

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ  
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

**МЕТОДИЧНІ ВКАЗІВКИ**  
**до лабораторних робіт**  
**«Створення веб-сайту за допомогою HTML»**  
з курсу «Основи Internet-технологій»

для студентів спеціальностей 124 «Системний аналіз»,  
186 «Видавництво і поліграфія»

Затверджено редакційно-  
видавничою  
радою університету,  
протокол № 3 від 22.12.16 р.

Харків  
НТУ «ХП»  
2017

Методичні вказівки до лабораторних робіт «Створення веб-сайту за допомогою HTML» з курсу «Основи Internet-технологій» для студентів спеціальностей 124 «Системний аналіз», 186 «Видавництво і поліграфія» / уклад.: Н. А. Марченко, В. О. Колбасін, В. П. Прокопенков. – Харків: НТУ «ХПІ», 2017. – 68 с.

Укладачі: Н. А. Марченко  
В. О. Колбасін  
В. П. Прокопенков

Рецензент Л.М. Любчик

Кафедра системного аналізу та інформаційно-аналітичних технологій

## ВСТУП

Створення глобальної мережі Інтернет докорінно змінило життя багатьох людей. Мережа Інтернет є не тільки новим засобом зв'язку, який об'єднує людей, що мешкають у різних куточках земної кулі, а і стала у певному сенсі новим способом життя. Майже повсякденно ми використовуємо можливості мережі: купуючи товари в Інтернет-магазинах, користуючись послугами Інтернет-телефонії та електронної пошти, перевіряючи прогноз погоди за он-лайн службами та виконуючи багато інших корисних дій за допомогою всесвітньої мережі. Але для повноцінного використання її можливостей треба знати принципи побудови мережі Інтернет та вміти розмовляти на її мові – мові розмітки гіпертекстових документів HTML.

Існує багато систем, які на базі офісного документа дозволяють створити веб-сторінку чи навіть сайт. Ці системи не потребують знання мови HTML та принципів побудови веб-застосувань, але платою за їх використання є нераціональний код html-документа, що створюється. Така сторінка не буде відображатися коректно в усіх браузерах та буде мати завеликий об'єм. Створені таким чином сторінки довго завантажуються, є незручними для кінцевого користувача. Таким чином, користувачі припиняють користуватися сайтами, які створені за допомогою подібних конвертерів.

Отже, без знання мови HTML неможливо стати спеціалістом у галузі інформаційних технологій та впевнено почувати себе на ринку праці.

Методичні вказівки містять опис принципів побудови мережі Веб, мови розмітки HTML та опис найбільш популярних середовищ розробки Microsoft Visual Studio та Eclipse. Завдання для лабораторних робіт наближені до практики та відображають повний цикл розробки статичного веб-застосування.

# 1. ТЕОРЕТИЧНИЙ МАТЕРІАЛ

## 1.1. Принцип роботи мережі веб

Веб є одним з застосувань глобальної мережі Інтернет і являє собою мережу інформаційних ресурсів, що розміщені на спеціальних, потужних комп'ютерах, які називаються *серверами*. До серверів можуть приєднуватися інші комп'ютери, так звані *клієнти*, користувачі яких отримують доступ до ресурсів серверів.

Для багатьох користувачів терміни Інтернет і Веб означають синоніми, але слід врахувати, що у функції мережі Інтернет входить технічне забезпечення роботи мережі Веб, і тільки цим її можливості не вичерпуються. Якщо виникнення мережі Інтернет можна віднести до 1961 р., то мережа Веб виникла в 1992 р. і її розвиток пов'язаний з появою гіпертекстових інформаційних систем.

У 1991 р. Пол Лінднер (Paul Lindner) і Марк П. МакКахілл (Mark P. McCahill) з університету Міннесоти створили систему Gopher, де був запропонований метод організації файлової системи за допомогою інтуїтивно зрозумілих меню. Сервери Gopher швидко поширилися по мережі.

Одночасно розвивався проект об'єднання комп'ютерних мереж європейських країн, рушійною силою якого була Європейська лабораторія фізики елементарних частинок (CERN) у Женеві. Результатом стало втілення концепції розподіленої комп'ютерної мережі, що отримала в 1992 р. назву World Wide Web, автором якої вважається Тім Бернерс-Лі (Tim Berners-Lee). Мережа WWW стала подальшим розвитком системи Gopher.

Для надання своїх ресурсів мережа Веб спирається на наступні *три механізми*:

- єдина система найменувань, за якими ресурси можуть бути знайдені в мережі, що базується на так званих адресах URL (Uniform Resource Locator – уніфікований покажчик інформаційного ресурсу);
- єдиний механізм для організації мережевого доступу до іменованих ресурсів, в якості якого виступає протокол HTTP (HyperText Transfer Protocol);

- єдиний засіб опису гіпертекстових документів, що забезпечує навігацію по ресурсах, він використовується як мова HTML (HyperText Markup Language – мова розмітки гіпертексту).

## 1.2. Базові поняття HTML

HTML (Hyper Text Markup Language, мова розмітки гіпертексту) є фундаментальною, базовою технологією Інтернет. Вона не є мовою програмування, а являє собою мову розмітки. Практично весь вміст веб-вузлів є набором документів, що містять програмний код HTML.

HTML дозволяє формувати на веб-сторінці текстові блоки, включати в них зображення, створювати таблиці, управляти відображенням кольору документа, додавати звуковий супровід і т.п. Файли, що містять гіпертекстовий код, мають розширення .htm або .html.

Однією з основних функціональних особливостей, характерних саме для HTML, завдяки якій вона і отримала свою назву, є посилання.

*Гіперпосилання* (Hyperlink) – це базовий функціональний елемент html-документа, що є реалізацією динамічного зв'язку будь-якого об'єкта даної веб-сторінки з контекстним змістом іншого документа.

На відміну від багатьох інших мов HTML є мовою, що інтерпретується. Вбудований у браузер інтерпретатор створює графічне представлення гіпертекстового документа у процесі відкриття документа. При цьому, виявивши в тексті помилку, він просто ігнорує цей рядок (якщо сторінка не містить вбудованих підпрограм, що називаються *клієнтськими скриптами*).

Первісними можливостями HTML були відображення тексту і організація посилань на інші документи. Потім було додано можливості відображення картинок, виділення тексту за допомогою курсивного або напівжирного накреслення. Розвиток мови HTML здійснювався і здійснюється в теперішній час шляхом випуску нових версій, в які додаються нові можливості. Більшість версій є стандартизованими. Найбільш поширені версії HTML, що використовувалися в Інтернеті:

- HTML 2.0 – 1995 р. – додано підтримку форм для забезпечення взаємодії користувача з сервером;
- HTML 3.2 – січень 1997 р. – були додані численні можливості, наприклад, таблиці, обтікання текстом зображень та ін.;

- HTML 4.0 – грудень 1997 р. – була додана підтримка таблиць стилів, що використовуються для форматування html-документа, скриптів та фреймів;
- HTML 4.01 – грудень 1999 р. – заміна версії HTML 4.0, що містить численні дрібні виправлення;
- HTML 5 – жовтень 2014 р. – має новий словник, побудований на основі HTML 4.01 та XHTML 1.0. Додано велику кількість нових елементів.

У грудні 2012 р. почав розроблятися проект HTML 5.1, де передбачаються такі нові технології як виведення титрів у відеороликах, заповнення форм з автозавершенням, а також перевірка правопису.

### **1.3. Правила створення html-документів**

Згідно з рекомендаціями організації World Wide Web Consortium (W3C), яка займається розробкою стандартів для Інтернет, при створенні html-документів необхідно дотримуватися певних принципів. Основний з них – відділення структури документа від його подання.

Мова HTML надає такі можливості:

- створювати документи із заголовками, текстом, таблицями, списками, зображеннями тощо;
- отримувати інформацію з мережі Веб за допомогою гіпертекстових посилань;
- створювати форми для відправлення запитів на віддалені комп'ютери з метою пошуку даних;
- включати електронні таблиці, відеокліпи, аудіокліпи та інші програмні додатки безпосередньо в документи.

Усі ці можливості забезпечуються трьома компонентами html-документа:

- структурою;
- стилем;
- змістом.

Використання клієнтських скриптів, що дозволяє розробляти інтерактивні html-сторінки, додає до них четверту компоненту – поведінку.

До структури належить все те, що визначає контекст, тобто логічну організацію документа, зокрема, поділ тексту документа на абзаци,

розділи, підрозділи і т. ін. Представлення документа на екрані браузера визначається його змістом і стилем, тобто текстовою інформацією та її форматкуванням.

#### 1.4. Тегова модель документа

*Тегом (дескриптором)* називають спеціальний засіб мови HTML, що управляє розміткою html-документа.

Кожен тег складається з імені тегу, укладеного в кутові дужки. За ім'ям тегу може слідувати необов'язковий список його атрибутів.

Як правило, теги є парними, тобто присутній початковий тег і тег з тим же ім'ям, але відрізняється від нього рискою – кінцевий тег. Парні теги називаються *контейнерними*, а все, що знаходиться між початковими і кінцевими тегами, називається *вмістом елемента HTML*. Теги, що не мають парного кінцевого тегу, називаються *порожніми*, або *одиначними*, а відповідні їм елементи – *автономними*.

Контейнерні теги разом з їх вмістом і одиначні теги являють собою елементи мови HTML. Контейнерні теги HTML разом з їх вмістом іноді також називають *контейнерами*.

Атрибути відокремлюються один від одного одним або декількома знаками табуляції, пробілами або символами переведення каретки. Порядок запису атрибутів у тезі не має значення. Значення атрибута, якщо таке є, слідує за знаком рівності, що знаходиться після імені атрибута. Якщо значення атрибута одне слово або число, то його можна вказати просто після знака рівності. Всі інші значення необхідно укладати в одинарні або подвійні лапки, особливо якщо вони містять декілька розділених пробілами слів.

Якщо деякий атрибут опущений, то він набуває *значення за замовчуванням*. Атрибути, яким присвоєно значення за замовчуванням, як правило, не вказуються.

```
<тег атрибут1="значення1" атрибут2="значення2" ...  
атрибутN-1 атрибутN="деяке значення">Вміст</тег>
```

Атрибут атрибутN-1 є *логічним* і не має значення, тобто його наявність установлює відповідну властивість.

Теги можна вкладати один в одне. При цьому вкладені контейнерні теги потрібно закривати, починаючи з самого останнього і рухаючись до першого:

```
<тег1>Вміст тегу <тег2><тег3> Текст </тег3>  
продовження </тег2></тег1>
```

Регістр символів в іменах тегів і атрибутів не враховується, але, тим не менш, рекомендується набирати їх у нижньому регістрі. Значення атрибутів, як правило, чутливі до регістру. Це особливо характерно при запису URL-адрес.

Якщо який-небудь тег або його атрибут записано невірно, то браузер проігнорує його і буде відображати текст так, ніби відповідного елемента немає. Це створено для можливості додавання нових тегів і атрибутів у мову HTML. Іншими особливостями інтерпретації html-документів є:

- будь-яка кількість пробілів, що стоять поруч, у браузері відображається як один пробіл;
- відсутня розстановка переносів у тексті;
- при відображенні текст займає всю ширину вікна браузера.

## 1.5. Загальна структура html-документа

Згідно з рекомендаціями W3C структура html-документа складається з компонентів:

- рядка із зазначенням номера версії мови HTML;
- декларативного розділу заголовка;
- тіла документа, що містить подану документом інформацію.

HTML слідує правилам, які містяться у файлі оголошення типу документа (Document Type Definition, або DTD). DTD являє собою XML-документ, що визначає, які теги, атрибути та їх значення дійсні для конкретного типу HTML. Для кожної версії HTML існує свій DTD.

З використанням тегової моделі структура html-документа виглядає так:

```
<!DOCTYPE html>  
<html>  
<head>  
<title> Заголовок </title>
```



```
</head>
<body>
Основний текст (контент сайту)
</body>
</html>
```

Розглянемо цю структуру більш докладно.

`<!DOCTYPE html>` – це рядок із зазначенням версії. Він відповідає за коректне відображення веб-сторінки браузером та визначає відповідний DTD-файл в Інтернеті.

`<html>` – тег верхнього рівня ієрархії. Він показує браузеру, що наступний текст треба трактувати як html-документ. Елементи, що знаходяться всередині тегу `<html>`, утворюють дерево документа, так звану *об'єктну модель документа* (Document Object Model, або DOM).

`<head>` – заголовок документа, призначеного для подання необхідної інформації браузеру. Елементи, що знаходяться всередині розділу `<head>` (крім назви документа, що задається за допомогою елемента `<title>`), не відтворюються на екрані.

Тег `<title>` є єдиним обов'язковим елементом заголовка документа і служить для визначення назви документа. Назва являє собою текстовий рядок. Не слід брати її в лапки, якщо немає необхідності, щоб вони відображалися на екрані в назві документа. Зазвичай `<title>` займає не більше одного рядка. Назви документів ураховуються пошуковими машинами, тому їх необхідно робити інформативними.

Ще одним поширеним тегом, що міститься у заголовку, є одиночний тег `<meta>`. Він визначає метатеги, які використовуються для зберігання інформації, призначеної для браузерів і пошукових систем. Наприклад, механізми пошукових систем звертаються до метатегів для отримання опису сайту, ключових слів та інших даних. Дозволяється використовувати більше ніж один тег `<meta>`. Як правило, атрибути будь-якого метатегу є пари "ім'я=значення", які визначаються ключовими словами `content`, `name` або `http-equiv`. Причому неможна одночасно використовувати атрибути `name` і `http-equiv`.

Атрибути тегу `<meta>`:

- `charset` – задає кодування документа, наприклад:

```
<meta charset="UTF-8">;
```

- `content` – установлює значення атрибута, заданого за допомогою `name` або `http-equiv`;
- `http-equiv` – призначений для конвертації метатегу в заголовки HTTP, наприклад `expires` – встановлює дату і час, після якої інформація в документі буде вважатися застарілою; `pragma` – спосіб кешування документа;
- `name` – ім'я метатегу, що також побічно встановлює його призначення, наприклад `author`, що задає ім'я автора документа, `description` – опис документа; `keywords` – перелік ключових слів, що зустрічаються на сторінці.

Тіло документа `<body>` – містить всю інформацію, яку html-документ повинен представити користувачам (текст, графічні знаки, рисунки і т. ін.). Тег `<body>` також може мати такі атрибути:

- `background` – вказує на URL-адресу зображення, яке використовується як фонове;
- `bgcolor` – задає колір фону документа;
- `text` – колір тексту документа;
- `alink`, `link`, `vlink` – вказує на колір відповідно активного, невідданого і відданого гіперпосилання документа.

Для розуміння взаємодії елементів веб-сторінки, необхідно розглянути так звані «родинні стосунки» між елементами. Для цього розглянемо наступний html-документ та його подання у вигляді дерева (рис. 1.1):

```
<!DOCTYPE html>
<html>
<head>
<title>Заголовок документа</title>
<link rel="stylesheet" type="text/css"
href="css1.css">
<script src = "script1.js"> </script>
</head>
<body>
<h1>Назва документа</h1>
<div class="main">
```

```

<p>Текст абзацу 1 <b>напівжирний текст</b>
<var>змінна</var></p>
<p><em><s>Початок абзацу 2</s></em> продовження
тексту.</p>
</div>
</body>
</html>

```

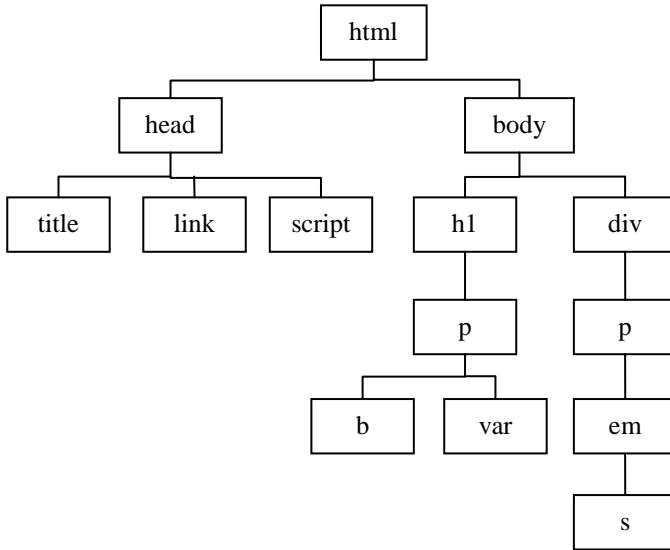


Рис. 1.1. Дерево html-документа

Відносини між множинними вкладеними елементами підрозділяються на батьківські, дочірні та сестринські:

- *предок* – елемент, який містить у собі інші елементи. На рис. 1.1 предком для всіх елементів є `<html>`. У той же час елемент `<body>` є предком для всіх тегів, що містяться в ньому: `<h1>`, `<div>`, `<p>`, `<b>` і т. ін.;

- *нащадок* – елемент, розташований всередині одного або більше елементів. Наприклад, `<body>` є нащадком `<html>`, а елемент `<p>` є нащадком одночасно для `<div>`, `<body>` і `<html>`;

- *батьківський елемент* – елемент, що пов’язаний з іншими елементами нижчого рівня і що знаходиться на дереві вище них. На рис. 1.1 `<html>` є батьківським тільки для `<head>` і `<body>`. Тег `<div>` є батьківським тільки для тегів `<p>`;

- *дочірній елемент* – елемент, безпосередньо підлеглий іншому елементу вищого рівня. На рис. 1.1 тільки елементи `<h1>` і `<div>` є дочірніми відносно `<body>`.

- *сестринський елемент* – елемент, що має загальний батьківський елемент з даним і є з ним одного рівня. На рис. 1.1 `<head>` і `<body>` – елементи одного рівня, так само, як і елементи `<h1>`, `<div>` є між собою сестринськими.

## 1.6. URL-адреси

Веб – це мережа інформаційних ресурсів, для доступу до яких використовується єдина система найменувань, що задається за допомогою *URL-адресів* (Uniform Resource Locator – уніфікований покажчик інформаційного ресурсу).

Мова HTML дозволяє задавати повні і відносні URL-адреси. *Повні адреси* зазвичай складаються з трьох частин:

- протоколу, який використовується для доступу до ресурсу;
- імені комп’ютера, на якому знаходиться ресурс;
- імені самого ресурсу, представленого як шлях до нього по каталогах файлової системи комп’ютера.

Загальна структура URL-адреси, що використовується для звернення до ресурсів HTTP-сервера, тобто до сервера, взаємодія з яким виконується на основі HTTP-протоколу:

```
"http:"//"host[":"port"]/"[abs_path["?"query]]
```

Тут частини URL-адреси, що можуть бути відсутніми, показано у квадратних дужках, а роздільники між частинами взяті в лапки. Інші частини мають такі значення:

- `host` – доменне ім’я сервера, наприклад, [www.kpi.kharkov.ua](http://www.kpi.kharkov.ua);
- `port` – ім’я порту сервера, через який виконується доступ до ресурсу, що запитується;

- `abs_path` – повний шлях до ресурсу, збереженого в певному каталозі сервера, наприклад «kn» в URL-адресі «<http://users.kpi.kharkov.ua/kn>»;

- `query` – параметри запиту до сервера.

*Відносна URL-адреса* не містить інформації про протокол. Як правило, вона задає шляхове ім'я до ресурсу, розташованого на тому ж комп'ютері, що і поточний документ. Відносні URL-адреси можуть містити шляхові імена, задані щодо певної базової адреси. Як базова адреса може використовуватися адреса самого поточного html-документа, або базова адреса, що задається за допомогою спеціального елемента `<base>`. Наприклад, адреса, позначена в атрибуті `src` теги `<img>`,

```

```

означає посилання на файл `logo.gif` із зображенням логотипу в каталозі `pictures`, розташованому в батьківському каталозі поточного каталогу html-документа.

## 1.7. Задання кольору в HTML

У HTML кольори визначаються цифрами в шістнадцятковому коді в адитивній колірній моделі RGB. Колір у моделі RGB представлено сумою яскравостей трьох базових кольорів – червоного (Red), зеленого (Green) і синього (Blue). Назва моделі утворена з перших букв англійських назв цих кольорів.

Яскравість (інтенсивність) кожного базового кольору може приймати 256 ( $2^8$ ) дискретних значень від 0 до 255. Змішування кольорів у різних пропорціях, варіюючи яскравість кожної складової, надає  $256 \times 256 \times 256 = 16\,777\,216$  кольорів.

У HTML для кожної колірної компоненти задається шістнадцяткове значення в межах від 00 до FF, що відповідає діапазону 0–255 у десятковому зчисленні. Потім ці значення об'єднуються в одне число, перед яким ставиться символ #. Наприклад, #800080 – темно-фіолетовий колір.

Щоб не запам'ятовувати сукупність цифр, яка задає колір, можна використовувати імена найбільш уживаних кольорів. Деякі з них подані в табл. 1.1.

Таблиця 1.1 – Стандартні кольори HTML

Найменування кольору	Шістнадцяткове значення RGB	Колір
black	#000000	чорний
silver	#C0C0C0	ясно-сірий
maroon	#800000	темно-червоний
red	#FF0000	червоний
green	#008000	зелений
lime	#00FF00	ясно-зелений
olive	#808000	оливковий
yellow	#FFFF00	жовтий
navy	#000080	темно-синій
blue	#0000FF	синій
purple	#800080	темно-фіолетовий
fuchsia	#FF00FF	ясно-фіолетовий
teal	#008080	синьо-зелений
aqua	#00FFFF	морської хвилі
gray	#808080	сірий
white	#FFFFFF	білий

## 1.8. Відтворення html-документів

Кожен html-документ являє собою послідовність символів з певного набору. Комп'ютерні системи ідентифікують кожен символ документа за його кодом. Наприклад, у наборі символів ASCII коди 65, 66 і 67 означають символи 'A', 'B' і 'C'.

Для забезпечення можливості мережної взаємодії застосувань з ресурсами Інтернету специфікація HTML вимагає, щоб кожен html-документ мав специфікований набір символів документа.

Набір символів документа складається з таких компонентів:

- набір абстрактних символів, що використовуються в документах, таких як латинські букви, літери кирилиці, китайських ієрогліфів і т. ін.;
- кодові позначення, тобто набір цілочисельних посилань на абстрактні символи.

Для Веб набору символів ASCII недостатньо. Тому в мові HTML використовується більш повний набір символів, званий *універсальним набором символів* (Universal Character Set – UCS), визначений у стандарті IS010646. Цей стандарт включає в себе набір тисяч символів, що використовуються у всьому світі. Він еквівалентний набору символів, що визначаються стандартом Unicode.

Для коректної інтерпретації html-документів клієнтськими браузерами недостатньо, щоб браузери були забезпечені різними наборами символів, тому що інформація html-документа при її збереженні у вигляді файла або при передачі по мережі зазвичай піддається перетворенню – кодується в послідовність байтів. Для перетворення послідовності байтів інформації в послідовність символів документа і навпаки використовується *кодування символів*.

Один із способів задання кодування документа – це його вказівка в розділі заголовка за допомогою теги `<meta>`:

```
<meta charset="UTF-8">
```

Може виявитися, що кодування символів не у змозі охопити всі символи з набору символів документа. Для таких кодувань або для таких конфігурацій обладнання та програмного забезпечення, які не дозволяють користувачам вводити певні символи, можуть бути використані *посилання на символи*.

Посилання на символи – це незалежний від кодування механізм введення будь-яких символів. Можуть використовуватися у двох видах:

- числові посилання на символи (десяткові або шістнадцяткові);
- символічні посилання, що являють собою комбінацію символів.

*Числові посилання* на символи вказують кодові позначення символів з набору символів документа. Для завдання числових посилань на символи можуть використовуватися такі синтаксичні конструкції:

- `&#D;` де D – десяткове число, що зазначає символ Unicode з десятковим номером D;
- `&#xH;` або `&#XH;` де H – шістнадцяткове число, яке вказує на символ Unicode з шістнадцятковим номером H. Шістнадцяткові числові посилання враховують регістр.

Приклади числових посилань на символи:

- `&#229;` (десятькове число) – це буква å, що використовується, наприклад у норвезькій мові;
- `&#xe5;` (шістнадцятькове число) – це той же символ;
- `&#1048;` (десятькове число) – велика кирилична літера "І".
- `&#x6c34;` (шістнадцятькове число) – це китайський ієрогліф 水 (вода).

*Символічні посилання* на символи використовують символічні позначення, що дозволяє не запам'ятовувати чисельні коди. Наприклад, комбінація `&aring;` позначає символ å, при цьому `&aring;` легше запам'ятати, ніж `&#229;`.

Підтримуються такі набори символічних посилань:

- символи набору ISO 8859-1 (Latin-1);
- символи, математичні символи і букви грецького алфавіту, які можуть виявлятися знаками шрифту Symbol;
- символи розмітки (наприклад, лапки) та інтернаціоналізації, які застосовуються для забезпечення коректності двонаправленого тексту, що перераховані у специфікації мови HTML.

У символічних посиланнях ураховується регістр. Наприклад, `&Aring;` вказує на символ у верхньому регістрі Å, `&aring;` позначає символ нижнього регістру å.

Список найбільш часто вживаних символічних посилань подано в табл. 1.2.

Таблиця 1.2 – Символічні посилання

Символ	Посилання
<	<code>&amp;lt;</code>
>	<code>&amp;gt;</code>
"	<code>&amp;quot;</code>
«	<code>&amp;laquo;</code>
»	<code>&amp;raquo;</code>
Нерозривний пробіл	<code>&amp;nbsp;</code>
©	<code>&amp;copy;</code>
™	<code>&amp;reg;</code>



## 1.9. Структурування тіла документа

Структурування тіла документа виконується всередині елемента `<body>` за допомогою заголовків, а також елементів розбивки.

Функції заголовків подібні звичайним стилям заголовків у текстових редакторах. У html-документах існує 6 рівнів заголовків, що задаються елементами `<h1> ... <h6>`, які відрізняються розміром шрифту при візуальному відтворенні у браузері. Причому `<h1>` – заголовок самого верхнього рівня, що відображається шрифтом найбільшого розміру, `<h6>` – найнижчий. Наприклад:

```
<H2 align="center"> Заголовок 2-го рівня з ви-  
рівнюванням по центру </H2>
```

Крім наведеного, значеннями атрибута `align`, що визначає вирівнювання тексту, можуть бути `left` (за замовчуванням), `center` і `right`.

Елементи `<div>` і `<span>` призначені для структурування html-документа шляхом групування його компонентів у розділи, кожен з яких має відмінне від інших форматування. Причому `<div>` належить до блокових елементів (наприклад, кілька абзаців тексту), а `<span>` – до рядкових, тобто всередині рядка. Наприклад:

```
<div align="значення"> Блок тексту </div>
```

Для кращого візуального сприйняття текст необхідно розбити на абзаци за допомогою тегу `<p>`. При цьому після абзацу буде вставлений порожній рядок:

```
<p align="значення">Текст абзацу </p>
```

Закриваючий тег для `<p>` не є обов'язковим.

Для примусового переходу на наступний рядок існує тег `<br>`. На відміну від `<p>` елемент `<br>` не вставляє порожній рядок і не має тегу, що його закриває, тобто є одиночним. Він не визначає структуру документа і ускладнює його автоматичний аналіз. Тому тег `<br>` бажано по можливості не використовувати.

Текст, позначений між контейнерними тегамі `<nobr>` і `</nobr>`, розташовується в одному рядку, без перенесення на інший.

У HTML передбачений контейнерний тег `<pre>`, в який можна включати попередньо відформатований текст. При цьому зазвичай використовується моноширинний шрифт, відключається автоматичний перенос слів на інший рядок, зберігаються додаткові символи пробілів. Тег може мати атрибут `width`, що задає ширину відтвореного фрагмента тексту.

Горизонтальні лінії зазвичай використовуються для візуального виділення області html-сторінки. Створюється цей ефект за допомогою одиночного тегу `<hr>`.

```
<hr align="значення" width="значення"  
size="значення" noshade color="значення">
```

Атрибути тегу `<hr>`:

- `width` – довжина лінії в пікселях або відсотках від ширини області виведення. В останньому випадку до числового значення додається символ «%»;
- `size` – ширина лінії в пікселях;
- `noshade` – логічний атрибут, який скасовує рельєфність лінії;
- `color` – колір лінії, заданий у RGB або за допомогою стандартного імені.

*Зауваження.*

Теги і атрибути, що задають форматування зовнішнього вигляду документа, є нерекомендованими, бо при їх використанні втрачається поділ структури документа і його зовнішнього вигляду. Замість цих тегів і атрибутів необхідно використовувати стилі.

## 1.10. Форматування тексту

Після розбиття тексту на абзаци і розділи, забезпечені заголовками, його необхідно відформатувати. При цьому подання html-документа на екрані кінцевого користувача залежить від багатьох факторів, зокрема від типу браузера і його налаштувань. Таке обмеження привело до розділення параметрів форматування на фізичні і логічні стилі форматування.

У фізичні стилі входять ті параметри форматування тексту, які не залежать від типу браузера. Логічні стилі форматування забезпечують

більшу гнучкість засобів структурування тексту. Вони використовуються для виділення в тексті різних фрагментів (цитат, програмних кодів та ін.).

Для форматування тексту *фізичними стилями* використовуються такі контейнерні теги:

- `<b>` – напівжирний;
- `<i>` – курсив;
- `<tt>` – моноширинний (телетайпний);
- `<u>` – підкреслений;
- `<strike>` або `<s>` – перекреслений;
- `<big>` – шрифт більшого розміру в порівнянні з установленим за замовчуванням;
- `<small>` – шрифт меншого розміру;
- `<sub>` – нижній індекс;
- `<sup>` – верхній індекс.

При завданні форматування теги можна комбінувати. Наприклад:

```
<P>Текст буде відображено <b><I>напівжирним курсивом</I></B>
```

Для форматування тексту *логічними стилями* використовуються такі контейнерні теги:

- `<cite>` – виділення цитат чи посилань на назву книг і статей, зазвичай курсив;
- `<code>` – висновок невеликої частини програмного коду, зазвичай моноширинний;
- `<em>` – виділення важливих фрагментів тексту, зазвичай курсив;
- `<kbd>` – текст, що вводиться користувачем з клавіатури, зазвичай моноширинний;
- `<samp>` – зразок інформації, виведеної програмою або сценарієм, зазвичай моноширинний;
- `<strong>` – виділення головних фрагментів тексту, зазвичай напівжирний;
- `<var>` – ім'я змінної, зазвичай курсив;
- `<dfn>` – визначення будь-якого терміну;

- `<acronym>`, `<abbr>` – виділення акронімів і аббревіатур. Для опису акроніма і аббревіатури використовується атрибут `title`.

```
<p>Це приклад аббревіатури: <abbr title="Word Wide Веб">WWW </abbr></P>
```

Для вставки довгих цитат у текст використовується тег `<blockquote>`. При цьому текст цитати зміщується вправо. Деякі браузерери позначають цитату символами «>», розташованими в крайньому лівому стовпчику екрана.

Ще одна можливість виділення цитат у рядку тексту – використання тегу `<q>`, причому браузерери повинні відображати текст, укладений в елемент `<q>`, у лапках.

Теги `<blockquote>` і `<q>` мають атрибут `cite`, значенням якого є URL-адреса сервера, звідки взято першоджерело цитати.

Елементи `<ins>` і `<del>` служать для позначки вставлених і видалених фрагментів у даній версії документа. При цьому видалені фрагменти будуть помічені закресленням, а вставлені – підкресленням. Вони мають такі атрибути:

- `cite` – URL-адреса документа-джерела або повідомлення, що містить інформацію про причини зміни документа;
- `datetime` – дата і час зміни у форматі TZD (позначення тимчасової зони): YYYY-MM-DDThh: mm: ss ± hh: mm;
- `title` – відображення коментаря з приводу внесених змін.

Наприклад:

```
<p>Штраф складає <del> 170 </del> <ins  
datetime="2017-01-01T12:25:13-02:00" title="За  
проханням ДАІ"> 540 </ins> грн.
```

## 1.11. Використання шрифтів

Використання шрифтів є одним з важливих засобів форматування html-документа. Для цього застосовуються елементи `<font>` і `<basefont>`.

Елемент `<font>` являє собою контейнер, який вимагає наявності як відкриваючого, так і закриваючого тегів. Він призначений для задання кольору і розміру шрифту тексту в контейнері і може вкладатися в будь-який інший контейнер.

Елемент `<basefont>` є одиночним і використовується для задання розміру базового шрифту. За замовчуванням, при відсутності `basefont`, використовується шрифт розміру 3. Елемент `<basefont>` розміщується в розділі заголовка документа.

Теги `<font>` і `<basefont>` мають такі атрибути:

- `size` – розмір шрифту. Значення задається в діапазоні від 1 до 7. Можна вказати також і відносні зміни, наприклад, `+2` – збільшення на 2 одиниці в порівнянні з використовуваним, `-1` – зменшення на 1;

- `color` – колір шрифту, заданий в RGB або за допомогою стандартного імені;

- `face` – список типів шрифтів у вигляді розділеного комами списку, з якого браузер вибирає відповідний шрифт. Наприклад:

```
<font face="Arial, Times" size=5  
color="blue"> Шрифт Arial </font>
```

## 1.12. Гіперпосилання в html-документах

Посилання бувають двох типів:

- на зовнішній ресурс;
- на розділи html-документа.

*Посилання на зовнішній ресурс* створюються таким чином:

```
<a href="url" hreflang="значення" type="значення"  
charset="значення">текст посилання</a>
```

Тег посилання на зовнішній ресурс має такі атрибути:

- `href` – обов'язковий атрибут, який визначає місце розташування веб-ресурсу, на який буде здійснено перехід;

- `hreflang` – задає базову мову ресурсу, зазначеного атрибутом `href` (ua, en і т.ін.);

- `type` – надає рекомендацію щодо типу вмісту ресурсу за цільовою адресою посилання. Зареєстровані типи вмісту визначаються стандартом MIME (Multipurpose Internet Mail Extensions – багатоцільові розширення електронної пошти в мережі Інтернет) і включає такі значення, як, наприклад, `"text/html"`, `"image/gif"`, `"video/mpeg"`, `"audio/basic"` та інші;

- `charset` – задає кодування символів ресурсу, зазначеного атрибутом `href`.

Посилання на розділи *html*-документа складаються з двох частин:

- самого посилання, що задається за допомогою конструкції  
`<a href="#ім'я_закладки">Текст посилання</a>;`
- закладки – цільового якоря, що створюється за допомогою  
`<a name="ім'я_закладки">Текст закладки</a>.`

Наприклад:

```
<a href="#section1">Вступ</a>
<a href="#section2">Основна частина</a>
<h1><a name="section1">Вступ</a></h1>
Текст вступу
<h1><a name="section2">Основна частина</a></h1>
Текст основної частини
```

### 1.13. Правила використання посилань

Посилання і якорі, що визначаються елементом `<a>`, не мають бути вкладеними, тобто тег `<a>` не повинен містити інших тегів `<a>`.

До імен якорів, що створюються за допомогою атрибутів `name` або `id`, застосовуються такі *правила*:

- *унікальність* – імена якорів мають бути унікальні в межах документа. В одному документі вони не можуть відрізнятися тільки регістром;
- *відповідність рядків* – порівняння між ідентифікаторами фрагментів та іменами якорів повинно проводитися на основі повного (з урахуванням регістру) збігу.

Атрибути `id` і `name` використовують той самий простір імен. Атрибут `id` може також використовуватися в сценаріях, при виборі з таблиці стилів і т.ін.

Елемент `<link>` визначає зв'язок між веб-ресурсами. На відміну від елемента `<a>` може бути присутнім тільки в контейнері `<head>` документа і може бути включений в нього необмежену кількість разів.

Елемент `<link>` не є контейнерним. Він містить інформацію про відносини, які можуть відображатися клієнтськими браузеромі різними способами (наприклад, у вигляді панелі із списком посилань).

Атрибути, характерні для `<link>`, збігаються з атрибутами елемента `<a>`. Однак у елемента `<link>` є ще один атрибут – `media`, який використовується, якщо елемент `<link>` є посиланням на *зовнішню*, тобто задану в будь-якому іншому документі, таблицю стилів документа.

```
<link href="mystyle.css" rel="stylesheet"
      type="text/css" media="projection">
```

- `REL` – атрибут описує посилання з поточного документа в якій, що задається атрибутом `HREF`;
- `MEDIA` – задає цільовий пристрій, на який орієнтована дана інформація про стиль. Значенням за замовчуванням є `screen`. Крім того, може набувати значень:
  - `projection` – для проекторів;
  - `print` – для принтерів;
  - `tv` – для виведення на екран телевізора;
  - `brailee` – для тактильних пристроїв з алфавітом Брайля;
  - `aural` – для синтезаторів мовлення;
  - `all` – для всіх типів пристроїв та ін.

## 1.14. Зображення в html-документах

### 1.14.1. Графічні формати Інтернету

Використання графіки робить html-сторінки візуально привабливіше. Зображення допомагають краще передати суть і зміст html-документа. Також за допомогою html-тегів можна робити карти-зображення з активними областями.

Існує велика кількість растрових графічних форматів, але в Інтернеті використовуються ті, що мають стиснення зображень. Це пов'язано з необхідністю забезпечення невеликого розміру html-сторінки. Таким чином, стандартними форматами Інтернету є:

- формат `GIF` (`Graphics Interchange Format`) – призначений для зберігання так званої бізнес-графіки (логотипів, діаграм тощо). Є ідеальним для стиснення зображень, в яких є області з суцільним кольо-

ром. GIF-файли дозволяють установити один з кольорів прозорим, завдяки чому фон html-сторінки буде видно через частину зображення. Також GIF-файли можуть містити просту анімацію. Формат використовує індексовану кольорову палітру, що складається лише з 256 кольорів. Через це зображення можуть виглядати плямистими і мати нереалістичні кольори, як плакати;

- формат JPEG (Joint Photographic Experts Group) – призначений для зберігання фотографій. JPEG-файли можуть містити мільйони різних кольорів та стискають зображення краще за GIF, але текст і великі площі з суцільним кольором можуть покритися плямами;

- формат PNG (Portable Network Graphics) – у свій час був створений як альтернатива GIF-формату, тому існує ще одне його неофіційне розшифрування – Picture is Not GIF. Містить кращі властивості GIF- і JPEG-форматів. Містить мільйони кольорів і дає можливість зробити один з кольорів прозорим, при цьому стискає зображення в менший розмір, ніж GIF-файл;

- формат APNG (Animated Portable Network Graphics) – формат зображення, що базується на форматі PNG. Дозволяє зберігати анімацію, а також підтримує прозорість;

- формат ICO (Windows icon) – використовується як іконка на сайтах в Інтернеті. Саме картинка такого формату відображається поруч з адресою сайту або на закладці в браузері. Один ICO-файл містить один або кілька значків, розмір і колір кожного з яких задається окремо. Розмір значка може бути будь-яким, але найбільш вживані квадратні значки зі сторонами 16, 32 і 48 пікселів.

Серед векторних форматів в Інтернеті широко використовується SVG (Scalable Vector Graphics). SVG-рисунок складається з набору геометричних фігур, описаних у форматі XML: лінія, еліпс, багатокутник і т. ін. У ньому підтримується як статична, так і анімована графіка. Набір функцій містить різні перетворення, альфа-маски, ефекти фільтрів, можливість використання шаблонів. Завдяки перевагам векторної графіки SVG-зображення можуть змінюватися в розмірі без зниження якості.

#### 1.14.2. Вставка зображень у html-документ

Зображення додаються на html-сторінки за допомогою одинарного тегу <img>.

```

```



Для тегу <img> доступні такі атрибути:

- `src` – задає URL-адресу зображення і є обов'язковим;
- `alt` – позначає альтернативний текст для зображення. Виводиться на місці появи зображення до його завантаження або при відключеній графіці, а також виводиться спливаючою підказкою при наведенні курсору миші на зображення;
- `align` – позиціонування в документі. Можливі такі значення:
  - `bottom` – вікно об'єкта повинно бути вертикально вирівняно відносно поточної базової лінії (за замовчуванням);
  - `middle` – центр об'єкта повинен бути вирівняний вертикально відносно поточної базової лінії;
  - `top` – верх об'єкта повинен бути вирівняний вертикально відносно верху поточного текстового рядка;
  - `left` – об'єкт прикріплюється до поточного лівого краю; поданий нижче текст обтікає зображення справа;
  - `right` – об'єкт прикріплюється до поточного правого краю; поданий нижче текст обтікає зображення зліва;
- `height` – задає висоту зображення в пікселях або відсотках від первісного зображення;
- `width` – аналогічно задає ширину зображення в пікселях або відсотках;
- `hspace` – горизонтальні відступи від зображення в пікселях;
- `vspace` – аналогічно вертикальні відступи від зображення в пікселях.

### 1.14.3. Графічні гіперпосилання

Графічні гіперпосилання створюються за допомогою комбінації тегів <a> і <img>:

```
<a href="url документа, куди необхідно перейти">  
  </a>
```

Атрибут `border` керує товщиною рамки, якою обводиться зображення при перетворенні його в гіперпосилання. У більшості випадків ця рамка псує дизайн сторінки, тому значення цього достатньо установити значення цього атрибуту рівним нулю.

#### 1.14.4. *Карти-зображення*

Карти-зображення дозволяють створити декілька гіперпосилань, використовуючи різні області одного зображення. В Інтернеті існують у двох варіантах – серверному і клієнтському.

У випадку використання *серверного варіанта* браузер посилає запит на сервер для отримання адреси обраного посилання і чекає відповіді з необхідною інформацією. Такий підхід вимагає додаткового часу на очікування і окремі файли-обробники для кожної карти-зображення.

Серверні карти-зображення використовуються у випадках, коли неможливо забезпечити їх обробку клієнтським браузером. Створюються серверні карти за допомогою комбінації тегів `<a>` і `<img>`. В `<img>` необхідно встановити логічний атрибут `ismap`, що і задає серверну карту. Файл-обробник результатів задається за допомогою атрибута `href` тегу `<a>`:

```
<a href="http://www.server.com/path">
</a>
```

У *клієнтському варіанті* карта знаходиться в тому ж html-документі, що і посилання на зображення. Для її створення треба установити в `<img>` атрибут `usemap`. Його значення повинно містити ім'я карти-зображення з початковим символом #:

```
.
```

Після цього створюється карта за допомогою контейнерного тегу `<map>` і задаються її активні області за допомогою одиночних тегів `<area>`:

```
<map name="ім'я_карти">
<area href="url" shape="параметр"
coords="значення" alt="альтернативний текст"
nohref>
</map>
```

Атрибут `name` тегу `<map>` задає унікальне ім'я карти, що записується латинськими символами та є чутливим до регістру.

Тег `<area>` безпосередньо визначає активну область зображення та має такі атрибути:

- href – задає адрес документа, на який здійснюється посилання;
- alt – дозволяє задати альтернативний текст;
- shape – керує формою активної області. Можливі значення:
  - default – область, що займає все зображення;
  - rect – прямокутна активна область, значення за замовчуванням;
  - circle – круга активна область;
  - poly – активна область у вигляді багатокутника;
- coords – дозволяє визначити відносні координати вершин активної області. Відлік ведеться в пікселях або відсотках за довжиною і шириною зображення від його лівого верхнього кута, що прийнятий за точку (0,0). Правила задання значень атрибута залежно від форми активної області наведено в табл. 1.3;
- nohref – необов'язковий логічний атрибут, що задає неактивні ділянки карти.

Таблиця 1.3 – Значення атрибута coords

Значення атрибута shape	Синтаксис запису coords	Значення параметрів coords
rect	coords="x1, y1, x2, y2"	x1, y1 – координати лівого верхнього кута, x2, y2 – координати правого нижнього кута
circle	coords="x, y, R"	x, y – координати центра, R – радіус
poly	coords="x1, y1, x2, y2, ..., xN, yN"	x1, y1, ..., xN, yN – координати вершин багатокутника

Нижче наведено приклад html-документа з картою-зображенням та його зовнішній вигляд у браузері (рис. 1.2).

```
<!DOCTYPE html>
<html>
```

```

<head>
<meta charset=utf-8">
<title>Карта-зображення</title>
</head>
<body>
<h1>Колірні моделі</h1>
<p></p>
<p>
<map name="map1">
<area shape="poly" alt="RGB"
      coords="123,31, 111,6, 116,0, 194,0,
      211,31" href="rgb.html">
<area shape="poly" alt="CMYK"
      coords="213,31, 201,6, 209,0, 285,1,
      302,31" href="cmyk.html">
<area shape="poly" alt="HSB"
      coords="303,31, 295,6, 302,0, 377,0,
      395,31" href="hsb.html">
</map>
</p>
<p>Для опису кольору пікселів використовують
універсальні моделі, де застосовується точний
опис кольору в стандартизованих цифрових і
математичних виразах.</p>
</body>
</html>

```

## 1.15. Списки в html-документах

Списки – один з найефективніших способів подання інформації в html-документі. Вони легко сприймаються і можуть являти послідовність процедур, параметри для вибору рішень, перелік посилань, меню і т. ін. Списки є фундаментальним способом організації і подання інформації.

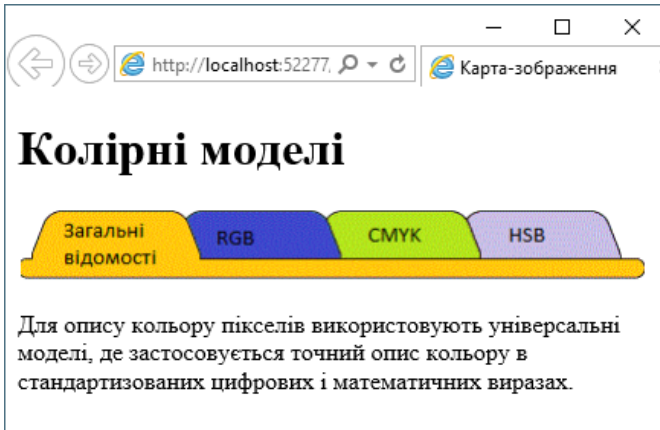


Рис. 1.2. Документ з картою-зображенням

У HTML є три типи списків:

- упорядкований (нумерований) список (елемент `ol`);
- неупорядкований (маркірований) список (елемент `ul`);
- список визначень (елемент `dl`).

Різні типи списків можна комбінувати.

*Упорядкований список* відкривається контейнерним тегом `<ol>`, а кожен його пункт починається стандартним тегом `<li>`. Програма перегляду, зустрівши тег `<ol>` впорядкованого списку, послідовно нумерує пункти списку: 1, 2, 3 і т. ін.

Для форматування заголовка списку може використовуватися тег `<lh>`.

Для закриття списку використовується тег `</ol>`.

Початковий і кінцевий теги забезпечують переведення рядка до і після списку, відокремлюючи таким чином список від решти тексту. Це виключає необхідність використовувати теги абзацу `<P>`.

Допускається вкладеність списків. Загальний вигляд упорядкованого списку наведено нижче.

```
<ol type="значення" start="значення">
<li> Пункт 1</li>
<li> Пункт 2</li>
...
<li> Пункт n</li>
```

</ol>

Атрибути тегу <ol>:

- `type` – встановлює стиль списку, який визначається одним з таких значень:
  - `1` – маркер у вигляді арабських цифр (значення за замовчуванням);
  - `I` – маркер у вигляді великих римських цифр;
  - `i` – маркер у вигляді малих римських цифр;
  - `A` – маркер у вигляді великих латинських букв;
  - `a` – маркер у вигляді малих латинських букв;
- `start` – початкове значення маркера списку (завжди число).  
Значення за замовчуванням – 1.

У *невпорядкованих списках* замість цифр використовують різні символи, що називаються маркерами списку.

Як і в упорядкованих списках, тут також забезпечується автоматичне переведення рядка до і після списку і допускається створення вкладених списків.

Список розміщується всередині контейнерного тегу <ul>. Клієнтські браузери автоматично створюють відступ для вкладених списків і вибирають тип маркерів (тип маркера залежить від типу браузера).

Загальний вигляд невпорядкованого списку такий:

```
<ul type="значення">  
<li> Пункт 1</li>  
<li> Пункт 2</li>  
...  
<li> Пункт n</li>  
</ul>
```

Атрибут `type` встановлює стиль списку, який визначається одним з таких значень:

- `disc` – маркер у вигляді жирної крапки (значення за замовчуванням);
- `square` – маркер у вигляді квадрата;
- `circle` – маркер у вигляді кола.

*Списки визначень* подають текст у формі словникової статті, що складається з терміну і подальшого текстового абзацу, що пояснює

його зміст. Такі списки також зручні для складання каталогів, опису функцій підрозділів організації, розмітки діалогів.

Елемент списку визначень `<dl>` є контейнером і відокремлює список від решти тексту html-документа порожніми рядками. Всередині контейнера кожний термін задається тегом `<dt>`, а абзац з його описом – тегом `<dd>`.

Текст після тегу `<dt>` має поміститися в один рядок. Якщо ця вимога порушується і рядок виходить за межі вікна браузера, виконуються переведення рядка, але вже без відступу.

Залежно від типу браузера текст, що стоїть за тегом `<dd>`, виводиться окремим абзацом з відступом вниз на один або два рядки щодо терміну, який розшифровується.

Нижче наведено приклад списку визначень та його зовнішній вигляд у браузері (рис. 1.3).

```
<dl>
<dt> А</dt>
<dd> Перша літера алфавітів</dd>
<dt> А КАПЕЛА (італ. а cappella)</dt>
<dd> Хоровий спів багатоголосний без інструмен-
тального супроводу. </dd>
</dl>
```

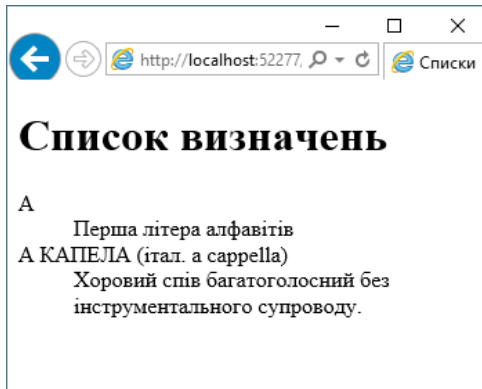


Рис.1.3. Списки визначень

## 1.16. Таблиці в html-документах

У текстових документах таблиці є одним з найпопулярніших засобів подання інформації. Мова HTML надає засоби створення таблиць, що дозволяють авторам html-документів упорядковувати дані – текст, форматований текст, зображення, посилання, форми, поля форм, інші таблиці, розміщуючи їх у рядки та стовпці комірок.

Відповідно до загальної стратегії W3C, що складається в розширенні кола користувачів мережі веб, з кожною таблицею у HTML може бути пов'язаний заголовок, який визначається елементом `<caption>`. Тема надає короткий опис таблиці. Це допомагає людям зі слабким зором оцінити вміст таблиці перед її переглядом.

З цією ж метою тепер можна вказати і більш довгий опис вмісту таблиці, скориставшись атрибутом `summary`, який задає альтернативний спосіб відтворення вмісту таблиць для користувачів невізуальних браузерів (наприклад, що базуються на азбуці Бройля або мовному відтворенні).

Рядки таблиці в html-документі можуть групуватися в розділи верхніх і нижніх заголовків, а також в тіло таблиці. Для цього в HTML включені елементи `<thead>`, `<tfoot>` і `<tbody>`. Рядки, що груповані, додатково структурують html-документ і можуть відтворюватися клієнтськими браузерами різними способами.

Можна також групувати стовпці таблиці для надання клієнтським браузерам додаткової інформації про структуру документа, що використовується браузерами при відтворенні.

У HTML можна задавати властивості стовпців на початку таблиці, скориставшись елементами `<colgroup>` і `<col>`. Це дозволяє клієнтським браузерам відтворювати таблицю послідовно, у міру надходження порцій інформації, а не чекати зчитування всіх даних таблиці перед її відтворенням.

Комірки таблиці можуть містити "заголовок" (елемент `<th>`) або "дані" (елемент `<td>`), а також займати кілька рядків або стовпців.

При використанні таблиць не слід нехтувати засобами альтернативного, невізуального відтворення і розглядати таблиці тільки як засіб компонування html-документа.

Нижче наведено приклад простої таблиці розміром 2×2 та її зовнішній вигляд у браузері (рис. 1.4).



```

<h1> Найпростіша таблиця </h1>
<table border=1>
<caption> Заголовок таблиці</caption>
<tr>
<td> Комірка 1-1 </td>
<td> Комірка 1-2 </td>
</tr>
<tr>
<td> Комірка 2-1 </td>
<td> Комірка 2-2 </td>
</tr>
</table>

```

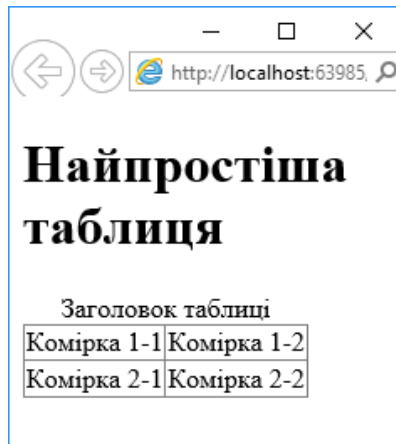


Рис.1.4. Проста таблиця

Елемент `<table>` являє собою контейнерний тег, в якому розміщується вміст таблиці.

Таблиця будується по рядках: для позначення рядка використовується контейнерний тег `<tr>`, для позначення заголовків стовпців – контейнерний тег `<th>`, для даних у комірках – контейнерний тег `<td>`.

Заголовки виділяються напівжирним шрифтом і вирівнюються по центру комірки.

Атрибути елемента `<table>`:

- `align` – задає вирівнювання таблиці щодо тексту документа. Може приймати одне з трьох значень:
  - `left` – таблиця знаходиться в лівій частині документа (значення за замовчуванням);
  - `center` – таблиця знаходиться в центрі документа;
  - `right` – таблиця знаходиться у правій частині документа;
- `width` – визначає необхідну ширину всієї таблиці в пікселях або відсотках від ширини екрана. Якщо ширина не вказана, то вона визначається клієнтським браузером;
- `summary` – стисла інформація про призначення та структуру таблиці для браузерів, що виконують невізуальні відтворення документа.

Елемент `<caption>` призначений для виведення назви таблиці. Елемент не є обов'язковим. Якщо він присутній, то повинен розташовуватися відразу після початкового тегу `<table>`. При цьому елемент `<table>` може включати тільки один елемент `<caption>`.

Елемент `<caption>` має тільки один характерний атрибут – `align`, який вказує на положення заголовка стосовно таблиці. Може набувати значень:

- `top` – заголовок знаходиться зверху таблиці і вирівняний по її центру (значення за замовчуванням);
- `bottom` – знизу таблиці, вирівняний по її центру;
- `left` – зверху таблиці, вирівняний по лівому краю;
- `right` – зверху таблиці, вирівняний по правому краю.

Елемент `<tr>` не має характерних атрибутів і служить контейнером для комірок, що входять у рядок таблиці.

Чарунки, що входять у рядок, задаються елементами `<th>` (заголовки) і `<td>` (дані). Вони можуть мати такі атрибути:

- `width` – рекомендована ширина комірки в пікселях або відсотках від ширини таблиці. Якщо ширина не вказана, то вона визначається клієнтським браузером;
- `height` – рекомендована висота комірки в пікселях;
- `nowrap` – логічний атрибут, який використовується для відключення в клієнтському браузері автоматичного розбиття тексту всередині комірки;

- `rowspan` – визначає число рядків, що займає поточна комірка. За замовчуванням значення дорівнює 1; 0 означає, що комірка займає всі рядки: від поточного до останнього рядка таблиці (іноді працює некоректно);

- `colspan` – аналогічно визначає число стовпців, що займає поточна комірка. За замовчуванням значення дорівнює 1; 0 означає, що комірка займає всі стовпці: від поточного до останнього стовпця таблиці.

Таким чином, атрибути `rowspan` і `colspan` використовуються для фізичного об'єднання комірок таблиці. Приклад таблиці з об'єднаними комірками наведено нижче, а її вигляд у браузері – на рис. 1.5.

```
<h1> Таблиця з об'єднаними комірками </h1>
<table border=1>
<caption> Приклад таблиці з об'єднанням
</caption>
<tr>
<td> Комірка 1 </td>
<td> Комірка 2 </td>
<td> Комірка 3 </td>
<tr>
<td> Комірка 4 </td>
<td rowspan=2> Комірка 5 </ td>
<td> Комірка 6 </td>
<tr>
<td> Комірка 7 </td>
<td> Комірка 9 </td>
</table>
```

Рядки таблиці можна групувати у верхні заголовки, нижні заголовки і один або кілька розділів тіл таблиці, що задаються відповідно контейнерними елементами `<thead>`, `<tfoot>`, `<tbody>`. Такий поділ таблиці дозволяє задавати для різних частин таблиці різне форматування.

При наявності `<thead>`, `<tfoot>`, `<tbody>` кожен з них визначає групу рядків, кожна з яких повинна містити принаймні один рядок, який визначається за допомогою елемента `<tr>`.

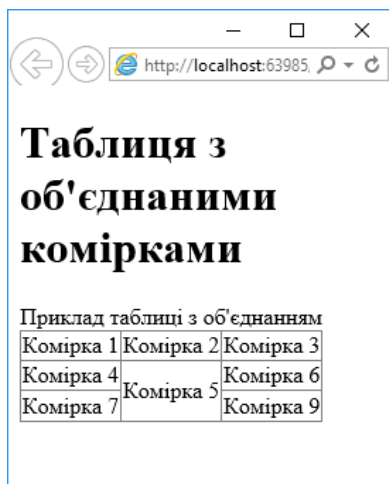


Рис.1.5. Таблиця з об'єднаними комірками

Об'єднання стовпців таблиці у групи дозволяє створювати структурні підрозділи всередині таблиці. Ця можливість здійснюється за допомогою елементів `<colgroup>` і `<col>`.

Елемент `<colgroup>` створює групу стовпців. Він призначений для задання ширини і стилю однієї або декількох колонок таблиці.

Атрибути `<colgroup>`:

- `span` – число стовпців у групі;
- `width` – ширина відображення стовпця групи в пікселях або відсотках. Також значенням атрибута може бути величина `0*`, тобто ширина стовпця за вмістом.

Приклад таблиці з групуванням стовпців наведено нижче, а її зображення в браузері – на рис. 1.6.

```
<h1> Таблиця з групуванням стовпців </h1>
<table border=1>
<colgroup span=3 width=50>
<colgroup span=2 width="0*">
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
```

```
<td>5</td>
</tr>
</table>
```

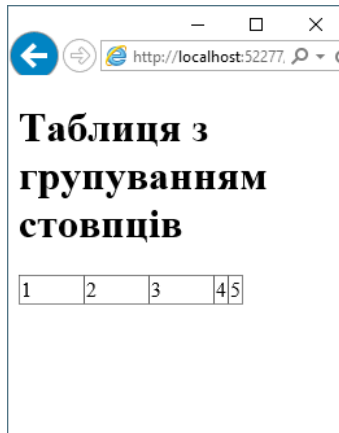


Рис. 1.6. Використання елемента `<colgroup>`

Елемент `<col>` дозволяє задавати різні значення атрибутів для різних наборів стовпців таблиці. Вони можуть міститися як всередині групи стовпців, що задаються елементом `<colgroup>`, так і поза групою.

Набір атрибутів збігається з `<colgroup>`, але на відміну від `<colgroup>`, елементи `<col>` не групують стовпці таблиці у структурні розділи, а тільки визначають атрибути. Елемент `<col>` є одиночним.

#### 1.16.1. Форматування таблиць

Для подання таблиць у візуальних браузерях мова HTML має багато можливостей, які можуть бути розділені на *чотири категорії*:

- засоби завдання зовнішніх рамок і внутрішніх розділових ліній;
- засоби управління кольором і фоном;
- засоби горизонтального і вертикального вирівнювання вмісту комірок;
- засоби завдання полів комірок.

Для задання зовнішніх рамок і внутрішніх розділових ліній використовуються такі атрибути тегу `<table>`:

- `frame` – вказує, які сторони рамки навколо таблиці повинні бути видимими. Може набувати таких значень:
  - `void` – видимих сторін немає (значення за замовчуванням);
  - `above` – видимою є тільки верхня сторона;
  - `below` – тільки нижня сторона;
  - `hsides` – тільки верхня і нижня сторони;
  - `vsides` – тільки ліва і права сторони;
  - `lhs` – тільки ліва сторона;
  - `rhs` – тільки права сторона;
  - `box` – всі чотири сторони є видимими;
  - `border` – всі чотири сторони, тобто аналогічно `box`;
- `rules` – вказує, які розділові лінії будуть відображатися між комірками. Відтворення цих ліній залежить від клієнтського браузера. Може набувати таких значень:
  - `none` – немає ліній (значення за замовчуванням);
  - `all` – лінії відображаються між рядками і стовпцями;
  - `rows` – лінії відображаються тільки між рядками;
  - `cols` – лінії відображаються між стовпцями;
  - `groups` – лінії відображаються тільки між групами рядків, що задаються за допомогою `thead`, `tfoot`, `tbody`, і групами стовпців, що задаються за допомогою `colgroup` і `col`;
- `border` – ширина в пікселях рамки навколо таблиці. Якщо просто вказаний атрибут без значення, то буде відображено рамку шириною 1px.

Для задання кольору і фону використовуються такі атрибути елементів `<table>` і `<td>`:

- `background` – задає фоновий рисунок.

```
<table background="url">
```

Фоновий рисунок завжди відображається в натуральну величину з масштабом 100 %. Якщо рисунок за розміром менше ширини або висоти, то картинка повторюється по горизонталі вправо і вниз, шикуючись, як мозаїка. Тому на місці стику фонових картинок можуть ви-

никнути видимі перепади. Обраний фоновий рисунок повинен забезпечити достатню контрастність між ним і вмістом клітинки. Як фон допускається використовувати анімовані зображення в форматі GIF, але вони відволікають увагу користувачів;

- `bgcolor` – колір фону елемента, ним можуть бути `<table>`, `<thead>`, `<tbody>`, `<tfoot>`, `<tr>`, `<td>`. Значення за замовчуванням залежить від браузера і його версії, зазвичай використовується білий колір фону;

- `bordercolor` – колір рамки навколо елемента, ним можуть бути `<table>`, `<thead>`, `<tbody>`, `<tfoot>`, `<tr>`, `<td>`. Рамку буде відображено лише в тому випадку, якщо встановлено параметр `border` з ненульовим значенням у тезі `<table>`.

Для задання вирівнювання тексту в комітках використовуються атрибути тегів `<td>`, `<tr>`, `<col>`, `<colgroup>`, `<tbody>`, `<thead>`, `<tfoot>`:

- `align` – горизонтальне вирівнювання тексту в комірці. Може набувати значень:

- `left` – по лівому краю (значення за замовчуванням);
- `center` – по центру;
- `right` – по правому краю;
- `justify` – по ширині;

- `valign` – вертикальне положення тексту в комірці:

- `top` – дані зсуваються вгору;
- `middle` – по центру (значення за замовчуванням);
- `bottom` – дані зсуваються вниз;
- `baseline` – дані розташовуються вздовж базової лінії.

### 1.16.2. Успадкування параметрів вирівнювання

Вирівнювання вмісту комірки таблиці може задаватися для кожної комірки або успадковуватися від елементів верхнього рівня, таких як рядок, стовпець або сама таблиця.

При цьому виникає питання пріоритету цих атрибутів, оскільки вони можуть суперечити один одному.

Пріоритети атрибута `align` встановлюються згідно з такими правилами (пріоритетність у порядку від вищого до нижчого):

1. Атрибут вирівнювання, встановлений в елементах форматування даних усередині комірки (наприклад, в елементі абзацу `<p>`).

2. Атрибут вирівнювання, встановлений в елементі комірки (`<th>` і `<td>`).

3. Атрибут вирівнювання, встановлений в елементі групування стовпців (елементи `<col>` і `<colgroup>`). Якщо комірка є частиною об'єднання декількох стовпців, властивість вирівнювання успадковується від визначення комірки на початку об'єднання.

4. Атрибут вирівнювання, встановлений в елементі рядка `<tr>` або елементах групування рядків (`<thead>`, `<tfoot>` і `<tbody>`). Якщо комірка є частиною об'єднання кількох рядків, властивість вирівнювання успадковується від визначення комірки на початку об'єднання.

5. Атрибут вирівнювання, встановлений в елементі таблиці `<table>`.

6. Значення вирівнювання, що застосовується за замовчуванням.

Пріоритет атрибута `valign` встановлюється за точно такими ж правилами, що і для атрибута `align`, але в них пункти 3 і 4 мають бути переставлені місцями. Крім того, при відтворенні комірок горизонтальне вирівнювання визначається спочатку для стовпців, а потім для рядків, а вертикальне вирівнювання – спочатку для рядків, а потім для стовпців.

### 1.16.3. Поля комірки

Для кожної комірки можна задати поля між границею комірки і її вмістом. Для цього використовуються атрибути `cellspacing` і `cellpadding` елемента `<table>`.

Атрибут `cellspacing` визначає розмір проміжку (в пікселях або відсотках від області екрана), який браузер установлює між границями таблиці і вмістом комірок. Він також визначає проміжок між комірками таблиці.

Атрибут `cellpadding` визначає проміжок (у пікселях або відсотках від області екрана) між границею комірки і її вмістом.

## 1.17. Вбудовані фрейми

Контейнерний тег `<iframe>` створює вбудований або плаваючий фрейм, що знаходиться всередині звичайного html-документа. Він до-



зволяє завантажувати в область заданих розмірів будь-які незалежні документи.

Атрибути тегу `<iframe>`:

- `align` – задає положення фрейму відносно html-документа та спосіб його обтікання текстом;
- `allowtransparency` – логічний атрибут, що встановлює прозорий фон для фрейму, через який видно фон головної html-сторінки;
- `frameborder` – задає правила відображення границі навколо фрейму. Можливі значення – `1` і `0` або `yes` і `no`;
- `height` – висота фрейму в пікселях або відсотках від батьківського елемента. Якщо батьківський елемент відсутній, то ним замінюється вікно браузера;
- `width` – аналогічно ширина фрейму;
- `hspace` – горизонтальний відступ в пікселях від фрейму до контенту, що його оточує;
- `vspace` – аналогічно вертикальний відступ від фрейму до контенту;
- `marginheight` – відступи зверху і знизу в пікселях від вміста до межі фрейму;
- `marginwidth` – аналогічно відступи справа і зліва;
- `name` – ім'я фрейму;
- `sandbox` – дозволяє задати обмеження на контент, що завантажувється у фреймі, наприклад блокувати форми і скрипти. Це дозволяє підвищити безпеку поточного документа. Можливі значення:
  - `allow-same-origin` – дозволяє завантажувати вміст фрейму з того ж джерела, що і батьківський документ, при цьому блокуються вікна, що впливають;
  - `allow-top-navigation` – дозволяє відкривати посилання фрейму в батьківському документі;
  - `allow-forms` – дозволяють вмісту фрейму відправляти форми;
  - `allow-scripts` – дозволяють запуск і виконання скриптів, при цьому створення вікон, що впливають, заборонено;

- можна вказувати декілька значень у будь-якому порядку через пробіл. Якщо вказано пусте значення, то задаються всі можливі обмеження;
- `scrolling` – задає спосіб відображення смуг прокручування у фреймі. Можливі значення:
  - `auto` – значення за замовчуванням, відображення смуг прокручування при необхідності, коли вміст фрейму перевищує його видиму частину;
  - `no` – заборонено відображення смуг прокручування;
  - `yes` – протилежне значення – відображення смуг прокручування, незалежно від обсягу інформації;
- `seamless` – логічний атрибут, який встановлює, що вміст фрейму повинен відображатися як частина батьківського документа;
- `src` – URL-адреса файла, вміст якого буде завантажуватися у фрейм;
- `srcdoc` – дозволяє зберігати вміст фрейму безпосередньо в атрибуті. При цьому можуть використовуватися html-теги.

Приклад вбудованого фрейму наведено нижче, а його зображення в браузері – на рис. 1.7.

```

<h1>Інформація про колірні моделі</h1>
<p><a href="rgb.htm" target="color">RGB</a> |
<a href="cmyk.htm" target="color">CMYK</a> |
<a href="hsb.htm" target="color">HSB</a>
</p>
<p><iframe src="model.htm" name="color"
width="100%" height="200">
</iframe>
</p>

```

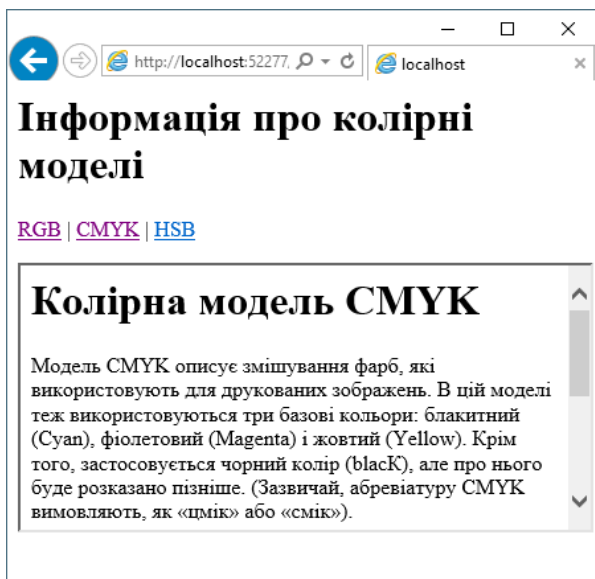


Рис. 1.7. Використання вбудованого фрейму

## 1.18. Форми в html-документах

### 1.18.1. Створення форми

*Форми* призначені для отримання від користувача будь-яких відомостей, що відправляються на веб-сервер. На сервері запускається програма, тобто *серверний сценарій*, що обробляє отримані дані і генерує відповідь клієнту.

Html-документ може містити декілька форм, але одночасно на сервер може бути відправлена тільки одна форма. Тому дані форм мають бути незалежними.

Для створення форм призначений контейнерний тег `<form>`, всередині якого знаходяться елементи керування форми. Дозволяється всередині контейнера `<form>` розміщати інші теги, при цьому сама форма ніяк не відображається на html-сторінці, будуть видні тільки її елементи і результати вкладених тегів.

```
<form action="URL" name="ім'я"  
method="значення">
```

Елементи керування форми

</form>

Атрибути тегу <form>:

- `accept-charset` – встановлює кодування, в якому сервер може приймати і обробляти дані. Значенням за замовчуванням є кодування сторінки;
- `action` – URL-адреса програми або документа, що обробляє дані форми. Якщо атрибут не задано, передбачається, що обробкою даних буде займатися сама сторінка;
- `autocomplete` – вмикає автоматичне заповнення елементів форми. Можливі значення `on` і `off`;
- `enctype` – спосіб кодування даних форми;
- `method` – метод відправлення даних за протоколом HTTP. Найбільш поширені два методи – `get` і `post`;
- `name` – ім'я форми;
- `novalidate` – логічний атрибут, вмикає вбудовану перевірку даних форми на коректність введення;
- `target` – ім'я вікна або фрейму, куди обробник буде завантажувати результат обробки даних.

### 1.18.2. Елементи керування форми

Одиночний тег <input> є одним з різнобічних елементів форми і дозволяє створювати різні елементи інтерфейсу і забезпечити взаємодію з користувачем. Головним чином <input> призначений для створення текстових полів, різних кнопок, перемикачів і прапорців.

Основний атрибут тегу <input>, що визначає вид елемента – `type`. Для кожного елемента існує свій список атрибутів, які визначають його вид і характеристики. Крім того, в HTML5 додано ще більше десятка нових елементів.

Значення атрибута `type`:

- `text` – створює поле для введення рядка символів; значення за замовчуванням;
- `password` – аналогічно `text`, але створює поле для введення пароля, тому символи, що вводяться, не відображаються;
- `checkbox` – прапорець – дозволяє обрати більше одного варіанта із запропонованих;

- `radio` – перемикач – використовується, коли слід вибрати один варіант з кількох запропонованих;
- `button` – кнопка, що буде реагувати на натискання на неї користувачем, якщо задано відповідний обробник подій;
- `submit` – кнопка для відправлення даних форми на сервер;
- `image` – графічна кнопка, при натисканні на рисунок дані форми відправляються на сервер;
- `reset` – кнопка для повернення даних форми в початкові значення;
- `file` – поле для вибору імені файлу, який пересилається на сервер;
- `hidden` – приховане поле, воно ніяк не відображається на html-сторінці, але може використовуватися для передачі параметрів на сервер.

У HTML5 додані нові значення:

- `color` – елемент для вибору кольору;
- `date` – поле для вибору календарної дати;
- `datetime` – завдання дати і часу в обраному форматі;
- `datetime-local` – завдання місцевої дати і часу;
- `email` – поле для введення адреси електронної пошти;
- `number` – поле для введення чисел;
- `range` – повзунок для вибору чисел у зазначеному діапазоні;
- `search` – поле для пошуку;
- `tel` – поле для введення телефонних номерів;
- `time` – поле для введення часу;
- `url` – поле для введення веб-адреси;
- `month` – вибір місяця;
- `week` – вибір тижня.

Інші атрибути тегу `<input>` залежать від вибраного типу елемента керування. Нижче наведено деякі з них. Так, наприклад, атрибут `value` дозволяє задати початкове значення елемента керування. Атрибут є обов'язковим тільки для перемикачів, для кнопок задає їх підпис.

Атрибут `name` дозволяє задати ім'я елемента керування. Особливістю однієї групи перемикачів є те, що всі елементи групи повинні мати *однакове ім'я*.

Логічний атрибут `checked` установлюється для перемикачів або прапорців та показує, що цей елемент вибрано.

Атрибут `size` задає ширину елемента керування в пікселях. Для текстових полів та полів введення пароля ширина задається кількістю символів.

Тег `<button>` створює на html-сторінці кнопки і за своєю дією нагадує результат, одержаний за допомогою тегу `<input>` з параметром `type` зі значеннями `button`, `reset` або `submit`. На відміну від `<input>`, тег `<button>` пропонує розширені можливості щодо створення кнопок. Наприклад, на кнопці можна розмішувати будь-які елементи HTML, у тому числі зображення (рис. 1.8).

```
<p align = center>  
<button id = button1> Кнопка з текстом  
</button>  
<button id = button2>  
  <img src = "pic1.jpg" alt = "Посилання"  
  vertical-align = middle>  
Кнопка з рисунком </button> </p>
```

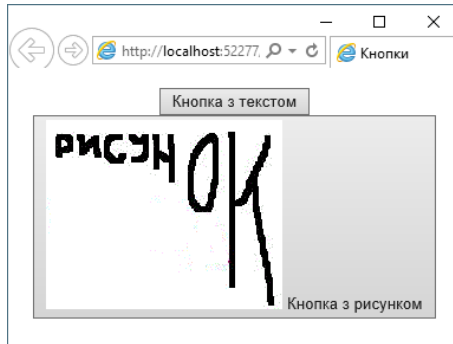


Рис. 1.8. Приклад кнопок

Атрибути тегу `<button>`:

- `name` – визначає ім'я кнопки;
- `value` – визначає початкове значення кнопки;
- `type` – визначає тип кнопки. Можливі значення `button`, `reset` або `submit`.

Тег `<select>` дозволяє створити список, що розкривається, а також список з одиночним або множинним вибором. Вид списку залежить від атрибута `size` тегу `<select>`, який встановлює висоту списку.

Атрибути тегу `<select>`:

- `name` – визначає ім'я списку;
- `size` – кількість елементів списку, що одночасно відображаються;
- `multiple` – логічний атрибут, при установленні якого зі списку можна здійснити множинний вибір.

Кожен пункт списку створюється за допомогою тегу `<option>`. Ширина списку визначається найбільш широким текстом, зазначеним у тезі `<option>`.

```
<select>
<option> пункт 1 </option>
<option> пункт 2 </option>
</select>
```

Атрибути тегу `<option>`:

- `value` – початкове значення елемента. Якщо не встановлено, то початкове значення дорівнює вмісту елемента `<option>`;
- `selected` – логічний атрибут, його встановлення показує, що відповідний пункт попередньо обраний.

Контейнерний тег `<textarea>` дозволяє створити багаторядкову область введення. На відміну від `<input>` у текстовому полі припустимо робити переноси рядків, причому вони будуть зберігатися при відправленні даних на сервер.

Атрибути тегу `<textarea>`:

- `name` – ім'я текстового поля;
- `rows` – висота поля, тобто кількість рядків, що відображаються без смуг прокручування;
- `cols` – ширина поля, що задається числом символів моноширинного шрифту.

Тег `<label>` встановлює зв'язок між певною міткою (текстом) і елементом форми. За допомогою `<label>` також можна встановлювати гарячі клавіші на клавіатурі і переходити на активний елемент.

Контейнерний елемент `<fieldset>` призначений для групування елементів форми, не має характерних атрибутів і використовується спільно з контейнерним тегом `<legend>`. Тег `<legend>` має атрибут `align`, який визначає положення легенди щодо набору елементів керування у групі. Можливі значення: `top`, `bottom`, `left`, `right`.

Приклад документа, що містить різні елементи керування форми, наведено нижче, а його зображення у браузері – на рис. 1.9.

```
<h1>Форма реєстрації</h1>
<p>Ім'я <input type=text name="f_name"
value="ім'я"><br>
Прізвище <input type="text" name="l_name"><br>
Е-mail <input type="email" name="email"></p>
<p>Оберіть місце проживання
<br><select size="2">
<option>столиця</option>
<option>обласний центр</option>
<option>районний центр</option>
<option>інше</option>
</select></p>
<p><fieldset>
<legend align="top">Оберіть курс:</legend>
<input type="radio" name="kurs" value="html"
checked> HTML <br>
<input type="radio" name="kurs" value="css">
CSS<br>
<input type="radio" name="kurs" value="PHP">
PHP
</fieldset></p>
<p>Додаткова інформація <br>
<textarea name="comment" cols=32 rows=5>
</textarea></p>
<p>
<input name="confirm" type="checkbox" checked>
Підтвердити отримання
<br> <input type="submit" value="Відправити">
<input type="reset" value="Відмінити"></p>
```



Форма реєстрації

Ім'я

Прізвище

E-mail

Оберіть місце проживання

столиця  
обласний центр

Оберіть курс:

HTML

CSS

PHP

Додаткова інформація

Підтвердити отримання

Рис. 1.9. Приклад форми

### 1.19. Універсальні атрибути html-елементів

Універсальні атрибути застосовуються практично до всіх тегів, тому виділені в окрему групу. До них належать:

- `accesskey` – дозволяє отримати доступ до елемента за допомогою заданого поєднання клавіш;
- `class` – визначає ім'я стильового класу, яке дозволяє зв'язати тег з оформленням;
- `dir` – задає напрямок і відображення тексту (зліва направо `ltr` або справа наліво `rtl`);
- `id` – вказує ім'я стильового ідентифікатора, також використовується для програмного доступу до елемента;

- `lang` – браузер використовує значення параметра для правильного відображення деяких національних символів, наприклад лапок;
- `style` – використовується для задання стиля елемента за допомогою правил CSS;
- `tabindex` – встановлює порядок отримання фокуса при переміщенні між елементами за допомогою клавіші Tab;
- `title` – описує вміст елемента у вигляді підказки;
- `contextmenu` – встановлює контекстне меню для елемента;
- `hidden` – приховує вміст елемента від перегляду;
- `contenteditable` – повідомляє, що вміст елемента доступний для редагування користувачем, тобто можна видаляти текст і вводити новий. Також працюють стандартні команди на кшталт скасування, вставки тексту з буфера та ін.;
- `spellcheck` – вказує браузеру, перевіряти чи ні правопис і граматику в тексті. Хоча атрибут можна встановлювати практично для всіх елементів, результат буде помітний лише для полів форм, що створюються за допомогою тегів `<input>` і `<textarea>`, а також елементів зі встановленим атрибутом `contenteditable`.

## 1.20. Події в html-документі

*Події* – це стандартні ситуації, що виникають при настанні деяких умов, наприклад при кліку мишею на елементі або при натисканні клавіші на клавіатурі. Повідомлення про подію передається об'єкту, в якому вона виникла. Якщо в даному об'єкті є обробник відповідної події, то виконуються дії, задані в ньому. Як обробники подій в html-документі виступають клієнтські скрипти.

Події і їх обробники додаються як атрибути до відповідних html-елементів, наприклад

```
<body onload="init()">.
```

У мові HTML визначені такі групи подій:

- *події документа*, що відбуваються в елементі `<body>` або фреймах. До них належать `onload` – завантаження документа у вікно та `onunload` – видалення документа з вікна;

- *події фокуса*, що відбуваються при отриманні елементом фокуса за допомогою миші або послідовності переходу (onfocus) та при втраті його (onblur);

- *події миші*, які можуть використовуватися з більшістю html-елементів. До них належать onclick, ondblclick, onmousedown, onmouseup, onmousemove, onmouseout і onmouseover;

- *події клавіатури*, що також можуть відбуватися з більшістю html-елементів. До них належать onkeypress, onkeydown і onkeyup;

- *події форм та їх елементів*. До них належать:

- onreset – відбувається при очищенні форми і використовується тільки з елементом <form>;
- onsubmit – відбувається при відправленні форми і використовується тільки з елементом <form>;
- onselect – відбувається при виділенні користувачем деякого тексту в поле і може використовуватися з елементами <input> і <textarea>;
- onchange – відбувається при втраті елементом керування фокуса, якщо в той час, коли він мав фокус, вміст елемента керування було змінено. Може використовуватися з елементами <input>, <select>, <textarea>.

## 1.21. Включення скриптів у html-документи

*Клієнтський скрипт* – це програма, яка може супроводжувати html-документ або безпосередньо бути вбудованою в нього. Підтримка скриптів у HTML не залежить від мови, якою написаний даний скрипт.

В html-документах існують два типи скриптів:

- виконувані один раз при завантаженні документа в браузер;
- виконувані кожен раз, коли відбувається певна подія.

Скрипти в html-документ включаються за допомогою елемента <script>:

```
<script type="тип" src="URL">  
Текст програми на мові сценаріїв  
</script>
```

Атрибути тегу <script>:

- `type` – задає MIME тип для мови, якою написаний скрипт. Можливі значення: `text/javascript`, `text/vbscript`;

- `src` – URL-адреса, за якою знаходиться файл з текстом програми-скрипта.

Елемент `<script>` може використовуватися в тому самому документі в елементах `<head>` і/або `<body>` необмежену кількість разів.

Якщо атрибут `src` не заданий, то браузер сприймає текст, розташований між тегами `<script>` і `</script>`, як текст програми-скрипта. Якщо ж атрибут `src` заданий, то браузер проігнорує текст, що міститься між цими тегами, а як джерело програми-скрипта використовує файл, розташований за адресою, вказаною в атрибуті `src`.

## 1.22. Особливості HTML 5

Як зазначалось раніше, стандарт мови HTML під керівництвом консорціуму W3C пройшов довгий еволюційний шлях. У 1999 р. вийшла версія HTML 4.01, після цього W3C зайнявся розробкою стандартів XML і XHTML, які повинні були цілком замінити HTML. Проте на HTML базувалась значна частина веб-контенту. Для переходу до веб-застосувань нового типу та подолання існуючих недоліків HTML у 2004 р. за ініціативою крупних виробників браузерів і сторін, зацікавлених у розробленні нових технологій, була створена спільнота WHATWG (Web Hypertext Application Technology Working Group – Робоча група з технологій застосування гіпертексту у веб). Завдяки їх зусиллям з'явилась специфікація HTML 5. У 2006 р. до роботи над HTML знов підключився консорціум W3C, в результаті чого в 2008 р. з'явився перший робочий проект рекомендацій з HTML 5.

28 жовтня 2014 р. консорціум W3C оголосив про надання набору специфікацій HTML 5 статусу рекомендованого стандарту, хоча на той час HTML 5 вже давно став стандартом де-факто і активно використовувався у веб-застосуваннях.

### Переваги HTML 5:

- мова здебільшого сумісна з існуючими стандартами для HTML 4. Нові засоби розмітки працюють згідно з існуючими, нові API (Application Programming Interface – інтерфейс програмування застосу-

вань) ґрунтуються на існуючих JavaScript / DOM, які розробники використовують протягом багатьох років;

- HTML 5 додає нові потужні засоби в HTML, які були раніше доступні в Веб тільки за допомогою технології плагінів, таких як Flash, або за допомогою складного коду JavaScript чи спеціальних прийомів;

- краще підходить для написання динамічних застосувань, ніж попередні версії HTML;

- має чіткий алгоритм синтаксичного аналізу, тому всі браузери, які реалізують HTML 5, будуть створювати однакове дерево DOM з однією розміткою, що є величезним виграшем для сумісності.

Мова HTML 5 містить багато нових властивостей, що робить HTML значно більш потужним і зручним для створення веб-додатків. Серед них можна виділити:

- нові семантичні елементи – для HTML дуже важливою є семантика, розробник повинен використовувати для роботи відповідні елементи. Засоби HTML 4 для визначення таблиць, списків, заголовків тощо не мають спеціальних елементів для цього. Зазвичай їх визначають за допомогою тегів `<div id="xxx"> </div>`, які пошукові машини не можуть визначити як змістовні. HTML 5 містить нові семантичні елементи, такі як `<nav>`, `<header>`, `<footer>` і `<article>`;

- нові властивості форм – HTML 4 має засоби для створення веб-форм, але загальні властивості форм є не дуже зручними і вимагають спеціальних зусиль для реалізації. HTML 5 надає стандартизований, простий спосіб реалізації таких властивостей, як вибір дати, повзунки і клієнтська перевірка;

- власна підтримка відео та аудіо – протягом багатьох років відео і аудіо в Веб втілювалися за допомогою Flash-технологій. Технологія Flash стала популярною на початку XXI століття, оскільки відкриті стандарти не змогли надати для різних браузерів сумісний механізм реалізації таких речей, тому різні браузери реалізували різні конкуруючі способи виконання одних функцій (наприклад, `<object>` і `<embed>`), роблячи тим самим весь процес дійсно складним. Flash надавав високоякісний, легкий спосіб реалізації роботи відео в різних браузерах. HTML 5 містить елементи `<video>` і `<audio>` для простої реалізації власних відео- і аудіоплеєрів за допомогою лише відкри-

тих стандартів, і також містить API, що дозволяє легко реалізувати індивідуальні елементи управління плеєром;

- API рисування на полотні – елемент `<canvas>` і відповідний API дозволяють визначити на сторінці область для рисування і використовувати команди JavaScript для рисування ліній, фігур і тексту, імпорту та маніпуляцій із зображеннями і відео, експорту в різні формати зображень і багатьох інших речей;

- веб-сокети – цей API, опис якого доступний за адресою <http://www.w3.org/TR/websockets/>, дозволяє відкривати постійне з'єднання між сервером і клієнтом на певному порті, і посилати дані в обох напрямках, поки порт не буде закрито. Це істотно покращує ефективність веб-застосувань, оскільки дані можуть безперервно передаватися між клієнтом та сервером без постійного перезавантаження сторінки і без постійного опитування сервера, чи немає доступних оновлень;

- автономні веб-застосування – HTML 5 надає можливості, що дозволяють веб-застосуванням виконуватися в автономному режимі. Кеші застосувань дозволяють зберігати копії всіх ресурсів, необхідних для локального виконання веб-застосувань, а бази даних WebSQL – локальну копію даних веб-програми. Спільно вони спроможні продовжувати використовувати застосування за відсутності з'єднання з мережею. Згодом, коли мережа стане доступною, зміни синхронізуються з основною версією на сервері;

- веб-сховище – cookies надають у певній мірі локальне сховище даних, але їх використання є достатньо обмеженим. Веб-сховище HTML 5 дозволяє зберігати значно більше даних і виконувати з ними значно більше дій;

- web workers – загальною проблемою веб-застосувань є зменшення продуктивності, коли потрібно обробити багато даних у зв'язку з тим, що все відбувається в одному процесі (в поточний момент може виконуватися тільки одна послідовність обробки). Web workers створюють фонові процеси для виконання значного обсягу обчислень, дозволяючи основному процесу продовжувати виконання інших завдань;

- геолокація – специфікація геолокації, що знаходиться за адресою <http://dev.w3.org/geo/api/spec-source.html>, визначає API, який дозволяє веб-застосуванню легко отримувати доступ до даних у будь-якому місці розташування, яке стало доступним, наприклад, за допомогою засобів GPS-пристроїв. Це дозволяє надавати застосуванню різних

корисних властивостей, що пов'язані з місцем розташування, наприклад, виділити контент, який найбільше підходить для місця розташування.

## 2. ПОСТАНОВКА ЗАВДАННЯ НА ЛАБОРАТОРНІ РОБОТИ

### 2.1. Завдання на лабораторну роботу «Базові можливості мови HTML»

1. Ознайомитись з основними можливостями html-редактора, що входить до складу обраного середовища розробки (Microsoft Visual Studio або Eclipse).

При використанні середовища розробки Microsoft Visual Studio потрібно запустити його, після чого з'явиться вікно Start Page, загальний вигляд якого наведено на рис. 2.1.

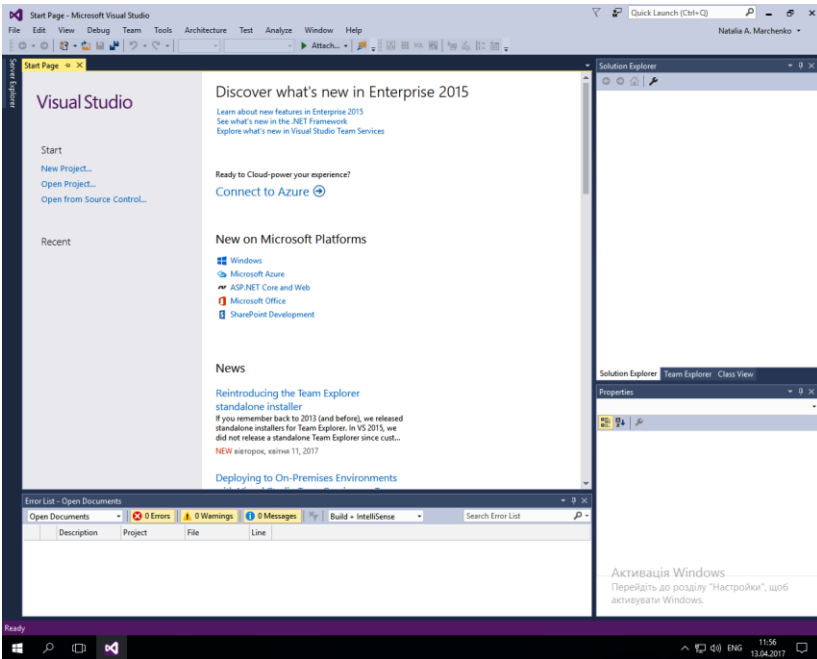


Рис. 2.1. Зовнішній вигляд стартового вікна Microsoft Visual Studio

Створити новий html-документ за допомогою меню File, підпункту New File або комбінації клавіш Ctrl+N. У запропонованому системному діалоговому вікні обрати пункт HTML Page. Після натискання на кнопку Open з'явиться вікно, що містить заготовку під html-документ. Зовнішній вигляд вікна наведено на рис. 2.2.

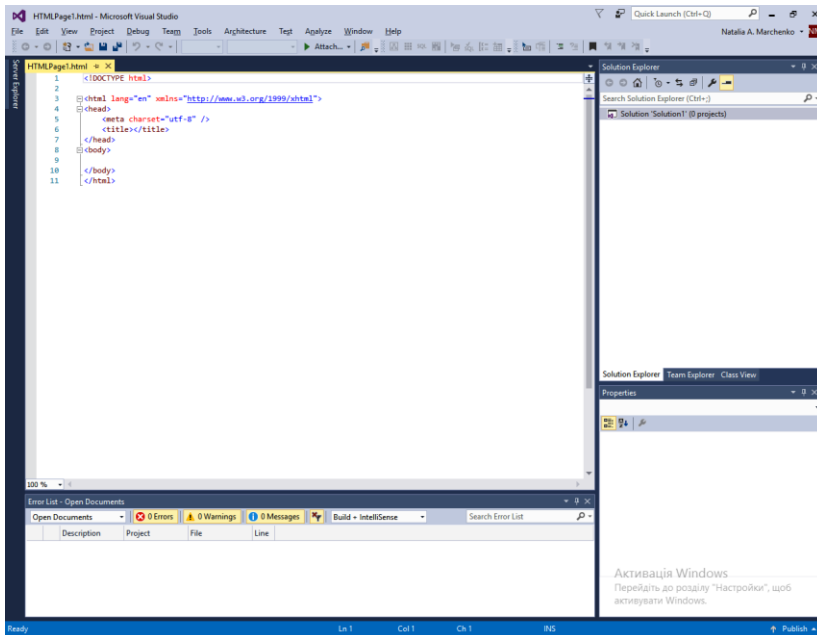


Рис. 2.2. Зовнішній вигляд вікна html-редактора Microsoft Visual Studio

Вікно html-редактора має декілька областей. Зліва знаходиться вікно для редагування вихідного тексту html-документа.

У правій частині розміщено вікна Solution Explorer та Properties. Вікно Properties дозволяє задавати або змінювати значення атрибутів тегів. Ці зміни буде відображено у вікні редагування. Якщо вікно Properties після запуску відсутнє, його можна відкрити за допомогою пункту меню View, Properties Window або натиснувши клавішу F4.

Якщо у вікні редагування поставити символ «<», що означає початок тегу, то з'явиться вікно підказки, з якого можна обрати відповідний тег. Теги можна також вводити, не використовуючи підказку, тоб-



то з клавіатури. Якщо введений тег парний, то редактор автоматично його закрити.

При введенні атрибутів тегів також передбачено вікно підказки. Воно з'являється, якщо в межах відкриваючого тегу натиснути на клавішу Пробіл.

При використанні середовища розробки Eclipse також потрібно запустити його, після чого з'явиться вікно Start Page, загальний вигляд якого наведено на рис 2.3.

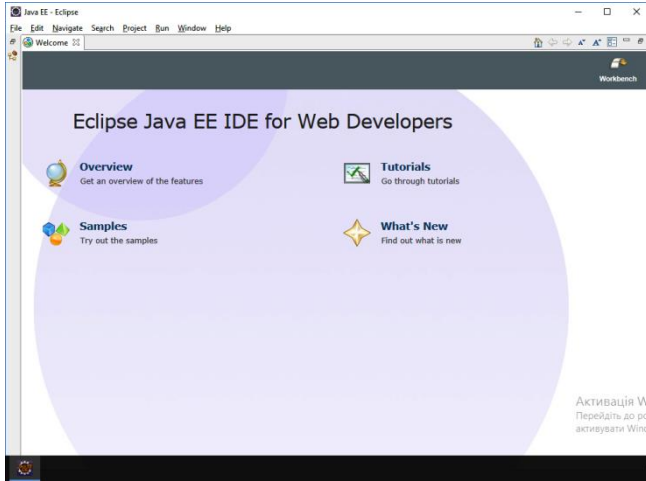


Рис. 2.3. Зовнішній вигляд стартового вікна Eclipse

Для створення нового html-документа в середовищі Eclipse має бути створений проект. У цій лабораторній роботі достатньо створити проект статичного веб-застосування. Для цього необхідно вибрати меню File, підпункти New, Static Веб Project. Або ж можна викликати вікно створення нового елемента за допомогою меню File, підпункту New, Other або комбінації клавіш Ctrl+N, і у цьому вікні обрати пункт Веб, Static Веб Project, як це показано на рис. 2.4.

У другій вкладці вказати назву проекту та, при необхідності, задати місце розташування файлів проекту.

Для створення нового html-документа треба в лівому вікні Project Explorer виділити проект, в якому він має бути створений. Після цього за допомогою меню File, підпункту New, HTML File або комбінації клавіш Ctrl+N та вибору пункту Веб, HTML File викликати діалогове

вікно створення нового html-документа. В цьому вікні треба вказати назву файла, що створюється, та місце його розташування. На наступній вкладці потрібно обрати шаблон, за яким буде створено новий файл. Приклади вкладок вікна створення нового html-документа наведені на рис. 2.5.

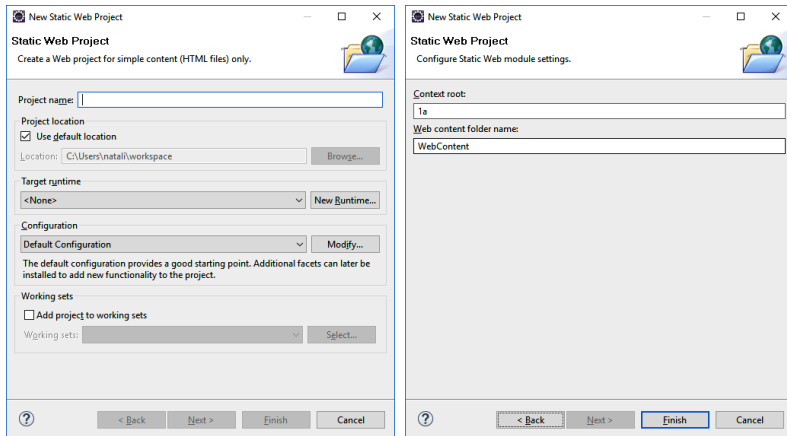


Рис. 2.4. Вкладки діалогового вікна вибору елемента, що створюється

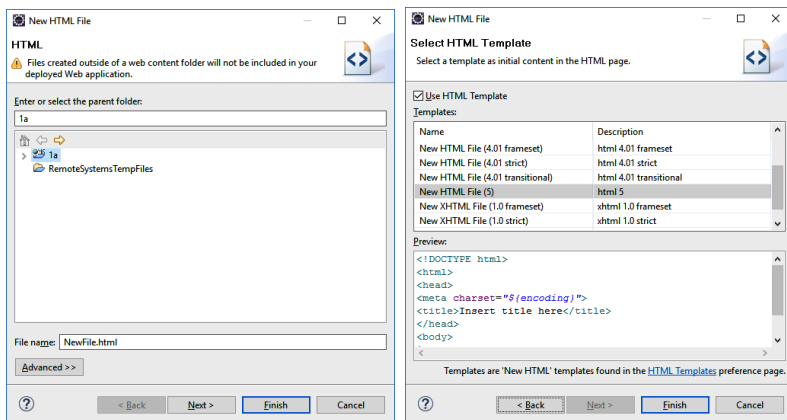


Рис. 2.5. Вкладки діалогового вікна створення html-документа

Після натиснення на кнопку Finish з'явиться вікно, що містить заготовку під html-документ. Зовнішній вигляд вікна наведено на рис.2.6.

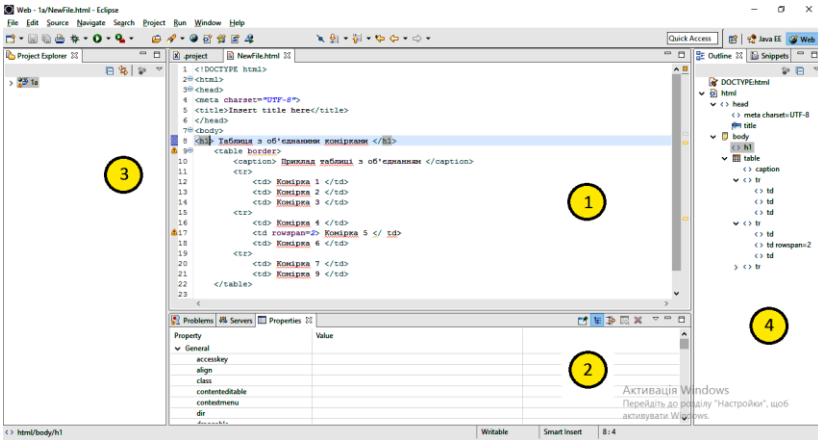


Рис. 2.6. Зовнішній вигляд вікна html-редактора Eclipse

Вікно html-редактора має декілька областей, які пронумеровано.

Номером 1 позначено область редагування вихідного тексту. Аналогічно, як і в Microsoft Visual Studio, якщо у вікні редагування поставити символ «<<», що означає початок тегу, то з'явиться вікно підказки, з якого можна обрати відповідний тег. Якщо введений тег парний, то редактор автоматично його закрити при наборі символів «<</>».

При введенні атрибутів тегів також передбачено вікно підказки. Воно з'являється, якщо в межах відкриваючого тегу натиснути на комбінацію клавіш Ctrl+Пробіл.

Внизу розміщено вікна Problems, Servers та Properties. У вікні Problems відображаються помилки та зауваження щодо вихідного тексту html-документа. Вікно Servers використовується для розміщення створеного проекту на веб-сервері. Вікно Properties, яке позначено цифрою 2, дозволяє задавати або змінювати значення атрибутів тегів. Ці зміни буде відображено у вікні редагування. Якщо вікно Properties після запуску відсутнє, його можна відкрити за допомогою пункту меню Window, Show view, Properties.

Цифрою 3 позначено вікно Project Explorer, в якому показано структуру поточного проекту. Праворуч розміщено вікно Outline, в якому показано структуру html-документа, що редагується. Воно позначено цифрою 4.

Для перегляду створеного html-документа також можна використовувати можливості Eclipse. Для цього треба натиснути праву кнопку

миші у вікні редагування та в контекстному меню обрати пункт Open With Web Browser. Після цього з'явиться нова вкладка, як це наведено на рис. 2.7.

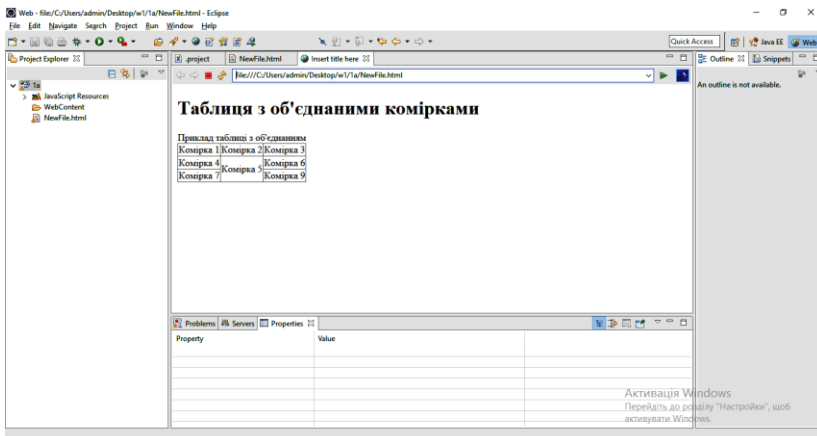


Рис. 2.7. Перегляд створеного html-документа в Eclipse

2. Вивчити структуру різних html-документів та дескриптори, що застосовуються в них для форматування тексту на прикладі даних файлів.

Для цього треба у вікні редагування набрати html-коди файлів та зберегти їх під відповідними іменами у власну папку за допомогою пункту меню File, Save As.

Файл first.html.

```
<html>
<head>
<title>Це перша найпростіша сторінка</title>
</head>
<body> Це проста html-сторінка </body>
</html>
```

Файл text.html.

```
<html>
<head> <title> Робота з текстом </title> </head>
<body bgcolor="white">
<font color="blue">
```

```

<h1 align= "center">HTML і створення веб-
сторінок</h1>
</font>
<font color="green">
<h2 align= "center">Що таке HTML?</h2>
</font>
<p>
<font color="maroon">
Мова HTML не є мовою програмування в тому сенсі, в
якому звичайно уявляють собі мови програмування (C,
C++, Java та ін.
</font>
<p align= "justify">За допомогою HTML неможливо
написати повноцінне програмне застосування, що пра-
цює у будь-якій операційній системі. HTML призначе-
на для створення документів, що розміщуються на
веб-сервері, доступ до яких виконується по мережі
Інтернет.
<p>Демонстрація зміни розмірів шрифтів.
<p align="center">Звичайний шрифт (його величину
можна вказати в тезі basefont.)
<p align="center">
<font size=+1>Указано розмір size=+1</font>.
<p align="center">
<font size=+2>Указано розмір size=+2</font>.
<p align="center">
<font size=+3>Указано розмір size=+3</font>.
<p align="center">
<font size=+4>Указано розмір size=+4</font>.
</body>
</html>

```

Для перегляду в браузері створених документів безпосередньо з html-редактора можна використати контекстне меню, що визивається при натисканні правою кнопкою миші, та обрати в ньому пункт View in Browser або натиснути комбінацію клавіш Ctrl+Shift+W.

3. Створити новий html-документ з ім'ям index.html. Він повинен містити заголовок з назвою лабораторної роботи, карту зображення для забезпечення навігації і вбудований фрейм завантаження подання ін-

формації з інших html-документів. У нижній частині html-документа під горизонтальною лінією повинна знаходитися інформація про автора і авторське право. Для введення інформації про авторське право використати посилання на відповідний стандартний символ.

4. Створити новий html-документ з ім'ям lists.html. Він повинен містити заголовок, нумеровані і нелінійні списки, а також їх комбінацію. Додати в цей документ список визначень з даного курсу. Додати в посилання на розділи списку визначень.

5. Створити html-документ з ім'ям tables.html. Він повинен містити заголовок та таблицю з об'єднаними комірками (за варіантами). Створена таблиця, наприклад, може мати такий вигляд (табл.2.1).

Таблиця 2.1 – Об'єднання комірок

Об'єднання 1-1, 1-2, 1-3			Об'єднання 1-4, 2-4, 3-4, 4-4
2-1	2-2	2-3	
3-1	Об'єднання 3-2, 3-3, 4-2, 4-3		
4-1			

5. Створити новий html-документ з ім'ям forms.html. Він повинен містити заголовок, а також форму для замовлення будь-якої продукції. При створенні форми використати різні елементи керування.

6. Забезпечити завантаження у вбудований фрейм документа index.html всіх інших розроблених документів.

## 2.2. Завдання на лабораторну роботу «Створення сайту за допомогою HTML»

Створити веб-сайт обсягом 4-5 html-сторінок, присвячений заданій предметній області, що обирається згідно з варіантом. Варіанти предметних областей наведено в табл. 2.2.

Передбачити розміщення навігаційних панелей та дублювання елементів навігації. У нижній частині кожної сторінки повинна знаходитися інформація про авторів та авторські права.

Сайт повинен містити такі структурні елементи html-коду:

- заголовки;
- списки;
- таблиці;

- рисунки;
- гіперпосилання на сторінки і гіперпосилання на позицію всередині сторінки.

Усі сторінки сайту оформити в єдиному стилі.

Таблиця 2.2 – Варіанти предметних областей

Номер варіанта	Предметна область
1	туристичне агентство
2	магазин компакт-дисків
3	магазин комп'ютерів
4	агентство з працевлаштування
5	енциклопедія музики
6	енциклопедія кіно
7	електронна бібліотека
8	сайт закладу освіти
9	сайт кафедри
10	сайт факультету
11	магазин автомобілів
12	туристичний довідник
13	сайт служби електронної пошти
14	сайт новин
15	сайт агентства нерухомості

### Контрольні запитання

1. Який принцип побудови мережі Веб?
2. Що означає термін «Мова розмітки гіпертексту» та які вона має особливості?
3. Які існують правила створення html-документів?
4. Які основні положення тегової моделі документа?
5. Які існують різновиди тегів?
6. Які компоненти має загальна структура html-документа?
7. Що являє собою URL-адреса та яке вона має призначення?

8. Які існують типи URL-адрес?
9. Які способи завдання кольору елементів існують у мові HTML?
10. З яких компонентів складаються символи документа?
11. Яке призначення мають посилання на символи html-документа?
12. Які існують типи посилань на символи?
13. Які існують засоби для структурування інформації в html-документі?
14. Які типи форматування тексту стилями існують у мові HTML? Наведіть приклади тегів, що належать кожному зі стилів.
15. Які можливості використання шрифтів передбачено в HTML?
16. Які типи гіперпосилань передбачено в мові HTML та якими дескрипторами вони описуються?
17. Які існують правила при використанні гіперпосилань?
18. Які типи списків передбачено в мові HTML та якими дескрипторами вони описуються?
19. Яке призначення таблиць у мові HTML та які основні дескриптори використовуються для задання таблиць?
20. Які можливості форматування таблиць передбачено в HTML?
21. Яким чином спаковуюються параметри вирівнювання даних у таблиці?
22. Які графічні формати використовуються в Інтернет? Наведіть сфери використання кожного з них.
23. Яким чином у html-документ включається графіка та створюються графічні гіперпосилання?
24. Що являють собою карти-зображення? За допомогою яких дескрипторів вони створюються?
25. Що означає термін «вбудований фрейм»? За допомогою яких тегів він створюється?
26. Проаналізуйте призначення форм у html-документах. Якими дескрипторами вони описуються?
27. Які існують елементи керування форм? Наведіть приклади тегів і їх атрибутів, що використовуються для створення елементів керування форми.
28. Які події можуть виникати в html-документі та його елементах? Наведіть приклади подій, що характерні для html-форм та їх елементів.



29. Що означає термін «універсальний атрибут»? Наведіть декілька прикладів з ним.

30. Яким чином підключаються клієнтські підпрограми в html-документ?

31. Який існує пріоритет при одночасному підключенні клієнтської підпрограми з файла та наявності вбудованого в html-документ JavaScript-коду?

32. Проаналізуйте переваги мови HTML 5 у порівнянні з попередньою версією HTML 4.01.

33. Перелічте особливості HTML 5.

34. Наведіть приклади нових елементів, доданих у HTML 5.

35. Яким чином реалізовано підтримку аудіо і відео в HTML 5?

36. Яким чином реалізовано можливості рисування в html-документах за допомогою HTML 5?

37. Поясніть зміст термінів «веб-сокет» і «веб-сховище». Для чого вони використовуються?

38. Яким чином реалізовано підвищення продуктивності веб-застосувань?

39. Яким чином HTML 5 дозволяє веб-застосуванням виконуватися в автономному режимі?

40. Поясніть призначення функції геолокації у веб-застосуваннях.

### Список літератури

1. Макдональд М. HTML5. Недостающее руководство / М. Макдональд. – СПб: БВХ-Петербург, 2016. – 480 с.

2. Макдональд М. Веб-разработка. Исчерпывающее руководство / М. Макдональд. – СПб: Питер, 2017. – 640 с.

3. Роббинс Дж. HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дж. Роббинс. – М.: Эскимо, 2014. – 528 с.

4. Робсон Э. Изучаем HTML, XHTML и CSS / Э. Робсон, Э. Фримен. – СПб: Питер, 2016. – 720 с.

5. Фрейн Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Б. Фрейн. – СПб: Питер, 2016. – 272 с.

## ЗМІСТ

ВСТУП.....	3
1. ТЕОРЕТИЧНИЙ МАТЕРІАЛ .....	4
1.1. Принцип роботи мережі веб.....	4
1.2. Базові поняття HTML .....	5
1.3. Правила створення html-документів .....	6
1.4. Тегова модель документа .....	7
1.5. Загальна структура html-документа .....	8
1.6. URL-адреси.....	12
1.7. Задання кольору в HTML .....	13
1.8. Відтворення html-документів.....	14
1.9. Структурування тіла документа .....	17
1.10. Форматування тексту.....	18
1.11. Використання шрифтів.....	20
1.12. Гіперпосилання в html-документах .....	21
1.13. Правила використання посилань .....	22
1.14. Зображення в html-документах .....	23
1.14.1. Графічні формати Інтернету.....	23
1.14.2. Вставка зображень у html-документ .....	24
1.14.3. Графічні гіперпосилання.....	25
1.14.4. Карти-зображення .....	26
1.15. Списки в html-документах .....	28
1.16. Таблиці в html-документах.....	32
1.16.1. Форматування таблиць.....	37
1.16.2. Успадкування параметрів вирівнювання .....	39
1.16.3. Поля комірки.....	40
1.17. Вбудовані фрейми.....	40

1.18. Форми в html-документах.....	43
1.18.1. Створення форми .....	43
1.18.2. Елементи керування форми .....	44
1.19. Універсальні атрибути html-елементів.....	49
1.20. Події в html-документі.....	50
1.21. Включення скриптів у html-документи.....	51
1.22. Особливості HTML 5 .....	52
2. ПОСТАНОВКА ЗАВДАННЯ НА ЛАБОРАТОРНІ РОБОТИ.....	55
2.1. Завдання на лабораторну роботу «Базові можливості мови HTML» .....	55
2.2. Завдання на лабораторну роботу «Створення сайту за допомогою HTML».....	62
Контрольні запитання .....	63
Список літератури .....	65

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ  
до лабораторних робіт  
«Створення веб-сайту за допомогою HTML»  
з курсу «Основи Internet-технологій»  
для студентів спеціальностей 124 «Системний аналіз»,  
186 «Видавництво і поліграфія»

Укладачі: МАРЧЕНКО Наталя Андріївна  
КОЛБАСІН Вячеслав Олександрович  
ПРОКОПЕНКОВ Володимир Пилипович

Відповідальний за випуск проф. Куценко О. С.  
Роботу до видання рекомендував проф. Безменов М. І.

Редактор О. І. Шпільова

План 2017 р., поз 106

Підписано до друку 17.05.17р. Формат 60x84 1/16. Папір офсетний.  
Друк – ризографія. Гарнітура Таймс. Ум. друк. арк. 3,5.  
Наклад 50 прим. Зам. № 286/17 . Ціна договірна

---

Видавничий центр НТУ «ХП»

Свідоцтво про державну реєстрацію ДК №3657 від 24.12.2009 р.  
61002, Харків, вул. Кирпичова, 2

---

Друкарня «ФОП Пісня О. В.»

Свідоцтво про державну реєстрацію ВО2 № 248750 від 13.09.2007 р.  
61002, Харків, вул. Гіршмана, 16а, кв. 21, тел. (057) 764-20-28