昨日知识点回顾：

01、线性渐变 径向渐变

02、background新增属性

03、图片边框的基本用法

04、过渡效果transition属性

05、2d&3d变换效果transform属性

今日知识点：

01、3d立方体

02、transform动画

03、字体图标

04、弹性布局

# 1 3d立方体

## 1.1 01-HTML5-立方体(上)

**1、3d立体属性的设置**

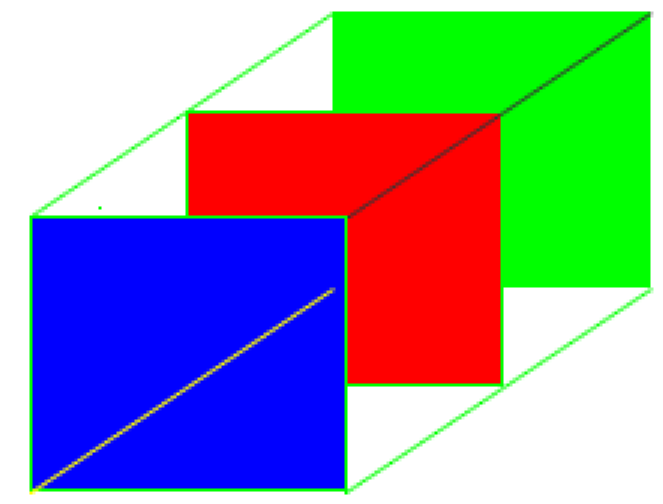
a) transform-style: 使被转换的子元素保留其3D转换(需要设置在父元素中)

|  |  |
| --- | --- |
| **值** | **描述** |
| flat | 子元素将不保留其 3D 位置-平面方式。 |
| preserve-3d | 子元素将保留其 3D 位置—立体方式。 |

## 1.2 02-HTML5-立方体(下)

**左手法则：大拇指指向当前坐标轴的方向，手指环绕的方向就是正方向，**

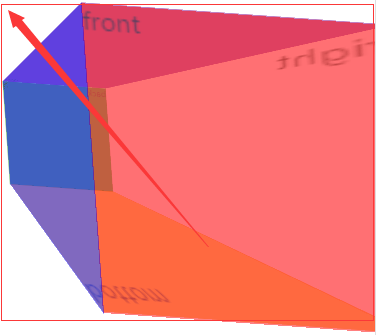
**从手背到手心就是旋转方向**



**步骤分析：**

1. 六张图片分别朝六个方向移动图片宽度高度的一半
2. 再根据左手定则分别朝不同方向旋转90度、180度

## 1.3 03-HTML5-3d变换(景深透视)

1. **perspective 的含义**
   1. 透视距离，也就是我们距离这个效果图有多远 (跟相机焦距一样的原理)，值越大距离图就越远，等于0或none 就相当于重合没有设置
2. **透视观察角度perspective-origin**
   1. perspective-origin：0px 0px 表示可视效果的原点在左上角
   2. 

# 2 transform动画

## 2.1 04-HTML5-动画(创建动画)

1. **如何理解keyframes中的50%，100%？**
   1. 表示的是整个动画耗时的百分比
2. **设置动画的步骤**
   1. 设置动画名 animation-name:动画名
   2. 设置动画总耗时 animation-duration:2s
   3. 创建动画分割节点 @keyframes 动画名 { 0%{} 50%{} 100{}}

div{  
 width: 100px;  
 height: 100px;  
 background-color: red;  
 animation-name: moveTest;  
 animation-duration: 2s;

}

@keyframes moveTest {  
 **0%{transform: translate(30px,20px);}**

**50%{transform: translate(50px,50px);}**

**100%{}**

}

## 2.2 05-HTML5-动画的属性设置

1. **设置动画的播放次数，默认为1次 可以指定具体的数值，也可以指定infinite(无限次)  
   animation-iteration-count: 1;  
   4.设置交替动画 alternate:来回交替  
   animation-direction: alternate;  
   5.设置动画的延迟  
   animation-delay: 2s;  
   6.设置动画结束时的状态：默认情况下，动画执行完毕之后，会回到原始状态  
   forwards:会保留动画结束时的状态，在有延迟的情况下，并不会立刻进行到动画的初始状态**

**即：会停在终点 再刷新若有延迟 不会立刻执行@keyframes中的0%，延迟完毕后立刻执行@keyframes中的0%**

**backwards:不会保留动画结束时的状态，在添加了动画延迟的前提下，如果动画有初始状态，那么会立刻进行到初始状态**

**即：动画执行完 再刷新若有延迟 会立刻执行@keyframes中的0%  
both:会保留动画的结束时状态，在有延迟的情况下也会立刻进入到动画的初始状态**

**即：会停在终点 再刷新若有延迟 会立刻执行@keyframes中的0%，   
animation-fill-mode: both;**  
**7.动画的时间函数  
animation-timing-function: linear;  
设置动画的播放状态 paused:暂停 running:播放  
animation-play-state: running;**

**总结：animation所有属性**

1. 动画序列的名称:animation-name: move;
2. 动画的持续时间:animation-duration: 1s;
3. 动画的延时:animation-delay: 1s;
4. 播放状态:animation-play-state: paused/running;
5. 播放速度:animation-timing-function: linear/ease;
6. 播放次数 反复:animation-iteration-count: 1/infinite;
7. 动画播放完结后的状态:animation-fill-mode: forwards/backwards/both;

h) 循环播放时，来回交替:animation-direction: alternate;

## 2.3 06-HMLT5-动画案例-无缝滚动

**步骤分析：**

1. 结构布局 (7张图片li盒子浮动，设置总盒子的宽度即7张图片盒子的宽度，ul宽度为100%)
2. 给图片盒子ul设置动画名、动画时间、动画次数、动画速度
3. 创建动画调用动画名 从起点到终点移动总盒子的宽度
4. 移动完会有空白，就在添加7张图片，永久执行动画，ul宽度此时就是总盒子宽度的200%，总盒子设置overflow:hidden
5. 实现无缝轮播，再添加hover就设置动画暂停

## 2.4 07-HTML5-时钟案例-结构分析

**原理分析：**

1. 做一个圆
2. 做时间格子，画一根在复制旋转不同的角度
3. 做一个白色圆覆盖中间不要的时间格子，
4. 做时分秒指针

## 2.5 08-HTML5-时钟案例-钟表结构代码实现

**结构步骤：**

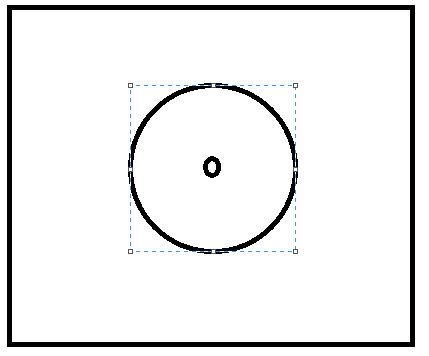
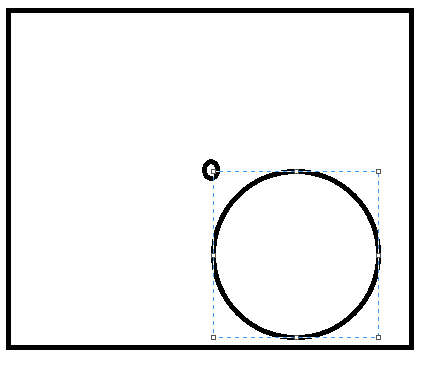
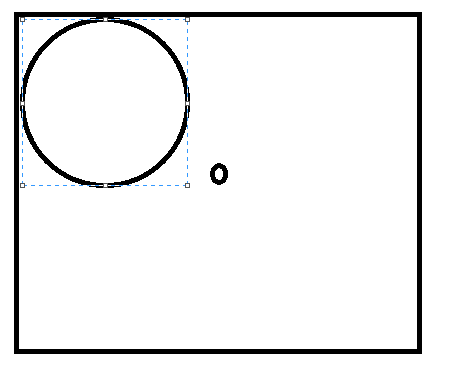
1. **钟盒子结构 钟盒子设置position：relative;**
   1. 6个时间盒子 1~12
   2. 一个圆覆盖中间时间不要的线条的圆
   3. 3根指针即时分秒
   4. 一个圆心
   5. 以上每个都是通过absolute定位

中间盒子定位：

先定位原始在图1，

设置top/left：50%到图2，

在设置偏移移动自身宽高的一半



1. **时间盒子旋转 (6个时间盒子旋转)**
   1. 每个盒子设置定位，位偏移父盒子的50%，在移动自身宽度的一半
   2. 每个分别旋转不同的角度 30度增加
   3. 旋转会出现有间隙，需旋转前加一个移动值，因为后面的旋转会覆盖前面设置的
   4. 第一个和第三个宽度加大
2. **覆盖**
   1. 圆定位top和left都50%；在移动自身宽度和高度的一半
3. **时分秒指针**
   1. 指针定位top和left都50%；宽度移动自身的一半，高度需移动自身高度的100%
4. **圆心**
   1. 定位top和left都50%；在移动自身宽度和高度的一半

<div class="clock">  
 <div class="line line1"></div>  
 <div class="line line2"></div>  
 <div class="line line3"></div>  
 <div class="line line4"></div>  
 <div class="line line5"></div>  
 <div class="line line6"></div>  
 <div class="cover"></div>  
 <div class="hour"></div>  
 <div class="minute"></div>  
 <div class="second"></div>  
 <div class="center"></div>  
</div>

## 2.6 09-HTML5-时钟案例-时钟案例动画添加

1. **动画animation简写**
   1. animation: 动画名 动画时间 动画速度 动画次数

animation: action 10s linear infinite

b)  **animation-timing-function:linear和steps()会冲突 故：一般只能设置一个**

1. **时分秒动画设置**

**a) 秒：**

**i.设置旋转点 : transform-origin:center bottom;**

**ii.设置动画名：animation: clockAnimation 60s infinite steps(60);**

**b) 分：**

**transform-origin: center bottom;  
animation: clockAnimation 3600s linear infinite;**

**b) 时：**

**transform-origin: center bottom;  
animation: clockAnimation 43200s linear infinite;**

# 3 字体图标

## 3.1 10-HTML5-web字体

1. **字体的几种格式**

1、TureTpe(.ttf)格式

2、OpenType(.otf)格式

3、Web Open Font Format(.woff)格式

4、Embedded Open Type(.eot)格式

5、SVG(.svg)格式

**2、iconfont字体的使用**

@font-face {  
 font-family: '**shuangyuan**'; **此处字体名可以自定义**  
 src: url('../fonts/webfont.eot'); /\* IE9\*/  
 src: url('../fonts/webfont.eot?#iefix') format('embedded-opentype'), /\* IE6-IE8 \*/  
 url('../fonts/webfont.woff') format('woff'), /\* chrome、firefox \*/  
 url('../fonts/webfont.ttf') format('truetype'), /\* chrome、firefox、opera、Safari, Android, iOS 4.2+\*/  
 url('../fonts/webfont.svg#webfont') format('svg'); /\* iOS 4.1- \*/  
} **注意：url的路径**  
.myFont{ **需声明引入的字体是跟上面同步的**  
 font-family: shuangyuan;  
}

## 3.2 11-HTML5-字体图标

1. **什么是字体图标？**

常见的是把网页常用的一些小的图标，借助工具帮我们生成一个字体包，然后就可以像使用文字一样使用图标了

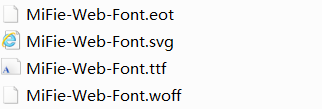
**2、字体图标的优缺点以及如何使用**

**A) 优点**

* + - * + 将所有图标打包成字体库，减少请求；
        + 具有矢量性，可保证清晰度；
        + 使用灵活，便于维护

**B) 生成字体图标文件**

**下载字体图标文件：http://www.iconfont.cn/的网站的介绍和使用**



**C) 自定义字体**

@font-face {  
 font-family: 'wjs'; //自定义的字体名称  
 src: url('../fonts/MiFie-Web-Font.eot'); /\* IE9\*/  
 src: url('../fonts/MiFie-Web-Font.eot') format('embedded-opentype'), /\* IE6-IE8 \*/  
 url('../fonts/MiFie-Web-Font.woff') format('woff'), /\* chrome、firefox \*/  
 url('../fonts/MiFie-Web-Font.ttf') format('truetype'), /\* chrome、firefox、opera、Safari, Android, iOS 4.2+\*/  
 url('../fonts/MiFie-Web-Font.svg') format('svg'); /\* iOS 4.1- \*/  
}

D) 通过css样式使用字体

/\*自定义字体图标\*/  
.wjs\_font\_icon{  
 font-family: wjs; //这里对应着自定义的字体名称  
}  
/\*手机图标对应的编码\*/  
.wjs\_font\_icon\_phone::before{  
 content: "\e908"; //指定显示的内容  
}

# 4 弹性布局

## 4.1 12-HTML5-多列布局

1. **多列布局常用属性**

**1、常用属性：**

a) column-count: 属性设置列的具体个数

b) column-rule: 规定列之间的宽度、样式和颜色

c) column-gap: 两列之间的缝隙间隔

d) column-width: 属性控制列的宽度

e) column-span: 规定元素应横跨多少列(n:指定跨n列 all:跨所有列)

**2、多列布局的使用**

.wrapper {  
 width: 100%;  
 padding: 20px;  
 box-sizing: border-box;  
 **1.设置列数** column-count: 3;  
 **2.添加列间隙样式,与边框样式的添加一样**  
 column-rule: dashed 3px red;  
 **3.设置列间隙大小** column-gap: 50px;  
 **4.设置列宽** **原则：取大优先  
 1.如果人为设置宽度更大，则取更大的值，但是会填充整个屏幕，意味最终的宽度可能也会大于设置的宽度--填充满整个屏幕  
 2.如果人为设置宽度更小，使用默认计算的宽度**  
 column-width: 200px;  
}

## 4.2 13-HTML5-传统布局方式的局限

**1、传统布局的缺陷**

布局的传统解决方案，基于[盒状模型](https://developer.mozilla.org/en-US/docs/Web/CSS/box_model" \t "_blank)，依赖 [display](https://developer.mozilla.org/en-US/docs/Web/CSS/display" \t "_blank)属性 + [position](https://developer.mozilla.org/en-US/docs/Web/CSS/position" \t "_blank)属性 + [float](https://developer.mozilla.org/en-US/docs/Web/CSS/float" \t "_blank)属性。它对于那些特殊布局非常不方便

## 4.3 14-HTML5-伸缩盒子的属性-justify-content

1. **justify-content属性：**
2. **设置父容器为伸缩盒子：会使每一个子元素自动变成伸缩项\*/  
   display: flex; 开启弹性布局**

**2、设置子元素的排列方式  
flex-start:让子元素从父容器的起始位置开始排列  
flex-end:让子元素从父容器的结束位置开始排列  
center:让子元素从父容器的中间位置开始排列  
space-between:左右对齐父容器的开始和结束中间平均分页，产生相同的间距  
space-around:将多余的空间平均的分页在每一个子元素的两边 margin:0 auto.造成中间盒子的间距是左右两边盒子间距的两倍  
justify-content: space-around;**

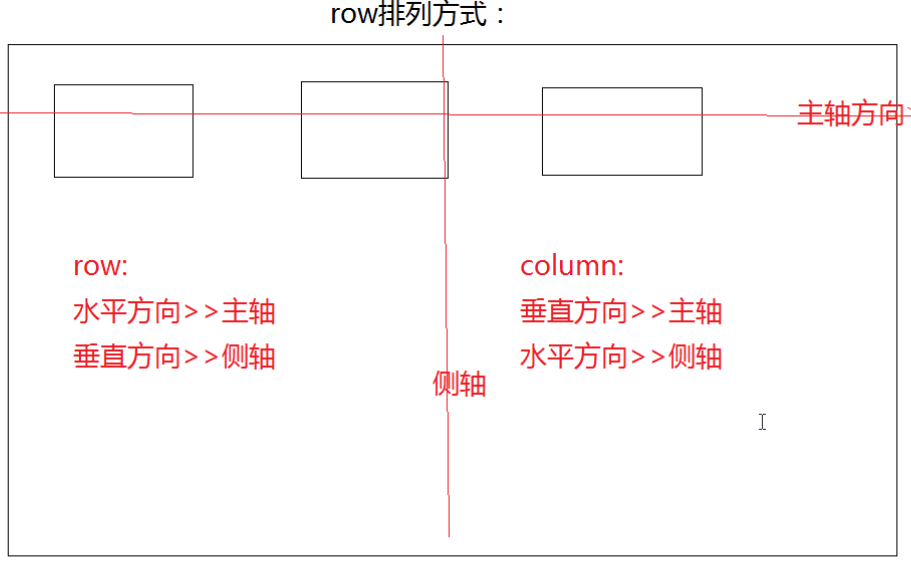
## 4.4 15-HTML5-伸缩盒子的属性-flex-flow

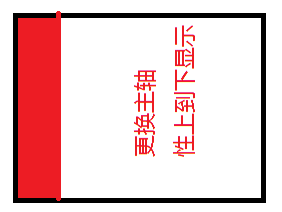
1. **flex-wrap/flex-direction/flex-flow属性的了解**
   1. **flex-wrap: (控制子元素是否换行，默认不换行)**

**nowrap:不换行--则收缩  
wrap:换行  
wrap-reverse:翻转，原来是从上到下，翻转后就是从下到上来排列**

* 1. **flex-direction:**

**设置子元素的排列方向：就是用来主轴方向，默认主轴方向是row(水平方向)  
 row:水平排列方向，从左到右  
 row-reverse:水平排列方向，从右到左  
 column:垂直排列方向，从上到下  
 column-reverse：垂直排列方向，从下到上**



 原先没更换前width会等于更换后height

**c) flex-flow (flex-wrap和flex-direction的结合)**

flex-flow: column wrap-reverse;

## 4.5 16-HTML5-伸缩盒子的属性-flex-grow

1. **flex-grow是如何计算的？**

flow-grow:可以来扩展子元素的宽度：设置当前元素应该占据剩余空间的比例值  
比例值计算 ：当前空间的flex-grow/所有兄弟元素的flex-grow的和  
**flex-grow的默认是0:**说明子元素并不会去占据剩余的空间\*/  
flex-grow: 1;

**例 ： 父盒子宽度是900px,三个div宽度分别是200px;**

**1、.First设置flex-grow=1,其余不设置就是默认为0，**

.first的增加宽度=1/1+0+0\*100%=100%\*300，所以三个盒子占了600px ，剩余300px的就全部到.first上，

.first的宽度=200+100%\*300就是500px

**2、.First设置flex-grow=1,.second设置flex-grow=1其余不设置就是默认为0**

.first的增加宽度=1/1+1+0\*100%=50%\*300,.second的增加宽度=1/1+1+0\*100%=50%\*300，所以三个盒子占了600px剩余300px就全部加到.first和.second上，

.first的宽度就是200+50%\*300=350px, .second的宽度就是200+50%\*300=350px

.first{  
 width: 200px;  
 height: 200px;  
 background-color: red;  
 flex-grow: 1;  
}  
.second{  
 width: 200px;  
 height: 200px;  
 background-color: green;  
 flex-grow: 0;  
}  
.third{  
 width: 200px;  
 height: 200px;  
 background-color: blue;  
 flex-grow: 0;  
}

## 4.6 17-HTML5-伸缩盒子的属性-flex-shink

**1、flex-shrink是如何计算的？**

**flex-shrink:定义收缩比例，通过设置的值来计算收缩空间  
 比例值计算 ：当前空间的flex-shrink/所有兄弟元素的flex-shrink的和**  
**flex-shrink的默认值为1**

**例 ： 父盒子宽度是500px,三个div宽度分别是200px;**

1. **.First设置flex-shrink=0,其余不设置就是默认为1，**

**三个盒子总宽度是600px, 超出100px**

.first的减去宽度=0/0+1+1\*100%=0\*超出100px，

.second的减去宽度=1/0+1+1\*100%=50%\*超出100px=50px，

.third的减去宽度=1/0+1+1\*100%=50%\*超出100px=50px，

最终三个盒子的宽度是：

.first的宽度=200-0=200px;

.second的宽度=200-50px=150px;

.third的宽度=200-50=150px

**2、如果三个盒子都设置flex-shrink=0,那么就会溢出了**

## 4.7 18-HTML5-伸缩盒子的属性-flex

1. **flex的含义和计算方法**
   1. **用来设置计算剩余空间 的比例值**
   2. **步骤分析：**
      1. **父盒子开启弹性布局**
      2. **.first设置flex=1，.second设置flex=1，表示分别占父盒子的50%，就各显示一半**

.box{  
 width: 100%;  
 height: 500px;  
 background-color: #ccc;  
 margin:0 auto;  
 display: flex;  
}  
.left{  
 flex: 1;  
 height: 500px;  
 background-color: red;  
}  
.right{  
 flex: 4;  
 height: 500px;  
 background-color: blue;  
}

## 4.8 19-HTML5-伸缩盒子的flex属性的应用案例

1. **给每个li设置flex=1 为什么就能均分？**
   1. **原理其实就是1/n个1 \*100%**

这里有五个li,相当于每个li就是一个flex=1,每个li就是1/1+1+1+1+1\*100%=20%

如果是三个li 计算原理就是：1/1\*n \*100%

## 4.9 20-HTML5-伸缩盒子的align-items属性

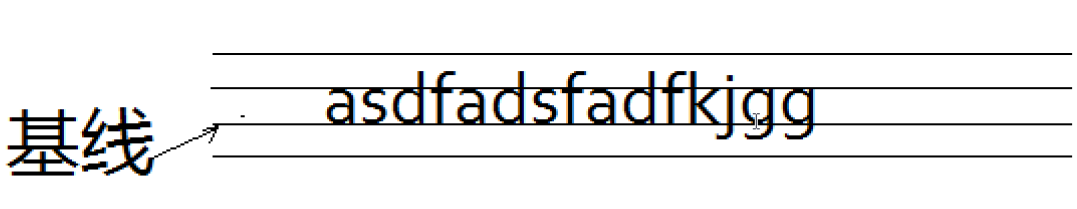
**1、align-items属性的应用**

**align-items:设置子元素(伸缩项)在侧轴方向上的对齐方式  
center:设置在侧轴方向上居中对齐  
flex-start:设置在侧轴方向上顶对齐  
flex:end:设置在侧轴方向上底对齐  
stretch:拉伸：让子元素在侧轴方向上进行拉伸,填充满整个侧轴方向> 默认值  
baseline:文本基线  
align-items: center;**

**2、分析：**

a) 父容器设置align-items：stretch 如果子元素没设置高度就会被拉伸

B) 父容器设置align-items：baseline 基线对齐显示





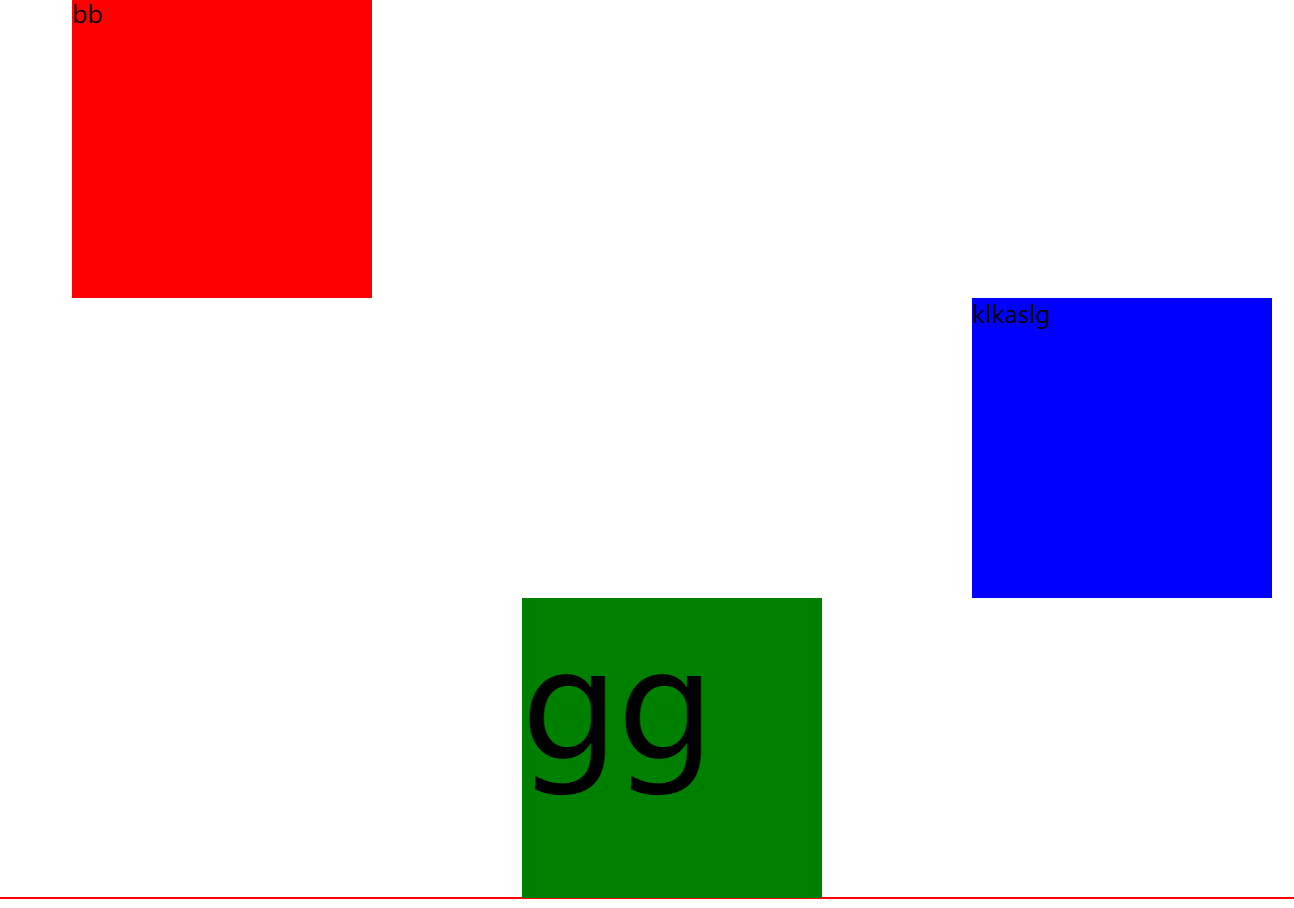
c) 父容器设置align-items：center 子元素不设置，子元素就都会居中

d) 子元素设置align-self:flex-start 显示在侧轴的顶部

align-self主要是对侧轴上中下设置

以下是父元素设置 align-items：center

子元素分别设置align-self:flex-start align-self:flex-end 最后一个不设置



## 4.10 21-HTML5-伸缩盒子案例-宽高自动适应

步骤分析：

1. layout开启弹性布局
2. header和footer在弹性布局中会水平平铺，main没有高度不会显示
3. 为达到header、main、footer显示上中下，需要更改父盒子的主轴，
4. 设置main的高度为自适应直接伸缩父容器剩余部分 flex=1
5. main盒子中有两个水平平铺的盒子article和aside 故需要在main中再次开启弹性布局
6. 设置main 中article 的flex=1 ，aside=3

<div class="layout">  
 <header></header>  
 <main>  
 <article></article>  
 <aside></aside>  
 </main>  
 <footer></footer>  
</div>

\*{  
 padding: 0;  
 margin: 0;  
}  
.layout{  
 width: 500px;  
 height: 600px;  
 background-color: #CCCCCC;  
 margin:10px auto;  
 /\*设置父容器为伸缩盒子\*/  
 display: flex;  
 /\*默认的主轴是row,这里需要以列的方式进行排列\*/  
 flex-direction: column;  
}  
header{  
 width: 100%;  
 height: 60px;  
 background-color: red;  
}  
main{  
 width: 100%;  
 background-color: green;  
 /\*让当前伸缩项占据父容器的剩余空间\*/  
 flex: 1;  
 /\*让main成为伸缩盒子\*/  
 display: flex;  
}  
main > article{  
 height: 100%;  
 flex: 1;  
 background-color: pink;  
}  
main > aside{  
 height: 100%;  
 flex: 3;  
 background-color: darkblue;  
}  
footer{  
 width: 100%;  
 height: 80px;  
 background-color: purple;  
}