Building International Applications with Vue.js

A Focus on vue-i18n and vuex-i18n

Dr. med. Ramon Saccilotto, MAS

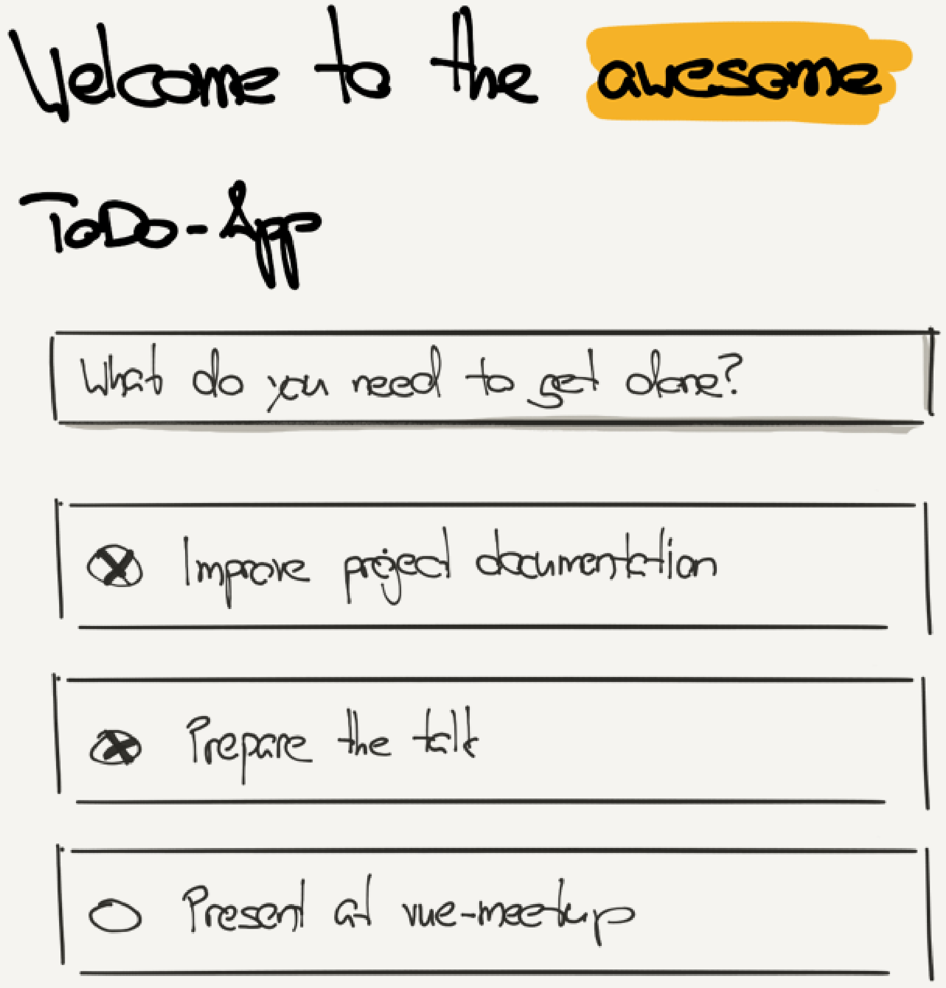
Team Leader Application Development
University of Basel | Department of Clinical Research
c/o University Hospital Basel | Schanzenstrasse 55 | CH-4031 Basel
Phone +41 61 328 51 42 | Email ramon.saccilotto@usb.ch

Internationalization

is the process of designing a software application so that it can be adapted to various languages and regions *without engineering changes*.

Localization

is the process of adapting internationalized software for a specific region or language by translating text and adding locale-specific components.



Welcome to the awesome ToDo-App

What do you need to get done?

- [x] Improve project documentation
- [x] Prepare the talk
- [ ] Present at vue-meetup

i18n

Internationalization / L10n / localization / translation

- vue-i18n - Internationalization plugin for Vue.js.
- vue-translate-plugin - VueJS plugin for translations.
- vuex-i18n - Localization plugin for vue.js 2.0 using vuex as store.
- vue-gettext - Translate your Vue.js applications with gettext.
- vue-i18n - A small plugin for implementing translations in Vue.js.
- vue-multilanguage - Support many languages in Vue.js 2.
- vue-ts-locale - A plugin for implementing translations using Intl in Vue.js 2 with typescript support.
- vue-i18next - A i18next wrapper to support translations in Vue.js 2.
- vue-polyglot - Basic translation plugin for Vue.js 2 with async loading.
- v-localize - Simple localization plugin for the amazing Vue.js.
- vue-simple-i18n - Probably the thinnest library to end all Vue i18n solutions within 1kb
- template-string-i18n - I18n using template strings with auto save/load translate documents.
- vue-translations - Translate your vuejs application easy with it.
- vue-i18n-service - Export and import @kazupon/vue-i18n's SFC translations simply using `npx vue-i18n-service export|import`.
- vue-i18n-filter - Vue filter extend for Vue-i18n, simply using `{{ hello world | t }}`.

Vue i18n

Vue I18n is internationalization plugin of Vue.js. It easily integrates some localization features to your Vue.js application.

Features includes:

- Various formats localization
- Pluralization
- DateTime localization
- Number localization
- Component based localization
- Component interpolation
- Fallback localization
- …

Setup

```
import Vue from 'vue';
import VueI18n from 'vue-i18n';

Vue.use(VueI18n);

const messages = {
  en: {
    message: {
      hello: 'hello world'
    }
  },
  ja: {
    message: {
      hello: 'こんにちは、世界'
    }
  }
}

// Create VueI18n instance with options
const i18n = new VueI18n({
  locale: 'ja', // set locale
  messages, // set locale messages
});

// Create a Vue instance with `i18n` option
new Vue({ i18n }).$mount('#app');
```

HTML Formatting

```
<template>
  <p v-html="$t('message.hello')"></p>
</template>

<script>
  const messages = {
    en: {
      message: {
      hello: 'hello <br> world'
      }
    }
  }
</script>

<html>
  <p v-html="$t('message.hello')"></p>
</html>
```

Named formatting

```
<template>
  <p>{{ $t('message.hello', { msg: 'hello' }) }}</p>
</template>

<script>
  const messages = {
    en: {
      message: {
        hello: '{msg} world'
      }
    }
  }
</script>

<template output>
  <p>hello world</p>
</template>
```

List interpolation

```
<template>
  <p>{{ $t('message.hello', ['hello']) }}</p>
</template>

<script>
  const messages = {
    en: {
      message: {
        hello: '{0} world'
      }
    }
  }
</script>

<template output>
  <p>hello world</p>
</template>
```

Pluralization

```
<template>
  <p>{{ $tc('car', 1) }}</p>
  <p>{{ $tc('car', 2) }}</p>

  <p>{{ $tc('apple', 0) }}</p>
  <p>{{ $tc('apple', 1) }}</p>
  <p>{{ $tc('apple', 10, { count: 10 }) }}</p>
</template>

<script>
  const messages = {
    en: {
      car: 'car | cars',
      apple: 'no apples | one apple | {count} apples'
    }
  }
</script>

<template output>
  <p>car</p>
  <p>cars</p>

  <p>no apples</p>
  <p>one apple</p>
  <p>10 apples</p>
</template>
```

DateTime Localization

```
<template>
  <div id="app">
    <p>{{ $d(new Date(), 'short') }}</p>
    <p>{{ $d(new Date(), 'long', 'ja-JP') }}</p>
  </div>
</template>

<script>
  const i18n = new VueI18n({
    dateTimeFormats
  })

  new Vue({
    i18n
  }).$mount('#app')
</script>

<template output>
  <p>Apr 19, 2017</p>
  <p>2017年4月19日(水) 午前2:19</p>
</template>
```

DateTime Format Specification

```
const dateTimeFormats = {
  'en-US': {
    short: {
      year: 'numeric', month: 'short', day: 'numeric'
    },
    long: {
      year: 'numeric', month: 'short', day: 'numeric',
      weekday: 'short', hour: 'numeric', minute: 'numeric'
    }
  },
  'ja-JP': {
    short: {
      year: 'numeric', month: 'short', day: 'numeric'
    },
    long: {
          year: 'numeric', month: 'short', day: 'numeric',
          weekday: 'short', hour: 'numeric', minute: 'numeric',
hour12: true
    }
  }
}
```

Number localization

```
<template>
  <div id="app">
    <p>{{ $n(100, 'currency') }}</p>
    <p>{{ $n(100, 'currency', 'ja-JP') }}</p>
  </div>
</template>

<script>
  const i18n = new VueI18n({
    numberFormats
  })

  new Vue({
    i18n
  }).$mount('#app')
</script>

<template output>
  <div id="app">
    <p>$100.00</p>
    <p>￥100</p>
  </div>
</template>
```

Structure of the locale definitions

```
{
  "en": {  // 'en' Locale
    "key1": "this is message1", // basic
    "nested": { // nested
      "message1": "this is nested message1"
    },
    "errors": [ // array
      "this is 0 error code message",
      {  // object in array
        "internal1": "this is internal 1 error message"
      },
      [  // array in array
        "this is nested array error 1"
      ]
    ]
  },
  "ja": { // 'ja' Locale
    // ...
  }
}
```

Usage in templates

```
<template>
  <div id="app">
    <!-- basic -->
    <p>{{ $t('key1') }}</p>
    <!-- nested -->
    <p>{{ $t('nested.message1') }}</p>
    <!-- array -->
    <p>{{ $t('errors[0]') }}</p>
    <!-- object in array -->
    <p>{{ $t('errors[1].internal1') }}</p>
    <!-- array in array -->
    <p>{{ $t('errors[2][0]') }}</p>
  </div>
</template>
```

Fallback localization

```
const i18n = new VueI18n({
  locale: 'ja',
  fallbackLocale: 'en',
  messages
})
```

Component based locale info

```
// setup locale info for root Vue instance
const i18n = new VueI18n({
  locale: 'ja',
  messages: { ... }
})

// Define component
const Component1 = {
  template: `...`,
  i18n: { // `i18n` option, setup locale info for component
    messages: { ... }
  }
}

new Vue({
  i18n,
  components: {
    Component1
  }
}).$mount('#app')
```

Component interpolation

```
<template output>
  <p>I accept xxx <a href="/term">Terms of Service Agreement</a></p>
</template>

<template>
  <p>{{ $t('term1') }}<a href="/term">{{ $t('term2') }}</a></p>
</template>

<script>
  const messages = {
    en: {
      term1: 'I Accept xxx\'s',
      term2: 'Terms of Service Agreement'
    }
  }
</script>
```

Component interpolation

The children of i18n functional component are interpolated by their orders of appearance.

```
<template>
  <i18n path="term" tag="label" for="tos">
    <a :href="url" target="_blank">{{ $t('tos') }}</a>
  </i18n>
</template>

<script>
  const messages = {
    en: {
      tos: 'Term of Service',
      term: 'I accept xxx {0}.'
    }
  };

  const i18n = new VueI18n({ locale: 'en', messages});

  new Vue({
    i18n,
    data: {
      url: '/term'
    }
  }).$mount('#app');
</script>
```

Component interpolation

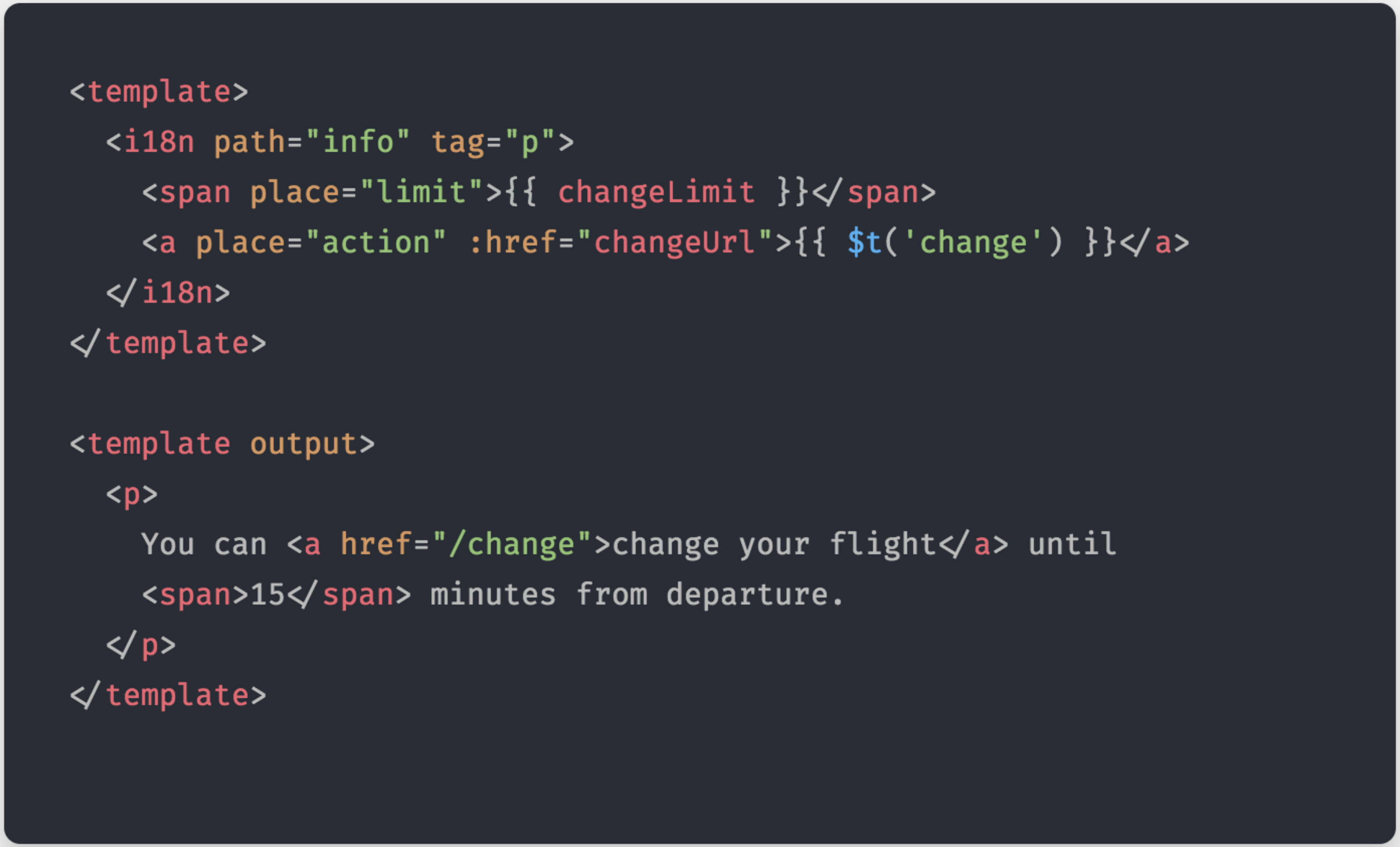
Advanced usage

```
<template>
  <i18n path="info" tag="p">
    <span place="limit">{{ changeLimit }}</span>
    <a place="action" :href="changeUrl">{{ $t('change') }}</a>
  </i18n>
</template>

<script>
  const messages = {
    en: {
      info: 'You can {action} until {limit} minutes from departure.'
,
      change: 'change your flight',
      refund: 'refund the ticket'
    }
  };

  const i18n = new VueI18n({ locale: 'en', messages});

  new Vue({
    i18n,
    data: {
      changeUrl: '/change',
      refundUrl: '/refund',
      changeLimit: 15,
      refundLimit: 30
    }
  }).$mount('#app')
</script>
```



```html
<template>
  <i18n path="info" tag="p">
    <span place="limit">{{ changeLimit }}</span>
    <a place="action" :href="changeUrl">{{ $t('change') }}</a>
  </i18n>
</template>

<template output>
  <p>
    You can <a href="/change">change your flight</a> until
    <span>15</span> minutes from departure.
  </p>
</template>
```

Changing the current locale

```
const i18n = new VueI18n({
  locale: 'ja', // set locale
  ...
});

// create root Vue instance
new Vue({
  i18n,
  ...
}).$mount('#app');

// change other locale
i18n.locale = 'en';
```

Changing the current locale

```
<template>
  <div class="locale-changer">

    <select v-model="$i18n.locale">

      <option v-for="(lang, i) in langs"
        :key="`Lang${i}`"
        :value="lang">
          {{ lang }}
      </option>

    </select>

  </div>
</template>

<script>
  export default {
    name: 'locale-changer',
    data () {
      return { langs: ['ja', 'en'] }
    }
  }
</script>
```

Lazy loading translations

```javascript
// load locale from server
lang = 'de-ch';
messages = { … };

// add messages to the translations
i18n.setLocaleMessage(lang, messages);

// set the current locale
i18n.locale = lang;

// optional set language attribute in html
document.querySelector('html').setAttribute('lang', lang);
```

Single File Components

Extraction and injection of translations through the use of a webpack loader.

(vue-i18n-loader)

```
<i18n>
  {
    "en": {
      "hello": "hello world!"
    }
  }
</i18n>

<template>
  <div id="app">
    <p>message: {{ $t('hello') }}</p>
  </div>
</template>

<script>
  export default {
    name: 'app',
    data () { return { locale: 'en' } },
    watch: {
      locale (val) {
        this.$i18n.locale = val
      }
    }
  }
</script>
```

Webpack loader configuration

```js
module.exports = {
  // ...
  module: {
    rules: [
      {
        test: /\.vue$/,
        loader: 'vue-loader',
      },
      {
        resourceQuery: /blockType=i18n/,
        type: 'javascript/auto',
        loader: '@kazupon/vue-i18n-loader'
      }
      // ...
    ]
  },
  // ...
}
```

Vuex-i18n

We are big fans of the awesome vue, vuex and vue-router libraries and were just looking for an easy to use internationalization plugin, employing as much of the "standard library" as possible.

The main difference to other internationalization plugins is the ease of use and support for locales directly with the application or later from the server.

Disclaimer: I am one of the authors of this library

Setup of the plugin

Translations will be stored inside the vuex store as a separate i18n module.

```
import Vue from 'vue';
Vue.use(Vuex);

// initilialize a new vuex store
const store = new Vuex.Store();

// initialize the vuex-i18 module
import vuexI18n from 'vuex-i18n';
Vue.use(vuexI18n.plugin, store);

// add some start translations
import en from './i18n/en.js';
Vue.i18n.add('en', en);

// default locale is english
Vue.i18n.set('en');

// initialize a new vue instance
new Vue({
    store,
    el: '#app',
    render: h => h(App)
});
```

Formatting of translations

Translation lookup key checks for several scenarios:

- Check current locale
- Check associated locales

```
<template>
  <div>
    {{ $t('Some localized information')}}
  </div>
</template>

<template output>
  <div>
    Some localized information
  </div>
</template>
```

Lookup of key follows specific path:

- Check current locale > de-CH
- Check associated locale > de

- Call async translation resolver to fetch locale information

- Check fallback locale > en
- Use the key itself as message

Parameter interpolation

```
<template>
  <div>
    {{ $t('You have {count} new messages', {count: 5}) }}
  </div>
</template>

<template output>
  <div>
    "You have 5 new messages"
  </div>
</template>
```

Pluralization

Pluralized messages can be denoted by using :::

Up to six pluralization forms are supported, based on the configuration taken from vue-gettext.

```
<template>
  <div>
    {{ $t('You have {count} new message ::: You have {count} new messages',
          {count: 5}, 5) }}
  </div>
</template>

<template output>
  <div>
    You have 5 new messages
  </div>

  <div>
    You have 1 new message
  </div>

  <div>
    You have 0 new messages
  </div>
</template>
```

Translation in specific language

$tlang allows to specify the language to be used for translation

```
<template>
  <div>
    {{ $tlang('en', 'You have {count} new messages', {count: 5}) }}
  </div>
</template>

<template output>
  <div>
    You have 5 new messages
  </div>
</template>
```

Lazy loading of translations

// Add locale translation to the storage. This will extend existing information (i.e. 'de', {'message': 'Eine Nachricht'})
$i18n.add(locale, translations)

// Replace locale translations in the storage. This will remove all previous locale information for the specified locale
$i18n.replace(locale, translations)

// Remove the given locale from the store
$i18n.remove(locale)

What about date and number localization

For date localization there is the great momentjs library. With 39192 stars on github.

For number localization there is numeraljs. Also a great library with 6831 stars on github.

Therefore we did not build these features into the vuex-i18n plugin. But it can easily be added it if there is a need for this.

Roadmap for vuex-i18n

Create an awesome documentation with vuepress. Because good documentation is one of the great features of vue and its core libraries.

```
Status: Documentation is setup and first version is on
github. Will be extended in the next days.
```

Roadmap for vuex-i18n

Create a todo app example to improve on-boarding experience by providing an extensive real life example.

```
Status: Initial version is setup and published on
github. Comparison version with vue-i18n is in the
works.
```

Roadmap for vuex-i18n

Implement proper automated testing for the complete library

```
Status: Tests are currently done manually. There is an
initial testing setup with jest by a contributor that
we will explore.
```

Roadmap for vuex-i18n

Implement a plugin for vue-router that allows matching of routes with localized strings

`Status: Planning`

`We should be able to specify` **`/about`** `and` **`/ueber-uns`** `as something akin to` **`/:locale['about']`**

Roadmap for vuex-i18n

Create a webpack loader or plugin to automatically extract all translation strings from the code and templates and generate the respective translation files.

```
Status: Planning

Max is going to tell you about the awesome utility
that he wrote that is doing something similar.

We would like to have a loader that parses the
javascript into an abstract syntax tree and allows us
to really find all occurences and maybe manipulate
some code.
```

Roadmap for vuex-i18n

Support for interpolation of components. Allowing translators to use components as part of the translation strings.

```
Status: Brainstorming, we have some great ideas.
```

Thank you for your attention.

Any feedback or ideas to improve the library are very welcome.