现代化 CSS 方法论

卢创旭

- 移动验房-前端 -

Part 1 CSS 模块化

时长约 5 分钟

CSS 模块化规范

- 约定式规范: SMACSS、 BEM、 SUIT、 ACSS、 ITCSS
- 工具式规范: css modules、vue-scoped、css in js、js in css

约定式规范

基本思想：

- 通过约定来规范css的命名，以解决命名冲突问题
- 通过提供或推荐分层设计好的css代码，来达到样式复用的目的

典型实现方式：

- BEM：BEM 的意思就是块（block）、元素（element）、修饰符（modifier）
- ACSS：引入实现 ATOMIC CSS 思想的 tailwind css 框架

工具式规范

基本思想：

- 编译时将类名替换成全局唯一的带有作用域的类名，来达到模块化 css 局部作用域的效果。
- 缺点是需要搭配相关工具实现，有一定的学习成本。

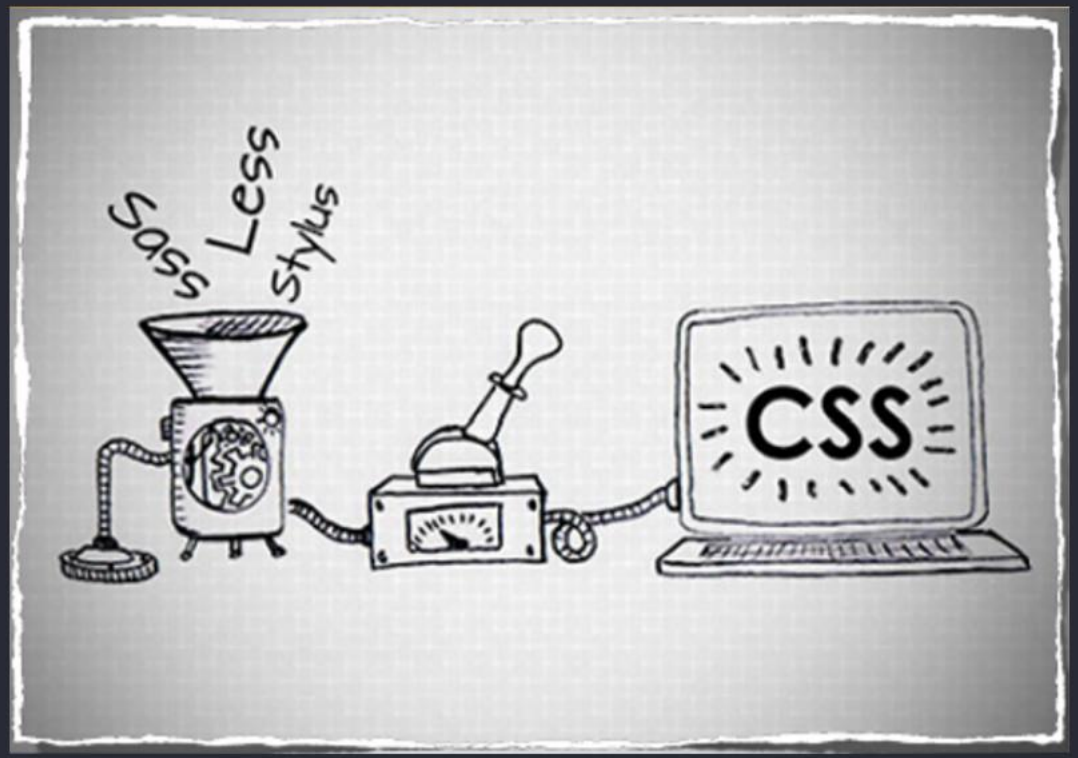
典型实现方式：

- css modules：通过 webpack 的 css-loader 中启用 css modules
- css in js：引入 styled-components 插件
- js in css：css hudini

Part 2 CSS 处理器

时长约 15 分钟

CSS 预处理器



“a css preprocessor is a program that lets you generate css from the preprocessor's own unique syntax.”

译：css预处理器是一个能让你通过预处理器自己独有的语法来生成css的程序。

预处理器变革

青铜时代（模板）

```
body {
  left: <%= left %>px;
  color: <%= color|height: 10% %>px;
}
```

特点

- 对 css 无感知
- 错误无法追踪
- 维护困难

白银时代（SASS）

sass是诞生最早，也是世界上最成熟、最稳定、最强大的专业级css扩展语言！

```
$font-stack: helvetica, sans-serif
$primary-color: #333

body
  font: 100% $font-stack
  color: $primary-color

// 转换后
body {
  font: 100% helvetica, sans-serif;
  color: #333;
}
```

特点

- 基于缩进，语法简洁，但可控性差
- 对于前端开发并不友好
- 实现对css的感知

黄金时代（CSS超集）

特点

- 对前端更加友好
- 不基于缩进，不简洁但更安全
- 与自定义DSL同样强大的语言能力
- 可以直接使用css书写

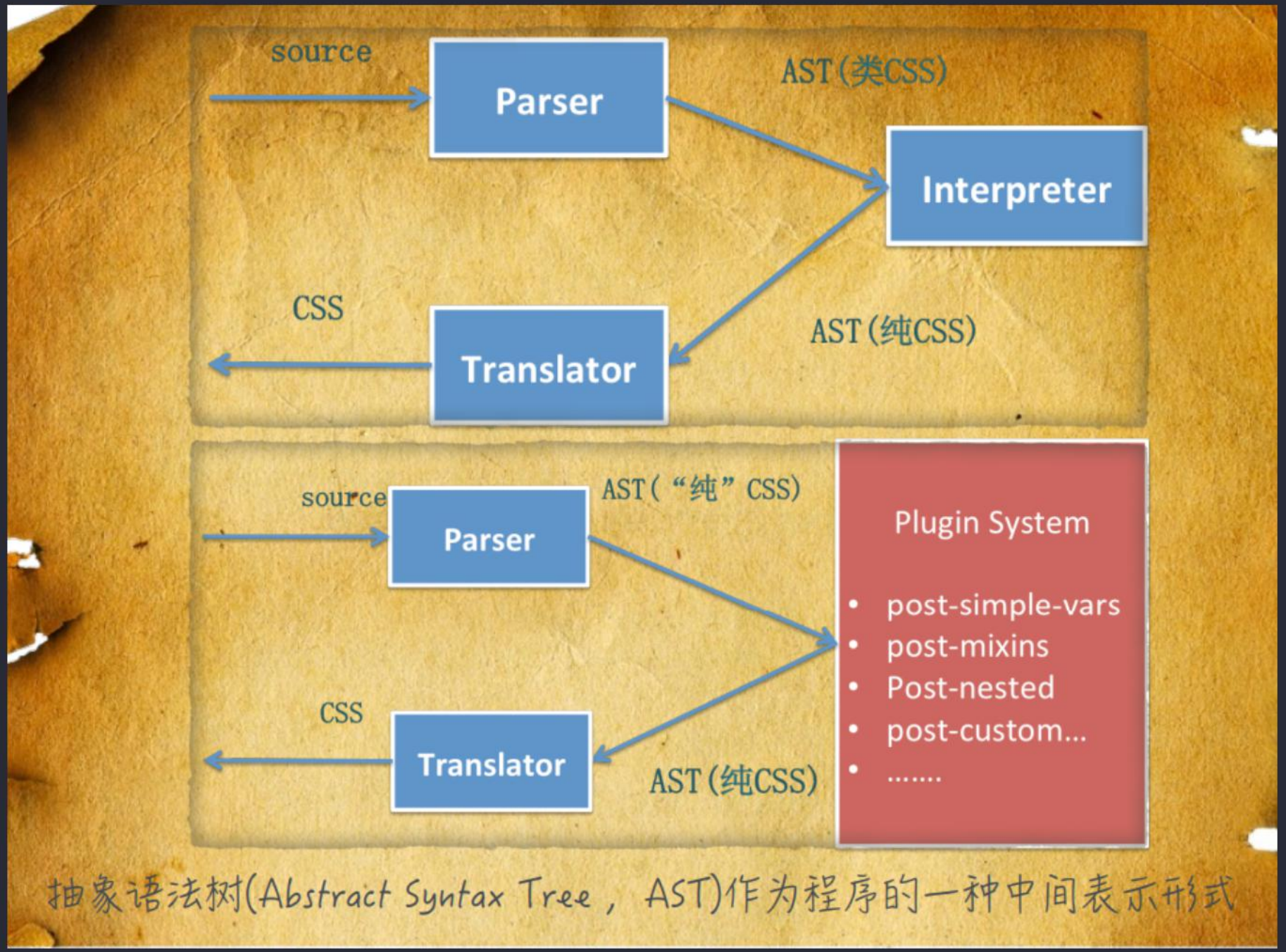
预处理器常用规范

- 变量
- 混合（mixin）extend
- 嵌套规则
- 运算
- 函数
- namespaces & accessors
- scope
- 注释

CSS后处理器

- css压缩 clean-css
- 自动添加浏览器前缀 autoprefixer
- css更加美观排序 csscomb
- rework取代stylus 后处理器发热
- 前后通吃 postcss


Sass
{less}


source
Parser
AST(类CSS)
Interpreter
CSS
Translator
AST(纯CSS)
source
Parser
AST(“纯”CSS)
Plugin System
post-simple-vars
post-mixins
Post-nested
post-custom...
.....
CSS
Translator
AST(纯CSS)
抽象语法树(Abstract Syntax Tree，AST)作为程序的一种中间表示形式

POSTCSS

css 界的 babel

安装使用

VUE-CLI

由于 vue-cli 脚手架默认使用 postcss@7，所以我们只需安装 postcss-import@12.0.0 postcss-preset-env@6.5.0

```
# postcss-import 用于支持 @import 引入 css
# postcss-preset-env 用于支持 css-next 特性

npm install -d postcss-import@12.0.0 postcss-preset-env@6.5.0
```

WEBPACK

```
npm install -d css-loader style-loader postcss
  postcss-loader postcss-import postcss-preset-env
```

```
10      port: 3000,
11    },
12    module: {
13      rules: [
14        {
15          test: /\.css$/,
16          use: [
17            "style-loader",
18            "css-loader",
19            "postcss-loader"
20          ],
21        },
22      ],
23    },
24    plugins: [
```

配置文件

在根目录新建 .postcssrc.js 文件

```
module.exports = {
  plugins: {
    "postcss-import": {},
    "postcss-preset-env": {
      stage: 3,
      features: {}
    },
  },
};
```

- postcss-preset-env
- postcss plugins

POSTCSS值得收藏的插件

- postcss-custom-properties 运行时变量
- postcss-utilities 包含常用mixins、shortcuts、helpers的工具集
- postcss-simple-vars 与scss一致的变量实现
- postcss-mixins 实现类似sass的@mixin的功能
- postcss-extend 实现类似sass的继承功能
- postcss-import 实现类似sass的import
- postcss-preset-env 面向未来语法

Part 3 CSS 魔法

时长约 20 分钟

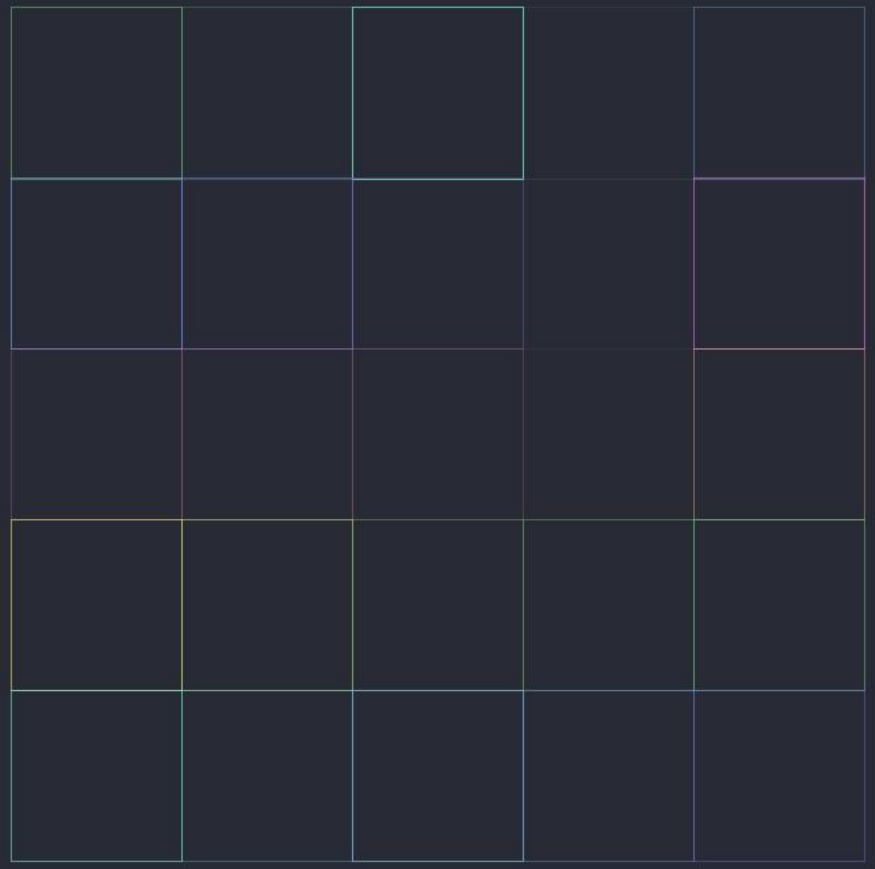
CSS-DOODLE

https://css-doodle.com

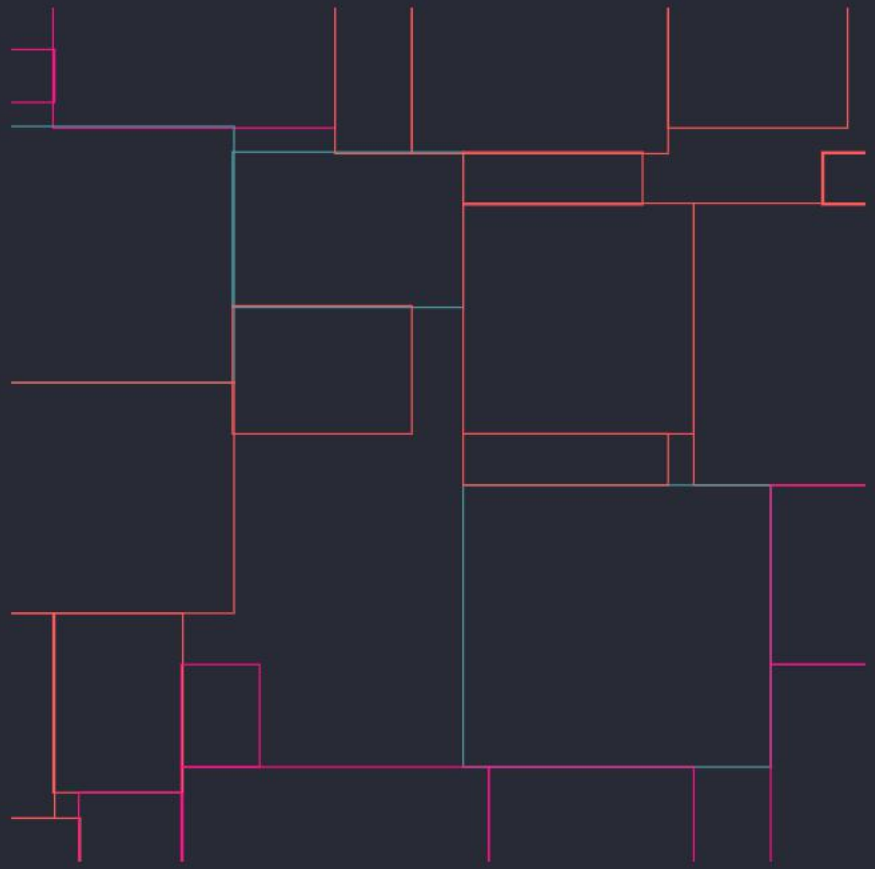
"A web component for drawing patterns with CSS."

译：一个用来绘制CSS图案的WEB组件。

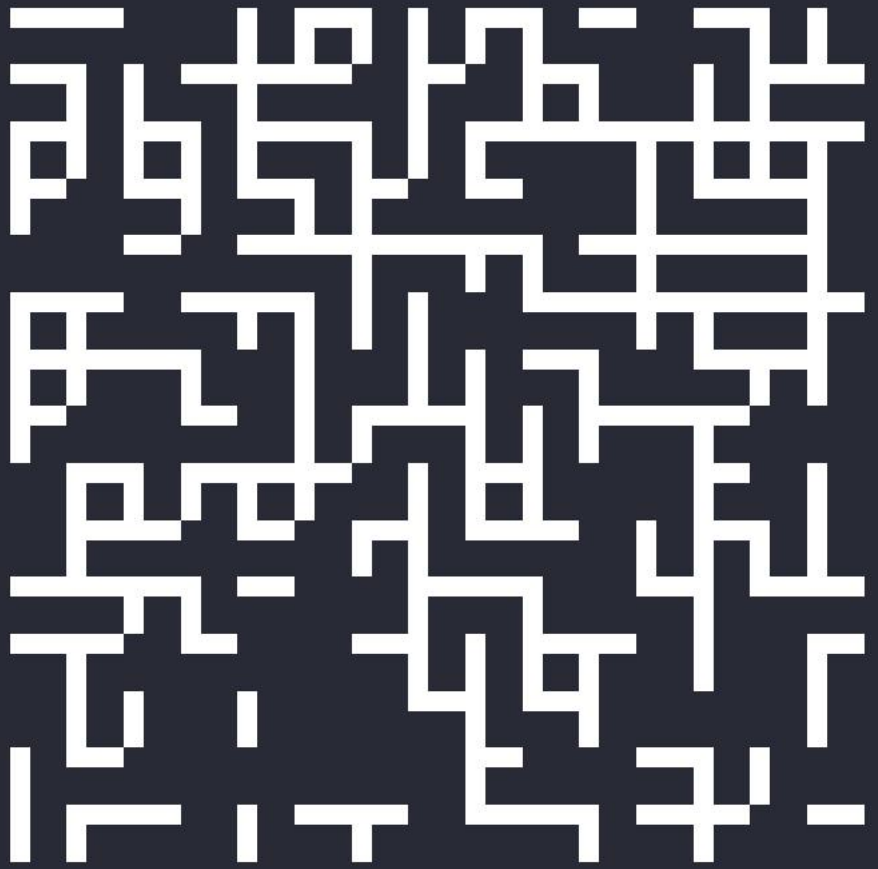
```
npm install css-doodle

/* import it */
import 'css-doodle';
```



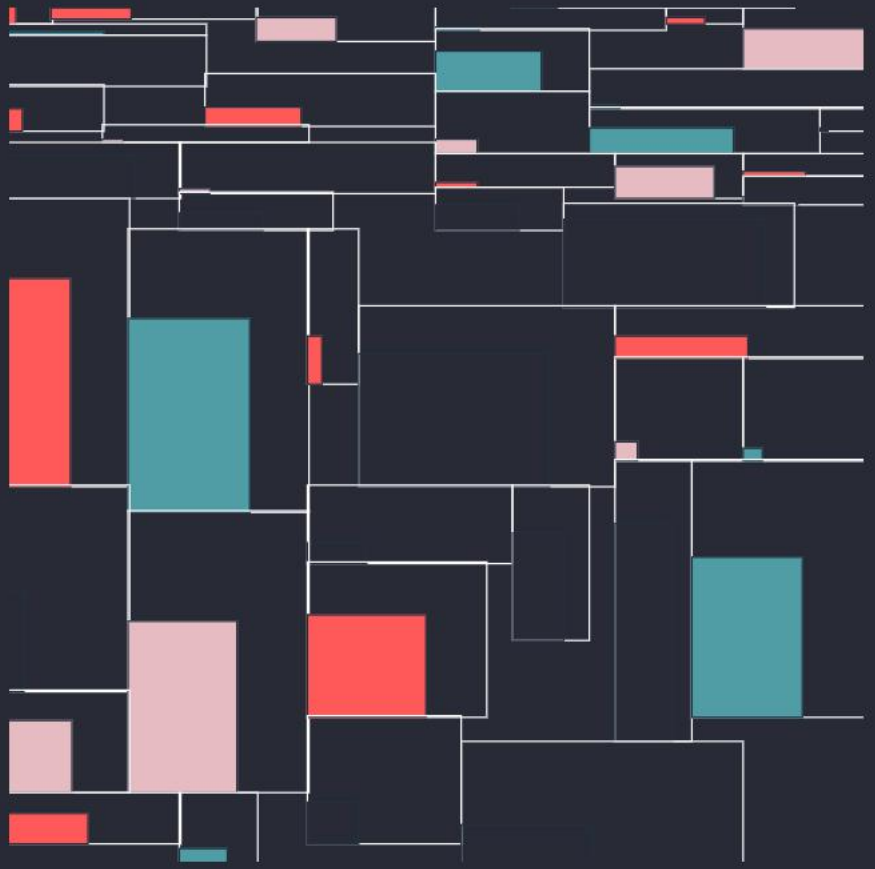
```
:doodle {
  @grid: 5 / 40vmin;
}

border-width: 1px;
border-style: solid;
border-color: hsla(
  calc(100 + 20 * @i), 70%, 68%, @r.8
);
margin: -.5px;
```



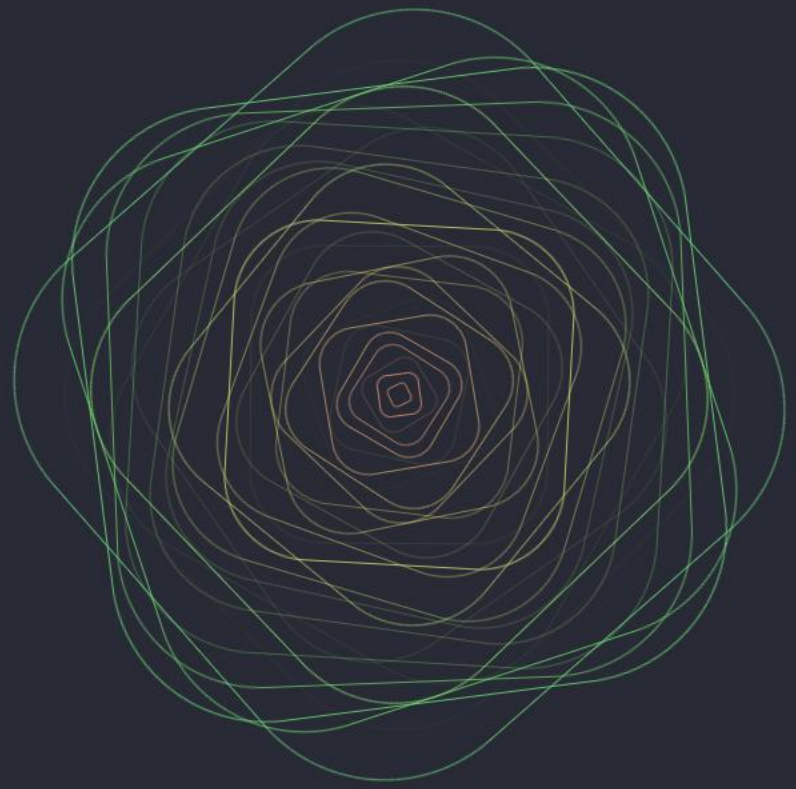
```
:doodle {
  @grid: 50x1 / 40vmin;
  overflow: hidden;
  grid-auto-flow: dense;
  grid-auto-rows: 1vmin;
  grid-template-columns:
    repeat(auto-fill, 1vmin);
}
:container {
  transform: scale(1.2);
}

margin: -.5px;
grid-row-end: span @ri(2, 12);
grid-column-end: span @ri(2, 12);
border: calc(@i/@I*2px) solid @p(
  transparent,
  #FF1489, #FF5A59, #509DA6
);
```



```
:doodle {
  @grid: 15 / 40vmin;
}

--s: 1vmin solid #fff;

@random {
  border-top: var(--s);
}

@random {
  border-left: var(--s);
}
```



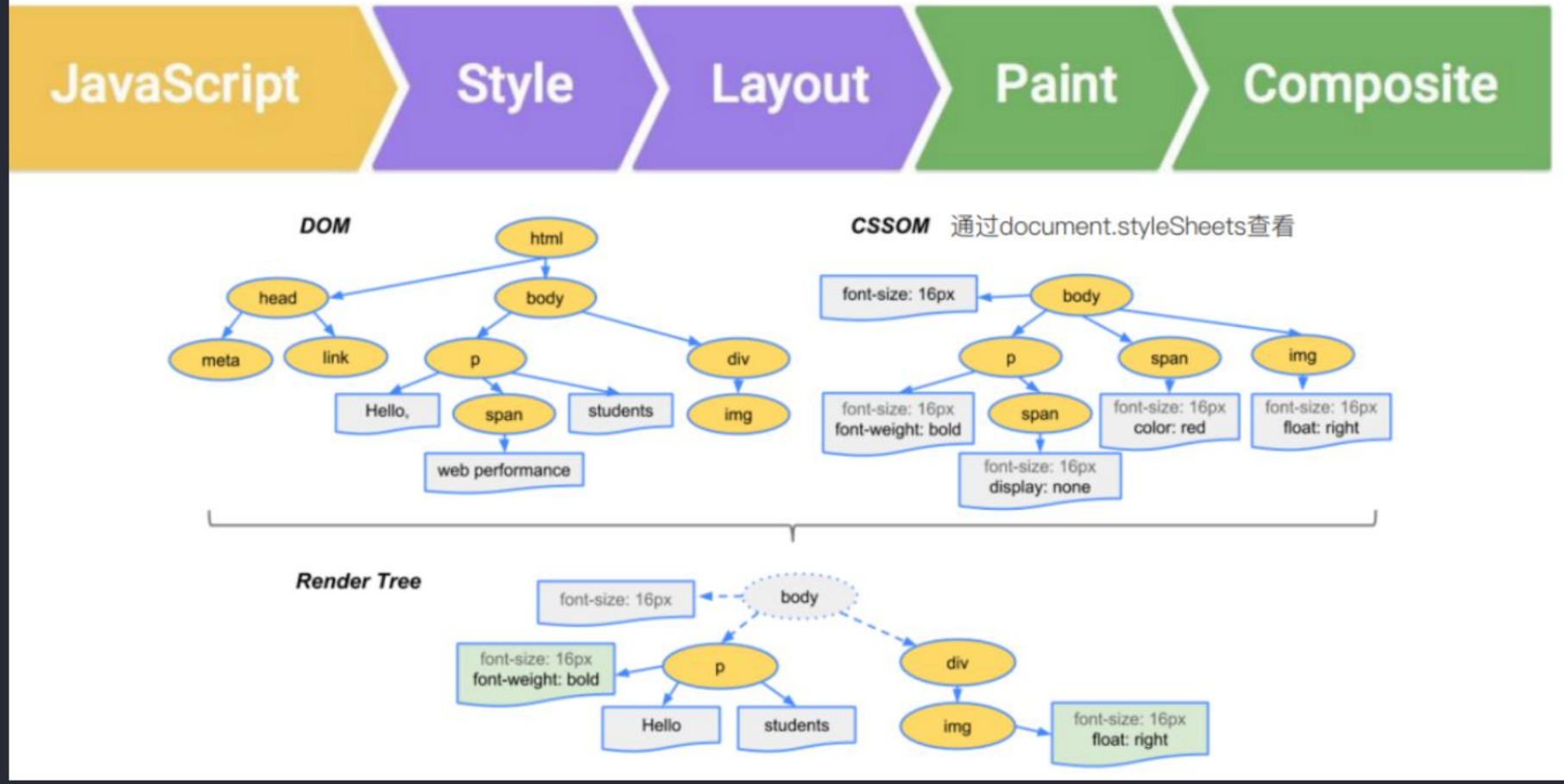
```
/* ... */

:after {
  position: absolute;
  content: '';
  border: 1px solid #223141;
  left: -1px;
  bottom: -1px;
  @size: @r(10%, 90%);
  background: @p(
    transparent,
    #FF1489, #FF5A59, #509DA6
  );
}
```



```
:doodle{
  @grid: 30 x 1 / 40vmin;
}
@size: calc(75% / @I * @i);
@place-cell: center;
border: 1px solid hsla(
  calc(10 + 4 * @i), 70%, 68%, @r.8
);
border-radius: 25%;
--d:@rand(20s,40s);
--rf:@rand(360deg);
--rt:calc(var(--rf) + @pick(1turn,-1turn));
animation:spin var(--d) linear infinite;
@keyframes spin {
    from{
        transform: rotate(var(--rf));
    }
    to{
        transform: rotate(var(--rt));
    }
}
```

CSS Houdini

Houdini API介绍



Houdini API介绍

在现今的 Web 开发中，JavaScript 几乎占据所有版面，除了控制页面逻辑与操作 DOM 对象以外，连 CSS 都直接写在 JavaScript 里面了，就算浏览器都还沒有实现的特性，总会有人做出对应的 Polyfills，让你快速的将新 Feature 应用到 Production 环境中，更别提我们还有 Babel 等工具帮忙转译。

而 CSS 就不同了，除了制定 CSS 标准规范所需的时间外，各家浏览器的版本、实战进度差异更是旷日持久，顶多利用 PostCSS、Sass 等工具來帮我們转译出浏览器能接受的 CSS。开发者们能操作的就是通过 JS 去控制 DOM 与 CSSOM来影响页面的变化，但是对于接下來的 Layout、Paint 与 Composite 就几乎沒有控制权了。

为了解决上述问题，为了让 CSS 的魔力不再浏览器把持，Houdini 就诞生了！（Houdini 是美国的伟大魔术师，擅长逃脱术，很适合想将 CSS 从浏览器中解放的概念）
CSS Houdini 是由一群來自 Mozilla, Apple, Opera, Microsoft, HP, Intel, IBM, Adobe 与 Google 的工程师所组成的工作小组，志在建立一系列的 API，让开发者能够介入浏览器的 CSS engine

Houdini APIs

“CSS Houdini是一组底层API，它们公开了CSS引擎的各个部分，从而使开发者可以通过这组API来扩展CSS。它让开发者拥有了直接访问CSSOM的能力，开发者可以通过这组API来编写浏览器可解析的CSS代码，这让开发者可以在不需要等待浏览器的实现的前提下实现自己想要的CSS功能。”

- Typed OM API
- Properties & Values API
- Paint API
- Layout API
- Animation worklet
- Parser API
- Font Metrics API

CSS Parser API

CSS Parser API 还没有被写入规范，所以下面我要说的内容随时都会有变化，但是它的基本思想不会变：允许开发者自由扩展 CSS 词法分析器，引入新的结构（constructs），比如新的媒体规则、新的伪类、嵌套、 @extends 、@apply 等等。

只要新的词法分析器知道如何解析这些新结构，CSSOM 就不会直接忽略它们，而是把这些结构放到正确的地方。

CSS Layout API

CSS Layout API允许开发者可以通过 CSS Layout API 实现自己的布局模块（layout module），这里的“布局模块”指的是display 的属性值。也就是说，这个 API 实现以后，开发者首次拥有了像 CSS 原生代码（比如display:flex 、display:table）那样的布局能力。

CSS Paint API

CSS Paint API Layout API 非常相似。它提供了一个 registerPaint 方法，操作方式和 registerLayout 方法也很相似。当想要构建一个CSS 图像的时候，开发者随时可以调用 paint() 函数，也可以使用刚刚注册好的名字。

Worklets实战

Is Houdini ready yet?

	Google Chrome	Microsoft Edge	Opera	Samsung Internet	Mozilla Firefox	Apple Safari	Spec
Engine	Blink				Gecko	WebKit	-
Paint API (Explainer Demos Article)	Shipped (Chrome 65) Details	Shipped (Edge 79) Details	Shipped (Opera 52) Details	Shipped (Internet 9.2) Details	Under consideration Details	In Development Details	Candidate Recommendation Spec
Properties & Values API (Demos Article)	Shipped (Chrome 78) Details	Shipped (Edge 79) Details	Shipped (Opera 65) Details	Shipped (Internet 12.0) Details	Under consideration Details	Partial support (Safari TP 67) Details	Working Draft Spec
Typed OM (Explainer Article)	Shipped (Chrome 66) Details	Shipped (Edge 79) Details	Shipped (Opera 53) Details	Shipped (Internet 9.2) Details	Under consideration Details	In Development Details	Working Draft Spec
Layout API (Explainer Demos)	Partial support (Canary) Details	Partial support (Canary) Details	Partial support (Developer) Details	No signal	Under consideration Details	Under consideration Details	First Public Working Draft Spec
AnimationWorklet (Explainer Demos Article)	Partial support (Chrome 71) Details	Partial support (Edge 79)	Partial support (Opera 58)	Partial support (Internet 10.2)	No signal	Under consideration Details	First Public Working Draft Spec
Parser API (Explainer)	No signal	No signal	No signal	No signal	No signal	No signal	Proposal Spec
Font Metrics API (Explainer)	No signal	No signal	No signal	No signal	No signal	No signal	Proposal Spec

Maintained by Surma, source code & license in the repository on GitHub, last updated on Fri May 21 2021 14:08:41 GMT+0000 (Coordinated Universal Time)

Part 4 Q&A

资料来源：

- https://blog.techbridge.cc/2017/05/23/css-houdini
- https://mp.weixin.qq.com/s/Zqw8f5jX6MmeURRzc44C6Q
- https://www.cnblogs.com/cyn941105/p/5590239.html
- https://yuanchuan.dev/talk/generative-art-with-css
- https://www.qed42.com/insights/coe/javascript/building-powerful-custom-properties-css-houdini
- https://github.com/GoogleChromeLabs/houdini-samples
- https://csstools.github.io/postcss-preset-env
- https://zhuanlan.zhihu.com/p/141725118
- https://www.postcss.com.cn/
- https://css-doodle.com