CSS03-定位,媒介查询,弹性布局

1. 定位

定位是排版页面元素的一种方式，可以做到精确控制元素的位置。

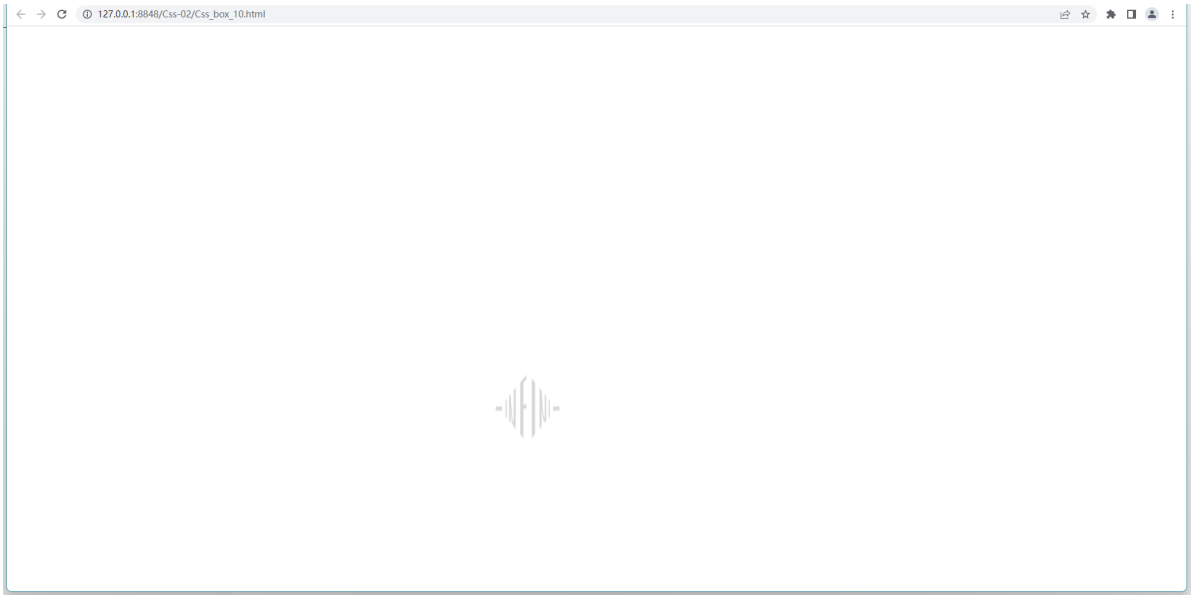
1.1. 定位方式

- 固定定位
- 相对定位
- 绝对定位
- 默认定位

1.2. 固定定位

当前元素以浏览器位置进行定位，即元素的位置相对于浏览器窗口是固定位置，浏览器窗口改变大小也不会改变其位置，设置后，left、top、bottom、right有效。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>固定定位</title>
        <style>
            img{
                width: 100px;
                height: 100px;
                position: fixed;
                top: 546px;
                left: 765px
            }
        </style>
    </head>
    <body>
        <img id="imgfixed" src="img/1.png">
    </body>
</html>
```

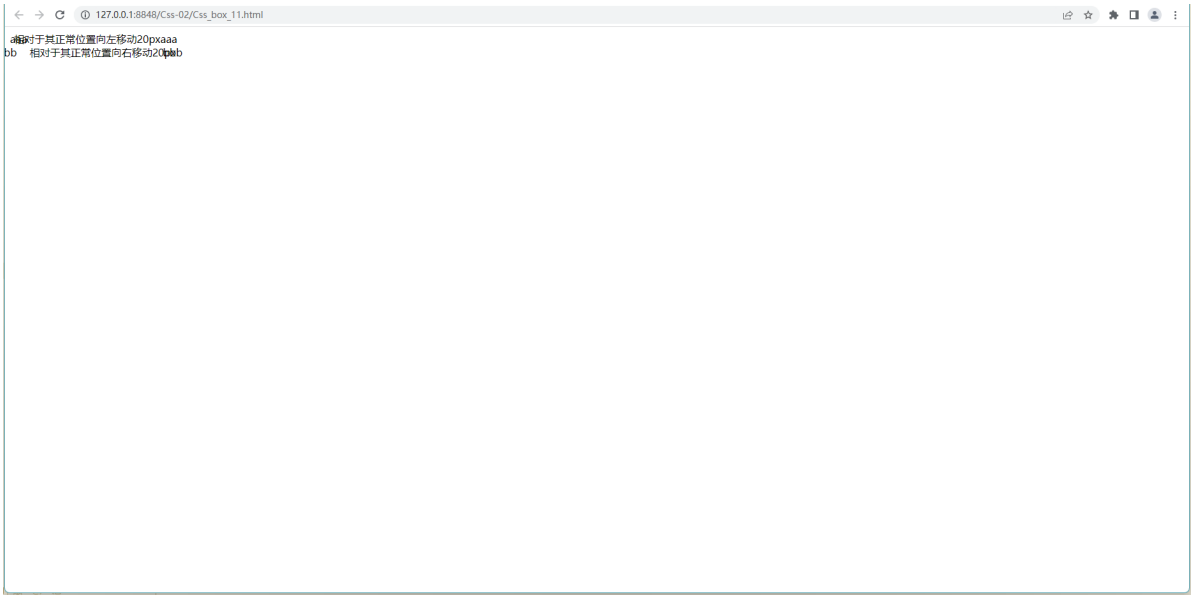


1.3. 相对定位

是相对他自己原先的位置移动，元素移动后，原来的坑不变。定位的元素，可以通过left、right、top、bottom这四个属性设置新的位置。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>相对定位</title>
        <style>
            .pos_left{
                position:relative;
                left:-20px;
            }

            .pos_right{
                position:relative;
                left:20px;
            }
        </style>
    </head>
    <body>
        aaa<span class="pos_left">相对于其正常位置向左移动20px<span>aaa
        <br>
        bbb<span class="pos_right">相对于其正常位置向右移动20px</span>bbb
    </body>
</html>
```



1.4. 绝对定位

绝对定位的元素的位置相对于最近的已定位父元素，如果元素没有已定位的父元素，则相对于<body>，如果浏览器窗口改变大小，则会跟着改变，如果子元素是绝对定位，一般父元素需要设置为相对定位。

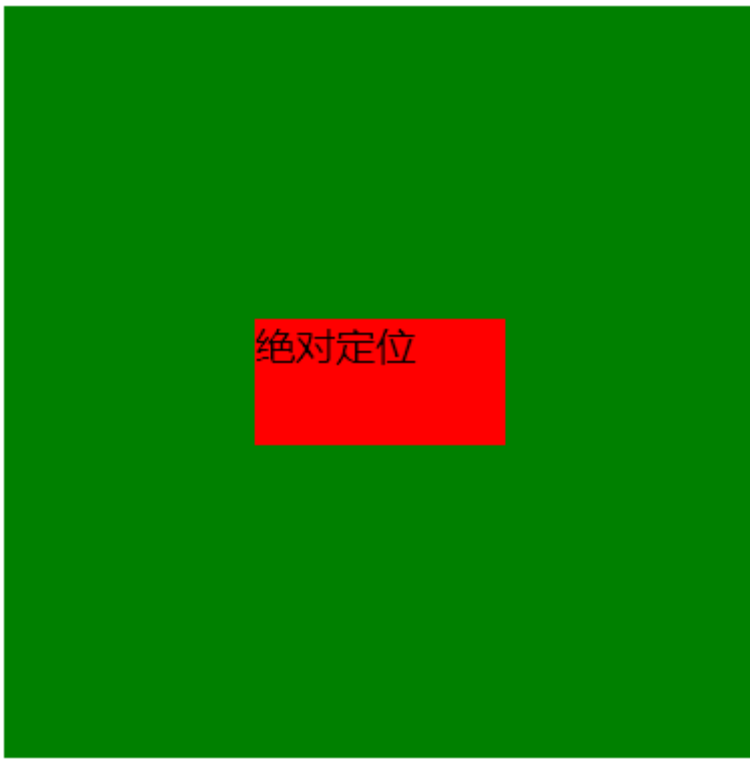
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>CSS绝对定位</title>

        <style>
            *{
                margin: 0;
                padding: 0;
            }
            .parentdiv{
                width: 300px;
                height: 300px;
                background-color: green;
                position: relative;
                top: 100px;
                left: 100px;
            }

            .son{
                width: 100px;
                height: 50px;
                background-color: red;
                position: absolute;
                top: 125px;
                left: 100px;
            }
        </style>

    </head>
    <body>
        <div class="parentdiv">
            <p class="son">绝对定位</p>
        </div>
```

```
    </body>
</html>
```



1.5. 默认定位

HTML 元素的默认值，即没有定位，遵循正常的文档流对象，定位的元素不会受到 top, bottom, left, right影响。

2. 媒介查询

媒介查询的含义

- 通过监听设备的尺寸变化, 触发对应的代码, 达到修改页面布局的目的。

媒介查询的作用

- 实现页面的动态布局。

媒介查询的适用场景

- 针对不同媒体类型定义不同的样式。
- 针对不同屏幕尺寸设置不同的样式，在设置响应式页面时非常有用。

媒介查询的具体用法

语法

```
@media 要监听的媒介类型 and （监听的阈值） {}
```

要监听的媒介类型

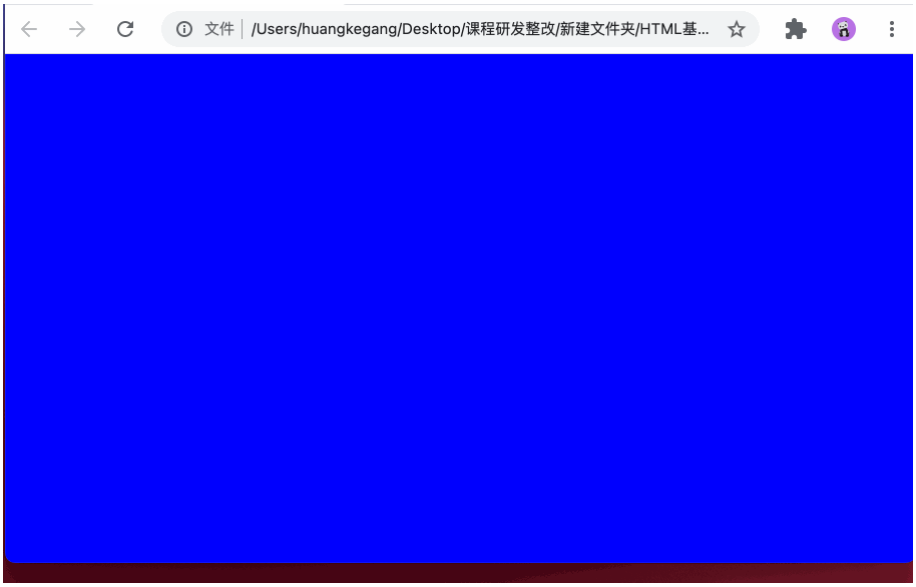
- only screen 手机, 平板, 电脑。

- print 打印机。
- speech 读音设备。
- all 所有设备。

> 监听小于某个临界值，max-width:1000px，表示当前媒介能监听到的最大值是1000px，其可监听的范围1000px以内。

监听大于某个临界值, min-width:1000px, 表示当前媒介能监听到的最小值是1000px, 其可监听的范围是1000px以上。

```
body {
  background-color: red;
}
@media only screen and (max-width: 600px) {
  body {
    background-color: yellow;
  }
}
@media only screen and (min-width: 800px) {
  body {
    background-color: blue;
  }
}
```



注意:所有跟媒介查询有关的代码尽可能的往后写!!

```
  <link rel="stylesheet" type="text/css" href="media.css" media="screen and (max-width: 800px)"/>
```

em和rem单位

em和rem的含义

- em和rem这两个单位都是 长度单位, 浏览器最终都会将其计算成px。

em和rem的作用

- 使用 em 和 rem 单位可以让我们的设计更加灵活，能够控制元素整体放大缩小，而不是固定大小。 我们可以使用这种灵活性，使我们在开发期间，能更加快速灵活的调整，允许浏览器用户调整浏览器大小来达到最佳体验。

em和rem的使用场景

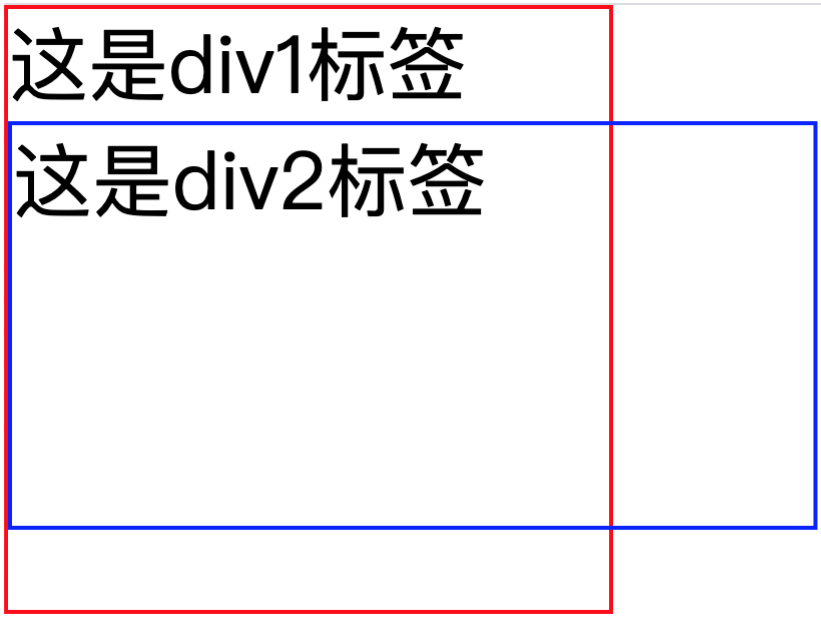
- 移动端布局。

em和rem的用法

- 一个em单位与当前元素的font-size的px值相等。
- 一个rem单位与当前文档根元素的font-size的px值相等。

```
<div class="div1">
  这是div1标签
  <div class="div2">
    这是div2标签
  </div>
</div>
```

```
* {
  margin: 0;
  padding: 0;
}
html {
  font-size: 20px;
}
body {
  font-size: 20px;
}
.div1 {
  width: 300px;
  height: 300px;
  border: 2px solid red;
  font-size: 2em;
}
.div2 {
  width: 10em;
  height: 10rem;
  border: 2px solid blue;
}
```



3. 弹性盒布局的含义

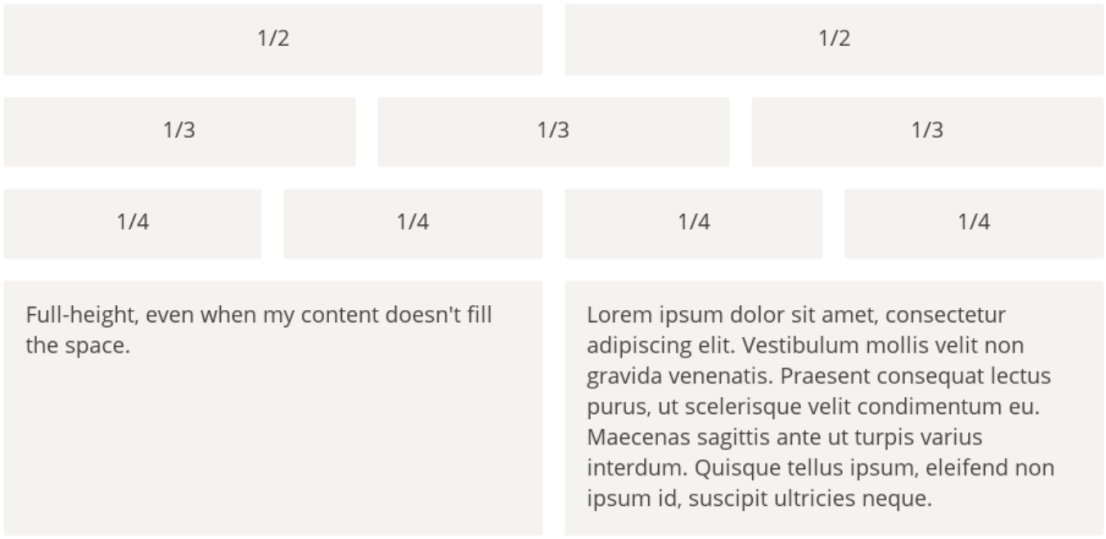
- Flex是Flexible Box的缩写，意为”弹性布局”，用来为盒状模型提供最大的灵活性,旨在提供一个更有效地布局、对齐方式，并且能够使容器中的子元素大小未知或动态变化情况下仍然能够分配好子元素之间的空间。

弹性盒布局的作用

- 我们之前的传统布局, 依赖于盒模型, display, position, float, 但是我们以上的布局方式在解决布局问题时, 多多少少会有一些不够优雅的地方, 比如: 元素的垂直居中, 而Flex(弹性布局) 是W3C推出的一种全新的布局方式, 可以简便,快捷, 响应式的完成各种页面的布局效果. 目前, flex布局已兼容所有浏览器。

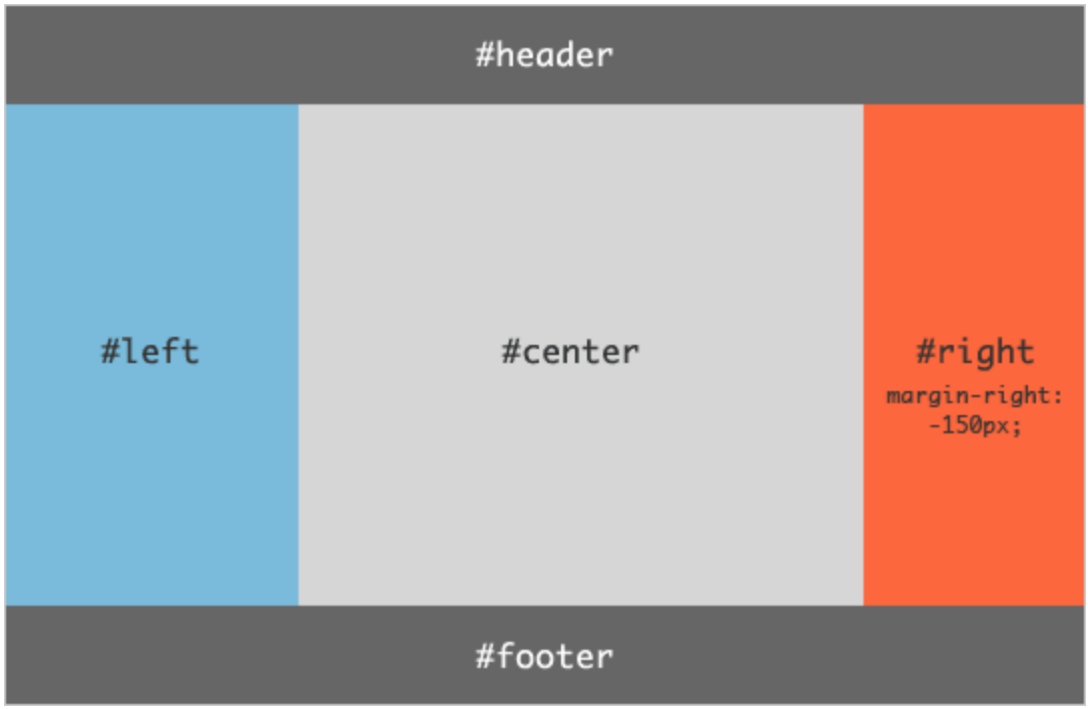
弹性盒布局的适用场景

- **基本网格布局**: 最简单的网格布局，就是平均分布。在容器里面平均分配空间，骰子布局很像，但是需要设置项目的自动缩放。

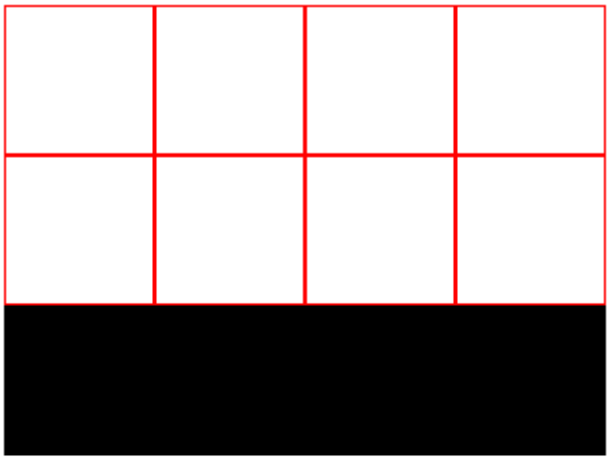


- **百分比布局**:某个网格的宽度为固定的百分比，其余网格平均分配剩余的空间。

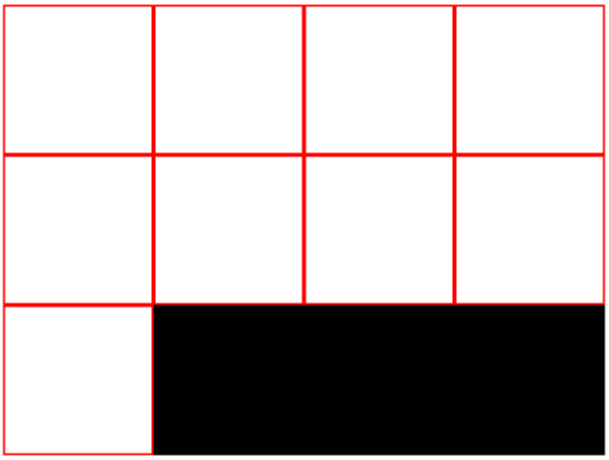
- **圣杯布局**: 圣杯布局(Holy Grail Layout）指的是一种最常见的网站布局。页面从上到下，分成三个部分：头部（header），躯干（body），尾部（footer）。其中躯干又水平分成三栏，从左到右为：导航、主栏、副栏。



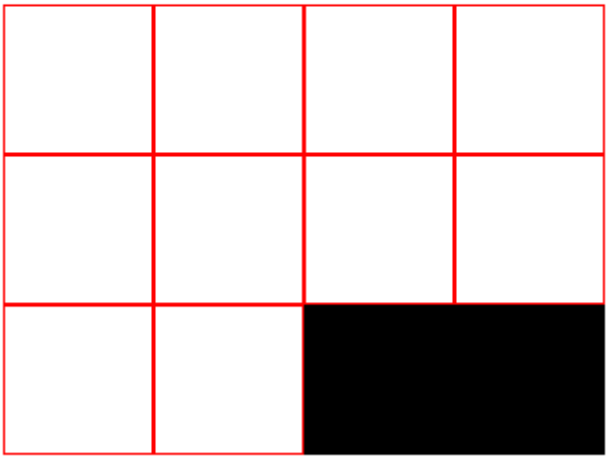
- **流式布局**



8个



9个



10个

- **悬挂式布局**

Basic Examples

Standard Media Object

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac nisl quis massa vulputate adipiscing. Vivamus sit amet risus ligula. Nunc eu pulvinar augue.

Standard Media Object

Donec imperdiet sem leo, id rutrum risus aliquam vitae. Cras tincidunt porta mauris, vel feugiat mauris accumsan eget.

Media Object Reversed

Phasellus vel felis purus. Aliquam consequat pellentesque dui, non mollis erat dictum sit amet. Curabitur non quam dictum, consectetur arcu in, vehicula justo. Donec tortor massa, eleifend nec viverra in, aliquet at eros. Mauris laoreet condimentum mauris, non tempor massa fermentum ut. Integer gravida pharetra cursus. Nunc in suscipit nunc.

Non-images

Using Icons

Donec imperdiet sem leo, id rutrum risus aliquam vitae. Vestibulum ac turpis non lacus dignissim dignissim eu sed dui.

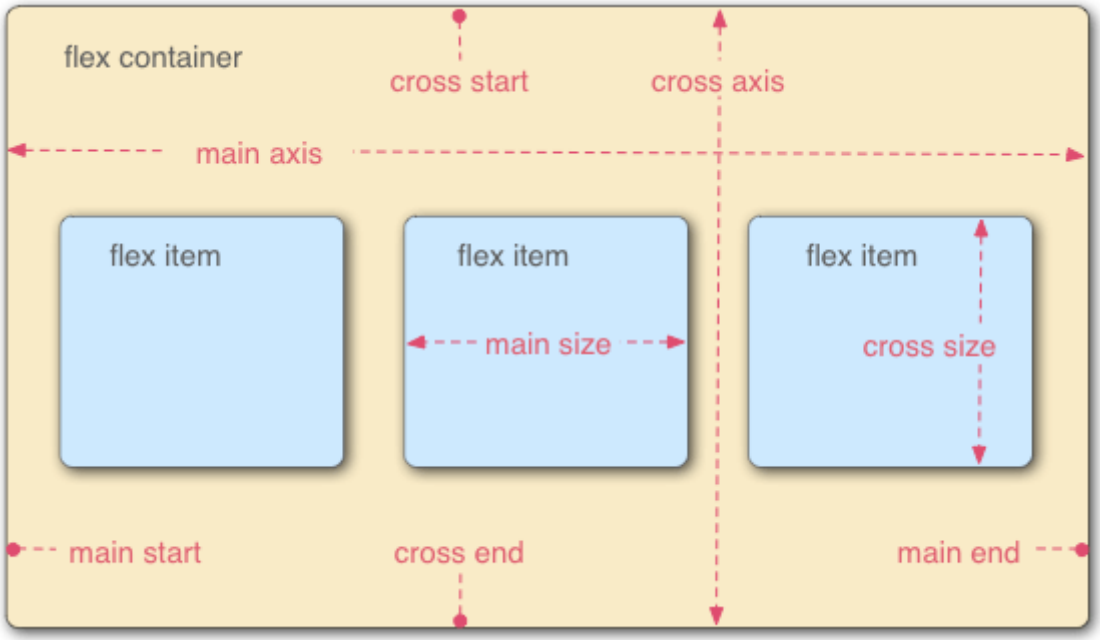
Vertically Centering the Figure

Nunc nec fermentum dolor. Duis at iaculis turpis. Sed rutrum elit ac egestas dapibus. Duis nec consequat enim.

弹性盒布局的具体用法

flex的基本概念:

- **容器 (flex container)**: 设置了display:flex或者inline-flex的元素
- **弹性子元素(flex item)**: 容器里的所有子元素
- **主轴(main axis)和侧轴(交叉轴)(cross axis)**
  - 这两个轴不真实存在。
  - 主轴"默认"是水平的, "默认"方向从容器的左边(main start)延伸至容器的右边(main end)。
  - 侧轴"默认"与主轴垂直, "默认"方向从容器的顶部(cross start)延伸至容器的底部(cross end)。

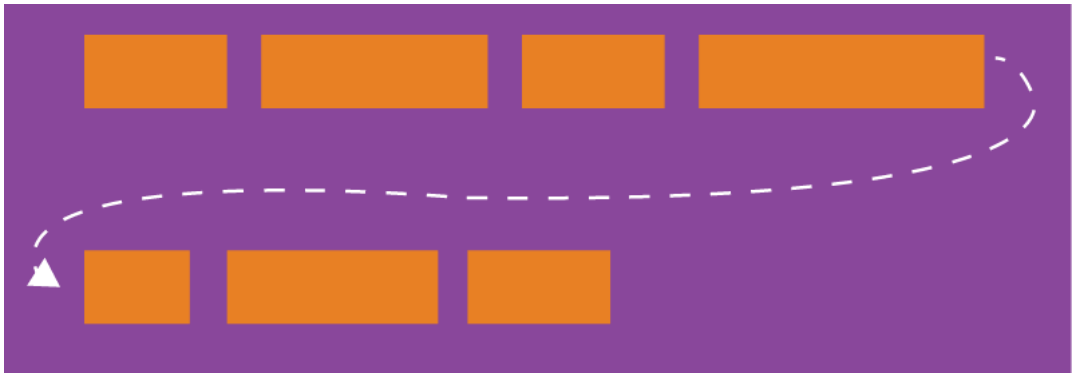


与弹性布局有关的样式

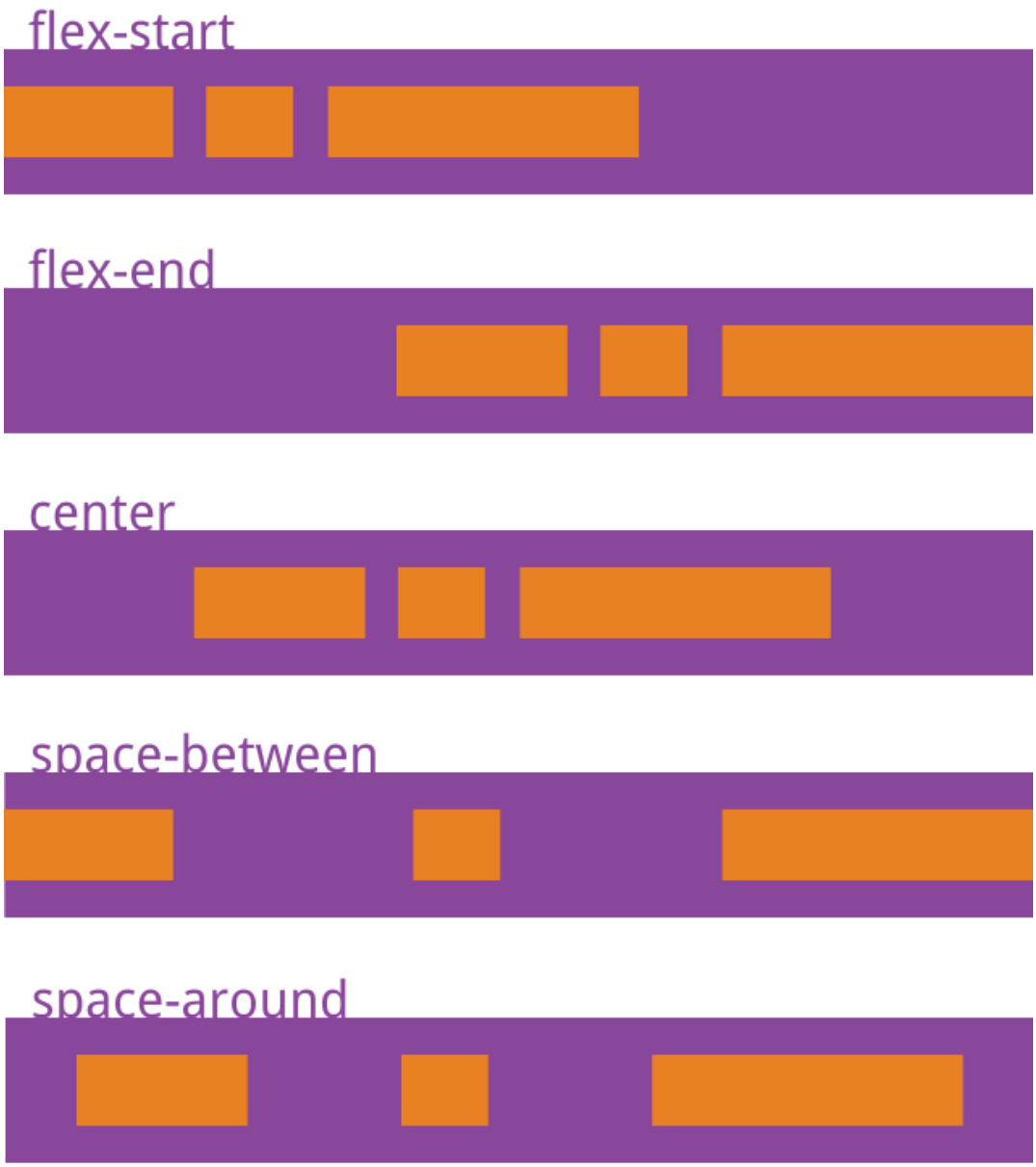
一. 容器有关的样式

- 将元素设置成容器: **display:flex/inline-flex**
- 设置主轴的方向: **flex-direction**
  - row 默认值 水平方向, 从左至右。
  - row-reverse 水平方向, 从右至左。
  - column 垂直方向, 从上至下。
  - column-reverse 垂直方向, 从下至上。

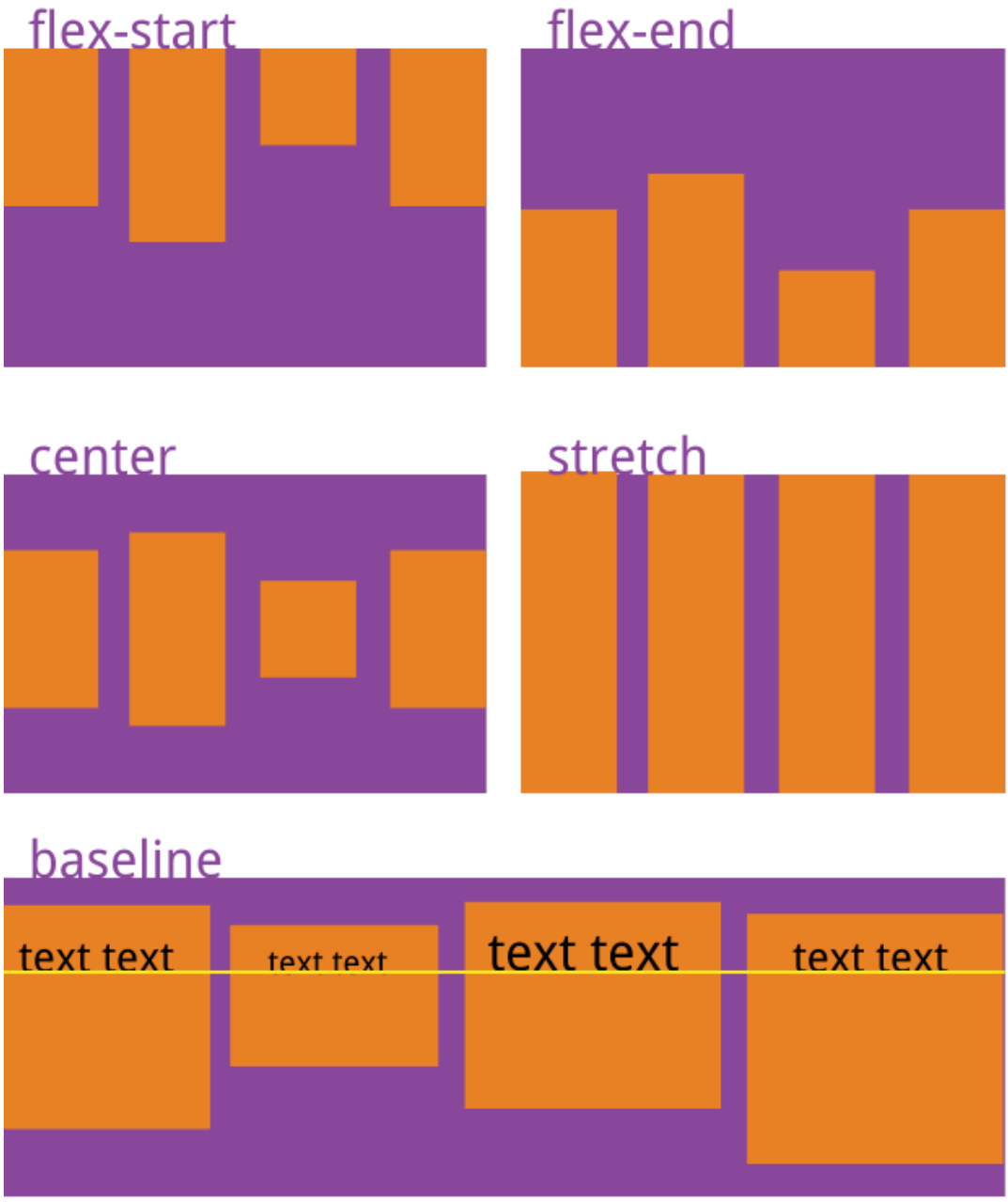
- 弹性子元素的换行方式: **flex-wrap**
  - nowrap 默认值, 不换行, 如果弹性子元素的总宽度超过容器的宽, 系统会强制压缩弹性子元素的宽度以适应容器。
  - wrap 换行。
  - wrap-reverse 反向换行。



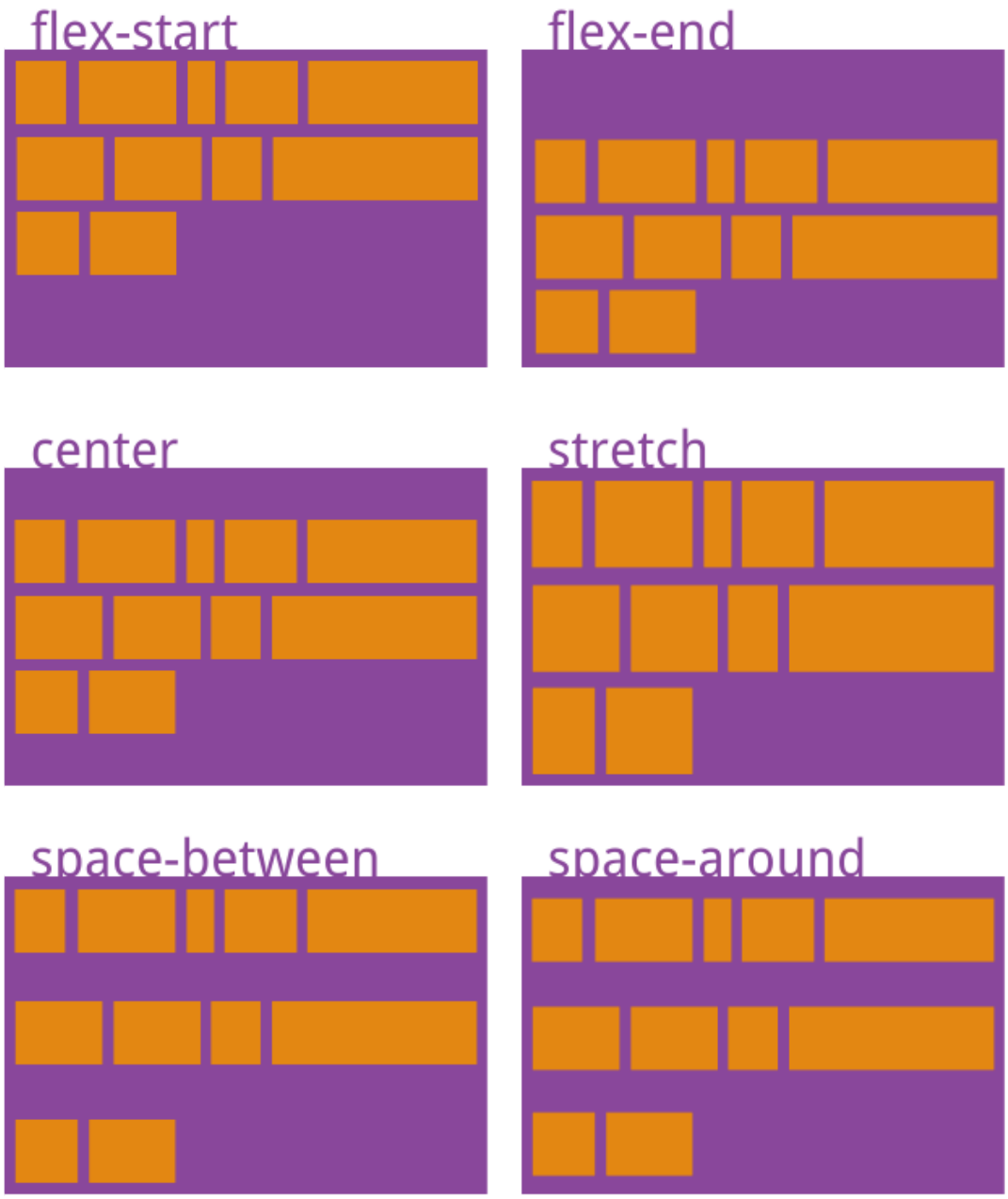
- 弹性子元素在主轴上的对齐方式: **justify-content**
  - flex-start 主轴起点对齐。
  - flex-end 主轴终点对齐。
  - center 主轴居中。
  - space-around 所有元素沿主轴两个方向等距。
  - space-between 主轴方向两端元素靠边, 每个元素间隔等距。



- 单行弹性子元素在侧轴上的对齐方式: **align-items**
  - flex-start 侧轴起点对齐。
  - flex-end 侧轴终点对齐。
  - center 侧轴居中。
  - stretch 默认值, 规定弹性子元素在侧轴方向上的高度, 默认填满整个侧轴, 该效果只在不设置弹性子元素高度的情况下才会出现。
  - baseline 基线 对齐。



- 多行弹性子元素在侧轴上的对齐方式: **align-content**
  - flex-start 侧轴起点对齐。
  - flex-end 侧轴终点对齐。
  - center 侧轴居中。
  - stretch 默认值, 规定弹性子元素在侧轴方向上的高度, 默认填满整行, 该效果只在不设置弹性子元素高度的情况下才会出现。
  - space-between 侧轴方向两端元素靠边, 每个元素间隔等距。
  - space-around 所有元素沿侧轴两个方向等距。



课堂总结

- 掌握em和rem相对单位的区别和联系。
- 掌握媒体查询的适用场景及语法规范。
- 掌握弹性盒模型的基本概念及实现原理。
- 掌握弹性盒布局的主轴和交叉轴的关系。
- 掌握主轴对齐方式和交叉轴对齐方式。