一、库、框架的选型（~/package.json）

1.构建、打包工具Webpack 4.x  
sass-loader/babel-loader/file-loader等loaders，web-webpack-plugin/mini-css-extract-plugin等plugins，建议模块化的代码编写、组织方式，对打包后的代码做些压缩、分割、按需加载等优化，webpack本身的优化特性可以活用，社区、github上有大量插件可选。  
2.业务使用到的第三方库、插件等  
axios/lodash/lottie-web/pdfjslib/snapsvg等

二、业务流程（前端）

入口~/js/index.js

1.多页面:课程列表页、登录页（学生听课需要登录）、上课页面（学生、旁观者）  
2.上课，服务器端推送的视频、音频流渲染到浏览器的vedio标签里,同时服务器端会推送房间的消息（包括同步），具体如下：

2.1 ~/js/jssdk/rtc\_gate.js,该对象会负责从进房（包括中途进入）、离开房间这整个过程的消息同步（~\json\GLOB.json）

2.2 进入房间后的消息同步，分3大块：课件消息、KV消息、笔迹消息

2.2.1 课件有几种: 白板、PDF、word、PPT、图像、文字等，接收到消息后，一般处理过程如下:收集课件同步消息（哪一页、堆叠顺序等），同步消息完成后的时机，渲染对应的课件，各种课件类型会分别去渲染。

2.2.2 KV消息则负责课件、笔迹等等资源的删除、顺序改变等等

2.2.3 笔迹消息（包括橡皮擦）单独处理，这里选用的技术方案是以snapsvg库为基础绘制的，笔迹的细节绘制处理是根据客户体验需求为基础的，笔迹的优化点则在于量大、以及课件的当前渲染区域上

2.3 媒体消息：视频流、音频流，上下台、开闭麦等，都是收到消息以后，对页面的用户控件进行响应处理的，前端主要负责的就是用户控件的渲染、更新等处理维护，偶尔会有一些动画的制作，也都是属于前端er的用武之地了

3.上课的互动小游戏、奖励，包括最新PC端有的如：掷骰子、答题卡、点赞排行榜、计时器、屏幕共享等，前端负责渲染展示交互，包括一些动画的处理。

三、与后端的交互  
后端负责消息等协议的制定、产出，页面需求都是跟着后端走的，所以相关需求应着手做之前确定

四、不足、优化、扩展

现有项目在发布打包规范化、测试流程、重构、代码风格上面都有很大的发展潜力，这些对于项目的成长、维护是有用的。