



All materials provided and/or made available contain Sigma Software LLC proprietary and confidential information and must not be copied, reproduced or disclosed to any third party or to any other person, other than those persons who have a bona fide need to review it for the purpose of participation in the online courses being provided by Sigma Software LLC. The intellectual property rights in all materials (including any trademarks) are owned by Sigma Software LLC or its associated companies, and a limited license, terminable at the discretion of Sigma Software LLC without notice, is hereby granted to you solely for the purpose of participating in the online courses being provided by Sigma Software LLC. Neither you or any other party shall acquire any intellectual property rights of any kind in such materials.

# Позиціювання

# Позиціонування

**Позиціонування дозволяє вилучати елементи з нормального потоку макета документа і змусити їх поводитися по-іншому; наприклад, розташовуватися один на одному або завжди залишатися на тому самому місці всередині вікна перегляду браузера.**



# Позиціонування

Базовою властивістю, що змінює спосіб позиціонування, є властивість **position**.

Можливі значення:

- **left**
- **right**
- **top**
- **bottom**



# Позиціонування

**position**

```
graph TD; position --> static; position --> relative; position --> fixed; position --> absolute; position --> sticky;
```

**static**

**relative**

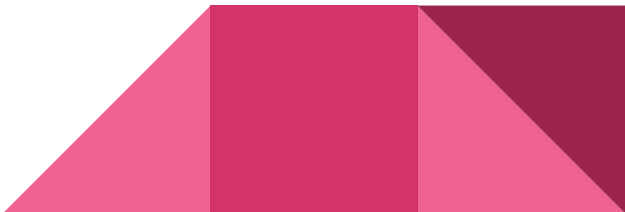
**fixed**

**absolute**

**sticky**

## Позиціонування - static

Значення за замовчуванням. Статичне позиціонування. Будь-який елемент, доданий на сторінку, матиме це значення і буде розташований у нормальному потоці документа відповідно до контексту форматування батьківського елемента. Властивості **left**, **right**, **top**, **bottom** та **z-index** ігноруються.



## Позиціонування - static

Елемент із значенням `position:static` не є  
позиційованим!!!



# Позиціонування

**Опорний елемент** - це такий, щодо якого або відносно якого позиціонуються дочірні елементи.

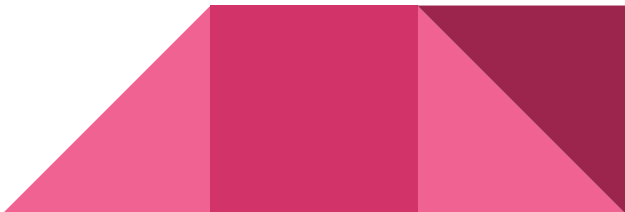




# Позиціонування - relative

**position: relative**

Елемент позиціонується так, як і статичний, але його відображення може бути зміщене щодо свого початкового положення за допомогою властивостей **left, right, top, bottom**



## Позиціонування - relative

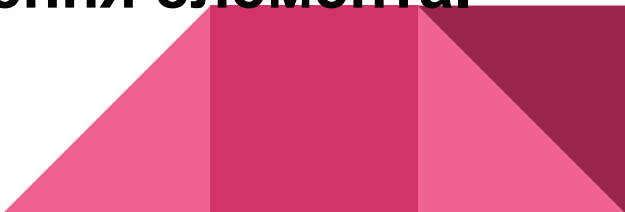
Елемент позиціонується так, як і статичний, але його відображення може бути зміщене щодо свого початкового положення за допомогою властивостей **left, right, top, bottom**

Example

# Позиціонування - absolute

**position: absolute**

Спосіб позиціонування, який здається найбільш зрозумілим. Ми просто задаємо абсолютне позиціонування та за допомогою властивостей **left**, **right**, **top**, **bottom** регулюємо положення елемента.



# Позиціонування - absolute

## Особливості:

- Елемент забирається з основного потоку документа. Тобто, перестає впливати на становище навколишніх елементів і розмір батька. Можна уявити, що елемент йде на шар вище і перестає взаємодіяти з усіма елементами, крім нащадків.



## Позиціонування - absolute

- Елемент позиціонується щодо найближчого позиціонованого предка. Тобто, браузер йде вгору по дереву елементів та шукає найближчого опорного батька. І потім має в своєму розпорядженні наш елемент щодо цього батька.



## Позиціонування - absolute

- Якщо елемент був частиною рядкового контексту форматування, він набуває блокового контексту форматування. До нього стає застосовна блокова модель.



## Позиціонування - absolute

- Якщо елемент був блоковим і займав всю ширину свого батька, то його ширина визначатиметься шириною контенту.



# Позиціонування - absolute

- Відступи елемента з **position: absolute** не схлопуються з відступами сусідніх елементів.

Example



# TEST YOUR SKILLS

➤ TEST



## Позиціонування - fixed

Іноді потрібно позиціонувати елемент не відносно батька, а щодо вікна браузера незалежно від вкладеності. Для вирішення такого завдання підходить **position: fixed**.



# Позиціонування - fixed

## Особливості:

- Елемент забирається з основного потоку документа. Тобто, перестає впливати на становище навколишніх елементів і розмір батька. Можна уявити, що елемент йде на шар вище і перестає взаємодіяти з усіма елементами, крім нащадків.

## Позиціонування - fixed

- Елемент позиціонується щодо вікна браузера, за винятком випадків, якщо один із батьків має значення властивостей **transform**, **perspective** або **filter**, відмінні від **none**. В цьому випадку блок стає опорним, і позиціонування буде проводитись вже щодо нього, а не вікна браузера.



## Позиціонування - fixed

- Елемент позиціонується щодо вікна браузера, за винятком випадків, якщо один із батьків має значення властивостей **transform**, **perspective** або **filter**, відмінні від **none**. В цьому випадку блок стає опорним, і позиціонування буде проводитись вже щодо нього, а не вікна браузера.

Example

# TEST YOUR SKILLS

## ➤ TEST - FIXED



## Позиціонування - sticky

- Елемент позиціонується в нормальному потоці так само, як і статичний, а потім зміщується за допомогою властивостей **left, right, top, bottom** щодо найближчого батька, який має прокручування. Ця властивість застосовується, коли нам потрібно зафіксувати якийсь елемент не відразу, а при прокручуванні батька до якогось відомого положення.

## Позиціонування - sticky

- Елемент позиціонується в нормальному потоці так само, як і статичний, а потім зміщується за допомогою властивостей **left, right, top, bottom** щодо найближчого батька, який має прокручування. Ця властивість застосовується, коли нам потрібно зафіксувати якийсь елемент не відразу, а при прокручуванні батька до якогось відомого положення.



# Позиціонування - sticky

## Особливості:

- Елемент веде себе як елемент з відносним (relative) позиціонуванням доти, доки його батько не буде прокручений до певної межі. Як правило, ця точка збігається зі становищем верхньої межі нашого елемента, але може бути змінена за допомогою властивості top.

# Позиціонування - sticky

## Особливості:

- Елемент залишається «приклеєним» під час прокручування батька доти, доки не «зустрітить» протилежний кордон свого батька.

Example

## Контекст накладання (stacking context)

Контекст накладання (stacking context) це концепція тривимірного розташування HTML-елементів вздовж осі Z стосовно користувача, що знаходиться перед екраном. HTML-елементи займають це місце по порядку, що базується на атрибутах елемента.



## Контекст накладання (stacking context)

Ми можемо не підкорятися стандартним правилам браузера і встановлювати порядок накладання елементів один на одного. Для цього знадобиться якість **z-index**.




## Контекст накладання (stacking context)

Ми можемо не підкорятися стандартним правилам браузера і встановлювати порядок накладання елементів один на одного. Для цього знадобиться властивість **z-index**.



# Контекст накладання (stacking context)

```
.selector {  
  z-index: auto;  
  z-index: 0;  
  z-index: -1;  
  z-index: 1;  
  z-index: 9999999;  
  z-index: -999999;  
}
```



# Контекст накладання (stacking context)

Властивість **z-index** відповідає за:


- Порядок накладання в контексті накладання.
- Можливість створення локального контексту накладання.
- Можемо застосовувати лише для позиційованого елемента( в якого **position** **не static**)

Example from MDN

# Контекст накладання (stacking context)

Найбільш базовий контекст накладання, що існує на будь-якій сторінці, формується кореневим елементом `<html>`.

Всі елементи всередині цього контексту сортуються та розташовуються з оглядкою один на одного. Але ми можемо створювати контексти накладання не тільки на сторінці, а й у кожному окремому блоці. Тоді вкладені в нього дочірні блоки сортуватимуться і розташовуватимуться вже за правилами цього нового, батьківського контексту накладання.

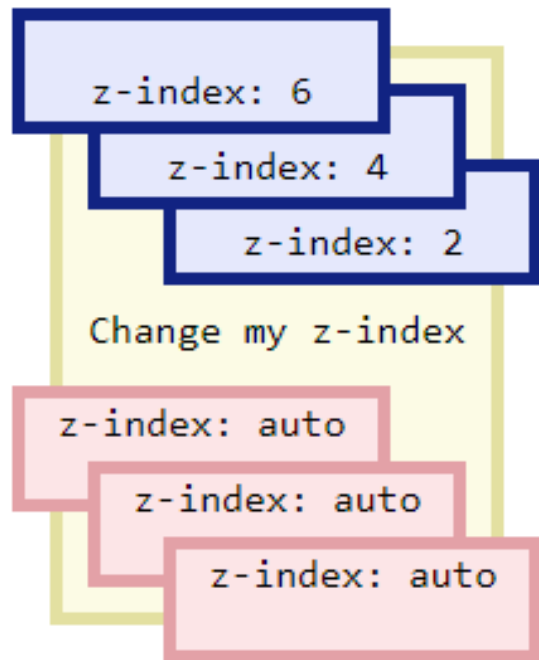




# Контекст накладання (stacking context)

## Різні колоди карт


Варто запам'ятати одну важливу річ, пов'язану з контекстом накладання: елементи можуть сортуватися, перебуваючи на одному рівні всередині контексту накладання. Елементи всередині блоку ніколи не можуть бути вищими або нижчими за свого батька.



# Властивість display

За замовчуванням всі елементи HTML бувають **блоковими і вбудованими**.

Але в верстці часто буває, що нам потрібно зробити елемент не вбудованим , а блоковим. І навпаки. Тут на допомогу приходить властивість **display**



## Властивість display

Крім значень **block** (блочне відображення) та **inline** (рядкове відображення) існує змішане значення **inline-block** (рядково-блочне відображення).

Бувають інші специфічні значення, наприклад, **flex, grid, table-cell**.



## Властивість display

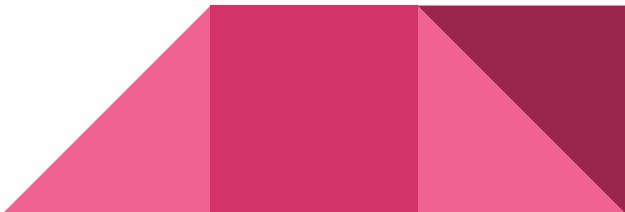
Крім значень **block** (блочне відображення) та **inline** (рядкове відображення) існує змішане значення **inline-block** (рядково-блочне відображення).

Бувають інші специфічні значення, наприклад, **flex, grid, table-cell**.



# Властивість display

## Властивості:

- **none** - повністю приховує елемент зі сторінки, не видаляючи його з HTML-розмітки.
  - **block** - елемент поводитьься як блоковий.
  - **inline-block** - елемент поводитьься зовні як малий, а всередині як блоковий.
- 

# Властивість display

## Властивості:

- **flex** - елемент стає флекс-контейнером, веде себе як блоковий, а вкладені елементи стають флекс-елементами. Докладніше в Гайді з flexbox.
- **grid** – елемент стає грід-контейнером. Зовні грід-контейнер поводитьься як блок. Дочірні елементи такого контейнера починають підкорятися правилам грід-розкладки. Докладніше в Гайді по grid.

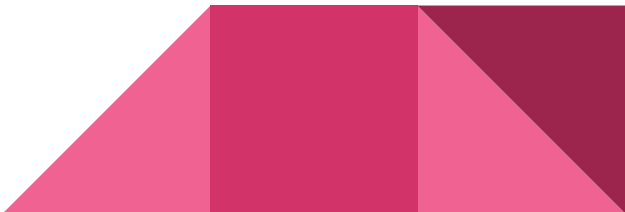
## Властивість display

Під час виведення іконок соцмереж фоном для посилань. Посилання спочатку малі. Для зміни розміру посилання та для можливості задати їй фон посилання потрібно зробити **блоковий** (`display: block`) або **рядково-блоковий** (`display: inline-block`).



## Властивість display

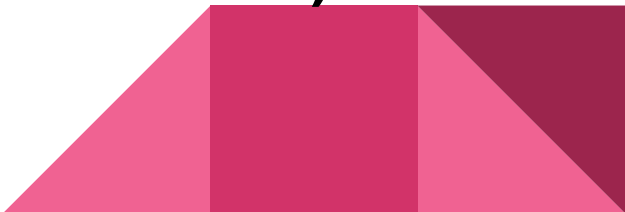
Під час виведення іконок соцмереж фоном для посилань. Посилання спочатку малі. Для зміни розміру посилання та для можливості задати їй фон посилання потрібно зробити **блоковий (display: block)** або **рядково-блоковий (display: inline-block)**.





## Властивість overflow

Властивість **overflow** дозволяє буквально визначити, що робити з вмістом блокового (тобто для елемента, у якого display визначається як block, inline-block, flex або grid) елемента, якщо воно не влізєть у розміри - відобразити або обрізати (зі смугами прокручування або без ).



# Властивість overflow

```
.selector {  
  overflow: visible;  
  overflow: hidden;  
  overflow: clip;  
  overflow: scroll;  
  overflow: auto;  
}
```



# Властивість overflow

```
.selector {  
  overflow: visible;  
  overflow: hidden;  
  overflow: clip;  
  overflow: scroll;  
  overflow: auto;  
}
```

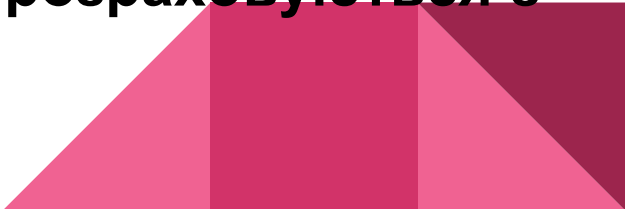


## Властивість overflow

Синтаксис властивості дозволяє задати два значення - для x-і y-осі (аналогічного результату можна досягти, використовуючи самостійні CSS-властивості **overflow-x** і **overflow-y**). Якщо вказано одне значення, воно рівнозначно застосовується обох осей.



## Властивість overflow

- visible** — вміст відображається зовні батьківського блоку, якщо його розміри більші за розміри «батьківського» блоку (значення за замовчуванням);
  - hidden** - вміст, що виходить за межі "батьківського" блоку, обрізається по його межах без прокручування;
  - clip** - нове значення (підтримується всіма основними браузерами, крім Safari, схоже за результатом на hidden, але межі батьківського блоку розраховуються з урахуванням внутрішніх відступів;
- 

## Властивість overflow

- scroll** — контент обрізається по границям «родительського» блока, но внутри этой области содержимое доступно с помощью прокрутки;
- auto** — если содержимое переполняет блок, контент будет доступен для прокрутки.



## Властивість overflow

- scroll** — контент обрізається по границям «родительського» блока, но внутри этой области содержимое доступно с помощью прокрутки;
- auto** — если содержимое переполняет блок, контент будет доступен для прокрутки.



**Дякую за увагу!**

