F.I.S

提升**产品质量**与**开发效率**的前端解决方案

walterShen

About me

- 沈洪顺
- Senior Web Developer
- 2010年加入百度
- FIS 团队技术负责人
- @walterShen
- @walterShen

高质量的软件研发

- 高可靠性
- 高性能
- 高可维护性
- …

高质量 VS 高效率

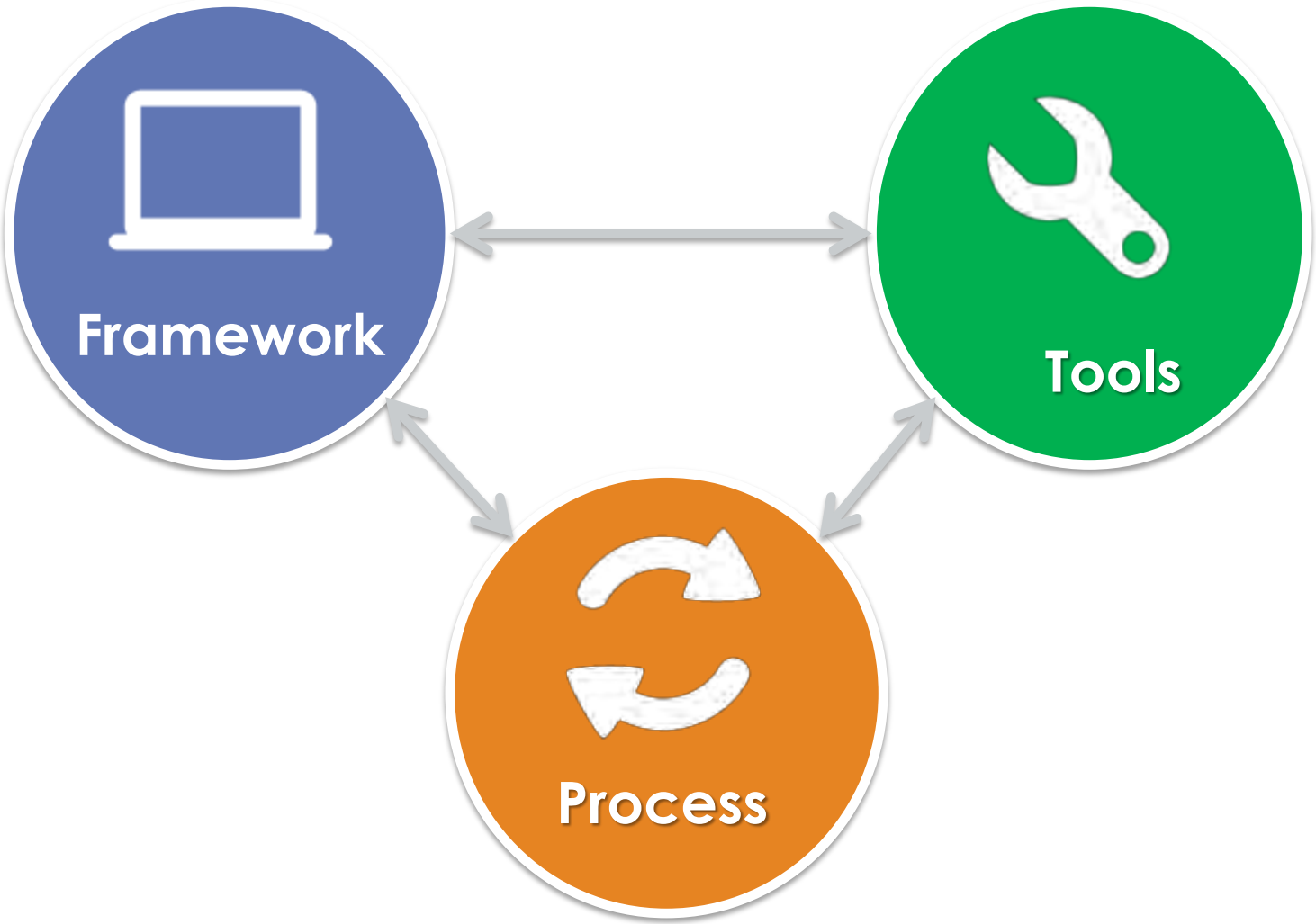

Develop & Test
Release
& Deploy
Monitor & Optimize
Plan &
Measure

前端集成解决方案

Front-end Integrated Solution

- **F.I.S** 是一系列提升产品质量与开发效率的**工程化**方案
  - 让前端团队可以**快速**进入角色，不用担心底层架构、性能优化等问题
  - 可以**减少**人工管理静态资源成本和风险，全自动优化页面性能
  - **简化**开发、提测、部署流程，来达到更快、更可靠、低成本的自动化项目交付

FIS Architecture



FIS Tools

FIS Tools -A **Toolset** For Production

三条命令-满足一切前端需求

```
Usage: fis <command>

Commands:

  release     build and deploy your project
  install     install components and demos
  server      launch a php-cgi server

Options:

  -h, --help     output usage information
  -v, --version  output the version number
  --no-color     disable colored output
```

fis install <name>


```
fis install <name>
```

fis release [options]

- 负责代码的编译与部署，它的参数囊括了前端开发所需的各种需求

  - 添加 --watch 或 -w 参数，支持对项目进行增量编译，监听文件变化再触发编译
  - 添加 --live 或 -L 参数，支持编译后自动刷新浏览器
  - 添加 –dest 或 -d 参数，来指定编译后的代码部署路径
  - 添加 --md5 或 -m 参数，在编译的时候可以对文件自动加md5戳
  - 添加 --lint 或 -l 参数，支持在编译的时候根据项目配置自动代码检查
  - 添加 --test 或 -t 参数，支持在编译的时候对代码进行自动化测试
  - 添加 --pack 或 -p 参数，对产出文件根据项目配置进行打包
  - 添加 --optimize 或 -o 参数，对js、css、html进行压缩
  - 添加 --domains 或 -D 参数，为资源添加domain域名

fis server <command> [options]

- 本地调试服务器，用于本地预览项目
  - fis server start, 启动服务器，默认8080端口，支持自定义rewrite规则
  - fis server open, 打开本地server目录，fis release默认发布到此目录
  - 支持本地数据模拟

静态资源自动管理

资源定位

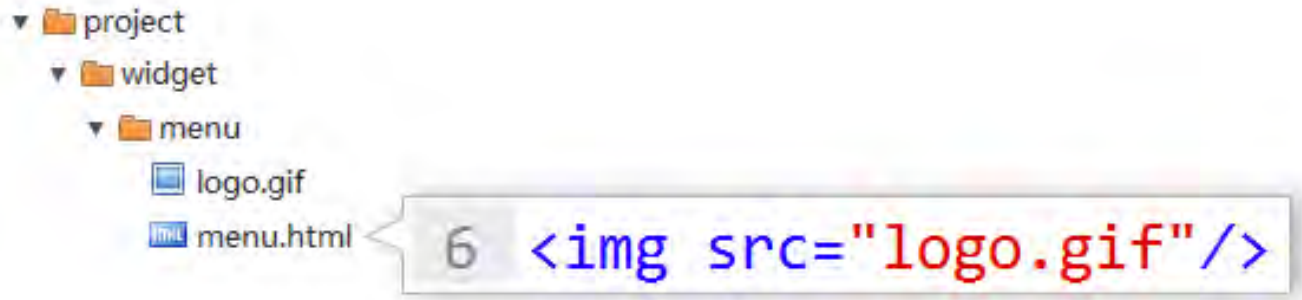
资源内嵌

依赖管理

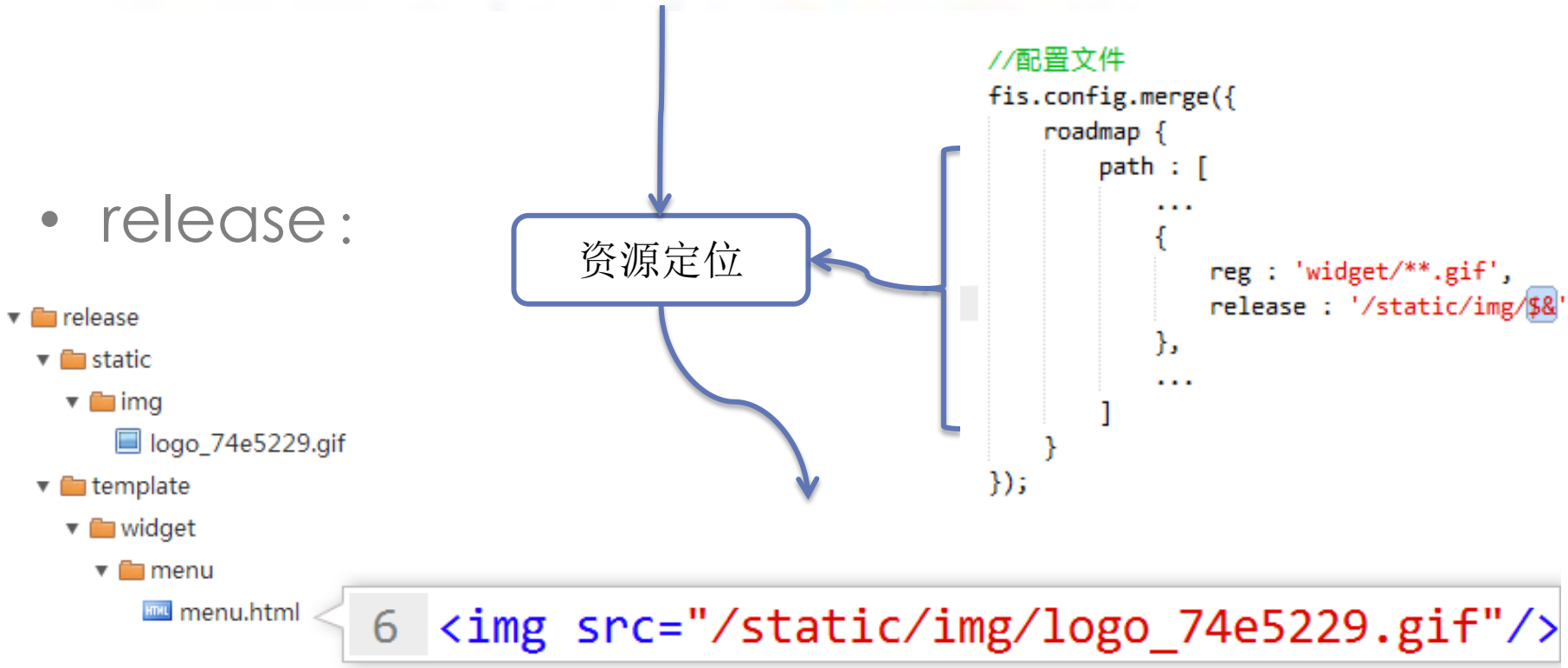
资源合并

资源发布

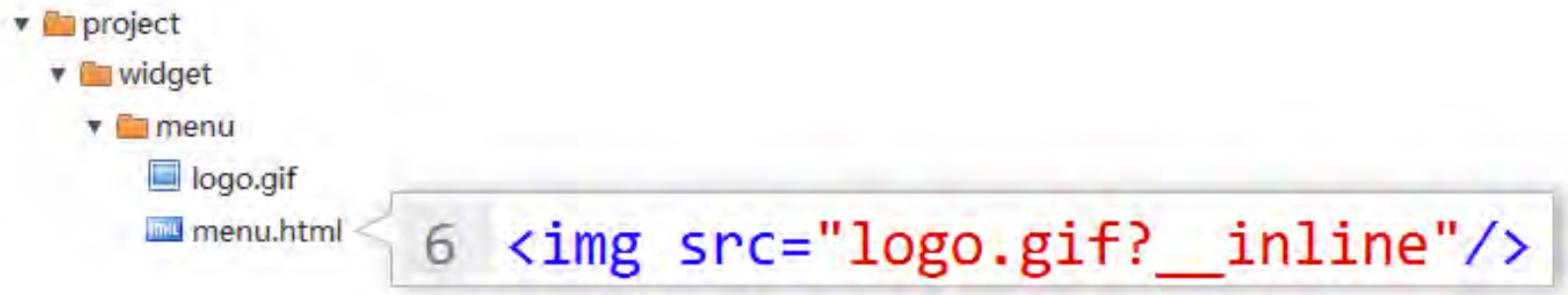
- develop :



- release :



- develop :



- release :

- 在html中声明依赖

```
<!--
    @require demo.js
    @require "demo.css"
-->
```

- 在javascript中声明依赖

```
//demo.js
/**
 * @require demo.css
 * @require list.js
 */
var $ = require('jquery');
```

- 在css中声明依赖

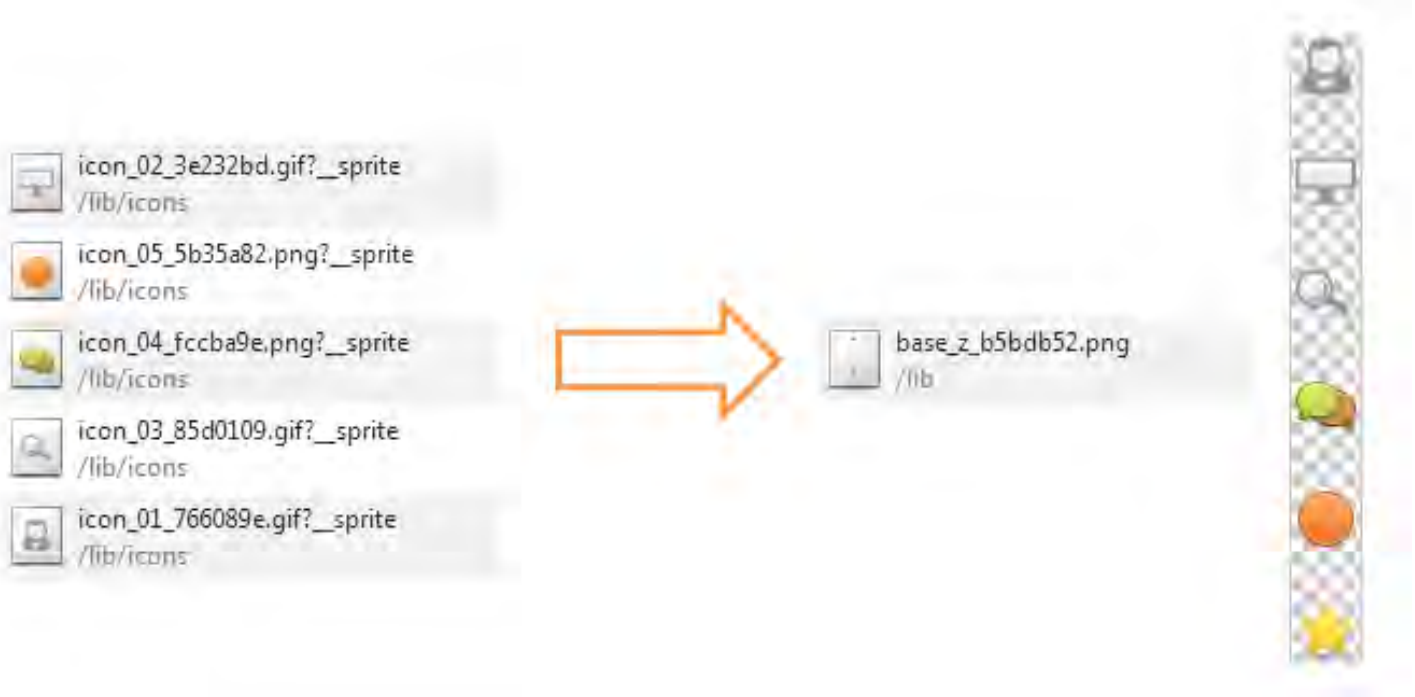
```
/**
 * demo.css
 * @require reset.css
 */
```

✓ map.json

```
{
    "res" : {
        "demo.css" : {
            "uri" : "/static/css/demo_7defa41.css",
            "type" : "css",
            "deps" : [ "reset.css" ]
        },
        "demo.js" : {
            "uri" : "/static/js/demo_33c5143.js",
            "type" : "js",
            "deps" : [ "demo.css" , "list.js" , "jquery" ]
        },
        "index.html" : {
            "uri" : "/index.html",
            "type" : "html",
            "deps" : [ "demo.js", "demo.css" ]
        }
    },
    "pkg" : {}
}
```

```
//file : fis-conf.js
//开启autoCombine可以将零散资源进行自动打包
fis.config.set('settings.postpackager.simple.autoCombine', true);
```

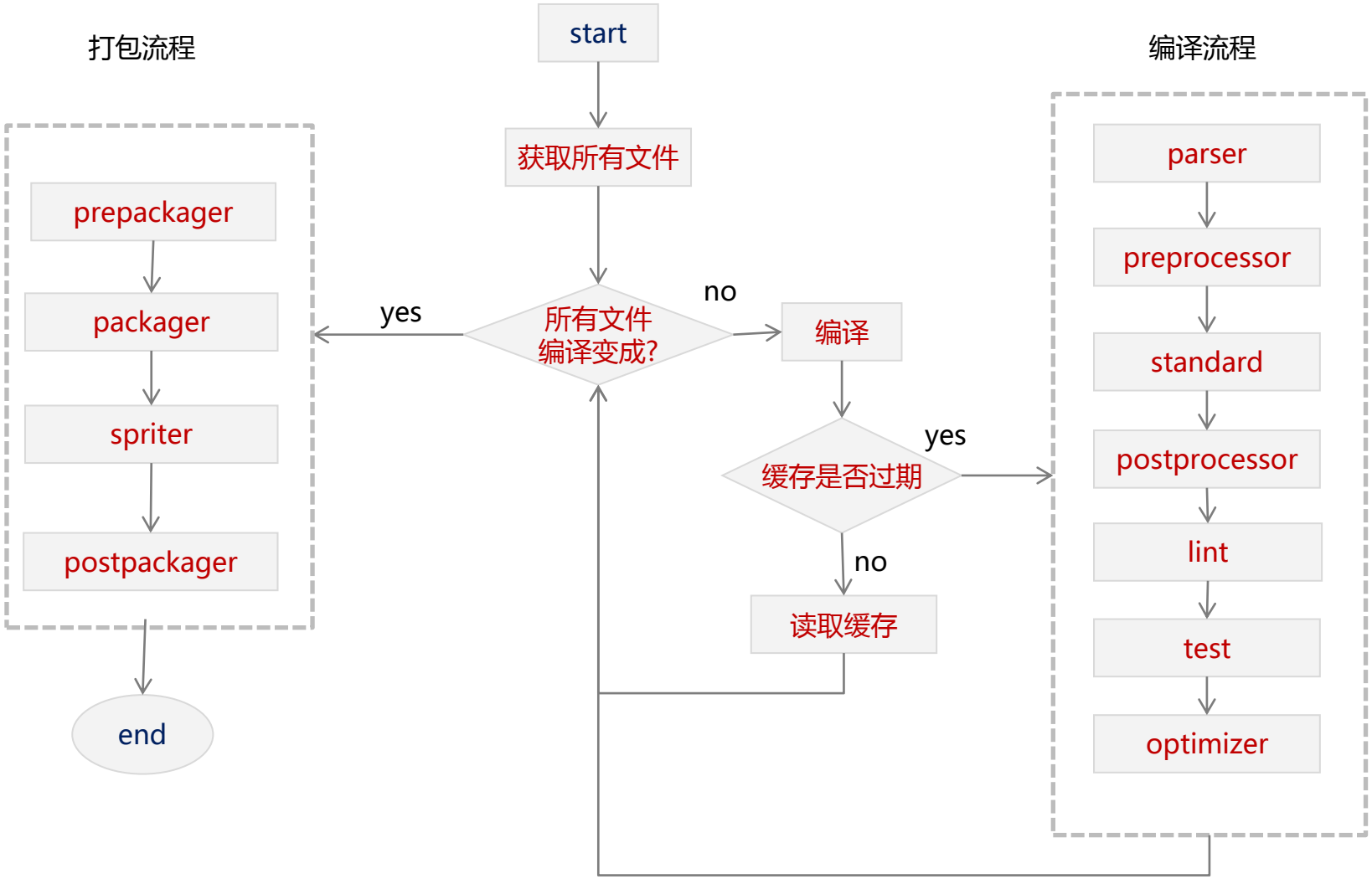
```
//为所有样式资源开启csssprites
fis.config.set('roadmap.path', [{
    reg: '**.css',
    useSprite: true
}]);
//设置csssprites的合并间距
fis.config.set('settings.spriter.csssprites.margin', 20);
```



灵活的发布部署

fis release –d qa,rd,output

- 有效的分离开发路径与部署路径之间的关系
  - 工程师不再关心资源部署到线上之后去了哪里，变成了什么名字，这些都可以通过配置来指定
- 代码具有很强的可移植性
  - 由于开发路径与部署路径对工程师透明，因此组件的资源依赖全部都是相对路径定位的，这样，对于两个同样使用 fis 作为开发平台的团队，即便他们的部署方式完全不同也可以有效移植代码。
- 轻松实现md5、域名添加等功能
  - 工程师完全不用关心上线后资源的静态服务器域名是什么
  - 资源会全部自动添加md5作为版本戳，服务器可以开启强缓存、回滚时不需要回滚静态资源，只须回滚html或模板即可


打包流程
start
获取所有文件
prepackager
packager
spriter
postpackager
end
yes
所有文件
编译变成?
no
编译
缓存是否过期
yes
no
读取缓存
编译流程
parser
preprocessor
standard
postprocessor
lint
test
optimizer

使用 FIS ，轻松定制属于你的解决方案

FIS Plus | Yogurt | Jello | pure | gois

模板框架解决方案，内置静态资源管理和加载框架，提高服务端的渲染效率和并行度，使得首屏及核心功能最快展现，适用于网络高延迟/低带宽、国际化等多种业务场景.

查看详细

FIS Framework

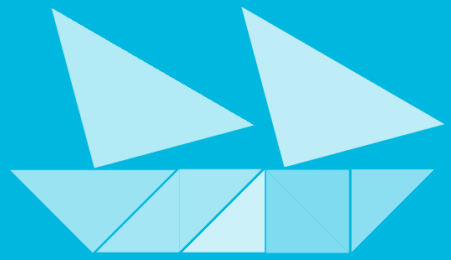
Plus

基于 **PHP/Smarty** 实现的 fis **展现层**解决方案

Pure

纯前端 fis **展现层**解决方案，不依赖于后端

Yogurt



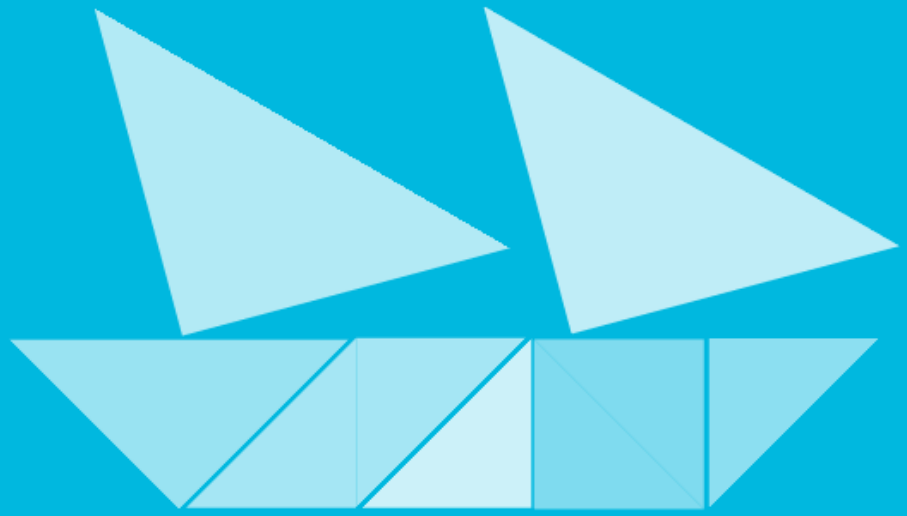
基于 **NodeJS/Express.js** 实现的 fis **前后端一体化**解决方案

Jello

基于**Java/Velocity** 实现的 fis **展现层**解决方案

Gois

基于 **Go/Martini** 实现的 fis **展现层**解决方案



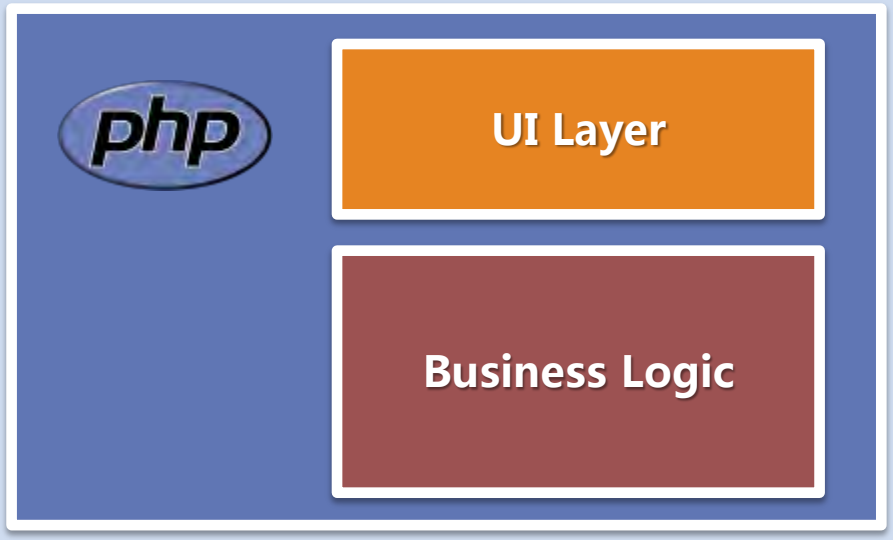
NodeJS 前后端一体化框架(Yogurt)

Front-End

Browser

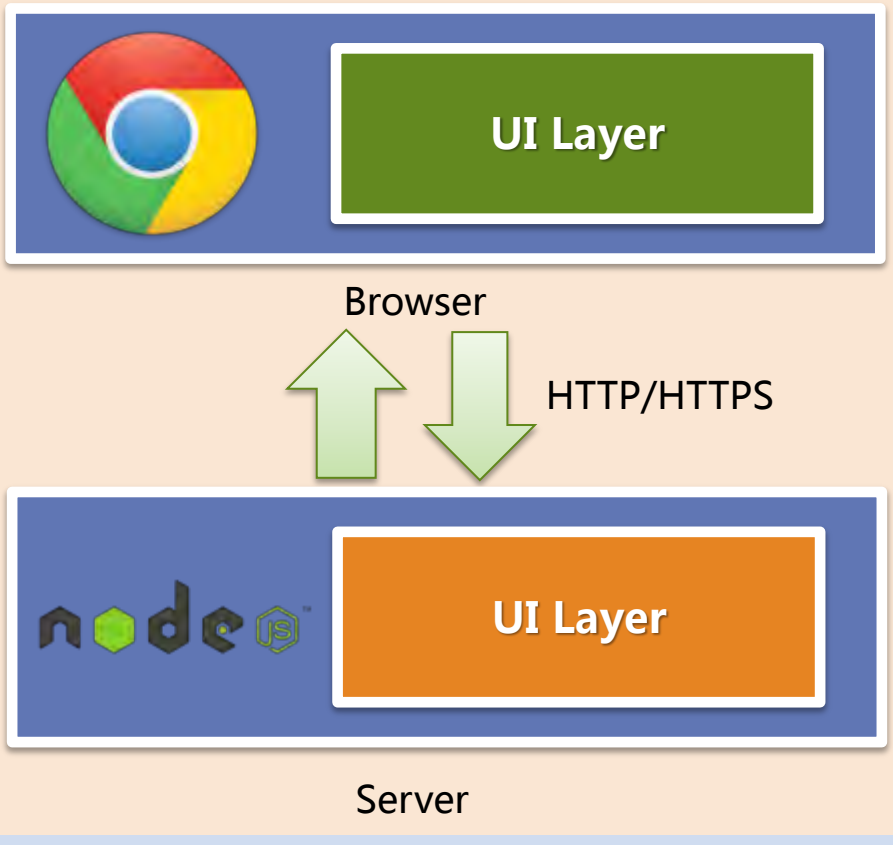
HTTP/HTTPS

Back-End

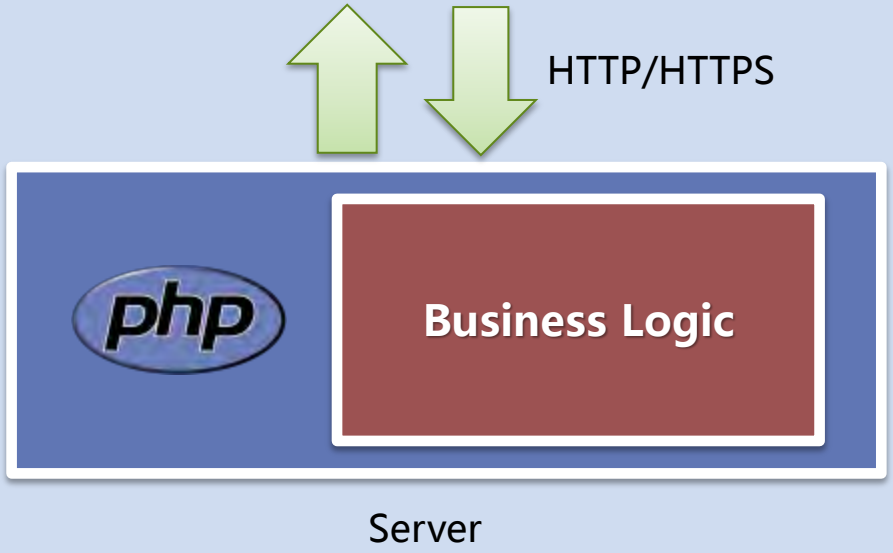


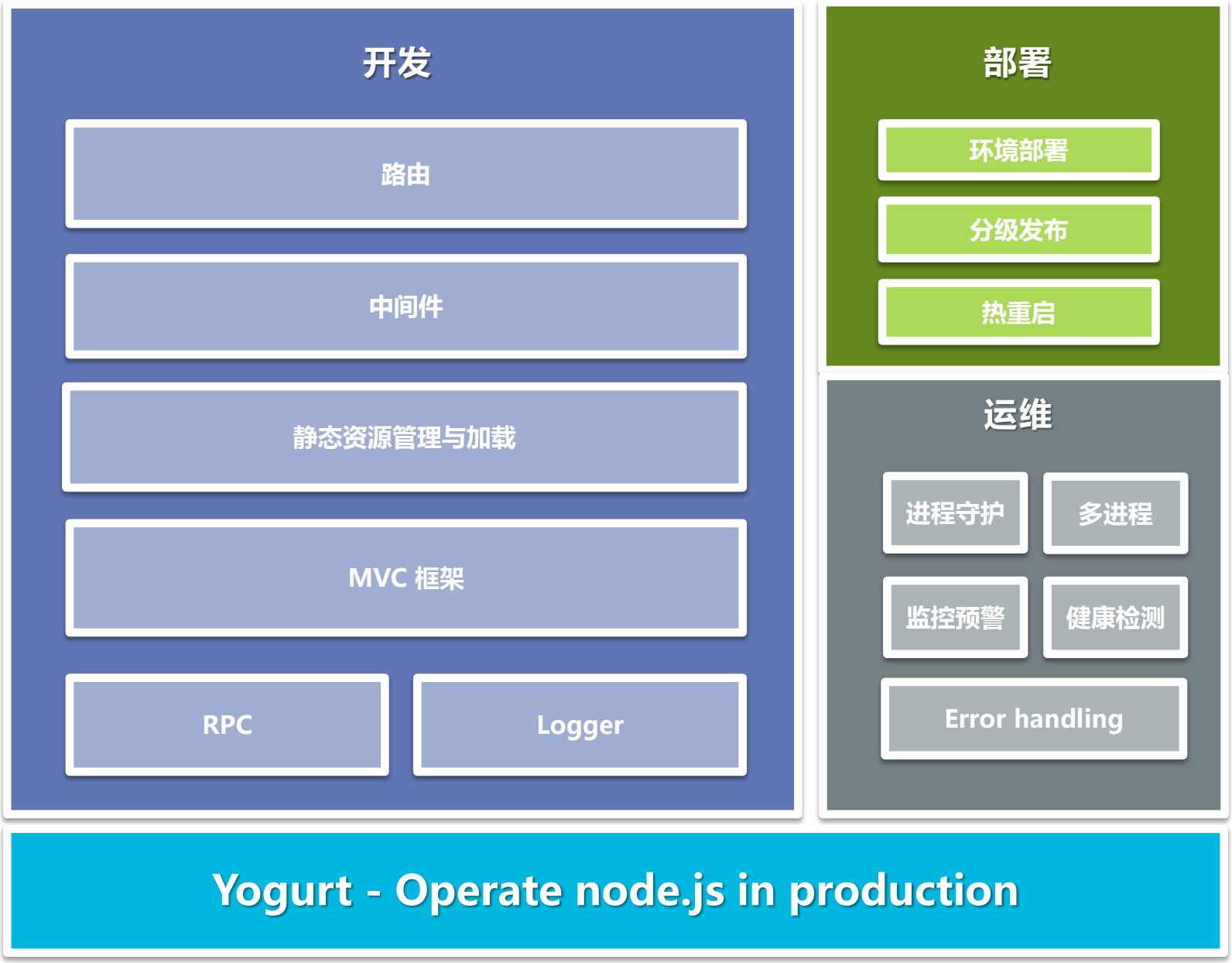
Server

Front-End

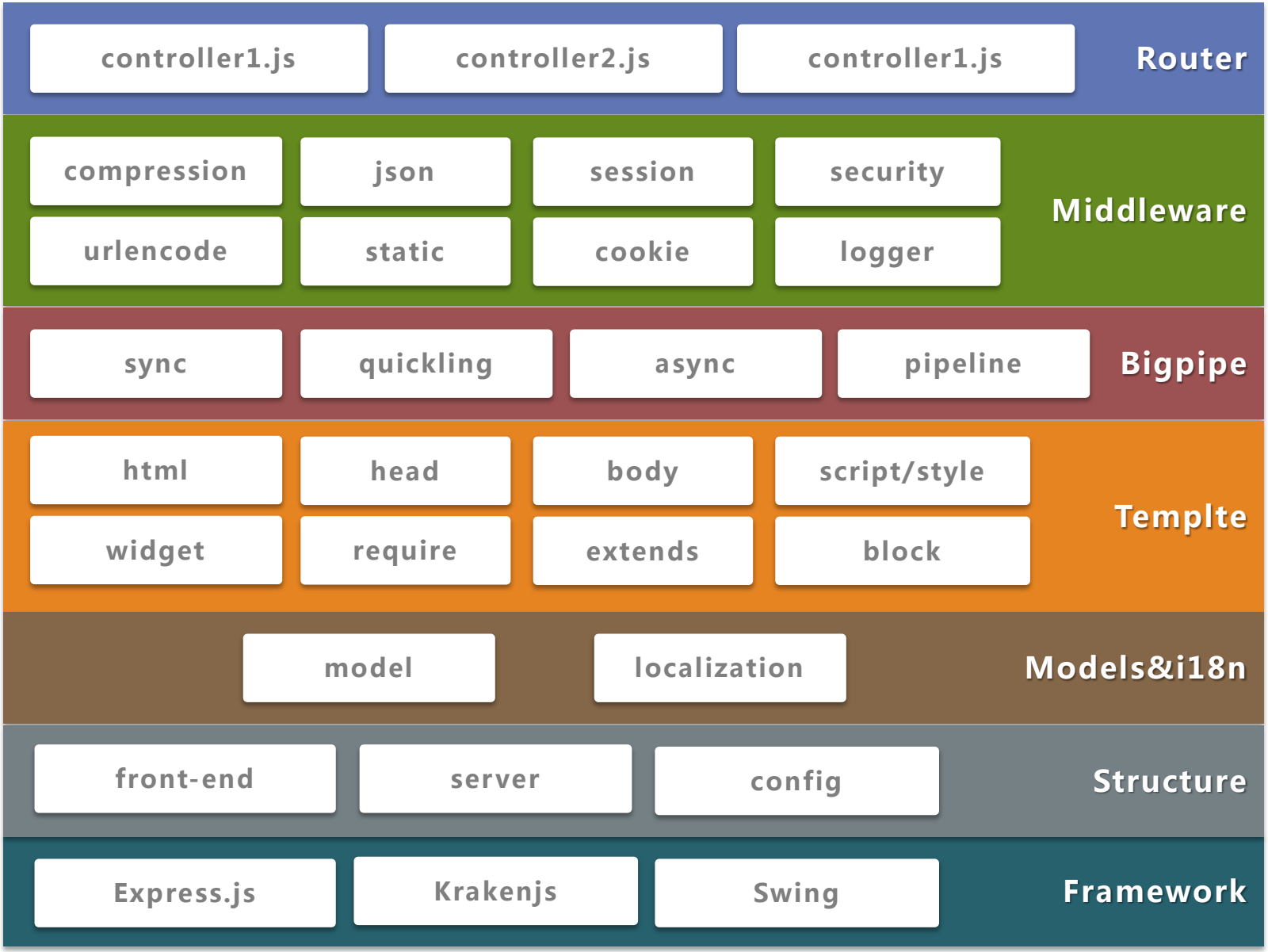


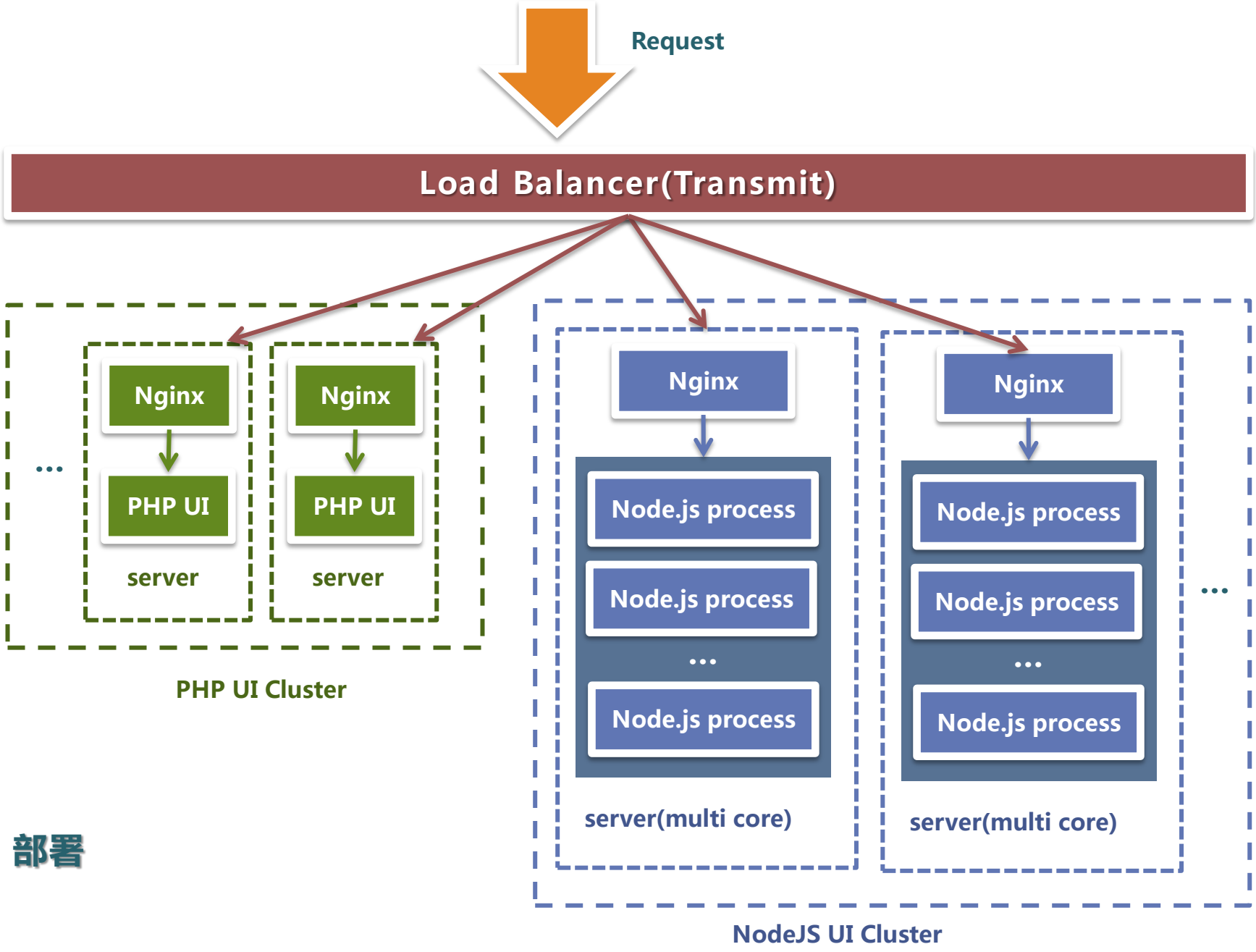
Back-End



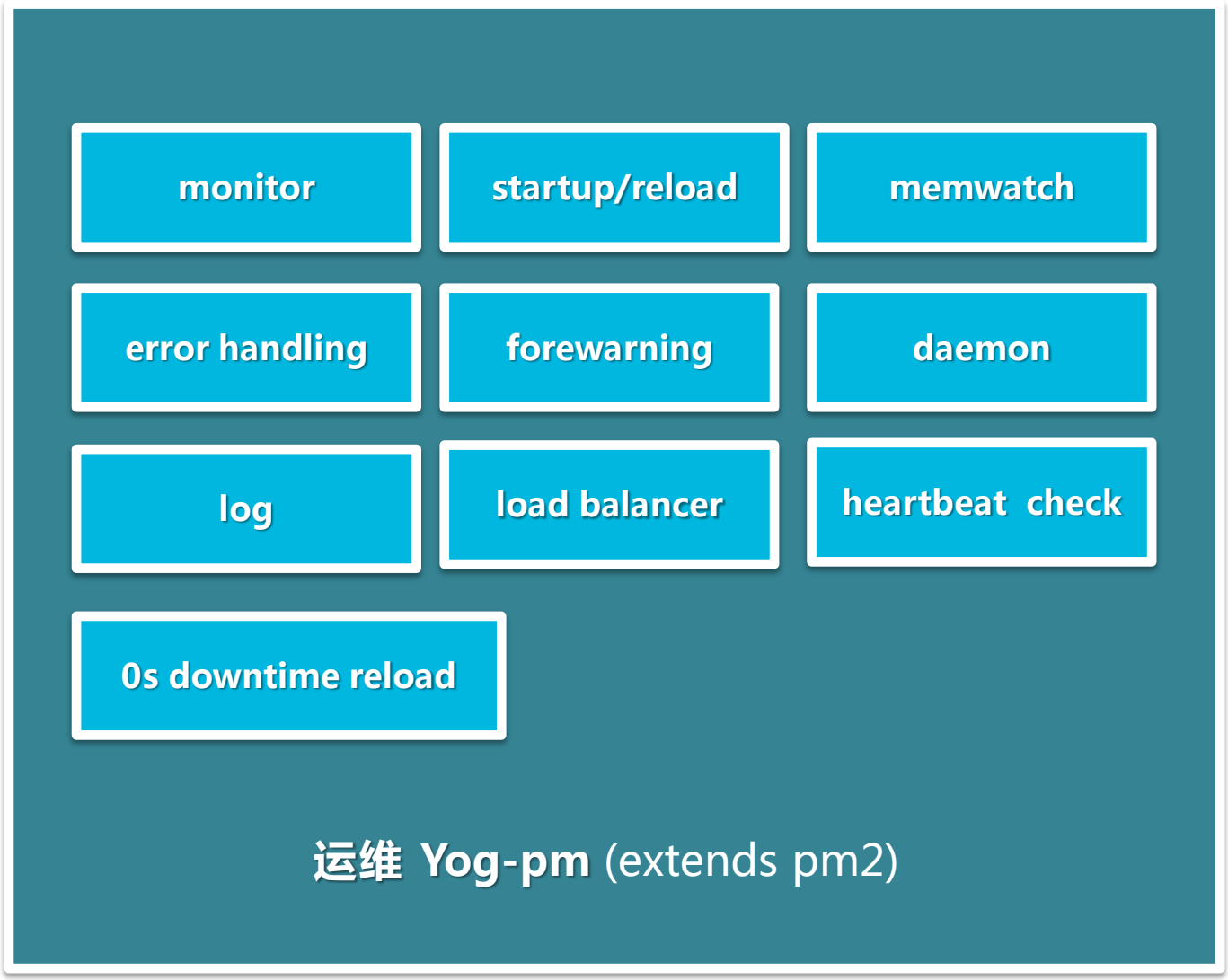

开发
路由
中间件
静态资源管理与加载
MVC 框架
RPC
Logger
部署
环境部署
分级发布
热重启
运维
进程守护
多进程
监控预警
健康检测
Error handling
Yogurt - Operate node.js in production

开发


controller1.js
controller2.js
controller1.js
Router
compression
json
session
security
urlencode
static
cookie
logger
Middleware
sync
quickling
async
pipeline
Bigpipe
html
head
body
script/style
widget
require
extends
block
Templte
model
localization
Models&i18n
front-end
server
config
Structure
Express.js
Krakenjs
Swing
Framework


Request
Load Balancer(Transmit)
Nginx
PHP UI
server
Nginx
PHP UI
server
PHP UI Cluster
Nginx
Node.js process
Node.js process
Node.js process
server(multi core)
Nginx
Node.js process
Node.js process
Node.js process
server(multi core)
NodeJS UI Cluster

部署


monitor
startup/reload
memwatch
error handling
forewarning
daemon
log
load balancer
heartbeat check
0s downtime reload
运维 Yog-pm (extends pm2)


Solar
FIS 云服务平台
静态资源自动合并系统
根据线上资源使用情况，从性能优化角度出发自动合并资源，解决人力成本，提升产品性能
查看详细
Feature Flag系统
无需分支上线，轻松控制线上新功能发布时机、面向人群、流量等。
查看详细
Lights前端资源聚合平台
便捷、易用的资源安装、发布、搜索，管理工具。快速共享团队资源，提升开发效率。
查看详细
FIS编译机插件管理平台
轻松管理多台线上编译机的FIS插件的同步、安装、更新等，让编译机不再黑盒，一目了然。
查看详细

FIS Process

FIS静态资源自动合并服务(Auto-Pack)

```
<html>
    <link href="A.css">
    <link href="B.css">
    <link href="C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    {if $user_has_C}
        <div>html of C</div>
    {/if}
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    {*if $user_has_C}
        <div>html of C</div>
    {/if*}
</html>
```

```
<html>
    <link href="A-B-C-D-E-F-G-H....css">
    <div>html of A</div>
    <div>html of B</div>
    ...
    {if $not_used_F}
        <div>html of E</div>
    {else}
        <div>html of F</div>
        <div>html of G</div>
    {/if}
    ...
</html>
```

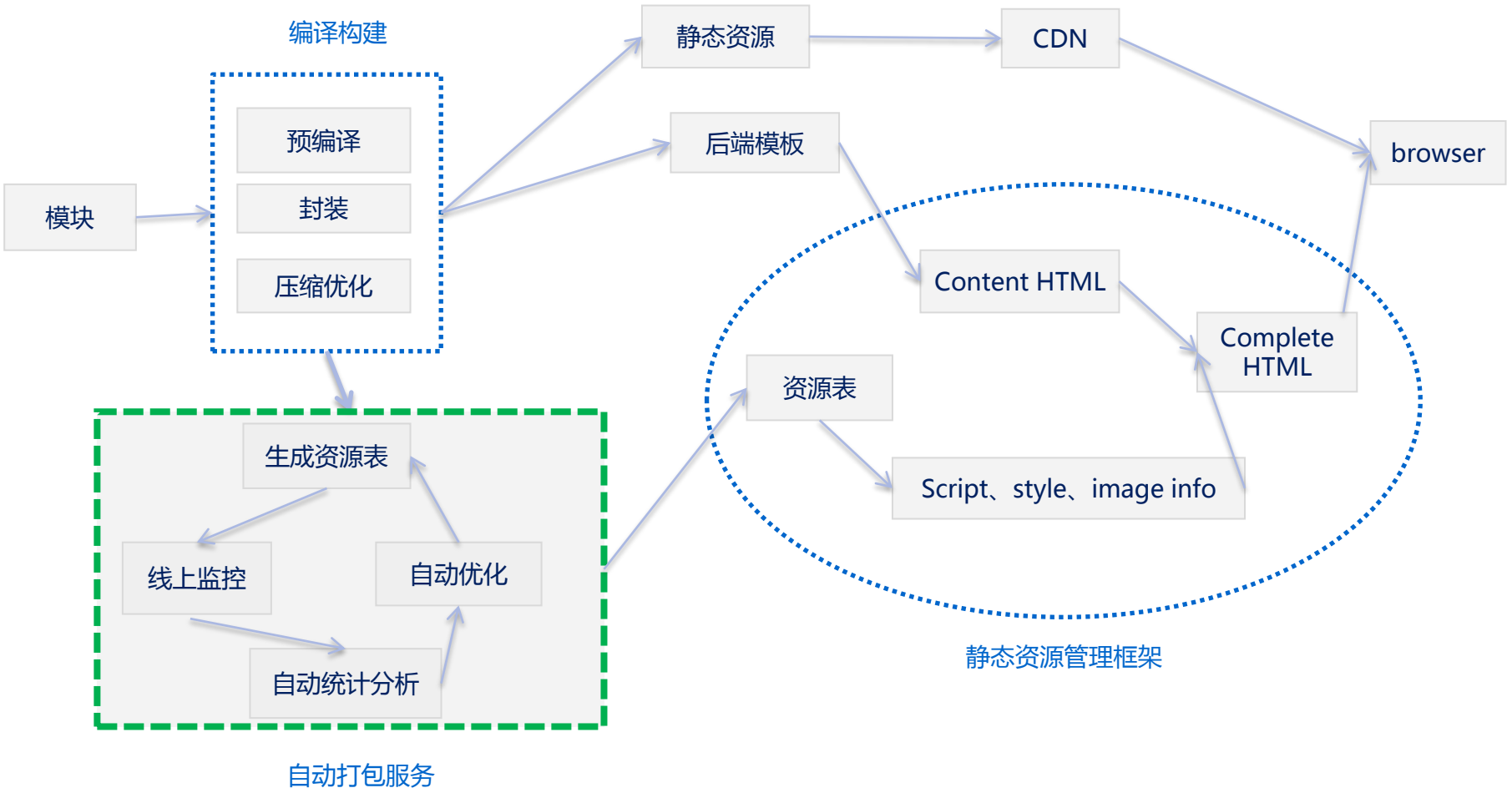
1. 性能优化投入成本高且不可持续性
2. 人工优化很难达到全局最优


logic

Auto-Pack

- 静态资源自适应优化合并服务
  - 根据网站页面pv以及页面静态资源使用情况，自动计算静态资源合并方案，减少人工管理静态资源成本和风险
  - 从网络请求、首屏渲染等方面优化网站性能、减少服务器开销

- 完全兼容FIS编译流程，极小学习成本
- 自动计算静态资源最佳合并方式
- 自动适应静态资源的添加与删除
- 适配不同网络、不同合并方式的情况


编译构建
静态资源
CDN
browser
预编译
后端模板
模块
封装
压缩优化
Content HTML
Complete HTML
资源表
生成资源表
Script、style、image info
线上监控
自动优化
自动统计分析
静态资源管理框架
自动打包服务

资源合并算法

- 合并收益 ：对于同时使用A和B的页面节省了网络来回时间(RTT)
- 合并损失 ：对于只使用A的页面，浪费的B的大小(损失的大小 / 下载速度)
- 纯收益 ： 合并收益 – 合并损失

	Page_1	Page_2	Page_3	Page_4	Page_5
访问量	10M	1M	200K	10K	1K
A.Js (1KB)	√	√	√	√	√
B.Js (1KB)	√	√	√	√	√
C.Js (300B)	√	√			
D.Js (2KB)					√
E.Js (700B)		√	√		
F.Js (600B)		√	√	√	√

区分首屏和延迟加载

区分网络

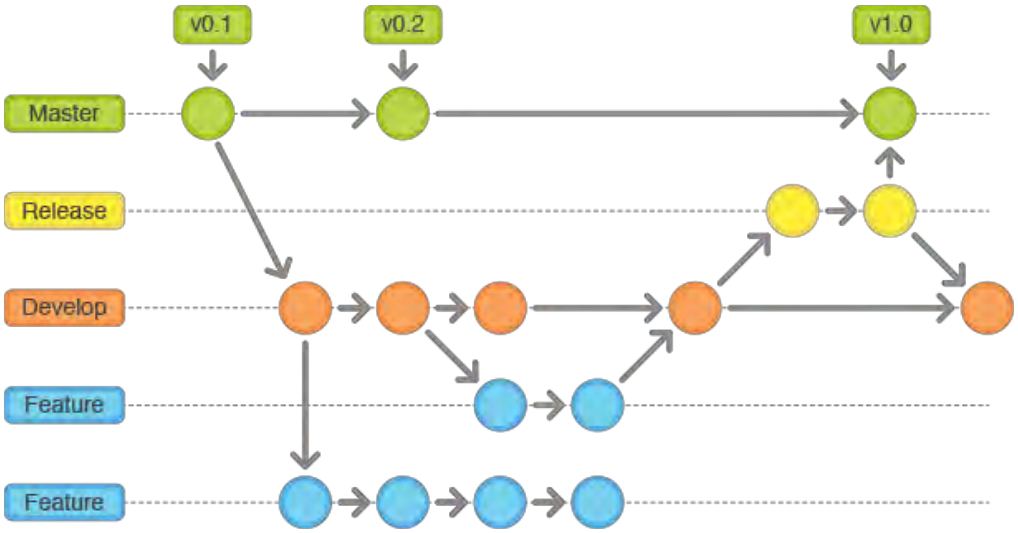
区分国家

...

功能发布控制系统(Feature-Flag)

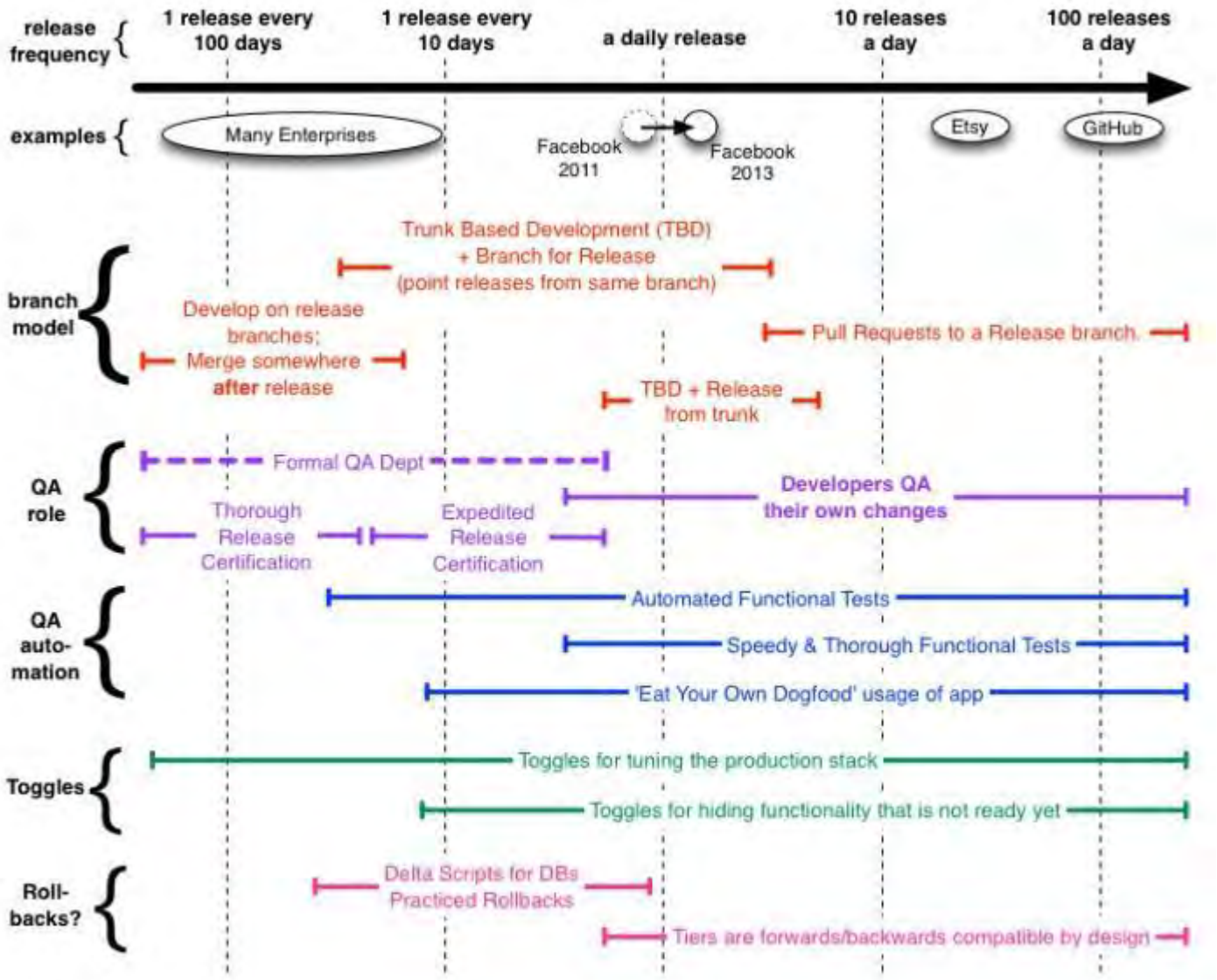
精准发布，控制自如

Feature Branches

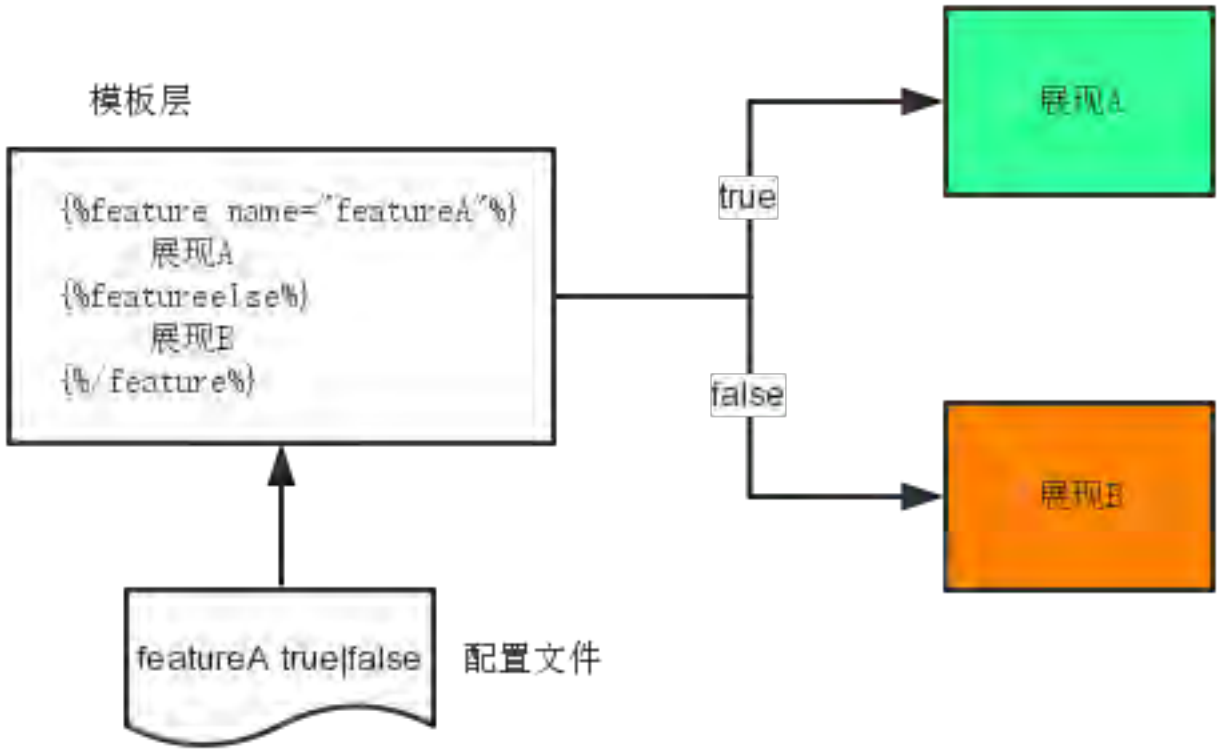


- 分支分出去时间越长往往代码merge难度越大、风险和成本越高
- 在一个分支中修改了函数名字可能会引入大量编译错误，重构成本很高
- 一旦代码库中存在了分支，无法很好的支持持续集成，迭代速度受影响
- 有多个feature branches的时候，无法测试这多个feature之间的影响

Facebook's Trunk Based Development



Feature-Flag



根据各种**场景**和**条件**配置控制是否展现页面某一区域或功能，不用重新发布代码

Feature-Flag

- Feature-Flag框架
  - 快速回滚
  - 小流量
  - A/B测试
  - 特定时间发布
  - 特定区域发布
  - 主干开发
- Flag管理平台，可视化管理产品中的所有 feature flag
- 小流量评估平台,结合 feature flag 自动分析、评估小流量的效果和收益

Feature-Flag 使用注意

- 如果某个功能最终不上线，后续需要手工删除相关代码
- 会出现因为配置错误某个 feature 没有完成就出现在线上
- 需要控制 flag 数量，有可能会被滥用

开源

- 百度FIS团队
- 用户：百度、阿里、腾讯、UC、去哪...
- QQ交流群：315973236
- web site： http://fis.baidu.com/
- github： https://github.com/fex-team/fis
- 招聘：http://fex.baidu.com/we-need-you/

<Thank You!>

@walterShen