

CSS-Übung: Vom Bild zum mehrspaltigen (!) Layout (Teil 2)

In dieser Übung macht ihr aus eurer einspaltigen Seite von letzter Stunde das **fertige Layout - mit Flexbox und Grid**.
Ihr arbeitet mit eurem Projekt aus Teil 1 weiter.

⚠️ Ihr baut auf eurem bestehenden Code auf – ihr fangt **nicht** von vorne an!

Was ihr braucht

Datei	Inhalt
<code>css-full-layout.png</code>	So soll eure Seite heute am Ende aussehen
<code>css-layout-einspaltig.png</code>	Euer Ausgangspunkt von letzter Stunde
Euer Projekt aus Teil 1	HTML + CSS, auf dem ihr heute aufbaut

Legt die beiden Bilder nebeneinander und schaut nochmal genau: **Was muss sich alles ändern?** Welche Elemente stehen jetzt nebeneinander, die vorher untereinander standen?

Flexbox oder Grid?

Bevor ihr loslegt – hier nochmal die wichtigsten Unterscheidungen:

Eine Dimension oder zwei?

- **Flexbox:** Ihr habt eine Reihe von Elementen, die in **einer Richtung** angeordnet werden – Navigation, Button-Reihe, eine Zeile mit Icon + Text
- **Grid:** Ihr wollt **Zeilen und Spalten gleichzeitig** kontrollieren – ein Seiten-Layout, eine Bildergalerie, ein Formular mit ausgerichteten Labels

💡 **Faustregel:** Fragt euch: „*Brauche ich Kontrolle über Zeilen UND Spalten – oder reicht mir eine Richtung?*“
Reicht eine Richtung: Grundsätzlich **Flexbox**.
Braucht ihr beides gleichzeitig: Grundsätzlich **Grid**.



Wer bestimmt die Größe? Eine andere wichtige Unterscheidung: Will ich die Größen der Spalten und Zeilen im Wrapper vorgeben, oder soll das der Content selbst machen?

- **Flexbox:** Die jeweiligen **Kinder** innerhalb der Flexbox geben ihre eigenen Größen vor
- **Grid:** Ihr gebt im **Wrapper** vor, wie groß der Platz ist, den die jeweiligen Kinder einnehmen dürfen

Ablauf

Ihr arbeitet wieder **zu zweit** in euren Teams – wenn möglich – vom letzten Mal.
Redet miteinander – das Besprechen ist auch heute ein wichtiger Teil der Übung!

Phase 1: Planen

Besprecht gemeinsam das **fertige Layout**. Macht wenn nötig Notizen und Skizzen zu den folgenden Fragen:

1. Was ändert sich?

- Welche Bereiche stehen im fertigen Layout **nebeneinander**?
- Welche Bereiche bleiben **untereinander** – und brauchen gar kein Flexbox/Grid?
- Zeichnet das fertige Layout als Kästchen-Skizze und markiert jeden Bereich, in dem sich etwas ändert.

2. Flexbox oder Grid?

Geht jeden markierten Bereich durch und entscheidet:

- Eine Richtung oder zwei? → **Flexbox** oder **Grid**?
- Sollen die Kinder ihre Größe selbst bestimmen, oder gebt ihr sie im Wrapper vor?
- Schreibt eure Entscheidung **und eure Begründung** direkt in die Skizze.

3. Passt eure HTML-Struktur?

Flexbox und Grid wirken immer auf die **direkten Kinder** eines Elements.

- Welches Element wird jeweils zum **Flex- oder Grid-Container**?

- Sind die Elemente, die nebeneinander stehen sollen, auch wirklich **direkte Kinder** dieses Containers?
- Braucht ihr irgendwo einen **zusätzlichen Wrapper** - oder ist einer überflüssig?
- Notiert, wo ihr euer HTML anpassen müsst.

4. Ausrichtung und Abstände

- Wie sind die Elemente **ausgerichtet**? Links, rechts, zentriert, gleichmäßig verteilt? (`justify-content`, `align-items`, ...)
- Wie groß sind die **Abstände zwischen** den Elementen? (`gap` statt `margin`?)
- Sind die Spalten gleich breit oder unterschiedlich? (`1fr`, `2fr`, feste Breite, ...)

✓ **Erst wenn ihr für jeden Bereich eine Entscheidung mit Begründung habt, geht es weiter.**

Phase 2: Umsetzen am Laptop

- Arbeitet **gemeinsam** - am besten an einem Laptop: eine Person tippt, die andere schaut mit, liest aus den Notizen vor und denkt mit. **Wechselt regelmäßig** (z. B. alle 15 Minuten)!
- Sprecht laut aus, was ihr macht und warum.
- Weicht ihr von euren Notizen ab? Kein Problem - aber besprecht, **warum**.

Reihenfolge - geht Bereich für Bereich vor:

1. Zuerst - wenn nötig - das **HTML anpassen** (Wrapper ergänzen oder entfernen).
2. Im Browser prüfen: Sieht die einspaltige Version noch genauso aus wie vorher?
3. Dann `display: flex` bzw. `display: grid` auf den Container setzen.
4. Ausrichtung, Spaltengrößen und `gap` ergänzen.
5. Alte Abstände (`margin`, ...) prüfen: Braucht ihr die noch, oder übernimmt das jetzt `gap`?
6. Immer wieder im Browser mit `css-full-layout.png` vergleichen.
7. Die **DevTools** (F12) helfen euch: Dort könnt ihr Flex- und Grid-Container einblenden lassen (das kleine `flex` / `grid` Badge im Elements-Tab).

Regeln:

- Kein Inline-CSS (`style="..."`) - alles kommt in die CSS-Datei.

- Kein `position: absolute` für das Layout, wo es auch anders gehen würde - dafür gibt es heute Flexbox und Grid!



Erst einen Bereich fertig machen, dann den nächsten!

Wenn ihr überall gleichzeitig anfangt, verliert ihr schnell den Überblick, was welche Änderung bewirkt.



Checkliste zum Abschluss

- ☐ Die Seite sieht aus wie `css-full-layout.png`.
- ☐ Für jeden Bereich könnt ihr erklären, warum ihr Flexbox oder Grid verwendet habt.
- ☐ Bereiche, die untereinander bleiben, kommen ohne Flexbox/Grid aus.
- ☐ Abstände zwischen Flex-/Grid-Kindern werden mit `gap` gelöst.
- ☐ Es gibt keine überflüssigen Wrapper im HTML.
- ☐ Kein `float`, kein `position: absolute` fürs Layout, kein Inline-CSS.



Zum Nachdenken

- Wo musstet ihr euer HTML aus Teil 1 anpassen? Warum?
- Gab es einen Bereich, wo ihr euch zwischen Flexbox und Grid nicht einigen konntet? Hätte beides funktioniert?
- Was passiert, wenn ihr das Browserfenster schmaler macht? Wo bricht das Layout zuerst?
- Stimmt eure Einschätzung von letzter Stunde - wo war die meiste Arbeit?

Speichert euer Projekt gut ab & ladet es jeweils in eure webspaces unter /css-layout hoch!