JS Accessing the Output Dataset

To access the data of the node's output ports `Output dataset[N]`, use the `OutputTable` an `OutputTables[]` objects. The datasets are referenced using their index numbers (`N`):

```
OutputTables[N]
```

The first port has an index number `0`. As it is included in the **JavaScript** node by default, it is accessed using a dedicated object `OutputTable`.

OutputTable properties

▼ Columns

`Columns`

Provides a read-only collection of the columns in the output dataset, returning an object that implements the `IColumns` interface (see the full API description for more information).

▼ ColumnCount

`ColumnCount`

Provides the total number of columns in the output dataset. This property is read-only. Returned value: `number`.

▼ RowCount

`RowCount`

Provides the total number of rows in the output dataset (read-only). Returned value: `number`.

OutputTable methods

▼ Get

`Get(row, col)`

- `row`: a row index. Takes a `number` value.

- `col` : a column index or name. Takes a `number` or `string` value.

Returns the value of a specified column within a specified row. Property value: `boolean` , `number` , `string` , `Date` , or `undefined` .

▼ IsNull

`IsNull(row, col)`

- `row` : a row index. Takes a `number` value.
- `col` : a column index or name. Takes a `number` or `string` value.

Checks if the value in a specific cell (identified by its column and row) is null. Returns `true` if the value is null, and `false` otherwise.

▼ GetColumn

`GetColumn(col)`

- `col` : a column index or name. Accepts `number` or `string` values.

Returns a column object, which implements the `IColumn` interface. See the full API description for more details.

▼ AssignColumns

`AssignColumns(array)`

- `array` : an iteratable object containing `string` type values or objects that implement the `IColumnInfo` interface (see the full API description for details).

The method generates output dataset columns from the collection of column names or objects implementing the `IColumnInfo` interface.

▼ AddColumn

`AddColumn(columninfo)`

- `columninfo` (optional): a `string` value or an object that implements the `IColumnInfo` interface. See the full API description.

Adds a column to the end of output dataset column list, returning an object that implements the `IOutputColumn` interface (see the full API description for details). The method accepts either a column name or an `IColumnInfo` object as an argument.

▼ InsertColumn

`InsertColumn(col, columninfo)`

- `col` : a column index. Takes a `number` value.
- `columninfo` (optional): an object that implements the `IColumnInfo` interface. See the full API description for more details.

The method inserts a column into the output dataset at a specific index. Returns an object that implements the `IOutputColumn` interface. See the full API description for more details.

▼ DeleteColumn

```
DeleteColumn(col)
```

- `col` : a column name or index. Takes a `string` or `number` value.

The method deletes the column specified by its name or index.

▼ ClearColumns

```
ClearColumns()
```

The method deletes all columns from the list. No arguments are required.

▼ Append

```
Append()
```

The method adds a new row to the output dataset. No arguments are required.

▼ Set

Set(col, value)

- `col` : a column index or name. Takes a `number` or `string` value.
- `value` : A value of one of the following types: `boolean` , `number` , `string` , `Date` , `null` , or `undefined` .

The method sets the value of a specified column in a row that was added by the `Append()` method.

Examples:

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

let outputTable0 = OutputTables[0],     // Dataset from output port #1
    outputTable1 = OutputTables[1];     // Dataset from output port #2
let colProductName0 = outputTable0.Columns.ProductName,     // Getting
column link by name
    colProductID0 = outputTable0.GetColumn("ProductID");
let colProductName1 = outputTable1.Columns[0],              // Getting
column link by index
    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});
}

// Adding a column array
OutputTable.AssignColumns(array);

// Deleting a column by index
OutputTable.DeleteColumn(0);

// Deleting a column by name
OutputTable.DeleteColumn("Test1");

// Deleting the entire list of columns
OutputTable.ClearColumns();

// Adding a column столбца to the end of the output set column list
OutputTable.AddColumn({Name: "COL0",
                       DisplayName: "Date/Time",
                       DataType: DataType.DateTime,
                       DataKind: DataKind.Continuous,
                       DefaultUsageType: UsageType.Active});

// Inserting a column by the specified index to the output set column list
OutputTable.InsertColumn(0, {Name: "COL1",
```

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

// Referencing a column by name
let COL0 = OutputTable.GetColumn("COL0");
let COL1 = OutputTable.Columns.COL1;

// Outputting the values of column properties
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);

// Appending a row to the output dataset
OutputTable.Append();

// Writing the current Date/Time to the field with index 0
OutputTable.Columns[0].Set(new Date());

// Writing true to the field with index 1
OutputTable.GetColumn(1).Set(true);

// Copying the values from the first row to the second row
OutputTable.Append();
for (let i = 0, c = OutputTable.ColumnCount; i < c; i++) {
    let value = OutputTable.Get(0, i);
    OutputTable.Set(i, value);
}

// Checking if the value in the column with index 1 is undefined
console.assert(OutputTable.IsNull(0, 1));
console.assert(typeof OutputTable.Get(0, 1) == "undefined");


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

Read on: JS Global Functions