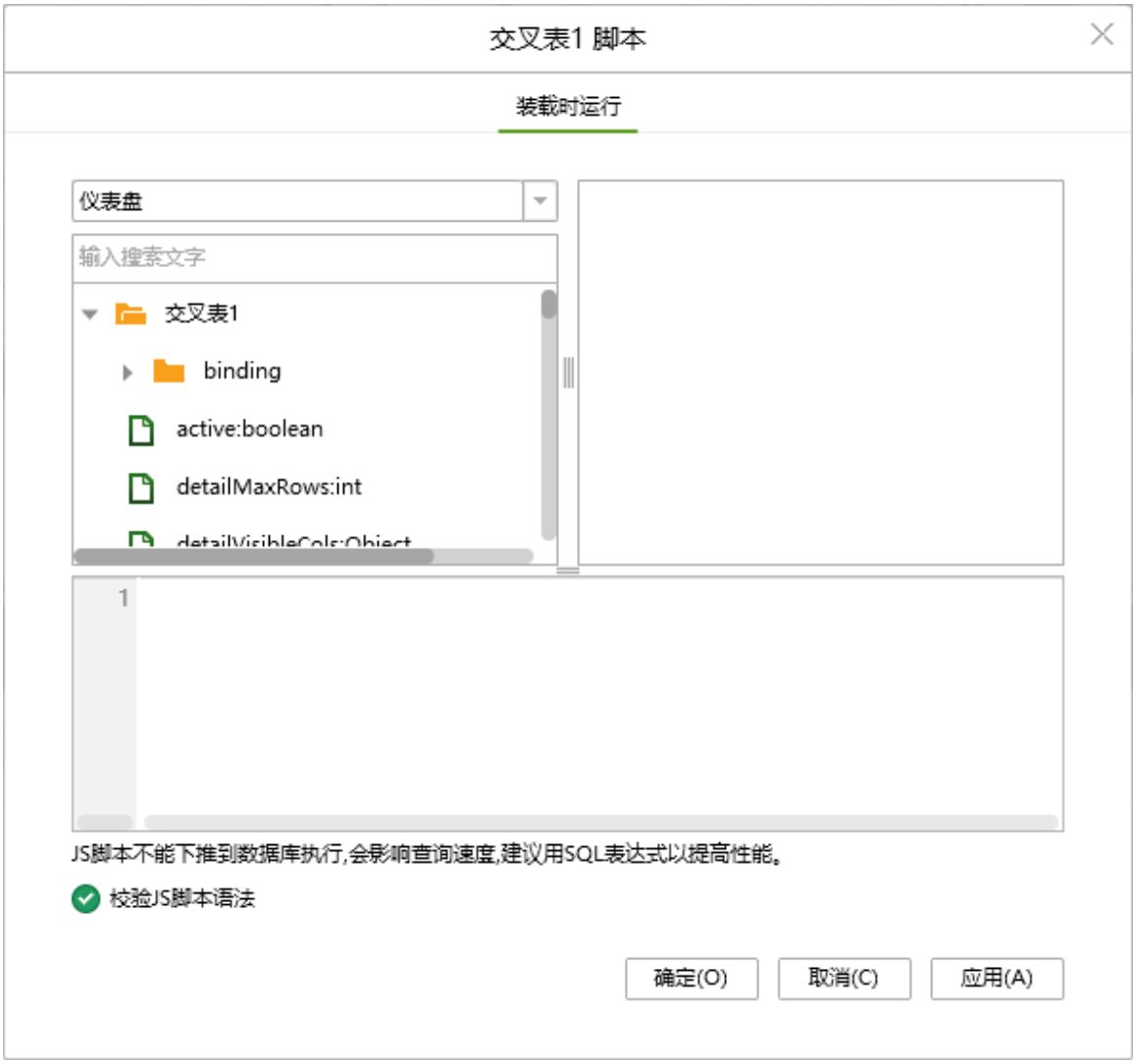
实际上，通过脚本可以访问到任意仪表盘上的组件。但通过某个组件上的脚本去修改其他组件的脚本是不建议使用的，可能会导致依赖关系混乱。

打开方式

在特定组件上右键选择脚本，即可打开组件的脚本对话框。

右键菜单如下图所示。

例如，在交叉表上右键打开脚本，结果如下图所示的对话框，交叉表对应的脚本文件夹展开，用户可输入脚本。



基本概念

本节讲述脚本使用的基本概念。

访问组件属性

可以通过组件的名字来访问一个组件。例如， “Table1” 是某个表格组件的名字。

```
Table1.width = 100;
```

如果组件名字中有空格或其他特殊字符，需要把名字用引号引起来，并作为 db 的一个属性来访问。这 db 是对当前仪表盘的一个引用。

```
db["Table1"].width = 100;
```

这句和上面那句脚本是等价的，都是修改 Table1 的宽度值。

作用域

在 Yonghong Z-Suite 的脚本环境中，包含多层次的作用域。对于仪表盘中的对象，有如下几层作用域：

- 顶级作用域
- 仪表盘作用域
- 组件作用域

当一个符号被用在脚本中，会首先检查它的作用域范围。如果在组件内部没找到，就检查仪表盘作用域，最后是顶级作用域。

这就意味着，在访问属性的时候，也可以不用带上组件名称。例如在一个组件 Table1 上加脚本：

```
height = 200;
```

如果在仪表盘上加脚本，不是在组件上加脚本，就只能通过组件名来访问：

```
Table1.height = 200;
```

如果这两句都是加到 Table1 上，那么就是等价的，都是修改 Table1 的高度。

脚本执行顺序

一个仪表盘上有很多组件，每个组件都可以引入脚本，而且还可以在一个组件的脚本里访问其他脚本的属性。脚本的最终执行效果是受执行时序控制的。如果修改同一个属性，哪个脚本是最后被触发的，最终效果就受它影响。例如，在 Table1 上修改了 Table2 的背景色，而 Table2 上也修改了自己的背景色。最终效果要看是 Table1 被后执行，还是 Table2 被后执行。如果 Table2 被最后执行，那就显示 Table2 里修改的背景色。

以下规则是用来评判在一个仪表盘的生命周期中，脚本的执行顺序的标准：

- 在装载时运行的脚本，是最先执行的。此脚本定义在仪表盘上。
- 用户被要求通过输入框输入的参数。
- 所有的组件的查询数据被执行。
- 组件上的脚本被执行。

如果在仪表盘级别里，" 在变化时运行 " 的脚本中定义了脚本。

- 当仪表盘上有任何变化时，设定在变化时运行的脚本被执行。
- 最后还是要执行组件上的脚本。

脚本的层次

通过脚本可以访问组件的属性，可以调用方法。可以修改一个组件的不同层次的内容。例如对于一个图表组件：

1. 通过修改属性，格式，超链接等的脚本，来达到修改图表的展现。此时的数据绑定是不修改的。
2. 通过修改轴，标记，图例等信息的脚本，添加字段，来达到修改绑定的信息。此时会影响到数据的输出。把城市字段换成国家字段。数据完全不同。
3. 通过直接修改数据和 BChart 等信息的脚本，来达到修改数据和画图的信息。此时就不需要组件上有绑定信息就能画出图表来。和 2 的区别就是，2 需要有绑定，通过绑定来生成数据。而 3 可以直接执行一个查询，放回查询结果作为图表的数据。此用法会限制图表的交互行为，多用在表格渲染上。例如某列的所有格子都用图表来显示，画出各个维度下的销售趋势线。

修改组件的属性

所有的组件有一些属性是通用的，例如前景色，背景色，高宽，位置，可见，可用，字体，字号，等等。当你在编辑区域给组件定义属性的时候，可以通过树状列表查阅到该对象的属性。

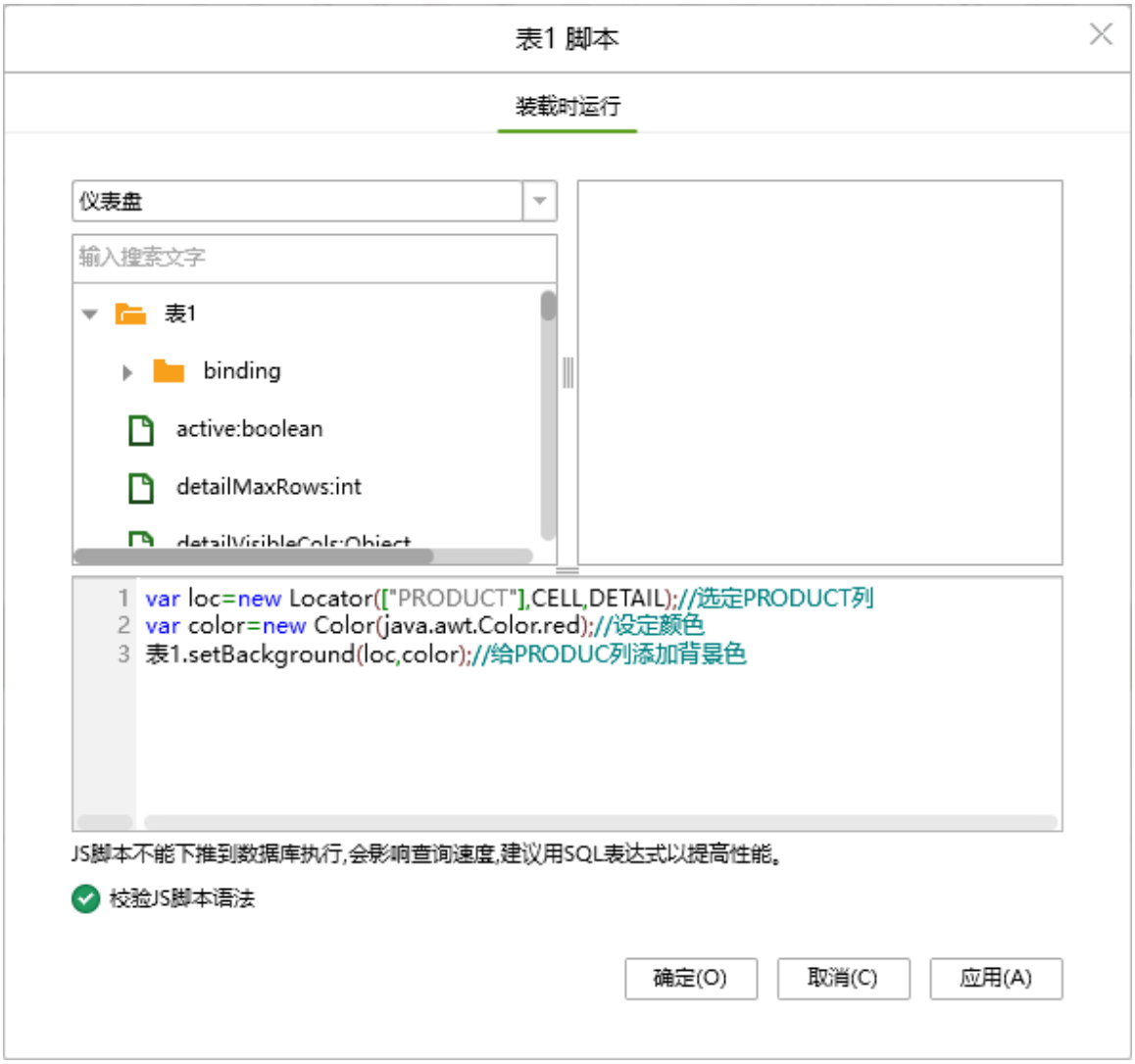
举例说明：

1. 存在一表组件，如下图所示。

咖啡销售统计	
PRODUCT	总和_SALES
Amaretto	26269
Caffe Latte	35899
Caffe Mocha	84904
Chamomile	75578
Columbian	128311
Darjeeling	73151
Decaf Espresso	78162
Decaf Irish Cream	62248
Earl Grey	66772
Green Tea	32850

2. 在仪表盘上右键选择脚本，打开脚本输入对话框，输入脚本：

```
var loc=new Locator(["PRODUCT"],CELL,DETAIL);// 选定 PRODUCT 列
var color=new Color(java.awt.Color.red);// 设定颜色
表 1.setBackground(loc,color);// 给 PRODUC 列添加背景色
```



3. 运行结果如下图所示。

咖啡销售统计	
PRODUCT	总和_SALES
Amaretto	26269
Caffe Latte	35899
Caffe Mocha	84904
Chamomile	75578
Columbian	128311
Darjeeling	73151
Decaf Espresso	78162
Decaf Irish Cream	62248
Earl Grey	66772
Green Tea	32850

第 7 章：表格渲染的脚本

表格渲染（Cell Renderer）可以将数据的变化通过画不同的图形来展示。是一种更为直观、方便的数据表现方式。该功能在表组件和交叉表组件及自由式表组件中支持。

本章介绍如何通过脚本来实现用图表来渲染格子中的数据。

打开方式

用户在表和交叉表及自由式表上的数据区域右键选择表格渲染，在打开的表格渲染对话框中选择动态渲染，则脚本处于激活状态，用户可输入脚本。

详细介绍

通过创建图表来进行表格渲染，主要包括三部分，第一部分是定义出图表的格式，第二部分是设定图表的数据源，第三部分是调用 getImage 函数生成图表。

定义图表格式

```
function createBMark(stack){
  var mark = new RangeBMark();
  mark.addDim("market");
  mark.addMeasure("sum_Sales");
  mark.setOpt(new StackMarkOpt());
  mark.setStack(stack);
  return mark;
}

function getBChart(){
  var axisType = CConstants.AXIS_BOTTOM_LEFT;// AXIS_UPPER_RIGHT
  var xscala = new DiscreteScala(new Array("market"));
  var aopt = new AxisOpt();
  aopt.setPType(axisType);
  var lbopt = new AxisLabelOpt();
  lbopt.setPType(CConstants.LABEL_OUTER);
  aopt.setLabelOpt(lbopt);
  xscala.setOpt(aopt);
  var yscala = new ContinuousScala(new Array("sum_Sales"));
  aopt = yscala.getOpt();
  aopt.setPType(axisType);
  var coord = new RectCoord(xscala, yscala);
  var mark = createBMark(false);
  var bchart = new BChart(coord, mark);
```

```
  return bchart;
}
var bc = getBChart();
```

设定图表的数据源

所画图表的数据源，在查询中用户可以根据需求使用 SQL 语句制作数据源。

1）数据必须是分组数据，使用 group by 来对数据进行分组

2）在数据的查询条件中可以添加参数，然后在 cell render 的脚本中给参数传值，从而使查询更加灵活、易用。

```
param['product'] = cols['Product'];
var data = execute(this,SQL,"Query_cellRender2");
```

调用 getImage 函数生成图表

表格宽度：cell.width

表格高度：cell.height

```
getImage(bc,data,cell.width,cell.height);
```

举例说明：

例如新建一个表如下所示：

日期	销售额	环比增长趋势
2009年01月	31,555	
2009年02月	32,092	0.017
2009年03月	32,245	0.005
2009年04月	32,943	0.022
2009年05月	33,692	0.023
2009年06月	35,125	0.043
2009年07月	36,161	0.029
2009年08月	36,029	-0.004
2009年09月	33,092	-0.082
2009年10月	32,849	-0.007
2009年11月	32,003	-0.026
2009年12月	33,373	0.043
2010年01月	35,316	0.058
2010年02月	34,192	-0.032
2010年03月	34,355	0.005
2010年04月	35,112	0.022
2010年05月	33,394	-0.049

如果用户需要将环比增长大于 0 的显示为向上的绿箭头，环比增长小于 0 的显示为向下的红箭头，则可以通过动态渲染来实现。

具体的操作为：先右击环比增长趋势列的单元格，选择表格渲染，在表格渲染对话框中选择动态渲染，输入脚本如下：

```
var data = cols. 环比增长趋势 ;

if(data<=0) {

url='file:///C:/warning/direction/arrow/arrow_bottom.png';// 本地图片

}

else if(data>0){

url='file:///C:/warning/direction/arrow/arrow_top.png';

}

else

{}
```

设置完成后，效果如图所示：

日期	销售额	环比增长趋势
2009年01月	31,555	
2009年02月	32,092	↑
2009年03月	32,245	↑
2009年04月	32,943	↑
2009年05月	33,692	↑
2009年06月	35,125	↑
2009年07月	36,161	↑
2009年08月	36,029	↓
2009年09月	33,092	↓
2009年10月	32,849	↓
2009年11月	32,003	↓
2009年12月	33,373	↑
2010年01月	35,316	↑
2010年02月	34,192	↓
2010年03月	34,355	↑
2010年04月	35,112	↑
2010年05月	33,394	↓

第 8 章：动态计算器的脚本

动态计算器指每个格子的数值是动态计算出来的，也可以称为是格间计算。主要解决在表格中沿着特定的方向来动态计算出数值。其中有两个要素，即计算方向和计算方法（或函数）。

计算方向就跟表的结构有关系。如果想做产品关于季度增量的环比计算，就需要把产品字段和季度字段作为分组，沿着季度字段来计算。我们把分组叫做分区 (Partition)。沿着什么方向叫做基于的方向（Addressing）。

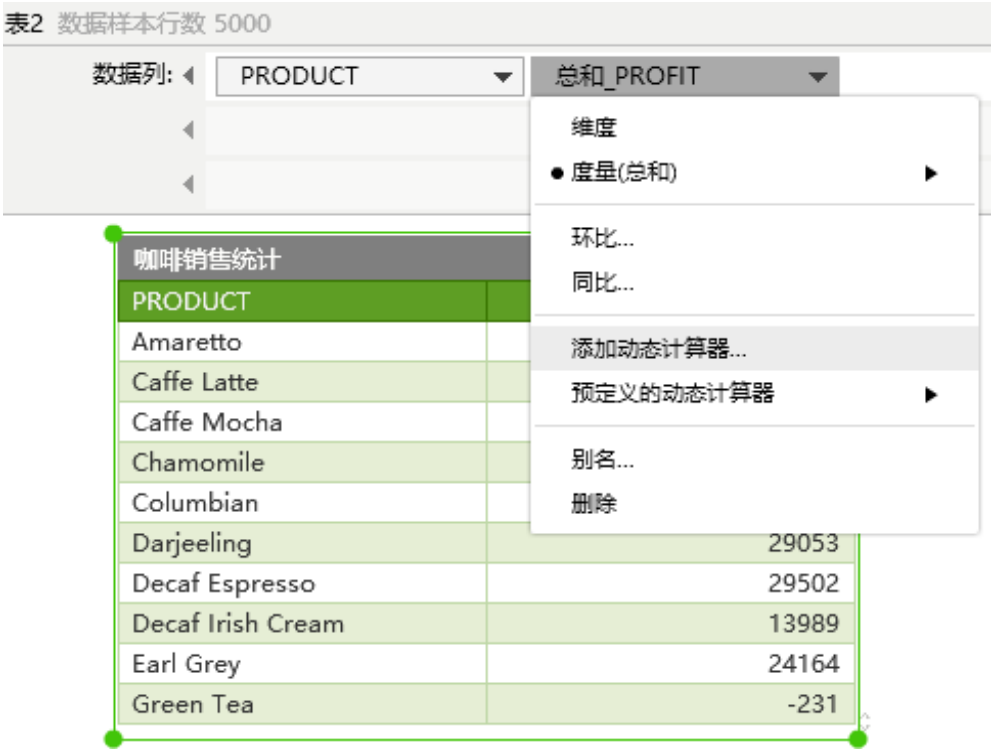
计算函数包括求差值，求百分比，求差值的百分比，求占总额的百分比，求累积计算和移动计算。每一种计算函数都有一些特殊的属性和参数。

用户也可以根据需要自定义计算函数和计算方向。对于自定义的计算函数，还可以使用二次计算的属性，即当前动态计算可使用其他动态计算的结果再做第二次计算。

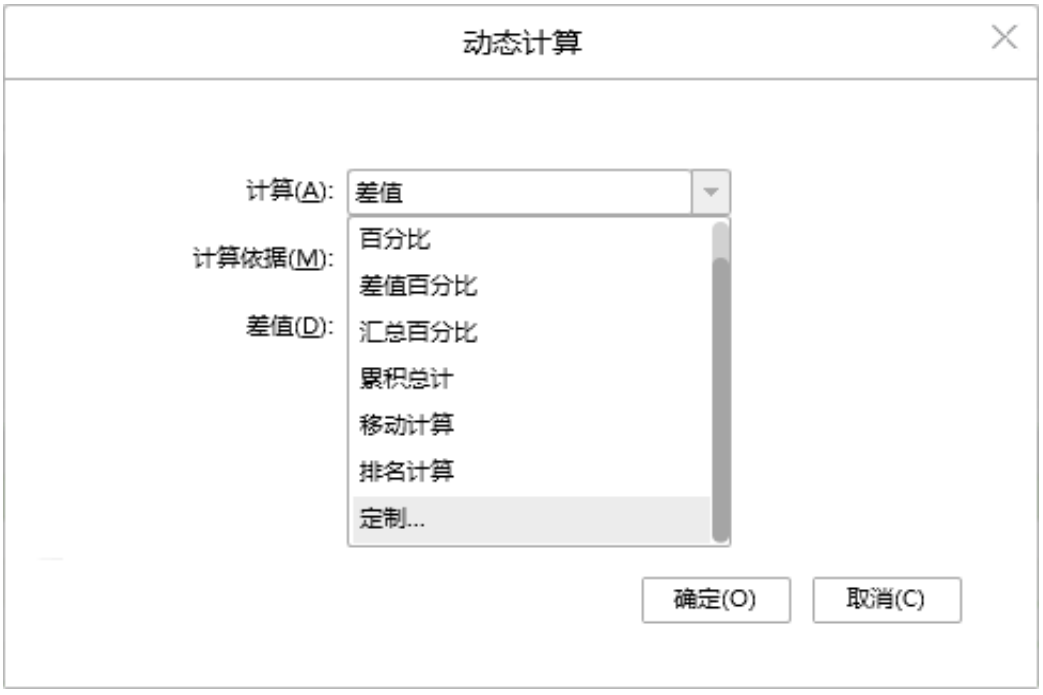
本章的主要是介绍如何通过脚本自定义计算函数。

打开脚本输入对话框

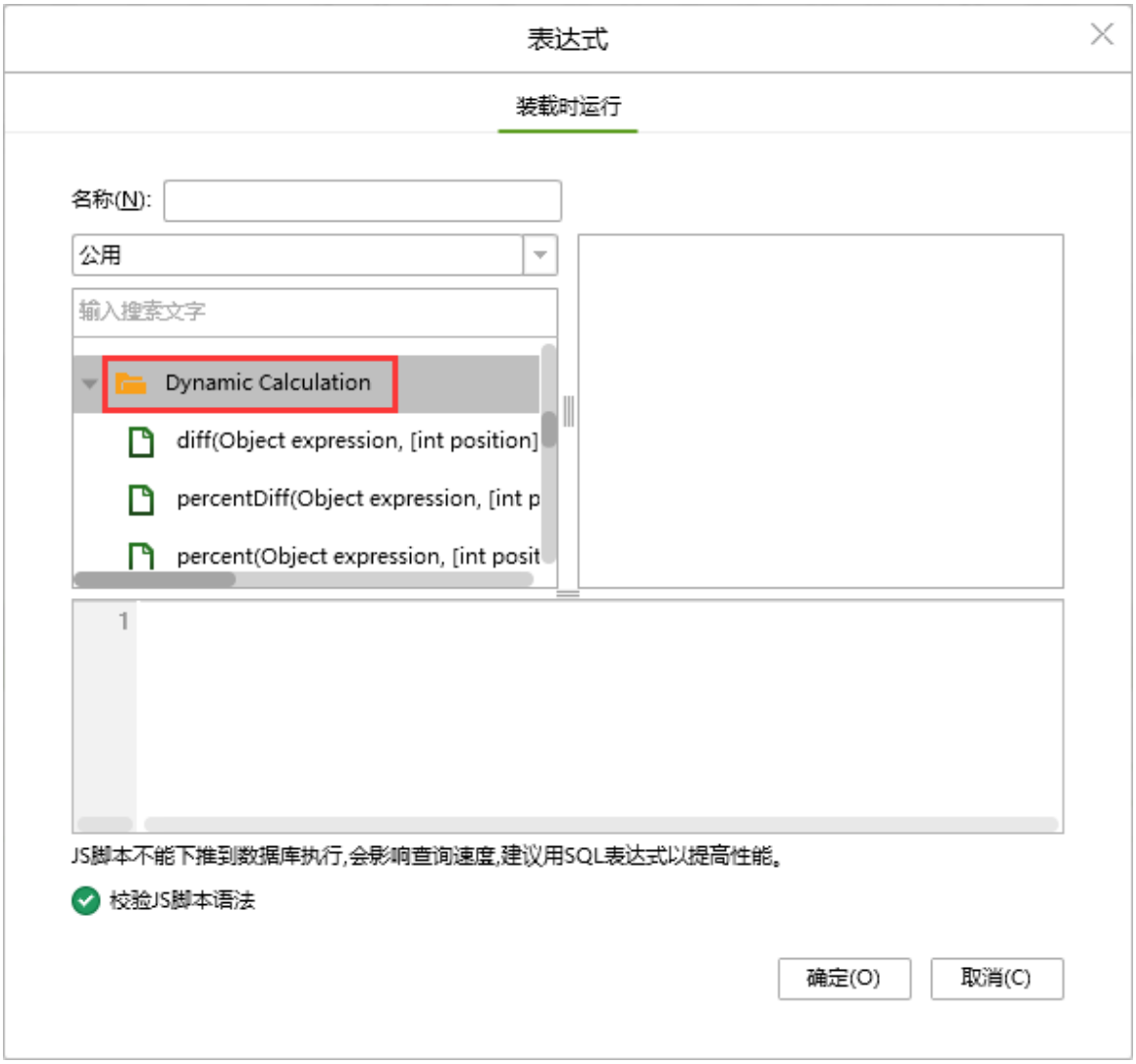
在表、交叉表、图表中的度量数据段的下拉列表中选择添加动态计算器，如下图所示。



在打开的动态计算器对话框中选择定制，如下图所示。



则打开脚本输入对话框，如下图所示。用户可调用 Dynamic Calculation 文件夹下提供的十五种动态计算函数。关于动态计算函数的详细介绍见动态计算脚本函数。



【名称】作为添加动态计算器脚本后数据段的名称。

当用户使用脚本进行动态计算，点击确定按钮后再次进入动态计算对话框，在此框中用户可通过计算基于选项来设定动态计算的方向。当用户需要再次编辑表达式时，点击编辑按钮，则可对表达式进行再次编辑。

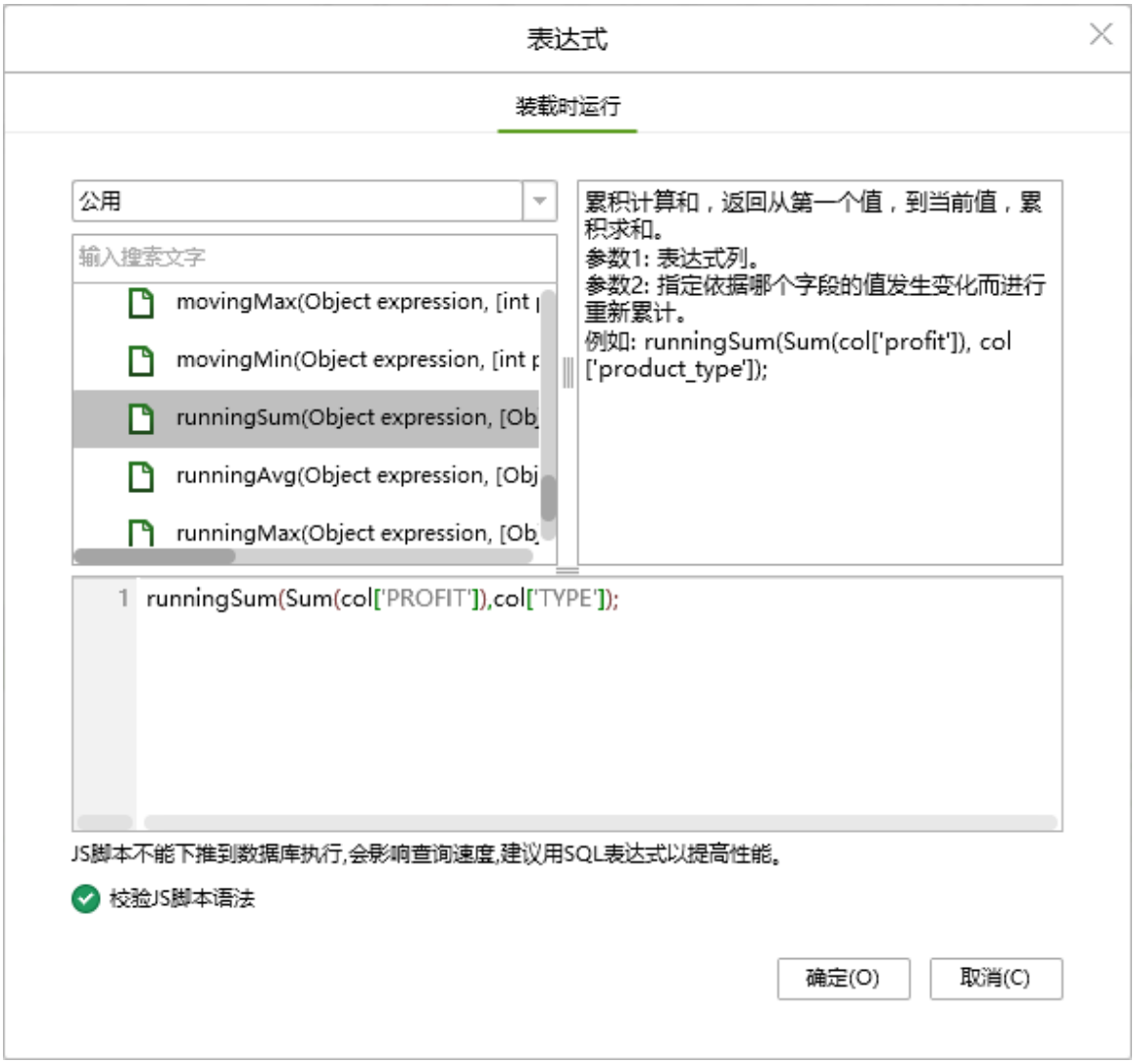
【二次计算】即在当前的表达式中存在其他脚本动态计算数据段名称。如已经存在一个脚本动态计算数据段，名称为 movingAVG，当用户在当前的表达式中再次使用 movingAVG 数据段，则属于第二次脚本动态计算，如表达式为 movingMin(col['movingAVG'],2,2,true,true); 时用户需要勾选二次计算选项。

举例说明：

存在一张表，如下图所示。

咖啡销售统计		
TYPE	MARKET	总和_PROFIT
Decaf	Central	43251
	East	11560
	South	14636
	West	37298
Regular	Central	50601
	East	47657
	South	17842
	West	36698

当用户需要统计每种类型下各市场的总利润累计时，可调用 runningSum() 函数，脚本语句为 runningSum(Sum(col['PROFIT']),col['TYPE']); 如下图所示。



计算结果如下图所示。

咖啡销售统计			
TYPE	MARKET	总和_PROFIT	runningsum
Decaf	Central	43251	43,251
	East	11560	54,811
	South	14636	69,447
	West	37298	106,745
Regular	Central	50601	50,601
	East	47657	98,258
	South	17842	116,100
	West	36698	152,798

第 9 章：常用的脚本函数参考列表

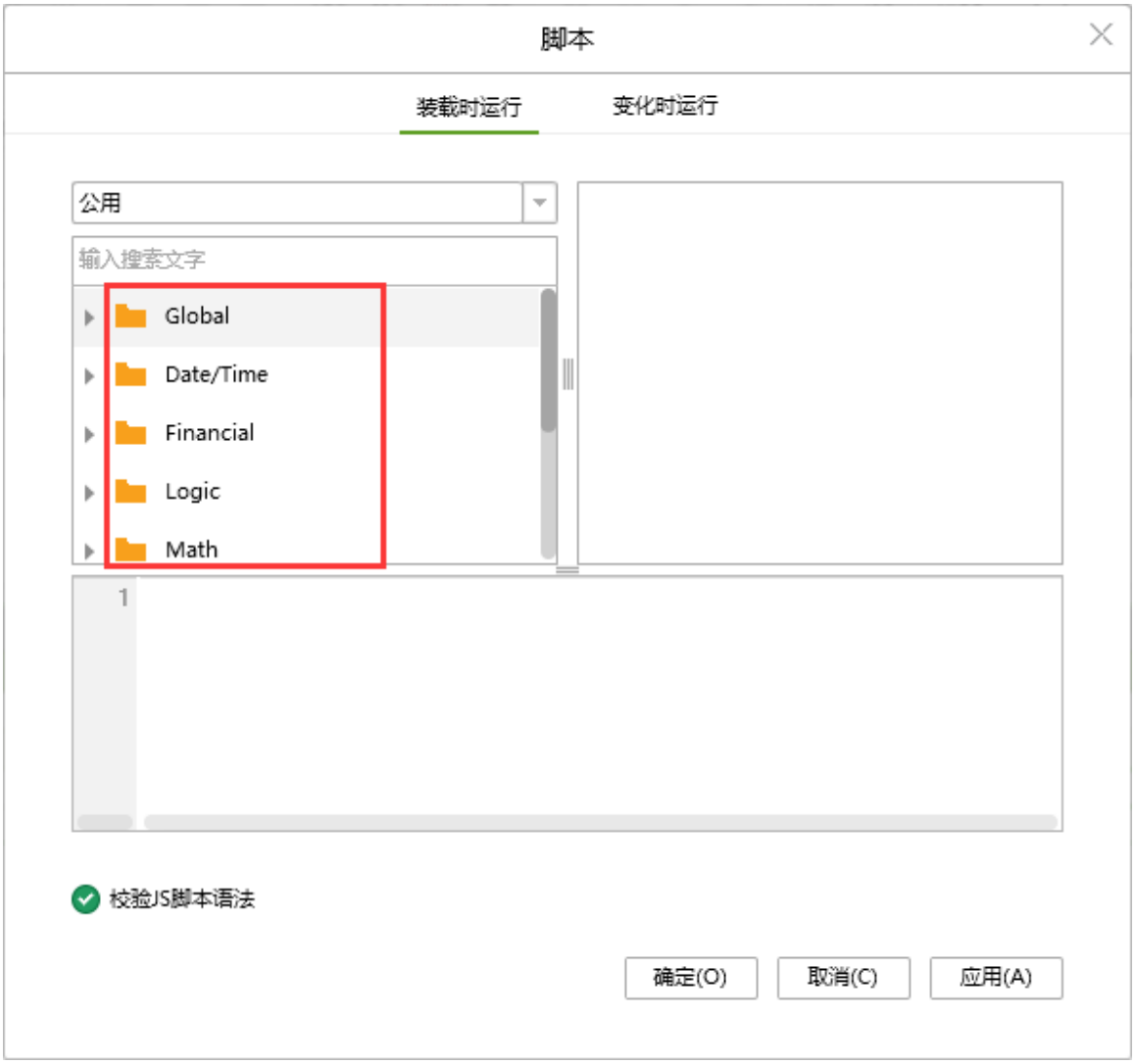
许多仪表盘环境需要动态定制仪表盘和实现自定义业务逻辑。这些需求从创建查询字段，到建立脚本查询；从修改文本颜色，到控制用户交互行为；从添加计算器，到控制表格渲染，等等。Yonghong Z-Suite 有一套完整的脚本环境体系，以支持用户的这种动态的需求。脚本体系是 Yonghong Z-Suite 产品中使用面较广的功能，让用户可以自己定制化一些高级需求，是决定产品是否强大的一个重要模块。

脚本系统是通过 JavaScript 的语言标准来支持的。脚本环境可以完全访问组件的绑定，属性，以及数据的输入；可以在仪表盘初始化的时候执行任务；可以处理用户交互行为。

本文从定制脚本的应用场景来分别介绍脚本的使用，包括查询的自定义字段，脚本查询，计算器，动态计算器，表格渲染，组件级别的脚本和仪表盘级别的脚本，等方面。

顶级作用域的函数介绍

顶级作用域的函数如下图所示。



下面对每个文件夹下的函数做详细介绍。

全局函数（Global）

函数	说明	举例
newInstance	创建一个对象实例。newInstance (String classname);	var a=newInstance(query);
isNull	检测对象是否为空。isNull (Object obj);	var a=isNull(query);
isNumber	检测对象是否是数值类型的。 isNumber(Object val);	var a=isNumber(query);
isDate	检测对象是否是日期类型的。 isDate(Object val)	var a=isDate(sell_date);
getWeek	从 Date 对象返回一年中的第几周 。	var d = new Date("July 10, 2012 01:15:00"); var a=getWeek(d);
getDate	返回一个 java 的日期	var d = new Date("July 21, 1983 01:15:00"); var a=getDate(d);
cloneDate	复制日期	var d = new Date("July 21, 1983 01:15:00"); var a=cloneDate(d);
formatDate	给日期设定显示格式。formatDate(Object val,String fmtstr);	var date= new Date("July 21, 1983 01:15:00"); var a=formatDate(date,"yyyy-MM-dd mm:hh");
parseDate	解析一个日期的字符串，并返回该日期距 1970 年 1 月 1 日午夜之间的毫秒数	var str = "1991-10-01"; parseDate(str, "yyyy-MM-dd");
dateAdd	dateAdd 用来给日期添加指定时间间隔	var date= new Date("July 21, 1983 01:15:00"); var d=dateAdd(date,"month",1);// 增加一月

dateGap	两个日期之间的时间差，支持 year /quarter/month/ weekofyear(dayofweek)/ dayofyear(dayofmonth) / hour minute /second	var date= new Date("July 21, 2012 01:15:00");
		var date2= new Date("June 10, 2012 01:15:00");
		var a =dateGap(date,date2,"month");

datePart	取出年、月、日等各部分的数值	var date= new Date("July 21, 1983 01:15:00"); var a=datePart(date,"year");// a=1983
split	把字符串分割为字符串数组。split(String str, String delim, Object limit);	var str="How are you doing today?"; var b=str.split(" "); 结果为 How,are,you,doing,today?
split2Array	将字符串分割为字符串数组。	var a ="How are you"; var b=split2Array(a,2);
formatNumber	函数可返回作为数字被格式化的表达式。	var a =formatNumber(498.8573945,"#,##0.##"); 就可以输出：498.86
toString	可把一个逻辑值转换为字符串，并返回结果。	var b=2.34; var a =toString(b);
substring	返回一个新的字符串，它是此字符串的一个子字符串。	var b=”Hello world”; var a =b.substring(0,3);
sqr	对数据求平方。 sqr(Object val);	var a=sqr(2);//a=4
sqrt	返回给定数据的平方根。 sqrt(9);	var a=sqrt(9);//a=3
abs	返回给定数据的绝对值。 abs(-7);	var a=abs(-7);//a=7
debug	向记录文件打印信息。 debug(Object msg);l	debug("msg");
execute	运行出一个查询。 execute（Scriptable script, int type, String path)//Scriptale script 是脚本的作用域， type 是查询的类型	var data=execute(this, Embed, "query1")；

preExecute	先预编译出一个查询 preExecute(Scriptable?script,int?type,?String?path, Object allcols)	var a=preExecute(this,SQL,"data",true);
		var b=executed(a);
		setData(" 表 1",b,DATA);

executed	运行出预编译出一个查询 executed(Object executedID)	var a=preExecute(this,SQL,"data",true);
		var b=executed(a);
		setData(" 表 1",b,DATA);

removeExecuted	删除预编译出的查询 removeExecuted(Object executedID)	var a=preExecute(this,SQL,"data",true);
		var b=removeExecuted(a);
		var c=executed(a);
		setData(" 表 1",c,DATA);

join	连接两个数据，将两个查询做连接。join(Scriptable scope, int jhint, int jop, Object jleft, Object jright, Object jlkeys, Object jrkeys, Object jlcols, Object jrcols)	例如：join(this, FINAL_JOIN | LEFT_MAIN, LEFT_JOIN, lt, rt, lkeys, rkeys, lcols, rcols);
union	合并两个数据网，将多个数据网做合并，求交集。union (Scriptable scope, Object ugrids)	Union(this, [a, b]);
columns	从数据网中提取出 N 列组成新的数据网。 columns (Scriptable scope, Object cgrid, Object ccols)	columns(this, query1, [3,1]);// 把 query1 中的第 3 和第 1 列取出形成新的数据网
sort	对数据网特定的数据段进行排序。 sort (Scriptable scope, Object sgrid, Object scols, Object sascs)	sort(this, query1, [3,1], [SORT_DESC, SORT_ASC]);// 先第 3 列降序，再第 1 列升序。
embed	数组转换为内嵌查询。 embed(Object val)	var a=[[" 编号 "," 2"," 3"],[" 姓 "," wang"," yao"]]; var b=embed(a); setData(" 表 1",b,DATA);

toArray	查询中的列转换为数组。
	toArray(Object gobj,Object cobj)
		var query1=execute(this,SQL, 'coffee3');
		var a=toArray(query1,"product");
		data=compare("Decaf Espresso",a[0]);

position	把经度和纬度转化为一个长整型来存储。 position(Object ox, Object oy)	position(10, 10);
putGlobal	全局常量。 putGlobal(String key,Object val)	putGlobal("a",1) var c=a;
toSQLDate	普通日期转换为 SQL 支持的时间戳	var a=new Date("January 12,2006 22:19:35"); var b=toSQLDate(a, DType.TIME).toString();
indexOf	返回某个指定的字符串值在字符串中首次出现的位置	var a="my hello world!" var b=a.indexOf("hello");
substring	返回某个指定位置的字符串的子集。	var a = "Hello World"; substring(a, 1, 3);
toDate	将后台取到的时间转换为另一种时间	var a=param["date"]; var b=toDate(a).toString().substring(0,10);

aggregates	对数据结果进行分组合计组成新的数据结果	var a = execute(this, SQL, "Coffee");
		var bcol1 = new BCol("TYPE", STRING, true);
		var bcol2 = new BCol("COGS", DOUBLE, false);
		var dimCol = new DimCol(bcol1);
		var meaCol = new MeasureCol(SUM, bcol2, null);
		var b = aggregates(this, a,[dimCol],[meaCol])
		setData(" 表 1", b, DATA);

日期时间函数 (Date/Time)

函数	说明	返回值类型	举例
date	返回特定日期的序列号	date	date(int year, int month, int day)
datevalue	将文本格式的日期转换为序列号，初始日期为 1900-1-1	int	datevalue(Object dateobj)
day	将序列号转换为月份日期	int	day(Object dateobj)
dayofyear	返回一年中的天数	int	dayofyear(Object dateobj)
days360	以一年 360 天为基准计算两个日期间的天数	int	days360(Object startobj, Object endobj, Object basis)
edate	返回用于表示开始日期之前或之后月数的日期的序列号	object	edate(Object dateobj, int number)
eomonth	返回指定月数之前或之后的月份的最后一天的序列号	object	eomonth(Object dateobj, int monthNo)
hour	将序列号转换为小时	int	hour(Object dateobj)
minute	将序列号转换为分钟	int	minute(Object dateobj)
month	将序列号转换为月	int	month(Object dateobj)
monthname	返回日期的月份	string	monthname(Object dateObj)
networkdays	返回两个日期间的全部工作日数	int	networkdays(Object startobj, Object endobj, Object holidaysObj)
now	返回当前日期和时间的序列号	date	now()
quarter	返回日期对象的季度 [1,4]	int	quarter(Object date)
second	将序列号转换为秒	int	second(Object dateobj)
time	返回特定时间的序列号	double	time(int hour, int minute, int second)
timevalue	将文本格式的时间转换为序列号	double	timevalue(Object dateobj)

today	返回今天日期的序列号	date	today()
weekday	将序列号转换为星期	int	weekday(Object dateobj, Object type)
weekdayname	返回当前日期是星期几	object	weekdayname(Object dateObject)
weeknum	将序列号转换为代表该星期为一年中第几周的数字	int	weeknum(Object dateobj, Object type)
workday	获取匹配工作日结束前或后的日期	object	workday(Object startobj, int days, Object holidaysObj)
year	将序列号转换为年	int	year(Object dateobj)
yearfrac	返回 start_date 和 end_date 之间的天数占全年天数的百分比。	double	yearfrac(Object start, Object end, Object basis)

金融 (Financial)

函数	说明	返回值类型	举例
accrint	返回定期付息证券的应计利息	double	double accrint(Object issue, Object firstIneterest, Object settlement, double rate, double par, double frequency, double basis)
accrintm	返回在到期日支付利息的债券的应计利息	double	accrintm(Object issue, Object maturity, double rate, double par,double basis)
amordegrc	返回使用折旧系数的每个记帐期的折旧值	double	amordegrc(double cost, Object date_purchased, Object first_period, double salvage, int period, double rate, intbasis)
amorlinc	返回每个记帐期的折旧值	double	amorlinc(double cost, Object date_purchased, Object first_period, double salvage, int period, double rate, int basis)
coupdaybs	返回从付息期开始到成交日之间的天数	double	coupdaybs(Object settlement, Object maturity, double frequency, int basis)
coupdays	返回在优惠期的天数，包括结算日期	int	coupdays(Object settlement, Object maturity, double frequency, double basis)
coupdaysnc	返回从成交日到下一付息日之间的天数	double	coupdaysnc(Object settlement, Object maturity, double frequency, double basis)
coupncd	返回成交日之后的下一个付息日	date	coupncd(Object settlement, Object maturity, double frequency, double basis)
coupnum	返回成交日和到期日之间的应付利息次数	int	coupnum(Object settlement, Object maturity, double frequency, double basis)

couppcd	返回成交日之前的上一付息日	date	couppcd(Object settlement, Object maturity, double frequency, double basis)
cumipmt	返回两个付款期之间累积支付的利息	double	cumipmt(double rate, int nper, double pv, double start,double end, int type)
cumprinc	返回两个付款期之间为贷款累积支付的本金	double	cumprinc(double rate, int nper, double pv, double start, double end,int type)
financialDb	使用固定余额递减法，返回一笔资产在给定期间内的折旧值	double	financialDb (double cost, double salvage, double life, double period, double month)
ddb	使用双倍余额递减法或其他指定方法，返回一笔资产在给定期间内的折旧值	double	ddb(double cost, double salvage, double life, double period, double factor)
disc	返回债券的贴现率	double	disc(Object settlement, Object maturity, double pr, double redemption, double basis)
duration	返回定期支付利息的债券的每年期限	double	duration(Object settlement, Object maturity, double coupon, double yld, double frequency, double basis)
effect	返回年有效利率	double	effect(double nominal_rate, double npery)
fv	返回一笔投资的未来值	double	fv(double rate, int nper, double pmt, double pv, int type)
fvschedule	返回应用一系列复利率计算的初始本金的未来值	double	fvschedule(double principal, Object scheduleObj)
intrate	返回完全投资型债券的利率	double	intrate(Object settlement, Object maturity, double investment,double redemption, double basis)
ipmt	返回一笔投资在给定期间内支付的利息	double	ipmt(double rate, int per, int nper, double pv, double fv, int type)

ispmt	计算特定投资期内要支付的利息	double	ispmt(double rate, int per, int nper, double pv)
mduration	返回假设面值为 ¥100 的有价证券的 Macauley 修正期限	double	mduration(Object settlement, Object maturity, double coupon, double yld,double frequency, double basis)
mirr	返回正和负现金流以不同利率进行计算的内部收益率	String	mirr(Object valuesObj, double finance_rate, double reinvest_rate)
nominal	返回年度的名义利率	double	nominal(double rate, double npery)
nper	返回投资的期数	double	nper(double rate, double pmt, double pv, double fv, int type)
npv	返回基于一系列定期的现金流和贴现率计算的投资的净现值	double	npv(double rate, Object valuesObj)
pmt	返回年金的定期支付金额	double	pmt(double rate, int nper, double pv, double fv, int type)
ppmt	返回一笔投资在给定期间内偿还的本金	double	ppmt(double rate, int per, int nper, double pv, double fv, int type)
price	返回每张票面为 ¥100 且定期支付利息的债券的现价	double	price(Object settlement, Object maturity, double rate, double yield,double redemption, double frequency, int basis)
pricedisc	返回每张票面为 ¥100 的已贴现债券的现价	double	pricedisc(Object settlement, Object maturity, double discount,double redemption, double basis)
pricemat	返回每张票面为 ¥100 且在到期日支付利息的债券的现价	double	pricemat(Object settlement, Object maturity, Object issue, double rate,double yield, double basis)
pv	返回投资的现值	double	pv(double rate, int nper, double pmt, double fv, int type)

received	返回完全投资型债券在到期日收回的金额	double	received(Object settlement, Object maturity, double investment, double discount, double basis)
sln	返回固定资产的每期线性折旧费	double	sln(double cost, double salvage, int life)
syd	资产在某一特定时期内年数总和折旧值	double	syd(double cost, double salvage, double life, double per)
tbilleq	返回国库券的等价债券收益	double	tbilleq(Object settlement, Object maturity, double discount)
tbillprice	返回面值 ￥100 的国库券的价格	double	tbillprice(Object settlement, Object maturity, double discount)
tbillyield	返回国库券的收益率	double	tbillyield(Object settlement, Object maturity, double par)
vdb	使用余额递减法，返回一笔资产在给定期间或部分期间内的折旧值	double	vdb(double cost, double salvage, double life, double start_period,double end_period, double factor, boolean flag)
xnpv	返回一组现金流的净现值，这些现金流不一定定期发生	double	xnpv(double rate, Object valuesObj, Object datesObj)
yielddisc	返回已贴现债券的年收益；例如，短期国库券	double	yielddisc(Object settlement, Object maturity, double pr, double redemption,double basis)
yieldmat	返回在到期日支付利息的债券的年收益	double	yieldmat(Object settlement, Object maturity, Object issue, double rate, double pr, double basis)

逻辑函数 (Logic)

函数	说明	返回值类型	举例
and	与，如果其所有参数均为 TRUE，则返回 TRUE	boolean	and(Object param1, Object param2, Object param3, Object param4)
or	或，如果任一参数为 TRUE，则返回 TRUE	boolean	or(Object param1, Object param2, Object param3, Object param4)
not	非，对其参数的逻辑求反	boolean	not(Object logic)
iif	可选的判断，如果你指定一个条件判断为 TRUE 并返回一个值；如果为 FALSE 则返回另一个值。	object	iif(Object condition, Object trueval, Object falseval)

数学函数 (Math)

函数	说明	返回值类型	举例
abs	返回参数的绝对值	double	abs(double val)
acos	反余弦函数 , 参数范围 [-1， 1]	double	acos(double val)
acosh	反双曲余弦值	double	acosh(double val)
asin	反正弦函数，参数范围 [-1,1]	double	asin(double val)
asinh	反双曲正弦值	double	asinh(double val)
atan	反正切函数	double	atan(double val)
atan2	返回指定坐标轴 x， y 轴的反正切值	double	atan2(double x,double y)
atanh	反双曲正切值 , 参数范围（ -1,1 ）	double	atanh(double val)
ceiling	返回参数 val 向上舍入（沿绝绝对值增大的方向）为最接近的 significance 的倍数	double	celling(double val, double significance)
combin	计算从给定数目的对象集合中提取若干对象的组合数	double	combin(double val, double chosen)
cos	返回参数的余弦值	double	cos(double val)
cosh	双曲余弦函数	double	cosh(double val)
degrees	函数返回弧度角的度数	double	degrees(double val)
even	函数返回沿绝对值增大方向取整后最接近的偶数	double	even(double val)
exp	函数返回以 e 为底的指数值	double	exp(double val)
fact	返回参数的阶乘值	double	fact(double val)
factdouble	返回参数阶乘的双倍值	double	factdouble(double val)

floor	返回参数 val 沿绝对值减小的方向向下舍入，使其等于最接近的 Significance 的倍数	double	floor(double val, double significance)
gcd	返回两个或多个整数的最大公约数。	int	gcd(Object numbersObj)
integer	对数进行取整	int	integer(double val)
lcm	返回整数参数的最小公倍数	int	lcm(double val)
ln	返回参数的自然对数	double	ln(double val)
log	返回以 base 为底的对数值	double	double log(double val, double base)
log10	返回以 10 为底的对数值	double	log10(double val)
mdeterm	返回一个数组的矩阵行列式的值	double	mdeterm(Object valuesObj)
mod	返回两数相除的余数	double	mod(double val, double divisor)
mround	返回参数按指定基数舍入后的数值	double	mround(double val, double multiple)
multinomial	获取参数和的阶乘除以参数阶乘的乘积，如 a1 + a2+...+an)!/(a1!*a2!....an!)	double	multinomial(Object valuesObj)
odd	返回对指定数值进行向上舍入后的奇数	int	odd(double val)
pi	返回 PI 值	double	pi()
power	返回给定数字的乘幂	double	power(double val, double power)
mathproduct	返回有以参数形式给出的数字相乘，并返回乘积值。	double	mathproduct(Object valuesObj)
quotient	返回 (int)(value/denominator) 的商	int	quotient(double value,double denominator)
radians	返回度数的弧度角	double	radians(double angle)
rand	返回大于等于 0 及小于 1 的均匀分布随机数，范围 [0,1)	double	rand()

randbetween	返回大于等于指定的最小值，小于指定最大值之间的一个随机整数	int	randbetween(int start, int end)
round	返回某个数字按指定位数取整后的数字	double	round(double val, int accurate)
rounddown	按指定位数舍去数字指定位数后面的小数	double	rounddown(double val, int accurate)
roundup	按指定位数向上舍入指定位数后面的小数	double	roundup(double val, int accurate)
seriessum	返回幂级数的和	double	seriessum(double x, double n, double m, Object coefficientsObj)
sign	返回数字的符号。当数字为正数时返回 1，为零时返回 0，为负数时返回 -1	int	sign(double val)
sin	返回参数的正弦值	double	sin(double val)
sinh	返回参数的双曲正弦值	double	sinh(double val)
sqrt	返回参数的平方根	double	double sqrt(double val)
sqrtpi	返回参数与 pi 的乘积的平方根	double	sqrtpi(double val)
subtotal	返回数据清单或数据库中的分类汇总，function_id 为 [1,11] 的功能值常量	double	subtotal(int function_id, Object valuesObj)
mathSum	返回参数的和	double	mathSum(Object valuesObj)
sumif	根据指定条件对若干单元格求和	double	sumif(Object valuesObj, String criteria, Object sumValuesObj)
sumproduct	在给定的几组数组中，将数组间对应的元素相乘，并返回乘积之和。	double	sumproduct(Object arraysObj)
mathSumsq	返回参数的平方和	double	mathSumsq(Object valuesObj)
sumx2py2	返回两数组中对应数值的平方差之和	double	sumx2py2(Object arrayObj1, Object arrayObj2)

sumxmy2	返回两个数组中对应数值之差的平方和	double	sumx2py2(Object arrayObj1, Object arrayObj2)
sumx2my2	返回两个数组中对应数值的平方差之和	double	sumx2my2(Object arrayObj1, Object arrayObj2)
tan	返回参数的正切值	double	tan(double val)
tanh	返回参数的双曲正切	double	tanh(double val)
trunc	将数字的小数部分截去，返回整数	double	trunc(double val)

统计函数 (Statistic)

函数	说明	返回值类型	举例
avedev	返回数据点与它们的平均值的绝对偏差平均值	double	avedev(Object valuesObj)
average	返回其参数的平均值	double	average(Object valuesObj)
averagea	返回其参数的平均值，包括数字、文本和逻辑值	double	averagea(Object valuesObj)
averageif	返回区域中满足给定条件的所有单元格的平均值（算术平均值）	double	averageif(Object valuesObj, String criteria, Object avgValuesObj)
binomdist	返回一元二项式分布的概率值	double	binomdist(double counts, double trials, double successP,boolean cummulative)
correl	返回两个数据集之间的相关系数	double	correl(Object valuesObj1, Object valuesObj2)
statisticCount	计算参数列表中参数的个数	int	statisticCount(Object valuesObj)
counta	计算参数列表中非空值的个数	int	counta(Object valuesObj)
countblank	计算区域内空白单元格的数量	int	countblank(Object valuesObj)
countdistinct	计算参数列表中非重复数字的个数	int	countdistinct(Object valuesObj)
countif	计算区域内符合给定条件的单元格的数量（目前只支持数字的表达式的形式）	int	countif(Object rangeObj, String criteria)
countn	返回数组中数字的个数	int	countn(Object valuesObj)
devsq	返回偏差的平方和	double	devsq(Object valuesObj)
expondist	返回指数分布	double	expondist(double x, double lambda, boolean cumulative)

fisher	返回 Fisher 变换值	double	fisher(double value)
fisherinv	返回 Fisher 变换的反函数值	double	fisherinv(double value)
forecast	返回沿线性趋势的值	double	forecast(double x, Object yObj, Object xObj)
frequency	以垂直数组的形式返回频率分布	int	frequency(Object datasObj, Object rangesObj)
geomean	返回几何平均值	double	geomean(Object valuesObj)
harmean	返回调和平均值	double	harmean(Object valuesObj)
hypgeomdist	返回超几何分布	double	hypgeomdist(double number_succ, double number_sample,double pop_succ, double number_pop, boolean cumulative)
intercept	返回线性回归线的截距	double	intercept(Object knownYObj, Object knownXObj)
kurt	返回数据集的峰值	double	kurt(Object valuesObj)
large	返回数据集中第 k 个最大值	double	large(Object valuesObj, int k)
statisticMax	返回参数列表中的最大值	double	statisticMax(Object valuesObj)
maxa	返回参数列表中的最大值，包括数字、文本和逻辑值	double	maxa(Object valuesObj)
statisticMedian	返回给定数值集合的中值	double	statisticMedian(Object valuesObj)
statisticMin	返回参数列表中的最小值	double	statisticMin(Object valuesObj)
mina	返回参数列表中的最小值，包括数字、文本和逻辑值	double	mina(Object valuesObj)
statisticMode	返回在数据集内出现次数最多的值	double	statisticMode(Object valuesObj)
negbinomdist	返回负二项式分布	double	negbinomdist(double nb_fail, double nb_succ, double pro_succ)

pearson	返回 Pearson 乘积矩相关系数	double	pearson(Object valuesObj1, Object valuesObj2)
percentile	返回区域中数值的第 k 个百分点的值	double	percentile(Object valuesObj, double k)
percentrank	返回数据集中值的百分比排位	double	percentrank(Object valuesObj, double target, double significance)
permut	返回给定数目对象的排列数	double	permut(double number, double chosen)
poisson	返回泊松分布	double	poisson(double x, double mean, boolean cumulative)
prob	返回区域中的数值落在指定区间内的概率	double	prob(Object rangeObj, Object probRangeObj, double llimit, double uplimit)
statisticQuartile	返回一组数据的四分位点	double	statisticQuartile(Object valuesObj, double quart)
rank	返回一列数字的数字排位	int	rank(double number, Object valuesObj, double order)
rsq	返回 Pearson 乘积矩相关系数的平方	double	rsq(Object arrayObj1, Object arrayObj2)
skew	返回分布的不对称度	double	skew(Object valuesObj)
slope	返回线性回归线的斜率	double	slope(Object yObjs, Object xObjs)
small	返回数据集中的第 k 个最小值	double	small(Object valuesObj, int k)
standardized	返回正态化数值	double	standardize(double x, double mean, double standard_dev)
stdev	基于样本估算标准偏差	double	stdev(Object valuesObj)
stdeva	基于样本（包括数字、文本和逻辑值）估算标准偏差	double	stdeva(Object valuesObj)
stdevp	基于整个样本总体计算标准偏差	double	stdevp(Object valuesObj)
stdevpa	基于总体（包括数字、文本和逻辑值）计算标准偏差	double	stdevpa(Object valuesObj)

steyx	返回通过线性回归法预测每个 x 的 y 值时所产生的标准误差	double	steyx(Object yObjs, Object xObjs)
trimmean	返回数据集的内部平均值	double	trimmean(Object valuesObj, double percent)
varn	基于样本估算方差（对应 Excel 中的 var，但是 var 是 script 中的关键字，固软件中改为ｖａｒｎ）	double	varn(Object valuesObj)
vara	基于样本（包括数字、文本和逻辑值）估算方差	double	vara(Object valuesObj)
varp	计算基于样本总体的方差	double	varp(Object valuesObj)
varpa	计算基于总体（包括数字、文本和逻辑值）的标准偏差	double	varpa(Object valuesObj)
weightedavg	返回两个数组的加权平均值	double	weightedavg(Object array1Obj, Object array2Obj)
weibull	返回威布尔概率密度	double	weibull(double x, double alpha, double beta, boolean cumulative)

文本 (Text)

函数	说明	返回值类型	举例
character	返回由代码数字指定的字符	char	character(int value)
code	返回文本字符串中第一个字符的数字代码	int	code(String str)
concatenate	将几个文本项合并为一个文本项	string	concatenate(Object strObj)
currency	将数字转换为文本格式，并适用于默认货币符号。	string	currency(double number, int decimals)
dollar	将数字转换为文本格式，并适用 $ 符号。	string	dollar(double number, int decimals)
exact	比较两个字符串	boolean	exact(String str1, String str2)
find	在一个文本值中查找另一个文本值（区分大小写）	int	find(String find, String src, int start)
fixed	将数字格式设置为具有固定小数位数的文本	string	fixed(double number, int decimals, boolean nocommas)
left	返回文本值中最左边的字符	string	left(String src, int length)
len	返回文本字符串中的字符个数	int	len(String text)
lower	将文本转换为小写	string	lower(String text)
mid	从文本字符串中的指定位置起返回特定个数的字符	string	mid(String src, int start, int length)
proper	将文本值的每个字的首字母大写	string	proper(String text)
replace	替换文本中的字符	string	replace(String src, int start, int length, String replaced)
rept	按给定次数重复文本	string	rept(String src, int times)
right	返回文本值中最右边的字符	string	right(String src, int length)

search	在一个文本值中查找另一个文本值（不区分大小写	int	search(String find, String src, int start)
substitute	在文本字符串中用新文本替换旧文本	string	substitute(String src, String old, String newtxt, int times)
t	将参数转换为文本	string	t(String value)
text	设置数字格式并将其转换为文本	string	text(double value, String patten)
trim	删除文本中没用的空格	string	trim(String str)
upper	将文本转换为大写形式	string	upper(String str)

脚本函数性能说明

目前，关于表达式的处理部分，一部分表达式使用产品自带的处理引擎，一部分表达式使用 js 引擎（jsEngine）来处理。一个表达式，既有系统支持的表达式，又有系统不支持的表达式，则还是会用 js 引擎（jsEngine）解析。

在处理大数据量的集市数据时，由 jsEngine 处理的表达式效率要比产品自身支持的表达式的效率低，因此为了提高表达式的处理效率，对产品自身支持的表达式范围进行了扩展，主要在产品支持的表达式中增加了对以下一些函数的支持，当表达式中包含以下的函数时，将由产品自带的处理引

函数	说明
right	返回文本值中最右边的字符
left	返回文本值中最左边的字符
len	返回文本字符串中的字符个数
replace	替换文本中的字符
trim	去掉字符串两边的空格
upper	将文本转换为大写形式
lower	将文本转换为小写形式
weekdayname	返回当前日期是星期几
year	返回日期的年份
month	返回日期的月份
parseDate	把字符串解析成日期
formatDate	把日期格式化成字符串
position	把经度和纬度转化为一个长整型来存储
substring	返回一个新的字符串，它是此字符串的一个子字符串
sqr	对数据求平方
sqrt	返回给定数据的平方根
abs	返回给定数据的绝对值
indexOf	返回某个指定的字符串值在字符串中首次出现的位置

第 10 章：永洪脚本对象参考列表

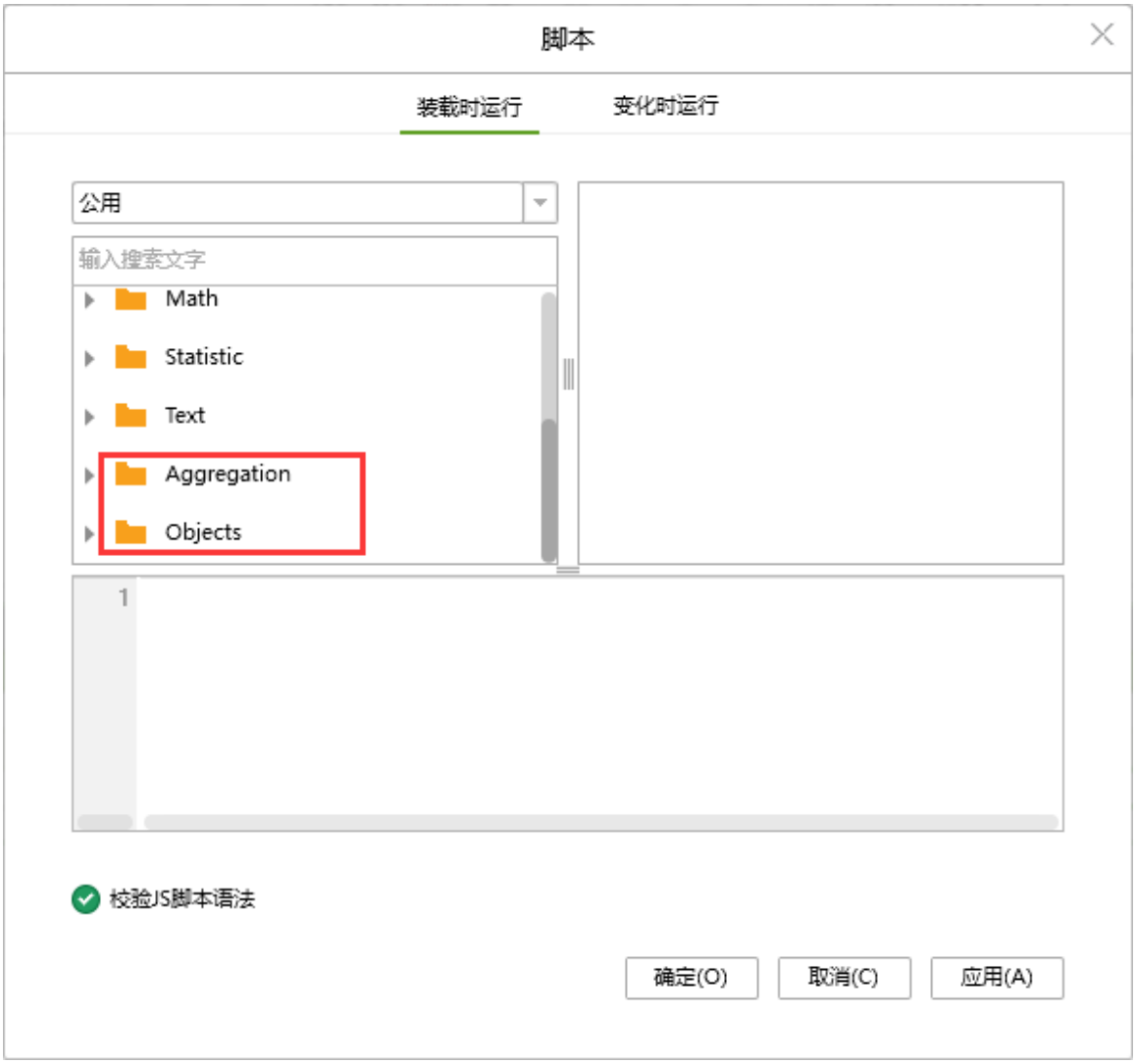
许多仪表盘环境需要动态定制仪表盘和实现自定义业务逻辑。这些需求从创建查询字段，到建立脚本查询；从修改文本颜色，到控制用户交互行为；从添加计算器，到控制表格渲染，等等。Yonghong Z-Suite 有一套完整的脚本环境体系，以支持用户的这种动态的需求。脚本体系是 Yonghong Z-Suite 产品中使用面较广的功能，让用户可以自己定制化一些高级需求，是决定产品是否强大的一个重要模块。

脚本系统是通过 JavaScript 的语言标准来支持的。脚本环境可以完全访问组件的绑定，属性，以及数据的输入；可以在仪表盘初始化的时候执行任务；可以处理用户交互行为。

本文从定制脚本的应用场景来分别产生脚本的使用，包括查询的自定义字段，脚本查询，计算器，动态计算器，表格渲染，组件级别的脚本和仪表盘级别的脚本，等方面。

顶级作用域的函数介绍

Yonghong Z-Suite 提供的顶级作用域的函数如下图所示。



下面对每个文件夹下的函数做详细介绍。

聚合函数（Aggregation）

函数	说明	举例
Sum	返回数据集中所有数据之和	Sum(col['sales'])
Avg	返回数据集中的平均值	Avg(col['sales'])
Count	返回数据集中的数据个数	Count(col['product'])
DistinctCount	返回数据集中不同值的数目	DistinctCount(col['product'])
Max	返回数据集中的最大数值	Max(col['product'])
Min	返回数据集中的最小值	Min(col['product'])
Range	返回数据集的范围	Range(col['sales'])
Product	返回一组数据的乘积	Product(col['sales'])
Median	返回给定数值集合的中位数	Median(col['sales'])
Mode	返回在某一数组或数据区域中的众数	Mode(col['sales'])
SumSQ	返回一组数据的平方和	SumSQ(col['sales'])
Quartile	返回一组数据的四分位点	Quartile(col['sales'], 4) //quart 是 0,1,2,3,4
PthPercentile	返回数值区域的 P 百分比数值点	PthPercentile(col['sales'], 10) //p 是 0—100. 百分比
Variance	返回一组数据的方差	Variance(col['sales'])
PopulationVariance	返回一组数据的总体方差	PopulationVariance(col['sales'])
StandardDeviation	返回一组数据的标准差	StandardDeviation(col['sales'])
StandardError	返回一组数据的标准误差	StandardError(col['sales'])
PopulationStandardDeviation	返回一组数据的总体标准差	PopulationStandardDeviation (col['sales'])
SumWT	返回一组数据的权重之和	SumWT(col['sales'], col['profit'])
WeightAvg	返回一组数据的权重的均值	WeightAvg(col['sales'], col['profit'])
Covariance	返回一组数据的协方差	Covariance(col['sales'], col['profit'])
Correlation	返回一组数据的相关系数	Correlation(col['sales'], col['profit'])

对象（Object）

BCol

创建一个底层字段，包含三个属性，名称、字段类型、是否是维度字段。如果字段存在别名的话则需要添加的属性的格式应该是：名称、别名、字段类型、是否是维度字段。

举例：

var b=new BCol("City",STRING,true);// BCol(String name, int type, boolean dim)

用户还可写成 var b=new BCol("City",2,true);

用户可通过 b.name 的形式来引用成员的属性。

如果有别名的情况：

var b=new BCol(times," 感染次数 ",LONG,false);//BCol(String name, String view, int type, boolean dim)

在本产品中数据类型也可以用常量来替换，对应关系如下表。

字段类型	相应的整数
STRING	2
BOOLEAN	3
FLOAT	4
DOUBLE	5
DECIMAL	6
CHAR	7
BYTE	8
SHORT	9
INTEGER	10
LONG	11
DATE_TIME	12
DATE	13
TIME	14
CHARACTER	15

DimCol

创建维度字段。

col: 获取底层字段 ;

sortRank: 设置高级排序中 Top N;

sortType: 设置排序类型 ;

sortGroupOthers: 设置排序区域 ;

sortBy: 设置高级排序字段 ;

showTotal: 设置组的分组合计 ;

groupSpan: 设置组的单元格是否合并 ;

举例 :

表达式：var a = new DimCol(bcol); // var bcol = new BCol("reseller", BOOLEAN, true);

假如有一个表绑定字段如下图所示：

1	
MARKET +	总和_AREA_CODE
Central +	775,889
East +	551,756
South +	364,466
West +	781,406

在表的脚本对话框中设置脚本如下：

```
var bcol = new BCol("TYPE", STRING, true);
var d = new DimCol(bcol);
d.sortType=2;
d.sortGroupOthers=true;
d.showTotal=true;// 设置分组合计，默认 为后置
d.groupSpan=true;
表 1.binding.addCol(0,d);
```

则结果如下图所示，表增加 “TYPE” 列，并且为降序排列，显示 TYPE 列的分组合计：

1		
TYPE ↓	MARKET +	总和_AREA_CODE
Regular	Central +	409,668
	East +	370,403
	South +	160,182
	West +	460,430
	小计	**1,400,683**
Decaf	Central +	366,221
	East +	181,353
	South +	204,284
	West +	320,976
	小计	**1,072,834**

MeasureCol

创建度量字段。

创建字段表达式：MeasureCol(total, col, col2);// 没有 col2 时设置为 unll

total: 统计函数；

col: 是需要做统计函数的度量字段；

col2: 做统计时需要的第二字段，对于部分函数需要，例如计算数据的权重之和函数 SumWT(col['sales'], col['profit']) ；

sortType: 设置字段排序类型；

view: 如果字段存在别名，设置上字段的别名；

举例：

```
var a = new MeasureCol(SUM,bcol,null);

// var bcol = new BCol("price", DOUBLE, false);
```

DateCol

创建日期类型字段。

col: 一个字段；

level：日期层次，是年，季度，月，周等；

创建字段表达式：DateCol(QCol col, int level);

QCol 包括 BCol、 DimCol、 MeasureCol、 ChartDimCol、 ChartMeasureCol

举例：

```
var b = new DateCol(k); // 假设 k = new g5.meta. DateCol();

var bcol = new BCol("orderdate", DATE, true);

var c=new DataCol(bcol,QUARTER_GROUP);
```

日期层次 (level)

区域类型	相应的整数	说明
YEAR	33	按年分组
QUARTER_GROUP	2	按季度分组
MONTH_GROUP	3	按月份分组
WEEK_GROUP	4	按周分组
DAY_GROUP	5	按天分组
HOUR_GROUP	6	按小时分组
MINUTE_GROUP	7	按分钟分组
SECOND_GROUP	8	按秒分组

日期层次 (level)

QUARTER	34	季度
MONTH	35	月份
WEEK	36	星期
DAY	37	天
DAY_OF_WEEK	41	返回当前日期是周几
HOUR	38	小时
MINUTE	39	分钟
SECOND	40	秒

ChartDimCol

创建图表所用的维度字段。

BCol 是最底层的字段，而 QCol 比 BCol 的属性更多，含有了轴的属性。

举例：

```
var b = new ChartDimCol(bcol); // 假设 var bcol = new BCol("reseller", BOOLEAN, true);
```

ChartMeasureCol

创建图表所用的度量字段。

col: 是需要做统计函数的度量字段 ;

col2: 做统计时需要的第二字段，对于部分函数需要，例如计算数据的权重之和函数 SumWT(col['sales'], col['profit']);

total: 统计函数

axisOpt: 该度量数据段对应的轴属性 // new AxisOpt();

举例：

```
var b = ChartMeasureCol(SumWT,bcol,bcol1);

// var bcol = new BCol("sales", DOUBLE, false);

// var bcol1=new BCol("profit",DOUBLE,false);
```

NamedGroupCol

创建图表所用的分组字段。

setGroups: 设置分组的名称 ;

getGroups: 获取分组的名称 ;

name: 获取分组字段的名称 ;

view: 获取分组字段的别名 ;

dtype: 获取字段的类型 ;

dim: 获取字段是度量还是维度 ;

举例 :

```
var ngcol = new NamedGroupCol("PROFIT", DOUBLE, false);
```

SingleMarkCol

创建非雷达图的标记字段。标记类型包括柱状图、线图、点图等。

col: 设置轴的字段 ;

type: 设置轴的标记类型 ;

axisOpt: 该度量数据段对应的轴属性 ;

举例 :

```
var a = new SingleMarkCol(mcol, BAR);

// var bcol1 = new BCol("customer_id", INTEGER, false);

//var mcol = new ChartMeasureCol(SUM,bcol1, null);
```

RadarMarkCol

创建雷达字段。

cols: 设置图表度量字段 ;

axisOpt: 设置轴的选项 ;

举例 :

```
var a = new RadarMarkCol(mcols);

// var bcol1 = new BCol("profit", INTEGER, false); var mcol1 = new ChartMeasureCol(SUM,
bcol1,null);

//var bcol2 = new BCol("sales", INTEGER, false); var mcol2 = new ChartMeasureCol(SUM,
bcol2,null,);

//var bcol3 = new BCol("cost", INTEGER,false); var mcol3 = new ChartMeasureCol(SUM,
bcol3,null);

//var mcols = [mcol1,mcol2, mcol3];
```

Locator

定位区域，不同组件会有不同的 type 和 level。

type：是指区域的类型，包括 CELL, ROW, COL, OBJECT, TITLE, SUB_TITLE, CHART_XTITLE, CHART_YTITLE, AXIS, AXIS_LINE, AXIS_LABEL, RADAR_AXIS, MARK, MARK_TEXT, LEGENDS, LEGEND, LEGEND_BAND, LEGEND_TITLE, LEGEND_ITEM, PLOT, GRID_LINE, LABEL_FORM,

SHAPE_FORM, TARGET_LINE, TREND_LINE ；

在本产品中数据类型也可以用常量来替换，对应关系如下表。

区域类型	相应的整数	说明
CELL	1	单元格
ROW	2	行
COL	3	列
OBJECT	4	对象
TITLE	5	标题
SUB_TITLE	6	次标题
CHART_XTITLE	10	图表的 X 轴标题
CHART_YTITLE	11	图表的 Y 轴标题
AXIS	12	轴的整体 (包括轴的刻度、标签、轴线等)
AXIS_LINE	13	轴线
AXIS_LABEL	14	轴的标签
RADAR_AXIS	15	雷达图的轴
MARK	16	标记
MARK_TEXT	17	标记值
LEGEND	18	图例
LEGEND_BAND	19	连续的图例区域
LEGEND_TITLE	20	图例的标题
LEGEND_ITEM	21	图例的内容
LEGENDS	22	全部图例
PLOT	23	图表区域
GRID_LINE	24	网格线
UNSELECTED_CELL	30	未选择的区域

level：表示级别，包括 HEADER DETAIL FOOTER COLOR_LEGEND SIZE_LEGEND TPATTERN_LEGEND SPATTERN_LEGEND ；

paths：是一个字符串数组，用来表示路径。

举例：

```
var b = new Locator (k); // 假设 k = new g5.Locator();

var c = new Locator (paths, type, level);
```

在本产品中数据类型也可以用常量来替换，对应关系如下表。

区域等级	相应的整数	说明
HEADER	1	表头区域
DETAIL	2	表的主体区域
FOOTER	3	表尾区域
COLOR_LEGEND	4	颜色图例
SIZE_LEGEND	5	大小图例
TPATTERN_LEGEND	6	图案图例
SPATTERN_LEGEND	7	形状图例

表组件的区域

区域类型	表达方式
表的整体区域	var loc=new Locator([],OBJECT,DETAIL);
表头区域	var loc=new Locator([],TITLE,DETAIL);
表的数据区域 (非汇总表)	var loc=new Locator(["Product"],CELL,DETAIL);
表的数据区域 (汇总表 - 分组合计)	var loc=new Locator(["Product","Product_Type"],CELL,DETAIL);// 注意 paths 为当前字段与最内层维度字段
表的分组合计区域 (汇总表 - 分组合计)	var loc=new Locator(["Product_Type","Product_Type"],CELL,DETAIL);
表的合计区域 (汇总表 - 合计)	var loc=new Locator(["ROOT","Sum_Sales"],CELL,FOOTER);

交叉表组件的区域

区域类型	表达方式
交叉表的整体区域	var loc=new Locator([],OBJECT,DETAIL);
交叉表表头区域	var loc=new Locator([],TITLE,DETAIL);
行表头与列表头的交叉区域	var loc=new Locator(["Product","Market"],CELL,HEADER);
行表头区域	var loc=new Locator(["Market","Market"],CELL,HEADER);
列表头区域	var loc=new Locator(["Product","Product","Sum_Sales"],CELL,HEADER);

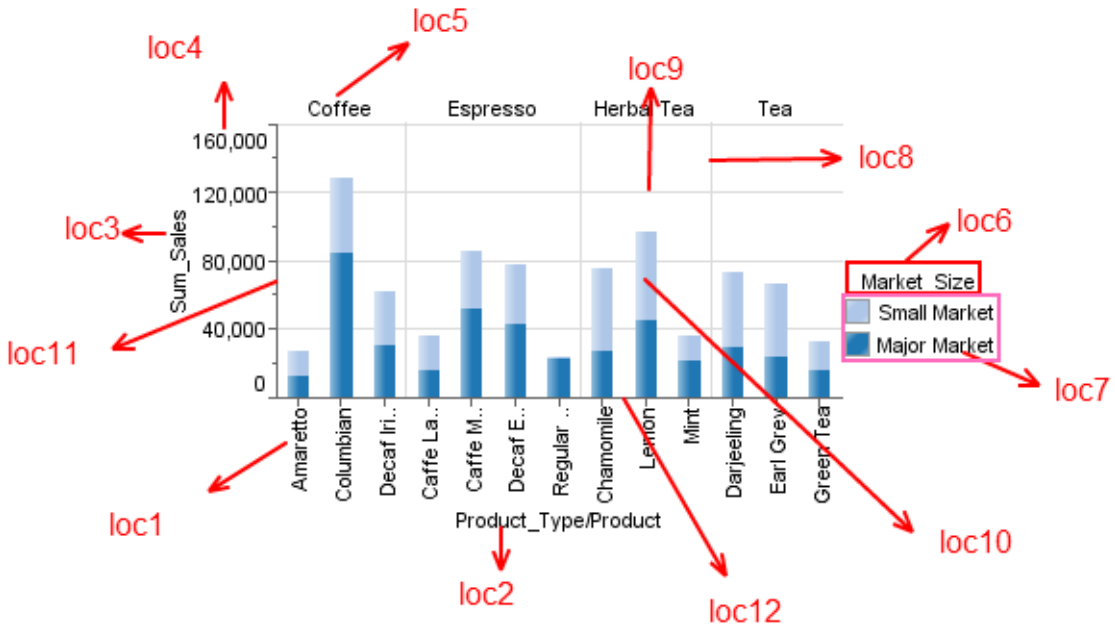
数据区域	var loc=new Locator(["Market","Product","Sum_Sales"],CELL,DETAIL);
指标名称的表头	var loc=new Locator(["Product","Sum_Sales"],CELL,HEADER);
指标名称与行（/列）表头的交叉区域	var loc=new Locator(["Market","Sum_Sales"],CELL,HEADER);

自由式表格组件的区域

区域类型	表达方式
自由式表的整体区域	var loc=new Locator([],OBJECT,DETAIL);
自由式表的表头区域	var loc=new Locator([],TITLE,DETAIL);
自由式表的数据区每一行	var loc=new Locator(["r-i"],ROW,DETAIL);// 第一行 i 取 0，第二行为 1，依次类推
自由式表的数据区每一列	var loc=new Locator(["c-i"],COL,DETAIL);// 第一列 i 取 0，第二列为 1，依次类推
自由式表的数据区域	var loc=new Locator(["i,j"],CELL,DETAIL);// 第一行第一列 i,j 都为 0，第一行第二列 i 为 0，j 为 1，依次类推

自由式表的行表头每一行	var loc=new Locator(["r-i"],ROW,HEADER);// 第一行 i 取 0 ，第二行为 1 ，依次类推
自由式表的列表头每一列	var loc=new Locator(["c-i"],COL,HEADER);//第一行i取0 ，第二行为 1 ，依次类推
行表头与列表头的交叉区域	var loc=new Locator(["0,0"],CELL,HEADER);

图表组件的区域



区域类型	表达方式
X 轴标签区域	var loc1=new Locator(["Product"],AXIS_LABEL,DETAIL);
X 轴的标题区域	var loc2=new Locator([],XTITLE,DETAIL);
Y 轴的标题区域	var loc3=new Locator([],YTITLE,DETAIL);

Y 轴标签区域	var loc4=new Locator(["Sum_Sales"],AXIS_LABEL,DETAIL);
X 轴的外层标签区域	var loc5=new Locator(["Product_Type"],AXIS_LABEL,DETAIL);
图例的标题区域	var loc6=new Locator(["Market_Size"],LEGEND_TITLE,COLOR_LEGEND);//level 分为 COLOR,TPATTERN/SPATTERN,SIZE
图例的数据区域	var loc7=new Locator(["Market_Size"],LEGEND_BAND,COLOR_LEGEND);//level 分为 COLOR_LEGEND,TPATTERN_LEGEND/ SPATTERN_LEGEND,SIZE_LEGEND
X 轴对应的网格线	var loc8=new Locator(["Product_Type"],GRID_LINE,DETAIL);
Y 轴对应的网格线	var loc9=new Locator(["Sum_Sales"],GRID_LINE,DETAIL);
图表标记区域	var loc10=new Locator(["Sum_Sales"],MARK,DETAIL);
Y 轴的轴线区域	var loc11=new Locator(["Sum_Sales"],AXIS_LINE,DETAIL);
X 轴的轴线区域	var loc12=new Locator(["Product_Type"],AXIS_LINE,DETAIL);
图表标记值区域	var loc13 = new Locator(["Sum_Sales"],MARK_TEXT,DETAIL);

文本 / 仪表 / 文本参数 / 图片 / 范围过滤 / 提交的区域

区域类型	表达方式
文本的区域	var loc=new Locator([],OBJECT,DETAIL);

列表过滤 / 树过滤 / 下拉参数 / 列表参数的区域

区域类型	表达方式
整体区域	var loc1=new Locator([],OBJECT,DETAIL);
表头区域	var loc=new Locator([],TITLE,DETAIL);
数据区域	var loc=new Locator([],CELL,DETAIL);

日期组件的区域

区域类型	表达方式
表头区域	var loc=new Locator([],TITLE,DETAIL);
数据区域	var loc=new Locator([],CELL,DETAIL);

选项卡组件的区域

区域类型	表达方式
整体区域	var loc1=new Locator([],OBJECT,DETAIL);
被选中的区域	var loc=new Locator([],CELL,DETAIL);
未被选中的区域	var loc=new Locator([],UNSELECTED_CELL,DETAIL);

Color

颜色有五种构造法。

- var a = new Color();// 无参数时取默认颜色，默认为黑色。
- var b = new Color(java.awt.Color.RED); //java.awt.Color 对象作为参数对象
- var c = new Color(11184810); // 颜色为 0xAAAAAA, 参数是一个十进制整型数据
- var d = new Color(22, 123, 222); // 创建具有指定红色、绿色和蓝色值的不透明的 sRGB 颜色，这些值都在 (0 - 255) 的范围内。
- var e = new Color(22, 123, 222, 10); // 创建具有指定红色、绿色、蓝色和 透明值的 sRGB 颜色，这些值都在 (0 - 255) 的范围内。

举例：

```
var loc=new Locator(["product"],CELL,DETAIL);
var color=new Color(java.awt.Color.RED);
Table1.setBackground(loc,color);
```

DataGrid

表示一个数据表，展现数据时所需要的数。

dim(int c) 检测当前字段是否是维度字段；

size(int c) 获取特定列的大小，如果是 -1，则获取所有列的大小；

csize() 获取列数；

hsize() 获取数据区域的表头；

header(int c) 获取指定列的标题；

ctype(int c) 获取指定列的数据类型；

getInt(int r, int c) 获取指定位置的整型值；

getDouble(int r, int c) 获取指定位置的双精度浮点类型值；

get(int r, int c) 获取指定位置的数据；

set(int r,int c,Object data) 在指定位置设置数据；

exists(int r,int c,boolean wait) 检测指定位置的数据，当 c 的值为 -1 时，检测所有数据，当需要检测的数据不存在是，是否继续等待；

cancel() 取消这个数据；

dispose() 布置数据；

removeLink(int r,int c) 移除指定位置的超链接；

indexOf(String Col) 获取指定列的索引 ;

print(int rows) 打印数据 ;

rows(Object state) 获取指定的行 ;

updateFlag(Object state) 提交指定的行 ;

GFont

字体，此字体对象能支持 UNDERLINE（下划线），STRIKETHROUGH（删除线），对应 g5.meta.GFont 的类。

name ：字体的类型，例如 Dialog、 Arial 等。

style ：样式，例如 PLAIN， BOLD 和 ITALIC

size ：字体大小。

strikeline：添加删除线，相应的常量值为 2。

underline：添加下划线，相应的常量值为 1。

有两种使用方法，一种是调用默认的字体 g5.meta.GFont(); 默认的字体大小为 11 磅，字体类型为 Arial，在图表组件中默认的字体类型为 Dialog。

- 通过 g5.meta.GFont(); 来调用默认字体。
- GFont(name,style,size); 来设定字体。

举例：

```
var k =GFont.getDefault();

var c = new GFont("Dialog",BOLD,12);
```

Insets

边界区域，有三种构造方法。

- var a = new Insets();// 调用默认的边界区域值。
- var c = new Insets(1,1,1,1);// 参数是四个整型值，指定顶部、左边、底部、右边区域对象。
- 通过 left， right， top， bottom 来对对象进行引用， a.left=1;a.right=1;a.top=1;a.bottom=1;

axisOpt

设定轴的各个属性。

属性类别	返回值类型	说明
labelVisible	boolean	轴的标签是否可见
lineVisible	boolean	轴线是否可见

tickVisible	boolean	轴的刻度是否可见
labelPosition	number	标签的位置
axisPosition	number	轴的位置
everyN	number	每隔几个刻度显示标签
tickPosition	number	刻度的显示位置设定，是在轴的内部、外部或横跨
gridLineVisible	boolean	网格线是否可见
timescale	boolean	是否是时间轴
discrete	boolean	是否是离散轴
ignoreNull	boolean	是否忽略空值
max	number	设定轴的最大值
min	number	设定轴的最小值
minorIncrement	number	次要刻度
majorIncrement	number	主要刻度
base	number	基值

举例：

var a = new AxisOpt();// 空构造，继承父亲属性。

Rectangle

指定坐标空间中的一个区域，有三种构造

- var a = new Rectangle ();
- var b = new Rectangle (k); // 假设 k = new java.awt.Rectangle ();
- var c = new Rectangle (1,1,1,1);// 左上角作为原点，指定位置为 (x,y)，区域的宽度和高度由 width，height 确定；

SimpleCol

简单动态计算列，它需要的参数有三种，第一个是 MeasureCol, 第二个参数 Option(对应的计算)，第三个是 Addressing(对应的计算依据)。

sort: 排序类型 ;

view: 获取字段别名 ;

Option(对应的计算)

简单的动态计算对应的计算包括七种格式：分别为 DiffOpt(差值)、 PercentFromOpt(百分比)、DiffPercentOpt (差值百分比)、 PercentTotalOpt(总数百分比)、 RunningTotalOpt(累计总计)、MoveOpt(移动计算)、 RankOpt(排名计算)。

DiffOpt：

差值，可以有 0 个或者 1 个参数。当为 0 个参数时，默认的百分比方式为向前。当为 1 个参数时，选择百分比的方式，有四个选项：FIRST、 PERVIOUS、 NEXT、 LAST, 也可以通过数字的方式，对应的分别是 1、 2、 3、 4。

举例：

```
var option = new DiffOpt();
var address = new PreDefinedAddressing(TABLE_DOWN);
var bcol = new BCol("SALES",  INTEGER , false);
var a = new MeasureCol(SUM,bcol,null);
var scol = new SimpleCol(a, option, address);
var b = new MeasureCol(NONE, scol, null);
表 1.binding.addCol(0, b);
```

PercentFromOpt

百分比，可以有 0 个或者 1 个参数，与 DiffOpt 类似。

举例：

```
var option = new PercentFromOpt (LAST);
var address = new PreDefinedAddressing(TABLE_DOWN_ACROSS);
var bcol = new BCol("SALES",  INTEGER , false);
var a = new MeasureCol(SUM, bcol, null);
var scol = new SimpleCol(a, option, address);
var b = new MeasureCol(NONE, scol, null);
交叉表 1.binding.addMeasure(0, b);
```

DiffPercentOpt

差值百分比，可以有 0 个或者 1 个参数，与 DiffOpt 类似。

举例：

```
var option = new DiffPercentOpt();
var address = new PreDefinedAddressing(PANE_DOWN);
var bcol = new BCol("SALES",  INTEGER , false);
var a = new MeasureCol(SUM,bcol,null);
```

```
var scol = new SimpleCol(a, option, address);

var b = new MeasureCol(NONE, scol, null);

表 1.binding.addCol(0, b);
```

PercentTotalOpt

汇总百分比，可以有 0 个或者 1 个参数，当为 0 个参数时，默认的汇总方式是总和 (SUM)，当 1 个参数时，需要传入一个字符串，当前支持四种汇总方式分别为：最大值、最小值、平均和总和，其对应的字符串分别为：MAX、 MIN、 AVG、 SUM(不区分大小写)。

举例：

```
var option = new PercentTotalOpt();

var address = new PreDefinedAddressing(ADRESS_CELL);

var bcol = new BCol("SALES",  INTEGER , false);

var a = new MeasureCol(SUM,bcol,null);

var scol = new SimpleCol(a, option, address);

var b = new MeasureCol(NONE, scol, null);

表 1.binding.addCol(1, b);
```

RunningTotalOpt

累计总计，可以有 0 个或者 1 个或者 2 个参数，当为 0 个或者 1 个参数时，和 PercentTotalOpt 类似；当为 2 个参数时，第一个参数对应汇总的方式，第二个参数是一个 QCol 对象。

举例：

```
var bcol1 = new BCol("MARKET"," 市场 ",STRING, true);

var dimCol1 = new DimCol(bcol1);

var address = [dimCol1];

var option = new RunningTotalOpt();

var address1 = new AdvancedAddressing(address);// 类似在高级中设置计算依据为市场列

var bcol = new BCol("SALES",  INTEGER , false);

var a = new MeasureCol(SUM,bcol,null);

var scol = new SimpleCol(a, option, address1);

var b = new MeasureCol(NONE, scol, null);

表 1.binding.addCol(0, b);
```

MoveOpt

移动计算 , 可以 0 个， 1 个， 2 个， 3 个， 4 个， 5 个参数，这 5 个参数分别对应的是汇总方式，前一个，后一个，是否包含当前值，没有足够值则取空。对应的参数类型分别为：字符串， short 类型， short 类型， boolean 类型， boolean 类型。其默认值分别为：”SUM”,2,2,true,false。

举例 :

```
var option = new MoveOpt(AVG);

var address = new PreDefinedAddressing();// 默认表格横向

var bcol = new BCol("SALES",  INTEGER , false);

var a = new MeasureCol(SUM,bcol,null);

var scol = new SimpleCol(a, option, address);

var b = new MeasureCol(NONE, scol, null);

交叉表 1.binding.addMeasure(0, b);
```

RankOpt

排名计算，无参数。

举例 :

```
var option = new RankOpt();

var address = new PreDefinedAddressing(ADRESS_CELL);

var bcol = new BCol("SALES",  INTEGER , false);

var a = new ChartMeasureCol(SUM,bcol,null);

var scol = new SimpleCol(a, option, address);

var b = new ChartMeasureCol(NONE, scol, null);

图表 1.binding.addYCol(0, b);// 给图表的 Y 轴添加动态计算列
```

Addressing(对应的计算依据)

在 Addressing 中，有两种 Addressing ，分别是 PreDefinedAddressing （预定义计算依据）、AdvancedAddressing（高级的计算依据）。

PreDefinedAddressing

创建一个动态计算器字段，预定义计算依据，可以有 0 个或 1 个参数。当 0 个参数，默认的计算依据为 TABLE_ACROSS, 当为 1 个参数，有 9 种选项，分别为 :TABLE_ACROSS （表格横向）、TABLE_DOWN （表格纵向）、 TABLE_ACROSS_DOWN （表格横向后纵向）、TABLE_DOWN_ACROSS （表格纵向后横向）、 PANE_ACROSS （平面横向）、 PANE_DOWN （平

面纵向）、 PANE_ACROSS_DOWN （平面横向后纵向）、 PANE_DOWN_ACROSS （平面纵向后横向）、 ADRESS_CELL （格子）

计算依据可以用常量替换，对应关系如下表：

计算依据	相应的整数
TABLE_ACROSS	1
TABLE_DOWN	2
TABLE_ACROSS_DOWN	3
TABLE_DOWN_ACROSS	4
PANE_ACROSS	5
PANE_DOWN	6
PANE_ACROSS_DOWN	7
PANE_DOWN_ACROSS	8
ADRESS_CELL	9

AdvancedAddressing

创建一个动态计算器字段，高级的计算依据，其中可以有 1 个参数或者 2 个参数，当一个参数的时候，是 QCol 对象的一个数组，表示计算依据的维度列。当有 2 个参数时，第一个参数是计算依据的维度列，第二个参数是 QCol 对象的一个数组，表示的是排序，即排序的度量列。

举例：

```
var bcol1 = new BCol("MARKET",STRING, true);// 定义列

var dimCol1 = new DimCol(bcol1);// 定义维度列，已经绑定的

var bcol2 = new BCol("SALES",INTEGER,false);// 定义列

var measure = new MeasureCol("AVG",bcol2,null);// 定义高级中的度量列，聚合方式为平均

measure.sortType=SORT_DESC;     // 排序类型设置为降序，默认为升序

var address1 = [dimCol1]; // 定义高级中的计算依据为已存在的维度列

var sort = [measure];// 高级中的排序度量列

var option = new DiffOpt(FIRST);

var address = new AdvancedAddressing(address1, sort);// 计算依据为高级

var bcol = new BCol("ID",  INTEGER , false);

var a = new MeasureCol(SUM,bcol,null);// 定义度量列

var scol = new SimpleCol(a, option, address);// 定义简单动态计算列
```

var b = new MeasureCol(NONE, scol, null);// 再将动态计算列定义为度量列

表 1.binding.addCol(4, b);// 给表 1 添加列

说明：当计算依据不为空，使用一个有别名的字段时，名称和别名都需取到。

例如：定义有别名的列 var bcol1 = new BCol("MARKET","MARKET 的别名 ",STRING, true);

AdvancedCol

定义定制的动态计算，在定制的动态计算列中，可以有 3 个或者 4 个参数。当有 3 个参数时，需要的是 name,script 和 addressing。其中 name 和 script 为字符串， addressing 为一个 addressing 对象。当有 4 个参数时，多了一个 secondary(二次计算参数)，是一个 boolean 类型，默认值为 false。

举例：

var script="diff(Sum(col['SALES']),PREVIOUS)" // 定义 script 参数值，对总和 _SALES 列求差值方向向前（FIRST（第一个）、 PREVIOUS（向前）、 NEXT（向后）、 LAST（最后一个））

var address = new PreDefinedAddressing(TABLE_DOWN);

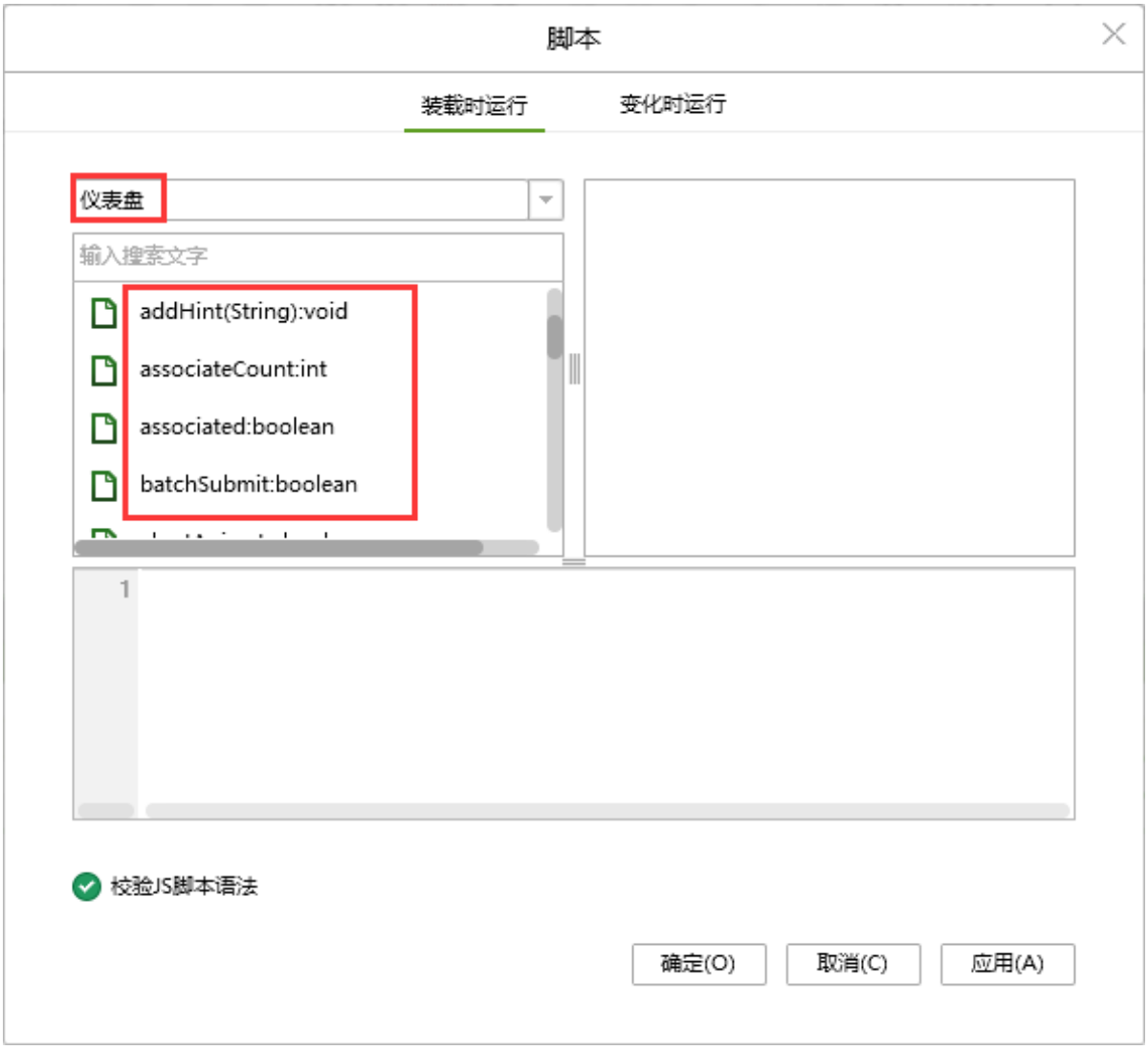
var scol = new AdvancedCol("diff", script, address);

var b = new MeasureCol(NONE, scol, null);

表 1.binding.addCol(1, b);

仪表盘级别的函数介绍

Yonghong Z-Suite 提供的仪表盘级别的函数如下图所示。



函数介绍如下表所示：

属性类别	说明	用法举例
addHint	设定仪表盘弹出提示框。 addHint(String);	addHint("Tip Box");
associated	设定过滤器之间是否有联动关系。	associated = false;
batchSubmit	设定批量提交。	batchSubmit = true;

createConnection	创建数据连接。 createConnection(int, String);	createConnection(SQL, "Folder1/ 学生表 ");
getData	获取组件的数据。 getData(String, Object);	getData(“Table1”,DATA);
getImage	创建图表组建。 getImage(Object bc, Object data, int width, int height);	getImage(bc,data,cell.width,cell.height);
getViewData	获取组件数据。 getViewData(String);	getViewData(" 表 1", DATA);
interval	刷新仪表盘的间隔时间。	dashboard.interval=1000;
param	设定参数	param["a"];
setData	给指定的组件设定绑定的数据。 setData(String,Object,Object);	setData("Table1",a,DATA);// var a = execute(this, SQL, "customer");
showLoadingDetail	显示加载的详细信息。加载报表时，是否显示 loading 的图标。	showLoadingDetail=true;
insert	插入数据到数据库。	FormTable1.insert();
update	更新数据到数据库。	FormTable1.update();
chartAnimate	设定图表是否显示动态效果。	chartAnimate = false;
refreshData	提交更新的数据到数据库中。	FormTable1.refreshData();
associateCount	支持允许关联的过滤组件个数。	associateCount = 5;
floatToolbar	设置工具栏显示效果。是否悬浮。	floatToolbar=true;

setDBBackground	设置报表背景色。 setDBBackground(Object);	var color = new Color(java.awt.Color.RED); setDBBackground(color);
getDBBackground	获取报表背景色。 getDBBackground(Object);	debug(getDBBackground());

user	访问仪表盘的用户，可访问属性包括 name(用户名)，group(组)，roles(角色)。	debug("user....."+user.group);
		debug("user....."+user.roles[0]);
		debug("user....."+user.name);

组件级别的函数

每个组件既含有通用的脚本函数，又含有特定的脚本函数，下面针对这两种脚本函数详细介绍。

通用的函数

函数	说明	举例
enabled	设定当前组件是否可用	id.enabled = false;
visible	设定当前组件是否可见	id.visible=false;
interval	设定当前组件的刷新时间间隔，单位是秒	id.interval=100000;
x,y,width,height	x,y 来确定起始位置，width 和 height 确定区域的宽度和高度。	id.x=100;
Format	设置文本的格式。详细介绍见下表。	id.setFormat(loc,DATE_FORMAT);//var loc=new Locator(["sell_date"],CELL,DETAIL);
FormatPattern	设置文本的格式类型，如设定日期的显示格式 MM/dd/yyyy	id.setFormatPattern(loc, "#,##0.##");// var loc = new Locator(["price"], CELL, DETAIL);

Font	设置字体的格式	var loc=new Locator("price",CELL,HEADER);
		var font=new GFont.getDefault();
		id.setFont (loc,font);

Foreground	设置前景色	id.setForeground (loc,color);

Background	设置背景色	var loc=new Locator(["product"],CELL,DETAIL);
		var color=new Color(0xAAAAAA);
		id.setBackground (loc,color);

Borders	设置边框，边框的类型见边框类型表	var loc=new Locator(["price"],CELL,DETAIL); Table1.setBorder(loc,THICK_LINE);

BorderColor	设置边框颜色	id.setBorderColor(loc, color);
		//var loc = new Locator("product"],CELL,DETAIL);
		//var color=newColor(java.awt.Color.red);

BorderColors	设置各个边框的颜色	id.setBorderColors(loc, color);
		//var loc = new Locator("product"],CELL,DETAIL);
		var color2 =[color1,color2,color3,color4];

HAlignment	设置水平对齐方式，见下表	Table1.setHAlignment(loc,2);
Rotation	设置旋转角度	id.setRotation(loc,ROTATION_45);
VAlignment	设置垂直对齐方式	Table1.setVAlignment(loc,8);
Alpha	设置透明度	id.setAlpha(loc,20);
Wrap	设定是否自动换行	Table1.setWrap(loc,true);
ColWidth	设置单元格的宽度	Table1.setColWidth(loc,100);
RowHeight	设置单元格的行高	Table1.setRowHeight(loc,100);
setLinkEnable	设置超链接是否可见	var loc = new Locator(["STATE"], CELL, DETAIL); Table1.setLinkEnable(loc,false);
isLinkEnbale	返回超链接是否可见	var a=Table1.isLinkEnbale(loc);
title	设置标题	id.title = "Yonghong";
showTitle	是否显示标题	id.showTitle=true;
active	组件有效	id.active = true;
detailMaxRows	显示具体数据时的最大行数	id.detailMaxRows = 1000;
detailVisibleCols	显示具体数据时的显示字段	id.detailVisibleCols = ["marketmysql", "sale_type"];
excelSheetName	组件导出到 excel 中 sheet 的名字	Table1.excelSheetName = "Yonghong";
exclude	排斥组件	id.exclude = true;
removeLinkParam	移除超链接参数	var loc = new Locator(["product"], CELL, DETAIL); Table1.removeLinkParam(loc, "market");

setLinkParam	设置超链接参数	var loc = new Locator(["product"], CELL, DETAIL); Table1.setLinkParam(loc, "marketmysql", "aa");
source	绑定的数据源	id.binding.source = "test/orders.sqry";
setFitContent	设置单元格自适应大小	var loc = new Locator(["product"], CELL, DETAIL); 表 1.setFitContent(loc, true);
isFitContent	返回单元格是否自适应大小	var loc = new Locator(["product"], CELL, DETAIL); 表 1.isFitContent(loc);
innerParam	用来给相同的参数名传递不同的参数值	id.innerParam[“market”]=”West”; id.innerParam[“market”]=”South”; //market 为参数 ,South,West 为值

setLink	给文本、仪表、表格、交叉表和图表组件设置超链接，链接包含四种：操作，仪表盘，数据列和 URL。操作包含六种：Excel，PDF，Word, PNG, CSV, Refresh。	链接到 Excel：
		var loc=new Locator([],OBJECT,DETAIL);
		文本 1.setLink(loc,"action://Excel")
		链接到 PDF：
		var loc = new Locator(["STATE","STATE"], CELL,HEADER);
		交叉表 1.setLink(loc, "action://PDF ");
		链接到仪表盘：
		var loc=new Locator([],OBJECT,DETAIL);
		仪表 1.setLink(loc,"db://folder1/aaa")
		链接到数据列：
		var loc = new Locator(["PRODUCT"], CELL, DETAIL);
		表 1.setLink(loc, "datalink://PRODUCT");
		链接到 URL：
		var loc = new Locator(["Sum_SALES"],MARK,DETAIL);
		图表 1.setLink(loc,"http://www.yonghongtech.com")

Link	给图片组件设置超链接，链接包含三种：操作，仪表盘，URL。操作包含六种：Excel， PDF， Word, PNG, CSV, Refresh。	链接到 Word： 图片 1.link = "action://Word"; 链接到仪表盘： 图片 1.link = "db://folder1/aaa"; 链接到 URL： 图片 1.link = "http://www.yonghongtech.com";
Span	设置合并单元格	var loc = new Locator(["product"], CELL, DETAIL); id.setSpan(loc, new Dimension(2, 1));

格式表

格式类型	相应的常量	说明
NONE_FORMAT	0	空
DATE_FORMAT	1	日期
DECIMAL_FORMAT	2	数字

格式表

CURRENCY_FORMAT	3	货币
PERCENT_FORMAT	4	百分比
MESSAGE_FORMAT	5	文本

边框类型表

边框类型	相应的常量	说明
NO_BORDER	Constants.NO_BORDER	无边框
VERY_THIN_LINE	Constants.VERY_THIN_LINE	较细
THIN_LINE	Constants.THIN_LINE	细线
MEDIUM_LINE	Constants.MEDIUM_LINE	中等
THICK_LINE	Constants.THICK_LINE	粗线
DOT_LINE	Constants.DOT_LINE	点线
DASH_LINE	Constants.DASH_LINE	虚线
MEDIUM_DASH	Constants.MEDIUM_DASH	短虚线
LARGE_DASH	Constants.LARGE_DASH	长虚线
DOUBLE_LINE	Constants.DOUBLE_LINE	双线

对齐方式

对齐方式	**相应常量**	**说明**
LEFT_ALIGN	0	左对齐
CENTER_ALIGN	1	水平居中
RIGHT_ALIGN	2	右对齐
TOP_ALIGN	0	顶部对齐
MIDDLE_ALIGN	4	垂直居中
BOTTOM_ALIGN	8	底部对齐

文本旋转角度的设定

旋转度数	相应的常量	说明
ROTATION_0	0	逆时针旋转 0 度
ROTATION_45	1	逆时针旋转 45 度
ROTATION_90	2	逆时针旋转 90 度
ROTATION_135	3	逆时针旋转 135 度
ROTATION_180	4	逆时针旋转 180 度
ROTATION_225	5	逆时针旋转 225 度

文本旋转角度的设定

ROTATION_270	6	逆时针旋转 270 度
ROTATION_315	7	逆时针旋转 315 度
ROTATION_360	8	逆时针旋转 360 度

表组件的脚本函数

函数	说明	举例

addCol	增加一列	Table1.binding.addCol(dimCol);
		//var bcol = new BCol("reseller", BOOLEAN, true);
		//var dimCol = new DimCol(bcol);

aggregate	是否是聚合的表	Table1.binding.aggregate=true;
clearCols	清空表	Table1.binding.clearCols();
colCount	返回表的列数	var a=Table1.binding.colCount;
getCol	获取表中的某一列	var ss = Table1.binding.getCol(0);
getCols	获取表中的若干列	var cc=Table1.binding.getCols(arr);//var arr = [dimCol, mcol];
removeCol	移除列	Table1.binding.removeCol(0);
setCol	给某一列赋值	Table1.binding.setCol(0,dimCol);
setCols	给表的若干列赋值	Table1.binding.setCols(arr);//var arr = [dimCol, mcol];
showTotal	是否显示合计	Table1.binding.showTotal=true;
totalTop	合计前置	Table1.binding.totalTop=true;
sortExclude	设定表的排序类型，详细介绍见下表（表排序）	Table1.sortExclude=0;//0,1,2
maxRows	设定表组件的最大行数	Table1.maxRows=100;// 默认是 0，自动设定表的行数。
headers	表头行数	Table1.headers = 2;
sortStrategy	设置排序策略	Table1.sortStrategy = 2;

表排序

排序类型	相应的常量	说明
EXCLUDE_NULL	0	字段间排序无影响
EXCLUDE_MEASURE	2	度量字段间排序互斥，与维度字段的排序没有关系
EXCLUDE_ALL	1	所有字段间排序进行互斥

交叉表组件的脚本函数

函数	说明	举例
addColHeader	给交叉表的列表头增加字段	Pivot1.binding.addColHeader(dimCol);
addMeasure	给交叉表增加汇总字段	Pivot1.binding.addMeasure(meaCol);
addRowHeader	给交叉表增加行表头字段	Pivot1.binding.addRowHeader(dimCol);
clearColHeaders	清空交叉表的列表头字段	Pivot1.binding.clearColHeaders();
clearMeasures	清空交叉表的汇总字段	Pivot1.binding.clearMeasures();
clearRowHeaders	清空交叉表的行表头字段	Piovt1.binding.clearRowHeaders();
colGrps	给交叉表的列表头绑定一个数组	Pivot1.binding.colGrps=arr;//var arr=[dimCol1,dimCol2];
colHeaderCount	返回交叉表的列表头数量	var c= pivot1.banding.colHeaderCount;
getColHeader	获取交叉表的列表头字段信息	var a =Pivot1.binding.getColHeader(0);
getMeasure	获取交叉表的汇总表头字段信息	var a =Pivot1.binding.getMeasure(0);
getRowHeader	获取交叉表的行表头字段信息	var a =Pivot1.binding.getRowHeader(0);
measureCount	获取交叉表的汇总字段数量	var a=Pivot1.binding.measureCount;
measures	给交叉表的汇总行绑定数据	Pivot1.binding.setMeasures=arr;//var arr=[mCol1,mCol2];
removeColHeader	移除交叉表的列字段	Pivot1.binding.removeColHeader(0);
removeMeasure	移除汇总数据段	Pivot1.binding.removeMeasure(0);

removeRowHeader	移除交叉表的行字段	Pivot1.binding.removeRowHeader(0);
rowGrps	给交叉表的行表头绑定一个数组	Pivot1.binding.rowGrps=arr;//var arr=[dimCol1,dimCol2];
rowHeaderCount	返回行表头的字段数量	var c =Pivot1.binding.rowHeaderCount;
setColHeader	给交叉表的列表头绑定字段	Pivot1.binding.setColHeader(0,dimcol);
setMeasure	给交叉表的汇总行绑定数据	Pivot1.binding.setMeasure(0，mcol);
setRowHeader	给交叉表的行表头绑定字段	Pivot1.binding.setRowHeader(0,dimcol);
showCTotal	是否显示交叉表的列数的分组合计	Pivot1.binding.showCTotal = true;
showMeasureHeader	在交叉表中显示汇总字段	Pivot1.binding.showMeasureHeader=true;
showRTotal	是否显示交叉表的行数的分组合计	Pivot1.binding.showRTotal = true;
vertical	交叉表的指标名称横向排列还是纵向排列显示	Pivot1.binding.vertical=true;
spanTopLeft	交叉表的行表头和列表头交叉是否进行合并	Pivot1.binding.spanTopLeft = true;
nullAsCharacter	交叉数据区域空数据是否显示为指定字符，例如空数据使用字母 'a' 表示	Pivot1.nullAsCharacter =’a’；
maxRows	设定交叉表的最大显示行数	Pivot1.maxRows=100;

图表组件的脚本函数

图表组建的脚本函数包含与绑定、图例、 X 轴的标题、 Y 轴的标题相

关的脚本函数，下面分别介绍。

图表组件的脚本函数

函数	说明	举例
dodgeMark	当有多个标记时，设定多个标记是否重叠显示	Chart1.dodgeMark=false;// 重叠显示
hiddenTip	当鼠标悬停时是否显示标记对应的值	Chart2.hiddenTip=true;// 隐藏显示值
overlapType	是否重叠图表标签，详细介绍见下表（图表标签）	Chart1.overlapType=OVERLAP_N;
LineColor	网格线条的颜色	Chart1.setLineColor(loc, 0xff2219);
LineStyle	网格线条的样式	Chart1.setLineStyle(loc, THICK_LINE);
Rotation	旋转角度	Chart1.setRotation(loc, ROTATION_90);
getTargetLine	获取图表的目标线	Chart1.getTargetLine(0).name = "targetline1";
getTargetLineSize	获取图表的目标线的数量	debug(Chart1.getTargetLineSize());
Bchart	原始图表	var bc= getBChart();

图表标签

标签重叠类型	相应的常量	说明
OVERLAP_A	0	自动
OVERLAP_N	1	保留重叠标签
OVERLAP_S	2	打散重叠标签
OVERLAP_R	3	删除重叠标签

图表组件关于绑定的脚本函数

函数	说明	举例
axisPosition	轴的显示位置，见下表	Chart1.binding.axis["Sum_price"].axisPosition=-2;
base	基值	Chart1.binding.axis["Sum_price"].base=500;
discrete	是否是离散轴	Chart1.binding.axis["product"].discrete=true;

图表组件关于绑定的脚本函数

everyN	每隔多少个刻度显示标签	Chart1.binding.axis["Sum_price"].everyN=3;
Values	刻度值	var vals = [0, 152579, 814929, 1017986]; Chart1.binding.axis["Sum_sales"].setValues(vals);
gridLineVisible	网格线是否可见	Chart1.binding.axis["product"].gridLineVisible=false;
ignoreNull	是否忽略空值	Chart1.binding.axis["product"].ignoreNull=true;
labelPosition	标签的位置	Chart1.binding.axis["product"].labelPosition=LABEL_INNER;//LABEL_OUTER
labelVisible	标签是否可见	Chart1.binding.axis["product"].labelVisible=false;
lineVisible	轴线是否可见	Chart1.binding.axis["product"].lineVisible=false;
logScala	是否为对数刻度	Chart1.binding.axis["price"].logScala=false;
majorIncrement	主要刻度	Chart1.binding.axis["Sum_price"].majorIncrement=10000;
max	最大值	Chart1.binding.axis["Sum_price"].max=10000;
min	最小值	Chart1.binding.axis["Sum_price"].min=0;
minorIncrement	次要刻度	Chart1.binding.axis["Sum_price"].minorIncrement=500;
shareGroup	共享轴	Chart1.binding.axis["Sum_profit"].shareGroup="shareAxis"; Chart1.binding.axis["Sum_sales"].shareGroup="shareAxis";
reversed	反转	Chart1.binding.axis["product"].reversed=true;
tickPosition	刻度位置	Chart1.binding.axis["product"].tickPosition=TICK_INNER;//TICK_OUTER、TICK_CROSS

图表组件关于绑定的脚本函数

tickVisible	刻度是否可见	Chart1.binding.axis["product"].tickVisible=false;
timeScala	是否是时间轴	Chart1.binding.axis["product"].timeScala=false;
visible	是否可见	Chart1.binding.axis["product"].visible=false;
colNumber	图例的显示列数	Chart1.binding.markCol["Sum_price"].colorLegend.colNumber=2;
title	标题	Chart1.binding.markCol["Sum_price"].colorLegend.title="a";
titleVisible	标题是否可见	Chart1.binding.markCol["Sum_price"].colorLegend.titleVisible=false;
hiddenMeasures	鼠标悬停时增加显示所设置的未绑定度量值	Chart1.binding.hiddenMeasures = [new MeasureCol(SUM, new BCol("ID", INTEGER, false))];// 鼠标悬停时增加显示 Sum_ID 的值
tipFormat	动态的显示 tip 的值	Chart1.binding.markCol["Sum_PROFIT"].tipFormat = "{col[\"PRODUCT\"]}";

colorField	给美化界面上的颜色行绑定字段	var bcol=new BCol("nation",STRING,true);
		var qcol=new ChartDimCol(bcol);
		Chart1.binding.markCol["Sum_price"].colorField=qcol;

colorGuide	设定颜色类型，详细介绍见下表 (美化界面 - 颜色)	var d = new RainbowColorGuide(); Chart1.binding.markCol["Sum_price"].colorGuide=d;
explode	分离饼图	Chart2.binding.markCol["Sum_price"].explode=20;
innerRadius	环状图的内半径	Chart3.binding.markCol["Sum_price"].innerRadius=30;
mType	图表标记类型，见下表 (标记类型)	Chart1.binding.markCol["Sum_price"].mType=BAR;

图表组件关于绑定的脚本函数

resetGroup	当标记类型为堆积时，是否重置	Chart1.binding.markCol["Sum_price"].resetGroup=false;

patternField	给美化界面上的图案 / 形状行绑定字段	var bcol=new BCol("nation",STRING,true);
		var qcol=new ChartDimCol(bcol);
		Chart1.binding.markCol["Sum_price"].patternField=qcol;

patternGuide	设定图案 / 形状类型，详细介绍见下表 (美化界面 - 图案 / 美化界面 - 形状)	var d=new DefSPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;
showPoints	是否显示点	Chart1.binding.markCol["Sum_price"].showPoints=false;
showRline	显示点图的基准线	Chart1.binding.markCol["Sum_price"].showRline=true;
showValues	显示标记值	Chart1.binding.markCol["Sum_price"].showValues=true;

sizeField	给美化界面上的大小行绑定字段	var bcol=new BCol("nation",STRING,true);
		var qcol=new ChartDimCol(bcol);
		Chart1.binding.markCol["Sum_price"].sizeField=qcol;

sizeGuide	设定显示大小类型，详细介绍见下表 (美化界面 - 大小)	var d=new ContinuousSizeGuide(); Chart1.binding.markCol["Sum_price"].sizeGuide=d;
sizeGuide.sizeRatio	设置柱子形状的图表中柱子显示的宽度为当前宽度的倍数，默认 size 为 0, 只支持正整数	Chart1.binding.markCol["Sum_ID"].sizeGuide.sizeRatio =2; (显示宽度为当前宽度的 2 倍)
stackDim	堆积字段	Chart1.binding.markCol["Sum_profit"].stackDim = "market";
stackAsBase	堆积基值	Chart1.binding.markCol["Sum_price"].stackAsBase=true;
stackNeg	堆积负数	Chart1.binding.markCol["Sum_ 利润 "].stackNeg=true;

图表组件关于绑定的脚本函数

stackValues	是否堆积标记值	Chart1.binding.markCol["Sum_price"].stackValues=true;
startAngle	为饼图设定开始角度	Chart1.binding.markCol["Sum_price"].startAngle="45";

textField	给美化界面上的标签行绑定字段	var bcol=new BCol("nation",STRING,true);
		var qcol=new ChartDimCol(bcol);
		Chart1.binding.markCol["Sum_price"].textField=qcol;

textGuide	标签	var d=TextGuide(); Chart1.binding.markCol["Sum_price"].textGuide=d;

addGrpCol	给图表的分组增加字段，用户需要指定增加字段的位置以及字段名	var bcol=new BCol("nation",STRING,true);
		var qcol=new ChartDimCol(bcol);
		Chart1.binding.addGrpCol(1,qcol);

addXCol	给图表的 X 轴增加字段，用户需要指定增加字段的位置以及字段名	var bcol=new BCol("nation",STRING,true);
		var qcol=new ChartDimCol(bcol);
		Chart1.binding.addXCol(1,qcol);

addYCol	给图表的 Y 轴增加字段，用户需要指定增加字段的位置以及字段名	var bcol1=new BCol("price2",DOUBLE,false);
		var qcol1=new ChartMeasureCol(SUM,bcol1,null);
		Chart1.binding.addYCol(1,qcol1);

aestheticSingle	是否是单美化指标	Chart1.binding.aestheticSingle=true;
aggregate	图表是否为聚合的	Chart1.binding.aggregate=true;
blend	混合类型	Chart1.binding.blend=COORD_V;
clearGrpCols	清除美化界面中的分组数据段	Chart1.binding.clearGrpCols();
clearXCols	清除 X 轴绑定的所有字段	Chart1.binding.clearXCols();
clearYCols	清除 Y 轴绑定的所有字段	Chart1.binding.clearYCols();

图表组件关于绑定的脚本函数

getGrpCol	获取组绑定字段的信息	var b =Chart1.binding.getGrpCol(0);
getXCol	获取 X 轴绑定字段的信息	var b=Chart1.binding.getXCol(0);
getYCol	获取 Y 轴绑定字段的信息	var b=Chart1.binding.getYCol(0);
grpColCount	获得组绑定的字段数量	var a =Chart1.binding.GrpColCount;
grpCols	分组的所有字段	var a=[qcol,qcol1]; Chart1.binding.yCols=a;
removeGrpCol	移除组中指定的字段	Chart1.binding.removeGrpCol(0);
removeXCol	移除 X 轴上指定的字段	Chart1.binding.removeXCol(0);
removeYCol	移除 Y 轴上指定的字段	Chart1.binding.removeYCol(0);

setGrpCol	更换组中的特定字段	var col=new BCol("nation",STRING,true);
		var qcol=new ChartDimCol(col);
		Chart1.binding.setGrpCol(0,qcol);

setXCol	更换 X 轴特定字段	var col=new BCol("nation",STRING,true);
		var qcol=new ChartDimCol(col);
		Chart1.binding.setXCol(0,qcol);

setYCol	更换 Y 轴特定字段	var bcol1=new BCol("price2",DOUBLE,false);
		var qcol=new ChartMeasureCol(SUM,bcol1,null);
		Chart1.binding.setYCol(0,qcol);

xColCount	获得 X 轴绑定的字段数量	var a=Chart1.binding.xColCount;
xCols	X 轴上的所有字段	var a=[qcol,qcol1]; Chart1.binding.xCols=a;
yColCount	获得 X 轴绑定的字段数量	var a=Chart1.binding.yColCount;
yCols	Y 轴上的所有字段	var a=[qcol,qcol1]; Chart1.binding.yCols=a;

轴的位置

轴的位置	相应的常量	说明
AXIS_BOTTOM_LEFT	-1	左 / 下
AXIS_UPPER_RIGHT	-2	右 / 上
AXIS_AUTO	-3	自动

标记类型

图表的标记类型	相应的常量	说明
AUTO	0	自动
BAR	1	柱状图
STACK_BAR	513	堆积柱状图
STACK_BAR_NORESET	1537	堆积柱状图 (不重置)
BAR_3D	257	3D 柱状图
STACK_BAR_3D	769	堆积 3D 柱状图
STACK_BAR_3D_NORESET	1793	堆积 3D 柱状图 (不重置)
LINE	2	线图
STACK_LINE	514	堆积线图
POINT	4	点图
STACK_POINT	516	堆积点图
STACK_PIONT_NORESET	1540	堆积点图 (不重置)
AREA	8	面积图
STACK_AREA	520	堆积面积图
STACK_AREA_NORESET	1544	堆积面积图 (不重置)
PIE	528	饼图
PIE_3D	784	3D 饼图
DONUT	544	环状图
RADAR	64	雷达图
FILLED_RADAR	128	填充雷达图
WATERFALL	1539	瀑布图
PARETO	5	帕累托图
MAP	6	地图
Organize	11	组织图

美化界面 - 颜色

颜色	说明	应用举例
DefColorGuide	默认的颜色类型	var d = new DefColorGuide(java.awt.Color.red); Chart1.binding.markCol['Sum_price'].colorGuide= d;// 调用 java 提供的颜色
DiscreteColorGuide	离散的颜色类型	var d=DiscreteColorGuide(); Chart1.binding.markCol["Sum_price"].colorGuide=d;

BrightnessColorGuide	量度，用户可以不设定参数，调用默认的颜色；或者调用 d.setBaseColor(Color color) 来设定基准色	var color=new Color(java.awt.Color.red);
		var d=new BrightnessColorGuide();
		d.setBaseColor(color);
		Chart1.binding.markCol["Sum_price"].colorGuide=d;

美化界面 - 颜色

GradientColorGuide	设定渐变的初始颜色和终止颜色	var color=new Color(java.awt.Color.red);
		var color2=new Color(java.awt.Color.blue);
		var d=new GradientColorGuide();
		d.setStartColor(color);
		d.setEndColor(color2);
		Chart1.binding.markCol["Sum_price"].colorGuide=d;

RainbowColorGuide	彩虹，无参数	var d = new RainbowColorGuide(); Chart1.binding.markCol["Sum_price"].colorGuide=d;

SaturationColorGuide	饱和度，用户可以不设定参数，调用默认的颜色；或者调用d.setBaseColor(Color color)来设定基准色	var color=new Color(java.awt.Color.red);
		var d=new SaturationColorGuide();
		d.setBaseColor(color);
		Chart1.binding.markCol["Sum_price"].colorGuide=d;

美化界面 - 形状

形状	说明	应用举例
DefSPatternGuide	默认的形状类型	var d=new DefSPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;
DBDiscreteSPatternGuide	离散的形状	var d=new DiscreteSPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;

美化界面 - 形状

TrapezoidSPatternGuide	梯形	var d=new TrapezoidSPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;
PolygonSPatternGuide	多边形图	var d=new PolygonSPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;
FillCircleSPatternGuide	填充圆	var d=new FillCircleSPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;

美化界面 - 图案

图案	说明	应用举例
DefTPatternGuide	默认的图案类型	var d=new DefTPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;
DBDiscreteTPatternGuide	离散的图案	var d=new DiscreteTPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;
VerticalTPatternGuide	纵向	var d=new VerticalTPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;
TiltGridTPatternGuide	斜格子	var d=new TiltGridTPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;
RTiltTPatternGuide	右倾	var d=new RTiltTPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;

美化界面 - 图案

LTiltTPatternGuide	左倾	var d=new RTiltTPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;
HorizontalTPatternGuide	水平	var d=new HorizontalTPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;
GridTPatternGuide	格子	var d=new GridTPatternGuide(); Chart1.binding.markCol["Sum_price"].patternGuide=d;

美化界面 - 大小

图案 / 形状	说明	应用举例
DefSizeGuide	产品默认	var d=new DefSizeGuide(); Chart1.binding.markCol["Sum_price"].sizeGuide=d;
DiscreteSizeGuide	离散	var d=new DiscreteSizeGuide(); Chart1.binding.markCol["Sum_price"].sizeGuide=d;
ContinuousSizeGuide	连续	var d=new ContinuousSizeGuide(); Chart1.binding.markCol["Sum_price"].sizeGuide=d;

混合

混合类型	相应的常量	说明
COORD_H	1	水平
COORD_V	2	纵向
COORD_O	4	重叠

图表组件关于图例的脚本函数

函数	说明	举例
bounds	图例的边界	Chart1.legends.bounds=[140,150,5,5]

图表组件关于图例的脚本函数

legendAlign	图例的对齐方式，见下表（图例对齐方式）	Chart1.legends.legendAlign=VERTICAL;
position	图例的位置	Chart1.legends.position=LEFT;// LEFT、RIGHT 、UPPER 、BOTTOM、FLOATABLE
scrollable	是否支持图例内容的滚动	Chart1.legends.scrollable=false;

图例对齐方式

对齐方式	相应的常量	说明
AUTO	0	自动
VERTICAL	1	垂直
HORIZONTAL	2	水平对齐

图表组件关于 X/Y 轴标题的脚本函数

函数	说明	举例
bounds	标题的边界	ytitle.bounds=[0,100,200,300];
position	标题的位置	Chart1.xtitle.position=UPPER;// LEFT、RIGHT、BOTTOM、FLOATABLE
title	标题的名称	ytitle.title="product";
visible	标题是否可见	ytitle.visible=true;

图表组件关于目标线的脚本函数

函数	说明	举例
name	设置目标线的名称	图表 1.getTargetLine(0).name=“目标线 1”；
val	设置目标线的值	图表 1.getTargetLine(0).val=“100000”；
startVal	设置目标线的起始值	图表 1.getTargetLine(0).startVal=“20000”；
measure	设置目标线的度量	图表 1.getTargetLine(0).measure=“Sum_profit”；
hiddenLB	设置标签是否显示	图表 1.getTargetLine(0).hiddenLB=true;

图表组件关于目标线的脚本函数

lineStyle	设置目标线的样式，见下表（边框类型表）	图表 1.getTargetLine(0).lineStyle=“49”；
lineColor	设置目标线颜色	图表 1.getTargetLine(0).lineColor=333;
fillColor	设置起始值到目标值之间的填充颜色	图表 1.getTargetLine(0).fillColor=333;
setFillAlpha(byte fillAlpha)	设置起始值到目标值之间的透明度	图表 1.getTargetLine(0).setFillAlpha(30);
getFillAlpha	获取起始值到目标值之间的透明度	图表 1.getTargetLine(0).getFillAlpha();
labelColor	设置标签颜色	图表 1.getTargetLine(0).labelColor=333;
labelAlign	设置标签对齐方式，见表（对齐方式）	图表 1.getTargetLine(0).labelAlign=LEFT_ALIGN;

labelFont	设置标签字体属性，包括风格、样式、大小、是否有下划线等	var font=new GFont(“宋体”,BOLD,30);
		font.underline=1;
		font.strikeline=2;
		图表 1.getTargetLine(0).font=font;

文本组件的脚本函数

函数	说明	举例
getCol	获得当前文本组件绑定的字段信息	var a=Text1.binding.getCol();// 无参数

setCol	给当前文本组件绑定字段	var col=new BCol("price",DOUBLE,false);
		var col1=new MeasureCol(SUM,col,null);
		Text1.binding.setCol(col1);

commit	展现样式	Text1.commit = true;
commitScript	提交脚本	Text1.commitScript = "test";
data	给文本组件添加文本	Text1.data="hello world!";

仪表组件的脚本函数

函数	说明	举例
getCol	获得当前仪表组件绑定的字段信息	var a=Gauge1.binding.getCol(col1);

setCol	给当前文本组件绑定字段	var col=new BCol("price",DOUBLE,false);
		var col1=new MeasureCol(SUM,col,null);
		Gauge1.binding.setCol(col1);

data	给仪表设定数据。获取仪表的绑定数据值。	Gauge1.data=234;//var a= 仪表1.data;
effect	是否显示渐变效果	Gauge1.effect=false;
effectType	渐变方式	Gauge1.effectType = 5;
gaugeType	仪表的类型，详细介绍见下表（仪表类型）	Gauge1.gaugeType=200;
majorIncrement	主要刻度	Gauge1.majorIncrement="10000";
max	设定仪表的最大值	Gauge1.max="50000";

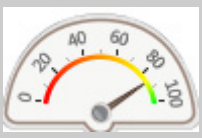
min	设定仪表的最小值	Gauge1.max="0";
minorIncrement	次要刻度	Gauge1.minorIncrement="5000";

rangeColors	设定各个范围的颜色	var color1=new Color(java.awt.Color.red);
		var color2=new Color(java.awt.Color.blue);
		var color3=new Color(java.awt.Color.green);
		var a=[color1,color2,color3];
		Gauge1.rangeColors=a;

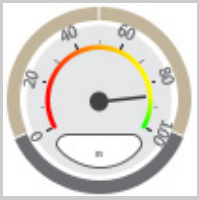
ranges	设定范围	var a=[10000,30000,50000]; Gauge1.ranges=a;
runTimeMajorIncrement	默认的主要刻度	var a=Gauge1.runTimeMajorIncrement;
runTimeMax	默认的最大值	var a=Gauge1.runTimeMax;
runTimeMin	默认的最小值	var a=Gauge1.runTimeMin;
runTimeMinorIncrement	默认的次要刻度线	var a=Gauge1.runTimeMinorIncrement;
themeType	是否为默认主题对应的仪表盘类型	Gauge1.themeType = true;

valueRangeColors	范围值的颜色。接收的值为数组，如果是单色，就传一样的颜色；如果是渐变色，就传不一样的颜色	var arr = [];
		arr[0] = 0xff0000;
		arr[1] = 0x00ff00;
		Gauge1.valueRangeColors[0];

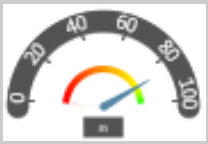
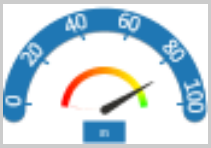
仪表类型

仪表类型	对应的仪表模型
40	
100	
200	
300	
400	
500	

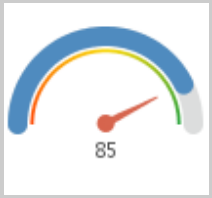
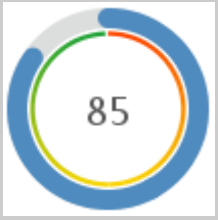
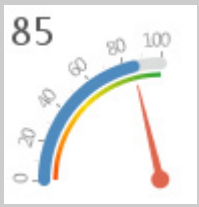
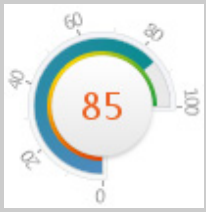
仪表类型

600	
700	
800	
900	
910	
920	

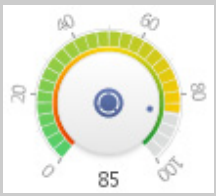
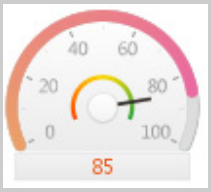
仪表类型

1000	
1010	
2000	
2100	
2200	
2300	
1950	

仪表类型

1960	
1970	
1980	
2410	
2420	
2430	

仪表类型

2440	
2450	
2460	
2470	
2480	

列表过滤的脚本函数

函数	说明	举例
getCol	获得当前列表过滤组件绑定的字段信息	var a=FilterList1.binding.getCol(col1);

setCol	给当前列表过滤组件绑定字段	var col=new BCol("product",2,true);
		var col1=new DimCol(col);
		FilterList1.binding.setCol(col1);

colNumber	设定列表过滤组件的显示列数	FilterList1.colNumber=2;
dropDown	以下拉列表的形式展现	FilterList1.dropDown=true;
dropRow	以下拉列表的形式展现时的行数设置	FilterList1.dropRow=3;
SeletedObjects	设定勾选的数据	FilterList1.setSelectedObjects(["1","5"],INTEGER);
single	转为单过滤器	FilterList1.single=true;
sortType	排序	FilterList1.sortType=SORT_DESC;
submitOnChange	是否改变时提交	FilterList1.submitOnChange=false;
ignoreNull	是否忽略空值	FilterList1.ignoreNull=true;
selectOnTop	选中值置顶	FilterList1.selectOnTop=true;

排序类型

排序类型	相应的常量	说明
SORT_NONE	0	无序
SORT_ASC	1	升序
SORT_DESC	2	降序
SORT_VALUE	4	按值进行排序，此类型适用于复 / 单选框、下拉列表

树状过滤组件的脚本函数

函数	说明	举例
getCols	获得当前树状态过滤组件绑定的字段信息	var a=FilterTree1.binding.getCols();

setCols	给当前树状过滤组件绑定字段	var col=new BCol("product",2,true);
		var col1=new DimCol(col);
		var col2=new BCol("product_type",2,true);
		var col3=new DimCol(col2);
		var arr=[col1,col3]
		FilterTree1.binding.setCols(arr);

dropDown	以下拉列表的形式展现	FilterTree1.dropDown=true;
dropRow	以下拉列表的形式展现时的行数设置	FilterTree1.dropRow=3;

SeletedObjects	设定勾选的数据	var sel = [];
		sel.push(["Decaf"]);
		sel.push(["Regular", "Small Market", "East"]);
		FilterTree1.setSelectedObjects(sel);

sortType	排序	FilterTree1.sortType=SORT_DESC;
submitOnChange	是否改变时提交	FilterTree1.submitOnChange=false;

日期组件的脚本函数

函数	说明	举例
getCol	获得当前日期组件绑定的字段信息	var a=Calendar1.binding.getCol(col1);

setCol	给当前日期组件绑定字段	var bcol=new BCol("sell_date",DATE_TIME,false);
		var c=new DateCol(bcol,QUARTER_GROUP);
		var d=new DimCol(c);
		Calendar1.binding.setCol(d);

dropDown	日期组件是否以下拉的形式展现	Calendar1.dropDown=false;
dropHeight	当日期组件以下拉的形式展现时，设定日期组件的高度	Calendar1.dropHeight=200;
firstDates	在范围模式或比较模式下，设定起始日期，日期设定格式见下表（日期格式）	Calendar1.firstDates=["y2002-5"];
mode	日期组件的模式转换，见下表（日期组件模式）	Calendar1.mode=RANGE_MODE;
secondDates	在范围模式或比较模式下，设定终止日期，日期设定格式见下表（日期格式）	Calendar1.secondDates=["y2003-6"];
showType	显示当前日期组件的日期显示类型，见下表（日期显示类型）	Calendar1.showType=1

日期组件模式

日期组件模式	相应的常量	说明
SINGLE_MODE	1	单独模式
RANGE_MODE	2	范围模式
PERIOD_MODE	3	比较模式

日期类型

日期类型	格式
季度	j2011-1(从 0-3 季度)
月份	y2011-0(从 0-11 月)
周	z2011-0-1
天	t2011-0-1

日期显示类型

日期显示类型	相应的常量	说明
QUARTER	1	当前的日期类型为季度
MONTH	2	当前的日期类型为月份
WEEK	3	当前的日期类型为周
DAY	4	当前的日期类型为天

范围组件的脚本函数

函数	说明	举例
getCol	获得当前范围组件绑定的字段信息	var a=Range1.binding.getCol(col1);

setCol	给当前范围组件绑定字段	var bcol=new BCol("customer_id",INTEGER,false);
		var d=new DimCol(bcol);
		Range1.binding.setCol(d);

includeMin	是否包含最小值	Range1.includeMin=false;
lBJumpSize	每隔多少刻度显示标签	Range1.lBJumpSize=3;
less	是否包含小于最小值的范围	Range1.less=true;
max	最大值	Range1.max=19;
maxLB	给范围组件设定最大值标签	var a="maxval"; Range1.maxLB=a;
min	最小值	Range1.max=1;
minLB	给范围组件设定最小值标签	var a="minval"; Range1.maxLB=a;
more	是否包含大于最大值的范围	Range1.more=true;
rangeStart	起始位置的设定	Range1.rangeStart=6;
showMax	是否显示最大值	Range1.showMax=false;
showMin	是否显示最小值	Range1.showMin=false;
showSelection	是否显示当前值	Range1.showSelection=false;
showTick	是否显示刻度线	Range1.showTick=false;
sparseLB	是否设定标签显示间隔	Range1.sparseLB=true;
step	步长	Range1.step=1000;

列表参数组件的脚本函数

函数	说明	举例
getLabelCol	获得当前组件绑定的标签字段信息	var a= 列表参数 1.binding.getLabelCol(col1);
getValueCol	获得当前组件绑定的值字段信息	var a= 列表参数 1.binding.getValueCol(col1);

setLabelCol	给当前复选框组件绑定字段	var col=new BCol("product",2,true);
		var col1=new DimCol(col);
		列表参数 1.binding.setLabelCol(col1);

setValueCol	设置当前组件绑定的值字段信息	var bcol1 = new BCol("id", STRING, true);
		var dimCol1 = new DimCol(bcol1);
		列表参数 1.binding.setValueCol(dimCol1);

colNumber	设定列表参数组件的显示列数	列表参数 1.colNumber=2;
dropDown	以下拉列表的形式展现	列表参数 1.dropDown=true;
dropRow	以下拉列表的形式展现时的行数设置	列表参数 1.dropRow=3;
page	显示第几页的数据 (总页数 = 总行数 / 每页行数)	列表参数 1.page=4;
pageSelection	在换页时是否全部勾选	列表参数 1.pageSelection=true;
dataRows	设定显示的数据行数	列表参数 1.dataRows=3;
SelectedObjects	给组件勾选数据	列表参数 1.setSelectedObjects(["2","3"],INTEGER);
single	转为单过滤器	列表参数 1.single=true;
sortType	排序	列表参数 1.sortType=SORT_DESC;
submitOnChange	是否改变时提交	列表参数 1.submitOnChange=false;
useDetailQuery	是否使用非聚合的查询	列表参数 1.useDetailQuery=true;

getObjects	获取组件的数据，返回二维数组	列表参数 1.setObjects(["East", "West"]); debug(列表参数 1.getObjects().length);
setObjects	设置组件的数据	列表参数 1.setObjects(["East", "West"]);
maxRows	设定组件的最大行数	列表参数 1.maxRows=3;

文本输入框组件的脚本函数

函数	说明	举例
dType	设定参数类型	TextParam1.dType=DOUBLE;
height	设定文本输入框的高度	TextParam1.height=34;
value	设定参数值	TextParam1.value=1000;

下拉列表组件的脚本函数

函数	说明	举例
getLableCol	获得当前组件标签行绑定的字段信息	var a=Combox1.binding.getLableCol(lcol);
getValueCol	获得当前组件值行绑定的字段信息	var a=Combox1.binding.getValueCol(vcol);

setLabelCol	给当前组件的标签行绑定字段	var bcol = new BCol("name", STRING, true);
		var dimCol = new DimCol(bcol);
		Combox1.binding.setLabelCol(dimCol);

setValueCol	给当前组件的值行绑定字段	var bcol1 = new BCol("id", STRING, true);
		var dimCol1 = new DimCol(bcol1);
		Combox1.binding.setValueCol(dimCol1);

dropRow	下拉列表的行数设定	ComboBox1.dropRow=3;
allowSelectNull	是否允许空选项	ComboBox1.allowSelectNull = true;
dataRows	设定数据行数	ComboBox1.dataRows = 5;
editable	是否可编辑	ComboBox1.editable=true;
fInput	表单输入框信息	var a=ComboBox1.fInput; debug(a);
page	设定页数	ComboBox1.page=3;

pageSelection	在换页时是否全部勾选	ComboBox1.pageSelection=true;
getObjects	获取组件的数据，返回二维数组	ComboBox1.setObjects(["East", "West"]); debug(ComboBox1.getObjects().length);
setObjects	设置组件的数据	ComboBox1.setObjects(["East", "West"]);
getSelectedIndex	选择值的索引	ComboBox1.getSelectedIndex();
setSelectedIndex	通过索引设置选中项	下拉参数 1.setSelectedIndex(2);// 选择第二项（不包括空选项）
sortType	排序	ComboBox1.sortType=SORT_DESC;
totalRows	总行数	ComboBox1.totalRows=5;
nullLB	空选项名称	ComboBox1.nullLB = "test";
optional	是否必选	ComboBox1.optional = true;
indexof	返回给定对象的索引	ComboBox1.indexOf("Central");
useDetailQuery	使用非聚合的查询	ComboBox1.useDetailQuery = true;

选项卡组件的脚本函数

函数	说明	举例
dir	选项卡的位置类型。//0 表示的上面，2 表示的左边	TabElement1.dir=0 ；
selIndex	表示选项卡的选择的第几项 // 从 0 开始	TabElement1.selIndex=1;
showIcon	是否显示图标	TabElement1.showIcon=true;
fitContent	自适应大小	TabElement1.fitContent = true;
padding	填充距离	TabElement1.padding = ['8', '8', '8', '8'];

图片组件的脚本函数

函数	说明	举例
fill	是否是填充图	Image1.fill=false;
image	设定图片	Image1.image="2.png";
scale	是否缩放	Image1.scale=true;
scale9Insets	设定九宫格缩放比例	var d=new Insets(22,22,24,24); Image1.scale9Insets=d;

动态计算脚本函数

本产品提供给用户更加丰富的动态计算函数，如下表所示。

动态计算函数	说明	举例

diff(Object expression, [int position])	返回两个指定值的差值。	diff(col['sales'], PREVIOUS)；
		// 计算 sales 字段的差值，当前值与前一个值的差值。
		// 如果当前值或前一个值为空，返回空。

percentDiff(Object expression, [int position])	返回两个指定值的差值百分比。	percentDiff (col['sales'], PREVIOUS);
		// 计算 sales 字段中当前值与前一个值的差值百分比。
		// 如果当前值或前一个值为空或零，返回空。

percent(Object expression, [int position])	返回占指定值的百分比。	percent (col['sales'], PREVIOUS);
		// 计算 sales 字段中当前值与前一个值的百分比。
		// 如果当前值或前一个值为空或零，返回空。

percentSum(Object expression)	当前值与总和的百分比。	percentSum(col['sales']); // 当前值除以总的求和值
percentMax(Object expression)	当前值与最大值的百分比。	percentMax(col['sales']); // 当前值除以总的最大值
percentMin(Object expression)	当前值与最小值的百分比。	percentMin(col['sales']); // 当前值除以总的最小值
percentAvg(Object expression)	当前值与总平均值的百分比值。	percentAvg(col['sales']); // 当前值除以总的平均值

movingSum(Object expression,[int prev, int next, boolean cincludedCurrent, boolean nappended])	移动计算求和 , 返回从前几个值到后几个值的和
	// prev 前几个值
	//next 后几个值
	//cincluded 是否包括当前值
	//nappended如果没有足够值，是否取空。
		movingSum(col['sales'], 2, 2, true, true);
		// 给 sales 字段，取当前值的前两个，后两个，加当前值，求和。没有足够值取空。

movingAvg(Object expression, [int prev, int next, boolean cincluded, boolean nappended])	移动计算平均值 , 返回从前几个值到后几个值的平均值	movingAvg(col['sales'], 2, 2, true, true); // 给 sales 字段，取当前值的前两个，后两个，加当前值，取平均值。没有足够值取空。
movingMax(Object expression, [int prev, int next, boolean cincluded, boolean nappended])	移动计算最大值 , 返回从前几个值，到后几个值之间的最大值	movingMax(col['sales'], 2, 2, true, true); // 给 sales 字段，取当前值的前两个，后两个，加当前值，求最大值。没有足够值取空。
movingMin(Object expression,[int prev, int next, boolean cincluded, boolean nappended])	移动计算最小值。（返回从前几个值，到后几个值之间的最小值）	movingMin(col['sales'], 2, 2, true, true); // 给 sales 字段，取当前值的前两个，后两个，加当前值，求最小值。没有足够值取空。
runningAvg(Object expression,[Object reset])	累积计算平均值 , 返回从第一个值，到当前值，累积求平均 //reset 是从哪个字段开始重新开始累积。	runningAvg(col.sales,col.city); // 计算 sales 字段的累积平均值。遇到 city 字段的分组，重新累积。

runningMin(Object expression,[Object reset])	累积计算最小值，返回从第一个值，到当前值，累积求最小值)	runningMin(col.product, col.city); // 计算 sales 字段的累积最小值。遇到 city 字段的分组，重新累积。
runningMax(Object expression,[Object reset])	累积计算最大值，返回从第一个值，到当前值，累积求最大值	runningMax(col.product, col.city); // 计算 sales 字段的累积最大值。遇到 city 字段的分组，重新累积。
runningSum(Object expression,[Object reset])	累积计算和，返回从第一个值，到当前值，累积求和)	runningSum(col.product, col.city); // 计算 sales 字段的累积求和。遇到 city 字段的分组，重新累积。

第 11 章：全局函数功能

全局函数功能是指在产品的外部定义一个函数能被产品进行调用，并且可以重复被调用。

比如，在很多报表里可能用到了同一个函数，如果在每一个报表里面都需要重新定义一下该函数，就嫌的比较繁琐，因此在产品的外部定义了此函数，用到时只需进行调用它就可以使用。

功能介绍：

使用时需要在 bi.properties 中添加一条 “script.functions.path=cmnetFunctions”，cmnetFunctions 为添加的文件的名称，如新建一个名为 test.java 的文件，内容是

```
function testAdd(a,b) {
   return a + b;
}
```

在 bi.properties 中配置：script.functions.path=test.java

那么在 dashboard 中新建一个文本组件，然后右键选择脚本，在脚本中输入

文本 1.data = testAdd("Hello ","Yonghong");

如下图所示：


文本1 脚本
装载时运行
提交时运行
仪表盘
输入搜索文字
文本1
binding
active:boolean
commit:boolean
1 文本1.data=testAdd("Hello ","Yonghong");
校验JS脚本语法
确定(O)
取消(C)
应用(A)

那么显示出来的效果是 :


Hello Yonghong