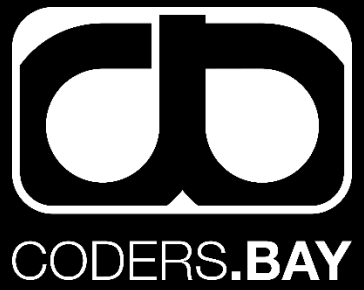


Flexbox

Grundsätzliche Funktionsweis

CODERS.BAY

- **Das sog. »CSS Flexible Box Layout Module« – kurz Flexbox – stellt neben CSS Grid eine der beiden wesentlichen Techniken zur Gestaltung von Layouts mit CSS dar.** Im Gegensatz zu älteren Techniken wie z.B. Floats wurde Flexbox für die Konstruktion von flexiblen und responsiven Layouts erfunden. Es stehen daher sehr mächtige und komfortable Möglichkeiten zur Verfügung.
- Flexbox arbeitet mit einem Container-Element – dem. sog. **Flex-Container** – und den darin enthaltenen Kind-Elementen der 1. Ebene – den sog. **Flex-Items**. Die Flexbox wirkt genau eine Ebene tief. Weitere Verschachtelungen werden ignoriert.

```
<div class="flex-container">
    <div class="flex-item"></div>
    <div class="flex-item"></div>
    <div class="flex-item"></div>
    <div class="flex-item"></div>
</div>
```

- Mit `display:flex;` auf dem Container-Element, wird die Flexbox aktiviert.

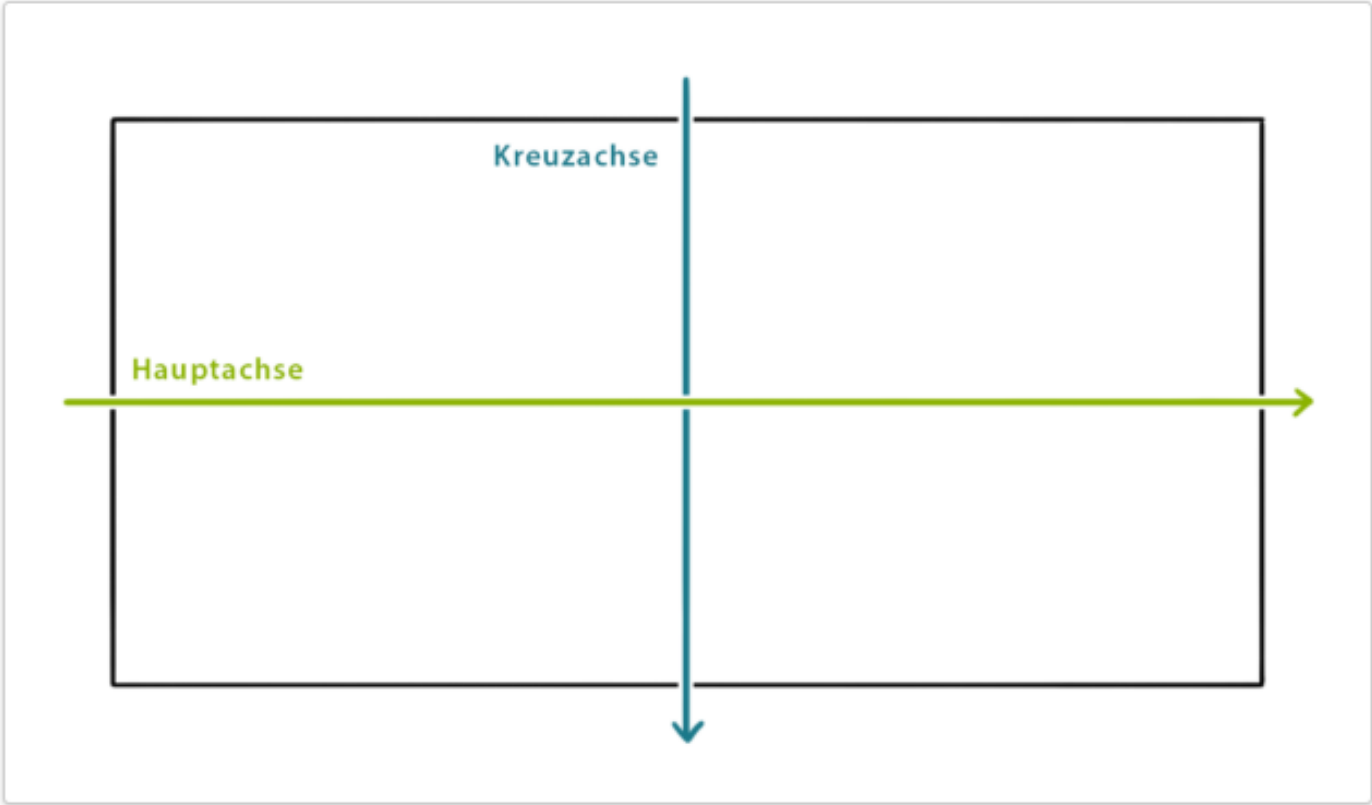
```
.flex-container { display:flex; }
```

Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Haupt-Achse & Kreuz-Achse der Flexbox

CODERS.BAY

- Innerhalb des Flex-Containers existieren zwei Achsen. Die Hauptachse verläuft standardmäßig von links nach rechts, die Kreuz-Achse von oben nach unten



Visualisierung der Flexbox mit Haupt- und Kreuzachse

Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Standard-Eigenschaften des Flex-Containers

CODERS.BAY

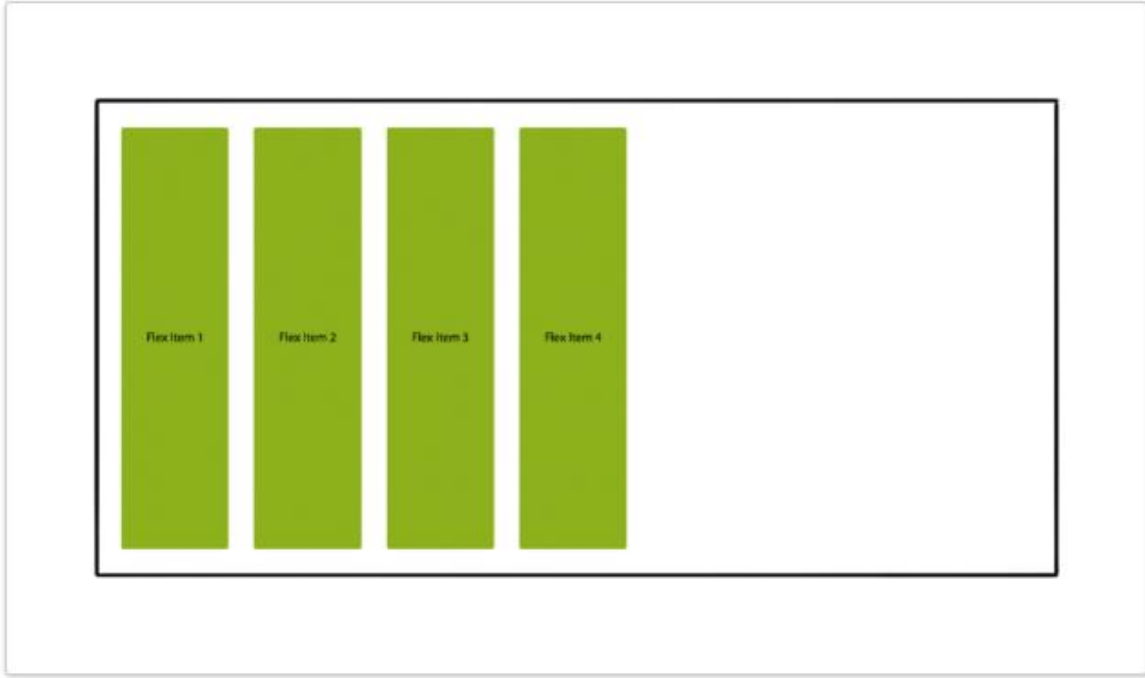
```
.flex-container { display: flex; }
```

Die Items können nun mit Hilfe der Container-Eigenschaften `justify-content` und `align-items` auf den beiden Achsen positioniert werden. Die Ausrichtung der Hauptachse kann mit `flex-direction` verändert werden.

- `justify-content` steuert die Items auf der Haupt-Achse
- `align-items` steuert die Items auf der Kreuz-Achse
- `flex-direction` dreht die Haupt-Achse

Wenn lediglich mit `display:flex;` die Flexbox eingeschaltet wurde, wirken die Standardwerte für `justify-content`, `align-items` und `flex-direction`.

- Der Standardwert für `flex-direction` ist `row` – von links nach rechts
- Der Standardwert für `justify-content` ist `flex-start`: Die Items stehen am Anfang der Hauptachse, was normalerweise einer Positionierung links entspricht.
- Der Standardwert für `align-items` ist `stretch`: Alle Items sind daher standardmäßig so hoch wie der Flex-Container.



Positionierung der Items, wenn die Standardwerte der Flexbox genutzt werden.

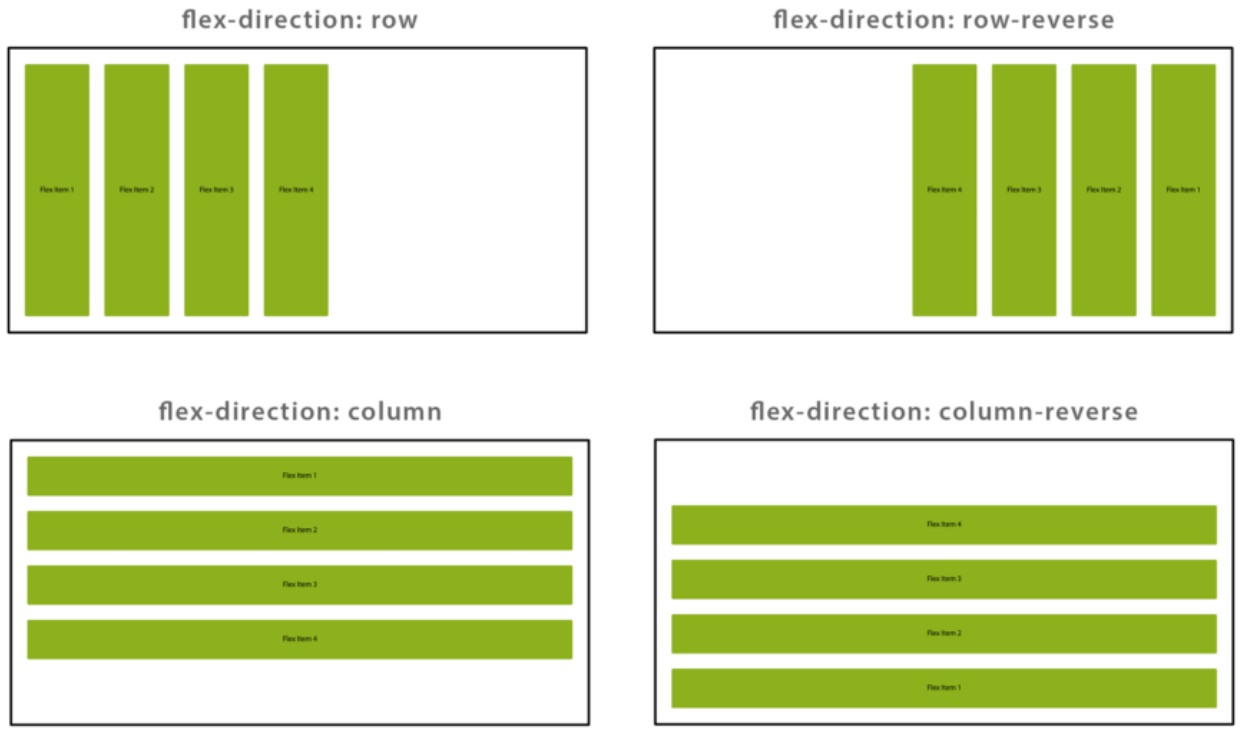
Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

FLEX-DIRECTION – DIE ACHSEN DREHEN

CODERS.BAY

Wie bereits beschrieben, läuft die Hauptachse standardmäßig von links nach rechts und die Kreuzachse von oben nach unten. Mit der Container-Eigenschaft flex-direction können die Achsen gedreht werden.

```
.flex-container {
    display: flex;
    flex-direction: row; /* Standardwert */
}
```



Es stehen folgende Werte zur Verfügung:

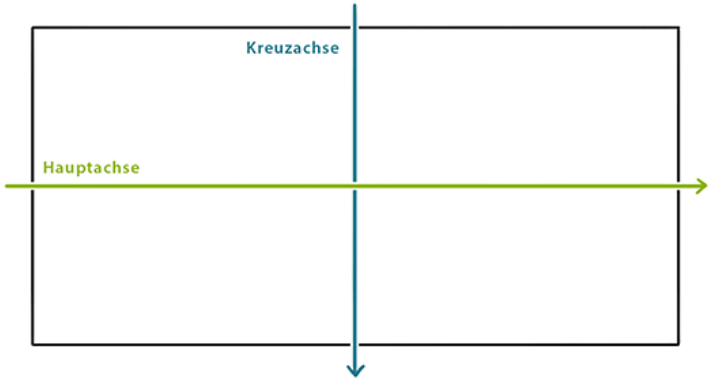
- `flex-direction: row` (von links nach rechts = Standardwert)
- `flex-direction: column` (von oben nach unten)
- `flex-direction: row-reverse` (von rechts nach links)
- `flex-direction: column-reverse` (von unten nach oben)

Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

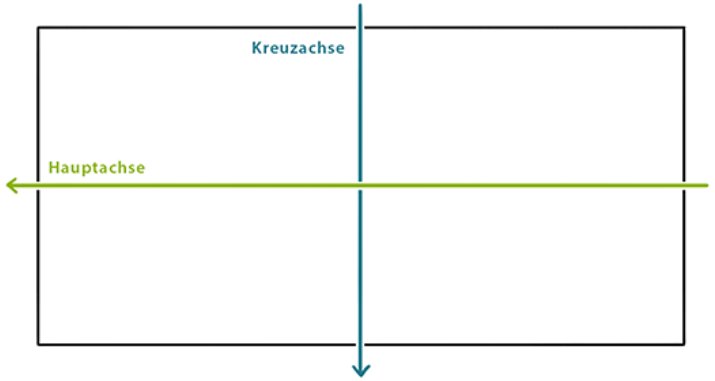
FLEX-DIRECTION – DIE ACHSEN DREHEN

Wichtig ist, dass mit `flex-direction` die beiden Achsen **nicht gemeinsam gedreht werden, sondern nur die Hauptachse**. Wie die Kreuzachse im Verhältnis zur Hauptachse positioniert ist, zeigt die folgende Grafik.

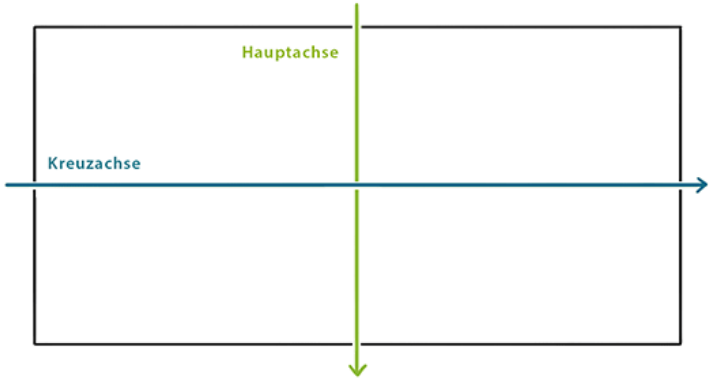
flex-direction: row;



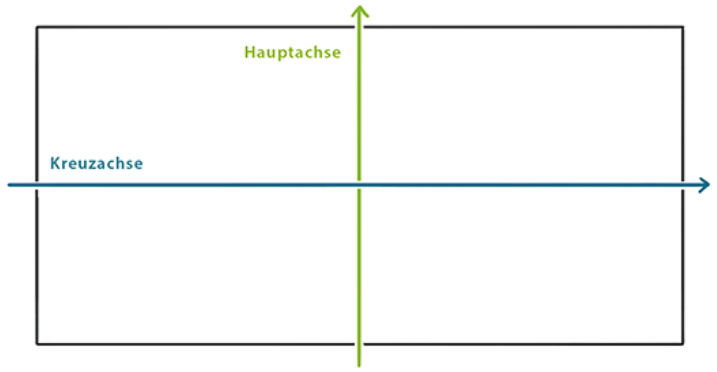
flex-direction: row-reverse;



flex-direction: column;



flex-direction: column-reverse;



Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

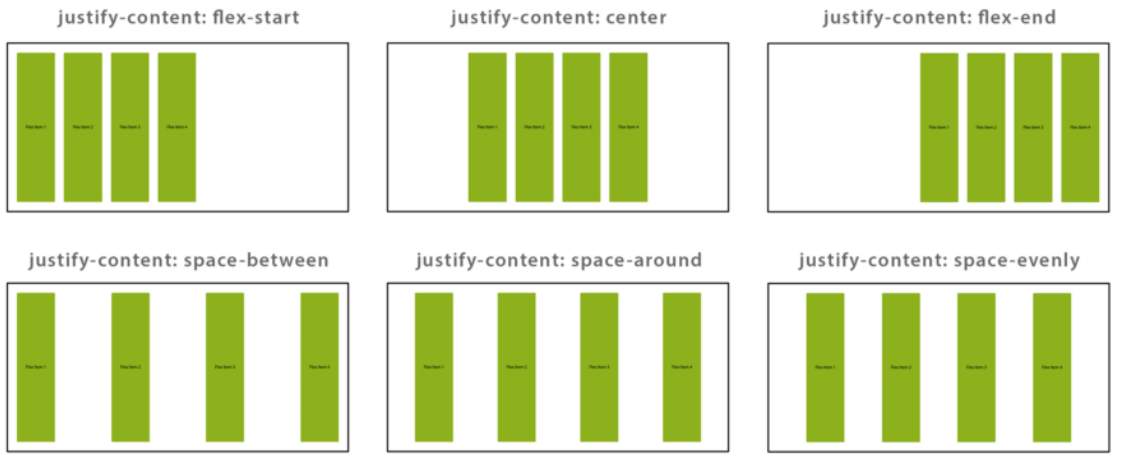
JUSTIFY-CONTENT – ITEMS AUF DER HAUPTACHSE POSITIONIEREN

CODERS.BAY

- Standardmäßig befinden sich die Flex-Items am Anfang der Hauptachse.

Um die Flex-Items auf der Hauptachse zu positionieren, existieren ebenfalls zahlreiche Werte:

- `justify-content: flex-start` (Am Anfang der Hauptachse)
- `justify-content: center` (In der Mitte der Hauptachse)
- `justify-content: flex-end` (Am Ende der Hauptachse)
- `justify-content: space-between` (Das 1. Flex-Item ist am ganz Anfang der Achse, das letzte ganz am Ende. Alle anderen Items werden gleichmäßig verteilt. Es entstehen also Abstände zwischen den Items.)
- `justify-content: space-around` (Die Flex-Items werden wie bei space-between verteilt, allerdings entsteht auch vor dem 1. Item und nach dem letzten Item ein Abstand)
- `justify-content: space-evenly` (Die Flex-Items werden wie bei den beiden vorherigen Methoden verteilt, allerdings werden die Abstände einheitlich gewählt)




```
.flex-container { display: flex; justify-content: flex-start; /* Standardwert */ }
```

Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

ALIGN-ITEMS – ITEMS AUF DER KREUZACHSE POSITIONIEREN

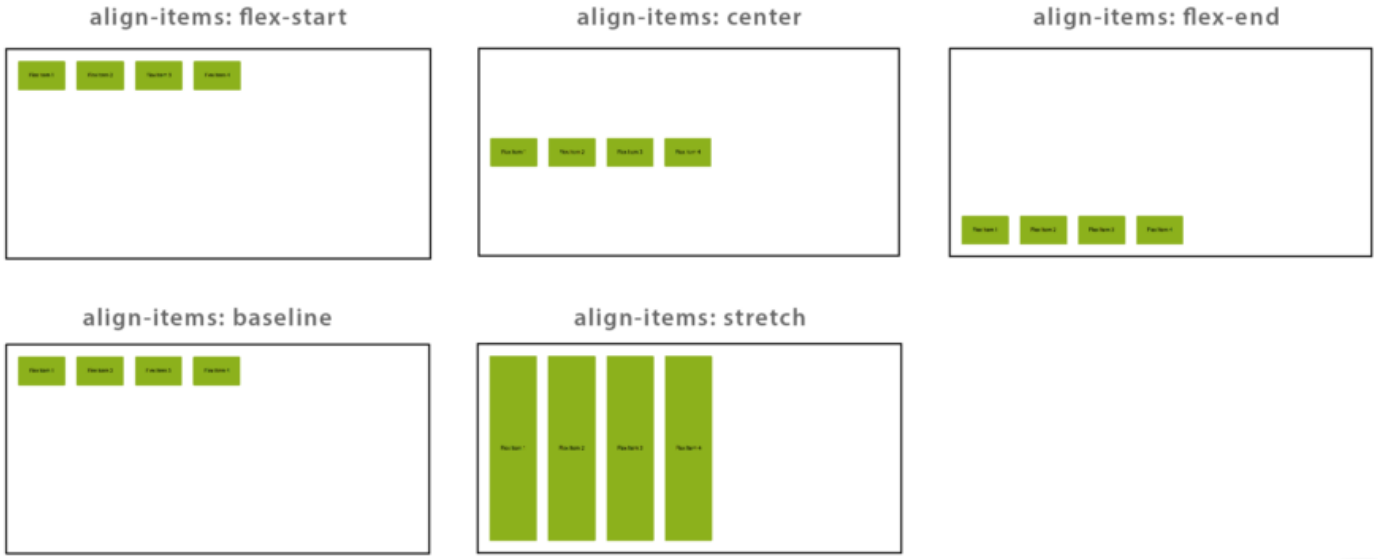
CODERS.BAY

Auf der Kreuzachse werden die Items normalerweise mit dem Wert stretch positioniert: Alle Items sind so hoch wie der Flex-Container

Auch für die Positionierung auf der Kreuzachse wurden in Flexbox einige Möglichkeiten bedacht. Folgende Werte stehen zur Verfügung:

- `align-items: flex-start` (Am Anfang der Kreuzachse)
- `align-items: center` (In der Mitte der Kreuzachse)
- `align-items: flex-end` (Am Ende der Kreuzachse)
- `align-items: baseline` (Die Flex-Items werden so ausgerichtet, dass die Grundlinie der Schrift bündig ist. Nur sichtbar bei unterschiedlichen Schriftgrößen)
- `align-items: stretch` (Alle Flex-Items sind so hoch wie die Flexbox)

```
.flex-container {
    display: flex;
    align-items: stretch; /* Standardwert */
}
```



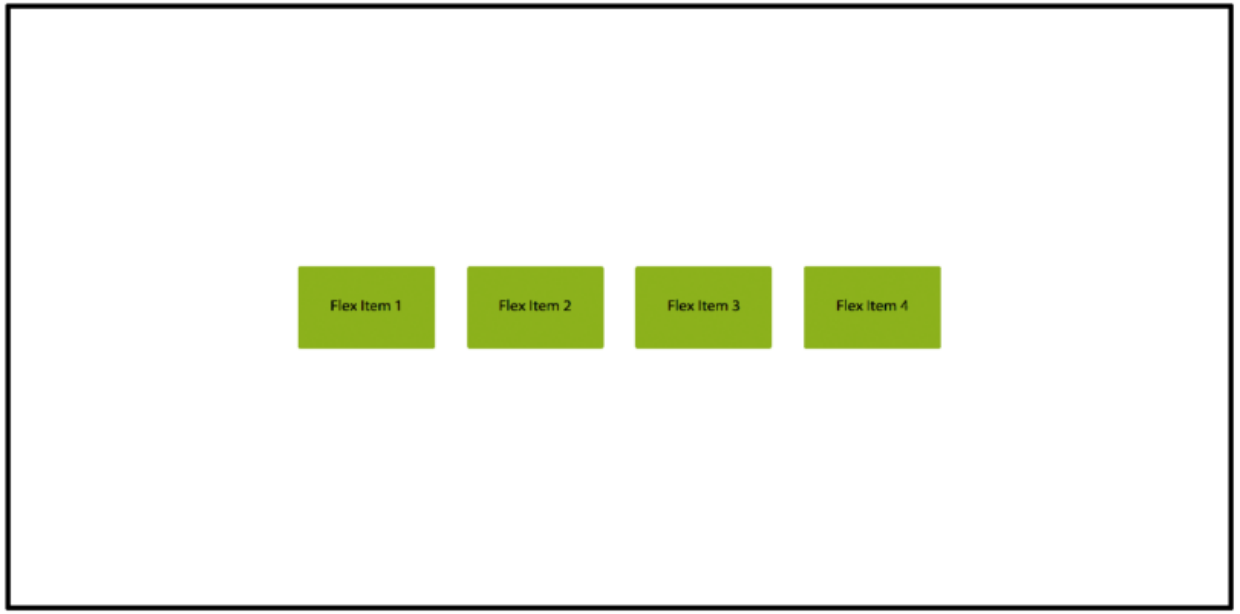
Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Beispiel
Horizontal und vertikal zentrieren

CODERS.BAY

- Wenn ihr mit Hilfe des Flex-Containers Elemente zentrieren möchtet, richtet die Items mittig auf beiden Achsen aus.

```
.flex-container { display: flex; justify-content: center; align-items: center; }
```



Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Standard-Eigenschaften der Flex-Items

CODERS.BAY

Neben der Positionierung über den Flex-Container können auch die Items selbst beeinflusst werden. Dies kann einerseits mit den klassischen CSS-Angaben für Größe und Box-Modell erledigt werden. Darüber hinaus existieren in Flexbox die Eigenschaften `flex-basis`, `flex-grow` und `flex-shrink` zur Beeinflussung der Größe eines Items sowie `order` zur Veränderung der Reihenfolge von Items.
Wenn keine speziellen Angaben für ein Flex-Item gemacht werden, besitzen die Items folgende Standardwerte:

- `flex-grow: 0` (Items dürfen nicht größer werden als festgelegt)
- `flex-shrink: 1` (Items dürfen kleiner werden als festgelegt)
- `flex-basis: auto` (Die Basis-Größe der Items wird automatisch festgelegt – also anhand der Angaben `height`, `width` bzw. auf Grundlage des enthaltenen Inhalts)
- `order: 0` (Die Reihenfolge entspricht der des HTML-Codes)

Hinweis: `flex-grow`, `flex-shrink` und `flex-basis` werden in der Kurzschreibweise `flex` zusammengefasst.

```css
/* Kurzschreibweise */
.flex-item {
        flex: 0 1 auto;
}

/* Lange Schreibweise */
.flex-item {
        flex-grow: 0;
        flex-shrink: 1;
        flex-basis: auto
}
```

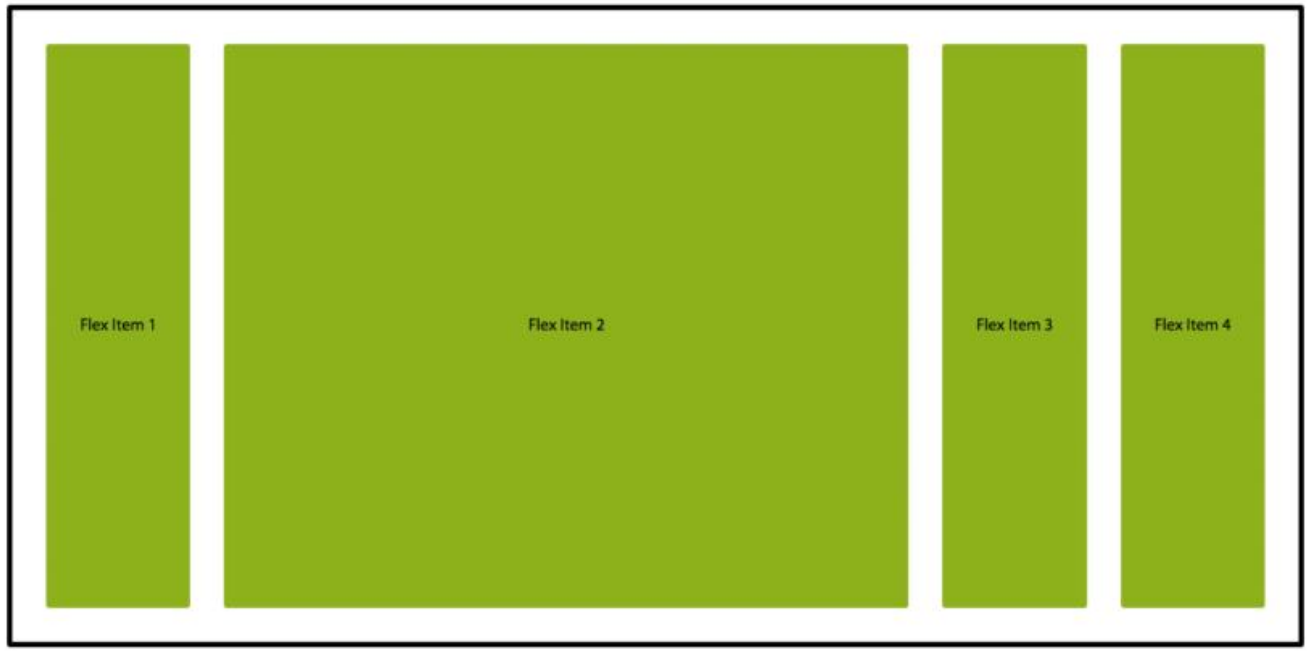
Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

FLEX-GROW, FLEX-SHRINK UND FLEX-BASIS

Mit den verschiedenen `flex-Werten` kann das gewünschte Layout innerhalb des Containers hergestellt werden. Die folgenden Beispiele mit jeweils vier Items zeigen was möglich ist.

Beispiel 1
Mit `flex-grow: 1;` erlauben wir dem 2. Item sich auszudehnen.

```
flex-item-2 { flex-grow: 1; }
```



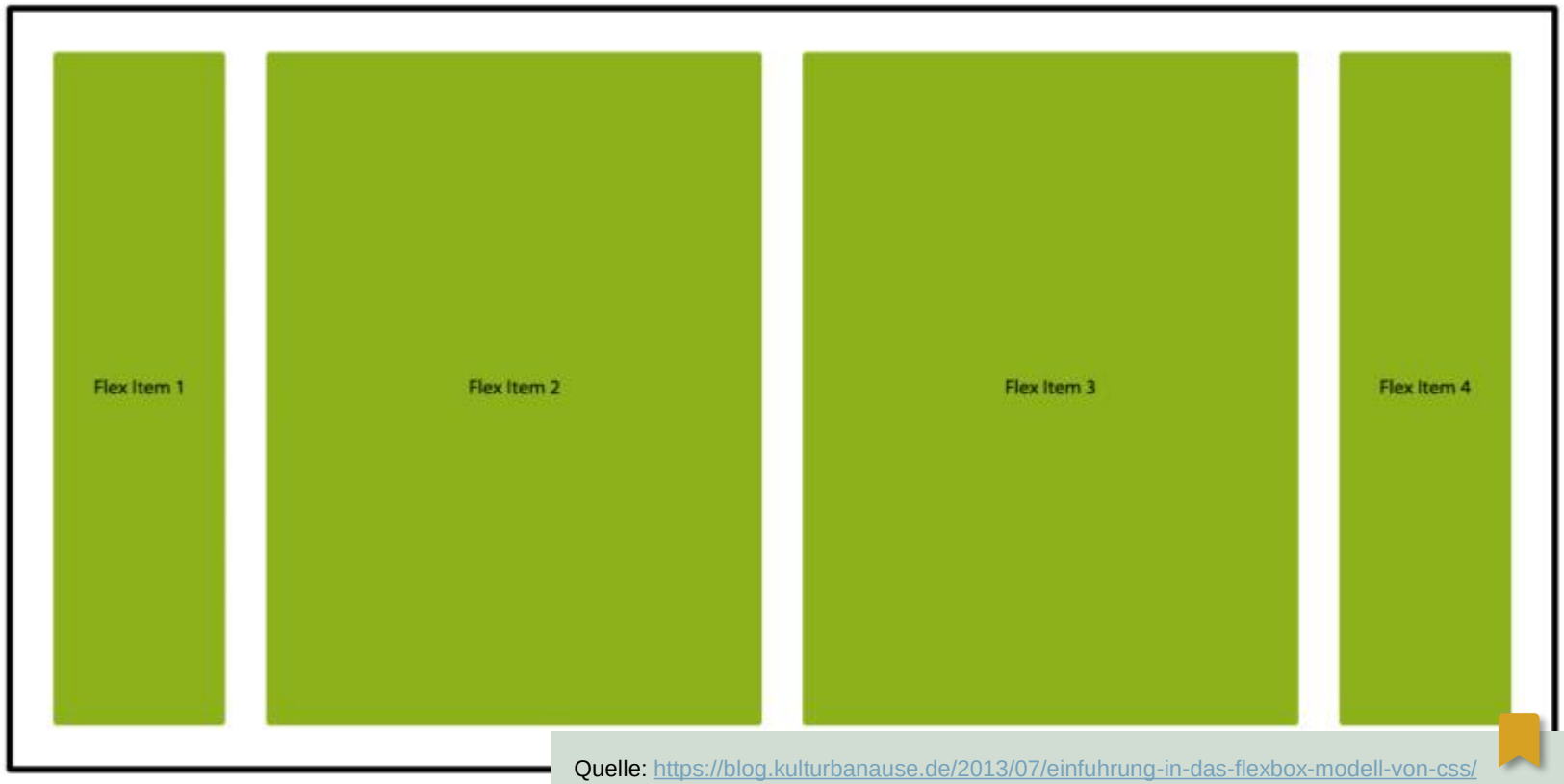
Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Beispiel 2

CODERS.BAY

Wenn mehrere Items `flex-grow: 1;` erhalten, teilen sie sich den verfügbaren Platz in gleichen Teilen auf.

```
.flex-item-2, .flex-item-3, { flex-grow: 1; }
```



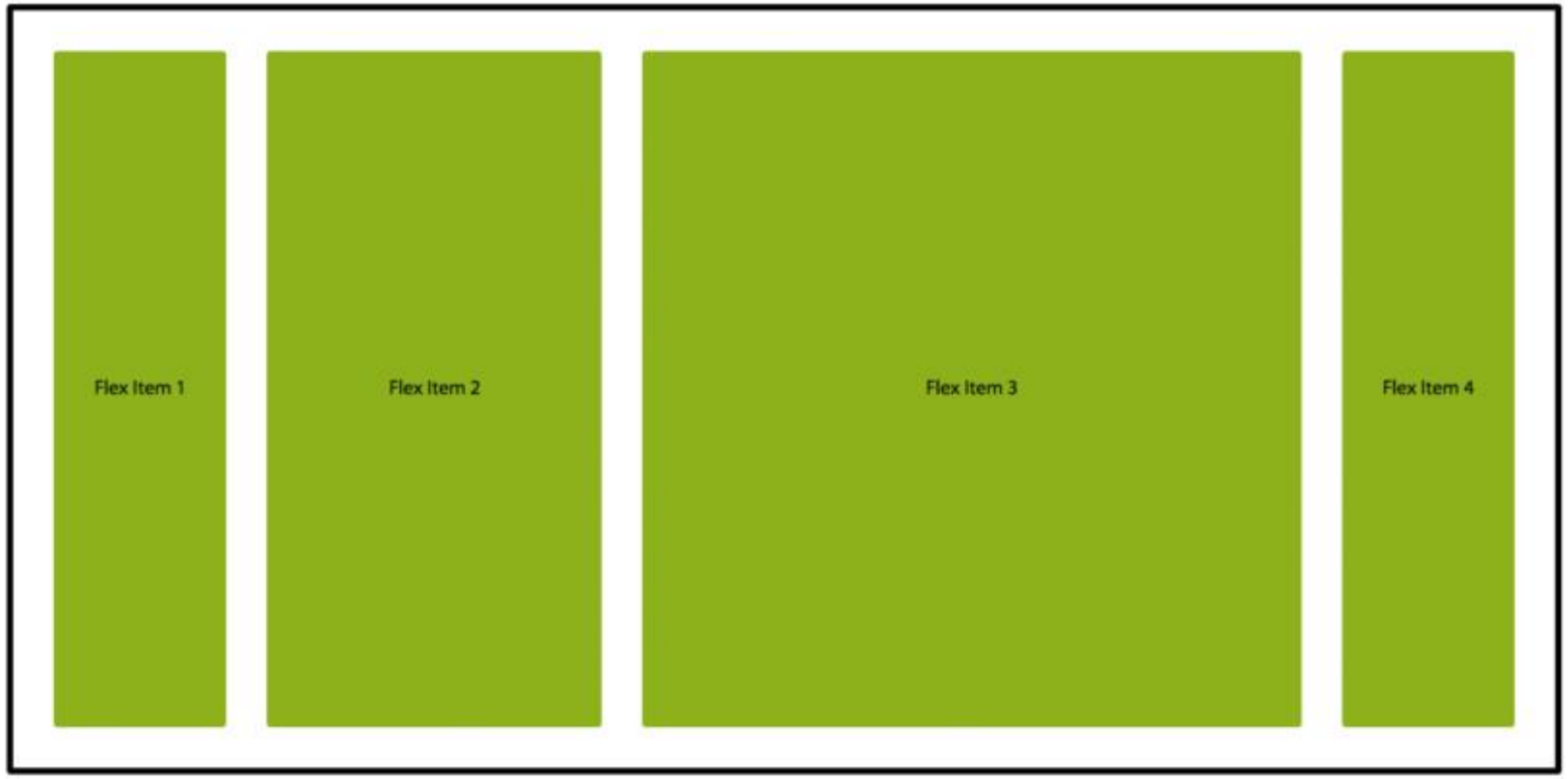
Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Beispiel 3

CODERS.BAY

Wenn Items unterschiedliche `flex-grow`-Werte erhalten, teilen Sie sich den verfügbaren Platz in ungleichen Teilen auf.

```
.flex-item-2 {
    flex-grow: 1;
}
.flex-item-3 {
    flex-grow: 3;
}
```

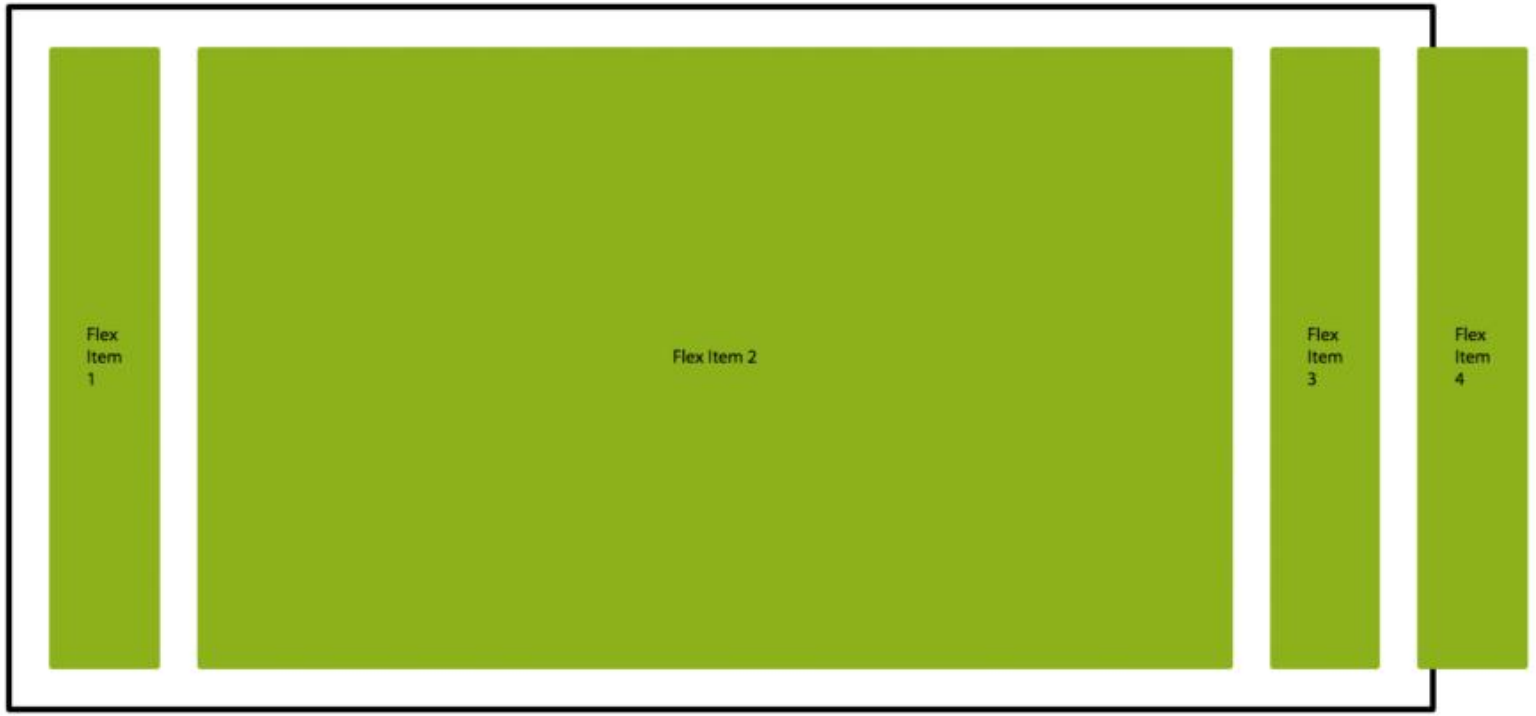


Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Beispiel 4

- Item 2 hat nun eine Ausgangsbreite von 75% erhalten und darf weder kleiner noch größer werden. Alle anderen Items nutzen die Standardwerte – dürfen also schrumpfen. Da der verbliebene Platz für diese drei übrigen Items sehr gering ist, werden sie zusammengedrückt, was erkennbar daran ist, dass die Texte umbrechen. Dennoch passen nicht alle Items gemeinsam in den Container und laufen daher nach rechts heraus.

```
.flex-item-2 {
    flex-grow: 0;
    flex-shrink: 0;
    flex-basis: 75%;
}
```



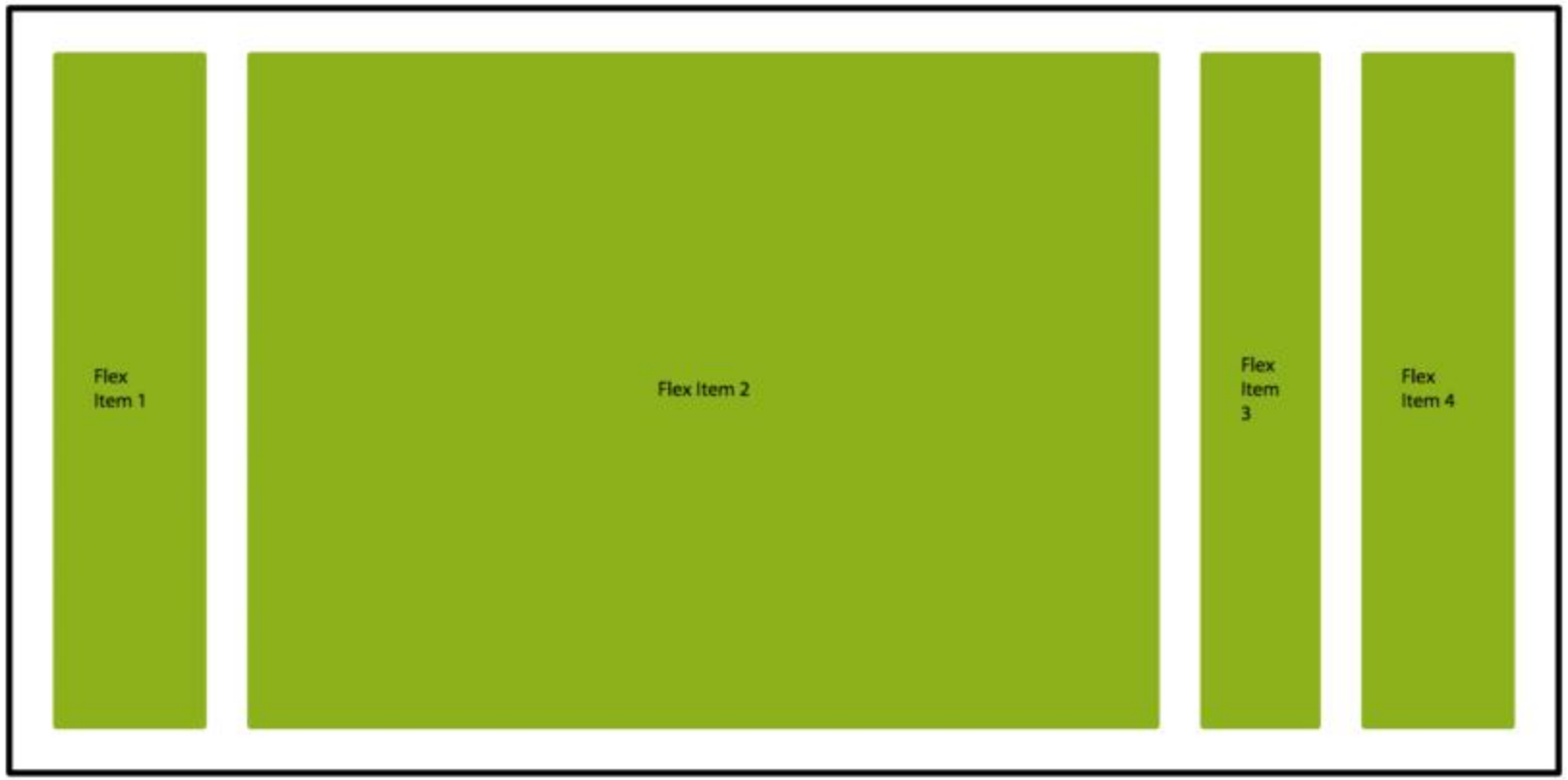
Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Beispiel 4

CODERS.BAY

- Um eine solche Fehldarstellung zu vermeiden, würde man im Normalfall dem Item 2 erlauben zu schrumpfen. Flexbox versucht dann grundsätzlich für das Item 2 eine Breite von 75% zu erreichen – weicht aber nach unten ab, wenn ansonsten die Items aus dem Container heraus laufen würden.

```
.flex-item-2 {
    flex-shrink: 1;
    flex-basis: 75%;
}
```



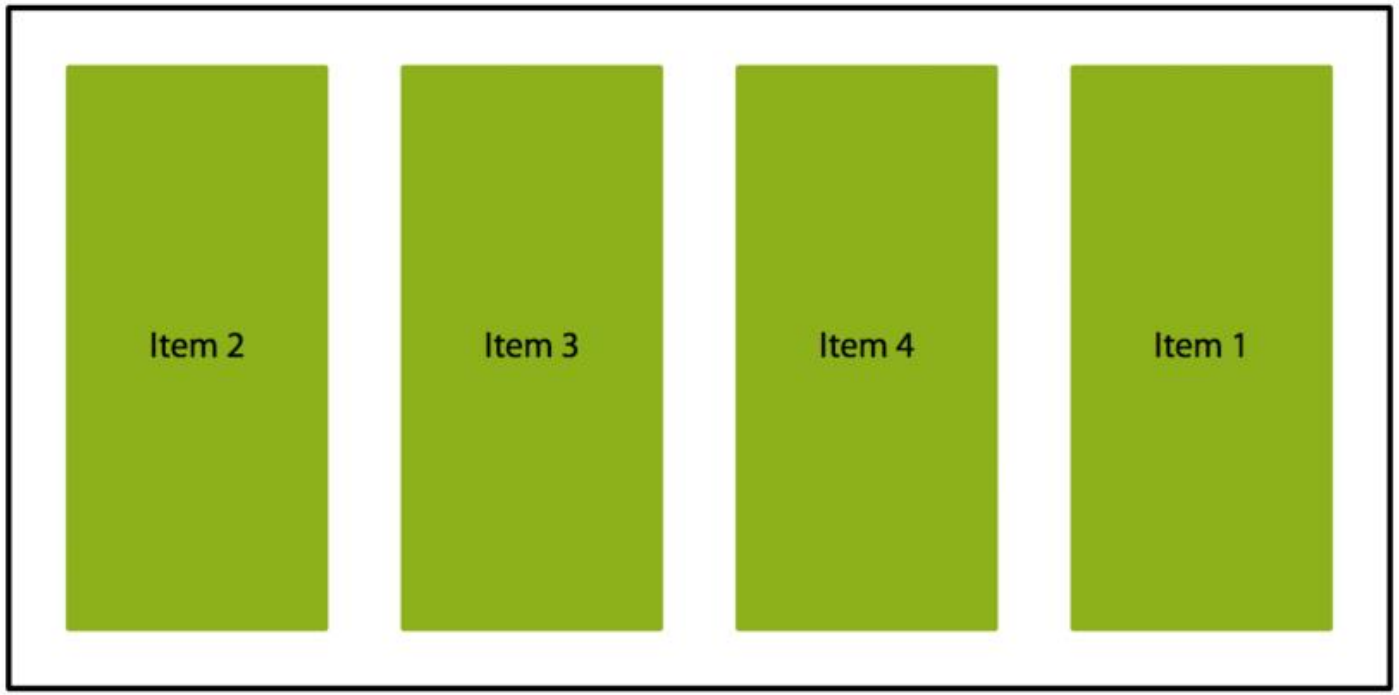
Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

ORDER – DIE REIHENFOLGE VON ITEMS BEEINFLUSSEN

CODERS.BAY

Normalerweise werden die Items in der Reihenfolge angezeigt, in der sie im HTML-Code notiert wurden. Mit der Eigenschaft `order` kann diese Reihenfolge geändert werden, ohne dass der HTML-Code angepasst werden muss.

```
.flex-item-1 {
order: 4;
}
.flex-item-2 {
order: 1;
}
.flex-item-3 {
order: 2;
}
.flex-item-4 {
order: 3;
}
```



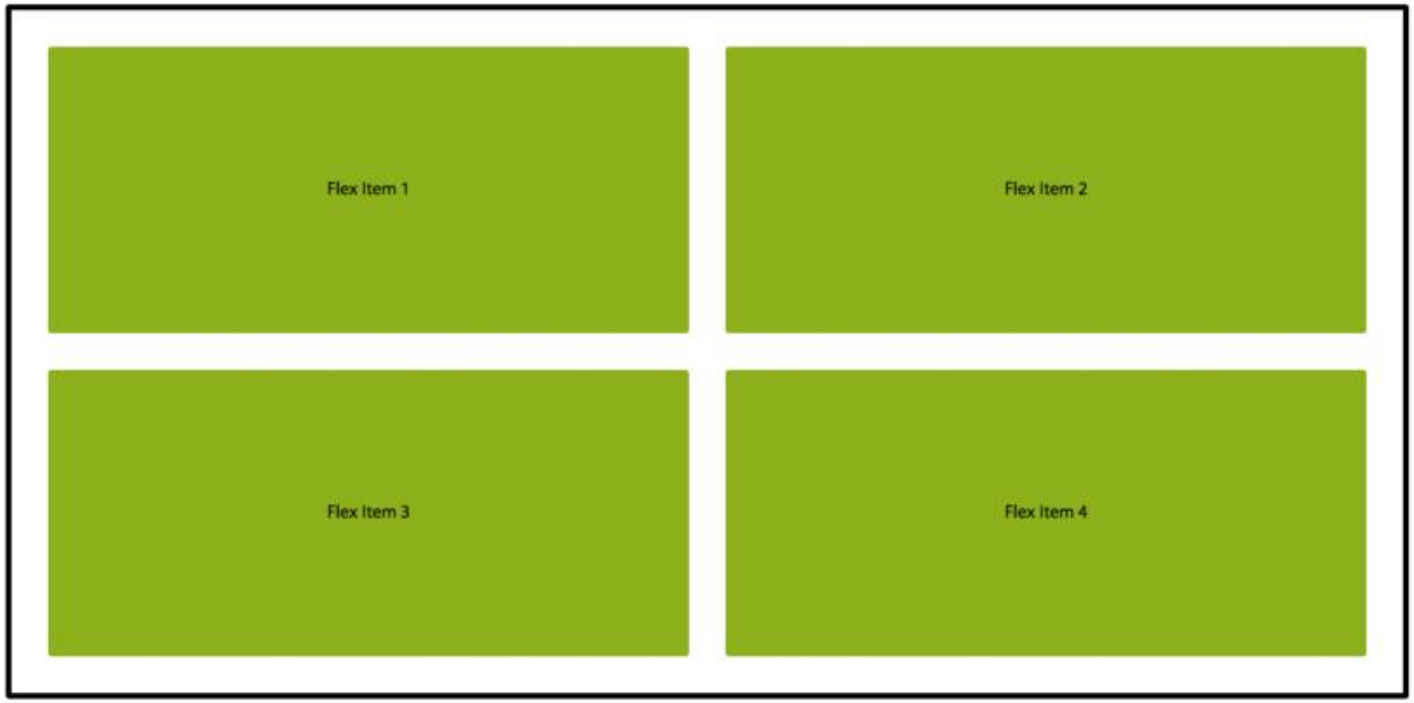
Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

FLEX-WRAP

CODERS.BAY

Mehrspaltigkeit über das Container-Element ermöglichen
Normalerweise werden die Items innerhalb der Flexbox linear nebeneinander oder untereinander auf den Achsen positioniert. Wenn mehrere Items von der Abmessung her nicht in den Container passen, kann mit flex-grow und -shrink festgelegt werden was passieren soll. Mit der **Container-Eigenschaft** `flex-wrap` kann festgelegt werden, dass Items in mehrere Spalten und Zeilen umbrechen.

```
.flex-container {
    display: flex;
    flex-wrap: wrap;
    /* Standardwert = nowrap */
}
.flex-item {
    flex-basis: 50%;
    flex-grow: 1;
    flex-shrink: 1;
}
```



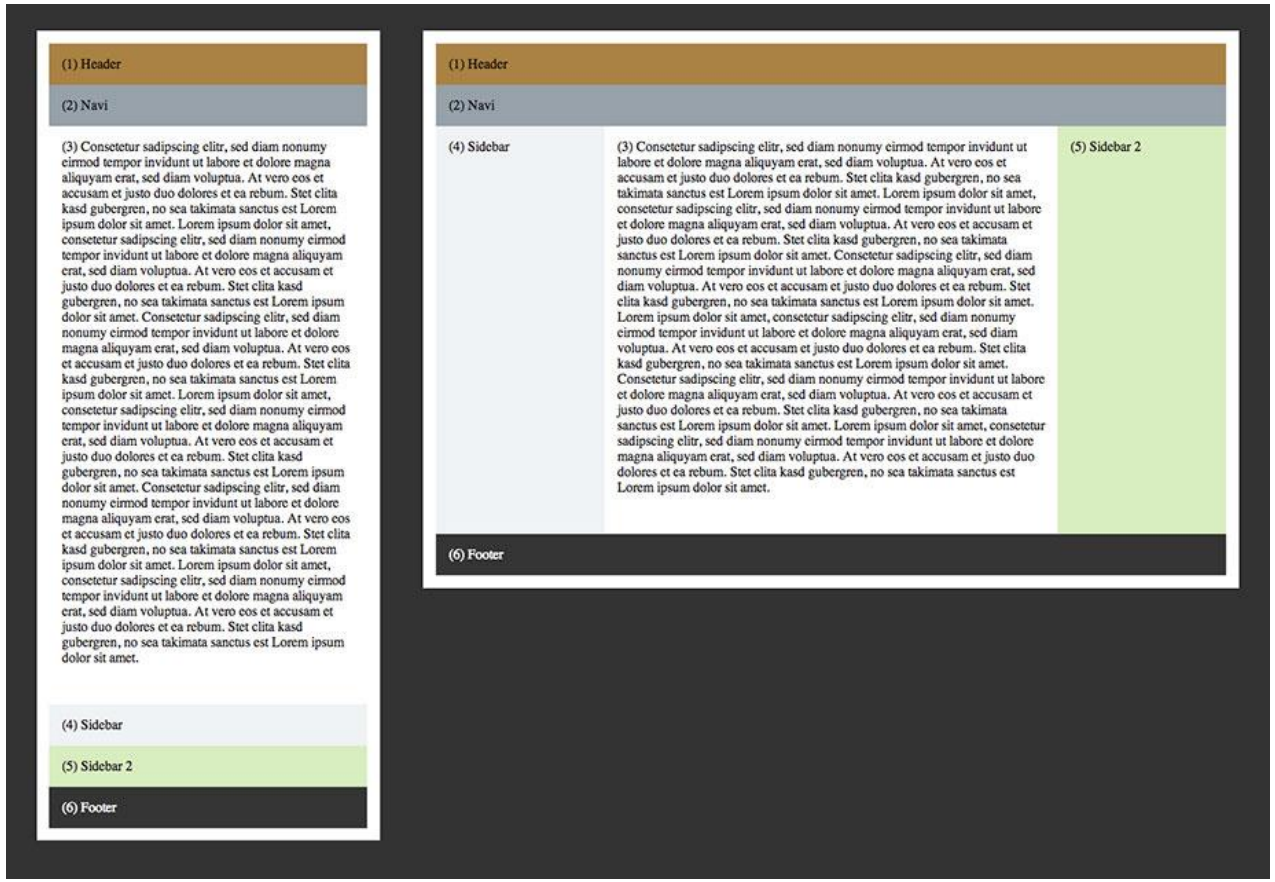
Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

Responsive Design mit Flexbox

CODERS.BAY

- Mit den nun kennengelernten Eigenschaften und einem Media Query lässt sich unkompliziert kleines Praxisbeispiel erstellen. Die folgende Website ist so aufgebaut, dass in der Smartphone-Ansicht (und im HTML-Quellcode) die beiden Sidebars nach dem Hauptinhalt angezeigt werden. In der Desktop-Ansicht wird mit Hilfe von Flexbox der Hauptinhalt zwischen den Sidebars dargestellt. Wir erreichen so gleichzeitig die perfekte Darstellung und den semantisch sinnvollsten HTML-Code.

```
/* Nur der für die mit Flexbox gebaute Desktop-
Darstellung notwendige Code wird gezeigt. */
@media screen and (min-width: 800px) {
    .content {
        display: flex;
    }
    .main {
        flex: 3;
        order: 2;
    }
    .sidebar {
        flex: 1;
        order: 1;
    }
    .sidebar2 {
        flex: 1;
        order: 3;
    }
}
```



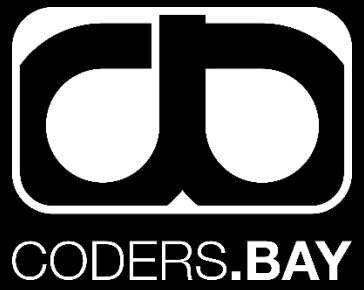
Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/

CSS Flexbox Beispiele

CODERS.BAY

- Wir haben im Laufe der Zeit einige Beispiele für Layouts oder Komponenten auf Grundlage von Flexbox erstellt. Zur Vertiefung der oben beschriebenen Grundlagen sind die folgenden Beispiele sicher hilfreich.
  - Text mit CSS vertikal zentrieren
  - Mehrspaltige Liste von Boxen mit vertikal zentrierten Inhalten
  - Responsive Sticky Footer
  - Prozess- / Fortschritts-Navigation mit CSS
- **Browser-Support**
  - Der Browser-Support für Flexbox ist sehr gut. Eine Übersicht über den stets aktuellen Browser-Support findet ihr auf Can I Use.
- **Tools und Hilfsmittel**
  - Auf Best Web Design Tools findet ihr hilfreiche Tools zum Thema Flexbox.

Quelle: https://blog.kulturbanause.de/2013/07/einfuhrung-in-das-flexbox-modell-von-css/



Flexbox Ende