**性能优化**

**指标**

|  |
| --- |
| [性能指标计算原理](https://bytedance.feishu.cn/wiki/wikcny7kzXdTeEQpCNLeyBq7rNe)  FCP、LCP、TTI、SI、CLS、TBT |

**数据采集**

|  |
| --- |
| [生产链路前端性能&异常埋点梳理 Copy](https://bytedance.feishu.cn/docx/MHT0dKoyfoK5c5xIjf9cgzypnA2)  [生产链路前端性能埋点方案 Copy](https://bytedance.feishu.cn/docx/TZeYdDs3Zo7D9nx9n99cl4LFnCc) |

**看板建设**

Slardar

**告警建设**

TODO

**优化（根据指标，分析优化）**

**分析**

|  |
| --- |
| [Chrome DevTools Performance 功能详解](https://bytedance.feishu.cn/wiki/wikcnsbu7Gaamrp1hMQQz6trnWd) |

**优化**

**网络优化**

|  |
| --- |
| [一、网络优化](https://bytedance.feishu.cn/wiki/wikcniuQP0alsXm2EMUzUuXBhgd#wc1aEU) |

**HTTP2**

[**开启 br 压缩**](https://site.bytedance.net/docs/5402/neptune/http_compress/)

**preconnect、dns-prefetch**

**资源加载优化**

**资源预处理**

|  |
| --- |
| [1. 资源预处理](https://bytedance.feishu.cn/wiki/wikcnYP4YriNp1MJy3r6tFY17Yd) |

preload、prefetch、prerender

**打包优化**

**拆/合包**

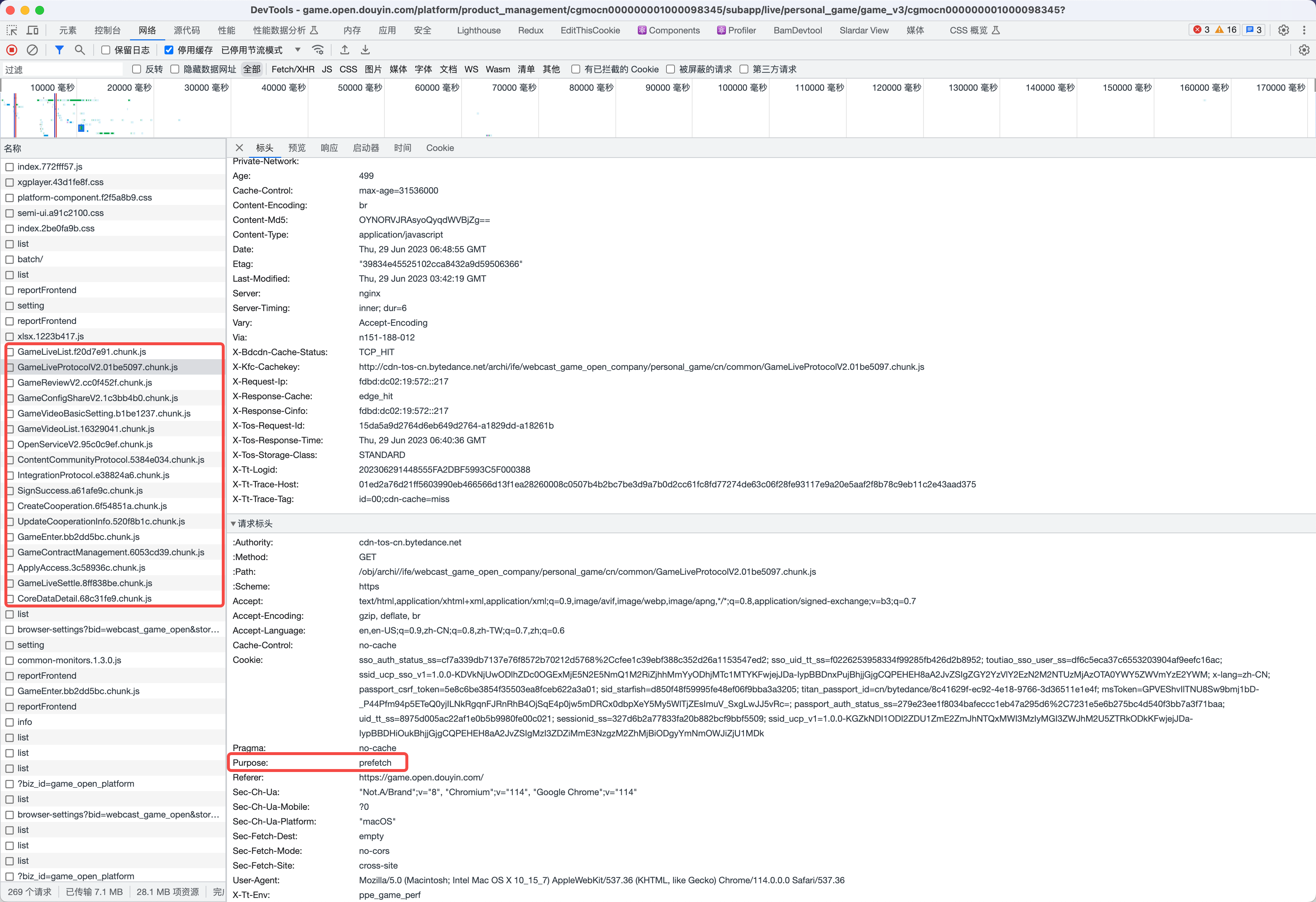
|  |
| --- |
| [webpack SplitChunksPlugin](https://webpack.js.org/plugins/split-chunks-plugin/) |

**拆包策略**

1. 尽可能按需加载 -- 减少首屏包体积
2. 路由按需加载，通过 [loadable](https://loadable-components.com/) 实现。

|  |
| --- |
| 加上 webpackChunkName，提高缓存使用率。通过 babel 插件实现，不需要手动填写。 |

|  |
| --- |
| TypeScript // 例如  const GameLiveList = loadable(  () => import(/\* webpackChunkName: "GameLiveList", webpackPrefetch: true \*/ '../pages/GameLiveList') ); |



1. 配置生成的路由 chunkName

|  |
| --- |
| TypeScript output: {  chunkFilename: 'common/[name].[chunkhash:8].chunk.js', } |

1. 根据动态 import 路径，自动生成 chunkName

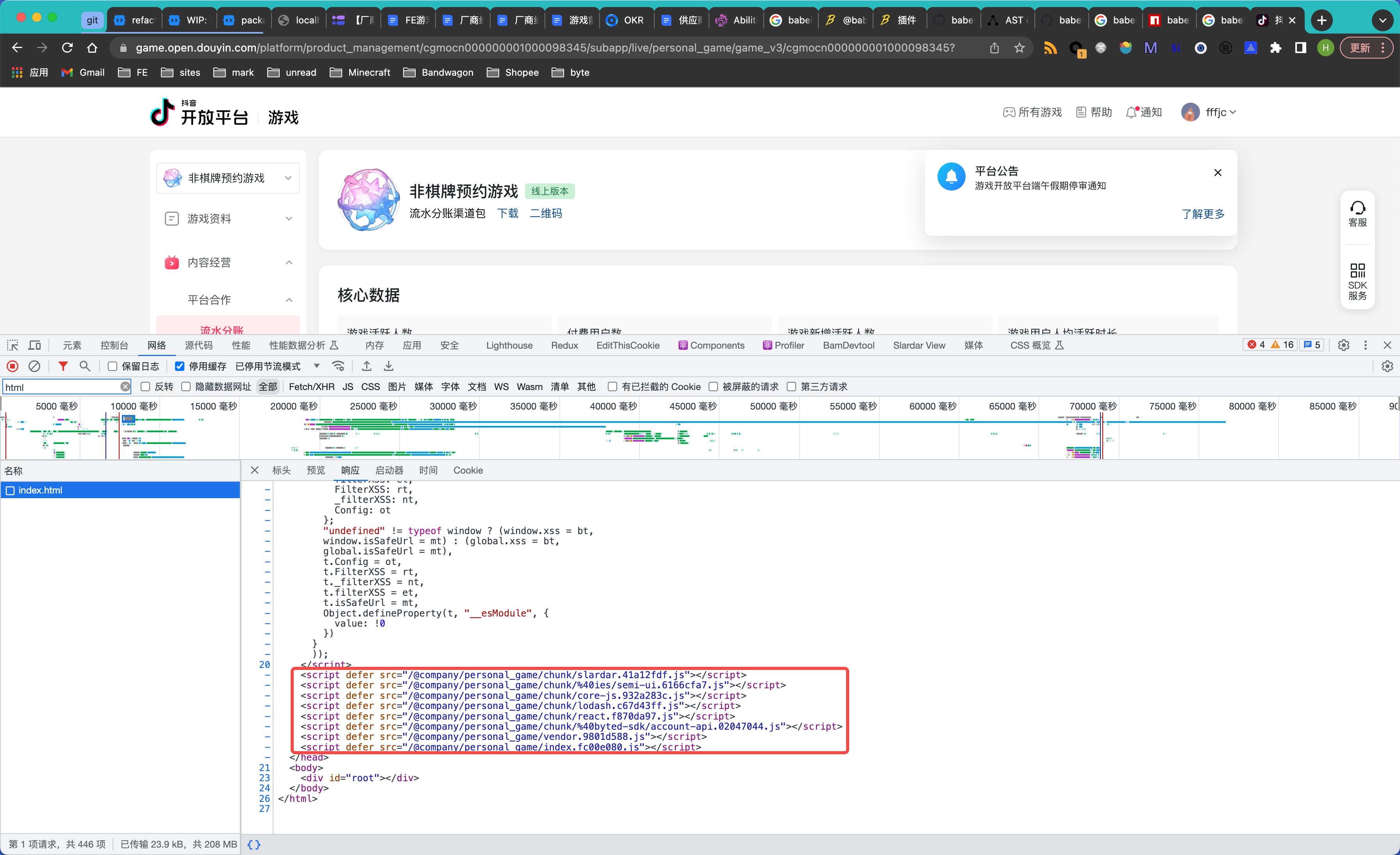
|  |
| --- |
| TypeScript  // babel 插件 function getChunkName(filename) {  return filename.replace(/^\.\.\//, '').replace(/(\.)\*\//g, '-'); }  function hasComment(comment) {  return comment && comment.value.replace(/\\*+/g, '').trim().startsWith('webpackChunkName'); }  module.exports = function ({ types: t }) {  return {  visitor: {  CallExpression(path) {  if (path.node.callee.type !== 'Import') {  return;  }   const [arg] = path.node.arguments;  const [comment] = arg.leadingComments || [];   if (!hasComment(comment)) {  t.addComment(arg, 'leading', `webpackChunkName: '${getChunkName(arg.value)}'`);  }  },  },  }; }; |

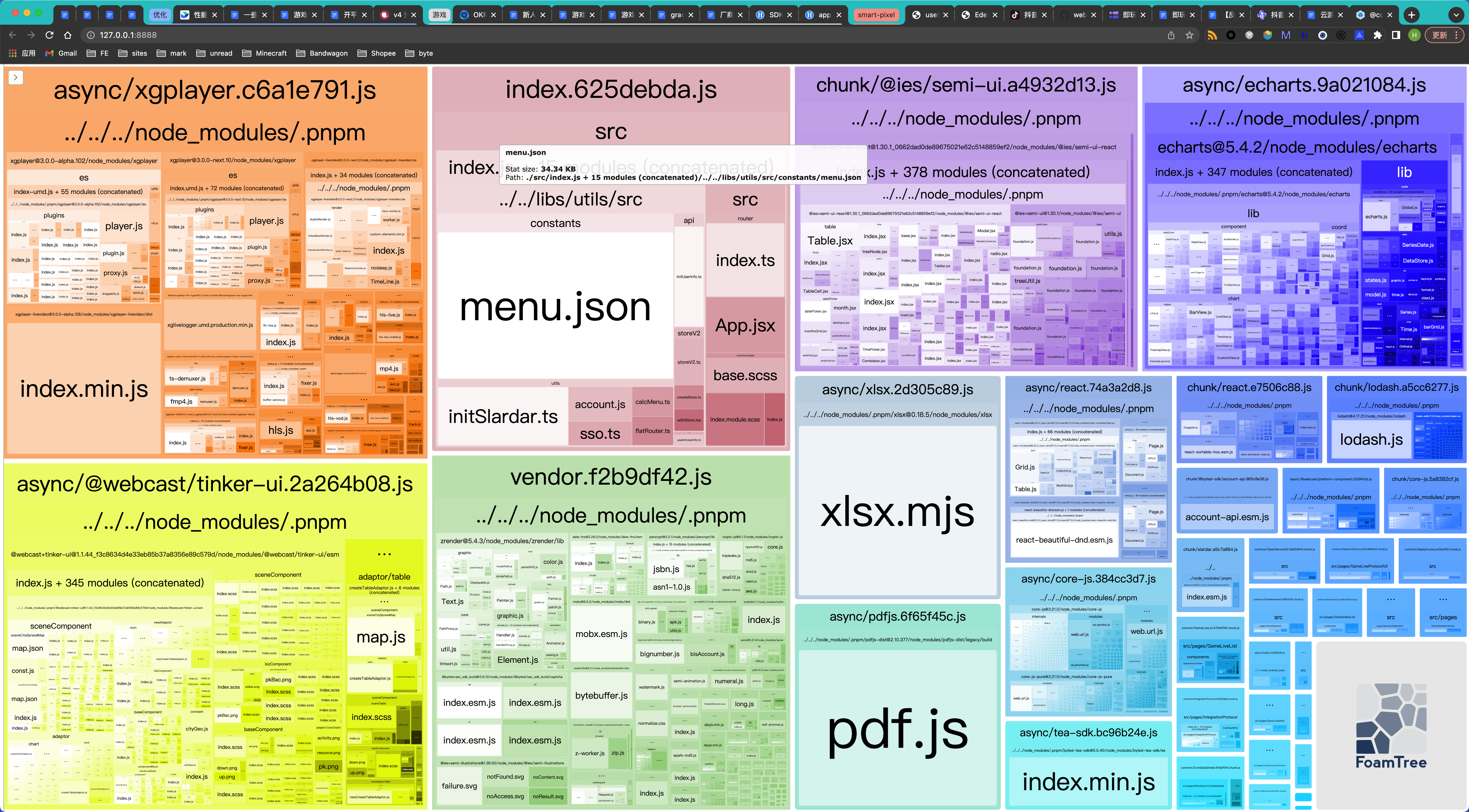
1. 依赖包的按需加载 async chunk，通过 [cacheGroups](https://webpack.js.org/plugins/split-chunks-plugin/#splitchunkscachegroups) 实现，保证 group 优先级最高。存放于目录 /async。
2. 通过分析入口文件，手动动态 import 实现按需加载，通过 [loadable](https://loadable-components.com/) 实现。
3. 合理拆包 -- 减少最大包体积
4. 将较大的依赖包单独打包 split chunk，后续也可抽离公共包，通过 [cacheGroups](https://webpack.js.org/plugins/split-chunks-plugin/#splitchunkscachegroups) 实现，优先级低于 async chunk，但高于其他 chunk。存放于目录 /chunk。
5. 其余的依赖包打包到一起作为 vendor chunk。
6. 入口文件通过 webpack 自动拆包，拆解为 index chunk，以及部分按需加载的 chunk。
7. 抽离公共依赖包。

|  |
| --- |
| TypeScript {  // webpack 拆包参数  chunks: 'all',  minRemainingSize: 0,  minSize: 20000,  minChunks: 1,  maxAsyncRequests: 30,  maxInitialRequests: 30,  enforceSizeThreshold: 50000,   // 自定义拆包规则  cacheGroups: {  defaultVendors: {  test: /[\\/]node\_modules[\\/]/,  priority: -10,  reuseExistingChunk: true,  name: 'vendor',  filename: '[name].[contenthash:8].js',  },  default: {  minChunks: 2,  priority: -20,  reuseExistingChunk: true,  filename: '[name].[contenthash:8].js',  },  ...moduleNames.reduce((groups, moduleName, idx) => {  return {  // async chunk，按需加载  [`async-${moduleName}`]: {  chunks: 'async',  test: new RegExp(`[\/]node\_modules[\/][^\/]\*${moduleName}`), // 只识别 node\_modules 下一级目录名称，保证匹配精确  // test: new RegExp(moduleName),  priority: 100 + idx,  reuseExistingChunk: false,  name: `async/${moduleName}`,  filename: '[name].[contenthash:8].js',  },  // split chunk，defer 加载  [`${moduleName}`]: {  test: new RegExp(`[\/]node\_modules[\/][^\/]\*${moduleName}`), // 只识别 node\_modules 下一级目录名称，保证匹配精确  // test: new RegExp(moduleName),  priority: 10 + idx,  reuseExistingChunk: true,  name: `chunk/${moduleName}`,  filename: '[name].[contenthash:8].js',  },  };  }),  }, } |

**拆包效果**

* 首屏只会加载 index chunk、vendor chunk，以及 /chunk 目录下的 split chunk。chunk 脚本直接嵌入 html 文件，defer 加载。
* 路由 chunk、async chunk按需加载。





**按需加载**

**依赖包按需加载处理**

**手动动态 import**

**删除无用依赖**

**解决依赖包重复引用**

* webpack alias
* overrides
* npm dedupe
* https://github.com/scinos/yarn-deduplicate

**用体积小的依赖包替换**

* moment -> dayjs
* lodash -> lodash-es

**CDN**

**Tree shaking**

* [sideEffects](https://webpack.js.org/guides/tree-shaking/#mark-the-file-as-side-effect-free)

**压缩**

**参考**

[Webapp 性能优化指导手册](https://bytedance.feishu.cn/wiki/wikcnUkz0RIRMbOIOOhWHCR4QVf#)

[性能优化实践指南](https://bytedance.feishu.cn/docx/doxcngTgupE8patDZDKiTPF9FGd)

[看懂 Lighthouse 中 Performance 核心指标](https://bytedance.feishu.cn/wiki/wikcn2NARSC9ezGpWugaTTZuqsc#leg9HM)