Форми. Текстові поля

Форми є невід'ємною частиною Інтернету, тому що пропонують сайтам метод збору інформації від користувачів і обробки запитів, а також елементи управління практично для будь-якого можливого застосування. За допомогою елементів управління або полів, форми можуть запросити невеликий обсяг інформації – часто це пошуковий запит, ім'я користувача або пароль, або великий обсяг інформації – можливо, дані про посилку, платіжна інформація або пропозиція роботи.

Ми повинні знати, як створювати форми, щоб отримати вхідні дані від користувача. У цьому розділі ми обговоримо, як використовувати HTML для розмітки форми, які елементи використовувати для захоплення різних типів даних. Ми не станемо занадто заглиблюватися в те, як інформація з форми обробляється на стороні сервера. Обробка форм являє собою глибоку тему, поза сферою цього посібника; тут ми будемо дотримуватися тільки створення форм.

Ініціалізація форми

Щоб додати форму на сторінку ми будемо використовувати елемент `<form>` . Даний елемент визначає, де на сторінці з'являться елементи управління. Крім того, елемент `<form>` обгортає всі елементи включені в форму, подібно елементу `<div>`.

```
<form action="/login" method="post">
  ...
</form>
```

До елементу `<form>` може застосовуватися декілька різних атрибутів, найбільш поширеними з яких є `action` і `method`. Атрибут `action` містить URL, на який інформація у формі буде відправлена для обробки сервером. Атрибут `method` є методом HTTP, який повинні використовувати браузери для відправки даних форми. Обидва ці атрибута `<form>` мають відношення до відправки і обробки даних.

Атрибут `method` має два значення – `get` і `post`.

- `get` – є одним з найпоширеніших методів і призначений для отримання необхідної інформації та передачі даних в адресному рядку. Пари «ім'я = значення» приєднуються в цьому випадку до адреси після знаку питання і розділяються між собою амперсандом (символ &). Зручність використання методу `get` полягає в тому, що адресу з усіма параметрами можна використовувати неодноразово, зберігши її, наприклад, в закладки браузера, а також змінювати значення параметрів прямо в адресному рядку. Це значення встановлено для атрибута `method` за замовчуванням.
- `post` – метод, який посилає на сервер дані в запиті браузера. Це дозволяє відправляти більшу кількість даних, ніж за методом `get`, оскільки у нього встановлено обмеження в 4 Кб. Великі обсяги даних використовуються у форумах, поштових службах, заповненні бази даних, при пересиланні файлів тощо.

Текстові поля і текстові області

Коли справа доходить до збору текстової інформації від користувачів, є кілька різних елементів, доступних для отримання даних в формах. Зокрема, для збору даних на основі тексту або рядків застосовуються текстові поля і текстові області. Ці дані можуть включати в себе уривки тексту, паролі, номери телефонів та іншу інформацію.

Текстові поля

Одним з основних елементів, які використовуються для отримання тексту від користувачів, є елемент `<input>`. Даний елемент включає атрибут `type` для визначення, який тип інформації буде отримано в елементі управління. Найбільш популярне значення атрибута `type` – це `text`, який позначає введення одного рядка тексту. Поряд з установкою атрибута `type`, гарною практикою буде також дати елементу `<input>` атрибут `name`. Значення атрибута `name` застосовується в якості імені елементу управління і відправляється разом з вхідними даними на сервер.

Атрибути текстового поля

- `name` – ім'я поля, призначене для того, щоб обробник форми міг його ідентифікувати.
- `disabled` – блокує поле для введення тексту і не відправляє дані на сервер.
- `form` – ідентифікатор форми для зв'язування текстового поля з елементом `<form>`.
- `type` – перелік значень для текстого поля, який наведено далі.
- `maxlength` – встановлює максимальне число символів, яке може бути введено користувачем у текстовому полі. Коли ця кількість досягається при наборі, подальше введення стає неможливим. Якщо даний атрибут опустити, то можна вводити рядок довший ніж саме поле.
- `readonly` – блокує поле для введення тексту.
- `size` – ширина текстового поля, яка визначається числом символів моноширинного шрифту. Зазвичай не вказують, тому що через стилі задати бажані розміри простіше і точніше.
- `value` – початковий текст, який відображається в полі.
- `autocomplete` – введений раніше текст запам'ятовується браузером і підставляється при наступному введенні.
- `autofocus` – поле отримує фокус після завантаження документа.
- `list` – значення атрибута `id` елемента `<datalist>`, для зв'язку з цим елементом.
- `pattern` – шаблон введення тексту. Вказує регулярні вирази, відповідно до яких треба вводитти і відправляти дані форми.
- `required` – вказує, що поле є обов'язковим для заповнення.
- `placeholder` – додає підказку, яка зникає при введенні тексту.
- `dirname` – ім'я змінної, яка відправляється на сервер і автоматично отримує значення `ltr` (для тексту зліва направо) або `rtl` (для тексту справа наліво).

Більшість атрибутів є типовими і застосовуються і до інших елементів форми. Деякі атрибути ми докладніше розглянемо в наступних уроках. Створення текстового поля показано в наступному прикладі.

Приклад: Текстове поле

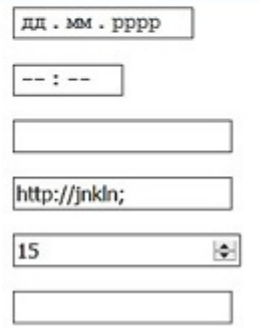
```
<form action="/login" method="post">
  <input type="text" name="username" placeholder="login"
  maxlength="25" size="40" >
</form>
```

Елемент `<input>` є самостійним, тобто він задіє тільки один тег і не обертає ніякого контенту. Значення елемента забезпечується його атрибутами і їх відповідними значеннями. Спочатку було тільки два текстових значення атрибута `type` – `text` і `password` (для введення паролів), проте стандарт HTML5 додав кілька нових значень атрибута `type`.

Ці значення були додані, щоб забезпечити чітке смислове значення для полів вводу, а також надати краще управління користувачам. Якщо браузер не розуміє одне з цих HTML5-значень атрибута `type`, він автоматично повернеться до значення `text`. На наступному рисунку наведено список нових типів HTML5 та декілька прикладів.

- text
- password
- color
- date
- datetime
- email
- month
- number
- range
- search
- tel
- time
- url
- week

```
<input type="date" name="birthday">
<input type="time" name="game-time">
<input type="email" name="email-address">
<input type="url" name="website">
<input type="number" name="cost">
<input type="tel" name="phone-number">
```



Для елемента `<input>` є прості правила, яких слід дотримуватися:

- `<input>` не можна вкладати всередину посилання `<a>`;
- `<input>` не можна вкладати всередину кнопки `<button>`;
- атрибут `list` повинен посилатися на елемент `<datalist>`.

Елемент `<datalist>` дозволяє створити список підказок, який з'являється при наборі тексту. Подібне можна спостерігати в пошуковій системі Google, коли вводиться пошуковий запит.

`<datalist>` пишеться зазвичай за межами форми, кожен пункт при цьому створюється елементом `<option>`, а потім отриманий список зв'язується з текстовим полем за допомогою атрибута `list`, значенням якого виступає значення `id` у `<datalist>` (дивіться наступний приклад).

Приклад: Підказки для текстового поля

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Підказки</title>
```

```
 </head>
 <body>
  <form>
   <p><strong>Уведіть ваше місто</strong></p>
   <p><input name="city" list="city"></p>
  </form>
  <datalist id="city">
    <option>Апостолове</option>
    <option>Верхівцеве</option>
    <option>Верхньодніпровськ</option>
    <option>Вільногірськ</option>
    <option>Дніпро</option>
    <option>Жовті Води</option>
    <option>Зеленодольськ</option>
    <option>Кам'янське</option>
    <option>Кривий Ріг</option>
    <option>Марганець</option>
    <option>Нікополь</option>
    <option>Новомосковськ</option>
    <option>П'ятихатки</option>
    <option>Павлоград</option>
    <option>Перещепине</option>
    <option>Першотравенськ</option>
    <option>Підгородне</option>
    <option>Покров</option>
    <option>Синельникове</option>
    <option>Тернівка</option>
  </datalist>
 </body>
</html>
```

При наборі перших букв показуються відповідні варіанти зі списку. Пункт списку можна вибрати курсором миші або за допомогою стрілок на клавіатурі і він буде доданий в текстове поле. При цьому можна вводити власний текст або редагувати вже доданий.

Текстові області

Ще одним елементом, який використовується для збору текстових даних, є елемент `<textarea>`. Він відрізняється від елемента `<input>` тим, що може приймати великі уривки тексту в декілька рядків. Елемент `<textarea>` також містить відкриваючий і закриваючий теги, які можуть обернути простий текст, який буде відображено у текстовому полі. При цьому у браузері буде відображено усі пробіли та переноси, які є у html-коді. Оскільки `<textarea>` приймає тільки один тип значення, атрибут `type` тут не застосовують, але атрибут `name` використовується як і раніше.

Для `<textarea>` діють ті ж обмеження, що і для `<input>`.

Окрім атрибутів, які `<textarea>` використовує як і елемент `<input>`, можливе використання наступних атрибутів:

- `<rows>` – висота поля в рядках тексту.

- `<wrap>` – значення `soft` передає на сервер текст одним рядком, а hard враховує значення `cols` і автоматично додає переноси.
- `<cols>` – ширина поля в символах.

Як правило, розміри `<textarea>` задаються за допомогою стилів, але якщо одночасно задані атрибути `rows` і `cols` з шириною і висотою через CSS, то стилі мають перевагу і значення атрибутів ігнорується.

Приклад: Текстова область

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Текстова область</title>
 </head>
 <body>
  <form action="comment.php" method="post">
   <p><strong>Уведіть ваш коментар:</strong></p>
   <p><textarea name="comment"></textarea></p>
   <p><input type="submit"></p>
  </form>
 </body>
</html>
```

Поля множинного вибору і меню

Крім текстових полів, HTML також дозволяє користувачам вибирати дані, використовуючи множинний вибір та списки, які випадають. Є кілька різних варіантів і полів для цих елементів форми, кожен з яких має свої відмінності та переваги.

Перемикачі

Це простий спосіб, який дозволяє користувачам зробити швидкий вибір з невеликого списку варіантів. Перемикачі дають користувачеві вибрати тільки один варіант з кількох.

Щоб створити перемикач, використовується елемент `<input>` зі значенням `radio` атрибута `type`. Кожен перемикач повинен мати однакове значення атрибуту `name`, щоб усі вони в групі були пов'язані один з одним.

З текстовими полями їх значення визначається тим, що користувач в них набирає; з перемикачами користувач робить вибір з деякої множини значень. Таким чином, ми повинні визначити вхідні значення. Використовуючи атрибут `value` ми можемо встановити конкретне значення для кожного елемента `<input>`.

Крім того, для попереднього вибору деякого значення ми можемо використовувати логічний атрибут `checked`.

Приклад: Перемикачі

```
<form action="radio.php" method="post">
  <input type="radio" name="day" value="Friday" checked> П'ятниця
  <input type="radio" name="day" value="Saturday"> Субота
  <input type="radio" name="day" value="Sunday"> Неділя
```

```
</form>
```

Прапорці

Прапорці дуже схожі на перемикачі. Вони використовують ті ж атрибути і шаблони, за винятком значення атрибута `type`. Різниця між ними полягає в тому, що прапорці дозволяють користувачам вибрати кілька значень і зв'язати їх усі з одним ім'ям, в той час як перемикачі обмежують користувачів одним значенням.

Приклад: Прапорці

```
<form action="checkbox.php" method="post">
  <input type="checkbox" name="day" value="Friday" checked> П'ятниця
  <input type="checkbox" name="day" value="Saturday"> Субота
  <input type="checkbox" name="day" value="Sunday"> Неділя
</form>
```

Списки, які випадають

Списки, які випадають, є ідеальним способом, щоб практично надати користувачам довгий список варіантів. Довгий стовпець перемикачів поряд зі списком різних варіантів не тільки візуально непривабливий, але крім того складний і важкий для розуміння, особливо на мобільному пристрої. З іншого боку, списки, які випадають, забезпечують ідеальний формат для довгого списку варіантів.

Для створення списку застосовують елементи `<select>` і `<option>`. Елемент `<select>` огортає всі пункти меню, а кожен пункт меню розмічений за допомогою елемента `<option>`. Атрибут `name` розташовується в елементі `<select>`, а атрибут `value` розташовується в елементах `<option>`, вкладених в елемент `<select>`. Таким чином, атрибут `value` у кожному елементі `<option>` пов'язаний з атрибутом `name` елемента `<select>`.

Кожен елемент `<option>` огортає текст, який видно користувачам, окремого пункту в списку.

Подібно логічному атрибуту `checked` у перемикачів і прапорців, для випадаючого меню можна використовувати логічний атрибут `selected`, щоб попередньо виділити пункт для користувачів.

Приклад: списки, які випадають

```
<form action="menu.php" method="post">
  <select name=“day”>
    <option value="Friday" selected>П'ятниця</option>
    <option value="Saturday">Субота</option>
    <option value="Sunday">Неділя</option>
  </select>
</form>
```

Множинний вибір

Логічний атрибут `multiple` при додаванні до елементу `<select>` для стандартного списку дозволяє користувачеві вибрати більше одного

варіанта зі списку одночасно. Крім того, за допомогою логічного атрибута `selected`, доданого до більш ніж одного елемента `<option>`, в меню можна заздалегідь вибрати кілька варіантів.

Розміром елемента `<select>` можна керувати за допомогою CSS і він повинен бути скорегований відповідним чином для множинного вибору. Можливо, є сенс інформувати користувачів, що для вибору декількох варіантів вони повинні утримувати клавішу `Shift` під час клацання мишкою, щоб зробити вибір.

Приклад: Множинний вибір

```
<form action="menu.php" method="post">
  <select name=“day” multiple>
    <option value="Friday" selected>П'ятниця</option>
    <option value="Saturday" selected>Субота</option>
    <option value="Sunday">Неділя</option>
  </select>
</form>
```

Кнопки в формі та інші поля

Кнопки

Кнопки є одним з популярних елементів інтерфейсу і часто застосовуються всередині форм, наприклад, для відправки введеної інформації в формі на сервер.

Кнопки на веб-сторінці можна створити декількома способами – за допомогою елемента `<input>` або елемента `<button>`.

Розглянемо спочатку додавання кнопки через `<input>` і його синтаксис.

```
<input type="button" value="Текст на кнопці">
```

Атрибути кнопки `<input>`:

- `name` – ім'я кнопки, призначене для того, щоб обробник форми міг її ідентифікувати.
- `disabled` – блокує кнопку і не дозволяє на неї натискати.
- `form` – ідентифікатор форми для зв'язування кнопки з елементом `<form>`.
- `type` – для звичайної кнопки значенням є `button`.
- `value` – напис на кнопці.
- `autofocus` – кнопка отримує фокус після завантаження документа.

Приклад: Створення кнопки

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Кнопка</title>
 </head>
 <body>
  <form>
```

```
   <p><input type="button" value=" Натиснути    тут "></p>
  </form>
 </body>
</html>
```

Зверніть увагу, що враховуються всі пробіли в написі на кнопці, на відміну від звичайного тексту HTML, тому можна ставити будь-яку кількість пробілів, які в результаті впливають на ширину кнопки. Але правильніше, звичайно ж, задавати вид кнопки через CSS.

Для таких кнопок є ряд обов'язкових правил:

- `<input>` не можна вкладати всередину посилання `<a>`;
- `<input>` не можна вкладати всередину кнопки `<button>`;
- значення атрибута `value` не може бути порожнім.

Другий спосіб створення кнопки заснований на використанні елемента `<button>`. Він за своєю дією нагадує результат, одержаний за допомогою `<input>`. Але, на відміну від нього, пропонує розширені можливості по створенню кнопок. Наприклад, на подібній кнопці можна розміщувати будь-які елементи HTML, включаючи зображення і таблиці. Синтаксис створення такої кнопки наступний.

```
 <button>Напис на кнопці</button>
```

Атрибути, перераховані для кнопки `<input>`, діють і у цьому випадку, але атрибут `value` визначає тільки значення, яке відправляється на сервер, а не напис на кнопці. Якщо потрібно вивести на кнопці зображення, то `<img>` додається всередину `<button>`, як показано в наступному прикладі.

Приклад: Кнопки `button`

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Кнопка</title>
 </head>
 <body>
  <form>
   <p><button>Кнопка з текстом</button></p>
   <p>
   <button>
   Кнопка з рисунком
    <img src="images/but.png" alt="" style="vertical-align:middle">
   </button>
   </p>
  </form>
 </body>
</html>
```

У даному прикладі показано створення звичайної кнопки з текстом, а також кнопки з одночасним використанням тексту і малюнка. Розмір кнопки залежить від вмісту `<button>`, а пробіли ігноруються, тому простим збільшенням їх кількості, як у випадку використання `<input>`, ширину кнопки змінити не вдасться.

Для `<button>` також є ряд обмежень:

- `<button>` не можна вкладати всередину посилання `<a>`;
- один `<button>` не можна вкладати всередину іншого.

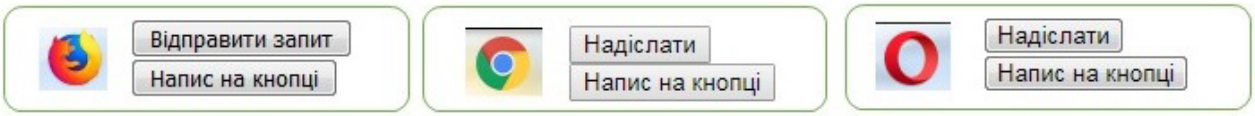
Кнопка `submit`

Для відправки даних на сервер призначена спеціальна кнопка `submit`. Її вигляд нічим не відрізняється від звичайних кнопок, але при натисканні на неї відбувається перехід до обробника форми за адресою, вказаною атрибутом `action` елементу `<form>`. Програма-обробник, отримує дані, введені користувачем в полях форми, проводить з ними необхідні маніпуляції, після чого повертає результат у вигляді HTML-документа. Що саме робить оброблювач, залежить від автора сайту, наприклад, така технологія застосовується при створенні опитувань, форумів, тестів тощо.

Синтаксис створення кнопки `submit` залежить від використовуваного елемента `<input>` або `<button>`.

```
<input type="submit">
<button type="submit">Напис на кнопці</button>
```

Атрибути ті ж, що і для звичайних кнопок `<input>`, але значення атрибута `value` тепер можна не вказувати, тому що браузер підставить текст самостійно, він різниться в залежності від браузера (дивіться наступний рисунок). Сам текст напису ніяк на функціонал кнопки не впливає.



Кнопка `reset`

При натисканні на кнопку `reset` дані форми повертаються до первинних значень. Як правило, цю кнопку застосовують для очищення введеної в полях форми інформації. Для великих форм від використання кнопки `reset` краще взагалі відмовитися, щоб помилково на неї не натиснути, адже тоді доведеться заповнювати форму заново.

Синтаксис створення зазначеної кнопки простий і схожий на інші кнопки.

```
<input type="reset">
<button type="reset">Напис на кнопці</button>
```

Значення кнопки `reset` ніколи не пересилається на сервер. Якщо напис на кнопці опустити, іншими словами, не ставити атрибут `value`, на кнопці за замовчуванням буде додано текст «Скинути».

У наступному прикладі показана форма з одним текстовим полем, яке вже містить попередньо введений текст за допомогою атрибута `value` елемента `<input>`. Після зміни тексту і натиснення на кнопку «Скинути», значення поля буде відновлено і в ньому знову з'явиться напис «Уведіть текст».

Приклад: Кнопки `submit` і `reset`

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Кнопки submit і reset</title>
 </head>
 <body>
  <form>
   <p><input value="Уведіть текст"></p>
   <p><input type="submit">
      <input type="reset"></p>
  </form>
 </body>
</html>
```

Інші поля

Крім застосуваннь, які були вже обговорені, елемент `<input>` має кілька інших варіантів використання. Вони включають в себе отримання прихованих даних і прикріплення файлів в процесі обробки форми.

Приховане поле

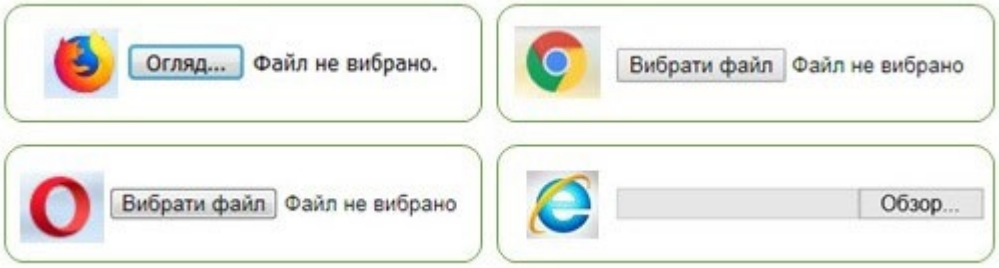
Приховані поля надають спосіб передачі даних на сервер без відображення їх користувачам. Приховані поля зазвичай використовуються для відстеження кодів, ключів або іншої інформації, яка не має відношення до користувача, але може бути корисна при обробці форми. Ця інформація не відображається на сторінці, однак може бути знайдена шляхом перегляду вихідного коду сторінки. З цієї причини вона не повинна застосовуватися для вразливої або захищеної інформації.

Щоб створити приховане поле використовуйте значення `hidden` атрибута `type`. Таке поле додатково включає в себе відповідні значення атрибутів `name` і `value`.

```
  <input type="hidden" name="tracking-code" value="abc-123">
```

Завантаження файлів

Для завантаження на сервер одного або декількох файлів на зразок фотографій, документів і відео застосовується спеціальне поле форми. У браузерах Internet Explorer такий елемент відображається як текстове поле, поруч з яким розташовується кнопка з написом «Огляд ...». У Safari, Opera і Google Chrome доступна тільки кнопка «Вибрати файл», а в Mozilla Firefox це тільки кнопка «Огляд».



При натисканні на кнопку відкривається вікно для вибору файлу, де можна вказати, який файл користувач бажає використовувати.

Синтаксис поля для відправки файлу наступний.

```
<input type="file" name="f">
```

Крім відомих атрибуті для елемента `<input>`, треба відзначити настіпні:

- `type` – для завантаження файлів значення має бути file;
- `accept` – встановлює фільтр на типи файлів, які можна відкрити через поле завантаження файлів;
- `multiple` – дозволяє вибирати і завантажувати відразу декілька файлів.

Для відправлення файлів в формі необхідно зробити наступне:

1. задати метод відправки даних POST (`method = "post"`);
2. встановити у атрибута enctype значення `multipart / form-data`.

Приклад: Завантаження файлів

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Відправка файлів на сервер</title>
 </head>
 <body>
  <p>Форма для завантаження одного файла</p>
  <form enctype="multipart/form-data" method="post">
   <p><input type="file" name="f">
   <input type="submit" value="Відправити"></p>
  </form>
  <p>Форма для завантаження декількох файлів</p>
  <form enctype="multipart/form-data" method="post">
   <p><input type="file" name="f" multiple>
   <input type="submit" value="Відправити"></p>
  </form>
 </body>
</html>
```

При створенні форми для завантаження декількох файлів слід нагадати користувачеві, що декілька файлів одночасно можна обрати, використовуючи клавіші `Ctrl` або `Shift`.

Якщо атрибут `accept` не вказувати, тоді додаються і завантажуються файли будь-якого типу. Наявність `accept` дозволяє обмежити вибір файлу, що особливо важливо, коли потрібно завантажити лише

зображення або відео. Як значення виступає MIME-тип, кілька значень розділяються між собою комою. Також можна використовувати такі ключові слова:

- `audio/*` – вибір музичних файлів будь-якого типу;
- `image/*` – графічні файли;
- `video/*` – відеофайли.

Ось деякі допустимі значення атрибута `accept`.

- `image/jpeg` – тільки файли в форматі JPEG;
- `image/jpeg, image/png` – тільки файли в форматі JPEG і PNG;
- `image/*` – будь-які графічні файли;
- `image/*, video/*` – будь-які графічні і відеофайли.