1. PostCSS -> 对Css进行工程化处理

- `PostCSS`本身是一个功能比较单一的工具。它提供了一种方式用 `JavaScript` 代码来处理`CSS`。利用`PostCSS`可以实现一些工程化的操作,如:自动添加浏览器前缀、代码合并、代码压缩等。
- 官网: https://postcss.org
- 中文文档: https://www.postcss.com.cn/

- 安装:
  1. 安装`node`环境（https://nodejs.org/en/download/）
     - 通过`cmd` `node -v` 可查看`node`版本确定是否安装成功
     - 可以安装`nvm`，管理`node`版本
  2. `npm install postcss-cli -g`
     - `-g`为全局安装,可以在任意目录下使用`post`命令操作
     - `postcss`安装失败解决： `npm i postcss autoprefixer@8.0.0`，版本问题-更新版本
  3. `-o`、`-w`
     - 进入要转换`css`的目录,在此目录`cmd`
       - `postcss 要转换文件的路径.css -o 转换好的目标文件.css -w`
       - `postcss src/demo.css -o dist/demo.css -w`
         - 将`src`中的`demo.css`通过`postcss`转换到`dist`中
       - `-w`代表实时监听, 修改文件之后不用手动转换.
       - 断开程序： `ctrl + c` 重复两次(断开实时监听)
  4. `postcss.config.js`
     - `postcss`的配置文件，在`源文件.css`目录下创建`postcss.config.js`， `src/postcss.config.js`

2. PostCSS常用插件

- 进入官网点击 `Plugins`(插件)，可以搜索插件, 可以查看插件的使用方法
- 插件安装: 1. 在项目目录`cmd`， 2. `npm i 插件名`

- `autoprefixer` ： 给浏览器自动添加前缀
  - 配置页面:

```
const autoprefixer = require('autoprefixer');
module.exports = {
  plugins : [
    autoprefixer({
      browsers : [' > 0% ',] //给市场份额大于0%的浏览器添加前缀
    })
  ]
}
```

- `postcss-import`： 合并css

- `src/demo.css` ： 使用`@import './reset.css';`
- 配置页面:

```
const pcImport = require('postcss-import');

module.exports = {
  plugins : [
    pcImport // 放入 pcImport
  ]
}
```

- `cssnano` ， 对`css`进行压缩处理, 去除一些不必要的空格和回车,节省空间
  - 配置页面:

```
const cssnano = require('cssnano');

module.exports = {
  plugins : [
    cssnano
  ]
}
```

- `postcss-cssnext` ： 处理比较高级的`CSS`语法(一些高级语法,很多浏览器不支持,可以使用它来给`css`降级)
  - 配置页面:

```
const cssnano = require('cssnano');

module.exports = {
  plugins : [
    cssnano
  ]
}
```

    - `src/demo.css`(自己编写的)

```
:root{              /* :root里面定义的是变量 */
  --color:red;
  --vw:100%;
}
div{
  background: var(--color);   /* 通过var(变量名引入) */
  width: var(--vw);
}
```

    - `dist/demo.css`(生成的)

```
div{
  background: red;
  width: 100%;
}
```

- `stylelint` ： 代码规范检测
  - 配置页面:

```
const stylelint = require('stylelint');

module.exports = {
  plugins : [
    stylelint({  // 在此填写，编写css的规范，和eslint一样
      "rules" : {
          "color-no-invalid-hex" : true
      }
    })
  ]
}
```

    - 代码不规范, 会在`cmd`窗口报错代码不规范的具体位置
- `postcss-sprites` ： 精灵图(雪碧图)，默认将`源文件.css`中引入的图片都合成为一张雪碧图,放入`spritePath`指定的目录中
  - 配置页面:

```
const sprites = require('postcss-sprites');

module.exports = {
  plugins : [
    sprites({
      spritePath:'./dist'
    })
  ]
}
```

- **注意**：以上的配置信息都放在`module.exports`对象中的`plugins`数组中，使用`,`隔开