### 工作的内容业务描述：

所在的项目组是“人员卡口项目组”，整个的业务流程大体为：通过抓拍设备抓拍人员录入到后台（**抓拍模块**）；用户可以往设备中布防一些人员库（**人员库模块、布防模块**），设备会根据“比对算法”对抓拍人员和布防库中的人员进行比对，比对成功，页面就会产生告警（**告警模块**）。

抓拍模块包括：实时抓拍、人员抓拍查询

人员库模块包括：人员库查询、人员入库（库的增删改查、单张入库、批量入库、入库申请、入库审核、库导入、库导出）

布防模块包括：布防名单库查询、布防任务申请，布防任务审核

告警模块包括：实时告警、告警研判、告警查询；无需审核的告警所有用户都能接收到告警推送；需审核的告警需要告警处理完成后推送的对应用户能接收到。

我负责的模块是：人员入库页面、人员库查询页面、实时浏览页面（实时抓拍及实时告警）

### 第一部分：web版

1. 人员卡口-查询检索-人员库查询

功能：人员库的数据查询显示

1. 人员卡口-查询检索-标记人员查询

功能：标记员库的数据查询显示（重点人员标记、嫌疑人员标记、关注人员标记）

1. 人员卡口-人员入库-人员库

功能：库的增删改查、单张入库、库导入、库导出

4）公共控件：

3.1)Object-fit:contain



3.2）人员详情弹框的封装

3.3) 查询结果样式的定义

### 第二部分：qt版

1. 人员卡口-实时浏览-实时浏览

功能：展示实时抓拍人员和实时告警人员

1. 人员卡口-实时浏览-历史任务浏览

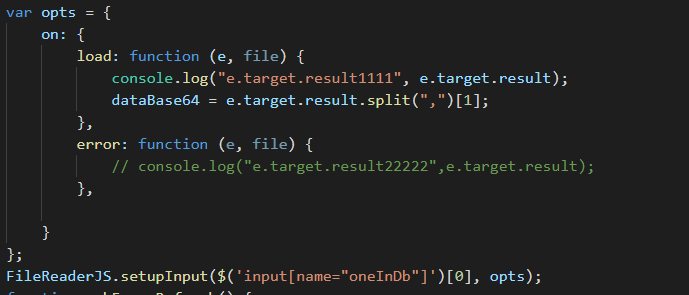
功能：展示由历史任务产生的抓拍和告警

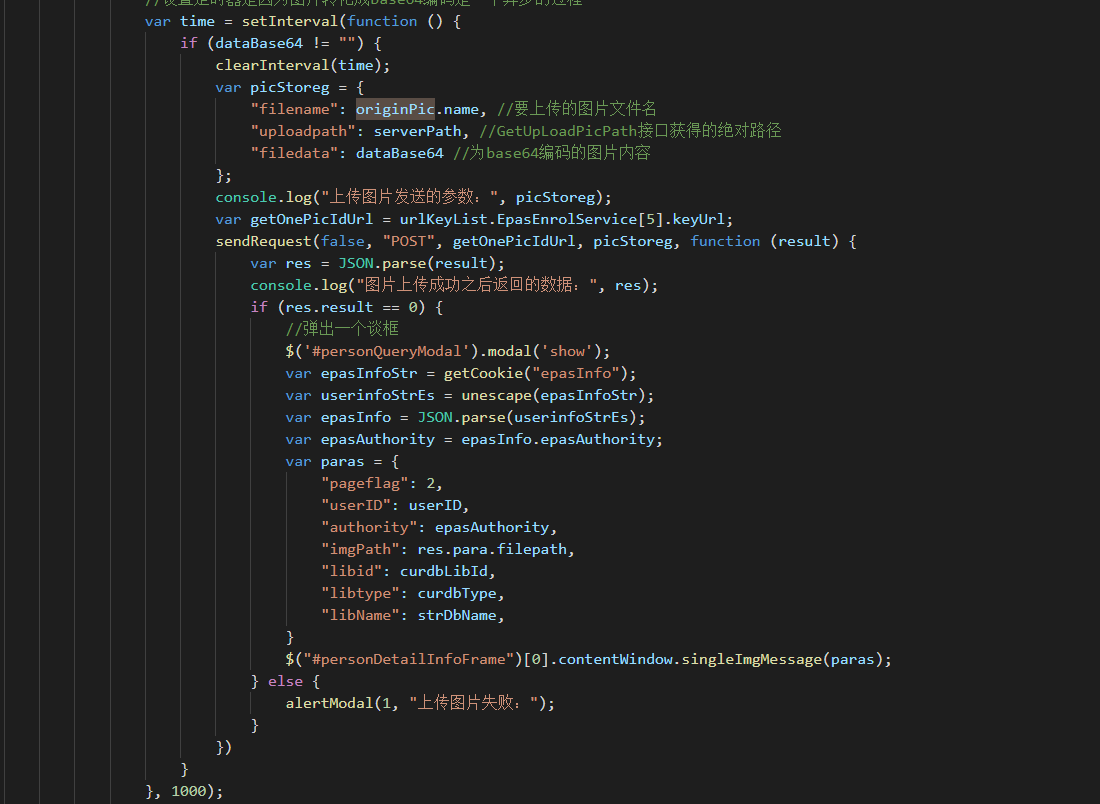
1. 人员卡口-人员入库-人员库：新增入库申请
2. 实时抓拍和实时告警组件封装

### 期间遇到的问题：

**1.图片上传的转码问题**

1. 业务场景：前端需要通过type = “file”的input框上传一张或多张图片给后台，后台接口接受参数为base64格式的数据，通过原始的文件转码，总是会出现转码不及时，或有些图片转码不成功的概率问题
2. 解决方法：运用了一个叫filereader.js的第三方库，它实现了一个转码成功和失败回调，即页面能收到转码成功或失败的回调信息，并且不会出现上面转码不成功或不及时的概率问题，然后我们在回调函数里拿到图片的转码信息
3. 代码实现：



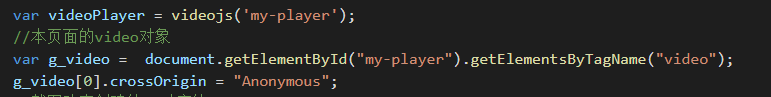


**2.video.js中视频截图的canvas跨域问题**

1）项目中是运用video.js做视频相关的一些功能，video.js提供的对外API能是实现大部分常用的功能（eg：暂停、播放、倍速、前进/后退、全屏等），但需要自己实现一个截图功能。

2）代码实现：





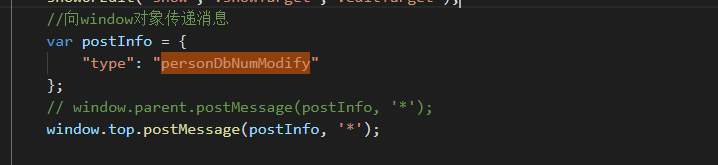
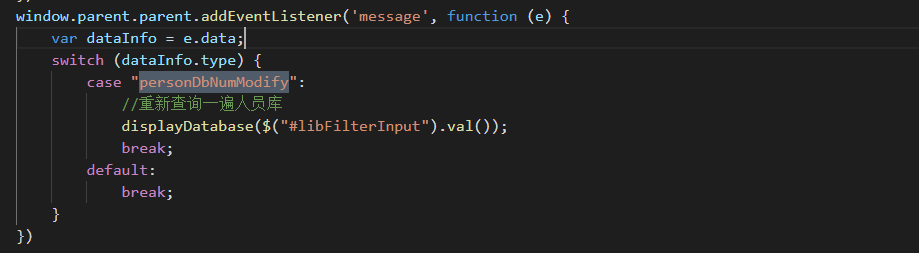
3）后端的视频源地址要设置允许跨域，前端设置crossOrigin且需要在videoIS实例化后立即设置，而不要在截图函数里面加，不然会出现由于操作DOM的时间差导致还是跨域失败

**3.在有多层ifream嵌套页面的情况下，怎么实现多个页面的信息传递（发布、订阅者模式）**

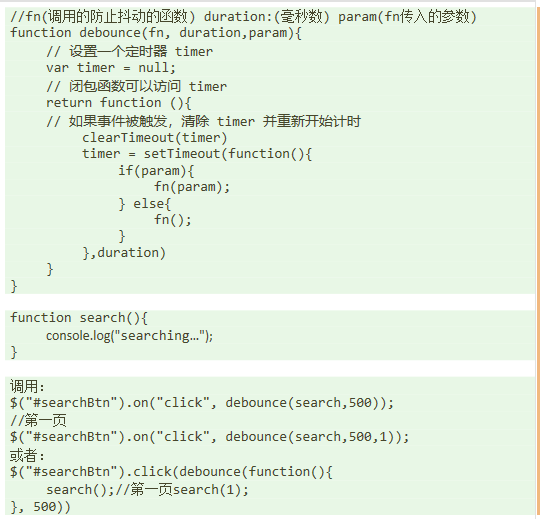
1）业务场景：多个页面都有人员标记的功能，在不同的页面进行人员标记后，标记的变更状态都要统计到一个“标记人员查询页面”，此时就需要在接受到标记变更信号的时候就进行页面刷新，而不是用户主动刷新

2）代码实现：

2,.1）

 2,.2) 

**4.多次点击防抖动，避免频繁请求**



**5.Ifream父子之间的数据传递**

1）父页面调用子页面的函数/父页面给子页面的变量赋值：

$("#personDetailInfoFrame")[0].contentWindow.markMessage(paras);

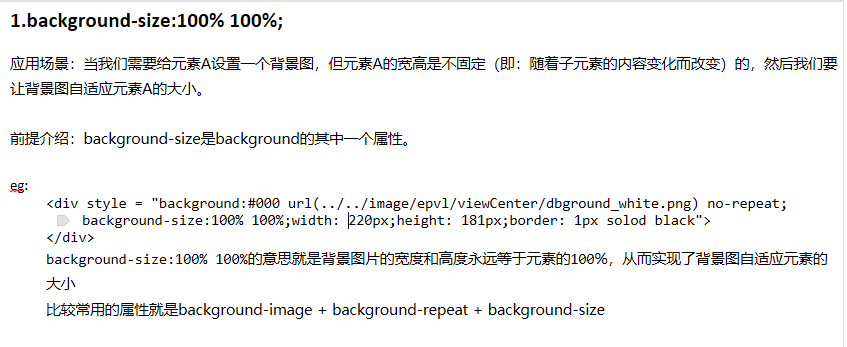
$("#personDetailInfoFrame")[0]..hellobaby="dsafdsafsdafsdafsdafsdafsadfsadfsdafsadfdsaffdsaaaaaaaaaaaaa";

2）子页面调用父页面的值：

 parent.ggMM();  parent.parent.ggMM();

**7.layui框架**

**8.背景图相关的操作**



**9**

1、问题的由来 ： <http://www.ruanyifeng.com/blog/2010/02/url_encoding.html>

2、网络标准[RFC 1738](http://www.ietf.org/rfc/rfc1738.txt" \t "_blank)做了硬性规定： 只有字母和数字[0-9a-zA-Z]、一些特殊符号"$-\_.+!\*'(),"[不包括双引号]、以及某些保留字，才可以不经过编码直接用于URL。

　  基于这个标准，所以浏览器必须对规定以外的字符进行编码。问题是不同的浏览器编码的规范是不一样的，后台解析的不可能把所有情况的考虑进去。

　  所以需要通过js 进行统一的编码，这样使得 url 符合 标准了，浏览器就没有插手 编码的就会了。

3、js 编码方法：

　  a、escape：实际上，escape()不能直接用于URL编码，它的真正作用是返回一个字符的Unicode编码值。比如"春节"的返回结果是 %u6625%u8282 。(已经不提倡使用了)

　  b、encodeURI：是 真正用来对URL编码的函数。因此除了常见的符号以外，对其他一些在网址中有特殊含义的符号 "; / ? : @ & = + $ , #"，也不进行编码。

　　　如果 encodeURI 不编码的部分（如 = 号）放在参数中，就会打乱 url 的 解析的结果。所以需要 encodeURIComponent 把参数中encodeURI 不编码的部分编码掉。

　  c、encodeURIComponent： 它用于对URL的组成部分进行个别编码，而不用于对整个URL进行编码。

　　　 "; / ? : @ & = + $ , #"，这些在encodeURI()中不被编码的符号，在encodeURIComponent()中统统会被编码。

<https://www.cnblogs.com/lvmylife/p/7595036.html> JS中对URL进行转码与解码

### 同步异步问题

### 缓存问题