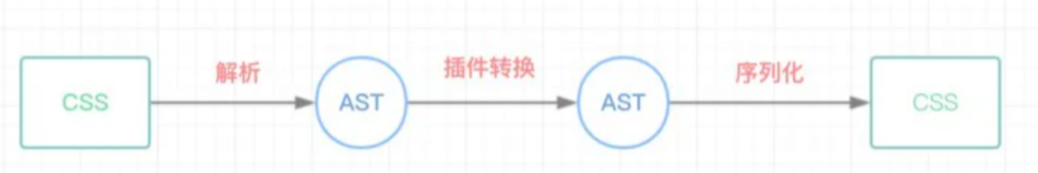
专题十: CSS工程化和框架应用

> 大纲目录:
> - PostCSS和基于PostCSS的CSS工程化体系
> - CSS模块化方案(css modules)
> - Vue框架中的CSS使用
> - React框架中的CSS使用

PostCSS和基于PostCSS的CSS工程化体系

> https://www.postcss.com.cn/
> postCSS简介
> - PostCSS是一个用 JavaScript 工具和插件转换CSS代码的工具。
> - PostCSS 是一个允许使用 JS 插件转换样式的工具。 这些插件可以检查（lint）你的 CSS，支持 CSS Variables 和 Mixins， 编译尚未被浏览器广泛支持的先进的 CSS 语法，内联图片，以及其它很多优秀的功能。
> - PostCSS 在工业界被广泛地应用，其中不乏很多有名的行业领导者，如：维基百科，Twitter，阿里巴巴， JetBrains。PostCSS 的 Autoprefixer 插件是最流行的 CSS 处理工具之一。
> - PostCSS 接收一个 CSS 文件并提供了一个 API 来分析、修改它的规则（通过把 CSS 规则转换成一个抽象语法树的方式）。在这之后，这个 API 便可被许多插件利用来做有用的事情，比如寻错或自动添加 CSS vendor 前缀。
>
> postCSS架构图
> 通过PostCSS将CSS转换成AST(抽象语法树)，对应的是JavaScript对象；然后通过插件遍历AST，进行增加，删除，修改；最后再生成CSS文件，这就是整个流程，跟babel的架构非常相似。



> postCSS作用
> - postCSS本身只有解析能力

- 各种神奇的特性全靠插件
- 目前至少有200多个插件
- 可以对CSS进行各种不同的转换和处理
- 把繁琐复杂的工作交由程序去处理
- 把开发人员解放出来

postCSS常用插件

- 插件查询地址：www.postcss.parts/
- 常用用插件列列表：github.com/postcss/pos...
- autoprefixer – 自动加前缀
- postcss-import – 模块合并
- cssnano– 压缩代码
- postcss-cssnext – 使用css新特性
- precss– 变量、条件(if)、循环、Mixin Extend、import，属性值引用等
- css语法检查 兼容性检查

postCSS支持的构建工具

- CLI命令行工具
- webpack postcss-loader
- Gulp gulp-postcss
- Grunt grunt-postcss
- Rollup rollup-postcss

postCSS的安装及使用

https://github.com/postcss/postcss-cli
https://github.com/postcss/postcss#usage
https://www.npmjs.com/package/autoprefixer

使用方法一：PostCSS CLI命令行工具

- 安装: npm i -g | -D postcss-cli
- 运行:
  - 通过cd node_modules/.bin/进入node_modules/.bin/目录内再执行postcss input.css -o output.css
  - 或者在package.json中配置"scripts": {"test": postcss input.css -o output.css}}
- 配置文件: postcss.config.js

```
module.exports = {
  parser: 'sugarss',
  plugins: [
    require('postcss-import')({ ...options }),
```

```
    require('postcss-url')({ url: 'copy', useHash: true }),
  ],
}

==================== example ==========================
const autoprefixer = require('autoprefixer');
// const cssnano = require('cssnano');
// const atImport = require('postcss-import');
// const cssnext = require('postcss-cssnext');
// const precss = require('precss');
module.exports = {
    plugins: [
        // atImport,
        // cssnext,
        // precss
        autoprefixer({
            browsers:['Firefox > 1']
        }),
        // cssnano
    ]
};
```

使用方法二：结合构建工具一起使用

PostCSS结合Gulp应用

https://github.com/postcss/gulp-postcss

https://www.cnblogs.com/rongfengliang/p/8143200.html

- 安装：$ npm install --save-dev gulp-postcss
- npm i gulp gulp-postcss autoprefixer autoprefixer-core postcss-cssnext --save-dev -d
- 增加gulp配置文件gulpfile.js
- 运行: gulp postcss

```
// gulpfile.js
const gulp = require('gulp');
const autoprefixer = require('autoprefixer');
// const cssnano = require('cssnano');
const atImport = require('postcss-import');

gulp.task('postcss', function () {
    var postcss    = require('gulp-postcss');
```

```
9
10      return gulp.src('./src/')
11          .pipe(postcss([
12              atImport,
13              autoprefixer(),
14              // cssnano
15          ]))
16          .pipe(gulp.dest('./build/'));
17  });
```

PostCSS结合Webpack应用

https://webpack.js.org/loaders/postcss-loader/

- 安装: $ npm i -D style-loader css-loader postcss-loader
- $ npm install --save-dev style-loader css-loader postcss-loader
- 增加webpack配置文件webpack.config.js
- 运行: webpack postcss

```
1  module.exports = {
2      mode: 'production',
3      entry: {
4          index: './src/main.js',
5          login: './src/login.js',
6      },
7      output: {
8          filename: '[name].[hash].min.js',
9          path: path.resolve(__dirname, 'build')
10     },
11     devServer: {
12         port: '3000',
13         compress: true,
14         open: true,
15         hot: true
16     },
17     // 配置WEBPACK的插件
18     plugins: [
19         ...htmlPlugins,
20         new CleanWebpackPlugin(),
21         // 抽离CSS到单独的文件
22         new MiniCssExtractPlugin({
```

```
            filename: '[name].[hash].min.css'
        })
    ],
    // 配置WEBPACK的加载器LOADER
    module: {
        // 设置规则和处理方案  默认执行顺序：从右到左、从下向上
        rules: [{
            // 匹配哪些文件基于正则处理（此处是处理CSS/LESS文件）
            test: /\.(css|less)$/i,
            use: [
                // "style-loader", // 把处理好的CSS插入到页面中（内嵌式）
                MiniCssExtractPlugin.loader,
                "css-loader", // 处理@import/URL这种语法
                "postcss-loader", // 设置CSS前缀（处理兼容 需要搭配autoprefixer一起使用，需要而外再配置一些信息）
                "less-loader" // 把LESS编译为CSS
            ]
        }]
    }
};
```

browserList

https://caniuse.com/

https://www.npmjs.com/package/browserslist

BrowserList

- `last 2 versions` : the last 2 versions for each browser.
- `last 2 Chrome versions` : the last 2 versions of Chrome browser.
- `last 2 major versions` : all minor/patch releases of the current and previous major versions.
- `last 2 iOS major versions` : all minor/patch releases of the current and previous major versions of iOS Safari.
- `> 5%` or `>= 5%` : versions selected by global usage statistics.
- `> 5% in US` : uses USA usage statistics. It accepts two-letter country code.
- `> 5% in alt-AS` : uses Asia region usage statistics. List of all region codes can be found at `caniuse-lite/data/regions`.
- `> 5% in my stats` : uses custom usage data.
- `ie 6-8` : selects an inclusive range of versions.
- `Firefox > 20` : versions of Firefox newer than 20.
- `Firefox >= 20` : versions of Firefox newer than or equal to 20.
- `Firefox < 20` : versions of Firefox less than 20.
- `Firefox <= 20` : versions of Firefox less than or equal to 20.
- `Firefox ESR` : the latest [Firefox ESR] version.
- `iOS 7` : the iOS browser version 7 directly.
- `extends browserslist-config-mycompany` : take queries from `browserslist-config-mycompany` npm package.
- `unreleased versions` : alpha and beta versions of each browser.
- `unreleased Chrome versions` : alpha and beta versions of Chrome browser.
- `not ie <= 8` : exclude browsers selected before by previous queries.
- `since 2013` : all versions released since year 2013 (also `since 2013-03` and `since 2013-03-10`).

文章推荐:

http://julian.io/some-things-you-may-think-about-postcss-and-you-might-be-wrong/

https://davidtheclark.com/its-time-for-everyone-to-learn-about-postcss/

https://webdesign.tutsplus.com/series/postcss-deep-dive--cms-889

CSS模块化方案(css modules)

css-modules 和 extract-text-plugin 避免CSS类名冲突;

CSS modules使用

- 解决类名冲突的问题
- 使用postCSS或者webpack等构建工具进行编译
- 在HTML模板中使用编译过程中产生的类名

使用JS来引用、加载CSS好处

- JS作为入口，管理资源有天然优势
- 将组件的结构、样式、行为封装到一起，增强内聚
- 可以做更多处理(webpack)

webpack

- JS是整个应用的核心入口
- 一切资源均由JS管理依赖
- 一切资源均由webpack打包

webpack和css

- css-loader 将css变成js
- style-loader 将js样式插入head
- ExtractTextPlugin 将CSS从JS中提取出来
- css modules 解决CSS命名冲突的问题
- less-loader sass-loader 各类预处理器
- postcss-loader PostCSS处理

Vue框架中的CSS使用

- vue cli安装 ： npm install -g @vue/cli　OR　yarn global add @vue/cli
- vue cli创建项目 ： vue create hello-world
- ./node_modules/.bin/vue init webpack demo
- 检查其版本是否正确：vue --version
- 运行：npm build
- 模拟Scoped CSS:
  - 方案一：随机选择器　css modules
  - 方案二：随机属性 <div abcdefg> div[abcdefg]{ }

React框架中的CSS使用

- React cli安装 ： install -g create-react-app　全局安装
- react-cli创建项目工程 ： create-react-app demo
- ./node_modules/.bin/vue init webpack demo
- 运行项目 yarn start
- 暴露隐藏配置可修改的文件　yarn eject
- 打包项目 yarn build
- 官方没有集成方案
- 社区方案丛多
  - css modules
  - (babel)react-css-modules
  - styled-components
  - styled　jsx