面向亿万级用户的Web同构直出

腾讯AlloyTeam 李强


首屏测速数据太差了，做直出吧！
好

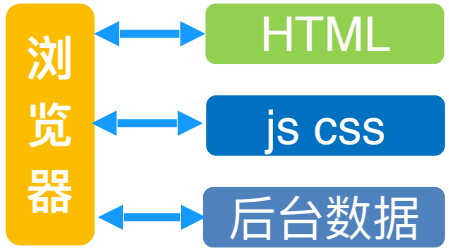
什么是 直出?

什么是直出?

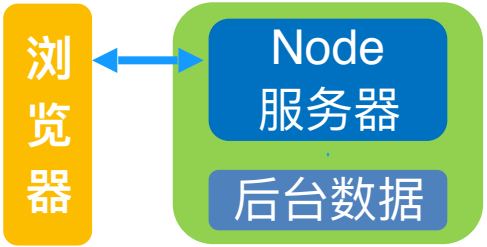
以 **Node** 作为后端语言实现的

服务器端渲染页面并输出 的技术

非直出首屏:



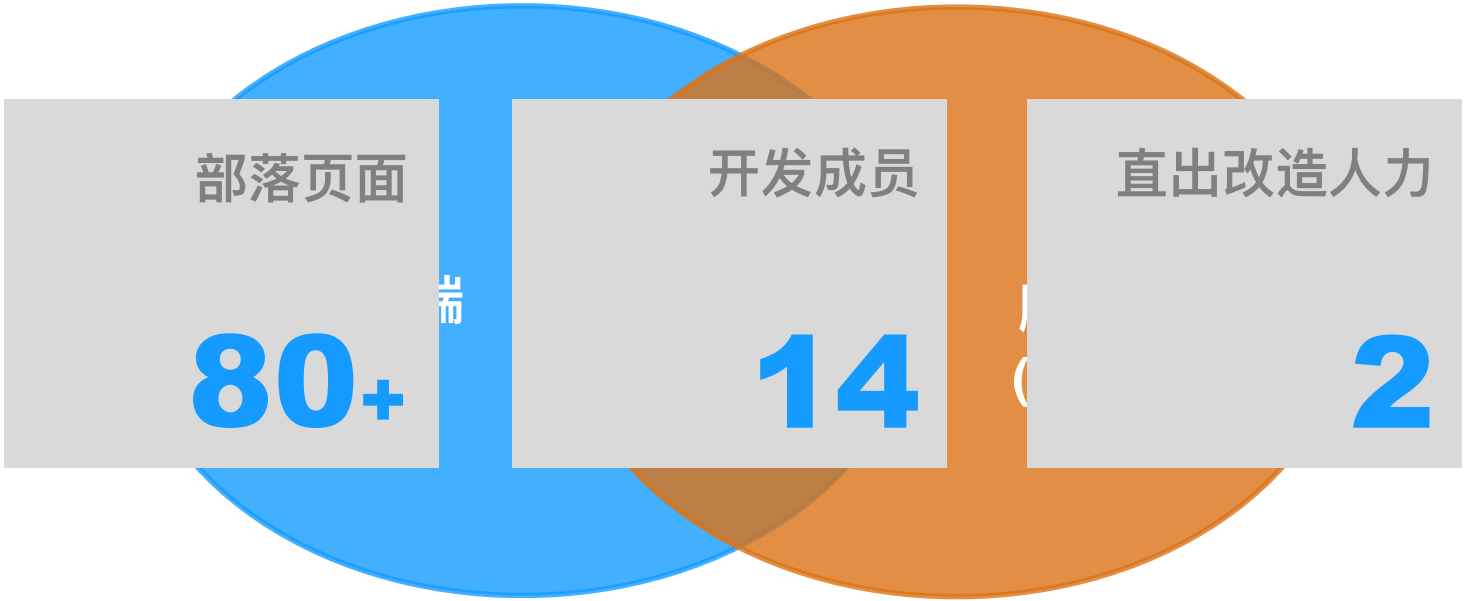
直出首屏:



什么是 同构?

什么是同构?

客户端与服务端可以 **共享** 部分代码

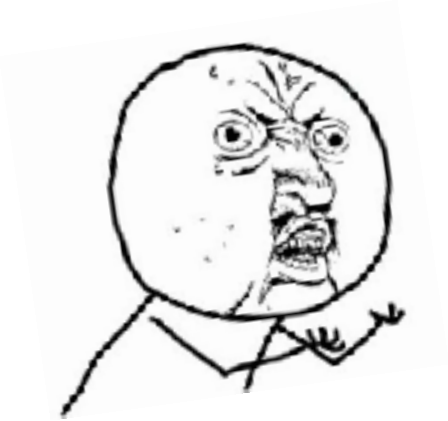


亿万级用户 意味着什么?

注册用户	日均访问量	峰值并发用户
5千万+	5亿+	(十万)10^5

亿万级用户意味着什么?

服务随时可能 **挂掉**



游戏-大家一起来找茬

直出

/mobile/v2/detail.html

非直出

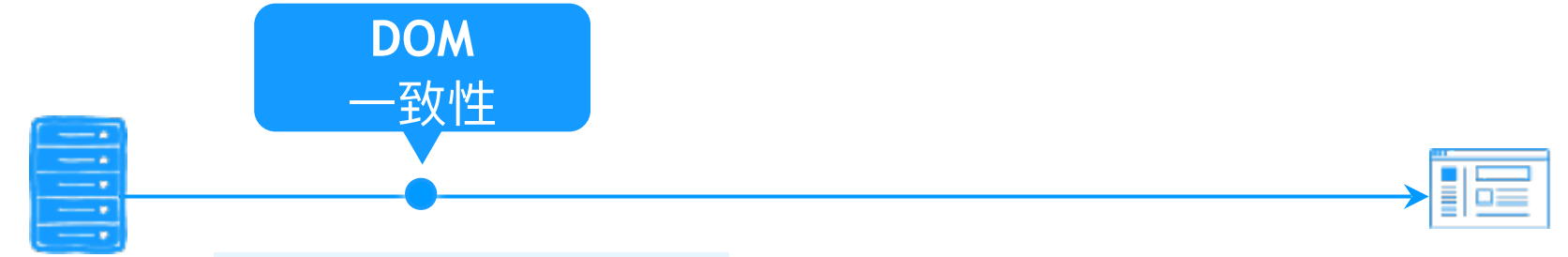
/mobile/detail.html

目录

同构直出

高可用性

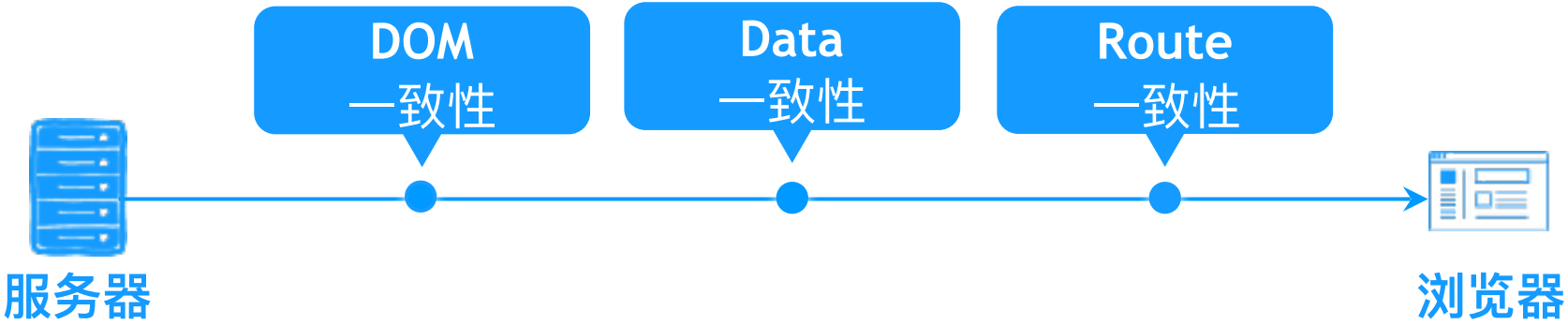
同构直出-三要素




```
<!DOCTYPE html>
<html>
<head></head>
<body>
<div id="root">


</div>
</body>
```

```
<!DOCTYPE html>
<html>
<head></head>
<body>
<div id="root">
<div>
    ... ...
</div>
</div>
</body>
```

同构直出-三要素




```
dom.addEventListener('click', function(data){});
```

同构直出-实现

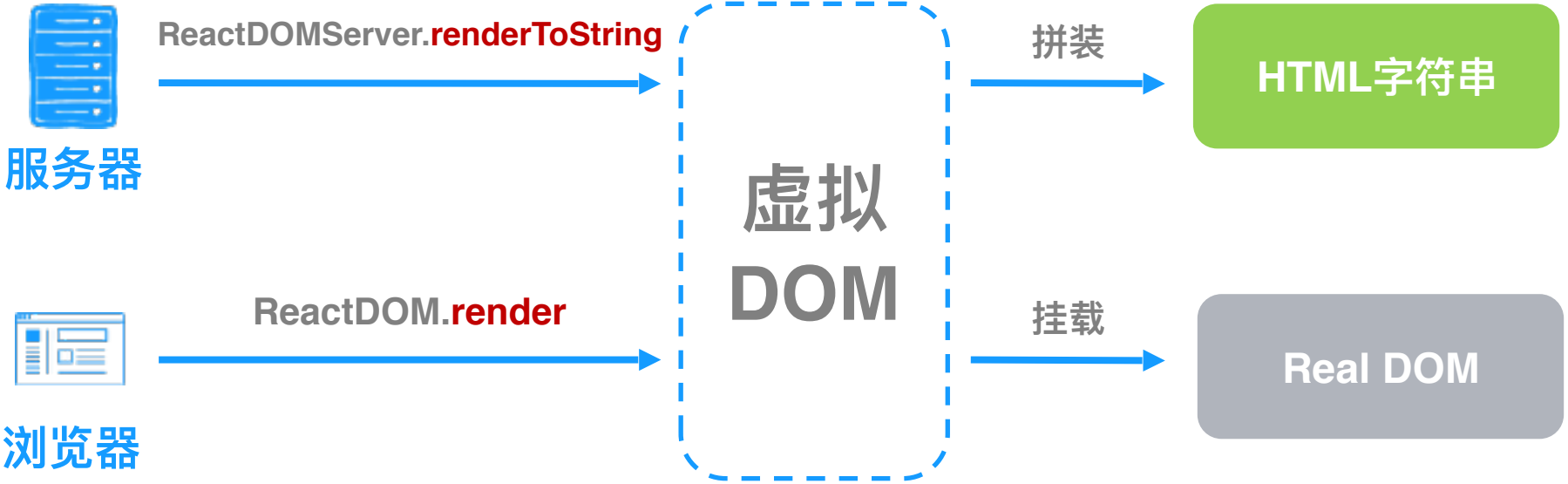
React

DOM 一致性

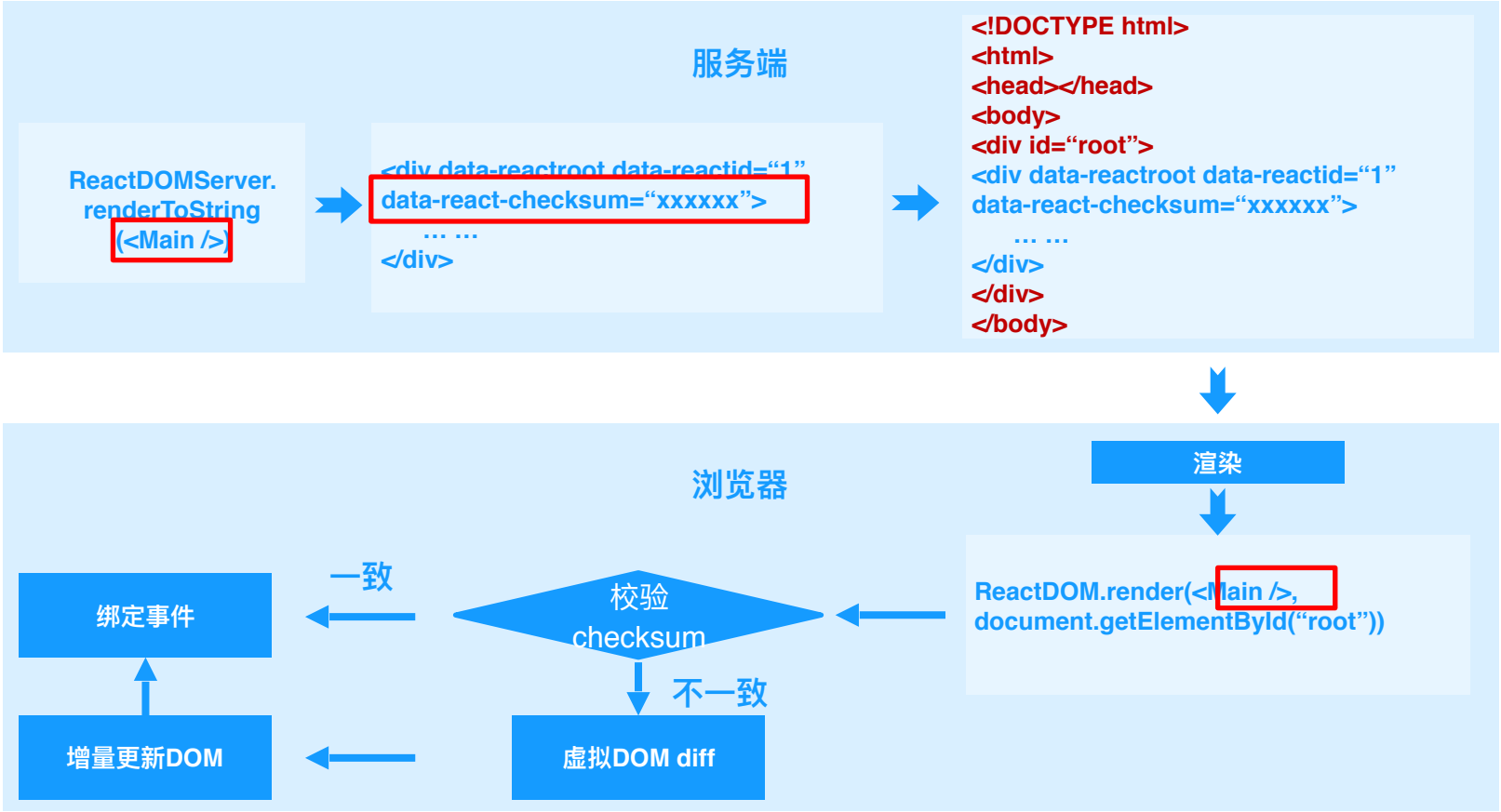
Redux

DATA 一致性

同构直出-DOM一致性-原理



同构直出-DOM一致性-流程图



同构直出-DOM一致性-问题

require.ensure

Window 对象

require('xxx.sass');

require('xxx.less');

require('xxx.css');

ajax

fetch

import‘xxx.gif’

import‘xxx.png’

import‘xxx.jpg’

... ...

如何保证源码同构，做好平台 区分?

同构直出-实现

React

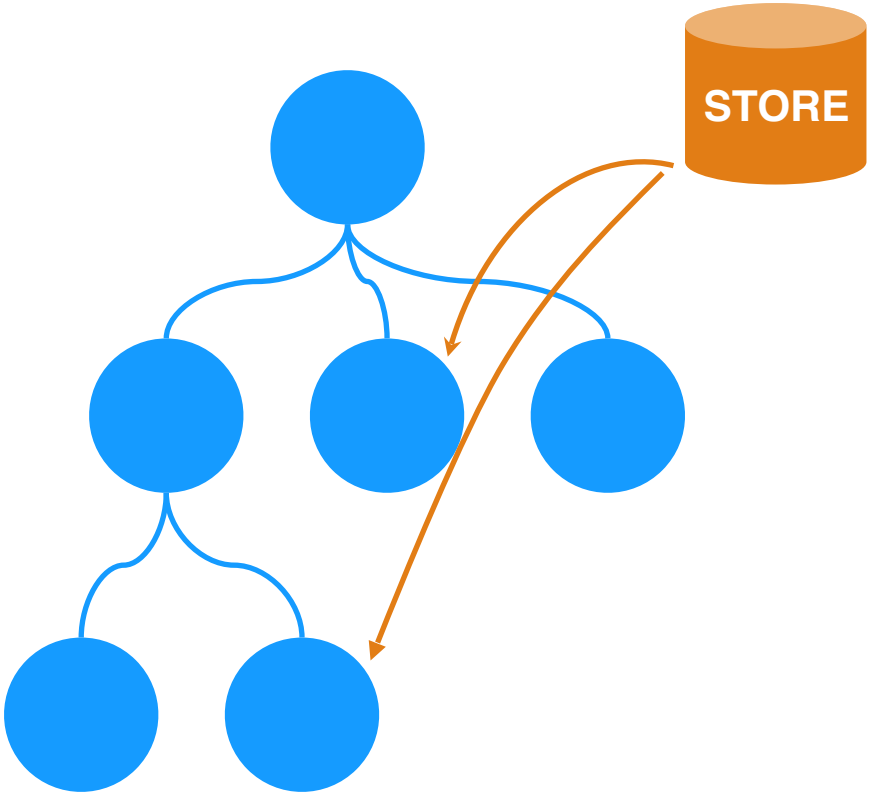
DOM 一致性

Redux

DATA 一致性

使用单一的 Store 对象

保存页面所有状态



同构直出-Data一致性-流程图

服务端

```
... ...
<script>
   window.__INITIAL_STATE=JSON.stringify(store.getState())
</script>
... ...
```

浏览器

```
const store = createStore(reducers, window.__INITIAL_STATE || {} );

ReactDOM.render(
   <Provider store={store}>
      <Main />
   </Provider>
)
```

同构直出-Data一致性-问题

浏览器使用 **ajax**

Node端使用 **http.request**

怎样保证源码 **同构?**

同构直出 解决方案

同构直出-方案

户型结构图
（源码）

设计师
（构建）

设计效果图
（bundle）

房1(已装修)
（浏览器）

房2(待装修)
（Node服务器）

同构直出-方案-不能在服务端执行的前端代码

户型结构图

设计师

同构直出-方案-不能在服务端执行的前端代码

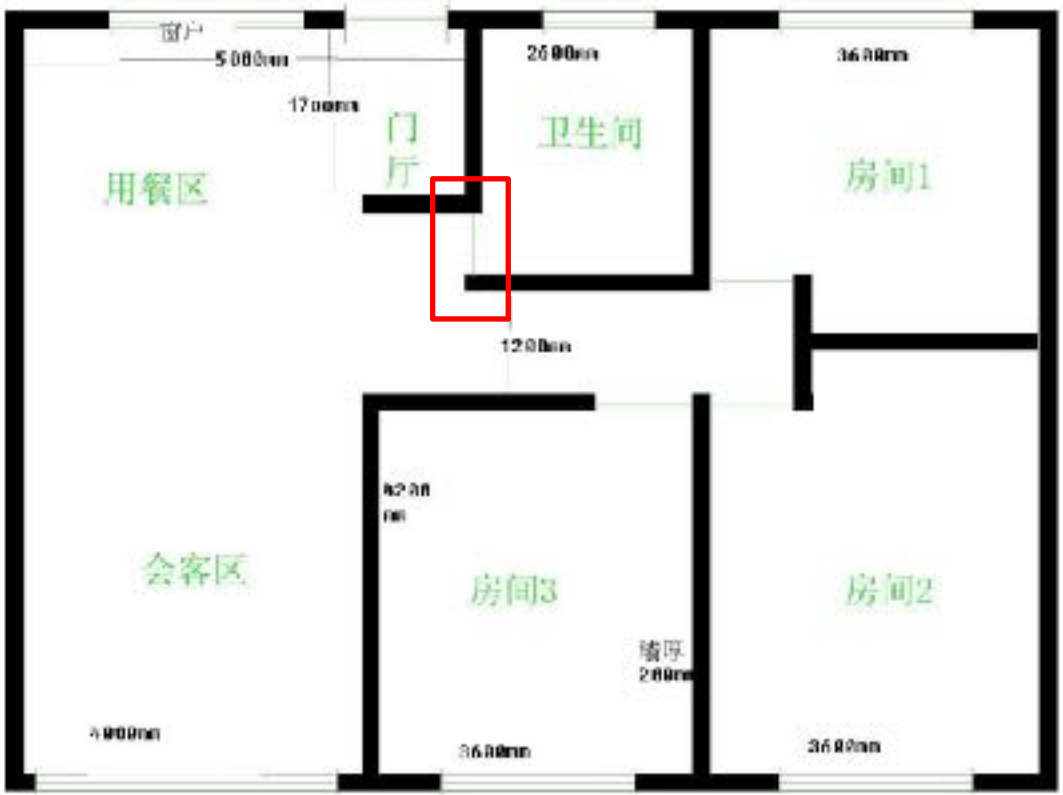
```
... ... // 两端都需执行的代码块

'__BROWSER__';

    ... ... // 不在Node端运行的代码块

'__END__';

... ... // 两端都需执行的代码块
```

源码

Loader
或
babel插件

Webpack
打包构建

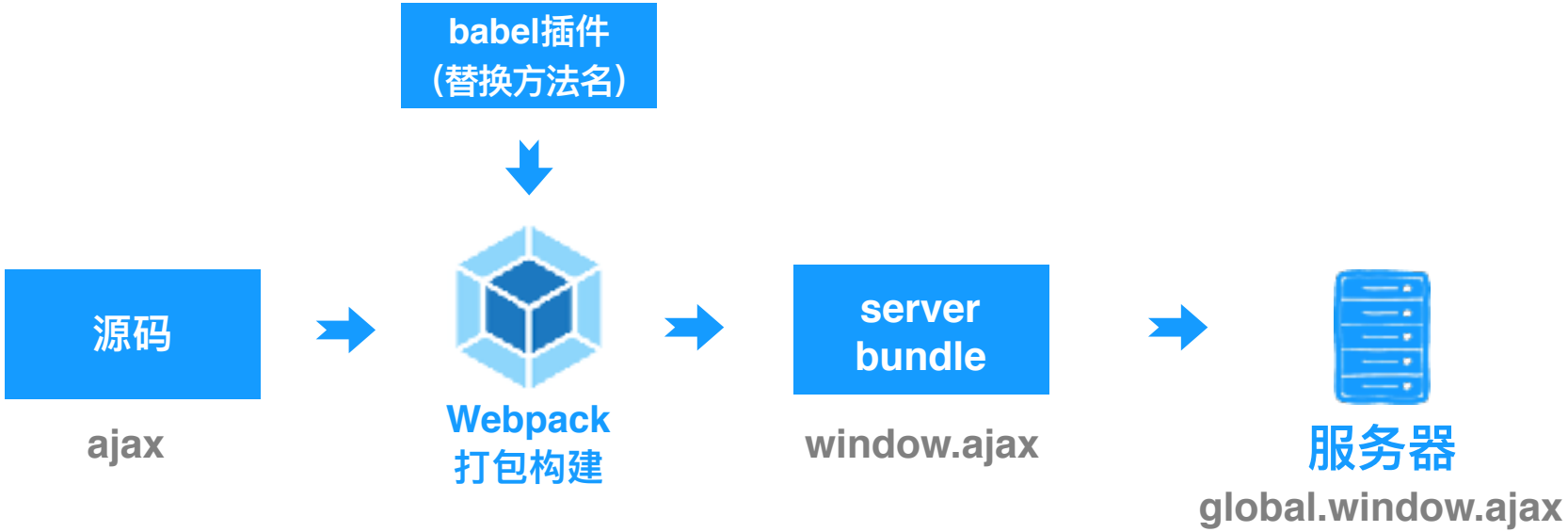
同构直出-方案-ajax方法替换



户型结构图

设计师

同构直出-方案-ajax方法替换



同构直出-方案

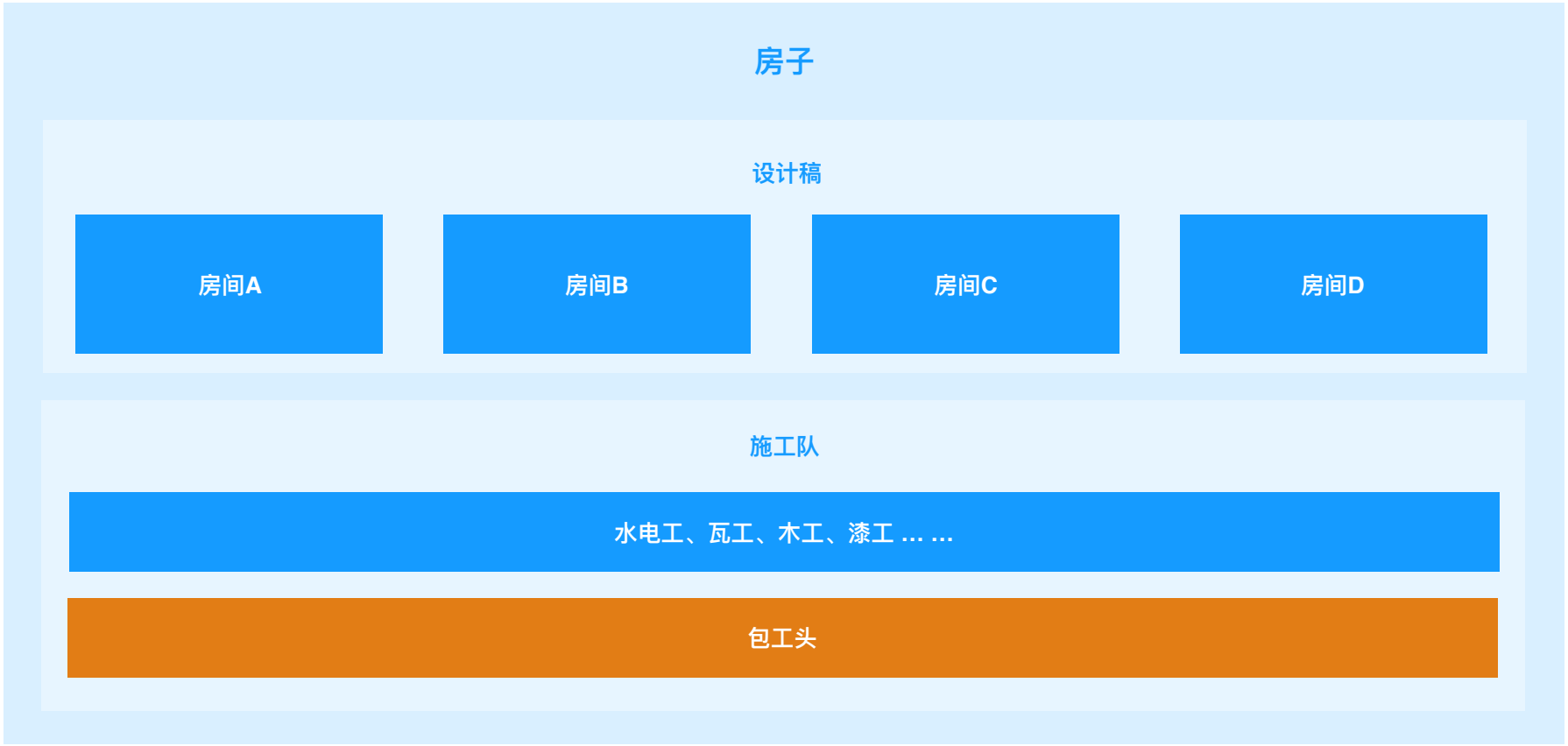
户型结构图
（源码）

设计师
（构建）

设计效果图
（server bundle）

房2（待装修）
（Node服务器）

同构直出-方案-框架机



同构直出-方案-框架机

同构直出-方案-构建

```
page
└── a                    # A页面源代码开发目录
    ├── ... ...          # 其它
    ├── action           # html模板
    │   ├── actions.js   # action生成函数
    │   └── firstAction.js  # 首屏action(仅服务端)
    ├── Main
    │   └── index.js     # 组件入口
    ├── store.js         # 创建store
    ├── index.jsx        # 客户端程序打包入口
    └── a.html           # html模板
```

源码结构

```
import ReactDOM from 'react-dom';
import { Provider, connect } from 'react-redux';

import store from './store';

import Main from './Main';

let ConnectedMain = connect(function (state) {
    return state;
})(Main);

ReactDOM.render(
    <Provider store={store}>
        <ConnectedMain />
    </Provider>,
    document.getElementById('root')
);
```

客户端程序打包入口

同构直出-方案-首屏actions提取

```
... ...
export let todo1 = data => async(dispatch) => {
    let res = await ajax({
        url: 'xxx',
        data: data
    });
    dispatch({
        type: 'TODO1_SUCCESS',
        data: res
    });
};
// todo2
// todo3
// todo4
... ...
```

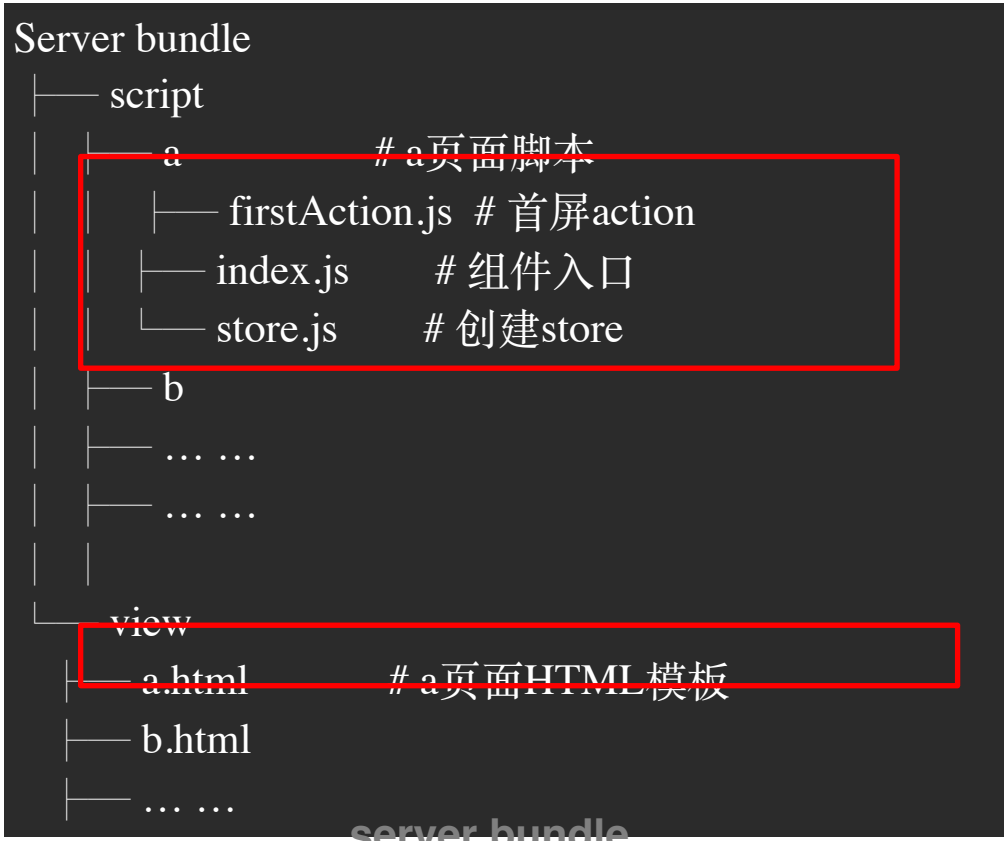
actions.js

```
// firstAction.js

import {todo1, todo2} from ‘./actions’

export default () => async(dispatch) => {
    await Promise.all([
        dispatch(todo1());
        dispatch(todo2());
    ]);
}
```

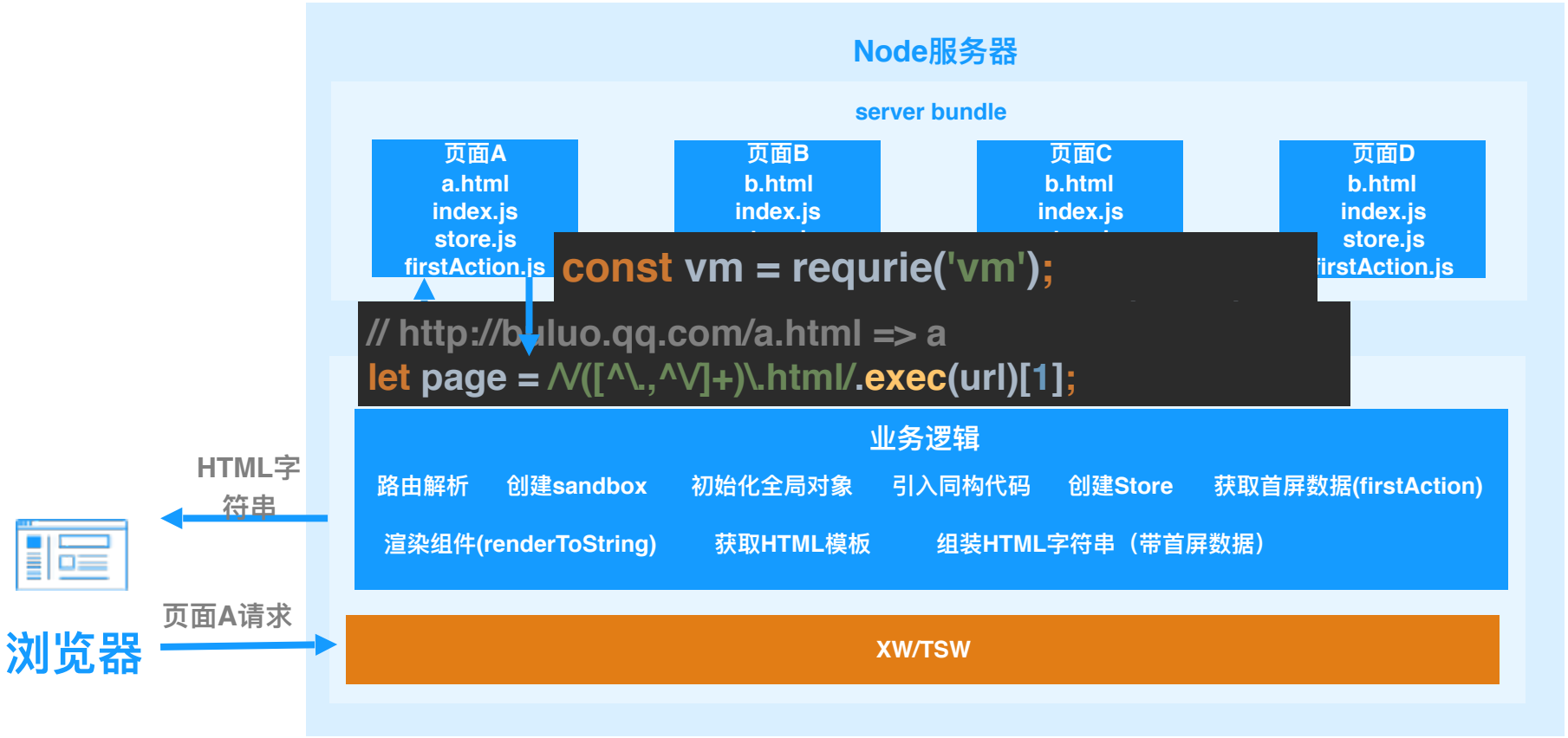
todo4

firstAction.js

同构直出-方案-构建


Server bundle
script
a # a页面脚本
firstAction.js # 首屏action
index.js # 组件入口
store.js # 创建store
b
… …
… …
view
a.html # a页面HTML模板
b.html
… …
server bundle

同构直出-方案-框架机工作流



同构直出-方案总结

```
... ...
'__BROWSER__';
    ... ...

'__END__';
... ...
```

源码

```
import {todo1, todo2} from './actions'

export default () => async(dispatch) => {
    await Promise.all([
        dispatch(todo1);
        dispatch(todo2);
    ]);
}
```

firstAction.js

Loader
或
babel插件

Webpack
打包构建

server
bundle

client
bundle

Node服务端

server
bundle

直出
框架机

浏览器

client
bundle

渲染HTML

目录

同构直出

高可用性


王者荣耀
最高同时在线人数100万

高可用性-目录

开发调试

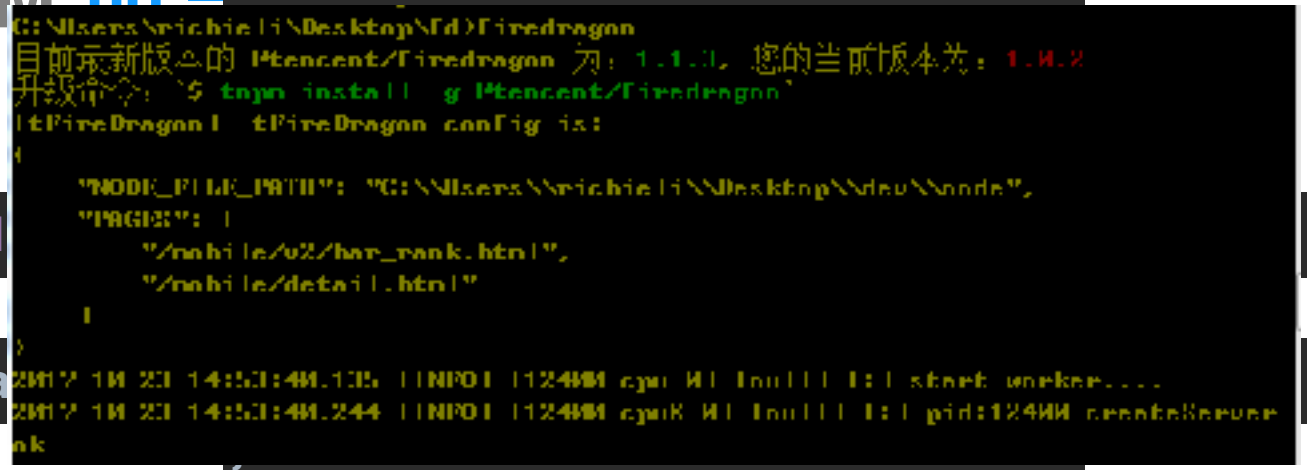
上战场前先保证 胜率


己方英雄
升级
重置等级
冷却
无敌
+1000
黑暗暴君buff
敌方英雄
升级
重置等级
AI
无敌
+1000
黑暗暴君buff
通用
兵线
野怪
暴君
黑暗暴君
主宰
主宰先锋荣誉
295
4 VS 1
13
35

高可用性-开发调试-框架机本地开发调试

NPM 命令行工具包：fireDragon

```
tnpm I
```




```
firedra
```

```
firedragon // 读取路由配置文件，启动服务
```

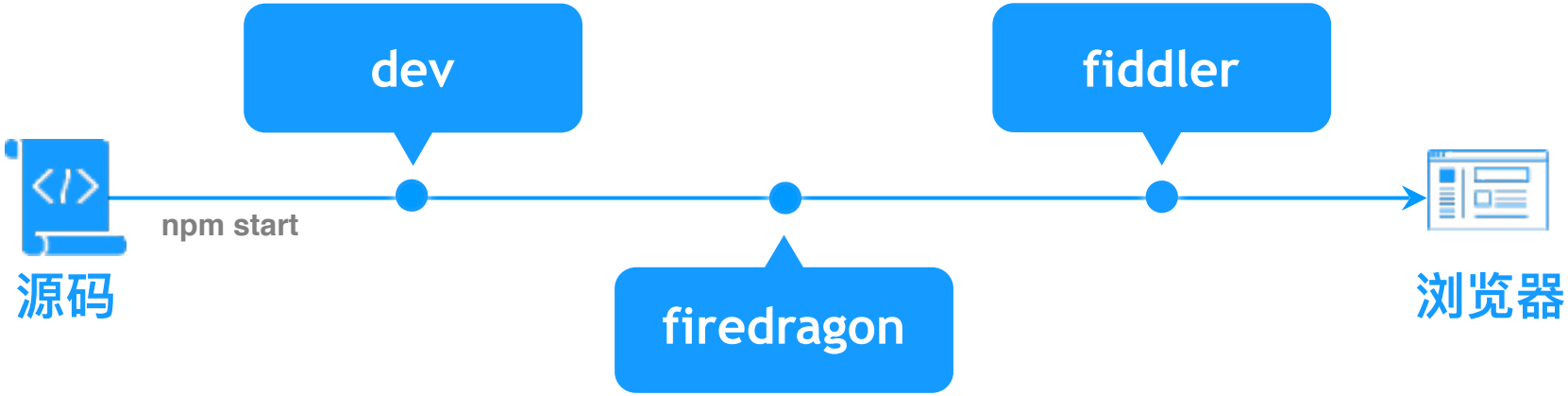
高可用性-开发调试-直出代码本地开发调试

```
tnpm install -g @tencent/firedragon // 安装命令
```

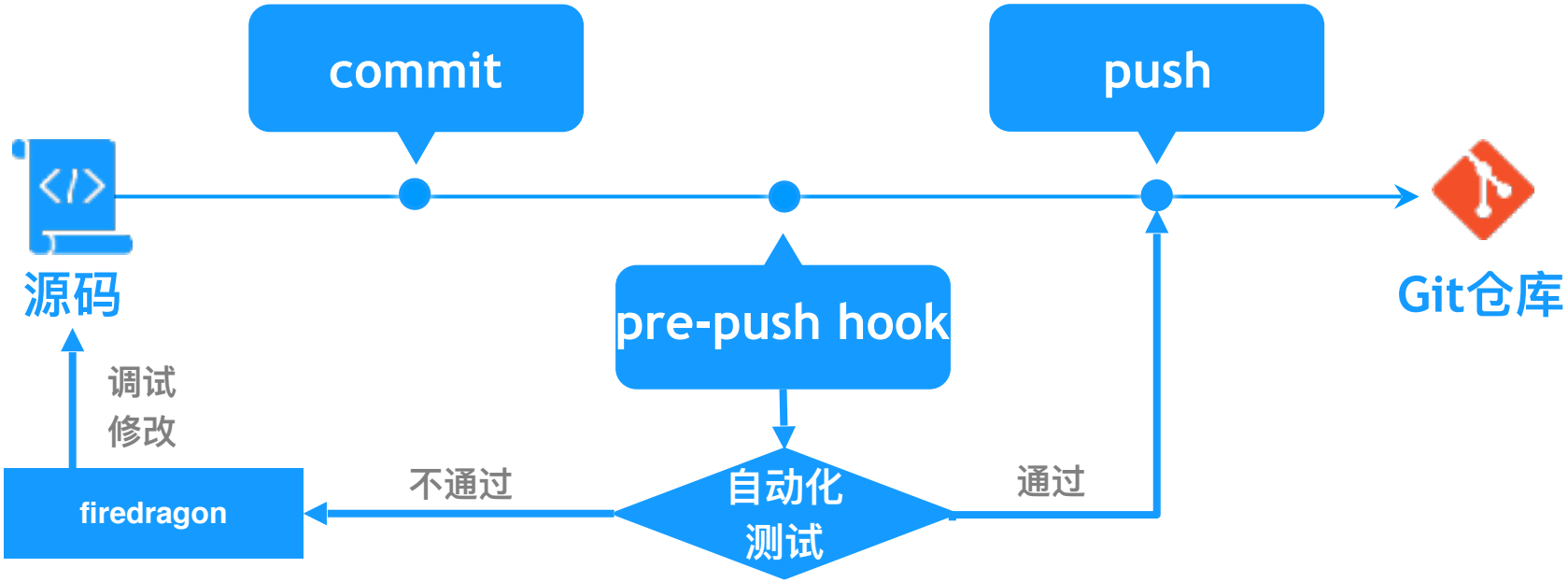
```
firedragon init // 初始化路由配置
```

```
firedragon // 读取路由配置文件，启动服务
```

高可用性-开发调试-流程



高可用性-开发调试-自动化测试



高可用性-开发调试-自动化测试

测试框架 Mocha

断言库 Chai

```
mocha index.js
```

```
const {assert, expect} = require('chai');
const request = require('request');
var r = request.defaults({
    'proxy': 'http://127.0.0.1:3000',
    headers: {
        xxx: xxx
    }
});
... ...
startFireDragon();
... ...
describe('自动化直出检查: ' + item , function () {
    testing.map(test => {
        it('渲染页面测试' + test.name, function(done){
            var selector = test.selector;
            r(item, function (error, response, body) {
                expect(error).to.be.null;
                assert.equal(200, response && response.statusCode);
                expect(body.indexOf(selector) > -1).to.be.true;
                done();
            });
        });
    });
});
```

启动firedragon

发起页面请求

返回状态码

页面关键元素

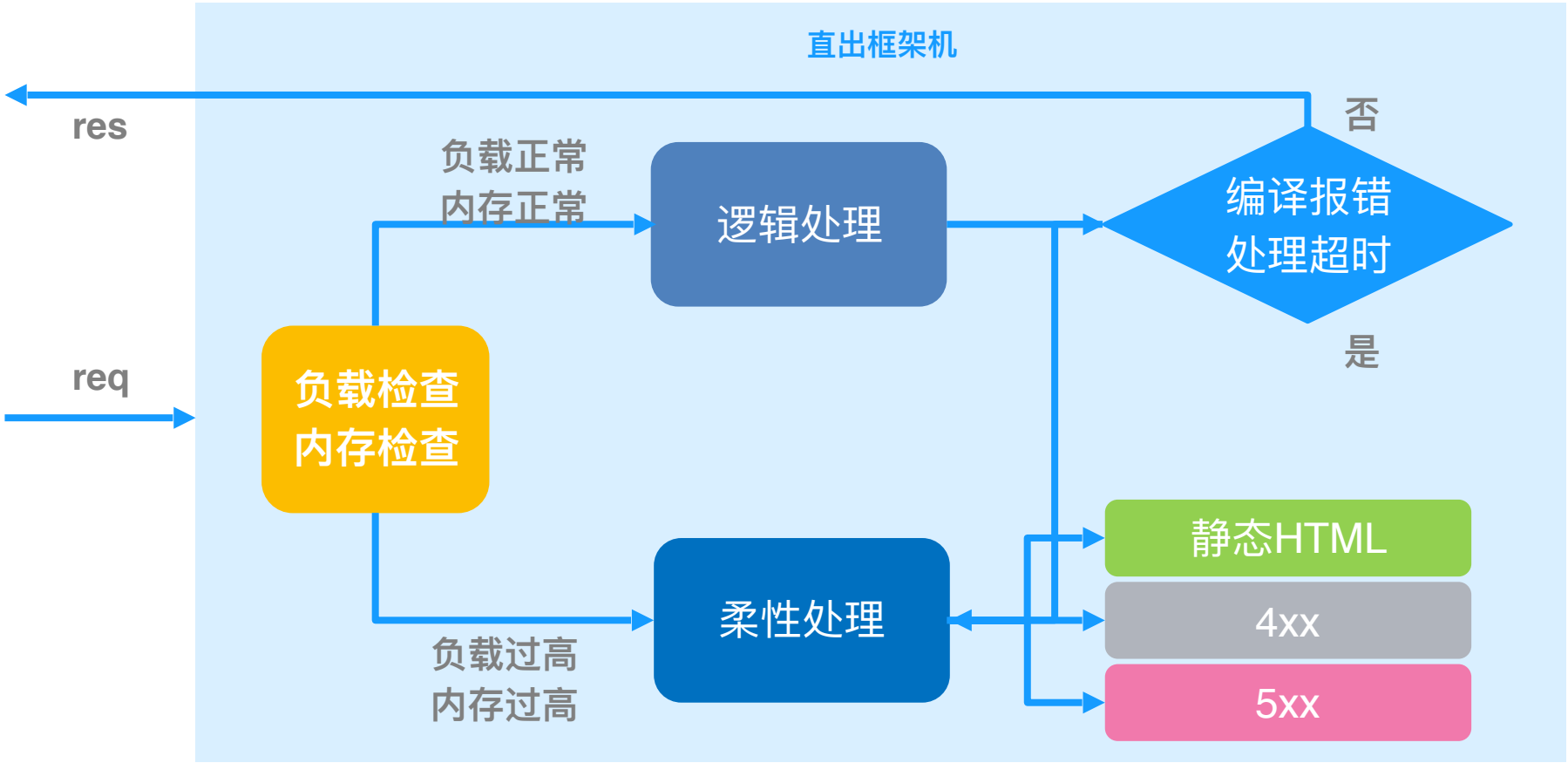
高可用性-目录

开发调试

容灾

阵容合理 取长补短 才不容易失败

高可用性-容灾-框架机

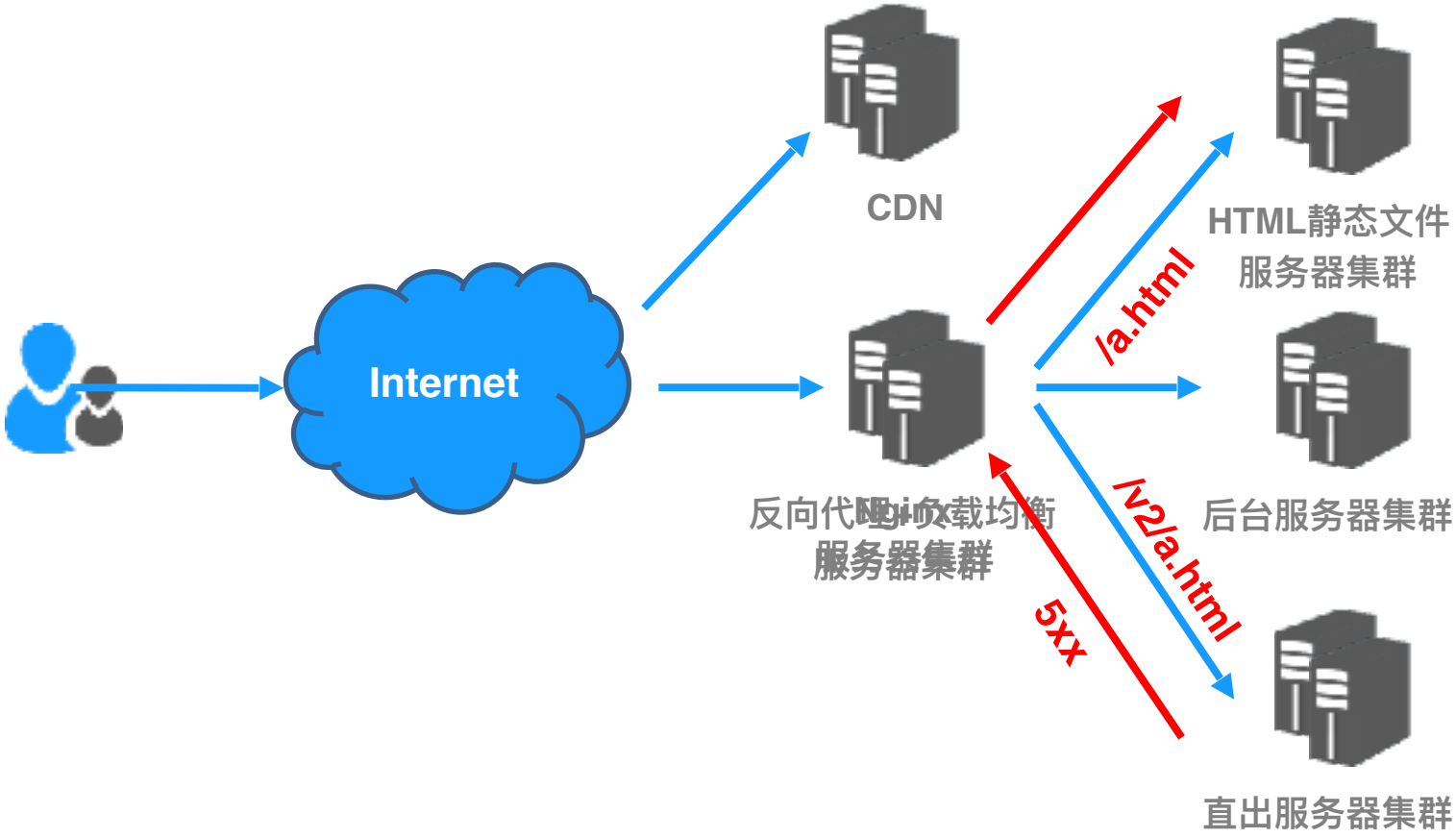

直出框架机
res
req
负载正常
内存正常
逻辑处理
否
编译报错
处理超时
是
负载检查
内存检查
静态HTML
4xx
5xx
柔性处理
负载过高
内存过高

直出服务挂掉 页面还可以正常使用

信不信?

高可用性-容灾-接入层

高可用性-容灾-接入层


CDN
HTML静态文件
服务器集群
Internet
/a.html
反向代理+负载均衡
服务器集群
后台服务器集群
/v2/a.html
5xx
直出服务器集群

接入层

配置实现

```
upstream static_env {
    server ip:port;
}
upstream nodejs_env {
    server ip:port;
}
server {
    listen 80;
    server_name xxx;
    ... ... // 其他配置
    location ^~ /mobile/ {
        proxy_pass http://static_env;
    }
    location ^~ /mobile/v2/ {
        proxy_pass http://nodejs_env;
        proxy_intercept_errors on;
        error_page 403 404 408 500 501 502 503 504 =200 @static_page;
    }
    location @static_page {
        rewrite_log on;
        error_log logs/rewrite.log notice;
        rewrite /mobile/v2/(.*)$ /mobile/$1 last;
    }
}
```

将非v2目录的请求转发到静态HTML文件服务器

将v2目录下的请求转发到Node直出服务器

拦截响应状态码

将这些状态码改为200响应，并指向新规则处理

去掉v2目录后重新定向地址

高可用性-目录

开发调试

容灾

压测

5 打 1 能不能挺得住?

高可用性-压测-目的

了解服务器的 **承受能力**

找出 **性能瓶颈**

有目的的进行开发和运营

高可用性-压测-平台-wetest.qq.com

高可用性-压测-平台-wetest.qq.com

高可用性-压测-平台-wetest.qq.com

高可用性-压测-平台-wetest.qq.com

高可用性-目录

开发调试

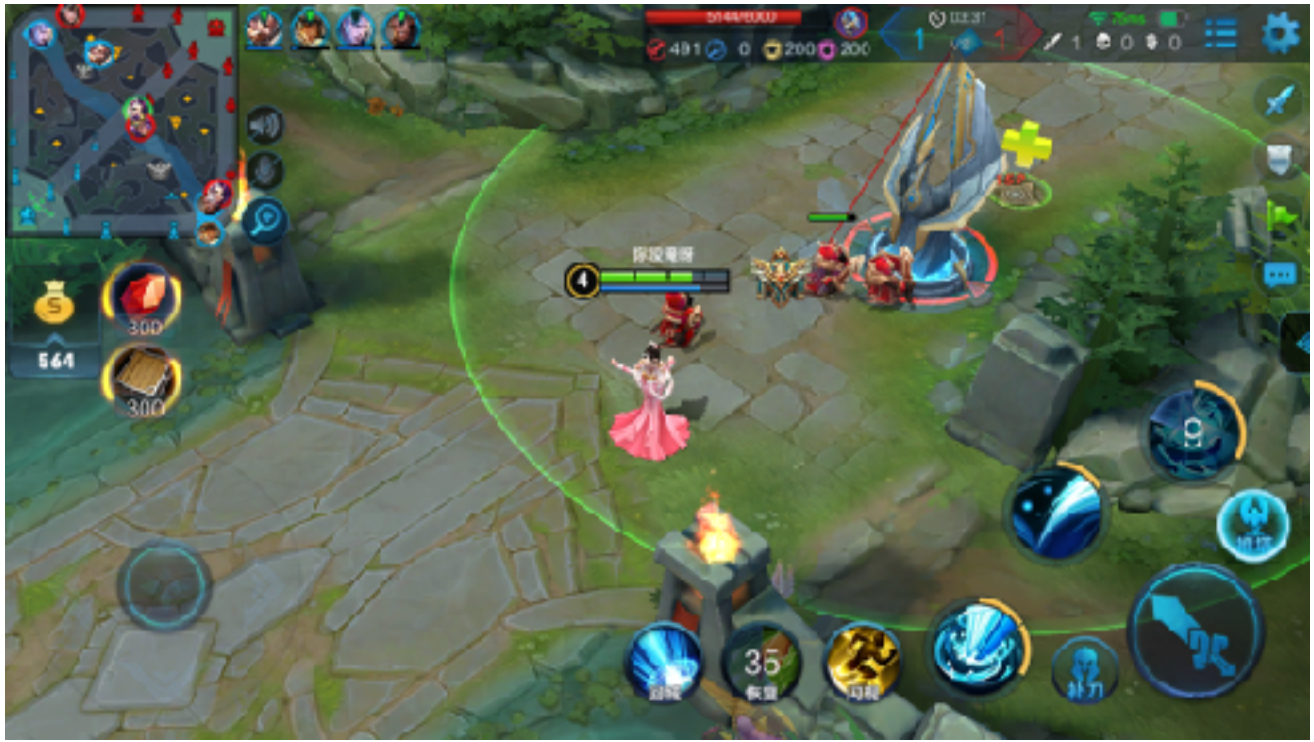
容灾

压测

灰度

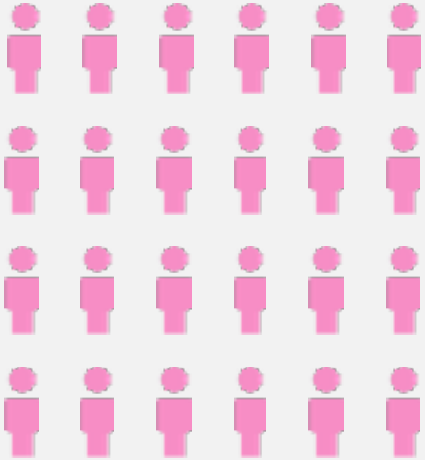
高可用性-灰度

防御塔要逐一击破，直接攻打 水晶 肯定是不可行的



高可用性-灰度

日均访问量



[redacted] 次

服务器

130 台

(虚拟机)

通过前端控制页面入口灰度

直出用户走 v2

高可用性-目录

开发调试

容灾

压测

灰度

监控告警

插眼

发信号

高可用性-监控告警

服务器性能

- ✓ CPU使用率
- ✓ CPU负载
- ✓ 内存占用
- ✓ 磁盘IO
- ✓ 网络IO

运行时数据

- ✓ 脚本错误
- ✓ 测速
- ✓ PV/UV
- ✓ 直出服务质量
- ✓ 后台接口质量
- ✓ 染色日志

高可用性-总结

开发调试

框架机本地开发调试

自动化测试

容灾

框架机柔性处理

接入层容错

压测

压测目的

压测方式

灰度

平滑过渡方式

灰度方式

监控告警

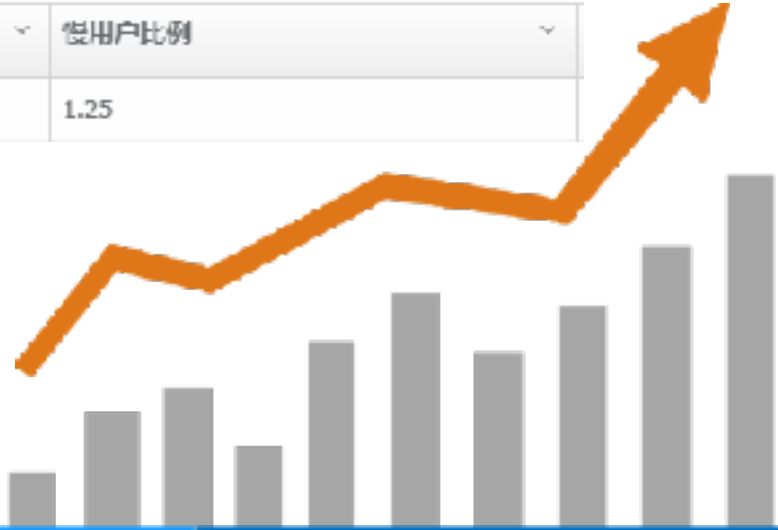
关键监控点

最后-成效

页面首屏测速：点击入口—可交互

平均耗时	使用户比例
2123.53	6.81

平均耗时	使用户比例
948.20	1.25



谢谢

Thanks For All

Q&A

欢迎随时沟通！