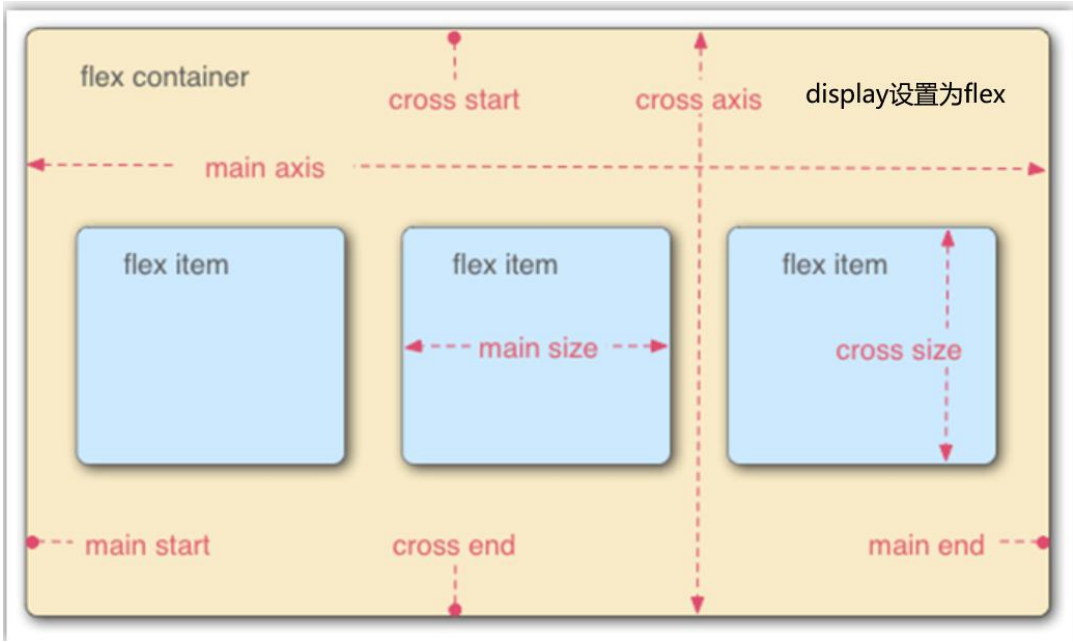
Flex 布局是什么？

设置 display：flex，可以实现响应式开发



以下 6 个属性设置在容器上。

flex-direction
flex-wrap
flex-flow
justify-content
align-items
align-content

1.flex-direction 属性

flex-direction 属性决定主轴的方向（即项目的排列方向）。

```
.box {
    flex-direction: row | row-reverse | column | column-reverse;}
```

row（默认值）：主轴为水平方向，起点在左端。
row-reverse：主轴为水平方向，起点在右端。
column：主轴为垂直方向，起点在上沿,自上而下。
column-reverse：主轴为垂直方向，起点在下沿,自下而上。

2.flex-wrap 属性：决定是否换行

默认情况下，项目都排在一条线（又称"轴线"）上。flex-wrap 属性定义，如果一条轴线排不下，应该如何换行。

.box{
  flex-wrap: nowrap | wrap | wrap-reverse;}
nowrap（默认）：不换行,宽度自动压缩。
wrap：换行，第一行在上方。
wrap-reverse：换行，第一行在下方。

3.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。

.box {
  flex-flow: <flex-direction> || <flex-wrap>;}.box{
  flex-flow:row||nowrap;}

4.justify-content 属性

justify-content 属性定义了项目在主轴上的对齐方式。

flex-start（默认值）：左对齐
flex-end：右对齐
center： 居中
space-between：两端对齐，组件之间的间隔都相等。
space-around：距边界两侧的间隔相等，元素之间的间隔比项目与边框的间隔大一倍。

5.align-items 属性

align-items 属性定义项目在交叉轴上(即纵向,垂直)如何对齐。

.box {
  align-items: flex-start | flex-end | center | baseline | stretch;}
flex-start：交叉轴的起点(顶部)对齐。
flex-end：交叉轴的终点(底部)对齐。
center：交叉轴的中点(中间)对齐。baseline：项目的第一行文字的基线(即根据内容对齐,不再根据容器)对齐。
stretch（默认值）：如果项目未设置高度或设为 auto，将占满整个容器的高度。

6.align-content 属性(说的是垂直方向的交叉轴)

align-content 属性定义了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。

.box {
  align-content: flex-start | flex-end | center | space-between | space-around | stretch;}
flex-start：与交叉轴的起点对齐。
flex-end：与交叉轴的终点对齐。
center：与交叉轴的中点对齐。
space-between：与交叉轴两端对齐，轴线之间的间隔平均分布。
space-around：每根轴线两侧的间隔都相等。所以，轴线之间的间隔比轴线与边框的间隔大一倍。
stretch（默认值）：轴线占满整个交叉轴

flex 为 1 调用了哪些属性

flex-grow：1
flex-shrink：1
flex-basis：0
  a. 第一个参数表示: flex-grow 定义项目的放大比例，默认为 0，即如果存在剩余空间，也不放大
  b. 第二个参数表示: flex-shrink 定义了项目的缩小比例，默认为 1，即如果空间不足，该项目将缩小
  c. 第三个参数表示: flex-basis（设置宽度，会覆盖 width）给之前两个属性分配多余空间之前，计算项目是否有多余空间，默认值为 auto，即项目本身的大小

以下 6 个属性设置在项目上。

order
flex-grow
flex-shrink
flex-basis
flex
align-self

1. order 属性

order 属性 flex item（子元素）进行排序定义项目的排列顺序。数值越小，排列越靠前，默认为 0。
.item {
  order: <integer>;}

2.flex-grow 属性

flex-grow 属性定义项目的放大比例，可以让子元素撑满父元素，默认为 0，即如果存在剩余空间，也不放大。

```
.item {
  flex-grow: <number>; /* default 0 */}
```

如果所有项目的 flex-grow 属性都为 1，则它们将等分剩余空间（如果有的话）。

如果一个项目的 flex-grow 属性为 2，其他项目都为 1，则前者占据的剩余空间将比其他项多一倍。

3.flex-shrink 属性

flex-shrink 属性定义了项目的缩小比例，默认为 1，即如果空间不足，该项目将缩小。

```
.item {
  flex-shrink: <number>; /* default 1 */}
```

如果所有项目的 flex-shrink 属性都为 1，当空间不足时，都将等比例缩小。

如果一个项目的 flex-shrink 属性为 0，其他项目都为 1，则空间不足时，为 0 的不缩小。

注:负值对该属性无效。

4.flex-basis 属性

flex-basis 属性可以覆盖 width 定义了在分配多余空间之前，项目占据的主轴空间(main size)。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。

```
.item {
    flex-basis: <length> | auto; /* default auto */}
```

它可以设为跟 width 或 height 属性一样的值（比如 350px），则项目将占据固定空间。

5.flex 属性

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。

```
.item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]}
```

该属性有两个快捷值：auto (1 1 auto) 和 none (0 0 auto)。

建议优先使用这个属性，而不是单独写三个分离的属性，因为浏览器会推算相关值。

6.align-self 属性

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。

```
.item {
  align-self: auto | flex-start | flex-end | center | baseline | stretch;}
```

移动端适配

动端响应式开发技术

1.十多年前:百分比 %

优点:兼容性好

缺点:计算麻烦

div 30% > p 50% > img 40%

2.近 5 年左右:flex CSS3 弹性盒模型

flex （等比分）--美团 --微信小程序

flex + vw/vh 响应式单位

vw viewport width:布局 viewport 100 等分 50vw

vh viewport height

3.现在:人人 rem+js

rem root em: root->html 标记

rem 值只和 html{font-size: px}

rem+js 原理：

网页中所有的元素包括文字都是用 rem 作为单位，rem 值又是根据 innerWidth（就是 viewport）计算

计算方法:

document.getElementsByTagName("html")[0].style.fontSize=比例系数*window.innerWidth

比例系数:

如果比例系数选 1

1rem=1*window.innerWidth

半屏

iphone6/7/8 plugs 1rem=414px 0.5rem=207px1

ipphone 6/7/8 1rem=375px 0.5rem=187.5px

如果比例系数选 0.1 1/10 手淘

10rem 0.1*window.innerWidth

半屏

iphone6/7/8 plugs 10rem= 414px 5rem=207px

ipphone 6/7/8 10rem= 375px 5rem=187.5px

如果比例系数选 0.15625 1/6.4 人人网

6.4rem 0.15625*window.innerWidth

半屏

iphone6/7/8 plugs 6.4rem= 414px 3.2rem=207px

ipphone 6/7/8 6.4rem= 375px　　3.2rem=187.5px

<4>.手淘:flexible

全平台响应式开发技术：pc 移动 平板都能自适应

原则：移动优先、渐进式增强

使用技术一般是 bootstrap

---后端管理系统 界面快速构建

---访问量不高的网站

技术:媒体查询+移动端

CSS3 媒体查询:不同分辨率，应用不同选择器或不同 CSS 文件

px2rem-loade、r 解决响应式单位计算复杂的问题

rem+js/flexible 虽然实现了响应式开发，但是计算繁琐

使用 rem+js/flexible 实现响应式开发

使用 px2rem-loader/postcss-pxtorem 插件，实现 px 转换为 rem

设置好 1rem 是多少像素

1.rem 是多少像素怎么算?

看公司的设计稿（效果图）

如果设计稿 375px,flexible 设置 1rem=37.5px

如果设计稿是 750px,flexible 设置 1rem=75px

2.打包的时候，自动将 px 转换为 rem

Pc 端适配

响应式开发:同一份源码，能够自动使用不同设备分辨率。

响应式开发方向:

1.PC 端响应式开发(比较少):根据不同设备分辨率，实现不同布局（etao 网）或特殊效果（tmall 侧边栏）

技术:媒体查询

不定宽高水平垂直居中问题

--①display:flex;
可以简便、完整、响应式地实现各种页面布局。

flex 是 Flexible Box 的缩写，翻译成中文就是“弹性盒子”，用来为盒装模型提供最大的灵活性。任何一个容器都可以指定为 flex 布局。

注意：设为 flex 布局以后，子元素的 float、clear 和 vertical-align 属性将失效。
此居中元素的父元素设置 display:flex;此居中元素设置 margin:auto;
--②position:absolute;
此居中元素设置 position:absolute;(父元素是窗口或也有定位属性)并设置 left:50%; top:50%; margin:-50%（居中元素的一半宽度）;
--③position:fixed;
margin:auto;
此居中元素设置 position:fixed;并设置 left:0px; top:0px; right:0px; bottm:0px;
--④transform 属性应用于元素的 2D 或 3D 转换。这个属性允许你将元素旋转，缩放，移动，倾斜等（相对于元素本身）。
transform:translateX()　做 X 轴上的移动　transform:translateY()　做 Y 轴上的移动
此居中元素设置 position:absolute;并设置 transform:translate(-50%,-50%)

左右固定 中间自适应

中间固定两边自适应　　（圣杯）
1、flex 布局
实现思路：
设置外层父容器 display:flex;后设置 justify-content: space-between
justify-content（主轴方向上的对齐方式）：
space-between(中间对齐两侧留有空白部分）
将中间元素设置为 100%宽度来填充空白，再利用 margin 值设置边距
Flex 为 1 即可
代码：

```
<style type="text/css">
    .wrap {display: flex; justify-content: space-between;}
    .left, .right, .middle {height: 100px;}
```

```
    .left {width: 200px; background: coral;}
    .right {width: 120px; background: lightblue;}
    .middle {background: #555; width: 100%; margin: 0 20px;}
</style>
```

2 position 实现

实现思路：

左右元素设置宽高，然后设置 position: absolute; top: 0; left: 0;

中间元素设置高度，不设置宽度，设置高度，然后设置他的 margin 上下是 0，左右距离是两侧宽度的或大于两侧宽度，实现自适应

代码：

```
<div class="wrap">
    <div class="left">左侧</div>
    <div class="middle">中间</div>
    <div class="right">右侧</div>
</div>
<style type="text/css">
    .left {width: height:              position: absolute; top: 0; left: 0; }
    .right {width:height:   position: absolute; top:0; right: 0; }
    .middle {height: 50px; margin: 0 140px 0 220px; }
</style>
```

3.float 实现

实现思路：

外侧容器标签设置 overflow：hidden

左右两侧分别左右浮动，并设置宽度

中间元素设置 margin 值

【【【【【【【【【【【【【【【【注意事项！！！！！！！！！

将中间的内容放在 HTML 结构的最后，否则右侧会沉在中间内容的下侧。因为元素浮动后，脱离文档流，后面的元素受浮动影响 ，然后设置受影响元素的 margin 值即可

！！！！！】】】】】】】】】】】

代码：

```
<div class="wrap">
    <div class="left">左侧</div>
    <div class="right">右侧</div>
    <div class="middle">中间</div>
</div>

<style type="text/css">
    .wrap {background: #eee; overflow: hidden; padding: 20px;}  <!-- 生成 BFC，计算高度时考虑浮动的元素 -->
    .left {width: 200px; height: 50px; float: left; background: coral;}
    .right {width: 120px; height:  200px; float: right; background: lightblue;}
```

```
    .middle {margin-left: 220px; background: lightpink; margin-right: 140px;}
</style>
```

4.float 和 BFC 配合圣杯布局

这种情况必须将中间部分的 HTML 结构写在最前面，内层包裹一个 div，三个元素均设置向左浮动。

两侧元素宽度固定，中间设置为 100%；

然后利用左侧元素负的 margin 值进行偏移，覆盖在中间的宽度之上；

右侧的元素同样利用负的 margin 值进行覆盖

存在的问题：不能自适应高度

代码：

```
<div class="wrap">
    <div class="middle">
        <div class="main">中间</div>
    </div>
    <div class="left">左侧</div>
    <div class="right">右侧</div>
</div>
```

一个 div，包裹三个 div，第一个 div 外部包裹一个 div

```
<style type="text/css">
    .wrap {overflow: hidden;}
    .left  左侧{float: left; width: 200px; height: 100px; background: coral; margin-left: -100%;}
    .middle {float: left; width: 100%; height: 100px; background: lightblue;}
    .right {float: left; width: 120px; height: 100px; background: gray; margin-left: -120px;}
    .main {margin: 0 140px 0 220px; background: lightpink;}
</style>
```

BFC

BFC block formatting context 块格式化上下文

----块元素的渲染环境

BFC:

进入 BFC 渲染环境--需要先激活元素的 BFC

一旦进入 BFC 渲染环境后，那么就必须符合 BFC 渲染规范

BFC 可以解决 CSS 开发及布局中一些问题

bfc 在布局中的应用：

（1）防止 margin 重叠（塌陷）

（2）相邻盒子水平方向 margin 重叠

（3）嵌套元素的 margin 重叠
（4）清除内部浮动

BFC 布局规则

1、内部的 Box 会在垂直方向，一个接一个地放置。
2、Box 垂直方向的距离由 margin 决定。同一个 BFC 的两个相邻 Box 的 margin 会发生重叠。
3、每个元素的 margin box 的左边，与包含块 border box 的左边相接触(对于从左往右的格式化，否则相反)。即使存在浮动也是如此。
4、BFC 的区域不会与 float box 重叠。
5、BFC 就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。反之也是如此。
6、计算 BFC 的高度时，浮动元素也参与计算。

触发 BFC

float 属性不为 none【会影响祖先元素】;
position 为 absolute 或 fixed【会影响祖先元素】;
display 为 inline-block, table-cell, table-caption, flex, inline-flex;
overflow 不为 visible【建议使用 auto/hidden】;（如果子元素没有超过父元素的区域，仅仅是为了激活 BFC，可使用此方法）
浏览器兼容问题

解决塌陷或清除内部浮动:

1.BFC
2.添加一个空 div
3.伪元素:after

浏览器兼容性问题

:同一份源代码，在不同浏览器中显示效果不同或出现错误

1.CSS2.1 兼容性:主要是 IE6/7 太老了，从 IE8+就标准了

(1)同一个标记在不同浏览器中(IE6/7 太老了)的默

认样式不同

body: IE6/7 margin:15px 10px;
IE8+ margin:8px;

ul: IE6/7 margin-left 不为零
IE8+ padding-left 不为零

解决方案:CSS 样式初始化:

```
body,ul,ol,p,h1,h2,h3,h4,h5,h6{
    margin:0px;
    padding:0px;
}
```

(2)不同浏览器对同一个标记显示效果不同

IE6-10 浏览器认为 input type="radio/checkbox"可以设置背景色和边框
其他浏览器认为不可以给 input type="radio/checkbox"设置背景色和边框

解决方案:都不设置背景色和边框

(3)浏览器渲染 bug

img 3 像素 bug

解决方案: img{vertical-align:top;}

(4)IE6/7 其他兼容性问题:

解决方案:IE 条件注释或 CSS HACK

(5)IE6 专有 bug

1、IE6 双倍边距 bug
当页面上的元素使用 float 浮动时，不管是向左还是向右浮动；只要该元素带有 margin 像素都会使该值乘以 2，例如“margin-left:10px” 在 IE6 中，该值就会被解析为 20px。想要解决

这个 BUG 就需要在该元素中加入 display:inline 或 display:block 明确其元素类型即可解决双倍边距的 BUG。

2、IE6 中 3 像素问题及解决办法

当元素使用 float 浮动后，元素与相邻的元素之间会产生 3px 的间隙。诡异的是如果右侧的容器没设置高度时 3px 的间隙在相邻容器的内部，当设定高度后又跑到容器的相反侧了。要解决这类 BUG 的话，需要使布局在同一行的元素都加上 float 浮动。

3、IE6 中图片链接的下方有间隙

IE6 中图片的下方会存在一定的间隙，尤其在图片垂直挨着图片的时候，即可看到这样的间隙。要解决此类问题，需要将 img 标签定义为 display:block 或定义 vertical-align 对应的属性。也可以为 img 对应的样式写入 font-size:0

盒模型

标准盒模型（W3C 盒子模型）

，设置的 width 或 height 是对 实际内容（content）的 width 或 height 进行设置，内容周围的 border 和 padding 另外设置，即盒子模型的 width（height）=设置的 content 的宽高+padding+border+margin

注：除内容 content 外，其他为上下左右都有

怪异盒模型（IE 盒子模型）

，设置的 width 或 height 是对 实际内容（content）+内边距（padding）+边框（border）之和的 width 和 height 进行设置的，其盒模型的 width（height）=设置的 width（height）+外边距 margin

Css3 动画 transform 和 transition

transform 和 transition

transform 的中文翻译是变换、变形，是 css3 的一个属性，和其他 width，height 属性一样

ransition 在一定时间之内，一组 css 属性变换到另一组属性的动画展示过程。可以用来实现动态效果，css3 的属性

语法 transition：需要变换的属性 变换需要的时间 控制动画速度变化 延期多少时间后开始执行

网站变灰色

第一种：修改 CSS 文件

我们可以在网页的 CSS 文件中添加以下的 CSS 代码，来实现网页黑白色，也就是网站变灰

CSS 代码

CSS

```
html {
    filter: progid:DXImageTransform.Microsoft.BasicImage(grayscale=1);
    -webkit-filter: grayscale(100%);}
```

第二种：在网页的<head>标签内加入以下代码

如果你不想改动 CSS 文件，你可以通过在网页头部中的<head>标签内部加入内联 CSS 代码的形式实现网站网页变灰

Markup

```
<style type="text/css">
html {filter: progid:DXImageTransform.Microsoft.BasicImage(grayscale=1);-webkit-filter: grayscale(100%);}</style>
```

第三种：修改<html>标签加入内联样式

如里上面的两种方式都不喜欢，可以通过修改<html>标签，以加入内联样式的方法，达到网页变灰的效果

Markup

```
<html style="filter: progid:DXImageTransform.Microsoft.BasicImage(grayscale=1);
-webkit-filter: grayscale(100%);">
```

独占一行

使用 css 选择器给外层 div 加上以下 flex 属性，则该 div 的子 div 可以在同一行中显示，

```
.runs-paginator-flex-container {
  flex: 1 1 auto;
  flex-direction: row-reverse;
  display: flex;
}
<div className="runs-paginator-flex-container">
    <div>1</div>
    <div>2</div></div>
```

`auto`
元素会根据自身的宽度与高度来确定尺寸，但是会自行伸长以吸收flex容器中额外的自由空间，也会缩短至自身最小尺寸以适应容器。这相当于将属性设置为"`flex: 1 1 auto`".

`initial`
属性默认值，元素会根据自身宽高设置尺寸。它会缩短自身以适应容器，但不会伸长并吸收flex容器中的额外自由空间来适应容器。相当于将属性设置为"`flex: 0 1 auto`"。

`none`
元素会根据自身宽高来设置尺寸。它是完全非弹性的：既不会缩短，也不会伸长来适应flex容器。相当于将属性设置为"`flex: 0 0 auto`"。

`<positive-number>`
元素会被赋予一个容器中自由空间的指定占比。这相当于设置属性为"`flex: <positive-number> 1 0`"。

默认情况下，元素不会缩短至小于内容框尺寸，若想改变这一状况，请设置元素的`min-width`与`min-height`属性。

Position 三种的区别

position 定位:

1.position:relative; 相对定位

不脱离文档流（不影响其他元素），相对于自己原来的位置或相对于父元素进行定位

2.position:absolute;绝对定位

脱离文档流
相对于最近的带定位样式的祖先元素进行定位，如果祖先没有元素带定位样式，那么相对于浏览器窗口定位

3.position:fixed; 固定定位

脱离文档流
相对于浏览器窗口定位--IE6 不支持
PC 端网站开发:常用版式--固定居中版式

固定居中
--大容器宽度是固定的-->好处是里面的元素宽度也是固定的
--居中

元素浮动后三大变化

1.浮动元素脱离当前文档流，后面的元素顶上去，父元素高度减少浮动元素高度
2.浮动元素的宽度变成盒子实际宽度
3.后面的元素的内容错开浮动元素实际宽度，不会覆盖后面元素内容

box-sizing 有哪几种取值

content-box 标准盒模型（默认值）
border-box 怪异盒模型又叫 ie 盒模型---width = border + padding + 内容的宽度/height = border + padding + 内容的高度

文字大小的相对单位

1.px 像素

2.em:

根据自己或最近的祖先元素的 font-size 值进行计算
如果自己或祖先元素都没有设置 font-size，那么值是 16px

3.%

根据最近的祖先元素的同名样式计算 缺点:层级多了，不好计算

4.Rem

root em--移动端响应式开发 root html 标记仅根据 html{font-size:值;}进行计算

em 和 rem 区别

--Rem 是根据其 html 中的 font-size 值来进行计算的
--Em 是根据自身或其父元素的 font-size 值进行计算的，如果父元素没有 font-size 值，就根据根元素计算
em：em 相对于父元素，如果自身定义了 font-size 按自 d 身来计算（浏览器默认字体是 16px）
rem：rem 相对于根元素。相对根节点 html 的字体大小 font-size 来计算，CSS3 新加属性，chrome/firefox/IE9+支持。 px:绝对单位,页面按精确像素展示

元素隐藏 display:none；不渲染--不占宽高 visibility:hidden；渲染--单不显示（有宽高）

浏览器是怎么解析 css 的

简单来说，从右往左，例如 #a .b，是先找.b，然后找其祖先元素，匹配 #a 的

样式系统从关键选择器开始匹配，然后左移查找规则选择器的祖先元素。
只要选择器的子树一直在工作，样式系统就会持续左移，直到和规则匹配，或者是因为不匹配而放弃该规则。

为了避免 css 选择器导致的冲突，或者为了避免 css 选择器冲突，而使用很长的组合写法，现在有什么比较好的解决方案么

css-modules（webpack 的 css-loader 的）；
css module 就是把 css 写在不同的文件中，最后通过 webpack 构建工具合并成一个文件。多个不同的文件有相同的类名，合并之后没有冲突的类名。
在 webpack 中，我们使用 css-loader 来处理 css 文件，它就实现了 css module 的思想（css-loader 使用在 webpack 常用插件中有讲述）。要启用 css module，需要将 css-loader 的配置 modules

设置为 true。

css module 原理非常简单，css-loader 会将样式中的类名进行转换，转换为一个唯一的 hash 值。由于 hash 值是根据模块路径和类名生成的，因此，不同的 css 模块，哪怕具有相同的类名，转换后的 hash 值也不一样。

vue-loader 的在 vue 标签里的 scope 局部作用域（实质是 css 选择器加上属性选择器，并且 html 加上相同的属性选择器）；

像 less，sass 等，实质上并不能解决冲突问题。

三角形

1，
```
width: 0;
height: 0;
border-left: 50px solid transparent;
border-right: 50px solid transparent;
border-top: 100px solid blue;
```
2 把上、左、右三条边隐藏掉（颜色设为 transparent）

```
#demo {
    width: 0;
    height: 0;
    border-width: 20px;
    border-style: solid;
    border-color: transparent transparent red transparent;
}
```

定义变量

css 中定义变量

定义变量可分多种情况：

1、定义全局变量
```
:root {
  --borderColor: #ccc;
}
```
2、定义某元素下的变量
```
.look{
```

```
    --borderColor: #ccc;
}
```

3、定义媒体查询下的变量

```
@media screen and (min-width: 1025px) {
        :root {
                --borderColor: #ccc;
         }
}
```

使用：

```
.has-border-table > tr > td {
    border-right: 1px solid var(--borderColor);
}
```

less 中定义变量

定义：通过@定义

@bg-color : #d9d9d9;

使用：

```
.has-border-table > tr > td {
    border-right: 1px solid var(@bg-color);
}
```

使用场景：

优点：

sass 中定义变量

定义：通过$定义

$bg-color : #d9d9d9;

使用：.has-border-table > tr > td {border-right: 1px solid var($bg-color);

}

css 有哪些选择器

1.id 选择器（ # myid）

2.类选择器（.myclassname）

3.标签选择器（div, h1, p）
4.相邻选择器（h1 + p）
5.子选择器（ul > li）
6.后代选择器（li a）
7.通配符选择器（ * ）
8.属性选择器（a[rel = "external"]）
9.伪类选择器（a:hover, li:nth-child）

选择器优先级

通用选择器，标签选择器，类选择器，伪类选择器，ID 选择器，属性选择器
选择器优先级计算方法：

a	b	c	d
行内样式	id 选择器数量	类选择器 伪类选择器 属性选择器 数量	标签选择器 数量

多个选择器应用给同一个标记，样式发生冲突，优先级高的选择器生效，如果想要优先级低　　的选择器生效那么，在这个冲突样式后加　!important;提升优先级

Css 样式可被继承的属性

1、字体属性：font，font-size。。。
2、文本系列属性：text-indent：文本缩进、　　text-align：文本水平对齐、color。。。
3、元素可见性：visibility
4、表格，列表布局属性：

css 伪类

伪类 用于选择处于特定状态的元素
:link 将样式添加到未被访问过的链接
:hover　将样式添加到鼠标悬浮的元素
:active　将样式添加到被激活的元素
:visited　将样式添加到被访问过的链接

css3 新增伪类

p:only-child
选择属于其父元素唯一的子元素的每个<p>元素。
p:nth-child(n)
选择属于其父元素的第 n 个子元素的每个<p>元素。
p:nth-last-child(n)
选择属于其父元素的倒数第 n 个子元素的每个<p>元素。
p:last-child
选择属于其父元素最后一个子元素的每个<p>元素。
p:target
选择当前活动的<p>元素。
:not(p)
选择非<p>元素的每个元素。
:enabled
控制表单控件的可用状态。
:disabled
控制表单控件的禁用状态。
:checked
单选框或复选框被选中。

CSS 伪元素：

用于将特殊的效果添加到某些选择器。伪元素代表了某个元素的子元素，这个子元素虽然在逻辑上存在，但却并不实际存在于文档树中。

伪元素	作用
::first-letter	将样式添加到文本的首字母
::first-line	将样式添加到文本的首行
::before	在某元素之前插入某些内容
::after	在某元素之后插入某些内容

清除浮动

--清除浮动影响/解决塌陷
--1.BFC　overflow:hidden;
--2.在浮动元素后添加块元素，并设置其 clear 属性.clear:both;
--3.after 伪类　.box:after{
content:"\20";
display:block;
height:0px;
clear:both;
}
(可以通过 after 伪类向元素的最后添加一个空白的块元素，然后对其清除浮动,而且不会在页面中添加多余的 div，这是最推荐的方式，几乎没有副作用
在 IE6 中，不支持 after，所以在 IE6 中还需要使用 hasLayout 来处理（zoom：1）)

Position 属性

1.absolute 绝对定位，相对与设置定位的父元素，如果没有就找最近的祖先元素。
特点：脱离文档流

2.relative 相对定位，相对自身定位，
特点：不脱离文档流

3.fixed 固定定位，相对于浏览器窗口定位。
特点：脱离文档流

4.static 默认值

Css 新特性

RGBA 和 透明度
background-image background-origin(content-box/padding-box/border-box) background-size background-repeat
word-wrap（对长的不可分割单词换行）word-wrap：break-word
文字阴影：text-shadow： 5px 5px 5px #FF0000;（水平阴影，垂直阴影，模糊距离，阴影颜色）
font-face 属性：定义自己的字体

圆角（边框半径）：border-radius 属性用于创建圆角
边框图片：border-image: url(border.png) 30 30 round
盒阴影：box-shadow: 10px 10px 5px #888888
媒体查询：定义两套 css，当浏览器的尺寸变化时会采用不同的属性

块级元素

<address>...</adderss>
... 地址文字
<h1>...</h1> 标题一级
<hr> 水平分割线
... 段落
<pre>...</pre> 预格式化
<marquee>...</marquee> 滚动文本
... 无序列表
... 有序列表
<dl>...</dl> 定义列表
<table>...</table> 表格
<form>...</form> 表单
...

行内元素

...
<a>... 链接

 换行
... 加粗
... 加粗
<img > 图片
^{...} 上标
_{...} 下标
... 斜体
... 斜体
~~...~~ 删除线
<u>...</u> 下划线
<input>...</input> 文本框
<textarea>...</textarea> 多行文本
<select>...<select>

HTML5 新增标签

. <header>头部</header>
<nav>导航</nav>
<section>内容</section>
<article>左边</article>
<aside>右边</aside>
<footer>底部</footer>

Link 和@import 的区别

①link 属于 HTML 标签，而@import 是 CSS 提供的。
②页面加载的时候，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。
③import 只在 IE5 以上才能识别，而 link 是 HTML 标签，无兼容问题。
④link 样式的权重高于@import 样式的权重。

Css 可继承父元素的属性

字体属性：
font-size 字体大小
font-style：字体的风格
文本属性：
Color
line-height：行高
word-spacing：增加或减少单词的空白（即字间隔）
letter-spacing：增加或减少字符的空白（字符间距）
text-transform：控制文本大小写
列表属性
list-style-type
list-style-image
list-style-position
list-style