CSS

Flexbox
Flex Properties

Flexbox

Módulo incorporado em CSS.

Permite a flexibilização de layouts.

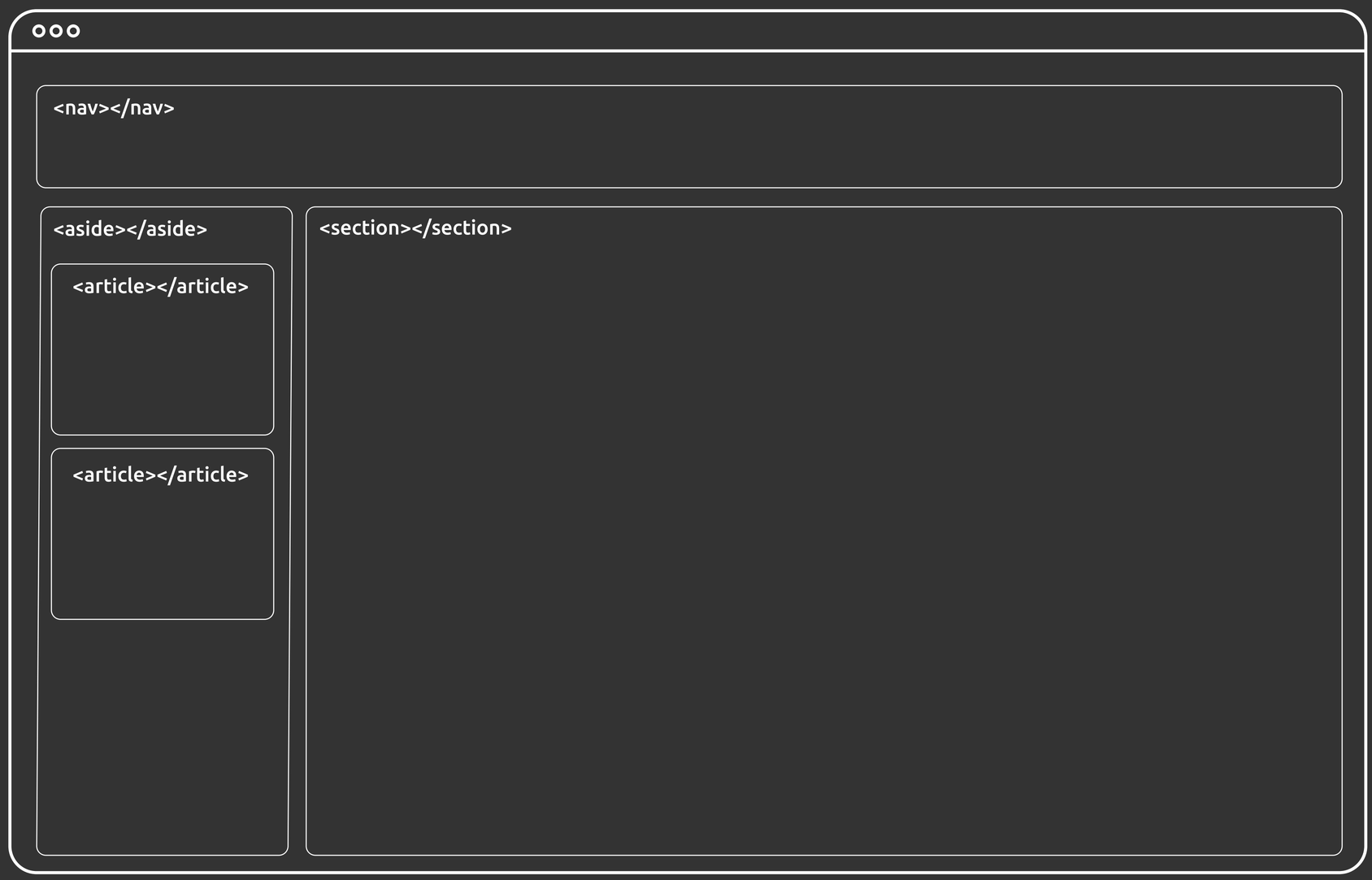
Flexbox

Ajuda-nos a construir layouts de
forma rápida e simples.

sem flexbox

Flexbox

Ajuda-nos a construir layouts de forma rápida e simples.



com flexbox

Relação parent > child(s)

Quando um pai é definido como display: flex;

Os filhos ganham acesso a várias propriedades que permitem definir posicionamento mais flexível.

```
.css
.container {
    display: block;
}
```

```
.html
<div class=“container”>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```

Relação parent > child(s)

Quando um pai é definido como display: flex;

Os filhos ganham acesso a várias propriedades que permitem definir posicionamento mais flexível.

```
.css
.container {
    display: flex;
}
```

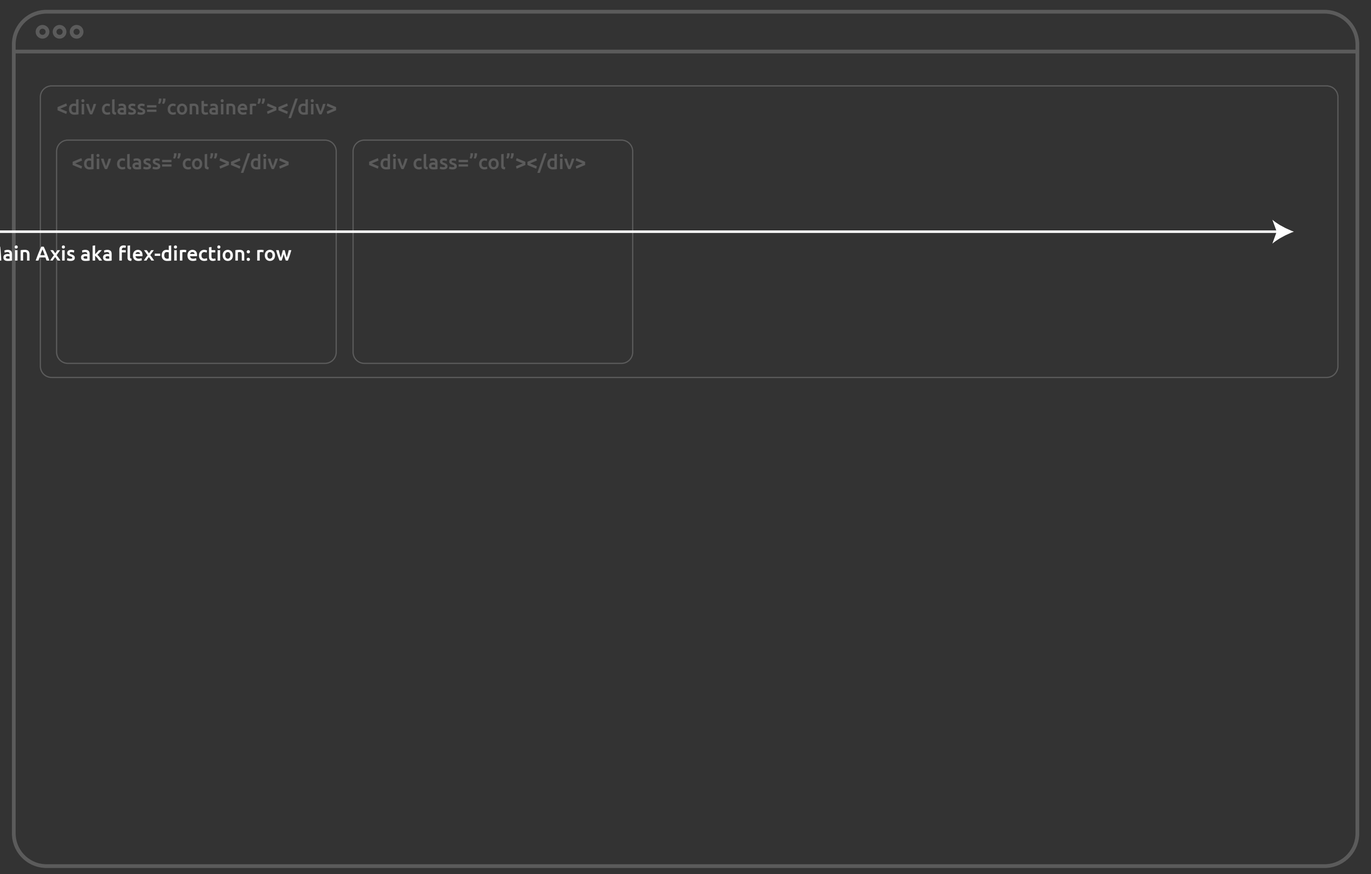
```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```

Relação parent > child(s)

Ao definir um container (parent) como flex, é automaticamente delineado um eixo de representação dos filhos. No default chamado flex-direction: row, por outras palavras o eixo X torna-se o fluxo normalmente dos nossos filhos.

```
.css
.container {
    display: flex;
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```



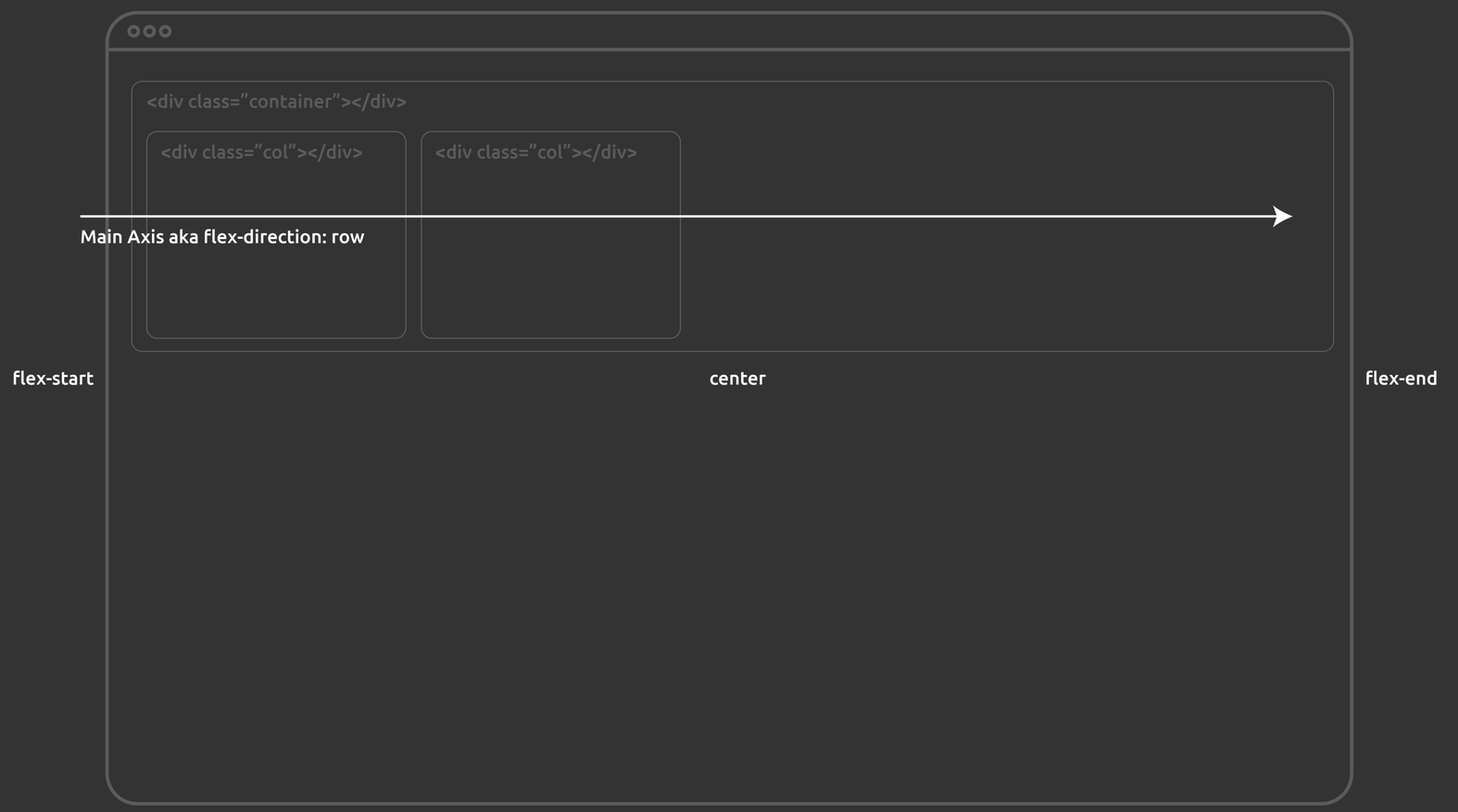
Relação parent > child(s)

O eixo X (flex-direction: row) tem o um início(flex-start) e um fim(flex-end).

Juntam-se a estas duas, o center, o space-between e o space-around.

```
.css
.container {
    display: flex;
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```



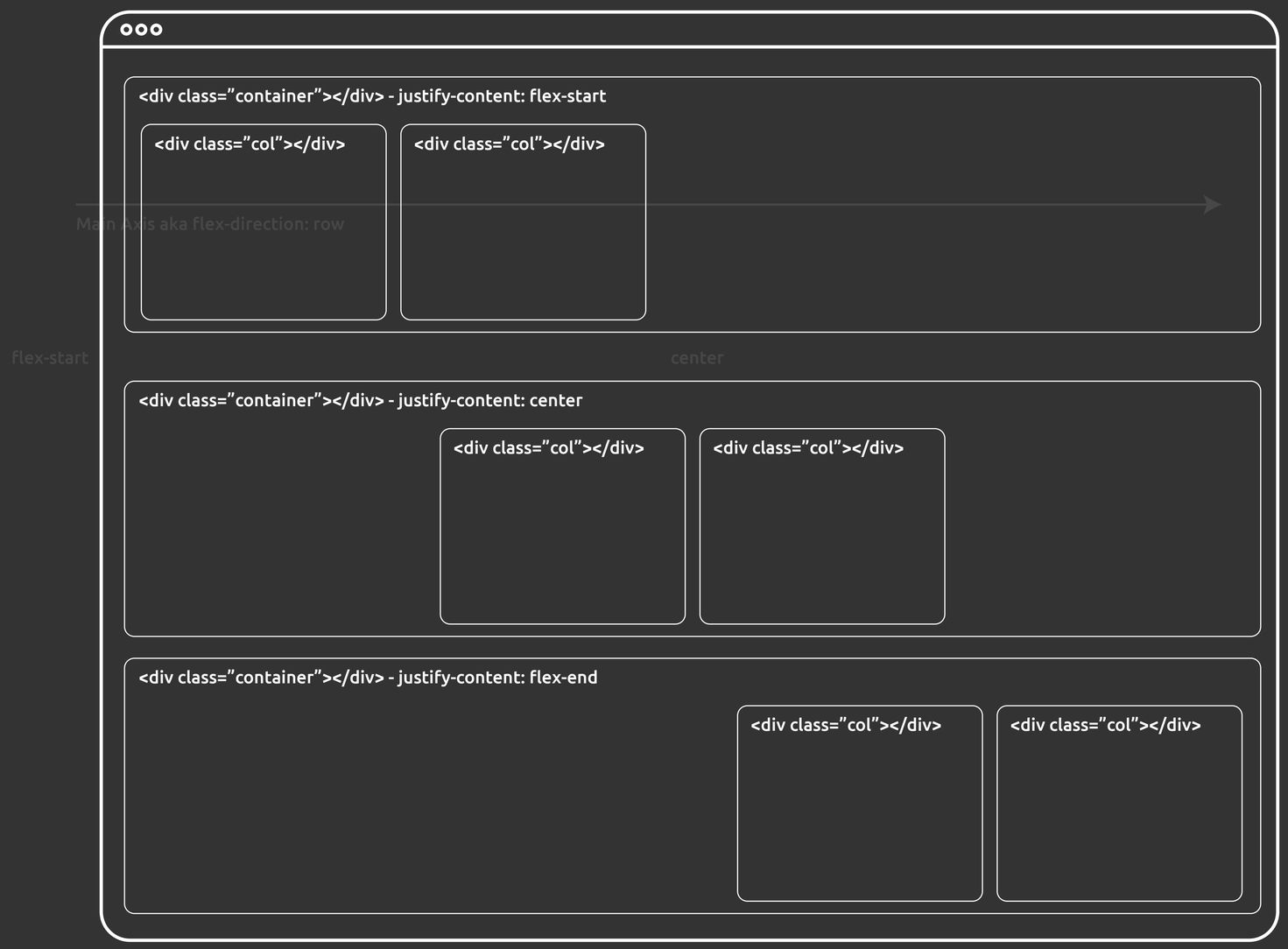
Relação parent > child(s)

O eixo X (flex-direction: row) tem o um início(flex-start) e um fim(flex-end).

Juntam-se a estas duas, o center, o space-between e o space-around.

```
.css
.container {
    display: flex;
    justify-content: flex-start || center || flex-end
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```

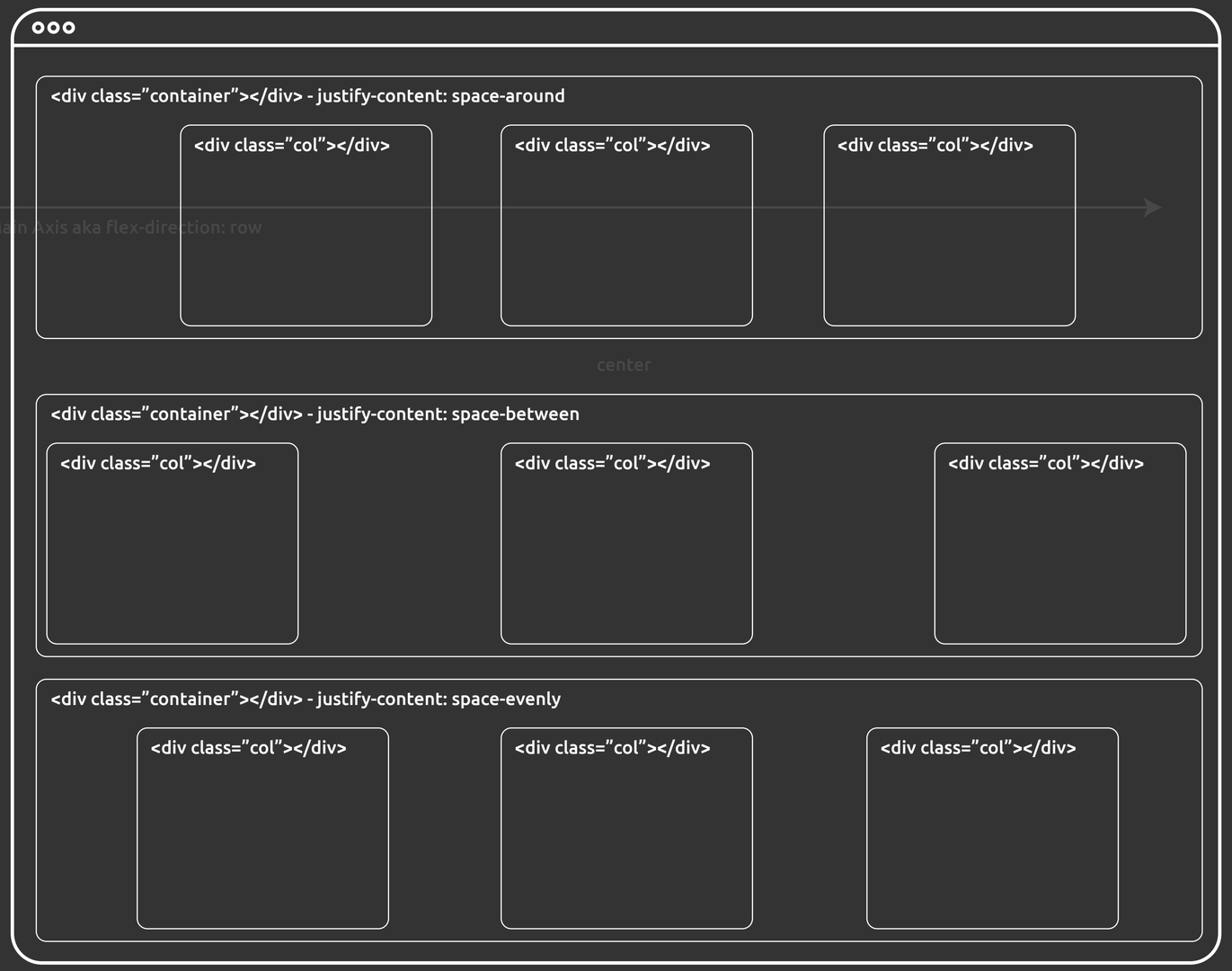


Relação parent > child(s)

Quando o justify-content é space-between || space-around || space-evenly,
os filhos são distribuídos pelo espaço disponível do pai.

```
.css
.container {
    display: flex;
    justify-content:
    space-between || space-around || space-evenly
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```



Relação parent > child(s)

Em flex-direction: row, podemos também alinhar verticalmente os nossos filhos utilizando a propriedade align-items.

```
.css
.container {
    display: flex;
    align-items: flex-start || center || flex-end;
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```

Relação parent > child(s)

Quando os nossos filhos ultrapassam a largura do pai, podemos construír um fluxo que nos permite agrupar os filhos num sistema de grelha. Por default o flex-wrap é nowrap.

```
.css
.container {
    display: flex;
    flex-wrap: nowrap;
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```

flex-wrap: nowrap;

flex-wrap: wrap;

Relação parent > child(s)

Para além de seguir um fluxo em eixo X (flex-direction: row), podemos também alterar o fluxo para o eixo Y (flex-direction: column)

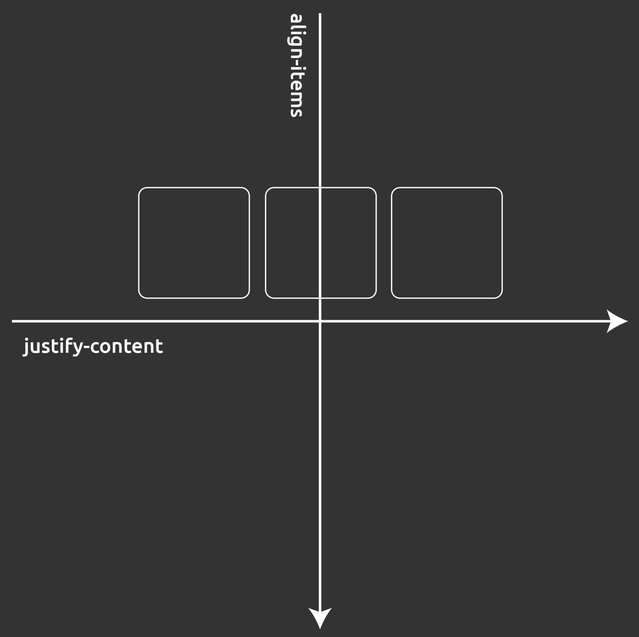
```
.css
.container {
    display: flex;
    flex-direction: column;
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```

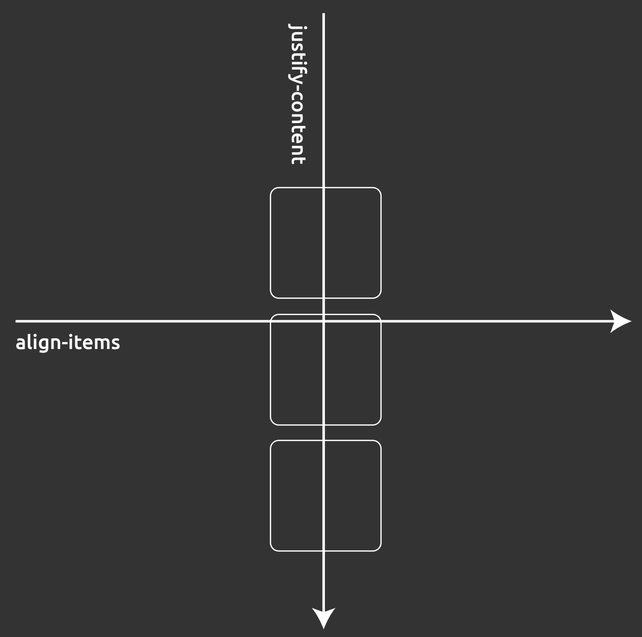
Relação parent > child(s)

Quando invertemos o eixo principal, as propriedades justify-content e align-items também alteram a forma como trabalham.

flex-direction: row



flex-direction: column



Relação parent > child(s)

Aquando em flex-direction: row, podemos alinhar agrupar os nossos filhos mediante o eixo Y.

A propriedade align-content alinhas os items, é similar ao justify-content, mas na vertical.

```
.css
.container {
    display: flex;
    flex-wrap: wrap;
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```

align-conten: space-between

Relação parent > child(s)

Aquando em flex-direction: row, podemos alinhar agrupar os nossos filhos mediante o eixo Y.

A propriedade align-content alinhas os items, é similar ao justify-content, mas na vertical.

```
.css
.container {
    display: flex;
    flex-wrap: wrap;
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```

align-content: flex-start

Relação parent > child(s)

Aquando em flex-direction: row, podemos alinhar agrupar os nossos filhos mediante o eixo Y.

A propriedade align-content alinhas os items, é similar ao justify-content, mas na vertical.

```
.css
.container {
    display: flex;
    flex-wrap: wrap;
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```

align-content: flex-end

Relação parent > child(s)

Aquando em flex-direction: row, podemos alinhar agrupar os nossos filhos mediante o eixo Y.

A propriedade align-content alinhas os items, é similar ao justify-content, mas na vertical.

```
.css
.container {
    display: flex;
    flex-wrap: wrap;
}
```

```
.html
<div class=“container>
    <div class=”col”></div>
    <div class=”col”></div>
</div>
```

align-content: center

Relação child(s) > parent

Quando falamos diretamente com os filhos, podemos alterar o posicionamento e a ordem dos mesmo através das propriedades align-self e order.

```
.css
.container {
    display: flex;
    flex-wrap: wrap;
}
```

```
.html
<div class=“container>
    <div class=”col col-1”></div>
    <div class=”col col-2”></div>
    <div class=”col col-2”></div>
</div>
```

Relação child(s) > parent

Quando falamos diretamente com os filhos, podemos alterar o posicionamento e a ordem dos mesmo através das propriedades align-self e order.

```
.css
.container {
    display: flex;
    flex-wrap: wrap;
}

.col-2 {
    order: 3;
}
```

```
.html
<div class=“container>
    <div class=”col col-1”></div>
    <div class=”col col-2”></div>
    <div class=”col col-2”></div>
</div>
```

Relação child(s) > parent

Quando falamos diretamente com os filhos, podemos alterar o posicionamento e a ordem dos mesmo através das propriedades align-self e order.

```
.css
.container {
    display: flex;
    flex-wrap: wrap;
}

.col-2 {
    order: 3;
    align-self: flex-end;
}
```

```
.html
<div class=“container>
    <div class=”col col-1”></div>
    <div class=”col col-2”></div>
    <div class=”col col-2”></div>
</div>
```

Relação child(s) > parent

Podemos também formatar a largura do elemento de forma flexível.

Por exemplo, ao aplicar flex-grow: 1; a .col-2 irá ocupar todo o restante espaço disponível no seu fluxo.

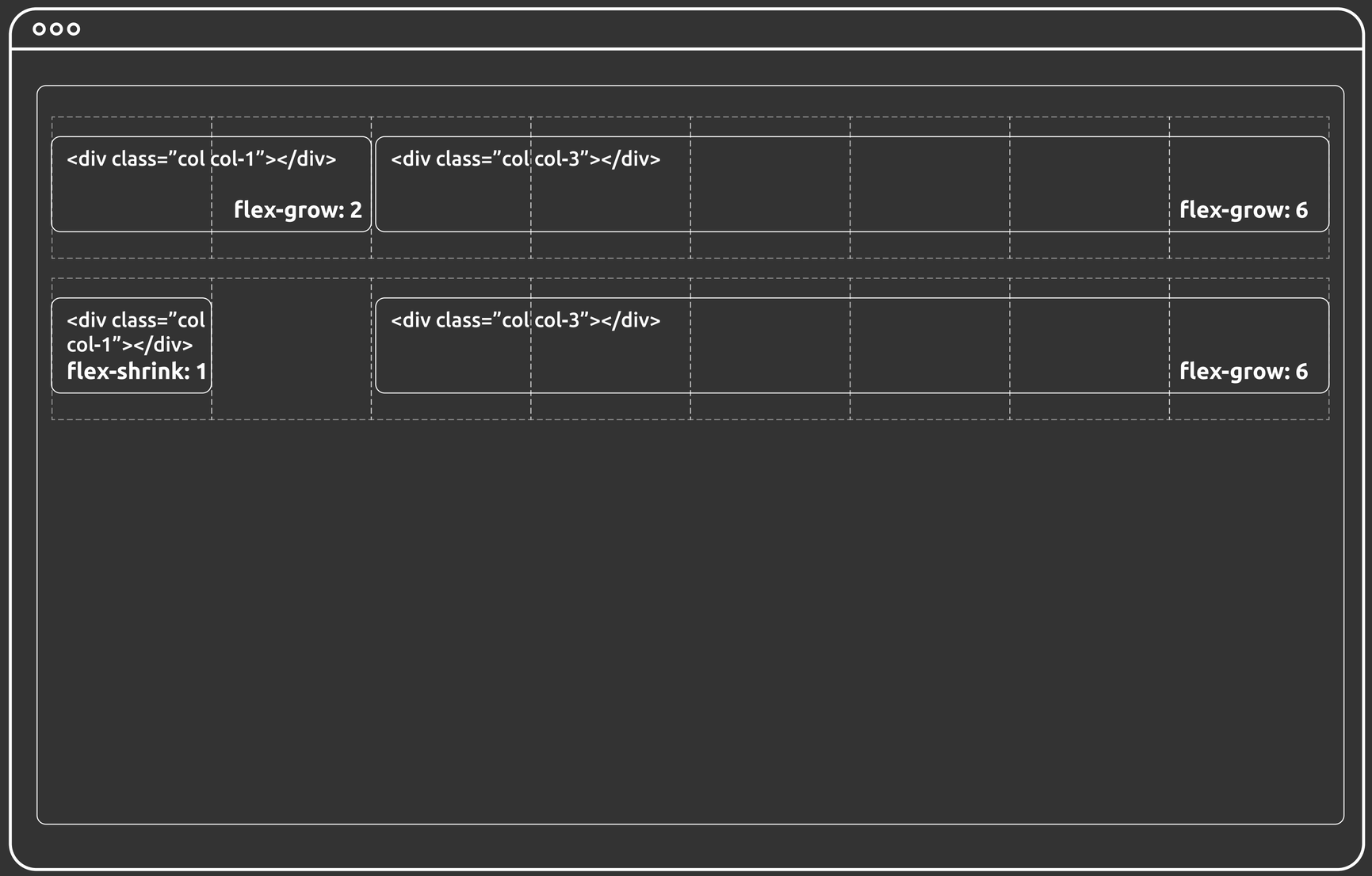
```
.css
.container {
    display: flex;
    flex-wrap: wrap;
}

.col-2 {
    order: 3;
    flex-grow: 1
}
```

```
.html
<div class=“container>
    <div class=”col col-1”></div>
    <div class=”col col-2”></div>
    <div class=”col col-2”></div>
</div>
```

Relação child(s) > parent

O flex-grow tem também o contrário que é o flex-shrink.

Estas duas propriedades trabalham dividindo os filhos por largura proporcionais, em que cada um ocupa n proporções que lhe são atribuídas.



Exercício