# 09 | CSS语法：除了属性和选择器，你还需要知道这些带@的规则

2019-02-07 winter

你好，我是winter。

今天我们进入CSS的学习。CSS是前端工程师几乎每天都要用的技术了，不过CSS的学习资料却是最糟糕的，这是因为CSS并没有像HTML和JavaScript那样的一份标准文档。

如果我们到W3C的网站上搜索看看，可以得到一些信息：

https://www.w3.org/TR/?title=css

在这里，我们一共看到了98份CSS相关的标准，它们各自从一些角度规定了CSS的特性。

这里我们暂且去掉Working Draft状态的标准，可以得到22份候选标准和6份推荐标准。

既然我们的专栏内容强调去系统性学习CSS，于是，面对这22+6份标准，我们就又需要一条线索，才能把这些离散的标准组织成易于理解和记忆的形式。

在这样的需求下，我找到的线索就是CSS语法，任何CSS的特性都必须通过一定的语法结构表达出来，所以语法可以帮助我们发现大多数CSS特性。

CSS语法的最新标准，你可以戳这里查看：

https://www.w3.org/TR/css-syntax-3/

这篇文档的阅读体验其实是非常糟糕的，它对CSS语法的描述使用了类似LL语法分析的伪代码，而且没有描述任何具体的规则。

这里你就不必自己去阅读了，我来把其中一些有用的关键信息抽取出来描述一下，我们一起来看看。

我们拿到这份标准可以看到，去除空格、HTML注释等无效信息，**CSS的顶层样式表由两种规则组成的规则列表构成，一种被称为 at-rule，也就是at 规则，另一种是 qualified rule，也就是普通规则。**

at-rule由一个 @ 关键字和后续的一个区块组成，如果没有区块，则以分号结束。这些at-rule在开发中使用机会远远小于普通的规则，所以它的大部分内容，你可能会感觉很陌生。

这些at规则正是掌握CSS的一些高级特性所必须的内容。qualified rule则是指普通的CSS规则，也就是我们所熟识的，由选择器和属性指定构成的规则。

## at 规则

好了，现在我们已经知道了，CSS语法的整体结构，接下来我们要做的是一个体力活，从所有的CSS标准里找到所有可能的 at-rule（不用谢，我已经帮你找好了，如果页面定位不准，你可以打开页面搜索关键字）。

- @charset： https://www.w3.org/TR/css-syntax-3/
- @import：https://www.w3.org/TR/css-cascade-4/
- @media：https://www.w3.org/TR/css3-conditional/
- @page： https://www.w3.org/TR/css-page-3/
- @counter-style：https://www.w3.org/TR/css-counter-styles-3
- @keyframes：https://www.w3.org/TR/css-animations-1/
- @fontface：https://www.w3.org/TR/css-fonts-3/
- @supports：https://www.w3.org/TR/css3-conditional/
- @namespace：https://www.w3.org/TR/css-namespaces-3/

这里的每一种@规则背后，都是一组CSS的知识。在我们的课程中，有些会重点介绍，不过，为了先给你建立起一个整体的认知，我们这里会给所有的@规则提供一些简单的例子和介绍。

## @charset

@charset用于提示CSS文件使用的字符编码方式，它如果被使用，必须出现在最前面。这个规则只在给出语法解析阶段前使用，并不影响页面上的展示效果。

```
@charset "utf-8";
```

## @import

@import用于引入一个CSS文件，除了@charset规则不会被引入，@import可以引入另一个文件的全部内容。

```
@import "mystyle.css";
@import url("mystyle.css");
```

```
@import [ <url> | <string> ]
      [ supports( [ <supports-condition> | <declaration> ] ) ]?
      <media-query-list>? ;
```

通过代码，我们可以看出，import还支持 supports 和media query形式。

## @media

media就是大名鼎鼎的media query使用的规则了，它能够对设备的类型进行一些判断。在media的区块内，是普通规则列表。

```
@media print {
    body { font-size: 10pt }
}
```

## @page

page用于分页媒体访问网页时的表现设置，页面是一种特殊的盒模型结构，除了页面本身，还可以设置它周围的盒。

```
@page {
  size: 8.5in 11in;
  margin: 10%;

  @top-left {
    content: "Hamlet";
  }
  @top-right {
    content: "Page " counter(page);
  }
}
```

### @ counter-style

counter-style产生一种数据，用于定义列表项的表现。

```
@counter-style triangle {
  system: cyclic;
  symbols: ;
  suffix: " ";
}
```

### @ key-frames

keyframes产生一种数据，用于定义动画关键帧。

```
@keyframes diagonal-slide {

  from {
    left: 0;
    top: 0;
  }

  to {
    left: 100px;
    top: 100px;
  }

}
```

### @ fontface

fontface用于定义一种字体，icon font技术就是利用这个特性来实现的。

```
@font-face {
  font-family: Gentium;
  src: url(http://example.com/fonts/Gentium.woff);
}

p { font-family: Gentium, serif; }
```

### @ support

support检查环境的特性，它与media比较类似。

### @ namespace

用于跟XML命名空间配合的一个规则，表示内部的CSS选择器全都带上特定命名空间。

### @ viewport

用于设置视口的一些特性，不过兼容性目前不是很好，多数时候被html的meta代替。

#### 其它

除了以上这些，还有些目前不太推荐使用的at规则。

- @color-profile 是 SVG1.0 引入的CSS特性，但是实现状况不怎么好。

- @document 还没讨论清楚，被推迟到了CSS4中。
- @font-feature-values 。todo查一下。

### 普通规则

接下来我们进入qualified rule，也就是普通规则的部分，看看这里有什么需要我们记住的内容。

qualified rule主要是由选择器和声明区块构成。声明区块又由属性和值构成。我在下面的列表中，介绍了这部分语法的组成要点。

- 普通规则
  - 选择器
  - 声明列表
    - 属性
    - 值
      - 值的类型
      - 函数

#### 选择器

我们先来看看选择器，它有一份独立的标准，我们可以参考这个网址：

https://www.w3.org/TR/selectors-4/

这份标准不在我们前面的过滤条件中，它属于CSS和HTML共用的标准。

关于选择器的叠加规则等知识我们后文会专门的一节课程来讲，这里我们就从语法的角度介绍一下选择器。

在选择器标准的最后，附有一张选择器的语法表，从这份语法表，我们可以理清楚记忆选择器的思路。

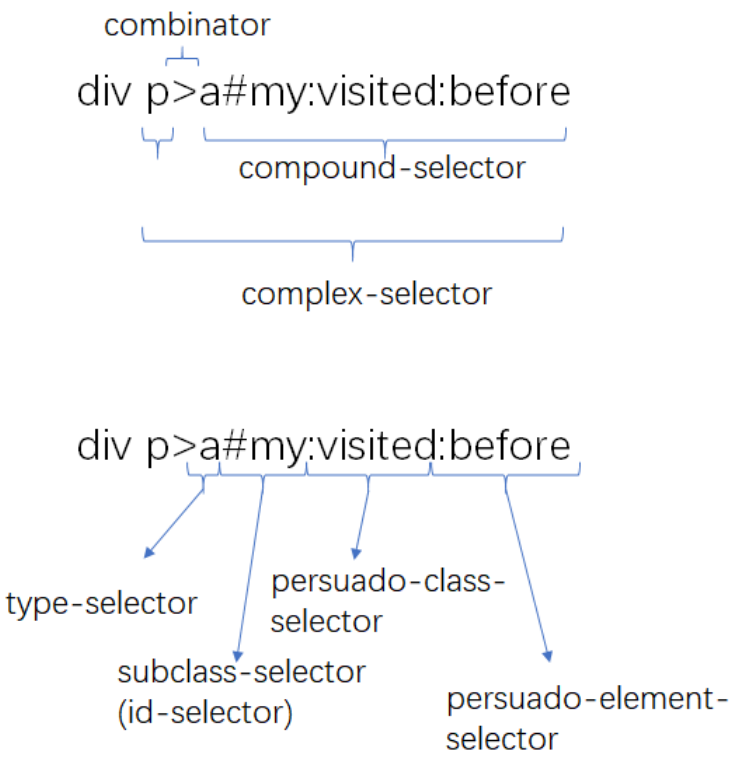
我们从语法结构可以看出，任何选择器，都是由几个符号结构连接的：空格、大于号、加号、波浪线、双竖线，这里需要注意一下，空格，即为后代选择器的优先级较低。

然后对每一个选择器来说，如果它不是伪元素的话，由几个可选的部分组成，标签类型选择器，id、class、属性和伪类，它们中只要出现一个，就构成了选择器。

如果它是伪元素，则在这个结构之后追加伪元素。只有伪类可以出现在伪元素之后。我在下面用一个列表（不太严谨地）整理了选择器的语法结构：

- complex-selector
  - combinator
    - 空格
    - >
    - +
    - ~
    - ||
  - compound-selector
    - type-selector
    - subclass-selector
      - id
      - class
      - attribute
      - pseudo-class
    - pseudo-element

我们在这里可以参考一个示例图：




（语法结构分析示例）

看完了选择器，我们继续来看看声明部分的语法。

## 声明：属性和值

声明部分是一个由“属性:值”组成的序列。

**属性**是由中划线、下划线、字母等组成的标识符，CSS还支持使用反斜杠转义。我们需要注意的是：属性不允许使用连续的两个中划线开头，这样的属性会被认为是CSS变量。

在CSS Variables标准中，以双中划线开头的属性被当作变量，与之配合的则是 var 函数：

```
:root {
  --main-color: #06c;
  --accent-color: #006;
}
/* The rest of the CSS file */
#foo h1 {
  color: var(--main-color);
}
```

**值**的部分，主要在标准 CSS Values and Unit，根据每个CSS属性可以取到不同的值，这里的值可能是字符串、标识符。

CSS属性值可能是以下类型。

- CSS范围的关键字：initial，unset，inherit，任何属性都可以的关键字。
- 字符串：比如content属性。
- URL：使用url() 函数的URL值。
- 整数/实数：比如flex属性。
- 维度：单位的整数/实数，比如width属性。
- 百分比：大部分维度都支持。
- 颜色：比如background-color属性。
- 图片：比如background-image属性。
- 2D位置：比如background-position属性。
- 函数：来自函数的值，比如transform属性。

这里我们要重点介绍一下函数。一些属性会要求产生函数类型的值，比如easing-function会要求cubic-bezier()函数的值：

CSS支持一批特定的计算型函数：

- calc()
- max()
- min()
- clamp()

- toggle()
- attr()

**calc()**函数是基本的表达式计算，它支持加减乘除四则运算。在针对维度进行计算时，calc()函数允许不同单位混合运算，这非常的有用。

例如：

```
section {
  float: left;
  margin: 1em; border: solid 1px;
  width: calc(100%/3 - 2*1em - 2*1px);
}
```

**max()、min()和clamp()**则是一些比较大小的函数，max()表示取两数中较大的一个，min()表示取两数之中较小的一个，clamp()则是给一个值限定一个范围，超出范围外则使用范围的最大或者最小值。

toggle()函数在规则选中多于一个元素时生效，它会在几个值之间来回切换，比如我们要让一个列表项的样式圆点和方点间隔出现，可以使用下面代码：

```
ul { list-style-type: toggle(circle, square); }
```

attr()函数允许CSS接受属性值的控制。

## 总结

在这一部分，我们介绍了CSS语法的总体结构，CSS的语法总体结构是由两种规则列表构成，一种是at 规则，另一种是普通规则。

在at规则中，我举了13个以上的例子，并逐个进行了简单的介绍。而在普通规则的部分，我介绍了选择器和声明区块是普通规则的主要组成部分。

并且，我给出了一个（不太严谨）的选择器语法结构，声明区块则由属性和值构成，这一部分我们重点介绍了函数。

从整体上去掌握内容，再去定位到单个细节，这对于我们学习CSS有非常重要的提示作用。

最后，给你留一个思考问题，CSS的函数有很多，本文也提到了不少，请你也一起查阅资料，试着总结一下，你能找到多少种CSS函数？


精选留言

Carson

18

在网站上搜索了一下，发现 css 函数有不少，尤其是近三年，增加的函数几乎超过过去的总和。

按照 winter 老师提到「知识完备性」的思路，尝试整理了一下 CSS 函数。

按照功能，分成以下 5 个类别（可能并不完全准确）：

# 1. 图片

* filter
* blur()
* brightness()
* contrast()
* drop-shadow()
* grayscale()
* hue_rotate()
* invert()
* opacity()
* saturate()
* sepia()
* cross-fade()
* element()
* image-set()

* imagefunction()

# 2. 图形绘制
* conic-gradient()
* linear-gradient()
* radial-gradient()
* repeating-linear-gradient()
* repeating-radial-gradient()
* shape()

# 3. 布局
* calc()
* clamp()
* fit-content()
* max()
* min()
* minmax()
* repeat()

# 4. 变形/动画
* transform
* matrix()
* matrix3d()
* perspective()
* rotate()
* rotate3d()
* rotateX()
* rotateY()
* rotateZ()
* scale()
* scale3d()
* scaleX()
* scaleY()
* scaleZ()
* skew()
* skewX()
* skewY()
* translate()
* translate3d()
* translateX()
* translateY()
* translateZ()

# 5. 环境与元素
* var()
* env()
* attr()

2019-02-07

Sevens 些粉 3

推荐一下《css世界》这本书，有理论基础也有实战应用和常遇坑，看了两章感觉不错。

2019-02-08

文全 3

@import 用于引入一个 CSS 文件，除了 @charset 规则不会被引入，@import 可以引入另一个 JavaScript 文件的全部内容。这段写错了 应该是css 文件全部内容

2019-02-07

莲 0

我看到winter老师讲解这些冷门的知识，忽然意识到什么叫做精通？要精通就要抠这种细节，这样才能做到精通

做就要做精通，前端是一种手艺人

2019-02-09



mimof9 0

试了一下 toggle这个函数 并没有效果。clac实测下来是有效果的。

2019-02-08

wenxueliu 0

每篇文章希望增加一些预备知识，后端程序员表示看不懂

2019-02-08

Aaaaaaaaaayou 0

“只有伪类可以出现在伪元素之后”是不是写反了

2019-02-07

hhk 0

css语法：at 规则 + 普通规则
普通规则：选择器 + 声明区块

另外，margin 的读音好像读错了

2019-02-07

Rocky 0

@import 那段写错了，不是引入 JavaScript， 是 css。

2019-02-07