Що таке Веб-форма?

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 елементу в перелік або відкриття і закриття елемента інтерфейсу).

Веб-форми - їх також часто називають **HTML-форми** - складаються з одного або декількох елементів **управління** форм (іноді їх також називають **віджетами**) і деяких додаткових елементів для структурування форми.

Елементами управління можуть бути однорядкові або багаторядкові текстові поля, списки, що випадають, кнопки, чекбокси, радіо-Баттон, більшість з яких створюються через html-елемент <input>, проте є й інші елементи, про які теж варто дізнатися.

Проектування форми

Перед тим, як почати програмувати, завжди краще зупинитися і подумати про вашу форму. Створення швидкого нарису допоможе визначити вірний набір даних, які ви хочете отримати від користувача. З точки зору UX, зручності використання інтерфейсу, важливо пам'ятати про те, що чим довше ваша форма, тим більше ризик втратити користувачів.

Зробіть форму короткою і лаконічною: запитуйте тільки про ту інформацію, яка вам дійсно необхідна.

Форма буде складатися з трьох текстових полів і однієї кнопки. Ми дізнаємося у користувача його ім'я, e-mail і повідомлення, яке він хоче відправити. Після натискання на кнопку дані будуть відправлені на веб-сервер.

Реалізація HTML-форми

Отже, тепер ми готові звернутися до HTML і створити нашу форму. Для цього ми будемо використовувати такі HTML-елементи: <form>, <label>, <input>, <textarea> і <button>. Створюється форма безпосередньо всередині HTML-шаблону.

Елемент <form>

Створення форми починається з елементу <form>

```
<form action="/my-handling-form-page" method="post">

</form>
```

Цей елемент формально визначає форму. Він є **елементом-контейнером**, як HTML-елементи 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 і method:

- Атрибут action визначає адресу, куди повинні бути надіслані дані після відправки форми.
- Атрибут method вказує, який HTTP-метод буде використаний при передачі даних (це може бути "get" або "post").

Атрибут action

Цей атрибут визначає, куди відправляються дані. Його значення має бути дійсною URL. Якщо цей атрибут не вказаний, дані будуть відправлені на URL-адресу сторінки, що містить форму.

Наприклад, тут дані відправляються на абсолютну URL - http://foo.com:

```
<form action="http://foo.com">
```

Елементи <label>, <input> і <textarea>

Наша контактна форма нескладна: частина, в яку будуть вводитися дані, складається з трьох текстових полів, кожне з яких пов'язане з HTML-елементом <label>:

- Поле введення для імені - **single-line text field**
- Поле введення для пошти - **input of type email**: однорядкове текстове поле, яке приймає тільки email адреси.
- Поле введення для повідомлення - **<textarea>**, многострочное текстове поле.

Тут **елементи ** використовуються для структурування коду і полегшення стилізації. Для доступності і зручності використання вказаний певний текст-підказка 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 до форми.

Такий підхід корисний тим, що дозволяє користувачам з мишею, трекпедом і сенсорним пристроєм клікнути на текст-підказку для активації пов'язаного з ним віджета форми, а також забезпечує читабельним ім'я для користувачів скрін-рідерів.

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

У нашому простому прикладі ми використовуємо <input / text> для першого поля введення - значення за замовчуванням для даного атрибута. Воно являє однорядкове текстове поле, яке може приймати будь-які значення.

Для другого поля введення використовується тип <input / email>, який представляє собою однорядкове текстове поле, яке приймає в якості значення коректно складену email адресу. Він робить просте текстове поле "розумним", дозволяючи перевіряти введений користувачем дані на коректність. Також це дозволяє відкривати більш підходящі для введення e-mail адреси клавіатури (наприклад, з символом @ при базовій розкладці) на пристроях з динамічною клавіатурою, таких як смартфони.

Останнє, але не менш важливе, зверніть увагу на різницю синтаксису у HTML-елементів <input> і <textarea> </ textarea>. Це одна з дивацтв HTML. Тег <input> - це порожній елемент, тобто він не потребує закриваючого тега. <Textarea> - це непорожній елемент, що говорить про те, що йому необхідно закриваючий тег. Це важливо при використанні однієї з властивостей форм: визначення значення за замовчуванням.

Для визначення початкового значення для HTML-елемента <input> необхідно використовувати атрибут value наступним чином:

```
<input type="text" value="по умолчанию в этом элементе находится этот текст" />
```

Для визначення значення за замовчуванням для HTML-елементів <textarea>, просто потрібно помістити це початкове значення між тегами:

```
<textarea>
по умолчанию в этом элементе находится этот текст
</textarea>
```

Елемент <button>

Розмітка форми майже готова, але ще необхідно додати кнопку, яка дозволить користувачеві відправляти або "представляти" інформацію після заповнення форми. Це робиться за допомогою HTML-елемента <button>. Необхідно додати наступний код перед закриваючим тегом </ form>:

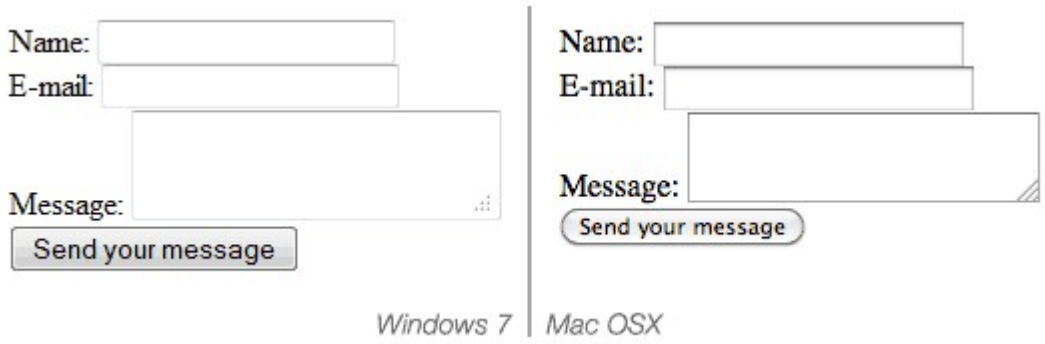
```
<li class="button">
  <button type="submit">Send your message</button>
</li>
```

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

- Клік по кнопці submit (значення за замовчуванням) відправляє дані з форми на сторінку, визначену в атрибуті action елементу <form>.
- Клік по кнопці reset скидає значення всіх елементів управління форми до їх початкового значення. З точки зору UX, це вважається поганою практикою.
- Клік по кнопці button не робить нічого! Звучить дивно, але насправді це дуже зручно використовувати для створення власних кнопок - ви можете визначити їх поведінку через JavaScript.

Базова стилізація форми

Тепер після того, як ви закінчили писати HTML-код форми, збережіть його і відкрийте в браузері. Ви побачите, що на даний момент форма виглядає досить не гарно.



Красиво стилізувати форми досить складно, тому на даний момент просто додамо деякий CSS-код для приведення форми в нормальний вигляд.

Спочатку необхідно додати HTML-елемент <style> на вашу сторінку всередину тега head в HTML:

```
<style>

</style>
```

Усередині тега стилів додайте наступний код:

```
form {
  /* Расположим форму по центру страницы */
  margin: 0 auto;
  width: 400px;
  /* Определим контур формы */
  padding: 1em;
  border: 1px solid #CCC;
  border-radius: 1em;
}

ul {
  list-style: none;
  padding: 0;
  margin: 0;
}

form li + li {
  margin-top: 1em;
}

label {
  /* Определим размер и выравнивание */
  display: inline-block;
  width: 90px;
  text-align: right;
}
```

```

input,
textarea {
  /* Убедимся, что все поля имеют одинаковые настройки шрифта
     По умолчанию в textarea используется моноширинный шрифт */
  font: 1em sans-serif;

  /* Определим размер полей */
  width: 300px;
  box-sizing: border-box;

```

```
  /* Стилизуем границы полей */
  border: 1px solid #999;
}

input:focus,
textarea:focus {
  /* Дополнительная подсветка для элементов в фокусе */
  border-color: #000;
}

textarea {
  /* Выровним многострочные текстовые поля с их текстами-подсказками */
  vertical-align: top;

  /* Предоставим пространство для ввода текста */
  height: 5em;
}

.button {
  /* Выровним кнопки с их текстами-подсказками */
  padding-left: 90px; /* same size as the label elements */
}

button {
  /* Этот дополнительный внешний отступ примерно равен расстоянию
     между текстами-подсказками и текстовыми полями */
  margin-left: .5em;
}
```

Посилання на код: https://mdn.github.io/learning-area/html/forms/your-first-HTML-form/first-form-styled.html

Відправка даних на сервер

Останнє і, напевно, найскладніше - це обробка даних форми на стороні сервера. HTML-елемент <form> визначає куди і яким способом відправити дані завдяки атрибутам action і method.

Ми визначаємо ім'я name для кожного віджета форми. Вказівка імен важлива як для браузера, так і для сервера: браузер дізнається, які імена дати кожній частині даних, а сервер може отримати ці дані, звернувшись до них по

заданому імені. Дані форми відправляються на сервер у вигляді пари – ім'я / значення.

Щоб пойменувати дані, вам необхідно використовувати атрибут name на кожному віджеті форми, який буде збирати певну частину інформації. Поглянемо на код ще раз:

```
<form action="/my-handling-form-page" method="post">
  <div>
    <label for="name">Name:</label>
    <input type="text" id="name" name="user_name" />
  </div>
  <div>
    <label for="mail">E-mail:</label>
    <input type="email" id="mail" name="user_email" />
  </div>
  <div>
    <label for="msg">Message:</label>
    <textarea id="msg" name="user_message"></textarea>
  </div>

  ...
```

У цьому прикладі форма відправить три частини даних з іменами "user_name", "user_email" і "user_message". Ці дані будуть відправлені на URL "/ my-handling-form-page" через метод HTTP POST.

На стороні сервера скрипт, розташований на URL "/ my-handling-form-page" отримає дані у вигляді списку з 3 елементів вигляду – ключ / значення, що містяться в HTTP-запиті. Те, як скрипт буде обробляти дані, залежить від вас. Кожна мова серверного програмування (PHP, Python, Ruby, Java, C # і т.д.) має свій механізм обробки даних з форми.

За матеріалами порталу: https://developer.mozilla.org/