第二天

项目首页

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

首页 - 通用轮播组件

轮播图组件需要在首页和分类页使用，封装成通用组件

1. 准备组件
2. 自动导入组件
3. 添加组件类型声明


```jsonc
// pages.json
{
  "easycom": {
    "autoscan": true,
    "custom": {
      // uni-ui 规则配置
      "^uni-(.*)": "@dcloudio/uni-ui/lib/uni-$1/uni-$1.vue",
      // 以 Xtx 开头的组件，在 components 目录中查找
      "^Xtx(.*)": "@/components/Xtx$1.vue"
    }
  }
}
```

温馨提示：修改 esaycom 需重启服务 pnpm dev:mp-weixin

```ts
// src/types/components.d.ts
import XtxSwiper from './XtxSwiper.vue'

declare module '@vue/runtime-core' {
  export interface GlobalComponents {
    XtxSwiper: typeof XtxSwiper
  }
}
```

Volar 官网

首页 - 轮播图指示点


```
<swiper @change="onChange"></swiper>
```

```
// UniHelper 为 uni-app 提供事件类型
const onChange: UniHelper.SwiperOnChange = (ev) => {
  // ! 非空断言主观上排除掉空值情况
  activeIndex.value = ev.detail!.current
}
```

```
<view class="indicator">
  <text
    v-for="(item, index) in 3"
    :key="item"
    class="dot"
    :class="{ active: index === activeIndex }"
  ></text>
</view>
```

知识点：

1. UniHelper 提供事件类型，文档说明
2. ? (可选链) 允许前面表达式为空值
3. ! (非空断言) 主观上排除掉空值情况

首页 - 获取轮播图数据

1. 封装获取轮播图数据API
2. 页面初始化调用API

```ts
// services/home.ts
export const getHomeBannerAPI = (distributionSite = 1) => {
  return http({
    method: 'GET',
    url: '/home/banner',
    data: {
      distributionSite,
    },
  })
}
```

```ts
// pages/index/index.vue
const getHomeBannerData = async () => {
  const res = await getHomeBannerAPI()
  console.log(res)
}

onLoad(() => {
  getHomeBannerData()
})
```

首页 - 轮播图数据类型并渲染

1. 定义轮播图数据类型
2. 指定类型并传值给子组件
3. 渲染轮播图数据

```ts
// types/home.d.ts
/** 首页-广告区域数据类型 */
export type BannerItem = {
  /** 跳转链接 */
  hrefUrl: string
  /** id */
  id: string
  /** 图片链接 */
  imgUrl: string
  /** 跳转类型 */
  type: number
}
```

```ts
// services/home.ts
export const getHomeBannerAPI = (distributionSite = 1) => {
  return http<BannerItem[]>({
    …省略
  })
}
```

父组件数据类型

```ts
const bannerList = ref<BannerItem[]>([])
```

```vue
<XtxSwiper :list="bannerList" />
```

子组件 props 类型

```ts
// src/components/XtxSwiper.vue
defineProps<{
  list: BannerItem[]
}>()
```

子组件轮播图渲染

```vue
<swiper-item v-for="item in list" :key="item.id">
  <navigator url="/pages/index/index" hover-class="none" class="navigator" >
    <image mode="aspectFill" class="image" :src="item.imgUrl" ></image>
  </navigator>
</swiper-item>
```

首页 - 轮播图总结

准备组件 → 自动导入组件 → easycom 配置 → 添加类型声明

↓

指示点高亮 → 事件类型 → UniHelper 提供

↓

获取数据 → 封装API → 初始化调用 → ref 数据类型

↓

数据渲染 → 父组件传值 → 子组件props类型 → 列表渲染

首页 - 前台分类组件

1. 准备组件 (只有首页使用)
2. 导入并使用组件
3. 设置首页底色为 `#F7F7F7`

`#F7F7F7`

小程序页面根标签是 page

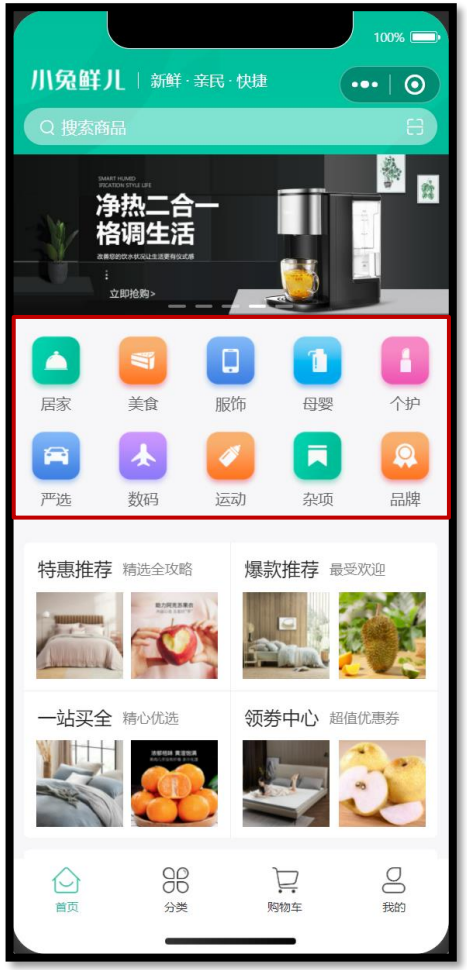
首页 - 获取前台分类数据

1. 封装获取前台分类数据API
2. 页面初始化调用API

```
// services/home.ts
export const getHomeCategoryAPI = () => {
  return http({
    method: 'GET',
    url: '/home/category/mutli',
  })
}
```

```
// pages/index/index.vue
const getHomeCategoryData = async () => {
  const res = await getHomeCategoryAPI()
  console.log(res)
}

onLoad(() => {
  getHomeCategoryData()
})
```



首页 - 前台分类数据类型并渲染

1. 定义前台分类数据类型
2. 指定类型并传值给子组件
3. 渲染前台分类数据

```ts
// types/home.d.ts
/** 首页-前台类目数据类型 */
export type CategoryItem = {
  /** 展示图标 */
  icon: string
  /** id */
  id: string
  /** 分类名称 */
  name: string
}
```

```ts
// services/home.ts
export const getHomeCategoryAPI = () => {
  return http<CategoryItem[]>({
    method: 'GET',
    url: '/home/category/mutli',
  })
}
```

父组件数据类型

```ts
const categoryList = ref<CategoryItem[]>([])
```

```vue
<CategoryPanel :list="categoryList" />
```

子组件 props 类型

```ts
// pages/index/components/CategoryPanel.vue
defineProps<{
  list: CategoryItem[]
}>()
```

子组件分类渲染

```vue
<view class="category">
  <navigator class="category-item" v-for="item in list" :key="item.id">
    <image class="icon" :src="item.icon" mode="aspectFit"></image>
    <text class="text">{{ item.name }}</text>
  </navigator>
</view>
```

首页 - 热门推荐组件

1. 准备组件(只有首页使用)
2. 导入并使用组件

首页 - 获取热门推荐数据

1. 封装获取热门推荐数据API
2. 页面初始化调用API

```ts
// services/home.ts
export const getHomeHotAPI = () => {
  return http({
    method: 'GET',
    url: '/home/hot/mutli',
  })
}
```

```ts
// pages/index/index.vue
const getHomeHotData = async () => {
  const res = await getHomeHotAPI()
  console.log(res)
}

onLoad(() => {
  getHomeHotData()
})
```

首页 - 热门推荐数据类型并渲染

1. 定义热门推荐数据类型
2. 指定类型并传值给子组件
3. 渲染热门推荐数据

```ts
// types/home.d.ts
/** 首页-热门推荐数据类型 */
export type HotItem = {
  alt: string
  id: string
  pictures: string[]
  target: string
  title: string
  type: string
}
```

```ts
// services/home.ts
/**
 * 首页-热门推荐-小程序
 */
export const getHomeHotAPI = () => {
  return http<HotItem[]>({
    method: 'GET',
    url: '/home/hot/mutli',
  })
}
```

父组件数据类型

```ts
const hotList = ref<HotItem[]>([])
```

```vue
<HotPanel :list="hotList" />
```

子组件 props 类型

```ts
// pages/index/components/HotPanel.vue
defineProps<{
  list: HotItem[]
}>()
```

子组件分类渲染

```vue
<view class="panel hot">
  <view class="item" v-for="item in list" :key="item.id">
    <view class="title">
      <text class="title-text">{{ item.title }}</text>
      <text class="title-desc">{{ item.alt }}</text>
    </view>
    <navigator hover-class="none" url="/pages/recommend/recommend" class="cards">
      <image v-for="src in item.pictures" :key="src" :src="src"></image>
    </navigator>
  </view>
</view>
```

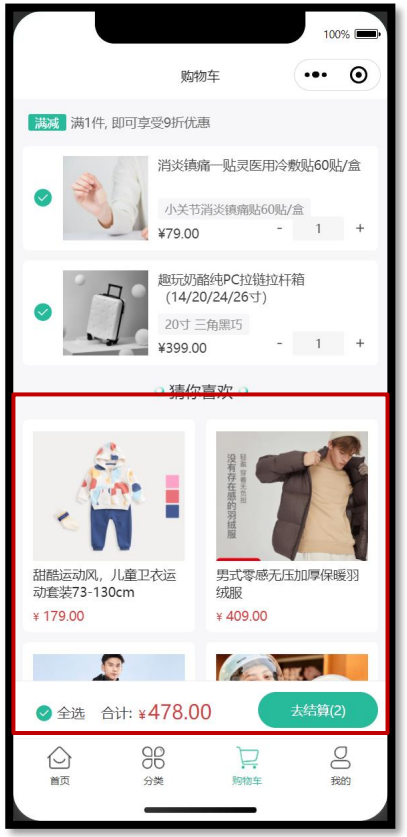
首页 - 猜你喜欢组件

1. 准备组件(通用组件)
2. 定义组件类型
3. 准备 scroll-view 滚动容器
4. 设置 page 和 scroll-view 样式

```ts
// types/components.d.ts
import XtxGuess from '@/components/XtxGuess.vue'
declare module '@vue/runtime-core' {
  export interface GlobalComponents {
    XtxGuess: typeof XtxGuess
  }
}
```

```vue
// pages/index/index.vue
<scroll-view scroll-y class="scroll-view">
  <!-- 自定义轮播图 -->
  <XtxSwiper :list="bannerList" />
  <!-- 分类面板 -->
  <CategoryPanel :list="categoryList" />
  <!-- 热门推荐 -->
  <HotPanel :list="hotList" />
  <!-- 猜你喜欢 -->
  <XtxGuess />
</scroll-view>
```

```scss
<style lang="scss">
page {
  background-color: #f7f7f7;
  height: 100%;
  display: flex;
  flex-direction: column;
}
.scroll-view {
  flex: 1;
}
</style>
```



首页 - 获取猜你喜欢数据

1. 封装获取猜你喜欢数据API
2. 组件挂载完毕调用API

```ts
// src/services/home.ts
/**
 * 猜你喜欢-小程序
 */
export const getHomeGoodsGuessLikeAPI = () => {
  return http({
    method: 'GET',
    url: '/home/goods/guessLike',
  })
}
```

```ts
// src/components/XtxGuess.vue
// 获取猜你喜欢列表数据
const getHomeGoodsGuessLikeData = async () => {
  const res = await getHomeGoodsGuessLikeAPI()
  console.log(res)
}

// 组件挂载完毕
onMounted(() => {
  getHomeGoodsGuessLikeData()
})
```

首页 - 猜你喜欢数据类型和列表渲染

数据类型


```ts
// types/global.d.ts
/** 通用分页结果类型 */
export type PageResult<T> = {
  /** 列表数据 */
  items: T[]
  /** 总条数 */
  counts: number
  /** 当前页数 */
  page: number
  /** 总页数 */
  pages: number
  /** 每页条数 */
  pageSize: number
}
```

```ts
// types/home.d.ts
/** 猜你喜欢-商品类型 */
export type GuessItem = {
  /** 商品描述 */
  desc: string
  /** 商品折扣 */
  discount: number
  /** id */
  id: string
  /** 商品名称 */
  name: string
  /** 商品已下单数量 */
  orderNum: number
  /** 商品图片 */
  picture: string
  /** 商品价格 */
  price: number
}
```

```
code: "1"
msg: "操作成功"
result
  counts: 345
  items: [{id: "3988335"
  page: 1
  pageSize: 10
  pages: 35
```

```ts
// services/home.ts
export const getHomeGoodsGuessLikeAPI = () => {
  return http<PageResult<GuessItem>>({
    method: 'GET',
    url: '/home/goods/guessLike',
  })
}
```

列表渲染

定义类型

v-for 循环

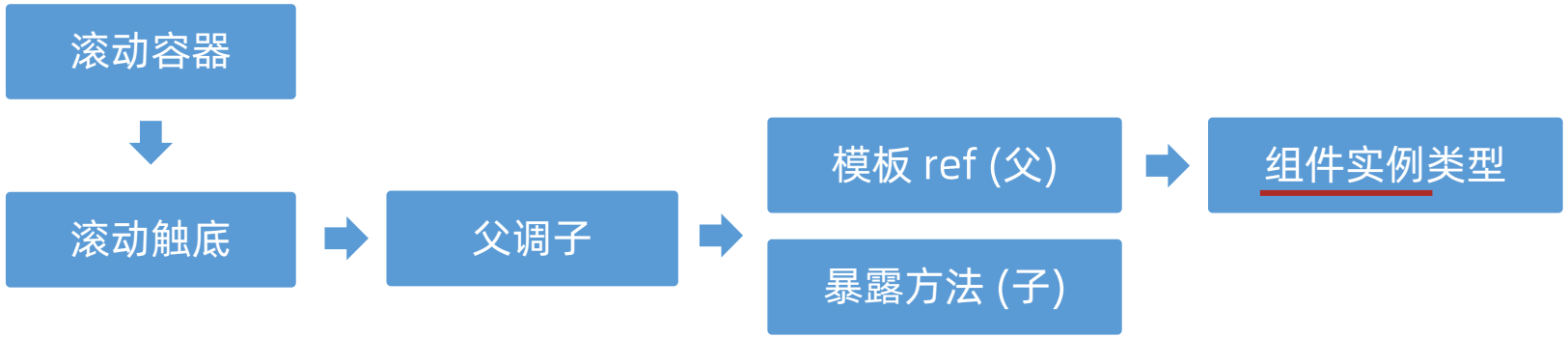
指定类型并保存数组

```ts
const guessList = ref<GuessItem[]>([])
const getHomeGoodsGuessLikeData = async () => {
  const res = await getHomeGoodsGuessLikeAPI()
  guessList.value = res.result.items
}
```

渲染猜你喜欢数据

```vue
<view class="guess">
  <navigator class="guess-item" v-for="item in guessList" :key="item.id">
    <image class="image" mode="aspectFill" :src="item.picture"></image>
    <view class="name">{{ item.name }}</view>
    <view class="price">
      <text class="small">¥</text>
      <text>{{ item.price }}</text>
    </view>
  </navigator>
</view>
```

首页 - 猜你喜欢分页准备



滚动触底 和 模板ref

```vue
// src/pages/index/index.vue
const guessRef = ref<XtxGuessInstance>()
const onScrolltolower = () => {
  guessRef.value?.getMore()
}


<!- 滚动容器 -->
<scroll-view @scrolltolower="onScrolltolower">
  <!-- …省略 -->
  <XtxGuess ref="guessRef" />
</scroll-view>
```

组件实例类型

```ts
// src/types/components.d.ts
export type XtxGuessInstance = InstanceType<typeof XtxGuess>
```

暴露方法

```vue
// src/components/XtxGuess.vue
defineExpose({
  getMore: getHomeGoodsGuessLikeData,
})
```

首页 - 猜你喜欢分页加载

升级 API 函数 → 添加分页参数 → 定义类型 → 调用 API 传参 → 数组追加 → 页码累加

滚动触底

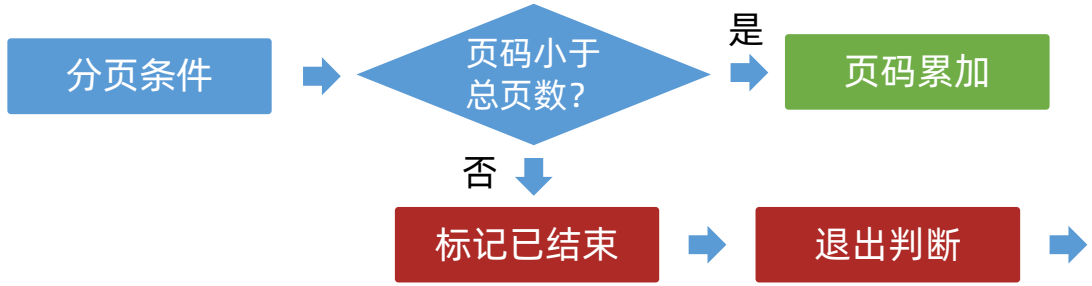
```ts
// src/services/home.ts
/** 猜你喜欢-小程序 */
export const getHomeGoodsGuessLikeAPI = (data?: PageParams) => {
  return http<PageResult<GuessItem>>({
    method: 'GET',
    url: '/home/goods/guessLike',
    data,
  })
}
```

```ts
// src/types/global.d.ts
/** 通用分页参数类型 */
export type PageParams = {
  /** 页码：默认值为 1 */
  page?: number
  /** 页大小：默认值为 10 */
  pageSize?: number
}
```

```ts
// src/components/XtxGuess.vue
// 分页参数
const pageParams: Required<PageParams> = {
  page: 1,
  pageSize: 10,
}

const getHomeGoodsGuessLikeData = async () => {
  const res = await getHomeGoodsGuessLikeAPI(pageParams)
  // 数组追加
  guessList.value.push(...res.result.items)
  // 页码累加
  pageParams.page++
}
```

首页 - 猜你喜欢分页条件

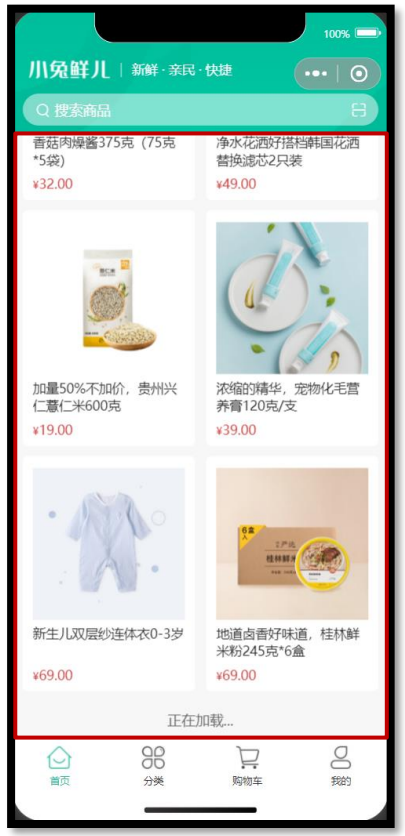



```vue
// src/components/XtxGuess.vue
// 是否已结束
const finish = ref(false)
const getHomeGoodsGuessLikeData = async () => {
  // 已结束退出函数并提示用户
  if (finish.value === true) {
    return uni.showToast({
      icon: 'none',
      title: '没有数据了~'
    })
  }
  const res = await getHomeGoodsGuessLikeAPI(pageParams)
  guessList.value.push(...res.result.items)
  // 当前页码是否小于总页数
  if (pageParams.page < res.result.pages) {
    // 页码累加
    pageParams.page++
  } else {
    // 标记为已结束
    finish.value = true
  }
}
```

```
code: "1"
msg: "操作成功"
result: {counts: 345, pag
  counts: 345
  items: [{id: "4007963",
  page: 1
  pageSize: 10
  pages: 35
```

```
code: "1"
msg: "操作成功"
result: {counts: 345,
  counts: 345
  items: []
  page: 36
  pageSize: 10
  pages: 35
```

```vue
<view class="loading-text">
  {{ finish ? '没有数据了~' : '正在加载...' }}
</view>
```



首页 - 下拉刷新

滚动容器 → 开启下拉刷新 → 监听事件 → 加载数据 → 关闭动画

```vue
// src/pages/index/index.vue
<!-- 滚动容器 -->
<scroll-view
  refresher-enabled
  @refresherrefresh="onRefresherrefresh"
  :refresher-triggered="isTriggered"
>
  …省略
</scroll-view>
```

```vue
// src/pages/index/index.vue
// 下拉刷新状态
const isTriggered = ref(false)
// 自定义下拉刷新被触发
const onRefresherrefresh = async () => {
  // 开启动画
  isTriggered.value = true
  // 加载数据
  await Promise.all([
    getHomeBannerData(),
    getHomeCategoryData(),
    getHomeHotData(),
  ])
  // 关闭动画
  isTriggered.value = false
}
```

首页 - 下拉刷新

滚动容器 → 开启下拉刷新 → 监听事件 → 加载数据 → 关闭动画

猪你喜欢组件 → 重置页码 / 重置列表 / 重置结束标记 → 父调子 → 重置数据 → 加载数据

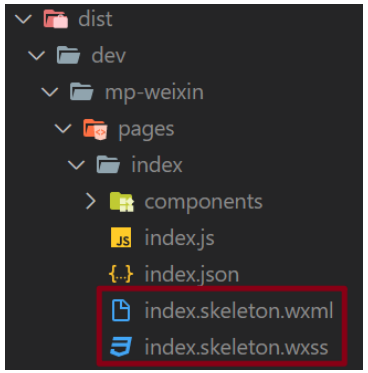
```
// src/components/XtxGuess.vue
// 重置数据
const resetData = () => {
  pageParams.page = 1
  guessList.value = []
  finish.value = false
}
// 暴露方法
defineExpose({
  resetData,
})
```

```
// src/pages/index/index.vue
// 自定义下拉刷新被触发
const onRefresherrefresh = async () => {
  // 重置数据
  guessRef.value?.resetData()
  // 加载数据
  await Promise.all([
    getHomeBannerData(),
    getHomeCategoryData(),
    getHomeHotData(),
    guessRef.value?.getMore(),
  ])
  // …省略
}
```

首页 - 骨架屏

微信开发者工具 → 生成骨架屏 → 转成vue组件 → 数据正在加载中？ 是 → 显示骨架屏

否 → 显示正常组件




```
// PageSkeleton.vue
<template>
  骨架屏结构
</template>

<style>
  骨架屏样式
</style>
```

```
// 加载中标记
const isLoading = ref(false)
onLoad(async () => {
  isLoading.value = true
  await Promise.all([
   getHomeBannerData(),
   getHomeCategoryData(),
   getHomeHotData()
  ])
  isLoading.value = false
})

<scroll-view>
  <PageSkeleton v-if="isLoading" />
  <template v-else>
    <XtxSwiper />
    <CategoryPanel />
    <HotPanel />
    <XtxGuess />
  </template>
</scroll-view>
```

骨架屏

正常展示

传智教育旗下高端IT教育品牌