前端工程化

Xaber

分享卖点

- 前端工程化包含哪些内容

- 前端工程化汲取的思考借鉴

前端工程化是什么

如果做一个全新的项目，
一开始会想到什么？

选型

- jQuery
- YUI
- Zepto / jquery mobile / kissy
- lodash / underscore
- Bootstrap / ant design
- React / Vue
- koa / express
- php / node / go / ruby / python
- ……

选型举例

- PC端jQuery都不用，手写代码是否合适?
- 小程序的框架是否需要?
- 结算页这种复杂交互的功能，用jQuery / zepto处理怎么样？—— 2000行代码
- 是否需要增加lodash，代码量会增大

选型总结

- 利益平衡的过程，加代码要增加代码量，多30kb有没有影响，用的多不多。大家会不会，学习成本如何，是否真实提升效率

- 根据具体业务场景与情况，选择合适的框架 / 库 / UI 库 / 组件

撸起袖子就是干

总觉得少了点什么

没有打包压缩，代码都写在一个文件里面

简单构建优化

简单构建优化

- ES6 + babel
- grunt / gulp
- webpack

简单构建举例

- grunt —— 3年前的PC站
- gulp 简单实用 —— PC、小程序
- webpack 简单应用 —— 微信编辑平台、beibei_h5

简单构建插件举例

- gulp
  - through2
  - gulp-rename
  - source-map / gulp-source-map
  - gulp-concat
  - gulp-uglify
  - gulp-concat-css
  - gulp-plumber
  - ……

- webpack
  - webpack-merge
  - TODO

简单构建优化总结

- 学会1-2个构建工具的使用和整合，很重要

那么问题来了，至少也得写一些公用代码，封装点模块吧?
难道只会复制粘贴?

好像也可以

模块化开发

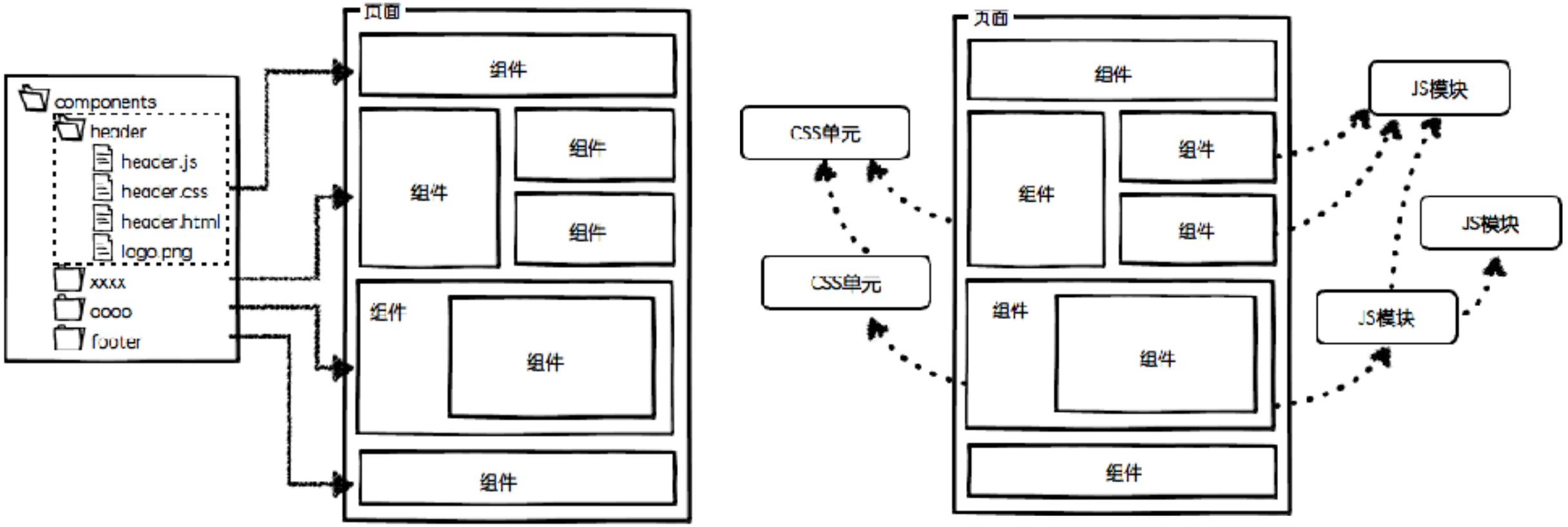
模块化开发

- JS： AMD/CommonJS/UMD/ES6 Module
- CSS：less、sass、stylus等预处理器
- html：ejs等模版引擎、PHP的include

模块化开发总结

- css、html的模块化，会比较弱的，还好有了组件化
- 模块化和组件化还是不一样的，需要分清概念
- 模块化的要素是封装，所以需要平时注重封装、抽离公共函数。可能今天我们当前页面代码里抽离了一个函数，往后，我们会抽象出一个公共模块（例如登录模块、懒加载模块、SKU模块等等）

组件化简单重温


components
header
heacer.js
heacer.css
heacer.html
logo.png
xxxx
oooo
footer
页面
组件
组件
组件
组件
组件
组件
组件
CSS单元
CSS单元
页面
组件
组件
组件
组件
组件
组件
JS模块
JS模块
JS模块

炸🐟啊 当然没完

水了20页PPT，下面开始干货

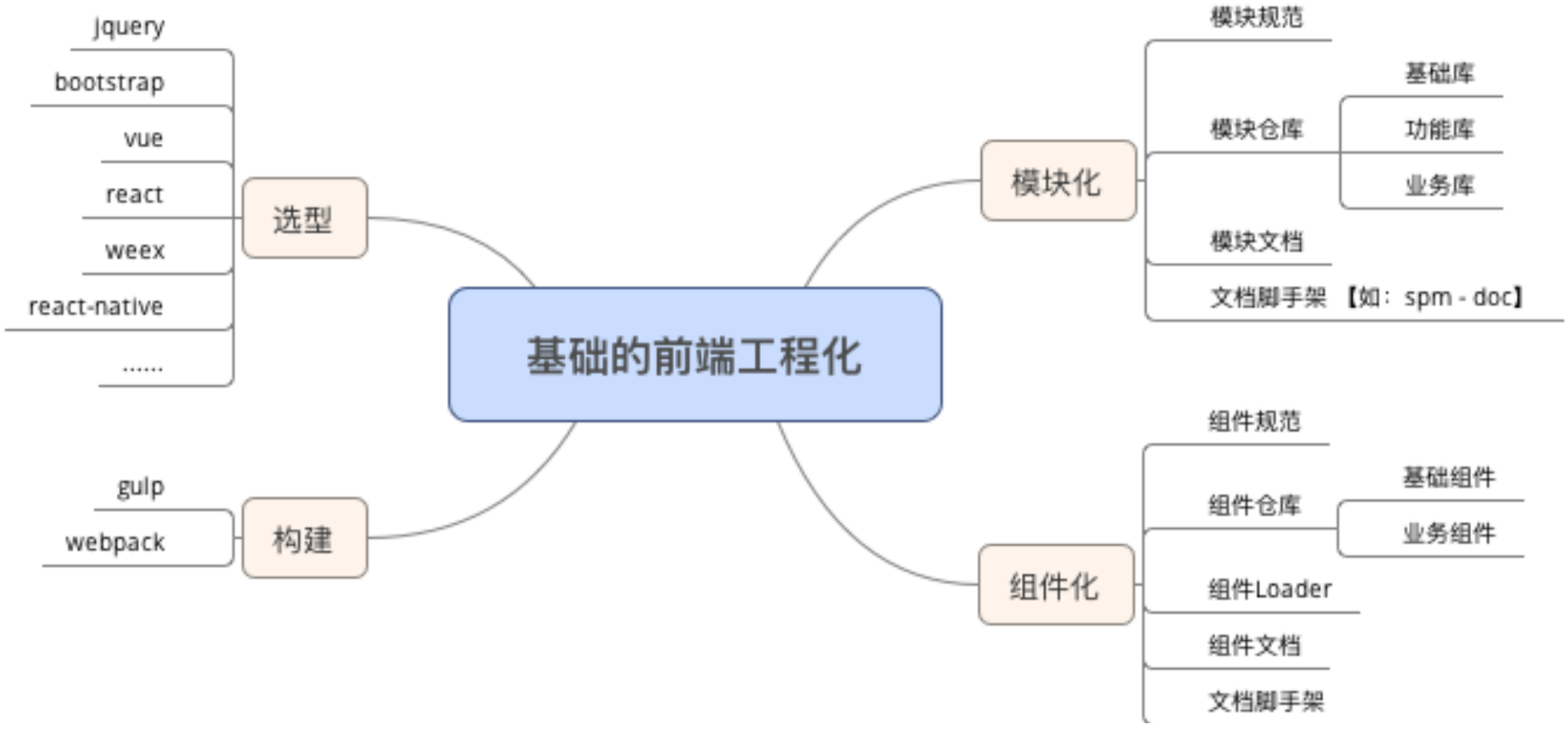
PS：干货会比较干

先看问题

- 大体量：多功能、多页面、多状态、多系统
- 大规模：多人甚至多团队合作开发
- 其他问题（开始开发到开发上线到上线反馈的系列流程中夹杂的各种问题）：
    - UI还原度
    - 本地调试
    - 联调
    - 发布
    - 质量
    - 性能
    - 安全
    - 劫持
    - 业务数据

基础的前端工程化

尝试解决前面两个问题



第三个更多问题

太庞大

进阶的前端工程化

- 发布
  - Jenkins
  - 持续集成
  - 集群部署
- 性能
  - 打点库
  - 性能监控
  - 缓存复用
    - CDN　　增量发布
    - 客户端本地缓存　　离线缓存功能
    - 服务器缓存控制
      - Expires
      - Cache-Control
    - 浏览器缓存
      - localStorage 【微信】
      - Expires 设置永久，from cache 缓存提取
      - 增量发布
  - 其他策略
    - webpack commonjs buddle
    - base64 图片转化
    - 雪碧图
    - 预加载　　例如小程序
  - 按需加载
    - 动态请求文件（seajs、raquire.js）
    - 服务器 combine 【淘宝】
  - 请求合并　　移动端首屏CSS内嵌
  - 服务端　　HTTP 2.0服务端资源推送
  - 其他　　BigRender、BigPipe、Quickling、PageCache
- 安全
  - 设备指纹
  - CSP 集成
- 劫持
  - https
  - 劫持拦截率
  - CSP防劫持
- 数据
  - airpst
  - 链路打点
  - 配置式打点
  - 数据图表与监控
- 质量
  - eslint
  - typescript、flow　　工具
  - 函数复杂度检查
  - precommit
  - 框架
  - 测试库例　　测试
  - 持续集成
  - sentry 质量监控
- 联调
  - mock 服务
- hybrid
  - 标准化
  - API 集成测试
- 本地调试
  - 统一的环境
  - Charles & Tribler
- UI / 动画还原度
  - 切图工具
  - sketch　　标注工具
  - 动画工具

个人点：前端是一种远程部署，运行时增量下载的GUI软件
个人理解：前端代码是访问的时候才获取的，而客户端、服务端代码，是一开始就部署在服务器上的

前端工程化汲取的思考

借鉴

前端工程化汲取的思考借鉴

- 最小模块化细分，根据具体需求，封装、拆分
- 思考问题进行上扩延伸发散，有一张大图
- 可能这不是个业务问题而是个工程问题；可能这不是个代码问题而是个产品UI问题；可能这不是个个性问题而是个共性问题

谢谢

扫左边好友
扫右边赞助

“https://github.com/fouber/blog”

—— 张云龙