# Формы

Веб быстро превратился из сети страниц для чтения в место, где можно делать покупки, бронировать билеты на самолет, подписывать петиции, искать информацию на сайте, публиковать твиты... и этот список можно продолжать! Веб-формы обрабатывают все эти взаимодействия.

Фактически, в ответ на этот переход от страницы к приложению, HTML5 представил целый ряд новых элементов управления и атрибутов форм, которые упрощают заполнение форм пользователями и их создание разработчиками. Задачи, которые традиционно зависели от JavaScript, теперь можно обрабатывать только с помощью разметки и встроенных функций браузера. HTML5 представляет ряд новых элементов, связанных с формами, новые типы ввода и множество новых атрибутов. Некоторые из этих функций ждут своей реализации в браузерах, поэтому будет отмечено, какие элементы управления могут поддерживаться не везде.

В этой лекции мы рассмотрим веб-формы, принципы их работы и разметку, используемую для их создания. Будет кратко рассмотрена важность дизайна веб-форм.

## КАК РАБОТАЮТ ФОРМЫ

Рабочая форма состоит из двух частей. Первая часть — это сама форма, которая отображается на странице и создана с использованием HTML-разметки. Формы состоят из кнопок, полей ввода и раскрывающихся меню (совместно именуемых элементами управления формы), используемых для сбора информации от пользователя. Формы также могут содержать текст и другие элементы.

Другой компонент веб-формы — это приложение или скрипт на сервере, который обрабатывает собранную формой информацию и возвращает соответствующий ответ. Именно он обеспечивает работу формы. Другими словами, публикации HTML-документа с элементами формы недостаточно. Веб-приложения и скрипты требуют навыков программирования, которые выходят за рамки данного курса.

## От ввода данных до ответа

Если вы собираетесь создавать веб-формы, полезно понимать, что происходит за кулисами. В этом примере прослеживаются этапы транзакции с использованием простой формы, которая собирает имена и адреса электронной почты для списка рассылки; однако это типичный процесс для многих форм.

1. Ваш посетитель (назовём её Салли) открывает страницу с веб-формой в окне браузера. Браузер видит элементы управления формы в разметке и отображает их вместе с соответствующими элементами управления формы на странице, включая два поля для ввода текста и кнопку «Отправить» (показано на РИСУНКЕ 1).

2. Салли хочет подписаться на эту рассылку, поэтому она вводит своё имя и адрес электронной почты в поля и отправляет форму, нажимая кнопку «Отправить».

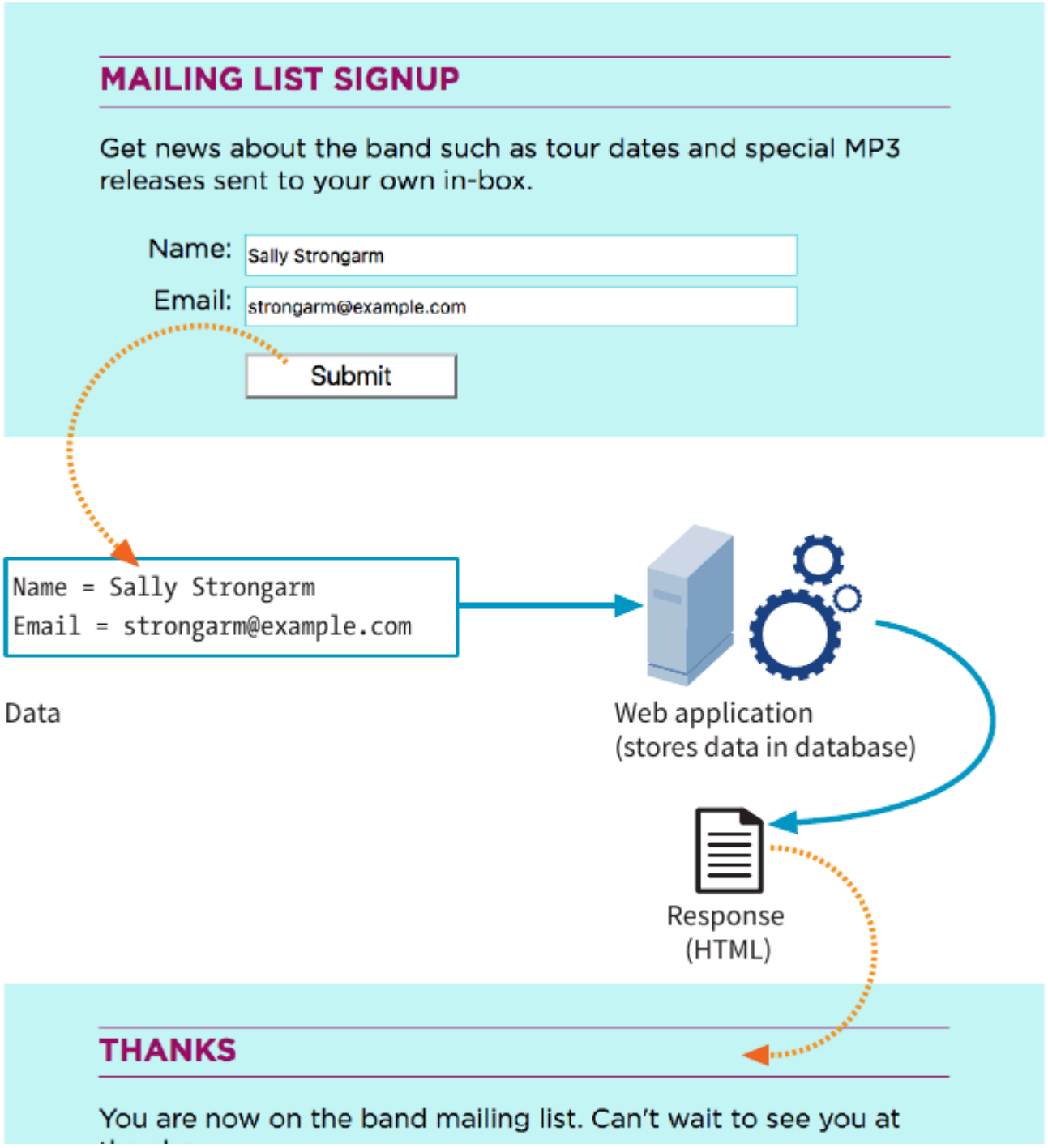
3. Браузер собирает введённую информацию, кодирует её и отправляет веб-приложению на сервере.

4. Веб-приложение принимает информацию и обрабатывает её (то есть выполняет запрограммированные действия). В этом примере имя и адрес электронной почты добавляются в базу данных списка рассылки.

5. Веб-приложение также возвращает ответ. Вид возвращаемого ответа зависит от содержания и назначения формы. В данном случае ответ представляет собой простую веб-страницу с благодарностью за подписку на рассылку. Другие приложения могут реагировать, перезагружая страницу формы с обновлённой информацией, перенаправляя пользователя на другую связанную страницу формы или выдавая сообщение об ошибке, если форма заполнена неправильно, и это лишь несколько примеров.

6. Сервер отправляет ответ веб-приложения обратно в браузер, где он отображается. Салли видит, что форма сработала и что она добавлена в

список рассылки.


MAILING LIST SIGNUP
Get news about the band such as tour dates and special MP3 releases sent to your own in-box.
Name:
Sally Strongarm
Email:
strongarm@example.com
Submit
Name = Sally Strongarm
Email = strongarm@example.com
Data
Web application
(stores data in database)
Response
(HTML)
THANKS
You are now on the band mailing list. Can't wait to see you at


Формы добавляются на веб-страницы с помощью элемента form. Элемент form представляет собой контейнер для всего содержимого формы, включая ряд элементов управления, таких как поля ввода текста и кнопки. Он также может содержать блочные элементы (например, h1, p и списки). Однако он может не содержать других элементов form.

Этот пример исходного документа содержит форму, похожую на показанную на РИСУНКЕ 1:

```
<!DOCTYPE html>
<html>
<head>
<title>Mailing List Signup</title>
<meta charset="utf-8">
</head>
<body>
<h1>Mailing List Signup</h1>
<form action="/mailinglist.php" method="`POST`">
<fieldset>
<legend>Join our email list</legend>
<p>Get news about the band such as tour dates and special MP3
releases sent to your own in-box.</p>
<ol>
<li><label for="firstlast">Name:</label>
<input type="text" name="fullname" id="firstlast"></li>
<li><label for="email">Email:</label>
<input type="text" name="email" id="email"></li>
</ol>
<input type="submit" value="Submit">
</fieldset>
```

```
22  </form>
23  </body>
24  </html>
```

Помимо того, что элемент формы является контейнером для элементов управления формы, он имеет ряд атрибутов, необходимых для взаимодействия с программой обработки форм на сервере. Давайте рассмотрим каждый из них.

## Атрибут `action`

Атрибут `action` указывает местоположение (URL) приложения или скрипта, который будет использоваться для обработки формы. В этом примере атрибут `action` отправляет данные в скрипт mailinglist.php:

```
1  <form action="/mailinglist.php" method="`POST`">...</form>
```

Суффикс .php указывает, что эта форма обрабатывается скриптом, написанным на языке PHP, но веб-формы могут обрабатываться с

помощью любой из следующих технологий:

- PHP (.php) — язык сценариев с открытым исходным кодом, наиболее часто используемый с веб-сервером Apache. Это самый популярный и широко поддерживаемый вариант обработки форм.
- Microsoft ASP (Active Server Pages; .asp) — среда программирования для Microsoft Internet Information Server (IIS).
- Microsoft ASP.NET (Active Server Page; .aspx) — новый язык программирования Microsoft, разработанный для конкуренции с PHP.
- Ruby on Rails. Ruby — язык программирования, используемый на платформе Rails. На нём построены многие популярные веб-приложения.
- JavaServer Pages (.jsp) — технология на основе Java, похожая на ASP.
- Python — популярный язык сценариев для веб- и серверных приложений.

Существуют и другие варианты обработки форм, которые могут иметь собственные суффиксы или не иметь их вовсе (как в случае с платформой Ruby on Rails).

## Атрибут метода

Атрибут метода определяет, как информация должна быть отправлена на сервер. Давайте используем данные, полученные из примера формы на РИСУНКЕ 1, в качестве примера.

```
fullname = Sally Strongarm
email = strongarm@example.com
```

Когда браузер кодирует эту информацию для передачи на сервер, это выглядит так:

```
fullname=Sally+Strongarm&email=strongarm%40example.com
```

Существует только два метода отправки этих закодированных данных на сервер: `POST` или `GET`, определяемые атрибутом method в элементе form. Этот метод необязателен и, если он не указан, по умолчанию будет `GET`. Мы рассмотрим разницу между этими двумя методами в следующих

разделах этой лекции. В нашем примере используется метод `POST`, как показано ниже:

```
<form action="/mailinglist.php" method="POST">...</form>
```

## Метод `GET`

При использовании метода `GET` закодированные данные формы добавляются непосредственно к URL-адресу, отправляемому на сервер. Вопросительный знак отделяет URL-адрес от следующих данных, как показано здесь:

```
get http://www.bandname.com/mailinglist.php?name=Sally+Strongarm&email=
→ strongarm%40example.com
```

Метод `GET` не подходит, если отправка формы подразумевает выполнение какого-либо `действия`, например, удаление чего-либо или добавление данных в базу данных, поскольку при возвращении пользователя форма будет отправлена повторно.

## Метод `POST`

Если в качестве метода формы выбран `POST`, браузер отправляет отдельный запрос к серверу, содержащий специальные заголовки, за которыми следуют данные. Теоретически содержимое этого запроса видит только сервер, поэтому это лучший метод для отправки защищённой информации, такой как домашний адрес или другая личная информация. На практике убедитесь, что на вашем сервере включён протокол HTTPS, чтобы данные пользователя были зашифрованы и недоступны при передаче.

Метод `POST` также предпочтителен для отправки больших объёмов данных, например, длинного текста, поскольку в нём нет ограничения на количество символов, как в случае `GET`.

Метод `GET` подходит, если вы хотите, чтобы пользователи могли добавлять в закладки результаты отправки формы (например, список результатов поиска). Поскольку содержимое формы находится на виду, метод `GET` не подходит для форм с личной или финансовой информацией. Кроме того, метод `GET` нельзя использовать при загрузке файла.

В этом курсе мы будем использовать более распространённый метод `POST`. Теперь, когда мы разобрались с техническими аспектами элемента формы, давайте обратим внимание на элементы управления формы.

## ПЕРЕМЕННЫЕ И СОДЕРЖИМОЕ

Веб-формы используют различные элементы управления, позволяющие пользователям вводить информацию или выбирать варианты. Типы элементов управления включают в себя различные поля ввода текста, кнопки, меню и несколько элементов управления со специальными функциями. Они добавляются в документ с помощью набора элементов управления формы, которые мы рассмотрим по отдельности.

Как веб-дизайнер, вы должны быть знакомы с параметрами элементов управления, чтобы сделать ваши формы простыми и интуитивно понятными. Также полезно иметь представление о том, что делают элементы управления формы «за кулисами».

## Атрибут name

Задача каждого элемента управления формы — сбор одного бита информации от пользователя. В предыдущем примере формы поля ввода текста собирают имя и адрес электронной почты посетителя. Если использовать технический термин, «fullname» и «email» — это две переменные, собираемые формой. Введённые пользователем данные («Sally Strongarm» и «strongarm@example.com») являются значением или содержимым переменных.

Атрибут name задаёт имя переменной для элемента управления. В этом примере текст, собранный элементом textarea, определён как переменная «comment»:

```
<textarea name="comment" rows="4" cols="45" placeholder="Leave us a comment."></textarea>
```

Когда пользователь вводит комментарий в поле («Это лучшая группа на свете!»), он будет передан на сервер в виде пары имя/значение

(переменная/содержимое) следующим образом:

```
comment=This+is+the+best+band+ever%21
```

Все элементы управления формы должны включать атрибут name, чтобы приложение, обрабатывающее формы, могло сортировать информацию. Вы можете включить атрибут name для кнопок «Отправить» и «Сброс», но это не обязательно, поскольку они выполняют специальные функции (отправка или сброс формы), не связанные со сбором данных.

## Именование переменных

Нельзя просто так присваивать элементам управления имена. Веб-приложение, обрабатывающее данные, запрограммировано на поиск конкретных имён переменных. Если вы разрабатываете форму для работы с существующим приложением или скриптом, вам необходимо узнать конкретные имена переменных для использования в форме, чтобы они работали на одном языке. Имена переменных можно получить из

инструкций, прилагаемых к готовому скрипту на вашем сервере, у вашего системного администратора или программиста, с которым вы работаете.

Если скрипт или приложение будут созданы позже, обязательно присваивайте переменным простые и описательные имена и хорошо документируйте их. Кроме того, во избежание путаницы рекомендуется называть каждую переменную уникально, то есть не использовать одно и то же имя для двух переменных (однако могут быть исключения, когда это желательно). Также следует избегать пробелов в именах переменных. Вместо этого используйте подчёркивание или дефис.

Мы рассмотрели основы элемента формы и то, как именуются переменные. Теперь перейдём к сути разметки формы: элементам управления.

## ОБЗОР ВЕЛИКОЛЕПНЫХ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ФОРМОЙ

Мы разберёмся с разметкой, которая добавляет элементы управления формы на страницу. В этом разделе рассматриваются элементы,

используемые для создания следующих элементов:

- Элементы управления для ввода текста
- Специализированные элементы управления для ввода текста
- Кнопки «Отправить» и «Сброс»
- Переключатели и флажки
- Выпадающие и прокручиваемые меню
- Элементы управления для выбора и загрузки файлов
- Скрытые элементы управления
- Дата и время
- Числовые элементы управления
- Элемент управления для выбора цвета

Как вы увидите, большинство элементов управления добавляются в форму через элемент ввода. Функциональность и внешний вид элемента ввода меняются в зависимости от значения атрибута type в теге. В HTML 5.2 существует двадцать два типа элементов управления вводом. Мы рассмотрим их все.

## Элементы управления вводом текста

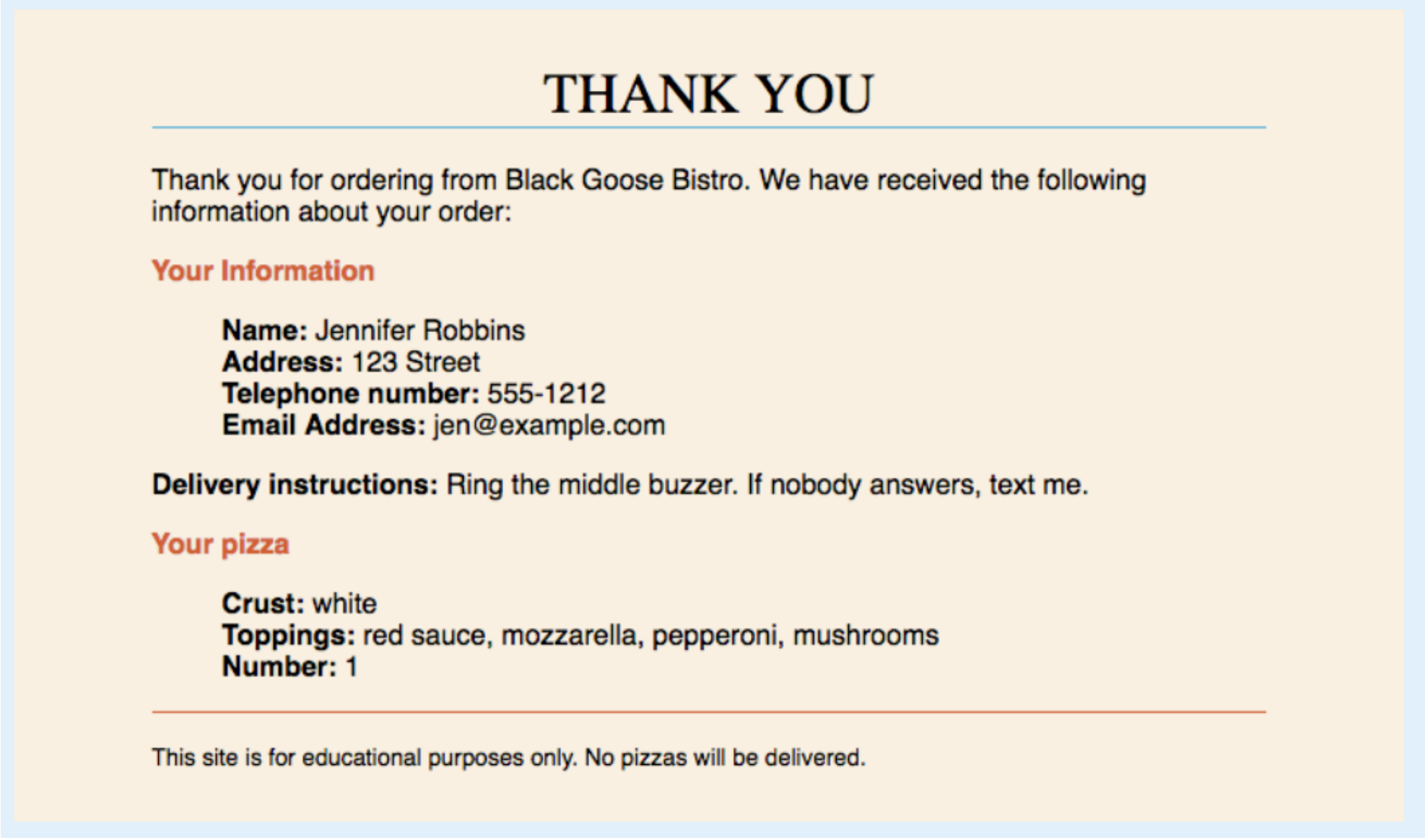
Одной из наиболее распространённых задач веб-форм является ввод текстовой информации. Выбор элемента для сбора текстовых данных зависит от того, требуется ли пользователю ввести одну строку текста (поле ввода) или несколько строк (текстовая область).

Имейте в виду, что если в вашей форме есть поля для ввода текста, она должна использовать безопасный протокол HTTPS для защиты введённого пользователем содержимого при передаче данных на сервер.

### Однострочное текстовое поле

Одним из самых простых типов элементов ввода формы является поле ввода текста для ввода одного слова или строки текста. Фактически, это тип ввода по умолчанию, то есть именно он будет отображаться, если вы забудете указать атрибут type или укажете нераспознанное значение. Добавьте поле ввода текста в форму, вставив элемент ввода с атрибутом type, установленным на text, как показано здесь и на РИСУНКЕ 3:

```
<li><label>Favorite color: <input type="text" name="favcolor"
value="Red" maxlength="50"></label></li>
```



Здесь есть несколько атрибутов, на которые следует обратить внимание:

**name**
Атрибут name необходим для указания имени переменной.

**value**
Атрибут value определяет текст по умолчанию, который отображается в поле при загрузке формы. При сбросе формы она возвращается к этому значению. Значение атрибута value отправляется на сервер, поэтому в этом примере значение «Red» будет отправлено вместе с формой, если пользователь не изменит его.
В качестве альтернативы вы можете использовать атрибут placeholder для подсказки, что нужно ввести в поле, например, «Мой любимый цвет». Значение placeholder не отправляется вместе с формой и является исключительно улучшением пользовательского интерфейса.

**maxlength, minlength**
По умолчанию пользователи могут вводить неограниченное количество символов в текстовое поле независимо от его размера (дисплей

прокручивается вправо, если текст превышает ширину поля). Вы можете установить максимальное количество символов с помощью атрибута maxlength, если это требуется используемой программой обработки форм. Атрибут minlength задаёт минимальное количество символов.

**size**
Атрибут size задаёт длину поля ввода в видимых символах. Однако чаще для задания размера области ввода используются таблицы стилей. По умолчанию виджет текстового ввода отображается с размером, вмещающим 20 символов.

## Многострочное поле ввода текста

Иногда вам может понадобиться, чтобы пользователи могли вводить более одной строки текста. Для таких случаев используйте элемент textarea, который при отображении в браузере заменяется многострочным прокручиваемым полем ввода текста (РИС. 3).

В отличие от пустого элемента input, в элементе textarea вы можете разместить содержимое между открывающим и закрывающим тегами. Содержимое элемента textarea отображается в текстовом поле при отображении формы в браузере. Оно также отправляется на сервер при отправке формы, поэтому внимательно продумайте, что именно туда помещается.

```
<p><label>Official contest entry: <br>
<em>Tell us why you love the band. Five winners will get backstage
passes!</em><br>
<textarea name="contest_entry" rows="5" cols="50">The band is totally
awesome!</textarea></label></p>
```

Атрибуты rows и cols позволяют задать размер текстовой области с помощью разметки. rows определяет количество строк, которые должна отображать текстовая область, а cols — ширину в символах (хотя чаще для задания ширины поля используется CSS). Если пользователь вводит больше текста, чем помещается в отведенное пространство, будут предоставлены полосы прокрутки.

Есть также несколько атрибутов, не показанных в примере. Атрибут wrap определяет, сохраняются ли мягкие переносы строк (когда текст естественным образом переносится по краю поля) при отправке формы. Значение soft (по умолчанию) не сохраняет переносы строк. Значение hard сохраняет переносы строк, когда атрибут cols используется для задания ширины поля. Атрибуты maxlength и minlength задают максимальное и минимальное количество символов, которые можно ввести в поле.

Разработчики часто не добавляют ничего между открывающим и закрывающим тегами, а вместо этого указывают, что должно быть там, с помощью атрибута-заполнителя. В отличие от содержимого текстовой области, текст-заполнитель не отправляется на сервер при отправке формы. Примеры содержимого текстовой области и текста-заполнителя показаны на РИСУНКЕ 3.

```
<p>Official contest entry:<br>
<em>Tell us why you love the band. Five winners will get backstage
passes!</em><br>
<textarea name="contest_entry" placeholder="50 words or less" rows="5"
cols="50"></textarea>
</p>
```

## disabled и readonly

Оба атрибута: отключено и только для чтения запрещают пользователям взаимодействовать с элементом управления формы, но работают немного по-разному.
Когда элемент формы отключен, его нельзя выбрать. Визуальные браузеры могут отображать элемент управления серым цветом по умолчанию (что, конечно, можно изменить с помощью CSS). Состояние «отключено» можно изменить только с помощью скрипта. Это полезный атрибут для ограничения доступа к некоторым полям формы на основе данных, введённых ранее, и может быть применён к любому элементу управления формы или набору полей.

Атрибут readonly запрещает пользователю изменять значение элемента управления формы (хотя его можно выбрать). Это позволяет разработчикам использовать скрипты для установки значений элементов управления в зависимости от других данных, введённых ранее в форму. Элементы ввода, доступные только для чтения, должны иметь чёткие визуальные подсказки, указывающие на то, что они чем-то отличаются от других элементов ввода, иначе они могут сбивать с толку пользователей, пытающихся изменить свои значения. Атрибут readonly можно использовать с текстовыми полями и текстовыми элементами управления (см. ТАБЛИЦУ 1 в самом конце главы).

Самое важное отличие заключается в том, что поля, доступные только для чтения, отправляются при отправке формы, а недоступные — нет.

## Специализированные поля для ввода текста

Помимо стандартного однострочного текстового поля, существует ряд типов полей для ввода определённой информации, такой как пароли,

поисковые запросы, адреса электронной почты, номера телефонов и URL-адреса.

## Поле для ввода пароля

Поле для ввода пароля работает так же, как и текстовое поле, за исключением того, что символы скрыты символами звёздочки (*), маркера (•) или другого символа, определяемого браузером.

Важно отметить, что, хотя символы, вводимые в поле пароля, не видны посторонним, форма не шифрует информацию, поэтому это не следует считать реальной мерой безопасности. Вот пример разметки для поля пароля. На РИСУНКЕ 4 показано, как это может выглядеть после того, как пользователь введет пароль в поле.

```
<li><label for="form-pswd">Password:</label><br>
<input type="password" name="pswd" maxlength="12" id="form-pswd"></li>
```

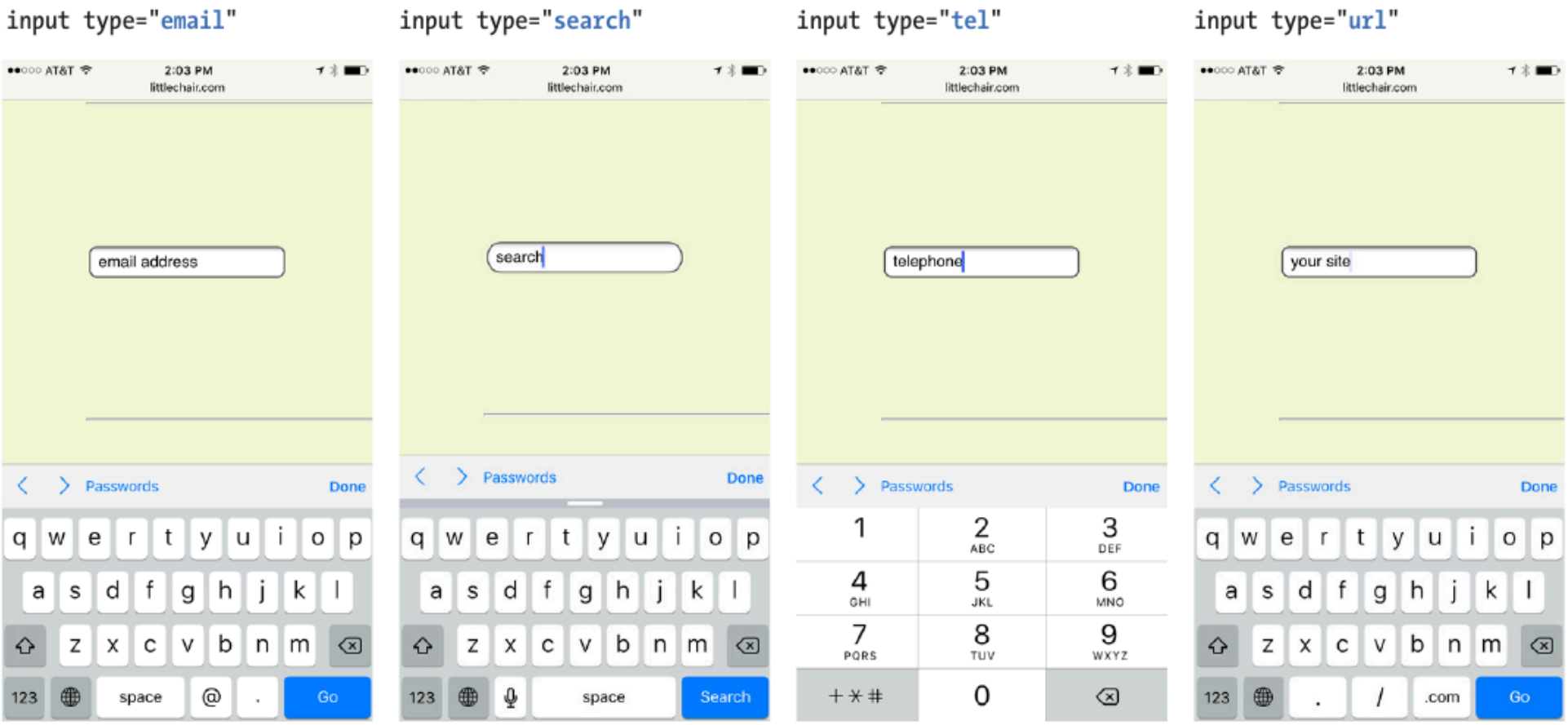
| Fat | |
|---|---|
| Saturated Fat (g) | Unsaturated Fat (g) |

## Search, email, telephone numbers, и URLs

До появления HTML5 единственным способом сбора адресов электронной почты, номеров телефонов, URL-адресов или поисковых запросов была вставка стандартного текстового поля ввода. В HTML5 типы ввода email, tel, url и search дают браузеру подсказку о том, какую информацию ожидать в поле. Эти типы ввода используют те же атрибуты, что и стандартный текстовый тип ввода, описанный ранее (name, maxlength, minlength, size и value), а также ряд других атрибутов.

Все эти типы ввода обычно отображаются как однострочные текстовые поля. Но браузеры, которые их поддерживают, могут делать некоторые интересные вещи с дополнительной семантической информацией. Например, Safari на iOS использует этот тип ввода для предоставления клавиатуры, хорошо подходящей для задачи ввода, например, клавиатуры

с кнопкой поиска для типа ввода поиска или кнопкой «.com», если тип ввода установлен как url (РИС. 5). Браузеры обычно добавляют в поля поиска значок «Очистить поле» одним щелчком мыши (обычно это маленький крестик). Поддерживающий браузер может проверить ввод пользователя на корректность, например, убедившись, что текст, введённый в поле электронной почты, соответствует стандартной структуре адреса электронной почты (ранее для проверки требовался JavaScript). Например, браузеры Opera (РИС. 6) и Chrome выводят предупреждение, если ввод не соответствует ожидаемому формату.




Хотя ввод адресов электронной почты, поисковых систем, телефонов и URL хорошо поддерживается современными браузерами, в их обработке могут наблюдаться несоответствия.

Старые браузеры, такие как Opera Mini и Internet Explorer до 11-й версии, вообще их не распознают, а отображают вместо них стандартный

текстовый ввод по умолчанию, который работает отлично.

## Подсказки в раскрывающемся списке

`<datalist>…</datalist>`

*Ввод в раскрывающемся меню*

Элемент datalist позволяет автору создать раскрывающееся меню с предлагаемыми значениями для любого типа текстового ввода. Он предоставляет пользователю несколько вариантов быстрого доступа, но даже если ни один из них не выбран, пользователь может ввести свой текст. В элементе datalist предлагаемые значения отмечены как элементы option. Используйте атрибут list в элементе input, чтобы связать его с идентификатором соответствующего списка datalist.

В следующем примере (РИС. 7) список datalist предлагает несколько вариантов уровня образования для текстового ввода:

```
<p>Education completed: <input type="text" list="edulevel"
name="education">
<datalist id="edulevel">
<option value="High School">
<option value="Bachelors Degree">
<option value="Masters Degree">
<option value="PhD">
</datalist>
```

На момент написания статьи поддержка datalist в браузерах оставалась неравномерной. Chrome и Opera поддерживают её, но из-за ошибки datalist становится непрокручиваемым (т.е. непригодным к использованию), если список слишком длинный, поэтому лучше всего использовать его для коротких списков. В IE11 и Edge реализация некорректна, а Safari и iOS вообще не поддерживают его. Хорошая новость заключается в том, что если он не поддерживается, браузеры предлагают простое текстовое поле ввода, что является вполне приемлемым запасным вариантом. Вы также можете использовать полифил JavaScript для реализации функциональности datalist.

## Кнопки «Отправить» и «Сбросить»

```
<input type="submit">
```

*Отправляет данные формы на сервер*

```
<input type="reset">
```

*Сбрасывает элементы управления формы к значениям по умолчанию*

Существует несколько типов кнопок, которые можно добавлять в веб-формы. Самая простая — кнопка «Отправить». При нажатии или касании кнопка «Отправить» немедленно отправляет собранные данные формы на сервер для обработки.

Кнопка «Сбросить» возвращает элементы управления формы в состояние, в котором они были при первоначальной загрузке формы. Другими словами, сброс формы не просто очищает все поля.

Кнопки «Отправить» и «Сбросить» добавляются через элемент input. Как упоминалось ранее, поскольку эти кнопки выполняют специфические функции, не включающие ввод данных, они являются единственными

элементами управления формы, которым не требуется атрибут name, хотя при необходимости его можно добавить.

Кнопки «Отправить» и «Сбросить» просты в использовании. Просто разместите их в подходящем месте формы, которое в большинстве случаев находится в самом конце. По умолчанию кнопка «Отправить» отображается с надписью «Отправить» или «Отправить запрос», а кнопка сброса — с надписью «Сброс». Вы можете изменить текст на кнопке, используя атрибут value, как показано на кнопке сброса в этом примере (РИС. 8).

```
<p><input type="submit"> <input type="reset" value="Start over"></p>
```

Кнопка сброса используется в формах не так часто, как раньше. Это связано с тем, что при разработке современных форм мы используем JavaScript для проверки корректности введенных данных, чтобы пользователи получали обратную связь по мере их заполнения.

## Переключатели и флажки

Благодаря продуманному дизайну и поддержке, меньше пользователей доходят до конца формы и им приходится сбрасывать её. Тем не менее, об этой функции полезно знать.

Как флажки, так и переключатели упрощают для ваших посетителей выбор из множества предлагаемых вариантов. Они схожи тем, что работают как небольшие переключатели, которые пользователь может переключать и которые добавляются вместе с элементом ввода. Однако они выполняют разные функции.

Элемент управления формы, состоящий из набора переключателей, подходит, когда разрешен только один вариант из группы, другими словами, когда выборы являются взаимоисключающими (например, «Да или Нет» или «Самовывоз или Доставка»). Когда один переключатель находится в положении «включено», все остальные должны быть в положении «выключено», подобно тому, как работали кнопки на старых радиоприемниках: нажимаешь на один, и остальные выдвигаются.

Однако, когда флажки сгруппированы вместе, можно выбрать любое количество пунктов из группы. Это делает их подходящим выбором для списков, в которых допустимо выбрать более одного пункта.

## Переключатели

Переключатели добавляются в форму через элемент ввода с атрибутом type, установленным на «radio». Ниже приведен синтаксис минимального переключателя:

```
<input type="radio" name="variable" value="value">
```

Атрибут name является обязательным и играет важную роль в объединении нескольких радиокнопок в набор. Если задать для нескольких радиокнопок одно и то же значение name (в следующем примере — «age»), они создадут группу взаимоисключающих вариантов. В этом примере радиокнопки используются в качестве интерфейса для ввода возрастной группы. Один человек не может принадлежать более чем к одной

возрастной группе, поэтому радиокнопки — правильный выбор. На РИСУНКЕ 11 показано, как радиокнопки отображаются в браузере.

```
<p>How old are you?</p>
<ol>
<li><input type="radio" name="age" value="under24" checked> under
24</li>
<li><input type="radio" name="age" value="25-34"> 25 to 34</li>
<li><input type="radio" name="age" value="35-44"> 35 to 44</li>
<li><input type="radio" name="age" value="over45"> 45+</li>
</ol>
```

Обратите внимание, что все элементы ввода имеют одинаковое имя переменной («age»), но их значения различаются. Поскольку это переключатели, одновременно может быть отмечен только один переключатель, и, следовательно, только одно значение будет отправлено на сервер для обработки при отправке формы. Вы можете указать, какая кнопка будет отмечена при загрузке формы, добавив атрибут checked к элементу ввода. В этом примере кнопка рядом с надписью «меньше 24» будет отмечена при загрузке страницы.

## Кнопки-флажки

Флажки добавляются через элемент ввода, тип которого — checkbox. Как и в случае с переключателями, группы флажков создаются путём присвоения им одинакового значения имени. Разница, как мы уже отмечали, заключается в том, что одновременно может быть отмечено несколько флажков. Значение каждого отмеченного переключателя будет отправлено на сервер при отправке формы. Вот пример группы кнопок-флажков, используемых для обозначения музыкальных интересов; на РИСУНКЕ 11 показано, как они выглядят в браузере:

```
<p>What type of music do you listen to?</p>
<ul>
<li><input type="checkbox" name="genre" value="punk" checked> Punk rock</li>
<li><input type="checkbox" name="genre" value="indie" checked> Indie rock</li>
<li><input type="checkbox" name="genre" value="hiphop"> Hip Hop</li>
<li><input type="checkbox" name="genre" value="rockabilly"> Rockabilly</li>
</ul>
```

Конечно, флажки не обязательно использовать группами. В этом примере один флажок используется, чтобы позволить посетителям подписаться на специальные предложения. Значение элемента управления будет передано на сервер только в том случае, если пользователь установит флажок.

```
<p><input type="checkbox" name="OptIn" value="yes"> Yes, send me news and special promotions by email.</p>
```

Кнопки-флажки также используют атрибут checked, чтобы сделать их предварительно выбранными при загрузке формы.

## Меню

Ещё один способ предоставить список вариантов — поместить его в раскрывающееся или прокручиваемое меню. Меню, как правило, компактнее, чем группы кнопок и флажков.

Вы добавляете как раскрывающиеся, так и прокручиваемые меню на форму с помощью элемента select. То, будет ли меню раскрываться или прокручиваться, зависит от того, как вы укажете его размер и позволите ли вы выбрать более одного варианта. Давайте рассмотрим оба типа меню.

### Раскрывающиеся меню

Элемент select по умолчанию отображается как раскрывающееся меню (также называемое раскрывающимся), если размер не указан или атрибут

size равен 1. В раскрывающихся меню можно выбрать только один пункт. Вот пример (показан на РИСУНКЕ 12):

```
<p>What is your favorite 80s band?
<select name="EightiesFave">
<option>The Cure</option>
<option>Cocteau Twins</option>
<option>Tears for Fears</option>
<option>Thompson Twins</option>
<option value="EBTG">Everything But the Girl</option>
<option>Depeche Mode</option>
<option>The Smiths</option>
<option>New Order</option>
</select>
</p>
```

Видно, что элемент select — это всего лишь контейнер для нескольких элементов option. Содержимое выбранного элемента option передаётся веб-приложению при отправке формы. Если по какой-то причине вы хотите отправить значение, отличное от того, что отображается в меню,

используйте атрибут value для указания переопределяющего значения. Например, если кто-то выбирает «Всё, кроме девушки» из примера меню, форма отправляет значение «EBTG» для переменной «EightiesFave». Для остальных элементов в качестве значения будет отправлено содержимое между тегами option.

## Прокручиваемые меню

Чтобы меню отображалось как прокручиваемый список, просто укажите количество видимых строк с помощью атрибута size. В этом примере меню те же параметры, что и в предыдущем, за исключением того, что оно настроено на отображение как прокручиваемый список высотой в шесть строк (РИС. 13):

```
<p>What 80s bands did you listen to?
<select name="EightiesBands" size="6" multiple>
<option>The Cure</option>
<option>Cocteau Twins</option>
<option selected>Tears for Fears</option>
<option selected>Thompson Twins</option>
<option value="EBTG">Everything But the Girl</option>
<option>Depeche Mode</option>
<option>The Smiths</option>
<option>New Order</option>
</select>
</p>
```

Вы можете заметить несколько минимизированных атрибутов. Атрибут multiple позволяет пользователям выбирать несколько пунктов из прокручиваемого списка. Обратите внимание, что раскрывающиеся меню не поддерживают множественный выбор; когда браузер обнаруживает атрибут multiple, он автоматически отображает небольшое прокручиваемое меню по умолчанию.

Используйте атрибут selected в элементе option, чтобы сделать его значением по умолчанию для элемента управления меню. Выбранные пункты подсвечиваются при загрузке формы. Атрибут selected также можно использовать с раскрывающимися меню.

## Группировка пунктов меню

Вы можете использовать элемент optgroup для создания концептуальных групп пунктов. Обязательный атрибут label задаёт заголовок группы. На РИСУНКЕ 14 показано, как группы пунктов отображаются в современных браузерах.

```
<select name="icecream" size="7" multiple>
<optgroup label="traditional">
<option>vanilla</option>
<option>chocolate</option>
</optgroup>
<optgroup label="fancy">
<option>Super praline</option>
<option>Nut surprise</option>
<option>Candy corn</option>
</optgroup>
</select>
```

## Управление выбором файлов

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

*File selection field*

Веб-формы могут собирать не только данные. Их также можно использовать для передачи внешних документов с жёсткого диска

пользователя. Например, типография может использовать веб-форму для загрузки изображений для заказа визитных карточек. Журнал может использовать форму для сбора цифровых фотографий для фотоконкурса.

Элемент выбора файла позволяет пользователям выбрать документ на жёстком диске для отправки вместе с данными формы. Мы добавляем его в форму с помощью элемента ввода, установив тип файла как файл. Приведённый здесь пример разметки (РИС. 15) демонстрирует элемент выбора файла, используемый для отправки фотографий:

```
<form `action`="/client.php" method="`POST`" enctype="multipart/form-data">
<label>Send a photo to be used as your online icon <em>(optional)
</em><br>
<input type="file" name="photo"></label>
</form>
```

Виджет загрузки файлов немного различается в зависимости от браузера и операционной системы, но обычно представляет собой кнопку,

позволяющую получить доступ к файловой системе вашего компьютера (РИС. 15).

Важно отметить, что если форма содержит элемент ввода для выбора файла, необходимо указать тип кодировки (enctype) как multipart/form-data в элементе формы и использовать метод `POST`.

У типа ввода файла есть несколько атрибутов. Атрибут accept информирует браузер о том, какие типы файлов могут быть приняты (аудио, видео, изображение или другой формат, определяемый типом носителя). Добавление нескольких атрибутов позволяет выбрать несколько файлов для загрузки. Атрибут required, как указано в нём, требует выбора файла.

## Скрытые элементы управления

Бывают случаи, когда необходимо отправить в приложение обработки форм информацию, которая не исходит от пользователя. В таких случаях можно использовать скрытый элемент управления формы, который

отправляет данные при отправке формы, но не отображается при её отображении в браузере.

Скрытые элементы управления добавляются через элемент ввода с типом hidden. Его единственное назначение — передать пару «имя/значение» на сервер при отправке формы. В этом примере скрытый элемент формы используется для указания местоположения соответствующего благодарственного документа, который будет отображаться после завершения транзакции:

```
<input type="hidden" name="success-link"
value="http://www.example.com/thankyou.html">
```

## Элементы управления датой и временем

Если вы когда-либо бронировали отель или авиабилет онлайн, вы, несомненно, использовали небольшой виджет календаря для выбора даты. Скорее всего, этот небольшой календарь был создан с помощью JavaScript. HTML5 представил шесть новых типов полей ввода, которые делают

виджеты выбора даты и времени частью стандартных встроенных возможностей отображения браузера, подобно тому, как сегодня они могут отображать флажки, всплывающие меню и другие виджеты.

На момент написания статьи элементы выбора даты и времени реализованы лишь в нескольких браузерах (Chrome, Microsoft Edge, Opera, Vivaldi и Android), но в браузерах, не поддерживающих эти типы полей, дата и время отображаются как вполне удобное текстовое поле. На РИСУНКЕ 16 показаны виджеты даты и времени, отображаемые в Chrome на macOS.

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

*Элемент ввода даты*

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

*Элемент ввода времени*

```
<input type="datetime-local">
```

*Элемент ввода даты/времени*

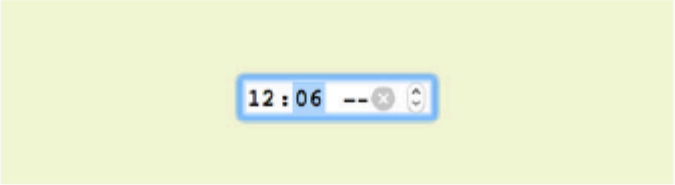
```
<input type="month">
```

*Указывает месяц в году*

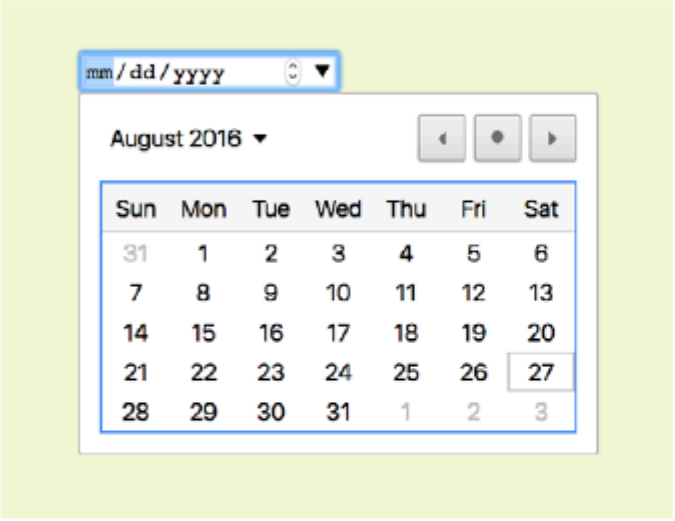
```
<input type="week">
```

*Указывает конкретную неделю в году*

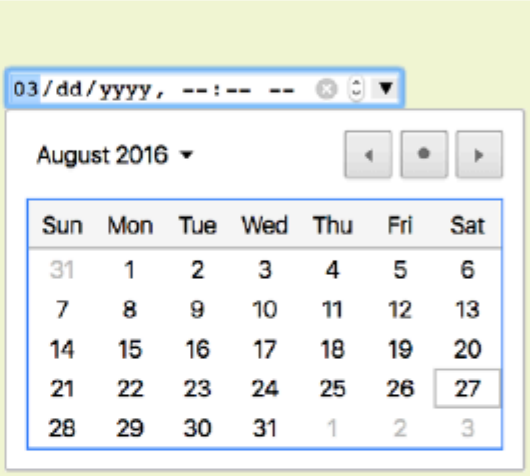
input type="time"



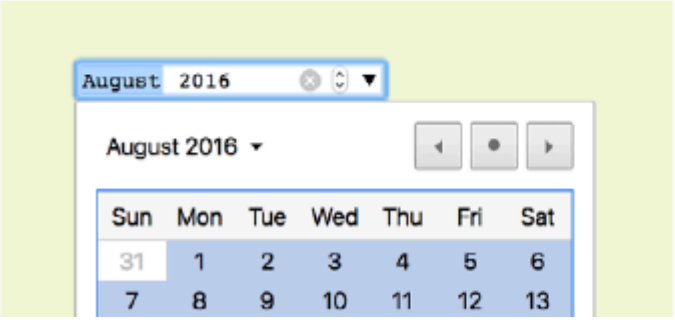

input type="date"



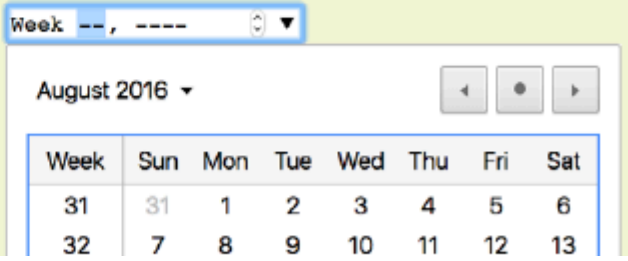

input type="datetime-local"

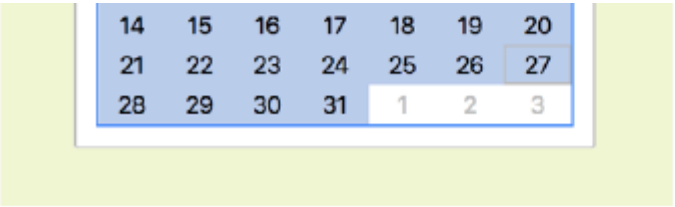


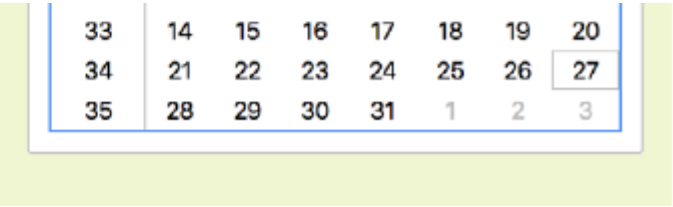

input type="month"




input type="week"






Новые типы данных, связанные с датой и временем:

```
<input type="date" name="name" value="2017-01-14">
```

Создаёт элемент ввода даты, например, всплывающий календарь, для указания даты (год, месяц, день). Начальное значение должно быть указано в формате даты ISO (ГГГГ-ММ-ДД).

```
<input type="time" name="name" value="03:13:00">
```

Создаёт элемент управления для ввода времени (часы, минуты, секунды, дробные части) без указания часового пояса. Значение указывается в формате чч:мм:сс.

```
<input type="datetime-local" name="name" value="2017-01-14T03:13:00">
```

Создаёт комбинированный элемент управления для ввода даты и времени без указания часового пояса (ГГГГ-ММ-ДДТчч:мм:сс).

```
<input type="month" name="name" value="2017-01">
```

Создаёт элемент управления для ввода даты, указывающий конкретный месяц года (ГГГГ-ММ).

```
<input type="week" name="name" value="2017-W2">
```

Создаёт элемент управления для ввода даты, позволяющий указать конкретную неделю в году, используя формат нумерации недель ISO (YYYY-W#).