**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);

**repeat (auto-fit, 100px)**

width: 200px;

width: 300px;

width: 400px;

**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.

**minmax (100px, 1fr)**

width: 200px;

width: 299px;

width: 300px;

**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:

1

2

3. Карточка, занимающая две колонки.

4

5

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

1

2

3. Карточка, занимающая две колонки.

4

5

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.

**Конспект «Микросетки. Продолжение». Раздел 1**

**Поперечная ось и свойство align-items (флекс)**

Во флексах свойство align-items управляет расположением элементов на поперечной оси.

Поперечная ось идёт перпендикулярно главной оси и по умолчанию направлена сверху вниз:

Во флекс-контейнере свойство align-items может иметь следующие значения:

* stretch — значение по умолчанию; элементы растягиваются на всю высоту поперечной оси.
* flex-start — элементы сжимаются до содержимого и располагаются в начале поперечной оси (по умолчанию сверху);
* flex-end — элементы сжимаются до содержимого и располагаются в конце поперечной оси (по умолчанию снизу);
* center — элементы сжимаются до содержимого и располагаются по центру поперечной оси;

**Поворот главной оси, свойство flex-direction**

Главную ось поворачивают, чтобы сохранить внутри флекс-контейнера направление потока сверху вниз. За направление главной оси отвечает свойство flex-direction. По умолчанию у него значение row (ряд), но его можно изменить на column (колонка):

flex-container {

display: flex;

flex-direction: column;

}

В этом случае главная ось будет направлена сверху вниз, а поперечная — слева направо. В результате флекс-элементы выстроятся сверху вниз.

По умолчанию флекс-элементы сжимаются по главной оси и растягиваются по поперечной. Таким образом, если главная ось направлена слева направо, то элементы сжимаются по горизонтали и растягиваются по вертикали.

Если же главная ось направлена сверху вниз, то сжатие происходит по вертикали, а растяжение — по горизонтали.

Получается, чтобы при повёрнутых осях выровнять элемент по горизонтали, нужно задать ему выравнивание по поперечной оси.

**Свойство align-self (флекс)**

Свойство align-self задаётся флекс-элементу и говорит, как ему расположиться на поперечной оси. Значения у этого свойства такие же, как у align-items: stretch (значение по умолчанию), flex-start, flex-end и center.

.element {

align-self: flex-end;

}

**Свойство order**

Чтобы изменить визуальный порядок элементов, удобно использовать свойство order. В качестве значения свойство принимает число, причём оно может быть как положительным, так и отрицательным. По умолчанию у всех элементов свойство order равно нулю.

.element {

order: 5;

}

Элементы выстраиваются от меньшего значения order к большему. Если у нескольких элементов одинаковое значение, используется их порядок в разметке.

Свойство order работает только в грид- и флекс-контейнерах.

**Отступы у флекс-элементов**

Внешние отступы у соседних флекс-элементов складываются.

Чтобы не запутаться и получить именно те размеры, которые указаны в макете, верстальщики добавляют элементам внешние отступы только с одной стороны. Часто внешние отступы задают в направлении потока. Если элементы выстроены горизонтально, то отступ задают справа, а у последнего элемента обнуляют:

Если элементы выстроены вертикально, то отступ добавляют снизу. Исключение — самый последний элемент (например, подвал страницы), ему при необходимости задают отступ сверху:

В этом случае, даже если изменить порядок секций, они не слипнутся, а между ними не появятся лишние отступы.

**Уменьшение изображений с сохранением пропорций**

Если использовать картинку большего размера, чем родительский элемент, она выпадет из контейнера. Чтобы не допускать подобного, верстальщики добавляют картинкам такие стили:

img {

max-width: 100%;

height: auto;

}

Свойство max-width задаёт максимальную ширину, а значение 100% говорит, что элемент не должен становиться больше ширины родителя.

Свойство height задаёт высоту элемента. Значение auto используют, чтобы изображение не деформировалось и сохраняло свои пропорции. Если его не указать, то будет использовано значение атрибута height из разметки, и картинка, скорее всего, исказится.

Этот способ работает, потому что у CSS-свойств max-width и height приоритет выше, чем у атрибутов width и height в разметке.

**Конспект «Микросетки. Продолжение». Раздел 2**

Придумать подходящую структуру грид-контейнера бывает непросто. Если элементы расположены асимметрично, часто требуется больше колонок, чем кажется на первый взгляд.

**Грид-области**

Грид-областью называют часть сетки грид-контейнера, у которой есть имя. Имя области придумывает сам разработчик. Оно должно начинаться с буквы и может включать цифры, дефис и знак подчёркивания. Например: header, section-2, user\_avatar. Следует выбирать такие имена, которые описывают содержимое области.

Чтобы описать структуру грида с помощью областей, используют свойство grid-template-areas. В нём указывают имена грид-областей. При этом каждый ряд оборачивают в кавычки, а колонки разделяют пробелом.

.card {

display: grid;

grid-template-areas: "title price";

}

В grid-template-areas можно указать сколько угодно рядов. При этом в кавычки оборачивают **каждый** ряд, но точка с запятой ставится только после последнего ряда! Для большей наглядности ряды записывают друг под другом:

grid-template-areas: "title price"

"options description";

Однако просто описать шаблон недостаточно, ведь браузер не знает, какие элементы мы имеем в виду. Чтобы связать имя области с соответствующим грид-элементом, используют свойство grid-area. Обратите внимание, в grid-area кавычки не нужны!

.title {

grid-area: title;

}

**Как растянуть грид-область**

Если требуется растянуть область на несколько колонок, её имя повторяют нужное число раз:

grid-template-areas: "title title price";

Чтобы растянуть область на несколько рядов, достаточно повторить её имя в разных рядах:

grid-template-areas: "options description"

"options disclaimer";

**Ограничения при использовании грид-областей**

Грид-области должны быть прямоугольными и непрерывными.

// Такой код не сработает

grid-template-areas: "logo main"

"main main";

// Такой – тоже

grid-template-areas: "nav logo nav";

При описании шаблона количество колонок в каждом ряду должно быть одинаковым:

// Такой код не сработает

grid-template-areas: "logo nav"

"aside main banner"

"footer";

Если задать неправильное значение grid-template-areas, браузер его проигнорирует, и вёрстка может сломаться.

Имена областей *в одном контейнере* должны быть уникальными. Если задать одинаковое имя нескольким элементам, они наложатся друг на друга. По этой причине грид-области не подойдут, например, для списка карточек, особенно если количество этих карточек неизвестно заранее.

**Свойство align-self (грид)**

Свойство align-self задаёт выравнивание по вертикали *одному элементу*. Значения align-self принимает те же, что и align-items: stretch (значение по умолчанию), start, end и center.

.element {

align-self: start;

}

**Свойство justify-self**

Свойство justify-self отвечает за выравнивание отдельного элемента по горизонтали:

.element {

justify-self: start;

}

Это свойство принимает следующие значения:

* stretch — значение по умолчанию; элемент занимает всё пространство по ширине.
* start — элемент сжимается до содержимого и прижимается к левой границе.
* end — элемент сжимается до содержимого и прижимается к правой границе.
* center — элемент сжимается до содержимого и располагается по центру.