LU01e - Display

Die CSS-Anweisung „display“ bestimmt, wie ein HTML-Element auf der Webseite dargestellt wird. Sie kontrolliert das Layout und das Verhalten des Elements im Browserfenster.

Die verschiedene Display-Werte

Die Auswahl des richtigen „display“-Werts hängt von den Anforderungen des Layouts und dem gewünschten Verhalten der Elemente ab. Durch die geschickte Verwendung von „display“ können Entwickler das Erscheinungsbild und die Funktionalität von Webseiten effektiv steuern. Hier ist eine Übersicht über die wichtigsten Werte, die für „display“ verwendet werden können:

Wert	Beschreibung
block (Block-Element)	Block-Elemente nehmen die gesamte verfügbare Breite ihres übergeordneten Elements ein und beginnen auf einer neuen Zeile. Beispiele für Block-Elemente sind `<div>`, `<p>`, `<h1>`-`<h6>`, `<header>`, `<footer>`.
inline (Inline-Element)	Inline-Elemente nehmen nur so viel Breite wie nötig ein und bleiben im selben Absatz oder auf derselben Zeile wie benachbarte Inline-Elemente. Beispiele für Inline-Elemente sind `<span>`, `<a>`, `<b>`.
inline-block (Inline-Block-Element)	Kombiniert Eigenschaften von Block- und Inline-Elementen. Es nimmt nur so viel Breite wie nötig ein und kann dennoch mit anderen Inline- oder Block-Elementen in derselben Zeile stehen. Häufig verwendet für Elemente, die als Block formatiert werden müssen, aber in einer Zeile bleiben sollen, z. B. `<button>`.
none (Ausgeblendetes Element)	Das Element wird nicht gerendert und nimmt keinen Platz in der Layout-Struktur ein. Es ist praktisch, um Elemente auszublenden oder dynamisch zu verbergen. Verwendet für Elemente, die vorübergehend nicht angezeigt werden sollen, z. B. auf verschiedenen Browsergrössen.
flex (Flexbox-Element)	Aktiviert das Flexbox-Layoutmodell für das Element, mit dem Sie flexible Layouts erstellen können, indem Sie den Inhalt auf verschiedene Weise anordnen und ausrichten. Flexbox ist besonders nützlich, um komplexere Layouts zu erstellen und das Responsiveness von Websites zu verbessern.
grid (Grid-Element)	Aktiviert das Grid-Layoutmodell für das Element, mit dem Sie Inhalte in einer zweidimensionalen Rasteranordnung organisieren können. Grid ist leistungsstark für die Erstellung von Layouts mit mehreren Spalten und Zeilen und bietet präzise Kontrolle über die Positionierung von Elementen.

Eine komplette Liste aller verfügbaren Werte finden Sie hier.

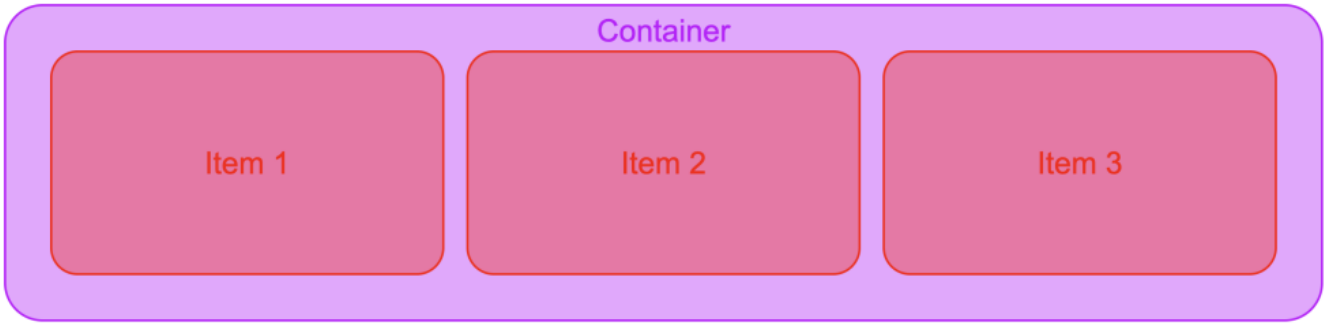
CSS Flexbox

CSS Flexbox ist ein leistungsstarkes Layoutmodell in CSS, das entwickelt wurde, um das Design von Webseiten und Webanwendungen zu vereinfachen und flexible, dynamische Layouts zu ermöglichen. Es bietet eine effiziente Möglichkeit, Elemente innerhalb eines Container-Elements zu organisieren,

auszurichten und zu verteilen, unabhängig von ihrer Grösse oder Reihenfolge.

Grundkonzept

Die Grundkonzepte von Flexbox umfassen das Container-Element und die darin enthaltenen Items. Der Flex-Container fungiert als Rahmen für das Flexbox-Layout und organisiert die Flex-Items innerhalb. Diese Items können horizontal oder vertikal angeordnet werden, je nach Ausrichtung der Hauptachse. Die Hauptachse definiert die Richtung, in der die Flex-Items im Container angeordnet sind, während die Querachse senkrecht dazu steht und die Ausrichtung der Items in Bezug auf die Hauptachse beeinflusst. Das Verständnis dieser Grundkonzepte ist entscheidend für die effektive Nutzung von Flexbox zur Erstellung dynamischer und flexibler Layouts in CSS.



Eigenschaften des Containers

Eigenschaft	Beschreibung
flex-direction	Bestimmt die Richtung der Hauptachse. Mögliche Werte sind: row, row-reverse, column, column-reverse.
flex-wrap	Kontrolliert, ob die Items in einer einzigen Zeile bleiben oder bei Bedarf in mehrere Zeilen umgebrochen werden. Mögliche Werte sind: nowrap, wrap, wrap-reverse.
justify-content	Definiert die Ausrichtung der Items entlang der Hauptachse. Dies beeinflusst, wie überschüssiger Platz auf der Hauptachse verteilt wird.
align-items	Bestimmt die Ausrichtung der Items entlang der Querachse, wenn sie auf der Hauptachse nicht den gesamten verfügbaren Platz einnehmen.
align-content	Steuert die Ausrichtung und Verteilung mehrerer Zeilen von Items entlang der Querachse.

Eigenschaften des Items

Eigenschaft	Beschreibung
order	Ändert die Reihenfolge der Items innerhalb des Flex-Containers. Items mit einer niedrigeren Reihenfolge werden zuerst angezeigt.
flex-grow	Legt fest, wie viel zusätzlichen Platz ein Flex-Item entlang der Hauptachse einnehmen kann, im Verhältnis zu anderen Flex-Items.
flex-shrink	Definiert, wie stark ein Flex-Item kleiner werden kann, um sich an den verfügbaren Platz anzupassen, im Verhältnis zu anderen Flex-Items.
flex-basis	Legt die bevorzugte Anfangsgrösse eines Flex-Items entlang der Hauptachse fest, bevor zusätzlicher Platz verteilt wird.
flex	Eine verkürzte Schreibweise für flex-grow, flex-shrink und flex-basis in einem Wert.

Eigenschaft	Beschreibung
align-self	Überschreibt die Ausrichtungsvorgaben des Flex-Containers für ein einzelnes Flex-Item.

CSS Grid

CSS Grid ist ein leistungsstarkes Layoutmodell in CSS, das entwickelt wurde, um das Design von Webseiten und Webanwendungen zu verbessern, indem es eine präzise und flexible Möglichkeit bietet, Elemente in einem zweidimensionalen Raster anzuordnen.

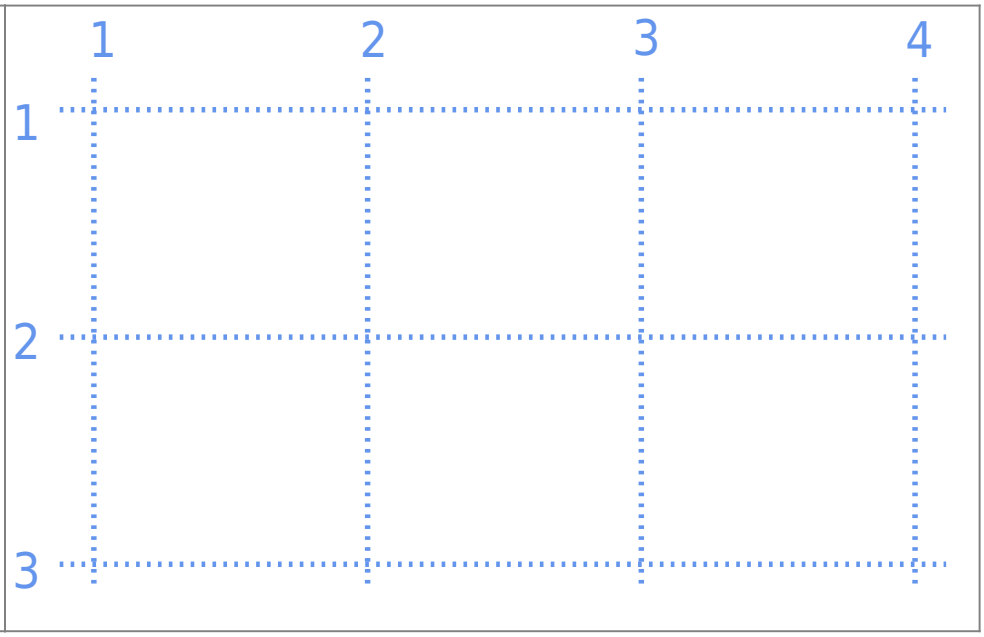
Ein HTML-Element wird durch die CSS-Eigenschaft `display: grid;` zu einem Grid-Container, der das Raster für das CSS Grid-Layout bildet. Die direkten Kind-Elemente dieses Containers werden als Grid-Items bezeichnet und innerhalb des Rasters positioniert. Der Raster bildet sich aus horizontalen und vertikalen Linien. Die Nummern der einzelnen Linien legen fest, wo ein Bereich anfängt und wo es endet.

Zeilen und Spalten

Mit grid-template-columns bzw. grid-template-rows legen wir die Höhe und Breite der Zeilen fest. Der Wert „fr“ (fraction, Anteil) definiert einen Bereich im Grid. Dieser kann auch durch fixe Werte wie `100px` ersetzt werden. Der Vorteil von „fr“ liegt in der responsivness. Die Breite des Anteils passt sich jeweils an die Gesamtbreite an.

Ein Beispiel mit zwei Zeilen und drei Spalten:

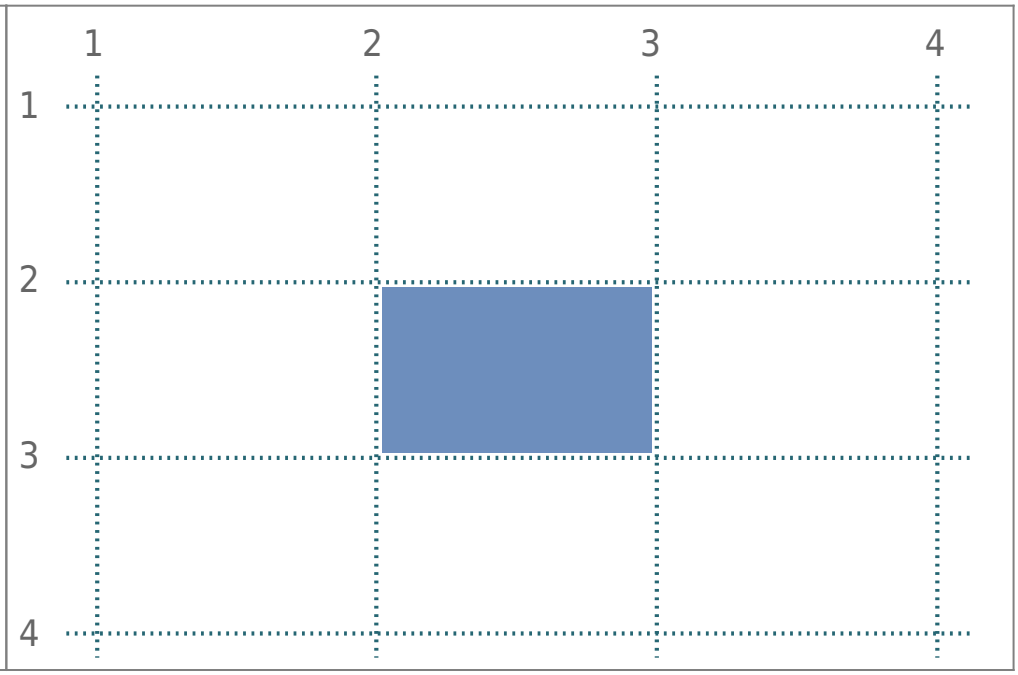
```
.grid-container {
    display: grid;
    grid-template-
columns: 1fr 1fr 1fr;
    grid-template-
rows: 1fr 1fr;
    width: 600px;
    height: 400px;
}
```



Platzieren von Elementen im Grid

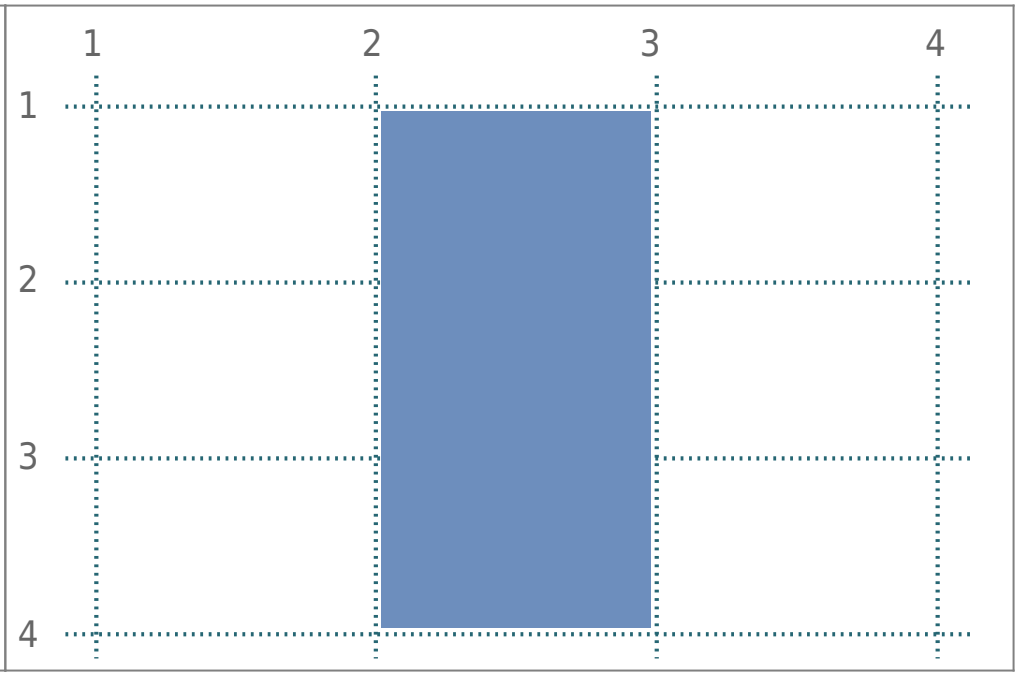
Nun können wir ein Element in diesem Raster platzieren, so wie wir es möchten. Dafür können wir mit `grid-column-start`, `grid-column-end`, `grid-row-start` und `grid-row-end` jeweils den Start und das Ende unseres Elements im Grid angeben. Wie im vorherigen Teil erwähnt, wird das Grid in Zahlen aufgeteilt. So kann man zum Beispiel in Element mittig von unserem Grid-Raster platzieren:

```
.grid-item {
   grid-column-start: 2;
   grid-column-end: 3;
   grid-row-start: 2;
   grid-row-end: 3;
   background-color: blue;
}
```



Es gibt auch eine Kurzform von der Schreibweise für die Reihen und Spalten. Mit den Anweisungen `grid-column` und `grid-row` können Sie jeweils getrennt mit einem `/` den Start und das Ende eines Elements im Raster angeben.

```
.grid-item {
   grid-column: 2/3;
   grid-row: 1/4;
}
```



Grid Areas

Mit `grid-template-areas` haben wir die Möglichkeit, verschiedene Bereiche in unserem Grid zu definieren. Mit `grid-area` können wir dann unsere Elemente innerhalb der vorgegebenen Bereichenen platzerein.

```
.grid-container {
   grid-column-template: "1fr 1fr 1fr";
   grid-template-areas: "header    header header"
                        "thumbnail text   links "
```

```
                        ".          text    .     ";
}

.article-title {
   grid-area: header;
}

.article-img {
   grid-area: thumbnail;
}
```

Ressourcen

- CSS-Tricks: A Complete Guide to Flexbox
- W3S: CSS Flexbox