Конспект «Микросетки. Начало». Раздел 1

Микросетки

Микросетки — сетки мелких элементов веб-страницы.

В отличие от крупных сеток, микросетки меньше зависят от макета и больше — от содержимого. Содержимое страниц со временем может меняться. Если не учитывать этого, вёрстка сломается.

Отступы у ссылок

Часто ссылкам добавляют внутренние отступы, чтобы увеличить область, по которой можно кликнуть (её ещё называют активной областью). Чем проще попасть по ссылке, тем удобнее интерфейс.

По умолчанию ссылки имеют строчный тип бокса. Браузер игнорирует внешние отступы по вертикали у строчных боксов, а их внутренние отступы сверху и снизу не влияют на расположение других элементов. Самый простой способ решить эту проблему — изменить у ссылок тип бокса. Например, сделать их блочными боксами:

```
.element {
  display: block;
}
```

Свойство align-items

По умолчанию грид-элементы занимают всё доступное пространство по высоте. Такое поведение можно изменить с помощью свойства `align-items`. Оно задаётся грид-контейнеру и управляет выравниванием грид-элементов по вертикали.

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

У `align-items` могут быть следующие значения:

- `stretch` — значение по умолчанию; элементы начинаются у верхней границы и растягиваются на всю высоту.
- `start` — элементы выстраиваются по верхней границе и, если содержимого немного, не тянутся на всю высоту;
- `end` — элементы выстраиваются по нижней границе;
- `center` — элементы располагаются по центру;

Свойство flex-wrap

Если элементам не хватает места, они могут вылезти за границы контейнера. Такое поведение называют *выпадением элементов*. Также говорят, что произошло *переполнение*.

По умолчанию флекс-контейнер однострочный. Чтобы элементы не выпадали из контейнера, его делают многострочным. Для этого используют свойство `flex-wrap` со значением `wrap`.

```
.flex-container {

  display: flex;

  flex-wrap: wrap;

}
```

Конспект «Микросетки. Начало». Раздел 2

Список на флексах

Свойство `justify-content` со значением `space-between` заставляет первый и последний элемент прижиматься к границам контейнера. Но если в ряду всего два элемента, то свободного пространства между ними может оказаться слишком много. В этом случае лучше использовать `margin`.

Чтобы убрать лишний отступ у последнего элемента в ряду, используют псевдокласс `:nth-child`. Он позволяет выбрать дочерний элемент по его порядковому номеру:

```
// Выберет второй элемент с классом item

.item:nth-child(2) { ... }



// Выберет каждый второй элемент с классом item

.item:nth-child(2n) { ... }
```

Если не известно, какой элемент окажется в ряду последним, этот способ не сработает.

repeat

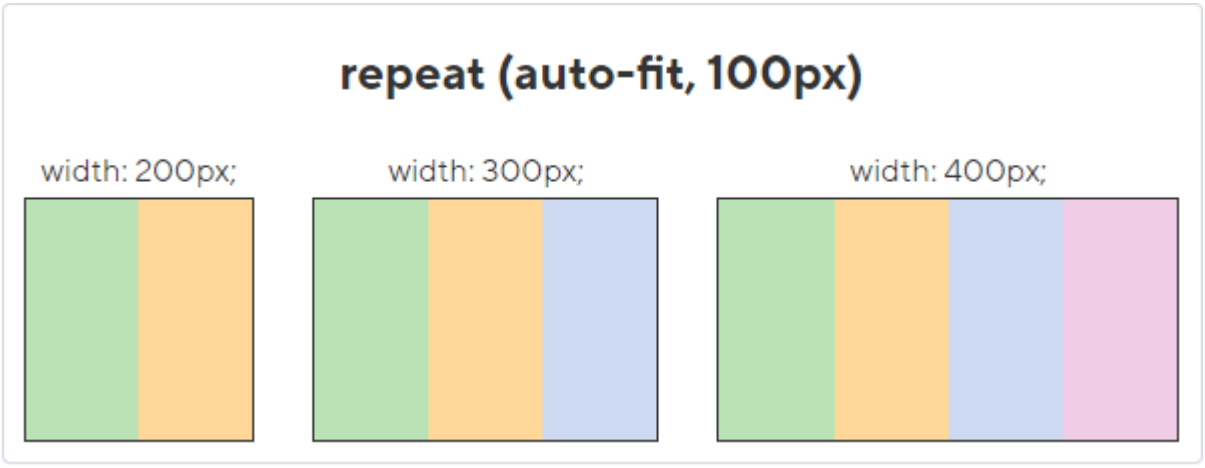
Если все колонки в грид-контейнере должны быть одинаковой ширины, то удобно использовать значение-функцию `repeat`. В скобках после `repeat` указывают количество колонок и их ширину. Значения разделяют запятой:

```
grid-template-columns: repeat(количество колонок, ширина колонки);
```

auto-fit

Если количество колонок зависит от ширины контейнера, используют специальное значение `auto-fit`. Его указывают в скобках после `repeat` вместо числа колонок:

```
grid-template-columns: repeat(auto-fit, ширина колонки);
```



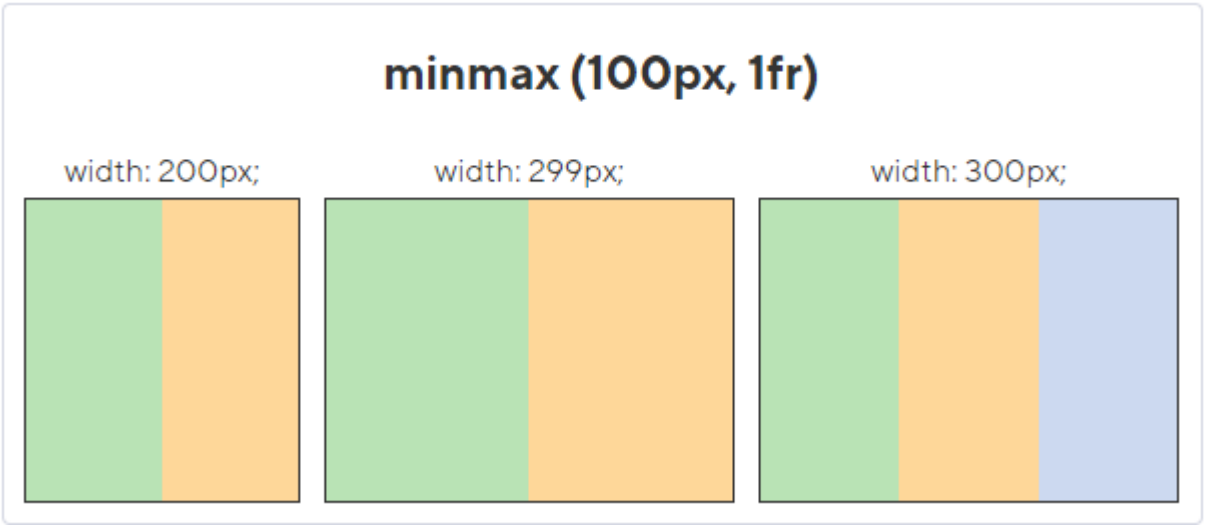
minmax

Чтобы ширина колонок изменялась пропорционально свободному пространству в контейнере, используют значение-функцию `minmax`.

Его указывают в `repeat` вместо фиксированной ширины колонок. В скобках после `minmax` задают минимальный и максимальный размеры колонок, они разделяются запятой:

```
repeat(auto-fit, minmax(минимальный размер, максимальный размер));
```

В `minmax` в качестве максимального значения часто используют единицу измерения `fr`. Она позволяет колонкам увеличивать ширину до тех пор, пока свободного пространства в контейнере не хватит на ещё одну колонку.



Свойства grid-column и grid-row

Чтобы растянуть элемент на несколько колонок используют свойство `grid-column`. Число после ключевого слова `span` указывает число колонок, которые элемент должен занять:

```
.element {
  grid-column: span 2;
}
```

Растянуть элемент на несколько рядов можно с помощью свойства `grid-row`. Ключевое слово `span` в нём означает количество рядов, которые элемент должен занять:

```
.long-element {
  grid-row: span 2;
}
```

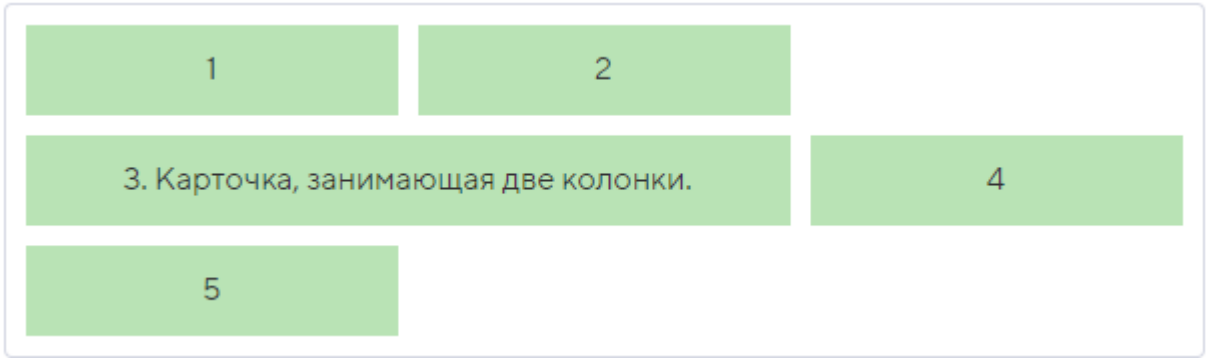
Свойства `grid-column` и `grid-row` можно использовать одновременно.

Свойство grid-auto-flow

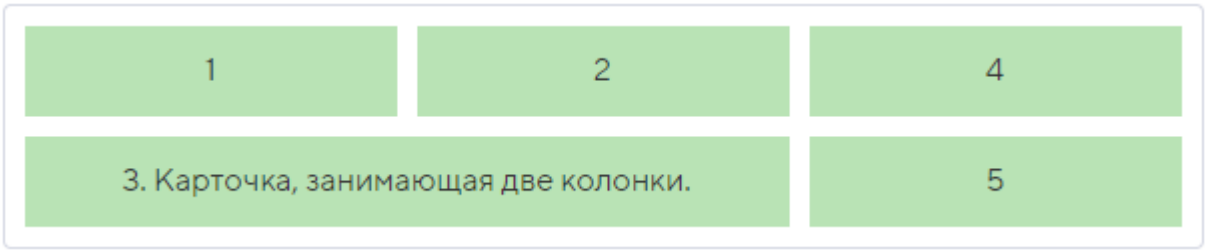
Свойство `grid-auto-flow` управляет автозаполнением грид-контейнера.

```
.grid-container {
  display: grid;
  grid-auto-flow: row;
}
```

Значение по умолчанию `row` говорит располагать элементы в том порядке, в котором они идут в разметке, и при необходимости создавать новые ряды:



Но если указать значение `dense`, то контейнер будет заполняться так, чтобы не было пропусков:



Значение `dense` заставляет грид-контейнер заполнять пустые ячейки первым подходящим по размеру грид-элементом. При этом визуальный порядок на странице может отличаться от порядка элементов в разметке. Если порядок элементов важен, лучше это значение не использовать.