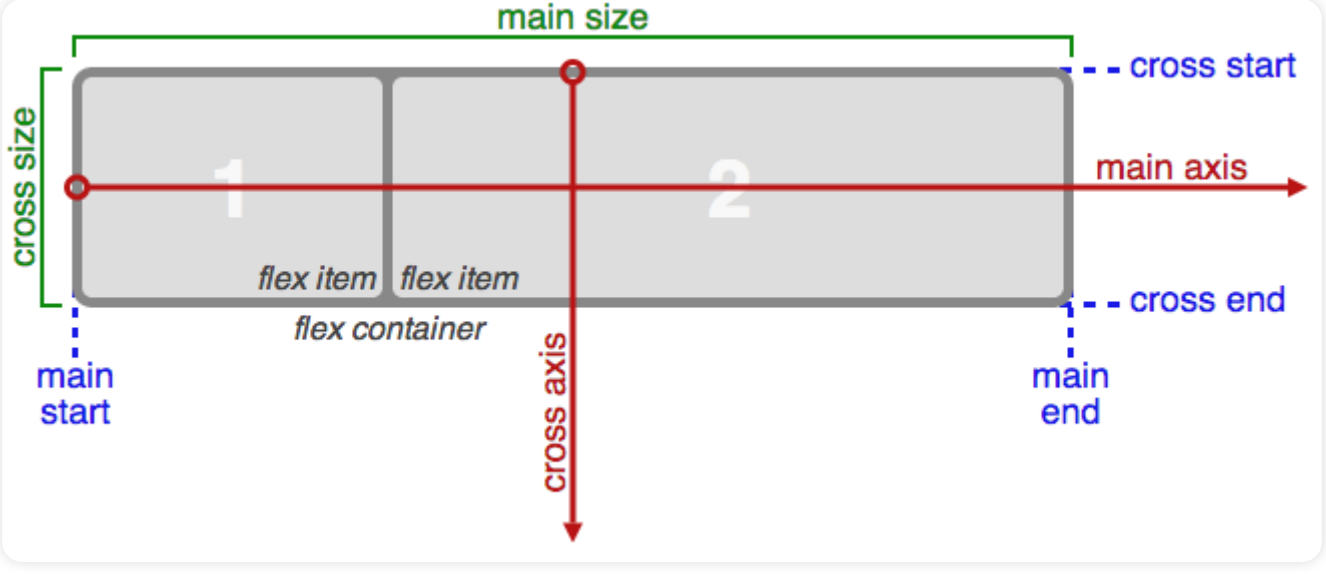
```
display: flex;
```

För att fördela `flex-items` säger vi hur de ska fördelas på huvudaxeln samt cross axis



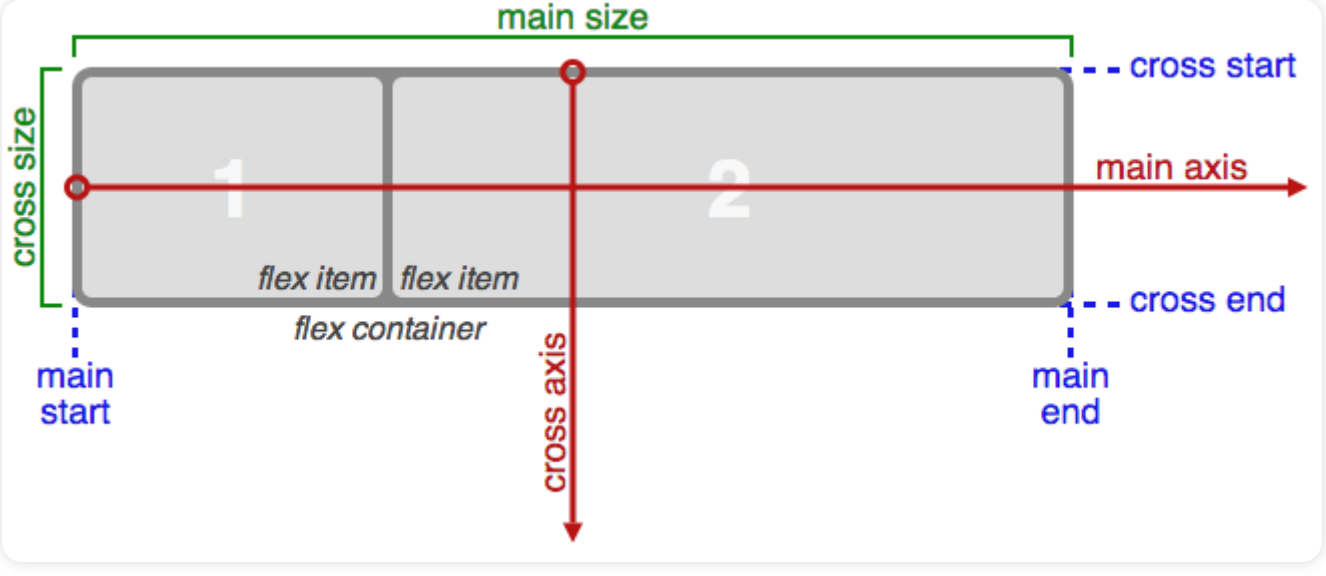
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

```html
<div class="parent">
  <div class="child"></div>
  <div class="child"></div>
</div>
```

```css
.parent{
  display: flex;
}
.child{
  flex: 1;         /* every child should take equal space */
  height: 200px; /* we must also have a height */
}
```

`justify-content` placerar ut flex items på huvudaxeln

`align-items` placerar ut flex items längs den motsatta axeln



CSS-tricks: A Complete Guide To Flexbox

SELECTORS

- `<tag>`
- `.class`
- `#id`

```
<article class="post">
  <h1> Title </h1>
  <p class="post-body">This is <em>some</em> text</p>
</article>
```

```
.post {}                  /* 0, 0, 1, 0 */
.post h1 {}               /* 0, 0, 1, 1 */
.post h1 em {}            /* 0, 0, 1, 2 */
.post .post-body em {} /* 0, 0, 2, 1 */
```

Välja flera element: `,`

```
h1, h2, h3, h4 {}
```

Välja flera element inuti en klass

```
.post h1, h2, h3, h4 {}
```

Välja ett element med en klass (`&&`)

```
div.grid {}
```

Går för det mesta att undvika

Gör en stark koppling mellan class och element, vi kan behöva använda klassen på ett annat element

```html
<header class="header">
  <h1 class="header-title"> Title </h1>
</header>
```

```css
.header h1 {}     /* 0,0,1,1 */
```

```css
.header-title {} /* 0,0,1,0 */
```

```css
.header * {}      /* 0,0,1,0 */
```

Asterisk (`*`) väljer allt

Descendant combinator

```
.header h1 {}     /* 0,0,1,1 */
```

Väljer alla `h1` inuti `.header`

Child combinator

```
div > p {}
```

Väljer alla element som är direkta barn av ett element

```css
ul > li {
  border: 3px solid red;
}
```

- Unordered list item
- Unordered list item
  1. Ordered List item
  2. Not direct child

Adjacent sibling combinator

```
div + p {}
```

The + combinator selects adjacent siblings. This means that the second element directly follows the first, and both share the same parent.

```
li + li {
  border: 3px solid red;
}
```

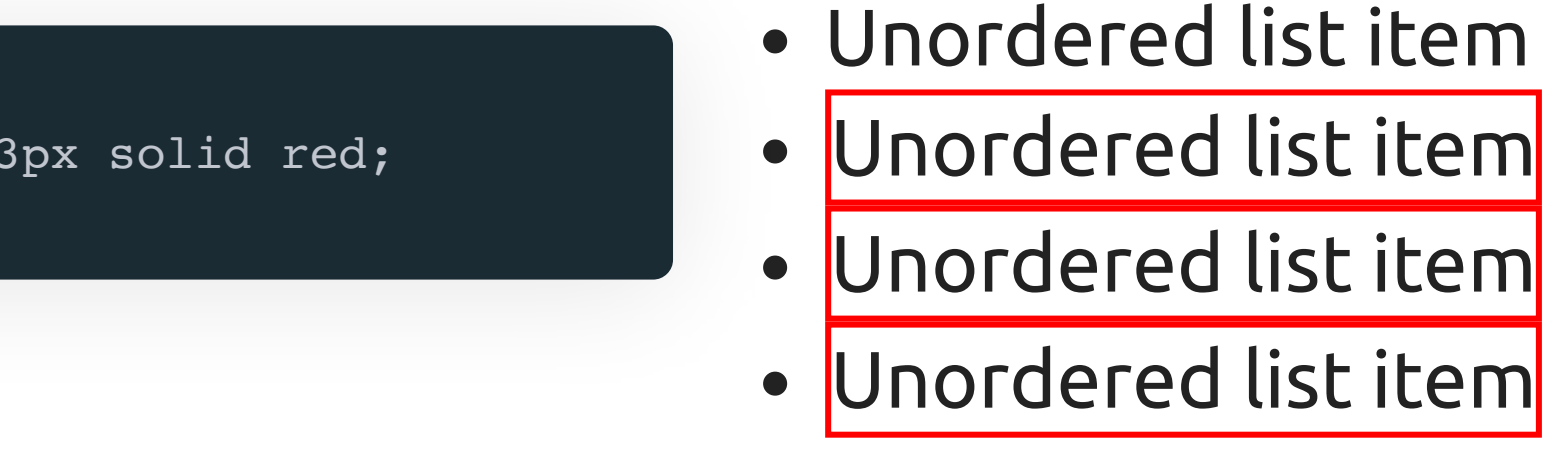
- Unordered list item
- Unordered list item
  1. Ordered List item
  2. Not direct child

General sibling combinator

```
div ~ p {}
```

The ~ combinator selects siblings. This means that the second element follows the first (though not necessarily immediately), and both share the same parent.

```
li ~ li {
  border: 3px solid red;
}
```



PSEUDO-CLASSES & PSEUDO-ELEMENTS

Pseudo

not actually but having the appearance of; pretended; false or spurious; sham. almost, approaching, or trying to be.

Pseudo-classes

- `:link`
- `:hover`
- `:visited`

Definierar olika `state`, i vilket tillstånd element är

```
h1 {
  color: red;
}
h1:hover {
 color: green;
}
```

Vi kan även använda pseudo-classes för att välja specifika element baserat på deras relation till resten av DOM

T.ex. välja utifrån vilken ordning barnet kommer i

- `:nth-child()`
- `:only-child`
- `:last-child`
- `:nth-last-child()`

```
li:nth-child(2){
 color: red;
}
```

- First li-item
- Second li-item
- Third li-item
- Fourth li-item

Pseudo-elements

- `::after`
- `::before`
- `::first-letter`
- `::first-line`

Funkar även med enkla `:`

`::after` / `::before` lägger till innehåll efter eller före ett element

```
div::after{
  content: "";
}
```

Gör det möjligt att lägga till element efter andra element

Ha inget viktigt innehåll i `::after/::before`

Innehållet är inte tillgängligt, går inte att markera

Vi frångår vår uppdelning av innehåll/presentation

Dock väldigt bra metoder, använd gärna

Validering

CSS validator

HTML validator