

CSS+SaSS

学习视频：【Udemy排名第一的高级CSS课程】Advanced CSS and Sass - 高级 CSS 和 Sass：
Flexbox、网格、动画（中英文字幕）_哔哩哔哩_bilibili

代码：<https://github.com/jonasschmedtmann/advanced-css-course.git>

常见的inline元素、block元素、inline-block元素有哪些？它们之间有什么区别？_天心天地生的博客-CSDN博客

Project Natours v1

`box-sizing: border-box` : 设置盒子长宽就包含了边框和内边距

初始化：

```
1 * {  
2     margin: 0;  
3     padding: 0;  
4     box-sizing: border-box;  
5 }  
6  
7 body {  
8     font-family: 'Lato', sans-serif;  
9     font-weight: 400;  
10    font-size: 16px;  
11    line-height: 1.7;  
12    color: #777;  
13 }
```

图片：多个背景可以在 `background-image` 这里加并用逗号分隔，靠前的在图层上面。可以用 `linear-gradient` 实现渐变。如果想让图片裁剪为多边形，使用 `clip-path` 后面加 `polygon()`，里面参数可以顺时针描述（坐标x和y相对于左上角的顶点），可以使用 bennettfeely.com/clippy/

```
1 .header {  
2     height: 95vh;  
3     background-image: linear-gradient(  
4         to right bottom,  
5         rgba(126, 213, 111, 0.8),
```

```
6     rgba(40, 180, 133, 0.8)
7     ),
8     url(..../img/hero.jpg);
9     background-size: cover;
10    background-position: top;
11 }
```

我们设置的 `absolute` 位置是相对父元素的，所以建议将父元素显示声明成 `relative`。

img图片一般只需要设置高度，宽度自动

H1一般是放标题的，谷歌通过这个标题了解你的网站是干什么的，但是如果放不下，可以在里面加两个span来区分文本的样式，但是span是不换行的，所以我们可以指定 `display: block`

居中使用 `absolute`：我们需要平移这个box的中心到不加transform的box的左上角，所以使用 `translate`

```
1 .text-box {
2     position: absolute;
3     top: 50%;
4     left: 50%;
5     transform: translate(-50%, -50%)
6 }
```

动画：注意动画移动的时候可能出现Y轴颤抖，可以在使用动画的元素的父元素上加上 `backface-visibility: hidden;`

```
1 .heading-primary-main {
2     animation-name: moveInLeft;
3     animation-duration: 3s;
4     /*
5      其他属性：
6      animation-delay 开始前延迟时间,
7      animation-iteration-count 动画执行次数
8      animation-timing-function: ease-out
9     */
10 }
11
12 @keyframes moveInLeft {
13     0% {
14         opacity: 0;
15         transform: translateX(-100px);
16     }
17 }
```

```
18     80% {
19         transform: translateX(20px);
20     }
21
22     100% {
23         opacity: 1;
24         transform: translate(0);
25     }
26 }
```

对于a标签，有好几种伪类，例如 `:link` 就是正常状态，`:visited` 点击之后，我们可以用 `a:link, a:visited` 来统一，另外 `a:hover / a:active` 中的 `transform` 都是相对于 `link` 状态。默认是 `inline` 元素，可以用 `display: inline-block;` 来给他加 padding 或者长宽，同时 `inline-block` 看作是 `text`，所以只需要父元素设置 `text-align: center` 就可以居中。动画除了使用关键帧这种复杂的方法，也可以用 `transition` 这种简单的方法：`transition: all .2s`，就是在 `link` 初始状态写 `transition`，在其他状态直接写 `transform`

```
1 .btn:link
2 .btn:visited {
3     text-transform: uppercase;
4     text-decoration: none;
5     padding: 15px 40px;
6     display: inline-block;
7     border-radius: 100px;
8     transition: all .2s;
9 }
10
11 /*
12     hover 的时候阴影看起来远，active 的时候看起来近，然后位置有点击的感觉，加上
13     transition 实现动画
14 */
15 .btn:hover {
16     transform: translateY(-3px);
17     box-shadow: 0 10px 20px rgba(0, 0, 0, 0.2);
18 }
19 .btn:active {
20     transform: translateY(-1px);
21     box-shadow: 0 5px 1px rgba(0, 0, 0, 0.2);
22 }
```

使用伪元素，必须需要填写 `content` 属性，可以为空，还必须有 `display`。我们希望它能够在我们定义的 `btn` 的下面，我们可以使用 `absolute`

在CSS中，`position: absolute` 是一种定位属性，用于指定元素的位置。与其他定位属性（如`position: static`、`position: relative` 和 `position: fixed`）相比，`position: absolute` 的参考位置是距离最近的已定位祖先元素（除了 `static`），或者如果没有已定位的祖先元素，则是相对于包含块（通常是文档的 `<body>` 元素）。

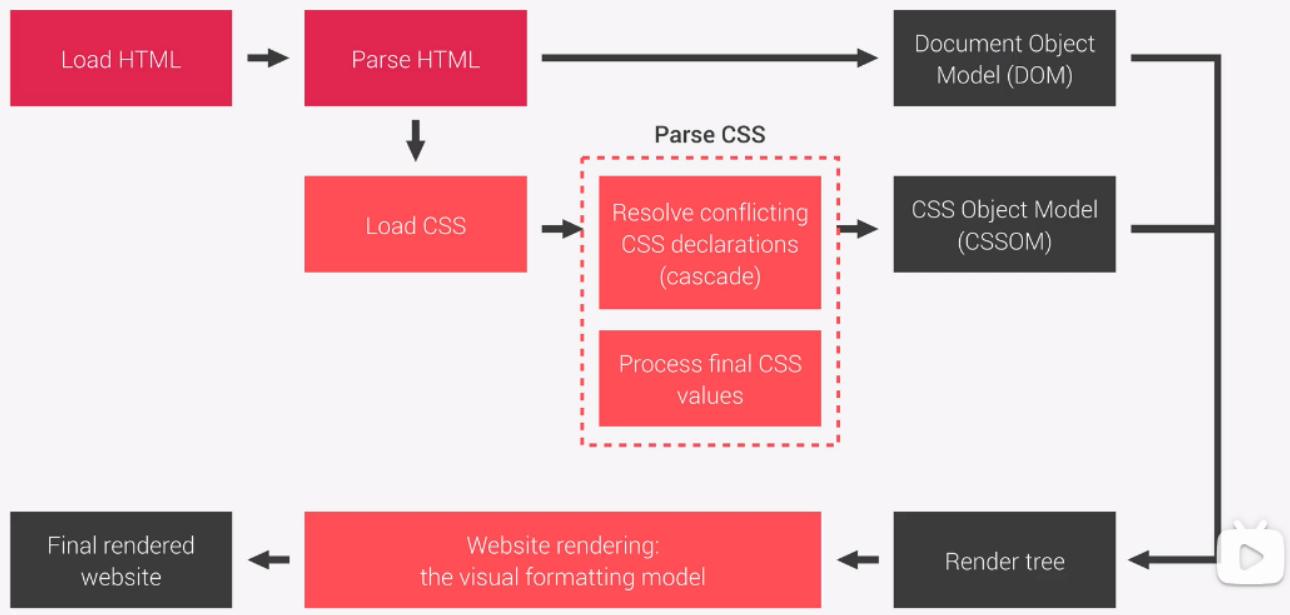
```
1 .btn::after {  
2     content: "";  
3     display: inline-block;  
4     height: 100%;  
5     width: 100%; /* 这里的比例是根据 btn 来的 */  
6     border-radius: 100px;  
7     position: absolute; /* 会找到的一个可以参考相对位置的元素 */  
8     left: 0;  
9     top: 0;  
10    z-index: -1;  
11 }  
12  
13 .btn-white::after {  
14     background-color: #fff;  
15 }  
16  
17 .btn:hover::after {  
18     /* 悬停时看到after */  
19     transform: scale(2);  
20 }
```

另外如果混用 keyframe 然后有时间前后，此时可能会出现按钮一开始就出现等，此时可以给后面的动画加一个 `animation-fill-mode: backwards`

CSS

- 三大支柱：响应式设计、可维护和可拓展的代码、网络表现

WHAT HAPPENS TO CSS WHEN WE LOAD UP A WEBPAGE?



- 常用测试链接: codepen.io
- 单位换算:

HOW CSS VALUES ARE PROCESSED



HOW UNITS ARE CONVERTED FROM RELATIVE TO ABSOLUTE (PX)

	Example (x)	How to convert to pixels	Result in pixels	
Font-based	% (fonts)	150%	$x\% * \text{parent's computed font-size}$	24px
	% (lengths)	10%	$x\% * \text{parent's computed width}$	100px
	em (font)	3em	$x * \text{parent computed font-size}$	72px ($3 * 24$)
	em (lengths)	2em	$x * \text{current element computed font-size}$	48px
	rem	10rem	$x * \text{root computed font-size}$	160px
	viewport-based	90vh	$x * 1\% \text{ of viewport height}$	90% of the current viewport height
Viewport-based	vw	80vw	$x * 1\% \text{ of viewport width}$	80% of the current viewport width

html, body { font-size: 16px; width: 80vw; }
 header { font-size: 150%; padding: 2em; margin-bottom: 10rem; height: 90vh; width: 1000px; }
 .header-child { font-size: 3em; padding: 10%; }



- 将所有px都转换成rem，首先我们在 `html { font-size: 10px }` 来设置根字体大小，设置成10px就可以让转换变得更加简单，但是这样会导致用户对页面字体的缩放失效，所以建议使用百分比，这里可以使用62.5%
- 可以在 `* { box-sizing: inherit }`，然后在 `body { box-sizing: border-box }`

```

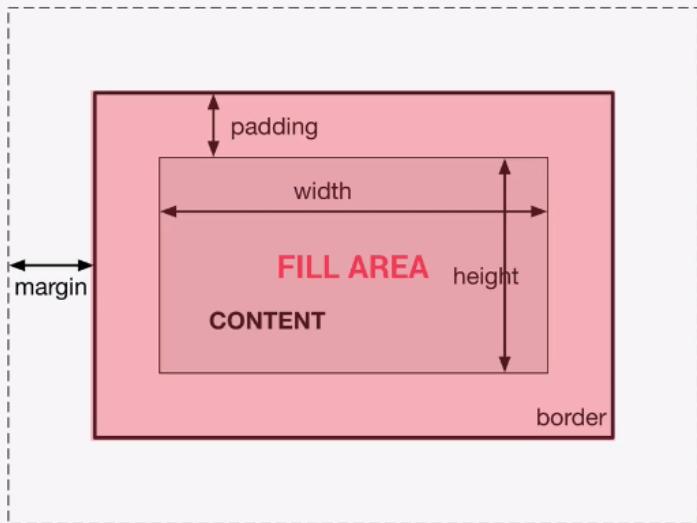
*, *::after,
*::before {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
}

html {
  font-size: 62.5%;
}

body {
  font-family: "Lato", sans-serif;
  font-weight: 400;
  /*font-size: 16px;*/
  line-height: 1.7;
  color: #777;
  padding: 3rem;

  box-sizing: border-box;
}
  
```

- 盒子模型：注意FILL AREA。当设置border-box的时候width就包括了padding和border



- **Content:** text, images, etc;
- **Padding:** transparent area around the content, inside of the box;
- **Border:** goes around the padding and the content;
- **Margin:** space between boxes;
- **Fill area:** area that gets filled with background color or background image.

- Block 和 inline 和 inline-block 的区别

2. BOX TYPES: INLINE, BLOCK-LEVEL AND INLINE-BLOCK



Block-level
boxes

Inline-block
boxes

Inline
boxes

- Elements formatted visually as blocks
- 100% of parent's width
- Vertically, one after another
- Box-model applies as showed

- A mix of block and inline
- Occupies only content's space
- No line-breaks
- Box-model applies as showed

- Content is distributed in lines
- Occupies only content's space
- No line-breaks
- No heights and widths
- Paddings and margins only horizontal (left and right)

```
display: block
(display: flex)
(display: list-item)
(display: table)
```

```
display: inline-block
```

```
display: inline
```



所以我们可以根据不同的情况下根据需要使用它们。191

- 定位

3. POSITIONING SCHEMES: NORMAL FLOW, ABSOLUTE POSITIONING AND FLOATS

Normal flow

FLOATS

Absolute
positioning

- Default positioning scheme;
- NOT floated;
- NOT absolutely positioned;
- Elements laid out according to their source order.

- Element is removed from the normal flow;
- Text and inline elements will wrap around the floated element;
- The container will not adjust its height to the element.

- Element is removed from the normal flow
- ≠
- No impact on surrounding content or elements;
 - We use top, bottom, left and right to offset the element from its relatively positioned container.

Default
position: relative

float: left
float: right

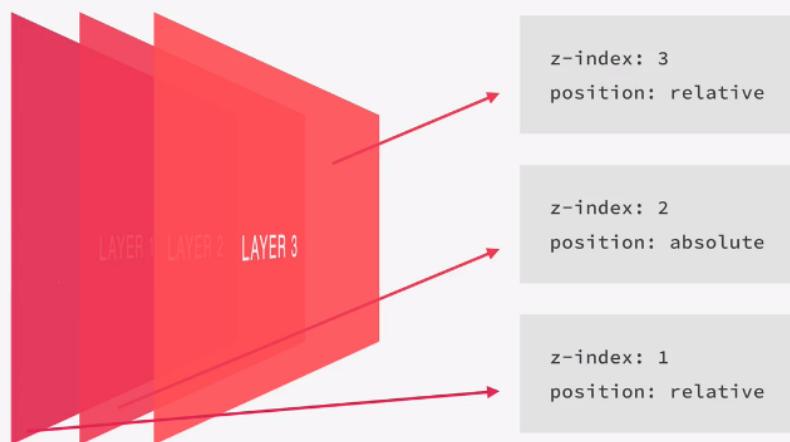
position: absolute
position: fixed



现在正如我提到的，一个绝对定位的元素

- 堆叠上下文，如果有多个页面重叠，那么会根据堆叠上下文显示，可以使用 `z-index` 以及其他一些属性，因此有时候只设置 `z-index` 是没有用的

4. STACKING CONTEXTS



- 命名规范：BEM，Block Element Modifier，Block是一个独立的component，然后它的子元素都采用block的名字开头，然后不同的子元素后面加modifier。要求所有标签都有一个类



THINK

BUILD

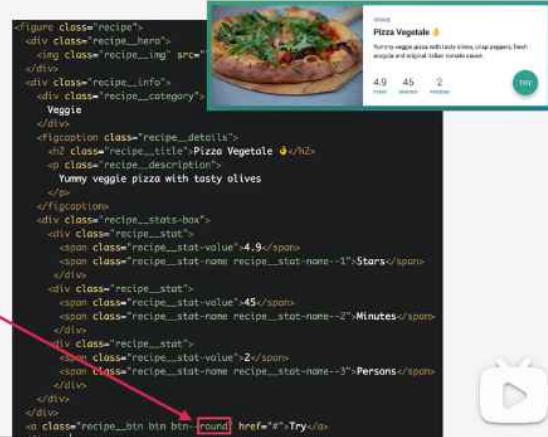
ARCHITECT

BEM

- **Block Element Modifier**
- **BLOCK**: standalone component that is meaningful on its own.
- **ELEMENT**: part of a block that has no standalone meaning.
- **MODIFIER**: a different version of a block or an element.

```
.block {}
.block_element {}
.block_element--modifier {}
```

Low-specificity BEM



现在再一次，你会习惯这一切的

- 架构：7-1

THINK

BUILD

ARCHITECT

THE 7-1 PATTERN

7 different folders for partial Sass files, and 1 main Sass file to import all other files into a compiled CSS stylesheet.

THE 7 FOLDERS

- base/
- components/
- layout/
- pages/
- themes/
- abstracts/
- vendors/



对于每个组件，布局文件夹，

SaSS

SaSS是CSS的预处理器

当一个容器内所有的元素都是 float，那么它的父元素会高度塌陷，解决方法是给父元素加一个

`clearfix`，其中 `.clearfix::after { content: "", clear: both, display: table }`

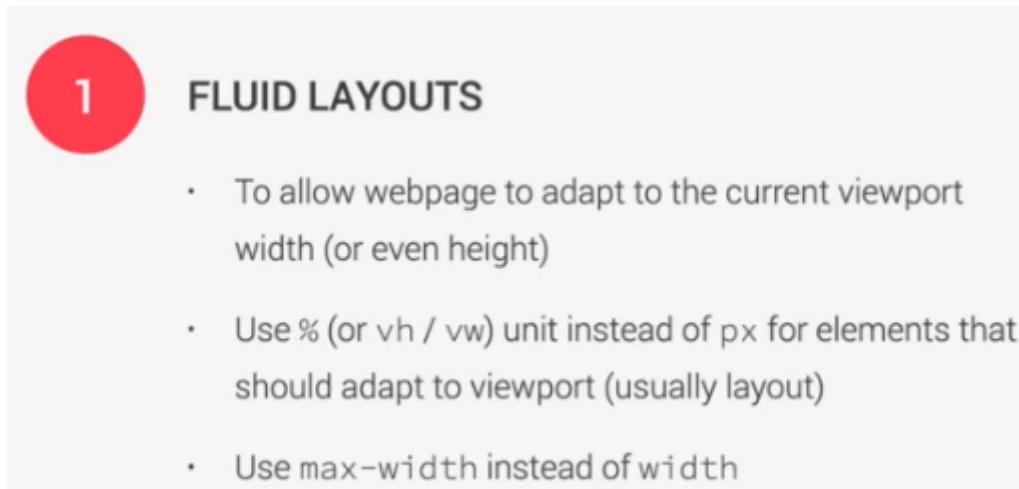
mixin的定义：`@mixin name($arg) {}`，调用：`@include name()`，另外还有

`@function name($a, $b) { @return $a / $b; }`

@extends 不推荐用

Project Natours v2

- 首先可以将 BEM 命令改称 `header { ...; &__logo-box: ...; }`
- 响应式设计的四种原则：
 - Fluid Layouts：使用百分比而不是 px，使用 max-width 而不是 width

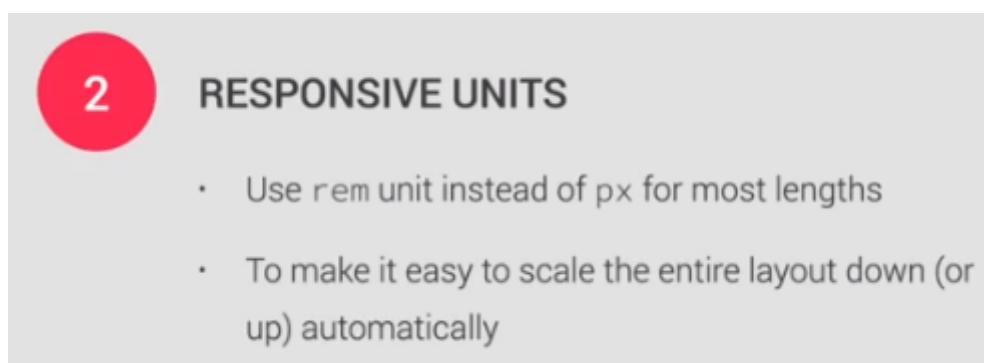


1 FLUID LAYOUTS

- To allow webpage to adapt to the current viewport width (or even height)
- Use % (or vh / vw) unit instead of px for elements that should adapt to viewport (usually layout)
- Use max-width instead of width

一共有三种layout方法

- Float，旧的方法
- Flexbox
- CSS Grid
- Responsive Units：使用 rem 而不是 px



2 RESPONSIVE UNITS

- Use rem unit instead of px for most lengths
- To make it easy to scale the entire layout down (or up) automatically

- Flexible Images：使用 % 以及 max-width

3

FLEXIBLE IMAGES

- By default, images don't scale automatically as we change the viewport, so we need to fix that
- Always use % for image dimensions, together with the max-width property

- Media Queries

4

MEDIA QUERIES

- To change CSS styles on certain viewport widths (called breakpoints)

- `max-width` 和 `width` 的区别是，如果窗口的宽度没有达到那个值就占满
- 想让一个 `block` 水平居中，可以设置 `margin: 0 auto`
- 有一个伪类 `not`，用法：`&:not(:last-child) {}`，可以选择除了最后一个之外的所有
- css自带 `calc` 函数能够计算不同单位的算式
- 有其他的选择器，例如 `[class^="col-"]` 就是表示所有class是以col开头，把`^`换成`*`就是表示class包含后面那个即可，`$` 表示以……结尾， `[src=""]` 和 `[alt=""]`
- html的 `<body>` 部分可以分成 `<header>` `<main>` `<footer>`，然后每个部分可以用 `<section>`
- html有一个Emmet Extension，需要一个带test的class的div只要输入 `.test` 加 tab 键即可，如果是带test类的其他标签需要输入 `section.section-about`
- 给一段文本作渐变，首先给这个 block 加上将渐变的背景：`background-image: linear-gradient(to right, ..., ...)`，然后我们要求背景只在文字上，我们就设置 `display: inline-block`，然后将背景裁剪成文字：`-webkit-background-clip: text;`，最后将文字设置成透明色：`color: transparent`
- `transform` 属性可以加多个动画
- 为了让文本居中 (inline，例如 img)，可以给文字外面包一个container，然后让文字居中，这种 container可以成为utility类，命名成 `u-text-center`，`text-align: center`
- 资源网站，例如 & 开头的字符：codingheroes.io/resources
- Emmet其他神奇的用法：`.composition>(img.composition__photo.composition__photo--p1)*3`

```
<div class="composition">
  <img src="" alt="" class="composition__photo composition__photo--p1">
  <img src="" alt="" class="composition__photo composition__photo--p1">
  <img src="" alt="" class="composition__photo composition__photo--p1">
</div>
```

```
.composition {
  position: relative;

  &__photo {
    width: 55%;
    box-shadow: 0 1.5rem 4rem rgba($color-black, .4);
    border-radius: 2px;
    position: absolute;

    &--p1 {
      left: 0;
      top: -2rem;
    }

    &--p2 {
      right: 0;
      top: 2rem;
    }

    &--p3 {
      left: 20%;
      top: 10rem;
    }
  }
}
```

图片最好使用百分比并且只指定 `width`

- 给图片增加有空隙的边框: `outline-offset: 2rem; outline: 1.5rem solid $color`
- 让一个组内其他没有被hover的元素变小: `&:hover &__photo:not(:hover) { transform: scale(.9); }`
- 图标下载: <linea.io>, 建议使用 iconfont 或者 svg, 前者可以用 `<i>` 作为引入, `i` 之前是斜体, 现在不用了所以常用作 iconfont, 前者的修改大小就是修改`font-size`
- 想要实现背景图片倾斜显示, 除了上面的`clip-path`, 还可以用 `transform: skewY(-7deg)`, 然后让它的子元素全都反向倾斜: `&>* { skewY(7deg); }`, 这个表示的是直接子元素
- 如果发生重叠且子元素是`absolute`, 那么没有办法不能使用 `clearfix`, 只能设置父子元素高度都是同一个高度
- 实现一个可旋转的卡片
 - 首先设置 `perspective`, 这个值尽量大, 为了支持 mozilla需要加一个 `-moz-perspective`
 - 然后设置 `backface-visibility`, 使得不能看到反面的字

- 卡片的两面，一个初始转180deg，另一个初始没有转。但是这里有一个小错误，或许是-180deg，这个需要自己尝试

```
.card {
    perspective: 150rem;
    -moz-perspective: 150rem;
    position: relative;
    height: 50rem;

    &__side {
        background-color: #orangered;
        color: #fff;
        font-size: 2rem;

        height: 50rem;
        transition: all .8s ease;
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        backface-visibility: hidden;

        &--front {
            background-color: #orangered;
        }

        &--back {
            background-color: green;
            transform: rotateY(180deg);
        }
    }

    &:hover &__side--front {
        transform: rotateY(180deg);
    }

    &:hover &__side--back {
        transform: rotateY(0);
    }
}
```

- 免费图片网站：unsplash.com，视频：cover.co
- `background-blend-mode: screen` 支持多个`background-image`，即可以在图片上加渐变
- 当父元素有弯曲的边角的时候，而子元素和它等大的时候，会出现`overflow`，所以可以在父元素上加 `overflow: hidden`
- `clip-path`:

```
-webkit-clip-path: polygon(0 0, 100% 0, 100% 85%, 0 100%);
clip-path: polygon(0 0, 100% 0, 100% 85%, 0 100%);
```

- 当一段文本被`break`的时候，加`padding`只会在第一行和最后一行加，所以需要加一个新的属性：

```
-webkit-box-decoration-break: clone;
box-decoration-break: clone;
```

- 无序列表：

- ul 是一个block，让block居中的方法是 margin: 0 auto

```
&__details {  
    padding: 3rem;  
  
    ul {  
        list-style: none;  
        width: 80%;  
        margin: 0 auto; }  
  
    li {  
        text-align: center;  
        font-size: 1.5rem;  
        padding: 1rem;  
  
        &:not(:last-child) {  
            border-bottom: 1px solid $color-grey-light-2;  
        }  
    }  
}
```

- 设置文字围绕圆形图片，我们可以用 <figure> 这个标签（加上caption的图片），然后我们要设置图片的长宽。另外我们就需要直接使用 clip-path 将图片真正裁剪。我们也不要使用 padding和margin，而是使用 translateX(-3rem) 来实现间隔。然后在 <figure> 里面加上 和 <figcaption>

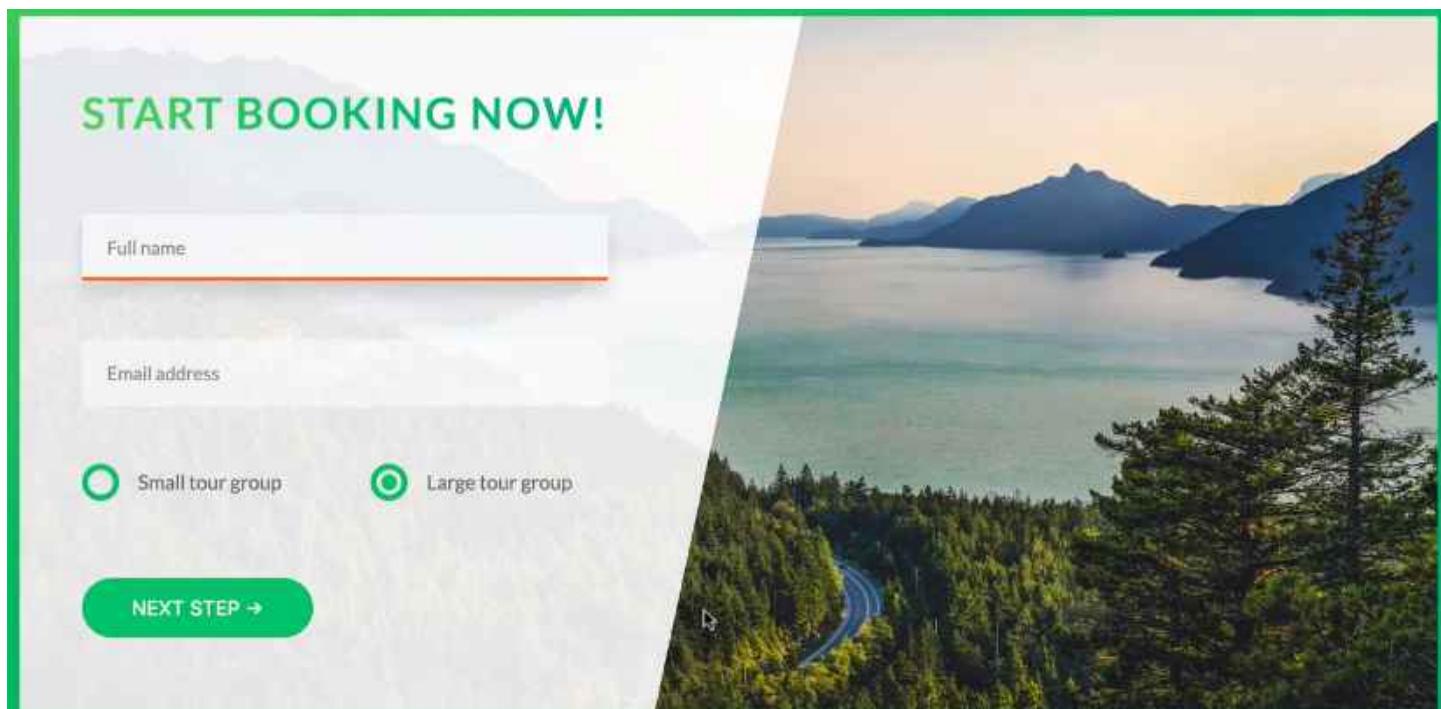
```
float: left;  
-webkit-shape-outside: circle(50% at 50% 50%);  
shape-outside: circle(50% at 50% 50%);
```

- filter: blur(3px) brightness(80%)
- 引入视频作为网页背景： object-fit: cover 和 background-size: cover 类似，即保持纵横比然后剪裁多余部分

```
<div class="bg-video">  
    <video class="bg-video__content" autoplay muted loop>  
        <source src="img/video.mp4" type="video/mp4">  
        <source src="img/video.webm" type="video/webm">  
        Your browser is not supported!  
    </video>  
</div>
```

```
.bg-video {  
    position: absolute;  
    top: 0;  
    left: 0;  
    height: 100%;  
    width: 100%;  
    z-index: -1;  
    opacity: .15;  
    overflow: hidden; }  
  
&__content {  
    height: 100%;  
    width: 100%;  
    object-fit: cover;  
}
```

- 使用渐变实现透明的效果，但设置颜色后面的百分比是一样的时候就会产生一条实线



```
background-image: linear-gradient(105deg,  
    transparent 0%,  
    transparent 50%,  
    url(..../img/nat-10.jpg);
```

- 和 之前的绑定通过 和 字段，然后当你选中label的时候也会输入。单选按钮的实现是通过同一个
- 用 可以去掉选中后整个框，但是对于使用键盘的用户最好加上一些特征，例如 和 ，由于是focus状态加border，所以会导致高度变化，所以可以在非focus的状态也加上，但是设置颜色是transparent
- 另外这些输入元素往往不会继承字体相关的内容，所以需要 以及
- 设置当输入有效和无效的时候的效果

- 兄弟选择器，如果是直接相邻用 `+`，否则用 `~`，另外前后顺序有关

```
<div class="form__group">
  <input type="text" class="form__input" placeholder="Full name" id="name" required>
  <label for="name" class="form__label">Full name</label>
</div>
```

```
&__input {
  font-size: 1.5rem;
  font-family: inherit;
  padding: 1.5rem 2rem;
  border-radius: 2px;
  background-color: rgba($color-white, .5);
  border: none;
  border-bottom: 3px solid transparent;
  width: 100%;
  display: block;
}

&__input:focus {
  outline: none;
  box-shadow: 0 1rem 2rem rgba($color-black, .1);
  border-bottom: 3px solid $color-primary;
}
```

```
&:focus:invalid {
  border-bottom: 3px solid $color-secondary-dark;
}

&::placeholder {
  color: $color-grey-dark-2;
}
```

```
&__label {
  font-size: 1.2rem;
  font-weight: 700;
  margin-left: 2rem;
  margin-top: .7rem;
  display: block;
  transition: all .3s;
}

&__input:placeholder-shown + &__label {
  opacity: 0;
  visibility: hidden;
  transform: translateY(-4rem);}
```

- 实现一个圈中套一个圆来实现单选按钮的效果，注意最后一行的伪类。我们想要替换掉之前的单选按钮，不显示的方法：`display: none`，且依旧能够点击

```
<div class="form__group">
  <div class="form__radio-group">
    <input type="radio" class="form__radio-input" id="small" name="size">
    <label for="small" class="form__radio-label">
      <span class="form__radio-button"></span>
      Small tour group
    </label>
  </div>

  <div class="form__radio-group">
    <input type="radio" class="form__radio-input" id="large" name="size">
    <label for="large" class="form__radio-label">
      <span class="form__radio-button"></span>
      Large tour group
    </label>
  </div>
</div>
```

```
&__radio-button {
  height: 3rem;
  width: 3rem;
  border: 5px solid $color-primary;
  border-radius: 50%;
  display: inline-block;
  position: absolute;
  left: 0;
  top: -.4rem;

  &::after {
    content: "";
    display: block;
    height: 1.3rem;
    width: 1.3rem;
    border-radius: 50%;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: $color-primary;
    opacity: 0;
    transition: opacity .2s;
  }
}

&__radio-input:checked ~ &__radio-label &__radio-button::after {
  opacity: 1;
}
```

- 在utility组件中加上 `!important` 是可以的
- 如何让border只显示到文字所在的位置: `display: inline-block`
- 要实现点击按钮扩展到整个屏幕, 我们要hack checkbox

```
<div class="navigation">
  <input type="checkbox" class="navigation__checkbox" id="navi-toggle">
  <label for="navi-toggle" class="navigation__button">MENU</label>
  <div class="navigation__background">&nbsp;</div>
  <nav class="navigation__nav">
    <ul class="navigation__list">
      <li class="navigation__item"><a href="#" class="navigation__link"></a>About Natous</li>
      <li class="navigation__item"><a href="#" class="navigation__link"></a>Your benfits</li>
      <li class="navigation__item"><a href="#" class="navigation__link"></a>Popular tours</li>
      <li class="navigation__item"><a href="#" class="navigation__link"></a>Stories</li>
      <li class="navigation__item"><a href="#" class="navigation__link"></a>Book now</li>
    </ul>
  </nav>
</div>
```

- 实现一个点击渐变从右到左的效果

```
&__link {
  &:link,
  &:visited {
    display: inline-block;
    font-size: 3rem;
    font-weight: 300;
    padding: 1rem 2rem;
    color: $color-white;
    text-decoration: none;
    text-transform: uppercase;
    background-image: linear-gradient(120deg, transparent 0%, transparent 50%, $color-white 50%);
    background-size: 220%;
    transition: all .4s;
  }
  &:hover,
  &:active {
    background-position: 100%;
    color: $color-primary;
    transform: translateX(1rem);
  }
}
```

- 贝塞尔曲线网站：easings.net, cubic-bezier.com
- 实现一个三横线的Icon

```
&__icon {
  position: relative;
  margin-top: 3.5rem;

  &,
  &::before,
  &::after {
    width: 3rem;
    height: 2px;
    background-color: $color-grey-dark-3;
    display: inline-block;
  }

  &::before,
  &::after {
    content: "";
    position: absolute;
    left: 0;
  }

  &::before { top: -.8rem; }
  &::after { top: .8rem; }
}
```

- `transform-origin` 设置变换的原点，一般用来作旋转原点的效果
- 让左右两个box高度相同的方法：父元素设置成 `table`，子元素设置成 `table-cell`

```
&__content {
  @include absCenter;

  width: 75%;
  background-color: $color-white;
  box-shadow: 0 2rem 4rem rgba($color-black, .2);
  border-radius: 3px;
  display: table;
}

&__left {
  width: 33.333333%;
  display: table-cell;
}

&__right {
  width: 66.666667%;
  display: table-cell;
  vertical-align: middle;
```

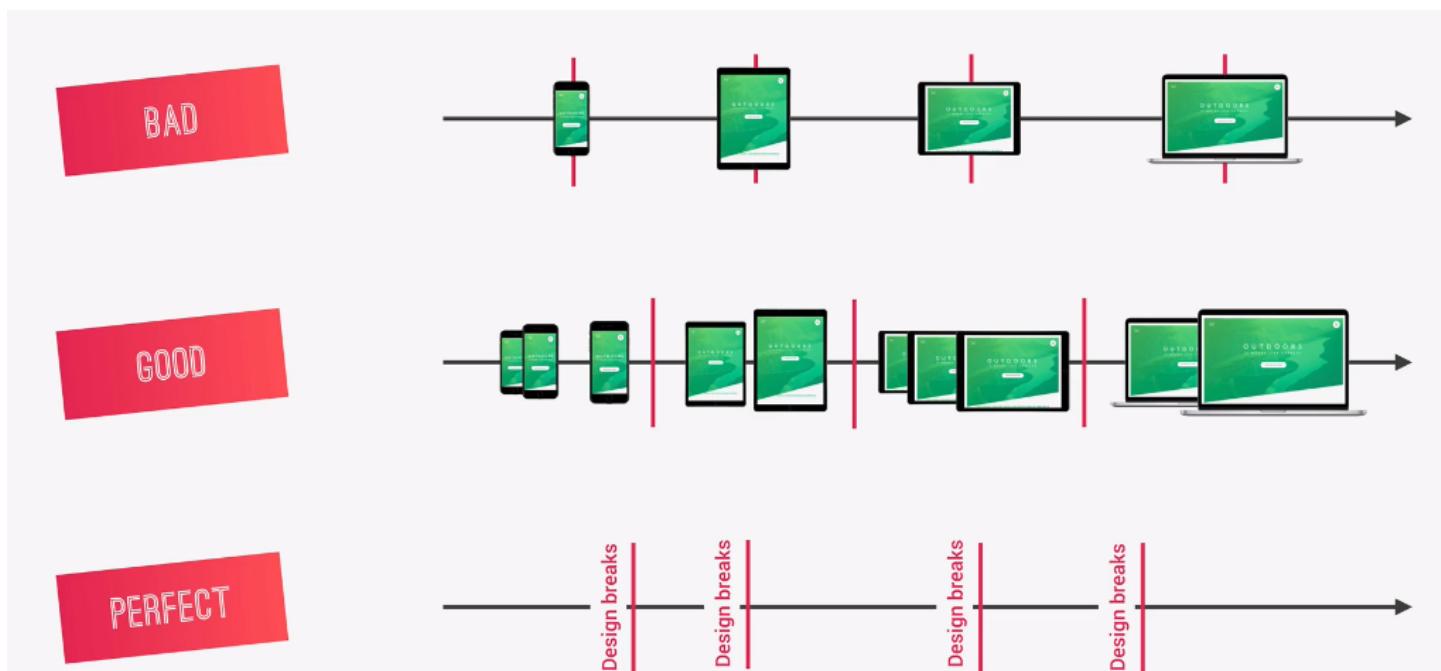
- 文本分列：`column-count: 2`。但是不建议自己加前缀

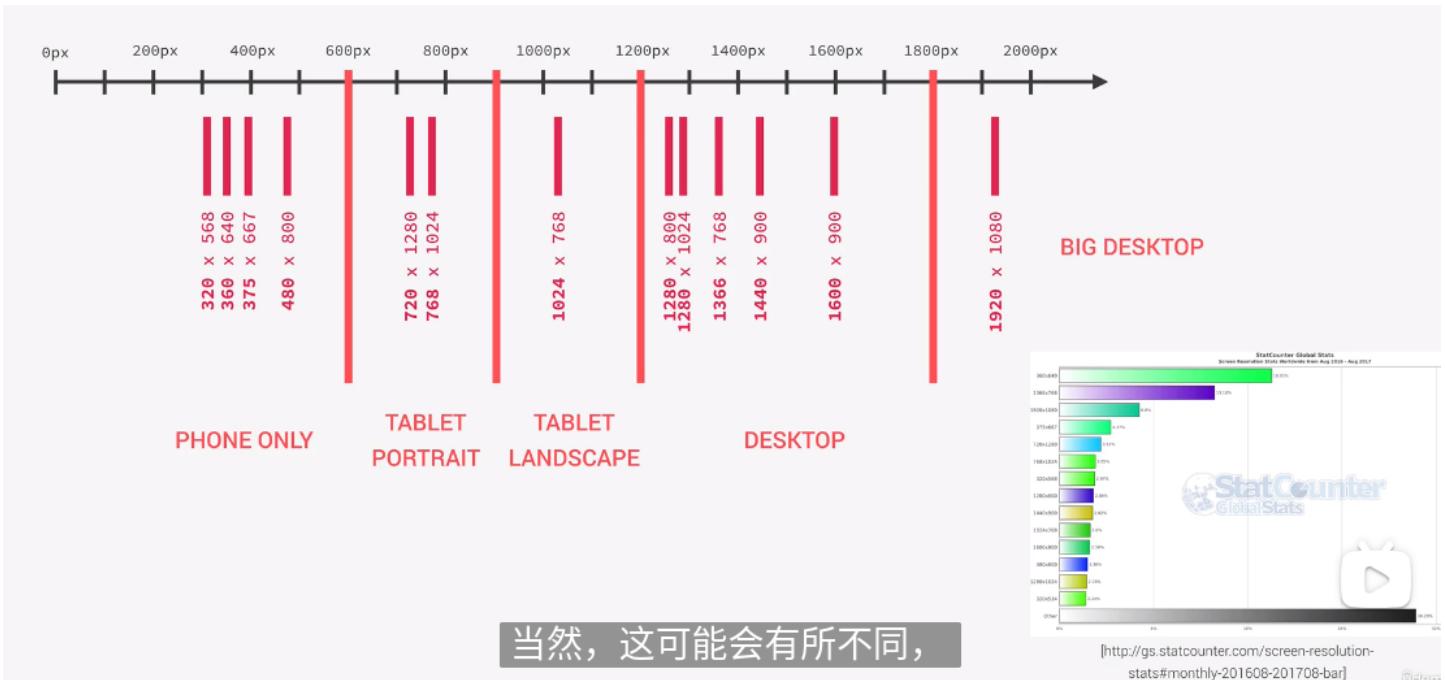
```
-moz-column-count: 2;  
-moz-column-gap: 4rem; //1em = 14px;  
-moz-column-rule: 1px solid $color-grey-light-2;  
  
column-count: 2;  
column-gap: 4rem; //1em = 14px;  
column-rule: 1px solid $color-grey-light-2;  
  
-moz-hyphens: auto;  
-ms-hyphens: auto;  
-webkit-hyphens: auto;  
hyphens: auto;
```

- 有一个伪类是 `:target`，就是当 `href` 索引到这个的时候会触发，例如有一个 `href='popup'`，另一个元素的 `id` 是 `popup` 就会触发

Project Natours v3 响应式设计

- 设置断点的三种方法：第一种是对特定设备设计，第二种是对一类产品进行设计，第三种是根据自己的 design 设计，一但观感不好就设置一个断点





当然，这可能会有所不同，

- 使用Sass的mixin

```
/*
$breakpoint argument choices:
- phone
- tab-port
- tab-land
- big-desktop
*/
@mixin respond($breakpoint) {
  @if $breakpoint == phone {
    |   @media (max-width: 600px) { @content };
  }
  @if $breakpoint == tab-port {
    |   @media (max-width: 900px) { @content };
  }
  @if $breakpoint == tab-land {
    |   @media (max-width: 1200px) { @content };
  }
  @if $breakpoint == big-desktop {
    |   @media (min-width: 1800px) { @content };
  }
}
```

但是在媒体查询里rem不起作用，需要换成em：

```

/*
$breakpoint argument choices:
- phone
- tab-port
- tab-land
- big-desktop

1em = 16px
*/
@mixin respond($breakpoint) {
  @if $breakpoint == phone {
    @media (max-width: 37.5em) { @content };      //600px
  }
  @if $breakpoint == tab-port {
    @media (max-width: 56.25em) { @content };      //900px
  }
  @if $breakpoint == tab-land {
    @media (max-width: 75em) { @content };        //1200px
  }
  @if $breakpoint == big-desktop {
    @media [min-width: 112.5em] { @content };      //1800px
  }
}

```

```

html {
  // This defines what 1rem is
  font-size: 62.5%; //1 rem = 10px; 10px/16px = 62.5%
  |
  @include respond(tab-land) { // width < 900?
    font-size: 56.25%; //1 rem = 9px, 9/16 = 50%
  }

  @include respond(tab-port) { // width < 600?
    font-size: 50%; //1 rem = 8px, 8/16 = 50%
  }

  @include respond(big-desktop) {
    font-size: 75%; //1rem = 12, 12/16
  }
}

```

`tab-port` 和 `phone` 一样，所以就都用 `font-size: 50%`，注意顺序是有关的

- 修改的顺序：base + typography >> general layout + grid >> page layout >> components
- 修改字体相关的例子，注意顺序。使用浏览器的DevTools的响应式功能查看

```
.heading-secondary {
  font-size: 3.5rem;
  text-transform: uppercase;
  font-weight: 700;
  display: inline-block;
  background-image: linear-gradient(to right, $color-primary-light, $color-primary-dark);
  -webkit-background-clip: text;
  color: transparent;
  letter-spacing: .2rem;
  transition: all .2s;

  @include respond(tab-port) {
    font-size: 2.5rem;
  }

  @include respond(phone) {
    font-size: 2rem;
  }
}
```

- 修改Grid，在手机上一般就直接变成列显示

```
.row {
  max-width: $grid-width;
  margin: 0 auto;

  &:not(:last-child) {
    margin-bottom: $gutter-vertical;

    @include respond(tab-port) {
      margin-bottom: $gutter-vertical-small;
    }
  }

  @include clearfix;

  [class^="col-"] {
    float: left;

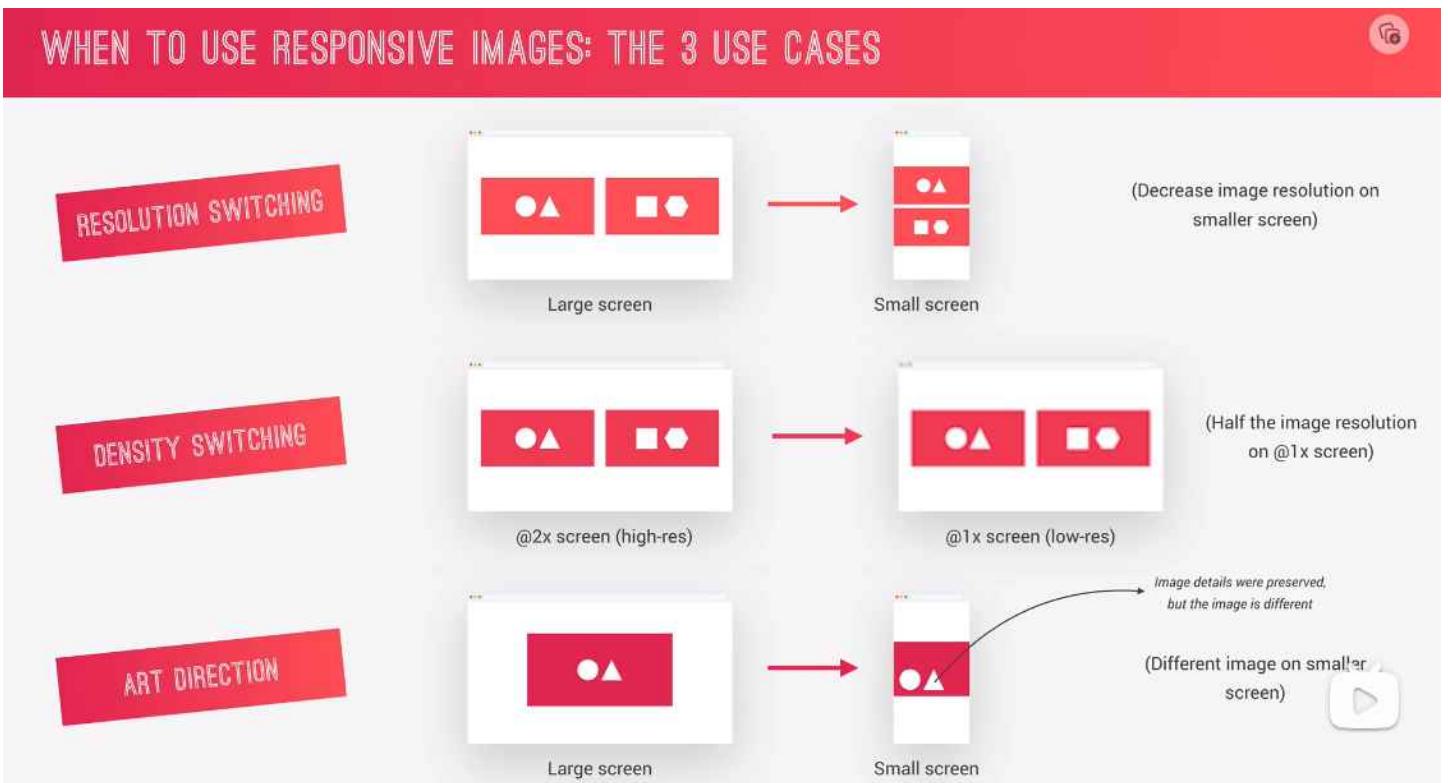
    &:not(:last-child) {
      margin-right: $gutter-horizontal;

      @include respond(tab-port) {
        margin-right: 0;
      }
    }

    @include respond(tab-port) {
      width: 100% !important;
    }
  }
}
```

- 测试不同的屏幕：sizzy.co
- 响应式图片对性能极其关键。小的设备要用小的图片，三种使用响应式图片的场景，不同的场景有不同的应对方法：
 - 分辨率
 - 像素密度

- 艺术相关



- HTML中的响应式图片

- 解决像素密度的方法：`` 的 `src` 修改成 `srcset`，这样可以指定多个图片
`srcset=xxx 1x, xxx 2x`
- 解决艺术相关的方法：使用 `<picture>` 和 `<source>`，其中后者可以指定 `media`

```
<picture class="footer__logo">
  <source srcset="img/logo-green-small-1x.png 1x, img/logo-green-small-2x.png 2x"
          media="(max-width: 37.5em)"
  <img srcset="img/logo-green-1x.png 1x, img/logo-green-2x.png 2x" alt="Full logo">
</picture>
```

- 解决分辨率相关的问题：`srcset` 中的宽度标识一般就是图片本身的尺寸加一个w，`sizes`是告知浏览器图片大概的宽度，其中的vw是用devtools调好之后看的。`src` 是避免浏览器不支持上面的操作

```

```

- CSS中的响应式图片

- 继续用媒体查询，主要使用了 `min-resolution`，这里 `192dpi` 表示的是 2x 屏幕，又因为小屏幕没必要使用高分辨率图片，所以这里使用了 `and`。另外可以使用 `,` 表示 or。注意使用 `em`。另外对于Safari加了-webkit那一行

```

.header {
    height: 95vh;
    background-image: linear-gradient(
        to right bottom,
        rgba($color-primary-light, 0.8),
        rgba($color-primary-dark, 0.8)),
    url(..../img/hero-small.jpg);
    background-size: cover;
    background-position: top;
    position: relative;
    -webkit-clip-path: polygon(0 0, 100% 0, 100% 75vh, 0 100%);
    clip-path: polygon(0 0, 100% 0, 100% 75vh, 0 100%);

    @media (min-resolution: 192dpi) and (min-width: 600px) {
        background-image: linear-gradient(
            to right bottom,
            rgba($color-secondary-light, 0.8),
            rgba($color-secondary-dark, 0.8)),
        url(..../img/hero.jpg);
    }

    @media (min-resolution: 192dpi) and (min-width: 37.5em),
        (-webkit-min-device-pixel-ratio: 2) and (min-width: 37.5em)
        (min-width: 125em) {
        background-image: linear-gradient(
            to right bottom,
            rgba($color-secondary-light, 0.8),
            rgba($color-secondary-dark, 0.8)),
        url(..../img/hero.jpg);
    }
}

```

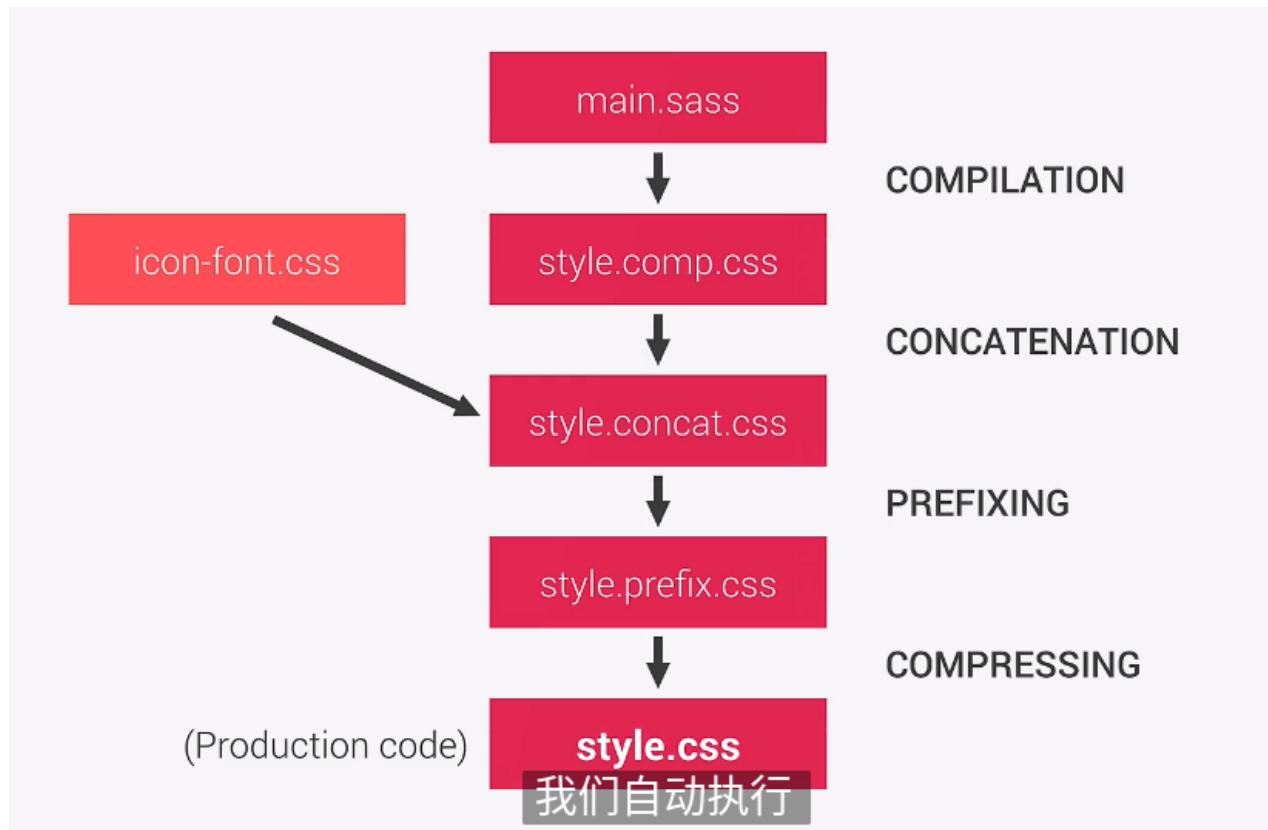
- 浏览器支持，检查网站：caniuse.com。对于旧的浏览器采用 graceful degradation，使用 `@supports`

```

@supports (-webkit-backdrop-filter: blur(10px)) or (backdrop-filter: blur(10px)) {
    -webkit-backdrop-filter: blur(10px);
    backdrop-filter: blur(10px);
    background-color: rgba($color-black, .3);
}

```

- Build process: 可以安装 `concat` 和 `autoprefixer` 和 `postcss-cli` 和 `npm-run-all`



```

1  {
2    "name": "natours",
3    "version": "1.0.0",
4    "description": "Landing page for natours",
5    "main": "index.js",
6    "scripts": {
7      "watch:sass": "node-sass sass/main.scss css/style.css -w",
8      "compile:sass": "node-sass sass/main.scss css/style.comp.css",
9      "concat:css": "concat -o css/style.concat.css css/icon-font.css css/style.comp.css",
10     "prefix:css": "postcss --use autoprefixer -b 'last 10 versions' css/style.concat.css -o css/style.prefix.css",
11     "compress:css": "node-sass css/style.prefix.css css/style.css --output-style compressed"
12   },
13   "author": "Jonas",
14   "license": "ISC",
15   "devDependencies": {
16     "autoprefixer": "^7.1.4",
17     "concat": "^1.0.3",
18     "node-sass": "^4.5.3",
19     "postcss-cli": "^4.1.1"
20   }
21 }
22

```

- Dev process

```

"watch:sass": "node-sass sass/main.scss css/style.css -w",
"devserver": "live-server",
"start": "npm-run-all --parallel devserver watch:css",

```

- 细节：选中文本

```

::selection {
  background-color: $color-primary;
  color: $color-white;
}

```

- 细节：媒体查询只支持屏幕，不支持打印

```
@mixin respond($breakpoint) {
  @if $breakpoint == phone {
    @media only screen and (max-width: 37.5em) { @content }; //600px
  }
  @if $breakpoint == tab-port {
    @media only screen and (max-width: 56.25em) { @content }; //900px
  }
  @if $breakpoint == tab-land {
    @media only screen and (max-width: 75em) { @content }; //1200px
  }
  @if $breakpoint == big-desktop {
    @media only screen and (min-width: 112.5em) { @content }; //1800
  }
}
```

- 细节：htm头部代码需要写，如果不写就不支持响应式

```
<html lang="en">
  <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">

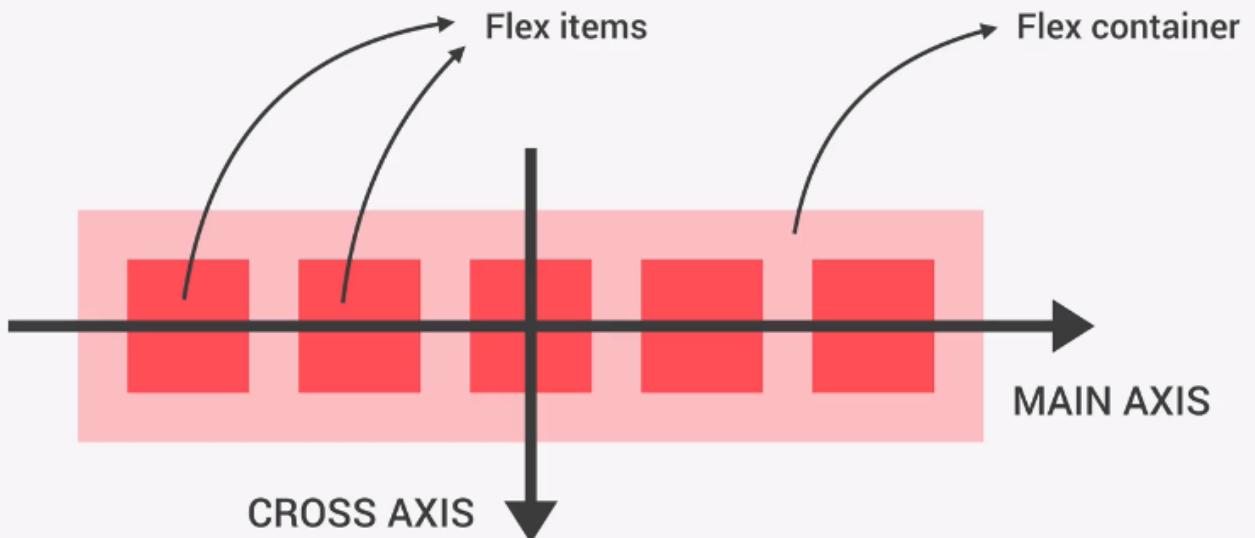
    <link href="https://fonts.googleapis.com/css?family=Lato:100,300,400,700,900" rel="stylesheet">
```

- 细节：有专门的选择器看设备是否支持触摸

```
//@include respond(tab-port) {
  @media only screen and (max-width: 56.25em),
        only screen and (hover: none) {
```

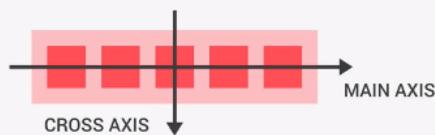
Flexbox

- 基础概念



`display: flex`

(`display: flex-inline`)



CONTAINER

- ① `flex-direction: row | row-reverse | column | column-reverse`
- ② `flex-wrap: nowrap | wrap | wrap-reverse`
- ③ `justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly`
- ④ `align-items: stretch | flex-start | flex-end | center | baseline`
- ⑤ `align-content: stretch | flex-start | flex-end | center | space-between | space-around`

ITEM

- ① `align-self: auto | stretch | flex-start | flex-end | center | baseline`
- ② `order: 0 | <integer>`
- ③ `flex-grow: 0 | <integer>`
- ④ `flex-shrink: 1 | <integer>`
- ⑤ `flex-basis: auto | <length>`

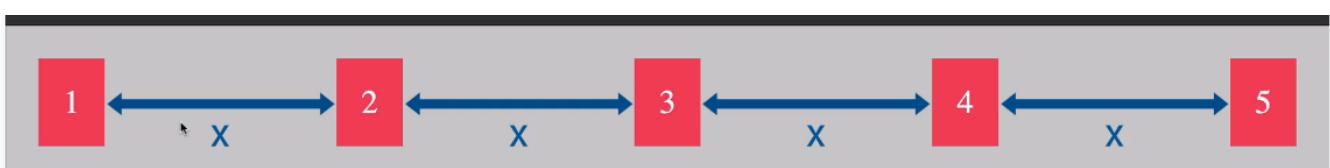
`flex: 0 1 auto | <int> <int> <len>`



- Flex-Container

- `justify-content:` 在主轴上的对齐方式

- Space-between



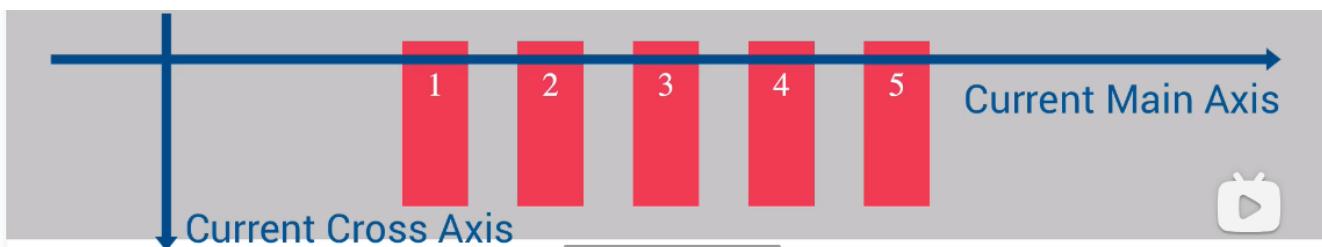
- Space-around



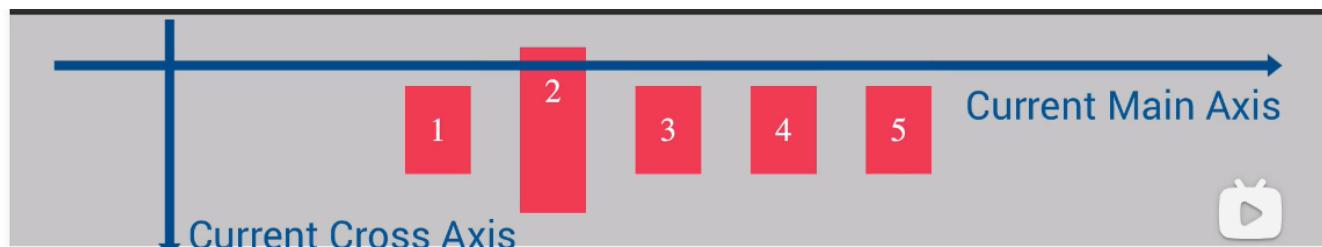
- Space-evenly



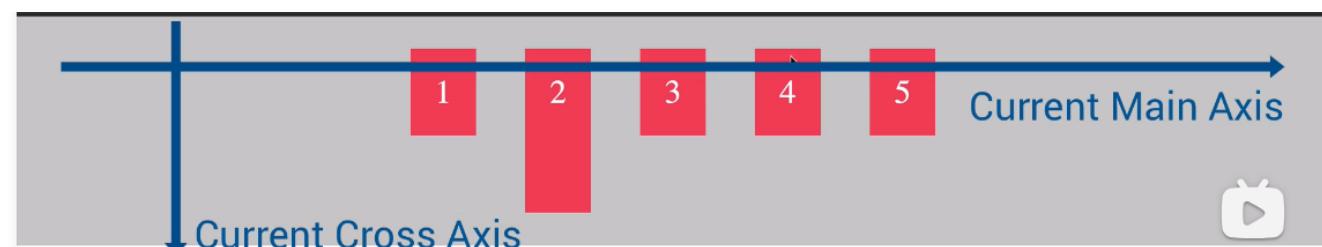
- Align-items
 - 默认值 stretch, 当第二个长度变长的时候其他也一样变



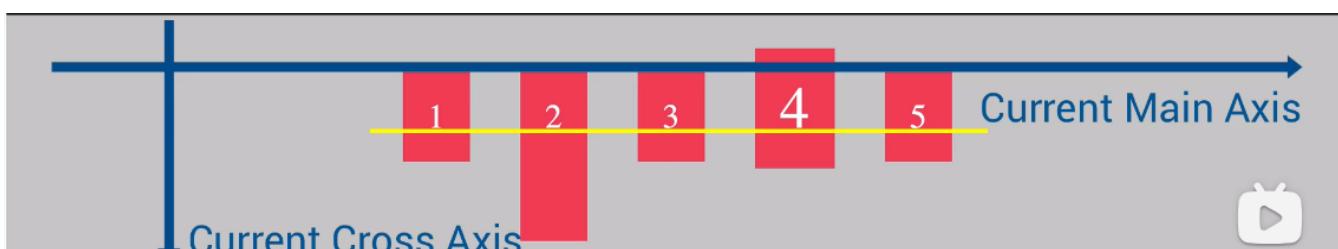
- Center



- Flex-Start: 顶部对齐

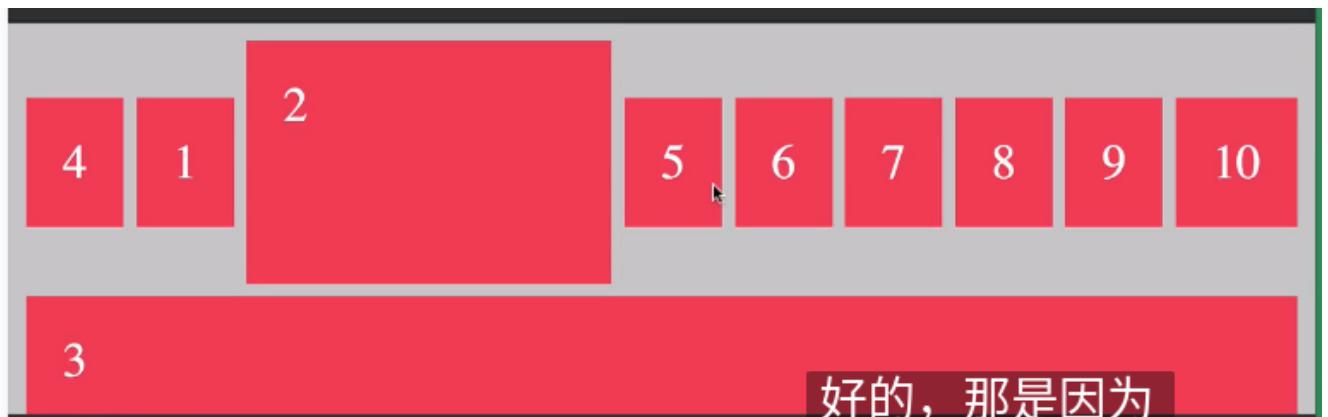


- Baseline: 当把第4个的字体变大的时候，所有div都根据文字对齐



- Flex-wrap
- Align-content

- Flex-start

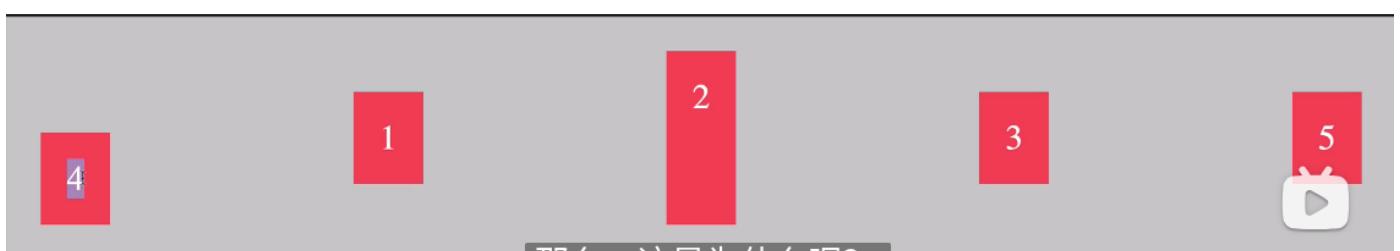


- Flex-Items: 处理多行的情况

- Align-self



- Order: 默认值是0，当给第四个设置负数的时候就排到了第一

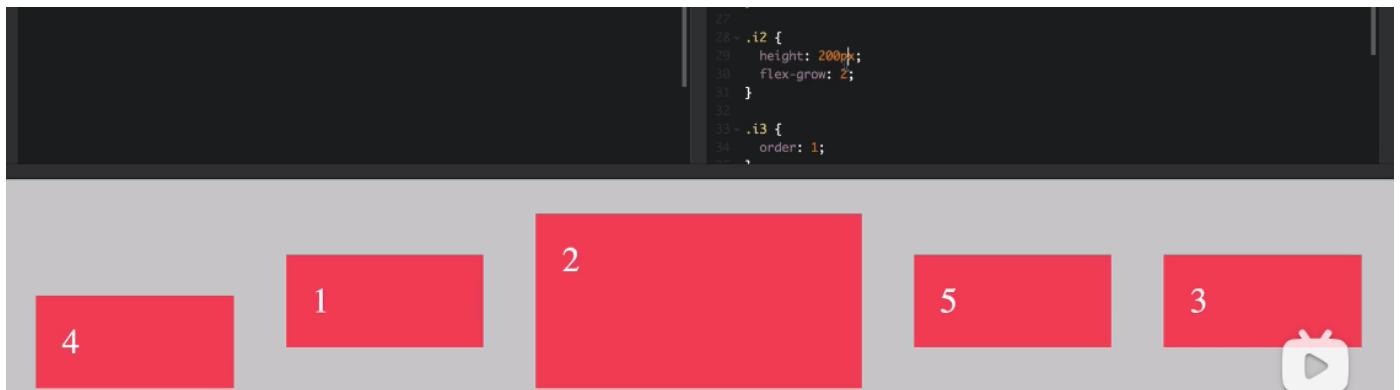


- Flex-grow, 可以直接写成 Flex

- 都设置成1，然后就会尽可能地占满整个空间



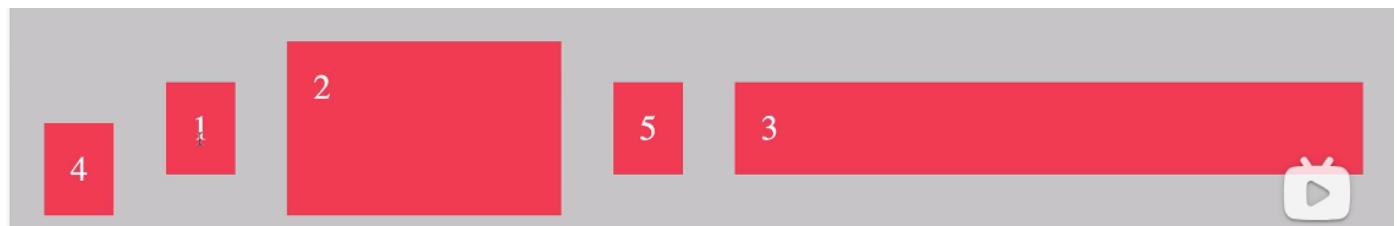
- 将其中一个设置成2，就会是其他的两倍



- 只设置其中一个为1，其他为0，那么就会填满剩下的空间



- Flex-basis
 - 将其中一个设置成20%，就会占整个container的20%；如果设置为300px，就会占300px空间



- Flex-shrink
 - 例如上面设置了 basis 是 300px，然后再设置 shrink 是 0 就会不允许缩放，默认值是 1 就是可以缩放

Project Trillo

- 初始化。设置body的背景色可以用 `min-height`

```
* {
  margin: 0;
  padding: 0;
}

*, *::before, *::after {
  box-sizing: inherit;
}

html {
  box-sizing: border-box;
  font-size: 62.5%; // 1rem = 10px, 10px/16px = 62.5%
}

body {
  font-family: 'Open Sans', sans-serif;
  font-weight: 400;
  line-height: 1.6;
}
```

```
body {
  font-family: 'Open Sans', sans-serif;
  font-weight: 400;
  line-height: 1.6;
  background-image: linear-gradient(to right bottom, var(--color-primary-light), var(--color-primary-dark));
  background-size: cover;
  background-repeat: no-repeat; }

  min-height: 100vh;
```

- CSS变量，`:root` 是所有元素的父元素，所以定义的变量都可以用

```
:root {
  --color-primary: #eb2f64;
  --color-primary-light: #FF3366;
  --color-primary-dark: #BA265D;
}
```

```
background-image: linear-gradient(to right bottom, var(--color-primary-light), var(--color-primary-dark));
background-size: cover;
```

- 常用svg网站：iconmoon.io，在这个网站上下载的svg有一个精灵文件 sprite.svg 和文件夹，使用如下方法引入。svg颜色的修改使用`fill`，可以使用`fill:currentColor`，代表当前元素或者父元素的颜色

```
<svg class="search__icon">
  <use xlink:href="img/sprite.svg#icon-magnifying-glass"></use>
</svg>
```

- 多个transform

```
        transform: scaleY(0);
        transition: transform .2s,
                    width .4s cubic-bezier(1,0,0,1) .2s;
    }

    &__item:hover::before {
        transform: scaleY(1);
        width: 100%;
    }
```

- `z-index` 只有在指定 `position` 的时候才生效
- flex中如果想让某个元素占满全部空间可以设置 `flex:1`，但是这样会导致这个元素的content也变长了，更好的办法是 `margin:auto`
- flex容器下面div下面的svg图片可能不能很好的垂直居中，可以设置div也为flex
- 动画持续播放

```
&:focus {
    outline: none;
    animation: pulsate 1s infinite;
}

@keyframes pulsate {
    0% {
        transform: scale(1);
        box-shadow: none;
    }

    50% {
        transform: scale(1.05);
        box-shadow: 0 1rem 4rem rgba(0,0,0,.25);
    }

    100% {
        transform: scale(1);
```

所以这将创建

- 给一个li前面加个箭头使用before，我们还可以用mask来做到改变background的颜色

```

&__item::before {
    content: "";
    display: inline-block;
    height: 2rem;
    width: 2rem;
    margin-right: .7rem;

    // Older browsers
    //background-image: url(../img/chevron-thin-right.svg);
    //background-size: cover;

    //Newer browsers - masks
    background-color: var(--color-primary);
    -webkit-mask-image: url(../img/chevron-thin-right.svg);
    -webkit-mask-size: cover;
}

}

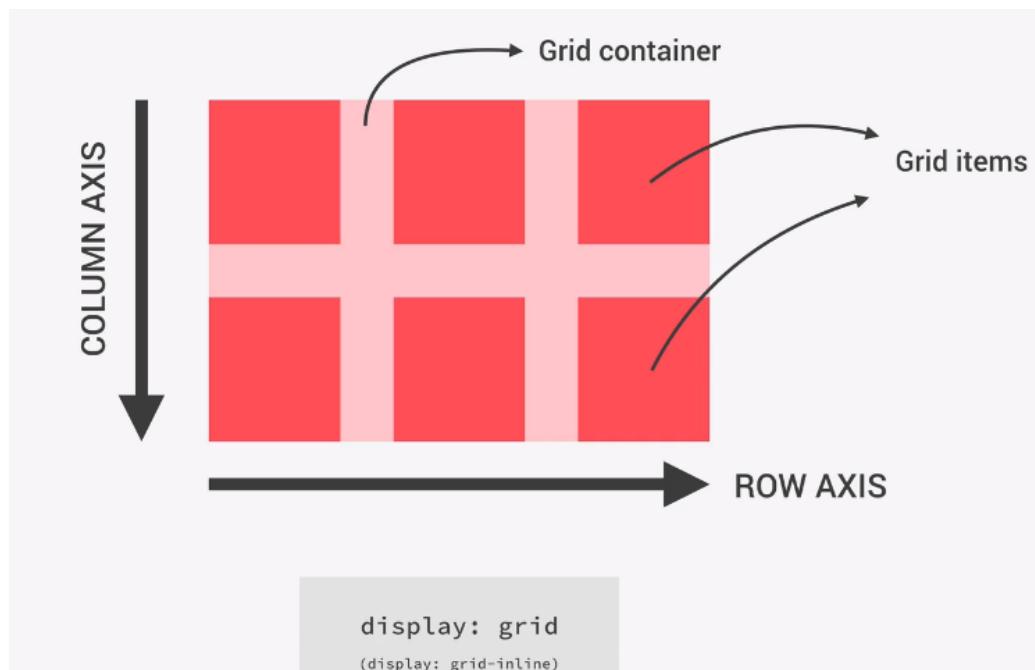
```

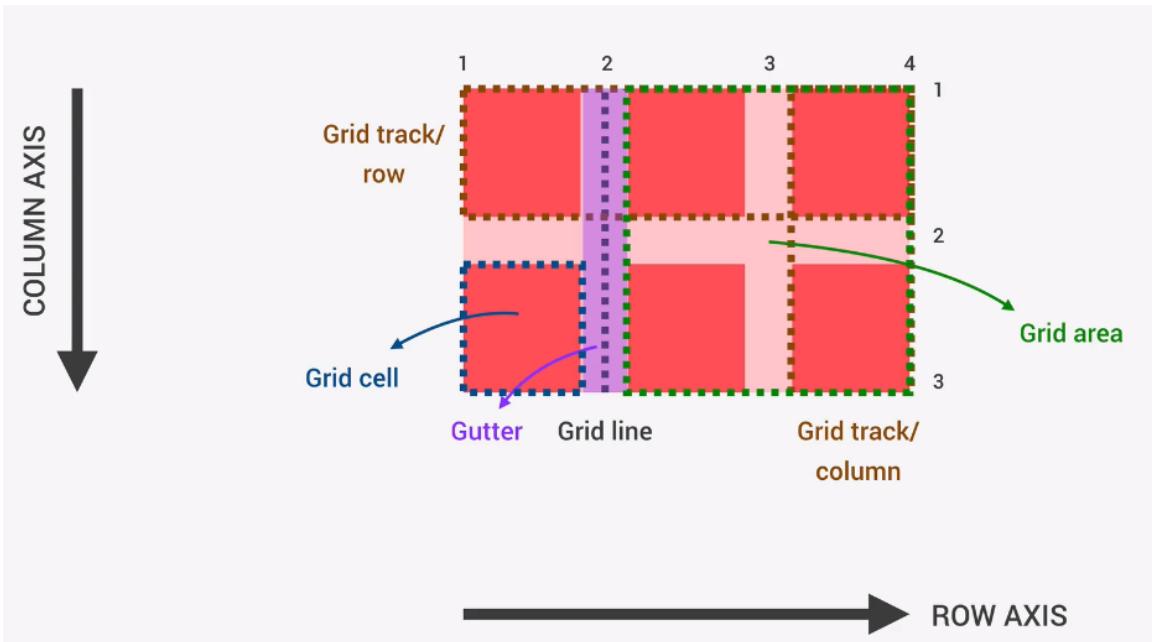
- 特殊字符 <https://css-tricks.com/snippets/html/glyphs/>，这些可以配合before after使用



Grid

- 基础
 - 有两个轴，不能修改





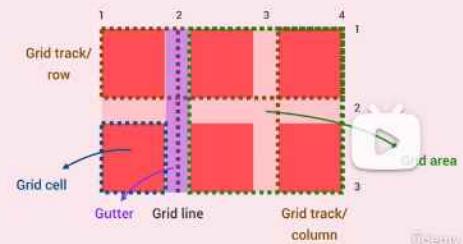
CSS GRID PROPERTIES OVERVIEW

CONTAINER

- ① grid-template-rows
grid-template-columns
grid-template-areas
 - ② grid-row-gap
grid-column-gap
 - ③ justify-items
align-items
justify-content
align-content
 - ④ grid-auto-rows
grid-auto-columns
grid-auto-flow
- grid-template
- grid-gap

ITEM

- ① grid-row-start
grid-row-end
grid-column-start
grid-column-end
 - ② justify-self
align-self
 - ③ order
- grid-row
- grid-column
- grid-area



- 设置grid的方法，例如2行3列：`display: grid; grid-template-rows: 150px 150px; grid-template-columns: 150px 150px 150px`，可以简写成 `repeat(2, 150px)`
- 给grid增加空隙：`grid-row-gap`
- 新的单位 `fr`，效果类似 `flex: 1`。所以三等分：`repeat(3, 1fr)`

HTML

```

1 <div class="container">
2   <div class="item item--1">1: Orange</div>
3   <div class="item item--2">2: Green</div>
4   <div class="item item--3">3: Violet</div>
5   <div class="item item--4">4: Pink</div>
6   <div class="item item--5">5: Blue</div>
7   <div class="item item--6">6: Brown</div>
8 </div>

```

CSS (SCSS)

```

4 margin: 30px auto;
5
6 display: grid;
7 grid-template-rows: repeat(2, 150px);
8 grid-template-columns: repeat(2, 150px) 1fr;
9
10 //grid-row-gap: 30px;
11 //grid-column-gap: 50px;
12 grid-gap: 30px;
13 }
14
15 .item {

```

- 设置item的位置，使用 `grid-row-start` 和 `grid-row-end`，其中的数字是看上面grid line 的数字。简写： `grid-row: 2 / 3`，还可以用 `grid-area`

HTML

```

1 <div class="container">
2   <div class="item item--1">1: Orange</div>
3   <div class="item item--2">2: Green</div>
4   <div class="item item--3">3: Violet</div>
5   <div class="item item--4">4: Pink</div>
6   <div class="item item--5">5: Blue</div>
7   <div class="item item--6">6: Brown</div>
8 </div>

```

CSS (SCSS)

```

20 padding: 10px,
21 font-size: 30px;
22 font-family: sans-serif;
23 color: white;
24 &--1 {
25   background-color: orangered;
26   grid-row-start: 2;
27   grid-row-end: 3;
28   grid-column-start: 2;
29   grid-column-end: 3;
30 }

```

```

HTML
1 <div class="container">
2   <div class="item item--1">1: Orange</div>
3   <div class="item item--2">2: Green</div>
4   <div class="item item--3">3: Violet</div>
5   <div class="item item--4">4: Pink</div>
6   <div class="item item--5">5: Blue</div>
7   <div class="item item--6">6: Brown</div>
8 </div>

SCSS (SCSS)
43 background-color: palevioletred;
44 }
45
46 &--5 {
47   background-color: royalblue;
48 /*grid-row: 1 / 2;
49 grid-column: 3 / 4;*/
50   grid-area: 1 / 3 / 2 / 4;
51 }
52
53 &--6 {
54   background-color: goldenrod;

```

- 如果想要扩展一个cell，只要上面的数字间隔不是1即可，但是会导致没有定位的cell挤出去产生一个新行，如果定位了就会被遮盖（如果就想overlap就指定好位置）。另外有一些trick，定位的时候可以是 `1 / span 2` 表示扩展两个cell，还有 `1 / -1` 表示扩展到最后（注意是显式网格的最后）
- 除了可以使用数字，还可以自己定义名字，中括号里可以有多个名字。然后数字都可以替换成名字。repeat里面设置的名字后面会自动加上一个数字

```

grid-template-rows: [header-start] 100px [header-end box-start]
200px [box-end main-start] 400px [main-end footer-start] 100px
[footer-end];

```

```

grid-template-columns: repeat(3, [col-start] 1fr [col-end]) 200px
[grid-end];

```

```

.header {
  grid-column: col-start 1 / grid-end;
}

```

- 第三种方法是给 grid-area 命名。另外使用 `.` 表示留空

```

grid-template-areas: "head head head head"
                    "box box box side"
                    "main main main side"
                    "foot foot foot foot";

```

```

.header {
  grid-area: head;
}

```

- 如果item大于grid定义的cell数，那么就会增加 implicit track。可以设置 `grid-auto-rows` / `grid-auto-columns` 来设置它的大小，另外可以设置 `grid-auto-flow: column` / `row` 来决定新增列是加上行上还是列上
- 对齐
 - Align-items
 - 默认是stretch
 - 设置center就是在cross-axis上居中，正常是垂直居中
 - Justify-items
 - 有items的属性都表示相对于这个cell
 - Justify-content / Align-content
 - 表示的是track在更大容器中的对齐方式
- 自动排序算法会尽量让cell保持顺序，因此可能出现空洞，如果想消除，可以在 `grid-auto-flow: dense`
- 设置grid长度的时候还有一些其他的单位
 - max-content: 尽量容纳所有的内容，且尽量不换行
 - min-content: 用更小的位置容纳内容，尽量不overflow
 - 可以使用 `minmax` 函数，这个函数确保返回值在输入的两个参数之间，例如 `minmax(150px, min-content)`
- repeat中的数量还有别的值
 - auto-fill，自动计算分的行数或者列数，没有cell的地方也会分出来
 - auto-fit，同上但是没有cell的部分会collapse（创建了但是宽度或高度为0）
 - 有个非常适合响应式的设置： `grid-template-rows: repeat(auto-fit, minmax(200px, 1fr))`

Project Nexter

- 设置网格
 - `row`，默认值是 `auto`，效果和 `min-content` 基本一致，可以直接用 `min-content`。行数一般没必要设置名称。一般甚至没必要设置 rows
 - `column`，这里使用8列分布，首先左边有一个sidebar，然后中间有8列使用minmax设置，然后为了让这个8列居中，所以加上两列 `1fr` 填充剩余空间。最好这里设置 `1fr` 为 `minmax(6rem, 1fr)` 留出一些空隙。

```
.container {  
    display: grid;  
    grid-template-rows: 80vh min-content 40vw repeat(3, min-content);  
    grid-template-columns: [sidebar-start] 8rem [sidebar-end full-start] 1fr [center-start] repeat(8,  
    [col-start] minmax(min-content, 14rem) [col-end]) [center-end] 1fr [full-end];  
}
```

- 如果想让图片fit到一个cell里，你可以给 `` 加一个 `<figure>` 的父元素，然后进行如下设置，注意 `object-fit` 必须在设置好 `width` 和 `height` 的时候才有效

```
&__img {  
    width: 100%;  
    height: 100%;  
    object-fit: cover;  
    display: block;  
}
```

- 设置 `background-image` 的时候，可以设置 `background-position: center`
- 当我们想要一个比较大的空隙的时候，我们可以用 `1fr`，这样可以较好适应响应式
- `::before / ::after` 的父元素可以用 `grid` 来排版
- 响应式设计
 - 图片可以设置 `max-width: 100%; max-height: 2.5rem`
 - `grid-template-columns` 之前是 `max-content` 导致不会换行，所以需要改成 `minmax(min-content, max-content)`。 `rows` 也尽量不要使用固定值，最好是用 `minmax` 和 `min-content` 的组合

随记

- Emmet中\$符号表示从1开始递增的数字，{}大括号表示的是标签中的文字内容（也可以使用\$），[]中括号里面写上其他的属性，一个中括号写一个属性
- img的alt可以用来SEO