

◦ Node 全栈

阿里巴巴·前端技术专家

i5ting 狼叔



AI时代的移动技术革新

Era of AI: Innovations in Mobile Technologies

APICloud

个人简介

他们叫我**狼叔**



- 姓名：桑世龙
- 部门：阿里巴巴·大文娱事业部
- 简要介绍：嗷呜。。。。

Node.js布道

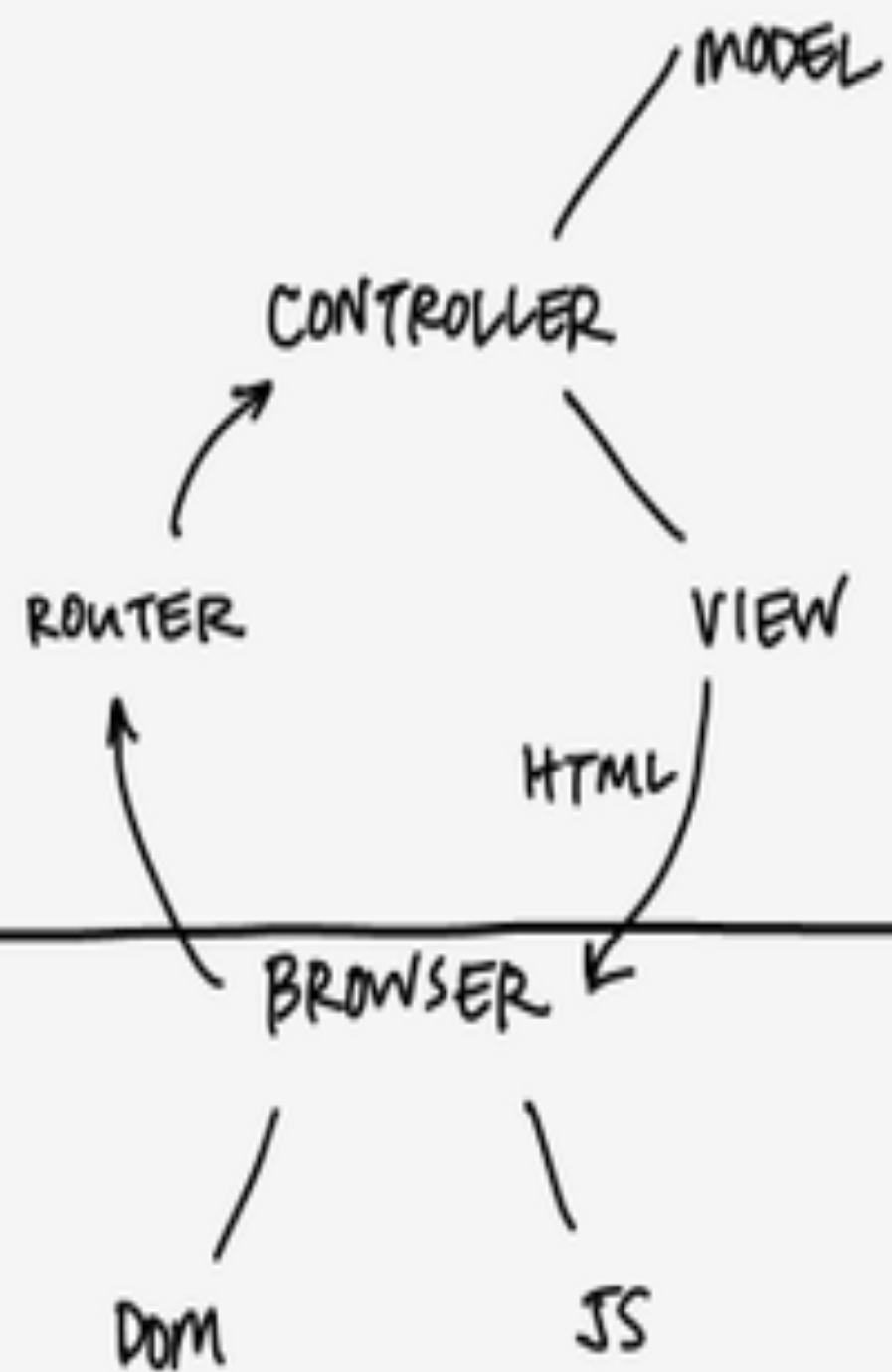
StuQ明星讲师

被坑的CTO

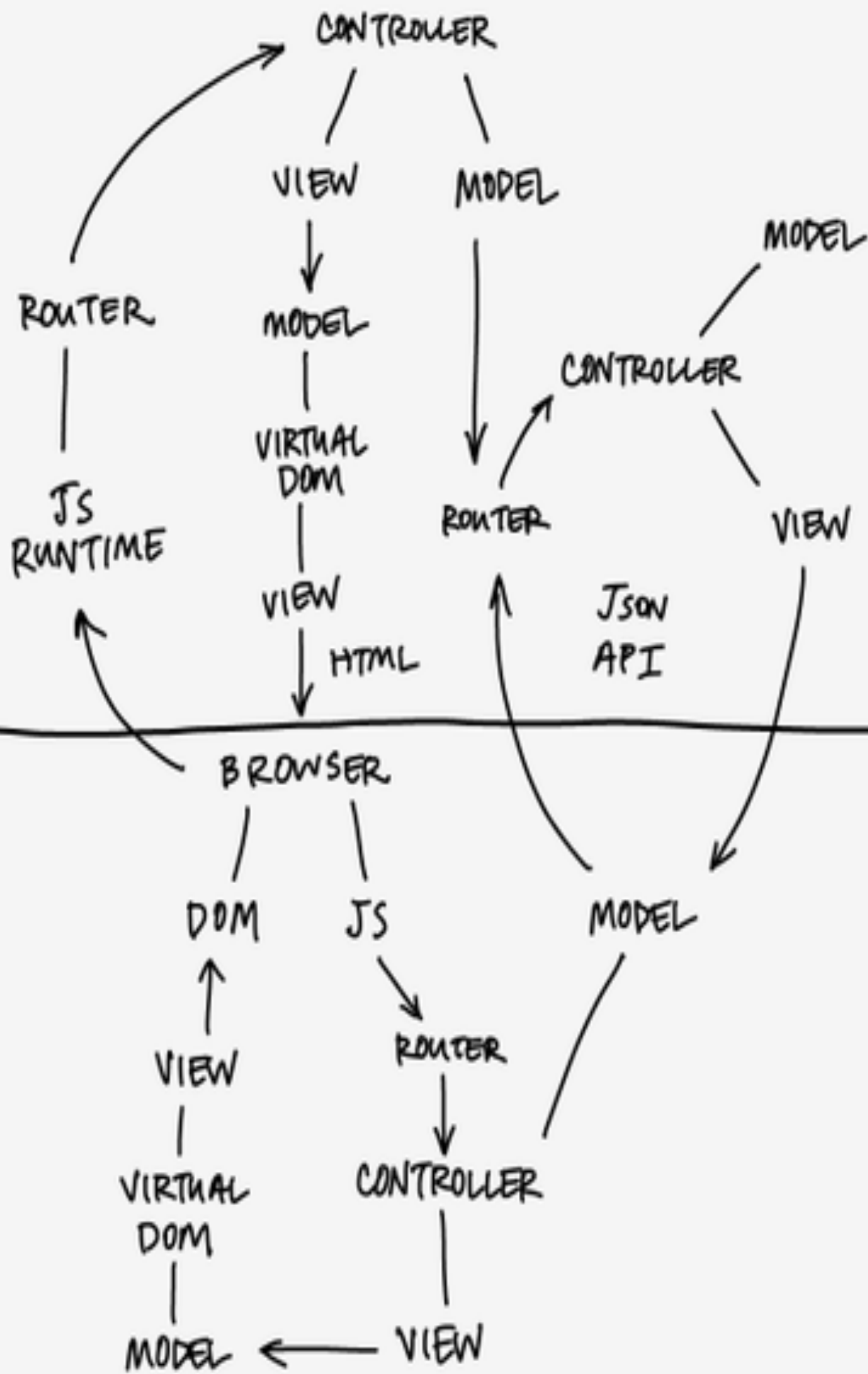
晒娃狂魔



2004



2016



2017

预编译

依赖管理
npm, bower

资源处理
DataURL

压缩
uglifyjs, jsmin, csso

React组件化、Vuejs (未来趋势)

图片压缩
imagemin

js友好语言
coffee, babel, typescript

CSS预处理器

Backbone, Angular

模块系统
CommonJs, AMD, ES6 Modules

流行

JavaS

基础)

模板引擎
jade, handlebars, nunjucks

模块加载器
require.js, jspm, sea, system.js

Cordova

Electron

Webpack

NPM Scripts

CSS预处理
PostCSS、less、sass

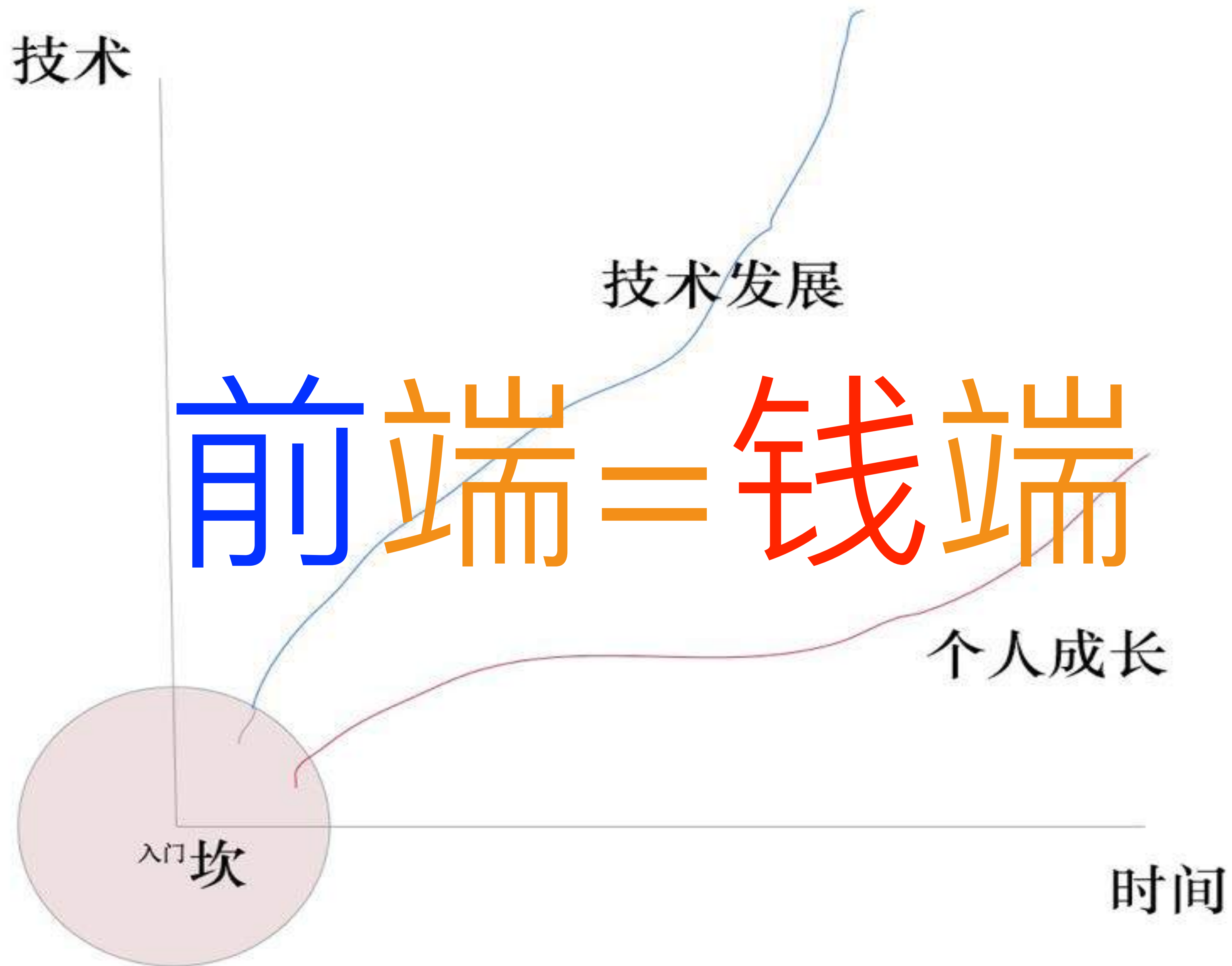
构建工具
grunt, gulp, make

平台工

构建系统

模块打包器
browserify, webpack

现代Web开发





目录

1. AI时代的三端分析
2. 前端工程化构建与Webpack
3. 更了不起的Node.js
4. 我眼中Node全栈



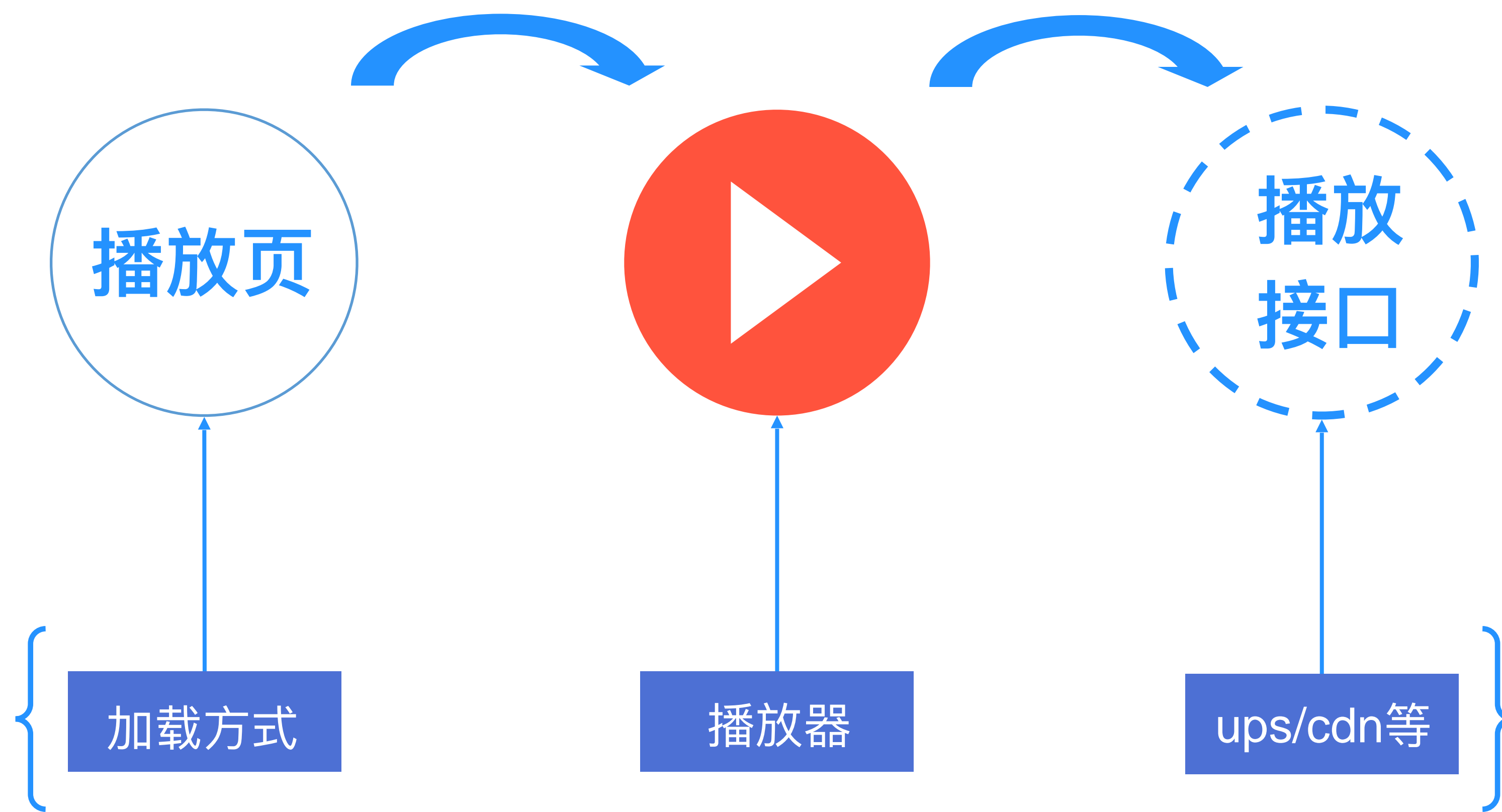


1、AI时代的三端分析

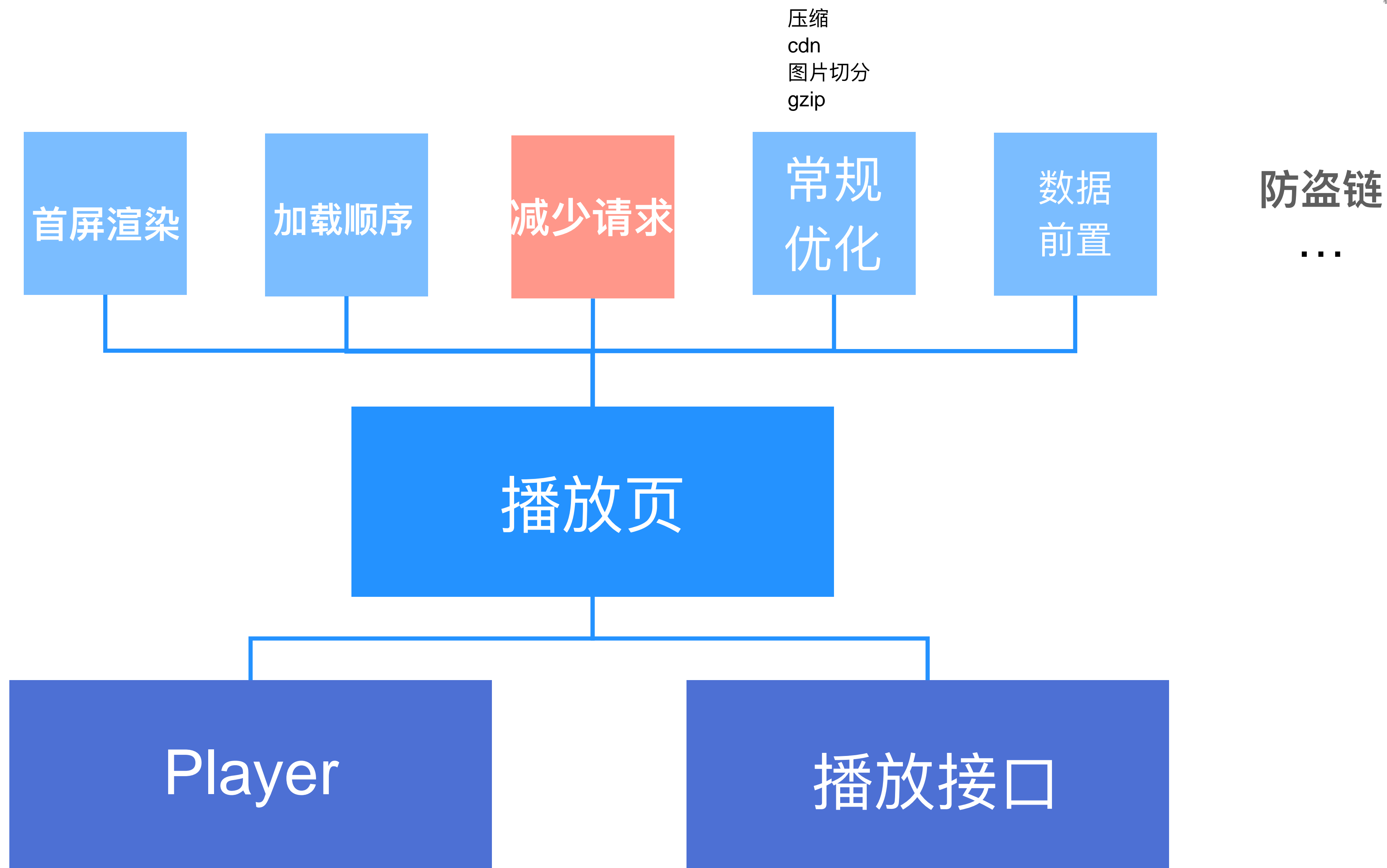


我在做什么

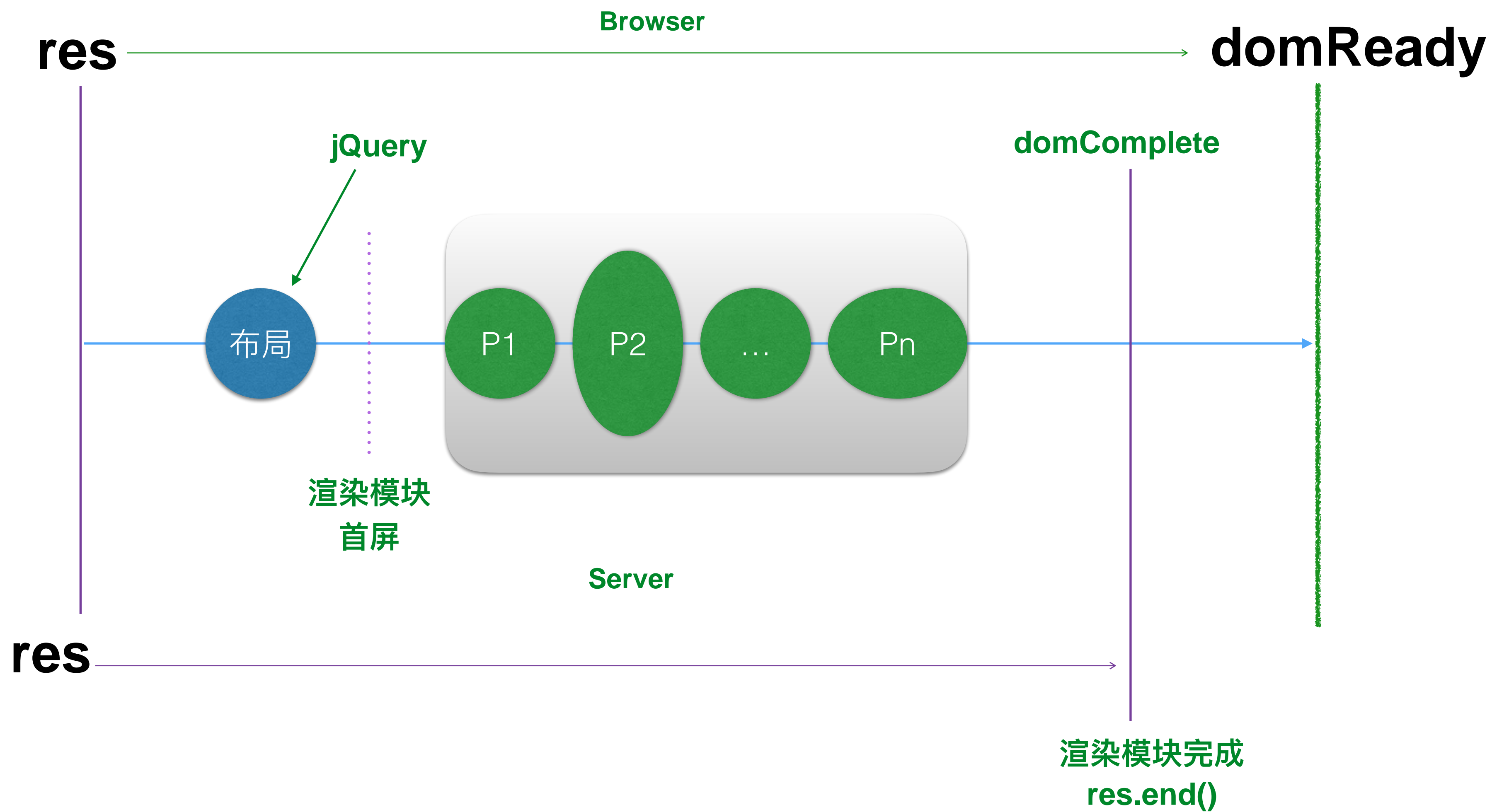
播放器是有前置和后置条件的，前置是播放页加载方式，后置是各种播放接口，非单一优化点



优化无止境



分块加载技术Bigpipe



PC & H5还有前途吗?



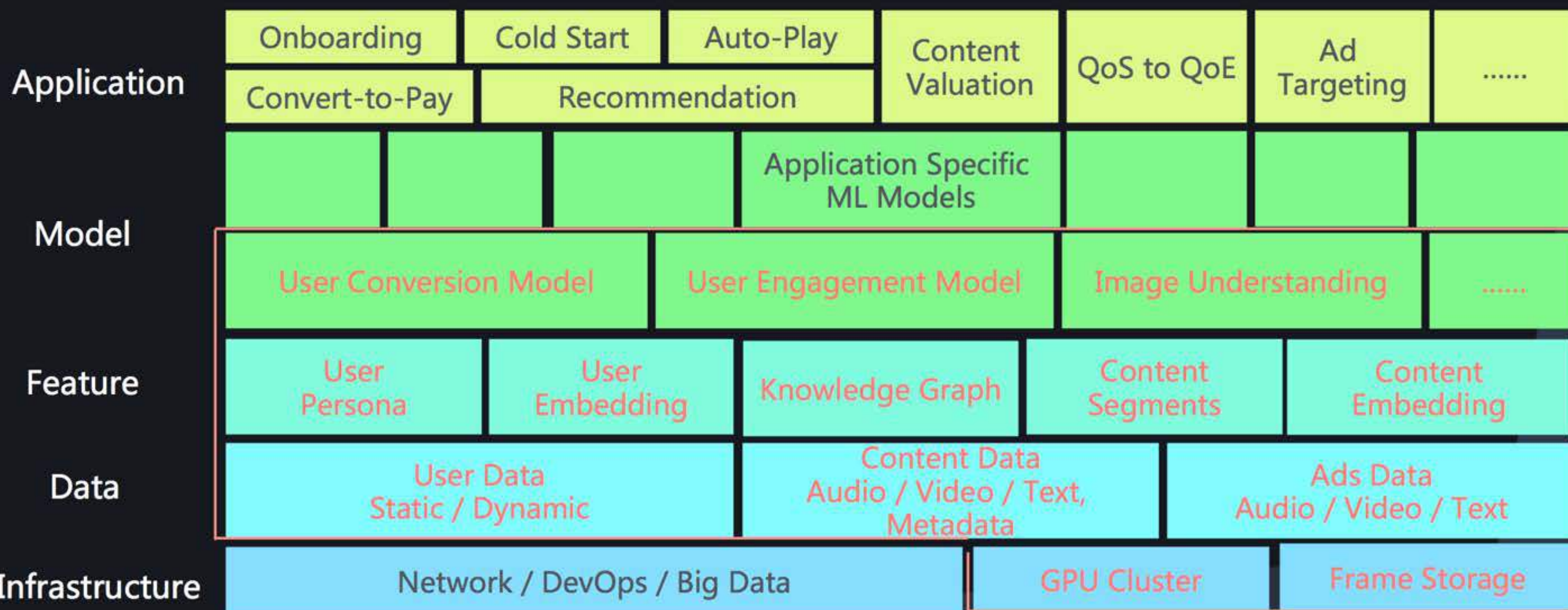
已死?

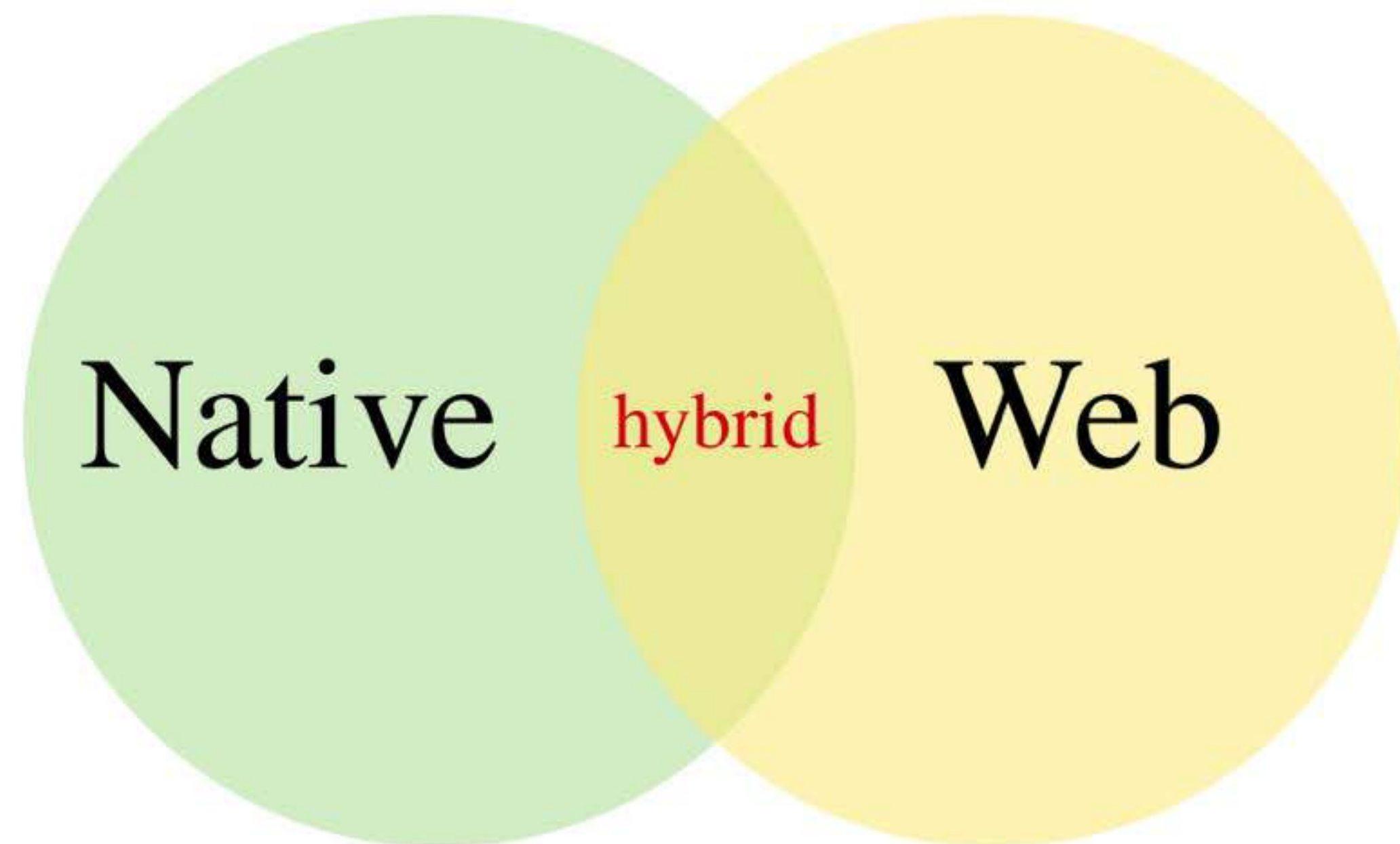
移动互联网时代



太远?

AI Platform – 人工智能平台





从单兵，到组团

ai三端对齐
移动端
pc
pc client

从移动端的发展看未来

native < hybrid < rn/weex < h5

c/s架构到b/s架构，该对瘦客户端予以一定的支持

重点是开发变得无比复杂

Here is the advantage of the layout of the way





2、前端工程化构建与Webpack



依赖下载

- 下载某个库或插件
- 下载它的依赖，以及依赖的依赖
- 无穷尽...

- npm/bower



```
1 | <script src="jquery.js"></script>  
2 | <script>$.uiBackCompat = false;</script>  
3 | <script src="jquery-ui.js"></script>
```

window

eval

iframe

\$.getScript()

- 使用标准的模块系统来处理依赖和导出
- 每个文件是一个模块
- 使用模块加载器或打包器进行处理

AMD, CommonJS, ES6 Modules

- require("..."): Loads module, returns exports
- require("./helpers.js"): ... by relative path, returns exports
- require("jquery"): ... from dependencies manager folder
- exports or module.exports export object

```
var $ = require("jquery");  
  
exports.doSomething = function() {  
    return 42;  
}
```

module
exports
require
global

```
define("module", ["dep1", "dep2"], function(d1, d2) {  
    return someExportedValue;  
});  
  
require(["module", "../file"], function(module, file) { /* ... */ });
```

CommonJS 中逐渐分裂出了 AMD，专门用于浏览器的
因为浏览器没有io读写api

<https://github.com/amdjs/amdjs-api/wiki/AMD>

核心代码

Amd

- define
- require

```
void function() {
    var mapping = {}, cache = {};
    window.define = function(id, func) {
        mapping[id] = func
    };
    window.require = function(id) {
        if (!/\.js$/.test(id)) {
            id += ".js"
        }
        if (cache[id]) {
            return cache[id]
        } else {
            return cache[id] = mapping[id]()
        }
    }
}

define("test.js", function(exports) {
    exports = {};

    exports.var = 1;
    exports.say = function(){
        alert(this.var);
    }

    return exports;
});
```

cache

按需加载

以前的做法

- 使用多个<script>标签加载
- 手动管理顺序
- 手动管理加载哪些

```
<script src="/wreq.js"></script>  
<script src="/js/entry.js" type="text/wreq"></script>  
  
<div id="res">[]</div>
```

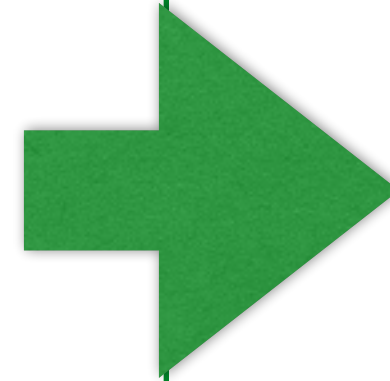

现在的做法

开发环境

产品环境

module loader

module bundler



依赖管理

模块系统

- runs in the browser and loads modules when they are requested
- easy to use
- less optimized for production usage

- runs in preparation and bundles modules into static files
- needs a preparation/build step
- more optimized for production usage

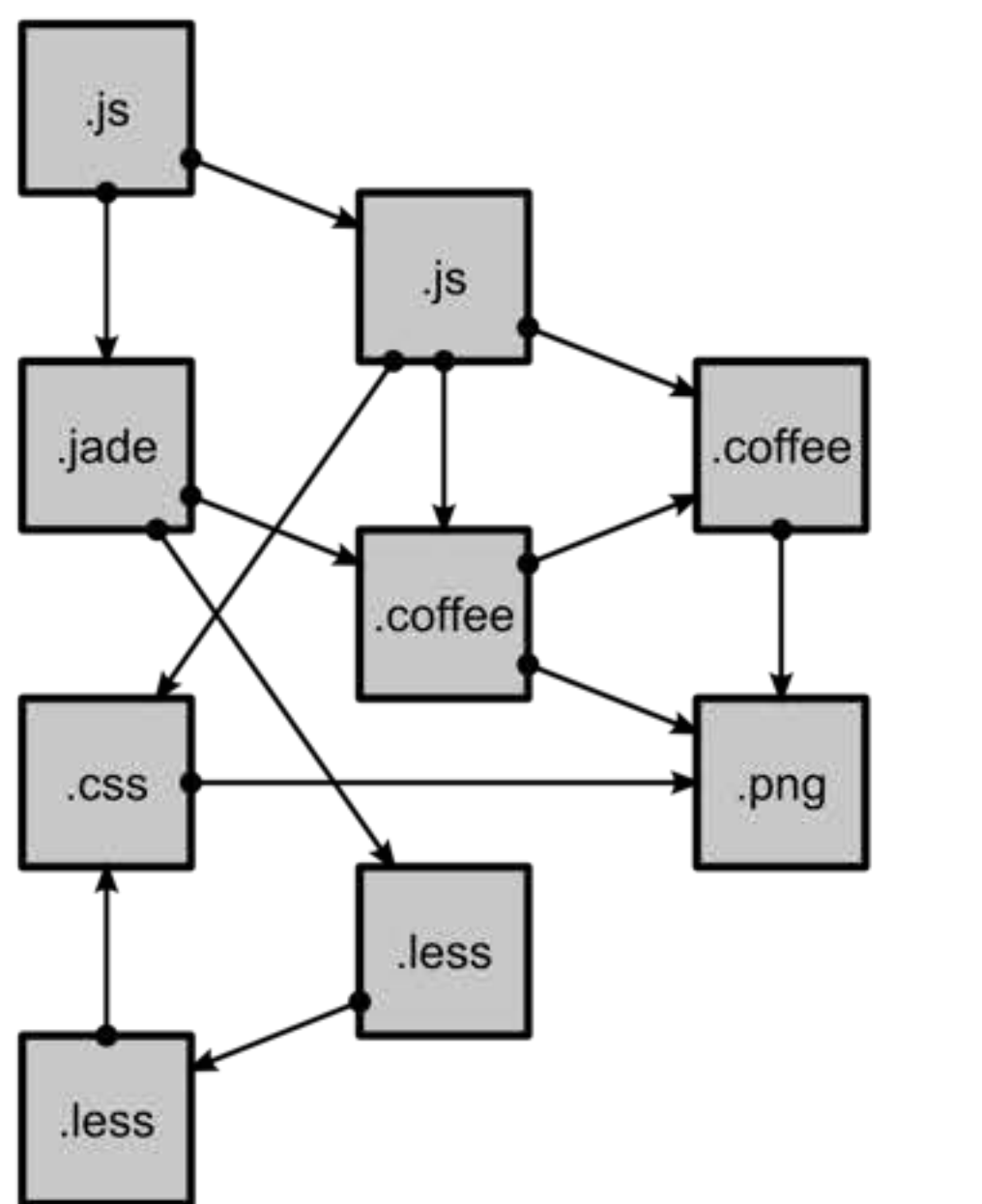


grunt/gulp

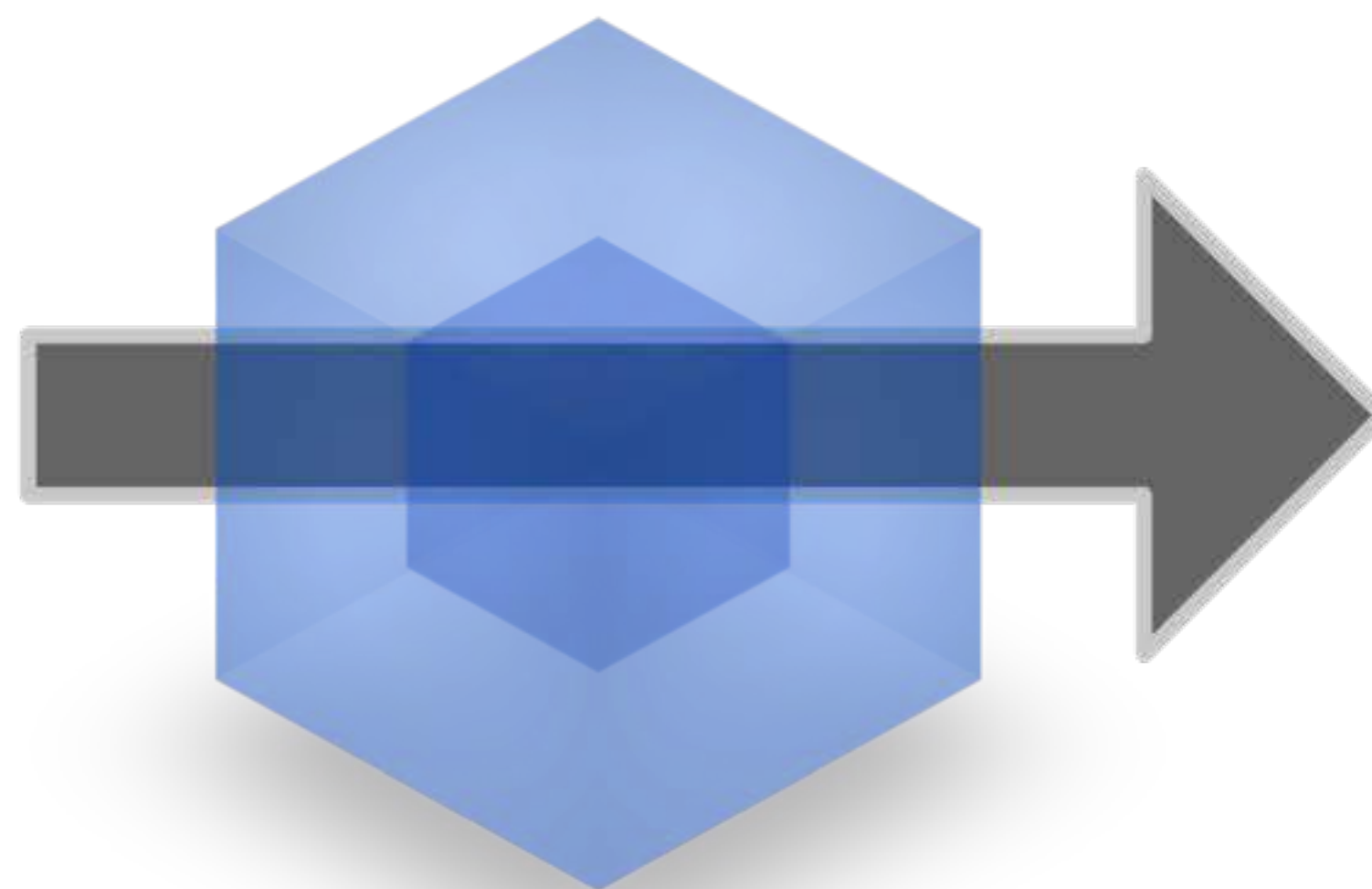
require.js
systemjs

r.js\browsersify\
webpackjspm

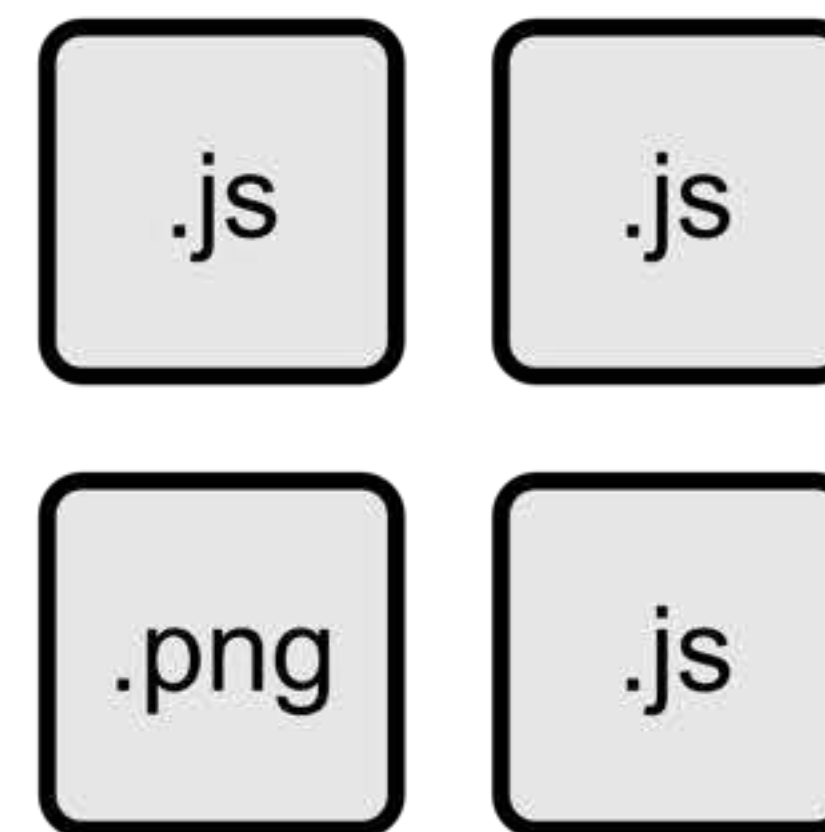
所有资源都是模块



modules
with dependencies



webpack
MODULE BUNDLER



static
assets

Webpack打包过程

- 从配置文件里找到entry point
- 解析模块系统
- 解决依赖
- 模块依赖处理（读取，解析，解决）
- 合并所有使用的模块
- 合并模块系统的运行时环境
- 产生打包后的文件

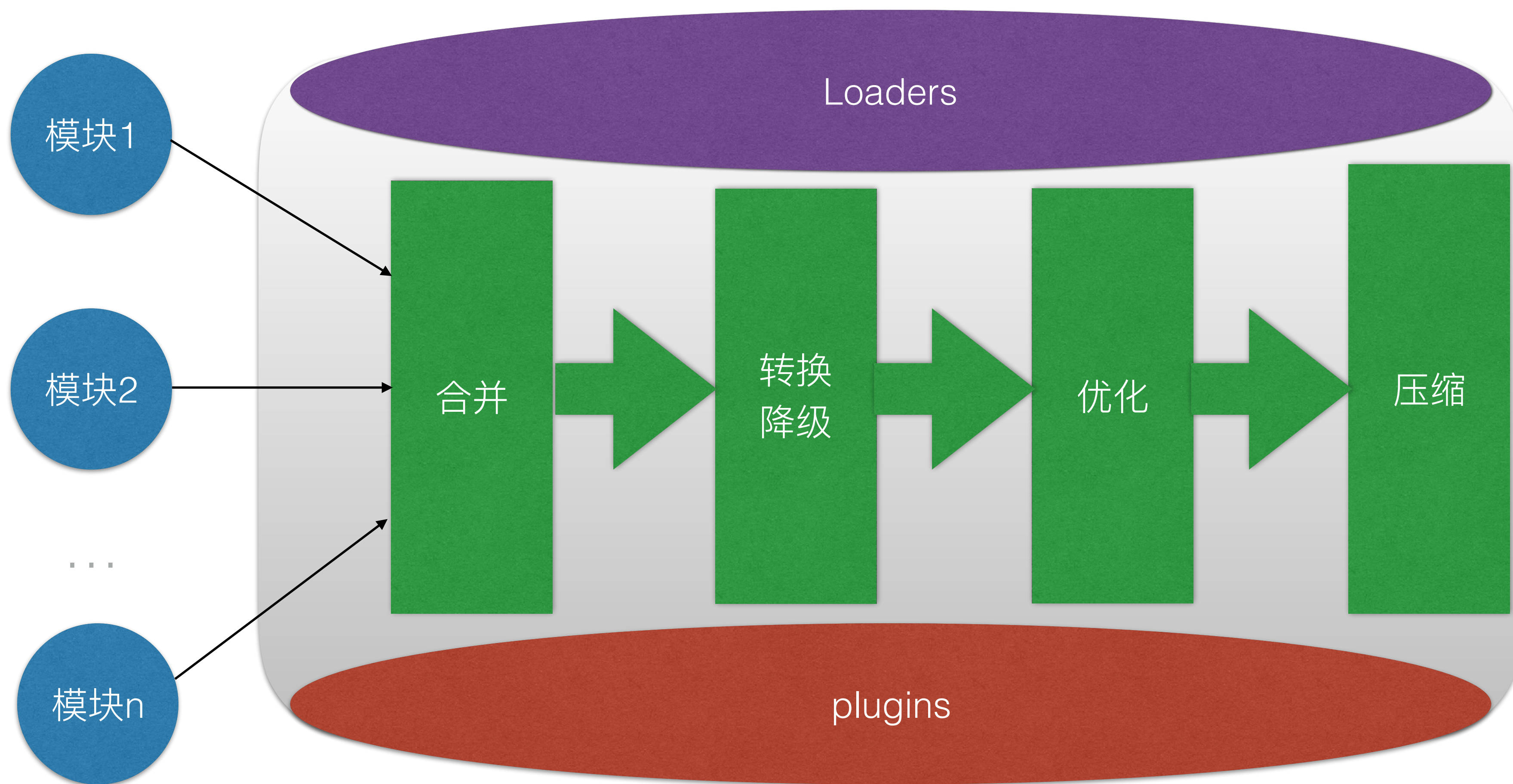
```
1 webpackJsonp([1], [
2     /* 0 */
3     /***/
4     (function (module, exports, __webpack_require__) {
5
6         __webpack_require__(1)
7
8         __webpack_require__.e /* require.ensure */ (0).then((function (require) {
9             __webpack_require__(6)
10        }).bind(null, __webpack_require__)).catch(__webpack_require__.oe)
11
12        /***/
13    }),
14    /* 1 */
15    /***/
16    (function (module, exports) {
17
18        console.log('a')
19
20        /***/
21    })
22 ], [0]);
```

浏览器里解包过程

- 通过<script>加载webpack打包后的文件
- 加载模块运行时环境
- 加载entry point
- 读取依赖
- 解决依赖
- 执行（带有依赖的）entry point

1. `<script>` 混乱加载
2. 各种模块系统标准, commonjs/amd/es6 module
3. 模块加载器, requirejs/sea/systemjs
4. 模块打包器, webpack/r.js/jspm/browserfy
 - A. 合并入口, 对外暴露的只有entry point
 - B. 提供浏览器运行环境 (内置模块加载器)
 - C. 优化 (tree-shaking、DCE无用代码移除等)

时髦的开发



你只管写就好了

其他的webpack来

你能写好，就好了。。。

你能写，就好了。。

你能，就好了。。

你，就好了。。

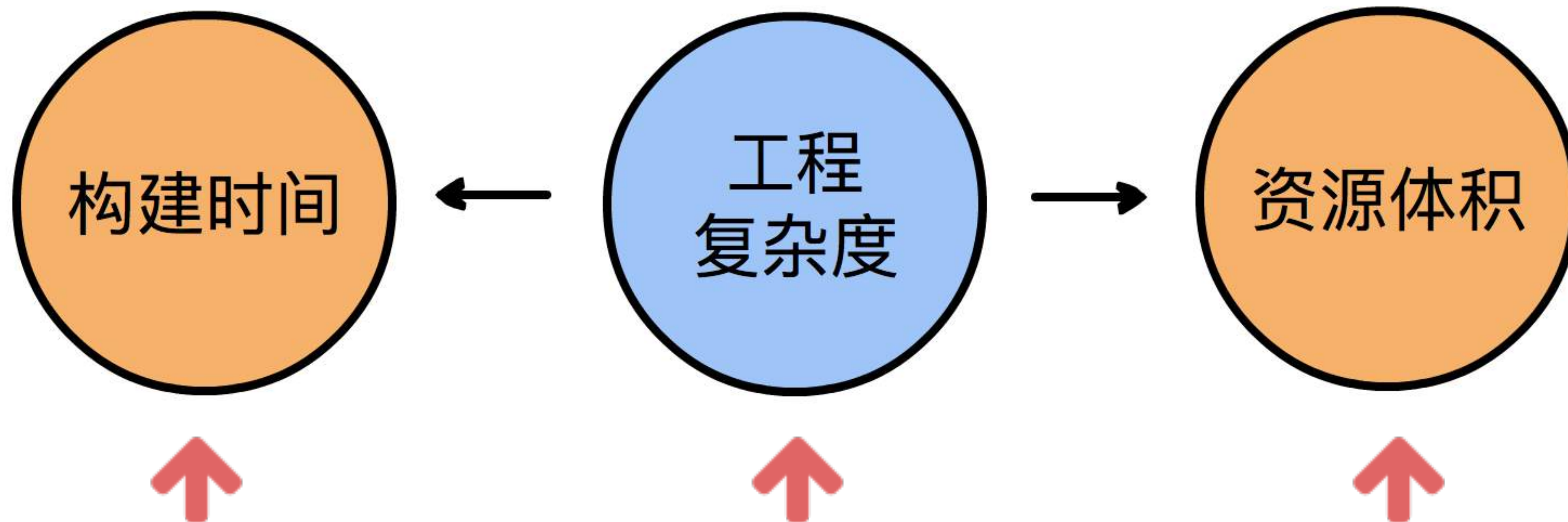
打包器的演进

- browserify
- webpack 1
- rollup
- webpack 2
- Parcel

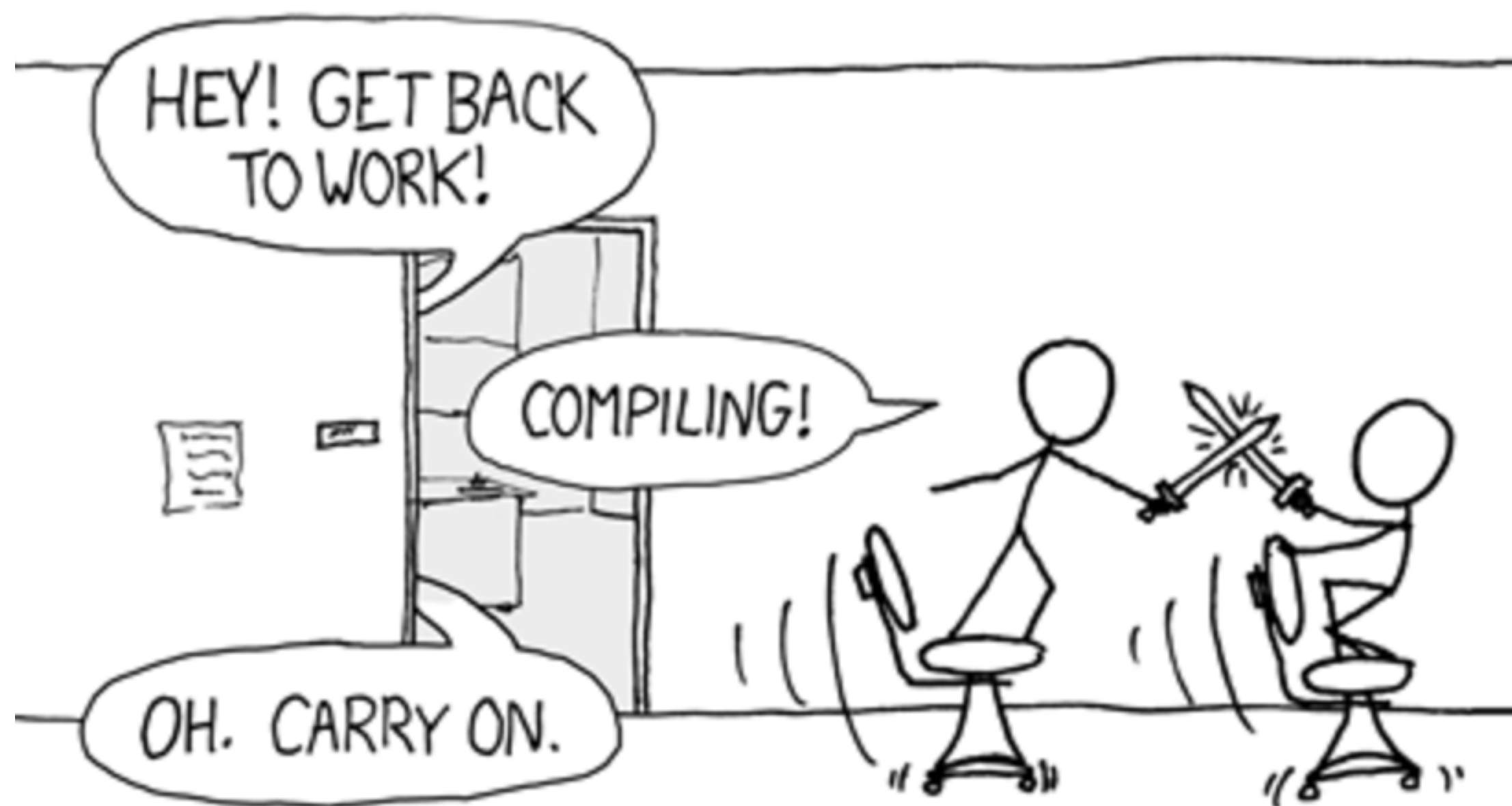
commonjs规范

多种模块

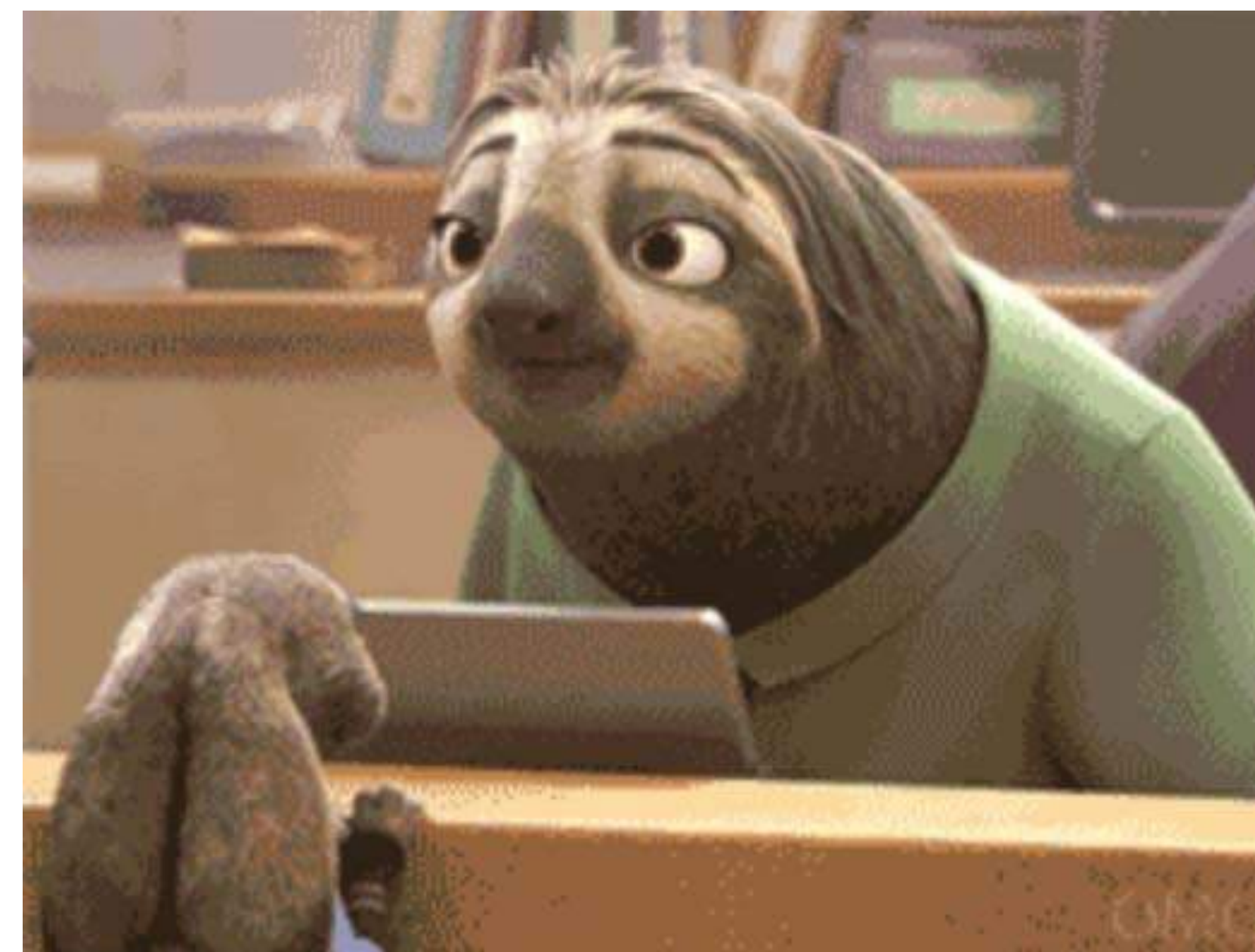
利用es6模块能静态分析语法树的特性，只将需要的代码提取出来打包，能大大减小代码体积



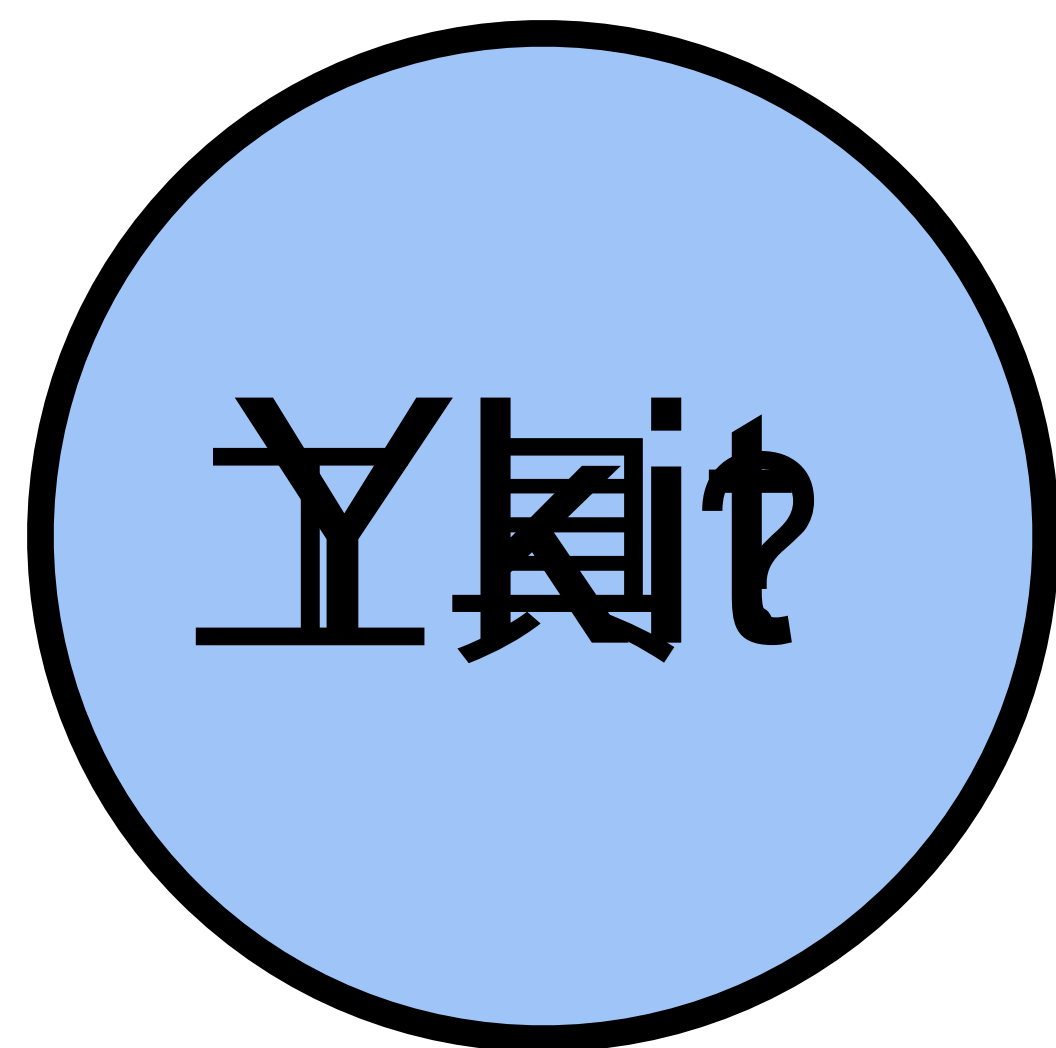
|| 当下面临的问题



开发测试成本增加

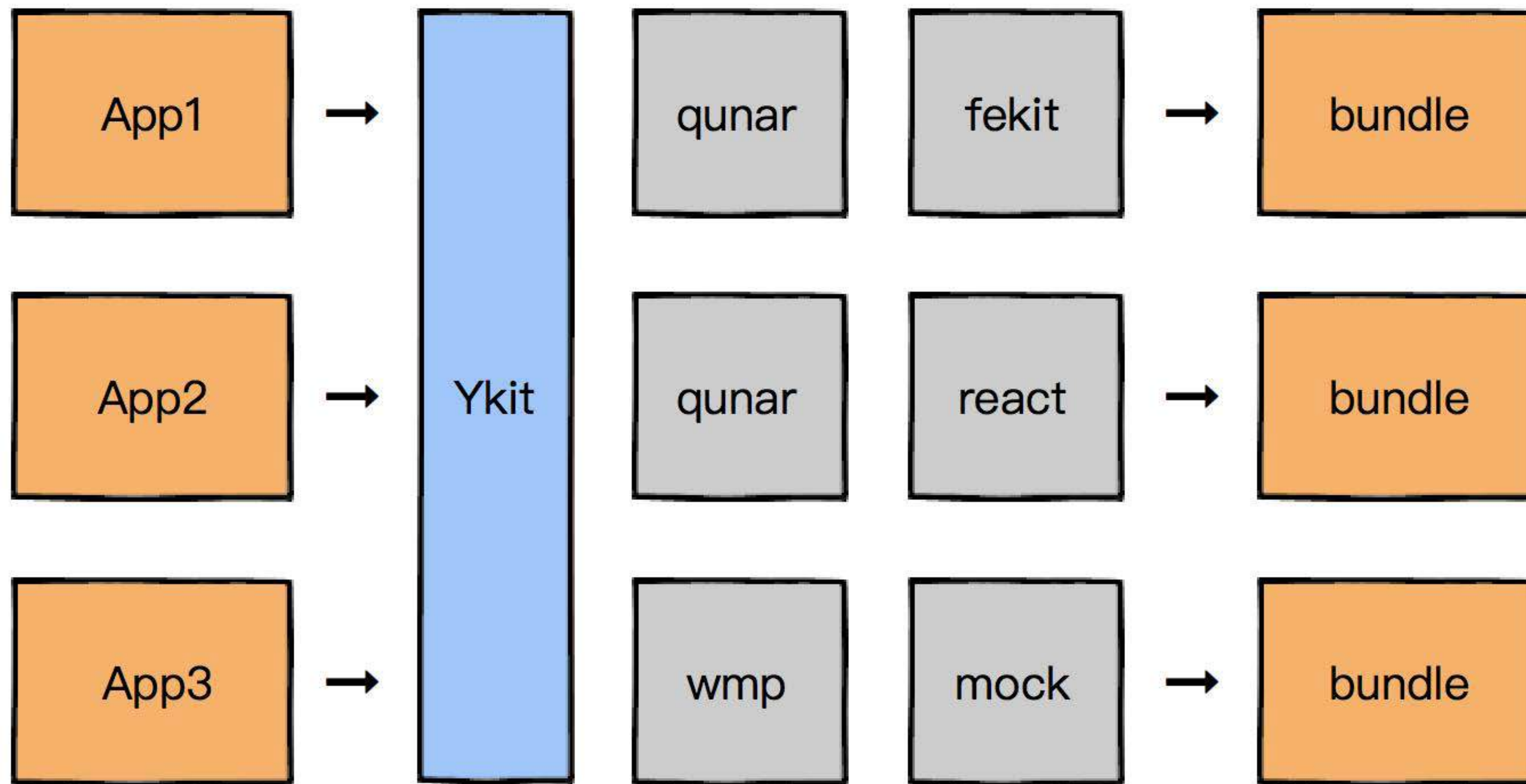


客户端性能受影响

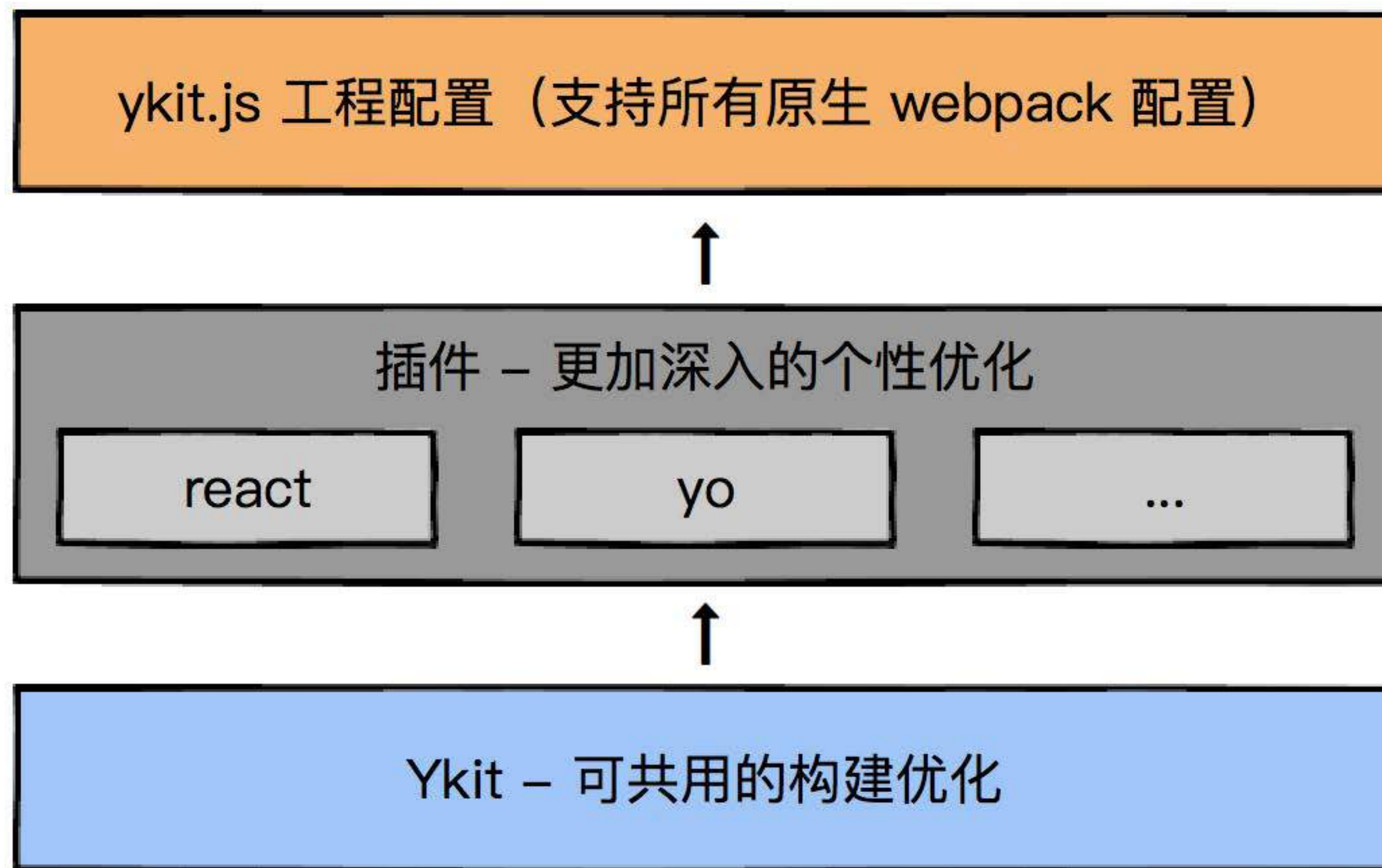


使业务同学
专注于研发

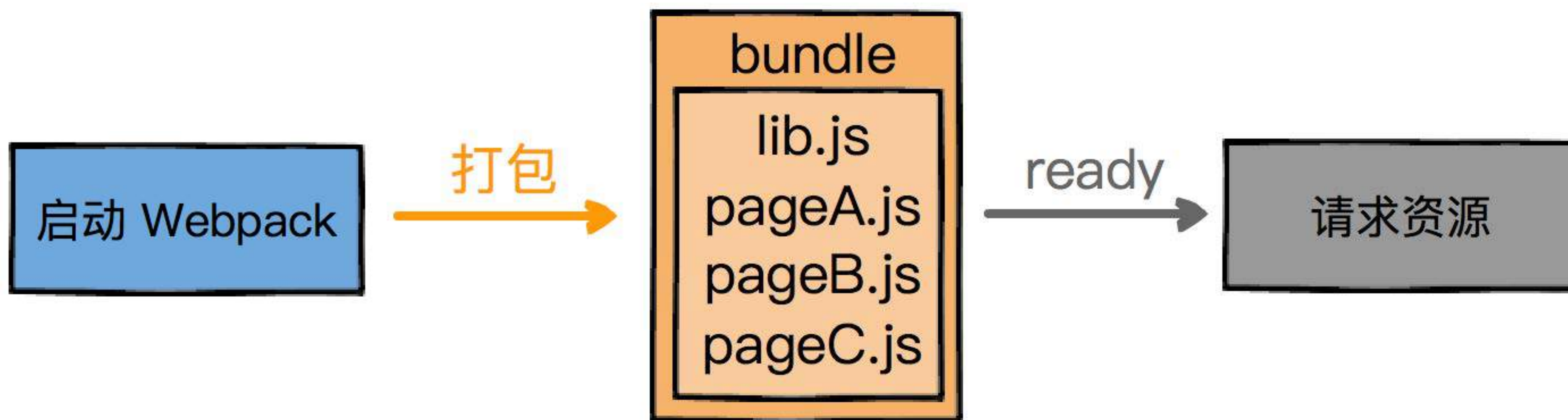
YKit工作原理

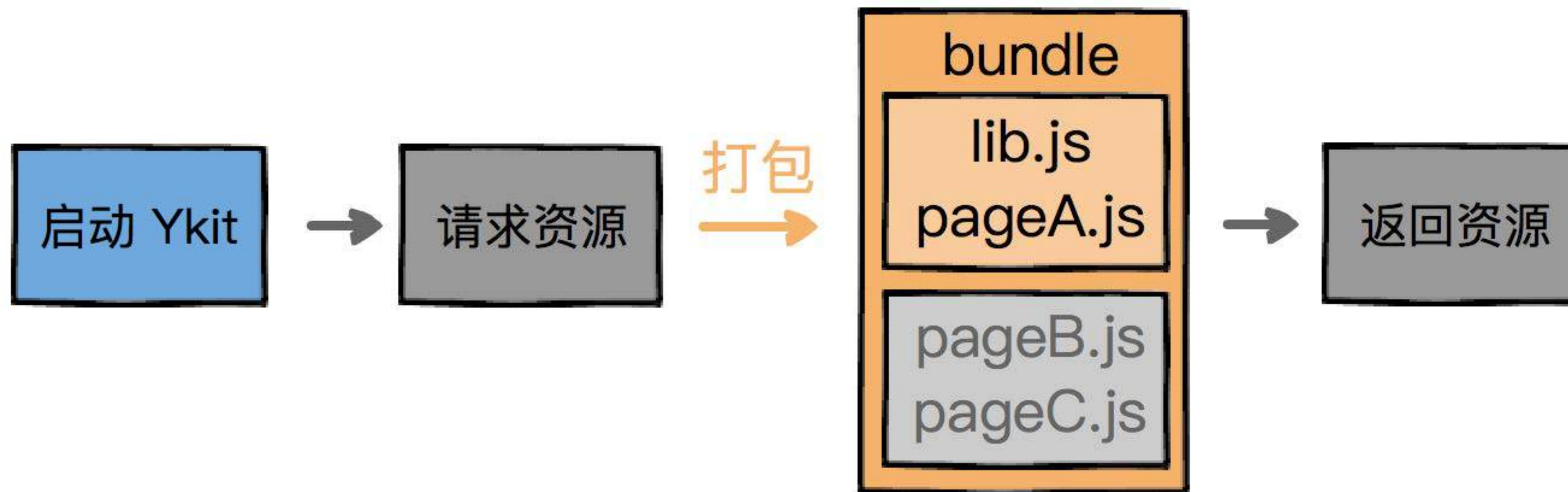


构建优化



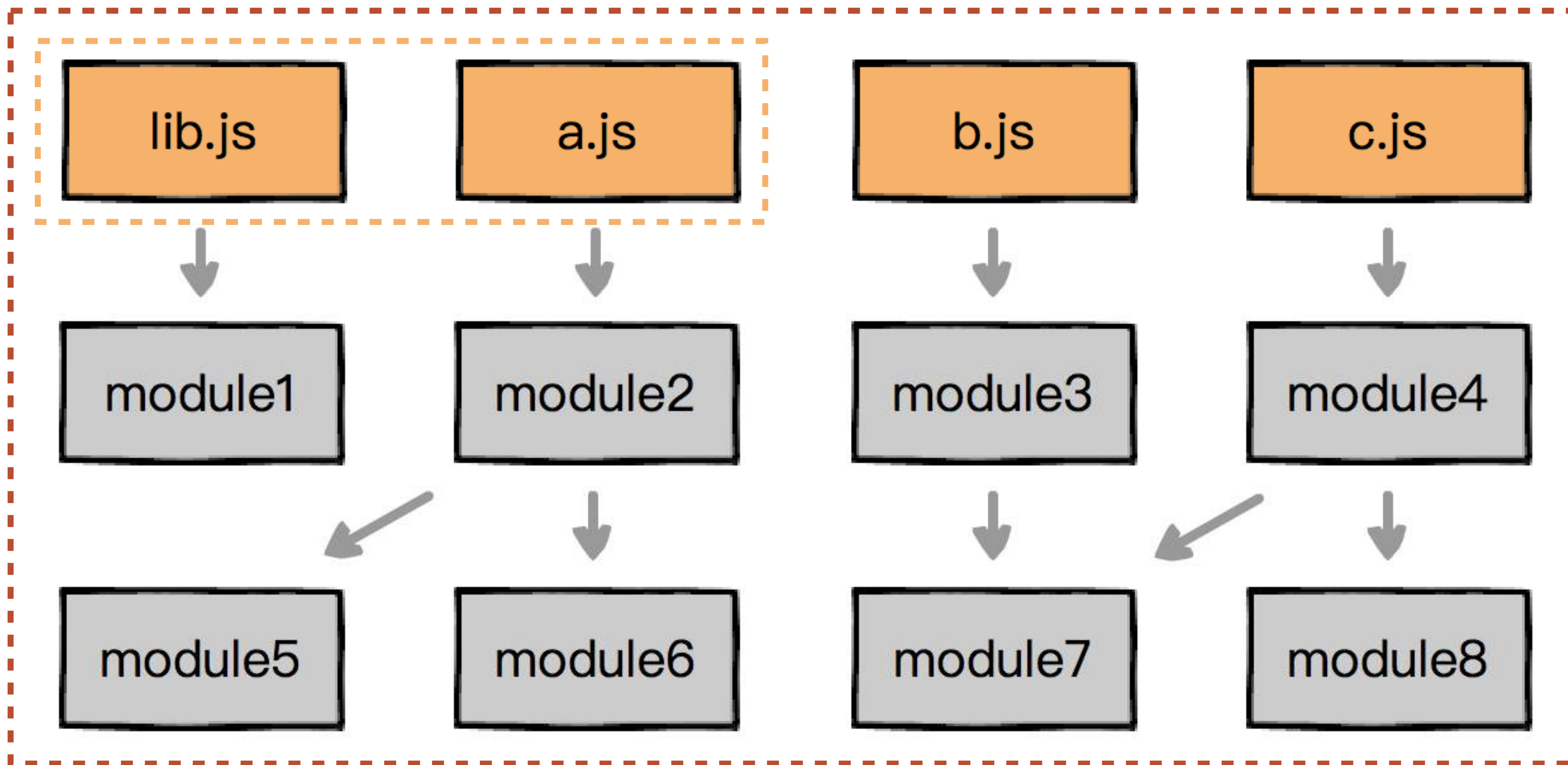
Webpack本地服务



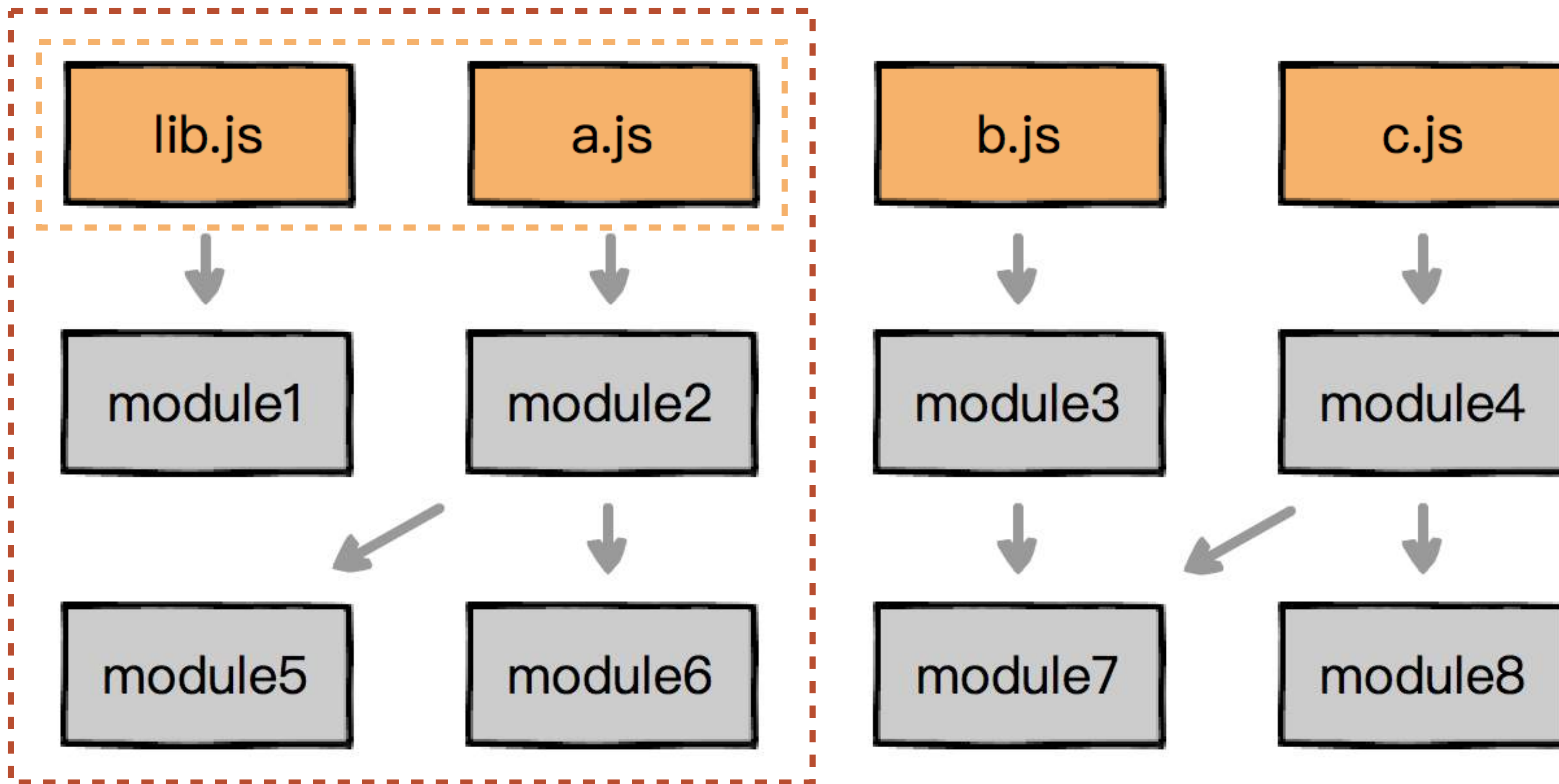


Ykit 会根据请求的资源进行入口过滤，仅打包页面所请求的资源

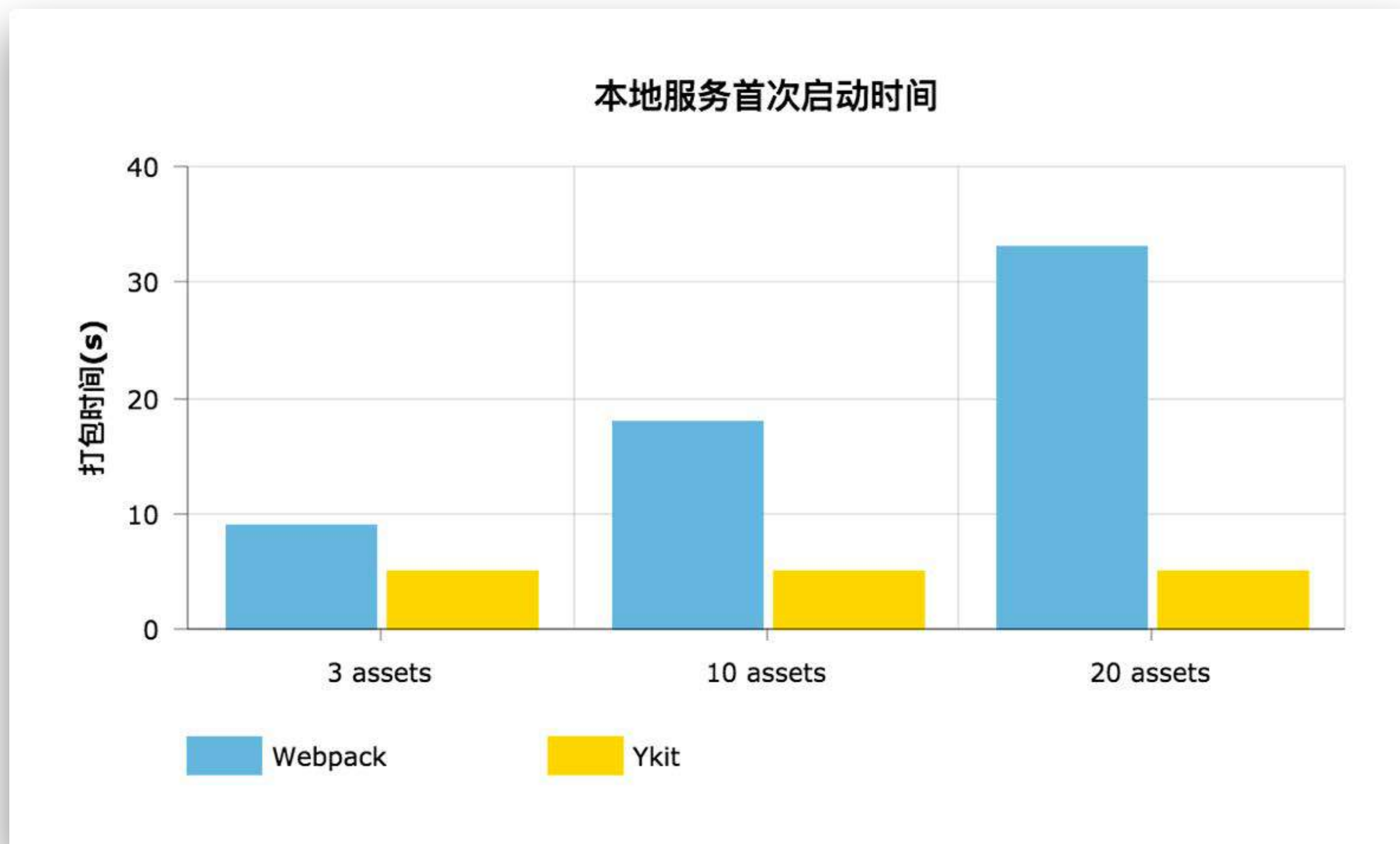
Webpack本地服务



YKit本地服务

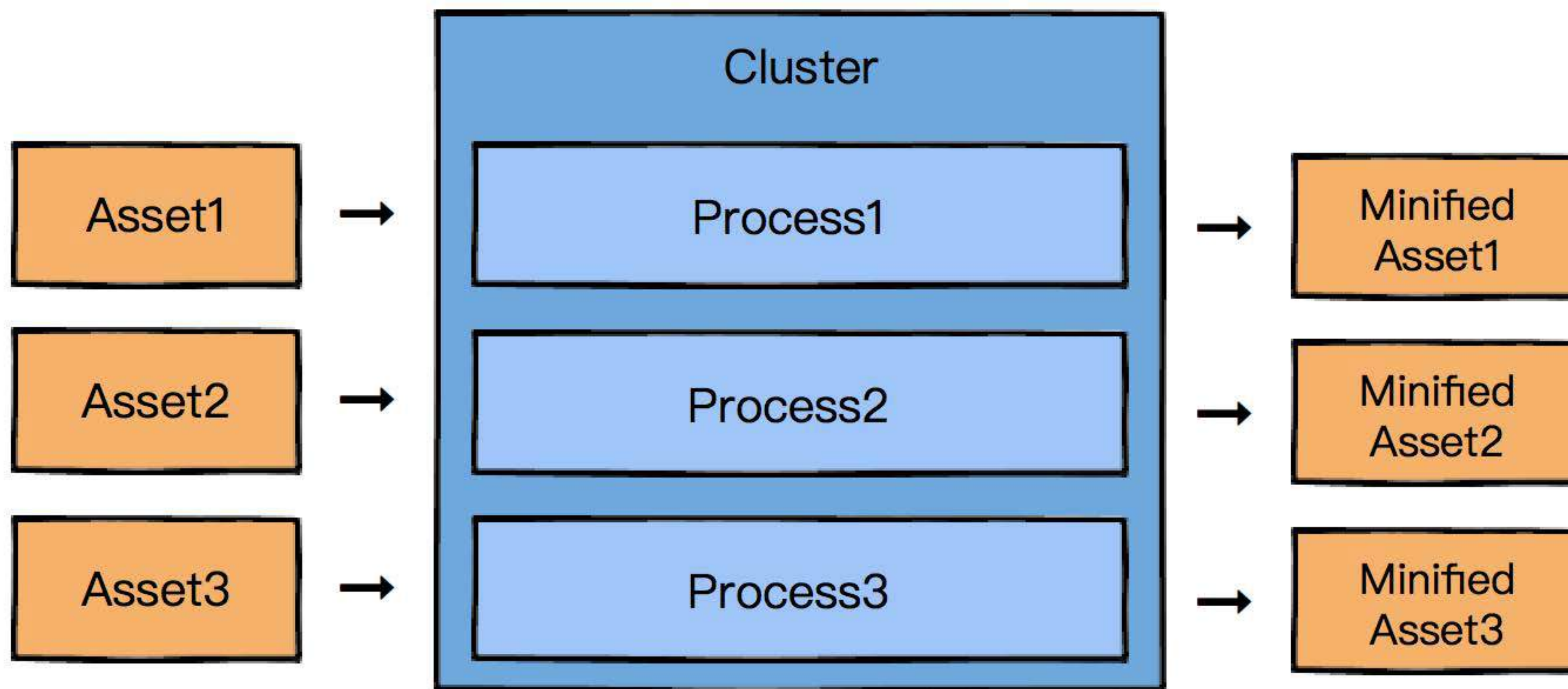


效果对比

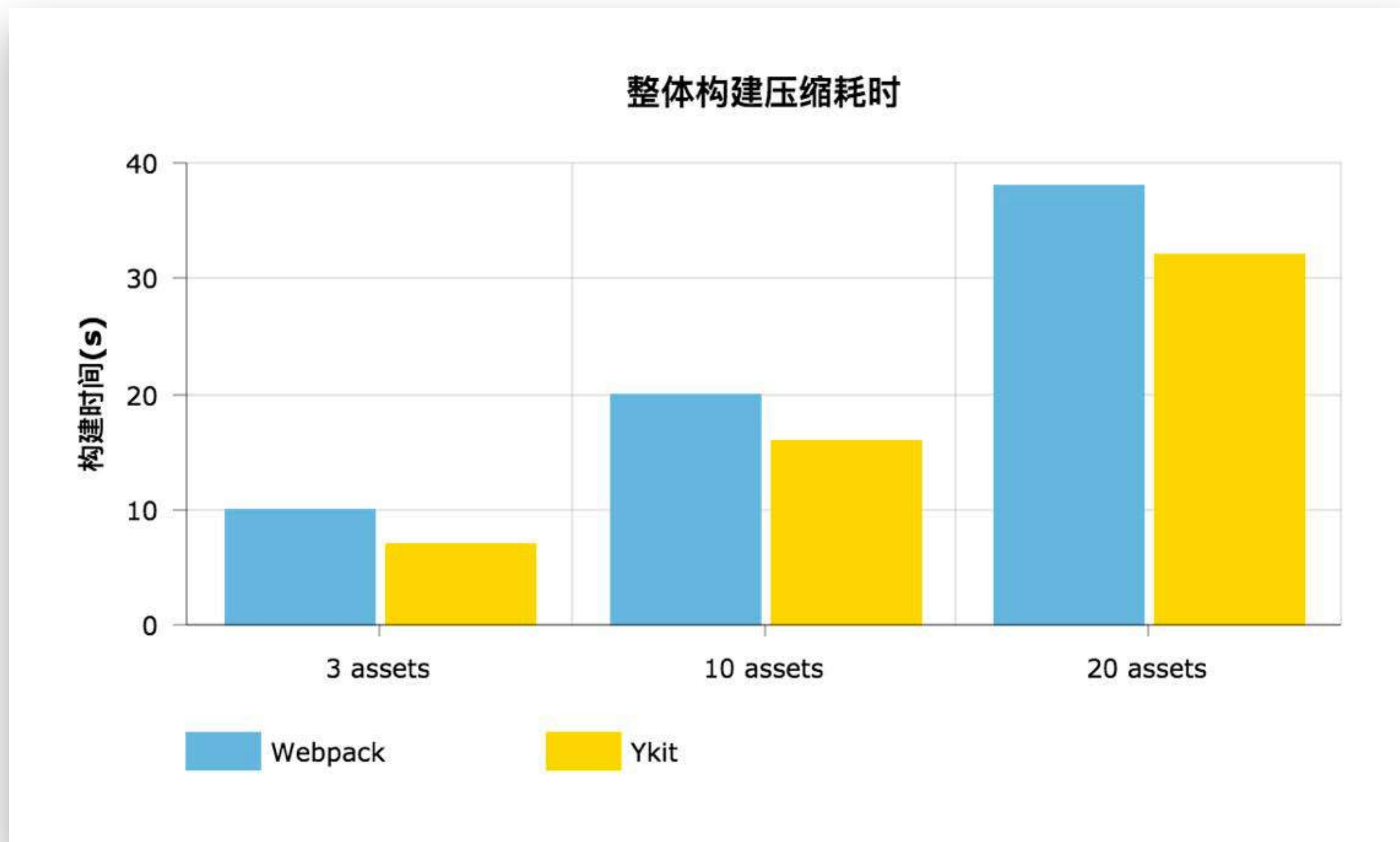


压缩优化

- 使用进程集群，一个进程压缩一个资源
- 防止 Webpack 内存溢出崩溃



效果对比



你觉得，复杂么？



3、更了不起的Node.js



一定有你不知道的应用场景

- 网站 (如express/koa等)
- im即时聊天(socket.io)
- api (移动端, pc, h5)
- HTTP Proxy (淘宝、Qunar、腾讯、百度都有)
- 前端构建工具(grunt/gulp/bower/webpack/fis3...)
- 写操作系统 (NodeOS)
- 跨平台打包工具 (PC端的electron、nw.js, 比如钉钉PC客户端、微信小程序IDE、微信客户端, 移动的cordova, 即老的Phonegap, 还有更加有名的一站式开发框架ionicframework)
- 命令行工具 (比如cordova、shell.js)
- 反向代理 (比如anyproxy, node-http-proxy)
- 编辑器Atom、VSCode等



更了不起的Node.js

跨平台 01

前端 (web+h5)

移动端 (hybrid)

PC端

Node后端 02

核心特性、Web应用、Api
rpc、测试、部署、最佳实践
微服务、厂商支持



03 前端

react\vue\angular

应用实践

架构

04 工具

各种预编译、构建工具

webpack/gulp、工程化,

hack技巧、npm等

做合适的事儿

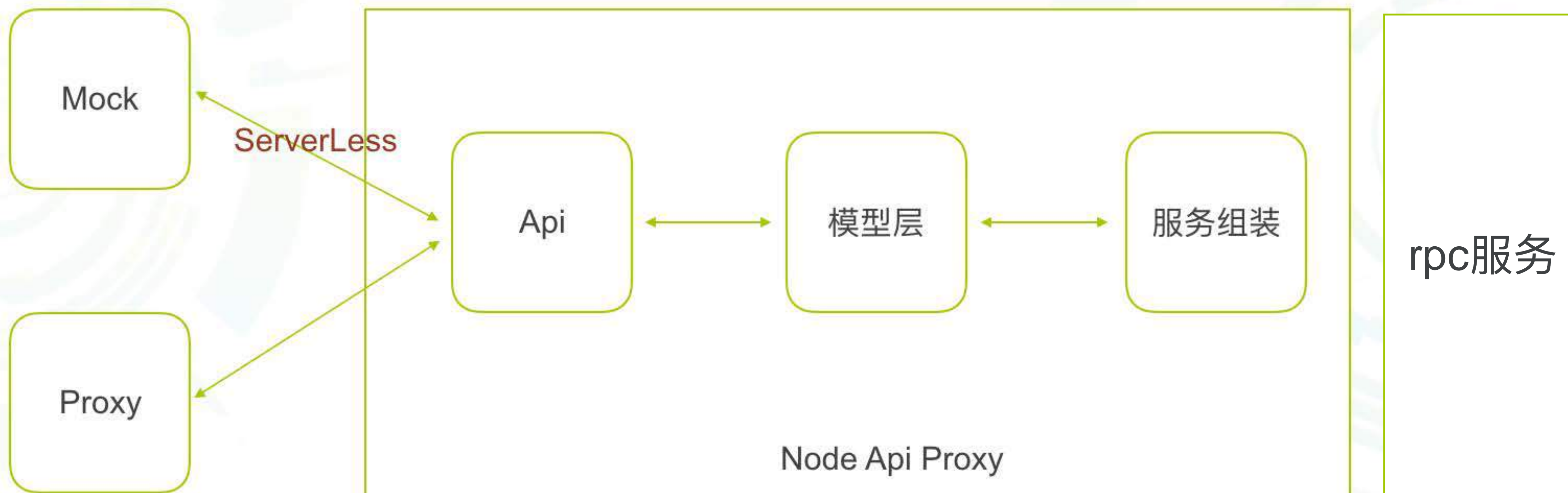
大前端 -> Api -> oltp(node | java | go)

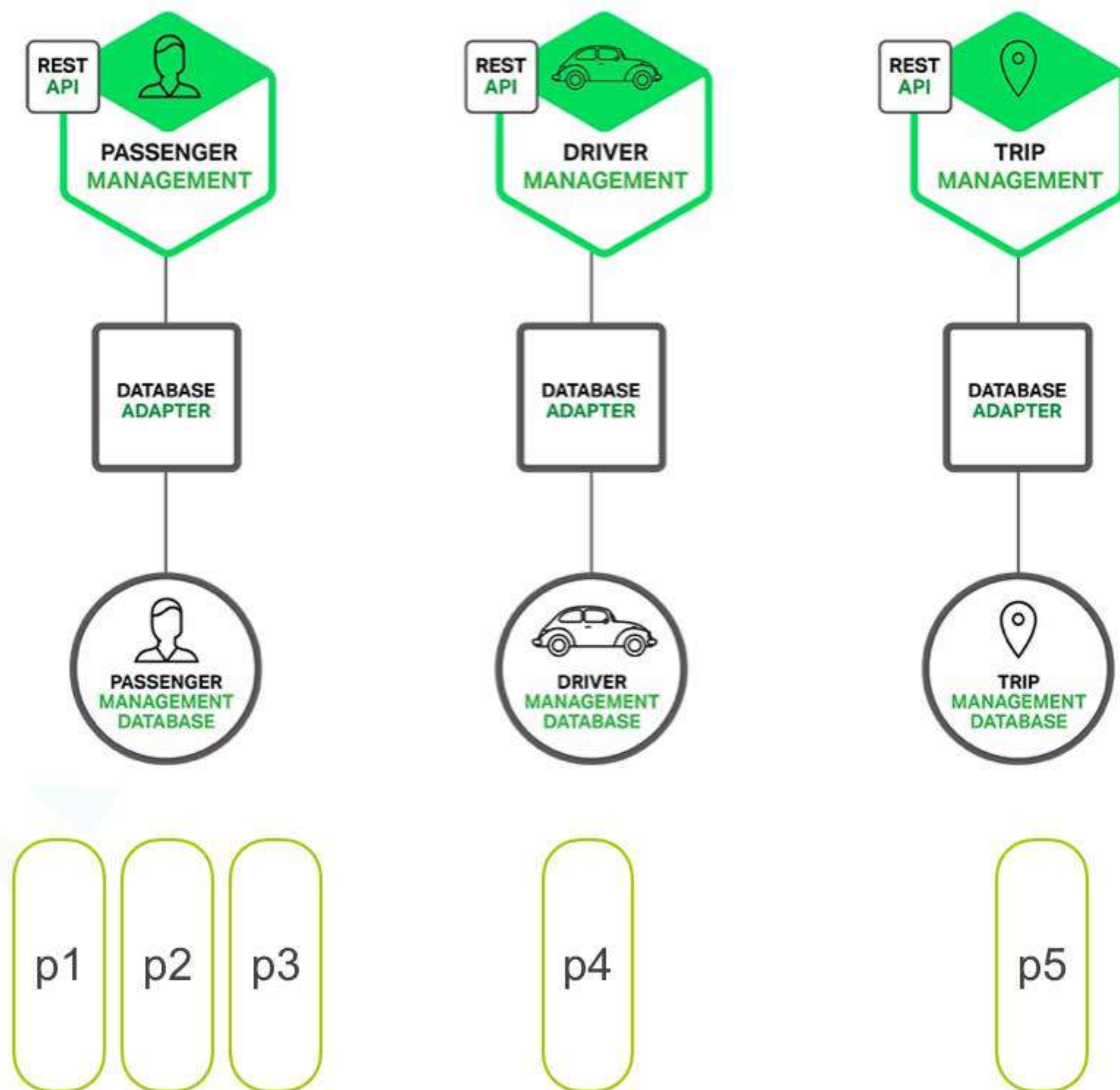
-> olap (dw && bi)

-> data mining -> ai

利益与时间

贯穿开发全过程





页面即服务

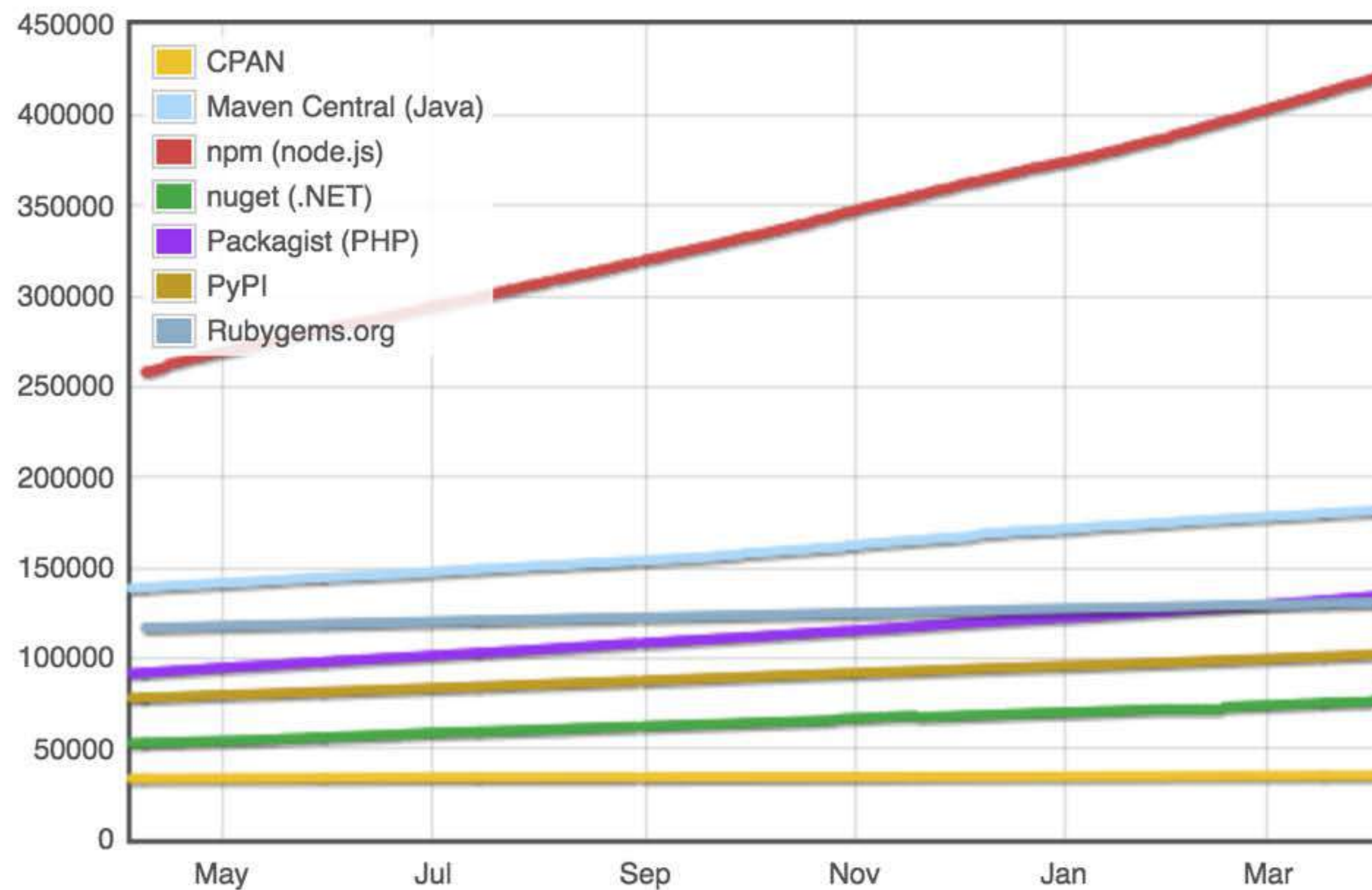
页面不会特别复杂，更容易把控

页面崩溃对整体来说影响绩效

页（必须配合devops）

npm上45w+的模块，叫生态

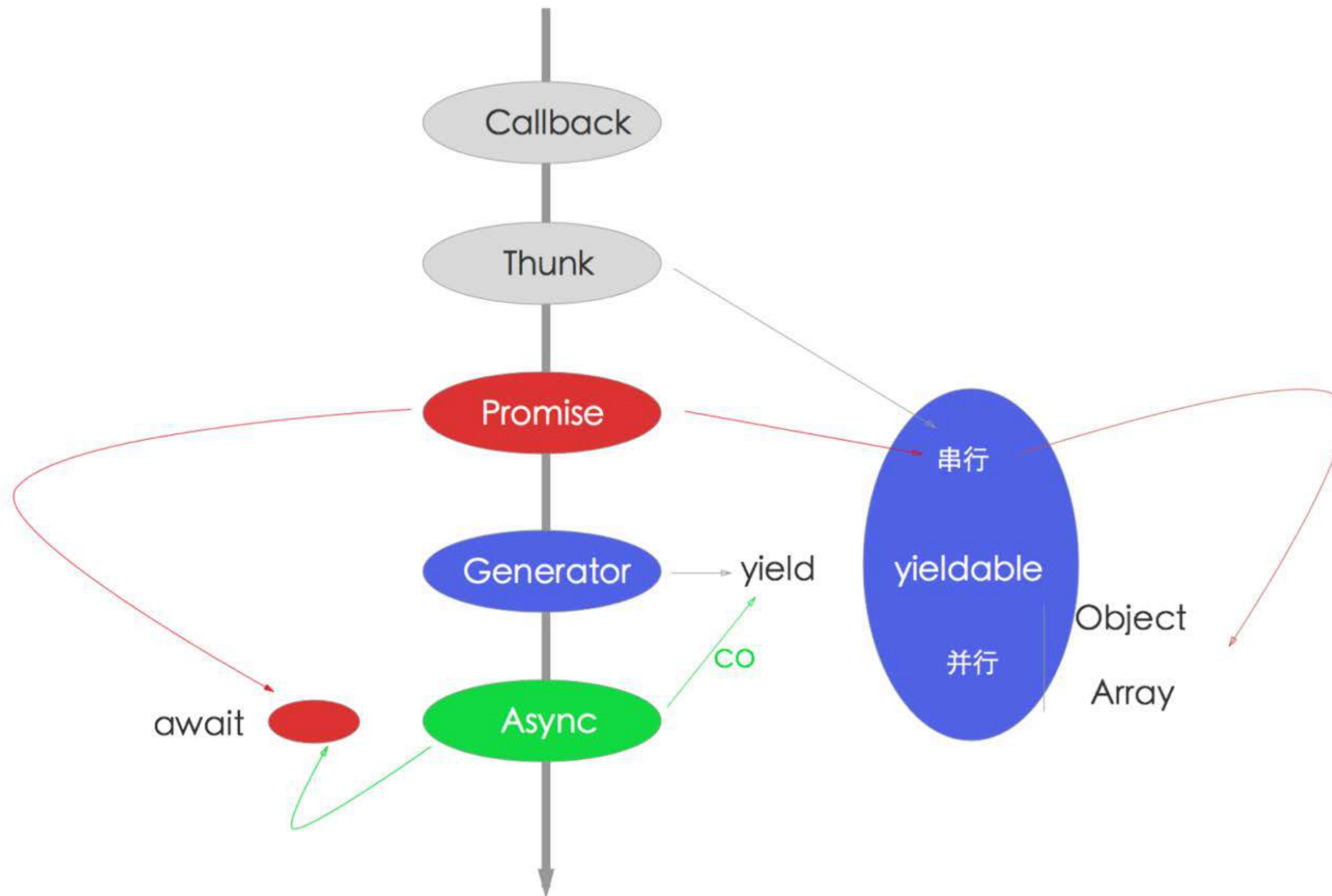
Module Counts



Include

- Clojars (Clojure)
- CPAN
- CPAN (search)
- CRAN (R)
- Crates.io (Rust)
- Drupal (php)
- DUB (dlang)
- Gopm (go)
- Hackage (Haskell)
- Hex.pm (Elixir/Erlang)
- Julia
- LuaRocks (Lua)
- Maven Central (Java)
- MELPA (Emacs)
- npm (node.js)
- nuget (.NET)
- Packagist (PHP)
- Pear (PHP)
- Perl 6 Ecosystem (perl 6)
- PyPI
- Rubygems.org

异步流程控制



学习重点

