Міністерство освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Технології підготовки мультимедійного контенту
Практикум

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Технології друкованих і електронних видань»
спеціальності 186 Видавництво та поліграфія

Укладач: О. В. Коротенко

Електронне мережеве навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2024

УДК 378:004.55
К68

Укладачі: *Коротенко Олена Володимирівна*, канд. техн. наук, доц.

Рецензент *Тріщук Р.Л.*,канд. техн. наук, доц.

Відповідальний редактор *Киричок Т.Ю.*, д-р техн. наук, проф.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 3 від 09.01.2025 р.)
за поданням вченої ради Навчально-наукового видавничо-поліграфічного інституту
(протокол № 5 від 25.12.2024 р.)

К68 Технології підготовки мультимедійного контенту : практикум [Електронний ресурс] : навч. посіб. для здобувачів ступеня бакалавра за освіт. програмою «Технології друкованих і електронних видань» спец. 186 Видавництво та поліграфія / КПІ ім. Ігоря Сікорського ; уклад.: О. В. Коротенко. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2024. – 127 с.

У навчальному посібнику «Технології підготовки мультимедійного контенту. Практикум» викладено теоретичний та практичний матеріал щодо методів та засобів підготовки, обробки та впровадження мультимедійних матеріалів у різні мережеві електронні видання. Посібник містить перелік робіт практикуму, повне виконання яких студентами сформує їх досвід проєктування, розробки, підготовки та розміщення на сторінках мережевих електронних видань текстового, графічного, анімаційного, аудіо- та відео-контенту. Навчальний посібник призначений для бакалаврів НН ВПІ КПІ ім. Ігоря Сікорського спеціальності 186 Видавництво та поліграфія.

УДК 378:004.55

Реєстр. № НП 24/25-194. Обсяг 5,25 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Берестейський, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

ПЕРЕДМОВА 4

МЕТА, ЗАВДАННЯ І ТЕМАТИКА РОБІТ ПРАКТИКУМУ 5

ОСНОВНІ ВИМОГИ ДО ВИКОНАННЯ РОБІТ ПРАКТИКУМУ..... 6

ЗМІСТ ТА ПЕРЕЛІК РОБІТ ПРАКТИКУМУ 7

КП1. Принцип побудови розмітки вебсторінки Flex. Створення розмітки односторінкового сайту..... 7

КП2. Методи позиціювання елементів на вебсторінці. Створення композиції зображень..... 21

КП 3. Візуалізація інформації. Оптимізація зображень. Створення та додавання інформаційної графіки на вебсторінку..... 29

КП 4. Методи впровадження графічного контенту у вебсторінку. Створення інтерактивного зображення..... 34

КП 5. Створення векторних ілюстрацій у форматі svg, оптимізація та вбудовування їх у вебсторінку 43

КП 6. Методи трансформації елементів на вебсторінці. Створення трансформації елементів при наведенні. 63

КП 7. Ефекти анімації та керування ними за допомогою CSS властивостей. Створення анімації появи елементів сторінки при її завантаженні 68

КП 8. Анімування векторних ілюстрацій на вебсторінці..... 76

КП 9. Підготовка та додавання аудіоконтенту на вебсторінку 89

КП 10. Підготовка та додавання відеоконтенту на вебсторінку 95

КП 11. Додавання адаптивності до вебсторінки..... 103

КП 12. Створення інтерактивного меню 117

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ..... 119

ДОДАТКИ..... 120

ПЕРЕДМОВА

Навчальний посібник «Технології підготовки мультимедійного контенту. Практикум» включає загальні поняття, термінологію, засоби розроблення мультимедійних видань різного виду. Посібник має на меті допомогти студентам оволодіти знаннями й практичними навичками у сфері підготовки, обробки та впровадження мультимедійних матеріалів у різні мережеві електронні видання. Посібник забезпечує формування у студентів здатностей розроблення мережевих електронних видань та їх наповнення підготовленим мультимедійним контентом різного виду, орієнтування у сучасних інструментах розробки та правилах підготовки мультимедійного контенту до розміщення на сторінках мережевих електронних видань.

Навчальний посібник містить комплекс робіт, спрямованих на створення розмітки структури майбутнього електронного видання; розроблення та оптимізації текстового, графічного, анімаційного, аудіо- та відео-контенту під електронні видання різного виду; адаптації мультимедійного контенту під різні пристрої та платформи; заглиблення у ньюанси css-анімації, створення різного роду ефектів; підготовки, впровадження на вебресурс та анімування svg-графіки; позиціонування та трансформації елементів вебсторінок. У результаті виконання комплексу робіт у студентів формується досвід проєктування, розробки, підготовки та розміщення на сторінках мережевих електронних видань текстового, графічного, анімаційного, аудіо- та відео-контенту.

Посібник призначено для студентів денної, інтегральної та заочної форми навчання рівня вищої освіти ступеня «бакалавр». Його можна використати також для підготовки до занять, заліків, екзаменів студентам усіх форм навчання, які вивчають подібний матеріал. Курс відповідає нагальній ринковій потребі підготовки сучасних фахівців.

МЕТА, ЗАВДАННЯ І ТЕМАТИКА РОБІТ ПРАКТИКУМУ

Мета робіт практикуму полягає в наданні знань студентам щодо сучасних технологій, засобів та інструментів розроблення мережевих електронних видань, наповнених мультимедійним контентом, та формуванні практичних навичок створення інтерактивного контенту, а також опановуванні принципів дизайну, верстки та інтеграції мультимедійних елементів з урахуванням вимог доступності та адаптивності для різних аудиторій.

Виконання усіх робіт практикуму сформує у студентів досвід проєктування, розробки, підготовки та розміщення на сторінках мережевих електронних видань текстового, графічного, анімаційного, аудіо- та відео-контенту для вирішення конкретних завдань та посприяє розвитку самостійності в аналізі та прийнятті важливих професійних рішень, які підвищать загальний технічний рівень підготовки студентів.

Для виконання робіт практикуму необхідне наступне технічне та програмне забезпечення:

- персональний комп’ютер (ноутбук);
- будь-який браузер із доступом до мережі;
- текстовий редактор на вибір
  (https://www.sublimetext.com/,https://code.visualstudio.com/);
- графічний редактор Figma (можлива браузерна версія)
  (https://figma.com).

Завдання кожної роботи практикуму є індивідуальним і базується на конкретній темі, яка вивчається. Проте всі практикуми взаємопов’язані між собою загальною ідеєю проєкту, над яким студент працює впродовж усього семестру. Результатом виконання усіх робіт практикуму має бути адаптована інтерактивна вебсторінка, наповненя різним типом мультимедійного контенту (текстом, растровою та векторною графікою, різними анімаційними ефектами, аудіо- та відеоконтентом). Це забезпечує системний підхід до вивчення матеріалу та сприяє глибшому зануренню в тему.

ОСНОВНІ ВИМОГИ ДО ВИКОНАННЯ РОБІТ ПРАКТИКУМУ

Усі роботи практикуму мають тему, мету, теоретичні відомості, хід виконання роботи, контрольні запитання.

Під час виконання робіт необхідно дотримуватися наведених нижче вимог. Роботи, виконані без дотримання цих правил, можуть бути повернені студенту для доопрацювання.

Для якісного вивчення дисципліни та засвоєння матеріалу, роботи практикуму мають виконуватися вчасно та послідовно згідно із затвердженим календарним планом. Під час виконання робіт необхідно використовувати технічне та програмне забезпечення, наведене вище. Етапи виконання кожної роботи мають відповідати пунктам, описаним у протоколі кожної роботи практикуму.

Протокол виконання комп'ютерного практикуму формується у вигляді .doc/.docx або pdf-файлу форматом А4, оформлення яких здійснюється із дотриманням вимог ДСТУ 3008-2015. Типова структура протоколу з комп'ютерного практикуму містить:

– титульний аркуш;

– аркуш завдання – містить мету, технічні і програмні засоби, які були використані для виконання роботи, умови завдання;

– основна частина – виконані завдання за пунктами;

– висновки щодо виконаної роботи;

– додатки (за необхідністю).

Студенти денної форми навчання роботи практикуму не лише виконують, а і захищають методом демонстрації виконаних завдань та наданні усних відповідей на поставлені викладачем контрольні запитання. Студенти заочної форми навчання роботи практики виконують, та письмово відповідають на усі контрольні запитання. Оцінювання робіт здійснюється згідно силабусу дисципліни.

ЗМІСТ ТА ПЕРЕЛІК РОБІТ ПРАКТИКУМУ

КП1. ПРИНЦИП ПОБУДОВИ РОЗМІТКИ ВЕБСТОРІНКИ FLEX. СТВОРЕННЯ РОЗМІТКИ ОДНОСТОРІНКОВОГО САЙТУ

Мета роботи – ознайомитися із режимом розмітки CSS3 Flexible box (Flexbox) та основними властивостями та правилами застосування.

Основні теоретичні відомості

Модуль Flexbox Layout (Flexible Box) (W3C Candidate Recommendation від жовтня 2017 р.) спрямований на забезпечення більш ефективного способу розміщення, вирівнювання та розподілу простору між елементами в контейнері, навіть якщо їх розмір невідомий та/або динамічний (Flex означає «гнучкий»).

Основна ідея flex layout полягає в тому, щоб дати контейнеру можливість змінювати ширину / висоту його елементів (і порядок), щоб якнайкраще заповнити доступний простір (головним чином, для відображення на всіх типах пристроїв з будь-яким розміром екрана). Flex контейнер розширює елементи, щоб заповнити доступний вільний простір, або стискає їх, щоб запобігти переповненню.

Оскільки flexbox – це цілий модуль, а не одна властивість, він включає безліч елементів з набором властивостей. Деякі з них призначені для встановлення в контейнері (батьківський елемент заведено називати «flex контейнер»), тоді як інші призначені для встановлення в дочірніх елементах (так звані «flex елементи»).

Якщо «звичайне» компонування засноване як на блоковому, так і на inline напрямках, flex layout заснована на «напрямках flex-flow».

Основна ідея гнучкого макета (рис. 1) полягає у тому, що елементи будуть розташовані або у напрямку головної осі (main axis від main-start до main-end) або у напрямку поперечної осі (cross axis від cross-start до cross-end).

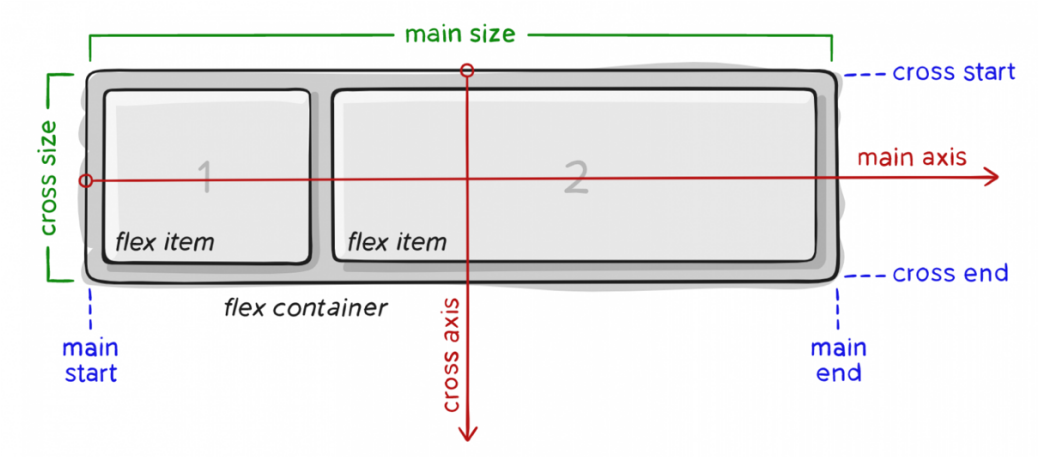


Рис. 1.1. Основна ідея гнучкого макета

Основна ідея гнучкого макета полягає у наступних складниках блоку (рис. 1.1):

- main axis – головна вісь flex контейнера – це основна вісь, вздовж якої розташовуються flex елементи. Будьте уважні, ця вісь не обов'язково горизонтальна; це залежить від flex-direction властивості (див. нижче).
- main-start | main-end – flex елементи поміщаються в контейнер, починаючи з main-start і закінчуючи main-end.
- main size – ширина чи висота flex елемента, залежно від цього, що в основному вимірі. Визначається основним розміром flex елементів, тобто. властивістю 'width' або 'height', залежно від того, що знаходиться в основному вимірі.
- cross axis – вісь перпендикулярна до головної осі, називається поперечною віссю. Її напрямок залежить від напрямку головної осі.
- cross-start | cross-end – flex рядки заповнюються елементами та поміщаються в контейнер, починаючи від cross-start flex контейнера до cross-end.
- cross size – ширина або висота flex елемента. Залежно від css властивості flex-direction, це ширина чи висота елемента. Це завжди поперечний розмір flex елементів.

Властивості батьківського елементу: display, flex-direction, flex-wrap, flex-flow, justify-content, align-items, align-content

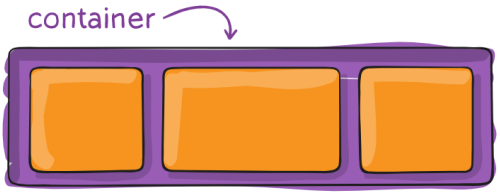


Рис. 1.2. Візуальна підказка, що є батьківським елементом (container)

Властивість **display** застосовується до батьківського елементу flex-контейнера.

Можливі значення властивості display:

```
display: other values | flex | inline-flex;
```

Властивість display: flex | inline-flex визначає flex-контейнер (інлайновий чи блоковий залежно від обраного значення), підключає flex-контекст всім його безпосереднім нащадкам.

Властивість **flex-direction** застосовується до батьківського елементу flex-контейнера.

Властивість flex-direction встановлює основну вісь, таким чином визначаючи напрямок flex елементів, що поміщаються у flex контейнер.

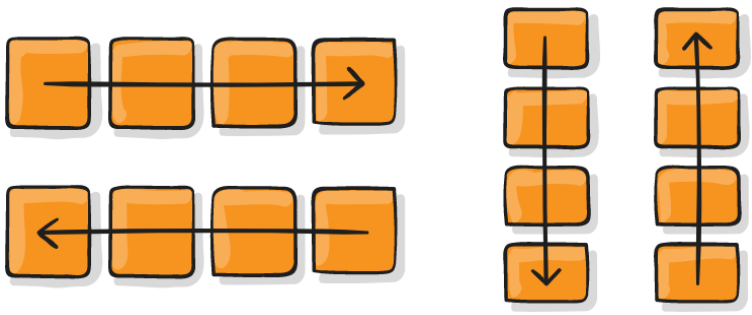


Рис. 1.3. Візуальна підказка щодо можливостей властивості flex-direction

Можливі значення властивості flex-direction:

```
flex-direction: row | row-reverse | column | column-reverse
```

- row (за замовчуванням): зліва направо для ltr, справа наліво для rtl;
- row-reverse: справа наліво ltr, зліва направо rtl;
- column: аналогічно row зверху вниз;
- column-reverse: аналогічно row-reverse, знизу догори.

Властивість **flex-wrap** застосовується до батьківського елементу flex-контейнера.

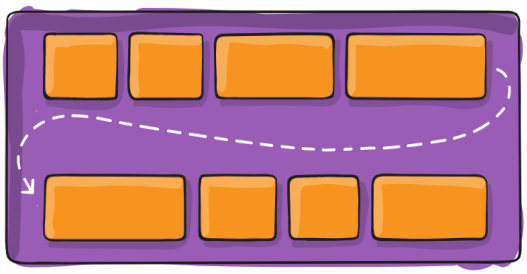


Рис. 1.4. Візуальна підказка щодо можливостей властивості flex-wrap

Властивість flex-wrap визначає, чи контейнер однорядковим або багаторядковим, а також напрямок поперечної осі, що визначає напрямок, в якому будуть розташовуватися нові рядки.

Можливі значення властивості flex-wrap:

```
flex-wrap: nowrap | wrap | wrap-reverse
```

- nowrap(за замовчуванням): однорядковий / зліва направо для ltr, праворуч наліво для rtl;
- wrap: багаторядковий / зліва направо для ltr, справа наліво для rtl;
- wrap-reverse: багаторядковий / справа наліво ltr, зліва направо rtl.

Властивість **flex-flow** застосовується до батьківського елементу flex-контейнера.

```
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

Це скорочення для flex-direction та flex-wrap властивостей, які разом визначають основні та поперечні осі flex контейнера. Значенням за замовчуванням є row nowrap.

Властивість **justify-content** застосовується до батьківського елементу flex-контейнера. Властивість justify-content визначає вирівнювання щодо головної осі. Допомагає розподілити вільне місце, що залишилося у випадку, коли елементи рядка не «тягнуться», або тягнуться, але вже досягли свого максимального розміру. Також дозволяє до певної міри керувати вирівнюванням елементів при виході за межі рядка.

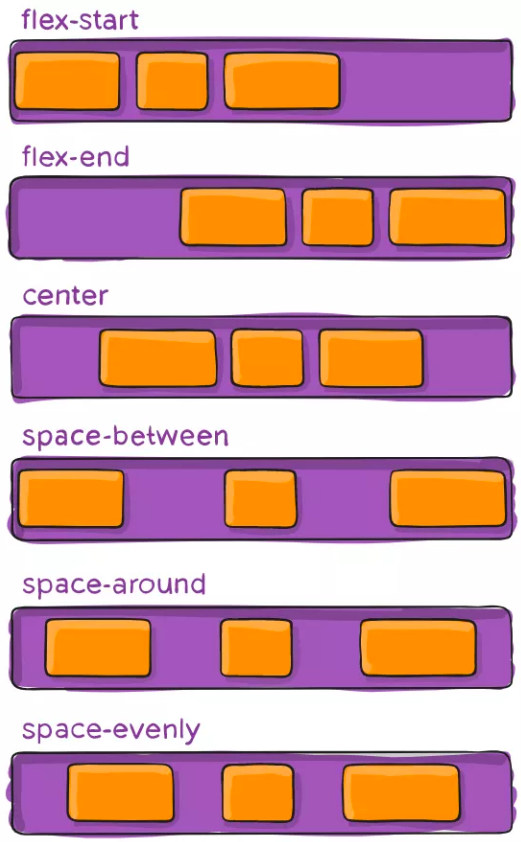


Рис. 1.5. Візуальна підказка щодо можливостей властивості justify-content

Можливі значення властивості justify-content:

```
justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right ... + safe | unsafe;
```

- flex-start(за замовчуванням): елементи зсуваються на початок рядка;
- flex-end: елементи зсуваються до кінця рядка;
- center: елементи вирівнюються по центру рядка;
- space-between: елементи розподіляються рівномірно (перший елемент на початку рядка, останній – наприкінці);
- space-around: елементи розподіляються рівномірно з рівною відстанню між собою та межами рядка;
- space-evenly : елементи розподіляються таким чином, щоб відстань між будь-якими двома елементами (і відстань до країв) була однаковою.
- start : елементи зрушені до початку writing-mode напряму.
- end : елементи зрушені в кінці writing-mode напряму.
- left : елементи зрушені до лівого краю контейнера, якщо це не стосується flex-direction, тоді він поводиться як start.
- right : елементи зрушені до правого краю контейнера, якщо це не стосується flex-direction, тоді він поводиться як start.

Властивість **align-items** застосовується до батьківського елементу flex-контейнера.

Ця властивість визначає поведінку за умовчанням того, як flex елементи розташовуються вздовж поперечної осі поточної лінії. Думайте про це як про justify-content версію для поперечної осі (перпендикулярної головної осі).

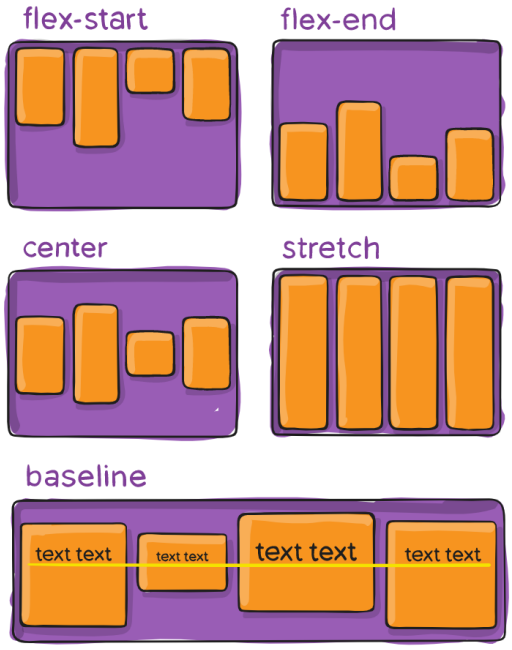


Рис. 1.6. Візуальна підказка щодо можливостей властивості align-items

Можливі значення властивості align-items:

```
align-items: stretch | flex-start | flex-end | center |
baseline | first baseline | last baseline | start | end |
self-start | self-end + ... safe | unsafe;
```

- stretch (за замовчуванням): розтягувати, щоб заповнити контейнер (досі дотримуються min-width / max-width)
- flex-start / start / self-start : елементи розміщуються на початку поперечної осі. Різниця між ними невелика і полягає у дотриманні flex-direction правил або writing-mode правил.
- flex-end / end / self-end : елементи розташовуються в кінці поперечної осі. Різниця знову-таки тонка і полягає у дотриманні flex-direction або writing-mode правил.
- center : елементи центровані по поперечній осі
- baseline : елементи вирівняні, з їхньої базової лінії

- safe і unsafe ключові слова модифікаторів можуть бути використані в поєднанні з усіма з цих ключових слів (хоча це підтримується не всіма браузерами), це допомагає запобігти вирівнюванню елементів таким чином, що зміст стає недоступним.

Властивість **align-content** застосовується до батьківського елементу flex-контейнера.

Ця властивість вирівнює лінії в межах flex контейнера, коли є додатковий простір на поперечній осі, подібно до того, як **justify-content** вирівнює окремі елементи в межах головної осі.

Примітка: ця властивість не діє, коли є лише один рядок flex елементів.

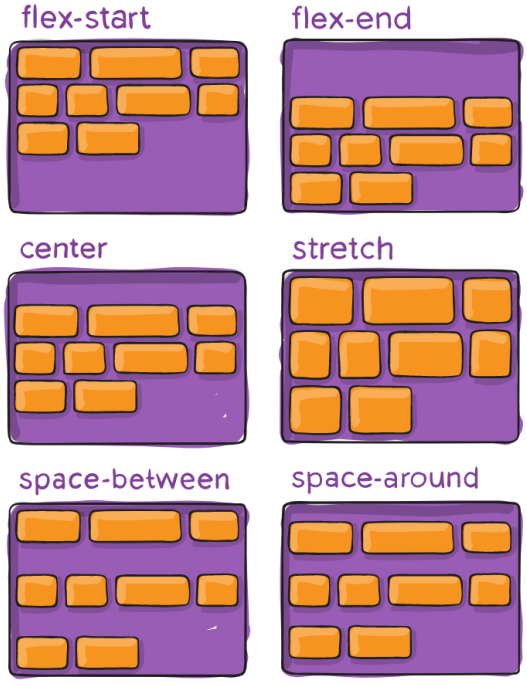


Рис. 1.7. Візуальна підказка щодо можливостей властивості align-content

Можливі значення властивості align-content:

```
align-content: flex-start | flex-end | center | space-between
| space-around | space-evenly | stretch | start | end |
baseline | first baseline | last baseline + ... safe |
unsafe;
```

- flex-start / start : елементи, зрушені на початок контейнера. Більш підтримуваний flex-start використовує, flex-direction в той час, як start використовує writing-mode напрямок.
- flex-end / end : елементи, зрушені в кінець контейнера. Більш підтримуваний flex-end використовує flex-direction в той час, як end використовує writing-mode напрямок.
- center : елементи вирівняні центром у контейнері
- space-between : елементи рівномірно розподілені; перший рядок знаходиться на початку контейнера, а останній – наприкінці
- space-around : елементи рівномірно розподілені з рівним простором навколо кожного рядка
- space-evenly : елементи рівномірно розподілені, навколо них однаковий простір
- stretch (за замовчуванням): лінії розтягуються, щоб зайняти простір, що залишився
- safe та unsafe ключові слова модифікаторів можуть бути використані у поєднанні з усіма з цих ключових слів (хоча це підтримується не всіма браузерами), це допомагає запобігти вирівнюванню елементів таким чином, що зміст стає недоступним.

Властивості для дочірніх елементів: order, flex-grow, flex-shrink, flex-basis, flex, align-self.

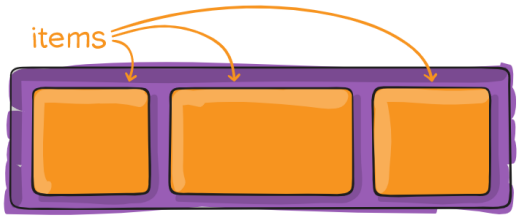


Рис. 1.8. Візуальна підказка, що є дочірніми елементами (items)

Властивість **order** застосовується до дочірнього елементу flex-контейнера.

Можливі значення властивості order:

```
order: <integer>/* default 0 */
```

За замовчуванням flex елементи розміщуються у початковому порядку. Однак властивість order управляє порядком їх появи у контейнері flex.

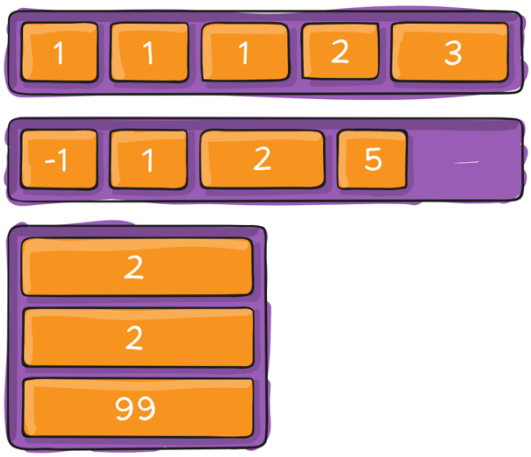


Рис. 1.9. Візуальна підказка щодо можливостей властивості order

Властивість **flex-grow** застосовується до дочірнього елементу flex-контейнера.

Можливі значення властивості flex-grow:

```
flex-grow: <number>; /* default 0 */
```

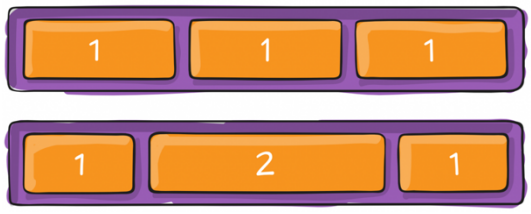


Рис. 1.10. Візуальна підказка щодо можливостей властивості flex-grow

Ця властивість визначає здатність flex елемента розтягуватися у разі потреби. Воно набуває значення від нуля, яке є пропорцією. Це властивість,

скільки доступного простору всередині гнучкого контейнера повинен займати елемент. Негативні числа не підтримуються.

Якщо для всіх елементів **flex-grow** встановлено значення 1, простір, що залишився в контейнері, буде рівномірно розподілено між усіма дочірніми елементами. Якщо один із дочірніх елементів має значення 2, цей елемент займе вдвічі більше місця, ніж інші (або спробує принаймні).

Властивість **flex-shrink** застосовується до дочірнього елементу flex-контейнера. Ця властивість визначає здатність гнучкого елемента стискатися за необхідності. Негативні числа не підтримуються.

Можливі значення властивості flex-shrink:

```
flex-shrink: <number>; /* default 1 */
```

Властивість **flex-basis** застосовується до дочірнього елементу flex-контейнера.

Можливі значення властивості flex-basis:

```
flex-basis: <length> | auto; /* default auto */
```

Ця властивість визначає розмір елемента за умовчанням перед розподілом простору, що залишився. Це може бути довжина (наприклад, 20%, 5rem і т.д.) або ключове слово. Ключове слово auto означає «дивися на мою width або height властивість». Ключове слово content означає «розмір на основі вмісту елемента» – це ключове слово все ще не дуже добре підтримується, так що важко перевірити, що для нього використовується max-content, min-content або fit-content. Якщо встановлено значення 0, додатковий простір навколо вмісту не враховується. Якщо встановлено значення auto, додатковий простір розподіляється залежно від його flex-grow значення.

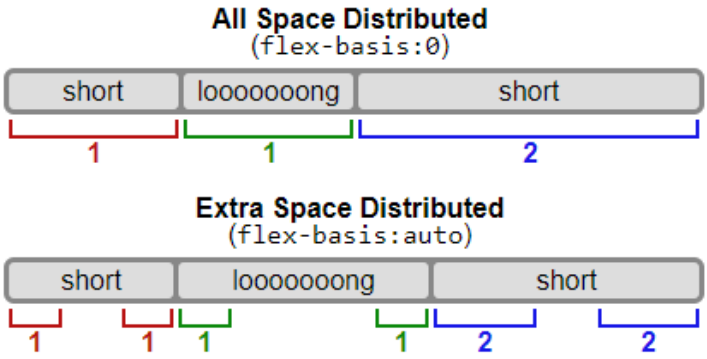


Рис. 1.11. Візуальна підказка щодо можливостей властивості flex-basis

Властивість **flex** застосовується до дочірнього елементу flex-контейнера.

Можливі значення властивості flex:

```
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
```

Це скорочення для використання flex-grow, flex-shrink та flex-basis разом. Другий і третій параметри (flex-shrink та flex-basis) є необов'язковими. За замовчуванням це 0 1 auto.

Властивість **align-self** застосовується до дочірнього елементу flex-контейнера. Ця властивість дозволяє перевизначити вирівнювання за умовчанням (або вказане за допомогою align-items) для окремих елементів flex.

Можливі значення властивості align-self:

```
align-self: auto | flex-start | flex-end | center | baseline | stretch;
```

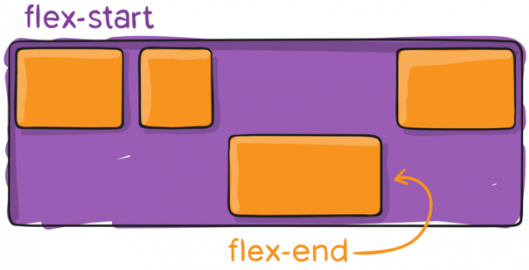
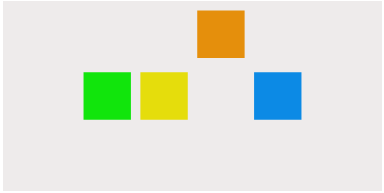
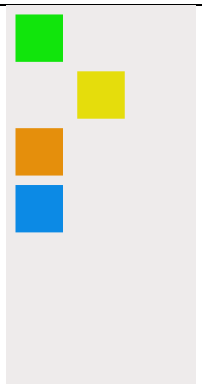
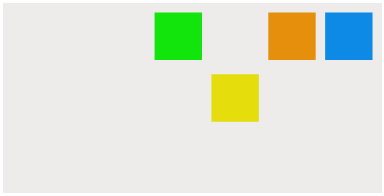
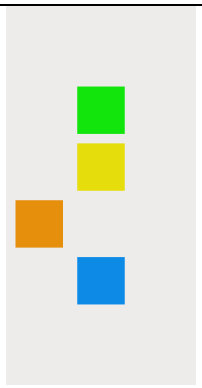
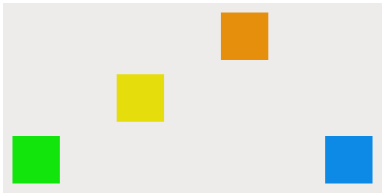
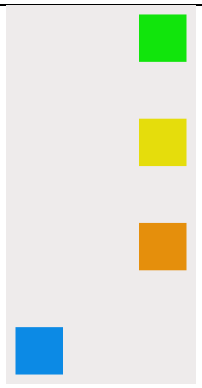


Рис. 1.12. Візуальна підказка щодо можливостей властивості align-self

Хід роботи

1. Створіть сторінку з контейнером, у який вкладіть 4 блоки (divs), задайте їм ширину та висоту та різний колір та розмістіть їх згідно з вашим варіантом* (табл. 1.1) за допомогою CSS3 Flexible box режиму розмітки.

Таблиця 1. 1. – Варіанти завдання

Номер варіанту	Завдання	Номер варіанту	Завдання
1		2	
3		4	
5		6	

**Варіант на завдання визначається згідно зі списком групи, де 1, 7, 13, 19, 25 людина за списком – виконує 1 варіант; 2, 8, 14, 20, 26 людина за списком – виконує 2 варіант; 3, 9, 15, 21, 27 людина за списком – виконує 3 варіант; 4, 10, 16, 22, 28 людина за списком – виконує 4 варіант; 5, 11, 17, 23, 29 людина за списком – виконує 5 варіант; 6, 12, 18, 24, 30 – виконує 6 варіант.*

2. Повторіть розмітку сторінки згідно прототипу сторінки (вайфрейму) (рис. 1.13), використовуючи режим розмітки CSS3 Flexible box.

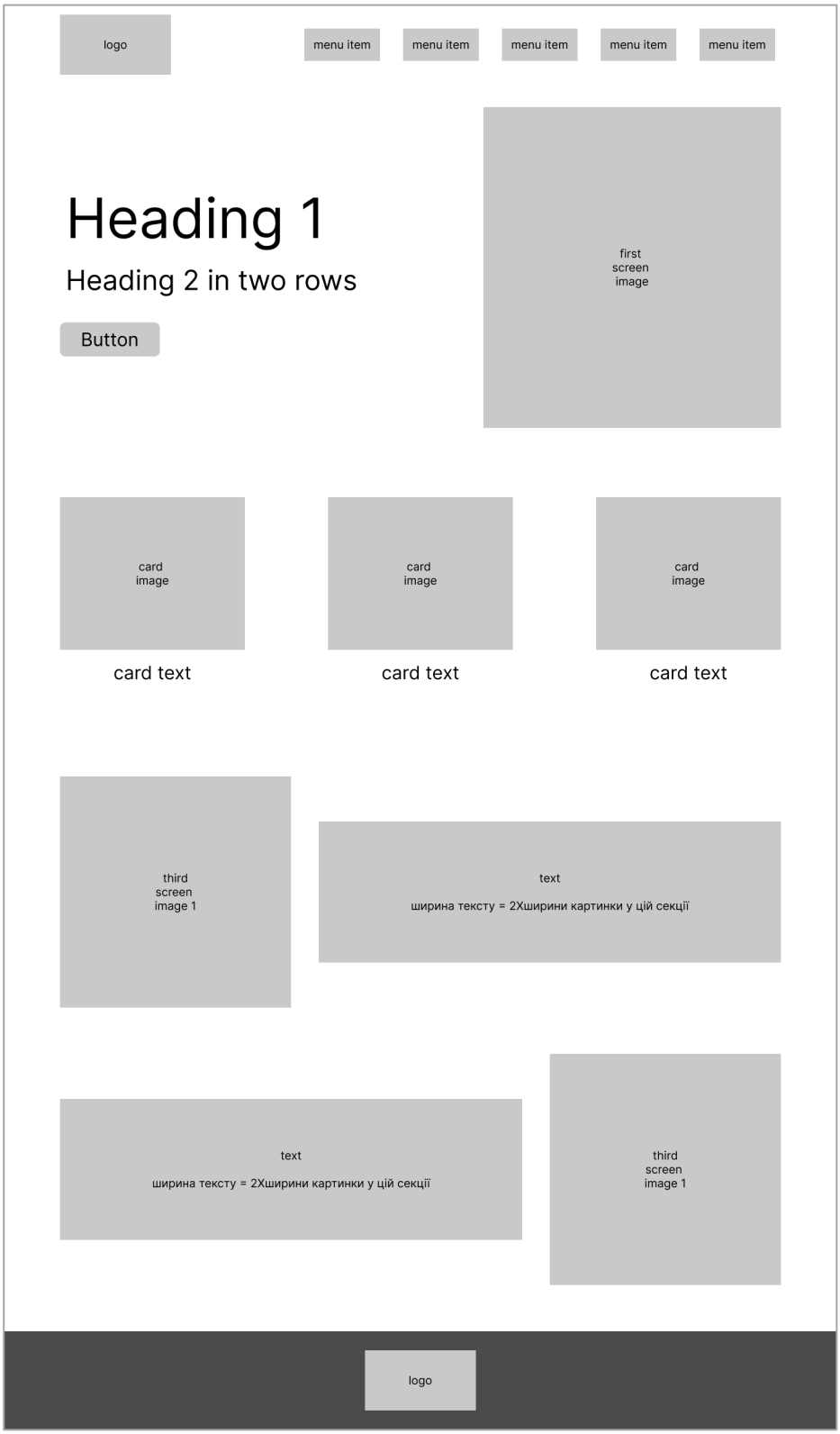


Рис. 1.13. Прототип сторінки (вайфрейм) до завдання 2 практикуму 1

Контрольні запитання

1. Що таке CSS3 Flexible box? Поясніть основну ідею CSS3 Flexible box?

2. Які переваги/недоліки відомі Вам застосування режиму розмітки CSS3 Flexible box?

3. Як вирівняти дочірній елемент по центру батьківського елемента одночасно і по вертикалі, і по горизонталі?

4. За допомогою якої властивості можливо змінити вирівнювання одного конкретного дочірнього елемента?

5. Яким чином домогтися того, щоб дочірні елементи не переходили на наступний рядок у разі перевищення їх загальної ширини за батьківську?

КП2. МЕТОДИ ПОЗИЦІЮВАННЯ ЕЛЕМЕНТІВ НА ВЕБСТОРІНЦІ. СТВОРЕННЯ КОМПОЗИЦІЇ ЗОБРАЖЕНЬ

Мета роботи – ознайомитися із методами позиціювання елементів на сторінці, навчитися використовувати основні сценарії властивості position для реальних задач.

Основні теоретичні відомості

Під *позиціюванням* мають на увазі розташування елемента в заданій системі координат. Можна виділити чотири типи позиціювання: нормальне (статичне), абсолютне, відносне, фіксоване. Залежно від вибраного типу, встановленого через властивість **position**, змінюватиметься система координат.

Разом з властивістю position доцільно використовувати наступну комбінацію властивостей (або лише кілька з них): *left, top, right, bottom* і *z - index*, за допомогою яких елемент можна позиціювати точніше й у певних випадках абсолютно незалежно від сусідніх елементів. Властивості *left, top, right, bottom* відповідають за переміщення елемента від його первісного положення в чотирьох напрямках: вправо, вниз, вліво, вгору відповідно. Властивість *z – index* відповідає за переміщення елемента на іншу площину (або рівень).

Статичне (нормальне) позиціювання

За замовчуванням, якщо для елемента не задано властивість position або йому встановлено значення static, шари позиціюються один за одним, при цьому кожен наступний починається з нового рядка. Причому шари розміщуються максимально близько до верхнього лівого кута сторінки та без відступів між кордонами. Крім цього, властивості left, top, right, bottom та їх значення, якщо такі визначені, ігноруються браузером.

`position: static`

Рис. 2.1. Візуальна підказка щодо статичного позиціонування

Відносне позиціювання

Елемент позиціюється згідно зі звичайним плином документа, а тоді відступає вбік відносно початкового положення на основі значень top, right, bottom та left. Відступ жодним чином не впливає на розміщення інших елементів; таким чином, простір, виділений для елемента у макеті сторінки, залишається таким самим, як був би, якби його position мала значення static.

`position: relative`

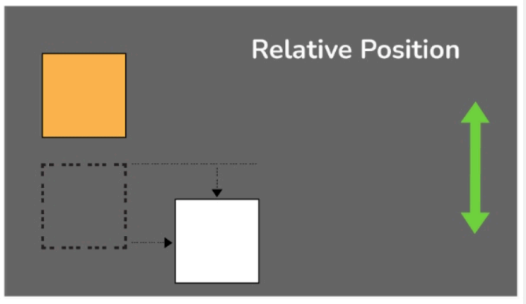


Рис. 2.2. Візуальна підказка щодо відносного позиціонування

Абсолютне позиціювання

Абсолютне позиціювання видаляє елемент з загального потоку (місце, що звільнив елемент заповнюється іншими елементами) і вставляється відносно свого батьківського, не статично позиційованого елемента, а якщо такого елемента немає, то їм вважається вікно браузера. Нове місце розраховується за допомогою властивостей left, top, right та bottom.

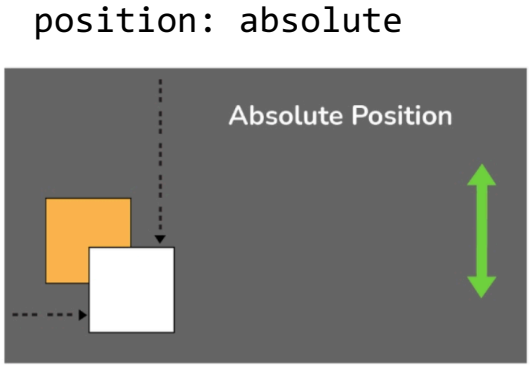


Рис. 2.3. Візуальна підказка щодо абсолютного позиціонування

На положення впливає значення властивості *position* батьківського елемента:

- якщо у батька значення position встановлено як static або батька немає, то відлік координат ведеться від краю вікна браузера.
- якщо у батька значення position задано як relative, то відлік координат ведеться від краю батьківського елементу.

Розміри абсолютно позиціонованого елементу (absolute), який має встановлене значення auto для властивостей width і height, відповідають його вмісту.

Тим часом блок може займати всю можливу висоту батьківського елементу, якщо має встановлене значення 0 для top і bottom, а також невизначену висоту (auto). Таким чином також може бути розтягнутий по горизонталі: із визначеними left і right та невизначеною шириною.

Варто відзначити такі особливості даного типу позиціювання:

- властивості top і left мають вищий пріоритет, ніж right і bottom, це означає, що при суперечності властивостей left і right значення right ігноруватиметься, також браузер поведеться щодо якості bottom.

- при використанні абсолютного позиціювання та заданні значень властивостям left, top, right, bottom та z - index початкове розташування шару стає доступним для заміщення іншими сусідніми елементами.

Фіксоване позиціювання

Якщо властивості position встановити значення fixed, елемент буде прив'язаний до певної властивості left, top, right, bottom точки на екрані та не змінюватиме свого положення при прокручуванні вебсторінки. Елемент зникає зі звичайного плину документа, і для нього не виділяється жодного простору у макеті сторінки.

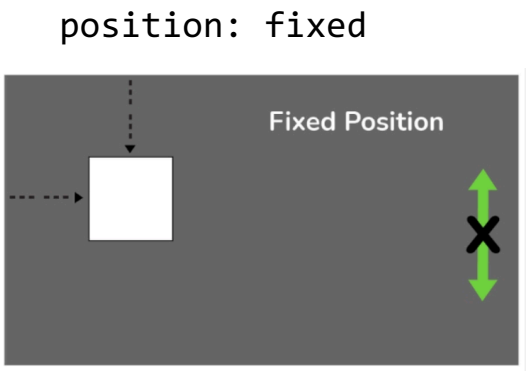


Рис. 2.4. Візуальна підказка щодо фіксованого позиціонування

При розміщенні фіксованого елемента за межами області видимості знизу або праворуч від неї не призводить до виникнення смуг прокручування.

Він позиціюється відносно до початкового контейнерного блока, встановленого областю перегляду, крім випадків,

- коли один з предків елемента має властивість transform, perspective або filter, значення котрої відмінне від none,

- коли властивість will-change має значення transform, – у таких випадках предок з однією з вищезгаданих властивостей поводиться як контейнерний блок.

Доцільним застосуванням цього типу позиціювання є створення меню, заголовків, нерухомих елементів, що містять форми швидкого зв'язку зі службою підтримки (online chat).

Фіксоване («липке», закріплене) позиціювання

Рис. 2.5. Візуальна підказка щодо «липкого» позиціонування

Елемент позиціюється згідно зі звичайним плином документа, а далі зміщується відносно до свого найближчого предка з прокручуванням та контейнерного блока (найближчого предка блокового рівня), на основі значень властивостей top, right, bottom та left. Відступ не впливає на позицію інших елементів.

Зверніть увагу, що липкий елемент «прилипає» до свого найближчого предка, який має «механізм прокручування» (утворений встановленням властивості overflow у значення hidden, scroll, auto чи overlay), навіть коли такий предок не є найближчим справді прокручуваним предком.

Липко позиціонований елемент – розглядається як відносно позиціонований, поки його контейнерний блок не перетне встановлений поріг (встановлений, наприклад, значенням властивості top, відмінним від auto) всередині свого кореня плину (чи контейнера, всередині якого він прокручується), після чого буде вважатись «застряглим», поки не зустріне протилежний край свого контейнерного блоку.

Z-index

```
z-index: auto| number | initial | inherit
```

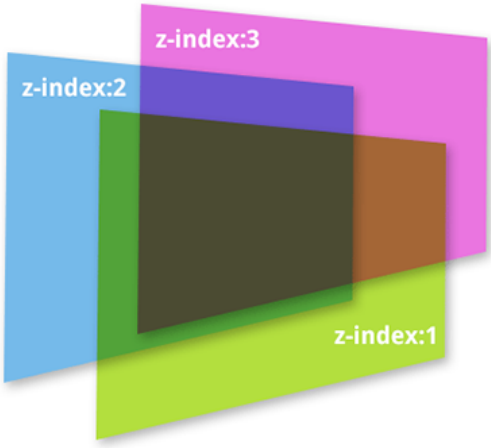


Рис. 2.6. Візуальний опис роботи властивості z-index

Властивість z-index дозволяє регулювати порядок шарування об’єктів під час відтворення вмісту.

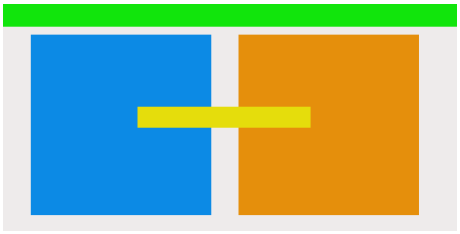
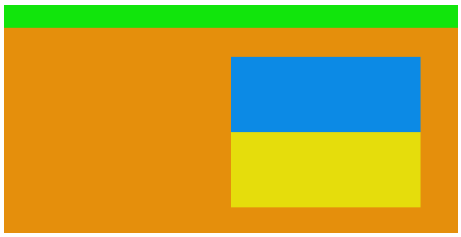
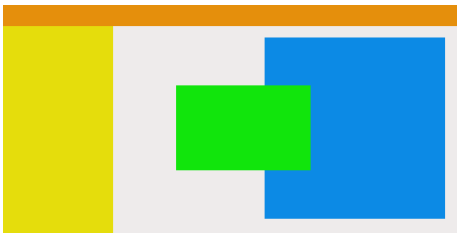
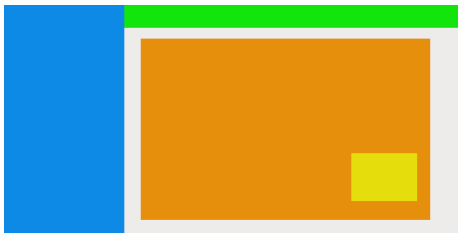
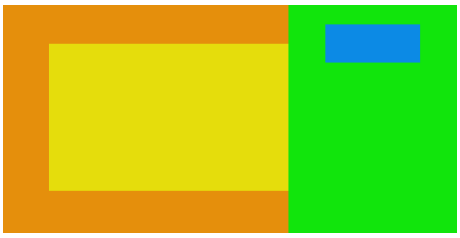
Z-index використовується для стабілізації порядку елементів, що перетинаються. Властивість z-index регулює вертикальний порядок перекриття елементів, а сам z-index визначає, який елемент буде розташовуватися вище за інші. Елемент з великим покажчиком порядку завжди буде розташовуватися вище (ближче до спостерігача) за елемент нижчого порядку.

Властивість z-index може бути задана цілим значенням (позитивним, нульовим або від’ємним), яке представляє положення елемента вздовж уявної осі z.

Хід роботи

1. Створіть сторінку з контейнером, у який вкладіть 4 блоки (divs), задайте їм ширину та висоту та різний колір та розмістіть їх згідно з вашим варіантом* (табл. 1.1) за допомогою різних методів позиціювання.

Таблиця 1. 1. – Варіанти завдання

Номер варіанту	Завдання	Номер варіанту	Завдання
1		2	
3		4	
5		6	

**Варіант на завдання визначається згідно зі списком групи, де 1, 7, 13, 19, 25 людина за списком – виконує 1 варіант; 2, 8, 14, 20, 26 людина за списком – виконує 2 варіант; 3, 9, 15, 21, 27 людина за списком – виконує 3 варіант; 4, 10, 16, 22, 28 людина за списком – виконує 4 варіант; 5, 11, 17, 23, 29 людина за списком – виконує 5 варіант; 6, 12, 18, 24, 30 – виконує 6 варіант.*

2. Згідно прототипу (вайфрейму) типової сторінки лендінг-пейдж, представленої у КП 1 (рис. 1.13), придумайте тематику Вашого проєкту та створіть візуальний дизайн першої секції та шапки сторінки (рис. 2.7). При цьому основну увагу у цій лабораторній роботі приділіть зображенню First screen image. Це зображення має являти собою композицію із різних елементів (можливо, зображень, тексту, іконок), які спозиціоновані між собою відповідним чином.

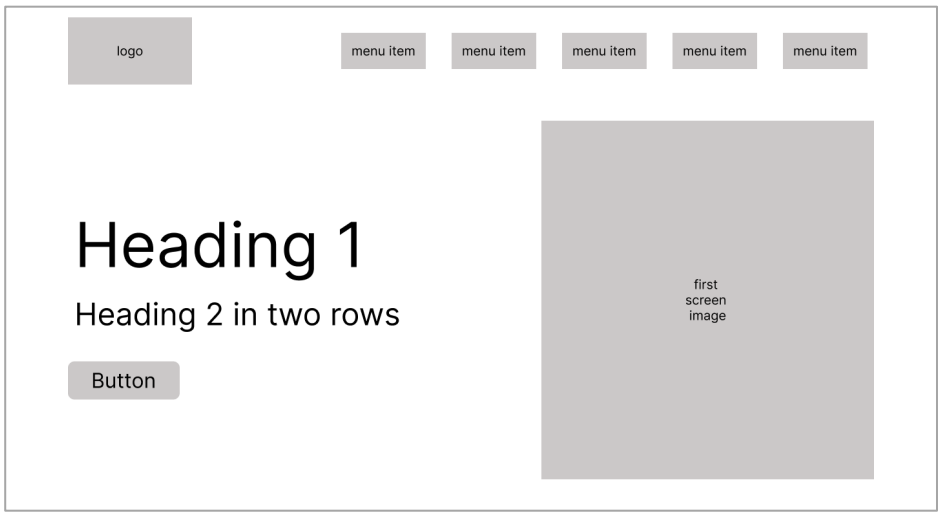


Рис. 2.7. Вайфрейм до завдання 2

3. Закодуйте розроблений макет першої секції, використавши розмітку, виконану у КП 1, додавши власних стилів та впровадивши створену композицію зображення, використовуючи необхідні методи позиціювання.

Контрольні запитання

1. Що таке властивість position? Які методи позиціювання Вам відомі, назвіть основні сценарії їх використання?

3. Як розтягнути абсолютно спозиціонований по вертикалі (від верху донизу), не використовуючи властивості висоти; та по горизонталі (від лівого краю до правого краю), не використовуючи властивості ширини?

5. Що таке z-index? Коли його використовують?

КП 3. ВІЗУАЛІЗАЦІЯ ІНФОРМАЦІЇ. ОПТИМІЗАЦІЯ ЗОБРАЖЕНЬ. СТВОРЕННЯ ТА ДОДАВАННЯ ІНФОРМАЦІЙНОЇ ГРАФІКИ НА ВЕБСТОРІНКУ

Мета роботи – здобути навички узагальнення інформації шляхом візуалізації, розвинути практичні навички використання графічного редактора Figma, ознайомитися із методами оптимізації графіки, навчитися використовувати різні методи оптимізації графіки для реальних задач.

Основні теоретичні відомості

Візуалізація інформації – це процес візуального представлення даних та інформації за допомогою графіків, діаграм, карт, зображень та інших візуальних елементів. Мета візуалізації полягає в тому, щоб допомогти сприйманню та розумінню складних даних, які можуть бути складними для сприйняття в табличному форматі або текстовому форматі великого обсягу.

Візуалізація інформації може бути використана в різних сферах, таких як бізнес, освіта, медицина, соціологія та багато інших. Наприклад, в бізнесі візуалізація може бути використана для аналізу фінансових даних, прогнозування продажів, аналізу ринку та ін.

При візуалізації інформації важливо враховувати аудиторію, яка буде сприймати дані. Ефективна візуалізація має бути зрозумілою та легко читабельною. Крім того, необхідно враховувати, що різні типи даних вимагають різних видів візуалізації. Наприклад, для відображення часових рядів може використовуватись лінійна діаграма, а для порівняння кількості елементів може бути використана стовпчаста діаграма.

В цілому, візуалізація інформації є потужним інструментом для розуміння та аналізу даних, і вона може бути використана для різних цілей в різних сферах діяльності.

Оптимізація зображень – це процес зменшення розміру файлу зображення, зберігаючи якість зображення на прийнятному рівні. Цей процес

може знизити час завантаження вебсторінок і поліпшити їх продуктивність, а також зменшити використання пропускної здатності мережі.

Оптимізація зображень на вебресурсі спрощує зчитування з нього інформації пошуковими алгоритмами й підвищує комфорт користувачів. Цілі оптимізації зображень:

- прискорення завантаження (*швидкість завантаження сторінки – один з найважливіших факторів ранжирування google. виражається він метрикою lcp (largest contentful paint) – час промальовування найбільшої картинки на екрані*);
- індексація картинок пошуковими системами;
- підвищення зручності використання (юзабіліті).

Загальноприйнятими параметрами зображень на вебсайтах є: якість, формат, розмір.

Якість. При роботі з зображенням важливо пам’ятати, що вони призначені для роботи з цільовою аудиторією сайту. Тому важливо зробити їх не тільки яскравими й інформативними, але ще і якісними. Дуже важливо звертати увагу на пропорційність зображення. Воно не повинно бути занадто розтягнутим або стисненим по одній зі сторін. На картинках при цьому не повинні зустрічатися такі помилки, як пікселізація, бляклість або каламутність.

Формат. У вебдизайні використовують такі формати:

- JPEG, JPG – Joint Photographic Experts Group – формат для фотографій з широким колірним спектром. Під час стиснення картинки втрачається якість: дрібні деталі пропадають, стають розмитими. На зображеннях з великою кількістю об’єктів під час стиснення на 25-30% різниця практично непомітна.

- PNG – Portable Network Graphic – хороший формат для логотипів, значків і маленьких зображень. Підтримує прозорість.

- GIF – Graphic Interchange Format – формат, схожий на PNG, але з можливістю анімувати зображення. Має відносну прозорість, коли ви можете зробити прозорим будь-який колір, але по всій картинці, а не на окремій ділянці. Підійде для значків, іконок, логотипів.

- SVG – Scalable Vector Graphics – формат для зображень векторної графіки. Технологія дозволяє масштабувати картинки без втрати якості. Формат хороший для діаграм, логотипів і значків.

- WEBP – формат розроблений в Google спеціально для прискореного завантаження зображень на вебресурсі. Він стискає картинки на 25-30% без видимої різниці. Стиснення працює краще, ніж в JPG внаслідок використання новішої технології. Недолік формату в підтримці: з WebP не працює WordPress, браузер Internet Explorer, більшість ресурсів не розпізнають його при спробі завантаження зображення через призначений для користувача інтерфейс.

- AVIF – абсолютно новий формат, який важить у 2 рази менше за JPEG і на 20% менше WebP (новий тренд для вебмайстрів). Проте може мати проблеми з кросбраузерністю.

Розмір графічних файлів. Даний фактор роботи з картинками можна розглядати з двох основних позицій:

- довжина і ширина фото, які позначаються в піксельному варіанті;
- вага файлу картинки, що вимірюється в мегабайтах.

З технічного боку питання оптимізовувати картинки необхідно з метою зменшення часу завантаження ресурсу. Оптимізовувати фото по довжині й ширині необхідно таким чином, щоб пропорції збігалися з бажаними й зручними для користувачів сторінки. Деякі сучасні сервери самостійно оптимізують зображення оригіналів, створюючи зручні версії для мобільного і десктопного варіантів.

Існує кілька *способів оптимізації зображень:*

1. Використання правильного формату зображення. Для фотографій зазвичай використовують формат JPEG, а для зображень з прозорістю – формат PNG. Використання правильного формату зображення допоможе знизити розмір файлу без втрати якості.

2. Зменшення розмірів зображення. Зменшення розміру зображення (ширини, висоти) до прийнятного розміру може допомогти знизити розмір файлу.

3. Використання стиснення зображень. Стиснення зображень може знизити розмір файлу без втрати якості. Існують різні алгоритми стиснення, такі як lossless (стиснення без втрат) та lossy (стиснення з втратами).

Таблиця 3.1. – Порівняння алгоритмів стиснення

Основа для порівняння	Стиснення з втратами	Стиснення без втрат
Алгоритм	Перетворення кодування, DCT, DWT, фрактальне стиснення, RSSMS.	RLW, LZW, арифметичне кодування, кодування Хаффмана, кодування Шеннона Фано.
Використовується в	Зображення, аудіо та відео.	Текст або програма, зображення та звук.
Застосування	JPEG, GUI, MP3, MP4, OGG, H-264, MKV тощо.	RAW, BMP, PNG, WAV, FLAC, ALAC тощо
Можливість зберігання даних каналу	Більше	Менш порівняно з методом з втратами

4. Використання інструментів оптимізації зображень. Існують безліч інструментів оптимізації зображень, які можуть допомогти знизити розмір файлу. Наприклад, онлайн інструменти: Compressor.io, TinyPNG, JPEGmini, Image.online-convert, Optimizilla тощо; десктопні інструменти: Adobe Photoshop, Total Image Converter, ImageOptim, ImageAlpha тощо; інструменти на сервері ресурсу: Kraken, Imagify тощо.

Важливо пам'ятати, що оптимізація зображень має бути здійснена таким чином, щоб якість зображення не була пошкоджена, і щоб вебсторінка залишалася естетично привабливою та легко зчитуваною.

Хід роботи

1. Відповідно до обраної теми вебсторінки згідно з попередніми практикумами підготуйте матеріал, який доцільно представити у вигляді інфографіки *.

2. Запустіть Figma та створіть новий файл. Визначте стиль своєї інфографіки. Ви можете використовувати шрифти, кольори та елементи дизайну, які відповідають вашому бренду або темі вашого загального проєкту.

3. Із використанням відповідних інструментів, розмістіть елементи на своїй інфографіці та відформатуйте їх за потреби.

4. Додайте текст та мітки до своєї інфографіки, щоб пояснити графік чи діаграму у випадку їх наявності.

5. Збережіть свою інфографіку. Експортуйте її у відповідному форматі, такому як PNG, SVG або JPG, щоб використовувати її на вашому сайті.

6. За допомогою відповідного інструмента оптимізуйте Ваше зображення та зробіть аналіз проведеної оптимізації: укажіть рівень оптимізації (на скільки відсотків «стиснулось») зображення, чіткості країв зображення, розмиття, пікселізації тощо. У протоколі відобразіть зображення до оптимізації та після.

7. Впровадьте оптимізоване зображення у код сторінки, застосовуючи необхідні Вам методи (теги *<figure>, <figcaption>*, *<img>* властивості *background-image, background-size, background-repeat, background-position*).

Контрольні запитання

1. Що таке візуалізація інформації? У яких випадках її застосовують?

2. Що таке оптимізація зображень? Які алгоритми використовуються для стиснення зображень?

3. Опишіть загальноприйняті вимоги до зображень при відображенні на сайтах.

4. Які основні формати зображень використовуються на вебресурсах?

5. Які теги використовуються для впровадження зображень на вебресурс?

** У випадку відсутності подібного матеріалу щодо Вашого проєкту Ви можете обрати тему для інфографіки з наступного списку варіантів:*

- *Географічний розподіл інформації – наприклад, розташування користувачів соціальної мережі в різних країнах світу.*
- *Статистика – візуалізація статистичних даних, таких як показники здоров'я нації, економічні показники країни тощо.*
- *Відношення – відображення відношення різних параметрів, наприклад, співвідношення доходів і видатків.*
- *Історія – візуалізація історичних подій та трендів у часі.*
- *Наука – візуалізація наукових даних, таких як молекулярні структури, геномні дані, клітинні процеси тощо.*
- *Природа – візуалізація природних явищ, таких як географічні формації, кліматичні зміни, розподіл тваринних видів тощо.*
- *Спорт – візуалізація спортивних рекордів та статистики, таких як рекорди велогонщиків, баскетбольних матчів тощо.*
- *Технології – візуалізація технічної інформації, такої як схеми та процеси виробництва, розробка нових технологій тощо.*
- *Мистецтво – візуалізація творів мистецтва, таких як картини, скульптури, архітектурні форми тощо.*
- *Психологія – візуалізація психологічних даних та трендів, таких як емоції, психологічні тестування, соціальна психологія тощо.*

КП 4. МЕТОДИ ВПРОВАДЖЕННЯ ГРАФІЧНОГО КОНТЕНТУ У ВЕБСТОРІНКУ. СТВОРЕННЯ ІНТЕРАКТИВНОГО ЗОБРАЖЕННЯ

Мета роботи – закріпити знання щодо впровадження графіки на вебсторінку різними методами, навчитися створювати зображення з інтерактивними зонами та впроваджувати його на вебсторінку.

Основні теоретичні відомості

Тег <img> для додавання зображення на веб-сторінку

Тег <img> використовується для вбудовування зображень у HTML-документ. Він є самозакритим, тобто не має пари у вигляді закриваючого тега. Зображення, додані за допомогою цього тега, є зовнішніми елементами, що не впливають на текстовий вміст документа, але додають до нього візуальну складову. Основні атрибути тега `<img>`:

- `src (source)` – визначає шлях до зображення, яке буде завантажено на вебсторінку. Форматом є абсолютний або відносний URL.

Синтаксис:

```
<img src="images/students-VPI.jpg" />
```

- `alt (alternative text)` – задає текст, який відображатиметься, якщо зображення не вдалося завантажити. Також використовується для опису зображення у цілях доступності (наприклад, для користувачів із вадами зору). Синтаксис:

```
<img src="images/students-VPI.jpg" alt="Студенти НН ВПІ" />
```

- `width` та `height` – контролюють розміри зображення. Якщо один із параметрів відсутній, зображення змінюється пропорційно другому. Синтаксис:

```
<img src="images/students-VPI.jpg" width="300" height="200"/>
```

- `title` – відображає підказку, яка з’являється під час наведення курсора на зображення. Синтаксис:

```
<img src="images/students-VPI.jpg" title="Студенти НН ВПІ"/>
```

- `align` – визначає вирівнювання зображення відносно тексту або інших елементів. Може приймати наступні значення: left – вирівнює зображення ліворуч; right – вирівнює зображення праворуч; top – вирівнює по верхньому краю; bottom – вирівнює по нижньому краю; middle – вирівнює по середній лінії тексту. Синтаксис:

```
<img src="images/ students-VPI.jpg " align="left" />
```

Примітка: Атрибут align є застарілим, сучасні розробники використовують CSS для вирівнювання.

Зображення як посилання

Тег <img> можна вкласти у тег <a> для створення посилання.

```
<a href="https://example.com">
    <img src="images/button.png" alt="Кнопка" />
</a>
```

Тег <figure>, <figcaption>

Тег <figure> визначає самодостатній вміст, як-от ілюстрації, діаграми, фотографії, переліки кодів тощо. Елемент <figcaption> використовується для додавання підпису для <figure>елемента. Синтаксис наступний:

```
<figure>
    <img src="посилання на зображення"
         alt="Альтернативний текст">
    <figcaption>Підпис до зображення</figcaption>
</figure>
```

Фонове зображення, background-властивості

Фонове зображення – це елемент дизайну вебсторінки, який використовується для додавання візуального контексту або покращення естетичного вигляду елемента. CSS надає кілька властивостей для роботи з фоном, що дозволяють налаштовувати його зображення, колір, розмір, позицію та повторення. Основні background-властивості

- `background-color` – властивість, що визначає колір фону для елемента. Значенням може бути будь-який валідний колір (наприклад, red, #ff0000, rgb(255, 0, 0) тощо).

```
div {
    background-color: lightblue;
}
```

- `background-image` – використовується для встановлення фонового зображення. Значенням є шлях до зображення у форматі URL або ключове слово none.

```
div {
    background-image: url('background.jpg');
}
```

- `background-repeat` – визначає, як фонове зображення повторюється. Значеннями може бути: repeat – повторює зображення і горизонтально, і вертикально (за замовчуванням); repeat-x – повторює лише горизонтально; repeat-y – повторює лише вертикально, no-repeat – не повторює.

```
div {
    background-image: url('background.jpg');
    background-repeat: no-repeat;
```

```
}
```

- background-position – визначає початкову позицію фонового зображення. Значеннями можуть бути ключові слова: top, right, bottom, left, center; або точні координати: px, %.

```
div {
    background-image: url('background.jpg');
    background-position: center;
}
```

- background-size – визначає розмір фонового зображення. Значеннями може бути: auto – зображення зберігає свій початковий розмір; cover – масштабування для повного покриття елемента з обрізанням частин; contain – масштабування для повного відображення всього зображення всередині елемента; точні розміри: px, %.

```
div {
    background-image: url('background.jpg');
    background-size: cover;
}
```

- background-attachment – визначає, чи фонове зображення буде прокручуватися разом зі сторінкою. Значення: scroll – зображення прокручується разом зі сторінкою; fixed – зображення закріплене і не прокручується.

```
div {
    background-image: url('background.jpg');
    background-attachment: fixed;
}
```

Скорочена властивість background

Всі зазначені вище властивості можна об’єднати у скорочений запис:

```
background: <color> <image> <position> / <size> <repeat>
                        <attachment>;
```

Наприклад:

```
div {
    background: lightblue url('background.jpg') center /
                cover no-repeat fixed;
}
```

Карта-зображення (image map)

Карта-зображення дозволяє створювати інтерактивні області (посилання) на зображенні. Це корисно для створення складних елементів навігації, інтерактивних карт або для виділення різних частин зображення, які можуть виконувати різні функції.

Переваги карти-зображення: легке створення інтерактивних областей на одному зображенні; підтримка кількох форм для визначення областей, покращення візуальної навігації на вебсторінках.

Недоліки карти-зображення: складність створення, якщо карта має багато областей; потреба у точних координатах для кожної області; обмежена адаптивність на мобільних пристроях без використання додаткових стилів.

Основні кроки створення карти-зображення

1. Використання тега <img> із атрибутом usemap. Атрибут usemap прив’язує зображення до певної карти. Значення цього атрибута починається з #, після якого йде ім’я карти, яке збігається з атрибутом name або id тега <map>.

```
<img src="map.jpg" usemap="#map" alt="Інтерактивна карта" />
```

2. Створення карти за допомогою тега <map>. Тег <map> визначає карту-зображення та її інтерактивні області.

```
<map name="map">
    <!-- Області зображення -->
</map>
```

3. Визначення інтерактивних областей за допомогою тега <area>. Тег <area> описує кожну інтерактивну область на зображенні. Основні атрибути тега <area>: shape – визначає форму області; rect – прямокутник; circle – коло; poly – багатокутник; coords – координати області, що відповідають формі, href – посилання на ресурс, який відкривається під час кліку (для областей без посилань використовується href="#"), alt – альтернативний текст для області, title – текст підказки, що відображається при наведенні на область.

Для rect (прямокутник) координати задаються чотирма числами x1, y1, x2, y2, що визначають верхній лівий і нижній правий кути.

```
<area shape="rect" coords="50,50,200,150" />
```

Для circle (коло) координати задаються трьома числами x, y, r, де x і y – центр кола, а r – радіус.

```
<area shape="circle" coords="300,100,50" />
```

Для poly (багатокутник) координати задаються набором чисел, що визначають послідовні вершини багатокутника.

```
<area shape="poly" coords="400,50,450,100,400,150,350,100" />
```

Приклад синтаксиса карти-зображення наступний:

```
<img src="world-map.jpg" usemap="#worldmap" alt="Карта світу"
/>
<map name="worldmap">
    <!-- Прямокутна область -->
    <area shape="rect" coords="50,50,200,150"
    href="https://example.com/region1" alt="Регіон 1"
    title="Регіон 1" />
    <!-- Кругла область -->
    <area shape="circle" coords="300,100,50"
    href="https://example.com/region2" alt="Регіон 2"
    title="Регіон 2" />
    <!-- Багатокутна область -->
    <area shape="poly" coords="400,50,450,100,400,150,350,100"
    href="https://example.com/region3" alt="Регіон 3"
    title="Регіон 3" />
</map>
```

Хід роботи

1. Проаналізуйте зображення, створене у попередній роботі. Це зображення буде використовуватись як карта.

2. На цьому зображенні виділіть мінімум дві ділянки (area 1, area 2), які будуть слугувати переходом на наступні сторінки.

Підказка: Обов'язкові елементи на даній сторінці – теги <img>, <map>, <area> атрибути usemap, name, id, shape, coords, href, alt, title. Необхідно використати різні форми (rect, circle або poly).

3. Створіть сторінку, на котру буде здійснено перехід із першої ділянки (area 1) карти зображень, яка матиме зображення, які обтікається текстом зліва та справа. Одне із використаних зображень має стати посиланням на наступну сторінку.

Підказка: Обов'язкова властивість – float. Додатково можна скористатись атрибутом align.

4. Створіть сторінку, на яку буде здійснено перехід із попередньо створеної сторінки. Сторінка має вміщувати текст на фоновому зображенні. У тексті створіть гіперпосилання, яке б вело на першу (головну) сторінку із картою-зображенням.

Підказка: Обов'язкові властивості – background-image, background-size, background-repeat, background-position.

5. Створіть сторінку, на котру буде здійснено перехід із другої ділянки (area 2) карти зображень, яка матиме зображення із підписом, яке обтікається текстом зліва або справа. У тексті створіть гіперпосилання, яке б вело на першу (головну) сторінку із картою-зображенням.

Підказка: Обов'язкові елементи – <figure>, <figcaption> властивість – float.

6. Занесіть результати роботи у протокол (прінтскіни створених сторінок – написаного коду та відображення у браузері) та сформулюйте висновки по роботі. Готовим результатом роботи є чотири створених html-сторінок із вищезазначеними вимогами та оформлений протокол із висновками. Ділянок на карті зображень та відповідних сторінок, на які здійснюється перехід, може за бажанням бути більше.

Рекомендація: Пропрацюйте із контентом, щоб переходи між сторінками були логічними.

Контрольні запитання

1. Як встановлюються зображення в HTML-документи?
2. Які основні формати зображень використовуються у вебі?
3. Яким чином можна змінити розміри зображення та його розташування на сторінці?
4. Які теги використовуються при створенні карти-изображення?
5. Яким тегом семантично правильно додавати підпис до зображення?

КП 5. СТВОРЕННЯ ВЕКТОРНИХ ІЛЮСТРАЦІЙ У ФОРМАТІ SVG, ОПТИМІЗАЦІЯ ТА ВБУДОВУВАННЯ ЇХ У ВЕБСТОРІНКУ

Мета роботи – ознайомитися із основними правилами створення, оптимізації та додавання на веб-сторінку векторних ілюстрацій у форматі svg.

Основні теоретичні відомості

SVG (скорочено від Scalable Vector Graphics) – масштабована векторна графіка. Це унікальний тип формату зображення для векторної графіки, написаною розширюваною мовою розмітки (XML).

Специфікація SVG є відкритим стандартом, розробленим Консорціумом Всесвітньої павутини (W3C) з 1999 року.

Зображення SVG та їх поведінка визначені в текстових файлах XML. Це означає, що їх можна шукати, індексувати, створювати сценарії та стискати.

Оскільки це файли XML, зображення SVG можна створювати та редагувати за допомогою будь-якого текстового редактора, але частіше вони створюються за допомогою програм для малювання.

Існує багато причин використовувати зображення SVG, ось деякі з них:

- Зображення SVG не втрачають своєї якості при збільшенні чи зміні розміру;
- Вони можуть бути створеними та редагованими за допомогою текстового редактора;
- Вони доступні та можуть бути анімовані;
- Вони невеликі за розміром файлу та добре масштабуються;
- Їх можна шукати, індексувати, скриптувати та стискати.

Методи додавання svg-графіки на сторінку:

- як елемент зображення <img>:

```
<img src = «graduation.svg» alt=«My Happy SVG»/> ;
```

- як background-image:

```
body { background-image: url(graduation.svg); };
```

- як <object>:

```
<object data=« graduation.svg « width=«300» height=«300»>
</object>;
```

- як <iframe>:

```
<iframe src=« graduation.svg.svg» width=«400px»
height=«400px»></iframe>
```

- як <embed>:

```
<embed src=«happy.svg» />
```

- як вбудований svg код:

```
<svg width=«500» height=«500» > … </svg>
```

Вбудований SVG

Сучасні браузери розуміють svg тег і можна розмістити файл безпосередньо. У цьому варіанті код SVG, отримати який можна відкривши будь-який файл SVG текстовим документом, вбудовується безпосередньо в код сторінки. Безперечно, така конструкція погіршує читаність коду, і збільшує його обсяг, але відкриваються нові можливості. Якщо ви використовуєте вбудоване зображення SVG в документі HTML, це зменшує час завантаження, оскільки служить як HTTP-запит.

Для вбудовування відкрийте зображення SVG у текстовому редакторі, копіюйте код та вставте його всередині елемента <body> (або іншому потрібному місці) у своєму документі HTML.

Структура SVG-файлу

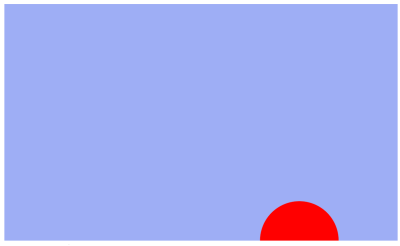
Атрибут **viewBox** (реєстр важливий) є певним прямокутником, який обмежує зону виведення намальованих фігур.

Наприклад, поставимо обмежувальну область 200 на 100 одиниць з початком координат (0, 0), яка знаходиться у верхній лівій точці. Одиниці виміру та початок координат є умовними поняттями, щоб було від чого відштовхуватися.

```
<svg viewBox=«0 0 200 100»>
```

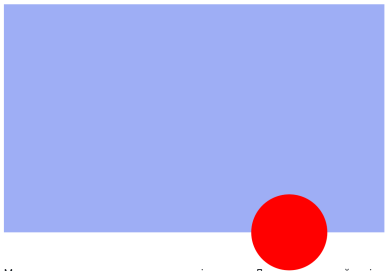
Частини фігури, які виходять за межі viewBox , обрізаються і ми їх не бачимо.

```
<svg viewBox=«0 0 200 120»
class=«box»>
  <circle cx=«150» cy=«120»
r=«20» fill=«red» />
</svg>
```



Використовуючи CSS-властивість overflow:visible можна запобігти такому «обрізанню» зображень:

```
<svg viewBox=«0 0 200 120»
style=«overflow:visible»
class=«box»>
<circle cx=«150» cy=«120»
r=«20» fill=«red» /> </svg>
```



Можна також використовувати негативні значення. Допустимо, є такий варіант зі стандартною точкою відліку (0, 0):

```
<svg viewBox=«0 0 500 100»
class=«box»>
<circle cx=«50» cy=«0»
r=«50» fill=«red» />
</svg>
```

Не змінюючи координати кола, а змінюючи координати у viewBox можемо вписати коло у прямокутник.

```
<svg viewBox=«-10 -50 500 100»
class=«box»>
<circle cx=«50» cy=«0»
r=«50» fill=«red» /> </svg>
```

Тег <g>

Елемент **<g>** використовується для логічного угруповання набору пов'язаних графічних елементів. Це можна порівняти з групуванням об'єктів у графічних редакторах. Він об'єднує у групу весь свій вміст.

Як правило, йому задається ідентифікатор, яким буде здійснюватися звернення до цієї групи елементів надалі. Будь-які стилі, що застосовуються до елемента **<g>**, також будуть застосовані до всіх його нащадків. Це дозволяє

задавати стилі та перетворення, а також додавати інтерактивність та анімацію одразу цілій групі об'єктів.

У прикладі наведено код svg-ілюстрація героя-магістра (рис. 5.1), код якого наведено нижче, видно, що основні частини тіла героя згруповані логічно і цим групам привласнено ідентифікатори.

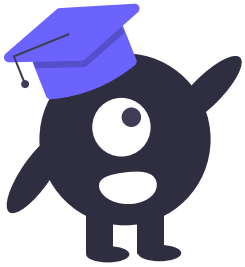


Рис. 5.1. Векторне зображення героя-магістра у форматі svg

```
<svg xmlns=«http://www.w3.org/2000/svg»
xml:space=«preserve» id=«graduate» x=«0» y=«0» style=«enable-
background:new 0 0 148.9 151.1» version=«1.1» viewBox=«0 0
148.9 151.1»>
<style>
.st0{fill:#2f2e41}
.st1{fill:#fff}
.st2{fill:#3f3d56}
.st3{fill:#6c63ff}
</style>
<g id=«hero»>
<circle id=«body» cx=«75.1» cy=«79» r=«43.1»
class=«st0»/>
<g id=«legs»>
<g id=«leg2»>
<path d=«M55.5 112.8h13.1v23.4H55.5z» class=«st0»/>
<ellipse cx=«66.4» cy=«136.6» class=«st0» rx=«10.9»
ry=«4.1»/>
```

```
    </g>
    <g id=«leg1»>
    <path d=«M81.6 112.8h13.1v23.4H81.6z» class=«st0»/>
    <ellipse cx=«92.5» cy=«136» class=«st0» rx=«10.9»
ry=«4.1»/>
    </g>
    </g>
    <g id=«hands»>
    <ellipse id=«hand1» cx=«116.2» cy=«54.9» class=«st0»
rx=«23.9» ry=«7.5» transform=«rotate(-45.02 116.19 54.93)»/>
    <ellipse id=«hand2» cx=«30.5» cy=«95.9» class=«st0»
rx=«23.9» ry=«7.5» transform=«rotate(-55.22 30.49 95.92)»/>
    </g>
    <g id=«eye»>
    <circle id=«eye_x5F_white» cx=«73.3» cy=«72.9» r=«14.7»
class=«st1»/>
    <circle id=«eye_x5F_black» cx=«78.3» cy=«67.9» r=«4.9»
class=«st2»/>
    </g>
    <path id=«mouth» d=«M90.9 100.2c1.4 5.2-3.8 10.2-11.7
11.1s-15.5-2.5-17-7.7c-1.4-5.2 4.7-7 12.6-8s14.7-.6 16.1
4.6z» class=«st1»/>
    <g id=«hat»>
    <path id=«hat3» d=«M60 53.7c-7.1 2.8-14.6 4.6-22.2 5.2-
1.1.1-2.1-.6-2.5-1.6L28.8 41c-.5-1.3.1-2.7 1.4-3.3L71
21.3c1.3-.5 2.7.1 3.3 1.4L80.8 39c.4 1 .1 2.2-.8 2.9-6.1 4.9-
12.8 8.9-20 11.8z» class=«st3»/>
    <path id=«hat2» d=«m73.7 23 1.1 2.7-19 16.6c-.4.3-.8.5-
1.3.5l-24.4.6-1-2.5c-.4-1 .1-2.2 1.1-2.6L71 21.9c1.2-.4 2.3.1
2.7 1.1z» style=«opacity:.2;enable-background:new»/>
```

```
    <path id=«hat1» d=«M54 39.4c-.3.1-.6.2-.9.2l-36.4.8c-1.4
0-2.5-1.1-2.6-2.4 0-.7.3-1.5.9-2l25.7-22.2c.4-.4.9-.6 1.5-
.6L79.8 11c1.4-.1 2.6 1 2.6 2.3 0 .8-.3 1.5-.8 2L54.7 39c-
.2.2-.5.3-.7.4z» class=«st3»/>
    <path id=«hat_x5F_lines» d=«m24.9 50.2-1 .4-5.4-13.5
28.3-11.4.4 1-27.4 11z» class=«st2»/><circle
id=«hat_x5F_circle» cx=«24.8» cy=«51.3» r=«2» class=«st2»/>
    </g>
    </g>
    </svg>,
```

Тепер стає можливим одночасно звертатись до всіх елементів, які входять до цих груп, одночасно за допомогою виклику того чи іншого ідентифікатора. Угрупування елементів може бути дуже корисним не тільки для організації та структурування документа. Особливу користь вона може принести, якщо ви захочете додати до SVG графіки інтерактивності або задати якісь перетворення. Згрупувавши елементи, можна переміщати їх, масштабувати або повертати всі разом, зберігаючи їхнє положення один щодо одного.

Так, у випадку зі згрупованим героєм-магістром можна масштабувати його лише одним рядком CSS:

```
#hero {transform: scale(2);}
```

Тег <use>

<use> – інструмент клонування елементів SVG. З його допомогою можна дублювати існуючі та задані елементи. Також цей інструмент дозволяє змінювати створені копії.

Елемент <use> приймає як атрибути координати x і y, висоту (height), ширину (width) і посилання на вихідний елемент (xlink:href). Як посилання виступає ідентифікатор об'єкта.

Допустимо, у нас є SVG-елемент із заданим ідентифікатором:

```
<circle cx=«50» cy=«50» r=«10» fill=«red» id=«red_circle» />
```

За допомогою <use> цей елемент можна скопіювати:

```
<use xlink:href=«#red_circle» />
```

Тег <defs>

Щоб уникнути проблем із властивостями, які ми хочемо перевизначити, використовують тег **<defs>.** У такому разі всі атрибути оригіналу не передаватимуться на його клон.

Зберігатися в **<defs>** може будь-що, починаючи з групи елементів, на зразок нашого героя, і закінчуючи маскою або градієнтом. Це шаблон для подальшого використання. Сам по собі він ніколи не відображається.

```
<defs>
    <circle id=«parent_circle» r=«20»/>
</defs>
```

Все, що описано в тезі defs, не буде відображено у SVG; задані елементи будуть відображатися лише через посилання. Тепер додамо клону атрибути, яких немає в оригіналу:

```
<use x=«70» y=«20» xlink:href=«#parent_circle» fill=«red»/>
```

Тег <symbol>

Елемент **<symbol>** схожий на <g>: він також надає можливість групувати елементи. Можна виділити дві основні відмінності:

- елемент <symbol> не відображається сам собою. Цим він схожий на <defs>;

- елемент <symbol> може мати власні атрибути viewBox та preserveAspectRatio. Це дозволяє йому уміститися в області перегляду (viewport) так, як цього ви хочете, а не як це визначено за умовчанням.

У більшості випадків <symbol> підходить для визначення елементів, що повторно використовуються (символів). Він так само служить шаблоном для <use>. А маючи власні атрибути viewBox і preserveAspectRatio, він може розтягуватися на прямокутну область перегляду, що задається в елементі <use>, що посилається на нього. Врахуйте, що елементи <symbol> визначають нову область перегляду щоразу, коли викликаються елементом <use>.

Ця особливість елемента <symbol> дозволяє визначати елементи, які не залежать від області перегляду, в яку вони потраплять. Вони завжди відображатимуться заданим чином.

Основні форми svg

Існує кілька основних форм, які використовуються для більшості малюнків SVG. Призначення цих фігур досить очевидне з їхніх назв. Усі основні форми показані на наступному зображенні.

```
<svg width=«200» height=«250»
version=«1.1»
xmlns=«http://www.w3.org/2000/svg»>
<rect x=«10» y=«10» width=«30»
height=«30» stroke=«black»
fill=«transparent» stroke-width=«5»/>
<rect x=«60» y=«10» rx=«10» ry=«10»
width=«30» height=«30» stroke=«black»
fill=«transparent» stroke-width=«5»/>
```

```
    <circle cx=«25» cy=«75» r=«20»
stroke=«red» fill=«transparent» stroke-
width=«5»/>
    <ellipse cx=«75» cy=«75» rx=«20»
ry=«5» stroke=«red» fill=«transparent»
stroke-width=«5»/>
    <line x1=«10» x2=«50» y1=«110»
y2=«150» stroke=«orange» stroke-
width=«5»/>
    <polyline points=«60 110 65 120 70
115 75 130 80 125 85 140 90 135 95 150
100 145» stroke=«orange»
fill=«transparent» stroke-width=«5»/>
    <polygon points=«50 160 55 180 70
180 60 190 65 205 50 195 35 205 40 190
30 180 45 180» stroke=«green»
fill=«transparent» stroke-width=«5»/>
    <path d=«M20,230 Q40,205 50,230
T90,230» fill=«none» stroke=«blue»
stroke-width=«5»/>
    </svg>
```

Елемент **<rect>**малює на екрані прямокутник.

```
<rect x=«60» y=«10» rx=«10» ry=«10» width=«30» height=«30»
stroke=«black» fill=«transparent» stroke-width=«5»/>,
```

де x – позиція x верхнього лівого кута прямокутника, y – позиція y верхнього лівого кута прямокутника, width – ширина прямокутника, height –

висота прямокутника, rx – радіус x кутів прямокутника, ry – радіус y кутів прямокутника.

Елемент **<circle>** малює коло на екрані. Для визначення форми та розміру елемента потрібні три основні параметри.

```
<circle cx=«25» cy=«75» r=«20» stroke=«red»
  fill=«transparent» stroke-width=«5»/>,
```

де r – радіус кола, cx – положення x центру кола, cy – положення y центру кола.

Елемент **<ellipse> –** це більш загальна форма елемента <circle>, у якій можна масштабувати радіус x і y (зазвичай у математиці їх називають *великою* та *малою* піввісями) кола окремо.

```
<ellipse cx=«75» cy=«75» rx=«20» ry=«5» stroke=«red»
  fill=«transparent» stroke-width=«5»/>
```

де rx – радіус x еліпса, ry – радіус y еліпса, cx – положення x центру еліпса, cy – положення y центру еліпса.

Елемент **<line>** приймає положення двох точок як параметри та малює пряму лінію між ними.

```
<line x1=«10» x2=«50» y1=«110» y2=«150» stroke=«orange»
  stroke-width=«5»/>
```

де x1 – положення x точки 1, y1 – позиція y точки 1, x2 – позиція x точки 2, y2 – позиція y точки 2.

<polyline> – група сполучених прямих. Оскільки список може бути досить довгим, усі пункти включені в один атрибут:

```
<polyline points=«60 110 65 120 70 115 75 130 80 125 85 140
90 135 95 150 100 145» stroke=«orange» fill=«transparent»
stroke-width=«5»/>
```

де points – список пунктів. Кожне число має бути розділене пробілом, комою, символом закінчення рядка або символом переводу рядка з додатковими дозволеними пробілами. Кожна точка повинна містити два числа: координату x і координату y. Отже, список (0,0), (1,1), і (2,2) можна записати як 0, 0 1, 1 2, 2.

<polygon> подібний до <polyline> тим, що він складається з прямих відрізків, що з'єднують список точок. Однак для багатокутників контур автоматично з'єднує останню точку з першою, створюючи замкнуту форму.

```
<polygon points=«50 160 55 180 70 180 60 190 65 205 50 195 35
205 40 190 30 180 45 180» stroke=«green» fill=«transparent»
stroke-width=«5»/>
```

де points – список точок, кожна цифра розділена пробілом, комою, символом закінчення рядка або символом переводу рядка з додатковими дозволеними пробілами. Кожна точка повинна містити два числа: координату x і координату ay. Отже, список (0,0), (1,1), і (2,2) можна записати як 0, 0 1, 1 2, 2. Потім малюнок замикає шлях, тож остаточна пряма лінія буде проведена від (2,2)до (0,0).

<path> – найзагальніша форма, яку можна використовувати у SVG. За допомогою path елемента можна малювати прямокутники (із заокругленими кутами або без них), кола, еліпси, полілінії та багатокутники. В основному будь-які інші типи фігур, криві Безьє, квадратичні криві та багато іншого.

```
<path d=«M20,230 Q40,205 50,230 T90,230» fill=«none»
stroke=«blue» stroke-width=«5»/>
```

Існує один параметр, який використовується для керування його формою – d – список точок та інша інформація про те, як намалювати шлях.

Об'єкт **Path** дозволяє створювати фігури, до яких можуть бути використані всі атрибути оформлення – заливка, градієнт, штрихування. Він може представляти замкнуті та розімкнені фігури.

Фігура, утворена об'єктом Path, складається з прямих і кривих ліній чи дуг, які будуються послідовно, від точки до точки.

Існують спеціальні команди для малювання:

- **M/m** (moveTo) – встановити опорну точку. Указуються координати точки;
- **L/l** (lineTo) – провести лінію від поточної точки до заданої. Указуються координати заданої точки. Додана точка стає опорною;
- **H/h** (horizontalLineTo) – провести горизонтальну лінію від опорної точки до заданої. Виявляється координата **x** заданої точки. Координата **y** буде рівна ординаті поточної точки. Додана точка стає опорною;
- **V/v** (verticalLineTo) – провести вертикальну лінію від поточної точки до заданої. Указується координата **y** заданої точки. Координата **x** буде рівна абсцисі поточної точки. Додана точка стає опорною;
- **A/a** (elliptical arc) – провести еліптичну криву;
- **C/s, S/s** (кубічна крива Безьє) – провести кубічну криву Безьє;
- **Q/q, T/t** (квадратична крива Безьє) – провести квадратичну криву Безьє;
- **Z/z** (closePath) – завершити побудову об'єкта Path від опорної точки. Без параметрів.

Різняться абсолютні та відносні команди. При роботі з абсолютними командами всі координати задаються в системному малюнку, тобто початок відліку розміщений у верхньому лівому куті документа.При роботі з великою кількістю точок, більш зручним представляється використання відносних команд. При використанні відносних команд виявляється рядкова буква команди. Якщо фігура складається з кількох точок, то при відповідній формі

запису кожна попередня точка стає точкою відліку для наступної команди. Відповідні цьому випадку координати називаються відносними.

Намалюємо один і той же прямокутник

- із застосуванням абсолютних координат:

```
<svg xmlns=«http://www.w3.org/2000/svg» width=«200» height=«100»>
<path d=«M15,15 L185,15 L185,85 L15,85 Z» style=«fill:none; stroke:blue; stroke-width:3»/>
</svg>
```

із застосуванням відносних координат:

```
<svg xmlns=«http://www.w3.org/2000/svg» width=«200» height=«100»>
<path d=«M15,15 l170,0 L185,85 l-170,0 z» style=«fill:none; stroke:blue; stroke-width:3»/>
</svg>
```

У першому випадку із застосуванням абсолютних координат на початку задається опорна точка M15,15. Проводиться з неї лінія в точку з координатами L185,15. Далі додаються ще дві лінії – L185,85 і L15,85. Команда Z завершує шлях об'єкта, з'єднуючи відрізком точку 15, 185 з опорною точкою. У випадку із застосуванням відносних координат на початку задається опорна точка M15,15. З неї виводиться лінія в точку з тією ж самою ординатою і лежачою прямою на 170 пікселів за допомогою відносної команди l170,0. Далі за допомогою абсолютної команди L185,85 додається лінія. Точка з координатами 185,85 стає точкою початку відліку.

Команда A/a (дуга) тегу <path>

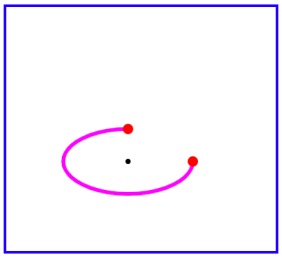
Еліптичні криві являють собою фрагменти еліпсів (дуги).

Рисування еліптичної кривої від опорної точки (mx, my) до заданої (x, y). Розмір і орієнтація еліпса задаються двома радіусами (rx, ry) і параметром x-axis-rotation, який визначає розташування еліпса як єдиного цілого відносно осі X поточної координатної системи. Центр еліпса вираховується автоматично на основі заданих координат точок і радіусів. Параметри large-arc-flag і sweep-flag визначають зовнішній вигляд еліпса (рис. 5.2).

Рис. 5.2. Параметри large-arc-flag і sweep-flag, які визначають вигляд дуги при малюванні

Наприклад, намалюємо 3/4 еліпс. Значення пари large-arc-flag, sweep-flag: 1,0. Для формування зображення про розміри самого документа SVG додано рамку.

```
<path d=«M100,100 A50,25 0 1,0
150,125» stroke-width=«3»
stroke=«magenta» fill=«none»/>
```



Команда C/c (кубічна крива) тегу <path>

Кубічна крива, C, є трохи складнішою кривою. Cubic Béziers приймає дві контрольні точки для кожної точки. Тому, щоб створити куб Безьє, потрібно вказати три набори координат :

```
C x1 y1, x2 y2, x y (абсолютні координати)
(or)
c dx1 dy1, dx2 dy2, dx dy (відносні координати)
```

Останній набір координат тут (x, y) визначає, де має закінчуватися лінія. Дві інші є контрольними. (x1, y1) – контрольна точка початку кривої, а (x2, y2) – контрольна точка кінця. Контрольні точки по суті описують нахил лінії, що починається в кожній точці. Потім функція Безьє створює плавну криву, яка переходить від нахилу, встановленого на початку лінії, до нахилу в іншому кінці.

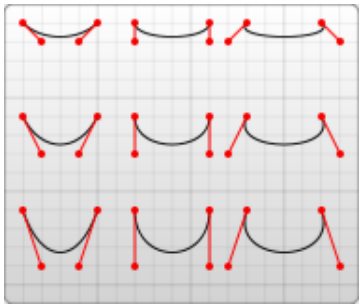


Рис. 5.3. Параметри, які визначають вигляд кубічної дуги при малюванні (код нижче)

```
<svg width=«190» height=«160»
xmlns=«http://www.w3.org/2000/svg»>
<path d=«M 10 10 C 20 20, 40 20, 50 10» stroke=«black»
fill=«transparent»/>
<path d=«M 70 10 C 70 20, 110 20, 110 10» stroke=«black»
fill=«transparent»/>
<path d=«M 130 10 C 120 20, 180 20, 170 10»
stroke=«black» fill=«transparent»/>
<path d=«M 10 60 C 20 80, 40 80, 50 60» stroke=«black»
fill=«transparent»/>
```

```
    <path d=«M 70 60 C 70 80, 110 80, 110 60» stroke=«black»
fill=«transparent»/>
    <path d=«M 130 60 C 120 80, 180 80, 170 60»
stroke=«black» fill=«transparent»/>
    <path d=«M 10 110 C 20 140, 40 140, 50 110»
stroke=«black» fill=«transparent»/>
    <path d=«M 70 110 C 70 140, 110 140, 110 110»
stroke=«black» fill=«transparent»/>
    <path d=«M 130 110 C 120 140, 180 140, 170 110»
stroke=«black» fill=«transparent»/>
    </svg>
```

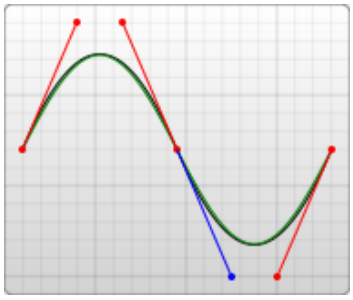
Команда S/s (кубічна крива) тегу <path>

Кілька кривих Безьє можна з’єднати разом, щоб створити розширені гладкі форми. Часто контрольна точка з одного боку точки буде відображенням контрольної точки, яка використовується з іншого боку, щоб підтримувати постійний нахил. У цьому випадку можна використати скорочену версію кубічного Безьє, позначену командою S(або s).

```
S x2 y2, x y (абсолютні координати)
(or)
s dx2 dy2, dx dy (відносні координати)
```

S створює криву того самого типу, що й раніше, але якщо вона слідує за іншою S командою або C командою, вважається, що перша контрольна точка є відображенням тієї, що використовувалася раніше. Якщо S команда не слідує за іншою командою S або C, то поточна позиція курсора використовується як перша контрольна точка.

```
<svg width=«190» height=«160»
xmlns=«http://www.w3.org/2000/svg»>
<path d=«M 10 80 C 40 10, 65 10,
95 80 S 150 150, 180 80»
stroke=«black» fill=«transparent»/>
</svg>
```



Команда Q/q (квадратична крива) тегу <path>

Інший тип кривої Безьє, квадратична крива, яка називається Q, насправді є простішою кривою, ніж кубічна. Для цього потрібна одна контрольна точка, яка визначає нахил кривої як у початковій, так і в кінцевій точках. Він приймає два параметри: контрольну точку та кінцеву точку кривої.

```
Q x1 y1, x y (абсолютні координати)
(or)
q dx1 dy1, dx dy(відносні координати)
```

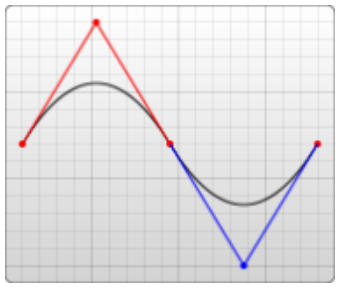
```
<svg width=«190» height=«160»
xmlns=«http://www.w3.org/2000/svg»>
<path d=«M 10 80 Q 95 10 180 80»
stroke=«black» fill=«transparent»/>
</svg>
```

Команда T/t (квадратична крива) тегу <path>

Як і у випадку з кубічною кривою Безьє, існує скорочення для об’єднання кількох квадратичних кривих Безьє, яке називається T.

```
T x y (абсолютні координати)
(or)
t dx dy (відносні координати)
```

```
    <svg width=«190» height=«160»
xmlns=«http://www.w3.org/2000/svg»>
      <path d=«M 10 80 Q 52.5 10, 95
80 T 180 80» stroke=«black»
fill=«transparent»/>
    </svg>
```



Особливості підготовки та оптимізації svg-ілюстрації для впровадження у код сторінки

Якщо відкрити будь-який файл, створений у різних версіях Adobe Illustrator, Figma або інших редакторах можна побачити дуже багато «зайвих» елементів, атрибутів, які не впливають ніяким чином на відображення самої ілюстрації на сторінці, а тільки перевантажують як візуально, так і фізично код сторінки. Тому такі елементи варто видаляти. Серед них можна спостерігати наступні: атрибут id часто додається ілюстратором або іншим редактором. він може стати корисним, якщо ви використовуєте css і javascript. в інших випадках можна вилучити; атрибут enable-background використовується для вказівки фону, але браузер не підтримує його – тож можна видалити; атрибути width і height також можна видалити здебільшого, але краще залишати, щоб не отримати сюрпризів. оголошувати їх можна у будь-якому порядку; атрибут xml:space=«preserve» не підтримується браузерами – значить, і його видаляємо; інші.

Щоб вручну не видаляти купу сміття з файлу, можна скористатися готовим інструментом онлайн SVGOMG – SVGO's Missing GUI . Останні версії Illustrator та Sketch створюють досить чистий SVG. Inkscape також має опцію Plain SVG у списку підтримуваних форматів.

Також при малюванні фігур слід стежити за значеннями атрибутів (зокрема тегу path). Бажано, щоб вони мали лише одне число після коми, а краще взагалі були цілими. Дотримуватись цього правила не обов'язково, але воно

зменшить розмір файлу, спростить подальшу обробку та візуально скоротить обсяг інформації.

Наприклад,

```
<path d=«M 17.7 29 C 28.2 12.9 47 5.6 62.8 10.4 c 28.2 8.5 30 50.5 24.8 53.1 c -2.6 1.3 -10.4 -6.1 -29.2 -34.6»/>
```

та

```
<path d=«M 17.651 28.956 c 10.56 -16.04 29.351 -23.359 45.12 -18.589 c 28.151 8.516 29.957 50.5 24.841 53.063 c -2.631 1.318 -10.381 -6.148 -29.235 -34.643»/>
```

- одна й та сама крива, але в першому випадку одна цифра після коми, а в другому три. Ця крива має всього 4 точки, а другий приклад на третину довше першого. Тому такого варто уникати.

Також досить важливо при підготовці файлів усі елементи, шари у графічних редакторах називати логічно, прив'язуючи назву до конкретного місця ілюстрації. Це надалі полегшить обробку зображення. Важливо використовувати у назві виключно латинські літери.

Якщо підготовка svg-файлу відбувається у середовищі Adobe Illustrator, то правильніше зберігати елемент через дію «Зберегти як», аніж «Експортувати як», оскільки перший варіант збереження містить розширені можливості впливу на остаточний вигляд коду ілюстрації, а також не привласнює елементам зайвих (інколи недоречних) атрибутів.

Хід роботи

1. Відповідно до обраної теми вебсторінка згідно з попередніми роботами, використовуючи графічний редактор Figma (можна інший), створіть три векторних зображення – набір іконок.

Рекомендується створювати прості іконки, які мають обведення та можуть мати заливку.

2. Згідно з рекомендаціями, наведеними у пункті Особливості підготовки та оптимізації svg-ілюстрації для впровадження у код сторінки збережіть Ваші ілюстрації та оптимізуйте їх. Опишіть, що саме було оптимізовано і як це вплинуло на загальний обсяг файлів та на їх якість.

3. Впровадьте Ваші ілюстрації у код вебсторінки, використовуючи три різних методи – через <img>, через <object>, <embed>, <iframe> (на вибір щось одне) та за допомогою вбудовування зображення у код сторінки через тег <svg>. Опишіть, які переваги та недоліки застосування кожного із методів впровадження svg графіки на вебсторінку*(3 можливих балів)*.

Контрольні запитання

1. Що таке властивість svg? Які особливості цього формату?
2. Які основні форми застосовуються при створенні svg зображення?
3. Опишіть основні структурні елементи svg зображення?
4. Які типи кривих можна задавати через тег <path>?
5. Що таке оптимізація зображень? Які особливості оптимізації svg графіки?

КП 6. МЕТОДИ ТРАНСФОРМАЦІЇ ЕЛЕМЕНТІВ НА ВЕБСТОРІНЦІ. СТВОРЕННЯ ТРАНСФОРМАЦІЇ ЕЛЕМЕНТІВ ПРИ НАВЕДЕННІ.

Мета роботи – ознайомитися із методами трансформації елементів на сторінці, навчитися використовувати основні сценарії властивості transform для реальних задач.

Основні теоретичні відомості

Властивість transform дозволяє застосовувати 2D або 3D трансформацію до елементу. Ця властивість дозволяє обертати, масштабувати, переміщувати, нахиляти елемент тощо.

Властивість transform отримує як значення «трансформуючі функції». Ці функції виглядають як scale(), translateX(), rotate() тощо. Вони приймають параметри для визначення ступеня трансформації.

Можна задавати декілька функцій одночасно. Якщо ця властивість приймає значення відмінне від none – створюється новий контекст накладення.

```
transform: none|transform-functions|initial|inherit;
```

Властивість transform може отримувати 24 значення:

Значення	Опис
`none`	Скасовує трансформацію. Без задання.
`matrix(n, n, n, n, n, n)`	2D перетворення, за допомогою матрицю з шести значень.
`matrix3d (n, n, n, n, n, n, n, n, n, n, n, n, n, n, n, n)`	3D перетворення, за допомогою матриці 4x4 з 16 значень.
`translate(x, y)`	2D зсув по горизонталі та вертикалі.
`translate3d(x, y, z)`	3D зсув за трьома осями.
`translateX(x)`	2D зсув по горизонталі (вісь X). Дозволені від'ємні значення. Додатне значення зсовує вправо, від'ємне вліво.
`translateY(y)`	2D зсув по вертикалі (вісь Y). Дозволені від'ємні значення. Додатне значення зсовує вниз, від'ємне вверх.
`translateZ(z)`	3D зсув по вісі Z. Дозволені від'ємні значення. Додатне значення зсовує вправо, від'ємне вліво.

Значення	Опис
`scale(x, y)`	Масштабування елемента по горизонталі та вертикалі. Значення менше ніж 1 – зменшує елемент, більше – збільшує.
`scale3d(x, y, z)`	Масштабування елемента тривимірному просторі.
`scaleX(x)`	Масштабування елемента по горизонталі (вісь X).
`scaleY(y)`	Масштабування елемента по вертикалі (вісь Y).
`scaleZ(z)`	Масштабування елемента по вісі Z.
`rotate(angle)`	Обертає елемент, на заданий кут, у двомірному просторі.
`rotate3d(x, y, z, angle)`	Обертає елемент в тривимірному просторі.
`rotateX(angle)`	3D обертання елемента по вісі X.
`rotateY(angle)`	3D обертання елемента по вісі Y.
`rotateZ(angle)`	3D обертання елемента по вісі Z.
`skew(x-angle, y-angle)`	2D перекіс за двома осями X – та Y (по горизонталі та вертикалі).
`skewX(angle)`	2D перекіс по горизонталі (вісь X).
`skewY(angle)`	2D перекіс по вертикалі (вісь Y).
`perspective(n)`	Перспективний вид для 3D трансформованого елемента.
`initial`	Встановлює властивість у значення без задання
`inherit`	Вказує на спадковість властивості від свого батьківського елемента (якщо відповідна властивість встановлена).

За допомогою списку, розділеного **пробілами**, можна додати кілька значень до transform властивості. Однак створюючи перетворення з кількома функціями, важливо мати на увазі, що браузер застосовуватиме значення в

порядку зліва направо. Це означає, що наступні перетворення візуально матимуть різні результати, попри те саме значення.

Хід роботи

1. Створіть сторінку з контейнером, у який вкладіть 5 блоків (divs), задайте їм ширину та висоту та різний колір.

2. Продублюйте створений контейнер із вкладеними у нього п’ятьма блоками та застосуйте до кожного із блоків трансформацію згідно з вашим варіантом* (табл. 1.1) за допомогою відповідних методів трансформацій.

Таблиця 6. 1. – Варіанти завдання

Номер варіанту	Перший блок	Другий блок	Третій блок	Четвертий блок	П’ятий
1	Пропорційно збільшити у 2 рази	Змістіть блок по горизонталі вправо на 50px	Поверніть блок по осі X на кут 30^0	Зробіть перекіс елемента по вертикалі на самостійно визначену величину	Застосуйте до блоку усі трансформації, які були застосовані до 1, 2 та 3 блоків
2	Збільшити ширину на 15%, висота – незмінна	Змістіть блок по вертикалі вниз на 50px	Поверніть блок проти годинникової стрілки на кут 30^0		
3	Зменшити пропорційно вдвічі	Змістіть блок по горизонталі вліво на 50px	Поверніть блок за годинниковою стрілкою на 1,5 оберту		
4	Зменшити висоту на 25%, ширина незмінна	Змістіть блок по горизонталі вправо на 50px, по вертикалі – вниз на 40px	Поверніть блок проти годинникової стрілки на 1,5 оберту	Зробіть перекіс по горизонталі на самостійно визначену величину	
5	Збільшити весь блок у 1,5 раза	Змістіть блок по горизонталі вліво на 50px, по вертикалі – вверх на 40px	Поверніть блок по осі Y на 40 радіан		
6	Зменшити висоту вдвічі, збільшити	Змістіть блок по вертикалі вверх на 50px	Поверніть блок по осі Z на 60 градусів		

	ширину вдвічі				

**Варіант на завдання визначається згідно зі списком групи, де 1, 7, 13, 19, 25 людина за списком – виконує 1 варіант; 2, 8, 14, 20, 26 людина за списком – виконує 2 варіант; 3, 9, 15, 21, 27 людина за списком – виконує 3 варіант; 4, 10, 16, 22, 28 людина за списком – виконує 4 варіант; 5, 11, 17, 23, 29 людина за списком – виконує 5 варіант; 6, 12, 18, 24, 30 – виконує 6 варіант.*

3. Згідно прототипу сторінки (вайфрейму) типової сторінки лендінг-пейдж, представленої у КП1 (рис. 1.2), продовжуючи тематику Вашого проєкту, створіть три картки (рис. 6.1), наповнивши ці картки зображеннями та текстом. При цьому створіть ефект появи тексту «card text» поверх зображення із-за меж картки при наведенні миші на конкретну картку. Зображення картки «card image» також може підлягати трансформаціям.

Підказка: для приховання елементів, які знаходяться поза межами картки використовуйте властивість overflow зі значенням hidden до блоку картки. Для переміщення текстового блоку можна використати властивість translate(). Текстовий блок може позиціюватися абсолютно у межах відносно позиційованої картки, та бути розміщеним поверх зображення з відповідним значенням z-index.

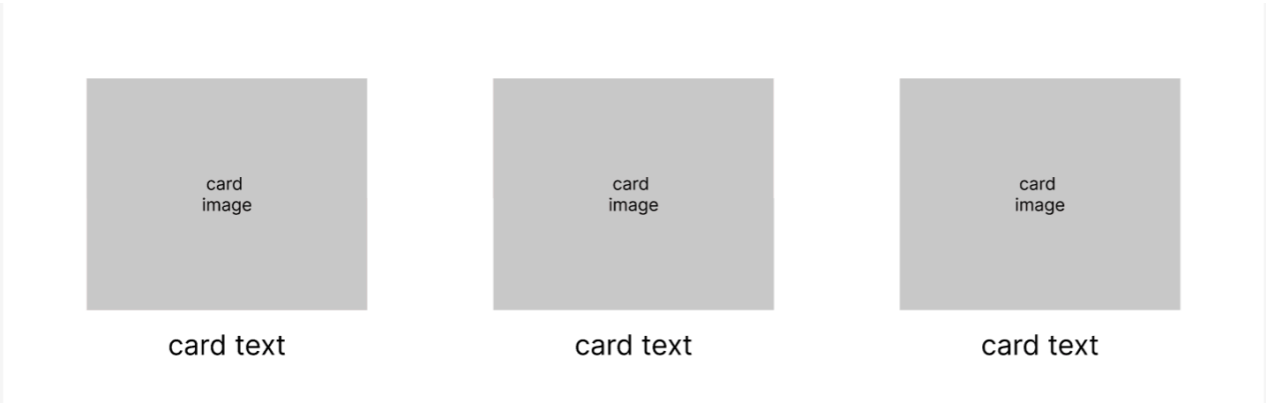


Рис. 6.1. Вайфрейм до завдання 2 КП6

Контрольні запитання

1. Що таке властивість transform? Які методи трансформацій Вам відомі, назвіть основні сценарії їх використання?

2. Як змістити елемент по вертикалі (від верху донизу)?

3. Як перевернути елемент на 2,5 оберти навколо центру?

4. Як задати обертання елементу навколо нижнього правого кута? Надайте варіанти, яким чином це можливо задати

5. Яким чином трансформуються елементи, якщо вказано кілька значень transform властивості?

КП 7. ЕФЕКТИ АНІМАЦІЇ ТА КЕРУВАННЯ НИМИ ЗА ДОПОМОГОЮ CSS ВЛАСТИВОСТЕЙ. СТВОРЕННЯ АНІМАЦІЇ ПОЯВИ ЕЛЕМЕНТІВ СТОРІНКИ ПРИ ЇЇ ЗАВАНТАЖЕННІ

Мета роботи – ознайомитися з основними ефектами анімації елементів на сторінці, навчитися використовувати CSS властивості animation для створення анімації на сторінці.

Основні теоретичні відомості

Анімація – це чудовий спосіб підкреслити інтерактивні елементи та додати інтересу та цікавості вашим дизайнам. Анімація дозволяє елементу поступово змінювати стиль з одного на інший.

Для створення анімації необхідні 3 основні складники:

- Ключові кадри – мають бути оголошені та містити стилі або вигляд об’єкта;
- Властивості – характеристики анімації, пов’язані з ключовим кадром;
- Тривалість – час, необхідний для завершення всієї процедури.

Ключові кадри – це механізм, який використовується для призначення станів анімації часовим міткам на часовій шкалі. Правило `@keyframes` встановлює ключові кадри при анімації елемента – це властивості елемента (прозорість, колір, положення та ін.), які повинні застосовуватися до елементу в заданий момент часу. Для створення анімації спершу потрібно описати правило ключових кадрів. Перша частина правила, на яку слід звернути увагу, – це

користувацький ідентифікатор (власний ідентифікатор) – або назва правила ключових кадрів. Ідентифікатор наступного правила my-animation:

```
@keyframes  my-animation  {
        from {transform: translateY(20px);}
        to {transform: translateY(0px);}
}
```

Спеціальний ідентифікатор працює як ім’я функції, тому дозволяє посилатися на правило ключових кадрів в іншому місці коду CSS. Можна додати скільки завгодно позицій на таймфреймі.

Щоб анімація працювала, необхідно прив’язати анімацію до елемента шляхом вказування у перелік властивостей елемента імені анімації (**animation-name**), який відповідає ідентифікатору правила ключових кадрів, та вказати тривалість **(animation-duration)** анімації.

Властивість **animation-name** визначає назву для анімації @keyframes.

```
animation-name: keyframename|none|initial|inherit;
```

Значення	Опис
keyframename	Визначає назву ключового кадру, який потрібно прив’язати до селектора
none	Значення за замовчуванням. Вказує, що анімації не буде (можна використовувати для перевизначення анімацій, що надходять із каскаду)
initial	Встановлює для цієї властивості значення за умовчанням
inherit	Успадковує цю властивість від батьківського елемента.

Властивість **animation-duration** визначає, скільки часу має тривати анімація для завершення одного циклу.

```
animation-duration: time|initial|inherit;
```

Значення	Опис
time	Визначає тривалість часу, який має знадобитися для завершення одного циклу анімації. Це можна вказати в секундах або мілісекундах. Значення за замовчуванням – 0, що означає, що анімації не буде
initial	Встановлює для цієї властивості значення за умовчанням
inherit	Успадковує цю властивість від батьківського елемента.

Визначає **animation-timing-function** криву швидкості анімації.

Крива швидкості визначає ЧАС, за який анімація змінює один набір стилів CSS на інший. Крива швидкості використовується для плавного внесення змін.

```
animation-timing-function: linear|ease|ease-in|ease-out|ease-
in-out|step-start|step-end|steps(int, start|end)|cubic-
bezier(n, n, n, n)|initial|inherit;
```

Значення	Опис
linear	Анімація має однакову швидкість від початку до кінця
ease	Значення за замовчуванням. Анімація починається повільно, потім швидко, а потім повільно закінчується
ease-in	Анімація має повільний початок
ease-out	Анімація має повільний кінець

Значення	Опис
ease-in-out	Анімація має як повільний початок, так і повільний кінець
step-start	Еквівалент до steps(1, start)
step-end	Еквівалент до steps(1, end)
steps(int, start|end)	Визначає покрокову функцію з двома параметрами: перший – визначає кількість інтервалів у функції, це має бути додатне ціле число (>0); другий, який є необов'язковим, – є значенням «початок» або «кінець» і вказує точку, в якій відбувається зміна значень у межах інтервалу. Якщо другий параметр опущено, йому надається значення «end»
cubic-bezier(*n*, n, n, n)	Визначте власні значення у функції кубічного Безьє. Можливими значеннями є числові значення від 0 до 1
initial	Встановлює для цієї властивості значення за умовчанням
inherit	Успадковує цю властивість від батьківського елемента.

Властивість **`animation-delay`** визначає затримку для початку анімації. Значення затримки анімації визначається в **секундах (с)** або **мілісекундах (мс)**.

```
animation-delay: time|initial|inherit;
```

Значення	Опис
time	Визначає кількість секунд (с) або мілісекунд (мс) очікування перед початком анімації. Значення за замовчуванням – 0. Допускаються від'ємні значення. Якщо ви використовуєте від'ємні значення, анімація розпочнеться так, ніби вона вже відтворювалася N с/мс.
initial	Встановлює для цієї властивості значення за умовчанням
inherit	Успадковує цю властивість від батьківського елемента.

Властивість **animation-iteration-count** визначає кількість разів відтворення анімації.

```
animation-iteration-count: number|infinite|initial|inherit;
```

Значення	Опис
number	Число, яке визначає, скільки разів має відтворюватися анімація. Значення за замовчуванням 1
infinite	Вказує, що анімацію слід відтворювати нескінченну кількість разів (назавжди)
initial	Встановлює для цієї властивості значення за умовчанням
inherit	Успадковує цю властивість від батьківського елемента.

Властивість **animation-direction** визначає, чи має відтворюватися анімація вперед, назад або по черзі.

```
animation-direction: normal|reverse|alternate|alternate-
reverse|initial|inherit;
```

Значення	Опис
normal	Значення за замовчуванням. Анімація відтворюється як зазвичай (вперед)
reverse	Анімація відтворюється у зворотному напрямку (назад)
alternate	Анімація відтворюється спочатку вперед, потім назад
alternate-reverse	Анімація відтворюється спочатку назад, потім вперед
initial	Встановлює для цієї властивості значення за умовчанням
inherit	Успадковує цю властивість від батьківського елемента.

Властивість **animation-fill-mode** визначає стиль для елемента, коли анімація не відтворюється (перед її початком, після її закінчення або обидва). CSS-анімація не впливає на елемент до відтворення першого ключового кадру або після відтворення останнього ключового кадру. Властивість animation-fill-mode може перевизначати цю поведінку.

```
animation-fill-mode:
none|forwards|backwards|both|initial|inherit;
```

Значення	Опис
none	Значення за замовчуванням. Анімація не застосовуватиме стилі до елемента до або після його виконання
forwards	Елемент збереже значення стилю, встановлені останнім ключовим кадром (залежить від animation-direction і animation-iteration-count)
backwards	Елемент отримає значення стилю, встановлені першим ключовим кадром (залежить від напрямку анімації), і збереже це протягом періоду затримки анімації
both	Анімація відповідатиме правилам, встановленим і першим, і останнім ключовим кадром, розширюючи властивості анімації в обох напрямках
initial	Встановлює для цієї властивості значення за умовчанням
inherit	Успадковує цю властивість від батьківського елемента.

Властивість **animation-play-state** визначає, чи працює анімація, чи призупинено.

```
animation-play-state: paused|running|initial|inherit;
```

Значення	Опис
paused	вказує на те, що анімацію призупинено
running	Значення за замовчуванням. Вказує, що анімація запущена
initial	Встановлює для цієї властивості значення за умовчанням
inherit	Успадковує цю властивість від батьківського елемента.

Замість того, щоб визначати всі властивості окремо, ви можете визначити їх в **`animation`** скороченому вигляді, що дає змогу визначати властивості анімації в такому порядку:

```
animation: animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, animation-direction, animation-fill-mode, animation-play-state
```

Хід роботи

1. Створіть сторінку з контейнером розміром 300x300px, у який вкладіть 1 блок (divs), та згідно з вашим варіантом* повторіть анімацію за допомогою властивостей css (табл. 7.1).

Підказкою для опису ключових кадрів може слугувати файл Figma, у середовищі якої були розроблені варіанти анімацій. Для перегляду анімацій необхідно під'єднати плагін Figmotion, та обравши фрейм, що відповідає вашому варіанту, запустити плагін.

2. Використовуючи результати лабораторної роботи 2, а саме позиційовані елементи композиції зображення першого екрану (First screen image), запроєктуйте анімацію появи цього зображення. А саме опишіть покроково, яким чином будуть з’являтися елементи зображення.

Таблиця 7. 1. – Варіанти завдання

Номер варіанту	Завдання	Номер варіанту	Завдання
1		2	
3		4	
5		6	

**Варіант на завдання визначається згідно зі списком групи, де 1, 7, 13, 19, 25 людина за списком – виконує 1 варіант; 2, 8, 14, 20, 26 людина за списком – виконує 2 варіант; 3, 9, 15, 21, 27 людина за списком – виконує 3 варіант; 4, 10, 16, 22, 28 людина за списком – виконує 4 варіант; 5, 11, 17, 23, 29 людина за списком – виконує 5 варіант; 6, 12, 18, 24, 30 – виконує 6 варіант.*

3. Закодуйте запроєктовану анімацію появи елементів композиції зображення першого екрану, використовуючи необхідні css властивості *(3 можливих балів)*.

Контрольні запитання

1. Що таке властивість animation? Синтаксис animation властивості.
2. Які основні складники для створення анімації?
3. Як зациклити анімацію (тобто створити постійне повторення анімації)?
4. Що таке ключові кадри?
5. Яка властивість використовується для зупинки анімації?
6. Як здійснити рух в інтервалах замість того, щоб рухатися вздовж кривої Безьє?

КП 8. АНІМУВАННЯ ВЕКТОРНИХ ІЛЮСТРАЦІЙ НА ВЕБСТОРІНЦІ

Мета роботи – ознайомитися з основними правилами створення, анімування властивостей штрихів векторних ілюстрацій у форматі svg та додавання їх на вебсторінку.

Основні теоретичні відомості

Як було зазначено вище, SVG (скорочено від Scalable Vector Graphics) – масштабована векторна графіка. Це унікальний тип формату зображення для векторної графіки, написаною розширюваною мовою розмітки (XML). Оскільки це файли XML, зображення SVG можна створювати та редагувати за допомогою будь-якого текстового редактора, але частіше вони створюються за допомогою програм для малювання.

Є два основні способи анімації SVG елемента:

- CSS анімація
- SMIL анімація, вбудована у SVG (насправді це анімація SVG, яка базується на SMIL і розширює його функціонал)

Інколи ці анімації називають «зовнішньою» та «внутрішньою». Даний поділ умовний, проте вони мають функціональні відмінності. Якщо говорити про переваги кожного із методів загалом: CSS – має найкращу підтримку браузерами; SMIL – має великий функціонал. Важко сказати, що використовувати краще, оскільки вони багато в чому схожі. SMIL є більш функціональним та складним інструментом. І саме тому його слід використовувати лише за необхідності. Якщо анімація проста і можна обійтися CSS, то так і слід зробити.

Анімація засобами CSS

Будь-який SVG елемент можна анімувати так само як ми це робимо з HTML. Всі анімації створюються за допомогою @keyframes. SVG документ має внутрішні таблиці стилів, тому можна скористатись наступною конструкцією для додавання анімаційних ефектів до конкретного svg-елемента:

```
<svg>
    <style>
        <!-- Тут анімація -->
    </style>
    <!—Тут svg елементи -->
</svg>
```

Анімувати атрибут SVG так само просто, як і CSS атрибути:

```
@keyframes change_fill {
    0% { fill: #49549E; }
    75% { fill: #1bceb1; }
    100% { fill: #1bce4f; }
}
```

Потім залишається просто застосувати створені анімації до потрібного елементу

```
.circle { animation: change_fill 1s}
```

Таким чином можна додати інтерактивності. Наприклад, змінити колір заливання елемента при наведенні курсора на нього :

```
.circle { fill: #49549E; transition: .3s; }
.circle:hover { fill: #1bceb1; }
```

Лінійна анімація

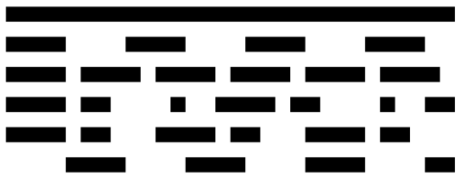
Лінійна анімація полягає в анімуванні властивостей штрихів (контур svg-елемента), зокрема анімації властивостей `stroke-dasharray` і `stroke-dashoffset.` Така анімація часто застосовується для імітування рукописного введення, самостійного малювання та ефекту »самовитирання».

Атрибут **`stroke-dasharray`** визначає шаблон штрихів і прогалин, які використовуються для малювання контуру фігури. Ви можете використовувати цей атрибут із такими елементами SVG: `<altGlyph>, <circle>, <ellipse>, <path>, <line>, <polygon>, <polyline>, <rect>, <text>, <textPath>, <tref>, <tspan>.`

`<dasharray>` – список розділених комами та/або пробілами значень (`<length>` або <percentage>),які вказують довжину тире та пропусків, що чергуються.

Якщо вказано непарну кількість значень, список значень повторюється, щоб отримати парну кількість значень. Таким чином, 5,3,2 еквівалентно 5,3,2,5,3,2.

Наприклад, Наступне зображення може бути реалізовано наступним чином:

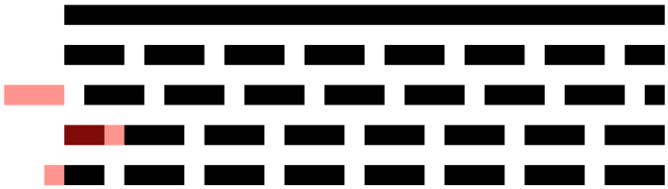


```
<svg viewBox=«0 0 30 12» xmlns=«http://www.w3.org/2000/svg»>
  <!-- No dashes nor gaps -->
  <line x1=«0» y1=«1» x2=«30» y2=«1» stroke=«black» />

  <!-- Dashes and gaps of the same size -->
  <line x1=«0» y1=«3» x2=«30» y2=«3» stroke=«black» stroke-
dasharray=«4»/>

  <!-- Dashes and gaps of different sizes -->
  <line x1=«0» y1=«5» x2=«30» y2=«5» stroke=«black» stroke-
dasharray=«4 1»/>

  <!-- Dashes and gaps of various sizes with an odd number of
values -->
  <line x1=«0» y1=«7» x2=«30» y2=«7» stroke=«black» stroke-
dasharray=«4 1 2»/>

  <!-- Dashes and gaps of various sizes with an even number
of values -->
  <line x1=«0» y1=«9» x2=«30» y2=«9» stroke=«black» stroke-
dasharray=«4 1 2 3»     />

  <!-- Dashes starting with a gap -->
  <line x1=«0» y1=«11» x2=«30» y2=«11» stroke=«black» stroke-
dasharray=«0 4 0» />
</svg>
```

Атрибут **stroke-dashoffset** визначає зміщення під час відтворення пов'язаного dash array. Ви можете використовувати цей атрибут із такими елементами SVG: <altGlyph>, <circle>, <ellipse>, <path>, <line>, <polygon>, <polyline>, <rect>, <text>, <textPath>, <tref>, <tspan>.

Зміщення зазвичай виражається в одиницях вимірювання користувача, pathLength, але якщо використовується <percentage>, значення розрізняється як відсоток від поточного вікна перегляду (viewport).

Наступне зображення може бути реалізовано наступним чином:



```
<svg viewBox=«-3 0 33 10» xmlns=«http://www.w3.org/2000/svg»>
  <!-- No dash array -->
  <line x1=«0» y1=«1» x2=«30» y2=«1» stroke=«black» />

  <!-- No dash offset -->
  <line x1=«0» y1=«3» x2=«30» y2=«3» stroke=«black» stroke-
dasharray=«3 1» />

  <!-- The start of the dash array computation is pulled by 3
user units -->
  <line x1=«0» y1=«5» x2=«30» y2=«5» stroke=«black» stroke-
dasharray=«3 1» stroke-dashoffset=«3» />
```

```
  <!--The start of the dash array computation is pushed by 3 user units-->
  <line x1=«0» y1=«7» x2=«30» y2=«7» stroke=«black» stroke-dasharray=«3 1» stroke-dashoffset=«-3» />

  <!-- The start of the dash array computation is pulled by 1 user units which ends up in the same rendering as the previous example-->
  <line x1=«0» y1=«9» x2=«30» y2=«9» stroke=«black» stroke-dasharray=«3 1» stroke-dashoffset=«1» />

  <!-- the following red lines highlight the offset of the dash array for each line -->
  <path d=«M0,5 h-3 M0,7 h3 M0,9 h-1» stroke=«rgba(255,0,0,.5)» />
</svg>
```

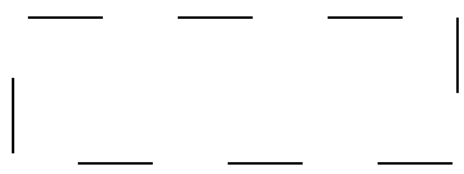
Таким чином, стає зрозуміло, що якщо змінювати значення цих властивостей у певному проміжку часу, можна досягти анімування контуру зображення.

Наприклад, маємо фігуру – прямокутник :

```
<svg width=«180px» height=«60px» viewBox=«0 0 180 60»>
  <polyline points=«179,1 179,59 1,59 1,1 179,1» class=«bg-line» />
  <polyline points=«179,1 179,59 1,59 1,1 179,1» class=«hl-line» />
</svg>
```

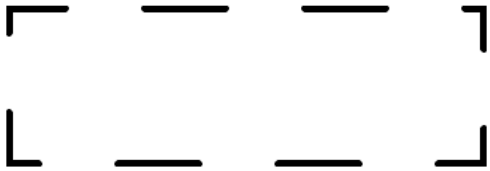
Привласнивши певний клас, та надавши цьому прямокутнику певних властивостей (а саме: прибрали заливання (`fill`) та створили обведення (`stroke`) і розбили це обведення на штрихи з проміжками (`stroke-dasharray`) розміром 30 одиниць), отримаємо наступний результат (можемо додати властивості або у svg, або в окремо винесеному css описі):

```
<svg width=«180px» height=«60px» viewBox=«0 0 180 60»
class=«border»>
    <style>
        .border {
            fill: none;
            stroke: #000;
            stroke-dasharray: 30;
            }
    </style>
 <polyline points=«179,1 179,59 1,59 1,1 179,1» class=«bg-
line» />
 <polyline points=«179,1 179,59 1,59 1,1 179,1» class=«hl-
line» />
</svg>
```



Можна збільшити товщину обведення та заокруглити штрихи, додавши властивості `stroke-width: 3;` та `stroke-linecap: round;`, то отримаємо наступне:

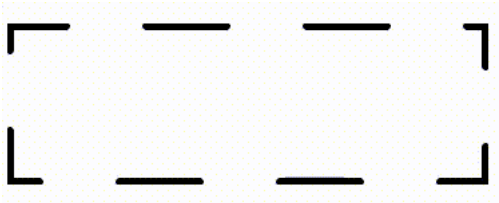
Тепер можемо змістити наші штрихи на 15 одиниць (`stroke-dashoffset: 15;`), то отримаємо наступне:



Це нам підказує, що ми можемо рухати ці штрихи по контуру, а, отже, ми можемо зробити анімацію цього руху. Описавши анімацію, а саме відступ штрихів (`stroke-dashoffset`) на певне значення, та додавши цю анімацію до svg-зображення, матимемо такий результат:

```
<svg width=«180px» height=«60px» viewBox=«0 0 180 60» class=«border»>
        <style>
            .border {
                fill: none;
                stroke: #000;
                stroke-dasharray: 30;
                stroke-width: 3;
                stroke-linecap: round;
                animation: dash 5s linear;
                }
```

```
            @keyframes dash {
              to {
                stroke-dashoffset: 1000;
              }
            }
          </style>
      <polyline points=«179,1 179,59 1,59 1,1 179,1»
class=«bg-line» />
      <polyline points=«179,1 179,59 1,59 1,1 179,1»
class=«hl-line» />
    </svg>
```



А тепер надамо заливання прямокутника сірим кольором (`fill: #ccc;`) збільшимо довжину штриха до 480 (у цьому випадку це приблизно відповідає периметру прямокутника, або сумі довжині контуру) і проміжки між штрихами збільшимо до 480, тобто `stroke-dasharray: 480.` Анімацію при цьому не застосовуємо. Отримаємо наступний результат:

Це відобразить контур контуру як безперервний шлях. Поекспериментуйте зі значенням і подивіться, що станеться з контуром.

Змістимо наші штрихи на 480 (`stroke-dashoffset: 480;`) – тобто на величину проміжку між штрихами. Властивість stroke-dashoffset – це той

атрибут (властивість), який визначає розташування точки уздовж шляху SVG, де почнеться штрих штриха.

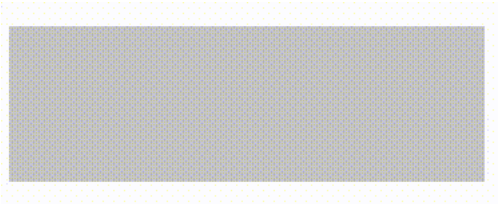
Отримаємо наступний результат:

Бачимо, що візуально контур зник, але у цьому випадку це зсунутий штрих контуру, який попередньо обводив прямокутник, утворюючи контур, бо довжина штриха дозволяла повністю «обгортати» прямокутник. Він зсунутий на таке значення, при якому по периметру прямокутника знаходиться проміжок контуру, який теж рівний 480. Це і створює такий візуальний ефект відсутності контуру.

Це наштовхує нас на думку, що ми можемо створити анімацію появи цього контуру, шляхом зміни властивості `stroke-dashoffset` до початкового значення, тобто до 0. Перевіримо:

```
<svg width=«180px» height=«60px» viewBox=«0 0 180 60» class=«border»>
        <style>
            .border {
                fill: #ccc;
                stroke: #000;
                stroke-dasharray: 480;
                stroke-width: 3;
                stroke-linecap: round;
                stroke-dashoffset: 480;
                animation: dash 5s linear forwards;
```

```
            }
          @keyframes dash {
            to {
              stroke-dashoffset: 0;
            }
          }
        </style>
      <polyline points=«179,1 179,59 1,59 1,1 179,1»
class=«bg-line» />
      <polyline points=«179,1 179,59 1,59 1,1 179,1»
class=«hl-line» />
    </svg>
```



Таким чином, ми досягли ефекту плавного обведення контуру. Цей ефект може бути застосований до різних об’єктів. **Головне, щоб ці об’єкти мали обведення (stroke).**

При створенні такої анімації стає важливим правильна підготовка svg-графіки. По-перше, графіка має бути оптимізована (див. теоретичні відомості до КП 5). По-друге, при її створенні необхідно контролювати початкову точку, із якої починається контур. Це полегшить подальші маніпуляції із контуром.

Надзвичайно важливим при цьому стає визначення довжини контуру. Досить часто цей крок здійснюється методом проб і помилок. Для цього можна скористатися декількома способами:

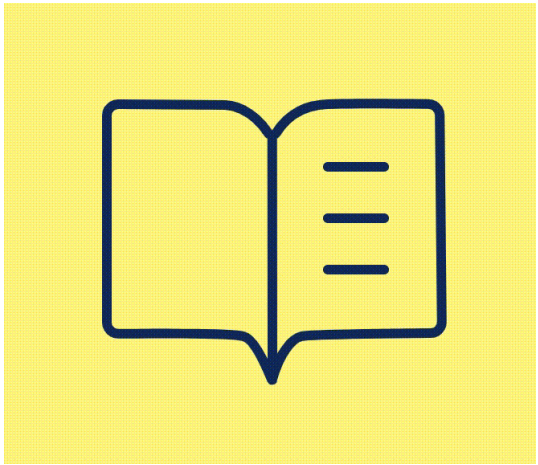
1) У графічному редакторі перетворити обведення у штрихове і змінювати довжину штриха до необхідного значення (мінус – не у всіх графічних редакторах спрацьовує);

2) Створену та вмонтовану у код svg-графіку переглядати через інспектора коду браузера та змінюючи значення властивості `stroke-dasharray` підібрати необхідну довжину штриха;

3) Використати JavaScript для знаходження довжини контуру, переглянувши її через консоль (мінус – потребує знань JS):

```
<script>
…
  var path = document.querySelector('svg path'),
    length = path.getTotalLength();
  console.log(length);
…
</script>
```

На основі отриманих знань можна легко створити такий ефект при наведенні (додаток 1):

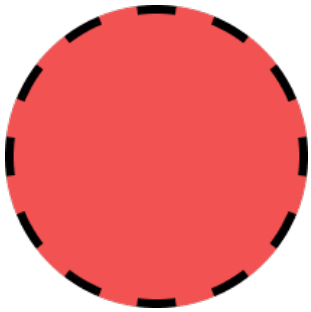
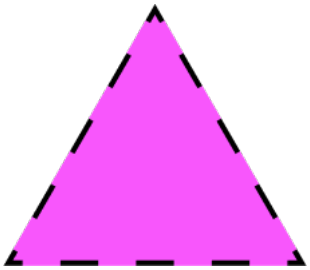
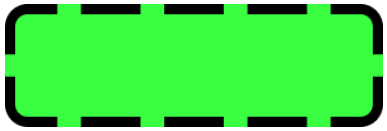
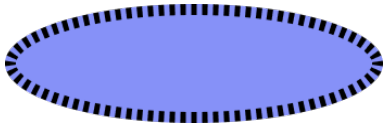


Хід роботи

1. Згідно з варіантом (табл. 8.1) створіть svg-зображення, та анімуйте рух штрихів по контуру вашого зображення.

2. Використовуючи результати КП 5, а саме одну зі створених іконок, створіть анімацію появи контуру зображення і заливання зображення (за прикладом, наведеним у теоретичних відомостях і додатку 1).

Таблиця 8. 1. – Варіанти завдання

Номер варіанту	Завдання	Номер варіанту	Завдання
1		2	
3		4	
5		6	

**Варіант на завдання визначається згідно зі списком групи, де 1, 7, 13, 19, 25 людина за списком – виконує 1 варіант; 2, 8, 14, 20, 26 людина за списком – виконує 2 варіант; 3, 9, 15, 21, 27 людина за списком – виконує 3 варіант; 4, 10, 16, 22, 28 людина за списком – виконує 4 варіант; 5, 11, 17, 23, 29 людина за списком – виконує 5 варіант; 6, 12, 18, 24, 30 – виконує 6 варіант.*

Контрольні запитання

1. Що таке властивість svg формат ілюстрацій? Назвіть основні переваги його застосування.

2. Які методи анімації svg-зображень є найбільш поширеними?

3. Які атрибути (властивості) необхідно змінювати, щоб анімувати контур svg-зображення?

4. Де найчастіше застосовується анімація svg-зображень?

КП 9. ПІДГОТОВКА ТА ДОДАВАННЯ АУДІОКОНТЕНТУ НА ВЕБСТОРІНКУ

Мета роботи – ознайомитися з основними властивостями аудіоконтенту, та навчитися вставляти аудіовміст у вебсторінку.

Основні теоретичні відомості

З роками здатність Інтернету представляти, створювати та керувати медіафайлом зростала все більшими темпами. Сьогодні існує велика кількість доступних API, а також елементів HTML, інтерфейсів DOM та інших функцій, які дозволяють не лише виконувати ці завдання, але й використовувати медіа в тандемі з іншими технологіями, щоб робити справді надзвичайні речі.

Як правило, медіаформати, підтримувані браузером, повністю залежать від творців браузера, що може ускладнити роботу веброзробник.

Аудіоконтент (аудіовміст) – це матеріал, який споживається під час прослуховування. Музика, радіо та подкасти – чудові приклади цієї форми медіа.

Методи відтворення медіа:

- традиційний метод (не HTML5): браузерні плагіни або зовнішні застосунки;

- використання HTML5 <audio>: браузерні вбудовані плеєри (built-in player).

Оскільки не всі браузери підтримують усі аудіоформати, аудіофайл кодується/декодується за допомогою аудіокодека (цифрового електронного пристрою або програмного забезпечення на основі комп'ютера, яке допомагає стискати та розпаковувати цифрові аудіодані).

Визначаючи різні формати файлів, рекомендовано визначати тип MIME для кожного файлу, щоб дозволити браузеру локалізувати підтримуваний файл.

Найпопулярніші аудіоформати:

- MP3 (.mp3) – найпопулярніший аудіоформат, який використовує стиснення з втратами даних і дозволяє зменшити розмір файлу. Попри популярність серед користувачів, телекомпанії та радіостанції використовують більш сучасні кодеки ISO-MPEG, такі як AAC або MPEG-H;
- AAC (Advanced Audio Codec) (.m4a) – закритий кодек, аналог MP3, але в порівнянні з останнім забезпечує кращу якість при такому ж або більшому ступені стиснення;
- Ogg Vorbis (.ogg, .oga) – безплатний формат із відкритим кодом, підтримується у Firefox, Opera та Chrome. Забезпечує якісний звук, але недостатньо підтримується програвачами пристрою;
- WAV (.wav) – формат аудіофайлу, розроблений компаніями Microsoft та IBM. Формат WAV був частково витіснений стисненими форматами, проте, завдяки своїй простоті, надалі знаходить широке використання в процесі редагування звуку та на переносних аудіопристроях, як програвачі та цифрові диктофони.

Елемент <audio> використовується для відтворення аудіо в вебконтексті. Їх можна використовувати непомітно як місце призначення для складніших медіафайлів або з видимими елементами керування для керованого користувачем відтворення аудіофайлів. Доступні з JavaScript як HTMLAudioElement об'єкти.

Елемент HTML5 <audio> використовується для відтворення аудіофайлу на вебсторінці. Тут визначається <audio> елемент із кількома джерелами – робиться так, оскільки не всі браузери підтримують однакові аудіоформати.

Щоб забезпечити прийнятне покриття, необхідно вказати принаймні два різні формати. Два формати, які забезпечать максимальне покриття, це mp3 та ogg vorbis:

```
<audio controls>
    <source src=«horse.ogg» type=«audio/ogg»>
    <source src=«horse.mp3» type=«audio/mpeg»>
     <p> Your browser does not support HTML audio, but you can still <a href=«audiofile.mp3»>download the music</a>. </p>
</audio>
```

Елемент `<source>` дозволяє вказати альтернативні аудіофайли, які браузер може вибрати. Браузер використовуватиме перший розпізнаний формат.

`<source>` елемент приймає атрибути `src` та `type`:

- `src` містить шлях до аудіофайлу, який потрібно завантажити (відносний або абсолютний).
- `type` використовується для інформування браузера про тип файлу (MIME-тип). Про відповідність формату файлу його типу можна знайти в табл. 9.1.

Таблиця 9.1 – Типи медіа

File Format	**Media Type**
MP3	audio/mpeg
OGG	audio/ogg
WAV	audio/wav

Атрибут `controls` додає елементи керування звуком, як-от відтворення, пауза та гучність. Якщо не вказувати цей атрибут, елементи керування не

відображатимуться – і натомість доведеться створювати власні елементи керування та запрограмовувати їхню функціональність за допомогою Media API. Однак це може бути хорошим підходом, оскільки елементи керування за замовчуванням виглядають по-різному в різних браузерах. Таким чином, створення власних елементів керування забезпечує узгоджений вигляд елементів керування в усіх браузерах.

Текст між тегами `<audio>` і `</audio>` відображатиметься лише в браузерах, які не підтримують цей `<audio>` елемент. Отже, ідеальне місце для створення резервного варіанту або повідомлення про несумісність – це перед закривальним `</audio>` тегом.

Якщо вказати атрибут **`autoplay`**, аудіо почне відтворюватися якнайшвидше й без будь-якої взаємодії з користувачем – коротко кажучи, аудіо відтворюватиметься автоматично.

```
<audio controls autoplay>
      …
</audio>
```

Це значення часто ігнорується на мобільних платформах, і його використання не рекомендується, якщо це не є дійсно необхідним. Автоматичне відтворення аудіо (і відео) зазвичай дуже дратує. Крім того, вебпереглядач мають політики, які повністю блокують автоматичне відтворення в багатьох ситуаціях.

Якщо додати атрибут **`muted`** після `autoplay`, то аудіофайл почне відтворюватися автоматично, але без звуку:

```
<audio autoplay muted>
      …
</audio>
```

Атрибут **loop** гарантує, що після досягнення кінця аудіокліпу аудіокліп повертатиметься на початок і відтворюватиметься знову.

```
<audio autoplay loop>
      …
</audio>
```

Атрибут **preload** дозволяє вказати параметри того, як браузер попередньо завантажує аудіо, іншими словами, яку частину файлу він завантажує під час <audio> ініціалізації елемента та до натискання кнопки відтворення. Preload може приймати 3 різні значення: none – не завантажуйте нічого, поки не натиснуто кнопку відтворення; metadata – завантажити аудіо метадані – зазвичай це найкращий варіант, оскільки він дозволяє отримати доступ і відобразити таку інформацію, як тривалість аудіофайлу, і дозволяє браузеру визначити, який аудіофайл йому слід використовувати; auto – завантажте весь аудіофайл якомога швидше. Як правило, це не дуже хороший варіант, якщо ви не можете гарантувати, що ваші користувачі матимуть швидке підключення до мережі.

Стилізація <audio> за допомогою CSS

Елемент <audio> не має власного внутрішнього візуального виведення, якщо вказано атрибут controls, у цьому випадку відображаються елементи керування браузера за замовчуванням.

Елементи керування за замовчуванням мають display значення inline і часто доцільно встановити значення для block покращення контролю над розміщенням і макетом.

Можна стилізувати елементи керування за замовчуванням за допомогою властивостей, які впливають на блок як єдине ціле. Наприклад, можна надати йому border, border-radius, padding, margin тощо. Однак не можна стилізувати окремі компоненти всередині аудіопрогравача (наприклад, змінити розмір кнопки або значки, змінити шрифт тощо).

Хід роботи

1. Запишіть власне аудіо (більшість телефонів сьогодні дозволяють легко записувати аудіо), та перенесіть їх до свого комп'ютера. Можливо, вам доведеться виконати деяке перетворення, щоб отримати MP3 та Ogg формати аудіо файлів, але існує достатньо програм, які дозволять вам зробити це без зайвих проблем, такі як Audacity онлайн перетворювачі.

Рекомендується задиктувати та записати текстову інформацію, яку Ви наводите у третій секції на сторінці згідно вайфрейму з попередніх робіт (з метою підвищення доступності Вашого сайту до всіх людей, у тому числі для людей з вадами зору).

2. Збережіть аудіофайл в новому каталозі (наприклад, тека «media») у загальному каталозі Вашого сайту на вашому комп'ютері.

3. Додайте аудіо до Вашого сайту за допомогою <audio> елементу. Надайте декілька форматів цього запису, щоб різні браузери знайшли аудіоформат, який вони найкраще підтримують, і завантажили його. Вони повинні містити відповідно різні type атрибути.

4. Змусьте браузер відображати елементи керування за замовчуванням.

5. Надайте певних особливостей відтворенню аудіофайлу на сторінці згідно з варіантом (табл. 9.2). *(Важливо! Політика певних браузерів, зокрема Google Chrome, забороняє автоматичне відтворення аудіо, тому рекомендовано перевіряти звук в інших браузерах)*

Контрольні запитання

1. Що таке елемент audio? Синтаксис audio елемента.
2. Які основні складники для додавання аудіо на сайт?
3. Як зациклити аудіо (тобто створити постійне повторення аудіо)?
4. Які формати аудіо ви знаєте?
5. Який атрибут надає контролерів для відтворення аудіо?

Таблиця 9. 2. – Варіанти завдання

Номер варіанту	Завдання	Номер варіанту	Завдання
1	Аудіофайл має почати відтворюватися автоматично	2	Аудіофайл має почати відтворюватися автоматично та починатися без звуку
3	Аудіофайл після досягнення кінця має повертатися на початок і відтворюватиметься знову.	4	Аудіофайл має почати відтворюватися автоматично
5	Аудіофайл має почати відтворюватися автоматично та починатися без звуку	6	Аудіофайл після досягнення кінця має повертатись на початок і відтворюватиметься знову.

**Варіант на завдання визначається згідно зі списком групи, де 1, 7, 13, 19, 25 людина за списком – виконує 1 варіант; 2, 8, 14, 20, 26 людина за списком – виконує 2 варіант; 3, 9, 15, 21, 27 людина за списком – виконує 3 варіант; 4, 10, 16, 22, 28 людина за списком – виконує 4 варіант; 5, 11, 17, 23, 29 людина за списком – виконує 5 варіант; 6, 12, 18, 24, 30 – виконує 6 варіант.*

КП 10. ПІДГОТОВКА ТА ДОДАВАННЯ ВІДЕОКОНТЕНТУ НА ВЕБСТОРІНКУ

Мета роботи – ознайомитися з основними властивостями відеоконтенту, та навчитися вставляти відеовміст у вебсторінку.

Основні теоретичні відомості

Відеоконтент – це будь-який формат контенту, який містить відео.

Поширені форми відеоконтенту включають відеоблоги, анімовані GIF-файли, живі відео, відгуки клієнтів, записані презентації та вебінари.

Відеофайл зазвичай складається з контейнера, що містить візуальні дані (відео без аудіо) у форматі кодування відео (наприклад, VP9) разом з аудіоданими у форматі кодування аудіо (наприклад, Opus). Контейнер також може містити інформацію про синхронізацію, субтитри та метадані, такі як заголовок. Стандартизований (або в деяких випадках фактично стандартний) тип відеофайлу, як-от .webm, – це профіль, визначений обмеженням формату контейнера та дозволених форматів стиснення відео та аудіо.

Програма (або обладнання), яка може декодувати стиснене відео або аудіо, називається кодеком; відтворення або кодування відеофайлу іноді вимагатиме від користувача встановлення бібліотеки кодеків, яка відповідає типу кодування відео та аудіо, що використовується у файлі.

Формат відеофайлу – це тип формату файлу для зберігання цифрових відеоданих у комп'ютерній системі. Відео майже завжди зберігається за допомогою стиснення з втратами для зменшення розміру файлу.

Найпопулярніші відеоформати:

MPEG-4, також відомий як MP4, є найпоширенішим типом файлів для відео. Формат MP4 є стандартним форматом для вебвідео, оскільки відео MP4 мають високу якість і відносно невеликий розмір файлу. MP4 є не лише стандартом для Інтернету, але цей формат відео також використовується для телебачення. Якщо ви додаєте відео на свій вебсайт або завантажуєте відео на YouTube, використання відеоформату MP4 є надійним вибором.

Єдиним реальним недоліком формату MP4 є той факт, що процеси кодування та декодування вимагають багато ресурсів. Хоча фактичні розміри файлів невеликі, стиснення відео під час його збереження є вимогливим до вашого ПК. Коли люди переглядають ці відео, їх потрібно розпакувати в режимі реального часу для досягнення максимальної якості.

Формат файлу **MOV** розроблено Apple для підтримки програвача Quicktime і використовується в основному для редагування відео. Люди

зазвичай не публікують відео MOV безпосередньо в Інтернеті або надсилають їх електронною поштою через великий розмір файлу. Однак такий великий розмір файлу означає, що файли MOV зазвичай пропонують вищу якість, ніж MP4 та інші типи відео. Люди використовують файли MOV на етапі редагування, тому що ви хочете редагувати найвищу якість версії свого відео. Однак після завершення редагування вам доведеться стиснути його та відформатувати для Інтернету, що може зайняти час і місце в пам'яті.

Файли **AVI** є унікальними, оскільки вони пропонують дуже високу якість звуку, яка є функцією, яку ви не маєте в деяких інших типах відео. Оскільки файли AVI пропонують чудову якість відео та аудіо, розміри файлів зазвичай набагато більші.

Коли ви використовуєте файли AVI без втрат, які не були стиснуті, – вам не потрібно декодувати ці файли, щоб переглянути відео, але кожна хвилина результатів відео займає гігабайти місця.

Файли AVI можна використовувати як для YouTube, так і для телевізійного виробництва, але вони не підходять для використання в Інтернеті через великий розмір файлу.

WMV або **Windows Media Video –** це формат відео, розроблений корпорацією Майкрософт для використання в операційних системах Windows. Оскільки WMV створено для Windows, пристрої Apple і Linux зазвичай не пропонують стандартну підтримку відтворення WMV. Однак файли WMV пропонують чудову якість із невеликими розмірами файлів, що робить їх популярним вибором для використання в Інтернеті в деяких випадках. З огляду на це, сумісність із пристроями Apple і Linux викликає занепокоєння.

AVCHD – це формат, спочатку розроблений Sony і Panasonic для відеокамер. Цей формат файлу призначений для запису високоякісної версії відео відразу після його запису, таким чином у вас є відео високої роздільної здатності, яке ви можете використовувати, коли будете готові редагувати відео.

Формат **WebM** був розроблений Google і випущений у 2019 році, коли популярність HTML5 зростала. Цей відеоформат розроблено спеціально для

Інтернету, але найбільшою проблемою, з якою він стикається, є відсутність підтримки. Відео WebM мають надзвичайно малий розмір файлу, не надто жертвуючи якістю. На жаль, Internet Explorer і Safari не пропонують підтримку відео WebM, якщо ви не використовуєте додаткові плагіни.

Був час, коли **Flash-відео** (**FLV**) було найпоширенішим типом відео, але це вже не так, оскільки Flash Player було припинено наприкінці 2020 року. Flash Player більше не входить до складу популярних вебпереглядачів, і кожен, хто намагається відтворити FLV-відео в Інтернеті замість відео побачить повідомлення про помилку. Ви все ще можете використовувати сторонній відеопрогравач для відкриття FLV-відео на телефоні чи комп'ютері, але ви не можете використовувати їх для Інтернету.

Визначаючи різні формати файлів, ми рекомендуємо визначати тип MIME для кожного файлу, щоб дозволити браузеру локалізувати підтримуваний файл.

Елемент **<video>** використовується для відтворення відео в вебконтексті. Їх можна використовувати непомітно як місце призначення для складніших медіафайлів або з видимими елементами керування для керованого користувачем відтворення відеофайлів.

Елемент HTML5 <video> використовується для відтворення відео на вебсторінці. Тут визначається <video> елемент із кількома джерелами – так здійснюється, оскільки не всі браузери підтримують однакові відеоформати.

Щоб забезпечити прийнятне покриття, необхідно вказати принаймні два різні формати. Включення джерел WebM і MP4 має бути достатнім для відтворення відео на більшості платформ і браузерів сьогодні.:

```
<video controls>
    <source src=«rabbit320.mp4» type=«video/mp4»/>
    <source src=«rabbit320.webm» type=«video/webm»/>
     <p> Your browser does not support HTML video, but you can still <a href=«videofile.mp3»>download the music</a>. </p>
</video>
```

Елемент `<source>` дозволяє вказати альтернативні відеофайли, які браузер може вибрати. Браузер використовуватиме перший розпізнаний формат. `<source>` елемент, який приймає атрибути `src` та `type:`

- src містить шлях до відеофайлу, який потрібно завантажити (відносний або абсолютний);
- type використовується для інформування браузера про тип файлу (MIME-тип).

Атрибут `controls` додає елементи керування відео, як-от відтворення, пауза та гучність. Якщо не вказувати цей атрибут, елементи керування не відображатимуться – і натомість необхідно створювати власні елементи керування та запрограмовувати їхню функціональність за допомогою Media API. Однак це може бути хорошим підходом, оскільки елементи керування за замовчуванням виглядають по-різному в різних браузерах. Таким чином, створення власних елементів керування забезпечує узгоджений вигляд елементів керування в усіх браузерах.

Текст між тегами `<video>` і `</video>` відображатиметься лише в браузерах, які не підтримують цей <video> елемент. Отже, ідеальне місце для створення резервного варіанту або повідомлення про несумісність – це перед закривальним </video> тегом.

Ви можете керувати розміром відео за допомогою атрибутів **width і height** або за допомогою CSS. В обох випадках відео зберігає рідне співвідношення ширини та висоти – відоме як співвідношення сторін.

Якщо співвідношення сторін не підтримується встановленими розмірами, відео буде збільшуватися, щоб заповнювати простір по горизонталі, а незаповненому простору за замовчуванням надаватиметься суцільний фоновий колір.

```
<video controls width=«400» height=«400»>
      …
</video>
```

Якщо вказати атрибут **autoplay** відео почне відтворюватися якнайшвидше й без будь-якої взаємодії з користувачем – коротко кажучи, відео відтворюватиметься автоматично.

```
<video controls autoplay>
      …
</video>
```

Це значення часто ігнорується на мобільних платформах, і його використання не рекомендується, якщо це не є дійсно необхідним. Автоматичне відтворення відео (і відео) зазвичай дуже дратує. Крім того, вебпереглядач мають політики, які повністю блокують автоматичне відтворення в багатьох ситуаціях.

Якщо додати атрибут **muted** після `autoplay`, то відеофайл почне відтворюватися автоматично, але без звуку:

```
<video autoplay muted>
      …
</video>
```

Атрибут **loop** гарантує, що після досягнення кінця відеокліпу відеокліп повертатиметься на початок і відтворюватиметься знову.

```
<video autoplay loop>
      …
</video>
```

Атрибут **preload** дозволяє вказати параметри того, як браузер попередньо завантажує відео, іншими словами, яку частину файлу він завантажує під час <video> ініціалізації елемента та до натискання кнопки

відтворення. Preload може приймати 3 різні значення: none – не завантажуйте нічого, поки не натиснуто кнопку відтворення; metadata – завантажити відео метадані, зазвичай це найкращий варіант, оскільки він дозволяє отримати доступ і відобразити таку інформацію, як тривалість відеофайлу, і дозволяє браузеру визначити, який відеофайл йому слід використовувати; auto – завантажте весь відеофайл якомога швидше. Як правило, це не дуже хороший варіант, якщо ви не можете гарантувати, що ваші користувачі матимуть швидке підключення до мережі.

Атрибут **`poster`** – URL-адреса зображення, яке відображатиметься перед відтворенням відео. Він призначений для використання як заставка або рекламного екрана.

```
<video controls poster=«poster.png»>
    …
</video>
```

Альтернативні методи додавання відео на сайт

Існує чимало **OVP** (**online video providers** – онлайн-постачальників відео), таких як YouTube, Dailymotion і Vimeo, а також онлайн-постачальників аудіо, таких як Soundcloud. Такі компанії пропонують зручний і простий спосіб розміщення та споживання відео, що потребує величезного споживання пропускної здатності. OVP навіть зазвичай пропонують готовий код для вбудовування відео/аудіо на ваші вебсторінки (рис. 10.1).

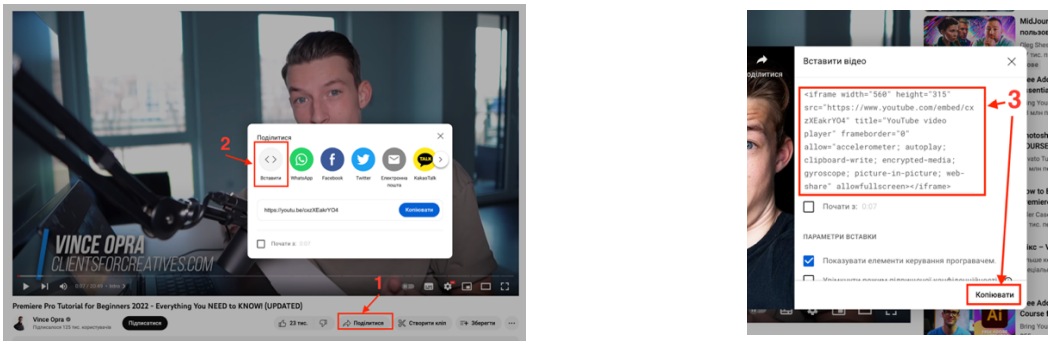


Рис. 10.1. Вбудовування відео із YouTube

Хід роботи

1. Запишіть власне відео та перенесіть його до свого комп'ютера, або скористайтесь готовими рішення (наприклад, завантажте із мережі). Можливо, вам доведеться виконати деяке перетворення, щоб отримати MP4 і Webm формати відео файлів. Для цих цілей скористайтеся або спеціальними програмами редагування відео (такими, як Apple iMovie, Adobe Premiere Pro, Final Cut Pro, Filmora), або онлайн-конвертерами (такими як convertio.co/, video.online-convert, cloudconvert.
2. Збережіть відеофайл в новому каталозі (наприклад, теці «video») у загальному каталозі Вашого сайту на вашому комп'ютері.
3. Додайте відео до Вашого сайту за допомогою <video> елементу. Надайте декілька форматів цього запису, щоб різні браузери знайшли відеоформат, який вони найкраще підтримують, і завантажили його. Вони повинні містити відповідно різні type атрибути. Задайте браузеру відображати елементи керування за замовчуванням.
4. Створіть постер до відео та додайте його. Задайте браузеру відображати елементи керування за замовчуванням.
5. Впровадьте те саме відео через онлайн-постачальник відео, такий як YouTube (у випадку впровадження власного відео – його необхідно попередньо завантажити на YouTube, можна скористатися інструкцією).
6. Сформулювати висновки щодо переваг та недоліків різних методів вбудовування відео на вебсторінку.

Контрольні запитання

1. Що таке елемент video? Синтаксис video елемента.
2. Які основні складники для додавання відео на сайт?
3. Як зациклити відео (тобто створити постійне повторення відеокліпу)?
4. Які формати відео ви знаєте?
5. Який атрибут надає контролерів для відтворення відео?

КП 11. ДОДАВАННЯ АДАПТИВНОСТІ ДО ВЕБСТОРІНКИ

Мета роботи – ознайомитися з основними властивостями адаптивного макета, та навчитися розробляти вебсторінки адаптивними.

Основні теоретичні відомості

Адаптивний вебдизайн – це динамічний макет, який використовує медіазапити для калібрування формату сайту відповідно до типу пристрою. Адаптивний дизайн зазвичай будується на гнучкій сітці з єдиним макетом, який інтуїтивно адаптується до інтерфейсу. Точки адаптивності (CSS breakepoints) мають вирішальне значення для адаптивного дизайну для масштабування елементів дизайну.

Основні переваги додавання адаптивності на сторінку:

- підхід є економніший за коштами та часом, оскільки дизайнери зосереджені на створенні єдиного макета;
- адаптивний макет легше підтримувати та оновлювати, оскільки він використовує одну URL-адресу для кожної сторінки;
- адаптивний макет SEO дружній;
- адаптивний макет забезпечує узгоджений інтерфейс користувача на всіх пристроях;
- оскільки цей підхід більш популярний, багато готових шаблонів доступні з розширеною функціональністю на популярних CMS, таких як WordPress або Joomla.

Недоліки додавання адаптивності на сторінку:

- процес вимагає від розробників додаткового кодування, щоб забезпечити оптимізацію на всіх пристроях.
- адаптивні сайти можуть мати довший час завантаження, оскільки деякі важчі елементи дизайну можуть завантажуватися швидше на настільних комп’ютерах, ніж на мобільних.

- щоб підтримувати візуальну ієрархію елементів дизайну зі змінними розмірами екрана, потрібно багато тестувати.

- підхід до дизайну дає менше контролю над макетом дизайну на екранах різних розмірів.

Адаптивність вебсайту закладається на перших етапах розробки. Для цього використовують різні типи гнучкого дизайну, наприклад:

- гумовий макет – спосіб, що базується на використанні гнучких сіток, які адаптують сайт до різних розмірів і орієнтації екрана;

- бибудовування блоків – спосіб, при якому блоки сайту переносять вниз сторінки при звуженні екрана;

- медіазапити CSS – спосіб, при якому відбувається перемикання стилів залежно від пристрою, яким користується відвідувач сайту. Макет підлаштовується під ширину, висоту, роздільну здатність і орієнтацію екрана;

- адаптивні зображення – спосіб, при якому налаштування медіафайлів таким чином, щоб відбувалась адаптація сайту під різну роздільну здатність екрана та його розміри;

- визначення пристроїв і їх можливостей – спосіб, при якому сервер визначає пристрій і браузер користувача, щоб підлаштувати сайт відповідно до їх налаштувань та обрати коректні розміри макетів для адаптивного дизайну.

- підлаштування до різних способів введення спосіб, при якому користувачі взаємодіють з сайтом різними способами: миша, дотик, клавіатура, голос. Адаптивна сторінка може підлаштовуватися під спосіб користування сайтом.

Перевірити адаптивність сайту можна, просто відкривши його на різних пристроях. Якщо у вас немає 5-10 різних смартфонів і планшетів для тестування, є онлайн-сервіси, які імітують різні розміри екранів, щоб зрозуміти, як на них виглядатиме макет адаптивного сайту. Наприклад: responsinator.com, responsivedesignchecker.com, responsivetesttool.com

Створення адаптивного макета

Перш за все, при створенні адаптивного вебсайту, необхідно додати такий `<meta>` елемент до всіх своїх вебсторінок:

```
<meta name=«viewport» content=«width=device-width, initial-scale=1.0»>
```

Мета-тег **`viewport`** повідомляє браузеру про те, як саме обробляти розміри сторінки, і змінювати її масштаб. Цей тег необхідно додати в розділ HEAD. Може мати наступні атрибути, указані через кому (,):

- `width` – ширина області перегляду. Значенням атрибута є ціле незначне число від 200 до 10000 пікселів або константа device-width, яка задає ширину сторінки відповідно до розміру екрана. Якщо значення не задано, за замовчуванням встановлюється – у мобільному Safari = 980px, Opera = 850px, Android WebKit = 800px, IE = 974px.

Для сайтів з адаптивним дизайном рекомендується використовувати: ***width=device-width****.*

- **`height`** – висота області перегляду. Значенням атрибута є ціле незначне число від 233 до 10000 пікселів або константа device-height, яка задає висоту сторінки відповідно до розміру екрана. Якщо вказано атрибут width, показувати атрибут height не обов'язково.

- **`initial-scale`** – початковий масштаб сторінки. Значенням атрибута є число від 0,1 до 1,0. Значення 1.0 визначає масштаб 1:1, тобто «не масштабувати». У деяких операційних системах (iOS, Windows Phone і т.д.) ширина сторінки, при повороті, залишається незмінною. Замість перерозподілу контенту здійснюється його масштабування. Тому *рекомендується використовувати: initial-scale=1.0.*

- **`user-scalable`** – доступність масштабування сторінки користувачем. Значення атрибута є логічним «yes» (1) – можна масштабувати або «no» (0) – не можна масштабувати. Рекомендується використовувати

значення «yes», а оскільки його встановлено за замовчуванням, то атрибут user-scalable можна і не вказувати.

- **minimum-scale** – мінімальний масштаб області перегляду. Значенням атрибута є число від 0,1 до 1,0. У мобільному браузері Safari за умовчанням 0.25. Значення 1.0 визначає масштаб 1:1, тобто «не масштабувати».

- **maximum-scale** – максимальний масштаб області перегляду. Значенням атрибута є число від 0,1 до 1,0. У мобільному браузері Safari за умовчанням 1.6. Значення 1.0 визначає масштаб 1:1, тобто «не масштабувати».

Гумовий макет часто базується на застосуванні респонсивних зображень та елементах, розміри яких задані у відносних одиницях.

Відносні одиниці вимірювання

Ем (em) – одиниця вимірювання, що дозволяє задавати розміри у співвідношенні до розмірів шрифту. 1 ем відповідає поточному розміру шрифту. У випадку застосування одиниці вимірювання у властивості font-size, використовується наслідуваний розмір шрифту. Таким чином, font-size: 0.5em установить розмір шрифту вдвічі менший за наслідуваний.

Рем (rem) – одиниця вимірювання, аналогічна до em, але в якості вихідної величини використовує розмір шрифту не поточного, а корінного елемента – тобто <html>. Не рекомендується застосовувати для задання розмірів шрифту на самому корінному елементі, оскільки в такому випадку в якості вихідного значення буде використаний розмір шрифту браузера за замовчуванням, який може бути різним для різних браузерів.

Відсотки (%) – дозволяє задавати розміри в частках від іншої величини. Яка саме вихідна величина використовується – залежить від властивості. Наприклад, у випадку властивості font-size наслідується розмір шрифту. Це означає, що для властивості font-size 100% = 1em. Для описання розмірів в якості вихідної величини можуть використовуватися ширина сторінки або певного елемента. Але зверніть увагу, що не всі властивості, що мають значення з одиницями вимірювання, дозволяють задавати значення у відсотках.

ch (від англ. «character» – «літера», «символ») – одиниця вимірювання, що в якості вихідної величини використовує ширину або висоту цифри 0, залежно від напрямку написання тексту. Так, для традиційних для нас горизонтальних мов написання текстів використовується ширина, натомість для деяких східних мов буде використана висота. У випадках, коли розміри цифри неможливо чи непрактично рахувати, використовуються значення 0.5em і 1em для ширини та висоти відповідно.

ex – одиниця вимірювання, що в якості вихідної величини використовує висоту латинської літери x. В більшості випадків, відповідає висоті рядкових літер шрифту.

vw (від англ. «viewport width» – «ширина вікна перегляду») – одиниця вимірювання, що дозволяє задавати розміри відносно ширини вікна браузера користувача. 1vw це 1% від ширини вікна. Якщо ширина вікна 1200 пікселів, то 1vw буде дорівнювати 12 пікселям, 10vw – 120 пікселям, а 100vw – повній ширині у 1200 пікселів.

vh (від англ. «viewport height» – «висота вікна перегляду») – одиниця вимірювання, аналогічна до vw, але в якості вихідної величини використовується не ширина, а висота вікна браузера.

vmin (від англ. «viewport minimum» – «мінімум вікна перегляду») – одиниця вимірювання, яка в якості вихідної величини використовує менший з вимірів вікна браузера. У випадках, коли ширина більша за висоту, ця одиниця вимірювання буде аналогічною до vh, а в інших випадках – до vw.

vmax (від англ. «viewport maximum» – «максимум вікна перегляду») – аналогічна до vmin, але використовує не менший, а більший з двох вимірів.

Респонсивні зображення – це зображення, які гарно масштабуються, щоб відповідати будь-якому розміру вебпереглядача.

Використання властивості width (ширина)

Якщо для властивості ширини CSS встановлено значення у відсотках, зображення буде масштабуватися під час зміни розміру вікна браузера.

```
<img src=«img_girl.jpg» style=«width:100%;»>
```

Використання властивості max-width

```
max-width: none|length|initial|inherit;
```

Якщо властивість max-width встановлено на 100%, зображення буде зменшуватися, якщо це необхідно, але ніколи не буде збільшуватися до розміру, що перевищує його вихідний розмір.

Особливості властивості max-width:

- властивість max-width визначає максимальну ширину елемента;
- якщо вміст більший за максимальну ширину, він автоматично змінить висоту елемента;
- якщо вміст менший за максимальну ширину, max-width властивість не впливає.

CSS Media Queries (Медіазапити)

Медіазапити відіграють важливу роль в адаптивних вебсторінках.

За допомогою медіазапитів ви можете визначити різні стилі для різних розмірів браузера.

Правило `@media`, введене в CSS2, дозволило визначити різні правила стилю для різних типів медіа. Наприклад, можна було мати один набір правил стилю для екранів комп'ютерів, один для принтерів, один для кишенькових пристроїв, один для пристроїв телевізійного типу тощо.

Медіазапити в CSS3 розширили ідею типів медіа CSS2: замість того, щоб шукати тип пристрою, вони дивляться на можливості пристрою.

Медіазапити можна використовувати для перевірки багатьох речей, наприклад:

- ширина та висота вікна перегляду;
- ширина і висота пристрою;

- орієнтація (планшет/телефон в альбомному чи портретному режимі);
- роздільна здатність пристрою.

Використання медіазапитів є популярною технікою для посилання на індивідуальної таблиці стилів на настільні комп'ютери, ноутбуки, планшети та мобільні телефони (наприклад, телефони iPhone й Android).

Синтаксис медіазапиту

Медіазапит складається з медіатипу та може містити один або кілька виразів, які перетворюються на true або false.

```
@media not|only mediatype and (expressions) {
    CSS-Code;
}
```

Результат запиту є істинним, якщо вказаний тип носія відповідає типу пристрою, на якому відображається документ, і всі вирази в медіазапиті є істинними. Якщо медіазапит має значення true, застосовуються відповідні таблиці стилів або правила стилю відповідно до звичайних каскадних правил.

Значення ключових слів: not, only й and :

- `not`: ключове слово not інвертує значення всього медіазапиту.
- `only`: ключове слово only не дозволяє старішим браузерам, які не підтримують медіазапити з медіафункціями, застосовувати вказані стилі. Це не впливає на сучасні браузери.
- `and:` Ключове слово and поєднує медіафункцію з типом медіа або іншими медіафункціями.

Можна мати різні таблиці стилів для різних носіїв:

```
<link rel=«stylesheet» media=«mediatype and|not|only(expressions)» href=«print.css»>
```

Типи носіїв CSS3

Значення	Опис
all	Використовується для усіх типів медіаносіїв
print	Використовується для принтерів
screen	Використовується для екранів комп'ютерів, планшетів, смартфонів тощо.
speech	Використовується для «читалок» (зчитувачів екрана, які «читають» зі сторінки вголос)

Один зі способів використання медіазапитів – мати альтернативний розділ CSS прямо всередині таблиці стилів.

Для адаптивного дизайну `min-width` та `max-width` є найбільш часто використовуваними медіафункціями. Вони дозволяють створювати стилі на основі ширини вікна перегляду.

Медіазапити `min width to max width` – можна використовувати значення для встановлення мінімальної та максимальної ширини.(max-width: ..) and (min-width: ..). Наприклад, якщо ширина браузера становить від 600 до 900 пікселів, змініть вигляд елемента :

```
@media screen and (max-width: 900px) and (min-width: 600px)
{
  div.example { ... }
}
```

Ви також можете використовувати висоту (`height, min-height,` and `max-height), aspect-ratio, resolution, and orientation` як медіазапити.

Медіазапити щодо `orientation: portrait / landscape`

Можна мати набір властивостей CSS, які застосовуватимуться лише тоді, коли вікно браузера ширше за його висоту (так звана «альбомна» орієнтація) та навпаки – коли ширина вікна браузера менша за його висоту («книжкова» орієнтація):

```
@media only screen and (orientation: landscape / portrait)
{
  body { ... }
}
```

Breakpoints (стоп-точки, точки зупину)

Стоп-точка – це точка, що відповідає певній величині (зазвичай ширини), на якій стиль вебсторінки адаптується певним чином, щоб забезпечити найкращу взаємодію з користувачем.

Існує два широких підходи до вибору точок зупину CSS; один базується на пристроях, а інший – на вмісті. Найпоширенішим підходом є групування пристроїв за форм-фактором (наприклад, мобільні пристрої, планшети, ноутбуки тощо) і вибір конкретних точок зупину, які будуть відповідати розмірам екрана якнайбільшої кількості відповідних пристроїв:

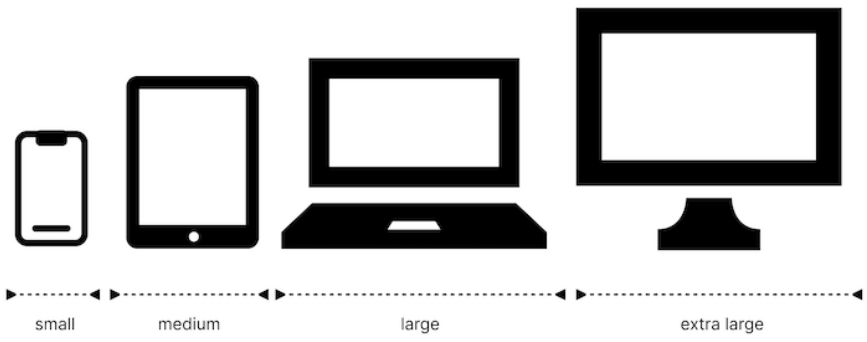


Рис. 11.1. Найтиповіші групи пристроїв, якими користується більшість користувачів

Аргументи, які можна використати, щоб обрати стоп-точки:

- Світова статистика найпоширеніших роздільних здатностей екранів;
- Аналітика даних з вашого сайту;
- Точки зупину, вибрані фреймворками CSS.

Приклади медіазапитів

Розповсюджене використання медіазапитів полягає в створенні гнучкого макета. У цьому прикладі створюється макет, який змінюється між чотирма, двома стовпцями та стовпцями на всю ширину залежно від розмірів екрана:

```
/* Створіть чотири рівних стовпці, які плавають поруч
один біля одного */
.column {
  float: left;
  width: 25%; }
/* На екранах завширшки 992 пікселі або менше
відбувається перехід від чотирьох стовпців до двох
стовпців */
@media screen and (max-width: 992px) {
  .column {
    width: 50%; }
}
/* На екранах шириною 600px або менше стовпці мають
розташовуватися один над одним, а не поруч один біля
одного. */
@media screen and (max-width: 600px) {
  .column {
    width: 100%;}
}
```

У наступному прикладі колір фону змінюється на світло-зелений, якщо область перегляду має ширину 480 пікселів або ширше (якщо область перегляду менше ніж 480 пікселів, колір фону буде рожевим):

```
@media screen and (min-width: 480px) {
     body {background-color: lightgreen;}
}
```

У наступному прикладі показано меню, яке буде виринати ліворуч від сторінки, якщо область перегляду має ширину 480 пікселів або ширше:

```
@media screen and (min-width: 480px) {
     #leftsidebar {width: 200px; float: left;}
     #main {margin-left: 216px;}
}
```

Інше розповсюджене використання медіазапитів – приховування елементів на екранах різних розмірів:

```
/* Якщо розмір екрана складає 600 пікселів в ширину або
менше, сховати елемент */
@media screen and (max-width: 600px) {
  div.example {display: none;}
}
```

Ви також можете використовувати медіазапити для зміни розміру шрифту елемента на екранах різних розмірів:

```
/* Якщо розмір екрана перевищує 600 пікселів, встановіть
розмір шрифту в <div> на 80px */
```

```
@media screen and (min-width: 600px) {
  div.example {font-size: 80px;}
}

/* Якщо розмір екрана складає 600 пікселів в ширину або
менше, встановіть розмір шрифту в <div> на 30px */
@media screen and (max-width: 600px) {
  div.example {font-size: 30px;}
}
```

Різні зображення для різних пристроїв

HTML5 представив елемент <picture>, який дозволяє вам визначити більше одного зображення.

Елемент <picture> працює аналогічно елементам <video> і <audio>. Ви встановлюєте різні джерела, і перше джерело, яке відповідає перевазі, є тим, що використовується:

```
<picture>
    <source srcset=«img_smallflower.jpg»
    media=«(max-width: 400px)»>
    <source                    srcset=«img_flowers.jpg»>
    <img      src=«img_flowers.jpg»      alt=«Flowers»>
</picture>
```

Атрибут `srcset` є обов'язковим і визначає джерело зображення.

Атрибут `media` є необов'язковим і приймає запити мультимедіа (CSS `@media` правила).

Також необхідно визначити елемент `<img>` для браузерів, які не підтримують елемент `<picture>`.

Хід роботи

1. Створіть сторінку та додайте до неї:

   - прямокутник блакитного кольору розміром 150px x 150px ;

   - зображення квітки шириною 100%;

   - текст «Адаптивність сторінки» гарнітурою Poppins, кеглем 18px.

2. Згідно з вашим варіантом (табл. 11.1) внесіть зміни до сторінки при її відображенні на різних пристроях. Адаптивність вибудовується за методикою «mobile first», тобто від найменшого розміру екрана – до більшого.

3. Використовуючи медіазапити, адаптуйте власну розроблену сторінку під 3 розміри – малі, середні, великі. Точки зупину (breakpoints) визначте самостійно, аргументуйте їх вибір. Зверніть увагу, що у цьому випадку адаптація відбуватиметься «від більшого – до меншого».

Таблиця 11. 1. – Варіанти завдання

<table>
<tr><td rowspan="2">Номер варіанту</td><td rowspan="2">Елемент</td><td colspan="4">Зміни елемента на різних екранах</td></tr>
<tr><td>Дуже малі пристрої ($\leq 600\ px$)</td><td>Малі пристрої ($> 600\ px$ та $\leq 768\ px$)</td><td>Середні пристрої ($> 768\ px$ та $\leq 992\ px$)</td><td>Великі пристрої ($> 992\ px$)</td></tr>
<tr><td rowspan="3">1</td><td>прямокутник</td><td>розмір 150px x 150px</td><td>розмір 200px x 200px</td><td>розмір 250px x 250px</td><td>розмір 300px x 300px</td></tr>
<tr><td>зображення</td><td>Зображення квітки</td><td>Зображення моря</td><td>Зображення неба</td><td>Зображення гір</td></tr>
<tr><td>текст</td><td>Кегль 18px</td><td>Кегль 22px</td><td>Кегль 26px</td><td>Кегль 32px</td></tr>
<tr><td rowspan="3">2</td><td>прямокутник</td><td>колір – блакитний</td><td>колір – зелений</td><td>колір – жовтий</td><td>колір – помаранчевий</td></tr>
<tr><td>зображення</td><td>Зображення квітки</td><td>Зображення райдуги</td><td>Зображення метелика</td><td>Зображення єдинорога</td></tr>
<tr><td>текст</td><td colspan="2">Гарнітура Poppins, Кегль 18px</td><td colspan="2">Гарнітура Times, Кегль 32px</td></tr>
<tr><td rowspan="3">3</td><td>прямокутник</td><td>Розмір 150px x 150px, колір – блакитний</td><td colspan="2">розмір 200px x 200px, колір – зелений</td><td>розмір 300px x 300px, колір – помаранчевий</td></tr>
<tr><td>зображення</td><td colspan="2">Зображення квітки, ширина 100%</td><td colspan="2">Зображення квітки, ширина 100%, максимальна ширина (max width) – 600 px</td></tr>
<tr><td>текст</td><td>Кегль 18px, регістр – всі великі літери</td><td colspan="2">Кегль 22px, регістр – нормальний (як у реченнях)</td><td>Кегль 32px, регістр – нормальний</td></tr>
</table>

Закінчення табл. 11.1

Номер варіанту	Елемент	Зміни елемента на різних екранах			
		Дуже малі пристрої ($\leq 600\ px$)	Малі пристрої ($> 600\ px$ та $\leq 768\ px$)	Середні пристрої ($> 768\ px$ та $\leq 992\ px$)	Великі пристрої ($> 992\ px$)
4	прямокутник	розмір 150px x 150px, радіус заокруглення кутів 75 px		розмір 300px x 300px, радіус заокруглення кутів 150 px	
	зображення	Зображення квітки	Зображення дівчини	Зображення будинку	Зображення міста
	текст	Кегль 18px	Кегль 22px	Кегль 26px	Кегль 32px
5	прямокутник	розмір 150px x 150px, чорне обведення розміром 1px		розмір 300px x 300px, чорне обведення розміром 3px	
	зображення	Зображення квітки, ширина 100%	Зображення квітки, ширина 80%	Зображення квітки, ширина 60%	Зображення квітки, ширина 40%
	текст	Кегль 18px, напівжирне накреслення	Кегль 22px, напівжирне накреслення	Кегль 26px, нормальне накреслення	Кегль 32px, нормальне накреслення
6	прямокутник	колір – блакитний	колір – зелений	колір – жовтий	колір – помаранчевий
	зображення	Зображення квітки, ширина 100%, максимальна ширина (max width) – 320 px		Зображення квітки, ширина 100%, максимальна ширина (max width) – 600 px	
	текст	Гарнітура Poppins, Кегль 18px	Гарнітура Arial, Кегль 22px		Гарнітура Times, Кегль 32px

**Варіант на завдання визначається згідно зі списком групи, де 1, 7, 13, 19, 25 людина за списком – виконує 1 варіант; 2, 8, 14, 20, 26 людина за списком – виконує 2 варіант; 3, 9, 15, 21, 27 людина за списком – виконує 3 варіант; 4, 10, 16, 22, 28 людина за списком – виконує 4 варіант; 5, 11, 17, 23, 29 людина за списком – виконує 5 варіант; 6, 12, 18, 24, 30 – виконує 6 варіант.*

4. Сформулювати висновки щодо виконаної роботи.

Контрольні запитання

1. Що таке адаптивний макет?
2. Які основні переваги додавання адаптивності на сторінку?
3. Які основні недоліки додавання адаптивності на сторінку?
4. Синтаксис медіазапиту.

КП 12. СТВОРЕННЯ ІНТЕРАКТИВНОГО МЕНЮ

Мета роботи – навчитися створювати ефективну та зручну для користувача навігацію у вебсайті, використовуючи сучасні вебтехнології.

Основні теоретичні відомості

Навігація є ключовим елементом будь-якого вебсайту. Вона допомагає користувачам переміщатися між сторінками сайту, швидко знаходити необхідну інформацію та робить взаємодію з ресурсом інтуїтивно зрозумілою.

Основні види навігації:

- горизонтальна – тип навігації, при якій меню розташоване вздовж верхньої частини сторінки.
- вертикальна – тип навігації, при якій меню розташоване збоку сторінки.
- багаторівнева – тип навігації, при якій підменю з додатковими категоріями, що з’являються при наведенні або натисканні.
- футерна – тип навігації, при якій дублювання основного меню в нижній частині сторінки.

Верстка меню може виконуватись різними способами залежно від його типу, структури та функціональних вимог. Для сучасних проектів бажано використовувати семантичні теги (<nav>, <ul>, <li>), а також гнучкі технології, такі як Flexbox або CSS Grid. Це забезпечує адаптивність, зручність підтримки коду та кращу сумісність із пристроями.

Способи верстання навігаційного меню:

1. Використання списків (впорядкованого (<ol>) або невпорядкованого (<ul>)). Це семантично правильний і найпоширеніший спосіб.

2. Горизонтальне меню за допомогою Flexbox, який дозволяє зручно розташувати елементи навігації горизонтально або вертикально з можливістю керування їх вирівнюванням.

3. Вертикальне меню, яке зазвичай використовуються для бокової навігації. Їх можна створити за допомогою списків або блоків.

4. Використання CSS Grid, яке забезпечує більше гнучкості у створенні меню з унікальними макетами.

5. Випадаюче меню, що дозволяють створювати багаторівневу навігацію.

Приклад реалізації горизонтальної навігації у вебпроєкті у додатку 2.

Хід роботи

1. Додайте у базову структуру документа з html-розміткою власної сторінки як мінімум чотири пункти меню. Наприклад, «Головна», «Про нас», «Послуги», «Контакти». Для прикладу можна скористатися кодуванням, наведеним у теоретичних відомостях.

2. Додайте випадаюче меню для одного із пунктів, яке міститиме як мінімум три підпункти.

3. Додайте оформлення базової навігації за допомогою стилів CSS.

4. Розробіть виведення випадаючого меню при наведенні. Для цього можна скористатися різними методами позиціювання, трансформування, приховування елементів, і т.п.

5. Використовуючи медіазапити, адаптуйте розроблену навігацію під 3 розміри – малі, середні, великі. Точки зупину (breakpoints) визначте самостійно, аргументуйте їх вибір. Спробуйте реалізувати мобільну навігацію із використанням іконки «бургер-меню», яка відкриватиме повний список пунктів меню при натисканні на мобільних пристроях.

6. Сформулювати висновки щодо виконаної роботи.

Контрольні запитання

1. Що таке навігація на вебсайті, і чому вона важлива?

2. Які основні типи навігації використовуються у вебпродуктах?

3. Як створити горизонтальне меню за допомогою HTML та CSS?

4. Як можна реалізувати випадаюче меню? Поясніть механізм роботи випадаючого меню.

5. Як можна зробити меню адаптивним для мобільних пристроїв?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Двірничук К.В., Вацек Д.О. Веб-програмування та веб-дизайн : навч. посіб. Чернівці : Чернівец. нац. ун-т ім. Ю. Федьковича, 2022. 472 с.

2. Веб-технології та веб-дизайн : навч. посібник / О. Г. Трофименко, О. Б. Козін, О. В. Задерейко, О. Є. Плачінда. Одеса : Фенікс, 2019. 284 с.

3. Баран С. В. Основи web-програмування: Навчальний посібник. Кривий Ріг: Державний університет економіки і технологій, 2023. 316 с.

4. Пасічник О. Г. Основи веб-дизайну: навч. посіб. / О. Г. Пасічник, О. В. Пасічник, І. В. Стеценко. К.: Вид. група BHV, 2009. 336 с.

5. Лобода Ю. Г. Вебтехнології та вебдизайн: навчально-методичний посібник для здобувачів вищої освіти галузі знань 12 «Інформаційні технології» [Електронне видання] / уклад.: Ю. Г. Лобода, А. А. Толокнов ; НУ «ОЮА». Одеса : Фенікс, 2023. 260 с.

6. Web-технології та Web-дизайн: застосування мови HTML для створення електронних ресурсів : навч. посіб. / І. Л. Бородкіна, Г. О. Бородкін. Київ: Видавництво Ліра-К, 2020. 212 с.

7. Бутенко В. М., Павленко Є. П., Головко О. В. Інженерія програмного забезпечення. WEB програмування: Навч. посібник. Харків: УкрДУЗТ, 2019. 127 с.

8. Мережеві електронні видання : довідник / Т. Ю. Киричок, О. І. Лотоцька. Київ : НТУУ «КПІ», Вид-во «Політехніка», 2016. 300 с.

ДОДАТКИ

Додаток 1

```
<!DOCTYPE html>
  <html lang=«en»>
   <head>
     <meta charset=«UTF-8»>
     <meta name=«viewport» content=«width=device-width, initial-
scale=1.0»>
     <title>Document</title>
     <style>
          * {
              box-sizing: border-box;
          }
          body {
                    background-color: #fff788;
                    margin: 0;
                    padding: 0;
               }

               .wrapper {
                    align-items: center;
                    display: flex;
                    justify-content: center;
                    height: 100vh;
               }

               .book {
                    height: auto;
                    width: auto;
               }

               .book.book_path {
                    fill: transparent;
                    stroke-dasharray: 888;
                    stroke-dashoffset: 0;
               }

               .stroke {
                    fill: transparent;
                    stroke-dasharray: 36;
                    stroke-dashoffset: 0;
               }

               /* Hover */
```

```
.book:hover .book_path {
    animation: bookAnim 1.5s ease-in-out forwards;
}

.book:hover .stroke-one,
.book:hover .stroke-two,
.book:hover .stroke-three {
    animation: strokeAnim 2s ease-in-out 200ms forwards;
}

/* Keyframes */

@keyframes bookAnim {

    0% {
        stroke-dashoffset: 0;
    }

    40% {
        stroke-dashoffset: 888;
    }

    60% {
        fill: transparent;
    }

    80% {
        fill: #19fffc;
        stroke-dashoffset: 1776;
    }

    100% {
        fill: #19fffc;
        stroke-dashoffset: 1776;
    }
}

@keyframes strokeAnim {

    0% {
        stroke-dasharray: 5;
        stroke-dashoffset: 5;
    }
```

```
                    50% {
                        /*stroke-dasharray: 10;  */
                       /* stroke-dashoffset: 3;*/
                        fill: transparent;
                    }

                   70% {
                        stroke-dasharray: 36;
                        stroke-dashoffset: 0;
                        fill: white;
                    }

                    100% {
                      fill: white;
                    }
                }
            </style>
</head>
<body>
     <div class=«wrapper»>
     <div class=«container»>
          <div class=«book»>
               <svg width=«215» height=«179» viewBox=«0 0 215 179»
fill=«none» xmlns=«http://www.w3.org/2000/svg»>
                    <path class=«book_path» d=«M106.499 175.42C110.851
158.936 118.5 150 124.5 148.5C129.42 147.27 185.115 146.712 204.998
146.551C208.876 146.52 211.967 143.347 211.926 139.47L210.572
10.8274C210.532 7.0295 207.489 3.95779 203.691 3.91333C186.76 3.71515
143.617 3.32219 136.668 4.36441C136.218 4.43194 135.77 4.5201 135.317
4.57273C122.937 6.01336 113.756 13.7203 109 21.7056V21.7056C107.931
23.5005 104.861 23.5948 103.739 21.8328C97.7981 12.5059 87.4994 4.84905
77 4.5L10.5476 4.04793C6.66311 4.0215 3.5 7.16316 3.5
11.0478V139.907C3.5 143.809 6.68719 146.96 10.5893 146.924C29.5985
146.747 80.7893 146.471 88.5 148.5C95.5107 150.345 103.026 167.943
106.137 175.604V175.604C106.215 175.795 106.5 175.739 106.499
175.533L106.405 25.3477» stroke=«#112F65» stroke-width=«6» stroke-
linejoin=«round»/>
                    <path class=«stroke stroke-one» d=«M141 43H176»
stroke=«#112F65» stroke-width=«6» stroke-linecap=«round» stroke-
linejoin=«round»/>
                    <path class=«stroke stroke-two» d=«M141 75H176»
stroke=«#112F65» stroke-width=«6» stroke-linecap=«round» stroke-
linejoin=«round»/>
```

```
                    <path class=«stroke stroke-three» d=«M141 107H176»
stroke=«#112F65» stroke-width=«6» stroke-linecap=«round» stroke-
linejoin=«round»/>
                    </svg>

          </div>
     </div>
</body>
</html>
```

```
<!DOCTYPE html>
  <html lang=«uk»>
    <head>
      <meta charset=«UTF-8»>
      <meta name=«viewport» content=«width=device-width,
      initial-scale=1.0»>
      <title>Навігація у вебпродукті</title>
      <link rel=«stylesheet» href=«style.css»>
    </head>
    <body>
      <header>
        <nav>
          <ul>
            <li> <a href=«#»>Головна</a> </li>
            <li> <a href=«#»>Про нас</a> </li>
            <li class=«dropdown»>
              <a href=«#»>Послуги</a>
              <ul class=«dropdown-content»>
                <li> <a href=«#»>Веброзробка</a> </li>
                <li> <a href=«#»>Графічний дизайн</a> </li>
                <li> <a href=«#»>SEO-оптимізація</a> <li>
              </ul>
            </li>
            <li> <a href=«#»>Контакти</a> </li>
          </ul>
        </nav>
      </header>
       <main>
        <h1>Ласкаво просимо на наш сайт!</h1>
      </main>
    </body>
  </html>
```

Рис. 2.1. Базова html-структура документа інтерактивного меню

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

nav ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    background-color: #333;
    overflow: hidden;
}

nav ul li {
    float: left;
}

nav ul li a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 20px;
    text-decoration: none;
}

nav ul li a:hover {
    background-color: #111;
}

.dropdown {
    position: relative;
    display: inline-block;
}

.dropdown-content {
    display: none;
    position: absolute;
    background-color: #f9f9f9;
    min-width: 160px;
    box-shadow: 0px 8px 16px rgba(0, 0, 0, 0.2);
    z-index: 1;
}

.dropdown-content a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
    text-align: left;
}

.dropdown-content a:hover {
    background-color: #f1f1f1;
}

.dropdown:hover .dropdown-content {
    display: block;
}

header {
    background-color: #333;
    color: white;
    padding: 10px 0;
    text-align: center;
}

h1 {
    padding: 20px;
    text-align: center;
}
```

Рис. 2.2. Оформлення стилів CSS для базової навігації

```
@media screen and (max-width: 600px) {
    nav ul {
        display: flex;
        flex-direction: column;
    }

    nav ul li {
        width: 100%;
        text-align: center;
    }}
```

Рис. 2.3. Адаптивність меню

```
<div class=«burger-menu»>
    <div class=«burger-icon»>&#9776;</div>
    <ul class=«burger-content»>
        <li><a href=«#»>Головна</a></li>
        <li><a href=«#»>Про нас</a></li>
        <li><a href=«#»>Послуги</a></li>
        <li><a href=«#»>Контакти</a></li>
    </ul>
</div>
```

Рис. 2.4. Базова html-структура документа бургер-меню

```
.burger-icon {
    display: none;
    font-size: 30px;
    cursor: pointer;
}

.burger-content {
    display: none;
}

@media screen and (max-width:
    600px) {
    .burger-icon {
        display: block;
    }

    .burger-content {
        display: block;
        background-color: #333;
        list-style-type: none;
        padding: 0;
    }

    .burger-content li {
        padding: 10px;
        text-align: center;
    }

    .burger-content li a {
        text-decoration: none;
        color: white;
    }

    .burger-content li a:hover {
        background-color: #111;
    }
```

Рис. 2.5. Оформлення стилів CSS для бургер-меню

```
document.querySelector('.burger-
    icon').addEventListener('click', function() {
    var menu = document.querySelector('.burger-content');
    if (menu.style.display === 'block') {
        menu.style.display = 'none';
    } else {
        menu.style.display = 'block';
    }
});
```

Рис. 2.6. JavaScript для відкриття/закриття меню