Data-Forge cheat sheet

Run this notebook with Data-Forge Notebook

Snippets of JS code that show how to work with data using Data-Forge.

From the book Data Wrangling with JavaScript

For more in-depth help please see the The Guide or the API docs.

Loading data into a dataframe

Load data from memory into a Data-Forge `DataFrame`.

```
let data = [{ A: 1, B: 10 }, { A: 2, B: 20 }, { A: 3, B: 30 }];
let df = new dataForge.DataFrame(data);
display(df);
```

__index__	A	B
0	1	10
1	2	20
2	3	30

Loading CSV files

Load data from a CSV file using `readFile` and `parseCSV`.

```
let df = await dataForge
    .readFile("./example.csv", { dynamicTyping: true })
    .parseCSV();
display(df.head(5)); // Preview first 5 rows.
```

__index__	Name	Sex	Age	Height (in)	Weight (lbs)
0	Alex	M	41	74	170
1	Bert	M	42	68	166
2	Carl	M	32	70	155
3	Dave	M	39	72	167
4	Elly	F	30	66	124

Loading JSON files

Load data from JSON files using `readFile` and `parseJSON`.

```
let df = await dataForge
    .readFile("./example.json")
    .parseJSON();
display(df.tail(5)); // Preview last 5 rows.
```

__index__	Name	Sex	Age	Height (in)	Weight (lbs)
13	Neil	M	36	75	160
14	Omar	M	38	70	145
15	Page	F	31	67	135
16	Quin	M	29	71	176
17	Ruth	F	28	65	131

Data transformation

Transform or rewrite your data set using the `select` function (similar to JavaScript's `map` function):

Example: Transforming the value of the Height column from inches to centimetres.

```
let df = await dataForge
    .readFile("./example.csv", { dynamicTyping: true })
    .parseCSV();

let transformed = df.select(row => {                          // Transform the dataframe, convert
        const clone = Object.assign({}, row);                 // Clone the original so that we do
        clone["Height (cm)"] = clone["Height (in)"] * 2.54; // Convert from inches to centimetre
        return clone;
    });

display(transformed.head(5));
```

__index__	Name	Sex	Age	Height (in)	Weight (lbs)	Height (cm)
0	Alex	M	41	74	170	187.96
1	Bert	M	42	68	166	172.72
2	Carl	M	32	70	155	177.8
3	Dave	M	39	72	167	182.88
4	Elly	F	30	66	124	167.64000000000001

Data filtering

Filter data with the the `where` function (similar to JavaScript's `filter` function).

Example: Filtering for tall people.

```
let df = await dataForge
    .readFile("./example.json")
    .parseJSON();

let filtered = df.where(row => row["Height (in)"] >= 70); // Filter for very tall people.

display(filtered);
```

__index__	Name	Sex	Age	Height (in)	Weight (lbs)
0	Alex	M	41	74	170
2	Carl	M	32	70	155
3	Dave	M	39	72	167
7	Hank	M	30	71	158
8	Ivan	M	53	72	175
11	Luke	M	34	72	163
13	Neil	M	36	75	160
14	Omar	M	38	70	145
16	Quin	M	29	71	176

Working with series (columns)

Removing one or more series

Example: Removing the Height and Weight columns using the `dropSeries` function.

```
let df = await dataForge.readFile("./example.json").parseJSON();

let modified = df.dropSeries(["Height (in)", "Weight (lbs)"]);

display(modified.head(3));
```

__index__	Name	Sex	Age
0	Alex	M	41
1	Bert	M	42
2	Carl	M	32

Renaming one or more series

Example: Renaming the Height and Weight columns using the `renameSeries` function so that the field names don't specify the unit of measurement.

```
let df = await dataForge.readFile("./example.json").parseJSON();

let modified = df.renameSeries({
    "Height (in)": "Height",
    "Weight (lbs)": "Weight",
});

display(modified.head(3));
```

__index__	Name	Sex	Age	Height	Weight
0	Alex	M	41	74	170
1	Bert	M	42	68	166
2	Carl	M	32	70	155

Extracting, transforming and merging a series

Example: converting the Height column from inches to centimeters.

```
let df = await dataForge.readFile("./example.json").parseJSON();
df = df.renameSeries({ "Height (in)": "Height"})    // Rename series.
    .setIndex("Name");                              // We need an index in order to merge data.

let heights = df.getSeries("Height");

// You can also do this:
// let heights = df.deflate(row => row.Height);

heights = heights.select(value => value * 2.54); // Convert from inches to centimeters.

df = df.withSeries("Height", heights); // Merge the modified series into the source data set.

display(df.head(3));
```

__index__	Name	Sex	Age	Height	Weight (lbs)
Alex	Alex	M	41	187.96	170
Bert	Bert	M	42	172.72	166
Carl	Carl	M	32	177.8	155

A simpler way to transform a series

Example: Using the DataFrame `transformSeries` function makes the previous example a bit simpler.

```
let df = await dataForge.readFile("./example.json").parseJSON();
df = df.renameSeries({ "Height (in)": "Height" });    // Rename series.
```

```
df = df.transformSeries({ Height: value => value * 2.54 }); // Convert Height from inches to cen

display(df.head(3));
```

__index__	Name	Sex	Age	Height	Weight (lbs)
0	Alex	M	41	187.96	170
1	Bert	M	42	172.72	166
2	Carl	M	32	177.8	155

Group and summarize

We can use the `groupBy` function to group our data set and then boil each group down to a summary.

Example: Getting the average height and weight for male and female groups.

```
let df = await dataForge.readFile("./example.json").parseJSON();

df = df.renameSeries({
    "Height (in)": "Height",
    "Weight (lbs)": "Weight",
});

let summary = df.groupBy(row => row.Sex) // Sort the data set into groups. This returns a series
    .select(group => {     // Transform each group into a summary.
        return {
            Sex: group.first().Sex,
            Count: group.count(),
            Height: group.deflate(row => row.Height).average(),
            Weight: group.deflate(row => row.Weight).average(),
        };
    })
    .inflate(); // Inflate the series to a dataframe (groupBy returns a series);

display(summary);
```

__index__	Sex	Count	Height	Weight
0	M	11	71.27272727272727	161.63636363636363
1	F	7	65.57142857142857	123.28571428571429

Aggregation

We can use the `aggregate` function (like the JavaScript `reduce` function) to boil our entire data set down to a simple summary.

Example: Get the average height and weight for the entire group.

```
let df = await dataForge.readFile("./example.json").parseJSON();

df = df.renameSeries({
    "Height (in)": "Height",
    "Weight (lbs)": "Weight",
});

let summary = df.aggregate((agg, row) => ({
    Height: (agg.Height + row.Height) / 2,
    Weight: (agg.Weight + row.Weight) / 2,
}));

display(summary);
```


```
"root" : {  2 items
   "Height" : 67.41633605957031
   "Weight" : 144.37367248535156
}
```

Save CSV files

Save your modified data to a CSV file using functions `asCSV` and `writeFile`.

```
let df = await dataForge
    .readFile("./example.csv", { dynamicTyping: true })
    .parseCSV();

let transformed = df.select(row => { // Transform data.
    const clone = Object.assign({}, row);
    clone["Height (cm)"] = clone["Height (in)"] * 2.54;
    return clone;
});

await df.asCSV().writeFile("./transformed.csv"); // Save CSV file.
```

Save JSON files

Save your modified data to a JSON file using function `asJSON` and `writeFile`.

```
let df = await dataForge
    .readFile("./example.json")
    .parseJSON();

let transformed = df.select(row => { // Transform data.
    const clone = Object.assign({}, row);
    clone["Height (cm)"] = clone["Height (in)"] * 2.54;
    return clone;
});

await df.asJSON().writeFile("./transformed.json"); // Save JSON file.
```

Getting data from a REST API

Use the `axios` module to retreive data from a REST API (with data from JSONPlaceholder).

```
const axios = require('axios');
const response = await axios("https://jsonplaceholder.typicode.com/todos");
const data = new dataForge.DataFrame(response.data);
display(data.head(5));
```

__index__	userId	id	title	completed
0	1	1	delectus aut autem	false
1	1	2	quis ut nam facilis et officia qui	false
2	1	3	fugiat veniam minus	false
3	1	4	et porro tempora	true
4	1	5	laboriosam mollitia et enim quasi adipisci quia provident illum	false