Sauce Labs Front-end Engineer Coding Challenge 1

This document describes a coding challenge for Front-End Engineer position candidates as Sauce Labs.

Interactive Scatterplot Component

Please take a look at `scatterplot.png`. This Scatterplot is mock-up that needs to be coded as a component.

Within your code should be a variable called “plotpoints” that contains an array of dictionaries.

Each dictionary in the "plotpoints" array represents a single point on the scatterplot. The three variables in the dictionaries are defined as:

- start_time: RFC3339 formatted string, e.g. "2017-11-29T04:56:12Z"
- status: string — one of “pass”, “error”, or “fail"
- duration: integer representing seconds

The string in the "status" variable maps to these three colors in the graph:

- "pass" --> green
- "error" --> orange
- "fail" --> red

Additionally, the dots should be clickable such that they have a “selected” style that appears when clicked. They should return to “unselected” style when clicked again.

You will need to create some more sample data yourself (see below in Example for a starting point). Make sure there are at least 20 sample data points spanning a few days.

Acceptance Criteria

1. X-axis represents a timeline.
2. Y-axis represents a duration.
3. Every dot in "plotpoints" should appear on the scatterplot.
4. Correctly map the "status" string into the appropriate color.
5. A click on a dot should toggle its style between `normal` and `selected`.
6. Component dimensions should adapt to container size.
7. Component should properly scale both axes depending on data. For example, if data contains items from 7 days, X-axis should start from T1 and end with T1 + 7 days.

Stretch Goals

1. Create a back-end service API that provides the "plotpoints" data when requested by your component.
2. Provide a method within the component to allow the user to adjust the time-scale (i.e. the range of points along their start_time -- the X axis).
3. If you make the first 2 stretch goals, have the time-scale part of your component request this specific range of points from the API. (So, the API would only return the desired range of plotpoints.)

Technical Requirements

1. Project should be available as a git repository.
2. Write a README to enable another developer to easily install dependencies and run the project. (We recommend to follow standard `npm` commands like `npm install` and `npm run`.)
3. There should be a way to start a local server in Terminal and open the component in a browser following an URL.
4. Project will be tested on the newest node.js LTS version, Chrome and Firefox browsers on Linux or macOS
5. It should be written in ES2015+.
6. Any tool like Webpack, Babel or SASS can be used.
7. Any library or framework can be used but React is preferable.

Example

If you decide to use React, the usage of the component may look like this:

```
const plotpoints = [
  {
    "start_time": "2017-11-29T04:56:12Z",
    "status": "pass",
    "duration": 126, // in seconds
  },
  {
    "start_time": "2017-11-28T03:22:12Z",
    "status": "error",
    "duration": 205,
  },
  {
    "start_time": "2017-11-28T02:24:12Z",
    "status": "fail",
    "duration": 20,
  },
  {
    "start_time": "2017-11-28T05:24:12Z",
    "status": "pass",
    "duration": 90,
  },
  {
    "start_time": "2017-11-29T06:24:12Z",
```

```
    "status": "error",
    "duration": 90,
  },
  {
    "start_time": "2017-11-28T14:12:12Z",
    "status": "pass",
    "duration": 200,
  }

];

const handleOnTestRunSelect = (item) => { ... };

<TestRunsScatterplot
  data={plotpoints}
  onTestRunSelect={handleOnTestRunSelect}
/>
```

Implementation Notes

A reasonable upper-limit for duration would be 300 seconds. Lower-limit should be 0 seconds.

Questions

For any engineering related questions, please contact naomi.most@saucelabs.com and adamb@saucelabs.com.