



# НЕСТАНДАРТНЫЕ ЭЛЕМЕНТЫ ФОРМ



АЛЕКСАНДР ФИТИСКИН / WEBZILLA



# АЛЕКСАНДР ФИТИСКИН

Front-end разработчик Webzilla



# НАША ЗАДАЧА

Нам поступила новая задача по верстке форм. Нам прислали первую версию формы регистрации на внутреннем сайте.

Как и раньше, уже есть написанный код, который нам нужно доработать. Сейчас форма выглядит так:

**Введите ваше имя**

Например, Иван

**Введите ваш email**

Например, ivan@gmail.com

**Отправить**

# НАША ЗАДАЧА

Так выглядит макет:

Введите ваше имя

Например, Иван

Введите ваш email

Например, ivan@gmail.com

Выберите город проживания

Москва



Выберите пол:

Мужской  Женский

Вы согласны на обработку персональных данных

Отправить

---

# РАСПОЛОЖИМ КНОПКУ ПО ЦЕНТРУ

Давайте посмотрим на свойство `display` и узнаем, каким элементом является `button`.

# display: inline-block

Ранее мы с вами изучили значения `block`, `inline` и `flex` для свойства `display`. Кроме этих значений, у свойства `display` есть значение `inline-block`.

Оно является особенным, потому что оно обладает свойствами как строчных, так и блочных элементов.

# СВОЙСТВА `width` И `height` РАССЧИТЫВАЮТСЯ ПО СОДЕРЖИМОМУ

Родительский контейнер



строчно-блочный элемент

A diagram illustrating the concept of inline-block elements. It shows a large rectangular container with a dashed blue border. Inside this container, there is a smaller rectangular element with a solid black border. The text 'строчно-блочный элемент' is positioned to the left of this inner element.

Когда браузер встречает элемент, у которого есть значение `inline-block`, то свойства `width` и `height` рассчитываются по содержимому элемента.

# для свойств `width` и `height` можно установить значения

У строчно-блочных элементов можно установить значения для свойств `width` и `height`. Например, для элемента `span` в следующем коде:

```
1 span {  
2   display: inline-block;  
3   width: 200px;  
4   height: 100px;  
5 }
```

Родительский контейнер

строчно-блочный  
элемент

# СТРОЧНО-БЛОЧНЫЕ ЭЛЕМЕНТЫ РАСПОЛАГАЮТСЯ В ЛИНИЮ

Строчно-блочные элементы располагаются в одну строку друг за другом.

Например, элементы `div` в следующем коде:

```
1  div {  
2      display: inline-block;  
3      background-color: #eeeeee;  
4      padding: 8px;  
5      border: 3px solid #800080;  
6  }
```

Родительский контейнер

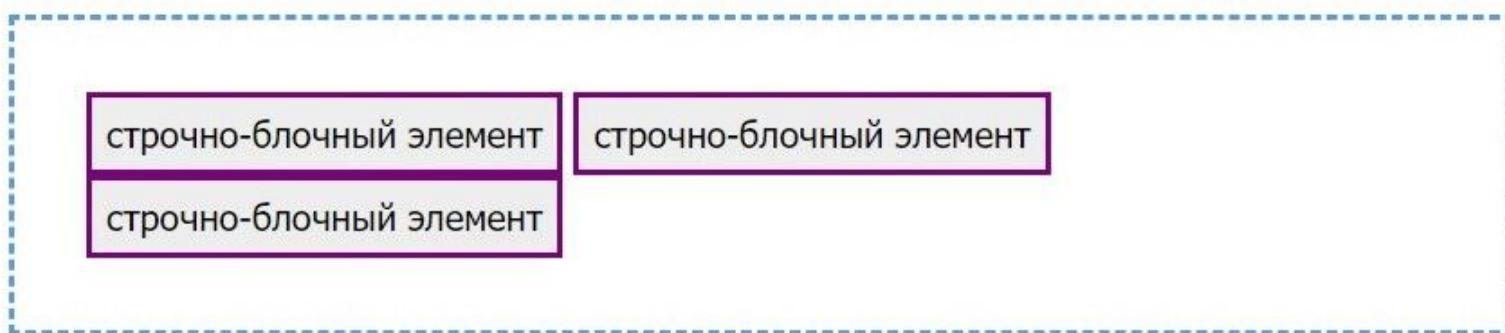
строчно-блочный элемент

строчно-блочный элемент

# ЭЛЕМЕНТЫ НЕ ПОМЕЩАЮТСЯ В СТРОКЕ

В этой ситуации элемент, который не поместился в строке, переносится на новую.

Родительский контейнер



# МЕЖДУ СТРОЧНО-БЛОЧНЫМИ ЭЛЕМЕНТАМИ СОЗДАЕТСЯ ПРОБЕЛ

Строчно-блочные элементы по умолчанию имеют расстояние между собой, которое берется из-за того, что в коде они написаны на новой строке.

```
1 | <body>
2 |   <div>строчно-блочный элемент</div>
3 |   <div>строчно-блочный элемент</div>
4 | </body>
```

Родительский контейнер



строчно-блочный элемент    строчно-блочный элемент

# НАПИСАТЬ ЭЛЕМЕНТЫ НА ОДНОЙ СТРОКЕ БЕЗ ПРОБЕЛА

```
1 <body>
2   <div>строчно-блочный элемент</div><div>строчно-блочный элемент</div>
3 </body>
```

# ЗАДАТЬ font-size: 0 ДЛЯ РОДИТЕЛЯ

При этом способе нужно обязательно задать `font-size` для строчно-блочных элементов, чтобы восстановить текст элементов

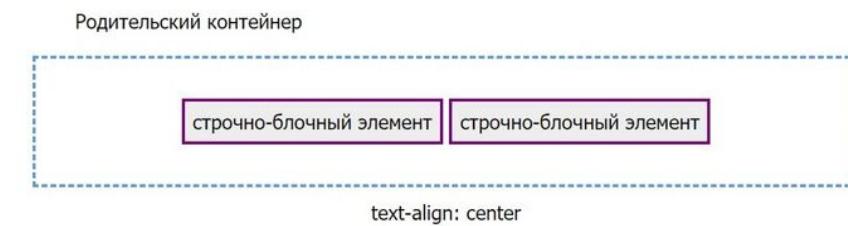
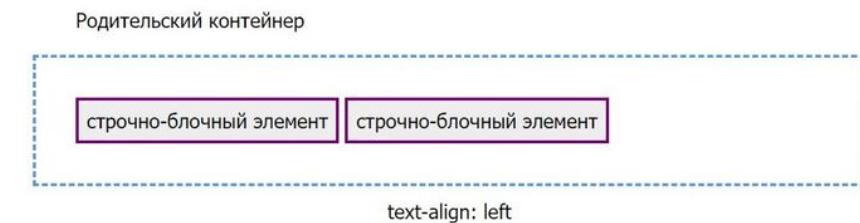
```
1 .parent {  
2     font-size: 0;  
3 }  
4  
5 .child {  
6     font-size: 20px;  
7 }
```

Родительский контейнер

строчно-блочный элемент    строчно-блочный элемент

# МОЖНО СПОЗИЦИОНИРОВАТЬ СТРОЧНО-БЛОЧНЫЙ ЭЛЕМЕНТ ПО ГОРИЗОНТАЛИ

Для этого нужно задать `text-align` для родительского элемента, который содержит строчно-блочные элементы.



# ПОСМОТРИМ НА СТРУКТУРУ ФОРМЫ

```
1 <form class="form">
2   <div class="form-row">
3     <!-- поле ввода имени -->
4   </div>
5   <div class="form-row">
6     <!-- поле ввода email -->
7   </div>
8   <div class="form-row">
9     <!-- кнопка -->
10  </div>
11 </form>
```

# ДОБАВИМ text-align

```
1 .form-row:last-child {  
2   text-align: center;  
3 }
```

Введите ваше имя

Например, Иван

Введите ваш email

Например, ivan@gmail.com

Отправить

[Демо](#)

# ВЕРНЕМСЯ К МАКЕТУ

Введите ваше имя

Например, Иван

Введите ваш email

Например, ivan@gmail.com

Выберите город проживания

Москва



Выберите пол:

Мужской  Женский

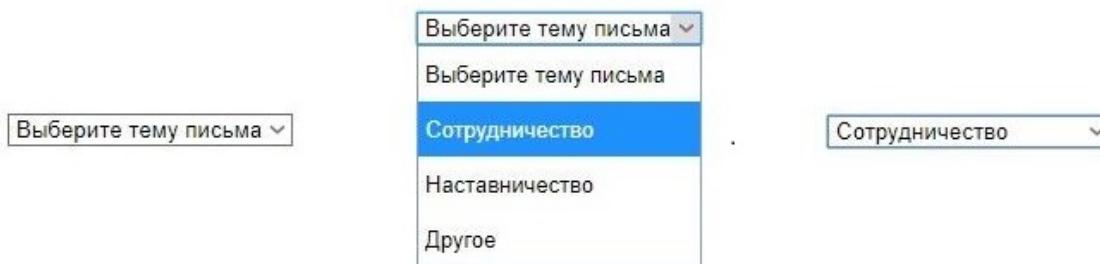
Вы согласны на обработку персональных данных

Отправить

# ЭЛЕМЕНТЫ `select` И `option`

Элемент `select` дает возможность пользователю выбрать вариант из предлагаемого списка. `select` является родительским блоком, в который вкладывают элементы `option`.

```
1 <select>
2   <option>Выберите тему письма</option>
3   <option>Сотрудничество</option>
4   <option>Наставничество</option>
5   <option>Другое</option>
6 </select>
```



# ДОБАВИМ СПИСОК ГОРОДОВ

```
1 <div class="form-row">
2   <span class="form-hint">Выберите город проживания</span>
3   <select class="select">
4     <option>Санкт-Петербург</option>
5     <option>Москва</option>
6     <option>Казань</option>
7   </select>
8 </div>
```

[Демо](#)

# РЕЗУЛЬТАТ

Например, Иван

Введите ваш email

Например, ivan@gmail.com

Выберите город проживания

Санкт-Петербург ▾

The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. In the main pane, a dropdown menu is open, displaying three options: 'Санкт-Петербург', 'Москва', and 'Казань'. A red box highlights this dropdown menu. The DOM tree on the left shows the corresponding HTML structure:

```
<body translate="no">
  <form class="form">
    <div class="form-row">...</div>
    <div class="form-row">...</div>
    <div class="form-row">
      <span class="form-hint">Выберите город проживания</span>
      <select class="select"> = $0
        <option>Санкт-Петербург</option>
        <option>Москва</option>
        <option>Казань</option>
      </select>
    </div>
  </form>
</body>
```

In the bottom status bar, the element path is shown as: html body div#result-iframe-wrap iframe#result,result-iframe html body form.form div.form-row select.select.

On the right, the 'Styles' panel shows the CSS rules applied to the selected element:

```
element.style {
}
select:not(:-internal-List-box) {
  overflow: visible !important;
}
select {
  border-radius: 0px;
  border-color: #rgb(169, 169, 169);
}
select {
  -webkit-appearance: menulist;
  box-sizing: border-box;
```

Демо

# АТРИБУТ `selected`

Атрибут `selected` показывает выбранный элемент `option` среди группы элементов `option`. Такой выбранный элемент будет отображаться пользователю до появления полного списка.

```
1 <select>
2   <option>Выберите тему письма</option>
3   <option>Сотрудничество</option>
4   <option selected>Наставничество</option>
5   <option>Другое</option>
6 </select>
```



# ДОБАВИМ `selected`

```
1 <div class="form-row">
2   <span class="form-hint">Выберите город проживания</span>
3   <select class="select">
4     <option>Санкт-Петербург</option>
5     <option selected>Москва</option>
6     <option>Казань</option>
7   </select>
8 </div>
```

[Демо](#)

# РЕЗУЛЬТАТ

The screenshot shows a browser's developer tools with the 'Elements' tab selected. A form is displayed with two input fields: one for email and one for city selection. The 'Styles' panel on the right shows the CSS rules applied to the selected element, which is a dropdown menu.

Form fields:

- Ведите ваш email: Например, ivan@gmail.com
- Выберите город проживания: Москва

Developer Tools Elements Panel:

- Elements tab is active.
- Selected element: <select class="select"> == \$0
- Content of the dropdown menu:
  - <option>Санкт-Петербург</option>
  - <option selected>Москва</option> (highlighted with a red box)
  - <option>Казань</option>
- Bottom status bar: html body div#result-iframe-wrap iframe#result.result-iframe html body form.form div.form-row select.select

Styles Panel:

- Styles tab is active.
- Applied styles:
  - element.style { } (User agent stylesheet)
  - select:not(:internal-list-box) { overflow: visible !important; } (User agent stylesheet)
  - select { border-radius: 0px; border-color: #rgb(169, 169, 169); } (User agent stylesheet)
  - select { -webkit-appearance: menulist; box-sizing: border-box; } (User agent stylesheet)

[Демо](#)

# ДОБАВИМ СЛЕДУЮЩИЕ СТИЛИ

```
1 .select {  
2     display: block;  
3     box-sizing: border-box;  
4     width: 480px;  
5     height: 45px;  
6  
7     padding-left: 20px;  
8     margin: 0;  
9     border: 1px solid #cbd1d4;  
10  
11    font-family: inherit;  
12    font-size: 14px;  
13    color: #333333;  
14 }
```

# РЕЗУЛЬТАТ

Ведите ваше имя

Например, Иван

Ведите ваш email

Например, ivan@gmail.com

Выберите город проживания

Москва

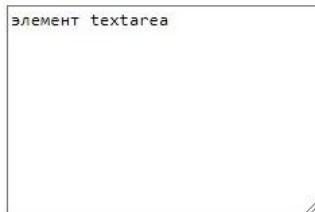
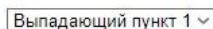
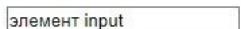
Отправить

Стрелочка не такая, как нам требуется.

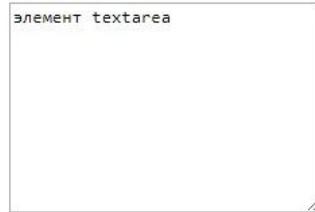
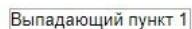
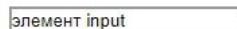
# ЗНАЧЕНИЕ `none` ДЛЯ СВОЙСТВ `-webkit-appearance` И `-moz-appearance`

С помощью данных свойств и значения `none` отключим стилизацию стандартных элементов форм.

Элементы без `-webkit-appearance` и `-moz-appearance`



Элементы с `-webkit-appearance` и `-moz-appearance`



# ДОБАВИМ `-webkit-appearance` И `-moz-appearance`

```

1 .select {
2   -webkit-appearance: none;
3   -moz-appearance: none;
4 }
```

The screenshot shows a browser window with developer tools open. At the top, there's a dropdown menu with the text "select.select | 480x45 вания" and a blue bar containing the word "Москва". Below this is a red "Отправить" button. The developer tools interface includes an Elements tab showing the DOM structure, a Styles tab where the CSS rule ".select { -webkit-appearance: none; -moz-appearance: none; }" is highlighted with a red box, and a Properties tab showing other styles like "display: block;" and "width: 480px;".

[Демо](#)

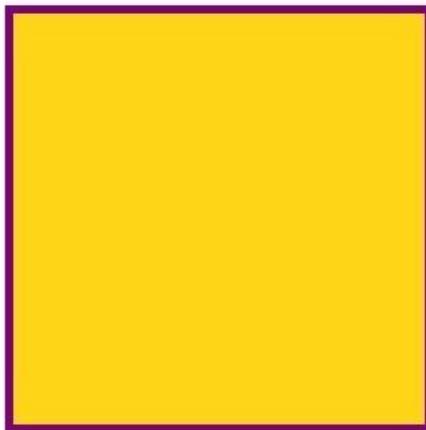
# Свойство background

Свойство `background` позволяет стилизовать фон элементов. Свойство состоит из следующих свойств.

- `background-color`
- `background-image`
- `background-repeat`
- `background-position`
- `background-size`

# background-color

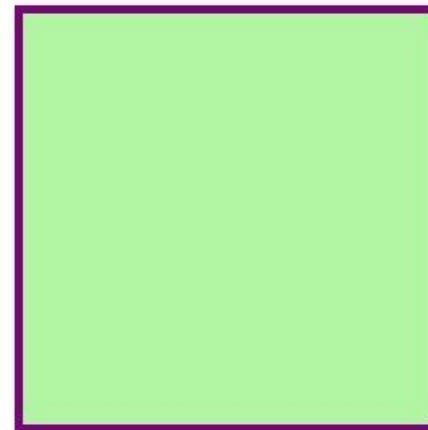
Свойство `background-color` позволяет задать цвет фона элемента. В качестве значения свойства используется шестнадцатеричный формат цвета.



`background-color: #ffd518`



`background-color: #eeeeee`



`background-color: #b2f5a3`

# background-image

Свойство `background-image` позволяет загрузить фоновое изображение для элемента. Свойство имеет следующие значения:

- `none` - отменяет загрузку изображения. Данное значение является значением по умолчанию.
- `url(путь к изображению)` - Функция `url` сообщает браузеру, по какому пути искать изображение для загрузки.



`background-image: url("web-developer.png")`

# background-repeat

Когда к элементу добавляется фоновое изображение, то по умолчанию оно повторяется по всей площади элемента.

- `repeat` - значение по умолчанию
- `repeat-x`

- `repeat-y`
- `no-repeat`



`background-repeat: repeat;`



`background-repeat: repeat-x;`



`background-repeat: repeat-y;`



`background-repeat: no-repeat;`

# background-position

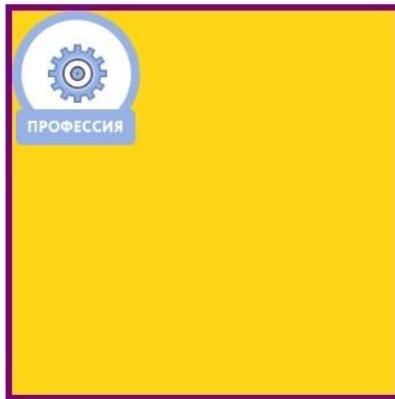
Свойство `background-position` определяет позицию фонового изображения. В качестве значения принимает координаты по осям X и Y.

```
background-position: X Y;
```

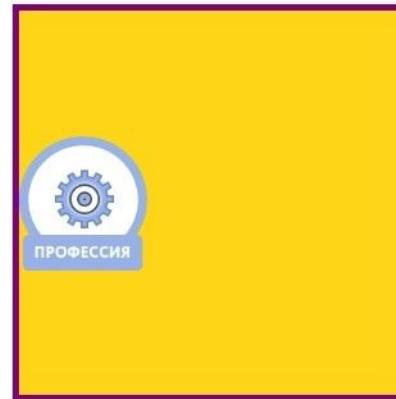
где X - координата по оси X, Y - координата по оси Y

# left для X, и top , center или right для Y

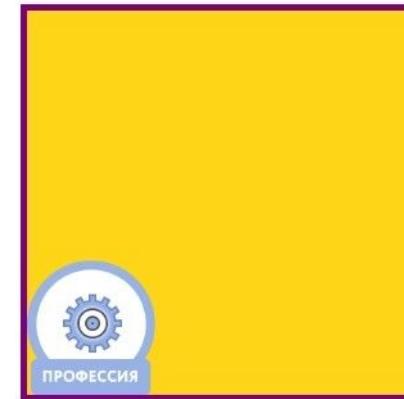
В этом случае браузер расположит изображение слева по оси X. По оси Y изображение будет сверху, если указано значение `top` ; по центру, если указано значение `center` ; снизу, если указано значение `bottom` .



`background-position: left top;`



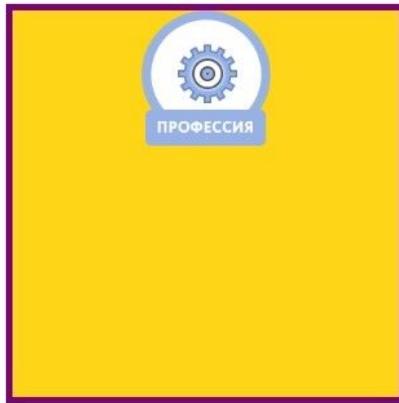
`background-position: left center;`



`background-position: left bottom;`

# center для X, и top , center или right для Y

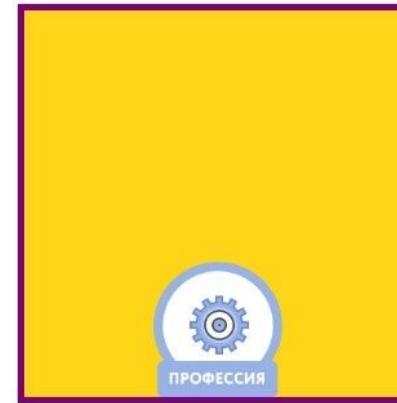
В этом случае браузер расположит изображение по центру по оси X. По оси Y изображение будет сверху, если указано значение `top`; по центру, если указано значение `center`; снизу, если указано значение `bottom`.



`background-position: center top;`



`background-position: center center;`



`background-position: center bottom;`

# right для X, и top , center или right для Y

В этом случае браузер расположит изображение справа по оси X. По оси Y изображение будет сверху, если указано значение `top` ; по центру, если указано значение `center` ; снизу, если указано значение `bottom` .



`background-position: right top;`

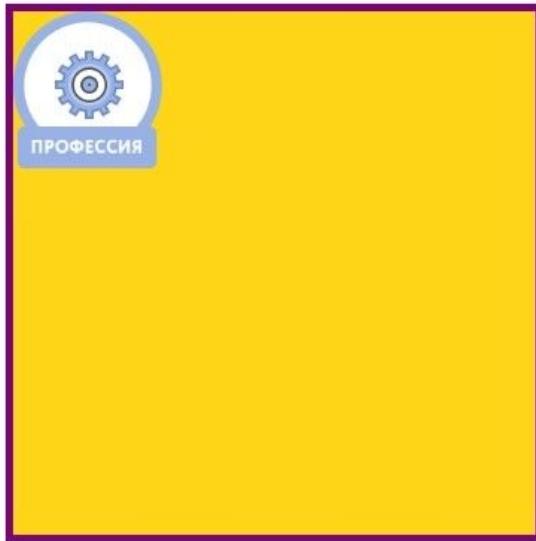


`background-position: right center;`

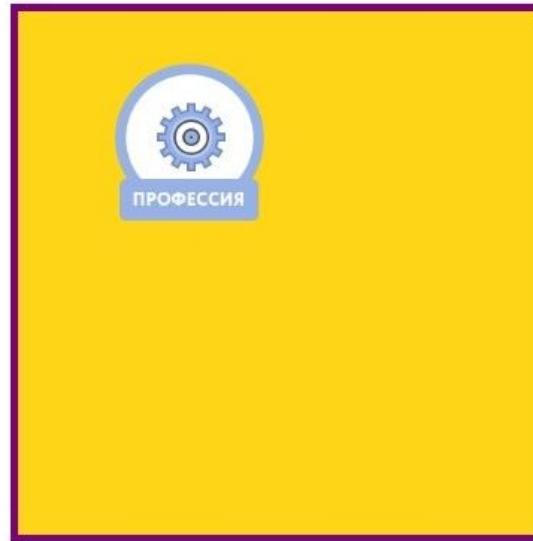


`background-position: right bottom;`

# ЗНАЧЕНИЕ В ПИКСЕЛЯХ ОТ ЛЕВОГО И ВЕРХНЕГО КРАЯ



background-position: left top;



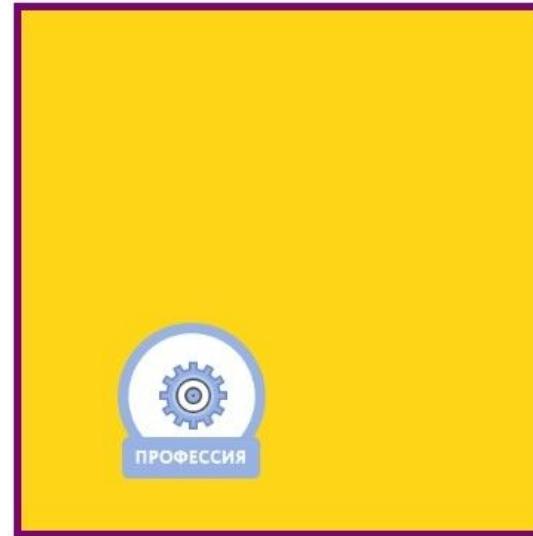
background-position: left 65px top 35px;

В этом случае браузер сделает отступ `65px` от левого края и `35px` от верхнего.

# ЗНАЧЕНИЕ В ПИКСЕЛЯХ ОТ ЛЕВОГО И НИЖНЕГО КРАЯ



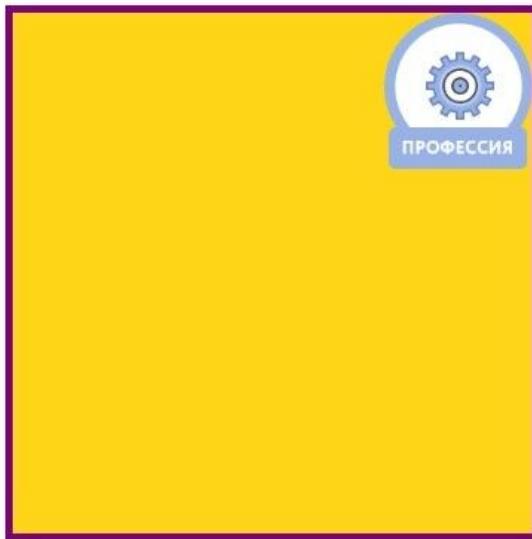
`background-position: left bottom;`



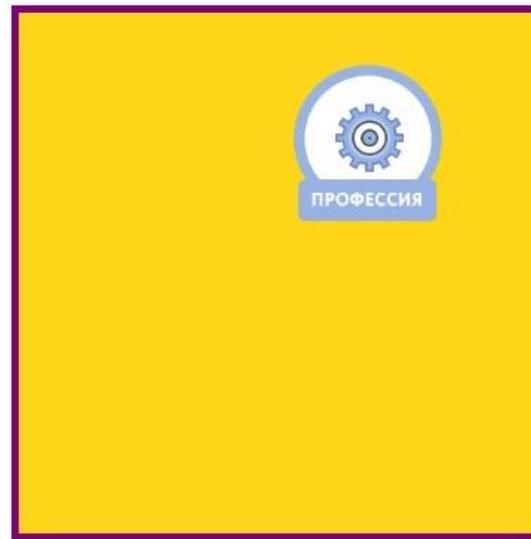
`background-position: left 65px bottom 35px;`

В этом случае браузер сделает отступ `65px` от левого края и `35px` от нижнего.

# ЗНАЧЕНИЕ В ПИКСЕЛЯХ ОТ ПРАВОГО И ВЕРХНЕГО КРАЯ



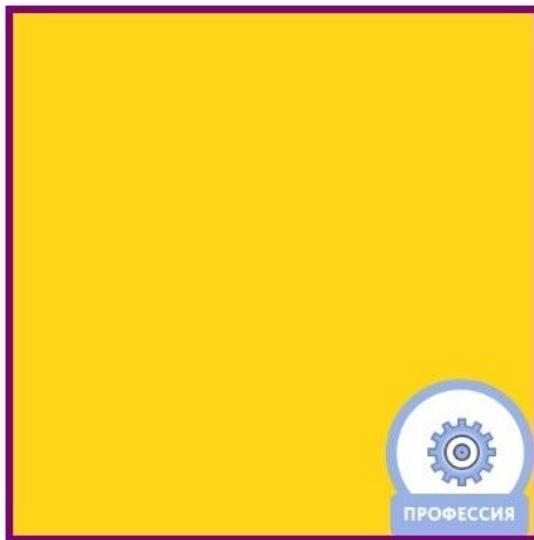
`background-position: right top;`



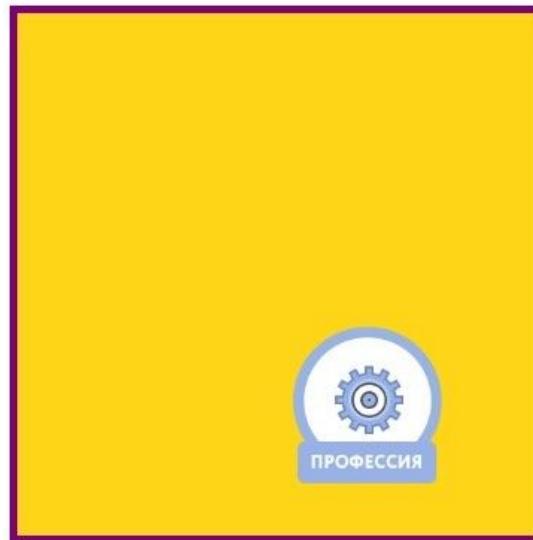
`background-position: right 65px top 35px;`

В этом случае браузер сделает отступ `65px` от правого края и `35px` от верхнего.

# ЗНАЧЕНИЕ В ПИКСЕЛЯХ ОТ ПРАВОГО И НИЖНЕГО КРАЯ



background-position: right bottom;



background-position: right 65px bottom 35px;

В этом случае браузер сделает отступ `65px` от правого края и `35px` от нижнего.

# background-size

С помощью свойства `background-size` браузер создает область внутри элемента, в которую он отобразит изображение. Для этого нам нужно указать сколько пикселей будет занимать картинка по оси X и Y, т.е ширину и высоту.

```
background-size: width height;
```

где `width` - ширина области, `height` - высота области

# ЗНАЧЕНИЕ `auto`



`background-size: auto;`

Когда для свойства задано значение `auto`, браузер создаст область для отображения изображения, исходя из его размеров.

# ЗНАЧЕНИЕ ДЛЯ ШИРИНЫ

Когда мы с вами указываем точные размеры изображения, то может случится так, что размеры изображения больше или меньше области, в которой браузер отобразит его. Поэтому рассмотрим все возможные варианты на примере изображения размером 135px на 143px .

- Ширина изображения совпадает с шириной области отображения;



`background-size: auto;`



`background-size: 135px;`

# ЗНАЧЕНИЕ ДЛЯ ШИРИНЫ

- Ширина изображения больше ширины области отображения;



`background-size: auto;`



`background-size: 120px;`

# ЗНАЧЕНИЕ ДЛЯ ШИРИНЫ

- Ширина изображения меньше, чем ширина области отображения.



background-size: auto;



background-size: 200px;

# ДОБАВИМ СТРЕЛКУ ДЛЯ `select`

```
1 .select {  
2   background-image: url("https://netology-code.github.io/resources/pics/select__mir  
3   background-repeat: no-repeat;  
4   background-position: right 5px center;  
5 }
```

[Демо](#)

# РЕЗУЛЬТАТ

Введите ваше имя

Например, Иван

Введите ваш email

Например, ivan@gmail.com

Выберите город проживания

Москва



Отправить

Попробуем нажать на надпись "Выберите город проживания". Ничего не произойдет.

[Демо](#)

# ПОСМОТРИМ НА СТРУКТУРУ

```
1 <div class="form-row">
2   <span class="form-hint">Выберите город проживания</span>
3   <select class="select">
4     <option>Санкт-Петербург</option>
5     <option selected>Москва</option>
6     <option>Казань</option>
7   </select>
8 </div>
```

# ДОБАВИМ `label` для `select`

```
1 <div class="form-row">
2   <label class="form-group">
3     <span class="form-hint">Выберите город проживания</span>
4     <select class="select">
5       <option>Санкт-Петербург</option>
6       <option selected>Москва</option>
7       <option>Казань</option>
8     </select>
9   </label>
10 </div>
```

[Демо](#)

# ПОПРОБУЕМ НАЖАТЬ НА ТЕКСТ

Фокус устанавливается на элементе `select`. Но сейчас у нас отображается стандартная синяя обводка.

Введите ваше имя

Например, Иван

Введите ваш email

Например, ivan@gmail.com

Выберите город проживания

Москва



Отправить

[Демо](#)

# ДОБАВИМ `select`

Посмотрим в CSS

```
1 .field:focus, .button:focus {  
2   outline: 2px solid #c53b2d;  
3   outline-offset: 5px;  
4 }
```

Нам остается добавить сюда элемент `select`.

```
1 .field:focus, .select:focus, .button:focus {  
2   outline: 2px solid #c53b2d;  
3   outline-offset: 5px;  
4 }
```

# РЕЗУЛЬТАТ

Введите ваше имя

Например, Иван

Введите ваш email

Например, ivan@gmail.com

Выберите город проживания

Москва



Отправить

[Демо](#)

# type="radio"

Часто при заполнении форм мы с вами встречаем элементы, которые требуют от нас выбора. Например, такой элемент есть при регистрации на сайте Reebok.ru.

The screenshot shows the Reebok.ru registration page. At the top, there's a navigation bar with the Reebok logo and links for МУЖЧИНЫ, ЖЕНЩИНЫ, ДЕТИ, ФИТНЕС, and CLASSICS. On the right side of the header is a search bar with a magnifying glass icon and the text 'поиск товаров', and a shopping cart icon. The main content area has a dark background. On the left, under the heading 'РЕГИСТРАЦИЯ', there's a section for 'ПЕРСОНАЛЬНЫЕ ДАННЫЕ' with fields for 'ИМЯ \*' and 'ФАМИЛИЯ \*'. Below that is a section for 'ДАТА РОЖДЕНИЯ' with three input fields for 'ДД \*', 'ММ \*', and 'ГГГГ \*'. A note at the bottom of this section states: 'Reebok собирает информацию о дате рождения в соответствии с политикой конфиденциальности. Это также позволит нам приготовить для тебя сюрприз ко дню рождения!' (Reebok collects information about your date of birth in accordance with the privacy policy. This will also allow us to prepare a surprise for you on your birthday!). On the right, under the heading 'ВАШ ЛИЧНЫЙ КАБИНЕТ', there's a section titled 'ПРЕИМУЩЕСТВА ЕДИНОГО ЛИЧНОГО КАБИНЕТА' with a list of benefits: '✓ Единый личный кабинет для доступа ко всем сервисам Reebok', '✓ Вы сможете просматривать историю заказов', '✓ Вы сможете быстрее совершать покупки', and '✓ Вы сможете менять настройки, добавлять товары в избранное и т.д.' (✓ A single personal cabinet for access to all Reebok services, ✓ You can view order history, ✓ You can quickly make purchases, ✓ You can change settings, add items to favorites, and so on.). At the bottom left, there's a red-bordered box containing a gender selection field with two options: 'Мужской' (Male) with a checked radio button and 'Женский' (Female) with an unchecked radio button. The entire registration form is set against a light gray background.

## type="radio"

```
1 <label>
2   <input type="radio">
3   мужской
4 </label>
5 <label>
6   <input type="radio">
7   женский
8 </label>
```

Но в данном случае браузер поймет, что существует две несвязанные в группы радиокнопки.

мужской  женский

## АТРИБУТ `name`

Используя атрибут `name`, браузер понимает, что перед ним объединенная группа радиокнопок, в которой может быть только одна активная.

```
1 <label>
2   <input type="radio" name="radio-group">
3   мужской
4 </label>
5 <label>
6   <input type="radio" name="radio-group">
7   женский
8 </label>
```

мужской  женский

мужской  женский

## ДОБАВИМ `input type="radio"`

```
1 <div class="form-row">
2   <span class="form-hint">Выберите пол:</span>
3   <div class="radio-group">
4     <input type="radio" class="radio" name="radio-group">
5     <span class="radio-group-text">Мужской</span>
6   </div>
7   <div class="radio-group">
8     <input type="radio" class="radio" name="radio-group">
9     <span class="radio-group-text">Женский</span>
10  </div>
11 </div>
```

[Демо](#)

# ЗАМЕНИМ div НА label

```
1 <label class="radio-group">
2   <input type="radio" class="radio" name="radio-group">
3   <span class="radio-group-text">Мужской</span>
4 </label>
5 <label class="radio-group">
6   <input type="radio" class="radio" name="radio-group">
7   <span class="radio-group-text">Женский</span>
8 </label>
```

Выберите пол:

- Мужской  Женский

[Демо](#)

# ПСЕВДОЭЛЕМЕНТЫ `::before` И `::after`

Псевдоэлементы `before` и `after` – это элементы, которые добавлены внутрь родительского элемента при помощи CSS. Для того чтобы создать псевдоэлемент, нужно рассмотреть следующий синтаксис.

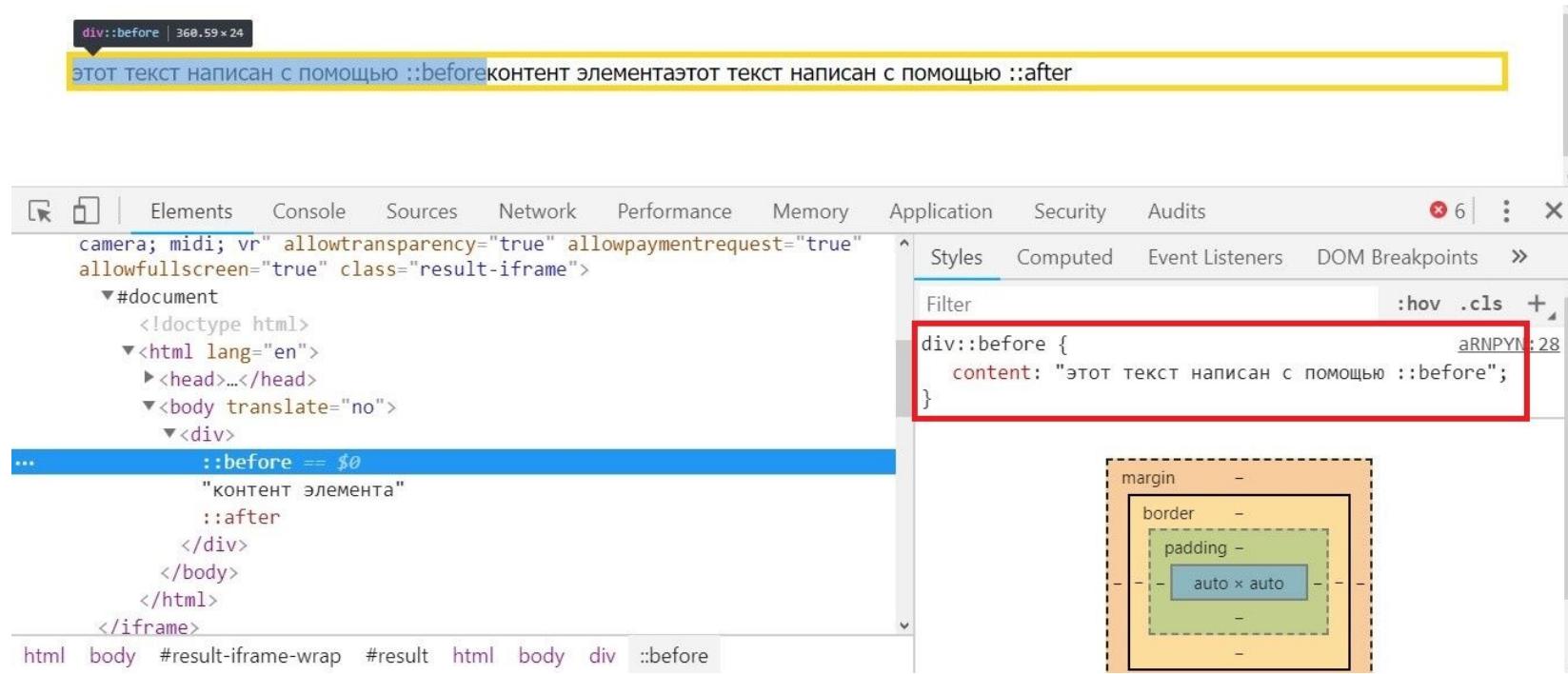
```
1 | родитель::before {  
2 |     content: "";  
3 | }  
4 |  
5 | родитель::after {  
6 |     content: "";  
7 | }
```

# СИНТАКСИС

После родителя через двойное двоеточие указывается тип псевдоэлемента – `before` или `after`. Псевдоэлемент `before` добавится перед контентом родительского элемента, а `after` – после. Каждый псевдоэлемент обязательно должен содержать свойство `content`, которое определяет содержимое псевдоэлемента.

```
1 div::before {  
2     content: "этот текст написан с помощью ::before";  
3 }  
4  
5 div::after {  
6     content: "этот текст написан с помощью ::after";  
7 }
```

# СИНТАКСИС



The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. On the left, the DOM tree displays an `<div>` element with a `::before` pseudo-element attached. The content of this pseudo-element is "ЭТОТ ТЕКСТ НАПИСАН С ПОМОЩЬЮ ::BEFORE". On the right, the 'Styles' panel shows the CSS rule for this pseudo-element:

```
div::before {  
    content: "ЭТОТ ТЕКСТ НАПИСАН С ПОМОЩЬЮ ::BEFORE";  
}
```

Below the styles panel, a visual representation of the box model is shown, illustrating the relationship between margin, border, padding, and content areas.

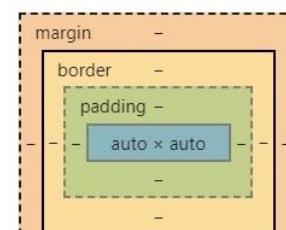
DOM Tree (Elements tab):

```
camera; midi; vr" allowtransparency="true" allowpaymentrequest="true"  
allowfullscreen="true" class="result-iframe">  
  #document  
    <!doctype html>  
    <html lang="en">  
      <head>...</head>  
      <body translate="no">  
        <div>  
          :before == $0  
          "контент элемента"  
          :after  
        </div>  
      </body>  
    </html>  
</iframe>
```

Styles Panel (Styles tab):

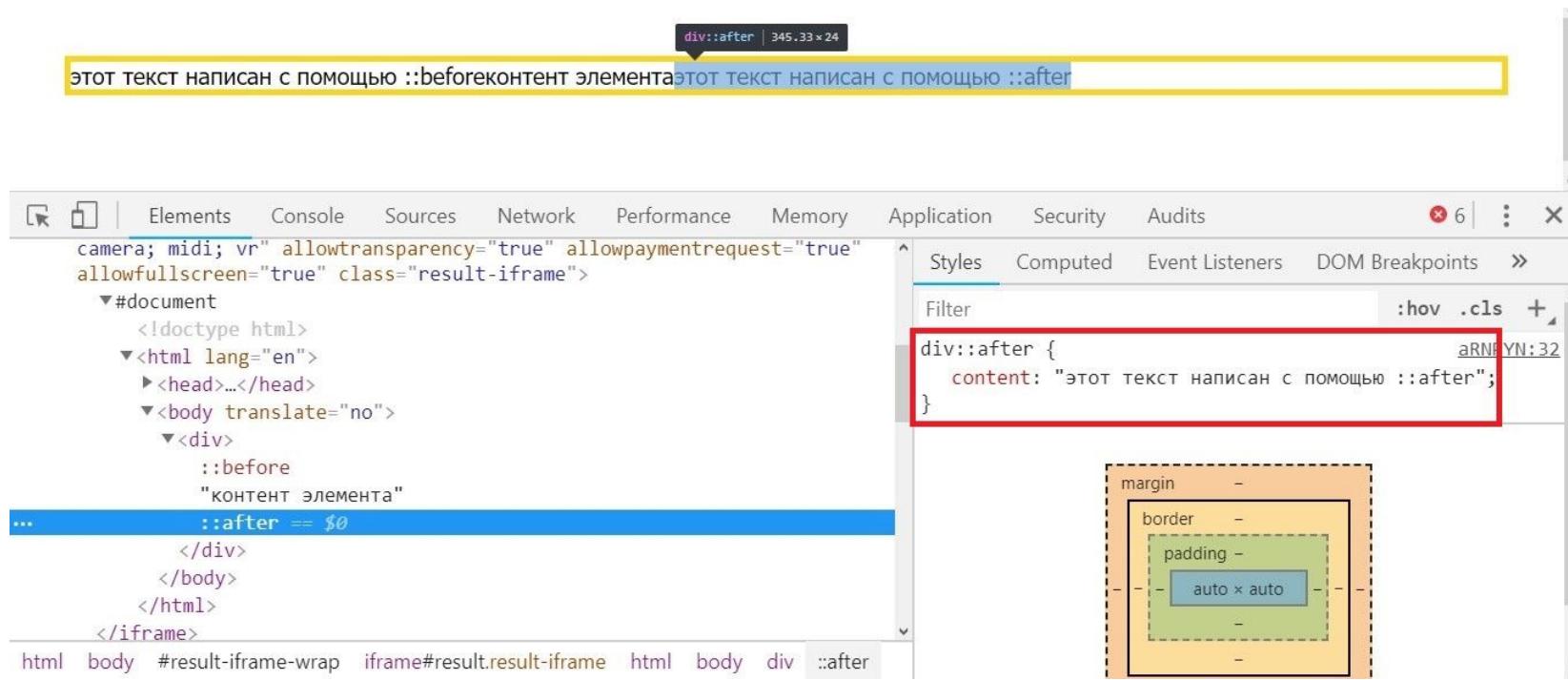
```
:hov .cls +  
div::before {  
    content: "ЭТОТ ТЕКСТ НАПИСАН С ПОМОЩЬЮ ::BEFORE";  
}
```

Visual Box Model Diagram:



The diagram illustrates the box model with nested dashed boxes. The outermost box is orange and labeled 'margin'. Inside it is a yellow box labeled 'border'. Inside the border is a green box labeled 'padding'. The innermost box is blue and labeled 'auto x auto', representing the content area.

# СИНТАКСИС



The screenshot shows the Chrome DevTools interface. The Elements tab is active, displaying the DOM structure of a page. A blue selection bar highlights the `::after` pseudo-element within a `div` element. The Styles tab is selected in the top navigation bar. A red box highlights the CSS rule for `div::after`:

```
div::after {  
    content: "этот текст написан с помощью ::after";  
}
```

The computed styles panel shows the following stack of styles for the highlighted element:

- margin -
- border -
- padding -
- auto × auto

The content area of the browser window displays the text "этот текст написан с помощью ::beforeконтент элемента" followed by "этот текст написан с помощью ::after".

# СИНТАКСИС

Если содержимого нет, то можно ничего не указывать.

```
1 div::before {  
2     content: "";  
3 }  
4  
5 div::after {  
6     content: "";  
7 }
```

# СИНТАКСИС

КОНТЕНТ ЭЛЕМЕНТА

The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. The left pane displays the DOM tree:

```
<html lang="en">
  <head>...</head>
  <body translate="no">
    <div>
      <::before> == $0
      "КОНТЕНТ ЭЛЕМЕНТА"
      <::after>
    </div>
  </body>
</html>
```

The node `<::before> == $0` is highlighted with a red box. The right pane shows the 'Styles' panel with the following CSS rule:

```
div::before {
  content: "";
}
```

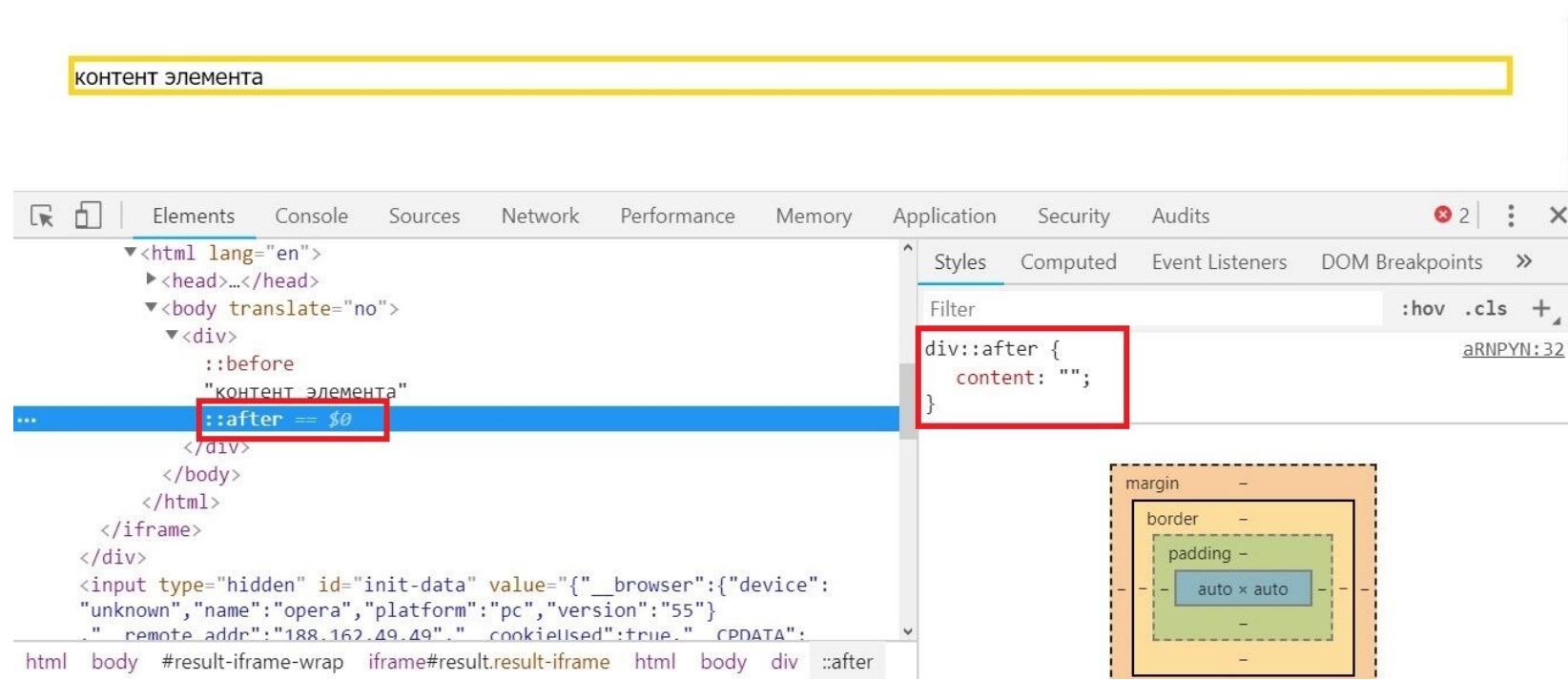
A red box highlights this rule. Below the styles panel, there is a visual representation of the CSS box model:

The diagram illustrates the box model with nested dashed boxes:

- Outermost box: `margin`
- Second box from outside: `border`
- Third box from outside: `padding`
- Innermost box: `auto x auto`

# СИНТАКСИС

контент элемента



The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. The left pane displays the DOM tree:

```
<html lang="en">
  <head>...</head>
  <body translate="no">
    <div>
      ::before
      "контент элемента"
      ::after == $0
    </div>
  </body>
</html>
</iframe>
</div>
```

The '::after' pseudo-element is highlighted with a red box. The right pane shows the 'Styles' tab with the following CSS rule:

```
div::after {
  content: "";
}
```

A detailed diagram on the right illustrates the box model components: margin (orange), border (inner orange), padding (green), and content (blue).

# СТИЛИЗАЦИЯ ПСЕВДОЭЛЕМЕНТОВ

```
1  div::before, div::after {  
2      padding-right: 10px;  
3      padding-left: 10px;  
4      background-color: purple;  
5      color: #ffffff;  
6  }  
7  
8  div::before {  
9      content: "этот текст написан с помощью ::before";  
10     margin-right: 10px;  
11  }  
12  
13 div::after {  
14     content: "этот текст написан с помощью ::after";  
15     margin-left: 10px;  
16  }
```

# СТИЛИЗАЦИЯ ПСЕВДОЭЛЕМЕНТОВ

The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. On the left, the DOM tree displays the following structure:

```
camera; midi; vr" allowtransparency="true" allowpaymentrequest="true"
allowfullscreen="true" class="result-iframe">
  #document
    <!doctype html>
    <html lang="en">
      <head>...</head>
      <body translate="no">
        <div>
          :before == $0
            "контент элемента"
            :after
          </div>
        </body>
      </html>
    </iframe>
```

The element with the class 'result-iframe' is highlighted in blue. The 'Styles' panel on the right shows two styles:

```
div::before {
  content: "этот текст написан с помощью ::before";
  margin-right: 10px;
}

div::before, div::after {
  padding-right: 10px;
  padding-left: 10px;
  background-color: purple;
  color: white;
}
```

The second style block is highlighted with a red rectangle. The 'content' property of the first block contains the text "этот текст написан с помощью ::before". The 'background-color' and 'color' properties of the second block are set to purple and white respectively.

# СТИЛИЗАЦИЯ ПСЕВДОЭЛЕМЕНТОВ

The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. The left sidebar displays the DOM tree, highlighting a specific element's pseudo-elements. The right panel shows the 'Styles' tab with two rules:

```
div::after {
  content: "Этот текст написан с помощью ::after";
  margin-left: 10px;
}

div::before, div::after {
  padding-right: 10px;
  padding-left: 10px;
  background-color: purple;
  color: white;
}
```

A red box highlights the second rule, which applies styles to both ::before and ::after pseudo-elements. The 'content' property in this rule is annotated with a tooltip: 'Этот текст написан с помощью ::after'. The 'margin-left' and 'background-color' properties are also annotated with their respective descriptions.

# СОЗДАЕМ КРУЖОК

Псевдоэлементы `::before` и `::after` являются очень удобным способом стилизации элементов без создания лишних тегов. Но есть один минус. Псевдоэлементы нельзя задать для элементов `input`, `textarea`, `select`.

Возвращаясь к верстке, нам нужно создать кружок, который добавится перед текстом "Мужской" и текстом "Женский".

```
1 <div class="radio-group">
2   <input type="radio" class="radio" name="radio-group">
3   <span class="radio-group-text">Мужской</span>
4 </div>
5 <div class="radio-group">
6   <input type="radio" class="radio" name="radio-group">
7   <span class="radio-group-text">Женский</span>
8 </div>
```

# ДОБАВИМ ::before

```
1 .radio-group-text::before {  
2   content: "";  
3   display: inline-block;  
4  
5   width: 20px;  
6   height: 20px;  
7  
8   background-image: url("https://netology-code.github.io/resources/pics/radio-no-active.  
9   background-repeat: no-repeat;  
10  background-position: center center;  
11  background-size: 20px;  
12 }
```

Выберите пол:

- Мужской  Женский

[Демо](#)

# Свойство vertical-align

Свойство `vertical-align` позволяет выровнять текст, строчные и строчно-блочные элементы по вертикали относительно друг друга. Для этого каждому строчно-блочному элементу нужно указать нужное значение свойства. У свойства есть следующие значения:

- `baseline`
- `top`
- `middle`
- `bottom`

# baseline

Родительский контейнер

Повседневная практика показывает, что дальнейшее развитие различных форм деятельности позволяет оценить значение позиций, занимаемых участниками в отношении поставленных задач.

Повседневная практика показывает, что дальнейшее развитие различных форм деятельности позволяет оценить значение позиций, занимаемых участниками в отношении поставленных задач. Значимость этих проблем настолько очевидна, что сложившаяся структура организации в значительной степени обуславливает создание системы обучения кадров, соответствует насущным потребностям.

vertical-align: baseline;

При этом значения строчно-блочные элементы выравниваются по последней строке блоков. Также это значение является значением по умолчанию.

# top

Родительский контейнер

Повседневная практика показывает, что дальнейшее развитие различных форм деятельности позволяет оценить значение позиций, занимаемых участниками в отношении поставленных задач.

Повседневная практика показывает, что дальнейшее развитие различных форм деятельности позволяет оценить значение позиций, занимаемых участниками в отношении поставленных задач. Значимость этих проблем настолько очевидна, что сложившаяся структура организации в значительной степени обуславливает создание системы обучения кадров, соответствует насущным потребностям.

vertical-align: top;

При этом значения строчно-блочные элементы выравниваются по верхней границе блоков.

# middle

Родительский контейнер

Повседневная практика показывает, что дальнейшее развитие различных форм деятельности позволяет оценить значение позиций, занимаемых участниками в отношении поставленных задач.

Повседневная практика показывает, что дальнейшее развитие различных форм деятельности позволяет оценить значение позиций, занимаемых участниками в отношении поставленных задач. Значимость этих проблем настолько очевидна, что сложившаяся структура организации в значительной степени обуславливает создание системы обучения кадров, соответствует насущным потребностям.

vertical-align: middle;

При этом значения строчно-блочные элементы выравниваются по середине блоков.

# bottom

Родительский контейнер

Повседневная практика показывает, что дальнейшее развитие различных форм деятельности позволяет оценить значение позиций, занимаемых участниками в отношении поставленных задач.

Повседневная практика показывает, что дальнейшее развитие различных форм деятельности позволяет оценить значение позиций, занимаемых участниками в отношении поставленных задач. Значимость этих проблем настолько очевидна, что сложившаяся структура организации в значительной степени обуславливает создание системы обучения кадров, соответствует насущным потребностям.

vertical-align: bottom;

При этом значения строчно-блочные элементы выравниваются по нижней границе блоков.

# ДОВЕРСТАЕМ РАДИОКНОПКИ

Выровним кружочек и текст по середине, добавив свойство vertical-align.

```
1 .radio-group-text::before {  
2   display: inline-block;  
3   vertical-align: middle;  
4 }
```

Добавим отступ между кружком и надписью.

```
1 .radio-group-text::before {  
2   margin-right: 8px;  
3 }
```

# ДОВЕРСТАЕМ РАДИОКНОПКИ

Расположим элементы в одну строку, зададим нужный размер текста и отступ.

```
1 .radio-group {  
2   display: inline-block;  
3   font-size: 15px;  
4 }  
5  
6 .radio-group:nth-of-type(n+2) {  
7   margin-left: 15px;  
8 }
```

Выберите пол:

Мужской      Женский

[Демо](#)

# ПСЕВДОКЛАСС :checked

Браузер дает возможность стилизовать элемент `input` с `type="radio"` только в активном состоянии и эта возможность называется псевдокласс `:checked`. Например, мы можем добавить обводку для элемента.

```
1 .radio:checked {  
2     outline: 3px solid #000000;  
3     outline-offset: 5px;  
4 }
```

# РЕЗУЛЬТАТ

мужской  женский

The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. The left pane displays the DOM tree, and the right pane shows the 'Styles' panel. A specific CSS rule for a checked radio button is highlighted with a red box:

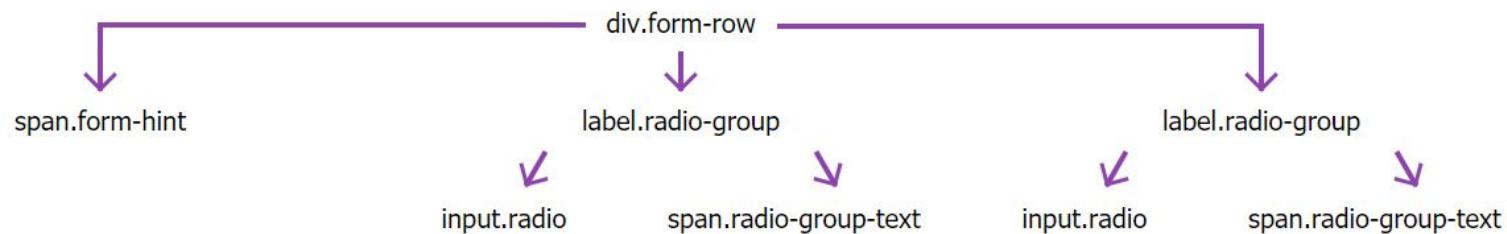
```
.radio:checked {  
    outline: 3px solid #000000;  
    outline-offset: 5px;  
}
```

The 'radio' class is defined in the user agent stylesheet, and the ':checked' pseudo-class is defined in both the user agent and page-specific stylesheets.

# СЕЛЕКТОРЫ + И ~

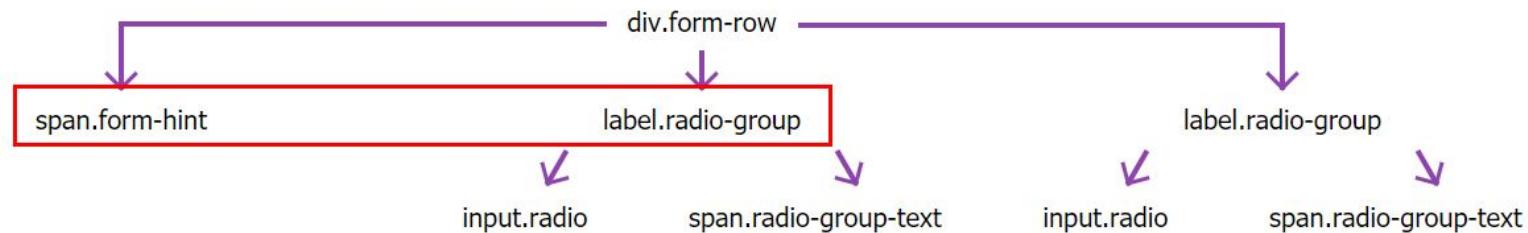
На предыдущей лекции мы узнали, что в верстке есть родители, дети и предки. Но у элементов также есть братья или сестры.

Рассмотрим структуру элемента `div` с классом `form-row`, в котором находятся радиокнопки.



# СОСЕДНИЙ СЕЛЕКТОР

Соседним селектором называют группу из двух элементов, когда они следуют непосредственно друг за другом в коде. Например, в нашем коде есть пара элементов `span.form-hint` и `label.radio-group`.

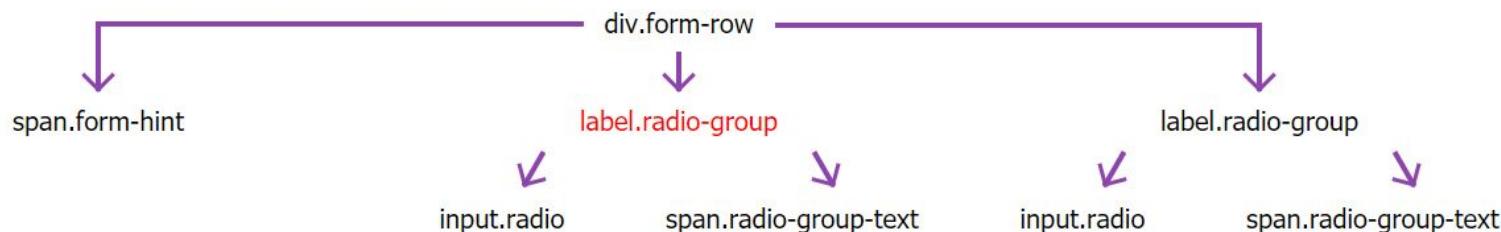


# СЕЛЕКТОР +

Из-за того что элемент `label.radio-group` идет сразу после элемента `span.form-hint`, мы можем воспользоваться соседним селектором. Для это между двумя элементами надо поставить знак `+`.

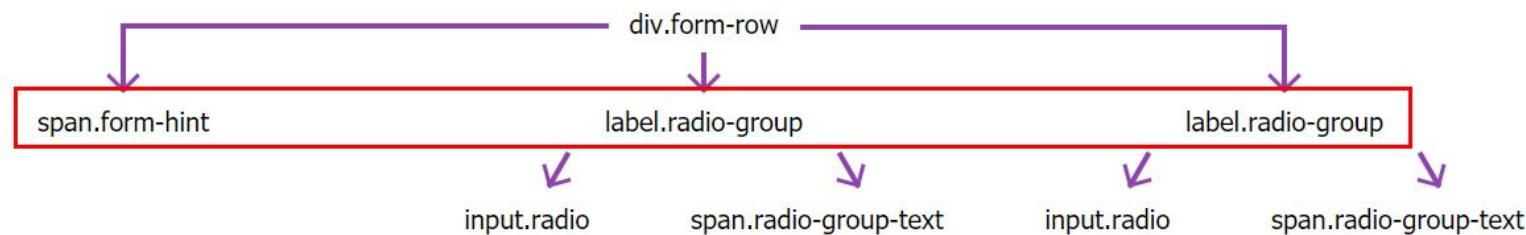
```
1 | span.form-hint + label.radio-group {  
2 |   color: #ff0000;  
3 | }
```

Браузер применит свойство `color` только для тех элементов `label`, у которых перед ними стоит элемент `span` с классом `form-hint`.



# РОДСТВЕННЫЕ СЕЛЕКТОРЫ

Родственным селектором называют группу элементов, которые имеют одного общего родителя. В нашем коде это элементы `span.form-hint` и два элемента `label.radio-group`.

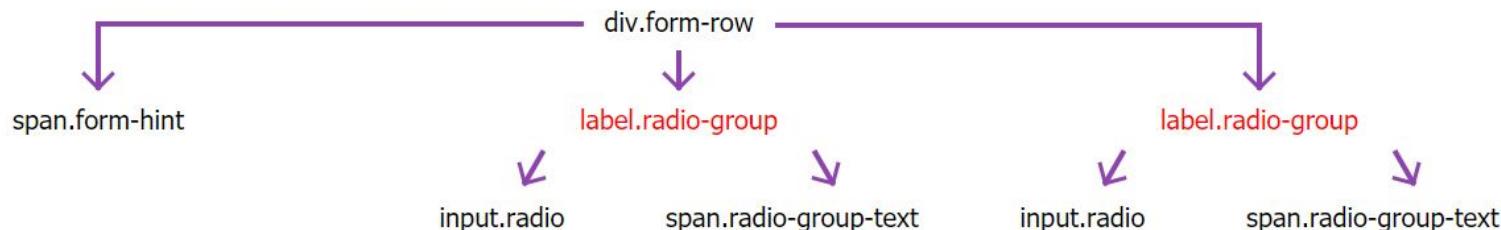


# СЕЛЕКТОР ~

Из-за того что у них один общий родитель, мы можем стилизовать, например, все элементы `label`, которые идут после элемента `span` с классом `form-hint`. Для этого используем символ `~`.

```
1 | span.form-hint ~ label.radio-group {  
2 |   color: #ff0000;  
3 | }
```

Браузер применит свойство `color` только для тех элементов `label`, которые идут после элемента `span` с классом `radio-group`.



# МЕНЯЕМ background-image у ::before

Мы можем использовать либо соседний селектор, либо родственный. В этом случае нет какой-либо разницы.

```
1 .radio:checked ~ .radio-group-text::before {  
2   background-image: url("https://netology-code.github.io/resources/pics/radio-active.svg");  
3 }
```

Выберите пол:

 Мужской      Женский

[Демо](#)

# position: absolute

Когда мы указываем свойство `position` со значением `absolute`, то элемент становится блочным, но с определенными свойствами.

- Свойства `width` и `height` рассчитываются по содержимому элемента;

элемент с `position: absolute`

элемент с длинным текстом и с `position: absolute`

- Для свойств `width` и `height` можно установить значения;

элемент с `position: absolute`

элемент `span`

# position: absolute

- Элемент с `position: absolute` не видим для родителя;

Родительский контейнер

элемент `div` без `position: absolute`

По изображению видно, что высота контентной области (свойство `height`) родителя рассчитывается в зависимости от высоты дочернего блочного элемента. Но когда у дочернего элемента есть `position: absolute`, то он становится не видимым для своего родителя.

Родительский контейнер

элемент `div` с `position: absolute`

# position: absolute

- Элемент с `position: absolute` не видим для соседних элементов.

По умолчанию два элемента `div` располагаются друг под другом, как показано на следующем изображении:

элемент `div` без `position: absolute`

Соседний элемент

Но когда элементу добавлено `position: absolute`, то он становится невидим для своих соседних элементов.

элемент `div` с `position: absolute`

# position: absolute для input СТИПОМ radio

```
1 .radio {  
2     position: absolute;  
3 }
```

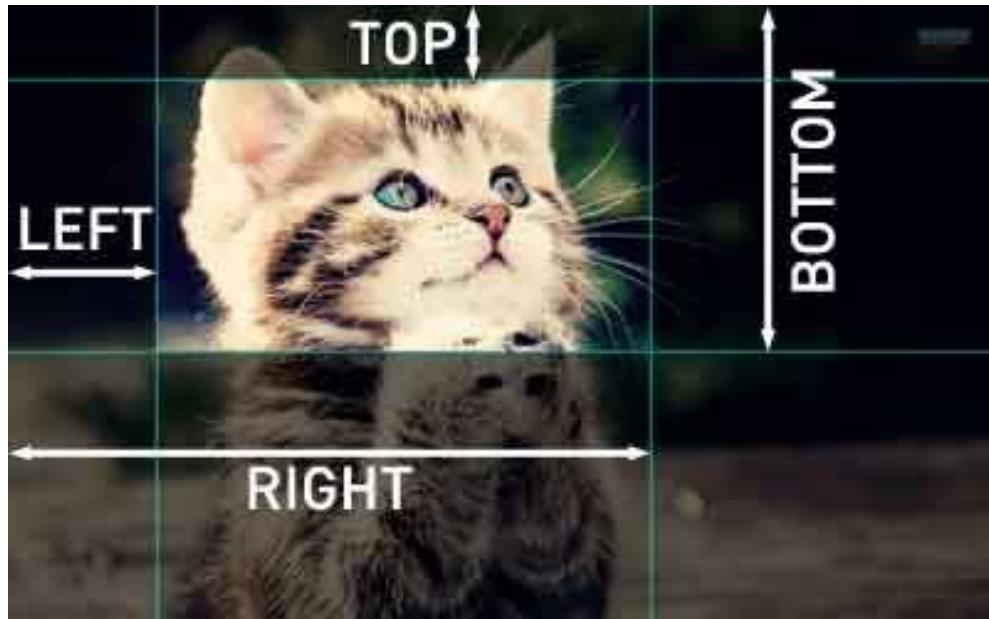
Выберите пол:

Мужской  Женский

# СВОЙСТВО `clip`

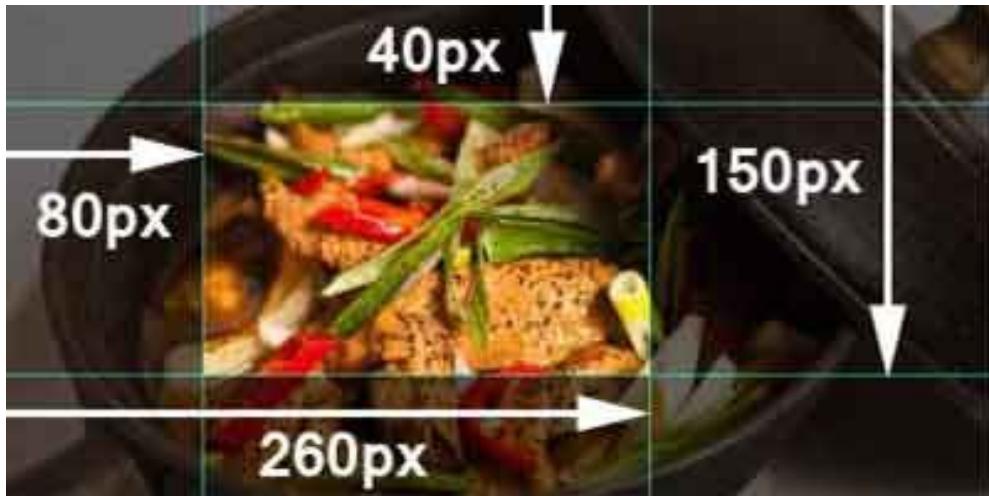
Свойство `clip` позволяет указать область элемента, которую нужно отобразить браузеру. Все, что не входит в эту область, отображаться не будет. Важно, чтобы у элемента было установлено `position: absolute`.

```
clip: rect(top, right, bottom, left);
```



## ПРИМЕР

```
clip: rect(40px, 260px, 150px, 80px);
```



# СКРЫВАЕМ input

```
1 .radio {  
2   position: absolute;  
3   clip: rect(1px, 1px, 1px, 1px);  
4   height: 1px;  
5   width: 1px;  
6 }
```

Выберите пол:

Мужской  Женский

[Демо](#)

# ДОБАВЛЯЕМ `outline` ДЛЯ ФОКУСА

Сейчас есть такой код:

```
1 .field:focus, .select:focus, .button:focus {  
2   outline: 2px solid #c53b2d;  
3   outline-offset: 5px;  
4 }
```

Стилизуем соседний элемент `span` с классом `radio-group-text`, а не сам элемент `input`.

```
1 .field:focus, .select:focus, .radio:focus ~ .radio-group-text, .button:focus {  
2   outline: 2px solid #c53b2d;  
3   outline-offset: 5px;  
4 }
```

# РЕЗУЛЬТАТ

Выберите пол:



Мужской



Женский

[Демо](#)

# ВЕРНЕМСЯ К МАКЕТУ

Введите ваше имя

Например, Иван

Введите ваш email

Например, ivan@gmail.com

Выберите город проживания

Москва



Выберите пол:

Мужской  Женский

Вы согласны на обработку персональных данных

Отправить

# type="checkbox"

Часто мы можем встретить элемент, с помощью которого мы соглашаемся на что-то. Например, на странице регистрации Reebok мы даем согласие, что ООО Адидас может связаться с нами.

НОВЫЙ ПАРОЛЬ \*

⚡

Пожалуйста, убедитесь, что ваш пароль содержит минимум одну букву, одну цифру и как минимум 8 символов

Правообладателем товарного знака Reebok является adidas group. Настоящим я соглашаюсь с тем, что ООО "АДИДАС" может связываться со мной по почте, электронной почте, смс, телефону и с помощью других средств связи в целях маркетинга, рекламы и изучения мнений. Для того, чтобы получать только интересную мне информацию, я соглашаюсь на анализ и обработку истории моего взаимодействия с ООО "АДИДАС". Я согласен на обработку моих персональных данных, включая трансграничную передачу и передачу третьим лицам, уполномоченным ООО "АДИДАС" для осуществления целей маркетинга, рекламы и изучения мнений группой компаний adidas. Я прочитал Политику Конфиденциальности и согласен с ее положениями. Я понимаю, что могу отзоваться свое согласие, следя по специальной ссылке в сообщениях от adidas.

# ДОБАВИМ <input type="checkbox">

```
1 <div class="form-row">
2   <label class="checkbox-group">
3     <input type="checkbox" class="checkbox" required>
4     <span class="checkbox-text">Вы согласны на обработку персональных данных</span>
5   </label>
6 </div>
```

[Демо](#)

# РЕЗУЛЬТАТ

Введите ваш email

Например, ivan@gmail.com

Выберите город проживания

Москва

Выберите пол:

Мужской  Женский

Вы согласны на обработку персональных данных

Отправить

[Демо](#)

# МЕНЯЕМ РАЗМЕР ТЕКСТА

```
1 .checkbox-group {  
2   font-size: 15px;  
3 }
```

Выберите пол:

Мужской  Женский

Вы согласны на обработку персональных данных

# СТИЛИЗУЕМ ЧЕКБОКС

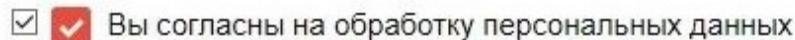
```
1 .checkbox-text::before {  
2   content: "";  
3   display: inline-block;  
4   vertical-align: middle;  
5   margin-right: 8px;  
6  
7   width: 20px;  
8   height: 20px;  
9  
10  background-image: url("https://netology-code.github.io/resources/pics/checkbox2-unchecke  
11  background-repeat: no-repeat;  
12  background-position: center center;  
13  background-size: 20px;  
14 }
```

Вы согласны на обработку персональных данных

# СТИЛИЗУЕМ ЧЕКБОКС

Для стилизации активного состояния элемента будем использовать псевдокласс `:checked`.

```
1 .checkbox:checked ~ .checkbox-text::before {  
2   background-image: url("https://netology-code.github.io/resources/pics/checkbox2-checked.png");  
3 }
```



Скрываем стандартный элемент.

```
1 .radio, .checkbox {  
2   position: absolute;  
3   clip: rect(1px, 1px, 1px, 1px);  
4   height: 1px;  
5   width: 1px;  
6 }
```

# СТИЛИЗУЕМ ЧЕКБОКС

И последним шагом будет добавление обводки для элемента при состоянии `:focus`.

```
1 .field:focus, .select:focus, .radio:focus ~ .radio-group-text,  
2 .checkbox:focus ~ .checkbox-text, .button:focus {  
3   outline: 2px solid #c53b2d;  
4   outline-offset: 5px;  
5 }
```

# РЕЗУЛЬТАТ

Введите ваше имя

Например, Иван

Введите ваш email

Например, ivan@gmail.com

Выберите город проживания

Москва

Выберите пол:

Мужской  Женский



Вы согласны на обработку персональных данных

Отправить

---

# ИТОГИ

- `display: inline-block` дает элементам свойства как строчных, так и блочных элементов;
- Элементы `select` и `option` создают список. `select` создает элемент в виде раскрывающегося списка, `option` является одним пунктом элемента `select`;
- Свойство `background` позволяет стилизовать фон элементов;
- Элемент формы `input type="radio"` позволяет создать переключатель;
- Псевдоэлементы `before` и `after` – это элементы, которые добавлены внутрь родительского элемента при помощи CSS;
- Для выбора соседних или родственных селекторов используем селекторы `+` и `~`;

---

# ИТОГИ

- `position: absolute` задает элементу абсолютное позиционирование, при котором элемент "выпадает из потока", размеры элемента по умолчанию зависят от содержимого, но их также можно задать.
- Свойство `clip` позволяет указать область элемента, которую нужно отобразить браузеру.
- Элемент формы `input type="checkbox"` позволяет создать флажок;



Ваши вопросы?

АЛЕКСАНДР ФИТИСКИН

f [fb.me/afitiskin](https://fb.me/afitiskin)