如何用Vue构建大型单页面应用

汽车之家 车服务-前端团队 王诗扬

关于我

王诗扬

- 2013年加入汽车之家，现任车服务部 前端负责人
- 致力于公司内Node和vue的推广和布道
- 乐于分享，喜欢研究新技术

技术栈

- 服务端：Node.js (v6.3)
- 前端框架：Vue (v2.1.10)
- 前端构建工具：webpack
- 代码检查：eslint
- 状态管理：Vuex
- 服务端通信：axios
- 进程管理：pm2
- 项目自动化部署：jenkins


上午9:41
100%
途虎养车
京东金融
沪江网校
北京银行
携程旅行
摩拜单车
乐视视频
铂涛旅行
途牛旅游
小米WiFi
ofo共享单车
智行火车票
闲鱼
知乎
圈子账本
动卡空间
链家
掌上生活
搜房网房天下
违章查询助手
ETCP停车
养车之家
相机
电话
信息
Safari

用户浏览器

Nginx（代理）

Node（中间层）

Java (API接口)

单页面开发的优点

- 良好的用户体验
- 前后端分离
- 减轻服务端压力
- 共用一套后端程序代码，适配多端

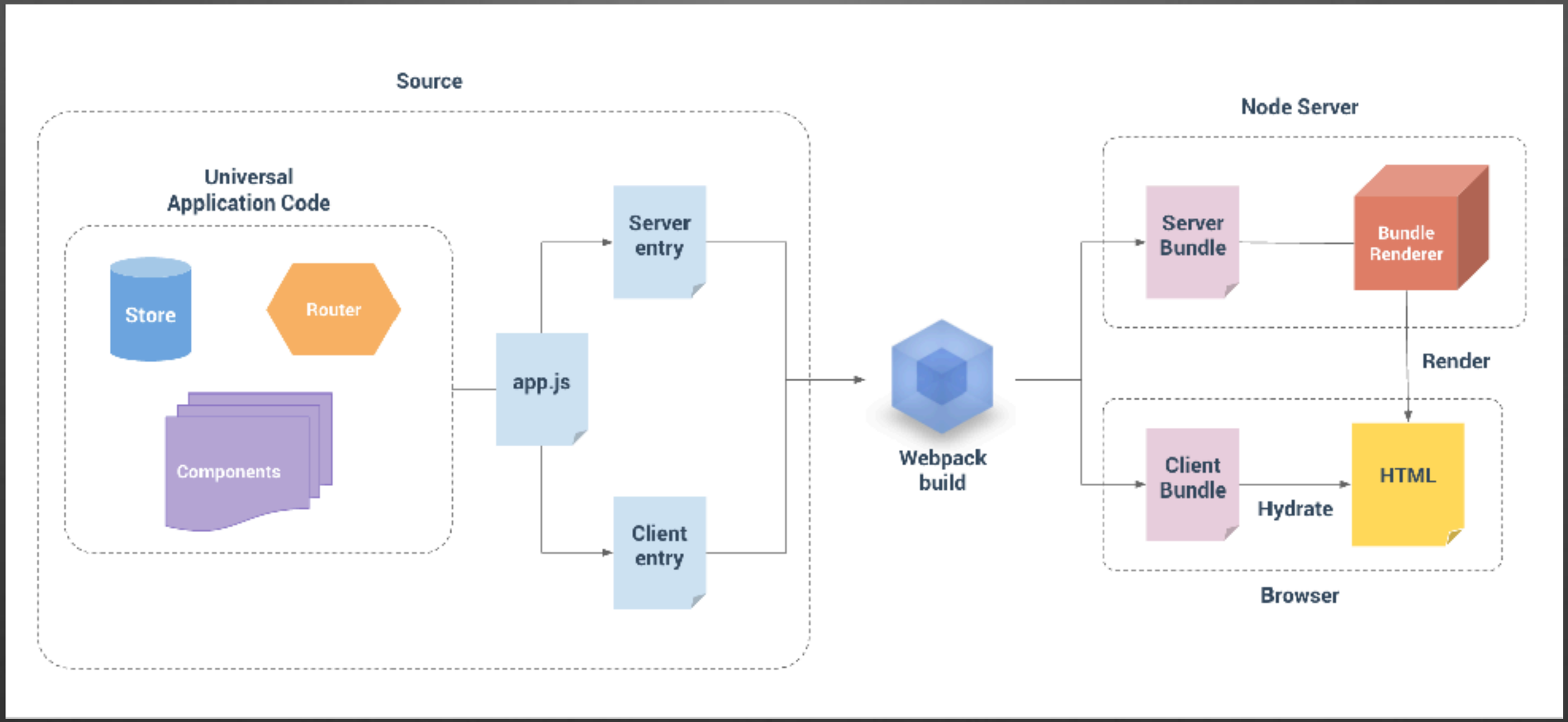
单页面开发的缺点是什么？

- 首屏加载过慢
- SEO

引入Vue SSR

- 服务端预渲染
- 流式渲染
- 对组件进行缓存
- 前后端复用一套代码

Vue SSR流程图


Source
Universal
Application Code
Store
Router
Components
Server
entry
app.js
Client
entry
Webpack
build
Node Server
Server
Bundle
Bundle
Renderer
Render
Client
Bundle
Hydrate
HTML
Browser

目录结构

api	api 接口文件
assets	静态资源目录
components	vue组件
filters	过滤器文件
router	路由文件
views	页面模板文件
utils	工具方法目录
store	vuex相关文件
app.js	vue入口文件
client-entry.js	客户端入口文件
server.entry.js	服务端入口文件
index.html	HTML入口文件

开启Vue SSR

- 使用服务端框架 express 或 koa
- 安装配置webpack 和 vue
- 拆分入口文件，服务端和浏览器要分开渲染

server-entry： 使用 vue ssr 功能将虚拟DOM渲染成网页

client-entry ： 使用 $mount 直接挂载

https://github.com/vuejs/vue-hackernews-2.0

server-entry

```
import { app, router, store } from './app';

const isDev = process.env.NODE_ENV !== 'production';

export default context => {
  const s = isDev && Date.now();

  // 调用 vue-router 切换到到对应的路由
  router.push(context.url);

  // 返回对应要渲染的组件，  检查组件是否有 preFetch 方法
  return Promise.all(router.getMatchedComponents().map(component => {
    if (component.preFetch) {
      return component.preFetch(store);
    }
  })).then(() => {
    isDev && console.log(`data pre-fetch: ${Date.now() - s}ms`);
    context.initialState = store.state;
    return app;
  });
};
```

client-entry

```
import 'es6-promise/auto';
import { app, store } from './app';

store.replaceState(window.__INITIAL_STATE__);

// 挂载到DOM上
app.$mount('#app');
```

组件化

组件化划分

业务组件集

筛选、面包屑　搜索　车型筛选　评价　· · ·

与业务深耦合，复用难度大，难抽象

基础组件集

提示窗　回到顶部　轮播图　表单　· · ·

与业务低耦合，可复用

组件化开发

```
├── components
│   ├── business          # 基础组件
│   └── base              # 业务组件
│       ├── header
│       │   ├── header.vue
│       │   ├── logo.png
│       │   └── header.scss
├── views                 # 页面组件
│   ├── index
│   │   ├── index.vue
│   └── comment
```

```vue
<template>
  <div id="app">
    <com-header :title="title" /></com-header>
    <router-view class="view"></router-view>
    <com-loading :loading="loading"></com-loading>
    <com-footer></com-footer>
  </div>
</template>
<style lang="sass" scoped>
  @import './assets/scss/core/base.scss';
</style>
<script>
  import Header from 'components/business/header';
  import Footer from 'components/business/footer';
  import Loading from 'components/base/loading';

  export default {
    data() {
      return {
        loading: true,
        title: '养车之家'
      };
    },
    components: {
      comHeader: Header,
      comLoading: Loading,
      comNoContent: Footer
    }
  };
</script>
```

- props
- $emit

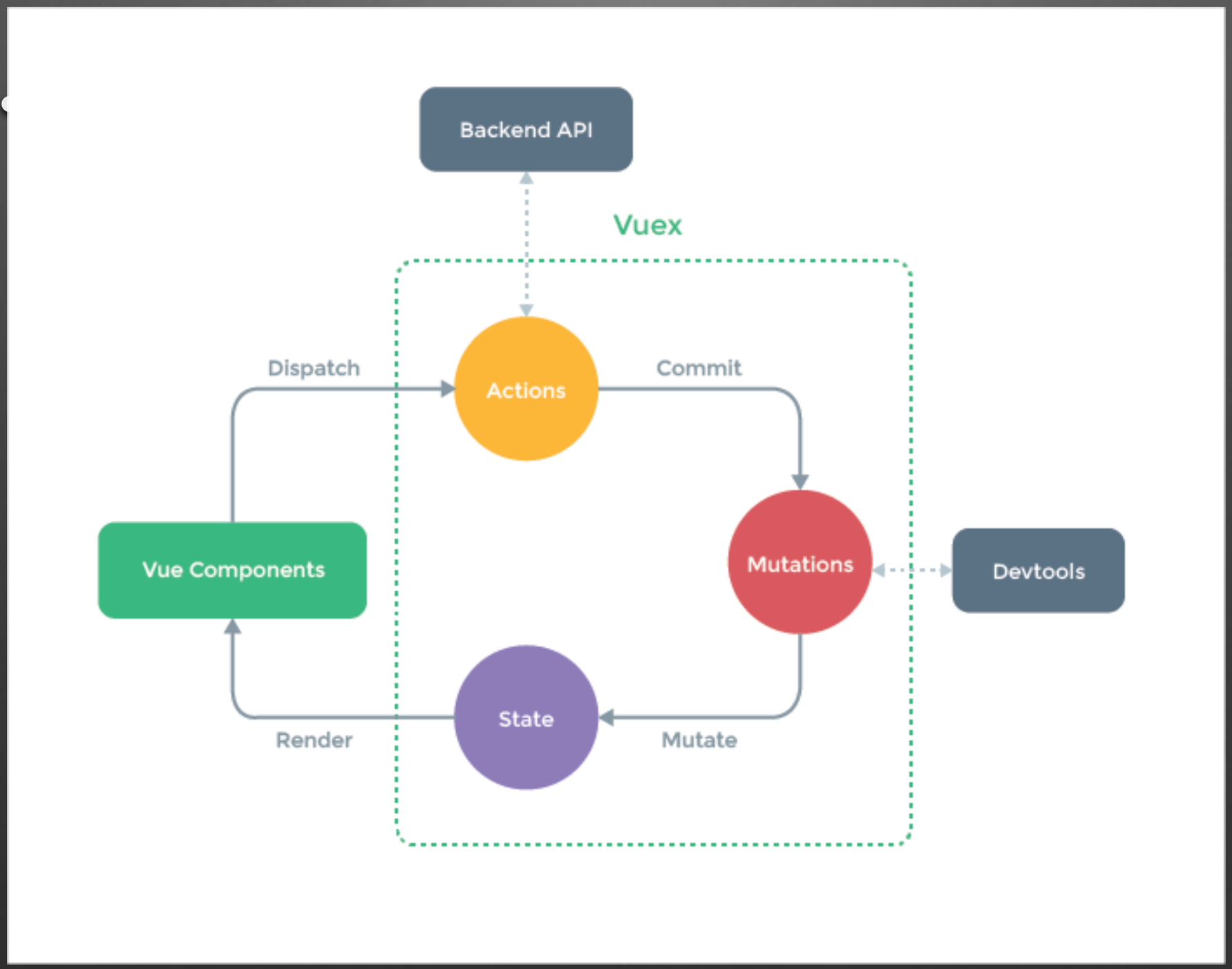
开发中大型项目遇到的问题

- 多个视图依赖于同一状态
- 传参的方法对于多层嵌套的组件将会非常繁琐
- 兄弟组件间的状态传递无能为力

Vuex

- Vuex 是专门为 Vue.js 设计的状态管理库
- 结合Vue实现页面的展示更新
- 统一页面状态管理以及操作处理

Vuex运行流程


Backend API
Vuex
Dispatch
Actions
Commit
Vue Components
Mutations
Devtools
State
Render
Mutate

Vuex项目结构

```
├──── index.html        # HTML模板
├──── app.js
├──── api               # API配置文件
├──── components        # 组件
└──── store
    ├──── index.js      # 入口文件，提供store的module构建
    └──── modules
        ├──── index.js      # 首页模块
        ├──── list.js       # 列表模块
        └──── common.js # 通用模块
```

module

```
const state = {
  banner: []
};

const actions = {
  getBanner: function({commit, state}) {
    // 异步操作
    setTimeout(function() {
      commit('setBanner', ['1.jpg', '2.jpg']);
    }, 1000);
  }
};

const getters = {
  getBannerList: state => state.banner // 获取当前state
};

const mutations = {
  setBanner: (state, res) => {
    // 同步操作，改变state
    state.banner = res;
  }
};

export default {
  state,
  actions,
  getters,
  mutations
};
```

- state：单一状态树
- getters：状态的获取
- mutations：触发同步事件
- action：提交mutation
  可以包含异步操作

数据处理

Axios

- 利用拦截器做预处理
- 对API进行封装
- 在组件中的使用

利用拦截器做预处理

- 请求时的拦截器

```
import axios from 'axios';
import store from '../store';

axios.interceptors.request.use(config => {
  // 这里可以加一些动作，比如来个进度条开始动作
  store.dispatch('SET_PROGRESS', 50);
  return config;
}, error => {
  return Promise.reject(error);
});
```

利用拦截器做预处理

- 请求完成后的拦截器

```
import axios from 'axios';

axios.interceptors.response.use((res) => {
  if (res.status >= 200 && res.status < 300) {
    return res;
  }
  // 由后端抛出的错误
  return Promise.reject(res);
}, (error) => {
  // 由网络或者服务器抛出的错误
  return Promise.reject(error);
});
```

对API进行封装

```
const axios = require('axios');
const cookie = global.__VUE_SSR_CONTEXT__.cookies;

export default {
  post: function(target, params = {}) {
    return new Promise((resolve, reject) => {
      axios({
        url: target,
        method: 'post',
        headers: {
          'X-Requested-With': 'XMLHttpRequest',
          'Content-Type': 'application/json;charset=UTF-8',
          cookie
        },
        data: JSON.stringify(params)
      }).then(res => {
        resolve(res.data);
      }).catch((error) => {
        reject(error);
      });
    });
  }
};
```

对API进行封装

```
/**
 * @file api配置
 */
import api from 'create-api';
import store from '../store';
const version = 'v1_0_0';

function checkStatus(response) {
  store.dispatch('SET_PROGRESS', 100);
  return response;
};

export default {
  /**
   * 支付测试接口
   */
  payTest: function(params) {
    return api.post('/trade/' + version + '/api/pay/notify', params).then(checkStatus);
  },
  /**
   * 获取banner列表
   */
  getBanner: function(params) {
    return api.post('/cus/' + version + '/api/dicBanner', params).then(checkStatus);
  }
};
```

在组件中的使用

```
import api from '../api.js';
export default {
  mounted() {
    api.getBanner().then((res) => {
      console.log(res); // 请求成功
    }).catch((error) => {
      console.log(error); // 请求失败，网络或服务端错误
    });
  }
};
```

优化策略

serviceWorker 预缓存

- 在serviceWorker进程中运行
- 缓存必要的 JS、CSS、font
- 大幅度减少CDN的压力
- 资源是持久性存储

serviceWorker存储空间

和indexDB、LocalStorage共用，存储空间约有**50M**

使用webpack插件

```
new SWPrecachePlugin({
  cacheId: 'yc',
  filename: 'service-worker.js',
  minify: true,
  mergeStaticsConfig: true,
  staticFileGlobs: [
    path.join(__dirname, '../dist/static/*.*')
  ],
  stripPrefixMulti: {
    [path.join(__dirname, '../dist/static')]: '/static'
  },
  // add hash to path
  dontCacheBustUrlsMatching: false,
  staticFileGlobsIgnorePatterns: [
    /index\.html$/,
    /\.map$/,
    /\.css$/,
    /\.svg$/,
    /\.eot$/
  ]
})
```

缓存

- 服务端API数据的缓存
- 组件的缓存

服务API数据的缓存

```
const axios = require('axios');
const md5 = require('md5');
const cached = require('lru-cache')({
  max: 1000,  // 最大数量
  maxAge: 1000 * 60 * 15 // 缓存15分钟
});
export default {
  get: function(target, params) {
    const key = md5(target + JSON.stringify(params));
    if (cached && cached.has(key)) { // 命中缓存
      return Promise.resolve(cached.get(key)); // 返回缓存
    }
    return new Promise((resolve, reject) => {
      axios({
        url: target,
        method: 'get',
        params,
        timeout: 10000
      }).then(res => {
        if (cached && params.cache) {
          cached.set(key, res.data); // 保存缓存数据
        }
        resolve(res.data);
      }).catch((error) => {
        reject(error);
      });
    });
  }
};
```

组件的缓存

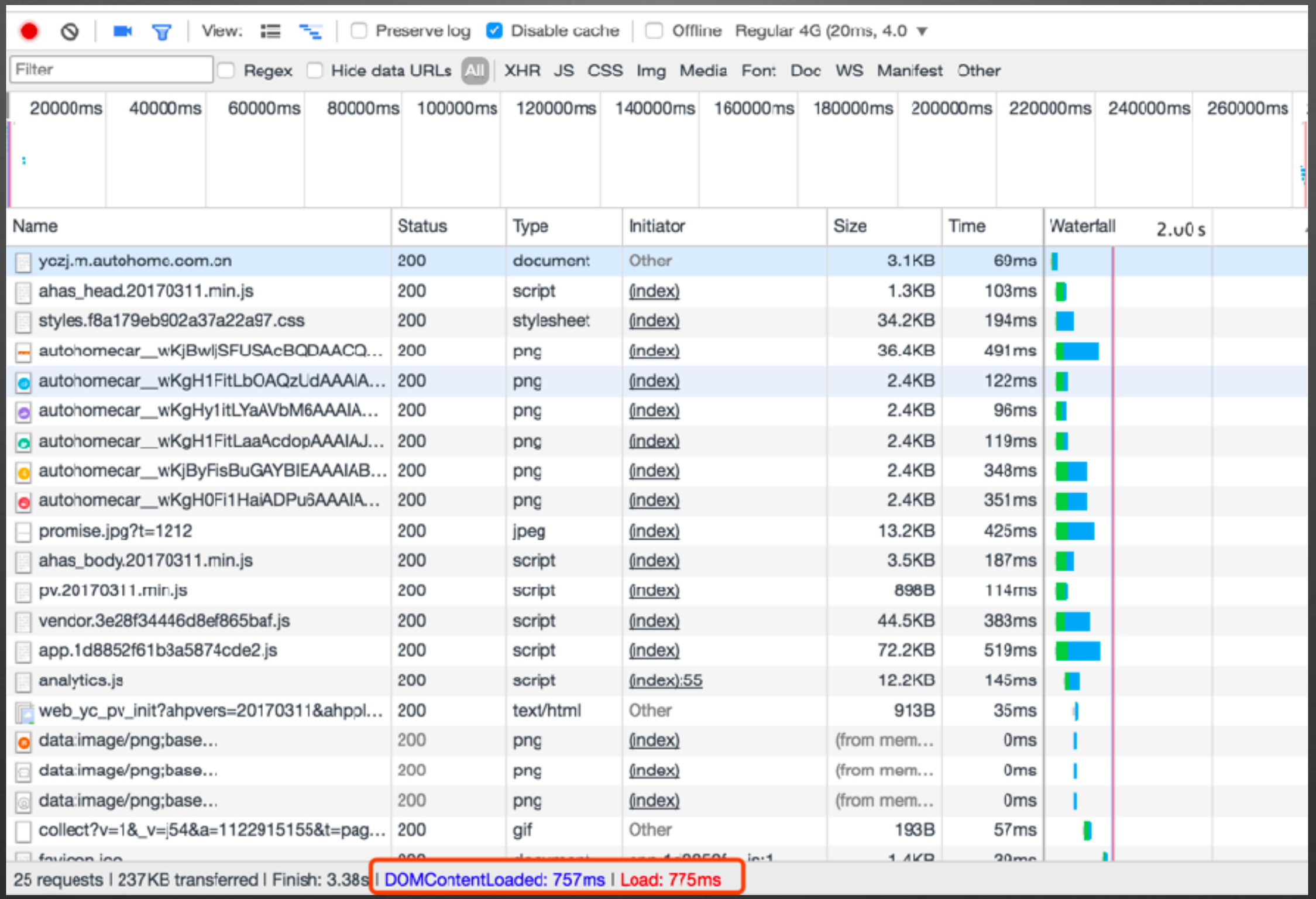
```
return require('vue-server-renderer').createBundleRenderer(bundle, {
  cache: require('lru-cache')({
    max: 1000,  // 缓存最大数量
    maxAge: 1000 * 60 * 15  // 15分钟缓存
  })
});
```

```
export default {
  props: {
    article: {
      type: Object,
      required: true
    }
  },
  serverCacheKey: props => {
    return `${props.article._id}::${props.article.updatedAt}`;
  }
};
```

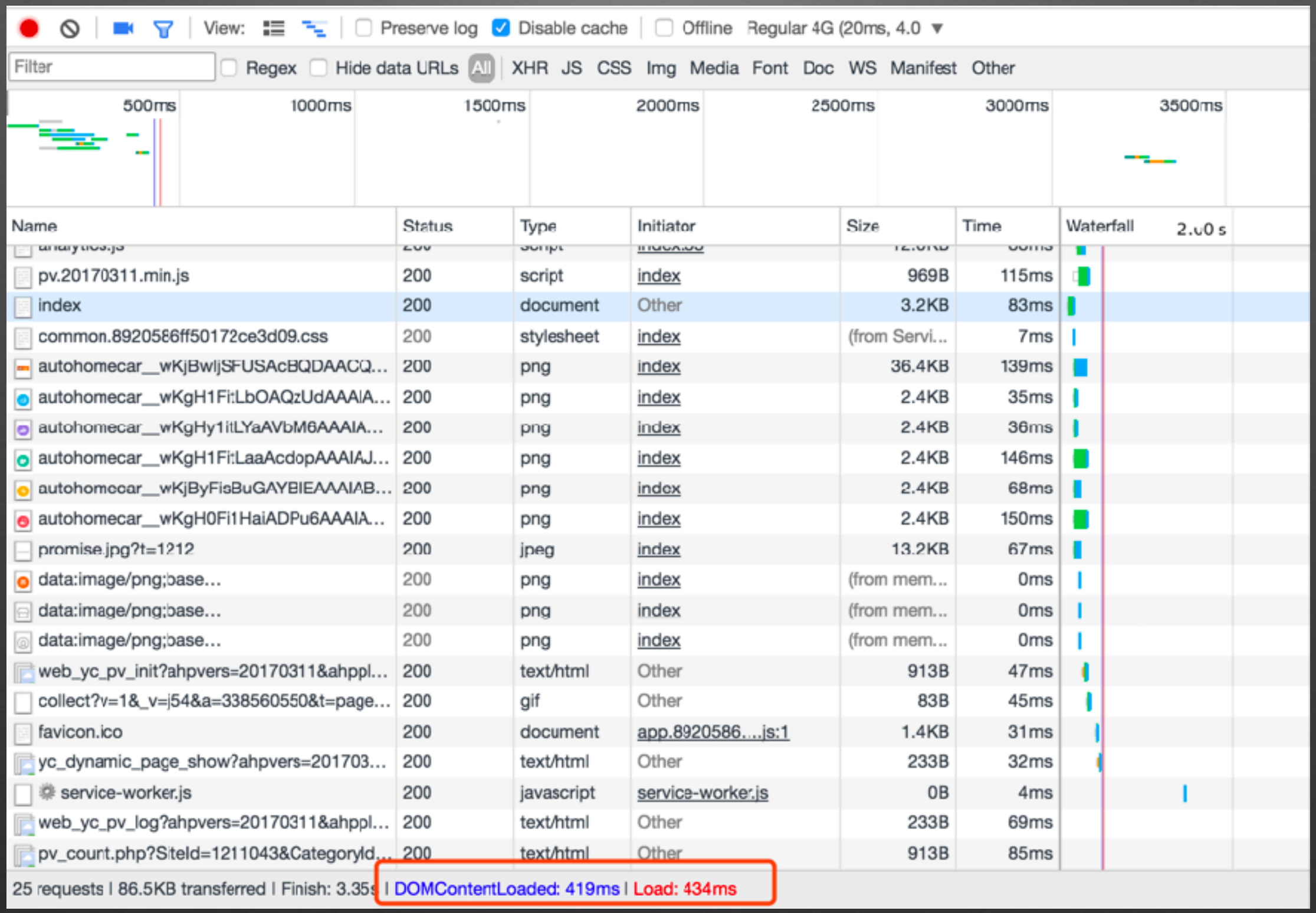
通用策略

- 将静态资源上传到CDN
- 使用Gzip压缩
- 图片压缩、懒加载和base64
- 首屏外使用异步加载

首页优化前

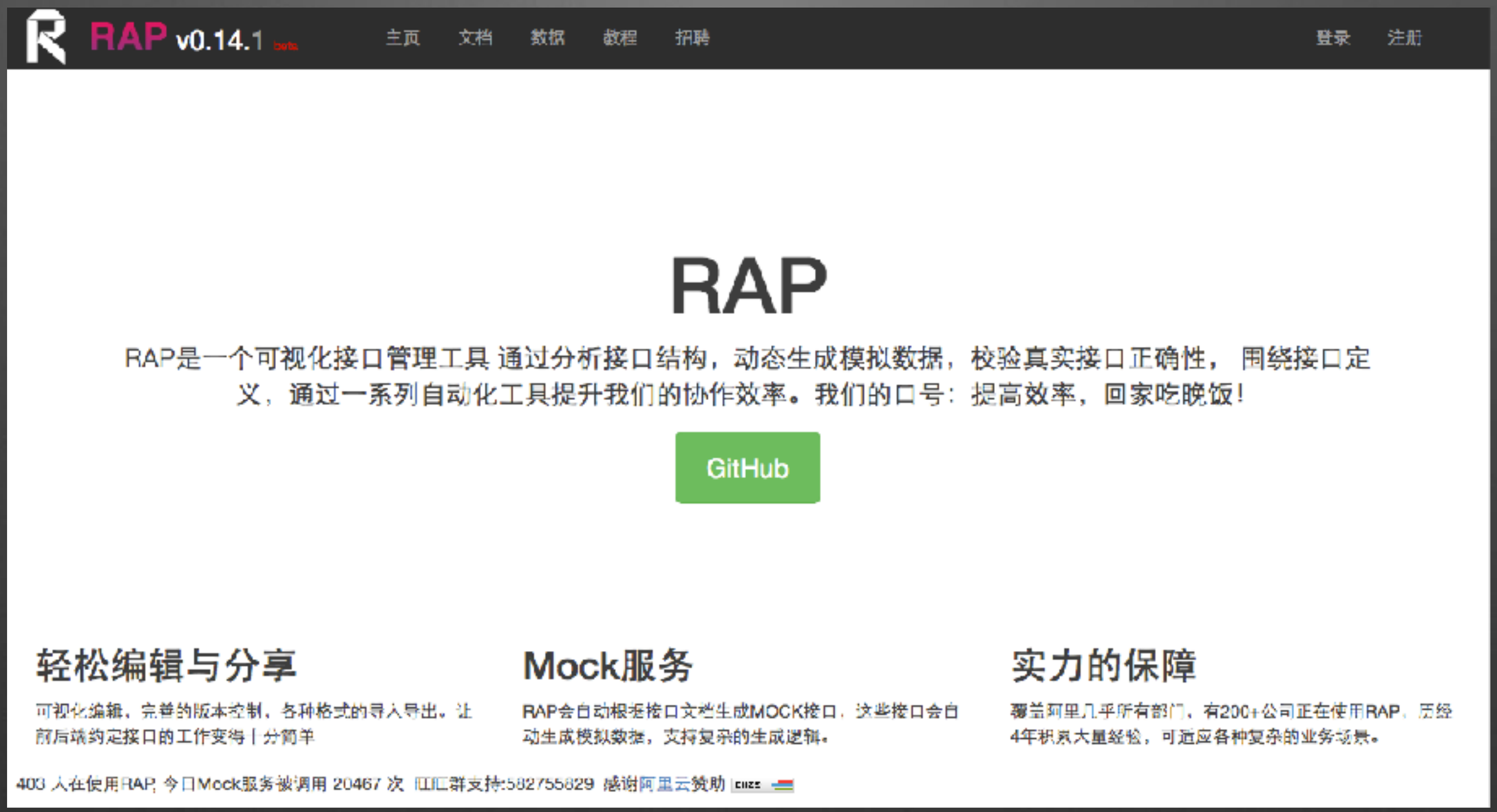

View:
Preserve log
Disable cache
Offline
Regular 4G (20ms, 4.0
Filter
Regex
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
20000ms
40000ms
60000ms
80000ms
100000ms
120000ms
140000ms
160000ms
180000ms
200000ms
220000ms
240000ms
260000ms
Name
Status
Type
Initiator
Size
Time
Waterfall
2.00 s
yczj.m.autohome.com.cn 200 document Other 3.1KB 69ms
ahas_head.20170311.min.js 200 script (index) 1.3KB 103ms
styles.f8a179eb902a37a22a97.css 200 stylesheet (index) 34.2KB 194ms
autohomecar__wKjBwljSFUSAcBQDAACQ... 200 png (index) 36.4KB 491ms
autohomecar__wKgH1FitLbOAQzUdAAAIA... 200 png (index) 2.4KB 122ms
autohomecar__wKgHy1itLYaAVbM6AAAIA... 200 png (index) 2.4KB 96ms
autohomecar__wKgH1FitLaaAcdopAAAIAJ... 200 png (index) 2.4KB 119ms
autohomecar__wKjByFisBuGAYBIEAAAIAB... 200 png (index) 2.4KB 348ms
autohomecar__wKgH0Fi1HaiADPu5AAAIA... 200 png (index) 2.4KB 351ms
promise.jpg?t=1212 200 jpeg (index) 13.2KB 425ms
ahas_body.20170311.min.js 200 script (index) 3.5KB 187ms
pv.20170311.min.js 200 script (index) 898B 114ms
vendor.3e28f34446d8ef865baf.js 200 script (index) 44.5KB 383ms
app.1d8852f61b3a5874cde2.js 200 script (index) 72.2KB 519ms
analytics.js 200 script (index):55 12.2KB 145ms
web_yc_pv_init?ahpvers=20170311&ahppl... 200 text/html Other 913B 35ms
data:image/png;base... 200 png (index) (from mem... 0ms
data:image/png;base... 200 png (index) (from mem... 0ms
data:image/png;base... 200 png (index) (from mem... 0ms
collect?v=1&_v=j54&a=1122915155&t=pag... 200 gif Other 193B 57ms
25 requests | 237KB transferred | Finish: 3.38s | DOMContentLoaded: 757ms | Load: 775ms

首页优化后


View:
Preserve log
Disable cache
Offline
Regular 4G (20ms, 4.0
Filter
Regex
Hide data URLs
All
XHR JS CSS Img Media Font Doc WS Manifest Other
500ms
1000ms
1500ms
2000ms
2500ms
3000ms
3500ms
Name
Status
Type
Initiator
Size
Time
Waterfall
2.00 s
pv.20170311.min.js 200 script index 969B 115ms
index 200 document Other 3.2KB 83ms
common.8920586ff50172ce3d09.css 200 stylesheet index (from Servi... 7ms
autohomecar__wKjBwIjSFUSAcBQDAACQ... 200 png index 36.4KB 139ms
autohomecar__wKgH1FiLbOAQzUdAAAIA... 200 png index 2.4KB 35ms
autohomecar__wKgHy1itLYaAVbM6AAAIA... 200 png index 2.4KB 36ms
autohomecar__wKgH1FiLaaAcdopAAAIAJ... 200 png index 2.4KB 146ms
autohomecar__wKjByFisBuGAYBIEAAAIAB... 200 png index 2.4KB 68ms
autohomecar__wKgH0Fi1HaiADPu6AAAIA... 200 png index 2.4KB 150ms
promise.jpg?t=1212 200 jpeg index 13.2KB 67ms
data:image/png;base... 200 png index (from mem... 0ms
data:image/png;base... 200 png index (from mem... 0ms
data:image/png;base... 200 png index (from mem... 0ms
web_yc_pv_init?ahpvers=20170311&ahppl... 200 text/html Other 913B 47ms
collect?v=1&_v=j54&a=338560550&t=page... 200 gif Other 83B 45ms
favicon.ico 200 document app.8920586....js:1 1.4KB 31ms
yc_dynamic_page_show?ahpvers=201703... 200 text/html Other 233B 32ms
service-worker.js 200 javascript service-worker.js 0B 4ms
web_yc_pv_log?ahpvers=20170311&ahppl... 200 text/html Other 233B 69ms
pv_court.php?SiteId=1211043&CategoryId... 200 text/html Other 913B 85ms
25 requests | 86.5KB transferred | Finish: 3.35s | DOMContentLoaded: 419ms | Load: 434ms

Mock

RAP



结合了文档、Mock.js、可视化接口过渡、文档修改提醒、支持本地部署

HTTP请求

Api proxy server

根据请求路径，代理到对应的服务器

请求成功

请求失败

返回响应数据

请求RAP接口

返回Mock数据

Api proxy server

发布

GitLab

上传测试版本库

发布到测试环境

删除原有项目，部署最新代码

上传线上版本库

发布到准生产环境

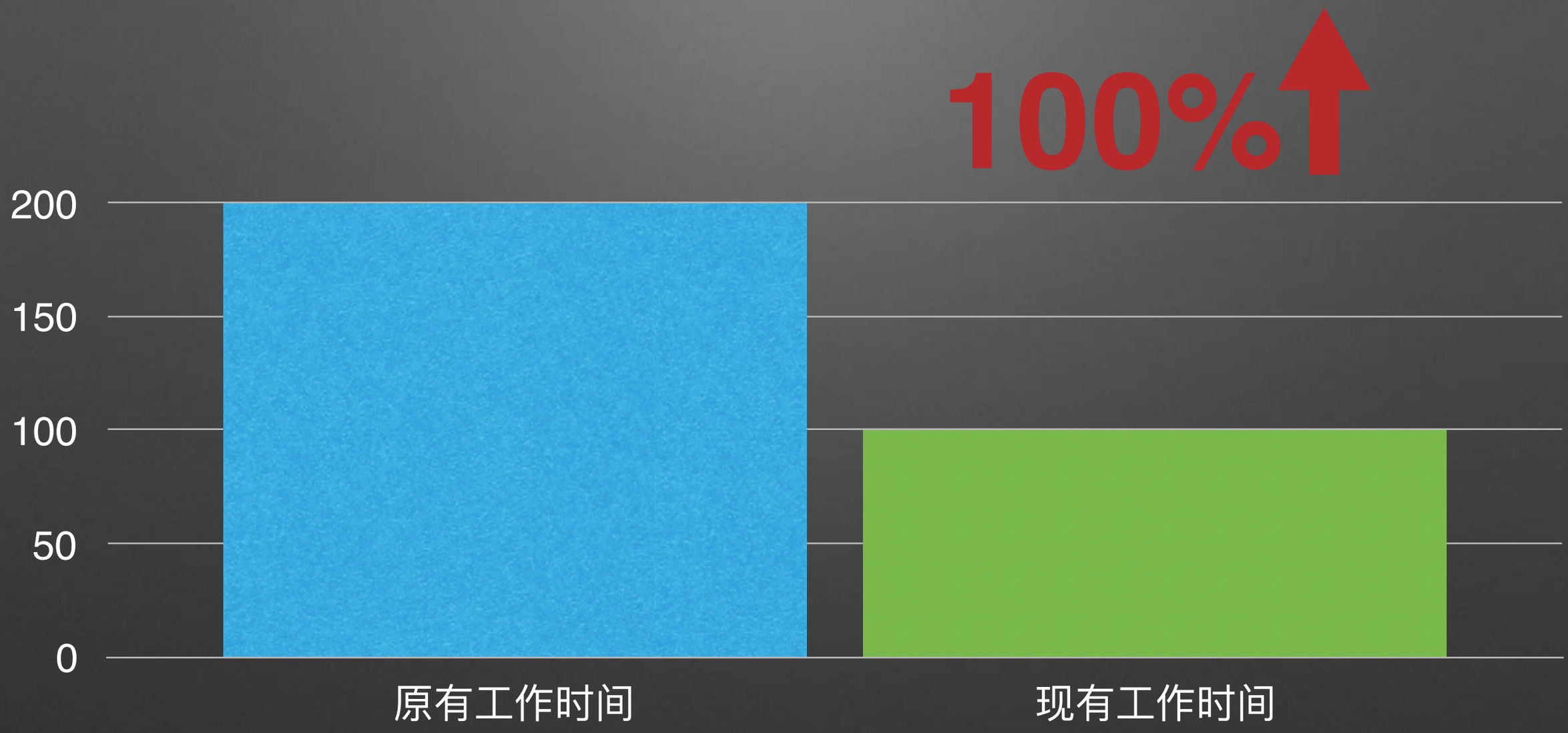
验证线上包

覆盖原有项目

发布到线上环境

验证线上环境

效率提升



总结

- Vue SSR
- Vuex
- 数据处理
- 优化策略
- Mock
- 发布

Thanks