ЛЕКЦІЯ 2

CSS: оформлення, макети та анімації

Продовжуємо портфоліо: від розмітки до адаптивної сторінки

Дисципліна «Веб-дизайн»
Викладач: Голенко Максим Юрійович

Що навчимося робити

01 **Завершимо HTML**

Таблиці, форми та перевірка полів

02 **Оформимо сторінку**

Селектори, каскад, колір і блокова модель

03 **Розташуємо елементи**

Flexbox, Grid та адаптивність

04 **Додамо рух**

transition, @keyframes і зменшення руху

Місце CSS у нашому проєкті

01 **Лекція 1: HTML**

Структура документа, семантичні елементи та DOM

02 **Лекція 2: CSS**

Оформлення, розміщення, адаптивність і рух

03 **Лекція 3: JavaScript**

Події, зміна DOM і взаємодія з користувачем

04 **Один проєкт: портфоліо**

Сьогодні завершуємо HTML і додаємо CSS до тієї самої сторінки

Знайома структура — нове оформлення

```
<header>

<nav>

<main>
  <section>
    <article>
  <aside>

<footer>
```

header — назва й навігація
main — зміст портфоліо
section — розділи
article — картка проєкту
aside — за потреби
footer — завершення

CSS оформлює ці елементи, а DOM зберігає їхні зв’язки.

Таблиці для табличних даних

HTML

```
<table>
  <caption>Мої навички</caption>
  <tr>
    <th scope="col">Технологія</th>
    <th scope="col">Результат</th>
  </tr>
  <tr>
    <th scope="row">HTML</th>
    <td>Структура</td>
  </tr>
</table>
```

Таблиця навичок

Мої навички

Технологія	Результат
HTML	Структура
CSS	Оформлення

table — таблиця; tr — рядок; th — заголовок; td — комірка.

Як форма передає дані та варіанти вибору

HTML

```
<form action="/contact" method="post">
  <label><input type="checkbox"
    name="updates" value="yes">Оновлення</label>
  <fieldset><legend>Спосіб зв’язку</legend>
    <label><input type="radio" name="contact"
      value="email" checked>Пошта</label>
    <label><input type="radio" name="contact"
      value="phone">Телефон</label>
  </fieldset>
  <label for="topic">Тема</label>
  <select id="topic" name="topic">
    <option value="project">Проєкт</option>
    <option value="question">Запитання</option>
  </select>
  <button type="submit">Надіслати</button>
</form>
```

action

Адреса обробника.

method

GET — дані в адресі; POST — у тілі запиту.

name / value

Ключ і значення; спільний name групує radio.

checkbox — незалежний вибір; radio — один у групі; select містить option. /contact потребує сервера.

Вбудована перевірка полів

HTML

```html
<input type="email"
       name="email"
       required>

<textarea name="message"
          minlength="10"
          required>
</textarea>
```

required

Поле має бути заповнене.

type="email"

Перевірка формату адреси.

minlength

Мінімальна довжина тексту.

Перевірка в браузері допомагає користувачу; сервер перевіряє дані повторно.

CSS: правила оформлення

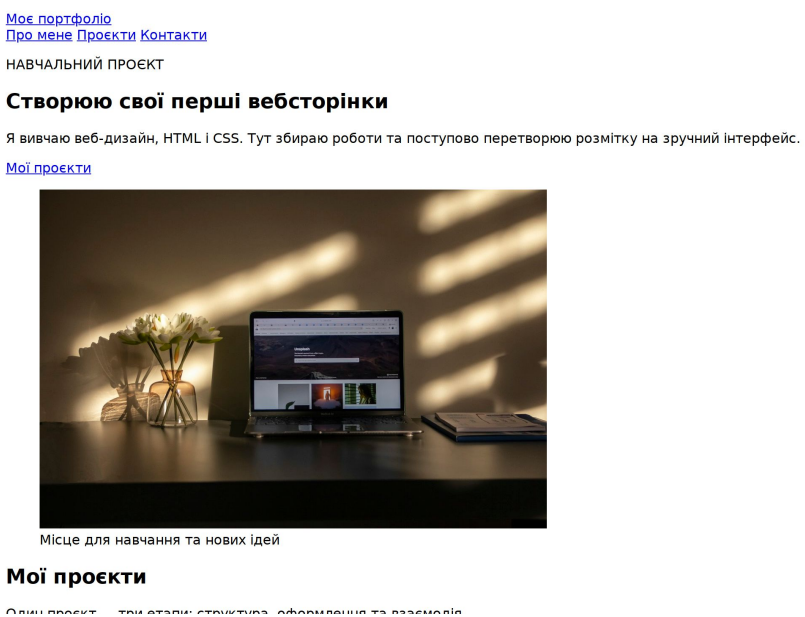
Як браузер вибирає елементи та застосовує до них стилі

Лекція 2 · CSS
Навчальний проєкт: портфоліо

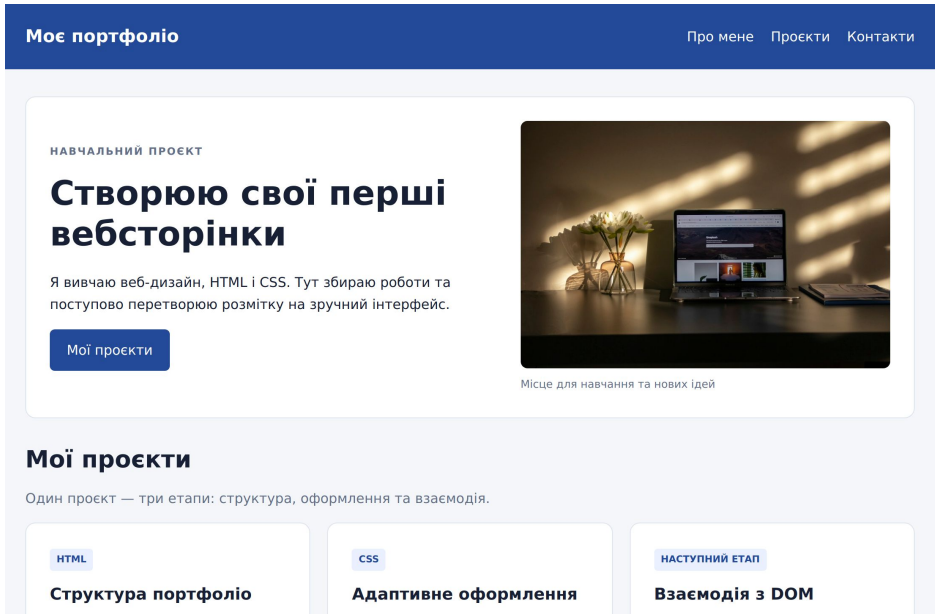
За що відповідає CSS

Та сама розмітка — інше оформлення.

БЕЗ CSS



З CSS



Колір, типографіка, відступи та розташування елементів.

Підключаємо файл стилів

HTML

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
    content="width=device-width,
             initial-scale=1.0">
  <link rel="stylesheet"
        href="css/style.css">
</head>
```

link

Підключає зовнішній ресурс.

rel="stylesheet"

Ресурс є таблицею стилів.

href

Шлях до CSS-файлу.

HTML — в index.html, стилі — у css/style.css.

Селектор, властивість і значення

```css
.card { padding: 28px; }
```

Селектор

.card — які елементи

Властивість

padding — що змінюємо

Значення

28px — як змінюємо

Декларація — властивість: значення; фігурні дужки об’єднують декларації.

Клас та ідентифікатор елемента

class

Група для оформлення

Клас може повторюватися. Один елемент може мати кілька класів: class="card hero".

id

Унікальне ім’я в документі

id="about" позначає один елемент. Його використовують у #about та посиланні href="#about".

У CSS: .card вибирає клас, #about вибирає ідентифікатор.

Що може вибирати селектор

01 **Елемент, клас або id**

h2 — тип елемента; .card — клас; #about — ідентифікатор

02 **Атрибут**

input[type="email"] — поля з відповідним значенням атрибута

03 **Зв’язок елементів**

.card h2 — нащадки; .card > h2 — лише прямі дочірні елементи

04 **Стан елемента**

:hover — наведення; :focus-visible — видимий клавіатурний фокус

Основні CSS-селектори

```css
h2 { color: #224a98; }

.card { padding: 28px; }

#about { margin-top: 32px; }
```

h2

Усі елементи h2.

.card

Елементи з класом card.

#about

Елемент з id="about".

Для повторюваного оформлення зручно використовувати класи.

Класи та вкладені елементи

HTML

```html
<section id="about"
         class="card hero">
  <h2>Про мене</h2>
  <p>Вивчаю CSS.</p>
</section>
```

CSS

```css
.card { padding: 28px; }

.hero { display: grid; }

.card h2 {
  color: #224a98;
}
```

Один елемент може мати кілька класів.

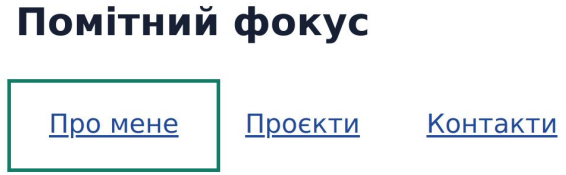
.card h2 вибирає заголовки h2 усередині .card.

Стани посилань і клавіатурний фокус

CSS

```css
a:hover {
  background: #eaf0ff;
}

a:focus-visible {
  outline: 3px solid #147d67;
  outline-offset: 4px;
}
```



hover — наведення; focus-visible — помітний клавіатурний фокус.

Каскад: як браузер обирає правило

01 **Які правила підходять**

Перевіряє селектор та умови на зразок @media

02 **Пріоритет декларацій**

Враховує джерело, !important і каскадні шари

03 **Специфічність селектора**

Порівнює вагу правил з однаковим пріоритетом

04 **Порядок у коді**

За решти однакових умов виграє пізніше правило

Чому правило перекриває інше

```css
h2 { color: #224a98; }

h2 { color: #147d67; }
```

Друге правило застосувалося

Про мене

Однакова специфічність → вирішує порядок.

Для цих звичайних правил специфічність однакова: пізніше правило перемагає.

Вага селектора: специфічність

01 **ID: перша складова**

Кожен #id додає 1 у першу позицію: #about має вагу (1, 0, 0)

02 **Класи: друга складова**

Клас, атрибут і звичайний псевдоклас додають (0, 1, 0)

03 **Типи: третя складова**

Назва елемента або псевдоелемент додає (0, 0, 1)

04 **Порівняння зліва направо**

(1, 0, 0) вище за (0, 3, 2). Це три складові, а не сума балів

Як порахувати вагу селектора

ОДИН ID + ОДИН ТИП

```
#about h2 {
  color: #224a98;
}

/* вага: (1, 0, 1) */
```

ДВА КЛАСИ

```
.card .section-title {
  color: #147d67;
}

/* вага: (0, 2, 0) */
```

За однакового пріоритету перемагає (1, 0, 1).

Спочатку порівнюємо ID. До класів переходимо лише за рівності.

Специфічність на простому прикладі

CSS

```css
h2 { color: #224a98; }

.section-title {
  color: #147d67;
}
```

Клас вибирає точніше

Про мене

Селектор класу має вищу специфічність.

У цьому прикладі клас має вищу специфічність за селектор елемента.

Успадкування стилів

CSS

```css
.card {
  color: #147d67;
  font-family: Arial;
}
```

Спільні стилі від батьківського елемента

Про мене

Цей текст також зелений.

Текст успадковує color та font-family. padding і border зазвичай не успадковуються.

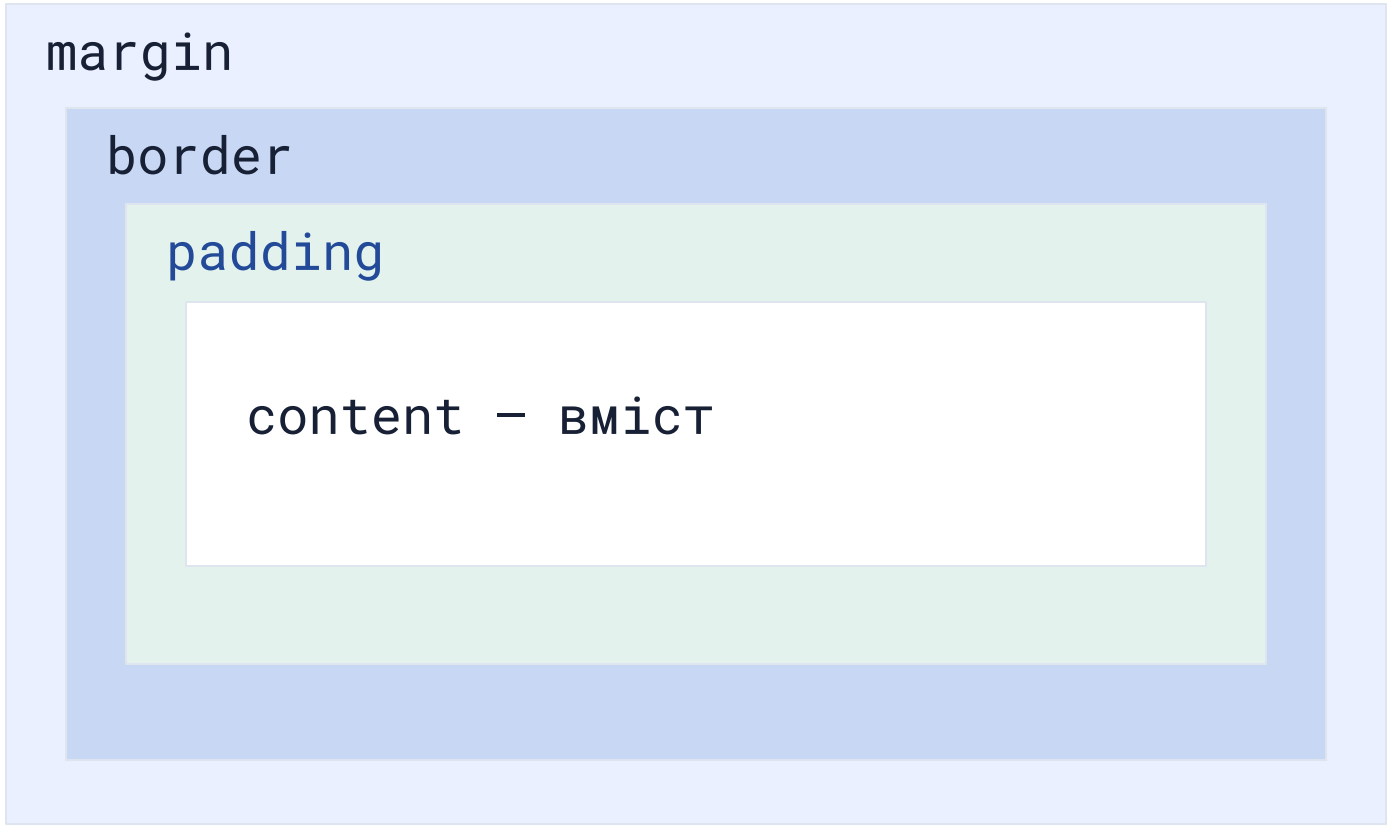
БЛОК 2 · ВИГЛЯД ЕЛЕМЕНТА

Розміри, відступи та оформлення

Блокова модель, типографіка, кольори й одиниці вимірювання

Лекція 2 · CSS
Навчальний проєкт: портфоліо

Блокова модель елемента



content — текст і медіа

padding — внутрішній відступ

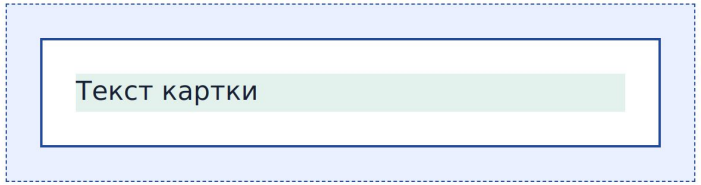
border — рамка

margin — зовнішній відступ

Внутрішні та зовнішні відступи

CSS

```css
.card {
  padding: 28px;
  margin: 28px;
  border: 2px solid #224a98;
}
```



padding — між вмістом і рамкою; margin — за рамкою елемента.

Розміри та box-sizing

CONTENT-BOX

```
.card {
  width: 300px;
  padding: 20px;
  border: 2px solid;
  box-sizing: content-box;
}
```

BORDER-BOX

```
.card {
  width: 300px;
  padding: 20px;
  border: 2px solid;
  box-sizing: border-box;
}
```

Зовнішня ширина рамки: 344px або 300px.

За border-box вміст має 256px. margin не входить до width.

Типографіка вебсторінки

CSS

```css
body {
  font-family: Arial, sans-serif;
  font-size: 1rem;
  line-height: 1.6;
}

h2 { font-size: 1.75rem; }
```

Читабельний текст

Про мене

Я вивчаю веб-дизайн, HTML і CSS. Відступи та міжрядковий інтервал допомагають читати текст.

Розмір, міжрядковий інтервал і довжина рядка визначають читабельність.

Колір, фон і контраст

```css
body {
  color: #172034;
  background: #f4f6fa;
}

.site-header {
  color: white;
  background: #224a98;
}
```

color

Колір тексту.

background

Фон елемента.

Контраст

Текст має читатися на фоні.

Колір підтримує ієрархію; важливий стан не позначаємо лише кольором.

Одиниці вимірювання

```css
h2 { font-size: 1.75rem; }

.card { padding: 28px; }

.container {
  width: 100%;
  max-width: 1100px;
}
```

px

CSS-піксель.

rem

Від розміру шрифту кореня.

%

База залежить від властивості.

Для width відсотки зазвичай рахуються від ширини блока-контейнера.

БЛОК 3 · РОЗТАШУВАННЯ

Потік документа та позиціювання

Як елементи займають місце і від чого залежить їхня позиція

Лекція 2 · CSS
Навчальний проєкт: портфоліо

Потік документа і display

```css
.block { display: block; }

.inline { display: inline; }

.hidden { display: none; }
```

block та inline

Блок 1 займає доступну ширину

Блок 2 починає новий рядок

HTML CSS JavaScript

display: none приховує елемент разом із його місцем у макеті.

Основні значення position

01 **static**

Звичайний потік. top, right, bottom і left не зміщують елемент

02 **relative**

Місце в потоці зберігається; можливий зсув від початкової позиції

03 **absolute**

Виходить із потоку; позиція залежить від блока-контейнера

04 **fixed та sticky**

fixed зазвичай прив’язаний до вікна; sticky утримується під час прокручування

Відносне й абсолютне позиціювання

relative

Від власного місця

Зсуває елемент від його позиції в потоці. Початкове місце залишається зайнятим.

absolute

Від блока-контейнера

Не займає місця в потоці. У простому випадку прив’язаний до найближчого предка з position ≠ static.

position: relative на картці задає основу для її абсолютної позначки.

Зміщення: top, right, bottom, left

01 **top та bottom**

Вертикальні зміщення або відстані від відповідного краю

02 **left та right**

Горизонтальні зміщення або відстані від відповідного краю

03 **Для relative**

top: 20px зсуває вниз, left: 20px зсуває вправо від власного місця

04 **Для absolute**

top: 20px; right: 20px задає відстані від країв блока-контейнера

Позиціювання усередині картки

```css
.card { position: relative; }

.tag {
  position: absolute;
  top: 16px;
  right: 16px;
}
```

absolute усередині relative

CSS

Мій проєкт

Позначка прив’язана до картки.

Позначку прив’язано до картки. absolute виводить елемент зі звичайного потоку.

Переповнення й перекриття елементів

overflow

Вміст за межами блока

visible — показати; hidden — обрізати; auto — прокручувати за потреби.

z-index

Порядок перекриття

Для позиційованих елементів: більше z-index — вище в одному контексті накладання.

```
.panel { max-height: 12rem; overflow: auto; }
.tag { position: absolute; z-index: 2; }
```

Позиціювання та transform

position

Модель розміщення

Визначає участь у потоці та основу для top, right, bottom і left.

transform

Візуальне перетворення

Зсуває, повертає або масштабує елемент. Початкове місце для сусідів зберігається.

transform зручний для невеликого руху, центрування й анімацій.

Основні функції transform

01 **translate(x, y)**

Зсув. translateX(20px) — горизонтально; translateY(20px) — вертикально

02 **scale(x, y)**

Масштаб. scale(1.1) збільшує в обох напрямках у 1,1 раза

03 **rotate(angle)**

Поворот. rotate(15deg) повертає елемент на 15 градусів

04 **skew(x, y)**

Нахил. skewX(10deg) змінює геометрію вздовж горизонтальної осі

Центрування через translate()

CSS

```css
.card { position: relative; }

.label {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

50% у top/left та 50% у translate мають різні бази обчислення.

top / left: 50%

Початок позначки потрапляє в центр картки.

translate(-50%, -50%)

Зсув на половину власної ширини й висоти.

Результат

Центр позначки збігається з центром картки.

Матриця 2D-перетворення: matrix()

CSS

```css
/* matrix(a, b, c, d, tx, ty) */
.box {
  transform:
    matrix(1, 0, 0, 1, 40, 20);
}

/* такий самий зсув */
.box {
  transform: translate(40px, 20px);
}
```

a, b, c, d

Коефіцієнти лінійного перетворення.

tx, ty

Зсув по X та Y у CSS-пікселях. Числа без px.

Координати

$x' = a \cdot x + c \cdot y + tx$
$y' = b \cdot x + d \cdot y + ty$

Координати відносні до transform-origin. matrix() — додатковий матеріал.

Flexbox: гнучке компонування

Контейнер керує осями, а елементи ділять доступний простір

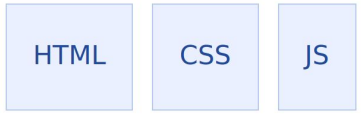
Лекція 2 · CSS
Навчальний проєкт: портфоліо

Flexbox: контейнер та елементи

CSS

```css
.items {
  display: flex;
  gap: 16px;
}
```

Один рядок, три елементи



Безпосередні дочірні елементи .items стають flex-елементами.

Головна і поперечна осі Flexbox

```
flex-direction: row
```



flex-direction
задає головну вісь.

justify-content
вирівнює вздовж неї.

align-items
вирівнює поперек.

За column осі змінюються.

Властивості flex-контейнера

01 **flex-direction та flex-wrap**

row / column задають напрямок; nowrap / wrap — перенесення

02 **justify-content**

Розподіл вільного місця вздовж головної осі: center, space-between

03 **align-items та align-content**

Елементи на поперечній осі; align-content — розподіл кількох рядків

04 **gap, row-gap, column-gap**

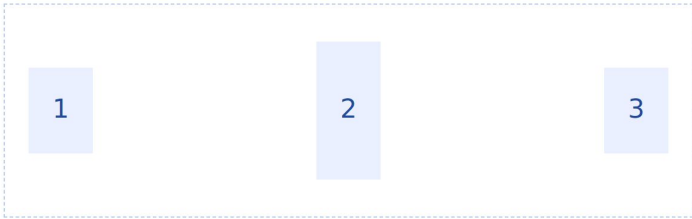
Проміжки між елементами та рядками без додаткових margin

Напрямок і вирівнювання

CSS

```css
.items {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
  gap: 16px;
}
```

justify-content і align-items



Для переносу рядків додаємо flex-wrap: wrap. Вісь залежить від flex-direction.

Властивості flex-елемента

01 **flex-basis**

Початковий розмір уздовж головної осі: auto або довжина

02 **flex-grow та flex-shrink**

Коефіцієнти зростання та звуження за надлишку або нестачі місця

03 **flex**

Скорочення grow, shrink, basis. Наприклад, flex: 1 1 200px

04 **align-self та order**

Власне поперечне вирівнювання та візуальний порядок елемента

Навігація нашого портфоліо

CSS

```css
.header-inner {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  align-items: center;
  gap: 20px;
}
.navigation { display: flex; }
```

Два контейнери

Назва + навігація у .header-inner.

Посилання — у .navigation.



Назва та навігація — flex-елементи; самі посилання також у flex-контейнері.

БЛОК 5 · GRID

Grid: двовимірна сітка

Колонки, рядки, лінії та області для узгодженого макета

Лекція 2 · CSS
Навчальний проєкт: портфоліо

Grid: колонки та рядки

CSS

```css
.projects {
  display: grid;
  grid-template-columns:
    repeat(3, minmax(0, 1fr));
  gap: 16px;
}
```



Grid допомагає узгодити колонки й рядки; 1fr — частка доступного простору.

Властивості grid-контейнера

01 **grid-template-columns / rows**

Кількість і розміри колонок та рядків: px, %, fr, minmax()

02 **gap, row-gap, column-gap**

Проміжки між треками сітки

03 **grid-template-areas**

Імена областей і схема розміщення на макеті

04 **justify-items та align-items**

Вирівнювання елементів усередині їхніх областей

Розбираємо сітку карток

```css
.projects {
  display: grid;
  grid-template-columns:
    repeat(3, minmax(0, 1fr));
  gap: 20px;
}
```

repeat(3, …)

Три колонки.

minmax(0, 1fr)

Колонки можуть звужуватися.

gap

Відстань між треками.

За однакових 1fr три колонки ділять доступний простір порівну.

Властивості grid-елемента

01 **grid-column**

Розміщення між лініями колонок: 1 / 3 займає дві колонки

02 **grid-row**

Розміщення між лініями рядків; span 2 охоплює два треки

03 **grid-area**

Прив’язка до іменованої області, наприклад grid-area: main

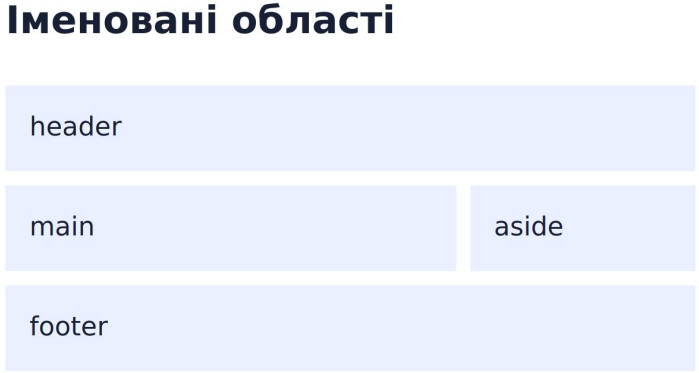
04 **justify-self та align-self**

Власне вирівнювання елемента всередині його області

Іменовані області Grid

CSS

```css
.page {
  display: grid;
  grid-template-columns: 2fr 1fr;
  grid-template-areas:
    "header header"
    "main   aside"
    "footer footer";
}
main { grid-area: main; }
```



Кожному елементу призначаємо grid-area. HTML залишається семантичним.

Коли обрати Flexbox, а коли Grid

Flexbox

Одна головна вісь

Навігація, кнопки, вирівнювання елементів у рядку або колонці.

Grid

Колонки та рядки

Сітка проєктів, узгоджені колонки та іменовані області макета.

В одному портфоліо використовуємо обидва інструменти.

БЛОК 6 · АДАПТИВНІСТЬ

Макет для різних екранів

Гнучкі розміри й медіазапити зберігають зручність сторінки

Лекція 2 · CSS
Навчальний проєкт: портфоліо

Що означає адаптивність

CSS

```css
.container {
  width: 100%;
  max-width: 1100px;
  margin: 0 auto;
  padding: 0 24px;
}
```

Гнучка ширина

Контент підлаштовується.

Обмеження

На широкому екрані є межа.

Перевірка

Читання й дії зручні всюди.

Точку зміни макета обираємо за вмістом, а не за конкретною моделлю телефона.

Гнучкі зображення

```css
img {
  display: block;
  max-width: 100%;
  height: auto;
}
```

max-width

Не виходить за контейнер.

height: auto

Зберігає пропорції.

width / height

В HTML задають початкові розміри.

Зображення адаптується без розтягування або спотворення.

Один медіазапит для портфоліо

CSS

```css
.projects {
  display: grid;
  grid-template-columns: 1fr;
}
@media (min-width: 48rem) {
  .projects {
    grid-template-columns:
      repeat(3, minmax(0, 1fr));
  }
}
```

Базовий стиль

Одна колонка.

Умова @media

Від 48rem — ширший макет.

Перевірка

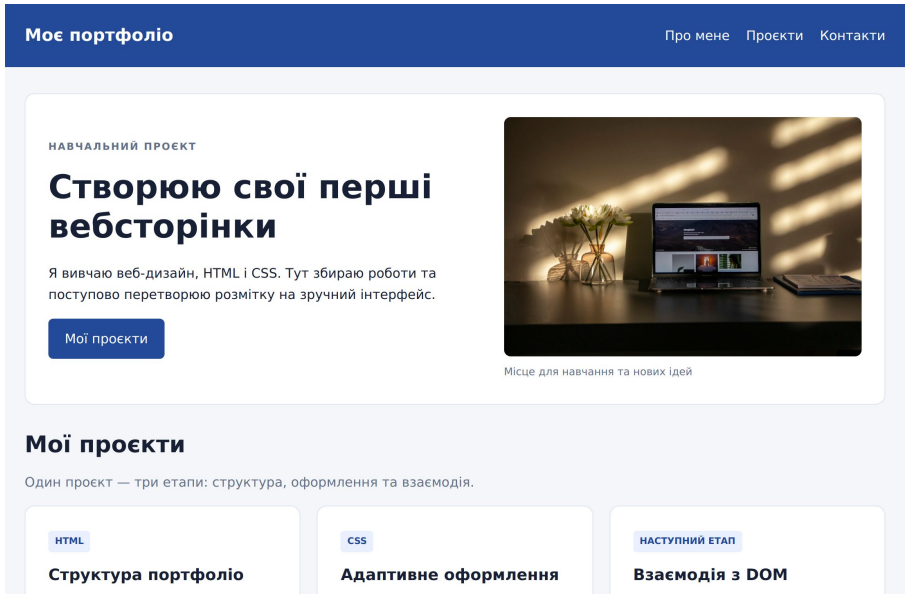
Змінюємо ширину браузера.

Починаємо з вузького екрана й додаємо колонки за наявності місця.

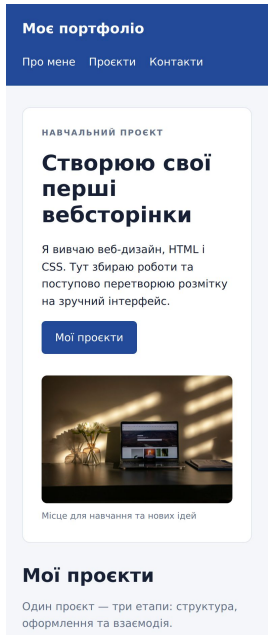
Портфоліо на різних екранах

Контент той самий; кількість колонок змінюється.

ШИРОКИЙ ЕКРАН



ВУЗЬКИЙ ЕКРАН



Сторінка перевірена на ширинах 320, 390, 768 та 1100 CSS-пікселів.

Переходи та анімації

transition змінює стан плавно; animation відтворює ключові кадри

Лекція 2 · CSS
Навчальний проєкт: портфоліо

Плавна зміна стану: transition

CSS

```css
.project-card {
  transition: transform 200ms ease;
}
.project-card:hover {
  transform: translateY(-4px);
}
```

Наведіть курсор на картку

Адаптивне оформлення

Плавна зміна від одного стану до іншого.

transition згладжує зміну властивості між двома станами.

Ключові кадри: @keyframes

КАДРИ

```
@keyframes reveal {
  from {
    opacity: 0;
    transform: translateY(12px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}
```

ЗАСТОСУВАННЯ

```
.hero {
  animation: reveal
    700ms ease-out both;
}
```

Ключові кадри описують стани; animation запускає їх.

from = 0%, to = 100%. Вступний блок проявляється один раз.

Параметри CSS-анімації

```css
.hero {
  animation-name: reveal;
  animation-duration: 700ms;
  animation-timing-function:
    ease-out;
  animation-iteration-count: 1;
  animation-fill-mode: both;
}
```

name / duration

Назва кадрів і тривалість.

timing-function

Як змінюється темп.

iteration-count

Кількість повторень.

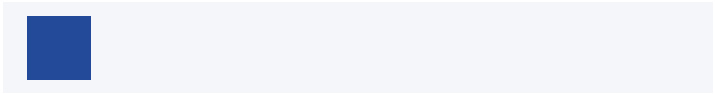
Для циклу є infinite, для зворотного руху — animation-direction: alternate.

Як браузер заповнює проміжні кадри

0%

Початковий стан

translateX(0)



ЗА LINEAR

```
transform: translateX(0);
```

50%

Проміжний стан

translateX(160px)



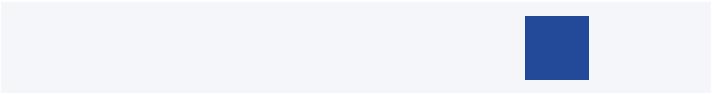
ЗА LINEAR

```
transform:
translateX(160px);
```

100%

Кінцевий стан

translateX(320px)



ЗА LINEAR

```
transform: translateX(320px);
```

Враховуємо зменшення руху

```css
@media
  (prefers-reduced-motion: reduce) {
  .hero { animation: none; }
  .project-card, .button {
    transition: none;
  }
}
```

Запит користувача

Користувач просить менше руху.

@media

Змінюємо поведінку стилів.

Результат

Контент і дії доступні без руху.

У портфоліо за цієї умови також вимикаємо зсув картки.

Як знайти помилку в CSS

01 **Перевіряємо підключення**

Правильний шлях; файл CSS завантажився

02 **Перевіряємо селектор**

Клас у HTML збігається з класом у CSS

03 **Дивимося застосовані стилі**

Вкладки Styles і Computed показують конфлікти

04 **Перевіряємо макет**

Блокова модель, Grid/Flex та ширина екрана

Практичне завдання: оформити портфоліо

01 **Оформлення**

Зовнішній CSS, кольори, типографіка та відступи

02 **Розташування**

Навігація — Flexbox; картки — Grid

03 **Адаптивність**

Гнучкі зображення та один медіазапит

04 **Перевірка і рух**

Видимий фокус, transition, reveal та reduced-motion

Від оформлення до взаємодії

Уже маємо

HTML + CSS

Структуру, оформлення, адаптивний макет і CSS-анімації.

Додамо на лекції 3

JavaScript + DOM

Події, зміну стану картки, тексту та локальний відгук форми.

Один проєкт: структура → оформлення → взаємодія.