



RT.Academy

# HTML

HyperText Markup Language

# Зміст

---

- [HTML](#)
- [Історія HTML](#)
- [Початок роботи з HTML](#)
- [Початок роботи з HTML: Частини елемента](#)
- [Початок роботи з HTML](#)
- [Початок роботи з HTML: Атрибути](#)
- [Початок роботи з HTML: Булеві атрибути](#)
- [Анатомія HTML-документа](#)
- [Кодування символів](#)
- [Кодування символів: Unicode](#)
- [Завдання #2.1](#)
- [Інструменти розробника: Chrome DevTools](#)
- [Інструменти розробника: Firefox Developer Tools](#)
- [Теги HTML: Main root](#)
- [Теги HTML: Document metadata](#)
- [Теги HTML: Sectioning root](#)
- [Теги HTML: Content sectioning](#)
- [Теги HTML: Content sectioning](#)
- [Теги HTML: Text content](#)
- [Теги HTML: Inline text semantics](#)
- [Теги HTML: Image and multimedia](#)
- [Теги HTML: Embedded content](#)
- [Теги HTML: SVG and MathML](#)
- [Теги HTML: Scripting](#)
- [Теги HTML: Demarcating edits](#)
- [Теги HTML: Table content](#)
- [Теги HTML: Forms](#)
- [Теги HTML: Interactive elements](#)
- [Теги HTML: Web Components](#)
- [Теги HTML: Obsolete and deprecated elements](#)
- [Розмітка тексту: Абзаци](#)
- [Розмітка тексту: Заголовки](#)
- [Розмітка тексту: Семантика та стилі тексту](#)
- [Семантичні елементи](#)
- [Блокові та рядкові елементи](#)

# Зміст

---

- [Елементи-контейнери](#)
- [Блокові та рядкові елементи](#)
- [Розмітка тексту: Списки](#)
- [Спеціальні символи HTML](#)
- [Завдання #2.2](#)
- [URI](#)
- [URL](#)
- [URL: Типи](#)
- [URL: Приклади відносних URL](#)
- [Зображення](#)
- [Посилання](#)
- [Посилання: Якір](#)
- [Посилання: Електронна пошта](#)
- [Посилання: Інші види](#)
- [Відео](#)
- [Аудіо](#)
- [<iframe>](#)
- [Завдання #2.3](#)
- [Завдання #2.4](#)
- [Структуровані дані](#)
- [Структуровані дані: Schema.org](#)
- [Структуровані дані: Open Graph](#)
- [Структуровані дані: Мікроформати](#)
- [Структуровані дані: JSON-LD+Schema.org: Приклад](#)
- [Структуровані дані: Приклади результатів](#)
- [Завдання #2.5](#)
- [Таблиці](#)
- [Таблиці: Заголовки](#)
- [Таблиці: Об'єднання кількох рядків або стовпців](#)
- [Завдання #2.6](#)
- [Форми](#)
- [Завдання #2.7](#)
- [Елементи форм: Поля введення тексту](#)
- [Елементи форм: Багаторядкові текстові поля](#)
- [Елементи форм: Випадаючі елементи](#)

# Зміст



- [Елементи форм: Check-елементи](#)
- [Елементи форм: Кнопки](#)
- [Елементи форм: Розширені елементи](#)
- [Елементи форм: Дата/час](#)
- [Завдання #2.8](#)
- [Завдання #2.8 \(Додатково\)](#)
- [HTML: Living Standard](#)
- [Шпаргалки по HTML](#)

# HTML



**HTML** (HyperText Markup Language — мова розмітки гіпертексту) — це мова тегів, засобами якої здійснюється розмічання вебсторінок для мережі Інтернет.

Браузери отримують HTML-документи з вебсервера та відображають їх на екрані монітора комп'ютера або мобільного пристрою.

**HTML** надає засоби для:

- створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації із Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об'єктів до тексту.

**HTML не є мовою програмування!** Це мова розмітки, і використовується, щоб повідомляти вашому браузеру, як відображати веб-сторінки, які ви відвідуєте.

# Історія HTML

## HTML

**03 листопада 1992**  
найперша версія,  
орієнтована лише на  
текст

1992

## HTML 3.2

**14 січня 1997**  
були додані численні  
можливості, такі як  
таблиці, обтікання  
текстом зображень

1997

## HTML 5.0

**28 жовтня 2014**  
розширює, покращує та раціоналізує  
розмітку для документів, додано  
семантичну розмітку, додано багато нових  
синтаксичних функцій.

2014

## HTML 2.0

**24 листопада 1995**  
версія з підтримкою  
форм

1995

## HTML 4.0

**18 грудня 1997**  
були додані таблиці стилів, скрипти та  
фрейми. Також відбулось розділення  
стандартів на Strict, Frameset,  
Transitional

1997

## HTML 5.3

**наступна версія**

<https://html.spec.whatwg.org/multipage/>

now

# Початок роботи з HTML

HTML-документ - це звичайний текстовий документ, може бути створений як в звичайному текстовому редакторі ("Блокнот"), так і в спеціалізованому, з підсвічуванням коду ("Notepad++", "Visual Studio Code" і т.п.). HTML-документ має розширення [.html](#)

HTML-документ складається з дерева HTML-елементів і тексту.

Кожний елемент позначається в вихідному документі **початковим тегом** (Opening tag) і **кінцевим тегом** (Closing tag).

У кінцевих тегах перед ім'ям ставиться символ / («слеш»).

Теги бувають парними і непарними. У парних тегів, на відміну від непарних, є кінцевий тег.

Приклади непарних тегів:      `<meta>`   `<img>`   `<br>`   `<hr>`

Приклади парних тегів:      `<p></p>`   `<b></b>`   `<span></span>`

# Початок роботи з HTML: Частини елемента

Початковий тег (Opening tag)

Кінцевий тег (Closing tag)

`<p>`Я дуже люблю котів`</p>`

Контент/Вміст (Content)

Елемент (Element)



# Початок роботи з HTML: Частина елемента



Основні частини елемента такі:

1. **Початковий тег** (Opening tag): містить назву елемента (в цьому випадку, "p"), загорнену в кутові дужки. Цей тег позначає початок елемента, або місце, де елемент починає діяти. У даному випадку — це місце, де починається параграф.
2. **Кінцевий тег** (Closing tag): такий самий тег, як і початковий, тільки тепер він містить скісну риску перед назвою елемента. Цей тег позначає місце закінчення елемента — у цьому випадку, місце, де закінчується параграф. Одна з найпоширеніших помилок початківців — це забути поставити кінцевий тег, що може призвести до несподіваних результатів.
3. **Контент/Вміст** (Content): вміст елемента, у цьому випадку — просто текст "Я дуже люблю котів".
4. **Елемент** (Element): початковий тег разом із кінцевим тегом та вмістом між ними складають елемент.

# Початок роботи з HTML



Ви також можете розміщувати елементи всередині інших елементів.

Якщо ми хочемо заявити, що ми **дуже** любимо котів, ми можемо огорнути слово "дуже" в елемент `<strong>`, який вказує, що слово має бути сильно акцентовано:

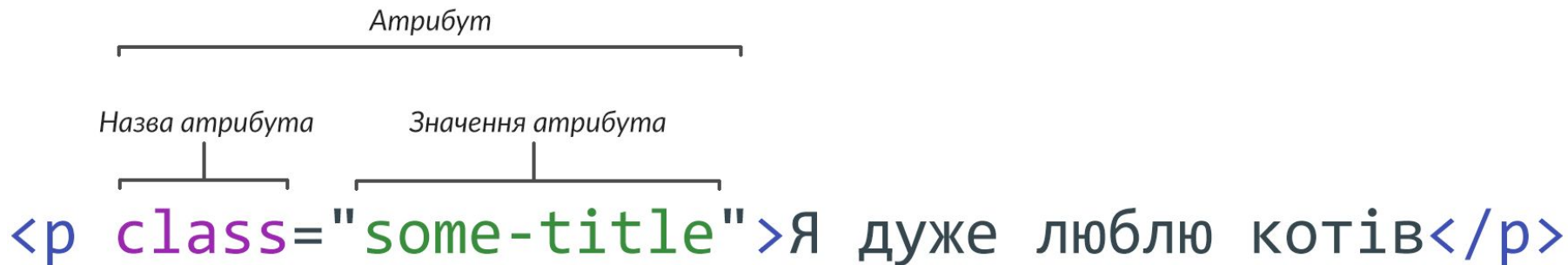
```
<p>Я <strong>дуже</strong> люблю котів.</p>
```

Але ви повинні переконатися, що ваші елементи правильно вкладені:

в прикладі вище ми відкрили першим елемент `<p>`, потім елемент `<strong>`, потім ми повинні закрити спочатку внутрішній елемент `<strong>`, потім зовнішній `<p>`.

# Початок роботи з HTML: Атрибути

Елементи також можуть мати атрибути (властивості), які виглядають так:



Атрибут завжди повинен мати:

1. Пробіл між ним та початковим тегом (або попереднім атрибутом, якщо елемент вже має один або кілька атрибутів).
2. Назва атрибута, за яким слідує знак рівності (не може бути в одному тегу однакових назв атрибутів).
3. Значення атрибута у лапках (у випадку декількох значень - відокремлюються пробілами).

# Початок роботи з HTML: Булеві атрибути

Іноді ви можете побачити в коді лише назву атрибута, без вказаного значення.

Це цілком нормальні атрибути, вони називаються булевими (bool).

Вони мають лише одне значення, зазвичай таке ж саме, як і назва самого атрибута.

Для прикладу візьмемо атрибут `disabled`, який робить поля форми неактивними, щоб користувачі не могли їх заповнити.

```
<input type="text" disabled="disabled">
```

Скорочений запис цілком може мати такий вигляд:

```
<input type="text" disabled>
```

# Анатомія HTML-документа



```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="utf-8">
  <title>Назва сторінки</title>
</head>
<body>
  <header>
    <h1>Назва сайту</h1>
  </header>
  <nav>
    Меню навігації
  </nav>
  <main>
    Основний текст
  </main>
  <footer>
    Футер сторінки
  </footer>
</body>
</html>
```

# Анатомія HTML-документа

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="utf-8">
  <title>Назва сторінки</title>
</head>
<body>
  <header>
    <h1>Назва сайту</h1>
  </header>
  <nav>
    Меню навігації
  </nav>
  <main>
    Основний текст
  </main>
  <footer>
    Футер сторінки
  </footer>
</body>
</html>
```

Елемент `<!DOCTYPE>` призначений для вказання типу поточного документа DTD (document type definition, опис типу документа).

Це необхідно, щоб браузер розумів, як слід інтерпретувати поточну веб-сторінку, тому що HTML існує в декількох версіях.

# Анатомія HTML-документа

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="utf-8">
  <title>Назва сторінки</title>
</head>
<body>
  <header>
    <h1>Назва сайту</h1>
  </header>
  <nav>
    Меню навігації
  </nav>
  <main>
    Основний текст
  </main>
  <footer>
    Футер сторінки
  </footer>
</body>
</html>
```

`<html></html>` — елемент `<html>`.

Цей елемент обертає весь контент на всій сторінці, і іноді відомий як кореневий елемент.

Обов'язковий атрибут `lang` для встановлення мови, що буде використаний на сторінці.

Базові коди мови:

- англійська - en
- американська англійська - en-us
- російська - ru
- українська - uk

# Анатомія HTML-документа

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="utf-8">
  <title>Назва сторінки</title>
</head>
<body>
  <header>
    <h1>Назва сайту</h1>
  </header>
  <nav>
    Меню навігації
  </nav>
  <main>
    Основний текст
  </main>
  <footer>
    Футер сторінки
  </footer>
</body>
</html>
```

`<head></head>` — елемент `<head>`.

Цей елемент виступає в якості контейнера для всього, що ви побажаєте включити на HTML сторінку, але не є контентом, який ви показуєте користувачам вашої сторінки.

До них відносяться такі речі, як ключові слова і опис сторінки, які будуть з'являтися в результатах пошуку, CSS стилі нашого контенту, кодування і багато іншого.

Щоб браузер розумів, що має справу з кодуванням [UTF-8](#) і додається

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



# Анатомія HTML-документа

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="utf-8">
  <title>Назва сторінки</title>
</head>
<body>
  <header>
    <h1>Назва сайту</h1>
  </header>
  <nav>
    Меню навігації
  </nav>
  <main>
    Основний текст
  </main>
  <footer>
    Футер сторінки
  </footer>
</body>
</html>
```

`<title></title>` — елемент `<title>`.

Цей елемент встановлює заголовок для вашої сторінки у вкладці браузера, результатів пошуку (наприклад у результатах Google) та для назви сторінки, коли ви додаєте її в закладки/обрані.

`<body></body>` — елемент `<body>`.

У ньому міститься весь контент, який ви хочете показувати користувачам, коли вони відвідують вашу сторінку, будь то текст, зображення, відео, аудіо або будь що ще.

# Анатомія HTML-документа

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="utf-8">
  <title>Назва сторінки</title>
</head>
<body>
  <header>
    <h1>Назва сайту</h1>
  </header>
  <nav>
    Меню навігації
  </nav>
  <main>
    Основний текст
  </main>
  <footer>
    Футер сторінки
  </footer>
</body>
</html>
```

`<header></header>` — являє собою вступний контент, зазвичай групу вступних або навігаційних засобів. Він може містити інші елементи-заголовки, а також логотип, форму пошуку, ім'я автора та інші елементи.

`<h1></h1>` - заголовок першого рівня. Бажано щоб він на сторінці був один. `<h1>` це найбільший заголовок і `<h6>` - найменший

`<nav></nav>` - визначає окрему секцію документа, призначення якої позначення посилань навігації (як всередині поточного документа, так і на інші сторінки).

# Анатомія HTML-документа

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="utf-8">
  <title>Назва сторінки</title>
</head>
<body>
  <header>
    <h1>Назва сайту</h1>
  </header>
  <nav>
    Меню навігації
  </nav>
  <main>
    Основний текст
  </main>
  <footer>
    Футер сторінки
  </footer>
</body>
</html>
```

`<main></main>` — призначений для основного контенту (вмісту) `<body>` документа (сторінки). Основний контент складається з контенту, який безпосередньо відноситься до головної теми документа або її розвиває.

`<footer></footer>` — є нижній колонтитул (футер, підвал) для свого найближчого секційного контенту (sectioning content) або секційного кореня (sectioning root). Футер зазвичай містить інформацію про автора розділу, інформацію про авторське право або посилання на пов'язані документи.

# Кодування символів



В обчислювальних машинах символи не можуть зберігатися інакше, як у вигляді послідовностей біт. Для передачі символу і його коректного відображення йому повинна відповідати унікальна послідовність нулів і одиниць. Для цього були розроблені таблиці кодувань.

**Кодування** або **таблиця кодування символів** (*character set, charset*) - це однозначна відповідність між підмножиною цілих чисел і деяким набором символів.

Символ може бути буквою, може відповідати звуку мови і може бути представлений графічним знаком.

Визначальним для будь-якого кодування є кількість охоплених ним кодів і, відповідно, символів.

Оскільки тексти в комп'ютері зберігаються в вигляді послідовності байтів, більшість кодувань природним чином розпадаються на:

- однобайтові (або восьмибітні) здатні закодувати не більш 256 символів.  
Наприклад: [ASCII](#), KOI8-R, Windows-1251
- двобайтові (або шістнадцятибітні) здатні закодувати максимально 65536 символів.  
Наприклад: [Unicode](#)

# Кодування символів: Unicode



У 1991 році була зроблена спроба створити єдину універсальну двобайтову таблицю кодування символів, що охоплювали б всі алфавіти і ієрогліфічні системи світу.

Результатом став стандарт під назвою **Unicode**, що покриває не тільки системи писемності всіх живих і більшості мертвих мов світу, а й безліч музичних, математичних, хімічних та інших символів.

Unicode став стандартом в Інтернет, а особливо його формат під назвою **UTF-8**.

Ця «похідна» кодування використовується для запису символів ланцюжками байтів різної довжини (від одного до шести), які за допомогою нескладного алгоритму перетворюються в Unicode-коди, причому більш вживаним символам відповідають більш короткі ланцюжки.

Головна перевага цього формату - сумісність з ASCII для кодування будь-якого з перших 128 символів в UTF-8 досить одного байта (хоча, наприклад, для букв кирилиці потрібно вже по два байта).

Окрім UTF-8, існує також UTF-16 та UTF-32.

Детальніше:

<https://en.wikipedia.org/wiki/UTF-8>

## Завдання #2.1

1. Встановіть та налаштуйте Google Chrome та JetBrains PhpStorm, використовуючи інструкцію [rtacademy.net/v3/settings.pdf](https://rtacademy.net/v3/settings.pdf)
2. Створіть файл `index.html` в корні вашого проекту з наступним змістом (що праворуч)
3. Відкрийте його в браузері Chrome, з використанням PhpStorm
4. Перевірте цю сторінку на помилки за допомогою HTML-валідатора [validator.w3.org](https://validator.w3.org)

*Примітка: рекомендується завжди перевіряти валідатором ваш HTML-документ.*

*Оскільки ви працюєте локально - для перевірки необхідно використовувати пункти або "Validate by File Upload" або "Validate by Direct Input"*

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <title>Назва сторінки</title>
</head>
<body>
    <header>
        <h1>Назва сайту</h1>
    </header>
    <nav>
        Меню навігації
    </nav>
    <main>
        Основний текст
    </main>
    <footer>
        Футер сторінки
    </footer>
</body>
</html>
```

# Інструменти розробника: Chrome DevTools



Chrome DevTools - це набір інструментів веб-розробника, вбудованих безпосередньо в браузер Google Chrome.

DevTools може допомогти вам швидко редагувати сторінки та швидко діагностувати проблеми, що, зрештою, допомагає швидше створювати кращі веб-сайти.

<https://developers.google.com/web/tools/chrome-devtools>

# Інструменти розробника: Firefox Developer Tools



Firefox Developer Tools — це набір інструментів веб-розробника, вбудованих у Firefox. Ви можете використовувати їх для вивчення, редагування та налагодження HTML, CSS та JavaScript.

<https://developer.mozilla.org/uk/docs/Tools>



# Теги HTML: Main root



`<html>`

The **HTML `<html>` element** represents the root (top-level element) of an HTML document, so it is also referred to as the *root element*. All other elements must be descendants of this element.

# Теги HTML: Document metadata

## `<base>`

The **HTML `<base>` element** specifies the base URL to use for all relative URLs in a document.

## `<head>`

The **HTML `<head>` element** contains machine-readable information (**metadata**) about the document, like its **title**, **scripts**, and **style sheets**.

## `<link>`

The **HTML External Resource Link element (`<link>`)** specifies relationships between the current document and an external resource. This element is most commonly used to link to **stylesheets**, but is also used to establish site icons (both "favicon" style icons and icons for the home screen and apps on mobile devices) among other things.

## `<meta>`

The **HTML `<meta>` element** represents **metadata** that cannot be represented by other HTML meta-related elements, like `<base>`, `<link>`, `<script>`, `<style>` or `<title>`.

## `<style>`

The **HTML `<style>` element** contains style information for a document, or part of a document.

## `<title>`

The **HTML Title element (`<title>`)** defines the document's title that is shown in a **browser's** title bar or a page's tab.

# Теги HTML: Sectioning root



`<body>`

The **HTML <body> Element** represents the content of an HTML document.

There can be only one `<body>` element in a document.

# Теги HTML: Content sectioning

## `<address>`

The **HTML `<address>` element** indicates that the enclosed HTML provides contact information for a person or people, or for an organization.

## `<article>`

The **HTML `<article>` element** represents a self-contained composition in a document, page, application, or site, which is intended to be independently distributable or reusable (e.g., in syndication).

## `<aside>`

The **HTML `<aside>` element** represents a portion of a document whose content is only indirectly related to the document's main content.

## `<footer>`

The **HTML `<footer>` element** represents a footer for its nearest [sectioning content](#) or [sectioning root](#) element. A footer typically contains information about the author of the section, copyright data or links to related documents.

## `<header>`

The **HTML `<header>` element** represents introductory content, typically a group of introductory or navigational aids. It may contain some heading elements but also a logo, a search form, an author name, and other elements.

## `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`

The **HTML `<h1>`–`<h6>` elements** represent six levels of section headings. `<h1>` is the highest section level and `<h6>` is the lowest.

## `<hgroup>`

The **HTML `<hgroup>` element** represents a multi-level heading for a section of a document. It groups a set of `<h1>`–`<h6>` elements.

## `<main>`

The **HTML `<main>` element** represents the dominant content of the `<body>` of a document. The main content area consists of content that is directly related to or expands upon the central topic of a document, or the central functionality of an application.

# Теги HTML: Content sectioning



## `<nav>`

The **HTML `<nav>` element** represents a section of a page whose purpose is to provide navigation links, either within the current document or to other documents. Common examples of navigation sections are menus, tables of contents, and indexes.

## `<section>`

The **HTML `<section>` element** represents a standalone section — which doesn't have a more specific semantic element to represent it — contained within an HTML document.

# Теги HTML: Text content

## `<blockquote>`

The **HTML `<blockquote>` Element** (or *HTML Block Quotation Element*) indicates that the enclosed text is an extended quotation. Usually, this is rendered visually by indentation (see [Notes](#) for how to change it). A URL for the source of the quotation may be given using the **cite** attribute, while a text representation of the source can be given using the `<cite>` element.

## `<dd>`

The **HTML `<dd>` element** provides the description, definition, or value for the preceding term (`<dt>`) in a description list (`<dl>`).

## `<div>`

The **HTML Content Division element** (`<div>`) is the generic container for flow content. It has no effect on the content or layout until styled using [CSS](#).

## `<dl>`

The **HTML `<dl>` element** represents a description list. The element encloses a list of groups of terms (specified using the `<dt>` element) and descriptions (provided by `<dd>` elements). Common uses for this element are to implement a glossary or to display metadata (a list of key-value pairs).

## `<dt>`

The **HTML `<dt>` element** specifies a term in a description or definition list, and as such must be used inside a `<dl>` element.

## `<figcaption>`

The **HTML `<figcaption>` or Figure Caption element** represents a caption or legend describing the rest of the contents of its parent `<figure>` element.

## `<figure>`

The **HTML `<figure>` (Figure With Optional Caption) element** represents self-contained content, potentially with an optional caption, which is specified using the (`<figcaption>`) element.

# Теги HTML: Text content



`<hr>`

The **HTML `<hr>` element** represents a thematic break between paragraph-level elements: for example, a change of scene in a story, or a shift of topic within a section.

`<li>`

The **HTML `<li>` element** is used to represent an item in a list.

`<ol>`

The **HTML `<ol>` element** represents an ordered list of items — typically rendered as a numbered list.

`<p>`

The **HTML `<p>` element** represents a paragraph.

`<pre>`

The **HTML `<pre>` element** represents preformatted text which is to be presented exactly as written in the HTML file.

`<ul>`

The **HTML `<ul>` element** represents an unordered list of items, typically rendered as a bulleted list.

# Teri HTML: Inline text semantics

`<a>`

The **HTML <a> element** (or *anchor* element), with its `href` attribute, creates a hyperlink to web pages, files, email addresses, locations in the same page, or anything else a URL can address.

`<abbr>`

The **HTML Abbreviation element** (`<abbr>`) represents an abbreviation or acronym; the optional `title` attribute can provide an expansion or description for the abbreviation.

`<b>`

The **HTML Bring Attention To element** (`<b>`) is used to draw the reader's attention to the element's contents, which are not otherwise granted special importance.

`<bdi>`

The **HTML Bidirectional Isolate element** (`<bdi>`) tells the browser's bidirectional algorithm to treat the text it contains in isolation from its surrounding text.

`<bdo>`

The **HTML Bidirectional Text Override element** (`<bdo>`) overrides the current directionality of text, so that the text within is rendered in a different direction.

`<br>`

The **HTML <br> element** produces a line break in text (carriage-return). It is useful for writing a poem or an address, where the division of lines is significant.

`<cite>`

The **HTML Citation element** (`<cite>`) is used to describe a reference to a cited creative work, and must include the title of that work.

`<code>`

The **HTML <code> element** displays its contents styled in a fashion intended to indicate that the text is a short fragment of computer code.



# Termin HTML: Inline text semantics

## `<data>`

The **HTML `<data>` element** links a given piece of content with a machine-readable translation. If the content is time- or date-related, the `<time>` element must be used.

## `<dfn>`

The **HTML Definition element (`<dfn>`)** is used to indicate the term being defined within the context of a definition phrase or sentence.

## `<em>`

The **HTML `<em>` element** marks text that has stress emphasis. The `<em>` element can be nested, with each level of nesting indicating a greater degree of emphasis.

## `<i>`

The **HTML Interesting Text element (`<i>`)** represents a range of text that is set off from the normal text for some reason.

## `<kbd>`

The **HTML Keyboard Input element (`<kbd>`)** represents a span of inline text denoting textual user input from a keyboard, voice input, or any other text entry device.

## `<mark>`

The **HTML Mark Text element (`<mark>`)** represents text which is **marked** or **highlighted** for reference or notation purposes, due to the marked passage's relevance or importance in the enclosing context.

## `<q>`

The **HTML `<q>` element** indicates that the enclosed text is a short inline quotation. Most modern browsers implement this by surrounding the text in quotation marks.

## `<rb>`

The **HTML Ruby Base (`<rb>`) element** is used to delimit the base text component of a `<ruby>` annotation, i.e. the text that is being annotated.

## `<rp>`

The **HTML Ruby Fallback Parenthesis (`<rp>`) element** is used to provide fall-back parentheses for browsers that do not support display of ruby annotations using the `<ruby>` element.

# Теги HTML: Inline text semantics

`<rt>`

The **HTML Ruby Text (`<rt>`) element** specifies the ruby text component of a ruby annotation, which is used to provide pronunciation, translation, or transliteration information for East Asian typography. The `<rt>` element must always be contained within a `<ruby>` element.

`<rtc>`

The **HTML Ruby Text Container (`<rtc>`) element** embraces semantic annotations of characters presented in a ruby of `<rb>` elements used inside of `<ruby>` element. `<rb>` elements can have both pronunciation (`<rt>`) and semantic (`<rtc>`) annotations.

`<ruby>`

The **HTML `<ruby>` element** represents a ruby annotation. Ruby annotations are for showing pronunciation of East Asian characters.

`<s>`

The **HTML `<s>` element** renders text with a strikethrough, or a line through it. Use the `<s>` element to represent things that are no longer relevant or no longer accurate. However, `<s>` is not appropriate when indicating document edits; for that, use the `<del>` and `<ins>` elements, as appropriate.

`<samp>`

The **HTML Sample Element (`<samp>`)** is used to enclose inline text which represents sample (or quoted) output from a computer program.

`<small>`

The **HTML `<small>` element** represents side-comments and small print, like copyright and legal text, independent of its styled presentation. By default, it renders text within it one font-size smaller, such as from `small` to `x-small`.

# Теги HTML: Inline text semantics

## `<span>`

The **HTML `<span>` element** is a generic inline container for phrasing content, which does not inherently represent anything. It can be used to group elements for styling purposes (using the `class` or `id` attributes), or because they share attribute values, such as `lang`.

## `<strong>`

The HTML **Strong Importance Element** (`<strong>`) indicates that its contents have strong importance, seriousness, or urgency. Browsers typically render the contents in bold type.

## `<sub>`

The HTML **Subscript element** (`<sub>`) specifies inline text which should be displayed as subscript for solely typographical reasons.

## `<sup>`

The **HTML Superscript element** (`<sup>`) specifies inline text which is to be displayed as superscript for solely typographical reasons.

## `<time>`

The HTML **`<time>` element** represents a specific period in time.

## `<u>`

The HTML **Unarticulated Annotation Element** (`<u>`) represents a span of inline text which should be rendered in a way that indicates that it has a non-textual annotation.

## `<var>`

The HTML **Variable element** (`<var>`) represents the name of a variable in a mathematical expression or a programming context.

## `<wbr>`

The **HTML `<wbr>` element** represents a word break opportunity—a position within text where the browser may optionally break a line, though its line-breaking rules would not otherwise create a break at that location.

# Теги HTML: Image and multimedia

## `<area>`

The **HTML `<area>` element** defines a hot-spot region on an image, and optionally associates it with a [hypertext link](#). This element is used only within a `<map>` element.

## `<audio>`

The **HTML `<audio>` element** is used to embed sound content in documents. It may contain one or more audio sources, represented using the `src` attribute or the `<source>` element: the browser will choose the most suitable one. It can also be the destination for streamed media, using a [MediaStream](#).

## `<img>`

The **HTML `<img>` element** embeds an image into the document.

## `<map>`

The **HTML `<map>` element** is used with `<area>` elements to define an image map (a clickable link area).

## `<track>`

The **HTML `<track>` element** is used as a child of the media elements `<audio>` and `<video>`. It lets you specify timed text tracks (or time-based data), for example to automatically handle subtitles. The tracks are formatted in [WebVTT format](#) (`.vtt` files) — Web Video Text Tracks or [Timed Text Markup Language](#) (TTML).

## `<video>`

The **HTML Video element (`<video>`)** embeds a media player which supports video playback into the document. You can use `<video>` for audio content as well, but the `<audio>` element may provide a more appropriate user experience.

# Теги HTML: Embedded content

## `<embed>`

The **HTML `<embed>` element** embeds external content at the specified point in the document. This content is provided by an external application or other source of interactive content such as a browser plug-in.

## `<iframe>`

The **HTML Inline Frame element (`<iframe>`)** represents a nested [browsing context](#), embedding another HTML page into the current one.

## `<object>`

The **HTML `<object>` element** represents an external resource, which can be treated as an image, a nested browsing context, or a resource to be handled by a plugin.

## `<param>`

The **HTML `<param>` element** defines parameters for an `<object>` element.

## `<picture>`

The **HTML `<picture>` element** contains zero or more `<source>` elements and one `<img>` element to offer alternative versions of an image for different display/device scenarios.

## `<portal>`

The **HTML Portal element (`<portal>`)** enables the embedding of another HTML page into the current one for the purposes of allowing smoother navigation into new pages.

## `<source>`

The **HTML `<source>` element** specifies multiple media resources for the `<picture>`, the `<audio>` element, or the `<video>` element.

# Теги HTML: SVG and MathML



## `<svg>`

The `svg` element is a container that defines a new coordinate system and [viewport](#). It is used as the outermost element of SVG documents, but it can also be used to embed an SVG fragment inside an SVG or HTML document.

## `<math>`

The top-level element in MathML is `<math>`. Every valid MathML instance must be wrapped in `<math>` tags. In addition you must not nest a second `<math>` element in another, but you can have an arbitrary number of other child elements in it.

# Теги HTML: Scripting



## `<canvas>`

Use the **HTML `<canvas>` element** with either the [canvas scripting API](#) or the [WebGL API](#) to draw graphics and animations.

## `<noscript>`

The **HTML `<noscript>` element** defines a section of HTML to be inserted if a script type on the page is unsupported or if scripting is currently turned off in the browser.

## `<script>`

The **HTML `<script>` element** is used to embed or reference executable code; this is typically used to embed or refer to JavaScript code.

# Теги HTML: Demarcating edits



`<del>`

The **HTML `<del>` element** represents a range of text that has been deleted from a document.

`<ins>`

The **HTML `<ins>` element** represents a range of text that has been added to a document.



# Теги HTML: Table content

## `<caption>`

The **HTML `<caption>` element** specifies the caption (or title) of a table.

## `<col>`

The **HTML `<col>` element** defines a column within a table and is used for defining common semantics on all common cells. It is generally found within a `<colgroup>` element.

## `<colgroup>`

The **HTML `<colgroup>` element** defines a group of columns within a table.

## `<table>`

The **HTML `<table>` element** represents tabular data — that is, information presented in a two-dimensional table comprised of rows and columns of cells containing data.

## `<tbody>`

The **HTML Table Body element (`<tbody>`)** encapsulates a set of table rows (`<tr>` elements), indicating that they comprise the body of the table (`<table>`).

## `<td>`

The **HTML `<td>` element** defines a cell of a table that contains data. It participates in the *table model*.

## `<tfoot>`

The **HTML `<tfoot>` element** defines a set of rows summarizing the columns of the table.

## `<th>`

The **HTML `<th>` element** defines a cell as header of a group of table cells. The exact nature of this group is defined by the `scope` and `headers` attributes.

## `<thead>`

The **HTML `<thead>` element** defines a set of rows defining the head of the columns of the table.

## `<tr>`

The **HTML `<tr>` element** defines a row of cells in a table. The row's cells can then be established using a mix of `<td>` (data cell) and `<th>` (header cell) elements.

# Теги HTML: Forms

## `<button>`

The **HTML `<button>` element** represents a clickable button, used to submit [forms](#) or anywhere in a document for accessible, standard button functionality.

## `<datalist>`

The **HTML `<datalist>` element** contains a set of `<option>` elements that represent the permissible or recommended options available to choose from within other controls.

## `<fieldset>`

The **HTML `<fieldset>` element** is used to group several controls as well as labels (`<label>`) within a web form.

## `<form>`

The **HTML `<form>` element** represents a document section containing interactive controls for submitting information.

## `<input>`

The **HTML `<input>` element** is used to create interactive controls for web-based forms in order to accept data from the user; a wide variety of types of input data and control widgets are available, depending on the device and [user agent](#).

## `<label>`

The **HTML `<label>` element** represents a caption for an item in a user interface.

## `<legend>`

The **HTML `<legend>` element** represents a caption for the content of its parent `<fieldset>`.

## `<meter>`

The **HTML `<meter>` element** represents either a scalar value within a known range or a fractional value.

## `<optgroup>`

The **HTML `<optgroup>` element** creates a grouping of options within a `<select>` element.

# Теги HTML: Forms



## `<option>`

The **HTML `<option>` element** is used to define an item contained in a `<select>`, an `<optgroup>`, or a `<datalist>` element. As such, `<option>` can represent menu items in popups and other lists of items in an HTML document.

## `<output>`

The **HTML Output element (`<output>`)** is a container element into which a site or app can inject the results of a calculation or the outcome of a user action.

## `<progress>`

The **HTML `<progress>` element** displays an indicator showing the completion progress of a task, typically displayed as a progress bar.

## `<select>`

The **HTML `<select>` element** represents a control that provides a menu of options

## `<textarea>`

The **HTML `<textarea>` element** represents a multi-line plain-text editing control, useful when you want to allow users to enter a sizeable amount of free-form text, for example a comment on a review or feedback form.

# Termin HTML: Interactive elements



## `<details>`

The **HTML Details Element** (`<details>`) creates a disclosure widget in which information is visible only when the widget is toggled into an "open" state.

## `<dialog>`

The **HTML `<dialog>` element** represents a dialog box or other interactive component, such as a dismissable alert, inspector, or subwindow.

## `<menu>`

The **HTML `<menu>` element** represents a group of commands that a user can perform or activate. This includes both list menus, which might appear across the top of a screen, as well as [context menus](#), such as those that might appear underneath a button after it has been clicked.

## `<summary>`

The **HTML Disclosure Summary element** (`<summary>`) element specifies a summary, caption, or legend for a `<details>` element's disclosure box.

# Тегі HTML: Web Components



## `<slot>`

The **HTML `<slot>` element**—part of the [Web Components](#) technology suite—is a placeholder inside a web component that you can fill with your own markup, which lets you create separate DOM trees and present them together.

## `<template>`

The **HTML Content Template (`<template>`) element** is a mechanism for holding [HTML](#) that is not to be rendered immediately when a page is loaded but may be instantiated subsequently during runtime using JavaScript.

# Teri HTML: Obsolete and deprecated elements

## `<acronym>`

The HTML Acronym Element (`<acronym>`) allows authors to clearly indicate a sequence of characters that compose an acronym or abbreviation for a word.

## `<applet>`

The obsolete **HTML Applet Element** (`<applet>`) embeds a Java applet into the document; this element has been deprecated in favor of `<object>`.

## `<basefont>`

The obsolete **HTML Base Font element** (`<basefont>`) sets a default font face, size, and color for the other elements which are descended from its parent element.

## `<bgsound>`

The Internet Explorer only **HTML Background Sound element** (`<bgsound>`) sets up a sound file to play in the background while the page is used; use `<audio>` instead.

## `<big>`

The obsolete **HTML Big Element** (`<big>`) renders the enclosed text at a font size one level larger than the surrounding text (medium becomes large, for example).

## `<blink>`

The **HTML Blink Element** (`<blink>`) is a non-standard element which causes the enclosed text to flash slowly.

## `<center>`

The obsolete **HTML Center Element** (`<center>`) is a [block-level element](#) that displays its block-level or inline contents centered horizontally within its containing element.

## `<command>`

The **HTML Command element** (`<command>`) represents a command which the user can invoke. Commands are often used as part of a context menu or toolbar.

# Teri HTML: Obsolete and deprecated elements

## `<content>`

The **HTML `<content>` element**—an obsolete part of the [Web Components](#) suite of technologies—was used inside of [Shadow DOM](#) as an [insertion point](#), and wasn't meant to be used in ordinary HTML.

## `<dir>`

The obsolete **HTML Directory element** (`<dir>`) is used as a container for a directory of files and/or folders, potentially with styles and icons applied by the [user agent](#).

## `<element>`

The obsolete **HTML `<element>` element** was part of the [Web Components](#) specification; it was intended to be used to define new custom DOM elements.

## `<font>`

The *HTML Font Element* (`<font>`) defines the font size, color and face for its content.

## `<frame>`

`<frame>` is an HTML element which defines a particular area in which another HTML document can be displayed. A frame should be used within a `<frameset>`.

## `<frameset>`

The **HTML `<frameset>` element** is used to contain `<frame>` elements.

## `<image>`

The obsolete **HTML Image element** (`<image>`) is an obsolete remnant of an ancient version of HTML lost in the mists of time; use the standard `<img>` element instead.

## `<isindex>`

`<isindex>` was an obsolete HTML element that put a text field in a page for querying the document.

# Teri HTML: Obsolete and deprecated elements

## `<keygen>`

The HTML `<keygen>` element exists to facilitate generation of key material, and submission of the public key as part of an [HTML form](#). This mechanism is designed for use with Web-based certificate management systems. It is expected that the `<keygen>` element will be used in an HTML form along with other information needed to construct a certificate request, and that the result of the process will be a signed certificate.

## `<listing>`

The *HTML Listing Element* (`<listing>`) renders text between the start and end tags without interpreting the HTML in between and using a monospaced font. The HTML 2 standard recommended that lines shouldn't be broken when not greater than 132 characters.

## `<marquee>`

The HTML `<marquee>` element is used to insert a scrolling area of text. You can control what happens when the text reaches the edges of its content area using its attributes.

## `<menuitem>`

The **HTML `<menuitem>` element** represents a command that a user is able to invoke through a popup menu. This includes context menus, as well as menus that might be attached to a menu button.

## `<multicol>`

The **HTML Multi-Column Layout element** (`<multicol>`) was an experimental element designed to allow multi-column layouts and must not be used.

## `<nextid>`

`<nextid>` is an obsolete HTML element that served to enable the NeXT web designing tool to generate automatic NAME labels for its anchors.

## `<nobr>`

The non-standard, obsolete HTML `<nobr>` element prevents the text it contains from automatically wrapping across multiple lines, potentially resulting in the user having to scroll horizontally to see the entire width of the text.



# Terri HTML: Obsolete and deprecated elements

## `<noembed>`

The **`<noembed>`** element is an obsolete, non-standard way to provide alternative, or "fallback", content for browsers that do not support the `<embed>` element or do not support the type of [embedded content](#) an author wishes to use.

## `<noframes>`

The obsolete HTML **No Frames** or **frame fallback** element, **`<noframes>`**, provides content to be presented in browsers that don't support (or have disabled support for) the `<frame>` element.

## `<plaintext>`

The *HTML Plaintext Element* (`<plaintext>`) renders everything following the start tag as raw text, ignoring any following HTML.

## `<shadow>`

The **HTML `<shadow>` element**—an obsolete part of the [Web Components](#) technology suite—was intended to be used as a shadow DOM [insertion point](#).

## `<spacer>`

**`<spacer>`** is an obsolete HTML element which allowed insertion of empty spaces on pages. It was devised by Netscape to accomplish the same effect as a single-pixel layout image, which was something web designers used to use to add white spaces to web pages without actually using an image. However, `<spacer>` no longer supported by any major browser and the same effects can now be achieved using simple CSS.

## `<strike>`

The **HTML `<strike>` element** (or *HTML Strikethrough Element*) places a strikethrough (horizontal line) over text.

## `<tt>`

The obsolete **HTML Teletype Text element** (`<tt>`) creates inline text which is presented using the [user agent's](#) default monospace font face.

## `<xmp>`

The *HTML Example Element* (`<xmp>`) renders text between the start and end tags without interpreting the HTML in between and using a monospaced font. The HTML2 specification recommended that it should be rendered wide enough to allow 80 characters per line.

# Розмітка тексту: Абзаци



Елемент `<p>` - це абзац (параграф).

Абзаци зазвичай представлені в візуальному середовищі у вигляді блоків тексту, відокремлених від сусідніх блоків порожніми рядками і/або відступом в першому рядку.

Крім цього HTML-абзаци можуть являти собою структуру однотипного вмісту, наприклад зображень або полів форми.

`<p>`Це одиночний абзац`</p>`

`<p>`Ще один одиночний абзац`</p>`

# Розмітка тексту: Заголовки



Елементи заголовку дозволяють вам вказувати певні частини вашого контенту в якості заголовків або підзаголовків.

Так само, як книга має назву, назви глав і підзаголовків, HTML документ може містити те ж саме.

HTML включає шість рівнів заголовків від `<h1>` до `<h6>`

`<h1>`Мій головний заголовок`</h1>`

`<h2>`Мій заголовок верхнього рівня`</h2>`

`<h3>`Мій підзаголовок`</h3>`

`<h4>`Мій під-підзаголовок`</h4>`

`<h5>`Мій під-під-підзаголовок`</h5>`

`<h6>`Мій під-під-під-підзаголовок`</h6>`

# Розмітка тексту: Семантика та стилі тексту



Елемент `<b>` є частиною тексту, що стилістично відрізняється від нормального тексту та не носить якогось спеціального значення або важливості, і як правило виділено жирним шрифтом.

Елемент `<em>` призначений для слів, які мають підкреслений акцент в порівнянні з іншим текстом, який часто обмежується словом або словами речення і впливає на зміст самого речення. Зазвичай цей елемент відображається курсивом.

Елемент `<mark>` представляє текст, виділений в довідкових цілях через свою актуальність в певному контексті. Наприклад, він може бути використаний на сторінці з результатом пошуку, в якій виділяється кожен екземпляр шуканого слова.

Елемент `<strong>` підкреслює важливість, серйозність або терміновість свого вмісту, також може бути використаний для позначення попередження або застереження. Як правило виділено жирним шрифтом.

# Семантичні елементи



- <article>
- <aside>
- <details>
- <em>
- <figcaption>
- <figure>
- <form>
- <footer>
- <header>
- <h1> ... <h6>
- <main>
- <mark>
- <nav>
- <section>
- <strong>
- <summary>
- <table>
- <time>
- <video>

# Блокові та рядкові елементи



Історично HTML-елементи було прийнято ділити на блокові (**block-level**) і рядкові (**inline**).

**Блокові елементи** займають всю ширину свого батька (контейнера), формально створюючи «блок» (звідси і назва).

Браузери зазвичай відображають блокові елементи з нового рядка до і після елемента.

**Рядкові елементи** це ті, які займають лише той простір, який обмежений цими тегами (яке не потребує нового рядка після кожного елемента).

Як правило, блокові елементи можуть містити рядкові елементи і інші блокові елементи. Але рядкові елементи не можуть містити в собі блокові.

Невід'ємною частиною цієї структурної відмінності є ідея, що блокові елементи створюють «більшу» структуру, ніж рядкові елементи.

За замовчуванням блокові елементи починаються з нового рядка, а рядкові можуть починатися в будь-якому місці рядка.

# Елементи-контейнери



Елемент поділу контенту HTML (`<div>`) є універсальним контейнером для потокового контенту. Він не впливає на контент або макет до тих пір, поки не буде стилізований за допомогою CSS.

Будучи "чистим" контейнером, елемент `<div>`, по суті, не представляє нічого.

```
<div>
  <p>Будь-який тип контенту.</p>
  <p>Все що завгодно!</p>
</div>
```

HTML-елемент `<span>` є основним рядковим контейнером для фразового контенту.

`<span>` дуже схожий на елемент `<div>`, але `<div>` є блоковим елементом, в той час як `<span>` є рядковим.

```
<p><span>Який-небудь</span> текст</p>
```

# Блокові та рядкові елементи

## Блокові елементи:

<address> <article> <aside> <blockquote> <details> <dialog> <dd> <div> <dl> <dt> <fieldset>  
<figcaption> <figure> <footer> <form> <h1> <h2> <h3> <h4> <h5> <h6> <header> <hgroup> <hr>  
<li> <main> <nav> <ol> <p> <pre> <section> <table> <ul>

## Рядкові елементи:

<a> <abbr> <acronym> <audio> <b> <bdi> <bdo> <big> <br> <button> <canvas> <cite> <code> <data>  
<datalist> <del> <dfn> <em> <embed> <i> <iframe> <img> <input> <ins> <kbd> <label> <map>  
<mark> <meter> <noscript> <object> <output> <picture> <progress> <q> <ruby> <s> <samp>  
<script> <select> <slot> <small> <span> <strong> <sub> <sup> <svg> <template> <textarea>  
<time> <u> <tt> <var> <video> <wbr>



# Розмітка тексту: Списки

Велика частина веб-контенту є списками і HTML має спеціальні елементи для них. Розмітка списку завжди складається щонайменше з двох елементів.

Найбільш поширеними типами списків є нумеровані і нелінійні списки:

1. Нелінійні списки - це списки, де порядок пунктів не має значення, як в списку покупок. Вони обертаються в елемент `<ul>` (unordered list).
2. Лінійні списки - це списки, де порядок пунктів має значення, як в рецепті. Вони обертаються в елемент `<ol>` (ordered list).

Кожен пункт всередині списків розташовується всередині елемента `<li>` (list item, елемент списку).

```
<ol>
  <li>Grow a long, majestic beard.</li>
  <li>Wear a tall, pointed hat.</li>
  <li>Have I mentioned the beard?</li>
</ol>
```

# Спеціальні символи HTML

Наступні спеціальні символи зарезервовані в HTML, оскільки саме вони складають розмітку мови HTML. Якщо ви використовуєте один із цих символів у статті, браузер спробує інтерпретувати його як HTML. Тому вам слід використовувати запис цих символів або за назвою об'єкта (&<назва>;) або за його ASCII-кодом (&#<код>;).

| Символ | Запис через ASCII-код | Запис через назву об'єкта |
|--------|-----------------------|---------------------------|
| "      | &#34;                 | &quot;                    |
| '      | &#39;                 | &apos;                    |
| &      | &#38;                 | &amp;                     |
| <      | &#60;                 | &lt;                      |
| >      | &#62;                 | &gt;                      |

Інші символи:

<https://www.html.am/reference/html-special-characters.cfm>

<https://psdtowp.net/html-codes-special-characters.html>

<https://dev.w3.org/html5/html-author/charref>

## Завдання #2.2

1. У створений файл [index.html](#) у [завданні #2.1](#) додайте заголовки від `<h1>` до `<h6>` на вашу сторінку в елемент `<main>`
2. Додайте кілька елементів абзац в кінець елемента `<main>`
3. Створіть два види списків (нумерований та нелінійний), мінімум на 3 пункти в кожному
4. Спробуйте змінити вміст елементів прямо на сторінці використовуючи [інструмент розробника Chrome DevTools](#)
5. Перевірте цю сторінку на помилки за допомогою HTML-валідатора [validator.w3.org](#)

### Назва сайту

Меню навігації

### Мій головний заголовок

#### Мій заголовок верхнього рівня

#### Мій підзаголовок

#### Мій під-підзаголовок

#### Мій під-під-підзаголовок

#### Мій під-під-під-підзаголовок

Це одиночний абзац

Ще один одиночний абзац

Елемент `<b>` є частиною тексту, що стилістично відрізняється від нормального тексту та не носить якогось спеціального значення або важливості, і як правило **виділено жирним шрифтом**.

Елемент `<em>` призначений для слів, які мають підкреслений акцент в порівнянні з іншим текстом, який часто обмежується словом або словами речення і впливає на зміст самого речення. Зазвичай цей елемент *відображається курсивом*.

Елемент `<mark>` представляє текст, виділений в довідкових цілях через свою актуальність в певному контексті. Наприклад, він може бути використаний на сторінці з результатом пошуку, в якій **виділяється кожен екземпляр шуканого слова**.

Елемент `<strong>` підкреслює важливість, серйозність або терміновість свого вмісту, також може бути використаний для позначення попередження або застереження. Як правило **виділено жирним шрифтом**.

- Пункт нелінійного списку №1
- Пункт нелінійного списку №2
- Пункт нелінійного списку №3

1. Пункт лінійного списку №1
2. Пункт лінійного списку №2
3. Пункт лінійного списку №3

Футер сторінки

# URI

**URI** (*Uniform Resource Identifier, Уніфікований ідентифікатор ресурсів*) - компактний рядок літер у кодуванні ASCII, який однозначно ідентифікує окремий абстрактний чи фізичний ресурс. Основне призначення таких ідентифікаторів — зробити можливою взаємодію з поданнями ресурсів через мережу, переважно Всесвітнє павутиння, використовуючи спеціальні протоколи.

**URI** визначається схемами, які визначають синтаксис та відповідні протоколи.

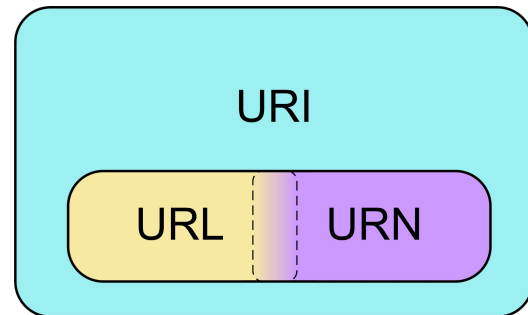
Найбільш поширеною формою **URI** є **URL** (*Uniform Resource Locator, уніфікований локатор ресурсів, єдиний вказівник на ресурс*), який неофіційно називають веб-адресою.

Рідше використовується уніфіковане ім'я ресурсів (**URN**), яке було розроблене, щоб доповнити **URL** забезпеченням механізму для ідентифікації ресурсів в просторі імен.

Раніше **URI** називався *Universal Resource Identifier* — *універсальний ідентифікатор ресурсів*.

Символи в **URI**, що не входять до ASCII, зашифровуються за допомогою [Відсоткового кодування](#).

<https://uk.wikipedia.org/wiki/URI>



# URL



**URL** (*Uniform Resource Locator, уніфікований локатор ресурсів*) - стандартизована адреса певного ресурсу (такого як документ, чи зображення) в Інтернет. Придуманий [Тімом Бернерс-Лі](#) для використання у [WWW](#).

## Структура URL

<схема> : // <логін> : <пароль> @ <хост> : <порт> / <шлях> ? <параметри> # <якір>

- схема — схема звернення до ресурсу, найчастіше це протокол (наприклад http, https, ftp, тощо)
- логін і пароль — відповідно ім'я користувача і пароль для доступу до ресурсу
- хост — повне [доменне ім'я](#) або [IP-адреса](#) ресурсу
- порт — порт, по якому буде проведено звернення до хоста
- шлях — використовується для уточнюючої вказівки місця знаходження ресурсу.
- параметри — рядок параметрів виду <параметр>=<значення>, розділених символом амперсанд (&)
- якір — призначений для внутрішньої адресації на ресурсі, який було отримано від сервера.

# URL: Типи



URL-адреса може бути як абсолютною, так і відносною.

**Абсолютна URL** вказує на місце розташування, яке визначається його абсолютним місцем розташування в Інтернет, включаючи протокол (зазвичай <http://> або <https://>) і доменне ім'я.

Абсолютна URL *завжди* буде вказувати на одне і те ж місце розташування, незалежно від того, де ця адреса використовується.

[https://www.google.com/images/branding/googlelogo/2x/googlelogo\\_color\\_272x92dp.png](https://www.google.com/images/branding/googlelogo/2x/googlelogo_color_272x92dp.png)

**Відносна URL** веде відлік від кореня сайту або поточного документа.

Відносна URL буде вказувати на різні місця, в залежності від того, де знаходиться файл, в якому він використовується.

[/images/branding/googlelogo/2x/googlelogo\\_color\\_272x92dp.png](/images/branding/googlelogo/2x/googlelogo_color_272x92dp.png)

# URL: Приклади відносних URL

## **/pages/about.html**

Така адреса буде коректною з будь-якої точки сайту. Це пов'язано з тим, що перший слеш означає якраз від чого вважати нашу адресу - від кореневого каталогу. Отже файл [about.html](#) буде доступним з будь-якого іншого файлу.

## **pages/about.html**

Така адреса буде коректно працювати тільки у випадку, якщо в директорії де знаходиться файл, у якому ми це робимо посилання, існує директорія [pages](#), а в ній документ [about.html](#).

Наприклад, якщо ми знаходимося на сторінці [/pages/index.html](#), то роблячи посилання [pages/about.html](#), воно буде вести на [/pages/pages/about.html](#)

## **about.html**

Така адреса буде коректно працювати тільки у випадку, якщо файл, у якому ми робимо посилання, знаходиться в тій же директорії, що і файл [about.html](#).

## **../index.html**

Конструкція [../](#) означає піднятися на один каталог вгору і вже щодо нього відкривати документ.

Наприклад, якщо ми знаходимося в файлі [/category1/subcategory1/index.html](#), то посилання [../](#) буде ввести на [/category1/index.html](#)

*Примітка:* можна скільки завгодно раз вказувати конструкцію [../](#) - кількість впливає на те, на скільки каталогів вище ми піднімаємося.

# Зображення

Елемент `<img>` вбудовує зображення в документі.

```

```

Атрибут `src` обов'язковий і містить шлях до зображення, яке ви плануєте включити в документ.

Бажано використовувати як в ім'ях файлів, так і у папках тільки латинські букви нижнього регістру (a-z), цифри (0-9), крапку (.), мінус (-) та нижнє підкреслення (\_).

Атрибут `alt` містить текстовий опис зображення, яке не обов'язково, але корисно для доступності.

Наприклад програми читання з екрану читають цей опис своїм користувачам, також цей текст відобразиться у випадку якщо зображення не може бути завантажено з якоїсь причини. А також цей текст використовують пошукові роботи для індексації зображення та співвідношення йому цього тексту.

Атрибут `width` містить ширину зображення в пікселях. Необов'язковий.

Атрибут `height` містить висоту зображення в пікселях. Необов'язковий.

Інші атрибути `<img>`:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img#attributes>



# Посилання

Посилання (гіперпосилання) - одне з найцікавіших нововведень Інтернет. Вони були особливістю Мережі з самого початку, але саме вони перетворюють Інтернет в Інтернет. Вони дозволяють нам пов'язувати наші документи з будь-яким іншим документом (або ресурсом). За їх допомогою ми також можемо пов'язувати документи з його конкретними частинами.

Просте посилання створюється шляхом обгортання тексту (або іншого вмісту), який ви хочете перетворити в посилання, в елемент `<a>`, і додання цьому елементу обов'язкового атрибуту `href`, який буде містити веб-адресу (а фактично URL), на яку ви хочете зробити посилання.

```
<a href="https://google.com">Google</a>
```

Необов'язкоюю атрибут `title` призначений для відображення корисної інформації про посилання, що з'явиться при наведенні на посилання курсором миші. Також цю інформацію використовують пошукові системи, аналогічно атрибуту `title` для тега `<img>`

```
<a href="https://google.com" title="Google - це така пошукова система">Google</a>
```

Інші атрибути `<a>`:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a#attributes>

# Посилання: Якір

Можна посилатися на певну частину документа HTML (відому як якір або фрагмент документа), а не тільки на верхню частину документа. Для цього вам спочатку потрібно призначити атрибут `id` елемента на який ви би хотіли потім зробити посилання. Зазвичай має сенс посилатися на певний заголовок, тому це виглядає приблизно так:

```
<h2 id="chapter2">Глава 2</h2>
```

Потім, щоб пов'язати з цим `id`, ви повинні включити його в кінець URL-адреси, якому передує знак решітки (`#`), наприклад:

```
<a href="#chapter2">Глава 2</a>
```

Приклад: <https://html.spec.whatwg.org/multipage/>

```
<ul>
  <li><a href="#chapter1">Глава 1</a></li>
  <li><a href="#chapter2">Глава 2</a></li>
  <li><a href="#chapter3">Глава 3</a></li>
</ul>
```

```
<h2 id="chapter1">Глава 1</h2>
<p>
  Тут текст першої глави
</p>
```

```
<h2 id="chapter2">Глава 2</h2>
<p>
  Тут текст другої глави
</p>
```

```
<h2 id="chapter3">Глава 3</h2>
<p>
  Тут текст третьої глави
</p>
```

# Посилання: Електронна пошта

Можна створювати посилання або кнопки, які при натисканні відкривають нове вихідне повідомлення електронної пошти, а не посилання на ресурс або сторінку.

Для цього використовується елемент `<a>` та `mailto:<адреса_електронної_пошти>` у значенні атрибута `href`.

```
<a href="mailto:roman@romantelychko.com">Надіслати лист для Романа</a>
```

Крім адреси електронної пошти, ви можете надати і іншу інформацію.

Фактично, будь-які стандартні поля для відправки пошти можуть бути додані до вказаної адреси `mailto`. Часто використовуваними з них є «`subject`», «`cc`» і «`body`». Кожне поле і його значення задаються в якості умови запиту розділюючи амперсандом (&), згідно формату [URL](#)

```
<a href="mailto:roman@romantelychko.com?subject=Це заголовок листа&body=Це вміст листа">
```

Надіслати лист для Романа з вже заповненою темою і текстом листа

```
</a>
```

# Посилання: Інші види



## Приклад

Телефон. Зателефонувати

```
<a href="tel:+380121234567">098-123-4567</a>
```

Skype. Зателефонувати

```
<a href="skype:username?call">Skype (call)</a>
```

Skype. Чат

```
<a href="skype:username?chat">Skype (chat)</a>
```

WhatsApp. Чат

```
<a href="whatsapp://send?phone="+380121234567">WhatsApp (chat)</a>  
<a href="https://wa.me/380121234567">WhatsApp (chat)</a>
```

Viber. Чат

```
<a href="viber://chat?number="+380121234567">Viber (chat)</a>
```

Telegram. Чат

```
<a href="tg://resolve?domain=username">Telegram</a>
```

Facebook Messenger. Чат

```
<a href="https://www.messenger.com/t/username">Facebook Messenger</a>
```

# Відео

Для вбудовування відео контенту в документ використовуйте елемент HTML `<video>`.

Відео елемент може містити один або кілька джерел відео.

Щоб вказати джерело відео, необхідно використовувати атрибут `src` або елемент `<source>`.

Браузер сам визначить джерело відео на основі форматів відео, що він підтримує.

За допомогою елемента `<track>` можливо вказівку субтитрів у [форматі WebVTT](#).

```
<video width="480" controls poster="/download/movie.gif">
  <source src="/download/movie.mp4" type="video/mp4">
  <source src="/download/movie.ogv" type="video/ogg">
  <source src="/download/movie.webm" type="video/webm">
  <track kind="subtitles" src="movie_en.vtt" srclang="en">
  <track kind="subtitles" src="movie_uk.vtt" srclang="uk">
```

Ваш браузер не підтримує HTML5 video.

```
</video>
```

Інші атрибути `<video>`:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/video#attributes>

# Аудіо



Елемент `<audio>` працює точно так само, як елемент `<video>`, з кількома невеликими відмінностями.

Елемент `<audio>` не підтримує атрибути `width/height` - оскільки привласнювати ширину або висоту ні до чого. Він також не підтримує атрибут `poster`.

```
<audio controls>
  <source src="viper.mp3" type="audio/mp3">
  <source src="viper.ogg" type="audio/ogg">
  Ваш браузер не підтримує HTML5 audio.
</audio>
```

Інші атрибути `<audio>`:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio#attributes>

# <iframe>



Елемент `<iframe>` (від англ. *inline frame* - вбудований фрейм) створює вбудований фрейм, який знаходиться всередині звичайного документа.

Елемент `<iframe>` дозволяє завантажувати в область заданих розмірів будь-які інші незалежні документи (інші сторінки). Він може інтегрувати контент в будь-якому місці на вашій сторінці, без необхідності включати їх в структуру веб-макету, як традиційний елемент.

Ви не повинні використовувати `<iframe>` надмірно, це може уповільнити роботу вашої сторінки і створити загрозу безпеці, особливо якщо ви використовуєте контент з підозрілого веб-сайту.

Розглядайте `<iframe>` як частина вашого контенту, але не як частина вашого сайту. Наприклад, якщо ви хочете додати відео з YouTube.

Детальніше:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe>

<https://habr.com/ru/post/488516/>

## Завдання #2.3

1. Додайте на сторінку [index.html](#) з попереднього [завдання #2.2](#) після елемента `<h6>` будь-яке зображення. Зображення необхідно покласти у директорію `images`.
2. Елемент з зображенням повинен мати наступні атрибути: `src`, `width`, `height` та `alt`.
3. Додайте інше зображення в будь-який існуючий елемент `<p>` всередину тексту. Зверніть увагу що трапиться з текстом. Чому?
4. В кінець елемента `<main>` додайте окремий `<div>`, а всередину нього додайте будь-яке відео. Відео необхідно покласти у директорію `videos`. Елемент `<video>` має містити зображення-poster.

*Підказка:*

*Можна швидко сконвертувати у будь-який формат відео з YouTube/Facebook/Instagram/.. за допомогою сервіса <https://savefrom.net>, просто вставивши посилання на відео.*

5. Перевірте цю сторінку на помилки за допомогою HTML-валідатора [validator.w3.org](https://validator.w3.org)



## Завдання #2.4

1. Створіть другу сторінку з базовою структурою за прикладом під назвою `contacts.html`, поряд з файлом `index.html` з завдання #2.2
2. Додайте на обидві сторінки в блок `<nav>` нумерований список з посиланнями одна на одну, що буде використовуватись у якості меню сайту. Наприклад:

### Назва сайту

- Перша сторінка
- Контакти

### Мій головний заголовок

3. Додайте на сторінку `contacts.html` у елемент `<main>` будь-яке відео з YouTube за допомогою кнопки "Поділитися" (Share) -> "Вставити" (Embed)
4. Перевірте цю сторінку на помилки за допомогою HTML-валідатора [validator.w3.org](https://validator.w3.org) (Помилки для атрибутів `iframe` можна ігнорувати)

# Структуровані дані

Мета використання структурованих даних - зробити інтернет зрозумілішим, структурованим і полегшити пошуковим системам і спеціальним програмам обробку інформації для зручного її відображення в результатах пошуку.

Пошукові системи підтримують структуровані дані у трьох форматах:

- [JSON-LD](#) (рекомендований)  
Анотація JavaScript, вбудована в тег <script> у заголовку сторінки або в тілі. Розмітка не чергується з видимим для користувача текстом.
- [Microdata](#) (мікродані)  
Використовується для вкладання структурованих даних у вміст HTML. Використовує атрибути HTML-тегів для іменування властивостей, які ви хочете представити як структуровані дані. Зазвичай він використовується в тілі сторінки.
- [RDFa](#)  
Розширення HTML5, яке підтримує пов'язані дані, використовуючи [атрибути тегів HTML](#), які відповідають видимому користувачеві вмісту, який ви хочете описати для пошукових систем. RDFa зазвичай використовується як в заголовку, так і в тілі HTML-сторінки.

# Структуровані дані: Schema.org



[Schema.org](https://schema.org) - це спільна ініціатива по розробці єдиної схеми для семантичної розмітки в HTML5.

Ініціативу було запущено 2 червня 2011 року творцями найбільших пошукових систем - компаніями Google, Yahoo! і Microsoft.

Основною метою schema.org є допомога веб-розробникам у створенні якісних метаданих, що, в свою чергу, дозволяє поліпшувати якість пошуку. Метадані на сайтах, що використовують схеми, описані на schema.org, можуть бути безпосередньо проаналізовані пошуковими роботами, допомагаючи останнім краще «розуміти» вміст веб-ресурсів.

В якості основного формату розмітки веб-сторінки метаданими розробники schema.org пропонують microdata (мікродані) - теги і атрибути для розмітки структурованої інформації на веб-сторінках, що з'явилися в стандарті HTML5.

Детальніше:

<https://schema.org/>

<https://ru.wikipedia.org/wiki/Schema.org>

<https://schema.org/docs/gs.html>

# Структуровані дані: Open Graph



Протокол [Open Graph](https://ogp.me/) розроблений соціальною мережею Facebook.

Він дозволяє контролювати превью, яке формується при публікації посилання на сайт в соціальних мережах, і передавати інформацію іншим інтернет-сервісам.

Розмітка Open Graph підвищує привабливість вашого контенту в соцмережах, а це означає - це один із пунктів в рамках стратегій SMM-просування та контент-маркетингу.

Детальніше:

<https://ogp.me/>

<https://developers.facebook.com/docs/sharing/webmasters?locale=ru> RU

# Структуровані дані: Мікроформати



Мікроформати - це формати семантичної розмітки HTML-сторінок, що дозволяють зробити контент доступним для обробки роботами. Мікроформати дають можливість явно вказати смислове значення окремих блоків тексту, доповнивши існуючу HTML-розмітку спеціальними блоками.

Мікроформати використовуються для розмітки інформації про людей ([h-card](#)), організацій (також [h-card](#)), подій ([h-event](#)), місць ([h-adr](#)), публікацій у блогах ([h-entry](#)), продуктів ([h-product](#)), оглядів ([h-review](#)), резюме ([h-resume](#)), рецептів ([h-recipe](#)), географічних координат ([h-geo](#)) тощо.

Мікроформати є відкритим стандартом, який використовується різними сервісами в усьому світі.

Детальніше:

<https://microformats.org/wiki/microformats2>

# Структуровані дані: JSON-LD+Schema.org: Приклад

Приклад використання структурованих даних для новини з використанням формату JSON-LD із структурованими даними [Article](#) (тип Schema.org).

Результат перевірки інструментом для тестування структурованих даних

<https://search.google.com/test/rich-results>

Перевірено: 14 квіт. 2021 р., 17:40

На сторінці відображаються розширені результати

Усі структуровані дані на сторінці можуть генерувати розширені результати.

ПЕРЕГЛЯНУТИ ВІДБРАЖУВАНИЙ HTML-КОД

Виявлені елементи

Статті

The Friends Reunion Is Coming in 2021

^

|               |                                       |
|---------------|---------------------------------------|
| type          | NewsArticle                           |
| headline      | The Friends Reunion Is Coming in 2021 |
| image         | https://i.imgur.com/jib7CY1.jpg       |
| datePublished | 2021-04-12T08:00:00+03:00             |
| dateModified  | 2021-04-12T09:20:00+03:00             |

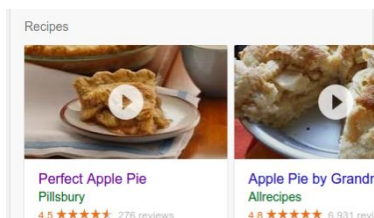
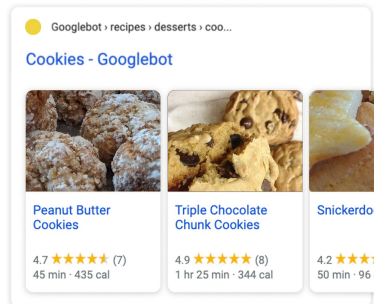
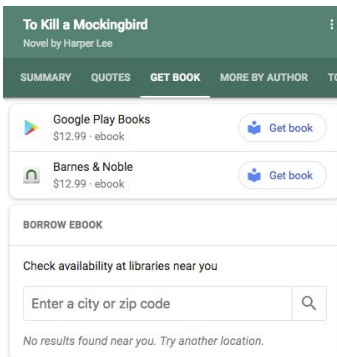
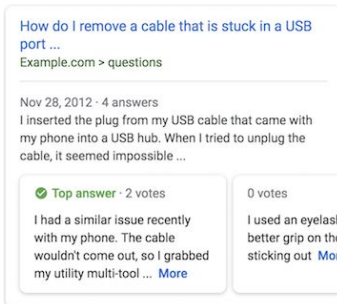
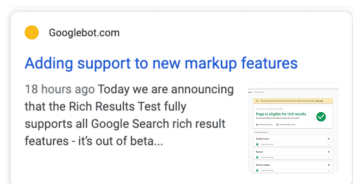
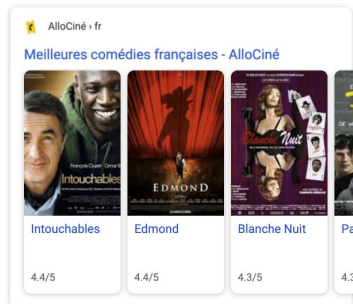
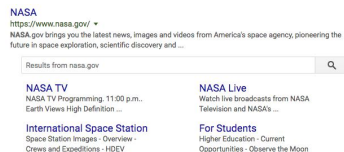
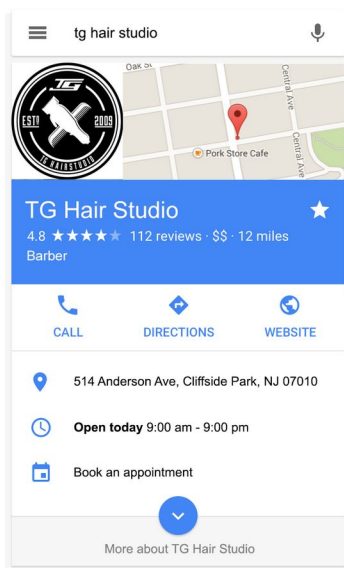
```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="utf-8">
  <title>Структуровані дані: JSON-LD: Приклад</title>
  <script type="application/ld+json">
  {
    "@context": "https://schema.org",
    "@type": "NewsArticle",
    "headline": "The Friends Reunion Is Coming in 2021",
    "image": [
      "https://i.imgur.com/jib7CY1.jpg"
    ],
    "datePublished": "2021-04-12T08:00:00+03:00",
    "dateModified": "2021-04-12T09:20:00+03:00"
  }
</script>
</head>
<body>

</body>
</html>
```

[Переглянути у JSFiddle](#)

# Структуровані дані: Приклади результатів

Приклади відображення структурованих даних на сторінці результатів Google відносно типу <https://developers.google.com/search/docs/guides/search-gallery>



## Завдання #2.5

1. Додайте на сторінку [index.html](#) у елемент `<head>` інформацію про завантажене відео у форматі JSON-LD типу [VideoObject](#), використовуючи за основу наступний приклад, що праворуч.
2. Перевірте сторінку на помилки за допомогою HTML-валідатора [validator.w3.org](#)
3. Перевірте структуровані дані на сторінці за допомогою інструменту для тестування структурованих даних <https://search.google.com/test/rich-results>

```
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "VideoObject",
  "name": "Introducing the self-driving bicycle in the
Netherlands",
  "description": "This spring, Google is introducing the
self-driving bicycle in Amsterdam, the world's premier cycling
city.",
  "thumbnailUrl": [
    "https://example.com/photos/1x1/photo.jpg",
    "https://example.com/photos/4x3/photo.jpg",
    "https://example.com/photos/16x9/photo.jpg"
  ],
  "uploadDate": "2016-03-31T08:00:00+08:00",
  "duration": "PT1M54S",
  "contentUrl": "https://www.example.com/video/123/file.mp4"
}
</script>
```



# Таблиці

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

Вміст будь-якої таблиці вкладається між двома тегами: `<table>` та `</table>`.

Елемент `<tr>` ('tr' - скорочення від 'table row') необхідно використовувати для вказання ряду таблиці.

Найменшим контейнером в таблиці є комірка ('cell'), вона створюється елементом `<td>` ('td' - скорочення від 'table data').

Комірка 1-1	Комірка 1-2	Комірка 1-3
Комірка 2-1	Комірка 2-2	Комірка 2-3

# Таблиці: Заголовки

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

В якості заголовків, візуально і семантично, можна використовувати елемент `<th>` ('th' скорочення від 'table header').

Він працює в точності як `<td>`, за винятком того, що позначає заголовок, а не звичайну комірку та виділяється за замовчуванням жирним шрифтом.

Заголовок 1	Заголовок 2	Заголовок 3
Комірка 1-1	Комірка 1-2	Комірка 1-3
Комірка 2-1	Комірка 2-2	Комірка 2-3

# Таблиці: Об'єднання кількох рядків або стовпців

```
<table>
  <tr>
    <th colspan="2">Заголовок 1+2</th>
    <th>Заголовок 3</th>
  </tr>
  <tr>
    <td>Комірка 1-1</td>
    <td>Комірка 1-2</td>
    <td rowspan="2">Комірка 1-3 + 2-3</td>
  </tr>
  <tr>
    <td>Комірка 2-1</td>
    <td>Комірка 2-2</td>
  </tr>
</table>
```

Табличні заголовки і комірки мають атрибути **colspan** і **rowspan**, які дозволяють об'єднати декілька рядків та/або стовпців.

Обидва атрибути беруть безрозмірне числове значення, яке дорівнює кількості рядків або стовпців, на які повинні поширюватися об'єднання.

Наприклад, **colspan="2"** поширює комірку на два стовпці, а **rowspan="3"** поширює комірку на 3 рядки.

Заголовок 1+2		Заголовок 3
Комірка 1-1	Комірка 1-2	Комірка 1-3 + 2-3
Комірка 2-1	Комірка 2-2	

## Завдання #2.6

Створіть таблицю, як на картинці.

*Примітка:*

Для зручності візуалізації, додайте в елемент `<table>` значення атрибута `border="1"`, бо таблиця за замовчуванням не відображає рамки.

Один			Два
Три	Чотири	П'ять	
Шість	Сім		Вісім
Дев'ять			

# Форми



Веб-форми є одним з основних елементів взаємодії між користувачем і сайтом або додатком.

Форми дозволяють користувачеві ввести дані, які потім відправляються на сервер для їх подальшої обробки та зберігання або використовуються на стороні клієнта для поновлення інтерфейсу (наприклад, додавання нового елемента в перелік або відкриття і закриття елемента інтерфейсу).

Веб-форми (або HTML-форми) складаються з одного або декількох елементів управління форм і деяких додаткових елементів для структурування форми.

Елементами управління можуть бути однорядкові або багаторядкові текстові поля (textarea), випадаючі списки (select), кнопки (button), прапорці/пташки (checkbox), перемикачі (radio) та інші.

Більшість з них створюються через HTML-елемент `<input>`.

В елементах управління форм можна задати правила, що вказують на певний формат даних або значень, які можуть бути введені (валідація форм), а також до них можуть бути додані текстові рядки, що описують ці елементи для зрячих і незрячих користувачів.

# Форми

Створення форми починається з елемента `<form>`:

```
<form action="/page-to-anywhere" method="get">
```

```
</form>
```

Цей елемент формально визначає форму. Він є елементом-контейнером, як HTML-елементи `<div>` або `<p>`, але при цьому він підтримує деякі специфічні атрибути для налаштування поведінки форми.

Всі атрибути є опціональними, але в стандартній практиці прийнято вказувати атрибути `action` та `method`:

Атрибут `action` визначає адресу, куди повинні бути надіслані дані після відправки форми. У випадку встановлення порожнього значення або не встановлення атрибуту взагалі - дані будуть надіслані на поточну сторінку з її перезавантаженням.

Атрибут `method` вказує, який HTTP-метод буде використано при передачі даних. Для HTML-форм це може бути або `"get"` або `"post"`.

# Форми

```
<form action="/feedback" method="post">
  <ul>
    <li>
      <label for="name">Ім'я:</label>
      <input type="text" id="name" name="username">
    </li>
    <li>
      <label for="mail">Email:</label>
      <input type="email" id="mail"
name="usermail">
    </li>
    <li>
      <label for="message">Повідомлення:</label>
      <textarea id="message"
name="usermessage"></textarea>
    </li>
    <li>
      <button type="submit">Надіслати</button>
    </li>
  </ul>
</form>
```

Тут елементи `<li>` використовуються для структурування коду і полегшення стилізації.

Для доступності і зручності використання ми вказали певний текст-підказку для кожного елемента управління. Зверніть увагу на використання атрибута `for` на кожному елементі `<label>`, який приймає в якості значення `id` елемента управління форми, з яким він пов'язаний - цей підхід дозволяє прив'язати тексти-підказки `<label>` до елементів управління форми (у прикладі це два `<input>` та `<textarea>`).

# Форми

```
<form action="/feedback" method="post">
  <ul>
    <li>
      <label for="name">Ім'я:</label>
      <input type="text" id="name" name="username">
    </li>
    <li>
      <label for="mail">Email:</label>
      <input type="email" id="mail"
name="usermail">
    </li>
    <li>
      <label for="message">Повідомлення:</label>
      <textarea id="message"
name="usermessage"></textarea>
    </li>
    <li>
      <button type="submit">Надіслати</button>
    </li>
  </ul>
</form>
```

Приклад контактної форми, що буде складатися з

- Поля для введення імені - `<input>`
- Поля для введення email - `<input>` з типом email: однорядкове текстове поле, яке приймає тільки email адреси.
- Поле для введення повідомлення - `<textarea>` - багаторядкове текстове поле.



# Форми

```
<form action="/feedback" method="post">
  <ul>
    <li>
      <label for="name">Ім'я:</label>
      <input type="text" id="name" name="username">
    </li>
    <li>
      <label for="mail">Email:</label>
      <input type="email" id="mail"
name="usermail">
    </li>
    <li>
      <label for="message">Повідомлення:</label>
      <textarea id="message"
name="usermessage"></textarea>
    </li>
    <li>
      <button type="submit">Надіслати</button>
    </li>
  </ul>
</form>
```

У HTML-елементі `<input>` найважливішим атрибутом є атрибут `type`, тому що він визначає зовнішній вигляд і поведінку елемента `<input>`.

Елемент `<textarea>` представляє собою багаторядковий елемент управління редагуванням тексту.

Використовується якщо ви хочете дозволити користувачам вводити значний обсяг тексту довільної форми, наприклад, коментар або форму зворотного зв'язку.

# Форми

```
<form action="/feedback" method="post">
  <ul>
    <li>
      <label for="name">Ім'я:</label>
      <input type="text" id="name" name="username">
    </li>
    <li>
      <label for="mail">Email:</label>
      <input type="email" id="mail"
name="usermail">
    </li>
    <li>
      <label for="message">Повідомлення:</label>
      <textarea id="message"
name="usermessage"></textarea>
    </li>
    <li>
      <button type="submit">Надіслати</button>
    </li>
  </ul>
</form>
```

Елемент `<button>` створює клікабельну кнопку, яка може бути використана в формах або в будь-якому іншому місці документа, який вимагає простої стандартної кнопки.

HTML-елемент `<button>` приймає атрибут `type`, який може дорівнювати одному з трьох значень: `submit`, `reset` або `button`.

- Клік по кнопці `submit` відправляє дані з форми на сторінку, визначену в атрибуті `action` елемента `<form>`.
- Клік по кнопці `reset` скидає значення всіх елементів управління форми до їх початкового значення.
- Клік по кнопці `button` не робить нічого ([детальніше тут](#))

## Завдання #2.7

1. Додайте на сторінку `contacts.html` в елемент `<main>` цю форму
2. Перевірте цю сторінку на помилки за допомогою HTML-валідатора [validator.w3.org](https://validator.w3.org)

```
<form action="" method="get">
  <ul>
    <li>
      <label for="name">Ім'я:</label>
      <input type="text" id="name" name="username">
    </li>
    <li>
      <label for="mail">Email:</label>
      <input type="email" id="mail" name="usermail">
    </li>
    <li>
      <label for="message">Повідомлення:</label>
      <textarea id="message" name="message"></textarea>
    </li>
    <li>
      <button type="submit">Надіслати</button>
    </li>
  </ul>
</form>
```

# Елементи форм: Поля введення тексту

Текстові поля `<input>` є найбільш базовими елементами форм. Ці поля найбільш зручні для користувача введення різної інформації.

Всі текстові поля мають загальні атрибути:

- **readonly** - (булевий) користувач не може змінювати початкове значення (значення за замовчуванням)
- **disabled** - (булевий) значення ніколи не надсилається разом з іншими даними форм
- **placeholder** - текст, який з'являється всередині поля форми і коротко описує, для чого використовується дане поле. Може як конкретизувати `<label>`, так і повністю його замінювати.
- **value** - значення поля по-замовчуванню, яке може змінити користувач. При відсутності дорівнює ""
- **name** - назва поля, що використовується при відправці на сервер. Бажано щоб кожний необхідний для надсилання елемент мав назву, бо значення елемента без вказаного атрибута **name** не надішлеться.

Всі доступні атрибути:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#attributes>

# Елементи форм: Поля введення тексту

## Однорядкові текстові поля (type=text)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `text`.

```
<input type="text" value="Я текстове поле">
```

## Поле введення пароля (type=password)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `password`.

Значення пароля не додає особливих обмежень до введенного тексту, але затінює значення, введене в поле (наприклад, точками або зірочками), тому інші користувачі не можуть його прочитати.

```
<input type="password">
```

# Елементи форм: Поля введення тексту

## Поле введення email (type=email)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `email`.

Користувач повинен ввести коректну адресу електронної пошти в поле (пошту в коректному форматі). Будь-який інший контент змушує браузер відображати помилку при відправці форми.

```
<input type="email">
```

## Поле введення рядка пошуку (type=search)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `search`.

Основна відмінність між текстовим полем і полем пошуку полягає в тому, як його стилізує браузер - часто поля пошуку відображаються з заокругленими кутами з відображенням кнопки «X» для скидання введеного значення в кінці поля.

```
<input type="search">
```

# Елементи форм: Поля введення тексту

## Поле введення телефону (type=tel)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `tel`.

Залежно від телефону (особливо на мобільних пристроях) може бути представлена інша віртуальна клавіатура, яка більше підходить для введення телефонного номера.

```
<input type="tel">
```

## Поле введення URL-адреси (type=url)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `url`.

Це додає спеціальні обмеження перевірки до поля, з браузером, який повідомляє про помилку, якщо введені неправильні URL.

```
<input type="url">
```

# Елементи форм: Багаторядкові текстові поля

Багаторядкове текстове поле вказується з використанням елемента `<textarea>`, а не за допомогою елемента `<input>`.

`<textarea>`Тут текст за замовчуванням`</textarea>`

Основна відмінність між багаторядковим текстовим полем і звичайним однорядковим текстовим полем складається в тому, що користувачам дозволяється набирати текст, що містить жорсткі розриви рядків (тобто натискання клавіші Enter).

Також присутні додаткові атрибути кількості рядків `rows` та кількості стовпців `cols`.

У деяких браузерях розмір цього поля можна змінити потягнувши за нижній правий кут.

```
<textarea rows="5" cols="10">  
    Тут текст за замовчуванням  
    Тут текст за замовчуванням  
</textarea>
```

```
Тут текст за замовчуванням  
Тут текст за замовчуванням  
Тут текст за замовчуванням  
Тут текст за замовчуванням
```



# Елементи форм: Випадаючі елементи

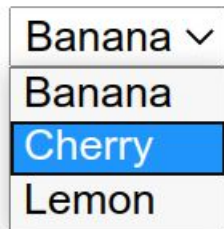
## Поле вибору (<select>)

Створюється елементом <select> з одним або декількома елементами <option> в якості дочірніх елементів, кожен з яких вказує одне з можливих значень.

Якщо елемент <option> встановлений з атрибутом value, значення цього атрибута відправляється при відправці форми.

Булевий атрибут selected вказує на поточне значення цього поля по-замовчуванню, яке користувач може змінити при виборі іншого пункта. У випадку якщо selected не вказано - автоматично буде обрано перший елемент випадаючого списку.

```
<select>
  <option value="banana">Banana</option>
  <option value="cherry" selected>Cherry</option>
  <option value="lemon">Lemon</option>
</select>
```



Banana
Banana
Cherry
Lemon

# Елементи форм: Випадаючі елементи

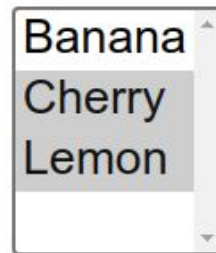
## Поле множинного вибору (<select multiple>)

За замовчуванням поле вибору дозволяє користувачеві вибрати тільки одне значення.

Додавши атрибут **multiple** до елемента `<select>`, ви можете дозволити користувачам обирати декілька значень, використовуючи механізм за замовчуванням, що надається операційною системою.

Але у такому разі поле більше не буде відображати значення у вигляді випадаючого списку (dropdown select) - замість цього всі значення будуть показані одразу у вигляді списку.

```
<select multiple>
  <option value="banana">Banana</option>
  <option value="cherry">Cherry</option>
  <option value="lemon">Lemon</option>
</select>
```



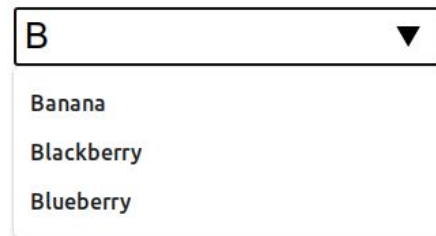
# Елементи форм: Випадаючі елементи

## Поле автодоповнення (autocomplete) (<input type=text> + <datalist>)

Ви можете надати список автоматично заповнених можливих значень елемента форми <input> (або іншого), використовуючи елемент <datalist> з дочірніми елементами <option> у якості значень.

Такий список з даними необхідно прив'язати до текстового поля (зазвичай до елемента <input>) за допомогою атрибута list (id елемента <datalist> має співпадати з вказаним значенням атрибута list).

```
<input type="text" list="suggestions">
<datalist id="suggestions">
  <option>Apple</option>
  <option>Banana</option>
  <option>Blackberry</option>
  <option>Blueberry</option>
  <option>Lemon</option>
  <option>Lychee</option>
  <option>Peach</option>
  <option>Pear</option>
</datalist>
```



B ▼
Banana
Blackberry
Blueberry

# Елементи форм: Check-елементи



Check-елементи - це елементи, стан яких ви можете змінити, натиснувши на них.

Існує два види check-елементів:

- **прапорець/пташка** (checkbox)
- **перемикач** (radio)

Обидва використовують атрибут **checked**, щоб вказати встановлений елемент чи ні.

Варто відзначити, що ці елементи не поведуться так само, як інші елементи форм.

Для більшості елементів форми після відправки форми відправляються всі елементи, що мають атрибут **name**, навіть якщо значення не було заповнене.

У разі check-елементів їх значення відправляються, тільки якщо вони встановлені активними. Якщо вони не встановлені активними, нічого не відправляється, навіть їх ім'я.

# Елементи форм: Check-елементи

## Поле прапорець/пташка/checkbox (type=checkbox)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `checkbox`.

```
<input type="checkbox" value="carrots" checked>
```



# Елементи форм: Check-елементи

## Поле перемикач/radio (type=radio)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `radio`.

```
<input type="radio" checked>
```

Кілька перемикачів можуть бути пов'язані один з одним.

Якщо вони мають однакове значення для свого атрибута `name`, вони будуть вважатися що знаходяться в тій же групі. Тільки один перемикач в даній групі може бути активним одночасно; це означає, що при установці активним одного з них всі інші автоматично відключаються.

При відправці форми відправляється значення тільки активного перемикача.

```
<input type="radio" name="meal" value="soup" checked>  
<input type="radio" name="meal" value="curry">  
<input type="radio" name="meal" value="pizza">
```



# Елементи форм: Кнопки

У HTML-формах є три види кнопок:

- **submit (type=submit)** - відправляє дані форми на сервер.
- **reset (type=reset)** - скидає всі елементи форми до значень за замовчуванням.
- **button (type=button)** - анонімні кнопки, які не мають автоматичного ефекту, але можуть бути налаштовані з використанням коду JavaScript.

Якщо ви не вкажете атрибут **type**, це буде значення за замовчуванням.

Кнопка створюється з використанням елемента `<button>` або елемента `<input>`.

Значення атрибута **type** визначає який вид кнопки відображається.

```
<button type="submit">Кнопка submit</button>  
<input type="submit" value="Це кнопка submit">
```

```
<button type="reset">Кнопка reset</button>  
<input type="reset" value="Це кнопка reset">
```

```
<button type="button">Анонімна кнопка</button>  
<input type="button" value="Це анонімна кнопка">
```

Кнопка submit

Це кнопка submit

Кнопка reset

Це кнопка reset

Анонімна кнопка

Це анонімна кнопка

# Елементи форм: Розширені елементи

## Поле введення чисел (number) (type=number)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `number`.

Цей елемент управління виглядає як текстове поле, але допускає тільки числа і зазвичай надає кілька кнопок для збільшення або зменшення значення (на значення `step`).

```
<input type="number" min="1" max="10" step="1">
```



## Поле введення чисел з діапазону (range) (type=range)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `range`.

Важливо правильно налаштувати слайдер; для цього рекомендується встановити атрибути `min`, `max` і `step`.

```
<input type="range" min="0" max="500" step="10">
```





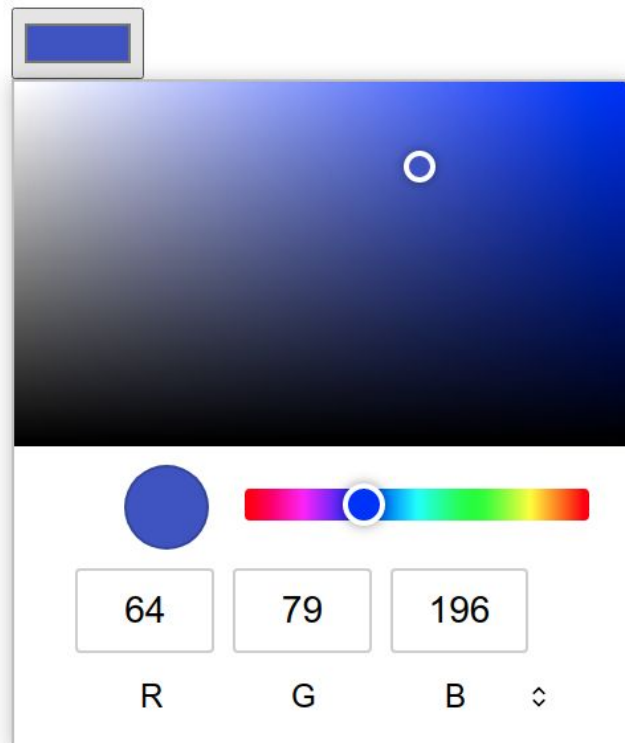
# Елементи форм: Розширені елементи

## Поле введення кольору (type=color)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `color`.

Елемент дозволяє користувачам обирати колір як текстовим (у форматах RGB, HSL, HEX), так і візуальним/інтерактивним способами.

```
<input type="color">
```



# Елементи форм: Розширені елементи

## Елемент вибору файлу (type=file)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `file`.

За допомогою цього елемента користувач може вибрати один або кілька файлів для відправки.

Типи файлів, які приймаються, можуть бути обмежені за допомогою атрибута `accept`.

Крім того, якщо ви хочете, щоб користувач міг вибрати більше одного файлу, ви можете зробити це, додавши атрибут `multiple`.

```
<input type="file" accept="image/*" multiple>
```

Вибрати файли    Файл не вибрано

# Елементи форм: Розширені елементи

## Поле прихованого контенту (hidden) (type=hidden)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `hidden`.

Використовується для створення елемента управління форми, який невидимий для користувача, але відправляється на сервер разом з іншими даними форми.

```
<input type="hidden" value="1593945719">
```

## Поле зображення (type=image)

Створюються з використанням елемента `<input>` значення `type` якого встановлено в `image`.

Елемент управління, який відображається в точності як елемент `<img>`, за винятком того, що коли користувач натискає на нього, він веде себе як кнопка відправки.

```
<input type="image" alt="Натисни на мене!" src="images/image.png" width="80" height="30">
```

Зробити все

# Елементи форм: Розширені елементи

## Індикатор ходу виконання (<progress>)

Елемент `<progress>` відображає індикатор, який показує хід виконання завдання, зазвичай відображається у вигляді прогрес бару (індикатора виконання).

Вміст всередині елемента `<progress>` є запасним варіантом для браузерів, які не підтримують цей елемент.

```
<progress max="100" value="75">75%</progress>
```



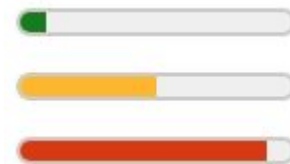
# Елементи форм: Розширені елементи

## Індикатор шкали (<meter>)

Елемент <meter> представляє фіксоване значення в діапазоні, обмеженому мінімальним і максимальним значенням.

Колір залежить від значення optimum:

- червоний -  $\min \leq \text{optimum} < \text{low}$
- жовтий -  $\text{low} \leq \text{optimum} \leq \text{high}$
- зелений -  $\text{high} < \text{optimum} \leq \text{max}$



```
<meter min="0" max="100" value="10" low="20" high="80" optimum="10">10%</meter>
```

```
<meter min="0" max="100" value="50" low="20" high="80" optimum="10">50%</meter>
```

```
<meter min="0" max="100" value="90" low="20" high="80" optimum="10">90%</meter>
```

# Елементи форм: Дата/час

## Поле введення дати/часу/дати і часу (type=date)

Створюються з використанням елемента `<input>` і відповідного значення для атрибута `type` в залежності від того, чи хочете ви збирати дати, час або і те, і інше.

Всі елементи дати і часу можуть бути обмежені діапазоном (датами) за допомогою атрибутів `min` і `max`.

```
<input type="date" name="date" min="2020-07-01"
max="2020-07-20" id="date">
```

15.07.2020

Липень 2020

П	В	С	Ч	П	С	Н
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Сьогодні

# Елементи форм: Дата/час

## type=date

Це створює елемент для відображення і вибору дати (рік, місяць і день, без часу).

```
<input type="date" name="date" id="date">
```

The screenshot shows a date picker for July 2020. The header displays '11.07.2020' and a calendar icon. Below the header, the month 'Липень 2020' is shown with up and down arrows. The calendar grid has days of the week (П, В, С, Ч, П, С, Н) and dates. The date '11' is highlighted in blue, and '15' is outlined. At the bottom right, there is a link 'Сьогодні'.

## type=datetime-local

Це створює елемент для відображення і вибору дати з часом.

```
<input type="datetime-local" name="datetimelocal" id="datetimelocal">
```

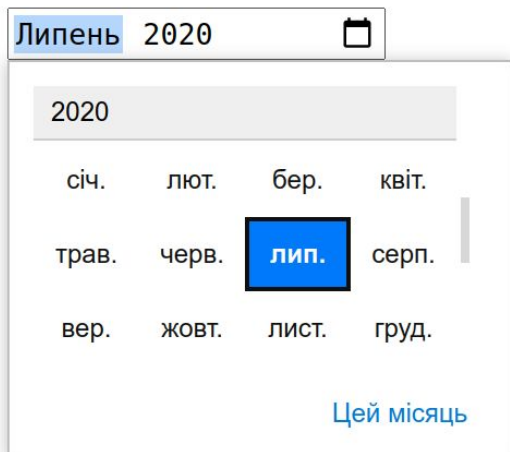
The screenshot shows a datetime-local picker for July 2020. The header displays '11.07.2020, 09:00' and a calendar icon. Below the header, the month 'Липень 2020' is shown with up and down arrows. The calendar grid is similar to the one in the first image, with '11' highlighted and '15' outlined. To the right of the calendar is a time selection area with two columns of buttons: '09' and '00' at the top, followed by '10', '11', '12', '13', '14', and '15' in the first column, and '01', '02', '03', '04', '05', and '06' in the second column. At the bottom right, there is a link 'Сьогодні'.

# Елементи форм: Дата/час

## type=month

Це створює елемент для відображення і вибору місяця з роком без часового поясу.

```
<input type="month" name="month" id="month">
```

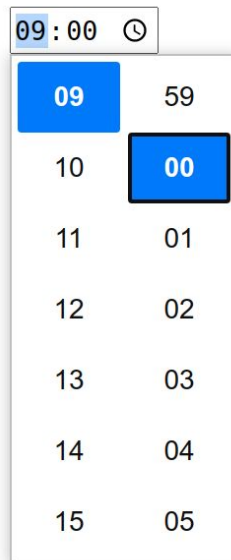


The image shows a date picker interface. At the top, it displays 'Липень 2020' (July 2020) with a calendar icon to the right. Below this is a dropdown menu showing '2020'. The main area is a grid of month abbreviations in Ukrainian: січ., лют., бер., квіт., трав., черв., **лип.** (highlighted in blue), серп., вер., жовт., лист., груд. At the bottom right, there is a link that says 'Цей місяць' (This month).

## type=time

Це створює елемент для відображення і вибору значення часу без часового поясу.

```
<input type="time" name="time" id="time">
```



The image shows a time picker interface. At the top, it displays '09:00' with a clock icon to the right. Below this is a grid of time values. The first column shows hours from 09 to 15, and the second column shows minutes from 59 down to 05. The '09' in the first column and '00' in the second column are highlighted in blue.




# Елементи форм: Дата/час

## type=week

Це створює елемент для відображення і вибору номера тижня і року без часового поясу.

```
<input type="week" name="week" id="week">
```

Тиждень 28, 2020 р. 

Липень 2020 ▾

↑ ↓

Тиждень	П	В	С	Ч	П	С	Н
27	29	30	1	2	3	4	5
28	6	7	8	9	10	11	12
29	13	14	15	16	17	18	19
30	20	21	22	23	24	25	26
31	27	28	29	30	31	1	2
32	3	4	5	6	7	8	9

Цей тиждень

## Завдання #2.8

1. Створіть файл `booking.html` поряд з файлами `index.html` та `contacts.html` з попередніх завдань.  
Додайте посилання на нього у елемент `<nav>` для всіх інших сторінок.
2. В елемент `<main>` додайте форму за прикладом, що зображено праворуч.  
Повний перелік полів [на наступному слайді](#).
3. Перевірте цю сторінку на помилки за допомогою HTML-валідатора [validator.w3.org](https://validator.w3.org)
4. Перевірте зовнішній вигляд під різними браузерами (Chrome, Firefox, Edge, Safari, ...)

### Бронювання залу фотостудії

\* Прізвище:

\* Ім'я:

\* Email:

\* Телефон:

Профіль Instagram:

\* Зал:

\* Дата і час початку:

\* Дата і час кінця:

\* Кількість людей в залі:

Колір фона:



Коментар:

☐ Я погоджуюсь з [правилами фотостудії](#)

## Завдання #2.8 (Додатково)



Поля форми:

- **Прізвище:** обов'язкове, текстове, від 2 до 64 символів довжиною
- **Ім'я:** обов'язкове, текстове, від 2 до 64 символів довжиною
- **Email:** обов'язкове, електронна пошта, від 6 до 255 символів довжиною
- **Телефон:** обов'язкове, телефон, 13 символів довжиною
- **Профіль Instagram:** адреса URL, від 23 до 255 символів довжиною
- **Зал:** обов'язкове, один варіант зі списку ("Зал 1", "Зал 2", "Зал 3")
- **Дата і час початку:** обов'язкове, дата і час, не раніше поточного часу, не старіше 01 січня 2023 00:00
- **Дата і час кінця:** обов'язкове, дата і час, не раніше поточного часу, не старіше 01 січня 2023 00:00
- **Кількість людей в залі:** обов'язкове, число від 1 до 10
- **Колір фона:** поле вибору кольору
- **Коментар:** багаторядкове текстове поле
- **Я погоджуюсь:** обов'язкове, пташка (checkbox)
- Кнопка **Очистити**
- Кнопка **Надіслати**

# HTML: Living Standard



Ця специфікація визначає велику частину веб-платформи з великою кількістю деталей.  
Постійно оновлюється до актуальної версії.

<https://html.spec.whatwg.org/multipage/>

# Шпаргалки по HTML



1. <https://digital.com/tools/html-cheatsheet/rtacademy.net/v3/02.html-cheat-sheet-1.pdf>
2. <rtacademy.net/v3/02.html-cheat-sheet-2.png>
3. <rtacademy.net/v3/02.html-cheat-sheet-3.pdf>