JS Accessing Input Datasets

Data from the node's input ports is accessed using the `InputTable` and `InputTables[]` objects:

- Use `InputTables[N]` to access the port with index *N*. Indexing is zero-based (i.e., the first port is `InputTables[0]`).
- For convenience, you can also use the `InputTable` object as a direct shortcut to access the first input port (present in the node by default). It is equivalent to `InputTables[0]`.

Data source properties

▼ Columns

`Columns`

Provides a read-only collection of columns, returning an object that implements the `IInputColumns` interface. Each collection item implements the `IInputColumn` interface. See the full API description for more information.

▼ ColumnCount

`ColumnCount`

Contains a read-only count of columns. Returns a `number` value.

▼ RowCount

`RowCount`

Contains a read-only count of rows. Returns a `number` value.

Data source methods

▼ Get

`Get(row, col)`

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

The method returns the value of the specified column in the specified row. The returned value can be `boolean`, `number`, `string`, `Date`, or `undefined`.

▼ IsNull

`IsNull(row, col)`

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

The method returns the boolean value `true` if the column in the specified row has the null value, and the `false` value otherwise.

▼ GetColumn

`GetColumn(col)`

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

The method returns the column's object which implements the `IInputColumn` interface (see Full API Description).

Examples:

```
import { InputTable, InputTables } from "builtIn/Data";

let inputTable0 = InputTables[0],              // Port 1 data source
    inputTable1 = InputTables[1];              // Port 2 data source
let colOutlook0 = inputTable0.Columns.OUTLOOK, // Referencing the column
by name
    colOutlook1 = inputTable1.Columns[1];      // Referencing the column
by index

// Getting the columns array from the Columns object
let arrayOfColumns = Array.from(InputTable.Columns);
// Outputting the column property values to the console
arrayOfColumns.forEach(column => {
    console.log("Index: ", column.Index);
    console.log("Name: ", column.Name);
    console.log("DisplayName: ", column.DisplayName);
    console.log("DataType: ", column.DataType);
    console.log("DataKind: ", column.DataKind);
    console.log("DefaultUsageType: ", column.DefaultUsageType);
    console.log("RowCount: ", column.RowCount);
    console.log("");
});

// Getting the values array from the "CLASS" column
let arrayOfColumnValues = Array.from(InputTable.Columns["CLASS"]);
// Outputting the "CLASS" column values
arrayOfColumnValues.forEach((value, index) => {
    console.log(index, ":", value);
});

// Reading the values from the input table using the Get method
for (let i = 0, c = InputTable.RowCount; i < c; i++) {
    // Outputting the values of the column indexed 0
    console.log(`InputTable.Get(${i}, 0) = `, InputTable.Get(i, 0));
    // Outputting the values of the column named "CLASS"
    console.log(`InputTable.Get(${i}, "CLASS") = `, InputTable.Get(i,
"CLASS"));
}
```

Read on: JS Input Variables