Enhancing date, time and time zone support in globalize.js

Manikandan Ramalingam Kandaswamy

Problem

- Time zone support in JavaScript is lacking
- Need to support processing dates in any time zone
  - Example: Shipping, Flight scheduling, etc.

Native JS solution

Intl.DateTimeFormat options.timeZone

Pros:

- Easy to use
- Light i18n solution (no data transfer needed)

Native JS solution

Intl.DateTimeFormat options.timeZone

Cons:

- Supported only in latest browser versions
- Doesn’t support important time zone format like v, V, z
    - (e.g. PT, Pacific Standard Time, Pacific Time, Los Angeles Time, GMT-7)
- Doesn’t support date parsing

User-land solutions

Moment, Moment-timezone

Pros:

- Popular
- Widely used
- Follows IANA

User-land solutions

Moment, Moment-timezone

Cons:

- Doesn’t follow CLDR data schema (difficult to override)
- Heavy (moment + locales + timezones = **106.4k** gzipped)
- Does not support important time zone formats like v, V, z
  (e.g. PT, Pacific Standard Time, Pacific Time, Los Angeles Time, GMT-7)
- Has Daylight savings (isDST) calculation issues

User-land solutions

Google Closure library

Pros:

- Follows CLDR format
- Follows IANA

User-land solutions

Google Closure library

Cons:

- CLDR is hard coded using different schema than Unicode official JSON bindings
- Requires a significant effort to override the data

User-land solutions

Google Closure library

Cons:

- Does not support important time zone formats like v, V (e.g. PT, Pacific Time, Los Angeles Time, GMT-7)
- Parsing doesn’t handle time zone

User-land solutions

Google Closure library

Cons:

- Tightly coupled with Closure compiler. Hard to extract only the pieces we want.
- Heavy (**100 kb+** closure compiled code only)
- Has Daylight savings (isDST) calculation issues

Time zone support needs

- Wide browser and Node.js support
- CLDR format and IANA support + ability to keep up to date
- Ability to provide custom/company overrides
- Solid Daylight Savings algorithm
- Date parsing with arbitrary time zone ID

isDST issues

```
America/Argentina/{Buenos_Aires, Catamarca, *} @1999

Local time = Sun Oct 3 02:59:59 1999

UTC = Sat Oct 2 23:59:59 1999

Offset = -03:00

isdst = 0
```

isDST issues

```
America/Argentina/{Buenos_Aires, Catamarca, *} @1999

Local time = Sun Oct 3 03:00:00 1999

UTC = Sun Oct 3 00:00:00 1999

Offset = -03:00

isdst = 1 (wrong in google closure and moment)
```

```
new Date(938919600000)
```

isDST issues

```
Antarctica/Palmer @2016

Local time = Sun Dec 4 02:59:59 2016

UTC = Sat Dec 3 23:59:59 2016

Offset = -03:00

isdst = 1
```

isDST issues

```
Antarctica/Palmer @2016

Local time = Sun Dec 4 03:00:00 2016

UTC = Sun Dec 4 00:00:00 2016

Offset = -03:00

isdst = 0 (wrong in google closure and moment)
```

isDST issues

```
Asia/Almaty @1991

Local time = Sat Mar 30 19:59:59 1991

UTC = Sun Mar 31 01:59:59 1991

Offset = +06:00

isdst = 0
```

isDST issues

```
Asia/Almaty @1991

Local time = Sat Mar 30 20:00:00 1991

UTC = Sun Mar 31 02:00:00 1991

Offset = +06:00

isdst = 1 (wrong in google closure and moment)
```

Daylight Savings Time Calculations

isDST Solutions

Globalize Simply selects the right isDST from the list of booleans 0, 1 of iana-tz-data given timestamp range.

Moment Deduces isDST by checking timestamp 6 months ahead, and assumes isDST is the one with biggest offset.

Google Deduces isDST by comparing current offset with standard offset.

Globalize.js solution

Wide browser support

- Chrome: (Current - 1) or Current
- Firefox: (Current - 1) or Current
- Safari: 5.1+
- Opera: 12.1x, (Current - 1) or Current
- IE9+

Globalize.js solution

CLDR compliant

- Globalize is UTS#35 compliant
- Supports all CLDR formats
    - Skeletons
    - Formats
        - short, medium, long and full
    - Patterns
        - V, v, z, O, X

Globalize.js solution

Easy to override IANA and CLDR

- Globalize defers data loading completely to user land (CLDR and IANA)
- Since version 1.3 it supports time zone via **zoned-date-time** library

Globalize.js solution

Lightweight library

- All globalize code including zoned-date-time library:
  - **25.9 KB** (Minified + gzipped size)
  - **10.3KB** (Runtime minified + gzipped size)
- One can leverage **globalize-compiler** to slice CLDR and IANA data to the very specific app usage.

Globalize.js solution

- **iana-tz-data**
  - A JSON representation of the time zone transitions & DST information for every time zone id
  - Used to calculate a date for a specific time zone

- **zoned-date-time**
  - A tiny JavaScript Date with IANA timezone support

Globalize

```
let Globalize = require( "globalize" );
```

Loading CLDR data (formats and display names)

```
let CldrData = require( "cldr-data" );

Globalize.load(
  CldrData.entireSupplemental(),
  CldrData.entireMainFor("en", "es", "pt", "de",
"zh", "ar")
);
```

Loading IANA data (to calculate local times)

Globalize.loadIANATimezone(ianaTzData)

This method allows you to load IANA time zone data to enable **options.timeZone** feature on **date formatters and parsers**.

ianaTzData: Get the data from **iana-tz-data**.

Loading IANA data (to calculate local times)

```
let ianaTzData = require( "iana-tz-data" );

Globalize.loadIANATimezone( ianaTzData );
```

Iana-tz-data (JSON representation)

```
{
  "zoneData": {
    ...
    "America": {
      ...
      "New_York": {
        abbrs: [],
        untils: [],
        offsets: [],
        isdsts: []
      ...
   }
```

Globalize formatDate + options.timeZone

```
Globalize("en").formatDate(
new Date("2017-03-19T00:00:00"), {
  datetime: "short",
  timeZone: "America/Los_Angeles"
});

// > '3/18/17, 5:00 PM'
```

Globalize formatDate + options.timeZone

```
Globalize("en").formatDate(
new Date("2017-03-19T00:00:00"), {
  datetime: "short",
  timeZone: "Europe/Berlin"
});

// > '3/19/17, 1:00 AM'
```

Globalize formatDate + options.timeZone

```
Globalize("en").formatDate(
new Date("2017-03-19T00:00:00"), {
  datetime: "full",
  timeZone: "America/Los_Angeles"
});

// > 'Saturday, March 18, 2017 at 5:00:00 PM
Pacific Daylight Time'
```

Globalize formatDate + options.timeZone

```
Globalize("en").formatDate(
new Date("2017-03-19T00:00:00"), {
  datetime: "full",
  timeZone: "America/New_York"
});

// > 'Saturday, March 18, 2017 at 8:00:00 PM
Eastern Daylight Time'
```

Globalize formatDate + options.timeZone

```
Globalize("en").formatDate(
new Date("2017-03-19T00:00:00"), {
  datetime: "full",
  timeZone: "America/Sao_Paulo"
});

// > 'Saturday, March 18, 2017 at 9:00:00 PM
Brasilia Summer Time'
```

Globalize formatDate + options.timeZone

```
Globalize("en").formatDate(
new Date("2017-03-19T00:00:00"), {
  datetime: "full",
  timeZone: "Europe/Berlin"
});

// > 'Sunday, March 19, 2017 at 1:00:00 AM
Central European Standard Time'
```

Globalize formatDate + options.timeZone

```
Globalize("pt").formatDate(
new Date("2017-03-19T00:00:00"), {
  datetime: "full",
  timeZone: "America/Sao_Paulo"
});

// > 'domingo, 19 de março de 2017 19:19:22
Horário Padrão de Brasília'
```

Globalize formatDate + options.timeZone

```
Globalize("de").formatDate(
new Date("2017-03-19T00:00:00"), {
  datetime: "full",
  timeZone: "Europe/Berlin"
});

// > 'Sonntag, 19. März 2017 um 01:00:00
Mitteleuropäische Normalzeit'
```

Globalize formatDate + options.timeZone

```
Globalize("zh").formatDate(
new Date("2017-03-19T00:00:00"), {
  datetime: "full",
  timeZone: "Asia/Shanghai"
});

// > '2017年3月19日星期日 中国标准时间 上午
8:00:00'
```

Globalize formatDate + options.timeZone

```
Globalize("ar").formatDate(
new Date("2017-03-19T00:00:00"), {
  datetime: "full",
  timeZone: "Africa/Cairo"
});

// > 'الأحد، ١٩ مارس، ٢٠١٧ ٢:٠٠:٠٠ ص توقيت شرق
أوروبا الرسم'
```

Globalize formatDate + options.timeZone

```
Globalize("en").formatDate(new Date(), {
  raw: "v",
  timeZone: "America/Los_Angeles"
});

// > 'PT'
```

Globalize formatDate + options.timeZone

```
Globalize("en").formatDate(new Date(), {
  raw: "vvvv",
  timeZone: "America/Los_Angeles"
});

// > 'Pacific Time'
```

Globalize formatDate + options.timeZone

```
Globalize("en").formatDate(new Date(), {
  raw: "VVV",
  timeZone: "America/Los_Angeles"
});

// > 'Los Angeles'
```

Globalize formatDate + options.timeZone

```
Globalize("en").formatDate(new Date(), {
  raw: "VVVV",
  timeZone: "America/Los_Angeles"
});

// > 'Los Angeles Time'
```

Globalize parseDate + options.timeZone

```
Globalize("zh").formatDate(new
Date('2017-10-18T07:45:45.000Z'), {datetime:
"medium", timeZone: "Asia/Shanghai"});

// > '2017年10月18日 下午3:45:45'

Globalize('zh').parseDate('2017年10月18日 下午
3:45:45', {datetime: "medium", timeZone:
"Asia/Shanghai"})

// >> 2017-10-18T07:45:45.000Z
```

Globalize parseDate + options.timeZone

```
> Globalize("zh").formatDate(new
Date('2017-10-18T07:45:45.000Z'), {datetime:
"full", timeZone: "Asia/Shanghai"});

// > '2017年10月18日星期三 中国标准时间 下午3:45:45'

> Globalize('zh').parseDate('2017年10月18日星期三 中
国标准时间 下午3:45:45', {datetime: "full",
timeZone: "Asia/Shanghai"})

// > 2017-10-18T07:45:45.000Z
```

Globalize vs Other i18n libraries

Libraries	CLDR	Standard time zone formats	Daylight Savings Time
moment.js	X	X (doesn't support V, v, z)	✓ (has issues)
Google closure	✓ (custom data schema)	X (doesn't support V, v)	✓ (has issues)
ECMA-402	✓	X (doesn't support V, v, z)	✓
globalize.js	✓	✓	✓

Globalize under the hood (zoned-date-time)

```
import ZonedDateTime from "zoned-date-time";
import {zoneData} from "iana-tz-data";

let date = new Date("2017-03-15T12:00:00Z");
```

Globalize under the hood (zoned-date-time)

```
let losAngelesDate = new ZonedDateTime(
  Date,
  zoneData.America.Los_Angeles
);

// > 2017-03-15T05:00:00.000 PDT (daylight savings)
```

Globalize under the hood (zoned-date-time)

```
let newYorkDate = new ZonedDateTime(
  Date,
  zoneData.America.New_York
);

// > 2017-03-15T08:00:00.000 EDT (daylight savings)
```

Globalize under the hood (zoned-date-time)

```
let saoPauloDate = new ZonedDateTime(
  date,
  zoneData.America.Sao_Paulo
);

// > 2017-03-15T09:00:00.000 -03
```

Globalize under the hood (zoned-date-time)

```
losAngelesDate.isDST(); // > true

saoPauloDate.isDST(); // > false
```

References

- Globalize.js: https://github.com/globalizejs/globalize#date-module
- iana-tz-data: https://github.com/rxaviers/iana-tz-data
- zoned-date-time: https://github.com/rxaviers/zoned-date-time

formatDateToParts

- New in 1.3.0

```
Globalize.locale( "en" );
Globalize.formatDateToParts(new Date(2010, 10, 30));
// > [
// { "type": "month", "value": "11" },
// { "type": "literal", "value": "/" },
// { "type": "day", "value": "30" },
// { "type": "literal", "value": "/" },
// { "type": "year", "value": "2010" }
// ]
```

Relative Time

```
Globalize.locale( "en" );
Globalize.relativeTimeFormatter( "day" )( 1 )
// > "tomorrow"

Globalize.relativeTimeFormatter( "month" )( -1 )
// > "last month"

Globalize.relativeTimeFormatter( "month" )( 3 )
// > "in 3 months"
```

Thank You!

User-land solutions

Moment, Moment-timezone

isDST issue:

```
moment = require('moment')
require('moment-timezone')
m = moment(938919600000).tz('America/Argentina/Buenos_Aires')
m.isDST()
// > false (correct)
m = moment(938919601000).tz('America/Argentina/Buenos_Aires')
m.isDST()
// > false (wrong)
```