CREATING WEB-PAGES
Using HTML5 and CSS3

STEP computer ACADEMY

Урок 7

Адаптивний дизайн

Зміст

Матеріали уроку прикріплені до цього PDF-файлу. Щоб отримати доступ до матеріалів, урок необхідно відкрити у програмі Adobe Acrobat Reader.

1. Що таке адаптивний дизайн?

В сучасному світі існує велика кількість пристроїв, за допомогою яких ми виходимо в Інтернет: комп'ютери, планшети, смартфони, телевізори та навіть годинники. І користувачу буде незручно дивитись вебсторінку, якщо вона на смартфоні відкриється в такому ж вигляді, що й на ноутбуці: елементи інтерфейсу будуть замалими або з'явиться горизонтальне прокручування. Користувач, швидше за все, вийде з такого сайту, якщо не застосувати до нього адаптивний дизайн.

Давайте розглянемо приклад як би відображався відомий пошуковик Google, якби він не мав адаптивного дизайну.

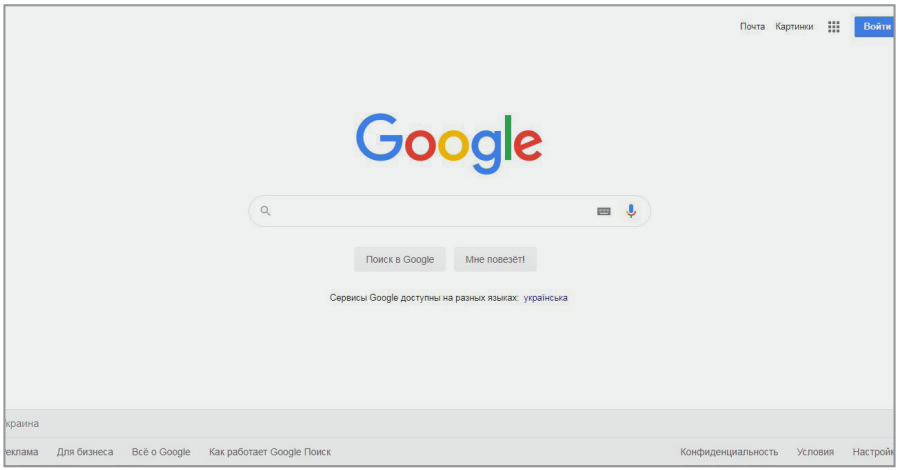

Рисунок 1. Відображення пошуковика Google на ноутбуці

На рисунку 1 всі елементи зручно розташовані для використання та займають всю видиму частину вікна браузера на ноутбуці.

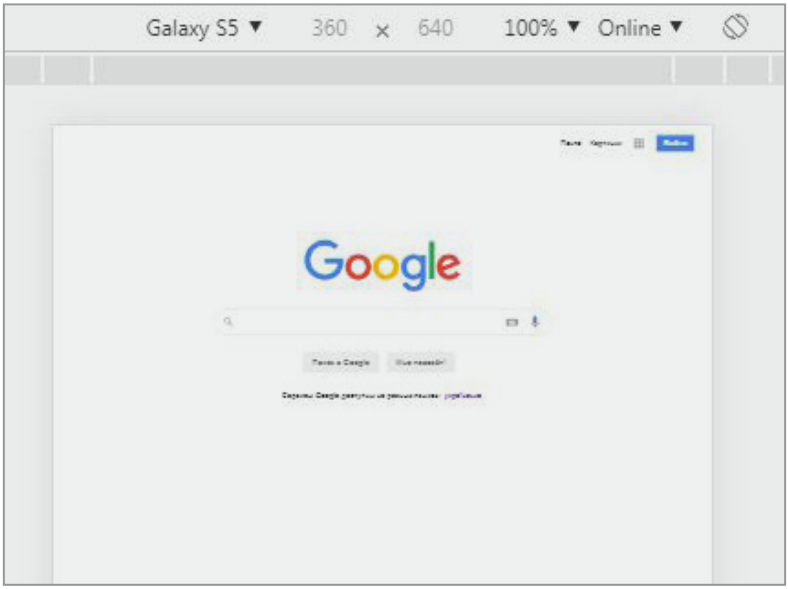

Рисунок 2. Відображення пошуковика Google
на Galaxy S5 без адаптивності

На рисунку 2 видно, що елементи сторінки стали дуже малими і незручними ні для перегляду, ні для використання.

На рисунку 3 видно, як адаптивний дизайн перебудував сторінку пошуковика: всі елементи знову зручно розташовані та їх розміри відповідають пристрою.

Адаптивний дизайн — це дизайн, який дозволяє правильно відображати вебсторінки на різних пристроях та плавно змінює елементи сторінки при зміненні розміру вікна браузера.

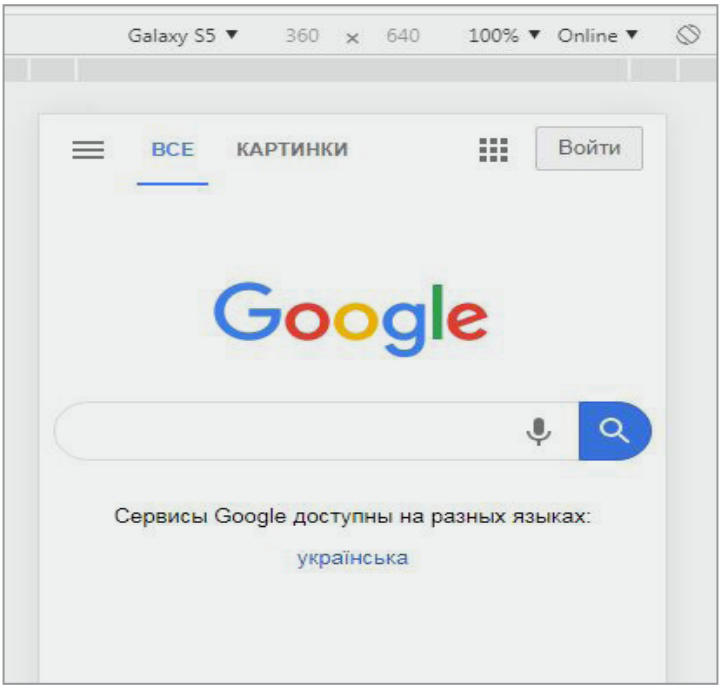

Рисунок 3. Відображення пошуковика Google
на Galaxy S5 з адаптивністю

Не так давно було прийнято створювати окрему «мобільну» версію вебсайту, яка насправді була копією контенту з іншим відображенням елементів, що призводило до проблем в просуванні таких сайтів. З приходом визначення адаптивного дизайну окремі версії сайту для різних пристроїв створювати непотрібно, тому що при одному й тому самому контенті ми отримуємо зручне розташування елементів на мобільних пристроях.

Переваги адаптивного дизайну:

- Відсутня необхідність створювати окрему версію вебсайту для конкретного пристрою.

- Універсальне представлення вебсторінок для різних пристроїв.
- Всі сторінки доступні за однією url-адресою, що позбавляє від проблем у просуванні сайту.
- Зручність використання інтерфейсу вебсторінок незалежно від гаджета.

Недоліки адаптивного дизайну:

- Повільне завантаження сайту через його велику вагу. Незалежно від пристрою завантажується повна версія сайту.
- Складніша верстка вебсторінок, через те що потрібно враховувати тонкощі відображення на всіх видах пристроїв.
- Неможливість «вимкнути» мобільне відображення на пристрої з маленькою роздільною здатністю, необхідно відкривати сторінку на іншому пристрої з більшою роздільною здатністю.
- Складніший та довший процес тестування сайту.

Але, навіть враховуючи недоліки, адаптивний дизайн є необхідним, тому що кількість мобільних користувачів і різноманітність гаджетів зростає щодня.

2. Принципи адаптивного дизайну

1. Використання відносних одиниць виміру. Для того, щоб елементи плавно змінювались в залежності від ширини вікна браузера, необхідно використовувати відносні одиниці виміру для вказування ширини, висоти, внутрішніх і зовнішніх відступів, розмірів шрифту. До відносних одиниць належать: %, em, ex, vh, vw, vmin.
2. Застосування межових значень для ширини контейнера. На маленькому пристрої контент на ширину всього вікна виглядає гарно, а таке ж відображення контенту на широкоформатному моніторі буде викликати дискомфорт під час перегляду. Тому рекомендовано використовувати межові значення для ширини/висоти в абсолютних величинах, а саме в пікселях. Для цього використовуються властивості: min-width/min-height і max-width/max-height.
3. Використання структури у вигляді сітки. У CSS3 існують різні інструменти, що дозволяють будувати гнучку структуру для розташування елементів. Такими інструментами є модуль Flexbox і сітка Grid Layout.
4. Використання медіа-запитів для перебудовування відображення елементів сторінки. Для того щоб сторінки не просто плавно стискали контент в залежності від ширини браузера, а перебудовували зміст під

гаджет для зручного перегляду, необхідно прописувати контрольні точки. Контрольні точки — це фізичний параметр пристрою, за яким визначається поточне відображення. Встановлюються контрольні точки за допомогою медіа-запитів.

5. Починати верстку з мобільного відображення та поступово просуватись до широкоформатного або навпаки. Прийнято починати верстку адаптивної вебсторінки, починаючи з маленьких пристроїв, через те що вони містять менше елементів і загалом просте й лаконічне відображення контенту, поступово переходячи до більших розмірів екрана. Та навпаки, можна рухатись від широкоформатного пристрою до мобільного представлення.
6. Використання системних шрифтів. Звичайно, ви будете використовувати в дизайні різні шрифти, які представлені в макеті. Також варто пам'ятати, що шрифт, завантажений з ресурсу в інтернеті уповільнює його завантаження. Системні шрифти завантажуються миттєво, чим істотно прискорюють завантаження вебсторінки.
7. Приховання або заміна елементів на різних пристроях. На мобільних пристроях часто приховують елементи, що не несуть інформативності (наприклад, банер з рекламою стороннього ресурсу), або приховують їх за межі екрана, щоб можна було розгорнути їх в будь-який момент (наприклад, фільтри товару).
8. Адаптація графічного та відео контенту. Для того, щоб сторінки відвантажувались швидше, необхідно використовувати зображення, адаптивні під поточне відображення. Наприклад, фонові зображення мен-

шого розміру для мобільних пристроїв. В тих випадках, коли це можливо, використовувати векторний формат зображень замість растрового.

Приклади адаптивного дизайну можна подивитись на ресурсі Media Queries (mediaqueri.es), де представлена галерея вебсайтів з адаптивним дизайном.

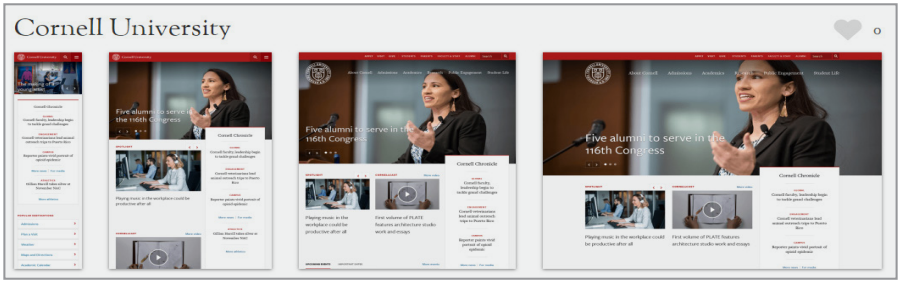

Рисунок 4

На рисунку видно, що елементи сайту перебудовуються в залежності від пристрою та його розміру.

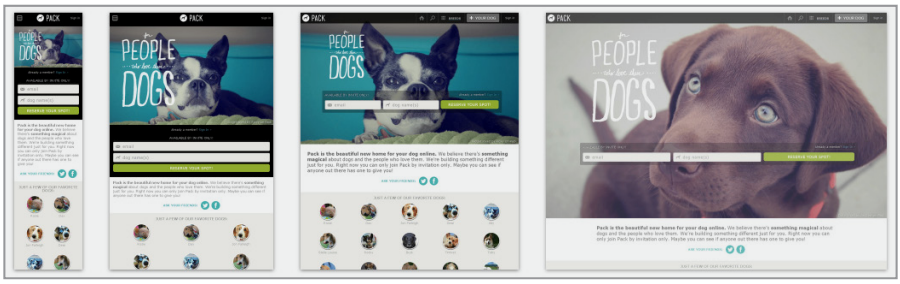

Рисунок 5

Елементи інтерфейсу перебудовуються під пристрій для комфортнішого використання, також відвантажуються різні зображення для швидшого завантаження сайту.

3. Метатег viewport

Viewport — це видима область вікна браузера, в яку останній вписує вебсторінку. Кожен браузер визначає ширину viewport по своєму. Наприклад, у Safari — 980 пікселів, у IE — 1024 пікселя. В середньому, ширина десь біля 1000 пікселів, тому що вважається, що вебсторінки призначені для десктопних моніторів.

Процес відображення сторінки складається з наступних етапів: браузер отримує сторінку з серверу, задає їй розміри своєї ширини viewport, а потім пропорційно стискає сторінку до розмірів відображеного пристрою.

Для того, щоб браузер не масштабував сторінку, прийнявши її ширину за ширину viewport встановлену за замовчуванням в його налаштуваннях, необхідно використовувати метатег viewport.

Метатеги призначені для вказування інформації для браузерів і пошукових систем. Метатег viewport вказує браузеру в якому масштабі необхідно відображати видиму частину сторінки на різних пристроях.

В таблиці 1 представлено перелік параметрів і їх значень, які може приймати метатег viewport.

Щоб браузер розумів, що сторінка є адаптованою під різні пристрої, необхідно прописати метатег viewport з наступними значеннями:

```
<meta name="viewport"
      content="width=device-width,
      initial-scale=1">
```

Таблиця 1

Параметр	Значення	Опис
width	Ціле число в пікселях або значення device-width, яке дорівнює ширині екрану в пікселях CSS при масштабі 100%.	Задає ширину області viewport. Ширина в пікселях CSS — це не фізична роздільна здатність екрану, а величина що регламентує розмір пікселя. Необхідна для того, щоб при більшій щільності пікселів, елементи виглядали однаково.
height	Ціле число в пікселях або значення device-height, яке дорівнює висоті екрану в пікселях CSS при масштабі 100%.	Задає висоту області viewport.
initial-scale	Дійсне число від 0.1 та вище	Задає коефіцієнт масштабування початкового розміру viewport (1.0 — відсутність масштабування).
user-scalable	no/yes	Забороняє/дозволяє користувачу масштабувати сторінку.
minimum-scale	Дійсне число від 0.1 і вище	Задає мінімальний масштаб розміру viewport. (1.0 — відсутність масштабування).
maximum-scale	Дійсне число від 0.1 і вище	Задає максимальний масштаб розміру viewport. (1.0 — відсутність масштабування).

Значення, вказані в атрибуті content, є значеннями за замовчуванням. Їх потрібно змінювати, якщо ваш сайт

повинен мати, наприклад, мінімальну ширину не менше за 450 пікселів. Тоді значення метатега

```
<meta name="viewport"
      content="width=450, initial-scale=1">
```

буде фактично задавати цю мінімальну ширину для viewport. Якщо екран пристрою буде шириною більше ніж 450 пікселів, то браузер буде розширювати область перегляду, а не зменшувати її.

Другий приклад використовують тоді, коли налаштування масштабу повинні залишатись незмінними при виключені орієнтації:

```
<meta name="viewport" content="initial-scale=1,
      maximum-scale=1">
```

В специфікації CSS3 визначення тегу <meta name=»viewport»> є ненормованим, тобто на цей момент для цього тегу немає встановленого стандарту.

4. Медіа-запити

Медіа-запит по суті є умовною конструкцією, яка запитує у пристрою, що відображає вебсторінку, його характеристики і виконує набір стильових правил, якщо отримані характеристики пристрою відповідають заданим в умові поточного медіа-запиту.

Медіа-запити використовуються в адаптивному дизайні коли необхідно застосувати різні CSS-стилі для відображення на різних пристроях з урахуванням їх характеристик. Наприклад, на рисунку 6 видно, як перебудовується сторінка від пристрою до пристрою. Зверніть увагу на елемент, що виконує пошук по сайту.

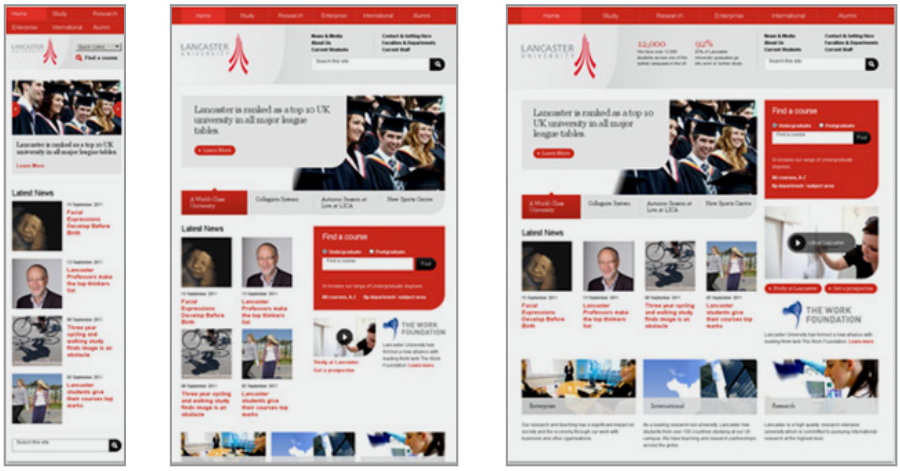

Рисунок 6

На десктопному відображенні він знаходиться в верхньому правому кутку сторінки, залишаючись на

тому ж місці при виведенні результатів пошуку. В мобільній версії цей елемент зміщується вниз сторінки, тому що це найзручніше місце для швидкого вводу шуканої інформації.

Медіа-запит складається з типу пристрою (необов'язковий параметр) і технічних характеристик даного пристрою. Його можна застосувати наступними способами:

За допомогою тегу link:

```
<link rel="stylesheet" media="screen and
      (min-width:900px)" href="width_900.css">
```

1. За допомогою правила @import:

```
@import url(width_900.css) screen and (min-width:900px);
```

2. За допомогою правила @media, вказаного всередині тегу style або в стильовому файлі:

```
@media screen and (min-width:900px){
    /*стилі для вказаного типу пристрою і його
      характеристик*/
}
```

Типи пристроїв:

- all — всі пристрої (використовується за замовчуванням);
- print — режим попереднього перегляду перед друком;
- screen — екрани моніторів;
- speech — синтезатори мови.

В CSS2 перелік типів пристроїв був більшим. Туди входили окремо проектори, телевізори, планшети та

інші пристрої, які в сучасному стандарті прийняті як застарілі.

Нижче наведена таблиця з характеристиками пристроїв, які є обов'язковою частиною медіа-запиту.

Таблиця 2

Найменування	Опис
aspect-ratio	Відношення ширини до висоти. Наприклад: (aspect-ratio: 12/5).
color	Кількість біт на компонент кольору.
color-gamut	Перевіряє кольорову гаму, що підтримується пристроєм: rgb (color-gamut:srgb), p3 (color-gamut:p3), BT.2020 (color-gamut:rec2020).
color-index	Перевіряє чи використовує пристрій таблицю відповідності кольорів. Значення: ціле число.
grid	Перевіряє чи пристрій виводу є сітковим чи растровим. Якщо пристрій виводу представляє сітку (наприклад, термінал або дисплей телефону з одним фіксованим шрифтом), то значення буде дорівнювати 1. В іншому випадку буде 0.
height	Висота області видимості. Задається в абсолютних і відносних одиницях виміру.
monochrome	Кількість біт на піксель монохромного пристрою. Задається цілим числом.
orientation	Орієнтація пристрою: портретна (orientation:portrait) або альбомна (orientation: landscape).
overflow-block	Описує поведінку пристрою, коли йде переповнення контентом по вертикальній осі в режимі горизонтального запису і по горизонтальній осі при вертикальному записі. Значення: немає (overflow-block:none), скролл (overflow-block:scroll), із завантаженням (overflow-block: optional-paged), посторінково (overflow-block: paged).

Таблиця 2 (продовження)

Найменування	Опис
overflow-inline	Описує поведінку пристрою, коли йде переповнення контентом по горизонтальній осі в режимі горизонтального запису і по вертикальній осі при вертикальному записі. Значення: немає (overflow-inline:none), скрол (overflow-block:scroll).
resolution	Роздільна здатність екрана — кількість пікселів на дюйм (dpi) або сантиметр (dpcm).
scan	Перевіряє процес рендерінгу пристроїв: черезрядковий (scan:interlace) і прогресуючий (scan:progressive).
update	Перевіряє можливість оновлювати зміст після його візуалізації (наприклад, анімацію CSS): ні (update: none), повільно (update:slow) і швидко (update:fast).
width	Ширина області видимості. Задається в абсолютних і відносних одиницях виміру.

Медіа-запит може мати в складі комплексну перевірку характеристик пристрою. Комбінований медіа-запит створюється за допомогою логічних операторів:

1. and — об'єднує декілька медіа-функцій в один медіа-запит. Запит виконується тільки в тому разі, якщо всі медіа-функції відповідають характеристикам пристрою. Наприклад:

```
@media (min-width:320px) and (max-width:480px){}
```

   Цей запит виконається лише тоді, якщо ширина екрана пристрою знаходиться в діапазоні від 320 до 480 пікселів.

2. Кома — поєднує декілька медіа-запитів в одне правило. Кожен запит оброблюється окремо від інших, таким

чином стилі застосовуються, якщо хоча б один запит відповідає характеристикам пристрою. Наприклад:

```
@media screen and (aspect-ratio: 16/9),
       screen and (aspect-ratio: 16/10){}
```

Цей запит виконається для моніторів, у яких пропорційне відношення ширини до висоти дорівнюватиме 16/9 або 16/10.

3. not — використовується для інвертування медіа-запиту. Наприклад:

```
@media not (color){}
```

Цей запит виконається для пристроїв, які не підтримують кольоровість.

4. only — використовується для застосування стилю лише в тому разі якщо відповідає всьому запитові. Наприклад, приховує стилі для старих браузерів.

Приклади формування медіа-запитів з урахуванням типу та характеристики пристрою:

- **Ширина/висота** — це найпопулярніші варіанти медіа-запитів, які зчитують ширину або висоту пристрою:

Таблиця 3

Пристрій та параметри	Запит
Смартфони з шириною екрана в діапазоні від 320 до 480 пікселів	@media only screen and (min-width:320px) and (max-width:480px) { /* стилі */ }
Смартфони з максимальною висотою екрана, що дорівнює 600 пікселів	@media only screen and (max-height:600px) {/*стилі*/ }

- **Орієнтація** (альбомна та портретна) — відбувається перевірка орієнтації пристрою, якщо вона існує та увімкнена:

Таблиця 4

Пристрій та параметри	Запит
Смартфони з альбомною орієнтацією	@media screen and (orientation: landscape) { /*стилі*/ }
Попередній перегляд на принтері з портретною орієнтацією	@media print and (orientation: portrait) {/*стилі*/ }

- **Колір** — відбувається перевірка пристрою на можливість відображати різні кольори:

Таблиця 5

Пристрій та параметри	Запит
Пристрої, що не підтримують різні кольори, наприклад, чорно-білий принтер	@media not (color) { /* стилі*/ }
Пристрої, що підтримують 4-бітний колір або менше	@media (max-color: 4) {/* стилі */}

- **Пропорції** — перевіряє пропорції екрана монітора, а саме відношення ширини пристрою до його висоти:

Таблиця 6

Пристрій та параметри	Запит
Широкоформатні монітори з пропорційним відношенням ширини до висоти дорівнюючим 16/9	@media screen and (aspect-ratio: 16/9){ /* стилі*/ }
Квадратні монітори з пропорційним відношенням ширини до висоти дорівнюючим 1/1	@media screen and (aspect-ratio: 1/1){ /* стилі*/ }

- **Розширення** — перевіряє розширення пристрою — кількість пікселів на дюйм або сантиметр:

Таблиця 7

Пристрій та параметри	Запит
Пристрої з розширенням 150dpi (кількість пікселей на дюйм)	@media (max-resolution: 150dpi){ /* стилі*/ }

- **Комплексні медіа-запити** використовуються в тих випадках, коли потрібно перевірити декілька параметрів або застосувати однакові стилі до різних пристроїв і їх технічним характеристикам:

Таблиця 8

Пристрій та параметри	Запит
Пристрій має мінімальну висоту 680 пікселів або є екранним пристроєм в портретному режимі	@media (min-height: 680px), screen and (orientation: portrait){ /* стилі*/ }

Давайте розглянемо на прикладі невеликої сторінки як реалізуються медіа-запити на практиці.

Створимо сторінку, в якій будуть відображатись основні елементи: header, main на три колонки, footer. В якості складу пропишемо назви тегів і їх розташування на десктопному відображенні сторінки.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width,
          initial-scale=1.0">
    <title>Media</title>
```

```
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <div class="container">
        <header>header top</header>
        <main>
            <aside>aside left</aside>
            <section>section center</section>
            <aside>aside right</aside>
        </main>
        <footer>footer bottom</footer>
    </div>
</body>
</html>
```

Ось що вийшло:

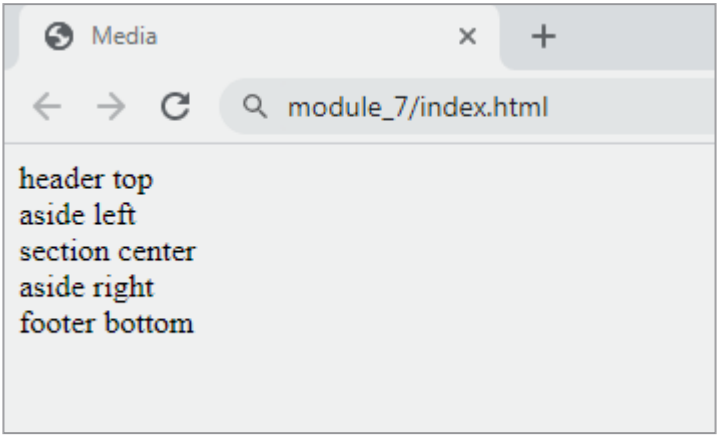

Рисунок 7

На рисунку вище видно, що всі елементи розташовуються один під одним, тому що вони блочні теги. Займемося стильовим оформленням.

```
/* звернемося до всіх елементів, щоб прибрати
   відступи, які назначить браузер за замовчуванням */
*{
    padding: 0;
    margin: 0;
}
/* для основного контейнера будемо використовувати
   фіксовано-резинову верстку */
.container{
    /* встановлюємо основну ширину, яка дорівнює
       ширині видимої частини вікна браузера */
    width: 100vw;
    /* встановлюємо мінімальну ширину, до якої можна
       стискати контейнер, якщо ширина пристрою буде
       менше — з'явиться горизонтальний скрол*/
    min-width: 320px;
    /* встановлюємо максимальну ширину, до якої можна
       розширювати контейнер, якщо ширина пристрою
       буде більша — з боків з'являться відступи */
    max-width: 1200px;
    /* для того, щоб контейнер знаходився завжди по
       центру по горизонталі, вказуємо значення для
       зовнішніх відступів */
    margin: auto;
    /* у випадку, якщо контенту буде мало, щоб
       сторінка повністю зайняла всю висоту вікна
       браузера, встановлюємо мінімальну висоту */
    min-height: 100vh;
    /* у випадку, якщо контенту буде багато, щоб
       з'явився вертикальний скрол, встановлюємо
       значення висоти з урахуванням вмісту*/
    height: auto;
    /* для простішого розташування елементів на
       сторінці, скористаймося технологією flexBox */
    display: flex;
    /* напрямок основної осі змінюємо на вертикальний */
    flex-direction: column;
}
```

```
/* кожному елементу контейнера будемо встановлювати
   фоновий колір, для наочності */
header{
    background-color: pink;
    /* задамо розмір header по основній осі */
    flex-basis: 10vh;
}
/* main — основний елемент сторінки є одночасно і
   flex-батьком і flex-контейнером */
main{
    /* щоб заповнювати весь простір по основній осі,
       основним елементом прописуємо властивість flex */
    flex: 1 1 auto;
    display: flex;
}
main>aside{
    background-color: cyan;
    /* задаємо розмір лівої частини по основній осі
       у відсотках */
    flex-basis: 20%;
}
main>section{
    background-color: coral;
    /* щоб заповнювати весь простір по основній осі,
       основним елементом прописуємо властивість flex */
    flex: 1 1 auto;
}
main>section+aside{
    background-color: lime;
    /* задаємо розмір лівої частини по основній осі
       у відсотках*/
    flex-basis: 20%;
}
footer{
    background-color: gray;
    /* задаємо розмір footer по основній осі */
    flex-basis: 5vh;
}
```

Наразі ми отримали наступне відображення для широкоформатного монітора:

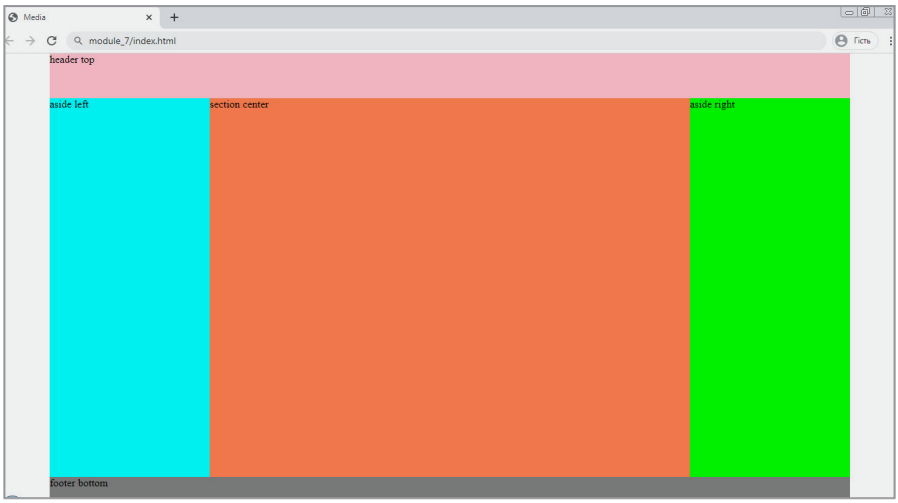

Рисунок 8

Переходимо до створення медіа-запитів. Через те, що спочатку ми будували сторінку для десктопного відображення, то тепер підемо за спадом поступово наближуючись до мобільного пристрою. Для цього створимо два медіа-запита для відображення на «квадратних» моніторах і на смартфонах.

```
/* запит для екранів з максимальною шириною 960px і
   менше*/
@media screen and (max-width:960px){
   main>section+aside{
       /* заховаємо правий елемент на всіх пристроях,
          з шириною 960px і менше*/
       display: none;
   }
```

```
   main>aside{
        /* задаємо розмір лівої частини по основній
           осі у відсотках */
        flex-basis: 30%;
   }
   main>section{
        /* задаємо розмір центральної частини по
           основній осі у відсотках */
        flex-basis: 70%;
   }
}
/* запит для екранів з максимальною шириною 570px
   і менше */
@media screen and (max-width:570px){
    main{
        /* змінюємо напрям основної осі елементу,
           що відображає основний контент*/
        flex-direction: column;
    }
    main>aside{
        /* задаємо розмір видимому елементу aside
           по основній осі*/
        flex-basis: 20vh;
    }
    main>section{
        /* задаємо розмір основному контенту по
           основній осі*/
        flex-basis: auto;
        /* змінимо порядок відображення елементів
           так, щоб контент знаходився вище*/
        order:-1;
    }
}
```

Отже, ми отримали ще два види відображення. Для квадратних моніторів:

Рисунок 9

А також для смартфонів:

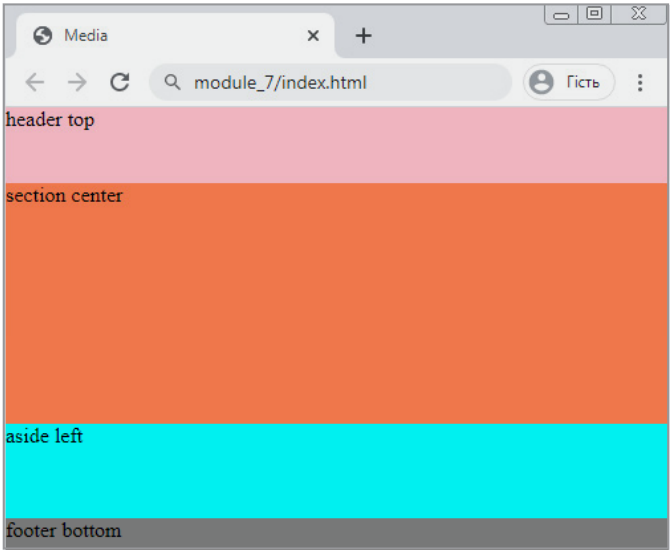

Рисунок 10

Приклад готового коду можна подивитись у файлі *example_module_7.zip* прикріпленого до цього PDF.

Розглянемо приклад створення невеликої сторінки сайту новин. На широкоформатних моніторах контент відображатиметься в трьох колонках, на квадратних — в двох і на мобільних — в одній колонці.

На рисунку 11 представлено відображення на широкоформатному моніторі.

Рисунок 11

Створимо сторінку *index.html* з наступним змістом:

5. Шапка сайту містить назву сайту і головне меню;
6. Основна частина сайту складається з:
   - колонки останніх новин, що знаходиться зліва,
   - основного контенту — статті з заголовком, зображенням і іконками соціальних мереж, який знаходиться посередині,
   - колонки цитат, що розташована справа.

7. Підвал сайту містить знак копірайту та поточний рік.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport"
         content="width=device-width,
                  initial-scale=1.0">
   <title>Media</title>
   <!-- підключаємо шрифт Bitter з ресурсу fonts.
        googleapis.com -->
   <link href="https://fonts.googleapis.com/
              css2?family=Bitter&display=swap"
         rel="stylesheet">
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <div class="container">
      <header>
          <h1>Media</h1>
          <nav>
             <a href="">Home</a>
             <a class="active" href="">News</a>
             <a href="">Gallery</a>
             <a href="">Contact</a>
          </nav>
      </header>
      <main>
         <aside>
            <div>
               <h3>Last News</h3>
               <article class="news">
                  <strong>business</strong>
                  <h2><a href="">Conseptur eos ipsam
                      possimus</a>
                  </h2>
```

```
            <em>by Modi Cumque</em>
            <p>Mollitia modi cumque architecto
            commodi illum, conseq sed nihil
            error aliquam accusamus.</p>
            <strong>2 hours ago</strong>
         </article>
         <article class="news">
            <strong>art</strong>
            <h2>
               <a href="">Lorem architecto
               commodi illum</a>
            </h2>
            <em>by Timus Onihil</em>
            <p>Dolore ipsam possimus mollitia
            modi cumque architecto commodi
            illum, consequuntur!</p>
            <strong>4 hours ago</strong>
         </article>
      </div>
   </aside>
   <section>
      <article>
         <h2>
            Conseptur eos ipsam possimus
         </h2>
         <em>by Modi Cumque</em>
         <img src="images/urbanization.jpg"
              alt="">
         <p>Facilis minus alias consequun-
         tur esse autem ducimus quaerat,
         delectus obcaecati quidem voluptatibus
         corrupti ex, aliquam, error quisquam
         dicta ut. Eum quasi quidem fugit
         corporis labore velit voluptatibus
         consectetur, laudantium cupiditate?</p>
         <p>Tempora earum iure corporis iste
         ea voluptates impedit doloribus minima,
         error nisi. Eum cum sunt inventore
```

```
            nesciunt quod dicta? Nostrum consectetur
            odit sit nulla ad odio debitis rem
            sapiente animi impedit aut tenetur
            autem vitae inventore numquam dolore
            iste quos, architecto sed? </p>
            <p>Repellat quisquam numquam a
            inventore dolores, modi quod autem
            voluptatibus! Lorem ipsum dolor
            sit amet consectetur adipisicing elit.
            Architecto, et. Tempora, doloremque
            inventore? Minima fugiat quis quas
            ipsam voluptates adipisci?</p>
            <div class="social">
               <a class="fb" href=""></a>
               <a class="tw" href=""></a>
               <a class="inst" href=""></a>
            </div>
      </article>
</section>
<aside>
  <div>
      <h3>Osed accusamus</h3>
      <figure>
         <img src="images/unnamed.png" alt="">
         <figcaption>
            <p>Consectetur eos sunt, dolore
            ipsam possimus mollitia modi cumque
            architecto commodi illum!</p>
            <em>by Modi Cumque</em>
         </figcaption>
      </figure>
      <figure>
         <img src="images/unnamed.png" alt="">
         <figcaption>
            <p>Odio in, delectus commodi sapiente
            nostrum ratione porro laborum
               facere quasi cumque!</p>
               <em>by Timus Onihil</em>
```

```
                    </figcaption>
                </figure>
            </div>
        </aside>
    </main>
    <footer>
        <p>&copy; 2020</p>
    </footer>
  </div>
</body>
</html>
```

Зараз всі елементи розташовані на сторінці послідовно, в порядку їх опису, як показано на рисунку 12:

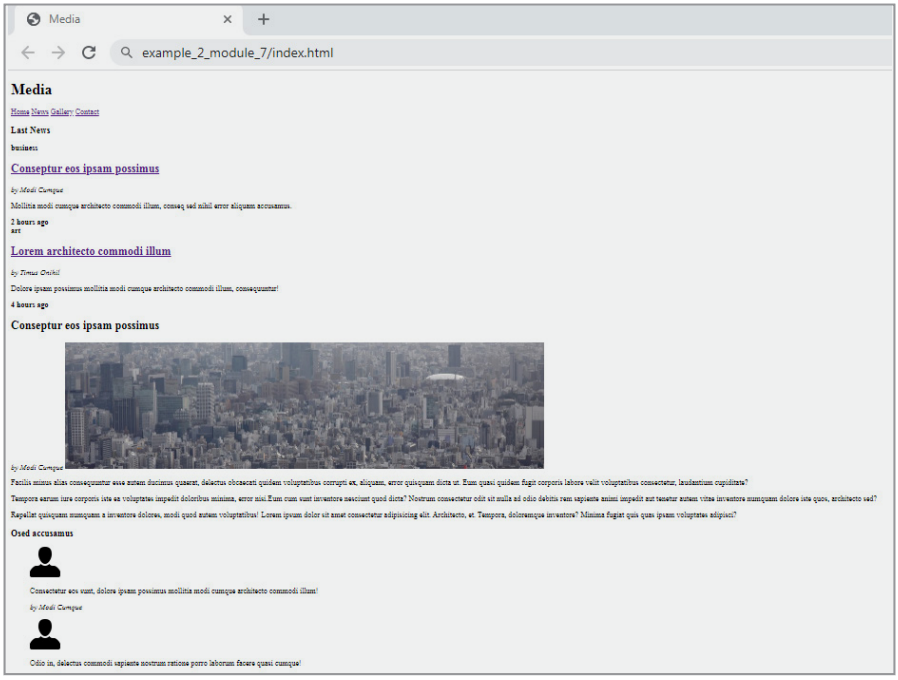

Рисунок 12

Приступимо до створення стильового файлу. Спочатку виконуємо стильове оформлення для основного відображення, в нашому випадку це широкоформатні монітори.

```
/* звернемося до всіх елементів, щоб прибрати
   відступи, які назначає браузер за замовчуванням */
*{
    padding: 0;
    margin: 0;
}
html {
    /* задаємо основний розмір шрифту, для того, щоб
       користуватись відносною одиницею rem*/
    font-size: 16px;
}
body{
    /* заховаємо контент, який перевищує ширину вікна
       браузера*/
    overflow-x: hidden;
    /* встановлюємо основний шрифт контенту — системний —
        це дозволить швидше завантажувати сторінку */
    font-family: Helvetica, Tahoma, Verdana, sans-serif;
    /* Одиниця rem задає розмір шрифту елементу <html> */
    font-size: 1rem;
    color: rgb(75, 75, 75);
}
h1,h2,h3,a{
    /* шрифт для заголовків і посилань буде 'Bitter',
       який ми підключили за допомогою тегу <link> */
    font-family: 'Bitter', serif;
    font-weight: 400;
    color: rgb(105, 105, 105);
}
/* для основного контейнеру будемо використовувати
   фіксовано-резинову верстку */
.container{
    /* встановлюємо основну ширину, що дорівнює
       видимій частині вікна браузера */
```

```
    width: 100vw;
    /* встановлюємо мінімальну ширину, до якої можна
       стискати контейнер, якщо ширина пристрою буде
       менше — з'явиться горизонтальний скрол */
    min-width: 320px;
    /* встановлюємо максимальну ширину, до якої можна
       розширювати контейнер, якщо ширина пристрою
       буде більша — з боків з'являться відступи */
    max-width: 1200px;
    /* для того, щоб контейнер знаходився завжди
       по центру, по горизонталі вказуємо значення
       для зовнішніх відступів */
    margin: auto;
    /* у випадку якщо контенту буде замало, щоб
       сторінка повністю зайняла всю висоту вікна
       браузера, встановлюємо мінімальну висоту */
    min-height: 100vh;
    /* у випадку, якщо контенту буде багато, щоб
       з'явився вертикальний скрол, встановлюємо
       значення висоти, яка дорівнює вмісту */
    height: auto;
    /* для простішого розташування елементів на
       сторінці, скористаємося технологією flexBox */
    display: flex;
    /* напрямок основної осі змінюємо на вертикальний */
    flex-direction: column;
}
/* кожному елементу контейнера встановлюватимемо
   фоновий колір, для наочності */
header{
    /* задамо розмір header по основній осі */
    flex-basis: 10vh;
    /* елементи шапки сайту розташовуються зліва на
       право*/
    display: flex;
    /* між елементами задаємо відступ */
    justify-content: space-between;
```

```
    /* відступ справа та зліва залежить від ширини
       вікна браузера */
    padding: 0 3vw;
}
header h1{
    /* вирівнюємо зміст дочірнього елемента за
       вертикаллю*/
    align-self: center;
}
header nav{
    /* вирівнюємо зміст дочірнього елемента за
       вертикаллю*/
    align-self: center;
    /* для того, щоб легше розташувати посилання,
       скористаємося технологією flexBox */
    display: flex;
}
/* стильове оформлення посилань основного меню */
header nav a{
    display: block;
    margin-left: 2vw;
    padding: 0.5vw 1vw;
    /**/
    font-size: 1.1rem;
    text-decoration: none;
    border: 2px solid transparent;
}
header nav a:hover,
header nav a.active{
    border-bottom: 2px solid rgba(105, 105, 105, 0.5);
}
/* main – основний елемент сторінки є одночасно і
   flex-батьком, і flex-контейнером */
main{
    /* щоб заповнювати весь простір по основній осі,
       основним елементом прописуємо властивість flex */
    flex: 1 1 auto;
    display: flex;
```

```
    /* дочірні елементи розташовуються зліва на право */
    justify-content: space-between;
    padding-top: 2vh;
}
main>aside{
    /* задаємо розмір лівої частини на основній осі
       у відсотках*/
    flex-basis: 20%;
}
main>aside>div{
    /* задаємо відступи у відсотках від ширини
       батьківського елементу*/
    padding: 3% 10%;
}
main>aside>div>*{
    border-bottom: 2px solid rgba(105, 105, 105, 0.3);
    margin-bottom: 2vh;
}
main>aside>div>h3{
    text-transform: capitalize;
}
.news{
    font-size: 0.9rem;
    padding-bottom: 5px;
}
.news>*{
    display: block;
    padding-bottom: 7px;
}
.news>h2>a{
    font-size: 1.1rem;
    text-decoration: none;
}
.news>strong, .news>em{
    color: rgb(155, 155, 155);
    font-size: 0.7rem;
}
```

```
.news>strong{
    text-transform: uppercase;
}
main>section{
    /* щоб заповнити весь простір по основній осі,
       основним елементом прописуємо властивість flex */
    flex: 1 1 55%;
}
main>section>article{
    padding: 1% 5%;
}
main>section>article>*{
    display: block;
    padding-bottom: 1.5vh;
}
main>section>article>h2{
    text-transform: uppercase;
}
main>section>article>em{
    color: rgb(155, 155, 155);
    font-size: 0.9rem;
}
main>section>article>img{
    /* задаємо зображенню ширину 100% від ширини
       батьківського елементу */
    width: 100%;
}
.social{
    text-align: right;
}
/* соціальні мережі підключаємо використовуючи
   зображення, що складається з трьох іконок
   соціальних мереж. В такому разі завантажується
   одне зображення замість трьох і, як наслідок,
   швидкість завантаження сайту вище.*/
.social>a{
    /* змінюємо відображення посилання з рядкового на
       рядково-блочний */
```

```
    display: inline-block;
    /* розмір посилання дорівнює розміру однієї іконки */
    width:32px;
    height: 32px;
    /* у випадку, якщо шлях до зображення не містить
       пробілів або спеціальних символів, можна
       прописувати його без лапок */
    background-image: url(images/social_icons.png);
    background-repeat: no-repeat;
    /* встановлюємо розмір так, щоб відображалася
       лише одна іконка*/
    background-size: 300%;
    opacity: 0.7;
}
/* змінюємо позиціонування фонового зображення у
   відповідності з класом для соцмережі */
.social>a.fb{
    background-position: 0 0;
}
.social>a.tw{
    background-position: 50% 0;
}
.social>a.inst{
    background-position: 100% 0;
}
main>section+aside{
    /* задаємо розмір лівої частини по основній осі
       у відсотках*/
    flex-basis: 20%;
}
main>section+aside figure *{
    display: block;
    padding-bottom: 7px;
}
main>section+aside figure img{
    opacity: 0.5;
    width: 32px;
```

```
    height:32px;
}
main>section+aside figure figcaption{
    font-size: 0.9rem;
}
main>section+aside figure figcaption em{
    color: rgb(155, 155, 155);
    font-size: 0.7rem;
}
footer{
    /* задаємо розмір footer по основній осі */
    flex-basis: 5vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
```

Зараз сторінка виглядає наступним чином:

Рисунок 13

Візьмемось за створення медіа-запитів:

```
/* запит для екранів з максимальною шириною 960px і менше */
@media screen and (max-width:960px){
    main>section+aside{
        /* заховаємо правий елемент на всіх пристроях
           з шириною 960px і менше*/
        display: none;
    }
    main>aside{
        /* задаємо розмір лівої частини по основній
           осі у відсотках */
        flex-basis: 30%;
    }
    main>section{
        /* задаємо розмір центральної частини по
           основній осі у відсотках*/
        flex-basis: 70%;
    }
}
/* запит для екранів з максимальною шириною 570px
   і менше */
@media screen and (max-width:570px){
    header{
        flex-basis: auto;
         /* змінюємо напрямок основної осі сайту */
        flex-direction: column;
    }
    main{
        /* змінюємо напрямок основної осі елементу,
           який відображає основний контент */
        flex-direction: column;
    }
    main>aside{
        /* задаємо розмір видимому елементу aside по
           основній осі*/
        flex-basis: 20vh;
    }
```

```
    main>aside>div {
        padding: 3% 5%;
    }
    main>section{
        /* задаємо розмір основному контенту по
           основній осі */
        flex-basis: auto;
        /* змінимо порядок відображення елементів
           так, щоб контент знаходився вище */
        order:-1;
    }
}
```

Тепер для квадратних моніторів сторінка має наступний вигляд:

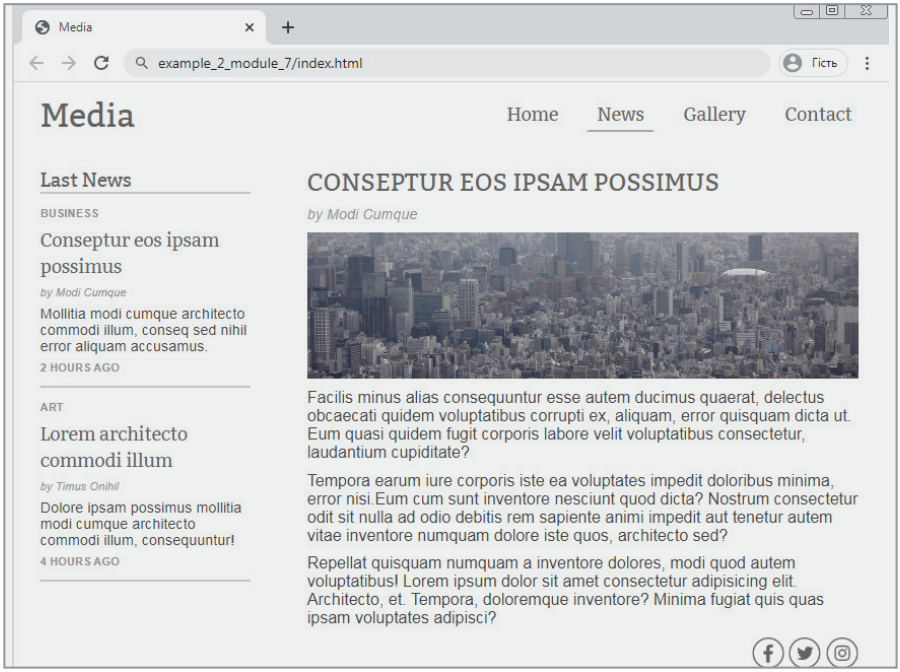

Рисунок 14

Для мобільних пристроїв сторінка виглядає так (рис. 15):

Рисунок 15

Приклад готового коду можна подивитись в файле *example_module_7.zip* який прикріплен до цього PDF.

5. Домашнє завдання

Виконати верстку сторінки за макетом, представленим на рисунку 16.

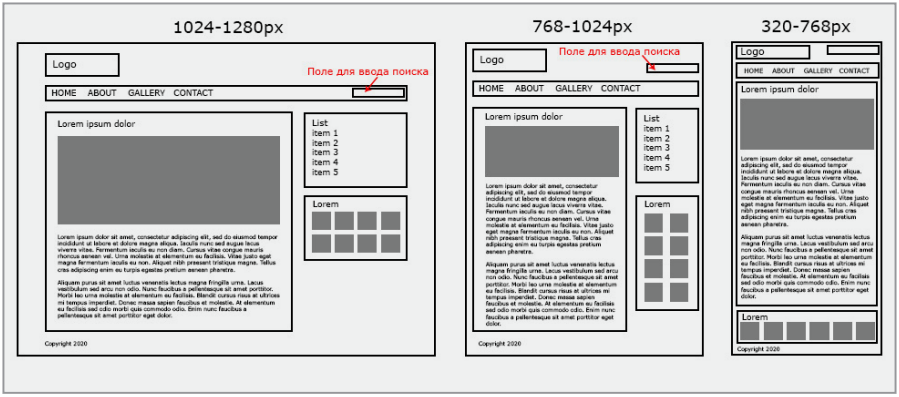

Рисунок 16

Урок 7
Адаптивний дизайн