

## Практическая работа №4 «Свойство `overflow`. Псевдоэлементы CSS. Единицы измерения в CSS. Элементы формы в HTML. Элемент `<iframe>`. Медиа-элемент `<video>`»

### 3.9 Свойство `overflow`

Свойство CSS `overflow` контролирует большой контент. Оно сообщает, следует ли обрезать содержимое или добавить полосы прокрутки, когда содержимое не помещается в поле родительского элемента (переполняется) в горизонтальном и/или вертикальном направлении.

`overflow` может включать в себя следующие значения:

- `visible`
- `hidden`
- `scroll`
- `auto`
- `inherit`

`visible`: по умолчанию; переполнение контента не обрезается и отображается за пределами поля элемента.

`hidden`: переполнение контента обрезается, а оставшая часть содержимого невидима.

`scroll`: содержимое обрезается и браузер использует элементы прокрутки, не важно было ли обрезано содержимое или нет. Полоса прокрутки может быть горизонтальной или вертикальной.

`auto`: зависит от агента пользователя; автоматически добавляет полосу прокрутки (как и `scroll`), если контент переполняется, но не добавляет ее, если содержимое подходит.

`inherit`: наследование значения от родительского элемента.

#### 3.9.1 Свойства `overflow-x` и `overflow-y`

Свойства `overflow-x` и `overflow-y` указывают, как изменить переполнение элементов. Параметр `overflow-x` работает только с осью `x` или слева направо. `Overflow-y` работает по оси `y` или сверху вниз.

К основным значениям свойства относятся:

- `visible`

- `hidden`
- `clip`
- `scroll`
- `auto`

### 3.9.2 Свойство `overflow-wrap`

Свойство `overflow-wrap` в CSS используется для указания того, что браузер может разбивать строки текста внутри любого целевого элемента, чтобы предотвратить переполнение, когда исходная строка слишком длинная и не помещается.

Значения свойства `overflow-wrap`:

- `normal`
- `break-word`
- `anywhere`

`normal`: строки будут разрываться в соответствии с исходными правилами разрыва строк, т.е. будет происходить переполнение контента без переноса (листинг 3.9.1, 3.9.2 и рисунок 3.9.1).

`break-word`: то же, что и значение `anywhere`, при этом обычно неразрывные слова могут быть разбиты в произвольных точках, если в строке нет других приемлемых точек разрыва, но возможности мягкого переноса, предоставляемые разрывом слова, НЕ учитываются при вычислении внутренних размеров минимального содержимого (листинг 3.9.1, 3.9.2 и рисунок 3.9.1).

`anywhere`: чтобы предотвратить переполнение, неразрывная строка символов (например, длинное слово или URL-адрес) может быть разорвана в любой точке, если в строке нет других приемлемых точек разрыва. В точке останова (breakpoint) не вставляется символ переноса. Возможности мягкого переноса, предоставляемые разрывом слова, учитываются при расчете внутренних размеров минимального содержимого.

Листинг 3.9.1 – Применение свойства overflow-wrap (CSS)

```
<head>
  <title>
    CSS overflow-wrap
  </title>
  <style>
    h1 {
      color: blue;
      margin-left: 260px;
    }
    div {
      width: 100px;
      outline: 1px solid #bbb;
      display: inline-block;
    }
    #div1 {
      overflow-wrap: normal;
      margin: 220px;
      margin-top: 0px;
      margin-right: 100px;
    }
    #div2 {
      overflow-wrap: break-word;
    }
    span {
      color: red;
    }
    code {
      font-size: larger;
      font-weight: bold;
    }
    #code1 {
      color: orange;
    }
    #code2 {
      color: blue;
    }
  </style>
</head>
```

Листинг 3.9.2 – Применение свойства overflow-wrap (HTML)

```
<body>
  <h1>overflow-wrap</h1>
  <div id="div1">
    <strong>#div1</strong>: Здесь для примера приведено длинное слово
    <span>"серобуромалиновый"</span> для применения значения <code id="code1">normal</code> свойства <code id="code2">overflow-wrap</code>.
  </div>
  <div id="div2">
    <strong>#div2</strong>: Здесь для примера приведено длинное слово
    <span>"серобуромалиновый"</span> для применения значения <code id="code1">break-word</code> свойства <code id="code2">overflow-wrap</code>.
  </div>
</body>
```

## overflow-wrap

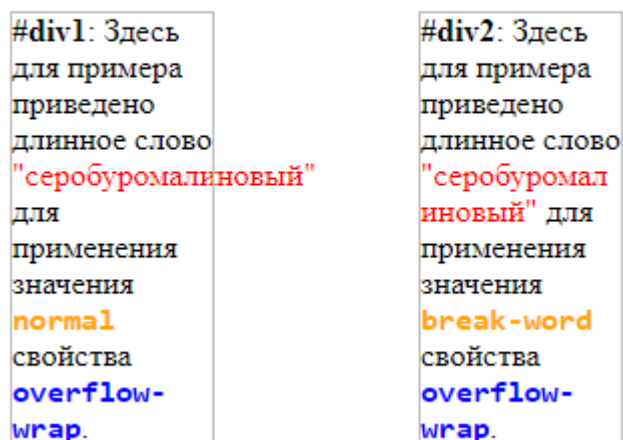


Рисунок 3.9.1 – Результат применения значений normal и break-word свойства overflow-wrap

## 4 Псевдоэлементы CSS

**Псевдоэлемент в CSS** – это ключевое слово, добавляемое к селектору, которое позволяет стилизовать определённую часть выбранного элемента. Псевдоэлементов не существует в HTML-разметке. Они создаются и позиционируются исключительно при помощи CSS. Чаще всего используются для создания различных декоративных элементов (которые не несут содержательного смысла). Также псевдоэлементы приходят на помощь, когда нужно наложить поверх картинки так называемый оверлей (перекрывающий слой).

**Синтаксис:**

```
selector::pseudo-element {  
    property: value;  
}
```

**В селекторе можно использовать только один псевдоэлемент.** Он должен находиться после простых селекторов в выражении.

**Примечание:** как правило, следует использовать двойное двоеточие (::) вместо одинарного (:). В этом состоит различие между псевдоклассами и псевдоэлементами. Однако, так как это различие не присутствовало в старых версиях спецификации W3C, большинство браузеров поддерживают оба синтаксиса для псевдоэлементов.

К стандартным псевдоэлементам относятся (таблица 4.1):

Таблица 4.1 – Стандартные псевдоэлементы CSS

Псевдоэлементы	Описание
::before	По умолчанию располагается перед содержимым элемента, для которого задаётся.
::after	По умолчанию располагается после содержимого элемента, для которого задаётся.
::first-letter	Выбирает первую букву в строке или абзаце текста.
::first-line	Выбирает первую строку текста. <i>Сработает только для блочных элементов.</i>
::selection	Управляет стилем текста, который пользователь выделяет при помощи мыши. Например, выделяет текст желтым цветом.

#### Продолжение таблицы 4.1

<code>::placeholder</code>	Позволяет стилизовать подсказку, которая выводится в поле ввода текста элемента <code>&lt;input&gt;</code> с атрибутом <code>placeholder</code> .
<code>::marker</code>	<p>Применяет стили к маркеру элемента списка, который содержит значок или номер. Обычно элементы списка являются элементами HTML <code>&lt;li&gt;</code>, но другие элементы также могут стать элементами списка с помощью <code>display: list-item</code>.</p> <p>Изменяет цвет, размер, интервал и т.д. маркера списка.</p>

#### Свойство `content`

Свойство, с помощью которого можно добавить на страницу то, чего нет в HTML-разметке. Используется для динамического создания содержимого (во время выполнения), т.е. `content` заменяет элемент сгенерированным значением содержимого (таблица 4.2). Он используется для создания контента с псевдоэлементами `::before` и `::after`.

Таблица 4.2 – Основные значения свойства `content`

Значение свойства <code>content</code>	Описание
<code>normal</code>	<p>Значение по умолчанию. Устанавливает содержимое, если оно указано, в обычное значение, которое по умолчанию равно «none» (то есть ничего).</p> <p><i>Нельзя комбинировать с другими значениями.</i></p>
<code>none</code>	<p>Устанавливает содержимое, если оно указано, в «ничего».</p> <p><i>Нельзя комбинировать с другими значениями.</i></p>
<code>counter()</code> <code>counters()</code>	<p>Если значением является результат выполнения функций <code>counter()</code> или <code>counters()</code>, то псевдоэлемент будет содержать вычисленное значение счётчика в текущий момент. Эти функции работают совместно со свойствами <code>counter-reset</code> и <code>counter-increment</code>;</p>
<code>attr()</code>	<p>С помощью функции <code>attr()</code> можно вывести в псевдоэлемент значение любого атрибута тега.</p> <p>HTML:</p>



url	<p>В этом случае содержимое элемента заменяется на это изображение. Может применяться к любому элементу. Нужно помнить о том, что при таком способе мы не можем управлять размером изображения.</p> <p><i>Псевдоэлементы <code>::before</code> и <code>::after</code> не требуются.</i></p> <pre>div {   content: url("http://www.pets.com/cat.png"); }</pre>
-----	---

Подробнее со свойством `content` можно ознакомиться по ссылке: <https://developer.mozilla.org/en-US/docs/Web/CSS/content> (англ.яз.).

## 5 Единицы измерения в CSS

В CSS есть разные единицы измерения. Больше всего известны пиксели, но есть и другие – не такие популярные, но весьма удобные в некоторых случаях.

В этом разделе будут рассмотрены относительные и абсолютные единицы измерения, а также единицы измерения области просмотра (*viewport*-единицы).

### 5.1 Относительные единицы измерения

Размеры в CSS можно указывать не только в абсолютных единицах, таких как пиксели (*px*), пункты (*pt*) или сантиметры (*cm*), но и в относительных – в процентах (%), *em* или *rem*.

**Примечание:** пункты (*pt*), пики (*pc*), сантиметры (*cm*) и миллиметры (*mm*) считаются устаревшими единицами измерения, так как браузер пересчитывает эти значения в пиксели.

В большинстве браузеров по умолчанию установлен размер шрифта 16px. Это значение можно использовать при расчетах (например, 16px равны 1em, 1rem или 100%).

**Единицы относительной длины определяют длину относительно другой длины.** Таблицы стилей, в которых используются относительные единицы измерения, легче масштабировать из одной среды вывода в другую.

Дочерние элементы не наследуют относительные значения, указанные для их родителя; они наследуют вычисленные значения.

Далее, в таблице 5.1, приведены существующие относительные единицы измерения:

Таблица 5.1 – Относительные единицы измерения

Единица измерения	Описание
%	Определяет размеры с точки зрения родительских объектов или текущего объекта в зависимости от значения размера шрифта ( <i>font-size</i> ).
em	Размер шрифта относительно обычного, т. е., если шрифт имеет размер 2em, значит, он в 2 раза больше обычного (текущего) шрифта. Единица <i>em</i> позволяет установить размер шрифта элемента относительно размера шрифта его родителя. Когда размер родительского элемента изменяется, размер дочернего элемента изменяется автоматически.



	При использовании <b>em</b> в других свойствах он зависит от размера шрифта самого элемента. Здесь только первое объявление принимает ссылку родителя.
<b>rem</b>	Основан на значении размера шрифта корневого элемента, которым является элемент <code>&lt;html&gt;</code> . И если элемент <code>&lt;html&gt;</code> не имеет указанного размера шрифта, используется значение браузера по умолчанию 16px. Так что здесь рассматривается только значение корня, а отношения с родительским элементом нет. В отличие от <b>em</b> , здесь размер относителен для всех объявлений, а не только для первого.
<b>ch</b>	Ширина символа «0». В моноширинных шрифтах, где все символы имеют одинаковую ширину, <b>1ch</b> это ширина одного символа. <i><b>Примечание:</b> используется крайне редко, так как <b>em</b> обычно вполне подходит.</i>
<b>ex</b>	x-высота текущего шрифта, измеряется в высоте символа «x» в нижнем регистре. <i><b>Примечание:</b> используется крайне редко, так как <b>em</b> обычно вполне подходит.</i>

**Viewport-единицы** (таблица 5.2) представляют собой процентное отношение к текущей величине области просмотра браузера. От простого выражения в процентах viewport-единицы отличаются тем, что они всегда высчитываются на основе *текущего* размера области просмотра. А размер, выраженный просто в процентах, вычисляется по отношению к родительскому элементу.

Эти значения были созданы, в первую очередь, для поддержки мобильных устройств.

Таблица 5.2 – Единицы измерения области просмотра

Единицы измерения viewport	Описание
<b>vw</b>	1% ширины области просмотра ( <i>viewport</i> )
<b>vh</b>	1% высоты области просмотра ( <i>viewport</i> )
<b>vmin</b>	1% от меньшего размера <b>vw</b> или <b>vh</b> (т. е., если ширина меньше высоты, то <b>vmin</b> рассчитывается от ширины, при этом $1 \text{ vmin} = 1 \text{ vw}$ )
<b>vmax</b>	1% от большего размера <b>vw</b> или <b>vh</b> (т. е., если высота больше ширины, то <b>vmax</b> рассчитывается от высоты, при этом $1 \text{ vmax} = 1 \text{ vh}$ )

## 5.2 Абсолютные единицы измерения

Абсолютные единицы измерения (таблица 5.3) фиксированы по отношению друг к другу и привязаны к некоторому физическому измерению. Они в основном полезны, когда известна среда вывода. Абсолютные единицы состоят из *физических единиц* (**in** (дюймы), **cm**, **mm**, **pt**, **pc**, **Q** – четверть миллиметра) и единица измерения угла зрения **px** (эталонный пиксель).

Все абсолютные единицы длины совместимы, и **px** является их канонической единицей.

Таблица 5.3 – Абсолютные единицы измерения

Единица измерения	Расшифровка	Соотношение с пикселем
<b>cm</b>	сантиметр	$1\text{ cm} = 38\text{ px}$
<b>mm</b>	миллиметр	$1\text{ mm} = 3.8\text{ px}$
<b>pc</b>	типографская пика	$1\text{ pc} = 16\text{ px}$
<b>pt</b>	типографский пункт	$1\text{ pt} = 4/3\text{ px}$
<b>px</b>	пиксель	--

Для *CSS* эти размеры привязаны, либо связывая физические единицы с их физическими измерениями, либо связывая единицу пикселя с эталонным пикселем.

Для *печатных материалов* на обычных расстояниях просмотра единицей измерения должна быть одна из стандартных физических единиц (дюймы, сантиметры и т.д.).

Для *экранных носителей* (в том числе устройств с высоким разрешением), устройств с низким разрешением и устройств с необычным расстоянием просмотра рекомендуется, чтобы главной единицей измерения был пиксель. Для таких устройств рекомендуется, чтобы единица пикселя относилась к целому количеству пикселей устройства, которое наилучшим образом соответствует эталонному пикселю.

**Эталонный пиксель** — это угол обзора одного пикселя на устройстве с плотностью пикселей 96 точек на дюйм и на расстоянии вытянутой руки от читателя. Таким образом, при номинальной длине руки в 28 дюймов угол обзора составляет около 0,0213 градуса. Таким образом, для чтения на расстоянии вытянутой руки 1 пиксель соответствует примерно 0,26 мм (1/96 дюйма).

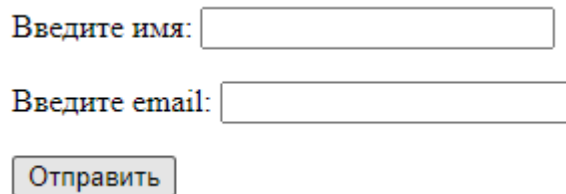
**Подробнее про шрифты** вы можете прочитать в статье по ссылке: <https://learn.javascript.ru/css-units> (русс. яз).

## 6 Элемент <form>

Тег <form> добавляет на страницу форму, которую пользователь может заполнить. Например, ввести своё имя, фамилию или почту (листинг 6, рисунок 6). Данные формы отправляются на сервер.

Листинг 6 – Пример создания формы <form>

```
<form action="" method="get">
  <p>
    <label for="name">Введите имя:</label>
    <input type="text" name="name" id="name" required>
  </p>
  <p>
    <label for="email">Введите email:</label>
    <input type="email" name="email" id="email" required>
  </p>
  <button type="submit">Отправить</button>
</form>
```



Введите имя:

Введите email:

Рисунок 6 – Результат создания формы

Интернет-ресурсы используют формы, чтобы получить какую-то информацию от пользователя. Это может быть форма заказа в онлайн-магазине или форма обратной связи. Пользователь заполняет поля или выбирает нужную опцию в списке, а после отправки формы эти данные можно обработать.

На странице можно сделать сколько угодно форм. Но одновременно пользователь сможет отправить только одну заполненную форму.

Существуют различные элементы формы, которые мы можем использовать, например, текстовые поля, текстовые области, раскрывающиеся списки, флажки, радиокнопки и т.д.

## Атрибуты тега <form>

В таблице 6 представлены основные атрибуты тега <form>.

Таблица 6 – Основные атрибуты тега <form>

Атрибут	Описание атрибута
<code>accept-charset</code>	Задаёт кодировку, в которой сервер принимает данные из формы. Самая распространённая кодировка — UTF-8. Можно указать один вариант или несколько.
<code>action</code>	Указывается ссылка на скрипт, который обработает форму. Это может быть полная URL-ссылка, а может быть относительная, типа <code>html/sendform</code> . Если не указать атрибут <code>action</code> , то страница будет просто обновляться каждый раз, когда отправляется форма.
<code>autocomplete</code>	<p>Включает или выключает автозаполнение для формы. Браузер может подставить данные, которые пользователь сохранил ранее, например, пароль, номер банковской карты или адрес.</p> <p>Атрибут <code>autocomplete</code> можно задать и для конкретных элементов. Есть два значения:</p> <ul style="list-style-type: none"> <li>• <code>on</code> — значение по умолчанию. Включает автозаполнение для этой формы.</li> <li>• <code>off</code> — выключает автозаполнение.</li> </ul>
<code>enctype</code>	Определяет, какой вид кодирования будет применён к данным из формы. Этот атрибут обязательно надо ставить, если через форму отправляются файлы, в остальных случаях — не обязательно.
<code>method</code>	<p>Определяет, каким способом будут отправлены на сервер данные, которые ввёл пользователь. Есть два варианта:</p> <ul style="list-style-type: none"> <li>• <code>get</code> — ответы пользователя дописываются в URL в формате «параметр=значение», например, «email=name@mail.ru». Выглядит это так: <code>site.com/form?name=Max&amp;email=name@mail.ru</code>. То есть параметр — это то, что вы спрашиваете у пользователя, а значение — его ответ. Пары «параметр=значение» разделяются знаком <code>&amp;</code>. Вариант <code>method="get"</code> используется по умолчанию, но у него есть ограничение: URL не должен получиться длиннее, чем 3000 символов.</li> </ul> <p>Если вы отправляете конфиденциальные данные после ввода, то не нужно использовать данное значение метода, потому что их можно будет легко прочитать в запросе, который отправляет форма, даже в адресной строке браузера.</p>

	<ul style="list-style-type: none"> <li>• <code>post</code> — данные из формы пакуются в тело формы и отправляются на сервер. В этом случае нет ограничений по объёму данных, поэтому этот способ подойдёт для заполнения базы данных или отправки файлов.</li> </ul>
<code>name</code>	Уникальное имя формы. Пользователь его не увидит, зато скрипты смогут найти нужную форму.

### 6.1. Элемент `<label>`

Тег `<label>` определяет подпись для элемента `<button>`, `<input>`, `<meter>`, `<output>`, `<progress>`, `<select>` или `<textarea>` (листинг 6.1, рисунок 6.1).

Листинг 6.1 – Создание подписи у чекбокса

```
<div class="form_check">
  <label for="bread">Купить хлеб</label>
  <input type="checkbox" name="bread" id="bread">
</div>
```

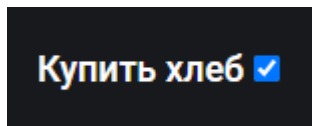


Рисунок 6.1 – Отображение чекбокса с подписью на странице

Тег `<label>` можно использовать двумя способами (листинг 6.1.1):

- Во-первых, используйте тег `<label>`, указав `<input>` и атрибут `id`. Тег `<label>` нуждается в атрибуте `for`, значение которого совпадает с `id` элемента `input`.
- В качестве альтернативы тег `<input>` можно использовать непосредственно внутри тега `<label>`. В этом случае атрибуты `for` и `id` не нужны, поскольку связь является подразумеваемой.

### Листинг 6.1.1 – Два способа создания label у элементов

```
<form action="">
  <label for="phone">Номер телефона:</label>
  <input type="tel" name="phone" id="phone">

  <label><input type="checkbox" name="save">Сохранить</label>
</form>
```

Связывать `<label>` с чекбоксами и радиокнопками необходимо для того, чтобы пользователь мог кликать по тексту подписи, а не целиться в сам чекбокс, так как в такие элементы страницы довольно сложно попасть курсором мыши или пальцем на мобильных устройствах.

## 6.2 Элемент `<input>`

В HTML `input` - поле ввода - может быть указано с помощью того, где пользователь может вводить данные. Тег `<input>` используется в элементе `<form>` для объявления элементов управления вводом, которые позволяют пользователям вводить данные. Поле ввода может быть разных типов в зависимости от типа атрибута. Тег `<input>` — это пустой элемент, который содержит только атрибуты. Для определения меток (*label*) для элемента ввода `input` можно использовать `<label>`.

Тег `<input>` позволяет создавать интерактивные элементы на сайте — поле для ввода текста, кнопка, ползунок, переключатель и другие.

Чтобы даже те, кто использует специальные средства для чтения с экрана (скринридеры), могли точно понять, что делает тот или иной `<input>`, используйте этот тег в паре с `<label>`.

Также, если хочется, чтобы введенные или выбранные в `<input>` данные отправились на сервер, нужно поместить этот тег в `<form>`, либо связать этот элемент с формой через атрибут `id`.

## 6.3 Атрибуты тега `<input>`

К элементу `input` можно применить все глобальные атрибуты: <https://html5book.ru/html-attributes/>.

На листинге 6.3.1 представлен пример создания элемента `<input>` с некоторыми его атрибутами, которые описаны далее по тексту.

### Листинг 6.3.1 – Пример создания элемента input

```
<label for="name">Введите название (от 5 до 11 символов):</label>

<input
  type="text"
  id="name"
  name="name"
  required
  minlength="5"
  maxlength="11"
  size="10"
>
```

#### 6.3.1 Атрибут type

Атрибут `type` определяет, какой вид примет элемент `<input>` и какую функцию будет выполнять.

В таблице 6.3.1, представлены значения атрибута `type` элемента `input`.

Таблица 6.3.1 – Значения атрибута `type` элемента `input`

Атрибут <code>type</code>	
<code>type</code> - тип элемента <code>&lt;input&gt;</code> : текстовое поле, кнопка, флажки выбора и т.д. Если не указать его, то элемент будет обычным текстовым полем <code>text</code> в одну строку.	
Типы для ввода данных разных форматов	Описание значений
<code>email</code>	Поле для ввода e-mail. Браузер проверит, есть ли в нём знак «@» и домен.
<code>number</code>	Ввод числа.
<code>password</code>	Ввод пароля. По мере ввода символы будут меняться на звёздочки <code>***</code> . Можно задать минимальное и максимальное количество символов с помощью атрибутов <code>minlength</code> и <code>maxlength</code> .
<code>search</code>	Поле для поиска.
<code>tel</code>	Поле для ввода номера телефона.
<code>text</code>	Поле для ввода текста в одну строку. Если попробовать вставить текст из нескольких абзацев, то они всё равно сложатся в одну строчку.
<code>url</code>	Поле для ввода URL-адреса.
Типы для кнопок	Описание значений
<code>button</code>	Обычная кнопка.
<code>image</code>	Кнопка «Отправить» в виде картинки.

### Продолжение таблицы 6.3.1

<code>reset</code>	Кнопка, которая сбрасывает всё, что пользователь ввёл в поля текущей формы ранее.
<code>submit</code>	Кнопка для отправки формы.
Типы для ввода дат	Описание значений
<code>date</code>	Ввод даты без времени: год, месяц и дата.
<code>datetime-local</code>	Ввод даты и времени в текущем часовом поясе, когда зоны UTC не указываются.
<code>month</code>	Ввод месяца и года, без указания часового пояса.
<code>time</code>	Ввод времени без указания часового пояса.
<code>week</code>	Ввод номера недели.
Элементы форм	Описание значений
<code>checkbox</code>	Флажки, или чекбокс, с несколькими вариантами выбора.
<code>color</code>	Виджет для выбора цвета. Иногда его называют колорпикер (от английского colorpicker).
<code>file</code>	Выбор файла для отправки.
<code>radio</code>	Круглая кнопка-переключатель для выбора одного из нескольких вариантов.
<code>range</code>	Ползунок для выбора чисел из заданного диапазона.
Отдельный атрибут	Описание значения
<code>hidden</code>	Скрытый ввод не будет виден пользователю, но тем не менее его значение будет отправлено на сервер при отправке формы.

Большое преимущество установки правильного значения для `type` — в мобильных браузерах будет появляться «специальная» клавиатура, упрощающая и улучшающая пользовательский опыт. Так, для `<input type="tel">` у пользователя отобразится клавиатура, содержащая лишь цифры и специальные символы, нужные для корректного ввода номера телефона.

### 6.3.2 Атрибут `value`

В таблице 6.3.2, представлены значения атрибута `value` элемента `input`.

`value` — значение элемента `<input>`, которое отправляется на сервер в формате «имя=значение». Имя задаётся атрибутом `name`, а значение — `value`.

Например, `<input type="radio" name="answer" value="b">`.



Таблица 6.3.2 – Значения атрибута value элемента input

Атрибут value	
Значения атрибута	Описание значений
Надпись на кнопке	Если <code>&lt;input&gt;</code> - кнопка <code>button</code>   <code>reset</code>   <code>submit</code> , то <code>value</code> будет надписью на ней. Значение <code>value</code> для кнопки <code>reset</code> не передаётся на сервер.
Координаты нажатия кнопки-изображения	Если <code>&lt;input&gt;</code> - кнопка в виде картинки <code>image</code> , то <code>value</code> будет передано в виде двух значений координат нажатия относительно изображения <code>имя.x</code> и <code>имя.y</code> , где <code>имя</code> — это значение атрибута <code>name</code> .
Текст в текстовом поле	Если <code>&lt;input&gt;</code> - текстовое поле ( <code>password</code>   <code>text</code> ), то <code>value</code> задаст текст, который появится в этом поле при загрузке страницы. Пользователь сможет его удалить или отредактировать.
Значение по умолчанию для чекбокса, флажка	Если <code>&lt;input&gt;</code> - флажок или переключатель <code>checkbox</code>   <code>radio</code> , то <code>value</code> задаст уникальное значение элемента, по умолчанию <code>on</code> .

### 6.3.3 Другие атрибуты

Прочие атрибуты с их описанием и возможными значениями представлены в таблице 6.3.3.

Таблица 6.3.3 – Другие атрибуты тега input

Атрибут	Описание атрибута
<code>accept</code>	Указание типов файлов, которые принимает сервер.
<code>align</code>	Используется для указания выравнивания <code>image</code> .
<code>alt</code>	Используется для предоставления пользователю альтернативного текста, если он не может просмотреть изображение.
<code>autocomplete</code>	<p>Разрешает автозаполнение. Когда браузер предлагает сохранить, например, пароль или номер банковской карты, чтобы данные подставлялись при следующем входе.</p> <p>Значение атрибута:</p> <ul style="list-style-type: none"> <li><code>on</code> — поле будет автоматически заполняться значением, сохранённым в браузере во время предыдущей отправки формы.</li> <li><code>off</code> — поле не будет заполняться браузером автоматически.</li> </ul>

Продолжение таблицы 6.3.3

<code>autofocus</code>	<i>Атрибут булевого типа (без значения, либо атрибут есть в теге, либо его нет совсем).</i> Фокусируется на элементе <code>&lt;input&gt;</code> сразу после загрузки страницы. Это значит, что пользователю не нужно специально нажимать на этот элемент, чтобы начать в нём что-то писать. В фокусе может быть только один элемент на странице. Но это не может быть элемент типа <code>hidden</code> .
<code>checked</code>	<i>Атрибут булевого типа.</i> Указывает, что элемент должен быть предварительно выбран (проверен) при загрузке страницы. Атрибут <code>checked</code> можно использовать с <code>&lt;input type=«checkbox»&gt;</code> и <code>&lt;input type=«radio»&gt;</code> .
<code>dirname</code>	Используется для указания направления текста.
<code>disabled</code>	<i>Атрибут булевого типа.</i> Отключает элемент. Пользователь не сможет, например, ввести текст или выбрать нужный параметр, а сам элемент будет отображаться в сером цвете, как неактивный.
<code>form</code>	Связывает отдельно стоящий элемент <code>&lt;input&gt;</code> с формой. Для этого укажите в качестве значения имя идентификатора, который задан нужному тегу <code>&lt;form&gt;</code> . Например, так <code>form="special-form"</code> . Этот атрибут позволяет разместить <code>&lt;input&gt;</code> в любом месте на странице за пределами <code>&lt;form&gt;</code> . Если его не указать, элемент <code>&lt;input&gt;</code> будет связан с ближайшей формой.
<code>formaction</code>	Указание URL-адреса файла, который будет обрабатывать элемент управления вводом при отправке формы (для <code>submit</code> и <code>image</code> ).
<code>formenctype</code>	Используется для указания того, как данные формы должны быть закодированы при отправке на сервер (для <code>submit</code> и <code>image</code> ).
<code>formmethod</code>	Определение HTTP-метода для отправки данных на URL-адрес действия (для <code>submit</code> и <code>image</code> ).
<code>formnovalidate</code>	Используется для определения того, что элементы формы не должны проверяться при отправке.
<code>formtarget</code>	Используется для указания, где отображать ответ, полученный после отправки формы (для <code>submit</code> и <code>image</code> ).
<code>height</code>	Указание ширины элемента <code>&lt;input&gt;</code> (только для <code>image</code> ).

Продолжение таблицы 6.3.3

<code>list</code>	Связывает <code>&lt;input&gt;</code> с элементом <code>&lt;datalist&gt;</code> через его <code>id</code> . В элементе <code>&lt;datalist&gt;</code> указывают значения, которые пользователь может выбрать, когда вбивает текст в пустое поле. Допустим, вы ввели букву «М» в поле ввода города, и браузер предложил варианты из <code>&lt;datalist&gt;</code> : «Москва», «Магадан» и т.д. Этот атрибут не подходит для таких типов <code>&lt;input&gt;</code> , как <code>hidden</code> , <code>password</code> , <code>checkbox</code> , <code>radio</code> , <code>file</code> и любых кнопок.
<code>max</code>	Указывает максимальные числовые значения для полей с типами <code>number</code> и <code>range</code> . Максимальные значения даты в формате ГГГГ-ММ-ДД для полей с типами <code>date</code> и <code>datetime-local</code> .
<code>maxlength</code>	Максимальное количество символов, разрешенных в элементе <code>&lt;input&gt;</code> .
<code>min</code>	Указывает минимальные числовые значения для полей с типами <code>number</code> и <code>range</code> . Минимальные значения даты в формате ГГГГ-ММ-ДД для полей с типами <code>date</code> и <code>datetime-local</code> .
<code>multiple</code>	<i>Атрибут булевого типа.</i> Указывает на то, что пользователь может ввести более одного значения в элемент <code>&lt;input&gt;</code> .
<code>name</code>	Уникальное имя элемента. Обычно имя нужно, чтобы связать <code>&lt;input&gt;</code> с формой и отправить данные на сервер.
<code>pattern</code>	Регулярное выражение, по которому проверяется значение элемента <code>&lt;input&gt;</code> .
<code>placeholder</code>	Используется для указания подсказки, описывающей ожидаемое значение поля ввода.
<code>readonly</code>	<i>Атрибут булевого типа.</i> Указывает, что поле ввода доступно только для чтения. Поле ввода только для чтения не может быть изменено. Не позволяет пользователю как-либо изменять элемент, оставляя его при этом рабочим. Применимо только для текстовых полей.
<code>required</code>	<i>Атрибут булевого типа.</i> Делает поле для ввода обязательным для заполнения. Пользователь не сможет нажать «Отправить», пока не введёт данные в поле. А если нажмет, то браузер выведет сообщение о необходимости заполнить поле.
<code>size</code>	Ширина в символах элемента <code>&lt;input&gt;</code> .

### Продолжение таблицы 6.3.3

<code>src</code>	URL-адреса изображения, используемого в качестве кнопки отправки (только для <code>image</code> ).
<code>step</code>	Указывает шаг, с которым будет изменяться значение поля. Задаётся в числовом формате и работает с любым из перечисленных типов: <code>number</code> , <code>range</code> , <code>date</code> и <code>datetime-local</code> .
<code>width</code>	Указание ширины элемента <code>&lt;input&gt;</code> (только для <code>image</code> ).

## 6.4 Элемент `<textarea>`

Тег `<textarea>` используется для создания многострочного поля ввода (листинг 6.4.1, рисунок 6.4.1). Например, когда нужно оставить комментарий, написать отзыв или статью. При необходимости поле может иметь изменяемый размер. Нужно отметить, что в этом поле можно писать только чистый текст.

Поле `<textarea>` стилизуется так же, как и поле ввода `<input>`. К нему применимы все свойства блочной модели.

По умолчанию поле ввода `<textarea>` может изменять свой размер, если потянуть за нижний правый угол. Это поведение можно изменить, управляя CSS-свойством `resize`.

### Листинг 6.4.1 – Пример создания элемента `<area>`

```
<label for="about">Расскажите о себе:</label><br>
<textarea id="about" name="about" rows="7" cols="35">Студент второго курса</textarea>
```

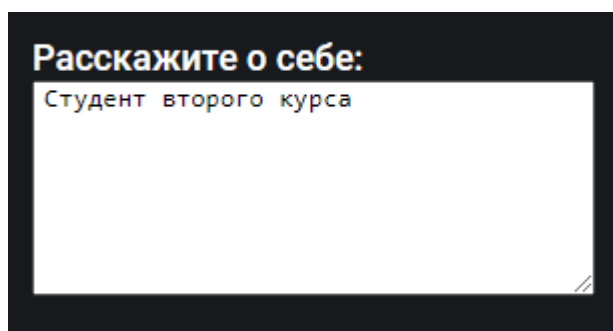


Рисунок 6.4.1 – Отображение поля ввода на странице

Далее, в таблице 6.4.1, представлены основные атрибуты тега `<area>` с их описанием.

Таблица 6.4.1 – Атрибуты тега <area>

Атрибут	Описание атрибута
<code>autocomplete</code>	Атрибут, указывающий, нужно ли поле заполнять автоматически сохранёнными в браузере значениями. Значения атрибута: <ul style="list-style-type: none"> <li><code>on</code> — поле будет автоматически заполняться значением, сохранённым в браузере во время предыдущей отправки формы</li> <li><code>off</code> — поле не будет заполняться браузером автоматически.</li> </ul> Также, это значение нужно использовать, если документ предоставляет собственный метод автозаполнения полей.
<code>autofocus</code>	<i>Атрибут булевого типа.</i> Если он указан, то при загрузке страницы фокус будет автоматически помещён в данное поле ввода.
<code>cols</code>	Задаёт ширину поля ввода в символах. Если атрибут задан, то должен иметь значением целое положительное число. Если не задан, то по умолчанию берётся как 20.
<code>disabled</code>	<i>Атрибут булевого типа.</i> Если задан, то поле отключается для взаимодействия с пользователем.
<code>form</code>	Атрибут указывает на элемент <code>&lt;form&gt;</code> , с которым связано поле ввода. Значением атрибута должен быть <code>id</code> формы в пределах текущего документа. Если атрибут не задан, то <code>&lt;textarea&gt;</code> обязательно должен находиться внутри тега <code>&lt;form&gt;</code> . Но если задать атрибут, то нахождение внутри формы не обязательно и <code>&lt;textarea&gt;</code> может находиться в любом месте страницы.
<code>maxlength</code>	Максимальное число символов в поле (включая пробелы и переводы строк), которое может вводить пользователь.
<code>minlength</code>	Минимальное число символов, которое должен ввести пользователь.
<code>name</code>	Имя поля. При отправке формы значение атрибута <code>name</code> будет ключом в отправляемом объекте.
<code>placeholder</code>	Подсказка для пользователя, что вводить в этом поле.
<code>readonly</code>	<i>Атрибут булевого типа.</i> Если он задан, то пользователь не может редактировать текст в поле, но по-прежнему может с ним взаимодействовать: кликать, копировать текст.

#### Продолжение таблицы 6.4.1

<code>required</code>	Атрибут булевого типа. Указывает, должно ли поле обязательно быть заполнено. Если поле не заполнить, то при попытке отправки формы браузер покажет ошибку.
<code>rows</code>	Задаёт высоту поля ввода в строках. Если атрибут не задан, то по умолчанию высота задаётся равной двум строкам.
<code>spellcheck</code>	Атрибут указывает, должна ли быть включена проверка правописания в поле. Может принимать следующие значения: <ul style="list-style-type: none"> <li><code>true</code> — проверка правописания включена.</li> <li><code>default</code> — указывает на поведение по умолчанию.</li> <li><code>false</code> — проверка правописания выключена.</li> </ul>
<code>wrap</code>	Атрибут определяет, будут ли добавлены символы переноса строк текста при отправке формы. Может принимать значения: <ul style="list-style-type: none"> <li><code>hard</code> — когда форма отправляется, то браузер, основываясь на значении атрибута <code>cols</code> добавляет в текст служебные символы переноса строки. Таким образом, сохраняется информация о переносах строк, сделанных пользователем в поле ввода.</li> <li><code>soft</code> — значение по умолчанию. При отправке формы символы переноса строк добавлены не будут, и текст будет отправлен одной длинной строкой</li> </ul>

### 6.5 Элемент `<button>`

Тег `<button>` в HTML используется для определения кликабельной кнопки и для отправки контента. Изображения и текстовое содержимое можно использовать внутри тега `<button>` (листинг 6.5.1, рисунок 6.5.1).

Тег `<input>` с атрибутом `type="button | reset | submit"` тоже создаёт кнопку, но `<button>` проще стилизовать, так как внутри `<button>` можно добавить любой HTML-контент, например, `<em>`, `<strong>` или `<img>`.

Листинг 6.5.1 – Создание кнопки через `<button>`

```
<button name="button">Нажми на кнопку</button>
```

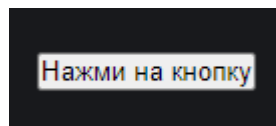


Рисунок 6.5.1 – Отображение стандартной кнопки на странице

Основные атрибуты тега `<button>` представлены в таблице 6.5.1.

Таблица 6.5.1 – Основные атрибуты тега `<button>`

Атрибут	Описание атрибута
<code>autofocus</code>	Атрибут булевого типа. Выбирает и подсвечивает кнопку при загрузке страницы, так что её можно нажать с клавиатуры клавишей Enter.
<code>disabled</code>	Атрибут булевого типа. Делает кнопку неактивной: нажать на неё нельзя, по умолчанию отображается серым цветом.
<code>form</code>	Связывает кнопку с любой формой в документе через <code>id</code> .
<code>formaction</code>	Задаёт URL-адрес, на который отправляются данные после нажатия на кнопку.
<code>formenctype</code>	Определяет формат файла, который пользователь может отправить при нажатии на кнопку.
<code>formtarget</code>	Показывает данные, которые ввёл пользователь. Где браузер откроет результат, зависит от ключевого слова: <ul style="list-style-type: none"><li>• <code>_self</code>: показывает данные в текущем окне. Это значение используется по умолчанию.</li><li>• <code>_blank</code>: показывает данные в новом окне браузера — его используют чаще всего.</li><li>• <code>_parent</code>: показывает данные внутри родительского элемента</li></ul>

	<p>фрейма, а если такого нет, то загружает в текущем окне.</p> <ul style="list-style-type: none"> <li>• <code>_top</code>: отменяет все родительские фреймы и загружает страницу в полном окне браузера. Если родительских фреймов нет, то показывает данные в текущем окне.</li> </ul>
<code>name</code>	Уникальное имя кнопки.
<code>type</code>	<p>Задаёт действие кнопки при нажатии:</p> <ul style="list-style-type: none"> <li>• <code>submit</code>: отправляет данные на сервер.</li> <li>• <code>reset</code>: удаляет введённые данные из формы.</li> <li>• <code>button</code>: просто кнопка. Действие для неё можно задать через скрипты.</li> </ul>
<code>value</code>	Задаёт исходное значение кнопки, которое отправляется на сервер вместе с данными пользователя.

## 6.6 Элементы `<select>` и `<option>`

Элемент `<select>` используется для выпадающего списка (листинг 6.6.1, рисунок 6.6.1).

Листинг 6.6.1 – Создание выпадающего списка

```
<form>
  <label for="city-select">Ваш город</label>
  <select name="city" id="city-select">
    <option value="">-- Выберите город --</option>
    <option value="moscow">Москва</option>
    <option value="petersburg">Санкт-Петербург</option>
    <option value="magadan">Магадан</option>
    <option value="kazan">Казань</option>
  </select>
</form>
```



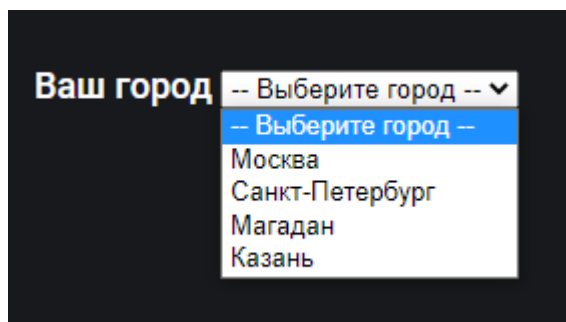


Рисунок 6.6.1 – Выпадающий список на странице

Элемент `<option>` используется в интерактивных элементах управления для вёрстки одиночного пункта списка. Тег `<option>` имеет следующие возможные атрибуты:

- Если нужно, чтобы изначально по умолчанию был выбран какой-то элемент из списка, нужно задать соответствующему тегу `<option>` атрибут `selected` (булевый тип).
- Если задан атрибут `disabled` (булевый тип), то пункт списка нельзя выбрать. Часто браузеры выделяют такой элемент управления серым цветом, и на нём не срабатывают события клика или фокуса.
- Значение атрибута `label` задаёт текст пункта в списке. Если атрибут не задан, то в качестве значения берётся текстовое содержимое тега `<option>`.

Некоторые специфические атрибуты тега `<select>` представлены в таблице 6.6.1.

Таблица 6.6.1 – Основные атрибуты тега `<select>`

Атрибут	Описание атрибута
<code>autocomplete</code>	Разрешает автозаполнение.
<code>autofocus</code>	<i>Атрибут булевого типа.</i> Если он указан, то при загрузке страницы фокус будет автоматически помещён на наш выпадающий список.
<code>disabled</code>	<i>Атрибут булевого типа.</i> Если задан, то выпадающий список отключается для взаимодействия с пользователем.
<code>form</code>	Атрибут указывает на элемент <code>&lt;form&gt;</code> , с которым связан выпадающий список.

	Значением атрибута должен быть <code>id</code> формы в пределах текущего документа.
<code>multiple</code>	Атрибут булевого типа. Включает возможность выбора сразу нескольких пунктов списка. Если атрибут задан, то внешний вид списка поменяется с однострочного на многострочный с возможностью скроллинга.
<code>name</code>	Имя выпадающего списка.
<code>required</code>	<i>Атрибут булевого типа.</i> Указывает, должен ли обязательно быть выбран какой-то пункт выпадающего списка, значение атрибута <code>value</code> которого — это не пустая строка.
<code>size</code>	Числовой атрибут. Если включён атрибут <code>multiple</code> , то это число указывает на количество видимых пунктов списка при скроллинге.

## 6.7 Элементы `<fieldset>` и `<legend>`

Тег `<fieldset>` группирует элементы формы (поля ввода `<input>`, `<textarea>`, выпадающие списки `<select>` и другие) в блок с характерным выделением границ. С помощью тега `<legend>` внутри `<fieldset>` можно задать заголовок для создаваемой группы (он может быть только один и обязательно должен идти первым вложенным элементом) (листинг 6.7.1 и рисунок 6.7.1).

Что удобно в использовании `<fieldset>`, так это возможность заблокировать все вложенные группы связанных элементов формы (набор полей) внутри тега одним атрибутом `disabled`.

## Листинг 6.7.1 – Создание группировки элементов с <fieldset>

```
<form>
  <fieldset>
    <legend>Сколько материков на Земле?</legend>
    <label>
      <input type="radio" name="answer" value="four">
      Четыре
    </label>
    <label>
      <input type="radio" name="answer" value="six">
      Шесть
    </label>
    <label>
      <input type="radio" name="answer" value="seven">
      Семь
    </label>
    <label>
      <input type="radio" name="answer" value="five">
      Пять
    </label>
  </fieldset>
</form>
```

Сколько материков на Земле?

☐ Четыре ☐ Шесть ☐ Семь ☐ Пять

Рисунок 6.7.1 – Группировка элементов с <fieldset> на странице

## 6.8 Инфографика по всем типам элемента <input>

На рисунке 6.8.1 представлены все типы элемента <input>.

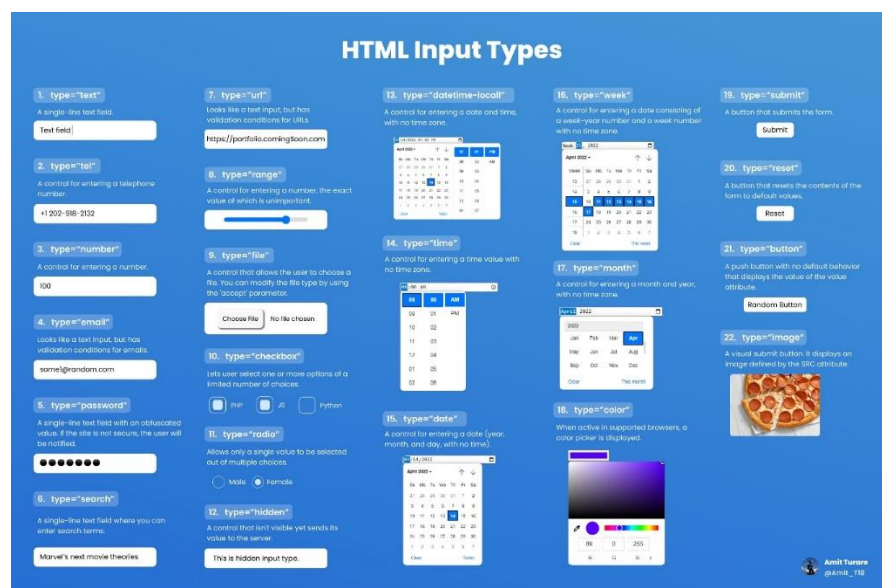


Рисунок 6.6.2 – Инфографика по типам элемента <input>

## 7 Элемент <iframe>

Элемент `<iframe>` - это контейнер, который позволяет вставить любой HTML-документ из другого источника. Часто этот документ интерактивный - например, карта, видео или пост из соцсети.

Этот контейнер обычно называют фреймом. Внешний вид фрейма и его положение на странице можно задать через CSS.

Встраивать файлы можно либо по ссылке с помощью атрибута `src`, либо целиком HTML-код файла с помощью `srcdoc`.

`<iframe>` - это как будто ещё одна страница, которая загружается внутри вашей страницы. Содержимое фрейма может загружаться дольше, чем ваш сайт.

В `<iframe>` стоит добавить атрибут `title` на случай, если пользователь не может увидеть страницу и использует инструмент чтения с экрана.

К атрибутам тега `<iframe>` относятся:

- `src` - URL-адрес файла, который вы встраиваете.
- `srcdoc` - позволяет встроить HTML-код целиком, так что браузеру не придётся подгружать что-то по внешней ссылке. Если используется этот атрибут, то ссылка `src` игнорируется.
- `sandbox` - ограничивает определённые действия и функции встраиваемого документа. Если просто указать атрибут `sandbox`, то он применит сразу все возможные ограничения.

**Пример синтаксиса:**

```
<iframe src="URL" title="description" width="400" height="400"></iframe>
```

## 8 Медиа-элемент <video>

Тег `<video>` добавляет на страницу видеоплеер, который может воспроизвести одно или несколько видео (листинг 8.1).

Листинг 8.1 – Пример добавления видео на страницу

```
<video controls width="250">  
  <source src="sky.webm" type="video/webm">  
  <source src="sky.mp4" type="video/mp4">  
  Ваш браузер не поддерживает встроенные видео!  
</video>
```

С помощью атрибутов тега можно настроить кнопки управления, зацикливание видео и другие параметры.

В атрибуте `src` указывается ссылка на видеофайл. Ещё ссылку можно задать через тег `<source>`.

#### **Атрибуты (параметры) воспроизведения и элементы управления:**

- `controls` — добавляет стандартные элементы управления видеоплеером: кнопку воспроизведения и паузы, полосу прокрутки, уровень громкости, полноэкранный режим и другие элементы, в зависимости от браузера.
- `autoplay` — видео воспроизводится автоматически после загрузки страницы.
- `loop` — зацикливает воспроизведение видео, так что оно снова начинается каждый раз после завершения.
- `muted` — видео начнёт играть без звука, пока его не включит пользователь.
- `playsinline` — контролирует воспроизведение видео на мобильных устройствах. Обычно видео открывается в полноэкранном режиме, но этот атрибут позволит запретить такое поведение, оставив видео в рамках элемента.

#### **Атрибуты (параметры) размеров и постера:**

- `width` — ширина видео в пикселях.
- `height` — высота видеоплеера в пикселях.
- `poster` — URL-адрес картинки, которая будет показываться, пока видео не загрузится. Обложка ролика.

#### **Атрибуты (параметры) обработки данных:**

- `buffered` — этот атрибут собирает информацию о том, какие минуты видео уже загрузились.
- `preload` — подсказывает браузеру, нужно ли загружать видео или информацию о нём сразу со страницей. Без этого атрибута предварительная загрузка видео будет зависеть от настроек конкретного браузера. Есть несколько значений:
  - `none` — видео не загружается предварительно.
  - `metadata` — загружается только информация о файле, например, размер, продолжительность или обложка.

- `auto` — видеофайл загружается со страницей, чтобы пользователь мог сразу начать смотреть его. Если стоит атрибут `autoplay`, то `preload` не работает.

**Примечание:** добавление элемента `<audio>` схоже с добавлением элемента `<video>`.

## Практическое задание

1. Создать форму `form` (можно разбить на несколько разных форм для использования всех необходимых элементов) с использованием атрибута `method` и следующими элементами:
  - Элементов `<input>` с атрибутом `type` со значениями:
    - `text`
    - `tel`
    - `email`
    - `password`
    - `file`
    - `checkbox`
    - `radio`
    - `submit`
    - `datetime-local`и со следующими другими атрибутами: `name`, `id` (зависит от метода добавления элемента), `checked`, `min`, `max`, `pattern`, `placeholder`, `required`.
  - Элемента(-ов) `<button>`.
  - Элемента(-ов) `<textarea>` с атрибутами `cols`, `rows`, `maxlength`, `minlength`, `placeholder`, `required`.
  - Элементов `<fieldset>` и `<legend>`.
  - Элементов `<label>` с атрибутами `for`, `name` (зависит от метода добавления элемента).
  - Элементов `<select>` и `<option>` с атрибутами `multiple`, `required`, `size`.
2. Задать стилизацию через CSS для всей(-х) формы (или форм): цвета фона/заливки элементов, цвета фона и границы полей ввода (использовать псевдокласс `:focus`, задающий стиль для элемента, который находится в фокусе, когда курсор установлен на то или иное поле формы) и других элементов, внешние и внутренние отступы (использовать относительные единицы измерения), шрифты и т.д.
3. Использовать псевдоэлемент `::placeholder` для стилизации подсказки под атрибутом `placeholder`.
4. Встроить карту (например, из Google Maps) на веб-страницу с помощью `<iframe>`.

5. Добавить элемент `<video>` (например, с YouTube) на веб-страницу через `<iframe>`.