11 CSS Grid Layout

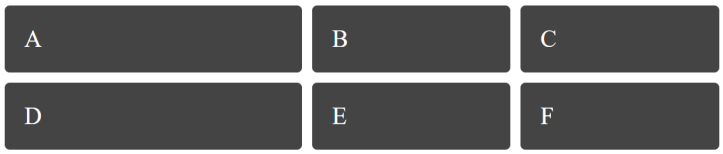
Komplexere Web-Layouts zu erstellen und diese dann auch noch responsive zu gestalten, ist für Webdesigner seit Jahren ein Problem. Während in den Anfangszeiten des Webs mit Tabellenlayout gearbeitet wurde, setzte sich dann das Spaltenlayout mit float durch, das allerdings den Nachteil hat, dass float eigentlich für in den Text eingefügte Bilder, die umflossen werden, gedacht war.
Seit 2009 wurde die flexbox Technik (CSS Flexible Box Layout) verstärkt genutzt, die allerdings nur eindimensional gedacht ist, da sie nur ein eine Richtung (row oder column) Elemente anordnet. Mit CSS-Grid soll dem nun Abhilfe geschafft werden. Echte Gestaltungsraster sollen möglich werden.

11.1 Erstes Beispiel testen und verstehen

→ Erstellen Sie das erste Beispiel und versuchen Sie es zu verstehen.
→ Was machen die folgende CSS-Befehle:

```
grid-template-columns: 300px 200px 200px;
```

```
grid-gap: 10px;
```



→ Fügen Sie anschließend hinzu:

```
grid-template-rows: 300px 200px;
```

→ Was geschieht, wenn der Inhalt größer ist als die Box? Fügen Sie Blindtext hinzu.

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>Grid Verstehen</title>
    <style>
        body {
            margin: 40px;
        }

        .wrapper {
            display: grid;
            grid-template-columns:
300px 200px 200px;
            grid-gap: 10px;
            background-color: #fff;
            color: #444;
        }

        .box {
            background-color: #444;
            color: #fff;
            border-radius: 5px;
            padding: 20px;
            font-size: 150%;
        }
    </style>
</head>

<body>
    <div class="wrapper">
        <!-- Fügen Sie hier eine
Überschrift und einen Absatz ein, was
geschieht?--->
        <div class="box a">A</div>
        <div class="box b">B</div>
        <div class="box c">C</div>
        <div class="box d">D</div>
        <div class="box e">E</div>
        <div class="box f">F</div>
        <!-- hier weitere hinzufügen,
was geschieht? --->
    </div>
```

11.2 Zweites Beispiel (beispiel2.html)

Während das erste Beispiel noch sehr statisch ist, soll das zweite Beispiel flexibler gestaltet werden.
Dazu sollen einerseits die Breiten der Spalten mit einer neuen Einheit `fr = fraction` angesprochen werden und andererseits die Reihenfolge mit Angaben für die untergeordneten Container verändert werden.

→ Betrachten Sie das Beispiel beispiel1.html und versuchen Sie erneut den Quellcode zu verstehen. Alle Klassen ab Klasse .b sind noch auskommentiert.
→ Fügen Sie die auskommentierten Klassen schrittweise hinzu und versuchen Sie zu verstehen.

```
grid-template-columns: 1fr 3fr 1fr;
```

_____

```
.a {
    grid-column-start: 2;
    grid-column-end: 3;
    grid-row-start: 1;
    grid-row-end: 2;
}
```

_____

_____

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

_____

_____

```
.c {
    grid-area: 2 / 3 / 3 / 4;
}

.d {
    grid-area: 1 / 1 / 2 / 2;
}
```

_____

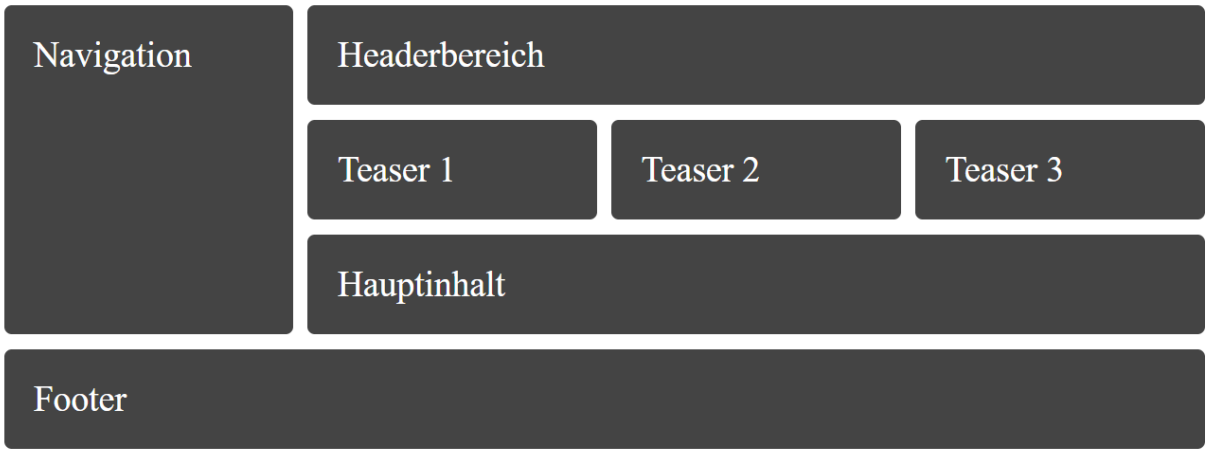
_____

_____

```
.wrapper {
    display: grid;
    grid-template-columns: 1fr 3fr 1fr;
    grid-gap: 10px;
    background-color: #fff;
    color: #444;
}

.box {
    background-color: #444;
    color: #fff;
    border-radius: 5px;
    padding: 20px;
    font-size: 150%;
}

.a {
    grid-column-start: 2;
    grid-column-end: 3;
    grid-row-start: 1;
    grid-row-end: 2;
}

.b {
    grid-column: 2 / 3;
    grid-row: 2 / 3;
}

.c {
    grid-area: 2 / 3 / 3 / 4;
}

.d {
    grid-area: 1 / 1 / 2 / 2;
}

.e {
    order: 1;
}

.f {
    order: 3;
}
```

11.3 Layout mit grid-template-areas

Aussehen:



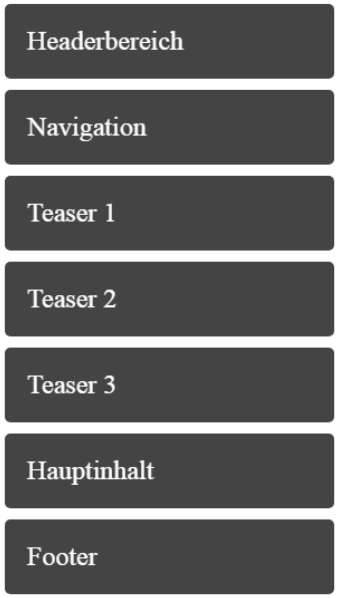
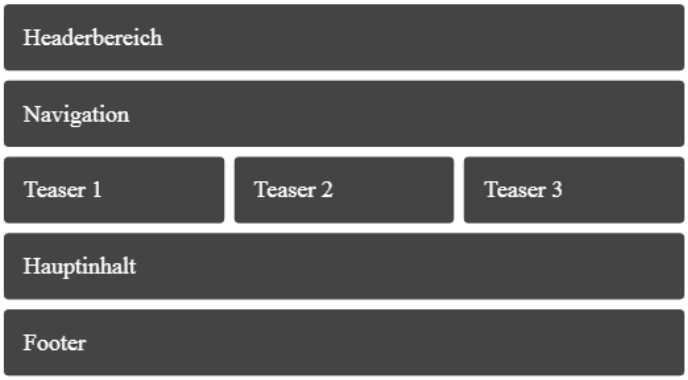
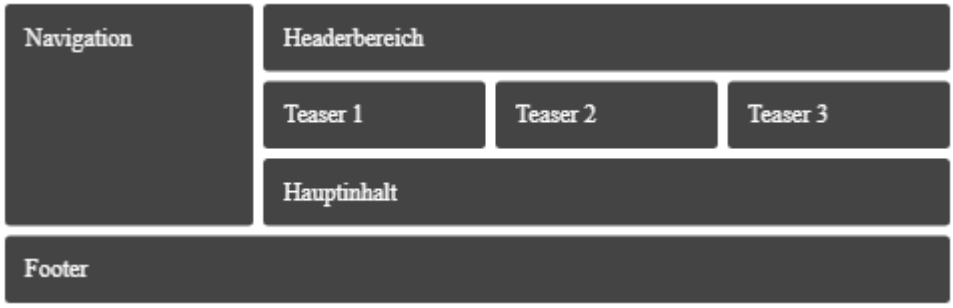
Teile der CSS:

```
        .wrapper {
            display: grid;
            grid-gap: 10px;
            grid-template-areas:
                "nav header header header"
                "nav teaser1 teaser2 teaser3"
                "nav main main main"
                "footer footer footer footer";
            background-color: #fff;
            color: #444;
        }

[…]
        nav {
            grid-area: nav;
        }
```

→ **Ergänzen sie den Rest sinnvoll, so dass obiges Aussehen erreicht wird.**

11.4 Responsive mit grid-tempate-areas

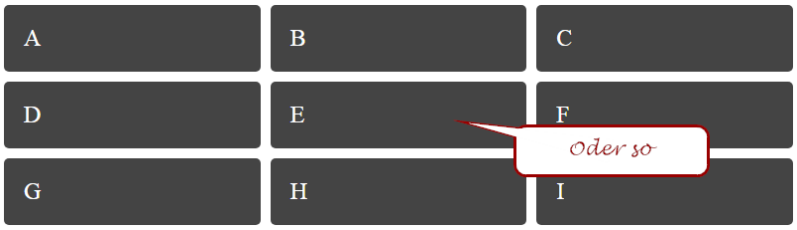
→ Erstellen Sie ausgehend von dem Beispiel oben ein responsives Layout, indem Sie mit media-queries einfach die grid-template-areas anders definieren.
→ Arbeiten Sie mobile first
→ Definieren Sie im head der Datei den viewport mit
<meta name="viewport" content="width=device-width, initial-scale=1">

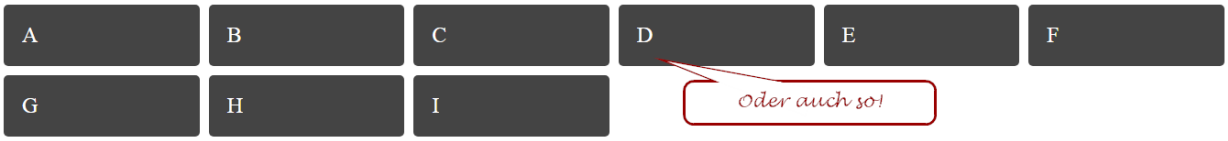
Schmale Ansicht (Smartphone)	
Mittlere Ansicht (ipad)	
Große Ansicht	

11.5 Komplex mit Unterelementen (beispiel4.html)

→ Folgender Quellcode erzeugt flexible Ansichten:


```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>Grid Learning</title>
    <style>
        body {
            margin: 40px;
        }
        .wrapper {
            display: grid;
            grid-gap: 10px;
            grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
            color: #444;
        }
        .box {
            background-color: #444;
            color: #fff;
            border-radius: 5px;
            padding: 20px;
            font-size: 150%;
        }
    </style>
</head>
<body>
    <div class="wrapper">
        <div class="box a">A</div>
        <div class="box b">B</div>
        <div class="box c">C</div>
        <div class="box d">D</div>
        <div class="box e">E</div>
        <div class="box f">F</div>
        <div class="box g">G</div>
        <div class="box h">H</div>
        <div class="box i">I</div>
    </div>
</body>
</html>
```



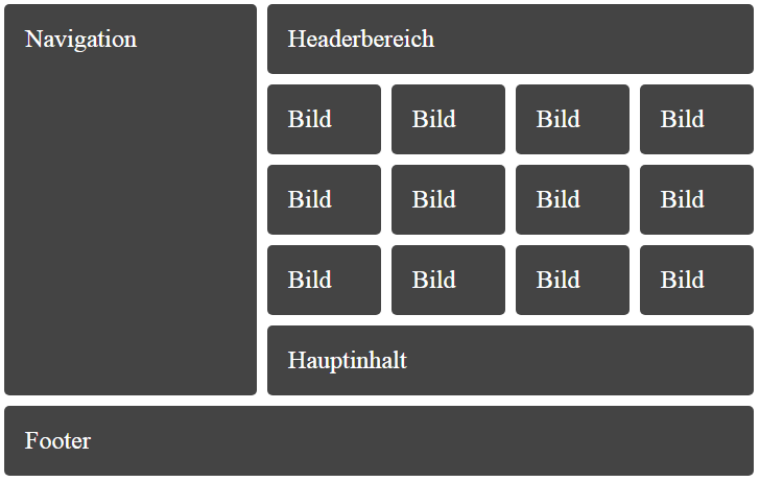


Aufgabe

→ Versuchen Sie die Lösung mit den grid-template-areas und diese so zu kombinieren, dass Sie folgendes Aussehen erhalten:

→ Arbeiten Sie mit einem Breakpoint.

→ Im schmalen Zustand dürfen nur noch die Bilder nebeneinander sein, alles andere untereinander. Lösung mit verschachtelten display: grid



11.6 Weitere Möglichkeit: responsiver Rand mit „. Wrapper .“

Um Bereiche ohne Inhalt zu definieren, kann auch der Punkt als Platzhalter dienen. Möglicher Einsatz ist z. B. die Zentrierung des gesamten Inhaltes und verschachteltes Grid-Layout.

```
body {
           display: grid;
            grid-template-areas: ". wrapper .";
            grid-template-columns: 0 1fr 0;
        }

        @media (min-width:768px) {
            body {
                grid-template-columns: 30px 1fr 30px;
            }
        }

        @media (min-width:1024px) {
            body {
                grid-template-columns: 10% 1fr 10%;
            }
        }

        .wrapper {
            grid-area: wrapper;
```

11.7 CSS Grid in CSS Frameworks

Bootstrap: https://getbootstrap.com/docs/5.1/layout/css-grid/
Tailwind CSS: https://tailwindcss.com/docs/grid-template-columns