Частное учреждение образования

Колледж бизнеса и права

|  |
| --- |
| УТВЕРЖДАЮ  Ведущий методист колледжа  \_\_\_\_\_\_\_\_\_\_\_Е.В. Паскал  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2021 |
| Специальность: 2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Программные средства создания Интернет-приложений» | |

Лабораторная работа

Инструкционно-технологическая карта

Тема: «Создание форм на HTML-странице и изменение свойств элементов формы»

Цель: Изучить основные тэги разработки форм, особенности подключения формы и атрибуты основных элементов управления формы, приобрести навыки разработки форм

Время выполнения: 2 часа

СОДЕРЖАНИЕ РАБОТЫ

1. Контрольные вопросы.
2. Теоретические сведения для выполнения работы.
3. Порядок выполнения работы.
4. Домашнее задание.
5. Литература.
6. КОНТРОЛЬНЫЕ ВОПРОСЫ
7. Какие значения может принимать атрибут Method? В каком случае используется то или иное значение?
8. Для чего предназначен атрибут Enctype?
9. Какие теги применяются для создания полей для ввода данных внутри формы?
10. Каково назначение тега <INPUT>?
11. Перечислите атрибуты тега <INPUT>. Каково их назначение?
12. Какие значения может принимать атрибут Type тега<INPUT>?
13. Какой элемент будет добавлен в форму, если атрибут Type тега<INPUT> имеет значение Text?
14. Какой элемент будет добавлен в форму, если атрибут Type тега<INPUT> имеет значение Password?
15. Какой элемент будет добавлен в форму, если атрибут Type тега<INPUT> имеет значение File?
16. Какой элемент будет добавлен в форму, если атрибут Type тега<INPUT> имеет значение Hidden?
17. Какой элемент будет добавлен в форму, если атрибут Type тега<INPUT> имеет значение Radio?

**2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

[**Что такое Веб-форма?**](https://developer.mozilla.org/ru/docs/Learn/Forms/Your_first_form#%D1%87%D1%82%D0%BE_%D1%82%D0%B0%D0%BA%D0%BE%D0%B5_%D0%B2%D0%B5%D0%B1-%D1%84%D0%BE%D1%80%D0%BC%D0%B0)

**Веб-формы** являются одним из основных элементов взаимодействия между пользователем и сайтом или приложением. Формы позволяют пользователю ввести данные, которые затем отправляются на сервер для их дальнейшей обработки и хранения или используются на стороне клиента для обновления интерфейса (например, добавление нового элемента в список или открытие и закрытие элемента интерфейса).

**Веб-формы** — их также часто называют **HTML-формы** — состоят из одного или нескольких **элементов управления форм** (иногда их также называют **виджетами**) и некоторых дополнительных элементов для структурирования формы. Элементами управления могут быть однострочные или многострочные текстовые поля, выпадающие списки, кнопки, чекбоксы, радиокнопки, большинство из которых создаются через html-элемент [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input), однако есть и другие элементы, о которых тоже стоит узнать.

В элементах управления форм можно задать правила, указывающие на определённый формат данных или значений, которые могут быть введены (**валидация форм**), а также к ним могут быть добавлены текстовые строки, описывающие эти элементы для зрячих и незрячих пользователей.

[**Проектирование формы**](https://developer.mozilla.org/ru/docs/Learn/Forms/Your_first_form#%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%84%D0%BE%D1%80%D0%BC%D1%8B)

Перед тем, как начать программировать, всегда лучше остановиться и подумать о вашей форме. Создание быстрого наброска поможет определить верный набор данных, которые вы хотите получить от пользователя. С точки зрения UX, удобства использования интерфейса, важно помнить о том, что чем длиннее ваша форма, тем больше риск потерять пользователей. Сделайте форму краткой и лаконичной: спрашивайте только о той информации, которая вам действительно необходима.

Проектирование форм является важным этапом при создании сайта или приложения. Удобство использования форм (UX) выходит за рамки данной статьи, однако если вы хотите углубиться в эту тему, то вам следует прочитать следующие статьи:

* На Smashing Magazine есть [хорошие статьи на тему UX в HTML-формах](https://www.smashingmagazine.com/2018/08/ux-html5-mobile-form-part-1/), включая старую, но всё ещё актуальную статью [Extensive Guide To Web Form Usability](https://www.smashingmagazine.com/2011/11/extensive-guide-web-form-usability/).
* Также есть UXMatters — хороший ресурс с полезными советами от [базовых лучших практик](http://www.uxmatters.com/mt/archives/2012/05/7-basic-best-practices-for-buttons.php) до сложных решений, таких как [мультистраничные формы](http://www.uxmatters.com/mt/archives/2010/03/pagination-in-web-forms-evaluating-the-effectiveness-of-web-forms.php).

В этой статье мы создадим простую контактную форму. Давайте сделаем набросок.

Наша форма будет состоять из трёх текстовых полей и одной кнопки. Мы узнаём у пользователя его имя, e-mail и сообщение, которое он хочет отправить. После нажатия на кнопку данные будут отправлены на веб-сервер.

[**Активное обучение: Реализация HTML-формы**](https://developer.mozilla.org/ru/docs/Learn/Forms/Your_first_form#%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_html-%D1%84%D0%BE%D1%80%D0%BC%D1%8B)

Итак, теперь мы готовы обратиться к HTML и создать нашу форму. Для этого мы будем использовать следующие HTML-элементы: [<form>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form), [<label>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label), [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input), [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea) и [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button).

Прежде, чем продолжить, скопируйте [простой HTML-шаблон](https://github.com/mdn/learning-area/blob/master/html/introduction-to-html/getting-started/index.html) — вы будете создавать свою форму внутри него.

**ЗАДАНИЕ 1**

1. [**Элемент**](https://developer.mozilla.org/ru/docs/Learn/Forms/Your_first_form#%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_htmlelementform)[**<form>**](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form)

Создание форм начинается с элемента [<form>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form):

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

</form>

Этот элемент формально определяет форму. Он является элементом-контейнером, как HTML-элементы [<div>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/div) или [<p>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/p), но при этом он поддерживает некоторые специфические атрибуты для настройки поведения формы. Все атрибуты являются опциональными, но в стандартной практике принято указывать атрибуты action и method:

* Атрибут action определяет адрес, куда должны быть посланы данные после отправки формы.
* Атрибут method указывает, какой HTTP-метод будет использован при передаче данных (это может быть "get" или "post").

Теперь добавьте указанный выше код с элементом [<form>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form) внутрь тега [<body>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/body) в вашем HTML.

**Внимание:** Строго запрещается размещать форму внутри другой формы. Такое размещение может привести к непредсказуемому поведению форм, в зависимости от браузера.

1. [**Элементы**](https://developer.mozilla.org/ru/docs/Learn/Forms/Your_first_form#%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%8B_htmlelementlabel_htmlelementinput_%D0%B8_htmlelementtextarea)[**<label>**](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label)**,**[**<input>**](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input)**и**[**<textarea> (en-US)**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea)

Наша контактная форма несложная: часть, в которую будут вводиться данные, состоит из трёх текстовых полей, каждое их которых связано с HTML-элементом [<label>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label):

* Поле ввода для имени — [single-line text field (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/text)
* Поле ввода для e-mail — [input of type email (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/email" \o "Currently only available in English (US)): однострочное текстовое поле, которое принимает только e-mail адреса.
* Поле ввода для сообщения — [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea), многострочное текстовое поле.

В терминах HTML нам нужен код наподобие представленного ниже, чтобы добавить виджеты форм:



Добавьте в вашу форму код, чтобы она выглядела так же, как форма выше.

Здесь элементы [<li>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/li) используются для структурирования кода и облегчения стилизации (будет разобрано далее в статье). Для доступности и удобства использования мы указали определённый текст-подсказку для каждого элемента управления. Обратите внимание на использование атрибута for на каждом элементе [<label>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label), который принимает в качестве значение id элемента управления формы, с которым он связан — этот подход позволяет привязать тексты-подсказки к форме.

Такой подход полезен тем, что позволяет пользователям с мышью, трекпадом и сенсорным устройством кликнуть на текст-подсказку для активации связанного с ним виджета формы, а также обеспечивает читабельное имя для пользователей скрин-ридеров. Вы найдёте более детальный разбор текстов-подсказок в статье [Как структурировать HTML-форму](https://developer.mozilla.org/ru/docs/Learn/Forms/How_to_structure_a_web_form).

В HTML-элементе [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input) самым важным атрибутом является атрибут type. Этот атрибут чрезвычайно важен, потому что он определяет внешний вид и поведение элемента [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input). Вы найдёте больше информации об этом далее в статье [Стандартные виджеты форм](https://developer.mozilla.org/ru/docs/Learn/Forms/Basic_native_form_controls).

* В нашем простом примере мы используем [<input/text> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/text) для первого поля ввода — значение по умолчанию для данного атрибута. Оно представляет однострочное текстовое поле, которое может принимать любые значения.
* Для второго поля ввода мы используем тип [<input/email> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/email), который представляет собой однострочное текстовое поле, которое принимает в качестве значения корректно составленный e-mail адрес. Он делает простое текстовое поле "умным", позволяя проверять введённые пользователем данные на корректность. Также это позволяет открывать более подходящие для ввода e-mail адреса клавиатуры (например, с символом @ при базовой раскладке) на устройствах с динамической клавиатурой, таких как смартфоны. Вы найдёте более подробную информацию про валидацию форм далее в статье [Валидация формы](https://developer.mozilla.org/ru/docs/Learn/Forms/Form_validation).

Последнее, но не менее важное, обратите внимание на разницу синтаксиса у HTML-элементов <input> и <textarea></textarea>. Это одна из странностей HTML. Тег <input> — это пустой элемент, то есть он не нуждается в закрывающем теге.  [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea) — это непустой элемент, что говорит о том, что ему необходим закрывающий тег. Это важно при использовании одного из свойств форм: определения значения по умолчанию. Для определения начального значения для HTML-элемента [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input) вам необходимо использовать атрибут value следующим образом:

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

Если вы хотите определить значение по умолчанию для HTML-элемента [<textarea> (en-US)](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/textarea), вам просто нужно поместить это начальное значение между открывающим и закрывающим тегами:

<textarea>

по умолчанию в этом элементе находится этот текст

</textarea>

1. [**Элемент**](https://developer.mozilla.org/ru/docs/Learn/Forms/Your_first_form#%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_htmlelementbutton)[**<button>**](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button)

Разметка нашей формы почти готова, но нам ещё необходимо добавить кнопку, которая позволит пользователю отправлять или "представлять" информацию после заполнения формы. Это делается с помощью HTML-элемента [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button). Необходимо добавить следующий код перед закрывающим тегом </form>:



HTML-элемент [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button) также принимает атрибут type, который может быть равен одному из трёх значений: submit, reset или button.

* Клик по кнопке submit (значение по умолчанию) отправляет данные из формы на страни<цу, определённую в атрибуте action элемента [<form>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form).
* Клик по кнопке reset сбрасывает значения всех элементов управления формы к их начальному значению. С точки зрения UX, это считается плохой практикой.
* Клик по кнопке button не делает ничего! Звучит странно, но на самом деле это очень удобно использовать для создания собственных кнопок — вы можете определить их поведение через JavaScript.

**Примечание**: вы также можете использовать HTML-элемент [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input) с соответствующим атрибутом type , чтобы создать кнопку:  <input type="submit">. Главным преимуществом HTML-элемента [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button) в сравнении с элементом [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input) заключается в том, что [<input>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/Input) может принимать в себя только простой текст, в то время как [<button>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/button) позволяет использовать весь HTML для создания более стилизованного текста внутри кнопки.

**(Дополнительная информация)** [**Отправка данных на сервер**](https://developer.mozilla.org/ru/docs/Learn/Forms/Your_first_form#%D0%BE%D1%82%D0%BF%D1%80%D0%B0%D0%B2%D0%BA%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_%D0%BD%D0%B0_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80)

Последняя и, наверно, самое сложное — это обработка данных формы на стороне сервера. HTML-элемент [<form>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/form) определяет куда и каким способом отправить данные благодаря атрибутам action и method.

Мы определяем имя name для каждого виджета формы. Указание имён важно как для браузера, так и для сервера: браузер узнаёт, какие имена дать каждой части данных, а сервер может получить эти данные, обратясь к ним по заданному имени. Данные форму отправляются на сервер в виде пары имя/значение.

Чтобы переименовать данные, вам необходимо использовать атрибут name на каждом виджете формы, который будет собирать определённую часть информации. Давайте взглянем на код нашей формы ещё раз:



В нашем примере форма отправит три куска данных с именами "user\_name", "user\_email" и "user\_message". Эти данные будут отправлены на URL "/my-handling-form-page" через метод [HTTP POST](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/POST).

На стороне сервера скрипт, расположенный на URL "/my-handling-form-page" получит данные в виде списка из 3 элементов вида ключ/значение, содержащихся в HTTP-запросе. То, как скрипт будет обрабатывать данные, зависит от вас. Каждый язык серверного программирования (PHP, Python, Ruby, Java, C# и т.д.) имеет свой механизм обработки данных из формы.

**ЗАДАНИЕ 2**

1. [**Элементы <fieldset> и <legend>**](https://developer.mozilla.org/ru/docs/Learn/Forms/How_to_structure_a_web_form#%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%8B_%3Cfieldset%3E_%D0%B8_%3Clegend%3E)

Элемент [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset) - это удобный способ стилистической и семантической группировки элементов формы. Вы можете установить заголовок [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset), добавив элемент [<legend>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/legend) сразу после открывающего тега [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset). Текст элемента [<legend>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/legend) формально описывает назначение содержимого [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset).

Различные вспомогательные технологии будут использовать [<legend>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/legend) как часть метки label всех элементов внутри [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset). Например, такие экранные дикторы как [Jaws](http://www.freedomscientific.com/products/fs/jaws-product-page.asp) или [NVDA](http://www.nvda-project.org/) произносят заголовок формы [<legend>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/legend) перед произношением названия меток элементов.

Небольшой пример:



Читая эту форму, экранный диктор произнесёт "Fruit juice size small" для первого элемента, "Fruit juice size medium" - для второго, "Fruit juice size large" - для третьего.

Вариант использования в этом примере является одним из наиболее важных. Каждый раз, когда у вас есть набор переключателей, вам нужно поместить их внутри [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset). Также [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset) можно использовать для разделения формы. В идеале, длинную форму разделяют на несколько страниц, однако, если она должна находиться на одной странице, распределение связанных элементов в разные [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset) может повысить удобство использования.

Из-за своего влияния на вспомогательные технологии элемент [<fieldset>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/fieldset) является одним из ключевых элементов для построения доступных форм; однако вы не должны им злоупотреблять. Если возможно, старайтесь проверять, как [экранный диктор](https://www.nvaccess.org/download/) интерпретирует вашу форму.

1. [**Элемент <label>**](https://developer.mozilla.org/ru/docs/Learn/Forms/How_to_structure_a_web_form#%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%3Clabel%3E)

Это самый важный элемент для построения доступных форм — при правильной реализации скринридеры будут озвучивать текст-подсказку вместе со связанными элементами. Посмотрите на этот пример из предыдущей статьи:

<label for="name">Name:</label> <input type="text" id="name" name="user\_name">

При правильно связанном элементе <label> с элементом <input> через атрибуты for и id соответственно (атрибут for ссылается на атрибут id соответствующего виджета формы), скринридер прочтёт вслух что-то наподобие "Name, edit text".

Если <label> не правильно установлен, скринридер прочитает это как "Edit text blank", что не несёт в себе никакой уточняющей информации, позволяющей понять предназначение данного текстового поля.

Обратите внимание на то, что виджет формы может быть вложен в элемент [<label>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label), как на примере:

<label for="name">

Name: <input type="text" id="name" name="user\_name">

</label>

Однако даже в таких случаях лучше всё равно указывать атрибут for, так как некоторые вспомогательные технологии не распознают неявную связь между текстами-подсказками и виджетами.

1. [**Лейблы тоже кликабельны!**](https://developer.mozilla.org/ru/docs/Learn/Forms/How_to_structure_a_web_form#%D0%BB%D0%B5%D0%B9%D0%B1%D0%BB%D1%8B_%D1%82%D0%BE%D0%B6%D0%B5_%D0%BA%D0%BB%D0%B8%D0%BA%D0%B0%D0%B1%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B!)

Ещё одно преимущество при правильно установленных текстах-подсказках заключается в том, что по ним можно кликнуть для активации связанных с ними виджетов. Это поддерживается во всех браузерах. Это удобно как для текстовых полей ввода, в которых устанавливается фокус при клике на текст-подсказку, так и для радио-кнопок и чекбоксов — область попадания такого элемента управления может быть очень маленькой, поэтому полезно сделать её как можно больше.

Например:



1. [**Несколько лейблов**](https://developer.mozilla.org/ru/docs/Learn/Forms/How_to_structure_a_web_form#%D0%BD%D0%B5%D1%81%D0%BA%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE_%D0%BB%D0%B5%D0%B9%D0%B1%D0%BB%D0%BE%D0%B2)

На самом деле вы можете добавить несколько текстов-подсказок на один виджет формы, но это не очень хорошая идея, так как у некоторых вспомогательных технологий могут возникнуть проблемы с обработкой такой структуры. Вместо использования нескольких лейблов, лучше вложить виджет формы внутрь одного элемента [<label>](https://developer.mozilla.org/ru/docs/Web/HTML/Element/label).

Рассмотрим этот пример:



Параграф на первой строке примера описывает правило для обязательных элементов. Вначале необходимо убедиться, что вспомогательные технологии, такие как программы чтения с экрана, отображают или озвучивают их пользователю, прежде чем он найдёт требуемый элемент. Таким образом они будут знать, что означает звёздочка. Программа чтения с экрана будет произносить звёздочку как «звёздочку» или «обязательно», в зависимости от настроек программы чтения с экрана — в любом случае, первый абзац даёт понимание того, что будет означать звёздочка далее в форме.

* В первом примере лейбл не будет прочитан вместе с текстовым полем — получится лишь "edit text blank" и отдельно читаемые тексты-подсказки. Множественные элементы <label> могут быть неправильно интерпретированы программой чтения с экрана.
* Второй пример немного лучше — лейбл будет прочитан вместе с текстовым полем и будет звучать как "name star name edit text", однако тексты-подсказки всё ещё разделены. Это всё ещё немного сбивает с толку, но на этот раз ситуация немного лучше, потому что с текстовое поле связано с текстом-подсказкой.
* Третий пример — лучший, так как весь лейбл будет связан с текстовым полем и озвучен целиком, а при чтении текст будет звучать как "name star edit text".

**ЗАДАНИЕ 3**. Создайте HTML-документ по предложенному образцу:

