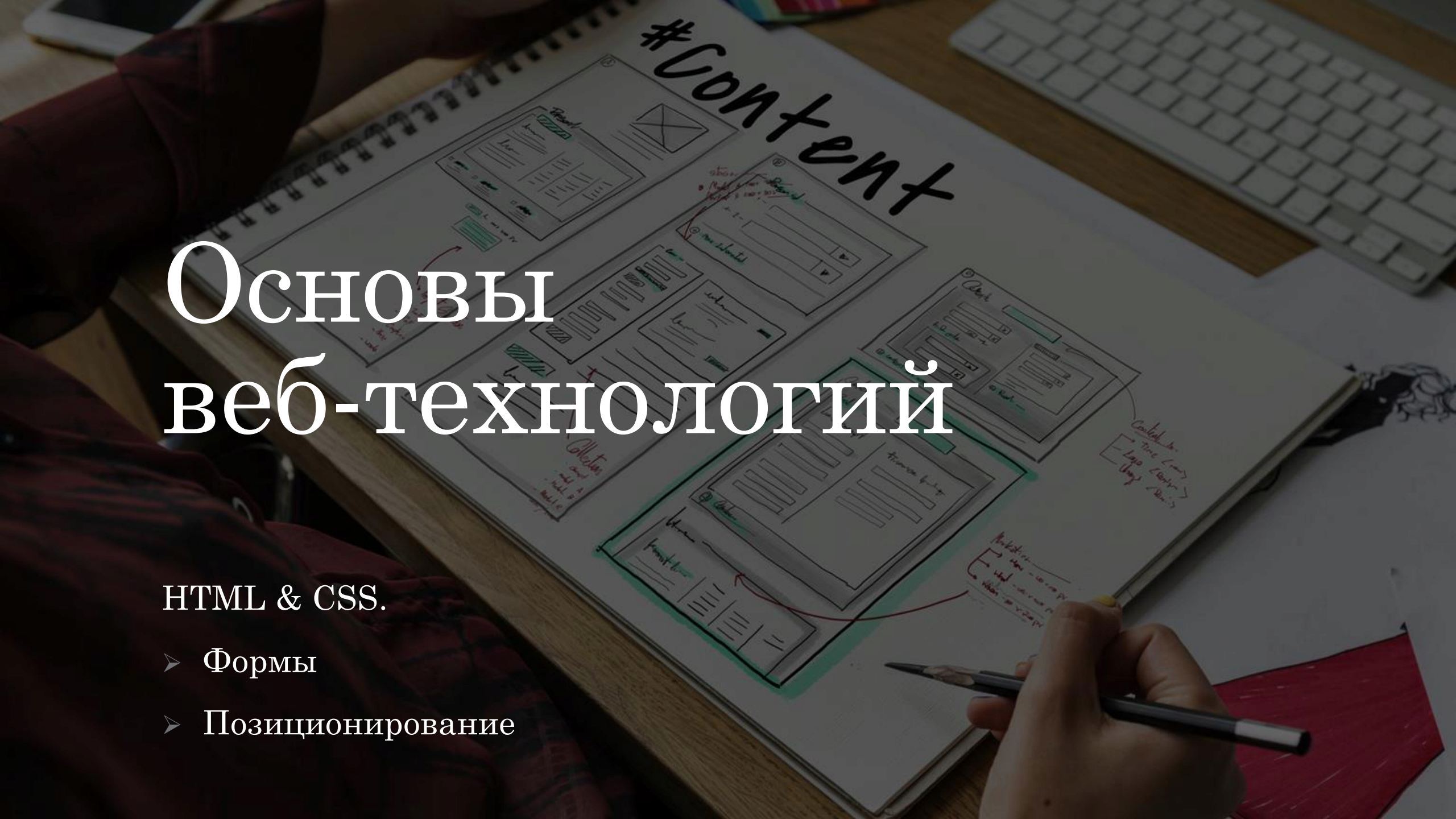

Основы веб-технологий

HTML & CSS.

- Формы
- Позиционирование

Иконки

Иконки Font Awesome: https://fontawesome.com/icons

После регистрации сервис бесплатно выдаёт персональную ссылку на набор компонентов и инструкцию, как внедрить его в веб-проект или приложение.

Также отдельные иконки можно включать напрямую в html как svg или скачать и вставлять как svg-файл.

Иконки Google https://fonts.google.com/icons

При выборе иконки справа появляется инструкция по использованию.

Аналогично отдельные иконки можно включать напрямую в html как svg или скачать и вставлять как svg-файл или png.

Иконки Flaticon https://www.flaticon.com/

Что доступно бесплатно: можно скачать бесплатно любую иконку, но только в формате PNG. Размер неограничен: выбирайте любой от 16 до 512 пикселей. Но есть лимит на количество скачиваний в день: не более десяти иконок или только одна коллекция.

Есть набор бесплатных svg иконок.

Формы HTML

Форма — это компонент веб-страницы с элементами управления, такими как текстовые поля, кнопки, флажки, диапазон или поле выбора цвета. Пользователь может взаимодействовать с такой формой, предоставляя данные, которые затем могут быть отправлены на сервер для дальнейшей обработки.

Тег `<form>` устанавливает форму на веб-странице. Форма предназначена для обмена данными между пользователем и сервером. Область применения форм не ограничена отправкой данных на сервер, с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению.

Документ может содержать любое количество форм, но одновременно на сервер может быть отправлена только одна форма. По этой причине данные форм должны быть независимы друг от друга.

Для отправки формы на сервер используется кнопка `Submit`, того же можно добиться, если нажать клавишу Enter в пределах формы. Если кнопка `Submit` отсутствует в форме, клавиша `Enter` имитирует ее использование.

Отправка формы серверу осуществляется чаще всего в виде запросов HTTP GET или POST.

Атрибуты элемента form

`action` – указывает url-адрес обработчика формы. Когда форма отправляется, данные в форме преобразуются в структуру в соответствии с указанным типом кодировки enctype, а затем отправляются в место, указанное action, с использованием метода отправки данных method.

`method` – указывает HTTP-метод для отправки формы.

- post — данные формы включаются в тело HTTP-запроса. Метод является более надежным и безопасным, чем get и не имеет ограничений по размеру.
- get — данные формы (пара имя-значение) добавляются в url-адрес с помощью разделителя ? и отправляются на сервер. Данный способ имеет ограничения по размеру отправляемых данных и не подходит для отправки паролей и конфиденциальной информации.

`name` – задает имя формы, которое будет использоваться для доступа к элементам формы через сценарии. Значение должно быть уникальным среди элементов формы в коллекции форм, в которой оно находится, если таковые имеются.

`enctype` – указывает MIME-тип данных формы для отправки на сервер только в случае method="post".

- application/x-www-form-urlencoded — значение по умолчанию. Данные формы кодируются как пары имя-значение, аналогично строке запроса URI.
- multipart/form-data — данные формы не кодируются. Это значение необходимо использовать для формы, содержащей элементы, управляющие загрузкой файлов.
- text/plain — символы не кодируются, а пробелы заменяются на символ +. Полезен только для отладки.

Атрибуты элемента form

`accept-charset` – определяет кодировку символов, которая должна использоваться для отправки формы.

`novalidate` – логический атрибут, который указывает, что форма не должна проверяться при отправке.

`autocomplete` – указывает, может ли значение элементов <input> автоматически заполняться браузером.

- off — пользователь должен явно вводить значение в это поле для каждого использования.

on — браузер может автоматически дополнять значение на основе значений, которые пользователь ввел во время предыдущих использований.

`target` - указывает, в каком окне выводить результат после отправки формы.

- _blank — загружает ответ в новое окно/вкладку.
- _self — загружает ответ в то же окно (значение по умолчанию).
- _parent — загружает ответ в родительский фрейм. Если родителя нет, этот параметр ведет себя так же, как _self.
- _top — загружает ответ во весь экран. Если родителя нет, этот параметр ведет себя так же, как _self.

Примеры

```
<!-- Простая форма, которая пошлёт GET запрос -->
<form action="url" >
    <label for="GET-name">Name:</label>
    <input id="GET-name" type="text" name="name">
    <input type="submit" value="Save">
</form>

<!-- Простая форма, которая пошлёт POST запрос -->
<form action="url" method="post">
    <label for="POST-name">Name:</label>
    <input id="POST-name" type="text" name="name">
    <input type="submit" value="Save">
</form>

<!-- Форма с fieldset, legend, и label -->
<form action="url" method="post">
    <fieldset>
        <legend>Title</legend>
        <input type="radio" name="radio" id="radio"> <label for="radio">Click me</label>
    </fieldset>
</form>
```

Основные элементы управления form

label: используется для вывода подписей для элементов управления формы

input: в зависимости от значения атрибута type данный элемент управления может использоваться для различных целей

button: кнопка, при нажатии которой могут выполняться различные действия в зависимости от значения атрибута **type**

select: элемент управления для выбора одного или нескольких вариантов из списка;

textarea: элемент управления, используемый для ввода фрагмента текста, состоящего из одной или более строк

fieldset: группирует наборы элементов управления по определенному критерию

Элемент <label>

Элемент <label> представляет надпись к элементу управления формы. Элемент может быть связан с конкретным полем формы с помощью атрибута for, либо путем помещения элемента управления формы внутрь <label>.

```
<div>
    <label for="firstname">Укажите ваше имя</label>
    <input type="text" id="firstname" name="firstname">
</div>
<div>
    <label>
        Укажите вашу фамилию
        <input type="text" name="secondname">
    </label>
</div>
```

Элемент <input>

Элемент <input> является одним из разносторонних элементов формы и позволяет создавать разные элементы интерфейса и обеспечить взаимодействие с пользователем. Главным образом <input> предназначен для создания текстовых полей, различных кнопок, переключателей и флажков.

Из-за огромного количества возможных сочетаний типов ввода и атрибутов это один из самых мощных и сложных элементов HTML.

Атрибут **type** устанавливает тип элемента <input>. В зависимости от типа могут дополнительно устанавливаться управляющие атрибуты.

Атрибут **name** определяет имя поля. Это имя передается вместе со значением при отправке данных формы. Если имя не указано или имя пусто, значение поля не отправляется вместе с формой. Не должно совпадать с именем типа поля.

Атрибут **value** — это

- строка, которая определяет текущее редактируемое значение для полей типа text и password;
- для полей типа button, reset и submit — текст на кнопке;
- для полей типа checkbox, radio и hidden — определяет связанное значение, которое отправляется на сервер.

Элемент <input>: type ="text"

Однострочное текстовое поле. Переносы строк автоматически удаляются из входного значения.

```
<div>
    <label for="username">Имя пользователя:</label>
    <input id="username" name="login" placeholder="login" maxlength="20" size="30"
autocomplete="off" >
</div>
```

Элемент <input>

`type="search"`
Однострочное текстовое поле для ввода строк поиска, разрывы строк автоматически удаляются из вводимого значения. Некоторые браузеры отображают иконку удаления — крестик, которую можно использовать для очистки поля.

`type="tel"`
Поле для ввода номера телефона. Отображает клавиатуру телефона на некоторых устройствах с динамической клавиатурой.

`type="url"`
Поле для ввода URL. Выглядит как поле для ввода текста, но имеет параметры проверки и соответствующую клавиатуру для поддержки браузеров и устройств с динамической клавиатурой.

`type="email"`
Поле для ввода адреса электронной почты. Выглядит как ввод текста, но имеет параметры проверки и соответствующую клавиатуру для поддержки браузеров и устройств с динамической клавиатурой.

`type="password"`
Однострочное текстовое поле, в котором вводимые пользователем символы заменяются на звездочки, маркеры, либо другие, установленные браузером значки. Предупредит пользователя, если сайт небезопасен.

Элемент <input>

`type="date"`
Элемент управления, открывает средство выбора даты (год, месяц и день, без времени).

`type="month"`
Элемент управления, открывает средство выбора месяца и года.

`type="week"`
Элемент управления, открывает средство выбора номера недели и года.

`type="time"`
Элемент управления, позволяет вводить время в 24-часовом формате по шаблону чч:мм.

```
<label>Выберите время доставки: <input type=time min="11:00" max="21:00" step="900"></label>
```

`type="datetime-local"`
Элемент управления, позволяет вводить дату и время по шаблону дд.мм.гггг чч:мм.

`type="number"`
Поле для ввода целочисленных значений. Атрибуты min, max и step задают верхний, нижний пределы и шаг между значениями соответственно. Эти атрибуты предполагаются у всех элементов, имеющих численные показатели, а их значения по умолчанию зависят от типа элемента.

Валидация форм на стороне клиента

Перед отправкой данных на сервер важно убедиться, что все обязательные поля формы заполнены данными в корректном формате. Это называется валидацией на стороне клиента и помогает убедиться, что данные, введённые в каждый элемент формы, соответствуют требованиям.

Валидация на стороне клиента — это первичная проверка введённых данных, которая существенно улучшает удобство взаимодействия с интерфейсом; обнаружение некорректных данных на стороне клиента позволяет пользователю немедленно их исправить. Если же проверка происходит только на сервере, процесс заполнения может быть более трудоёмким, так как требует повторения одних и тех же действий отправки данных на сервер для получения обратного ответа с сообщением о том, что нужно исправить.

Любые данные, отправляемые через форму, необходимо дополнительно проверять на безопасность и на стороне сервера, поскольку валидацию на стороне клиента достаточно просто обойти и она может не остановить злоумышленников.

Типы валидации на стороне клиента

Существует два типа валидации на стороне клиента, с которыми вы столкнётесь в Интернете:

- Встроенная валидация форм использует функционал валидации HTML5.
- JavaScript-валидация кодируется с помощью JavaScript.

Встроенная валидация форм выполняется с помощью атрибутов валидации у элементов формы:

- required: определяет , что для отправки формы данное поле предварительно должно быть заполнено.
- minlength и maxlength: задаёт минимальную и максимальную длину текстовых данных (строк)
- min и max: задаёт минимальное и максимальное значение для поля, расчитанного на числовой тип данных
- type: определяет тип данных, на который рассчитано поле: число, email-адрес или какой-то другой предустановленный тип
- pattern: с помощью регулярного выражения, определяет шаблон, которому должны соответствовать вводимые данные.

Селектор псевдокласса (для форм)

:valid — поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;

:invalid — поля формы, содержимое которых не соответствует указанному типу данных;

:enabled — все активные поля форм;

:disabled — заблокированные поля форм, т.е., находящиеся в неактивном состоянии;

:in-range — поля формы, значения которых находятся в заданном диапазоне;

:out-of-range — поля формы, значения которых не входят в установленный диапазон;

:not(селектор) — элементы, которые не содержат указанный селектор — класс, идентификатор, название или тип поля формы — :not([type="submit"]);

:checked — выделенные (выбранные пользователем) элементы формы.

Стилизация

```
<form action="url" method="get" >
    <div>
        <label for="username">Имя пользователя:</label>
        <input type="text" id="username" name="login" placeholder="login" maxlength="20" size="30" autocomplete="off" pattern="[A-Za-z0-9]{3,15}" required>
    </div>
    <div>
        <label for="email">Почта:</label>
        <input type="email" id="email" name="email"  maxlength="25" size="30" placeholder="username@domain.ru" required>
    </div>
    <input type="submit" value="Отправить данные">
</form>
```

```
form{
    max-width: 320px;
    border: 1px solid grey;
    padding: 10px;
}
label{
    display: block;
    padding: 5px;
    font-weight: bold;
}
```

```
input[type=text], input[type=email]{
    width: 100%;
    border: 1px solid grey;
    margin-bottom: 10px;
    padding: 3px;
    font-size: 1.2em;
    box-sizing: border-box;
    background-color: #eee;
}
input:invalid {
    border: 1px solid red;
    background-color: pink;
}
```

Элемент <input>: type="hidden"

Создает элемент управления, который не отображается, но значение которого отправляется на сервер.

```
<input type="hidden" name="hidden_param" value="1234567">
```

Элемент <input>: type="checkbox"

Отображает флажок, позволяющий выбирать/отменять выбор отдельных значений. Для каждой **группы чекбоксов** указывается своё **имя**, по которому, в дальнейшем, возможно отделить группы чекбоксов при обработке на сервере.

```
<form action="url" method="get" >
    <label>
        <input type="checkbox" name="languages" value="HTML">
        Хочу изучать HTML
    </label>
    <label>
        <input type="checkbox" checked name="languages" value="CSS">
        Хочу изучать CSS
    </label>
    <label>
        <input type="checkbox" name="languages" value="JS">
        Хочу изучать JS
    </label>
    <input type="submit">
</form>
```

Пример стилизации checkbox

```
<form action="url"  method="get" >
    <label>
        <input type="checkbox" name="languages" value="HTML" class="checkbox-1">
        <span>Хочу изучать HTML</span>
      </label>
      <br>
      <label>
        <input type="checkbox" checked name="languages" value="CSS" class="checkbox-1">
        <span>Хочу изучать CSS</span>
      </label>
      <br>
      <label>
        <input type="checkbox" name="languages" value="JS"  class="checkbox-1">
        <span>Хочу изучать JS</span>
      </label>
      <br>
</form>
```

```
.checkbox-1{
    appearance: none;
    width: 1em;
    height: 1em;
    border: 1px solid red;
    border-radius: 3px;
    position: relative;
    top: 4px;
    transition: 0.2s all linear;
}
.checkbox-1:checked{
    background-color: coral;
}
```

Элемент <input>: type="radio"

Для создания переключателей, которые умеют обрабатывать только один из множества вариантов, существуют радиокнопки. Название они получили от старых автомобильных радиоприёмников, в которых выбор радио осуществлялся нажатием на одну из множества кнопок для выбора частоты.

```
<form action="url" method="get">
    <label>
        <input type="radio" name="languages" value="HTML">
        Хочу изучать HTML
    </label>
    <br>
    <label>
        <input type="radio" checked name="languages" value="CSS">
        Хочу изучать CSS
    </label>
    <br>
    <label>
        <input type="radio" name="languages" value="JS">
        Хочу изучать JS
    </label>
    <br>
    <input type="submit">
</form>
```

Элемент <input>: type="submit" | "reset" | "button"

`type="submit"`
Отображает кнопку отправки формы.

`type="reset"`
Создает кнопку, которая сбрасывает содержимое формы к значениям по умолчанию.

`type="button"`
Создает кнопку без поведения по умолчанию, надписью к кнопке является значение атрибута value, по умолчанию пустое.

```css
input[type=submit]{
    appearance: none;
    padding: 5px;
    margin: 5px;
    border: none;
    background-color: bisque;
    border-radius: 5px;
    box-shadow: 0 2px 0 orange;
    cursor: pointer;
}
input[type=submit]:hover{
    box-shadow: none;
    transform: translateY(2px);
}
```

Элемент <input>: type="file"

Поле для выбора одного или нескольких файлов из хранилища своего устройства. После выбора файлы можно загрузить на сервер с помощью отправки формы или манипулировать ими с помощью кода JavaScript. Атрибут `multiple` элемента `input` позволяет пользователю выбрать несколько файлов.

```
<form action="url" method="post" enctype="multipart/form-data">
    <div>
        <label for="profile_pic">Выберите фотографию</label>
        <input type="file" id="profile_pic" name="profile_pic" accept=".jpg, .jpeg, .png">
    </div>
    <div>
        <input type="submit" value="Загрузить">
    </div>
</form>
```

Элемент <input>: type="range"

Элемент управления для ввода числа, точное значение которого не важно. Отображается как виджет диапазона со средним значением по умолчанию. Используется вместе с min и max для определения диапазона допустимых значений.

Для каждого <input type="range"> задается:

- min - минимальное значение ползунка;
- max - максимальное значение ползунка;
- step - шаг изменения ползунка;
- value - начальное значение ползунка.

```
<label for ="price-range"> Ползунок </label>
<input type="range" name="price" id="price-range" min="50" max="200" step="2" value="70" >
```

Элемент <button>

Элемент `<button>` создает на веб-странице кнопки и по своему действию напоминает результат, получаемый с помощью тега `<input>` (с атрибутом `type="button | reset | submit")`. В отличие от этого тега, `<button>` предлагает расширенные возможности по созданию кнопок. Например, на подобной кнопке можно размещать любые элементы HTML, в том числе изображения. Используя стили можно определить вид кнопки путем изменения шрифта, цвета фона, размеров и других параметров.

```
<button type="button" onclick="alert('Привет, я кнопочка!')">Кнопочка</button>

<button type="submit"> <img src="image.png"  width="50px"> Кнопочка с картинкой </button>
```

Элемент <select>

В различных формах пользователю часто приходится выбирать один из множества вариантов. Это могут быть категории, по которым мы хотим произвести поиск, выбор различных опций для поиска. Наиболее распространённым решением является использование выпадающих списков.

Для создания такого выпадающего списка используется тег `<select>` с вложенными внутри него тегами `<option>`. Всё это похоже на создание обычных списков, где вместо ul/ol используется `<select>`, а вместо `<li>` используется `<option>`.

Часто первый пункт списка используется для заголовка всего выпадающего списка. В таком случае для него используют атрибут `disabled`, чтобы заблокировать его для выбора.

```
<form action="url" method="get">
    <select name="languages">
        <option disabled>Какой курс вы хотите пройти?</option>
        <option value="js">JS</option>
        <option value="ruby">Ruby</option>
        <option value="python">Python</option>
        <option value="java">Java</option>
        <option value="html">HTML</option>
        <option value="css">CSS</option>
    </select>
    <button type="submit">Жми</button>
</form>
```

Элемент <select>

Атрибут `multiple` – логический атрибут, который указывает, что в списке можно выбрать несколько параметров. Большинство браузеров будут отображать окно списка с прокруткой вместо выпадающего списка с одной строкой.

Атрибут `size` – если для элемента `<select>` указан атрибут `multiple`, этот атрибут представляет количество строк в списке, которые должны быть видны одновременно. Значение по умолчанию — 0.

```
<form action="url" method="get">
    <select name="languages" multiple size=5 >
        <option disabled>Какой курс вы хотите пройти?</option>
        <option value="js">JS</option>
        <option value="ruby">Ruby</option>
        <option value="python">Python</option>
        <option value="java">Java</option>
        <option value="html">HTML</option>
        <option value="css">CSS</option>
    </select>
    <button type="submit">Жми</button>
</form>
```

Элемент <select>: optgroup

Элемент `<optgroup>` представляет собой группу элементов `<option>` с общей надписью, обычно выделенной жирным шрифтом. Браузеры отображают такие группы как связанные друг с другом, отдельно от других элементов `<option>`.

```
<form action="url" method="get">
    <select name="languages">
        <option disabled>Какой курс вы хотите пройти?</option>
        <option value="js">JS</option>
        <option value="ruby">Ruby</option>
        <option value="python">Python</option>
        <option value="java">Java</option>
        <optgroup label="HTML & CSS">
            <option value="html">HTML</option>
            <option value="css">CSS</option>
        </optgroup>
      </select>
      <button type="submit">Жми</button>
</form>
```

Элемент <textarea>

Поле `<textarea>` представляет собой элемент формы для создания области, в которую можно вводить несколько строк текста. В отличие от тега `<input>` в текстовом поле допустимо делать переносы строк, они сохраняются при отправке данных на сервер.

Между тегами `<textarea>` и `</textarea>` можно поместить любой текст, который будет отображаться внутри поля.

```
<form action="url" method="get">
    <label for="msg">Форма для ввода сообщения</label>
    <br>
    <textarea placeholder="Сообщение...." name="msg" id="msg" cols="30" rows="5"></textarea>
    <br>
    <input type="submit" value="Отправить">
</form>
```

Атрибуты:

- cols ширина поля в символах.
- disabled блокирует доступ и изменение элемента.
- maxlength максимальное число введенных символов.
- placeholder указывает замещающийся текст.
- readonly устанавливает, что поле не может изменяться пользователем.
- required обязательное для заполнения поле.
- rows высота поля в строках текста.

Элемент <textarea>. Пример стилизации

```
<textarea placeholder="Сообщение...." name="msg" id="msg"></textarea>
```

```
textarea::placeholder {
    font-size: 1.5rem;
    color: red;
}
textarea {
    font-size: 1.2rem;
    width: 100%;
    max-width: 900px;
    height: 300px;
    min-height: 200px;
    max-height: 500px;
    resize: vertical;
    background-color: #f5f5f5;
    color: blue;
    box-shadow: inset 0 0 4px rgba(0, 0, 0, 0.4);
}
textarea:focus {
    outline: none;
}
```

Группировка элементов формы

Элемент `<fieldset>...</fieldset>` предназначен для группировки элементов, связанных друг с другом, разделяя таким образом форму на логические фрагменты.

Каждой группе элементов можно присвоить название с помощью элемента `<legend>`, который идет сразу за открывающим тегом элемента `<fieldset>`. Название группы проявляется слева в верхней границе `<fieldset>`. Например, если в элементе `<fieldset>` хранится контактная информация:

Если атрибут `disabled` присутствует, то группа связанных элементов формы, находящихся внутри контейнера `<fieldset>`, отключены для заполнения и редактирования. Используется для ограничения доступа к некоторым полям формы, содержащих ранее введенные данные.

```
<form action="url" method="get">
    <fieldset>
        <legend>Контактная информация</legend>
        <div>
            <label for="name">Имя</label>
            <input type="text" name="name" id="name">
        </div>
        <div>
            <label for="email">E-mail</label>
            <input type="email" name="email" id="email">
        </div>
      </fieldset>
      <input type="submit" value="Отправить">
</form>
```

CSS-верстка

- Нормальный поток
- Методы позиционирования
- Обтекание
- Макет таблицы
- Grid Layout
- Flexible Box Layout
- Макет с несколькими столбцами

Нормальный поток

Поток — одно из важнейших базовых понятий в вёрстке. Это принцип организации элементов на странице при отсутствии стилей: если мы напишем HTML и не напишем CSS, то отображение в браузере будет предсказуемо благодаря тому, что мы абсолютно точно знаем, как браузер располагает элементы в потоке.

Если вообще не применять никаких стилей, браузер формирует из элементов нормальный поток. Поведение блочных элементов в нормальном потоке отличается от поведения строчных.

Блочные элементы в нормальном потоке располагаются друг под другом, всегда занимая всю доступную ширину родителя. Высота блочного элемента по умолчанию равна высоте его содержимого. Три абзаца, идущие друг за другом в HTML, будут располагаться точно в таком же порядке и на странице.

Строчные элементы располагаются друг за другом, как слова в предложении. В зависимости от направления письма в конкретном языке элементы могут располагаться слева направо (например, в русском языке), справа налево (как, например, в иврите) и даже сверху вниз (как иероглифы и знаки слоговых азбук в японском вертикальном письме). Ширина и высота строчного элемента равна ширине и высоте содержимого. В отличие от блочного элемента, мы не можем управлять шириной и высотой строчного элемента через CSS. Несколько строчных элементов будут стремиться уместиться на одной строке, насколько хватает ширины родителя. Если ширины родителя не хватает, то лишний текст строчного элемента переносится на следующую строку.

Нормальный поток

В нормальном потоке:

- Вывод элементов на страницу браузера осуществляется в том порядке, в котором они следуют в HTML коде.
- В коде элементы вложены друг в друга, и чтобы это учитывать при выводе используют так называемые **воображаемые слои для отображения элементов**. При этом слой элемента тем выше (ближе к нам), чем данный элемент является более вложенным в коде, т.е. глубже расположен в нём.
- Положение элемента в потоке зависит от значения свойства display.

Все элементы по умолчанию находятся в нормальном потоке. Но это поведение можно поменять при помощи некоторых CSS-свойств. При изменении значений этих свойств элемент перестаёт взаимодействовать с остальными блоками в потоке. Говорят, что он «вышел из потока». **Элементом «вне потока» может быть плавающий, абсолютно позиционированный или корневой элемент.**

Тут нужно отметить, что элементы, вышедшие из потока, создают внутри себя своего рода мини-поток. Их дочерние элементы будут подчиняться правилам взаимодействия в потоке в пределах родителя.

CSS position

CSS-свойство `position` определяет положение элемента на веб-странице. Существует несколько типов позиционирования: `static`, `relative`, `absolute`, `fixed`, `sticky`, `initial` и `inherit`.

- `static` – это значение свойства по умолчанию. Блок располагается в соответствии с нормальным потоком. Сдвиги блока не применяются.
- `relative` – элемент позиционируется относительно своего стандартного положения.
- `absolute` – элемент позиционируется абсолютно по отношению к первому позиционированному родителю.
- `fixed` – элемент позиционируется относительно окна браузера.
- `sticky` – элемент позиционируется на основе позиции прокрутки пользователя.

Containing block (содержащий блок)

Положение и размер блока элемента может вычисляться относительно некоторого прямоугольника, называемого содержащим блоком элемента (containing block).

Для некорневого элемента с `position: static;` или `position: relative;` содержащий блок формируется краем области содержимого ближайшего родительского блока уровня блока, ячейки таблицы или уровня строки.

Содержащим блоком элемента с `position: fixed;` является окно просмотра.

Для некорневого элемента с `position: absolute;` содержащим блоком устанавливается ближайший родительский элемент со значением `position: absolute/relative/fixed` следующим образом:

- если предок — элемент уровня блока, содержащим блоком будет область содержимого плюс поля элемента `padding`;
- если предок — элемент уровня строки, содержащим блоком будет область содержимого;
- если предков нет, то содержащий блок элемента определяется как блок, в котором находится корневой элемент.

Для «липкого» блока содержащим блоком является ближайший предок с прокруткой или окно просмотра в противном случае.

Относительное позиционирование

`position: relative;`

Положение блока рассчитывается в соответствии с нормальным потоком. Затем блок смещается относительно его нормального положения и во всех случаях, включая элементы таблицы, не влияет на положение любых следующих блоков. Тем не менее, такое смещение может привести к перекрытию блоков, а также к появлению полосы прокрутки в случае переполнения.

Относительно позиционированный блок сохраняет свои размеры, включая разрывы строк и пространство, первоначально зарезервированное для него.

Относительно позиционированный блок создает новый содержащий блок для абсолютно позиционированных потомков.

Относительно позиционированный элемент ведёт себя как элемент в потоке за исключением того, что его текущее положение можно при помощи определённых CSS свойств сместить. К этим CSS свойствам относятся `left, top, right и bottom`.

Относительное позиционирование

Например, для того чтобы элемент сдвинуть вверх или вниз относительного его исходного положения к нему нужно применить CSS свойство top или bottom:

```
position: relative;
/* для сдвига элемента вверх на 10px */
top: -10px; /* или bottom: 10px; */
/* для сдвига элемента вниз на 10px */
top: 10px; /* или bottom: -10px; */
```

Если одновременно установить top и bottom, то будет применено значение top, т.к. оно является более приоритетным, чем bottom.

Для сдвига элемента вправо или влево используется CSS свойство left или right:

```
position: relative;
/* для сдвига элемента влево на 20px */
left: -20px; /* или right: 20px; */
/* для сдвига элемента вправо на 20px */
left: 20px; /* или right: -20px; */
```

Если одновременно установить left и right, то приоритетным будет значение, находящееся в left:

Для сдвига по двум осям нужно использовать top или bottom, и left или right.

Относительное позиционирование

```
<div>  Текст 1 </div>
<div class="relative-1">  Текст 2 </div>
<div>  Текст 3  </div>
<div class="relative-2">  Текст 4 </div>
```

```
div {
    border: 1px solid #aaa;
    background: #9f9;
    font-size: 2em;
}
```

```
.relative-1{
    background: #9ff;
    position: relative;
    left: 20px;
    bottom: 10px;
}

.relative-2{
    background: #9ff;
    position: relative;
    left: 50%;
    top: 30px;
}
```

Абсолютное позиционирование (absolute)

Установка абсолютного позиционирования элементу осуществляется посредством задания ему `position: absolute`.

Этот тип позиционирования позволяет разместить элемент именно там, где вы хотите.

Позиционирование выполняется относительно ближайшего позиционированного родителя.

Под позиционированным элементом понимается элемент с position, равным `relative, absolute, fixed.`

Абсолютно позиционированный блок создает новый содержащий блок для дочерних элементов нормального потока и потомков с `position: absolute`;.

Содержимое абсолютно позиционированного элемента не может обтекать другие блоки. Абсолютно позиционированный блок могут скрывать содержимое другого блока (или сами могут быть скрыты), в зависимости от значения z-index перекрывающихся блоков.

Абсолютное позиционирование (absolute)

Абсолютное позиционирование делает две вещи:

- Элемент исчезает с того места, где он должен быть и позиционируется заново. Остальные элементы, располагаются так, как будто этого элемента никогда не было.
- Координаты top/bottom/left/right для нового местоположения отсчитываются от ближайшего позиционированного родителя, т.е. родителя с позиционированием, отличным от static. Если такого родителя нет – то относительно документа.

Кроме того:

- Ширина элемента с position: absolute устанавливается по содержимому.
- Элемент получает display: block, который перекрывает почти все возможные display.
- CSS-свойства для управления положением абсолютно позиционированного элемента работают по-другому чем с position: relative. Установить ширину (высоту) абсолютно позиционированному можно с помощью установки ему двух координат top и bottom (left и right).
- Если элементу одновременно установить top, bottom и height, то предпочтение будет отдано top и height.

Абсолютное позиционирование

```
<div>
    <p>Заголовок в правом-верхнем углу документа.</p>
    <h2>Заголовок</h2>
    <p>Сюда мы разместим разный текст</p>
    <p>И сюда мы тоже разместим разный текст</p>
</div>
```

```
div {
    background-color: beige;
    width: 500px;
    position: relative;
}

h2 {
    background-color: lightgreen;
    margin: 0;
    position: absolute;
    right: 0;
    top: 0;
}
```

Стилизация checkbox

```
<input type="checkbox" name="cb" class="checkbox-2">
```

```
.checkbox-2 {/* основная фиолетовая часть */
    appearance: none;
    background-color: #dfe1e4;
    border-radius: 72px;
    border-style: none;
    height: 20px;
    width: 30px;
    margin: 0;
    position: relative;
}
.checkbox-2::after { /* добавляем псевдоэлемент белый кружок */
    content: "";
    background-color: #fff;
    border-radius: 50%;
    height: 14px;
    width: 14px;
    position: absolute;
    top: 3px;
    left: 3px;
    transition: all 100ms ease-out;
}
.checkbox-2:checked { /* меняем цвет фона у отмеченного checkbox */
    background-color: #6e79d6;
}
.checkbox-2:checked::after { /* сдвигаем белый кружок влево у отмеченного checkbox */
    left: 13px;
}
```

Фиксированное позиционирование (fixed)

Задание элементу фиксированного позиционирования осуществляется посредством установки ему `position: fixed`.

Фиксированное позиционирование аналогично абсолютному позиционированию, с отличием в том, что для содержащим блоком устанавливается окно просмотра. Такой блок полностью удаляется из потока документа и не имеет позиции относительно какой-либо части документа. Фиксированные блоки не перемещаются при прокрутке документа. В этом отношении они похожи на фиксированные фоновые изображения.

При печати фиксированные блоки повторяются на каждой странице, содержащим блоком для них устанавливается область страницы. Блоки с фиксированным положением, которые больше области страницы, обрезаются.

Фиксированное позиционирование

```
<body>
    <a href="#" class="a-top-fixed"><img src="arrow-up.png" width="32px" alt="Наверх"></a>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aspernatur mollitia veniam porro deserunt placeat hic fuga quas dignissimos, vitae iusto sunt sint, illum unde odit adipisci suscipit! Illum, fuga tempore!
    </p>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aspernatur mollitia veniam porro deserunt placeat hic fuga quas dignissimos, vitae iusto sunt sint, illum unde odit adipisci suscipit! Illum, fuga tempore!
    </p>
</body>
```

```
.a-top-fixed {
    position: fixed;
    bottom: 0px;
    right: 0px;
}
```

Липкое позиционирование (sticky)

Элемент с `position: sticky` «прилипает» к экрану при прокрутке, пока не встретится с границей родительского блока.

«Липкий» контейнер — это HTML-элемент, который оборачивает «липкий» элемент. Это максимальная область, в которой может перемещаться «липкий» элемент.

Когда задается элементу position: sticky, его родитель автоматически становится «липким» контейнером. Контейнер будет являться областью видимости для элемента. «Липкий» элемент не может выйти за пределы своего «липкого» контейнера.

Липкое позиционирование (sticky)

```html
<body>
  <div class="wrap">
        <div class="block c2"></div>
        <div class="block c1"></div>
        <div class="block c1"></div>
  </div>
  <div class="wrap">
        <div class="block c2"></div>
        <div class="block c1"></div>
        <div class="block c1"></div>
  </div>
</body>
```

```css
.wrap{
  border: 1px solid grey;
}
.block{
  width: 200px;
  height: 300px;
  border: 1px solid black;
  margin: 5px;
}
.c1{
  margin-left: 225px;
  background-color: tomato;
}
.c2{
  position: sticky;
  top: 20px;
  background-color: blue;
}
```