1. Какие способы подключения и использования CSS стилей на странице вы знаете?

Существует три основных способа подключения CSS:

Внешняя таблица стилей: CSS-код находится в отдельном файле с расширением .css. Это самый рекомендуемый способ, так как позволяет управлять стилями для всего сайта из одного файла.

html

<head>

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

</head>

Внутренние стили (встроенные в документ): CSS-код помещается внутри тега <style> в секции <head> HTML-документа. Используется для стилей, уникальных для одной страницы.

html

<head>

<style>

body { background-color: lightblue; }

h1 { color: navy; }

</style>

</head>

Встроенные стили (inline-стили): CSS-код добавляется напрямую к HTML-элементу через атрибут style. Используется редко, так как усложняет поддержку кода и имеет наивысший приоритет, что может мешать каскаду.

html

<p style="color: red; font-size: 16px;">Этот текст красный.</p>

2. Перечислите и опишите все допустимые значения свойства transform.

Свойство transform позволяет визуально трансформировать элемент (сдвигать, поворачивать, масштабировать и т.д.).

none: Элемент не трансформируется (значение по умолчанию).

Функции трансформации:

Перемещение (Translate): translateX(tx), translateY(ty), translate(tx, ty), translate3d(tx, ty, tz). Сдвигает элемент по осям.

Пример: transform: translate(50px, 20px);

Масштабирование (Scale): scale(sx), scale(sx, sy), scaleX(sx), scaleY(sy), scale3d(sx, sy, sz). Увеличивает или уменьшает элемент.

Пример: transform: scale(1.5); (увеличит элемент в 1.5 раза)

Поворот (Rotate): rotate(angle), rotateX(angle), rotateY(angle), rotateZ(angle), rotate3d(x, y, z, angle). Поворачивает элемент на заданный угол.

Пример: transform: rotate(45deg);

Наклон (Skew): skew(ax), skew(ax, ay), skewX(angle), skewY(angle). Деформирует элемент, наклоняя его стороны.

Пример: transform: skewX(20deg);

Перспектива (Perspective): perspective(n). Задает глубину сцены для 3D-трансформаций, определяя, насколько сильно эффект 3D виден зрителю.

Пример: transform: perspective(500px) rotateY(45deg);

Можно комбинировать несколько функций в одном свойстве:

css

transform: translateX(100px) rotate(45deg) scale(0.8);

3. Какие единицы измерения в CSS3 вы знаете? Для чего используются те или иные единицы измерения? Приведите примеры.

Единицы измерения делятся на несколько типов:

Абсолютные: Не зависят от контекста. (Используются реже, в основном для печати).

px (пиксели) — базовая единица для экранов.

cm, mm, in (дюймы), pt (пункты), pc (пики).

Относительные: Зависят от других параметров (размера штата, окна и т.д.).

Относительно шрифта:

em — относительно размера шрифта родительского элемента.

rem — относительно размера шрифта корневого элемента (<html>). Лучшая практика для предсказуемости.

ex, ch — относительно высоты символа x или ширины символа 0.

Относительно области просмотра (Viewport):

vw — 1% от ширины окна браузера.

vh — 1% от высоты окна браузера.

vmin — 1% от меньшего из размеров (ширины или высоты).

vmax — 1% от большего из размеров.

Пример: width: 50vw; — ширина элемента будет равна половине ширины окна.

Проценты (%): Относительно размера родительского элемента.

Пример: width: 50%; — элемент займет половину ширины своего контейнера.

4. Какие свойства переходов в CSS3 вы знаете? Приведите примеры.

Переходы (Transitions) позволяют плавно изменять значения свойств за определенное время.

transition-property: Указывает, какое свойство CSS должно анимироваться.

Пример: transition-property: background-color, transform;

transition-duration: Определяет, сколько времени займет переход (в s или ms).

Пример: transition-duration: 0.3s;

transition-timing-function: Задает кривую скорости перехода (как изменение происходит во времени).

Пример: ease, linear, ease-in, ease-out, ease-in-out, cubic-bezier(...).

transition-delay: Определяет задержку перед началом перехода.

Пример: transition-delay: 0.1s;

Сокращенное свойство:

css

.element {

transition: all 0.3s ease-in-out 0.1s;

/\* property | duration | timing-function | delay \*/

}

5. Опишите механизм наследования и каскадирования стилей.

Наследование: Некоторые CSS-свойства (например, color, font-family, line-height) автоматически передаются от родительского элемента к дочерним. Это позволяет задавать стили для базовых элементов (например, для <body>) и наследовать их всему документу. Наследование можно контролировать с помощью значения inherit (принудительно наследовать) и initial (установить значение по умолчанию).

Каскадирование: Это алгоритм, который определяет, какие именно стили будут применены к элементу, когда на него влияет несколько правил CSS. Он работает по трем основным критериям (в порядке важности):

Важность (!important): Объявление с !important имеет наивысший приоритет.

Специфичность: Вычисляет "вес" селектора. Порядок (от низкого к высокому): \* -> теги (div) -> классы/псевдоклассы (.class, :hover) -> ID (#id) -> инлайн-стили.

Порядок следования: Если специфичность одинакова, побеждает то правило, которое объявлено в коде последним.

6. Опишите блочную модель расположения элементов.

Блочная модель (Box Model) описывает, как устроен каждый элемент на странице. Она состоит из четырех областей (от центра к краю):

Содержимое (Content): Область, где находится контент (текст, изображение).

Внутренний отступ (Padding): Прозрачное пространство между содержимым и границей.

Граница (Border): Линия, окружающая padding и content.

Внешний отступ (Margin): Прозрачное пространство за пределами границы, которое отделяет элемент от соседей.

Важно: По умолчанию свойства width и height задают размер только контента. Чтобы они включали в себя padding и border, используется свойство box-sizing: border-box; (современный и рекомендуемый подход).

7. Какие вы знаете строчные и блочные элементы? В чем различие? Как перейти от строчного элемента к блочному?

Блочные элементы (display: block):

Примеры: <div>, <h1>-<h6>, <p>, <section>, <ul>, <li>.

Особенности: Занимают всю доступную ширину родителя, начинаются с новой строки. Можно задавать width, height, все отступы (margin, padding).

Строчные элементы (display: inline):

Примеры: <span>, <a>, <strong>, <em>, <img>.

Особенности: Занимают только место, необходимое их содержимому. Не начинаются с новой строки. Нельзя задавать width, height, вертикальные margin и padding.

Различие: Главное отличие в их поведении в потоке документа и в том, какие CSS-свойства к ним можно применять.

Смена типа: Чтобы превратить строчный элемент в блочный и наоборот, используется свойство display.

display: block; — делает элемент блочным.

display: inline; — делает элемент строчным.

display: inline-block; — гибрид: элемент ведет себя как строчный (не переносится на новую строку), но ему можно задавать ширину, высоту и отступы, как блочному.

8) Что такое медиа-запрос, как он работает и для чего применяется?

Медиа-запрос (Media Query) — это технология в CSS3, которая позволяет применять разные CSS-стили в зависимости от характеристик устройства или области просмотра (viewport).

Как работает: Браузер проверяет условия, указанные в медиа-запросе (например, ширина экрана). Если условие истинно (true), то блок CSS-стилей внутри этого запроса применяется. Если ложно (false) — игнорируется.

Для чего применяется: Основа адаптивного веб-дизайна. Позволяет создавать сайты, которые оптимально отображаются на устройствах с разным размером экрана (от смартфонов до десктопов), а также учитывать ориентацию, разрешение и другие параметры.

9) Опишите структуру медиа-запроса.

Структура медиа-запроса состоит из нескольких частей:

Ключевое слово: @media (обязательно).

Тип носителя: all, screen, print и т.д. (необязательно, по умолчанию all).

Логические операторы: and, not, only, , (запятая как "или").

Медиа-функции (условия): Условия в круглых скобках, например (min-width: 768px), (orientation: landscape), (max-resolution: 300dpi).

Пример структуры:

css

/\* Ключевое слово | Условие \*/

@media (min-width: 768px) and (max-width: 1024px) {

/\* Стили для планшетов \*/

body { background-color: lightgreen; }

}

10) Какие типы носителя для медиа-запросов вы знаете? Перечислите и опишите известные типы.

all: Подходит для всех устройств. (Используется по умолчанию).

screen: Предназначен для экранов (мониторы, планшеты, смартфоны). Это самый распространенный тип.

print: Предназначен для печати и предпросмотра страницы перед печатью. Позволяет скрыть ненужные элементы (меню, кнопки) и оптимизировать страницу для бумаги.

speech: Предназначен для синтезаторов речи, которые зачитывают страницу вслух.

11) С помощью каких логических операторов можно комбинировать медиа-запросы? Приведите примеры.

and: Объединяет несколько условий. Все условия должны быть истинны.

css

/\* Применяется только для экранов И при ширине минимум 768px \*/

@media screen and (min-width: 768px) { ... }

/\* Применяется для экранов И в диапазоне ширины от 768px до 1024px \*/

@media screen and (min-width: 768px) and (max-width: 1024px) { ... }

, (запятая): Работает как логическое "ИЛИ". Стили применятся, если истинно хотя бы одно из условий.

css

/\* Применяется, если устройство в альбомной ориентации ИЛИ ширина больше 1200px \*/

@media (orientation: landscape), (min-width: 1200px) { ... }

not: Отрицает все медиа-запрос. Применяет стили, если условие ложно.

css

/\* Применяется для всех устройств, которые НЕ являются принтерами \*/

@media not print { ... }

only: Скрывает таблицу стилей от старых браузеров, которые не поддерживают медиа-запросы. Они проигнорируют стили, начинающиеся с only.

css

/\* Старые браузеры увидят 'only' и проигнорируют весь блок \*/

@media only screen and (min-width: 768px) { ... }