1. Какой тег используется для задания стилей на html странице?

Тег **style** в теге head.

1. Как подключить внешний файл со стилями к html странице?

Элемент <link> в HTML-документе сообщает браузеру, где находится файл css для форматирования страницы. Не нуждается в закрывающем теге. Должен использовать атрибут href (путь к css файлу) и rel (отношение между html-страницей и связанным файлом), в rel значение stylesheet.

1. Может ли css файл подключить внутри себя другой css файл? Ответ пояснить.

Для импорта одного css в другой используется конструкция **@import**. Один файл со стилями импортируется в другой файл со стилями. Указать внутри **url** путь к нужному файлу. **@import url(“custom.css”);**

Во время загрузки страницы браузер пройдет по всем указанным в импортах путям и загрузит таблицы стилей. Затем прочитает правила ниже

импортов и отрисует страницу на основании комбинации всего CSS.

1. Что такое селекторы? Приведите их основные виды.

**Это инструкции**, которые позволяют выбирать определенные html-элементы и применять к ним стили.

Универсальный селектор \*{ color:pink; } - применяется ко все элементам в документе

Селектор имени тега h1{ color:pink; } – применяется ко всем элементам с таким тегом

Селектор класса .className{ color:pink; } – применяется ко всем элементам с таким классом

Комбинированный селектор класса h2.className{ color:pink; } – применяется к элементом с указанным тегом, у которых токой класс.

Селектор идентификатора #id1{ color:pink; } - применяется ко всем элементам с таким id

Дочерний селектор li > a { color:pink; } – к элементам, являющимся прямым потомком указанного элемента

Селектор потомка li a { color:pink; } – к элементам-потомкам, не обязательно прямым

Проверка комбинированных селекторов происходит справа налево.

1. Что такое специфичность CSS правил и как она рассчитывается?

**Приоритет применения правил к элементу**. **Очки специфичности**: 0 – универсальный селектор, 1 – по имени тега, 10 – по классу, 100 – по идентификатору, 1000 – inline стили в атрибуте style.

1. Что такое конструкция !important и зачем она нужна?

Это правило для придания большей важности свойству/значению, чем обычно. **Задает наивысшую важность**, выше, чем у inline стилей. Если несколько одних и тех же свойств в разных местах имеют !important, применится то, селектор которого имеет большую специфичность.

1. Чем отличаются наследуемые и ненаследуемые свойства?

**Наследуемые упрощают таблицу стилей**, применяются к большинству дочерних элементов. Например, color. **Не наследуемые** не наследуются дочерними элементами. Например, border.

1. Как заставить ненаследуемое свойство наследовать значение родителя?

Задать дочернему элементу свойство **inherit**.

1. Как задается цвет текста и цвет фона элемента?

Свойства color и background-color. Цвет задается через шестнадцатеричный код, через rgba или через имя цвета.

1. Какие единицы измерения в CSS вы знаете? В чем их особенности?

Абсолютные – px (базовая единица измерения), и устаревшие mm, cm, pt (пункты), pc (пики)

Относительные - %, em, rem. % рассчитывается браузером по отношению к свойству родителя (100% - размер в соответствующем свойстве родителя). em - рассчитывается браузером по отношению к текущему размеру шрифта (1em = текущий размер шрифта=100%, 1.4em = 140%). rem – относительная единица, рассчитывается браузером по отношению к размеру шрифта элемента <html> (1rem = размер шрифта элемента <html>)

1. Как центрировать текст внутри блока?

Свойство **text-align** выравнивает текст внутри блоков. Положение самих этих блоков и их размер остаются неизменными. По умолчанию блочные элементы стремятся растянуться по ширине на всю страницу.

1. Как центрировать блок внутри блока?

**margin: auto**. Заставляет браузер выделять все доступное пространство по блокам.

1. Как задаются границы поля и отступы у блока? Что такое “схлопывание отступов”?

**Границы** отделяют края одного блока от краев других. Задаются свойством border. Каждый блок имеет границы, даже если они не видны или их толщина равна 0.

**Поля** находятся за пределами границ, задаются для создания интервала между границами соседних блоков. Свойство margin.

**Отступы** – это интервал между границами блока и его содержимым. Свойство padding.

**Схлопывание отступов** – если один блок находится поверх другого, то будут использоваться поля, имеющие наибольшее значение.

1. Что происходит если контент не влезает в размеры блока, в котором находится?

По умолчанию **не влезающий контент отображается вне блока**. Свойство **overflow** по умолчанию visible.

hidden – все, что не влезает в размеры блока, станет невидимым. scroll – добавится полоса прокрутки, даже если контент не вышел за пределы блока. auto – полоса прокрутки появится при необходимости. Есть также overflow-x и overflow-y. Если у x и y разные visible, то visible для первого станет auto.

1. Как установить и настроить фоновое изображения элемента?

**Background-image**: url(“image.jpg”);

Повторение **Background-repeat**: repeat; Или repeat-x, repeat-y или no-repeat.

Позиционирование **Background-position**: 25% 50%. Позиция фона относительно верхнего левого угла элемента.

Размер **Background-size**: 200px или auto. Contain – чтобы картинка накрыла весь блок, не обрезается. Cover – масштабирует изображение с сохранением пропорций, обрезается. Auto – размер фона как размер картинки.

1. Что такое псевдоэлементы? Как работают ::before и ::after?

Это ключевое слово, добавляемое к селектору, которое позволяет стилизовать определенную часть выбранного элемента.

::before и ::after позволяют добавлять содержимое (стили) до и после элемента, к которому были применены. Содержимое псевдоэлементов задается с помощью свойства content. К ним можно применять и обычные стили CSS. ::before и ::after аналогично добавлению <span> перед и после содержимого указанного элемента, в который будет помещено значение свойства content псевдоэлемента, и к которому будет применены стили, заданные в псевдоэлементе.

li::before { content: “smile” }

.new::after { content: “- Новинка”; color: red; }

1. Что такое псевдоклассы? Как работают :hover, :focus и :active?

**Псевдоклассы** - это ключевое слово, добавленное к селектору, которое позволяет указать стили для особого состояния выбранного элемента. :hover, :focus, :active меняют внешний вид элементов в ответ на действия пользователя.

**:hover** – применяется, когда пользователь наводит на элемент указатель мыши

**:focus** – применяется, когда элемент имеет фокус

**:active** - применяется, когда элемент активируется пользователем, например, нажатие кнопки или щелчок по ссылке. Применяется в промежуток между нажатием и отпусканием кнопки мыши.

1. Какие бывают виды позиционирования в CSS? В чем их отличия?

**Нормальный поток** – по умолчанию. position: static; Каждый последующий блочный элемент располагается на странице ниже, чем предыдущий.

**Относительное позиционирование** - position: relative; Положение элемента задается свойствами top, left, right, bottom относительно места, в котором он бы находился при использовании схемы нормального потока.

**Абсолютное позиционирование** - position: absolute; + **фиксированное** позиционирование position: fixed; Блок исключается из нормального потока и не влияет на положение других элементов страницы. Для них этого блока не существует. Свойства top, left, right, bottom указывают, где должен быть расположен элемент относительно контейнера (первого позиционированного элемента-предка).

**Фиксированное позиционирование** – относительно окна браузера. При прокрутке страницы элемент остается на том же месте.

1. Как решить проблему игнорирования браузером плавающих элементов при расчете размера родителя.

**Добавить в конец контейнера после последнего плавающего** элемента пустой элемент со свойством **clear: both**;

1. С помощью каких способов в CSS можно скрыть элемент?

**display:none**; - полностью исключает элемент из нормального потока static. Действует так, будто элемента нет. Содержимое блока видно при просмотре кода.

**visibility:hidden**; - элемент заменяется на пустоту эквивалентного размера, взаимодействовать с ней нельзя.

**opacity: 0**; - позволяет взаимодействовать с элементом. Текст можно скопировать, если выделить эту пустоту.

1. Что такое flexbox? Основные свойства для его настройки.

Это модуль гибкого контейнера, который представляет собой одномерную модель макета. Помогает с выравниванием, распределением и направлением элементов страницы. Чтобы сделать контейнер flexbox-ом, надо указать его свойство display: flex.

**Направление**

**flex-direction** – главная ось и направление размещения элементов контейнера. Значение row – расположить элементы в строку. column – в столбец. В обратном порядке: row-reverse, column-reverse.

**Интервал**

**gap** – отступы между строками и столбцами. Есть column-gap, есть row-gap.

**Выравнивание**

**justify-content** выравнивает элементы относительно главной оси flexbox (задаваемой flex-direction). Значение flex-start выравнивает элементы по левому краю (по умолчанию), flex-end выравнивает элементы по правому краю, center – выравнивает по центру, space-between распределяет элементы горизонтально равномерно по всей ширине, space-around распределяет элементы горизонтально равномерно по всей ширине с пробелами по краям.

**align-items** выравнивает элементы относительно побочной оси. Значение flex-start выравнивает элементы по верхнему краю, flex-end выравнивает элементы по нижнему краю, center – выравнивает по центру, baseline – выравнивает элементы по вертикали, чтобы их базовые линии совпадали, stretch растягивает дочерние элементы до высоты контейнера, стоит по умолчанию.

**align-content** похоже на align-items, но выравнивает вдоль побочной оси не элементы flex контейнера, а его линии. Значения stretch – линии растягиваются на все свободное пространство, flex-end линии располагаются к концу контейнера, space-between линии распределяются равномерно в контейнере, space-around линии равномерно распределяются в контейнере, пустое пространство перед первой строкой и после последней строки равно половине пространства между двумя соседними строками, space-evently линии равномерно распределяются таки образом, чтобы пространство между двумя соседними линиями, а также пространство перед первой и после последней линии было одинаковым.

**Перенос**

**flex-wrap** – определяет, будет ли флекс контейнер однострочным или многострочным. Значение nowrap – по умолчанию, элементы выстраиваются в одну строку. wrap – флекс элементы в случае переполнения переносятся. wrap-reverse – флекс элементы размещаются в обратном порядке.

Есть стенографическое свойство: .container{flex-direction: row-reverse; flex-wrap: wrap;} равно .container{flex-flow: row-reverse wrap;}

**Порядок**

**order** – задает порядок блока в контейнере

**flex-grow** – увеличить или уменьшить блок по отношению к остальным. Чтобы растянуть элементы на всю ширину контейнера, надо указать flex-grow для каждого блока внутри flex контейнера. По умолчанию flex-grow:0.

**flex-shrink** – задает, насколько элемент сжать относительно остальных. По умолчанию flex-shrink:1.

**flex-basis** – начальная длина элемента внутри контейнера (em, px, %). В отличие от flex-grow на распределяет пространство между элементами равномерно.

Есть стенографическое свойство flex.

#el1{ flex-grow: 2; flex-shrink: 1; flex-basis: 200px;} равно #el1{ flex: 2 1 200px;}

1. Что такое grid? Основные свойства для его настройки.

Это система разметки на основе сетки со строками и столбцами. Чтобы контейнер стал grid, надо указать его свойству display:grid(занимает всю доступную ширину) или dislay:inline-grid (устанавливает ширину чтобы вмещать содержимое).

**grid-template-columns** и **grid-template-rows** задают размеры строк и столбцов сетки. По ним вычисляется нужное количество строк и столбцов.

grid-template-columns: 200px 100px;

grid-template-rows: 100px 100px 100px;

Размер может быть auto. auto для столбцов растягивает их на все оставшееся доступное место, для строк дает минимально необходимую высоту для вмещения содержимого.

Функция **repeat** задает первым параметром количество повторений размера, вторым – сам размер.

grid-template-columns: repeat(2, 100px);

grid-template-rows: repeat(3, 100px);

Можно указать параметром функцию **calc** чтобы вычислить размер.

grid-template-columns: repeat(2, calc(200px / 2));

grid-template-rows: repeat(3, calc(100px / 3));

**calc**(100% / <количество столбцов>) – сетка займет всю доступную область

fr – единицы указывает сколько долей от общего пространства займет столбец или строка. grid-template-columns: 1fr 2fr 1fr 1fr 1fr;

**minmax** – указывается в качестве размера колонки или строки, чтобы ограничить ее размеры сверху и снизу.

grid-template-columns: 100px 100px 100px minmax(100px 300px) 100px;

**auto-fill** используется для указания размеров колонок в связке с repeat. Поместить в строку столько колонок, сколько можно с учетом их ширины, переданной вторым аргументом.

**auto-fit** работает как auto-fill, но не добавляет пустые колонки, если все непустые уместились в одну строку.

grid-template-columns: repeat(auto-fill, 125px));

**grid-auto-rows** и **grid-auto-columns** устанавливает размер строк и столбцов с не указанными размерами. По умолчанию auto.

grid-auto-rows: 70px. Если не указали высоту строки, установится высота 70px

**gap** – отступы между строками и столбцами. Есть column-gap, есть row-gap.

**grid-column-start**, **grid-column-end**, **grid-row-start**, **grid-row-end** – на какие строки и столбцы будет простираться элемент.

1. Что такое зоны grid (grid-area) и как ими пользоваться?

Это способ размещения элементов на сетке с помощью областей grid. Вместо задания размеров столбцов и строк с помощью grid-template-columns и grid-template-rows задается шаблон контейнера с помощью grid-template-areas, где указываются имена нужных областей. Далее имена присваиваются нужным элементам с помощью свойства grid-area. И элементы выстраиваются в нужном порядке на сетке.

1. Что вы знаете про выравнивание элементов/ячеек во Flex и grid?

justify-items – выравнивание дочерних элементов контейнера внутри ячеек сетки.

align-items – вертикальное выравнивание дочерних элементов контейнера внутри ячеек сетки.

justify-content – горизонтальное выравнивание самих ячеек сетки внутри контейнера

align-content - вертикальное выравнивание самих ячеек сетки внутри контейнера

justify-self – переназначает горизонтальное выравнивание (justify-items) для конкретного дочернего элемента

align-self - переназначает вертикальное выравнивание (align-items) для конкретного дочернего элемента

1. Что такое viewport?

Это видимая пользователем область веб-страницы. Ее размер зависит от устройства. Viewport контролируется с помощью тега meta:

<meta name=”viewport” content=”width=device-width, initia-scale=1.0”>

width=device-width задает ширину страницы в соответствии с шириной устройства

initia-scale – начальный уровень масштабирования при загрузке страницы

1. Как задавать различные стили в зависимости от ширины экрана браузера?

С помощью медиа-функции. В медиафункции указывается min-width и max-width. И применяются стили в зависимости от ширины экрана.

<link rel="stylesheet" media="screen and (min-width: 500px) and (max-width: 700px)" href="min500max700.css" />

1. Что такое Materialize?

Это CSS фреймворк для создания сайтов, основанный на принципах material design. Он предоставляет набор инструментов для верстки и готовые компоненты, например, сетка, карусель, адаптивное меню. Ссылки на стили materialize добавляются в head через тег link, скрипты через тег script.