

## ЛЕКЦІЯ 4

### Основи верстки за допомогою Flexbox технології

#### ПЛАН:

- 1 Поняття про Flexbox.
- 2 Основи синтаксису.
- 3 Основні властивості Flex-контейнера.
- 4 Багаторядкова організація блоків всередині Flex-контейнера
- 5 CSS правила для дочірніх елементів Flex-контейнера (Flex-блоків)



Сучасний Front-End розробник активно повинен вміти застосовувати на практиці різні інструменти, що дозволяють автоматизувати процес верстки макетів і програмування клієнтської складової проекту. Для цього вже існує безліч фреймворків, як великих, так і малих, системи складання, пакетні менеджери, ціла купа пакетів для задач будь-якого рівня, препроцесори, шаблонізатори та інші методи, які створені спростити і підвищити продуктивність роботи фахівця в даній області.

На жаль, початківцям буває складно розібратися в достатку цих інструментів, так як тут потрібно постійно розвиватися і знати основи верстки. І багато хто з цієї причини продовжують верстати звичайний css, з великою кількістю повторюваних селекторів, верстка без розуміння принципу різних сіток, пуста трата часу з різного роду позиціонуванням і т.д.

#### 1 Поняття про Flexbox

**Flexbox (флексбокс)** – це новий модуль, який є в CSS3, призначений для більш ефективної верстки за рахунок гнучкого розташування елементів, як по горизонталі, так і по вертикалі. Даний модуль вийшов на зміну звичній верстці за допомогою float і position. Він був розроблений для того, щоб запропонувати інший спосіб розташування елементів, і заснований на

іншому способі мислення. Флексбокс пропонує більше можливостей позиціонування і можливостей визначати відносини між елементами, ніж властивості float і position – бо флексбокс дозволяє вам розташовувати елементи відносно один одного. Це важливо. З float і position елементи розташовуються відносно сторінки. З флексбоксом елементи співвідносяться тільки між собою (див. рисунок 1).

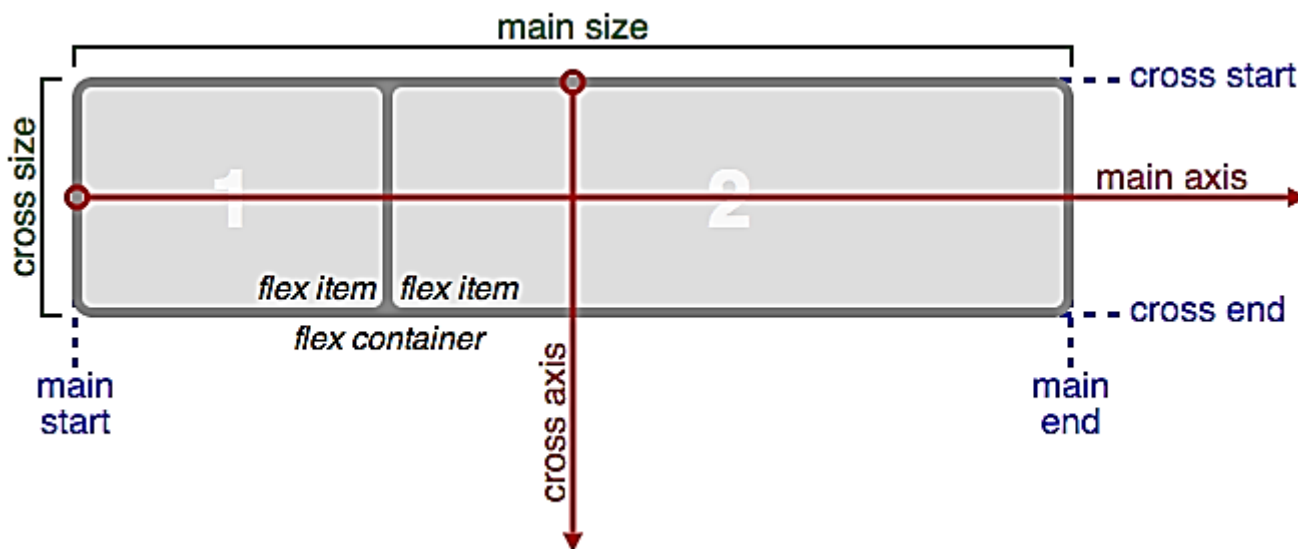


Рисунок 1 – Використання Flexbox

Вся сила Flexbox вражає, тому що ті елементи, які раніше доводилося верстати використовуючи різні способи, тепер можна створювати дуже просто.

На поточний момент більше 90% браузерів в тій чи іншій мірі підтримують Flexbox. А це значить, що ми можемо потихеньку впроваджувати цю технологію (див. рисунок 2).

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			62		10.2				
		57	63		10.3				4
11	16	58	64	11	11.2	all	64	11.8	6.2
	17	59	65	11.1	11.3				
		60	66	TP					
		61	67						

Рисунок 2 – Браузерна підтримка технології Flex

### Flexbox має ряд переваг:

1. Сама назва «flex» дає зрозуміти, що всі блоки дуже легко можна зробити гнучкими. Необхідні елементи можна стиснути і розтягнути за спеціальними правилами, зайнявши таким чином весь потрібний простір.
2. Базова лінія тексту легко вирівнюється по вертикалі і горизонталі.
3. Порядок розташування елементів в шаблоні не грає вирішальну роль. При необхідності його можна поміняти в стилях, що є особливо важливим для деяких аспектів адаптивної верстки.

4. Для заповнення всього заданого місця елементи можуть автоматично вирівнюватися в кілька рядків або стовпців.

5. Більшість мов світу мають форму написання справа-наліво rtl (right-to-left), на відміну від звичного нам ltr (left-to-right). Особливість Flexbox полягає в тому, що він адаптований для цього, так як для нього є поняття початку і кінця, а не права і ліва. У браузерях з налаштуванням rtl всі елементи автоматично розташовуються в реверсному порядку.

6. Синтаксис правил CSS інтуїтивно простий і досить швидко освоюється.

## 2 Основи синтаксису:

Спочатку ми повинні «повідомити» браузер, що хочемо використовувати адаптивне представлення, відповідно до якого будь-який браузер буде відкривати сторінку в цьому режимі, робиться це ось таким оголошенням між тегами head документа:

```
01 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-
02 to-fit=no">
```

Так як **flexbox** – це цілий модуль, а не просто одинична функція, він об'єднує в собі безліч властивостей. Деякі з них мають застосовуватися до контейнера (батьківського елемента, так званому флекс-контейнеру), в той час як інші властивості застосовуються до дочірніх елементів, або флекс-елементів.

Для початку використання потрібно вказати контейнеру `display:flex` або `display: inline-flex`.

**Display** (дисплей) – допоміжна складова гнучкого інлайнового або блочного контейнера, за допомогою якої активуються дочірні елементи програми для подальшої роботи з ними. Варто відзначити, що колонки, які входять в концепцію функціонування CSS ніяким чином не впливають на контейнер, а лише виступають в якості допоміжних ланок.

### HTML

```
01 <div class="flex_container">
02   <div class="flex_block">Flex блок 1</div>
03   <div class="flex_block">Flex блок 2</div>
04   <div class="flex_block">Flex блок 3</div>
05   <div class="flex_block">Flex блок 4</div>
06 </div>
```

### CSS

```
01 .flex_container {
02   display: flex; /*або inline-flex*/
03 }
```

## 3 Основні властивості флекс-контейнера

**main start / main end** – основні елементи контейнера, розташовані уздовж головних осей, при цьому починається рух з точок start і закінчується – в точці end;

Одне з основних понять в flexbox – це осі.

**main axis** – головна вісь флекс-контейнера, що задає напрямок розташування всіх дочірніх елементів контейнера, при цьому її рух не прив'язаний до горизонтального розміщення і залежить від властивостей `flex-direction`;

**cross axis** – перпендикулярна вісь по відношенню до головної осі, яка може змінювати напрямок в залежності від встановлених координат основної осі;

**cross end / cross start** – осі, які заповнюються елементами контейнера по точках.

**main size (cross size)** – висота або ширина елементів flex;

### Напрямок головної осі – властивість **flex-direction**

За замовчуванням, головна вісь в ltr локалі розташовується зліва направо, а перпендикулярна вісь – зверху вниз. При використанні базової css властивості flex-direction можна задавати напрямок головної осі flex-контейнера (див. рисунок 3).

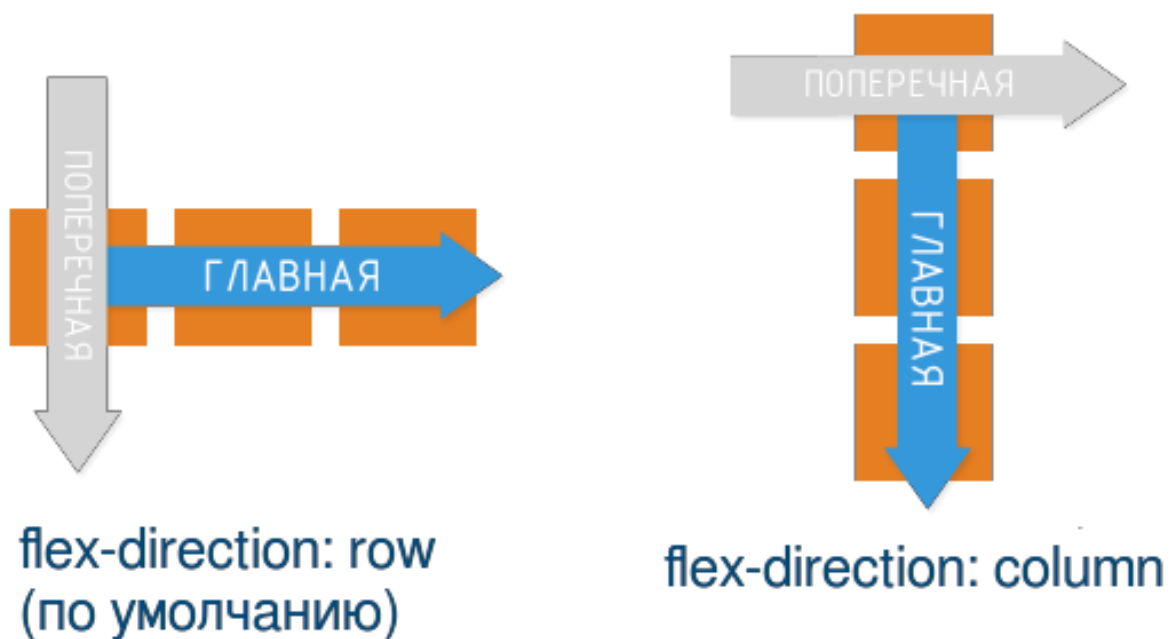


Рисунок 3 – Напрямок головної осі flex-direction

### Доступні значення **flex-direction** (див. рисунок 4):

- row (значення за замовчуванням): зліва направо (в rtl справа наліво);
- row-reverse: справа наліво (в rtl зліва направо);
- column: зверху вниз;
- column-reverse: знизу вверху;

```
01 .flex_container {  
02   flex-direction: row | row-reverse | column | column-reverse  
03 }
```

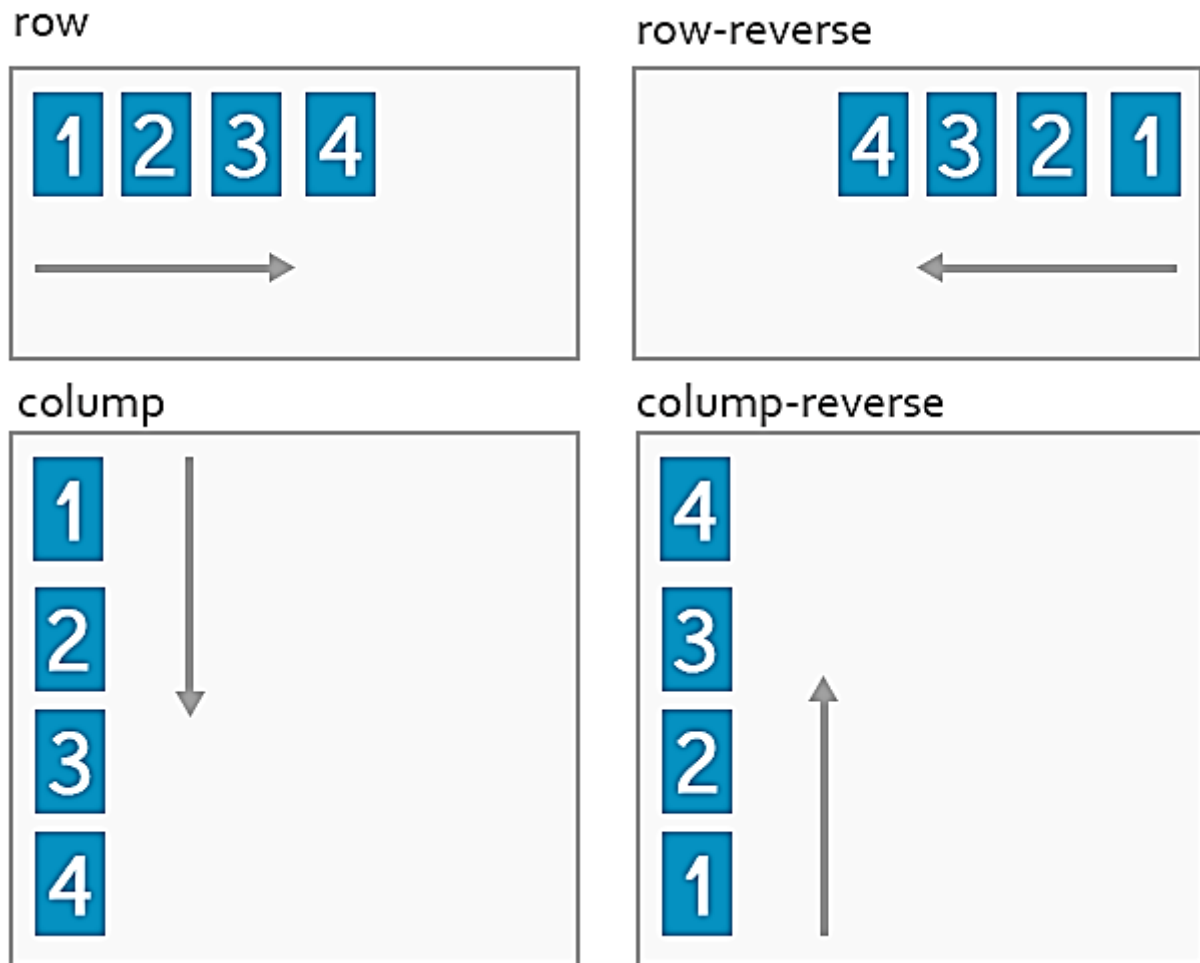


Рисунок 4 – Доступні значення flex-direction

### Вирівнювання елементів по головній осі – властивість justify-content

CSS властивість **justify-content** визначає те, як будуть вирівнюватися елементи уздовж головної осі. Крім цього, дана властивість допоможе встановити максимальну або допустиму відстань між блоками, що дозволить грамотно скоординувати вільний простір.

CSS

```
01 .flex_container {
02   justify-content: flex-start | flex-end | center | space-between |
03   space-around
04 }
```

#### Доступні значення justify-content (див. рисунок 5):

- flex-start (значення за замовчуванням): блоки притискаються до початку головної осі
- flex-end: блоки притиснуті до кінця головної осі
- center: блоки розташовуються в центрі головної осі
- space-between: перший блок розташовується на початку головної осі, останній блок – в кінці, всі інші блоки рівномірно розподілені просторі який залишився.
- space-around: всі блоки рівномірно розподіляються вздовж головної осі, розділяючи порівну весь вільний простір.

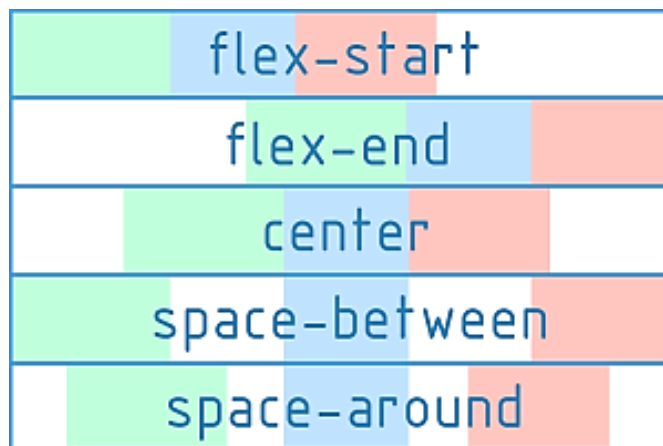


Рисунок 5 – Доступні значення justify-content

### Вирівнювання по перпендикулярній осі – властивість align-items

Css властивість align-items визначає те, як будуть вирівнюватися елементи уздовж перпендикулярної осі.

CSS

```
01 .flex_container {
02   align-items: flex-start | flex-end | center | space-between | space-
03   around
04 }
```

#### Доступні значення align-items (див. рисунок 6):

- flex-start: блоки притиснуті до початку поперечної осі;
- flex-end: блоки притиснуті до кінця поперечної осі;
- center: блоки розташовуються в центрі поперечної осі;
- baseline: блоки вирівняні по початковому рівні (базовому);
- stretch (значення за замовчуванням): блоки розтягнуті, займаючи все доступне місце по перпендикулярній осі, при цьому все-таки враховуються min-width / max-width, якщо такі задані.

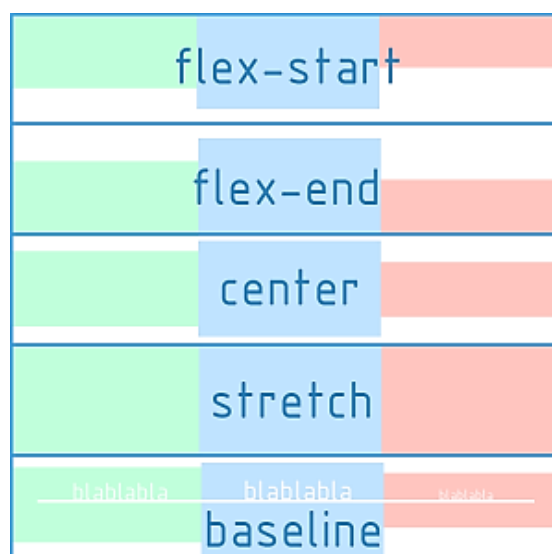


Рисунок 6 – Доступні значення align-items

CSS властивості **flex-direction**, **justify-content**, **align-items** повинні застосовуватися безпосередньо до флекс-контейнеру, а не до його дочірніх елементів (див. рисунок 7).

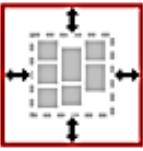
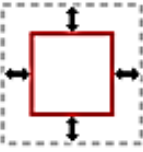
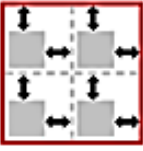
Common	Axis	Aligns	Applies to
' <a href="#">justify-content</a> '	inline	content within element (effectively adjusts padding) 	block containers, <a href="#">flex containers</a> , and <a href="#">grid containers</a>
' <a href="#">align-content</a> '	stacking		block containers, <a href="#">flex containers</a> , and <a href="#">grid containers</a>
' <a href="#">justify-self</a> '	inline	element within parent (effectively adjusts margins) 	block-level elements and <a href="#">grid items</a>
' <a href="#">align-self</a> '	stacking		<a href="#">flex items</a> and <a href="#">grid items</a>
' <a href="#">justify-items</a> '	inline	items inside element (controls child items' justify-self: auto) 	<a href="#">grid containers</a>
' <a href="#">align-items</a> '	stacking		<a href="#">flex containers</a> and <a href="#">grid containers</a>

Рисунок 7 – Застосування властивостей до самого контейнера

#### 4 Багаторядкова організація блоків всередині флекс-контейнера

##### Flex-wrap

Всі вищенаведені приклади, були побудовані з урахуванням однорядкового (одностовпцевого) розташування блоків. Слід сказати, що за замовчуванням флекс-контейнер завжди буде мати у своєму розпорядженні блоки всередині себе в одну лінію. Але специфікація також підтримує багаторядковий режим. За багаторядковість всередині флекс-контейнера відповідає CSS властивість **flex-wrap**.

При цьому доступні такі варіанти (див. рисунок 8):

- wrap – блоки вільно переносяться і розміщуються в кілька рядів (напрямок зліва направо);
- nowrap – початкове значення, перший рядок залишається без переносів;
- wrap-reverse – перенесення блоків за потребою зліва направо і навпаки.

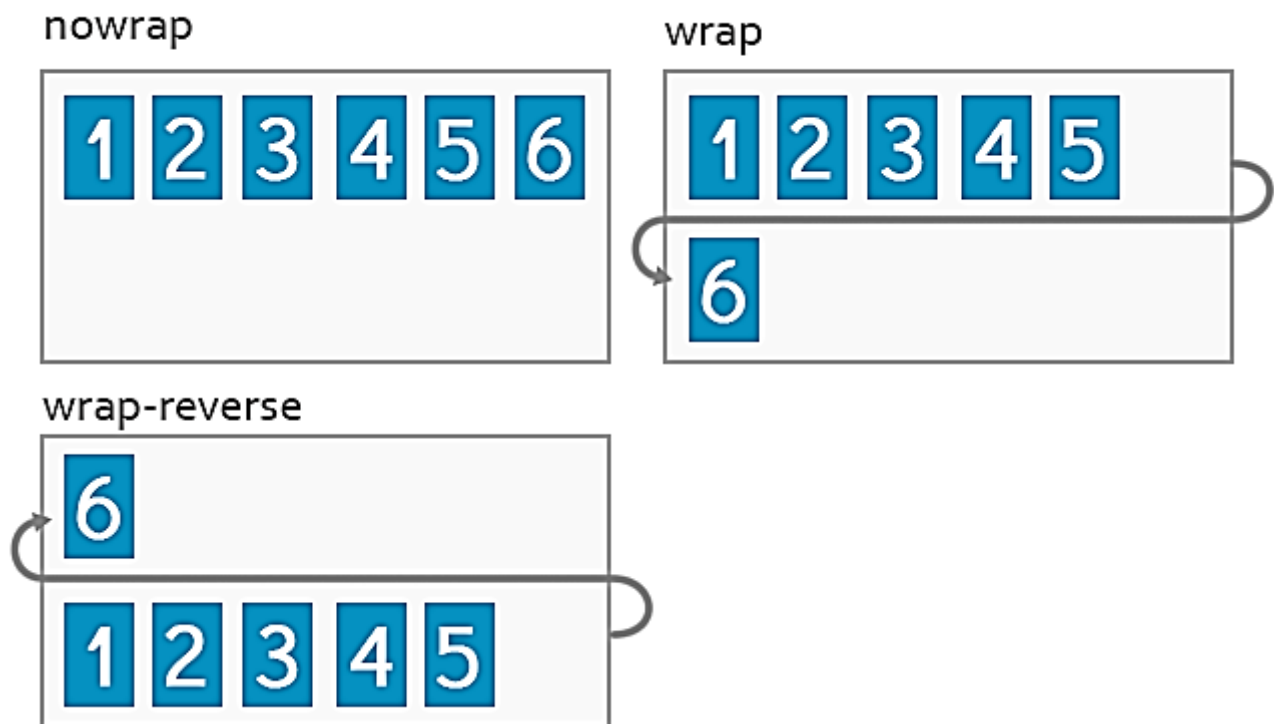


Рисунок 8 – Багаторядкове розміщення блоків всередині флекс-контейнера

### Flex-flow – зручне скорочення для flex-direction та flex-wrap

По суті, **flex-flow** надає можливість в одну властивість описати напрямок головної і багаторядкової перпендикулярної осі.

CSS

```
01 .flex_container {
02     flex-direction: column;
03     flex-wrap: wrap;
04 }
05 /*або*/
06 .flex_container{
07     flex-flow: column wrap;
08 }
```

За замовчуванням **flex-flow: row nowrap**.

### align-content

Існує також властивість **align-content**, яка визначає те, яким чином утворені ряди блоків будуть вирівняні по вертикалі і як вони поділять між собою весь простір флекс-контейнера.

**Важливо:** *align-content працює тільки в багаторядковому режимі (тобто у випадку flex-wrap:wrap; або flex-wrap:wrap-reverse;).*

```
01 .flex_container {
02     align-content: flex-start | flex-end | center | space-between | space-
03     around
04 }
```



Доступні значення align-content (див. рисунок 9):

- flex-start: ряди блоків притиснуті до початку flex-контейнера.
- flex-end: ряди блоків притиснуті до кінця flex-контейнера
- center: ряди блоків знаходяться в центрі flex-контейнера
- space-between: перший ряд блоків розташовується на початку flex-контейнера, останній ряд блоків – вкінці, всі інші ряди рівномірно розподілені в останньому просторі.
- space-around: ряди блоків рівномірно розподілені від початку до кінця flex-контейнера, розділяючи весь вільний простір порівну.
- stretch (значення за замовчуванням): Ряди блоків розтягнуті, щоб зайняти весь наявний простір.

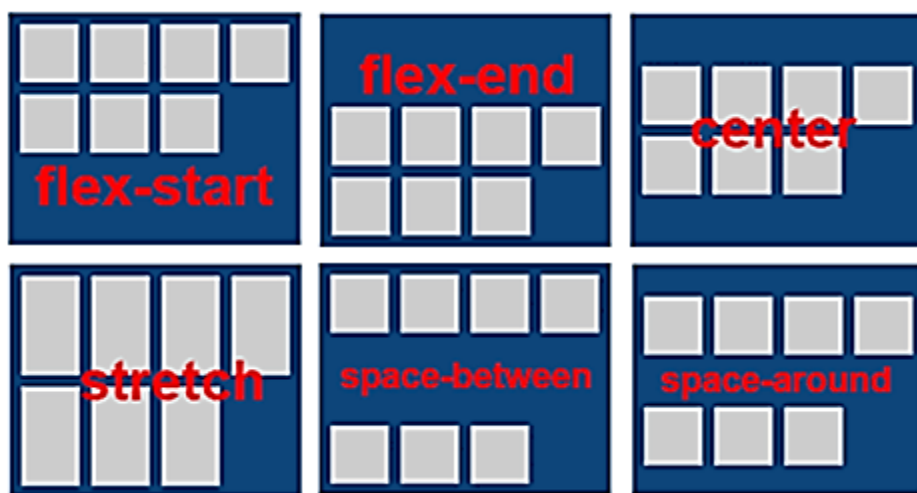


Рисунок 9 – Доступні значення align-content

## 5 CSS правила для дочірніх елементів flex-контейнера (flex-блоків)

**flex-basis** – базовий розмір окремо взятого flex-блоку. Задає початковий розмір по головній осі для flex-блоку до того, як до нього будуть застосовані перетворення, засновані на інших flex-факторах. Може бути заданий в будь-яких одиницях вимірювання довжини (px, em, %) або auto (за замовчуванням). Якщо розмір заданий як auto – за основу беруться розміри блоку (width, height), які, в свою чергу, можуть залежати від розміру контенту, якщо не вказані явно.

**flex-grow** – «жадібність» окремо взятого flex-блоку. Визначає те, на скільки окремих flex-блок може бути більший сусідніх елементів, якщо це необхідно. flex-grow приймає безрозмірне значення (за замовчуванням 0)

### Приклад 1:

Якщо всі flex-блоки всередині flex-контейнера мають flex-grow: 1, то вони будуть однакового розміру. Якщо один з них має flex-grow: 2, то він буде в 2 рази більше, ніж всі інші.

### Приклад 2:

Якщо всі flex-блоки всередині flex-контейнера мають flex-grow: 3, то вони будуть однакового розміру. Якщо один з них має flex-grow: 12, то він буде в 4 рази більше, ніж всі інші.

Тобто абсолютне значення flex-grow не визначає точну ширину. Воно визначає його ступінь «жадібності» по відношенню до інших flex-блоків того ж рівня.

**flex-shrink** – фактор «стиснення» окремо взятого flex-блоку. Визначає, наскільки flex-блок зменшиться відносно сусідніх елементів у flex- контейнері у випадку нестачі вільного місця. По замовчуванню дорівнює 1.

**flex** – короткий запис для властивостей **flex-grow**, **flex-shrink** і **flex-basis**:

CSS

```
01 .flex_block{
02     flex-grow:12;
03     flex-shrink:3;
04     flex basis: 30em;
05 }
06 /* або */
07 .flex_block{
08     flex: 12 3 30em;
09 }
```

**align-self** – вирівнювання окремо взятого flex-блоку по перпендикулярній осі. Дас можливість перевизначати властивість flex-контейнера **align-items** для окремого flex-блоку.

**Доступні значення align-self (ті ж 5 варіантів, що і для align-items):**

- flex-start: flex-блок притиснутий до початку перпендикулярної осі;
- flex-end: flex-блок притиснутий до кінця перпендикулярної осі;
- center: flex-блок розташовуються в центрі перпендикулярної осі;
- baseline: flex-блок вирівнюються по початковій лінії
- stretch (значення за замовчуванням): flex-блок розтягнутий, щоб зайняти весь

доступний простір по перпендикулярній осі, при цьому враховуються min-width / max-width, якщо такі задані.

**order** – порядок розміщення окремо взятого flex-блоку всередині flex-контейнера. За замовчуванням всі блоки будуть розміщуватися один за одним в порядку, заданому в html. Однак цей порядок можна змінити за допомогою властивості вирівнювання по baseline. Він задається цілим числом і за замовчуванням дорівнює 0. Значення order не задає абсолютну позицію елемента в послідовності. Воно визначає вагу позиції елемента.

HTML

```
01 <div class="flex_container">
02     <div class="flex_block" style="order: 5" >Flex блок 1</div>
03     <div class="flex_block" style="order: 10" >Flex блок 2</div>
04     <div class="flex_block" style="order: 5" >Flex блок 3</div>
05     <div class="flex_block" style="order: 5" >Flex блок 4</div>
06     <div class="flex_block" style="order: 0" >Flex блок 5</div>
07 </div>
```

В даному випадку, блоки будуть розміщуватися один за іншим уздовж головної осі в наступному порядку: Flex блок 5, Flex блок 1, Flex блок 3, Flex блок 4, Flex блок 2.

Підводячи підсумок варто зазначити, що Flexbox відкриває нові можливості для верстальників і розробників, при цьому наповнення контентом сайту стає менш трудомістким і більш продуктивним.

Підвищуйте свою оперативність і користуйтеся властивостями CSS на професійному рівні у створенні креативних інтернет проектів.

### КОНТРОЛЬНІ ЗАПИТАННЯ:

1. Що таке Flexbox?
2. Назвіть основні переваги використання технології
3. Які ви знаєте властивості flex-контейнера?
4. Що таке властивість flex-direction. Назвіть її доступні значення?
5. Що таке властивість justify-content?
6. Вкажіть основні значення властивості justify-content.
7. Що таке flex-wrap?
8. Які варіанти flex-wrap вам відомі?
9. Опишіть властивість align-content та її основні значення.
10. Назвіть основні CSS правила для дочірніх елементів flex-контейнера.