Converters

Converters are pieces of JavaScript code that take the result from a query (and optionally metadata), process it into a custom array of objects and then pass it on to the template. Of course you can do arbitrary calculations, formattings and renamings there too. Converters are required mostly when dealing with JSON responses from REST APIs, but they can be applied to SQL results as well.

Properties

Each converter entity consists of the following properties:

- **Name**: The Name identifying the converter. Names must only contain letters [A-Za-z_-], numbers, hyphens and underscores.
- **Content**: Javascript code that receives the query result and returns an array of objects. Use the text link on the top right above the input field to insert the scaffold. In most cases, you will only need to adjust the code within the method call *result.push()*, where you walk through the result line by line and extract the information you need.

For various examples on how to use converters, refer to our List Of REST Examples.

Note: since version 3.7.0 converters may optionally receive a second object as argument that contains the following query:

- **queryName** (String)*:* the name of the query currently executing this converter
- **queryParameters** (Object): the parameters of the query currently executing this converter
- **originalQueryParameters** (Object): same as *queryParameter* but before any processing took place (e.g. wildcard replacements)

Also see the last example below for more information how to use the second argument.

Example Converters

Converter scaffold:

```
function convert(json) { // json is the result as json string
    var result = []; // the array-of-objects we will return
    var parsedJsonObject = JSON.parse(json); // parse json string
    var current, index; // loop variables
    for (index in parsedJsonObject) { // iterate through the result
        // only continue if this property is not inherited
        if (parsedJsonObject.hasOwnProperty(index)) {
            current = parsedJsonObject[index]; // current object
            result.push({ // add a converted object
                'Column 1 name': current.column1Name,
                'Column 2 name': current.column2Name
                // ...
            });
        }
    }
    return result; // return the converted result
}
```

Example converter for a facebook REST call that fetches all pages liked by the user:

```
function convert(json) {
    var result = [];
    var parsedJsonObject = JSON.parse(json);
    var current, index;
    for (index in parsedJsonObject) {
        if (parsedJsonObject.hasOwnProperty(index)) {
            current = parsedJsonObject[index];
            result.push({
                'Page name': current.name,
                'Date of pressing Like': PocketQuery.formatDate(current.created_time,'YYYY-MM-DD (dddd), hh:
mm') + ' Uhr'
            });
        }
    }
    return result;
}
```

Example converter using metadata (available since version 3.7.0):

```
function convert(json, metadata) {
        return [{
                "QueryName": metadata.queryName,
                "QueryParameter Continent": metadata.queryParameters.Continent,
                "QueryParameter MinPopulation": metadata.queryParameters.MinPopulation,
                "Original QueryParameter Continent": metadata.originalQueryParameters.Continent,
                "Original QueryParameter MinPopulation": metadata.originalQueryParameters.MinPopulation
        }];
}
```