Grid Layout

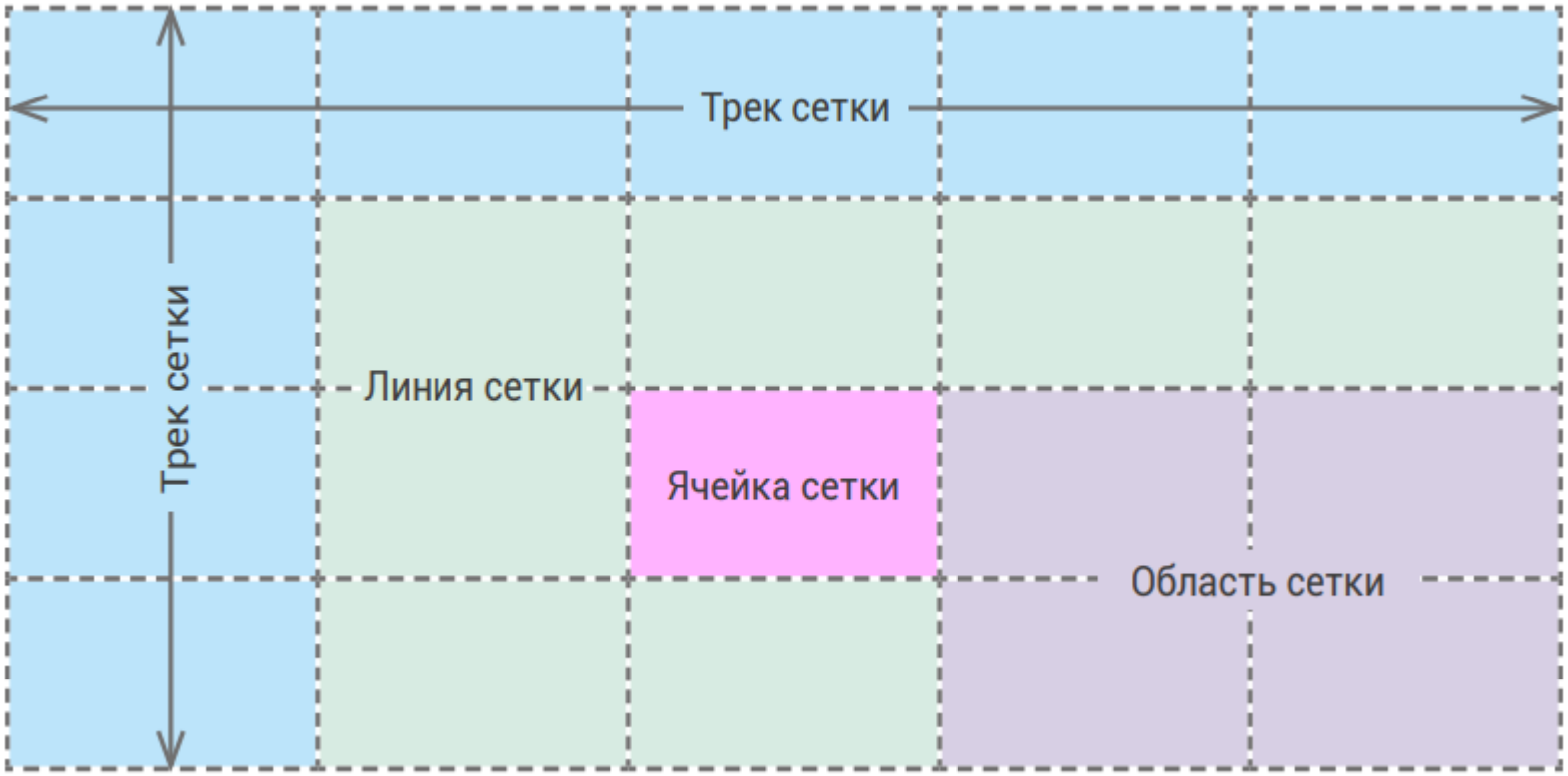
Модуль 8 (2 пары)

Знакомство с Grid Layout

- Появление CSS Grid Layout — это важное событие во Front-end разработке, призванное упростить построение веб-макетов. CSS уже давно используется для компоновки элементов на странице. Сначала веб-разработчики использовали таблицы, позже — плавающие элементы (свойство float), позиционирование и строчные блоки. Однако эти инструменты и средства не всегда гибко и качественно выполняли работу по структурированию страниц. С появлением модели Flexbox ситуация улучшилась, но эта модель рассчитана на одномерные макеты. Правильно будет сказать, что Flexbox выполняет свои задачи, а Grid — свои.

- **Grid Layout** — система двумерной компоновки, используемая для создания дизайна пользовательского интерфейса. В основе макета лежит сетка, разделяющая страницу на строки и столбцы, и позволяющая управлять размещением, размерами и выравниванием дочерних элементов. Благодаря свойству явно размещать элементы в сетке, Grid позволяет кардинально преобразовывать структуру макета, не изменяя при этом разметку документа.

Основные понятия и компоненты сетки



- **Контейнер сетки (Grid Container)** — главный родительский элемент для всех элементов сетки.
- **Элемент сетки (Grid Item)** — дочерний элемент (прямой потомок) контейнера сетки.

Основные понятия и компоненты сетки

- **Линия сетки (Grid Line)** — линия, которая разделяет элементы сетки. Она может быть горизонтальной или вертикальной.
- **Трек сетки (Grid Track)** — пространство между двумя параллельными линиями сетки. Можно считать, что это строка или столбец сетки.
- **Ячейка сетки (Grid Cell)** — пространство между двумя соседними строками и двумя соседними столбцами линий сетки, аналогично ячейке в таблицы. Это область, в которую можно что-то поместить. Ячейка является наименьшей единицей сетки, на которую можно ссылаться при позиционировании элементов сетки.
- **Область сетки (Grid Area)** — любое пространство между четырьмя линиями сетки может состоять из любого количества ячеек. Область сетки может быть не больше одной ячейки или размером со все ячейки сетки.

Как создать Grid-контейнер?

- Для построения макета на основе сетки, сначала необходимо определить **контейнер сетки**. Контейнер сетки — это блок, который создает область с сеткой и устанавливает контекст форматирования по типу сетки, то есть дочерние элементы располагаются согласно правилам компоновки сетки, а не блочной компоновки. Контейнер сетки определяется при помощи display: grid или display: inline-grid. Как только мы это сделаем, все прямые дочерние элементы контейнеры автоматически станут элементами сетки.
- Рассмотрим небольшой пример. Создадим разметку:

```
<div class="grid-container">
        <div class="item-1">child-item-1</div>
        <div class="item-2">child-item-2</div>
        <div class="item-3">child-item-3</div>
</div>
```

- Зададим стили:

```
.grid-container{
            display: grid;
            margin: 15px 30px;
            border-radius: 5px;
            background-color: #0d233d;
            color: white;
            font-size: 20px;
            font-family: Arial, Helvetica, sans-serif;
        }
```

Как создать Grid-контейнер?

```
.grid-container>div{
            margin: 10px 0;
            padding: 0.3em;
            border-radius: 3px;
            font-size: 130%;
            font-weight: bold;
}
.item-1{
            background-color: pink;
}
.item-2{
            background-color: cadetblue;
}
.item-3{
            background-color: coral;
}
```

Для элемента с классом grid-container мы задали display: grid. Таким образом все три непосредственных дочерних элемента этого контейнера автоматически стали элементами сетки.

Как создать Grid-контейнер?

- Когда мы создаем контейнер сетки, сетка по умолчанию имеет один столбец и одну строку, которые занимают полный размер контейнера. Контейнеры сетки не являются блочными контейнерами, поэтому некоторые CSS-свойства в контексте контейнера сетки работать не будут. Свойство контейнера сетки display объявляет, что текущий элемент будет использоваться как контейнер сетки и устанавливает новый контекст форматирования для его дочерних элементов. Возможные значения:
- **grid** — генерирует контейнер сетки уровня блока.
- **inline-grid** — генерирует контейнер сетки уровня строки.
- **subgrid** — если контейнер также является элементом другой сетки (вложенный контейнер), можно указать, что параметры для текущей сетки следует взять из родительской.

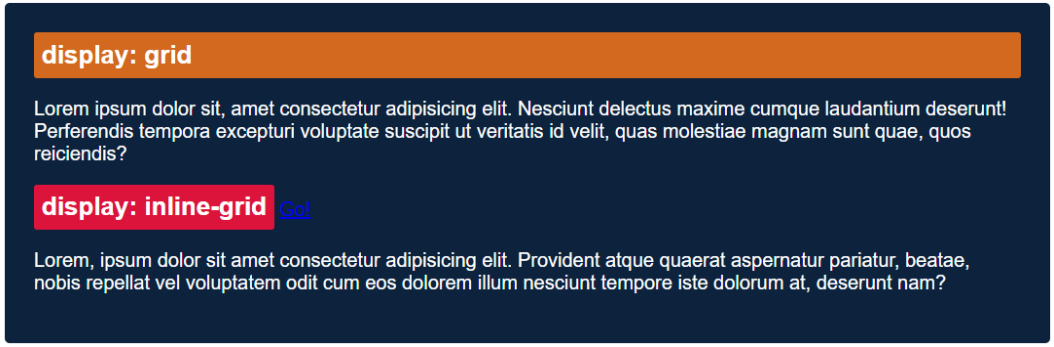
Рассмотрим пример:

```
<div class="container">
        <div class="grid-container">
            display: grid
        </div>
        <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Nesciunt d
electus maxime cumque laudantium deserunt! Perferendis tempora excepturi volupt
ate suscipit ut veritatis id velit, quas molestiae magnam sunt quae, quos reici
endis?</p>
        <div class="inline-grid-container">
            display: inline-grid
        </div>
        <a href="#">Go!</a>
        <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Provident
atque quaerat aspernatur pariatur, beatae, nobis repellat vel voluptatem odit c
um eos dolorem illum nesciunt tempore iste dolorum at, deserunt nam?</p>
    </div>
```

Как создать Grid-контейнер?

- Зададим для них стили:

```
.container{
            background-color: #0d233d;
            padding: 30px;
            font-size: 20px;
            font-family: Arial, Helvetica, sans-serif;
            border-radius: 5px;
            color: white;
        }
        .container>div{
            padding: 0.3em;
            font-size: 130%;
            font-weight: bold;
            border-radius: 3px;
        }
        .grid-container{
            display: grid;
            background-color: chocolate;
        }
        .inline-grid-container{
            display: inline-grid;
            background-color: crimson;
        }
```



Строка, столбец

- Количество строк и столбцов определяется с помощью свойств grid-template-rows и grid-template-columns.
- Например, создадим разметку:

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

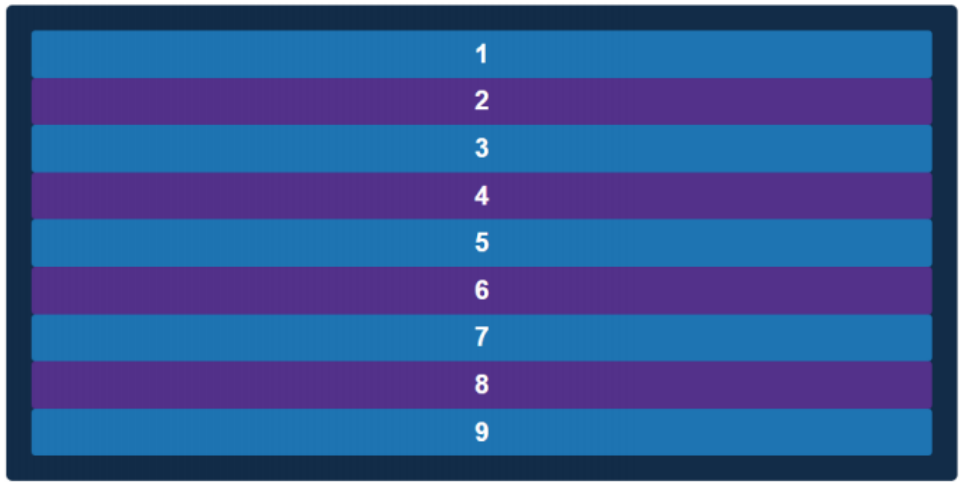
Строка, столбец

- Добавим стили:

```
.grid {
            display: grid;
            padding: 20px;
            font-size: 20px;
            font-family: Arial, Helvetica, sans-serif;
            color: white;
            background-color: #0d223d;
        }

        .grid>div {
            padding: 0.3em;
            border-radius: 3px;
            background-color: #482880;
            color: #ffffff;
            font-size: 130%;
            font-weight: bold;
            text-align: center;
        }

        .grid div:nth-child(odd) {
            background-color: #1769aa;
        }
```



Определение количества и размера столбцов

- Чтобы задать количество и размеры столбцов, используется свойство grid-template-columns:

```
.grid {
        display: grid;
                grid-template-columns: 100px 1200px 300px;
    }
```

- Результат:

- Мы создали 3 столбца и для каждого из них задали ширину 100px, 200px и 300px соответственно. Теперь элементы в сетке расположились в 3 строках и 3 столбцах.

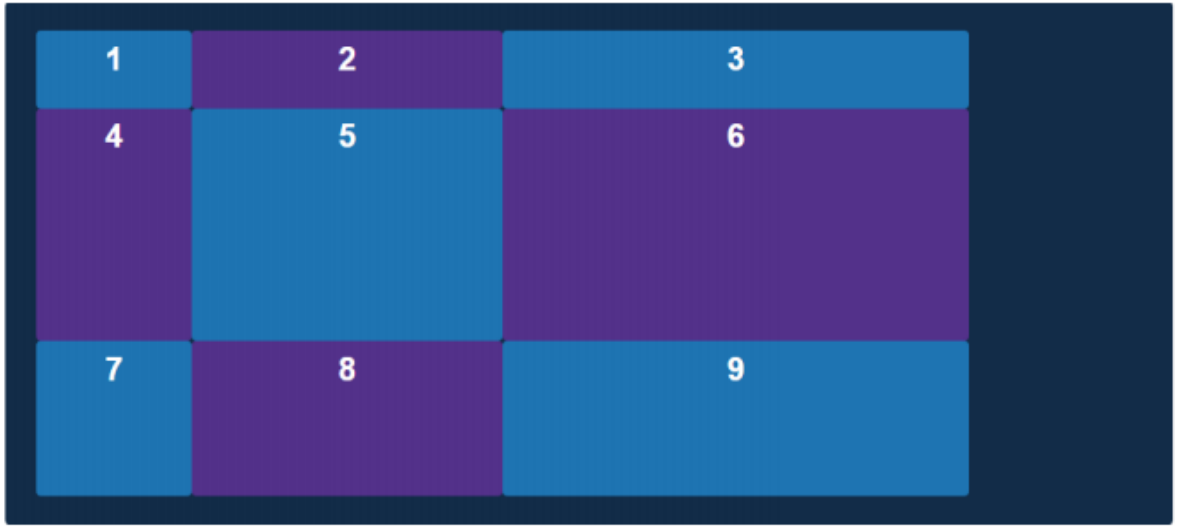
Определение количества и размера строк

- Чтобы задать количество и размеры столбцов, используется свойство grid-template-columns:

```
.grid {
          display: grid;
          grid-template-rows: 50px 150px 100px;

       }
```

- Результат:



- Для каждой строки мы задали высоту 50px 150px и 100px соответственно.

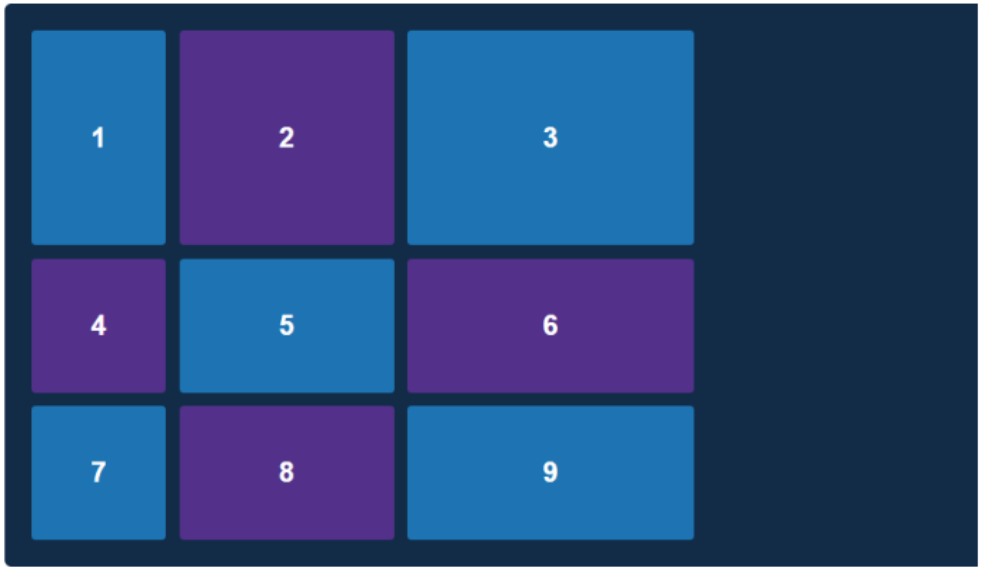
Значение для определения треков сетки

- Размеры треков сетки можно задавать с помощью различных значений, используя относительные единицы длины, например, em, vh, vw; абсолютные единицы длины px и проценты %. Размеры в % высчитываются по ширине или высоте контейнера-сетки. Зададим значение для размеров треков сетки в разных единицах измерения:

```
.grid {
          grid-template-columns: 100px 10em 30%;
          grid-template-rows:  10em 150px 100px;

       }
```

- Теперь первый столбец сетки занимает фиксированную ширину 100px, второй — 10em — ширина элемента будет пропорционально зависеть от размера шрифта, использованного для текста в блоке. В данном случае ширина элемента в 10 раз больше размера шрифта. Третий столбец занимает 30% от ширины контейнера сетки.
- Для строк заданы размеры 10em, 100px и 100px соответственно



Гибкие размеры треков: единица измерения fr

- Fr — единица измерения длины, которая позволяет создавать гибкие треки. При определении размеров треков, 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. Если сумма размеров всех гибких треков меньше 1, они будут занимать только соответствующую часть оставшегося пространства, а не расширяться, чтобы заполнить все пространство полностью.

- Если ширина или высота контейнера-сетки не заданы, треки сетки масштабируются по их содержимому, сохраняя при этом пропорции. Зададим значение для размеров треков в единицах fr:

```
.grid {
          display: grid;
          grid-template-columns: 2fr 1fr 1fr;
       }
```

-

Гибкие размеры треков: единица измерения fr

- Результат:

- В примере выше сетка разделена на 3 столбца. Первый столбец занимает половину ширины контейнера, а второй и третий распределили оставшееся пространство поровну между собой. Аналогичный результат получим, если зададим размеры для столбцов следующим образом: 50% 25% 25%.

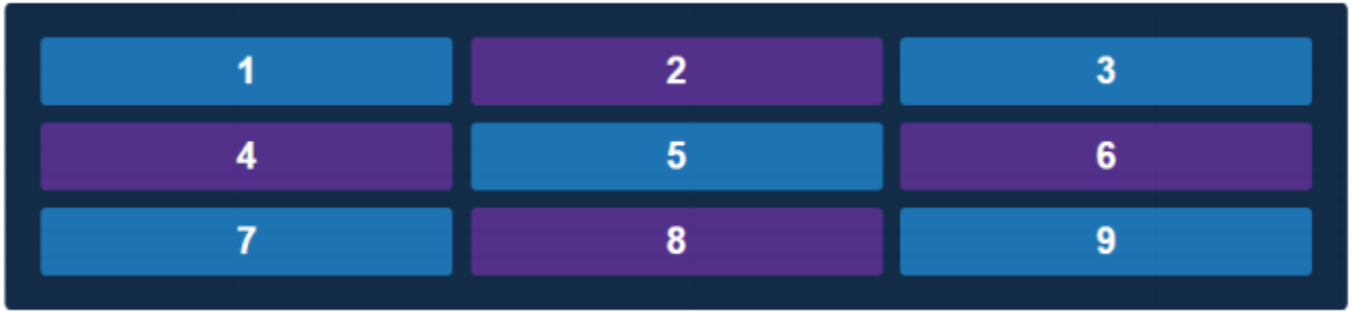
Автоматические размеры

- Значение auto ориентируется на содержание элементов сетки одного трека. Минимальный размер трека рассматривается как минимальный размер элемента сетки, то есть Grid использует свойства min-width или minheight. Максимальный размер трека устанавливается из расчета на значение свойства max-content. Трек может растягиваться при использовании свойств align-content и justify-content. Зададим размеры треков, используя значения auto:

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

        }
```

- Результат:



- Из примера видно, что каждая строка и столбец заняли поровну часть пространства контейнера сетки.

Функция repeat

- Функция repeat() позволяет создавать фрагменты списка треков, которые повторяются. Это позволяет записать в более компактной форме большое количество одинаковых по размерам столбцов или строк. Для примера создадим несколько столбцов с одинаковыми размерами:

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

- Такое написание можно заменить на:

```
.grid {
        display: grid;
        grid-template-columns: repeat(3, 1fr);
      }
```

- Рассмотрим еще один пример использования функции repeat():

```
.grid {
        display: grid;
        grid-template-columns: repeat(2, 1fr 2fr);
      }
```

Функция repeat

- Сейчас одна строка сетки состоит из столбцов, которые занимают ширину в 1fr и 2fr соответственно и повторяются 2 раза в пределах одной строки. Последний элемент сетки занимает ширину в 1fr. Если мы добавим еще один блок в сетку, он будет занимать 2fr, следующий — 1fr и т.д.
- Еще один пример:

```
.grid {
          display: grid;
          grid-template-columns: 50px repeat(2, 1fr 2fr);
       }
```

- Результат:

- Теперь первый элемент в строке занимает фиксированную ширину в 100px, а следующие 4 блока занимают ширину 1fr и 2fr соответственно и повторяются 2 раза в пределах строки.

Позиционирование элементов и Grid-линии

- Свойства размещения и компоновки позволяют свободно размещать и изменять порядок расположения элементов сетки таким образом, что визуальное представление может значительно отличаться от порядка элементов в html-документе.
- Позиционирование элементов сетки определяется расположением линий сетки и диапазоном сетки — количеством треков, которые занимает элемент. По умолчанию элемент сетки занимает одну ячейку трека на каждой оси.
- Свойства позиционирования элементов сетки — **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.
- При размещении элементов мы скорее отдаем предпочтение линиям, а не трекам, поскольку позиционирование в основном зависит от индексов начальных и конечных линий сетки, а не от порядковых номеров треков.

Позиционирование элементов сетки с помощью номеров линий сетки

- Можно определить расположение элементов, устанавливая номера начальной и конечной линий для элементов сетки.
- Свойства:
- **grid-column-start** — задает начальную линию расположения элемента в строке.
- **grid-column-end** — задает конечную линию расположения элемента в строке.
- **grid-row-start** — задает начальную линию расположения элемента в столбце.
- **grid-row-end** — задает конечную линию расположения элемента в столбце.

Позиционирование элементов сетки с помощью номеров линий сетки

- Например, создадим разметку:

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

- И зададим некоторые CSS-свойства для элементов:

```
.grid {
            display: grid;
            padding: 20px;
            font-size: 20px;
            font-family: Arial, Helvetica, sans-serif;
            color: white;
            background-color: #0d223d;
            grid-template-columns: repeat(3, 1fr);
            grid-template-rows: auto auto auto auto;
            gap: 10px;
        }
```

Позиционирование элементов сетки с помощью номеров линий сетки

```
.grid>div {
            padding: 0.3em;
            border-radius: 3px;
            background-color: #482880;
            color: #ffffff;
            font-size: 130%;
            font-weight: bold;
            text-align: center;
        }

        .grid div:nth-child(odd) {
            background-color: #1769aa;
        }

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

            grid-column-start: 3;
            grid-column-end: 4;
        }
```

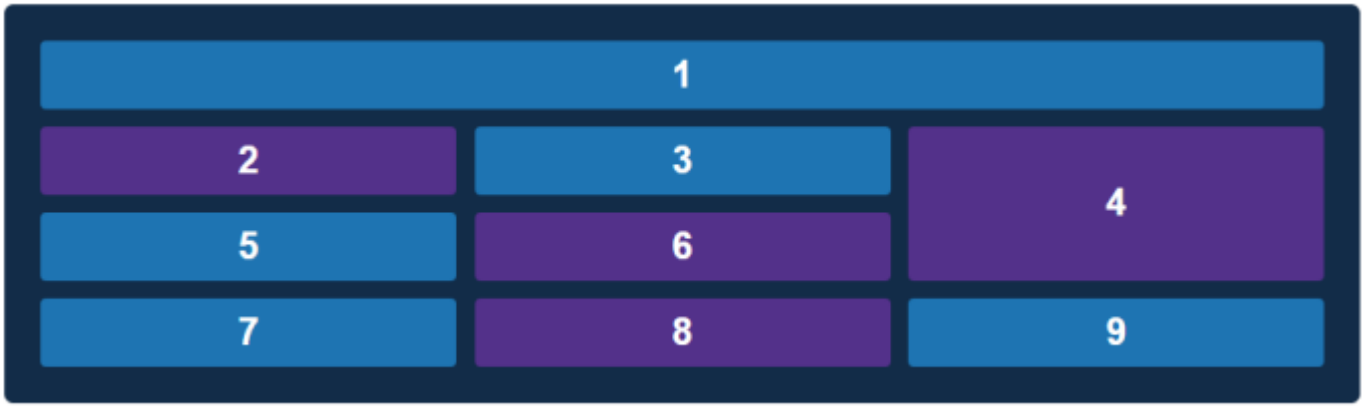
Позиционирование элементов сетки с помощью номеров линий сетки

- В примере сетка состоит из 4 строк и 3 столбцов. Мы размещаем первый элемент сетки, применяя к ней свойства grid-column-start, grid-column-end, grid-row-start и grid-row-end. Двигаясь слева направо, первый элемент начинается с вертикальной линии 1 и продолжается до вертикальной линии 4, которая находится справа в нашей сетке. Он начинается на горизонтальной линии 1 и заканчивается на горизонтальной линии 2, то есть элемент занимает 3 столбца и 1 строку сетки.
- Второй элемент начинается с вертикальной линии 1 и заканчивается на вертикальной линии 2, то есть элемент занимает 1 столбец и 1 строку.
- Третий элемент располагается по аналогии со вторым элементом и занимает столько же ячеек.
- Четвертый элемент начинается с вертикальной линии 3 и заканчивается на вертикальной линии 4, начинается на горизонтальной линии 2 и заканчивается на горизонтальной линии 4, то есть занимает 1 столбец и 2 строки.

Позиционирование элементов сетки с помощью номеров линий сетки

- Если элемент будет занимать только одну ячейку сетки, можно опустить значение -end, как показано в следующем примере:

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



Сокращенные формы записи для позиционирования

- Мы можем достичь такого же результата, как в предыдущем примере, используя сокращенный синтаксис, который объявляет значение для начальной и конечной линий сетки сразу.
- Свойства:
- **grid-column** — задает начальную и конечную линию расположения элемента в строке.
- **grid-row** — задает начальную и конечную линию расположения элемента в столбце.
- **grid-area** — задает начальную и конечную линию расположения элемента в строке и столбце.
- В следующем примере элемент начинается с вертикальной линии 1 и заканчивается на вертикальной линии 4, начинается на горизонтальной линии 1 и заканчивается на горизонтальной линии 2, то есть занимает 3 столбца и 1 строку:

```
.item-1{
            grid-column: 1/4;
            grid-row: 1/2;
        }
```

Сокращенные формы записи для позиционирования

- Если элемент занимает только одну ячейку, можно задать только значение для начальной линии сетки:

```
.item-1{
            grid-column: 1/4;
            grid-row: 1;
        }
```

- Можно определить сразу целую область, которую элемент будет занимать:

```
.item-4{
            grid-area: 2/3/4/4;
        }
```

- Порядок значений следующий: **row-start / column-start / row-end / column-end**.
- В примере выше элемент начинается с горизонтальной линии 2 и заканчивается на горизонтальной линии 4, начинается с вертикальной линии 3 и заканчивается на вертикальной линии 4, то есть занимает 2 строки и 1 столбец.

Объединение ячеек с помощью ключевого слова span

- Ключевое слово span дает возможность объединять ячейки без указания начальной и конечной линии трека.
- В следующем примере элемент начинается с вертикальной линии 1 и заканчивается на вертикальной линии 2, начинается на горизонтальной линии 1 и объединяет 3 ячейки по вертикали, т.е. заканчивается на горизонтальной линии 4. Элемент занимает 1 столбец и 3 строки.

```
.item-1{
            grid-column: 1;
            grid-row: 1/span 3;
        }
```

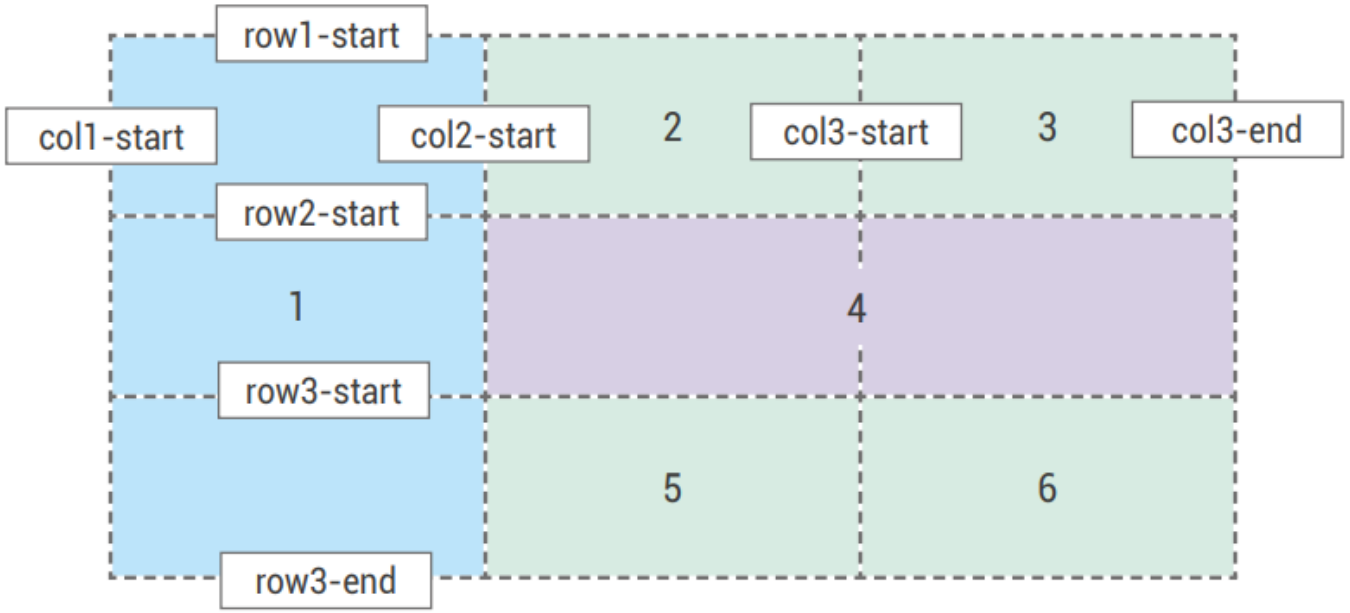
Именованные линии сетки

- Хотя на линии сетки можно ссылаться по их числовому индексу, именованные линии облегчают понимание и использование свойств размещения сетки.
- Имена линиям можно задать явно, используя свойства **grid-template-rows** и **grid-template-columns** или неявно — путем создания именованных областей сетки с помощью свойства **grid-template-areas**. Имя линии может быть любым. При задании в значении свойства, его следует взять в квадратные скобки. Ключевое слово span не может использоваться в качестве названия линии сетки.
- Даже если мы задаем линиям имена, мы все равно можем обращаться к ним по индексу.
- В следующем примере создается 3 строки и 3 столбца и для каждой линии сетки задается имя:

```
 .grid {
     display: grid;
     grid-template-columns: [col1-start] auto [col2-start] auto [col3-
start] auto [col3-end];
     grid-template-rows: [row1-start] auto [row2-start] auto [row3-
start] auto [row3-end];
     gap: 10px;
}
```

Именованные линии сетки

- На рисунке показано расположение именованных линий:



- Теперь для позиционирования элементов в сетке можно использовать имена линий сетки, например:

```
.item-1{
          grid-column: col1-start/col3-end;
          grid-row: row1-start;
      }
```

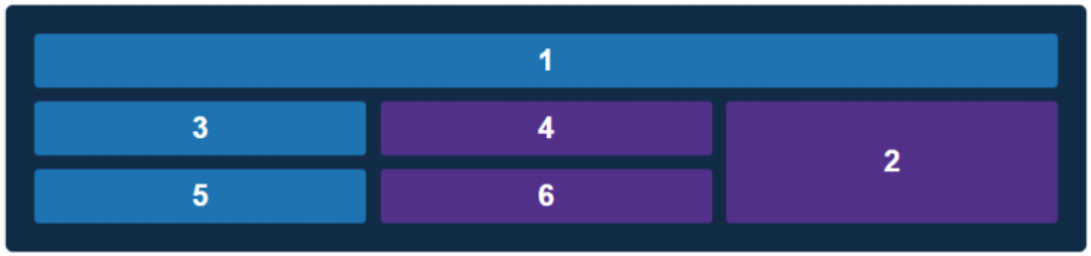
Области сетки

- Мы можем создать именованные участки сетки для размещения содержимого. Для этого мы сначала определяем элементы нашей разметки для сетки, используя свойство **grid-area**, например:

```
.item-1{
    grid-area: area1;
}
.item-2{
    grid-area: area2;
}
.item-3{
    grid-area: area3;
}
```

- А потом мы можем использовать свойство **grid-template-areas**, чтобы сказать, где именно на сетке должны располагаться эти элементы:

```
.grid-container{
    display: grid;
    grid-template-areas: "area1 area1 area1"
                         "area3 area4 area2"
                         "area5 area6 area2";
    gap: 10px;
}
```



Области сетки

- Этот подход очень удобен при создании и компоновке небольших макетов. Рассмотрим еще один пример. Создадим несколько HTML-элементов:

```
<div class="grid-container">
        <header>Header</header>
        <nav>Menu</nav>
        <main>Main</main>
        <aside>Right</aside>
        <footer>Footer</footer>
    </div>
```

- Добавим стили:

```
*,
*::after,
*::before{
    box-sizing: border-box;
}
*{
    margin: 0;
    padding: 0;
}
header, nav, footer{
    text-align: center;
}
```

Области сетки

```
 .grid-container{
     display: grid;
     color: white;
     background-color: #0d233d;
     font-size: 20px;
     padding: 15px;
     font-family: Arial, Helvetica, sans-serif;
     border-radius: 5px;
     gap: 10px;
     grid-template-areas: "header header header header header header"
            "menu main main main right right"
     min-height: 100vh;
     height: auto;
     grid-template-columns: repeat(6, 1fr);
}
header, nav, footer{
     text-align: center;
}
.grid-container>*{
     border-radius: 5px;
     padding: 10px;
}
.grid-container>div{
}
header{
     grid-area: header;
     background-color: crimson;
}
```

Области сетки

```
nav{
     grid-area: menu;
     background-color: gold;
}
main{
    grid-area: main;
    background-color: cornflowerblue;
}
aside{
    grid-area: right;
    background-color: cadetblue;
}
footer{
    grid-area: footer;
    background-color: blueviolet;
}
```

- Результат:

Спасибо за внимание.