Day13笔记

预习动画和响应式

二十一、弹性布局

弹性布局是 css3 主推的一个布局，传统布局是基于文档流和盒子模型的，主要以浮动布局，定位布局等。

给父元素家 display：flex；使子元素横向排列，不用每个子元素浮动了

父元素不需要清除浮动了，高度依然被子元素撑开

1.弹性布局的基本概念

(1)容器和项目

使用 flex 布局的元素（父元素），称为"容器"

它的所有“子元素”自动成为容器成员，称为 "项目"

图片包含 日程表

描述已自动生成它们各自有控制布局的属性，必须是容器包裹着项目

图示

描述已自动生成

(2)主轴

主轴就是项目的排列方向，主轴会出现四个方向：

- x 轴：起点在左侧

- x 轴：起点在右侧

- y 轴：起点在顶端

- y 轴：起点在底端

(3)交叉轴

交叉轴就是在主轴的垂直方向，项目可在交叉轴上移动

2.将容器指定为 flex 布局

- 任何一个容器都可以指定为 Flex 布局:

- 行内元素也可以使用 Flex 布局:

- 注意，设为 Flex 布局以后，子元素的 float、clear 等属性将失效。

3.容器的属性

(1)主轴的方向

`flex-direction` 代表主轴的方向，即项目的排列方向

- row 左向右

- row-reverse 右向左

- column 上向下

- column-reverse 下向上

手机屏幕截图

低可信度描述已自动生成

|  |
| --- |
| .box {  /\* 左向右 右向左 上向下 下向上 \*/  flex-direction: row | row-reverse | column | column-reverse;  } |

(2)项目换行

默认情况下，项目都排在一条线上。`flex-wrap`属性定义如果一条轴线排不下，如何换行的模式。

- nowrap 默认不换行

图标

描述已自动生成

- wrap 换行

图标

描述已自动生成

- wrap-reverse 反向换行

图标

描述已自动生成

|  |
| --- |
| .box {  /\* 默认不换行 换行 反向换行 \*/  flex-wrap: nowrap | wrap | wrap-reverse;  } |

(3)主轴和换行的简写方式

`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`横向排列不换行。

|  |
| --- |
| .box {  flex-flow: <flex-direction> || <flex-wrap>;  flex-flow: row wrap; /\*横向换行\*/  flex-flow: column nowrap; /\*纵向不换行\*/  } |

(4)主轴上的对齐方式

`justify-content`属性定义了项目在主轴上的对齐方式。是默认起点对齐、终点对齐还是居中对齐等。

图表, 条形图

描述已自动生成前提是，如果项目会换行,则下行按照下一行的数量对齐，间隔与上一行可能不同。不换行元素会在容器缩小是缩小，不会出现对齐效果。

- flex-start 默认起点对齐

- flex-end 终点对齐

- center 居中对齐，一起居中

- space-between 两端对齐，项目之间的间隔都相等，左右贴边

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

|  |
| --- |
| .box {  justify-content: flex-start | 默认起点对齐  flex-end | 终点对齐  center | 居中对齐，一起居中  space-between | 两端对齐，项目之间的间隔都相等，左右贴边  space-around; 每个项目两侧的间隔相等，项目之间的间隔比项目与边框的间隔大一倍  } |

(5)交叉轴对齐方式

`align-items`属性定义项目在交叉轴上如何对齐，"前提是"前提是项目换行,容器的垂直轴方向是有独立高度或者宽度的。注意：多行项目会有行间距离

图表, 树状图

描述已自动生成

- 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,需要给项目设置单独的高度和内上边距和行高可以看出来

- 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

|  |
| --- |
| .box {  align-items: 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,需要给项目设置单独的高度和内上边距和行高可以看出来  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)多轴线对齐方式

图表, 树状图

描述已自动生成`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。

- 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 第一行和最后一行贴开始和结束，其他均分中间位置

- space-around 每行间距两侧或上下间距相等，比边框间距大一倍

- stretch 默认值

|  |
| --- |
| .box {  align-content: 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 |第一行和最后一行贴在开始位置和结束位置，其他均分中间位置  space-around |每行间距两侧或上下间距相等，比边框间距大一倍  stretch;(默认值)  } |

【练习】素材在flex27 任选一题

图形用户界面, 网站

描述已自动生成



4.项目属性

(1)项目排列数次序（排队拿号）

- `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。

- 相当于拿号排队，默认是原有元素的顺序，这个是可以重新更改的，单独设置给每个项目，值越大排的越靠后。

- 调整顺序用的不是很多，因为有点反人类思维

|  |
| --- |
| .item {  order: ; /\*数值，值越大排的越靠后，默认为0;\*/  } |

(2)项目的放大比例（多吃多占）

- `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

- 当项目设置了该属性之后，原来的宽度或宽度失效。

- 项目不换行时：

- 父元素有多余的空间，设置放大比例的元素按照比例占据多余空间

- 父元素没有多余空间，当父元素总长度等于和低于元素宽度或者高度相加总和时，元素们同等比例缩小，无特殊化，放大比例失效。

- 项目换行时：

- 父元素会有多余空间，因为只要空间不够就换行了，只要有多余空间，放大比例元素会在当前行按照自己的比例分隔剩余空间。默认比例的元素会正常按照宽度占位

- 父元素宽度与元素宽（高度）相加正好相等，多余空间是 0，那么设置放大比例的元素，则暂时不会变更宽度。

|  |
| --- |
| .item {  flex-grow: number比例值; /\* default 0 \*/  } |

(3)项目的缩小比例（缩水比例）

- `flex-shrink`属性定义了项目的缩小比例，默认为 1，负值无效。

- 前提是容器设置不换行才能看出效果，容器设置宽度看的更清晰

- `flex-shrink：0`：项目不换行容器缩小，其中的项目同等压缩，但设置了 flex-shrink：0 的项目不会压缩，会保持自己的宽度（高度）也就是不参与压缩

- `flex-shrink`：其他值如 2、10 等，缩小比例值越大，缩小的比重就越大

|  |
| --- |
| .item {  flex-shrink: <number>缩小比例值; /\* default 1 \*/  } |

(4)项目的自动尺寸（特立独行）

- 当`flex-basis`和`width`属性同时存在时，`width`属性不生效

- flex 布局有多余空间时，item 的宽度为`flex-basis`设置的宽度

- 当 flex 空间不够时，由于`flex-shrink`的默认值为 1，所以所有 flex items 容器等比例被压缩，设置`flex-basis`的项目也会按照最大值的相对比例压缩，不会和其他元素比例相同。

|  |
| --- |
| .item {  flex-basis: <length>如200px | auto; /\* default auto \*/  } |

(5)flex 属性简写

flex 属性是 flex-grow(放大比例), flex-shrink(缩小比例) 和 flex-basis(自动尺寸)的简写，默认值为 0 1 auto。后两个属性可选。

|  |
| --- |
| 常用设置是：flex:0 0 ??px; 不放大，不缩水，占多少尺寸 |

二十二、过渡

transition 在英文中是过渡的意思，过渡可以为一个元素在不同状态之间切换不同的过渡效果。它由四个部分构成，分别是：过渡属性的名称，过渡需要的时间，过渡的方式和过渡的延迟时间。

1.过渡属性的名称

- `transition-property` 过渡样式

过渡一定是有变化都，在 css 中变化都是样式，比如我们需要过渡一个颜色，那么你要过渡的属性名称就是`background-color`

- 当过渡多个样式的时候可以写 all

2.过渡需要的时间

- `transition-duration`如果让过渡的感觉柔和一下，那就是需要过渡的时间不是一瞬间的而是慢慢的。

- 过渡的时间直接是以秒 s 或者 ms 为单位，默认 0s

3.过渡的方式

- `transition-timing-function`过渡方式,过渡方式的不同而在运动中改变速度

- 贝塞尔曲线函数是非常复杂`http://www.css3beziercurve.net/` 函数是非常复杂的，但是在

- css 中为我们封装了 5 种

- 默认值，先慢再快最后慢`transition-timing-function:ease;`

- 先慢，后越来越快`transition-timing-function:ease-in;`

- 速度在开始和结束时都很慢,中间不加速`transition-timing-function:ease-in-out;`

- 先快，后越来越慢`transition-timing-function:ease-out;`

- 匀速`transition-timing-function:linear;`

|  |
| --- |
| div {  transition-property: all;  transition-duration: 1s;  /\*默认值，先慢再快最后慢\*/  transition-timing-function: ease;  /\*先慢，后越来越快\*/  transition-timing-function: ease-in;  /\*速度在开始和结束时都很慢,中间不加速\*/  transition-timing-function: ease-in-out;  /\*先快，后越来越慢\*/  transition-timing-function: ease-out;  /\*匀速\*/  transition-timing-function: linear;  } |

4.过渡的延迟时间

- 过渡的延迟时间 `transition-delay`在过渡效果开始作用之前需要等待的时间，值以秒（s）或毫秒（ms）为单位。

- 取值为正时会延迟一段时间来响应过渡效果；取值为负时会导致过渡立即开始。

5.简化写法

- 顺序是 transition：过渡时间 延迟时间 过渡方式 过渡样式

- 注意“执行时间和延迟时间的顺序”不能改变

- 最简写法：transition：过渡时间；

CSS3 过渡动画模板、CSS3 Transition 动画模板 http://web.chacuo.net/css3transition

6.多重样式过渡

使用 transition 进行多个样式并且不同时过渡样式时，每一个不同时间的过渡样式需要用逗号分隔。

|  |
| --- |
| div {  width: 100px;  height: 100px;  position: relative;  background-color: red;  left: 0;  /\* 注意第二个的延迟时间 \*/  transition: 1s linear background-color, 1s 1s left;  }  body:hover div {  left: 400px;  background-color: blue;  } |

二十六、变化

CSS 里变化这个属性属于 css 的一个精彩的革新，transform 属性允许多种变化效果的函数对元素进行改变。

变换分为两种：2D（重点）和 3D，需要知道三个轴 xyz。

图片包含 图形用户界面

描述已自动生成

1.transform 变化属性

四个最常用的变化函数，分别是：

- translate 位移

- scale 缩放

- rotate 旋转

- skew 扭曲

2.translate()位移函数

translate()在变换中位移可以有 X 轴、 Y 轴、Z轴的位移方向，参数可以使用长度单位，百分比对应的是自己的宽度和高度，正直或者负值均可。Z轴的位移只有在父元素具有透视的效果下才可以看到。

- `transform: translateX(x); ` 沿 X 轴方向平移 正值右移 负值左移

- `transform: translateY(y);` 沿 Y 轴方向平移 正值下移 负值上移

- `transform: translate(x, y);` 沿 X 轴和 Y 轴同时平移

- 注意：位移和固定定为不同，它并没有脱离文档流，也不会影响其他元素的布局

元素居中，之前 margin 负值是需要知道元素的宽度和高度，使用 translate 百分比值则不需要

|  |
| --- |
| .box div {  position: absolute;  top: 50%;  left: 50%;  transform: translate(-50%, -50%);  } |

3.rotate()旋转函数

- rotate()函数在 2d 变换中旋转只能以 Z 轴进行旋转，所以 rotate()函数默认为 Z 轴旋转函数。

- 参数为旋转角度，deg 单位为旋转角度。角度可以为正值或负值。

- 旋转中心点，是元素最中心的点

4.scale()缩放函数

- scale()缩放函数中的参数是以倍数为基础的，1 代表当前大小

- 1 以上的“正数”为当前大小的倍数。

- 1 以下 0 以上的“正数”为缩小的倍数。

- 0 倍为消失

- 当参数值为负值的时候，元素进行镜面翻转，同样倍数会起作用

5.skew()倾斜扭曲函数

- skew()在 2d 变换中倾斜可以有 X 轴和 Y 轴的倾斜角度

- 填写旋转角度，deg 单位为旋转角度，角度可以为正值或负值。

- skew()默认为 X 轴倾斜函数

7.变换函数的执行顺序

当变换属性需要叠加使用时，不可以生明多个 transform 属性，而是需要把所有要使用的变换函数添加在一个 transform 属性中，用空格分隔。但存在前后顺序问题。

图示

描述已自动生成

|  |
| --- |
| 用控制台改变旋转角度看效果 div:nth-child(1):hover {  transform: translateX(200px) rotate(60deg);  }  用控制台改变旋转角度看效果 div:nth-child(2):hover {  transform: rotate(60deg) translateX(200px);  } |

8.基点

transform-origin 属性是可以改变元素变化时的原点，默认情况下，元素的中心原点位于 x 轴和 y 轴的 50%处。

图片包含 图示

描述已自动生成

- 在变化中，任何元素都有一个中心原点。默认情况下，元素的中心原点位于 x 轴和 y 轴的 50%处。

- transform-origin 属性取值有两种：一种是“长度值”；另外一种是“关键字”。

- 当取值为长度值时，单位可以为 px、em 和百分比等。

表格

描述已自动生成