basic grid

```
.container {
  display: grid;
  grid-template-columns: 100px auto 100px;
  grid-gap: 20px;
}
```

grid-auto-flow

```
.container {
  display: grid;
  grid-gap: 20px;
  grid-template-columns: 400px 200px;
  grid-auto-flow: column; /* vs row - won't wrap */
  /*
    grid-auto-columns: 200px; sizes the implicitly created cols
  */
}

/* grid-auto-flow: dense; fills in empty spaces */
```

fractional unit

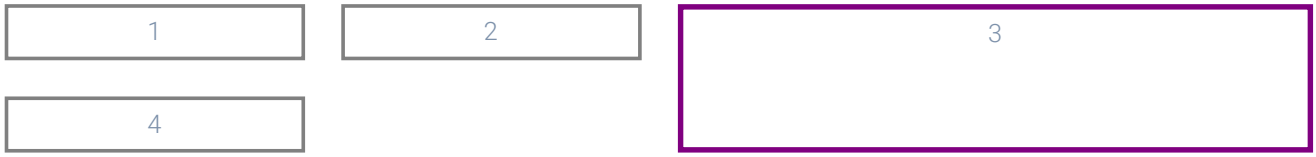
```
.container {
  display: grid;
  grid-gap: 20px;
  grid-template-columns: 400px 1fr 2fr;
  grid-template-rows: 1fr 2fr;
}
```

repeat()

```
.container {
  display: grid;
  grid-gap: 20px;
  grid-template-columns: repeat(3, 1fr 2fr);
}
```

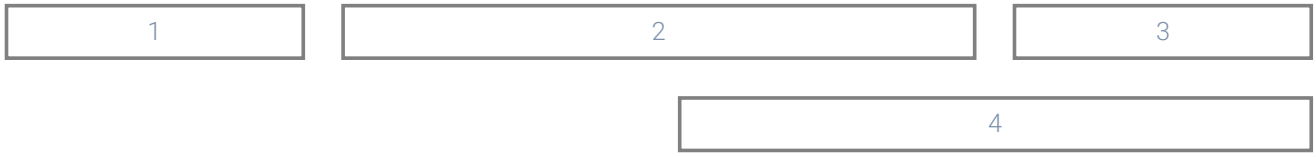
span

```
.container {
  display: grid;
  grid-gap: 20px;
  grid-template-columns: repeat(4, 1fr);
}
.item-3 {
  grid-row: span 2;
  grid-column: span 2; /* same as grid-column-start: span 2; grid-column-end: auto; */
}
```



placement

```
.item-2 {
  grid-column: 2 / span 2;
}
.item-4 {
  grid-column-start: 3;
  grid-column-end: -1;
}
```



auto-fill

```
.container {
  grid-template-columns: repeat(auto-fill, 150px); /* expands or shrinks when necessary */
}

.item-4 {
  grid-column: auto / -1;
}
```

auto-fit

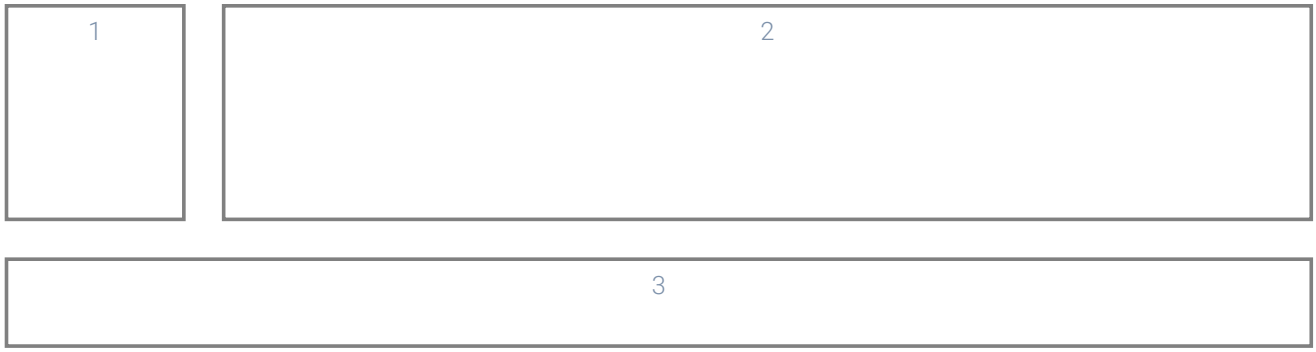
```
.container {
  grid-template-columns: repeat(auto-fit, 150px); /* ends cols at end of elements */
}
```

minmax()

```css
.container {
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr)); /* responsive to screen */
}
```

areas

```css
.container {
  grid-template-columns: 100px 1fr 100px;
  grid-template-rows: repeat(3, 50px);
  grid-template-areas:
    "sidenav   main   main"
    "sidenav   main   main"
    "footer    footer footer"
  /* combines well with media queries */
}
.sidenav {
  grid-area: sidenav;
}
.main {
  grid-area: main;
}
.footer {
  grid-area: footer;
}
```



named lines

```
.container {
  grid-template-columns: [content-start content-1] 75px [content-2] 1fr [content-3] 200px [content-end];
  grid-template-rows: [content-top] repeat(3, auto) [content-end];
}

.item-3 {
  border: 3px solid purple;
  grid-column: content-2;
  grid-row: content-top / content-end;
}
```

---

reordering

- default order is 0
- change doesn't get picked up by screenreader

```
.item-3 {
  order: -1;
}
```

---

alignment & centering

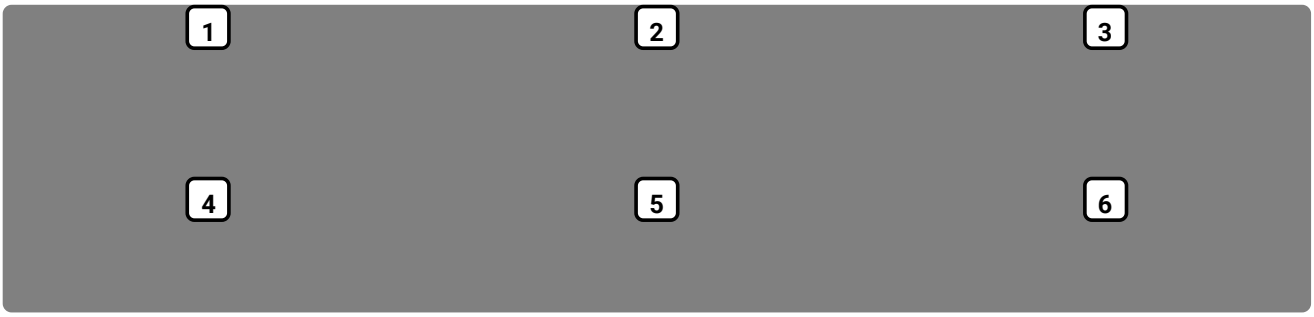
- justify = row axis (left to right)
- align = column axis (top to bottom)

justify-items

- justify-items defaults to 'stretch' otherwise fits content/explicit width of item
- also 'start', 'end'

```
.container {
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(2, 75px);

  justify-items: center;
}
```



align-items

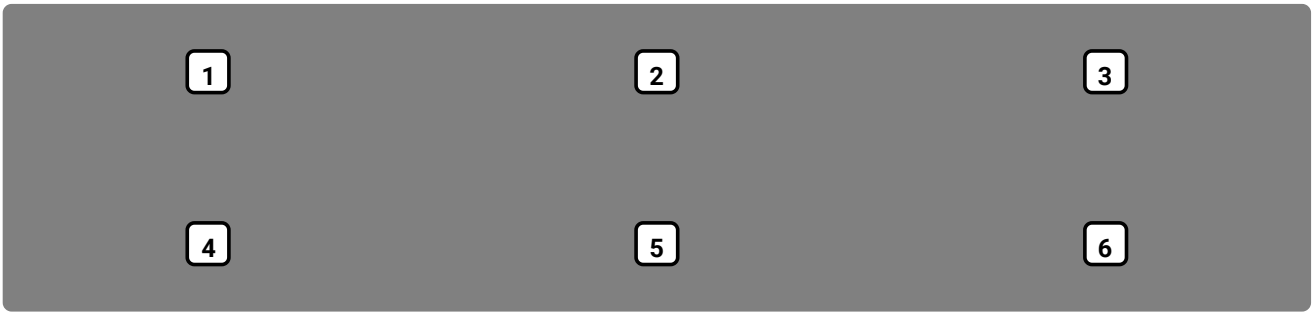
- vertical positioning
- stretch, center, start, end

```
.container {
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(2, 75px);
  justify-items: end;
  align-items: end;
}
```

place-items

- limited browser support
- align, justify

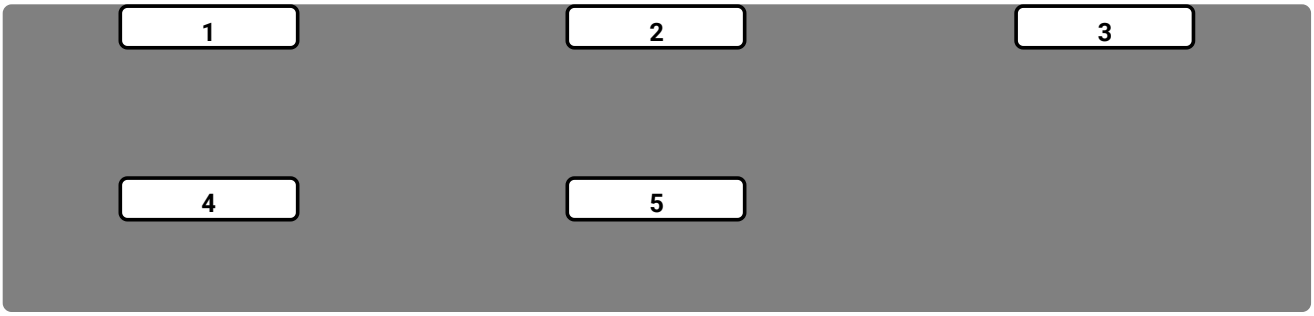
```
.container {
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(2, 75px);
  place-items: center center;
}
```



justify-content and align-content

- start, end, center, space-around, space-between

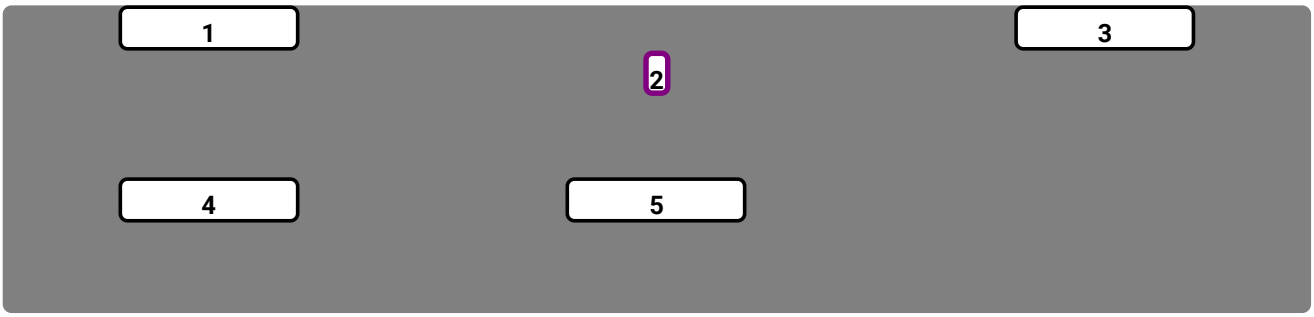
```
.container {
  grid-template-columns: repeat(5, 100px);
  grid-template-rows: repeat(2, auto);
  justify-content: space-around;
  /* align-content: space-between; works if there's a fixed height on the grid */
}
```



align-self and justify-self

- start, end, center

```css
.container {
  grid-template-columns: repeat(5, 100px);
  grid-template-rows: 75px;
  justify-content: space-around;
}
.item-2 {
  justify-self: center;
  align-self: center;
}
```



term	definition - main source
tracks	the lines, not the rows/cols
grid-gap	sets the gaps (gutters) between rows and columns. Shorthand for row-gap and column-gap.
row-gap	sets the size of the gap (gutter) between an element's grid rows.
grid-template-columns	defines the line names and track sizing functions of the grid columns.
grid-template-rows	defines the line names and track sizing functions of the grid rows.
grid-template-areas	names areas within a string separated by spaces only
grid-area	assigns a name to an area to go with the above
grid-auto-columns	CSS property specifies the size of an implicitly-created grid column track or pattern of tracks.
grid-auto-rows	CSS property specifies the size of an implicitly-created grid row track or pattern of tracks.

term	definition - main source
grid-auto-flow	controls how the auto-placement algorithm works, specifying exactly how auto-placed items get flowed into the grid. - row/column/dense
auto	keyword that is identical to maximal content if it's a maximum. As a minimum it represents the largest minimum size (as specified by min-width/min-height) of the grid items occupying the grid track.
fr unit	fractional (amount of space remaining once elements are laid out) e.g. 1fr is for 1 part of the available space.
repeat()	CSS function represents a repeated fragment of the track list, allowing a large number of columns or rows that exhibit a recurring pattern to be written in a more compact form. w grid-template-rows/columns
implicit grid/tracks	created by the browser e.g. wrapping row (dotted lines on Firefox devtools)
explicit grid/tracks	created by our code (solid line in Firefox)
auto-fit	FILLS the row with as many columns as it can fit. So it creates implicit columns whenever a new column can fit, because it's trying to FILL the row with as many columns as it can. The newly added columns can and may be empty, but they will still occupy a designated space in the row. source
auto-fill	FITS the CURRENTLY AVAILABLE columns into the space by expanding them so that they take up any available space. The browser does that after FILLING that extra space with extra columns (as with auto-fill) and then collapsing the empty ones. source
minmax()	CSS function defines a size range greater than or equal to min and less than or equal to max.
fit-content()	CSS function clamps a given size to an available size according to the formula min(maximum size, max(minimum size, argument)). %/px/ch/vw/cm
subgrid	grid will adopt the spanned portion of its parent grid in that axis. Rather than being specified explicitly, the sizes of the grid rows/columns will be taken from the parent grid’s definition.