# Лабораторная работа №2

Тема: Изучение CSS

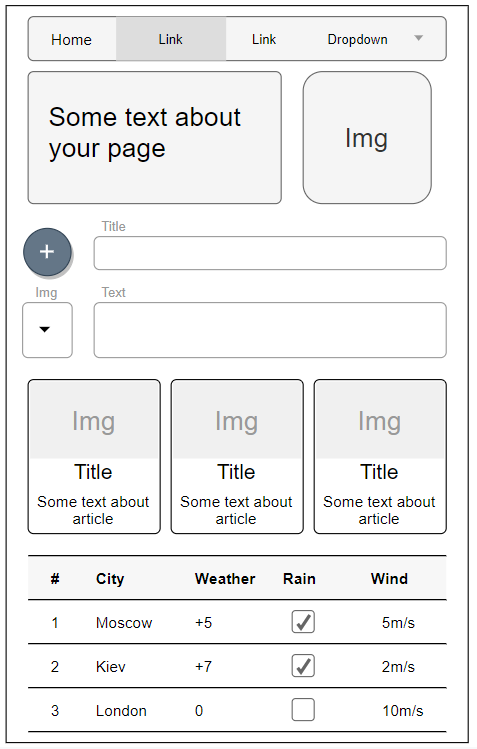
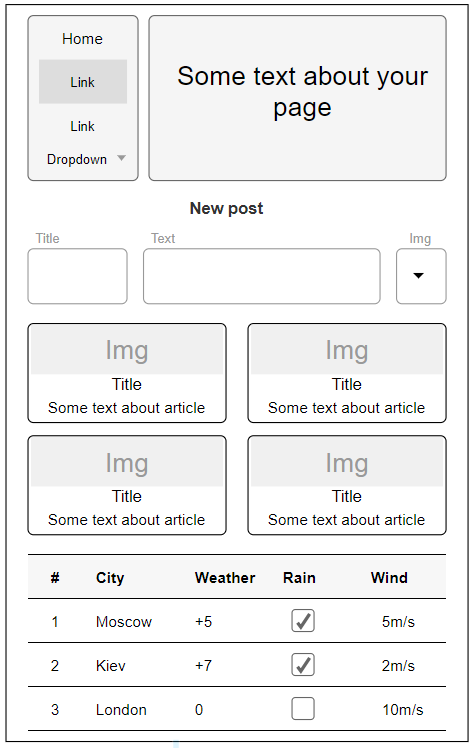
**Задание**

На основе лабораторной работы №1 выполнить оформление страницы и верстку адаптивного макета средствами CSS, CSS Grid и CSS Flexbox.

**Требования**

1. Использование селекторов элементов, классов, идентификаторов;
2. Добавление псевдоклассов, например hover, active, visited, focus.
3. Разметка страницы с помощью css-grid и выравнивание с помощью flexbox;
4. Добавление элементов оформления: цветов, шрифтов, размеров и т.д.
5. Использование медиа-запросов для обеспечения адаптивности страницы;

Примеры макетов

# Справочная информация

## Определение

CSS расшифровывается как Cascading Style Sheets или каскадные таблицы стилей и является формальным языком описания внешнего вида документа, написанного с использованием языка разметки. Это подразумевает настройку цвета, шрифтов, размеров, полей, положения элементов веб-страницы и всего, что касается внешнего вида веб-страниц.

## Подключение к HTML-документу

<link rel="stylesheet" href="style.css">

Значение атрибута тега rel означает, что подключаемый файл - таблица стилей. Значение href задаёт путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Таким образом можно подключать таблицу стилей, которая находится на другом сайте.

Файл со стилем не хранит никаких данных, кроме синтаксиса CSS. В свою очередь и HTML-документ содержит только ссылку на файл со стилем, т. е. таким способом в полной мере реализуется принцип разделения кода и оформления сайта. Поэтому использование связанных стилей является наиболее универсальным и удобным методом добавления стиля на сайт.

## Синтаксис

Общий способ записи имеет следующий вид:

селектор, селектор {

свойство: значение;

свойство: значение;

свойство: значение;

}

Селектор — это то, к чему применяются указанные стили. В качестве селектора выступают теги, классы и идентификаторы, а также псевдоклассы и псевдоэлементы.

Стили могут быть заданы как к одному, так и сразу к нескольким селекторам. Например, в записи p, div, h1, h2, h3 {color: red} стили будут применены к каждому из селекторов.

CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции.

Комментарии в css имеют следующую конструкцию: /\* ... \*/

## Селекторы и псевдоклассы

**1. Универсальный селектор**

Соответствует любому HTML-элементу. Например, \* {margin: 0;} обнулит внешние отступы для всех элементов сайта. Также селектор может использоваться в комбинации с псевдоклассом или псевдоэлементом: \*:after {CSS-стили}, \*:checked {CSS-стили}.

**2. Селектор элемента**

Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта. Например, h1 { CSS-стили } задаст общий стиль форматирования всех заголовков h1.

**3. Селектор класса**

Селекторы класса позволяют задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта. Например, для создания заголовка с классом main необходимо добавить атрибут class со значением main в открывающий тег и задать стиль для указанного класса.

HTML

<h1 class=" main ">Заголовок</h1>

<p class = " main ">Текст</p>

CSS

.main { CSS-стили }

Если элемент имеет несколько атрибутов класса, их значения объединяются с пробелами.

<h1 class="main anotherclass"> Заголовок </h1>

**4. Селектор идентификатора**

Селектор идентификатора позволяет форматировать один конкретный элемент. Значение id должно быть уникальным, на одной странице может встречаться только один раз и должно содержать хотя бы один символ. Значение не должно содержать пробелов.

Нет никаких других ограничений на то, какую форму может принимать id, в частности, идентификаторы могут состоять только из цифр, начинаться с цифры, начинаться с подчеркивания, состоять только из знаков препинания и т. д.

Уникальный идентификатор элемента может использоваться для различных целей, в частности, как способ ссылки на конкретные части документа с использованием идентификаторов фрагментов, как способ нацеливания на элемент при создании сценариев и как способ стилизации конкретного элемента из CSS.

HTML

<div id="sidebar"></div>

CSS

#sidebar { CSS-стили }

**5. Селектор потомка**

Селекторы потомков применяют стили к элементам, расположенным внутри элемента-контейнера. Например, ul li { CSS-стили } — выберет все элементы li, являющиеся потомками всех элементов ul.

Если нужно отформатировать потомки определенного элемента, этому элементу нужно задать стилевой класс:

p.first a { CSS-стили } — данный стиль применится ко всем ссылкам, потомкам абзаца с классом first;

p .first a { CSS-стили } — если добавить пробел, то будут стилизованы ссылки, расположенные внутри любого тега класса .first, который является потомком элемента <p>;

.first a { CSS-стили} — данный стиль применится к любой ссылке, расположенной внутри другого элемента, обозначенного классом .first.

**6. Дочерний селектор**

Дочерний элемент является прямым потомком содержащего его элемента. У одного элемента может быть несколько дочерних элементов, а родительский элемент у каждого элемента может быть только один. Дочерний селектор позволяет применить стили только если дочерний элемент идёт сразу за родительским элементом и между ними нет других элементов, то есть дочерний элемент больше ни во что не вложен.

Например, p > strong — выберет все элементы strong, являющиеся дочерними по отношению к элементу p.

**7. Сестринский селектор**

Сестринские отношения возникают между элементами, имеющими общего родителя. Селекторы сестринских элементов позволяют выбрать элементы из группы элементов одного уровня.

h1 + p — выберет все первые абзацы, идущие непосредственно за любым тегом <h1>, не затрагивая остальные абзацы;

h1 ~ p — выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку h1 и идущие сразу после него.

**8. Селектор псевдокласса**

Псевдоклассы — это классы, фактически не прикрепленные к HTML-тегам. Они позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу. Псевдоклассы характеризуют элементы со следующими свойствами:

:link — не посещенная ссылка;

:visited — посещенная ссылка;

:hover — любой элемент, по которому проводят курсором мыши;

:focus — интерактивный элемент, к которому перешли с помощью клавиатуры или активировали посредством мыши;

:active — элемент, который был активизирован пользователем;

:valid — поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;

:invalid — поля формы, содержимое которых не соответствует указанному типу данных;

:enabled — все активные поля форм;

:disabled — заблокированные поля форм, т.е., находящиеся в неактивном состоянии;

:in-range — поля формы, значения которых находятся в заданном диапазоне;

:out-of-range — поля формы, значения которых не входят в установленный диапазон;

:lang() — элементы с текстом на указанном языке;

:not(селектор) — элементы, которые не содержат указанный селектор — класс, идентификатор, название или тип поля формы — :not([type="submit"]);

:target — элемент с символом #, на который ссылаются в документе;

:checked — выделенные (выбранные пользователем) элементы формы.

**9. Селектор структурных псевдоклассов**

Структурные псевдоклассы отбирают дочерние элементы в соответствии с параметром, указанным в круглых скобках:

:nth-child(odd) — нечётные дочерние элементы;

:nth-child(even) — чётные дочерние элементы;

:nth-child(3n) — каждый третий элемент среди дочерних;

:nth-child(3n+2) — выбирает каждый третий элемент, начиная со второго дочернего элемента (+2);

:nth-child(n+2) — выбирает все элементы, начиная со второго;

:nth-child(3) — выбирает третий дочерний элемент;

:nth-last-child() — в списке дочерних элементов выбирает элемент с указанным местоположением, аналогично с :nth-child(), но начиная с последнего, в обратную сторону;

:first-child — позволяет оформить только самый первый дочерний элемент тега;

:last-child — позволяет форматировать последний дочерний элемент тега;

:only-child — выбирает элемент, являющийся единственным дочерним элементом;

:empty — выбирает элементы, у которых нет дочерних элементов;

:root — выбирает элемент, являющийся корневым в документе — элемент html.

**10. Селектор структурных псевдоклассов типа**

Указывают на конкретный тип дочернего тега:

:nth-of-type() — выбирает элементы по аналогии с :nth-child(), при этом берёт во внимание только тип элемента;

:first-of-type — выбирает первый дочерний элемент данного типа;

:last-of-type — выбирает последний элемент данного типа;

:nth-last-of-type() — выбирает элемент заданного типа в списке элементов в соответствии с указанным местоположением, начиная с конца;

:only-of-type — выбирает единственный элемент указанного типа среди дочерних элементов родительского элемента.

**11. Селектор псевдоэлемента**

Псевдоэлементы используются для добавления содержимого, которое генерируется с помощью свойства content:

:first-letter — выбирает первую букву каждого абзаца, применяется только к блочным элементам;

:first-line — выбирает первую строку текста элемента, применяется только к блочным элементам;

:before — вставляет генерируемое содержимое перед элементом;

:after — добавляет генерируемое содержимое после элемента.

**12. Комбинация селекторов**

Для более точного отбора элементов для форматирования можно использовать комбинации селекторов:

a[href][title] — выберет все ссылки, для которых заданы атрибуты href и title;

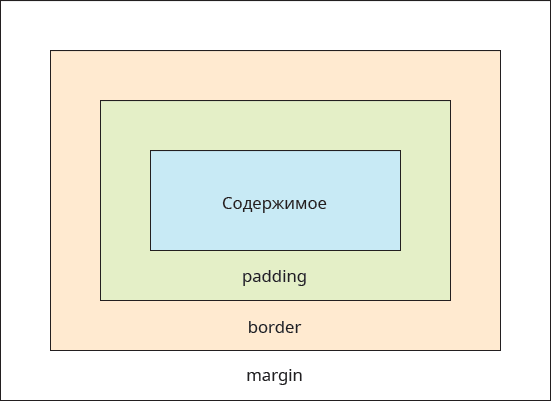
img[alt\*="css"]:nth-of-type(even) — выберет все четные картинки, альтернативный текст которых содержит слово css.

## Свойства блока

Когда в браузере открывается веб-страница, он воспринимает ее теги в виде отдельных прямоугольных блоков (контейнеров), наполненных содержимым. У каждого такого элемента есть свой контент (текст, графика и прочее) и свои настройки CSS. Для блоков можно определять размеры, отступы, поля, рамки, фон, тем самым создавая для них стиль. Рассмотрим основные свойства CSS, устанавливаемые для блоков:

**padding** (и производные padding-left, padding-right, padding-top, padding-bottom) – это внутренний отступ от края содержимого до границы блока. Можно устанавливать отступы либо от всех четырех сторон контента, либо только от нужных вам.

**border** (и производные border-left, border-right, border-top, border-bottom) – это рамка (граница), которую можно задать для блока. Поля margin остаются за пределами рамки, всё остальное находится внутри неё. Рамку можно установить на всех четырех сторонах или только на нужных вам.

**margin** (и производные margin-left, margin-right, margin-top, margin-bottom) – это невидимое пустое пространство от внешнего края границ, которое являет собой расстояние от границы одного блока до другого. 

Любой элемент на веб-странице представляет собой прямоугольный блок, который состоит из набора свойств, накладываемых друг на друга. Основой блока выступает его содержимое (это может быть текст, изображение, видео и др.), ширина которого задается свойством width, а высота через height;

## Фон и цвет

**background** – позволяет установить либо цвет, либо фоновую картинку на сайт.

**background-image** – устанавливает фоновое изображение элемента:

background-image: url (img/image.png);

**сolor** – описывает цвет текстового содержимого элемента(значения: ключевые слова, цвета модели RGB и RGBA)

Например, черный цвет можно описать как black, #000000 или rgb(0,0,0).

**background-color** – позволяет установить фоновый цвет элемента так же, как и color.

CSS также позволяет определять цветовые градиенты в качестве фонового изображения, двух различных видов:

**linear-gradient** для градиентов в одном направлении, в прямоугольной форме;

**radial-gradient** для градиентов во всех направлениях, в круговой форме.

background-image: linear-gradient(white, blue); }

**background-position** – указывает на начальное положение фонового рисунка, в качестве значения использует 2 параметра, 1 – положение по горизонтали (left, center, right), 2 – положение по вертикали (left, center, right);

**background-repeat** – повторение фонового изобжения.

По умолчанию фоновое изображение будет повторяться до бесконечности. Вы можете выбрать, чтобы оно повторялось только по горизонтали, только по вертикали или не повторялось вообще.

background-repeat: repeat-x; /\* Только по горизонтали \*/

background-repeat: repeat-y; /\* Только по вертикали \*/

background-repeat: no-repeat; /\* Фоновое изображение появится только один раз \*/

## Шрифты и текст

**font-family** – Устанавливает шрифт;

**font-size** – Устанавливает размер шрифта;

**font-stretch** – Устанавливает начертание шрифта( узкое,нормальное,или широкое);

**font-style** – Определяет начертание шрифта( обычное, курсив, наклонное);

**font-variant** – Определяет как нужно представлять строчные буквы;

**font-weight** – Устанавливает насыщенность( жирность) шрифта;

## Размер элементов

**width** — устанавливает ширину для блочных элементов (px, em, %, auto, inherit);

**height**— устанавливает высоту для блочных элементов (px, em, %, auto, inherit);

**max-width** – свойство устанавливает максимальную ширину элемента(px, em, %, auto, inherit);

**max-height** – свойство устанавливает максимальную высоту элемента(px, em, %, auto, inherit);

**min-width** – Задает минимальную ширину элемента(px, em, %, auto, inherit).

**min-height** – Задает минимальную высоту элемента(px, em, %, auto, inherit).

**text-align** – отвечает за горизонтальное выравнивание текста;

Значения:

Auto - Вычисляется в зависимости от значений других свойств. Значение по умолчанию.

Inherit- Наследует значение свойства от родительского элемента.

% - Вычисляется относительно содержащего блока. Для абсолютно позиционированных элементов процент вычисляется с учетом ширины области отступов padding содержащего блока.

em – относительный размер, определяется по текущему контексту.

## Позиционирование элементов

**Position** - изменение типа позиционирования. Используется как правило в комбинации со свойствами left, right, top, bottom, которые определяют, соответственно, положение элемента слева, справа, сверху и снизу.

У position есть пять значений.

static — нормальное или статичное позиционирование, при этом элементы отображаются на веб-странице в том порядке, в каком они идут в исходном коде HTML сверху вниз.

relative — относительное позиционирование. Изменяет положение элемента от его исходного расположения.

absolute — абсолютное позиционирование. Элемент при этом не существует в потоке документа и его положение задаётся относительно краёв браузера.

fixed — фиксированное позиционирование. По своему действию похоже на абсолютное позиционирование, но в отличие от него привязывается к указанной свойствами left, top, right и bottom точке на экране и не меняет своего положения при прокрутке веб-страницы.

sticky — липкое позиционирование. Обычно применяется для фиксации заголовка на одном месте, пока содержимое, к которому относится заголовок, прокручивается на странице.

## Адаптивность веб-страниц

**Адаптивный макет сайта** — это макет, который может «приспосабливаться» под различные устройства (ширину рабочей области окна браузера). Т.е. на одних устройствах он может иметь одну структуру, а на других - другую.

## Особенности мобильных устройств

Страницы, адаптированные для просмотра на разных типах устройств, должны содержать в разделе <head> метатег viewport.

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

## Медиа запросы

Медиа запросы — это правила CSS, которые позволяют управлять стилями элементов в зависимости от значений технических параметров устройств. Иными словами, это конструкции, которые позволяют определять на основании некоторых условий какие стили необходимо использовать на веб-странице, а какие нет.

Все запросы начинаются с правила @media, после чего следует условие, в котором используются типы носителей, логические операторы и медиа-функции.

Типом носителя может быть экран монитора (screen), cмартфоны (handheld), принтеры (print), телевизоры (tv) и другие устройства. Но, чаще всего, этих значений недостаточно, поэтому используются контрольные точки в пикселях.

@media (min-width: 576px) {

Стили для маленьких экранов (дисплей 576 пикселей и более)

}

@media (min-width: 768px) {

Стили для средних экранов (планшеты, дисплей 768 пикселей и более)

}

@media (min-width: 992px) {

Стили для больших экранов (ПК, дисплей 992 пикселей и более)

}

@media (min-width: 1200px) {

Стили для очень больших экранов (ПК с большим монитором, 1200 пикселей и более)

}

Также можно использовать логические операторы and (логическое и), not (логическое не), и запятую (логическое или).

@media screen and (max-width: 600px) { CSS-стили }

@media all and (not handheld) { CSS-стили }

@media all and (orientation: landscape), all and (min-width: 480px) { CSS-стили }

**Медиа функции**

Для составления условия в @media можно использовать следующие фукнции:

**width** - указывает требования к ширине области просмотра устройства (браузера).

**min-width** - задаёт минимальную ширину области viewport в px, em или других единицах.

**max-width** - указывает на то, какой должна быть максимальная рабочая область устройства (браузера).

**height, min-height и max-height** - задают требования аналогично вышеприведённым функциям, но в отношении высоты viewport.

**orientation** - функция, которая проверяет то, в каком режиме (portrait или landscape) отображается страница. Landscape (альбомный) - это режим, в котором наоборот ширина viewport больше её высоты. Portrait (портретный) - это режим, в котором высота viewport больше ширины \*/

**aspect-ratio** (min-aspect-ratio, max-aspect-ratio) - позволяют указать то, как ширина устройства должна относиться к высоте. В качестве значений допускается использовать только целые значения.

**resolution** (min-resolution, max-resolution) - указывает разрешение (плотность пикселей) устройства вывода. В качестве единиц измерения разрешения используются следующие величины: dpi (количество точек на дюйм), dpcm (количество точек на сантиметр), dppx (количество точек на пиксель).

## СSS Flexbox

CSS Flexible Box Layout Module (CSS модуль для макетов с гибкими блоками), коротко flexbox, позволяет контролировать параметры элементов внутри контейнера: направление, порядок, ширину, высоту, выравнивание вдоль и поперек, распределение свободного места, растяжение и сжатие элементов. FlexBox состоит из контейнера и его дочерних элементов (items) (гибких элементов). Селектором может служить любой селектор.

**display** - определяет flex-контейнер и подключает flex-контекст для всех его непосредственных потомков. Бывает в значении flex (блочный элемент) и inline-flex (строчный).

flex - элемент растягивается на всю ширину и имеет свое полное пространство среди окружающих блоков. Происходит перенос строк в начале и в конце блока.

inline-flex - элемент обтекается другими элементами. При этом его внутренняя часть форматируется как блочный элемент, а сам элемент — как встроенный.

**flex-direction** - позволяет управлять направлением оси flex-контейнера.

row (default) - направление элементов слева направо (→)

column - направление элементов сверху вниз (↓)

row-reverse - направление элементов справа налево (←)

column-reverse - направление элементов снизу вверх (↑)

**flex-wrap** - обеспечивает многострочность во flex-контейнере. Свойство принимает следующие значения:

nowrap (значение по умолчанию) — flex-элементы размещаются в одной линии, слева направо (либо справа налево для локации RTL).

wrap — flex-элементы выстраиваются горизонтально в несколько рядов (при условии, что они не помещаются в один ряд). Направление элементов — слева направо (или справа налево для RTL).

wrap-reverse — принцип действия идентичен предыдущему свойству, с той лишь разницей, что расположение flex-элементов происходит в реверсном порядке.

**justify-content –** узказывает, каким образом будут выравниваться его дочерние элементы вдоль главной оси.

flex-start (значение по умолчанию) — flex-элементы прижимаются к началу главной оси.

flex-end — flex-элементы прижимаются к концу главной оси.

center — flex-элементы центрируются по главной оси.

space-between — первый flex-элемент находится в начале главной оси, последний flex-элемент — в ее конце, а все остальные flex-элементы равномерно распределяются в пределах оставшегося пространства.

space-around — все flex-элементы равномерно распределяются на главной оси, при этом свободное пространство поровну делится между ними.

**align-items** - выравнивание его дочерних элементов вдоль поперечной оси.

stretch (значение по умолчанию) — flex-элементы растягиваются вдоль поперечной оси (если при этом указаны свойства min-width/max-width, они принимаются во внимание).

flex-start — flex-элементы прижимаются к началу поперечной оси.

flex-end — flex-элементы прижимаются к концу поперечной оси.

center — flex-элементы центрируются по поперечной оси.

baseline — flex-элементы выравниваются по своим базовым линиям.

**align-content -** работает в том случае, если flex-контейнер поддерживает многострочность. При помощи align-content можно указать, как будут выравниваться ряды flex-элементов по вертикали.

stretch (значение по умолчанию) — ряд flex-элементов растягивается по вертикали, пока не упрется в следующий ряд (если при этом указаны свойства min-width/max-width, они принимаются во внимание).

flex-start — ряды flex-элементов прижимаются к началу flex-контейнера.

flex-end — ряды flex-элементов прижимаются к концу flex-контейнера.

center — ряды flex-элементов вертикально центрируются во flex-контейнере.

space-between — первый ряд flex-элементов находится в начале flex-контейнера, последний ряд flex-элементов — в конце, а все остальные ряды равномерно распределяются в пределах оставшегося пространства.

space-around — все ряды flex-элементов равномерно распределяются в вертикальном пространстве flex-контейнера, при этом свободное пространство поровну делится между ними.

## CSS Grid Layout

Если Flexbox позволяет оптимально размещать элементы для одного измерения, то Grid стоит использовать при создании двумерного макета.

Работа с Grid Layout похожа на Flexbox и заключается в применении специальных CSS-правил к родительскому элементу (grid-контейнеру) и к его дочерним элементам (grid-элементам).

Для начала необходимо:

* определить grid-контейнер путем добавления к нему свойства display со значением grid;
* задать размеры строк и столбцов, используя свойства grid-template-rows и grid-template-columns;
* поместить дочерние элементы в сетку при помощи свойств grid-row и grid-column.

**Компоненты Grid**

**Сетка (grid)** – это набор пересекающихся горизонтальных и вертикальных линий, делящих пространство grid-контейнера на области сетки, в которые могут быть помещены элементы сетки.

**Контейнер-сетка (grid container)** — это блок, который создает область с сеткой и является родительским для всех grid-элементов.

**display** – используется для инициализации grid-контейнера, может иметь одно из трех значений:

grid (создает блочный grid-контейнер);

inline-grid (создает строчный grid-контейнер);

subgrid (если ваш grid-контейнер сам по себе является grid-элементом, вы можете использовать это свойство, чтобы размеры его строк/столбцов были взяты из его родительского элемента).

Для выравнивания элементов используются те же свойства align-items и justify-content, что и в flexbox.

**Grid-элементы** **(grid items)** — прямые потомки grid-контейнера.

Мы определяем ряды и столбцы в нашей сетке при помощи свойств grid-template-columns и grid-template-rows. Это определяет полосы сетки. Полоса сетки — это место между любыми двумя линиями сетки. На пересечении строк и столбцов образуются ячейки.

**grid-template-columns** — определяет столбцы

**grid-template-rows** — определяет строки

В качестве значения свойствам передается ширина столбцов. Значения разделяются пробелами, количество которых равно количеству колонок или строк.

Если столбцы или строки имеют одинаковые размеры, то можно использовать специальную функцию **repeat**(), которая позволит настроить строки и столбцы:

grid-template-columns: 8em 8em 8em = grid-template-columns: repeat(3, 8em)

В качестве меры измерения в Grid Layot можно использовать знакомые px, em, rem, pt, % и auto, а также значение fr.

**fr** - специальная единица измерения, применяемая для установки пропорциональных размеров. Она представляет собой часть пространства (fraction), которое отводится для данного столбца или строки. Значение fr еще называют flex-фактором (flex factor).

Вычисление пропорциональных размеров производится по формуле:

flex-фактор \* доступное\_пространство / сумма всех flex-факторов

При этом под доступным пространством понимается все пространство grid-контейнера за исключением фиксированных значений строк и столбцов.

**grid-column-gap и grid-row-gap** - свойства, применяемые для определения ширины отступов между столбцами и строками соответственно.

Сетка представляет собой набор ячеек, которые образуются на пересечении столбцов и строк. Но сами строки и столбцы образуются с помощью grid-линий, которые рассекают сетку по вертикали и горизонтали. По умолчанию каждый элемент в гриде позиционируется в одну ячейку по порядку, но мы можем более точно настроить расположение элемента в гриде с помощью ряда свойств:

**grid-row-start**: задает начальную горизонтальную grid-линию, с которой начинается элемент

**grid-row-end**: указывает, до какой горизонтальной grid-линии надо растягивать элемент

**grid-column-start**: задает начальную вертикальную grid-линию, от которой начинается элемент

**grid-column-end**: указывает, до какой вертикальной grid-линии нужно растягивать элемент

Таким образом мы можем установить элемент, который занимает несколько ячеек и образует некоторую область в сетке.

.special-item {

grid-column-start: 2;

grid-column-end: 5;

}

Также с помощью этого свойства можно осуществить наложение элементов. В этом случае перекрывать и отображаться сверху будет тот элемент, который находится ниже в коде. Изменить порядок можно с использованием свойства z-index. Чем выше его значение, тем выше элемент будет отображаться.

**grid-template-areas**: определяет шаблон сетки ссылаясь на имена областей, которые заданы с помощью свойства grid-area. Повторение названия области приводит к тому, что содержимое охватывает эти ячейки. Точка означает пустую ячейку. Сам синтаксис предоставляет визуализацию структуры сетки.

**Например**:

.wrapper {

display: grid;

grid-template-rows: repeat(3, 1fr);

grid-template-columns: 100px 2fr 1fr;

grid-template-areas:

"header header header"

"nav main sidebar"

"footer footer footer";

}

Данный контейнер создаст три колонки, при этом размер первой будет 100 пикселей, а вторая и третья поделят оставшееся пространство между собой в отношении 2 к 1. В каждой колонке будет по три равных строки.

Чтобы использовать компоненты template-areas, нужно создать класс компонента и прописать в нем то название, которое было использовано в grid-контейнере.

.navbar{

grid-area: nav;

}