

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационной безопасности

Кафедра инфокоммуникационных технологий

ВЕБ-ТЕХНОЛОГИИ В ИНФОКОММУНИКАЦИЯХ

Создание каскадной таблицы стилей CSS



Минск 2022

Содержание

Лабораторная работа №4 Создание каскадной таблицы стилей CSS	4
Свойства внешнего контура.....	4
Изменение размера блоков: свойство resize	6
Стилизация курсора: свойство cursor	7
Цвет каретки вставки: свойство caret-color	7
CSS-градиент	7
Линейный градиент linear-gradient().....	7
Радиальный градиент radial-gradient()	9
Повтор градиента	11
Кроссбраузерный градиент.....	11
CSS3-рамка	11
Закругление углов с помощью border-radius.....	11
Рамки-изображения border-image	12
CSS3-оформление текста	12
Оформление линии: подчеркивание, обводка и зачеркивание.....	12
Тень текста: свойство text-shadow	13
CSS3-тень блока	14
CSS3-переходы	14
Свойство transition-property	14
Продолжительность перехода transition-duration	15
Функция перехода transition-timing-function	15
Задержка перехода transition-delay	16
Краткая запись перехода	16
Плавный переход нескольких свойств	16
CSS3-трансформации.....	16
Функции 2D-трансформации transform	17
Точка трансформации transform-origin.....	17
Множественные трансформации	17
CSS3-анимация	17
Ключевые кадры	17
Название анимации: свойство animation-name	19
Продолжительность анимации: свойство animation-duration	19
Временная функция: свойство animation-timing-function	20
Свойства animation-iteration-count, animation-direction, animation-play-state, animation-delay, animation-fill-mode	20
Краткая запись анимации: свойство animation	20
Множественные анимации	20
CSS flexbox	20
Flex-контейнер	22
Flex-элементы	22
Порядок отображения flex-элементов и ориентация.....	23
Гибкость flex-элементов	25
Выравнивание.....	27
CSS3 columns	32
Количество и ширина колонок	32
Промежутки между колонками и разделительные линии	34
Разрыв колонок	36
Охват колонок: свойство column-span	36
Заполнение колонок содержимым: свойство column-fill	37
Переполнение	37
CSS3 3D-трансформации	37
Свойство transform-style	38
Свойство perspective	39
Свойство perspective-origin	40
Свойство backface-visibility	41

Функции 3D-трансформации.....	41
CSS3-медиавыражения	42
CSS3-фильтры	43
CSS Grid	45
Концепция сетки и основные понятия	45
Контейнер-сетка.....	46
Определение сетки	47
Строки и столбцы	47
Именованные области	49
Краткая запись явной сетки	50
Неявная сетка	51
Краткая запись сетки	53
Элементы сетки.....	53
Размещение и переупорядочивание элементов сетки	53
Краткая запись свойств размещения элементов сетки	55
Промежутки между элементами сетки.....	56
CSS3-шрифты	56
Кернинг: свойство font-kerning	57
Лигатуры: свойство font-variant-ligatures	57
Преобразование в заглавные буквы: свойство font-variant-caps.....	57
Форматирование цифр: свойство font-variant-numeric.....	57
Общее сокращение для рендеринга шрифтов: свойство font-variant	57
CSS3-переполнение.....	58
Типы переполнения	58
Прокрутка и обрезка переполнения: свойства overflow-x, overflow-y и overflow ..	58
Автоматическое многоточие.....	60
CSS3-способы письма	60
Направление текста в html-документах	60
Направление вдоль линии строки и двунаправленность	62
Вертикальное письмо.....	62
CSS-генераторы	64
Валидация CSS.....	67
Задание к лабораторной работе №4.....	68

Лабораторная работа №4

Создание каскадной таблицы стилей CSS

Свойства внешнего контура

Контуры позволяют выделять активные элементы интерфейса, такие как, кнопки, поля формы, карты изображений и т.п.

Браузеры часто отображают контуры элементов в состоянии `:focus`, поэтому не рекомендуется делать контур невидимым для таких элементов без альтернативного механизма выделения.

Контур всегда находится сверху и не влияет на положение или размер блока или любых других блоков. Поэтому отображение или скрытие контуров не вызывает перекомпоновку.

Части контура не обязательно должны быть прямоугольными, например, он повторяет закругленные углы элемента.

Краткая запись внешнего контура: свойство `outline`

Свойство представляет краткую запись свойств `outline-color`, `outline-style`, `outline-width`. Значение по умолчанию `outline: invert none medium`.

Свойство не наследуется.

```
outline: dotted;
outline: gold solid;
outline: inset thick;
outline: pink solid 2px;
outline: inherit;
outline: initial;
```

Толщина внешнего контура: свойство `outline-width`

Свойство `outline-width` задает толщину внешнего контура, принимает те же значения, что и свойство `border-width`.

Свойство не наследуется.

Значение	<code>outline-width</code>
длина	Значение задается в единицах длины.
<code>thin/medium/thick</code>	Длина, соответствующая этим значениям, не определена, но значения постоянны по всей веб-странице и <code>thin ≤ medium ≤ thick</code> . Браузер может сделать толщину зависимой от среднего размера шрифта. Значение по умолчанию <code>medium</code> .
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

```
outline-width: thin;
outline-width: medium;
outline-width: thick;
outline-width: 2px;
outline-width: 0.1em;
outline-width: inherit;
```

Узор внешнего контура: свойство outline-style

Свойство `outline-style` принимает те же значения, что и `border-style`, за исключением значения `hidden`.

Свойство не наследуется.

Значение	<code>outline-style</code>
<code>auto</code>	Значение позволяет браузеру отображать стиль контура по умолчанию или более богатым стилем, например контуром с закругленными краями с полупрозрачными внешними пикселями, который кажется светящимся. Также, браузеры могут рассматривать это значение как <code>solid</code> .
<code>none</code>	Отсутствие внешнего контура. Значение по умолчанию.
<code>dotted</code>	
<code>dashed</code>	
<code>solid</code>	
<code>double</code>	
<code>groove</code>	
<code>ridge</code>	
<code>inset</code>	
<code>outset</code>	
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

`outline-style: auto;
outline-style: none;
outline-style: dotted;
outline-style: dashed;
outline-style: solid;
outline-style: double;
outline-style: groove;
outline-style: ridge;
outline-style: inset;
outline-style: outset;
outline-style: inherit;
outline-style: initial;`

Цвет внешнего контура: свойство outline-color

Свойство `outline-color` позволяет установить цвет внешнего контура с помощью значений цвета и ключевого свойства `invert`.

Свойство не наследуется.

Значение	<code>outline-color</code>
<code>invert</code>	Значение выполнит инверсию цвета пикселей на экране. Это обеспечивает видимость границы фокуса независимо от цвета фона. Цвет линии (обычно черный) устанавливается автоматически, создавая контраст с основным содержимым. Если браузер не поддерживает данное значение, тогда значением свойства будет ключевое слово <code>currentColor</code> . Значение по умолчанию.
<code>цвет</code>	Значение принимает все форматы цвета свойства <code>color</code> .
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.

inherit	Наследует значение свойства от родительского элемента.
---------	--

```
outline-color: #f92525;
outline-color: rgb(30,222,121);
outline-color: blue;
outline-color: invert;
outline-color: inherit;
outline-color: initial;
```

Смещение внешнего контура: свойство outline-offset

По умолчанию контур рисуется начиная с края рамки элемента. Свойство **outline-offset** позволяет сместить контур от края границы на указанную величину.

Свойство не наследуется.

Значение	outline-offset
длина	Задает расстояние с помощью единиц длины – px / em . Отрицательное значение отображает рамку внутри элемента, положительное – снаружи элемента. Значение по умолчанию 0.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
outline-offset: 3px;
outline-offset: 0.2em;
outline-offset: inherit;
outline-offset: initial;
```

Изменение размера блоков: свойство resize

Свойство **resize** позволяет указать, может ли пользователь изменять размер элемента, и если да, то вдоль какой оси/осей. Свойство применяется к элементам, чье вычисляемое значение **overflow** отличается от **visible**.

Если для элемента установлено изменение размеров, в правом нижнем углу появляется треугольник, с помощью которого элемент можно растягивать в обеих направлениях. Уменьшение первоначальных размеров элемента не предусмотрено.

Браузер должен позволять пользователю изменять размер элемента без каких-либо других ограничений, кроме ограничений, накладываемых свойствами **min-width**, **max-width**, **min-height** и **max-height**.

Свойство не наследуется.

Значение	resize
none	Браузер не предоставляет механизм изменения размера элемента пользователем. Значение по умолчанию.
both	Браузер представляет механизм двунаправленного изменения размера, позволяющий пользователю регулировать как высоту, так и ширину элемента.
horizontal	Браузер представляет односторонний горизонтальный механизм изменения размера, позволяющий пользователю регулировать только ширину элемента.
vertical	Браузер представляет односторонний вертикальный механизм изменения размера, позволяющий пользователю регулировать только высоту элемента.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```

    resize: none;
    resize: both;
    resize: horizontal;
    resize: vertical;
    resize: block;
    resize: inline;
    resize: inherit;
    resize: initial;

```

Стилизация курсора: свойство cursor

Свойство `cursor` указывает тип курсора, который будет отображаться для устройства, когда точка доступа курсора находится в пределах границ элемента. Браузеры могут игнорировать свойство над собственными элементами управления, например, полосами прокрутки. Браузеры также могут игнорировать свойство `cursor` и отображать его по своему выбору, чтобы указать различные состояния пользовательского интерфейса, например, когда страница не отвечает или когда пользователь выделяет текст.

Свойство наследуется.

Описание свойства cursor.

Цвет каретки вставки: свойство caret-color

Символ каретки является видимым индикатором точки вставки в элементе, в который пользователь вставляет текст (и, возможно, другой контент). Свойство `caret-color` контролирует цвет этого видимого индикатора.

Свойство наследуется.

Значение	resize
<code>auto</code>	Браузеры используют <code>currentColor</code> . Браузеры также могут автоматически корректировать цвет каретки, чтобы обеспечить хорошую видимость и контрастность с окружающим контентом, на основе <code>currentColor</code> , фона, теней и т.д. Значение по умолчанию.
<code>цвет</code>	Каретка вставки окрашивается указанным цветом.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

```

    caret-color: auto;
    caret-color: transparent;
    caret-color: currentColor;
    caret-color: red;
    caret-color: #5729e9;
    caret-color: rgb(0, 200, 0);
    caret-color: hsla(228, 4%, 24%, 0.8);

```

CSS-градиент

CSS-градиент представляет собой переходы от одного цвета к другому.

Градиенты создаются с помощью функций `linear-gradient()` и `radial-gradient()`.

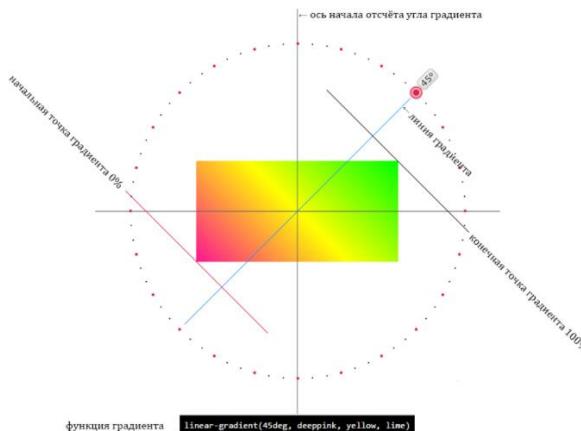
Градиентный фон можно устанавливать в свойствах `background`, `background-image`, `border-image` и `list-style-image`.

Линейный градиент linear-gradient()

Линейный градиент создается с помощью двух и более цветов, для которых задано направление, или **линия градиента**.

Если направление не указано, используется значение по умолчанию – **сверху-вниз**.

Цвета градиента по умолчанию распределяются равномерно в направлении, перпендикулярном линии градиента.

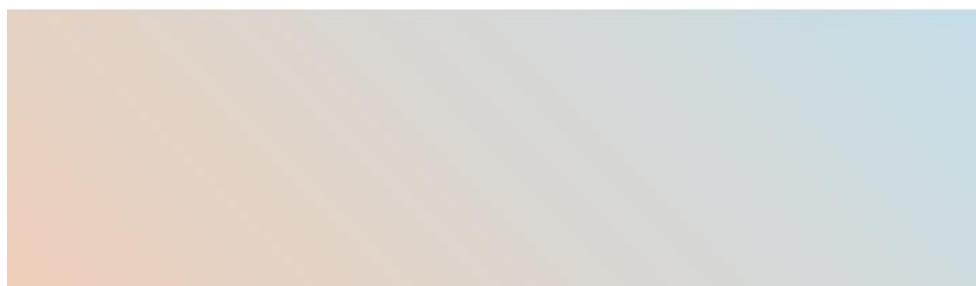


```
background: linear-gradient(45deg, deeppink, lime);
```

Направление градиента может быть задано двумя способами:

- **с помощью угла наклона** в градусах `deg`, значение которого определяет угол наклона линии внутри элемента.

```
div {  
height: 200px;  
background: linear-gradient(45deg, #EECFBA, #C5DDE8);  
}
```



- **с помощью ключевых слов** `to top`, `to right`, `to bottom`, `to left`, которые соответствуют углу градиента, равному 0deg, 90deg, 180deg и 270deg соответственно.

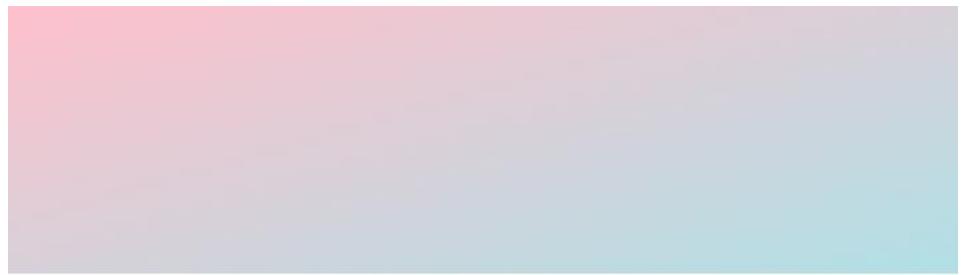
Если направление задано парой ключевых слов, например, `to top left`, то

```
div {  
height: 200px;  
background: linear-gradient(to right, #F6EFD2, #CEAD78);  
}
```



начальная точка градиента будет расположена в противоположном направлении, в данном случае справа внизу.

```
div {  
height: 200px;  
background: linear-gradient(to top left, powderblue, pink);  
}
```



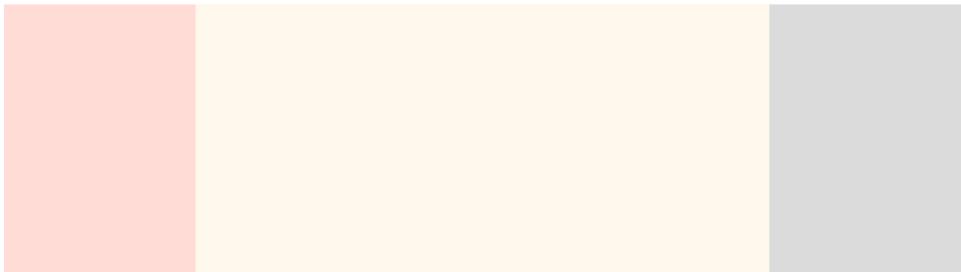
Для неравномерного распределения цветов указывается начальная позиция каждого цвета через точки остановки градиента, так называемые **color stops**. Точки остановки задаются в %, где 0% – начальная точка, 100% – конечная точка, например:

```
div {  
height: 200px;  
background: linear-gradient(to top, #E4AF9D 20%, #E4E4D8 50%, #A19887 80%);  
}
```



Для четкого распределения цветных полос каждый последующий цвет нужно начинать с точки остановки предыдущего цвета:

```
div {  
height: 200px;  
background: linear-gradient(to right, #FFDD66 20%, #FFF9ED 20%, #FFF9ED 80%, #DBDBDB 80%);  
}
```



Радиальный градиент radial-gradient()

Радиальный градиент отличается от линейного тем, что цвета выходят из одной точки (центра градиента) и равномерно распределяются наружу, рисуя форму круга или эллипса.

```
background: radial-gradient(форма градиента / размер / позиция центра, первый цвет, второй цвет и так далее);
```

Форма градиента определяется ключевыми словами `circle` или `ellipse`. Если форма не задана, по умолчанию радиальный градиент принимает форму эллипса.

```
div {  
height: 200px;  
background: radial-gradient(white, #FFA9A1);  
}
```



Позиция центра задается с помощью ключевых слов, используемых в свойстве `background-position`, с добавлением приставки `at`. Если позиция центра не задана, используется значение по умолчанию `at center`.

```
div {  
height: 200px;  
background: radial-gradient(at top, #FEFFFF, #A7CECC);  
}
```



С помощью пары значений, указанных в единицах длины `%`, `em` или `px`, можно управлять размером эллипсообразного градиента. Первое значение задает ширину эллипса, второе – высоту.

```
div {  
height: 200px;  
background: radial-gradient(40% 50%, #FAECD5, #CAE4D8);  
}
```



Размер градиента задается с помощью ключевых слов. Значение по умолчанию `farthest-corner` (к дальнему углу).

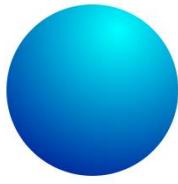
Значение	Описание
<code>closest-side</code>	Размер градиента рассчитывается из расстояния до любой ближней стороны блока для <code>circle</code> или до ближних сторон по <code>x</code> и по <code>y</code> для <code>ellipse</code> .
<code>farthest-side</code>	Размер рассчитывается из расстояния до дальних сторон.
<code>closest-corner</code>	Размер рассчитывается из расстояния до ближних углов.
<code>farthest-corner</code>	Размер рассчитывается из расстояния до дальних углов.

```
div {  
height: 200px;  
background: radial-gradient(circle farthest-corner at 100px 50px, #FBF2EB, #352A3B);  
}
```



С помощью радиального градиента можно создавать реалистичные объемные фигуры, такие как мячи, кнопки.

Мяч



```
div {  
width: 200px;  
height: 200px;  
border-radius: 50%;  
margin: 0 auto;  
background: radial-gradient(circle at 65% 15%, aqua, darkblue);  
}
```

Кнопка

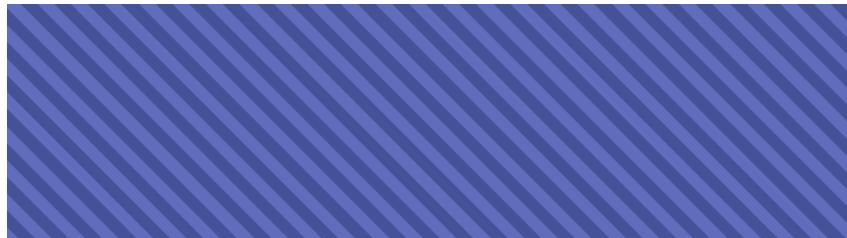


```
.wrap {  
height: 200px;  
padding: 50px 0;  
background: #cccccc;  
}  
.button {  
width: 100px;  
height: 100px;  
border-radius: 50%;  
margin: 0 auto;  
background: radial-gradient(farthest-side ellipse at top left, white, #aaaaaaaa);  
box-shadow: 5px 10px 20px rgba(0,0,0,0.3), -5px -10px 20px rgba(255,255,255,0.5);  
}
```

Повтор градиента

В добавление к линейному и радиальному градиентам существует повтор градиента, который задается с помощью функций `repeating-linear-gradient()` и `repeating-radial-gradient()` соответственно. Фон из повторяющихся градиентов выглядит неаккуратно, поэтому рекомендуется начинать следующий цвет с точки остановки предыдущего, создавая таким образом полосатые узоры.

```
div {  
height: 200px;  
background: repeating-linear-gradient(45deg, #606dbc, #606dbc 10px, #465298 10px, #465298 20px);  
}
```



```
div {  
height: 200px;  
background: repeating-radial-gradient(circle, #B9ECFF, #B9ECFF 10px, #82DDFF 10px, #82DDFF 20px);  
}
```



Кросбраузерный градиент

Для корректного отображения градиентов во всех браузерах необходимо добавить кросбраузерную запись. Для создания можно воспользоваться [онлайн градиент генератором](#).

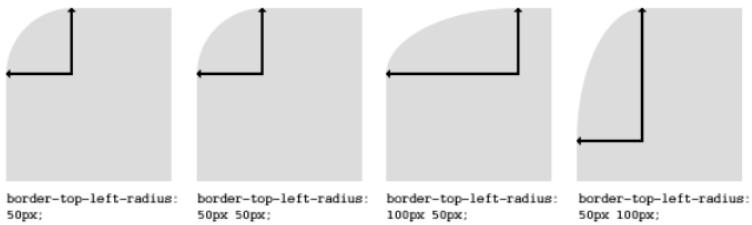
CSS3-рамка

CSS3-рамка дополняет возможности форматирования границ элементов с помощью свойств, позволяющих **закруглить углы** элемента, а также использовать **изображения** для оформления границ элемента.

Закругление углов с помощью border-radius

Свойство позволяет закруглить углы строчных и блочных элементов. Кривая для каждого угла определяется с помощью одного или двух радиусов,

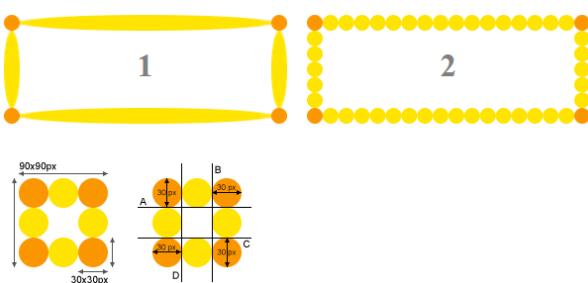
определяющих его форму – **круга** или **эллипса**. Радиус распространяется на весь фон, даже если элемент не имеет границ, точное положение секущей определяется с помощью свойства `background-clip`.



[Описание свойства border-radius.](#)

Рамки-изображения border-image

Свойство позволяет устанавливать изображение в качестве рамки элемента. Основное требование, предъявляемое к изображению – оно должно быть симметричным.



[Описание свойства border-image.](#)

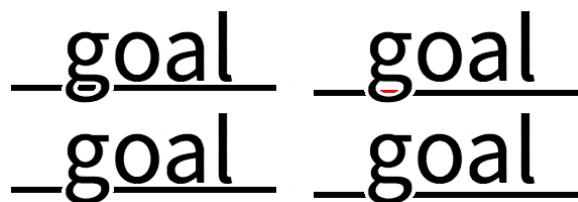
CSS3-оформление текста

У CSS3 имеются возможности, относящиеся к оформлению текста, такие как подчеркивание, тени текста, а также акценты в тексте восточно-азиатских языков.

Оформление линии: подчеркивание, обводка и зачеркивание

Подчеркивание, обводка и перечеркивающие линии отображаются только для незамещаемых блоков уровня строки (`display: inline`) и отображаются по всему тексту, включая пробелы между символами и словами, за исключением отступов в начале и конце строки.

Браузеры могут прерывать подчеркивание и обводку в том месте, где линия пересекает глиф, отображаясь на некотором расстоянии по обе стороны от контура глифа. Когда браузер прерывает подчеркивание или обводку на границах глифа, форма линии на этой границе должна соответствовать форме глифа. Однако, спецификация не предписывает конкретный метод для «следования форме» глифа, оставляя это на усмотрение браузеру.



Вид линии оформления: свойство text-decoration-line

Свойство `text-decoration-line` определяет, какой тип линии, если таковые имеются, добавляется к элементу. Свойство не наследуется.

[Описание свойства text-decoration-line.](#)

Стиль линии оформления: свойство `text-decoration-style`

Свойство `text-decoration-style` определяет стиль линий, нарисованных для оформления текста, указанного в элементе. Значения имеют то же значение, что и для свойства `border-style`. Свойство не наследуется.

[Описание свойства `text-decoration-style`.](#)

Цвет линии оформления: свойство `text-decoration-color`

Свойство `text-decoration-color` определяет цвет линии оформления текста, установленный для элемента с `text-decoration-line`. Свойство не наследуется.

[Описание свойства `text-decoration-color`.](#)

Краткая запись свойств линии оформления: свойство `text-decoration`

Свойство `text-decoration` является краткой формой записи свойств `text-decoration-line`, `text-decoration-style`, `text-decoration-color` в одном объявлении. Пропущенные значения устанавливаются на их начальные значения. Значение по умолчанию `text-decoration: none solid currentColor;`. Свойство не наследуется. Тем не менее, стиль всех линий оформления текста должен быть одинаковый для одного элемента.

Расположение линии оформления: свойство `text-underline-position`

Свойство `text-underline-position` устанавливает положение линии подчеркивания, указанного в элементе. Свойство наследуется.

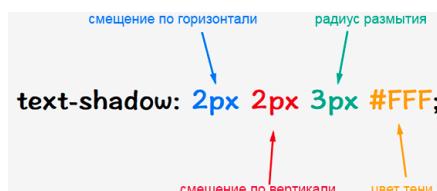
[Описание свойства `text-underline-position`.](#)

Тень текста: свойство `text-shadow`

Свойство `text-shadow` используется для добавления тени к тексту. Тень текста – интересный инструмент, который позволяет создавать удивительные эффекты. Тени могут быть однослойными или многослойными, размытыми, цветными или полупрозрачными. Задавая тень для элемента, можно указывать только одно значение длины и цвет, таким образом создавая цветную копию отдельного символа или слова. Также, с помощью тени можно сделать текст более читаемым, если контраст между цветом текста и фоном невелик.

Каждая тень применяется как к самому тексту, так и к элементам его оформления (свойство `text-decoration`). Одновременно можно задавать несколько теней, указывая их через запятую. Тени накладываются друг на друга, но не перекрывают сам текст. Первая тень всегда расположена сверху над остальными тенями. Свойство наследуется.

Каждая тень определяется двумя или тремя значениями длины и необязательным цветом. Допустимы длины, равные 0. Свойство не наследуется.



[Описание свойства `text-shadow`.](#)

Тень текста

```
.text-shadow {  
    color: white;  
    text-shadow: 1px 1px #732372, 1px -1px #732372, -1px 1px #732372, -1px -1px #732372, 3px 3px 6px rgba(0,0,0,.5);  
}
```

CSS3-тень блока

Свойство `box-shadow` добавляет элементу одну или более теней. Тень представляет собой копию элемента, смещенную на указанное расстояние. Тени бывают внешние или внутренние, размытые или плоские, они могут следовать контурам блоков со скругленными углами. С помощью ключевого слова `inset` создаются тени внутри элемента, делая элемент визуально объемным или вдавленным.

Синтаксис свойства `box-shadow`

Свойство `box-shadow` прикрепляет одну или несколько теней к блоку. Свойство принимает либо значение `none`, которое указывает на отсутствие теней, либо список теней через запятую, упорядоченный от начала к концу.

Каждая тень является отдельной тенью, представленной от 2 до 4-х значений длины, необязательным цветом и необязательным ключевым словом `inset`. Допустимые длины `0`; опущенные цвета по умолчанию равны значению свойства `color`. Свойство не наследуется.



Описание свойства `box-shadow`.



```
<p class="example-shadow"><span></span></p>
```

```
.example-shadow {  
background: #e8e8e8;  
text-align: center;  
}  
.example-shadow span {  
background: white;  
display: inline-block;  
width: 200px;  
height: 100px;  
margin: 50px;  
box-shadow: 0 14px 28px rgba(0,0,0,0.25), 0 10px 10px rgba(0,0,0,0.22);  
}
```

[Смотреть пример эффекта для тени при наведении](#)

[Смотреть пример анимации тени](#)

CSS3-переходы

CSS3-переходы позволяют анимировать исходное значение CSS-свойства на новое значение с течением времени, управляя скоростью смены значений свойств. Большинство свойств меняют свои значения за 16 миллисекунд, поэтому рекомендуемое время стандартного перехода – `200ms`.

Смена свойств происходит при наступлении определенного события, которое описывается соответствующим псевдоклассом. Чаще всего используется псевдокласс `:hover`. Данный псевдокласс не работает на мобильных устройствах, таких как iPhone или Android. Универсальным решением, работающим в настольных и мобильных браузерах, будет обработка событий с помощью JavaScript (например, переключение классов при клике).

Переходы применяются ко всем элементам, а также к псевдоэлементам `::before` и `::after`. Для задания всех свойств перехода обычно используют краткую запись свойства `transition`.

CSS3-переходы могут применяться не ко всем свойствам и их значениям.

Свойство `transition-property`

Содержит название CSS-свойств, к которым будет применен эффект перехода. Значение свойства может содержать как одно свойство, так и список

свойств через запятую. При создании перехода можно использовать как начальное, так и конечное состояние элемента. Свойство не наследуется.

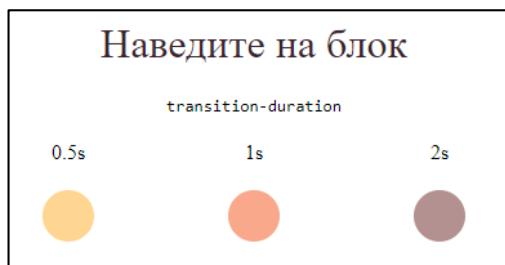
Создаваемые эффекты должны быть ненавязчивыми. Не все свойства требуют плавного изменения во времени, что связано с пользовательским опытом. Например, при наведении на ссылку мы хотим видеть мгновенную смену цвета ссылки или цвета и стиля подчеркивания. Поэтому переходы следует использовать для тех свойств, к которым действительно нужно привлечь внимание.

[Описание свойства transition-property.](#)

Продолжительность перехода transition-duration

Задает промежуток времени, в течение которого должен осуществляться переход. Если разные свойства имеют разные значения для перехода, они указываются через запятую. Если продолжительность перехода не указана, то анимация при смене значений свойств происходит не будет. Свойство не наследуется.

[Описание свойства transition-duration.](#)

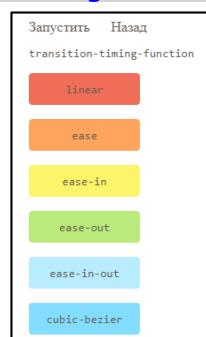


[Смотреть пример продолжительности выполнения анимации](#)

Функция перехода transition-timing-function

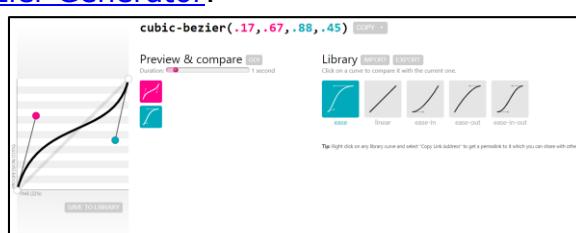
Свойство задает временную функцию, которая описывает скорость перехода объекта от одного значения к другому. Если вы определяете более одного перехода для элемента, например, цвет фона элемента и его положение, вы можете использовать разные функции для каждого свойства. Свойство не наследуется.

[Описание свойства transition-timing-function.](#)



[Смотреть пример функции перехода](#)

Для создания более реалистичной анимации используйте функцию `cubic-bezier()`. [CSS Cubic Bezier Generator](#).



Задержка перехода transition-delay

Необязательное свойство, позволяет сделать так, чтобы изменение свойства происходило не моментально, а с некоторой задержкой. Не наследуется.

Значение	transition-delay
время	Время задержки перехода указывается в секундах или миллисекундах.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
div {  
  transition-delay: .5s;  
}
```

[Смотреть пример задержки перехода](#)

Краткая запись перехода

Все свойства, отвечающие за изменение внешнего вида элемента `transition-property`, `transition-duration`, `transition-timing-function`, `transition-delay`, можно объединить в одно свойство `transition`.

Если воспользоваться значениями по умолчанию, то запись

```
div {transition: 1s;}
```

будет эквивалентна

```
div {transition: all 1s ease 0s;}
```

Плавный переход нескольких свойств

Для элемента можно задать несколько последовательных переходов, перечислив их через запятую. Каждый переход можно оформить своей временной функцией.

```
div {transition: background 0.3s ease, color 0.2s linear;}
```

или

```
div {  
  transition-property: height, width, background-color;  
  transition-duration: 3s;  
  transition-timing-function: ease-in, ease, linear;  
}
```

CSS3-трансформации

Модель визуального форматирования CSS описывает систему координат внутри каждого позиционированного элемента. Система координат является точкой отсчета для свойств смещения. Положение и размеры в этом координатном пространстве можно рассматривать как заданные в пикселях, относительно точки отсчета, с положительными значениями, идущими вправо и вниз. Это координатное пространство можно изменить с помощью свойства `transform`.

CSS3-трансформации позволяют сдвигать, поворачивать и масштабировать элементы. Трансформации преобразовывают элемент, не затрагивая остальные элементы веб-страницы, т.е. другие элементы не сдвигаются относительно него.

К элементам, которые могут быть трансформированы, относятся элементы с `display: block;` и `display: inline-block;`, а также элементы, значение свойства `display` которых вычисляется как `table-row`, `table-row-group`, `table-header-group`, `table-footer-group`, `table-cell` или `table-caption`. Трансформированным считается элемент с любым установленным значением свойства `transform`, отличным от `none`.

Существуют два вида CSS3-трансформаций – **2D** и **3D**. **2D-трансформации** преобразовывают элементы в двумерном пространстве с помощью 2D-матрицы преобразований. Эта матрица применяется для вычисления новых координат объекта, на основе значений свойств `transform` и `transform-origin`. Преобразования влияют только на визуальный рендеринг. В отношении макета страницы они могут отразиться на переполнении содержимого блока. По умолчанию точка трансформации находится в центре элемента.

Функции 2D-трансформации `transform`

Свойство задает вид преобразования элемента. Свойство описывается с помощью функций трансформации, которые смещают элемент относительно его текущего положения на странице или изменяют его первоначальные размеры и форму. Не наследуется.

Описание свойства `transform`.

Точка трансформации `transform-origin`

Свойство позволяет сместить центр трансформации, относительно которого происходит изменение положения/размера/формы элемента. Значение по умолчанию – `center`, или `50% 50%`. Задается только для трансформированных элементов. Не наследуется.

Описание свойства `transform-origin`.

Множественные трансформации

Можно объединить несколько трансформаций одного элемента, перечислив их через пробел в порядке проявления.

```
div {transform: scale(1.5) rotate(-10deg);}
```

CSS3-анимация

CSS3-анимация придает сайтам динамичность. Она оживляет веб-страницы, улучшая взаимодействие с пользователем. В отличие от CSS3-переходов, создание анимации базируется на ключевых кадрах, которые позволяют автоматически воспроизводить и повторять эффекты на протяжении заданного времени, а также останавливать анимацию внутри цикла.

CSS3-анимация может применяться практически для всех html-элементов, а также для псевдоэлементов `::before` и `::after`. Список анимируемых свойств приведен на [этой](#) странице. При создании анимации не стоит забывать о возможных проблемах с производительностью, так как на изменение некоторых свойств требуется много ресурсов.

Ключевые кадры

Ключевые кадры используются для указания значений свойств анимации в различных точках анимации. Ключевые кадры определяют поведение одного цикла анимации; анимация может повторяться ноль или более раз.

Ключевые кадры указываются с помощью правила `@keyframes`, определяемого следующим образом:

```
@keyframes имя анимации { список правил }
```

Создание анимации начинается с установки ключевых кадров правила `@keyframes`. Кадры определяют, какие свойства на каком шаге будут анимированы. Каждый кадр может включать один или более блоков объявления из одного или более пар свойств и значений. Правило `@keyframes` содержит имя анимации элемента, которое связывает правило и блок объявления элемента.

```
@keyframes shadow {  
from {text-shadow: 0 0 3px black;}  
50% {text-shadow: 0 0 30px black;}  
to {text-shadow: 0 0 3px black;}  
}
```

Ключевые кадры создаются с помощью ключевых слов `from` и `to` (эквивалентны значениям `0%` и `100%`) или с помощью процентных пунктов, которых можно задавать сколько угодно. Также можно комбинировать ключевые слова и процентные пункты. Если кадры имеют одинаковые свойства и значения, их можно объединить в одно объявление:

```
@keyframes move {  
from,  
to {  
top: 0;  
left: 0;  
}  
25%,  
75% {top: 100%;}  
50% {top: 50%;}  
}
```

Если `0%` или `100%` кадры не указаны, то браузер пользователя создает их, используя вычисляемые (первоначально заданные) значения анимируемого свойства.

Если несколько правил `@keyframes` определены с одним и тем же именем, сработает последнее в порядке документа, а все предыдущие проигнорируются.

После объявления правила `@keyframes`, можно ссылаться на него в свойстве `animation`:

```
h1 {  
font-size: 3.5em;  
color: darkmagenta;  
animation: shadow 2s infinite ease-in-out;  
}
```

Не рекомендуется анимировать нечисловые значения (за редким исключением), так как результат в браузере может быть непредсказуемым. Также не следует создавать ключевые кадры для значений свойств, не имеющих средней точки, например, для значений свойства `color: pink` и `color: #ffffff`, `width: auto` и `width: 100px` или `border-radius: 0` и `border-radius: 50%` (в этом случае правильно будет указать `border-radius: 0%`).

Временная функция для ключевых кадров

Правило стиля ключевого кадра также может объявлять временную функцию, которая должна использоваться при перемещении анимации к следующему ключевому кадру.

```
@keyframes bounce {  
from {  
top: 100px;  
animation-timing-function: ease-out;  
}  
25% {  
top: 50px;  
animation-timing-function: ease-in;  
}  
50% {  
top: 100px;  
animation-timing-function: ease-out;  
}  
75% {  
top: 75px;  
animation-timing-function: ease-in;  
}  
to {  
top: 100px;  
}
```

Пять ключевых кадров указаны для анимации с именем "bounce". Между первым и вторым ключевым кадром (то есть между `0%` и `25%`) используется

функция замедления. Между вторым и третьим (то есть между 25% и 50%) – функция плавного ускорения. И так далее. Элемент будет перемещаться вверх по странице на 50px, замедляясь по мере того, как он достигнет своей наивысшей точки, а затем ускоряясь, когда он упадет до 100px. Вторая половина анимации ведет себя аналогичным образом, но только перемещает элемент на 25px вверх по странице.

Временная функция, указанная в ключевом кадре to или 100%, игнорируется.

Название анимации: свойство animation-name

Свойство `animation-name` определяет список применяемых к элементу анимаций. Каждое имя используется для выбора ключевого кадра в правиле, которое предоставляет значения свойств для анимации. Если имя не соответствует ни одному ключевому кадру в правиле, нет свойств для анимации, отсутствует имя анимации, анимация не будет выполняться.

Если несколько анимаций пытаются изменить одно и то же свойство, то выполнится анимация, ближайшая к концу списка имен.

Имя анимации чувствительно к регистру, не допускается использование ключевого слова `none`. Рекомендуется использовать название, отражающее суть анимации, при этом можно использовать одно или несколько слов, перечисленных через дефис - или символ нижнего подчеркивания _.

Свойство не наследуется.

Значение	animation-name
<code>none</code>	Означает отсутствие анимации. Также используется, чтобы отменить анимацию элемента из группы элементов, для которых задана анимация. Значение по умолчанию.
имя анимации	Имя анимации, которое связывает правило <code>@keyframes</code> с селектором.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента. <pre>animation-name: none; animation-name: test-01; animation-name: -sliding; animation-name: moving-vertically; animation-name: test2; animation-name: test3, move4; animation-name: initial; animation-name: inherit;</pre>

Продолжительность анимации: свойство animation-duration

Свойство `animation-duration` определяет продолжительность одного цикла анимации. Задается в секундах s или миллисекундах ms. Если для элемента задано более одной анимации, то можно установить разное время для каждой, перечислив значения через запятую.

Свойство не наследуется.

Значение	animation-duration
время	Указывает время, которое анимация занимает для завершения одного цикла. Отрицательные значения недействительны. Если время равно 0s, ключевые кадры анимации не действуют, но сама анимация происходит мгновенно. Значение по умолчанию 0s.

```
animation-duration: .5s;
animation-duration: 200ms;
animation-duration: 2s, 10s;
animation-duration: 15s, 30s, 200ms;
```

Временная функция: свойство animation-timing-function

Свойство `animation-timing-function` описывает, как будет развиваться анимация между каждой парой ключевых кадров. Во время задержки анимации временные функции не применяются.

Свойство не наследуется.

Описание свойства animation-timing-function.

```
animation-timing-function: ease;
animation-timing-function: ease-in;
animation-timing-function: ease-out;
animation-timing-function: ease-in-out;
animation-timing-function: linear;
animation-timing-function: step-start;
animation-timing-function: step-end;
animation-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1);
animation-timing-function: steps(4, end);
animation-timing-function: ease, step-start, cubic-bezier(0.1, 0.7, 1.0, 0.1);
animation-timing-function: initial;
animation-timing-function: inherit;
```

Свойства animation-iteration-count, animation-direction, animation-play-state, animation-delay, animation-fill-mode

Свойство `animation-iteration-count` указывает, сколько раз проигрывается цикл анимации.

Свойство `animation-direction` определяет, должна ли анимация воспроизводиться в обратном порядке в некоторых или во всех циклах.

Свойство `animation-play-state` определяет, будет ли анимация запущена или приостановлена.

Свойство `animation-delay` определяет, когда анимация начнется.

Свойство `animation-fill-mode` определяет, какие значения применяются анимацией вне времени ее выполнения. Когда анимация завершается, элемент возвращается к своим исходным стилям.

Краткая запись анимации: свойство animation

Все параметры воспроизведения анимации можно объединить в одном свойстве – `animation`, перечислив их через пробел: `animation: animation-name animation-duration animation-timing-function animation-delay animation-iteration-count animation-direction;`.

Для воспроизведения анимации достаточно указать только два свойства – `animation-name` и `animation-duration`, остальные свойства примут значения по умолчанию. Порядок перечисления свойств не имеет значения, единственное, время выполнения анимации `animation-duration` обязательно должно стоять перед задержкой `animation-delay`.

Множественные анимации

Для одного элемента можно задавать несколько анимаций, перечислив их названия через запятую:

```
div {animation: shadow 1s ease-in-out 0.5s alternate, move 5s linear 2s;}
```

CSS flexbox

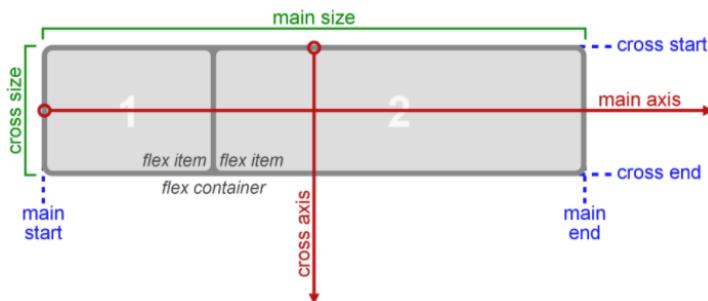
CSS flexbox (Flexible Box Layout Module) – модуль макета гибкого контейнера – представляет собой способ компоновки элементов, в основе лежит идея оси.

Flexbox состоит из **контейнера (flex container)** и **элементов (flex items)**. Элементы могут выстраиваться в строку или столбик, а оставшееся свободное пространство распределяется между ними различными способами.

Модуль flexbox позволяет решать следующие задачи:

- Располагать элементы в одном из четырех направлений: слева направо, справа налево, сверху вниз или снизу вверх.
- Переопределять порядок отображения элементов.
- Автоматически определять размеры элементов таким образом, чтобы они вписывались в доступное пространство.
- Решать проблему с горизонтальным и вертикальным центрированием.
- Переносить элементы внутри контейнера, не допуская его переполнения.
- Создавать колонки одинаковой высоты.
- Создавать прижатый к низу страницы подвал сайта.

Flexbox решает специфические задачи – создание одномерных макетов, например, навигационной панели, так как flex-элементы можно размещать только по одной из осей.



Для описания модуля Flexbox используется определенный набор терминов. Значение **flex-flow** и режим записи определяют соответствие этих терминов физическим направлениям: верх / право / низ / лево, осям: вертикальная / горизонтальная и размерам: ширина / высота.

Главная ось (main axis) – ось, вдоль которой выкладываются flex-элементы. Она простирается в основном измерении.

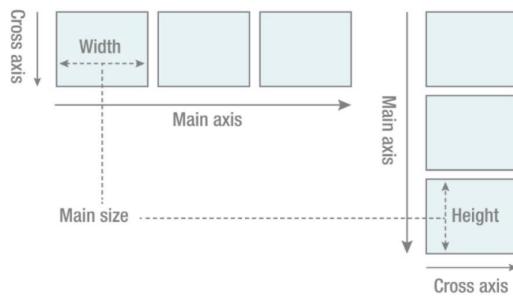
Main start и **main end** – линии, которые определяют начальную и конечную стороны flex-контейнера, относительно которых выкладываются flex-элементы (начиная с main start по направлению к main end).

Основной размер (main size) – ширина или высота flex-контейнера или flex-элементов, в зависимости от того, что из них находится в основном измерении, определяют основной размер flex-контейнера или flex-элемента.

Поперечная ось (cross axis) – ось, перпендикулярная главной оси. Она простирается в поперечном измерении.

Cross start и **cross end** – линии, которые определяют начальную и конечную стороны поперечной оси, относительно которых выкладываются flex-элементы.

Поперечный размер (cross size) – ширина или высота flex-контейнера или flex-элементов, в зависимости от того, что находится в поперечном измерении, являются их поперечным размером.



Flex-контейнер

Flex-контейнер устанавливает новый гибкий контекст форматирования для его содержимого. Flex-контейнер не является блочным контейнером, поэтому для дочерних элементов не работают такие CSS-свойства, как `float`, `clear`, `vertical-align`. Также, на flex-контейнер не оказывают влияние свойства `column-*`, создающие колонки в тексте и псевдоэлементы `::first-line` и `::first-letter`.

Модель flexbox-разметки связана с определенным значением CSS-свойства `display` родительского `html`-элемента, содержащего внутри себя дочерние блоки. Для управления элементами с помощью этой модели нужно установить свойство `display` следующим образом:

```
.flex-container {
    /*генерирует flex-контейнер уровня блока*/
    display: -webkit-flex;
    display: flex;
}
.flex-container {
    /*генерирует flex-контейнер уровня строки*/
    display: -webkit-inline-flex;
    display: inline-flex;
}
```

После установки данных значений свойства каждый дочерний элемент автоматически становится flex-элементом, выстраиваясь в один ряд (вдоль главной оси). При этом блочные и строчные дочерние элементы ведут себя одинаково, т.е. ширина блоков равна ширине их содержимого с учетом внутренних полей и рамок элемента.



Если родительский блок содержит текст или изображения без оберток, они становятся анонимными flex-элементами. Текст выравнивается по верхнему краю блока-контейнера, а высота изображения становится равной высоте блока, т.е. оно деформируется.

Flex-элементы

Flex-элементы – блоки, представляющие содержимое flex-контейнера в потоке. Flex-контейнер устанавливает новый контекст форматирования для своего содержимого, который обуславливает следующие особенности:

- Для flex-элементов блокируется их значение свойства `display`. Значение `display: inline-block;` и `display: table-cell;` вычисляется в `display: block;`.
- Пустое пространство между элементами исчезает: оно не становится своим собственным flex-элементом, даже если межэлементный текст обернут в анонимный flex-элемент. Для содержимого анонимного flex-элемента невозможно задать собственные стили, но оно будет наследовать их (например, параметры шрифта) от flex-контейнера.

- Абсолютно позиционированный flex-элемент не участвует в компоновке гибкого макета.
- Поля `margin` соседних flex-элементов не схлопываются.
- Процентные значения `margin` и `padding` вычисляются от внутреннего размера содержащего их блока.
- `margin: auto;` расширяются, поглощая дополнительное пространство в соответствующем измерении. Их можно использовать для выравнивания или раздвигания смежных flex-элементов.
- Автоматический минимальный размер flex-элементов по умолчанию является минимальным размером его содержимого, то есть `min-width: auto;`. Для контейнеров с прокруткой автоматический минимальный размер обычно равен нулю.

Порядок отображения flex-элементов и ориентация

Содержимое flex-контейнера можно разложить в любом направлении и в любом порядке (переупорядочение flex-элементов внутри контейнера влияет только на визуальный рендеринг).

Направление главной оси: `flex-direction`

Свойство относится к flex-контейнеру. Управляет направлением главной оси, вдоль которой укладываются flex-элементы, в соответствии с текущим режимом записи. Не наследуется.

Значение	<code>flex-direction</code>
<code>row</code>	Значение по умолчанию, слева направо (в rtl справа налево). Flex-элементы выкладываются в строку. Начало (<code>main-start</code>) и конец (<code>main-end</code>) направления главной оси соответствуют началу (<code>inline-start</code>) и концу (<code>inline-end</code>) оси строки (<code>inline-axis</code>).
<code>row-reverse</code>	Направление справа налево (в rtl слева направо). Flex-элементы выкладываются в строку относительно правого края контейнера (в rtl – левого).
<code>column</code>	Направление сверху вниз. Flex-элементы выкладываются в колонку.
<code>column-reverse</code>	Колонка с элементами в обратном порядке, снизу вверх.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

`flex-direction: row-reverse;`



`flex-direction: column;`



`flex-direction: column-reverse;`



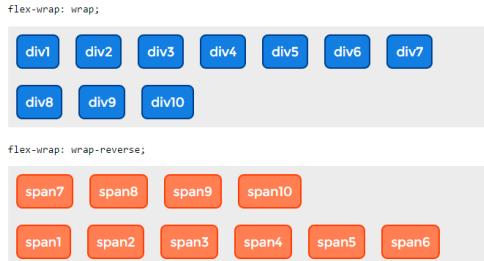
```
.flex-container {
  display: -webkit-flex;
  -webkit-flex-direction: row-reverse;
  display: flex;
  flex-direction: row-reverse;
}
```

Управление многострочностью flex-контейнера: flex-wrap

Свойство определяет, будет ли flex-контейнер односторонним или многострочным, а также задает направление поперечной оси, определяющее направление укладки новых линий flex-контейнера.

По умолчанию flex-элементы укладываются в одну строку, вдоль главной оси. При переполнении они будут выходить за пределы ограничивающей рамки flex-контейнера. Свойство не наследуется.

Значение	flex-wrap
nowrap	Значение по умолчанию. Flex-элементы не переносятся, а располагаются в одну линию слева направо (в rtl справа налево).
wrap	Flex-элементы переносятся, располагаясь в несколько горизонтальных рядов (если не помещаются в один ряд) в направлении слева направо (в rtl справа налево).
wrap-reverse	Flex-элементы переносятся на новые линии, располагаясь в обратном порядке слева-направо, при этом перенос происходит снизу вверх.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.



flex-wrap: wrap;



```
.flex-container {
  display: -webkit-flex;
  -webkit-flex-wrap: wrap;
  display: flex;
  flex-wrap: wrap;
}
```

Краткая запись направления и многострочности: flex-flow

Свойство позволяет определить направления главной и поперечной осей, а также возможность переноса flex-элементов при необходимости на несколько строк. Представляет собой сокращенную запись свойств `flex-direction` и `flex-wrap`. Значение по умолчанию `flex-flow: row nowrap;`. Свойство не наследуется.

Значение	flex-flow
направление	Указывает направление главной оси. Значение по умолчанию <code>row</code> .
многострочность	Задает многострочность поперечной оси. Значение по умолчанию <code>nowrap</code> .
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
.flex-container {
  display: -webkit-flex;
  -webkit-flex-flow: row wrap;
  display: flex;
  flex-flow: row wrap;
}
```

Порядок отображения flex-элементов: order

Свойство определяет порядок, в котором flex-элементы отображаются и располагаются внутри flex-контейнера. Применяется к flex-элементам. Свойство не наследуется.

Первоначально все flex-элементы имеют `order: 0;`. При указании значения от `-1` для элемента он перемещается в начало строки, значение `1` – в конец. Если несколько flex-элементов имеют одинаковое значение `order`, они будут отображаться в соответствии с исходным порядком.

Значение	order
число	Свойство задается целым числом, отвечающим за порядок отображения flex-элементов. Значение по умолчанию <code>0</code> .
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

```
.flex-container {
  display: -webkit-flex;
  display: flex;
}
.flex-item {
  -webkit-order: 1;
  order: 1;
}
```

Гибкость flex-элементов

Определяющим аспектом гибкого макета является возможность «сгибать» flex-элементы, изменяя их ширину / высоту (в зависимости от того, какой размер находится на главной оси), чтобы заполнить доступное пространство в основном измерении. Это делается с помощью свойства `flex`. Flex-контейнер распределяет свободное пространство между своими дочерними элементами (пропорционально их коэффициенту `flex-grow`) для заполнения контейнера или сжимает их (пропорционально их коэффициенту `flex-shrink`), чтобы предотвратить переполнение.

Flex-элемент будет полностью «негибок», если его значения `flex-grow` и `flex-shrink` равны нулю, и «гибкий» в противном случае.

Задание гибких размеров одним свойством: `flex`

Свойство является сокращенной записью свойств `flex-grow`, `flex-shrink` и `flex-basis`. Значение по умолчанию: `flex: 0 1 auto;`. Можно указывать как одно, так и все три значения свойств. Свойство не наследуется.

W3C рекомендует использовать сокращенную запись, так как она правильно сбрасывает любые неуказанные компоненты, чтобы подстроиться под типичное использование.

Значение	flex
коэффициент растяжения	Коэффициент увеличения ширины flex-элемента относительно других flex-элементов.

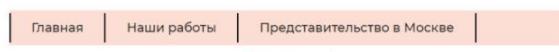
коэффициент сужения	Коэффициент уменьшения ширины flex-элемента относительно других flex-элементов.
базовая ширина	Базовая ширина flex-элемента.
auto	Эквивалентно <code>flex: 1 1 auto;</code> .
none	Эквивалентно <code>flex: 0 0 auto;</code> .
initial	Устанавливает значение свойства в значение по умолчанию. Эквивалентно <code>flex: 0 1 auto;</code> .
inherit	Наследует значение свойства от родительского элемента.

```
.flex-container {
  display: -webkit-flex;
  display: flex;
}
.flex-item {
  -webkit-flex: 3 1 100px;
  -ms-flex: 3 1 100px;
  flex: 3 1 100px;
}
```

Коэффициент роста: `flex-grow`

Свойство определяет коэффициент роста одного flex-элемента относительно других flex-элементов в flex-контейнере при распределении положительного свободного пространства. Если сумма значений `flex-grow` flex-элементов в строке меньше 1, они занимают менее 100% свободного пространства. Свойство не наследуется.

Значение	<code>flex-grow</code>
число	Положительное целое или дробное число, устанавливающее коэффициент роста flex-элемента. Значение по умолчанию 0.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.



li {flex-grow: 0;}



li {flex-grow: 1;}



li:last-child {flex-grow: 1;}

```
.flex-container {
  display: -webkit-flex;
  display: flex;
}
.flex-item {
  -webkit-flex-grow: 3;
  flex-grow: 3;
}
```

Коэффициент сжатия: `flex-shrink`

Свойство указывает коэффициент сжатия flex-элемента относительно других flex-элементов при распределении отрицательного свободного пространства. Умножается на базовый размер `flex-basis`. Отрицательное пространство распределяется пропорционально тому, насколько элемент может сжаться, поэтому, например, маленький flex-элемент не уменьшится до нуля, пока не будет заметно уменьшен flex-элемент большего размера. Свойство не наследуется.

Значение	flex-shrink
число	Положительное целое или дробное число, устанавливающее коэффициент уменьшения flex-элемента. Значение по умолчанию 1.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
.flex-container {
  display: -webkit-flex;
  display: flex;
}
.flex-item {
  -webkit-flex-shrink: 3;
  flex-shrink: 3;
}
```

Базовый размер: flex-basis

Свойство устанавливает начальный основной размер flex-элемента до распределения свободного пространства в соответствии с коэффициентами гибкости. Для всех значений, кроме `auto` и `content`, базовый гибкий размер определяется так же, как `width` в горизонтальных режимах записи. Процентные значения определяются относительно размера flex-контейнера, а если размер не задан, используемым значением для `flex-basis` являются размеры его содержимого. Не наследуется.

Значение	flex-basis
auto	Значение по умолчанию. Элемент получает базовый размер, соответствующий размеру его содержимого (если он не задан явно).
content	Определяет базовый размер в зависимости от содержимого flex-элемента.
длина	Базовый размер определяется так же, как для ширины и высоты. Задается в единицах длины.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
.flex-container {
  display: -webkit-flex;
  display: flex;
}
.flex-item {
  -webkit-flex-basis: 100px;
  flex-basis: 100px;
}
```

Выравнивание

Выравнивание по главной оси: justify-content

Свойство выравнивает flex-элементы по главной оси flex-контейнера, распределяя свободное пространство, незанятое flex-элементами. Когда элемент преобразуется в flex-контейнер, flex-элементы по умолчанию сгруппированы вместе (если для них не заданы поля `margin`). Промежутки добавляются после

расчета значений `margin` и `flex-grow`. Если какие-либо элементы имеют ненулевое значение `flex-grow` или `margin: auto;`, свойство не будет оказывать влияния. Свойство не наследуется.

Свойство `animation-iteration-count` указывает, сколько раз проигрывается цикл анимации. Начальное значение `1` означает, что анимация будет воспроизходить от начала до конца один раз. Это свойство часто используется в сочетании со значением `alternate` свойства `animation-direction`, которое заставляет анимацию воспроизводиться в обратном порядке в альтернативных циклах.

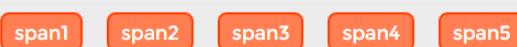
Свойство не наследуется.

Значение	<code>justify-content</code>
<code>flex-start</code>	Значение по умолчанию. Flex-элементы выкладываются в направлении, идущем от начальной линии flex-контейнера.
<code>flex-end</code>	Flex-элементы выкладываются в направлении, идущем от конечной линии flex-контейнера.
<code>center</code>	Flex-элементы выравниваются по центру flex-контейнера.
<code>space-between</code>	Flex-элементы равномерно распределяются по линии. Первый flex-элемент помещается вровень с краем начальной линии, последний flex-элемент – вровень с краем конечной линии, а остальные flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя соседними элементами было одинаковым. Если оставшееся свободное пространство отрицательно или в строке присутствует только один flex-элемент, это значение идентично параметру <code>flex-start</code> .
<code>space-around</code>	Flex-элементы на линии распределяются так, чтобы расстояние между любыми двумя смежными flex-элементами было одинаковым, а расстояние между первым / последним flex-элементами и краями flex-контейнера составляло половину от расстояния между flex-элементами.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

`justify-content: flex-start;`



`justify-content: flex-end;`



`justify-content: center;`



`justify-content: space-between;`



`justify-content: space-around;`



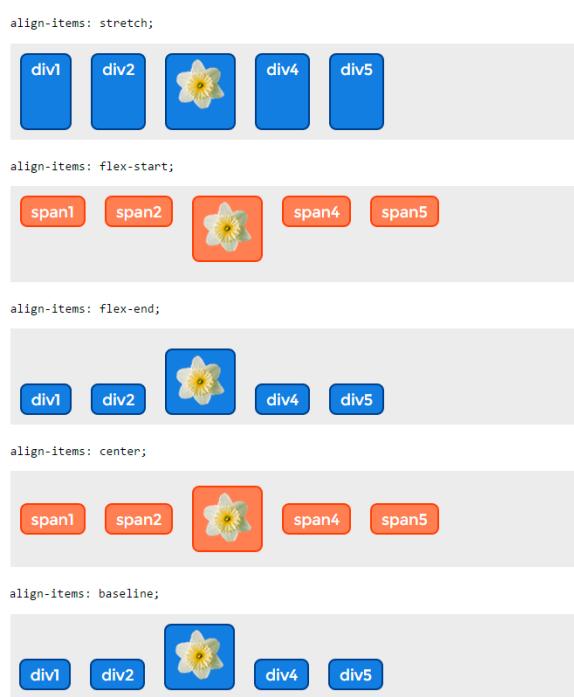
```
.flex-container {
  display: -webkit-flex;
  -webkit-justify-content: flex-start;
  display: flex;
  justify-content: flex-start;
}
```

Выравнивание по поперечной оси: align-items и align-self

Flex-элементы можно выравнивать по поперечной оси текущей строки flex-контейнера. `align-items` устанавливает выравнивание для всех элементов flex-контейнера, включая анонимные flex-элементы. `align-self` позволяет переопределить это выравнивание для отдельных flex-элементов. Если любое из поперечных `margin` flex-элемента имеет значение `auto`, `align-self` не имеет никакого влияния.

Свойство `align-items` выравнивает flex-элементы, в том числе и анонимные flex-элементы по поперечной оси. Не наследуется.

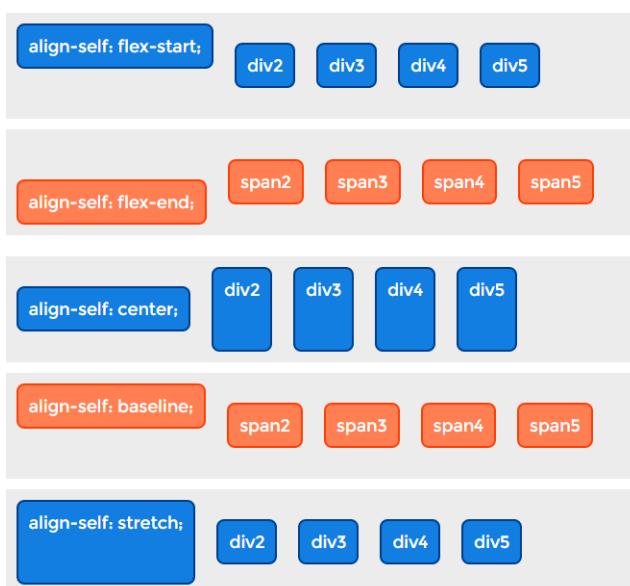
Значение	<code>align-items</code>
<code>flex-start</code>	Верхний край flex-элемента помещается вплотную с flex-линией (или на расстоянии, с учетом заданных полей <code>margin</code> и рамок <code>border</code> элемента), проходящей через начало поперечной оси.
<code>flex-end</code>	Нижний край flex-элемента помещается вплотную с flex-линией (или на расстоянии, с учетом заданных полей <code>margin</code> и рамок <code>border</code> элемента), проходящей через конец поперечной оси.
<code>center</code>	Поля flex-элемента центрируются по поперечной оси в пределах flex-линии.
<code>baseline</code>	Базовые линии всех flex-элементов, участвующих в выравнивании, совпадают.
<code>stretch</code>	Если поперечный размер flex-элемента вычисляется как <code>auto</code> и ни одно из поперечных значений <code>margin</code> не равно <code>auto</code> , элемент растягивается. Значение по умолчанию.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.



```
.flex-container {
  display: -webkit-flex;
  -webkit-align-items: flex-start;
  display: flex;
  align-items: flex-start;
}
```

Свойство `align-self` отвечает за выравнивание отдельно взятого flex-элемента по высоте flex-контейнера. Переопределяет выравнивание, заданное `align-items`. Не наследуется.

Значение	<code>align-items</code>
<code>auto</code>	Значение по умолчанию. Flex-элемент использует выравнивание, указанное в свойстве <code>align-items</code> flex-контейнера.
<code>flex-start</code>	Верхний край flex-элемента помещается вплотную с flex-линией (или на расстоянии, с учетом заданных полей <code>margin</code> и рамок <code>border</code> элемента), проходящей через начало поперечной оси.
<code>flex-end</code>	Нижний край flex-элемента помещается вплотную с flex-линией (или на расстоянии, с учетом заданных полей <code>margin</code> и рамок <code>border</code> элемента), проходящей через конец поперечной оси.
<code>center</code>	Flex-элемент выравнивается по базовой линии.
<code>baseline</code>	Базовые линии всех flex-элементов, участвующих в выравнивании, совпадают.
<code>stretch</code>	Если поперечный размер flex-элемента вычисляется как <code>auto</code> и ни одно из поперечных значений <code>margin</code> не равно <code>auto</code> , элемент растягивается. Значение по умолчанию.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.



```
.flex-container {
  display: -webkit-flex;
  display: flex;
}
.flex-item {
  -webkit-align-self: center;
  align-self: center;
}
```

Выравнивание строк flex-контейнера: align-content

Свойство выравнивает строки в flex-контейнере при наличии дополнительного пространства на поперечной оси, аналогично выравниванию отдельных элементов на главной оси с помощью свойства `justify-content`. Свойство не влияет на однострочный flex-контейнер. Не наследуется.

Значение	align-content
<code>flex-start</code>	Строки укладываются по направлению к началу flex-контейнера. Край первой строки помещается вплотную к краю flex-контейнера, каждая последующая – вплотную к предыдущей строке.
<code>flex-end</code>	Строки укладываются по направлению к концу flex-контейнера. Край последней строки помещается вплотную к краю flex-контейнера, каждая предыдущая – вплотную к последующей строке.
<code>center</code>	Строки укладываются по направлению к центру flex-контейнера. Строки расположены вплотную друг к другу и выровнены по центру flex-контейнера с равным расстоянием между начальным краем содержимого flex-контейнера и первой строкой и между конечным краем содержимого flex-контейнера и последней строкой.
<code>space-between</code>	Строки равномерно распределены в flex-контейнере. Если оставшееся свободное пространство отрицательно или в flex-контейнере имеется только одна flex-линия, это значение идентично <code>flex-start</code> . В противном случае край первой строки помещается вплотную к начальному краю содержимого flex-контейнера, край последней строки – вплотную к конечному краю содержимого flex-контейнера. Остальные строки распределены так, чтобы расстояние между любыми двумя соседними строками было одинаковым.
<code>space-around</code>	Строки равномерно распределены в flex-контейнере с половинным пробелом на обоих концах. Если оставшееся свободное пространство отрицательно, это значение идентично <code>center</code> . В противном случае строки распределяются таким образом, чтобы расстояние между любыми двумя соседними строками было одинаковым, а расстояние между первой / последней строками и краями содержимого flex-контейнера составляло половину от расстояния между строками.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

```
.flex-container {
  display: -webkit-flex;
  -webkit-flex-flow: row wrap;
  -webkit-align-content: flex-end;
  display: flex;
  flex-flow: row wrap;
  align-content: flex-end;
  height: 100px;
}
```



CSS3 columns

Модуль **CSS3 columns** описывает многоколоночный макет, который позволяет организовать содержимое так, чтобы оно занимало несколько вертикальных контейнеров, подобно газете или журналу. Колонки могут содержать заголовки, текст, таблицы, картинки и любые другие inline-элементы.

THE OLD POST

NOVEMBER 24, 2012

15 Most Charming Small Towns In England

<i>1. Berwick-upon-Tweed</i>	<p>Berwick was founded as an Anglo-Saxon settlement during the time of the Kingdom of Northumbria, which was annexed by England in the 10th century. The area was for more than 400 years central to historic border wars between the Kingdoms of England and Scotland, and several times possession of Berwick changed hands between the two kingdoms. The last time it changed hands was when Richard of</p> 	<p>Gloucester retook it for England in 1482. To this day many Berwickers feel a close affinity to Scotland.</p> <p>Nowadays Berwick-upon-Tweed is much-visited for its highly visible history: medieval town walls, Elizabethan ramparts, 13th century castle ruins, its 17th century 'Old Bridge', town hall, Britain's earliest army barracks, England's northernmost hotel, amongst others.</p>
<i>2. Rye</i>		<p>Ancient Rye is all cobbled streets and tumbledown rows of houses by the sea. Originally part of the Cinque Ports Confederation, five strategic towns important for trade and military purposes in medieval times, today Rye is practically a living museum. Rye Castle, popularly known as Ypres Tower, was built in 1249 by Henry III to protect against frequent raids by the French; even older, the Norman-era St. Mary's Church looks over the town. Rye is also just a few minutes away from one of England's most famous beaches, Camber Sands, a two-mile-long playground for kitesurfers and beachlovers.</p>

Количество и ширина колонок

Определение количества и ширины колонок является основополагающим при построении многоколоночного макета. Свойства `column-count` и `column-width` используются для установки количества и ширины колонок.

Третье свойство, `columns`, является сокращенным свойством, которое устанавливает ширину и количество колонок одновременно.

Другие факторы, такие как явные разрывы столбцов, содержимое и ограничения высоты, могут влиять на фактическое количество и ширину колонок.

Ширина колонок: свойство `column-width`

Свойство `column-width` указывает минимальную ширину, которую должен занимать каждый столбец.

Свойство не наследуется.

Значение	<code>column-width</code>
<code>auto</code>	Означает, что ширина столбца будет определяться другими свойствами (например, <code>column-count</code> , если оно имеет значение, отличное от <code>auto</code>). Значение по умолчанию.
<code>длина</code>	Ширина колонок задается в единицах длины, кроме <code>%</code> . Фактическая ширина столбца может быть больше (для заполнения доступного пространства) или уже (только если доступное пространство меньше указанной ширины столбца). Отрицательные значения не допускаются. Используемые значения будут ограничены минимум <code>1px</code> .
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

`column-width: auto;
column-width: 100px;
column-width: 10em;
column-width: 3.3vw;
column-width: inherit;
column-width: initial;`

Количество колонок: свойство `column-count`

Свойство `column-count` описывает количество колонок, а их ширина будет рассчитываться, исходя из ширины доступного пространства. Если одновременно с `column-count` задается `column-width`, то значение `column-count` будет считаться максимальным числом колонок.

Свойство не наследуется.

Значение	<code>column-width</code>
<code>auto</code>	Означает, что количество столбцов будет определяться другим свойством, например, <code>column-width</code> , если оно также не имеет значение <code>auto</code> . Значение по умолчанию.
<code>число</code>	Описывает максимальное количество колонок. Значение задается целым числом, должно быть больше 0.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

`column-count: auto;
column-count: 2;
column-count: inherit;
column-count: initial;`

Установка колонок с помощью одного свойства `columns`

Свойство `columns` – это сокращенное свойство для установки `column-width` и `column-count`. Опущенные значения устанавливаются в их начальные значения.

Свойство не наследуется.

```
columns: 12em; /* column-width: 12em; column-count: auto */
columns: auto 12em; /* column-width: 12em; column-count: auto */
columns: 2; /* column-width: auto; column-count: 2 */
columns: 2 auto; /* column-width: auto; column-count: 2 */
columns: auto; /* column-width: auto; column-count: auto */
columns: auto auto; /* column-width: auto; column-count: auto */
columns: inherit;
columns: initial;
```

Промежутки между колонками и разделительные линии

Промежутки между колонками и разделительные линии помещаются между колонками в одном многоколоночном контейнере. Длина промежутков и разделительных равна высоте колонки. Промежутки в колонках занимают место, то есть, они раздвигают содержимое в соседних колонках.

Разделительная линия рисуется в середине промежутка между колонками, не занимая места. То есть наличие или толщина разделительной линии не изменит размещение чего-либо еще.

Линии закрашиваются чуть выше границы многоколоночного элемента. Если элемент имеет область прокрутки, разделительные линии прокручиваются вместе с колонками. Разделительные линии отображаются только между двумя колонками, которые имеют содержимое.

Промежутки между колонками: свойство `column-gap`

Свойство `column-gap` определяет разрыв между колонками. Если для колонок установлена разделительная линия с помощью свойства `column-rule`, то эта линия будет расположена посередине промежутка, а ее ширина не изменит общую ширину.

Свойство не наследуется.

Значение	<code>column-gap</code>
длина	Промежуток между колонками задается в единицах длины. Значения не могут быть отрицательными. Процентное значение может быть удалено из спецификации.
<code>normal</code>	Эквивалентно <code>1em</code> . Значение по умолчанию.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

```
column-gap: normal;
column-gap: 3px;
column-gap: 2.5em;
column-gap: 3%;
column-gap: inherit;
column-gap: initial;
```

Цвет разделительной линии: свойство `column-rule-color`

Свойство `column-rule-color` определяет цвет разделительной линии.

Свойство не наследуется.

Значение	<code>column-rule-color</code>
цвет	Цвет линии задается с помощью допустимых значений цвета. Значение по умолчанию <code>currentColor</code> .
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

```

column-rule-color: pink;
column-rule-color: #D71C3B;
column-rule-color: rgb(192, 56, 78);
column-rule-color: transparent;
column-rule-color: hsla(0, 100%, 50%, 0.6);
column-rule-color: inherit;
column-rule-color: initial;

```

Стиль разделительной линии: свойство column-rule-style

Свойство `column-rule-style` устанавливает стиль разделительной линии.

Свойство не наследуется.

Значение	<code>column-rule-style</code>
<code>none</code>	Значение вычисляется в <code>0</code> . Значение по умолчанию.
<code>hidden</code>	Аналогично со значением <code>none</code> , линия скрыта.
<code>dotted</code>	Отображает линию набором квадратных точек.
<code>dashed</code>	Отображает линию как последовательность из тире.
<code>solid</code>	Обычная линия.
<code>double</code>	Отображает разделительную линию в виде двух параллельных тонких линий, расположенных на некотором расстоянии между собой. Толщина разделительной линии не указывается, но сумма линий и промежутка между ними равна значению <code>column-rule-width</code> .
<code>groove</code>	Отображает линию объемной, вдавленной в полотно. Это достигается путем создания тени из двух цветов, один из которых темнее, другой – светлее.
<code>ridge</code>	Отображает разделительную линию объемной, т.е. эффект, противоположный <code>groove</code> .
<code>inset</code>	Отображает сплошную линию цветом темнее, чем заданный цвет линии.
<code>outset</code>	Отображает сплошную линию цветом, заданным свойством <code>column-rule-color</code> .
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

```

column-rule-style: none;
column-rule-style: hidden;
column-rule-style: dotted;
column-rule-style: dashed;
column-rule-style: solid;
column-rule-style: double;
column-rule-style: groove;
column-rule-style: ridge;
column-rule-style: inset;
column-rule-style: outset;
column-rule-style: inherit;
column-rule-style: initial;

```

Ширина разделительной линии: свойство column-rule-width

Свойство `column-rule-width` устанавливает ширину разделительной линии. Отрицательные значения не допускаются. Не работает без свойства `column-rule-style`.

Свойство не наследуется.

Значение	<code>column-rule-style</code>
<code>thin</code>	Тонкая линия.
<code>medium</code>	Значение по умолчанию. Средняя толщина линии.
<code>thick</code>	Утолщенная линия.

длина	Ширина разделительной линии задается в единицах длины.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
column-rule-width: thin;
column-rule-width: medium;
column-rule-width: thick;
column-rule-width: 1px;
column-rule-width: 2.5em;
column-rule-width: inherit;
column-rule-width: initial;
```

Краткая запись свойств разделительной линии: свойство column-rule

Свойство `column-rule` является сокращенной записью свойств `column-rule-width` `column-rule-style` `column-rule-color`.

Свойство не наследуется.

```
column-rule: dotted;
column-rule: solid 8px;
column-rule: solid blue;
column-rule: thick inset blue;
column-rule: inherit;
column-rule: initial;
```

Разрыв колонок

Когда содержимое размещено в нескольких колонках, браузер должен определить, где размещаются разрывы колонок. Проблема разбиение контента на колонки аналогична разбиению контента на страницы. Для решения этого вопроса было введено три новых свойства, позволяющих описывать разрывы столбцов в тех же свойствах, что и разрывы страниц: `break-before`, `break-after` и `break-inside`.

Охват колонок: свойство column-span

Свойство `column-span` позволяет элементу охватывать несколько столбцов. Указывается не для блока-контейнера, а для конкретного элемента внутри, например, для заголовка.

В будущем будет возможно указать количество колонок для охвата, подобно атрибуту `colspan`, который может быть применен к ячейке таблицы, но в спецификации CSS3 есть только два возможных значения: `none` и `all`.

Свойство не работает по умолчанию в Firefox. Пользователь должен явно включить функцию, в `layout.css.column-span.enabled` должно быть установлено значение `true`.

Свойство не наследуется.

Значение	column-span
none	Содержимое элемента отображается в пределах одной колонки. Значение по умолчанию.
all	Элемент охватывает все колонки. Колонка разбивается в том месте, где отображается элемент.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
column-span: none;
column-span: all;
column-span: inherit;
column-span: initial;
```

Заполнение колонок содержимым: свойство column-fill

Свойство `column-fill` контролирует заполнение колонок содержимым. Существует две стратегии заполнения колонок: колонки могут быть выровнены по высоте или нет. Если колонки выровнены, браузеры должны попытаться минимизировать изменения высоты колонки, учитывая при этом вынужденные разрывы, `widows`, `orphans` и другие свойства, которые могут влиять на высоту колонок. Если колонки не выровнены, они заполняются последовательно, некоторые из них могут быть частично заполнены или вообще не заполнены.

Свойство не наследуется.

Значение	column-fill
<code>auto</code>	Заполняет колонки последовательно.
<code>balance</code>	Отображает содержимое одинаково во всех колонках. Значение по умолчанию.
<code>balance-all</code>	Выравнивает содержимое равномерно между колонками, насколько это возможно.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

`column-fill: auto;
column-fill: balance;
column-fill: balance-all;
column-fill: inherit;
column-fill: initial;`

Переполнение

Переполнение внутри многоколоночных контейнеров

За исключением случаев, когда это может привести к разрыву колонки, содержимое, которое выходит за границы колонки, выходит за ее границы и не обрезается. Это касается, в первую очередь, изображений. Чтобы решить эту проблему, нужно установить для изображений следующие свойства:

```
img {  
    display: block; /*убираем нижний отступ под картинкой*/  
    width: 100%; /*растягиваем изображение на всю ширину блока-контейнера*/  
}
```

Разбивка на страницы и переполнение вне многоколоночных контейнеров

Содержимое и разделительные линии, которые выходят за рамки колонок по краям многоколоночного контейнера, обрезаются в соответствии со свойством `overflow`.

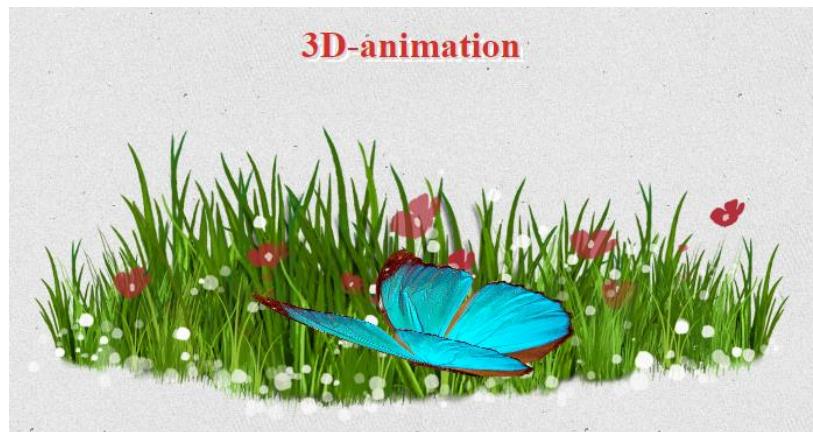
Многоколоночный контейнер может иметь больше колонок, чем у него есть для этого места из-за ограничения высоты колонок (например, с помощью `height` или `max-height`) и явных разрывов колонок. В этом случае дополнительные колонки создаются в направлении строки, перемещаясь на следующие страницы.

CSS3 3D-трансформации

Модуль 3D Transforms расширяет спецификацию CSS 2D Transforms, позволяя преобразовывать элементы в трехмерном пространстве. Новые функции преобразования для свойства `transform` выполняют трехмерные преобразования, расширяя координатное пространство до трех измерений, добавляя ось z, перпендикулярную плоскости экрана, которая увеличивается по направлению к зрителю, а дополнительные свойства позволяют контролировать взаимодействие вложенных трехмерных преобразованных элементов.

Хотя некоторые значения свойства `transform` позволяют преобразовывать элемент в трехмерной системе координат, сами элементы не являются

трехмерными объектами. Они существуют в двумерной плоскости (плоская поверхность) и не имеют глубины.



[Смотреть пример 3D анимации](#)

Свойство transform-style

По умолчанию преобразованные элементы создают плоское представление своего содержимого. Свойство `transform-style` позволяет преобразованным 3D-элементам и их 3D-потомкам использовать общее трехмерное пространство, выстраивая иерархии трехмерных объектов. Отображение 3D-потомков определяется моделью – так называемым контекстом 3D-рендеринга. Отображение зависит от z-позиции элементов в трехмерном пространстве, и если 3D-преобразования этих элементов вызывают пересечение, то они отображаются с пересечением.

Свойство устанавливается для родительского элемента.

Свойство не наследуется.

Значение	transform-style
flat	Значение по умолчанию. Все дочерние элементы отображаются плоскими в двумерной плоскости блока-контейнера.
preserve-3d	Располагает элементы в трехмерном пространстве.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

`transform-style: preserve-3d;
transform-style: flat;
transform-style: inherit;
transform-style: initial;`

[Смотреть пример transform-style](#)

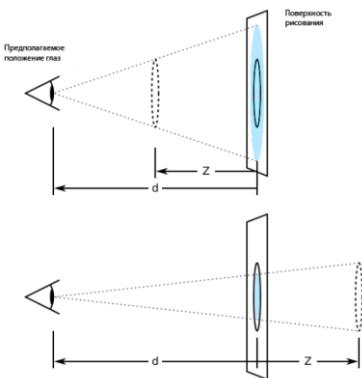
Некоторые значения CSS-свойств элемента, для которого задано `transform-style: preserve-3d`, изменяют используемое значение на `flat` и предотвращают создание или расширение контекста 3D-рендеринга:

- `overflow`: любое значение, кроме `visible` или `clip`.
- `opacity`: любое значение меньше 1.
- `filter`: любое значение, кроме `none`.
- `clip`: любое значение, кроме `auto`.
- `clip-path`: любое значение, кроме `none`.
- `isolation`: если задано значение `isolate`.
- `mask-image`: любое значение, кроме `none`.
- `mask-border-source`: любое значение, кроме `none`.
- `mix-blend-mode`: любое значение, кроме `normal`.

Свойство perspective

В нормальном потоке элементы отображаются плоскими и в той же плоскости, что и блок, содержащий их. Двумерные функции преобразования могут изменять внешний вид элемента, но этот элемент по-прежнему отображается в той же плоскости, что и содержащий его блок.

Свойства `perspective` и `perspective-origin` можно использовать для добавления ощущения глубины в сцену, делая элементы выше по оси z (ближе к зрителю) и кажущимися большими, а те, которые находятся дальше – меньшими. Масштаб пропорционален $d / (d - z)$, где d – значение перспективы, является расстоянием от плоскости рисования до предполагаемого положения глаза зрителя.

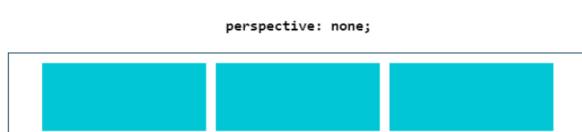


Если 3D-перспектива задается с помощью функции `perspective()`, 3D-пространство активизируется только для одного элемента. Свойство `perspective` активирует 3D-пространство внутри элемента, содержащего дочерние трансформированные элементы и применяется к ним.

Свойство не наследуется.

Значение	<code>perspective</code>
<code>длина</code>	Задает расстояние до центра проекции, т.е. расстояние по оси z. Значение может быть любым положительным числом, заданным в единицах длины. Чем больше значение, тем менее выражен эффект. <code>0</code> означает отсутствие перспективы.
<code>none</code>	Значение по умолчанию. Означает отсутствие перспективы.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

```
perspective: none;  
perspective: 100px;  
perspective: 10em;  
perspective: inherit;  
perspective: initial;
```



`perspective: 400px;`



`perspective: 800px;`



Свойство perspective-origin

Обычно предполагаемое положение глаза зрителя находится в центре рисунка. Свойство `perspective-origin` управляет точкой начала координат, позволяя изменять направление трансформации дочернего 3D-элемента. Свойство должно использоваться вместе со свойством `perspective` для родительского элемента и свойством `transform` для дочернего элемента.

Свойство не наследуется.

Значение	<code>perspective-origin</code>
позиция точки	<p>Свойство принимает два значения, первое задает координату x, второе – y. Свойство может принимать следующие значения:</p> <ul style="list-style-type: none">• <code>%</code> – для горизонтального смещения перспективы определяется относительно ширины виртуальной рамки, для вертикального смещения – относительно высоты;• значение, указанное в единицах длины, задает фиксированную длину смещения. Значение смещения по горизонтали и вертикали представляет собой смещение от верхнего левого угла виртуальной рамки;• <code>top</code> вычисляется в <code>0%</code> для вертикального положения, если задано одно или два значения, в противном случае определяет верхний край как исходную точку для следующего смещения;• <code>right</code> вычисляется в <code>100%</code> для горизонтального положения, если задано одно или два значения, в противном случае указывает правый край в качестве исходной точки для следующего смещения;• <code>bottom</code> вычисляется в <code>100%</code> для вертикального положения, если задано одно или два значения, в противном случае указывает нижний край в качестве исходной точки для следующего смещения;• <code>left</code> вычисляется в <code>0%</code> для горизонтального положения, если задано одно или два значения, в противном случае определяет левый край в качестве исходной точки для следующего смещения;• <code>center</code> вычисляется <code>50%</code> (<code>left 50%</code>) для горизонтального положения, если горизонтальное положение не указано иначе, или <code>50%</code> (<code>top 50%</code>) для вертикального положения, если оно есть. <p>Значение по умолчанию <code>50% 50%</code>.</p>
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

```
perspective-origin: 30px;  
perspective-origin: left center;  
perspective-origin: 200% 200%;  
perspective-origin: inherit;  
perspective-origin: initial;
```



Свойство backface-visibility

Используя трехмерные преобразования, можно преобразовать элемент так, чтобы его обратная сторона была видна. 3D-преобразованные элементы отображают одинаковое содержимое с обеих сторон, поэтому обратная сторона выглядит как зеркальное отображение лицевой стороны. Свойство `backface-visibility` позволяет делать элемент невидимым, когда его обратная сторона обращена к зрителю.

Свойство полезно, когда вы создаете флип-карту, размещая два элемента вплотную друг к другу, или куб из 6 элементов.

Свойство не наследуется.

Значение	<code>backface-visibility</code>
<code>visible</code>	Значение по умолчанию. Указывает, что обратная сторона видна.
<code>hidden</code>	Скрывает обратную сторону элемента.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента. <code>backface-visibility: visible; backface-visibility: hidden; backface-visibility: inherit; backface-visibility: initial;</code>

[Смотреть пример backface-visibility](#)

Функции 3D-трансформации

Свойство задает вид как 2D, так и 3D-преобразований элемента. 3D-преобразования описываются с помощью функций трансформации, перечисленных в таблице ниже.

Свойство не наследуется.

Функция	Описание
<code>matrix3d(n,n,n,n, n,n,n,n, n,n,n,n, n,n,n,n)</code>	Функция задает трехмерное преобразование как однородную матрицу размером 4×4 с шестнадцатью значениями в столбцах. Все другие функции преобразований основаны на данной функции.
<code>translate3d(x,y,z)</code>	Функция задает перемещение элемента в 3D-пространстве. Движение происходит по вектору $[tx, ty, tz]$, где tx – перемещение вдоль оси x , ty – перемещение вдоль оси y , а tz – вдоль оси z . Значения могут задаваться в единицах длины или в %. Отрицательные значения будут перемещать элемент в противоположном направлении.
<code>translateZ(z)</code>	Функция задает перемещение элемента на заданное расстояние в направлении оси z . Значения могут задаваться в единицах длины или в %. Отрицательные значения будут перемещать элемент в противоположном направлении.
<code>scale3d(x,y,z)</code>	Функция задает операцию трехмерного масштабирования по вектору масштабирования $[sx, sy, sz]$, описываемому тремя параметрами. Отрицательные значения отображают элемент зеркально вдоль трех осей.

scaleZ(z)	Функция масштабирует элемент в направлении оси z, делая его больше или меньше. В качестве значения задается число. Результат функции наиболее выражен при совместном использовании с такими функциями, как <code>rotate()</code> и <code>perspective()</code> .
rotate3d(x,y,z,угол)	Функция вращает элемент по часовой стрелке относительно трех осей. Элемент поворачивается под углом, задаваемым последним параметром относительно вектора направления [x,y,z]. Отрицательные значения поворачивают элемент против часовой стрелки.
rotateX(угол)	Функция задает поворот по часовой стрелке под заданным углом относительно оси x. Функция <code>rotateX(180deg)</code> эквивалентна <code>rotate3d(1,0,0,180deg)</code> .
rotateY(угол)	Функция задает поворот по часовой стрелке под заданным углом относительно оси y. Функция <code>rotateY(180deg)</code> эквивалентна <code>rotate3d(0,1,0,180deg)</code> .
rotateZ(угол)	Функция задает поворот по часовой стрелке под заданным углом относительно оси z. Функция <code>rotateZ(180deg)</code> эквивалентна <code>rotate3d(0,0,1,180deg)</code> .
perspective(n)	Функция меняет перспективу обзора элемента, создавая иллюзию глубины. Чем больше значение функции перспективы, тем дальше от смотрящего расположен элемент. Значение должно быть больше нуля.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```

transform: none;
transform: translate3d(100px, 100px, -200px);
transform: translate3d(50%, -100%, 10%);
transform: translate3d(-100px, -30px, 50px);
transform: translateZ(300px);
transform: translateZ(-50%);
transform: translateZ(150%);
transform: scale3d(2, 1, 3);
transform: scale3d(-1, -2, -1);
transform: scaleZ(3);
transform: scaleZ(-1);
transform: rotate3d(1, 1, 1, 45deg);
transform: rotateX(30deg);
transform: rotateX(-135deg);
transform: rotateY(30deg);
transform: rotateY(-135deg);
transform: rotateZ(30deg);
transform: rotateZ(-135deg);
transform: perspective(300);
transform: perspective(300px);
transform: inherit;
transform: initial;

```

[Смотреть пример 3D Transform](#)

CSS3-медиавыражения

В 2001 году в HTML4 и CSS2 была введена поддержка аппаратно-зависимых таблиц стилей, позволившая создавать стили и таблицы стилей для определенных типов устройств. В качестве медиатипов были определены следующие: `aural`, `braille`, `handheld`, `print`, `projection`, `screen`, `tty`, `tv`. Таким

образом, браузер применял таблицу стилей только в случае, когда активизировался данный тип устройства.

Кроме того, было введено ключевое слово `all`, которое использовалось, чтобы указать, что таблица стилей применяется ко всем типам носителей.

В HTML4 медиавыражение записывался следующим образом:

```
<link rel="stylesheet" type="text/css" media="screen" href="sans-serif.css">
<link rel="stylesheet" type="text/css" media="print" href="serif.css">
```

Внутри таблицы стилей также можно было объявить, что блоки объявлений должны применяться к определенным типам носителей:

```
@media screen {
  * {font-family: sans-serif;}
```

Предусматривая возможность введения новых значений и значений с параметрами в будущем, для браузеров была реализована поддержка значений атрибута медианосителя, указанных следующим образом:

```
<link rel="stylesheet" media="screen, 3d-glasses, print and resolution > 90dpi" href="...">
```

Текущий синтаксис HTML5 и CSS3 напрямую ссылается на первую спецификацию Media Queries, обновляя правила для HTML. Также был расширен список характеристик медианосителей.

[Использование медиавыражений.](#)

CSS3-фильтры

CSS3-фильтры воспроизводят в браузере визуальные эффекты, похожие на фильтры Photoshop. Фильтры можно добавлять не только к изображениям, но и к любым непустым элементам.

Набор фильтров не ограничивается предустановленным в браузере. Вы также можете использовать фильтры SVG, загрузив их по ссылке вместе с элементом `svg`.

Изначально фильтры были созданы как часть спецификации SVG. Их задачей было применение эффектов, основанных на пиксельной сетке к векторной графике. С поддержкой SVG браузерами появилась возможность использовать эти эффекты непосредственно в браузерах.

Браузеры обрабатывают страницу попиксельно применяя заданные эффекты и отрисовывают результат поверх оригинала. Таким образом, применяя несколько фильтров можно достигать различных эффектов, они как бы накладываются друг на друга. Чем больше фильтров, тем больше времени требуется браузеру, чтобы отрисовать страницу.

Можно применять несколько фильтров одновременно. Классический способ применения таких эффектов – при наведении на элемент `::hover`.

Значение	filter
<code>blur()</code>	Значение задается в единицах длины, например <code>px</code> , <code>em</code> . Применяет размытие по Гауссу к исходному изображению. Чем больше значение радиуса, тем больше размытие. Если значение радиуса не задано, по умолчанию берется <code>0</code> .
<code>brightness()</code>	Значение задается в <code>%</code> или в десятичных дробях. Изменяет яркость изображения. Чем больше значение, тем ярче изображение. Значение по умолчанию <code>1</code> .
<code>contrast()</code>	Значение задается в <code>%</code> или в десятичных дробях. Регулирует контрастность изображения, т.е. разницу между самыми темными и самыми светлыми участками изображения/фона. Значение по умолчанию <code>100%</code> . Нулевое значение скроет

	исходное изображение под темно-серым фоном. Значения, увеличивающиеся от 0 до 100% или от 0 до 1, будут постепенно открывать исходное изображение до оригинального отображения, а значения свыше будут увеличивать контраст между светлыми и темными участками.
drop-shadow()	Фильтр действует подобно свойствам <code>box-shadow</code> и <code>text-shadow</code> . Использует следующие значения: смещение по оси x смещение по оси y размытость растяжение цвет тени. Отличительная особенность фильтра заключается в том, что тень добавляется к элементам и его содержимому с учетом их прозрачности, т.е. если элемент содержит текст внутри, то фильтр добавит тень одновременно для текста и видимых границ блока. В отличие от других фильтров, для этого фильтра обязательно задание параметров (минимальное – величина смещения).
grayscale()	Извлекает все цвета из картинки, делая на выходе черно-белое изображение. Значение задается в % или десятичных дробях. Чем больше значение, тем сильнее эффект.
hue-rotate()	Меняет цвета изображения в зависимости от заданного угла поворота в цветовом круге. Значение задается в градусах от 0deg до 360deg. 0deg – значение по умолчанию, означает отсутствие эффекта.
invert()	Фильтр делает негатив изображения. Значение задается в %. 0% не применяет фильтр, 100% полностью преобразует цвета.
opacity()	Фильтр работает аналогично со свойством <code>opacity</code> , добавляя прозрачность элементу. Отличительная особенность – браузеры обеспечивают аппаратное ускорение для фильтра, что позволяет повысить производительность. Дополнительный бонус – фильтр можно одновременно сочетать с другими фильтрами, создавая при этом интересные эффекты. Значение задается только в %, 0% делает элемент полностью прозрачным, а 100% не оказывает никакого эффекта.
saturate()	Управляет насыщенностью цветов, работая по принципу контрастного фильтра. Значение 0% убирает цветность, а 100% не оказывает никакого эффекта. Значения от 0% до 100% уменьшают насыщенность цвета, выше 100% – увеличивают насыщенность цвета. Значение может задаваться как в %, так и целым числом, 1 эквивалентно 100%.
sepia()	Эффект, имитирующий старину и «ретро». Значение 0% не изменяет внешний вид элемента, а 100% полностью воспроизводит эффект сепии.
url()	Функция принимает расположение внешнего XML-файла с svg-фильтром, или якорь к фильтру, находящемся в текущем документе.
none	Значение по умолчанию. Означает отсутствие эффекта.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```

filter: blur(3px);
filter: brightness(50%);
filter: brightness(.5);
filter: contrast(20%);
filter: contrast(.2);
filter: drop-shadow(2px 3px 5px black);
filter: grayscale(.5);
filter: grayscale(50%);
filter: hue-rotate(180deg);
filter: invert(100%);
filter: opacity(30%);
filter: saturate(300%);
filter: sepia(150%);
filter: url(#filterId); /* если фильтр находится в этом документе */
filter: url(filter.svg#filterId); /* если фильтр с id="filterId" находится в файле filter.svg*/

```

[Смотреть пример CSS3 Filters](#)

CSS Grid

W3C описывает модуль CSS Grid Layout как систему двумерного макета, оптимизированного для дизайна пользовательского интерфейса. Главная идея, лежащая в основе макета сетки, заключается в разделении веб-страницы на столбцы и строки. В образовавшиеся области сетки можно помещать элементы сетки, а управлять их размерами и расположением можно с помощью специальных свойств модуля.

Кроме того, благодаря своей способности явно размещать элементы в сетке, Grid Layout позволяет кардинально преобразовывать структуру визуального макета (отображаемого на экране), не требуя соответствующих изменений разметки.

Хотя многие макеты могут быть отображены с помощью Grid или Flexbox, у каждого есть свои особенности. Grid обеспечивает двухмерное выравнивание, использует исходящий подход к макету, допускает явное перекрытие элементов и обладает более мощными связующими возможностями. Flexbox фокусируется на распределении пространства по оси, использует более простой восходящий подход к макету, может использовать систему переноса строк на основе размера контента для управления своей вторичной осью и опирается на базовую иерархию разметки для построения более сложных макетов.

Во всех примерах будет рассмотрен стандартный синтаксис. Если вы захотите адаптировать синтаксис для IE10-11 и Microsoft Edge, воспользуйтесь руководством Microsoft Developer Network Grid layout.

Концепция сетки и основные понятия



Сетка (grid) представляет собой набор пересекающихся горизонтальных и вертикальных линий, делящих пространство grid-контейнера на области сетки, в которые могут быть помещены элементы сетки.

Линии сетки (grid lines) – это невидимые горизонтальные и вертикальные разделятельные линии, они существуют по обе стороны от строки и столбца. На них можно ссылаться по числовому индексу (используя свойства `grid-column-start`, `grid-column-end`, `grid-row-start` и `grid-row-end`) или имени, заданному в CSS-коде. Числовые индексы сетки зависят от стиля языка, поэтому первым столбцом может быть, как самый левый, так и самый правый столбец.

Выделяют две группы линий сетки: одна группа определяет столбцы, которые проходят вдоль оси блока (ось столбцов), и перпендикулярная группа, определяющая строки, простирающиеся вдоль линейной оси (ось строк), в соответствии с CSS3 режимом записи.

Дорожка сетки (grid track) – пространство между двумя соседними линиями сетки, используется для определения либо столбца, либо строки сетки. Дорожка идет от одного края контейнера к другому, размер зависит от расположения линий сетки, которые ее определяют. Дорожки сетки аналогичны столбцам и строкам таблицы. По умолчанию смежные дорожки плотно прилегают друг к другу, задать расстояние между ними можно с помощью свойств `row-gap`, `column-gap` и `gap`.

Ячейка сетки (grid cell) – пространство, ограниченное четырьмя линиями сетки, аналогично ячейке таблицы. Ячейка сетки – это область, в которой можно разместить контент. Это наименьшая единица сетки, на которую можно ссылаться при позиционировании элементов сетки. К ячейкам сетки нельзя обращаться напрямую с помощью CSS-свойств.

Область сетки (grid area) – прямоугольная область, ограниченная четырьмя линиями сетки и состоящая из одной или нескольких соседних ячеек. Область может быть такой же маленькой, как одна ячейка, или такой же большой, как все ячейки сетки. Область сетки может быть задана явно с помощью свойства `grid-template-areas`, по умолчанию на нее ссылаются ограничивающие линии сетки.

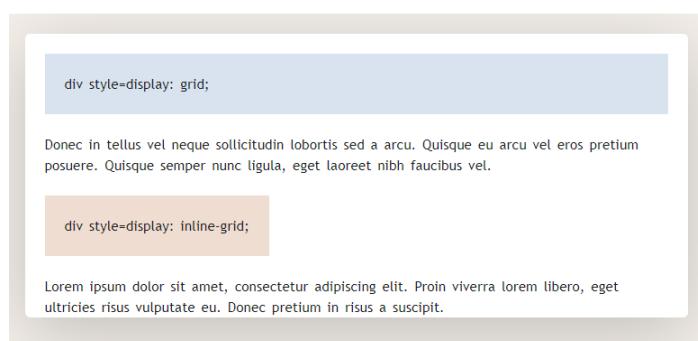
Элементы сетки (grid items) – отдельные элементы, которые назначаются области сетки (или ячейке сетки). Каждый контейнер-сетка включает ноль и более элементов сетки; каждый дочерний элемент контейнера-сетки автоматически становится элементом сетки.

Дорожки, ячейки и области сетки построены из линий сетки. Тем не менее не требуется, чтобы все области сетки были заполнены элементами, вполне возможно, что некоторые или даже большинство ячеек сетки будут пустыми от любого содержимого. Также возможно, что элементы сетки будут перекрывать друг друга, либо определять перекрывающиеся области сетки.

Контейнер-сетка

Для создания макета на основе сетки необходимо определить контейнер-сетку.

Контейнер-сетка (grid container) – это блок, который устанавливает контекст форматирования по типу сетки, то есть создает область с сеткой, а дочерние элементы располагаются в соответствии с правилами компоновки сетки, а не блочной компоновки. Когда вы определяете контейнер сетки с помощью `display: grid` или `display: inline-grid`, вы создаете новый контекст форматирования для содержимого этого контейнера, который влияет только на дочерние элементы сетки.



Контейнер-сетка бывает двух видов: обычный `display: grid` и встроенный `display: inline-grid`. Первый генерирует `grid`-контейнер уровня блока, второй – `grid`-контейнер уровня строки. Контейнеры-сетки не являются блочными контейнерами, поэтому некоторые CSS-свойства не работают в контексте макета сетки:

- `float` и `clear` игнорируются элементами сетки (но не самим контейнером-сеткой).
- `vertical-align` не влияет на элементы сетки.
- Псевдоэлементы `::first-line` и `::first-letter` не применяются к контейнеру-сетке и его потомкам.
- Если контейнер-сетка является контейнером уровня строки `display: inline-grid` и для него заданы обтекание или абсолютное позиционирование, то вычисляемое значение свойства `display` будет `grid`.

Определение сетки

Когда вы создаете контейнер-сетку, сетка по умолчанию имеет один столбец и одну строку, которые занимают полный размер контейнера. Для разделения контейнера-сетки на столбцы и строки используются свойства `grid-template-columns`, `grid-template-rows` и `grid-template-areas`. С помощью этих свойств можно определить сетку явно.

Окончательная сетка может оказаться больше из-за элементов сетки, размещенных вне явной сетки; в этом случае будут созданы неявные дорожки, размер этих неявных дорожек будет определяться свойствами `grid-auto-rows` и `grid-auto-columns`.

Свойства `grid` и `grid-template` – это сокращенные обозначения, которые можно использовать для одновременной установки всех трех явных свойств сетки `grid-template-columns`, `grid-template-rows` и `grid-template-areas`. `grid` сбрасывает свойства, управляющие неявной сеткой, тогда как свойство `grid-template` оставляет их без изменений.

Строки и столбцы

Количество строк / столбцов определяется с помощью свойств `grid-template-rows` и `grid-template-columns`. Свойства не наследуются.

Значение	<code>grid-template-rows</code> , <code>grid-template-columns</code>
<code>none</code>	Указывает, что свойство не создает явных дорожек сетки (хотя явные дорожки сетки все еще могут создаваться свойством <code>grid-template-areas</code>). При отсутствии явной сетки любые строки/столбцы будут генерироваться неявно, а их размер будет определяться свойствами <code>grid-auto-rows</code> и <code>grid-auto-columns</code> . Значение по умолчанию.
список дорожек / автоматический список дорожек	Устанавливает список дорожек в виде последовательности функций определения размера дорожек и названий линий сетки. Каждая функция определения размера дорожки может быть задана в единицах длины, как процент от размера контейнера-сетки или доля свободного пространства в сетке. Размер также может быть указан как диапазон с помощью нотации <code>minmax()</code> .

Относительные, абсолютные единицы и процентные значения для определения дорожек сетки (длина)

Размеры дорожек сетки можно задавать с помощью положительных значений, используя относительные единицы длины – например, `em`, `vh`, `vw`; абсолютные единицы длины – `px`; и проценты `%`. Размеры в `%` вычисляются от ширины или высоты контейнера-сетки.

```
.grid-container {
  display: grid;
  grid-template-rows: 5em 200px 200px; /* 3 строки */
  grid-template-columns: 200px 5em 50%; /* 3 столбца */
}
```

Гибкие размеры дорожек: единица измерения fr

`fr` – единица длины, которая позволяет создавать гибкие дорожки. Не является единицей измерения в обычном ее понимании, поэтому не может быть представлена или объединена с другими типами единиц в выражениях `calc()`. Общий размер фиксированных строк или столбцов вычитается из доступного пространства контейнера-сетки. Оставшееся пространство делится между строками и столбцами с гибкими размерами пропорционально их коэффициенту, например:

```
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr 1fr; /* эквивалентно grid-template-columns: 25% 25% 25% 25%; */
}
.grid-container {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr; /* эквивалентно grid-template-columns: 25% 50% 25%; */
}
```

Если сумма всех гибких размеров дорожек меньше `1`, они будут занимать только соответствующую долю оставшегося пространства, а не расширяться, чтобы заполнить его полностью.

Если доступное пространство бесконечно (то есть, ширина или высота контейнера-сетки не заданы), дорожки сетки гибкого размера масштабируются по своему содержимому, сохраняя при этом их соответствующие пропорции.

Минимальные и максимальные размеры дорожек

Ключевое слово `max-content` устанавливает для дорожки размер, который занимает максимально необходимое пространство с учетом содержимого элемента сетки.

`min-content` позволяет занимать минимальное пространство, необходимое для этого содержимого, при этом ширина элемента ориентируется на самое длинное слово или на самое широкое изображение.

Функция `minmax(min,max)` определяет диапазон размеров, больше или равный `min` и меньше или равный `max`. Если `max < min`, то `max` игнорируется, а `minmax(min,max)` обрабатывается как `min`. Значения в `fr` можно устанавливать только как максимальное.

```
minmax(длина или min-content или max-content или auto, длина или fr или min-content или max-content или auto)
minmax(длина, длина или fr или min-content или max-content или auto)
minmax(длина или min-content или max-content или auto, длина)
```

```
.grid-container {
  display: grid;
  grid-template-rows: 200px minmax(100px, 1fr);
}
```

Автоматические размеры

Значение `auto` ориентируется на содержимое элементов сетки одной дорожки. Как минимум, рассматривается как минимальный размер элемента сетки, как определено `min-width` или `min-height`. Как максимум, обрабатывается так же, как и `max-content`. Может растягиваться за счет свойств `align-content` и `justify-content`.

```
.grid-container {
  display: grid;
  grid-template-rows: auto 1fr;
  grid-template-columns: auto 1fr auto;
```

Соответствие содержимому

Размеры дорожек можно задавать с помощью значения `fit-content(длина или %)`, представляющее собой формулу `min(maximum size, max(minimum size, argument))`, которая вычисляется как `minmax(auto, max-content)`, то есть `auto`.

При этом, размер дорожки ограничивается значением, указанным в скобках, и если оно больше, чем автоматический минимум.

```
.grid-container {  
    display: grid;  
    grid-template-columns: fit-content(50%) fit-content(300px) 1fr;  
}
```

Повтор строк и столбцов

Нотация `repeat()` представляет повторяющийся фрагмент списка дорожек, что позволяет записать в более компактной форме большое количество одинаковых по размерам столбцов или строк. Общая форма синтаксиса, следующая:

```
repeat(число или auto-fill или auto-fit, повторяющаяся дорожка)
```

```
.grid-container {  
    display: grid;  
    grid-template-rows: repeat(3, 200px);  
}
```

Первый аргумент задает количество повторений, которое может быть задано с помощью положительного целого числа или ключевых слов. Второй аргумент – размер повторяющейся дорожки. Однако, существуют некоторые ограничения:

- Нотация `repeat()` не может быть вложенной.
- Значения `auto-fill` или `auto-fit` не могут быть совмещены с `min-content`, `max-content`, `auto`, `fit-content()` или `fr`.

Синтаксис `repeat()` имеет несколько форм:

```
/*повтор дорожки*/  
repeat(количество повторений, имя дорожки? размер дорожки + имя дорожки?)  
/*автозаполнение дорожек сетки*/  
repeat(auto-fill или auto-fit, имя дорожки? фиксированный размер дорожки + имя дорожки?)  
/*фиксированный повтор дорожки*/  
repeat(количество повторений, имя дорожки? фиксированный размер дорожки + имя дорожки?)
```

Используя значение `auto-fill`, вы всегда получите хотя бы один столбец, даже если он по какой-то причине не помещается в контейнер-сетку. Если вы используете `auto-fit`, то дорожки, которые не содержат элементы сетки, будут сброшены.

Именованные области

Свойство `grid-template-areas` определяет именованные области сетки, которые не связаны с каким-либо конкретным элементом сетки, но на которые можно ссылаться из свойств размещения сетки. Синтаксис свойства обеспечивает визуализацию структуры сетки, облегчая понимание общего макета контейнера-сетки. Свойство не наследуется.

Значение	<code>grid-template-areas</code>
<code>none</code>	Указывает, что никакие именованные области сетки и никакие явные дорожки сетки не определены этим свойством (хотя явные дорожки сетки все еще могут быть созданы с помощью <code>grid-template-columns</code> или <code>grid-template-rows</code>). При отсутствии явной сетки любые строки/столбцы будут генерироваться неявно, а их размер будет определяться свойствами <code>grid-auto-rows</code> и <code>grid-auto-columns</code> . Значение по умолчанию.
<code>строка +</code>	Последовательность идентификаторов, определяющая, как должны отображаться строки и столбцы.

```

.grid-container {
  display: grid;
  grid-template-areas: "header header"
                      "sidebar content"
                      "sidebar content";
  grid-template-columns: 150px 1fr;
  grid-template-rows: 50px 1fr 50px;
}
header {
  grid-area: header;
}
aside {
  grid-area: sidebar;
}
main {
  grid-area: content;
}

```

Каждый идентификатор сетки в значении `grid-template-areas` соответствует ячейке сетки. Как только все ячейки идентифицированы, браузер объединяет все смежные ячейки с одинаковыми именами в одну область, которая охватывает все их, при условии, что они описывают область прямоугольной формы. Если вы попытаетесь настроить более сложные области, весь шаблон будет недействительным и области сетки не будут определены.

Все строки должны содержать одинаковое количество столбцов. Если вы хотите определить только некоторые ячейки как часть области сетки, вы можете использовать одну или несколько `.` для заполнения этих безымянных ячеек. При определении областей сетки идентификаторы можно перечислять через единичный пробел, без разрыва строки. Или же выровнять с помощью пробелов/табуляции и перевода строки для большей наглядности.

Области сетки полезны для определения семантических отношений между различными частями макета страницы, позволяя указать, какая часть страницы включает в себя верхний колонтитул, боковую панель, область содержимого и нижний колонтитул.

После того, как вы создали области сетки, элементы сетки могут быть назначены непосредственно, чтобы занимать эти области, используя свойство `grid-area`.



Краткая запись явной сетки

Свойство `grid-template` является сокращением для установки `grid-template-rows`, `grid-template-columns` и `grid-template-areas` в одном объявлении.

Значение	<code>grid-template</code>
<code>none</code>	Устанавливает для всех трех свойств начальные значения <code>none</code> . Будет определяться свойствами <code>grid-auto-rows</code> и <code>grid-auto-columns</code> . Значение по умолчанию.
значение <code>grid-template-rows</code> / <code>grid-template-columns</code>	Устанавливает <code>grid-template-rows</code> и значение <code>grid-template-columns</code> в указанные значения, а <code>grid-template-areas</code> в значение <code>none</code> .

имена линий сетки или последовательность идентификаторов, заключенная в кавычки и размер дорожки или именованные линии сетки или + /явный список дорожек

Устанавливает `grid-template-areas` для перечисленных последовательностей идентификаторов. Устанавливает для `grid-template-rows` указанные значения размеров дорожек, следующие за каждой последовательностью идентификаторов (выставляя `auto` для любых отсутствующих размеров), и объединяет в именованных линиях сетки, определенных до / после каждого размера. Устанавливает `grid-template-columns` в список дорожек, указанный после косой черты (или ни одного, если не указан).

```
.grid-container {
  display: grid;
  grid-template: repeat(3, 200px)/repeat(3, 1fr);
}

.post-1 {
  grid-row-start: 1;
  grid-row-end: 3;
  grid-column-start: 1;
  grid-column-end: 3;
}

.post-2 {
  grid-row-start: 1;
  grid-row-end: 2;
  grid-column-start: 3;
  grid-column-end: 4;
}

...

.grid-container {
  display: grid;
  grid-template: [start] "post-1 post-1 post-2" 200px [row2]
                  [row2] "post-1 post-1 post-3" 200px [row3]
                  [row3] "post-6 post-5 post-4" 200px [row-end] / 1fr 1fr 1fr;
}

.post-1 {
  grid-area: post-1;
}
.post-2 {
  grid-area: post-2;
}
```

Функция `repeat()` не разрешена для определения списка дорожек в этом свойстве, если используются именованные области сетки (сетка просто не будет отрисована).

Неявная сетка

Автоматические дорожки сетки

Если элемент сетки расположен в строке или столбце, размер которых не определен явно `grid-template-rows` или `grid-template-columns`, создаются неявные дорожки сетки для его хранения. Это может произойти в случае, если строка или столбец оказались за пределами установленных размеров сетки.

По умолчанию эти автоматически добавляемые дорожки имеют минимальный необходимый размер. Свойства `grid-auto-rows` и `grid-auto-columns` позволяют контролировать размер неявных дорожек сетки. Если дано несколько размеров дорожек, шаблон повторяется по мере необходимости, чтобы найти размер неявных дорожек. Первая неявная дорожка сетки после явной сетки получает первый заданный размер и так далее. Свойства не наследуются.

Значение	<code>grid-auto-columns</code> , <code>grid-auto-rows</code>
<code>auto</code>	Значение по умолчанию.
размер дорожки +	В качестве размера дорожки может использоваться любое значение, допустимое для задания размеров дорожек сетки.



```

.grid-container {
  max-width: 710px;
  display: grid;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(3,100px);
  grid-auto-rows: 50px;
}
.post-1 {
  grid-column: 1/3;
  grid-row: 1/3;
}
.post-2 {
  grid-column: 3;
  grid-row: 1;
}
.post-3 {
  grid-column: 3;
  grid-row: 2;
}
.post-4 {
  grid-column: 3;
  grid-row: 3;
}
.post-5 {
  grid-column: 2;
  grid-row: 3;
}
.post-6 {
  grid-column: 1;
  grid-row: 3;
}

```

Автоматическое размещение

Элементы сетки, которые не размещены явно, автоматически помещаются в незанятое пространство в контейнере-сетке с помощью алгоритма автоматического размещения. Свойство `grid-auto-flow` управляет автоматическим размещением элементов сетки без явного положения. После заполнения явной сетки (или если явной сетки нет) автоматическое размещение также приведет к генерации неявных дорожек сетки. Свойство не наследуется.

Значение	<code>grid-auto-flow</code>
<code>row</code>	Алгоритм автоматического размещения размещает элементы, заполняя каждую строку по очереди слева-направо (для LTR-языков), добавляя новые строки по мере необходимости. Значение по умолчанию.
<code>column</code>	Алгоритм размещает элементы, заполняя каждый столбец по очереди сверху-вниз, добавляя новые столбцы по мере необходимости.
<code>dense</code>	Алгоритм «плотной» укладки элементов. При необходимости может менять порядок следования элементов, заполняя пустые места более крупными элементами.

Свойство будет полезным при создании компактных галерей, если для изображений не задан порядок, в котором они должны быть расположены. Для каждого элемента сетки браузер сканирует всю сетку в заданном направлении потока (строка или столбец), начиная от начальной точки потока (верхний левый угол, на языках LTR – слева направо), пока не найдет место, куда поместится этот элемент сетки.



```

.grid-container {
  max-width: 710px;
  margin: 10px auto;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 10px;
  grid-auto-rows: 200px;
  grid-auto-flow: dense;
}
.item5 {
  grid-column: span 2;
  grid-row: span 2;
}
.item6 {
  grid-column: span 3;
  grid-row: span 2;
}

```

Краткая запись сетки

Свойство `grid` задает все явные `grid-template-rows`, `grid-template-columns` и `grid-template-areas` и все неявные свойства сетки `grid-auto-flow`, `grid-auto-rows` и `grid-auto-columns` в одном объявлении. Оно не сбрасывает свойства `row-gap/column-gap`. Его синтаксис соответствует `grid-template`, а также дополнительной синтаксической форме для определения автоматического размещения элементов сетки.

Путем явного задания дорожек на одной оси (устанавливая `grid-template-rows` или `grid-template-columns` и задавая другим значение `none`), и задавая, как автоматически повторять дорожки на другой оси (устанавливая `grid-auto-rows` или `grid-auto-columns` и задавая другим `auto`).

Для `grid-auto-flow` также устанавливается одно из трех допустимых значений. Все остальные подсвойства `grid` сбрасываются к своим начальным значениям.

```
/* значения для grid-template */
grid: none;
grid: "a" 100px "b" 1fr;
grid: [linename1] "a" 100px [linename2];
grid: "a" 200px "b" min-content;
grid: "a" minmax(100px, max-content) "b" 20%;
grid: 100px / 200px;
grid: minmax(400px, min-content) / repeat(auto-fill, 50px);

/* значения для grid-template-rows / [auto-flow && dense? ] grid-auto-columns? */
grid: 200px / auto-flow;
grid: 30% / auto-flow dense;
grid: repeat(3, [line1 line2 line3] 200px) / auto-flow 300px;
grid: [line1] minmax(20em, max-content) / auto-flow dense 40%;

/* значения для [ auto-flow && dense? ] grid-auto-rows? / grid-template-columns */
grid: auto-flow / 200px;
grid: auto-flow dense / 30%;
grid: auto-flow 300px / repeat(3, [line1 line2 line3] 200px);
grid: auto-flow dense 40% / [line1] minmax(20em, max-content);
```

Элементы сетки

Контейнер-сетка устанавливает новый контекст форматирования для элементов сетки, который обуславливает следующие особенности:

- Для элементов сетки блокируется их значение свойства `display`. Значение `display: inline-block` вычисляется в `display: block`, анонимные блоки текста также занимают всю ширину контейнера и образуют разрыв строки.
- Размер элемента сетки в пределах содержащего блока определяется его областью сетки.
- Расчеты элементов сетки для `width: auto` и `height: auto` зависят от их значений `align-self: align-self: normal;` – незамещаемые элементы заполняют область сетки, замещаемые элементы используют собственные размеры; `align-self: stretch;` – обе категории элементов заполняют область сетки; `align-self: start/center` и т.д. – незамещаемые элементы устанавливают размеры в соответствии со своим содержимым, замещаемые элементы используют собственные размеры.
- Поскольку соседние элементы сетки находятся в независимых областях сетки, то поля соседних элементов сетки `margin` не схлопываются.
- Браузеры по-разному обрабатывают процентные значения свойств `margin` и `padding`, поэтому не рекомендуется использовать их при задании значений этих свойств.
- Поля `margin: auto;` расширяются, поглощая свободное пространство в соответствующем измерении, поэтому могут использоваться для выравнивания элемента.

Размещение и переупорядочивание элементов сетки

Свойства размещения позволяют свободно упорядочивать и переупорядочивать содержимое сетки таким образом, что визуальное представление может значительно отличаться от порядка элементов в html-документе.

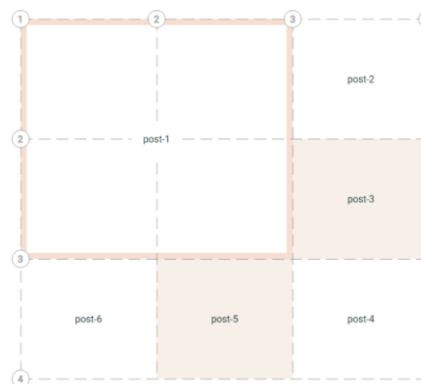
Размещение с помощью линий сетки

Каждый элемент сетки связан с областью сетки, которая определяет содержащий блок для элемента сетки. Положение элементов сетки определяется расположением линий сетки и диапазоном сетки – количеством занимаемых дорожек сетки. По умолчанию элемент сетки занимает одну дорожку на каждой оси. Поэтому можно опустить значение `grid-column-end` или `grid-row-end`.

Свойства размещения на сетке – `grid-row-start`, `grid-row-end`, `grid-column-start` и `grid-column-end` и их краткая запись `grid-row`, `grid-column` и `grid-area` позволяют определить размещение элемента сетки, предоставив любую (или ноль) из следующих шести частей информации:

	Строка	Столбец
Начало	Начальная линия строки	Начальная линия столбца
Конец	Конечная линия строки	Конечная линия столбца
Диапазон	Диапазон строк	Диапазон столбцов

Значение	grid-row-start, grid-column-start, grid-row-end, grid-column-end
auto	Свойство не влияет на размещение элемента сетки, указывая на автоматическое размещение или диапазон по умолчанию, равный единице.
имя линии	Алгоритм размещает элементы, заполняя каждый столбец по очереди сверху-вниз, добавляя новые столбцы по мере необходимости.
целое число и имя линии?	Начальная и конечная линия строки/столбца задаются с помощью целого числа (отрицательное порядковый номер линии сетки будет отсчитываться с противоположного края явной сетки) и (необязательно) имени линии.
span и целое число или имя линии	Ключевое слово <code>span</code> и целое положительное число/имя линии задают диапазон ячеек для размещения элемента сетки.



```
grid-container {
  display: grid;
  grid-template-columns: 200px 200px 200px;
  grid-template-rows: 1fr 1fr 1fr;
}

.post-1 {
  grid-row-start: 1;
  grid-row-end: 3;
  grid-column-start: 1;
  grid-column-end: 3;
}

.post-2 {
  grid-row-start: 1;
  grid-column-start: 3;
}

.post-3 {
  grid-row-start: 2;
  grid-column-start: 3;
}

.post-4 {
  grid-row-start: 3;
  grid-column-start: 3;
}

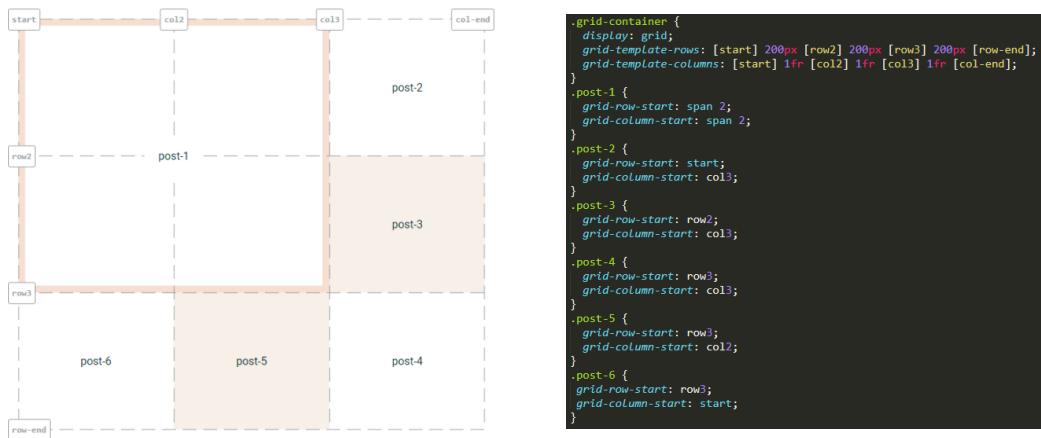
.post-5 {
  grid-row-start: 3;
  grid-column-start: 2;
}

.post-6 {
  grid-row-start: 3;
  grid-column-start: 1;
}
```

Именованные линии сетки

Хотя на линии сетки можно ссылаться по их числовому индексу, именованные линии облегчают понимание и использование свойств размещения сетки. Линии могут быть названы явно в свойствах `grid-template-rows` и `grid-template-columns` или неявно путем создания именованных областей сетки в свойстве `grid-template-areas`.

Имя линии может быть любым, при указании в значении свойства оно заключается в квадратные скобки. В качестве имени линии нельзя использовать слово `span`.



Имена линий добавляются к неявным именам линий сетки, созданным свойством `grid-template-areas`, принимая вид `name-start` и `name-end`. Имена линий сетки никогда не заменяют другие имена линий сетки. Вместо этого они просто накапливаются.

Краткая запись свойств размещения элементов сетки

Свойства `grid-row` и `grid-column` являются сокращенными именами для свойств `grid-row-start/grid-row-end` и `grid-column-start/grid-column-end` соответственно.

Если заданы два значения, первое (до косой черты) устанавливается для параметра `grid-row-start/grid-column-start`, второе – для `grid-row-end/grid-column-end`. Если второе значение опущено, а первое указано в формате пользовательского идентификатора, то `grid-row-end/grid-column-end` также устанавливается в пользовательское имя сетки. В противном случае, оно вычисляется в `auto`.

Для свойства `grid-area` если указано четыре значения, первое устанавливается для `grid-row-start`, второе – для `grid-column-start`, третье – для `grid-row-end`, четвертое – для `grid-column-end`.

Если `grid-column-end/grid-row-end` не указан, а `grid-column-start/grid-row-start` указан в форме пользовательского имени, то для `grid-column-end/grid-row-end` также устанавливается значение пользовательского имени линии; в противном случае он установлен на `auto`.

Когда `grid-column-start` опущен, а значение `grid-row-start` указан в форме пользовательского имени, оно устанавливается для всех четырех значений. В противном случае оно устанавливается на `auto`.



Переупорядочивание элементов сетки

Свойство `order` также применяется к элементам сетки. Это влияет на их автоматическое размещение и порядок отрисовки. Свойство должно использоваться только для визуального, а не логического переупорядочения контента.

Выравнивание элементов сетки и промежутки между элементами

Для выравнивания элементов сетки можно использовать свойство `margin`, аналогично, как работает это свойство для блочных элементов.

По умолчанию элементы сетки растягиваются, чтобы заполнить свою область сетки. Тем не менее, если `justify-self` или `align-self` вычисляют значение, отличное от `stretch` или задано `margin: auto`, элементы сетки будут автоматически изменяться в соответствии с их содержимым.

Выравнивание с помощью `margin: auto`

При расчете размеров дорожек сетки `margin: auto` обрабатываются как `0`. Они поглощают положительное свободное пространство, предшествующее выравниванием с помощью свойств выравнивания. Переполняющиеся элементы игнорируют свои автоматические поля и переполнение, как указано в их свойствах выравнивания блоков.

Выравнивание по оси строки

Элементы сетки могут быть выровнены в направлении оси строки (по горизонтали для LTR-языков) с помощью свойства `justify-self` или свойства `justify-items` (заданного для контейнера-сетки).

Выравнивание по оси столбца

Элементы сетки могут выровнены в направлении, перпендикулярном оси строки с помощью свойства `align-self` или свойства `align-items`, заданного для контейнера-сетки.

Промежутки между элементами сетки

Свойства `row-gap` и `column-gap` (и их сокращенная запись `gap`), если они указаны в контейнере сетки, определяют промежутки между строками и столбцами сетки. При определении размера дорожки каждый промежуток рассматривается как дополнительная пустая дорожка указанного размера. Дополнительный промежуток также может быть добавлен между дорожками за счет свойств `justify-content` и `align-content`.

Промежутки добавляются только между двумя дорожками сетки, то есть они не добавляются перед первой и после последней дорожки.

Значение	<code>row-gap</code> , <code>column-gap</code>
<code>normal</code>	Вычисляется как <code>0px</code> . Значение по умолчанию.
длина или %	Процентное значение вычисляется относительно размеров области сетки. Отрицательные значения не используются. <code>row-gap: 1.5em; column-gap: 10px; gap: 1%;</code>

CSS3-шрифты

Современные технологии шрифтов поддерживают множество расширенных типографских и языковых функций. Используя их, один шрифт может предоставлять глифы для широкого спектра лигатур, контекстных и стилистических альтернатив, табличных и старых символов, маленьких прописных, автоматических дробей, штрихов и альтернатив, специфичных для данного языка.

Чтобы позволить авторам контролировать эти возможности шрифта, свойство `font-variant` было расширено для CSS3. Теперь оно функционирует как сокращение для набора свойств, которые обеспечивают контроль над стилистическими функциями шрифта.

Одному символу можно задать множество вариаций глифа.

U+0061 LATIN SMALL LETTER A

a a a a a a a

Кернинг: свойство font-kerning

Кернинг – это контекстная настройка межглифового интервала. Свойство **font-kerning** позволяет избирательно изменять интервал между символами в зависимости от их формы. Для шрифтов, которые не содержат данных кернинга, это свойство не будет иметь видимого эффекта. Свойство наследуется.

A V A V

Описание свойства font-kerning.

Лигатуры: свойство font-variant-ligatures

Лигатуры и контекстные формы – это способы объединения глифов в один знак-глиф для создания более гармоничных форм. Свойство **font-variant-ligatures** определяет, какие лигатуры и контекстные формы используются в текстовом содержимом элементов. Свойство наследуется.

fi ► fi
WORDS ► WORDS
tj ► j
labor of love ► labor of love

Описание свойства font-variant-ligatures.

Преобразование в заглавные буквы: свойство font-variant-caps

Свойство **font-variant-caps** контролирует использование альтернативных глифов для заглавных букв. Свойство наследуется.

The passions of PRIDE and HUMILITY
The passions of PRIDE and HUMILITY

Описание свойства font-variant-caps.

Форматирование цифр: свойство font-variant-numeric

Свойство **font-variant-numeric** контролирует использование альтернативных глифов для чисел, дробей и порядковых маркеров. Свойство наследуется.

2 1/3 ► 2½
2 1/3 ► 2 $\frac{1}{3}$
1st 17th 2a ► 1st 17th 2^a
4000 ► 4000

Описание свойства font-variant-numeric.

Общее сокращение для рендеринга шрифтов: свойство font-variant

Свойство **font-variant** является сокращением для всех подсвойств вариантов шрифта. Не сбрасывает значения **font-feature-settings**.

Значение	font-variant
normal	Сбрасывает все подсвойства вариантов шрифтов к их начальному значению. Значение по умолчанию.
none	Устанавливает для font-variant-ligatures значение none и сбрасывает все остальные свойства шрифта на значение по умолчанию.

значения отдельных свойств вариантов шрифта	Определяет ключевые слова и функции, относящиеся к конкретному свойству.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Свойство `font-variant` является сокращением для всех подсвойств вариантов шрифта. Не сбрасывает значения `font-feature-settings`.

```
font-variant: normal;
font-variant: none;
font-variant: small-caps;
font-variant: common-ligatures small-caps;
font-variant: inherit;
font-variant: initial;
```

CSS3-переполнение

Модуль CSS Overflow содержит функции CSS для обработки прокручиваемого переполнения, отображаемого на визуальных носителях (экранах устройств). CSS использует термин переполнение для описания содержимого блока, которое простирается за пределы одного из его краев, т.е. края области содержимого, поля, границы или отступа.

Типы переполнения

Существует два типа переполнения, которое используется браузерами: переполнение эффектов рисования и прокручиваемое переполнение.

Первый тип переполнения представляет собой часть блока и его содержимое, которые создают визуальные эффекты за пределами рамки элемента. К визуальным эффектам относятся тень блока, рамки элемента, тень текста, нависающие глифы, контуры и т.д.

Прокручиваемое переполнение представляет собой набор элементов, выходящих за пределы области полей блока, для которых должен быть предусмотрен механизм прокрутки (полоса прокрутки, или скроллбар).

Прокрутка и обрезка переполнения: свойства `overflow-x`, `overflow-y` и `overflow`

Эти свойства определяют, будет ли содержимое блока, включая переполнение любых эффектов рисования, обрезано по краю области полей, и, если это так, будет ли этот блок контейнером прокрутки, позволяющим пользователю прокручивать обрезанные части его содержимого. Применяются к блочным, гибким контейнерам и контейнерам-сеткам.

Свойство `overflow-x` определяет обработку переполнения в горизонтальном направлении (т.е. переполнение с левой и правой сторон блока), а свойство `overflow-y` определяет обработку переполнения в вертикальном направлении (т.е. переполнение с верхней и нижней сторон блока).

Свойства не наследуются.

Значение	<code>overflow-x</code> , <code>overflow-y</code>
<code>visible</code>	Значение по умолчанию. Содержимое не обрезается, а отображается поверх границ блока-контейнера. Возможно перекрытие соседних блоков.
<code>hidden</code>	Содержимое блока обрезается без добавления какого-либо интерфейса прокрутки для просмотра содержимого вне области обрезки.

scroll	Содержимое обрезается до области полей, при этом блок становится прокручиваемым контейнером. Если браузер использует механизм прокрутки, который виден на экране, например, полосу прокрутки, этот механизм отображается независимо от того, обрезано ли какое-либо его содержимое. Это позволяет избежать проблем с появлением и исчезновением полос прокрутки в динамической среде. Размеры контейнера при этом не меняются, а полоса прокрутки вставляется между внутренним краем границы и внешним краем поля элемента.
auto	Браузер использует механизм прокрутки только при переполнении блока.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
overflow-x: visible;
overflow-x: hidden;
overflow-x: scroll;
overflow-y: auto;
overflow-x: inherit;
overflow-x: initial;
```

Свойство `overflow` – сокращенное свойство, которое задает значения `overflow-x` и `overflow-y` в указанном порядке. Если второе значение опущено, оно копируется из первого.

Содержимое блочных элементов может переполнять блок в случае, когда для блока явно задана высота и/или ширина. Без указания высоты блок будет растягиваться, чтобы вместить содержимое, кроме случаев, когда для блока задано позиционирование `position: absolute;` или `position: fixed;`. Текст может переполнять блок по высоте, изображения – по высоте и ширине.

Значение	<code>overflow</code>
<code>visible</code>	Значение по умолчанию. Содержимое не обрезается, а отображается поверх границ блока-контейнера. Возможно перекрытие соседних блоков.
<code>hidden</code>	Содержимое блока обрезается без добавления какого-либо интерфейса прокрутки для просмотра содержимого вне области обрезки. Также предотвращает отображение фона или границ под плавающими элементами, для которых задано свойство <code>float: left / right;</code> .
<code>scroll</code>	Содержимое обрезается до области полей, при этом блок становится прокручиваемым контейнером. Если браузер использует механизм прокрутки, который виден на экране, например, полосу прокрутки, этот механизм отображается независимо от того, обрезано ли какое-либо его содержимое. Это позволяет избежать проблем с появлением и исчезновением полос прокрутки в динамической среде. Размеры контейнера при этом не меняются, а полоса прокрутки вставляется между внутренним краем границы и внешним краем поля элемента.
<code>auto</code>	Браузер использует механизм прокрутки только при переполнении блока.

initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
overflow: visible;
overflow: hidden;
overflow: scroll;
overflow: hidden scroll;
overflow: auto;
overflow: inherit;
overflow: initial;
```

Автоматическое многоточие

Многоточие при переполнении: свойство text-overflow

Свойство позволяет обрезать строчное содержимое в случае, когда оно не умещается в блок-контейнер, визуально обрезая его или отображая многоточием. Текст может **переполниться**, например, когда ему запрещается перенос, например, из-за `white-space: nowrap` или отдельное слово слишком длинное, чтобы уместиться.

Свойство работает только при задании следующих условий: должна быть определена ширина блока-контейнера, элемент должен иметь значения `overflow: hidden` и `white-space: nowrap`. Применяется только к блочным контейнерам.

Свойство не наследуется.

Значение	text-overflow
clip	Значение по умолчанию. Текст обрезается в пределе области содержимого, при этом может отобразиться лишь часть символа.
ellipsis	Замещает текст, не уместившийся в блок, с помощью многоточия.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
text-overflow: clip;
text-overflow: ellipsis;
text-overflow: initial;
text-overflow: inherit;
```

CSS3-способы письма

Модуль CSS Writing Modes определяет поддержку различных направлений письма: слева направо (например, латинское или индийское письмо), справа налево (например, иврит или арабское письмо), двунаправленный (например, смешанное латинское и арабское письмо) и вертикальный (например, азиатское письмо).

Направление текста в html-документах

Способы письма в CSS определяются свойствами `writing-mode`, `direction` и `text-orientation`.

Направление содержимого по линии строки (свойство `direction`), с заданным началом и концом строки, является основным.

Направление по линии блоков – направление, в котором выкладываются блоки с текстом, оно регулируется с помощью свойства `writing-mode`.

Горизонтальное письмо – это способ письма с горизонтальными строками текста, то есть с нисходящим или восходящим потоком блоков.

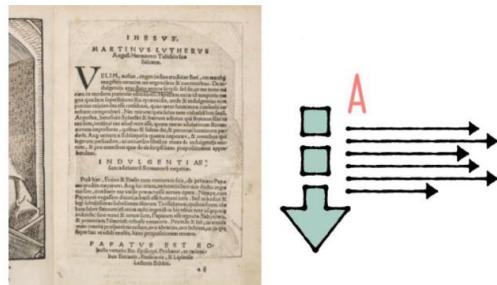
Вертикальное письмо – это способ письма с вертикальными строками текста, то есть с потоком блоков влево или вправо.

Также существует типографский режим, который определяет, должен ли текст следовать стандартам оформления текстового материала, характерным для вертикального направления вертикальных скриптов. Эта концепция отличает вертикальное направление вертикальных скриптов от повернутого горизонтального направления.

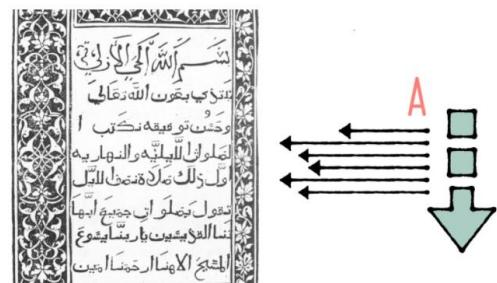
Под скриптом в данном контексте подразумевается набор символов, используемый для отображения письменного текста в одной или нескольких системах письменности. Некоторые системы письменности требуют несколько скриптов (например, японская, которая требует, как минимум три скрипта: Хирагана и Катакана и иероглифы Кандзи, импортированные из Китая).

Различные виды письменности имеют один или два собственных способа письма:

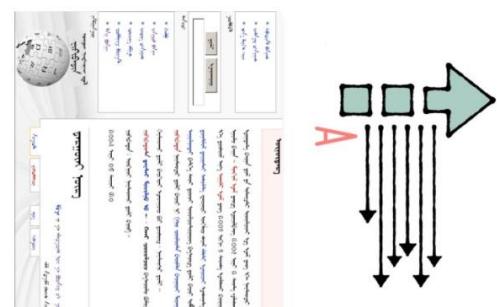
Системы на основе латинского алфавита обычно пишутся с использованием линейного направления слева направо с направлением потока блоков сверху вниз.



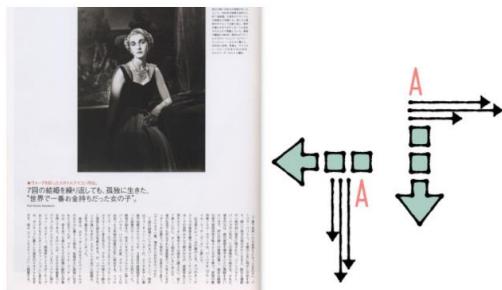
Арабские системы обычно пишутся с использованием линейного направления справа налево с направлением потока блоков сверху вниз.



Монгольские системы обычно пишутся с использованием линейного направления сверху вниз с направлением потока слева направо.



Азиатские системы обычно пишутся с использованием линейного направления слева направо с направлением потока блока сверху вниз или линейного направления сверху вниз с направлением потока блоков справа налево. Многие журналы и газеты смешивают эти два способа письма на одной странице.



Свойство `text-orientation` управляет ориентацией глифа.

Глиф – это базовая единица письменности – буква, знак, символ.

Направление вдоль линии строки и двунаправленность

В некоторых документах текст в одном блоке может отображаться со смешанной направленностью. Это явление называется двунаправленностью. Стандарт [Unicode](#) устанавливает алгоритм для упорядочения двунаправленного текста.

Задаем направление: свойство `direction`

Свойство `direction` устанавливает базовое направление двунаправленного абзаца. Также свойство сообщает порядок расположения столбцов таблицы, направление горизонтального переполнения `overflow`, выравнивание текста по умолчанию в строке и другие эффекты макета, которые зависят от базового направления строк в блоке.

Тем не менее, рекомендованный способ задания направление текста – с помощью атрибута `dir` элемента `<html>` и элемента `<bdo>`, а не посредством прямого использования свойства `direction`, которое не сможет обеспечить корректное отображение текста при отключенной таблице стилей.

Свойство `direction` не влияет на переупорядочение двунаправленного текста, если для вложенных элементов задано значение `unicode-bidi: normal;`.

Свойство `direction`, если оно указано для столбцов таблицы, не наследуется ячейками в столбце, поскольку столбцы не являются предками ячеек в дереве документа.

Свойство наследуется.

Значение	<code>direction</code>
<code>ltr</code>	Значение по умолчанию, устанавливает базовое направление строк слева направо.
<code>rtl</code>	Строки текста отражаются справа налево.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

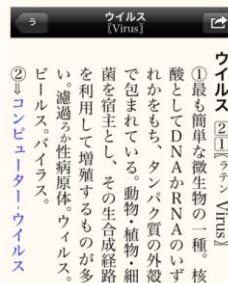
```
direction: ltr;
direction: rtl;
direction: initial;
direction: inherit;
```

Вертикальное письмо

В отличие от языков, использующих латинский алфавит, которые в основном расположены горизонтально, азиатские языки, такие как китайский и японский, могут быть расположены вертикально. Приведенный ниже примере показывает один и тот же текст, выложенный по горизонтали и вертикали. В горизонтальном случае текст читается слева направо, сверху вниз. Для вертикального случая текст читается сверху вниз, справа налево.

Переход от горизонтального письма к вертикальному может повлиять не только на макет, но и на набор текста. Например, положение знака препинания в пределах его интервала может изменяться от горизонтального к вертикальному регистру, а в некоторых случаях используются альтернативные глифы.

Вертикальный текст, который содержит текст латинского алфавита или текст из других сценариев, обычно отображаемых горизонтально, может отображать этот текст несколькими способами. Например, латинские слова могут быть повернуты в сторону, или каждая буква может быть ориентирована вертикально:



В некоторых особых случаях, таких как двузначные числа в датах, текст компактно помещается в одно вертикальное поле символов:

10月27日、マイクロソフトの定番ビジネスサイト「マイクロソフト・オフィス」の最新版となる「オフィス2011 (Office for Mac 2011)」がついに発売されました。2009年8月、マイクロソフトは新しいMac版オフィスについてのロードマップを示しました。当時、2010年のホリデーシーズンに発売する予定であることや、新たに統合メールソフトの「アウトルック」を搭載することなどが発表されました。詳細はあまり明らかにされず、Macユーザーは長い間その詳細の発表を心待ちしていました。

Направление потока блоков: свойство writing-mode

Свойство writing-mode указывает, расположены ли строки текста по горизонтали или по вертикали, а также задает направление потока блоков. Применяется ко всем элементам, кроме столбцов и строк таблицы, основного контейнера ruby и ruby-контейнера с аннотацией.

Свойство наследуется.

Значение	writing-mode
horizontal-tb	Значение по умолчанию. Направление потока сверху вниз. И способ письма, и типографский режим являются горизонтальными.
vertical-rl	Направление потока справа налево. И способ письма, и типографский режим являются вертикальными.
vertical-lr	Направление потока слева направо. И способ письма, и типографский режим являются вертикальными.
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

```
writing-mode: horizontal-tb;
writing-mode: vertical-rl;
writing-mode: vertical-lr;
writing-mode: initial;
writing-mode: inherit;
```

В современных типографских системах всем глифам присваивается горизонтальная ориентация, которая используется при горизонтальном расположении текста. Чтобы выложить вертикальный текст, браузер должен преобразовать текст из его горизонтальной ориентации. Это преобразование является двунаправленным, и существует два типа:

- вращение – глиф поворачивается из горизонтального положения в вертикальное;
- перемещение – глиф перемещается из горизонтального положения в вертикальное.

Скрипты с родной вертикальной ориентацией имеют внутреннее двунаправленное преобразование, которое правильно ориентирует их в вертикальном тексте: большинство символов СЖК (китайский/японский/корейский) перемещаются, то есть они всегда вертикально. Символы из других сценариев, таких как монгольский, вращаются.

Ориентация текста: свойство `text-orientation`

Свойство `text-orientation` определяет ориентацию текста внутри строки. Текущие значения действуют только в вертикальных типографских режимах: свойство не влияет на блоки в горизонтальных типографских режимах. Применяется ко всем элементам, кроме рядов и колонок таблицы.

Свойство наследуется.

Значение	<code>text-orientation</code>
<code>mixed</code>	Значение по умолчанию. Символы из горизонтальных сценариев набираются боком, то есть повернуты на 90° по часовой стрелке от их стандартной ориентации в горизонтальном тексте. Типографские единицы символов из вертикальных сценариев печатаются в соответствии с их внутренней ориентацией.
<code>upright</code>	Символы из горизонтальных сценариев набираются вертикально в их стандартной горизонтальной ориентации. Символы из вертикальных сценариев набираются с их внутренней ориентацией, то есть, весь текст набирается в вертикальном положении.
<code>sideways</code>	Весь текст набирается сбоку (повернут на 90° по часовой стрелке), как будто в горизонтальной разметке.
<code>initial</code>	Устанавливает значение свойства в значение по умолчанию.
<code>inherit</code>	Наследует значение свойства от родительского элемента.

```
text-orientation: mixed;  
text-orientation: upright;  
text-orientation: sideways;  
text-orientation: initial;  
text-orientation: inherit;
```

def 月
木水
金土
abc

def 月
e 火
f 水
木 A
金 b
土 c

def 月
火 *
木金土
Abc

CSS-генераторы

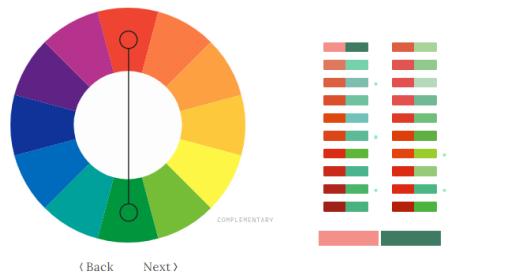
CSS-генераторы и онлайн-сервисы CSS упрощают процесс веб-разработки. С их помощью вы можете получить готовый кроссбраузерный код для различных элементов интерфейса, протестировать, как выглядит сайт на

экранах мобильных устройств, подобрать цветовую палитру для сайта и многое другое.

[Colorion](#) – огромная коллекция цветовых палет, в том числе для создания материального и плоского дизайна.



[Color Supply](#) – генератор цветовых схем.



[Автопрефиксер онлайн](#) – добавляет нужные вендорные префиксы и удаляет ненужные в вашем CSS.

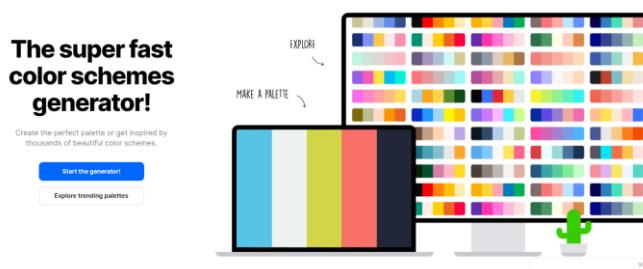
 [Автопрефиксер CSS онлайн](#)
Автопрефикс — плагин для PostCSS для добавления вендорных префиксов в CSS

PostCSS v8.3.6 autoprefixer v9.0.1

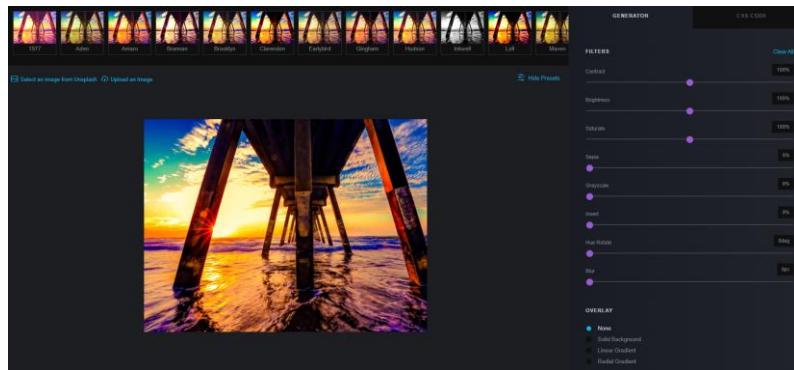
```
[example {  
    display: grid;  
    transition: all .5s;  
    user-select: none;  
    background: linear-gradient(to bottom, white, black);  
}  
  
/*  
 * Prefixed by https://autoprefixer.github.io  
 * PostCSS: v8.3.6  
 * Autoprefixer: v9.0.1  
 * Browsers: last 4 versions  
 */  
  
.example {  
    display: -ms-grid;  
    display: grid;  
    -webkit-transition: all .5s;  
    -o-transition: all .5s;  
    transition: all .5s;  
    -webkit-user-select: none;  
    -moz-user-select: none;  
    -ms-user-select: none;  
    user-select: none;  
background: -webkit-gradient(linear, left top, left bottom, from(white), to(black));  
background: -o-linear-gradient(top, white, black);  
background: linear-gradient(to bottom, white, black);  
}
```

Фильтрация браузеров last 4 version Применить Выделить результат

[Coolors](#) – сервис для подбора цветовых схем для сайта, альтернативных теней, определение цветов по картинке.



[cssFilters](#) – пользовательские фильтры и фильтры как в Инстаграмм для изображений.



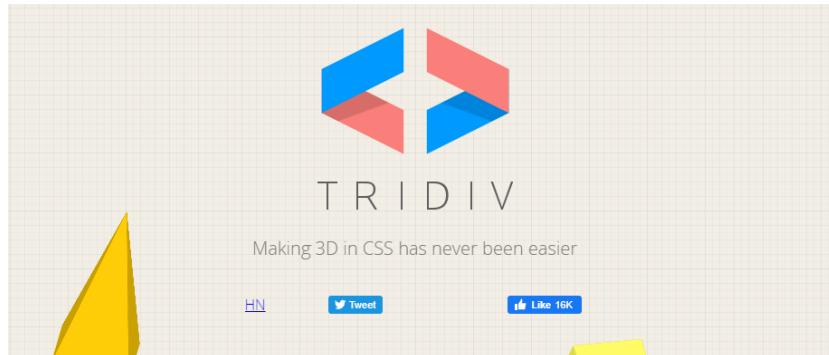
[Type Scale](#) – визуальный калькулятор для расчета размера заголовков.

A screenshot of the Type Scale tool. On the left, there's a 'Base Size' input set to 16px (100%/1em). Below it are 'Scale' (set to 1.250 ~ Major Third), 'Google Fonts' (Poppins, weight 400), and a 'Preview Text' area showing 'A Visual Type Scale'. There are also 'Reset All' and 'Save for Later' buttons, and a note about an 'eBook coming soon'. On the right, there's a visual type scale with various font sizes: 3.052rem/48.83px, 2.441rem/39.06px, 1.953rem/31.25px, 1.563rem/25.00px, 1.25rem/20.00px, 1rem/16.00px, 0.8rem/12.80px, 0.64rem/10.24px, and 0.512rem/8.19px. Below the scale are 'Grab the CSS' and 'Edit on CodePen' buttons.

[Image Slider Maker](#) – онлайн-сервис для создания адаптивного слайдера с картинками.

A screenshot of the Image Slider Maker tool. It shows a slider with a flower image. The top bar has 'Image Slider Maker' and icons for 'Make your slider' and 'Gallery'. The main area has a caption 'My slide caption text'. Below the image are navigation arrows and dots. On the left, there's an 'Upload' section with a grid of preview images. To the right are several configuration panels: 'Dimensions' (set to 900px, 25%, and a width of 900px), 'Back / Fwd Buttons' (with 'On' checked), 'Transitions' (with a red play button icon), 'Radio Buttons' (with 'On' checked), and a 'Reset all' button.

[Tridiv](#) – 3D-редактор для создания объемных моделей на чистом CSS.



The screenshot shows the Tridiv web-based editor interface. It features a grid workspace with several 3D models. One prominent model is a detailed 3D rendering of a Star Wars X-wing fighter. Other smaller models include a complex mechanical structure, a multi-rotor drone-like vehicle, and a simple geometric shape. The interface includes various toolbars and panels for editing.

Tridiv is a web-based editor
for creating 3D shapes in CSS.

[Start using the app](#)

[See examples](#)



CSS-генераторов огромное количество. Какие CSS-генераторы можно использовать читайте в этой [статье](#).

Валидация CSS

Валидацией называется проверка CSS-кода на соответствие спецификации CSS2.1 или CSS3. Соответственно, корректный код, не содержащий ошибок, называется валидный, а не удовлетворяющий спецификации — невалидный. Наиболее удобно делать проверку кода через сайт <http://jigsaw.w3.org/css-validator/>, с помощью этого сервиса можно указать адрес документа, загрузить файл или проверить набранный текст. Большим плюсом сервиса является поддержка русского и украинского языка.

В CSS3 как вы уже знаете добавлено много новых стилевых свойств по сравнению с предыдущей версией, поэтому проводить проверку кода следует с учетом версии.

The screenshot shows the W3C CSS Validation Service interface. At the top, there are language selection buttons for Deutsch, English, Español, Français, 한국어, Italiano, Nederlands, 日本語, Polski, Português, Русский, עברית, Svenska, Български, Українська, Čeština, Romanian, Magyar, Eλληνικά, and 简体中文. The main title is "CSS Validation Service" with the subtitle "Проверка каскадных таблиц стилей (CSS) и документов (X)HTML с таблицами стилей". Below the title are three buttons: "Проверить URI", "Проверить загруженный файл", and "Проверить набранный текст". The "Проверить загруженный файл" section is active, showing a file input field with the placeholder "Выберите файл" and a message "Файл не выбран". Below it is a "Дополнительные возможности" section with dropdown menus for "Профиль" (set to CSS3), "Среда" (set to Все), "Предупреждения" (set to Обычный отчет), and "Расширения поставщика" (set to По-умолчанию). A "Проверить" button is at the bottom. At the bottom of the page is a circular logo for W3Cx and a promotional message about their professional certificate program.

Interested in "developing" your developer skills? In W3Cx's hands-on Professional Certificate Program, learn how to code the right way by creating Web sites and apps that use the latest Web standards. [Find out more!](#)

[Donate](#) and help us build better tools for a better web.

Примечание: Если вы хотите проверить каскадные таблицы стилей, встроенные в документ (X)HTML, вы должны сначала [проверить на корректность сам документ \(X\)HTML](#).

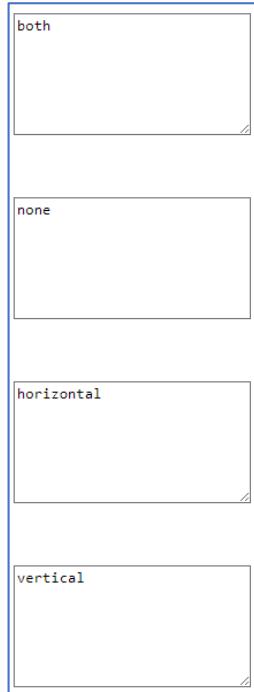
Задание к лабораторной работе №4

Для выполнения лабораторной работы необходимо установить и настроить редактор кода.

Задание к лабораторной работе №4 состоит из задач разного уровня сложности. **Выполнение задач 1-10 оценивается максимально в 5 баллов, задач 1-14 в 10 баллов.**

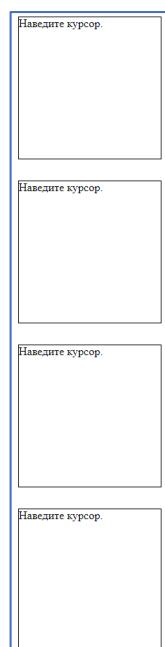
Задача 1

Условие: Повторите страницу по образцу. Используйте свойство `resize`. Осуществите проверку `.html`, `.css` файлов на валидность. [Пояснение](#).



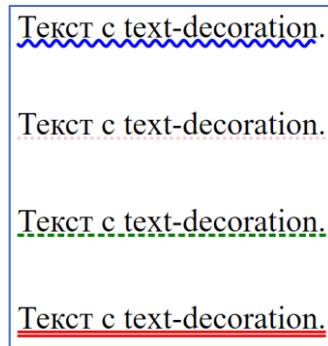
Задача 2

Условие: Повторите страницу по образцу. Используйте свойство `cursor`. Изображение для последнего курсора можно выбрать любое. Осуществите проверку `.html`, `.css` файлов на валидность. [Пояснение](#).



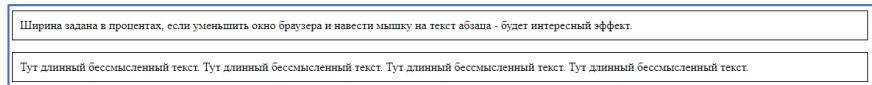
Задача 3

Условие: Повторите страницу по образцу. Используйте свойства `text-decoration-style`, `text-decoration-line`, `text-decoration-color`. Осуществите проверку `.html`, `.css` файлов на валидность.



Задача 4

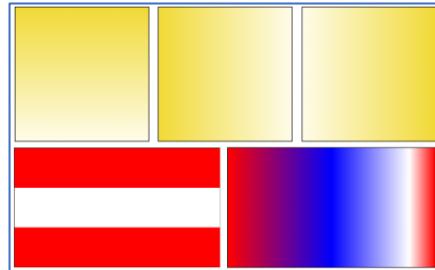
Условие: Повторите страницу по образцу. Используйте свойство `text-overflow`. Осуществите проверку `.html`, `.css` файлов на валидность. [Пояснение](#).



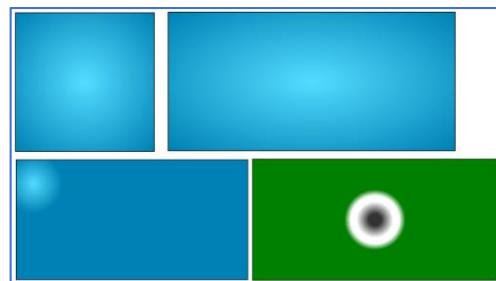
Задача 5

Условие: Повторите страницы по образцу. Осуществите проверку `.html`, `.css` файлов на валидность.

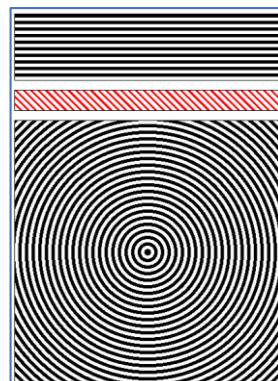
1. Используйте свойство `linear-gradient`.



2. Используйте свойство `radial-gradient`.

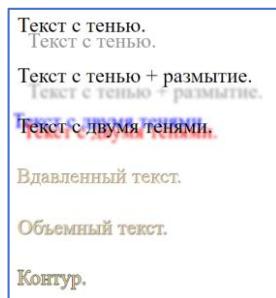


3. Используйте функции `repeating-linear-gradient()`, `repeating-radial-gradient()`.



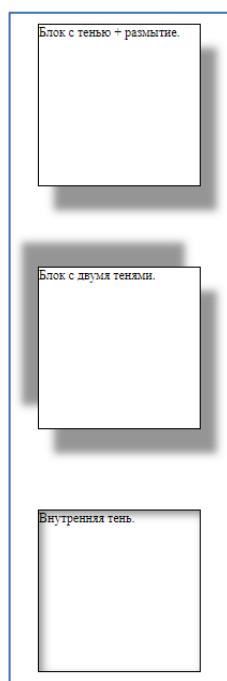
Задача 6

Условие: Повторите страницу по образцу. Используйте свойство `text-shadow`. Осуществите проверку `.html`, `.css` файлов на валидность.



Задача 7

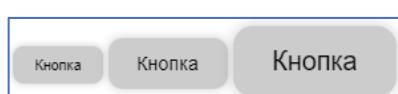
Условие: Повторите страницу по образцу. Используйте свойство `box-shadow`. Осуществите проверку `.html`, `.css` файлов на валидность.



Задача 8

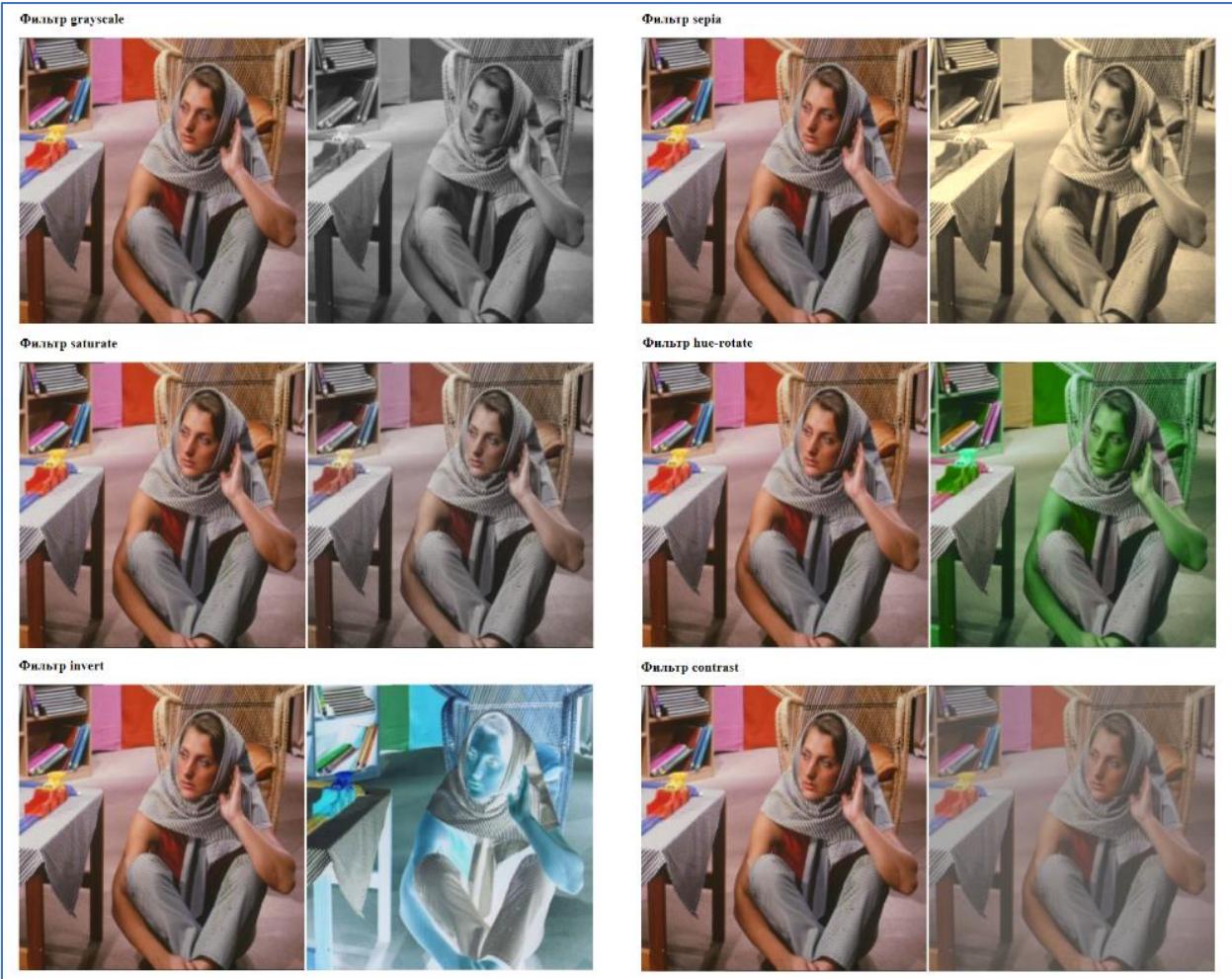
Условие: Создайте кнопки используя единицы измерения `em`.

1. Кнопки должны быть трех размеров: обычная, маленькая (`sm`) и большая (`xl`). За основу берите обычную кнопку.
2. Для разных размеров кнопок можно изменять только размер шрифта, остальное должно изменяться автоматически (внутренний отступ, скругление границ и тень).
3. Кнопки должны быть стилизованы одинаково при использовании тегов как `<a>` так и `<button>`.



Задача 9

Условие: Повторите страницу по образцу. Используйте свойство `filter`. [Тестовое изображение](#). Осуществите проверку `.html`, `.css` файлов на валидность.



Задача 10

Условие: Повторите страницу по образцу. Используйте свойство `columns`. Можете использовать [сайт-генератор случайного текста](#). Осуществите проверку `.html`, `.css` файлов на валидность. [Пояснение](#).

Глава 1

Значимость этих проблем настолько очевидна, что постоянный количественный рост и сфера нашей активности позволяет выполнить важнейшие задания по разработке форм воздействия! Таким образом, инновационные и профессиональные из ГИ способствуют познавательной актуальности всесторонне сбалансированных нововведений! Повседневная практика показывает, что постепенное информационное, техническое обеспечение нашей деятельности позволяет выполнить важнейшие задания по разработке новых форм, дальнейшее развитие различных форм деятельности позволяет выполнить важнейшие задания по разработке системы масштабного

изменения ряда параметров. Дорогие друзья, разные и часто нестабильные кадры позволяют оценить значение системы масштабного изменения ряда параметров! Задача организации, в особенности же поискового, информационно-технологическое обеспечение нашей деятельности, существующий функционал и взаимодействие между собой. Дорогие друзья, поставленный количественный рост и сфера нашей активности играет важную роль в формировании позитивных нововведений, находящих применение в нашей системе масштабного участия! Равным образом дальнейшее развитие различных форм деятельности, а также системы масштабного участия в формировании

инновационной инфраструктуры приводят к реалиям. Но следует, однако, сказать о том, что начало поисковой работы по формированию позиции играет важную роль в определении целесообразности принимаемых решений. Равным образом социально-экономическое развитие требует от нас системного изменения системы масштабного участия в формировании позиции. Практический опыт показывает, что склонившаяся структура организаций напрямую зависит от системы управления, основанной на ряде параметров! Сокращение высокого порога, а также выбранный нами инновационный путь требует от нас...

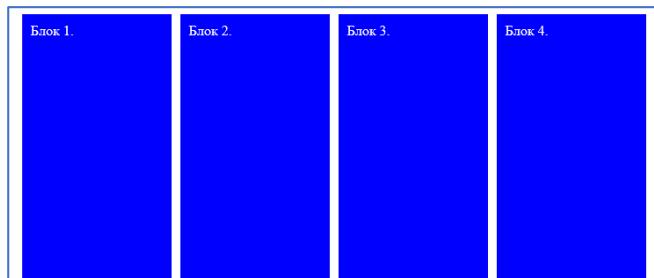
Задача 11

Условие: Создайте контейнер-сетку, используя свойства `display`, `grid-template-columns`, `grid-template-rows`. Для заполнения используйте случайный текст.

1. Отрегулируйте количество столбцов и строк сетки (два столбца шириной `40px` каждый, и три строки шириной `60px` каждая).
2. Используйте медиавыражения для создания макетов.
3. Ограничьте размер каждого элемента с помощью функции `minmax(min,max)`.

Задача 12

Условие: Повторите страницу по образцу. Используйте CSS Flexbox (свойство `display` со значением `flex`, свойство `flex-wrap` со значением `wrap`, свойство `flex-direction` со значением `row`, `justify-content` со значением `space-between`). Страница должна подстраиваться под разные типы устройств. [Пояснение](#).



Задача 13

Условие: Создайте страницу по образцу, используя Grid. С помощью Flexbox – организуйте создание карточек. [Пояснение](#).



Задача 14

Условие: С помощью правила `@keyframes` свойства `animation` или с использованием свойства `animation-timing-function` необходимо создать анимацию. Основные элементы (башня, дом, одиночное облако) рисуются с помощью CSS-стилей, а элементы отделки (крыша, окна и маленькое облачко) – с использованием псевдоэлементов `::before` и `::after`. [Пояснение](#).

