

注释

如何写 CSS 注释，作用是什么？

以`/`开头，`/`结束。可阻止其中 CSS 解析，给代码增加说明，提升可读性

如何去除 CSS 注释？

- 用正则`/\/[^\s\S]?*\//`全局匹配注释，替换为空字符串
- 用工程化工具，如`cssnano`来去除注释

选择器

CSS 中哪些选择器？

选择器	示例
通配符选择器	<code>*</code>
ID选择器	<code>#id</code>
类选择器	<code>.class</code>
元素选择器	<code>div</code>
属性选择器	<code>[attr="value"]</code>
伪类选择器	<code>a:hover</code>
伪元素选择器	<code>a::before</code>

CSS 选择器有哪些组合方式？

① 运算符组合选择器（关系选择器）

选择器	示例
后代选择器	<code>#id a</code>
子代选择器	<code>#id > a</code>
相邻兄弟选择器	<code>#id + a</code>
通用兄弟选择器	<code>#id ~ a</code>

② 属性组合选择器

选择器	示例
标签属性选择器	input[type="button"]
类属性选择器	.class[type="button"]
通配符属性选择器	*[type="button"]

③ 群组选择器

选择器	示例
交集选择器	.class#id, div#id
并集选择器	#id, .class, div

对比伪类，伪元素

伪类和伪元素都是选择器，适用于：

- 无法更改 HTML
- 要选择的元素不固定
- 要选择的元素状态或位置关系固定

示例：
::first-line可以选中块级元素的第一行。由于内容、设备宽度变化，第一行的内容也在变化。比起更改 HTML，直接定位某个元素的第一行在实际应用中更加方便。

两者区别如下：

伪类

用于选择元素的特定状态。单冒号开头。分为：

- 位置关系伪类举例：

选择器	选取范围
:first-child	第一个元素
:last-child	最后一个元素
:first-of-type	第一个指定类型的元素
:last-of-type	最后一个指定类型的元素

- 用户行为伪类举例：

选择器	选取范围
:hover	被用户鼠标指向
:checked	被用户选中

- 其它状态伪类举例：

选择器	选取范围
:enabled	当前元素可用
:lang(en)	选取语言为英文的元素

伪元素

用于选择元素的 特定部分。双冒号开头。举例：

选择器	选取范围	用途
::first-letter	第1行的第1字母	首字下沉
::first-line	第1行	首行缩进
::before	元素的第一个子元素	配合 content 使用
::after	元素的最后一个子元素	配合 content 使用
::placeholder	表单默认文本	更改提示样式

为兼容早期浏览器，实际使用时，多将双冒号写作单冒号

原因：早期规范没有明确伪元素必须双冒号，当时浏览器多用单冒号，现代浏览器同时兼容单和双冒号

如果不考虑兼容，规范应使用双冒号

CSS 选择器命名规则

大小写字母，数字，连字符 (-)，下划线 (_) 以及 ISO 10646 字符编码 U+00A1 及以上

- 不能以数字、连字符 (-) 开头
 - 类名或 ID 以数字、连字符 (-) 开头.或#后加\3
- ID 选择器以一个 (#) 开头
- 类选择器以一个 (.) 开头

什么是 CSS 无效选择器？

无效选择器，即运行环境无法解析的 CSS 选择器

选择器	示例
命名错误	.1, .-, ..class, ##id
运算符未知	#id >> a, #id ++ a
包含无效选择器的群组选择器	#id, div, ..class
不兼容的选择器	如 IE8 不兼容 CSS3 选择器

运行环境会跳过无效选择器

群组选择器包含无效选择器，该群组选择器都被跳过

开发者工具中，调试 CSS 时，现代浏览器会自动更正错误命名

选择器	更正后
..class	.class
div >> p	div > p
.1	.\31
#-1	#\31

无效选择器有什么用途？

利用浏览器跳过无效选择器和无效属性的特性

无效选择器，无效属性常用于区分运行环境（多为浏览器），进而：

- 改善 CSS 兼容性。常用的 CSS Hacks：

① 选择器

属性	适配	示例
*	IE6	*div { }
*+	IE6-IE7	*+div { }
后缀\9	IE6-IE7	div\9 { }
\0前缀	IE8	\0div { }
后缀\0	IE8-IE10	div\0 { }

② 属性

属性	适配	示例
-	IE6	-width
*/+	IE6-IE7	*width +width
_	IE6-IE9	_width
\0	IE8-IE10	width\0
\9	IE6-IE10	width\9
\9\0	IE9-IE10	width\9\0

- 使用运行环境的 私有属性，渐进性增强样式：

私有前缀	适配
-webkit-	webkit 引擎的 Safari、Blink 引擎的 Chrome、Opera、Edge, EdgeHTML 引擎的 Edge 部分支持
-moz-	Gecko 引擎 Firefox
-ms-	Trident 引擎的 IE 和 EdgeHTML 引擎的 Edge
-o-	Presto 引擎的 Opera

什么是渐进性增强样式？

通过组合 CSS hacks 和私有属性，在较低版本的浏览器中，保持页面布局和基本样式，在高版本浏览器，使用更高级属性或运行环境的私有属性，增强页面样式，提升用户体验。

渐进性增强，是改善 CSS 兼容性，开发框架的设计思想之一，核心是：

- 低版本的用户少的浏览器，保障基本功能，舍弃不必要的 CSS hacks 和 Polyfills
- 高版本，利用新属性，新接口，提高性能和体验

示例：

```
<style>
div { /* All */
-width: expression(this.width > 500 ? '500px' : 'auto'); /* IE6 */
max-width: 500px; /* IE7+ */
color: black; /* IE6 黑 */
}
p + div { /* IE7+ */
color: blue; /* IE7 - IE8 蓝 */
}
:not(p) { /* IE9+ */
color: red; /* IE9+ 红 */
-webkit-font-smoothing: antialiased; /* webkit浏览器 MacOS 平滑文字 */
-moz-osx-font-smoothing: grayscale; /* 火狐浏览器 MacOS 平滑文字 */
}
</style>
<p>p</p>
<div>div</div>
```

如何优化选择器，提高性能？

- CSS 选择器效率从高到底：

ID > 类 > 类型（标签） > 相邻 > 子代 > 后代 > 通配符 > 属性 > 伪类

- CSS 选择器的读取顺序：从右到左，逐级匹配

示例：

.content * {}，会先匹配到所有元素，再向上筛选出父元素 className 为 content 的子元素

根据效率和读取顺序，提高性能的方法如下：

1. 多使用高效率选择器

工作中，多使用类名，少使用关系、伪类选择器，可读和可维护性也会提高

2. 减少选择器层级

工作中，选择器层次不建议超过 4 级。用 SASS 或 LESS 时，应避免不必要的嵌套

3. 高效率选择器在前

工作中，避免使用类、标签选择器限制 ID 选择器，避免使用标签选择器限制类选择器。

低效率选择器在前，通常可以优化。示例 `.content #id {}` 可直接写成 `#id {}`，`div .content` 可以给 `div` 起类名

4. 避免使用通配符选择器

通配符选择器效率低，且使用 `*` 的场景，通常有其他解决方案

5. 多用 继承

示例：

```
<style>
div * { /* bad case */
font-size: 12px;
}
div { /* good case */
font-size: 12px;
}
</style>

<div>
<p></p>
<p></p>
<p></p>
</div>
```

继承

什么是继承？

CSS 属性分为非继承属性和 继承属性，继承属性的默认值为父元素的该属性的 计算值，非继承属性和根元素的继承属性的默认值为初始值。

对于非继承属性，可以显示的声明属性值为 `inherit`，让子元素的属性继承父元素。

常见的继承属性：

- 字体 font 系列
- 文本 text-align text-indent line-height letter-spacing
- 颜色 color
- 列表 list-style
- 可见性 visibility
- 光标 cursor

容易被误认为继承属性的 非继承属性：

- 透明度 opacity
- 背景 background系列

如何重置元素的属性值到初始值？

属性值initial可以将属性设为W3C规范初始值

属性all可以将元素的所有属性重置

在规范之外，浏览器还会为部分元素，如表单元素设置默认样式

属性的值来源于开发者定义，用户配置和浏览器默认

all:initial相当于清空了用户配置和浏览器默认样式

工作中，我们更希望重置到默认样式，而不是清空它们

all:revert属性还原。可以将子元素的属性重置按如下规则重置：

- 继承属性：重置到父元素的属性值
- 非继承属性或父元素继承属性都未设置：重置到用户配置和浏览器默认属性

```
<style>
button {
  color: yellow;
  border: 1px solid red;
  background-color: red;
}
button:nth-of-type(2) {
  all: initial; /* 清空按钮的样式 */
  color: blue;
}
button:last-of-type {
  all: revert; /* 保留按钮的默认样式 */
  color: blue;
}
</style>
<button>按钮1</button>
<button>按钮2</button>
<button>按钮3</button>
```

按钮1

按钮2

按钮3

优先级

为什么要定义 CSS 优先级？

同一元素可能会被多个 CSS 选择器选中。

选择器中可能包含对相同属性的不同设置值。

需要定义优先级规则，只让优先级最高的选择器的设置值生效。

CSS 优先级规则是什么？

选择器与元素的相关度越高，优先级越高，具体规则如下：

- 开发者定义选择器 > 用户定义选择器 > 浏览器默认选择器

- 内联样式 (style="") > 内 (<style>) 、外部样式 (<link />)
- ID 选择器 > 类选择器、属性选择器、伪类选择器 > 类型选择器、伪元素选择器
- 相同优先级，书写顺序后 > 前
- 同级选择器，复合选择器 > 单选择器
- 自身的选择器 > 继承自父级的选择器
- 用户配置 !important 声明 > 开发者 !important 声明 > 其它

!important 的作用和弊端，如何避免？

- 作用
!important 可以忽略选择器 CSS 选择器优先级，让声明的属性总是生效
- 弊端
 - 破坏原 CSS 级联规则，增加调试难度
 - 修改样式变得困难
 - 污染全局样式
- 避免
 - 用 CSS 选择器优先级解决样式冲突
 - 不在全局、会被复用的插件中使用 !important
 - 通过 CSS 命名或 Shadow DOM 限制 CSS 作用域

如何计算 CSS 选择器的优先级？

我们将 CSS 选择器优先级量化为权重，为不同类型的 CSS 选择器设置初始权重。

选择器的组合，即初始权重的累加。累加值越高，优先级越高。

初始权重：

选择器	权重
行间 style=""属性	1000
ID选择器	100
类、属性、伪类	10
类型（标签）、伪元素	1

可以重复选择器，增加优先级，但累加结果不进位，示例：

```
<style>
div { /* 优先级 1 */
color: red;
}
div.c { /* 优先级 1 + 10 = 11 */
color: gray;
}
.c.c.c.c.c.c.c.c.c.c.c.c { /* 优先级 10 * 11 = 110, 不进位, 90 */
color: blue;
}
#id { /* 优先级 100 */
color: yellow
}
</style>
<div>红字</div>
```



```
<div class="c">蓝字</div>
<div id="id" class="c">黄字</div>
```

如何限制 CSS 选择器的作用域？

- 通过 CSS 命名限制
通过后代选择器，将一组 CSS 选择器放入命名唯一的父级选择器中
唯一命名，可以自己指定，依靠工程化工具，通过 hash 或页面路径等规则，自动生成
CSS 变量的作用域由选择器，例如最常用的根选择器 :root

示例：

```
<style>
.unique-name p {
/* 这里设置的样式，不会影响其他p标签 */
}
</style>
<div class="unique-name">
<p></p>
</div>
<div>
<p></p>
</div>
```

- 通过 Shadow DOM 限制
通过 JS 给已有元素创建影子 DOM，将样式通过 <style> 标签写入影子 DOM

示例：

```
<div></div>
<p></p>

<script>
const shadowDom = document.querySelector('div').attachShadow({mode: 'open'})
shadowDom.innerHTML = '<p></p>'
shadowDom.innerHTML = `<style>p {
/* 这里设置的样式，并不会影响原来的p标签 */
}</style>
</script>
```

- 通过 @document 限制
通过 CSS at-rule 新增的 @document 限制 CSS 仅在被满足条件的 URL 生效
可以限定 URL 的地址，前缀，域名和子域名或者根据正则匹配
- 通过 CSS Modules 限制
将 CSS 作为资源引入，CSS Modules 会根据内容生成哈希字符串，字符串可以作为独一无二的类名

单位

CSS 中有哪些单位？

绝对长度单位

单位	名称	场景
px	像素	屏幕
pt	点	打印、UI稿

换算：1pt = 96 / 72 px

常见：9pt = 12px

此外，浏览器还支持打印常用单位

cm、mm、in、pc 和 Q

相对长度单位

单位	名称	场景
em	font-size：相对父元素 width 等：相对于自身的 font-size	自适应布局
rem	相对于根元素的字体大小	移动端
vw	视窗宽度1%	
vh	视窗高度1%	高度自适应

此外，浏览器还支持ex、ch、lh、vmin 和 vmax

百分比 % 相对于谁？

百分比总是相对于父元素，无论是设置 font-size 或 width 等。如果父元素的相应属性，经浏览器计算后，仍无绝对值，那么 % 的实际效果等同于 默认值，如 height: 100%

如何使用 rem 自适应布局？

```
/* 假设UI稿的设计宽度是720px，默认字号是16px */
<style>
div { font-size: 1rem; }
</style>
<div>您好</div>
<script>
function rem () {
document.documentElement.style.fontSize = document.body.clientWidth / 720 * 16 +
'px'
}
window.onresize = rem
rem()
</script>
```

颜色值都有哪几种表示方法？

关键字

- 颜色关键字：white等

- 透明: transparent
- 当前颜色: currentcolor

RGB color model

- 十六进制 RGB: #ffffff 简写: #fff
- 十六进制 RGBA: #ffffffff 简写: #ffff
- 函数 rgb(): rgb(255, 255, 255)
- 函数 rgba(): rgba(255, 255, 255, 1)

HSL color model

- 函数 hsl(): hsl(0, 0%, 100%)
- 函数 hsla(): hsla(0, 0%, 100%, 1)

盒模型

什么是盒模型?

盒模型由内向外: 内容 content + 内边距 padding + 边框 border + 外边距 margin

分为两类:

- 标准盒模型: border-sizing:content-box
width 和 height 设置内容 content 的宽和高
- 替代盒模型: border-sizing:border-box
width 和 height 设置内容 content + 内边距 padding + 边框 border 的宽和高

对比块、内联和内联块盒子

块盒子: display:block

- 换行
- width和height生效
- 竖直方向padding和margin生效

内联盒子: display:inline

- 不换行
- width和height无效
- 竖直方向padding和margin无效

内联块盒子: display:inline-block

- 不换行
- width和height生效
- 竖直方向padding和margin生效

什么是弹性盒模型?

弹性盒模型基于盒模型, 其宽度、高度、外边距都可以弹性变化, 以适应弹性布局

可以给父元素设置 display:flex 或 display:inline-flex, 让子元素成为弹性盒子

布局

浏览器默认是如何布局的?

流布局是浏览器布局的基本方式。包括块布局和内联布局。

浏览器根据书写顺序writing-mode，决定块布局和内联布局方向

horizontal-tb是writing-mode默认值，也是中英文的常用书写顺序

块布局 的方向从上到下，即块盒子从上到下换行，相邻块盒子的外边距折叠

内联布局 的方向由文本、列表水平对齐方向direction决定

- 左对齐ltr时，内联盒子在同一行，从左到右排列
- 右对齐rtl时，内联盒子在同一行，从右到左排列

什么是弹性布局？

给父元素设置display:flex，父元素表现为块盒子，开启弹性布局。

给父元素设置display:inline-flex，父元素表现为内联块盒子，开启弹性布局。

区别于默认布局，弹性布局中：

- 子元素成为弹性盒子，宽度、高度、外边距可以弹性变化，自适应父元素的尺寸
- 子元素可以在垂直、水平方向上，正向或反向排列
- 父元素通过justify-content决定子元素在主轴的对齐方式
- 父元素通过align-items决定子元素在交叉轴的对齐方式
- 父元素通过align-content决定多行子元素整体在交叉轴的对齐方式
- 子元素通过align-self决定自身在交叉轴的对齐方式

什么是外边距折叠，如何避免？

相邻块盒子的上下边距没有累加，而是重叠取其中最大值的现象，称为外边距折叠

常见的外边距折叠：

- 上下相邻块盒子的间距 = 上盒子下边距和下盒子上边距的最大值

```
<style>
div:first-of-type {
margin-bottom: 10px;
/* display: inline-block; 解决方法：将任意盒子转为内联块盒子 */
}
div:last-of-type {
margin-top: 10px;
}
</style>
<!-- 两个DIV的实际间距 10px -->
<div>1</div>
<div>2</div>
```

避免：将其中一个盒子转为内联块盒子

- 空块盒子

其上边距 <= 上盒子下边距，其下边距 <= 下盒子的上边距，相当于不存在

上下块盒子的间距 = 上盒子下边距 和 下盒子上边距的最大值

```
<style>
div:first-of-type {
```

```

margin-bottom: 10px
}
div:nth-of-type(2) {
margin-top: 10px;
margin-bottom: 10px;
}
/* 解决方法之1: 通过伪元素增加内容
div:nth-of-type(2)::before {
content: '内容';
display: block;
} */
div:last-of-type {
margin-top: 10px
}
</style>
<!-- 中间的空DIV, 只要上外边距<=10px, 下外边距<=10px, 都不影响1和3的实际间距 -->
<div>1</div>
<div><!-- 内容 解决方法之2: 直接增加内容 --></div>
<div>3</div>

```

避免: 增加内容, 直接写入或使用伪元素

- 没有触发BFC, 边框, 内边距和内容的父块盒子与子块盒子

父块盒子的上边距 = 父块盒子的上边距 和 子块盒子的上边距的最大值

父块盒子的下边距 = 父块盒子的下边距 和 子块盒子的下边距的最大值

```

<style>
<style>
.parent {
margin-top: 10px;
margin-bottom: 10px;
/* overflow: hidden; 解决方法之1: 触发BFC*/
/* border: 1px solid black; 解决方法之2: 设置边框*/
/* padding-top: 1px; 解决方法之3: 设置内边距*/
}
.sub {
margin-top: 10px;
margin-bottom: 10px;
}
</style>
<!-- 父块盒子和子块盒子的上外边距都是10px, 下外边距都是10px -->
<div class="parent">
<!-- <div>内容</div> 解决方法之4: 增加内容 -->
<div class="sub"></div>
</div>

```

避免外边距重叠:

- 设置边框
- 设置内边距
- 增加内容, 直接写入或使用伪元素
- 触发 BFC 块级格式上下文的方法都可以

什么是块级格式上下文？

定义

块级格式上下文，英文全称是 Block Formatting Context，简称 BFC

它声明了一块布局区域，浏览器对区域内盒子按照一定方式布局，包括默认布局、弹性布局、网格布局、表格布局等

- 默认布局时，区域高度包含浮动元素高度
- 不同区域间相互独立，区域内的盒子和区域外的盒子互不影响
- 不同区域不会发生外边距折叠

创建

我们可以根据布局、溢出处理和有限布局，用不同方法创建块级格式上下文

- 根元素<html>
- 无副作用：display:flow-root
- 默认布局
 - 绝对定位：position:absolute和position:fixed
 - 浮动：float:left``float:right
 - 行内块元素：display:inline-block
- 溢出处理
 - overflow:hidden隐藏滚动条，裁剪溢出内容
 - overflow:scroll显示滚动条，裁剪溢出内容
 - overflow:auto未溢出，隐藏滚动条。溢出，显示滚动条
- 有限布局
 - contain属性值不为none
- 弹性布局
 - display:flex直接子元素
 - display:inline-flex直接子元素
- 网格布局
 - display:grid直接子元素
 - display:inline-grid直接子元素
- 多列布局（分栏布局）
 - column-count分栏数属性值不为auto
 - column-width分栏列宽属性值不为auto
 - column-span:all跨越所有列，表现为不分栏
- 表格布局
 - display:table表格
 - display:inline-table内联表格
 - display:table-cell单元格
 - display:table-caption表格标题
 - display:table-row行
 - display:table-row-grouptbody
 - display:table-header-groupthead
 - display:table-footer-grouptfoot

用途

通过创建块级格式上下文，我们可以：

- 清除浮动

- 解决外边距折叠
- 限定布局范围，提高渲染性能

有哪些定位方式？

- 静态定位：position:static
- 相对定位：position:relative
 - 创建BFC和定位上下文
 - 当z-index不为auto时，创建层叠上下文
- 绝对定位：position:absolute
 - 创建BFC和定位上下文
 - 当z-index不为auto时，创建层叠上下文
- 固定定位：position:fixed，相对transform``perspective和filter不为none的最近父元素，没有是视窗
 - 创建BFC和定位上下文
 - 创建层叠上下文
- 粘性定位：position:sticky
 - 创建定位上下文
 - 创建层叠上下文
 - 滚动最近overflow不为visible父元素
 - 未被卷曲：表现为未创建BFC的相对定位
 - 将被卷曲：表现为绝对定位

什么是定位上下文？

position不为static时，可以通过top right bottom``left设置元素位置偏移量，并且不会影响其它元素的位置

定位上下文，决定元素相对于哪个父元素偏移

非静态定位元素，设置偏移量后

- 相对于最近的非静态定位的父元素偏移
 - 没有，则相对于根元素的父级，即视窗偏移
- 可以给父元素设置position不为static改变定位上下文，决定子元素相对于谁偏移

其中position:relative对父元素的副作用最小

子绝父相常用于组件内部的绝对定位，而不影响组件外元素的位置关系

什么是层叠上下文？

定义

层叠上下文是元素在Z轴上的层次关系集合并影响渲染顺序，设置z-index可改变position不为static的元素的层叠顺序

层叠上下文中父元素层级决定了子元素层级，兄弟元素间的层级由z-index影响

创建

- 根元素html
- 无副作用：isolation不为auto
- 定位：position不static

- fixed和sticky一定创建
 - relative和absolute当z-index不为auto时创建
- 显示类型：弹性布局和网格布局的子元素，z-index不为auto时创建
- 透明元素：opacity < 1
- 变形：transform不为none
- 滤镜：filter不为none
- 性能优化：
 - contain包含layout或paint
 - will-change不为auto

用途

理解层叠上下文，设置z-index，常用来：

- 改善兼容性
 - 解决遮挡问题
 - 解决滚动穿透问题
- 提升移动端体验
 - 如通过-webkit-overflow-scrolling: touch增加滚动回弹效果
- 性能优化
 - 将频繁变化的内容单独一层放置

什么是浮动，如何清除浮动？

定义

在默认布局流中，浮动元素脱离文档流，沿内联布局的开始或结束位置放置

与绝对和固定定位不同，浮动元素的宽高、内外边距和边框，依然影响相邻元素的位置，相邻元素环绕浮动元素流式布局

创建

- float不为none即可创建浮动元素
- 弹性布局的父元素不能浮动

清除浮动

浮动元素脱离文档流，只包含浮动元素的父元素高度为0，带来问题

- 父元素高度不会随内容高度变化，内外边距、边框和背景无法自适应内容
- 父元素后的元素，与父元素内的浮动元素重叠
- 父元素后的元素，外边距属性无效

解决问题的思路：

- 设置高度
 - 通过JS将父元素高度设为获取浮动元素的最大高度
 - 通过CSS将父元素高度height设置固定值，然后设置溢出属性overflow，裁剪超出高度的部分或添加滚动条
- 清除浮动
 - 通过HTML在父元素内部末尾添加清除浮动的元素

```
<style>
```



```

.box > div {
float: left;
width: 33.33%;
}
.clearfix {
clear: both;
}
.margin-top {
margin-top: 10px;
}
</style>
<div class="box">
<div></div>
<div></div>
<div></div>
<div class="clearfix"></div>
</div>
<div class="margin-top"></div>

```

- 通过 `css` 在父元素内部末尾添加清除浮动的伪元素

```

<style>
.box::after {
/** 清除浮动 **/
content: ' ';
display: block;
clear: both;
/** 隐藏 **/
visibility: hidden;
width: 0;
height: 0;
/** 隐藏: 兼容IE低版本 **/
line-height: 0;
}
.margin-top {
margin-top: 10px;
}
</style>
<div class="box">
<div></div>
<div></div>
<div></div>
</div>
<div class="margin-top"></div>

```

- 通过 `HTML` 在父元素下边添加清除浮动的元素（解决外边距问题）

```

<style>
.box > div {
float: left;
width: 33.33%;
}
.clearfix {
clear: both;

```

```
}  
.margin-top {  
margin-top: 10px;  
}  
</style>  
<div class="box">  
<div></div>  
<div></div>  
<div></div>  
</div>  
<div class="clearfix"></div>  
<div class="margin-top"></div><!-- 清除浮动后，外边距生效 -->
```

什么是滚动穿透，如何解决？

滚动穿透是在移动端，尤其是 iOS 中，弹出模态框，用户模态框中的滚动/滑动操作，穿过弹窗，导致页面滚动的现象

滚动穿透不是 BUG，只是运行环境对溢出、滑动事件处理略有差异

示例：

```
<style>  
body {margin:0;}  
.fixed {  
overflow: auto;  
margin: auto;  
position: fixed;  
width: 50vw;  
height: 50vh;  
top: 0;  
right: 0;  
bottom: 0;  
left: 0;  
background-color: #efefef;  
}  
.content, .list {  
height: 100vh;  
background-image: linear-gradient(to bottom, #efefef, #000);  
}  
.list {  
height: 200vh;  
}  
</style>  
<body>  
<div class="list"></div>  
<div class="fixed">  
<div class="content"></div>  
</div>  
</body>
```

请在真实的移动浏览器中，滑动中间的弹窗，你可能会发现页面也跟随滚动
具体表现因浏览器而异

在 CSS 中，有两个属性：

- pointer-events: none 禁止元素成为鼠标事件的目标对象
- touch-action: none 禁止元素响应任何触摸事件

它们都不能阻止滚动事件，冒泡到父级，让父级继续滚动

解决滚动穿透问题，比较的流行的方法是：

- 当模态框弹出时，通过position:fixed创建层叠上下文，让不该滚动的父级的脱离文档流，设置width:100%保留宽度，并将body的卷曲高度保存起来。
- 当模态框关闭时，通过position:static，让父级回归文档流，恢复之前的卷曲高度，即滚动位置

让我们解决示例的滚动穿透问题：

```
<style>
body {margin: 0;}
.modal {
display: none;
overflow: auto;
margin: auto;
position: fixed;
width: 50vw;
height: 50vh;
top: 0;
right: 0;
bottom: 0;
left: 0;
background-color: #efefef;
}
.content, .list {
height: 100vh;
background-image: linear-gradient(to bottom, #efefef, #000);
}
.list {
height: 200vh;
}
.open, .close {
position: fixed;
text-align: center;
}
</style>
<body>
<div class="open">打开</div>
<div class="list"></div>
<div class="modal">
<div class="close">关闭</div>
<div class="content"></div>
</div>
</body>
<script>
var openBt = document.querySelector('.open'),
closeBt = document.querySelector('.close'),
modal = document.querySelector('.modal'),
list = document.querySelector('.list'),
scrollTop = 0
openBt.onclick = function() {
scrollTop = document.body.scrollTop || document.documentElement.scrollTop
modal.style.display = 'block'
list.style.cssText = 'position: fixed; width: 100%; top: -' + scrollTop + 'px'
```

```
}
closeBt.onclick = function() {
  modal.style.display = 'none'
  list.style.cssText = 'position: static;'
  window.scrollTo({ top: scrollTop })
}
</script>
```

多方法实现水平居中

- text-align: center 适合内联或内联块元素的父元素设置
- position: static
 - margin: 0 auto 适合宽度已知元素
 - 块元素 display: block 宽度固定 width: 100px
 - 块元素 display: block 宽度自适应内容 width: fit-content
 - 表格元素 display: table
- position: relative / position: absolute / position: fixed / position: sticky 脱离文档流后
 - 偏移 left: 50% 或 right: 50%
 - margin-left 或 margin-right 设置宽度一半
 - transform: translateX(-50%)
 - 偏移 left: 0 和 right: 0
 - 宽度固定 width: 100px 或 自适应 width: fit-content
 - margin: 0 auto
- display: flex 父元素
 - justify-content: center 每行内部元素沿主轴居中排列

多方法实现垂直居中

- 内联父元素
 - line-height: 100px 行高固定值
- 内联块或块元素
 - line-height 等于 元素 height
 - padding-top 等于 padding-bottom
- 内联或内联块元素
 - vertical-align: middle 元素与行的基线+字母x的高度的一半对齐
- 表格单元格元素: display: table-cell 或
 - vertical-align: middle 内容或子元素垂直居中
- position: static
 - margin-top: 50% 适合宽度已知元素 transform: translateY(-50%)
 - 块元素 display: block 高度固定 height: 100px
 - 块元素 display: block 宽度自适应内容 height: fit-content
 - 表格元素 display: table
- position: relative / position: absolute / position: fixed / position: sticky 脱离文档流后
 - 偏移 top: 50% 或 bottom: 50%
 - margin-top 或 margin-bottom 设置高度一半
 - transform: translateY(-50%)
 - 偏移 top: 0 和 bottom: 0

- 宽度固定height:100px或自适应height:fit-content
 - margin: auto 0
- display:flex父元素
 - align-content: center多行内部元素整体沿交叉轴居中排列
 - align-items: center每行内部元素整体沿交叉轴居中排列
 - align-self: center单个内部元素沿交叉轴居中排列

多方法实现高度 100% 撑满视窗

如果视窗高度是变化的，纯CSS撑满视窗，可以用相对单位

- 百分比
 - 百分比相对于父元素设置，height默认为auto，即内容高度
 - 从根元素html向内到body，高度都设置为height:100%

```
<style>
html, body {
height: 100%;
}
div {
height: 100%;
background-color: azure;
}
body { margin: 0; }
</style>
<div></div>
```

- vh，直接设置相对视窗高度

```
<style>
div {
height: 100vh;
background-color: azure;
}
body { margin: 0; }
</style>
<div></div>
```

如果对于内容高度，多用视窗高度减去固定元素高度（如导航栏，状态栏），可用函数calc()

- calc(100% - 50px)
- calc(100vh - 50px)

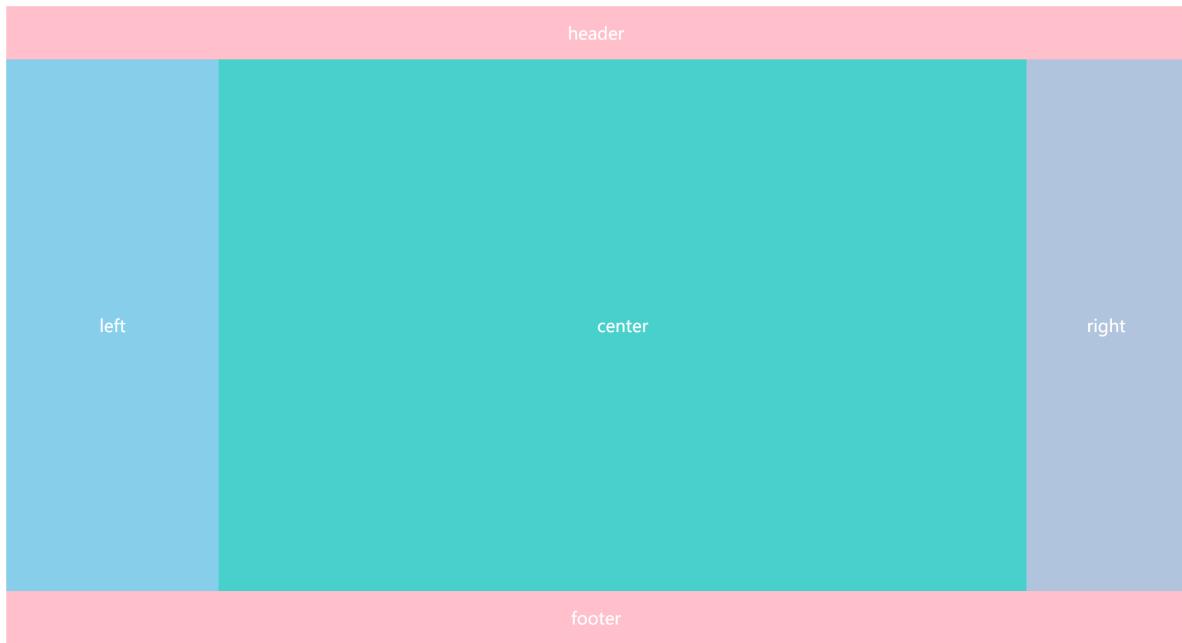
圣杯布局

在弹性布局以前，圣杯布局是通过浮动和定位实现三栏布局的一种方案之一

圣杯布局的特点：

- 上下为通栏，下通栏清浮动
- 中间为三栏布局，子元素按中左右的顺序浮动float:left
 - 宽度

- 左边栏宽度固定 = 父元素的左内边距padding-left
- 右边栏宽度固定 = 父元素的右内边距padding-right
- 中间内容宽度 = 100%
- 位置
 - 左右边栏相对定位position:relative
 - 左边栏左外边距margin-left: -100%, right:宽度
 - 右边栏左外边距margin-left:-宽度, right:-宽度
- 注意
 - 需设置最小宽度min-width, 避免视窗宽度过小, 浮动错位



```
<style>
body { margin: 0; }
div {
  text-align: center;
  color: #fff;
}
.header, .footer {
  height: 50px;
  background-color: pink;
  line-height: 50px;
}
.content {
  padding-left: 200px;
  padding-right: 150px;
  min-width: 500px;
  line-height: 500px;
}
.content > div {
  float: left;
  height: 500px;
}
.center {
  width: 100%;
  background-color: mediumturquoise;
}
.left, .right {
  position: relative;
```

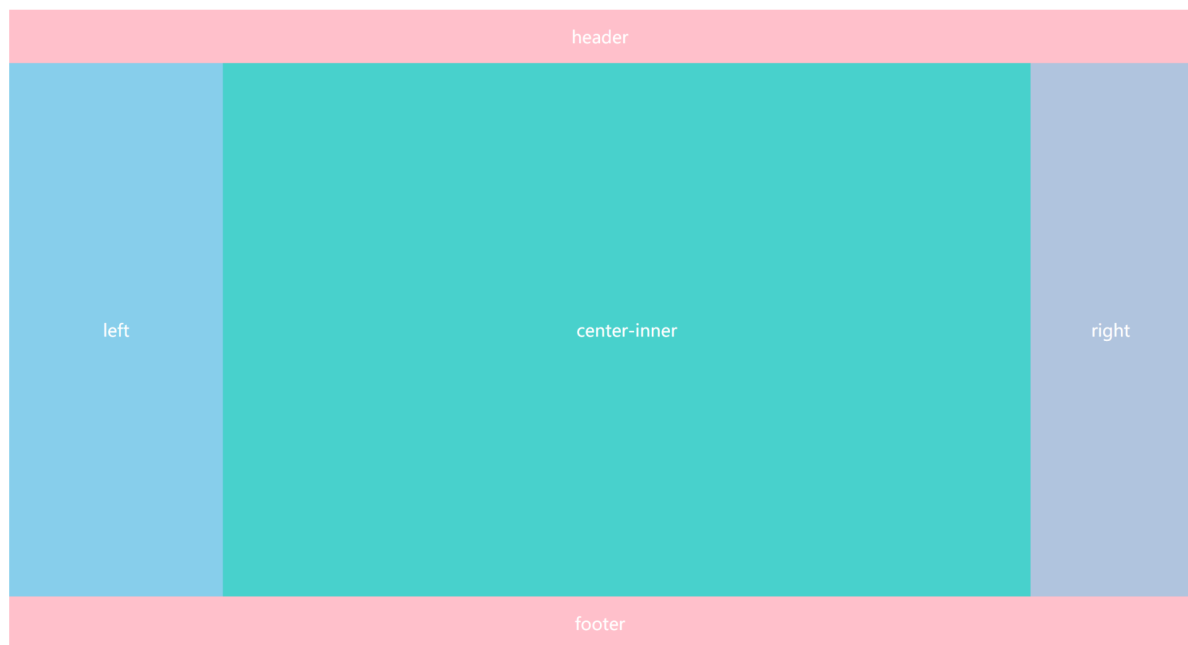
```
}
.left {
  width: 200px;
  right: 200px;
  margin-left: -100%;
  background-color: skyblue;
}
.right {
  width: 150px;
  right: -150px;
  margin-left: -150px;
  background-color: lightsteelblue;
}
.footer {
clear: both;
}
</style>
<div class="header">header</div>
<div class="content">
  <div class="center">center</div>
  <div class="left">left</div>
  <div class="right">right</div>
</div>
<div class="footer">footer</div>
```

双飞翼布局

双飞翼布局由淘宝UED发扬，是通过浮动和定位实现三栏布局的一种方案之一

双飞翼布局的特点：

- 上下为通栏，下通栏清浮动
- 中间为三栏布局，子元素按中左右的顺序浮动float:left
 - 宽度
 - 左边栏宽度固定 = 中间内容子元素左外边距margin-left
 - 右边栏宽度固定 = 中间内容子元素右外边距margin-right
 - 中间内容宽度 = 100%
 - 位置
 - 左边栏左外边距margin-left: -100%
 - 右边栏左外边距margin-left:-宽度



```
<style>
body { margin: 0; }
div {
  text-align: center;
  color: #fff;
}
.header, .footer {
  height: 50px;
  background-color: pink;
  line-height: 50px;
}
.content > div {
  float: left;
  height: 500px;
  line-height: 500px;
}
.center {
  width: 100%;
  background-color: mediumturquoise;
}
.inner {
  height: 500px;
  margin-left: 200px;
  margin-right: 150px;
}
.left {
  margin-left: -100%;
  width: 200px;
  background-color: skyblue;
}
.right {
  margin-left: -150px;
  width: 150px;
  background-color: lightsteelblue;
}
.footer {
  clear: both;
}
```



```

</style>
<div class="header">header</div>
<div class="content">
<div class="center">
<div class="inner">center-inner</div>
<div>
  <div class="left">left</div>
  <div class="right">right</div>
</div>
<div class="footer">footer</div>

```

多方法实现三栏布局

按照出现顺序，三栏布局包括：

分栏布局、表格布局、流式布局、弹性布局、网格布局

- 分栏布局
 - columns: 3分成3列, word-break: break-all;强制换行
 - 内容满1列后, 自动移到下一行

contentcontentcontentcontentcontentcont	ntcontentcontentcontentcontentcontentco	tentcontentcontentcontentcontentcontent
entcontentcontentcontentcontentcontentc	ntentcontentcontentcontentcontentcontent	contentcontentcontentcontentcontentcont
ontentcontentcontentcontentcontentconte	tcontentcontentcontentcontentcontentcon	entcontentcontentcontent

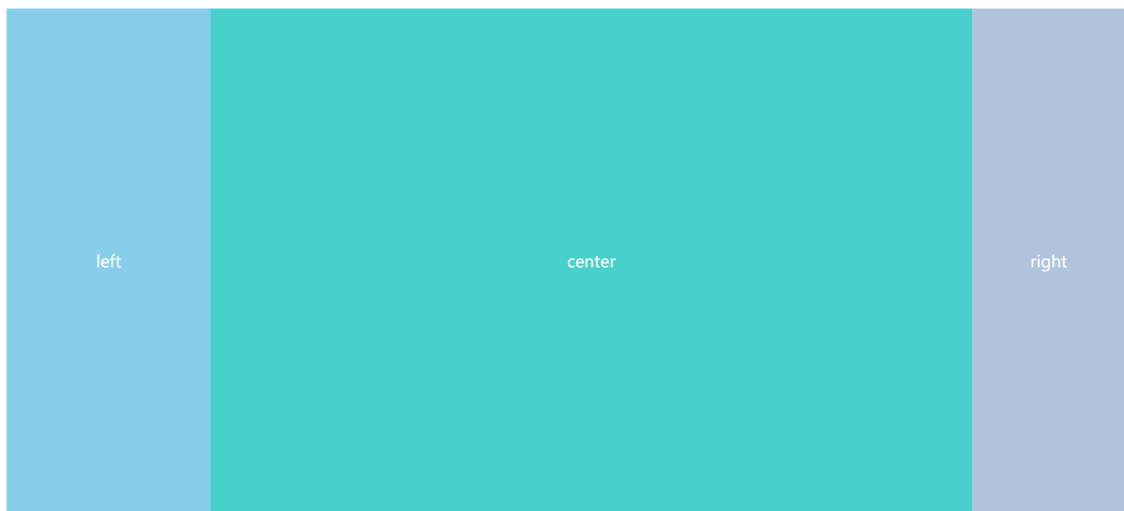
```

<style>
.content {
  columns: 3;
  word-break: break-all;
  background-color: mediumturquoise;
  color: white;
}
</style>

<div
class="content">contentcontentcontentcontentcontentcontentcontentcontentcontentcontentc
ontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcont
entcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontent
contentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcontentcon
tentcontentcontentcontentcontent</div>

```

- 表格布局
 - 宽度
 - 表格宽度 100%
 - 左单元格宽度固定
 - 右单元格宽度固定
 - 垂直居中
 - vertical-align:middle



```
<style>
.content {
  width: 100%;
  height: 500px;
  border-collapse: collapse;
  color: white;
  vertical-align: middle;
  text-align: center;
}
.left {
  width: 200px;
  background-color: skyblue;
}
.center {
  background-color: mediumturquoise;
}
.right {
  width: 150px;
  background-color: lightsteelblue;
}
</style>
<table class="content">
  <tbody>
    <tr>
      <td class="left">left</td>
      <td class="center">center</td>
      <td class="right">right</td>
    </tr>
  </tbody>
</table>
```

- 流式布局
 - 圣杯布局 (同上题)
 - 双飞翼布局 (同上题)
- 弹性布局
 - 左/右边栏宽度固定: flex-basis初始宽度 或width宽度, 前者优先级大于后者
 - 中间内容宽度: flex-grow: 1剩余空间分配比例 或 简写flex: 1

```
<style>
```

```

.content {
  display: flex;
  height: 500px;
  line-height: 500px;
  text-align: center;
  color: #fff;
}
.left {
  flex-basis: 200px;
  background-color: skyblue;
}
.center {
  flex-grow: 1;
  background-color: mediumturquoise;
}
.right {
  flex-basis: 150px;
  background-color: lightsteelblue;
}
</style>
<div>
  <div class="content">
    <div class="left">left</div>
    <div class="center">center</div>
    <div class="right">right</div>
  </div>

```

- 网格布局

- 设置网格列的宽度grid-template-columns: 200px auto 150px或 简写grid: auto-flow / 200px auto 150px

```

<style>
.content {
  display: grid;
  grid-template-columns: 200px auto 150px;
  height: 500px;
  line-height: 500px;
  text-align: center;
  color: #fff;
}
.left {
  background-color: skyblue;
}
.center {
  background-color: mediumturquoise;
}
.right {
  background-color: lightsteelblue;
}
</style>
<div>
  <div class="content">
    <div class="left">left</div>
    <div class="center">center</div>
    <div class="right">right</div>
  </div>

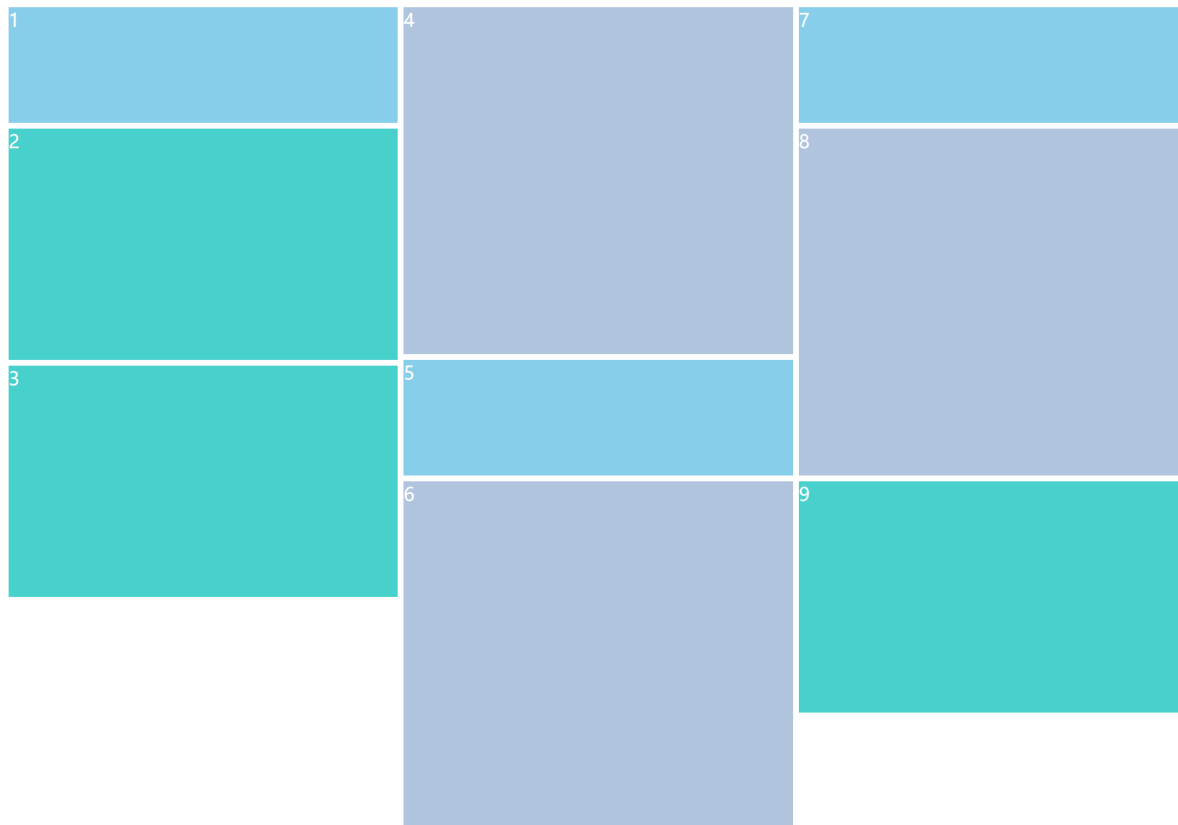
```

```
</div>
```

瀑布流布局

瀑布流布局的特点是等宽分栏，每一栏的高度可能不同

纯 CSS 瀑布流布局只考虑外观。如果考虑数据的加载顺序需要JS辅助



- 多列布局（分栏布局）
 - 通过 `column-count` 设置栏数，`column-gap` 设置间距
 - 设置 `break-inside: avoid` 避免元素在分栏时中断

```
<style>
.content {
  column-count: 3;
  column-gap: 5px;
}
.content > div {
  margin-bottom: 5px;
  break-inside: avoid;
  color: white;
}
.d1, .d5, .d7 {
  height: 100px;
  background-color: skyblue;
}
.d2, .d3, .d9 {
  height: 200px;
  background-color: mediumturquoise;
}
```

```

    }
    .d4, .d6, .d8 {
        height: 300px;
        background-color: lightsteelblue;
    }
</style>
<div class="content">
<div class="d1">1</div>
<div class="d2">2</div>
<div class="d3">3</div>
<div class="d4">4</div>
<div class="d5">5</div>
<div class="d6">6</div>
<div class="d7">7</div>
<div class="d8">8</div>
<div class="d9">9</div>
</div>

```

- 弹性布局

- 水平：栏目间弹性布局，通过 `margin` 设置间距
- 垂直：栏目内弹性布局，通过 `flex-direction: column` 沿垂直轴方向布局

```

<style>
.content {
    display: flex;
}
.column {
    flex: 1;
    display: flex;
    flex-direction: column;
    margin-right: 5px;
}
.column:last-child {
    margin-right: 0;
}
.content div {
    margin-bottom: 5px;
    color: white;
}
.d1, .d5, .d7 {
    height: 100px;
    background-color: skyblue;
}
.d2, .d3, .d9 {
    height: 200px;
    background-color: mediumturquoise;
}
.d4, .d6, .d8 {
    height: 300px;
    background-color: lightsteelblue;
}
</style>
<div class="content">
    <div class="column">
        <div class="d1">1</div>

```

```
<div class="d2">2</div>
<div class="d3">3</div>
</div>
<div class="column">
  <div class="d4">4</div>
  <div class="d5">5</div>
  <div class="d6">6</div>
</div>
<div class="column">
  <div class="d7">7</div>
  <div class="d8">8</div>
  <div class="d9">9</div>
</div>
</div>
```

品布局

品布局，顾名思义，与「品」字相像，第一行横跨两列，第二行分两栏



几乎所有 CSS 布局方式都可以用来实现品布局

- 浮动布局：注意清除浮动
- 内联块布局：注意设置父元素font-size:0px，避免display:inline-block的间隙
- 表格布局

display:table不支持跨行跨列，这里用

标签实现

border-collapse: collapse合并表格边框

- 弹性布局：flex-wrap: wrap开启多行显示
- 网格布局：grid-column-start`grid-column-end设定网格起止边缘

示例：

```
<style>
body > div {
  margin: 5px 0;
}
div div, .table td {
  height: 50px;
}
div:first-child, .table thead td {
  line-height: 50px;
  text-align: center;
  font-size: 16px;
  background-color: pink;
  color: white;
}
div:nth-child(2), .table tbody td:first-child {
  background-color: skyblue;
```

```

}
div:last-child, .table tbody td:last-child {
    background-color: lightcyan;
}
/** 浮动布局 */
.float {
    display: flow-root;
}
.float div:first-child ~ div {
    float: left;
    width: 50%;
}
/** 内联块布局 */
.inline-block {
    font-size: 0px
}
.inline-block div:first-child ~ div {
    display: inline-block;
    width: 50%;
}
/** 表格布局 */
.table {
    width: 100%;
    border-collapse: collapse;
}

/** 弹性布局 */
.flex {
    display: flex;
    flex-wrap: wrap;
}
.flex div:first-child {
    width: 100%;
}
.flex div:first-child ~ div {
    width: 50%;
}

/** 网格布局 */
.grid {
    display: grid;
}
.grid div:first-child {
    grid-column-start: 1;
    grid-column-end: 3;
}
</style>
<div class="float">
    <div>float</div>
    <div></div>
    <div></div>
</div>
<div class="inline-block">
    <div>inline-block</div>
    <div></div>
    <div></div>
</div>
<table class="table">

```

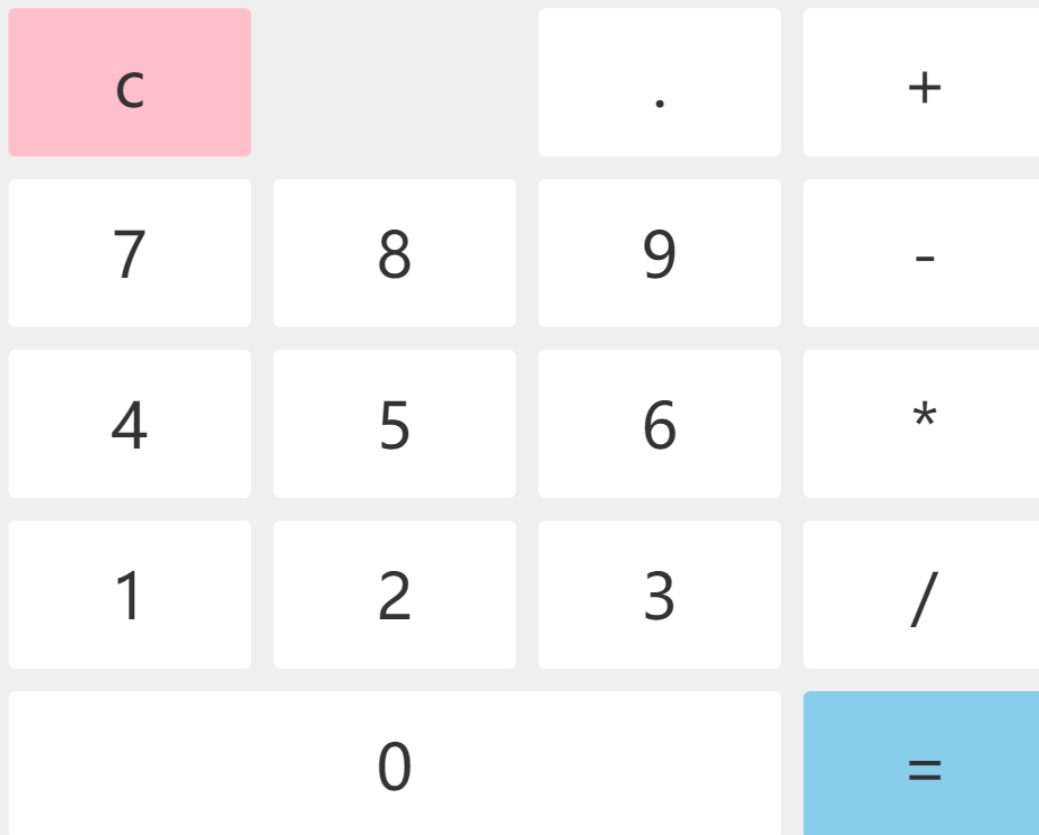
```
<thead>
  <tr>
    <td colspan="2">table</td>
  </tr>
</thead>
<tbody>
  <tr>
    <td></td>
    <td></td>
  </tr>
</tbody>
</table>
<div class="flex">
  <div>flex</div>
  <div></div>
  <div></div>
</div>
<div class="grid">
  <div>grid</div>
  <div></div>
  <div></div>
</div>
```

实现简易计算器

计算器最适合网格布局，可分为以下几步：

- 创建父元素，设置网格布局display:grid
- 设置网格行高度grid-template-rows和列高度grid-template-columns
 - 2.1 计算器按键宽和高大多相同，可以使用repeat简写重复的值
- 设置行间距row-gap列间距column-gap，如果行列间距相等，则设置grid-gap
- 合并
 - 4.1 合并行：grid-row-start grid-row-end设定网格行起止边缘，简写grid-row
 - 4.2 合并列：grid-column-start grid-column-end设定网格列起止边缘，简写grid-column

27



```
<style>
.calculator {
  padding: 30px;
  width: 460px;
  background-color: #efefef;
  font-size: 28px;
  color: #333;
}
.btns {
  margin: 0;
  padding: 0;
  list-style: none;
  display: grid;
  grid-template-columns: repeat(4, 108px);
  grid-template-rows: repeat(5, 66px);
  grid-gap: 10px;
}
.btn {
  cursor: pointer;
  text-align: center;
  line-height: 66px;
  border-radius: 3px;
  background-color: white;
```

```

}
.btn:first-child {
  background-color: pink;
}
.btn:nth-of-type(2) {
  background-color: transparent;
}
.btn:nth-last-of-type(2) {
  grid-column: 1/4;
}
.btn:last-child {
  background-color: skyblue;
}
.btn:hover {
  opacity: .65;
}
.echo {
  margin-bottom: 30px;
  width: 460px;
  height: 66px;
  line-height: 66px;
  font-size: 30px;
  text-align: right;
  border-bottom: 1px solid white;
}
</style>
<div id="app">
<div class="calculator">
<div class="echo" v-text="echo"></div>
<ul class="btns">
<li class="btn" v-for="btn in btns" v-text="btn" @click="cal(btn)"></li>
</ul>
</div>
</div>
<script src="https://cdn.bootcdn.net/ajax/libs/vue/2.6.12/vue.min.js"></script>
<script>
var app = new Vue({
  el: '#app',
  data: {
    echo: '',
    btns: ['c', '', '.', '+', '7', '8', '9', '-', '4', '5', '6', '*', '1', '2', '3', '/', '0', '=']
  },
  methods: {
    cal(v) {
      switch(v){
        case '=':
          this.echo = eval(this.echo.toString().replace(/\.+/g, '.'));
          break;
        case 'c':
          this.echo = '';
          break;
        default:
          this.echo += v;
      }
    }
  }
})
</script>

```

视差滚动

夜晚，坐在车里，两侧树木飞快向后移动，抬头看月亮，月亮总是停在原地

车速固定，树木和月亮离车的距离不同，这样产生的视觉差异，给人以立体感觉

在 CSS 动画中，有两种方式实现视差滚动

- 固定背景：设置属性background-attachment: fixed，让背景图片相对视窗固定

示例：

```
<style>
body {
  margin: 0;
}
.box {
  height: 100vh;
  overflow: auto;
}
.box > div {
  height: 50vh;
}
.title {
  font-size: 6em;
  line-height: 50vh;
  text-align: center;
  background-color: black;
  color: white;
}
.bg1, .bg2, .bg3 {
  background-attachment: fixed;
}
.bg1 {
  background-image: linear-gradient(to bottom, black, pink);
}
.bg2 {
  background-image: linear-gradient(to bottom, black, skyblue);
}
.bg3 {
  background-image: linear-gradient(to bottom, black, lightcyan);
}
</style>

<div class="box">
  <div class="title">title1</div>
  <div class="bg1"></div>
  <div class="title">title2</div>
  <div class="bg2"></div>
  <div class="title">title3</div>
  <div class="bg3"></div>
  <div class="title">title4</div>
</div>
```

- 使用3D效果

- 滚动元素
 - 指定观众与初始平面的距离perspective: 1px
 - 自动溢出处理overflow:auto
- 父元素：撑开滚动元素
- 子元素：通过translateZ或translate3d指定子元素在 Z 轴位置

示例：

```
<style>
.wrapper {
  perspective: 1px;
  height: 30vh;
  overflow: auto;
}
.box {
  height: 40vh;
}
.cell1, .cell2, .cell3{
  display: inline-block;
  width: 50px;
  height: 50px;
  margin-top: 90px;
}
.cell1 {
  background-color: pink;
  transform: translateZ(0px);
}
.cell2 {
  background-color: skyblue;
  transform: translateZ(-1px);
}
.cell3 {
  background-color: lightcyan;
  transform: translateZ(-2px);
}
</style>

<div class="wrapper">
  <div class="box">
    <div class="cell1"></div>
    <div class="cell2"></div>
    <div class="cell3"></div>
  </div>
</div>
```

动画

对比过渡和动画

过渡transition：元素在两个状态间切换时的过渡效果

动画animation：通过@keyframe设定动画关键帧，可以同时设定多组动画

根据定义，transition可以看最作animation中，一组@keyframe包含from（0%）和to（100%）两个关键帧（状态）的特殊情况

区别如下：

transition

- 事件驱动：需要访客或JS使状态发生变化后触发
- 一次性：想要循环，需要反复修改状态
- 定义两个状态：开始状态和结束状态
- 可指定唯一过渡属性：transition-property:all默认两个状态间所有可过渡属性，都会有补间动画。可以通过transition-property指定可以唯一过渡属性

animation

- 自动执行或事件驱动
- 循环或指定执行次数
- 定义多个状态
- 不可指定唯一过渡属性
- 可控制：暂停，播放等

如何优化 CSS 动画的性能？

一方面，减少不必要动画

- 设备：尽可能适应低配置设备
- 耗电量：减少耗电量
- 引发不适：避免闪烁、变化强烈、3D 眩晕动画
- 用户原因（喜好、疾病、工作等）已配置减少动画

综上，通过 CSS 媒体查询，在以下场景减少动画，始终避免引起不适的动画

- 移动设备
- prefers-reduced-motion设置值为reduce
- 主体内容不应只有纯动画，适当的文字说明，利于读屏软件和 SEO

另一方面，提升动画体验

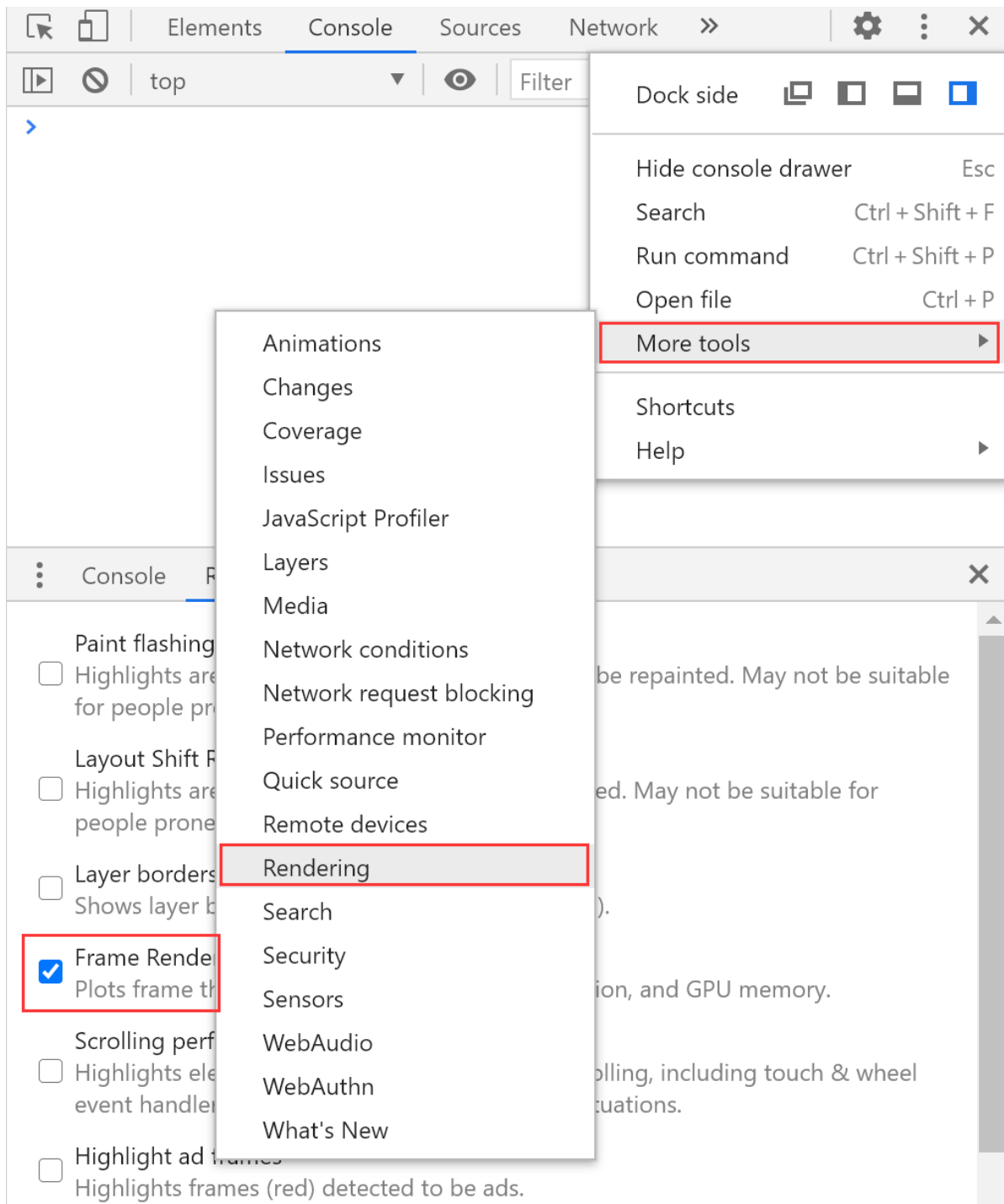
- 减少不必要动画元素和属性，避免使用transition:all
- 减少重排、重绘

不同属性值引起改变，重新渲染有三种执行路径，尽量只使用合成属性

属性类型	执行路径	属性举例
重排属性	layout > paint > composite	盒模型: displaypaddingmarginwidthheightmin-heightborderborder-width 定位及浮动: positiontopbottomleftrightfloatclear 文字及溢出: font-familyfont-sizefont-weightline-heighttext-alignvertical-alignwhite-spaceoverflowoverflow-y
重绘属性	paint > composite	colorborder-styleborder-radiusvisibilitytext-decorationbackgroundbackground-imagebackground-positionbackground-repeatbackground-sizeoutlineoutline-coloroutline-styleoutline-widthbox-shadow
合成属性	composite	transformopacitybackface-visibilityperspectiveperspective-origin

- 避免需要大量计算的属性: box-shadow `` filter
- 开启硬件加速: transform: translate3d(0, 0, 0) transform: translateZ(0) will-change;transform可开启, 内存占用可能增加
- 创建层叠上下文, 动画元素脱离文档流, 独立图层, 例如:
 - position为absolute或fixed
 - contain包含layout或paint

Chrome 浏览器提供了 FPS 帧率显示工具, 方便实时测量动画性能



对比 CSS 动画和 JS 动画?

JS 动画:

- IE5.5 - 9支持
- 报错时, 影响或被其他脚本影响。定时器不准确
- 复杂逻辑 (条件判断、递归) 和交互 (监听事件) 的动画
- JS在主线程的解析的环境中, 阻塞或被阻塞其他操作。早期运行效率低。V8和现代架构浏览器已经改善

CSS 动画:

- transition `` animation IE10+ 支持, 早期私有属性多, 不同环境效果略不同
- 不支持的环境, 直接跳过
- 补间动画支持, 设置关键帧, 更容易。正在完善的事件和更细粒度Animation()支持
- 可通过合成属性, 如transform在 compositor 线程实现动画

JS 动画依赖 CSS 的样式

- 通过 CSS 开启硬件加速
- 都应减少重排和重绘

CSS 动画的更细粒度控制

- 依赖 JS

综上：

- 原来用JS的定时器，绘制每一帧，监听悬浮、获取焦点事件实现的动画和交互，都可以通过纯CSS动画配合伪类实现
- 未来JS更专注于复杂逻辑动画或对CSS的动画更细粒度的交互

纯 CSS 实现打字机效果

CSS 动画默认带过渡效果，浏览器会自动完成两个状态（关键帧）的补间动画，平滑过渡

step(num, start / end)属性，可以从两个状态的补间动画中，平均取num个帧（过渡状态），让动画在这些帧间直接跳转

step-start是step(10, start)的简写，动画从开始状态起

step-end是step(10, end)的简写，动画从结束状态起

实现打字机效果，思路：

给一个DIV添加动画，宽width从0到文字总长度变化

有多少字，就从补间动画平均取多少个过渡状态，每状态对应显示一个字

光标闪烁，只有显示和隐藏两个状态，期间不需要过渡效果，并且无限循环

代码：

```
<style>
div {
  border-right: 1px solid #ccc; /* 光标 */
  width: 5em; /* 5个字的宽度 */
  white-space: nowrap; /* 不换行 */
  overflow: hidden; /* 溢出内容裁剪 */
  /** step(5, end), 直接跳到5个居中状态，省略过渡 step-end 直接跳到最终状态 **/
  animation: type 3s steps(5, end), cursor .5s step-end infinite alternate;
}
@keyframes type {
  0% {
    width: 0;
  }
}
@keyframes cursor {
  50% {
    border-color: transparent;
  }
}
</style>
<div>打字机效果</div>
```


纯 CSS 实现暗黑、夜间模式

- 匹配暗黑模式: `prefers-color-scheme: dark` 表示当前系统的主题色为暗色
 - CSS 媒体查询

```
@media (prefers-color-scheme: dark) {  
  /** 暗黑模式的CSS **/  
}
```

- Link 媒体查询

```
<link rel="stylesheet" href="dark.css" media="(prefers-color-scheme: dark)" />
```

- 实现暗黑模式
 - 变量: 自己写具体样式

```
@media (prefers-color-scheme: dark) {  
  :root {  
    --color: white; /** 为暗黑模式配置变量值 **/  
  }  
  }  
  html {  
    color: var(--color);  
  }  
}
```

- 混合模式: `* {mix-blend-mode: difference}` 影响图片和背景
- 反转 + 滤镜: `html, img {filter: invert(1) hue-rotate(180deg);}`

纯 CSS 实现骨架屏

骨架屏在数据加载前, 优先展示页面的大体结构, 通常用灰色的明暗变化的块, 提高用户体验

纯 CSS 骨架屏适合 Ajax 或 Fetch 请求数据, 填充模板引擎渲染的场景。如果请求的是页面本身, 那么预渲染生成骨架图, 是更好的方式

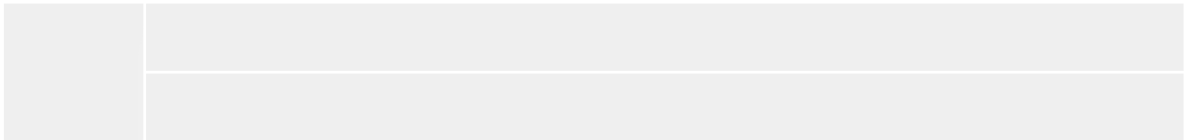
用子绝父相的定位方式, 在需要显示为骨架屏的元素内, 放置两个伪元素

- 一个撑满父容器, 表示未填充数据前的元素
- 一个填充倾斜的渐变条纹, 移动条纹位置, 模拟进度条动画

```
<style>  
@keyframes skeleton-anime {  
  from { transform: translateX(-100%); }  
  to { transform: translateX(100%); }  
}  
.skeleton {  
  position: relative;  
  overflow: hidden;  
}  
.skeleton::before, .skeleton::after {  
  position: absolute;  
  content: '';  
  display: block;  
  width: 100%;  
}
```

```
height: 100%;
background-color: #efefef;
}
.skeleton::after {
  background-image: linear-gradient(-75deg, #efefef, white 30%, #efefef);
  animation: skeleton-anime 1s infinite;
  opacity: .3;
}
.avater {
  float: left;
  margin-right: 2px;
  width: 100px;
  height: 100px;
}
.title, .des {
  height: 48px;
  margin-top: 2px;
}
</style>
<div class="avater skeleton"></div>
<div class="title skeleton"></div>
<div class="des skeleton"></div>
```

显示效果如图所示：



原理

对比 CSS 加载的方式？

四种加载方式：

- style属性（内联样式）
 - 定义的属性优先级高于所有选择器和其它加载方式
 - 无法复用
 - 覆盖样式只能使用!important
- <style></style>标签（嵌入样式）
 - 作用域：只对当前页面有效
 - 单页应用，减少请求，提高速度
 - 支持媒体查询
- <link>（外链样式）
 - CSS 和 HTML 分离，便于复用和维护
 - 利于缓存
 - 支持媒体查询
- @import（导入样式）
 - 模块化
 - 利于缓存
 - 支持媒体查询
 - 需先下载并解析引用 CSS，才可以继续下载导入 CSS。现代通常使用模块化工具在编译时合并 CSS

加载 CSS 是否会阻塞页面渲染？

- CSS 不阻塞 HTML 转 DOM 树

HTML 是超文本标记语言，<元素名>作起始标签，</元素名>结束标签或唯一标签

浏览器将元素和文本作为节点，将父元素作为父节点，将 HTML 转为 DOM 树。这个过程与 CSS 无关

- CSS 阻塞 CSS 树和 Render 树

CSS 会被转为 CSS 树，并与 DOM 树结合成 Render 树等待渲染

如果 CSS 下载很慢，那么 CSS 解析会被阻塞，后续的渲染也无法进行

通过媒体查询，可以设置 link 引用的 CSS 不阻塞渲染，但仍会下载

- CSS 阻塞 JS 执行
JS 可能需要查询、修改 DOM 树和 CSS 样式，本身阻塞 DOM 解析
如果 JS 不依赖样式，可以放在 CSS 前，避免被阻塞

所以，通常情况下：

- CSS 和不依赖 DOM 的 JS 放 </head> 标签前
- 依赖 DOM 的 JS 放在 </body> 标签前

浏览器是如何解析和渲染 CSS 的？

浏览器解析和渲染 CSS 的步骤：

- 解析

将 CSS 字符串转换为包括选择器、属性名、属性值的数据结构，长度单位被转换为像素，关键字被转换为具体值，需要计算的函数转为具体值

- 级联

相同元素相同属性的最终值基本由书写顺序，按先后决定，此外：

- !important > 其它
- style 属性 > 其它
- 选择器优先级

ID > 类 > 类型（标签） > 相邻 > 子代 > 后代 > 通配符 > 属性 > 伪类

- 开发者 > 用户配置 > 浏览器默认属性
- 层叠

根据 position 不为 static 等属性或弹性布局中的子元素等情况创建层叠上下文。根据 z-index 决定层的叠加顺序

- Render Tree

深度优先遍历之前解析 HTML 得到的 Dom Tree，匹配解析 CSS 得到的 CSSOM

计算元素的位置、宽高，将可见元素的内容和样式放入 Render Tree

- 布局

分层按照流式布局（块、内联、定位、浮动）、弹性布局、网格布局或表格布局等布置元素，按照尽可能多地展示内容的原则，处理溢出

- 绘制

分层绘制颜色、边框、背景、阴影

- 合成

将不同图层分格渲染出位图，可交由 GPU 线程处理

处理图层的透明度opacity，和变形transform等

将所有图层合到一起

- 重新渲染

JS 更改 CSS 属性，CSS 动画以及伪类（如hover），内容变更等，可能会引起浏览器重新布局、绘制或者合成

对比获取 CSS 样式的接口

- style
 - 可读写
 - 属性名驼峰写法
 - 通过内联样式style读写属性
- currentStyle
 - 可读
 - 兼容连字符-写法
 - IE5.5 - IE8
- getComputedStyle
 - 可读
 - 兼容连字符-写法
 - IE9+
 - 来自CSS Object Model，计算后的属性
 - 支持伪类
- document.styleSheets
 - 可读
 - 获取规则列表
 - IE9+
 - 可写支持insertRule``deleteRule

什么是重排和重绘，更改哪些属性会触发重排和重绘，如何避免？

什么是重排和重绘？

当DOM的样式或内容会被修改时，将触发重新渲染。除了属性值计算、单位换算外，渲染主要分为三个步骤：

- 布局：计算盒模型的位置，大小
- 绘制：填充盒模型的文字、颜色、图像、边框和阴影等可视效果
- 合并：所有图层绘制后，按层叠顺序合并为一个图层

重新渲染一般有三种执行路径：

- 重排：布局 → 绘制 → 合并

- 重绘：绘制 → 合并
 - 合并
- 不同属性的修改，会触发不同的渲染路径

更改哪些属性会触发重排和重绘？

引起重排的属性，即布局类属性，包括：

类型	属性名
盒模型	displaypaddingmarginwidthheightmin-heightmax-heightborderborder-width
定位和浮动	positiontopbottomleftrightfloatclear
文字及溢出	font-familyfont-sizefont-weightline-heighttext-alignvertical-alignwhite-spaceoverflowoverflow-y

引起重绘的属性，即绘制类属性，包括：

类型	属性名
颜色	color
边框	border-colorborder-styleborder-radius
背景	backgroundbackground-imagebackground-positionbackground-repeatbackground-size
轮廓	outlineoutline-coloroutline-styleoutline-width
可见性	visibility
文字方向	text-decoration
发光	box-shadow

如何避免重排和重绘？

- 尽量使用仅引起合成的属性

类型	属性名
变形	transform
透明度	opacity

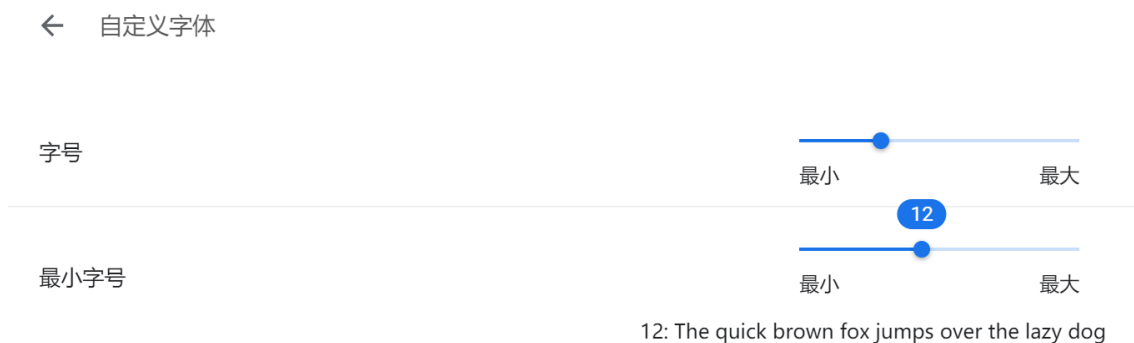
- 限制重新渲染区域
 - 使用position:absolute或position:fixed等方法创建层叠上下文
 - 使用contain:layout或contain:paint等属性值，让当前元素和内容独立于 DOM 树
- 减少使用display:table或
表格布局
- 利用浏览器自身优化
- 引起回流、重绘的属性操作会放入队列，达到一定数量或时间，再一次渲染

- 用变量缓存元素的属性值
- 要设置的属性值减少依赖其它属性值
- 避免频繁读取计算属性值
- 手动一次渲染
- 强制使用style.cssText或setAttribute('style', 样式)将所有设置的属性，一次写入内联样式
- 优化 DOM 树
 - 使用文档碎片或display:none隐藏节点，缓存要插入的节点，之后将缓存结果一次性插入 DOM 树并显示
 - 使用replaceChild cloneNode减少先删除、创建再插入 DOM 的场景

设计

何写小于 10 像素的字

Chrome 浏览器在 chrome://settings/fonts 中允许用户设置最小字号



最小字号默认值为12px，font-size小于最小字号，会被强制设为最小字号

可以通过-webkit-transform来缩小字号，并且只对webkit内核浏览器生效

```
<style>
  font-size : 10px;
  -webkit-transform : scale(0.8333);
</style>
```

实现小于 1px 像素的边

随时显示屏技术的发展，很多移动设备的分辨率媲美甚至超过大屏设备

在 CSS 中设置的像素宽度1px，在高分辨率的小屏设备上，可能会变粗

现代webkit内核浏览器提供私有属性-webkit-min-device-pixel-ratio或-webkit-max-device-pixel-ratio

用来当前设备的物理像素分辨率与CSS像素分辨率比值的最小值和最大值

- 通过媒体查询，将边框宽度设为1px / devicePixelRatio

```
.border { border: 1px solid }
@media screen and (-webkit-min-device-pixel-ratio: 2) {
  .border { border: 0.5px solid }
}
@media screen and (-webkit-min-device-pixel-ratio: 3) {
  .border { border: 0.333333px solid }
}
```

- 伪类 + resolution + transform
resolution是标准属性，可以实现与上面的私有属性相同的设置
transform缩小伪类的高度为1 / devicePixelRatio

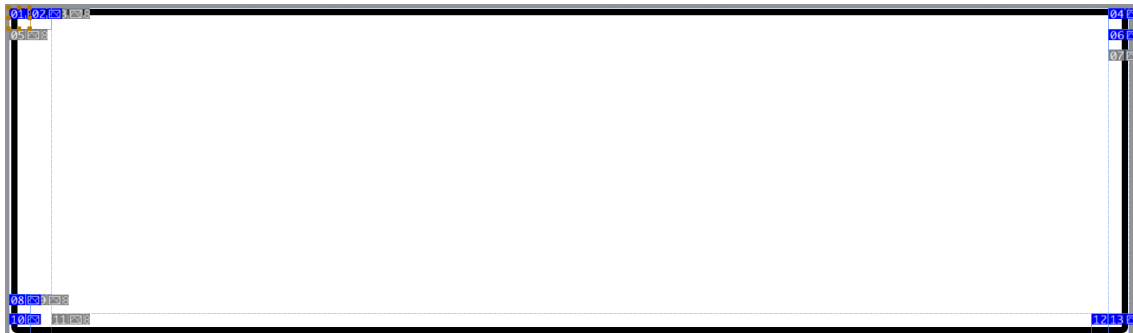
```
<style>
div::after {
  content: '';
  display: block;
  border-bottom: 1px solid;
}
@media only screen and (min-resolution: 2dppx) {
  div::after {
    -webkit-transform: scaleY(0.5);
    transform: scaleY(0.5);
  }
}
@media only screen and (min-resolution: 3dppx) {
  div::after {
    -webkit-transform: scaleY(0.33);
    transform: scaleY(0.33);
  }
}
</style>
<div></div>
```

对比多方法实现图标

实现方式	特殊符号	PNG / GIF	Sprites	Icon Font	SVG
大小	font-size	width height	background-size	font-size	width height
无锯齿	受 font-family 影响	放大可能有锯齿	放大可能有锯齿	Windows 字号较小时可能有锯齿	√
颜色	单色 用 color 设置	图片本身颜色 可用滤镜修改	图片本身颜色 可用滤镜修改	单色 用 color 设置	彩色 可修改
兼容性	不同浏览器显示效果可能不同	IE6 不支持 PNG透明	改大小IE9+ SVG Sprites IE9+		IE9+
响应式	-	√	√	-	√
透明	√	√	√	√	√
场景	简单图标 + emoji	常用	HTTP1.1时合并请求	自定义字体	可定制图标

多方法实现圆角边框

- 背景图片
 - 绘制圆角边框，切成 4 个圆角 + 4 个边框的小图片，拼成圆角边框



```
<style>
div {
  height: 250px;
```


background-image:

```
url(data:image/png;base64,ivBORw0KGgoAAAANSUheUGAAACIAAAAIcAIAAAC1JZyVAAAAGX
RFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAA01pVFh0WE1MOmNvbS5hZG9iZS54bX
AAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh6cmVtek5UY3prYzlkIj
8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWVrYmU6bnM6bWV0YS8iIHg6eG1wdGs9IkFkb2JlIFhNUC
BDb3JlIDYuMC1jMDAYIDc5LjE2NDQ4OCwgMjAyMC8wNy8xMCM0YmJowNjo1MyAgICAgICAgIj4gPH
JkZjpSREYgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzA5LzIyLXJkZi1zew50YX
gtbnMjIj4gPHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwOi8vb3
MuYWRvYmUuY29tL3hhcC8xLjAvIiB4bWxuczp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbS54YX
AvMS4wL21tLyIgeG1sbnM6c3RSZWY9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS
9SZXNvdXJzZVJlZiMiIHhtcDpDcmVhdG9yVGV9vbd0iQWRvYmUgUGhvdG9zaG9wIDIlYlJAgKDIIW
AwODE4Lm0uMTAxMjAyMDIwLzA4LzE4OjBmODJmNDMzNyYxKSAgKFdpbmRvd3MpIiB4bXBNTTJpbn
N0YW5jZU1EPSj4bXAuawlkOKU5RDQ1RjRENZlFOTExRUI5NkQ4QkU0NEUwMDgxOTI1IiB4bXBNTT
pEb2N1bWVudE1EPSj4bXAuZGlkOKU5RDQ1RjRfNzlfOTExRUI5NkQ4QkU0NEUwMDgxOTI1Ij4gPH
htcE1NOKRlcm12ZWRGcm9tIHN0UmVmOm1uc3RhbmNlSUQ9InhtcC5pawQ6RTlENDVGNEI3OUU5MT
FFQjk2RDhCRTQ0RTAwODE5MjUiIHN0UmVmOmRvY3VtZW50SUQ9InhtcC5kawQ6RTlENDVGNEI3OU
U5MTFFQjk2RDhCRTQ0RTAwODE5MjUiLz4gPC9yZGY6RGVzY3JpcHRpb24+IDwvcmlrOm1JERj4gPC
94OnhtcG1ldGE+IDw/eHBhY2tldCB1bmQ9InIiPz5lJHAbAAAAdE1EQVR42uzUsRHAIAgFUIh7MJ
RzsJND0CCx8nk5mIpp9X9l5VMQNSKkIHhOGKP3bmatNQizgLU79jbuDi8axHgqx1YVtYc5H58M8C
oi117NOSvMBliXJ5Mwg+fo/mxU9f80Ob2BhgwZMmTikCFDpIc3AAMAHlIubLgq7pnKAAAAASUVORK
5CYII=),
```

```
url(data:image/png;base64,ivBORw0KGgoAAAANSUheUGAAACEAAAAIcAIAAABeIewAAAAGX
RFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAA01pVFh0WE1MOmNvbS5hZG9iZS54bX
AAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh6cmVtek5UY3prYzlkIj
8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWVrYmU6bnM6bWV0YS8iIHg6eG1wdGs9IkFkb2JlIFhNUC
BDb3JlIDYuMC1jMDAYIDc5LjE2NDQ4OCwgMjAyMC8wNy8xMCM0YmJowNjo1MyAgICAgICAgIj4gPH
JkZjpSREYgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzA5LzIyLXJkZi1zew50YX
gtbnMjIj4gPHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwOi8vb3
MuYWRvYmUuY29tL3hhcC8xLjAvIiB4bWxuczp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbS54YX
AvMS4wL21tLyIgeG1sbnM6c3RSZWY9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS
9SZXNvdXJzZVJlZiMiIHhtcDpDcmVhdG9yVGV9vbd0iQWRvYmUgUGhvdG9zaG9wIDIlYlJAgKDIIW
AwODE4Lm0uMTAxMjAyMDIwLzA4LzE4OjBmODJmNDMzNyYxKSAgKFdpbmRvd3MpIiB4bXBNTTJpbn
N0YW5jZU1EPSj4bXAuawlkOKU5RDlEOUVBNZlFOTExRUI5NkQ4QkU0NEUwMDgxOTI1IiB4bXBNTT
pEb2N1bWVudE1EPSj4bXAuZGlkOKU5RDlEOUVBNZlFOTExRUI5NkQ4QkU0NEUwMDgxOTI1Ij4gPH
htcE1NOKRlcm12ZWRGcm9tIHN0UmVmOm1uc3RhbmNlSUQ9InhtcC5pawQ6RTlEOUQ5RTg3OUU5MT
FFQjk2RDhCRTQ0RTAwODE5MjUiIHN0UmVmOmRvY3VtZW50SUQ9InhtcC5kawQ6RTlEOUQ5RTk3OU
U5MTFFQjk2RDhCRTQ0RTAwODE5MjUiLz4gPC9yZGY6RGVzY3JpcHRpb24+IDwvcmlrOm1JERj4gPC
94OnhtcG1ldGE+IDw/eHBhY2tldCB1bmQ9InIiPz5h4NMzAAAAC01EQVR42uzU0QnAIAwE0Mbs4V
CZw50yha0mg1AE21Kov/px9yv4SEIIEBH9PKqaczYzd4+LrPQadiqtr6yUAjdmBjwnswkoo80Gbo
y1JJxRa4WvztEu7Hp2Q146InLz2j9PH9wbGjRo0KBBgwanJ9kFGAB4JIBHzc6fbAAAAABJRUErk
Jggg==),
```

url(data:image/png;base64,ivBORw0KGgoAAAANSuHEUGAAACIAAAAhCAIAAAZse47AAAAGX
RFWHRtb2Z0d2FyZQBZBG9iZSBjbWFnZVJlYWRS5cc1lPAAAA01pVFh0WE1MOmNvbS5hZG9iZS54bX
AAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh6cmVtek5UY3prYzlkIj
8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWVrYmU6bnM6bWV0YS8iIHg6eG1wdGs9IkFkb2JlIFhNUC
BDb3JlIDYuMC1jMDAYIDc5LjE2NDQ4OCwgMjAyMC8wNy8xMC0yMjowNjo1MyAgICAgICAgIj4gPH
JkZjpSREYgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zew50YX
gtbnMjIj4gPHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwOi8vb
nMuYWRvYmUuY29tL3hhcC8xLjAvIiB4bWxuczp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbS54YX
AvMS4wL21tLyIgeG1sbnM6c3RSZWY9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS
9SZXNvdXJjZVJlZiMiIHhtcDpDcmVhdG9yVGV9bD0iQWRvYmUgUGhvdG9zaG9wIDIlYlJAgKDIIWm
AwODE4Lm0uMTAxMjAyMDIwLzA4LzE4OjBmODJmNDMzNyYxKSAgKFdpbmRvd3MpIiB4bXBNTTJpbn
N0YW5jZU1EPSJ4bXAuawlkOKU5RjM1MzVENZlFOTExRUI5NkQ4QkU0NEUwMDgxOTI1IiB4bXBNTT
pEb2N1bWVudE1EPSJ4bXAuZGlkOKU5RjM1MzVFNzFOTExRUI5NkQ4QkU0NEUwMDgxOTI1Ij4gPH
htcE1NOKRlcm12ZWRGcm9tIHN0UmVmOm1uc3RhbmNlSUQ9InhtcC5pawQ6RTlGMzUzNUI3OUU5MT
FFQjk2RDhCRTQ0RTAwODE5MjUiIHN0UmVmOmRvY3VtZW50SUQ9InhtcC5kawQ6RTlGMzUzNUM3OU
U5MTFFQjk2RDhCRTQ0RTAwODE5MjUiLz4gPC9yZGY6RGVzY3JpcHRpb24+IDwvcmlRmO1JERj4gPC
94OnhtcG1ldGE+IDw/eHBhY2tldCB1bmQ9InIiPz7rnsZYAAAAek1EQVR42uzXsQ2AIBCFYU7GIG
EK1riGodiJUC9TEytpRF8o3l+TfAECATGzcCYiYdw97F1bgESGDBkyZMiQWZQ53hJXMUBefJCKK
aUgmBUfEbE3Xs4zWfTeEYzXWkmwf0n2lK/ex/tk4xyrtbo3f3Jl8kex1p22CzAA0z2GxKYdg5IAAA
AASUVORK5CYII=),

url(data:image/png;base64,ivBORw0KGgoAAAANSuHEUGAAACEAAAAAhCAIAAADYh1u4AAAAGX
RFWHRtb2Z0d2FyZQBZBG9iZSBjbWFnZVJlYWRS5cc1lPAAAA01pVFh0WE1MOmNvbS5hZG9iZS54bX
AAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh6cmVtek5UY3prYzlkIj
8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWVrYmU6bnM6bWV0YS8iIHg6eG1wdGs9IkFkb2JlIFhNUC
BDb3JlIDYuMC1jMDAYIDc5LjE2NDQ4OCwgMjAyMC8wNy8xMC0yMjowNjo1MyAgICAgICAgIj4gPH
JkZjpSREYgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zew50YX
gtbnMjIj4gPHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwOi8vb
nMuYWRvYmUuY29tL3hhcC8xLjAvIiB4bWxuczp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbS54YX
AvMS4wL21tLyIgeG1sbnM6c3RSZWY9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS
9SZXNvdXJjZVJlZiMiIHhtcDpDcmVhdG9yVGV9bD0iQWRvYmUgUGhvdG9zaG9wIDIlYlJAgKDIIWm
AwODE4Lm0uMTAxMjAyMDIwLzA4LzE4OjBmODJmNDMzNyYxKSAgKFdpbmRvd3MpIiB4bXBNTTJpbn
N0YW5jZU1EPSJ4bXAuawlkOKU5RjdGMDUwNzFOTExRUI5NkQ4QkU0NEUwMDgxOTI1IiB4bXBNTT
pEb2N1bWVudE1EPSJ4bXAuZGlkOKU5RjdGMDUwNzFOTExRUI5NkQ4QkU0NEUwMDgxOTI1Ij4gPH
htcE1NOKRlcm12ZWRGcm9tIHN0UmVmOm1uc3RhbmNlSUQ9InhtcC5pawQ6RTlGN0YwNEU3OUU5MT
FFQjk2RDhCRTQ0RTAwODE5MjUiIHN0UmVmOmRvY3VtZW50SUQ9InhtcC5kawQ6RTlGN0YwNEU3OU
U5MTFFQjk2RDhCRTQ0RTAwODE5MjUiLz4gPC9yZGY6RGVzY3JpcHRpb24+IDwvcmlRmO1JERj4gPC
94OnhtcG1ldGE+IDw/eHBhY2tldCB1bmQ9InIiPz5qWIGeAAAAeu1EQVR42uzVsQkAIRAEQ0+1C8
EqbMPk6rAns70XRDB4THQ/2k2FG3RRxczczQURkszczQPw4fGjRo0KBBgwY2Yf/pn8R7D99HjBFu5J
zhRikF23NkyWZQRmsNa9RabQkcuGz0DtYjumb0i9ZHq+rn9JEA7P3P9+oVYAANKmvpw7ZFZQAAAA
BJRU5ErkJggg==),

url(data:image/png;base64,ivBORw0KGgoAAAANSuHEUGAAACIAAAAIcAIAAAC1JZyVAAAAGX
RFWHRtb2Z0d2FyZQBBZG9iZSBjbWFnZVJlYWR5ccllPAAAA01pVfh0WE1MOMNvbs5hZG9iZS54bX
AAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh6cmVtek5UY3prYzlkIj
8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iIHg6eG1wdGs9IkFkb2JlIFhNUC
BDb3JlIDYuMC1jMDAYIDc5LjE2NDQ4OCwgMjAyMC8wNy8xMCM0YmJownJo1MyAgICAgICAgIj4gPH
JkZjpSREYgeG1sbnM6cmRmPSJodHRwoi8vd3d3LnczLm9yZy8xOTk5LzAylZiYlXJkzi1zew50YX
gtbnMjIj4gPHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwoi8vbn
MuYWRvYmUuY29tL3hhcC8xLjAvIiB4bWxuczp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbs94YX
AvMS4wL21tLyIgeG1sbnM6c3RSZWY9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS
9SZXNvdXJzZVJlZiMiIHhtcDpDcmVhdG9yVG9vbD0iQWRvYmUgUGhvdG9zaG9wIDIlYlJAgKDIIwMj
AwODE4Lm0uMTAxMjAyMDIwLzA4LzE4OjBmODJmNDMZNyYKSAGKFdpbmRvd3MpiIiB4bXBNTTJpbn
N0YW5jZU1EPSj4bXAuawlkOKU5RDQ1RjUxNzlfOTExRUI5NkQ4QkU0NEUwMDgxOTI1IiB4bXBNTT
pEb2N1bWVudElEPSj4bXAuZGlkOKU5RDQ1RjUyNzlfOTExRUI5NkQ4QkU0NEUwMDgxOTI1Ij4gPH
htcE1NOKRlcm12ZWRGcm9tIHN0UmVmOm1uc3RhbmNlSUQ9InhtcC5pawQ6RTlENDVGNEY3OUU5MT
FFQjk2RDhCRTQ0RTAwODE5MjUiIHN0UmVmOmRvY3VtZW50SUQ9InhtcC5kawQ6RTlENDVGNTA3OU
U5MTFFQjk2RDhCRTQ0RTAwODE5MjUiLz4gPC9yZGY6RGVzY3JpcHRpb24+IDwvcmlrOm1JERj4gPC
94OnhtcG1ldGE+IDw/eHBhY2tldCB1bmQ9InIiPz66lMmZAAAMElEQVR42uzUSREAAwCStj774
ynMJV+AR0Nk+RJ0nXTOZvfwYPBYDAYDAZTAUUYAHB9BimMTLcaAAAAAE1FTkSuQMCC),

url(data:image/png;base64,ivBORw0KGgoAAAANSuHEUGAAACEAAAAhCAIAAADYh1U4AAAAGX
RFWHRtb2Z0d2FyZQBBZG9iZSBjbWFnZVJlYWR5ccllPAAAA01pVfh0WE1MOMNvbs5hZG9iZS54bX
AAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh6cmVtek5UY3prYzlkIj
8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iIHg6eG1wdGs9IkFkb2JlIFhNUC
BDb3JlIDYuMC1jMDAYIDc5LjE2NDQ4OCwgMjAyMC8wNy8xMCM0YmJownJo1MyAgICAgICAgIj4gPH
JkZjpSREYgeG1sbnM6cmRmPSJodHRwoi8vd3d3LnczLm9yZy8xOTk5LzAylZiYlXJkzi1zew50YX
gtbnMjIj4gPHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwoi8vbn
MuYWRvYmUuY29tL3hhcC8xLjAvIiB4bWxuczp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbs94YX
AvMS4wL21tLyIgeG1sbnM6c3RSZWY9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS
9SZXNvdXJzZVJlZiMiIHhtcDpDcmVhdG9yVG9vbD0iQWRvYmUgUGhvdG9zaG9wIDIlYlJAgKDIIwMj
AwODE4Lm0uMTAxMjAyMDIwLzA4LzE4OjBmODJmNDMZNyYKSAGKFdpbmRvd3MpiIiB4bXBNTTJpbn
N0YW5jZU1EPSj4bXAuawlkOKU5RUEZRTc4NzlfOTExRUI5NkQ4QkU0NEUwMDgxOTI1IiB4bXBNTT
pEb2N1bWVudElEPSj4bXAuZGlkOKU5RUEZRTc5NzlfOTExRUI5NkQ4QkU0NEUwMDgxOTI1Ij4gPH
htcE1NOKRlcm12ZWRGcm9tIHN0UmVmOm1uc3RhbmNlSUQ9InhtcC5pawQ6RTlEOUQ5RjA3OUU5MT
FFQjk2RDhCRTQ0RTAwODE5MjUiIHN0UmVmOmRvY3VtZW50SUQ9InhtcC5kawQ6RTlFQTNFNzc3OU
U5MTFFQjk2RDhCRTQ0RTAwODE5MjUiLz4gPC9yZGY6RGVzY3JpcHRpb24+IDwvcmlrOm1JERj4gPC
94OnhtcG1ldGE+IDw/eHBhY2tldCB1bmQ9InIiPz7oTCF7AAAAAUlEQVR42uzNSQ0AIAwDwZj9dw
5C9GkQ3X1ryZfuroeSD0s9X/U/BoPBYDAYDAaDwThTAQYAYx8GP5x48QYAAAAASUVORK5CYII=),

url(data:image/png;base64,ivBORw0KGgoAAAANSuHEUGAAABsAAAAhCAIAAACNv0fJAAAAGX
RFWHRtb2Z0d2FyZQBBZG9iZSBjbWFnZVJlYWR5ccllPAAAA01pVfh0WE1MOMNvbs5hZG9iZS54bX
AAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh6cmVtek5UY3prYzlkIj
8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iIHg6eG1wdGs9IkFkb2JlIFhNUC
BDb3JlIDYuMC1jMDAYIDc5LjE2NDQ4OCwgMjAyMC8wNy8xMCM0YmJownJo1MyAgICAgICAgIj4gPH
JkZjpSREYgeG1sbnM6cmRmPSJodHRwoi8vd3d3LnczLm9yZy8xOTk5LzAylZiYlXJkzi1zew50YX
gtbnMjIj4gPHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwoi8vbn
MuYWRvYmUuY29tL3hhcC8xLjAvIiB4bWxuczp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbs94YX
AvMS4wL21tLyIgeG1sbnM6c3RSZWY9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS
9SZXNvdXJzZVJlZiMiIHhtcDpDcmVhdG9yVG9vbD0iQWRvYmUgUGhvdG9zaG9wIDIlYlJAgKDIIwMj
AwODE4Lm0uMTAxMjAyMDIwLzA4LzE4OjBmODJmNDMZNyYKSAGKFdpbmRvd3MpiIiB4bXBNTTJpbn
N0YW5jZU1EPSj4bXAuawlkOKU5RjdGMDRDNzlfOTExRUI5NkQ4QkU0NEUwMDgxOTI1IiB4bXBNTT
pEb2N1bWVudElEPSj4bXAuZGlkOKU5RjdGMDRENzlfOTExRUI5NkQ4QkU0NEUwMDgxOTI1Ij4gPH
htcE1NOKRlcm12ZWRGcm9tIHN0UmVmOm1uc3RhbmNlSUQ9InhtcC5pawQ6RTlGN0YWNEE3OUU5MT
FFQjk2RDhCRTQ0RTAwODE5MjUiIHN0UmVmOmRvY3VtZW50SUQ9InhtcC5kawQ6RTlGN0YWNEI3OU
U5MTFFQjk2RDhCRTQ0RTAwODE5MjUiLz4gPC9yZGY6RGVzY3JpcHRpb24+IDwvcmlrOm1JERj4gPC
94OnhtcG1ldGE+IDw/eHBhY2tldCB1bmQ9InIiPz5dZ8F+AAAAlk1EQVR42mL8//8/A1UBew01wa
iJoya0mjhq4qiJoya0glEwcsgFjEOgtQCQYACtHAYnh7f45gAAAAABJRUErKjggg==),

```
url(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAACIAAAAFCAIAAAANwYjmAAAAGX
RFWHRtb2Z0d2FyZQBZBG9iZSBjbWFnZVJlYW5cc1lPAAAA01pVFh0WE1MOmNvbS5hZG9iZS54bX
AAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkIj
8+IDx4OnhtcG1ldGEgeG1sbnM6eD0iYWVrVYmU6bnM6bWV0YS8iIHg6eG1wdG9iKfkb2JlIFhNUC
BDdb3JlIDYuMC1jMDAyIDc5LjE2NDQ4OCwgMjAyMC8wNy8xMC0yMjowNjo1MyAgICAgICAgIj4gPH
JkZjpsREYgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zew50YX
gtbnMjIj4gPHJkZjpeZXNjcm1wdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwOi8vbn
MuYWRvYmUuY29tL3hhcC8xLjAvIiB4bWxuczp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbs94YX
AvMS4wL21tLyIgeG1sbnM6c3RSZWY9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS
9SZXNvdXJzZVJlZiMiIHhtcDpDcmVhdG9yVGV9bD0iQWRvYmUgUGhvdG9zaG9wIDIlYlJAgKDIIWmJ
AwODE4Lm0uMTAxMjAyMDIwLzA4LzE4OjBmODJmNDMzNyXKSAGkFdpbmRvd3MpIiB4bXBNTTJbn
N0Yw5jZU1EPSJ4bXAuawlkOKU5RUEzRTgwNz1FOTExRUI5NkQ4QkU0NEUwMDgxOTI1IiB4bXBNTT
pEb2N1bWVudE1EPSJ4bXAuZGlkOKU5RUEzRTgxNz1FOTExRUI5NkQ4QkU0NEUwMDgxOTI1Ij4gPH
htcE1NOKRlcm12ZWRGcm9tIHN0UmVmOm1uc3RhbmNlSUQ9InhtcC5pawQ6RT1FQTNFN0U3OUU5MT
FFQjk2RDhCRTQ0RTAwODE5MjUiIHN0UmVmOmRvY3VtZW50SUQ9InhtcC5kawQ6RT1FQTNFN0Y3OU
U5MTFFQjk2RDhCRTQ0RTAwODE5MjUiLz4gPC9yZGY6RGVzY3JpcHRpb24+IDwvcmlrOm1JERj4gPC
94OnhtcG1ldGE+IDw/eHBhY2tldCB1bmQ9InIiPz40NhQHAAAAAN01EQVR42uzNIQ4AIBAesVv+/+
cjQeAwCILo6Ema7q5VkjQ3t7tGPQmDWWAWGAWGg8F8yUwBBgDDegY72UAQEAAAAABJRUS5ErkJggg
==);
background-repeat: no-repeat, no-repeat, no-repeat, no-repeat, repeat-x,
repeat-y, repeat-x, repeat-y;
background-position: top left, top right, bottom left, bottom right, top,
right, bottom, left;
}
</style>
<div></div>
```

- border-radius

```
<style>
div {
  height: 250px;
  border: 10px solid;
  border-radius: 10px;
}
</style>
<div></div>
```

- clip-path: 创建裁剪区域

```
<style>
div {
  height: 250px;
  border: 10px solid;
  clip-path: inset(0 round 10px);
}
</style>
<div></div>
```

多方法实现小三角

- 伪类 + 特殊符号



```
<style>
div {
float: left;
margin-right: 10px;
}
.top::before {
content: '▲'
}
.right::before {
content: '▶';
}
.bottom::before {
content: '▼';
}
.left::before {
content: '◀';
}
</style>

<div class="top"></div>
<div class="right"></div>
<div class="bottom"></div>
<div class="left"></div>
```

- 特殊符号 + 缩进 + 溢出 + 旋转



```
<style>
div {
float: left;
margin-right: 6px;
font-size: 20px;
font-family: '宋体';
overflow: hidden;
}
.top {
text-indent: -10px;
transform: rotate(-90deg);
}
.right {
text-indent: -10px;
}
.bottom {
width: 10px;
transform: rotate(-90deg);
}
.left {
```

```
width: 10px;
}
</style>

<div class="top">◆</div>
<div class="right">◆</div>
<div class="bottom">◆</div>
<div class="left">◆</div>
```

- 宽高为 0 的边框



```
<style>
div {
float: left;
margin-right: 6px;
width: 0;
height: 0;
border: 10px solid transparent;
}
.top {
border-top-color: black;
}
.right {
border-right-color: black;
}
.bottom {
border-bottom-color: black;
}
.left {
border-left-color: black;
}
</style>

<div class="top"></div>
<div class="right"></div>
<div class="bottom"></div>
<div class="left"></div>
```

多方法隐藏元素

属性	不占位	读屏软件隐藏	用于隐藏主体内容 SEO
display: none	√	√	不抓取
visibility: hidden	×	√	可能抓取
opacity: 0	×	×	疑似作弊
input type = hidden	√	√	不抓取
position: absolute / fixed	√	×	疑似作弊
aria-hidden = true	×	√	未知
text-indent < 0	√	×	常用于在 Logo 处标识网站名称
font-size: 0	√	×	疑似作弊
overflow: hidden	√ (裁剪)	×	抓取
clip-path: polygon(0 0, 0 0, 0 0, 0 0, 0 0)	√	×	未知

实现文字描边

- text-shadow

文字描边

```
<style>
div {
font-size: 100px;
color: #fff;
text-shadow: 0 0 3px black;
}
</style>
<div>文字描边</div>
```

- -webkit-text-stroke

文字描边

```
<style>
div {
font-size: 100px;
color: #fff;
-webkit-text-stroke: 3px black;
}
</style>
<div>文字描边</div>
```

- position: relative/position: absolute子绝父相

文字描边

```
<style>
div {
position: relative;
font-size: 100px;
color: #fff;
}
div p {
position: absolute;
margin: 0;
}
div p:first-child {
font-weight: bolder;
color: black;
}
</style>
<div>
<p>文字描边</p>
<p>文字描边</p>
</div>
```

实现渐变背景



- 背景图片

一张宽 1px 像素，高度沿渐变方向固定，重复铺满即可

```
<style>
div {
  height: 250px;
  background-image:
url(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAEAAAAD6CAIAAAADViroEAAAAGX
RFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAA01pVFh0WE1MOmNvbS5hZG9iZS54bX
AAAAAADw/eHBhY2tldCBiZWdpbj0i77u/IiBpZD0iVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkIj
8+IDX40nhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iIHg6eG1wdGs9IkFkb2JlIFhNUC
BDb3JlIDYuMC1jMDAyIDc5LjE2NDQ4OCwgMjAyMC8wNy8xMCM0YmJownJo1MyAgICAgICAgIj4gPH
JkZjpSREYgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zew50YX
gtbnMjIj4gPHJkZjpEZXNjcm1wdGlvbiByZGY6YWJvdXQ9IiIgeG1sbnM6eG1wPSJodHRwOi8vbn
MuYWRvYmUuY29tL3hhcC8xLjAvIiB4bWxuczp4bXBNTT0iaHR0cDovL25zLmFkb2JlLmNvbS94YX
AvMS4wL21tLyIgeG1sbnM6c3RSZWY9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC9zVHlwZS
9SZXNvdXJzZVJlZiMiIHhtcDpDcmVhdG9yYyY9Ym90IiQWRvYmUgUGhvdG9zaG9wIDIlYlAgKDlW
AwODE4Lm0uMTAxMiAyMDIwLzA4LzE4OjBmODJmNDMzNzYxKSAgKFdpbmRvd3MpIiB4bXBNTT0jbn
N0YW5jZU1EPSj4bXAuawlkokQxQ0JDMDQyZnZlEOTExRUJBQkYyQzQzRUVCMDZDNjg0IiB4bXBNTT
pEb2N1bWVudE1EPSj4bXAuZGlkQkQxQ0JDMDQyZnZlEOTExRUJBQkYyQzQzRUVCMDZDNjg0Ij4gPH
htcE1NOKRlcm12ZWRGcm9tIHNOUmVmOm1uc3RhbmNlSUQ9InhtcC5pawQ6RDFDQkMwNDA3OUQ5MT
FFQkFCRjJDNDFRUIzRkM2ODQiIHNOUmVmOmRvY3VtZW50SUQ9InhtcC5kawQ6RDFDQkMwNDE3OU
Q5MTFFQkFCRjJDNDFRUIzRkM2ODQiLz4gPC9yZGY6RGVzY3JpcHRpb24+IDwvc3RmOm1JERj4gPC
94OnhtcG1ldGE+IDw/eHBhY2tldCB1bmQ9InIiPz470lwnAAAAp0lEQVR42pRTQQ6DMAXLInHc/x
+0M/8BawAKbQQobTQdkGmtuLHTCr6zydtNRazlq6jvtRZEQ8Y3vP81cBewekArMp5ouy70k/dFee
+B+OprEF9ei1NTu3UisPo7M2k89ZVkeu0jng33GM9odbv5efIqSS4dTz4jlvlA/SsXKM98JFfnH7
6qxxh93hd7jbn7ZvDwbJX5G5h3fQcy31i4b7DOZrL/9EGAAXQGYLZj/m8EAAAAASUVORK5CYII=);
  background-repeat: repeat-x;
  background-size: 1px 100%;
}
</style>
<div></div>
```

- linear-gradient

```
<style>
div {
  height: 250px;
  background-image: linear-gradient(to bottom, pink, skyblue);
}
</style>
<div></div>
```

对比常见图片格式和 base64 图片？

拓展名	MIME	透明	动画	IE兼容性	特点
.ico	image/x-icon	√	×	IE	早期浏览器只支持 favicon.ico
.bmp	image/bmp	×	×	IE	windows 系统原生支持，图片信息丰富
.jpg	image/jpeg	×	×	IE	可控制品质，模糊到清晰或渐进加载，常见于照片
.png	image/png	√	×	IE6（不透明）	可控制品质、位数，常见于绘画
.gif	image/gif	√	√	IE	基于颜色透明，有毛边，256颜色上限，适合动画
.svg	image/svg+xml	√	×	IE9+	无损放大，可调色、修改，更常见于图标
.apng	image/apng	√	√	Edge	Mozilla主导，兼容PNG浏览器即可显示第一帧，压缩率较高，无毛边
.webp	image/webp	√	√	Edge	Cooogle主导，压缩率更高，无毛边，常见于微信公众号
.avif	image/avif	√	√	×	Netflix主导，压缩率更高，无毛边，同压缩比效果有时超过 webp

可以将图片转为 Base64 编码，直接将编码放入 CSS 中，即可引入图片

编码后的图片通常比原图大 30% 或更多，但可以与 CSS 一起被 gzip 或 br 压缩

适用小图片和没有权限上传图片的场景，来减少请求，但也应设置代码编辑器不换行或折叠图片编码区域，避免影响 CSS可读性

为什么要重置浏览器默认样式，对比 Reset.css 和 Normalize.css?

什么是浏览器默认样式

对于部分HTML标签，如段落、列表，部分表单元素，浏览器会提供默认样式，包含外观及交互，开发者只需引入标签，不需要重复定义这些样式，便于开发

为什么要重置默认样式

- 不同浏览器，默认样式可能不同，特别是尺寸，位置的不同，让开发者无法统一用户体验，甚至有错位的风险
- 只使用标签的语义，而不想引入样式
- UI稿与浏览器默认样式不同

基于以上，都需要开发者重置或部分重置浏览器的默认样式，以满足开发需求

对比常用的 Reset.css 和 Normalize.css

共同点

- 两者都能抹平浏览器间的默认样式差异
- 都部分重置了浏览器默认样式，尤其是内外边距属性

不同点

- Reset.css让元素在不同浏览器样式完全一样
- Normalize.css适当保留部分浏览器默认样式，只重置影响开发的样式，此外
 - Normalize.css修复了表单、SVG溢出等BUG
 - Normalize.css适当提高了可用性
 - Normalize.css避免大量使用群组选择器，通过注释提高调试体验

最佳实践

对于绝大多数小型项目：

- 只重置在页面中使用到的标签
- 只重置有默认属性的属性名
- 适当保留浏览器的默认样式，如表单的outline

工程化

对比 Less、Sass、Stylus、PostCSS?

Less、Sass 和 Stylus 是 CSS 预处理器，PostCSS 是转换 CSS 工具的平台

Less

- 变量：\$标识符变量，使用{}插值
- 嵌套：支持{}大括号嵌套
- 混入器：支持 选择器 混入 或使用.selector(@param)创建纯混入器
- 扩展 / 继承 / 运算符 / @import：支持
- 流程：支持if条件判断，支持when递归模拟循环
- 映射：支持@声明和使用 Map
- 特有：提供 Less.js，直接在浏览器使用 Less

SaSS

- 变量：支持\$标识符变量，使用#{ }插值
- 嵌套：SCSS 支持{ }大括号嵌套 SASS 支持缩进嵌套
- 混入器：@mixin创建@include使用
- 扩展 / 继承 / 运算符 / @import：支持
- 流程：支持if else条件判断，支持for while each循环
- 映射：支持\${ }声明 Map，提供map-get(map, key) map-keys(map) map-values(map)等一系列方法操作 Map，支持遍历 Map
- 特有：支持 compass，内含 自动私有前缀 等一系列有用SASS模块，支持压缩输出

Stylus

- 变量：支持\$标识符变量 和 赋值表达式变量，使用{}插值

- 嵌套：支持{ }大括号嵌套 和 缩进嵌套
- 混入器：像函数一样创建和使用
- 扩展 / 继承 / 运算符 / @import：支持
- 流程：支持if else unless三元 条件判断，支持for循环
- 映射：像 JS 一样创建和使用对象
- 特有：支持中间件，自动分配函数变量，提供 JS API。支持压缩输出

PostCSS

- 接受 CSS 文件，把 CSS 规则转换为抽象语法树
- 提供 API 供插件调用，对 CSS 处理
- 插件：支持 Autoprefixer 自动添加私有前缀、css-modules CSS 模块 stylelint CSS 语法检查等插件，PostCSS 是工具的工具

Webpack 处理 SASS 文件时，sass-loader, css-loader, style

作用

- sass-loader
 - 将 SASS / SCSS 文件编译成 CSS
 - 调用node-sass，支持options选项向node-sass传参
- css-loader
 - 支持读取 CSS 文件，在 JS 中将 CSS 作为模块导入
 - 支持 CSS 模块 @规则@import @import url()
- style-loader
 - 将 CSS 以<style>标签的方式，插入 DOM

配置顺序

Webpack中 loader 的加载顺序是从后往前

在处理 SASS / SCSS 文件时，三者的配置顺序 style-loader css-loader sass-loader

可以用插件 ExtractTextWebpackPlugin 替换 style-loader，生成新的 CSS 文件

如何压缩 CSS 大小，如何去除无用的 CSS?

压缩 CSS，简单可分为 3 步骤：

- ① 去除 CSS 中的换行和空格
- ② 去除 每个选择器，最后一个属性值后的分号;
- ② 去除 注释，正则/[^\s][^\s]*+//

此外，包括使用 缩写属性，z-index值的重新排序，也可以减少代码量。但通过工具进行时，结果不一定总是安全

常用的 CSS 压缩工具有：

- YUI Compressor
 - 基于 Java，本来是 JS 压缩，也兼容 CSS 压缩，配置较少，但保障压缩安全
- clean-css
 - 基于 NodeJS，可以根据 浏览器版本、具体的 CSS 标准模块，详细配置兼容性
- cssnano

基于 PostCSS，是 Webpack5 推荐的 CssMinimizerWebpackPlugin 默认的 CSS 压缩工具
如何去除无用的 CSS：

- Chrome 开发者工具 Lighthouse
- 打开 http / https 网址，勾选 Performance 选项，在性能报告中，会列出 unused CSS，可以人工去除
- UnCSS
 - 通过 jsdom 加载 HTML 并执行 JS
 - 通过 PostCSS 解析样式
 - 通过 document.querySelector 查找 HTML 中不存在的 CSS 选择器
 - 返回 剩余样式
- PurgeCSS
 - 可以对 HTML JS 和 VUE 等框架中 CSS 使用情况分析，并删除无用 CSS
 - 提供 Webpack Gulp Grunt 等工程化工具的插件
- cssnano
 - 提供 discardUnused 选项，用于删除与当前 CSS 无关的规则
 - 不支持内容分析
 - 同样支持工程化工具

什么是 CSS 模块化，有哪几种实现方式？

什么是 CSS 模块化？

CSS 模块化是将 CSS 规则 拆分成相对独立的模块，便于开发者在项目中更有效率地组织 CSS：

- 管理层叠关系
- 避免命名冲突
- 提高复用度，减少冗余
- 便于维护或扩展

CSS 模块化的方式：

- 基于文件拆分
- 不拆分但设置作用域
- CSS in JS
- 内联样式、Shadow DOM 等

无论哪种方式，核心都是通过 保证 CSS 类命名唯一，或者 避免命名使用内联样式，来模拟出 CSS 模块作用域的效果

基于文件的 CSS 模块的加载

- <link>
将不同模块的 CSS 分文件存放，通过 <link> 标签按需引入
- @import
@规则，将其它 CSS 嵌入当前 CSS
除现代浏览器外，也得到了 CSS 预处理器 Less、Sass 和 Stylus 的支持
- import
在 Webpack 中，将 CSS 作为资源引入，并可以通过 CSS Modules 为 CSS 生成独一无二的类名

CSS 模块化的实现方式

- 分层拆分
将 CSS 规则 分层存放，并约束不同层次间的命名规则
 - SMACSS：按功能分成 Base Layout Module State Theme 五层

- ITCSS：按从通用到具体分成 Settings Tools Generic Base Objects Components 七层
- 分块拆分

将页面中视觉可复用的块独立出来，仅使用类选择器，并确保类名唯一

 - OOCSS
 - 将盒模型规则与颜色、背景等主题规则拆分
 - 将视觉可复用的块类名与页面结构解耦
 - BEM
 - 将页面按 Block Element Modifier 划分
 - 类名规则 block-name__element-name--modifier-name
- 原子化拆分

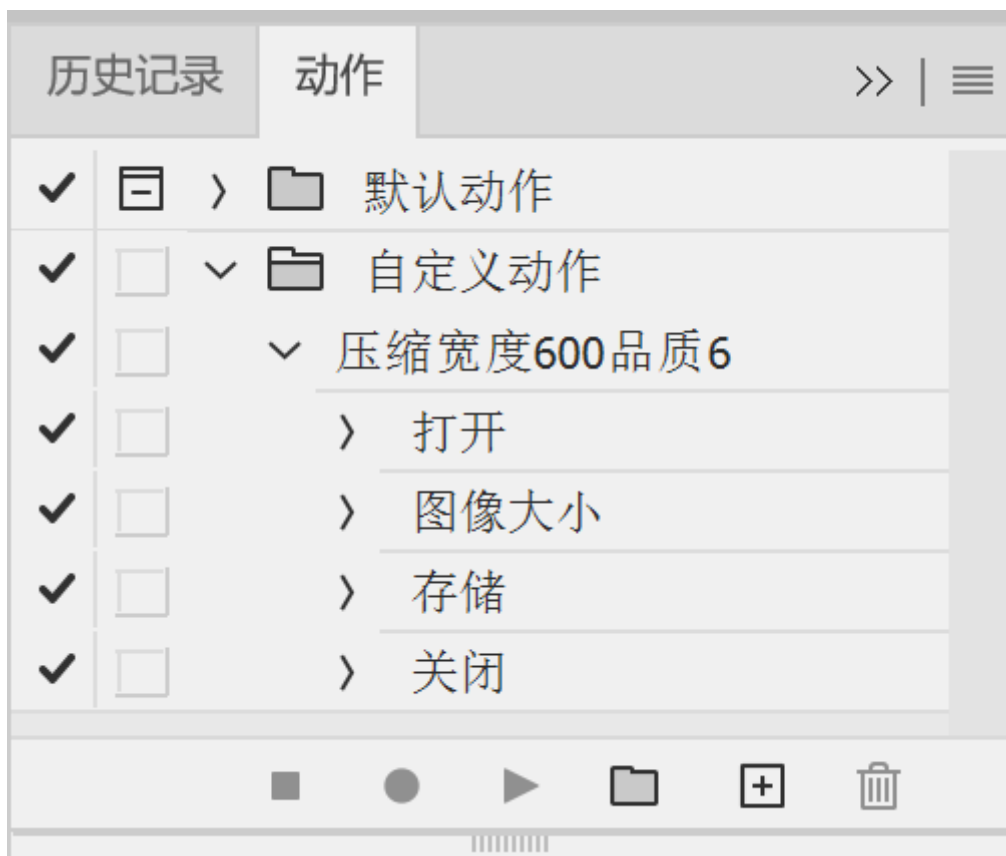
每个选择器只包含1个或少量属性，通过组合选择器定义复杂样式

 - ACSS：属性名像函数，属性值像函数参数，像写内联样式一样组合类名
 - Utility-First CSS：提供一套可配置的 CSS 实用类库，使用时按需编译
- CSS in JS
 - CSS Modules
 - 将 CSS 作为资源引入
 - 根据内容的哈希字符串，计算出独一无二的类名，CSS 规则 只对该类名下的元素生效
 - styled-components Aphrodite Emotion 等
 - 通过 JS 创建包含属性定义的组件
 - 生成独一无二的类名
 - Radium 等
 - 通过 JS 创建包含属性定义的组件
 - 生成内联样式
- Shadow DOM

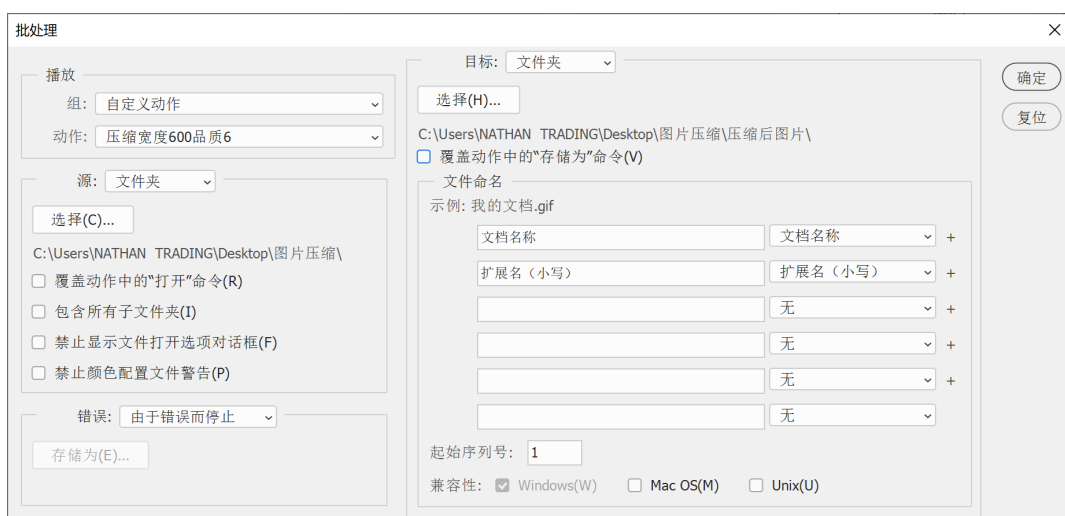
通过attachShadow给元素的影子DOM，附加<style>标签，其中规则不会影响外部元素。代表的框架有 Ionic 等

如何自动压缩图片？

- 使用或调用软件
 - 使用 PhotoShop 自动批处理功能
 - 录制动作：打开文件，调整大小，另存为调整品质，关闭文件



- 文件菜单，自动，批处理，选择 源文件夹，批量执行动作



- 使用 开源图片处理软件 XnViewr, 工具, 批量转换功能



XnConVenter 提供更复杂的图片批量处理功能

- 使用 命令行工具, 包括 NConvert ImageMagic 等以及类似 img2webp 的专门格式的转换工具
- 工程化配置图片压缩
这里以Webpack为例
 - CSS 内联图片
配置url-loader小于指定尺寸图片转 base64, 减少请求

```
module: {
  rules: [{
    test: /\. (png|jpe?g|gif|svg) (\?.*)?$/,
    loader: 'url-loader',
    options: {
      limit: 10, // 小于10KB图片, 转base64编码
    }
  }]
}
```

- 图片模块
import / require 到 .js 或 .html 以及 require({图片路径}) 的图片
这些图片被打包器看作是资源模块引入
配置file-loader+image-webpack-loader调节品质


```
rules: [{
  test: /\.png$/i, // 以png为例
  use: [
    'file-loader',
    {
      loader: 'image-webpack-loader',
      options: {
        pngquant: {
          quality: [0.65, 0.90] // 设置png的品质区间
        },
      }
    },
  ],
}]
```

◦ 图片目录

图片以静态资源附件的形式，放在同一目录下

通过CopyPlugin将图片从源目录（src）复制到发布目录（dist）

可以安装imagemin-webpack-plugin，在复制时，批量调整图片尺寸

```
npm i imagemin-webpack-plugin -D
plugins: [
  new CopyPlugin({
    patterns: [{
      from: resolve('src-es6/images'),
      to: resolve('dist/images'),
    }]
  }),
  new ImageminPlugin({ // 在 CopyPlugin 之后。如果是 merge 多配置文件，CopyPlugin
    放 common 被合并的配置里
    test: /\. (png|jpe?g)$/i,
    pngquant: {
      quality: '70-75' // 设置 png 的品质区间
    },
    plugins: [ // 可以引入其它图片压缩插件
      ImageminMozjpeg({
        quality: 70, // 设置 jpeg / jpg 的品质
        progressive: true // 渐进式: true: 由模糊到清晰加载 false: 从上到下加载
      })
    ]
  }),
]
```

代码中使用的 ImageminMozjpeg 需要额外安装，您也可以安装其它压缩插件

```
npm i imagemin-mozjpeg -D
```

◦ 使用反向代理

- 部分CDN服务商，提供一键 图片瘦身，自动转 webp 等能压缩图片服务
- 您可以自建反向代理节点，将图片缓存在节点上，根据客户端接受的MIME文件类型列表，如支持 image/webp 的客户端，将图片转成 webp 格式。根据用户 UserAgent，调节品质、尺寸后，再返回给客户端

需要提醒的是，避免使用 deflate gzip 和 br 等压缩算法，再次压缩图片，这样带来的性能损耗通常高于传输收益

如何自动添加浏览器私有前缀属性？

- 使用 SASS
 - 自定义添加浏览器私有前缀的 `Mixins`，不能适应所有属性

```
@mixin autoPrefix($propertyName, $propertyValue, $prefixs: ()) {
  @each $prefix in $prefixs {
    -#{$prefix}-#{$propertyName}: $propertyValue;
  }
  #{$propertyName}: $propertyValue;
}
div {
  @include autoPrefix(box-shadow, 0 0 1px black, ('webkit', 'moz'))
}
转换为 CSS
div {
  -webkit-box-shadow: 0 0 1px black;
  -moz-box-shadow: 0 0 1px black;
  box-shadow: 0 0 1px black;
}
```

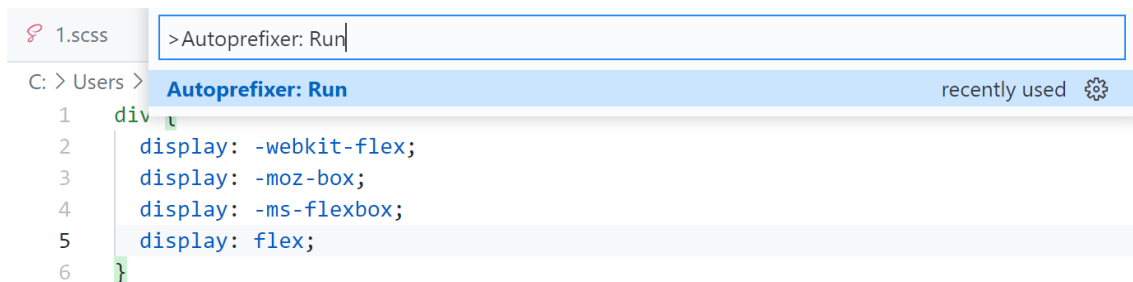
- 搭配 compass 或 Bourbon

```
@import "compass/css3";
div {
  @include box-shadow(0 0 1px black);
}
```

- 使用 Less，搭配 LESS Elements

```
@import "elements.less";
div {
  .box-shadow(0 0 1px black);
}
```

- 使用 Autoprefixer
 - 支持补全属性值，支持按浏览器兼容情况精确补全



- VsCode
 - 安装 Extensions: Autoprefixer

进入 Vscode Preferences Settings 搜索 Autoprefixer
配置支持的浏览器版本

```
"autoprefixer.options": {  
  "browsers": [  
    "ie >= 6",  
    "firefox >= 8",  
    "chrome >= 24",  
    "opera >= 10"  
  ]  
}
```

F1 后, Run Autoprefixer, 自动补全

- Webpack
安装并添加 postcss-loader

```
module.exports = {  
  module: {  
    rules: [  
      {  
        test: /\.css$/,  
        use: ["css-loader", "postcss-loader"]  
      }  
    ]  
  }  
}
```

创建 postcss.config.js

```
module.exports = {  
  plugins: [  
    require('autoprefixer')  
  ]  
}
```

创建 .browserslistrc

```
ie >= 6  
firefox >= 8  
chrome >= 24  
opera >= 10
```

或在 package.json 添加 browserslist 属性

```
{
  "browserslist": [
    "ie >= 6",
    "firefox >= 8",
    "chrome >= 24",
    "opera >= 10"
  ]
}
```

对比媒体查询与按需引入 CSS?

- 媒体查询
 - 允许我们通过设备、屏幕、使用场景、用户偏好来解析符合条件的 CSS，忽略不符合条件的 CSS
 - 即使通过 Link 标签附加媒体查询条件引入的 CSS，不符合条件时依然会被下载
- 按需引入

按需引入，可以避免冗余下载的问题，允许我们按照目标环境引入CSS
但条件没有媒体查询丰富，多数时，无法及时对运行环境的变化作出响应
- IE 浏览器中，可以通过 HTML 注释，附加 if 条件，阻止其中的 HTML 被解析，相应的 CSS 也不会被下载

```
<!--[if lt IE 9]><link rel="stylesheet" href="ie&ltlt9.css" /><![endif]-->
```

- 工程化及ES6等模块化环境中，我们可以通过import或require只引入需要的css，通过PurgeCSS和cssnano去除未被使用的CSS
- 在webpack配置DefinePlugin自定义环境变量，在uni-app中 使用条件注释等利用打包工具提供的 条件编译 功能

方法论

列举 CSS3 新特性

什么是 CSS3?

自 CSS2.1 后，CSS 标准被拆解成多个模块，每个模块有自己的版本并独立更新

CSS3 泛指这些模块的总和，作为 CSS 的第3版本的 CSS3 事实上已不存在

什么是 CSS3 新特性?

CSS 标准的各个模块都在快速更新，其中已经进入到候选推荐、建议推荐和推荐 的模块被称为稳定模块

稳定模块中新增的特性大多已获得浏览器广泛支持，使用不需要加私有前缀

这里的新特性通常指这些模块中新增的标准

CSS3 新特性有哪些，举例说明?

- 颜色模块
 - 新增opacity属性
 - 新增hsl() hsla() rgb() rgba()方法
 - 新增 颜色关键字 currentColor
 - 定义transparent为rgb(0, 0, 0, 0.0)
- 选择器模块
 - 新增 属性选择器: [attribute^="value"] [attribute\$="value"] [attribute*="value"]

- 新增 伪类: :target :enabled :disabled :checked :indeterminate :root :nth-child :nth-last-child :nth-of-type :nth-last-of-type :last-child :first-of-type :last-of-type :only-child :only-of-type :empty :not
 - 新增 普通兄弟选择器: ~
 - 规范 伪元素表示为两个冒号: ::after ::before ::first-letter ::first-line
- 命名空间模块
 - 新增 @规则: @namespace
- 媒体查询模块
 - 支持更多媒体查询条件: tv color resolution等
 - 支持<link>标签的媒体查询
- 背景和边框模块
 - 支持渐变linear-gradient背景
 - 支持多背景图片
 - 新增background-origin background-size background-clip
 - 新增 圆角边框: border-radius
 - 新增 边框图片: border-image
 - 新增 边框阴影: box-shadow
- 多列布局模块
 - 支持多列布局: columns column-count column-width等
- 值和单位模块
 - 新增 相对长度单位: rem ch vw vh vmax vmin
 - 新增方法: calc()
- 弹性盒布局模块
 - 支持弹性布局: display:flex flex-direction flex-wrap等
- 文本装饰模块
 - 新增 着重符号: text-emphasis
 - 新增 文本阴影: text-shadow

什么是 CSS 组件化和原子化?

组件化

CSS 组件是样式页面样式可复用的最小元素组合

过去, 前端常会按照页面结构, 将公共部分提取成组件, 方便在其它页面调用
好处:

- 无需重复编写 CSS, 结构级别的复用
- 保持风格统一
- 一次修改, 所有页面都生效
- 通过新建修饰符类的方式, 个性化组件

缺陷:

- 限制设计师发挥, 设计师可能需要先参考组件库已有组件的设计
- 需要个性化的属性较多时, 修饰原有组件和新建组件, 都可能产生新的代码冗余
- 组件库随项目越来越大, 逐渐难以维护

原子化

CSS 原子化的每个原子类, 几乎只包含一种属性定义

这些原子类可以继续组合, 生成属性更丰富的组件类

前端更关注不同元素间的相同属性, 即使这些元素的结构完全不同

好处:

- 无需重复编写 CSS，属性级别的复用
- 保持风格统一
- 一次修改，所有页面都生效
- 更灵活，设计师可以发挥，前端可以自由组合
- 类名与业务松耦合，甚至可以为不同业务定制
- 组件的样式一目了然，容易维护

缺陷：

- 无统一标准，类名自己起，他人使用需要学习成本
- HTML 类名顺序，无法决定 CSS 优先级，HTML 在后面的类名，可能因为在 CSS 定义中在前，相同属性的值被 CSS 定义中后面的类名覆盖。违反心智模型
- 项目较小时，编写原子库不经济
- CSS-in-JS 改进原子化缺陷
 - 类名无统一标准，需要学习

类名由 JS 根据内容生成，相同 CSS 的类名相同，不同 CSS 的类名不同且唯一，前端不用起类名
 - 使 HTML 类名顺序生效

原子类只有一个属性，这意味着，我们把 CSS 类定义写进 JS，由 JS 根据 类名顺序，若属性名重复，则采用最后出现的属性值
 - 此外，CSS-in-JS 还将样式与组件放到一起，能够从根本上避免组件移除，而样式忘记移除的问题

原子化思想早已有之，Atomic CSS 和 tailwindcss 最有代表性

更多组件库并不是只用组件化或原子化，而是通过组件化提供易用的组件和有限的可定制接口，通过原子化提供响应式布局、边距、颜色、字体等细粒度控制类 在工作中，组件化和原子化都能满足绝大多数的业务场景，是否应用更多地受设计思想和开发习惯的影响

对于业务组件来说，使用公用原子类，影响组件的独立性。而仅在组件内部使用原子类，又会大大降低原子类的优势

开发业务类组件，使用组件化的 CSS 依然是首选，除非我们决定引入或定义一套原子类 CSS，作用于整个组件库

长期来看，原子化正越来越得到关注，结合 CSS-in-JS 的原子化未来可期

对比多种编写规则

OOCSS，SMACSS，BEM，ITCSS，Utility-First CSS，ACSS 都是 CSS 编写规则，是结构化编写和组织 CSS 的指南，掌握一种 CSS 方法论，更利于他人理解您的代码和团队协作

- OOCSS

面向对象的 CSS，全称是 Object Oriented CSS

我们把样式重复的代码片段，定义为 CSS 对象

- 盒模型与皮肤无关：盒模型属性与颜色、背景等皮肤主题属性分选择器定义
- 样式与位置无关：避免使用标签、关系选择器

应用时，我们通常提取多个元素的公用属性，作为基本类

在其基础上，通过其它类扩展每个元素的个性属性

这是避免 CSS 代码冗余的常用方法

- SMACSS

可扩展模块化 CSS 架构，全称是 Scalable and Modular Architecture for CSS

定义了 CSS 分类和命名规则，并对 CSS 书写提出优化建议

- Base：默认，基础，通用规则，包括重置浏览器样式的规则

- Layout: 布局规则, 以l-或layout-开头
 - Module: 可复用模块规则
 - State: 布局或模块的特殊状态规则, 如隐藏, 激活等, 以is开头
 - Theme: 皮肤或主题规则, 可能包含另一种配色方案
- BEM

块元素修饰符, 全称是 Block Element Modifier

定义了一种 CSS 的命名规则, 用于解决命名冲突:

.block-nameelement-name--modifier-name

.块名元素名--修饰符 (元素名和修饰符都可为空)

其中:

 - Block: 块, 忽略结构和优先级, 具有独立意义的实体
 - Element: 元素, 块内部没有独立意义的实体
 - Modifier: 修饰符, 标识块或元素的外观、行为、状态被修改

含有修饰符的类名不可独立出现, 通常跟在不含修饰符的类名后
- ITCSS

倒三角形CSS, 全称是 Inverted Triangle CSS, 主要用SASS实现

提供了一种 CSS 由通用到具体的分层 (分类) 方法, 层次可以按需增删:

 - Settings: 全局变量、方法
 - Tools: 全局使用函数、混入器
 - Generic: 浏览器默认样式重置
 - Base: 仅可使用类型 (标签) 选择器
 - Objects: 遵循OOCSSS的对象的盒模型, 无颜色、背景等
 - Components: 可复用的组件
 - Trumps: 对组件的微调和其他样式定义, 可使用 !important
- Utility-First CSS

实用类优先CSS

提供了一种外观组件的构建方法, 以 tailwindcss 为代表

 - 按照一定规则, 基于 CSS 实用类构建复杂外观组件
 - 不用起类名
 - 样式文件体积不会随项目无限增长
 - 增删类名, 比直接修改属性更安全, 维护更容易
- ACSS

原子CSS, 全称是 Atomic CSS, 可以看成 Utility-First CSS 的极致抽象

提供了一种 CSS 类的定义方法, 提升大型项目 CSS 复用度

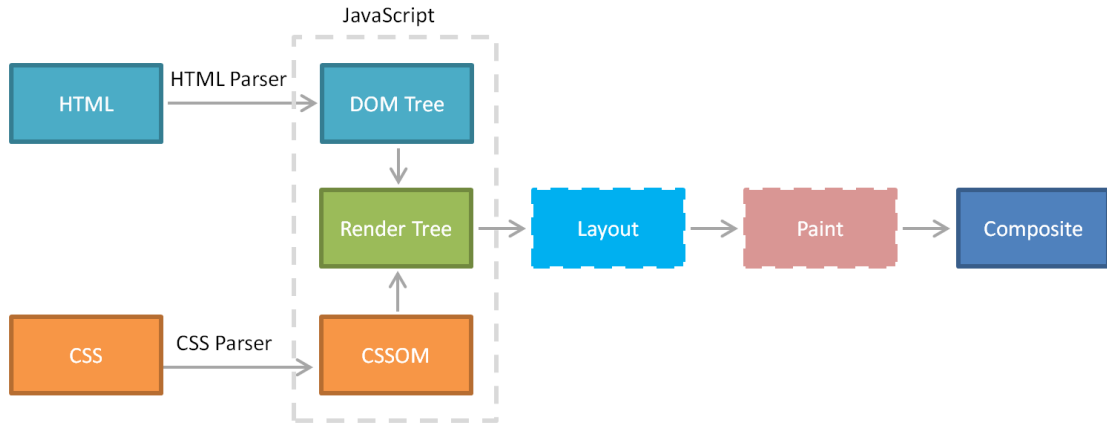
 - 每个类选择器中只包含一条属性定义, 一次编写, 到处运行
 - 属性名类似函数, 属性值类似函数参数, 像写内联样式一样写类名, 适合 CSS-in-JS
 - 提供工具, 可按需编译

CSS 属性的建议书写顺序, 为什么?

- 为什么要规定 CSS 属性的书写顺序?

以前, 我们规定 CSS 属性的书写顺序, 期望其浏览器浏览器渲染顺序, 减少不必要的重排和重绘

浏览器渲染流程如下：



- ① 解析 HTML 为 DOM Tree
- ② 解析 CSS 为 CSSOM
- ③ 深度遍历 Dom Tree，适配CSSOM。计算样式。将可见节点内容和计算后的样式放入 Render Tree
- ④ 布局 Render Tree
- ⑤ 绘制 Render Tree
- ⑥ 合成 Render Tree
- ⑦ 重排 Reflow
- ⑧ 重绘 RePaint

现代浏览器可以边下载边进行第 ① 到 ⑤ 渲染，开发者也可以手动实现DOM懒加载 对单个样式的修改，不会马上引起重排或重绘，而是会加入队列

当队列达到一定长度或一定时间后统一渲染，综上：

- ① 虽然 CSS 按书写顺序解析，但书写顺序对渲染的影响已经很小
- ② 有助于提升代码的可读性，可维护性，利于团队协作
- ③ 有效避免重复或遗漏声明属性
- ④ 避免实验中的私有属性与已加入规范的属性在具体实现的冲突
 - 合适的 CSS 属性书写顺序是？
根据浏览器的渲染顺序，我们总是将影响元素文档流占位的属性提前，参考 @mdu 的 CSS 编码规范：
 - ① 定位属性
 - ② 盒模型
 - ③ 排版属性
 - ④ 视觉属性
 - ⑤ 杂项属性

遇到私有属性和标准属性时，将私有属性放在前面

每个属性的具体顺序，可以参考推特或其他 CSS 规范

综合

HTTP2.0 时代，CSS Sprites 还有用吗，为什么？

HTTP1.1 下的 CSS Sprites

HTTP1.1 自身的管道化并不完善。浏览器实际通过为同一域名下的资源同时建立多个 TCP 连接，来实现并行传输，并且限制了最大连接数

通过手动或者工程化的方式，将小图片合并成大图，即 Sprites图（又称精灵图，雪碧图）。可以将多个对小图请求，合并为对单张大图请求，通过background-position定位在文档显示

优点：

- 避免大量小图请求阻塞其他资源下载
- 减少多连接的重复的从发起请求到首字节响应的时间
- 避免由于网络不稳定、客户端、服务端限制，导致部分小图丢失的情况
- 图标、界面一次显示全，提升用户体验

缺点：

- 修改、增加和删除图片、修改位置，颜色麻烦
- 请求图片的连接失败，整个界面图片效果丢失
- 宽度和高度通常固定，自适应需要额外设置
- 实现响应式图片时可能需要维护多张合成图，而且灵活度依然不够
- HTTP1.1 下的 SVG Sprites
- 合并 SVG 到一张 SVG

```
<svg>
  <symbol id="svg1"><!-- SVG① --></symbol>
  <symbol id="svg2"><!-- SVG② --></symbol>
  <symbol id="svg3"><!-- SVG③ --></symbol>
</svg>
```

- 在文档中通过use引用其中的一张 SVG

```
<svg>
  <use xlink:href="#svg1"></use>
</svg>
```

SVG 具有无锯齿放大，容易修改尺寸好颜色等优点

SVG Sprites 可以作为不考虑 IE8- 时，CSS Sprites替代

但开发者仍然需要 手工 或 工程化方式，合成 SVG Sprites

HTTP2.0 支持多路复用

HTTP2.0 下，一个 TCP 会话上可以进行任意数量的HTTP请求

理论上，所有图片资源几乎可以并行下载

CSS Sprites 最初要解决的问题，已经不复存在

基于 TCP 本身的局限性（丢包时，整个会话都要重传），考虑网络的不稳定

实际体验中，CSS Sprites 对比 HTTP2.0，通常仍会有轻微的速度提升

是否使用 CSS Sprites，或者升级到 SVG Sprites 可以结合维护复杂度代入具体项目决定

你在开发中都遇到过哪些 CSS 兼容性问题，你是如何解决的？

IE 兼容性问题

PC 时代，Trident 内核的 IE 在 5.5 - 8 时占据主导地位

- 问题：IE6 PNG 图片不透明
 - 答案：使用 progid:DXImageTransform.Microsoft.AlphaImageLoader 滤镜重复加载 PNG
- 问题：IE6 浮动元素设置 margin，边距变 2 倍
 - 答案：设置浮动元素display: inline

- 问题: IE6 不支持绝对定位 position: fixed

- 答案:

```
position: absolute;
top: expression(((t = document.documentElement.scrollTop) ? t :
document.body.scrollTop) + 'px');
```

- 问题: IE8 有链接的图片四周由黑边框

- 答案: `img { border: none }`

测试

- IETester 比 IE9+ 自带 Simulator 更接近真实效果
- 使用装有 低版本IE 的虚拟机镜像, 模拟完全真实的低版本 IE 环境
 - 微软甚至专门提供了一套90天激活的 win7 镜像
- 非标与标准浏览器兼容性问题
标准属性不被支持, 或者实现不同
- 问题: IE兼容模式 与 标准浏览器 宽度和高度表现不同
 - 答案: IE 兼容模式默认 box-sizing: border-box 而不是 box-sizing: content-box, 可以统一声明为 前者, 或者用 CSS hacks 设置值对 IE 兼容模式生效的宽高或边距
- 问题: IE9- 不支持 opacity
 - 答案:

```
opacity: .8;
filter: alpha(opacity = 80);
filter: progid:DXImageTransform.Microsoft.Alpha(style = 0, opacity = 80);
```

- 问题: IE9- 不支持 background-size 等 CSS3 新特性
 - 答案: IE6+ 支持behavior, 可以设置或检索对象的 DHTML 行为。引入 css3pie 其用这个属性让 IE 支持部分 CSS3 新特性
- 问题: cursor: hand在标准浏览器无效
 - 答案: 用 IE 也支持的标准属性值cursor:pointer替代

CSS3 新特性兼容性问题

CSS3 起, 标准被模块化, 每个模块相对独立。浏览器可能会将处于 非推荐 的阶段属性实验性实现, 或者某个处于 候选推荐 阶段的标准又被否决。导致不同浏览器, 以及同一浏览器的不同版本, 对于新特性的实现差异。

解决:

- 人工或使用混入器添加私有前缀
 - 工作量大, 开发阶段冗余代码多
 - 同一属性, 有无私有前缀, 后面接收的属性值类型、顺序等可能不同
 - 部分私有前缀, 需要加到属性值, 而不是属性名上
- PostCSS 的 autoprefixer 可以配置 Browserslist, 按需对浏览器及其不同版本添加私有前缀

移动端兼容问题

主要是 iOS 和 Android 下移动浏览器兼容以及使用 Webview 开发的 APP 兼容问题

注意: 以下这些问题只会在特定版本的系统下出现

- 问题: 溢出含滚动条元素在 iOS 滑动体验卡顿
 - 答案: 给溢出滚动元素设置 `-webkit-overflow-scrolling: touch`
- 问题: iOS 禁用按钮后, 按钮样式异常

- 答案：给按钮设置 opacity: 1
- 问题：iOS 监听 <label>点击无响应
 - 答案：给元素设置 cursor:pointer
- 问题：Android 输入框显示语音输入按钮
 - 答案：隐藏语音输入按钮 input::-webkit-input-speech-button { display: none }

想要让 APP 用户获得体验更一致，减少兼容性问题：

- 将渲染引擎与源码一起打包，比如使用腾讯基于 X5 内核的浏览服务，安装包会变大
- 采用 React Native Flutter UniApp + Weex 等使用 HTML + CSS + JS 开发，原生渲染的方案



当你忘记一个 CSS 属性时，你一般如何做？

- Emmet

Emmet 能够提高 HTML 和 CSS 的书写效率，我们只需键入缩写，代码提示就会帮助我们找到相应属性名或属性值

IDE 如 VsCode 集成了 Emmet，文本编辑器 Sublime Notepad++ VIM 等都有相应插件可以安装

df

 display: flex;	Emmet Abbreviation
 stroke-dashoffset	

- 查看说明

IDE 如 VsCode 支持鼠标悬停属性名 或 属性值，查看说明

点击其中的链接，可以直达 MDN 对应文档

In combination with 'float' and 'position', determines the type of box or boxes that are generated for an element.

Syntax: [<display-outside> || <display-inside>] | <display-listitem> | <display-internal> | <display-box> | <display-legacy>

[MDN Reference](#)

display: flex;
- 搜索

MDN，里面有属性的用法，在浏览器实现的说明及对应规范的链接

Can I use，里面有属性在各个运行环境的兼容性情况，移动端版本数据想丢丰富
- 记忆

重新手打一遍遗忘的属性和属性名，注释搜索中发现的信息或链接，在后续重构中思考最佳实践，这样能收获更多

结合项目谈谈你是如何学习 CSS 的？

- 发现问题

思考你在项目中遇到过哪些问题，你是否如何解决的

搜索，查文档、查规范是比请教他人更快速的问题解决途径

 - Bad：同一问题，同一属性，每次都要搜索
 - Good：高频的属性用法，尽量记住
 - Bad：反复复制粘贴代码
 - Good：找出代码中真正解决问题的部分，自己敲一遍
 - Great：明白问题的原因和解决问题的思路
- 提升效率、规范

思考项目有哪些可以改进的地方，你是如何改进的，成果怎样

- 效率：是否可以使用工具、脚本，将原有 CSS 及相关的图片编译、压缩、上传、部署 等流程化、自动化
- 规范：是否可以改进 CSS 文件、规则的组织形式，是否可以参考或者根据业务，拟定一套利于团队协作的开发规范，并通过工程化的方式推广规范（比如，在Commit前，对 CSS 代码进行规范性校验和自动更正）
- 技术分享
 - 回顾一次你做的技术分享，你通过哪些渠道获取了权威的信息，是否手写代码验证了这些信息，是否有通过 脑图、动画、PPT 等提高信息分享的效率 - 回顾一次你与同事或同学的争论，你是如何证明自己观点是正确的，是拿出了权威的资料，请权威的人帮你说服，还是通过实际测试后，用数据说话
- 学无止境
 - CSS 最近有哪些新鲜事，可以新的草案，新的工具，新的方法论等，简单分享你的观点
 - 项目中所用的技术栈，最近更新的内容是什么，比过去改进了什么，解决了什么问题，怎样解决的？

你可以根据上面的提纲自由发挥，每个问题举 1 个例子即可，避免长篇大论，不要谈太多感受，重点体现方法，结果，涉及的知识点到为止