电商项目实战03

知识要点

header组件

1. 显示标题
2. 返回按钮，返回上一个路由
3. 支持右侧扩展按钮

```
<template>

<div class="header">
  <h1>{{title}}</h1>
  <i v-if="showback" @click="back" class="cubeic-back"></i>
  <div class='extend'>
    <slot></slot>
  </div>
</div>
</template>

<script>
export default {
  props:{
    title:{
      type:String,
      default:'',
      required:true
    },
    showback:{
```

```
      type:Boolean,
      default:false
    }
  },
  methods:{
    back(){
      this.$router.back()
    }
  }
}
</script>

<style lang="stylus">
.header
  position relative
  height 44px
  line-height 44px
  text-align center
  background #edf0f4
  .cubeic-back
    position absolute
    top 0
    left 0
    padding 0 15px
    color #fc915b
  .extend
    position absolute
    top 0
    right 0
    padding 0 15px
    color #fc915b
</style>
```

Home.vue

```
<k-header title="开课吧">
  <i class="cubeic-tag" @click='showCatg'></i>
</k-header>
```

动画

tabbar移动到router-view下方，设置绝对定位

```
.cube-tab-bar
  position fixed
  bottom 0
  left 0
  right 0
  background #edf0f4
```

添加购物车右侧显示Goodlist.vue

```
.right
    margin-left 120px
    text-align  right
```

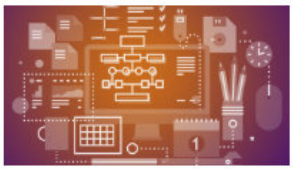
开课吧

Vue2.x实战

100人购买

React16.x实战

100人购买

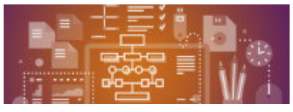


nodejs实战

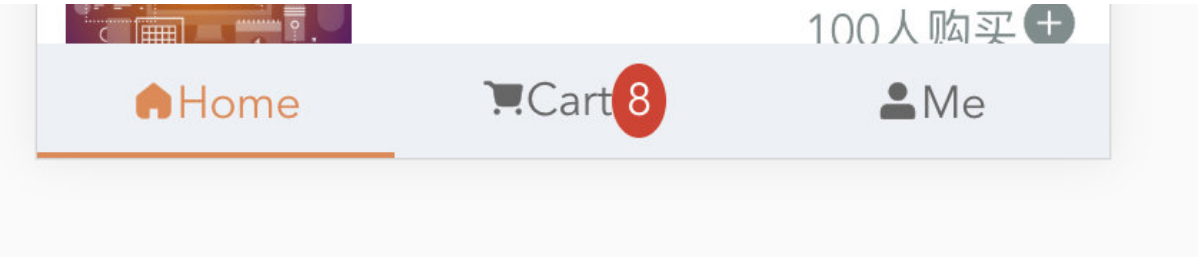
100人购买

前端工程化

100人购买



面试



为了实现购物车的动画，我们需要新增一个小球，默认隐藏在购物车里，点击购物车的时候，小球移动到点击的位置 显示，然后动画飞回去即可

1. 新增动画小球 默认隐藏在购物车的位置
2. 点击获取dom位置
3. 小球显示 并且移动到dom位置
4. 动画飞回去
5. 小球隐藏

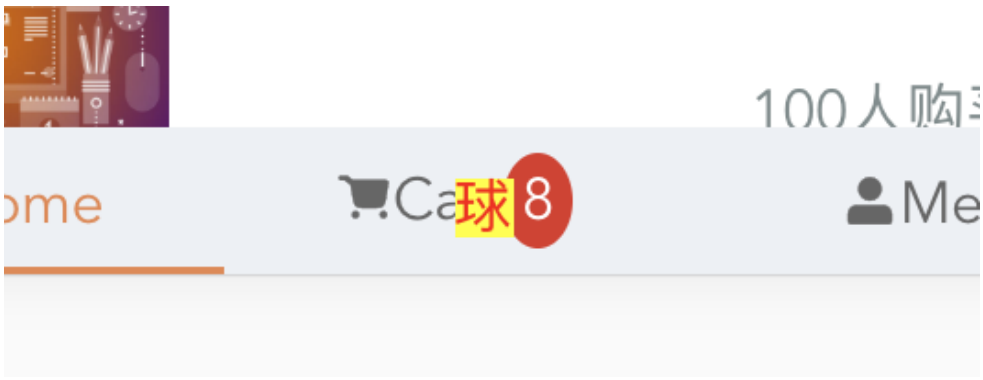
动画小球

Home.vue里新增小球 显示在购物车的位置

```
ball:{
    show:true, // 先显示出来调试
    el:null // 目标dom
}
```

```
<div class="ball-wrap">
  <transition
    @before-enter="beforeEnter"
    @enter="enter"
    @afterEnter="afterEnter"
  >
    <div class="ball" v-show="ball.show">
        球
    </div>
  </transition>
</div>
```

```
.ball-wrap
  .ball
    position fixed
    left 50%
    bottom 10px
    z-index 200
    color red
```



开始动画

点击事件在goodlist组件 使用$emit通知父组件

```
<i class="cubeic-add" @click.stop.prevent="addCart($event,item)"></i>
```

```
addCart(event,item){
  this.$store.commit('addcart',item)
  // 把点击的dom传递出去了
  this.$emit('addcart',event.target)
},
```

Home中监听addcart 把默认ball的show改为false 点击之后再显示

```
<goods-list @addcart="onAddcart" :data="all"></goods-list>
```

```
onAddcart(el){
  this.ball.show = true // 显示ball
  // 获取点击的元素 待会要用来获取位置
  this.ball.el = el
},
```

开始动画

使用vue动画的js钩子 https://cn.vuejs.org/v2/api/#transition

```
<transition
@before-enter="beforeEnter"
@enter="enter"
@afterEnter="afterEnter"
>
```

```
beforeEnter(el){
  // 小球移动到点击的位置
  // 1. 获取点击的dom
  const dom = this.ball.el
  const rect = dom.getBoundingClientRect()
  console.log(rect.top,rect.left)
  // 小球移动到点击的位置
  const x = rect.left-window.innerWidth/2
  const y = -(window.innerHeight-rect.top-10-20)
  el.style.display = ''
  el.style.transform = `translate3d(${x}px, ${y}px, 0)`

}
```


开课吧
top
left
点击位置
屏幕高度-top-字体
left-屏幕宽度/2
Vue2.x实战
100人购买
React16.x实
100人购买
100人购买
前端工程化
100人购买
面试
Home
Cart
11
Me

开始动画

enter钩子，把小球移动到初始位置 加上动画

```
enter(el, done){
    // 获取offsetHeight就会重绘，前面的变量名随意 主要为了eslint校验
  this._kaikeba = document.body.offsetHeight
  // 动画开始，移动到初始位置
  // 小球移动到购物车位置
  el.style.transform = `translate3d(0, 0, 0)`

  el.addEventListener("transitionend",done)
},
afterEnter(el){
  // 结束 隐藏小球
  this.ball.show = false
  el.style.display = 'none'
}
```

设置css动画 贝塞尔曲线

```
.ball-wrap
  .ball
    position fixed
    left 50%
    bottom 10px
    z-index 200
    background yellow
    color red
    transition all 0.5s cubic-bezier(0.49, -0.29, 0.75, 0.41)
```

如果动画更好看些，向做一个抛物线，就需要x洲和y轴两个动画，y轴贝塞尔曲线，x轴匀速运动

ball负责y轴吗，内部添加一个inner的dom，负责x轴，

```
<div class="ball" v-show="ball.show">
    <div class="inner">
        <div class="cubeic-add"></div>
    </div>
</div>
```

```
beforeEnter(el){
  // 小球移动到点击的位置
  // 1. 获取点击的dom
  const dom = this.ball.el
  const rect = dom.getBoundingClientRect()
  console.log(rect.top,rect.left)
  // 小球移动到点击的位置
  const x = rect.left-window.innerWidth/2
  const y = -(window.innerHeight-rect.top-10-20)
  el.style.display = ''
  // ball只移动y
  el.style.transform = `translate3d(0, ${y}px, 0)`
```

```
    const inner = el.querySelector('.inner')
    // inner只移动x
    inner.style.transform = `translate3d(${x}px,0,0)`
  },
  enter(el, done){
    this._kaikeba = document.body.offsetHeight
    // 动画开始，移动到初始位置
    // 小球移动到购物车位置
    el.style.transform = `translate3d(0, 0, 0)`
    const inner = el.querySelector('.inner')
    inner.style.transform = `translate3d(0,0,0)`
    el.addEventListener("transitionend",done)
  },
```

```
.ball-wrap
  .ball
    position fixed
    left 50%
    bottom 10px
    z-index 200
    color red
    transition all 0.5s cubic-bezier(0.49, -0.29, 0.75, 0.41)
    .inner
      width 16px
      height 16px

      transition all 0.5s linear
.cubeic-add
  font-size 22px
```

自定义事件

登录失败的时候，我们使用了cube-ui的toast服务，如果我们想自己设计一个alert报错 大概长这个样子

```
<div class="alert" v-if="showError">
    {{errorMsg}}
</div>
```

```
this.showError=true
this.errorMsg = ret.message||'未知错误'
```

如果想把这个alert设计为公用组件，缺点贼明显

1. 需要预先将 `<Alert>` 放置在模板中；
2. data里定义数据 来控制 Alert 的显示状态；
3. Alert 的位置，是在当前组件位置，并非在 body 下，有可能会被其它组件遮挡。也有可能会被父元素的transform属性影响

如何在js里生成一个组件呢，我们需要$mount手动挂在组件

$mount

先把组件写好，支持显示多个

```
<template>
  <div class="alert">
    <div class="alert-container" v-for='item in alerts' :key="item.id">
      <div class="alert-content">
          {{item.content}}
      </div>
    </div>
  </div>
</template>

<script>
export default {
  data(){
    return {
      alerts:[]
    }
  },
  created(){
    // 不是响应式的 所以不放在data里
    this.id = 0
  },
  methods:{
    add(option){
      // 新增消息
      const id = 'id_'+(this.id++)
      const _alert = {...option, id:id}
      this.alerts.push(_alert)
      // 延迟关闭
      const duration = option.duration
      setTimeout(()=>{
        this.del(id)
      },duration*1000)
    },
    del(id){
      // 删除消息
      for(let i=0;i<this.alerts.length;i++){
        if(this.alerts[i].id==id){
          this.alerts.splice(i,1)
          break
        }
      }
    }
  }
}
</script>

<style lang="stylus">
```

```
.alert
  position fixed
  width 100%
  top 30px
  left 0
  text-align center
  .alert-content
    display inline-block
    padding 8px
    background #fff
    margin-bottom 10px
</style>
```

$Notice服务

新建notification.js

```
import Notice from './Notice.vue'
import Vue from 'vue'

// 手动创建Alert组件，加载到body里，不受付组件影响

Notice.newKaikeba = props=>{
  const Instance = new Vue({
    data(){
      return props || {}
    },
    render(h){
      // <Notive a="1" b="2">
      // h就是类似react李的createElement
      return h(Notice,{
        props
      })
    }
  })
  const comp = Instance.$mount() //渲染
  console.log(comp)
  // 挂载在body之上 而不是组件内部
  document.body.appendChild(comp.$el)
  // notive还是vue的实例 并不是dom
  const notice = Instance.$children[0]

  // notice的实例
  return {
    add(optipns){
      notice.add(optipns)
    },
    del(id){
      notice.del(id)
    }
```

```
  }
}
export default Notice
```

notice.js

对外暴露的notice.js

```
// 最终对外暴露的接口
import Notification from './notification'

let msgInstance

function getInstance(){
  msgInstance = msgInstance || Notification.newKaikeba()
  return msgInstance
}
function info({duration=2, content=""}){
  let ins = getInstance()
  ins.add({
    content,
    duration
  })
}
export default {
  info
}
```

mainjs挂在在原型链上，类似于$axios

```
Vue.prototype.$notice = notice
```

使用

```
this.$notice.info({
  duration:3,
  content:ret.message||'未知错误'
})
```

思考

1. 贝塞尔
    1. https://blog.csdn.net/wjnf012/article/details/78795573
2. 浏览器重绘机制

1. https://blog.fundebug.com/2019/01/03/understand-browser-rendering/

> **1）常见引起回流属性和方法**
>
> 任何会改变元素几何信息(元素的位置和尺寸大小)的操作，都会触发回流，
>
> - 添加或者删除可见的DOM元素；
> - 元素尺寸改变——边距、填充、边框、宽度和高度
> - 内容变化，比如用户在input框中输入文字
> - 浏览器窗口尺寸改变——resize事件发生时
> - 计算 offsetWidth 和 offsetHeight 属性
> - 设置 style 属性的值

作业

1. 如果频繁点击，怎么实现多个ball的动画

回顾