用友
yonyou

用友云平台
yonyou cloud platform

互联网开发培训教程
iuap 前端技术框架

2019 年

版权

目录

1 课堂目标

本节课讲结合案例学习 iuap 开发框架，具体目标：

1、 结合了解业界前端技术趋势；

2、 掌握 iuap 前端开发框架

3、 掌握企业级应用前端开发技能；

2 前端技术发展趋势

2.1 后端模板为主的早期时代

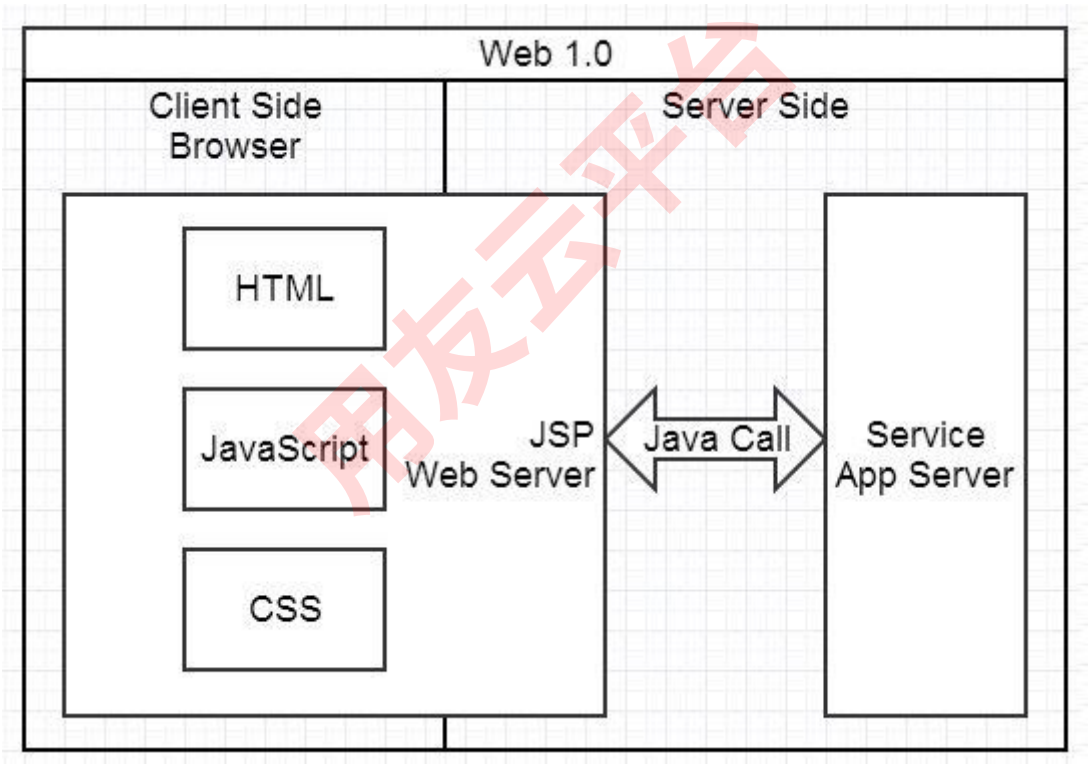


图 1

2.2 后端为主的 MVC 时代

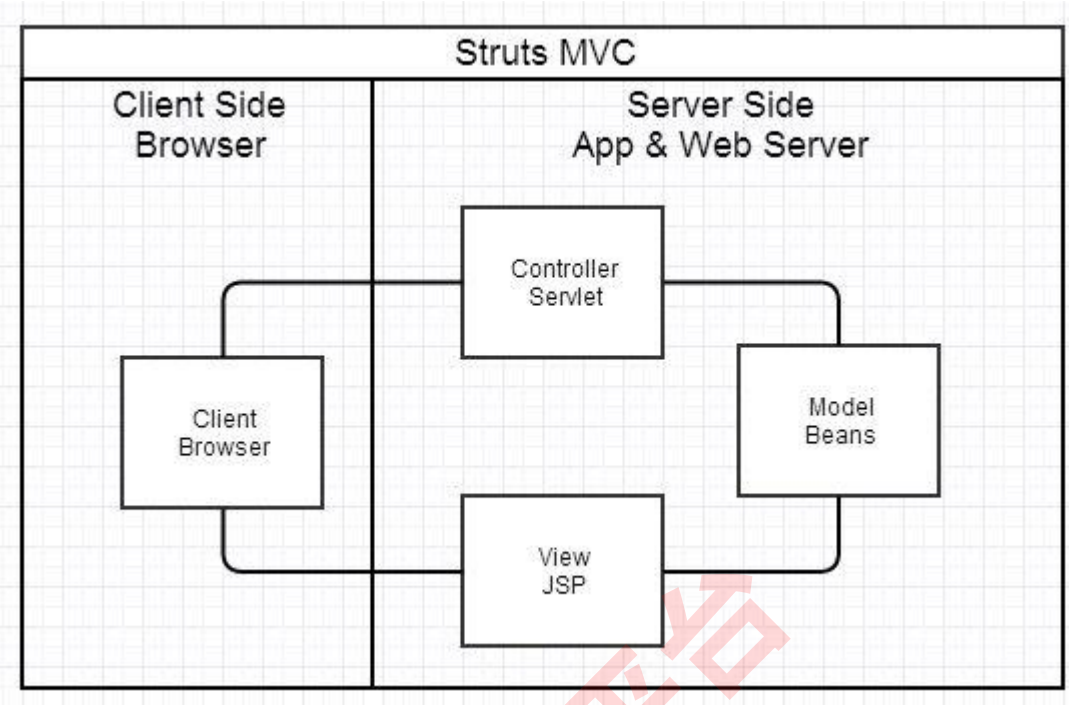


图 2

2.3 AJAX 技术带来的 SPA 时代

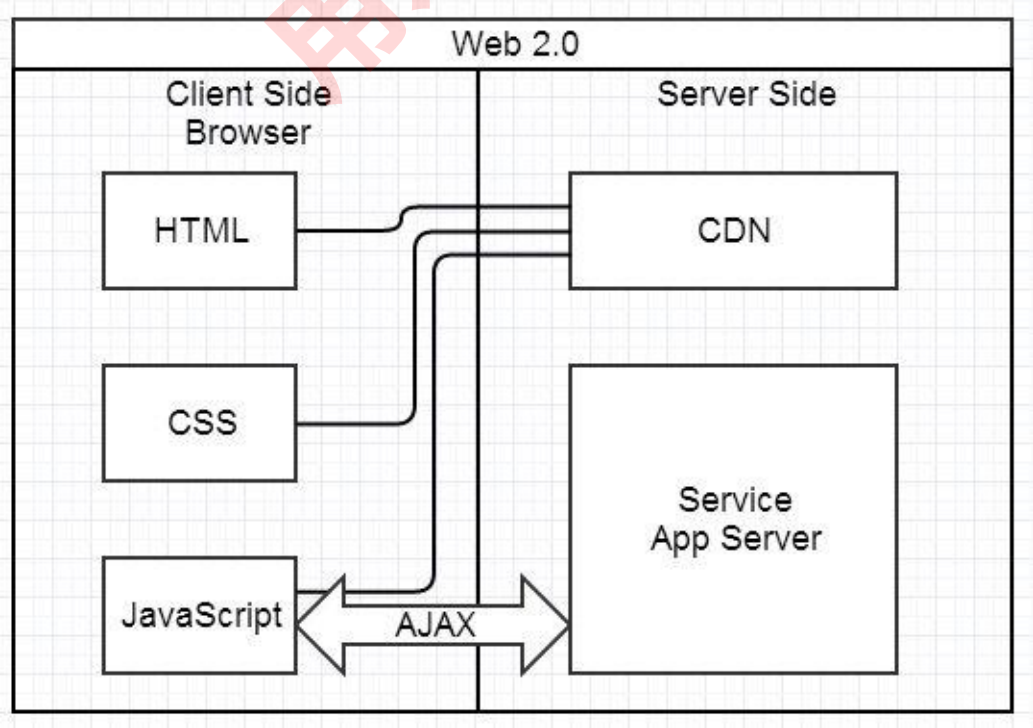


图 3

2.4 前端为主的 MV* 时代

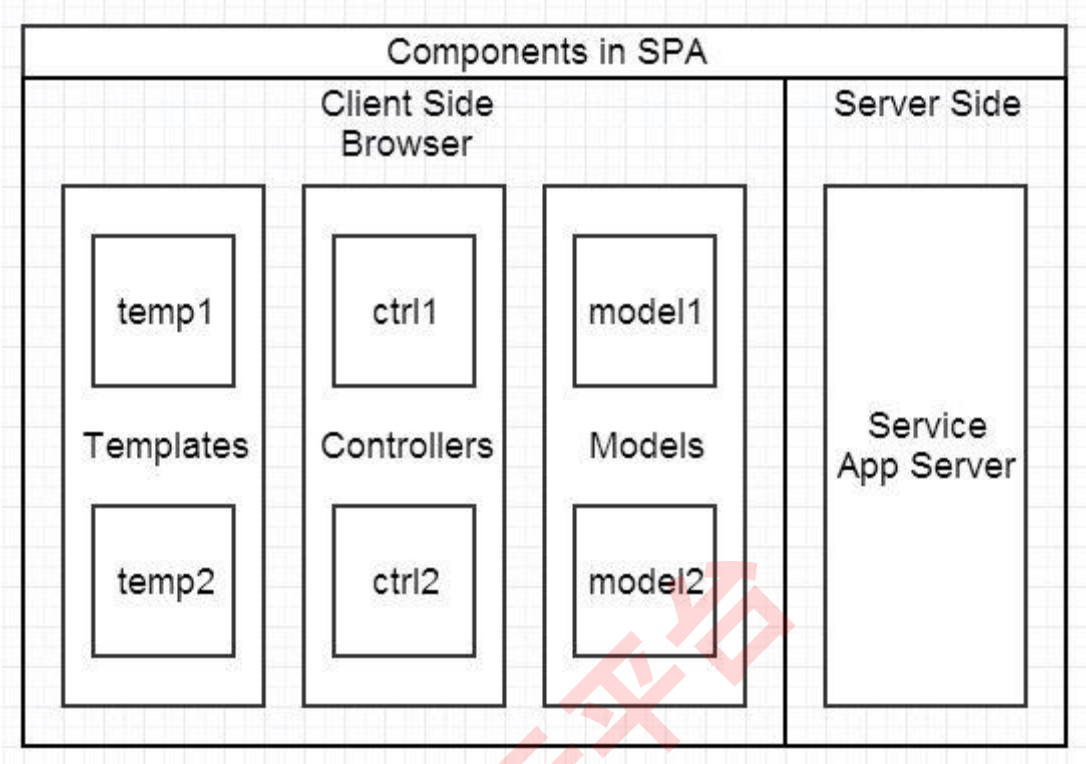


图 4

2.5 Node.js 兴起的新 Web 全栈时代

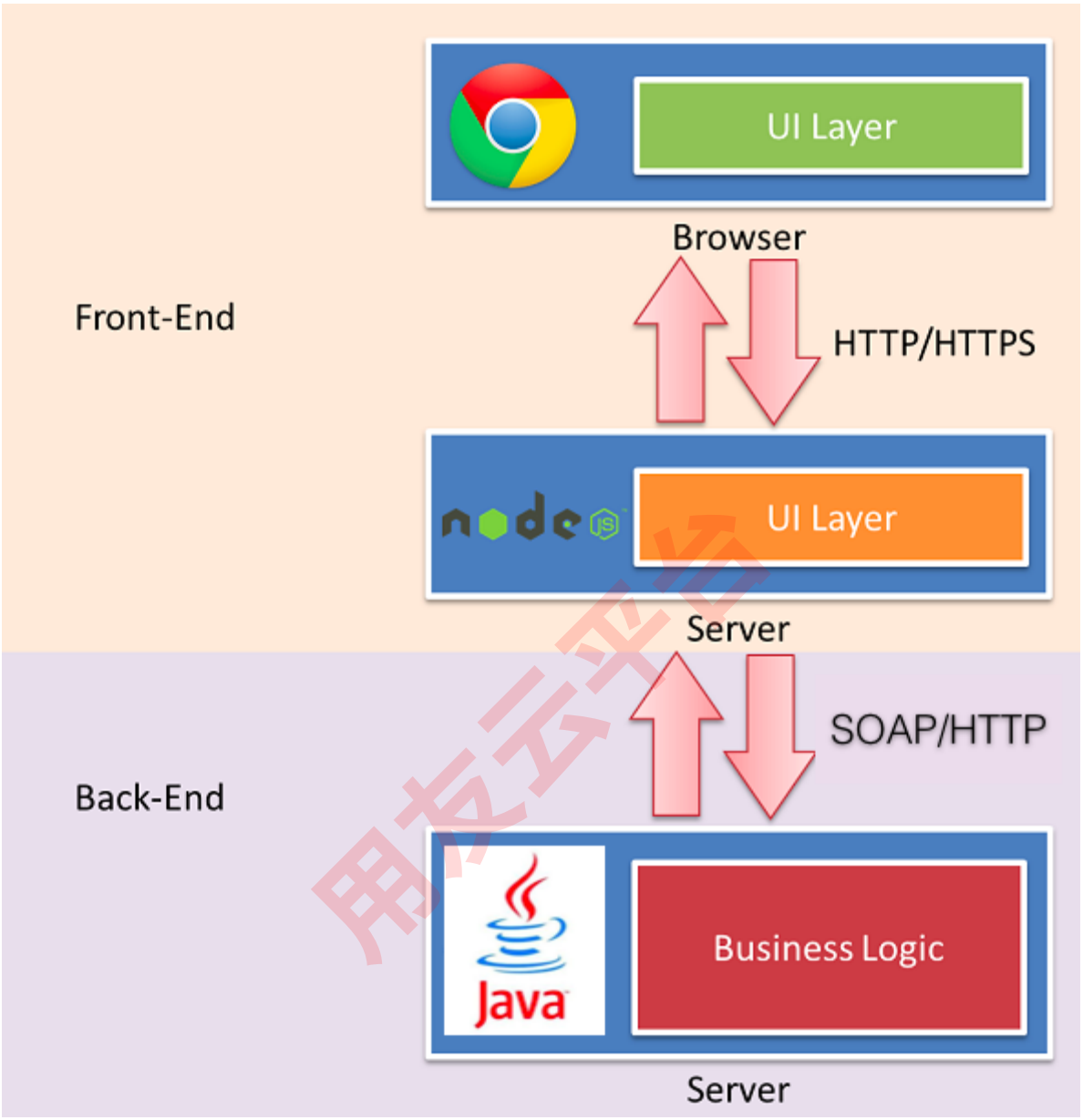


图 5

2.6 以三大框架为主的新成熟阶段

	Angular	React	Vue
类型	一个框架	用于构建UI的库	图书馆
为什么选择	如果要使用TypeScript	如果你想要“一切都是JavaScript”的方法	简单的JavaScript和HTML
创始人	由Google提供支持	由Facebook维护	由前Google员工创建
初始发行	2016年9月	2013年3月	2014年2月
应用类型	如果您想开发Native应用程序，混合应用程序和Web应用程序	如果您想开发SPA和移动应用程序	高级SPA并开始支持Native应用程序
非常适合	如果您想专注于大规模，功能丰富的应用程序	适用于iOS和Android的现代Web开发和原生渲染应用程序	适用于Web开发和单页面应用程序
学习曲线	陡峭的学习曲线	比Angular容易一点	一个小的学习曲线
开发者友好	如果要使用基于结构的框架	如果您希望在开发环境中具有灵活性	如果你想分离关注点
模型	基于MVC（模型 - 视图 - 控制器）架构	基于Virtual DOM（文档对象模型）	基于Virtual DOM（文档对象模型）
写在	打字稿	JavaScript的	JavaScript的
社区支持	庞大的开发者和支持者社区	Facebook开发者社区	开源项目通过众包赞助
语言首选项	建议使用TypeScript	建议使用JSX - JavaScript XML	HTML模板和JavaScript
声望	在开发人员中广受欢迎	全年增加了27,000多颗星	在这一年里，GitHub上增加了超过40,000颗星
公司使用	由Google，Forbes，Wix和weather.com使用	用于Facebook，Uber，Netflix，Twitter，Reddit，Paypal，Walmart等	由阿里巴巴，百度，GitLab等人使用

图 6

自 2016 年左右，纯 web 前端技术的演进和发展不再剧烈，经过无数开发人员的实践和沉淀，React、Vue、Angular 三大框架成为了成熟阶段的 UI Libarary 和 Framework。其中，React 在全球使用者中占比最高；Vue 也以其文档和上手快的特点快速上升；Angular 也快速的发布了 7.X 版本。

Tinper 前端技术平台选择基于 React.js 作为 Basic UI Library，其 Web 组件化能力、基于 Fiber 算法的渲染引擎、Vitual DOM 机制、组件生命周期管理等特性，能够让我们专注于应用层的开发。并且，基于 React 的社区生态，能够拥抱更多技术红利。

3 tinper-react 前端技术框架

3.1 整体架构介绍

图 7

如上图所示，tinper-react 是一个基于 React 的现代化前端框架，包含前端工程化、开发语言及规范、一致性用户体验及前端组件库等能力，集成了全局应用状态管理方案和路由方案，并提供了国际化、全键盘支持、响应式适配等能力，具备完整的前端架构所需的各项能力，适用于复杂中后台业务系统的大规模快速前端开发。

3.2 前置知识说明

想要快速的掌握 tinper-react 开发框架，对于零基础人员是不太合适的，所以我们约定，需要掌握或了解以下能力：

1、 掌握基本的 Javascript 语言能力，能够使用 HTML/CSS 开发静态页面，具备初级前端工程师的水平；

2、 掌握 React.js 基础，熟悉组件定义、生命周期、事件等能力，并且从以操作 DOM 为核心的思维中（以 Jquery 使用者为代表）向组件化思维转变。

3、 掌握基本且常用的 ES6 新语法特性，如箭头函数、ESM 模块化规范、对象扩展、解构赋值等。

4、 了解 Redux 应用状态管理方案和 React Router 路由管理方案，尽管我们基于二者做了简化开发的封装，但了解他们的设计思想和基本用法会很有帮助。

5、 了解基本工具，如 Webpack 构建工具、Babel 解析器、PostCSS 预处理器等，他们已经经过封装和集成在 uba 中使用，如果有时间，可适当了解。

3.3 框架目录结构

3.3.1 整体目录介绍

```
├── LICENSE
├── README.md
├── doc
├── gulpfile.js
├── mock
├── node_modules
├── package-lock.json     // npm 锁包文件，自动生成
├── package.json          // npm 包管理文件
├── postcss.config.js     // PostCSS 插件配置
├── src                   // 开发态源代码
├── uba.config.js         // uba 配置文件
└── uba.mock.js
```

3.3.2 源码目录结构

```
├── app.less          // 项目级公共 Styles 样式
├── components        // 项目级公共 UI 组件
├── layout            // 布局类组件，如 Header，Sidebar 等
├── pages             // 各业务模块，一个目录一个业务；如果分领域，则 pages 变更为 modules，modules 下的目录为业务领域目录
|   ├── allowances
|   ├── masterdetail-many
|   ├── masterdetail-one
```

```
│   ├── singletable-grouping
│   ├── singletable-inline-edit
│   ├── singletable-popupedit
│   ├── singletable-query
│   ├── singletable-viewtemplate
│   └── tree
├── static          // 公用静态资源，统一维护
│   ├── font
│   ├── images
│   └── trd
└── utils           // 抽离封装的公共工具类函数
```

3.3.3 前端节点代码结构

节点主要按照视图 UI 层、数据模型层、容器层、服务请求层、通用工具层、路由配置层，其中

① **视图 UI 层** 用于设计视图中的 UI 组件，视图 UI 层放置于节点目录 components 文件夹下

② **路由配置层** 用于配置页面访问路径（通过输入路由地址，展示相应的组件）、路由配置层位于节点 routes 文件夹

③ **数据模型层** 用于配置数据模型，供 UI 层使用。位于 model.js 中

④ **容器层** 容器层用于连接数据模型层与视图 UI 层 位于 container.js 中

⑤ **服务请求层** 用于定义具体的请求结构及完成最终请求 位于 service.js 中

⑥ **通用工具类** 用于定义通用的处理方法 位于 src/utils/ 文件夹中

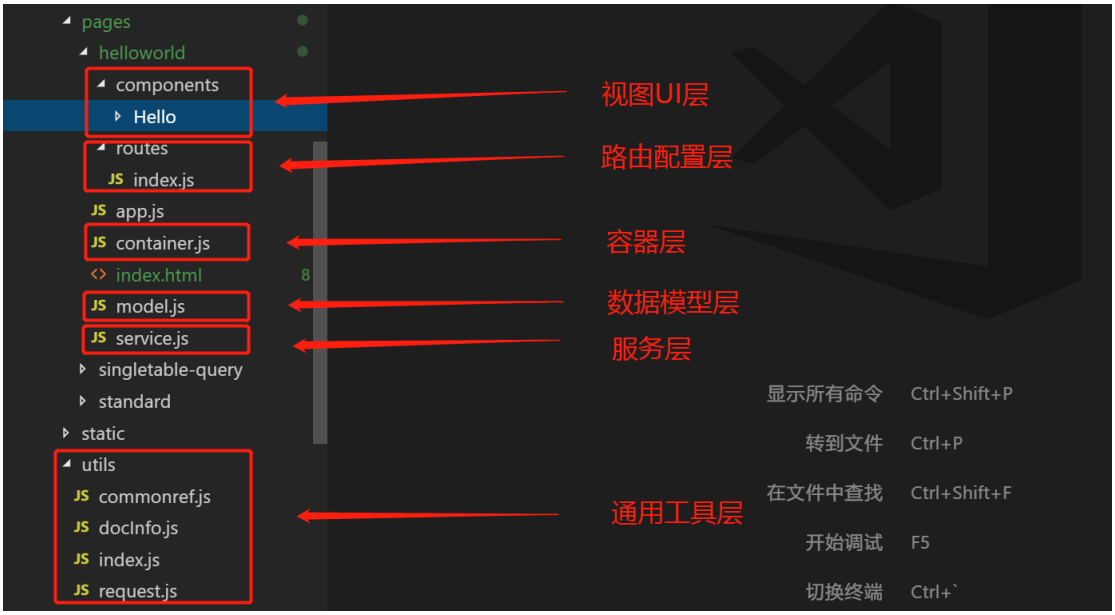


图 8

3.4 开发框架运行原理说明

3.4.1 开发阶段的静态资源构建

tinper-react 开发框架中使用 uba 前端构建工具进行开发态资源的动态的编译，其中 uba 内置了本地的 Node 服务，并集成了 Webpack，所以本地直接会起一个 server 进行开发态资源的访问和调试。

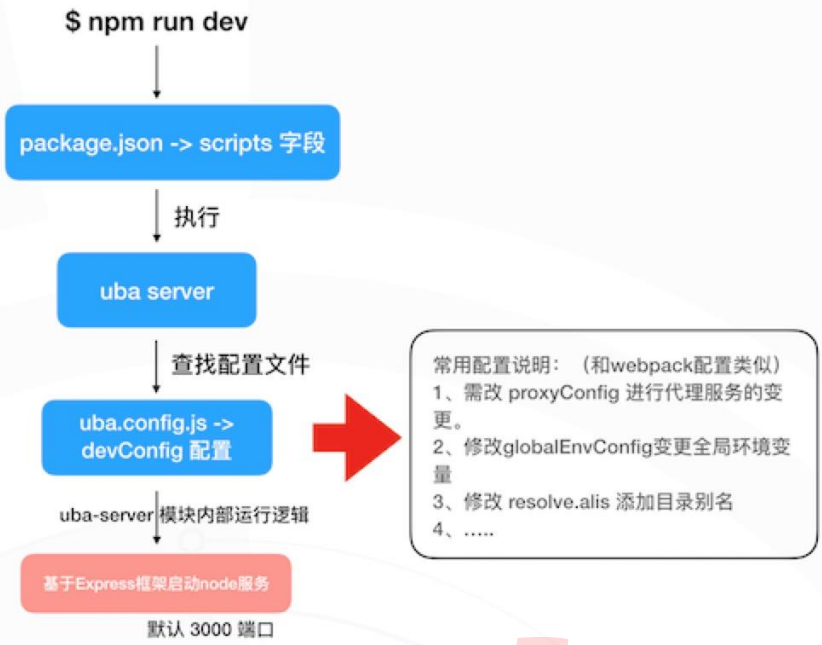


图 9

3.4.2 生产环境的静态资源构建

和开发态不一样，构建出适用于生产环境的静态资源将去除掉很多开发态的辅助工具，并将资源进行编译输出和资源压缩等处理。

图 10

3.4.3 静态资源运行时解析渲染

在开发阶段使用 ES6/7/8 等新语法特性相关的代码，会被 Babel 解析成 ES5 规范的代码，可以运行在不同厂商的浏览器；使用 React 的 JSX 语法写的代码，也会通过 babel-preset-react 这个工具解析成 React API 的形式，通过全局加载 react.js 和 react-dom.js 两个文件进行解析渲染。

3.5 基于 uba 的前端工程化

图 11

3.6 一致性 UI 组件能力

3.6.1 基础组件库：tinper-bee

`tinper-bee`（bee.tinper.org） 是一套基于 `React.js` 的开源组件库，它从丰富的企业级中后台应用场景中实战沉淀而来，为复杂应用的快速开发提供一致性 `UI` 解决方案。基本以下基本特性：

- 功能丰富、企业应用场景、高质量
- 友好易用的 API ，详细的文档
- 一致性 UI、遵循设计语言

- 满足各种复杂场景的 Grid 组件能力
- 支持兼容性
- 支持全键盘能力
- 支持国际化多语言
- 支持多端适配

3.6.2 业务组件库：tinper-acs

1、 参照组件（树形、表型、树表型、树表穿梭型、Combobox….）
2、 附件管理组件
3、 权限组件
4、 图表类组件
5、 富文本编辑组件
6、 复杂 Grid 组件
7、 多语录入组件
8、 BPM 流程组件
9、 …..

4 企业级前端开发实战

4.1 前端开发规范和代码质量

4.1.1 HTML/CSS 规范

1、https://github.com/iuap-design/YY-Code-Guide/blob/master/HTML.md
2、https://github.com/iuap-design/YY-Code-Guide/blob/master/CSS.md

4.1.2 JavaScript 规范

https://github.com/iuap-design/YY-Code-Guide/blob/master/JavaScript.md

4.1.3 React 组件规范

https://github.com/iuap-design/YY-Code-Guide/blob/master/React%E9%A1%B9%E7%9B%AE%E5%BC%80%E5%8F%91%E8%A7%84%E8%8C%83.md

4.1.4 项目开发规范

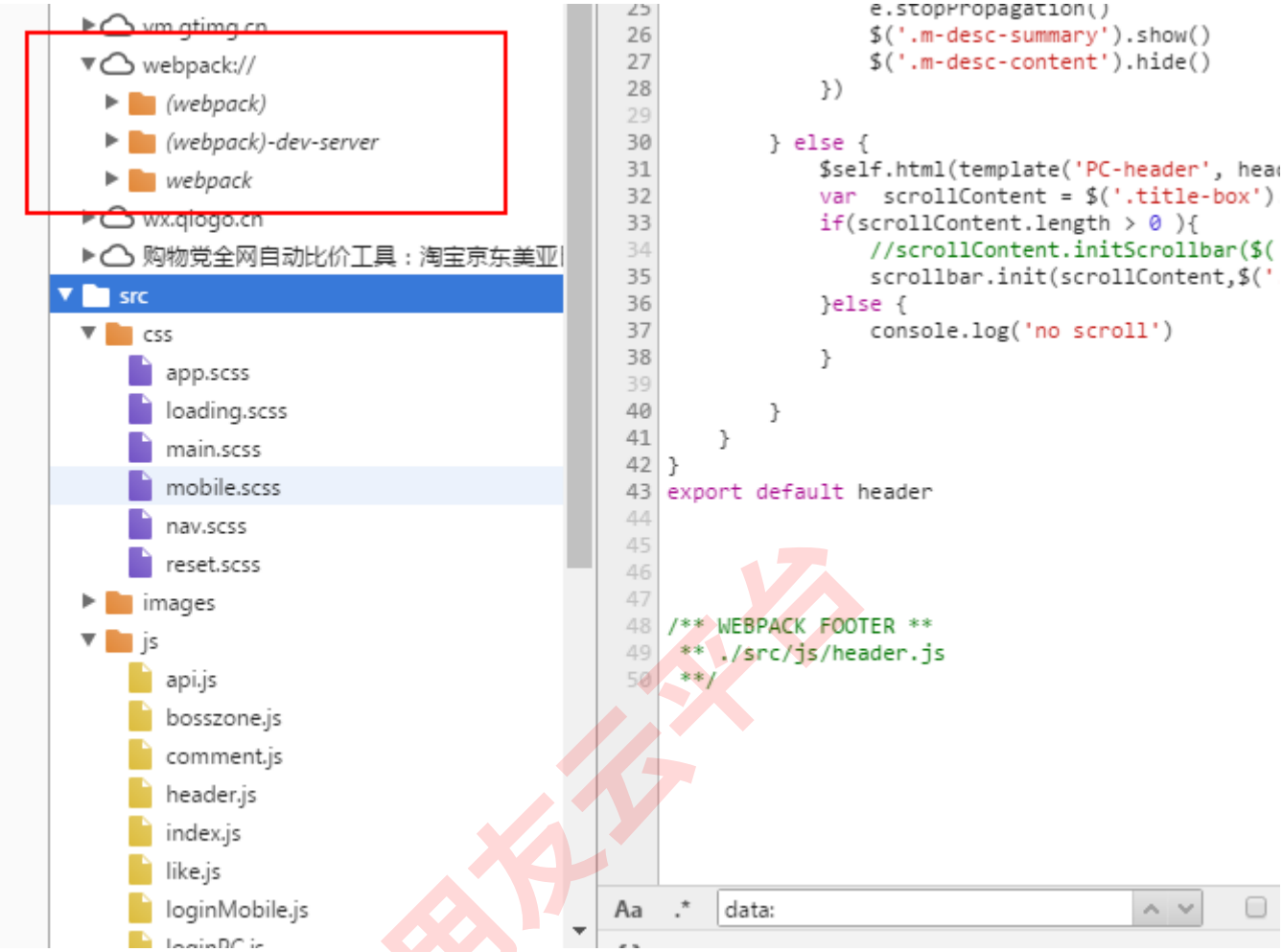
https://github.com/iuap-design/YY-Code-Guide/blob/master/walsin-code-guide.md

4.2 开发调试技巧

4.2.1 浏览器插件

React Developer Tools 是一款由 facebook 开发的一款非常好用的 chrome 调试插件。

4.2.2 基于 source map 的浏览器源文件调试



4.2.3 fiddler 抓包调试

Fiddler 是强大的抓包工具，它的原理是以 web 代理服务器的形式进行工作。

4.3 并行开发神器：mock 平台和数据代理

4.3.1 数据代理的配置方式

```
const proxyConfig = [
  // 应用平台
  {
    enable: false,
```

```
        headers: {
            // 这是之前网页的地址，从中可以看到当前请求页面的链接。
            "Referer": "http://159.138.20.189:8080"
        },
        // context，如果不配置，默认就是代理全部。
        router: [
            '/wbalone'
        ],
        url: 'http://159.138.20.189:8080'
    },
    // 后台开发服务
    {
        enable: false,
        headers: {
            // 这是之前网页的地址，从中可以看到当前请求页面的链接。
            "Referer": "http://159.138.20.189:8180"
        },
        // context，如果不配置，默认就是代理全部。
        router: [
            '/iuap_pap_quickstart'
        ],
        url: 'http://159.138.20.189:8180'
    }
];
```

4.3.2 集成 Mock 接口管理平台

https://mock.yonyoucloud.com/

4.4 前后端集成部署

4.4.1 CDN 资源发布

发布到 OSS 服务器，

4.4.2 Nginx 负载均衡服务器

4.4.3 静态资源集成到 Tomcat

4.4.3.1 代码拷入后端后端

基于 Git 仓库，将前端 build 后的静态资源，直接拷贝到 java 工程的 src/main/webapp 目录下。

4.4.3.2 打成 war 发布到 maven

使用 gulp、webpack 将 build 后的静态资源打包，生成 war 包，利用本地配置的 mvn 工具，将 war 包发布到内网的 maven 镜像服务器。

4.5 性能调优和常见误区

4.5.1 组件性能优化

1、 合理定义 state\props
2、 合理使用 setState
3、 合理使用纯函数组件
4、 组件生命周期函数的使用
5、 事件的解绑和 DOM 的销毁
6、 组件的嵌套层级
7、 ….

4.5.2 资源体积打包策略优化

1、 体积过大
2、 分包策略
3、 抽取 chunk
4、 区分开发环境和生产环境的资源
5、 资源压缩
6、 服务器上对资源 Gzip 处理