

WEB前端的工业化时代

自我介绍

- ◎ 赵雨森
- ◎ 2014年加入网易
- ◎ 杭州研究院 - 云计算平台产品部 - 网易蜂巢
- ◎ rainfore@github, rainforest92@126.com
- ◎ [regular-ui](#), [pursuit-cli](#)

前端发展的几个阶段

前端发展的几个阶段

刀耕火种

WebPage, JSP/ASP/PHP

前端发展的几个阶段

刀耕火种

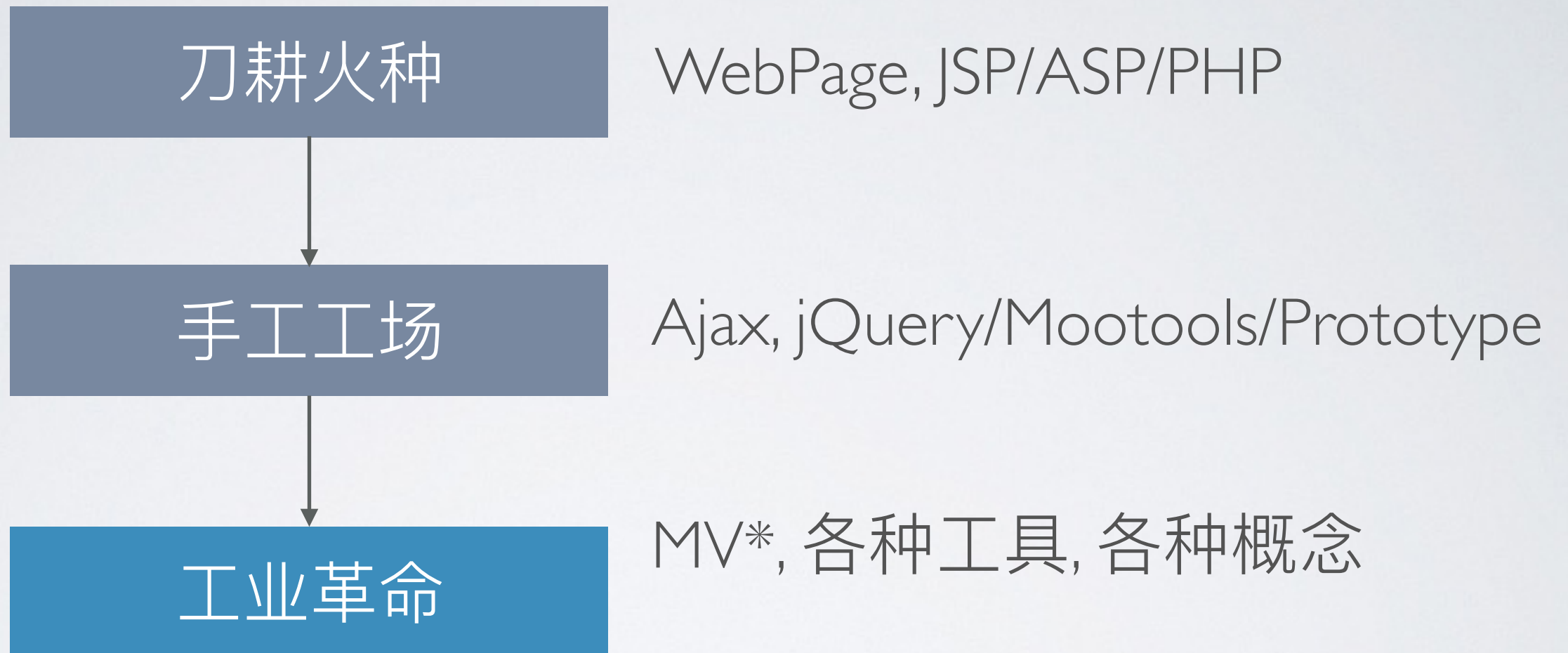
WebPage, JSP/ASP/PHP



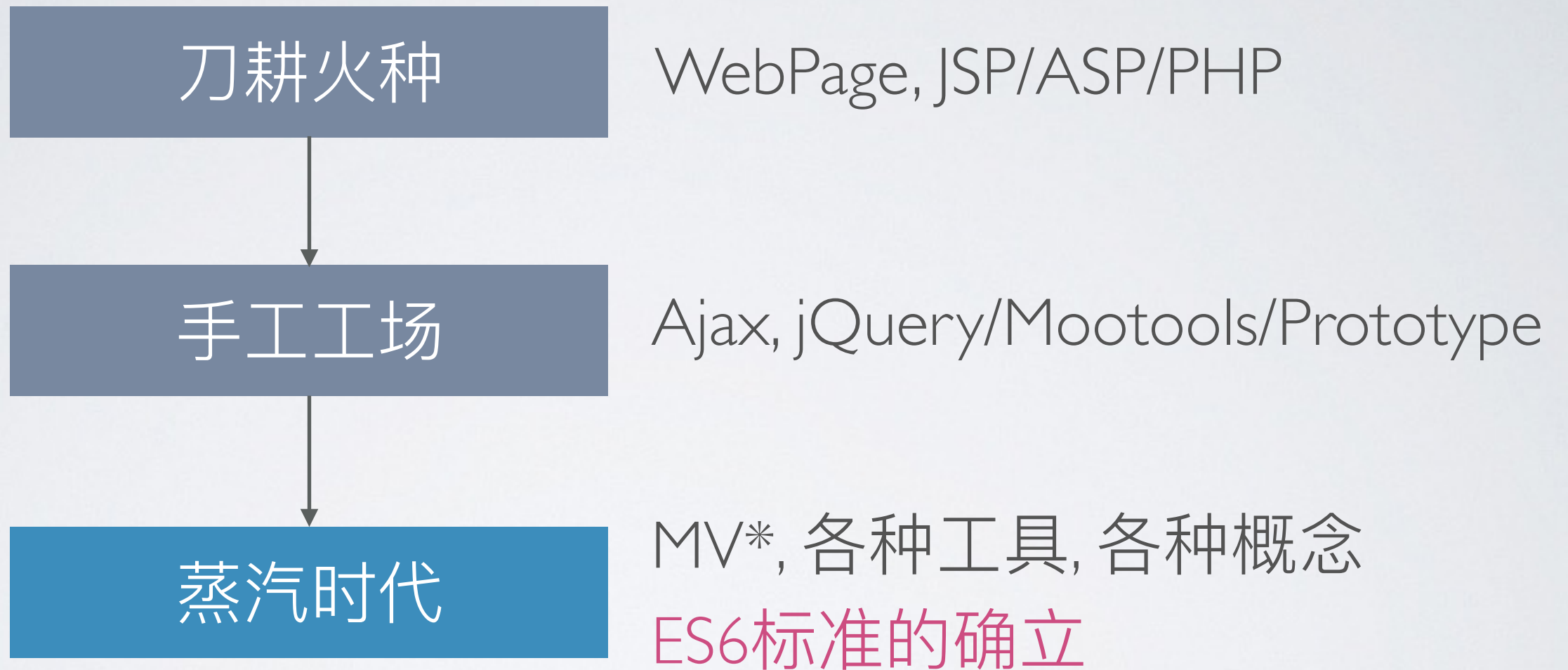
手工工场

Ajax, jQuery/Mootools/Prototype

前端发展的几个阶段



前端发展的几个阶段



前端发展的几个阶段



前端发展的几个阶段



工业化的四个方面

模块化

组件化

规范化

自动化

模块化

模块化的概念

将一个大文件拆分成相互依赖的小文件，再进行统一的拼装或加载。

JS的模块化

- ◎ Java有import
- ◎ C++有include
- ◎ Ruby有require
- ◎ CSS有@import
- ◎ JS没有

JS的模块化

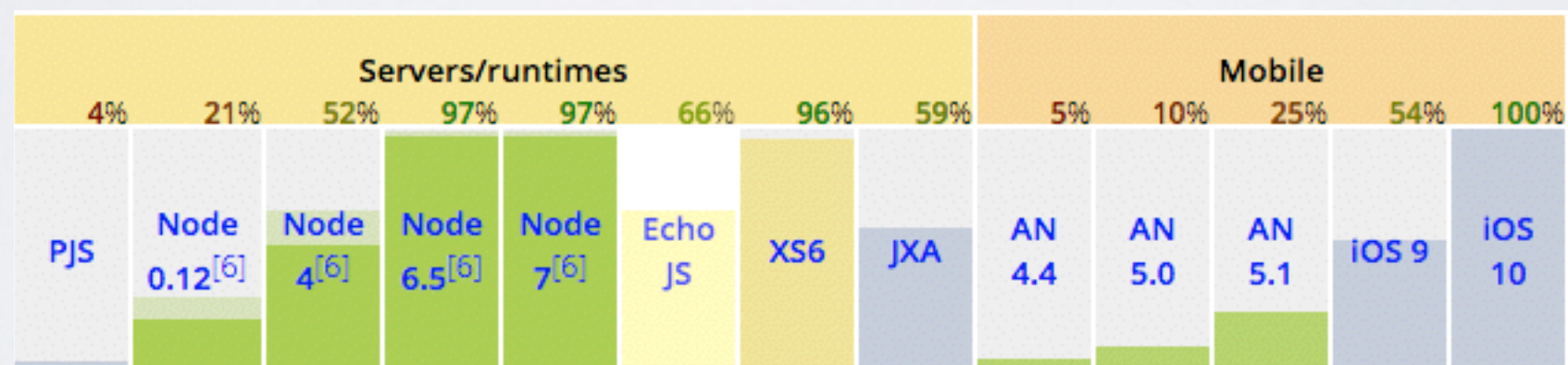
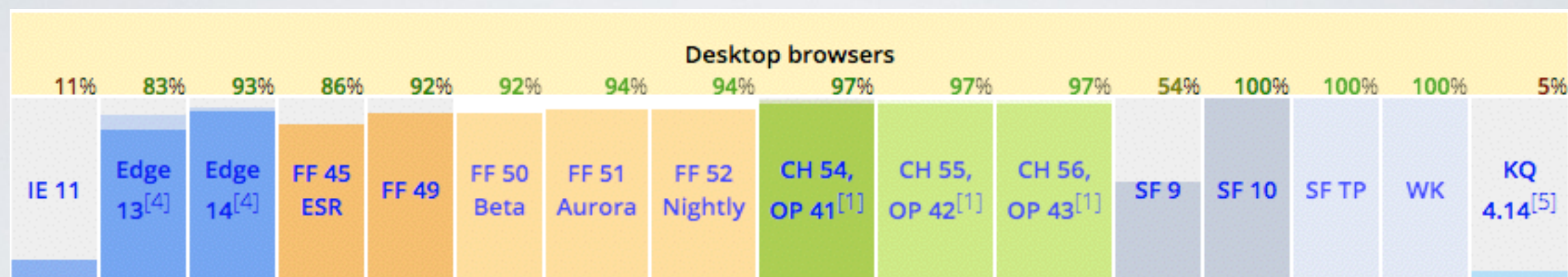
- CommonJS
- AMD
- CMD
- UMD

```
(function (root, factory) {  
    if (typeof exports === 'object')  
        module.exports = factory();  
    else if (typeof define === 'function' && define.amd)  
        define(factory);  
    else  
        root[library] = factory();  
})(this, function () {  
    //module ...  
});
```

ES6的模块体系

- import
- import ... as
- export
- export default

各平台对ES6的支持率



[combat-table](#)

兼容性处理

- 如果使用的是Babel：
 - IE6、7：就不用考虑了
 - IE8：
 - 使你的 React 应用兼容 IE8
 - export-all-loader(export * from 'xxx';)
 - IE9：只需吃个babel-polyfill即可

模块的打包与加载

- Webpack
- System.js

技术选型: Webpack/System.js + Babel + ES6

CSS的模块化

- ◎ SASS
- ◎ LESS
- ◎ Stylus
- ◎ MCSS

痛点

选择器的私有化问题（全局污染问题）

各厂的命名风格

- ◎ BEM风格；
- ◎ Bootstrap风格；
- ◎ Semantic UI风格；
- ◎ 我们公司的NEC风格；
- ◎ ...

各厂的命名风格

- ◎ BEM风格；
- ◎ Bootstrap风格；
- ◎ Semantic UI风格；
- ◎ 我们公司的NEC风格；
- ◎ ...

弱约束

“与其费尽心思地告诉别人要遵守某种规则，以规避某种痛苦，倒不如从工具层面就消灭这种痛苦。”

——知乎段子手

工具层面的三种解决方案

- Shadow DOM
- CSS in JS
- CSS Modules

技术选型: PostCSS + CSS Modules

组件化

1. 组件化 \neq 模块化

- 模块化是语言层面的
- 组件化是设计层面的

组件

每个包含模板(HTML)+样式(CSS)+逻辑(JS)功能完备的结构单元，我们称之为**组件**。

组件化要解决的问题

- 组件封装
- 逻辑 (JS) 继承
- 样式 (CSS) 扩展
- 模板 (嵌套) 嵌套

组件化要解决的问题


- 组件封装
- 逻辑 (JS) 继承
- 样式 (CSS) 扩展
- 模板 (嵌套) 嵌套

2. 组件化是对面向对象的更高级的抽象

组件之间的关系

- 继承
- 扩展
- 嵌套
- 包含

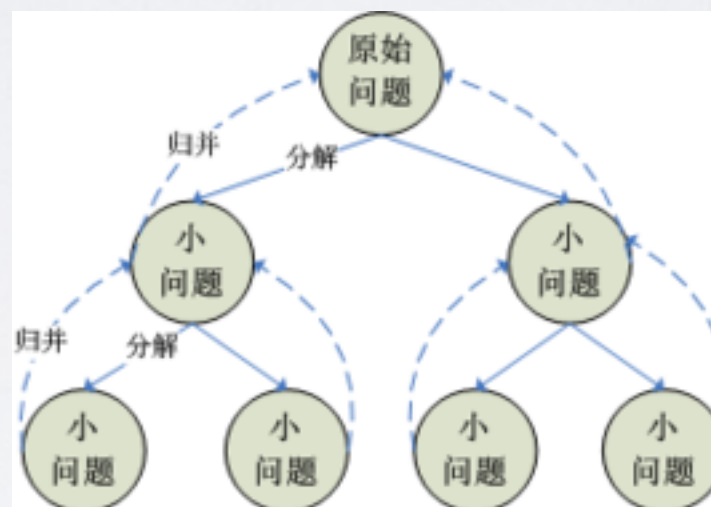
组件之间的关系

- 继承
 - 扩展
 - 嵌套
 - 包含
- 
- 依赖

“Keep Simple. Everything can be a component.”

——React

3. 分治（分而治之）思想



思维上的区别

- 传统框架/类库是DOM优先
- 组件化框架是组件优先

标签化

- 一个标签可以代表一个组件
- 解析模板即可知道页面全貌
- 为可视化前端开发提供了可能

```
<pager current="1" total="8" />
```

技术选型: Vue/React/Angular/Regular + 组件库

规范化

编码规范

- ◎ 命名规范
- ◎ CSS编码规范
- ◎ JavaScript编码规范
- ◎ 组件设计规范

ESLint

- ⦿ 花一天时间先遍历ESLint的437条规则；
- ⦿ 筛选需要的规则，并在组内讨论；
- ⦿ 先将确定的规则全部配error，然后再根据情况降级；
- ⦿ lint存在error时禁止提交代码。

技术选型：ESLint + StyleLint

自动化

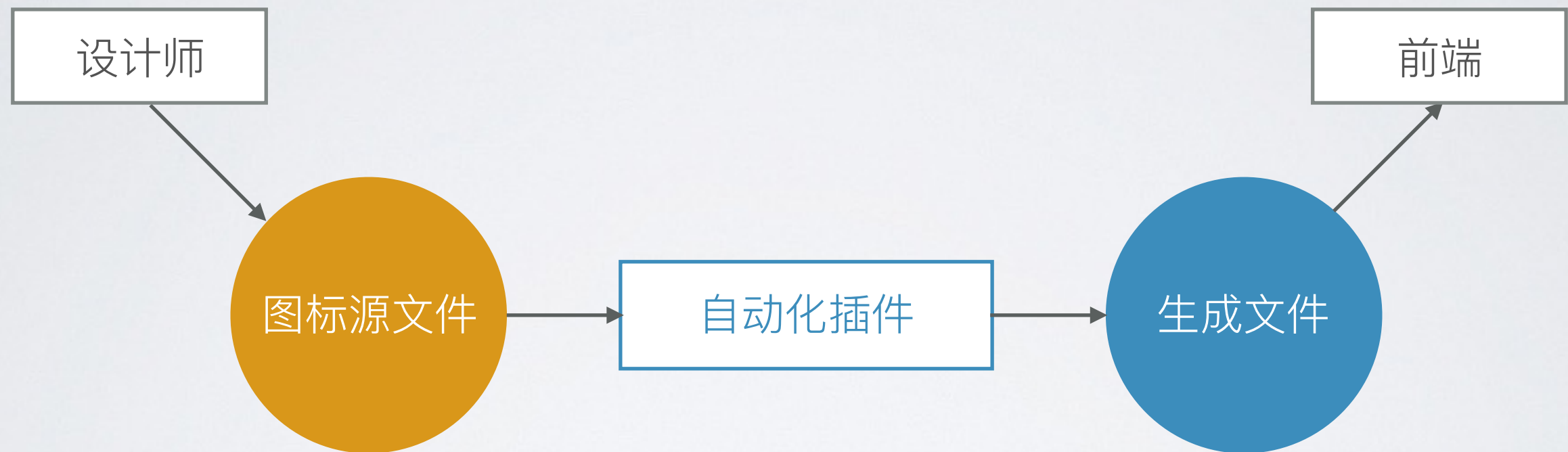
“任何简单机械的重复劳动都应该让机器去完成。”

——Yusen Zhao

手工合并图标

- 雪碧图用PS手动拼接
- 字体图标用Icomoon管理

自动化合并图标



技术选型: SpriteSmith + FontCustom

可视化组件文档

- PostMark
- JSDoc

技术选型：PostMark + JSDoc

前端自动化测试

- 单元测试
- UI测试

技术选型: Karma + Mocha + Chai

构建工具

技术选型：Gulp

QUESTIONS



THANKS