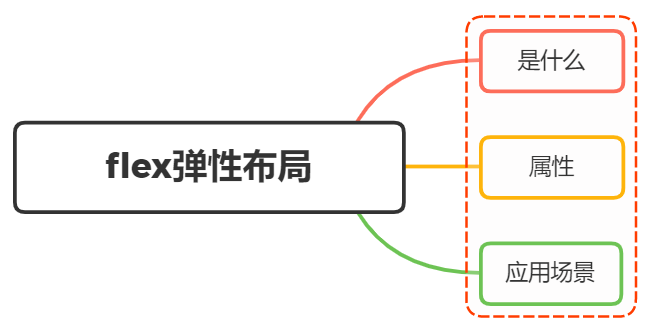
# 面试官：说说flexbox（弹性盒布局模型）,以及适用场景？

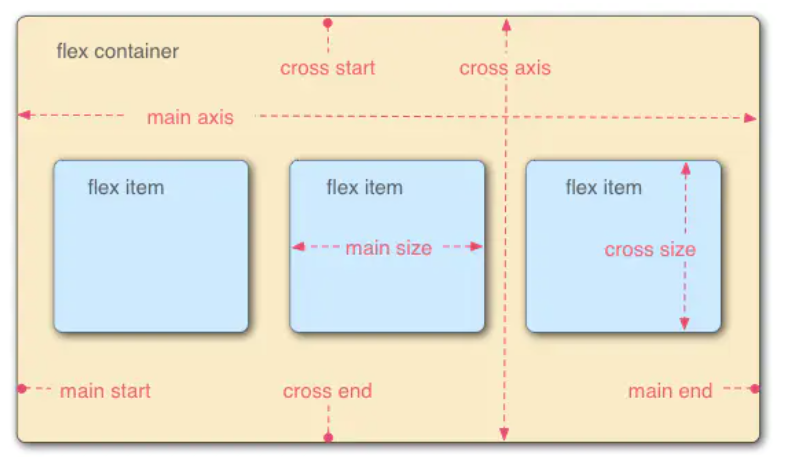


## 一、是什么

Flexible Box 简称 flex，意为”弹性布局”，可以简便、完整、响应式地实现各种页面布局

采用Flex布局的元素，称为flex容器container

它的所有子元素自动成为容器成员，称为flex项目item



容器中默认存在两条轴，主轴和交叉轴，呈90度关系。项目默认沿主轴排列，通过flex-direction来决定主轴的方向

每根轴都有起点和终点，这对于元素的对齐非常重要

## 二、属性

关于flex常用的属性，我们可以划分为容器属性和容器成员属性

容器属性有：

* flex-direction
* flex-wrap
* flex-flow
* justify-content
* align-items
* align-content

### flex-direction

决定主轴的方向(即项目的排列方向)

.container {   
 flex-direction: row | row-reverse | column | column-reverse;   
}

属性对应如下：

* row（默认值）：主轴为水平方向，起点在左端
* row-reverse：主轴为水平方向，起点在右端
* column：主轴为垂直方向，起点在上沿。
* column-reverse：主轴为垂直方向，起点在下沿

如下图所示：



### flex-wrap

弹性元素永远沿主轴排列，那么如果主轴排不下，通过flex-wrap决定容器内项目是否可换行

.container {   
 flex-wrap: nowrap | wrap | wrap-reverse;  
}

属性对应如下：

* nowrap（默认值）：不换行
* wrap：换行，第一行在下方
* wrap-reverse：换行，第一行在上方

默认情况是不换行，但这里也不会任由元素直接溢出容器，会涉及到元素的弹性伸缩

### flex-flow

是flex-direction属性和flex-wrap属性的简写形式，默认值为row nowrap

.box {  
 flex-flow: <flex-direction> || <flex-wrap>;  
}

### justify-content

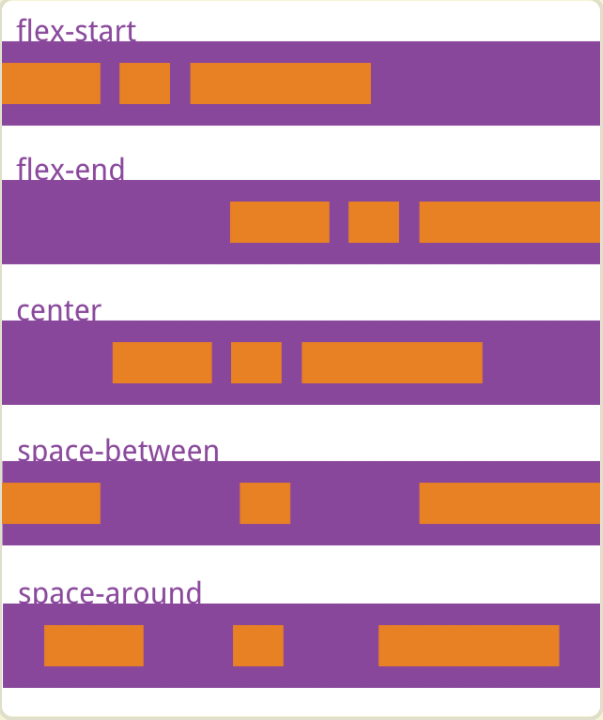
定义了项目在主轴上的对齐方式

.box {  
 justify-content: flex-start | flex-end | center | space-between | space-around;  
}

属性对应如下：

* flex-start（默认值）：左对齐
* flex-end：右对齐
* center：居中
* space-between：两端对齐，项目之间的间隔都相等
* space-around：两个项目两侧间隔相等

效果图如下：



### align-items

定义项目在交叉轴上如何对齐

.box {  
 align-items: flex-start | flex-end | center | baseline | stretch;  
}

属性对应如下：

* 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
* 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

### align-content

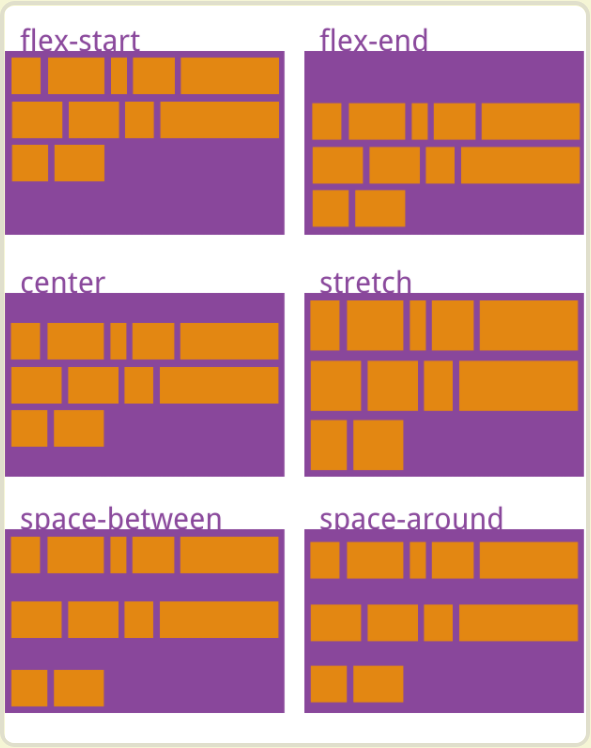
定义了多根轴线的对齐方式。如果项目只有一根轴线，该属性不起作用

.box {  
 align-content: flex-start | flex-end | center | space-between | space-around | stretch;  
}

属性对应如吓：

* 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：与交叉轴两端对齐，轴线之间的间隔平均分布
* space-around：每根轴线两侧的间隔都相等。所以，轴线之间的间隔比轴线与边框的间隔大一倍
* stretch（默认值）：轴线占满整个交叉轴

效果图如下：



容器成员属性如下：

* order
* flex-grow
* flex-shrink
* flex-basis
* flex
* align-self

### order

定义项目的排列顺序。数值越小，排列越靠前，默认为0

.item {  
 order: <integer>;  
}

### flex-grow

上面讲到当容器设为flex-wrap: nowrap;不换行的时候，容器宽度有不够分的情况，弹性元素会根据flex-grow来决定

定义项目的放大比例（容器宽度>元素总宽度时如何伸展）

默认为0，即如果存在剩余空间，也不放大

.item {  
 flex-grow: <number>;  
}

如果所有项目的flex-grow属性都为1，则它们将等分剩余空间（如果有的话）



如果一个项目的flex-grow属性为2，其他项目都为1，则前者占据的剩余空间将比其他项多一倍



弹性容器的宽度正好等于元素宽度总和，无多余宽度，此时无论flex-grow是什么值都不会生效

### flex-shrink

定义了项目的缩小比例（容器宽度; /\* default 1 \*/ } ```

如果所有项目的flex-shrink属性都为1，当空间不足时，都将等比例缩小

如果一个项目的flex-shrink属性为0，其他项目都为1，则空间不足时，前者不缩小



在容器宽度有剩余时，flex-shrink也是不会生效的

### flex-basis

设置的是元素在主轴上的初始尺寸，所谓的初始尺寸就是元素在flex-grow和flex-shrink生效前的尺寸

浏览器根据这个属性，计算主轴是否有多余空间，默认值为auto，即项目的本来大小，如设置了width则元素尺寸由width/height决定（主轴方向），没有设置则由内容决定

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

当设置为0的是，会根据内容撑开

它可以设为跟width或height属性一样的值（比如350px），则项目将占据固定空间

### flex

flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto，也是比较难懂的一个复合属性

css .item { flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ] }

一些属性有：

* flex: 1 = flex: 1 1 0%
* flex: 2 = flex: 2 1 0%
* flex: auto = flex: 1 1 auto
* flex: none = flex: 0 0 auto，常用于固定尺寸不伸缩

flex:1 和 flex:auto 的区别，可以归结于flex-basis:0和flex-basis:auto的区别

当设置为0时（绝对弹性元素），此时相当于告诉flex-grow和flex-shrink在伸缩的时候不需要考虑我的尺寸

当设置为auto时（相对弹性元素），此时则需要在伸缩时将元素尺寸纳入考虑

注意：建议优先使用这个属性，而不是单独写三个分离的属性，因为浏览器会推算相关值

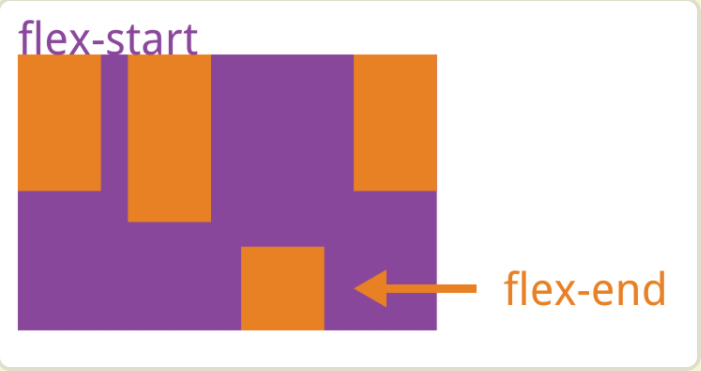
### align-self

允许单个项目有与其他项目不一样的对齐方式，可覆盖align-items属性

默认值为auto，表示继承父元素的align-items属性，如果没有父元素，则等同于stretch

.item {  
 align-self: auto | flex-start | flex-end | center | baseline | stretch;  
}

效果图如下：



## 三、应用场景

在以前的文章中，我们能够通过flex简单粗暴的实现元素水平垂直方向的居中，以及在两栏三栏自适应布局中通过flex完成，这里就不再展开代码的演示

包括现在在移动端、小程序这边的开发，都建议使用flex进行布局

## 参考文献

* https://developer.mozilla.org/zh-CN/docs/Web/CSS/flex
* http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html