文章首页开发

heima-leadnews-app

前端工程

VERSION - 1.0.0

|  |  |  |
| --- | --- | --- |
| 文档版本说明 | | |
| 文件状态  [ ] 草稿  [ √ ] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本 |  |
| 作者 | miukoo |
| 完成日期 | 2019-06-19 |

目录

[目录 3](#_Toc18511829)

[1. 需求分析 5](#_Toc18511830)

[1.1 首页需求 5](#_Toc18511831)

[1.2 列表需求 6](#_Toc18511832)

[2. 定义 7](#_Toc18511833)

[2.1 组件定义 7](#_Toc18511835)

[2.2 页面定义 7](#_Toc18511836)

[2.3 路由定义 8](#_Toc18511837)

[2.4 结构定义 8](#_Toc18511838)

[3. 开发 9](#_Toc18511839)

[3.1 路由开发 9](#_Toc18511841)

[3.1.1 二级路由 9](#_Toc18511842)

[3.1.2 一级路由 13](#_Toc18511843)

[3.2 底部菜单 14](#_Toc18511844)

[3.2.1 文件创建 14](#_Toc18511845)

[3.2.2 实现菜单VIEW 14](#_Toc18511846)

[3.2.3 实现菜单Model 20](#_Toc18511847)

[3.2.4 效果演示 21](#_Toc18511848)

[3.3 搜索组件 21](#_Toc18511849)

[3.3.1 文件创建 21](#_Toc18511850)

[3.3.2 实现VIEW 22](#_Toc18511851)

[3.4 无图列表项 25](#_Toc18511852)

[3.4.1 文件创建 25](#_Toc18511853)

[3.4.2 通用样式 25](#_Toc18511854)

[3.4.3 Model定义 30](#_Toc18511855)

[3.4.4 时间格式化 30](#_Toc18511856)

[3.4.5 实现VIEW 34](#_Toc18511857)

[3.5 单图列表项 36](#_Toc18511858)

[3.5.1 文件创建 36](#_Toc18511859)

[3.5.2 实现VIEW 36](#_Toc18511860)

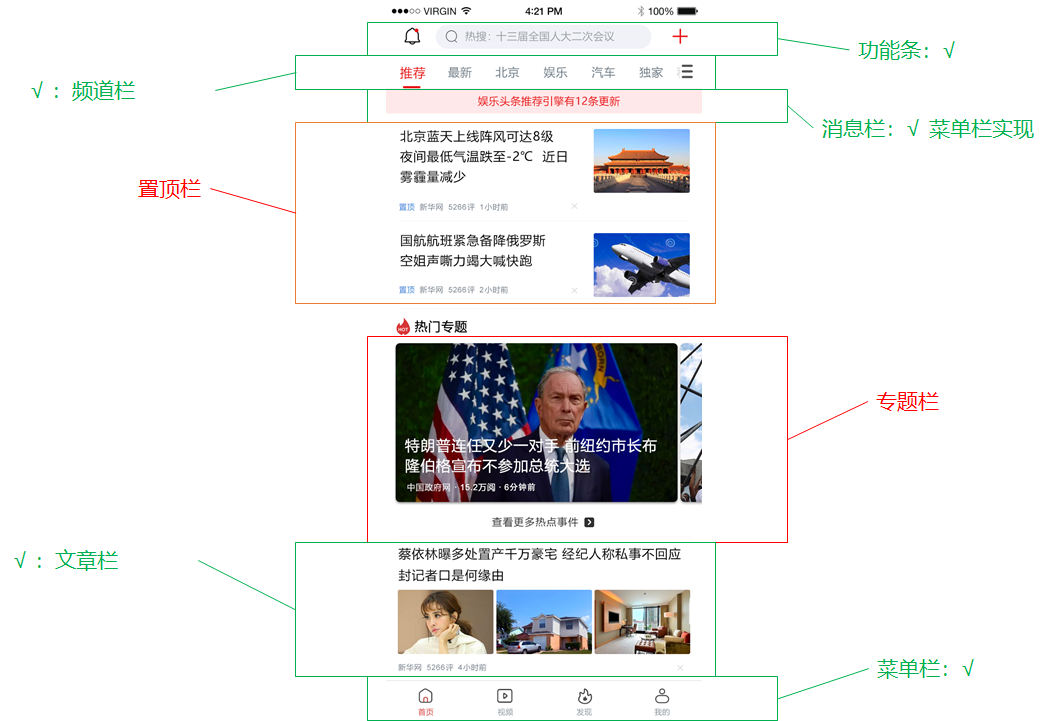
[3.6 三图列表项 39](#_Toc18511861)

[3.6.1 文件创建 39](#_Toc18511862)

[3.6.2 实现VIEW 39](#_Toc18511863)

# 需求分析

## 首页需求



首页页面常用布局主要分为以下几个模块：

* 功能条：放置主要的功能便捷入口；黑马头条主要放置频道编辑和搜索两大功能
* 频道栏：放置平台推荐或者个人喜好的频道
* 消息栏：放置用户阅读时间里面，或发布的消息提示；黑马头条将使用气泡方式替代
* 置顶栏：放置平台核心价值关推荐的精华文章；黑马头条不实现
* 专题栏：放置精选文章或热门专题入口；黑马头条不实现
* 文章栏：放置当前频道下的文章信息
* 菜单栏：放置平台内容模块的导航菜单；黑马头条放置主页、问题、视频、我的4个菜单

注：黑马头条课程不实现置顶栏、专题栏，消息栏在菜单栏中用气泡数字替代。

## 列表需求



文章列表中的文章展现布局常见的有三种：

* 无图：适合没有图片的文章；显示文章的标题、作者、评论、发布时间
* 单图：适合只有一张图片的文章；显示文章的标题、作者、评论、发布时间、一张图片
* 多图：适合有多张图片的文章；显示文章的标题、作者、评论、发布时间、三张图片

# 定义





## 组件定义

按照代码重用性的规划，此部分VIEW可抽取以下几个公用组件：

* 首页搜索组件（home\_bar）：实现功能条中的点击菜单编辑频道、点击搜索框跳转搜索页面两大功能
* 导航菜单组件（menu）：实现首页、问题、视频、我的是个板块的快速进入链接功能
* 无图列表项组件（article\_0）：实现没有图片的文章列表展现布局和相关功能
* 单图列表项组件（article\_1）：实现一张图片的文章列表展现布局和相关功能
* 多图列表项组件（article\_3）：实现三张图片的文章列表展现布局和相关功能

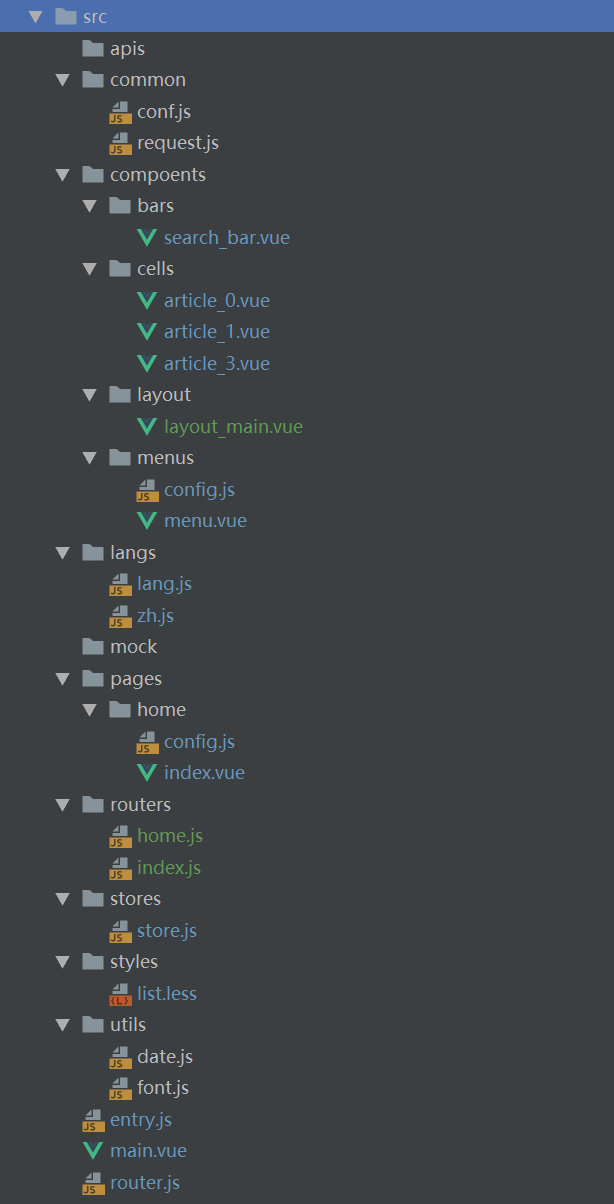
## 页面定义

* 一级路由（main）：引入顶层vue-router路由视图（router-view：路由顶级页面），初始化字体
* 二级路由（layout\_main）：引入通用菜单，一级菜单路由视图（router-view：路由菜单入口页面）
* 主页入口（index）：导入搜索、菜单、文章列表等组件，并进行布局设置

## 路由定义

* [/]：指向layout\_main页面
* [/home/index]：指向‘主页’入口页面

## 结构定义



# 开发



## 路由开发

### 二级路由

#### 文件创建

* 在src/compoents/layouts文件夹下面新建layout\_main.vue二级路由布局文件
* 在src/compoents/menus文件夹下面新建menu.vue导航菜单文件
* 在src/pages/home文件夹下面新建index.vue‘主页’菜单入口布局文件
* 在src/routers文件夹下面新建home.js ，用于存放’主页相关配置’
* 修改src/index.vue文件为main.vue一级路由布局文件

注意：需要修改entry.js文件中的以下代码：

1. //const App = require('@/index.vue');
2. const App = require('@/main.vue');

#### 实现路由VIEW

* 修改layout\_main.vue文件内容为：

<template>  
 <div class="wrapper">  
 <!-- 二级路由视图 -->  
 <div class="router-body"><router-view/></div>  
 <!-- 底部菜单 -->  
 <div class="main-body"><Menu/></div>  
 </div>  
</template>  
<script>  
 // 引入底部菜单  
 import Menu from "@/compoents/menus/menu";  
 export default {  
 name: 'HeiMa-Layout-Mian',  
 components: {Menu}  
 }  
</script>  
<style scoped>  
 .wrapper{  
 flex-direction : column;  
 flex-wrap:wrap;  
 }  
 .router-body{  
 flex: 1;  
 flex-direction : column;  
 border: 1px solid red;  
 }  
 .main-body{  
 position: fixed;  
 left: 0;  
 bottom: 0;  
 }  
</style>

* 修改menu.vue文件内容为：

<template>  
 <div>我是黑马菜单</div>  
</template>  
<script>  
 export default {  
 name: "menu"  
 }  
</script>

#### 实现路由Model

修改home.js文件内容为：

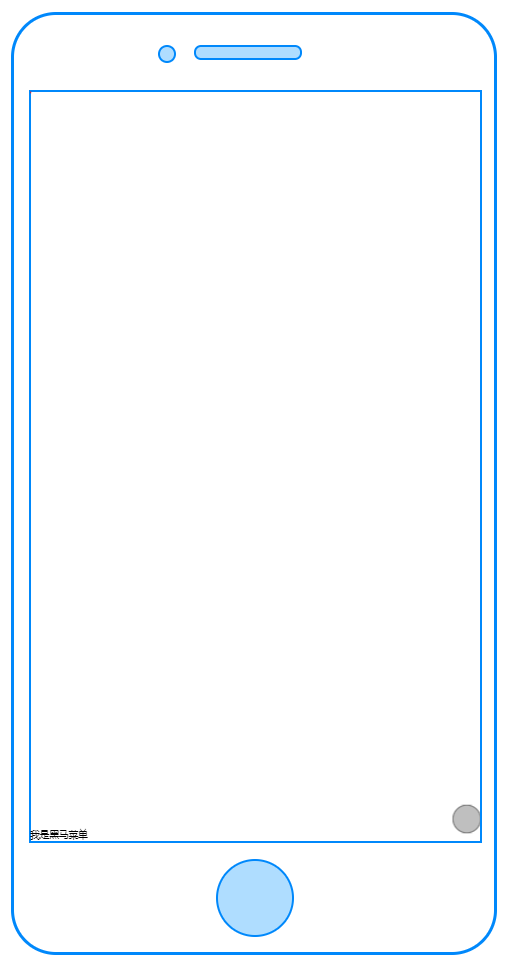
// ============ 主页路由MODEL ==================  
import Layout from '@/compoents/layouts/layout\_main'  
  
let routes = [  
 {  
 path: '/',  
 name: 'LayoutMain',  
 component: Layout  
 }  
]  
  
export default routes;

#### 实现路由VM

修改routers/index.js文件内容，绑定‘主页’相关页面的路由。

import Home from './home'  
let routes = []  
let concat = (router) => {  
 routes = routes.concat(router)  
}  
// 合并‘主页’相关路由  
routes = routes.concat(Home)  
export default routes;

#### 效果演示



### 一级路由

一级路由（src/router.js）主要功能就是加载routers/index.js文件，相关代码为：

import Router from 'vue-router'  
import routes from '@/routers/index'  
Vue.use(Router)  
export const router = new Router({  
 routes:routes  
});

## 底部菜单

为提高开发效率，项目可使用weex-ui快速开发，选用的菜单样式参考：<https://alibaba.github.io/weex-ui/#/cn/packages/wxc-tab-bar/>

**注：菜单功能暂不实现。**

### 文件创建

* compoents/menus/文件夹下创建config.js文件，用于存放菜单内容配置

### 实现菜单VIEW

你会发现直接复制weex-ui的示例代码，在项目中是不可用的，不显示或者样式错乱。这是因为未设置组件的高和宽。

在这里还需要指出这个组件每个菜单都有一个内容区，而作为首页或公用菜单，我们不想菜单加载或包含太多的功能，在这个项目里我们只需要他的bar样式和事件。

基于以上两点情况，我们可以在weex-ui源码的基础上简化，实现我们想要的效果。

#### 复制源码

源码参考：<https://github.com/alibaba/weex-ui/blob/master/packages/wxc-tab-bar/index.vue>

#### 删除多余代码

* 删除template 第一段

<div class="tab-page-wrap" ref="tab-page-wrap">  
 <div ref="tab-container"  
 class="tab-container">  
 <slot></slot>  
 </div>  
</div>

* 删除template 第二段

<text v-if="titleType === 'iconFont' && v.codePoint && !titleUseSlot"  
 :style="{fontFamily: 'wxcIconFont',fontSize: tabStyles.iconFontSize+'px', marginBottom:tabStyles.iconFontMarginBottom ? (tabStyles.iconFontMarginBottom +'px') : '8px', color: currentPage == index ? tabStyles.activeIconFontColor : tabStyles.iconFontColor}">{{v.codePoint}}</text>

* 删除style

.tab-page-wrap {  
 width: 750px;  
 flex: 1;  
 /\* overflow: hidden; \*/  
}  
.tab-container {  
 flex: 1;  
 flex-direction: row;  
 position: absolute;  
}

* 删除script 第一段

删除methods里面的除setPage之外的全部方法。

#### 优化代码

* template优化

把image元素换成text元素，我们使用Fontawesome字体作为我们的图标。

<text class="icon" :style="{color:getFillColor(index)}">{{v.icon}}</text>

* style 优化

增加以下样式：

.icon{  
 color: #666666;  
 font-family: fontawesome;  
 font-size: 48px;  
 background-color: transparent;  
}  
.wxc-tab-page {  
 width: 750px;  
 position: absolute;  
 left: 0;  
 bottom: 0;  
 background-color: #f2f3f4;  
}

* script 优化

<script>  
 const dom = weex.requireModule('dom');  
 const animation = weex.requireModule('animation');  
 import { Utils } from '\_weex-ui@0.6.14@weex-ui';  
 import config from './config';  
  
 export default {  
 props: {  
 tabTitles: {  
 type: Array,  
 default: () => (config.tabTitles)  
 },  
 tabStyles:{  
 type: Object,  
 default: () => ({  
 bgColor: '#f2f3f4',  
 titleColor: '#666666',  
 activeTitleColor: '#3194FF',  
 activeBgColor: '#f2f3f4',  
 activeIconColor:'#3194FF',  
 iconColor:'#666666',  
 isActiveTitleBold: true,  
 iconWidth: 70,  
 iconHeight: 70,  
 width: 160,  
 height: 120,  
 fontSize: 24,  
 activeBottomColor: '#FFC900',  
 activeBottomWidth: 120,  
 activeBottomHeight: 6,  
 textPaddingLeft: 10,  
 textPaddingRight: 10  
 })  
 },  
 titleUseSlot: {  
 type: Boolean,  
 default: false  
 }  
 },  
 data: () => ({  
 currentPage: 0  
 }),  
 created () {  
 this.isIPhoneX = Utils.env.isIPhoneX();  
 },  
 methods: {  
 getFillColor (index){  
 if(this.currentPage==index){  
 return this.tabStyles.activeIconColor;  
 }else{  
 return this.tabStyles.iconColor;  
 }  
 },  
 setPage(page, url = null, animated = true) {  
 this.currentPage = page;  
 }  
 }  
 };  
</script>

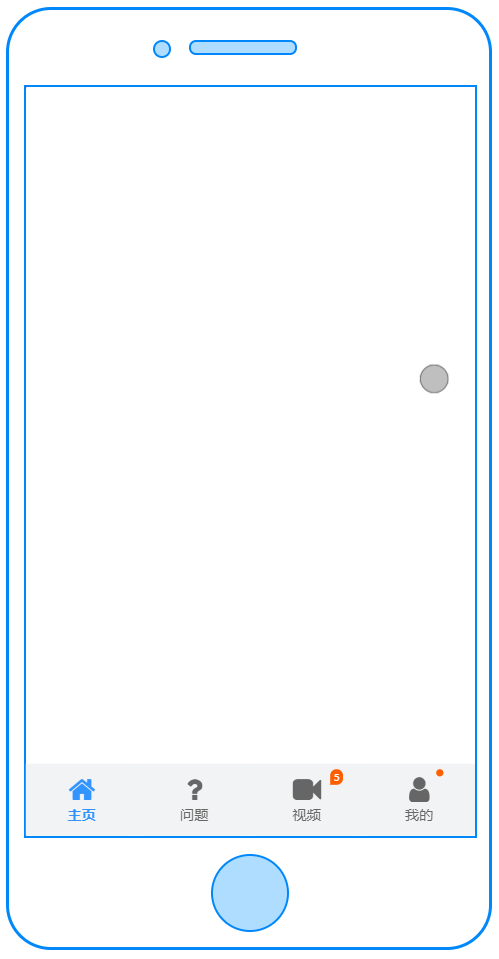
### 实现菜单Model

菜单Model用于设置菜单的文字、图标等内容，图标编码可参考：

<https://fontawesome.com/cheatsheet?from=io>

export default {  
 tabTitles: [  
 {  
 title: '主页',  
 icon:'\uf015'  
 },  
 {  
 title: '问题',  
 icon:'\uf128'  
 },  
 {  
 title: '视频',  
 badge: 5,  
 icon:'\uf03d'  
 },  
 {  
 title: '我的',  
 dot: true,  
 icon:'\uf007'  
 }  
 ]  
}

### 效果演示



## 搜索组件

### 文件创建

* 在compoents\bars文件夹下新建home\_bar.vue文件，存放首页搜索组件的代码

### 实现VIEW

首页搜索组件，放置在顶部，参考<https://alibaba.github.io/weex-ui/#/cn/packages/wxc-minibar/>组件实现；重新定义了左中右的子组件。

注：这里只做VIEW的开发，具体功能后续实现。

<template>  
 <wxc-minibar background-color="#009ff0" @wxcMinibarLeftButtonClicked="minibarLeftButtonClick"  
 @wxcMinibarRightButtonClicked="minibarRightButtonClick">  
 <text class="icon"  
 slot="left"  
 style="text-align: left">&#xf039;</text>  
 <div slot="middle" class="middle">  
 <text class="icon middle-icon" >&#xf002;</text>  
 <text style="color:#6d6d6d;font-size: 24px" class="middle-title"> 请输入搜索关键词</text>  
 </div>  
 <text class="icon"  
 slot="right"  
 style="text-align: right">&#xf002;</text>  
 </wxc-minibar>  
</template>  
  
<script>  
 import { WxcMinibar } from '\_weex-ui@0.6.14@weex-ui';  
 const modal = weex.requireModule('modal');  
  
 export default {  
 name: "index",  
 components: { WxcMinibar},  
 methods: {  
 minibarLeftButtonClick () {  
 },  
 minibarRightButtonClick () {  
 modal.toast({ 'message': 'click rightButton!', 'duration': 1 });  
 }  
 }  
 };  
</script>  
  
<style scoped>  
 .icon{  
 color: #ffffff;  
 font-family: fontawesome;  
 font-size: 28px;  
 }  
 .middle{  
 border: 1px #ffffff solid;  
 background-color: #ffffff;  
 border-radius: 20px;  
 flex: 1;  
 flex-direction : row;  
 cursor: pointer;  
 }  
 .middle-icon{  
 height: 24px;  
 width: 30px;  
 margin-top: 4px;  
 margin-left: 5px;  
 color: #adadad;  
 }  
</style>

## 无图列表项

### 文件创建

* 创建src/styles/article.less文件，用于存放文章列表的公共样式
* 创建src/compoents/cells/article\_0.vue文件，用于实现无图列表项组件
* 创建src/utils/date.js文件，用于实现项目日志格式化功能

### 通用样式

通用样式包括无图、单图、三图三个列表项的公共样式。

// 基本样式-颜色  
@body-background : #fff;  
@component-background : #fff;  
// 基本样式-字体  
@font-size : 12px;  
@font-family : "Helvetica Neue",Helvetica,"PingFang SC","Hiragino Sans GB","Microsoft YaHei","微软雅黑",Arial,sans-serif;  
@code-family : Consolas,Menlo,Courier,monospace;  
@icon-font :fontawesome;  
// 主页文章列表样式 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ START  
@list-0-height :160px;  
@list-1-height :190px;  
@list-2-height :270px;  
// 列表通用参数  
@list-lr-margin :10px;  
@list-tb-margin :5px;  
@list-bottom-color :#dfdfdf;  
  
@title-size :36px;  
@title-family :@font-family;  
@title-color :#666666;  
  
@tag-size :18px;  
@tag-family :@font-family;  
@tag-color :#666666;  
@tag-right-margin :21px;  
  
@tag-icon-color :red;  
@tag-icon-size :24px;  
@tag-icon-font :@icon-font;  
@tag-icon-bg :transparent;  
  
.list-item {  
 width: 750px;  
 flex-direction: column;  
}  
.title {  
 font-size: @title-size;  
 font-family: @title-family;  
 color: @title-color;  
 justify-content:center;  
 align-items: center;  
 margin: @list-tb-margin @list-lr-margin;  
}  
  
.tags-text{  
 font-size: @tag-size;  
 color: @tag-color;  
 font-family: @tag-family;  
 margin-right: @tag-right-margin;  
}  
.tags-icon{  
 font-family: @tag-icon-font;  
 font-size: @tag-icon-size;  
 color: @tag-icon-color;  
 background-color: @tag-icon-bg;  
}  
.tags{  
 flex-direction: row;  
 margin: @list-tb-margin @list-lr-margin;  
 justify-content: left;  
}  
.date {  
 margin-top: 3px;  
}  
.line {  
 height: 2px;  
 background-color: @list-bottom-color;  
 margin: @list-lr-margin;  
}  
.loading {  
 flex-direction: row;  
 justify-content: center;  
 align-items: center;  
 width: 750px;  
 height: 100px;  
}  
.loading-text{  
 font-size: @title-size/1.1;  
 color: @list-bottom-color;  
}  
.loading-icon{  
 width:100px;  
 height: 100px;  
 color:saturate(@list-bottom-color,120%);  
}  
.image{  
 border-radius: 5px;  
}  
// 主页文章列表样式 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ ENT

### Model定义

Model是列表传给列表项展示的参数，其必要参数项包含如下：

{

id:1,  
 title:'这是测试标题sssssdfsdfs近段时间技术的uu吗水电费水电费水电费收费',  
 tags:'这是来源',  
 source:'这是来源',  
 commit:1,  
 date:1559789043,

image:[‘地址’],  
 icon:"\uf06d",  
}

### 时间格式化

#### 时间格式化

##### 功能实现

项目中的时间支持10位、13位的数字串，能格式化可阅读的时间，以及与当前时间对比显示，其实现代码如下：

function FormatDate(){}  
FormatDate.prototype= {  
 formatDate:function(date, fmt) {  
 if (/(y+)/.test(fmt)) {  
 fmt = fmt.replace(RegExp.$1, (date.getFullYear() + '').substr(4 - RegExp.$1.length))  
 }  
 let o = {  
 'M+': date.getMonth() + 1,  
 'd+': date.getDate(),  
 'h+': date.getHours(),  
 'm+': date.getMinutes(),  
 's+': date.getSeconds()  
 }  
 for (let k in o) {  
 if (new RegExp(`(${k})`).test(fmt)) {  
 let str = o[k] + ''  
 fmt = fmt.replace(RegExp.$1, RegExp.$1.length === 1 ? str : this.padLeftZero(str))  
 }  
 }  
 return fmt  
 },  
 padLeftZero:function (str) {  
 return ('00' + str).substr(str.length)  
 },  
 format10:function(time){  
 return this.format13(time\*1000);  
 },  
 format13:function(time){  
 let date = new Date(time);  
 return this.formatDate(date,'yyyy-MM-dd')  
 },  
 diffTime:function(time){  
 if(time.length==10){  
 time = parseInt(time)\*1000;  
 }  
 var nowDate = new Date().getTime(),  
 oldDate = new Date(time).getTime(),  
 diffTime = parseInt((nowDate - oldDate)/1000,10),  
 oneMinute = 60,  
 oneHour = 60 \* oneMinute,  
 oneDay = 24 \* oneHour,  
 oneMonth = 30 \* oneDay,  
 oneYear = 12 \* oneMonth,  
 compareArr = [oneYear,oneMonth,oneDay,oneHour,oneMinute],  
 postfix = ['年前','个月前','天前','个小时前','分钟前','1分钟内'],  
 diffYear,diffMonth,diffDay,diffHour,diffMinute,len=5;  
 for(var i =0; i< len ;i++){  
 var diff = Math.floor(diffTime/compareArr[i]);  
 if(diff > 0){  
 return diff + postfix[i];  
 }  
 else if(i === len -1 && diff === 0){  
 return postfix[len];  
 }  
 }  
 }  
}  
export default new FormatDate()

##### 全局设置

在src/entry.js中配置全局变量：

import date from '@/utils/date'  
Vue.prototype.$date = date

### 实现VIEW

无图列表项是一个column布局，其中的辅助信息为row布局，整体代码如下：

<template>  
 <div class="list-item">  
 <text class="title">{{data.title}}</text>  
 <div class="tags">  
 <text class="tags-text tags-icon">{{data.icon}}</text>  
 <text class="tags-text">{{data.source}}</text>  
 <text class="tags-text">评论 {{data.commit}}</text>  
 <text class="tags-text date">{{formatDate(data.date)}}</text>  
 </div>  
 <div class="line"></div>  
 </div>  
</template>  
  
<script>  
 export default {  
 name: "article\_0",  
 props:{  
 data:{  
 type:Object  
 }  
 },  
 methods : {  
 formatDate:function(time){  
 return this.$date.format10(time);   
 }  
 }  
 }  
</script>  
  
<style lang="less" scoped>  
 @import '../../styles/article';  
 .list-item{  
 height: @list-0-height;  
 }  
</style>

## 单图列表项

### 文件创建

* 创建src/compoents/cells/article\_1.vue文件，用于实现单图列表项组件

### 实现VIEW

单图列表项Model和样式与无图列表项公用，其是一个row布局，其中的辅助信息为row布局，整体代码如下：

<template>  
 <div class="list-item">  
 <div class="list-lr">  
 <div class="item-l">  
 <text class="title">{{formatTitle(data.title)}}</text>  
 <div class="tags">  
 <text class="tags-text tags-icon">{{data.icon}}</text>  
 <text class="tags-text">{{data.source}}</text>  
 <text class="tags-text">评论 {{data.commit}}</text>  
 </div>  
 </div>  
 <div class="item-r">  
 <image class="image" :src="data.image[0].url"/>  
 </div>  
 </div>  
 <div class="line"></div>  
 </div>  
</template>  
  
<script>  
 export default {  
 name: "article\_1",  
 props:{  
 data:{  
 type:Object  
 }  
 },  
 methods : {  
 formatDate:function(time){  
 return this.$date.format10(time);  
 },formatTitle:function(title){  
 if(title.length>32){  
 return title.substring(0,31);  
 }  
 return title;  
 }  
 }  
 }  
</script>  
  
<style lang="less" scoped>  
 @import '../../styles/article';  
 .list-item{  
 height: @list-1-height;  
 }  
 .list-lr{  
 flex-direction: row;  
 justify-content: space-around;  
 }  
 .item-l{  
 width: 550px;  
 }  
 .item-r{  
 width: 180px;  
 margin: @list-tb-margin @list-lr-margin;  
 }  
</style>

## 三图列表项

### 文件创建

* 创建src/compoents/cells/article\_3.vue文件，用于实现三图列表项组件

### 实现VIEW

三图列表项Model和样式与无图列表项公用，其是一个row布局，其中的辅助信息为row布局，整体代码如下：

<template>  
 <div class="list-item">  
 <text class="title">{{formatTitle(data.title)}}</text>  
 <div class="item-image">  
 <image class="image" v-for="img in data.image" :src="img.url"></image>  
 </div>  
 <div class="item-l">  
 <div class="tags">  
 <text class="tags-text tags-icon">{{data.icon}}</text>  
 <text class="tags-text">{{data.source}}</text>  
 <text class="tags-text">评论 {{data.commit}}</text>  
 <text class="tags-text">{{formatDate(data.date)}}</text>  
 </div>  
 </div>  
 <div class="line"></div>  
 </div>  
</template>  
  
<script>  
 export default {  
 name: "article\_3",  
 props:{  
 data:{  
 type:Object  
 }  
 },  
 methods : {  
 formatDate:function(time){  
 return this.$date.format10(time);  
 },  
 formatTitle:function(title){  
 if(title.length>36){  
 return title.substring(0,31);  
 }  
 return title;  
 }  
 }  
 }  
</script>  
  
<style lang="less" scoped>  
 @import '../../styles/article';  
 .list-item{  
 height: @list-2-height;  
 }  
 .item-image{  
 flex-direction: row;  
 padding: 0px 10px;  
 justify-content: space-around;  
 }  
 .image{  
 width: 240px;  
 }  
</style>