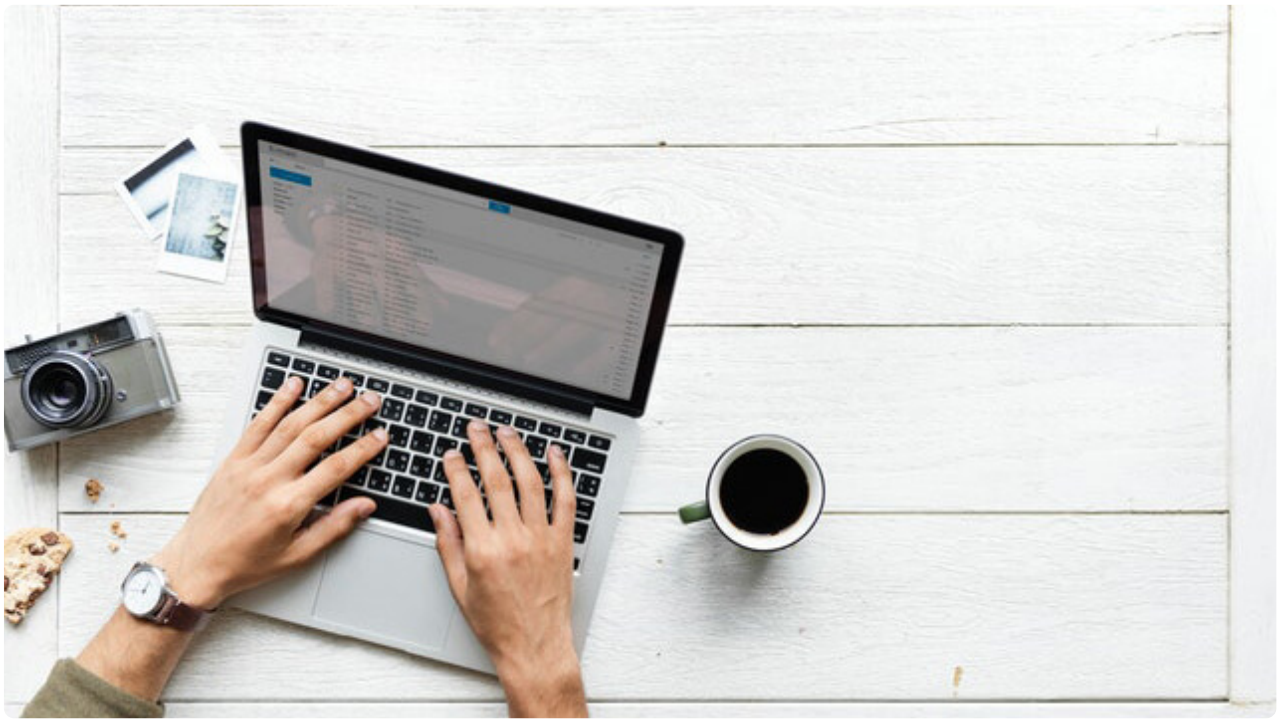
30 页面转场动画

更新时间：2019-09-27 09:38:33



> 没有智慧的头脑，就像没有蜡烛的灯笼。
>
> ——托尔斯泰

到本章节为止，我们所有的页面业务相关的逻辑都已完成，接下来的任务就是对之前页面，做响应的优化，首先，来做一下页面转场效果的优化，这里将会结合vue-router的监听路由变化和Animate.css来实现，同学可以事先了解一下基本的知识点，传送门。Vue Router，Animate.css

本章节完整源代码地址，大家可以事先浏览一下：

Github

页面转场动画概述

页面专场也可以叫做页面跳转，意思就是从一个页面跳转到另外一个页面，传统的H5项目页面跳转分为2种场景：

1. 单页面的H5项目：由于所有页面都是在一个html里面，所以页面的切换一般是将上一个页面的dom内容移除，将下一个页面的dom内容append上去，在此之间可以做一些动画效果，例如从左往右，从下往上。
2. 多页面的H5项目：由于是多个html，可以采取`window.open('page.html')`来实现页面的跳转，此跳转在页面返回时，可能不会保留上一个页面的状态，例如输入框的值，滚动的距离等等，如果H5页面是内嵌在一个HybridApp里面，也可以借助Native提供的接口来实现页面跳转，这样的跳转体验要好很多，可以利用Native的页面切换动画，同时在页面返回时可以保持上一个页面的状态，实际上是一个多webview应用，大家可以参考微信-》发现-》游戏这里的业务，就是采用这种方案。

我们的项目是一个单页面的H5项目，所以采用第一种场景，来实现页面的转场切换动画，做好转场切换动画是衡量一个H5项目用户体验的一个关键指标，是让用户用起来像是一个 Native APP 的重要体现，下面就来实现它，先看一下最终效果如下：

接下来，就让我们进入改造开发吧！

vue-router的router切换

我们的项目是采用的 vue-router 来实现的页面路由，所以页面的切换也是集成在vue-router里面，还记得前面的代码中，采用 this.$router.push({}) 方法实现页面切换。

在vue-router的接口中，可以在根实例中，借助 watch 来监听 $route 找到切换前和切换后时间点的钩子，从而来添加我们动画相关的逻辑。

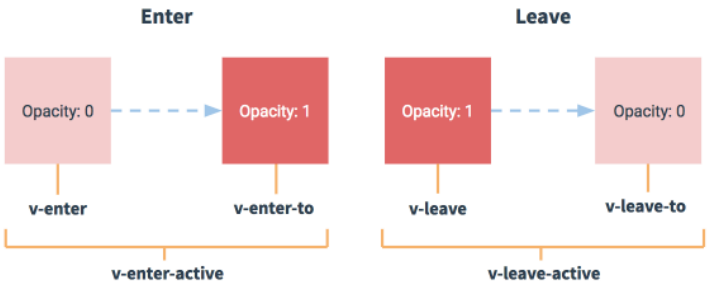
找到前端项目的入口文件 App.vue，并修改下面代码：

```
watch: {// 使用watch 监听$router的变化
  $route (to, from) {
    ...
    //to 表示下一个页面的路由
    //from 表示上一个页面的路由
  }
}
```

使用和animate.css实现页面切换动画

使用方法：

在之前章节，我们讲解过vue中 <transition> 实现动画的方法：



这里，我们将 <transition> 直接应用在 <router-view> 就可以达到对路由页面的过渡动画添加。代码如下。

首先将 <router-view> 用 <transition> 包裹：

```
<div id="app">
  <transition :enter-active-class="transitionNameIn" :leave-active-class="transitionNameOut" :duration="duration">
    <router-view></router-view>
  </transition>
</div>
```

上面代码逻辑总结一下：

1. enter-active-class 和 leave-active-class 分别表示动画执行时，使用的class类名。
2. duration 表示动画执行的时间，这里不采用CSS的动画时间，我们需要手动指定。
3. class 类名就用到了Animate.css的动画名称。例如fadeIn，slideIn等等这些。

下载**animate.css**:

Animate.css是一个使用CSS3的animation制作的动画效果的开源CSS效果集合，里面预设了很多种常用的动画，且使用非常简单，我们在项目中主要用来实效一些特效动画，以及结合vue-router实现各种页面转场效果。

GitHub地址：https://daneden.github.io/animate.css/

在前端项目中，找到 main.js 并里引入：

```
import './assets/animate.css'
```

这里说明一下为何要在JavaScript里引入而不是直接在 html 页面上：

由于我们的项目采用了 vw 的适配方案，所以所有的 css 都需要经过 PostCss 插件 postcss-px-to-viewport 来处理，结合webpack，我们在这里采用 import 时，就会应用上对应的loader来处理了。同理，我们之前的common.css也要这样引入。

然后，修改上面代码中预留 watch 里的逻辑，增加动画功能：

```
watch: {// 使用watch 监听$router的变化
  $route (to, from) {

    // 持续时间
    this.duration = 500
    // 从下往上切换
    if (to.name === 'publish' || to.name === 'login') {
      this.transitionNameIn = 'animated faster slideInUp'
      this.transitionNameOut = 'slideOutIng'

    } else if (from.name === 'publish' || from.name === 'login') {
      this.transitionNameIn = ''
      this.transitionNameOut = 'animated faster slideOutDown'

    }
    // 从左往右切换
    else {
      // 后退
      if (this.$router.backFlag) {
        this.transitionNameOut = 'animated faster slideOutRight'
        this.transitionNameIn = 'animated faster slideInLeft'
      } else { // 前进
        this.transitionNameIn = 'animated faster slideInRight'
        this.transitionNameOut = 'animated faster slideOutLeft'
      }
    }
    //重置返回的标志位
    this.$router.backFlag = false
  }
}
```

1. 这里我们实现了2种动画效果，一种是从下往上，一种是从左往右。
2. 默认情况下使用从左往右效果，发表和登录页面使用从下往上。

`slideOutRight`和`slideInLeft`和`slideOutDown`和`slideInUp`等等这些class类名，分别可以在Animate.css的源码里找到，代码如下：

```
@-webkit-keyframes slideInUp {
  from {
    -webkit-transform: translate3d(0, 100%, 0);
    transform: translate3d(0, 100%, 0);
    visibility: visible;
  }

  to {
    -webkit-transform: translate3d(0, 0, 0);
    transform: translate3d(0, 0, 0);
  }
}

@keyframes slideInUp {
  from {
    -webkit-transform: translate3d(0, 100%, 0);
    transform: translate3d(0, 100%, 0);
    visibility: visible;
  }

  to {
    -webkit-transform: translate3d(0, 0, 0);
    transform: translate3d(0, 0, 0);
  }
}

.slideInUp {
  -webkit-animation-name: slideInUp;
  animation-name: slideInUp;
}
```

faster 是Animate.css里内置动画执行的时间，类似的还有 fast， slow， delay-xx 等等，我们将 faster 统一修改成了500ms，直接修改 animate.css 的源码，找到 .animated.faster ，修改如下：

```
.animated.faster {
  -webkit-animation-duration: 500ms;
  animation-duration: 500ms;
}
```

页面切换动画注意点

由于页面切换动画大部分是基于 css3 的 transform 位移属性，所以当页面比较长时可能会出现页面跳动的情况，尤其在从下往上的动画中，所以可以在 transitionNameOut 上一个页面即将离开时，将页面设置成绝对定位，切换完成后在复原，来 hack 解决一下，在 App.vue 中添加代码如下：

```
.slideOutIng {
  position: absolute;
  left: 0;
  right: 0;
}
```

在朋友圈首页，但我们滚动到中间时，如果此时点击了头像，页面会切换到个人信息页面，但是如果返回之后，页面会回到顶部，这里我们需要记录一下页面跳转前的位置，好在 vue-router 给我们提供了这个功能 scrollBehavior ，来解决这个问题，在 App.vue 中添加代码如下：

```
scrollBehavior (to, from, savedPosition) {
  if (savedPosition) {
    return savedPosition
  } else {
    return { x: 0, y: 0 }
  }
}
```

注意，此功能只是用于 body 滚动的情况，局部滚动是不适合的哦。

小结

本章节主要讲解了页面切换的基本方案。

相关知识点：

1. 页面转场用户交互的流程讲解。
2. vue-router 利用`watch` `$route`来监听页面路由切换事件。
3. 结合Animate.css和`<transition>`，实现页面转场的动画效果。

本章节完整源代码地址：

Github

}

29 聊天列表页面后端接口开发

31 使用 PWA 改造真正的 webApp