Open MCT Coding exercise

This exercise will involve connecting to a telemetry data server, and displaying the telemetry produced by it in a table. This will require you to demonstrate the ability to issue HTTP requests, manage WebSocket connections, and update a web page using JavaScript.

For this exercise we require you to use native ES6 JavaScript, HTML, and CSS, without relying on any JavaScript libraries or frameworks. HTML and CSS frameworks and themes are OK so long as they're used simply for layout, and look and feel. (eg. Bootstrap is fine; React, Angular, Vue, lodash etc. are not)

In this exercise we will be using the Open MCT tutorial server as the source of telemetry data. The Open MCT tutorial server provides a very simple simulation of a spacecraft generating data for various "telemetry points". Telemetry points are the term we use to describe measurable "things" on the spacecraft. These might be sensors or instruments measuring things like temperature, voltage, current, etc. in different systems on the spacecraft.

Exercise

Write a single page JavaScript web application that will:

1. Query the given telemetry server for the last 15 minutes of historical telemetry data for either or both of the "pwr.v" and "pwr.c" telemetry points. The telemetry points should be selectable by the user. Changing the selected telemetry points should result in only data from the selected telemetry points being shown in the table. ie. data for a deselected telemetry point should be cleared from the table.
2. Display the returned telemetry data in an HTML table sorted ascending or descending by timestamp. The sort order should be selectable by the user.
3. Subscribe for new telemetry from the selected telemetry points and add rows to the table as new data becomes available, maintaining the selected sort order. When the user changes the selected telemetry point, you should only maintain subscriptions to selected telemetry points, and any telemetry data for a deselected telemetry point should be removed.

An example of the expected table layout is given below:

ID	**Timestamp**	**Value**
pwr.v	2019-06-29T22:43:16.179Z	30.902893460672324
pwr.c	2019-06-29T22:43:16.179Z	7.907333750000001

pwr.v	2019-06-29T22:44:16.179Z	30.04244516070705
pwr.c	2019-06-29T22:44:16.179Z	7.788723743750001

Please spend no more than 4 hours on this exercise.

You should provide your solution in the form of a public (or private, if you prefer) GitHub repository that includes a README file describing how to run it.

Your code will be assessed on:

- Correctness - does it do what we're asking?
- Code organization
- Code cleanliness
- Demonstrated knowledge of ES6

Dependencies

This exercise requires git and node to be installed in order to run the telemetry server.

Downloading and Running the Telemetry Server

```
git clone https://github.com/nasa/openmct-tutorial.git
cd openmct-tutorial
npm install
npm start
```

This will expose two endpoints

- `http://localhost:8080/history` - An http server for "historical" data queries.
- `ws://localhost:8080/realtime` - A websocket endpoint for subscribing to realtime data.

These are detailed in the following sections.

Requesting Historical Data

A note on Cross-origin Resource Sharing (CORS)

By default your browser will not allow requests to a server hosted on another port unless the server explicitly allows it. In order to enable cross-origin requests from the Open MCT tutorial server, you will need to add the following to `example-server/server.js`

```
app.use(function(req, res, next) {
     res.header("Access-Control-Allow-Origin",
"http://localhost:8081");
     res.header("Access-Control-Allow-Headers", "Origin,
X-Requested-With, Content-Type, Accept");
     next();
+});
```

Where `http://localhost:8081` should be replaced with the details of the http server hosting your web application.

Query Parameters

`http://localhost:8080/history/:pointId?start=:start&end=:end`

- `:pointId` is the ID of the telemetry point you're requesting data for. In this case we will use the `pwr.v` telemetry point, which simulates spacecraft voltage.
- `:start` and `:end` determine the historical period that you're requesting data for, and are represented as milliseconds since 01/01/1970. The output of Date.now() uses this convention. An example is provided below.

Example Request

`http://localhost:8080/history/pwr.v?start=1561846955890&end=1561848320152`

This will request all available telemetry data for the "pwr.v" telemetry point between 2019-06-29T22:22:35.890 UTC and 2019-06-29T22:45:20.152 UTC. The example server will only begin to generate data when it is run, and the data will only exist while the server is running, so your server will not have any data for the period specified in this example.

Example Response

Data is returned as an array of JSON objects.

Eg.

`[{"timestamp":1561848196179,"value":241.70507498990156,"id":"pwr.v"},{"timestamp":1561848197181,"value":238.10697171005611,"id":"pwr.v"},{"timestamp":1561848198182,"value":233.80084678058353,"id":"pwr.v"}]`

Real-time Data Subscriptions

WebSocket Endpoint

`ws://localhost:8080/realtime`

A websocket endpoint for subscribing to realtime data. This will accept websocket subscriptions. A subscription can be started by sending a Subscription message specifying the telemetry points to subscribe to.

Subscribe Message

A new subscription can be initiated by connecting to the WebSocket endpoint specified previously, and sending a subscription message of the form

```
subscribe :pointId
```

- `:pointId` is the ID of the telemetry point to subscribe to.

Example

```
subscribe pwr.v
```

Unsubscribe Message

A subscription can be ended by sending an unsubscribe message over an established WebSocket connection. This message takes the form

```
unsubscribe :pointId
```

- `:pointId` is the ID of the telemetry point to unsubscribe from.

Example

```
unsubscribe pwr.v
```

Telemetry Data Messages

When new telemetry becomes available it will be sent over the WebSocket connection as JSON formatted of the same form as historical telemetry data, eg.

{"timestamp":1561848196179,"value":241.70507498990156,"id":"pwr.v"}