**Comparação entre as formas de posicionamento**

display: inline

Colocando display: inline nos elementos permite eles se posicionarem um do lado do outro, o problema do display: inline é que os elementos não aceitam mais que seja modificada tanto a width quanto a height. Isso limita MUITO nossas possibilidades.

display: inline-block

O display: inline-block permite os elementos se posicionarem um do lado do outro porém, diferentemente do display: inline ele permite que os elementos tenham sua width e height modificadas. Por esse motivo o display: inline-block é muito mais interessante na maioria dos casos do que o display: inline.

O problema de usar display: inline-block é espaçar os elementos entre si. Para fazer isso teríamos que colocar margins e fazer contas.

float: left | right

O float é mais complicado, ele empurra o elemento para um dos lados (left | right) e os elementos que estão embaixo sobem. Isso nem sempre é o que a gente quer. Além do mais o float não permite que usemos a propriedade vertical-align: middle para alinhar os elementos verticalmente. Ou seja, para contornar isso uma possibilidade seria ter que colocar margin-top ou bottom nos elementos e usar os temidos números mágicos!

display: flex

O display: flex veio com o intuito de facilitar nossa vida nesses aspectos de posicionamento. Ele permite os elementos ficarem um do lado do outro, permite espaçar os elementos de forma mais intuitiva e sem ter que fazer cálculos. Além disso ele também permite alinhar os elementos verticalmente de forma fácil.

**Forma de espaçar os elementos**

Podemos distribuir os elementos dentro do pai de diversas formas, podemos por exemplo:

Colocar todo espaço à esquerda, jogando o conteúdo para direita com justify-content: flex-end.

Colocar todo espaço à direita, jogando o conteúdo para esquerda com justify-content: flex-start (*que é o padrão).*

Colocar todo espaço à esquerda e à direita, jogando o conteúdo para o meio com justify-content: center.

Colocar todo espaço entre os elementos como vimos antes usando justify-content: space-between.

E uma possibilidade bem interessante também é colocar o espaço em volta dos elementos. Podemos usar o justify-content: space-around para isso.

|  |
| --- |
| quando colocamos display: flex em algum elemento, os seus filhos ficam um do lado do outro. Porém existem casos em que queremos que os filhos fiquem um em baixo do outro, como podemos fazer isso? |
| **flex-direction: column no elemento pai** |
| dessa forma ele muda o fluxo o qual o flexbox renderiza os elementos.  Assim eles ficam um em baixo do outro. |

|  |
| --- |
| Por que às vezes pode ser complicado utilizar justify-content: space-between ou space-around para o grid? |
| É complicado utilizá-las porque elas colocam comportamentos que não são legais para grids. Tanto space-between quanto space-around deixariam super estranha a última linha do grid caso essa tivesse menos elementos do que o total de elementos por linha. |

|  |
| --- |
| Quando temos que adaptar nossos sites para mobile, qual o fluxo que normalmente os elementos seguem para aproveitar melhor o espaço?  Como fazer com que esse fluxo seja seguido utilizando o flexbox? |
| Para aproveitar o espaço precisamos colocar os elementos um em baixo do outro.  Para fazer isso devemos colocar flex-direction: column. |

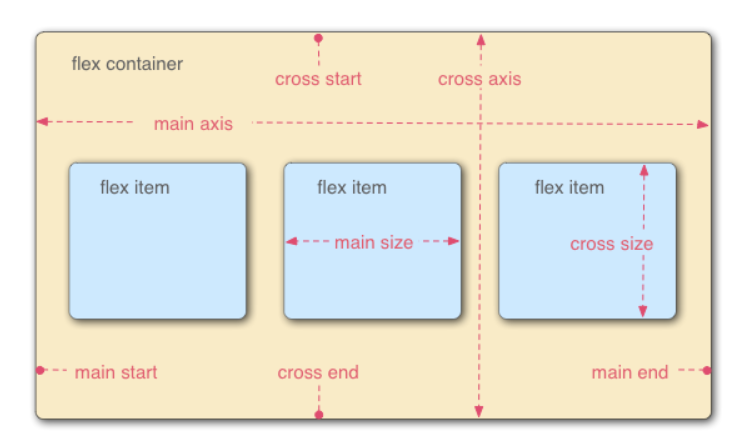
**Flex container e flex items**

Quando colocamos display: flex em um elemento, o navegador passa a considerar esse elemento como um flex container, ou seja, cria todo aquele comportamento que vimos anteriormente no curso, os filhos ficam um do lado do outro e podemos aplicar propriedades para espaçá-los.

Os filhos de um flex container por sua vez também ganham um nome, são chamados de flex items.

Quando utilizamos flexbox temos que ficar atentos em quem colocamos as propriedades de espaçamento e distribuição do flex. Por exemplo, existem algumas propriedades que devem ser aplicadas à flex container e outras que devem ser aplicadas nos flex items.

Para melhor ilustrar esses conceitos veja a seguinte imagem:



|  |
| --- |
| propriedades que funcionam SOMENTE nos flex container e quais são as que funcionam nos flex items? |
| flex container:  justify-content, align-items, flex-direction, flex-wrap  flex items:  order, flex-grow, flex-shrink  A propriedade display: flex pode ser usada nos dois. Se for usada em um flex item esse elemento será tanto um flex item quanto um flex container. |
| flex container:  justify-content, align-items, display: flex,  flex items:  order, flex-grow, flex-shrink |

|  |
| --- |
| Para falar para um elemento / flex item crescer e ocupar todo o espaço que está sobrando dentro do flex container devemos usar a propriedade flex-grow. |
| .flex-item {  flex-grow: 1;  } |

|  |
| --- |
| **propriedade flex-basis** |
| A propriedade flex-basis serve para definir uma largura para o elemento caso o flex container esteja com flex-direction: row.  flex-basis: 500px; |
| A propriedade flex-basis serve para definir uma altura para o elemento caso o flex container esteja com flex-direction: column.  flex-basis: 500px; |