1、脚手架-cli的工具介绍

- 大家好，我是来自软研运营平台开发部的朋学友，欢迎大家。加今天的培训，上一节，涂工介绍了移动应用开发平台，这一节，我们进入实战，真刀真枪地从零开始，开发出一款app应用。
- 这是我们今天的学习目标，通过这一节课，我们需要掌握hatom2-cli脚手架工具的使用、熟悉移动端H5开发调试技巧并且了解hatom2插件的使用，最后可以通过hatom2-cli工具开发出一款app来
- 今天我们将分四个章节进行讲解，第一章将介绍hatom2-cli脚手架工具的安装以及使用，第二章，将从移动端开发和PC端开发的区别以及工程组织方面，对应用开发中的一些要点进行说明，第三部分，将分别介绍在开发过程中如何在PC端和真机上进行调试，最后，将介绍并演示hatom插件的使用方法以及插件的调试方法。
- 4、首先，我们进入第一章节，hatom2-cli脚手架工具。
- 5、在这一章中，我们需要掌握脚手架工具的安装和使用，以及如何打包，并且在移动应用开发平台上进行云编译，生成一个android应用程序apk
- 这里贴出了hatom2-cli脚手架工具的文档地址，我们今天的讲解也将围绕这份文档进行说明，需要特别强调的是，脚手架的名称是hatom2-cli而不是hatom-cli,这个不要安装错误
- 就像vue官方的vue-cli脚手架工具，可以快速搭建起vue工程一样。hatom2-cli脚手架基于这一思想，也提供了一组灵活的交互命令，可以快速生成移动H5项目工程，从而避免babel、loader、plugin等繁琐的配置，使前端开发人员可以直接进行业务开发。其次，脚手架封装了jsbridge的一些核心代码，对开发者而言，可以直接调用硬件设备的原生能力，而不需要额外的配置。第三点，脚手架集成了cube-ui、Vant这样的UI组件库，只需要通过简单的交互命令进行选择，即可实现开箱即用，同时，也可以选择自定义模式引入你熟悉的UI组件，使用起来非常灵活，后续我们也会根据大家的反馈，把更多的UI库集成进来。最后，一点建议，使用lint工具进行代码开发是一个不错的习惯，脚手架提供了standard和hatom两种风格的代码规范，当然，你也可以配置自己熟悉的lint规范或者干脆不使用lint工具。
- 脚手架工具需要nodejs版本高于10.12.0以及npm版本高于6.4.1，如果觉得更换node版本麻烦，强烈推荐使用nvm 这样的node版本管理工具。脚手架安装非常简单，通过npm -g install 进行全局安装或者npx install进行非全局安装。
- 这里列出了脚手架提供的一些命令，都是标配，大家简单了解一下即可
- 模板方面，目前我们暂时提供了spa-tpl和mpa-tpl这两个模板以及spa-demo和mpa-demo这两个示例工程，spa-tpl是基于最新的webpack5和vue的一个单页工程，mpa-tpl是基于webapck5+vue2的hatom多页应用模板。大家可以用这个模板快速搭建起相对应的项目工程。我们也将这两个模板以及两个demo在公司仓库上进行了发布，gitee访问限制的同事只需要下载或更新hatom2-cli至2.0.0版本以上即可以顺利下载模板.
- 现在，我们将演示使用脚手架工具开发一个hello app的整个过程。
- 这里，列举了我们hello工程的目录，每一部分都进行了详细的注释。
- 通过run build命令进行打包，把zip包上传到平台上进行编译，具体过程我稍后进行演示过。以上，第一张节的内容我们就介绍到这里。

2、第二章，我们将介绍，应用开发中一些 要点的介绍。

- 这里是在这一章中的主要脉络，我将从PC端开发和移动端开发的特异性以及工程组织方面进行介绍。特异性方面，主要介绍页面缓存和路由动画这两个方面。
- 首先，大家可以想一下为什么要进行页面缓存呢。这里就涉及到了PC端开发和移动端开发的不同。在PC端的开发中，分页功能通常是采用页码的形式实现的，而在移动端，我们很少看到采用页码实现的分页，大多采用上拉加载、下拉刷新的形式实现。当我们点击列表进入详情页面后再返回到列表页面时，如果不进行列表页面的缓存，就需要重新请求列表数据，这在流量需要付费的情况下一般是不能接受的，而另一方面，当重新请求列表数据后，无论我们上一次查看到了哪一条列表，列表都会回到了第一条的位置，当数据量大的时候我们基本上找不到上一次浏览的位置，试想一下，如果我们在刷头条的时候，如果每查看一条新闻，就必须重头刷起，这种用户体验你们是否接受吗？所以呢，在移动端的开发过程中需要对部分页面进行缓存。

- 在脚手架工具和demo里，我们采用vue的内置组件keep-alive实现页面的缓存,这种方式基本可以满足全部场景。当然，也有很多其他的方案，例如使用vue-page-stack这类第三方组件库，但是这类插件往往存在这样或那样的不足，是否满足你的要求就需要自行评估了，这里就不再展开讨论了。下面呢，我们介绍hatom用keep-alive实现页面缓存的基本原理。
- 首先，在标签外，包裹组件，keep-alive 有三个属性，include、exclude和max，从字面意思很容易理解这三个属性所代表的含义。这里我们用到了include、exclude这两个属性，通过路由的meta属性，动态的去控制哪些页面需要缓存，哪些页面不需要缓存，
- 在watch中，我们观察route的变化，实时更新include和exclude的路由列表。
- 在全局路由守卫中，我们通过query属性全局性更新页面meta的cache属性，动态地控制一个页面的缓存与否。这里为什么不直接改变meta而用query呢，大家可以思考一下。
- 这时，我们已经进行了页面缓存，但还需要记录上一次页面离开时的位置。实现起来也非常简单，在页面离开时，我们调用beforeRouteLeave这个生命周期函数，将滚动元素当前滚动的高度保存在scrollTop这个变量中。再次进入这个页面后，我们在activated生命周期函数中，把scrollTop变量中保存的滚动高度信息赋值给滚动元素。其中activated生命周期函数会在keep-alive缓存的页面之间切换时对应执行。
- 最后，我们只需在路由跳转时，通过query的cache属性，设置页面是否需要被缓存就行了。

- 除了页面缓存，PC端开发和移动端开发另一点不同，就是在路由跳转的时候需要添加一些过渡动画，这样可以很好的提高用户体验。我们也在脚手架中进行了一些简单的封装，实现了路由跳转的过渡效果，如这个gif图中的形式。当页面由浅层跳转到深层时，页面从右向左过渡。反之，从左往右过渡。
- 配置这个动画很简单，只需要三步。
- 首先在路由信息的meta属性中，根据实际业务，配置每一个页面的深度deep，
- 第二步，在全局路由守卫中根据原始页面和目标页面deep大小，动态设置全局变量$direction，从而配置动画的过渡方向，其中Vue.prototype原型可以设置Vue的全局变量
- 最后一步，在标签外包裹动画组件，并且配置name属性为$direction这个全局变量就可以了。顺便提一下，动画样式可以在animation.css文件中进行修改。
- 第一部PC端开发和移动端开发的不同点介绍完了，下面，我们开始工程组织方面的介绍。
- 采用合理、相同的工程组织，不仅有助于提高代码质量，而且便于合作开发、提高协同效率，最重要的一点，当你在开发中遇到了问题，可以方便我们及时进行问题的排查与定位。接下来，我将一一介绍工程组织方面的要点。
- 在我们的脚手架工程里，公共组件是一个很重要的部分。可以将一些通用的组件进行全局注册，从而方便在任何vue组件中直接使用，而不用再在每个文件中import导入以及在components中进行注册。
- 这里是一个tabbar的公共组件的目录结构，命名采用kebab-case（短横线命名）的形式进行命名，整个目录结构借鉴了element的组织方式，业务代码在i-tabbar.vue中，index.js中提供了install注册方法，下面我们来看这些文件。
- 在i-tabbar.vue 中，强烈建议更节点class直接采用组件名进行命名，而name属性一定要是ITabbar这种双驼峰形式，否则组件注册会失败
- 组件的index.js文件中暴露了install方法，最后在components的index.js中进行了统一注册，在任意的vue组件中的标签就可以直接使用组件了。
- 全局方法和全局的正则表达式，也是使用频率很高的部分。有序的组织不仅可以使代码更加整洁，而且使用起来也会很方便。在util.js 和 reg.js 中进行全局方法和全局正则表达式的添加，在各个vue组件中通过$+方法名 和$+正则表达式名的方式使用，如this.$isMobile()方法和this.$EMAIL_TEST邮箱合法性校验。

- 过滤器也是开发中经常用到的，可以在filter.js文件中添加一些全局性过滤器，这样在每个vue组件中都可以使用。例如，脚手架工程使用dayjs这个第三方组件，实现了一个timeFormat时间格式转换多的过滤器， 它将时间对象转换为YYYY-MM-DD 这样一个日期时间格式。
- 紧接着，将工程中常用到布局、进行全局配置可以有效地提高代码复用效率，这类布局可以集中放置在layouts这个目录下，脚手架中提供了两个布局，如layout1，它是一个包含头部、内容、以及底部四个模块切换的一个常见的布局。在路由的配置中，通过子路由的形式复用layouts下面的布局。
- 在移动应用开发中，将一些小图标制作成一张雪碧图，在某种情况下，是一个减少http资源请求数是一个比较好的主意。但是对于不常使用photoshop这类图像处理软件的同事，经常要麻烦设计同事帮助实现雪碧图的制作，如果遇到修改的情况，又需要重新制作雪碧图，效率很低。这里就是hatom2-cli脚手架工具的一个特色了，可以将这类图标都放置于src/assets/icons这个目录下，脚手架会自动把icons目录下的所有图标生成一张雪碧图，在使用的时候只需要通过*标签直接引用就可以了。举个例子，假如我将arrow-left.png这样一个图标放到icons目录下，然后在使用的时候将标签的class改为icon-arrow-left就可以了，别的什么都不需要做。*
- *工程配置都在config目录下，*
- *api.js文件是我们接口地址，可以在该文件里面配置后端服务的请求地址，如配置一个BAIDU的接口地址，在vue组件中就可以使用this.$BAIDU进行引*
- *const.js文件中是一些常量的配置，如配置页面高度SHEIGHT，在任何vue组件中可以就可以通过ths.$SHEIGHT进行引用*
- *http.js文件是对axios的一些封装，可以在request.interceptors.request和response中进行一些鉴权、添加头部信息这些业务配置，在使用时，通过调用this.$http就可以引用axios 暴露的实例*
- *webAPP.json 是app的相关参数配置，具体配置参数如下，*
- *需要重点说明的一点是，vue支持hash和history两种模式的路由配置，但基于某些原因，hatom只能支持hash模式的路由配置，否则就有bug了呗。*
- *最后说一下成果物，成果物的目录结构如图所示。脚手架在运行npm run build 命令后，会生成一个webapp+h5packCode的zip文件。其中，webapp的前缀是一定不能少的，而且文件名中不能出现中文。这里，我们不强迫使用脚手架工具进行开发，也可以使用自己的方式来，只需要保证最终的成果物目录结构一致就可以了*

3、下面，我们开始第三章 应用调试方面的介绍

在这一章节里，简单概括，我们需要掌握在PC端谷歌浏览器中以及在真机环境下如何进行移动端的开发调试。

- *PC调试非常简单，对于一些页面布局的简单调试，在浏览器中运行我们的工程，切换至device toolbar模式，相信大家基本上都会。这里不做太多的介绍。*
- *这里主要介绍的是，我们平台中推出的的调试工具，使用调试工具，首先第一步，登录移动应用开发平台，在工具下载模块中，下载远程调试工具，并安装到你的设备中，这里我们我们用模拟器演示一下。*
- *第一步启动项目的调试模式，npm run dev*
- *接下来，在工程中的入口文件main.js中去掉_this.$openConsole()方法注释，*
- *紧接着，将设备连接到跟PC端同一网域下，打开我们的调试工具，在地址栏输入运行地址和端口号，并点击确定按钮。*
- *这时，就看到我们的工程在手机上已经正常运行了，如左图所示。对代码的任何修改都会实时更新，如果没有更新的话，退出app再次进入即可。在这里，屏幕的右下角多了一个vConsole的按钮，点击这个按钮，就会显示控制台面板，如右图所示，面板使用跟PC端相同，在控制台上，我们可以console一些关键信息，或者添加断点进行代码的调试。*
- *以上，就是第二张关于调试方面的主要内容，不知道有没有帮助解决移动应用开发过程中调试的问题，如果仍有疑问，可以在课后的答疑过程中进行提问，或者在hiklink群中与我们沟通。*

4、hatom插件的调用

- *这个版本中，我们提供了相机、二维码扫描、GPS定位等11个插件，大家可以也可以把自己的需求反馈给我们，促使我们不断完善hatom的插件集。*
- *这里是二维码扫描插件的一个调用示例。*
- *这里我来实际的演示一下插件的调试*
- *脚手架除了npm run dev启动命令外，还有调试模式的启动， npm run debug*
- *在浏览器中，打开我们的调试地址，在地址栏输入运行地址和端口号，并点击确定按钮*
- *在设备上安装调试工具，并与PC连接至同一网域，打开工具app*