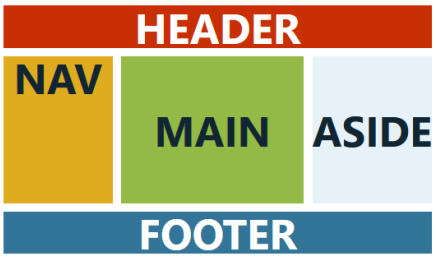
Grid Layout -Spickzettel



Das **CSS Grid Layout Modul** ermöglicht responsive Flexibilität mit nur wenigen Festlegungen, was spätere Änderungen sehr erleichtert.

display

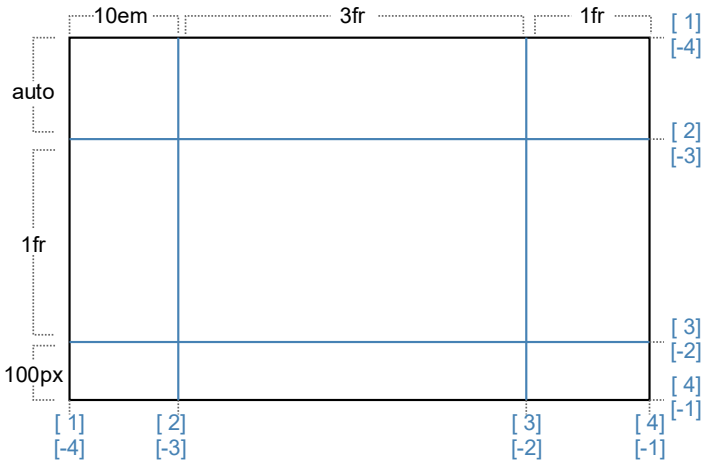
```
.container {
  display: grid;    |inline-grid |subgrid
}
```

Grid-Container werden durch display: grid bzw. inline-grid und subgrid definiert und können eine Breite, bzw. Mindest- oder Maximalbreite haben.

Beachte: Dieses Raster wirkt nur auf direkte Kind-Elemente des Grid-Containers.

Explizite Raster festlegen

```
body {
  grid-template-columns: 10em 3fr 1fr;
  grid-template-rows: auto 1fr 100px;
}
```



Die Eigenschaft **grid-template-columns** legt Anzahl, Größe und optional auch die Benennung von Spalten - **grid-template-rows** von Zeilen fest.

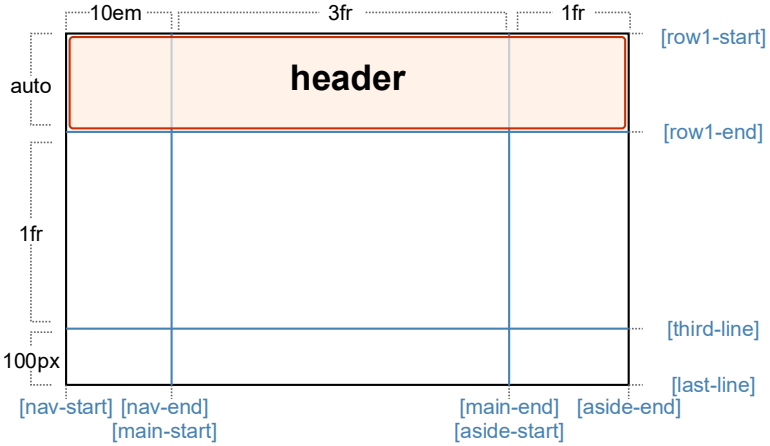
Die Werte können normale Längenangaben, Prozentwerte – oder mit **fr** Bruchteile der verfügbaren Viewport-Maße sein.
Spalten und Zeilen werden durch Rasterlinien getrennt. Sie sind fortlaufend nummeriert, negative Werte werden vom Ende her gezählt.

Rasterlinien können auch benannt werden. Innerhalb der eckigen Klammern sind auch mehrere Namen möglich.

```
.container {
  display: grid;
  grid-template-columns:
    [nav-start] 10em [nav-end main-start] 3fr [main-end aside-start] 1fr [aside-end];
  grid-template-rows:
    [row1-start] auto [row1-end] 1fr [third-line] 100px [last-line];
  gap: 0;
}
```

```
body{
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  grid-template-areas: "head head"
                       "nav  nav"
                       "main main"
                       "foot foot";
}
@media (min-width: 30em) {
  body{
    grid-template-columns: repeat(3, 1fr);
    grid-template-areas: "head head head"
                         "nav  news aside"
                         "main main main"
                         "foot foot foot";
  }
}
```

grid-template-areas legen in einer Liste das Schena der Seite fest.
Mit *media queries* können so unterschiedliche Layouts für größere Viewports festgelegt werden.



Implizite Raster festlegen

```
.container {
  display: grid;
  grid-auto-flow: column;
  grid-auto-columns: minmax(150px, 1fr);
}
```

grid-auto-columns legt Größe von Rasterzellen fest.
Das umgebende Grid wird implizit vom Browser erzeugt

Rasterzellen anordnen

```
nav {
  grid-column-start: 1;
  grid-column-end:   2;
  grid-row-start:    [row1-start];
  grid-row-end:      3;
}
header {
  grid-column-start: span 3;
}
```

```
nav {
  grid-column: 1 / 2;
  grid-row:    1 / -1;
}
header {
  grid-column: span 3;
}
```

```
section {
  grid-area: 2 / 2 / auto / span 3;
}
header {
  grid-area: head;
}
```

grid-column-start

grid-column-end

legen Beginn und Ende der Rasterzelle fest.

- Ganzzahl. Negative Werte zählen vom Ende des expliziten Grids her.
- benannter Name der Rasterlinie

Schlüsselwort `span` mit Anzahl der Zellen, die überspannt werden sollen.

grid-column: Anfangs- und Endwerte für Spalte

grid-row: Anfangs- und Endwerte für Zeile

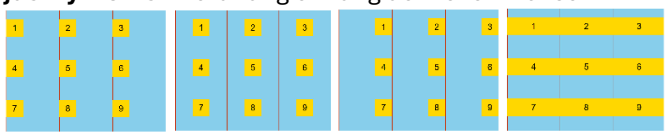
grid-area:

- 4 Werte für Anfangs- und Endlinien
- benannter Name aus **`grid-template-areas`**

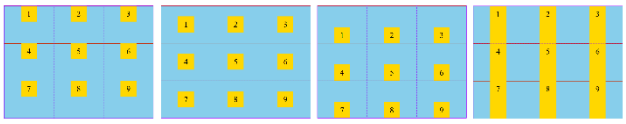
Rasterzellen ausrichten

Im Grid-Container

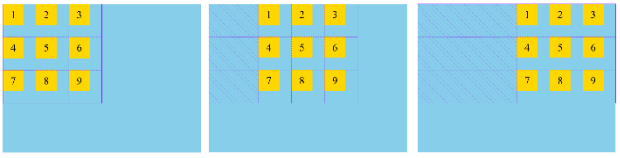
justify-items Anordnung entlang der Zeilen-Achse



align-items Anordnung entlang der Spalten-Achse



justify-content positioniert Raster entlang der Grid-Achse, wenn diese kleiner als der verfügbare Platz ist

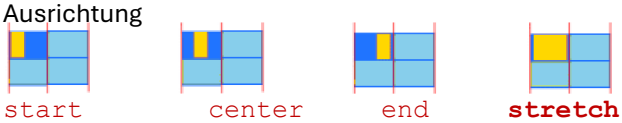


align-content positioniert Raster quer zur Grid-Achse, wenn diese kleiner als der verfügbare Platz ist

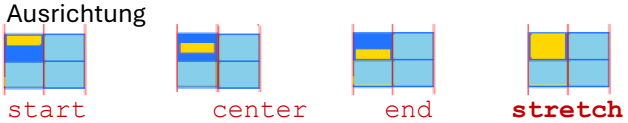
Beachte: Normalerweise nehmen Rasterzellen (*items*) den gesamten verfügbaren Platz ein **(`auto|stretch`).**

Im Raster-Element

justify-self Anordnung eines Items entlang der Ausrichtung



align-self Anordnung eines Items quer zur Ausrichtung



CC-BY-SA 3.0 (de) 2025 selfhtml.org selfhtml.org/wiki/CSS/Tutorials/Grid