

页面可视化搭建工具实践

陈韩杰 [@CatChen](#)

自我介绍

陈韩杰 [@CntChen](#)

负责过运营页面搭建工具的落地

页面可视化搭建框架 [page-pipeline](#) 作者

现为腾讯 [AllInyTeam](#) 成员

目录

1. 活动页面开发痛点(10%)
2. 可视化搭建工具前世今生(20%)
3. 可视化搭建工具技术要点(35%)
4. 理想的活动页面搭建工具(25%)
5. 开源搭建框架 *pipeline* (10%)

活动页面

锦瑟 -- 李商隐



锦瑟无端五十弦，一弦一柱思华年。
庄生晓梦迷蝴蝶，望帝春心托杜鹃。
沧海月明珠有泪，蓝田日暖玉生烟。
此情可待成追忆，只是当时已惘然。

锦瑟 -- 李商隐



锦瑟无端五十弦，一弦一柱思华年。
庄生晓梦迷蝴蝶，望帝春心托杜鹃。
沧海月明珠有泪，蓝田日暖玉生烟。
此情可待成追忆，只是当时已惘然。

送杜少府之任蜀州



城阙辅三秦，风烟望五津。
与君离别意，同是宦游人。
海内存知己，天涯若比邻。
无为在歧路，儿女共沾巾。

送杜少府之任蜀州 -- 王勃



城阙辅三秦，风烟望五津。
与君离别意，同是宦游人。
海内存知己，天涯若比邻。
无为在歧路，儿女共沾巾。

活动页面特点

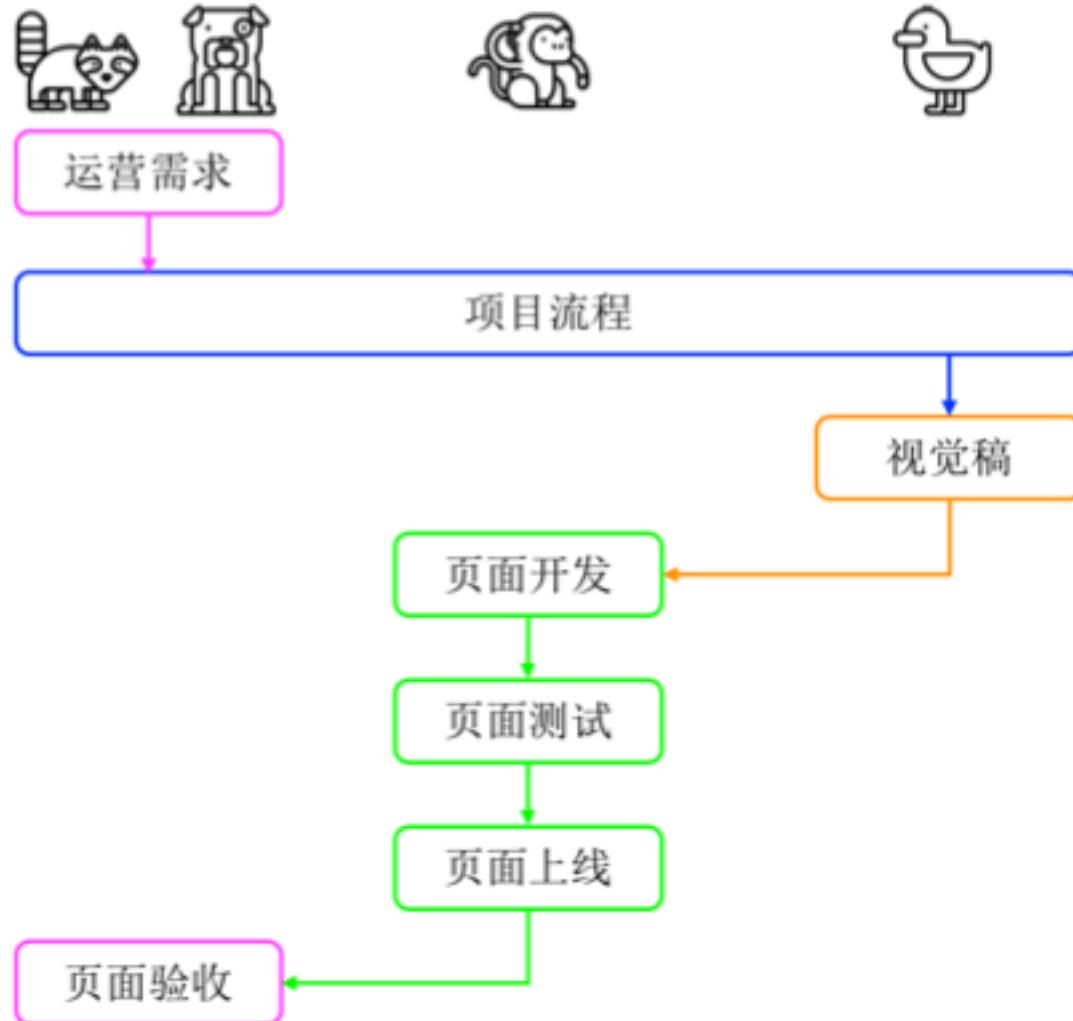
页面类似: 页面布局和业务逻辑较固定

需求高频: 每周甚至每天有多个这种需求

迭代快速: 开发时间短, 上线时间紧

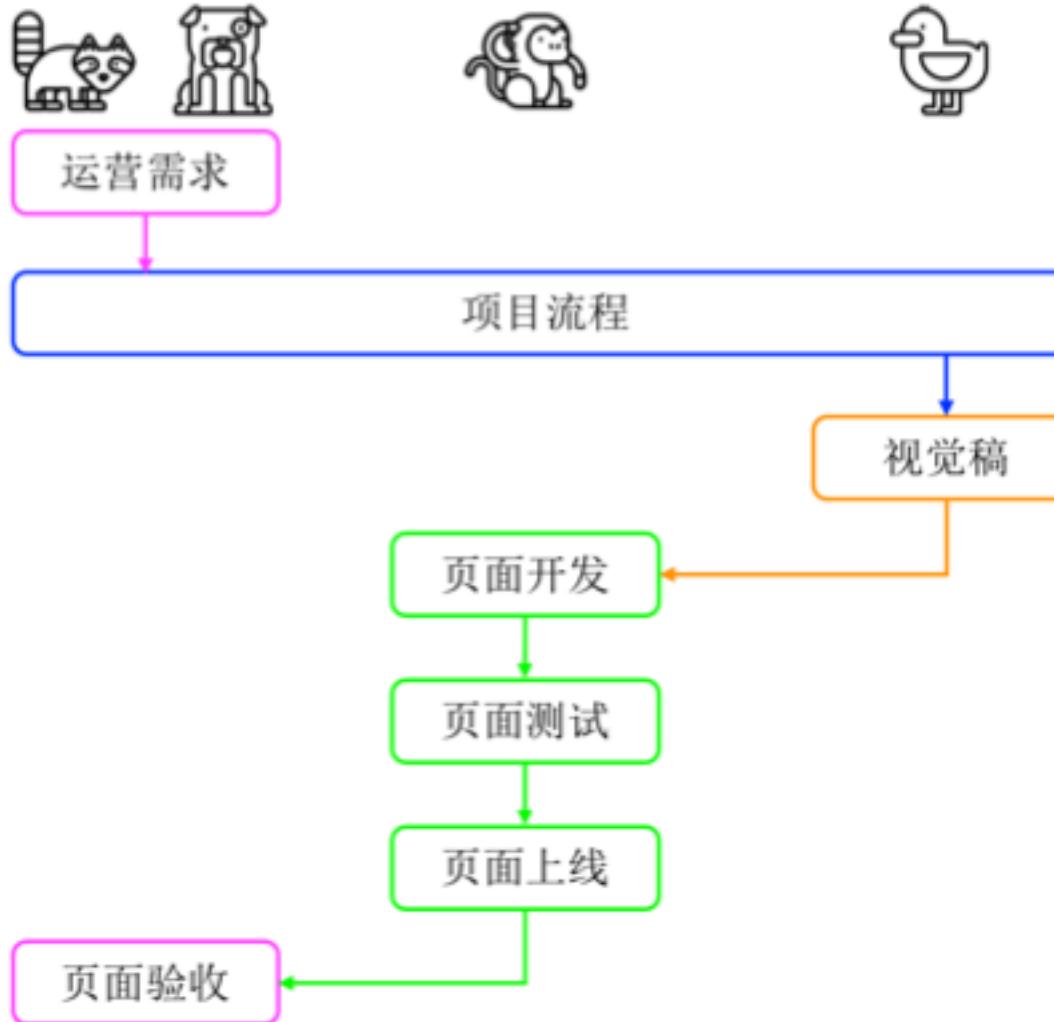
开发性价比低: 开发任务重复, 消耗时间和人力

常规开发流程



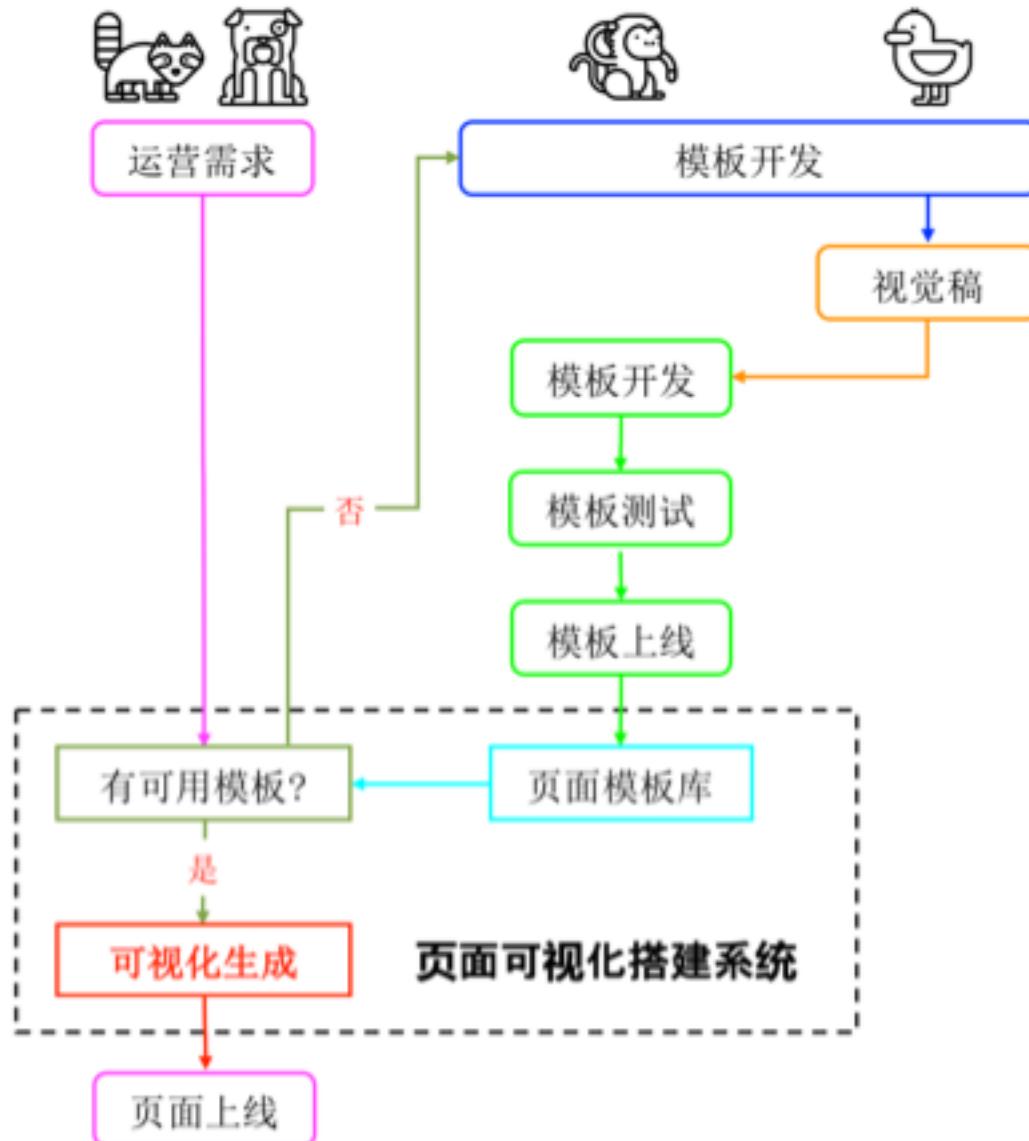
1. 运营/产品提出页面需求
2. 走项目流程进入开发环节
3. 开发根据设计稿完成页面开发
4. 测试进行页面测试
5. 走发布流程上线页面
6. 运营/产品进行页面验收

常规开发流程痛点



- 方多参与, 反复沟通, **串行流程**
- 页面上线周期长, 无法快速响应活动需求
- 人力陷入重复工作泥潭, 忙碌而低效

更优的流程



1. 运营/产品提出页面需求
2. 运营/产品在**页面可视化搭建系统**中选取合适的页面模板进行页面搭建
3. 页面自动化发布上线, 流程完结
4. 如果运营/产品没有找到合适的模板
5. 开发进行页面模板开发, 并将页面模板添加到**页面可视化搭建系统**
6. 运营/产品继续流程2

目录

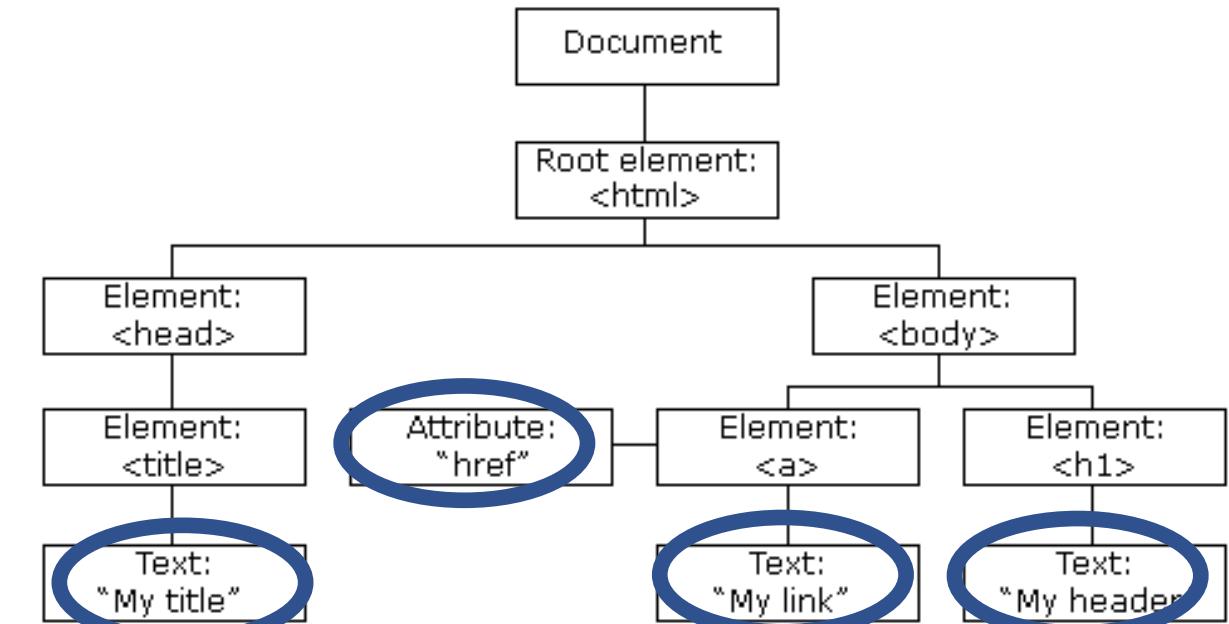
1. 活动页面开发痛点(10%)
2. 可视化搭建工具前世今生(20%)
3. 可视化搭建工具技术要点(35%)
4. 理想的活动页面搭建工具(25%)
5. 开源搭建框架 *pipeline* (10%)

页面是什么？

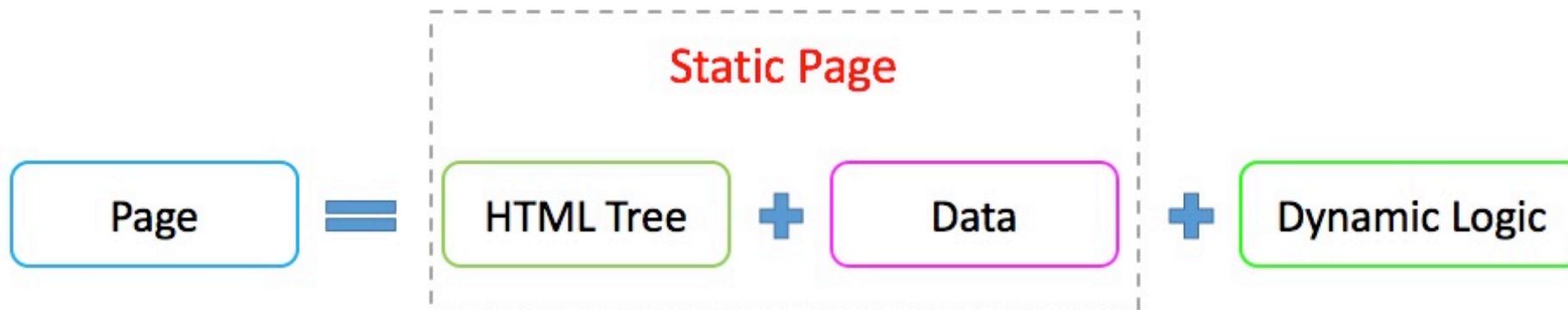
HTML ▾

```
<!DOCTYPE html>
<html>
<head>
  <title>My title</title>
</head>
<body>
  <h1>My Header</h1>
  <a href="#">My link</a>
</body>
</html>
```

My Header

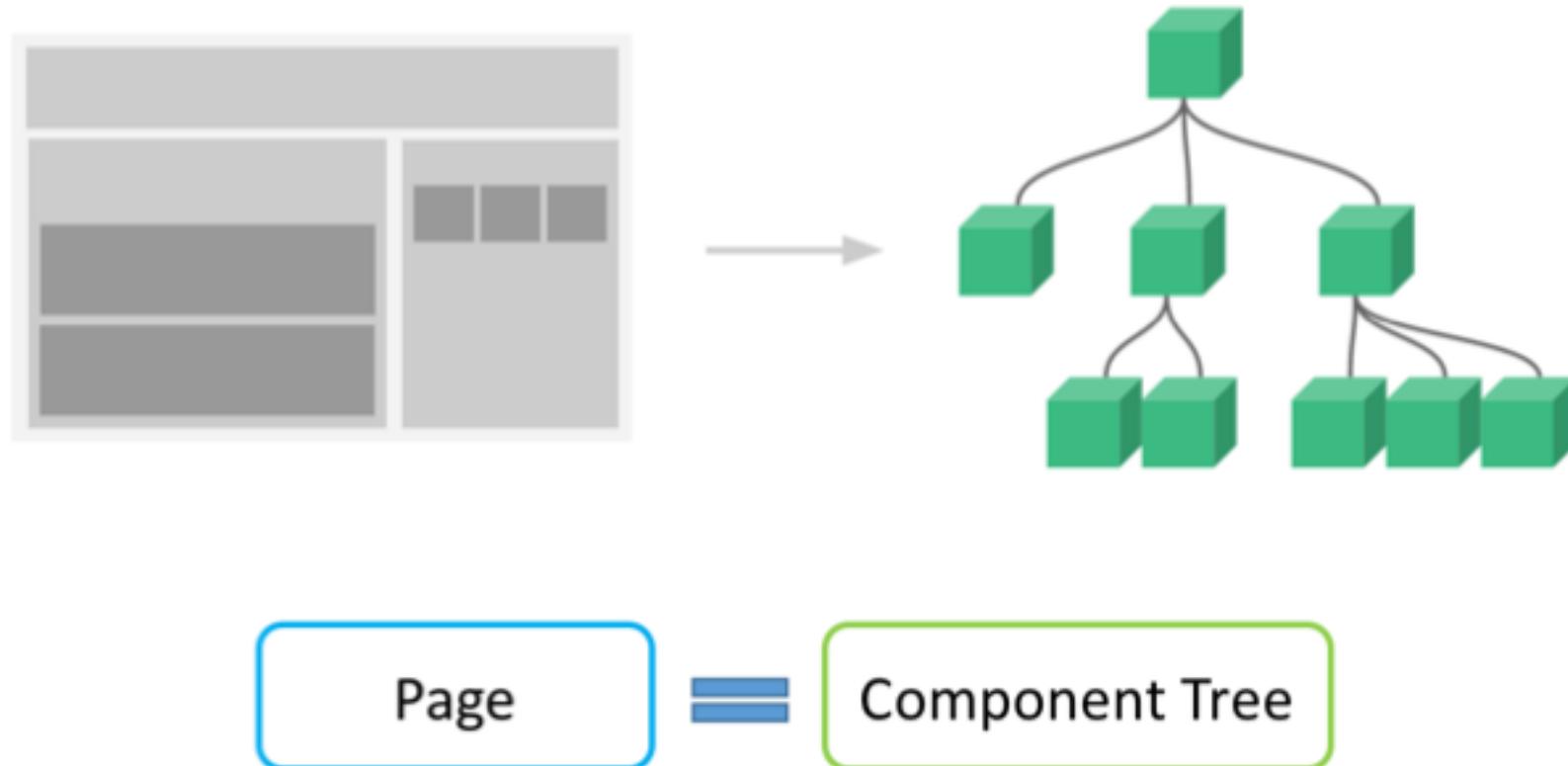
[My link](#)

静态页面和动态页面



页面组件化

页面组件化组合 HTML 元素, 实现了功能封装和可复用的页面组件



页面可视化搭建的定义

页面开发: 用编程工具(**IDE**)来编辑页面

页面可视化搭建: 用可视化交互的方式编辑页面

	编程开发页面	可视化搭建页面
技能要求	需要编程基础	可以没有编程基础
操作方式	在代码编辑器中编写代码	在可视化搭建工具中 拖拉/填表 等
使用人员	前端工程师	前端小白/ 运营和产品 /开发人员

页面可视化搭建, 是前端服务化的一种形式.

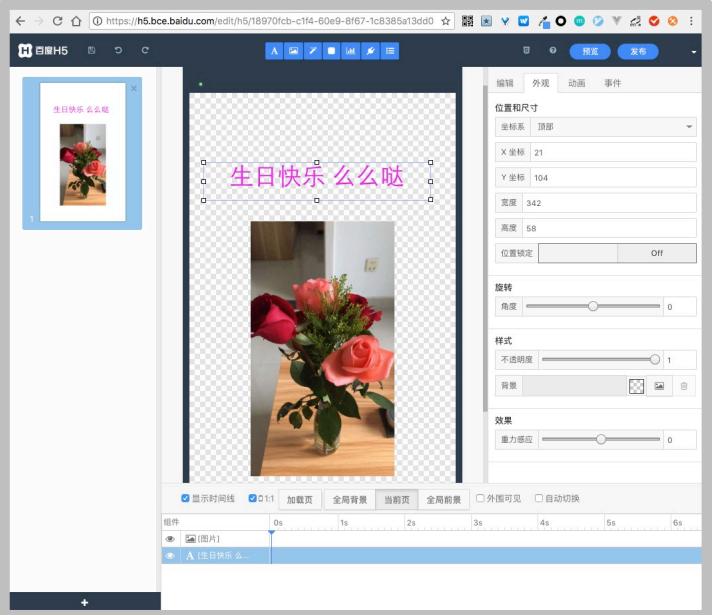
搭建工具区分维度

当我们讨论页面可视化搭建工具时，怎么进行描述和区分？

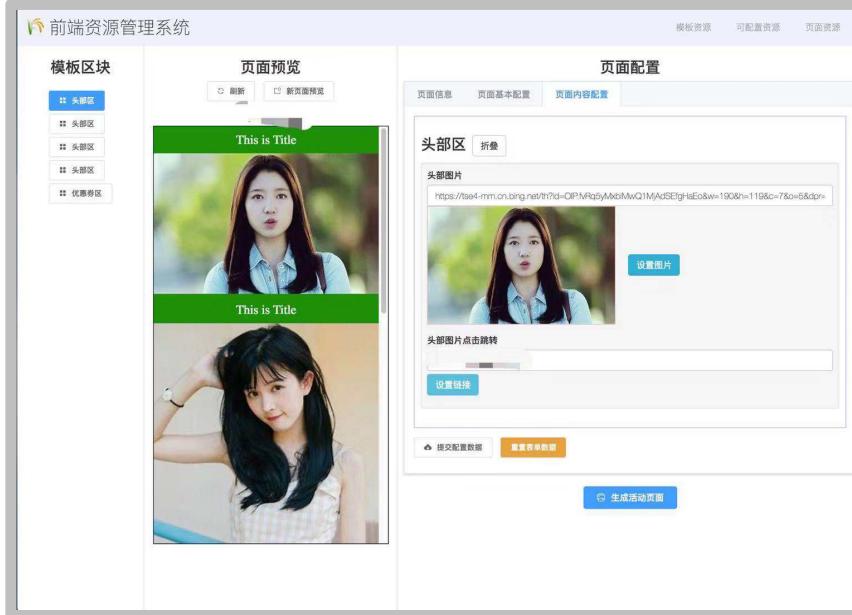
区分维度：

- **面向客群**: 工具的主要服务人群
- **系统功能**: 工具提供的核心能力
- **编辑自由度**: 页面可编辑单元的粒度

区分维度/面向客群



页面小白



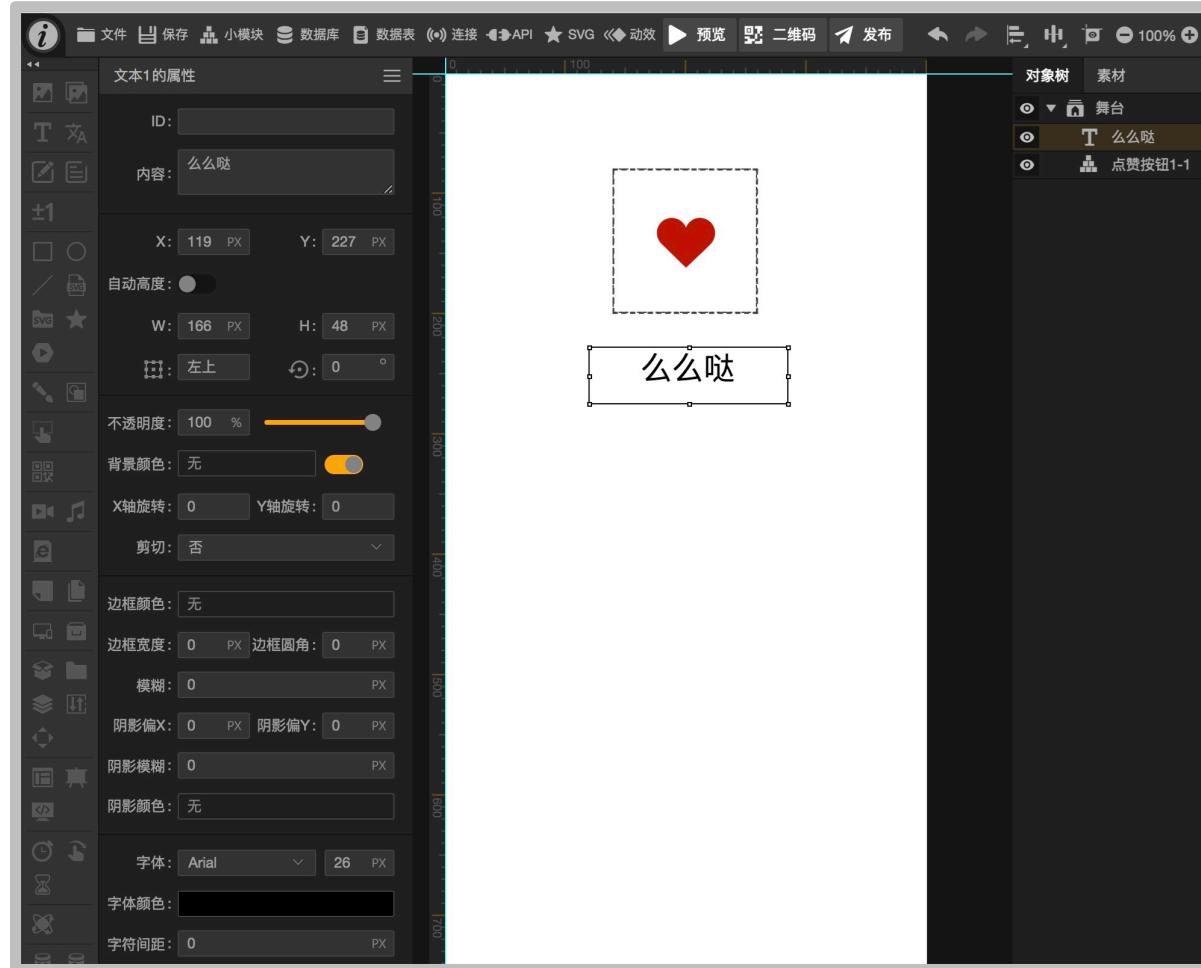
运营和产品



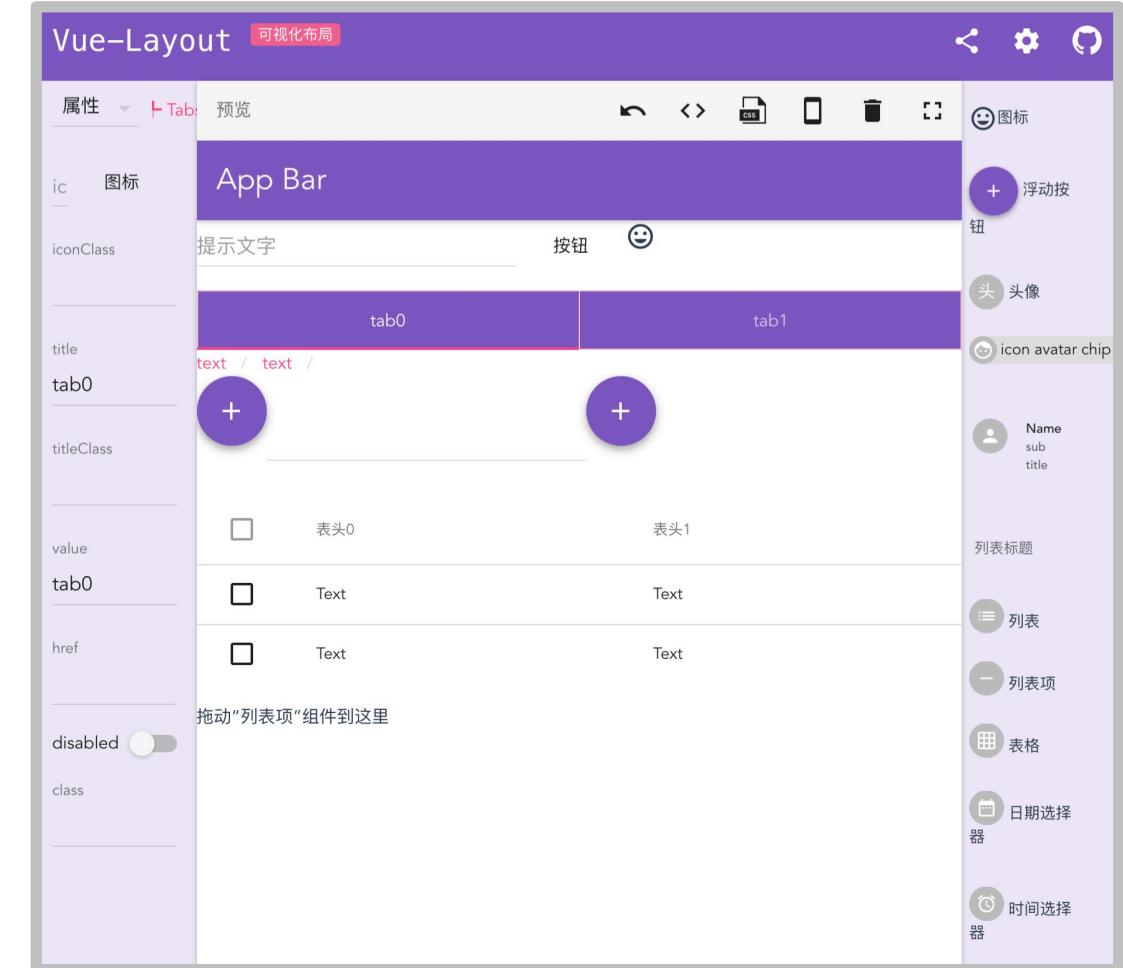
中后台开发人员

2 可视化搭建工具前世今生

区分维度/系统功能



HTML Tree 编辑



Component Tree 编辑

区分维度/系统功能



未命名标题

组件：卡片列表

[↓ 下载 excel](#) [↑ 导入 excel](#)

1. 云凤蝶在巴黎

小标题
Pairs

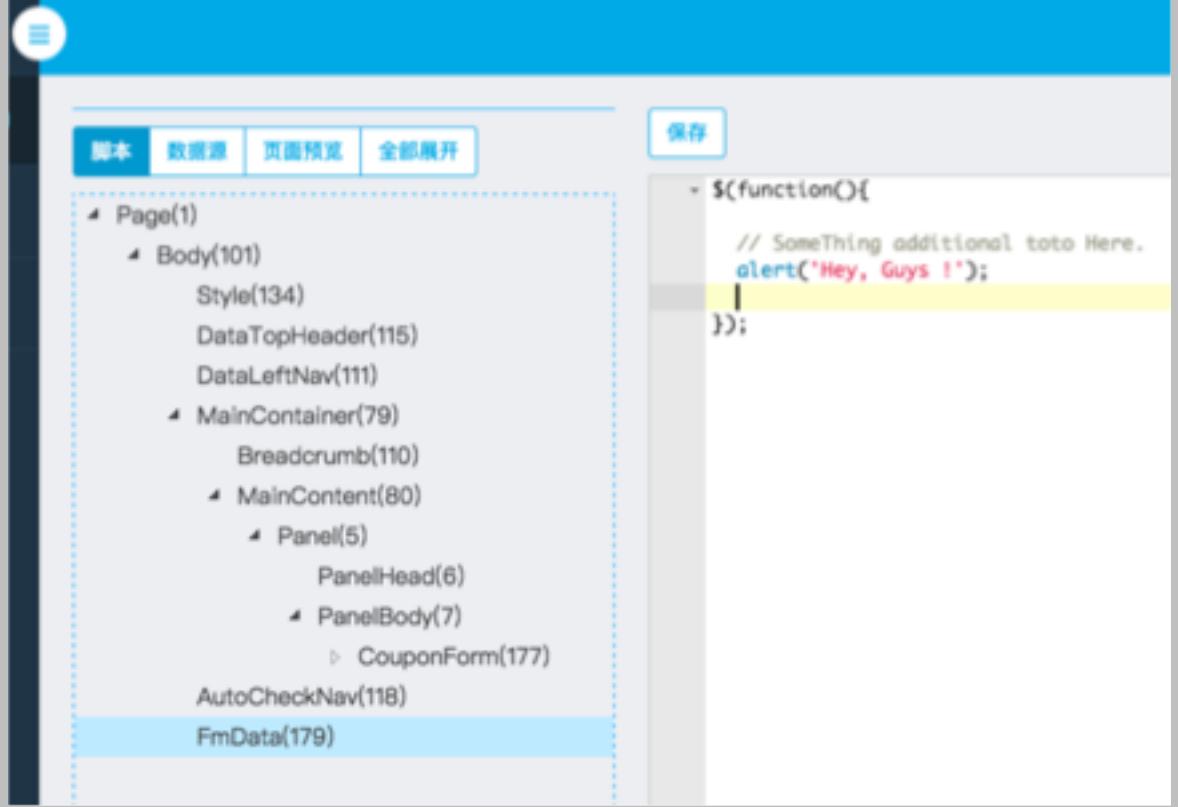
大标题
云凤蝶在巴黎

图片，建议尺寸 750x320
[↑ 选择图片](#)

 <https://gw.alipayobjects.com/zos/rmsportal/xFNIVxfXpGuRXPzaO> xjC.png

跳转链接
<https://www.alipay.com/>

Data 编辑



剧本

- Page(1)
 - Body(101)
 - Style(134)
 - DataTopHeader(115)
 - DataLeftNav(111)
 - MainContainer(79)
 - Breadcrumb(110)
 - MainContent(80)
 - Panel(5)
 - PanelHead(6)
 - PanelBody(7)
 - CouponForm(177)
 - AutoCheckNav(118)
 - FmData(179)

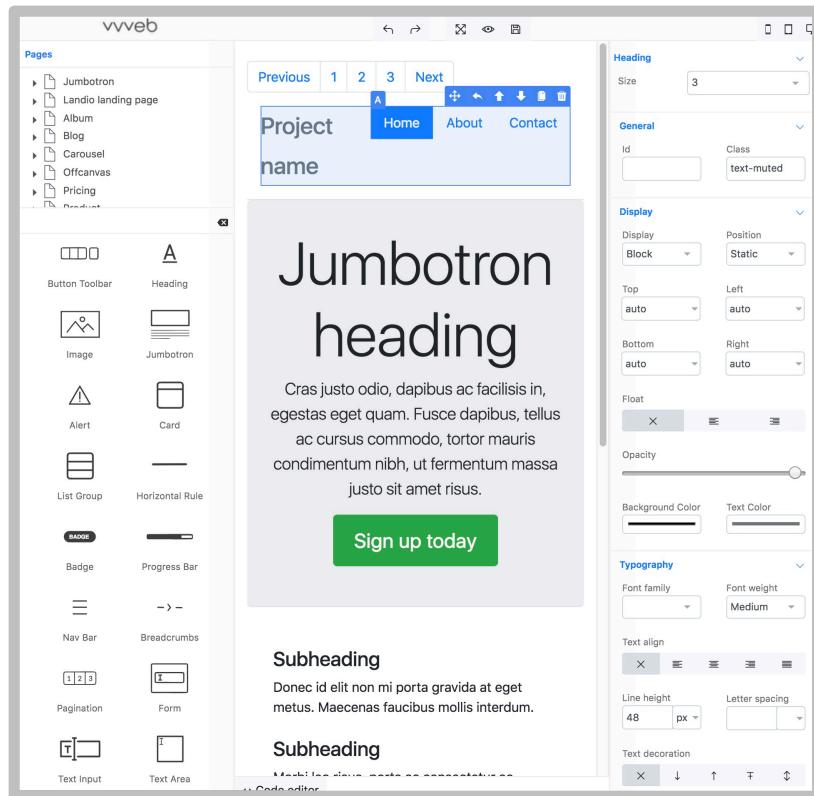
```

- ${function(){
  // Something additional todo here.
  alert('Hey, Guys !');
});
  
```

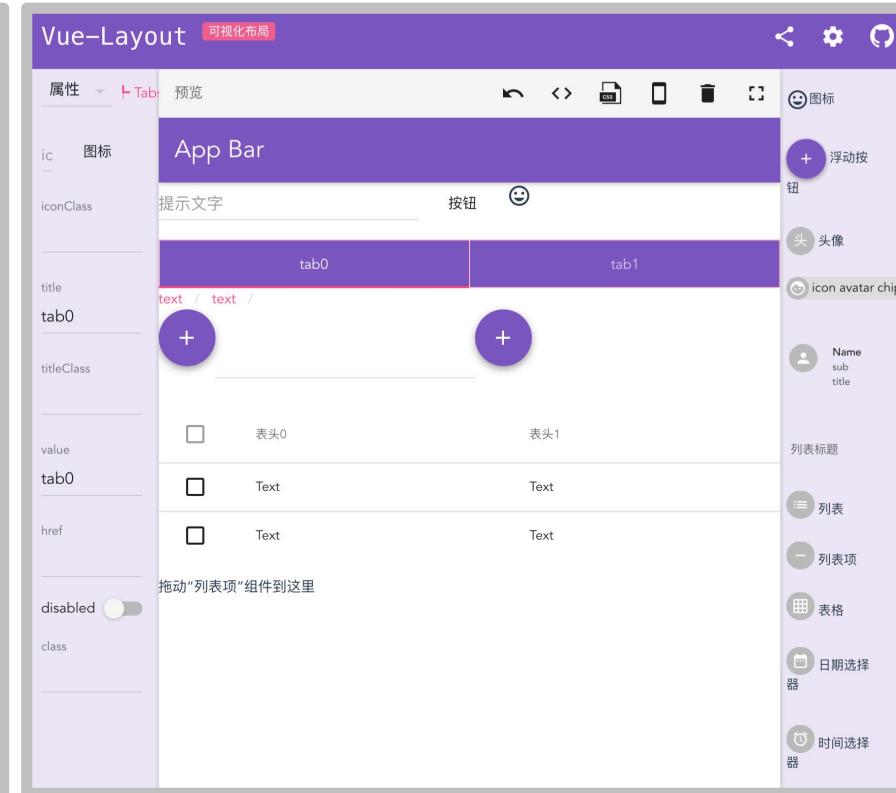
Dynamic Logic 编辑

2 可视化搭建工具前世今生

区分维度/编辑自由度



自由度为 HTML



自由度为 Component



自由度为不嵌套的组件

活动页面可视化搭建工具

区分维度:

- **面向客群:** 工具的主要服务人群
- **系统功能:** 工具提供的核心能力
- **编辑自由度:** 页面可编辑单元的粒度



需要什么:

- **运营和产品**
- **Data 编辑功能**
- **组件级别**

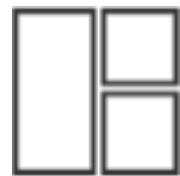
目录

1. 活动页面开发痛点(10%)
2. 可视化搭建工具前世今生(20%)
3. 可视化搭建工具技术要点(35%)
4. 理想的活动页面搭建工具(25%)
5. 开源搭建框架 *pipeline* (10%)

六个技术要点



页面组件化



页面模板



页面编辑



组件层级关系



实时预览

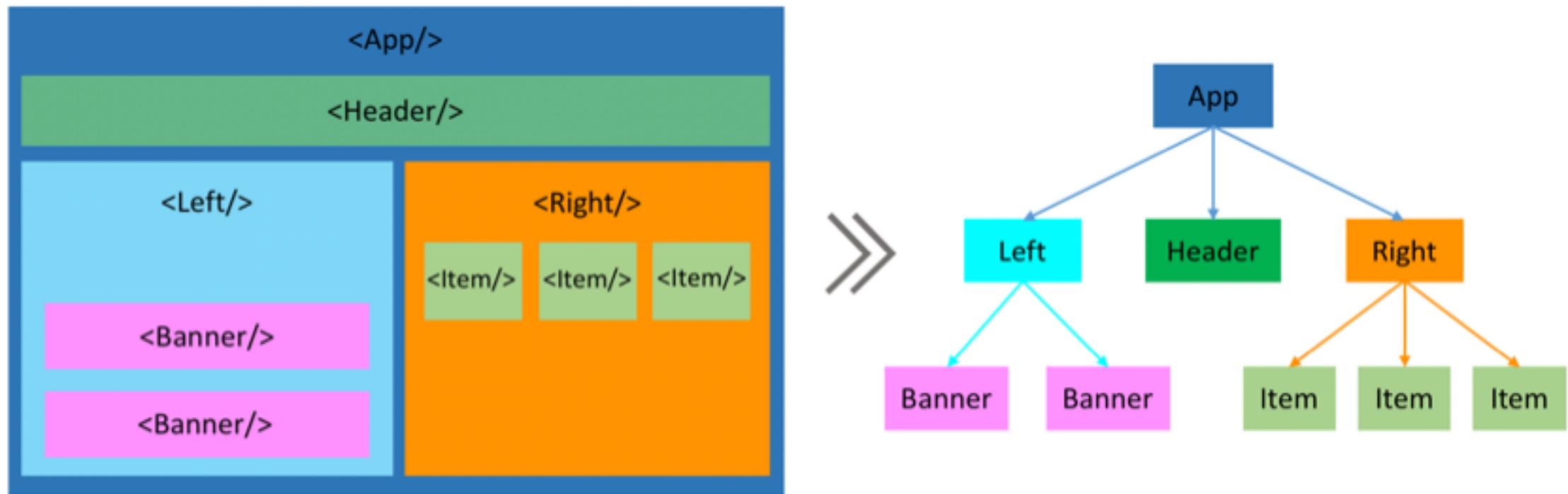


组件开发体验

1. 页面组件化/ 组件化的优点

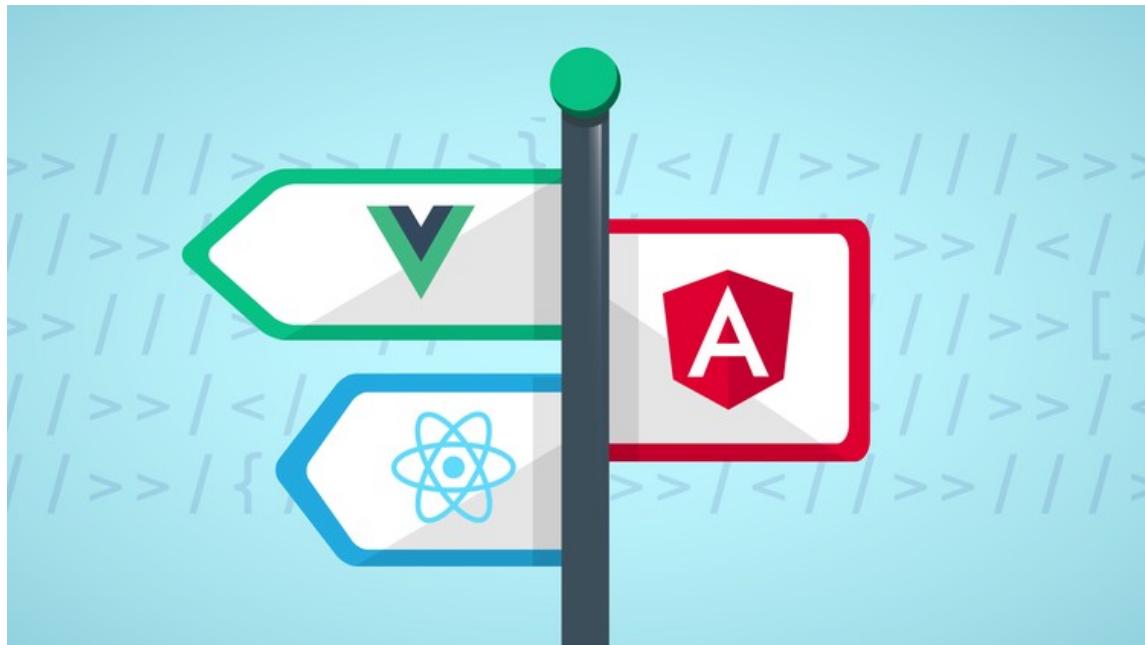
页面搭建问题简化:

- 组件树组合
- 组件props编辑



1. 页面组件化/页面前端框架

页面组件化依靠**前端框架**实现，可视化搭建工具需适配页面前端框架

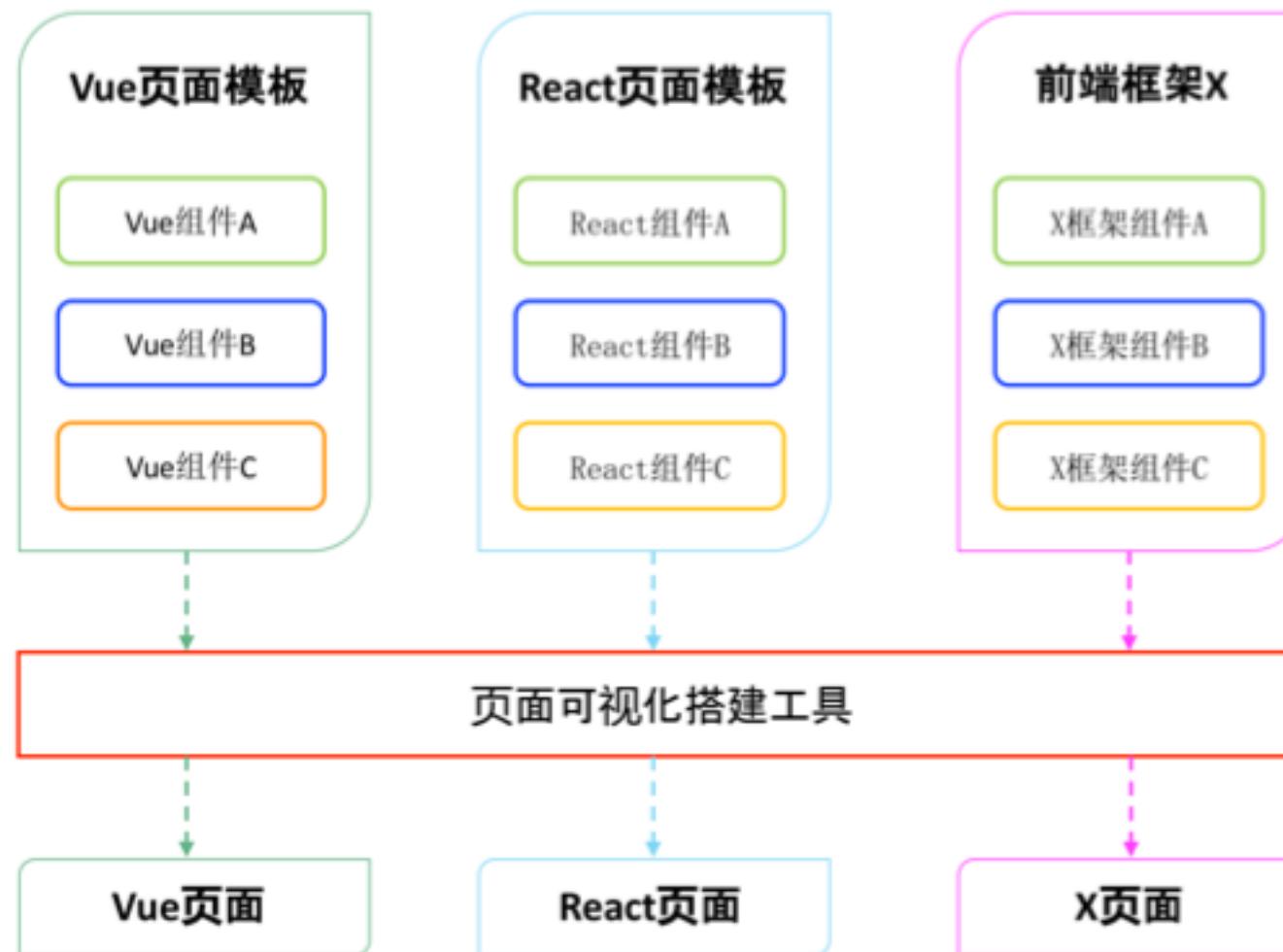


业务前端框架：

- 沉淀了大量技术组件和业务组件
- 开发人员熟悉业务前端框架

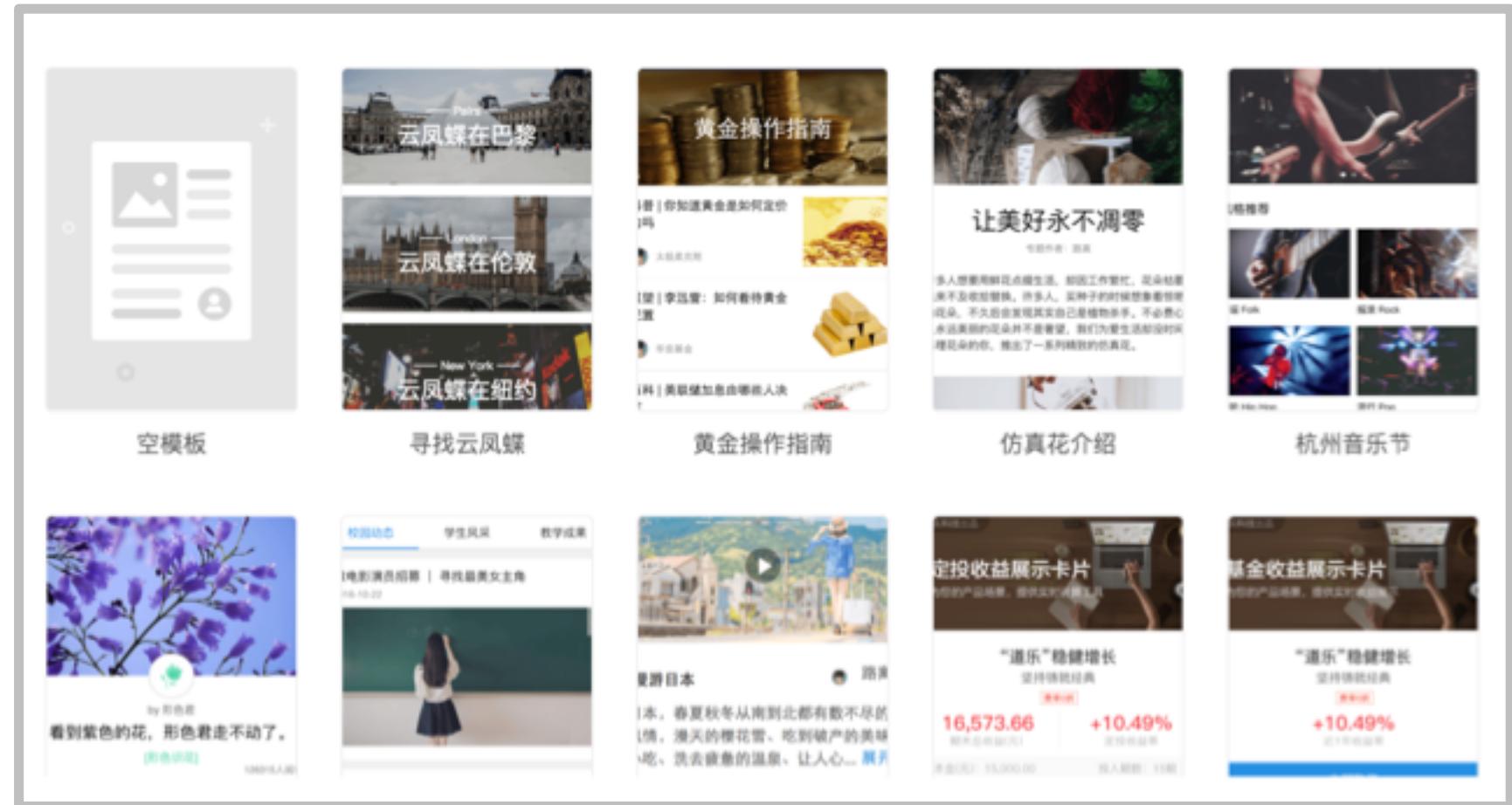
1. 页面组件化/页面前端框架

难点1: 如何实现页面可视化搭建工具与页面前端框架的解偶



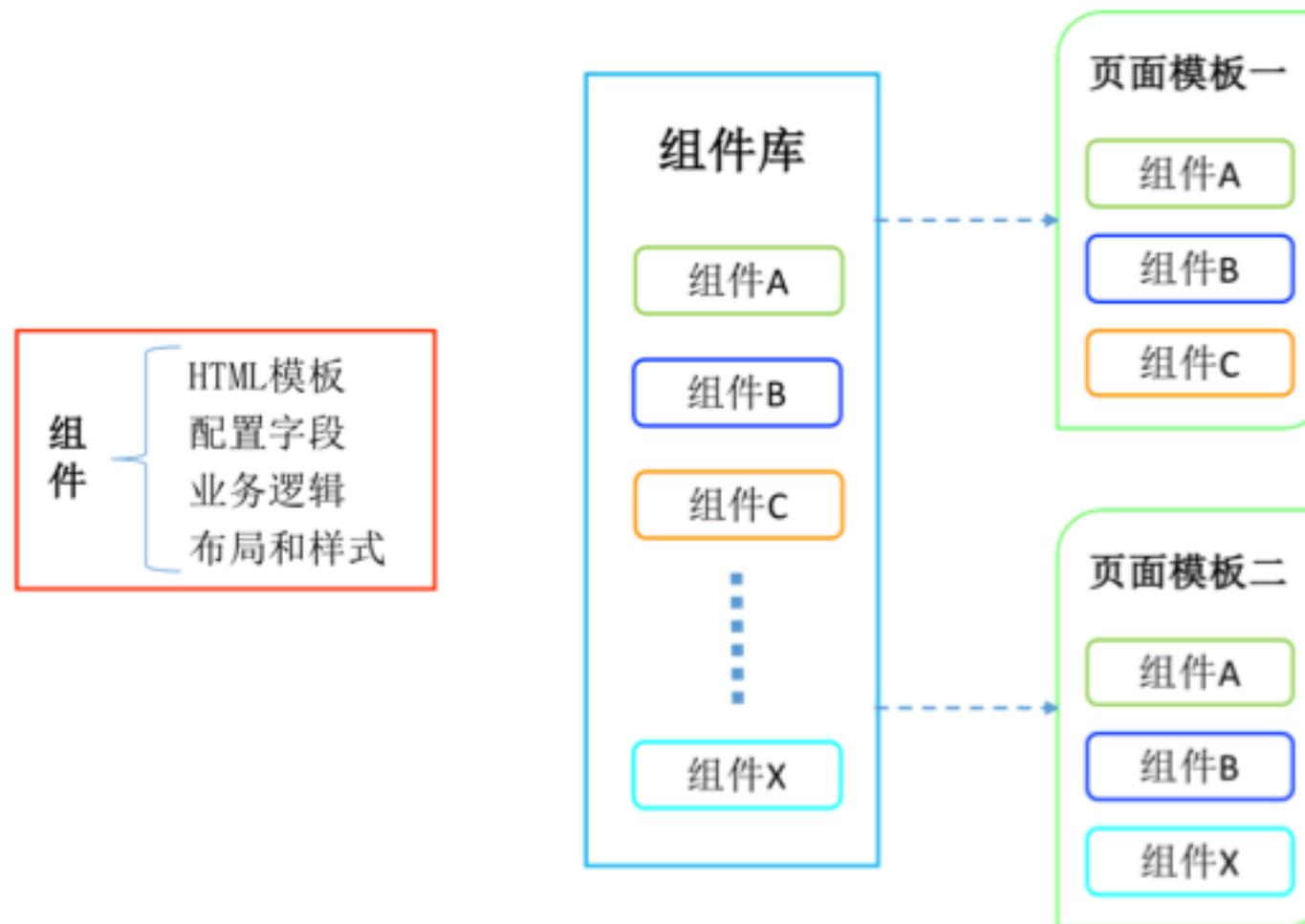
2. 页面模板

模板是带有默认数据的页面



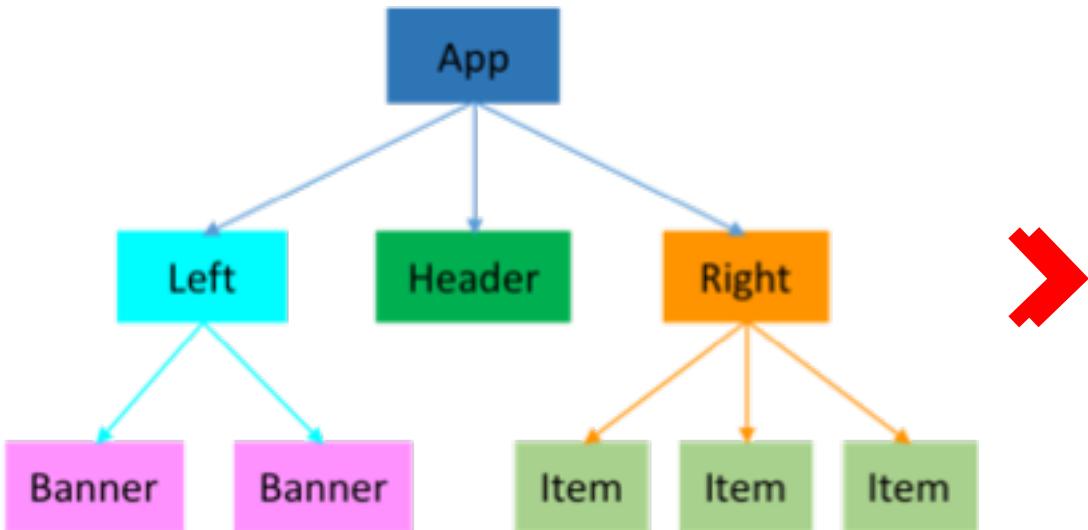
2. 页面模板

模板是从组件库中选取部分组件，并带有各个组件的默认配置数据



3. 页面编辑/组件树编辑

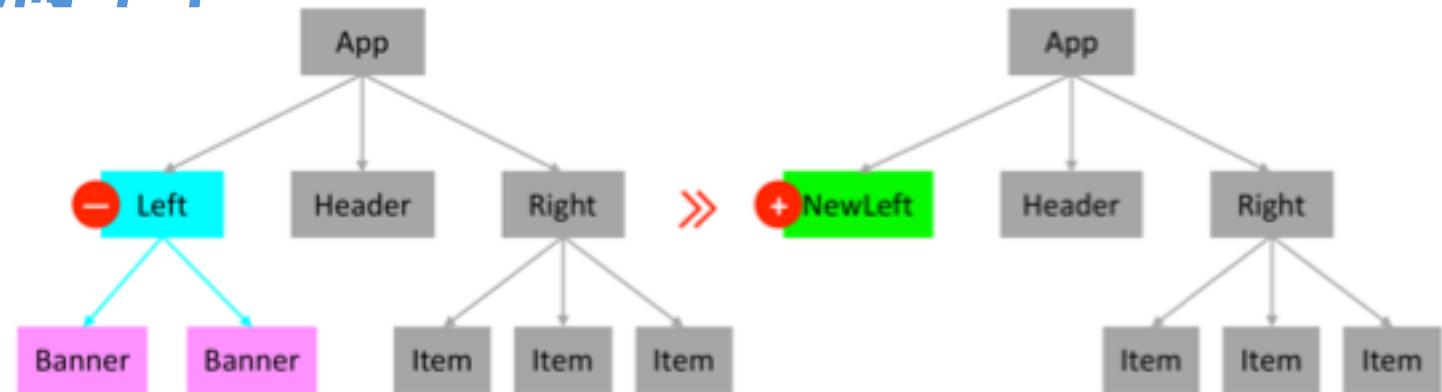
组件树代码写法



```
render() {
  return (
    <App>
      <Header></Header>
      <Left>
        <Banner type={'a'}></Banner>
        <Banner type={'b'}></Banner>
      </Left>
      <Right>
        { [1, 2, 3].map(i => <Item>i</Item>) }
      </Right>
    </App>
  )
}
```

3. 页面编辑/组件树编辑

更新源码, 重新打包



```
68 render() {  
69   ...  
70   ... <App>  
71   ...   <Header></Header>  
72 -   ...   <Left>  
73 -     ... <Banner type={a}></Banner>  
74 -     ... <Banner type={b}></Banner>  
75 -   ... </Left>  
76   ...   <Right>  
77   ...     { [1, 2, 3].map(i => <Item>i</Item>) }  
78   ... </Right>  
79   ... </App>  
80   ...;  
81 }
```

```
68 render() {  
69   ...  
70   ... <App>  
71   ...   <Header></Header>  
72 +   ...   <NewLeft>  
73 +     ... </NewLeft>  
74   ...   <Right>  
75   ...     { [1, 2, 3].map(i => <Item>i</Item>) }  
76   ... </Right>  
77   ... </App>  
78   ...);  
79 }
```

3. 页面编辑/组件树编辑

动态组件，编辑组件树后用新的组件声明渲染出页面

```
[  
  ...  
  {"  
    "id": "header-demo",  
    "name": "header-demo"  
  },  
  ...  
  {"  
    "id": "info-demo",  
    "name": "info-demo"  
  },  
  ...  
  {"  
    "id": "gap-demo",  
    "name": "gap-demo"  
  },  
  ...  
  {"  
    "id": "weather-demo",  
    "name": "weather-demo"  
  }  
]
```

Vue组件树声明



```
<template>  
  <div class="vue-main">  
    <component  
      v-for="oneComponent in mycomponents"  
      :key="oneComponent.id"  
      :is="oneComponent.name">  
    </component>  
  </div>  
</template>
```

Vue动态组件

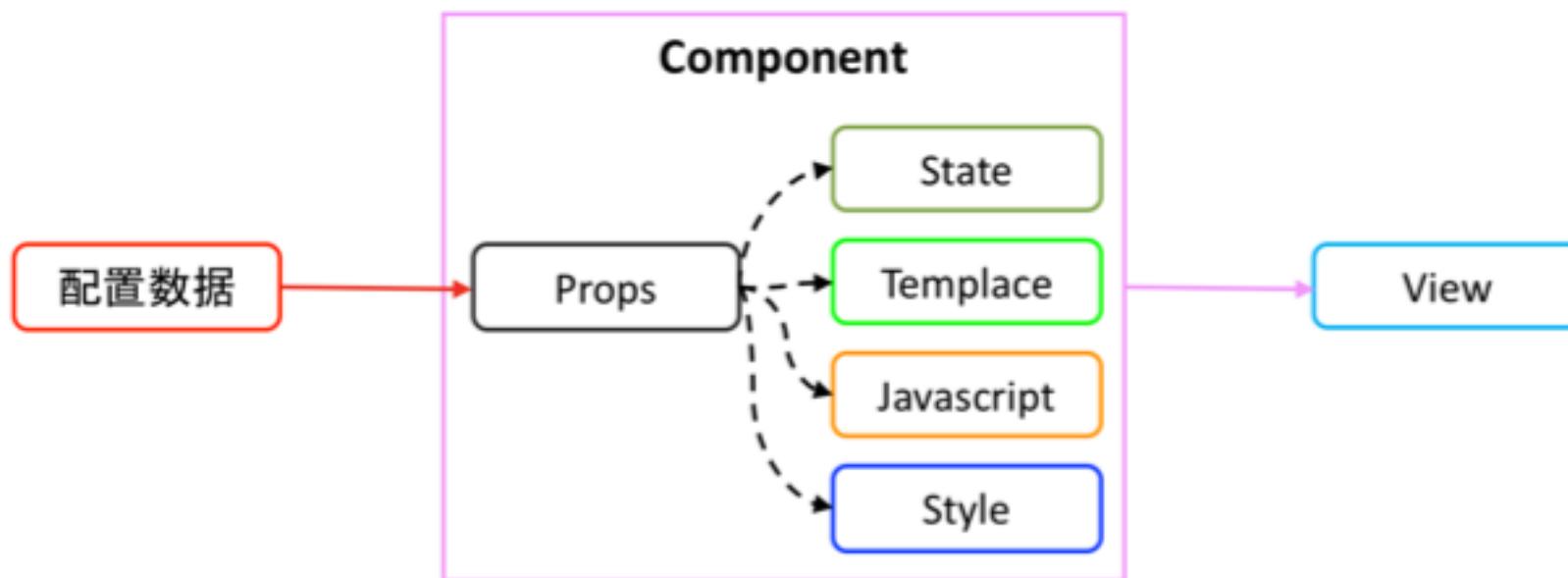


动态组件渲染成页面

3. 页面编辑/页面内容

内容编辑, 页面中各组件的组件属性进行配置

组件包含组件属性(*Props*), 组件状态(*State*), 组件模板(*Template*),
组件业务逻辑(*Javascript*), 组件样式布局(*Style*)等



3. 页面编辑/页面内容

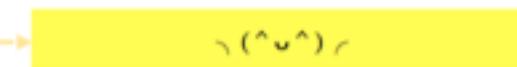
组件是差异化的，组件的配置属性也是差异化的

理想的数据格式为 [JSON](#)

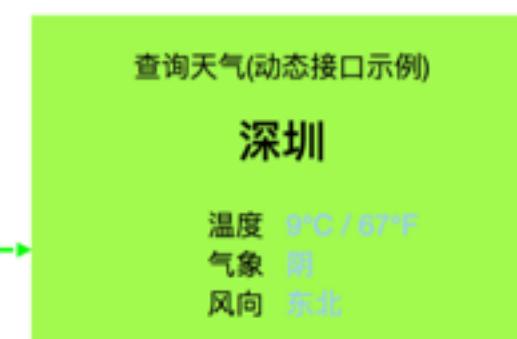
```
{  
  "title": "页面可视化搭建框架",  
  "src": "https://github.com/u/38666040",  
  "link": "https://github.com/page-pipeline"  
}
```



```
{  
  "backgroundColor": "#FFFF00",  
  "textColor": "#000000",  
  "text": "\u0329(^o^)\u0329",  
  "height": 48  
}
```

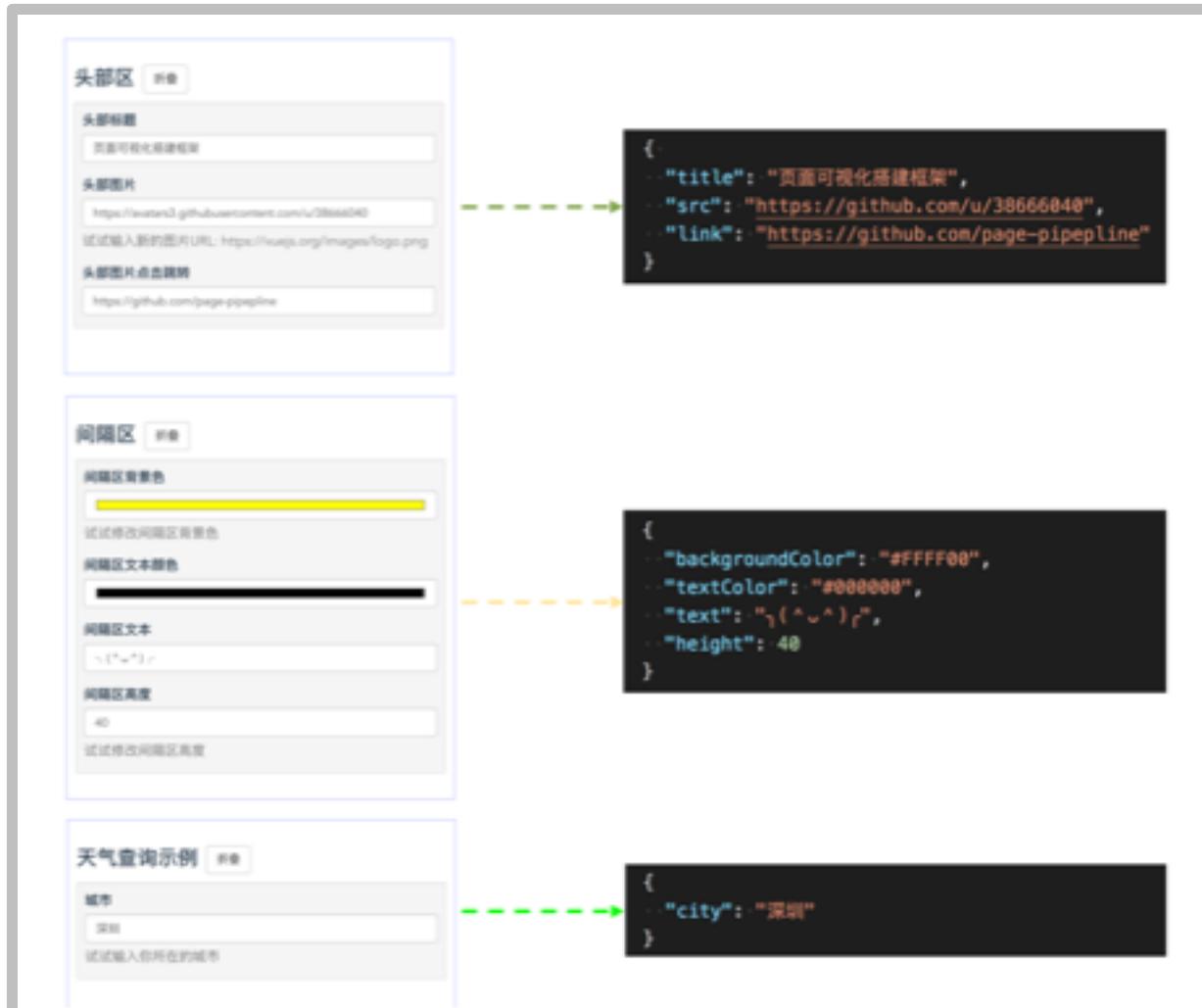


```
{  
  "city": "深圳"  
}
```



3. 页面编辑/ 页面内容配置表单

提供可视化的编辑方式 -- 使用组件配置表单来填入配置数据



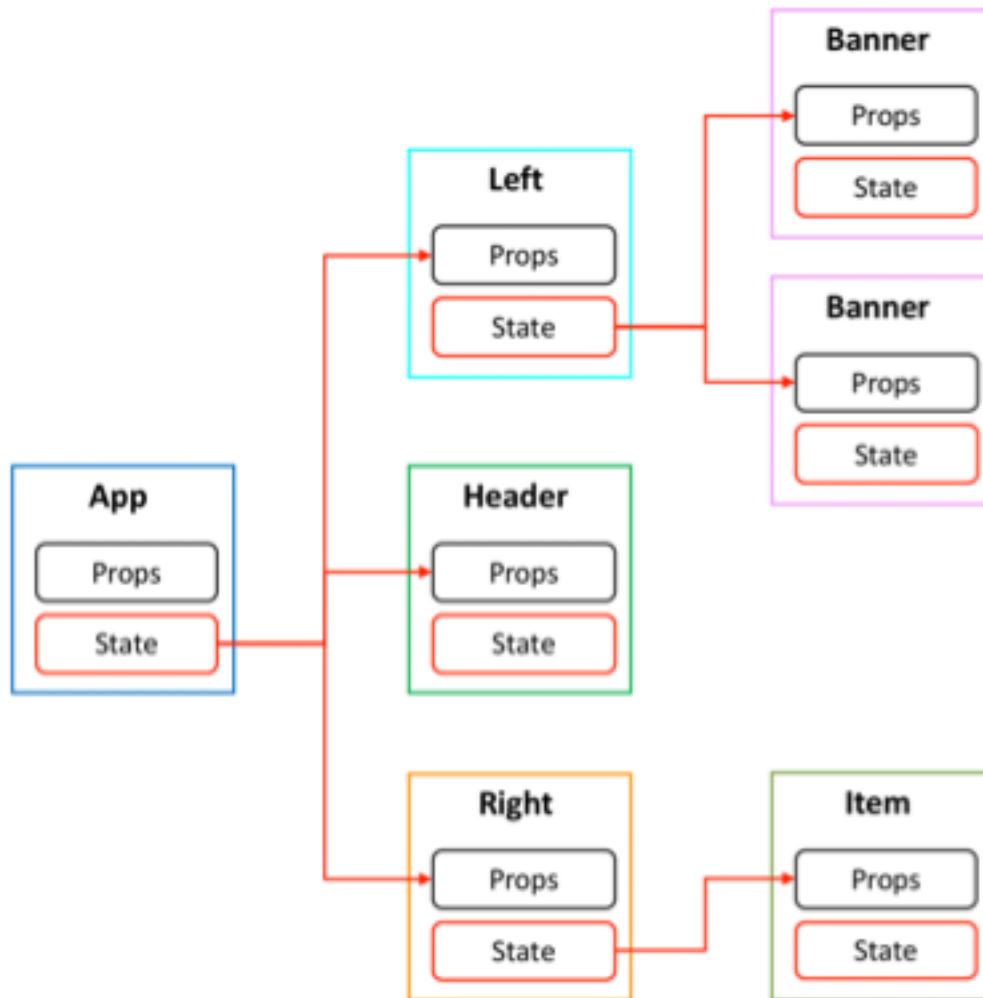
配置表单交互功能完善, 容易使用

配置表单可以内置类型限定, 避免填入错误的配置数据类型

难点2: 如何快速生成配置数据编辑表单

4. 组件层级关系

组件树定义了组件层级关系，父子组件通过数据流和事件进行关联

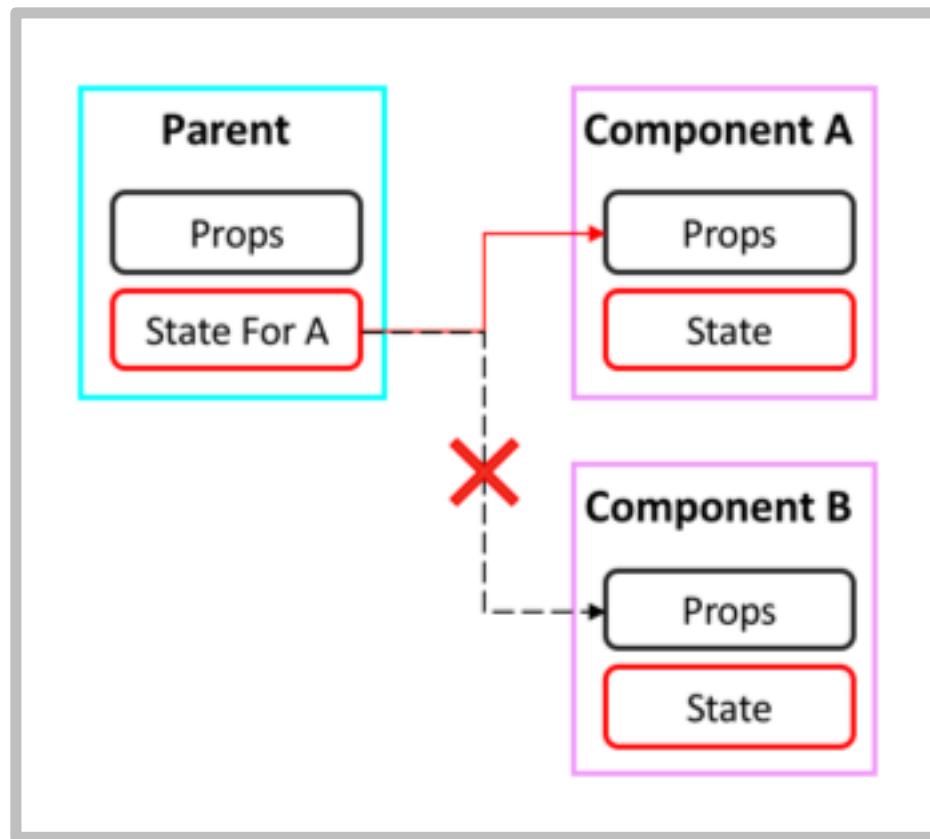


数据从父组件的 *State* 传递到子组件的 *Props*

子组件的变更触发 *Event* 通知父组件

4. 组件层级关系

组件层级关系变化影响数据流和布局



HTML ▾

```
<!DOCTYPE html>
<html>
<head>
  <title>组件嵌套</title>
</head>
<body>
  <span>行内组件</span>

  <span>
    行内组件包含块级组件
    <p>块级组件</p>
  行内组件包含块级组件
  </span>
</body>
</html>
```

行内组件 行内组件包含块级组件
块级组件
行内组件包含块级组件

难点3: 如何组织页面组件的层级关系

5. 页面实时预览

组件库

- ★ 间隔区示例
- ★ 头部区示例
- ★ 信息区示例
- ★ 查询天气示例
- (拖拽或双击)

模板组件

- 头部区示例
- 信息区示例
- 间隔区示例
- 查询天气示例
- 间隔区示例
- 头部区示例

页面预览

刷新 新页面预览

Hello Pipeline

页面可视化搭建框架



框架特性

- 可视化页面搭建框架
- 支持自由拓展页面组件
- 自定义页面可配置字段
- 模板工程/编辑器/后台服务解耦

页面配置

页面基本配置 页面内容配置

头部区 折叠

头部标题

页面可视化搭建框架

头部图片

https://avatars3.githubusercontent.com/u/3

试试输入新的图片URL:
https://vuejs.org/images/logo.png

头部图片点击跳转

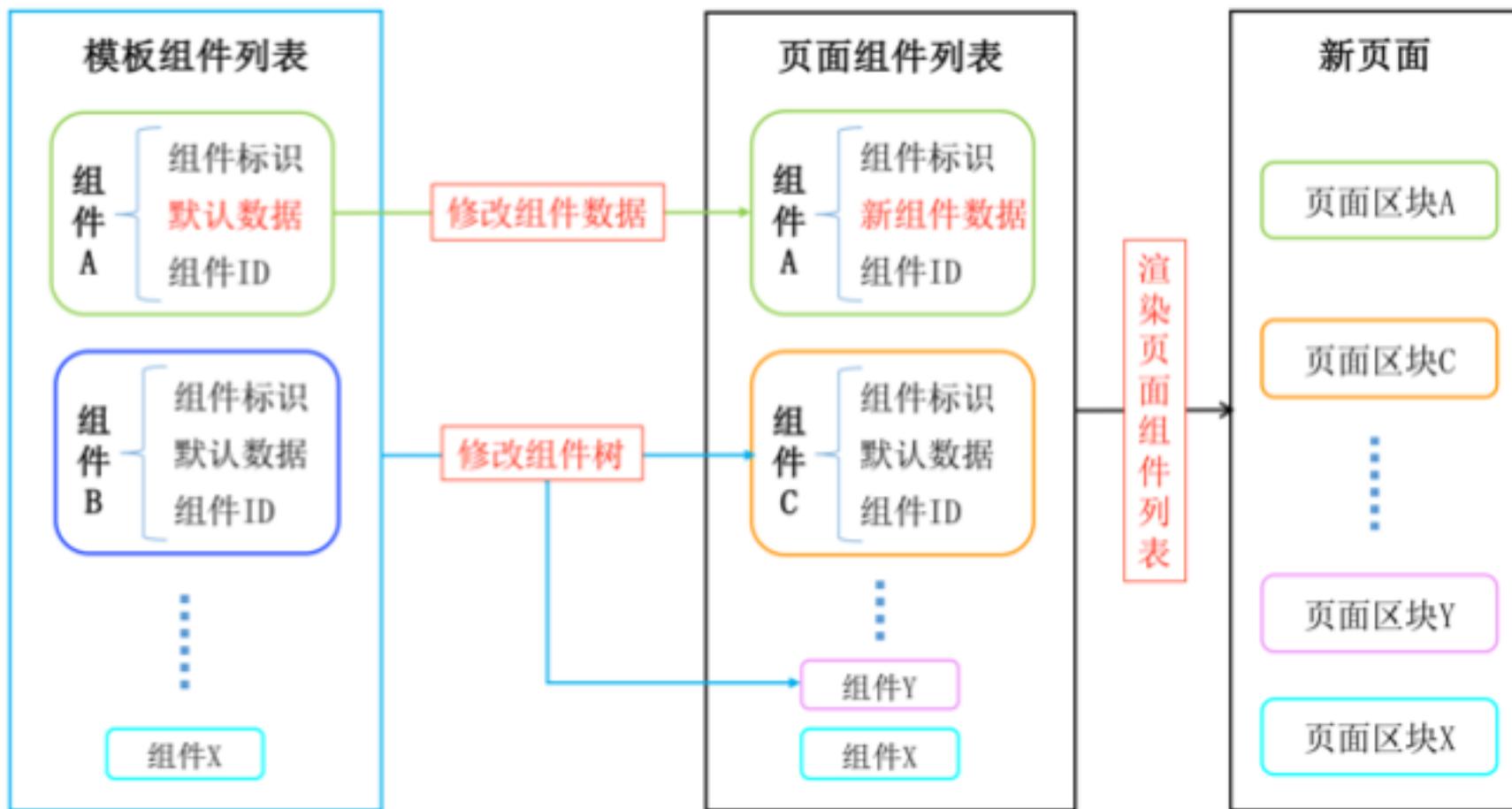
https://github.com/page-pipeline

提交配置数据

5. 页面实时预览

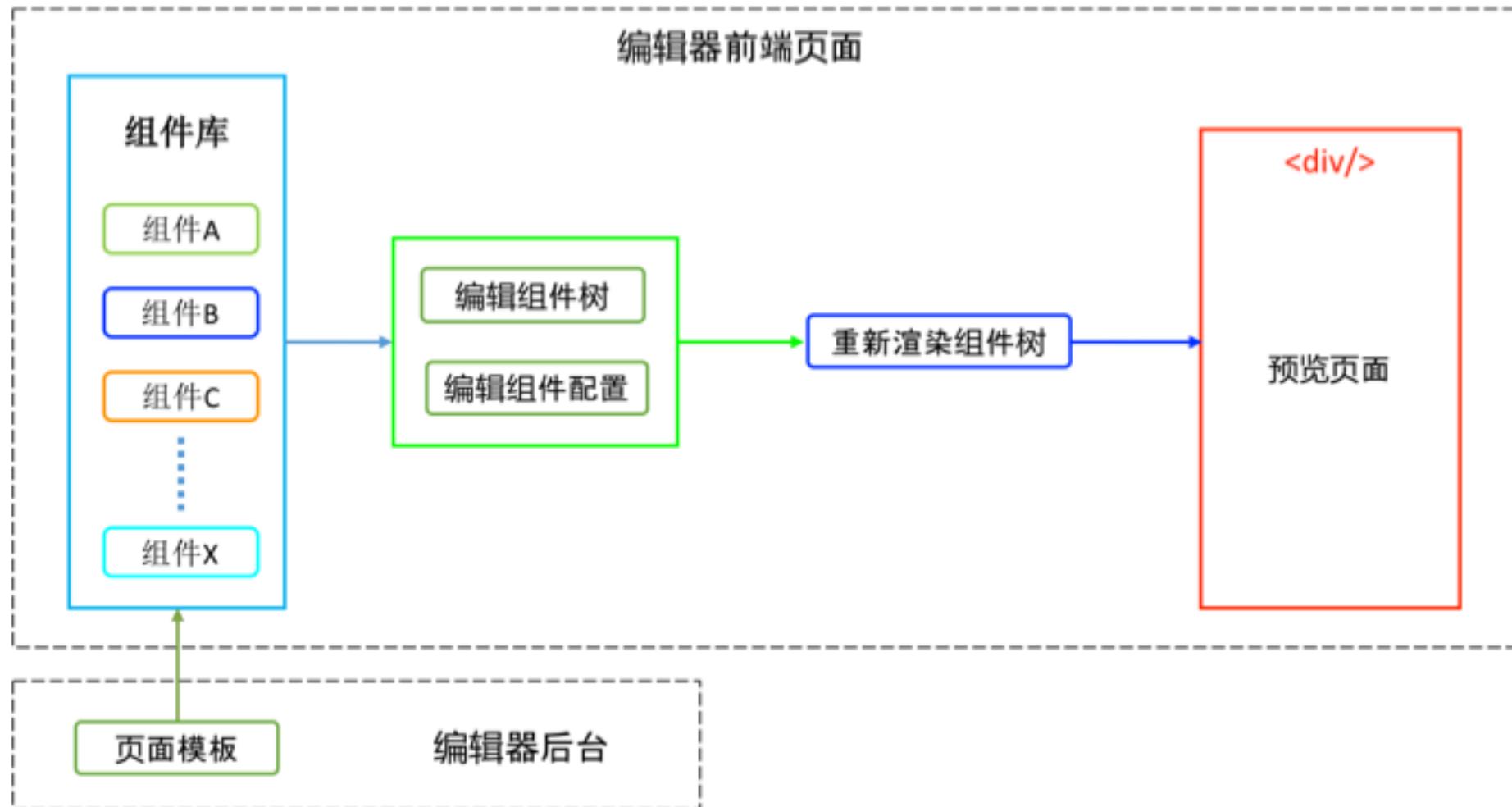
修改组件树或修改组件数据后生成新页面

实现页面预览两种方式: **页面挂载和后台渲染**



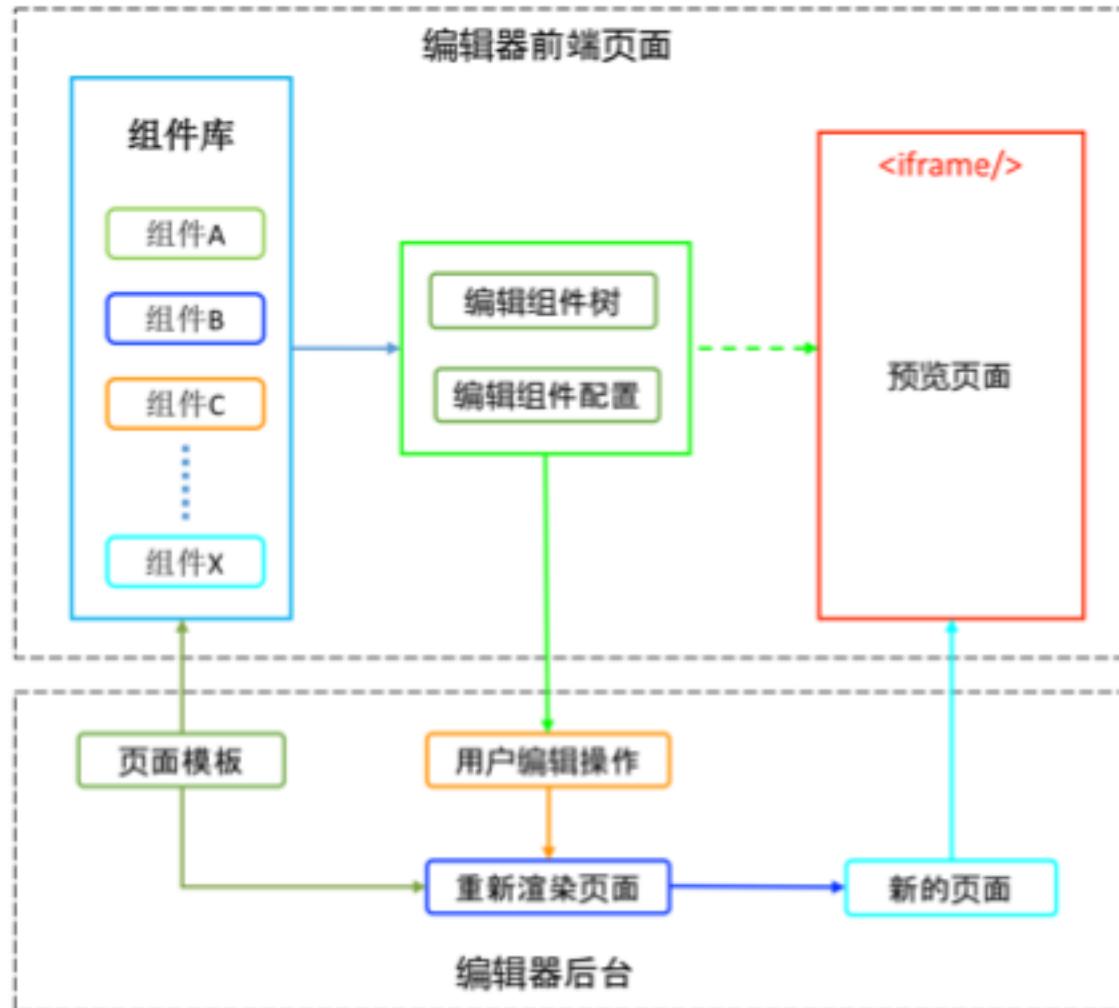
5. 页面实时预览

页面挂载: 在编辑器前端页面的某个元素节点(`div`)上渲染出用户编辑结果.



5. 页面实时预览

后台渲染: 后台进行页面渲染和生成, 编辑器前端页面通过 `iframe` 展示



难点4: 如何实现后台渲染

6. 组件开发体验

业务需求: 根据不同业务场景进行业务组件和页面模板的自定义开发

要求:

- 页面可视化搭建工具支持业务前端框架
- 组件和模板的编写方式需遵循**较简单的编写约定**
- 自定义模板和组件在开发模式下可以**调试和测试**

目录

1. 活动页面开发痛点(10%)
2. 可视化搭建工具前世今生(20%)
3. 可视化搭建工具技术要点(35%)
4. 理想的活动页面搭建工具(25%)
5. 开源搭建框架 *pipeline* (10%)

技术难点

难点1: 页面前端框架解偶

难点2: 快速生成编辑表单

难点3: 组织组件层级关系

难点4: 实现后台渲染

实现组件嵌套 解决组件嵌套

组件铺满页面宽度，在页面高度方向顺序排列



解决前端框架依赖：动态组件

组件列表声明 *./config/components.json*

```
[  
  {  
    "id": "pipeline-header-demo",  
    "name": "pipeline-header-demo",  
    "data": {  
      "title": "页面可视化搭建框架",  
      "src": "https://avatars3.githubusercontent.com/u/38666040",  
      "link": "https://github.com/page-pipeline"  
    }  
  },  
  { ...  
  }]
```

解决前端框架依赖: 动态组件

```
import mycomponents from './config/components'; // 引入组件声明文件

// 组件声明优先从全局对象上取
const pipelineComponents = (typeof window !== 'undefined') ?
  window.INIT_DATA || mycomponents : mycomponents;

// 页面中的组件类
const components: {
  'pipeline-gap-demo': PipelineGapDemo,
  'pipeline-info-demo': PipelineInfoDemo,
  'pipeline-header-demo': PipelineHeaderDemo,
  'pipeline-weather-demo': PipelineWeatherDemo,
};

// 根据组件声明实例化组件
const Components = pipelineComponents.map(oneComponent => {
  const OneComponent = components[oneComponent.name];
  return OneComponent && (
    <OneComponent
      key={oneComponent.id}
      data-component-id={oneComponent.id}
      data-component-name={oneComponent.name}
      config={oneComponent.data}/>
  );
});

// 渲染页面组件
return (
  <div className="App">
    {Components}
  </div>
);
```

引入组件列表声明

匹配到组件实例进行渲染

通过 *props* 传递配置数据

解决前端框架依赖: 动态组件

组件内部从 `props.config` 取数据, 决定组件渲染和功能逻辑

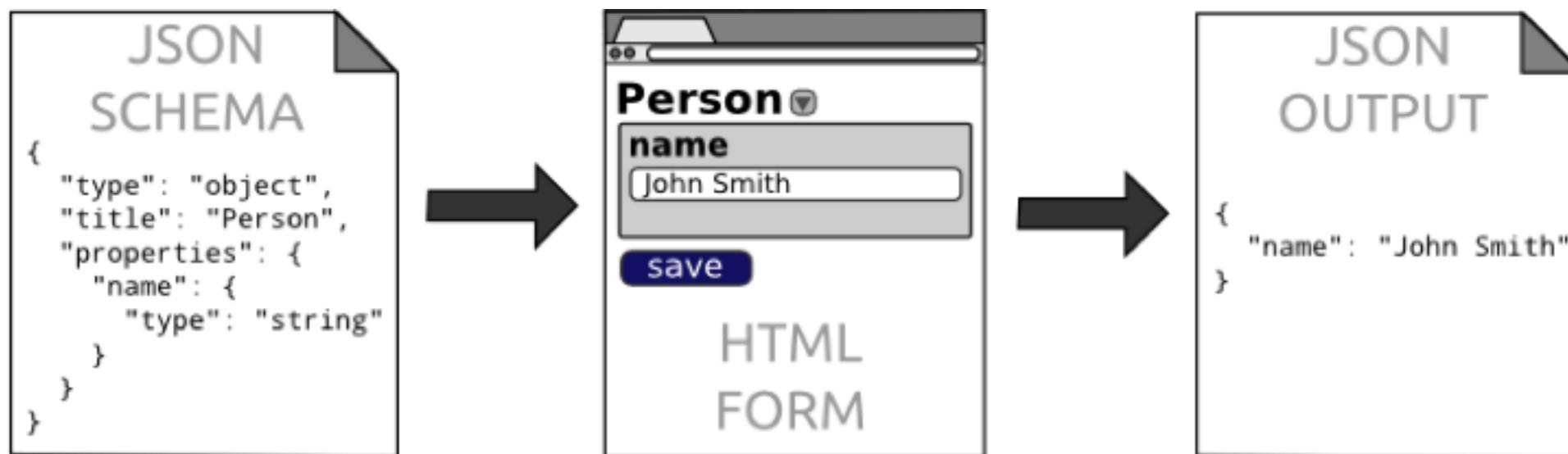
```
● ● ●

class PipelineHeaderDemo extends React.Component {
  imgClick = () => {
    window.location.href = this.props.config.link;
  }
  render() {
    return (
      <div className="header-container">
        <div className="title">{this.props.config.title}</div>
        <div className="header">
          <img className="header__img"
            src={this.props.config.src}
            onClick={this.imgClick}/>
        </div>
      </div>
    );
  }
}
```

实现配置表单生成

按照 JSON Schema 对 JSON 数据的描述, 动态渲染出配置表单

对编辑后的数据做格式校验, 避免编辑错误



实现配置表单生成



```
./pipeline-header-demo
├── config // 组件配置字段声明目录
│   └── data.json // 组件默认配置字段
│       └── schema.js // 组件配置字段的 Schema 声明
├── index.js // 组件源码
└── style.less
└── package.json // 组件描述文件
```

实现配置表单生成



```
module.exports = {  
  title: '头部区',  
  type: 'object',  
  properties: {  
    title: {  
      title: '头部标题',  
      type: 'string',  
      default: '头部标题',  
    },  
    src: {  
      title: '头部图片',  
      description: '试试输入新的图片URL ...',  
      type: 'string',  
      default: '头部图片'  
    },  
    link: {  
      title: '头部图片点击跳转',  
      type: 'string',  
      format: 'url',  
      default: '头部图片点击跳转',  
    },  
  }  
};
```

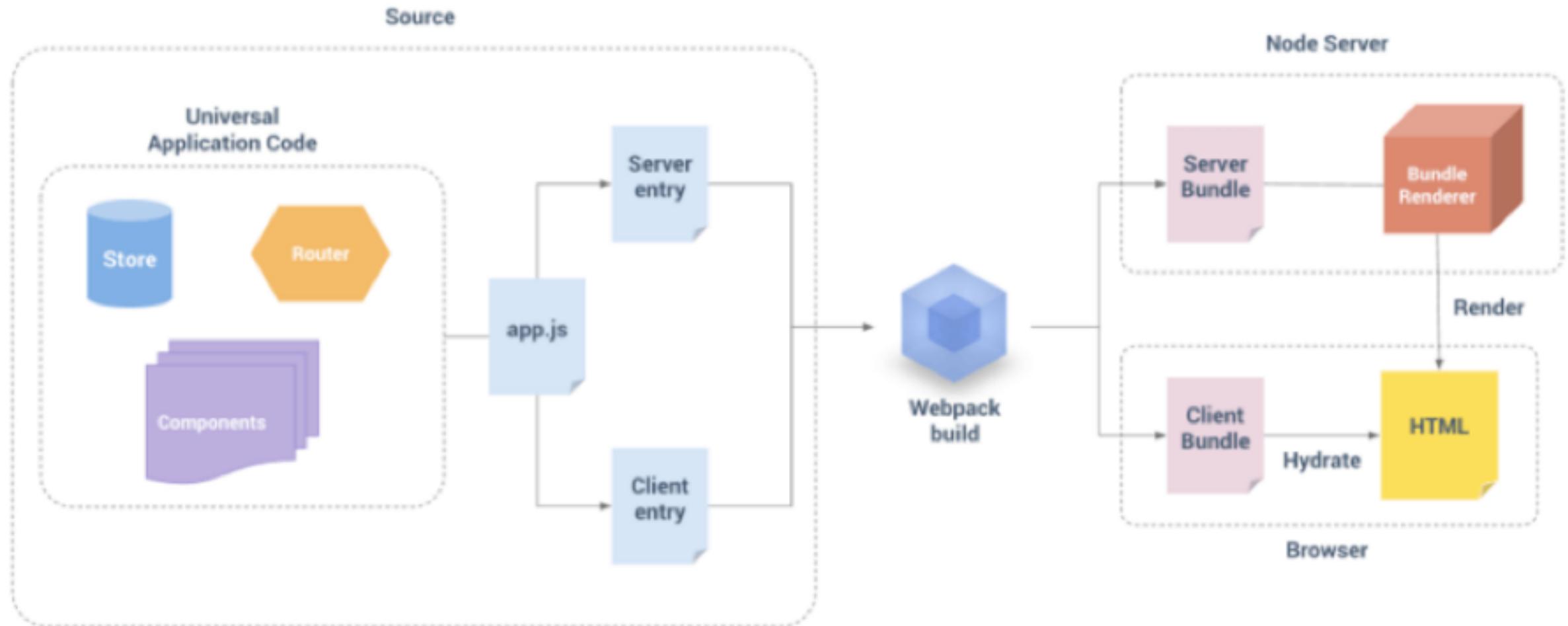
头部区 折叠

头部标题
页面可视化搭建框架

头部图片
<https://avatars3.githubusercontent.com/u/38666040>
试试输入新的图片URL: <https://vuejs.org/images/logo.png>

头部图片点击跳转
<https://github.com/page-pipeline>

实现后台渲染—SSR



实现后台渲染--SSR

浏览器运行组件声明优先从全局对象上取



```
import mycomponents from './config/components';

const pipelineComponents = (typeof window !== 'undefined') ?
  window.INIT_DATA || mycomponents : mycomponents;
```

实现后台渲染--SSR

SSR webpack 构建忽略组件列表声明 config/components.json

```
const webpackConfig = {  
  ...  
  externals: {  
    './config/components': {  
      commonjs2: '../config/components.json',  
      commonjs: '../config/components.json',  
    },  
  },  
  ...  
}
```

实现后台渲染--SSR

SSR 运行注入组件列表声明 config/components.json



```
const render = require('react-dom/server');
const App = require('./dist/server.js');

const SSR_string = render.renderToString(App.default);
const markup = `<div id="root">\n${SSR_string}\n</div>`;

const template = fs.readFileSync('./dist/index-origin.html', 'utf-8');
const initdata = fs.readFileSync('./config/components.json', 'utf-8');

const initdataScript = `<script type="text/javascript">
    window.INIT_DATA = ${initdata.replace(/\n+$/, '')};
</script>`;

const html = template.replace(/<!--react-ssr-outlet-->/, `${markup}\n${initdataScript}`);
```

实现后台渲染--SSR

后台渲染结果



```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div class="App" data-reactroot="">
      ...
      <div class="gap-container" style="height:40px;color:#000000;" data-component-id="pipeline-gap-demo" ...>
        <p>^ ^ ^</p>
      </div>
      ...
    </div>
    <script type="text/javascript">
      window.INIT_DATA = [...];
    </script>
    <script type="text/javascript" src="./js/main.a7048ff5.js"></script>
  </body>
</html>
```



```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div id="root">...</div>
    <script type="text/javascript">
      window.INIT_DATA = [
        {
          "id": "pipeline-header-demo",
          "name": "pipeline-header-demo",
          "data": {
            "title": "页面可视化搭建框架",
            "src": "https://avatars3.githubusercontent.com/u/38666040",
            "link": "https://github.com/page-pipeline"
          }
        },
        {...},
        {...}];
      </script>
      <script type="text/javascript" src="./js/main.a7048ff5.js"></script>
    </body>
</html>
```

理想工具

声明页面配置数据并提供配置表单, 通过对配置表单的数据填充, 实现基于模板的页面生成.



理想工具特点

采用组件化和页面模板提升页面生成效率

采用不嵌套的组件层级简化数据流和样式布局

采用 *JSON Schema* 声明配置数据, 自动生成配置表单

采用后台渲染, 实现编辑系统与组件前端框架解耦

在遵循编辑系统约定下, 组件可自由拓展, 前端框架可自由选择

目录

1. 活动页面开发痛点(10%)
2. 可视化搭建工具前世今生(20%)
3. 可视化搭建工具技术要点(35%)
4. 理想的活动页面搭建工具(25%)
5. 开源搭建框架 *pipeline* (10%)

Pipeline 概览



项目地址: <https://github.com/paoe-pipeline>

体验地址: <https://paoe-pipeline.github.io/pipeline-editor/dist/#/>

项目文档: [pipeline-document](#)

pipeline-editor

页面可视化搭建框架的 web 编辑器 -- <https://page-pipeline.github.io/pipeline-editor/dist/#/>

JavaScript 83 ★ 442 ① 2 0 Updated on Apr 25

pipeline-node-server

页面可视化搭建框架的后台服务

JavaScript 10 ★ 24 ① 0 0 Updated on Apr 25

pipeline-document

页面可视化搭建框架的文档

10 ★ 19 ① 2 0 Updated on Apr 16

pipeline-mongo

页面可视化搭建框架的 mongo 数据库

JavaScript 8 ★ 2 ① 0 0 Updated on Nov 16, 2018

pipeline-template-omi

页面可视化搭建框架的页面模板 - 基于 Omi

JavaScript 3 ★ 1 ① 0 0 Updated on Apr 17

pipeline-template

页面可视化搭建框架的页面模板 - 基于 Vue

JavaScript 15 ★ 51 ① 0 0 Updated on Apr 17

pipeline-template-react

页面可视化搭建框架的页面模板 - 基于 React

JavaScript 9 ★ 15 ① 0 0 Updated on Apr 17

pipeline-template-angular

页面可视化搭建框架的页面模板 - 基于 Angular

JavaScript 1 ★ 1 ① 0 0 Updated on Apr 1

Pipeline 支持的前端框架

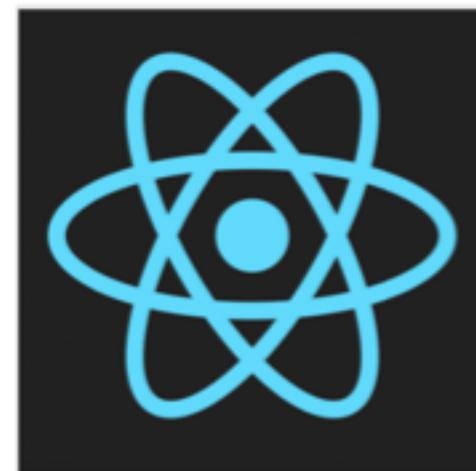
支持的前端框架模板列表

pipeline 支持不同前端框架开发的页面模板, 无缝对接业务现有前端技术栈.



Vue框架模板

使用 修改 删除



React框架模板

使用 修改 删除



Omi框架模板

使用 修改 删除



Angular框架模板

使用 修改 删除

5 开源搭建框架 pipeline

Pipeline 编辑器

组件库

- ★ 间隔区示例
- ★ 头部区示例
- ★ 信息区示例
- ★ 查询天气示例

(拖拽或双击)

模板组件

- 头部区示例
- 信息区示例
- 间隔区示例
- 查询天气示例
- 间隔区示例
- 头部区示例

页面预览

刷新 新页面预览

Hello Pipeline

页面可视化搭建框架

框架特性

可视化页面搭建框架
支持自由拓展页面组件
自定义页面可配置字段
模板工程/编辑器/后台服务解偶

页面配置

页面基本配置

页面内容配置

头部区 折叠

头部标题

页面可视化搭建框架

头部图片

试试输入新的图片URL:
<https://vuejs.org/images/logo.png>

头部图片点击跳转

<https://github.com/page-pipeline>

提交配置数据

Pipeline 编辑器

页面预览

页面配置表单

组件库

模板组件

组件树编辑区

头部区示例

信息区示例

间隔区示例

查询天气示例

(拖拽或双击)

头部区

头部标题

头部图片

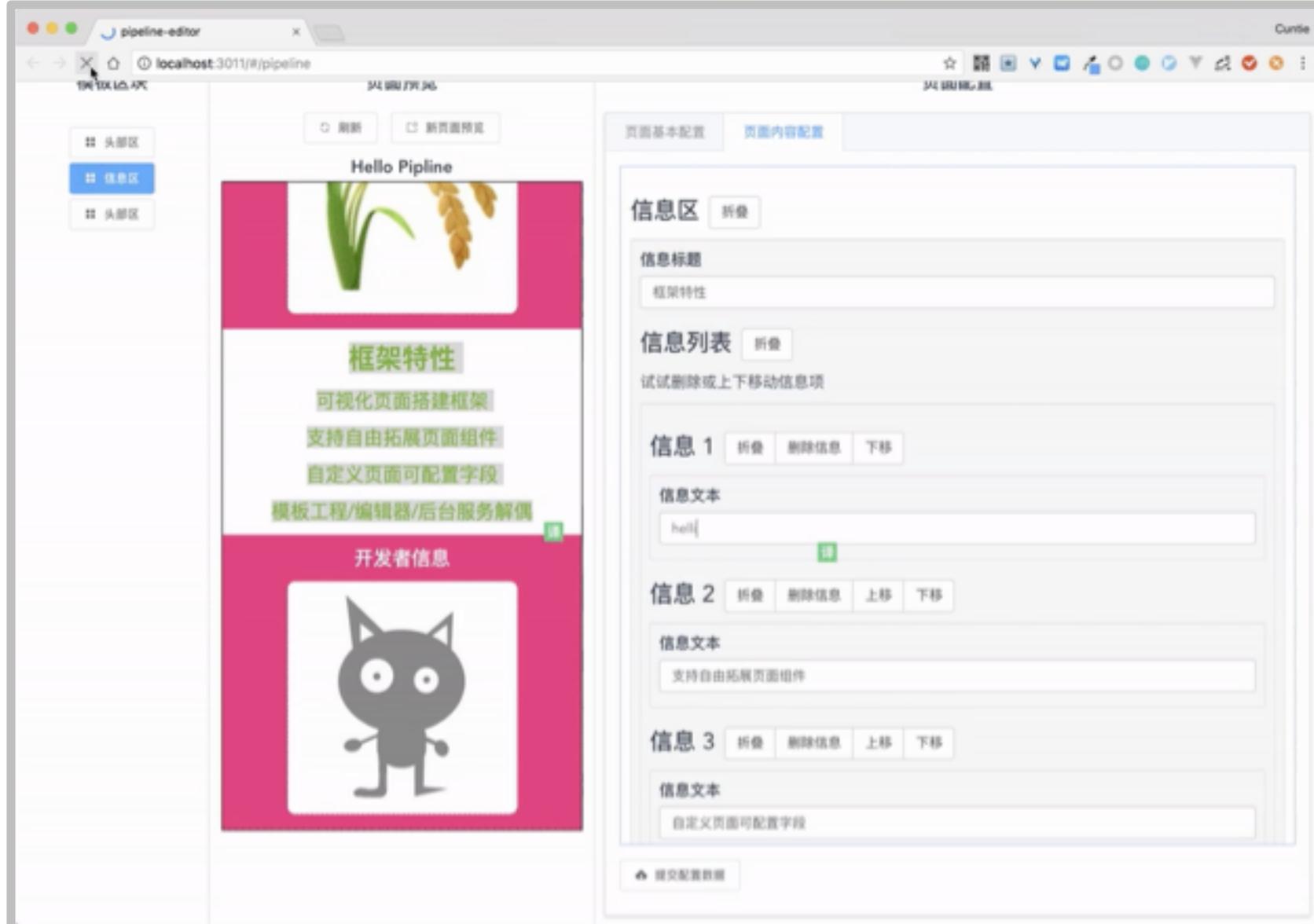
头部图片点击跳转

刷新 新页面预览

Hello Pipeline

页面可视化搭建框架

基本操作demo



修改页面全局配置
可视化修改组件内容
页面实时预览
即刻获取结果页面
页面支持业务逻辑

组件拖拽demo



动态增删页面组件
可视化的组件拖拽
页面支持业务逻辑

Pipeline 特点

- 模板工程/编辑器/后台服务解偶
- 支持自由拓展页面组件
- 自定义页面可配置字段
- 组件动态增减, 组件拖拽
- 模板工程前端框架无关: 支持主流前端框架

下一步工作

- 完善项目文档
- 增强编辑功能
- 支持更多前端框架
- 推广到社区

目录

1. 活动页面开发痛点(10%)
2. 可视化搭建工具前世今生(20%)
3. 可视化搭建工具技术要点(35%)
4. 理想的活动页面搭建工具(25%)
5. 开源搭建框架 *pipeline* (10%)

参考材料

前端服务化——页面搭建工具的死与生

前端即服务-通向零成本开发之路

阿里云凤蝶

页面可视化搭建工具前生今世

页面可视化搭建工具技术要点

Pipeline 前端框架配置约定

页面可视化搭建工具实践

陈韩杰 [@CatChen](#)