Документация

JS Доступ к выходным наборам данных

Для доступа к данным выходных портов `Выходной набор данных[N]` используются объекты `OutputTables[]` и `OutputTable`. Обращение к источнику данных порта происходит через его номер:

`OutputTables[N]`, где N — номер (индекс) порта. Первый порт имеет индекс 0.

Поскольку первый порт присутствует в узле *JavaScript* по умолчанию, для доступа к его данным существует отдельный объект `OutputTable`.

Свойства выходных портов

▼ Columns

Columns

Содержит доступную для чтения коллекцию столбцов выходного набора данных. Возвращает объект, реализующий интерфейс `IColumns` (см. Полное описание API).

▼ ColumnCount

ColumnCount

Содержит доступное для чтения количество столбцов выходного набора данных. Возвращает значение типа `number`.

▼ RowCount

RowCount

Содержит доступное для чтения количество строк выходного набора данных. Возвращает значение типа `number`.

Методы OutputTable

▼ Get

Get(row, col)

- row — индекс строки. Принимает значение типа `number`.
- col — индекс или имя столбца. Принимает значение типов `number` или `string`.

Метод возвращает значение заданного столбца в заданной строке. Возвращаемое значение может иметь типы: `boolean`, `number`, `string`, `Date`, `undefined`.

IsNull

IsNull(row, col)

- row — индекс строки. Принимает значение типа `number`.
- col — индекс или имя столбца. Принимает значение типов `number` или `string`.

Метод возвращает булево значение `true`, если столбец в заданной строке имеет пропущенное значение. В противном случае возвращается `false`.

GetColumn

GetColumn(col)

- col — индекс или имя столбца. Принимает значение типов `number` или `string`.

Метод возвращает объект столбца, реализующий интерфейс `IColumn` (см. Полное описание API).

AssignColumns

AssignColumns(array)

- array — итерируемый объект, содержащий значения типа `string` или объекты, реализующие интерфейс `IColumnInfo` (см. Полное описание API).

Метод создает столбцы выходного набора из коллекции имен столбцов или объектов, реализующих интерфейс `IColumnInfo`.

AddColumn

AddColumn(columninfo)

- columninfo — значение типа `string` или объект, реализующий интерфейс `IColumnInfo` (см. Полное описание API). Необязательный аргумент.

Метод добавляет столбец в конец списка столбцов выходного набора, принимая в качестве аргумента имя столбца или объект, реализующий интерфейс `IColumnInfo`. Возвращает объект, реализующий интерфейс `IOutputColumn` (см. Полное описание API).

InsertColumn

InsertColumn(col, columninfo)

- col — индекса столбца. Принимает значение типа `number`.
- columninfo — объект, реализующий интерфейс `IColumnInfo` (см. Полное описание API). Необязательный аргумент.

Метод вставляет столбец по заданному индексу в выходной набор. Возвращает объект, реализующий интерфейс `IOutputColumn` (см. Полное описание API).

DeleteColumn

DeleteColumn(col)

- col — индекс или имя столбца. Принимает значение типов `number` или `string`.

Метод удаляет столбец по имени или индексу.

ClearColumns

ClearColumns()

Не имеет аргументов. Метод очищает список столбцов.

Append

Append()

Метод добавляет новую строку в выходной набор данных. Не имеет аргументов.

Set

Set(col, value)

- col — индекс или имя столбца. Принимает значение типов `number` или `string`.
- value — значение. Принимает значения следующих типов: `boolean`, `number`, `string`, `Date`, `null`, `undefined`.

Метод задает значение заданного столбца в строке, добавленной методом `Append()`.

Примеры

```
import { OutputTable, OutputTables, DataType, DataKind, UsageType } from
"builtIn/Data";

let outputTable0 = OutputTables[0],     // Набор данных выходного порта №1
    outputTable1 = OutputTables[1];     // Набор данных выходного порта №2
let colProductName0 = outputTable0.Columns.ProductName,     // Получение
ссылки на столбец по имени
    colProductID0 = outputTable0.GetColumn("ProductID");
let colProductName1 = outputTable1.Columns[0],              // Получение
ссылки на столбец по индексу
    colProductID1 = outputTable1.GetColumn(1);

let array = [];
for (let i = 0; i < 3; i++){
    array.push({Name: `Test${i}`, DisplayName: `Тест${i}`, DataType:
DataType.Integer, DefaultUsageType: UsageType.Active});
}

// Добавление массива столбцов
OutputTable.AssignColumns(array);

// Удаление столбца по индексу
OutputTable.DeleteColumn(0);

// Удаление столбца по имени
OutputTable.DeleteColumn("Test1");

// Удаление всего списка столбцов
OutputTable.ClearColumns();

// Добавление столбца в конец списка столбцов выходного набора
OutputTable.AddColumn({Name: "COL0",
                       DisplayName: "Дата/Время",
                       DataType: DataType.DateTime,
                       DataKind: DataKind.Continuous,
                       DefaultUsageType: UsageType.Active});

// Вставка столбца по заданному индексу в список столбцов выходного набора
OutputTable.InsertColumn(0, {Name: "COL1",
```

```
                                DisplayName: "Признак",
                                DataType: DataType.Boolean});

// Получение ссылки на столбец по имени
let COL0 = OutputTable.GetColumn("COL0");
let COL1 = OutputTable.Columns.COL1;

// Вывод значений свойств столбца
console.log("Index: ", COL1.Index);
console.log("Name: ", COL1.Name);
console.log("DisplayName: ", COL1.DisplayName);
console.log("DataType: ", COL1.DataType);
console.log("DataKind: ", COL1.DataKind);
console.log("DefaultUsageType: ", COL1.DefaultUsageType);

// Добавление строки в выходной набор данных
OutputTable.Append();

// В поле с индексом 0 записываются текущие Дата/Время
OutputTable.Columns[0].Set(new Date());

// В поле с индексом 1 записывается значение true
OutputTable.GetColumn(1).Set(true);

// Копирование значений первой строки во вторую
OutputTable.Append();
for (let i = 0, c = OutputTable.ColumnCount; i < c; i++) {
    let value = OutputTable.Get(0, i);
    OutputTable.Set(i, value);
}

// Проверка, что значение в столбце с индексом 1 не определено
console.assert(OutputTable.IsNull(0, 1));
console.assert(typeof OutputTable.Get(0, 1) == "undefined");


console.log("RowCount = ", OutputTable.RowCount);
// Вывод: RowCount =  2
```