# 前端监控基础知识

## 一、为什么要有前端监控?

- 用户体验直接决定用户留存，众多的体验问题（页面加载慢，页面白屏，操作卡顿，响应慢，请求报错，排版错乱等）单靠用户反馈并不现实
- 移动互联网时代，用户要求越来越高，耐心越差，对页响应慢，页面不可用容忍度低
- 使用场景复杂，h5、app、web、浏览器版本，手机机型，用户网速等等，都可能对业务带来影响

## 二、前端监控能来什么 ？

### 提升稳定性，更快地发现异常，定位异常，解决异常

- js错误
- 接口错误
- 资源错误
- 白屏
- ....

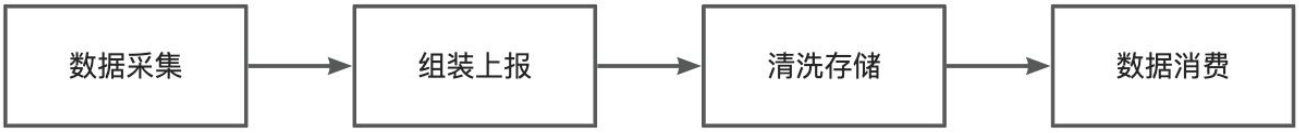
### 提升用户体验，长期关注性能优化

- 页面性能
- 接口性能
- 资源加载性能
- 卡顿监控

### 了解业务数据，指导产品升级

- PV && UV
- 业务数据
- 行为监控

## 三、前端监控的流程




### 数据采集

- pv监控：页面切换后新的URL、页面切换的原因
- js错误：错误对应的类型，描述，行列号、堆栈，错误发生前的用户交互，错误的上下文等
- 请求监控：请求路径、状态码、请求头、响应头、请求各阶段耗时等
- 性能监控：首屏加载各阶段耗时、各性能指标、SPA切换耗时等等
- 白屏监控：白屏发生的页面、关联的异常、相关上下文等等
- 静态资源监控 && 用户行为监控 && 自定义监控....

### 组装上报

- 基础数据包装：页面路径、页面标识、部署版本、部署环境、网络等等
- 采样逻辑
- 队伍暂存
- 上报方式：http，透明gif图片，sendBeacon(sendBeacon：在页面内关闭时发送请求，不阻塞页面卸载)
- 上报时机： requestIdleCallback/setTimeout 延时上报、beforeunload 回调函数内、缓存达到限定量

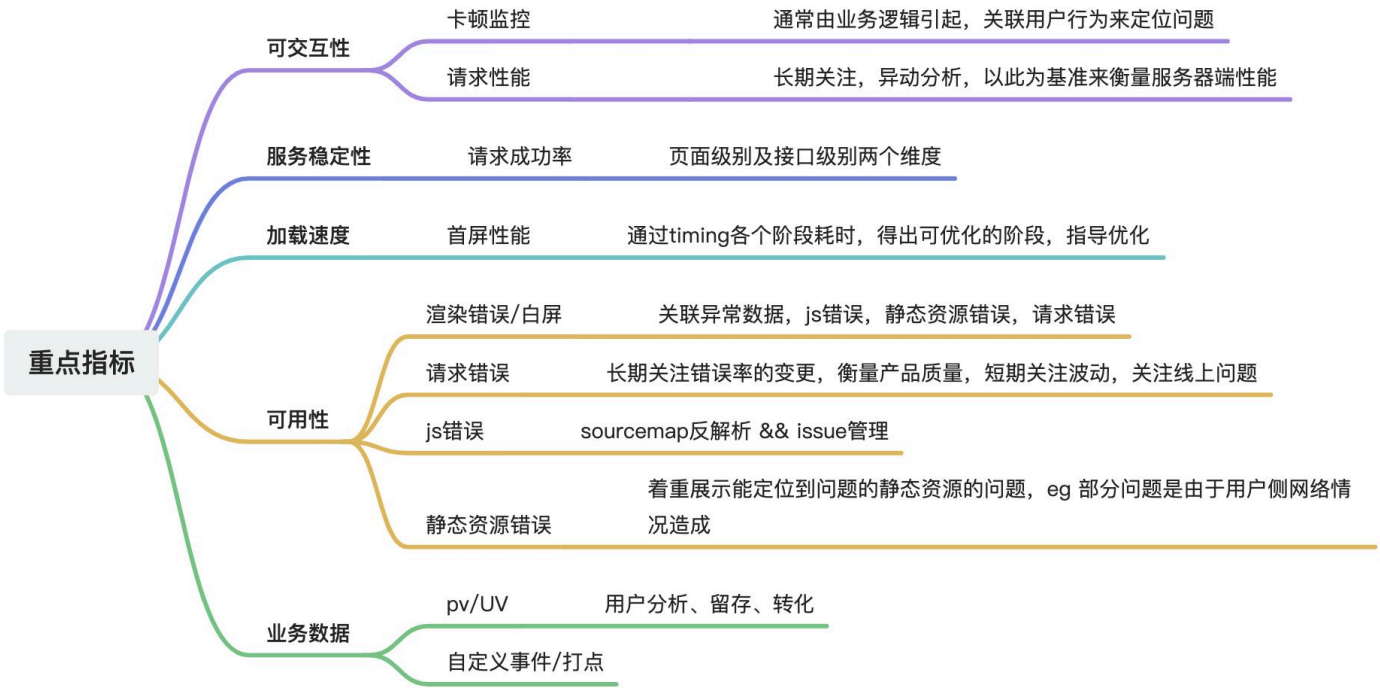
### 清洗存储

- user-agent解析：浏览器版本、系统版本、机型、设备品牌等等
- IP解析：地区、省份、城市、运营商、地理位置等等
- 处理js错误：堆栈归一化、堆栈反解析（堆栈格式化，反解析，将混淆后的代码通过sourcemap 反解析成原始堆栈）
- 分类型落表落库

### 数据消费

- 总览分析
  能够看到站点的整体情况（整体的性能评分和异常情况，有效的展现需要关注的重点问题和需要跟进的异常情况）
- 各功能模块消费视角 && 多维分析（不同功能模块的关注点不同，eg js错误关注的还是错误的趋势，具体错误的跟进和修复，性能主要关注单个页面的性能指标进行针对性的优化）
- 单点查询，针对用户全生命周期上报数据的重建展示（还原单一用户从进入站点开始到结束会话的整个生命周期，发生的所有事情，快速定位问题，上面的一些消费视角都需要业务主动去数据，更重要的是主动通知业务，引导业务去发现问题和解决问题）
- 数据订阅 && 实时报警（ 数据订阅： 将业务关注的数据，根据固定的周期推送给业务，实时报警：引导业务或者默认配置一些异常情况的报 警，当满足条件时，主动推送给业务）
- issue管理 （将需要跟进的问题都管理起来，分配责任人）

## 四、前端监控的关注点




## 五、前端监控的最终效果

及时发现线上问题，让一切有迹可循

监控站点体验，指导性能优化

运行稳定建设