JavaScript implementation of Ti.Locale

Introduction

Proposed is an implementation of `Ti.Locale` in JavaScript.

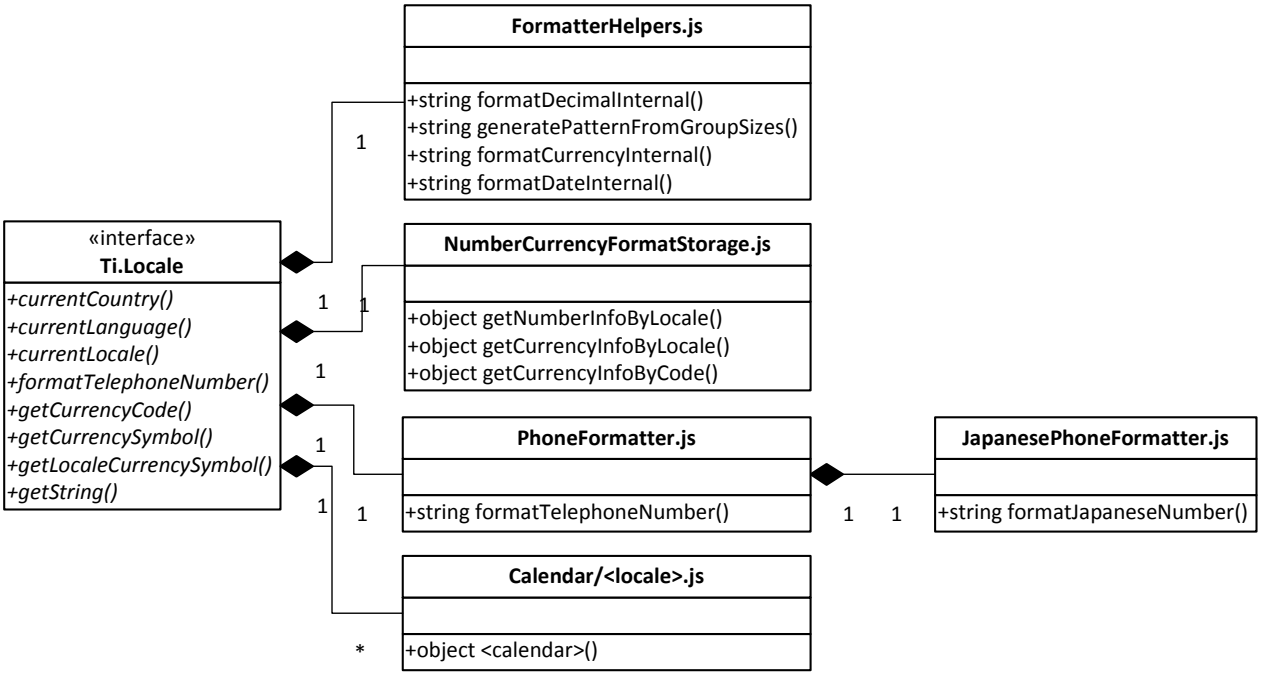
This implementation is platform-independent, and largely browser-independent. It does not rely on JavaScript locale-related capabilities. Instead, it encodes, stores, and uses locale-related information. Data is stored inline in JavaScript files. However, in some places JavaScript locale-related functions are used as a fallback.

Unneeded locales can be easily removed to reduce the size.

High-level design

The proposed implementation resides in the following files:

- **Locale.js** (titanium/Ti/Locale.js). Implements interface methods of `Ti.Locale` and locale-related methods of `String`.
- **FormatterHelpers.js** (titanium/Ti/_/Locale/FormatterHelpers.js). Implements functions for formatting date/time, numbers, and currency.
- **PhoneFormatter.js** (titanium/Ti/_/Locale/PhoneFormatter.js). Contains implementation of, and data for, formatting telephone numbers.
- **JapanesePhoneFormatter.js** (titanium/Ti/_/Locale/JapanesePhoneFormatter.js). Contains implementation of, and data for, formatting Japanese telephone numbers.
- **NumberCurrencyFormatStorage.js** (titanium/Ti/_/Locale/NumberCurrencyFormatStorage.js). Implements an efficient (from the memory standpoint) storage of number and currency locale data. Includes the data for all locales, and the extraction methods.
- **Calendar folder** (titanium/Ti/_/Locale/Calendar). Contains calendar data for different locales. The data is stored in separate JavaScript files (one file per locale). Additionally, the **defaultCalendar.js** file (titanium/Ti/_/Locale/defaultCalendar.js) is used by default if a given locale is unknown.



Detailed design

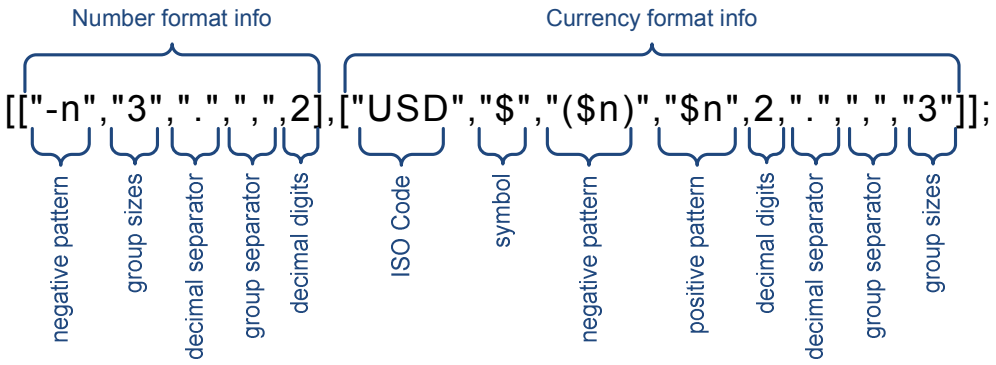
Currency and numbers formatting

Currency data (it includes locale specific number formatting rules, as they are required to format currency) is placed in the NumberCurrencyFormatStorage.js file and provides 3 functions: `getNumberInfoForLocale`, `getCurrencyInfoByLocale` and `getCurrencyInfoByCode`. These functions are called internally from Locale.js. Based on them, the following functions are implemented:

- `String.formatCurrency`
- `String.formatCurrency`
- `Locale.getCurrencyCode`
- `Locale.getCurrencySymbol`
- `Locale.getLocaleCurrencySymbol`

The default locale is en-US. If there is no data for any requested locale, the default one will be returned.

All locale-specific data is stored internally as an array for each locale. For the sake of memory efficiency, locale data is stored in a packed format. When locale infrastructure is used for the first time, it is prepared by filling empty placeholders with default values, and creating a convenient high-level structure used later for parsing or generating text.



For example: for "en-SG" locale's data is stored simply as `[,["SGD"]]`, because all values, except the ISO currency code, are the same as in default locale. For the "es-DO" locale, the data looks like: `[,["DOP","RD$"]]`.

Usually decimal and group separators in the currency format info, group sizes and decimal digits are the same as in the number format info (for each locale). Thus they are specified in the currency format info only if they are different.

Negative and positive pattern – allows us to format numbers in specific locales to place properly currency sign (for currency) and minus sign. In some locales minus is placed in the end of stings, in some locales negative currency values are written in brackets. General pattern format looks like:

> **"$"** – currency sign. Should be replaced with locale-specific currency symbol
>
> **"n"** – modulus of number. Should be replaced with formatted according to locale rules value.
>
> All other symbols unchanged.

Phone numbers formatting

The current implementation for MobileWeb (and Tizen) is based on the Androidi 4.1.1 implementation. So, calling `Ti.Locale.formatTelephoneNumber()` will have the same result on MobileWeb, Tizen and on Android 4.1.1.

The result was checked against `PhoneNumberUtils.formatNumber(unformattedNumber)` on Android.

Date and time formatting

Calendar formatting data is stored in the "Ti_\Locale\Calendar" folder. For each locale, there is one file named as [localeName].js. Example: en-GB.js, de-DE.js, es-MX.js. In case some browsers don't know full locale name, only language – there are short names for all possible values. So if there is no file for "ru-RU" locale ("ru-Ru.js") file with "language only" will be loaded – "ru.js"

Each file is a commonJS module. It returns an object which contains information about day names, month names, date and time part dividers, basic formatting patterns etc. Please refer below to the

```
[NoInterfaceObject] interface dateTimeFormattingData {
      attribute DOMString AM;
      attribute DOMString PM;
      attribute DOMString "/";
      attribute DOMString ":";
      attribute dayNames days;
      attribute monthsNames month;
      attribute monthsNames monthsGenitive;
      attribute formattingPatterns patterns;
  };


[NoInterfaceObject] interface dayNames {
      attribute DOMString[] names;
      attribute DOMString[] namesAbbr;
      attribute DOMString[] namesShort;
}


[NoInterfaceObject] interface monthsNames {
      attribute DOMString[] names;
      attribute DOMString[] namesAbbr;
}

[NoInterfaceObject] interface formattingPatterns {
      attribute DOMString d;
      attribute DOMString D;
      attribute DOMString t;
      attribute DOMString T;
}
```

Date and time formatting rules (format templates) are based on MS .NET styled mask (when “ddd” – is the abbreviated name of the day, “dddd” - full name of the day, etc). Currently available patterns for date: D – long date format, d – short date format. For time: T – long time format, t – short time format. Medium format right now has no valid pattern.

Right now the calendar supports Gregorian calendars.