1.css预处理

sass>less>stylus

2.css后处理

clean-css:压缩css

AutoPrefixer:自动添加css3属性各浏览器前缀

Rework：取代stylus的插件话框架

Postcss

3.Postcss

PostCSS 一开始是从 AutoPrefixer 项目中抽象出来的框架，它本身并不对CSS做具体的业务操作，只是将CSS解析成抽象语法树（AST），样式的操作由之后运行的插件系统完成。正如其本身所言“Transforming styles with JS plugins”

4.oocss（面向对象css）

主要分为两个核心：分离结构与皮肤和分离容器和内容。

分离结构与皮肤:皮肤即一些重复的视觉特征，如边框、背景、颜色，分离是为了更多的复用；结构是指元素大小特征，如高度，宽度，边距等等。

分离容器和内容:打破容器内元素对于容器的依赖，元素样式应该独立存在。

5.SMACSS（模块化架构的可扩展css）

优点：按照不同的业务逻辑，将整个 CSS 结构化分更加细致，约束好命名，最小化深度，在编写的时候，使用SMACSS规范能够更好的组织好 CSS 文件结构和 class 命名。

基础(Base)：定义基础全局样式　eg：html,body,form{};

布局(Layout):将页面分为各个区域的元素块　eg：header{}　footer{}

模块(Module)：可复用的单元。在模块中需要注意的是选择器一律选择 class selector，避免嵌套子选择器，减少权重，方便外部覆盖。

状态(State):通过JS动态挂载到元素上，可以根据状态覆盖元素上特定的属性。

主题：可选的视觉外观。一般根据需求有颜色，字体，布局等等，实现是将这些样式单独抽出来，根据外部条件（ data 属性，媒体查询等）动态设置。

6.BEM

类名命名规则： Block__Element—Modifier

- Block 所属组件名称

- Element 组件内元素名称
- Modifier 元素或组件修饰符

其核心思想就是**组件化**。首先一个页面可以按层级依次划分未多个组件，其次就是单独标记这些元素。BEM通过简单的块、元素、修饰符的约束规则确保类名的唯一，同时将类选择器的语义化提升了一个新的高度。

```
.form { }
    .form--theme-xmas { }
    .form--simple { }
    .form__input { }
    .form__submit { }
    .form__submit--disabled { }
```

7.Css In Js

高度复用组件，用高度语义化的方案，不过会有一定的成本。

8.Css Modlue

选择用js来管理样式与元素的关联，Css Module通过为每个本地定义的类名动态创建一个全局唯一类名，然后注入到UI上。实现编写样式规则的局部模块化。

9.styled-components

完全的css-in-js方案。

```
// button
import styled from'styled-compenents'
const Button=styled.button`
   padding: 10px;
   ${props=>props.primary?'palevioletred':'white'};`
<Button>按钮</Button>
<Buttonprimary>按钮</Button>
```

总结

我们在开发的之前，面对各种技术方案，一定要选取并组合出最适合自己项目的方案，是选用传统的CSS预处理器， 还是选用 PostCSS？是全局手动维护模块，还是完全交个程序随机生成类名？都需要结合业务场景、团队习惯等等因素。另一方面，CSS 本身并无编程特性，但在其工程化技术的发展中缺不乏很多优秀的编程思想，无论是自定义的 DSL 还是基于 JS，这其中带给我们思考的正是“编译思想”。