**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Запорізький національний технічний університет**

# МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисципліни

# "Web-технології та Web-дизайн"

для студентів
напряму підготовки 6.050101
"Комп'ютерні науки"
(всіх форм навчання)

2016

Методичні вказівки до виконання лабораторних робіт з дисципліни "Web-технології та Web-дизайн" для студентів напряму підготовки 6.050101 "Комп'ютерні науки" (всіх форм навчання) / Т.О. Колпакова. – Запоріжжя: ЗНТУ, 2016. – 80 с.

Автори: Т. О. Колпакова, ст. викладач

Рецензент: А.О. Олійник, канд. техн. наук, доцент

Відповідальний за випуск: В.І. Дубровін, канд. техн. наук, професор

Затверджено
на засіданні кафедри
програмних засобів

Протокол №5
від "29" січня 2016 р.

# ЗМІСТ

**Вступ ..... 6**

**1 Лабораторна робота № 1 Загальна структура HTML-документа ..... 7**
1.1 Мета роботи ..... 7
1.2 Короткі теоретичні відомості ..... 7
1.2.1 Перед початком роботи. Середовище розробки. ..... 7
1.2.2 Елементи HTML. Теги ..... 9
1.2.3 Базова структура документа ..... 9
1.2.4 Meta-теги ..... 10
1.2.5 Основні теги для оформлення тексту ..... 11
1.2.5.1 Заголовки тексту ..... 11
1.2.5.2 Абзаци (параграфи) ..... 11
1.2.5.3 Елементи фізичного форматування ..... 12
1.2.5.4 Елементи логічного форматування ..... 12
1.2.5.5 Елементи списків ..... 13
1.2.5.6 Базові атрибути елементів оформлення тексту ..... 14
1.3 Завдання на лабораторну роботу ..... 15
1.4 Зміст звіту ..... 15
1.5 Контрольні запитання ..... 16

**2 Лабораторна робота № 2 Розширення функціонвальності HTML-Сторінки ..... 18**
2.1 Мета роботи ..... 18
2.2 Короткі теоретичні відомості ..... 18
2.2.1 Гіпертекстові посилання ..... 18
2.2.1.1 Абсолютні та відносні адреси ..... 18
2.2.1.2 Посилання на елементи поточної сторінки ..... 20
2.2.1.3 Спеціальні посилання ..... 20
2.2.2 Додавання зображень до сторінки ..... 21
2.2.3 Таблиці в HTML ..... 22
2.2.3.1 Об’єднання комірок таблиці ..... 23
2.2.3.2 Атрибути елементів таблиці ..... 24
2.2.4 HTML-форми ..... 25
2.2.4.1 Атрибути форм ..... 26
2.2.4.2 Елементи форм ..... 26
2.3 Завдання на лабораторну роботу ..... 30
2.4 Зміст звіту ..... 30

2.5 Контрольні запитання ....31

**3 Лабораторна робота № 3 Управління зовнішнім виглядом HTML-сторінки за допомогою каскадних таблиць стилів....33**
3.1 Мета роботи ....33
3.2 Короткі теоретичні відомості ....33
3.2.1 Підключення CSS ....33
3.2.1.1 Таблиця зв'язаних стилів....33
3.2.1.2 Таблиця глобальних стилів....33
3.2.1.3 Внутрішні стилі ....34
3.2.2 Базовий синтаксис. Селектори ....34
3.2.2.1 Селектори тегів ....35
3.2.2.2 Класи....35
3.2.2.3 Ідентифікатори....36
3.2.2.4 Контекстні селектори ....36
3.2.2.5 Групування селекторів ....37
3.2.2.6 Псевдоселектори....38
3.3 Завдання на лабораторну роботу ....39
3.4 Зміст звіту....40
3.5 Контрольні запитання ....41

**4 Лабораторна робота № 4 Блочна модель документа ....42**
4.1 Мета роботи ....42
4.2 Короткі теоретичні відомості ....42
4.2.1 Блочні та рядкові елементи ....42
4.2.2 Властивості блочних елементів....43
4.2.3 Розміри блочного елемента ....44
4.2.4 Керування положенням блочного елемента ....46
4.2.5 Розмітка сторінки за допомогою блочних елементів....47
4.3 Завдання на лабораторну роботу ....50
4.4 Зміст звіту....50
4.5 Контрольні запитання ....51

**5 Лабораторна робота № 5 Відображення документа на багатьох пристроях ....52**
5.1 Мета роботи ....52
5.2 Короткі теоретичні відомості ....52
5.2.1 Кросбраузерне верстання....52
5.2.2 Кросплатформне верстання ....54
5.2.2.1 Завдання початкового розміру екрану....56

5.2.2.2 Одиниці виміру розмірів web-елементів ..... 56
5.2.2.3 Адаптивне та респонсивне верстання ..... 58
5.3 Завдання на лабораторну роботу ..... 61
5.4 Зміст звіту ..... 61
5.5 Контрольні запитання ..... 61

**Лабораторна робота № 6 ..... 63**

**Основи Javascript ..... 63**
6.1 Мета роботи ..... 63
6.2 Короткі теоретичні відомості ..... 63
6.2.1 Синтаксис JavaScript ..... 63
6.2.1.1 Ідентифікатори ..... 63
6.2.1.2 Змінні ..... 64
6.2.1.3 Оператори ..... 65
6.2.1.4 Функції ..... 66
6.2.1.5 Вбудовані об'єкти JavaScript ..... 66
6.2.1.6 Виведення даних в JavaScript ..... 67
6.2.2 DOM модель ..... 68
6.3 Завдання на лабораторну роботу ..... 70
6.4 Зміст звіту ..... 70
6.5 Контрольні запитання ..... 71

**Література ..... 73**

**Додаток А Варіанти завдань до лабораторної роботи №2 ..... 74**

**Додаток Б Варіанти завдань до лабораторної роботи №3 ..... 76**

**Додаток В Варіанти завдань до лабораторної роботи №4 ..... 79**

# ВСТУП

Дане видання призначене для вивчення та практичного засвоєння студентами всіх форм навчання основ web-технологій та web-дизайну. Ця галузь IT-технологій охоплює принципи функціонування мережі Інтернет на прикладному рівні а також включає проектування призначених для користувача web-інтерфейсів для сайтів або web-застосунків.

В даних методичних вказівках містяться основні, базові теоретичні відомості, необхідні для виконання лабораторних робіт. Таким чином для успішного виконання лабораторної роботи та при підготовці до її захисту відповідно до графіка студенти повинні ознайомитися з конспектом лекцій та рекомендованою літературою.

Для одержання заліку з кожної роботи студент здає викладачу цілком оформлений звіт, а також демонструє на екрані комп'ютера результати виконання лабораторної роботи.

Звіт має містити:

– титульний аркуш;

– тему та мету роботи;

– завдання до роботи;

– код програми;

– знімки екрану, що відображають результати роботи;

– змістовний аналіз отриманих результатів та висновки.

Звіт виконують на білому папері формату А4 (210 × 297 мм). Текст розміщують тільки з однієї сторони листа. Поля сторінки з усіх боків – 20 мм. Аркуші скріплюють за допомогою канцелярських скріпок або вміщують у канцелярський файл.

Під час співбесіди при захисті лабораторної роботи студент повинен виявити знання про мету роботи, по теоретичному матеріалу, про методи виконання кожного етапу роботи, по змісту основних розділів оформленого звіту з демонстрацією результатів на конкретних прикладах. Студент повинен вміти правильно аналізувати отримані результати. Для самоперевірки при підготовці до виконання і захисту роботи студент повинен відповісти на контрольні запитання, наведені наприкінці опису відповідної роботи.

# 1 ЛАБОРАТОРНА РОБОТА № 1 ЗАГАЛЬНА СТРУКТУРА HTML-ДОКУМЕНТА

## 1.1 Мета роботи

Ознайомитись із загальною структурою HTML-документа. Створити базову web-сторінку згідно з поточним стандартом мови HTML.

## 1.2 Короткі теоретичні відомості

### 1.2.1 Перед початком роботи. Середовище розробки.

На відміну від PHP, Java, C++ та інших мов, HTML *(HyperText Markup Language)* є не мовою програмування, а мовою розмітки тексту, або, точніше, гіпертексту.

Незважаючи на це, мова HTML має власний синтаксис, який декларується стандартами W3C HTML 4.01 (прийнятий 24 грудня 1999 р. Наразі вважається застарілим) та W3C HTML 5 (прийнятий 28 жовтня 2014 р.).

Фактично HTML документ є текстовим файлом з певною структурою тексту та розширенням .html або .htm, тобто створити його можна у будь-якому текстовому редакторі, навіть у стандартному Блокноті. Але для спрощення процесу розробки рекомендується використовувати програмне забезпечення з функціями підсвічування синтаксису, перевірки орфографії, автодоповнення введення тексту тощо. До таких програм відносяться як більш професійні Adobe Dreamweaver, Microsoft Visual Web Developer, PHP Storm, так і простіші безкоштовні програми з відкритим програмним кодом, наприклад Notepad++.

Щоб створити новий HTML документ у Notepad++ відкрийте програму та оберіть «Файл – Створити», або натисніть комбінацію клавіш Ctrl+N. Після цього ви побачите порожній документ без розширення. Оберіть «Файл – Зберегти як» та назвіть файл, наприклад, index.html, або оберіть тип документа з відповідного випадаючого списку (рис. 1.1).

Після збереження файла, згідно з налаштуваннями Notepad++ за замовчуванням, весь код, що буде додано до цього файла, буде підсвічено за стандартом HTML.

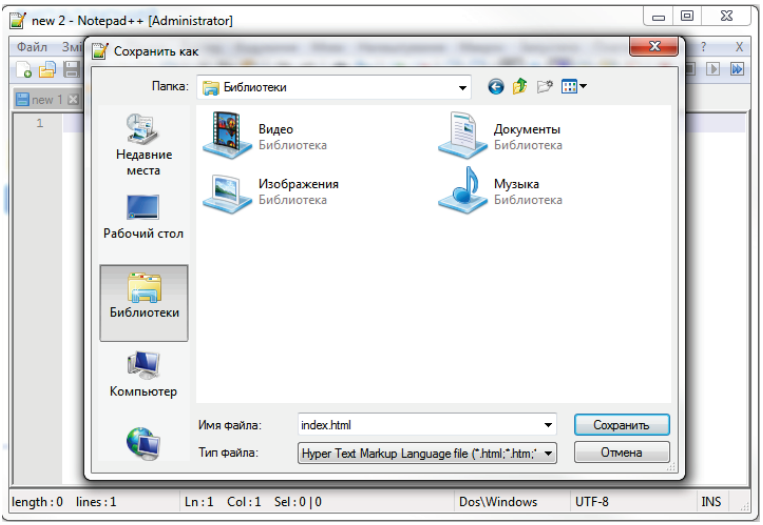



Рисунок 1.1 – Створення та збереження нового HTML-документа

Другим способом задати синтаксис документа є вибір відповідної мови у пункті меню «Мова» (рис. 1.2).

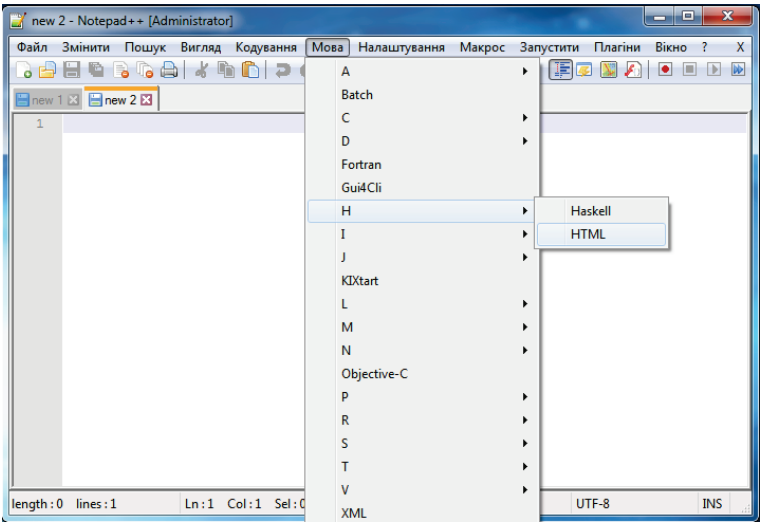



Рисунок 1.2 – Вибір мови документа

### 1.2.2 Елементи HTML. Теги

Елементи – це базові компоненти розмітки HTML. Кожен елемент має дві основні властивості: атрибути та зміст. Існують певні настанови щодо кожного атрибута та змісту елемента, які треба виконувати для того, щоб HTML-документ був визнаний валідним (міг бути коректно обробленим та відображеним браузером).

У елемента є початковий тег, який має вигляд <tag>, та кінцевий тег, який має вигляд </tag>. Атрибути елемента записуються в початковому тегу одразу після назви елемента, зміст елемента записується між його двома тегами. Наприклад:

```
<tag attribute="value">Зміст</tag >
```

Деякі елементи (br, hr, img) не містять змісту, тож не мають і кінцевого тега. В стандарті HTML 4.01 такі теги записуються як відкриваючий тег без кінцевого:

```
<br>
```

В стандарті HTML 5 такі теги позначають як самозакриті:

```
<br />
```

### 1.2.3 Базова структура документа

Окрім видимої користувачу інформації HTML документ включає певну частину службової інформації, яка підказує браузеру, як коректно відобразити даний документ.

HTML

```
<!DOCTYPE HTML>
<html>
<head>
    <title>Заголовок документа</title>
    <meta http-equiv="content-type" content="text/HTML; charset=utf-8" />
</head>
<body>
  Зміст документа
</body>
</html>
```

Першим рядком документа зазвичай є <!DOCTYPE>, який вказує стандарт, згідно з яким браузеру слід інтерпретувати дану сторінку. Стандарти HTML 5 та HTML 4.01 мають деякі відмінності, тож для коректної роботи слід визначити, який з них використовується у даному документі.

Для HTML 5 <!DOCTYPE> має наступний запис:

```
<!DOCTYPE html>
```

А для строгого синтаксису HTML 4.01 використовують наступний формат:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/HTML4/strict.dtd">
```

Тег <html> визначає початок документа, а </html> зазвичай є останнім тегом документа.

Всередині тега <html> документ розділено на дві частини: заголовок документа, який огорнуто тегами <head> </head>, та тіло документа, яке визначається парою тегів <body> </body>.

Заголовок документа включає в себе службову інформацію, яка необхідна для коректної обробки сторінки браузером, пошуковими машинами тощо, і яка не показується користувачу у браузері, окрім змісту тега <title>. Текст, що розміщено всередині парного тега <title>, вважається заголовком сторінки і показується, наприклад, в заголовку вікна браузера, у видачі пошукової машини, тощо.

Весь текст, що розміщено всередині тега <body>, буде показано користувачу у вікні браузера. Це – основний зміст сторінки.

### 1.2.4 Meta-теги

Тег <meta> визначає групу тегів, що використовуються для збереження службової інформації для браузера або пошукової машини. Найважливішими його атрибутами є *name* та *content*. *Name* визначає, яку саме службову інформацію містить мета-тег, а *content* – зміст цієї інформації.

Нижче приведено декілька найбільш популярних мета-тегів, хоча стандарт передбачає значно більше можливих значень.

HTML

```
<meta name="description" content="опис сторінки" />
<meta name="keywords" content="ключові, слова, через, кому" />
<meta name="author" content="автор" />
<meta name="robots" content="index, follow" />
<meta http-equiv="content-type" content="text/HTML; charset=utf-8" />
```

**Description** – включає опис сторінки у вигляді зв'язного тексту.

**Keywords** – перелік ключових слів та словосполучень через кому, за якими бажано знаходити цю сторінку у пошукових системах.

**Author** – визначає ім'я автора сторінки.

**Robots** – дає вказівку пошуковим ботам, чи слід сканувати зміст сторінки (index/noindex) та чи переходити до інших сторінок, на які є посилання в даному документі (follow/nofollow).

Останній наведений тег є прикладом того, як вказати браузеру створити http заголовок сторінки. Даний приклад вказує, яке кодування використовувати для сторінки.

Якщо створена сторінка некоректно відображається у браузері (порожні клітинки або спеціальні символи замість літер), перевірте чи відповідає цей мета-тег кодуванню створеної сторінки. Notepad++ вказує кодування документа в правому нижньому кутку (ANSI або UTF-8). Щоб змінити кодування сторінки, скористайтесь пунктом меню «Кодування – Перетворити в UTF-8 без BOM».

### 1.2.5 Основні теги для оформлення тексту

#### 1.2.5.1 Заголовки тексту

В HTML передбачено до 6 рівнів заголовків, які дозволяють відображати структуру документа. Для того щоб відобразити заголовок на web-сторінці, необхідно використати тег <hx></hx>, де x – ціле число від 1 до 6, що позначає номер рівня заголовка. Найвищим рівнем прийнято вважати 1. Пропускати рівні не рекомендується, тобто після <h1> не повинен бути тег <h3>, якщо між ними немає <h2>.

#### 1.2.5.2 Абзаци (параграфи)

Для поділу тексту на змістовні фрагменти використовується тег .

Браузери за замовчуванням ігнорують невидимі символи, окрім одинарного пробілу. Тобто табуляції та переноси, які буде додано в самому HTML коді буде пропущено та замінено на звичайний пробіл.

Елемент <p> повідомляє браузеру про те, що весь текст, вміщений між його відкриваючим і закриваючим тегами, – це єдиний абзац. Коли інтерпретатор браузера зустрічає в коді відкриваючий тег <p>, він вставляє порожній рядок, позначаючи тим самим початок нового абзацу. За замовчуванням кожен абзац має відступи, що візуально відділяють його від сусідніх елементів.

Якщо треба перенести частину тексту на новий рядок в межах одного абзацу, можна використати тег переносу на нову строку <br />. На відміну від тега <p> він не додає додатковий відступ до тексту.

**1.2.5.3 Елементи фізичного форматування**

Текст може бути візуально виділений напівжирним, курсивом або підкресленням. Ці елементи є абсолютними, а отже, будь-який браузер зобов'язаний відображати їх однаково. Основні елементи фізичного форматування наведені в таблиці 1.1.

Таблиця 1.1 – Елементи фізичного форматування

| Елемент | Призначення |
|---|---|
|   | **Жирний** |
|   | *Курсив* |
| <u> </u> | Підкреслений |
| <sub> </sub> | Нижній $_{\text{індекс}}$ |
| <sup> </sup> | Верхній $^{\text{індекс}}$ |

**1.2.5.4 Елементи логічного форматування**

Логічним є стиль, конкретні параметри якого задаються браузером. Браузер робить текст жирніше, виділяє його курсивом, розфарбовує в зелений колір або вимовляє голосніше (голосові браузери) залежно від можливостей конкретної програми. Будь-який елемент логічного стилю має відкриваючий і закриваючий тег, що формує контейнер для тексту, що виділяється. В таблиці 1.2 наведені елементи логічного форматування.

Зазвичай  відповідає курсиву, а  – жирному шрифту. Але це не завжди так. Іноді  може означати підкреслення, а  – виділення яскравістю. У голосових браузерах текст, позначений , вимовляється голосніше, ніж звичайний, а  – ще голосніше. Основна ідея полягає в тому, що  робить текст виділеним, а  – посилено виділеним.

Таблиця 1.2 – Елементи логічного форматування

| Елемент | Призначення |
|---|---|
|   | Виділення |
|   | Посилене виділення |
| <cite> </cite> | Цитата або посилання на зовнішнє джерело |
| <dfn> </dfn> | Вихідний код програми |
| <samp> </samp> | Приклад роботи програми, часто відображається так само, як і попередній |
| <kbd> </kbd> | Текст, що вводиться з клавіатури |
| <var> </var> | Змінна або значення |

Інші стилі використовуються для деяких специфічних цілей, найчастіше пов'язаних з набором технічної або наукової літератури.

Може здатися, що логічні елементи є надлишковими. Насправді це не так, більше того, саме їм і віддають перевагу, тому що не всі браузери можуть відображати фізичні стилі, наприклад виділення жирним шрифтом. Якщо, наприклад, браузер, убудований у мобільний телефон, не вміє цього, то він просто проігнорує тег <b>. Якщо ж використати логічні елементи, то телефон постарається виділити це місце в тексті якось по-іншому.

**1.2.5.5 Елементи списків**

Список – це структурований перелік подібних об’єктів. У списку в HTML завжди повинно бути два елементи, один із яких задає тип списку, а інший відповідає за один конкретний пункт. Цими пунктами можуть бути слова, речення й інші елементи HTML, наприклад зображення. У більшості списків використовується наступний формат:

HTML

```
<тип_списку>
    <li>Перший пункт</li>
    <li>Другий пункт</li>
    <li>Третій пункт</li>
</тип_списку>
```

Кожен елемент <li> – це пункт списку, що замовчуванням відображається з нового рядка. З чого починається цей рядок, залежить від того, маркований список або нумерований.

Нумерований/упорядкований список задається елементом-обгорткою <ol>, а маркований – елементом <ul>. Але пункти кожного з них позначаються за допомогою <li>. При цьому якщо список упорядкований (<ol>), то в початок кожного рядка вставляється його порядковий номер у списку (1,2,3…); якщо він неупорядкований (<ul>), то вставляється позначка маркування (•).

**1.2.5.6 Базові атрибути елементів оформлення тексту**

Важливим атрибутом, який мають теги заголовків, абзаців та списків є атрибут *align*. Він визначає, який тип вирівнювання буде застосовано до змісту тега. Наприклад

```
<h1 align="center"> Заголовок </h1>
```

означає, що слово "Заголовок", яке є заголовком першого рівня, вирівнюється посередині сторінки.

Можливі значення атрибута *align* наведено у таблиці 1.3.

Таблиця 1.3 – Значення атрибута align

| Значення | Пояснення |
|---|---|
| left | Вирівнювання змісту за лівим краєм |
| center | Вирівнювання змісту за центром |
| right | Вирівнювання змісту за правим краєм |
| justify | Вирівнювання змісту за лівим і правим краєм одночасно |

Слід зазначити, що цей атрибут вважається застарілим в стандарті HTML 5 і нерідко замінюється аналогічним рішенням за допомогою CSS стилів.

### 1.3 Завдання на лабораторну роботу

1.3.1 Ознайомитися з теоретичними відомостями, необхідними для виконання роботи.

1.3.2 Створити HTML-шаблон сторінки, що містить теги, які визначають структуру HTML-документа.

1.3.3 Наповнити створену сторінку текстом, що представляє собою резюме студента. Резюме має містити:

- назву документа;
- персональні дані, такі як прізвище, ім’я, по батькові, дату народження;
- дані про освіту у зворотному хронологічному порядку;
- інформацію про досвід роботи (якщо є);
- професійні навички (загальний рівень володіння комп'ютером, перелік відомих офісних програм та середовищ розробки та рівень знайомства з ними), знання мов;
- контактну інформацію (email, телефон).

1.3.4 Оформити резюме, використовуючи якомога більше тегів форматування тексту: <h1>, <h2>, <h3>, <p>, <b>, <u>, <i>, <ul>, <li> та ін.

Резюме має бути добре структурованим, візуально розділеним на розділи і секції за допомогою заголовків та підзаголовків. Професійні навички та рівень володіння ПЗ слід оформити у вигляді нумерованого списку, а дані про освіту – у вигляді маркованого.

1.3.4 Перевірити створену сторінку за допомогою online-валідатора https://validator.w3.org/ (розділ Validate by Direct Input), та виправити знайдені помилки, якщо такі є.

1.3.5 Оформити звіт з роботи.

1.3.6 Відповісти на контрольні питання, наведені в кінці роботи.

### 1.4 Зміст звіту

1.4.1 Тема та мета роботи.

1.4.2 Завдання до роботи.

1.4.3 Результати виконання роботи у вигляді знімка екрана, що відображає зовнішній вигляд розробленої сторінки у вікні браузера.

1.4.4 Код HTML-документа.

1.4.5 Висновки, що містять відповіді на три контрольних запитання на вибір, а також відображають результати виконання роботи.

### 1.5 Контрольні запитання

1.5.1. Що таке HTML?

1.5.2. Якими стандартами декларується синтаксис мови HTML?

1.5.3. Наведіть три приклади коду, що відрізняється у стандартах HTML 4.01 та HTML 5.

(**Наприклад** HTML 4.01: <br>, HTML 5: <br />)

1.5.4. Які корисні функції пропонує спеціалізоване ПЗ для розробки коду на HTML?

1.5.5. Як увімкнути підсвічування синтаксису HTML в програмі Notepad++?

1.5.6. Які розширення можуть мати файли з HTML кодом?

1.5.7. Які компоненти може мати елемент коду HTML?

1.5.8. Що таке тег?

1.5.9. Де розміщується зміст елемента?

1.5.10. Чому деякі елементи мають кінцевий тег, а деякі – ні?

1.5.11. Наведіть як мінімум три одинарних тега (таких, що не мають кінцевого тега).

1.5.12. Для чого потрібен тег <!DOCTYPE>?

1.5.13. Які структурні теги ви знаєте?

1.5.14. Для чого потрібен тег <html>?

1.5.15. Які частини змісту тега <html> побачить кінцевий користувач у браузері?

1.5.16. Для чого потрібен тег <head>?

1.5.17. Що включає тег <title>?

1.5.18. Для чого потрібен тег <body>?

1.5.19. Що таке мета-теги? Назвіть декілька основних мета-тегів та їхнє призначення.

1.5.20. Що таке кодування сторінки? Як воно впливає на відображення документа у браузері?

1.5.21. Скільки рівнів заголовків передбачено в HTML?

1.5.22. Які текстові теги ви знаєте?

1.5.23. Які теги можуть виконати перенос тексту на новий рядок? Чим вони відрізняються?

1.5.24. Які дії виконує тег <hr />?

1.5.25. Які елементи фізичного форматування вам відомі?

1.5.26. Які елементи логічного форматування вам відомі?

1.5.27. Що спільного і чим відрізняються елементи фізичного та логічного форматування?

1.5.28. Що таке списки? Які види списків передбачає мова HTML?

1.5.29. Наведіть приклад коду маркованого списку.

1.5.30. Наведіть приклад коду нумерованого списку.

# 2 ЛАБОРАТОРНА РОБОТА № 2 РОЗШИРЕННЯ ФУНКЦІОНВАЛЬНОСТІ HTML-СТОРІНКИ

## 2.1 Мета роботи

Навчитися створювати посилання на різні типи документів, використовуючи різні типи URL, та додавати до HTML-сторінок графічні елементи, таблиці та форми.

## 2.2 Короткі теоретичні відомості

### 2.2.1 Гіпертекстові посилання

Гіпертекст – це така форма організації тексту, при якій його фрагменти представлені не в лінійній послідовності, а як система явно вказаних можливих переходів, зв'язків між ними.

Зв’язки та переходи утворюються за допомогою «якоря», тега <a> </a>. Цей тег обов'язково має атрибут *href*, що містить URL (адресу), на який користувач перейде за цим посиланням.

```
<a href="URL">Текст посилання</a>
```

URL – уніфікована адреса, що використовується для однозначної ідентифікації будь-яких web-компонентів.

В загальному випадку URL складається із двох частин: імені протоколу й шляху призначення. Найпоширенішими в Інтернеті протоколами є *http ma https*, за допомогою яких відбувається звернення до web-документів.

Шлях призначення може бути іменем файла, каталогу або комп'ютера в мережі Інтернет чи локальній мережі.

#### 2.2.1.1 Абсолютні та відносні адреси

До будь-якого файла в Інтернеті можна звернутись за абсолютною адресою, наприклад:

```
http://some-domain.com/index.html
http://some-domain.com/readme.txt
http://some-domain.com/logo.png
```

Гіперпосилання виду:

```
<a href="http://some-domain.com/logo.png">Логотип</a>
```

виконає перехід на відповідне зображення незалежно від того, на якій сторінці знаходиться посилання.

Абсолютні посилання використовувати зручно, якщо треба звернутись до файлів, що зберігаються на іншому сервері. Якщо ж файл, на який треба зробити посилання, знаходиться на тому ж сервері що й html-сторінка, ефективніше використовувати відносні посилання. Використання відносних посилань дозволяє уникнути проблем при перенесенні проекту на інший домен, наприклад з *http://some-domain.com на http://some-domain.net*, або у іншу папку на тому ж сервері (наприклад, з *http://some-domain.com/new/* до *http://some-domain.com/release/*).

Відносне посилання дозволяє опустити частину URL, що містить протокол та ім'я домену та виконувати навігацію за папками у межах сервера.

Наприклад, якщо потрібно потрапити на сторінку, розташовану в підкаталозі відносно поточного, можна скористатися URL виду ./new/index.html або ./documents/december/report.html.

На файл у тому ж каталозі, що й вихідний файл, можна зробити відносне посилання двома способами:

destination.html
**.**/destination.html

Крапка зі скісною рискою означають інструкцію шукати файл на тому ж рівні, де знаходиться вихідний файл.

Щоб звернутись до файла, який знаходиться у каталозі на рівень вище, використовують наступний запис:

../destination.html

Дві крапки у даному випадку означають інструкцію піднятись деревом каталогів на один рівень вище до кореня сайта. Таким чином звернення до файла на два рівня вище може виглядати як:

**..**/**..**/destination.html

Посилання наступного виду означає інструкцію шукати відповідний файл у корені сайта (розташування кореня сайта залежить від налаштувань сервера):

/destination.html

При створенні посилань слід додержуватись певних рекомендацій:

– всередині тега <a> </a> має бути зміст, об'єкт, на який користувач може натиснути для здійснення переходу. Це може бути текст або зображення;

– посилання не є ієрархічними. Перш ніж створювати нове посилання, потрібно закрити попереднє;

– можна створювати посилання не тільки на HTML-документи, але й на будь-які інші файли, документи, зображення, тощо. Якщо браузер користувача не може обробити та відобразити такий документ самостійно, він завантажить цей файл для подальшої обробки операційною системою.

**2.2.1.2 Посилання на елементи поточної сторінки**

HTML дозволяє створювати посилання на конкретні місця всередині документа. Так, наприклад, можна організувати зміст на початку великого документа з можливістю переходу на відповідні розділи.

Для цього потрібно спочатку привласнити імена тим частинам, на які необхідно посилатися, а потім встановити якоря, що вказують на них.

Тег, з якого починається потрібний розділ, має містити спеціальний атрибут *id*:

```
<h1 a id="chapter1">Розділ 1</h1></a>
```

Кожне значення *id* має бути унікальним в межах сторінки. Значення *id* може містити букви, цифри, дефіси та підкреслення однак воно повинне починатися з букви.

Гіперпосилання на місце в межах документа виглядає наступним чином:

```
<a href="#chapter1">До розділу 1</a>
```

Можна зробити посилання на конкретне місце в іншому документі. Для цього до звичайного URL документа додається посилання на ідентифікатор.

```
<a href="../section2.html#chapter1">До розділу 1 глави 2</a>
```

**2.2.1.3 Спеціальні посилання**

Спеціальні посилання не виконують безпосередньо перехід до якоїсь сторінки чи документа. Натомість вони використовують

функції операційної системи для виконання певних дій. Найбільш ефективні вони при використанні, наприклад, на мобільних пристроях з сучасним браузером.

Посилання, що дозволяє відправити електронного листа на задану скриньку за допомогою поштового клієнта за замовчуванням:

```
<a href="mailto:info@some-domain.com">Напишіть нам листа</a>
```

Посилання, що дозволяє зателефонувати на певний номер:

```
<a href="tel:01234567890">Консультації</a>
```

Посилання, що дозволяє здійснити дзвінок на певний номер чи обліковий запис через Skype:

```
<a href="callto:01234567890">01234 567 890</a>
```

### 2.2.2 Додавання зображень до сторінки

Для додавання зображення на сторінку використовують тег <img/>. Цей тег інформує браузер про необхідність зарезервувати на сторінці певне місце для зображення.

Тег <img/> має обов'язковий атрибут *src*, що містить URL файла з зображенням. Як і у атрибуті тега <a> цей URL може бути як абсолютним так і відносним.

```
<img src="http://some-domain.com/images/logo.png" />
<img src="./images/logo.png" />
```

HTML дозволяє додавати альтернативний текст, він задається за допомогою атрибута *alt*, що є обов'язковим атрибутом елемента <img />. У ньому міститься рядок тексту, що замінює зображення у випадку неможливості його виводу на екран через якісь причини. Найчастіше цей текст виводиться в окремому прямокутнику, що окреслює те місце, де повинне бути зображення.

Також зображення може мати атрибути *width* та *height*, які містять числові значення та визначають ширину та висоту зображення в пікселях.

```
<img src="http://some-domain.com/images/logo.png" width="64"
height="64" alt="Малий логотип" />
```

### 2.2.3 Таблиці в HTML

У найпростішій таблиці інформація розміщена у комірках, утворених у результаті поділу прямокутника на стовпці та рядки.

Деякі комірки, зазвичай розміщені у верхній або боковій частині таблиці, містять заголовки. У HTML-документах таблицю заповнюють зліва направо, зверху вниз, комірку за коміркою, починаючи з лівого верхнього кута і закінчуючи правим нижнім.

Елемент <table> використовується для того, щоб окреслити межі таблиці та забезпечити контейнер для групи елементів <tr> (table row), що визначають рядки, які в свою чергу складаються з комірок з заголовками <th> (table header) або з даними <td> (table data). Кожна окрема комірка містить дані, при цьому комірки можуть вміщувати все що завгодно – текст, зображення, гіперпосилання тощо.

Тож для створення найпростішої таблиці необхідно використати чотири елементи:

– таблицю описують за допомогою тегів <table>...</table>;

– таблиця повинна мати один або кілька рядків <tr>...</tr>;

– у кожному з рядків може міститися заголовок <th>...</th> або дані <td>...</td>.

Приклад коду для створення таблиці наведено нижче:

HTML

```
<table>
    <tr>
        <th>Заголовок стовпця 1</th>
        <th>Заголовок стовпця 2</th>
        <th>Заголовок стовпця 3</th>
    </tr>
    <tr>
        <td> Комірка  1-1</td>
        <td> Комірка  1-2</td>
        <td> Комірка  1-3</td>
    </tr>
    <tr>
        <td> Комірка  2-1</td>
        <td> Комірка  2-2</td>
        <td> Комірка  2-3</td>
    </tr>
</table>
```

### 2.2.3.1 Об’єднання комірок таблиці

Кілька комірок можуть бути об’єднані в одну як по горизонталі, так і по вертикалі. Об’єднання першого типу застосовують тоді, коли потрібно створити для кількох стовпців спільний заголовок. Коли вміст кількох комірок поспіль у стовпці однаковий, їх об’єднують по вертикалі.

Для об’єднання комірок використовують такі атрибути: *colspan* (об’єднання по горизонталі, у рядку) і *rowspan* (по вертикалі, у стовпці) тега <td>. Значеннями цих атрибутів є кількість об’єднуваних стовпців або рядків. Наприклад, colspan="3" означає, що комірка розтягнута на 3 стовпці, а rowspan="2" – що комірка займає 2 рядки. У цьому випадку комірка, місце якої займає «збільшена» комірка, опускається.

Нижче наведений приклад об’єднання комірок по вертикалі:

| | |
|---|---|
| Студентські групи | КНТ-715 |
| | КНТ-215 |

HTML

```
<table>
    <tr>
        <td rowspan="2"> Студентські групи </td>
        <td>КНТ-715</td>
    </tr>
    <tr>
        <td>КНТ-215</td>
    </tr>
</table>
```

Нижче наведений приклад об’єднання комірок по горизонталі:

| | |
|---|---|
| Студентські групи | |
| КНТ-715 | КНТ-215 |

HTML

```
<table>
    <tr>
        <td colspan="2"> Студентські групи </td>
    </tr>
    <tr>
        <td>КНТ-715</td>
        <td>КНТ-215</td>
    </tr>
</table>
```

### 2.2.3.2 Атрибути елементів таблиці

За замовчуванням ширина таблиці розраховується браузером автоматично згідно зі змістом таблиці. Щоб вказати ширину таблиці вручну використовують атрибут *width* тега <table>, який може мати значення без одиниць виміру, що буде означати ширину в пікселях, або значення у процентах, що буде розраховуватись як зазначена частка ширини батьківського елемента.

```
<table width="320">…</table>

<table width="100%">…</table>
```

Керувати товщиною рамки таблиці та комірок можна за допомогою атрибута *border*, який має числове значення без одиниці виміру і означає товщину рамки у пікселях.

```
<table border="2">…</table>
```

Для таблиці можна задавати відступи від змісту до краю комірки за допомогою атрибута *cellpadding* тега <table>. Значення цього атрибута визначає відстань від межі комірки до її змісту у пікселях і за замовчуванням дорівнює 0.

```
<table cellpadding="5">...</table>
```

Відступи між комірками (відстань між межами комірок) задається атрибутом *cellspacing* тега <table>. Значення цього атрибута задається у пікселях і за замовчуванням дорівнює 0.

```
<table cellspacing="2">...</table>
```

За замовчуванням межа  таблиці та межа комірки є окремими елементами, між якими є невелика відстань. Щоб зовнішні межі комірок утворювали єдину сітку (рис. 2.1), слід використо-

вувати атрибут *border-collapse* тега <table> зі значенням "collapse". За замовчуванням він має значення "separate".

**Border Collapse - Separate**

| Lorem | Ipsum | Dolor |
|---|---|---|
| Lorem | Ipsum | Dolor |

**Border Collapse - Collapse**

| Lorem | Ipsum | Dolor |
|---|---|---|
| Lorem | Ipsum | Dolor |

Рисунок 2.1 – Приклад використання атрибута border-collapse

Зміст у рядках та безпосередньо комірках можна вирівнювати по горизонталі та вертикалі за допомогою атрибутів *align* та *valign*. При цьому слід пам'ятати, що використання *valign* хоч і можливе, але засуджується стандартом HTML5.

Атрибут *align* відповідає за вирівнювання то горизонталі та дозволяє вирівнювати зміст по лівому краю, по центру або по правому краю. За замовчуванням значення цього атрибута – "left".

```
<td align="left | center | right">...</td>
```

Атрибут *valign* відповідає за вирівнювання то вертикалі та дозволяє вирівнювати зміст по верхньому краю, по середині комірки, по нижньому краю або по базовій лінії змісту.

```
<td valign="top | middle | bottom | baseline">...</td>
```

### 2.2.4 HTML-форми

Форми є одним з важливих елементів сайта й призначені для обміну даними між користувачем і сервером.

Для вказівки браузеру де починається й закінчується форма, використовується елемент <form></form>. Між відкриваючим і закриваючим тегами <form> й </form> можна розміщувати

будь-які необхідні теги HTML, а не тільки елементи форми. Це дозволяє форматувати та впорядковувати елементи форми, використовувати зображення тощо. Документ може містити кілька форм, але вони не повинні бути вкладені одна в іншу.

**2.2.4.1 Атрибути форм**

Кожна форма характеризується параметрами, які вказуються в атрибутах тега <form>. Ці параметри задають ім'я форми, її оброблювача (адресу на сервері, що відповідає за отримання та обробку даних), метод відправлення даних на сервер та деякі інші характеристики.

<form name="sample-form" action="/handler.php" method="POST">

Атрибут *action* вказує оброблювач, до якого відправляються дані форми при їхньому відправленні на сервер. Якщо атрибут *action* відсутній або порожній, поточна сторінка перезавантажується, повертаючи всі елементи форми до їхніх значень за замовчуванням.

Атрибут *method* повідомляє серверу про формат відправлення запиту. Розрізняють два методи – GET і POST. Метод GET є одним з найпоширеніших і призначений передачі даних в адресному рядку. Пари "ім'я=значення" приєднуються в цьому випадку до адреси після знаку питання й розділяються між собою амперсандом (&). При використанні методу GET користувач має змогу бачити і змінювати параметри форми безпосередньо в адресному рядку браузера.

http://some-domain.com/handler.php?fname=Ім’я&lname=Прізвище

Метод POST посилає на сервер дані в запиті браузера, приховано від користувача. Це дозволяє відправляти більший обсяг даних, а також, наприклад забезпечити безпеку пересилання паролів.

**2.2.4.2 Елементи форм**

Форма може містити поля для введення текстової інформації, списки для вибору заздалегідь визначених відповідей, прапорці, перемикачі, кнопки та інші елементи керування.

Для всіх способів **введення даних** використовують тег <input> з різними атрибутами.

Для коректної обробки на сервері кожний тег <input> повинен мати атрибут *name*, що задає ім'я, унікальне в межах форми.

В тега <input> є атрибут *type,* завдяки якому цей тег можна використовувати для різних потреб. Наприклад наступні атрибути використовують для створення полів, що очікують введення текстової інформації:

– type="text" – простий текст;

– type="password" – в залежності від налаштувань браузера символи, що вводяться,приховуються символами «*»;

Для полів, що очікують введення тексту, атрибут *value* може задавати початкове значення поля, або значення за замовчуванням. Якщо необхідно вказати не початкове значення, а, наприклад, підказку «Введіть ім'я», слід використовувати атрибут *placeholder*. Його значення не відправляється на сервер, але відображається для користувача як текст всередині поля вводу.

Атрибут *maxlength* дозволяє обмежити максимальну кількість символів для введення в поле.

Наступні елементи відображаються по різному в залежності від браузера та вимагають від користувача відповідної дії:

– прапорець (type="checkbox") використовують, коли можливо вибрати більше одного варіанту з запропонованого списку;

– перемикач (type="radio") використовують, коли необхідно вибрати один єдиний варіант із декількох запропонованих;

– приховане поле (type="hidden") не показується на сторінці й ховає свій вміст від користувача. Його можна використовувати для пересилання службової інформації;

– кнопка (type="reset" – кнопка, призначена для скасування введення;

– кнопка (type="submit") – кнопка, призначена для підтвердження введення. Натискання на неї ініціює пересилання даних на сервер;

– поле із зображенням (type="image") аналогічне по дії кнопці submit, але являє собою малюнок;

– відправлення файла (type="file") використовують для того, щоб відправити на сервер файл. Такий елемент форми відображається як текстове поле, поруч із яким розташовується кнопка «Browse» (або «Файл» в залежності від встановленої мови). При натисканні на цю кнопку відкривається системне вікно для вибору файла.

У загальному виді формат тега <input /> наступний:

<input type="тип" name="ім’я" />

Обов'язковими є атрибути *type* й *name*. Крім того, існує ряд додаткових атрибутів, специфічних для кожного типу <input>.

Для текстів великого обсягу зі смугами прокручування використовують елемент **введення багаторядкового тексту**, парний тег <textarea></textarea> з такими атрибутами:

– *rows* – висота текстового прямокутника в символах;
– *cols* – ширина текстового прямокутника в символах;
– *disabled* – блокує доступ і зміну елемента;
– *readonly* – встановлює, що поле не може змінюватися користувачем;
– *name* – унікальне ім’я в межах форми.

<textarea name="message" rows="3" cols="45">Текст</textarea>

Поле зі списком ще називають **спадаюче меню.** Залежно від налаштувань, у списку можна вибирати одне або кілька значень

Тег <select> дозволяє створити елемент інтерфейсу у вигляді списку, що розкривається, а також список з одним або множинним вибором. Ширина списку визначається найширшим текстом, що знаходиться у тега <option>, а також може змінюватися за допомогою стилів. Кожен пункт створюється за допомогою тега <option>, що повинен бути вкладений у контейнер <select></select>.

HTML

```
<form action="select.php" method="post">
    <select size="3" multiple name="country">
        <option disabled="disabled">Оберіть країну</option>
        <option value="es">Іспанія</option>
        <option value="it">Італія</option>
        <option value="de">Німеччина</option>
        <option value="fr" selected="selected ">Франція </option>
    </select>
    <input type="submit" value="Відправити">
</form>
```

Наступні параметри тега <select>дозволяють змінювати вид і подання списку:

– *multiple* – наявність параметра multiple повідомляє браузеру відображати вміст елемента <select></select> як список множинного вибору;

– *name* – визначає унікальне ім'я елемента;

– *size* – встановлює висоту списку.

Тег <option> має параметри, що впливають на вид списку:

– *selected* – робить пункт списку виділеним;

– *disabled* – робить пункт списку недоступним для вибору;

– *value* – визначає значення пункту списку, що буде відправлено на сервер.

Тег <button> створює кнопки і по своїй дії нагадує тег <input> з атрибутом type = "button | reset | submit". На відміну від цього тега, <button> пропонує розширені можливості оформлення. Наприклад, на подібній кнопці можна розміщувати зображення, текст у декілька рядків тощо.

Стандарт HTML5 запровадив нові, більш **специфічні типи полів вводу** (табл. 2.1).

Таблиця 2.1 – Нові типи тега <input>

| Тип | Описание |
|---|---|
| color | Віджет для вибору кольору |
| date | Поле для вибору календарної дати |
| datetime | Зазначення дати і часу |
| datetime-local | Вказівка місцевої дати і часу |
| email | Адреса електронної пошти |
| number | Введення чисел |
| range | Повзунок для вибору чисел у вказаному діапазоні |
| search | Поле для пошуку |
| tel | Телефонний номер |
| time | Час |
| url | web-адреса |
| month | Вибір місяця |
| week | Вибір тижня |

**2.3 Завдання на лабораторну роботу**

2.3.1 Ознайомитися з теоретичними відомостями, необхідними для виконання роботи.

2.3.2 Використовуючи результати лабораторної роботи №1 доповнити резюме студента наступним чином:

- після назви документа перед розділом «Персональні дані» розмістити власну фотографію невеликого розміру;
- доповнити розділ з контактною інформацією гіперпосиланнями на сторінки в соціальних мережах. При цьому гіперпосиланням на кожну соціальну мережу має бути іконка цієї мережі;
- після переліку соціальних мереж розмістити форму зворотного зв'язку, що містить поля «Ім'я», «Електронна пошта», «Телефон» як input з відповідними типами та «Текст повідомлення» як textarea.

2.3.3 Створити окрему HTML-сторінку, розмістити на ній таблицю зі структурою відповідно варіанту з додатку А, заповнивши її довільними даними. Заштриховані комірки вважати заголовками.

2.3.4 На сторінці резюме після основного змісту розмістити гіперпосилання на сторінку, створену у п. 2.3.3 з текстом «Я вмію створювати таблиці», відділивши його від основного тексту горизонтальною лінією.

2.3.5 Перевірити створені сторінки за допомогою online-валідатора https://validator.w3.org/ (розділ Validate by Direct Input), та виправити знайдені помилки, якщо такі є.

2.3.6 Оформити звіт з роботи.

2.3.7 Відповісти на контрольні питання.

**2.4 Зміст звіту**

2.4.1 Тема та мета роботи.

2.4.2 Завдання до роботи.

2.4.3 Результати виконання роботи у вигляді знімка екрана, що відображає зовнішній вигляд розроблених сторінок у вікні браузера.

2.4.4 Знімок екрана, що відображає структуру файлів проекту.

2.4.5 Код HTML-документа.

2.4.6 Висновки, що містять відповіді на три контрольних запитання на вибір, а також відображають результати виконання роботи.

## 2.5 Контрольні запитання

2.5.1. Дайте визначення терміну гіперпосилання.

2.5.2. Що таке URL?

2.5.3. Чим відрізняються абсолютні та відносні адреси гіперпосилань?

2.5.4. Як відносно звернутись до файла що знаходиться в тому ж каталозі, що й вихідний файл? У батьківському каталозі? У сусідньому каталозі? У кореневому каталозі сайта?

2.5.5. Перерахуйте особливості тега <a>.

2.5.6. Для чого можуть знадобитись посилання на елементи в межах одного документа?

2.5.7. Яка основна вимога до атрибута *id*?

2.5.8. Які спеціальні посилання ви знаєте?

2.5.9. Який тег відповідає за розміщення зображень? Які його обов'язкові атрибути?

2.5.10. Що містить атрибут *alt*? Де може бути відображене його значення?

2.5.11. Що таке таблиця в HTML?

2.5.12. Опишіть базову структуру таблиці HTML?

2.5.13. Що містять теги <tr>, <th>, <td>?

2.5.14. За допомогою яких атрибутів відбувається об'єднання комірок?

2.5.15. Як можна задати ширину таблиці?

2.5.16. Як задається товщина межі таблиці?

2.5.17. Що таке *cellpadding* та *cellspacing*?

2.5.18. Що визначає атрибут *border-collapse*?

2.5.19. Які існують способи вирівнювання змісту в комірці?

2.5.20. Для чого використовуються форми?

2.5.21. Які основні атрибути тега <form>?

2.5.22. Яка різниця між методами GET та POST?

2.5.23. Яке призначення тега <input>?

2.5.24. Які атрибути тега <input> є обов'язковими?

2.5.25. Чим відрізняються <input type="text" /> та <input type="password" />?

2.5.26. Який вигляд має <input type="checkbox" /> ?

2.5.27. Якщо на сторінці є декілька елементів типу radio, як об’єднати їх у групу?

2.5.28. Який вигляд має <input type="radio" /> ?

2.5.29. Що таке <label>, яке його відношення до елементів форми?

2.5.30. Якщо на сторінці є декілька елементів типу checkbox, як поставити <label> у відповідність необхідному елементу?

2.5.31. Що таке <textarea>? Чим цей елемент відрізняється від <input type="text" />?

2.5.32. Як створити спадаюче меню?

2.5.33. Який атрибут дозволяє увімкнути множинний вибір у спадаючому меню?

2.5.34. Що міститься у тега <option>?

2.5.35. Перерахуйте п’ять будь-яких типів тега <input>, впроваджених стандартом HTML5 та поясніть їхні можливості.

# 3 ЛАБОРАТОРНА РОБОТА № 3 УПРАВЛІННЯ ЗОВНІШНІМ ВИГЛЯДОМ HTML-СТОРІНКИ ЗА ДОПОМОГОЮ КАСКАДНИХ ТАБЛИЦЬ СТИЛІВ

## 3.1 Мета роботи

Навчитися управляти зовнішнім видом HTML-сторінки за допомогою каскадних таблиць стилів.

## 3.2 Короткі теоретичні відомості

Каскадні таблиці стилів (css – *cascading style sheets*) – це набір параметрів форматування, що застосовується до елементів web-сторінки для керування їх виглядом і положенням. Стилі є зручним, практичним й ефективним інструментом при розмітці web-сторінок й оформленні тексту, посилань, зображень й інших елементів.

### 3.2.1 Підключення CSS

Для додавання стилів на web-сторінку існує кілька способів, які відрізняються своїми можливостями й призначенням.

#### 3.2.1.1 Таблиця зв'язаних стилів

При використанні таблиці зв'язаних стилів опис селекторів та їхніх властивостей міститься в окремому файлі, як правило, з розширенням css, а для зв'язування документа із цим файлом застосовується тег <link />. Даний тег розміщується в контейнері <head></head>.

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

Атрибут *href* задає шлях до CSS-файла, він може бути заданий як відносно, так і абсолютно. Використання таблиці зв'язаних стилів є найбільш універсальним і зручним методом додавання стилю на сайт, адже стилі зберігаються в одному файлі, а в HTML-документах вказується тільки посилання на нього.

#### 3.2.1.2 Таблиця глобальних стилів

При використанні таблиці глобальних стилів властивості CSS описуються в самому документі й зазвичай розташовуються

в заголовку web-сторінки. За своєю гнучкістю й можливостями цей спосіб додавання стилю поступається попередньому, але також дозволяє розміщувати всі стилі в одному місці. У цьому випадку стилі розміщують прямо в тілі документа, за допомогою елемента <style></style>.

```
<style type="text/css"> ... </style>
```

Таблиця глобальних стилів може розміщуватися не тільки всередині контейнера <head></head>, але також у будь-якому місці коду HTML-документа.

**3.2.1.3 Внутрішні стилі**

Внутрішній стиль є розширенням для одиночного тега, що використовується на web-сторінці. Для визначення стилю використовується атрибут тега *style*, а його значення вказуються за допомогою синтаксису таблиці стилів.

```
<h1 style="font-size: 120%; color: #ff0000">Заголовок</h1>
```

В даному прикладі стиль тега <h1> задається за допомогою параметра *style*, у якому через крапку з комою перераховуються стильові атрибути. Внутрішні стилі рекомендується застосовувати на сайті обмежено або взагалі відмовитися від їхнього використання.

### 3.2.2 Базовий синтаксис. Селектори

CSS має власний стандартизований спосіб запису та у загальному виді має наступний синтаксис.

```
селектор {
    властивість1:значення;
    властивість2:значення;
    ...
}
```

Селектором називається ім'я стилю, в якому зазначені параметри форматування. Селектор дозволяє групувати властивості за певною ознакою. Селектори поділяються на декілька типів: селектори тегів, ідентифікатори й класи. Також існує таке поняття як псевдоселектори (псевдокласи).

Після вказівки селектора йдуть фігурні дужки, у яких записується необхідна стильова властивість, її значення вказується

після двокрапки. Параметри розділяються між собою крапкою з комою, в кінці цей символ можна опустити. CSS не чутливий до регістру, переносу рядків, пробілів і символів табуляції. Імена селекторів обов'язково повинні починатися з латинського символу (a-z, A-Z), можуть містити цифри, дефіс або знак підкреслення.

### 3.2.2.1 Селектори тегів

Селектором може виступати будь-який тег HTML, для якого визначаються правила форматування. При цьому зазначені правила будує застосовано до всіх відповідних тегів у документі.

Правила задаються в наступному виді:

```
тег {
    властивість1:значення;
    властивість2:значення;
    ...
}
```

Наприклад:

CSS

```
button {
        background: #fff;
        border: 2px solid #ff0000;
        border-radius: 5px;
        color: #ff0000;
        font-size: 18px;
        font-weight: bold;
        text-transform: uppercase;
}
```

В даному прикладі всі кнопки, визначені тегом button будуть мати білий фон (#fff), червону межу товщиною 2 пікселя, та жирний текст червоного кольору, розміром 18 пікселів де всі літери прописні.

### 3.2.2.2 Класи

Класи застосовують, коли необхідно визначити стиль для групи елементів сторінки або задати різні стилі для одного тега. Ім’я класу в CSS починається з крапки. Базовий синтаксис для класів наступний:

```
.клас {
    властивість1:значення;
    властивість2:значення;
}
```

Щоб вказати в коді HTML, що тег повинен мати певний клас, до тега додається атрибут *class*="ім'я класу". Імена класів вибираються за бажанням розробника, і можуть відображати суть застосованого стилю, розташування або призначення елемента, тощо. Бажано, щоб вони були зрозумілі й відповідали їхньому використанню.

Класи зручно використовувати, коли потрібно застосувати стиль до різних тегів web-сторінки: комірок таблиці, посилань, параграфів.

**3.2.2.3 Ідентифікатори**

Ідентифікатор визначає унікальне ім'я елемента, та дозволяє однозначно ідентифікувати елемент на сторінці.

В CSS назва ідентифікатора починається зі знаку «решітка». Синтаксис використання ідентифікатора наступний.

```
#ідентифікатор {
    властивість1:значення;
    властивість2:значення;
}
```

**3.2.2.4 Контекстні селектори**

Одному і тому самому елементу сторінки можна призначити стилі багатьма способами: за іменем тега, за класом, за ідентифікатором. Також селектори можна групувати, визначаючи більш вузький перелік елементів, до яких буде застосовано стиль.

Наприклад можна одночасно встановити більш глобальний стиль для тега, стиль для цього ж тега, що має певний клас та окремий стиль для тега, що перебуває всередині іншого. При цьому стилі, які не було перепризначено, успадковуються від більш глобального стиля.

Контекстний селектор складається із простих селекторів, розділених пробілом. Так, для селектора тега синтаксис буде наступний:

```
тег1 тег2 {
    ...
}
```

В даному випадку стиль буде застосовуватися до тега2 тільки якщо він розміщується всередині тега1, як показано нижче:

```
<тег1>
    <тег2> ... </тег2>
</тег1>
```

Не обов'язково контекстні селектори містять тільки один вкладений тег. Залежно від ситуації припустимо застосовувати два й більш послідовно вкладених один в інший тегів. Більш широкі можливості контекстні селектори дають при використанні ідентифікаторів і класів. Це дозволяє встановлювати стиль тільки для того елемента, що знаходиться всередині певного класу.

При такому підході легко управляти стилем однакових елементів, оформлення яких повинне розрізнятися в різних областях web-сторінки.

CSS

```
/* Посилання синього кольору з розміром літер 14 пікселів */
a{
    color:#0000ff;
    font-size:14px;
}
/* Посилання з класом bigger буде синього кольору з розміром літер
16 пікселів */
        a.bigger{
            font-size: 16px;
        }
/* Посилання, що знаходиться всередині маркованого списку, буде
червоного кольору з розміром літер 14 пікселів */
        ul a {
            color:#ff0000;
        }
/* Посилання з класом bigger, що знаходиться всередині маркованого
списку, буде червоного кольору з розміром літер 16 пікселів */
```

### 3.2.2.5 Групування селекторів

При створенні стилю для сайта, коли одночасно використовується багато селекторів, можлива поява повторюваних параметрів. Щоб не повторювати двічі однакові властивості, їх можна згрупувати для зручності подання й скорочення коду. Селектори

групуються у вигляді списку тегів, розділених між собою комами. У групу можуть входити не тільки селектори тегів, але також ідентифікатори й класи. Загальний синтаксис наступний.

```
селектор1,
селектор2,
...
селекторN {
    властивість1:значення;
    властивість2:значення;
}
```

### 3.2.2.6 Псевдоселектори

Псевдоселектори – це спеціальні ключові слова, що дозволяють описати стан елемента (найбільш актуально для посилань), положення в структурі сторінки, тощо. Псевдоселектор записується через двокрапку після селектора елемента, до якого він відноситься.

Для посилань використовують псевдоселектори :link, :hover, :visited та :active.

– :link – відповідає за початкові стилі посилання;
– :hover – стан об'єкта при наведенні на нього курсору;
– :active – стан активного об'єкта, наприклад, під час натискання мишкою;
– :visited – стан відвіданого посилання.

Псевдоселектор :hover можна застосувати до будь-якого елемента на сторінці. Наприклад фон кнопки при наведенні буде змінюватись на червоний:

CSS

```
button {
        background: #fff;
}

        button:hover{
                background:#ff0000;
        }
```

Оскільки HTML є ієрархічною структурою, за допомогою псевдоселекторів можна звертатись до конкретних елементів відносно ієрархії:

– :first-child – перший дочірній елемент поточного елемента;

– :last-child – відповідно, останній дочірній елемент;

– :only-child – застосовує стиль до елемента, якщо він єдиний дочірній елемент;

– :nth-child() – конкретний за порядком дочірній елемент, рахуючи за початку (p:nth-child(4) – четвертий абзац);

– :nth-last-child() – конкретний за порядком елемент, рахуючи з кінця.

В селекторах :nth-child() та :nth-last-child() можна задавати не тільки конкретні цифри, але й ключові слова та лічильники, наприклад :nth-child(2n) та :nth-child(even) – всі парні елементи.

У даному прикладі кожен другий абзац буде мати світло-сірий фон, а у першого абзацу шрифт буде більший, ніж в інших.

CSS

```
p{
        font-size:14px;
}

        p:first-child{
                font-size:16px;
        }

        p:nth-child(2n) {
                background: #eeeeee;
        }
```

### 3.3 Завдання на лабораторну роботу

3.3.1 Ознайомитися з теоретичними відомостями, необхідними для виконання роботи.

3.3.2 Використовуючи результати лабораторної роботи №2 доповнити резюме студента наступним чином:

– змінити зовнішній вигляд документа, використовуючи каскадні таблиці стилів: змінити відступи між елементами, їхнє вирівнювання; змінити колір, розмір та насиченість текстів заголовків всіх рівнів та основного тексту. *При виконанні цього завдання слід дотримуватись певного стилю, щоб створюваний документ виглядав про-*

*фесійно та зручно для читання. Наприклад, слід використовувати не більше двох шрифтів на одній сторінці (один для заголовків, інший для основного тексту), не більше 2-3 яскравих кольорів, не дуже маленький розмір літер, тощо;*

- додати до документа фон згідно з варіантом з додатку Б, використовуючи де можливо колір та повторювані зображення.

3.3.3 Змінити таблицю, створену при виконанні лабораторної роботи №2 наступним чином:

- призначити кожній комірці таблиці унікальний клас. Використовуючи псевдоселектори зробити так, щоб при наведенні та натисканні на комірку вона змінювала колір;
- зробити так, щоб комірки останнього рядка змінювали колір плавно протягом деякого часу (0,5 – 2с на вибір студента).

3.3.4 Оформити звіт з роботи.

3.3.5 Відповісти на контрольні питання.

### 3.4 Зміст звіту

3.4.1 Тема та мета роботи.

3.4.2 Тема, обрана для проектування.

3.4.3 Результати виконання роботи у вигляді знімка екрана, що відображає зовнішній вигляд розроблених сторінок у вікні браузера.

3.4.4 Знімок екрана, що відображає структуру файлів проекту.

3.4.5 Код HTML-документа.

3.4.6 Код CSS-документа.

3.4.7 Висновки, що містять відповіді на три контрольних запитання на вибір, а також відображають результати виконання роботи.

### 3.5 Контрольні запитання

3.5.1. Що таке CSS?

3.5.2. Як підключити таблицю зв'язаних стилів у документі?

3.5.3. Що таке таблиця глобальних стилів?

3.5.4. У чому перевага використання таблиці зв’язаних стилів над таблицею глобальних стилів?

3.5.5. Що таке внутрішній стиль тега?

3.5.6. Який атрибут дозволяє підключити до тега внутрішній стиль? Який його базовий синтаксис?

3.5.7. Що таке селектор? Які існують селектори?

3.5.8. Що таке клас в CSS? Який синтаксис запису селектора класу?

3.5.9. Як задається клас елемента в HTML?

3.5.10. Чи може елемент мати декілька класів? Чи можуть декілька елементів мати однаковий клас?

3.5.11. Що таке ідентифікатор елемента в HTML?

3.5.12. Чи може елемент мати декілька ідентифікаторів? Чи можуть декілька елементів мати однаковий ідентифікатор?

3.5.13. В чому різниця при використанні класів та ідентифікаторів?

3.5.14. Що таке контекстний селектор? Наведіть кілька прикладів.

3.5.15. Навіщо може знадобитись групування селекторів? Наведіть приклад.

3.5.16. Що таке псевдоселектор?

3.5.17. Які псевдоселектори застосовують для посилань?

3.5.18. Яка функція псевдоселектора :hover?

3.5.19. Поясніть принцип використання псевдоселекторів :first-child, :last-child, :nth-child() та :nth-last-child().

3.5.20. Наведіть приклад використання псевдоселектора :not().

3.5.21. За допомогою якого стилю можна змінити колір фона документа?

3.5.22. Як додати межу певного кольору та товщини до елемента?

3.5.23. Наведіть три стиля для керування шрифтом елемента.

# 4 ЛАБОРАТОРНА РОБОТА № 4
# БЛОЧНА МОДЕЛЬ ДОКУМЕНТА

## 4.1 Мета роботи

Вивчити способи групування та організації елементів. Навчитись керувати розташуванням елементів на сторінці.

## 4.2 Короткі теоретичні відомості

### 4.2.1 Блочні та рядкові елементи

Кожний тег HTML обов'язково є або блочним (block) або рядковим (inline) елементом. Принципова різниця між цими двома типами полягає в тому, як браузер відображає положення цього елемента відносно сусідніх.

Блочні елементи можна представляти як прямокутні області на сторінці. Вони мають такі особливості:

– до і після блочного елемента існує перенесення рядка;

– блочним елементам можна задавати ширину, висоту, внутрішні і зовнішні відступи;

– блочні елементи займають весь доступний простір по горизонталі.

Відповідно рядковий елемент не має керованої висоти чи ширини і відображається поряд з іншими рядковими елементами.

До блочних елементів відносяться, наприклад, теги <p>, <ul>, <h1> та інші. До рядкових – <b>, <i>, <ul>. Кожен елемент можна примусово відображати як блочним, так і рядковим, використовуючи CSS властивість *display:*

– display: block;

– display: inline;

– display: inline-block.

Окрім вже згаданих вище тегів існують спеціальні теги, які виступають як контейнери і дозволяють структурувати сторінку.

 – блочний елемент, що позначає просто «блок» або «прямокутний контейнер». Цей тег найчастіше використовується для створення сіток розмітки сторінки (заголовок сторінки, підвал, бокова колонка тощо). На відміну від тегів <h1>, <p> та

інших, які мають певний набір властивостей за замовчуванням, у тега  є одна властивість – display: block.

Тег  – це рядковий контейнер, який можна використовувати для виокремлення певної частини контенту, не застосовуючи ніяких стилів за замовчуванням як <b> чи <i>. За замовчуванням тег  має властивість display: inline.

У HTML5 додано кілька нових тегів розмітки для позначення різних типових блоків сторінки. Наприклад, <header> і <footer> використовуються для створення «шапки» і «підвалу», <nav> для навігації, <aside> для бічній панелі. Включення в специфікацію HTML подібних елементів покликане знизити використання тега  підвищити наочність коду.

### 4.2.2 Властивості блочних елементів

Блочний елемент складається з набору властивостей, які сумарно завдають його розміри на сторінці. Основою блоку виступає його контент (це може бути текст, зображення або інші вкладені в нього блоки). Навколо контенту йдуть поля (*padding*), вони створюють порожній простір від контенту до внутрішнього краю меж елемента. Потім йдуть власне самі межі (*border*). Завершують структуру блока відступи (*margin*), невидимий порожній простір від зовнішнього краю кордонів до сусіднього елемента. На рис. 4.1 показаний блок у вигляді набору цих властивостей.

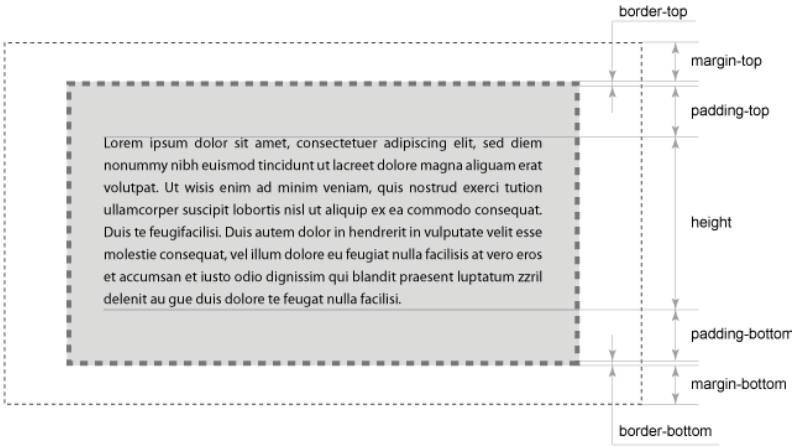



Рисунок 4.1 – Властивості блочного елемента

Значення кожної з властивостей padding, margin, border можна задавати як однакові для всіх чотирьох сторін так і окремо для кожної. У цьому випадку вони перераховуються через пробіл за годинниковою стрілкою починаючи зверху (top right bottom left).

Нижче наведено кілька варіантів запису на прикладі властивості *margin.*

CSS

```
/* Всі відступи дорівнюють 10px */
margin: 10px;

/* Верхній та нижній відступи 10px, лівий та правий 20px */
margin: 10px 20px;

/* Верхній відступ 10px, лівий та правий 20px, нижній 0  */
margin: 10px 20px 0;

/* Верхній відступ 10px, лівий 5px, правий 20px, нижній 0  */
margin: 10px 20px 0 5px;
```

Кожною стороною блока можна керувати окремо, використовуючи властивості *margin-top, margin-bottom, margin-right, margin-left, padding-top, padding-bottom, padding-right, padding-left та border-top, border-bottom, border-right, border-left.*

### 4.2.3 Розміри блочного елемента

Для блочного елемента можна задавати ширину та висоту за допомогою властивостей *width* та *height*. За замовчуванням значення, задані цими властивостями, є параметрами тільки контенту всередині блока. Тобто, наприклад, місце, яке весь блок займає по ширині, буде складатись з суми значення *width*, значень лівого та правого полів, значень ширини лівої та правої межі, та значень відступів.

Завдяки властивості *box-sizing*, яку було введено в стандарті CSS3, керувати шириною і висотою можна наступним чином:

– box-sizing: content-box – значення за замовчуванням; властивості *width* і *height* задають ширину і висоту контенту і не включають в себе значення відступів, полів і меж;

– box-sizing: border-box – властивості *width* і *height* включають в себе значення полів і меж, але не відступів (*margin*);

– box-sizing: padding-box – властивості *width* і *height* включають в себе значення полів, але не відступів (*margin*) і меж (*border*).

Ширину блока можна задавати не тільки в абсолютних одиницях, але й у процентах. В цьому випадку ширина буде розраховуватись як частка ширини батьківського блоку. Існують властивості *max-width, min-width, max-height, min-height*, які дозволяють обмежити розміри блока у разі динамічного розрахунку.

Якщо після завдання розмірів зміст блоку виявляється більшим за розміром, ніж може поміститись в блок, контент вийде за межі блоку. Для керування відображенням у такому випадку використовують властивість *overflow* і введені в стандарті CSS3 властивості *overflow-x* та *overflow-y*.

– overflow: visible – відображається весь вміст елемента, навіть за межами встановленої висоти і ширини;

– overflow: hidden – відображається лише область всередині елемента до правої чи нижньої межі відповідно, все, що не вміщується,буде приховано;

– overflow: scroll – до блока завжди додаються смуги прокрутки.

– overflow: auto – смуги прокрутки додаються тільки при необхідності, якщо контент більший за розміри блока.

CSS

```
/* блок з класом container шириною 100% батьківського блоку, але не більше 1000px і не менше 320px, та з висотою 400px. Контент, що не вміщується по горизонталі, буде приховано. Контент, що не вміщується по вертикалі буде викликати появу смуги прокрутки всередині блоку */
div.container{
        width:100%;
        max-width:1000px;
        min-width:320px;
        height:400px;
        overflow-x:hidden;
        overflow-y:auto;
}
```

### 4.2.4 Керування положенням блочного елемента

Навіть якщо обмежити ширину блочного елемента, він все одно буде займати цілий рядок 100% в ширину. Щоб розмістити кілька блочних елементів поруч, використовують властивість *float*. Ця властивість визначає, по якій стороні буде вирівнюватися елемент. При цьому інші елементи будуть обтікати його з інших сторін.

– float: left – вирівнює елемент по лівому краю, а всі інші елементи, обтікають його по правій стороні;

– float: right – вирівнює елемент по правому краю, а всі інші елементи обтікають його по лівій стороні.

Щоб припинити обтікання та відображати елемент з нового рядку використовують властивість *clear*. Вона встановлює, з якого боку елемента заборонено його обтікання іншими елементами. Якщо задано обтікання елемента за допомогою властивості *float*, то *clear* скасовує його дію для зазначених сторін, при чому ця властивість застосовується до першого елемента, який має не обтікати елемент з властивістю *float*. Можливі значення: both, left, right, none.

Браузер відображає елементи на сторінці у тій послідовності, в якій вони вказані в коді HTML базуючись на заданих розмірах. Але є можливість примусово змінювати положення елемента відносно інших елементів. Для цього використовується властивість *position*, яка встановлює спосіб позиціонування елемента відносно вікна браузера або інших об'єктів на сторінці. Для того, щоб побачити результат її застосування до блока, слід використовувати декілька з властивостей *left, right, top, bottom,* які в залежності від значення *position* мають різний фізичний зміст.

– position: static – елементи відображаються звичайно. Використання властивостей *left, top, right* і *bottom* не призводить до будь-яких результатів;

– position: relative – положення елемента встановлюється щодо його початкового місця. Додавання властивостей *left, top, right* і *bottom* змінює позицію елемента і зрушує його в ту чи іншу сторону від первісного розташування. Сусідні блоки ігноруються, тобто зміщений елемент може «наповзати» на сусідні;

– position: absolute – вказує, що елемент абсолютно позиціонується, при цьому інші елементи відображаються на сторінці немов абсолютно позиціонованого елемента і немає. Положення елемента задається властивостями *left, top, right* і *bottom*. На положення впливає значення властивості *position* батьківського елемента. Так, 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 fixed, relative або absolute, то відлік координат ведеться від краю батьківського елемента.

– position: fixed – за своєю дією це значення близьке до absolute, але на відміну від нього прив'язується до зазначеної властивостями *left, top, right* і *bottom* точці на екрані і не змінює свого положення при прокручуванні сторінки.

Якщо елементи відображаються на сторінці за замовчуванням, зазвичай не виникає проблем з накладенням елементів один на одного. Але якщо задати позиціонування декількох елементів вручну, може виникнути ситуація, при якій вони перекриваються. Будь-які елементи на сторінці можуть накладатися один на одного в певному порядку, імітуючи тим самим третій вимір, перпендикулярний екрану. Кожний елемент може перебувати як нижче, так і вище інших об'єктів сторінки. Їх розміщенням по z-осі управляє *z-index*. Це властивість працює тільки для елементів, у яких значення *position* задано як absolute, fixed або relative.

В якості значень *z-index* використовуються цілі числа (додатні, від’ємні і нуль). Чим більше значення, тим вище знаходиться елемент в порівнянні з тими елементами, у яких воно менше. При рівному значенні *z-index*, на передньому плані знаходиться той елемент, який в коді HTML описаний нижче. Крім числових значень застосовується значення auto – порядок елементів в цьому випадку розраховується автоматично, виходячи з позиції в коді HTML і позиції батька, оскільки дочірні елементи мають той же *z-index*, що їх батьківський елемент.

### 4.2.5 Розмітка сторінки за допомогою блочних елементів

Щоб реалізувати типову структуру HTML сторінки слід використовувати подібний HTML код:

HTML

```
<div id="header">
        <h1>Заголовок</h1>
</div>

<div id="main">
    <div id="side-menu">
        <h2>Навігація</h2>
        <ul>
                <li><a href="#">Розділ 1</li>…
        </ul>
    </div>
    <div id="content">
        <p>Зміст</p>
    </div>
</div>

<div id="footer">
        <p>…</p>
</div>
```

Або ж, використовуючи терміни HTML5:

HTML

```
<header>
        <h1>Заголовок</h1>
</header>

<div id="main">
    <nav>
        <h2>Навігація</h2>
        <ul>
                <li><a href="#">Розділ 1</li>…
        </ul>
    </nav>
    <div id="content">
        <p>Зміст</p>
    </div>
</div>

<footer>
        <p>…</p>
</footer>
```

Для кожного з використаних блоків слід створити стилі у таблиці стилів, задаючи розміри, відступи та інші налаштування, наприклад:

CSS

```
#header {
        background:#dbdbdb;
        border-bottom:3px solid #777777;
    margin:0 0 30px;
    text-align:center;
}
        #header h1{
                font-size: 36px;
        }

#main {
        margin:0px;
        padding:0px;
        width:100%;
}

#side-menu {
        box-sizing:border-box;
        float:right;
        padding:20px;
        width:20%;
        }

#content {
        float:left;
        width:80%;
}

#footer {
        clear:both;
        margin: 30px 0 0;
        padding:15px;
}

        #footer p{
                font-size:12px;
                text-align: center;
        }
```

### 4.3 Завдання на лабораторну роботу

4.3.1 Ознайомитися з теоретичними відомостями, необхідними для виконання роботи.

4.3.2 Обрати персональну тему за варіантом з додатку В.

4.3.3 Підготувати сторінку, що є енциклопедичною довідкою з теми. За зразок можна взяти структуру сторінки Вікіпедії. При створенні сторінки використати отримані навички:

- структурувати сторінку, розділивши її на змістовні блоки (шапка, підвал, скорочена довідка в правій колонці, підрозділи);
- оформити текст, застосовуючи заголовки, списки, абзаци, зображення. Кожен елемент повинен мати стилі, всі стилі мають бути винесені в таблицю зв'язаних стилів;
- у підвал сторінки додати власне прізвище, що є посиланням на сторінку-резюме, розроблену в попередніх роботах.

4.3.4 Скопіювати проект та розділити сторінку, створену в пункті 4.3.3, на декілька сторінок, кожна з яких містить окремий підрозділ:

- на кожній зі сторінок зберегти бокову колонку зі скороченою довідкою;
- на всіх сторінках між шапкою та основним змістом додати горизонтальне навігаційне меню, що містить посилання на створені сторінки;
- виділяти стилем поточну сторінку в навігаційному меню.

4.3.5 Оформити звіт з роботи.

4.3.6 Відповісти на контрольні питання.

### 4.4 Зміст звіту

4.4.1 Тема та мета роботи.

4.4.2 Тема, обрана для проектування.

4.4.3 Результати виконання роботи у вигляді знімка екрана, що відображає зовнішній вигляд розроблених сторінок у вікні браузера.

4.4.4 Знімок екрана, що відображає структуру файлів проекту.

4.4.5 Код HTML-документів.

4.4.6 Код CSS-документа.

4.4.7 Висновки, що містять відповіді на три контрольних запитання на вибір, а також відображають результати виконання роботи.

**4.5 Контрольні запитання**

4.5.1. Чим відрізняються блочні та рядкові елементи?

4.5.2. Перерахуйте при блочних та три рядкових тега.

4.5.3. Як значення block, inline та inline-block властивості display впливає на відображення елемента?

4.5.4. Що означає тег ?

4.5.5. Що означає тег ?

4.5.6. Які теги HTML5 призвані зменшити надмірне використання тега  для структурування сторінки?

4.5.7. Які властивості впливають на розмір місця, яке блок займає на сторінці?

4.5.8. Що таке padding?

4.5.9. Що таке margin?

4.5.10. Що означає запис padding: 30px 50px 10px;?

4.5.11. Що означає запис margin: 0 auto;?

4.5.12. Як буде виглядати блок з властивістю border-bottom: 3px dotted #ff0000;?

4.5.13. На що впливає властивість box-sizing?

4.5.14. Як задати ширину блока в абсолютних одиницях вимірювання та в процентах?

4.5.15. Як обмежити розмір блока при динамічному розрахунку ширини?

4.5.16. Для чого використовують параметр overflow? У чому різниця між overflow, overflow-x та overflow-y?

4.5.17. Поясніть використання властивості float.

4.5.18. Як відмінити обтікання контенту, задане за допомогою float?

4.5.19. Які значення може приймати властивість position?

4.5.20. Як впливають на розташування елемента властивості left, right, top, bottom при різних значеннях position?

4.5.21. Що таке z-index?

# 5 ЛАБОРАТОРНА РОБОТА № 5 ВІДОБРАЖЕННЯ ДОКУМЕНТА НА БАГАТЬОХ ПРИСТРОЯХ

## 5.1 Мета роботи

Дослідити проблеми відображення web-сторінок на різних пристроях та вивчити способи їх усунення.

## 5.2 Короткі теоретичні відомості

На даний час Інтернетом користуються майже 3,4 мільярдів користувачів по всьому світу. Це 46% всього населення Землі. Очевидно, що вони користуються найрізноманітнішими технологіями. Щоб створити web-документ, який буде доступним максимальній кількості користувачів, слід забезпечити:

- підтримку різних браузерів;
- підтримку різних розмірів екрану;
- підтримку різних платформ.

### 5.2.1 Кросбраузерне верстання

Статистика використання браузерів відрізняється для різних регіонів, але зазвичай виділяють наступні найбільш розповсюджені браузери, за процентом аудиторії у світі станом на грудень 2015:

- Google Chrome – 53.71%
- Mozilla Firefox – 14.29%
- Internet Explorer -15.16%
    - Internet Explorer 11 – 10.26%
    - Internet Explorer 10 – 1.37%
    - Internet Explorer 9 – 1.44%
    - Internet Explorer 6/7/8 – 2.09%
- Apple Safari – 9.93%

- Opera – 2.07%
- Інші – 4.84%

Більшість браузерів запровадили механізм автоматичного оновлення, тобто можна з високою вірогідністю вважати, що користувач Chrome чи Firefox буде мати актуальну версію браузеру. Але застарілі браузери, такі як Internet Explorer 6-11 не мають механізму оновлення а також не завжди підтримують нові технології, впроваджені у стандартах HTML 5 та CSS3. Застосовуючи такі властивості слід перевіряти, чи підтримуються вони необхідними браузерами.

В деяких випадках CSS дозволяє задавати стилі окремо для різних браузерів за допомогою префіксів, кожен з яких є директивою відповідному браузеру: префікс -moz- для Firefox, -ms- для Internet Explorer, -webkit- для Chrome і Safari, -o- для Opera. Їх використання виправдане, якщо необхідно забезпечити підтримку старих версій браузерів, а також для властивостей, що ще чітко не закріплені в специфікаціях та трактуються різними браузерами по-різному.

CSS

```
.elementClass {
   -moz-border-radius: 10px;
   -ms-border-radius: 10px;
   -o-border-radius: 10px;
   -webkit-border-radius: 10px;
   border-radius: 10px;
}

.gradient-fill{
      /* Якщо не підтримуються градієнти */
      background: #00b7ea;

      /* FF3.6-15 */
      background: -moz-linear-gradient(left, #00b7ea 0%, #009ec3
      100%);

      /* Chrome10-25,Safari5.1-6 */
      background: -webkit-linear-gradient(left, #00b7ea 0%,#009ec3
      100%);

      /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
```

```
background: linear-gradient(to right, #00b7ea 0%,#009ec3
100%);
}
```

Спеціально для підтримки застарілих неоновлюваних версій браузера Internet Explorer 6-9 було впроваджено так звані коментарі з умовами (conditional comments). Контент, що розміщується між відкриваючим та закриваючим коментарями, буде застосовано тільки якщо версія браузера відповідає умові:

```
<!--[if IE]>
    Для всіх версій
<![endif]-->
<!--[if IE 6]>
    Тільки для версії 6
<![endif]-->
<!--[if IE 7]>
    Тільки для версії 7
<![endif]-->
<!--[if IE 8]>
    Тільки для версії 8
<![endif]-->
<!--[if IE 9]>
    Тільки для версії 9
<![endif]-->
<!--[if gte IE 8]>
    Для версій від 8 та новіших (gte – greater than or equal)
<![endif]-->
<!--[if lt IE 9]>
    Для версій до 9 (lt – less than)
<![endif]-->
<!--[if !IE]> -->
    Для всіх браузерів крім Internet Explorer 6-9
<!-- <![endif]-->
```

Розробники зазвичай створюють окремий файл зі зв'язними таблицями стилів спеціально для Internet Explorer та підключають його використовуючи умовні коментарі.

### 5.2.2 Кросплатформне верстання

За статистикою у 2015 році доля користувачів, що виходять в Інтернет з мобільних пристроїв вперше в історії перевищила

долю користувачів настільних комп'ютерів та ноутбуків. Очевидно, що між комп'ютерами та мобільними пристроями є велика різниця, в розмірах, пропорціях, доступних технологіях тощо. Наприклад, комп'ютери та ноутбуки використовують горизонтальну орієнтацію екрану, смартфони – вертикальну, а планшети однаково часто використовують обидві. Комп'ютери орієнтовані на керування з курсором, а мобільні пристрої – пальцями. Все це впливає на процес розробки web-сторінок.

Якщо порівняти FullHD монітор з роздільною здатністю 1920x1080px та сучасний смартфон з такою ж роздільною здатністю, можна побачити, що їх фізичні розміри в дюймах відрізняються в рази. Тож якщо задавати розміри елементів на сторінці та розмір літер у пікселях, на одному з типів пристроїв сторінка буде виглядати або дуже малою, або завеликою.


Рисунок 5.1 – Неправильне та правильне відображення сторінки на мобільному пристрої

При порівнянні мобільних і десктопних браузерів найбільш очевидна відмінність – розмір екрану. Мобільні браузери за замовчуванням відображають сайти, створені для звичайних браузерів, набагато гірше, ніж могли б: або шляхом зменшення масштабу, роблячи текст і інший контент занадто дрібним і незручним

для читання, або відображаючи лише невелику частину сайта, яка вміщується на екрані.

У цьому випадку розробник має покладатись на операційну систему та браузер, використовуючи для завдання розмірів відносні параметри.

**5.2.2.1 Завдання початкового розміру екрану**

Щоб забезпечити адекватне відображення в першу чергу на мобільних пристроях, слід вказати браузеру, на яку ширину вікна розраховувати при побудові сторінки. Для цього використовують мета-тег viewport. Він може містити числове значення ширини в пікселях або у вигляді змінної device-width, початкове масштабування сторінки та обмеження для масштабування користувачем.

```
<meta name="viewport" content="width=device-width">
<meta name="viewport" content="320">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=2.0, maximum-scale=2.0 ">
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
```

Задавши мета-тегу viewport значення "device-width", ми повідомляємо, що ширина області перегляду дорівнює ширині цього пристрою або ширині вікна браузера для платформ з підтримкою багатовіконних інтерфейсів.

**5.2.2.2 Одиниці виміру розмірів web-елементів**

Для забезпечення динамічності розмірів елементів сторінки використовують інші одиниці виміру крім пікселя, але треба пам’ятати, що для досягнення найкращого ефекту слід максимально дотримуватись обраної одиниці виміру по всьому документу.

**Піксель (px)** – це базова, абсолютна одиниця виміру. Пікселі мають фіксований розмір і дорівнюють одній точці на екрані комп'ютера (найменший елемент розподільної здатності екрану). Одна з проблем використання пікселів полягає в тому, що ці одиниці не дозволяють динамічно змінювати масштаб для мобільних пристроїв.

**Відносна одиниця (em)** – це одиниця, що масштабується. «em» дорівнює поточному font-size, наприклад, якщо font-size в документі 12px, 1em дорівнює 12px. Можна брати будь-які про-

порції від поточного шрифту: 2em, 0.5em і т.п. Так 2em буде дорівнює 24px, 0.5em дорівнюватиме 6px і т.д.

Реальний розмір в абсолютних одиницях задається для одного батьківського елемента (наприклад <body>), а всі інші розміри розраховуються відносно нього, тобто для зміни масштабу всього документа достатньо змінити одне значення і всю сторінку буде пропорційно збільшено чи зменшено.

```
<div style="font-size:24px">
        Заголовок
        <div style="font-size:24px">
            Вкладений зміст
        </div>
</div>
```

Заголовок
Вкладений зміст

У першому прикладі літери в обох блоках, батьківському і дочірньому, мають однаковий фіксований розмір.

```
<div style="font-size:1.5em">
    Заголовок
    <div style="font-size:1.5em">
        Вкладений зміст
    </div>
</div>
```

Заголовок
Вкладений зміст

У другому прикладі слово «Заголовок» в півтора рази більше базового розміру, що за замовчуванням дорівнює 16px на настільному комп'ютері, а «вкладений зміст» в півтора рази більше заголовка, свого батьківського елемента.

**Проценти (%)** – відносні одиниці виміру, що схожі на «em», за винятком кількох принципових відмінностей. При установці властивості *margin-left* в %, відсоток береться від ширини батьківського блоку, а не від його *margin-left*. При установці властивості *line-height* в %, відсоток береться від поточного розміру шрифту, а не від *line-height* батька. Для *width* / *height* зазвичай відсоток від ширини / висоти батька, але при position: fixed, відсоток береться від ширини / висоти вікна (а не батька і не документа).

### 5.2.2.3 Адаптивне та респонсивне верстання

Завдання фіксованих розмірів для елементів та шрифтів не завжди є найкращим підходом. Також, для різних розмірів екрану може знадобитись задати різне відносне розташування блоків, приховати деякі елементи тощо.

Деякі спеціалісти розрізняють такі поняття як респонсивний та адаптивний дизайн.

Респонсивний дизайн – це динамічне масштабування інтерфейсу під пристрій користувача, що дозволяє задавати різні стилі (або навіть таблиці стилів) в залежності від розподільної здатності екрану, його розмірів і інших характеристик.

Такий підхід втілює концепцію «багато пристроїв – один сайт». Переваги такого підходу:

- однаковий зовнішній вигляд ресурсу в різних браузерах і на різних платформах;
- однакова адресація, сторінка завжди має один і той самий URL;
- розробники підтримують тільки один сайт, що сприяє швидкості розробки.

Хоча позитивні сторони респонсивного дизайну очевидні, у цього методу існує ряд недоліків, і найбільшим з них є швидкість завантаження, яка значно знижується через розмір зображень, орієнтованих на максимально доступну розподільну здатність, і інших візуальних елементів, необхідних для оформлення зовнішнього вигляду ресурсу.

Адаптивний дизайн сегментує користувачів на категорії в залежності від того, з якого пристрою вони переглядають сайт, завдяки концепції «багато пристроїв – багато сайтів».

Сервер визначає, з якого пристрою користувач заходить на сайт, і демонструє ту його версію, яка була розроблена спеціально для цього типу пристроїв. Наприклад замість того, щоб показувати на мобільному пристрої масштабовану копію сторінки, сервер ідентифікує тип пристрою користувача і відображає спрощену версію, яка містить тільки найнеобхідніші елементи інтерфейсу і зображення більш низької якості.

Серед переваг адаптивного дизайну можна виділити наступні:

- завантаження сайта відбувається швидше, так як сервер визначає тип пристрою користувача та завантажує відповідний програмний код та зменшені зображення;
- розробники можуть створювати різні версії сайтів підтримують відповідні типи пристроїв, та робити їх більш зручними для конкретних користувачів.

Однак через необхідність адаптації дизайну до різних пристроїв, час, що витрачається на розробку, значно збільшується, так як зміни необхідно вносити зміни в усі існуючі версії.

На даний час завдяки своїй простоті та швидкості застосування с перший підхід використовується значно частіше. Головним його інструментом, що дозволяє досягти найефективніших результатів, є медіа-запити (media query).

Медіа запит можна використовувати як підключення до документа різних зв'язаних таблиць стилів, так і для виділення окремих глобальних стилів або як конструкцію у CSS файлі:

```
@media <умови зпиту> {
        селектор{
                перелік правил
        }
}
```

В першу чергу директива @media використовується для уточнення типа носія, для якого буде застосовуватися вказаний стиль. Як типів виступають різні пристрої, наприклад, принтер, монітор та ін. Найбільш відомі медіа-типи: screen, print, all.

```
<link rel="stylesheet" media="print" href="print.css">
<link rel="stylesheet" media="screen" href="main.css">
```

Крім того, медіа-запит дозволяє визначати певні умови, пов'язані з розмірами вікна браузера, його пропорціями, орієнтацією екрана тощо. На даний час повністю підтримуються браузерами лише параметри width та height.

Умови можна комбінувати використовуючи логічний оператор **and**. Щоб завдати аналог логічного оператора **or** альтернативні умови перераховують через кому.

CSS

```
@media screen and (max-width:995px), all and (max-height:700px) {
        ...
    }

.container{
    width:100%;
    max-width:1000px;
    font-size:1.2em;
}

@media screen and (min-height: 1000px){
    .container{
        margin:50px auto;
    }
}

@media screen and (min-height: 500px) and (max-height: 999px){
    .container{
        margin:25px auto;
    }
}

@media screen and (max-height: 499px){
    .container{
        margin:10px auto;
    }
}

@media screen and (max-width: 320px){
    .container{
        font-size:0.8em;
    }
}
```

Завдяки медіа-запитам можна змінювати зовнішній вигляд сторінки «на льоту», завдаючи розміри елементів та шрифтів, правила відносного розташування елементів для кожного діапазону розмірів екрана: смартфонів, планшетів, комп’ютерів.

### 5.3 Завдання на лабораторну роботу

5.3.1 Ознайомитися з теоретичними відомостями, необхідними для виконання роботи.

5.3.2 Змінити стилі сторінок, розроблених в лабораторній роботі №4 таким чином, щоб вони стали зручними для використання на пристроях з розмірами екрану від 320px до 1920px. Для перевірки відображення можна використовувати вбудований емулятор інспектора коду Google Chrome.

4.3.3 Оформити звіт з роботи.

4.3.4 Відповісти на контрольні питання.

### 5.4 Зміст звіту

5.4.1 Тема та мета роботи.

5.4.2 Тема, обрана для проектування.

5.4.3 Результати виконання роботи у вигляді знімків екрана, що відображають зовнішній вигляд розроблених сторінок у вікні браузера з шириною 320px, 768px, 1200px, 1920px.

5.4.4 Код HTML-документів.

5.4.5 Код CSS-документа.

5.4.6 Висновки, що містять відповіді на три контрольних запитання на вибір, а також відображають результати виконання роботи.

### 5.5 Контрольні запитання

5.5.1. Перерахуйте відомі вам web-браузери.

5.5.2. Що таке браузерні префікси в CSS властивостях?

5.5.3. Які браузерні префікси ви знаєте?

5.5.4. Для чого потрібні браузерні префікси в CSS властивостях?

5.5.5. Які CSS властивості можуть використовуватись з префіксами?

5.5.6. Що таке коментар з умовами?

5.5.7. Які умови можна використовувати в коментарях з умовами?

5.5.8. Яка область застосування коментарів з умовами?

5.5.9. Що таке роздільна здатність екрана?

5.5.10. Для чого використовують мета-тег viewport?

5.5.11. Які атрибути може мати мета-тег viewport?

5.5.12. Які одиниці виміру розмірів web-елементів ви знаєте?

5.5.13. Які переваги та недоліки побудови інтерфейсу із завданням розмірів у пікселях?

5.5.14. Які переваги та недоліки побудови інтерфейсу із завданням розмірів у відносних одиницях?

5.5.15. Яка різниця між одиницями виміру em та процентами?

5.5.16. Чим відрізняються адаптивний та респонсивний дизайн?

5.5.17. Який селектор активно використовується в інтерфейсі, керованому мишею, але не може бути використаний у інтерфейсі з сенсорним вводом?

5.5.18. Що таке медіа-запит?

5.5.19. Які є способи завдання стилів за допомогою медіа-запитів?

5.5.20. Які типи можна задавати в медіа-запиті?

5.5.21. На що впливає тип, вказаний в медіа-запиті?

5.5.22. Які умови дозволяють встановлювати медіа-запити?

5.5.23. Які існують можливості комбінувати запити?

# ЛАБОРАТОРНА РОБОТА № 6
# ОСНОВИ JAVASCRIPT

## 6.1 Мета роботи

Вивчити основи мови програмування JavaScript та навчитись застосовувати її для створення динамічних сторінок.

## 6.2 Короткі теоретичні відомості

HTML є мовою розмітки і не має вбудованих засобів, які могли б використовуватися для динамічної зміни вмісту сторінки. Цю проблему вирішує використання мови DHTML (Dynamic HTML), що підтримує скриптову мову JavaScript.

Додавання коду JavaScript виконується за допомогою тега <script></script>, який може виділяти фрагмент вбудованого вихідного коду або вказувати на зовнішній файл з вихідним кодом. Додавання скриптів у вигляді окремого файла дозволяє створювати свого роду бібліотеки скриптів і використовувати їх на всіх сторінках сайта.

```
<script type="text/javascript">
    код скрипта
</ script>

<script  type="text/javascript" src="scripts.js"></script>
```

Якщо цей тег використовується в тілі документа (всередині тега body), то виконання скрипта здійснюється в процесі відображення сторінки в браузері. Якщо ж контейнер script описаний всередині тега head, то звернення до скрипта можливо тільки явно, наприклад, через виклик функції.

### 6.2.1 Синтаксис JavaScript

#### 6.2.1.1 Ідентифікатори

Ідентифікатори використовуються в JavaScript як імена змінних. Вони можуть починатися з латинської букви, символу "$" (знак долара) або символу "_" (підкреслення) і повинні складатися з латинських букв, цифр і символів, "$" і "_". Знак долара рекомендується використовувати тільки в ідентифікаторах про-

грам. Ідентифікатори не можуть збігатися з зарезервованими словами (табл. 6.1). Два ідентифікатора вважаються рівними, якщо вони збігаються, тобто JavaScript проводить відмінність між малими та великими літерами.

Таблиця 6.1 – Ключові слова мови JavaScript та її розширень що не можуть вживатися як ідентифікатори

| | | | | |
|---|---|---|---|---|
| break | case | catch | continue | default |
| delete | do | else | finally | for |
| function | if | in | instanceof | new |
| return | switch | this | throw | try |
| typeof | var | void | while | with |
| absract | import | volatile | long | goto |
| double | protected | char | static | package |
| implements | transient | extends | const | synchronized |
| private | byte | interface | float | |
| throws | export | short | native | |
| boolean | int | class | super | |
| enum | public | final | debugger | |

### 6.2.1.2 Змінні

Для оголошення змінних використовується ключове слово *var*. Змінні можна відразу ініціалізувати. Можна оголосити кілька мінливих відразу, розділивши їх комами.

```
var color = "#FFF", fsize = 1024 , total_count = 0, i;
var average = null;
var c = 3;
d = 0; //Помилка!
```

Неініціалізовані змінні матимуть невизначене значення (*undefined*). Ініціалізувати змінні можна в будь-якому місці скрипта, але до першого звернення.

Типи даних змінних в javascript визначаються автоматично. Так само автоматично виконується приведення типів.

Оголошення масивів даних може виконуватися статично та динамічно. Індексування елементів починається з нуля. Елементи масиву можуть бути проініціалізовані при створенні.

```
var weekdays = ["Пн", "Вт", "Ср", "Чт", "Пт"];
myarr;
myarr = new Array(256);
myarr[0] = 255;
```

### 6.2.1.3 Оператори

Коментарі – записи, що призначені для пояснення фрагментів коду або виключення фрагментів коду з обробки та які ігноруються при виконанні програми.

```
// Це однорядковий коментар.
/*
Це багаторядковий коментар.
Він може об'єднувати кілька рядків і його можна використовувати в будь-якому місці програми
*/
```

Умовний оператор **if** призначений для розгалуження програми в залежності від значення (true|false) логічного виразу:

```
if (умова) {блок операторів1} [else {блок операторів2}]
```

Оператор вибору **switch** як і умовний оператор призначений для виконання розгалуження алгоритму, але дозволяє аналізувати безліч можливих результатів перевірки умови. Оператор **break** дозволяє перервати виконання оператора, якщо його не вказати, то будуть виконані всі наступні оператори.

```
switch (умова) {
    case значення1: {блок операторів1; break;}
    case значення2: {блок операторів2; break;}
    case значення3: {блок операторів3; break;}
     [default: {блок операторів за замовчуванням};]
}
```

Цикл з лічильником **for**. Використовується для циклів з заданим числом ітерацій.

```
for ([початкове значення]; [умова]; [приріст])
    {блок операторів;}
```

Цикл з постумовою **do...while**. Виконується, поки умова є істинною. Завжди виконується хоча б один раз.

```
do {блок операторів;} while (умова)
```

Цикл з передумовою **while**. Виконується, якщо умова є істинним. Може не виконатися жодного разу.

```
while (умова) {блок операторів;}
```

Оператори **break** і **continue** використовуються для переривання виконання циклу або завершення поточної ітерації.

Поелементний цикл **for(...in...)** застосовується для виконання команд над кожним елементом масиву або колекції.

```
for (змінна in [масив | об'єкт | колекція]) {блок операторів;}
```

Оператор об'єднання **with** представляє собою звернення до властивостей і методів об'єкта через загальне ім'я.

```
with (об'єкт) {блок операторів;}
```

#### 6.2.1.4 Функції

Функції надають можливість створення повторно використовуваного коду. Функція може приймати параметри і повертати значення в програму, що її викликала. Якщо у функції не передбачено повернення значення, то вона працює як процедура.

```
function ім’яФункції ([список параметрів])
    {блок операторів;
    [return результат;]
    }
```

#### 6.2.1.5 Вбудовані об'єкти JavaScript

JavaScript пропонує розробнику деякий набір бібліотечних функцій, оформлених у вигляді властивостей і методів різних об'єктів.

**Об'єкт Math** представляє собою математичні константи і функції. Константи представлені властивостями об'єкта, а функції – його методами:

- властивості E, LN2, LN10, LOG2E, LOG10E, PI, SQRT1_2, SQRT2;
- методи abs, acos, asin, atan, ceil, cos, exp, floor, log, max, min, pow, random, round, sin, sqrt, tan.

В об’єкт Math вбудовано генератор випадкових чисел.

```
var rnd = Math.round(Math.random()*99)+1;
```

**Об'єкт string** представляє собою рядок символів, виділений одинарними або подвійними лапками, або обчислюваний вираз, який може бути інтерпретовано як рядок.

Для об'єкта string визначені наступні властивості і методи:

- властивості length (довжина рядка);
- методи anchor (якір), bold (напівжирний шрифт), charAt (символ в позиції), fixed (преформат), fontcolor (колір шрифту), fontsize (розмір шрифту), indexOf (індекс першого входження символу), italics (курсив), link (гіперпосилання), substring (підрядок), toLowerCase (рядкові), toUpperCase (прописні).

```
var hello = "Hello,", w = "World!";
var str = hello + w; // Конкатенація рядків
```

### 6.2.1.6 Виведення даних в JavaScript

Результати роботи скрипта JavaScript можуть бути відображені щонайменше двома способами: у вікно поточного документа і в діалогове вікно.

Для виведення даних в документ можна використовувати метод write об'єкта document.

JS

```
document.write ( "Hello, World!");

// Виведення HTML в JavaScript
document.write ( "<h1>Hello, World!</h1>");

// Використання вкладених лапок
document.write ( "<p><a href='#'>Link</a> </ p>");
```

Для виведення різних інформаційних повідомлень, що не відносяться безпосередньо до вмісту web-сторінки слід використовувати методи alert, prompt чи confirm, які є об'єктами window та виводять модальне діалогове вікно, що блокує виконання скрипта до натискання користувачем кнопки в цьому вікні:

- alert виводить просте повідомлення;

• prompt виводить повідомлення і чекає, доки користувач введе текст, а потім повертає введене значення або null, якщо введення скасоване (CANCEL / Esc);

• confirm виводить повідомлення і чекає, поки користувач натисне «OK» або «CANCEL» і повертає true / false.

JS

```
alert( "Повідомлення" );

var result = prompt("Введіть ціле число", 0);

var quit = confirm("Ви впевнені, що хочете закрити сторінку?");
```

### 6.2.2 DOM модель

Основним інструментом роботи і динамічних змін на сторінці є DOM (Document Object Model) – об'єктна модель, використовувана для XML / HTML-документів.

Згідно з DOM-моделлю, документ є ієрархією. Кожен HTML-тег утворює окремий елемент-вузол, кожен фрагмент тексту – текстовий елемент, і т.п., тобто DOM – це уявлення документа у вигляді дерева тегів. Це дерево утворюється за рахунок вкладеної структури тегів та текстових фрагментів сторінки, кожен з яких утворює окремий вузол.

У вузлів є атрибути: style, class, id. Взагалі атрибути теж вважаються вузлами в DOM-моделі, батьком яких є елемент DOM, якому вони належать. Однак, в web-програмуванні атрибути вважають властивостями DOM-вузла, які можна встановлювати і змінювати за бажанням програміста.

Для маніпуляцій з DOM використовується об'єкт **document**, за допомогою якого можна отримувати потрібний елемент дерева і міняти його зміст. З вершини дерева document можна піти далі вниз. Кожен DOM-елемент є об'єктом і надає властивості для маніпуляції своїм вмістом, для доступу до батьків і нащадкам.

• всі дочірні елементи, включаючи текстові, знаходяться в масиві childNodes;

• властивості firstChild і lastChild показують на перший і останній дочірні елементи і дорівнюють null, якщо дітей немає;

• властивість parentNode вказує на батька. Наприклад, для <body> таким елементом є <html>;

• властивості previousSibling і nextSibling вказують на лівого і правого братів вузла.

```
for(var i=0; i<document.body.childNodes.length; i++) {
    var child = document.body.childNodes[i];
    alert(child.tagName);
}
```

Стандарт DOM передбачає кілька засобів пошуку елемента в ієрархії.

Це методи getElementById, getElementsByTagName та getElementsByName.

Найзручніший спосіб знайти елемент в DOM – це отримати його по id. Для цього використовується виклик document.getElementById(id).

Наступний спосіб – це отримати всі елементи з певним тегом, і серед них шукати потрібний. Для цього служить document.getElementsByTagName(tag). Він повертає масив з елементів, що мають такий тег.

Метод document.getElementsByName(name) повертає всі елементи, у яких ім'я (атрибут name) дорівнює даному. Він працює тільки з тими елементами, для яких в специфікації явно передбачений атрибут name: це form, input, a, select, textarea і ряд інших, більш рідко вживаних.

Після знаходження потрібного елемента DOM можна виконувати з ним бажані маніпуляції: отримувати чи змінювати стилі, зміст, тощо.

HTML

```
<p id="message"></p>
<input type="text" id="price" placeholder="Вартість" />
<input type="submit" name="button" value="Зберегти" />
```

JS

```
document.getElementById("price").value = '100 грн.';
document.getElementById("message").innerHTML =
'<b>Вартість оновлено!</b>';
document.getElementById("message").style.color = '#ff0000'
document.getElementsByName("button")[0].class = 'btn-success';
```

### 6.3 Завдання на лабораторну роботу

6.3.1 Ознайомитися з теоретичними відомостями, необхідними для виконання роботи.

6.3.2 Використовуючи результати лабораторної роботи №2 додати валідацію (перевірку) тексту, введеного у поля форми резюме студента. Код валідації розмістити як вбудований вихідний код в тілі сторінки. Виводити помилки, знайдені в результаті перевірки, у вигляді діалогового вікна:

- перевіряти наявність у полі «Ім'я» символів, відмінних від українських чи англійських літер;
- перевіряти, чи відповідає зміст поля «Електронна пошта» стандартному формату адреси електронної пошти (*symbols@symbols.symbols*);
- перевіряти, чи складається зміст, введений в поле «Телефон» тільки з арабських цифр, пробілів та символів "+", "-", "(", ")";
- визначати довжину поля «Текст повідомлення» та виводити її в окремий блок під формою.

6.3.3 Перетворити навігаційне меню на сторінках, розроблених в лабораторній роботі №4, на кнопку, при натисканні на яку з'являється випадаюче меню зі списком сторінок. Код винести в окремий файл та застосувати на всіх сторінках, що мають меню. Впевнитись, що розроблене рішення працює на всіх розмірах екрана, розглянутих в роботі №5.

6.3.4 Створити функцію «Цитата дня», що виводить випадкову цитату із заздалегідь заданого масиву висловлювань. Винести розроблену функцію в окремий файл та викликати її у підвалі сторінок, розроблених в лабораторній роботі №4.

6.3.5 Оформити звіт з роботи.

6.3.6 Відповісти на контрольні питання.

### 6.4 Зміст звіту

6.4.1 Тема та мета роботи.

6.4.2 Тема, обрана для проектування.

6.4.3 Результати виконання роботи у вигляді знімків екрана, що відображають результати роботи: діалогові повідомлення, згорнуте та розгорнуте навігаційне меню, сторінку з цитатою.

6.4.4 Знімок екрана, що відображає структуру файлів проекту.

6.4.5 Код HTML-документів.

6.4.6 Код JS-документів.

6.4.7 Висновки, що містять відповіді на три контрольних запитання на вибір, а також відображають результати виконання роботи.

### 6.5 Контрольні запитання

6.5.1. Що таке DHTML?

6.5.2. Як підключити JavaScript за допомогою тега <script>?

6.5.3. Які особливості підключення <script> всередині <body> та <head>?

6.5.4. Що таке ідентифікатор в JavaScript?

6.5.5. Як оголосити змінну в JavaScript?

6.5.6. Які типи даних передбачає JavaScript?

6.5.7. Які є способи оголосити масив в JavaScript?

6.5.8. Як звернутись до елемента масива за індексом? Який індекс є початковим?

6.5.9. Які є оператори в JavaScript?

6.5.10. Що таке поелементний цикл? Які його переваги над циклом з лічильником?

6.5.11. Яка базова конструкція функції в JavaScript?

6.5.12. Що таке вбудований об'єкт Math? Яке його призначення?

6.5.13. Як за допомогою об'єкта Math обрати випадкове число в заданому діапазоні?

6.5.14. Які операції можна виконувати з об'єктом string?

6.5.15. Яке призначення методу alert?

6.5.16. Яке призначення методу prompt?

6.5.17. Яке призначення методу confirm?

6.5.18. Що таке DOM-модель?

6.5.19. Яка основна концепція DOM-моделі?

6.5.20. Що зберігає масив childNodes?

6.5.21. Що зберігає властивість parentNode?

6.5.22. Для чого потрібні властивості previousSibling і nextSibling?

6.5.23. Як знайти елемент по його ідентифікатору?

6.5.24. Як вибрати всі елементи з заданим тегом?

6.5.25. Як вибрати елементи з заданим ім'ям?

6.5.26. Як змінити колір тексту елемента, знаючи його id?

6.5.27. Чим відрізняються властивості innerText, innerHTML, outerText, outerHTML?

6.5.28. Які можливості пропонує властивість classList?

# ЛІТЕРАТУРА

1. Бикнер К. Экономичный Web-дизайн / Кэрри Бикнер. – М.: НТ Пресс, 2005. – 238 с.

2. Гото К. Веб-редизайн: книга Келли Гото и Эмили Котлер / Келли Гото, Эмили Котлер. – 2-е изд. – СПб: Символ-Плюс, 2006. – 416 с.

3. Дронов В.А. JavaScript и AJAX в Web-дизайне / В.А. Дронов. – СПб.: БХВ-Петербург, 2012. – 736 р.

4. Круг С. Веб-дизайн: книга Стива Круга или «Не заставляйте меня думать!" / Стивен Круг. – 2-е изд. – СПб: Символ-Плюс, 2008. – 224 с.

5. Купер А. Интерфейс. Основы проектирования взаимодействия / Алан Купер, Роберт М. Рейманн, Дэвид Кронин, Кристофер Носсел. – СПб.: Питер, 2016. – 720 с.

6. Маркотт И. / Отзывчивый веб-дизайн / Итан Маркотт. – М.: Манн, Иванов и Фербер, 2012. – 176 с.

7. Нильсен Я.Х. Веб-дизайн / Якоб Х. Нильсен. – М.: Символ-Плюс, 2006. – 512 с.

8. Нильсен Я.Х. Дизайн Web-страниц. Анализ удобства и простоты использования 50 узлов / Я. Нильсен, М. Тахир. – М. : Издательский дом «Вильямс», 2002. – 326 с.

9. Пасічник О.Г. Основи веб-дизайну / О.Г. Пасічник, О.В. Пасічник, І.В. Стеценко : [Навч. посіб.]. – К.: Вид. група BHV. – 2009. – 336 с.

10. Пауэлл Т.А. Web-дизайн / Томас А. Пауэлл. – 2-е изд. – СПб. : БХВ-Петербург, 2004. – XVI, 1045 с.

11. Холл М. Программирование для Web. Библиотека профессионала / Марти Холл, Лэрри Браун. – М.: Издательский дом «Вильямс», 2002. – 1264 с.

12. Хольцнер С. HTML5 за 10 минут / Стивен Хольцнер. – 5-е изд. – М. : Издательский дом «Вильямс», 2011. – 240 с.

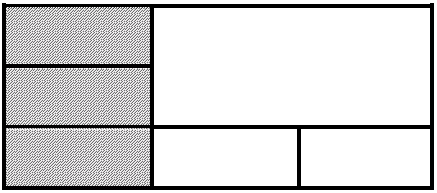
13. Шмитт К. HTML5. Рецепты программирования / К. Шмитт, К. Симпсон. – СПб.: Питер, 2012. – 288 с.

14. Stocks E.J. Sexy web design / E.J. Stocks. – Sitepoint, 2009. – 177 p.

# ДОДАТОК А
# ВАРІАНТИ ЗАВДАНЬ ДО ЛАБОРАТОРНОЇ РОБОТИ №2

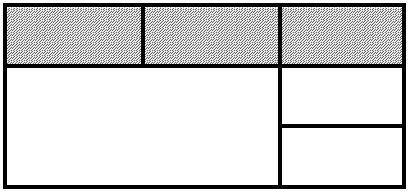
**Варіант 1**

Рамка 1px, ширина 300px



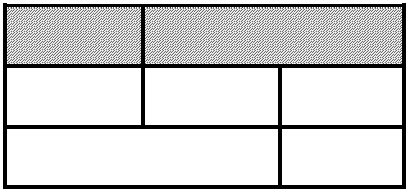
**Варіант 2**

Рамка 2px, ширина 100%



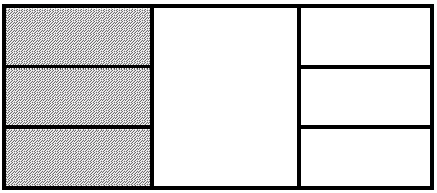
**Варіант 3**

Рамка 3px, ширина 500px



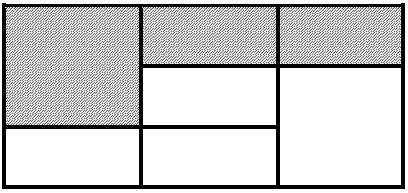
**Варіант 4**

Рамка 4px, ширина 50%



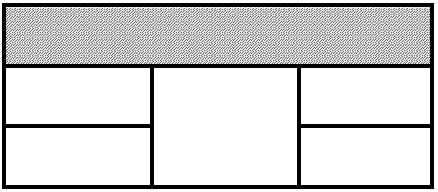
**Варіант 5**

Рамка 5px, ширина 500px



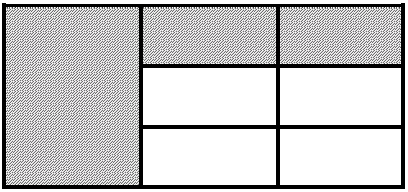
**Варіант 6**

Рамка 5px, ширина 80%



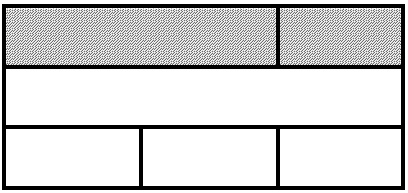
**Варіант 7**

Рамка 1px, ширина 400px



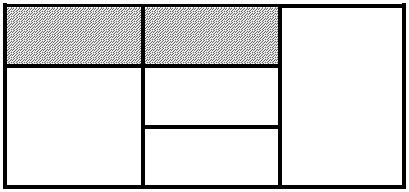
**Варіант 8**

Рамка 2px, ширина 60%



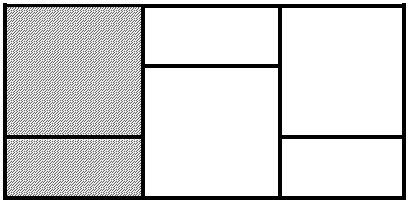
**Варіант 9**

Рамка 3px, ширина 700px



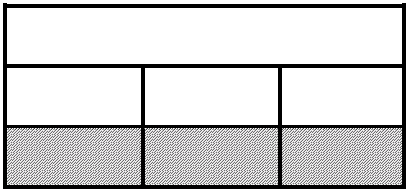
**Варіант 10**

Рамка 4px, ширина 100%



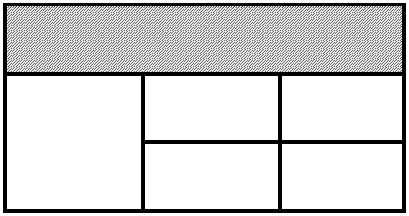
**Варіант 11**

Рамка 5px, ширина 750px



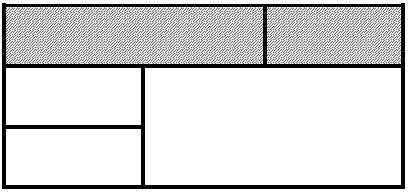
**Варіант 12**

Рамка 1px, ширина 50%



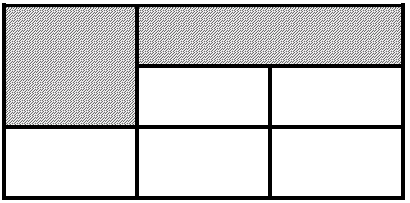
**Варіант 13**

Рамка 2px, ширина 800px



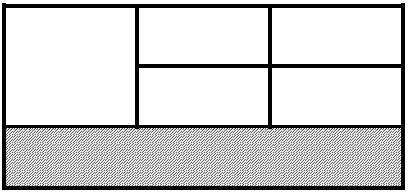
**Варіант 14**

Рамка 3px, ширина 60%



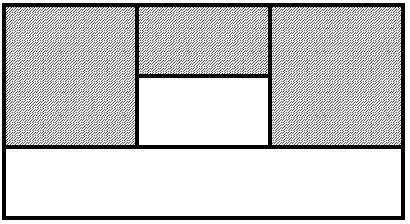
**Варіант 15**

Рамка 4px, ширина 500px



**Варіант 16**

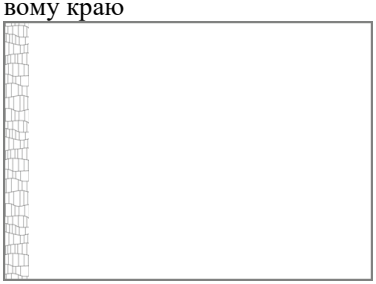
Рамка 5px, ширина 100%



# ДОДАТОК Б
# ВАРІАНТИ ЗАВДАНЬ ДО ЛАБОРАТОРНОЇ РОБОТИ №3

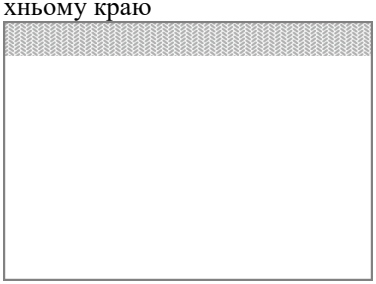
**Варіант 1**

Повторюване зображення по лівому краю



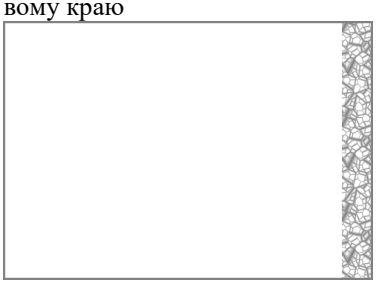
**Варіант 2**

Повторюване зображення по верхньому краю



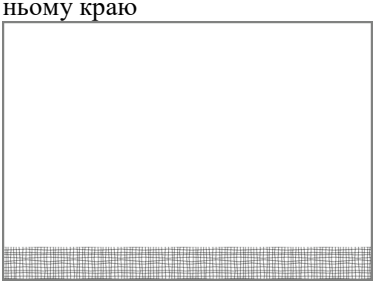
**Варіант 3**

Повторюване зображення по лівому краю



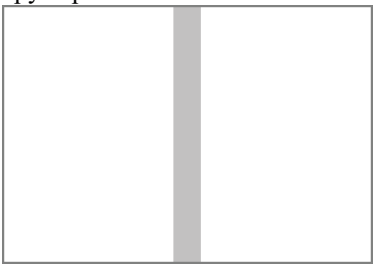
**Варіант 4**

Повторюване зображення по нижньому краю



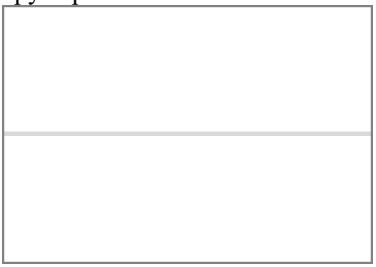
**Варіант 5**

Повторюване зображення по центру вертикально



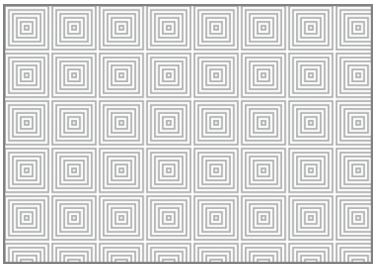
**Варіант 6**

Повторюване зображення по центру горизонтально



**Варіант 7**

Повторюване зображення по всій площині



**Варіант 8**

Фотографія

**Варіант 9**

Лінійний градієнт зверху вниз

**Варіант 10**

Лінійний градієнт зліва направо

**Варіант 11**

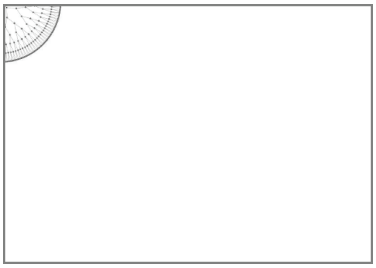
Одиничне зображення по центру


**Варіант 12**

Одиничне зображення по центру поверх кольору

**Варіант 13**

Одиничне зображення у верхньому лівому кутку



**Варіант 14**

Одиничне зображення у верхньому правому кутку

**Варіант 15**

Одиничне зображення у нижньому лівому кутку

**Варіант 16**

Одиничне зображення у нижньому правому кутку

# ДОДАТОК В
# ВАРІАНТИ ЗАВДАНЬ ДО ЛАБОРАТОРНОЇ РОБОТИ №4

| Варіант | Тема | Підрозділи |
|---|---|---|
| 1 | 2 | 3 |
| 1 | Стів Джобс | Скорочена довідка<br>Біографія<br>Досягнення<br>Персонаж у літературі та кіно<br>Цікаві факти |
| 2 | Білл Гейтс | Скорочена довідка<br>Біографія<br>Досягнення<br>Книги Білла Ґейтса |
| 3 | Лінус Торвальдс | Скорочена довідка<br>Біографія<br>Розробка Linux<br>Нагороди та звання |
| 4 | Марк Цукерберг | Скорочена довідка<br>Біографія<br>Створення мережі Facebook<br>Цікаві факти |
| 5 | UNIX | Скорочена довідка<br>Історія<br>Стандарти<br>Файлова система Unix<br>Канонічні команди Unix |
| 6 | Microsoft Windows | Скорочена довідка<br>Версії<br>Інтегровані програмні продукти<br>Популярність |
| 7 | Android | Скорочена довідка<br>Історія<br>Розробка та версії<br>Характеристики<br>Розповсюдженість<br>Переваги та недоліки |

| 1 | 2 | 3 |
|---|---|---|
| 8 | iOS | Скорочена довідка<br>Історія створення та версії<br>Особливості<br>Конкуруючі продукти |
| 9 | Apple | Скорочена довідка<br>Історія<br>Програмне забезпечення Apple |
| 10 | Google | Скорочена довідка<br>Історія<br>сервіси<br>Цікаві факти |
| 11 | Java | Скорочена довідка<br>Історія<br>Концепція<br>Приклад програми |
| 12 | C++ | Скорочена довідка<br>Особливості<br>Приклад програми<br>Хронологія розвитку<br>Переваги та недоліки |
| 13 | Ruby | Скорочена довідка<br>Історія створення<br>Семантика<br>Можливості Ruby |
| 14 | Python | Скорочена довідка<br>Історія<br>Переваги<br>Філософія<br>Приклади програм |
| 15 | PHP | Скорочена довідка<br>Історія<br>Особливості<br>Синтаксис<br>Недоліки |
| 16 | Ліцензії Creative Commons | Скорочена довідка<br>Призначення<br>Існуючі версії ліцензій<br>Вільний вміст |