# 1 Вопрос

<!DOCTYPE html>

<html>

<head></head>

<body></body>

</html>

<!DOCTYPE html>. DOCTYPE (document type) — служебная информация для браузера, где описывается стандарт HTML, который нужно обработать.

<html lang="ru"> - тег содержащий в себе всю страницу html, является корнем

<head> - тег содержащий в себе информацию о странице для бразуера и содержит в себе технические, не визуальные теги

<body> - тег в котором содержится визуальный контент страницы, отображаемый у пользователя

# 2 Вопрос

теги верхнего уровня - Эти теги предназначены для формирования структуры веб-страницы и определяют раздел заголовка и тела документа. Примеры: html body head

теги заголовка документа - К этим тегам относятся элементы, которые располагаются в контейнере **<head>**. Все эти теги напрямую не отображаются в окне браузера, за исключением тега **<title>**, который определяет название веб-страницы. Примеры: title, meta

блочные элементы - Блочные элементы характеризуются тем, что занимают всю доступную ширину, высота элемента определяется его содержимым, и он всегда начинается с новой строки. Примеры: h1-h6, div, p

строчные элементы - Строчными называются такие элементы веб-страницы, которые являются непосредственной частью другого элемента, например, текстового абзаца. В основном они используются для изменения вида текста или его логического выделения. Примеры: a, b, br, em, I, img, span

универсальные элементы - Особенность этих тегов состоит в том, что они в зависимости от контекста могут использоваться как блочные или встроенные элементы. Примеры: del ( используется для выделения текста, который был удален в новой версии документа.**)**, ins (предназначен для акцентирования вновь добавленного текста и обычно применяется наряду с тегом)

списки - Списком называется взаимосвязанный набор отдельных фраз или предложений, которые начинаются с маркера или цифры. Примеры: ol, ul, li

таблицы - Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Примеры: table, td, tr, th

фреймы - *Фреймы* разделяют окно браузера на отдельные области, расположенные вплотную друг к другу. В каждую из таких областей загружается самостоятельная веб-страница определяемая с помощью тега **<frame>**. Примеры: frame, iframe

# 3 Вопрос

HTML-тег - это элемент языка разметки HTML, который используется для структурирования и отображения веб-страницы и её контента

HTML-теги могут также содержать атрибуты, которые предоставляют дополнительную информацию об элементе

# 4 Вопрос

Внутрь HTML-тега можно помещать не только текст, но и другие теги

<ul>

<li>Первый элемент списка</li>

<li>Второй элемент списка</li>

</ul>

В этом примере, <ul> - это родительский элемент, а <li> - это дочерние элементы

<div id="main">

<p>Первый абзац</p>

<p>Второй абзац</p>

</div>

Здесь, <p> являются потомками элемента с идентификатором main2

Важно помнить, что некоторые элементы в HTML требуют вложенности одних тегов в другие, но есть исключения. Например, в HTML нельзя вкладывать параграфы в параграфы

# 5 Вопрос

1. **Теги <h1> - <h6>**: Эти теги используются для структурирования заголовков HTML<h1> используется для обозначения самого важного, а <h6> - наименее важного заголовка.
2. **Теги <b> и <strong>**: Эти теги задают полужирное начертание шрифта. Тег <b> выделяет текст без акцента на его важность, тогда как тег <strong> определяет текст, которому придают особую важность.
3. **Теги <i> и <em>**: Эти теги задают курсивное начертание шрифта. Тег <i> отличает текст только визуально, тогда как тег <em> экспрессивно-эмоционально выделяет фрагмент текста.
4. **Тег <pre>**: Этот тег используется для включения в HTML-документ предварительно отформатированного текста.
5. **Тег <mark>**: Этот тег определяет выделенный / подсвеченный текст.

# 6 Вопрос

Семантика в HTML относится к использованию HTML-тегов для обеспечения смысла и контекста содержимого веб-страницы1. Семантические элементы четко описывают свое значение как для браузера, так и для разработчика2.

Примеры несемантических элементов:

<div> и <span>: Они не говорят ничего о своем содержимом2.

Примеры семантических элементов:

<form>, <table>, и <article>: Они четко определяют свое содержание2.

Семантические теги HTML5 предоставляют дополнительную информацию о типе содержимого, что помогает браузерам и поисковым системам лучше понимать и интерпретировать содержимое3.

Основные семантические теги HTML5:

<article>: Определяет независимое, автономное содержимое.

<aside>: Определяет контент, слабо связанный с остальным содержимым страницы.

<details>: Определяет дополнительные детали, которые пользователь может просмотреть или скрыть.

<figcaption>: Определяет подпись к <figure>.

<figure>: Определяет самостоятельный контент, например иллюстрацию, диаграмму, фотографию, код и т.д.

<footer>: Определяет подвал для документа или раздела.

<header>: Определяет заголовок для документа или раздела.

<main>: Определяет основное содержимое документа.

<nav>: Определяет навигационные ссылки.

<section>: Определяет раздел в документе.

# 7 Вопрос

Блочные элементы23:

Начинаются с новой строки.

Занимают всю ширину экрана.

Свойства width и height будут применяться.

Внешние и внутренние отступы, рамка будут отодвигать от него другие элементы.

Примеры: параграфы, списки, <pre>, <div>, заголовки, таблицы3.

Строчные элементы31:

Не начинаются с новой строки.

Занимают только то пространство, которое им необходимо.

Свойства width и height не будут применяться.

Вертикальные внешние и внутренние отступы, рамки будут применяться, но не будут отодвигать другие строчные элементы.

Примеры: ссылки (<a>), теги начертания текста (<b>, <strong>, <i>, <em>), картинки, <span>3.

Изменение поведения элементов4:

Чтобы блочный элемент вел себя как строчный, для него необходимо записать правило: display: inline;

Если же необходимо отобразить строчный элемент как блочный, запишите следующее: display: block;

# 8 Вопрос

Гиперссылки в HTML создаются с помощью тега <a>, который определяет ссылку. Основные атрибуты этого тега:

href: Этот атрибут содержит URL-адрес ссылки1234. Это может быть ссылка на другую веб-страницу или на определенный раздел на той же странице2.

target: Этот атрибут определяет, где открывать связанный документ5. Например, значение \_blank открывает документ в новом окне или вкладке5.

title: Этот атрибут предоставляет дополнительную информацию о ссылке, которая отображается при наведении курсора6.

id: Этот атрибут используется для создания закладок на странице5. Значение атрибута должно быть уникальным6.

Помимо этих атрибутов, тег <a> может содержать другие атрибуты, такие как class, style и т.д., для дополнительной настройки ссылки6.

# 9 Вопрос

Списком называется взаимосвязанный набор отдельных фраз или предложений, которые начинаются с маркера или цифры. Списки предоставляют возможность упорядочить и систематизировать разные данные и представить их в наглядном и удобном для пользователя виде.

<ol>

Тег <ol> устанавливает нумерованный список, т.е. каждый элемент списка начинается с числа или буквы и увеличивается по нарастающей.

<ul>

Устанавливает маркированный список, каждый элемент которого начинается с небольшого символа — маркера.

<li>

Тег <li> определяет отдельный элемент списка. Внешний тег <ul> или <ol> устанавливает тип списка — маркированный или нумерованный.

# 10 Вопрос

Правило @keyframes в CSS используется для создания анимаций12. Оно позволяет определить стили для ключевых кадров (или промежуточных этапов) в последовательности анимации12.

Вот базовый синтаксис:

@keyframes имя-анимации {

ключевой-селектор {

css-стили;

}

}

имя-анимации: Это имя, которое вы выбираете для анимации.

ключевой-селектор: Это процентное значение, которое указывает, когда должен произойти ключевой кадр. Вы можете использовать ключевые слова from (эквивалент 0%) и to (эквивалент 100%).

css-стили: Это стили, которые будут применены в этом ключевом кадре.

# 11 Вопрос

Селекторы CSS используются для “поиска” (или выбора) HTML-элементов, которые вы хотите стилизовать12. Они бывают достаточно сложными, указывая не просто конкретный элемент, но и место, где он находится или внутри какого тега34.

Основные типы селекторов:

1. Селекторы по типу элемента: Выбирают тип элементов, к которым будет применяться правило1. Например, селектор input выберет все элементы <input>.
2. Селекторы по классу: Выбирают элементы, основываясь на значении их атрибута class. Например, селектор .index выберет все элементы с соответствующим классом.
3. Селекторы по идентификатору: Выбирают элементы, основываясь на значении их id атрибута. Например, селектор #toc выберет элемент с идентификатором toc.
4. Селекторы по атрибуту: Выбирают все элементы, имеющие данный атрибут или атрибут с определённым значением. Например, селектор [autoplay] выберет все элементы, у которых есть атрибут autoplay.

Комбинаторы в CSS - это символы, которые определяют отношения между селекторами. Основные комбинаторы включают:

1. Комбинатор потомков (пробел): Выбирает элементы, которые находятся внутри указанного элемента. Например, селектор div span выберет все элементы <span>, которые находятся внутри элемента <div>.
2. Дочерние селекторы (>): Выбирает только те элементы, которые являются дочерними непосредственно по отношению к указанному элементу5. Например, селектор ul > li выберет только дочерние элементы <li>, которые находятся внутри, на первом уровне вложенности по отношению к элементу <ul>.
3. Комбинатор всех соседних элементов (~): Выбирает элементы, которые находятся на этом же уровне вложенности, после указанного элемента, с тем же родителем. Например, p ~ span выберет все элементы <span>, которые находятся после элемента <p> внутри одного родителя5.
4. Комбинатор следующего соседнего элемента (+): Выбирает элемент, который находится непосредственно после указанного элемента, если у них общий родитель5. Например, p + img выберет все элементы <img>, которые идут сразу после элементов <p>.

# 12 Вопрос

Переменные в CSS используются для хранения конкретных значений, которые можно повторно использовать в документе. Они объявляются с использованием двойного дефиса и доступны через функцию var(). Например:

:root {

--main-color: black;

}

element {

color: var(--main-color);

}

В этом примере, --main-color - это переменная, которая хранит значение black. Затем мы используем функцию var() для применения этого значения к свойству color элемента.

Функция calc() в CSS позволяет выполнять простые вычисления для значений свойств CSS2. Она может быть использована везде, где применимы <length>, <frequency>, <angle>, <time>, <number>, или <integer>. Например:

.main-content {

/\* Вычесть 80px из 100vh \*/

height: calc(100vh - 80px);

}

# 13 Вопрос

Технология CSS Flexbox Layout предоставляет эффективный способ распределения пространства между элементами интерфейса, включая возможность выравнивания и распределения этих элементов. Flexbox упрощает создание сложных макетов и облегчает работу с макетами.

Основные концепции Flexbox:

1. **Flex контейнер**: Элемент, для которого задано свойство display: flex или display: inline-flex. Все дочерние элементы становятся flex элементами.
2. **Главная ось и побочная ось**: Flex элементы располагаются вдоль этих осей. Главная ось определяется свойством flex-direction.
3. **Flex элементы**: Дочерние элементы flex контейнера. Можно управлять их размерами, порядком и выравниванием.

Важные свойства Flexbox:

1. flex-direction: Определяет главную ось (row или column).
2. flex-wrap: Управляет переносом элементов на новую строку.
3. flex-flow: Сокращенное свойство для flex-direction и flex-wrap.
4. justify-content: Выравнивает элементы вдоль главной оси.
5. align-items: Выравнивает элементы вдоль побочной оси.
6. align-content: Выравнивает строки flex контейнера вдоль побочной оси.

Flexbox значительно упрощает создание адаптивных макетов и предоставляет больше гибкости, чем традиционные методы CSS.

# 14 Вопрос

Технология CSS Grid Layout представляет собой двумерную систему сеток в CSS, которая подходит как для верстки основных областей страницы, так и для небольших элементов пользовательского интерфейса.

Основные понятия CSS Grid Layout:

Grid контейнер: Элемент, для которого задано свойство display: grid или display: inline-grid. Все дочерние элементы становятся grid элементами.

Grid элементы: Дочерние элементы grid контейнера. Можно управлять их размерами, порядком и выравниванием.

Ячейки Grid: Пространство между четырьмя grid линиями. Grid элементы занимают одну или несколько таких ячеек.

Grid линии: Вертикальные или горизонтальные линии, которые разделяют пространство grid на ячейки.

Важные свойства CSS Grid Layout:

grid-template-columns и grid-template-rows: Определяют структуру grid — количество и размеры колонок и строк.

grid-column и grid-row: Определяют, на каких линиях grid должен располагаться элемент.

grid-area: Определяет местоположение элемента в grid и/или дает элементу имя.

justify-items, align-items, justify-content, align-content: Выравнивают элементы внутри grid контейнера.

# 15 Вопрос

Правила построения CSS-стиля:

CSS-стиль состоит из набора правил, каждое из которых включает в себя селектор и блок объявлений.

Селектор определяет, к каким элементам HTML применяется стиль.

Блок объявлений содержит одно или несколько объявлений, разделенных точкой с запятой.

Каждое объявление состоит из свойства CSS и его значения, разделенных двоеточием.

Правило @import в CSS-стиле:

Правило @import в CSS позволяет импортировать стили из другого CSS-файла.

# 16 Вопрос

Создание CSS-анимации:

CSS-анимация создается с помощью правила @keyframes, которое определяет изменение стилей от начального до конечного состояния1. Свойство animation применяется к элементу, который вы хотите анимировать, и настраивает детали анимации.

Использование свойства transition в CSS-анимации

Свойство transition в CSS используется для создания плавного перехода между двумя состояниями элемента. Оно позволяет контролировать скорость изменения свойства в течение заданного времени.

# 17 Вопрос

Селектор потомков (пробел): Выбирает элементы, которые находятся внутри указанного элемента. Например, селектор div span выберет все элементы <span>, которые находятся внутри элемента <div>.

Дочерние селекторы (>): Выбирает только те элементы, которые являются дочерними непосредственно по отношению к указанному элементу. Например, селектор ul > li выберет только дочерние элементы <li>, которые находятся внутри, на первом уровне вложенности по отношению к элементу <ul>.

# 18 Вопрос

Псевдоклассы в CSS - это ключевые слова, добавляемые к селектору, которые определяют его особое состояние. Например, :hover может быть использован для изменения цвета кнопки при наведении курсора на неё. Псевдоклассы позволяют стилизовать элемент на основе не только отношений в DOM-дереве, но и основываясь на внешних факторах, таких как история посещений (например, :visited), состояние содержимого (вроде :checked у некоторых элементов формы) или позиции курсора мыши (например, :hover определяет, находится ли курсор мыши над элементом)1.

Псевдоэлементы в CSS - это ключевые слова, добавляемые к селектору, которые позволяют стилизовать определённую часть выбранного элемента. Например, псевдоэлемент ::first-line может быть использован для изменения шрифта первой строки абзаца. В отличие от псевдоклассов, псевдоэлементы могут быть использованы для стилизации определённой части элемента.

# 19 Вопрос

Модуль CSS Box Alignment Level 3 содержит функции CSS, относящиеся к выравниванию блоков внутри их контейнеров в различных моделях компоновки блоков CSS: блочная компоновка, таблица, flex и grid. Модуль создает единый метод выравнивания по всему CSS.

Важные элементы, содержащиеся в CSS Box Alignment Module Level 3, включают:

Выравнивание блока: Определяет, как блоки выравниваются по вертикали.

Выравнивание по горизонтали: Определяет, как блоки выравниваются по горизонтали.

Выравнивание по оси: Определяет, как блоки выравниваются по оси.

Выравнивание само-позиционированных элементов: Определяет, как само-позиционированные элементы выравниваются.

Выравнивание элементов с переполнением: Определяет, как выравниваются элементы, которые переполняют свой контейнер.

# 20 Вопрос

Модуль CSS Box Sizing Level 3 определяет, как вычисляется общая ширина и высота элемента. Он включает в себя свойства width и height, которые определяют размеры контента, а также свойства padding, border и margin, которые определяют размеры других областей.

Основные элементы, содержащиеся в CSS Box Sizing Module Level 3:

Свойство box-sizing: Определяет, как вычисляется общая ширина и высота элемента. Оно может принимать два значения:

content-box: Это значение по умолчанию. Свойства width и height включают только контент, и не включают padding и border.

border-box: Свойства width и height включают контент, padding и border, но не включают margin.

Свойства width и height: Они определяют размеры контента.

Свойства padding, border и margin: Они определяют размеры других областей.

# 21 Вопрос

Модуль CSS Display Module Level 3 описывает, как создается CSS форматирующее дерево блоков из дерева элементов документа, и определяет свойство display, которое его контролирует.

Основные элементы, содержащиеся в CSS Display Module Level 3:

<display-outside>: Определяет внешний тип отображения элемента. Может принимать значения block, inline или run-in.

<display-inside>: Определяет внутренний тип отображения элемента. Может принимать значения flow, flow-root, table, flex, grid или ruby.

<display-listitem>: Определяет, является ли элемент пунктом списка.

<display-internal>: Определяет внутренний тип отображения элемента в зависимости от его роли в составе другого элемента.

<display-box>: Определяет, генерируется ли для элемента блок.

<display-legacy>: Определяет предварительно составленные значения отображения

# 22 Вопрос

Модуль CSS Cascading and Inheritance Level 4 описывает, как собирать стилевые правила и присваивать значения всем свойствам на всех элементах. С помощью каскадирования и наследования значения распространяются на все свойства всех элементов.

В этом уровне новые элементы - это ключевое слово revert и <supports-condition> для правила @import1. Одним из основных принципов дизайна CSS является каскадирование правил.

# 23 Вопрос

Модуль CSS Color Adjustment Level 1 вводит модель и контролирует автоматическую коррекцию цвета пользовательским агентом для обработки пользовательских предпочтений, таких как “Темный режим”, коррекция контрастности или определенные желаемые цветовые схемы.

Этот модуль вводит три новых элемента, связанных с контролем того, как и когда цвета автоматически корректируются пользовательским агентом:

Свойство color-scheme, которое контролирует, учитывают ли части пользовательского интерфейса, предоставляемые браузером, выбор пользователя.

Свойство forced-color-adjust, которое контролирует, разрешено ли применение к данному элементу.

Свойство print-color-adjust, которое контролирует, разрешено ли браузеру автоматически корректировать цвета в соответствии с предполагаемыми предпочтениями пользователя, например, подавлять цвета фона при печати для экономии чернил.

# 24 Вопрос

Ответ в вопросе 21

# 25 Вопрос

Модуль CSS Flexible Box Level 1, или коротко flexbox, оптимизирует модель CSS для дизайна пользовательского интерфейса. В модели flex layout, дочерние элементы flex контейнера могут быть расположены в любом направлении и могут “гибко” менять свои размеры, либо растягиваясь, чтобы заполнить неиспользуемое пространство, либо сжимаясь, чтобы избежать переполнения родителя.

С помощью flexbox, расположение элементов на странице, виджете, приложении или галерее становится почти простым. Виджеты, карусели, адаптивные функции - что бы ваш дизайнер не придумал, будет легко реализовать.

Оба горизонтальное и вертикальное выравнивание дочерних элементов можно легко манипулировать. Вложение этих блоков (горизонтальное внутри вертикального или вертикальное внутри горизонтального) может быть использовано для создания макетов в двух измерениях.

# 26 Вопрос

Модуль CSS Fonts Level 4 расширяет возможности CSS Fonts 3, определяющий правило @font-face и поддержку функций OpenType. CSS Fonts 4 добавляет поддержку переменных шрифтов OpenType и предоставляет дизайнерам таблиц стилей способ стилизации цветов в хроматических (многоцветных) шрифтах.

Этот модуль определяет изменения к существующей спецификации CSS Fonts 3 вместе с дополнительными функциями. CSS - это язык для описания отображения структурированных документов (таких как HTML и XML) на экране, на бумаге и т.д.

В этом модуле представлены следующие новые элементы:

Свойство font-language-override.

Свойство font-palette и правило '@font-palette-values'.

Дескриптор font-language-override.

Пожалуйста, обратите внимание, что эти функции находятся под угрозой и могут быть удалены во время периода CR.

# 27 Вопрос

Модуль CSS Grid Layout Level 1 определяет двухмерную систему компоновки на основе сетки, оптимизированную для дизайна пользовательского интерфейса. В модели компоновки сетки, дочерние элементы контейнера сетки могут быть размещены в произвольных слотах в предопределенной гибкой или фиксированной сетке.

Все прямые дети контейнера сетки автоматически становятся элементами сетки. Линии между столбцами называются столбцовыми линиями. Линии между строками называются строковыми линиями.

# 28 Вопрос

Модуль CSS Grid Layout Level 2 определяет двухмерную систему компоновки на основе сетки, оптимизированную для дизайна пользовательского интерфейса. В модели компоновки сетки, дочерние элементы контейнера сетки могут быть размещены в произвольных слотах в предопределенной гибкой или фиксированной сетке.

Уровень 2 расширяет Grid, добавляя возможности “подсетки” для вложенных сеток, чтобы участвовать в размере их родительских сеток. Это позволяет содержимому обеих сеток выравниваться.

В этом модуле представлены следующие новые элементы:

Подсетки: В некоторых случаях может потребоваться, чтобы содержимое нескольких элементов сетки выравнивалось друг с другом. Контейнер сетки, который сам является элементом сетки, может отложить определение своих строк и столбцов на свой родительский контейнер сетки, делая его подсеткой.

Символы подсетки: Они особенно интересны, потому что они иллюстрируют различия между подсеткой и ее родительской сеткой.

Контроль соотношения сторон и соответствие также определены в черновике и стоит прочитать.

# 29 Вопрос

Модуль CSS Logical Properties and Values Level 1 вводит логические свойства и значения, которые предоставляют возможность контролировать компоновку через логические, а не физические, направления и отображения размеров. Этот модуль также определяет логические свойства и значения для свойств, ранее определенных в CSS 2.1.

Логические свойства определяют эквиваленты своих соответствующих физических свойств, относительно направления. Новые свойства и значения в CSS были изначально разработаны только с физическими координатами в своих контролях. Этот модуль определяет новые эквиваленты, относительные к потоку, для многих значений и свойств.

Некоторые физические свойства теперь имеют логические эквиваленты. Свойства, которые принимают физические значения (верх, низ, лево, право), теперь также принимают логические значения, относительные к потоку (block-start, block-end, inline-start, inline-end).

В этом модуле представлены следующие новые элементы:

Свойства для размеров блока: block-size, inline-size, max-block-size, max-inline-size, min-block-size, min-inline-size.

Свойства для отступов: margin (логический экспериментальный ключевой), margin-block, margin-block-end, margin-block-start, margin-inline, margin-inline-end, margin-inline-start.

Свойства для отступов: padding (логический экспериментальный ключевой), padding-block, padding-block-end, padding-block-start, padding-inline, padding-inline-end, padding-inline-start.

Свойства для границ: border-block, border-block-color, border-block-end, border-block-end-color, border-block-end-style, border-block-end-width, border-block-start, border-block-start-color, border-block-start-style, border-block-start-width, border-block-style, border-block-width, border-color, border-inline, border-inline-color, border-inline-end, border-inline-end-color, border-inline-end-style, border-inline-end-width, border-inline-start, border-inline-start-color, border-inline-start-style, border-inline-start-width, border-inline-style, border-inline-width, border-style, border-width.

Свойства для радиуса границы: border-radius, border-start-start-radius, border-start-end-radius, border-end-start-radius, border-end-end-radius.

Свойства для позиционирования: inset, inset-block, inset-block-end, inset-block-start, inset-inline, inset-inline-end, inset-inline-start.

Свойства для ограничения размера.

# 30 Вопрос

Модуль CSS Positioned Layout Level 3 содержит функции CSS уровня 3, относящиеся к позиционированию и укладке элементов. Он включает и расширяет функциональность CSS уровня 2, которая строится на основе CSS уровня 1.

Этот модуль определяет координатные схемы позиционирования и смещения CSS: относительное позиционирование, липкое позиционирование, абсолютное позиционирование и фиксированное позиционирование

В этом модуле представлены следующие новые элементы:

Свойства для позиционирования: top, right, bottom, left, inset-block-start, inset-inline-start, inset-block-end, и inset-inline-end.

Сокращенные свойства для позиционирования: inset-block, inset-inline, и inset.

Относительное позиционирование.

Липкое позиционирование.

# 31 Вопрос

Модуль CSS Text Decoration Level 3 содержит функции CSS, относящиеся к декорированию текста, такие как подчеркивания, тени текста и знаки ударения

Этот модуль добавляет возможность изменять цвет, стиль, позицию и непрерывность этих декораций, а также вводит знаки ударения, традиционно используемые в Восточной типографии.

В этом модуле представлены следующие новые элементы:

Сокращенное свойство text-decoration для установки внешнего вида декоративных линий на текст. Оно является сокращением для text-decoration-line, text-decoration-color, text-decoration-style, и нового свойства text-decoration-thickness.

Декорации текста рисуются через дочерние текстовые элементы.

# 32 Вопрос

Модуль CSS Text Level 3 определяет свойства для манипуляции с текстом и определяет их модель обработки. Он охватывает разрывы строк, выравнивание и оправдание, обработку пробелов и преобразование текста.

Этот модуль включает и расширяет функциональность CSS уровня 2, которая строится на основе CSS уровня 1. Он определяет свойства для манипуляции с текстом и определяет их модель обработки.

В этом модуле представлены следующие новые элементы:

Свойства для манипуляции с текстом: line breaking, justification and alignment, white space handling, и text transformation.

Он охватывает разрывы строк, выравнивание и оправдание, обработку пробелов и преобразование текста.

# 33 Вопрос

Модуль CSS Transforms Level 2 расширяет возможности CSS Transforms Level 1, добавляя новые функции преобразования и свойства для трехмерных преобразований, а также удобные функции для простых преобразований. CSS Transforms позволяют элементам, стилизованным с помощью CSS, преобразовываться в двухмерном или трехмерном пространстве.

В этом модуле представлены следующие новые элементы:

Новые функции преобразования и свойства для трехмерных преобразований.

Удобные функции для простых преобразований.

# 34 Вопрос

Модуль CSS Transitions Level 1 предоставляет способ контролировать скорость анимации при изменении свойств CSS. Вместо того, чтобы изменения свойств происходили мгновенно, вы можете заставить изменения в свойстве происходить в течение определенного периода времени. Например, если вы изменяете цвет элемента с белого на черный, обычно изменение происходит мгновенно.

Этот модуль определяет API для переходов представления, вместе с соответствующими свойствами и псевдо-элементами, которые позволяют разработчикам создавать анимированные визуальные переходы, представляющие изменения в состоянии документа.

# 35 Вопрос

Модуль CSS Animations Level 1 предоставляет способ контролировать скорость анимации при изменении свойств CSS. Вместо того, чтобы изменения свойств происходили мгновенно, вы можете заставить изменения в свойстве происходить в течение определенного периода времени. Например, если вы изменяете цвет элемента с белого на черный, обычно изменение происходит мгновенно.

Этот модуль определяет API для переходов представления, вместе с соответствующими свойствами и псевдо-элементами, которые позволяют разработчикам создавать анимированные визуальные переходы, представляющие изменения в состоянии документа.

В этом модуле представлены следующие новые элементы:

Свойство animation, которое является сокращением для animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, animation-direction, animation-fill-mode, animation-play-state, и animation-timeline.

Ключевые кадры, которые содержат стили, которые элемент будет иметь в определенное время.

Правило @keyframes, которое определяет стили CSS внутри него, анимация будет постепенно изменяться от текущего стиля к новому стилю в определенное время.

# 36 Вопрос

Модуль CSS Writing Modes Level 4 определяет поддержку CSS для различных режимов написания и их комбинаций, включая текст с направлением слева направо и справа налево, а также горизонтальные и вертикальные ориентации. CSS - это язык для описания отображения структурированных документов (таких как HTML и XML) на экране, на бумаге и т.д.

В этом модуле представлены следующие новые элементы:

Свойство writing-mode, которое определяет, располагаются ли строки текста горизонтально или вертикально, а также направление, в котором блоки прогрессируют.

Ключевые значения: horizontal-tb, vertical-rl, vertical-lr, sideways-rl, sideways-lr.

Глобальные значения: inherit, initial, unset.

Связанные свойства: direction, unicode-bidi, text-orientation, text-combine-upright.

# 37 Вопрос

Модуль CSS Filter Effects Level 1 предоставляет способ обработки отображения элемента перед его отображением в документе. Обычно отображение элемента с помощью CSS или SVG можно описать так, как если бы элемент, включая его дочерние элементы, был нарисован в буфер (например, растровое изображение), а затем этот буфер совмещался с родительским элементом.

Этот модуль определяет API для переходов представления, вместе с соответствующими свойствами и псевдо-элементами, которые позволяют разработчикам создавать анимированные визуальные переходы, представляющие изменения в состоянии документа.

В этом модуле представлены следующие новые элементы:

Свойства backdrop-filter и filter.

Функции blur(), brightness(), contrast(), drop-shadow(), grayscale(), hue-rotate(), invert(), opacity(), saturate(), и sepia().

# 38 Вопрос

Модуль CSS Media Queries Level 4 описывает механизм и синтаксис медиа-запросов, типов медиа и медиа-функций. Он расширяет и заменяет функции, определенные в Media Queries Level 3. Медиа-запросы позволяют авторам тестировать и запрашивать значения или функции пользовательского агента или дисплейного устройства, независимо от отображаемого документа.

Они используются в CSS-правиле @media для условного применения стилей к документу и в различных других контекстах и языках, таких как HTML и JavaScript. CSS - это язык для описания отображения структурированных документов (таких как HTML и XML) на экране, на бумаге и т.д.

# 39 Вопрос

Модуль CSS Selectors Level 3 описывает механизм и синтаксис селекторов, типов медиа и медиа-функций. Он расширяет и заменяет функции, определенные в CSS2. Селекторы позволяют авторам тестировать и запрашивать значения или функции пользовательского агента или дисплейного устройства, независимо от отображаемого документа.

Они используются в CSS-правиле @media для условного применения стилей к документу и в различных других контекстах и языках, таких как HTML и JavaScript.

В этом модуле представлены следующие новые элементы:

Свойства для манипуляции с текстом: line breaking, justification and alignment, white space handling, и text transformation.

Он охватывает разрывы строк, выравнивание и оправдание, обработку пробелов и преобразование текста.

# 40 Вопрос

Свойство innerHTML в JavaScript позволяет получить или изменить ВНУТРЕННЕЕ HTML-содержимое элемента. Вот как это работает:

1. **Получение HTML-содержимого элемента:**

let html = document.getElementById("myElement").innerHTML;

В этом примере мы получаем HTML-содержимое элемента с id “myElement”.

1. **Изменение HTML-содержимого элемента:**

document.getElementById("myElement").innerHTML = "Новое содержимое";

В этом примере мы заменяем HTML-содержимое элемента с id “myElement” на “Новое содержимое”.

# 41 Вопрос

Метод addEventListener() в JavaScript позволяет добавить обработчик событий к элементу. Вот пример его использования:

element.addEventListener("click", function() {

console.log("Элемент был нажат!");

});

В этом примере мы добавляем обработчик событий “click” к элементу, и когда на элемент нажимают, в консоль выводится сообщение “Элемент был нажат!”.

Вот более подробное объяснение синтаксиса addEventListener():

target.addEventListener(type, listener [, options]);

* target: Это может быть Element, Document, Window или любой другой объект, поддерживающий события.
* type: Чувствительная к регистру строка, представляющая тип обрабатываемого события.
* listener: Объект, который принимает уведомление, когда событие указанного типа произошло. Это должен быть объект, реализующий интерфейс EventListener или просто функция JavaScript.
* options: Необязательный объект, который определяет характеристики объекта, прослушивающего событие. Он может включать следующие параметры:
  + capture: Boolean, указывает, что события этого типа будут отправлены зарегистрированному обработчику listener перед отправкой на EventTarget, расположенный ниже в дереве DOM.
  + once: Boolean, указывает, что обработчик должен быть вызван не более одного раза после добавления. Если true, обработчик автоматически удаляется при вызове.
  + passive: Boolean, указывает, что обработчик никогда не вызовет preventDefault(). Если всё же вызов будет произведён, браузер должен игнорировать его и генерировать консольное предупреждение.

Пожалуйста, будьте осторожны при использовании addEventListener(), так как неправильное использование может привести к непредвиденным результатам.

# 42 Вопрос

Тип данных Symbol в JavaScript представляет собой уникальный и неизменяемый примитивный тип данных, который может быть использован в качестве ключа для свойства объекта. Вот как вы можете создать и использовать символ:

let sym = Symbol("мой символ");

let obj = {};

obj[sym] = "Значение";

console.log(obj[sym]); // Выводит: "Значение"

В этом примере мы создаем новый символ sym и используем его в качестве ключа для свойства объекта obj. Затем мы выводим значение этого свойства.

Символы полезны, когда вы хотите создать уникальные идентификаторы для свойств объекта. Они также используются для объявления специальных методов в спецификации JavaScript.

Однако, стоит отметить, что свойства объекта, ключами которых являются символы, не перечисляются при использовании цикла for...in и не включаются в результат вызова Object.keys(obj). Для доступа к таким свойствам можно использовать Object.getOwnPropertySymbols(obj).

# 43 Вопрос

Метод Object.assign() в JavaScript используется для копирования значений всех собственных перечисляемых свойств из одного или более исходных объектов в целевой объект. Вот как это работает:

let target = {a: 1};

let source1 = {b: 2};

let source2 = {c: 3};

Object.assign(target, source1, source2);

console.log(target); // {a: 1, b: 2, c: 3}

В этом примере мы копируем свойства из source1 и source2 в target. Обратите внимание, что свойства в целевом объекте будут перезаписаны свойствами из исходных объектов, если они имеют одинаковые ключи.

# 44 Вопрос

Обработка браузерных событий мыши и клавиатуры включает в себя следующие основные аспекты:

События мыши: События мыши включают в себя события, такие как click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter и mouseleave.

События клавиатуры: События клавиатуры включают в себя события, такие как keydown, keypress и keyup.

События указателя: События указателя предназначены для создания единой модели обработки действий мыши, пера/стилуса или касания. Они включают в себя события, такие как pointerdown, pointermove, pointerup, pointerover, pointerout, pointerenter и pointerleave.

# 45 Вопрос

Событие onclick в JavaScript срабатывает при клике левой кнопкой мыши на элементе. Это событие часто используется в связке с HTML-элементами, такими как кнопки.

# 46 Вопрос

Объектная модель браузера (Browser Object Model, BOM) в JavaScript описывает все объекты, через которые JavaScript взаимодействует с браузером. BOM предоставляет набор свойств и методов, позволяющих получить доступ непосредственно к браузеру.

Вот некоторые ключевые объекты в BOM:

window: Глобальный объект JavaScript, который представляет окно браузера. Он содержит свойства и методы для работы с браузером и его компонентами.

location: Объект, который содержит информацию о текущем URL и позволяет JavaScript перенаправлять браузер на новый URL.

history: Объект, который содержит историю сессии браузера для текущей вкладки.

navigator: Объект, который содержит информацию о браузере.

document: Объект, который представляет документ, загруженный в окне браузера. Это входная точка для работы с DOM (Document Object Model).

# 47 Вопрос

Метод querySelectorAll() в JavaScript возвращает статический (не динамический) NodeList, содержащий все найденные элементы документа, которые соответствуют указанному CSS-селектору.

elementList = document.querySelectorAll(selectors);

selectors: Строка DOMString, содержащая один или более CSS-селекторов. Эта строка должна быть валидным CSS-селектором. Если это не так, то генерируется SyntaxError.

Возвращаемое значение - это статический (non-live) NodeList, содержащий все элементы в пределах документа, которые соответствуют как минимум одному из указанных селекторов, или пустой NodeList в случае отсутствия совпадений.

# 48 Вопрос

Метод closest() в JavaScript возвращает ближайший родительский элемент (или сам элемент), который соответствует заданному CSS-селектору. Если таких элементов нет, метод возвращает null.

# 49 Вопрос

document.addEventListener('mousemove', function(e) {

console.log('Координаты мыши: ', e.clientX, e.clientY);

});

# 50 Вопрос

Оператор in в JavaScript возвращает true, если свойство содержится в указанном объекте или в его цепочке прототипов. Синтаксис оператора in выглядит следующим образом:

# 51 Вопрос

Свойства firstChild и lastChild в JavaScript позволяют получить первый и последний дочерний узел элемента соответственно.

Важно отметить, что свойства firstChild и lastChild могут возвращать не только элементы, но и другие типы узлов, такие как текстовые узлы и комментарии.

# 52 Вопрос

Событие window.onload в JavaScript используется как указатель, что веб-страница полностью загружена, включая содержание, изображения, стилевые файлы и внешние скрипты. Это событие часто используется для запуска скриптов, которые необходимо выполнить после полной загрузки страницы.

# 53 Вопрос

События change, input и submit в JavaScript используются для обработки различных действий пользователя в браузере.

1. **Событие change**: Это событие срабатывает при изменении значения элемента формы и потере фокуса этим элементом. Обычно используется с элементами формы, такими как <input>, <select> и <textarea>.
2. **Событие input**: Это событие срабатывает каждый раз, когда значение <input> или <textarea> изменяется. В отличие от события change, оно срабатывает немедленно после изменения значения, а не после потери фокуса.
3. **Событие submit**: Это событие срабатывает при отправке формы. Оно обычно используется для валидации формы перед её отправкой на сервер.

# 54 Вопрос

Создание анимации в JavaScript может быть выполнено различными способами. Один из наиболее распространенных подходов - использование window.requestAnimationFrame(). Эта функция говорит браузеру, что вы хотите выполнить анимацию и просит его вызвать указанную функцию перед следующей перерисовкой.

Вот базовый пример анимации с использованием requestAnimationFrame():

let start;

function step(timestamp) {

if (start === undefined)

start = timestamp;

const elapsed = timestamp - start;

// Этот код будет выполняться каждые 60 кадров в секунду

// Если elapsed > 1000 мс, то прошла 1 секунда и можно обновить анимацию

if (elapsed > 1000) { // Если прошла 1 секунда

// Здесь можно обновить анимацию

console.log('1 секунда прошла');

// Сбросить таймер

start = timestamp;

}

window.requestAnimationFrame(step);

}

window.requestAnimationFrame(step);

В этом примере функция step() вызывается при каждой перерисовке браузером. Внутри функции step() вы можете обновлять свою анимацию.

# 55 Вопрос

<!DOCTYPE html>

<html>

<body>

<canvas id="myCanvas" width="500" height="500" style="border:1px solid #d3d3d3;">

Ваш браузер не поддерживает элемент HTML5 canvas.

</canvas>

<script>

let canvas = document.getElementById("myCanvas");

let ctx = canvas.getContext("2d");

ctx.fillStyle = "#FF0000";

ctx.fillRect(0, 0, 80, 80);

</script>

</body>

</html>

В этом примере мы создаем холст с помощью тега <canvas>, а затем используем JavaScript для рисования квадрата на холсте. Метод getContext("2d") возвращает контекст рисования на холсте и используется для рисования на нем.

# 56 Вопрос

Спецификация ECMA-262, также известная как ECMAScript, определяет стандарт для скриптового языка, на котором основан JavaScript. Эта спецификация опубликована Ecma International и включает в себя определение типов данных, синтаксиса, операторов, объектов и методов языка.

Спецификация ECMA-262 обновляется ежегодно, и каждое обновление включает новые функции и улучшения. Например, 14-е издание, выпущенное в июне 2023 года, определяет спецификацию языка ECMAScript 2023.

Последние черновики спецификации доступны на сайте TC39. Если вы хотите сообщить об ошибках в спецификации, вы можете сделать это на GitHub.

# 57 Вопрос

Операторы сравнения в JavaScript используются для сравнения двух значений. Вот основные операторы сравнения:

== (равно): Этот оператор сравнивает равенство значений, но не типов. Например, 1 == '1' вернет true, потому что значения равны, несмотря на различие в типах.

=== (строгое равенство или идентично): Этот оператор сравнивает равенство значений и типов. Например, 1 === '1' вернет false, потому что типы значений различаются.

!= (не равно): Этот оператор проверяет, не равны ли значения, но не учитывает типы.

!== (строгое неравенство): Этот оператор проверяет, не равны ли значения, учитывая их типы.

> (больше), < (меньше), >= (больше или равно), <= (меньше или равно): Эти операторы используются для сравнения двух значений.

# 58 Вопрос

В Document Object Model (DOM) различают 12 типов узлов. Однако, наиболее часто используется тип Node.ELEMENT\_NODE, номер которого равен 1.

DOM представляет HTML- или XML-документы в виде древовидных структур, где каждый узел является объектом документа. Представление DOM состоит из структурированной группы узлов и объектов, которые имеют свойства и методы.

# 59 Вопрос

В JavaScript есть два основных условных оператора: if и ? (тернарный оператор).

Оператор if: Оператор if выполняет блок кода, если заданное условие истинно. Если условие ложно, может быть выполнен другой блок кода с помощью оператора else.

Тернарный оператор (? :): Тернарный оператор является сокращенной формой оператора if...else. Он принимает три операнда: условие, за которым следует знак вопроса (?), затем выражение, которое выполняется, если условие истинно, сопровождается двоеточием (:), и, наконец, выражение, которое выполняется, если условие ложно.

# 60 Вопрос

Оператор || (логическое ИЛИ): Оператор || возвращает первое истинное значение среди своих операндов или последнее значение, если все операнды ложные.

Оператор ?? (nullish coalescing): Оператор ?? возвращает первый операнд, если он не null и не undefined. В противном случае, он возвращает второй операнд.

# 61 Вопрос

Цикл for...of в JavaScript используется для перебора итерируемых объектов, включая DOM-коллекции

Также можно использовать цикл for...of для перебора других итерируемых объектов, таких как массивы, строки, Map, Set и другие.

# 62 Вопрос

Атрибут async: Этот атрибут позволяет браузеру продолжать обработку страницы, пока скрипт загружается в фоновом режиме. Когда скрипт загружается, он немедленно выполняется. Это означает, что скрипты с атрибутом async не блокируют отрисовку страницы, но их выполнение может прервать её.

Атрибут defer: Этот атрибут также позволяет браузеру продолжать обработку страницы во время загрузки скрипта. Однако скрипты с атрибутом defer начинают выполняться только после полной загрузки страницы, и они выполняются в том порядке, в котором они появляются на странице.

# 63 Вопрос

Методы getElementsBy\* в JavaScript используются для получения элементов DOM по определенным критериям. Вот некоторые из этих методов:

getElementsByName(): Этот метод возвращает коллекцию элементов с указанным именем

getElementsByClassName(): Этот метод возвращает коллекцию элементов с указанным классом

getElementsByTagName(): Этот метод возвращает коллекцию элементов с указанным тегом

Важно отметить, что эти методы возвращают живую коллекцию, которая автоматически обновляется при изменении DOM. Это означает, что если вы добавите или удалите элемент, который соответствует критериям выборки, коллекция автоматически обновится.

# 64 Вопрос

Добавление элементов: Чтобы добавить элемент в DOM, можно использовать методы parentElem.appendChild(elem) и parentElem.insertBefore(elem, nextSibling). Метод appendChild() добавляет elem в список дочерних элементов parentElem, а метод insertBefore() вставляет elem в список дочерних parentElem перед элементом nextSibling.

Удаление элементов: Для удаления элементов из DOM используются методы parentElem.removeChild(elem) и parentElem.replaceChild(elem, currentElem). Метод removeChild() удаляет elem из списка детей parentElem, а метод replaceChild() заменяет currentElem на elem среди детей parentElem.

# 65 Вопрос

Создание элементов: Для создания новых элементов используются методы document.createElement(tag) и document.createTextNode(text). Метод createElement() создает новый элемент с указанным тегом, а метод createTextNode() создает новый текстовый узел с заданным текстом.

# 66 Вопрос

Использование свойства className: Свойство className отвечает за значение атрибута class элемента. Вы можете установить класс для элемента следующим образом: element.className = "myStyle".

# 67 Вопрос

Использование свойств innerHTML и outerHTML:

innerHTML: Это свойство устанавливает или получает HTML или XML разметку дочерних элементов.

outerHTML: Это свойство получает сериализованный HTML-фрагмент, описывающий элемент, включая его потомков.

# 68 Вопрос

Коллекция childNodes в документной объектной модели: Свойство childNodes возвращает коллекцию дочерних элементов данного элемента. Элементы в коллекции — объекты, а не строки. Чтобы получить данные из этих объектов, вы должны использовать их свойства.

# 69 Вопрос

В JavaScript, функции являются объектами, поэтому их можно копировать в другие переменные.

function myFunction() {

console.log("Hello World!");

}

var copyOfMyFunction = myFunction;

copyOfMyFunction(); // Выводит: "Hello World!"

# 70 Вопрос

Методы для вставки узлов или строк в HTML-документ: В JavaScript существуют различные методы для вставки узлов или строк в HTML-документ, включая Element.append(), Element.innerHTML, и другие.

# 71 Вопрос

Доступ к вложенным объектам в JavaScript: Для доступа к свойствам вложенных объектов в JavaScript, вы можете использовать точечную нотацию или скобочную нотацию. Например, если у вас есть объект person с вложенным объектом address, вы можете получить доступ к свойству city следующим образом: person.address.city или person["address"]["city"].

# 72 Вопрос

Перебор узлов DOM-коллекции свойствами nextSibling и previousSibling: Свойства nextSibling и previousSibling в JavaScript представляют собой ссылки на узлы-соседи элемента DOM в его родительском элементе nextSibling возвращает узел, непосредственно следующий за данным узлом в списке childNodes его родительского элемента, или null если данный узел последний в этом списке.

# 73 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

width: 500px;

flex-wrap: wrap;

}

.flex-item {

background-color: #f1f1f1;

width: 200px;

height: 200px;

margin: 10px;

}

</style>

</head>

<body>

<div class="flex-container">

<div class="flex-item"></div>

<div class="flex-item"></div>

<div class="flex-item"></div>

<div class="flex-item"></div>

<div class="flex-item"></div>

</div>

</body>

</html>

# 74 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

p:nth-child(even):hover {

background-color: red;

}

p:nth-child(odd):hover {

background-color: blue;

}

</style>

</head>

<body>

<p>Это первый параграф.</p>

<p>Это второй параграф.</p>

<p>Это третий параграф.</p>

<p>Это четвертый параграф.</p>

</body>

</html>

# 75 Вопрос

<!DOCTYPE html>

<html>

<body onclick="changeBackgroundColor()" style="height: 100vh; margin: 0;">

<script>

function changeBackgroundColor() {

document.body.style.backgroundColor = '#FFCCCC';

}

</script>

</body>

</html>

# 76 Вопрос

<!DOCTYPE html>

<html>

<body>

<button onclick="alert('Кнопка нажата!')">Нажми на меня</button>

</body>

</html>

# 77 Вопрос

<!DOCTYPE html>

<html>

<body>

<form action="https://example.com/submit" method="post">

<input type="text" name="textfield">

<input type="submit" value="Отправить">

</form>

</body>

</html>

# 78 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

width: 100%;

}

.fixed-column {

width: 200px;

height: 100px;

background-color: #f1f1f1;

margin: 10px;

}

.flex-column {

flex-grow: 1;

height: 100px;

background-color: #f1f1f1;

margin: 10px;

}

</style>

</head>

<body>

<div class="flex-container">

<div class="fixed-column"></div>

<div class="fixed-column"></div>

<div class="flex-column"></div>

</div>

</body>

</html>

# 79 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

img {

transition: transform 2s;

}

img:hover {

transform: scale(1.5);

}

</style>

</head>

<body>

<img src="https://upload.wikimedia.org/wikipedia/commons/thumb/8/8f/Example\_image.svg/600px-Example\_image.svg.png" alt="Your Image">

</body>

</html>

# 80 Вопрос

<!DOCTYPE html>

<html>

<body>

<table id="colorTable"></table>

<script>

var colors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet', 'pink', 'brown', 'gray'];

function generateTable() {

var table = document.getElementById('colorTable');

for (var i = 0; i < 5; i++) {

var row = table.insertRow(i);

var colorRow = colors.slice(i \* 5, i \* 5 + 5);

for (var j = 0; j < 5; j++) {

var cell = row.insertCell(j);

cell.style.backgroundColor = colorRow[j];

cell.style.width = '50px';

cell.style.height = '50px';

}

}

}

generateTable();

</script>

</body>

</html>

# 81 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: 100px 100px 100px;

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

gap: 0px 0px;

}

.grid-item {

background-color: blue;

}

.grid-item:nth-child(even) {

background-color: red;

}

</style>

</head>

<body>

<div class="grid-container">

<div class="grid-item"></div>

<div class="grid-item"></div>

<div class="grid-item"></div>

<div class="grid-item"></div>

<div class="grid-item"></div>

<div class="grid-item"></div>

<div class="grid-item"></div>

<div class="grid-item"></div>

<div class="grid-item"></div>

</div>

</body>

</html>

# 82 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

.flex-container {

display: flex;

justify-content: space-between;

}

.flex-item {

width: 100px;

height: 100px;

}

</style>

</head>

<body>

<div class="flex-container">

<img class="flex-item" src="URL-изображения" alt="Image">

<img class="flex-item" src="URL-изображения" alt="Image">

<img class="flex-item" src="URL-изображения" alt="Image">

<img class="flex-item" src="URL-изображения" alt="Image">

</div>

</body>

</html>

# 83 Вопрос

<!DOCTYPE html>

<html>

<body>

<form action="/submit\_form" method="post">

<label for="name">Имя:</label><br>

<input type="text" id="name" name="name"><br>

<label for="email">Email:</label><br>

<input type="email" id="email" name="email"><br>

<input type="submit" value="Отправить">

</form>

</body>

</html>

# 84 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

@keyframes spin {

from {transform:rotate(0deg);}

to {transform:rotate(360deg);}

}

.spin-div {

width: 100px;

height: 100px;

background-color: red;

animation: spin 2s linear infinite;

}

</style>

</head>

<body>

<div class="spin-div"></div>

</body>

</html>

# 85 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

a:hover {

color: red;

text-decoration: underline;

}

</style>

</head>

<body>

<a href="#">Это ссылка</a>

</body>

</html>

# 86 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

table {

border-collapse: collapse;

}

td {

border: 2px solid black;

padding: 10px;

text-align: center;

}

</style>

</head>

<body>

<table>

<tr>

<td>Ячейка 1</td>

<td>Ячейка 2</td>

<td>Ячейка 3</td>

</tr>

<tr>

<td>Ячейка 4</td>

<td>Ячейка 5</td>

<td>Ячейка 6</td>

</tr>

<tr>

<td>Ячейка 7</td>

<td>Ячейка 8</td>

<td>Ячейка 9</td>

</tr>

</table>

</body>

</html>

# 87 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

.circle {

width: 100px;

height: 100px;

border-radius: 50%;

margin: 10px;

}

.red {background-color: red;}

.orange {background-color: orange;}

.yellow {background-color: yellow;}

.green {background-color: green;}

.blue {background-color: blue;}

</style>

</head>

<body>

<div class="circle red"></div>

<div class="circle orange"></div>

<div class="circle yellow"></div>

<div class="circle green"></div>

<div class="circle blue"></div>

</body>

</html>

# 88 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

@keyframes blink {

0% {opacity: 1;}

50% {opacity: 0;}

100% {opacity: 1;}

}

.blink-div {

width: 100px;

height: 100px;

background-color: red;

animation: blink 1s infinite;

}

</style>

</head>

<body>

<div class="blink-div"></div>

</body>

</html>

# 89 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

.card {

border: 2px solid black;

border-radius: 10px;

padding: 10px;

margin: 10px;

width: 200px;

}

.card h2 {

color: blue;

}

.card p {

color: green;

}

</style>

</head>

<body>

<div class="card">

<h2>Заголовок 1</h2>

<p>Текст 1</p>

</div>

<div class="card">

<h2>Заголовок 2</h2>

<p>Текст 2</p>

</div>

<div class="card">

<h2>Заголовок 3</h2>

<p>Текст 3</p>

</div>

</body>

</html>

# 90 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

.responsive-block {

width: 100%;

height: auto;

background-color: #f1f1f1;

padding: 20px;

}

@media (min-width: 600px) {

.responsive-block {

width: 50%;

}

}

@media (min-width: 900px) {

.responsive-block {

width: 33.33%;

}

}

</style>

</head>

<body>

<div class="responsive-block">

Это адаптивный блок.

</div>

</body>

</html>

# 91 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

.container {

display: flex;

justify-content: space-between;

}

.block {

width: 100px;

height: 100px;

border: 2px solid black;

}

</style>

</head>

<body>

<div class="container">

<div class="block"></div>

<div class="block"></div>

</div>

</body>

</html>

# 92 Вопрос

<!DOCTYPE html>

<html>

<body>

<form action="/submit\_form" method="post">

<label for="name">Имя:</label><br>

<input type="text" id="name" name="name" placeholder="Введите ваше имя"><br>

<input type="submit" value="Отправить">

</form>

</body>

</html>

# 93 Вопрос

<!DOCTYPE html>

<html>

<body>

<div id="colorfulDiv" style="width: 200px; height: 200px;"></div>

<script>

function changeBackgroundColor() {

var letters = '0123456789ABCDEF';

var color = '#';

for (var i = 0; i < 6; i++) {

color += letters[Math.floor(Math.random() \* 16)];

}

document.getElementById('colorfulDiv').style.backgroundColor = color;

}

setInterval(changeBackgroundColor, 1000);

</script>

</body>

</html>

# 94 Вопрос

<!DOCTYPE html>

<html>

<body>

<div id="dateTimeDiv"></div>

<script>

window.onload = function() {

var now = new Date();

document.getElementById('dateTimeDiv').innerHTML = now;

}

</script>

</body>

</html>

# 95 Вопрос

<!DOCTYPE html>

<html>

<body>

<button id="colorButton">Наведите курсор на меня</button>

<div id="textDiv">Это текст</div>

<script>

document.getElementById('colorButton').onmouseover = function() {

var letters = '0123456789ABCDEF';

var color = '#';

for (var i = 0; i < 6; i++) {

color += letters[Math.floor(Math.random() \* 16)];

}

document.getElementById('textDiv').style.color = color;

}

</script>

</body>

</html>

# 96 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

.dropdown {

position: relative;

display: inline-block;

}

.dropdown-content {

display: none;

position: absolute;

min-width: 160px;

box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);

padding: 12px 16px;

z-index: 1;

}

.dropdown:hover .dropdown-content {

display: block;

animation: fadeIn 1s;

}

@keyframes fadeIn {

0% {opacity: 0;}

100% {opacity: 1;}

}

</style>

</head>

<body>

<div class="dropdown">

Наведите курсор на меня

<div class="dropdown-content">

<p>Ссылка 1</p>

<p>Ссылка 2</p>

<p>Ссылка 3</p>

</div>

</div>

</body>

</html>

# 97 Вопрос

<!DOCTYPE html>

<html>

<body>

<form id="taskForm">

<input type="text" id="taskInput" placeholder="Введите название задачи">

<input type="submit" value="Добавить">

</form>

<ul id="taskList"></ul>

<script>

document.getElementById('taskForm').onsubmit = function(event) {

event.preventDefault();

var task = document.getElementById('taskInput').value;

var li = document.createElement('li');

li.textContent = task;

document.getElementById('taskList').appendChild(li);

document.getElementById('taskInput').value = '';

}

</script>

</body>

</html>

# 98 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

.neon {

font-size: 50px;

color: #ffffff;

text-align: center;

animation: neon 1s ease-in-out infinite alternate;

text-shadow: 0 0 5px #ff0000, 0 0 10px #ff0000, 0 0 15px #ff0000, 0 0 20px #ff0000;

}

@keyframes neon {

from {

text-shadow: 0 0 5px #ff0000, 0 0 10px #ff0000, 0 0 15px #ff0000, 0 0 20px #ff0000;

}

to {

text-shadow: 0 0 10px #ff0000, 0 0 20px #ff0000, 0 0 30px #ff0000, 0 0 40px #ff0000;

}

}

</style>

</head>

<body>

<div class="neon">A W E S O M E</div>

</body>

</html>

# 99 Вопрос

<!DOCTYPE html>

<html>

<body>

<ul id="colorfulList">

<li>Элемент 1</li>

<li>Элемент 2</li>

<li>Элемент 3</li>

<li>Элемент 4</li>

<li>Элемент 5</li>

</ul>

<script>

var listItems = document.getElementById('colorfulList').children;

for (var i = 0; i < listItems.length; i++) {

listItems[i].onclick = function() {

var letters = '0123456789ABCDEF';

var color = '#';

for (var i = 0; i < 6; i++) {

color += letters[Math.floor(Math.random() \* 16)];

}

this.style.backgroundColor = color;

}

}

</script>

</body>

</html>

# 100 Вопрос

<!DOCTYPE html>

<html>

<head>

<title>Таблица 4x4</title>

<style>

td { width: 50px; height: 50px; }

</style>

</head>

<body>

<table id="myTable" border="1">

</table>

<script>

var table = document.getElementById("myTable");

for (var i = 0; i < 4; i++) {

var row = table.insertRow(i);

for (var j = 0; j < 4; j++) {

var cell = row.insertCell(j);

var button = document.createElement("button");

button.innerHTML = "Нажми меня";

button.onclick = (function() {

var counter = 0;

return function() {

counter++;

this.innerHTML = counter;

};

})();

cell.appendChild(button);

}

}

</script>

</body>

</html>

# 101 Вопрос

<!DOCTYPE html>

<html>

<head>

<title>Добавление элементов в список</title>

</head>

<body>

<form id="myForm">

<input type="text" id="inputField" placeholder="Введите текст">

<input type="button" value="Добавить" onclick="addItem()">

</form>

<ul id="myList">

</ul>

<script>

function addItem() {

var ul = document.getElementById("myList");

var li = document.createElement("li");

var input = document.getElementById("inputField");

li.appendChild(document.createTextNode(input.value));

ul.appendChild(li);

input.value = "";

}

</script>

</body>

</html>

# 102 Вопрос

<!DOCTYPE html>

<html>

<head>

<title>Форма с выводом в консоль</title>

</head>

<body>

<form id="myForm" onsubmit="event.preventDefault(); printToConsole();">

<input type="text" id="firstName" placeholder="Имя">

<input type="text" id="lastName" placeholder="Фамилия">

<input type="submit" value="Отправить">

</form>

<script>

function printToConsole() {

var firstName = document.getElementById("firstName").value;

var lastName = document.getElementById("lastName").value;

console.log("Имя: " + firstName + ", Фамилия: " + lastName);

}

</script>

</body>

</html>

# 103 Вопрос

<!DOCTYPE html>

<html>

<head>

<title>Текущая дата и время</title>

</head>

<body onload="displayDateTime()">

<p id="dateTime"></p>

<script>

function displayDateTime() {

var now = new Date();

var dateTimeString = now.toLocaleString();

document.getElementById("dateTime").innerHTML = dateTimeString;

}

</script>

</body>

</html>

# 104 Вопрос

<!DOCTYPE html>

<html>

<head>

<title>Список фруктов</title>

</head>

<body>

<ul id="fruitList">

<li onclick="alertItem(this)">Яблоко</li>

<li onclick="alertItem(this)">Груша</li>

<li onclick="alertItem(this)">Апельсин</li>

</ul>

<script>

function alertItem(item) {

alert(item.textContent);

}

</script>

</body>

</html>

# 105 Вопрос

<!DOCTYPE html>

<html>

<head>

<title>Текущая дата и время</title>

</head>

<body onload="displayDateTime()">

<div id="dateTime"></div>

<script>

function displayDateTime() {

var now = new Date();

var dateTimeString = now.toLocaleString();

document.getElementById("dateTime").innerHTML = dateTimeString;

}

</script>

</body>

</html>

# 106 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

body {

overflow: hidden;

}

.marquee {

position: relative;

overflow: hidden;

--animate-duration: 5s;

animation: marquee var(--animate-duration) linear infinite;

}

@keyframes marquee {

0% {

transform: translateX(100%);

opacity: 1;

}

90%, 100% {

transform: translateX(-50%);

opacity: 0;

}

}

</style>

</head>

<body>

<div class="marquee">

<p>Это ваша пробегающая строка!</p>

</div>

</body>

</html>

# 107 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

#animatedElement {

position: absolute;

width: 50px;

height: 50px;

background-color: red;

}

</style>

</head>

<body>

<div id="animatedElement"></div>

<script>

var element = document.getElementById('animatedElement');

var angle = 0;

setInterval(function() {

angle += 0.1;

element.style.left = 100 + 100 \* Math.cos(angle) + 'px';

element.style.top = 100 + 100 \* Math.sin(angle) + 'px';

}, 100);

</script>

</body>

</html>

# 108 Вопрос

<!DOCTYPE html>

<html>

<head>

<style>

button {

padding: 10px;

font-size: 16px;

}

</style>

</head>

<body>

<button onclick="changeBackgroundColor()">Нажми на меня</button>

<script>

function changeBackgroundColor() {

var randomColor = '#' + Math.floor(Math.random()\*16777215).toString(16);

document.body.style.backgroundColor = randomColor;

}

</script>

</body>

</html>

# 109 Вопрос

Элементы списка раскрываются при наведении и закрываются при убирании курсора, задание «скрытие при повторном наведении нереализуемо только при помощи HTML и CSS

<!DOCTYPE html>

<html>

<head>

<style>

    body {

        display: flex;

        justify-content: flex-start;

    }

ul {

  list-style-type: none;

  padding: 0;

}

li > ul {

  max-height: 0;

  overflow: hidden;

  transition: max-height 0.5s ease-out;

  padding-left: 15px;

}

li:hover > ul {

  max-height: 500px;

  transition: max-height 1s ease-in;

}

</style>

</head>

<body>

<ul>

  <li>Элемент 1

    <ul>

      <li>Подэлемент 1

        <ul>

            <li>Подэлемент 1</li>

            <li>Подэлемент 2</li>

          </ul>

      </li>

      <li>Подэлемент 2</li>

    </ul>

  </li>

  <li>Элемент 2

    <ul>

      <li>Подэлемент 1</li>

      <li>Подэлемент 1

        <ul>

            <li>Подэлемент 1    <ul>

                <li>Подэлемент 1</li>

                <li>Подэлемент 2</li>

              </ul></li>

            <li>Подэлемент 2</li>

          </ul>

      </li>

    </ul>

  </li>

</ul>

</body>

</html>

# 110 Вопрос

<!DOCTYPE html>

<html>

<body>

<canvas id="myCanvas" width="500" height="500" style="border:1px solid #d3d3d3;">

Ваш браузер не поддерживает элемент HTML5 Canvas.

</canvas>

<script>

var canvas = document.getElementById('myCanvas');

var ctx = canvas.getContext('2d');

var mouseX;

var mouseY;

var circle = {x: 100, y: 100, r: 50};

var isDragging = false;

canvas.addEventListener('mousedown', mouseDown, false);

canvas.addEventListener('mouseup', mouseUp, false);

canvas.addEventListener('mousemove', mouseMove, false);

function mouseDown(e) {

mouseX = e.pageX - this.offsetLeft;

mouseY = e.pageY - this.offsetTop;

if (Math.sqrt(Math.pow(mouseX - circle.x, 2) + Math.pow(mouseY - circle.y, 2)) < circle.r) {

isDragging = true;

}

}

function mouseUp() {

isDragging = false;

}

function mouseMove(e) {

if (isDragging) {

circle.x = e.pageX - this.offsetLeft;

circle.y = e.pageY - this.offsetTop;

drawCircle();

}

}

function drawCircle() {

ctx.clearRect(0, 0, canvas.width, canvas.height);

ctx.beginPath();

ctx.arc(circle.x, circle.y, circle.r, 0, 2 \* Math.PI);

ctx.stroke();

}

drawCircle();

</script>

</body>

</html>

# 111 Вопрос

<!DOCTYPE html>

<html>

<body>

<!-- Пример структуры товаров -->

<div class="product">

<h2>Товар 1</h2>

<p>Цена: <span class="price">100</span></p>

<input type="checkbox" class="select-product">

</div>

<div class="product">

<h2>Товар 2</h2>

<p>Цена: <span class="price">200</span></p>

<input type="checkbox" class="select-product">

</div>

<div class="product">

<h2>Товар 3</h2>

<p>Цена: <span class="price">150</span></p>

<input type="checkbox" class="select-product">

</div>

<button onclick="calculateTotal()">Подсчитать общую сумму</button>

<script>

function calculateTotal() {

var total = 0;

var products = document.getElementsByClassName('product');

for (var i = 0; i < products.length; i++) {

var checkbox = products[i].getElementsByClassName('select-product')[0];

var price = products[i].getElementsByClassName('price')[0];

if (checkbox.checked) {

total += parseFloat(price.textContent);

}

}

alert('Общая сумма: ' + total);

}

</script>

</body>

</html>

# 112 Вопрос

<!DOCTYPE html>

<html>

<body>

<div id="draggable" style="width: 100px; height: 100px; background: red; position: absolute">

Перетащи меня

</div>

<script>

var draggable = document.getElementById('draggable');

draggable.onmousedown = function(event) {

draggable.style.position = 'absolute';

draggable.style.zIndex = 1000;

function moveAt(pageX, pageY) {

draggable.style.left = pageX - draggable.offsetWidth / 2 + 'px';

draggable.style.top = pageY - draggable.offsetHeight / 2 + 'px';

}

moveAt(event.pageX, event.pageY);

function onMouseMove(event) {

moveAt(event.pageX, event.pageY);

}

document.addEventListener('mousemove', onMouseMove);

draggable.onmouseup = function() {

document.removeEventListener('mousemove', onMouseMove);

draggable.onmouseup = null;

};

};

draggable.ondragstart = function() {

return false;

};

</script>

</body>

</html>