Loader开发

- Why

  当下前端社区里最为流行的三大框架 React、Angular.js 和 Vue，它们的一些语法，比如 JSX 和 Vue 指令，这些都是浏览器无法直接解析的，也需要经过构建工具进行转换，而 webpack 毫无疑问是前端构建领域里最耀眼的一颗星。无论你前端走那条线，你都要有很强的 webpack 知识。熟悉 webpack 的使用和原理可以让你拓宽前端技术栈，在发现页面打包的速度和资源体积的问题时能够知道如何排查问题和优化手段，同时，熟悉 webpack 的原理将会对其它跨端开发比如小程序、Weex、ReactNative、Electron 等框架的打包快速上手。

- How

  因此本课程设计的时候遵循由浅入深的原则，从最简单的webpack loader和plugin的写法开始，逐步学习编写自定义loader和plugin的方法与技巧，并在阅读现有流行的loader和plugin源码后，编写自己项目中可用的loader和plugin

- Loader概述
    - **概念：**

> loader 用于对模块的源代码进行转换。loader 可以使你在 import 或 "load(加载)" 模块时预处理文件。因此，loader 类似于其他构建工具中“任务(task)”，并提供了处理前端构建步骤的得力方式。loader 可以将文件从不同的语言（如 TypeScript）转换为 JavaScript 或将内联图像转换为 data URL。loader 甚至允许你直接在 JavaScript 模块中 import CSS 文件！

    - **使用方法：**
        - 安装

```
npm install xxxx
```

        - 配置

```
// 只需要一个loader
module.exports = {
    module: {
        rules: [
            { test: /\.ts$/, use: 'ts-loader' }
        ]
    }
};
```

```
// 多个loader配合
module.exports = {
    module: {
        rules: [
            {
                test: /\.css$/,
                use: [
                    { loader: 'style-loader' },
                    { loader: 'css-loader' }
                ]
            }
        ]
    }
};
```

        - 执行时机

webpack在编译（Compile）源代码时，会递归的处理被import/require的模块，使用对应的loader进行代码转换。