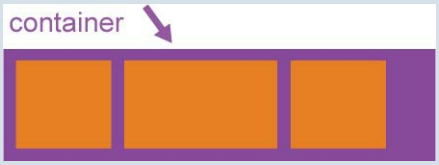
Hintergrund

Um einen mit Flex Boxen zu arbeiten, muss man einen Flex-Container definieren, welcher, als Eltern-Element, die Flex Elemente umschließt. Ob sich der Flex Container wie ein inline oder ein block-Element verhält, bestimmt der gegebene Wert.
Hat man einen Flex Container definiert, so bestimmt dieser die flex Eigenschaften der beinhalteten Kind-Elemente.
Das Flexbox Layout ist ein Modul zielt darauf ab, einen effizienteren Weg anzubieten, um den Platz zu verteilen, aufzuteilen und auszurichten, welchen Elemente einnehmen. Selbst wenn deren Größe nicht bekannt ist und/oder dynamisch. Daher auch der Begriff "Flex".
Die Hauptidee hinter dem Flex Layout ist es, einem Container die Fähigkeit zu verleihen, die Höhe/Breite der Elemente seines Inhaltes zu verändern und anzuordnen, um den verfügbaren Platz bestmöglich auszunutzen. Primär um sich bestmöglich an alle Arten von Displays und Auflösungen anzupassen.

Das Flexbox Layout ist ideal geeignet für die Komponenten einer Anwendung und Layout mit einem kleinen Maßstab. Das Raster Layout (Grid) hingegen ist für größere Maßstäbe gedacht.

Eltern Container



display

Definiert einen Flex Container
Code:

```
.container {
display: flex / oder inline -flex / ;
}
```

CSS-Columns haben keinen Einfluss auf einen Flex-Container

flex-direction

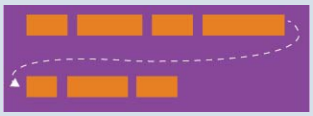
flex-direction

definiert die Hauptachse, an der die Flex-Elemente im Flex-Container a
werden. Flexbox ist ein Layout Konzept, gedacht ist um nur in eine Rich
gerichtet ist. Flex Elemente werden entweder in horizontalen oder vertik
ausgerichtet.
Code:

```
.container {
flex-d ire ction: row | row-re verse | column | colu
se ;
}
```

row(default): ordnet von links nach rechts an
row-revers: ordnet von rechts nach links an
column: ordnet von oben nach unten an
column-reverse: ordnet die Elemente von unten nach oben an.

flex-wrap



flex-wrap

Von Haus aus versuchen alle Flex-Elemente sich in einer Zeile anzuordnen. Ändern kann man dies mit dem `flex-wrap` Attribut.
Die Richtung spielt auch eine Rolle, in dem sie bestimmt, in welcher Richtung die neue Zeile aufgefüllt wird.
Code:

```
.container {
flex-wrap: nowrap | wrap | wrap-r everse ;
}
```

nowrap (Standard): Anordnung in einer Zeile von lnks nach rechts
wrap: Anordnung in mehreren Zeilen von links nach rechts
wrap-reverse: Anordnung in mehreren Zeilen von rechts nach links

align-items

Definiert das Standardverhalten wie Kind Elemente entlang der Haupta
werden.
Code:

```
.container {
align- items: flex-start | flex-end | center | stret
}
```

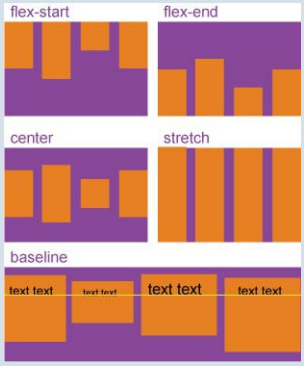
flex-start: Alle Elemente werden am oberen Rand des Eltern Containers
flex-end: Alle Elemente werden am unteren Rand des Eltern Containers

align-items (cont)

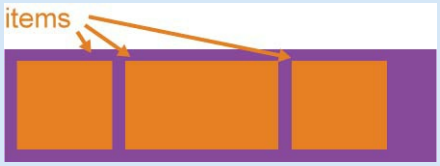
center: Alle Elemente werden mittig im Eltern Container angeordnet.
stretch (Standard): Streckt alle Kind Elemente bis an den oberen und unteren Rand des Eltern Containers.
baseline: Alle Elemente werden entlang der Basislinie ihres Inhaltes ausgerichtet.

Man kann `align- items` als vertikale Version von `justif y-c o ntent` betrachten.

align-items



Kind Elemente

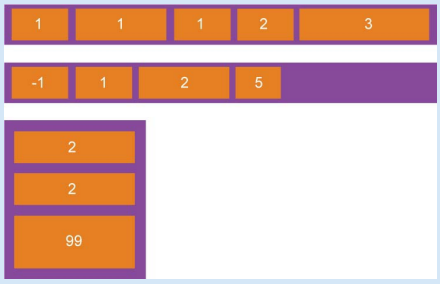


order

Grundsätzlich werden flex-Elemente in der Reihenfolge, wie sie im Quelltext stehen angezeigt.
Jedoch lässt sich, mittels `order`, eine eigene Reihenfolge vorgeben, in welcher die Elemente angezeigt werden.
Code:

```
.item {
order: <in teg er> <in teg er> ...;
}
```

order



align-self

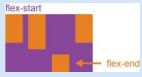
Erlaubt es, die Standardausrichtung oder der Ausrichtung, welche mitte wurde, für ein einzelnes Kind Element zu überschreiben.
Code:

```
.item {
align- self: auto | flex-start | flex-end | center |
}
```

Es gelten die gleichen Erklärungen der Attributwerte, wie sie für `align`

`float`, `clear` und `vertic al- align` haben keinerlei Auswirkungen auf ein Flex Element.

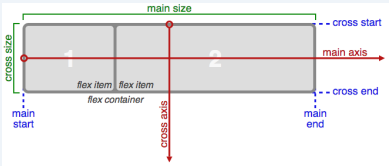
align-self



Grundlagen und Terminologie

Da Flexbox ein komplettes Modul ist und nicht nur ein einzelnes Attribut, beinhaltet es selbst eine ganze Anzahl an Attributen. Einige davon werden auf den umschliessenden Eltern Container angewendet, andere auf dessen Kind-Elemente.
Während ein normales Layout sowohl auf block als auch auf inline Fließrichtungen aufgebaut ist, basiert das Flexbox Layout auf dem *flex-flow*-Fluss.

Flexbox Model



Legende

Attribute welche auf die Eltern Container angewendet werden, haben ein dunkleres Blau und Kind Elemente sind mit einem helleren Blau als dem, welches für Informationen verwendet wurde.

flex-flow

Dies ist eine Kurzform für die Attribute `flex-d ire ction` und `flex-wrap, die zusammen das Verhalten für Quer und Längsachse eines Flex Containers definieren.

Code:

```
.container {
flex-flow: <fl ex- dir ect ion> || <fl ex- wra p> ;
}
```

Standardverhalten ist `row nowrap`

justify-content

Definiert die Anordnung entlang der hori- zontalen Achse. `justif y-c ontent` hilft den übrigen freien Raum zu verteilen, wenn alleElemente in der Zeile unflexibel sind, oder wenn sie zwar flexibel sind, aber ihre maximale Größe bereits erreicht haben. Ermöglicht weiter die Anordnung von Elementen, wenn sie die Zeilenbreite überschreiten würden.

Code:

```
.container {
justif y-c ontent: flex-start | flex-end | center |space -be tween | space- arou
nd ;
}
```

flex-start (Standard): Die Elemente werden zum Anfang hin, nach links, verschoben.

flex-end: Die Elemente werden zum Ende hin, nach rechts, verschoben.

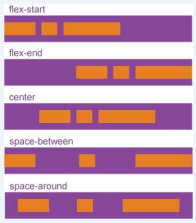
center: Die Elemente werden in der Mitte angeordnet.

space-between: Die Elemente werden gleichfürmig über die Zeile verteilt. Erstes Element ganz links, das Letzte ganz rechts.

space-around: Die Elemente werden gleichmäßig über die Zeile veteilt, haben aber an allen seiten d selben Abstand, sowohl zum Zeilenanfang/ende, als auch zum nächsten Element.

Bei `space- around` muss beachtet werden, da jedes Element immer den selben Abstand zum nächstgen Element hat, erzeugt das eine optisch ungleiche Verteilung. Das liegt daran, dass beim ersten Element links nur ein Abstand ist, aber zwischen den Elementen immer sich zwei Abstände addieren.

justify-content



align-content

Ordnet die Zeilen der Kind Elemente an, sollte es noch Platz geben in c `nt` individuelle Elemente an der Hauptachse ausrichtet.

Code:

```
.container {
align- con tent: flex-start | flex-end | center | st
d ;
}
```

flex-start: Richtet alle Zeilen nach oben links am Eltern Container aus.

flex-end: Richtet alle Zeilen nach unten links am Eltern Container aus.

center: Richtet alle Elemente mittig im Eltern Container aus.

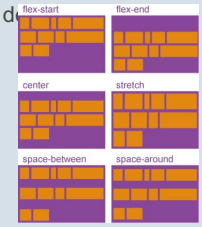
stretch (Standard): Richtet alle Elemente mittig im Eltern Container aus Elemente.

space-between: Richtet alle Zeilen über die volle Höhe des Eltern Cont an den Eltern Container angelegt.

space-around: Verteilt die Platz zwischen den Zeilen gleichmäßig. Jedo cht" zu beachten, wie es auch bei `justif y-c ontent` entsteht.

Hat keine Auswirkungen wenn es nur eine Zeile an Kind Elementen gibt.

align-content



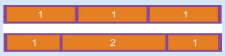
flex-grow

Definiert die Fähigkeit eines Elementes seine Größe zu verändern, sollte es nötig sein. `flex-grow` akzeptiert einheitenlose Werte als Proportion. Dies definiert wie viel des maximalen Platzes ein Elementes, innerhalb eines Containers, ausnutzen kann.
Haben alle Elemente das Attribut `flex-grow`auf 1 gesetzt bekommen, wird der Platz zwischen den Elementen gleichmäßig unter seinen Kind-Elementen aufgeteilt.
Code:

```
.item {
flex-grow: <Za hl> ;
}
```

Negative Zahlen sind nicht zulässig! Der Standardwert für `flex-grow` ist 0.

flex-grow



flex-shrink

Definiert die Fähigkeit eines Kind Elementes zu schrumpfen.
Code:

```
.element {
flex-s hrink: <za hl> ;
}
```

Standardeinstellung für den Zahlenwert ist 1. Negative Zahlen sind nicht erlaubt.

flex-basis

Definiert die Ausgangsgröße eines Elementes bevor der verbliebene Platz verteilt wird. Es kann sowohl ein Prozentwert (z.B. 25%), als auch ein fixer Wert (z.B. 3em|125px) angegeben werden. Aber auch ein Schlüsselwort ist möglich. Der Schlüssel `auto` bedeutet, "-beziehe dich auf die mir zugewiesene Breite/Höhe". Der Schlüssel `content` bedeutet, die Größe ist anhängig vom Inhalt. Leider wird dieser Schlüssel noch nicht sonderlich gut unterstützt, daher ist es schwierig zu sagen, wie das Element sich verhält, ganz zu schweigen von seinen Abarten `max-content`, `min-content`und `fit-content`..

flex-basis (cont)

Code:

```
.element {
flex-b asis: <we rt> | <sc hlü sse lwo rt> ;
}
```

Wird der Wert auf `0`gesetzt,

flex-basis