2021/5/29 - B 站直播

前端早早聊大会 - 免费福利专场

早

如何基于 Node 做 DevOps 实战

工程化：Cookie@涂鸦 | 14:00

如何针对研发全链路做性能优化

工程化：Yck@酷家乐 | 15:00

如何深入认知 Flutter 手势体系

Flutter：张风捷特烈 | 16:00

wx：wongmicky，进群领取录播/讲稿/PPT

全链路研发性能优化

yck

yck @ 酷家乐

基础架构组

两年多前端，技术栈很杂

会讲哪些链路上的东西?

- 本地开发
  - 本地构建
- CI 构建
  - 依赖安装
  - 代码质量保障
  - 构建
- 线上
  - 页面性能

今天只聊方案

开发构建层面

这块内容是日常工作最常接触的，感知很强

假设老王每天修改保存 100 次代码并查看效果

整整 1000 秒，可能还不止！

在这里做点性能优化，说不定就能早点回家了呢

Nobundle 方案

三大产品

Vite

Snowpack

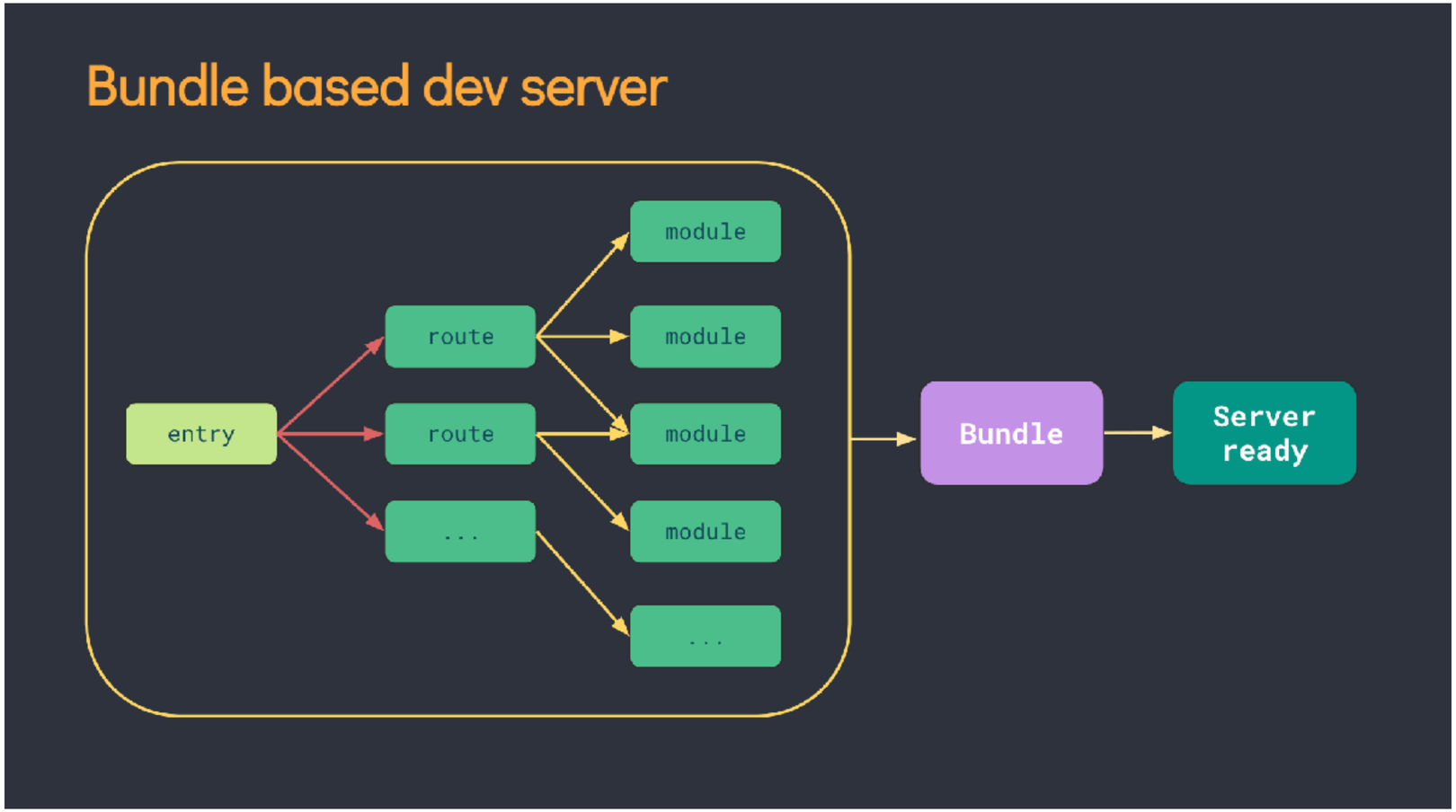
Wmr

效果感人，保存代码页面即变化

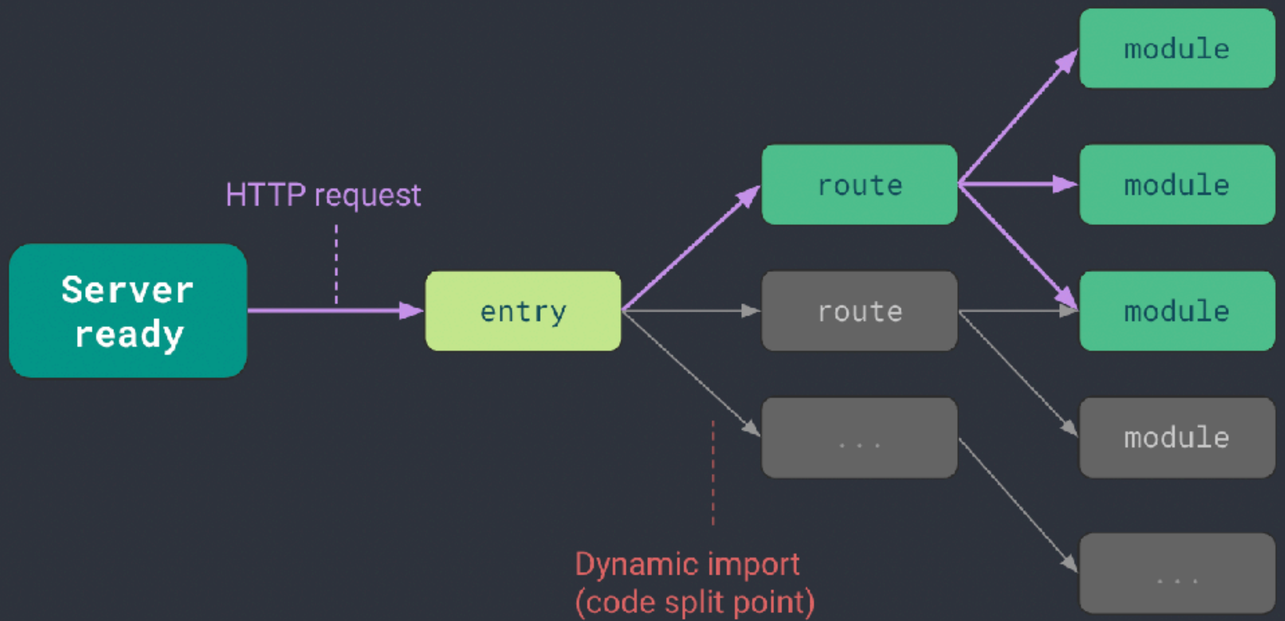
为什么 Nobundle 快?

Nobundle 的基石是 ESM

<script type=module >


Bundle based dev server
entry
route
route
...
module
module
module
module
...
Bundle
Server ready

Native ESM based dev server



放弃代码构建，让浏览器按需请求文件

Nobundle 方案真的不需要构建了?

首先在开发阶段我们需要将 CMD 的包转成 ESM

在生产中也不是用的 ESM，而是打包成 CMD

JavaScript modules via script tag - LS

Usage: Global

% of all users ?

92.65% + 0.18% = 92.83%

Loading JavaScript module scripts (aka ES6 modules) using `<script type="module">` Includes support for the `nomodule` attribute.

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
	12-14															
	1 6 15	2-53	4-59	3.1-10	10-46	3.2-10.2										
	6 16-18	2 54-59	1 60	4 5 10.1	1 47	4 5 10.3							4-7.4			
6-10	79-89	60-87	61-89	11-14	48-74	11-14.4		2.1-4.4.4	12-12.1				8.2-13.0			
11	90	88	90	14.1	75	14.5	all	90	62	90	87	12.12	14.0	10.4	7.12	2.5
		89-90	91-93	TP												

但是 Nobundle 方案接入现有业务会有很多痛点

Evan You
@youyuxi

I feel I may never be able to go back to webpack

翻译推文

上午7:11 · 2020年4月30日 · Twitter for Android

93 转推 **24** 引用推文 **1,202** 喜欢次数

💎🙌🚀🚀 **Sean Larkin** @TheLarkInn · 2020年4月30日

回复 @youyuxi

大哥...😂

17 5 125

Evan You @youyuxi · 2020年4月30日

对不住了兄弟 😂

2 3 67

CI 构建层面，大体分为三块流程

安装依赖 -> 代码质量保障 -> 应用构建

yarn 慢的原因可能是什么?

源

利用好 Docker cache

```
COPY package.json yarn.lock ./
RUN yarn install
```

尝试 yarn 2?

代码质量保障

主要是跑单测，几百上千个 case 全部跑一遍就是几分钟过去了

增量执行，只执行修改文件关联的单测

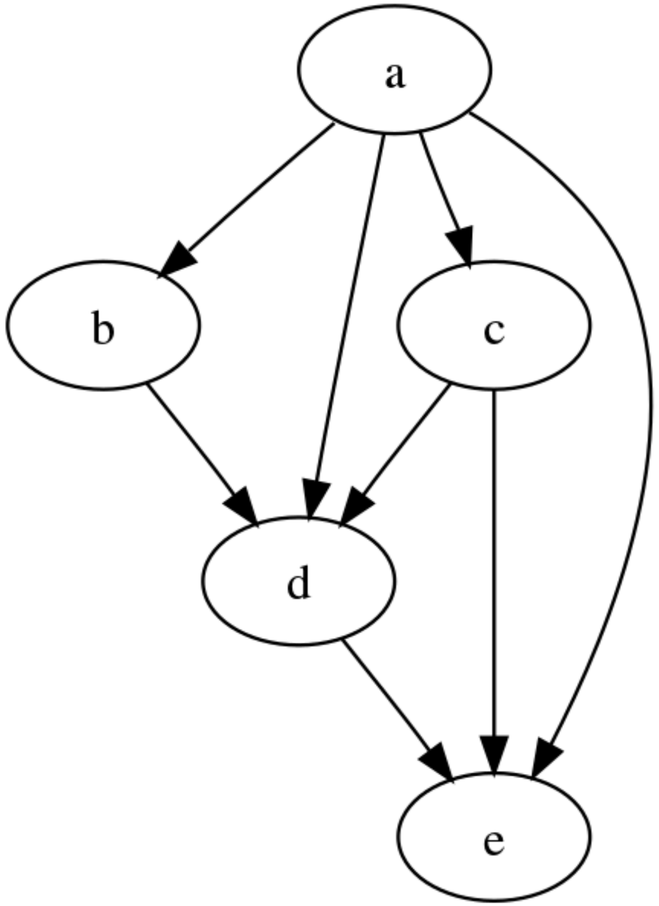
了解下 Jest CLI 中和 —onlyChanged 相关的参数

Build 慢？这题我会！

Webpack 这个话题绕不开，但是我们不讲怎么改配置

升级 Webpack 版本，会有很大惊喜

持久化缓存


a
b
c
d
e

依旧是增量方案，官方宣称 98 % 的提升

更好用的 Tree shaking 以及新增的 Prepack 能力

重大变更: 构建优化

嵌套的 tree-shaking

webpack 现在能够跟踪对导出的嵌套属性的访问。这可以改善重新导出命名空间对象时的 Tree Shaking（清除未使用的导出和混淆导出）。

```
// inner.js
export const a = 1;
export const b = 2;

// module.js
export * as inner from './inner';
// 或 import * as inner from './inner'; export { inner };

// user.js
import * as module from './module';
console.log(module.inner.a);
```

在这个例子中，可以在生产模式下删除导出的 `b`。

内部模块 tree-shaking

webpack 4 没有分析模块的导出和引用之间的依赖关系。webpack 5 有一个新的选项 `optimization.innerGraph`，在生产模式下是默认启用的，它可以对模块中的标志进行分析，找出导出和引用之间的依赖关系。

在这样的模块中：

```
import { something } from './something';

function usingSomething() {
  return something;
}

export function test() {
  return usingSomething();
}
```

内部依赖图算法会找出 `something` 只有在使用 `test` 导出时才会使用。这允许将更多的出口标记为未使用，并从代码包中省略更多的代码。

Input

```
(function () {
  function hello() { return 'hello'; }
  function world() { return 'world'; }
  global.s = hello() + ' ' + world();
})();
```

Output

```
s = "hello world";
```

Elimination of abstraction tax

Input

```
(function () {
  var self = this;
  ['A', 'B', 42].forEach(function(x) {
    var name = '_' + x.toString()[0].toLowerCase
    var y = parseInt(x);
    self[name] = y ? y : x;
  });
})();
```

Output

```
_a = "A";
_b = "B";
_4 = 42;
```

Fibonacci

Input

```
(function () {
  function fibonacci(x) {
    return x <= 1 ? x : fibonacci(x - 1) + fibon
  }
  global.x = fibonacci(15);
})();
```

Output

```
x = 610;
```

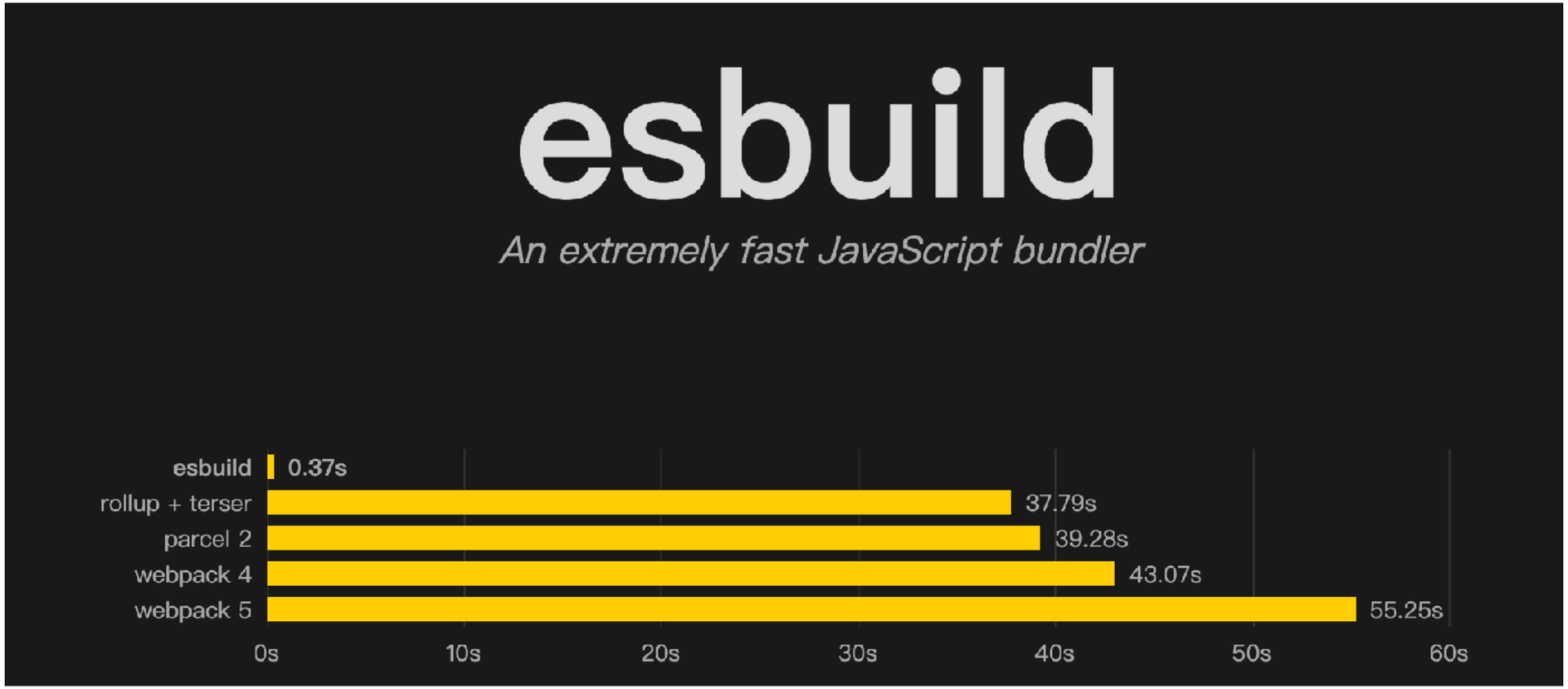
Build 过程中，其实压缩代码也花了挺多时间

不信的话试试这个 Speed Measure Plugin 插件

```
SMP ⏱
General output time took 2 mins, 46.422 secs

SMP ⏱ Plugins
ExtractTextPlugin took 1 mins, 56.121 secs
UglifyJSPlugin took 1 mins, 56.015 secs
ForceCaseSensitivityPlugin took 22.758 secs
IgnorePlugin took 0.51 secs
ManifestPlugin took 0.4 secs
SpriteLoaderPlugin took 0.047 secs
DefinePlugin took 0.002 secs

SMP ⏱ Loaders
(none) took 46.763 secs
    module count = 1576
extract-text-webpack-plugin, and
style-loader, and
css-loader, and
sass-loader took 24.012 secs
    module count = 192
coffee-loader took 5.337 secs
    module count = 29
```


esbuild
An extremely fast JavaScript bundler
esbuild
0.37s
rollup + terser
37.79s
parcel 2
39.28s
webpack 4
43.07s
webpack 5
55.25s
0s
10s
20s
30s
40s
50s
60s

当打包器风险略大，用来压缩代码效果同样群拔

实测业务项目中能带来 30 - 40% 的压缩速度提升

多页应用打包神器：依旧是增量

改点代码就得全部入口都打包?

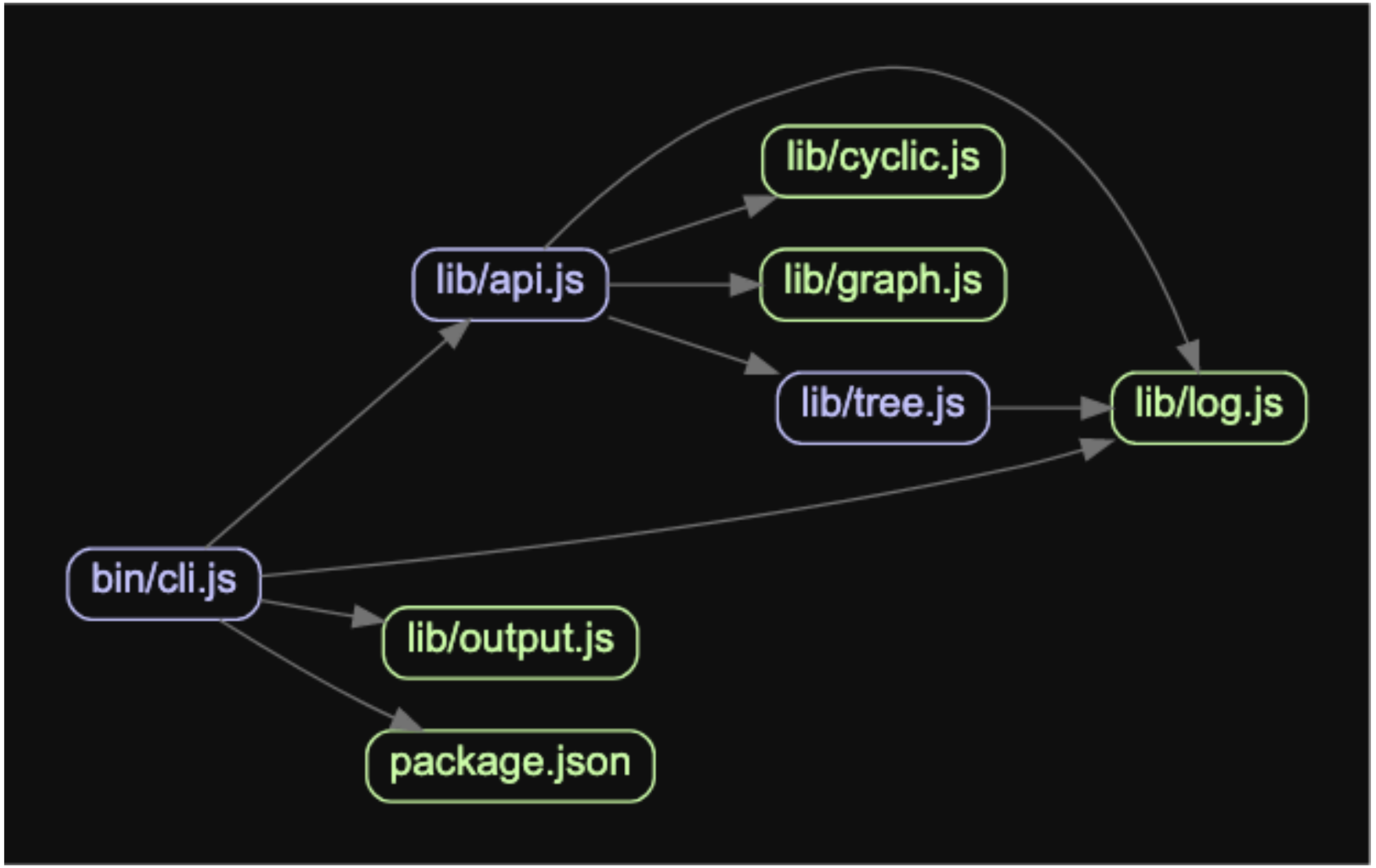
能不能改了哪个入口里的文件，打包哪个入口

实现很简单，效果绝对群拔！

首先找到距离上次构建后所有修改过的文件

git diff --name-only {git tag / commit sha}

接下来构建依赖树，使用 madge 库或者别的都行


lib/cyclic.js
lib/api.js
lib/graph.js
lib/tree.js
lib/log.js
bin/cli.js
lib/output.js
package.json

最后根据生成的依赖树及修改过的文件查找入口即可

生产环境页面性能

先拿指标再谈优化

你了解的指标可能已经过时

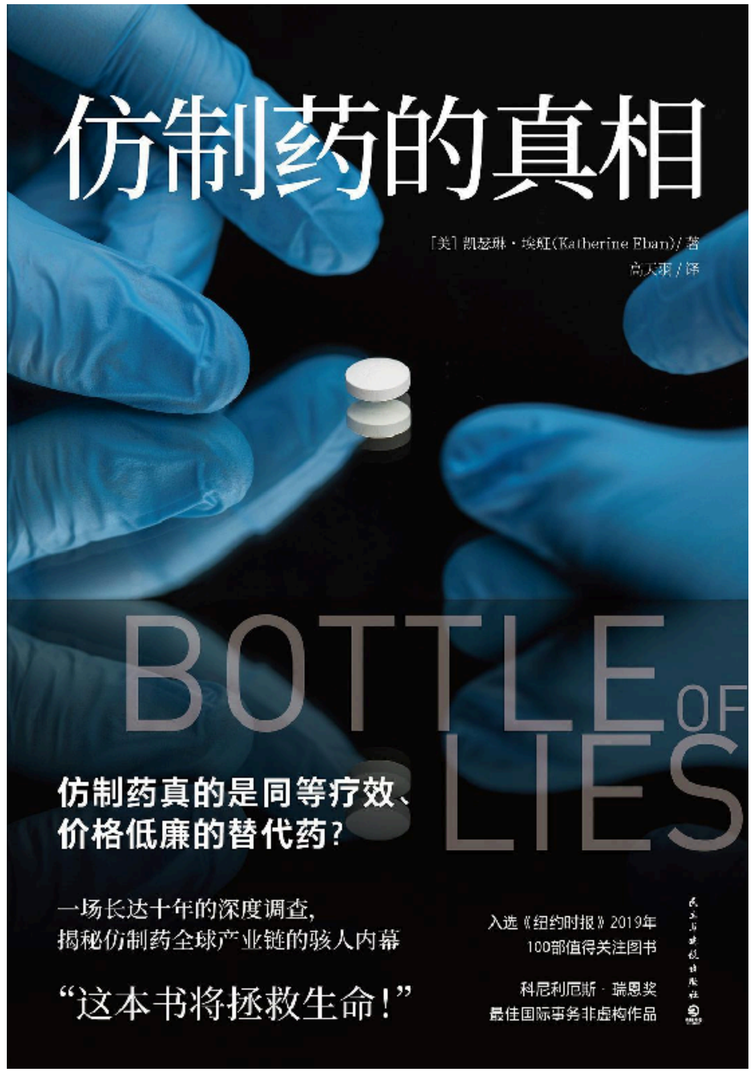

用户体验指标
TTI
首次可交互时间，连续五秒内没有长任务和两个以上的 GET 请求时，那五秒前的最后一个长任务结束时间就是 TTI 记录的时间
FID
首次输入延迟，记录在 FCP 和 TTI 之间用户与页面交互时响应的延迟
TBT
阻塞总时间，记录在 FCP 和 TTI 之间长任务的阻塞时间
FP
首次绘制，页面第一次绘制元素的时候
FCP
首次内容绘制，首次绘制文本、图片、非空白 Canvas 或 SVG 的时间
LCP
最大内容绘制，视窗内最大的元素绘制的时间
CLS
累计位移偏移，记录了页面上非预期的位移波动

获取指标方式有很多

- Dev Tools
- Lighthouse
- 官方插件
- Performance API

如何通过 API 做性能检测

如何针对差指标做性能优化



我不是药神大家可能看过，讲述了一个仿制药拯救了很多人的故事。但是仿制药真的如这部电影说的如此嘛？到底有哪些东西是被这部电影掩盖掉的?

感谢聆听

欢迎大家私聊交流

另外我司上海、杭州、成都、北京、广州

招人，有兴趣的聊聊

前端早早聊大会直播

2020 PK 2021

已举办 16 期 100 场

计划举办至少 20 期 140 场

1/11 前端转管理	6/20 前端跨端跨栈
2/29 前端搞基建	6/27 前端女生专场
3/28 前端搞搭建	7/18 前端搞可视化
4/11 前端搞规划	8/15 前端搞构建
4/25 前端搞监控	8/29 前端成长晋升
5/16 Serverless	9/26 前端搞报表
5/30 前端搞微前端	10/17 前端搞组件
5/31 前端搞面试	11/21 前端搞框架
6/13 前端搞文档	12/26 前端搞性能

1/10 前端搞副业	5/15 前端搞互动
1/23 前端搞管理	5/29 跨端 Flutter
2/06 前端搞小程序	6/12 前端搞可视化
2/27 可视化搭建	6/26 前端搞 WebGL
3/06 前端搞搭建	7/10 Webassembly
3/20 前端搞面试	7/24 前端搞 BFF
3/27 菜鸟大前端	8/28 前端搞安全
4/10 CI/CD	9/11 Serverless
4/24 前端搞算法	9/25 前端搞 IoT
5/08 前端搞述职	10/16 前端搞监控
New! 11/20 前端搞 IDE	New! 12/11 玩转 Node.js

(以实际举办为准，行程/话题/场次会做动态调整)

早

一招鲜走天下 组合拳闯四海
单主题多讲师 听得懂抄得走

前端早早聊大会

2021年票

解锁 2021 年 140 场干货技术直播

单场大会用户	年票 VIP 用户
平均每期 77 元	平均每期 25 元
平均每场 12 元	平均每场 5 元
-	不限次数发布招聘
-	获得优质简历模板
-	Scott 简历指导及内推
-	2022 年票 <= 7 折
-	其他神秘福利...

￥660

每个月有直播
每一场有录播

早

第二十七届前端早早聊大会

Web 渲染 | UI 框架 | 性能 | Flutter 跨端实践

6 月 5 日
下午直播

学义	政采云	政采云移动端 核心开发者	《如何借助 Flutter 从 0 到 1 重构原生移动端》	13:00
染陌	淘宝	「北海 Kraken」 核心开发者	《如何深度实践基于 Flutter 的 Web 渲染引擎》	14:00
之航	满帮	「Thresh」 核心开发者	《如何基于 Flutter 引擎构建 TS 应用》	15:00
核潜艇	京东	JD Flutter 核心开发者	《如何探索和定制 Flutter 热重载》	16:00
皓黯	闲鱼	基础链路 客户端研发	《如何提升 Flutter 应用的性能与体验》	17:00
祈晴	闲鱼	闲鱼客户端 核心研发	《如何设计和实现 Flutter IM 架构跨端方案》	18:00

2021 全年行程

- 1/9 自由职业/副业
- 1/23 前端团队管理
- 2/6 小程序|组件化
- 2/27 页面搭建场
- 3/20 前端搞面试
- 4/10 前端搞CI/CD
- 4/24 前端搞算法
- 5/09 前端搞述职
- 5/15 前端搞互动
- 5/29 跨端 Flutter
- 6/12 数据可视化
- 6/26 前端 WebGL
- 7/10 WebAssembly
- 7/24 前端搞 BFF
- 8/14 前端搞 AI
- 8/28 前端搞安全
- 9/11 Serverless
- 9/25 前端搞 IDE
- 10/16 前端搞监控
- 11/20 玩 Node.js

一招鲜走天下・组合拳闯四海
单主题・多讲师・有录播
无所不聊・**长按扫码报名**

掘金