day05

课程回顾

购物车的案例

全选和单选
表单标签我们没有click，只有change事件
查漏补缺，数组操作方法，
map,forEach,every,some,filter...

过滤器

文本格式化，封装了一个工具函数
创建方式
一、局部创建过滤器
视图 通过“|”
{{data数据 | 过滤器名称}}

```
//代码
filters:{
    //你要创建的过滤器
    过滤器名称(被过滤的内容，被过滤的内容的传参){
        return 你要的格式
    }
}
```

二、全局创建过滤器
视图 通过“|”
{{data数据 | 过滤器名称}}

```
Vue.filter(过滤器名称,()=>{
    //过滤器的内容
})
```

过渡动画

一、内置过渡动画
六种状态
离开
leave
leave-active
leave-to
进入
enter
enter-active
enter-to

二、引入动画库
http://animate.style
下载命令：
npm i animate.css

组件

组件的特点：可复用
为什么要创建组件？
提高开发效率，节省开发时间，利于代码优化
创建组件的方式
一、局部创建组件
组件在渲染的时候当做标签去使用

```
components:{
    组件名称：{
        //组件的属性方法
```

```
    }
}
二、全局创建组件
Vue.component('组件名称',{
    //组件的属性和方法
})
组件的嵌套
一定要分析清楚逻辑层级
在哪里注册在哪里渲染
```

动态组件

```
<component is='组件名称'></component>
相当于
<组件名称></组件名称>

如果要切换组件
<组件名称 v-if=''></组件名称>
<组件名称 v-if=''></组件名称>
<组件名称 v-if=''></组件名称>
这么写比较麻烦，我们可以进行代码优化，通多动态组件切换
<component :is='变量'></component>
```

实际案例之选项卡

通过v-show去切换你的选项卡

```
视图
<div class="nav">
        <span @click="select=1" :class="
[select==1?'active':'']">推荐音乐</span>
```

```
        <span @click="select=2" :class="
[select==2?'active':'']">排行榜</span>
        <span @click="select=3" :class="
[select==3?'active':'']">搜索</span>
      </div> -->

<div v-show="select==1" class="content">推荐音乐
的内容</div>
<div v-show="select==2" class="content">排行榜
</div>
<div v-show="select==3" class="content">搜索的内
容</div>

代码逻辑中
data:{
    select:1
}
```

通过v-if去操控组件

```
视图
<div class="nav">
        <span @click="select=1" :class="
[select==1?'active':'']">推荐音乐</span>
        <span @click="select=2" :class="
[select==2?'active':'']">排行榜</span>
        <span @click="select=3" :class="
[select==3?'active':'']">搜索</span>
      </div> -->

<commend v-if='select==1'></commend>
<rank  v-if='select==2'></rank>
<search  v-if='select==3'></search>
```

```
代码逻辑中
data:{
    select:1
}
```

通过动态is组件去切换

```
视图
<div class="nav">
        <span @click="changeTab='commend'"
:class="[changeTab=='commend'?'active':'']">推荐
音乐</span>
        <span @click="changeTab='rank'"
:class="[changeTab=='rank'?'active':'']">排行榜
</span>
        <span @click="changeTab='search'"
:class="[changeTab=='search'?'active':'']">搜索
</span>
      </div>
<component :is='changeTab'></component>
代码
data:{
    changeTab:'commend'
}
```

缓存组件

有一些组件内容，不需要变化，我们利用动态is组件切换的时候，每一次都重新加载dom，浪费性能
组件缓存的标签
<keep-alive>
    <组件名称></组件名称>
</keep-alive>
它包裹的内容就会被缓存。缓存的时候，前四个生命周期函数，beforeCreate/created/beforeMount/mounted 都只触发一次。

还有一些需求，缓存组件中，某几个组件，内容需要更新，只有activated(){} 这个函数被触发。 activated激活的意思
因为这个函数即使在缓存状态下，依然会被触发。

脚手安装

安装环境（这个步骤一台电脑只需要操作一次）
第一、全局安装webpack
npm install（i） -g webpack
查看版本 在小黑框中（cmd）输入 webpack -v
第二、全局安装脚手
npm install(i) -g @vuel/cli
这个命令下载的脚手架版本，是最新版本
查看版本 vue -V或者 vue --version

2.x旧环境命令（建议不用）
npm isntall(i) -g vue-cli
==========
=====
第三步创建项目
每次做项目的时候，你自己都应该创建一个新的项目
vue init webpack mydemo(项目名称)

这个命令，创建项目，创建的是2.x版本的项目
旧项目命令，在新环境下无法执行，我要初始化一下，搭建一个桥接工具
执行这个npm install(i) -g @vuel/cli-init
执行成功之后，我再创建项目： vue init webpack mydemo(项目名称)

第四、启动项目
进入项目文件夹，你看到node_modules
命令： npm run dev 或者 npm start

注意！！！
如果项目中没有node_modules,我们无法启动
npm install(i) 安装依赖

项目目录结构

```
README.md          阅读指南
package.json       配置管理文件
index.html         这是整个项目唯一的html文件
.postcssrc.js      css配置
.gitignore         用git上传的时候忽略的文件
.editorconfig      编辑器的配置
.babelrc           解析es6
static         静态资源文件夹，用于存储js，css,图片
node_modules       包依赖
build          打包构建的文件夹
    build.js       打包构建文件
    check-versions.js    版本检查，检查你本地的node和npm
    utils.js       实用工具
    vue-loader.conf.js vue加载器用来解析.vue文件
    webpack.base.conf.js     webpack基础配置
```

```
    webpack.dev.conf.js      webpack开发环境配置
    webpack.prod.conf.js     webpack生产环境配置
config       vue的配置文件夹
    index.js 这是基本配置
    dev.env.js   开发环境的配置
    prod.env.js 生产环境的配置

！！！！注意！！！
凡是修改配置文件，要重新启动服务器，重新运行，npm run dev
如果你只是修改组件，保存，浏览器会自动跟新，如果你没更新，自己手动刷新浏览器

src      代码集合地，我们的主战场
         assets   静态资源
         components   组件文件夹（里面是.vue文件）
         router   路由文件夹
             index.js   路由文件
         App.vue 根组件也可以叫主组件
         main.js 主的js文件，这里面是vue的实例化


assets和static有什么区别？
打包之后，assets里面的内容会被压缩，static里面的内容原样输出不会被压缩
```

单页应用

目前市场上主流的框架，vue，angular以及react(只是库)，创建出来的项目都是单页应用

什么是单页应用?

简单理解：只有一个html
SPA(single-page-application) 单页应用
单页和多页的区别？？？
单页：一个html。比较经典的一个案例：
https://es6.ruanyifeng.com/ 。es6这个网站就是一个单页应用。
优点：用户体验度高，切换的过程中不会整体刷新，不会出现空白页面
缺点：首次加载时，要加载出全部的内容，首屏加载过慢。不利于SEO优化
多页：多个html组件
优点：利于SEO优化，首次加载很快
缺点：切换的时候出现空白页面，用户体验度不高。切换页面过程中很慢

初始化页面结构

第一步、清空app.vue文件

```
<template>
  <div id="app">
  </div>
</template>

<script>
export default{
  //组件中的属性和方法
}
</script>

<style>

</style>
```

第二步、删除components文件夹下的默认的helloword.vue

第三步、在router文件夹下找到index.js 删除helloword.vue相关的内容

```
import Vue from 'vue'
import Router from 'vue-router'

Vue.use(Router)

export default new Router({
  routes: []
})
```

组件嵌套

```
第一步、 引入你创建好的组件（你要渲染谁，你就引入谁）
import 组件名称 from '组件的路径'
第二步、注册这个组件
components:{//局部创建注册组件
      组件名称：组件名称
}
第三步，在视图中渲染，把组件名称当做标签去使用
<组件名称></组件名称>
```

scoped

```
当前样式在当前组件有效
<style scoped>
</style>
```

组件通信

无论是面试还是做项目都是重点！！！！

父子组件通信

父组件

```
视图
<子组件  自定义属性的名称='你要传递的值' :自定义属性的名称='你要传递的变量'></子组件>
```

子组件

```
视图
{{自定义属性的名称}}
逻辑代码
props:['自定义属性的名称1','自定义属性的名称2','自定义属性的名称3'...]
```

props属性的验证

这个验证只是为了传值更加的严谨，如果你觉得麻烦，你可以不用写，你必须会看会用

props是自定义属性，可以接收，父组件传递的数据。

props:['自定义属性的名称1']，这种形式并没有对传递数据的类型进行验证

```
验证规则：
一、只验证数据类型
props:{

```

```
    自定义属性的名称:String/Number/Boolean/Function/Array,
    自定义属性的名称:[String,Number,Boolean]//多数据类型验证
}
二、添加默认值
props:{
    自定义属性的名称:{

type:String/Number/Boolean/Function/Array,
        default:''//默认值
    }
}
三、必填项的验证requried
props:{
    自定义属性的名称:{

type:String/Number/Boolean/Function/Array,
        default:''//默认值,
        requried:true
    }
}
四、自定义规则验证
props:{
    自定义属性的名称:{

type:String/Number/Boolean/Function/Array,
        default:''//默认值,
        requried:true,
        validator(val){
        //在这个函数中，我们需要返回一个验证的状态，true是成功 false是失败
        return true/false //根据你自己的规则
```

```
        }
    }
}
```

子父组件通信

非父子组件通信

作业：

一、整理今天的笔记

二、今天的案例最少敲4遍！！！！！

三、扩展： 实现子传父