# 复习

过渡：
transition:porperty duration timing-function delay;
transition-porperty:all/width/width,height,background-color;
transition-duration: s/ms;
transition-timing-function:ease/ease-in/ease-out/ease-in-out/linear/贝塞尔曲线;
transition-delay:s/ms;

特点：需要伪类或者js事件触发
不能自动运行，需要触发一次运行一次
只能支持两种状态的变化（简单动画）

transform
2d变形：
位移
translate(x,y)
translateX()
translateY()
旋转
rotate()
缩放
scale()
scaleX(2)
scaleY(0.5)
倾斜
skew(x,y)
skewX()
skewY()

3d变形：
位移
translateZ()
translate3d(x,y,z)
旋转
rotateX()
rotateY()
rotateZ()

rotate3D(1,0,-1,deg)
缩放
scaleZ()
scale3D()

transform-origin:left/center/right/100px/50% top/center/bottom/100px/50%;

perspective:600px-2000px;
transform-style:preserve-3d;

关键帧动画：
创建关键帧：
@keyframes ani-name{
0%{

```
        }
        ...
        50%{

        }
        ...
        100%{

        }
    }
```

调用：

```
    animation:name duration timing-function delay iteration-count direction fill-mode;
         animation-name
         animation-duration
         animation-timing-function
         animation-delay
         animation-iteration-count:3/infinite;
         animation-direction:alternate;
         animation-fill-mode:forwards;
```

特点：
可以自动运行
可以指定运行的次数，可以无限次运行
可以设置多个状态，实现复杂动画

# animate动画库使用 "https://animate.style/"

1、引入库文件：

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.css">
```

2、使用：

```
<h1 class="animate__animated animate__fadeInLeft">闪烁</h1>
```

# 弹性盒

- css3中的一种新的布局方式，弹性盒布局，弹性盒布局提供一种更加有效地方式规定盒子中的元素的排列方式，对齐方式以及空白空间的分配。
  1，设置为弹性盒容器
  display：flex；设置为弹性盒，元素本身保留块级特点
  display：inline-flex；设置为弹性盒，元素本身保留行内特点
  2，设置主轴方向
- 主轴方向决定了项目的排列方向
  flex-direction：
  -row 默认 从左到右
  -row-reverse 从右到左
  - column 从上到下
  - column-reverse 从下到上
  3，设置项目主轴上的对齐方式
- 对齐方式跟主轴有关，以下参数以主轴为row为例
  justify-connect：
  - flex-start；
  -flex-end：主轴结束
  -center 居中
  -space-around 每个项目前后空白相同，两个项目中间有双倍空白
  -space-between 每两个项目之间的空白相同，主轴的开始和结束位置没有空白
  -space-evenly 主轴开始和结束以及两个项目之间的空白都相同
  4，设置项目在交叉轴上的对齐方式
  align-items：
  -stretch 拉伸，如果项目没有固定高，被拉伸占满交叉轴
  -flex-start 交叉轴顶部对齐
  -flex-end 交叉轴底部对齐
  -center 交叉中间对齐
  -baseline 第一行字的基线对齐
  5，设置项目换行
  flex-wrap：

-nowrap 不换行
-wrap 换行
-wrap-reverse 换行，行排序
6,设置多行在交叉轴上的对齐
align-content：
-stretch拉伸
-flex-start 顶部对齐
- flex-end 底部对齐
-space-around 每一行的前后空白相同，两行之间的有双倍空间
-space-between 每行之间的空白相同，主轴的开始和结束位置没有空白
-space-evenly 交叉轴的开始和结束以及两行之间的空白都相同

# 项目属性

1，设置项目排序
order：n；默认值都为0，数值越大，排序越靠前

2，设置项目的放大比例
flex-grow：n；默认值为0表示不放大，数值越大表示放大值越多

3，设置项目缩小比例
flex-shrink：n；默认值为1，0表示不缩小，数值越大缩小越多，距离不够等比例缩小
4，设置单个项目在交叉轴上的对齐方式
align-self：
-stretch 拉伸，如果项目没有固定高，被拉伸占满交叉轴
-flex-start 交叉轴顶部对齐
-flex-end 交叉轴底部对齐
-center 交叉轴中间对齐
-baseline 第一行字的基线对齐

# 计算函数

width：calc（（100px+200）* 2 / 10）;

- 支持加减乘除运算
- 运算符前后保留空格
- 可以（）提升计算优先度

# less 预处理

- less是css一种预处理语言
- less提供一套特殊的语法，按照特定的语法编写样式代码 ，再用less所提供的编译器，编译成css文件提供项目上的使用

# less的编译

- 安装 Easy LEss
- 编写less文件--保存--生成.css文件

# 语法

## 1、变量

```
@m-color:gold;
@le:left;
.box{
    width: 100px;
    height: 100px;
    background-color: @m-color;
    border-@{le}:10px solid black;
}
```

编译后：

```
.box{
    width: 100px;
    height: 100px;
    background-color: gold;
    border-left:10px solid black;
}
```

## 2、注释

```
    /* css的注释，编译后会保留在css文件中 */

    // less注释，编译后不会存在css文件中
```

编译后：

```
/* css的注释，编译后会保留在css文件中 */
```

# 3、导入

```
@import "reset.css";  /* 导入css文件，编译后保留的是这个导入语法本身 */
@import "reset";  /* 导入less文件，编译后是把less中的代码编译并且复制到目标文件中 */
```

编译后：

```
@import "reset.css";

*{
    margin:0;
    padding:0;
}
ul,ol{
    list-style:none;
}
...
```

# 4、选择器嵌套

```
.header{
    line-height: 50px;
    background-color: #333;
    .left{
        float: left;
        li{
            float: left;
            margin-left: 10px;
            a{
                color: #333;
                text-decoration: none;
                &:hover{
                    color: orange;
                }
            }
        }
    }
    .right{
        float: right;
        &::after{
            content: '';
        }
    }
}
```

编译后：

```
.header {
  line-height: 50px;
  background-color: #333;
}
.header .left {
  float: left;
}
.header .left li {
  float: left;
  margin-left: 10px;
}
.header .left li a {
  color: #333;
  text-decoration: none;
}
.header .left li a:hover {
  color: orange;
}
.header .right {
  float: right;
}
.header .right::after {
  content: '';
}
```

# 5、混入

## 基本混入

```
.size{
    width: 100px;
    height: 100px;
}
.box1{
    .size;
}
.box2{
    .size;
}
```

编译后：

```
.size {
  width: 100px;
  height: 100px;
}
.box1 {
  width: 100px;
  height: 100px;
}
.box2 {
  width: 100px;
  height: 100px;
}
```

## 带参数的混入

```
.size1(@w,@h){  /* @w,@h两个形参 */
    width: @w;
    height: @h;
}
.box3{
    .size1(100px,200px); /* 调用时传入实参 */
}
.box4{
    .size1(150px,250px);
    background-color: red;
}
```

编译后：

```
.box3 {
  /* @w,@h两个形参 */
  width: 100px;
  height: 200px;
  /* 调用时传入实参 */
}
.box4 {
  /* @w,@h两个形参 */
  width: 150px;
  height: 250px;
  background-color: red;
}
```

## 带参数，带默认值的混入

```
.size2(@w:100px,@h:100px){
    width: @w;
    height: @h;
}
.box5{
    .size2;
}
.box6{
    .size2(200px);
}
.box7{
    .size2(@h:200px);
}
```

编译后：

```
.box5 {
  width: 100px;
  height: 100px;
}
.box6 {
  width: 200px;
  height: 100px;
}
.box7 {
  width: 100px;
  height: 200px;
}
```

## @arguments 所有参数

```
.box-shadow(@h,@v,@blur,@color){
    -webkit-box-shadow:@arguments;
    -moz-box-shadow:@arguments;
    -o-box-shadow:@arguments;
    -ms-box-shadow:@arguments;
    box-shadow:@arguments;
}
.box8{
    .box-shadow(1px, 1px, 10px, red);
}
```

编译后：

```
.box8 {
  -webkit-box-shadow: 1px 1px 10px red;
  -moz-box-shadow: 1px 1px 10px red;
  -o-box-shadow: 1px 1px 10px red;
  -ms-box-shadow: 1px 1px 10px red;
  box-shadow: 1px 1px 10px red;
}
```

# 9、继承

```
.list{
    list-style: none;
}
.list1{
    &:extend(.list);
    background-color: yellow;
}
```

编译后：

```
.list,
.list1 {
  list-style: none;
}
.list1 {
  background-color: yellow;
}
```

# 10、计算

```
.box9{
    width: 100% - 200px;
    height:1000px / 5;
}
```

编译后：

```
.box9 {
  width: -100%;
  height: 200px;
}
```