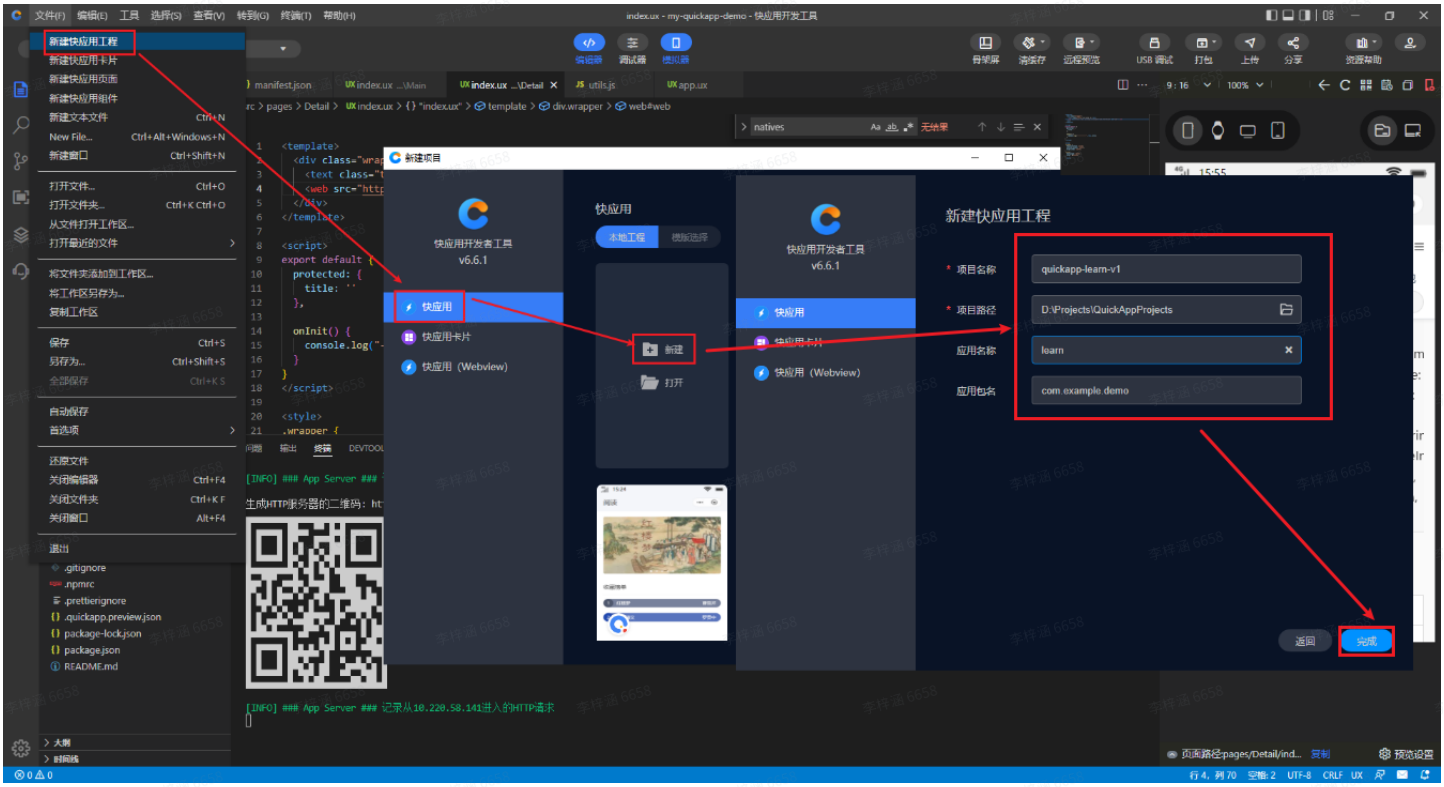
快应用项目搭建与开发

开发前准备

创建并初始化项目

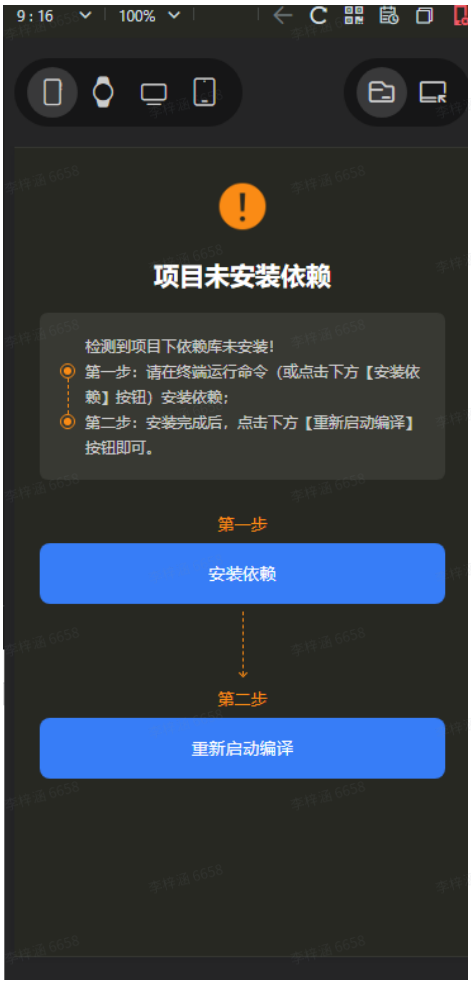
直接利用 `快应用开发工具` 创建新项目



按照图示进行：

1. 安装依赖
2. 重新启动编译

即可运行创建的快应用项目



命令行创建并初始化项目

a. 安装NodeJS环境

b. 安装hap-toolkit

hap-toolkit是快应用开发者工具，类似于vue-cli这类标准工具，可以直接创建模板工程。

```
npm install -g hap-toolkit
# 查看版本号(是否安装成功)
hap -v
```

c. 创建项目

```
hap init quickapp-learn-v1
```

```
├──src
    |   app.ux          APP 文件，可引入公共脚本，暴露公共数据和方法等
```

```
3       |  manifest.json  项目配置文件，配置应用图标、页面路由等
4       ├─assets          存放静态资源的文件夹
5       ├─helper
6       |   |  utils.js    封装常用工具类
7       |   └─apis         存放封装后的api的文件夹
8       ├─pages
9       |   ├─Demo
10      |       └─index.ux  页面文件，可自定义页面名称
```

d. 安装依赖 --- `npm install`

e. 打包项目 --- `npm run build`

f. 扫码安装并预览

   i. 安装手机调试器快应用调试器

   ii. 本地启动sever服务 --- `npm run server` --- 扫码预览即可

   iii. 可以打开出现的网址，进行web端预览或者扫码安装

开发初体验

List组件

使用 list

list组件可以替代div组件实现长列表布局，Native会复用type属性相同的list-item。

注意：type是必选属性。

```
<template>
  <div class="wrapper">
    <text class="title">{{ title }}</text>
    <list>
      <list-item type="review" for="{{item in list}}">
        <text>{{ item.name }}</text>
      </list-item>
      <list-item type="loadMore">
        <progress type="circular"></progress>
        <text>加载更多</text>
      </list-item>
    </list>
  </div>
</template>
```

暴露自定义的工具类 -- 以包装原生fetch为例

在 `app.ux` 中将所有公共方法注入全局 global中，方便页面直接调用 使用 async

1. 包装并暴露myFetch工具

```
/* helper/myFetch/index.js */
import nativeFetch from '@system.fetch'

const myFetch = {
    /**
     * 网络请求
     * @parmas options
     * @return {Promise}
     */
    async fetch(options) {
        const p1 = new Promise((resolve, reject) => {
            options.success = function (data, code) {
                resolve({ data, code });
            }
            options.fail = function (data, code) {
                reject({ data, code });
            }
            nativeFetch.fetch(options);
        })
        return p1;
```

```
    }
}

export default { myFetch }
```

2. 导入并挂载myFetch工具

```
<!-- app.ux -->
<script>
const $myFetch = require('./helper/sever/index').default

/* @desc: 注入方法至全局 global,以便页面调用 */
const hook2global = global.__proto__ || global
hook2global.$myFetch = $myFetch

export default {
  onCreate() { }
}
</script>
```

3. 使用myFetch工具

```
<!-- pages/Main/index.ux -->
<script>
  onInit() { },
  async onReady() {
    const ret1 = await $myFetch.myFetch.fetch({
      url: 'https://statres.quickapp.cn/quickapp/quickapptool/release/platform/q
    })
    console.info('fetch成功结果: ', JSON.stringify(ret1))
  },
}
</script>
```

路由跳转

路由跳转利用 `@system.router` 接口，传入url和param组成的对象 页面切换

1. 在 `manifest.json` 中注册页面
2. 触发页面跳转，并传递参数

```
<script>
import router from '@system.router'
export default {
  goDetail(title) {
    router.push({
      uri: 'pages/Detail',
      params: { title }
    })
  },
}
</script>
```

3. 接受参数

接收参数直接在页面的ViewModel的protected属性中声明属性，属性名和传入的属性名相同即可

注意：private等属性中不能声明相同名称的属性

```
<script>
export default {
  protected: {
    title: ''
  },
  onInit() {
    console.log("-----------", this.title)
  }
}
</script>
```

web标签的使用

web标签能直接显示在线的html页面，主要属性为id和src web

```
<web src="https://doc.quickapp.cn/widgets/web.html?h=web" id="web"></web>
```