Front-End Web Development

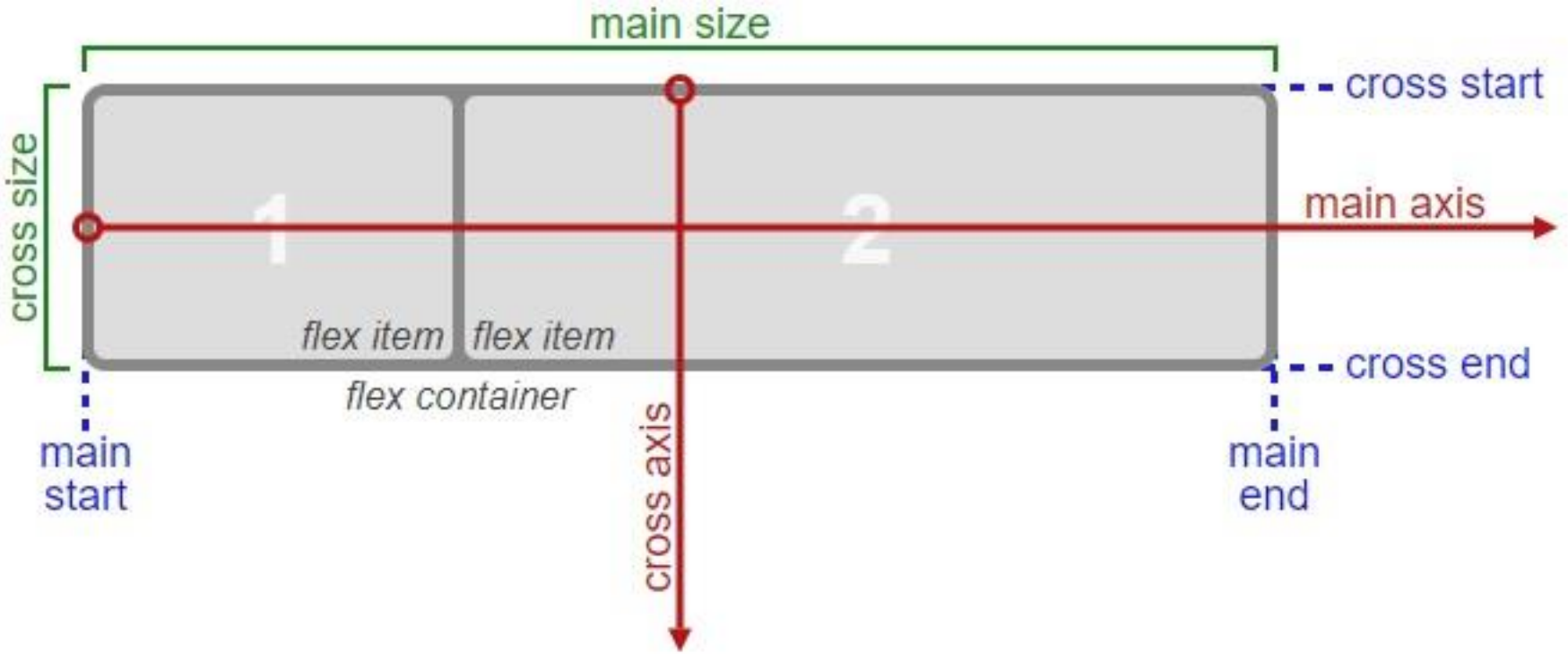
CSS Flexbox

Flexible Box Layout Module

- **CSS Flexbox** е мощен инструмент при респонсив дизайна. Създава гъвкаво, отзивчиво и ефективно оформление на елементите на уеб страницата. Основава се на съдържанието. Обработва оформлението за конкретен контейнер и елементите, които са пряко дъщерни на този контейнер. Гъвкавите елементи от своя страна също могат да бъдат гъвкави контейнери за елементите, които съдържат, което улеснява създаването на сложни оформления.
- Flexbox оформлението позволява на контейнера да подравнява, разпределя, премества и да променя ширината и/или височината на своите елементи и реда им, за да запълни най-добре наличното пространство. Голямото предимство е, че оформленията на flexbox работят, без да знаят размера на елементите във всеки контейнер или когато размерите са динамични. Тези гъвкави контейнери ще променят размера на елементите в тях, за да паснат на екраните. Разширяват елементите, за да запълнят свободното пространство или ги намаляват, за да предотврати преливане. Освен това, не се редактират със свойства като float, clear, vertical-align и други.

- Flexbox се основава на концепцията за редове и колони, но работи само върху редове или колони наведнъж. Едноизмерен. Във flexbox оформлението главната ос по подразбиране е „ред“, хоризонтална, а напречната ос е перпендикулярна на главната ос (по подразбиране е „колона“, вертикална).
- Гъвкавото flexbox оформление се основава и на „гъвкави посоки на потока“

  flex-flow directions.
- Спецификация по W3C:



Съдържанието на гъвкав контейнер:

- може да се постави във всяка посока на потока (наляво, надясно, надолу или дори нагоре),
- редът на показване на елементите може да бъде обърнат или пренареден в стиловия слой (т.е. визуалният ред може да бъде независим от реда на източника и речта),
- елементите могат да бъдат разположени линейно по една (главна) ос или «увити» в множество линии по второстепенна (напречна) ос,
- елементите могат да „огъват“ размерите си, за да отговорят на наличното пространство,
- елементите могат да бъдат подравнени по отношение на техния контейнер или един към друг на вторичния (напречен),
- съдържанието може да бъде динамично сгънато или разгънато по главната ос, като същевременно се запази напречният размер на контейнера.

- **Flexbox** е цял модул и има собствен набор от свойства. Някои от тях са предназначени да бъдат зададени върху контейнера (родителски елемент, известен като „гъвкав контейнер“), докато другите са предназначени да бъдат зададени върху дъщерните (наричани „гъвкави елементи“).
- .container {

  display: flex;

  }

Всички елементи, които се намират в div с клас име container ще се подредят един до друг. Веднъж приел свойство flex, контейнерът вече може да „преподрежда“ child елементите си в най-различни конфигурации.

- display: flex; Тази стойност кара даден елемент да генерира гъвкава контейнерна кутия, която е на блоково ниво, когато е поставена в оформление на потока.
- display: inline-flex; Тази стойност кара даден елемент да генерира гъвкава контейнерна кутия, която е на поредово ниво, когато е поставена в оформление на потока.

Оформление на parent контейнер

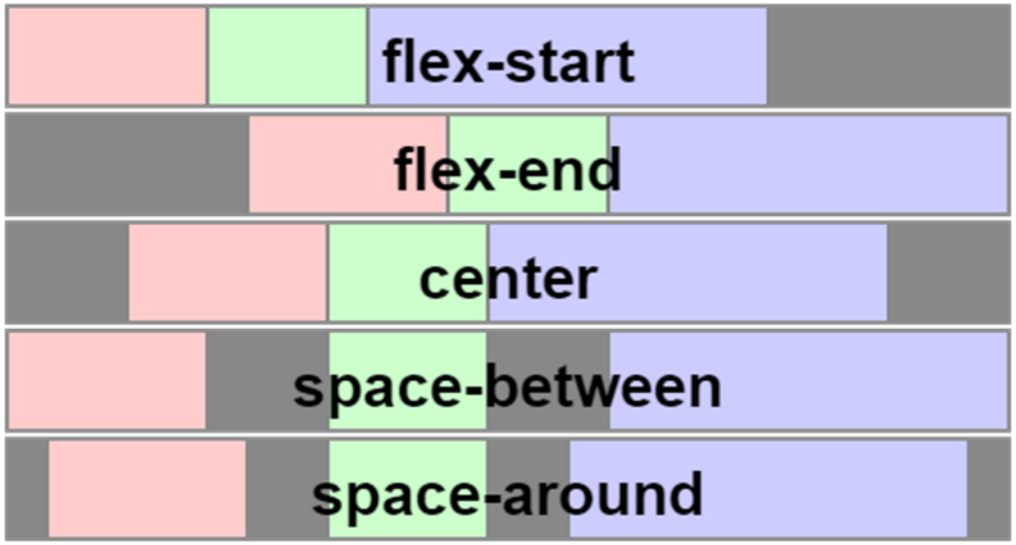
- Хоризонтално подравняване – **justify-content** - определя подравняването по главната ос.
- Посредством justify-content се разпределя допълнителното свободно пространство, което остава между child елементите на контейнера. Гъвкавите елементи могат да се подравнят към началото на хоризонталната линия (по подразбиране), към края, към средата (ако пространството е по-малко - гъвкавите елементи ще препълват еднакво и в двете посоки), с еднакво разстояние между елементите (залепени първи и последен елемент по границите), с еднакво разстояние преди, между и след елементите (така между елементите визуално разстоянията са двойни), с еднакво разстояние около тях.

- Възможните стойности на justify-content са:

justify-content: flex-start(първоначално)|flex-end|center|space-between|space-around|space-evenly|initial|inherit;

- Свойството justify-content подравнява елементите, когато те не използват цялото налично пространство на главната ос (хоризонтално).

justify-content



Оформление на parent контейнер

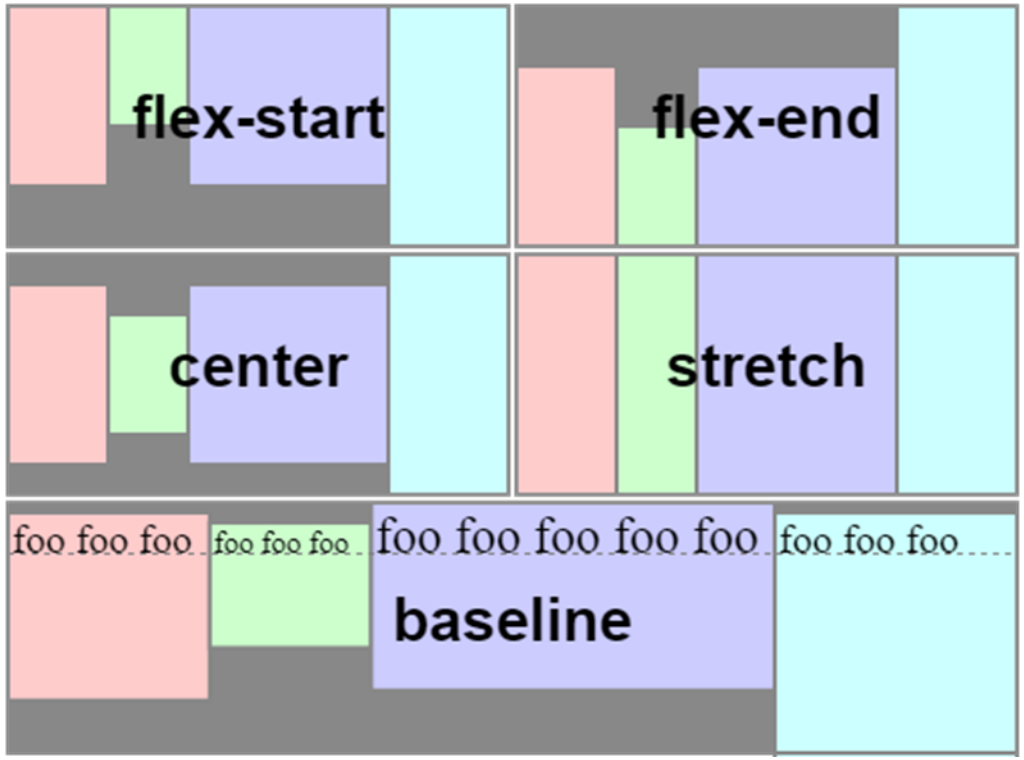
- Вертикално подравняване – **align-items** - указва на контейнера, който вече има display: flex; как да подреди съдържащите си child елементи по напречната ос. Гъвкавите елементи могат да се подравнят към напречния начален ръб на линията, към края, към средата (ако пространството е по-малко - гъвкавите елементи ще препълват еднакво и в двете посоки), с разтягане (по подразбиране – подходящо, ако текстът в различните елементи е с различна дължина), с подравнени базови линии (базовите линии да са подравнени, а елементът с най-голямо разстояние между основната линия и ръба на напречния начален margin се поставя изравнен с напречния начален ръб на линията).

- Възможните стойности на align-items са:

align-items: normal|stretch|center|flex-start|flex-end|baseline|initial|inherit;

- Вертикално подравнява гъвкавите елементи, когато те не използват цялото налично пространство по напречната ос.

align-items


flex-start
flex-end
center
stretch
foo foo foo
foo foo foo
foo foo foo foo foo
foo foo foo
baseline

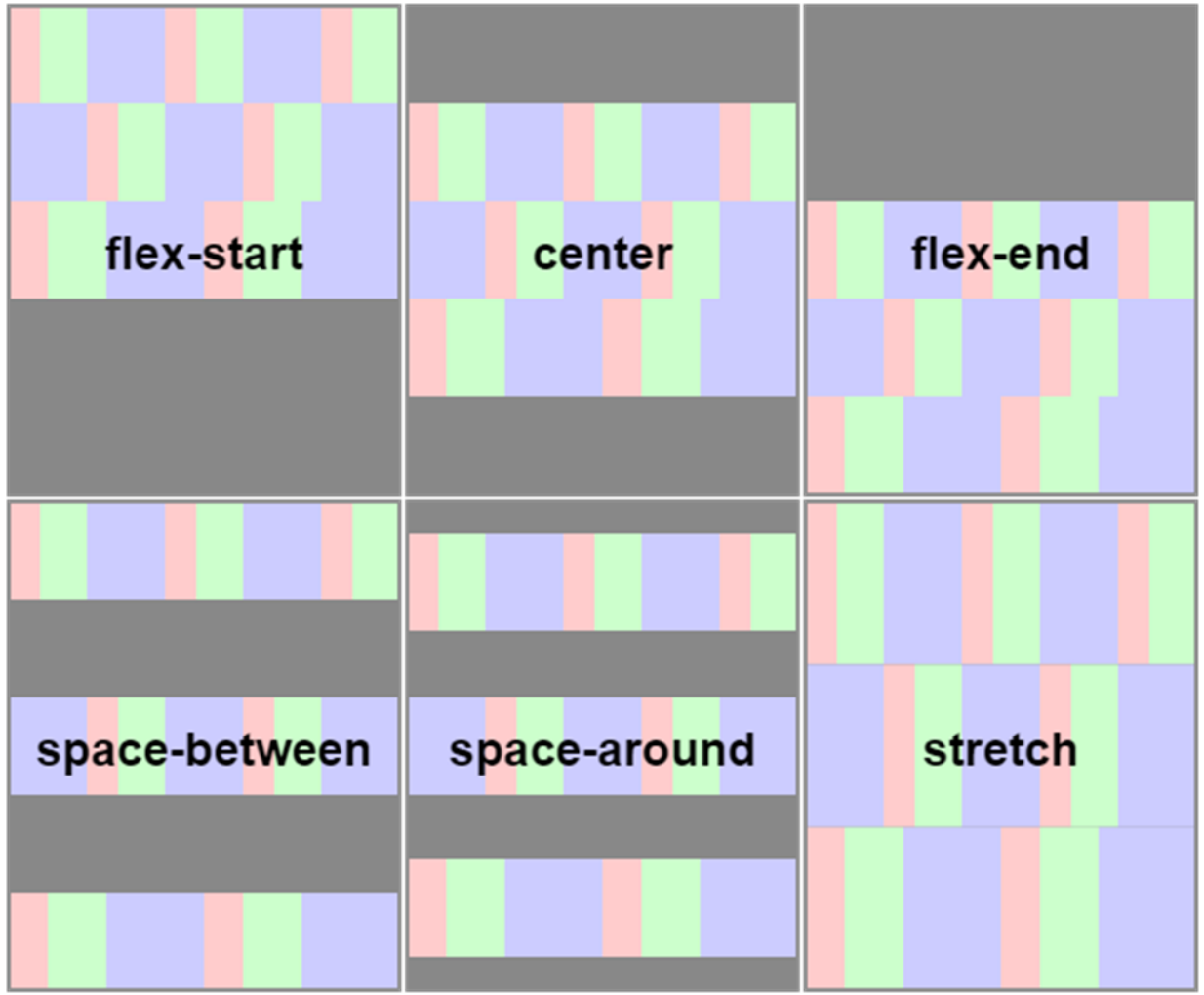
Оформление на parent контейнер

- Вертикално подравняване – **align-content** - указва на контейнера как да подреди съдържащите си елементи по перпендикулярната ос (аналогично на justify-content, но по вертикалната ос). Няма ефект върху едноредов гъвкав контейнер.

- Възможните стойности на align-content са:

align-content: stretch|center|flex-start|flex-end|space-between|space-around|space-evenly|initial|inherit;

align-content



Оформление на parent контейнер

- **Flex-wrap** - По подразбиране всички елементи в контейнер с display: flex; ще се съберат в един ред (независимо колко на брой са). С flex-wrap може да се контролира това поведение и да се зададе дали елементите да се побират в един ред или да продължават на втори ред, в случай, че общата им ширина надвишава тази на родителя им. Flex-wrap се използва често, когато child елементите в контейнера трябва да са с фиксирана ширина.

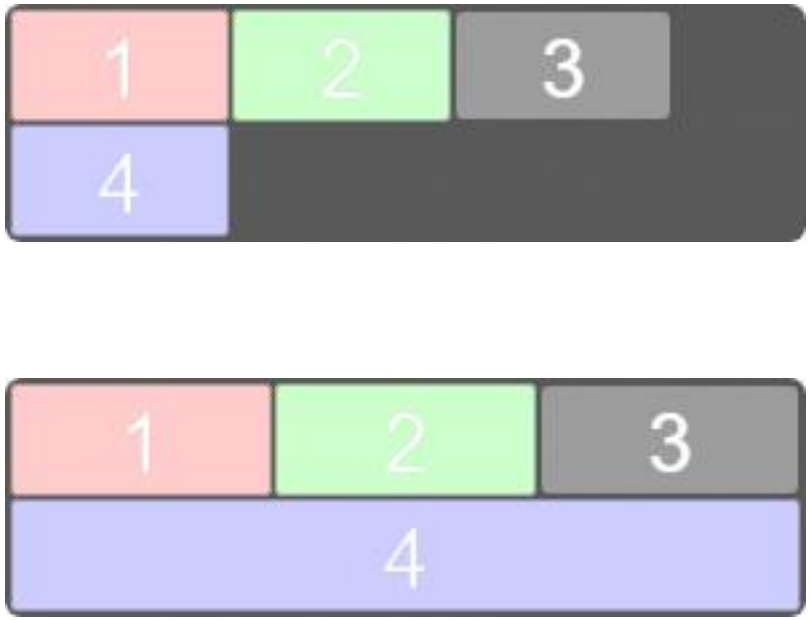
- Възможните стойности на flex-wrap са:

flex-wrap: nowrap (по подразбиране – едноредов, с препълване)|

wrap (многоредов, без препълване, попълване от горе на долу)|

wrap-reverse (попълване от долу на горе)|initial|inherit;

flex-wrap



+ flex: auto

Оформление на parent контейнер

- **Flex-direction** указва посоката на гъвкавите елементи. Използва се за смяна на позициите при различни екрани само с един ред в media query.
- Възможните стойности на flex-direction са:

flex-direction: row(по подразбиране)|row-reverse|column|column-reverse |initial|inherit;

- flex-direction: row|row-reverse – отговарят на inline ос
- flex-direction: column|column-reverse – отговарят на block ос

Оформление на parent контейнер

- **Flex-flow** е сборен.

Синтаксисът на flex-flow е:

flex-flow: flex-direction flex-wrap |initial|inherit;

- div { flex-flow: row; } /* по подразбиране - row nowrap.

- div { flex-flow: column wrap; } /* Основната ос е block посока (top към bottom).

- div { flex-flow: row-reverse wrap-reverse; }

Оформление на child елементите

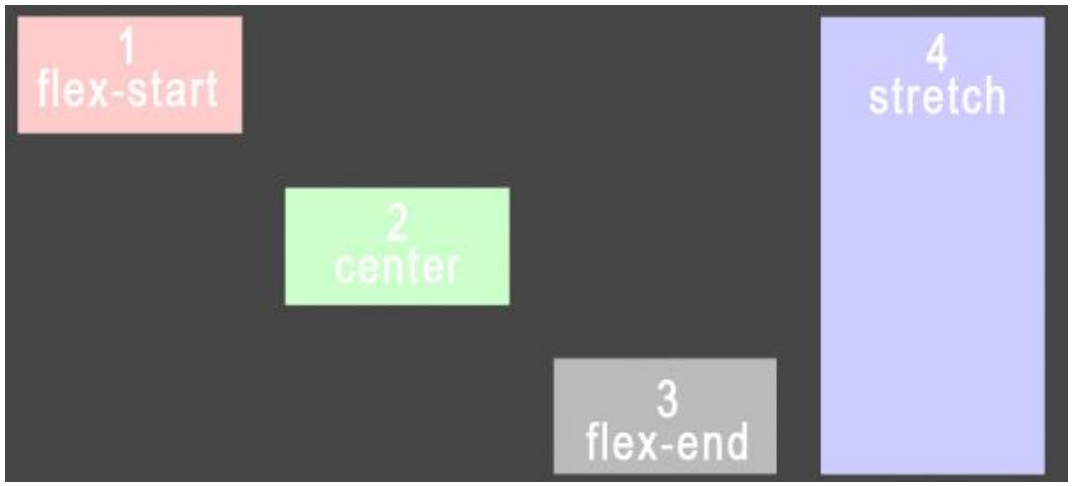
С **Flexbox** може да се зададат настройки и на самите child елементи в контейнера.

- **align-self** указва къде по вертикалната ос да се подравни даден (само той) child елемент във flex контейнера. Това ще замени align-items настройката на самия контейнер за този елемент.
- Възможните стойности на align-self са:

align-self: auto (първоначална стойност, предоставя подравняване на родителя)|

flex-start | flex-end | center| stretch|

baseline (елементът е позициониран към базовата линия на контейнера;}



Оформление на child елементите

- order - по подразбиране, елементите в flex контейнера се подреждат в реда, в който се появяват в кода. С order може да се промени това и да се зададе последователност на зареждане в DOM дървото на всеки един от елементите. Касае визуалното подреждане. Полезно, когато трябва различна подредба на елементите за различните резолюции (например за smart устройства child елемент с основното съдържание да става първи, преди лява секция).
- Възможните стойности на order са:

order: number(цяло число)|initial|inherit;

.item {border: 2px;}

.item-1 {order: 5; }

. item-2 {order: 1;}

.item:nth-child(1) {order: 3;} .item:nth-child(4) {order: 1;}

Оформление на child елементите

Определящият аспект на гъвкавото оформление е възможността да се направят гъвкавите елементи „гъвкави“, променяйки тяхната ширина / височина, за да запълнят наличното пространство в основното измерение.

- flex-grow - указва колко ще нарасне елементът спрямо останалите гъвкави елементи в същия контейнер. Примерно, даден елемент с flex-grow: 2; ще заема двойно повече място от други два елемента с flex-grow: 0;. Когато е пропуснато, стойността е 0 - означава, че елементът няма да заема налично празно място. Когато е 1, елементите могат да растат.

flex-grow: number|initial|inherit;

- .item {flex-grow: 1;}

Оставащото пространство в контейнера ще бъде разпределено поравно между всички.

- .item1 {flex-grow: 1; } .item2 {flex-grow: 1; } .item3 {flex-grow: 1; } .item4 {flex-grow: 2; }

Четвъртият елемент е два пъти по-голям, а другите са равномерни.

- .item1 {flex-grow: 0.5;} .item2 {flex-grow: 0.3;}

Между 0 и 1 – ако сборът е по-малък от 1, заемат по-малко от 100% пространство.

Оформление на child елементите

- **flex-shrink** указва как елементът ще се свие спрямо останалите гъвкави елементи в същия контейнер. Когато е пропуснато, стойността е 1 – т.е. Елементите могат да се свиват.

flex-shrink: number|initial|inherit; .item4 {flex-shrink: 2; }

- **flex-basis** задава гъвкавата основа - дефинира размера по подразбиране на елемент (например 20%, 5 rem и т.н.), преди да бъде разпределено оставащото пространство - преди да се случи каквото и да е опаковане, нарастване или свиване. Може да се използва вместо свойството ширина или височина.

flex-basis: number|auto (извлича основния размер, като width)|content (размер въз основа на съдържанието на гъвкавия елемент)|initial|inherit;

- .item4 { flex-basis: 200px }
- .item1-3 { flex-basis: auto; /* default auto */ } (приемаме, че flex-grow: 0, защото не е споменато)

Ако е зададено auto, допълнителното пространство се разпределя въз основа на неговата стойност за flex-grow. Ако е зададено 0, допълнителното пространство около съдържанието не се взема под внимание.

- flex-grow и flex-shrink имат стойности (0, 1, 2 и т.н.),
- flex-basis приема стойности (px, rem, content и т.н.).

Оформление на child елементите

- Flex е сборен. Насърчава се да се ползва, защото правилно нулира всички неуточнени компоненти, за да се приспособят към обичайните употреби, а и трите свойства, които съдържа, работят заедно.

flex: flex-grow flex-shrink flex-basis|auto|initial|inherit;

.item4 {flex: 1 1 200px; }

- Първоначални стойности: flex: 0 1 auto; Оразмерява елемента въз основа на свойствата за ширина / височина. (Ако основното свойство за размер на елемента се изчислява автоматично, това ще оразмери гъвкавия елемент въз основа на неговото съдържание). Прави гъвкавия елемент негъвкав, когато има положително свободно пространство, но му позволява да се свие до минималния си размер, когато няма достатъчно място.
- flex: 0 0 auto; Тази стойност оразмерява елемента според свойствата на ширината / височината, но прави гъвкавия елемент напълно негъвкав. Това е подобно на първоначалното, с изключение на това, че гъвкавите елементи не могат да се свиват, дори в ситуации на преливане. flex: none;
- flex: 1 1 auto; Оразмерява елемента въз основа на свойствата ширина / височина, но ги прави напълно гъвкави, така че да поемат всяко свободно пространство по главната ос.
- Ако е само число - flex: 5; означава flex-grow:5 flex-shrink:1 flex-basis:0% т.е. flex: 5 1 0; Елементът получава определената част от свободното пространство.
- Ако е 1 - flex: 1; всички гъвкави елементи са с еднаква дължина, независимо от съдържанието им.

1 „включва“ - позволява на даден елемент да расте или да се свива.

0 „изключва“ - предотвратява нарастването или свиването на елемента.

Оформление на child елементите

- flex: 1;

/* One relative value: flex-grow */

- flex: 20rem;
- flex: 50px;

/* One absolute value: flex-basis */

- flex: 1 20rem;

/* One relative and one absolute value: flex-grow | flex-basis */

- flex: 1 2;

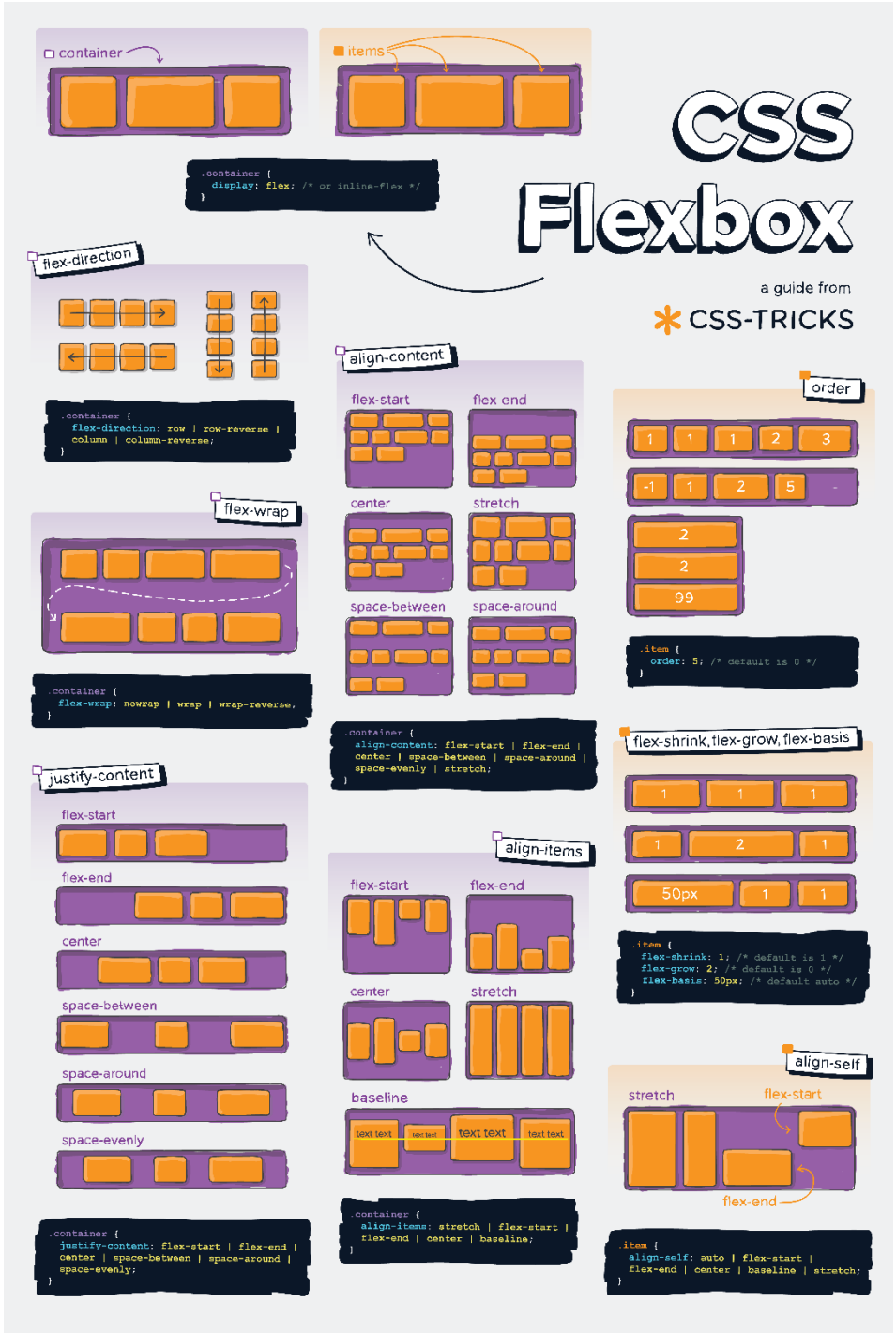
/* Two relative values: flex-grow | flex-shrink */

- flex: 1 2 20rem;

/* Three values: flex-grow | flex-shrink | flex-basis */

- flex: none;

/* Fully inflexible layout: equivalent of flex: 0 0 auto */


container
items
.container {
display: flex; /* or inline-flex */
}
CSS
Flexbox
a guide from
CSS-TRICKS
flex-direction
.container {
flex-direction: row | row-reverse |
column | column-reverse;
}
flex-wrap
.container {
flex-wrap: nowrap | wrap | wrap-reverse;
}
justify-content
flex-start
flex-end
center
space-between
space-around
space-evenly
.container {
justify-content: flex-start | flex-end |
center | space-between | space-around |
space-evenly;
}
align-content
flex-start
flex-end
center
stretch
space-between
space-around
.container {
align-content: flex-start | flex-end |
center | space-between | space-around |
space-evenly | stretch;
}
align-items
flex-start
flex-end
center
stretch
baseline
text text
.container {
align-items: stretch | flex-start |
flex-end | center | baseline;
}
order
1 1 1 2 3
-1 1 2 5
2
2
99
.item {
order: 5; /* default is 0 */
}
flex-shrink, flex-grow, flex-basis
1 1 1
1 2 1
50px 1 1
.item {
flex-shrink: 1; /* default is 1 */
flex-grow: 2; /* default is 0 */
flex-basis: 50px; /* default auto */
}
align-self
stretch
flex-start
flex-end
.item {
align-self: auto | flex-start |
flex-end | center | baseline | stretch;
}

Респонсив дизайн с flexbox

- **Промяна на посоката** - пример: Да се създаде оформление с две колони за по-големи екрани и оформление с една колона за малки размери (като телефони и таблети). Променя се посоката на гъвкавост от ред на колона в конкретна точка на прекъсване (примерно при 800px):

```
.container { display: flex; flex-direction: row;}
@media (max-width: 800px) {
    .container { flex-direction: column; } }
```

https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_responsive

- Друг начин е **промяната на процента** на свойството flex на гъвкавите елементи, за да се създадат различни оформления за различни размери на екрана. Трябва да се включи и flex-wrap: wrap; задължително.

```
.container { display: flex; flex-wrap: wrap;}
.flex-item-left { flex: 50%;}
.flex-item-right { flex: 50%;}
@media (max-width: 800px) {
    .flex-item-right, .flex-item-left { flex: 100%; } }
```

https://www.w3schools.com/css/tryit.asp?filename=trycss_mediaqueries_img_gallery

https://www.w3schools.com/css/tryit.asp?filename=trycss_mediaqueries_flex

https://www.digitalocean.com/community/cheatsheets/css-flexbox

https://flexboxfroggy.com/#bg