title: 直播中心-代码评审

组件设计

React组件层级

PC端

```
App
├── Player (可优化，未抽成单独的组件， 包含videojs和直播的的初始化逻辑)
│ ├── videojs
│ ├── Poster(直播未开始、暂停、结束的海报)
│ ├── FlashCheck(暂时弃用, flv)
├── TabsPage (IM 初始化, 注册IM的事件监听)
│ ├── Tabs (列表渲染,避免组件的重新渲染;)
│ │ ├── ChatPage (聊天页面)
│ │ │ ├── CloseChatModal
│ │ ├── LiveInfo (直播简介)
│ │ ├── Slides(Enow, iFrame嵌入)
│ ├── Toast
├── OperationBar
├── Share
```

Q: 海报组件为什么只有异常的组件做成videojs组件，不全部放在videojs组件里做？ - 因为代码的加载顺序,根据接口返回的liveStatus, 如果异常，跳过player初始化的逻辑，直接显示海报

移动端

```
App
├── Player (可优化，未抽成单独的组件， 包含videojs和直播的的初始化逻辑)
│ ├── videojs
│ ├── Poster(直播未开始、暂停、结束的海报)
├── ContentArea(IM 初始化, 注册IM的事件监听)
│ ├── LiveInfo
│ ├── SlidesTitle
│ ├── TabsContainer (与PC端不同, tabs没有列表渲染)
│ │ ├── Slides
│ │ ├── Tabs
│ │ │ ├── ChatPanel
│ │ │ │ ├── CloseChatModal
│ │ │ ├── LiveIntro
│ ├── Toast
```

Videojs 组件

```
- ControlBar
- Modal(切流的弹窗)
- Poster/Exception(直播异常的海报)
- WaitingTip (卡顿逻辑的文案和卡顿监听/上报)
- WaterMark
```

卡顿判断

```
```javascript
/*
    直播加载时也会触发waiting事件，只有直播开始后的waiting才算做卡顿
*/
player.on('waiting', () => {
  if (self.played && !self.startTime) {
    self.startTime = Date.now()
  }
})
player.on('playing', () => {
  self.played = true
  if (self.startTime) {
    self.endTime = Date.now()
    const diff = self.endTime - self.startTime
    self.startTime = null
    self.FeMonitor.pushPerformance('waiting_time', diff)
  }

  if (self.timer) {
    clearTimeout(self.timer)
    self.timer = null
  }
})
```
```

flv插件

flv播放过程

1. videojs实例化
2. player.src()方法
    1. 选择合适的Tech
    2. Tech实例化

本质: 将flv.js作为video的数据处理方(sourceHandler)

必要接口

- isSupported()

- isSupported()
- canPlayType()
- setSrc() 将flv.js实例化