



艺龙Enjoy

艺龙基于React(Native)的跨平台开发实践



❖ 一套代码、多端运行

❖ 优化启动速度

❖ 出错降级

一套代码、多端运行



A decorative geometric pattern in the top right corner, consisting of a cluster of colorful, multi-faceted shapes in shades of blue, green, yellow, and pink, resembling a stylized explosion or a cluster of crystals.

写ReactNative —> 兼容Web

or

写Web —> 兼容ReactNative

写Web —> 兼容ReactNative

❖ 让RN支持HTML标签

丰富的语义化标签

❖ 让RN支持CSS样式表

丰富的选择器、强大的样式继承

❖ 统一API

抹平各平台下API使用上的差异



如何让RN支持所有HTML标签？



将每种标签封装一个RN组件？



将HTML标签分类

❖ 布局类

div、p、span、h1-6...

❖ 功能类

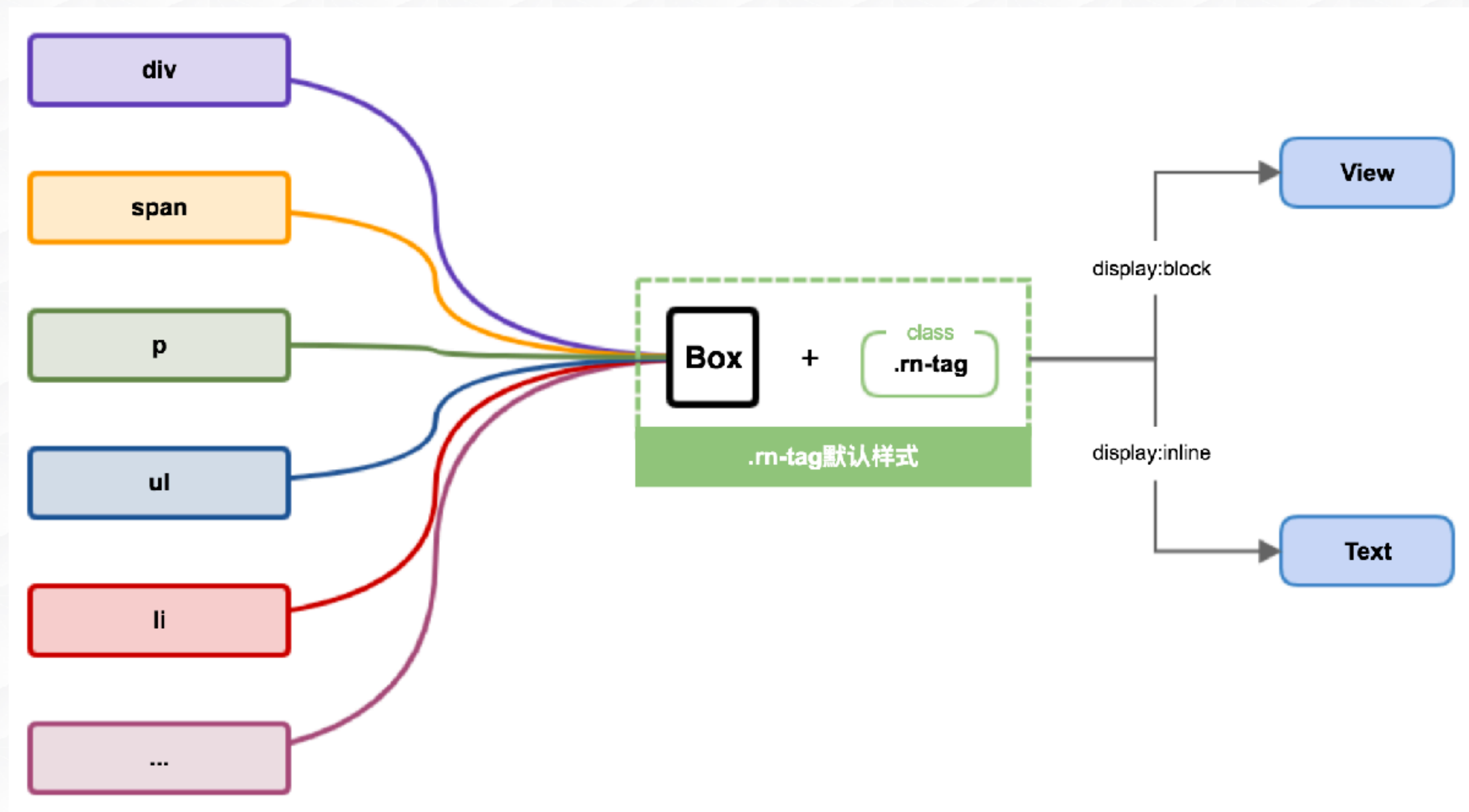
input、img...

❖ 混合类

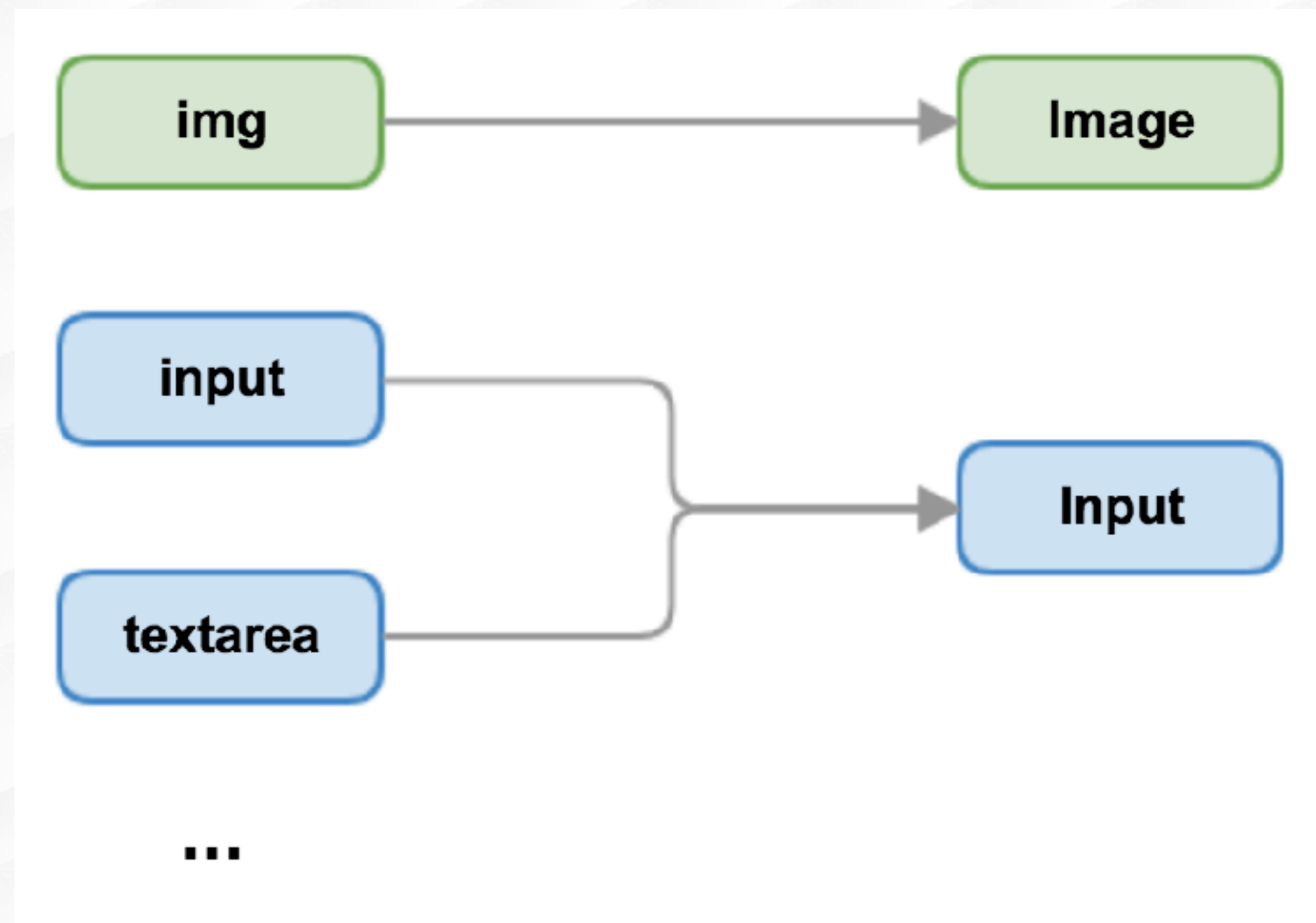
a、form...



布局类标签



功能类标签



混合类标签




混合类标签，在布局类标签的基础上，又包含了一些特殊功能，下面以a标签为例：

1. 继承自Box
2. 编译时，将页面的路由实例注入到A标签的属性中
3. 监听click事件，调用路由实例，实现href跳转

如何让RN支持CSS样式表？



- 
1. CSS -> JSON
 2. 建立节点关系树
 3. 关联HTML与CSS样式表
 4. 解析选择器，匹配、计算每个节点应用的样式规则
 5. 抹平RN与WEB的样式差异，实现RN不支持的样式规则

CSS -> JSON编译时

- ❖ 转换CSS属性名
- ❖ 拆分缩写格式
- ❖ 过滤平台前缀
- ❖ 兼容性校验

```
.class1,.class2{  
  padding: 10px 20px 15px;  
  background: #fff url(./pic/bg.png) no-repeat center center;  
}  
.class1 span{  
  color: red;  
}  
.class2 span{  
  color: blue;  
}
```

```
export default new StyleSheet({  
  rules: {  
    "0": {  
      paddingTop: "10px",  
      paddingHorizontal: "20px",  
      paddingBottom: "15px",  
      backgroundColor: "#fff",  
      backgroundImage: require("./pic/bg.png"),  
      backgroundRepeat: "no-repeat",  
      backgroundPosition: "center center"  
    },  
    "1": {  
      color: "red"  
    },  
    "2": {  
      color: "blue"  
    }  
  },  
  index: {  
    ".class1": [{  
      key: "0"  
    }],  
    ".class2": [{  
      key: "0"  
    }],  
    ".rn-span": [{  
      selector: ".class1 .rn-span",  
      key: "1"  
    }, {  
      selector: ".class2 .rn-span",  
      key: "2"  
    }]  
  }  
});
```


CSS -> JSON运行时

- ❖ 格式化选择器
- ❖ 用末级选择器建立索引

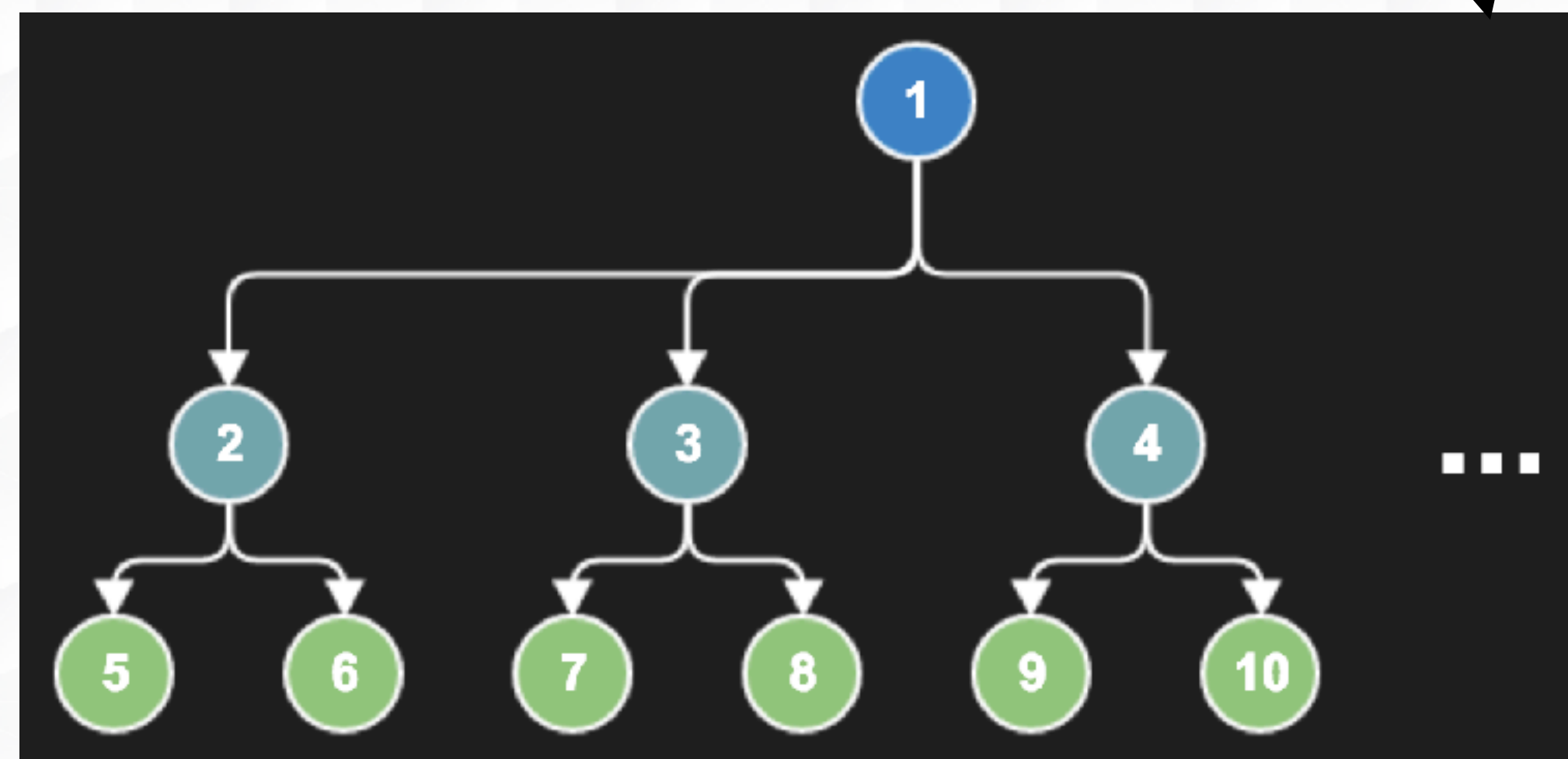
```
"#mod .container>.item[a=123] .rn-a:focus"
=>
{
  selector: ".rn-a",
  psuedoClass: ["focus"],
  parent: {
    selector: ".item",
    attrs: ["a=123"],
    onlyParent: false,
    parent: {
      selector: ".container",
      onlyParent: true,
      parent: {
        selector: "#mod",
        onlyParent: false
      }
    }
  }
}
```


建立节点关系树

```
<ul>
  {
    this.state.items.map(
      item => <li>
        <span>{ item.name }</span>
        <span>{ item.price }</span>
      </li>
    )
  }
</ul>
```

```
<ul _id="1">
  <li _id="2">
    <span _id="5">...</span>
    <span _id="6">...</span>
  </li>
  <li _id="3">
    <span _id="7">...</span>
    <span _id="8">...</span>
  </li>
  <li _id="4">
    <span _id="9">...</span>
    <span _id="10">...</span>
  </li>
  ...
</ul>
```

```
<ul _id={ _id() }>
  {
    this.state.items.map(
      item => <li _id={ _id() }>
        <span _id={ _id() }>{ item.name }</span>
        <span _id={ _id() }>{ item.price }</span>
      </li>
    )
  }
</ul>
```



关联HTML与CSS



1. 将组件关联的CSS样式表，注入到组件类的一个静态属性中
2. 将这个静态属性注入到HTML片段最外层标签的属性中

解析选择器，匹配、计算节点样式



1. 节点根据关系树，递归查找所有父级节点中关联的CSS样式表
2. 用当前节点的tagName、class、id、属性等，通过样式表末级选择器索引，找出所有匹配的选择器
3. 根据节点关系树，过滤出有效的选择器
4. 计算选择器权重，进行排序
5. 前面加入父节点可继承样式，后面加入节点内联样式
6. 合并样式规则，生成当前节点生效的样式规则

抹平RN与WEB样式差异

❖ 抹平RN与WEB表现不一致的样式

如flex布局，WEB默认是横向排列，RN默认是纵向排列

如line-height，WEB、IOS、Android纵向对齐方式不一致

...

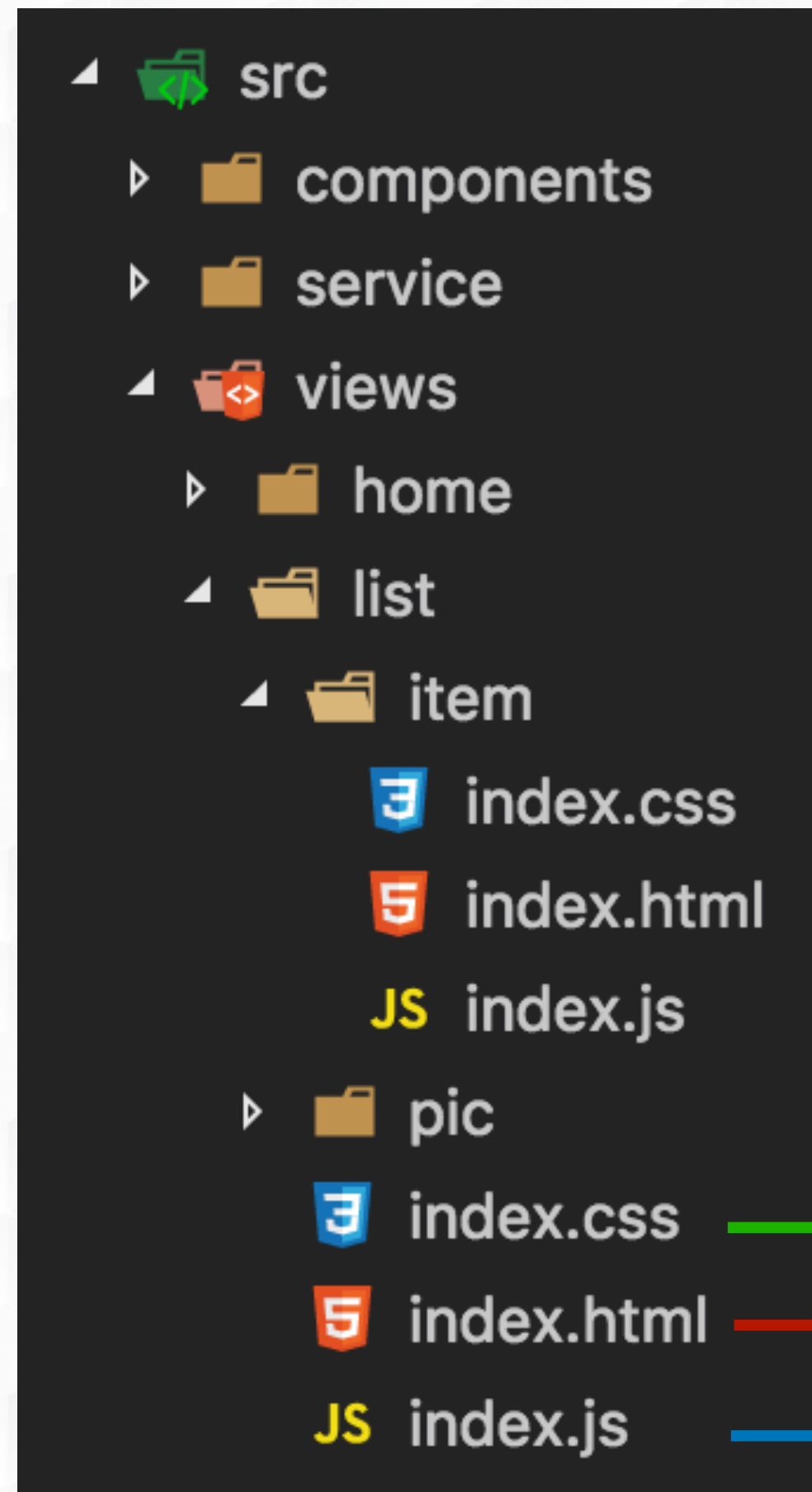
❖ 实现RN不支持的WEB样式规则

如background-image、css3动画、rem单位等

抹平各平台的API差异



源代码示例



```
.list-con-item:last-child .con-item{
  border-right: none !important;
}
.list-con-item span{
  max-width: 60px;
  overflow: hidden;
  text-align: center;
  font-size: 1rem;
  white-space: nowrap;
}
.list-con-item i{
  background: transparent url(./pic/down@3x.png) center center no-repeat;
  background-size: 50% 50%;
  width: 15px;
  height: 15px;
  margin-left: 3px;
}
```

```
<html>
  <import type="component" name="Item" path="./item" />
  <head>
    <title>酒店列表</title>
  </head>
  <body>
    <div class="main">
      <!-- top -->
      <div class="list-top">...
      </div>
      <!-- component -->
      <ul class="list-condition">
        <li class={{
          "list-con-item": true,
          "list-con-active": this.state.sortShow || this.state.sortShowValue !== '',
          "list-con-show": this.state.sortShow
        }}>
```

```
@layers({
  priceStart,
  keyword,
  citySelector
})
export default class List extends Component {
  static DEFAULT_TEXT = {
    keywordText: '关键词/酒店/地址',
    priceStarText: '价格/星级'
  }
  constructor(props){
    super(props);
    this.state = {
      city: {
        cityName: '北京',
        cityId: '0101'
      },
      checkIn: (new Date).format('yyyy-MM-dd'),
```


运行示例



IOS



Android



H5

优化启动速度





❖ 拆包

将代码按照不同的维度，拆分成粒度不同的多个文件

❖ 加载

在不同的时机，按需加载文件

❖ 模块管理

多业务线，框架共享、业务隔离

拆包

一个bundle包



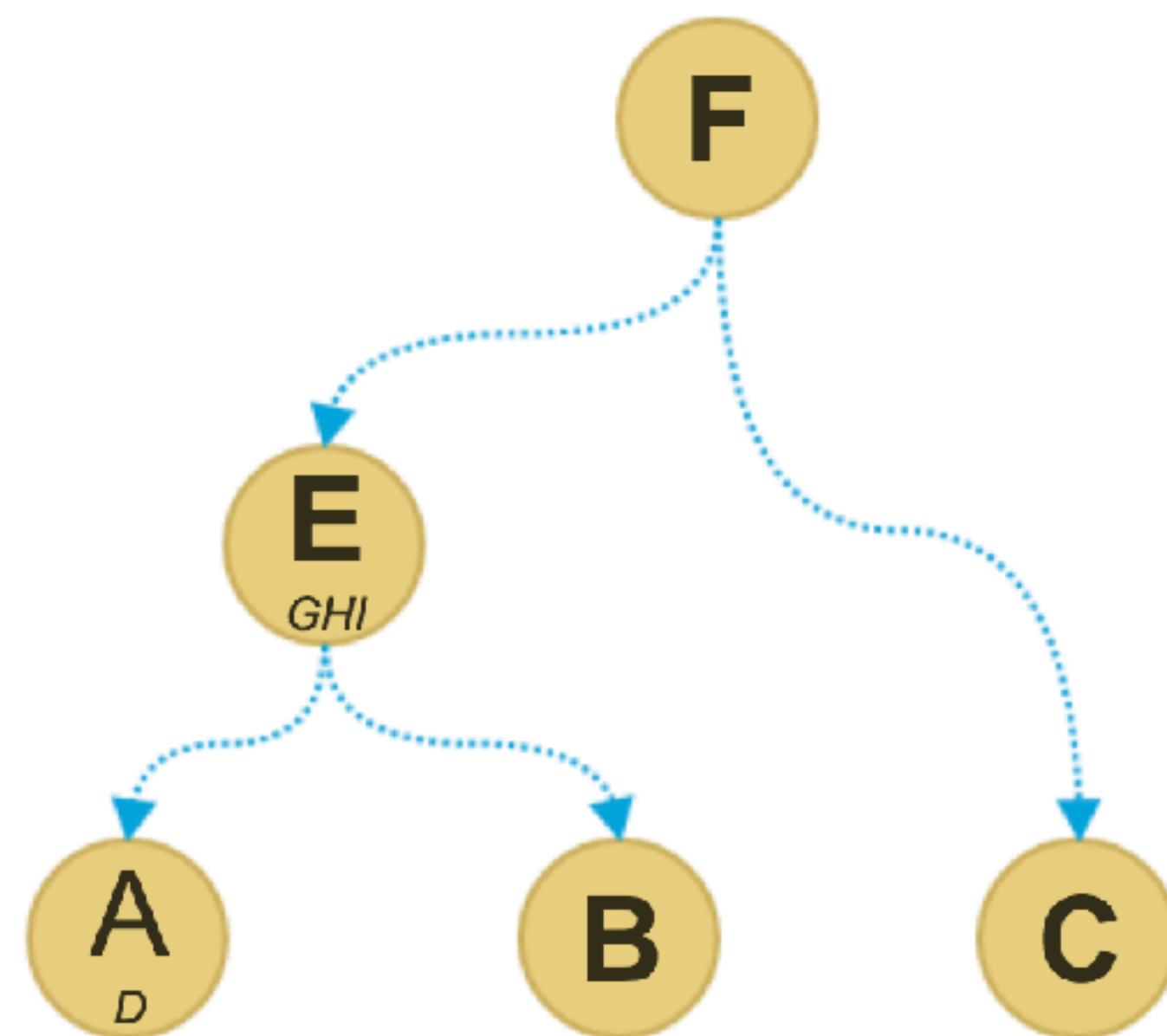
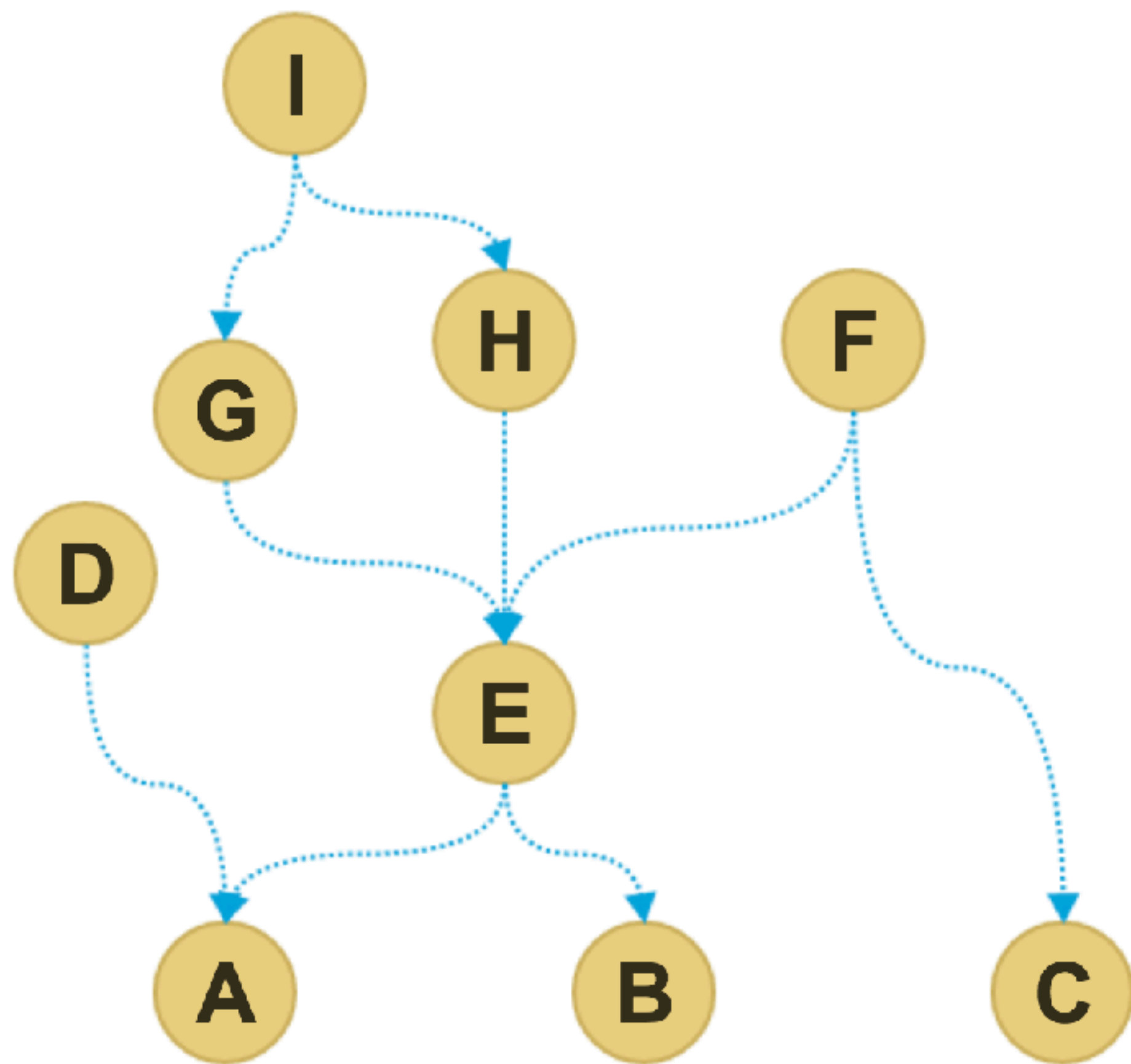
framework + business包



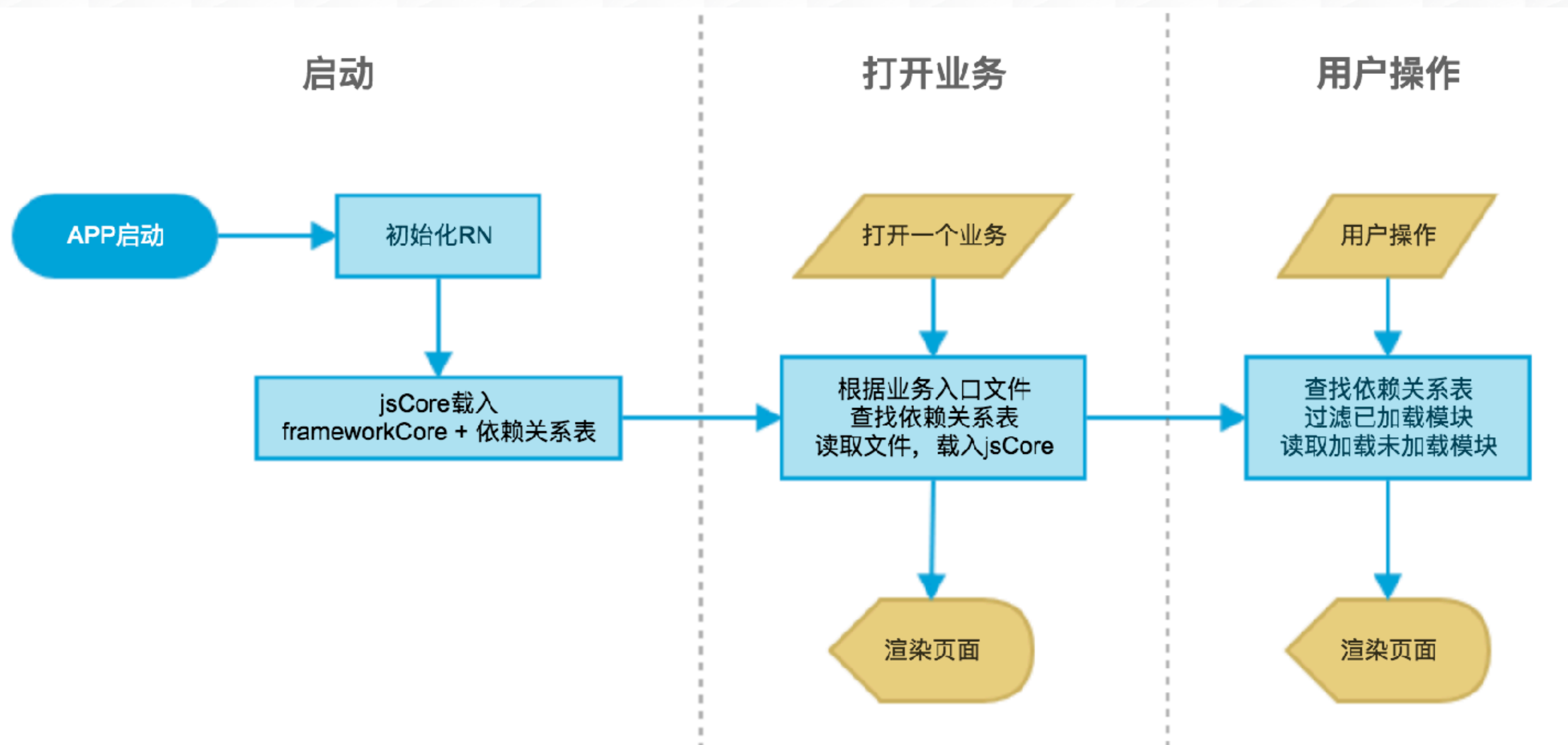
frameworkCore + 依赖关系表 + 内聚包



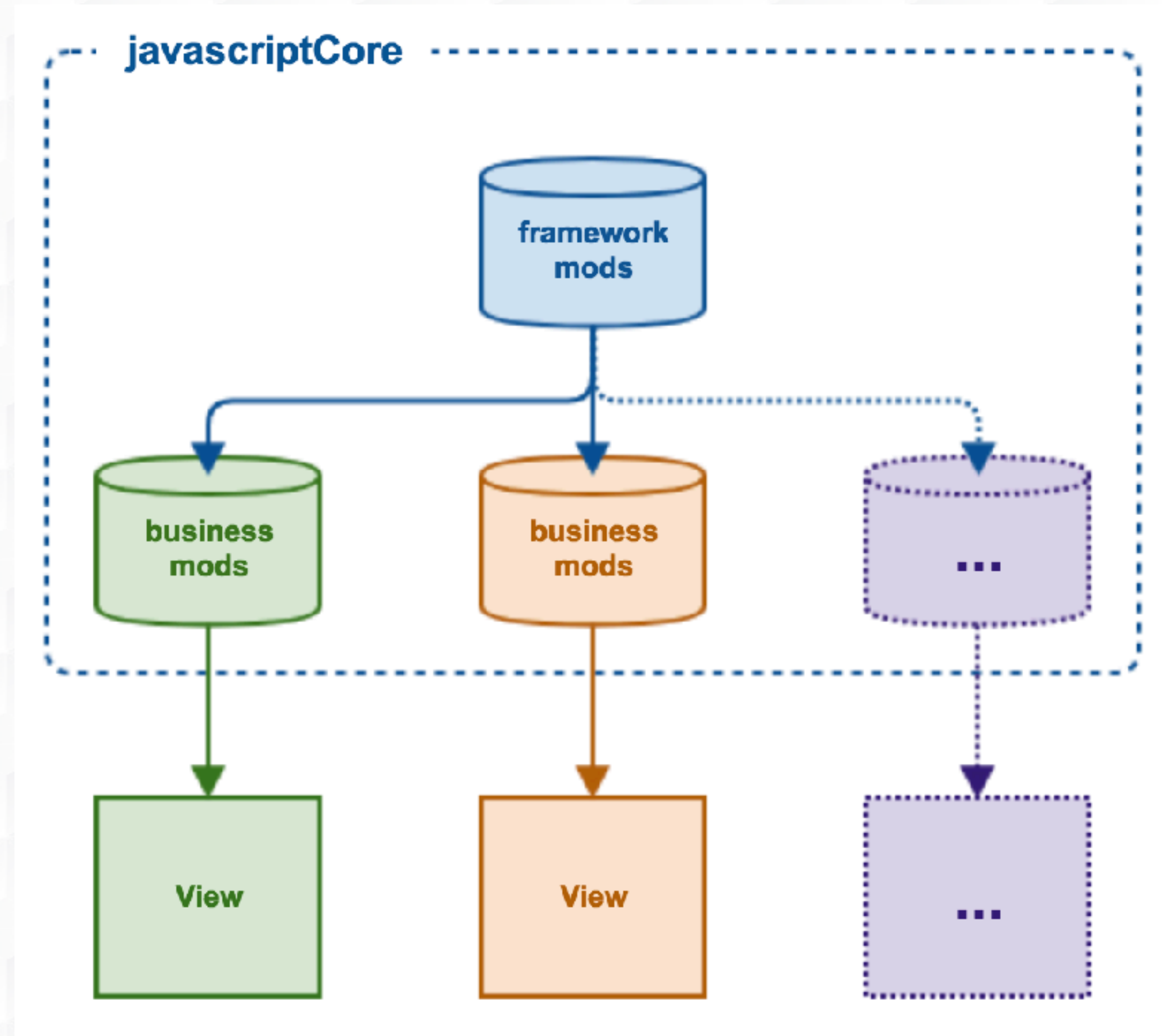
内聚包



加载



模块管理



框架共享

业务隔离

避免不同业务单例模块冲突

方便按业务卸载模块

运行出错，无缝降级



为什么需要降级?

- ❖ 提高框架上线初期可用性
- ❖ RN某些错误会造成功能无法使用
- ❖ WEB容错性好
- ❖ RN框架与WEB框架同时出错概率较小



如何降级

- ❖ 编译出hybrid代码
- ❖ 打通RN与hybrid的数据存储模块 (Storage + Cache)
- ❖ RN将路由信息、状态数据存入存储模块中
- ❖ 捕捉RN报错，切换到WebView载入hybrid对应页面
- ❖ hybrid读取存储模块中的数据，恢复页面



谢谢

