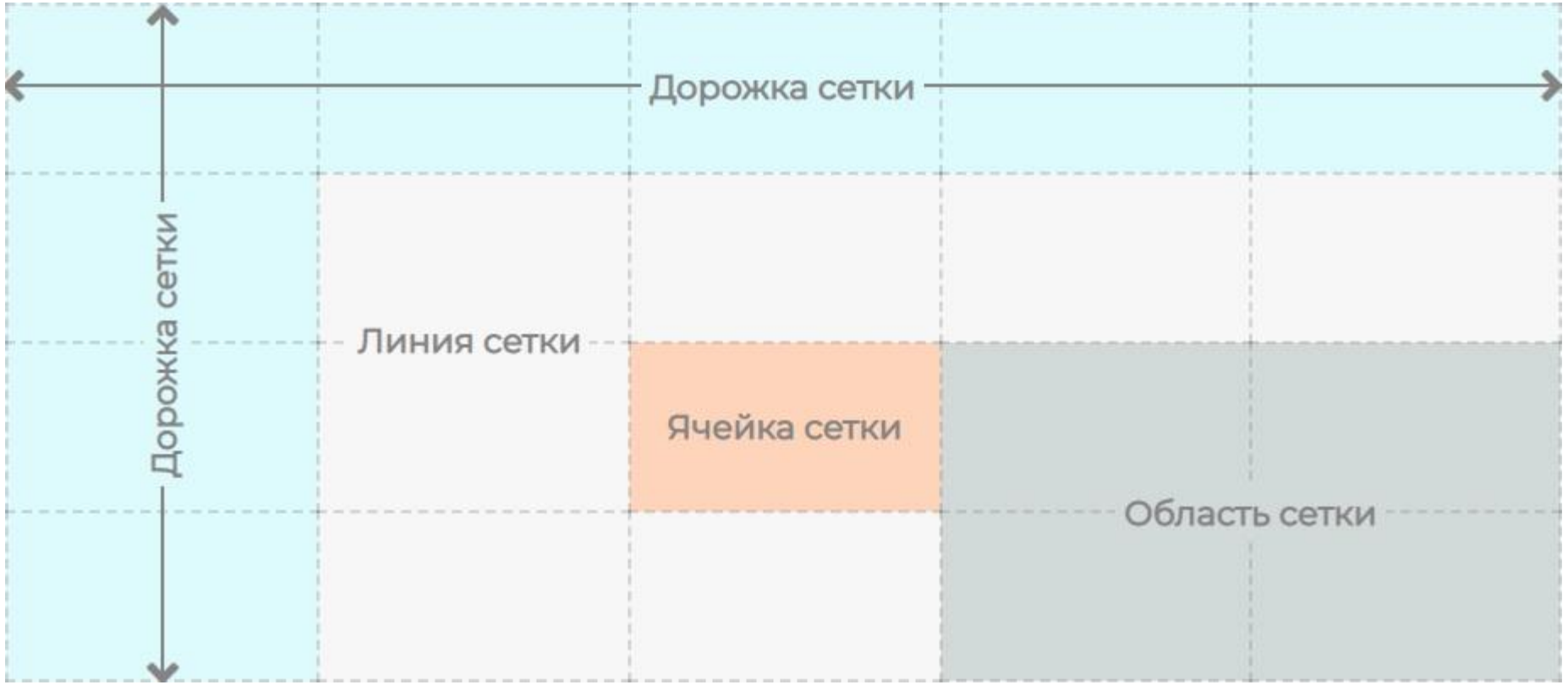
Grid и Flexible Box Layout

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.

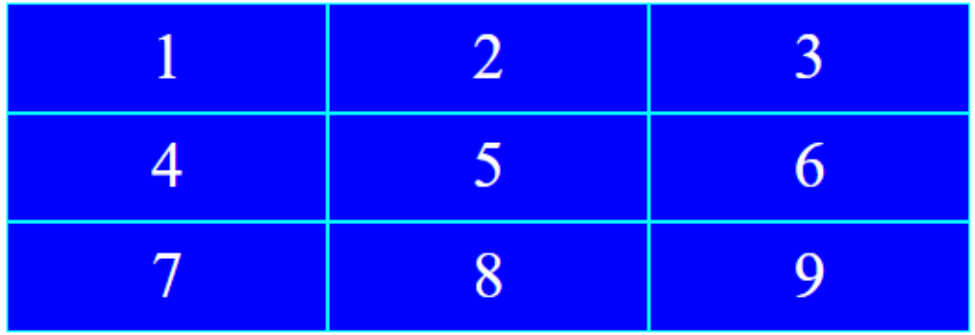
Контейнер-сетка бывает двух видов: обычный display: grid и встроенный display: inline-grid. Первый генерирует grid-контейнер уровня блока, второй — grid-контейнер уровня строки. Контейнеры-сетки не являются блочными контейнерами, поэтому некоторые CSS-свойства не работают в контексте макета сетки:

- 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.

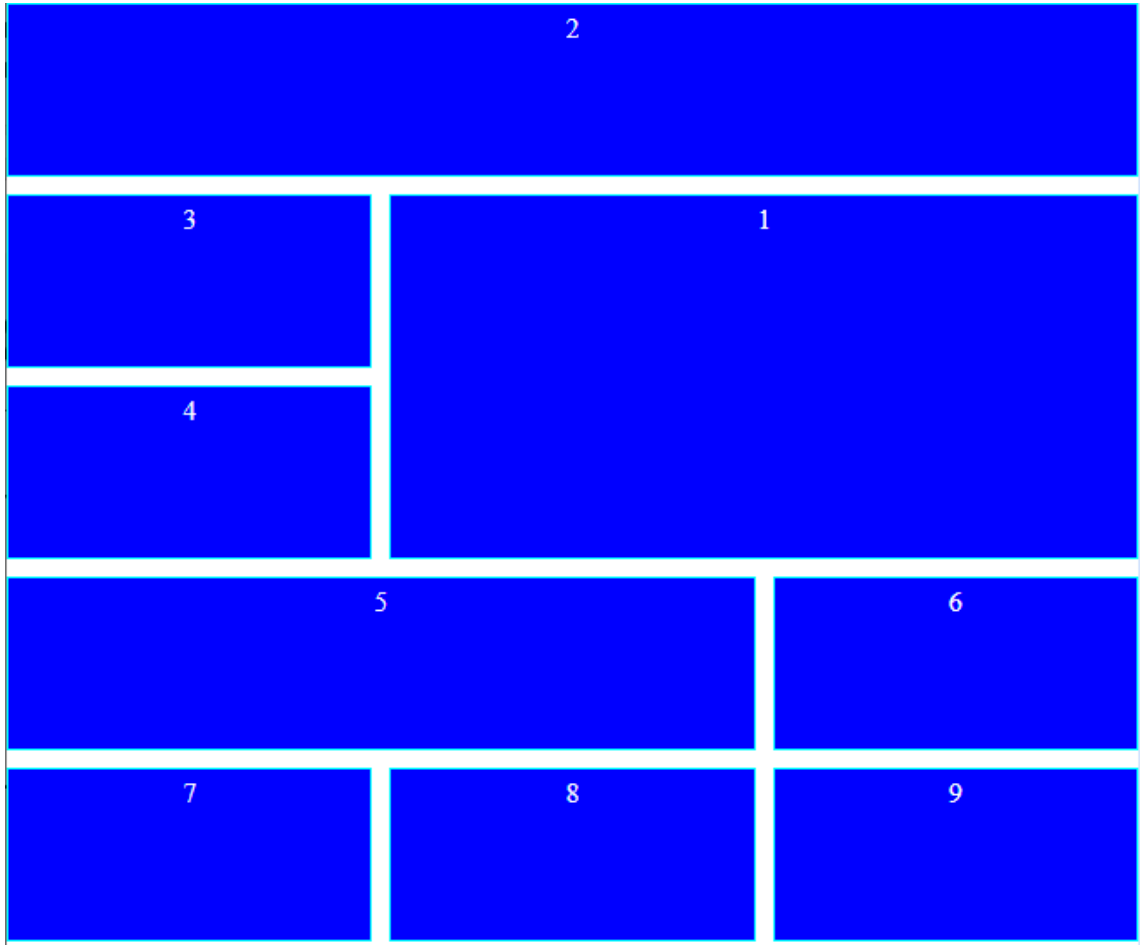
```
.grid-container{
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  grid-auto-rows: minmax(100px, auto);
}
```

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`

```
.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

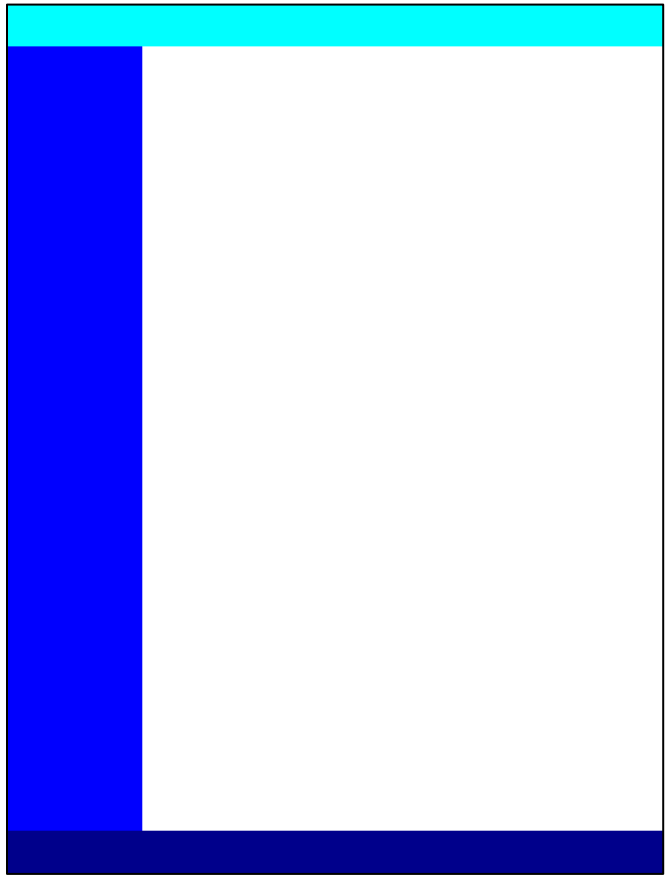
Свойство `grid-template-areas` определяет именованные области сетки, которые не связаны с каким-либо конкретным элементом сетки, но на которые можно ссылаться из свойств размещения сетки. Синтаксис свойства обеспечивает визуализацию структуры сетки, облегчая понимание общего макета контейнера-сетки.

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.

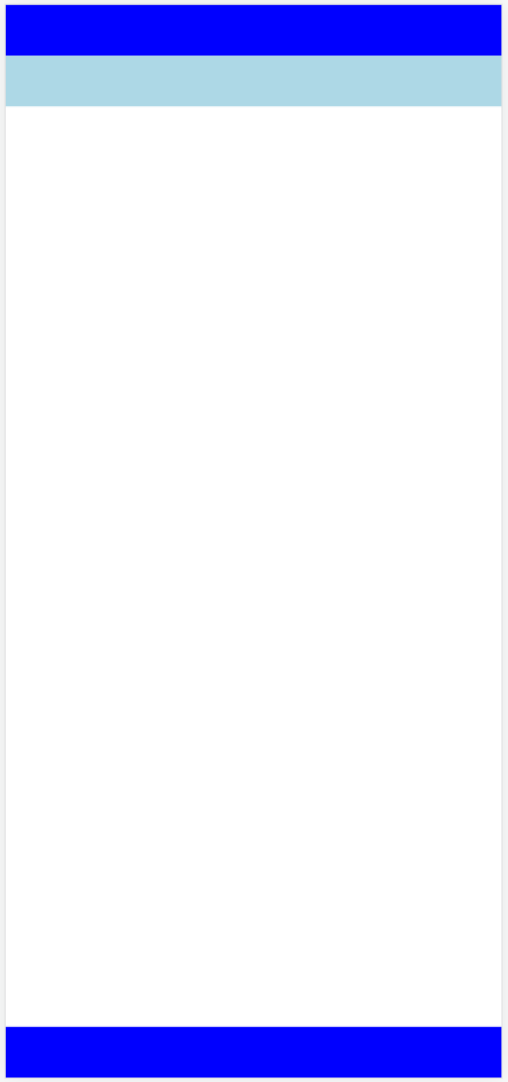
Именованные области: свойство grid-template-areas

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



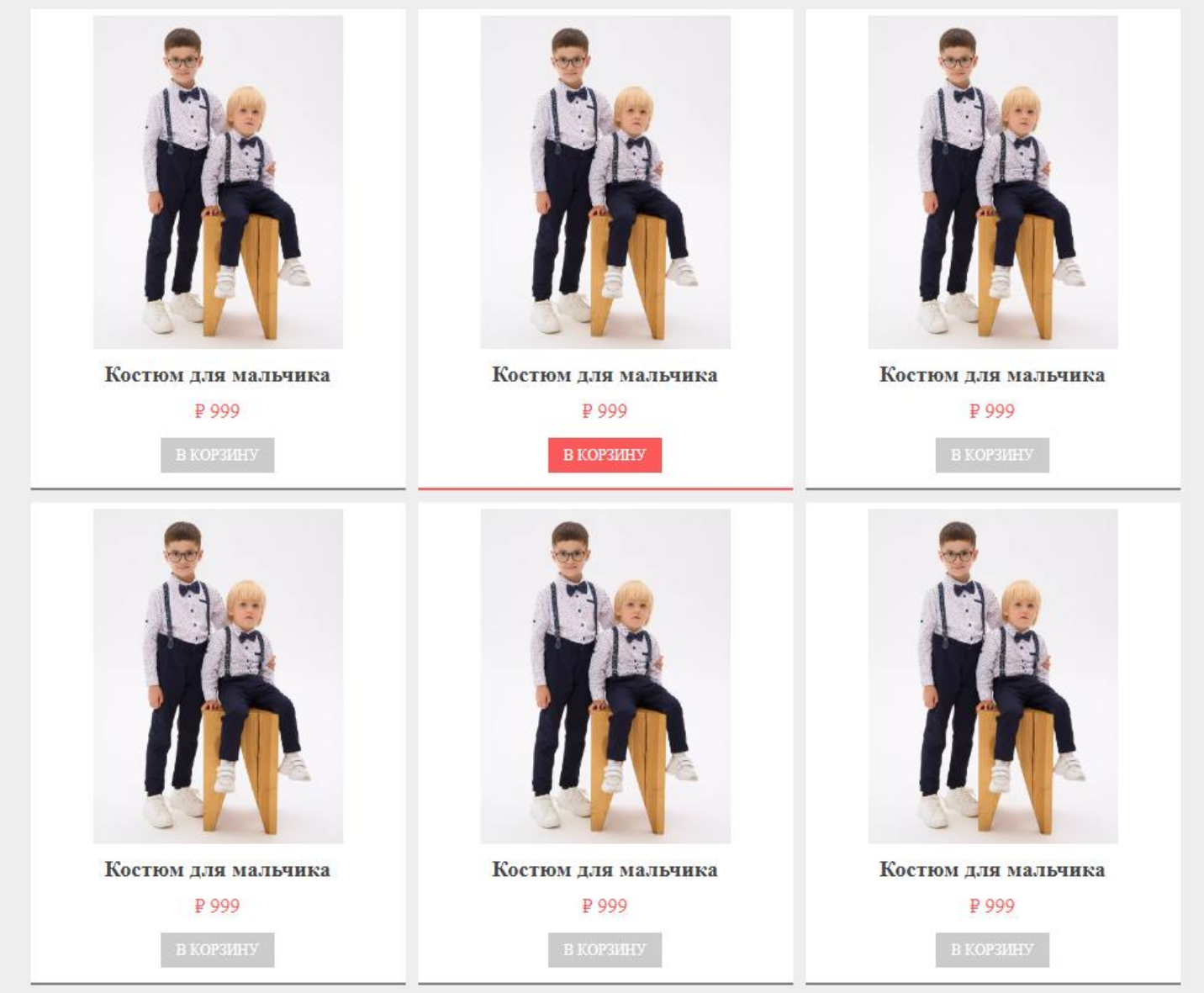
```css
html,
body {
    margin: 0;
}

.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: 100%;
    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;
}
```