README.md 10.7 KB

Assessment: GDL javascript software engineer

This document describes the assessment we'd like you to do, with the aim of giving us a bit more insights in your abilities, development approach, and motivations to take this role.

The assigment in short

In short, the assignment is to create a simple version of the Global Data Layer (GDL) library, and use it to collect data from a provided sample application.

The assignment is intended to take you somewhere between 4 and 8 hours to complete, depending on the amount of detail you want to apply. We will look for cleanly coded solutions which will provide us a good idea of what your work will look like once you really come to work for FedEx.

After you have completed it and shared your output with us, we will invite you for a followup session in which we will ask you to present your result and process leading up to it, after which we will try to ask you a few interesting questions!

Since you have made it to the assessment round, we have good faith that you will be able to complete a quality result, and we wish you a lot of fun and good luck!

First of all you will find some context; read further down for all the details of the assigment.

About the Global Data Layer

The core purpose of this javascript libary is to facilitate data collection from web pages and applications in a generic way. This provides flexibility: the data can be ported to multiple analytics tools with ease, as well as standardisation: all teams within the FedEx Digital community have the same solution to work with.

All valid events published to GDL can result in the following actions:

1. Storage of the relevant data in the dataLayer object maintained by GDL. The datalayer should capture the data in so that at all times it contains a useful representation of the current state of the application.
2. Transmission of the event towards one or more analytics platforms. Depending on the requirements of the platform, certain transformations of the data (in the event, or in the datalayer) have to be applied before it can be forwarded.

A GDL event has the following structure:

property	type	description
name	`string`	identifies the event, describes what happened in the app
payload	`any`	relevant context for the event

This assignment already contains a basic skeleton for GDL, inside the `src` directory. It will be up to you to add / modify the code you find there to complete this assignment.

The core of the library is formed by Redux, which provides a mechanism for state management, but which also provides a neat solution to handle side effects (such as calls to analytics platforms) on the same actions that pass through it.

About the sample application

We have created a very simple "FedEx shipping app", with which users are able to book their FedEx shipment providing only very sketchy details. To be honest, we're not quite sure how packages ever get from A to B with this tool, but that's not really our problem is it. ;)

We encourage you to investigate the application by using it yourself to see exactly how it works. It is quite straightforward.

The source for the sample application is in the `sample-app` directory. Feel free to have a look there as well, but know that for this assignment we don't expect you to change any code of the app.

The tracking plan

In the sample application, the following tracking plan has been implemented for you to use:

description	event	payload type
The page loads	`pageInfo`	`{ pageId: string }`

description	event	payload type
The user arrives at one of the steps	`viewStep`	`{ stepNumber: number, stepName: string }`
Completion of the 'origin' step	`originComplete`	`{ country: string, city: string }`
Completion of the 'destination' step	`destinationComplete`	`{ country: string, city: string }`
Completion of the 'packages' step	`packagesComplete`	`{ packageCount: number }`
User has given all input and sees the price	`viewRate`	`{ currency: string, amount: number }`
User has finalized the shipment	`shipmentComplete`	n/a
User clicks the 'ship again' button	`shipAgain`	n/a
An error occurs in the application	`error`	`{ id: number, message: string }`

Requirements overview

Below is an overview the 4 individual requirements of the assignment. For each part we will go in-depth in the next sections.

1. Implement logic which updates the data layer according to the specifications.
2. Implement the "Analytics" tracker functionality according to the specifications.
3. Create a useful debugging mechanism which provides details of the events being published, and how they are handled by GDL.
4. Write some unit tests for the code you have created.

Part 1: Datalayer specification

The datalayer is a simple Javascript object which can be inspected in the brower by accessing: `window.gdl.dataLayer`. Its goal is to provide a useful representation of the current state of the application.

The datalayer should consist of two namespaces: `page` and `app`. It is your job to populate these namespaces as the data comes in through GDL events.

The `page` namespace should have the following structure. Update the contents of the namespace upon receiving the `pageInfo` event.

property	type	comment
`pageId`	`string`	collected via `pageInfo`
`countryCode`	`string`	derive this property - section 1 of the `pageId`
`languageCode`	`string`	derive this property - section 2 of the `pageId`
`pageName`	`string`	derive this property - sections 3 and further of the `pageId`

The `app` namespace should have the following structure. Update the contents of the namespace as the data becomes available.

property	type	comment
`stepName`	`string`	collected via `viewStep`
`stepNumber`	`number`	collected via `viewStep`
`origin`	`Origin`	collected via `originComplete`
`destination`	`Destination`	collected via `destinationComplete`
`packageCount`	`number`	collected via `packagesComplete`
`rate`	`Rate`	as collected via `viewRate`
`errors`	`Error[]`	collected via `error`

A stub for a `dataLayerReducer` is provided. Use / adapt it if you like, however the only hard requirement is that you use Redux to update the state. How you build your solution in the end is up to you.

Part 2: Analytics tracker specification

A mock analytics tracker is provided. The tracker has three main functions: `trackPage`, `trackEvent`, `trackConversion`.

A quick guide on implementing each of the functions:

function	primary param	extra param
`trackPage`	`pageName (string)`	`dimensions (object)`
`trackEvent`	`eventName (string)`	`dimensions (object)`
`trackConversion`	`revenue (number)`	`dimensions (object)`

Page views

The `trackPage` function is called for each `pageInfo` event.

Primary param

Provide the `page.pageName` property from the data layer.

Extra dimensions

The `dimensions` for this call should be:

dimension	value
`dimensions.dimension01`	`page.pageId`
`dimensions.dimension02`	`page.countryCode`
`dimensions.dimension03`	`page.languageCode`

Events & revenue

The `trackEvent` function is called for each step completion event, *except* for `shipmentComplete`. Additionally call it for the `shipAgain` GDL event.

For the `shipmentComplete` event, implement the `trackConversion` function.

Primary params

For `trackEvent`, provide the the GDL `eventName` as primary parameter.

For `trackConversion`, provide `app.rate.amount` from the data layer as primary parameter.

Extra dimensions

Below is the overview of extra dimensions for both `trackEvent` and `trackConversion`. Send these when they are available. Note that all dimensions should be string values, meaning that objects will need to be stringified according to the format provided.

property	datalayer value	format
`dimensions.dimension01`	`page.pageId`	
`dimensions.dimension02`	`page.countryCode`	
`dimensions.dimension03`	`page.languageCode`	
`dimensions.dimension04`	`app.stepName`	
`dimensions.dimension05`	`app.stepNumber`	
`dimensions.dimension06`	`app.origin`	`<country>_<city>`
`dimensions.dimension07`	`app.destination`	`<country>_<city>`
`dimensions.dimension08`	`app.packageCount`	
`dimensions.dimension09`	`app.rate.currency`	
`dimensions.dimension10`	`app.errors`	`<id>_<message>,<id>_<message>,...`

Note on 'missing' GDL events

Not all GDL events have been used in this analytics implementation. This is intentional.

Part 3: Provide useful debug output

For debugging purposes and as a utility towards analysts, we need to understand what occurs in run-time in our library. To accomplish this, output the event stream to the browser console.

This part of the assignment is more free-form. Think about creating output which is useful and highly readable for the stakeholders mentioned.

Part 4: Write some unit tests

If all went well, by now you'll have written some nice code. Time to show you also know how to write some unit tests.

100% code coverage is not required. But show with your choice of unit tests you have an eye for the important parts of the codebase, and how to guard its functionality.

Overall technical remarks

- Please us Redux as the central 'backbone' of your GDL library.
- We have added a basic Typescript setup, please use it and adjust if needed.
- For unit testing, we recommend Jest and provided a basic setup. If you want to use something else that is fine as well.
- Use additional 3rd party modules at your own discretion.
- For anything you do, a functional solution is required. For bonus points, think about a generic solution which scales with ease. Show us you can help us build something which can easily be applied to 100 additional apps like this.
- Being compatible with older generation browsers is not part of this assignment.

Running the sample application and GDL

A `package.json` is provided with the basic requirements to run the sample app alongside a development version of GDL. The following commands are available out of the box:

command	description
`npm install`	install dependencies
`npm run build`	create build using webpack
`npm run test`	run unit tests (none provided OOTB)
`npm run watch`	create webpack development server at http://localhost:8888

Questions?

It is not our plan to make you guess about our intentions with this assigment. If for any reason something is not completely clear, please reach out to us. We will be happy to help you out.