1. **Styled-component声明全局样式：**

export const GlobalStyle = createGlobalStyle``

通过createGlobalStyle声明全局样式，然后在项目根文件中导入即可

function App() {

  return (

    <Provider store={store}>

      <HashRouter>

        <GlobalStyle></GlobalStyle>

        <IconStyle></IconStyle>

        { renderRoutes(routes) }

      </HashRouter>

    </Provider>

  )

}

1. **CSS样式最高优先级！Important**

例：

font-family: "iconfont" !important;

！important可以突破css引用的自然优先级，优先使用！Important标记的样式

1. **’>’选择器**

'>'代表后代第一层选择器

1. **Flex布局常用属性：**

**主轴方向：flex-direction**

**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。

column-reverse：主轴为垂直方向，起点在下沿。

**项目溢出换行：flex-wrap**

**flex-wrap**: nowrap | wrap | wrap-reverse;

nowrap（默认）：不换行。

wrap：换行，第一行在上方。

wrap-reverse：换行，第一行在下方。

**★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。

**★主轴对齐方式：justify-content**

**justify-content**: flex-start | flex-end | center | space-between | space-around;

flex-start（默认值）：左对齐

flex-end：右对齐

center： 居中

space-between：两端对齐，项目之间的间隔都相等。

space-around：每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。

**★交叉轴对齐方式：align-items**

**align-items**: flex-start | flex-end | center | space-between | space-around | stretch;

flex-start：交叉轴的起点对齐。

flex-end：交叉轴的终点对齐。

center：交叉轴的中点对齐。

baseline: 项目的第一行文字的基线对齐。

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。

**项目属性：**

**flex-grow**属性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。flex-grow: <number>

**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。flex-shrink: <number>

**flex-basis**属性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。flex-basis: <length> | auto;

**★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。flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]

**★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。align-self: auto | flex-start | flex-end | center | baseline | stretch;

1. **元素高度百分比设置无效的问题**

元素的高度默认是缺省值，所以特别要注意当高度设置为百分比时，父元素的高度要确定，否则设置无效，尤其是在height：100%时，最终可能会追溯到设置html，body的高度为100%

1. **display属性对布局带来的影响**

**块级元素：**每个块级元素通常都会独占一行或者是多行，可以对其单独设置高度,宽度以及对齐等属性。

**特点：**

块级元素会独占一行

高度，行高，外边距和内边距都可以单独设置

宽度默认是容器的100%

可以容纳内联元素和其他的块级元素

**行内元素：**不占有独立的区域，仅仅依靠自己的字体大小或者是图像大小来支撑结构。一般不可以设置宽度，高度以及对齐等属性。

**特点：**

和相邻的行内元素在一行上

高度和宽度无效，但是水平方向上的padding和margin可以设置，垂直方向上的无效

默认的宽度就是它本身的宽度

行内元素只能容纳纯文本或者是其他的行内元素（a标签除外）

#### **行内块级元素：**

和相邻的行内元素（行内块）在一行上，但是中间会有空白的间隙

默认的宽度就是本身内容的宽度

高度，行高，内边距和外边距都可以设置

1. **react-lazyload使用方法**

1.在图片的外面包裹一层LazyLoad，并设置placeholder属性为要懒加载时显示的内容（可以是图片或者其他）

2.向scroll的onscroll属性传入执行函数，当滑动的时候执行该函数，即传入forcecheck函数，作用是解除懒加载状态显示图片，实现手段是。。。有待考证

forcecheck

1. **占位容器的技巧**

容器的margin\padding使用百分比时是以父元素的width为基准的，所以'padding-top: 100%'就是父元素的width，同时不能使用margin的理由是如果用了margin容器就没有高度了，但是用padding有，因为默认的元素的高度是content的高度+padding的高度，如果用了margin，高度会受到挤压，反之padding则不会

1. **利用背景阴影做衬底的技巧**

background: linear-gradient(hsla(0,0%,43%,.4),hsla(0,0%,100%,0));

hsla(hue, saturation, lightness, alpha)

hue- 色相:范围0-360，其中0 (或 360) 为红色, 120 为绿色, 240 为蓝色

saturation - 饱和度：定义饱和度; 0% 为灰色， 100% 全色

lightness - 亮度：定义亮度 0% 为暗, 50% 为普通, 100% 为白

alpha - 透明度：定义透明度 0（透完全明） ~ 1（完全不透明）

linear-gradient(direction, color-stop1, color-stop2, ...)

direction：  用角度值指定渐变的方向（或角度）。

color-stop1, color-stop2,...：用于指定渐变的起止颜色。

1. **CSS动画的编写**

keyframes是专门用来写动画用的. "from" 和 "to"，等同于 0% 和 100%，0% 是动画的开始，100% 是动画的完成.

animation要规定至少两个属性：动画名称、动画时长

步骤：

1.编写keyframes

@keyframes myfirst

{

from {background:red;}

to {background:yellow;}

}

2.在animation属性调用keyframes

div

{

animation:myfirst 5s;

}

1. **CSS过渡属性transition使用：**

语法：

transition: property duration timing-function delay;

[property](https://links.jianshu.com/go?to=http://www.runoob.com/cssref/css3-pr-transition-property.html" \t "https://www.jianshu.com/p/_blank):规定应用过渡的 CSS 属性的名称(默认为all)。(可选)

[duration](https://links.jianshu.com/go?to=http://www.runoob.com/cssref/css3-pr-transition-duration.html" \t "https://www.jianshu.com/p/_blank):定义过渡效果花费的时间。默认是 0s。（必需）

[timing-function](https://links.jianshu.com/go?to=http://www.runoob.com/cssref/css3-pr-transition-timing-function.html" \t "https://www.jianshu.com/p/_blank):规定过渡效果的时间曲线。默认是 "ease"函数。(可选)

[delay](https://links.jianshu.com/go?to=http://www.runoob.com/cssref/css3-pr-transition-delay.html" \t "https://www.jianshu.com/p/_blank):规定过渡效果何时开始。默认是 0s.(可选)

[timing-function](https://links.jianshu.com/go?to=http://www.runoob.com/cssref/css3-pr-transition-timing-function.html" \t "https://www.jianshu.com/p/_blank)表：

| 值 | 描述 |
| --- | --- |
| linear | 规定以相同速度开始至结束的过渡效果，即匀速。（等于 cubic-bezier(0,0,1,1)）。 |
| ease | 规定慢速开始，然后变快，然后慢速结束的过渡效果（cubic-bezier(0.25,0.1,0.25,1)）。 |
| ease-in | 规定以慢速开始的过渡效果（等于 cubic-bezier(0.42,0,1,1)）。 |
| ease-out | 规定以慢速结束的过渡效果（等于 cubic-bezier(0,0,0.58,1)）。 |
| ease-in-out | 规定以慢速开始和结束的过渡效果（等于 cubic-bezier(0.42,0,0.58,1)）。 |
| cubic-bezier(n,n,n,n) | 在 cubic-bezier 函数中定义自己的值。可能的值是 0 至 1 之间的数值。 |

例子：

div

{

transition: width 2s;

}

1. **CSS transform属性**
2. **CSS xx:nth-child(y)属性**

规定xx元素的父元素的第y个子元素的属性

例：这里就是让包裹2的span标签字体变红

<style>

div:nth-child(2){

Color:red

}

</style>

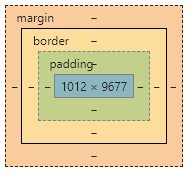
<div>

<span>1</span>

<span>2</span>

</div>

1. **offset：元素盒模型的宽度**
2. **box-sizing：**



border-box：宽高的范围是border+padding+content

content-box:宽高的范围是content

1. **单位vh、和vw**

vh vw是不包含页面滚动条的视窗宽度(innerwidth)，%包含了滚动条的宽度在里面了(outerwidth)

vw：1vw等于视口宽度的1%。

vh：1vh等于视口高度的1%。

1. **伪类和伪元素**

伪类： :hove、:activer、:nth-child(n)

伪元素： ::before、::after

其中:nth-child(n)规定了使用该样式的元素的父元素的第n个子元素使用该样式

当n=odd时，标志着奇数位使用该样式；当n=even时，标志着奇数位使用该样式

1. **Translate偏移时的相对对象**

无论是在动画中还是在react-transition-group中translate都是相对于初始位置做移动。

     也就是说如果你在20%的时候执行translate（50px,0）向右平移了50px

     在30%的时候执行translate（0,0）则会向左平移50px

     因为你的移动的相对对象永远是初始位置

1. **sass中父选择器&的妙用**

<div className="background">

<div className="background layer"></div>

  .background {

    position: absolute;

    left: 0;

    top: 0;

    width: 100%;

    height: 100%;

    z-index: -1;

    opacity: 0.6;

    filter: blur(20px);

    &.layer {

      background: ${style["font-color-desc"]};

      opacity: 0.3;

      filter: none;

    }

  }

**backgroud是公用样式，.&layer是同时声明background和layer的元素才享有的独有样式。**

1. **dangerouslySetInnerHTML**

react的dangerouslySetInnerHTML在一个dom里直接筛入一段纯html

格式：dangerouslySetInnerHTML={{ \_\_html: content }} content为元素显示的内容

**该操作是不安全的，会引发XSS攻击，所以他才叫做‘dangerouslySetInnerHTML’ 是有风险的  能不用则不用**

1. **clientWidth、offsetWidth、scrollWidth的区别**

**clientWidth =** width+左右padding

**offsetWidth =** width + 左右padding + 左右boder

**scrollWidth：**获取指定标签内容层的真实宽度（可视区域宽度+被隐藏区域宽度）。

1. **SVG**

首先需要定义一个<svg></svg>标签来作为svg图像的容器，后面可以在这个容器中画线画圆画正方形...

<svg width={radius} height={radius} viewBox="0 0 100 100" version="1.1" xmlns="http://www.w3.org/2000/svg">

    <circle r="50" cx="50" cy="50" fill="transparent"/>

    <circle r="50" cx="50" cy="50" fill="transparent"

   strokeDasharray={dashArray}strokeDashoffset={dashOffset}/>

</svg>

**viewBox：**一个用来适配svg内容和这个容器本身的工具，viewbox的大小是相对于svg的内容的，并不是相对于svg容器。先用viewbox将将好装下了svg的内容 ，再由svg自动适配viewbox，使得viewbox能正好完整展示在svg中。具体过程就是一个可变大小的框子（viewbox），去框svg的内容，然后svg容器又是个框子，不过是固定大小的，这个时候viewbox调整自己的大小（svg的内容与viewbox同步变大变小），正好能套进svg容器中，O了！

**viewBox**="x, y, width, height"

**x**:左上角横坐标，

**y**:左上角纵坐标，

**width**:宽度，

**height**:高度

<circle r="50" cx="50" cy="50" fill="transparent"

        strokeDasharray={dashArray}

        strokeDashoffset={dashOffset}/>

**Circle：圆**

**fill：**填充色

**r:**半径

**cx：**x坐标

**cy：**y坐标

**transparent：**透明色

**strokeDasharray：**用于描绘出虚线边框  格式：strokeDasharray={线长度，缺口长度， 线长度，缺口长度...}strokeDasharray内容的数值个数必须是偶数

个否则自动复制原内容个数自动翻倍

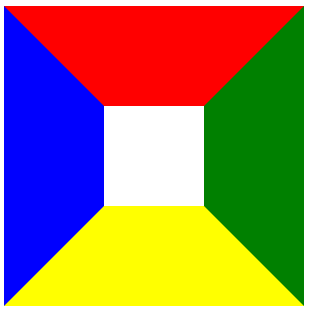
**例：**321 -> 321321  这样321321就是一个循环周期了

**strokeDashoffset：**起始点的偏移量  偏移方向：正左负右

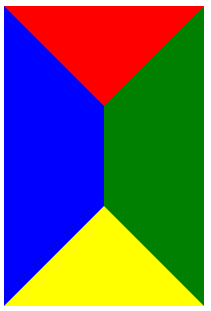
<https://segmentfault.com/a/1190000009226427>

<https://www.cnblogs.com/daisygogogo/p/11044353.html>

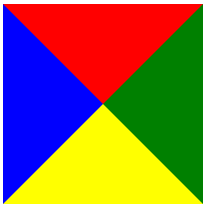
1. **CSS画三角形**



**原理：**边框的的交界处其实是一条斜线，如果不设置宽度，那么上下两个边会变成三角形，如果想要看上去明显，放大边框宽度，缩小边框长度就能看出来了。



同理高度为0左右两边也是三角形了。

****

那么想要单独获取三角形只要吧其他三边设置为透明就可以了

.trangle{

　　width: 0px;

　　height: 0px;

　　border: 100px solid #000;

　　border-top-color: red;

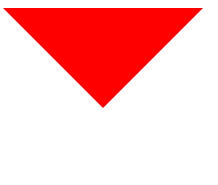
　　border-bottom-color: transparent;

　　border-left-color: transparent;

　　border-right-color: transparent;

}

<div class="trangle"></div>



1. **实现水平垂直居中的方法：**

**常规居中方法无法实现float元素居中**

**水平居中：**

1. 父元素设置text-align:center，让子元素水平居中（适用于子元素为行内元素或者类行内元素）
2. margin:0 auto（适用于块级元素）

**垂直居中：**

1. 添加等值的padding-top和padding-bottom（行内元素或类行内元素）
2. 设置line-height的值等于height（行内元素或类行内元素）
3. Flex直接设置轴居中（行内元素或类行内元素）
4. 设置top:50%,margin-top为‘-自身高度的一半’或者直接translateY(-50%),当translateY(-50%)的计算结果有小数时会造成元素模糊，这个时候需要在父元素添加样式transform-style:preserve-3d.**就算是浮动元素的居中也可以是通过上述方法实现。**（块级元素）

**Margin:auto auto水平垂直居中：**

子元素absolute布局，且top、bottom、left、right均为0，即可实现水平垂直居中。

1. **CSS优先级：**

!important权重最高 10000

行内样式 1000

ID选择器 100

类、伪类、属性选择器 10

元素、伪元素选择器 1

1. **Position:absolute的问题**

Position:absolute的父元素的position值一定不能是static。

1. **弹性布局（flex）适用场景**

弹性布局适合于移动前端开发，在Android和ios上也完美支持。

1. **满屏品字布局**
2. **常见兼容性问题**
3. **文字溢出用省略号表示**

**overflow**:visible|hidden|scroll|auto|inherit

visible:默认值。内容不会被修剪，会**呈现在元素框之外**。

hidden：内容会被修剪，并且其余内容是**不可见**的。

scroll：内容会被修剪，但是浏览器会**显示滚动条**以便查看其余的内容

auto：如果内容被修剪，则浏览器会显示**滚动条**以便查看其余的内容

inherit：规定应该从父元素**继承** overflow 属性的值

**text-overflow**:clip|ellipsis|string

Clip:隐藏

Ellipsis:省略号

String:自定义显示内容