# CSS

本文内容部分引用自 [front-end-interview-handbook](https://github.com/yangshun/front-end-interview-handbook/) ### CSS 选择器的优先级是如何计算的？

浏览器通过优先级规则，判断元素展示哪些样式。优先级通过 4 个维度指标确定，我们假定以a、b、c、d命名，分别代表以下含义：

1. a表示是否使用内联样式（inline style）。如果使用，a为 1，否则为 0。
2. b表示 ID 选择器的数量。
3. c表示类选择器、属性选择器和伪类选择器数量之和。
4. d表示标签（类型）选择器和伪元素选择器之和。

优先级的结果并非通过以上四个值生成一个得分，而是每个值分开比较。a、b、c、d权重从左到右，依次减小。判断优先级时，从左到右，一一比较，直到比较出最大值，即可停止。所以，如果b的值不同，那么c和d不管多大，都不会对结果产生影响。比如0，1，0，0的优先级高于0，0，10，10。

当出现优先级相等的情况时，最晚出现的样式规则会被采纳。如果你在样式表里写了相同的规则（无论是在该文件内部还是其它样式文件中），那么最后出现的（在文件底部的）样式优先级更高，因此会被采纳。

在写样式时，我会使用较低的优先级，这样这些样式可以轻易地覆盖掉。尤其对写 UI 组件的时候更为重要，这样使用者就不需要通过非常复杂的优先级规则或使用!important的方式，去覆盖组件的样式了。

参考资料：

* https://www.smashingmagazine.com/2007/07/css-specificity-things-you-should-know/
* https://www.sitepoint.com/web-foundations/specificity/

### 重置（resetting）CSS 和 标准化（normalizing）CSS 的区别是什么？你会选择哪种方式，为什么？

* **重置（Resetting）**： 重置意味着除去所有的浏览器默认样式。对于页面所有的元素，像margin、padding、font-size这些样式全部置成一样。你将必须重新定义各种元素的样式。
* **标准化（Normalizing）**： 标准化没有去掉所有的默认样式，而是保留了有用的一部分，同时还纠正了一些常见错误。

当需要实现非常个性化的网页设计时，我会选择重置的方式，因为我要写很多自定义的样式以满足设计需求，这时候就不再需要标准化的默认样式了。

参考资料：

* https://stackoverflow.com/questions/6887336/what-is-the-difference-between-normalize-css-and-reset-css

### 请阐述Float定位的工作原理。

浮动（float）是 CSS 定位属性。浮动元素从网页的正常流动中移出，但是保持了部分的流动性，会影响其他元素的定位（所有在浮动下面的自身不浮动的内容都将围绕浮动元素进行包装）。这一点与绝对定位不同，绝对定位的元素完全从文档流中脱离。

有一种简单的方法可以解决这个问题——clear 属性。当你把这个应用到一个元素上时，它主要意味着"此处停止浮动"——这个元素和源码中后面的元素将不浮动，除非您稍后将一个新的float声明应用到此后的另一个元素。

div {  
 clear: both;  
}

clear 可以取三个值： \* left：停止任何活动的左浮动 \* right：停止任何活动的右浮动 \* both：停止任何活动的左右浮动

参考资料： \* [清除浮动](https://developer.mozilla.org/zh-CN/docs/Learn/CSS/CSS_layout/Floats#%E6%B8%85%E9%99%A4%E6%B5%AE%E5%8A%A8)

### 请阐述z-index属性，并说明如何形成层叠上下文（stacking context）。

CSS 中的z-index属性控制重叠元素的垂直叠加顺序。z-index只能影响position值不是static的元素。

没有定义z-index的值时，元素按照它们出现在 DOM 中的顺序堆叠（层级越低，出现位置越靠上）。非静态定位的元素（及其子元素）将始终覆盖静态定位（static）的元素，而不管 HTML 层次结构如何。

层叠上下文是包含一组图层的元素。 在一组层叠上下文中，其子元素的z-index值是相对于该父元素而不是 document root 设置的。每个层叠上下文完全独立于它的兄弟元素。如果元素 B 位于元素 A 之上，则即使元素 A 的子元素 C 具有比元素 B 更高的z-index值，元素 C 也永远不会在元素 B 之上.

每个层叠上下文是自包含的：当元素的内容发生层叠后，整个该元素将会在父层叠上下文中按顺序进行层叠。少数 CSS 属性会触发一个新的层叠上下文，例如opacity小于 1，filter不是none，transform不是none。

参考资料：

* https://css-tricks.com/almanac/properties/z/z-index/
* https://philipwalton.com/articles/what-no-one-told-you-about-z-index/
* https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\_Positioning/Understanding\_z\_index/The\_stacking\_context

### 请阐述块格式化上下文（Block Formatting Context）及其工作原理。

块格式上下文（BFC）是 Web 页面的可视化 CSS 渲染的部分，是块级盒布局发生的区域，也是浮动元素与其他元素交互的区域。

一个 HTML 盒（Box）满足以下任意一条，会创建块格式化上下文：

* float的值不是none.
* position的值不是static或relative.
* display的值是table-cell、table-caption、inline-block、flex、或inline-flex。
* overflow的值不是visible。

在 BFC 中，每个盒的左外边缘都与其包含的块的左边缘相接。

两个相邻的块级盒在垂直方向上的边距会发生合并（collapse）。更多内容请参考[边距合并（margin collapsing）](https://www.sitepoint.com/web-foundations/collapsing-margins/)。

浮动定位和清除浮动时只会应用于同一个BFC内的元素。浮动不会影响其它BFC中元素的布局，而清除浮动只能清除同一BFC中在它前面的元素的浮动。

外边距折叠（Margin collapsing）也只会发生在属于同一BFC的块级元素之间。

参考资料：

* https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Block\_formatting\_context
* https://www.sitepoint.com/understanding-block-formatting-contexts-in-css/

### 请解释什么是雪碧图（css sprites），以及如何实现？

雪碧图是把多张图片整合到一张上的图片。它被运用在众多使用了很多小图标的网站上（Gmail 在使用）。实现方法：

1. 使用生成器将多张图片打包成一张雪碧图，并为其生成合适的 CSS。
2. 每张图片都有相应的 CSS 类，该类定义了background-image、background-position和background-size属性。
3. 使用图片时，将相应的类添加到你的元素中。

好处：

* 减少加载多张图片的 HTTP 请求数（一张雪碧图只需要一个请求）。但是对于 HTTP2 而言，加载多张图片不再是问题。
* 提前加载资源，防止在需要时才在开始下载引发的问题，比如只出现在:hover伪类中的图片，不会出现闪烁。

参考资料：

* https://css-tricks.com/css-sprites/

### 如何解决不同浏览器的样式兼容性问题？

* 在确定问题原因和有问题的浏览器后，使用单独的样式表，仅供出现问题的浏览器加载。这种方法需要使用服务器端渲染。
* 使用已经处理好此类问题的库，比如 Bootstrap。
* 使用 autoprefixer 自动生成 CSS 属性前缀。
* 使用 Reset CSS 或 Normalize.css。

### 如何为功能受限的浏览器提供页面？ 使用什么样的技术和流程？

* 优雅的降级：为现代浏览器构建应用，同时确保它在旧版浏览器中正常运行。
* Progressive enhancement - The practice of building an application for a base level of user experience, but adding functional enhancements when a browser supports it.
* 渐进式增强：构建基于用户体验的应用，但在浏览器支持时添加新增功能。
* 利用 [caniuse.com](https://caniuse.com/) 检查特性支持。
* 使用 autoprefixer 自动生成 CSS 属性前缀。
* 使用 [Modernizr](https://modernizr.com/)进行特性检测。

### 有什么不同的方式可以隐藏内容（使其仅适用于屏幕阅读器）？

这些方法与可访问性（a11y）有关。

* visibility: hidden：元素仍然在页面流中，并占用空间。
* width: 0; height: 0：使元素不占用屏幕上的任何空间，导致不显示它。
* position: absolute; left: -99999px： 将它置于屏幕之外。
* text-indent: -9999px：这只适用于block元素中的文本。
* Metadata： 例如通过使用 Schema.org，RDF 和 JSON-LD。
* WAI-ARIA：如何增加网页可访问性的 W3C 技术规范。

即使 WAI-ARIA 是理想的解决方案，我也会采用绝对定位方法，因为它具有最少的注意事项，适用于大多数元素，而且使用起来非常简单。

参考资料：

* https://www.w3.org/TR/wai-aria-1.1/
* https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA
* http://a11yproject.com/

### 除了screen，你还能说出一个 @media 属性的例子吗？

* all 适用于所有设备。
* print 为了加载合适的文档到当前使用的可视窗口. 需要提前咨询 paged media（媒体屏幕尺寸）, 以满足个别设备网页尺寸不匹配等问题。
* screen 主要适用于彩色的电脑屏幕
* speech 解析speech这个合成器. 注意: CSS2已经有一个相似的媒体类型叫aural. https://developer.mozilla.org/zh-CN/docs/Web/CSS/@media

### 编写高效的 CSS 应该注意什么？

首先，浏览器从最右边的选择器，即关键选择器（key selector），向左依次匹配。根据关键选择器，浏览器从 DOM 中筛选出元素，然后向上遍历被选元素的父元素，判断是否匹配。选择器匹配语句链越短，浏览器的匹配速度越快。避免使用标签和通用选择器作为关键选择器，因为它们会匹配大量的元素，浏览器必须要进行大量的工作，去判断这些元素的父元素们是否匹配。

[BEM (Block Element Modifier)](https://bem.info/) methodology recommends that everything has a single class, and, where you need hierarchy, that gets baked into the name of the class as well, this naturally makes the selector efficient and easy to override. [BEM (Block Element Modifier)](https://bem.info/)原则上建议为独立的 CSS 类命名，并且在需要层级关系时，将关系也体现在命名中，这自然会使选择器高效且易于覆盖。

搞清楚哪些 CSS 属性会触发重新布局（reflow）、重绘（repaint）和合成（compositing）。在写样式时，避免触发重新布局的可能。

参考资料：

* https://developers.google.com/web/fundamentals/performance/rendering/
* https://csstriggers.com/

### 使用 CSS 预处理的优缺点分别是什么？

优点：

* 提高 CSS 可维护性。
* 易于编写嵌套选择器。
* 引入变量，增添主题功能。可以在不同的项目中共享主题文件。
* 通过混合（Mixins）生成重复的 CSS。
* Splitting your code into multiple files. CSS files can be split up too but doing so will require a HTTP request to download each CSS file.
* 将代码分割成多个文件。不进行预处理的 CSS，虽然也可以分割成多个文件，但需要建立多个 HTTP 请求加载这些文件。

缺点：

* 需要预处理工具。
* 重新编译的时间可能会很慢。

### 对于你使用过的 CSS 预处理，说说喜欢和不喜欢的地方？

喜欢：

* 绝大部分优点上题以及提过。
* Less 用 JavaScript 实现，与 NodeJS 高度结合。

**Dislikes:**

* 我通过node-sass使用 Sass，它用 C ++ 编写的 LibSass 绑定。在 Node 版本切换时，我必须经常重新编译。
* Less 中，变量名称以@作为前缀，容易与 CSS 关键字混淆，如@media、@import和@font-face。

### 如何实现一个使用非标准字体的网页设计？

使用@font-face并为不同的font-weight定义font-family。

### 解释浏览器如何确定哪些元素与 CSS 选择器匹配。

这部分与上面关于编写高效的 CSS 有关。浏览器从最右边的选择器（关键选择器）根据关键选择器，浏览器从 DOM 中筛选出元素，然后向上遍历被选元素的父元素，判断是否匹配。选择器匹配语句链越短，浏览器的匹配速度越快。

例如，对于形如 p span 的选择器，浏览器首先找到所有 <span> 元素，并遍历它的父元素直到根元素以找到 <p> 元素。对于特定的 <span>，只要找到一个<p>，就知道 <span> 已经匹配并停止继续匹配。

参考资料：

* https://stackoverflow.com/questions/5797014/why-do-browsers-match-css-selectors-from-right-to-left

### 描述伪元素及其用途。

CSS 伪元素是添加到选择器的关键字，去选择元素的特定部分。它们可以用于装饰（:first-line，:first-letter）或将元素添加到标记中（与 content:...组合），而不必修改标记（:before，:after）。

* :first-line和:first-letter可以用来修饰文字。
* 上面提到的.clearfix方法中，使用clear: both来添加不占空间的元素。
* 使用:before和after展示提示中的三角箭头。鼓励关注点分离，因为三角被视为样式的一部分，而不是真正的 DOM。如果不使用额外的 HTML 元素，只用 CSS 样式绘制三角形是不太可能的。

参考资料：

* https://css-tricks.com/almanac/selectors/a/after-and-before/

### 说说你对盒模型的理解，以及如何告知浏览器使用不同的盒模型渲染布局。

CSS 盒模型描述了以文档树中的元素而生成的矩形框，并根据排版模式进行布局。每个盒子都有一个内容区域（例如文本，图像等）以及周围可选的padding、border和margin区域。

CSS 盒模型负责计算：

* 块级元素占用多少空间。
* 边框是否重叠，边距是否合并。
* 盒子的尺寸。

盒模型有以下规则：

* 块级元素的大小由width、height、padding、border和margin决定。
* 如果没有指定height，则块级元素的高度等于其包含子元素的内容高度加上padding（除非有浮动元素，请参阅下文）。
* 如果没有指定width，则非浮动块级元素的宽度等于其父元素的宽度减去父元素的padding。
* 元素的height是由内容的height来计算的。
* 元素的width是由内容的width来计算的。
* 默认情况下，padding和border不是元素width和height的组成部分。

参考资料：

* https://www.smashingmagazine.com/2010/06/the-principles-of-cross-browser-css-coding/#understand-the-css-box-model

### \* { box-sizing: border-box; }会产生怎样的效果？

* 元素默认应用了box-sizing: content-box，元素的宽高只会决定内容（content）的大小。
* box-sizing: border-box改变计算元素width和height的方式，border和padding的大小也将计算在内。
* 元素的height = 内容（content）的高度 + 垂直方向的padding + 垂直方向border的宽度
* 元素的width = 内容（content）的宽度 + 水平方向的padding + 水平方向border的宽度

### display的属性值都有哪些？

* none, block, inline, inline-block, table, table-row, table-cell, list-item.

### inline和inline-block有什么区别？

我把block也加入其中，为了获得更好的比较。

|  |  |  |  |
| --- | --- | --- | --- |
|  | block | inline-block | inline |
| 大小 | 填充其父容器的宽度。 | 取决于内容。 | 取决于内容。 |
| 定位 | 从新的一行开始，并且不允许旁边有 HTML 元素（除非是float） | 与其他内容一起流动，并允许旁边有其他元素。 | 与其他内容一起流动，并允许旁边有其他元素。 |
| 能否设置width和height | 能 | 能 | 不能。 设置会被忽略。 |
| 可以使用vertical-align对齐 | 不可以 | 可以 | 可以 |
| 边距（margin）和填充（padding） | 各个方向都存在 | 各个方向都存在 | 只有水平方向存在。垂直方向会被忽略。 尽管border和padding在content周围，但垂直方向上的空间取决于'line-height' |
| 浮动（float） | - | - | 就像一个block元素，可以设置垂直边距和填充。 |

### relative、fixed、absolute和static四种定位有什么区别？

经过定位的元素，其position属性值必然是relative、absolute、fixed或sticky。

* static：默认定位属性值。该关键字指定元素使用正常的布局行为，即元素在文档常规流中当前的布局位置。此时 top, right, bottom, left 和 z-index 属性无效。
* relative：该关键字下，元素先放置在未添加定位时的位置，再在不改变页面布局的前提下调整元素位置（因此会在此元素未添加定位时所在位置留下空白）。
* absolute：不为元素预留空间，通过指定元素相对于最近的非 static 定位祖先元素的偏移，来确定元素位置。绝对定位的元素可以设置外边距（margins），且不会与其他边距合并。
* fixed：不为元素预留空间，而是通过指定元素相对于屏幕视口（viewport）的位置来指定元素位置。元素的位置在屏幕滚动时不会改变。打印时，元素会出现在的每页的固定位置。fixed 属性会创建新的层叠上下文。当元素祖先的 transform 属性非 none 时，容器由视口改为该祖先。
* sticky：盒位置根据正常流计算(这称为正常流动中的位置)，然后相对于该元素在流中的 flow root（BFC）和 containing block（最近的块级祖先元素）定位。在所有情况下（即便被定位元素为 table 时），该元素定位均不对后续元素造成影响。当元素 B 被粘性定位时，后续元素的位置仍按照 B 未定位时的位置来确定。position: sticky 对 table 元素的效果与 position: relative 相同。

参考资料：

* https://developer.mozilla.org/en/docs/Web/CSS/position

### 你使用过哪些现有的 CSS 框架？你是如何改进它们的？

* **Bootstrap**： 更新周期缓慢。Bootstrap 4 已经处于 alpha 版本将近两年了。添加了在页面中广泛使用的微调按钮组件。
* **Semantic UI**：源代码结构使得自定义主题很难理解。非常规主题系统的使用体验很差。外部库的路径需要硬编码（hard code）配置。变量重新赋值没有 Bootstrap 设计得好。
* **Bulma**： 需要很多非语义的类和标记，显得很多余。不向后兼容，以至于升级版本后，会破坏应用的正常运行。

### 你了解 CSS Flex 和 Grid 吗？

Flex 主要用于一维布局，而 Grid 则用于二维布局。 ### Flex 使用了 display: flex 的元素称为容器(flex container)。flex 容器中存在两条轴，水平方向的主轴和垂直方向的交叉轴。容器中的每个单元称为flex item（项目）。

在 flex 中有两种类型的属性：容器属性和项目属性。顾名思义，容器属性是在容器上配置的，项目属性则是针对项目的。

在容器上可以设置6个属性： \* flex-direction（设置主轴方向） \* flex-wrap（主轴换行方式） \* flex-flow（flex-flow 属性是 flex-direction 属性和 flex-wrap 属性的简写形式，默认值为row nowrap） \* justify-content（项目在主轴上的对齐方式） \* align-items（项目在交叉轴上的对齐方式） \* align-content（多根轴线的对齐方式。如果项目只有一根轴线，该属性不起作用）

**注意**：设置 flex 布局之后，子元素的 float、clear、vertical-align 的属性将会失效。

#### Flex 项目属性

有六种属性可运用在 item 项目上: \* order（项目的排列顺序。数值越小，排列越靠前，默认为0） \* flex-grow（项目的放大比例，默认为0，即如果存在剩余空间，也不放大） \* flex-shrink（项目的缩小比例，默认为1，即如果空间不足，该项目将缩小） \* flex-basis（定义了在分配多余空间之前，项目占据的主轴空间。浏览器根据这个属性，计算主轴是否有多余空间。它的默认值为auto，即项目的本来大小。） \* flex（flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选。） \* align-self（align-self属性允许单个项目有与其他项目不一样的对齐方式，可覆盖align-items属性。默认值为auto，表示继承父元素的align-items属性，如果没有父元素，则等同于stretch。）

### Grid

CSS网格布局用于将页面分割成数个主要区域，或者用来定义组件内部元素间大小、位置和图层之间的关系。

像表格一样，网格布局让我们能够按行或列来对齐元素。 但是，使用CSS网格可能还是比CSS表格更容易布局。 例如，网格容器的子元素可以自己定位，以便它们像CSS定位的元素一样，真正的有重叠和层次。

参考资料： \* [Flex 布局教程：语法篇](http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html) \* [Flex 布局教程：实例篇](https://www.ruanyifeng.com/blog/2015/07/flex-examples.html)

### 响应式设计与自适应设计有何不同？

响应式设计和自适应设计都以提高不同设备间的用户体验为目标，根据视窗大小、分辨率、使用环境和控制方式等参数进行优化调整。

响应式设计的适应性原则：网站应该凭借一份代码，在各种设备上都有良好的显示和使用效果。响应式网站通过使用媒体查询，自适应栅格和响应式图片，基于多种因素进行变化，创造出优良的用户体验。就像一个球通过膨胀和收缩，来适应不同大小的篮圈。

自适应设计更像是渐进式增强的现代解释。与响应式设计单一地去适配不同，自适应设计通过检测设备和其他特征，从早已定义好的一系列视窗大小和其他特性中，选出最恰当的功能和布局。与使用一个球去穿过各种的篮筐不同，自适应设计允许使用多个球，然后根据不同的篮筐大小，去选择最合适的一个。

参考资料：

* https://developer.mozilla.org/en-US/docs/Archive/Apps/Design/UI\_layout\_basics/Responsive\_design\_versus\_adaptive\_design
* http://mediumwell.com/responsive-adaptive-mobile/
* https://css-tricks.com/the-difference-between-responsive-and-adaptive-design/

### 你有没有使用过视网膜分辨率的图形？当中使用什么技术？

我倾向于使用更高分辨率的图形（显示尺寸的两倍）来处理视网膜显示。更好的方法是使用媒体查询，像@media only screen and (min-device-pixel-ratio: 2) { ... }，然后改变background-image。

对于图标类的图形，我会尽可能使用 svg 和图标字体，因为它们在任何分辨率下，都能被渲染得十分清晰。

还有一种方法是，在检查了window.devicePixelRatio的值后，利用 JavaScript 将<img>的src属性修改，用更高分辨率的版本进行替换。

参考资料：

* https://www.sitepoint.com/css-techniques-for-retina-displays/

### 什么情况下，用translate()而不用绝对定位？什么时候，情况相反。

translate()是transform的一个值。改变transform或opacity不会触发浏览器重新布局（reflow）或重绘（repaint），只会触发复合（compositions）。而改变绝对定位会触发重新布局，进而触发重绘和复合。transform使浏览器为元素创建一个 GPU 图层，但改变绝对定位会使用到 CPU。 因此translate()更高效，可以缩短平滑动画的绘制时间。

当使用translate()时，元素仍然占据其原始空间（有点像position：relative），这与改变绝对定位不同。

参考资料：

* https://www.paulirish.com/2012/why-moving-elements-with-translate-is-better-than-posabs-topleft/

### 其他答案

* https://neal.codes/blog/front-end-interview-css-questions
* https://quizlet.com/28293152/front-end-interview-questions-css-flash-cards/
* http://peterdoes.it/2015/12/03/a-personal-exercise-front-end-job-interview-questions-and-my-answers-all/

### 一边固定宽度一边宽度自适应

可以使用flex布局 复制下面的HTML和CSS代码 用浏览器打开可以看到效果

<div class="wrap">  
 <div class="div1"></div>  
 <div class="div2"></div>  
</div>

.wrap {  
 display: flex;  
 justify-content: space-between;  
}  
.div1 {  
 min-width: 200px;  
}  
.div2 {  
 width: 100%;  
 background: #e6e6e6;  
}  
html,  
body,  
div {  
 height: 100%;  
 margin: 0;  
}

### 水平垂直居中的方式

#### flex

display: flex;  
justify-content: center;  
align-items: center;

#### position

/\* 父容器 \*/  
position: relative;  
  
/\* 子容器 \*/  
position:absolute;  
margin:auto;  
top:0;  
bottom:0;  
left:0;  
right:0;

#### position+transform

/\* 父容器 \*/  
position: relative;  
  
/\* 子容器 \*/  
position: absolute;  
top: 50%;  
left: 50%;  
transform: translate(-50%, -50%);

#### table-cell

<div class="box">  
 <div class="content">  
 <div class="inner"></div>  
 </div>  
</div>  
  
html, body {  
 height: 100%;  
 width: 100%;  
 margin: 0;  
}  
.box {  
 display: table;  
 height: 100%;  
 width: 100%;  
}  
.content {  
 display: table-cell;  
 vertical-align: middle;  
 text-align: center;  
}  
.inner {  
 background-color: #000;  
 display: inline-block;  
 width: 200px;  
 height: 200px;  
}

### display:none、visibile:hidden、opacity:0的区别

|  |  |  |  |
| --- | --- | --- | --- |
|  | 是否隐藏 | 是否在文档中占用空间 | 是否会触发事件 |
| display: none | 是 | 否 | 否 |
| visibile: hidden | 是 | 是 | 否 |
| opacity: 0 | 是 | 是 | 是 |

### CSS中link和@import的区别

* link属于HTML标签，而@import是CSS提供的
* 页面被加载的时，link会同时被加载，而@import引用的CSS会等到页面被加载完再加载
* import只在IE5以上才能识别，而link是HTML标签，无兼容问题
* link方式的样式的权重 高于@import的权重

### 如何用css实现瀑布流布局

利用column-count和break-inside这两个CSS3属性即可，复制如下代码即可察看效果

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <style>  
 body {  
 margin: 0;  
 }  
 .waterfall-container {  
 /\*分几列\*/  
 column-count: 2;  
 width: 100%;  
 /\* 列间距 \*/  
 column-gap: 10px;  
 }  
  
 .waterfall-item {  
 break-inside: avoid;  
 width: 100%;  
 height: 100px;  
 margin-bottom: 10px;  
 background: #ddd;  
 column-gap: 0;  
 text-align: center;  
 color: #fff;  
 font-size: 40px;  
 }  
 </style>  
</head>  
<body>  
 <div class="waterfall-container">  
 <div class="waterfall-item" style="height: 100px">1</div>  
 <div class="waterfall-item" style="height: 300px">2</div>  
 <div class="waterfall-item" style="height: 400px">3</div>  
 <div class="waterfall-item" style="height: 100px">4</div>  
 <div class="waterfall-item" style="height: 300px">5</div>  
 <div class="waterfall-item" style="height: 600px">6</div>  
 <div class="waterfall-item" style="height: 400px">7</div>  
 <div class="waterfall-item" style="height: 300px">8</div>  
 <div class="waterfall-item" style="height: 700px">9</div>  
 <div class="waterfall-item" style="height: 100px">10</div>  
 </div>  
</body>  
</html>

## 文本超出部分显示省略号

#### 单行

overflow: hidden;  
text-overflow: ellipsis;  
white-space: nowrap;

#### 多行

display: -webkit-box;  
-webkit-box-orient: vertical;  
-webkit-line-clamp: 3; // 最多显示几行  
overflow: hidden;

## 利用伪元素画三角

.info-tab {  
 position: relative;  
}  
.info-tab::after {  
 content: '';  
 border: 4px solid transparent;  
 border-top-color: #2c8ac2;  
 position: absolute;  
 top: 0;  
}

## 已知父级盒子的宽高，子级img宽高未知，想让img铺满父级盒子且图片不能变形

需要用到css的object-fit属性

div {  
 width: 200px;  
 height: 200px;  
}  
img {  
 object-fit: cover;  
 width: 100%;  
 height: 100%;  
}

[MDN](https://developer.mozilla.org/zh-CN/docs/Web/CSS/object-fit)

## iframe的作用

iframe是用来在网页中插入第三方页面，早期的页面使用iframe主要是用于导航栏这种很多页面都相同的部分，这样在切换页面的时候避免重复下载。

优点 1. 便于修改，模拟分离，像一些信息管理系统会用到。 2. 但现在基本不推荐使用。除非特殊需要，一般不推荐使用。

缺点 1. iframe的创建比一般的DOM元素慢了1-2个数量级 2. iframe标签会阻塞页面的的加载，如果页面的onload事件不能及时触发，会让用户觉得网页加载很慢，用户体验不好，在Safari和Chrome中可以通过js动态设置iframe的src属性来避免阻塞。 3. iframe对于SEO不友好，替换方案一般就是动态语言的Incude机制和ajax动态填充内容等。

## css hack是什么

由于不同的浏览器，比如Internet Explorer 6,Internet Explorer 7,Mozilla Firefox等，对CSS的解析认识不一样，因此会导致生成的页面效果不一样，得不到我们所需要的页面效果。

这个时候我们就需要针对不同的浏览器去写不同的CSS，让它能够同时兼容不同的浏览器，能在不同的浏览器中也能得到我们想要的页面效果。

这个针对不同的浏览器写不同的CSS code的过程，就叫CSS hack,也叫写CSS hack。

具体请看： http://www.cnblogs.com/Renyi-Fan/p/9006084.html

## 过渡与动画的区别是什么

* transition 可以在一定的时间内实现元素的状态过渡为最终状态，用于模拟以一种过渡动画效果，但是功能有限，只能用于制作简单的动画效果而动画属性
* animation 可以制作类似Flash动画，通过关键帧控制动画的每一步，控制更为精确，从而可以制作更为复杂的动画。

## 什么是外边距合并

外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。

合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

## 去除inline-block元素间间距的方法

* 移除空格
* 使用margin负值
* 使用font-size:0
* letter-spacing
* word-spacing

更详细的介绍请看:[去除inline-block元素间间距的N种方法](https://www.zhangxinxu.com/wordpress/2012/04/inline-block-space-remove-%E5%8E%BB%E9%99%A4%E9%97%B4%E8%B7%9D/)

## 为什么要初始化CSS样式

* 因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对 CSS 初始化往往会出现浏览器之间的页面显示差异。
* 去掉标签的默认样式如：margin,padding，其他浏览器默认解析字体大小，字体设置。

## 利用html css 编写样式，div垂直body居中、div内的text垂直居中，div高度等于body宽度的一半

padding-top/bottom 和 margin-top/bottom 都是相对于父元素的宽度来计算的。

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>demo</title>  
 <style>  
 body {  
 height: 100vh;  
 }  
 body,  
 div {  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 }  
 div {  
 width: 100%;  
 height: 0;  
 padding-bottom: 50%;  
 position: relative;  
 background: #ddd;  
 }  
 .text {  
 position: absolute;  
 top: 50%;  
 left: 50%;  
 transform: translate(-50%,-50%);  
 }  
 </style>  
</head>  
<body>  
 <div>  
 <p class="text">  
 这里是文本信息  
 </p>  
 </div>  
</body>  
</html>

## 行内格式化上下文 IFC

行内格式化上下文是一个网页的渲染结果的一部分。其中，各行内框（inline boxes）一个接一个地排列，其排列顺序根据书写模式（writing-mode）的设置来决定：

* 对于水平书写模式，各个框从左边开始水平地排列
* 对于垂直书写模式，各个框从顶部开始水平地排列

## margin 塌陷

父子嵌套元素在垂直方向的 margin, 父子元素是结合在一起的, 他们两个的 margin 会取其中最大的值。

### 解决方法——触发 BFC

只要元素满足下面任一条件即可触发 BFC 特性：

* body 根元素
* 浮动元素：float 除 none 以外的值
* 绝对定位元素：position (absolute、fixed)
* display 为 inline-block、table-cells、flex
* overflow 除了 visible 以外的值 (hidden、auto、scroll)

参考资料： \* [margin合并和塌陷的问题](https://www.jianshu.com/p/3b499982bcb0) \* [10 分钟理解 BFC 原理](https://zhuanlan.zhihu.com/p/25321647)

## css 动画优化 GPU 合成层

[一篇文章说清浏览器解析和CSS（GPU）动画优化](https://segmentfault.com/a/1190000008015671)

## css 作用域隔离方法

* 命名空间，加不同的前缀
* module，例如 vue 的 scoped
* css-in-js，直接写成内联样式
* Shadow DOM，其实就是 web components，作用域隔离