移动海量服务下基于React的高性能同构实践

IMWEB-willliang

介绍

- 腾讯高级工程师—梁伟盛（大圣）

- IMWeb团队架构师

- 先后参与花样，交友，NOW直播等业务的核心开发和架构设计

- 现在负责互动视频业务前端架构设计与开发

NOW直播

素人直播


年糕呀
132人
+ 关注
魅力 727272
one
送一个爱你哦
×1
聪明的我
送一个爱你哦
×1
一个人世界有没有陪: 那么厉害
呀惹初♡ 来了
xuer: cf，dnf
陌浅离 来了
纯属“虚构”: 你说的我都成大神了
聪明的我 送了 x 1个

花样直播

秀场直播

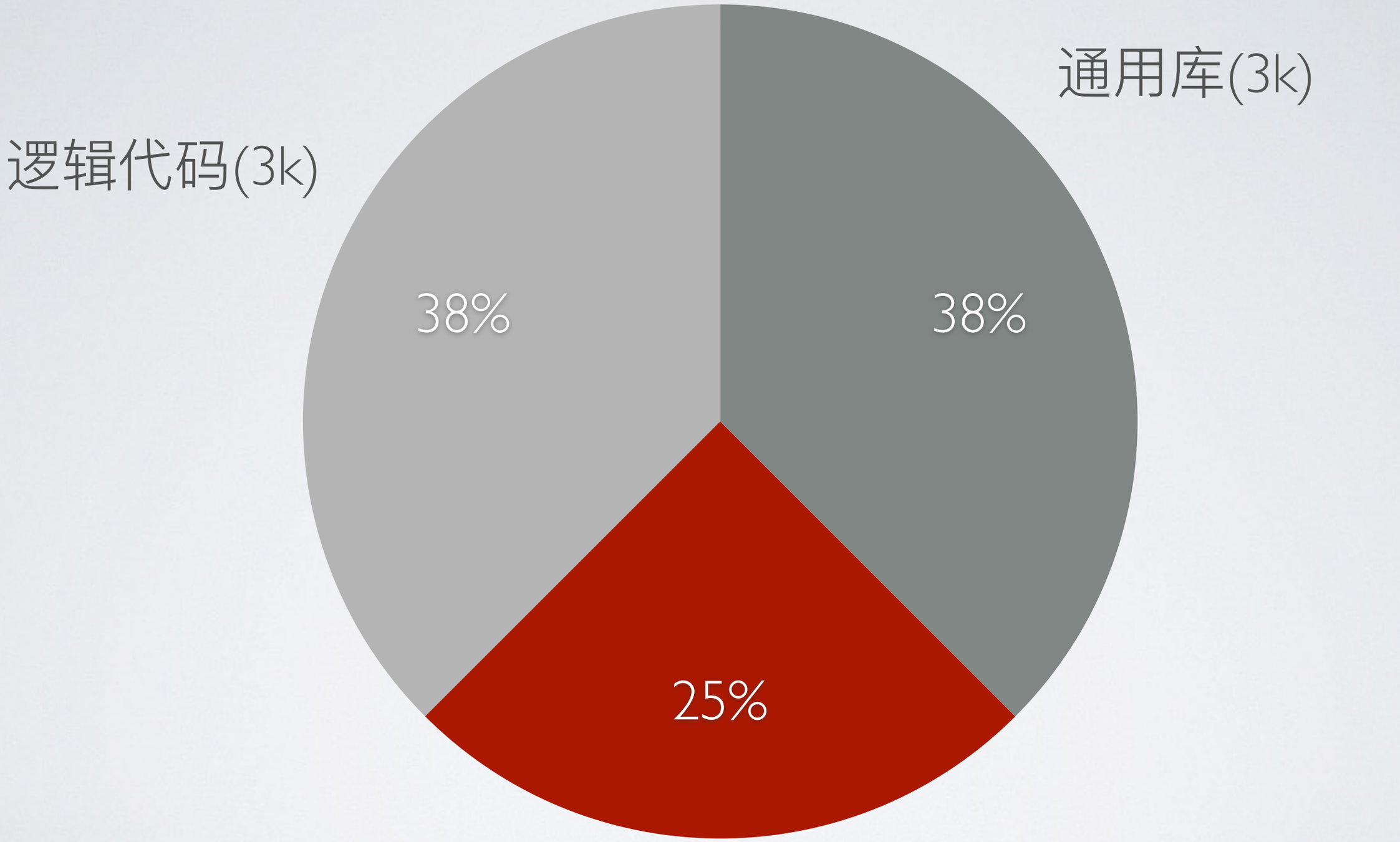

返回
子岩是个男孩...
8688人观看
APP
大厅
关注
子岩是个男孩子耶_____ 乖
子岩是个男孩子耶_____ 继续
流浪汉丨安静听歌丨
黄片百度云加QQ120... 送给主播 学习宝典 x 1
子岩是个男孩子耶_____ 继续啊
恋染到底丶醉酒 为毛我没有子岩徽章？？？
子岩是个男孩子耶_____ 刷10个
曾经我也开心过...隐形人 希望别辣耳朵
快与主播互动吧~

CONTENTS

拨号时代

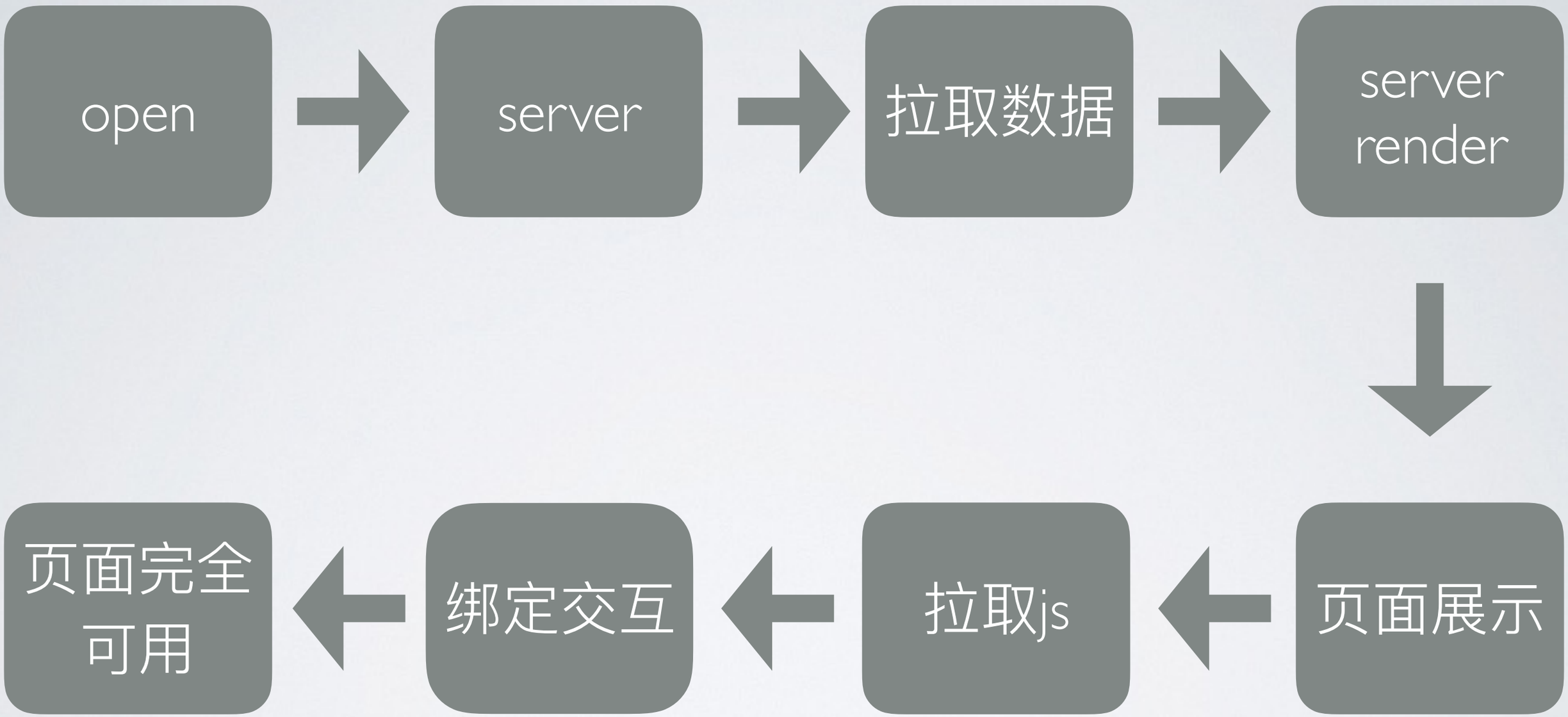
内置56K调制调解器

页面交互简单



GZIP后大小

HTML页面服务器端渲染（直出）



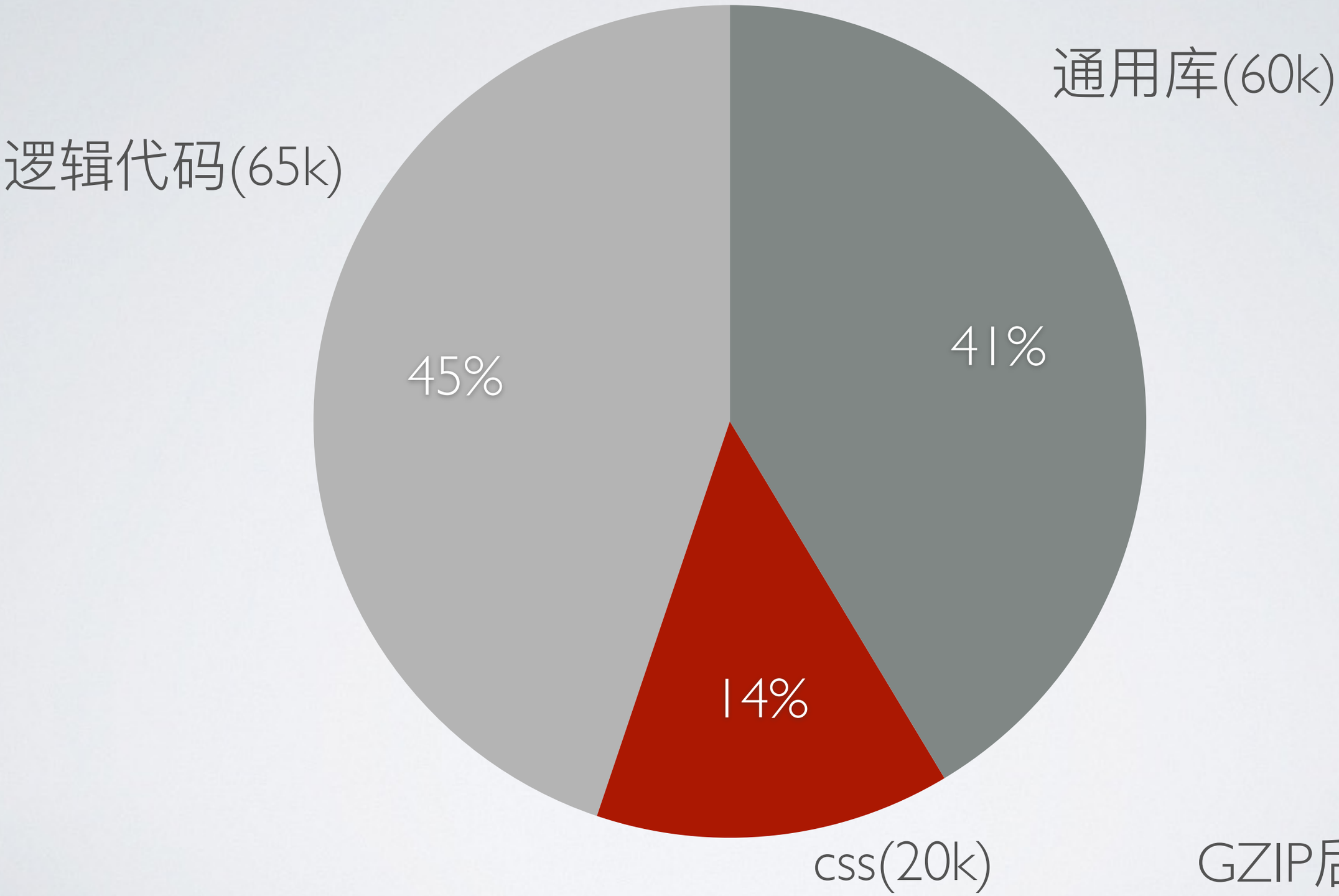
完全展示用时10s

10s

宽带时代

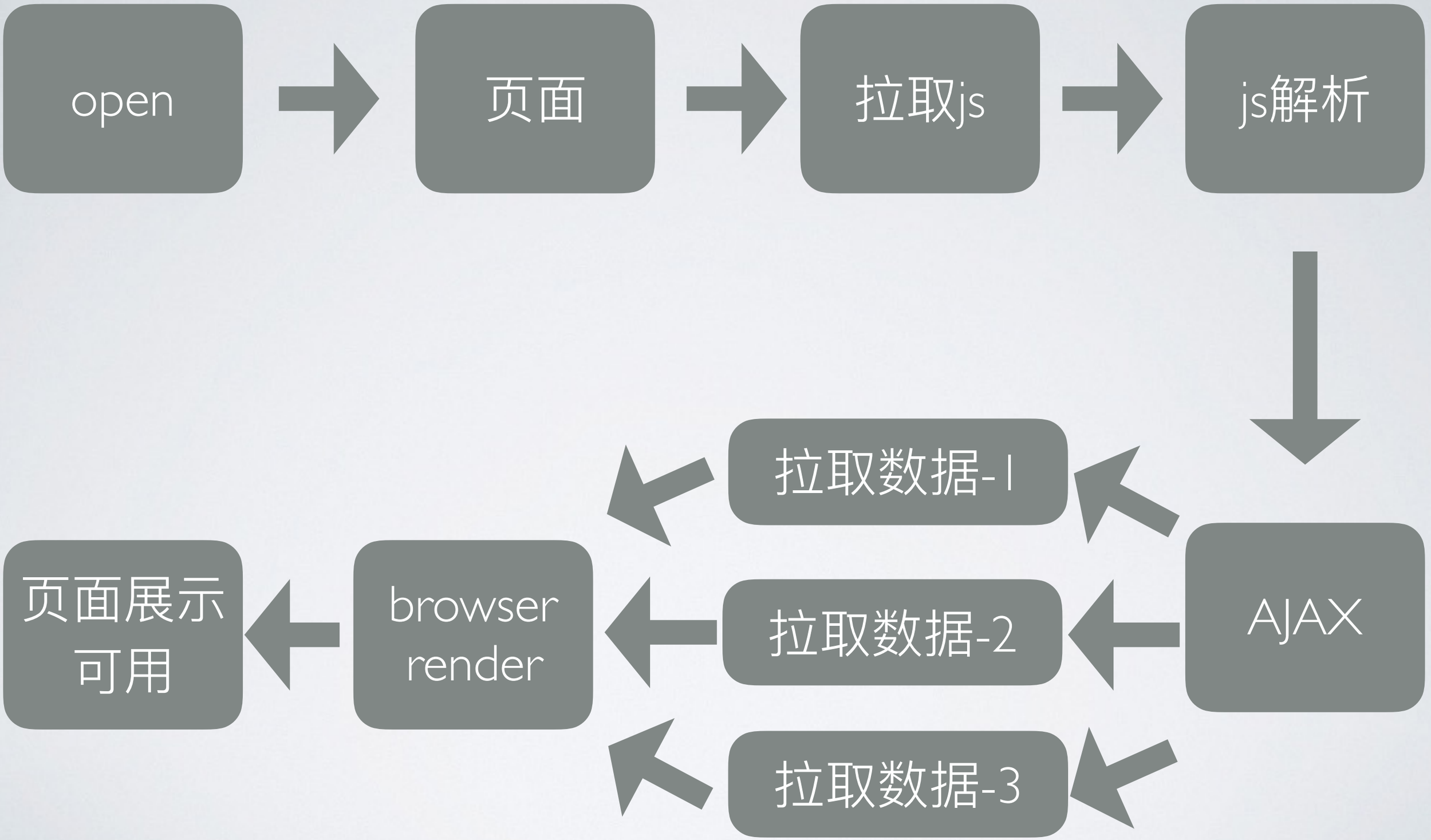
ADSL“猫”(1M/2M)

前端交互开始复杂，js代码越来越大



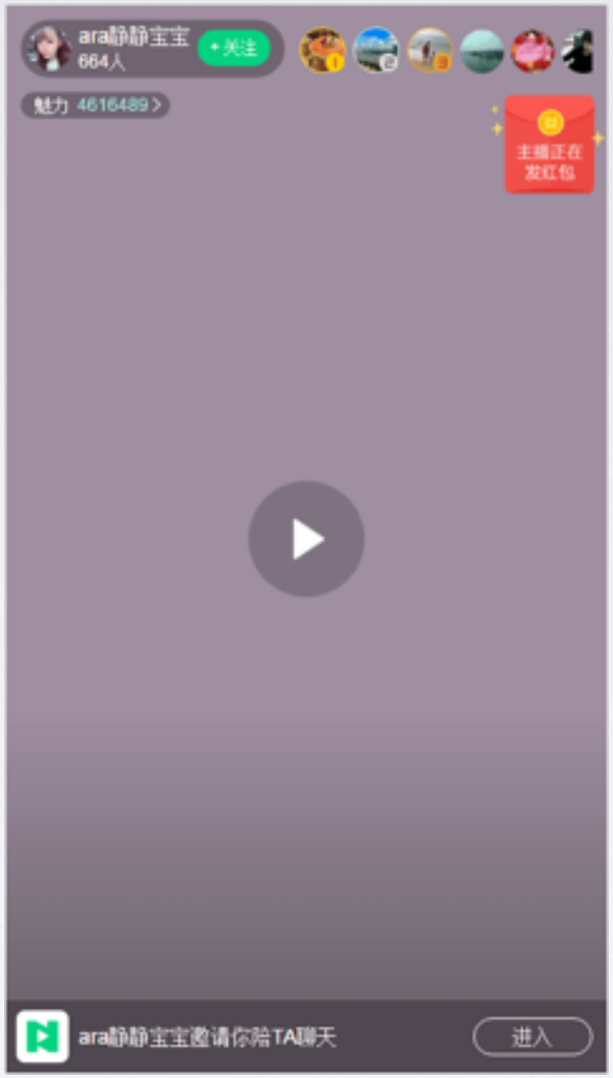
GZIP后大小

前后端开始分离，异步加载数据



完全展示一共用时4s

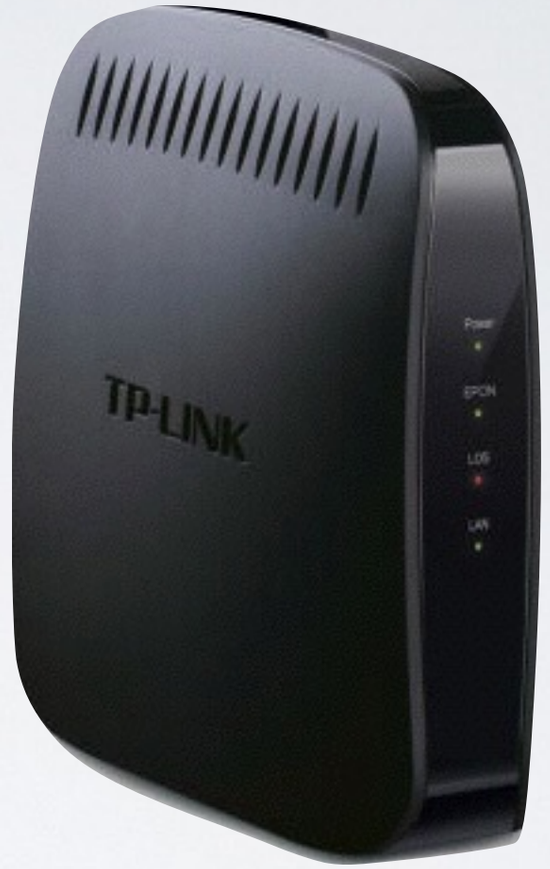
2.5s



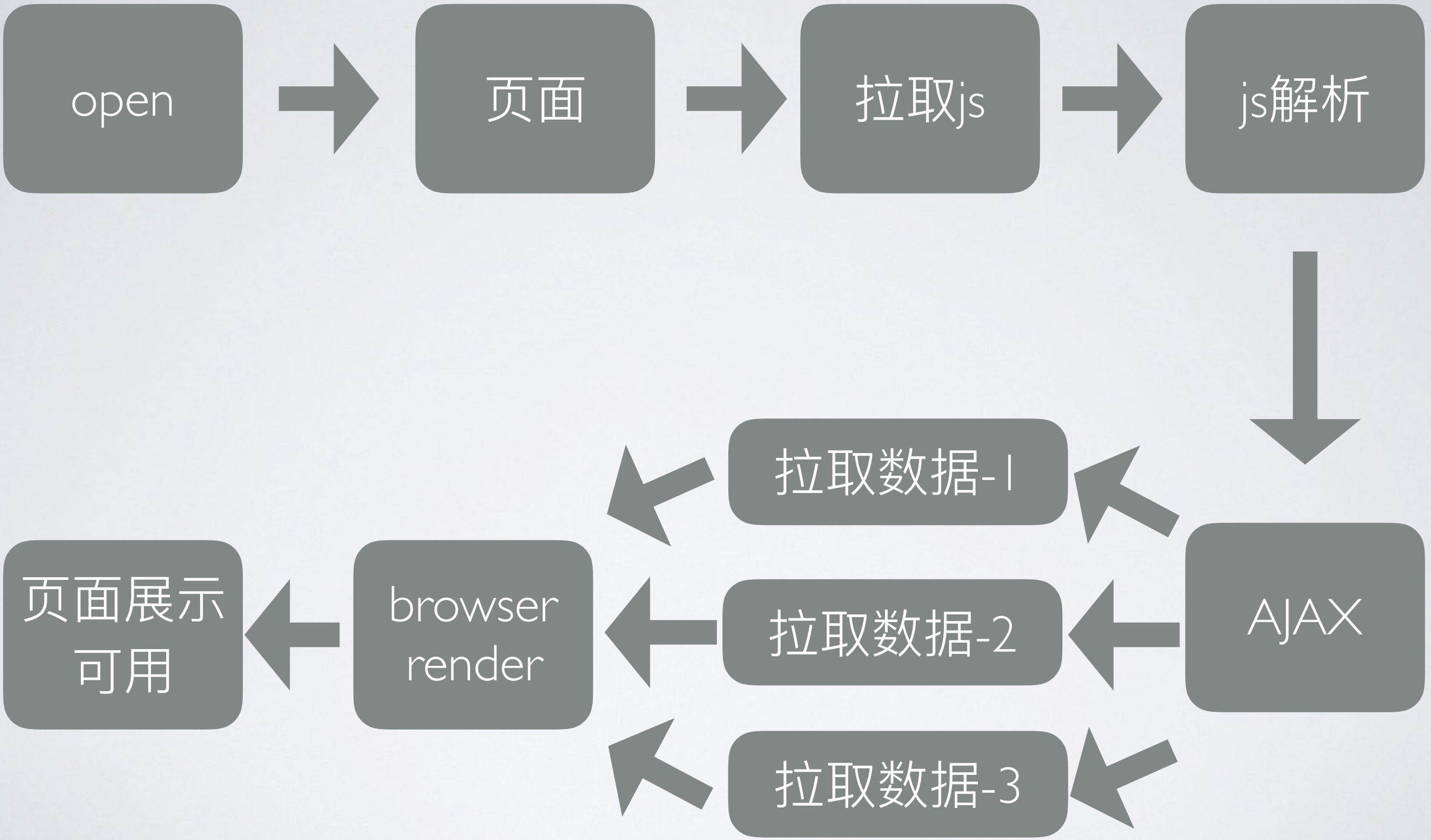
1.5s

光纤时代

光“猫” （20M/50M/100M）

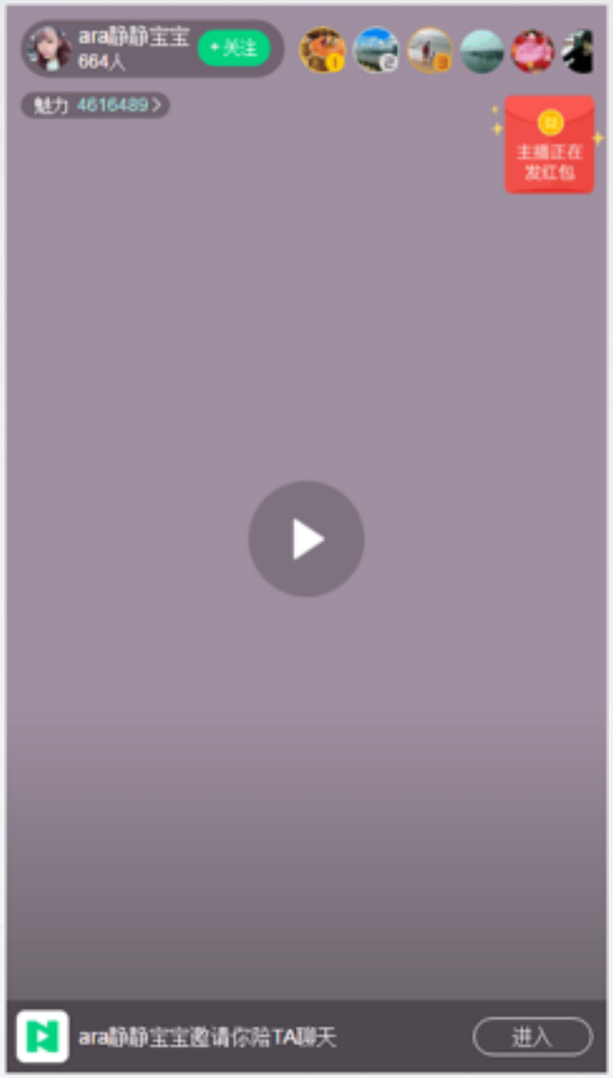


前后端开始分离，异步加载数据



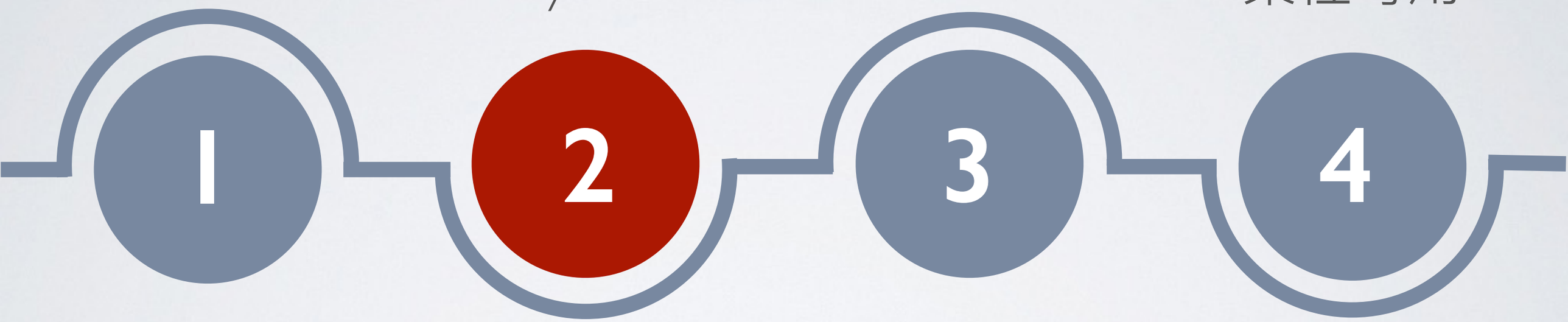
完全展示一共用时2.2s

1.2s



1s

CONTENTS

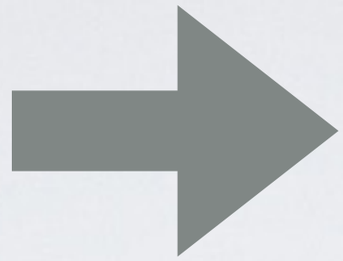


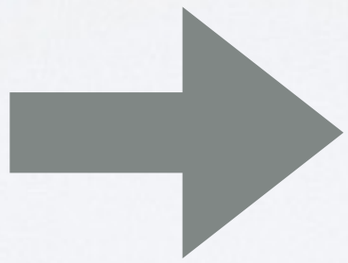
时代变迁

why react

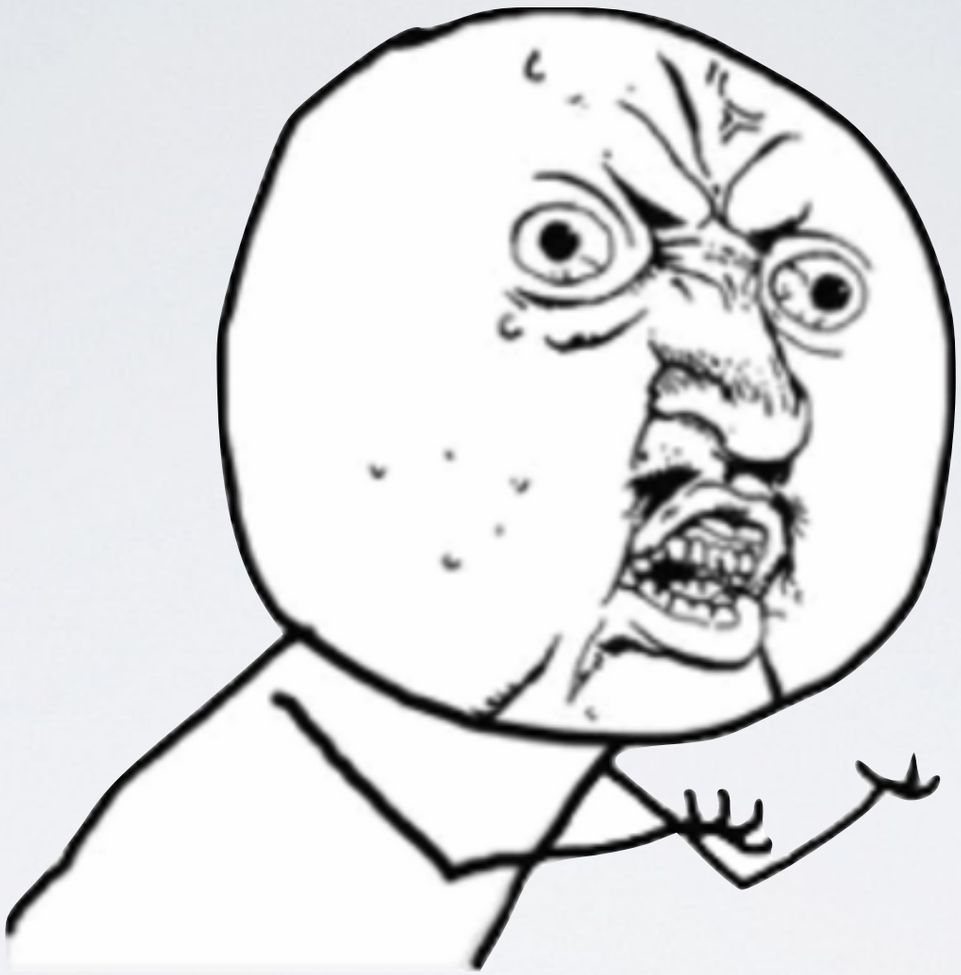
性能优化

柔性可用

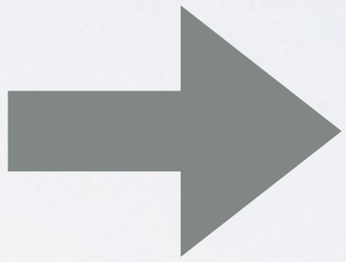




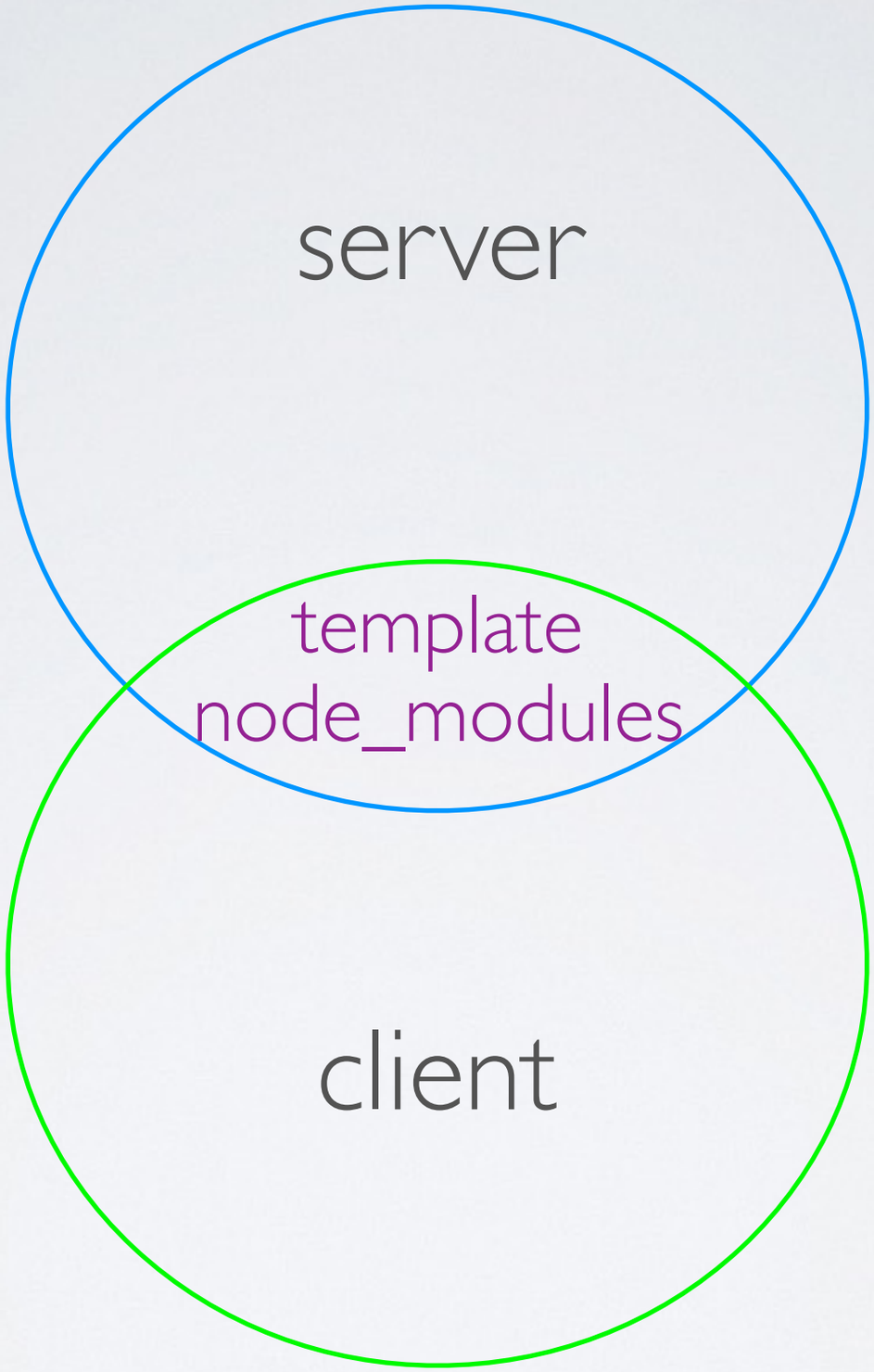
Velocity



我们是javascript程序员啊！



EJS

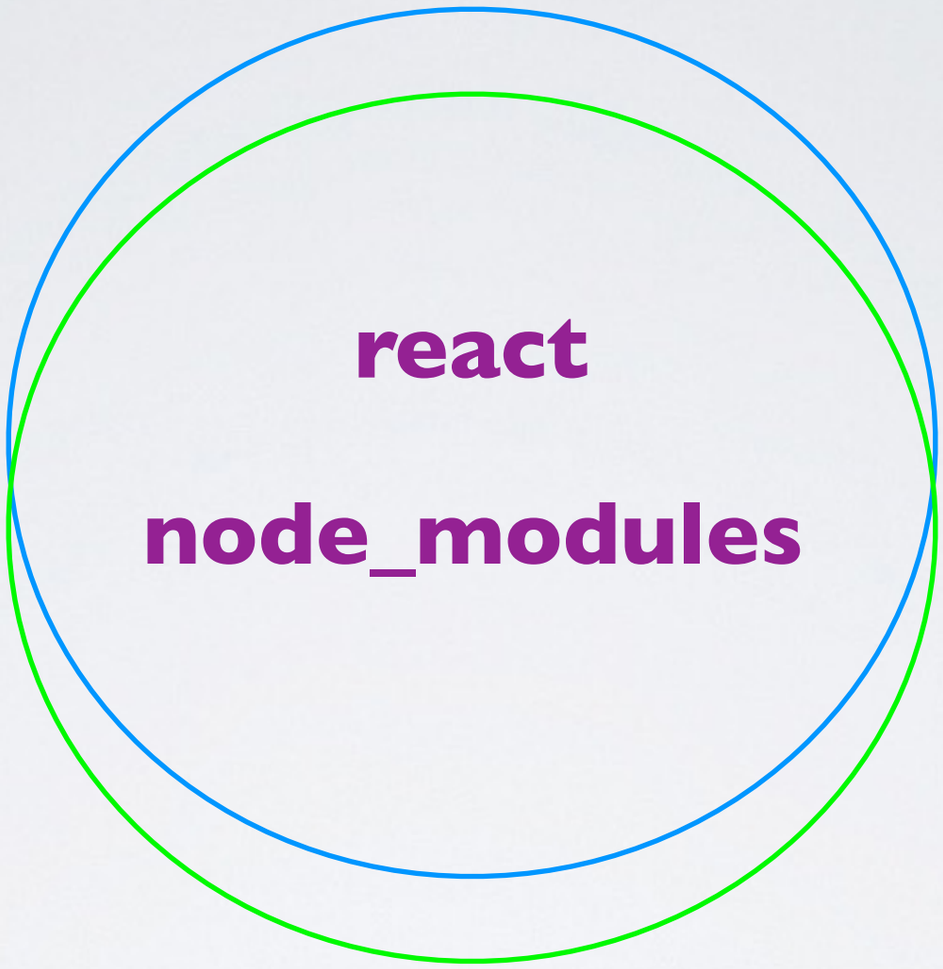

server
template
node_modules
client

一处编译，到处运行

同构(isomorphic)

Node.js+React.js

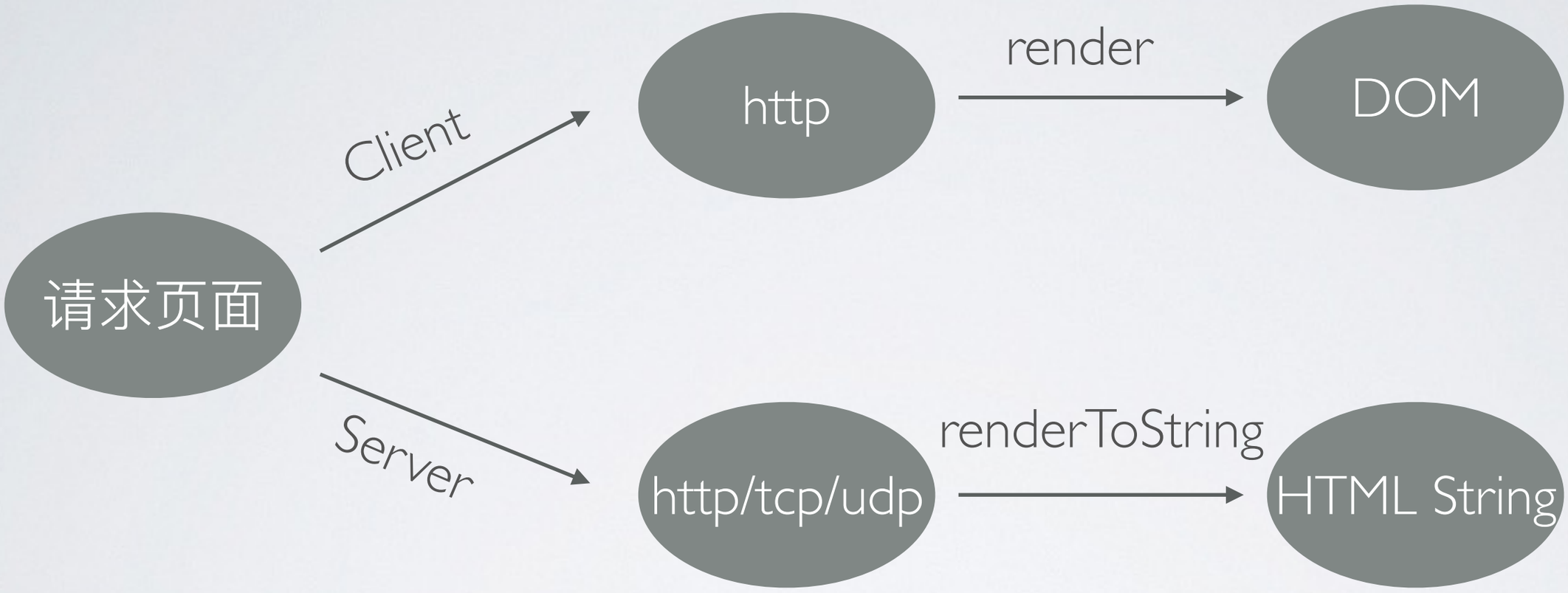
server



client

一处编写，到处运行

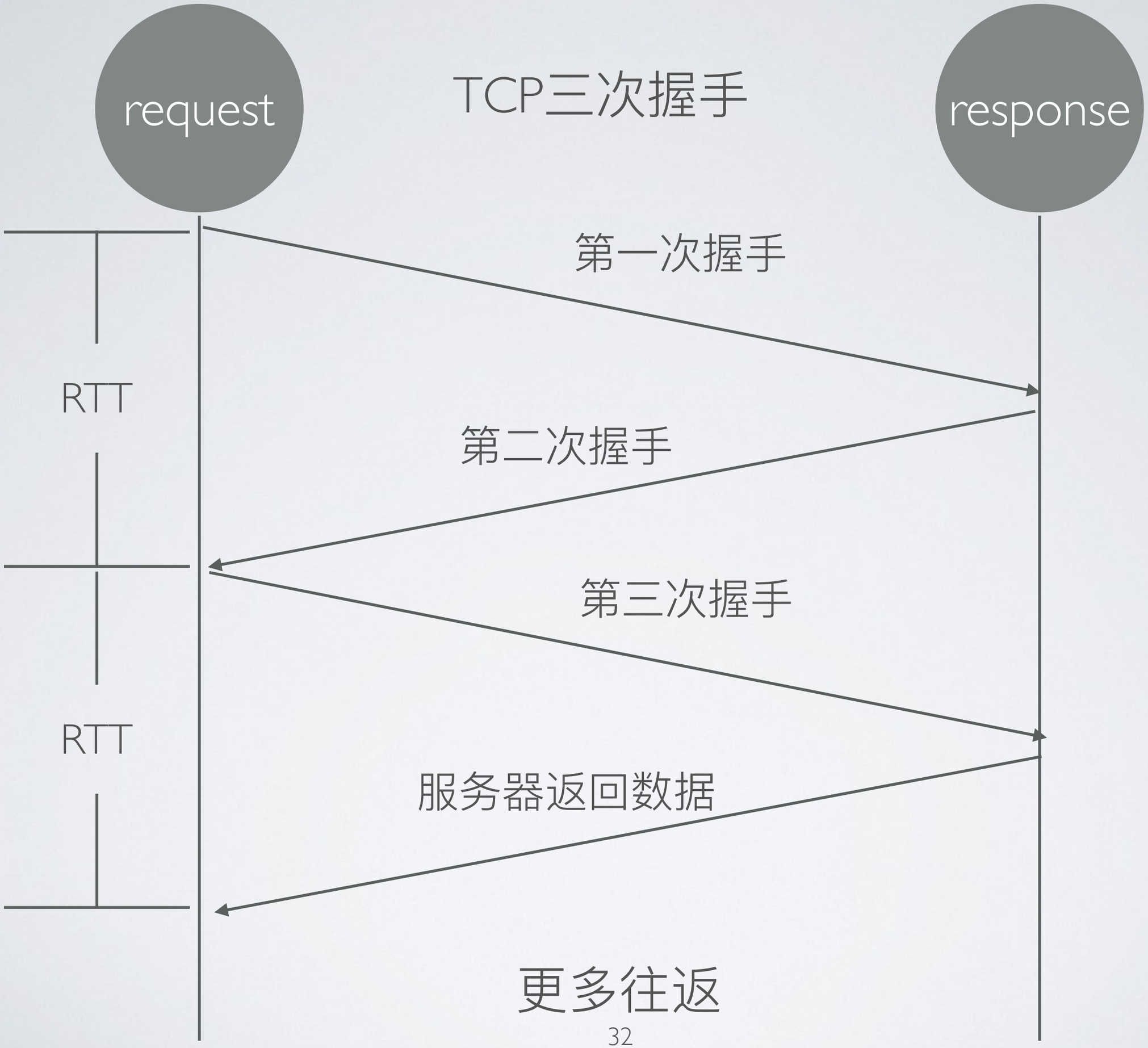
JSX


Client
http
render
DOM
请求页面
Server
http/tcp/udp
renderToString
HTML String

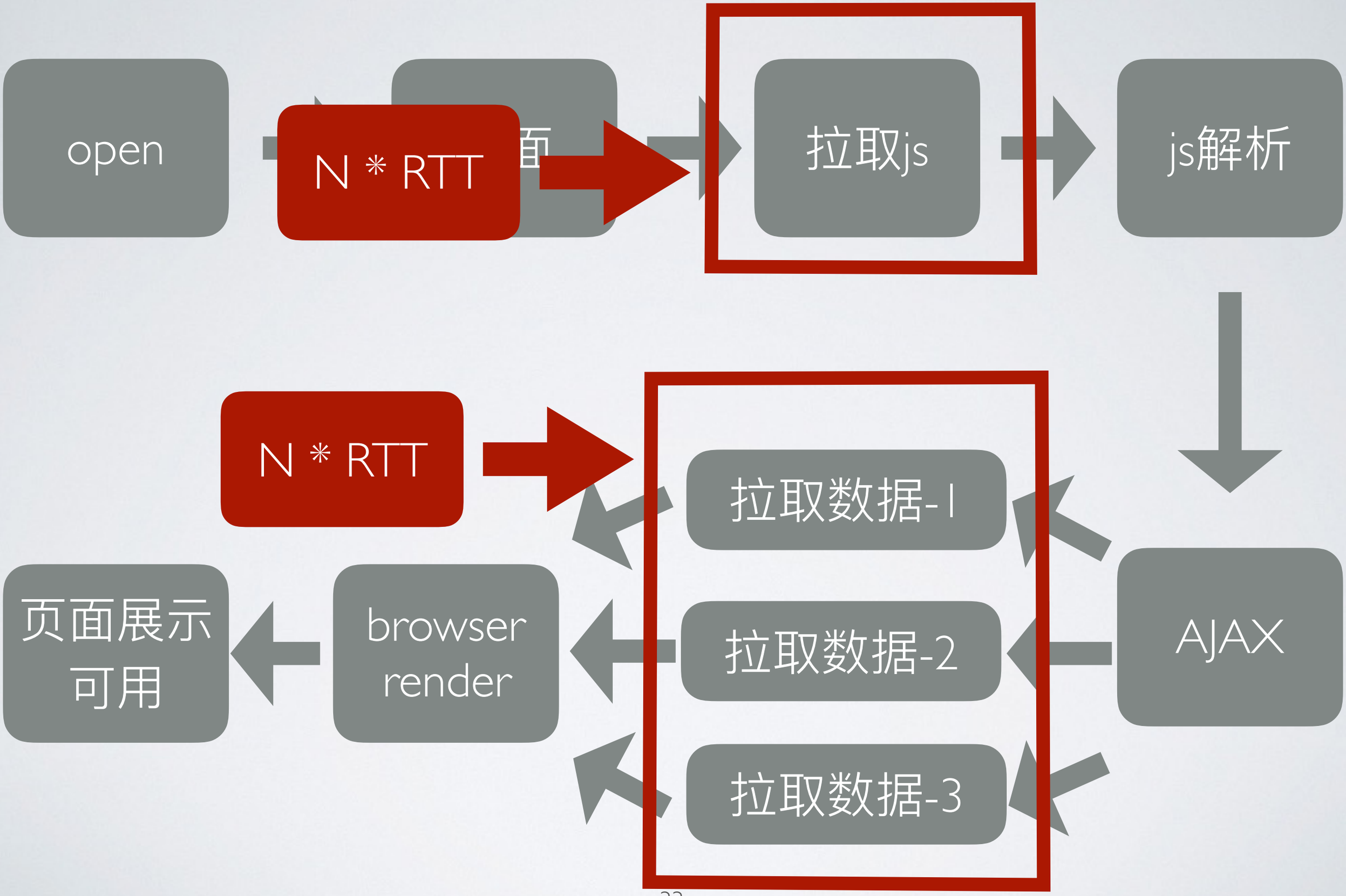
CONTENTS

RTT

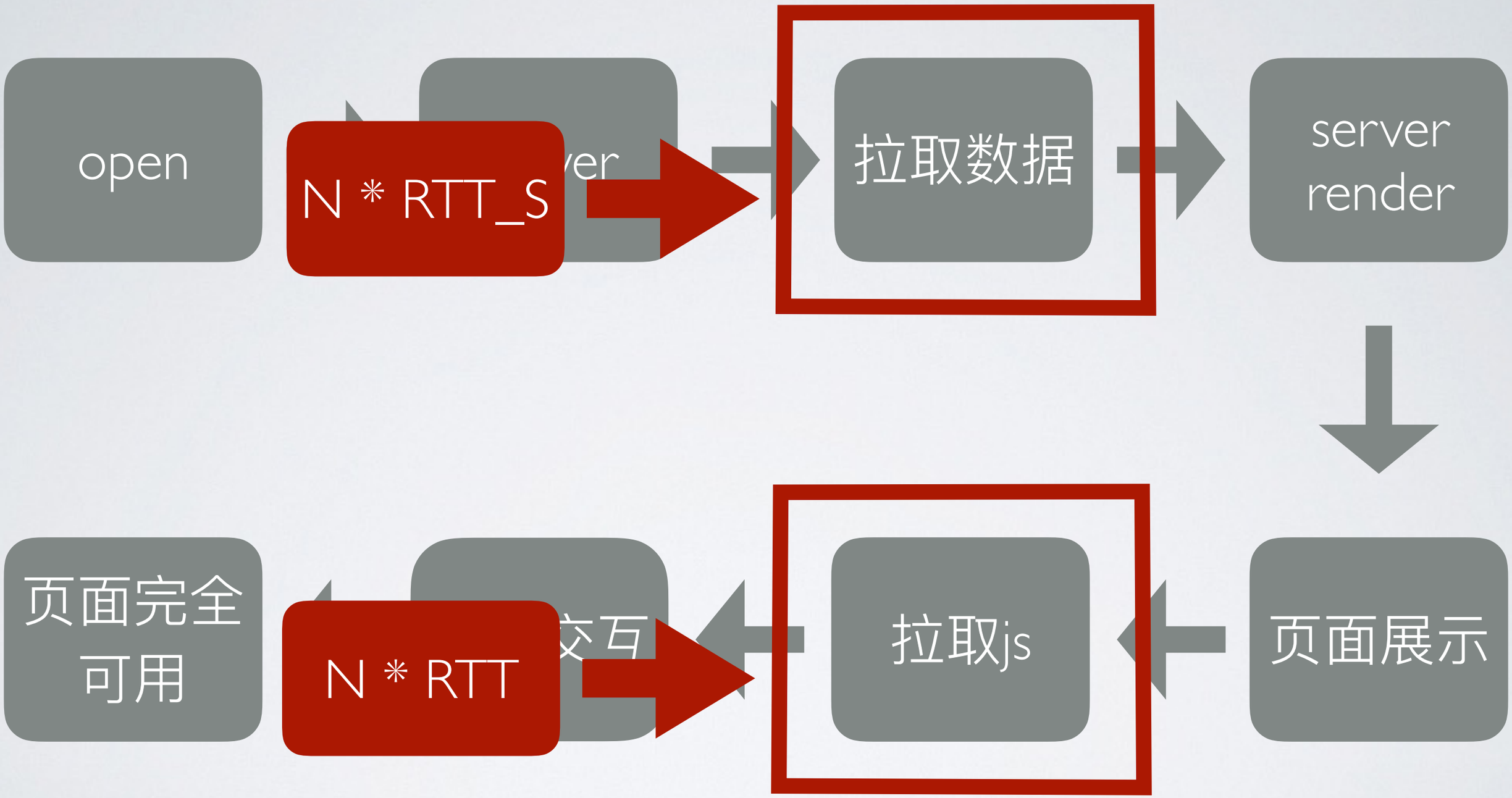
- RTT 表示 Round-Trip Time，即“往返时延”，表示从发送端发送数据开始，到发送端收到来自接收端的确认（接收端收到数据后便立即发送确认），总共经历的时延。


request
TCP三次握手
response
第一次握手
RTT
第二次握手
第三次握手
RTT
服务器返回数据
更多往返

前后端开始分离，异步加载数据



HTML页面服务器端渲染（直出）

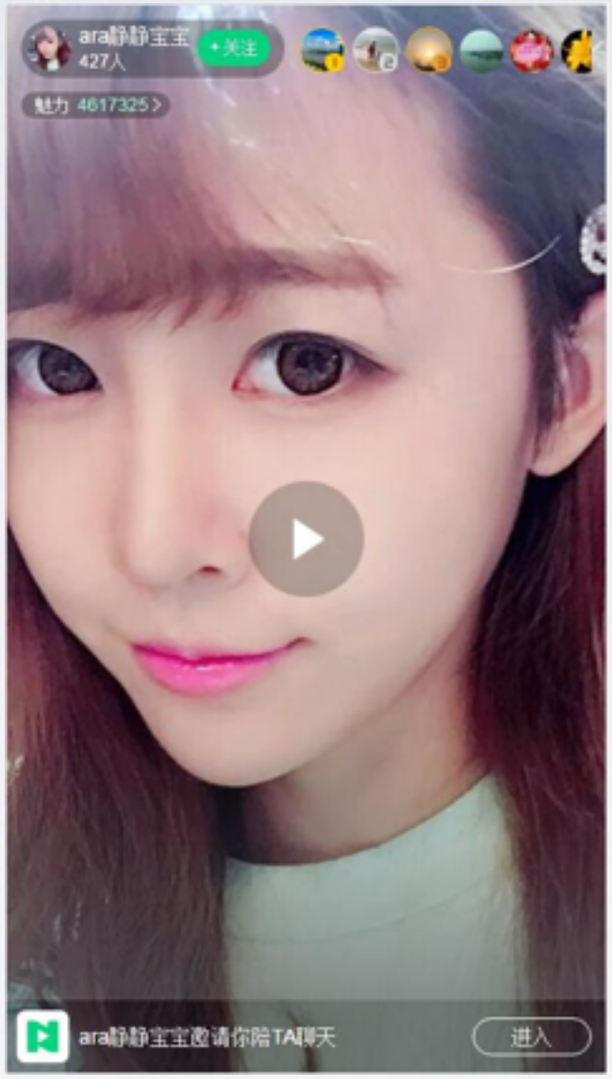


内网链路延迟时间(RTT)

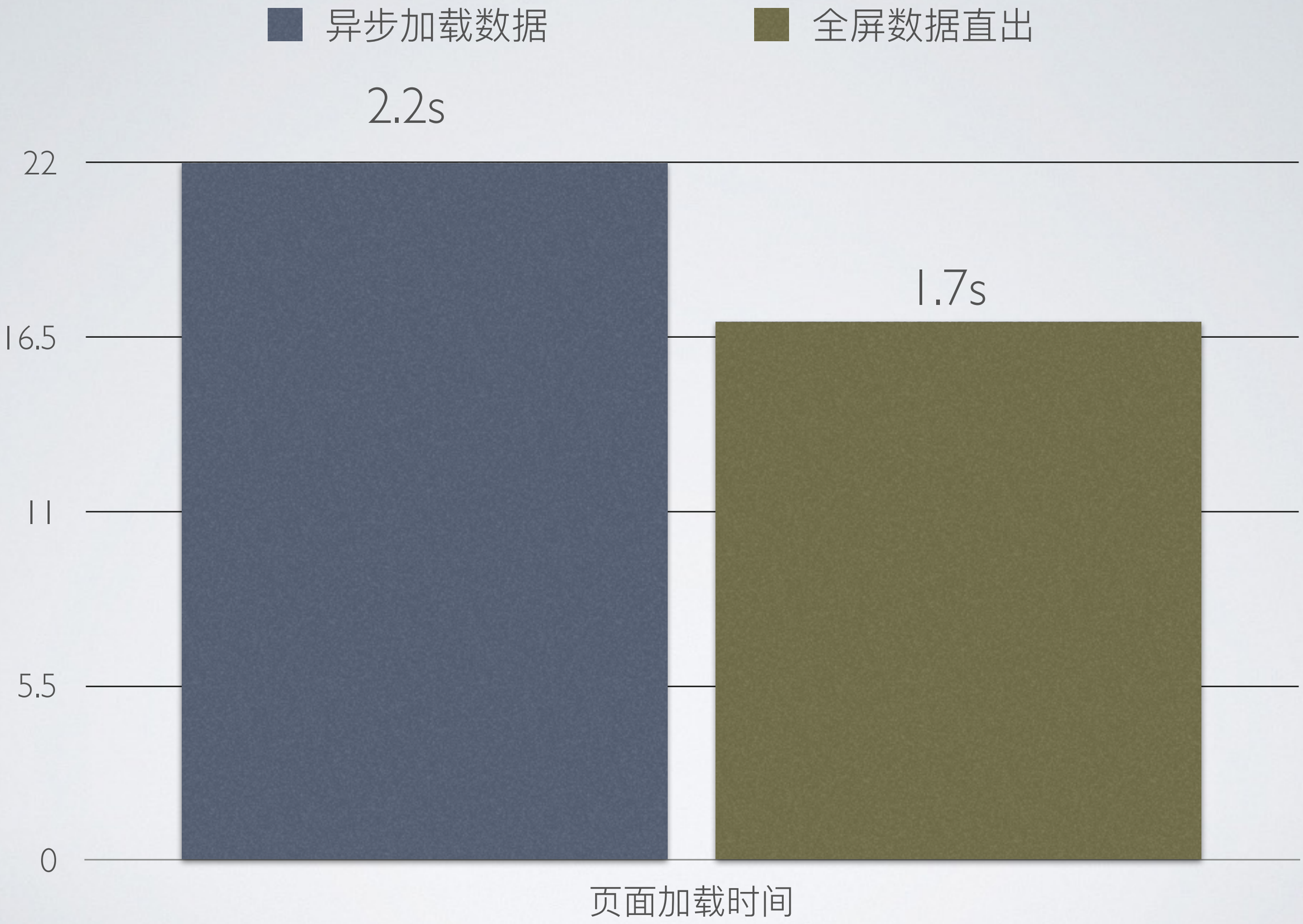
绝对比外网短

完全展示一共用时1.7s

1.4s



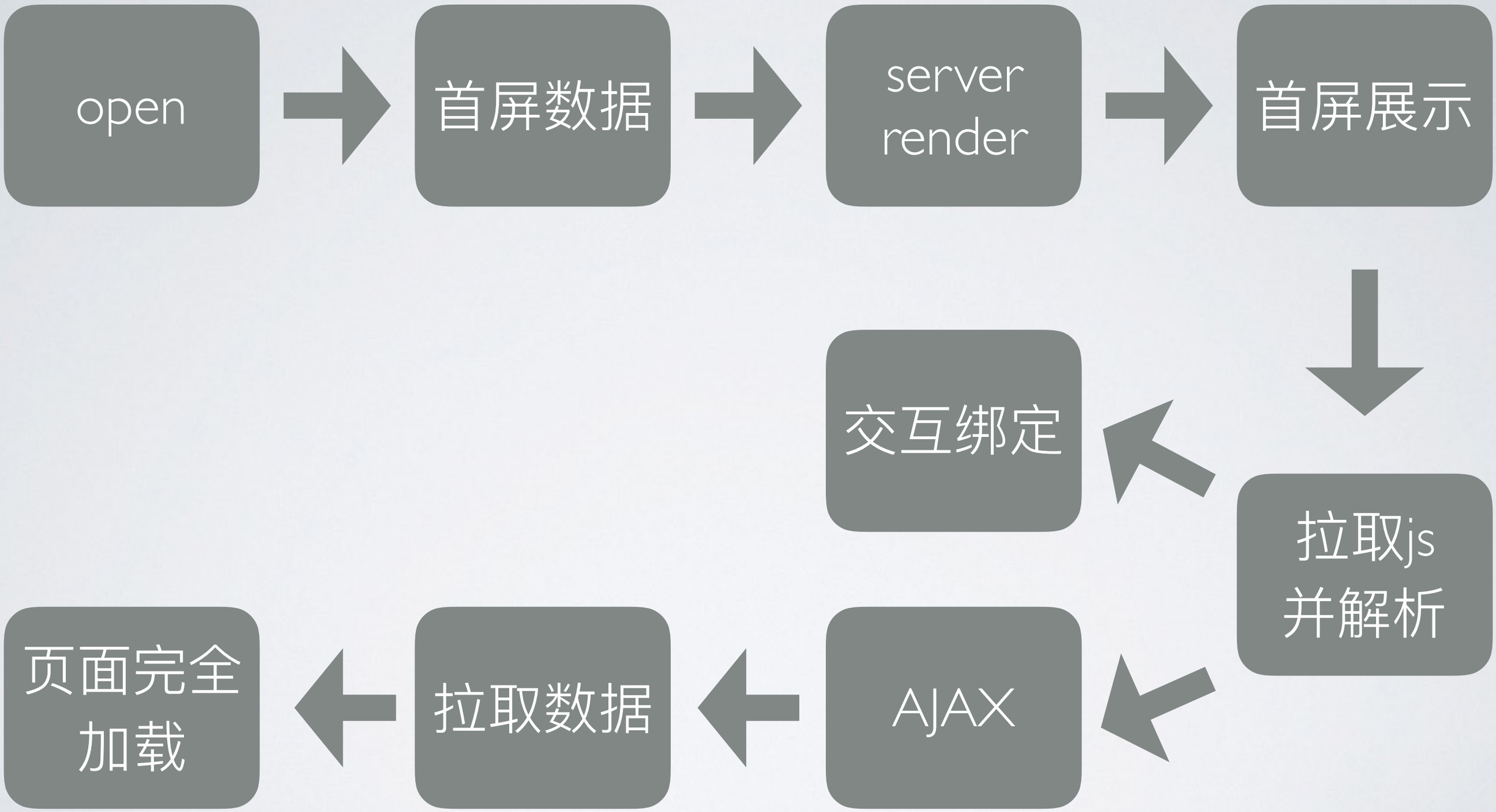
0.3s


异步加载数据
全屏数据直出
2.2s
1.7s
22
16.5
11
5.5
0
页面加载时间

首屏

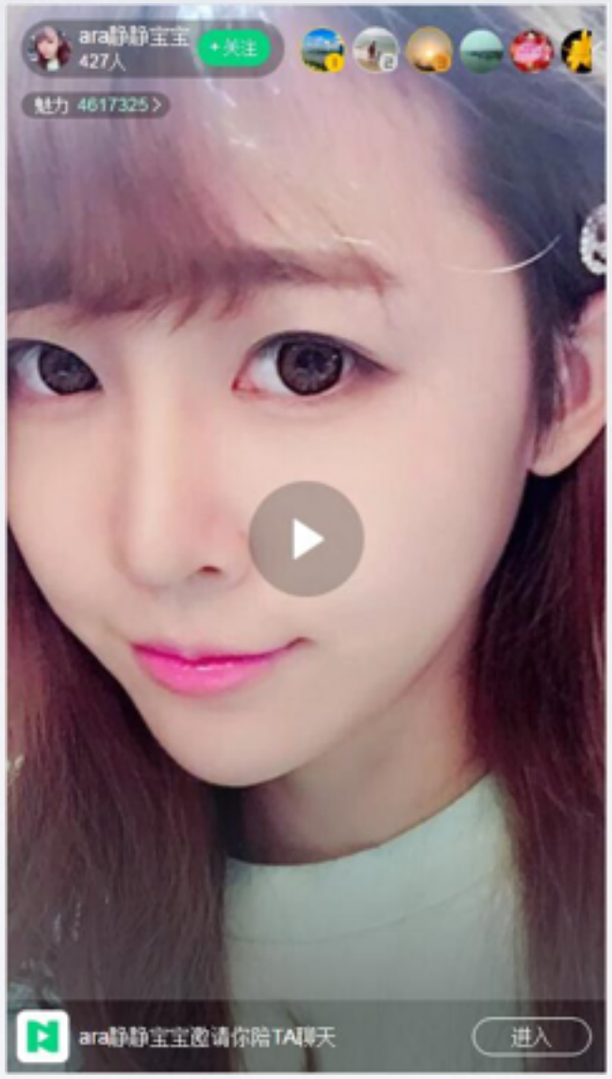
- 浏览器完成页面第一屏渲染的时间，是用户的第一感知，已经成为业界衡量网页速度快慢的首要指标

服务器端渲染（直出），首屏概念



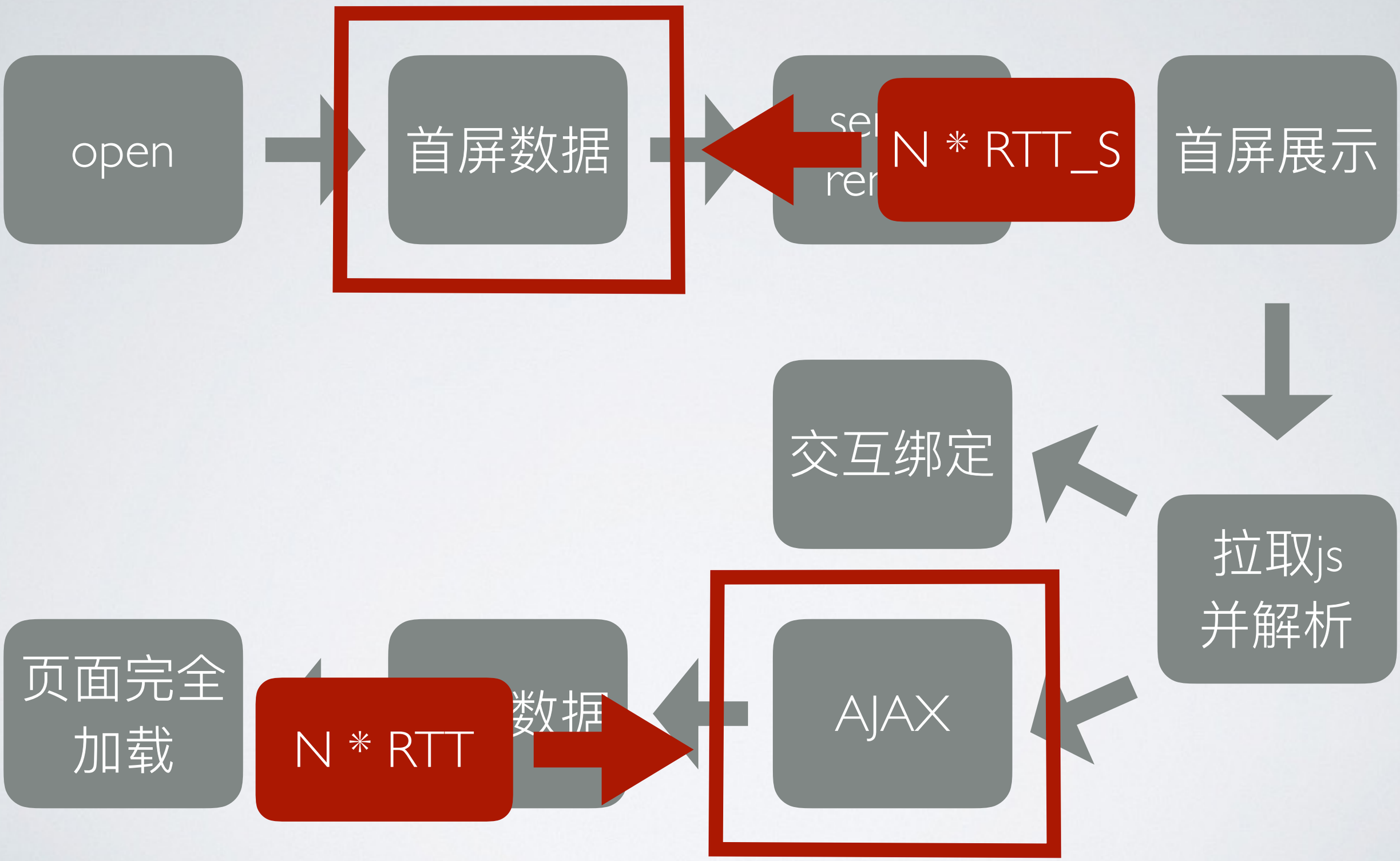
首屏用时1s

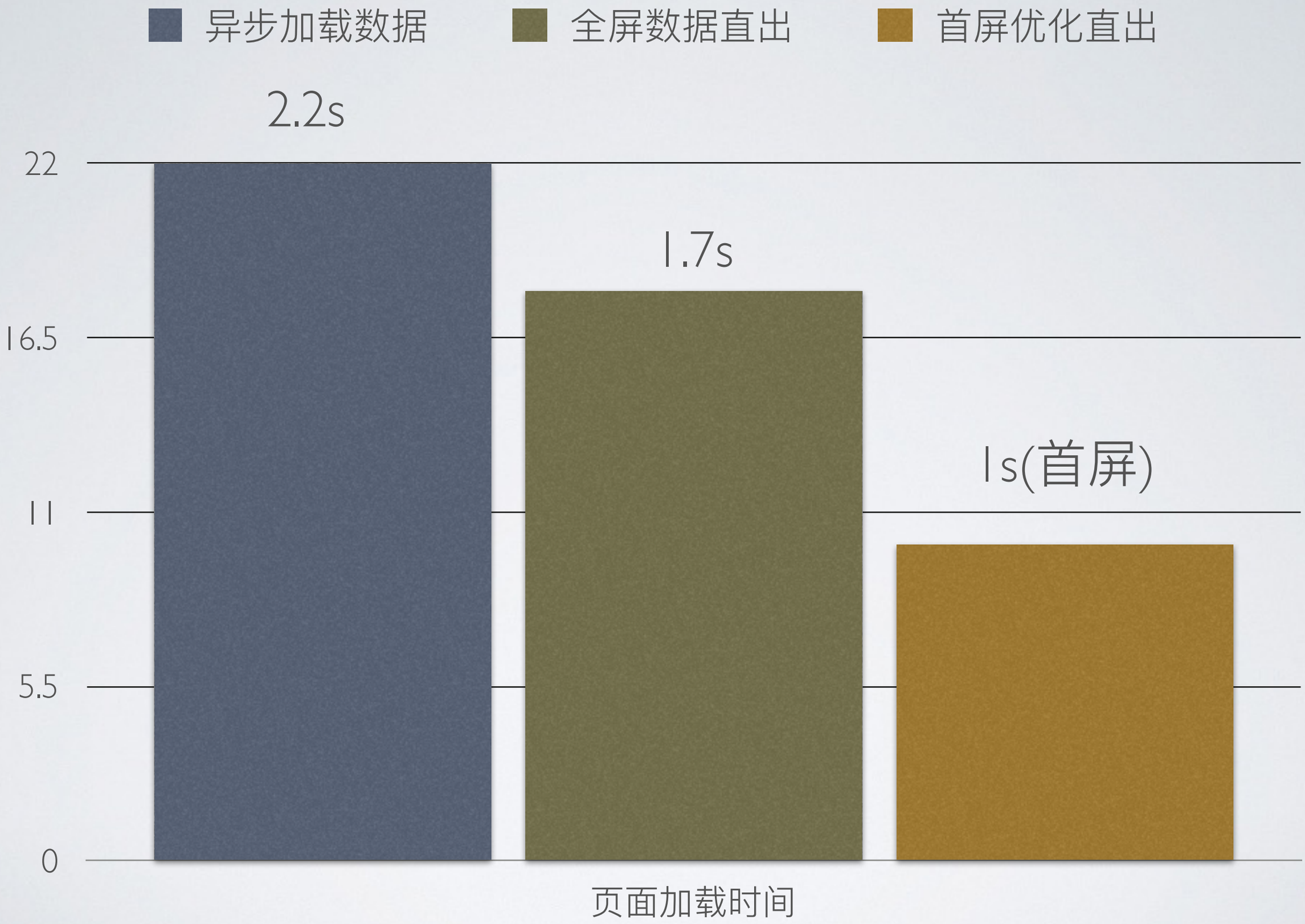
1s



0.3s

首屏概念，减少非首屏的RTT




异步加载数据
全屏数据直出
首屏优化直出
2.2s
1.7s
1s(首屏)
22
16.5
11
5.5
0
页面加载时间

减少RTT非常重要?

UDP大法好?

现有方案

http请求数据

N * RTT_S

JSON文本协议

改造方案

udp请求数据

N * RTT_S

二进制协议

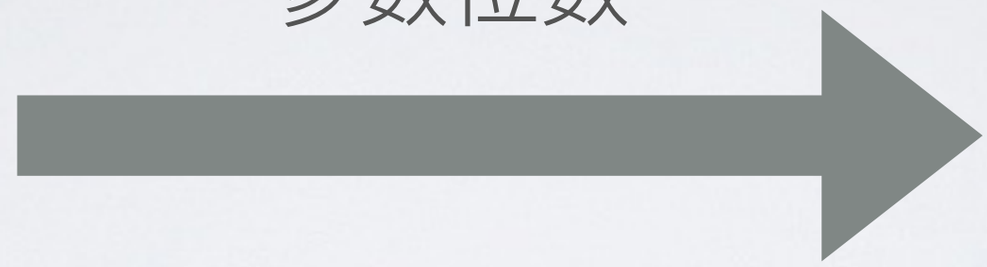
简单的例子

- json: {id: 101, str: 'Hello'}
- {id: 101, str: 'Hello'}: <Buffer 7b 22 69 64 22 3a 31 30 31 2c 22 73 74 72 22 3a 22 48 65 6c 6c 6f 22 7d>
- 101Hello: <Buffer 65 48 65 6c 6c 6f>

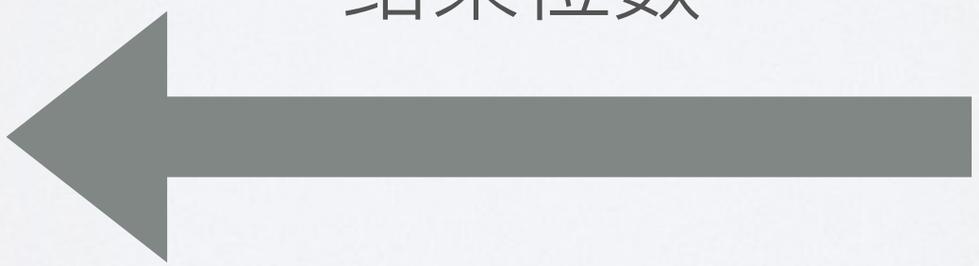
24 vs 6

按位截取

参数位数



结果位数

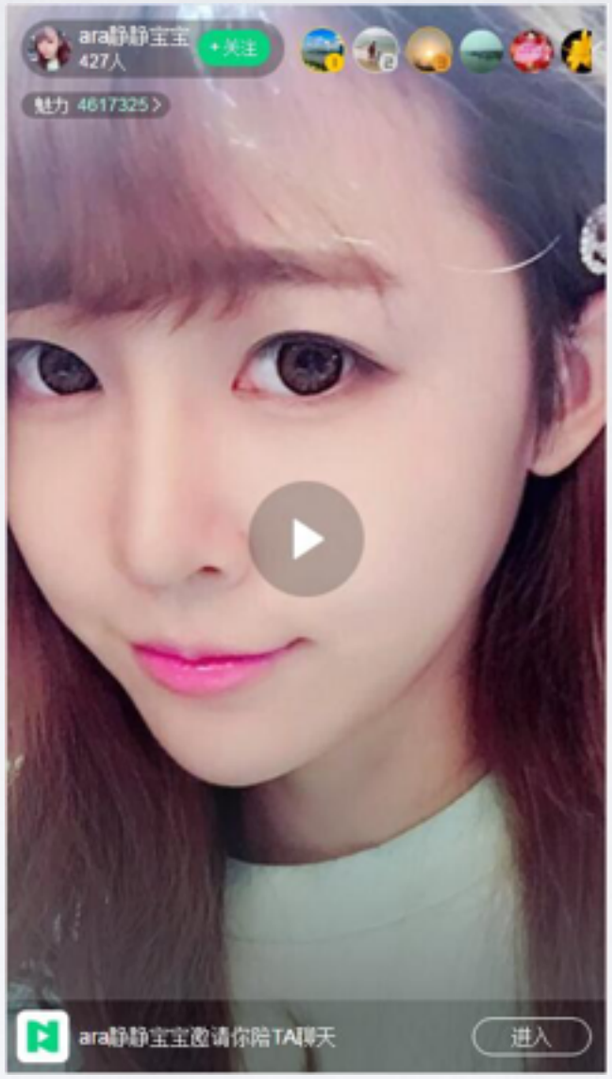


- Goo
- Prot 据存储格 合做数据
- http 6c?utm

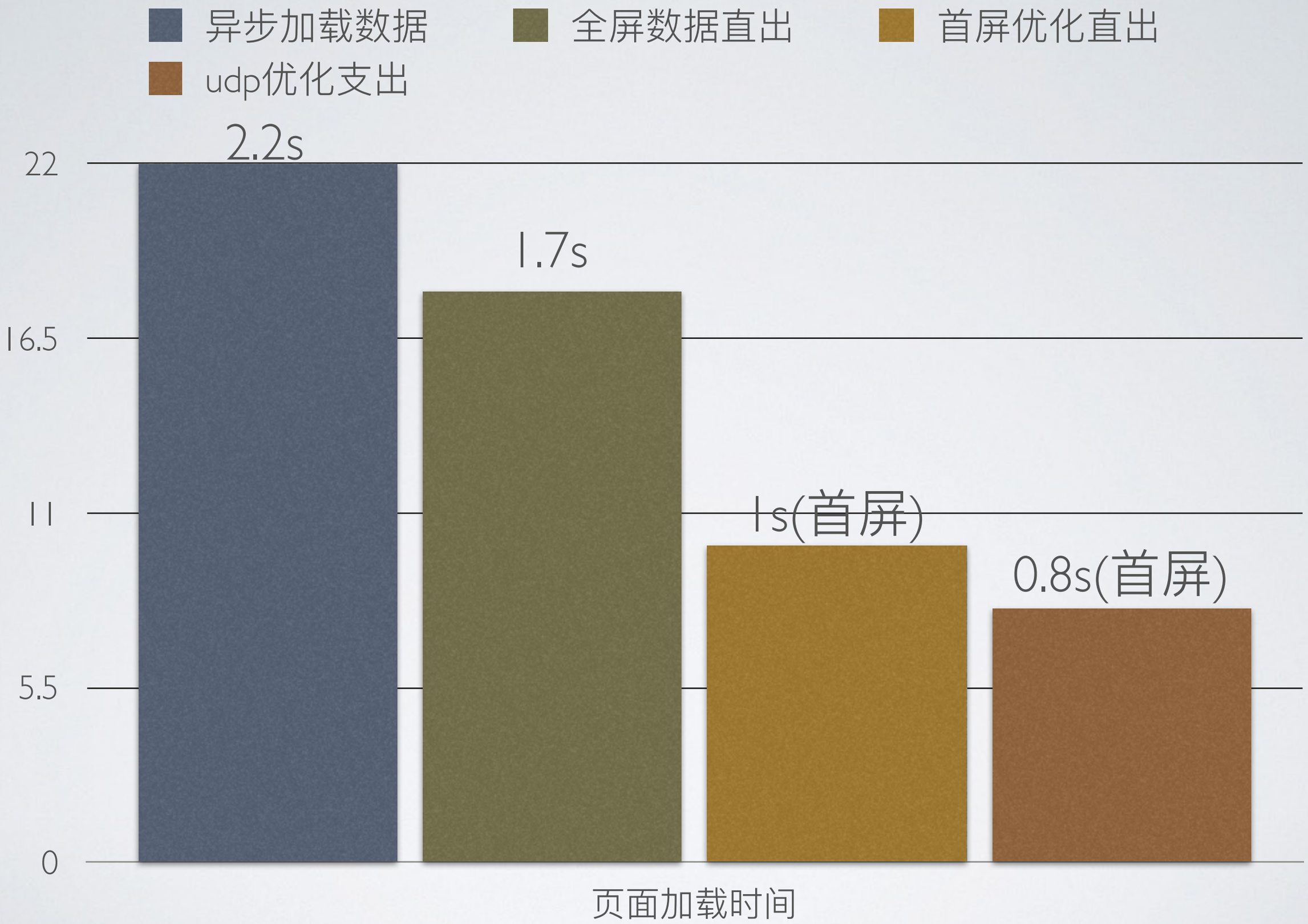
```
package test;

message hello {

    message helloReq {
        required int32 id = 1;
        required string str = 2;
    }

    message helloRsp {
        required int32 retcode = 1;
        optional string reply = 2;
    }
}
```

首屏用时0.8s

0.8s

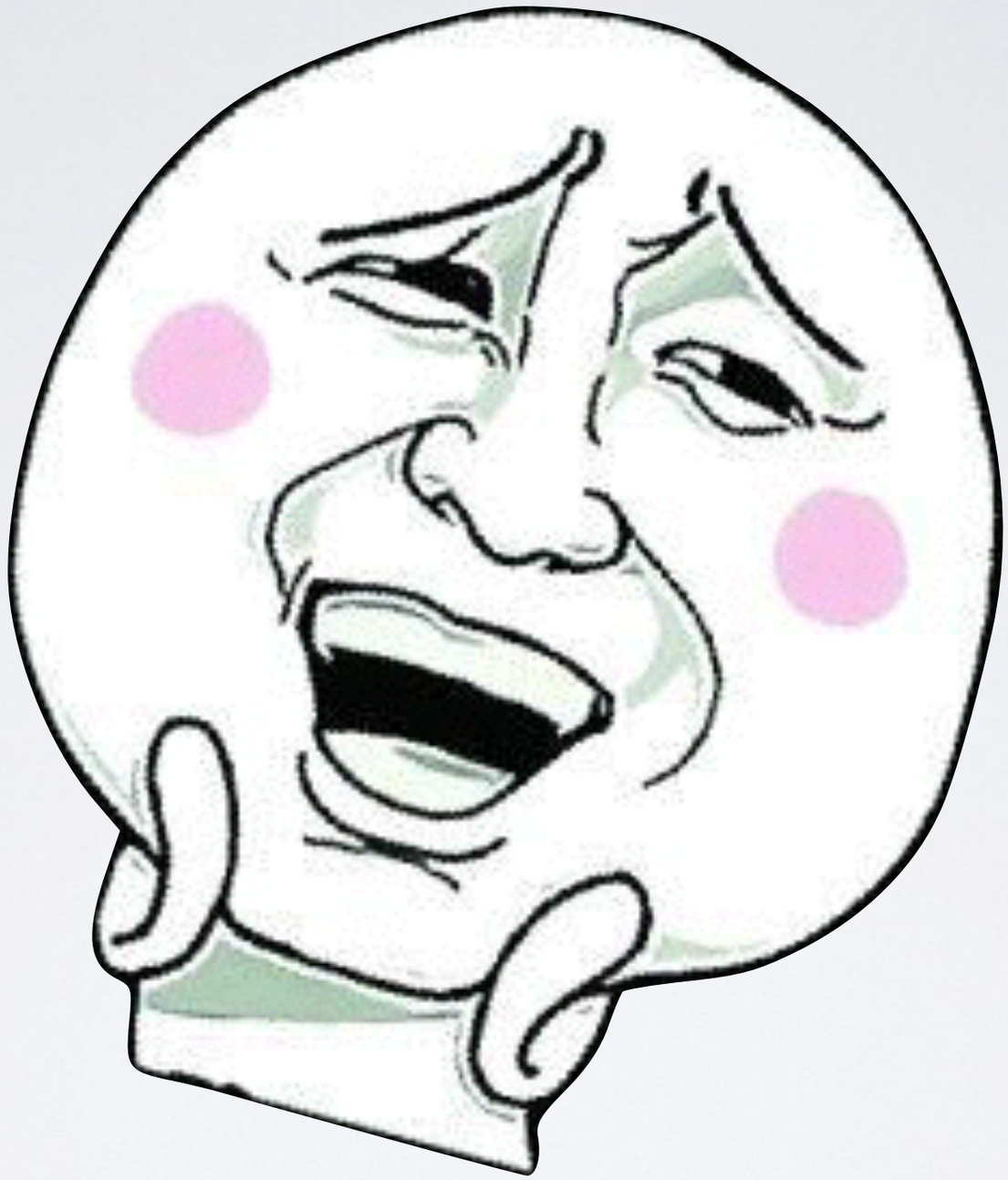


0.3s


异步加载数据
全屏数据直出
首屏优化直出
udp优化支出
22
16.5
11
5.5
0
2.2s
1.7s
1s(首屏)
0.8s(首屏)
页面加载时间

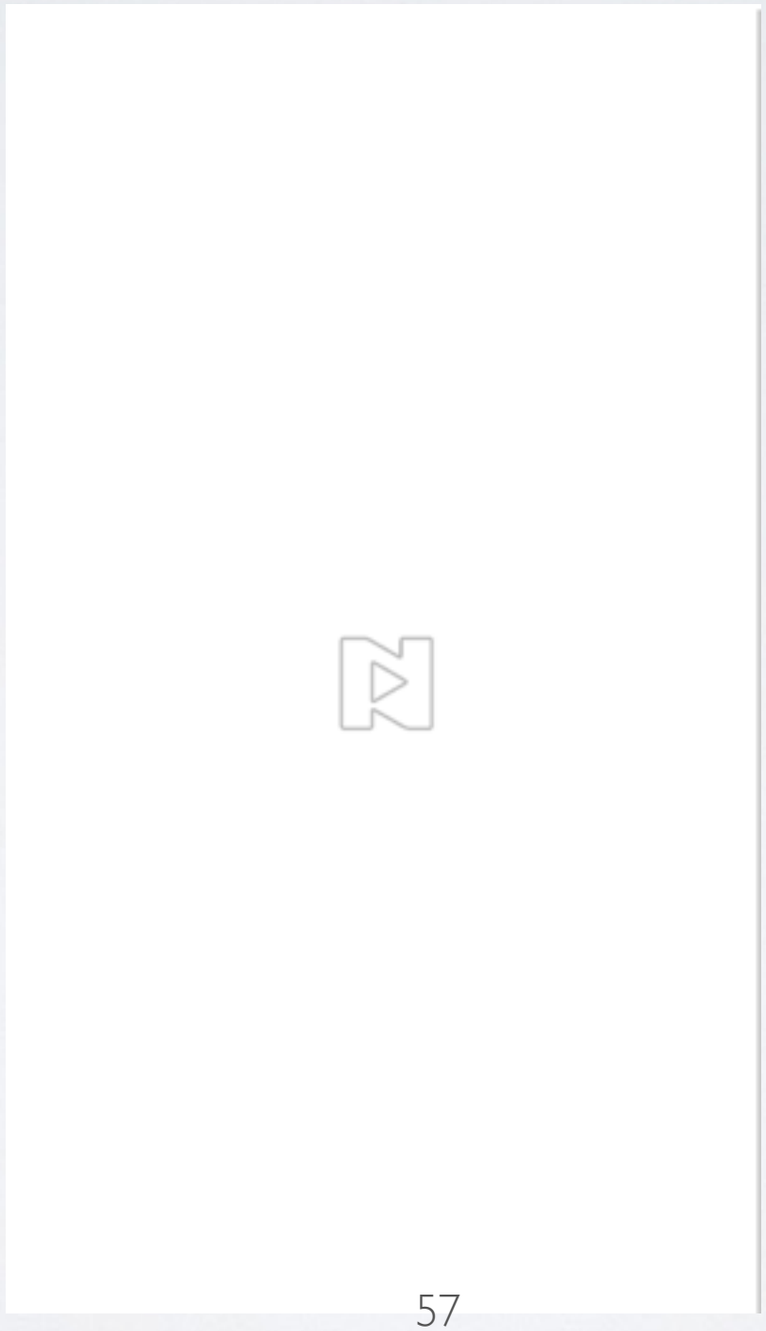
优化方案

- 页面首屏直出，其他使用异步加载
- 减少RTT请求
- 使用UDP协议，节省字节，加快数据的返回
- 不常变更的数据接入cache

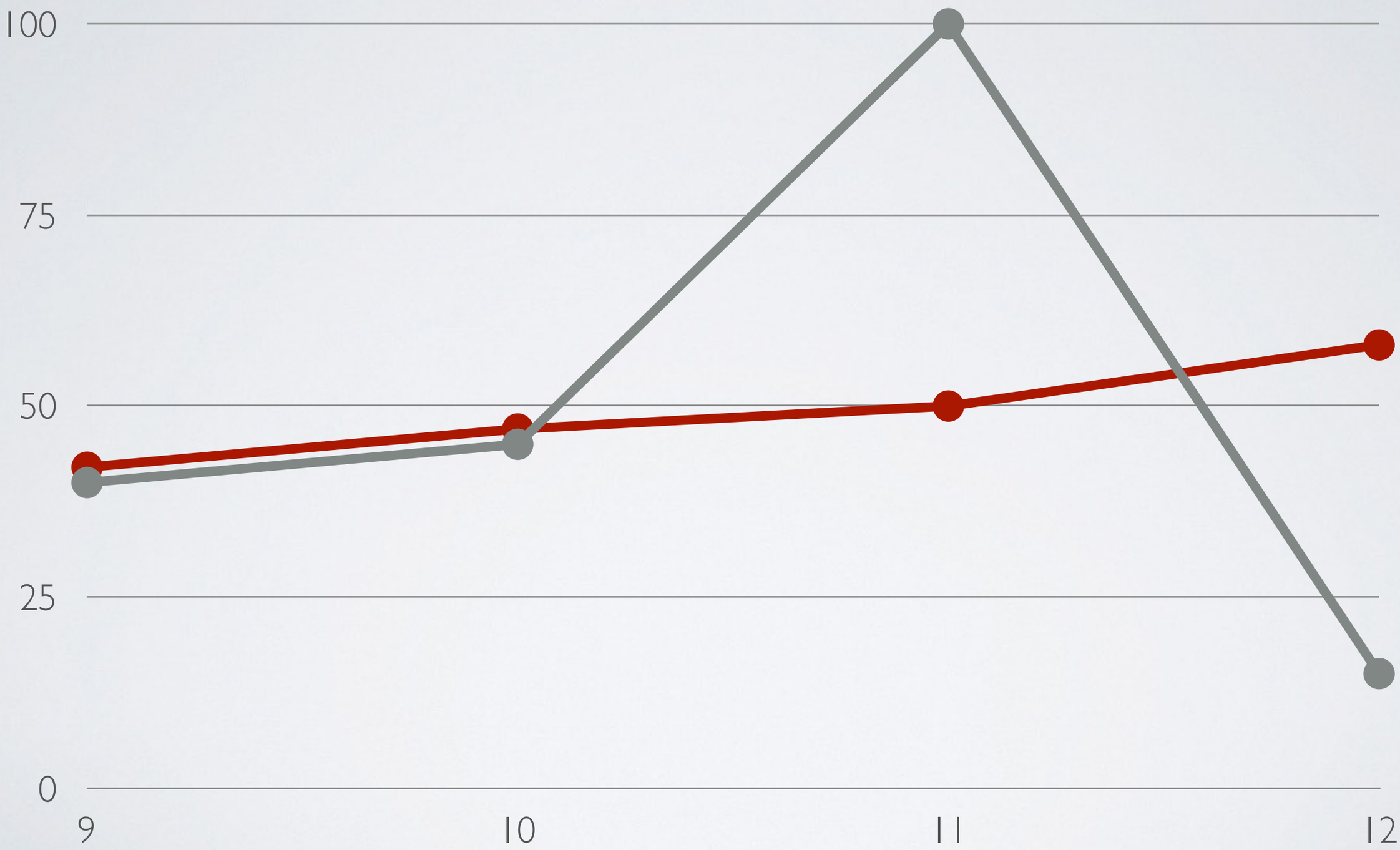



2天后有个活动，PCU大概是现在的2.5倍
产品经理
前端
UI
没问题
没问题
没问题
后台

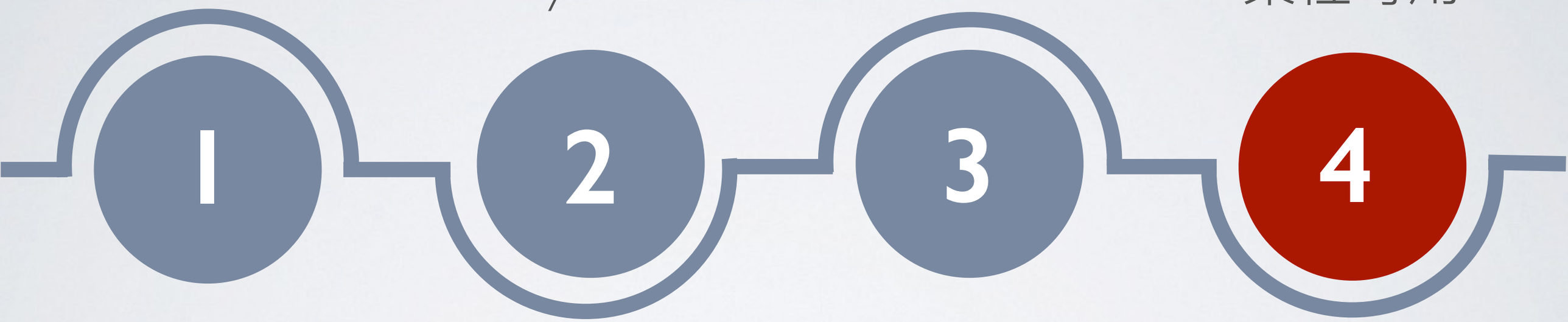
直到活动上线那一天



shenmegui



CONTENTS



时代变迁

why react

性能优化

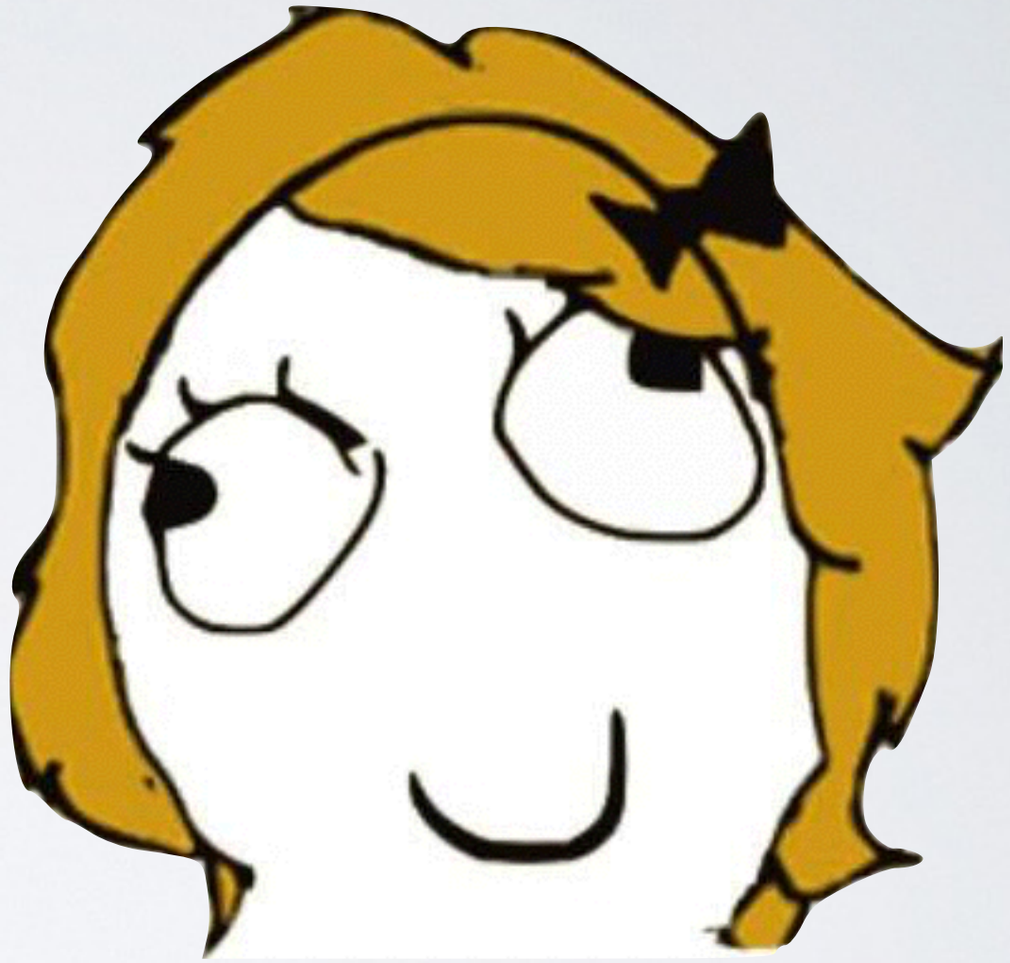
柔性可用

服务器压力

柔性可用


年糕呀
132人
+ 关注
魅力 727272
one
送一个爱你哦
×1
聪明的我
送一个爱你哦
×1
一个人世界有没有陪：那么厉害
呀惹初♡ 来了
xuer：cf，dnf
陌浅离 来了
纯属“虚构”：你说的我都成大神了
聪明的我 送了 x 1个

看美女

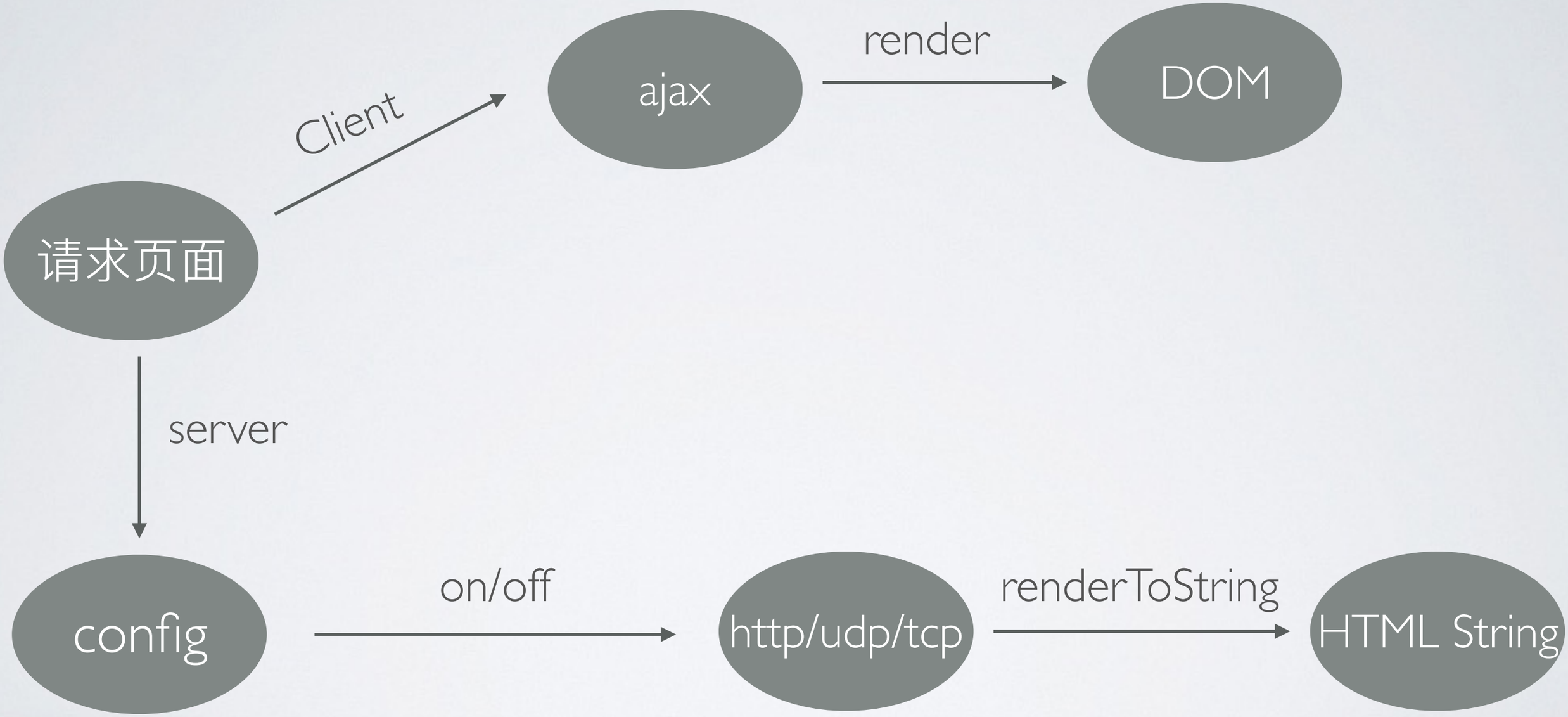


人肉
非常粗暴

组件划分

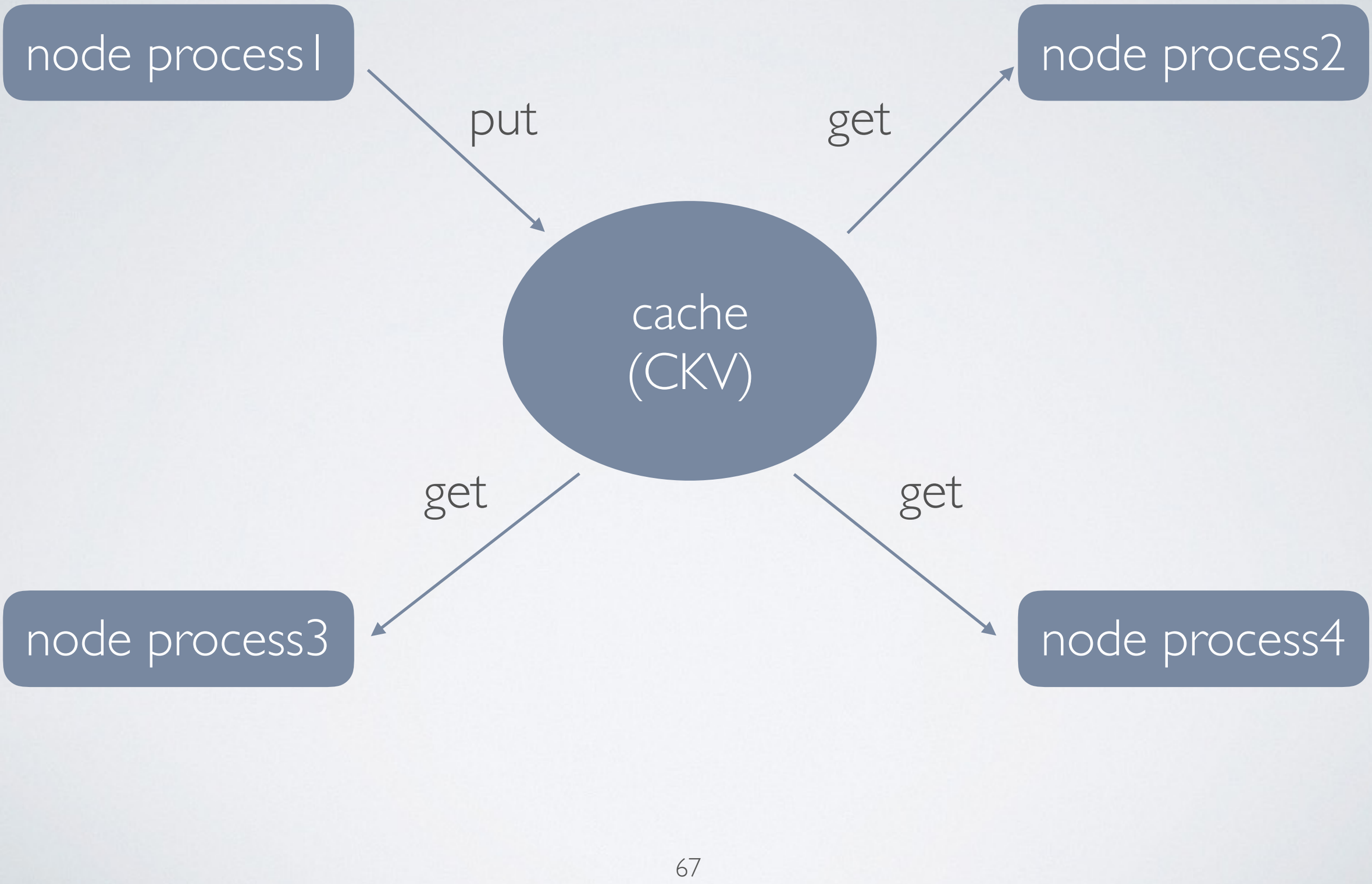
- 主播头像
- 用户列表
- 魅力值
- 聊天信息
- 。。。

config配置组件开关

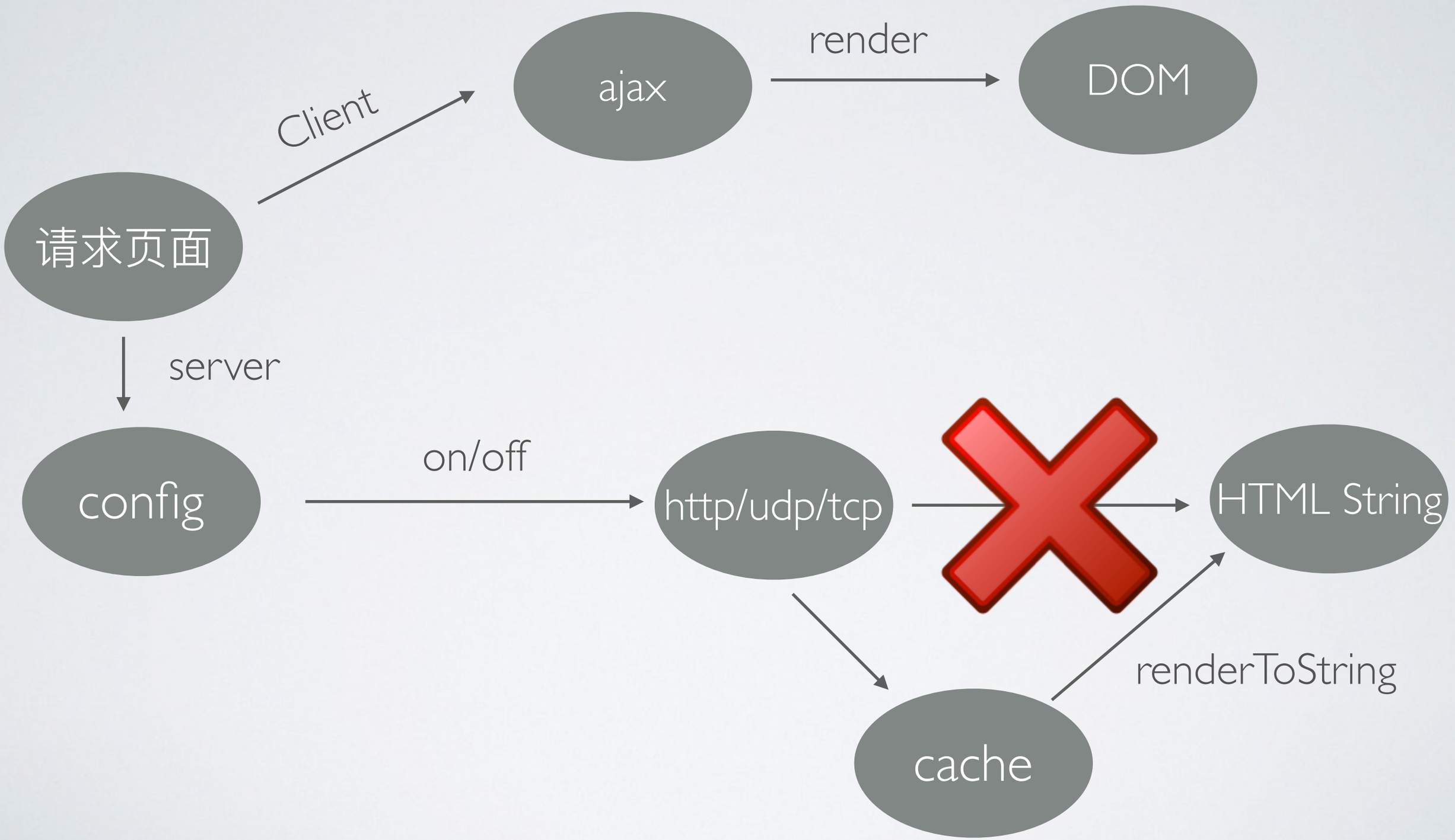


Cache

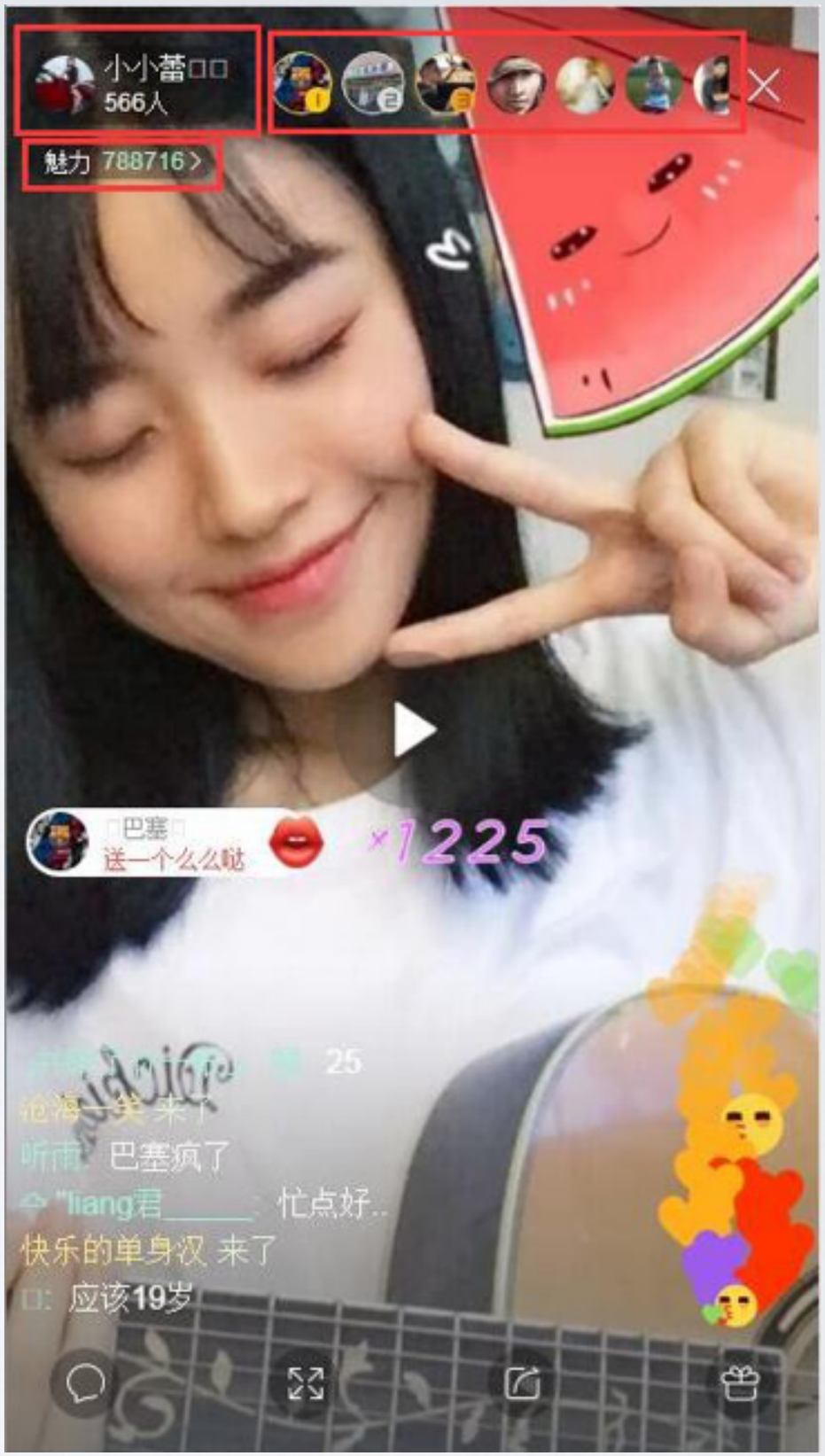
定时服务


node process1
put
get
node process2
cache
(CKV)
get
get
node process3
node process4

一级缓存，首屏数据都接入cache(ckv)

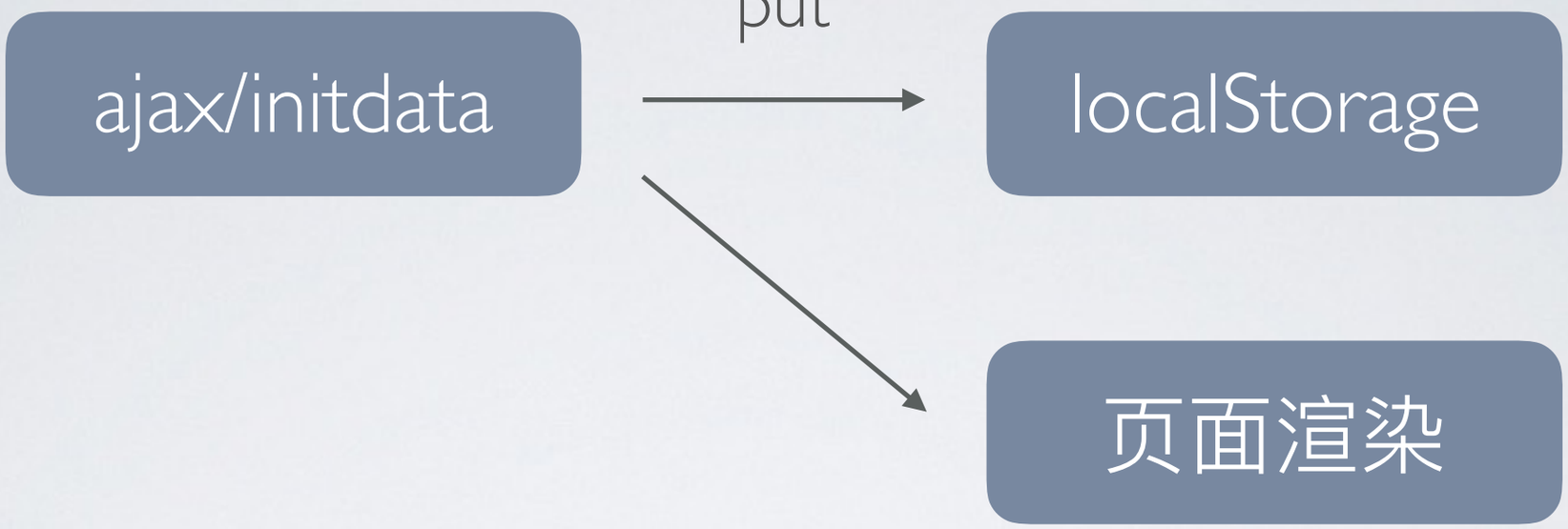


- 头像
- 用户列表
- 魅力值

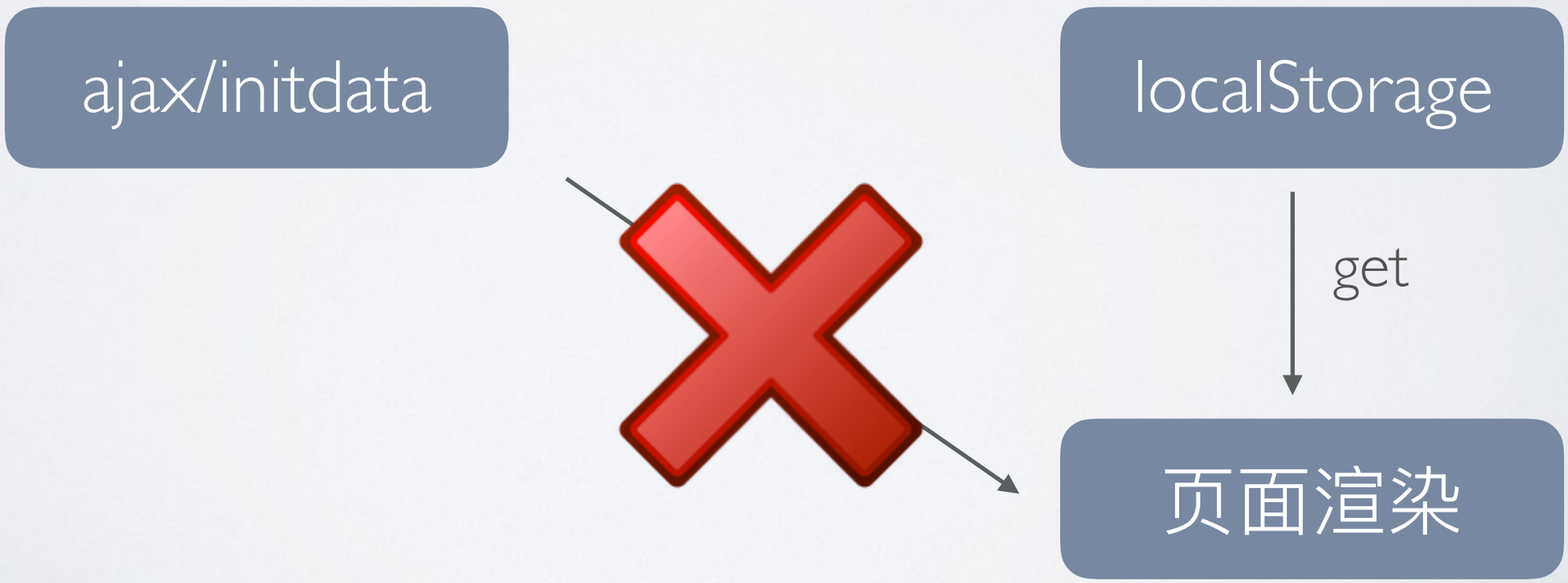


二级缓存，页面接口都缓存到localStorage

场景1



场景2

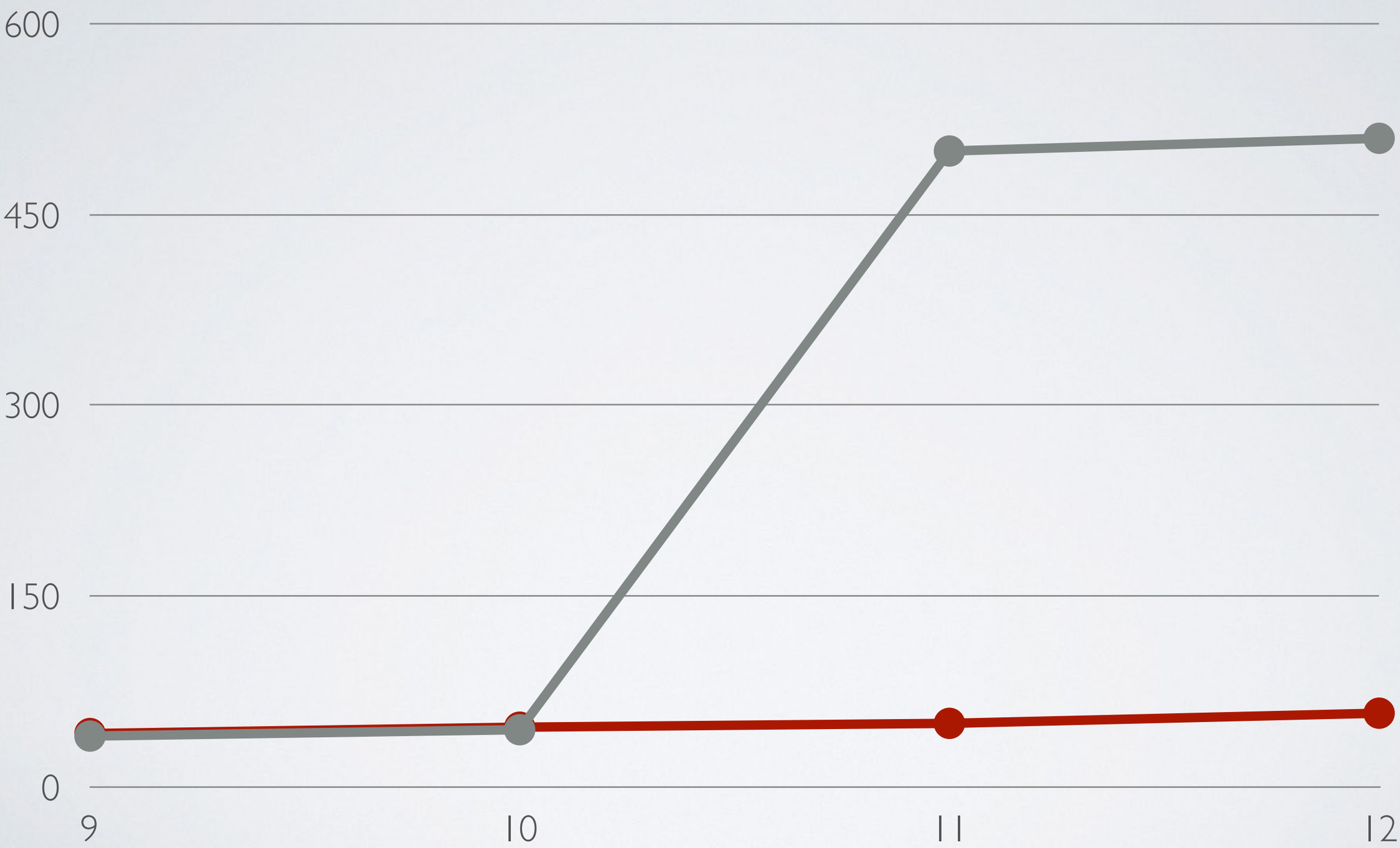


柔性服务

- 组件开关可配置
- 多级缓存，cache(ckv)缓存，本地缓存，默认数据
- 。。。


2天后有个活动，PCU大概是现在的10倍
产品经理
前端
UI
没问题
没问题
没问题
后台

完美



Thank You