第一节 前端工程化概述

1.定义

全服武装: 通过工程化提升战斗力
遵循一定的标准和规范,通过工具提升开发效率,降低成本的一种手段

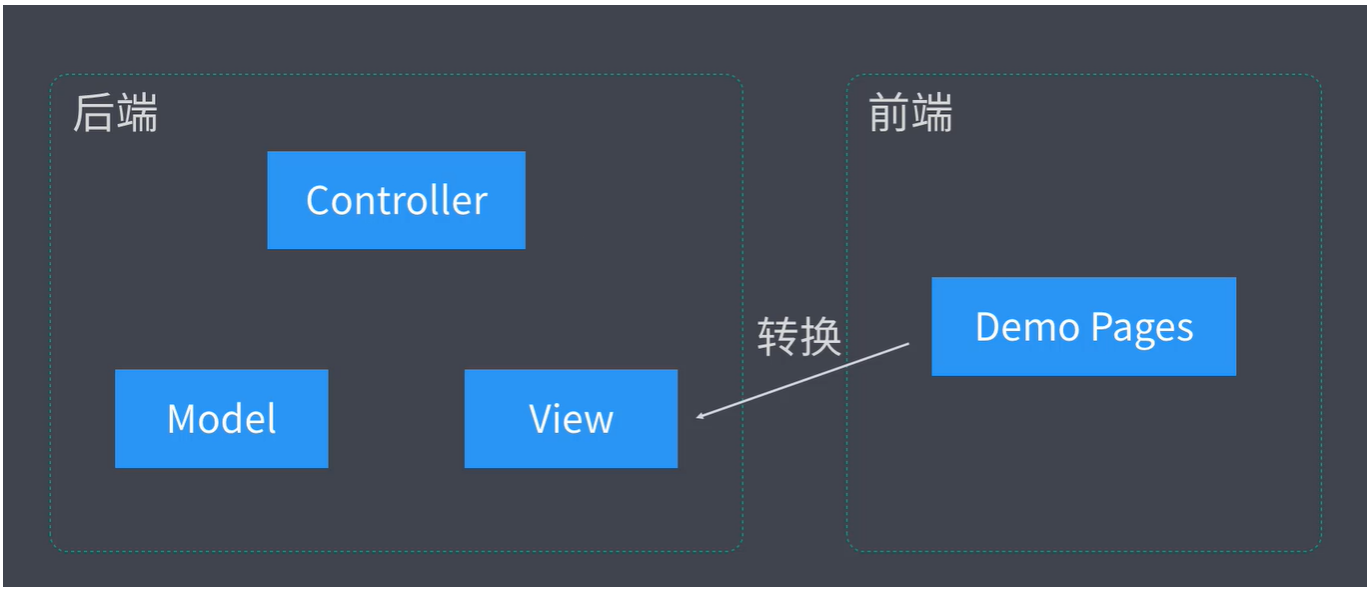
2.为什么需要前端工程化

- 背景:

  前端应用功能要求不断提高,业务逻辑日益复杂
  从传统的网站,到h5,小程序,桌面应用,跨多端应用,后台开发,前端应用覆盖广泛
  行业对前端开发者的要求发生了天翻地覆的变化

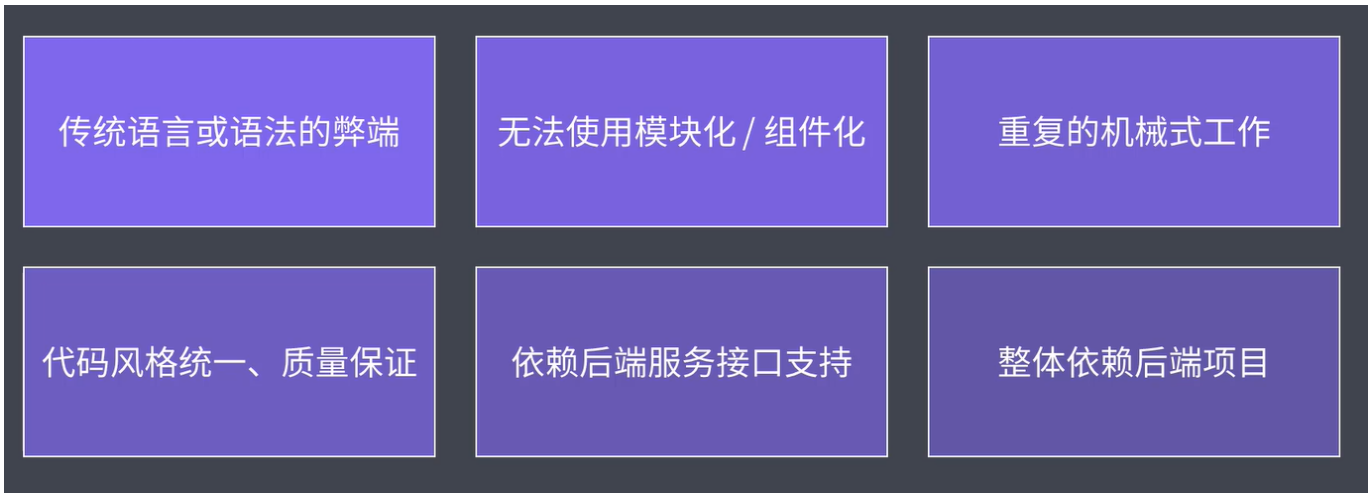
- 历史:

前端只关注Demo页面,套模板的时代



面临着日益复杂的前端需求,前端工程化孕育诞生了

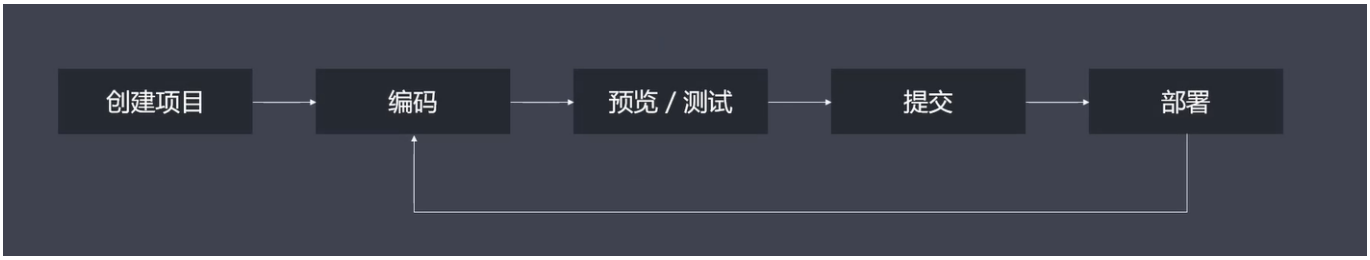
- 为什么需要前端工程化
  - 想要使用ES6+的新特性,但是浏览不支持
  - 想使用Less/Sass/PostCSS增强CSS的编程型,但是运行环境不支持
  - 想要使用模块化的方式提高项目的可维护性
  - 部署上线前需要手动压缩代码及资源文件
  - 部署过程需要手动上传代码到服务器
  - 多人协作开发,无法硬性统一代付风格
  - 部分功能开发需要等待后端服务接口提前完成
- 可以解决如下问题



3.工程化的表现

一切以提高效率,降低成本,质量保证为目的的手段手属于"工程化"

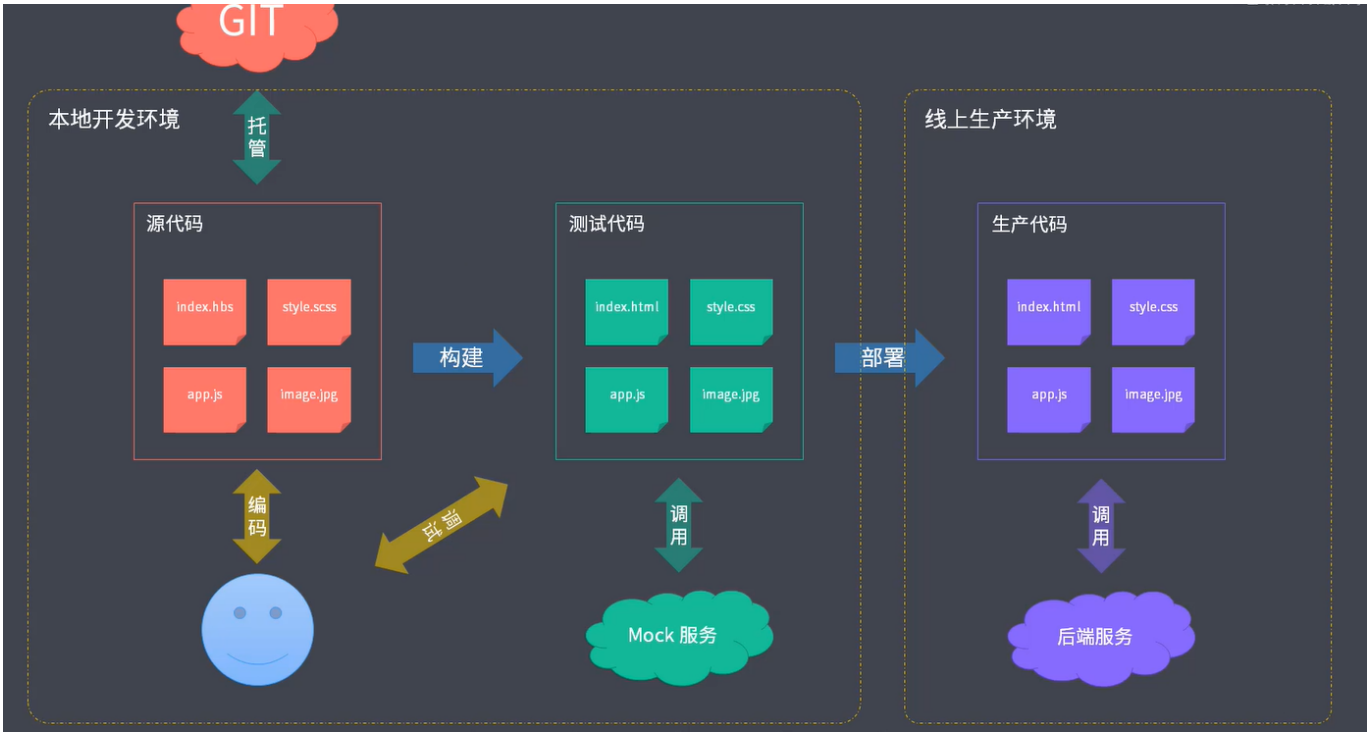
如下图所示是一个项目的开发流程,每一个环节都可以通过工程化方式,大大提高我们的效率



- 创建项目
  - 可以通过脚手架自动创建项目结构及各种文件
- 编码
  - 代码格式化
  - 代码风格校验
  - 使用编译工具例如babel,帮我们编译ES6+的语法,构建成es3-5的代码
- 预览测试
  - 传统的项目使用apache或者nginx来搭建web服务器进行测试预览,没有热更新的体验
  - 可以借助如下工具进行web服务器热更新,测试数据mock,通过Source Map来定位源代码那个地方出现问题
  - WebServer / Mock
  - live Reloading / HMR
  - Source Map
- 提交
  - git hooks进行代码检查(质量和风格检查)
  - lint-staged
  - 持续集成
- 部署
  - CI/CD
  - 自动发布 代码提交之后,通过持续集成,自动部署到机器,自动化的发布代码

4.工程化不等于某个工具

工程化是对项目的整体规划和架构,而工具只是帮我们落地实现这种规划和架构的一种手段



规划一个工作流的整体架构

- 文件的组织结构
- 源代码的开发范式(需要用什么让的语法,风格)
- 需要用什么方式进行前后端分离(基于ajax或者中间层)
- 持续集成
- 持续发布

以下一些特定项目成熟的工程化方案

5.工程化与node.js

Powered by Node.js

前端工厂化依赖于node.js,让前端工程化发生了天翻地覆的变化

6.脚手架工具作用

- 创建项目基础结构,提供项目规范和约定
  - 相同的组织结构
  - 相同的开发范式
  - 相同的模块依赖
  - 相同的工具配置
  - 相同的基础代码
- 根据脚手架创建的项目骨架进行后续的开发

7.常用的脚手架工具

- 特定的脚手架工具

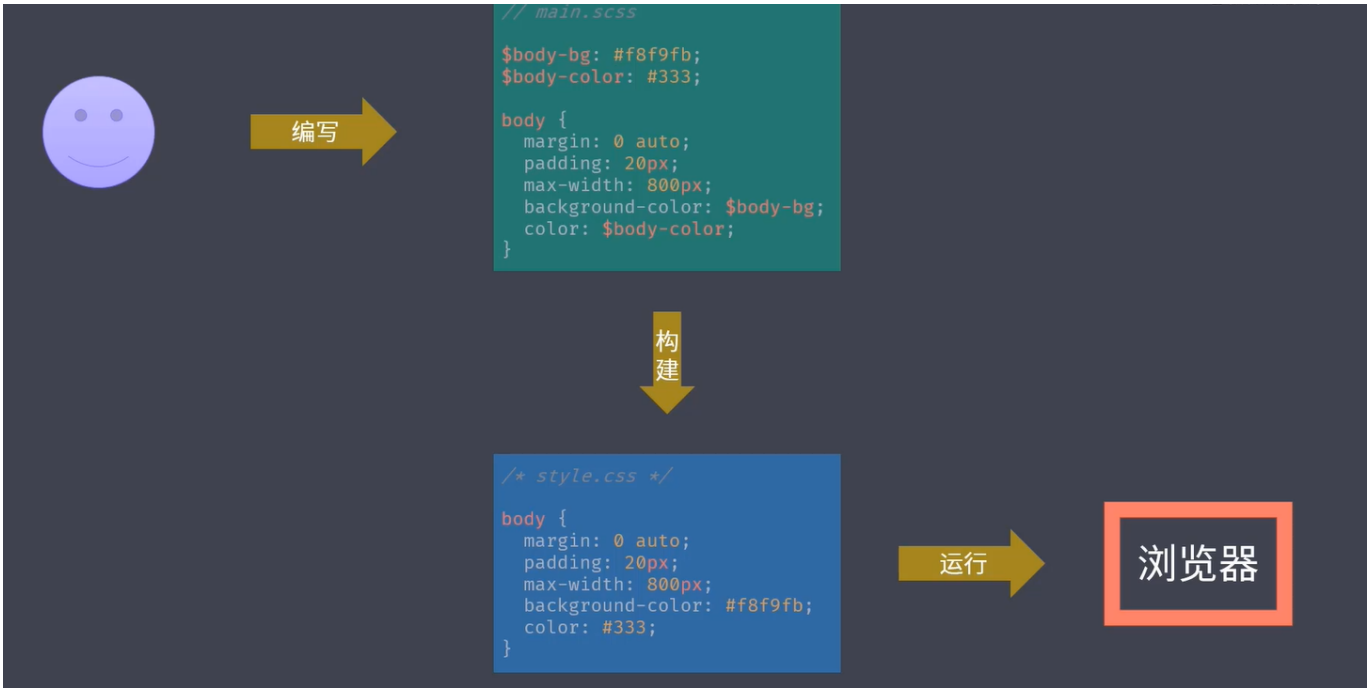
比较成熟的脚手架工具(根据信息创建对应的项目基础结构)

- 通用型项目脚手架工具
  - Yeoman
  - Plop(创建一个组件/模块所需要的的文件)

8.自动化构建

- 自动化
  - 使用机器代替手工,完成一些工作
- 构建
  - 可以理解称为转换(一个东西转换成另外一个东西)
- 开发阶段的构建如下图(自动化构建工作流)
  - 脱离运行环境兼容带来的问题
  - 使用提高效率的语法,规范,和标准(这些浏览器是不支持的)
    - ES6+
    - Sass
    - 模板引擎
    - eslint
  - 构建转换那些不被支持的"特性"

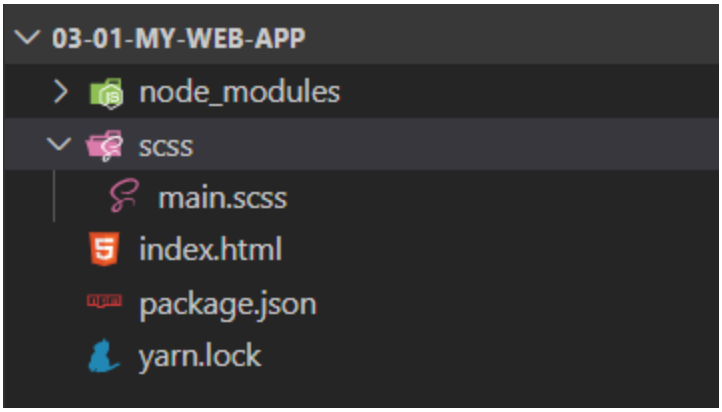
例如项目中使用scss编写样式,但是浏览器不支持scss的语法
我们需要把scss编译成css,然后才能在浏览器中使用



添加sass npm模块

```
1 yarn add sass -D
```

基础项目如下



可以使用如下命令转换scss

```
1 npx sass scss/main.scss css/style.css
```

问题: 重复执行这些很长的命令,很不友好(可以使用npm scripts)

--watch可以检测文件改变自动编译

问题: 需要在服务器中运行,可以使用browser-sync npm模块,打开一个测试服务器

```
1 yarn add browser-sync -D
```

问题: 需要在运行服务器的时候实时监控文件,然后进行热刷新

- --files \"css/*.css\"" 指定检测的文件
- 安装npm-run-all模块,运行多个命令

```
1 yarn add npm-run-all -D
```

修改package.json

```
1 {
2   "name": "my-web-app",
3   "version": "0.1.0",
4   "main": "index.js",
5   "author": "zce <w@zce.me> (https://zce.me)",
6   "license": "MIT",
7   "scripts": {
8     "build": "sass scss/main.scss css/style.css --watch",
9     "serve": "browser-sync . --files \"css/*.css\"",
```

```
    "start": "run-p build serve"
  },
  "devDependencies": {
    "browser-sync": "^2.26.7",
    "npm-run-all": "^4.1.5",
    "sass": "^1.26.8"
  }
}
```

9.常用的自动化构建工具

npm script只能完成一些简单的自动化,,所以需要更专业的工具

(webpack是模块打包工具)

- Grunt是基于临时文件进行构建的,效率相对来说比较慢,需要很多次的io操作
- Gulp是基于内存实现的,很多操作都是基于内存的数据流进行的,同时执行多个任务
- fis3是百度开源的一个工具,FIS3采取了类似CSS语法一样的配置风格,集成了很多前端常用的自动化功能,但是已经不维护了
- 推荐使用glup