**Конспект «Формы. Знакомство». Раздел 1**

Чтобы создать форму, нужно использовать парный тег <form>, внутри которого размещаются поля формы. У тега <form> есть два важных атрибута:

* action задаёт URL, адрес для отправки формы;
* method задаёт метод отправки формы.

Пример:

<form action="https://echo.htmlacademy.ru/courses" method="get">

*поля формы*

</form>

Для отправки формы обычно используют методы get или post. Если не указать атрибут method, то будет использован get.

Метод get посылает данные формы в строке запроса, то есть они видны в адресной строке браузера и следуют после знака вопроса.

Метод post посылает данные в теле HTTP-запроса и используется, когда нужно отправить много данных и ссылка на результат обработки этих данных не нужна.

**Текстовое поле ввода**

Большинство полей форм создаётся с помощью одиночного тега <input>. У этого тега два обязательных атрибута:

* type задаёт тип поля;
* name задаёт имя поля.

От типа поля зависит то, как оно будет отображаться и вести себя. Самый распространённый тип — это text, который обозначает *текстовое поле*. Данный тип используется по умолчанию.

Для того, чтобы правильно обработать данные на сервере нужно имя поля. Обычно, имя поля должно быть уникальным в пределах формы. Для задания имени поля используют латинские буквы и цифры.

**Идентификатор и значение по умолчанию**

Атрибут id поля ввода обозначает *идентификатор*. Он должен быть уникальным на всей странице.

Обычно идентификаторы используют для повышения удобства работы с формой, например, создают подписи, связанные с мелкими полями. При нажатии на такие подписи активируется связанное поле. Также идентификаторы используют в JavaScript для работы с полями.

Идентификатор не передаётся на сервер. Лучше использовать идентификаторы, отличающиеся от имени поля.

Атрибут value задаёт значение поля ввода по умолчанию.

**Подпись для поля ввода**

Парный тег <label> создаёт подпись к полю формы.

Он связывает текст и поле ввода логически. Если нажать на текст в такой подписи, то курсор переместится в соответствующее поле.

Первый способ создать подпись — просто обернуть текст подписи и тег поля в тег <label>, вот так:

<label>

Подпись <input type="text" name="username">

</label>

При оборачивании текста в тег <label> он визуально никак не меняется.

Связь подписи и поля по id

Можно связать подпись с полем с помощью атрибута id. Алгоритм такой:

1. Добавляем к полю ввода идентификатор с помощью атрибута id;
2. Оборачиваем текст подписи в тег <label>;
3. Добавляем тегу <label> атрибут for;
4. В атрибут for записываем такое же значение, что и в атрибуте id у поля.

Например:

<label for="*user-field-id*">Имя пользователя</label>

...

*много-много других тегов*

...

<input id="*user-field-id*" type="text" name="username">

**Поле ввода пароля**

Чтобы сделать поле для ввода пароля, в котором текст будет отображаться «звёздочками», нужно изменить значение атрибута type на password.

**Кнопка отправки формы**

Кнопка для отправки формы создаётся с помощью тега <input> c типом submit.

Надпись на кнопке можно задать с помощью атрибута value. Для кнопки отправки формы задавать имя необязательно, но если оно задано, то на сервер будут отправляться имя и значение кнопки. Обычно имя для кнопки отправки задают, когда в форме несколько кнопок, отвечающих за разные действия.

**Многострочное поле ввода**

Многострочное текстовое поле создаётся с помощью парного тега <textarea>. У него есть атрибуты name и id, аналогичные атрибутам текстового поля.

Атрибут rows принимает целочисленное значение и задаёт высоту многострочного поля в строках. Атрибут cols задаёт ширину поля в символах. В качестве ширины символа берётся некоторая «усреднённая ширина».

Атрибут value у многострочного поля отсутствует. Текст, расположенный между открывающим и закрывающим тегом <textarea> является значением по умолчанию.

**Чекбокс**

Чекбокс — это тег <input> с типом checkbox.

Он работает по принципу «либо да, либо нет». Если галочка стоит, то браузер посылает переменную с именем поля на сервер, если галочки нет, то не посылается ничего. Атрибут value не является обязательным.

Чтобы поле было отмечено по умолчанию, нужно добавить к тегу атрибут checked.

**Имена чекбоксов в рамках одной формы должны быть разными, так как чекбокс не подразумева**

# Конспект «Формы. Знакомство». Раздел 2

## Радиокнопка

Радиокнопка — это тег <input> с типом radio.

Значение радиокнопки задаётся c помощью value. Атрибут value является для радиокнопок обязательным. Браузер отправляет на сервер значение value выбранной радиокнопки.

## Группа радиокнопок

Если добавить ещё несколько <input> с таким же именем, но другими значениями value, то мы получим группу радиокнопок. Имя поля у радиокнопок одной группы должно быть одинаковым, но идентификаторы всегда должны быть уникальными.

Чтобы сделать какой-либо вариант выбранным по умолчанию, нужно добавить к соответствующему тегу <input> атрибут checked.

## Раскрывающийся список, или «селект»

Раскрывающийся список позволяет выбрать один вариант ответа из нескольких. Такой список создаётся с помощью парного тега <select>, у которого есть атрибуты name и id.

Варианты ответов задаются с помощью парных тегов <option>, которые должны располагаться внутри тега <select>. Например:

<select name="theme">

<option value="light">Светлая тема</option>

<option value="dark">Тёмная тема</option>

...

</select>

В атрибуте value тега <option> задаётся значение варианта ответа, а внутри этого тега располагается подпись варианта ответа.

Если при отправке формы у выбранного варианта задан value, то на сервер отправится значение этого атрибута. В противном случае будет отправлен текст подписи.

## «Мультиселект»

Раскрывающийся список можно превратить в так называемый «мультиселект», то есть список, в котором можно выбрать несколько вариантов. Для этого нужно добавить к тегу <select> атрибут multiple.

Высоту мультиселекта можно изменять с помощью атрибута size тега <select>.

Чтобы отметить как выбранные по умолчанию одно или несколько значений, нужно к соответствующим тегам <option> добавить атрибут selected.

При отправке данных мультиселекта на сервер с PHP после имени в значении атрибута name ставятся символы квадратных скобок []. Например, <select name="days[]">. Это необязательное требование для имени мультиселекта, оно нужно только для корректной обработки данных в PHP.

## Поле для загрузки файлов

Поле для загрузки файлов — это тег <input> с типом file. Для этого поля обязательным атрибутом является имя.

Чтобы поле заработало и браузер смог передать выбранный файл на сервер, необходимо добавить всей форме атрибут enctype со значением multipart/form-data. Этот атрибут указывает браузеру, в каком виде пересылать данные. Если вы хотите отправить на сервер файл, данные из формы эффективнее всего передавать по частям. За это и отвечает значение multipart/form-data.

Внешний вид полей с типом file очень сильно отличается в зависимости от операционной системы и очень плохо изменяется с помощью стилей.

## Скрытое поле

Скрытое поле используют, когда в форме нужно отправить какие-то дополнительные служебные данные, которые не вводятся пользователем.

Скрытое поле — это тег <input> с типом hidden.

ет выбор одного элемента из нескольких.

**Конспект «Формы. Погружение»**

**Простая кнопка**

Самая простая кнопка создаётся с помощью тега <input> c типом button. При нажатии на такую кнопку никаких действий не происходит, а все необходимые действия обычно задаются при помощи JavaScript.

Пример записи:

<input type="button" value="Кнопка">

**Сброс введенных значений**

В HTML-формах есть специальная кнопка, которая сбрасывает введённые значения и возвращает изначально установленные. Это поле ввода с типом reset.

Пример использования:

<input type="reset" value="Сбросить">

Кнопка возвращает те значения, которые были установлены в полях формы по умолчанию.

**Кнопка-изображение**

В качестве кнопки отправки формы можно использовать изображение. Для этого у тега input нужно указать тип image.

У кнопки-изображения есть еще два атрибута:

* src задаёт адрес изображения;
* alt задаёт альтернативный текст, отображаемый в том случае, если изображение не загружено.

Кнопка-изображение работает аналогично кнопке submit, но на сервер дополнительно передаются координаты точки, по которой был произведен щелчок.

**Альтернативный способ задания кнопок**

Помимо тега <input> для добавления кнопок можно использовать тег <button>. Внутри тега <button> можно размещать любые HTML-элементы, в том числе изображения. Например:

<button>Календарь <img src="calend.png" alt=""></button>

По умолчанию значение атрибута type у тега <button> — submit.

Значениями атрибута type также могут быть button и reset.

Кнопка button со значением type="reset" сбрасывает значения полей формы к изначальным.

Значение type="button" избавит кнопку от всей изначальной функциональности.

**Обязательные поля**

Чтобы указать, что поле обязательно для заполнения, нужно добавить ему пустой атрибут required:

<input type="text" required>

При попытке отправить форму с незаполненными обязательными полями браузер выведет всплывающее предупреждение. Эта проверка работает на клиентской части и упрощает валидацию форм. Но всегда нужно проверять отправленные данные и на стороне сервера.

**Поле выбора даты и времени**

Для выбора даты из календаря существует новый тип поля ввода — date. При клике на данное поле в форме всплывает календарик.

Пример записи:

<input type="date">

Для указания времени существуют дополнительные «временные» типы полей, например, time для выбора времени.

<input type="time">

Также существуют следующие типы полей:

* datetime-local задаёт дату с указанием времени (без учета временной зоны);
* week задаёт порядковый номер недели в году и года;
* month задаёт месяц и год.

Если браузер не поддерживает указанный тип поля, то вместо него отображается обычное текстовое поле.

**Список возможных значений**

Для текстовых полей можно заранее определить список возможных значений, которые отображаются, когда вы начинаете вводить текст в поле. Для этого существует специальный тег <datalist>.

Пример использования:

<input type="text" list="browsers" name="browser">

<datalist id="browsers">

<option value="Firefox"></option>

<option value="Chrome"></option>

<option value="Safari"></option>

</datalist>

Связывание текстового поля и списка осуществляется при помощи атрибута list у тега input — значение list должно быть таким же, как значение атрибута id у списка.

Если тег input имеет специфический тип, например email или другие, то в списке отображаются только корректные для данного типа значения.

**Поле ввода числового значения**

Для ввода числовых значений существует специальный тип поля ввода number. Рядом с полем браузер автоматически подставляет две стрелочки для увеличения и уменьшения числового значения.

Пример записи:

<input type="number">

Верхняя и нижняя границы числовых значений задаются при помощи вспомогательных атрибутов min и max. А атрибут step устанавливает величину шага изменения значения.

**Поле поиска**

Поле поиска задаётся при помощи тега <input> с типом search.

Пример записи:

<input type="search">

В некоторых браузерах внутри него появляется крестик для сброса введённого значения.

**Автофокус**

Для того, чтобы отметить поле, в которое по умолчанию нужно установить курсор используется пустой атрибут autofocus.

Пример записи:

<input type="text" autofocus>

Такой атрибут должен быть только один на странице.

**Выбор из диапазона**

Для выбора значения из диапазона подходит тип поля range.

Такое поле выглядит как шкала с ползунком и позволяет выбрать число из некоторого интервала значений.

Пример записи:

<input type="range" min="1" max="10">

Атрибуты min и max устанавливают нижнюю и верхнюю границу допустимых значений, а атрибут step устанавливает величину шага изменения значения.

**Область для вывода результата**

Тег <output> представляет собой область, куда выводятся какие-либо результаты вычислений, обычно полученные при помощи JavaScript.

Пример записи:

<output name="sum">[значение по умолчанию]</output>

Значение по умолчанию при этом можно не задавать, тогда область вывода будет пустой.

**Группировка полей формы**

Для того, чтобы зрительно отделить одни поля от других используется тег fieldset.

Пример:

<fieldset>

<input type="text">

</fieldset>

<fieldset>

<textarea></textarea>

</fieldset>

По умолчанию браузеры отображают результат в виде рамки вокруг этой группы полей, но при помощи CSS можно изменить его внешний вид.

Для каждой группы можно добавить её заголовок. Для этого внутрь тега fieldset надо поместить тег legend:

<fieldset>

<legend>Заголовок группы</legend>

<input type="text">

</fieldset>

**Паттерны значений полей**

Для создания поля, принимающего данные определённого формата, используется атрибут pattern, в котором с помощью регулярного выражения описывается требуемый формат.

**Поле ввода телефона**

За ввод телефонных номеров отвечает тип поля tel. В такое поле полезно добавлять атрибут pattern, чтобы избежать ошибки при вводе данных.

В мобильных браузерах при фокусе на такое поле появляется клавиатура, позволяющая вводить только цифры и символы телефонных номеров.

**Подсказка при заполнении полей**

Для добавления подсказки к полю формы используется специальный атрибут placeholder:

<input type="text" placeholder="Текст подсказки">

Текст подсказки выводится внутри текстового поля, а при вводе значения — автоматически убирается.

**Поля ввода адресов сайтов и email**

В HTML5 добавлены два типа полей email и url, предназначенные для ввода электронной почты и адреса сайта. Эти поля автоматически проверяют формат введённых данных.

Пример записи:

<input type="email">

<input type="url">

**Поле выбора цвета**

В HTML5 добавили новый тип color, предназначенный для полей выбора цвета. При клике на такое поле появляется окно с возможностью выбрать цвет из палитры.

Пример записи:

<input type="color">

Если браузер не поддерживает поле для выбора цвета, то вместо него отображается обычное текстовое поле.

**Группировка элементов списка**

В теге select можно объединять option в группы.

Для формирования группы используется тег optgroup. Атрибут label этого тега определяет заголовок группы.

Пример использования:

<select name="variants">

<optgroup label="Группа вариантов 1">

<option value="1">Вариант 1</option>

<option value="2">Вариант 2</option>

<option value="3">Вариант 3</option>

</optgroup>

<optgroup label="Группа вариантов 2">

<option value="4">Вариант 4</option>

<option value="5">Вариант 5</option>

<option value="6">Вариант 6</option>

</optgroup>

</select>

Вложенность групп не ограничена, внутрь каждой группы можно вложить другие группы.

Аналогично можно группировать элементы и в списках со множественным выбором.

**Запрет редактирования полей**

Для того, чтобы сделать поле недоступным для редактирования есть два способа: использование атрибута readonly и использование атрибута disabled

Атрибут readonly не дает пользователю изменять поле (вводить новый текст, модифицировать существующий). Введенное значение можно выделить и скопировать. Данные из этого поля отправляются на сервер.

Атрибут disabled не дает пользователю изменять поле (вводить новый текст, модифицировать существующий). Нельзя поставить фокус в это поле, введенное значение нельзя выделять и копировать. Данные из этого поля НЕ отправляются на сервер.

**Управление автозаполнением полей**

Параметрами автозаполнения можно управлять используя атрибут autocomplete. Он может принимать два значения on и off. Первое включает автозаполнение, второе — отключает.

Пример использования:

<input type="text" autocomplete="off">

Значение по умолчанию зависит от настроек браузера.

**Переключение между полями**

При нажатии клавиши Tab браузер передает управление (фокус) от одного элемента к другому в том порядке, в котором они были объявлены на странице. Этим порядком можно управлять при помощи атрибута tabindex.

Пример записи:

<input type="text" tabindex="3">

В качестве значения может использоваться любое целое положительное число. Значения выстраиваются последовательно и переход между элементами происходит от меньшего значения к большему. Если представлено отрицательное значение — элемент может быть выделен, однако не участвует в последовательной навигации

Если представлен 0 — элемент может быть выделен и достигнут с помощью последовательной навигации, однако порядок навигации определён платформой

Из соображений доступности не рекомендуется менять порядок навигации по полям ввода по умолчанию.

**localStorage**

Если данные из формы по каким-то причинам не переданы на сервер, они могут пропасть. Справиться с этой довольно часто встречающейся проблемой поможет localStorage.

Суть localStorage заключается в том, что в него можно записывать данные, которые будут сохраняться в браузере.

Можно сохранять данные из формы в хранилище при работе с формой, а при загрузке страницы проверять хранилище на наличие данных, и если они есть, то подставлять их в форму. Таким образом можно предотвратить потерю данных при работе с формами.

<script>

if (window.localStorage) {

var elements = document.querySelectorAll('[name]');

for (var i = 0, length = elements.length; i < length; i++) {

(function(element) {

var name = element.getAttribute('name');

element.value = localStorage.getItem(name) || element.value;

element.onkeyup = function() {

localStorage.setItem(name, element.value);

};

})(elements[i]);

}

}

</script>