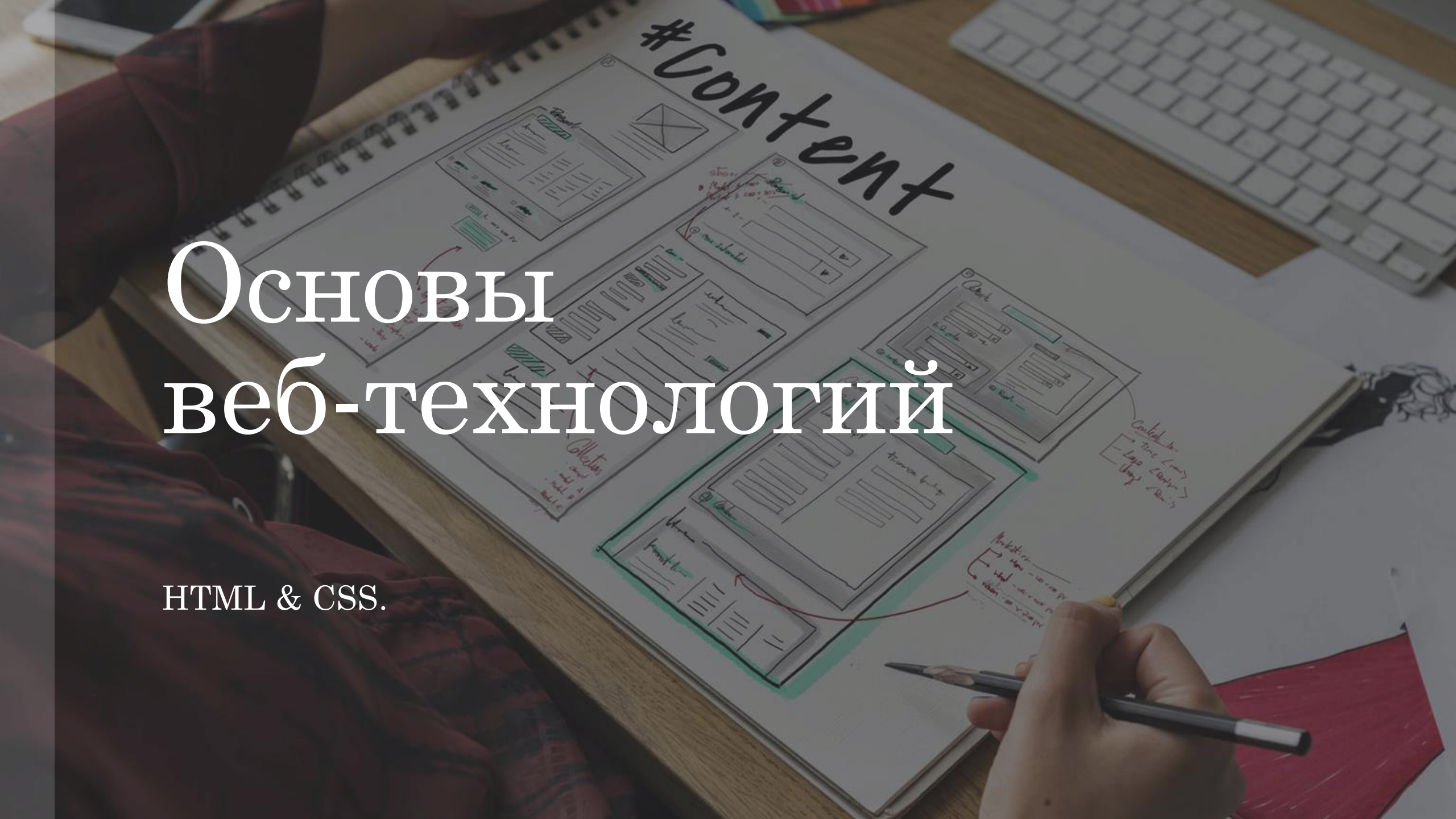


Основы веб-технологий

HTML & CSS.

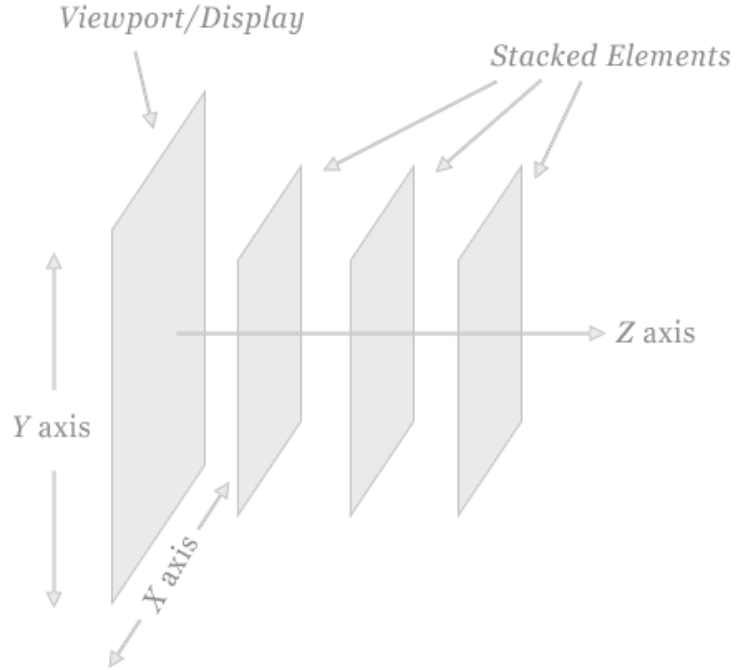
Определение контекста наложения: свойство z-index

В CSS каждый блок имеет позицию в трех измерениях. В дополнение к горизонтальному и вертикальному положению, блоки выкладываются вдоль оси Z друг над другом. Положение вдоль оси Z особенно важно, когда блоки визуально накладываются друг на друга.

Порядок, в котором дерево документа отрисовывается на экране, описывается с помощью контекста наложения. Каждый блок принадлежит одному контексту наложения. Каждый блок в данном контексте наложения имеет целочисленный уровень, который является его положением на оси Z относительно других блоков в том же контексте наложения.

Блоки с более высокими уровнями всегда отображаются перед блоками с более низкими уровнями, а блоки с одинаковым уровнем располагаются снизу вверх в соответствии с порядком следования элементов в исходном документе. Блок элемента имеет ту же позицию, что и блок его родителя, если только ему не присвоен другой уровень свойством z-index.

Свойство z-index позволяет изменить порядок наложения позиционированных элементов в случае, когда они накладываются друг на друга.



Контекст наложения

Если для элементов свойства z-index и position не заданы явно, контекст наложения равен порядку их расположения в исходном коде и браузер отображает элементы на странице в следующем порядке:

- Корневой элемент <html>, который содержит все элементы веб-странице.
- Блочные элементы, неплавающие и непозиционированные.
- Плавающие float непозиционированные элементы в порядке их расположения в исходном коде.
- Строковые непозиционированные элементы (текст, изображения).
- Позиционированные position элементы в порядке их следования в исходном коде. Последний из них будет расположен на переднем плане.

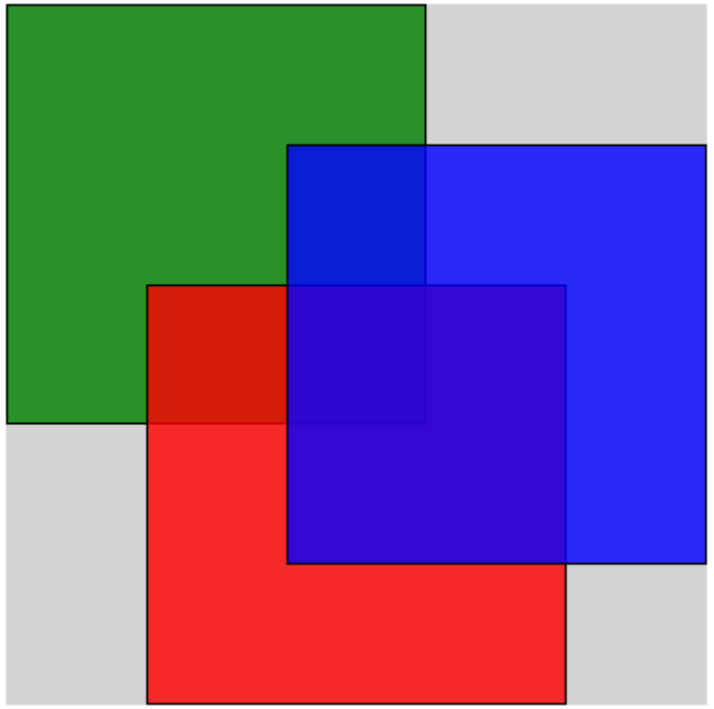
Свойство z-index создает новый контекст наложения. Оно позволяет изменить порядок наложения **позиционированных** элементов. Элементы будут отображаться на странице в следующем порядке (если для них не заданы свойства, влияющие на контекст наложения — opacity, filter, transform):

- Корневой элемент <html>, который содержит все элементы веб-странице.
- Позиционированные элементы с отрицательным значением z-index.
- Блочные элементы, неплавающие и непозиционированные.
- Плавающие float непозиционированные элементы в порядке их расположения в исходном коде.
- Строковые непозиционированные элементы (текст, изображения).
- Позиционированные элементы со значениями z-index: 0; и z-index: auto;.

Пример Z-index

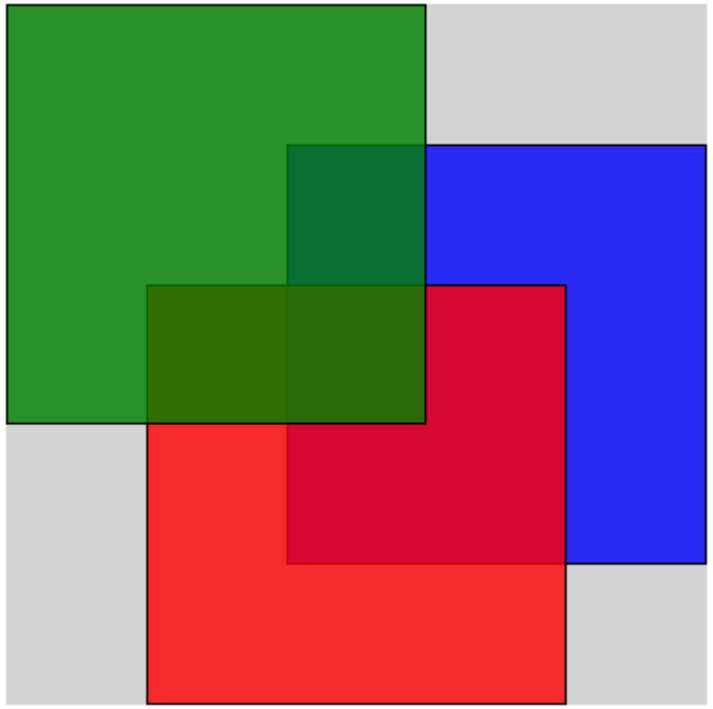
```html
<div class="container">
    <div class="block block-1"></div>
    <div class="block block-2"></div>
    <div class="block block-3"></div>
</div>
```

```css
.container{
    background-color: lightgrey;
    width: 200px;
    height: 200px;
    margin: 0 auto;
    position: relative;
}
.block{
    width: 120px;
    height: 120px;
    border: 1px solid black;
    box-sizing: border-box;
    position: absolute;
}
.block-1{
    background-color: rgba(0, 128, 0, 0.8); /* зеленый */
    left: 0px;
    top: 0px;
}
.block-2{
    background-color: rgba(255, 0, 0, 0.8); /* красный */
    left: 40px;
    bottom: 0px;
}
.block-3{
    background-color: rgba(0, 0, 255, 0.8); /* синий */
    right: 0px;
    bottom: 40px;
}
```



Пример Z-index

```
.container{
    background-color: lightgrey;
    width: 200px;
    height: 200px;
    margin: 0 auto;
    position: relative;
}
.block{
    width: 120px;
    height: 120px;
    border: 1px solid black;
    box-sizing: border-box;
    position: absolute;
}
.block-1{
    background-color: rgba(0, 128, 0, 0.8);
    left: 0px;
    top: 0px;
    z-index: 700;
}
.block-2{
    background-color: rgba(255, 0, 0, 0.8);
    left: 40px;
    bottom: 0px;
    z-index: 300;
}
.block-3{
    background-color: rgba(0, 0, 255, 0.8);
    right: 0px;
    bottom: 40px;
    z-index: 100;
}
```



Обтекание: свойство float

Изначально свойство float было создано для обтекания текстом картинок. Однако достаточно быстро его стали использовать для создания макетов из нескольких столбцов на веб-страницах. С появлением flexbox и grid свойство float снова рекомендуется использовать только как и задумывалось изначально, но в сети достаточно часто можно встретить подготовленные ранее макеты, использующие float, не смотря на то, что это считается устаревшей техникой.

При использовании свойства float для элементов рекомендуется задавать ширину. Тем самым браузер создаст место для другого содержимого. Если для плавающего элемента недостаточно места по горизонтали, он будет смещаться вниз до тех пор, пока не уместится. При этом остальные элементы уровня блока будут его игнорировать, а элементы уровня строки будут смещаться вправо или влево, освобождая для него пространство и обтекая его.

Правила, регулирующие поведение плавающих боков, описываются свойством `float: left | right;`

Свойство автоматически изменяет вычисляемое (отображаемое в браузере) значение свойства `display` на `display: block`. Не применяется к абсолютно позиционированным элементам.

```
<main>
    <h1>Енот</h1>
    <div class="float"><img src="racoons.jpg"></div>
    <p>Енот-полоскун, или американский енот — хищное млекопитающее рода еноты семейства енотовых.</p>
    <p>Енот-полоскун ростом с кошку. Длина тела 65—80 см, хвоста 20—25 см; масса 5—9 кг (к зимней спячке ввиду накопления жировой прослойки). </p>
    <p>Лапы короткие, с настолько развитыми пальцами, что следы похожи на отпечаток человеческой ладони. Енот может передними лапами захватывать и удерживать предметы, в том числе и мыть еду. Высокая чувствительность лап заменяет дальнозоркому еноту зрение вблизи. Мех у енота густой, коричневато-серый. </p>
</main>
```

```
.float{
    width: 100px;
    float: left;
    margin: 0 10px 10px 10px;
    box-shadow: 2px 2px 2px grey;
}
.float img{
    width: 100%;
}
```

Очистка блока

Попробуем сделать так, чтобы картинку обтекал только первый абзац. Добавим фон для первого абзаца для наглядности.

```
<p class="bg-1">Енот-полоскун, или американский енот — хищное млекопитающее рода еноты семейства енотовых.</p>
```

Енот-полоскун, или американский енот — хищное млекопитающее рода еноты семейства енотовых.

Енот-полоскун ростом с кошку. Длина тела 65—80 см, хвоста 20—25 см; масса 5—9 кг (к зимней спячке ввиду накопления жировой прослойки).

Обтекаемый объект удалён из нормального потока и другие элементы располагаются за ним, поэтому если мы хотим остановить перемещение следующего элемента нам необходимо очистить его, что достигается при помощи свойства clear: left | right | both;
Мы можем добавить очистку слева для следующего абзаца.

```
<p style="clear: left;">Енот-полоскун ростом с кошку. Длина тела 65–80 см, хвоста 20–25 см; масса 5–9 кг (к зимней спячке ввиду накопления жировой прослойки). </p>
```

Енот-полоскун, или американский енот — хищное млекопитающее рода еноты семейства енотовых.

Енот-полоскун ростом с кошку. Длина тела 65—80 см, хвоста 20—25 см; масса 5—9 кг (к зимней спячке ввиду накопления жировой прослойки).

Очистка блока

Попробуем сделать так, чтобы картинку обтекал только первый абзац. Добавим фон для первого абзаца для наглядности.

```
<p class="bg-1">Енот-полоскун, или американский енот — хищное
млекопитающее рода еноты семейства енотовых.</p>
```

Енот-полоскун, или американский енот — хищное млекопитающее рода еноты семейства енотовых.

Енот-полоскун ростом с кошку. Длина тела 65—80 см, хвоста 20—25 см; масса 5—9 кг (к зимней спячке ввиду накопления жировой прослойки).

Обтекаемый объект удалён из нормального потока и другие элементы располагаются за ним, поэтому если мы хотим остановить перемещение следующего элемента нам необходимо очистить его, что достигается при помощи свойства clear: left | right | both | none;
Свойство `clear` указывает, какие стороны блока/блоков элемента не должны прилегать к плавающим блокам, находящемся выше в исходном документе.

```
<p style="clear: left;">Енот-полоскун ростом с кошку. Длина
тела 65–80 см, хвоста 20–25 см; масса 5–9 кг (к зимней спячке
ввиду накопления жировой прослойки). </p>
```

Енот-полоскун, или американский енот — хищное млекопитающее рода еноты семейства енотовых.

Енот-полоскун ростом с кошку. Длина тела 65—80 см, хвоста 20—25 см; масса 5—9 кг (к зимней спячке ввиду накопления жировой прослойки).

Очистка блока. Clearfix

Как видно из примера, если у вас имеется высокий обтекаемый объект и короткий обтекающий блок, то эффект получается не совсем корректный. Для корректного решения необходимо, чтобы и обтекаемый объект и обтекающие блоки были размещены в отдельный блок, к которому применяются очищающие стили.

```
<div class="wrapper clearfix">
    <div class="float"><img src="racoons.jpg"></div>
    <p >Енот-полоскун, или американский енот – хищное млекопитающее рода еноты семейства енотовых.</p>
</div>
```

```
.wrapper{
    background-color: lightblue;
}
.clearfix::after{
    content: "";
    display: block;
    clear: both;
}
```

Енот-полоскун, или американский енот — хищное млекопитающее рода еноты семейства енотовых.

Енот-полоскун ростом с кошку. Длина тела 65—80 см, хвоста 20—25 см; масса 5—9 кг (к зимней спячке ввиду накопления жировой прослойки).

Переполнение блока

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 предел

```
<div class="word1">Overflow</div>
<div class="word2"> Переполнение — это то, что случается, когда
у вас слишком много контента в блоке, так что он не помещается в
данный ограниченный размеры одного из его краев, т.е. края области
содержимого, поля, границы или отступа.
</div>
```

```
.word1 {
  border: 1px solid #333333;
  width: 100px;
  font-size: 250%;
}
.word2 {
  margin: 20px;
  border: 1px solid #333333;
  width: 200px;
  height: 100px;
}
```

Свойство overflow

Свойство overflow позволяет взять под контроль переполнение элемента и подсказать браузеру, как он должен себя вести. Значение overflow по умолчанию – visible, что означает - «показывать контент, когда он выходит за границы блока».

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.

visible	Значение по умолчанию. Содержимое не обрезается, а отображается поверх границ блока-контейнера. Возможно перекрытие соседних блоков.
hidden	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.

Автоматическое многоточие

Свойство `text-overflow` 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.

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.

Макет таблицы

CSS-свойство `display: table; display: table-row; и display: table-cell;` делают вывод группы элементов подобно таблице `<table>`, но с ограничением – объединения ячеек `colspan и rowspan` не поддерживаются.

Значение `display: table`; задает начальный тип представления, а строки и ячейки можно имитировать при помощи значений `display: table-row; и display: table-cell`;.

display также может принимать значения `table-column, table-row-group, table-column-group, table-header-group, table-footer-group и table-caption`, которые говорят сами за себя.

Значение `inline-table` по существу устанавливает "строчную" таблицу, т.е. без перевода строки перед и после нее.

Тем не менее, чрезмерное увлечение CSS таблицами может самым серьезным образом разрушить доступность вашего сайта. HTML должен использоваться для определения значения контента, а значит если у вас есть табличные данные, то и размещаться они должны в HTML таблицах.

Часто табличная верстка используется в качестве сеточной разметки, хотя это семантически неправильно и возникнут проблемы у мобильной или адаптивной версии. Для сеток специально ввели CSS Grid Layout.

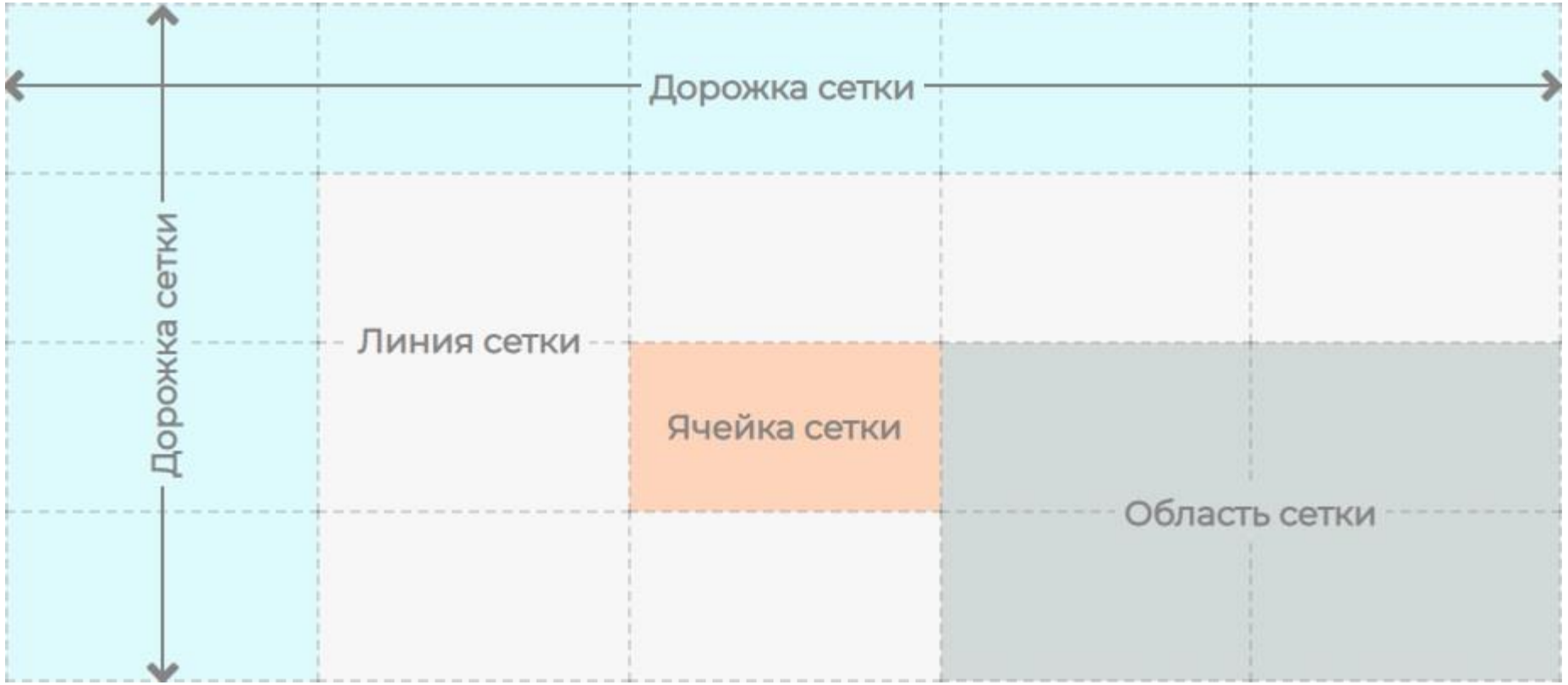
Grid и Flexible Box Layout

CSS Grid Layout это система 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.

CSS flexbox (Flexible Box Layout Module) — модуль макета гибкого контейнера — представляет собой способ компоновки элементов, в основе лежит идея оси. Flexbox состоит из гибкого контейнера (flex container) и гибких элементов (flex items). Гибкие элементы могут выстраиваться в строку или столбик, а оставшееся свободное пространство распределяется между ними различными способами. Flexbox решает специфические задачи — создание одномерных макетов, например, навигационной панели, так как flex-элементы можно размещать только по одной из осей.

Хотя многие макеты могут быть отображены с помощью Grid или Flexbox, у каждого есть свои особенности. Grid обеспечивает двухмерное выравнивание, использует нисходящий 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

Grid Layout: Концепция сетки



Создание контейнера-сетки

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.

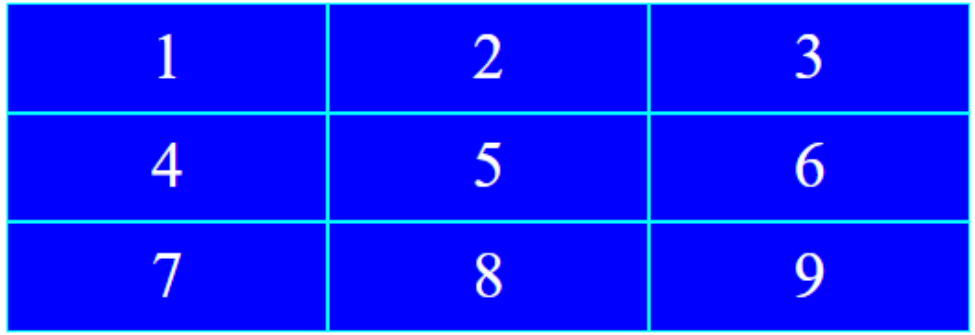
Контейнер-сетка бывает двух видов: обычный display: grid и встроенный display: inline-grid. Первый генерирует grid-контейнер уровня блока, второй — grid-контейнер уровня строки. Контейнеры-сетки не являются блочными контейнерами, поэтому некоторые CSS-свойства не работают в контексте макета сетки:

- float и clear игнорируются элементами сетки (но не самим контейнером-сеткой).
- vertical-align не влияет на элементы сетки.
- 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.

Создание контейнера-сетки

```html
<div class="grid-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
    <div>7</div>
    <div>8</div>
    <div>9</div>
</div>
```

```css
.grid-container div{
    background-color: blue;
    color: white;
    border: 1px solid cyan;
    padding: 5px;
    font-size: 20px;
    text-align: center;
}
.grid-container{
    display: grid;
    grid-template-columns: 100px 100px 100px;
}
```



Создание контейнера-сетки: размеры дорожек

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.

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: fr — единица длины, которая позволяет создавать гибкие дорожки (fraction). 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%; */
}
```

Создание контейнера-сетки: 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.

Иногда нужно задать минимальный размер строке, чтобы при заполнении она растягивалась под контент. С этим легко справляется функция `minmax()`, которая принимает минимальный и максимальный размер:

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

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`.

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

Создание контейнера-сетки: 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. Первым параметром функция **`repeat()`** принимает количество, а вторым — ширину/высоту (или список ширин/высот).

```
grid-template-columns: repeat(3, 1fr); /* 1fr 1fr 1fr */

grid-template-columns: repeat(3, 1fr 2fr);  /* 1fr 2fr 1fr 2fr 1fr 2fr */
grid-template-rows: repeat(2, auto);
```

Создание контейнера-сетки: неявные дорожки

Окончательная сетка может оказаться больше из-за элементов сетки, размещенных вне явной сетки; в этом случае будут созданы неявные дорожки.

Явные дорожки указываются при помощи свойств `grid-template-columns` и `grid-template-rows`, когда мы явно указываем количество строк и столбцов. Размеры неявных колонок и строк будут рассчитаны автоматически, в зависимости от контента внутри них. Размер неявных дорожек можно определить свойствами `grid-auto-rows` и `grid-auto-columns`

```
.grid-container {
   display: grid;
   grid-template-columns: repeat(2, 1fr);
   grid-template-rows: repeat(2, 100px);
   grid-auto-rows: 30px;
}
```

Создание контейнера-сетки: адаптивная верстка

Grid предоставляет нам возможности создавать простые адаптивные раскладки без применения медиа выражений. Функция `repeat()` первым аргументом может принимать не только количество, но и ключевые слова – `auto-fill / auto-fit`.

Следующее определение grid-template-columns поможет в ширину экрана вместить максимальное количество ячеек сетки, у которых есть минимальный возможный размер, при этом содержимое ячеек будет растягиваться на всю доступную ширину и выстраиваться по сетке:

```
grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
```

Ключевое слово auto-fill автоматически заполнит сетку колонками, ширина которых будет минимум 200px. Если есть свободное место, колонки будут растягиваться в ширину до тех пор, пока нет места для ещё одной колонки в 200px.
Если ширины и количества элементов не хватает, чтоб заполнить весь экран, начинают работать свойства auto-fill / auto-fit.

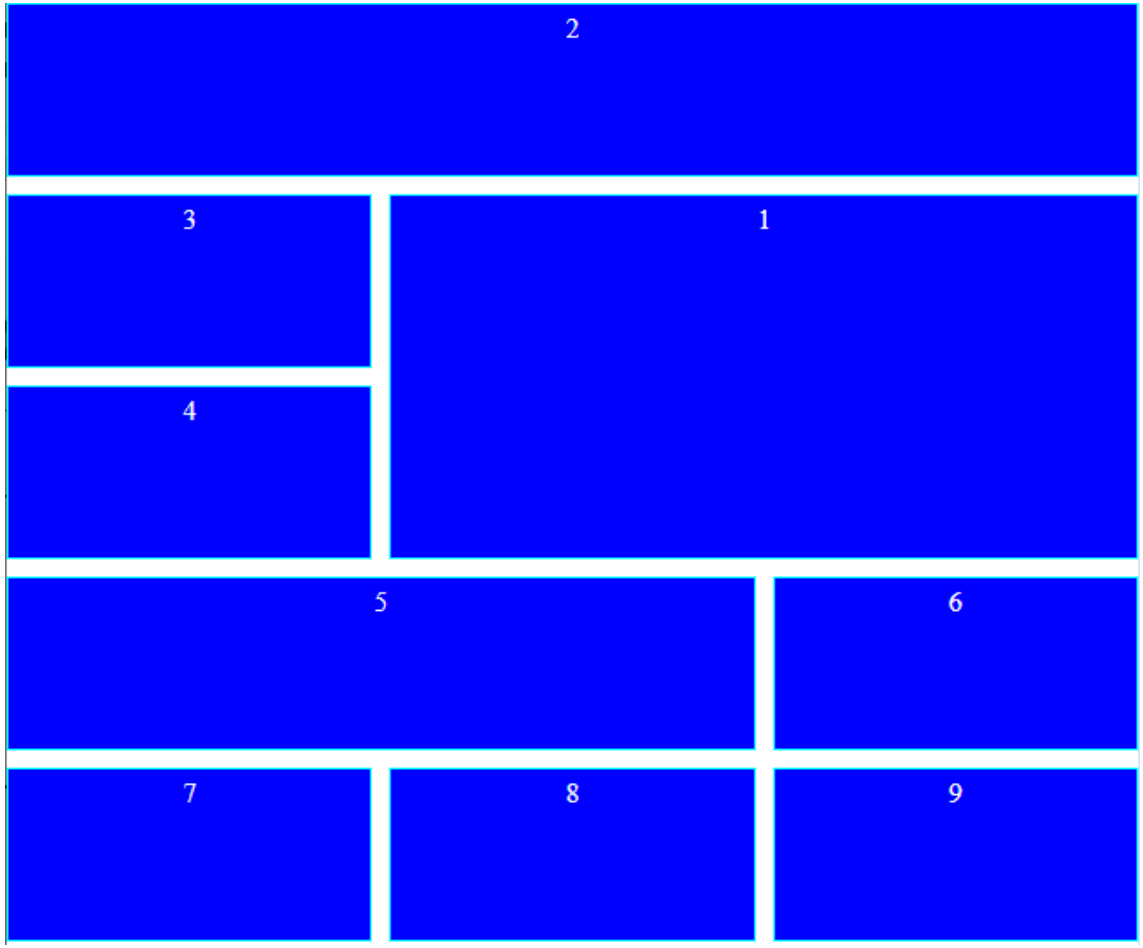
```
.grid-container{
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  grid-auto-rows: minmax(100px, auto);
}
```

Линии сетки и размещение элементов

В devtools в закладке «Макет» можно включить просмотр номеров линий сетки. Можно легко размещать элементы в Grid-сетке относительно нумерованных линий. Для этого существуют следующие свойства:

```
grid-column-start, grid-column-end
grid-row-start, grid-row-end
grid-column
grid-row
```

```css
.grid-container{
    display: grid;
    gap: 10px 10px;
    grid-template-columns: repeat(3, 1fr);
    grid-template-rows: minmax(100px, auto);
    grid-auto-rows: minmax(100px, auto);
}

.grid-container div:nth-child(1){
    grid-column: 2/-1;
    grid-row: 2/4;
}

.grid-container div:nth-child(2){
    grid-column: 1/4;
    grid-row: 1;
}
.grid-container div:nth-child(5){
    grid-column: 1/3;
    grid-row: 4;
}
```



Линии сетки и размещение элементов

Ключевое слово span позволит растянуть элемент на нужное количество ячеек:

```
.grid-container div:nth-child(2){
    grid-column: 1/ span 3;  /* Позволяет растянуть на 3 элемента, аналог 1/4 */
    grid-row: 1; /* короткая запись 1/2 */
}
```

Алгоритм размещения элементов:

1. В начале располагаются те элементы, которые были вырваны из обычного потока размещения. Чтобы вырвать элемент из этого потока, достаточно явно указать линию, от которой должен быть расположен элемент.
2. Далее все оставшиеся элементы ищут свободное место, и заполняют его слева направо, сверху вниз.

Именованные области: свойство grid-template-areas

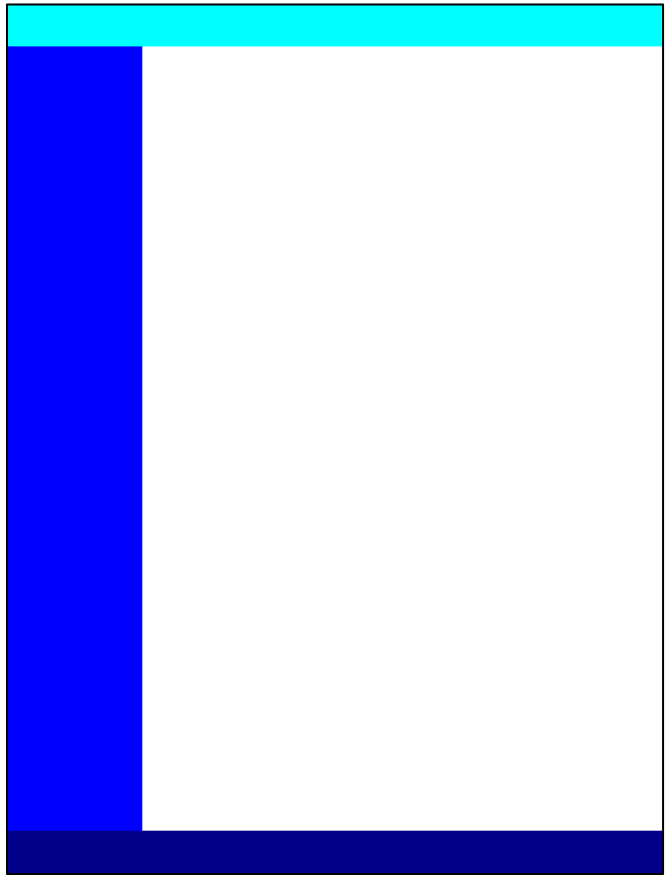
Свойство `grid-template-areas` определяет именованные области сетки, которые не связаны с каким-либо конкретным элементом сетки, но на которые можно ссылаться из свойств размещения сетки. Синтаксис свойства обеспечивает визуализацию структуры сетки, облегчая понимание общего макета контейнера-сетки.

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.

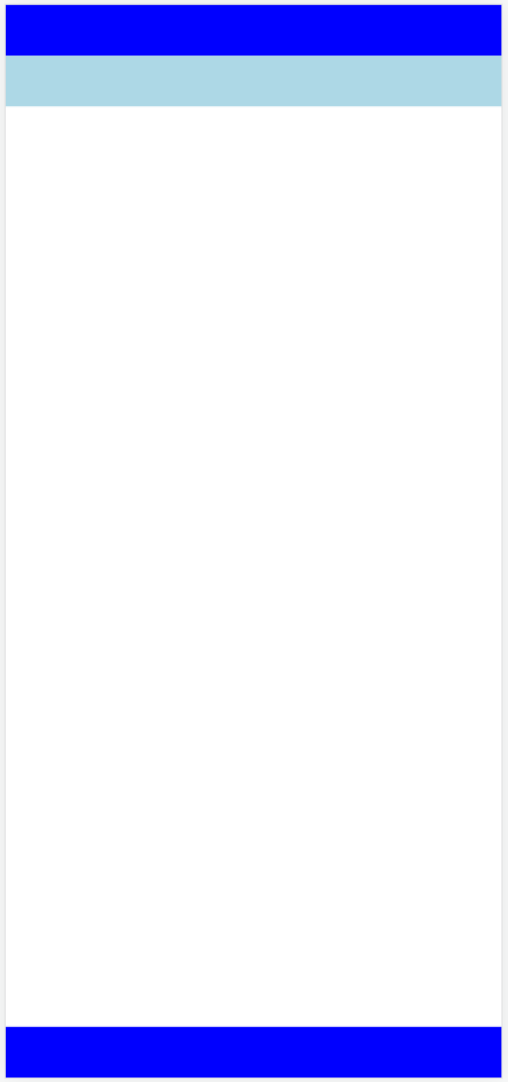
Именованные области: свойство grid-template-areas

```html
<body >
    <div class="main-layout">
        <header> </header>
        <aside> </aside>
        <main> </main>
        <footer> </footer>
    </div>
</body>
```



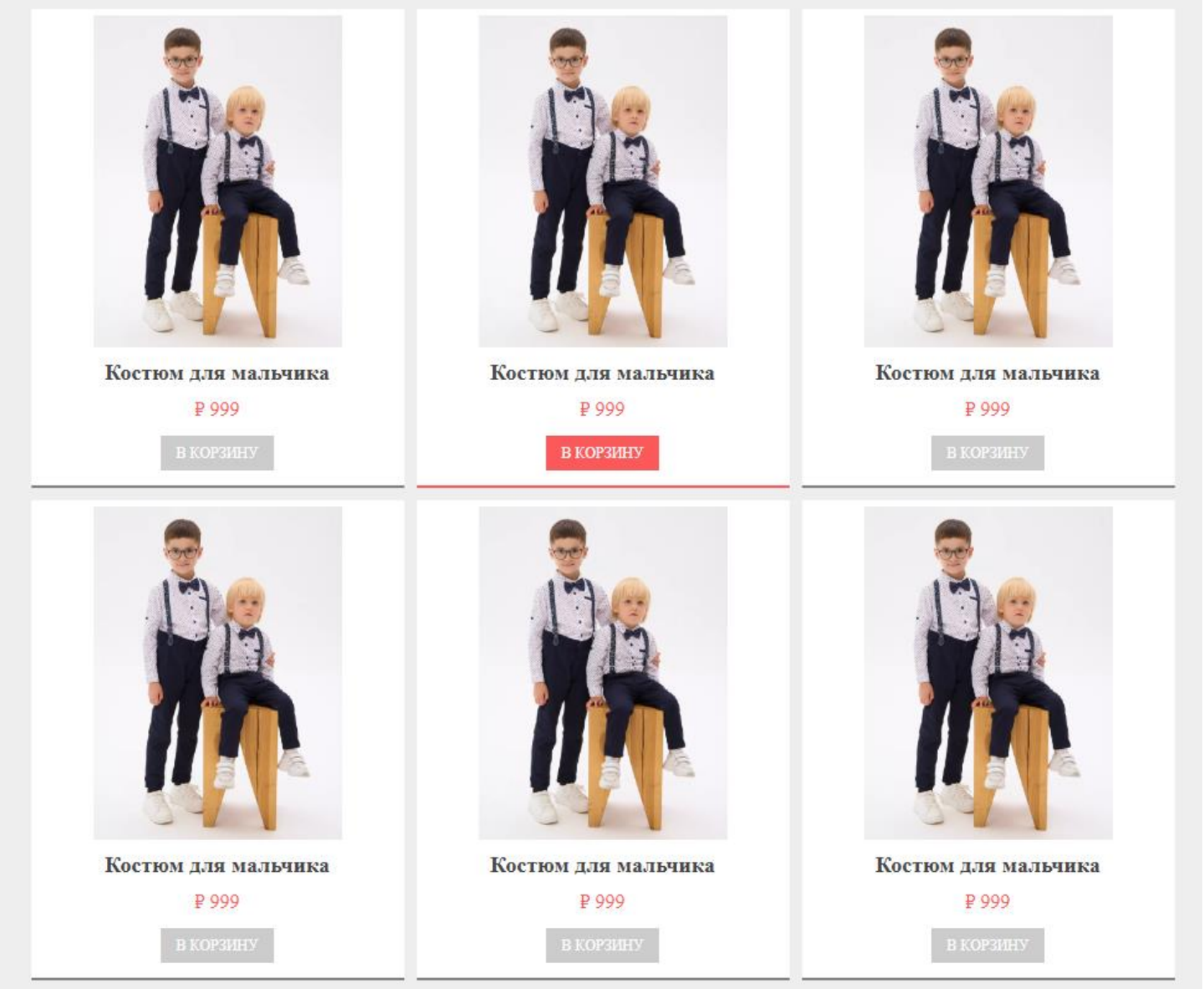
```css
html,
body {
    margin: 0;
}

.main-layout {
    display: grid;
    min-height: 100vh;
    max-width: 920px;
    margin: 0 auto;
    grid-template-areas:
        "header header"
        "sidebar content"
        "footer footer";
    grid-template-columns:  100px 1fr;
    grid-template-rows: 50px  minmax(200px, 1fr) 50px;
}

header {
    grid-area: header;
    background-color: cyan;
}
aside {
    grid-area: sidebar;
    background-color: blue;
}
main {
    grid-area: content;
    background-color: white;
}
footer {
    grid-area: footer;
    background-color: darkblue;
}
```

Версия разметки для телефона



```
.main-layout {
    display: grid;
    min-height: 100vh;
    margin: 0 auto;
    grid-template-areas:
    "header"
    "sidebar"
    "content"
    "footer";
    grid-template-columns: 1fr;
    grid-template-rows: 50px 50px 1fr 50px;
}
```

Пример Grid Layout для карточек



HTML

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Карточки</title>
    <link rel="stylesheet" href="grid-cards.css">
</head>
<body>
    <div class="card-list">
        <div class="card">
            <img src="foto.png">
            <div class="card-info">
                <h3>Костюм для мальчика</h3>
                <span class="price">₽ 999</span>
                <a href="#" class="button">В корзину</a>
            </div>
        </div>
        <div class="card">
            <img src="foto.png">
            <div class="card-info">
                <h3>Костюм для мальчика</h3>
                <span class="price">₽ 999</span>
                <a href="#" class="button">В корзину</a>
            </div>
        </div>
        .........
    </div>
</body>
</html>
```

CSS

```css
body{
    background-color: #eee;
}
.card-list {
    max-width: 920px;
    margin: 0 auto;
    padding: 10px 0;
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 10px;
}
.card {
    width: 100%;
    margin: 0 auto;
    background: white;
    text-align: center;
    border-bottom: 2px solid grey;
}
.card img {
    display: block;
    min-width: 150px;
    max-width: 200px;
    margin: 5px auto;
    width: 100%;
}
```

```css
.card h3 {
    font-size: 18px;
    font-weight: bold;
    color: #444444;
    margin: 10px 0;
}
.card .price {
    font-size: 16px;
    color: #fc5a5a;
    display: block;
    margin-bottom: 12px;
}
.card .button {
    text-decoration: none;
    display: inline-block;
    padding: 0 12px;
    background: #cccccc;
    color: white;
    text-transform: uppercase;
    font-size: 12px;
    line-height: 28px;
    margin-bottom: 12px;
    transition: .3s ease-in;
}
.card:hover .button{
    background: #fc5a5a;
}
.card:hover {
    border-bottom: 2px solid #fc5a5a;
}
```