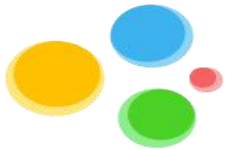


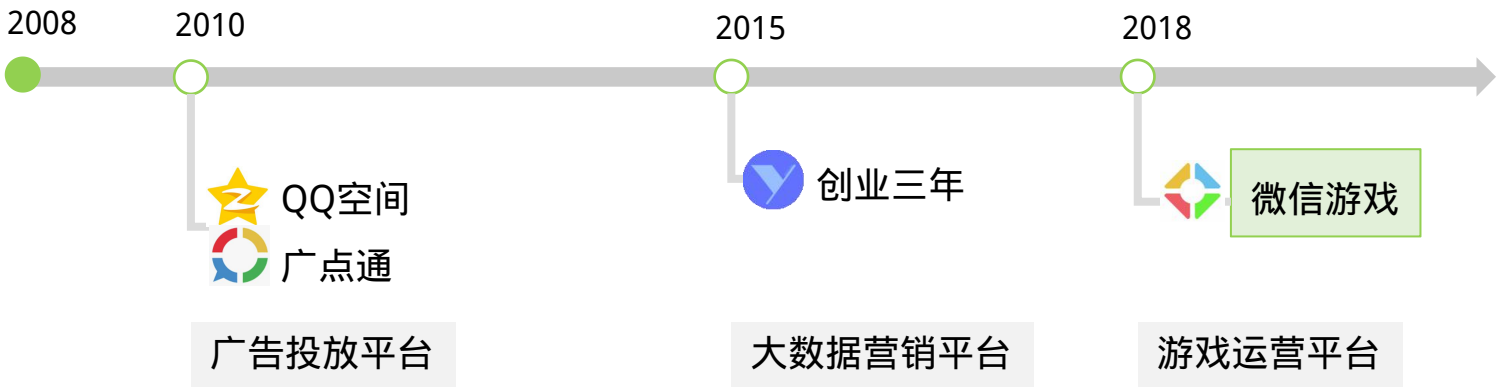
基于CLS的B端提效实践

彭自强 2019-11

关于我

damonpeng@qq.com

大魔@微信游戏



一句话概括...

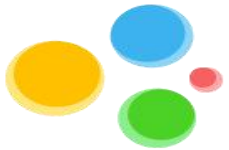
VS

业务诉求 & 个人诉求：

如何快速支撑业务，不做流水线工人？

探索和实践

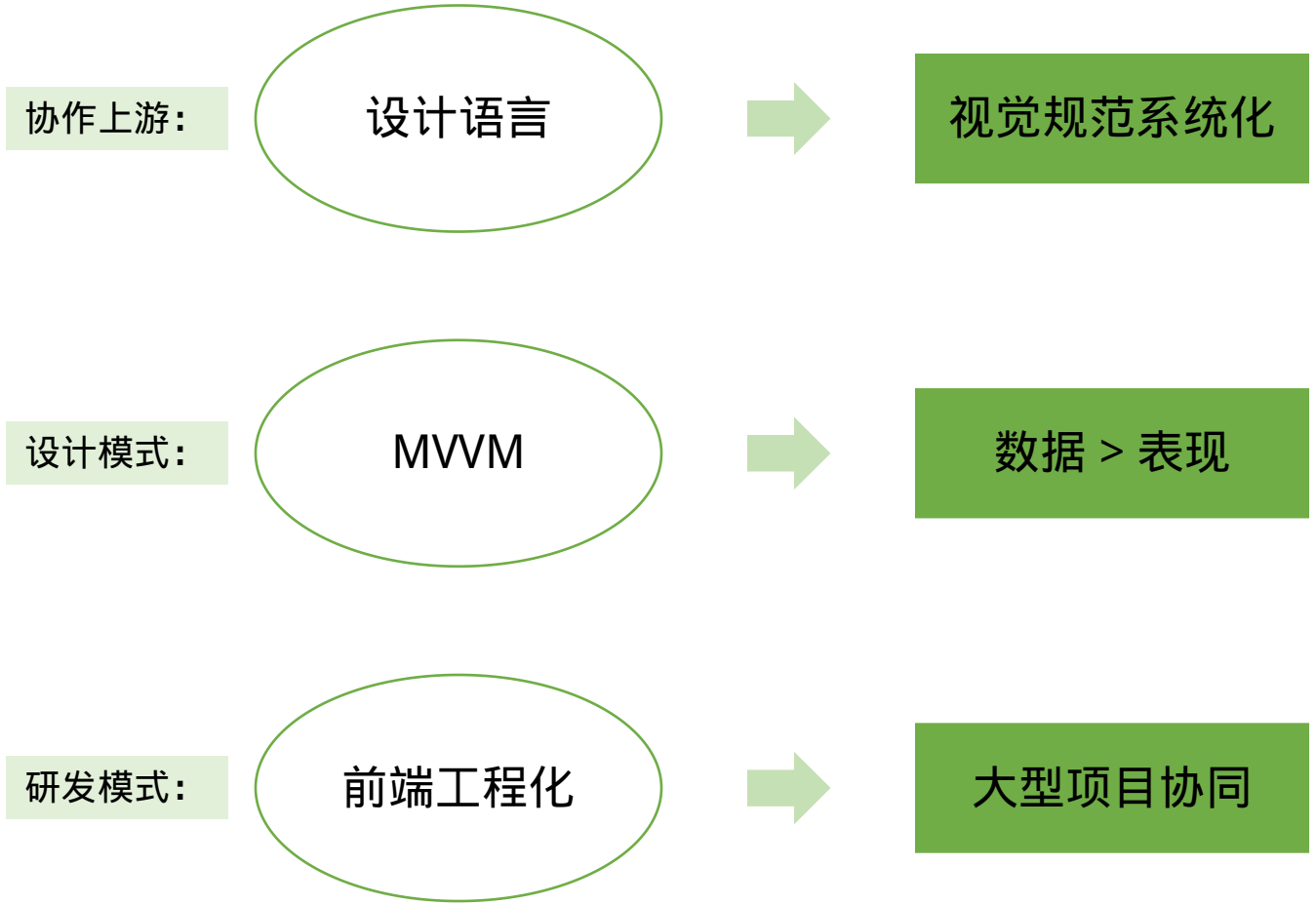
1. B端系统开发的特点
2. 通用组件语言规范
3. 解放前端
4. 解放产品
5. 解放后台
6. 展望



1

B端系统开发的特点：破局

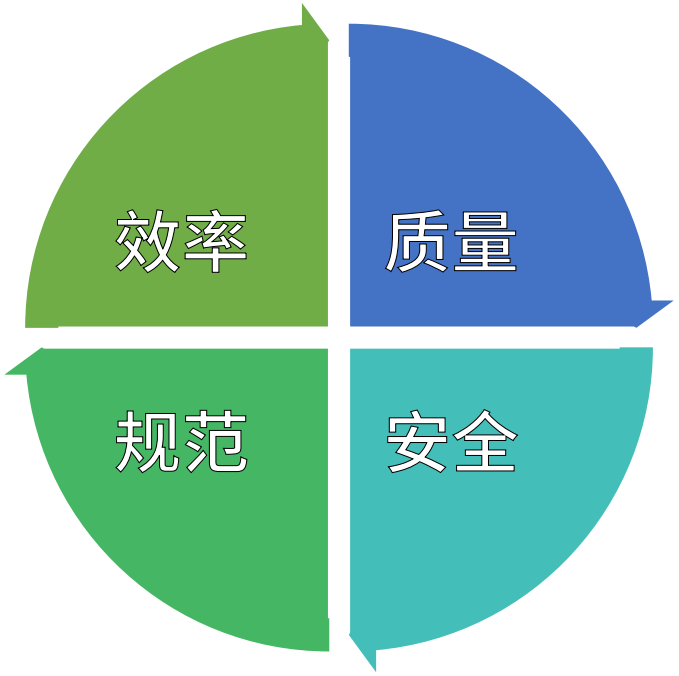
外因：行业的发展



内因：B端前端的特点

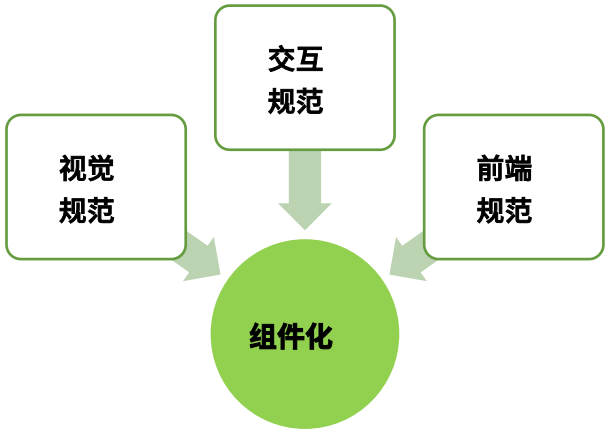
聚焦体系设计，专注逻辑编码

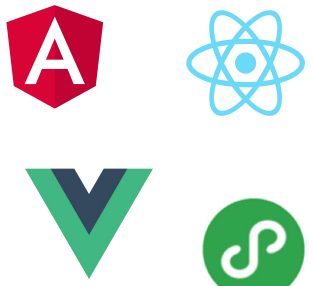
B端系统的目标



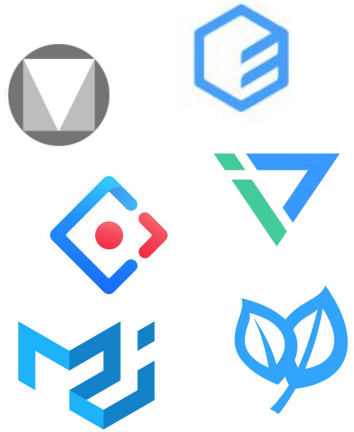
提升效率，降低成本，成为基础要求

提效的一般方式





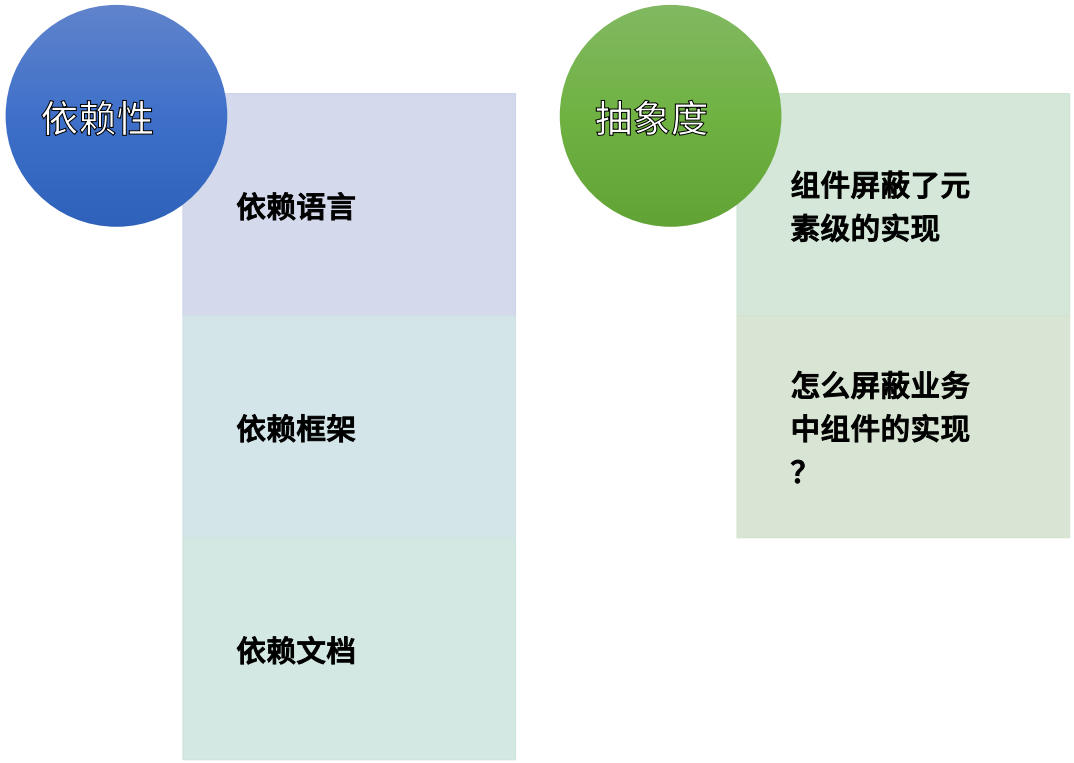
不同基础框架



不同组件库

不同应用场景

有什么问题？



业务需求并不需要关注组件实现，
真实场景可能是：

PM：这里有一个搜索框，点击查询按钮后，出来结果列表。

FE：需要翻页吗？

PM：要

（而且要求快！）

PM：很简单吧，下午给我

B端常见需求

WeChat Game Center

需求场景：

实现形式：

信息录入

表单

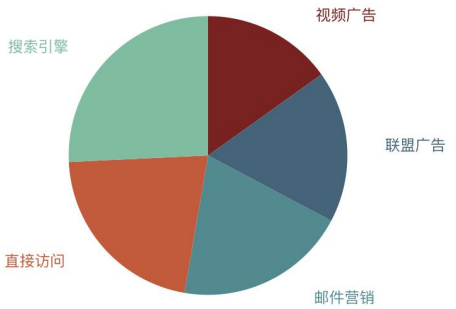
信息呈现

版本号	命名
10.15	Catalina
10.14	Mojave
10.13	High Sierra
10.12	Sierra

共4条 首页 1 / 1 尾页 跳转

列表

信息可视化



图表

形态相对比较固化

页面代码的一般构成

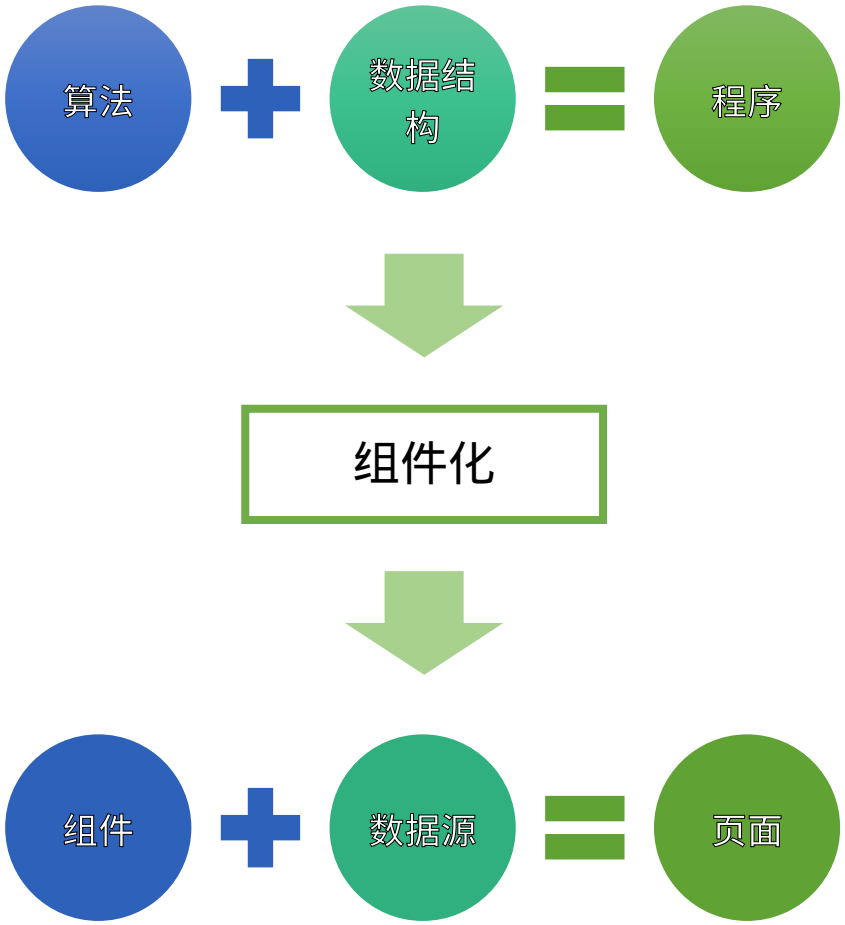

```vue
<template>
    <div>
        <div :class="userClass">{{user.name}}</div>
    </div>
</template>

<script>
import request from 'axios';

export default {
    data() {
        return {
            user: { name: 'damon'},
            userClass: ''
        }
    },

    created() {
        this.init();
    },

    methods: {
        init() {
            request.get('/cgi-bin/userinfo').then(response => {
                this.user = response.user;
            });
        }
    }
}
</script>
```

界面

数据

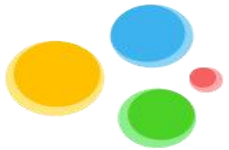
逻辑

以vuejs为例



弱化表现

聚焦驱动组件的数据结构



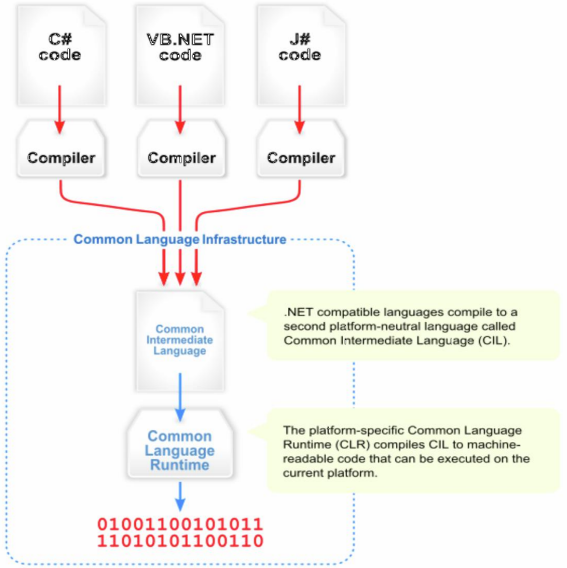
2

通用组件语言规范：CLS

Inspired by

.NET Framework

Common Language Specification



SQL [编辑]

维基百科，自由的百科全书

SQL（/ˈɛs kjuː ˈɛl/[4]或/ˈsiːkwəl/[5]，Structured Query Language:结构化查询语言

Common Components Language Specification

通用组件语言规范

将组件抽象为**统一模型**的一种标准的描述方式，以**JSON**的形式呈现。

设计原则

数据驱动

- UI 是数据
- 基于组件描述页面
- 以数据结构描述组件

组件封装表现，逻辑关注数据

开发者友好

- 低门槛，学习一个组件语言，就学会了整个组件库的使用
- 易理解，易记忆，易使用

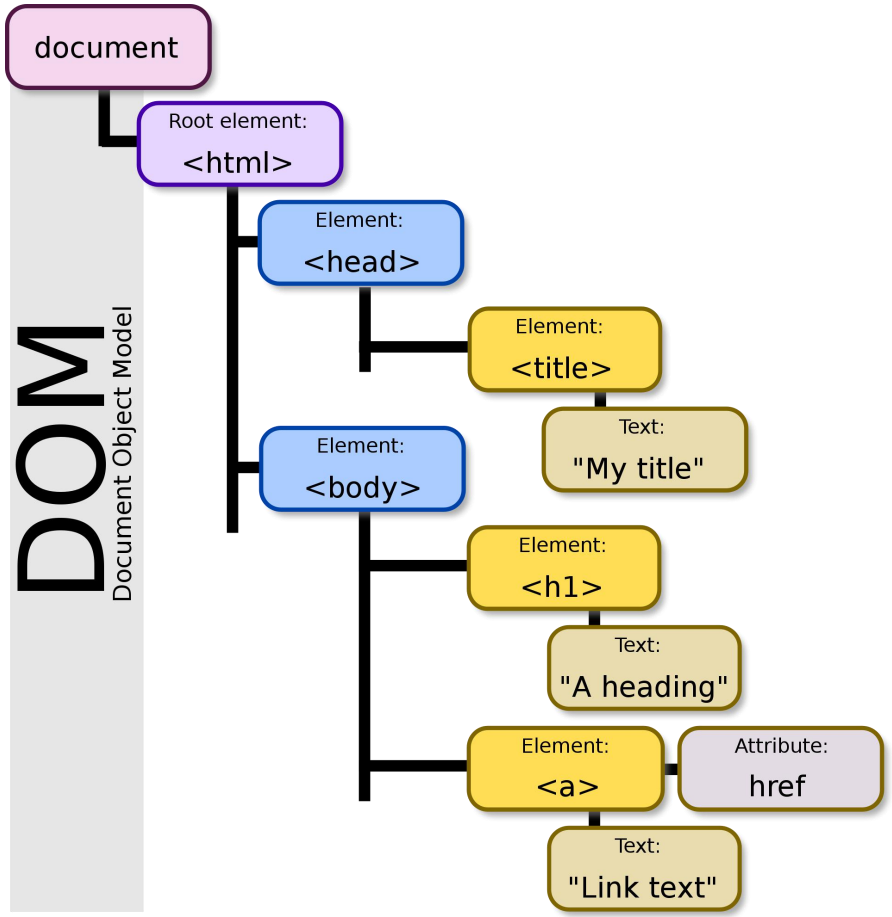
快速上手，快速开发

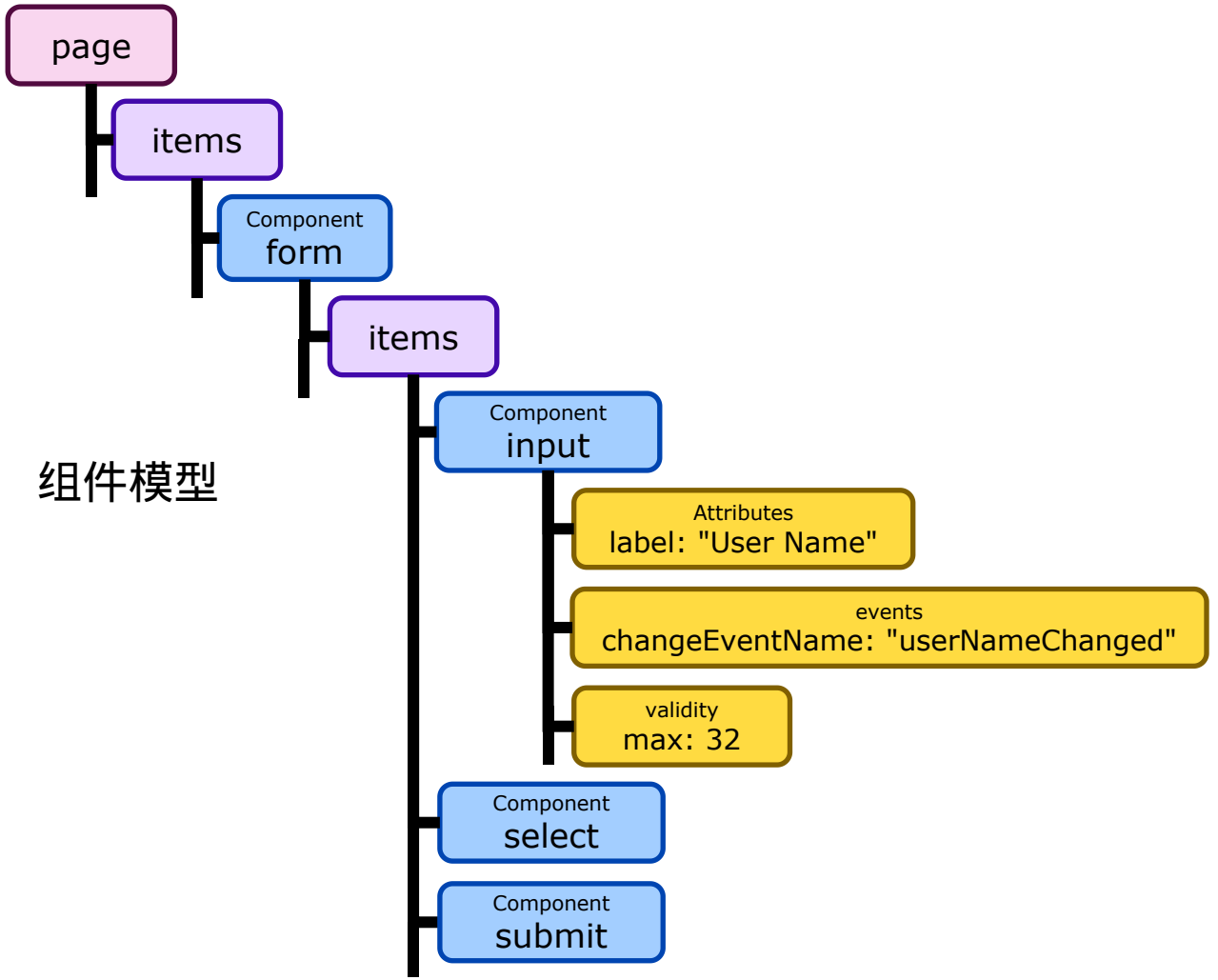
语义化

- 符合一般人对功能的认知
- 合适的场景，合适的支持

指导组件开发者

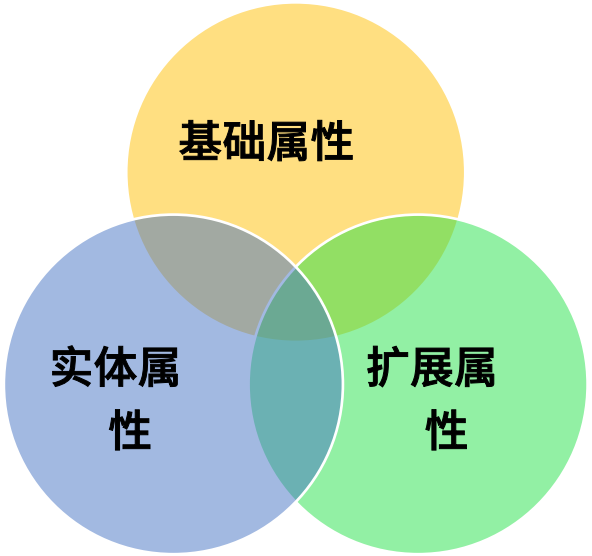
组件统一模型抽象





组件模型

CLS 主要属性定义



名称定义

参数名	类型	必填	默认值	说明
id	String	是	''	唯一标识，用于区分每一个组件实例
name	String	否	''	数据标识，用于提交到服务端
label	String	否	''	若传入String，则显示为纯文本
component	String	否	-	组件类型，取值依赖组件的实现
value	String/Array/Object	否	-	组件取值，具体类型依赖validity.format 决定

实体定义

名称	类型	必填	默认值	说明
attributes	Attributes	否	-	组件属性，详见 Attributes 定义
events	Events	否	-	组件触发的事件，详见 Events 定义
items	Array	否	-	下个层级，嵌套的树状结构
validity	Object	否	-	数据有效性校验，详见 Validity 定义
extra	Object	否	-	扩展属性，透传

Animate.css

JSON Schema

https://weadmin.github.io/WeComponentsDocs/#/doc/principle/cls

CLS 主要属性定义

Attributes 定义

名称	类型	必填	默认值	说明
class	String	否	-	组件容器的样式名
style	String	否	-	组件容器的style
placeholder	String	否	''	默认占位文案
defaults	Array	否	-	默认展示，如items的默认展示
hide	Boolean	否	false	是否隐藏
multiple	Boolean	否	false	是否items子项可多选
disabled	Boolean	组件是否禁用编辑		
readonly	Boolean	否	false	是否只读
repeated	Boolean	否	false	是否组件子项为可重复的
sortable	Boolean	否	false	是否支持对items子项排序
searchable	Boolean	是否支持对items子项搜索		
resetable	Boolean	否	false	是否支持清空value

Events 定义

名称	参数	必填	默认值	说明
click	Function	否	onClick	抛出当前组件的点击事件
dblClick	Function	否	onDblClick	抛出当前组件的双击事件
mouseEnter	Function	否	onMouseEnter	抛出当前组件的鼠标移入事件
mouseLeave	Function	否	onMouseLeave	抛出当前组件的鼠标移出事件
change	Function	否	onChange	抛出当前组件的值变化
dragEnd	Function	否	onDragEnd	抛出当前组件的拖拽结束事件

Validity 定义

名称	类型	必填	默认值	说明
required	String	否	false	组件值是否必填
format	String	否	-	组件值的数据类型，包含String/Number/Boolean
min	Number	否	-	最小值/最小长度
max	Number	否	-	最大值/最大长度
pattern	String	否	-	正则表达式

https://weadmin.github.io/WeComponentsDocs/#/doc/principle/cls

CLS主要方法定义

方法名	说明
getValue(name)	获取组件的value取值，并结合validity的format字段，返回对应数据类型的value
setValue(name, value)	设置组件的value取值，value入参需结合validity的format字段，传入对应的数据类型
getAttribute(name)	获取组件属性
setAttribute(name, value)	设置组件属性
getAttributes(name)	获取组件属性
setAttributes(name, values)	设置组件属性
show(id)	展示指定组件
hide(id)	隐藏制定组件
getItem(id)	获取组件和其完整的配置
removeItem(id)	删除组件
trigger(eventData)	触发对应事件

https://weadmin.github.io/WeComponentsDocs/#/doc/principle/cls

定位

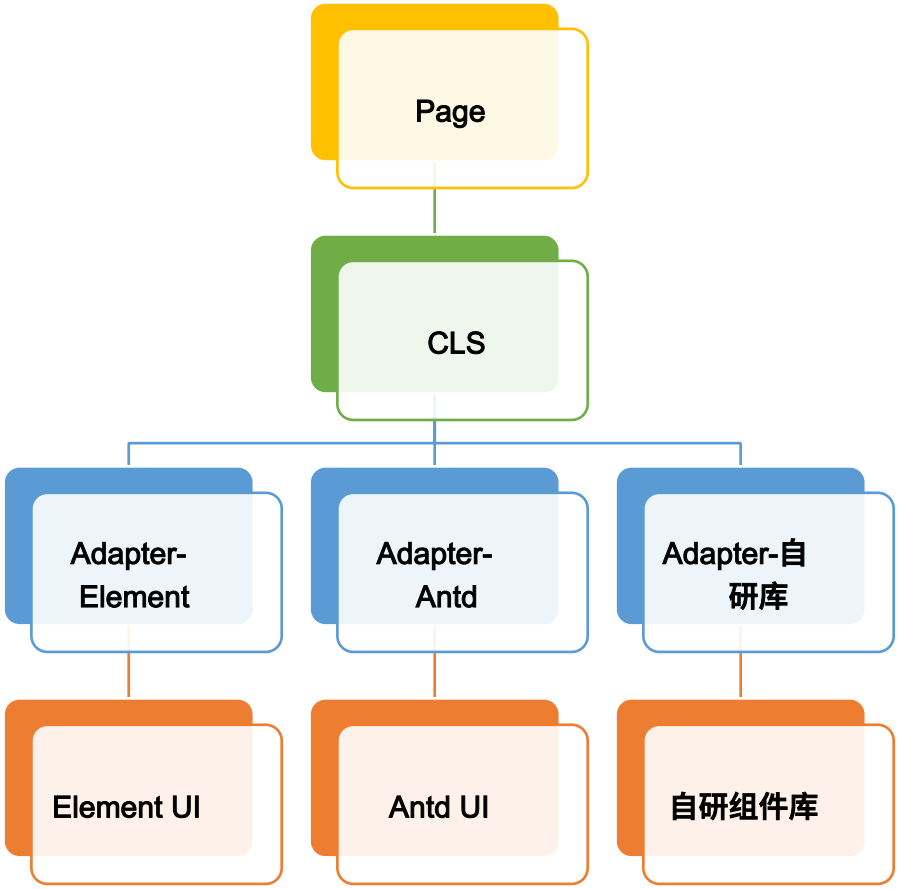
	SwiftUI	Flutter	CLS（通用组件语言规范）
编程范式	声明式	声明式	声明式
语言	Swift	Dart	JSON
产物	框架&组件库	框架&组件库	只是规范
平台	Apple APP	所有APP/Web	跨端，依赖实现

我们采用Vue.js实现了一套基于CLS的组件库

WeComponents

https://github.com/Tencent/WeComponents

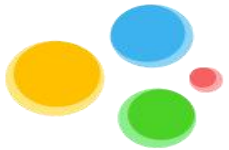
现有组件库的对接



组件规范　JSON

适配层

现有组件库　模版、属性和方法



3

解放前端：写完 JSON 就做好了页面

例子 —— 表单


```
{
    component: 'container',
    label: '2017TGA年度游戏登记',
    attributes: {
        layout: 'column',
        flexJustify: 'space-around',
        flexWrap: 'wrap'
    },
    items: [
        {
            component: 'input',
            label: '游戏名称',
            value: '塞尔达传说 荒野之息',
            validity: {
                required: true
            }
        },
        {
            component: 'input',
            label: '评分',
            value: 200,
            validity: {
                format: "Number",
                min: 0,
                max: 100
            }
        },
        {
            component: 'calendar',
            label: '发布日期',
            value: '2017-03-03'
        },
        {
            component: 'button',
            attributes: {
                buttonType: 'primary',
                clickEventName: "submit"
            },
            label: '提交'
        }
    ]
},
```

例子 —— 列表


```
{
    component: 'table',
    label: '示例表格',
    attributes: {
        showIndex: true,
        selectable: true,
        wrap: false,
        filterItems: [],
        exportData: true
    },
    items: [
        {
            name: 'appname',
            label: '游戏名称',
            attributes: {
                width: 100
            }
        },
        {
            name: 'price',
            label: '价格',
            attributes: {
                sort: true
            }
        },
        {
            name: 'score',
            label: '评分'
        },
        {
            name: 'introduce',
            label: '游戏介绍'
        },
    ],
```

示例表格

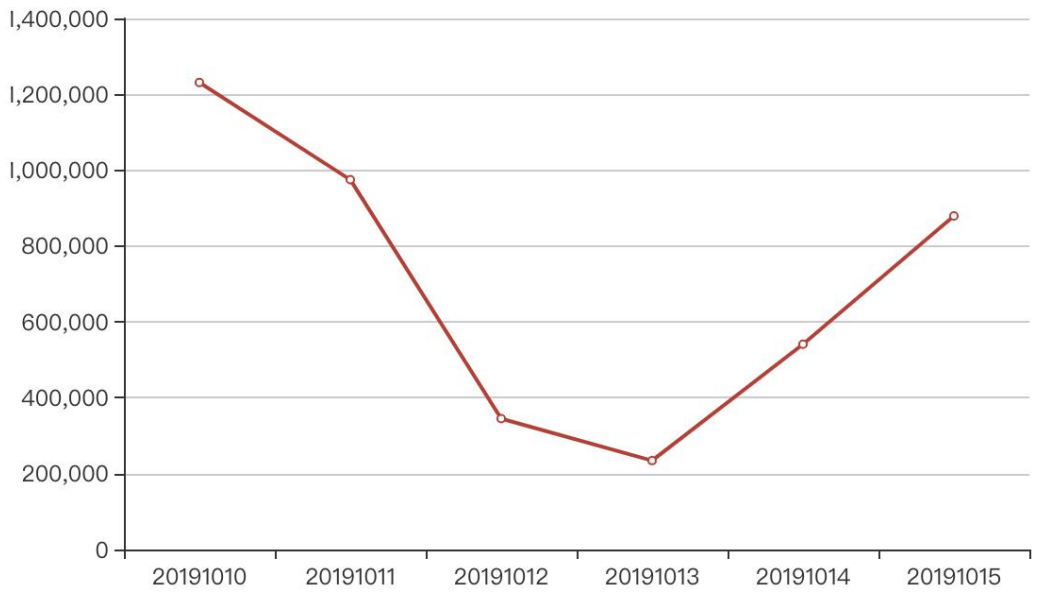
筛选表格字段　导出excel

#	游戏名称	价格	评分	游戏介绍
1	结果游戏	648	99	搜索结果项
2	结果游戏	648	99	搜索结果项
3	结果游戏	648	99	搜索结果项

```
value: [
    {
        appname: '怪物猎人世界',
        price: 388,
        score: 97,
        introduce: '猛汉王'
    },
    {
        appname: '尼尔机械纪元',
        price: 528,
        score: 98,
        introduce: '废土世界的人工智能'
    },
    {
        appname: '黑暗之魂3',
        price: 88,
        score: 95,
        introduce: '卑鄙的外乡人之旅'
    }
]
```

例子 —— 图表


```
{
    component: "line",
    items: [
        {
            label: "日期",
            name: "date",
            attributes: {
                perspective: "xAxis"
            }
        },
        {
            label: "曝光",
            name: "pv",
            attributes: {
                perspective: "yAxis"
            }
        },
        {
            label: "点击",
            name: "click"
        }
    ],
    value: [
        { date: 20191010, pv: 1231231, click: 31331 },
        { date: 20191011, pv: 975123, click: 65234 },
        { date: 20191012, pv: 345123, click: 12321 },
        { date: 20191013, pv: 234561, click: 12117 },
        { date: 20191014, pv: 541234, click: 45678 },
        { date: 20191015, pv: 879452, click: 764322 }
    ]
},
```



声明式编程 —— 像产品提需求一样写代码

WeChat Game Center

回到这个例子。

PM：这里有一个搜索框，点击查询按钮后，出来结果列表。

FE：需要翻页吗？

PM：要

三分钟后……

业务需求并不需要关注组件实现

FE：做完了

```
{
    component: 'form',
    attributes: { layout: 'row' },
    items: [
        {
            name: 'search',
            label: '搜索',
            component: 'input',
            attributes: { placeholder: '输入游戏名称进行搜索' }
        },
        {
            label: '查询',
            component: 'submit',
            attributes: { type: 'primary', submitEventName: 'searchTable' }
        }
    ]
},
```

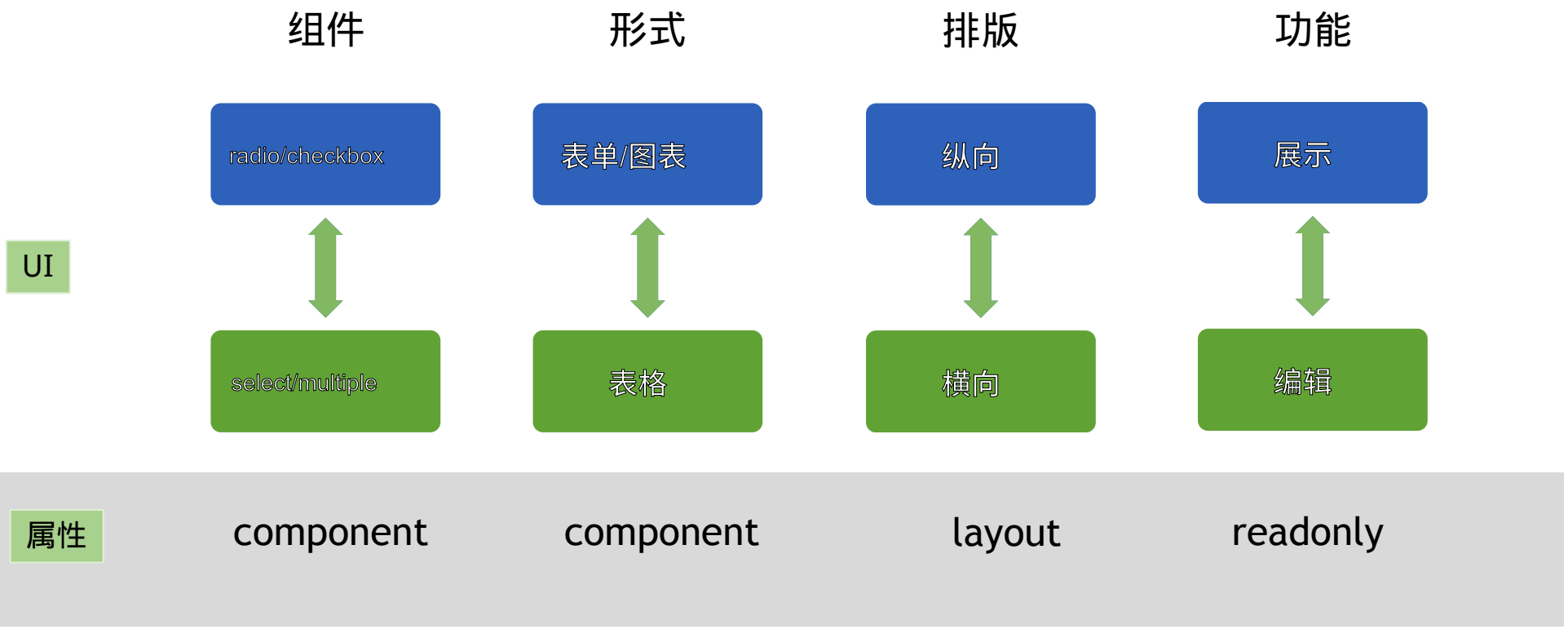
```
component: 'table'
attributes: { placeholder: '暂无数据', pagination: 'default' },
items: [
    {
        name: 'icon',
        label: '游戏图标',
        attributes: { width: 60, textAlign: 'center' },
        value(v, row) {
            return `<img src="${v}" width="28" />`;
        }
    },
    { name: 'name', label: '游戏名称' },
    {
        name: 'size',
        label: '大小',
        attributes: { textAlign: 'right' },
        value(v, row) {
            return `${(v/1000/1000/1000).toFixed(1)} G`;
        }
    },
    { name: 'intro', label: '简介' },
    {
        label: '操作',
        value() {
            return {
                component: 'container',
                items: [
                    {
                        label: '查看',
                        component: 'button',
                        attributes: {
                            type: 'link', clickEventName: 'checkDetails'
                        }
                    }
                ]
            };
        }
    }
],
```

搜索 输入游戏名称进行搜索 查询

游戏图标	游戏名称	大小	简介	操作
	王者荣耀	3.7 G	爽快超神，腾讯5v5英雄公平对战手游	查看
	和平精英	3.9 G	大吉大利，腾讯光子自研军事竞赛体验	查看

共2条 首页 1 / 1 尾页 跳转

数据驱动 —— 数据结构一致下的表现变化



最佳实践1 —— 快速组件切换

WeChat Game Center


```
{
    name: 'front_version_control_type',
    label: '托管方式',
    component: 'radio',
    items: [
        { label: 'Git', value: 'Git' },
        { label: 'SVN', value: 'SVN' },
    ],
    value: 'Git',
    attributes: {
        help: '选择源码管理工具',
    },
    validity: {
        required: true,
        format: 'String'
    }
}
```

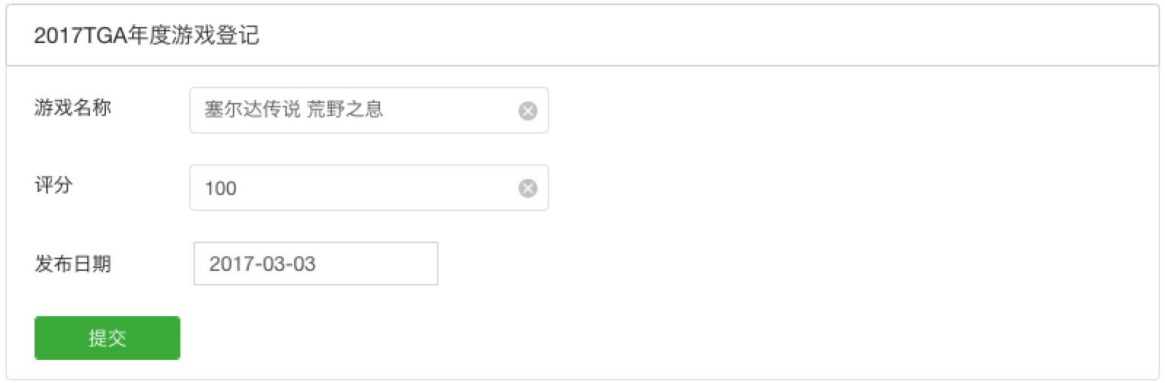
```
{
    name: 'front_version_control_type',
    label: '托管方式',
    component: 'select',
    items: [
        { label: 'Git', value: 'Git' },
        { label: 'SVN', value: 'SVN' },
    ],
    value: 'Git',
    attributes: {
        help: '选择源码管理工具',
    },
    validity: {
        required: true,
        format: 'String'
    }
}
```

最佳实践2 —— 组合的变换

WeChat Game Center

传统表单

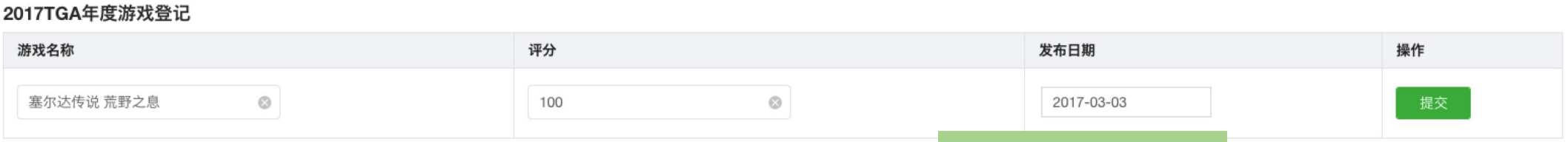
形式变化



component: form

attribtues.layout: row

类excel



component: table

功能变化

查询框

attribtues.layout : row

最佳实践3 —— 表格和图表

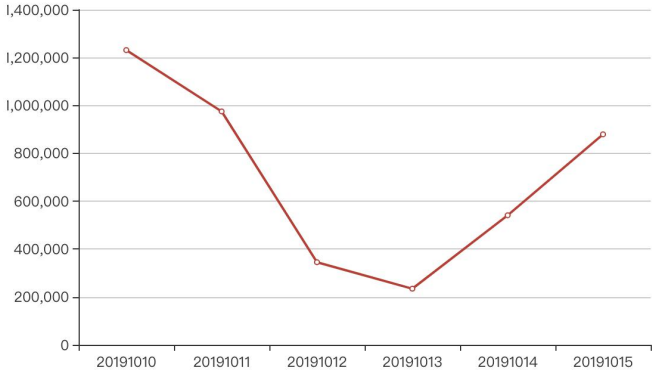
WeChat Game Center

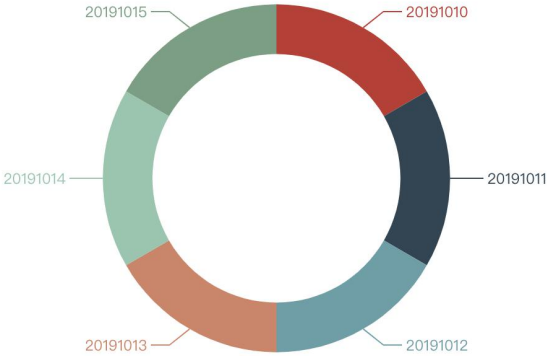
```
{
    component: "table",
    attributes: {
        type: "ring"
    },
    items: [
        {
            label: "日期",
            name: "date",
            attributes: {
            }
        },
        {
            label: "曝光",
            name: "pv",
            attributes: {
                textAlign: "right"
            }
        },
        {
            label: "点击",
            name: "click",
            attributes: {
                textAlign: "right"
            }
        }
    ],
    value: [
        { date: 20191010, pv: 1231231, click: 31331 },
        { date: 20191011, pv: 975123, click: 65234 },
        { date: 20191012, pv: 345123, click: 12321 },
        { date: 20191013, pv: 234561, click: 12117 },
        { date: 20191014, pv: 541234, click: 45678 },
        { date: 20191015, pv: 879452, click: 764322 }
    ]
},
```

```
{
    component: "line",
    items: [
        {
            label: "日期",
            name: "date",
            attributes: {
                perspective: "xAxis"
            }
        },
        {
            label: "曝光",
            name: "pv",
            attributes: {
                perspective: "yAxis"
            }
        },
        {
            label: "点击",
            name: "click"
        }
    ],
    value: [
        { date: 20191010, pv: 1231231, click: 31331 },
        { date: 20191011, pv: 975123, click: 65234 },
        { date: 20191012, pv: 345123, click: 12321 },
        { date: 20191013, pv: 234561, click: 12117 },
        { date: 20191014, pv: 541234, click: 45678 },
        { date: 20191015, pv: 879452, click: 764322 }
    ]
},
```

```
{
    component: "ring"
    items: [
        {
            label: "日期",
            name: "date",
            attributes: {
                perspective: "xAxis"
            }
        },
        {
            label: "曝光",
            name: "pv",
            attributes: {
                perspective: "yAxis"
            }
        },
        {
            label: "点击",
            name: "click"
        }
    ],
    value: [
        { date: 20191010, pv: 1231231, click: 31331 },
        { date: 20191011, pv: 975123, click: 65234 },
        { date: 20191012, pv: 345123, click: 12321 },
        { date: 20191013, pv: 234561, click: 12117 },
        { date: 20191014, pv: 541234, click: 45678 },
        { date: 20191015, pv: 879452, click: 764322 }
    ]
},
```

日期	曝光	点击
20191010	1231231	31331
20191011	975123	65234
20191012	345123	12321
20191013	234561	12117
20191014	541234	45678
20191015	879452	764322

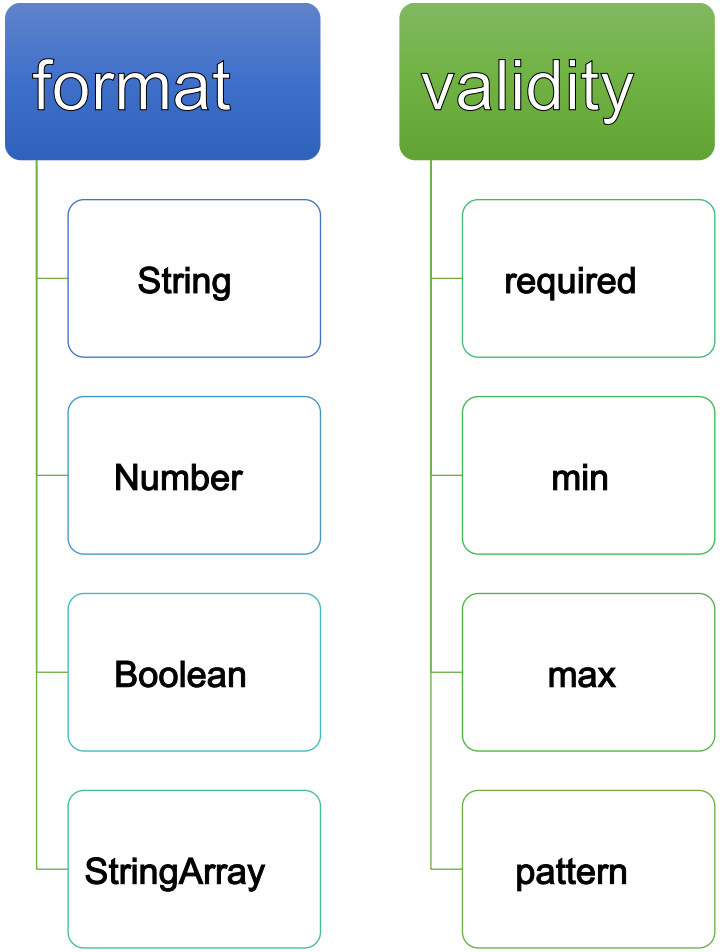




最佳实践3 —— 弥补泛类型的不足

声明式解决前后台参数的严格类型问题

```
validity: {
    required: true,
    format: 'String'
}
```



基于 CLS 的运作流程


```json
{
    "id": "user_form",
    "name": "user_form",
    "component": "form",
    "items": [
        {
            "id": "user_name",
            "name": "user_name",
            "label": "用户名",
            "component": "input",
            "attributes": {
                "placeholder": "你的名号",
                "help": "可输入中文、英文、下划线"
            },
            "validity": {
                "required": true,
                "format": "String",
                "maxLength": 10
            }
        },
        {
            "id": "submit",
            "name": "submit",
            "label": "提交",
            "component": "button",
            "attributes": {
                "class": "button-primary",
                "loading": false
            }
        }
    ]
}
```

```json
{
    "id": "version_list",
    "name": "version_list",
    "label": "MacOS 版本命名表",
    "component": "table",
    "attributes": {
        "pageSize": 15
    },
    "items": [
        {
            "name": "version_code",
            "label": "版本号",
            "attributes": {
                "width": 100
            }
        },
        {
            "name": "version_name",
            "label": "命名"
        }
    ],
    "value": [
        {
            "version_code": "10.15",
            "version_name": "Catalina"
        },
        {
            "version_code": "10.14",
            "version_name": "Mojave"
        },
        {
            "version_code": "10.13",
            "version_name": "High Sierra"
        },
        {
            "version_code": "10.12",
            "version_name": "Sierra"
        }
    ]
}
```

划重点

复杂度低
重复性高

业务代码

复杂度高
沉淀性好

解析框架

{ JSON }

- form
- table
- chart

解析引擎

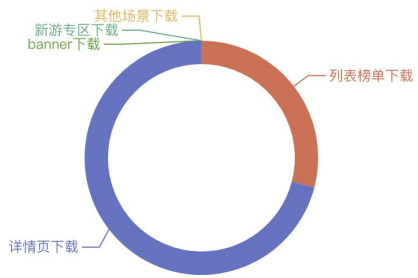
- 通用组件
- 业务组件
- 事件触发

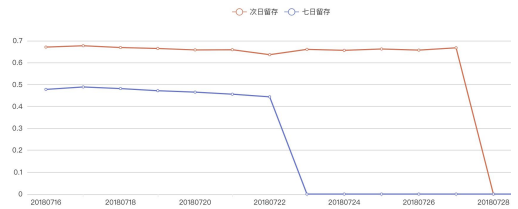
页面渲染

- 表单
- 列表
- 图表

版本号	命名
10.15	Catalina
10.14	Mojave
10.13	High Sierra
10.12	Sierra

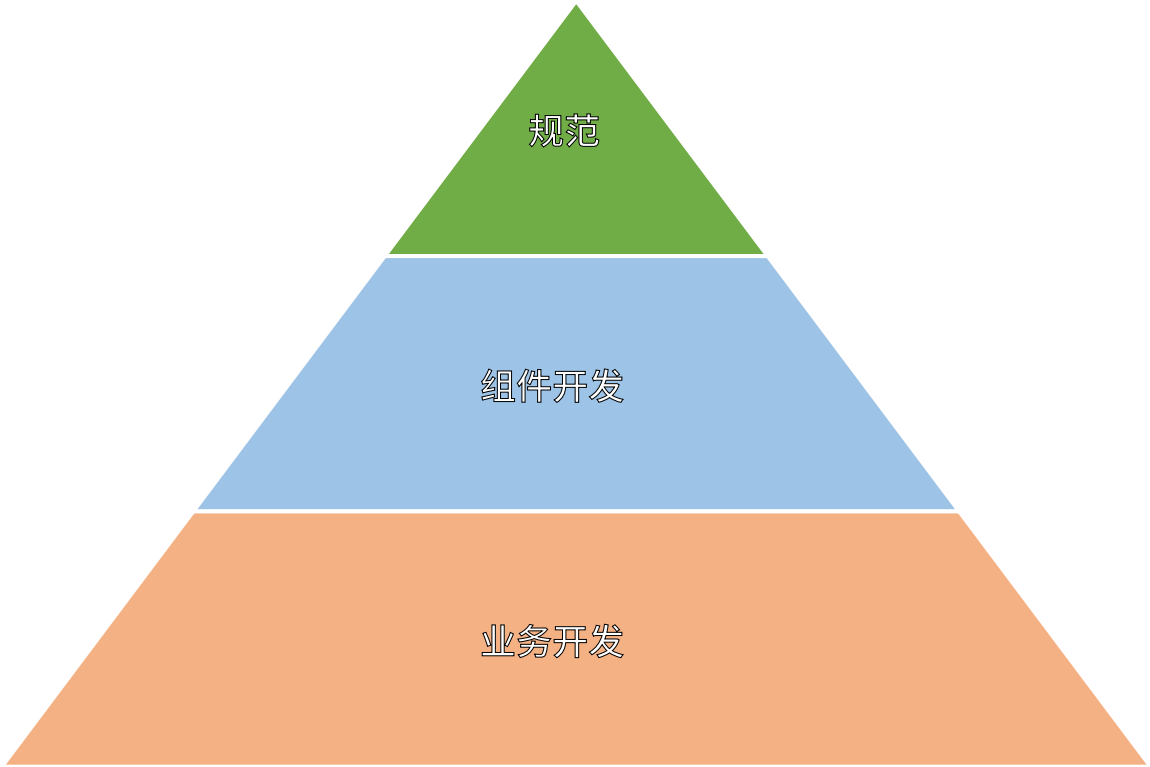
共4条 首页 1 / 1 尾页 跳转





写完JSON，就做好了页面

金字塔梯队



4

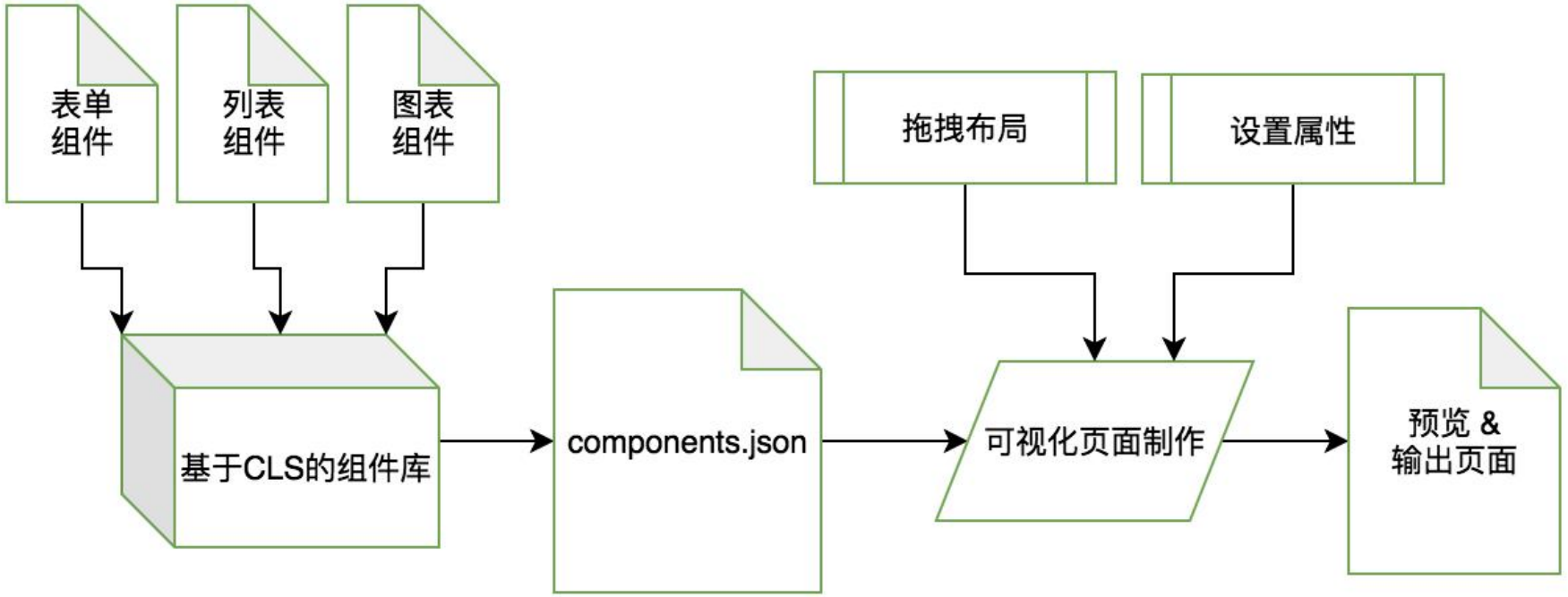
解放产品：可视化页面制作

可视化

写JSON也是很机
械式的工作……

Viz (CLS-based-Components) => Pages

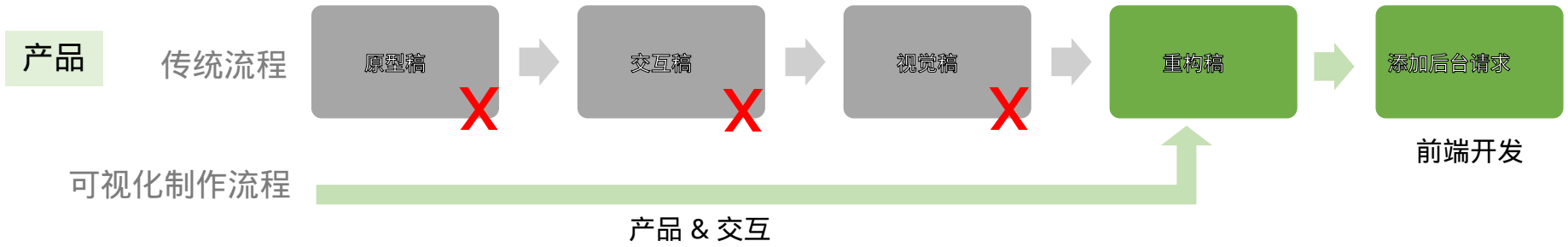
实现流程


表单
组件
列表
组件
图表
组件
拖拽布局
设置属性
基于CLS的组件库
components.json
可视化页面制作
预览 &
输出页面

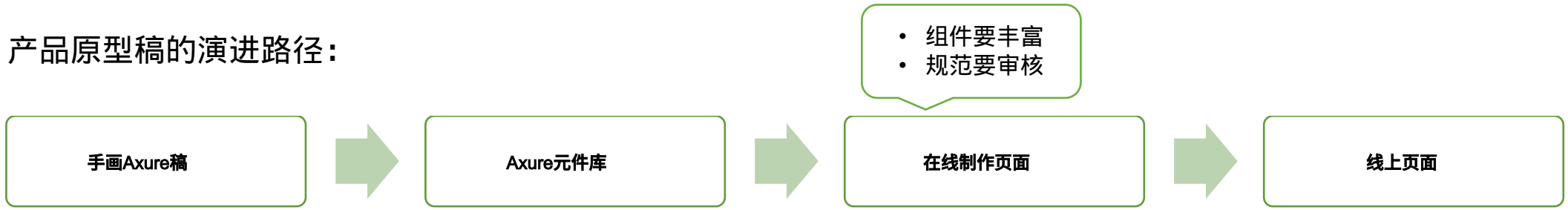
可视化页面制作工具

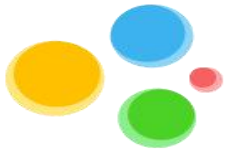
- 通用制作工具，可加载基于CLS的组件库
- 基于组件，不基于元素
- 可按模板生成代码

优化流程，非开发也能做页面



产品原型稿的演进路径：



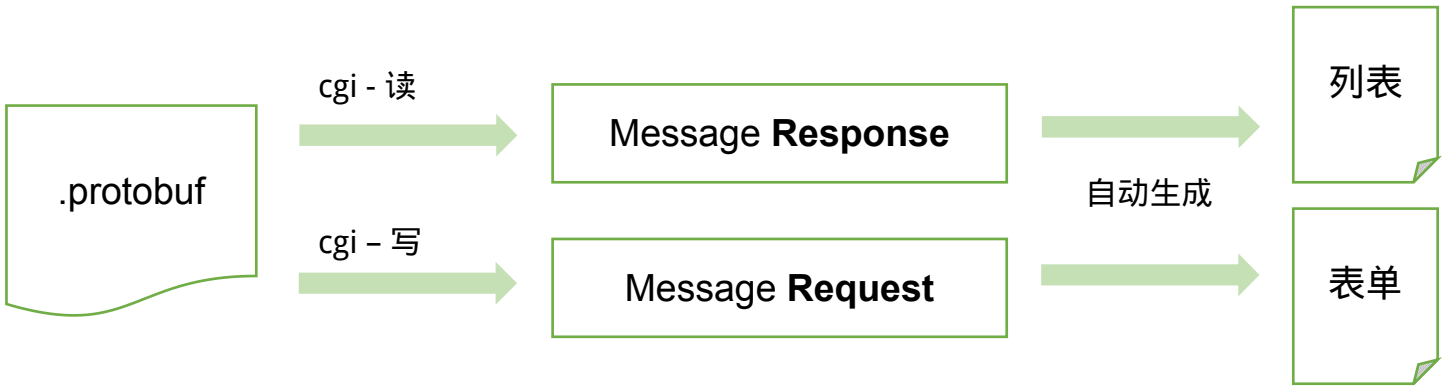


5

解放后台：自动生成

自动化，反射接口协议生成页面

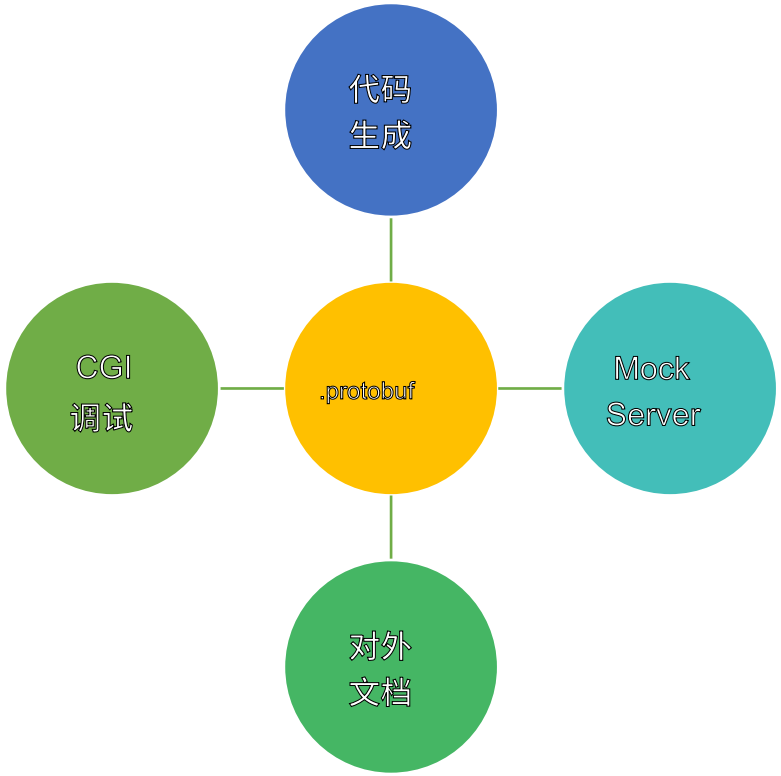
对接数据源：从pb协议反射出页面



- Message => container
- enum => select
- int/string => input
- Boolean => radio
- Repeated message => fieldset
- Repeated int/string => input with multiple attribute

面向协议编程：以结构化协议为核心


```
/**********************************
 * @message GetComponentsList
 * @path    /cgi-bin/getcomponentslist
 * @method  GET
 * @brief   获取组件列表
 */
message GetComponentsListRequest {
    required uint32 cur_page = 2;
    required uint32 per_page = 3;
    optional string appid    = 4;
}
message GetComponentsListResponse {
    required int32 code = 1;
    optional string message = 2;
    optional Result data = 3;

    message Result {
        required uint32 total_count = 1;
        repeated ComponentInfo list = 2;
    }
}
```

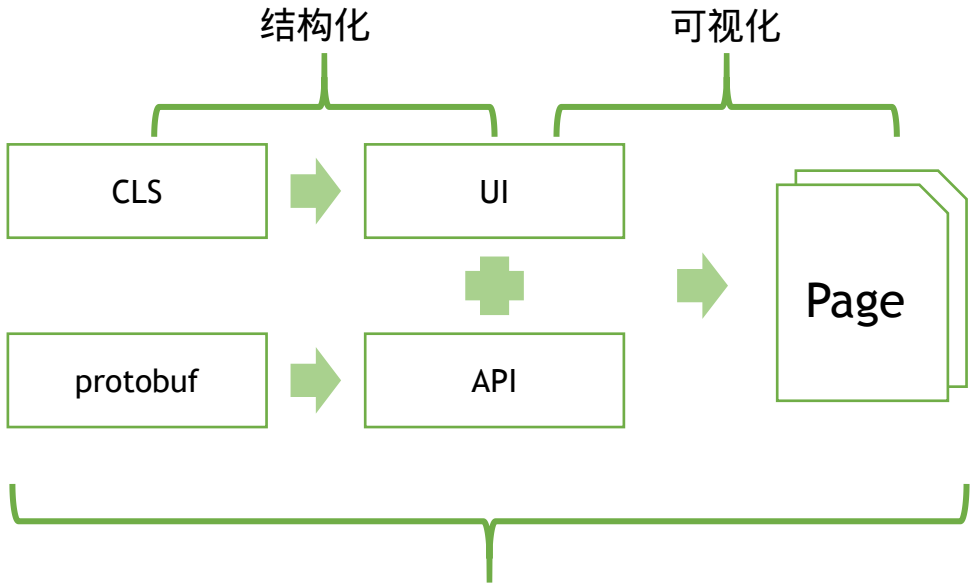


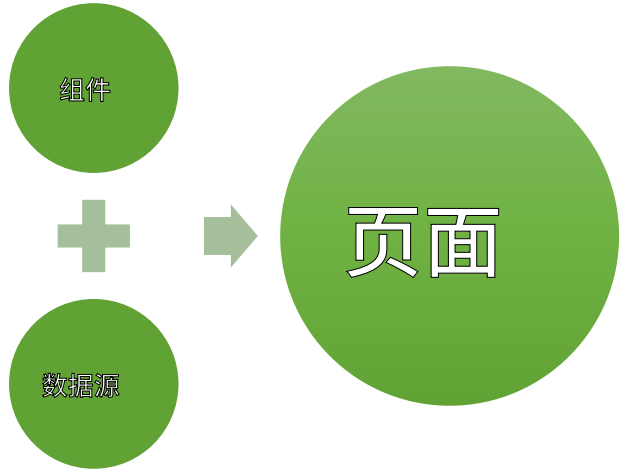
前提：

- 结构化的接口协议，如protobuf
- 结构化的注释

- 解析后台cgi协议，程序化理解请求和参数
- 可视化页面制作，直接绑定组件和cgi（即数据源）

提效总结





Viz (components) => Pages

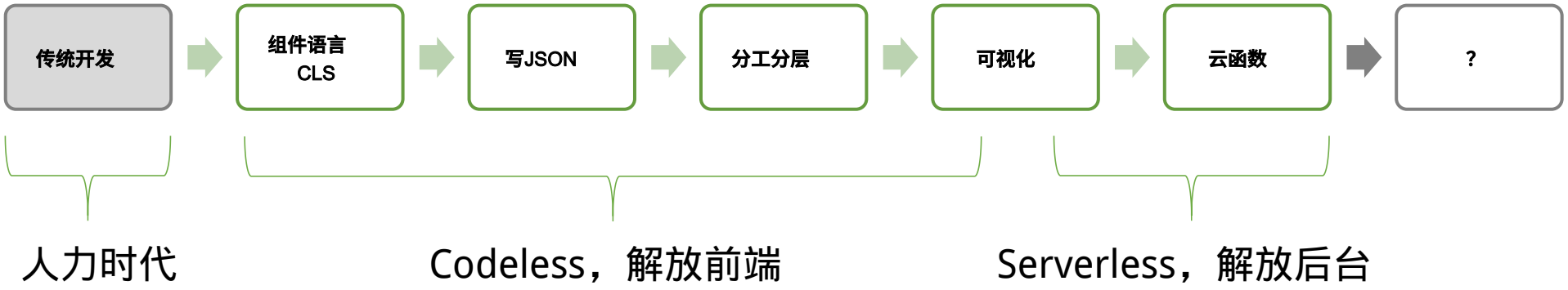
WeComponents

https://github.com/Tencent/WeComponents

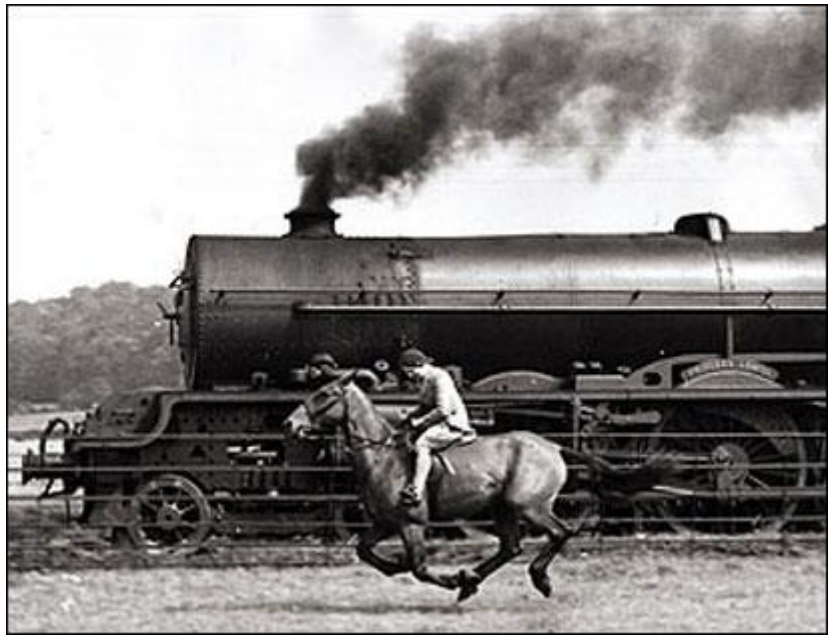
6

展望

演进



以规则、规范、工具实现熵减



工业革命是以机器取代人力，以大规模工厂化生产取代个体工场手工生产的一场生产与科技革命。

Thx.
大魔@微信游戏