Flexbox-Spickzettel

CSS Flexible Box Layout ordnet Elemente eindimensional an, verteilt freien Platz und richtet Items aus.

Flex-Container und Flex-Items

```
.container {
  display: flex; /* oder inline-flex */
  gap: 1rem;
}
```

`display: flex` Erzeugt einen Flex-Container.
`gap` Abstand zwischen Flex-Items.

Nur direkte Kindelemente sind Flex-Items; Verschachteltes braucht einen eigenen Container.

Hauptachse und Querachse

Die **Hauptachse** folgt `flex-direction`. `justify-content` verteilt die Items auf dieser Achse.

Die **Querachse** steht senkrecht zur Hauptachse. `align-items` richtet Items dort aus.

`flex-direction` legt die Hauptachse fest: `row` folgt der Schreibrichtung, `column` läuft senkrecht.

Richtung und Umbruch

```
.container {
  flex-flow: row wrap;
  /* Richtung und Umbruch */
}
```

`flex-flow` kombiniert `flex-direction` und `flex-wrap`: Richtung, Umbruch.
`row wrap` ordnet in Schreibrichtung an und bricht um.
Standard: `row nowrap`.

Horizontal nach rechts

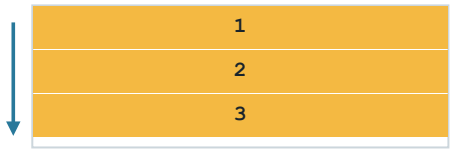
`flex-direction: row;`

Horizontal nach links

`flex-direction: row-reverse;`

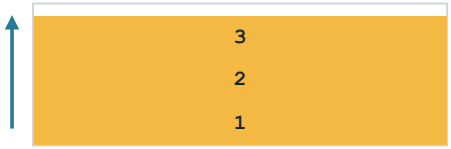
Vertikal nach unten

`flex-direction: column;`



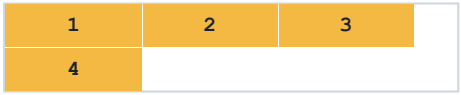
Vertikal nach oben

`flex-direction: column-reverse;`



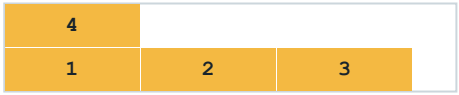
Umbruch

`flex-wrap: wrap;`



Umbruch umkehren

`flex-wrap: wrap-reverse;`



Responsiv ohne Media Query

```
.cards {
  display: flex;
  flex-wrap: wrap;
  gap: 1rem;
}
.cards > * { flex: 1 1 8rem; }
```

`flex-basis: 8rem` setzt die Ausgangsbreite. Umbruch entsteht durch Inhalt, nicht durch Gerätebreiten.

Intrinsisches Mehrspaltenlayout

`#card2 { flex: 1 1 14rem; }`

- `flex-basis` 14rem setzt den Umbruchpunkt.

Freien Platz verteilen: `justify-content` (Hauptachse)

Am Anfang

`justify-content: flex-start;`

In der Mitte

`justify-content: center;`

Am Ende

`justify-content: flex-end;`

Nur dazwischen

`justify-content: space-between;`

Halber Randabstand

`justify-content: space-around;`

Gleicher Abstand

`justify-content: space-evenly;`

Items ausrichten: `align-items` (Querachse)

Am Anfang

`align-items: flex-start;`

Mittig

`align-items: center;`

Am Ende

`align-items: flex-end;`

Text an der Grundlinie

`align-items: baseline;`

Strecken

`align-items: stretch;`

Mehrere Zeilen verteilen

`align-content: center;`

`align-content` verteilt mehrere Flex-Zeilen. Wirkt nur mit `flex-wrap` und freiem Platz.

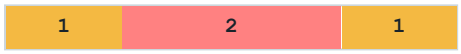
Flex-Items: Größe und Reihenfolge

```
.item {
  flex: 1 1 12rem;
  /* wachsen schrumpfen Ausgangsgröße */
}
```

`flex-grow` Verteilt positiven freien Platz. Standard: `0`.
`flex-shrink` Schrumpfen bei Platzmangel. Standard: `1`.
`flex-basis` Ausgangsgröße vor dem Verteilen. Standard: `auto`.

Stärker wachsen

`flex-grow: 2;`



Item 2 erhält den doppelten Anteil.

Ein Item ausrichten

`align-self: flex-end;`

Visuell nach vorn

`order: -1;`

Item 3 rückt nur visuell nach vorn.

Barrierefreiheit: `order` ändert nur die Darstellung, nicht Dokument-, Vorlese- oder Tab-Reihenfolge.

Wichtige Praxismuster

Gebräuchliche Kurzformen

`flex: 1` `1 1 0%`: gleicher Anteil am freien Platz.
`flex: auto` `1 1 auto`: Inhalt oder Breite bestimmen die Ausgangsgröße.
`flex: none` `0 0 auto`: wächst und schrumpft nicht.

Schrumpfen ermöglichen

```
.item {
  min-width: 0;
  min-height: 0;
}
```

Lange Wörter, URLs oder Code verhindern sonst oft das Schrumpfen.

Abstände mit `gap`

`gap` Zeilen- und Spaltenabstand gemeinsam.
`row-gap` Abstand in Zeilenrichtung.
`column-gap` Abstand in Spaltenrichtung.

`flex-basis` und `width`

`flex-basis` bestimmt die Ausgangsgröße auf der Hauptachse. Bei `auto` gelten `width` oder `height`.

Prozentwerte brauchen eine bestimmte Hauptgröße, sonst wirken sie wie `auto`.

Item ans Ende schieben

`margin-inline-start: auto;`

Der automatische Außenabstand nimmt freien Platz ein.

Logische Richtungen

`start` und `end` folgen Schreib- und Textrichtung; `row` hängt von `direction` und `writing-mode` ab.