TL;DR

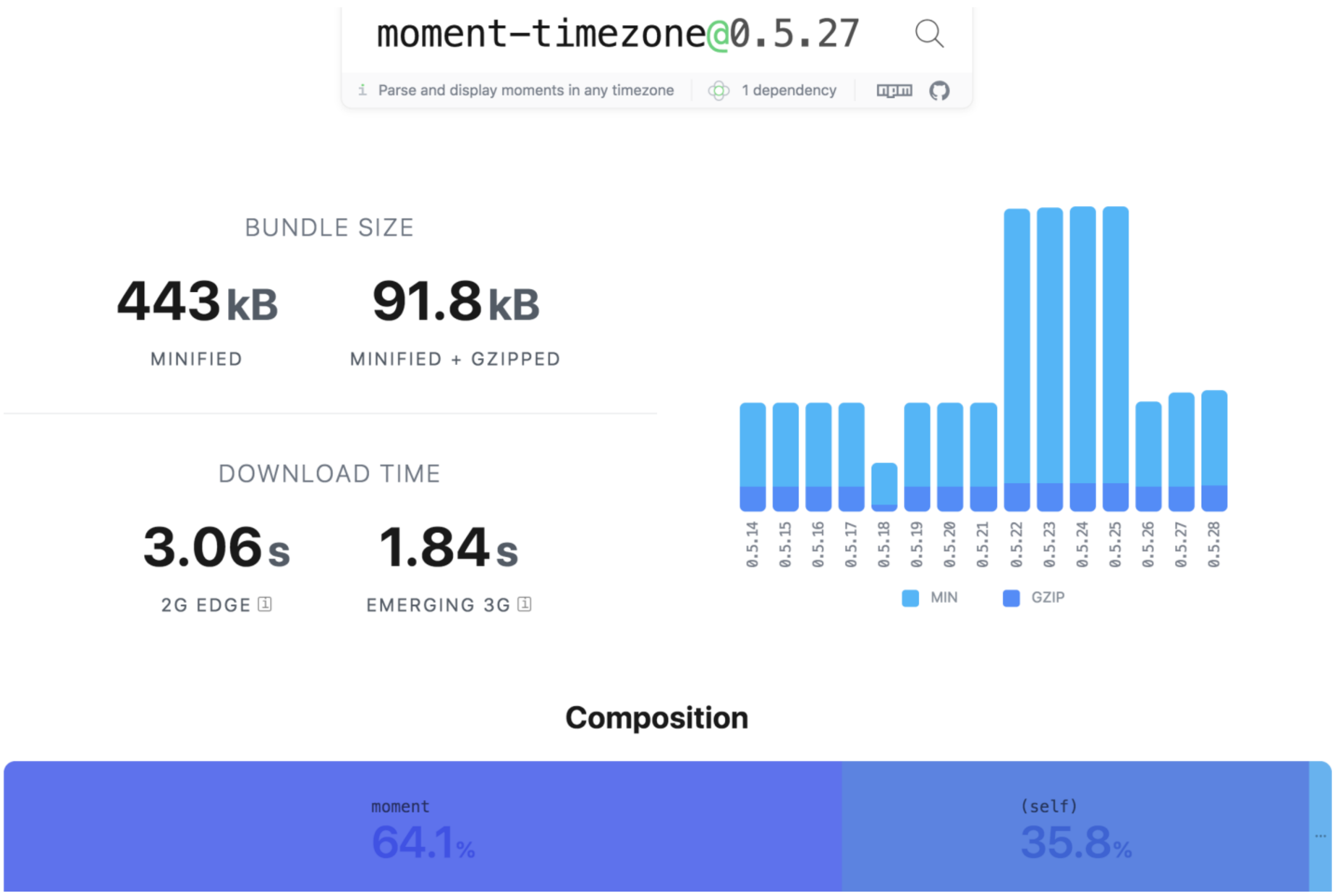
- As the bundle size started to grow in **React convergence app (new ongoing react app)**, we removed 'moment-timezone' dependency, implemented moment-timezone ourself and used only the necessary data from moment-timezone
- Made changes in **DateWidget** and **DateTime** used in @zohodesk/components that computes timezone offsets internally, using the timeZone data that is passed as props (details given below).
- Moment-timezone is not removed from DeskComponent branch and details are given below on how **DeskComponent (live react app)** developers should adapt to the new DateWidget and DateTime Components from @zohodesk/components

Problem we faced in React convergence app:

Last year, while the team is working on the react convergence app, the size of the bundles started to grow as we are converging all the modules. The large initial bundle size started to slow down the initial render. One of the key factors contributing to the large initial bundle size is the 'moment-timezone' library (Total minified + gzipped size of ~91Kb).

Why was the bundle size of 'moment-timezone' large?

'moment-timezone' library includes all the locale and time zone data for all the countries/regions along with source code by default and more than 95% of moment-timezone library size comes from its data.


moment-timezone@0.5.27

Parse and display moments in any timezone | 1 dependency

BUNDLE SIZE

443 kB MINIFIED

91.8 kB MINIFIED + GZIPPED

DOWNLOAD TIME

3.06 s 2G EDGE

1.84 s EMERGING 3G

Composition

moment 64.1%

(self) 35.8%

Let's look at some terminologies before going into the solution:

- Locales:

Country/Region locales data included the month names, week names, date-time formats and relative date-time strings for each country.

Reference:

Source: Locale Moment Data

- Time zones

Due to the earth's own revolution, different regions are assigned different clocks based on the regions days and nights. Thus, regions located in different latitudes will have a different time. IST, Asia/Kolkata, America/Los_Angeles, Europe/Dublin, GMT, UTC, Zulu, Etc/GMT etc. are some examples of different time zones.

- Offsets

Greenwich Mean Time (GMT) or Universal Coordinated Time (UTC) is designated with a +0:00 offset. Other countries are assigned offsets with respect to the GMT based on their latitude. Indian Standard Time(IST) or 'Asia/Kolkata' has an offset of '+5:30'.

- Day light savings

Some countries have different offsets for different seasons. This is due to the variation in their day light times for different seasons and for various political and economic reasons. For example, 'America/Los_Angeles' has a day light time with an offset of '-7:00' from March 8, 2:00 AM to Nov 3, 2:00 AM and a non-daylight offset of '-8:00' with respect to GMT during other time of the year. Moment maintains the offset data for all time zones in its repository.

References:

Source: Time zone moment data

More Info: Moment time zone data format

Implementation we came up with in React app:

We didn't need locale data from moment-timezone in our app, but only needed the timezone data. Also, since the user will be using our application with only one timezone, we created separate JSON chunks for each timezone from moment's time zone data. Before the first render of our application, the timezone data based on user's preference will be downloaded and cached on the client. Using the data, we computed the offset for the dates.

Changes in @zohodesk/components

The new **DateWidget** and **DateTime** components makes use of moment's timezone data (in unpacked format) to compute the timezone offsets for the corresponding date and region. This data is passed using 'timeZone' prop. For React-Convergence app, this data is downloaded before the render and is passed as props using React Context.

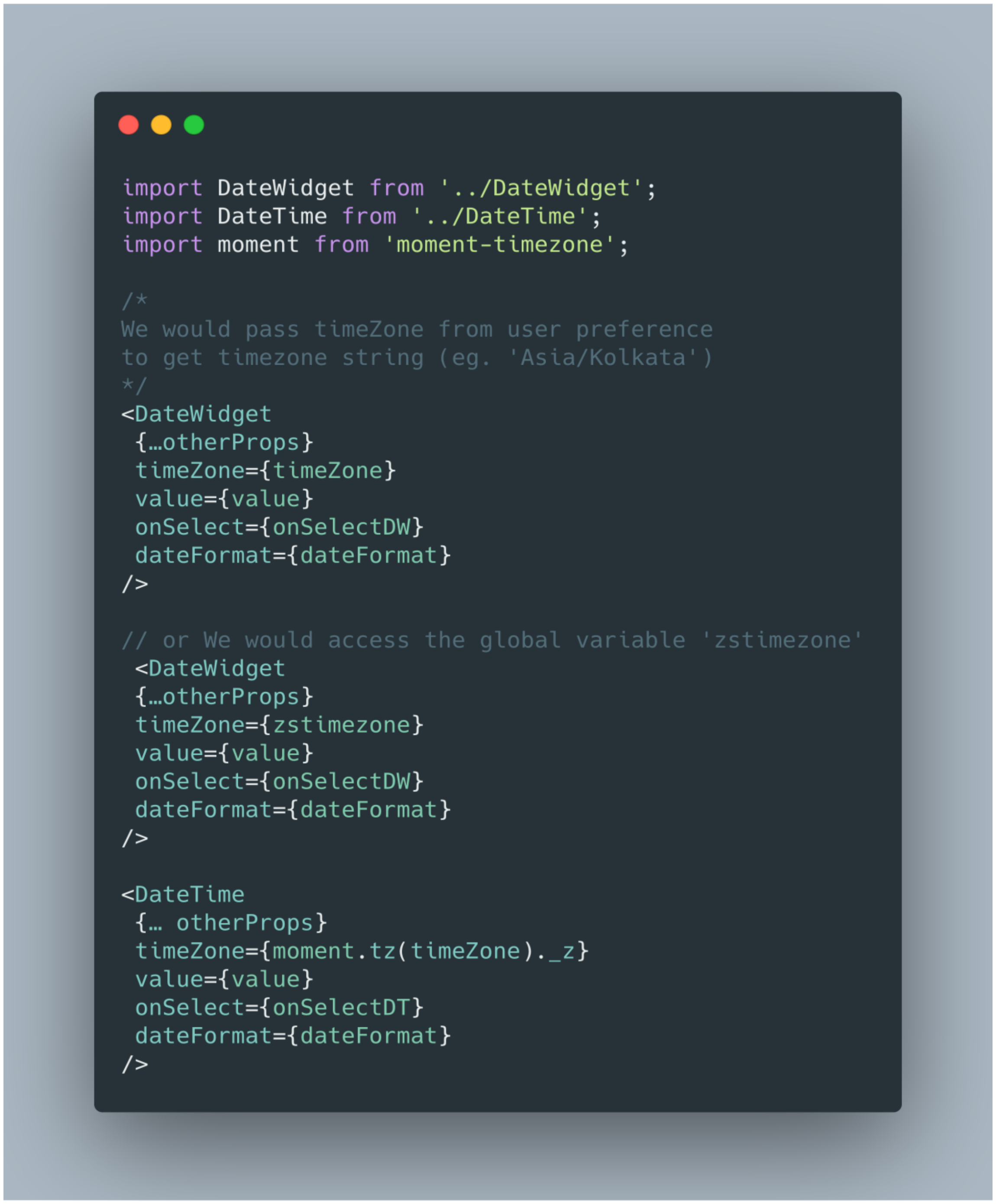
How DeskComponent Developers should adapt these new components?

For DeskComponent, timezone data that is used for computing offset should be passed using moment (since they still use moment-timezone) as given below. Currently, the **DateWidget** used internally in DeskComponent has been replaced with DateWidget from

@zohodesk/component. Kindly make sure to pass timeZone prop as given below for future usage of the DateWidget component.

Note: Please note that you need not convert the date-time (**from onSelect func**) to the corresponding time zones externally, the DateWidget will do it internally using the timeZone prop and pass as an argument to the **onSelect** func.

How it was used before?



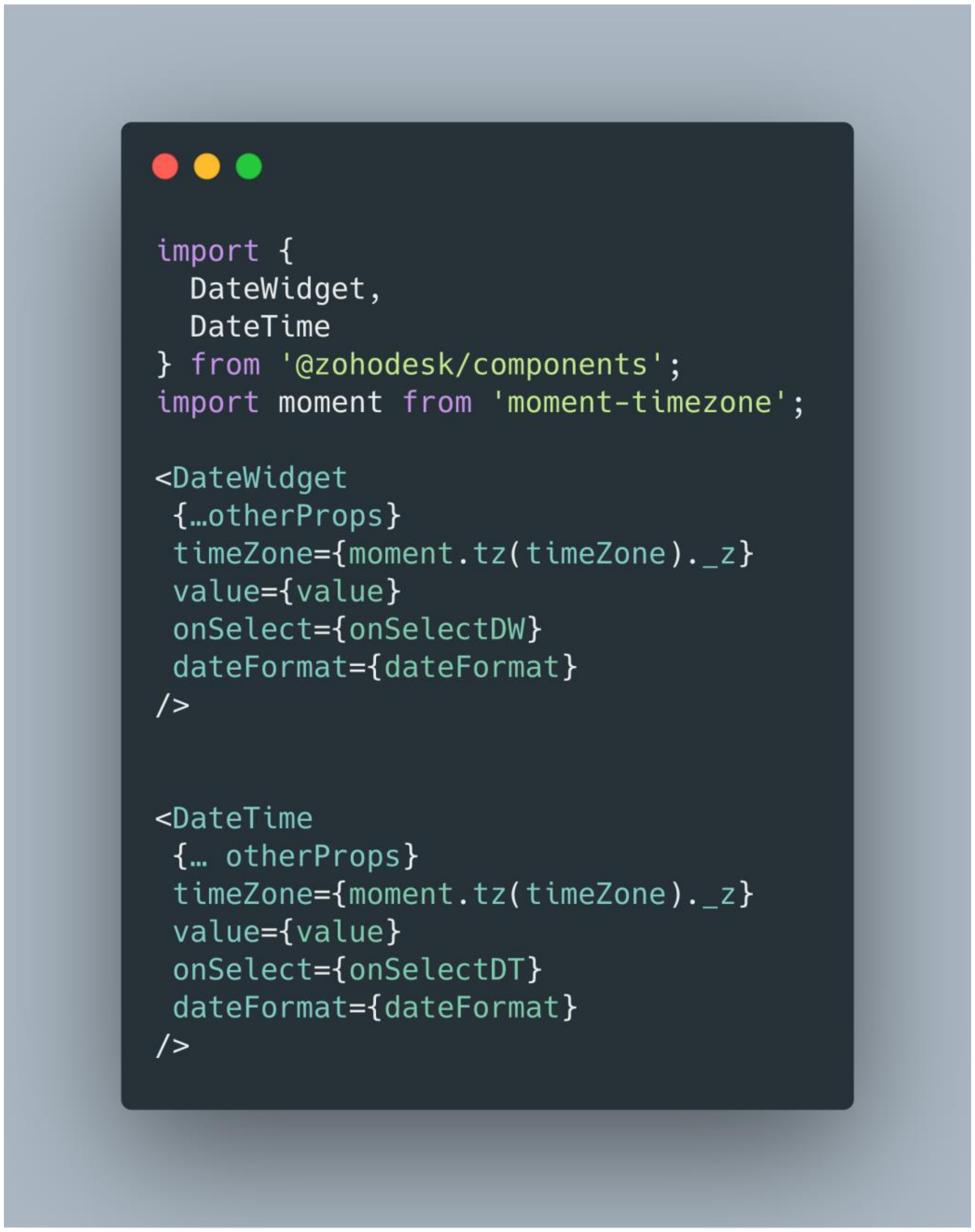
```
import DateWidget from '../DateWidget';
import DateTime from '../DateTime';
import moment from 'moment-timezone';

/*
We would pass timeZone from user preference
to get timezone string (eg. 'Asia/Kolkata')
*/
<DateWidget
 {…otherProps}
 timeZone={timeZone}
 value={value}
 onSelect={onSelectDW}
 dateFormat={dateFormat}
/>

// or We would access the global variable 'zstimezone'
 <DateWidget
 {…otherProps}
 timeZone={zstimezone}
 value={value}
 onSelect={onSelectDW}
 dateFormat={dateFormat}
/>

<DateTime
 {… otherProps}
 timeZone={moment.tz(timeZone)._z}
 value={value}
 onSelect={onSelectDT}
 dateFormat={dateFormat}
/>
```

How to use these components here after?



```
import {
  DateWidget,
  DateTime
} from '@zohodesk/components';
import moment from 'moment-timezone';

<DateWidget
 {…otherProps}
 timeZone={moment.tz(timeZone)._z}
 value={value}
 onSelect={onSelectDW}
 dateFormat={dateFormat}
/>


<DateTime
 {… otherProps}
 timeZone={moment.tz(timeZone)._z}
 value={value}
 onSelect={onSelectDT}
 dateFormat={dateFormat}
/>
```

Prop	Description
timeZone	timezone value (ex. 'Asia/Kolkata') that set in UserPreferences.
value	Should always be in ISO format (ex. **'2020-03-03T08:52:04.546Z'**, +0:00 offset) for DateTime and should be in **'YYYY-MM-DD'** format for dates
onSelectDW(userTime, id)	userTime is the changed **value** prop (format as described above)

onSelectDT(userTime, formattedTime)	userTime is the changed **value** prop and formattedTime is formatted based on 'dateFormat' prop

For Future...

It is strongly recommended not to make use of moment-timezone for any new apps that we build. We are working on moving all the timezone related utils and its data to a separate package. Once completed, developers can make use of that to build their own DateTime Widgets/Components.