
HTML5 DREAMWORKS
IT大咖说
知识分享平台
前端开发配置化方案
徐辛承
2017.05

SELF INTRODUCTION

徐辛承

百度外卖 高级前端工程师

- 2012年毕业于**北京航空航天大学**（本科 -软件工程）
- 2012年3月加入**人人网**人人小站
- 2013年10月加入**百度**网址导航事业部
- 2016年4月加入百度外卖**用户产品部**

CONTENT

PART 1

内部平台开发中的痛点

内部平台是什么？

举个栗子

服务端存储数据

内部平台

商户管理

商品管理

营销活动

运营资源

用户增长

数据统计

……

服务端存储数据

重复工作多！

以banner配置为例

页面

- 列表页（所有banner配置的列表）
- 新建页（新建某banner配置）
- 详情页（展示某banner配置）
- 日志页（记录操作日志）
- ……

以banner配置为例

功能

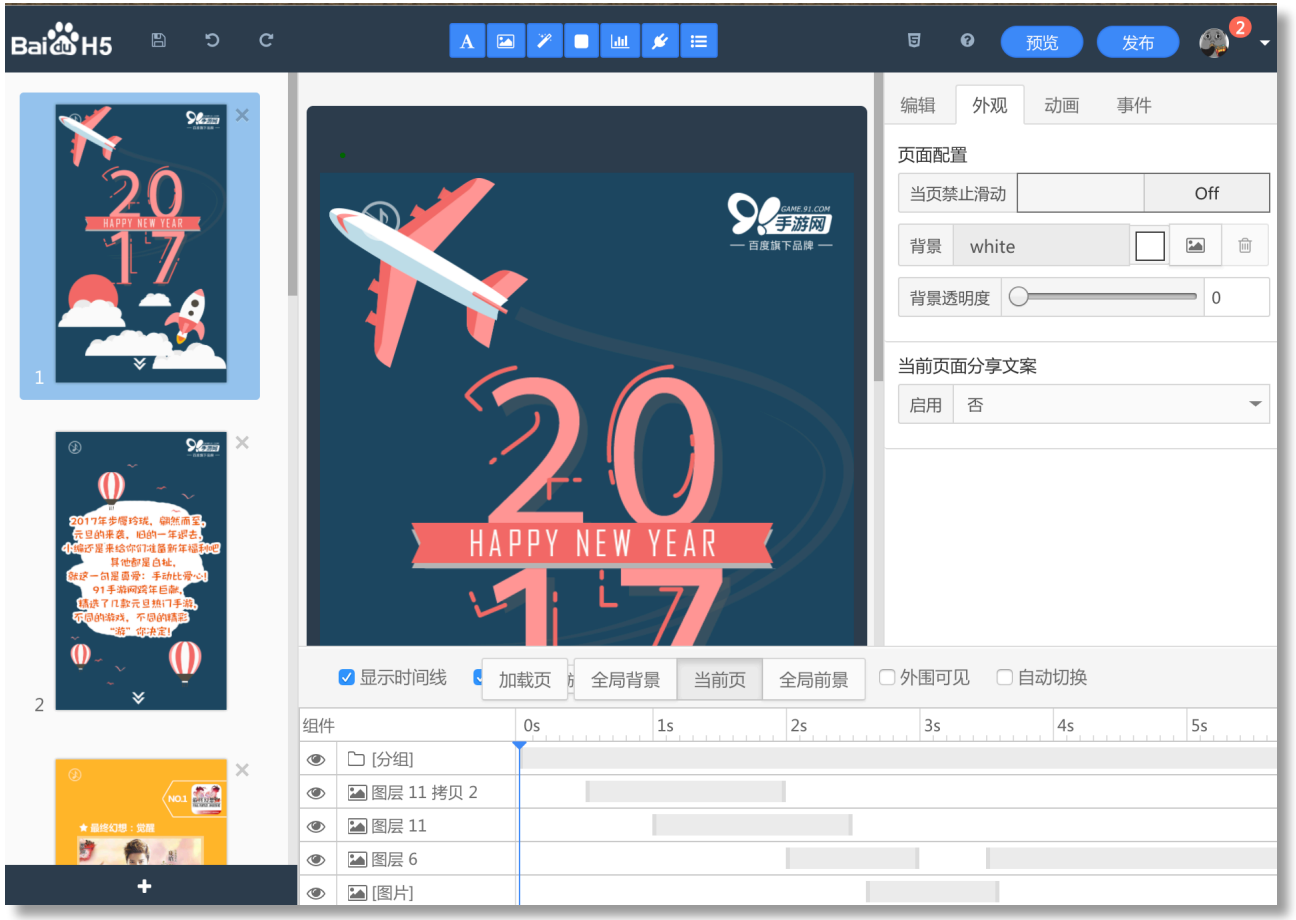
- 表单（输入框、单选、多选等）
- 表格（数据展示、翻页等）
- 各类按钮
- 弹框（对话框、提示框等）
- ……

我们发现……

- 页面功能相似度高，重复工作较多
- 基础页面搭建主要靠复制粘贴
- 多人开发，代码风格统一困难

需要平台来解决这些问题

快速生成页面方案



百度H5

优点：

- 拖拽+设置生成页面
- 组件为最小单位，组件库丰富
- 动画功能

缺点：

- 内部平台的交互无法支持
- 组件不能拓展

我们对平台的要求

简单

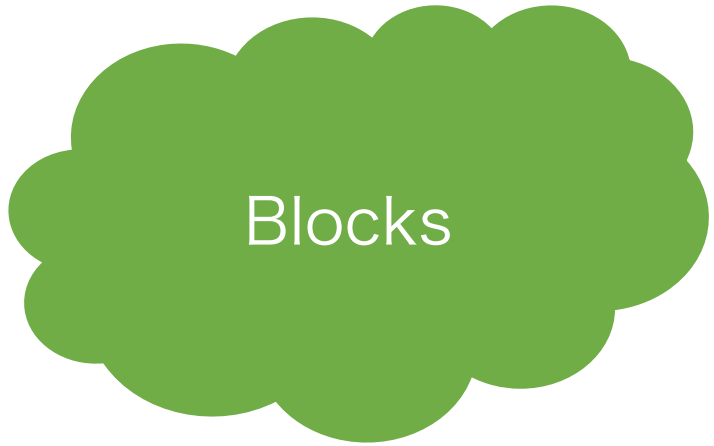
- 所有角色
- 拖拽界面
- 易于使用

100%

- 丰富的组件库
- 交互的配置
- 自定义拓展脚本
- 配置化生成页面

可维护性

- 熟悉的技术栈
- 可扩展性强
- 模块拆分详细



PART 2

页面配置平台Blocks

什么是Blocks

一套通过**拖拽 + 配置**生成页面的系统

- 以**组件**为最小单位
- **拖拽**形式的页面画布
- 支持组件添加和**拓展**
- 支持**自定义扩展脚本**
- 基于 vue2.0 开发

Blocks架构图

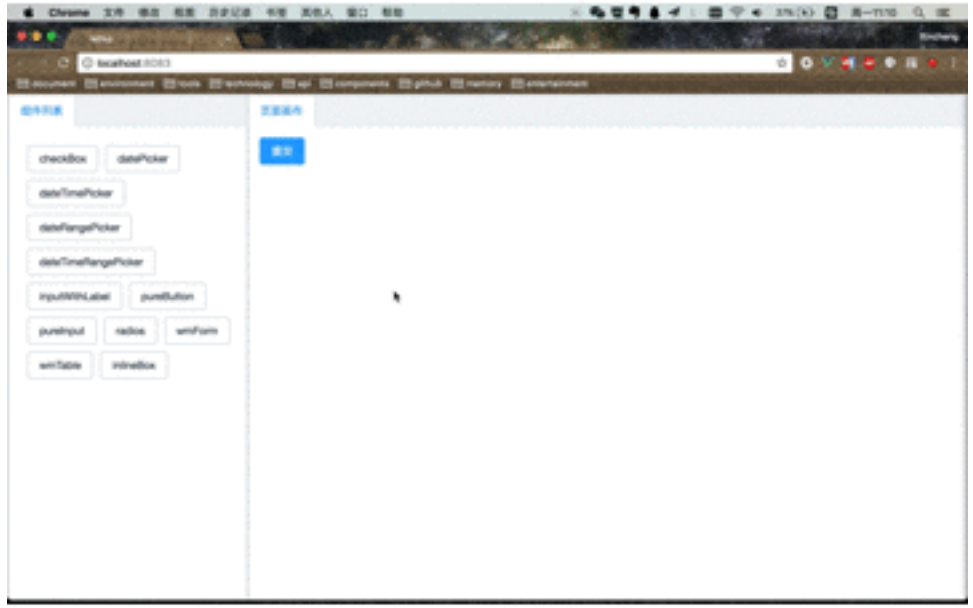

页面配置模块
组件列表
组件画布
组件设置表单
脚本配置模块
脚本配置
自定义扩展脚本
页面渲染引擎
配置文件解析
渲染DOM
加载脚本

页面配置模块

通过**拖拽 + 设置**的方式生成配置文件

- 组件列表
- 页面画布
- 设置组件属性
- 输出为config.js
- 同时会在mapConfig.js里预留

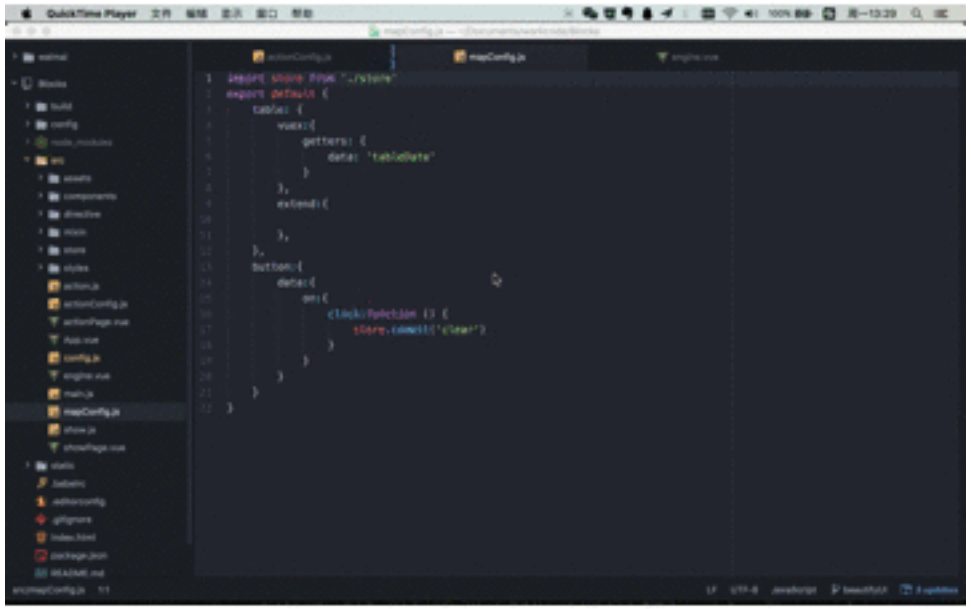
  事件钩子



脚本配置模块

在 JSON 文件里开发扩展脚本

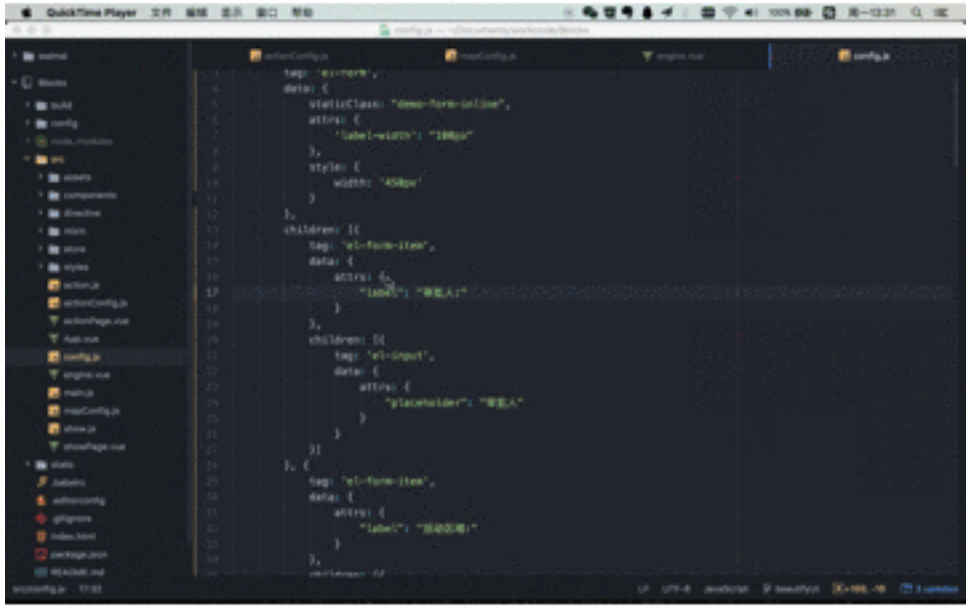
- 组件事件的拓展
- 页面整体交互开发
- 输出为mapConfig.js



页面渲染引擎

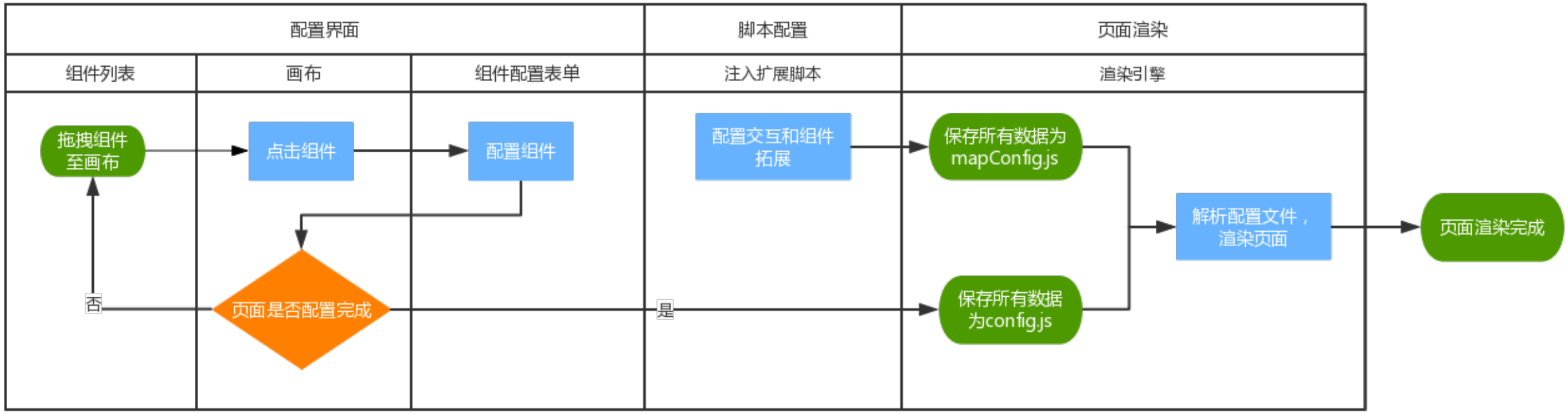
通过 JSON 配置文件生成页面

- 解析组件的配置和层级关系
- 解析配置文件里的自定义扩展脚本
- 渲染页面



核心

整体使用流程


配置界面
脚本配置
页面渲染
组件列表
画布
组件配置表单
注入扩展脚本
渲染引擎
拖拽组件至画布
点击组件
配置组件
配置交互和组件拓展
保存所有数据为mapConfig.js
解析配置文件，渲染页面
页面渲染完成
否
页面是否配置完成
是
保存所有数据为config.js

PART 3

平台核心设计

核心思想

JSON Web Page

核心思想

Config.js

Dom结构

MapConfig.js

页面交互

Web Page

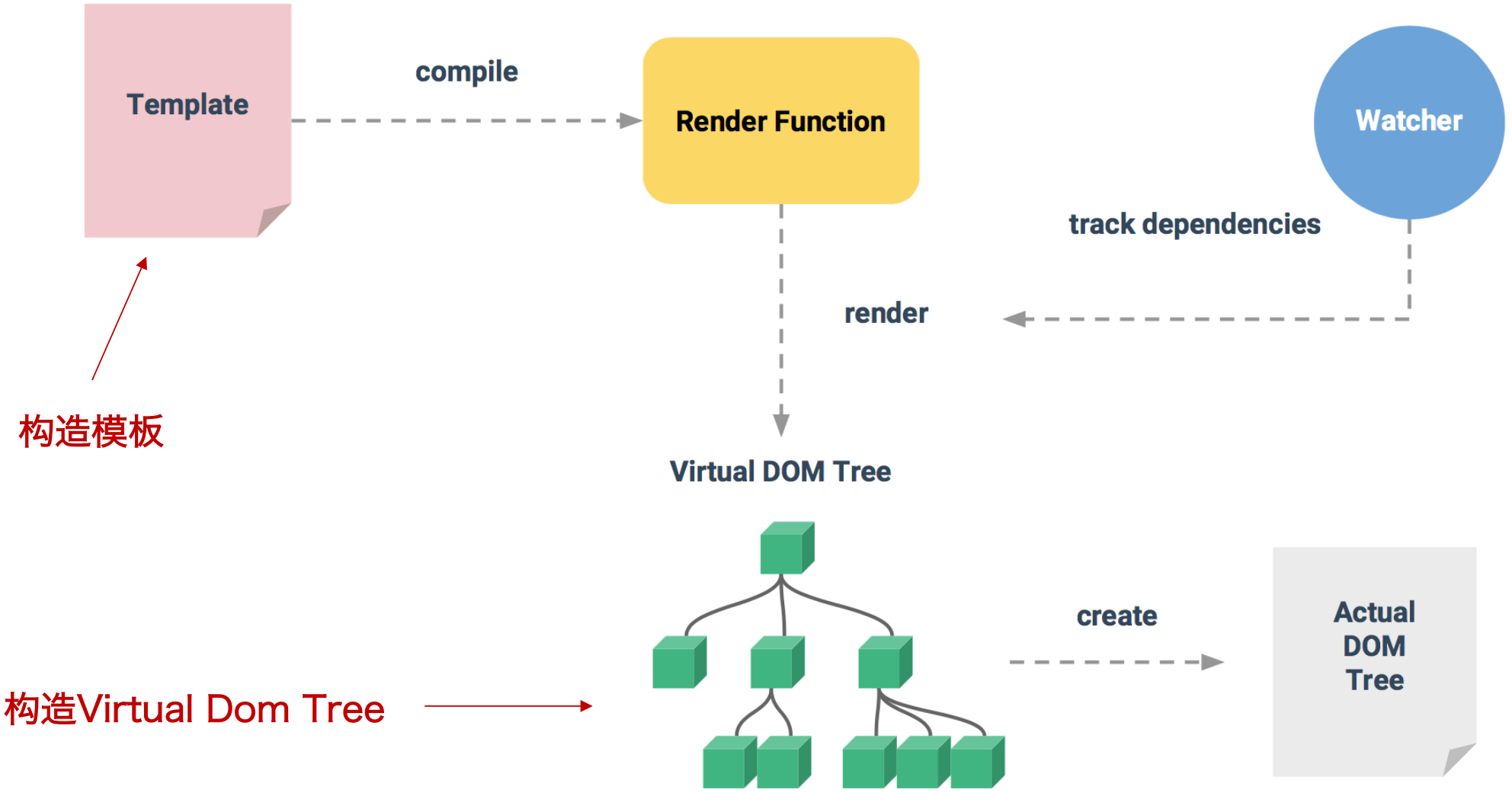
核心思想

Config.js

MapConfig.js

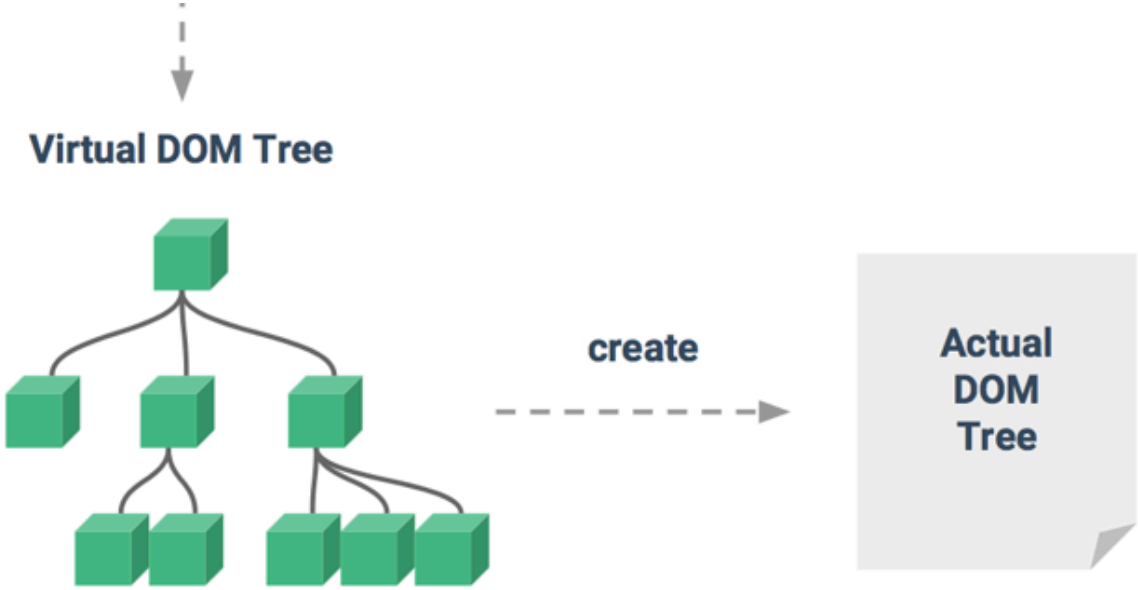
Web Page

页面渲染


Template
compile
Render Function
Watcher
track dependencies
render
构造模板
Virtual DOM Tree
构造Virtual Dom Tree
create
Actual
DOM
Tree

为什么这么做

- Virtual Dom Tree结构为object
- 基于Virtual Dom的实现，拓展性强



每一个节点

Tag

- 节点名称

Data

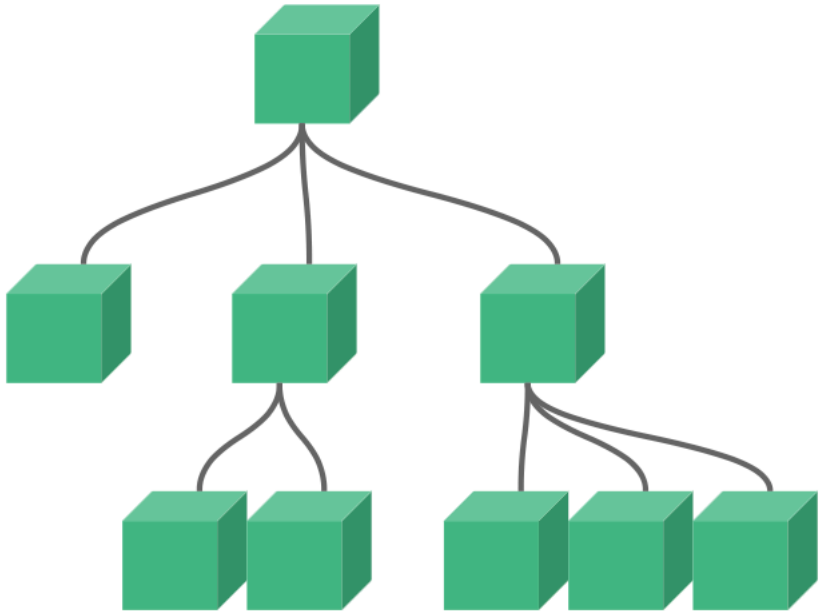
- 节点属性等

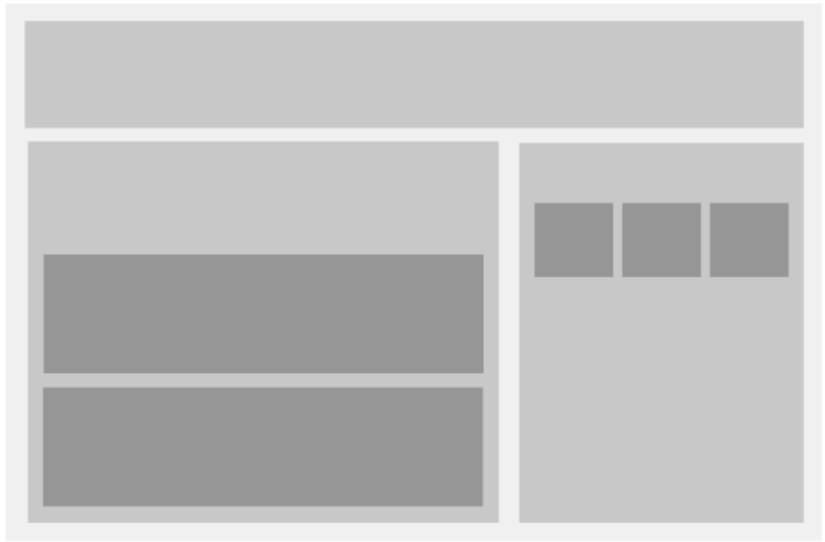
Children

- 该节点的所有子节点

```
tag: 'el-form-item',
data: {
    attrs: {
        "label": "审批人:"
    }
},
children: [{
    tag: 'el-input',
    data: {
        attrs: {
            "placeholder": "审批人"
        }
    }
}]
```

整个Virtual Dom Tree





深度优先遍历

核心思想

Config.js

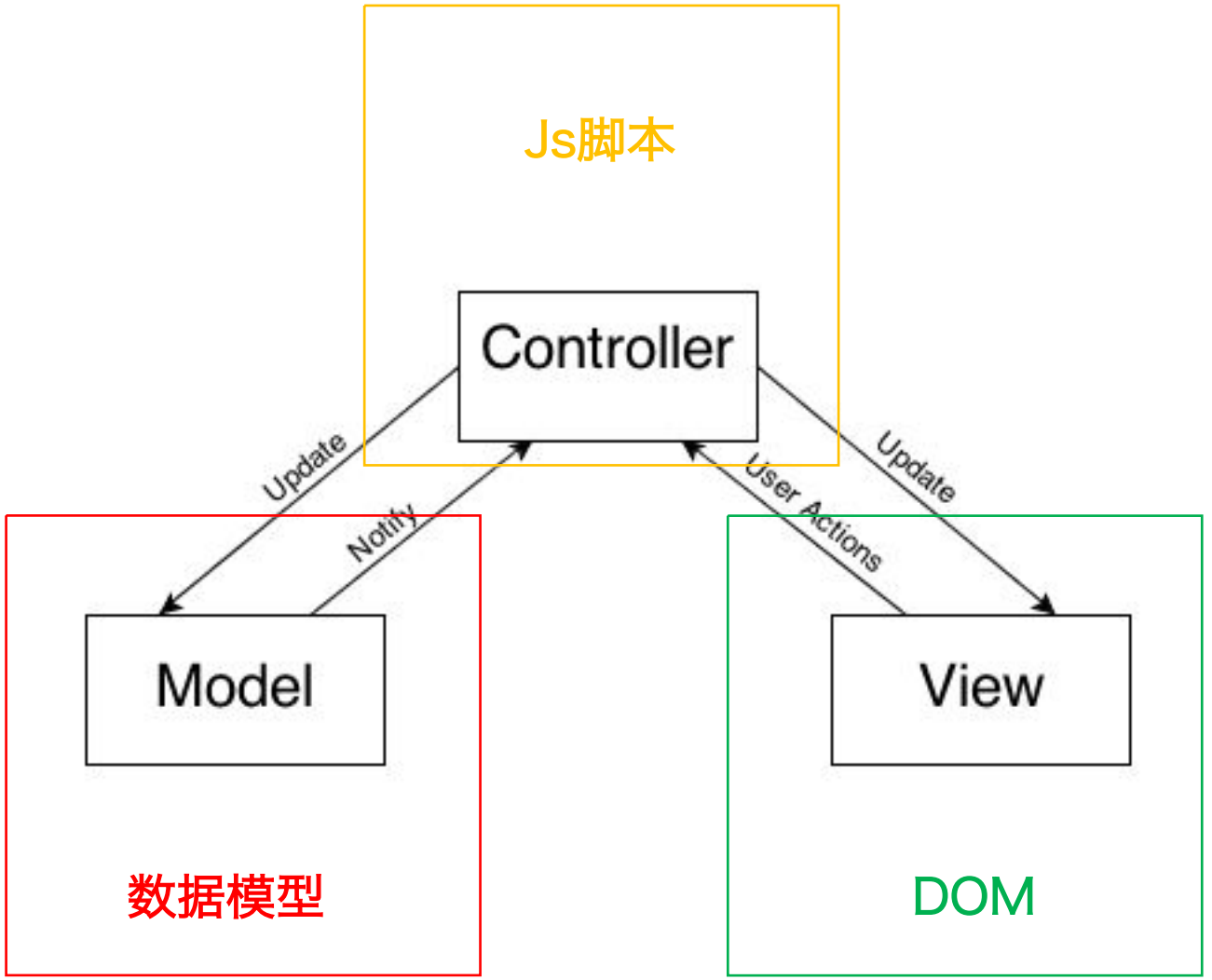
MapConfig.js

Web Page

原来的开发方式

MVC

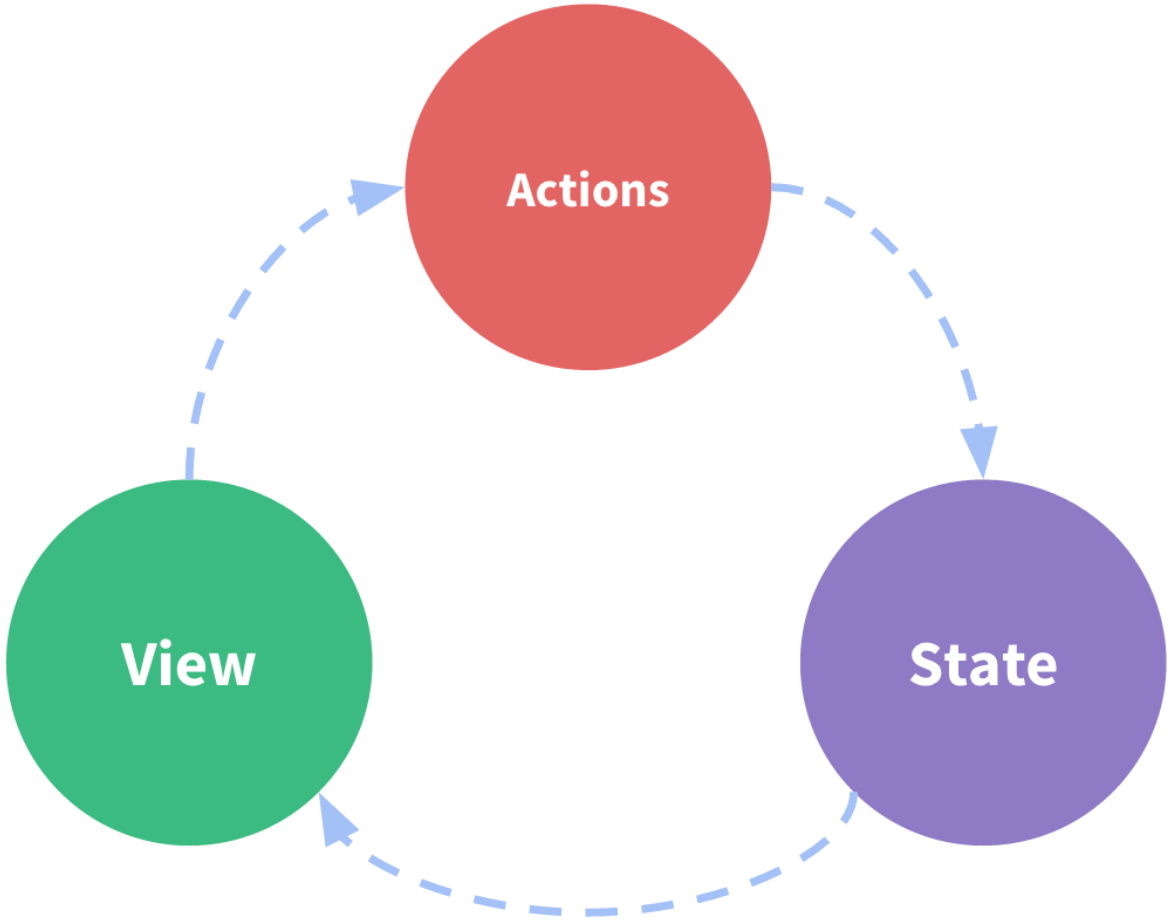
- Model <=> View 没有框架实现，需要通过Controller
- Model <=> View 间交互非常多
- 导致的问题：
  - Controller**开发效率低下**
  - 容易变得**臃肿和难以维护**



脚本配置

状态管理模式：

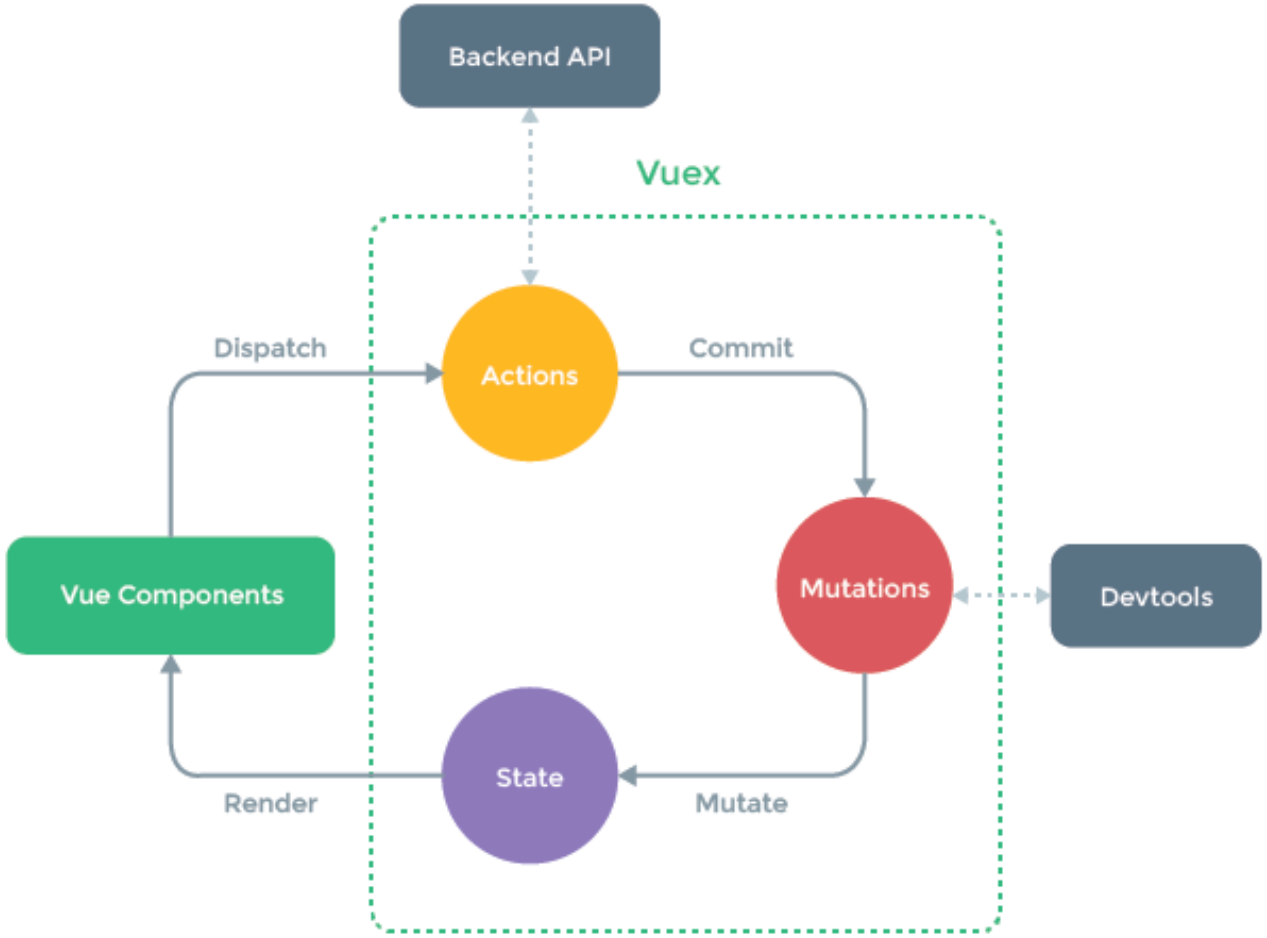
- state，驱动应用的数据源；
- view，以声明方式将state映射到视图；
- actions，响应在view上的用户输入导致的状态变化。



脚本配置

Vuex

- 专为 Vue.js 应用程序开发的**状态管理模式**
- 集中管理状态
- 以相应的规则保证状态以一种可预测的方式发生变化



Vuex

Vuex.store:

- State
- Getters
- Mutations
- Actions

```
vuex: { //vuex.store
    state: {
        count: 0
    },
    getters: {
        ...
    }
    mutations: {
        increment(state) {
            state.count++
        }
    },
    actions: {
        increment(context) {
            context.commit('increment')
        }
    }
},
```

脚本配置

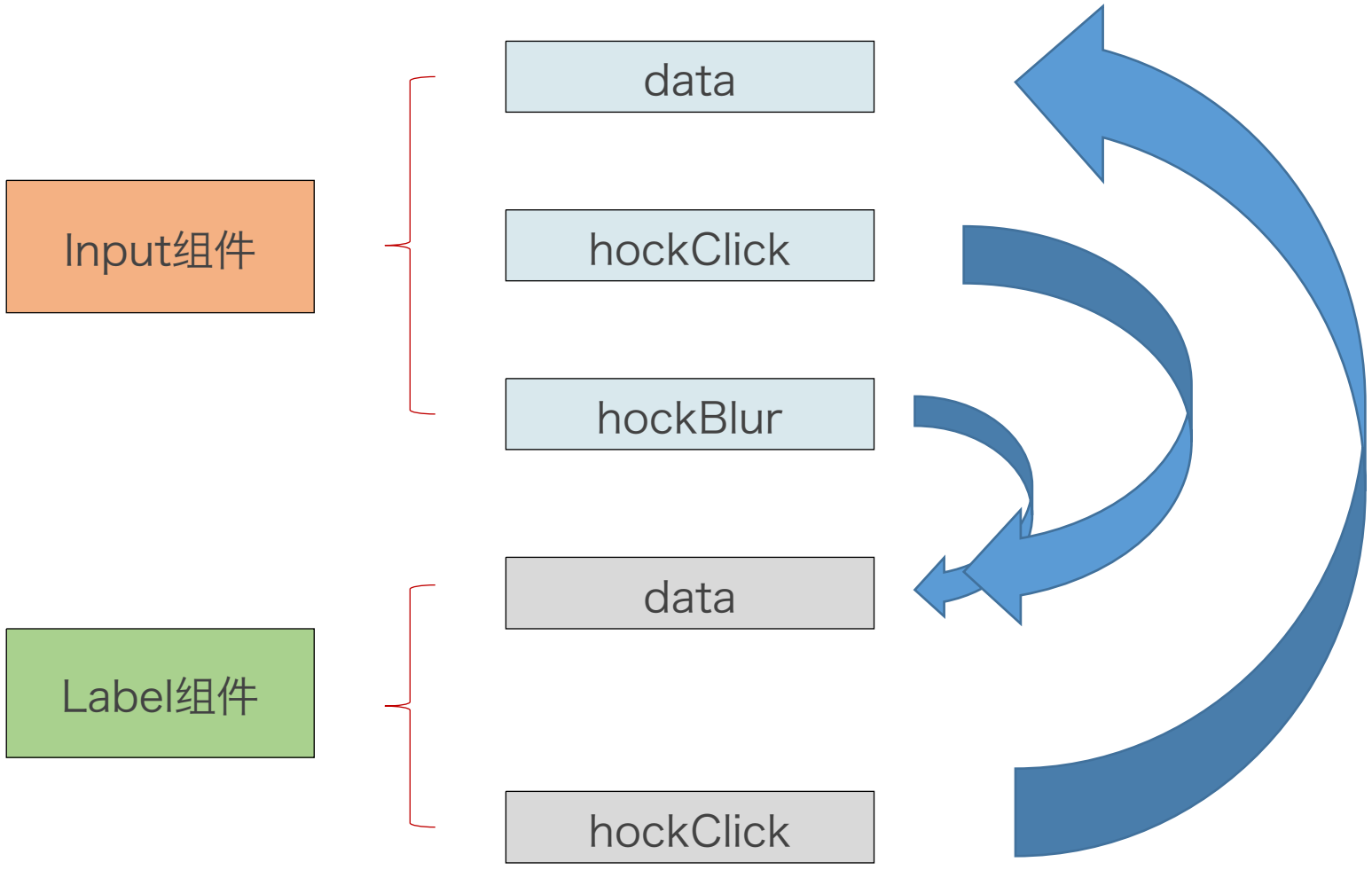
Vuex辅助函数:

- mapState
- mapGetters
- mapMutations
- mapActions

```
vuex: { //vuex.store
    state: {
        count: 0
    },
    getters: {
        ...
    }
    mutations: {
        increment(state) {
            state.count++
        }
    },
    actions: {
        increment(context) {
            context.commit('increment')
        }
    }
},
```

```
...mapState(vuexSetting.state ? vuexSetting.state : {}),
...mapGetters(vuexSetting.getters ? vuexSetting.getters : {})
...mapMutations(vuexSetting.mutations ? vuexSetting.mutations : {}),
...mapActions(vuexSetting.actions ? vuexSetting.actions : {})
```

事件钩子



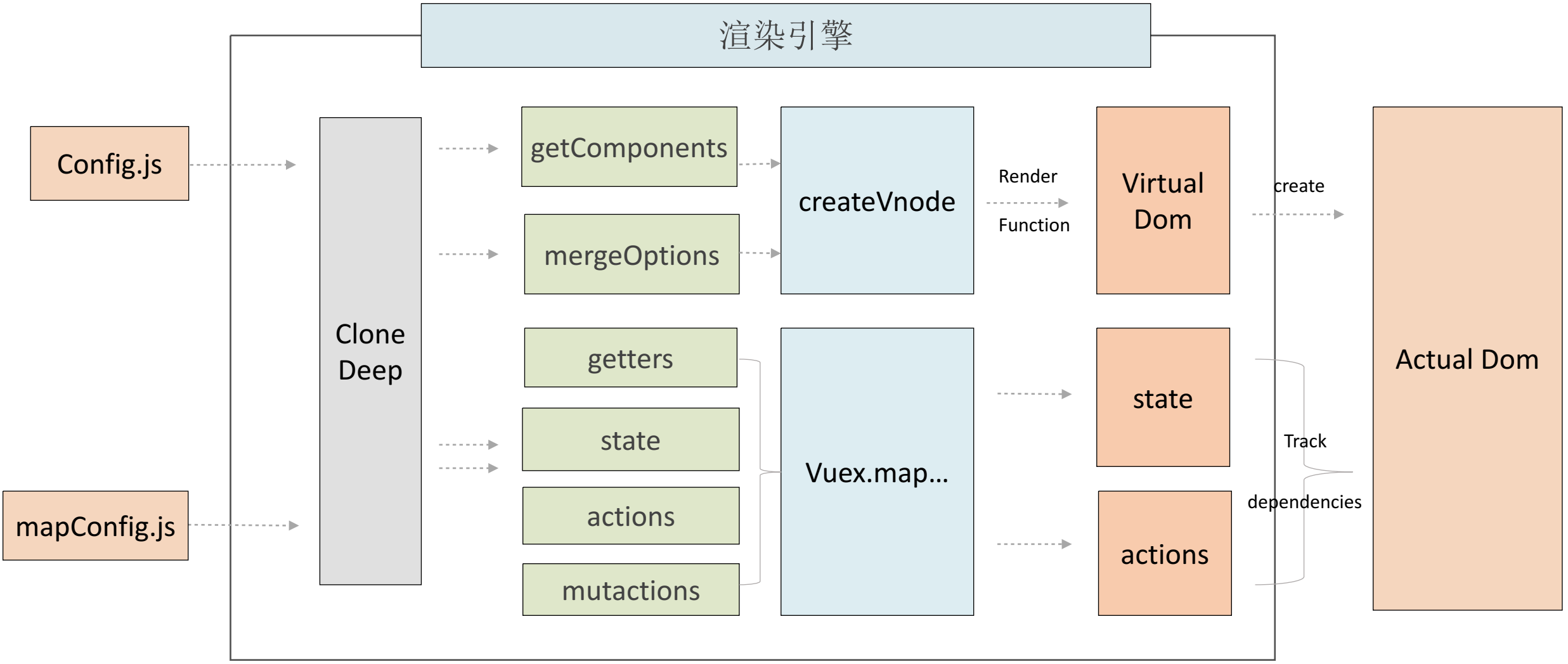
脚本配置

组件拓展

- MergeExtend

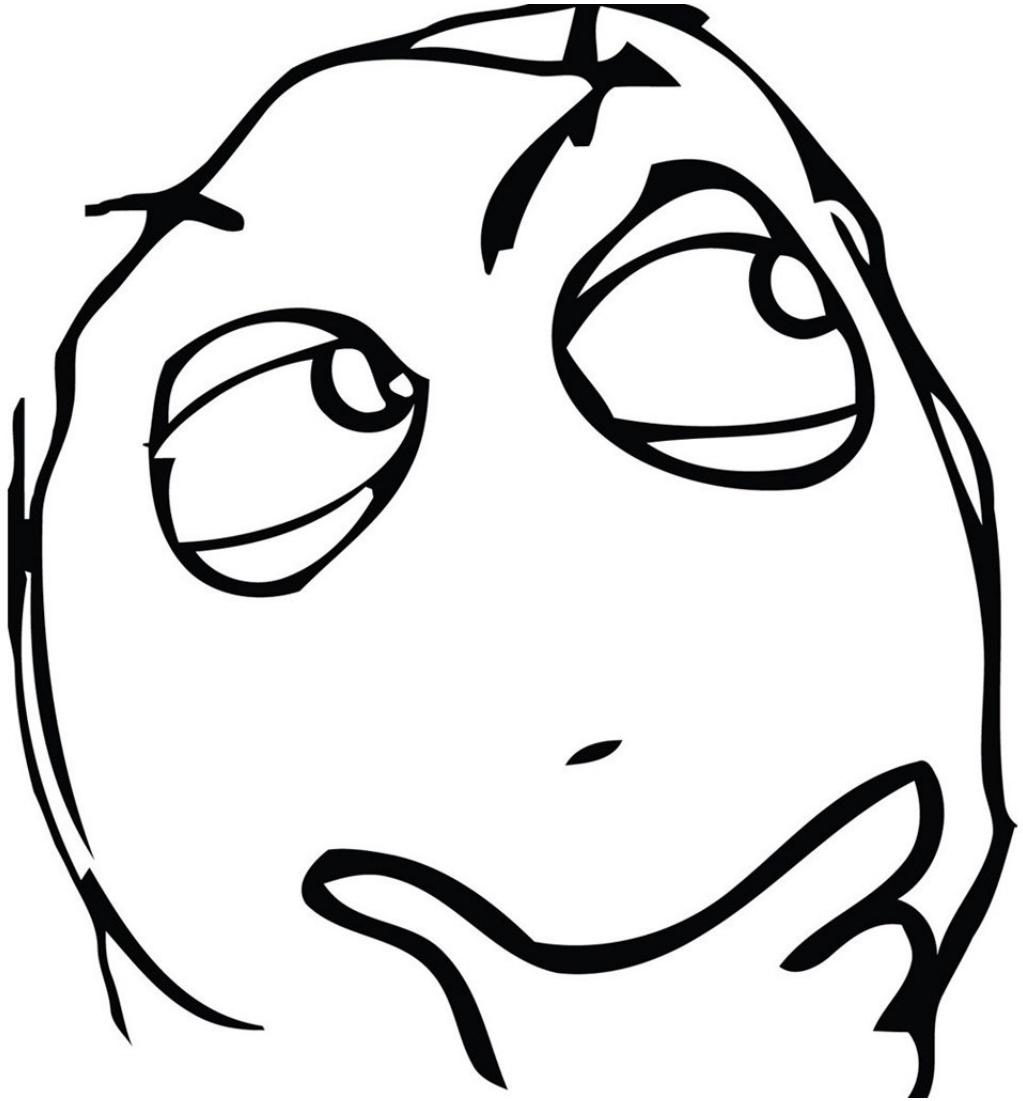
```
options = {
    computed: {
        ...extend.computed,
        ...mapState(vuexSetting.state ? vuexSetting.state : {}),
        ...mapGetters(vuexSetting.getters ? vuexSetting.getters : {})
    },
    methods: {
        ...extend.methods,
        ...mapMutations(vuexSetting.mutations ? vuexSetting.mutations : {}),
        ...mapActions(vuexSetting.actions ? vuexSetting.actions : {})
    }
};
```

渲染引擎架构图

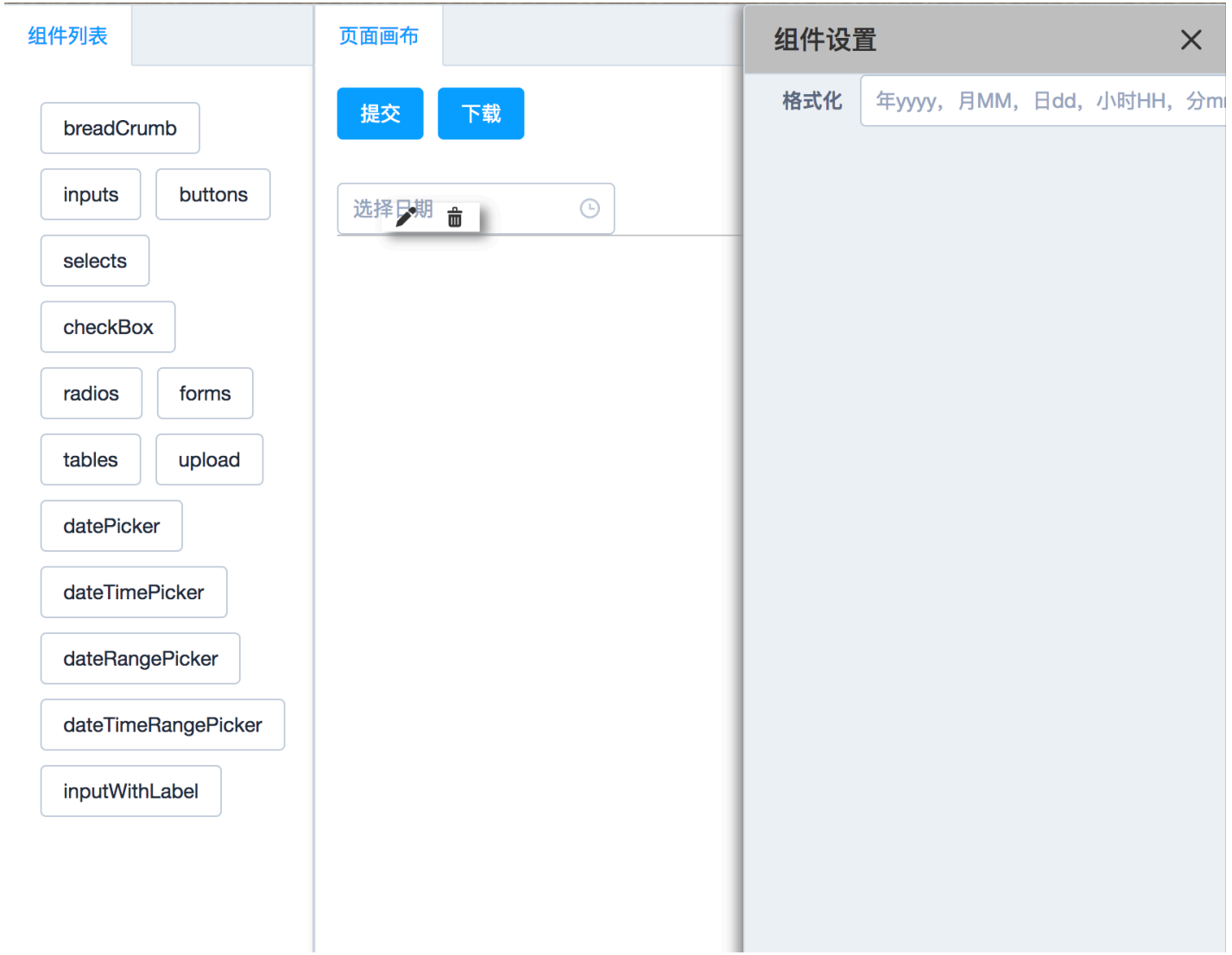

渲染引擎
Config.js
mapConfig.js
Clone Deep
getComponents
mergeOptions
getters
state
actions
mutactions
createVnode
Vuex.map...
Render
Function
Virtual Dom
state
actions
create
Track
dependencies
Actual Dom

如何生成这两份配置文件

- 页面配置模块
- 脚本配置模块



页面配置模块



组件列表

- 监听drag事件，在页面画布中渲染

页面画布

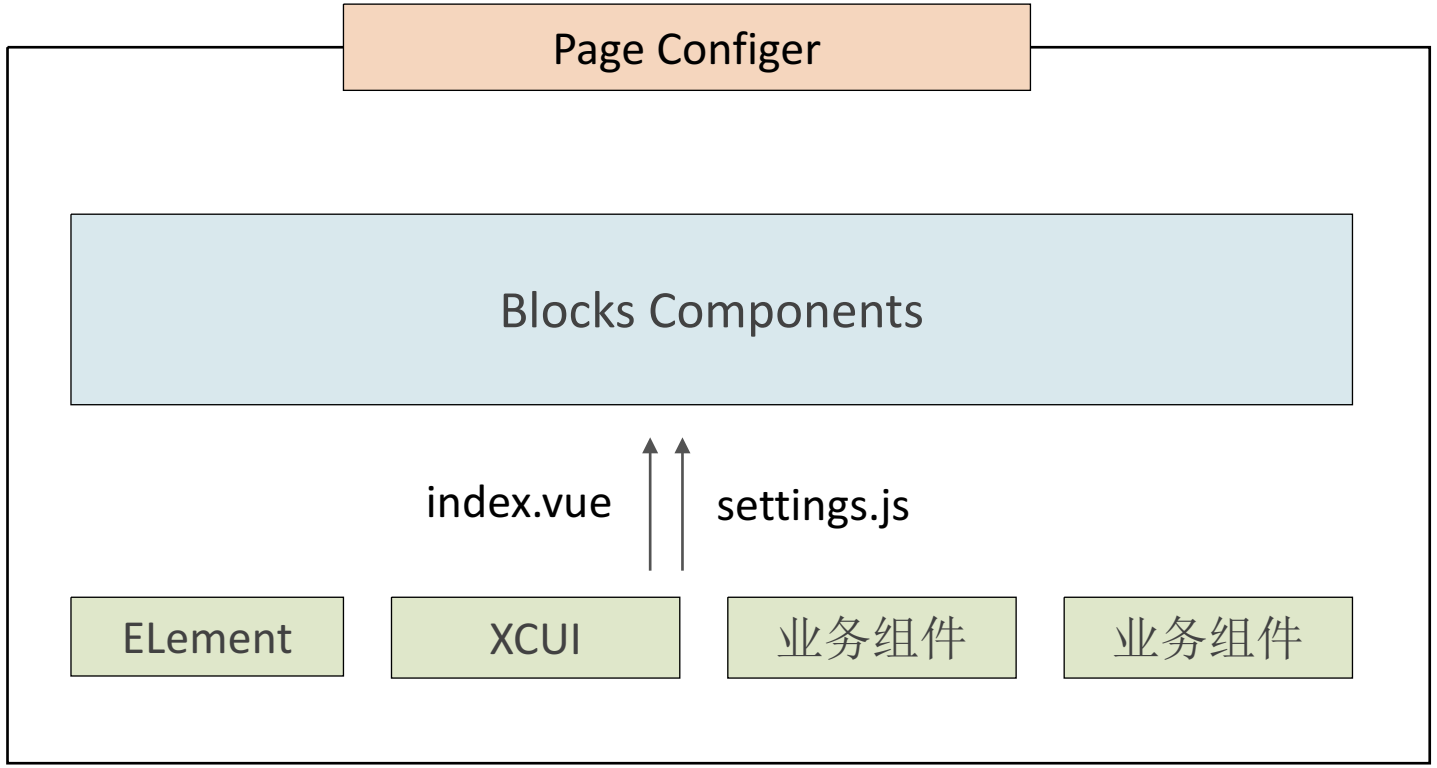
- 点击画布组件，组件设置加载对应的配置

页面画布

- 改变属性，改变Virtual Dom，渲染画布区域

组件的引入

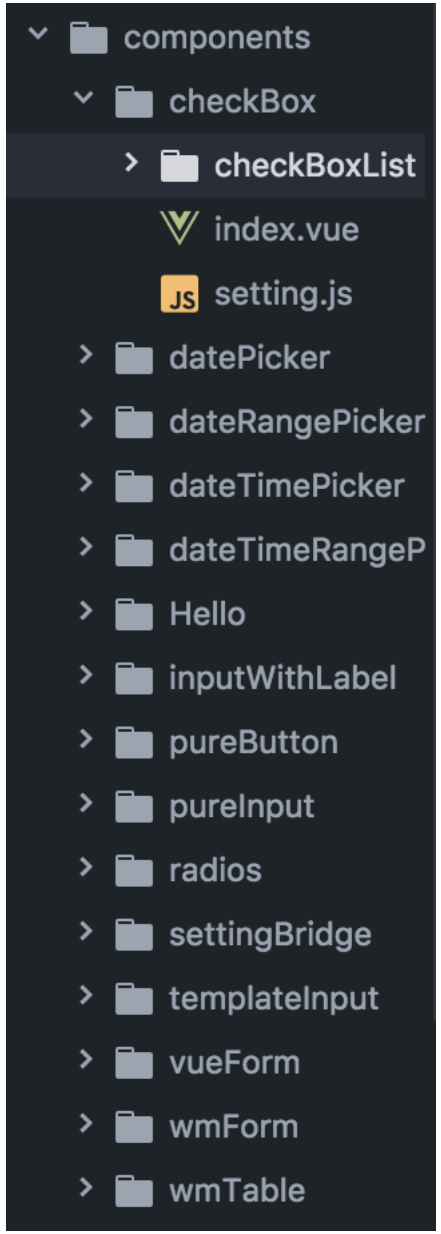
- 对通用组件库和业务组件做封装




```
tag: 'el-form-item',
data: {
    attrs: {
        "label": "审批人:"
    }
},
children: [{
    tag: 'el-input',
    data: {
        attrs: {
            "placeholder": "审批人"
        }
    }
}]
```

组件的引入

- index.vue
  - 渲染在画布和页面中
- setting.js
  - 组件设置表单

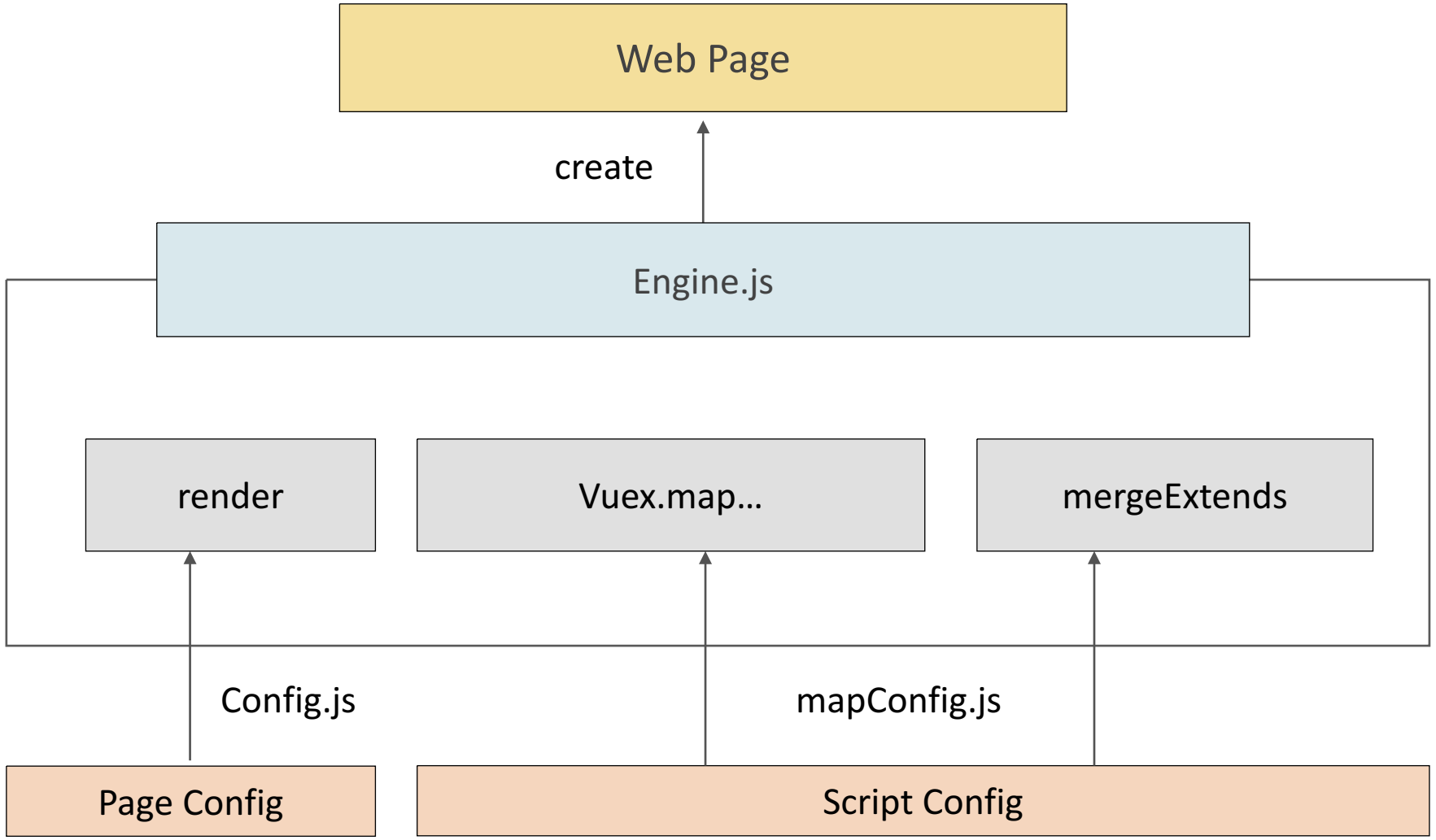


脚本配置模块

- Vuex.store的开发
- 组件内部事件扩展
  - mergeExtend
- 输出为配置文件mapConfig.js
- 未来会增加常规事件的可视化配置
- 固定组件属性，可以约束RD接口规范

```
table#1: {
    vuexSetting:{ // vuex.store的开发
        state: {
            count: 0
        },
        getters: {
            ...
        },
        mutations: {
            increment (state) {
                state.count++;
            }
        },
        actions:{
            ...
        }
    },
    extend:{ // 支持对table#1内部的事件进行拓展
        ...
    }
}
```

整体架构图


Web Page
create
Engine.js
render
Vuex.map...
mergeExtends
Config.js
mapConfig.js
Page Config
Script Config

PART 4

平台现状和规划

收益对比

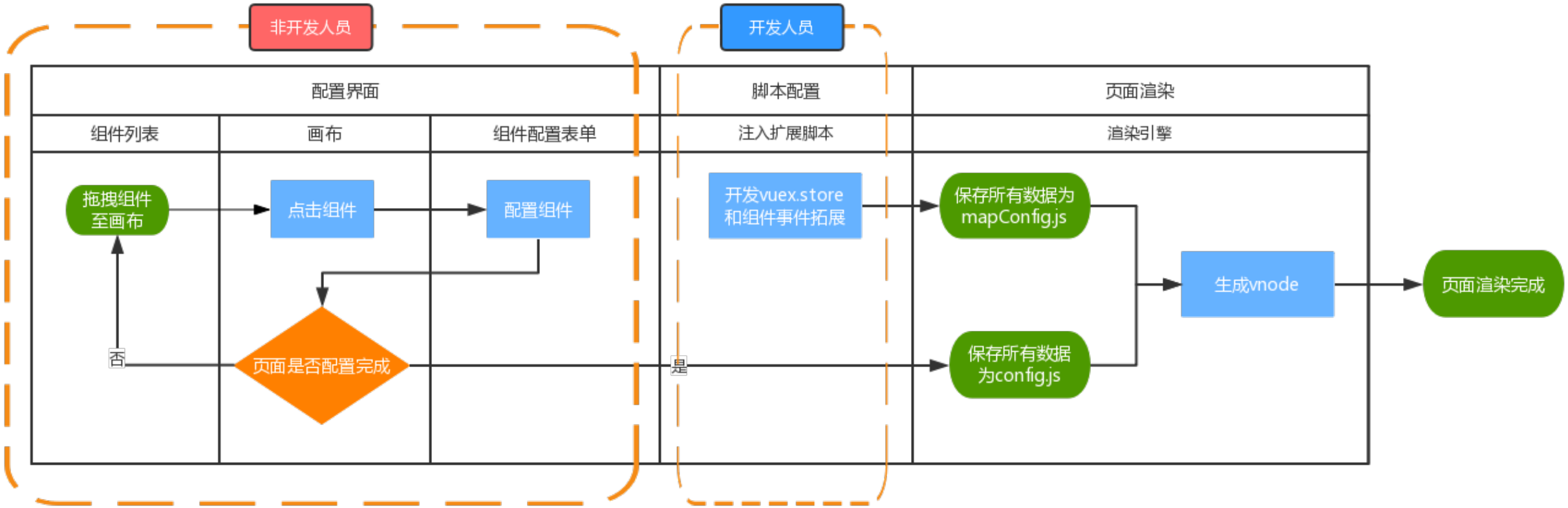
过去

- 接口无规范
- 每人代码风格不同，代码质量低
- 所有的工作研发完成

现在

- 约束接口规范
- 拖拽生成页面，代码质量高
- 部分工作可以外放，甚至0成本

系统人员使用情况


非开发人员
开发人员
配置界面
脚本配置
页面渲染
组件列表
画布
组件配置表单
注入扩展脚本
渲染引擎
拖拽组件至画布
点击组件
配置组件
开发vuex.store和组件事件拓展
保存所有数据为mapConfig.js
生成vnode
页面渲染完成
否
页面是否配置完成
是
保存所有数据为config.js

未来要做的工作

功能

- 丰富组件库
- 自定义组合件
- 可视化交互配置
- ……

易用性

- UI
- 系统交互
- ……

落地

- 老项目重构
- 新项目开发


THANK
YOU