Формы

Формы — важный интерактивный элемент веб-страницы. Как и ссылки, формы обеспечивают взаимодействие пользователя и страницы, позволяя отправлять данные. Для этого в HTML существуют специальные конструкции, которые говорят браузеру, какие поля может использовать пользователь и как их обрабатывать.

Тема форм достаточно большая, чтобы описать её в рамках одного урока, ведь со всеми возможными нюансами это может вылиться в целый курс. Не бойтесь — в этом уроке вы узнаете достаточно, чтобы научиться делать хорошие формы.

Создание формы

Для отделения формы от остальных участков вёрстки используется специальный тег `<form>`. Если его не использовать, то браузер не поймёт, какие именно данные нужно отправить. Ведь на странице может быть не одна, а сразу несколько различных форм.

```
<form>
  <!-- Элементы формы и кнопка отправки -->
</form>
```

Сервер — удалённый компьютер, на котором хранится весь проект. Сейчас вы смотрите на страницу, которая сгенерирована на сервере и отдана вам по определённому адресу. Этот адрес указан сейчас в вашем браузере.

Примером взаимодействия с формой является любой сайт-поисковик. Например, Google или Yandex. Вы вводите поисковый запрос, который отправляется на сервер. Сервер обрабатывает результат и выдаёт вам подходящие сайты. Это взаимодействие происходит с помощью серверных языков программирования, таких как PHP, Ruby и так далее. В рамках вёрстки мы не можем влиять на то, как обработается результат. Результат вёрстки — набор тегов, с помощью которых браузер соберёт данные и отправит их на сервер.

Тег `<form>` может принимать несколько различных атрибутов. Они не являются обязательными, но их использование может серьёзно повлиять на работу. Самым главным из них является атрибут `action`. Этот атрибут позволяет указать директорию или файл на сервере куда будут отправлены данные. Если этот атрибут не указан, то данные будут отправлены на ту же страницу, где форма и находится.

```
<form action="/forms/helper.php">
  <!-- Данные из формы будут отправлены в PHP файл helper, который расположен в директории
forms на сервере -->
</form>
```

Смысл этого действия вы лучше поймёте, если будете заниматься Backend разработкой. Например, на языке PHP или Python. Если сейчас вам кажется это сложным, то не отчаивайтесь. Разработчики всегда вам подскажут, куда стоит отправлять данные.

Поля формы

Одного тега `<form>` недостаточно для полноценного создания формы, ведь пользователь должен ввести какие-то данные. Самый простой способ — дать какое-то поле для ввода текста, куда пользователь может ввести информацию. В HTML для этого используется два тега:

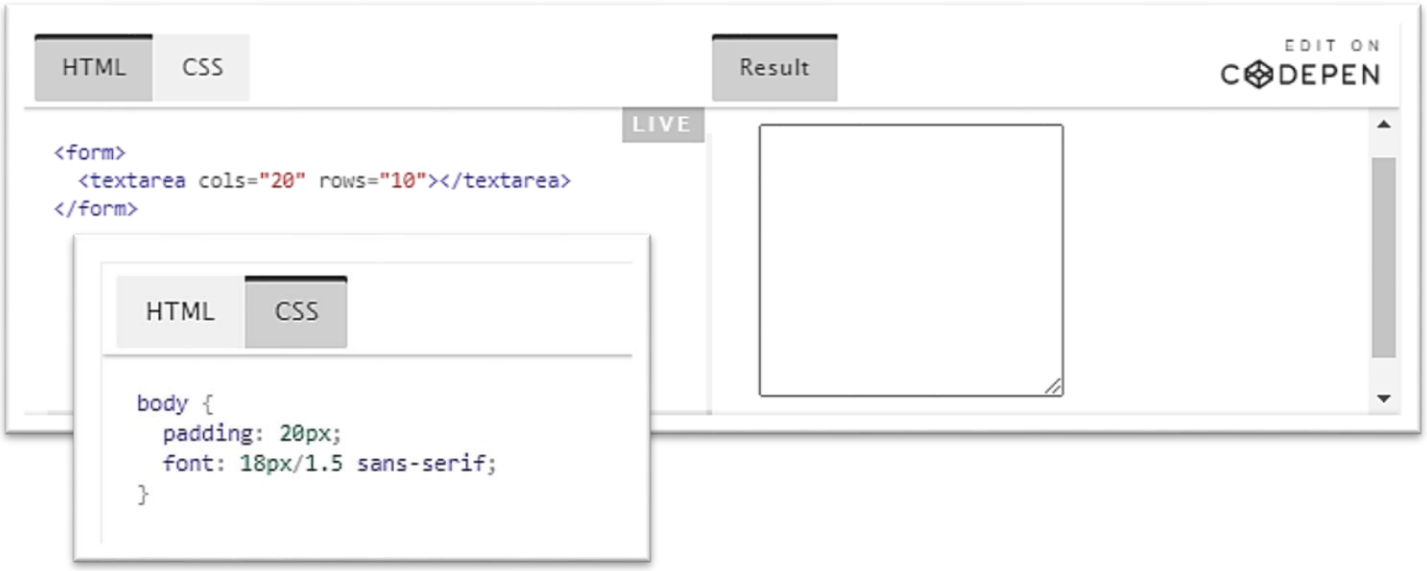
- `<input>` — одиночный тег, в котором располагается небольшая информация. Это может быть телефон, email, дата рождения, имя и так далее.
- `<textarea>` — парный тег, позволяющий пользователю ввести длинное сообщение. Это могут быть комментарии к заказу, отзыв.

textarea

Самый простой тег — `<textarea>`. Его задача — создать большое поле для ввода текста. Если ввести

```
<form>
  <textarea></textarea>
</form>
```

Результат:



В примере вы можете увидеть два атрибута тега `<textarea>`: `cols` и `rows`. Они отвечают за количество символов и строк, которые доступны внутри *textarea*. Если контента будет больше, то появится горизонтальная полоса прокрутки.

Попробуйте изменить размер элемента `<textarea>`. Это можно сделать потянув за правый нижний угол. Так вы можете не только увеличить ширину, но и высоту элемента.

Такое поведение зачастую вредит нашему дизайну. Поэтому разработчики в большинстве случаев запрещают такое поведение. Тут есть несколько вариантов:

- Можно задать высоту и ширину элемента через CSS. Тогда браузер не даст растягивать элемент.

- Использовать свойство `resize` со значением `none`. Его можно использовать, если не требуется поддержка некоторых мобильных браузеров и Internet Explorer.

input

Большую группу полей можно создать используя тег `<input>`, главным атрибутом которого является `type`. Он указывает на то, как именно браузер должен обработать элемент. Рассматривать каждое значение бессмысленно, так как такое количество информации быстро забудется, если её не использовать.

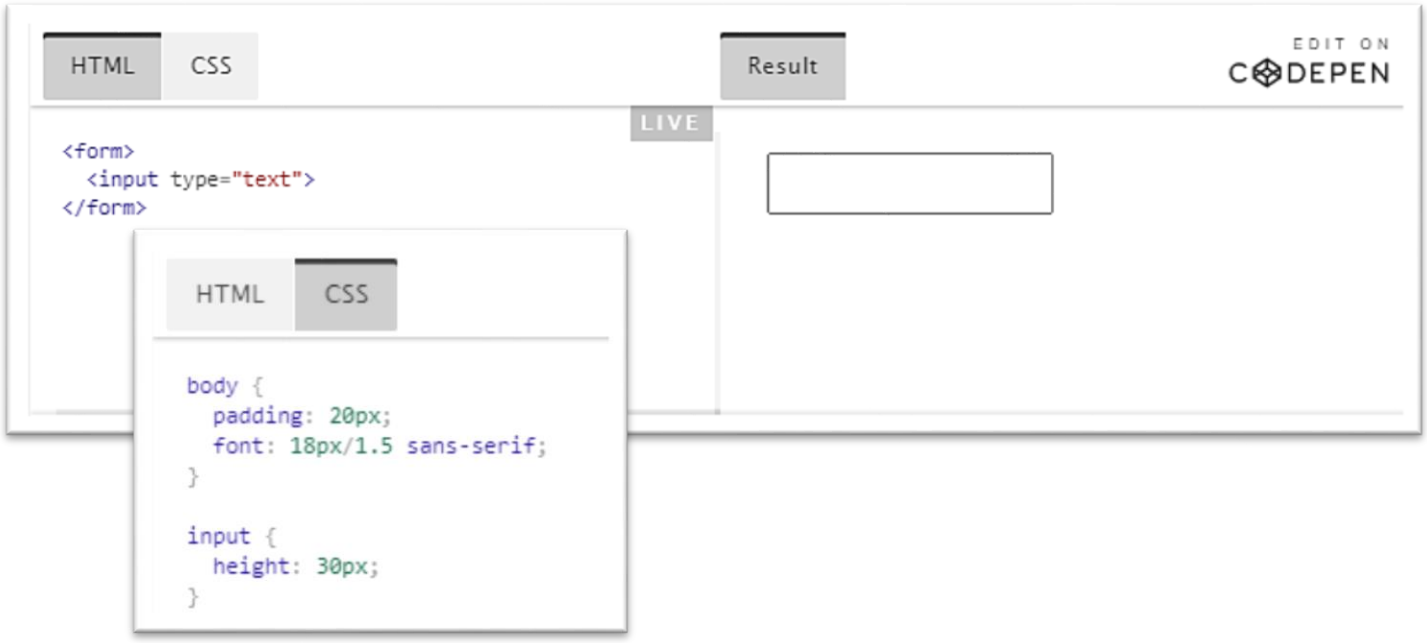
В этом уроке рассмотрим одни из самых популярных значений атрибута `type`.

type="text"

Самый простой вид *input*, который позволяет ввести любую пользовательскую информацию. Никаких проверок полей не происходит. Единственное исключение — удаление переносов строк перед отправкой на сервер. Если нужны данные с переносом текста, то используйте тег `<textarea>`.

```
<form>
  <input type="text">
</form>
```

Результат:



Заметьте, что сейчас поле никак не подписано и из-за этого абсолютно непонятно что надо ввести пользователю. Первое, что приходит в голову — добавить перед полем заголовок или параграф. Да, это создаст видимость описания поля. Но только видимость! С точки зрения семантики в таком варианте нет никакой связи между текстом и полем ввода. Это критично для слепых, которые пользуются скринридерами, так как они не смогут связать название и поле для ввода.

Скринридер (Screen Reader) — устройство для чтения текста с экрана. Используется слепыми или слабовидящими людьми.

Для связи поля и его названия используется тег `<label>`, внутри которого вставляется текст. Чтобы связать `<label>` и `<input>` используется один из двух вариантов. Они равнозначны, поэтому можете использовать тот, который удобен в конкретной ситуации.

- Связь по идентификатору. Для этого тегу `<input>` устанавливается уникальный `id`. Для тега `<label>` устанавливается атрибут `for` значением которого является идентификатор инпута.

```
<form>
  <label for="name">Ваше имя</label>
  <input id="name" type="text">
</form>
```

- Вложение `<input>` внутрь `<label>`. Такой способ помогает избавиться от указания идентификаторов, но может немного усложнить процесс стилизации.

```
<form>
  <label>Ваше имя
    <input type="text">
  </label>
</form>
```

Важно: все элементы, которые доступны пользователю для заполнения должны иметь тег `<label>`. Это элементы `<input>` и `<textarea>`. Это справедливо даже в случае визуального отсутствия подписи к полю.

Для скрытия элемента можно воспользоваться следующим CSS-кодом. В процессе вашего роста как разработчика, вы поймёте все правила, описанные в данном коде. Обычно класс для скрытия элемента называют `.sr-only`. Такой элемент будет невидимым для пользователя, но его сможет «прочитать» скринридер.

```
.sr-only {
  position: absolute;

  width: 1px;
  height: 1px;
  padding: 0;
  border: 0;
  overflow: hidden;

  white-space: nowrap;
}
<form>
  <label for="name" class="sr-only">Ваше имя</label>
  <input id="name" type="text">

  <!-- Теперь label не виден для пользователя, но может быть прочтён скринридером -->
</form>
```

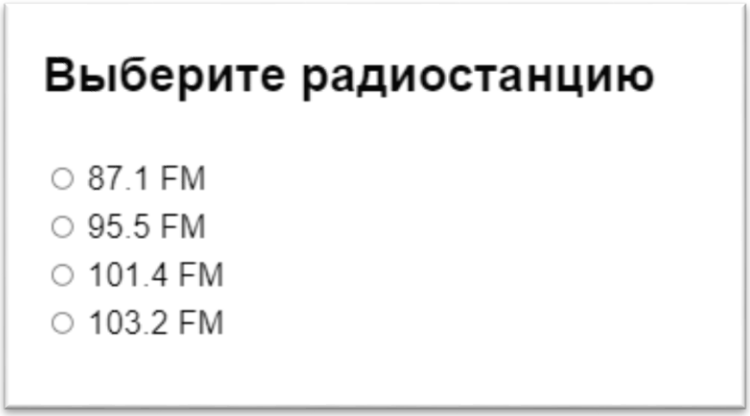
type="radio"

Радиокнопки используются для выбора одного варианта из множества доступных. Почему такой тип называется radio? Есть мнение, что такое название тип получил от первых автомобильных радиоприёмников. В них было доступно несколько радиостанций, но выбрать можно было только одну. Здесь смысл очень похож на принцип таких приёмников.

Помимо атрибутов, которые использовались в прошлых примерах, для радиокнопок есть ещё один важный атрибут — `name`. Он является уникальным для *группы* радиокнопок.

```
<form>
  <h2>Выберите радиостанцию</h2>
  <div>
    <label>
      <input type="radio" name="radio-fm" value="87.1 FM">
      87.1 FM
    </label>
    <br>

    <label>
      <input type="radio" name="radio-fm" value="95.5 FM">
      95.5 FM
    </label>
    <br>

    <label>
      <input type="radio" name="radio-fm" value="101.4 FM">
      101.4 FM
    </label>
    <br>

    <label>
      <input type="radio" name="radio-fm" value="103.2 FM">
      103.2 FM
    </label>
  </div>
</form>
```

Результат:



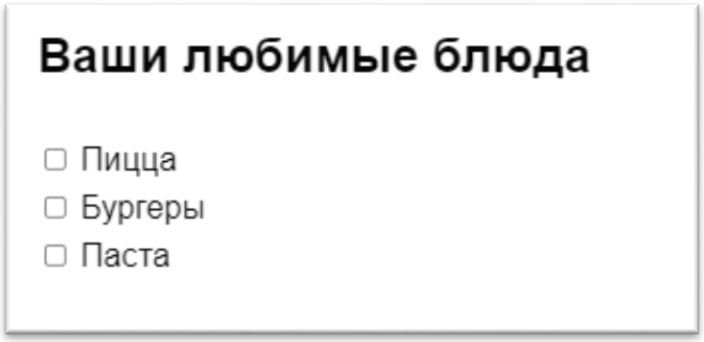
Помимо атрибута `name` у радиокнопок используется атрибут `value`. Внутри него находится то значение, которое отправится на сервер.

type="checkbox"

Чекбоксы немного похожи на радиокнопки, но имеют существенное отличие — возможность выбора сразу нескольких значений. Представьте, что пользователь выбирает любимые блюда. Скорее всего это не одно блюдо, а сразу несколько. Можно дать возможность просто их написать с помощью `<textarea>`, но грамотнее будет использовать чекбоксы.

```
<form>
  <h2>Ваши любимые блюда</h2>
  <div>
    <label>
      <input type="checkbox" name="dishes" value="pizza">
      Пицца
    </label>
    <br>

    <label>
      <input type="checkbox" name="dishes" value="burger">
      Бургеры
    </label>
    <br>

    <label>
      <input type="checkbox" name="dishes" value="paste">
      Паста
    </label>
  </div>
</form>
```

Результат:



Списки

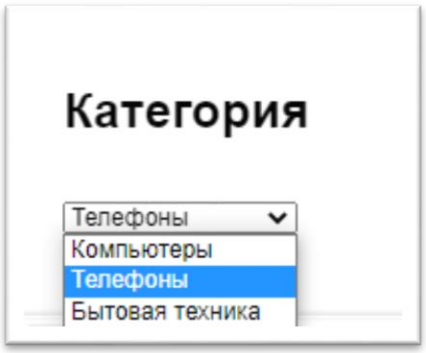
Отдельным элементом формы являются списки. Представьте себе фильтр на сайте по продаже техники. Одновременно вы можете отфильтровать товары только в одной категории. Для этого можно использовать радиокнопки или даже чекбоксы в случае множественного выбора. Но есть ещё один способ — использование списка.

Создание списка очень похоже на создание списка текста. В формах для этого используется тег `<select>` пункты которого лежат внутри тегов `<option>`.

```
<form>
  <h2>Категория</h2>
  <select name="category">
    <option value="computer">Компьютеры</option>
```

```
    <option value="phone" selected>Телефоны</option>
    <option value="appliances">Бытовая техника</option>
  </select>
</form>
```

Результат:



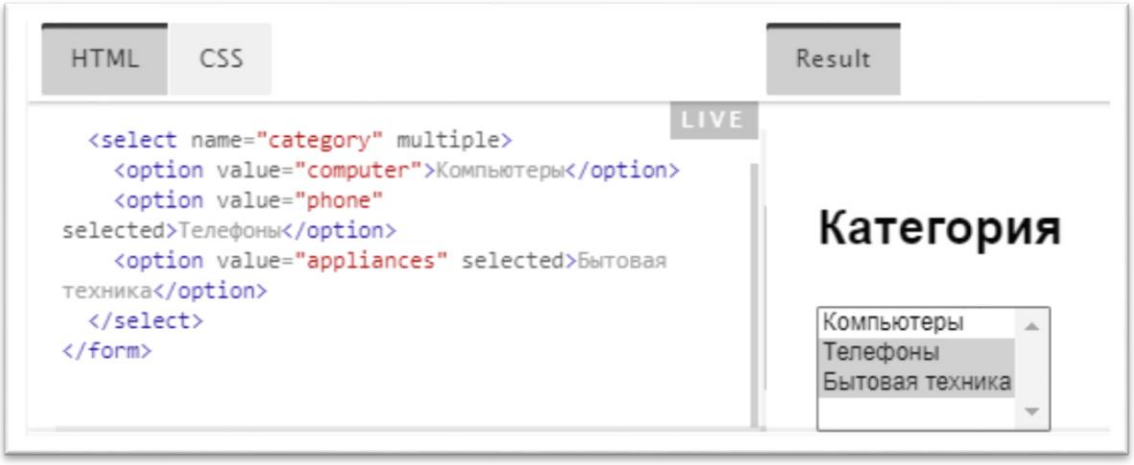
В примере появился новый атрибут — `selected`. Он отвечает за то, какой элемент выбран по умолчанию. Если он не указан, то будет выбран первый элемент списка.

Для элементов радиокнопок и чекбоксов для выбора по умолчанию используется атрибут `checked`

Для списков доступен ещё один интересный атрибут — `multiple`. С его помощью список становится похож на чекбоксы и позволяет выбрать сразу несколько пунктов.

```
<form>
  <h2>Категория</h2>
  <select name="category" multiple>
    <option value="computer">Компьютеры</option>
    <option value="phone" selected>Телефоны</option>
    <option value="appliances" selected>Бытовая техника</option>
  </select>
</form>
```

Результат:



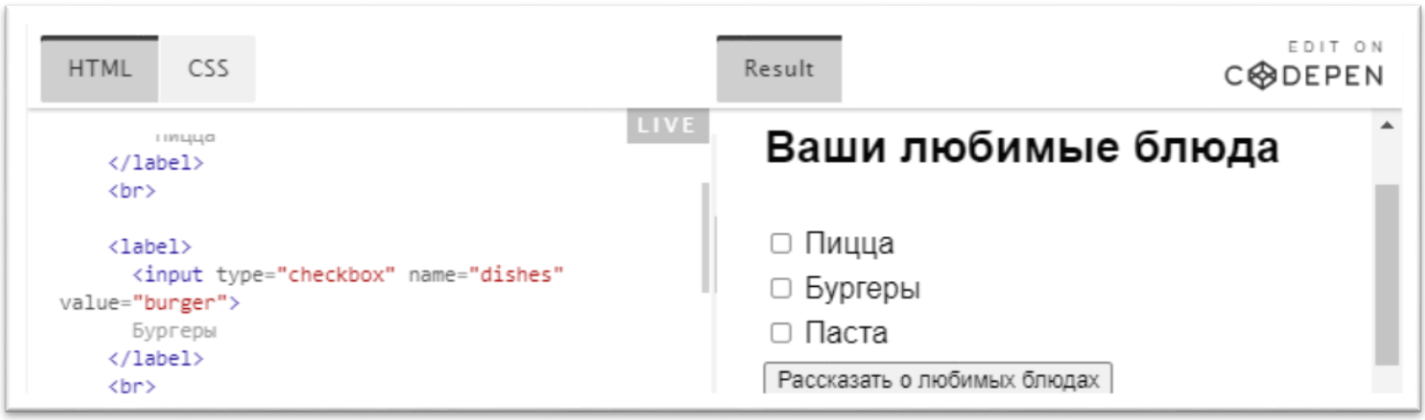
Отправка формы

В прошлом разделе мы рассмотрели различные способы взаимодействия пользователя и формы. Но это не имеет никакого смысла, если форма не будет отправлена. Для этого нужно создать элемент, при взаимодействии с которым будет отправлена форма.

Отправка формы может быть осуществлена одним из двух способов:

- Создание `<input>` с типом `submit`. В этом случае создастся элемент в виде кнопки с именем, указанным в качестве значения атрибута `value`.
- Использование парного тега `<button>`. Такой способ больше развязывает руки, так как внутри такой кнопки может содержаться любой HTML код, при клике на который будет отправлена форма. Такой способ очень удобен для создания кнопки в виде иконки. Сейчас такой способ наиболее актуальный.

Результат:



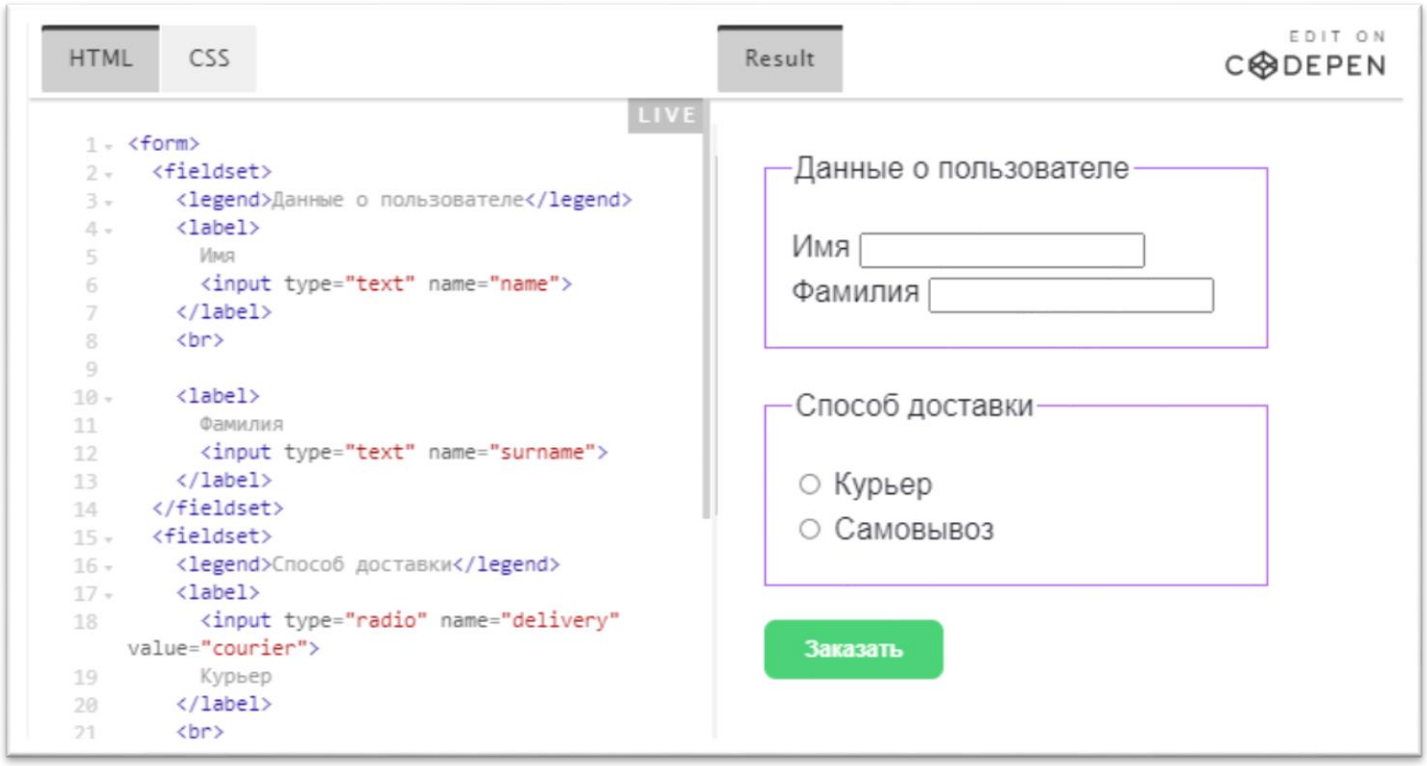
Разделение участков формы

Страшный сон пользователя — длинная форма без разделения на секции. К центру такой формы уже непонятно, для чего именно заполняются те или иные поля. Можно отделять секции с помощью заголовков или визуально. Но браузеры предоставляют специальный механизм для разделения участков формы на логические секции — `fieldset`. Их можно рассматривать как альтернативу тегу `<section>`, но только для форм. В качестве заголовка выступает тег `<legend>`.

```
<form>
  <fieldset>
    <legend>Данные о пользователе</legend>
    <label>
      Имя
      <input type="text" name="name">
    </label>
    <br>

    <label>
      Фамилия
      <input type="text" name="surname">
    </label>
  </fieldset>
  <fieldset>
```

```
    <legend>Способ доставки</legend>
    <label>
      Курьер
      <input type="radio" name="delivery" value="courier">
    </label>
    <br>

    <label>
      Самовывоз
      <input type="radio" name="delivery" value="pickup">
    </label>
  </fieldset>

  <button>Заказать</button>
</form>
```

Результат:



Текст внутри текстового поля

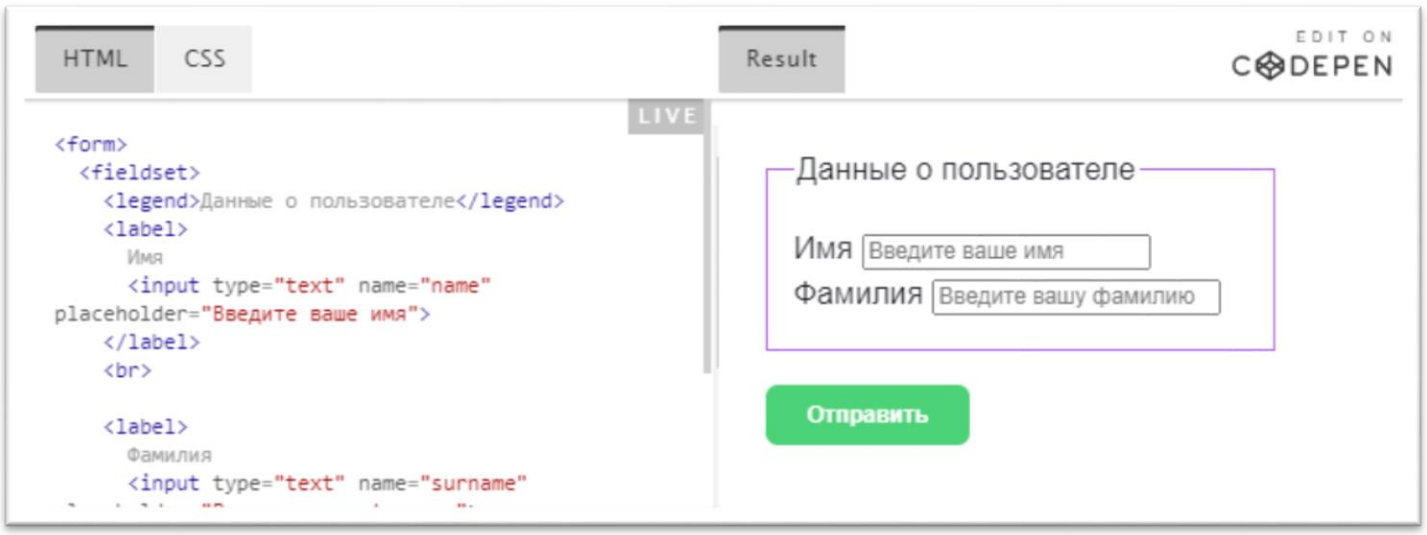
Во всех примерах этого урока текстовые поля изначально были пустыми. Наверное вы часто видели, что до фокуса на поле есть текст, предлагающий ввести данные или показывает формат, в котором ожидаются данные.

За вывод такого текста отвечает атрибут `placeholder`, значением которого является текст, выводимый до возникновения события фокуса.

```
<form>
  <fieldset>
    <legend>Данные о пользователе</legend>
    <label>
      Имя
      <input type="text" name="name" placeholder="Введите ваше имя">
    </label>
```

```
    <br>

    <label>
      Фамилия
      <input type="text" name="surname" placeholder="Введите вашу фамилию">
    </label>
  </fieldset>
  <button>Отправить</button>
</form>
```

Результат:



Дополнительное задание

Создайте форму для регистрации пользователя на сайте. Форма должна принимать следующие данные:

- Имя
- Фамилия
- Логин
- Дата рождения
- Пароль

Используйте различные вариации типов у `<input>`. Все возможные типы вы сможете найти в дополнительной информации этого урока.

Дополнительная информация

- Атрибуты тега form
- Типы элемента input
- Свойство resize