



# CSS

Cascading Style  
Sheets  
(Каскадные  
таблицы стилей.)



# ПЛАН

**1**

Выравнивание, списки, таблицы.

**2**

Высота, ширина, max-width, min-width.

**3**

Overflow, псевдо-классы,  
псевдо-элементы.

# ВЫРАВНИВАНИЕ

## Выравнивание блочных элементов

Выравнивание блочных элементов с использованием `margin` и `auto` в CSS осуществляется путем установки `auto` для `margin-left` и `margin-right`. Это приводит к автоматическому распределению свободного пространства между левым и правым краями элемента, что в результате центрирует его горизонтально.



1. Установка **display: block**: Убедитесь, что элемент, который вы хотите выровнять, имеет установленное свойство `display: block`. Блочные элементы занимают всю доступную ширину контейнера и могут быть выравнены горизонтально.
2. Установка отступов: Для выравнивания элемента по центру горизонтали, нужно установить левый и правый отступ на `auto`:
3. Когда значения `margin-left` и `margin-right` установлены на `auto`, браузер автоматически распределит доступное горизонтальное пространство между левым и правым отступами элемента. Это приводит к центрированию элемента по горизонтали.
4. Выравнивание с использованием `margin: auto` работает только для выравнивания по горизонтали. Для вертикального выравнивания блочных элементов, особенно внутри родительского контейнера, можно использовать другие методы

**Примечание:** Выравнивание по центру не будет иметь никакого эффекта, если свойство `width` не установлено (или установлено на 100 пикселей).

```
1  .center {  
2      margin: auto 0;  
3      width: 60%;  
4      border: 3px solid #73AD21;  
5      padding: 10px;  
6  }
```

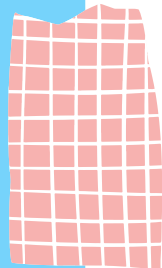
# TEXT-ALIGN

Свойство text-align применяется к контейнеру и определяет горизонтальное выравнивание текста внутри элемента. Оно может принимать следующие значения:

- left: Выравнивание текста по левому краю элемента.
- right: Выравнивание текста по правому краю элемента.
- center: Центрирование текста по горизонтали.
- justify: Выравнивание текста по ширине элемента, создавая равные промежутки между словами.

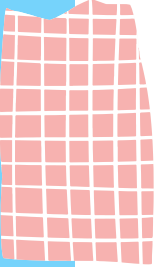
```
.center-text {  
  text-align: center;  
  border: 3px solid green;  
}
```

# **WIDTH, HEIGHT, MAX-WIDTH, MIN-WIDTH**



width и height устанавливают точные значения для высоты и ширины элемента, в то время как max-width позволяет элементу растягиваться до определенного предела. Эти свойства могут быть полезными при создании адаптивного дизайна и контроле размеров элементов на веб-странице.

**width:** Свойство width устанавливает ширину элемента. Как и height, вы можете задавать ширину в пикселях (px), процентах (%) или других единицах измерения.



**height:** Свойство height устанавливает высоту элемента. Вы можете задавать высоту в пикселях (px), процентах (%) или других доступных единицах измерения.

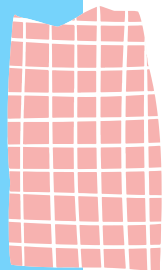
**max-width:** Свойство max-width устанавливает максимально допустимую ширину элемента. Это ограничивает расширение элемента, если его содержимое или другие факторы пытаются увеличить его ширину.

В этом примере элементу с классом .element будет разрешено иметь ширину до 500 пикселей. Однако, если его содержимое или другие факторы пытаются увеличить ширину сверх, то она будет ограничена максимальным значением 500 пикселей.

```
.width {  
  height: 100px;  
  width: 290px;  
  background-color: powderblue;  
}
```

```
.element {  
  max-width: 500px;  
  height: 100px;  
  background-color: powderblue;  
}
```





Свойство `max-width` особенно полезно при создании адаптивного дизайна, когда вы хотите, чтобы элемент сохранял ограниченную максимальную ширину на различных устройствах или при изменении размеров окна браузера.

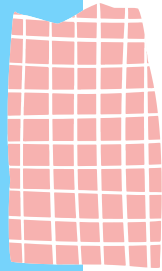
Приоритет `max-width` и `width`: Если одновременно заданы `max-width` и `width`, то значение `max-width` будет действовать, пока оно не будет превышено значением `width`. Если `width` больше, чем `max-width`, то элемент будет иметь ширину, равную `max-width`.

В этом примере элементу с классом `.max-width` задана ширина `600px`, но так как она больше `max-width` (`500px`), элемент будет иметь ширину `500px`. Обратите внимание, что если `width` меньше или равно `max-width`, то `max-width` не будет влиять на ширину элемента.

```
.max-width {  
  width: 600px;  
  max-width: 500px;  
  background-color: sandybrown;  
}
```



**min-width:** Свойство min-width в CSS устанавливает минимально допустимую ширину элемента. Это означает, что элемент не будет сжиматься до размера меньше, указанного значения min-width.



Свойство min-width особенно полезно при создании адаптивного дизайна, когда вы хотите, чтобы элемент сохранял свою минимальную ширину на разных устройствах или при изменении размеров окна браузера.

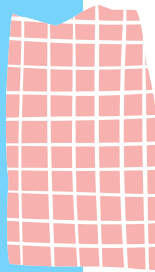
Приоритет min-width и width: Если одновременно заданы min-width и width, то значение width будет действовать, пока оно не превысит значение min-width. Если width меньше, чем min-width, то элемент будет иметь ширину, равную min-width.

В этом примере элементу с классом .element задана ширина 150px, но так как она меньше min-width (200px), элемент будет иметь ширину 200px.

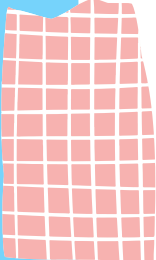
Обратите внимание, что если width больше или равно min-width, то min-width не будет влиять на ширину элемента.

```
.min-width {  
  width: 150px;  
  min-width: 200px;  
  height: 100px;  
  background-color: sandybrown;  
}
```

# СПИСКИ

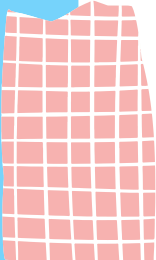


Списки в CSS используются для стилизации маркированных (ненумерованных) и нумерованных списков на веб-странице. CSS предоставляет ряд свойств, которые позволяют изменять внешний вид списков.



**list-style-type:** Свойство `list-style-type` определяет тип маркера для маркированных списков. Оно может принимать различные значения, такие как `disc` (круглый маркер по умолчанию), `circle` (пустой круг), `square` (квадрат), `none` (отключение маркера), `upper-roman`, `lower-alpha` и другие.

```
ul.a {  
    list-style-type: circle;  
}
```



Свойство `list-style-type:none` также может быть использован для удаления маркеров/кружков. Обратите внимание, что в списке также есть поля и отступы по умолчанию. Чтобы удалить или добавить `margin:0` или `padding:0` в `<ul>` или `<ol>`:

```
ul.b {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
}
```

# ТАБЛИЦЫ

Внешний вид HTML-таблицы может быть значительно улучшен с помощью CSS:

## Границы таблицы

Чтобы указать границы таблицы в CSS, используйте свойство `border`.

## Таблица во всю ширину

Приведенная выше таблица в некоторых случаях может показаться небольшой. Если вам нужна таблица, которая должна занимать весь экран (во всю ширину), добавьте `width: 100%` к элементу `<таблица>`.

## Свернуть границы таблицы

Свойство `border-collapse` задает, следует ли сворачивать границы таблицы в единую границу.

## Высота таблицы

Высота определяется свойством `height`.

# CSS OVERFLOW

Свойство `overflow` в CSS позволяет управлять отображением содержимого элемента, когда его размеры превышают размеры контейнера.

Свойство `overflow` полезно для создания контейнеров с фиксированной областью просмотра, скрывания лишнего содержимого, а также для добавления полос прокрутки для просмотра большого или не помещающегося содержимого.

Свойство `overflow` может принимать следующие значения:

1. **visible** (по умолчанию): При значении `visible`, содержимое элемента может выходить за пределы контейнера. Это означает, что содержимое может перекрывать другие элементы на странице.
2. **hidden**: Значение `hidden` обрезает содержимое элемента, которое не помещается внутри контейнера, и скрывает его. То есть, если содержимое выходит за пределы контейнера, оно не будет видно для пользователя.
3. **scroll**: При значении `scroll`, появляются горизонтальная и вертикальная полосы прокрутки внутри контейнера, даже если содержимое помещается внутри него. Это позволяет пользователю прокручивать содержимое, если оно не полностью видимо.
4. **auto**: Значение `auto` автоматически определяет, нужно ли добавлять полосы прокрутки внутри контейнера, исходя из необходимости. Если содержимое помещается в контейнер, полосы прокрутки не появляются. Если же содержимое не помещается, появляются полосы прокрутки для прокрутки содержимого.

## Переполнение по горизонтали и вертикали

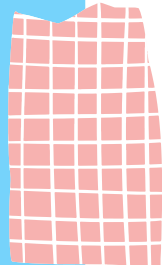
Свойства `overflow-x` и `overflow-y` может изменить переполнение содержимого по горизонтали или по вертикали (одновременно):

Свойство `overflow-x` указывает, что делать с левый/правый краями содержания.

Свойство `overflow-y` указывает, что делать с верхним/нижним краями содержания.

# ПСЕВДО-КЛАССЫ

Псевдо-класс используется для определения особого состояния элемента.



Псевдо-классы в CSS используются для выбора и стилизации определенных состояний или частей элементов веб-страницы. Они позволяют применять стили к элементам на основе их состояния, позиции в структуре или взаимодействия с пользователем.



:hover: Псевдо-класс :hover применяет стили к элементу, когда указатель мыши находится над ним.

```
.hover:hover {  
    color: red;  
}
```

:active: Псевдо-класс :active применяет стили к элементу во время активного состояния, когда он нажат левой кнопкой мыши.

В этом примере фон кнопки будет окрашен в синий цвет во время ее нажатия.

```
button:active {  
    background-color: blue;  
}
```

:focus: Псевдо-класс :focus применяет стили к элементу, когда он находится в фокусе, то есть активирован и готов к взаимодействию

```
input:focus {  
    border: 2px solid green;  
}
```

:first-child соответствует заданному элементу, который является первым ребенком элемента, другого элемента.

```
p:first-child {  
    color: blue;  
}
```

:last-child: Псевдо-класс :last-child выбирает последний потомок выбранного элемента внутри его родительского контейнера.

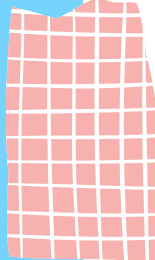
:nth-child(n): Псевдо-класс :nth-child(n) позволяет выбрать элементы в соответствии с их позицией внутри родительского контейнера.

```
table tr:nth-child(odd) {  
    background-color: salmon;  
}
```

Селектор	Пример	Описание
<u>:active</u>	a:active	Устанавливает активную ссылку
<u>:checked</u>	input:checked	Устанавливает каждый элемент проверки в <input>
<u>:disabled</u>	input:disabled	Устанавливает каждый элемент отключение в <input>
<u>:empty</u>	p:empty	Устанавливает каждый элемент <p>, который не имеет детей
<u>:enabled</u>	input:enabled	Устанавливает каждый элемент включение в <input>
<u>:first-child</u>	p:first-child	Устанавливает каждый элемент <p>, является первым ребенком своего родителя
<u>:first-of-type</u>	p:first-of-type	Устанавливает каждый элемент <p>, является первым элементом <p> своего родителя
<u>:focus</u>	input:focus	Устанавливает элемент <input>, который имеет фокус
<u>:hover</u>	a:hover	Выбирает ссылку при наведение курсором
<u>:in-range</u>	input:in-range	Выбирает элементы <input> в пределах указанного диапазона значений
<u>:invalid</u>	input:invalid	Выбирает все элементы <input> с недопустимым значением
<u>:lang(<i>language</i>)</u>	p:lang(it)	Выбирает каждый элемент <p> со значением атрибута lang, начиная с "it"
<u>:last-child</u>	p:last-child	Выбирает каждый элемент <p> что является последним ребенком своего родителя
<u>:last-child</u>	<u>:last-of-type</u>	Выбирает каждый элемент <p>, который является последним элементом <p> своего родителя
<u>:link</u>	a:link	Выбирает все непосещенные ссылки
<u>:not(selector)</u>	:not(p)	Выбирает каждый элемент которого нет в элементе <p>
<u>:nth-child(n)</u>	p:nth-child(2)	Выбирает каждый элемент <p>, что это второй ребенок своего родителя
<u>:nth-last-child(n)</u>	p:nth-last-child(2)	Выбирает каждый элемент <p>, который является вторым ребенком своего родителя, считая от последнего ребенка

<u>:nth-child(n)</u>	p:nth-child(2)	Выбирает каждый элемент <p>, что это второй ребенок своего родителя
<u>:nth-last-child(n)</u>	p:nth-last-child(2)	Выбирает каждый элемент <p>, который является вторым ребенком своего родителя, считая от последнего ребенка
<u>:nth-last-of-type(n)</u>	p:nth-last-of-type(2)	Выбирает каждый элемент <p> вторичного элемента <p> своего родителя, считая от последнего ребенка
<u>:nth-of-type(n)</u>	p:nth-of-type(2)	Выбирает каждый элемент <p> вторичного элемента <p> своего родителя
<u>:only-of-type</u>	p:only-of-type	Выбирает каждый элемент <p> , который является единственным элементом <p> своего родителя
<u>:only-child</u>	p:only-child	Выбирает каждый элемент <p>, который является единственным ребенком своего родителя
<u>:optional</u>	input:optional	Выбирает элемент <input> без атрибута "required"
<u>:out-of-range</u>	input:out-of-range	Выбирает элемент <input> со значением, выходящим за пределы указанного диапазона
<u>:read-only</u>	input:read-only	Выбирает элемент <input> с определенным атрибутом "readonly"
<u>:read-write</u>	input:read-write	Выбирает элемент <input> без атрибута "readonly"
<u>:required</u>	input:required	Выбирает элемент <input> с определённым атрибутом "required"
<u>:root</u>	root	Выбирает корневой элемент документа
<u>:target</u>	#news:target	Выбирает текущий активный #news элемент (нажали на URL-адрес, содержащий имя якоря)
<u>:valid</u>	input:valid	Выбирает все элементы <input> с допустимым значением
<u>:visited</u>	a:visited	Выбирает все посещенные ссылки

# ПСЕВДО-ЭЛЕМЕНТЫ



Псевдо-элементы в CSS позволяют создавать и стилизовать внутренние части элементов или добавлять дополнительные элементы на страницу без необходимости изменять HTML-структуру. Они представлены специальными ключевыми словами, которые добавляются после селектора и обозначаются двумя двоеточиями (::).

**::first-letter:** Псевдо-элемент ::first-letter применяет стили к первой букве первого слова внутри выбранного элемента.

```
.first::first-letter {  
    color: #ff0000;  
    font-size: xx-large;  
}
```

**::first-line:** Псевдо-элемент ::first-line применяет стили к первой строке текста внутри выбранного элемента.

```
.second::first-line {  
    text-transform: uppercase;  
    font-weight: bold;  
}
```

**::selection:** Псевдо-элемент ::selection в CSS позволяет стилизовать выделенный текст на веб-странице. Он применяет стили к выделенному фрагменту текста пользователем с помощью мыши или другими способами.

```
::selection {  
    color: red;  
    background: yellow;  
}
```



**::before:** Псевдо-элемент ::before позволяет вставить содержимое перед указанным элементом. Вы можете использовать его для добавления дополнительных элементов или декоративных элементов перед выбранным элементом.

```
.text::before {  
    content: "@@";  
    font-weight: bold;  
}
```

**::after:** Псевдо-элемент ::after позволяет вставить содержимое после указанного элемента. Он часто используется для создания дополнительных элементов или декоративных элементов после выбранного элемента.

```
.text-two::after {  
    content: "!!!";  
    color: red;  
}
```

**::selection:** Псевдо-элемент ::selection в CSS позволяет стилизовать выделенный текст на веб-странице. Он применяет стили к выделенному фрагменту текста пользователем с помощью мыши или другими способами.



Селектор	Пример	Описание примера
<u>::after</u>	p::after	Вставить что-то после содержания каждого элемента <p>
<u>::before</u>	p::before	Вставить что-то перед содержанием каждого элемента <p>
<u>::first-letter</u>	p::first-letter	Выбирает первую букву каждого элемента <p>
<u>::first-line</u>	p::first-line	Выбирает первую строку каждого элемента <p>
<u>::marker</u>	::marker	Выбирает маркеры элементов списка
<u>::selection</u>	p::selection	Выбирает часть элемента, выбранного пользователем

# ПСЕВДО-КЛАССЫ И ПСЕВДО-ЭЛЕМЕНТЫ В **CSS** ОТЛИЧАЮТСЯ ПО СВОЕМУ НАЗНАЧЕНИЮ И СПОСОБУ ПРИМЕНЕНИЯ СТИЛЕЙ.

## Псевдо-классы:

- Псевдо-классы применяются к элементам на основе их состояния, позиции в структуре или взаимодействия с пользователем.
- Они указывают на конкретное состояние или свойство элемента, такое как `:hover`, `:active`, `:focus`, `:first-child` и другие.
- Псевдо-классы обозначаются символом двоеточия (`:`) и добавляются после селектора элемента.
- Они используются для выбора и стилизации элементов на основе их состояния или положения в документе.

## Псевдо-элементы:

- Псевдо-элементы используются для создания и стилизации внутренних частей элементов или добавления дополнительных элементов на страницу без изменения HTML-структуры.
- Они представлены специальными ключевыми словами, такими как `::before`, `::after`, `::first-letter`, `::first-line` и другие.
- Псевдо-элементы обозначаются двумя двоеточиями (`::`) и добавляются после селектора элемента.
- Они используются для добавления и стилизации внутренних элементов или декоративных элементов внутри других элементов.



**THANK  
YOU!**

Have a  
great day  
ahead.