数亿级用户规模下的 React native工程实践

berg

ABOUT ME

- 雷志兴 - berg
- 手机百度架构师
- 微信公众号：DevLife

- 前端、Native端融合
- 前端基础技术、工程化

OUTLINE

- 为什么选择React native
- 与现有业务和迭代融合
- 性能优化实践
- React Native的工程价值

手机百度

- 迭代周期：4周
- 主要Feature：数十个
- 开发工时：数千人时

- 跨部门，沟通成本高
- 业务种类多，集成成本高

手机百度

- 迭代周期：4周
- 主要Feature：数十个
- 开发工时：数千人时

- 跨部门，沟通成本高
- 业务种类多，集成成本高

过去的解耦方案

迭代速度慢

开发、集成成本高

准入流程长

APK插件/SDK

Hybrid

体验、效果差

本地能力依赖发版

React Native@手机百度

React Native@手机百度

- RN版本：**v0.35**
- RN用户数：**2亿+**
- RN启动次数：**15亿+**
- 总崩溃率：**0.07%**
- 24h收敛率：**98%**

5-8倍的迭代速度

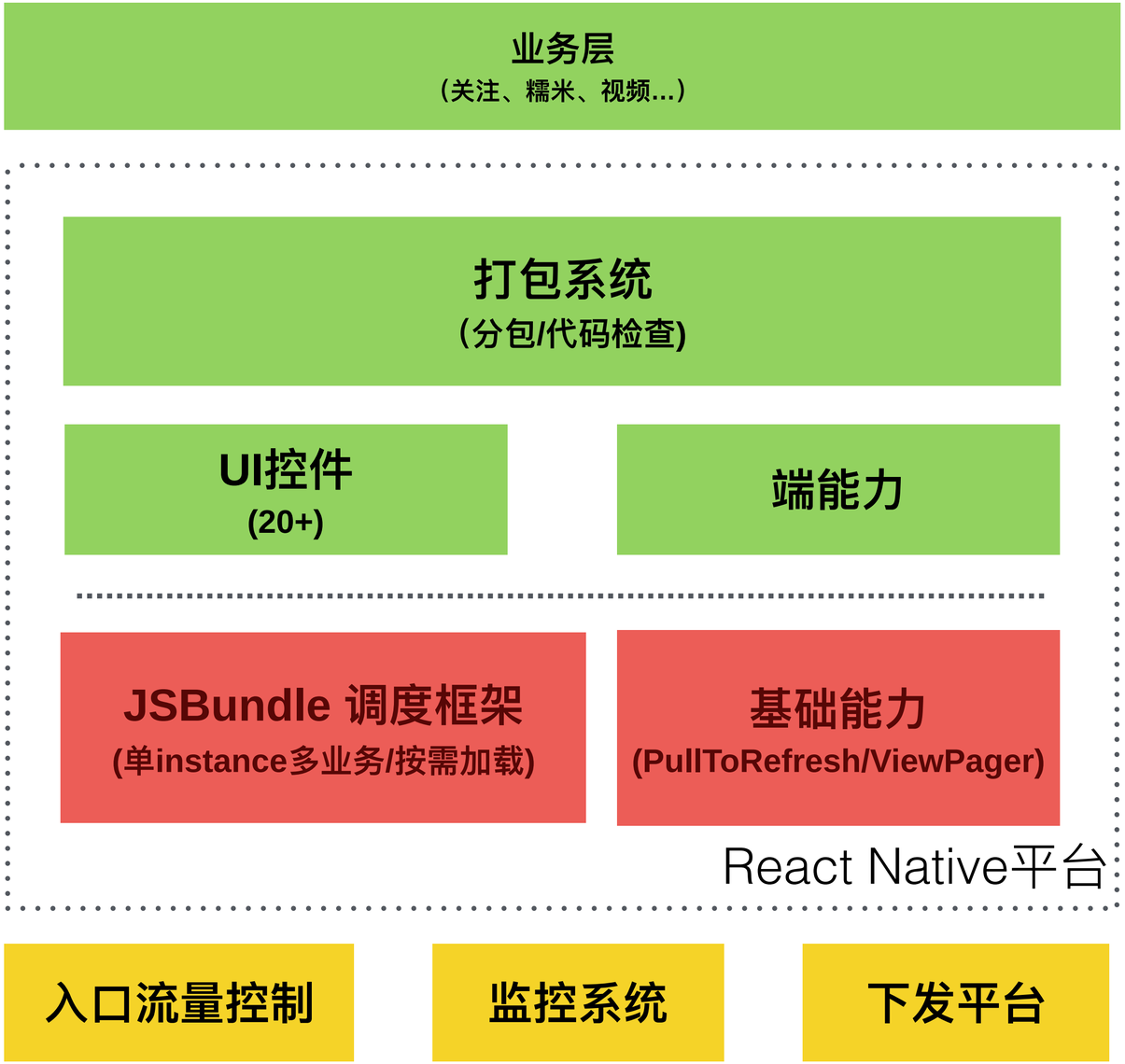
React Native 的工程实践

- 与现有业务融合
  选型、人力投入、回退方案…
- 迭代流程、质量控制
  准入、灰度、发布、回滚、监控…
- 性能优化
  加载时间、Listview、动画…

需要系统性的考虑

与现有业务融合

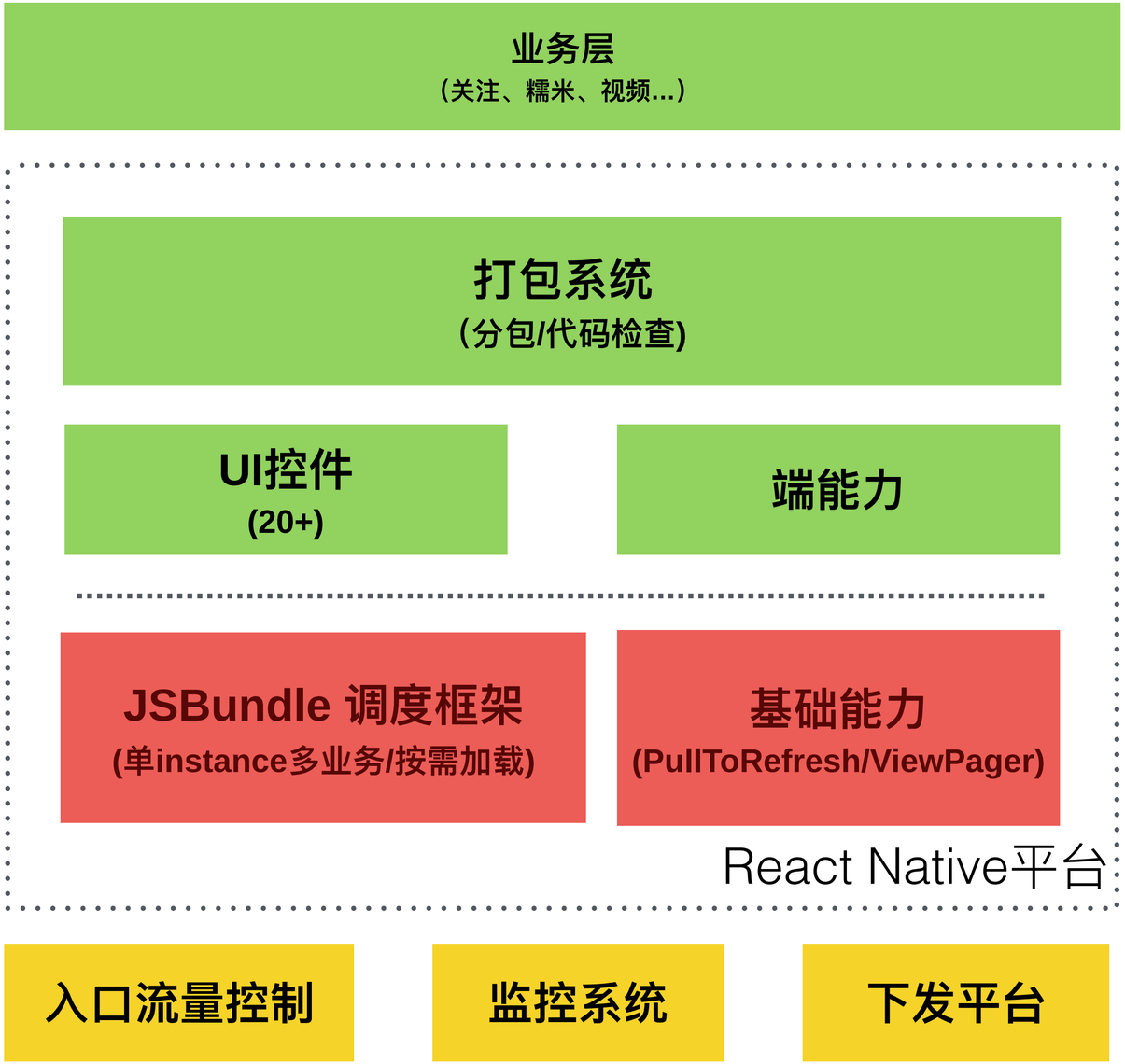
整体架构 – 兼容性



- Android：
  - >=4.1 (99%)
  - V29 >=4.4 (88%)
    listview滑动内存释放缓慢，或与硬件加速有关
- iOS： >=8.0（95%）

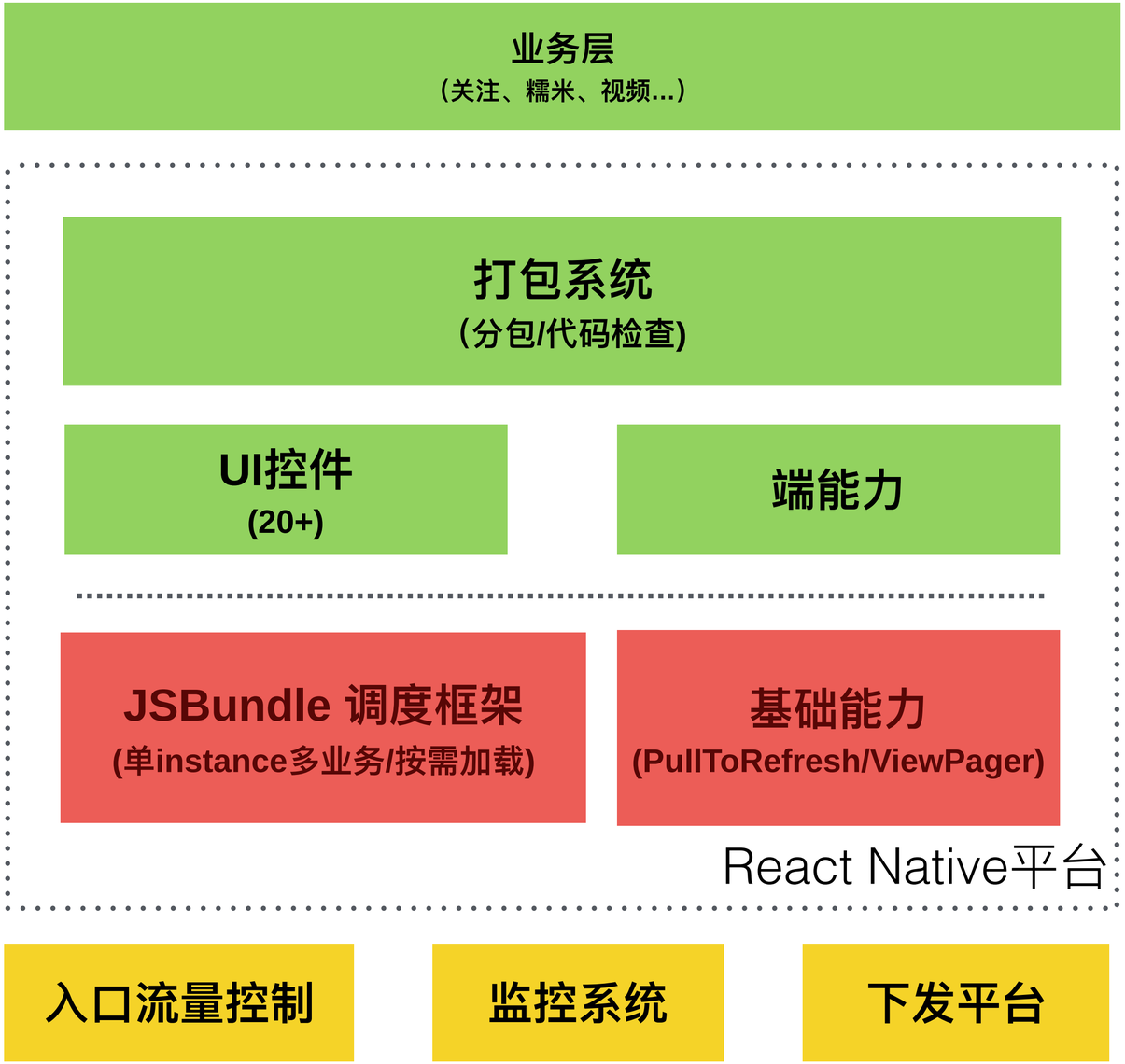
- 与宿主工程的兼容性
  soLoader、Fresco、OKHttp

整体架构 – 兼容性



- 降级和兼容方案
  - 随时回撤
  - AB Test

整体架构 – 业务层



- HTML -> JSX
- CSS -> CSS-layout
  无float/px/css继承/css sprites
- ES 5 -> ES 6
- DOM -> React native View
- 推荐Redux
  统一管理state，减少事件滥用

业务层

- HTML -> JSX
- CSS -> CSS-layout
  无float/px/css继承/css sprites
- ES 5 -> ES 6
- DOM -> React native View

- 性能
  - **空间换时间、便利性换性能**
  - **减少重绘、减少通讯**

统一交互协议

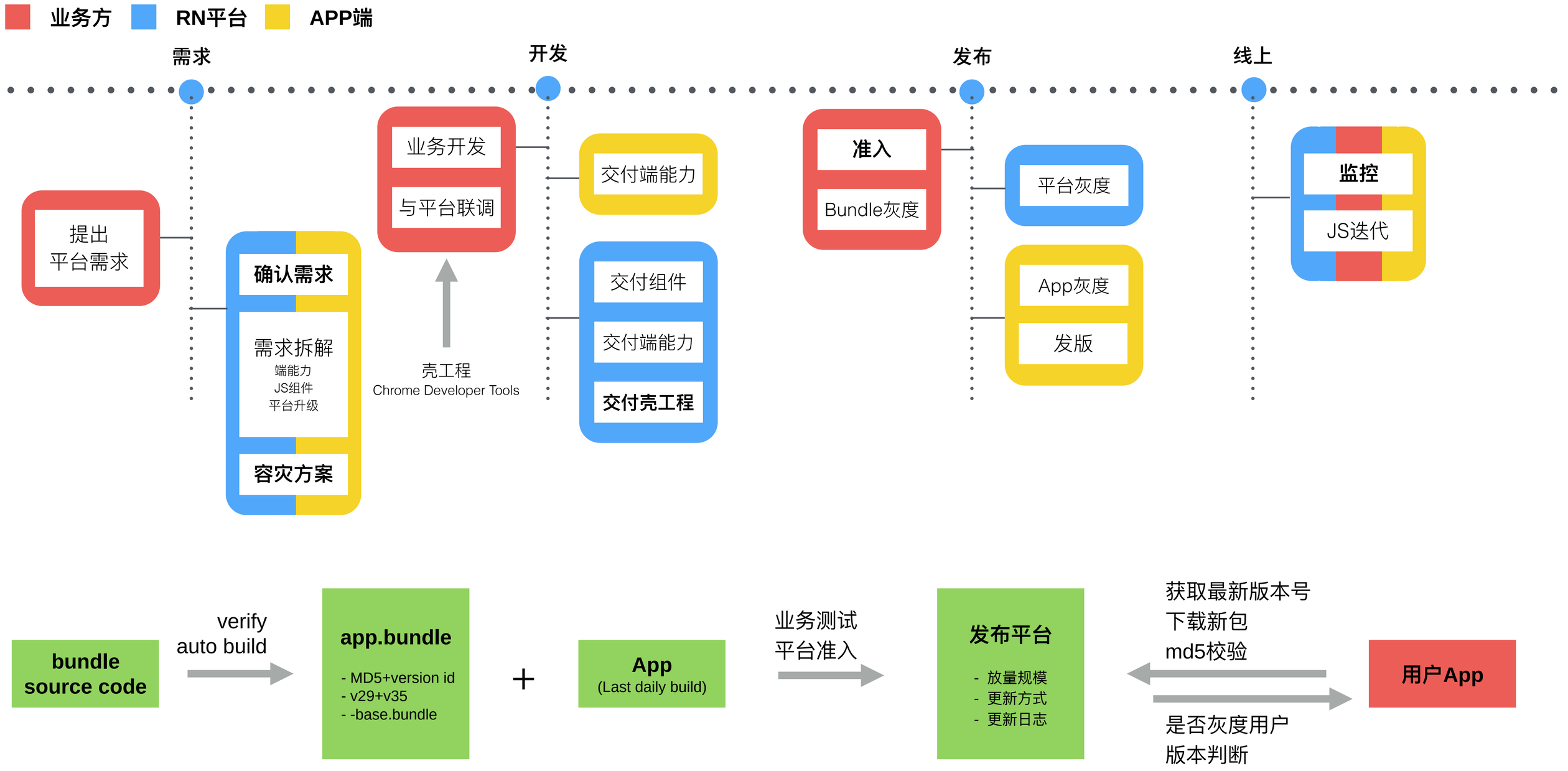

```
import {invoke} from 'baidu-invoke';
invoke('ClassName.methodName', {
    key: value
})
  .then(data => console.log(data.result))
  .catch(error => console.log(error.code, error.message));
```


```
import {eventListener} from 'baidu-invoke';
let event = eventListener.addListener($type, evt => {});
event.remove();
eventListener.removeAllListeners($type);
```


```
{
    "rn_bundle_id": "rnplugin.myattention",
    "rn_component_name": "MyAttention"
}
```

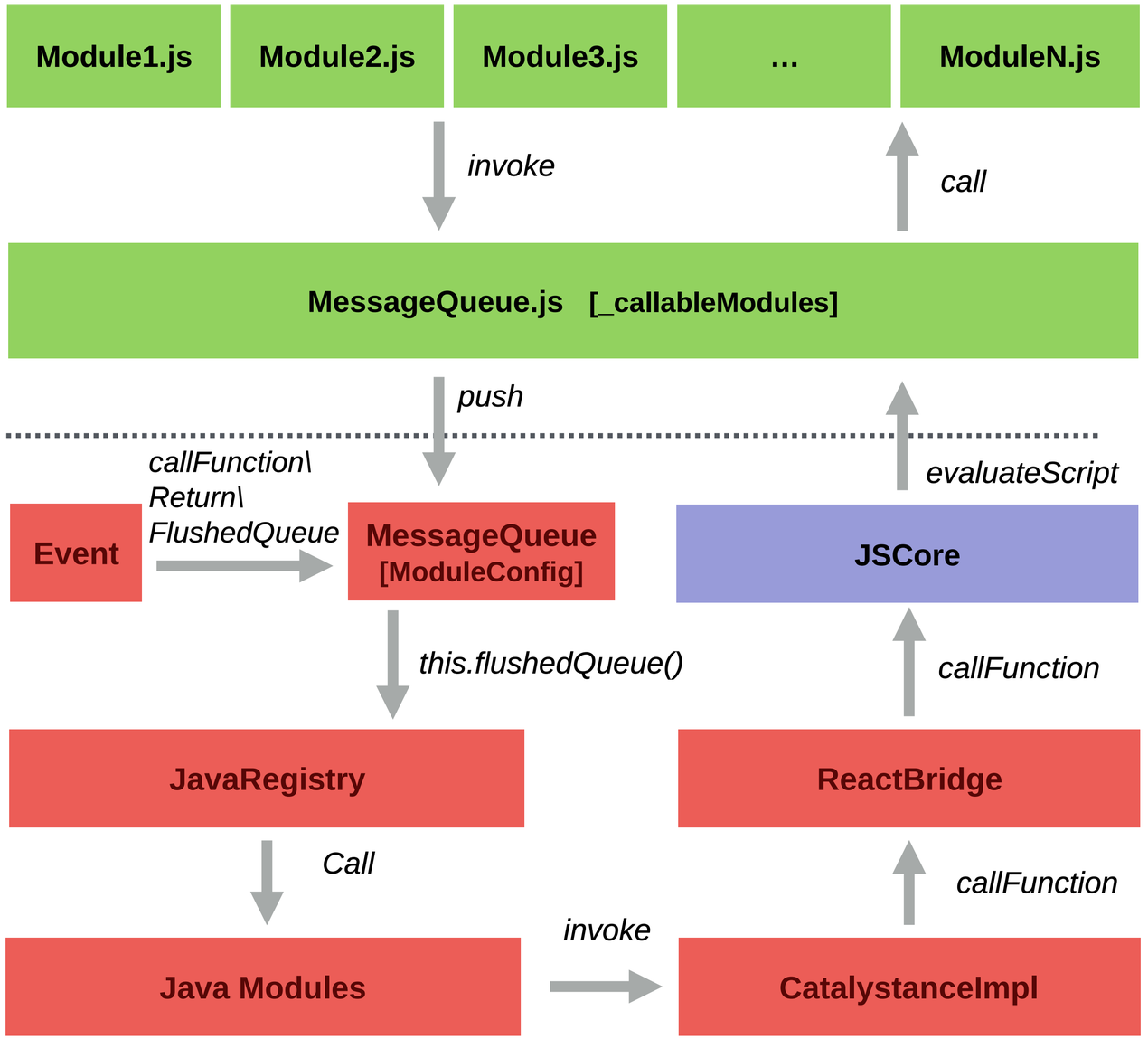
通用增量接口

融入迭代流程

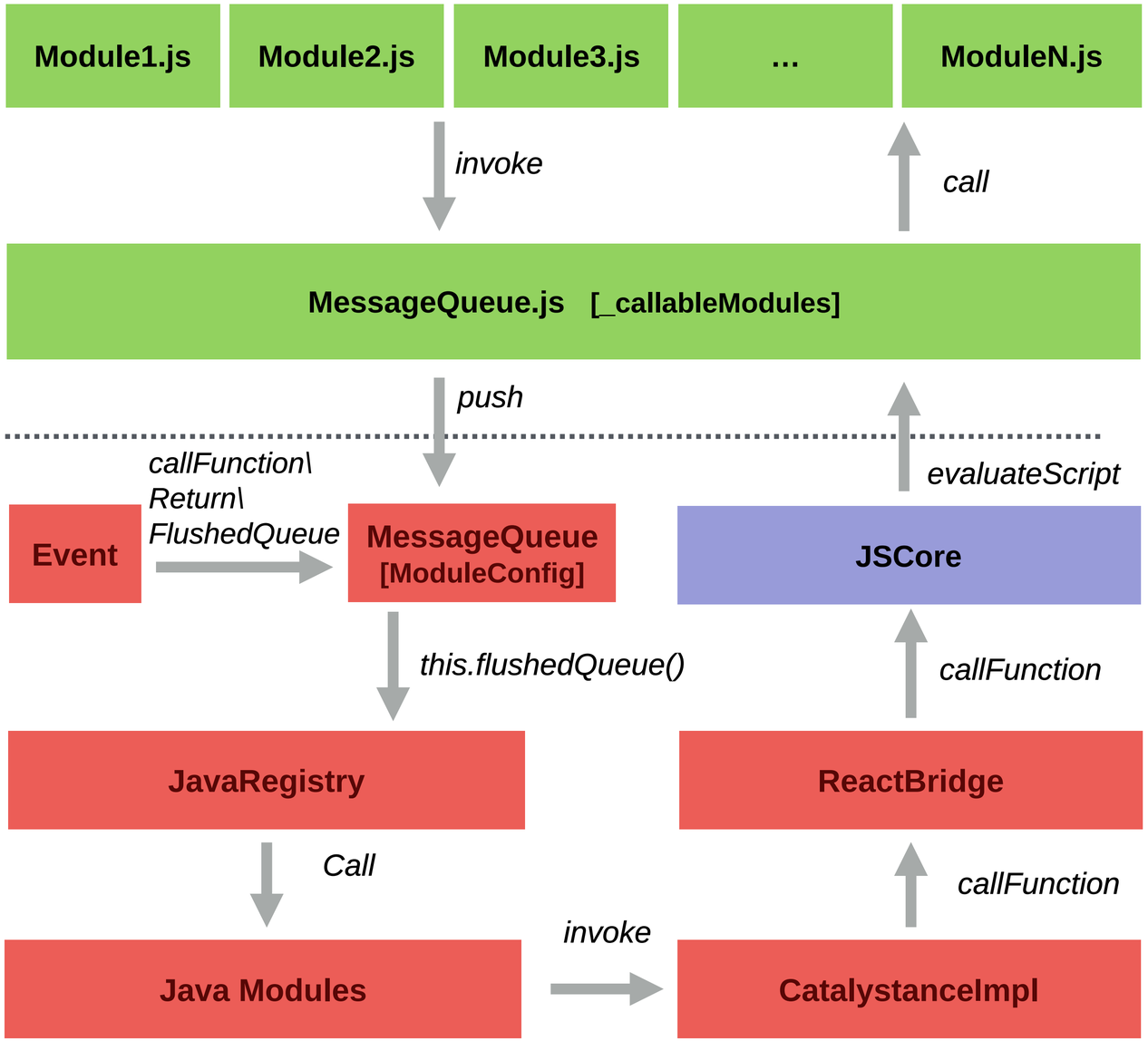

业务方
RN平台
APP端
需求
开发
发布
线上
提出
平台需求
确认需求
需求拆解
端能力
JS组件
平台升级
容灾方案
业务开发
与平台联调
壳工程
Chrome Developer Tools
交付端能力
交付组件
交付端能力
交付壳工程
准入
Bundle灰度
平台灰度
App灰度
发版
监控
JS迭代
bundle
source code
verify
auto build
app.bundle
- MD5+version id
- v29+v35
- -base.bundle
+
App
(Last daily build)
业务测试
平台准入
发布平台
- 放量规模
- 更新方式
- 更新日志
获取最新版本号
下载新包
md5校验
是否灰度用户
版本判断
用户App

性能优化

React Native 通讯流程



React Native 通讯流程



- 通讯频繁 by design
  JS拥有足够的灵活性
- 易导致卡顿
  - **事件频率高**
  - **JS运行时间过长**
  - **UI频繁重绘**
- 初始化成本较高
  - **生成模块配置表**

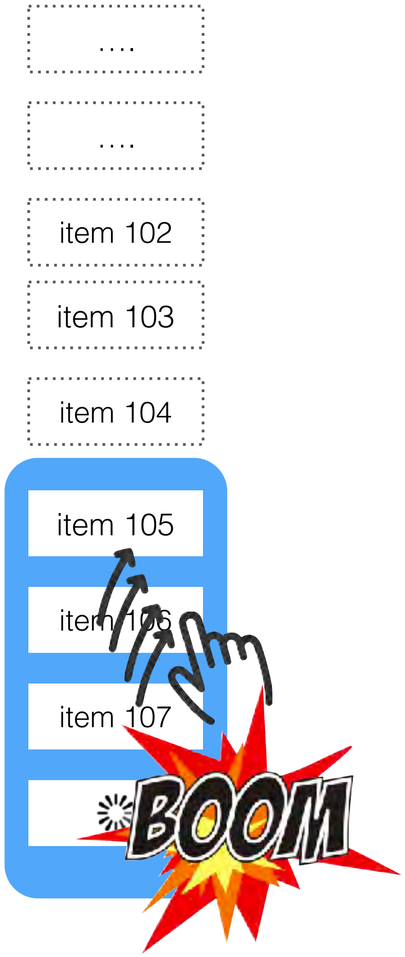
Listview优化

Listview @ React Native

initialListSize

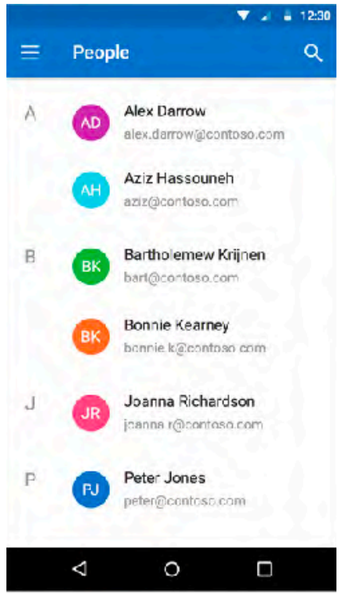
pageSize

renderFooter



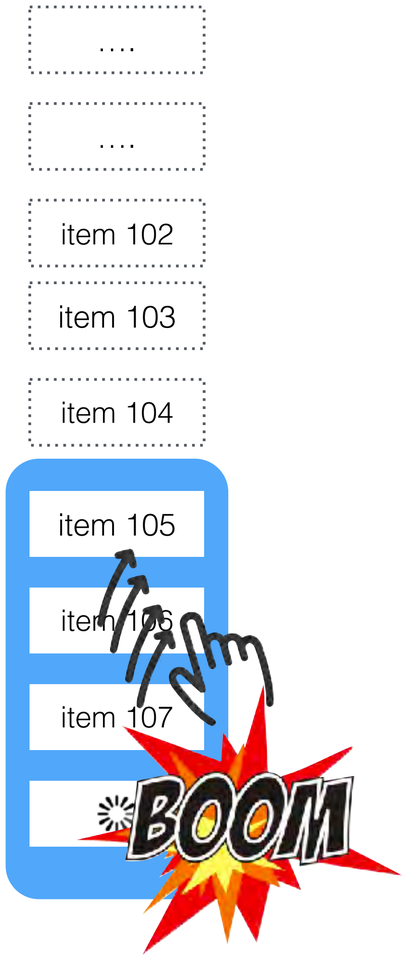
OOM

Listview @ Native



Anchor

Listview @ React Native

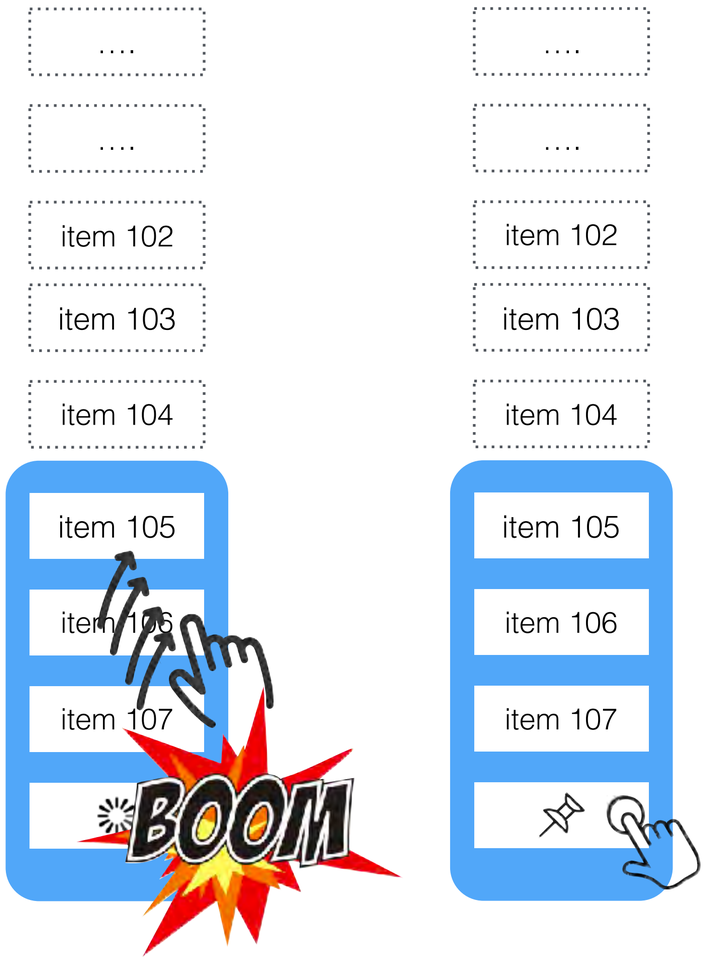


OOM

置顶

http://facebook.github.io/react-native/releases/0.31/docs/performance.html#listview-initial-rendering-is-too-slow-or-scroll-performance-is-bad-for-large-lists

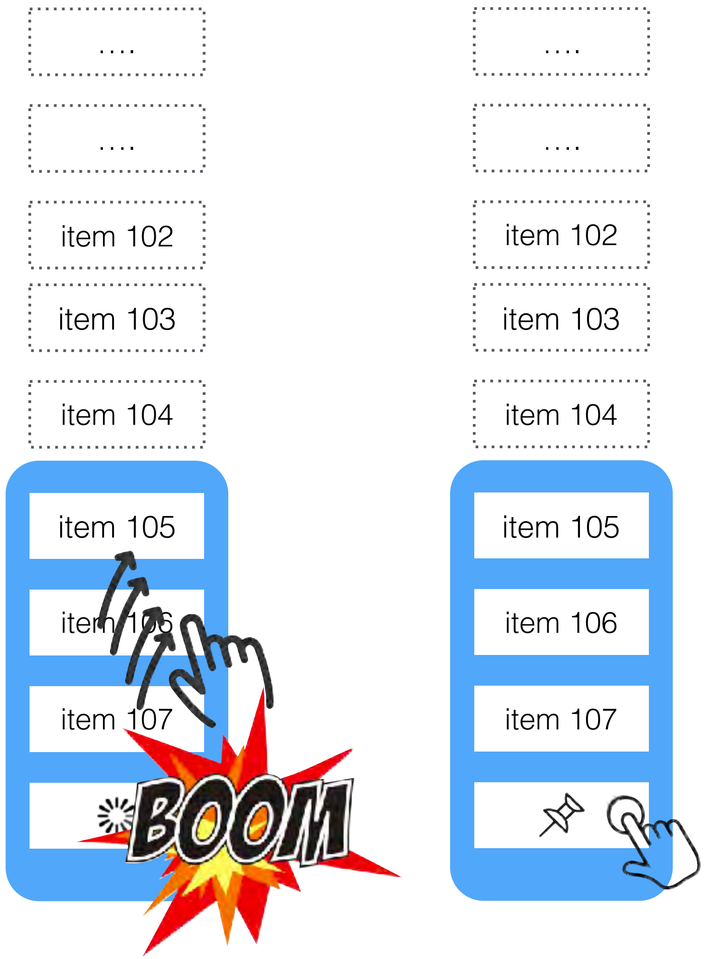
Listview 优化方案



OOM　　置顶

- 设计决定了View无法复用
  - **内存紧张**
- DOM diff雪上加霜
  - **大量重新渲染**
- 业务级优化
  - **预留空单元格**
  - **需大批量刷新时，从头渲染**

Listview 优化方案



OOM

置顶

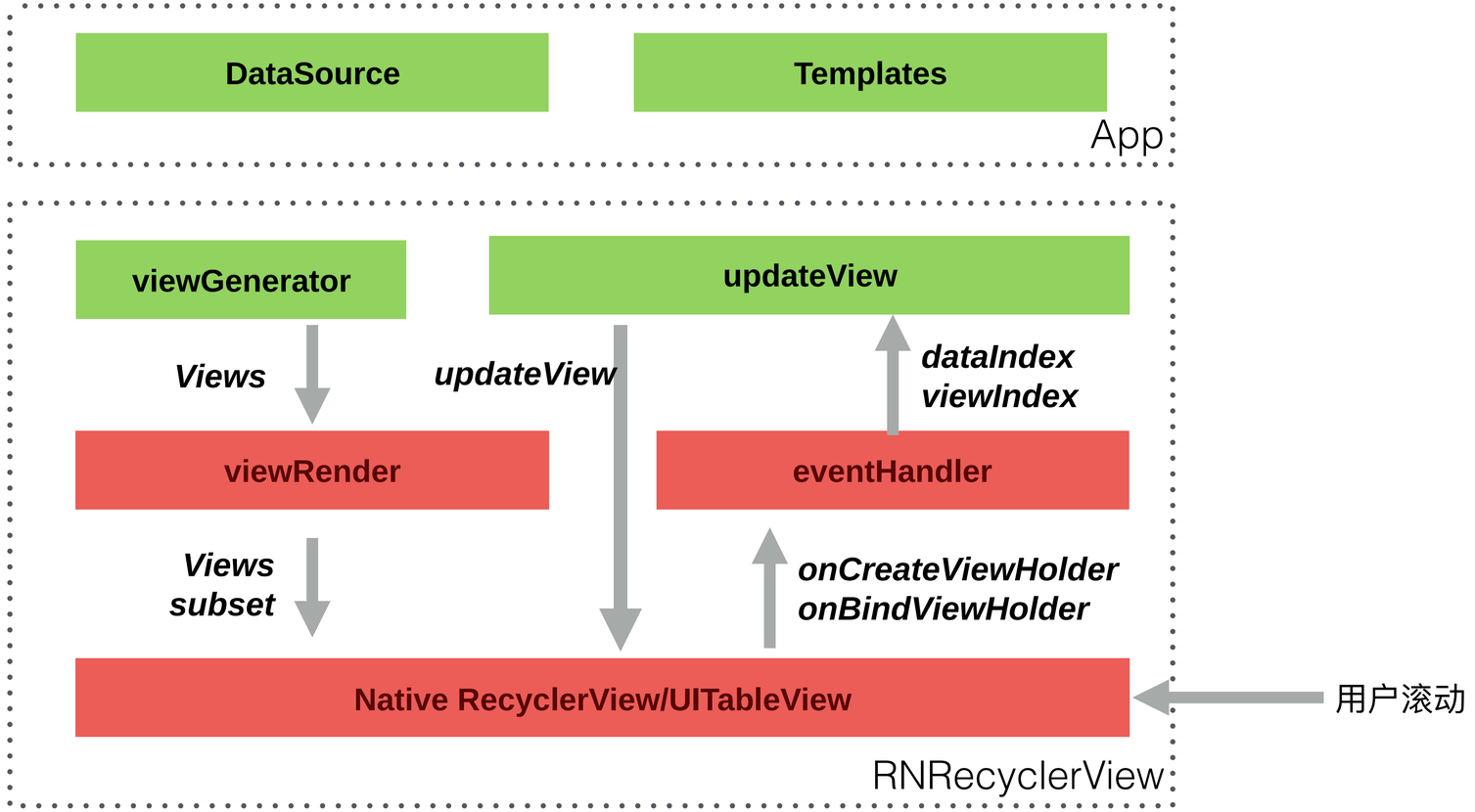
删除非可视区view


```
render() {
    return createElement(
        emptyView, {
            style: {
                width: this.viewProperties.width,
                height: this.viewProperties.height
            }
        });
}
```

- 实现十分简单
- 适合渲染强度低的列表
- 滚动过快会闪烁

Listview 优化方案

模板复用

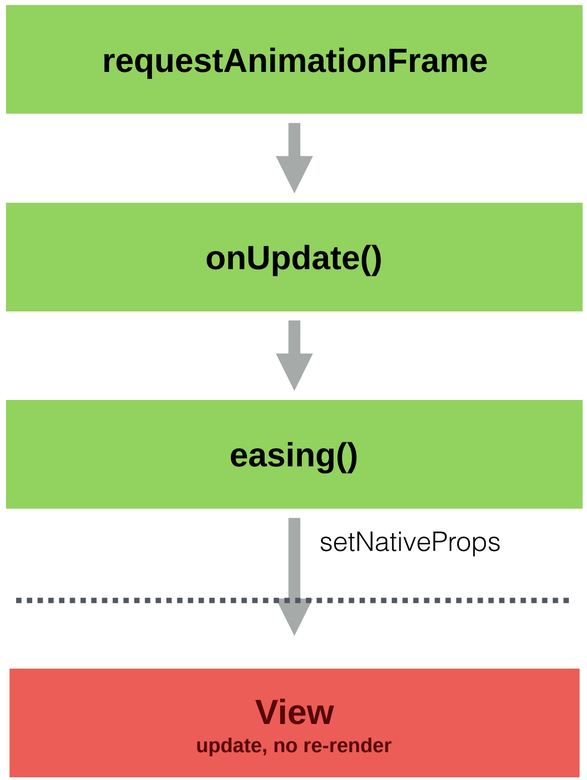


- 可彻底解决OOM

- 一定程度上违背了RN的设计原则
- item在滚动时可能会跳动、闪烁

动画优化

Animations @ React Native

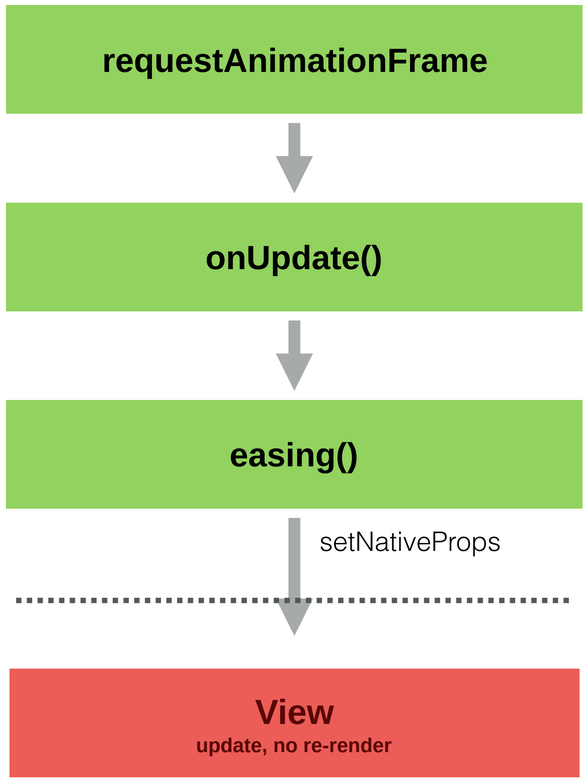


RN默认实现

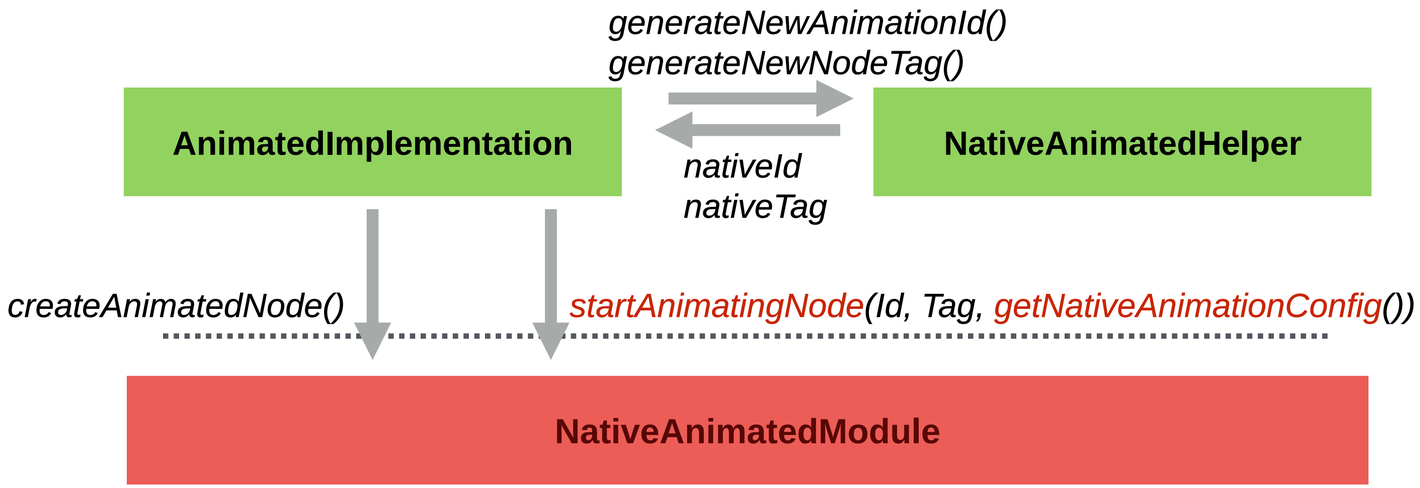
- 16ms
  - Javascript计算Props
  - 通讯
  - View Update
- 易导致丢帧
- systrace.py

https://facebook.github.io/react-native/docs/android-ui-performance.html
https://facebook.github.io/react-native/docs/animations.html
https://facebook.github.io/react-native/docs/direct-manipulation.html

Animations @ React Native



RN默认实现



useNativeDriver : true

- Native 直接完成动画绘制
  - **Javascript计算frames**
  - **通讯一次**
  - **View Update**
- 实验阶段，不够可靠

Animations 优化

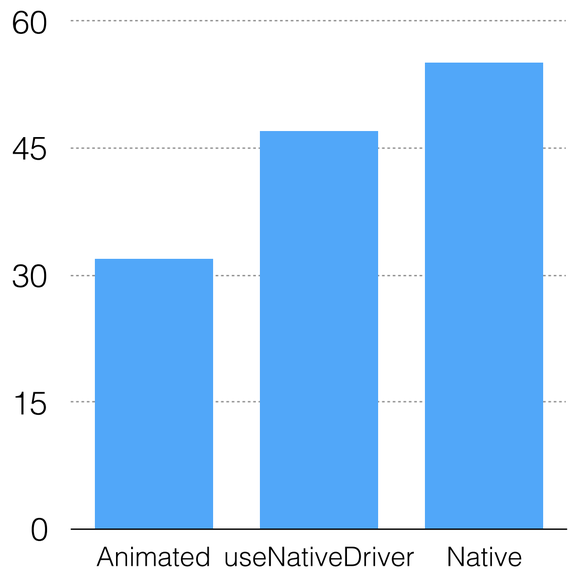
<AnimationView />

```
this.refs.view.
setTranslate({from, to}).
setOpacity(from, to).
setEasing({type: “bounce”}).
setDuration(1000).
onStart(() => {}).
start()
```

Pre-defined Animations

提供最常用的动画和easing实现

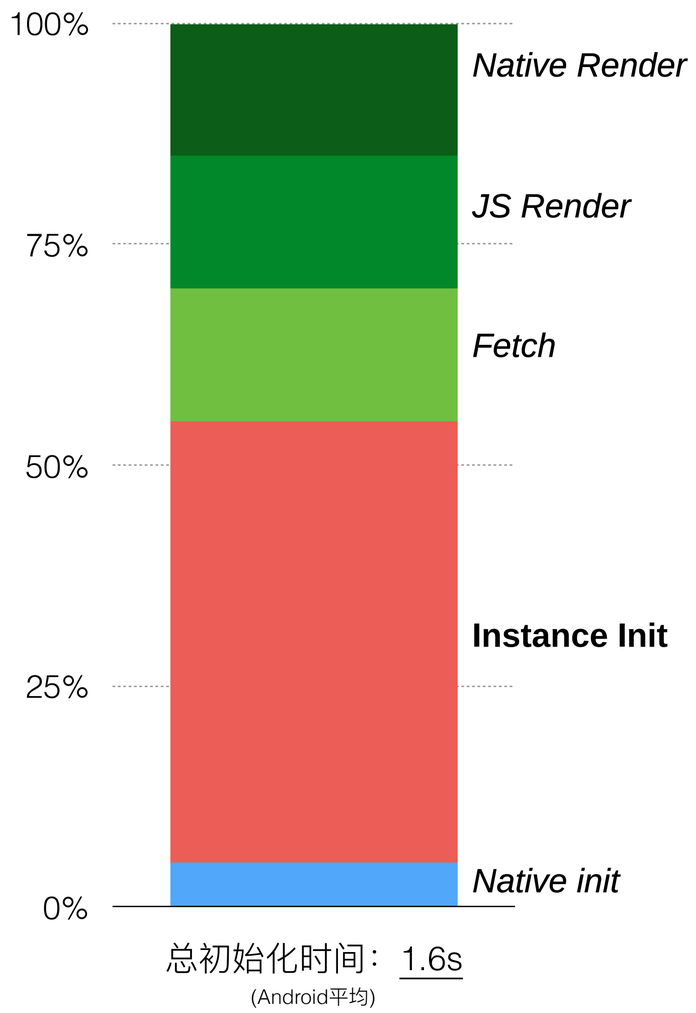
- **性能完全等价Native实现**
- **在长动画下优势明显**

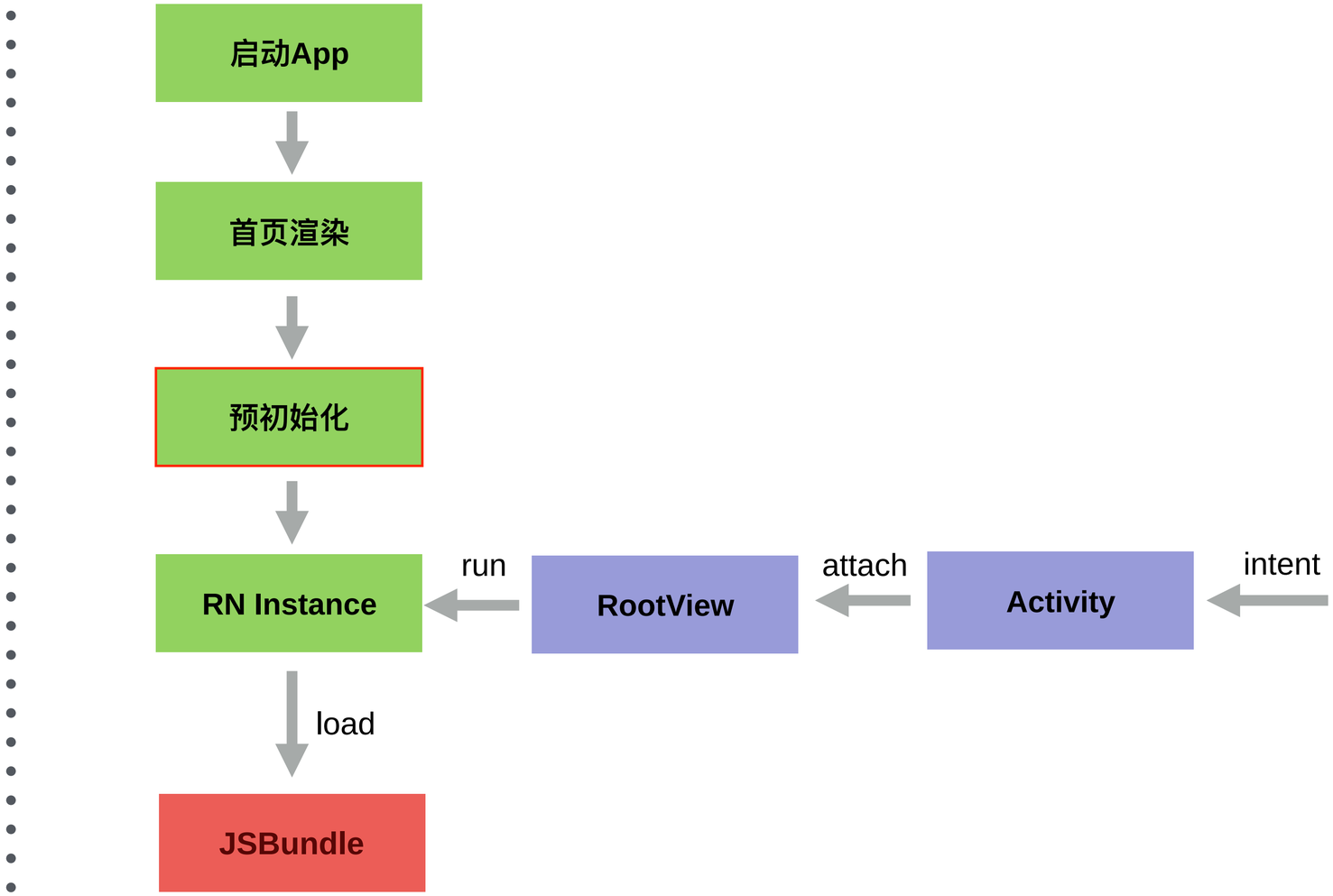


10条弹幕滚屏

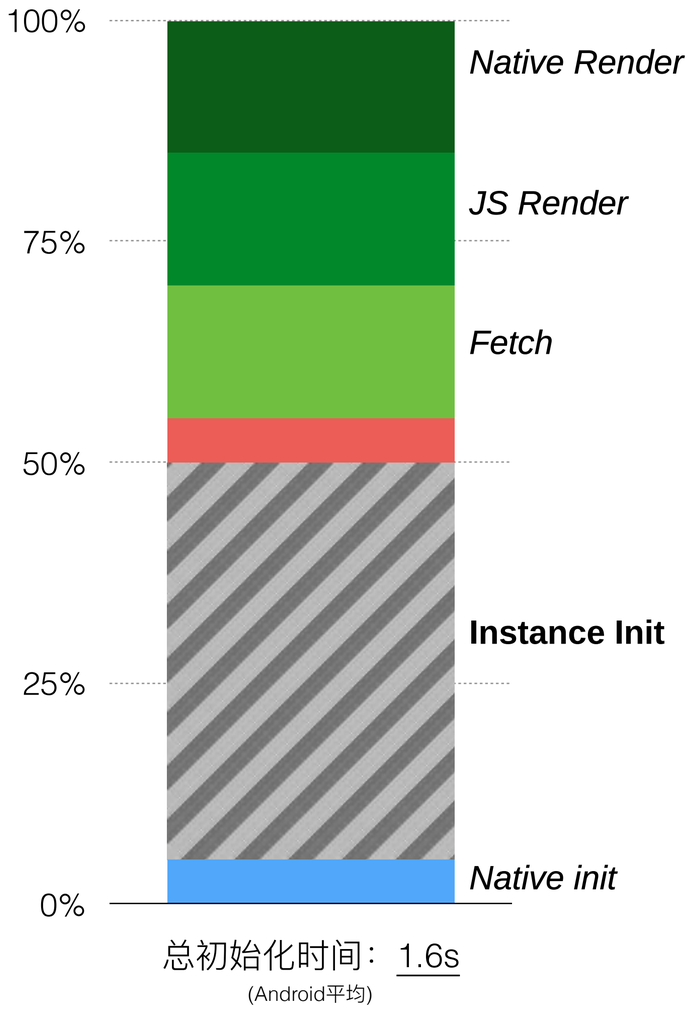
启动速度优化

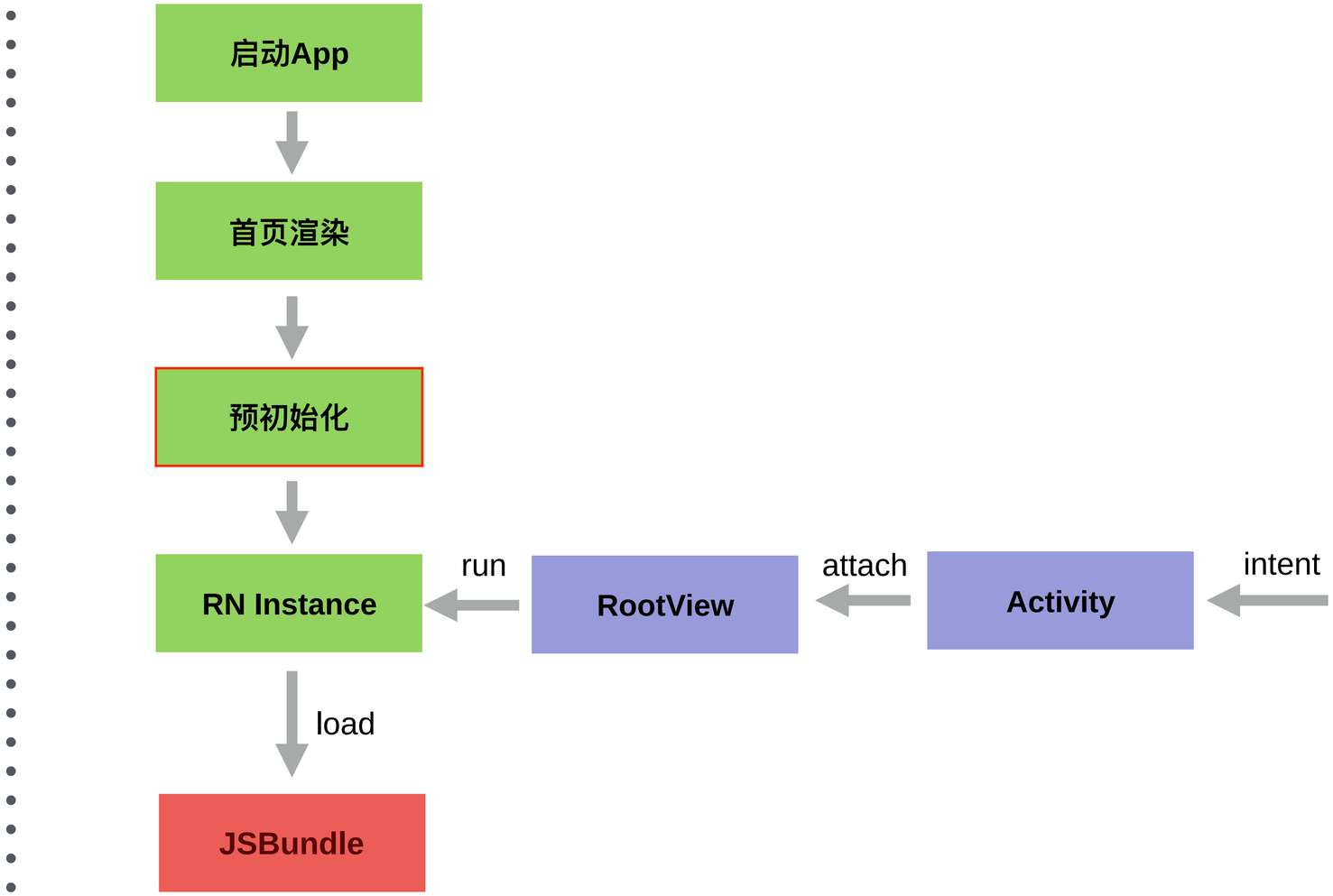
启动速度优化


100%
75%
50%
25%
0%
Native Render
JS Render
Fetch
Instance Init
Native init
总初始化时间：1.6s
(Android平均)

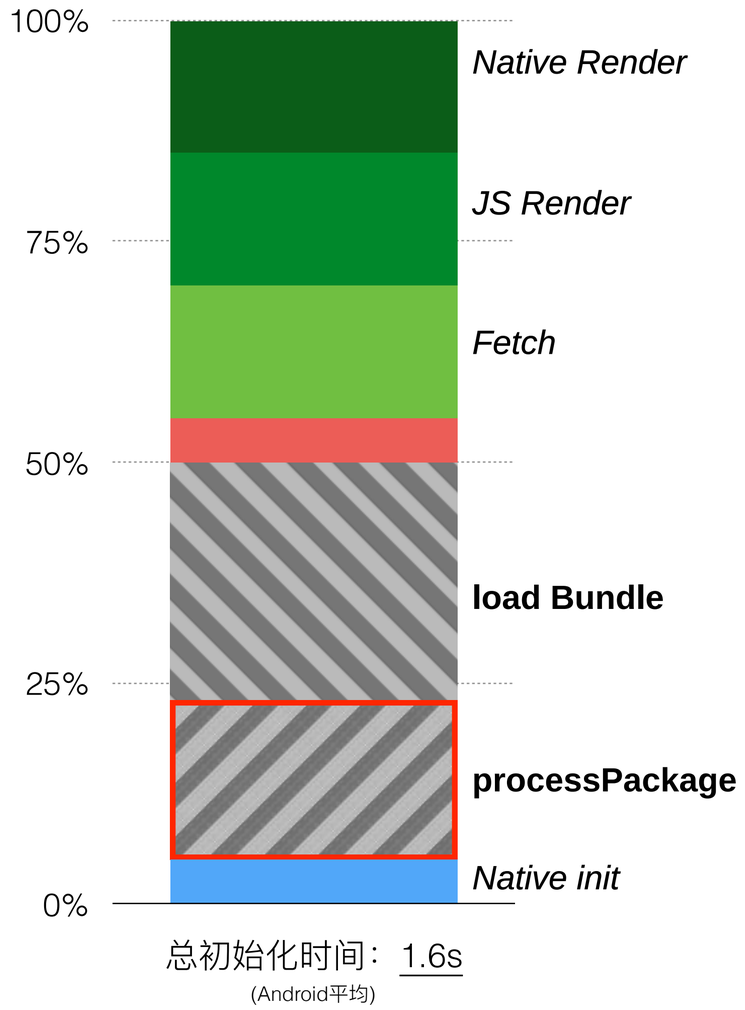

启动App
首页渲染
预初始化
RN Instance
run
RootView
attach
Activity
intent
load
JSBundle

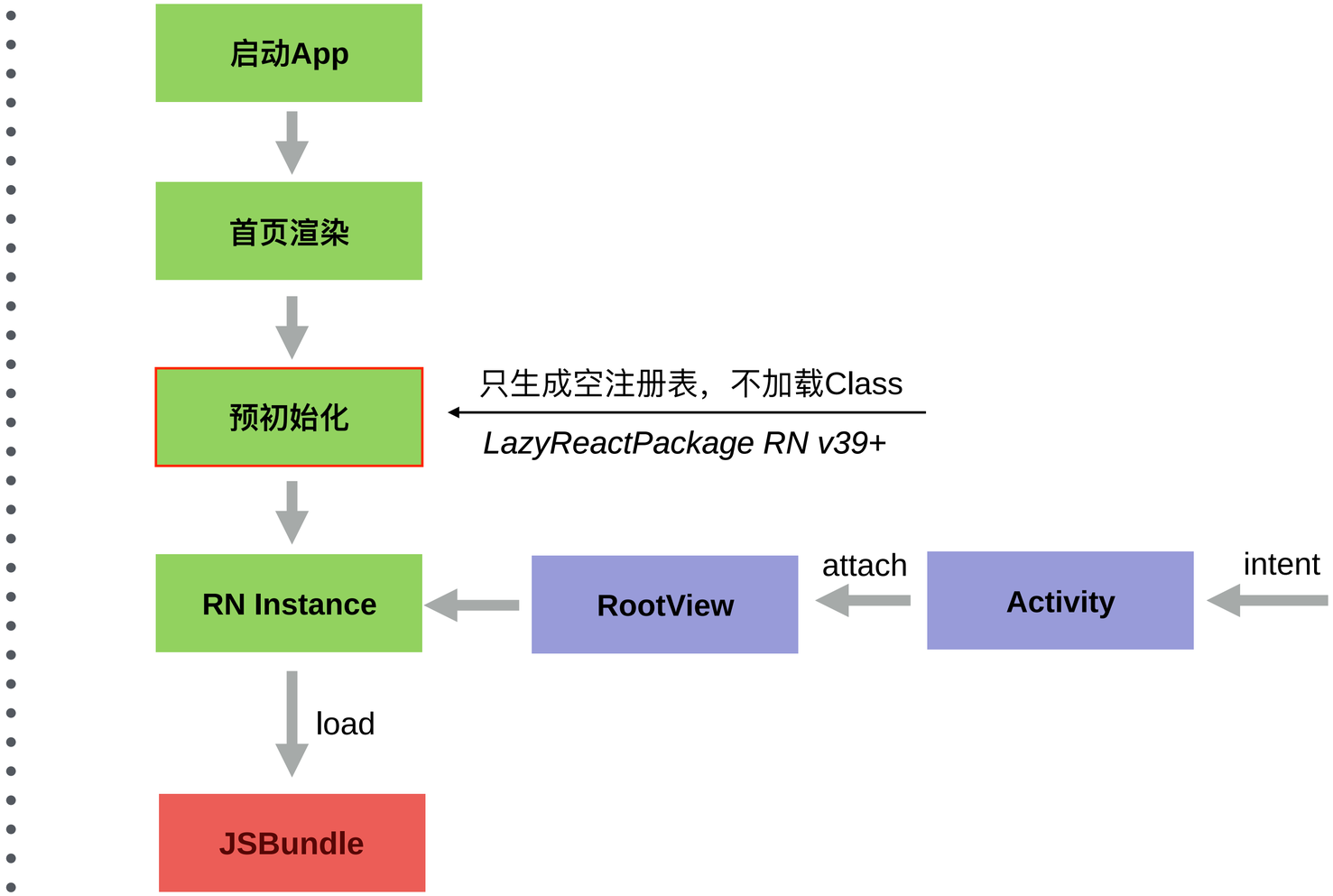
启动速度优化 – 预初始化


100%
75%
50%
25%
0%
Native Render
JS Render
Fetch
Instance Init
Native init
总初始化时间：1.6s
(Android平均)


启动App
首页渲染
预初始化
RN Instance
run
RootView
attach
Activity
intent
load
JSBundle

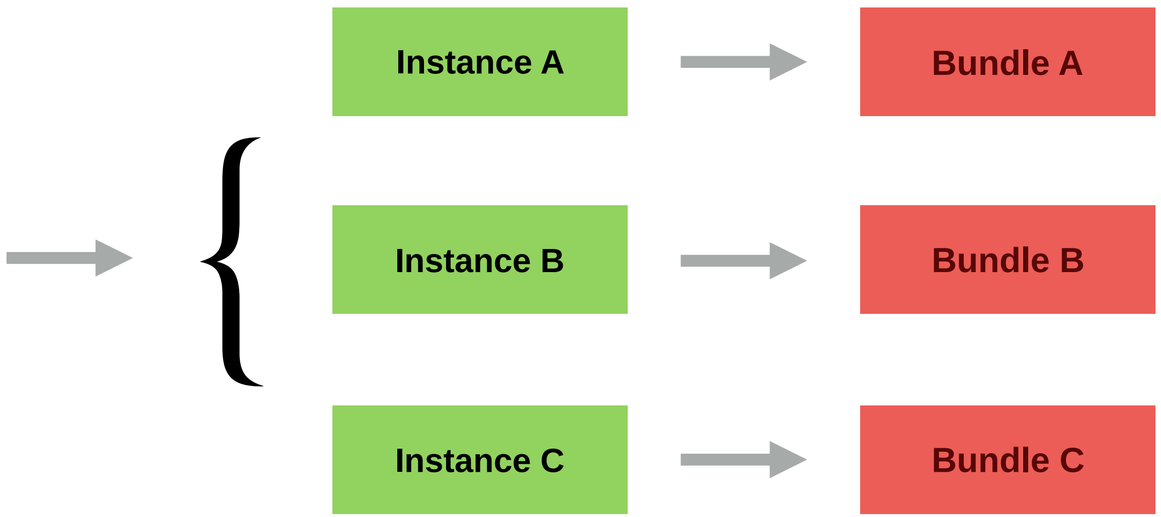
启动速度优化 – 延迟生成模块配置表





启动速度优化


中国联通
VPN
23:03
90%
2°
海淀
19 优
Bai du 百度
关注
视频
糯米
贴吧
新闻
小说
我现在就叫它神作，是不是有种钦定的感觉?
差评
跑男5杨洋替代鹿晗，取代杨颖居然瘦成竹竿的她
东方头条
谷歌地球VR震撼来袭 坐在家里环游世界成现实
按住说话 百度一下


Instance A
Bundle A
Instance B
Bundle B
Instance C
Bundle C

启动速度优化 – 单Instance多业务


```
Import App1 from ‘./path/to/app1’;
Import App2 from ‘./path/to/app2’;
Import App3 from ‘./path/to/app3’;
```

```
AppRegistry.registerComponent("App1", () => App1)
AppRegistry.registerComponent("App2", () => App2)
AppRegistry.registerComponent("App3", () => App3)
```

RCTRootView A

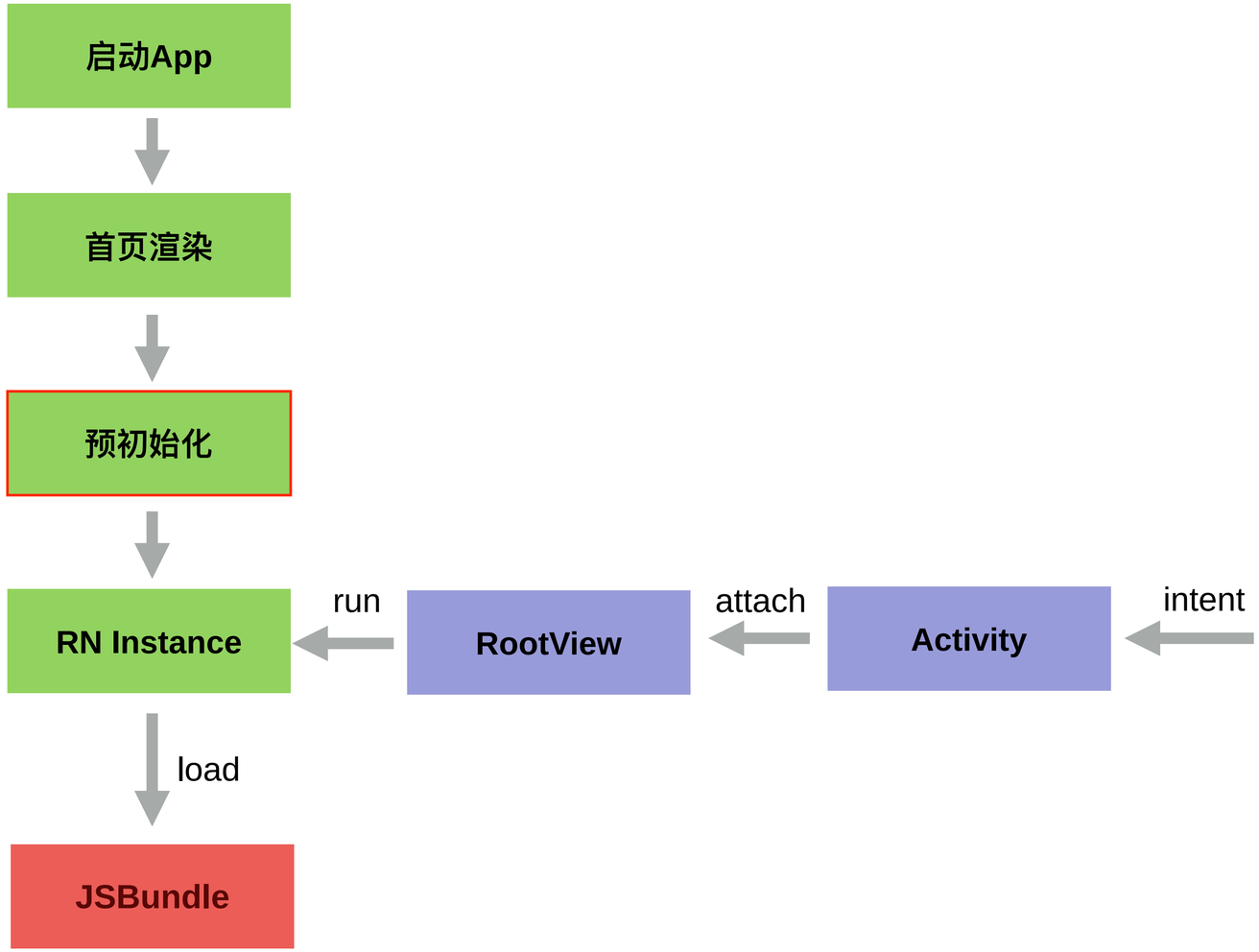
RCTRootView B

RCTRootView C

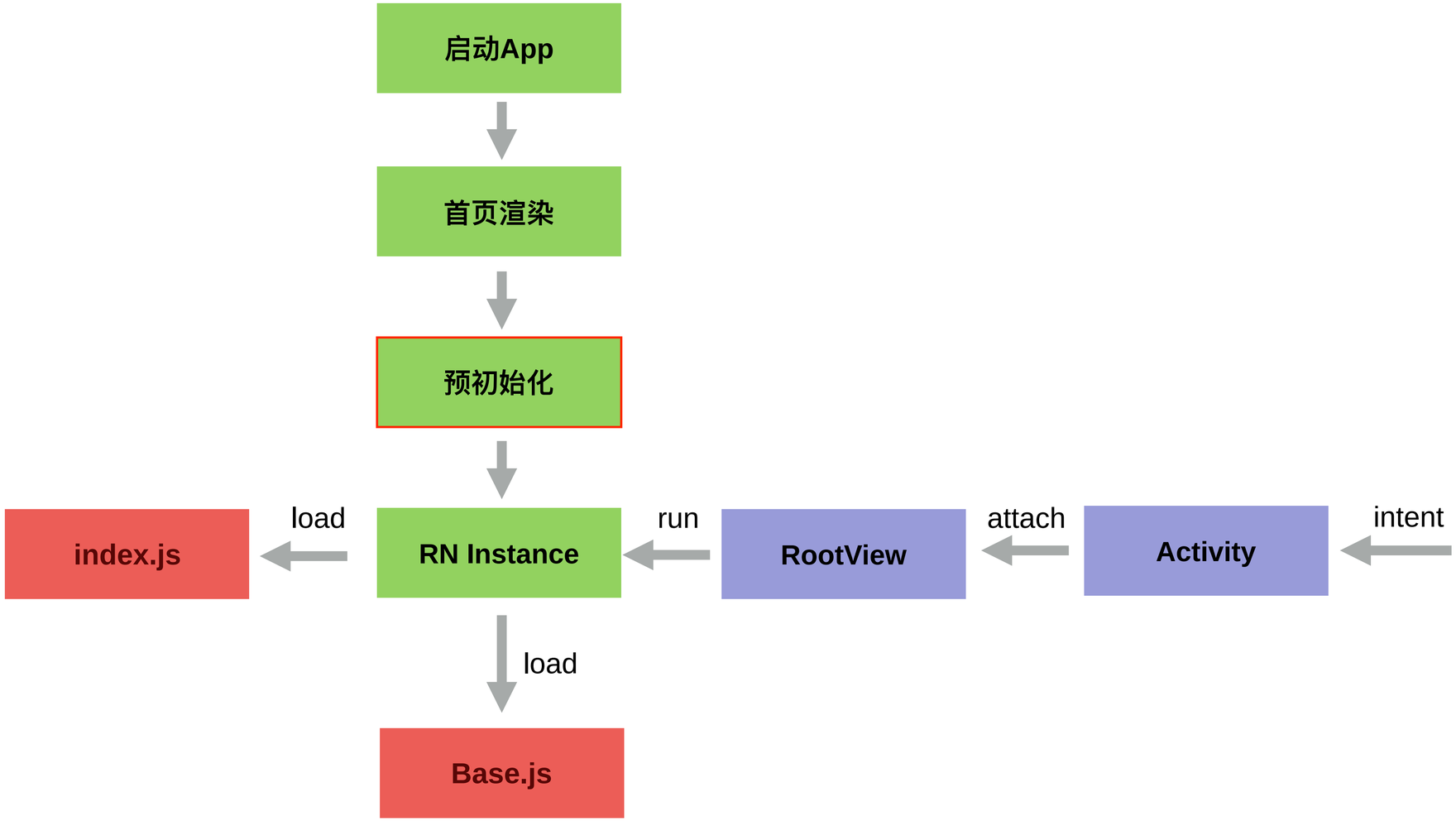
moduleName → Instance → Index Bundle

- 减轻预加载开销
  - 集中打包
  - component方式加载业务
  - 无法使用global变量

启动速度优化 – 按需加载


启动App
首页渲染
预初始化
RN Instance
run
RootView
attach
Activity
intent
load
JSBundle

启动速度优化 – 按需加载

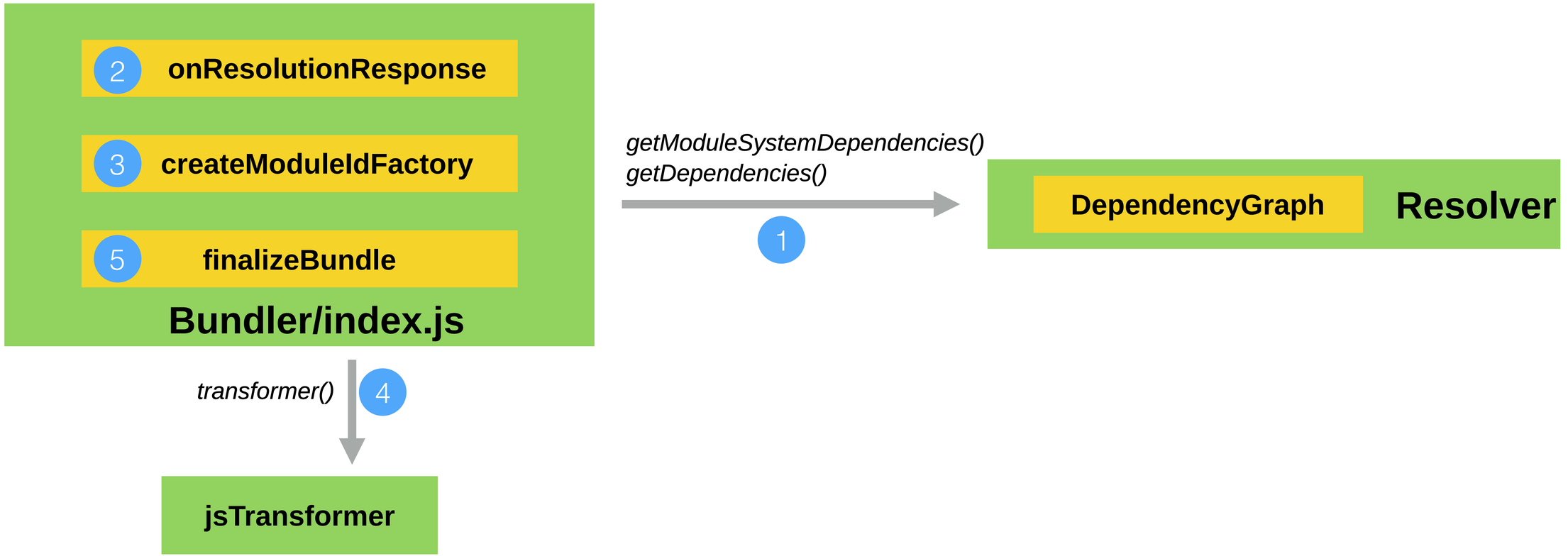

启动App
首页渲染
预初始化
index.js
load
RN Instance
run
RootView
attach
Activity
intent
load
Base.js

启动速度优化 – 按需加载

```
/*系统模块，全局定义，polyfill,babel*/
/*prelude.js*/!function(_){_.__DEV__=!1,_.__BUNDLE_START_TIME__=Date.now()}("undefined"!=typeof global?global:"undefined"
/*require.js*/!function(r){"use strict";function t(r,t){r in u||(u[r]={factory:t,hasError:!1,isInitialized:!1,exports:voi
/*polyfills.js*/!function(n){Object.assign=function(n,e){for(var f=1;f<arguments.length;f++){var l=arguments[f];if(null!=
/*console.js*/!function(n){function e(n){function e(e){return function(){var t;t=1===arguments.length&&"string"==typeof a
/*error-guard.js*/!function(r){function n(){var n=function(r){throw r};r.ErrorUtils.setGlobalHandler(n)}var l={_inGuard:0
/*Number.es6.js*/!function(e){void 0===Number.EPSILON&&Object.defineProperty(Number,"EPSILON",{value:Math.pow(2,-52)}),vo
/*String.prototype.es6.js*/!function(t){String.prototype.startsWith||(String.prototype.startsWith=function(t){"use strict
/*Array.prototype.es6.js*/!function(e){function r(e,r){if(null==this)throw new TypeError("Array.prototype.findIndex calle
/*Array.es6.js*/!function(n){Array.from||(Array.from=function(n){if(null==n)throw new TypeError("Object is null or undefi
/*Object.es7.js*/!function(e){!function(){var e=Object.prototype.hasOwnProperty;"function"!=typeof Object.entries&&(Objec
/*babelHelpers.js*/!function(e){var r=e.babelHelpers={};r.createRawReactElement=function(){var e="function"==typeof Symbo
/*入口模块*/
__d(0/*index.js*/,function(e,t,r,n){var i=t(12),u=(babelHelpers.interopRequireDefault(i),t(41)),l=t(377),o=babelHelpers.i
/*react-native基础模块*/
__d(12 /*./node_modules/react/react.js*/,function(t,s,c,e){"use strict";c.exports=s(13)});
__d(13 /*./node_modules/react/lib/React.js*/,function(e,t,r,n){"use strict";var a=t(14),o=t(15),c=t(27),l=t(30),s=t(31),i
__d(14 /*./node_modules/object-assign/index.js*/,function(r,e,t,n){"use strict";function o(r){if(null===r||void 0===r)thr
__d(15 /*./node_modules/react/lib/ReactChildren.js*/,function(t,n,u,e){"use strict";function r(t){return(""+t).replace(k,
__d(16 /*./node_modules/react/lib/PooledClass.js*/,function(n,o,e,t){"use strict";var r=o(17),i=(o(18),function(n){var o=
...
__d(177 /*./node_modules/react-native/Libraries/JavaScriptAppEngine/Initialization/InitializeJavaScriptAppEngine.js*/,fun
...
__d(375,function(r,t,e,n){"use strict";var i=t(17),a=t(15),o=t(19),u=t(22),f=(t(18),t(21),{create:function(r){if("object"
__d(376,function(r,o,n,i){"use strict";function t(r){return Array.isArray(r)?r.concat():r&&"object"==typeof r?e(new r.con
/*业务模块*/
__d(377,function(e,t,r,l){Object.defineProperty(l,"__esModule",{value:!0});var a=t(12),n=babelHelpers.interopRequireDefau
__d(378,function(e,t,d,n){"use strict";function o(e){return e&&e.__esModule?e:{default:e}}n.__esModule=!0,n.connect=n.Pro
__d(379,function(e,t,r,o){"use strict";function n(e){return e&&e.__esModule?e:{default:e}}function i(e,t){if(!(e instance
__d(380,function(e,s,u,i){"use strict";i.__esModule=!0;var p=s(12);i.default=p.PropTypes.shape({subscribe:p.PropTypes.fun
__d(381,function(o,e,n,r){"use strict";function t(o){"undefined"!=typeof console&&"function"==typeof console.error&&conso
__d(382,function(t,e,r,o){"use strict";function s(t){return t&&t.__esModule?t:{default:t}}function n(t,e){if(!(t instance
__d(383,function(e,t,r,n){"use strict";function u(e,t){if(e===t)return!0;var r=Object.keys(e),n=Object.keys(t);if(r.lengt
__d(384,function(n,t,r,u){"use strict";function e(n){return function(t){return(0,i.bindActionCreators)(n,t)}}u.__esModule
__d(385,function(e,t,d,o){"use strict";function r(e){return e&&e.__esModule?e:{default:e}}o.__esModule=!0,o.compose=o.app
__d(386,function(e,t,n,r){"use strict";function o(e){return e&&e.__esModule?e:{default:e}}function i(e,t,n){function r(){
__d(387,function(t,r,n,o){function c(t){if(!a(t)||e(t)!=i)return!1;var r=u(t);if(null===r)return!0;var n=s.call(r,"constr
__d(388,function(n,t,o,i){function e(n){return null==n?void 0===n?f:d:(n=Object(n),g&&g in n?r(n):u(n))}var c=t(389),r=t(
/*执行系统模块和运行系统模块*/
;require(177);/*InitializeJavaScriptAppEngine*/
;require(0);/*入口模块*/
```

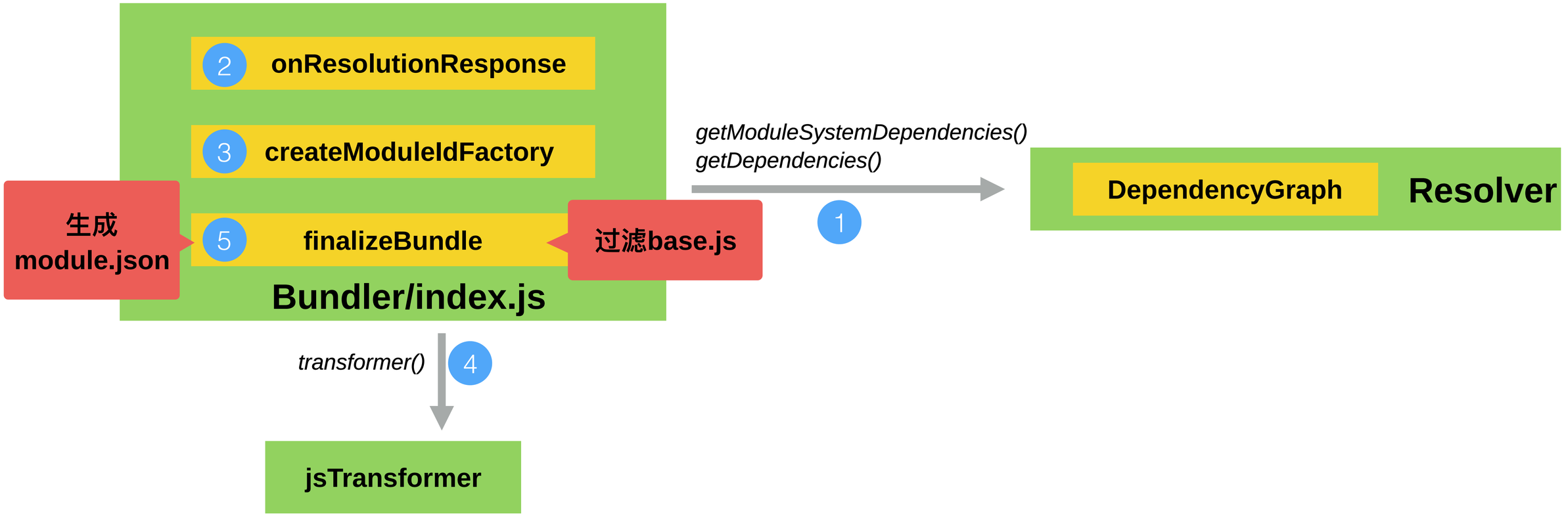
- bundle.js
  - 模块化加载每个module
  - 每行一个module
  - moduleId是自增数字

启动速度优化 – 按需加载



react-packger打包流程

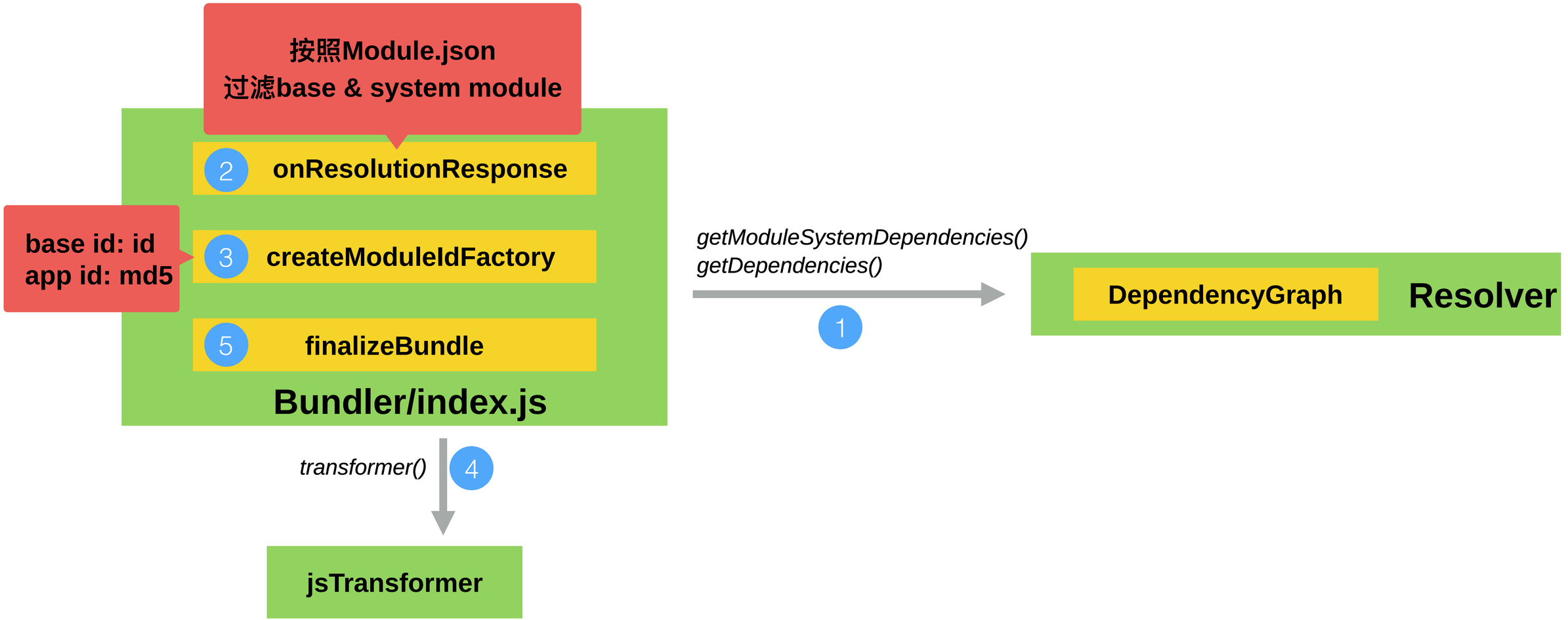
启动速度优化 – 按需加载



打包 base.bundle（修改后）

react-native —entry-file base.js —bundle-output ./base.bundle **—module-ouput ./module.json**

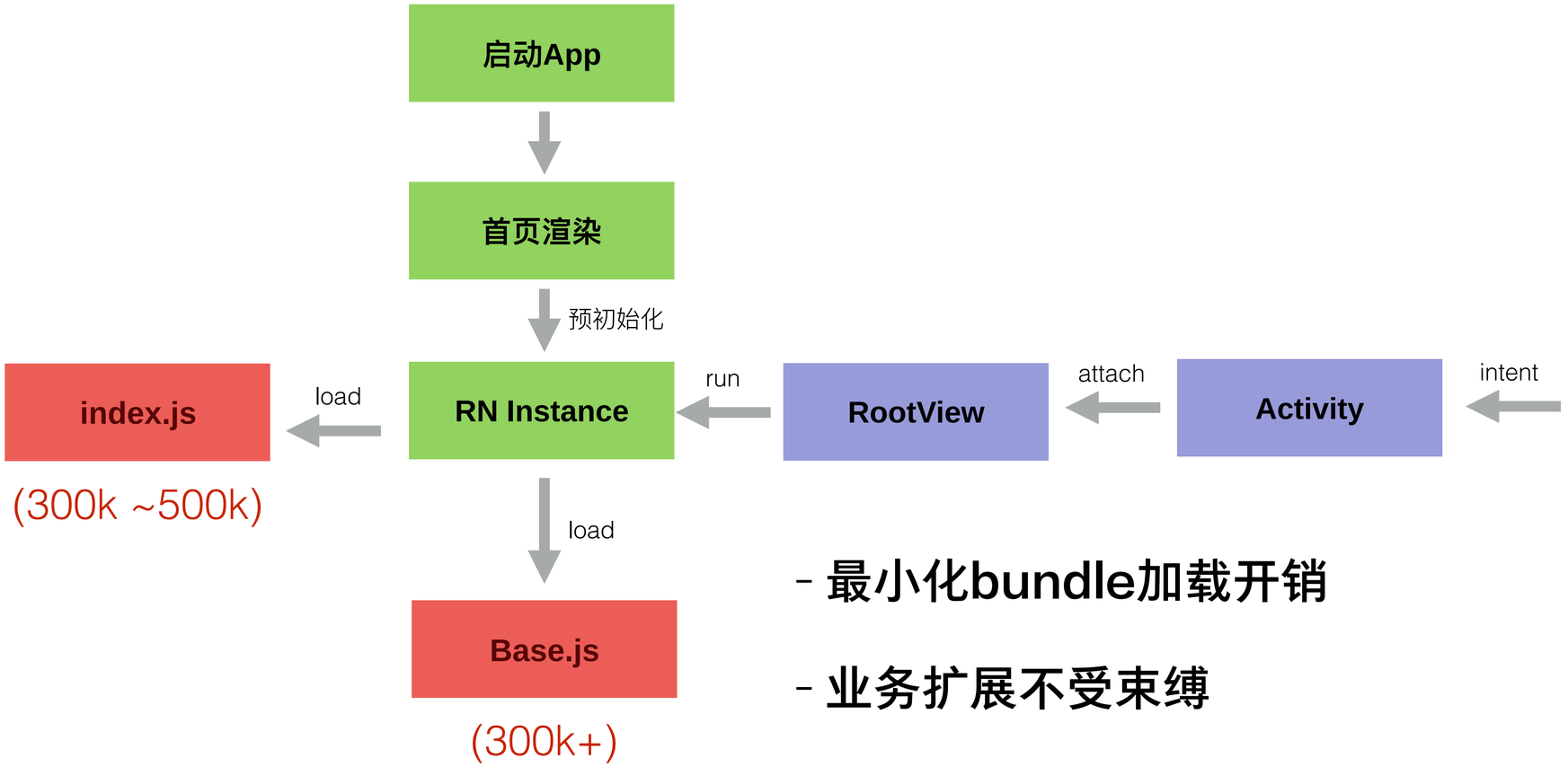
启动速度优化 – 按需加载



打包 app.bundle（修改后）

react-native —entry-file index.js —bundle-output ./index.bundle **—module-input ./module.json**

启动速度优化 – 按需加载



- 最小化bundle加载开销
- 业务扩展不受束缚

其他

- 延迟加载、渲染
- rowHasChanged/shouldComponentUpdate/pureRender
- requestAnimationFrame

- https://facebook.github.io/react-native/docs/performance.html
- https://code.facebook.com/posts/895897210527114/dive-into-react-native-performance/

React Native的工程价值

跨端方案对比

	优点/特点	缺点	适用场景
APK插件	框架成本低 性能、体验一步到位	开发成本高 iOS依赖发版	对性能和体验要求高 同时**相对稳定**
Hybrid	移植成本极低 通过端能力优化特定交互 业务随时发版	交互相对简陋 端能力依赖发版	**快速迭代 或 短期运营** 需要接入第三方 支持**自定义内容展示**
React Native	流畅度近似原生应用 开发成本相对Native较低 业务随时发版	有学习和开发成本 要求稳定性和平台化程度	**兼顾快速迭代和体验** **平台级app**

SUMMARY

- React Native 已能在超大规模的app中使用
  - **在发版周期、开发成本和使用体验之间找平衡**
- 使用React Native是个系统性工程
  - **做到有机融入App**
  - **确保全流程控制和质量保证**
- 几个作用较大的性能优化点
  - **Listview/动画/启动速度**

Thanks

@berg

Baidu 百度