# 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.

# Сброс введенных значений

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

# 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.

# Обязательные поля

Чтобы указать, что поле обязательно для заполнения, нужно добавить ему пустой атрибут **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.

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>**.

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.

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**.

# Автофокус

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

# 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.

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.

# Группировка полей формы

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

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

# 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**

# 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.

# 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.

# 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.

# 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**.

# 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.