Vue国际化

紧接之前的，

**简单使用**

// html 需要使用 {{}} 将 name包装起来

{{$t('name')}}

// js

$t('name')

#### 可传入变量

// 命名传入参数

// messages:

{

cn: {

name: '名字：{name}'

},

us: {

name: 'Name: {name}'

}

}

$t('name', {name: 'Jack'}) // Name：Jack

// 列表传入参数

// messages:

{

cn: {

name: '名字：{0}{1}'

},

us: {

name: 'Name: {0}{1}'

}

}

// array

$t('name', ['Jack', 'Job'])

// object

$t('name', {'0':'Jack', '1': 'Job'}}

#### 复数格式

使用分隔符 |

// messages:

{

us: {

car: 'car | cars | {count} cars'

}

}

$tc('car', x) // 选用不同的car类型

#### 时间格式

const dateTimeFormats = {

'us': {

short: {

year: 'numeric', month: 'short', day: 'numeric'

},

long: {

year: 'numeric', month: 'short', day: 'numeric',

weekday: 'short', hour: 'numeric', minute: 'numeric'

}

},

cn: {

xxx

}

}

// 需要放入配置项中

const i18n = new VueI18n({

locale: '',

messages,

dateTimeFormats

})

// 使用

$d(new Date(), 'short')

$d(new Date(), 'long')

$d(new Date(), 'short', 'cn')

#### 金额符号

const numberFormats = {

'en-US': {

currency: {

style: 'currency', currency: 'USD'

}

},

'ja-JP': {

currency: {

style: 'currency', currency: 'JPY', currencyDisplay: 'symbol'

}

}

}

// 同样也要加入配置项中

const i18n = new VueI18n({

numberFormats

})

// 使用

$n(100, 'currency') // $100.00

$n(100, 'currency', 'ja-JP') // ￥100

#### 提供一个默认语言设置fallbackLocale

当某个语言不存在时，提供一个默认全的语言去处理

const messages = {

cn: {

name: 'Name:'

}

us: {

}

}

// us 的语言包中不存在 name , 我们默认cn，当us不存在时，使用cn的值

const i18n = new VueI18n({

locale: 'us',

fallbackLocale: 'cn',

messages

})

#### v-t 可以用于变量的引用，类似于$t

v-t 指令

// 官网的列子

new Vue({

i18n: new VueI18n({

locale: 'en',

messages: {

en: { hello: 'hi {name}!' },

ja: { hello: 'こんにちは、{name}！' }

}

}),

computed: {

**nickName** () { return 'kazupon' }

},

data: { path: 'hello' }

}).$mount('#object-syntax')

<div id="object-syntax">

<!-- literal -->

<p v-t="{ path: 'hello', locale: 'ja', args: { name: 'kazupon' } }"></p>

<!-- data biniding via data -->

<p v-t="{ path: path, args: { name: nickName } }"></p>

</div>

<div id="object-syntax">

<p>こんにちは、kazupon！</p>

<p>hi kazupon!</p>

</div>

$t 与 v-t 对比

* $t 是方法调用，v-t 是一个指令
* v-t 性能比$t更好，有自定义指令的缓存
* $t 使用更灵活，使用方法更简单

还有一些其他的用法，具体的请参考[官方文档](https://link.juejin.im/?target=https%3A%2F%2Fkazupon.github.io%2Fvue-i18n)。。

切换语言的话，可以使用内置变量：

// 通过设置 locale 来切换语言

this.$i18n.locale = cn | us

#### 插入组件

如果遇到这样的场景，如何去处理？

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

第一反应是分成**两个字段**，a标签不属于翻译的内容，只要写成：

<p>{{ $t('xx1') }}<a href="/term">{{ $t('xx2') }}</a></p>

看了官网的介绍，说这种处理太笨拙了，可以通过**组件**的方式去处理

// 这里使用了i81n 组件

<i18n path="term" tag="label" **for**="tos">

<a :href="url" target="\_blank">{{ $t('tos') }}</a>

</i18n>

const messages = {

en: {

tos: 'Term of Service',

term: 'I accept xxx {0}.'

}

}

new Vue({

el: '#app',

i18n,

data: {

url: '/term'

}

})

可以看到，仍然使用了两个变量存储信息，但是通过tag来生产标签，path来制定标签的内容。

更高级的用法，可以控制html元素的**插入位置**，通过place来指定出现在html中的位置。

<i18n path="info" tag="p">

<span place="limit">{{ changeLimit }}</span>

<a place="action" :href="changeUrl">{{ $t('change') }}</a>

</i18n>

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')

// result

<p>

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

</p>

保留了语句的一体性，但是如果只是针对名词进行多语言的翻译，不对语法进行要求的话，可能使用不到。

### 动态加载语言包

一次加载所有的语言包是没有必要的，特别是语言包过多的情况下

//i18n-setup.js

import Vue from 'vue'

import VueI18n from 'vue-i18n'

import messages from '@/lang' // 语言包的地址，随项目本身设置修改

import axios from 'axios' // 根据项目中使用api请求模块去设置，不一定是axios

Vue.use(VueI18n)

export const i18n = new VueI18n({

locale: 'en', // set locale

fallbackLocale: 'en', // 默认语言设置，当其他语言没有的情况下，使用en作为默认语言

messages // set locale messages

})

const loadedLanguages = ['en'] // our default language that is prelaoded

**function** setI18nLanguage (lang) {

i18n.locale = lang

axios.defaults.headers.common['Accept-Language'] = lang // 设置请求头部

document.querySelector('html').setAttribute('lang', lang) // 根元素增加lang属性

return lang

}

export **function** loadLanguageAsync (lang) {

**if** (i18n.locale !== lang) {

**if** (!loadedLanguages.includes(lang)) {

return import(/\* webpackChunkName: "lang-[request]" \*/ `@/lang/${lang}`).then(msgs => {

i18n.setLocaleMessage(lang, msgs.default)

loadedLanguages.push(lang)

return setI18nLanguage(lang)

})

}

return Promise.resolve(setI18nLanguage(lang))

}

return Promise.resolve(lang)

}

// 在vue-router的beforeEach的全局钩子里处理

router.beforeEach((to, from, next) => {

const lang = to.params.lang

loadLanguageAsync(lang).then(() => next())

})

# vue中引入vue-i18n实现国际化时与ElementUI库的兼容性问题解决方法：

<https://zhang122622623.github.io/2018/01/29/vue%E4%B8%AD%E5%BC%95%E5%85%A5vue-i18n%E5%AE%9E%E7%8E%B0%E5%9B%BD%E9%99%85%E5%8C%96%E6%97%B6%E4%B8%8EElementUI%E5%BA%93%E7%9A%84%E5%85%BC%E5%AE%B9%E6%80%A7%E9%97%AE%E9%A2%98%E8%A7%A3%E5%86%B3%E6%96%B9%E6%B3%95/>