**H5C3**

**Day 01**

**1.关于浏览器私有前缀：**

谷歌、苹果浏览器：-webkit-

火狐浏览器：-moz-

IE浏览器：-ms-

欧朋浏览器：-o-

/\*谷歌浏览器和safari浏览器的前缀 -webkit-\*/

-webkit-transform: rotate(45deg);

/\*火狐浏览器的前缀 -moz-\*/

-moz-transform: rotate(45deg);

/\*ie浏览器的前缀 -ms-\*/

-ms-transform: rotate(45deg);

/\*opera浏览器的前缀 -o-\*/

-o-transform: rotate(45deg);

/\*规范化后的写法\*/

transform: rotate(45deg);

transition: all 1s; transition过渡 all 所有的属性 1s 过渡的时间

**浏览器前缀的存在, 主要是为了兼容各大版本的浏览器**

1. **选择器模块**
2. **基本选择器：**

通配符选择器 \* \* {......}

标签选择器 p {......}

类选择器 .one {.....}

id选择器 #id {.....}

1. **关系选择器：**

子代选择器： ul > li

后代选择器： ul li

相邻选择器： .now + li（找到下一个兄弟元素, 且这个兄弟元素必须是+后面的元素）

兄弟选择器： .now ~ li（找到后面所有的兄弟元素, 再判断类型是不是 li）

.now ~ \*（找到后面的所有兄弟元素）

1. **属性选择器：**

E[attr] ====== >只要有 attr 属性, 就会被选中

E[attr = “value”] ===== >有 attr 属性, 且值为 value

E[attr ^= “value” ] ====== > 有 attr 属性, 且值以 value 开头

E[attr $= “value”] =======> 有attr属性, 且值以 value 结尾

E[atttr \*= “value”]=======> 有 attr 属性, 且值包含 value

1. **伪类选择器：**
2. **child系列==================================**
   1. :first-child ======== li 是父元素ul 的第一个子元素（在一些特定的父子级关系里插入别的标签，在制定第一个时是无效的，比如在ul里插入p是错误的，也是无效的 但是在nav 里可以制定）
   2. :last-child ======== li 是父元素ul 的最后一个子元素
   3. :nth-child(3) ======== li 是父元素ul 的第三个子元素
   4. :nth-last-child(3)======== li 是父元素ul 的倒数第三个子元素
   5. :nth-child(odd) ======== 奇数
   6. :nth-child(even) ======== 偶数
   7. :nth-child(2n) ========= 偶数
   8. :nth-child(2n+1) ========= 奇数
   9. :nth-child(-n+x) ========== 前x 个被选中
   10. :nth-last-child(-n+y) ======= 最后y 个被选中
3. **of-type系列(了解)=================================**
   1. :nth-of-type(3) =====> 先找到所有该类型的元素, 再进行编号
4. **其他伪类选择器===================================**
   1. :focus 查找获取到焦点的文本框
   2. :checked 获得选中的checkbox
   3. :disabled 获得不可用的框（可选到有disable属性的表单，并且有相应的改变）
   4. :enabled获得可用的框
   5. :not(selector)选择不匹配selector的那些元素
   6. :target 获取当前活跃的锚链接
   7. :empty (选中没有内容和文本的)
5. **伪元素选择器 ==============================================**
   1. **::before （hover 写在before前面）**

必须设置 content 属性，默认是行内元素，需要制定display：block 或者 position：absolute

E::before :在元素**子节点**的最前面添加一个内容。

* 1. **::after**

必须设置 content 属性，默认是行内元素，需要制定display：block 或者 position：absolute

E::before :在元素**子节点**的最后面添加一个内容。

* 1. **::first-letter ========= :获取元素的第一个字符（用左浮动 有首字下沉的效果）**
  2. **::first-line ========= 获取元素的第一行**
  3. **::placeholder ========= 占位元素**
  4. **::selection ========= 获取选中的元素**

**设置 css3 盒子模型, 盒子宽度就是设置的 width, 如果设置padding和border自动内减**

**box-sizing: border-box;**

**3.文字阴影 text-shadow:文字阴影**

语法： text-shadow：水平偏移 垂直偏移 羽化大小 颜色

**4.边框阴影 box-shadow:边框阴影**

语法： box-shadow：水平偏移 垂直偏移 羽化大小 （外阴影） 颜色 （内阴影inset）========== 默认是外阴影

5.让所有属性实现过渡动画, 需要有两个状态 ===== **transition: all 1s;**

6.translateY 让盒子在原有位置的基础上, 在垂直方向上位移, 正方向往下

**transform: translateY(-10px)**

1. box-sizing: border-box;
   1. 标准盒子模型 盒子的大小, 宽度 = border + padding + content;

**box-sizing: content-box;**

* 1. Css3盒子模型 盒子的大小 宽度 = 设置的 width, 如果有border, 或者 padding 自动内减

**Day02**

CSS3背景

**background-size 用户设置背景图片的尺寸大小**

注意：**这两种设置方式会导致图片失真。**

（1）background-size:设置背景图片的大小\*/

background-size: 600px 400px;

/ （2）百分比是相对于盒子自身的宽度和高度 \*/

background-size: 100% 100%;

**不失真的设置方式（等比例缩放**）

（1）containe保证等比例缩放,但是会出现留白\*/

background-size: contain;

（2）cover保证等比例缩放,并且不会留白，但是出现有一部分图片不显示\*/

background-size: cover;

**background-clip ：** 设置背景区域的大小

盒子的背景区域是整个盒子，包括边框和padding\*/

默认值，设置背景区域包括了边框 background-clip: border-box;

背景区域只包含padding和content background-clip: padding-box;

背景区域只包含content background-clip: content-box;

**background-origin ：** 设置背景图片的原点的位置，默认是padding的地方开始

background-image: url(images/bg.jpg);

/\*设置原点从border开始\*/ background-origin: border-box;

/\*设置原点从padding开始,默认值\*/ background-origin: padding-box;

/\*设置原点从content开始\*/ background-origin: content-box;

**多重背景**

background设置背景的时候，可以设置多个背景图片，使用逗号隔开。注意颜色只能设置一次，并且通常来说，颜色都是在最后面进行设置。

background是一个合写的属性，如果在background之前设置了background相关的样式，会被覆盖掉。

background: url(images/mn1.jpg) no-repeat top left,

url("images/mn2.jpg") no-repeat right bottom, pink;

# CSS3渐变

1.线性渐变 ===== linear-gradient指沿着某条直线朝一个方向产生的渐变效果。

**注意：渐变实际上相当与一张图片，因为需要加给background-image才会生效**

**渐变的两个要求：有区间，有颜色变化。**

1.最简单的渐变

background-image: linear-gradient(red, green);

2.设定渐变的方向

background-image: linear-gradient(to right, red, green);

3.也可以设定渐变的角度

background-image: linear-gradient(45deg, red, green);

4.设定渐变的范围

background-image: linear-gradient(to right, red 20%, green 80%)

5.每一个区间表示渐变颜色的范围

background-image: linear-gradient(to right, red 20%, green 20%)

2.径向渐变 ===== radial-gradient指从一个中心点开始沿着四周产生渐变效果

核心：圆的大小，圆心的位置，颜色

1. 最简单的渐变

background-image: radial-gradient(red, green);

2. 指定圆的半径和圆心

background-image: radial-gradient(200px at center, red, green);

3. 指定椭圆

background-image: radial-gradient(200px 80px at center, red, green);

4. 指定范围

background-image: radial-gradient(200px at center, green 50%, red 50%);

1. 过渡
   1. 过渡的属性
      1. transition-property： 设置过渡属性，也可以是width,height

transition-property:all;

* + 1. transition-duration: 设置过渡时间

transition-duration:1s;

* + 1. transition-delay： 设置过渡延时

transition-delay:2s;

* + 1. transition-timing-function: 设置过渡的速度

（linear，ease，ease-in，ease-out，ease-in-out， steps(10)）

transition-timing-function:linear;

1. 属性合写

属性 时间 延时 速度 ========= transition: width 1s 3s linear;

**过渡的注意点：**

过渡必须要有两个状态的变化。

过渡可以写在A状态，也可以写在B状态，但是如果写在B状态，那么回来的时候，就没有过渡效果了。

# 2D转换

transform: 转换，是CSS3最具颠覆性的几个特性之一，既可以用于2D转换，也可以用于3D转换。

transform：2D转换，元素在平面上实现移动、旋转、缩放、斜切等操作

（1）scale缩放

transform: scaleX(0.5); 让宽度变化

transform: scaleY(0.5); 让高度变化，注意不能写多个transform，不然会覆盖。

transform: scale(0.5); 让宽度和高度同时变化

**注意：**

1.- scale接收的值是倍数，因此没有单位

2.- scale可以是一个值，如果是一个值，不是说仅仅缩放宽度，高度也会等比例的缩放。

3.- 可以通过transform-origin设定旋转原点

1. rotate旋转

transform: rotate(360deg);//旋转360度

transform: rotate(-360deg);//逆时针旋转360度

1. translate平移

transform: translateX(100px);

transform: translateY(100px);

transform: translate(100px, 100px);

transform: translate(50%, 50%);

**注意：**

- translate的值可以是px，也可以是百分比，如果是百分比，那么参照的是自身的宽高。

- translate移动的元素并不会影响其他盒子，类似于相对定位。

1. skew斜切(变形)

/\*在水平方向倾斜30deg\*/ ======== transform: skewX(30deg);

/\*在垂直方向倾斜30deg\*/ ======== transform: skewY(30deg);

1. transform-origin转换原点

通过transform-origin可以设置转换的中心原点。

transform-origin: center center;

transform-origin: 40px 40px;

1. 转换合写问题

transform:translateX(800px) scale(1.5) rotate(360deg) ;

1. transform属性只能写一个，如果写了多个会覆盖

2. transform属性可以连写，但是顺序对效果影响的，因为它会在第一个效果的基础上执行第二个效果，然后执行第三个效果（通常会把rotate放后面）

3. 如果对transform进行过度效果的时候，初始状态和结束状态一一对应

# **3D转换**

坐标轴

1. perspective透视

通过**perspective**属性，可以定义3D 元素距视图的距离，单位是px。

注意：当为元素定义 perspective 属性时，其子元素会获得透视效果，而不是元素本身**。**

## （2）rotate旋转

transform: rotate(45deg); 让元素在平面2D中旋转

transform: rotateX(45deg); 让元素沿着X轴转45度

transform: rotateY(45deg); 让元素沿着Y轴转45度

transform: rotateZ(45deg); 让元素沿着Z轴转45度

（3）translate平移

沿着X轴的正方向移动45px ======= transform: translateX(45px);

沿着Y轴的正方向移动45px ======= transform: translateY(45px);

沿着z轴的正方向移动45px ======= transform: translateZ(45px);

1. transform-style **这个属性只能给父元素添加。**

transform-style与perspective的区别

透视：透视只是相当于设置了一个距离，辅助我们查看3D效果的工具，

preserve-3d:给父盒子添加，让子元素保留3D的位置，说白了，只有设置了preserve-3d，这个元素才能被称之为3d元素

一个3d元素可以没有perspective，但是不能没有transform-style

动画

**动画与过渡的区别**

1. 过渡必须触发，需要两个状态的改变。

2. 动画可以一直运行下去，不需要触发。实现效果与过渡差不多

**使用一个动画的基本步骤：**

1.通过@keyframes指定动画序列

2.通过百分比或者from/to将动画分割成多个节点

3.在各个节点中分别定义样式

4.通过animation将动画应用于相应的元素

animation详解 ====== animation是一个复合属性，一共有8个参数

animation-name:动画名称，由@keyframes定义的

animation-duration：动画的持续时间

animation-timing-function：动画的过渡类型

animation-delay：动画的延迟时间

animation-iteration-count：动画的循环次数

animation-direction：设置动画在循环中是否反向运动

animation-fill-mode：设置动画时间之外的状态

animattion-play-state:设置动画的状态。

动画库的使用

字体图标

我们经常把网页常用的一些小的图标，做成精灵图，然后通过background-position去调整位置，但是这个需要引入图片，并且图片大小改变之后如果失真。在CSS3中可以使用字体图片，即使用图标跟使用文字一样。

优点：

1、将所有图标打包成字体库，减少请求；

2、具有矢量性，可保证清晰度；

3、使用灵活，便于维护；

弹性布局(伸缩布局) 布局：其实就是调整元素在水平和垂直方向上的布局方式。

当给一个盒子设置了display：flex之后，这个盒子就有了**主轴 和侧轴** 的概念。

主轴：Flex容器的主轴主要用来配置Flex项目，默认是水平方向

侧轴：与主轴垂直的轴称作侧轴，默认是垂直方向

方向：默认主轴从左向右 ，默认侧轴从上到下

（1）flex-direction ===== 主要是用来调整主轴的方向的，默认是水平方向

了解即可，一般来说，很少调整主轴的方向。

row：主轴方向为水平向右

column：主轴方向为竖直向下

row-reverse:主轴方向为水平向左

column-reverse:主轴方向是竖直向上。

## justify-content(\*+) ===== **主要用来设置主轴方向的对齐方式**

flex-start: 弹性盒子元素将向起始位置对齐

flex-end: 弹性盒子元素将向结束位置对齐。

center: 弹性盒子元素将向行中间位置对齐

space-around: 弹性盒子元素会平均地分布在行里

space-between: 第一个贴左边，最后一个贴右边，其他盒子均分，保证每个盒子之间的空隙是相等的。

1. align-items(\*+) ====== **用于调整侧轴的对其方式**

flex-start： 元素在侧轴的起始位置对其。

flex-end： 元素在侧轴的结束位置对其。

center： 元素在侧轴上居中对其。

stretch： 元素的高度会被拉伸到最大（不能给死高度）。

1. flex-wrap ===== **控制flex容器是单行或者多行,默认不换行**

nowrap： 不换行（默认），会压缩子盒子的宽度。

wrap： 当宽度不够的时候，会换行。

1. **align-content ======= 用来设置多行的flex容器的排列方式。**

flex-start： 各行向侧轴的起始位置堆叠。

flex-end： 各行向弹性盒容器的结束位置堆叠。

center： 各行向弹性盒容器的中间位置堆叠。

space-between： 各行在侧轴中平均分布。

space-around： 第一行贴上边，最后一个行贴下边,其他行在弹性盒容器中平均分布。

stretch：拉伸，不设置高度的情况下。

**align-items与align-content的区别**

align-items调整的是侧轴的对其方式，不换行一般用align-items

align-content:必须是多行才生效，如果单行，没有效果。换行了就用align-content。

## （6）**flex属性**

上述讲的属性都是给父盒子设置的，接下来的几个属性是给子盒子设置的。

flex属性用来设置子盒子如何分配主轴空间

flex:1

## **（7）order属性**

order属性定义项目的排列顺序。数值越小，排列越靠前，默认为0。

order:1;

## （8）**align-self align-items:**

align-self也是用于设置在侧轴的位置，但是align-self给子元素设置，优先级比align-items的优先级高。

取值与align-items的取值一样。