

CSS-Grids

Ansätze für Seitenlayout

- Tabellen als Layouthilfe (ausgestorben)
- Float (noch verwendet)
- CSS-Frameworks wie Bootstrap (noch verwendet)
- Flexboxen (noch verwendet)

- **CSS-Grids (aktuellste Technik)**

unterstützt von Chrome, Firefox, Safari, Edge,...



Vorteil von CSS-Grids

Flexboxen: Anpassung der Elemente an verfügbaren Platz
entweder horizontal **oder** vertikal (zeilenbasiert,
eindimensional wie float)

CSS-Grids: Anpassung der Elemente an verfügbaren
Platz horizontal **und** vertikal (zweidimensional)

Beispiel

`<body>`

`<header> Überschrift </header>`

`<nav> Navigation </nav>`

`<main> Inhalt </main>`

`<footer> Footer </footer>`

`</body>`



```
body{height:100vh;margin:0;display:grid;}
```

Grid-Struktur - Parent-Eigenschaften

Titel:

Parent-Element: Grundlagen

Inhalt:

1. **display: grid;** - Aktiviert das Grid-Layout.
 2. **grid-template-columns:** Definiert die Spaltenbreite, z.B. 100px 30% 1fr auto.
 3. **grid-template-rows:** Definiert die Zeilenhöhe, z.B. 100px 100px 100px.
 4. **gap:** Definiert den Abstand zwischen den Grid-Elementen in Zeilen und Spalten (z.B. 20px 30px für row-gap und column-gap).
-

Beispiel

```
.container {  
    display: grid;  
    grid-template-columns: 100px 30% 1fr auto;  
    grid-template-rows: 100px 100px 100px;  
    gap: 20px 30px;  
}
```

Justify-Content und Align-Content

Titel:

Ausrichtung des gesamten Grids

Inhalt:

- `justify-content`: Bestimmt die horizontale Ausrichtung des gesamten Grids im Container.
 - Werte: `center`, `start`, `end`, `space-between`, `space-around`.
 - `align-content`: Bestimmt die vertikale Ausrichtung des Grids.
 - Werte: `center`, `start`, `end`, `space-between`, `space-around`.
-

Beispiel

```
.container {  
    display: grid;  
    grid-template-columns: 100px 30% 1fr auto;  
    grid-template-rows: 100px 100px 100px;  
    justify-content: center;  
    align-content: space-around;  
}
```

Wiederholungen mit repeat()

Titel:

Wiederholungen mit der repeat() Funktion

Inhalt:

- Mit repeat() können sich wiederholende Spalten oder Zeilen effizienter definiert werden.



Beispiel

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



Platzierung von Child-Elementen

Titel:

Child-Elemente positionieren

Inhalt:

- grid-column: Definiert, in welchen Spalten sich das Element befindet.
 - Beispiel: grid-column: 1 / 3; – Das Element erstreckt sich über die Spalten 1 und 2.
 - grid-row: Definiert, in welchen Zeilen sich das Element befindet.
 - Beispiel: grid-row: 1 / 2; – Das Element befindet sich in Zeile 1.
-

Beispiel

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



Aufgabe

Schaffen Sie ein Grid der Größe 3x3 und neun div-Boxen, die sich auf die neun Zellen verteilen.




Aufgabe

Geben Sie Ihren neun Grid-Feldern Namen und weisen Sie die neun div-Boxen den benannten Feldern zu (über IDs der div-Boxen).



Aufgabe: "Getting Started with CSS Grid"

 WEB DESIGN PLAYGROUND

Run Code ▶ MENU ☰

Getting Started with CSS Grid

◀ Previous Page

Next Page ▶

CSS Grid

Introduction

For what seems like centuries, web designers have been using unwieldy libraries such as Bootstrap to lay out page elements using one or more rows and one or more columns. This so-called *grid* layout gave designers decent control over where each page element appeared. The cost, though, was high, since grid layout libraries were often complex and were almost always weighed down by too much extraneous CSS or even JavaScript code.

That's about to change as a new CSS technology called CSS Grid comes online. Supported now by the current versions of all the major browsers (see below for the current CSS Grid total market share), CSS Grid is the standards-friendly and no-library-required way to implement a grid layout on your pages. This tutorial introduces you to the basics of CSS Grid.

Here's a roadmap for this tutorial:

- [Creating a Grid Container](#)
- [Defining Grid Rows and Columns](#)
- [Introducing the `fr` Unit](#)
- [Adding Grid Gaps](#)

HTML

1

CSS

1

Hide Editors

New Sandbox



Hausaufgabe

Schaffen Sie ein Grid-Layout für Ihre Website.

Pseudoelemente

Pseudoelemente

Einige Phänomene in HTML-Dateien lassen sich nicht über einen Element- oder Klassenselektor selektieren, z.B.:

- Text der in einer bestimmten Sprache vorliegt
- unbesuchte, aktive oder besuchte Hyperlinks
- das jeweils erste Kindelement eines Elementes.

Link-Pseudoelemente

a:link	unbesuchter Hyperlink
a:visited	besuchter Hyperlink
a:hover	beim Überfahren mit der Maus
a:active	gerade angeklickter Link

Beispiel:

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_links_colors

Aufgabe: Links

Formatieren Sie diese vier Klassen individuell und ohne Unterstreichungen in ihrem Stylesheet.

Weitere Pseudoelemente

:first-line 1. Textzeile eines Elementes

- https://www.w3schools.com/CSS/tryit.asp?filename=trycss_firstline

:first-letter 1. Buchstabe eines Elementes

- https://www.w3schools.com/CSS/tryit.asp?filename=trycss_firstletter

:before Textinhalt/Grafik vor einem
Element einfügen

- https://www.w3schools.com/CSS/tryit.asp?filename=trycss_before

:after Textinhalt/Grafik nach einem
Element einfügen

- https://www.w3schools.com/CSS/tryit.asp?filename=trycss_after

::marker

- https://www.w3schools.com/CSS/tryit.asp?filename=trycss_marker
-

Beispiele

```
p:first-letter {font-size:300%;  
                font-weight:bold;  
                color:red  
            }
```

```
p:before {content:"Text vor dem Element"}
```

```
p:after {content:url(grafik.jpg)}
```