1. Базовая структура HTML-документа. Назначение каждого элемента/тега базовой

структуры.

- <!DOCTYPE html> - объявление типа документа

- <html> - корневой элемент HTML-документа

- <head> - содержит метаинформацию о документе

- <title> - заголовок документа, отображается в строке заголовка браузера

- <body> - основное содержимое документа

2. Виды тегов и их описание. Примеры использования разных видов тегов.

- Теги блочных элементов, например <div> - используются для создания блоков содержимого

- Теги строчных элементов, например <span> - используются для выделения отдельных частей текста

- Теги заголовков, например <h1> - определяют уровень заголовка

- Теги списка, например <ul> (ненумерованный список) или <ol> (нумерованный список)

- Тег формы <form>

- Теги <header> и <footer> <main>

<img> <a>

3. Определения понятий “тег” и “атрибут”. Примеры использования атрибутов с тегами.

- Тег - это конструкция в HTML, которая определяет начало или конец элемента

- Атрибут - это дополнительная информация, которая указывается внутри тега и задает его свойства

Примеры использования атрибутов с тегами:

- <img src="image.png" alt="описание изображения"> - атрибуты src и alt задают путь к изображению и его описание

- <a href="https://www.example.com">ссылка</a> - атрибут href задает адрес ссылки

4. Организация вложенности элементов HTML. Примеры дочерних элементов. Примеры родственных элементов-потомков.

Организация вложенности элементов HTML осуществляется с помощью открывающих и закрывающих тегов. В HTML каждый элемент может быть вложен в другой элемент, образуя древовидную структуру. Например, элемент <div> может содержать вложенные элементы <p>, <h1>, <img> и т.д.

Вложенность элементов HTML позволяет создавать сложные структуры документа и управлять их внешним видом с помощью CSS. Например, создание макета веб-страницы с помощью вложенных блоков (<div>) и их содержимого (<h1>, <p>, <img>) позволяет легко управлять расположением и отображением контента на странице.

Примеры дочерних элементов:

- <ul> (список) может содержать элементы <li> (элемент списка)

- <div> (блок) может содержать другие элементы, такие как <p> (параграф) или <img> (изображение)

Примеры родственных элементов-потомков:

- <h1> (заголовок) может быть потомком элемента <header> (шапки документа)

- <li> (элемент списка) может быть потомком элемента <ul> (ненумерованного списка)

5. Основные теги для форматирования текста, их описание и назначение.

* <b>: выделяет текст жирным
* <del>: зачеркивает текст
* <i>: выделяет текст курсивом
* <em>: выделяет текст курсивом, в отличие от тега <i> носит логическое значение, придает выделяемому тексту оттенок важности
* <s>: зачеркивает текст
* <small>: делает текст чуть меньше размером, чем окружающий
* <strong>: выделяет текст жирным. В отличие от тега <b> предназначен для логического выделения, чтобы показать важность текста.
* <sub>: помещает текст под строкой
* <sup>: помещает текст над строкой
* <u>: подчеркивает текст
* <ins>: определяет вставленный (или добавленный) текст
* <mark>: выделяет текст цветом, придавая ему оттенок важности

Назначение: эти теги используются для управления видом текста и его отображения в браузере.

6. Семантика в форматировании текста в HTML. Примеры и назначение семантических тегов. (смысловые теги)

Семантика в форматировании текста в HTML отвечает за описание содержимого веб-страницы, делая код более понятным для поисковых систем и пользователей. Примеры семантических тегов: <header> (заголовок), <article> (статья), <section> (секция), <nav> (навигация), <footer> (подвал). Эти теги помогают организовать информацию на странице и улучшить ее доступность.

7. Блочные и строчные элементы в HTML. Изменение их поведения.

Блочные элементы в HTML занимают всю доступную ширину, а строчные элементы только ширину своего содержимого. Изменить поведение элементов можно с помощью CSS, например, с помощью свойства display: inline или display: block для того, чтобы сделать блочный элемент строчным или наоборот.

Блочные элементы: <div>, <h1>, <p>

Строчные элементы: <span>, <a>, <strong>

8. Добавление гиперссылок в HTML. Тег и его атрибуты с их описанием.

Для добавления гиперссылок в HTML используется тег <a> с атрибутом href, который указывает URL-адрес, на который будет произведен переход. Пример: <a href="https://www.example.com">Ссылка</a>.

1. href - указывает URL-адрес, на который будет осуществляться переход после клика на ссылку.

2. target - определяет, в каком окне или фрейме будет открыта новая страница после перехода по ссылке.

\_blank Открывает документ в новом окне.

\_self Открывает документ в том же фрейме или окне, в котором расположена гиперссылка.

\_parent Открывает документ в родительском фрейме (если таковой имеется). Если родительский фрейм отсутствует, то документ открывается в текущем окне.

\_top Открывает документ в текущем окне.

<имя-фрейма> Открывает документ в фрейме с указанным уникальным именем.

3. title - предоставляет всплывающую подсказку при наведении курсора на ссылку.

4. rel - определяет отношение между текущим документом и целевым ресурсом. Например, "nofollow" указывает на то, что поисковые системы должны игнорировать эту ссылку.

5. download - указывает, что целевой ресурс будет загружен вместо открыт в браузере, если это возможно.

6. type - указывает MIME-тип целевого ресурса, например, "image/jpeg" для изображения или "application/pdf" для PDF-файла.

7. media - определяет условия, при выполнении которых ссылка является активной, например, "print" для печати или "screen" для экрана компьютера.

Пример использования атрибутов в HTML-коде:

<a href="https://www.example.com" target="\_blank" title="Перейти на сайт" rel="noopener noreferrer" download="file.pdf" type="application/pdf" media="screen">Ссылка на файл</a>

9. Создание списков в HTML. Атрибуты списков, их описание и примеры использования.

Для создания списков в HTML используются теги <ul> (ненумерованный список) и <ol> (нумерованный список), а также <li> для элементов списка. Атрибуты списков включают type (тип нумерации), start (начальное значение нумерации), и value (значение элемента списка). Пример ненумерованного списка:

<ul>

<li>Пункт 1</li>

<li>Пункт 2</li>

</ul>

1. type - определяет тип маркированного списка. Может принимать значения:

- "1" - для нумерованного списка (по умолчанию)

- "A" - для списка с буквами

- "a" - для списка с маленькими буквами

- "I" - для списка с римскими цифрами (заглавные буквы)

- "i" - для списка с римскими цифрами (строчные буквы)

- "disc" - для списка с кружочками

- "circle" - для списка с кружками с внутренней заливкой

- "square" - для списка с квадратиками

2. start - устанавливает начальное значение для нумерации в списке.

3. compact - указывает браузеру на компактное отображение списка (на меньшем расстоянии между элементами).

4. reversed - определяет, нужно ли поменять порядок элементов в нумерованном списке.

10. Использование CSS-правила @keyframes.

CSS @keyframes - это механизм для создания анимации в CSS. Он позволяет определить изменения свойств CSS в течение определенного времени. Ключевые кадры указывают начальное и конечное состояние анимации, а система между ними автоматически создает плавное изменение. Применяя @keyframes к элементам, можно создавать различные виды анимации, такие как движение, изменение цвета, масштабирование и т. д.

@keyframes move {

from { opacity: 0; }

to { opacity: 1; transform: translateX(100px); }

}

.element {

animation: move 2s ease-in-out infinite;

}

11. Селекторы и комбинаторы CSS.

Селекторы в CSS используются для выбора определенных элементов на веб-странице. Существует множество различных типов селекторов, таких как селекторы по элементам, классам, ID, атрибутам и т. д. Комбинаторы позволяют определить отношения между различными элементами на странице, каких-то находятся внутри других, некоторые имеют определенные отношения друг с другом, например соседние элементы и др.

1. Селектор по типу:

Пример: p {

color: red;

}

Назначение: Применяет стили ко всем элементам указанного типа, например, ко всем абзацам (<p>).

2. Идентификаторный селектор:

Пример: #header {

background-color: grey;

}

Назначение: Применяет стили к элементу, который имеет определенный идентификатор.

3. Классовый селектор:

Пример: .btn {

background-color: blue;

}

Назначение: Применяет стили ко всем элементам, которые имеют определенный класс.

4. Потомковый селектор:

Пример: div p {

color: green;

}

Назначение: Применяет стили к элементам, которые являются потомками указанного элемента.

5. Групповой селектор:

Пример: h1, h2, h3 {

font-family: Arial, sans-serif;

}

Назначение: Применяет одни и те же стили к нескольким элементам.

6. Последовательный селектор (комбинатор потомков):

Пример: div > p {

font-size: 16px;

}

Назначение: Применяет стили только к непосредственным потомкам указанного элемента.

7. Соседний селектор (комбинатор соседей):

Пример: h2 + p {

margin-top: 20px;

}

Назначение: Применяет стиль к ближайшему соседу

8. Общий родственный комбинатор

Пример: img ~ p{

font-size: 12px;

}

Назначение: Применяет стиль ко всем родственным элементам, даже если они не соседи (в любом месте после)

12. Использование переменных и функции calc() в CSS.

Переменные в CSS позволяют задавать значение один раз и затем использовать его в нескольких местах, обеспечивая более простое и удобное обновление стилей.

Объявление переменной:

element {

--main-bg-color: brown;

}

Использование переменной:

element {

background-color: var(--main-bg-color);

}

Функция calc() используется для выполнения математических операций в CSS и может быть использована для вычисления значений свойств, таких как ширина, высота, отступы и т. д.

13. Использование технологии CSS Flexbox Layout при построении сайта. Основные методы

Flexbox Layout - это метод создания гибких и адаптивных макетов в CSS. Он позволяет управлять распределением пространства между элементами в контейнере, контролировать их выравнивание, изменять порядок отображения элементов и автоматически растягивать или уменьшать их размеры.

Когда элементы размещаются как гибкие элементы, они располагаются по двум осям:

Основная ось — это ось, проходящая в направлении расположения гибких элементов.

Поперечная ось — это ось, проходящая перпендикулярно направлению расположения гибких элементов.

flex-direction, которое определяет, в каком направлении движется главная ось

align-items: center; контролирует расположение гибких элементов на поперечной оси.

justify-content: space-around;

14. Использование технологии CSS Grid Layout при построении сайта. Основные методы

CSS Grid Layout предоставляет более мощные возможности для создания сложных макетов на веб-страницах. Он позволяет создавать сетки с определенным количеством строк и столбцов, позиционировать элементы внутри сетки и управлять их размерами и отступами. Это полезно для создания сложных макетов с большим количеством разнообразных элементов, расположенных на веб-странице.

Грид-элемент: дочерний элемент, прямой потомок грид-контейнера. Подчиняется правилам раскладки гридов.

Грид-линия: разделительная линия, формирующая структуру грида. Может быть как вертикальной (грид-линия колонки), так и горизонтальной (грид-линия ряда). Располагается по обе стороны от колонки или ряда. Используется для привязки грид-элементов.

Грид-ячейка: пространство между соседними грид-линиями. Единица грид-сетки.

Грид-полоса: пространство между двумя соседними грид-линиями. Может быть проще думать о грид-полосе как о ряде или колонке.

/\* Будет создано 3 колонки \*/

grid-template-columns: 150px auto 40%;

/\* Будет создано 3 ряда \*/

grid-template-rows: 250px 10vw 15rem;

Будет создано 3 колонки по 250 пикселей:

grid-template-columns: repeat(3, 250px);

С появлением гридов у нас появилась и новая единица измерения: fr

fr (от fraction — доля, часть) отвечает за свободное пространство внутри грид-контейнера.

Например, этот код создаст три колонки, каждая из которых будет занимать 1/3 ширины родителя:

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

Если элементов внутри грид-контейнера больше, чем может поместиться в заданные явно ряды и колонки, то для них создаются автоматические, неявные ряды и колонки. При помощи свойств grid-auto-columns и grid-auto-rows можно управлять размерами этих автоматических рядов и колонок.

Если грид-элементов больше, чем явно объявленных колонок или рядов, то они автоматически размещаются внутри родителя. А вот каким образом — в ряд или в колонку — можно указать при помощи свойства grid-auto-flow. По умолчанию значение у этого свойства row, лишние элементы будут выстраиваться в ряды в рамках явно заданных колонок.

Возможные значения:

row (значение по умолчанию) — автоматически размещаемые элементы выстраиваются в ряды.

column — автоматически размещаемые элементы выстраиваются в колонки.

dense — браузер старается заполнить дырки (пустые ячейки) в разметке, если размеры элементов позволяют. Можно сочетать с остальными значениями.

gap: 50px;

grid-column, grid-row Скопировать ссылку "grid-column, grid-row"

Свойства-шорткаты для grid-column-start, grid-column-end и grid-row-start, grid-row-end соответственно.

Значения для \*-start и \*-end разделяются слэшем.

Можно использовать ключевое слово span, буквально говорящее «растянись на столько-то». А на сколько, указывает стоящая за ним цифра.

.item1 {

grid-column: 3 / span 2;

grid-row: 3 / 4;

}

15. Правила построения CSS-стиля. Правило @import в CSS-стиле.

- CSS-стили пишутся внутри тега <style> внутри тега <head> или в отдельном файле с расширением .css

- Селектор определяет, к какому элементу или группе элементов будут применены стили

- Объявление содержит свойства и их значения, разделенные двоеточием и заканчивающиеся точкой с запятой

- Несколько стилей для одного селектора разделяются точкой с запятой

Правило @import в CSS-стиле:

- Правило @import позволяет подключать один CSS-файл в другой

- Указывается в начале стиля и имеет следующий синтаксис: @import url("stylesheet.css");

16. Создание CSS-анимации. Использование свойства transition в CSS-анимации. Основные методы

- Анимация создается с помощью ключевых кадров (keyframes), которые определяют изменение стилей элемента на определенных этапах анимации

- Свойство animation указывает параметры анимации, такие как длительность, количество циклов, задержка и т. д.

Использование свойства transition в CSS-анимации:

Свойство transition в CSS предоставляет возможность задать анимацию для перехода между двумя состояниями элемента, такими как изменение цвета, размера, положения и других параметров.

Основные методы для использования свойства transition:

1. transition-property: указывает, к каким свойствам CSS должна применяться анимация. Например, можно задать переход только для изменения цвета элемента.

2. transition-duration: указывает продолжительность анимации в секундах или миллисекундах. Например, можно задать переходную анимацию для изменения размера элемента за 1 секунду.

3. transition-timing-function: задает скорость изменения свойств во время анимации. Например, можно задать плавное изменение свойства в начале анимации и быстрое в конце.

4. transition-delay: указывает задержку перед началом анимации. Например, можно задать, чтобы анимация началась через 1 секунду после изменения свойства.

Пример использования свойства transition:

.element {

transition-property: width;

transition-duration: 1s;

transition-timing-function: ease-in-out;

transition-delay: 0.5s;

}

В данном примере определен переход для изменения ширины элемента с продолжительностью 1 секунда, плавным изменением и задержкой в полсекунды перед началом анимации.

17. Использование вложенных селекторов в CSS.

- Вложенные селекторы позволяют определять стили для элементов, которые находятся внутри других элементов

- Они указываются через пробел и определяются относительно родительского элемента

18. Использование псевдоклассов и псевдоэлементов в CSS.

- Псевдоклассы используются для определения стилей элемента в определенном состоянии, например :hover для наведения мыши

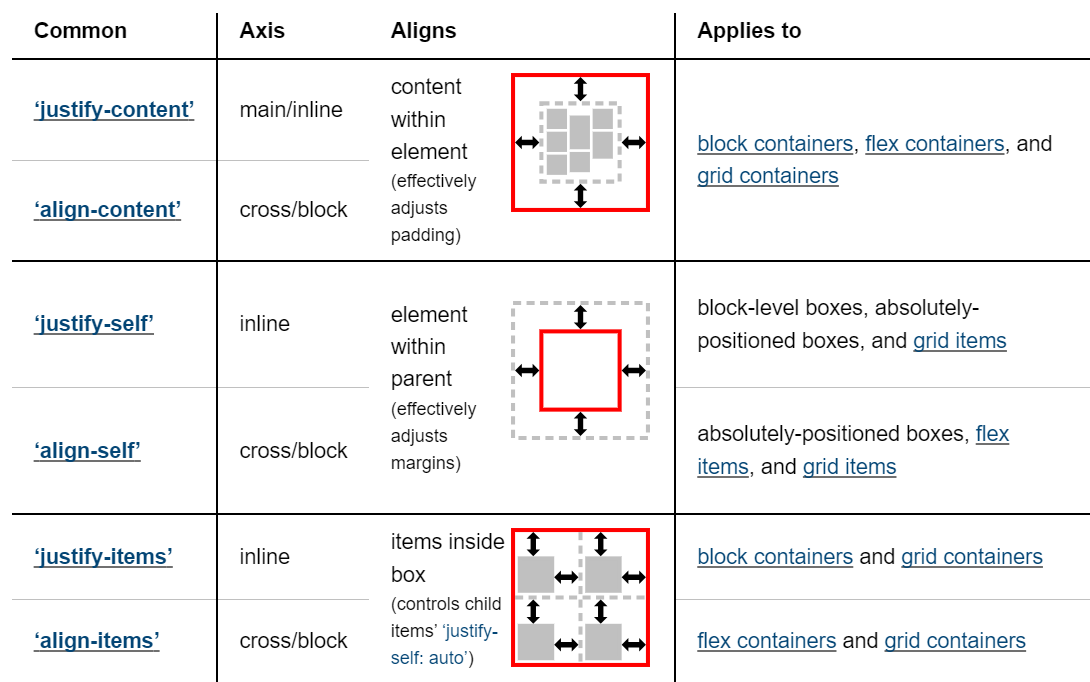
- Псевдоэлементы используются для добавления стилей к определенным частям элемента, таким как ::before или ::after

Псевдоклассы используются для изменения стиля элементов в различных состояниях, таких как :hover (при наведении), :active (при активации), :focus (при фокусировке), :nth-child() (выбор по номеру), :not() (исключение определенного элемента) и т. д. Примеры использования псевдоклассов:

Псевдоэлементы используются для добавления специальных стилей к определенным частям элементов, таким как ::before (добавляет контент перед элементом), ::after (добавляет контент после элемента), ::first-line (для стилизации первой строки), ::first-letter (для стилизации первой буквы) и т. д. Примеры использования псевдоэлементов:

19. Назначение модуля и элементы, содержащиеся в CSS Box Alignment Module Level 3.

CSS Box Alignment Module Level 3 предназначен для управления выравниванием элементов внутри блоков. Элементы, содержащиеся в этом модуле, включают в себя свойства для задания горизонтального и вертикального выравнивания, а также управления пространством между элементами.



Positional Alignment: the center, start, end, self-start, self-end, flex-start, flex-end, left, and right keywords

Distributed Alignment: the stretch, space-between, space-around, and space-evenly keywords

20. Назначение модуля и элементы, содержащиеся в CSS Box Sizing Module Level 3.

CSS Box Sizing Module Level 3 предназначен для управления размером элементов блоков. Элементы, содержащиеся в этом модуле, включают в себя свойство для задания размеров блока, включая padding и border, чтобы управлять их внутренним или внешним размером.

size

A one- or two-dimensional measurement: a block size and/or inline size; alternatively a width and/or height.

inner size

The content-box size of a box.

outer size

The margin-box size of a box.

max-content size (min)

21. Назначение модуля и элементы, содержащиеся в CSS Display Module Level 3.

24. Назначение модуля и элементы, содержащиеся в CSS Display Module Level 3.

CSS Display Module Level 3 предназначен для управления отображением элементов на веб-странице. Элементы, содержащиеся в этом модуле, включают в себя свойства для управления типом отображения элементов, таких как блочный, строчный, инлайн-блок и т. д.

none, block, inline, flex, grid,

22. Назначение модуля и элементы, содержащиеся в CSS Cascading and Inheritance Module Level 4.

Модуль CSS Cascading and Inheritance Level 4 предназначен для управления каскадированием стилей и наследованием свойств. Элементы, содержащиеся в этом модуле, включают в себя специфичность стилей, каскадирование и порядок применения стилей, а также наследование свойств от родительских элементов.

CSS Cascading and Inheritance Module Level 4 является частью CSS, которая определяет способы наследования и каскадирования стилей веб-документов. Этот модуль включает в себя несколько ключевых элементов.

Переменные: в этом модуле определены переменные CSS, которые позволяют задавать значение одной и той же свойству в нескольких местах кода. Переменные позволяют упростить структуру стилей и облегчить их поддержку и изменение.

Наследование: этот элемент модуля определяет способы наследования стилей от родительских элементов к дочерним. Наследование позволяет использовать общие стили для множества элементов без необходимости повторять их в коде.

Каскадирование: этот элемент модуля определяет способы определения приоритета стилей, если на элемент применяется несколько правил. Каскадирование позволяет установить порядок применения стилей и предотвратить конфликты.

Специфичность: этот элемент модуля определяет способы определения веса стилей на основе их селекторов. Специфичность позволяет установить, какой стиль будет применен к элементу, если есть несколько конфликтующих правил.

Эти элементы являются основой для определения и управления стилями веб-документов с помощью CSS Cascading and Inheritance Module Level 4. Они обеспечивают более гибкое и эффективное управление стилями, что улучшает поддержку и разработку веб-сайтов.

23. Назначение модуля и элементы, содержащиеся в CSS Color Adjustment Module Level 1.

Модуль CSS Color Adjustment Level 1 предназначен для управления цветовыми настройками элементов. Элементы, содержащиеся в этом модуле, включают в себя возможность управления насыщенностью, яркостью, оттенком и прозрачностью цвета, а также фильтры для коррекции цвета.

CSS Color Adjustment Module Level 1 включает в себя несколько элементов, которые позволяют разработчикам веб-сайтов точно настраивать цвета и их отображение. Некоторые из этих элементов включают в себя:

1. color-adjust: Это свойство позволяет разработчикам контролировать насыщенность и яркость цветов в изображениях на веб-странице. Оно позволяет подогнать цвета изображений под фоновые цвета и создать более естественное визуальное восприятие.

2. color-saturation: Это свойство позволяет управлять насыщенностью цвета в изображениях. Оно позволяет увеличить или уменьшить насыщенность, что может быть полезно для создания эффектов или улучшения отображения цветов.

3. color-contrast: Это свойство обеспечивает контроль контрастности цветов в изображениях. Оно позволяет увеличить или уменьшить контрастность, что помогает сделать изображения более четкими и легкими для восприятия.

4. цветовые функции: Модуль содержит несколько функций, таких как color() и colorspace(), которые позволяют разработчикам использовать математические операции для изменения цветового пространства и обработки цветов.

Эти и другие элементы модуля CSS Color Adjustment Level 1 предоставляют разработчикам широкий спектр возможностей для более точной настройки цветов и их отображения на веб-страницах.

25. Назначение модуля и элементы, содержащиеся в CSS Flexible Box Module Level 1.

Модуль гибких блоков CSS (CSS Flexible Box Module Level 1) предназначен для управления размещением элементов в контейнере, используя гибкие блочные макеты. Элементы этого модуля включают свойства для определения порядка и направления размещения дочерних элементов, распределения свободного пространства и выравнивания.

Ordering and Orientation

5.1Flex Flow Direction: the flex-direction property

5.2Flex Line Wrapping: the flex-wrap property

5.3Flex Direction and Wrap: the flex-flow shorthand

Flexibility

7.1The flex Shorthand

7.1.1Basic Values of flex

7.2Components of Flexibility

7.2.1The flex-grow property

7.2.2The flex-shrink property

7.2.3The flex-basis property

Alignment

8.1Aligning with auto margins

8.2Axis Alignment: the justify-content property

8.3Cross-axis Alignment: the align-items and align-self properties

8.4Packing Flex Lines: the align-content property

26. Назначение модуля и элементы, содержащиеся в CSS Fonts Module Level 4.

Модуль шрифтов CSS (CSS Fonts Module Level 4) предназначен для управления установкой и отображением шрифтов на веб-странице. Элементы этого модуля включают в себя свойства для задания семейства шрифтов, начертания, размера, кернинга и других параметров шрифта.

Font family: the font-family property

Font weight: the font-weight property

Font width: the font-stretch property

2.4Font style: the font-style property

2.5Font size: the font-size property

27. Назначение модуля и элементы, содержащиеся в CSS Grid Layout Module Level 1.

Модуль макета сетки CSS (CSS Grid Layout Module Level 1) предназначен для управления созданием сеточных макетов на веб-странице. Элементы этого модуля включают свойства для определения размеров ячеек сетки, их распределения, выравнивания и поведения при responsiv-дизайне.

CSS Grid Layout Module Level 1 включает в себя несколько основных элементов, которые обеспечивают создание гибкой и адаптивной сетки на веб-странице:

1. Контейнер сетки (grid container): Это родительский элемент, который содержит все элементы сетки. Он определяет контекст, в котором располагаются ячейки сетки.

2. Ячейки сетки (grid items): Это дочерние элементы контейнера сетки, которые размещаются в упорядоченной сетке с помощью свойств, определенных в CSS Grid Layout.

3. Ряды (rows) и колонки (columns): С помощью CSS Grid Layout Module Level 1 можно определять размеры и распределение рядов и колонок в сетке, задавая ширину, высоту и промежутки между ними.

4. Размещение элементов (grid placement): Элементы сетки могут быть размещены в определенных ячейках с помощью свойств grid-row и grid-column, позволяя контролировать их расположение внутри контейнера сетки.

5. Автоматическое размещение (auto-placement): CSS Grid Layout Module Level 1 также поддерживает автоматическое размещение элементов, что позволяет сетке самостоятельно определить расположение элементов в зависимости от доступного пространства.

Эти основные элементы обеспечивают мощные возможности для создания гибких и адаптивных сеток на веб-страницах с помощью CSS Grid Layout Module Level 1.

28. Назначение модуля и элементы, содержащиеся в CSS Grid Layout Module Level 2.

Модуль макета сетки CSS (CSS Grid Layout Module Level 2) расширяет функционал предыдущего уровня, добавляя возможность управления сеткой с помощью подсеток и шаблонов. Элементы этого модуля включают в себя дополнительные свойства для более гибкого создания макетов на веб-странице.

1. Subgrid: Этот элемент позволяет элементам вложенной сетки использовать родительскую сетку для определения своих собственных размеров и положений.

2. Content-based sizing: Этот элемент позволяет задавать размеры элементов на основе содержимого, а не задавать их явно.

3. Aspect ratio control: Этот элемент позволяет контролировать соотношение сторон элементов, что полезно при создании респонсивного дизайна.

4. Grid line names: Этот элемент позволяет задать имена сеточным линиям, что делает код более читаемым и понятным.

5. GridLayout: Этот элемент предоставляет возможность создавать сложные и гибкие макеты с использованием гридов, строки и столбцы которых можно определять, комбинировать и изменять в потоке.

6. GridGap: Этот элемент позволяет задавать пространство между элементами в сетке, что улучшает управление пространством и делает дизайн более гибким.

7. Auto-placement: Этот элемент автоматически размещает элементы в сетке, используя доступное пространство оптимальным образом.

8. Named Lines: Этот элемент позволяет задавать имена сеточным линиям, что делает код более читаемым и понятным.

29. Назначение модуля и элементы, содержащиеся в CSS Logical Properties and Values Module Level 1.

Назначение модуля CSS Logical Properties and Values Level 1 заключается в обеспечении возможности использования логических свойств и значений в CSS, что позволяет создавать адаптивные макеты без привязки к конкретным направлениям и языкам письма. Этот модуль содержит элементы, такие как block-start, block-end logical-inline-size, logical-block-size, inline-start, и inline-end.

CSS Logical Properties and Values Module Level 1 включает в себя следующие элементы:

1. Logical Properties: Это набор логических свойств, которые позволяют задавать стили элементам в зависимости от направления текста. Например, свойства для регулирования отступов - logical-margin-start, logical-margin-end и т.д.

2. Logical Values: Это значения, которые указывают направление расположения элементов на основе логических свойств. Например, значения 'start' и 'end' используются для определения начального и конечного положения элемента в зависимости от направления текста.

3. Block Flow: Этот термин относится к направлению, в котором элементы отображаются внутри блока. Например, значение 'block' указывает, что элементы отображаются сверху вниз, а значение 'inline' указывает, что элементы отображаются слева направо.

4. Inline Axis: Это ось, по которой располагаются элементы внутри блока. Например, значение 'block' указывает, что элементы располагаются горизонтально, а значение 'inline' указывает, что элементы располагаются вертикально.

5. Logical Direction: Это направление, в котором отображается текст и элементы на странице. Например, значение 'ltr' указывает, что направление текста слева направо, а значение 'rtl' указывает, что направление текста справа налево.

Эти элементы помогают разработчикам более гибко управлять расположением и отображением элементов на веб-страницах в зависимости от языка и направления текста.

30. Назначение модуля и элементы, содержащиеся в CSS Positioned Layout Module Level 3.

Модуль CSS Positioned Layout Level 3 предназначен для управления позиционированием элементов на веб-странице. Он включает в себя элементы, такие как top, right, bottom, left, inset-block-start, inset-inline-start, inset-block-end, inset-inline-end для управления положением элементов на странице.

static

The box is not a positioned box, and is laid out according to the rules of its parent formatting context. The inset properties do not apply.

relative

The box is laid out as for static, then offset from the resulting position. This offsetting is a purely visual effect, and does not affect the size or position of any other box, except insofar as it increases the scrollable overflow area of its ancestors. This positioning scheme is called relative positioning.

sticky

Identical to relative, except that its offsets are automatically adjusted in reference to the nearest ancestor scroll container’s scrollport (as modified by the inset properties) in whichever axes the inset properties are not both auto, to try to keep the box in view within its containing block as the user scrolls. This positioning scheme is called sticky positioning.

absolute

The box is taken out of flow such that it has no impact on the size or position of its siblings and ancestors, and does not participate in its parent’s formatting context.

fixed

Same as absolute, except the box is positioned and sized relative to a fixed positioning containing block

31. Назначение модуля и элементы, содержащиеся в CSS Text Decoration Module Level 3.

CSS Text Decoration Module Level 3 предназначен для управления оформлением текста на веб-страницах. Этот модуль содержит элементы для добавления декоративных элементов к тексту.

Line Decoration: Underline, Overline, and Strike-Through

2.1Text Decoration Lines: the text-decoration-line property

2.2Text Decoration Style: the text-decoration-style property

2.3Text Decoration Color: the text-decoration-color property

2.4Text Decoration Shorthand: the text-decoration property

2.5Text Underline Position: the text-underline-position property

Emphasis Marks

3.1Emphasis Mark Style: the text-emphasis-style property

3.2Emphasis Mark Color: the text-emphasis-color property

3.3Emphasis Mark Shorthand: the text-emphasis property

3.4Emphasis Mark Position: the text-emphasis-position property

4Text Shadows: the text-shadow property

32. Назначение модуля и элементы, содержащиеся в CSS Text Module Level 3.

CSS Text Module Level 3 предназначен для управления отображением текста на веб-страницах. Этот модуль включает в себя элементы, такие как text-align, text-indent, white-space для управления отступами, выравниванием и пробелами в тексте.

Case Transforms: the text-transform property

White Space and Wrapping: the white-space property

Breaking Rules for Letters: the word-break property

5.3Line Breaking Strictness: the line-break property

5.4Hyphenation: the hyphens property

Text Alignment: the text-align shorthand

6.2Default Text Alignment: the text-align-all property

6.3Last Line Alignment: the text-align-last property

6.4Justification Method: the text-justify property

Word Spacing: the word-spacing property

7.2Tracking: the letter-spacing property

33. Назначение модуля и элементы, содержащиеся в CSS Transforms Module Level 2.

CSS Transforms Module Level 2 предназначен для управления трансформацией элементов на веб-странице. Он содержит элементы, такие как translate, scale, rotate, для изменения размера, положения и поворота элементов.

transform-origin: Это свойство позволяет задать точку, относительно которой будет происходить трансформация элемента.

Свойство transform может принимать следующие значения:

none — никакого преобразования не применяется.

rotate(angle) — вращает элемент на заданный угол.

scale(x,y) — масштабирует элемент по осям X и Y.

translate(x,y) — перемещает элемент на заданные координаты.

skew(x-angle,y-angle) — наклоняет элемент на заданные углы по осям X и Y.

34. Назначение модуля и элементы, содержащиеся в CSS Transitions Module Level 1.

Назначение модуля CSS Transitions Level 1 - предоставить возможность создавать плавные переходы для свойств CSS, таких как цвет, размер, положение и прозрачность элементов. Элементы, содержащиеся в модуле, включают свойства transition-property, transition-duration, transition-timing-function и transition-delay.  
  
CSS Transitions Module Level 1 включает в себя следующие элементы:

1. transition-property: Этот атрибут определяет CSS свойства, которые будут изменяться с использованием перехода. Например, можно указать, что при наведении курсора на ссылку будет происходить плавное изменение цвета текста.

2. transition-duration: Этот атрибут определяет длительность перехода, то есть время, за которое будут происходить изменения CSS свойств. Например, можно задать, что изменение цвета текста при наведении будет происходить за 0.5 секунды.

3. transition-timing-function: Этот атрибут определяет функцию времени, которая управляет тем, как изменения CSS свойств будут происходить во времени. Например, можно задать, что цвет текста будет изменяться по равномерному временному интервалу или же с ускорением или замедлением.

4. transition-delay: Этот атрибут определяет задержку перед началом применения перехода. Например, можно указать, что изменение цвета текста начнется через 1 секунду после наведения курсора на ссылку.

35. Назначение модуля и элементы, содержащиеся в CSS Animations Level 1.

Назначение модуля CSS Animations Level 1 - предназначен для создания анимаций веб-страниц с помощью CSS. Элементы, содержащиеся в модуле, включают свойства animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count и другие свойства для управления анимацией.

CSS Animations Level 1 включает в себя несколько основных элементов, которые используются для создания анимаций на веб-страницах:

1. Ключевые кадры (keyframes): Этот элемент позволяет задать анимацию через определенные этапы, указывая начальное и конечное состояние, а также промежуточные кадры для создания плавного перехода.

2. Анимационные свойства: CSS Animations Level 1 поддерживает ряд специальных свойств, которые могут быть использованы для управления анимацией, такие как animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count и другие.

3. Анимационные ключевые слова: Эти ключевые слова могут быть использованы для определения времени и скорости анимации, такие как ease, ease-in, ease-out, ease-in-out, linear и другие.

4. Пользовательские анимации: CSS Animations Level 1 также позволяет создавать собственные анимации, определяя новые ключевые кадры и анимационные свойства, чтобы создавать уникальные и интересные эффекты.

5. Контроль анимации: С помощью CSS Animations Level 1 можно управлять анимацией, путем указания времени начала, остановки и повторения анимации, а также последовательности запуска нескольких анимаций.

36. Назначение модуля и элементы, содержащиеся в CSS Writing Modes Level 4.

Назначение модуля CSS Writing Modes Level 4 - предоставить возможность управлять направлением и расположением текста на веб-странице. Элементы, содержащиеся в модуле, включают свойства для установки горизонтального или вертикального направления текста, а также для управления блочным и строчным расположением текста.

CSS Writing Modes Level 4 defines CSS features to support for various international writing modes, such as left-to-right (e.g. Latin or Indic), right-to-left (e.g. Hebrew or Arabic), bidirectional (e.g. mixed Latin and Arabic) and vertical (e.g. Asian scripts).

CSS Writing Modes Level 4 определяет спецификацию для изменения направления и расположения текста на веб-страницах. В этом уровне включены различные элементы, которые влияют на представление текста, такие как:

1. direction: Этот свойство определяет направление текста, которое может быть слева направо (ltr) или справа налево (rtl).

2. writing-mode: Это свойство определяет режим написания для текста, который может быть горизонтальным (horizontal-tb), вертикальным слева направо (vertical-lr) или вертикальным справа налево (vertical-rl).

3. text-orientation: Это свойство определяет ориентацию текста, которая может быть горизонтальной (mixed), вертикальной (upright) или наклонной (sideways).

4. unicode-bidi: Это свойство определяет порядок отображения символов в тексте для языков, использующих письменность справа налево, таких как арабский и иврит.

5. overflow-wrap: Это свойство определяет, должен ли текст переноситься на новую строку, если он выходит за границы контейнера.

37. Назначение модуля и элементы, содержащиеся в Filter Effects Module Level 1.

Назначение модуля Filter Effects Level 1 - предоставить возможность применять визуальные эффекты к элементам с помощью CSS. Элементы, содержащиеся в модуле, включают свойства для наложения различных фильтров, таких как размытие, изменение цвета, контрастности и яркости, а также свойства для управления тенями.

Filter primitive representation

13.1.1grayscale

13.1.2sepia

13.1.3saturate

13.1.4hue-rotate

13.1.5invert

13.1.6opacity

13.1.7brightness

13.1.8contrast

13.1.9blur

13.1.10drop-shadow

38. Назначение модуля и элементы, содержащиеся в Media Queries Level 4.

Назначение модуля Media Queries Level 4 - предназначен для создания адаптивных веб-страниц, которые могут менять свой внешний вид в зависимости от характеристик устройства, таких как размер экрана, ориентация и плотность пикселей. Элементы, содержащиеся в модуле, включают медиа-запросы для определения различных аспектов устройства.

4.1Width: the width feature

4.2Height: the height feature

4.3Aspect-Ratio: the aspect-ratio feature

4.4Orientation: the orientation feature

@media <условие> {

<стиль>

}

all — все устройства;

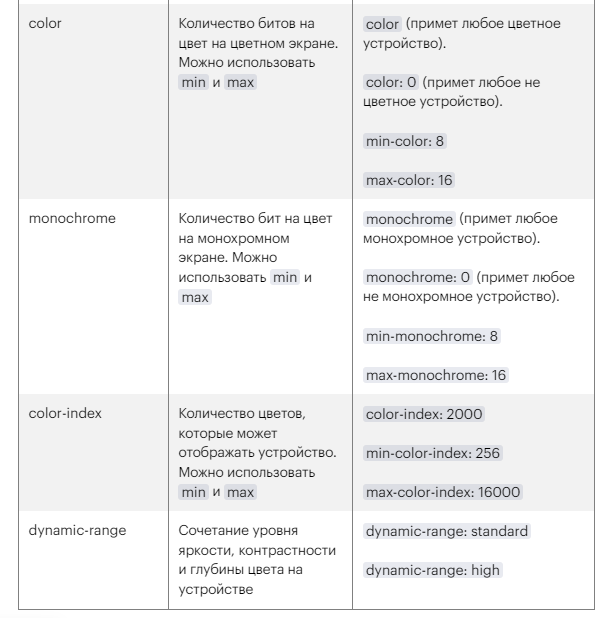
screen — устройства с экранами;

print — устройства в режиме предварительного просмотра страницы перед печатью;

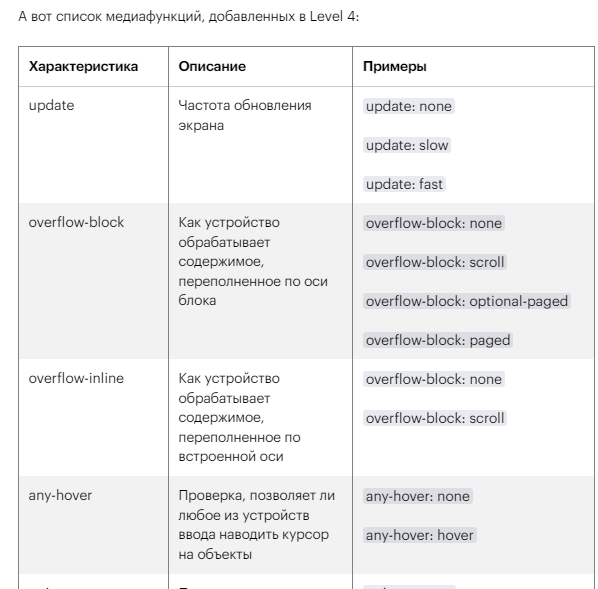
speech — программы чтения с экрана.

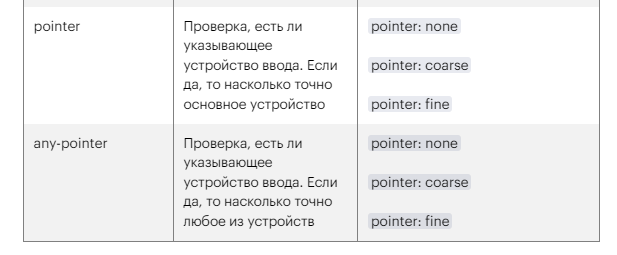
lvl3





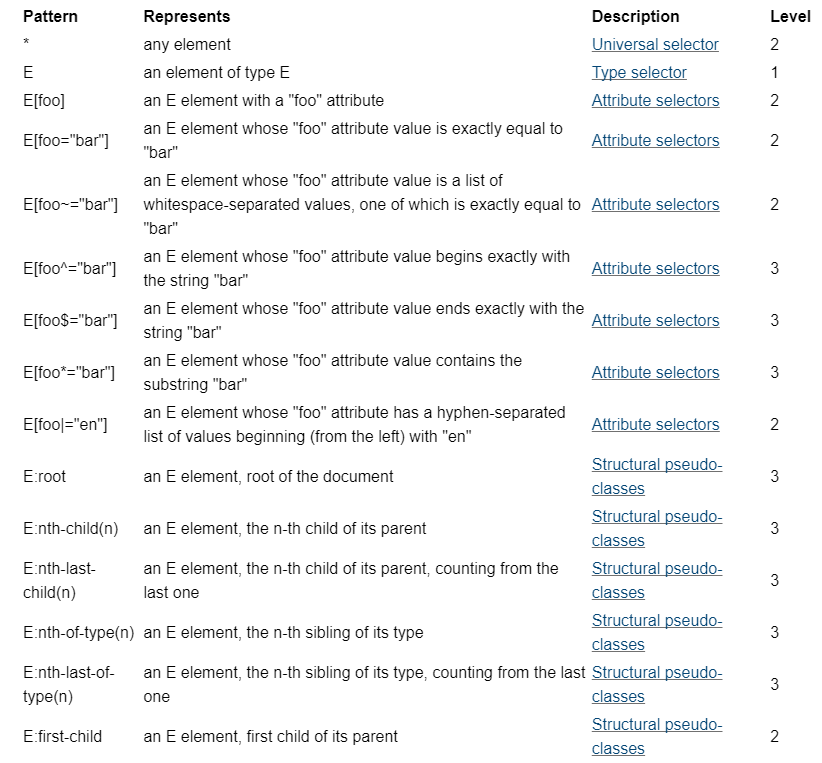
lvl4



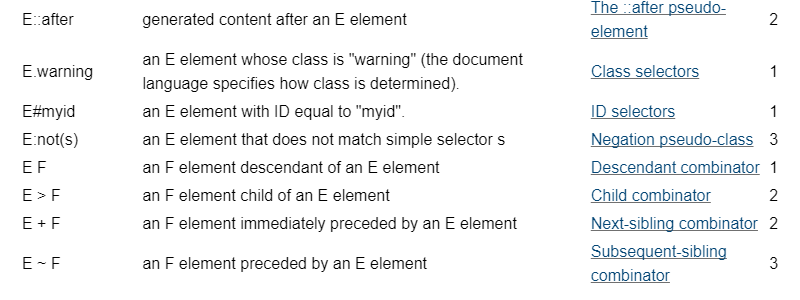


39. Назначение модуля и элементы, содержащиеся в Selectors Level 3.

Назначение модуля Selectors Level 3 - предоставить возможность выбирать и стилизовать элементы на веб-странице с помощью CSS. Элементы, содержащиеся в модуле, включают различные типы селекторов, такие как селекторы элементов, классов, идентификаторов, атрибутов и псевдо-элементов.







40. Изменение HTML-документа через свойство innerHTML.

Использование свойства innerHTML позволяет изменять содержимое HTML-элемента с помощью JavaScript. Например, если нужно изменить содержимое элемента <div> с id "example" на "Новый текст", можно использовать следующий код:

document.getElementById("example").innerHTML = "Новый текст";

41. Использование addEventListener.

Метод addEventListener используется для добавления обработчика события к элементу. Например, чтобы добавить обработчик клика на кнопку с id "btn", можно использовать следующий код:

document.getElementById("btn").addEventListener("click", function() {

// код обработчика события

});

42. Использование типа данных Symbol.

Тип данных Symbol добавлен в ECMAScript 6 и представляет собой уникальное и неизменяемое значение. Он часто используется для создания уникальных ключей в объектах. Например:

const mySymbol = Symbol("описание");

const obj = {

mySymbol: "значение"

};

В JavaScript тип данных Symbol представляет собой уникальный и неизменяемый примитивный тип данных. Symbol может быть использован для создания уникальных идентификаторов, которые не могут быть перезаписаны или изменены.

Создать символ можно с помощью функции Symbol(), передав ей опциональную строку в качестве описания. Например:

const mySymbol = Symbol('описание символа');

Symbol имеет несколько особенностей, которые делают его полезным:

1. Уникальность: каждый символ создает уникальное значение, что позволяет избежать конфликтов и пересечений имен переменных.

2. Неизменяемость: символы не могут быть изменены или перезаписаны. Это означает, что один и тот же символ, созданный дважды, будет представлять разные значения.

3. Легкая доступность: даже если два символа имеют одинаковое описание, они будут представлять разные значения.

Symbolы могут быть использованы, например, для создания уникальных ключей в объектах, обозначения определенных действий в приложении или для предотвращения изменений значений переменных. Они также могут быть использованы в качестве свойств объектов, вместо строковых ключей, чтобы избежать возможных пересечений имен.

43. Копирование и перезапись свойств объектов с использованием Object.assign.

Метод Object.assign используется для копирования свойств из одного или нескольких объектов в другой объект. Например:

const obj1 = {a: 1, b: 2};

const obj2 = {b: 3, c: 4};

const obj3 = Object.assign({}, obj1, obj2);

console.log(obj3); // {a: 1, b: 3, c: 4}

Копирование и перезапись свойств объектов с использованием Object.assign - это возможность скопировать свойства одного объекта и добавить их в другой объект, при этом перезаписывая уже существующие свойства в целевом объекте.

Метод Object.assign принимает первым аргументом целевой объект, в который будут добавляться свойства. Затем следуют объекты-источники, из которых будут скопированы свойства. Сначала копируются все перечислимые собственные свойства из каждого источника в целевой объект. При этом, если в целевом объекте уже есть свойство с таким же именем, оно будет перезаписано данным из источника.

Пример:

let target = { a: 1, b: 2 };

let source = { b: 3, c: 4 };

Object.assign(target, source);

console.log(target); // { a: 1, b: 3, c: 4 }

В данном примере метод Object.assign скопировал свойства из объекта source в объект target. Свойство 'b' в целевом объекте было перезаписано значением из объекта source, а свойство 'c' было добавлено в целевой объект.

44. Обработка браузерных событий мыши и клавиатуры.

Обработка событий мыши и клавиатуры происходит с помощью добавления обработчиков событий к соответствующим элементам или документу. Например, для обработки события нажатия клавиши "Enter" можно использовать следующий код:

document.addEventListener("keypress", function(event) {

if (event.key === "Enter") {

// код обработчика события

}

});

45. Обработка события onclick при нажатии на кнопку.

Для обработки события клика на кнопку используется добавление обработчика события при помощи метода addEventListener или прямого присвоения функции обработчика свойству onclick. Например:

document.getElementById("btn").addEventListener("click", function() {

// код обработчика события

});

или

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

<script>

function myFunction() {

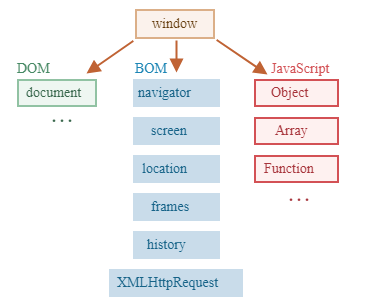
// код обработчика события

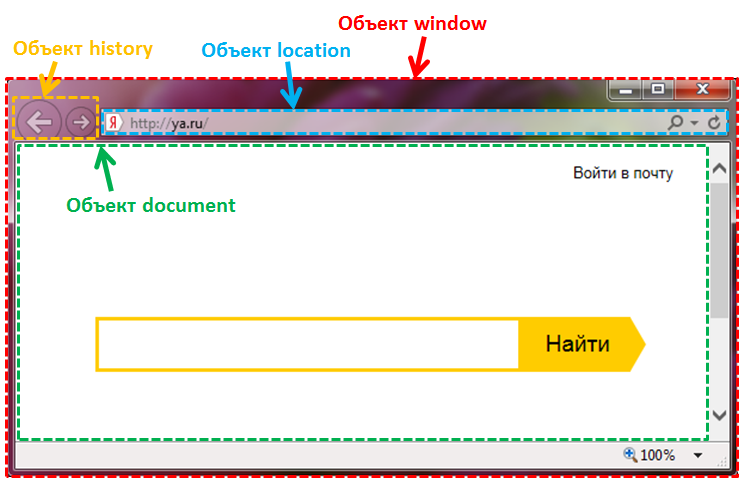
}

</script>

46. Объектная модель браузера.

<https://itchief.ru/javascript/bom>





В сценариях JavaScript браузер веб-разработчику предоставляет множество "готовых" объектов, с помощью которых он может взаимодействовать с элементами веб-страницы и самим браузером. В совокупности все эти объекты составляют объектную модель браузера (BOM – Browser Object Model).

На самом верху этой модели находится глобальный объект window. Он представляет собой одно из окон или вкладку браузера с его панелями инструментов, меню, строкой состояния, HTML страницей и другими объектами. Доступ к этим различным объектам окна браузера осуществляется с помощью следующих основных объектов: navigator, history, location, screen, document и т.д. Так как данные объекты являются дочерними по отношению к объекту window, то обращение к ним происходит как к свойствам объекта window.

47. Поиск CSS-селекторов через метод elem.querySelectorAll(css).

Поиск CSS-селекторов через метод elem.querySelectorAll(css) позволяет найти все элементы на странице, которые соответствуют указанному CSS-селектору. Метод возвращает NodeList, который представляет собой коллекцию найденных элементов.

const buttons = document.querySelectorAll('.button');

48. Поиск ближайшего предка через метод elem.closest(css).

Поиск ближайшего предка через метод elem.closest(css) позволяет найти ближайший родительский элемент, который соответствует указанному CSS-селектору. Метод возвращает найденный элемент или null, если ничего не было найдено.

var divElem = document.querySelector('div');

var closestItem = divElem.closest('.item');

49. Получение координат мыши и обработка движения мыши.

Получение координат мыши и обработка движения мыши в JavaScript можно осуществить с помощью событий мыши, таких как 'mousemove' для отслеживания движения мыши и 'click' для отслеживания кликов. Это позволяет получить координаты мыши и выполнять различные действия в зависимости от их положения.

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

var x = event.clientX; // Координата по оси X

var y = event.clientY; // Координата по оси Y

console.log('Координаты мыши: X - ' + x + ', Y - ' + y);

}

50. Проверка существования свойства, оператор in в JavaScript.

Проверка существования свойства, оператор in в JavaScript позволяет проверить, существует ли свойство в объекте или его прототипе. Например, можно использовать конструкцию if ('property' in object) { // код }, чтобы проверить, существует ли свойство 'property' в объекте и выполнить определенные действия в зависимости от результата.

51. Свойства firstChild и lastChild в документной объектной модели.

Свойство firstChild возвращает первый потомок узла в дереве документа, включая узлы текста и комментарии. Свойство lastChild возвращает последний потомок узла в дереве документа, включая узлы текста и комментарии.

52. Событие window.onload для работы после загрузки страницы.

Событие window.onload срабатывает после полной загрузки всех ресурсов страницы, таких как изображения и стили. Это событие можно использовать для запуска JavaScript кода после загрузки страницы.

53. События change, input, submit для обработки нажатия на тег <button>.

Событие change срабатывает при изменении значения элемента формы, например, при выборе нового значения в поле select.

Событие change происходит по окончании изменения значения элемента формы, когда это изменение зафиксировано.

Для текстовых элементов это означает, что событие произойдёт не при каждом вводе, а при потере фокуса.

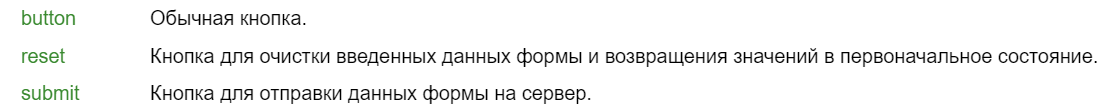
Например, пока вы набираете что-то в текстовом поле ниже – события нет. Но как только вы уведёте фокус на другой элемент, например, нажмёте кнопку – произойдёт событие onchange.

Событие input срабатывает при вводе текста в элемент формы.

Событие input срабатывает тут же при изменении значения текстового элемента и поддерживается всеми браузерами, кроме IE8-.

В IE9 оно поддерживается частично, а именно – не возникает при удалении символов (как и onpropertychange).

Событие submit срабатывает при отправке формы. При активации данной кнопки происходит отправка всех данных формы на сервер



54. Создание JavaScript-анимации.

Для создания JavaScript-анимации нужно использовать CSS или JavaScript для изменения свойств элементов страницы с течением времени. Например, можно использовать методы setInterval или requestAnimationFrame для обновления свойств элементов и создания визуальных эффектов.

Создание JavaScript-анимации - это процесс создания анимированных элементов на веб-странице с использованием языка программирования JavaScript. Анимация может включать в себя перемещение, изменение размера, изменение цвета, вращение и другие виды изменения внешнего вида элементов веб-страницы.

Для создания JavaScript-анимации разработчики используют различные методы и техники, такие как изменение CSS свойств, использование таймеров и интервалов, использование библиотек анимации и другие.

Процесс создания JavaScript-анимации включает в себя следующие шаги:

1. Определение анимируемых элементов на веб-странице.

2. Написание кода JavaScript, который определяет, какие изменения должны происходить с этими элементами в процессе анимации.

3. Запуск анимации с помощью вызова соответствующих функций и методов в JavaScript.

JavaScript-анимация может быть использована для создания интерактивных и привлекательных пользовательских интерфейсов, демонстрации слайд-шоу, создания игр и других видов визуальных эффектов на веб-страницах.

Анимация реализуется через последовательность кадров, каждый из которых немного меняет HTML/CSS-свойства.

Например, изменение style.left от 0px до 100px – двигает элемент. И если мы будем делать это с помощью setInterval, изменяя на 2px с небольшими интервалами времени, например 50 раз в секунду, тогда изменения будут выглядеть плавными. Принцип такой же, как в кино: 24 кадров в секунду достаточно, чтобы создать эффект плавности.

Теперь давайте представим, что у нас есть несколько анимаций, работающих одновременно.

Для разработки JavaScript-анимаций необходимо учитывать несколько важных элементов:

1. HTML-разметка: Определение элементов, которые будут анимироваться на веб-странице. Обычно это могут быть изображения, текст, кнопки и другие элементы, которые требуют анимации.

2. CSS-стили: Описание CSS-свойств элементов, которые будут изменяться в процессе анимации. Например, свойства, такие как позиция, размер, цвет, прозрачность и т.д. могут быть изменены с помощью CSS-анимации.

3. JavaScript: Непосредственно сам код для создания и управления анимацией. Обычно используются методы и функции JavaScript для изменения CSS-свойств элементов, таких как getElementById, addEventListener, setInterval и другие.

4. Тайминг и интерполяция: Определение скорости и плавности анимации с помощью методов и функций JavaScript. Это может включать в себя установку интервалов времени для выполнения анимации, использование ключевых кадров и другие методы управления временем.

55. Создание растрового двухмерного изображения, используя тег <canvas>.

Для создания растрового двухмерного изображения с помощью тега <canvas> нужно использовать JavaScript для рисования на холсте, задавать цвета, формы и другие свойства объектов, чтобы создать желаемый рисунок.  
  
Элемент <canvas> имеет только два атрибута — ширину и высоту. Если атрибуты высоты и ширины не установлены, то согласно спецификации html5 ширина элемента canvas будет равна 300 пикселям, а высота 150. При изменении этих атрибутов canvas очищается.

Затем, используя JavaScript, можно получить доступ к этому элементу и получить контекст рисования (2D) для него:

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

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

Теперь, имея контекст рисования, можно выполнять рисование различных объектов на холсте <canvas>. Например, чтобы нарисовать красный прямоугольник:

ctx.fillStyle = "red";

ctx.fillRect(50, 50, 100, 100);

Или нарисовать синий круг:

ctx.beginPath();

ctx.arc(200, 200, 50, 0, Math.PI\*2);

ctx.fillStyle = "blue";

ctx.fill();

Таким образом, можно использовать JavaScript для создания растрового изображения на <canvas>, рисуя на нем различные фигуры, текст и другие элементы. При этом <canvas> предоставляет возможность создавать анимации, обрабатывать пользовательский ввод и многое другое.

56. Спецификация ECMA-262.

ECMA-262 - это стандарт, который определяет язык программирования JavaScript. В этом стандарте описаны все основные элементы и характеристики языка JavaScript, включая его синтаксис, типы данных, операторы, выражения, методы и свойства объектов, структуры управления, функции, обработка ошибок, и многое другое.

Элементы спецификации ECMA-262 включают в себя:

1. Синтаксис и грамматика: Описывает правила написания корректного JavaScript кода, включая правила и порядок использования ключевых слов, операторов, выражений и т.д.

2. Типы данных: Описывает различные типы данных в JavaScript, такие как числа, строки, булевы значения, объекты, массивы, функции и т.д.

3. Операторы: Описывает различные операторы и их использование в выражениях, такие как арифметические, логические, сравнения и т.д.

4. Функции: Описывает создание и использование функций, их параметры, возвращаемые значения и область видимости.

5. Объекты: Описывает создание и использование объектов, их свойства и методы, прототипы, контекст выполнения и т.д.

6. Структуры управления: Описывает условные конструкции, циклы, операторы прерывания и т.д.

7. Обработка ошибок: Описывает механизмы обработки ошибок в JavaScript, такие как исключения, операторы try/catch и т.д.

Эти элементы составляют основу языка программирования JavaScript и позволяют разработчикам создавать сложные и интерактивные веб-приложения.

57. Операторы сравнения в JavaScript, их назначение.

Операторы сравнения в JavaScript используются для сравнения двух значений и возвращают булево значение true или false. Они включают в себя операторы == (равенство), != (неравенство), === (строгое равенство), !== (строгое неравенство), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно).

В JavaScript "==" (равенство) используется для сравнения значений двух операндов без учета их типов, в то время как "===" (строгое равенство) используется для сравнения значений и их типов.

- "5" == 5 вернет true, так как значения равны, несмотря на разные типы.

- "5" === 5 вернет false, так как значения и их типы разные.

58. Типы узлов в документной объектной модели.

Типы узлов в документной объектной модели (DOM) включают в себя узлы элементов, атрибутов, текста, комментариев, документов, документ-фрагментов и другие. Каждый тип узла имеет свои специфические характеристики и методы доступа к данным.

Типы узлов в документной объектной модели (DOM) представляют собой различные элементы, которые могут быть найдены в документе HTML или XML. Они могут быть классифицированы в несколько категорий в зависимости от их функций и свойств.

1. Узлы элементов (Element nodes): представляют собой HTML или XML теги, такие как <div>, <p>, <a> и т. д. Узлы элементов содержат содержимое, атрибуты и другие дочерние элементы.

2. Текстовые узлы (Text nodes): представляют собой текстовое содержимое внутри элементов. Они могут содержать текст, пробелы, переносы строк и другие символы.

3. Узлы атрибутов (Attribute nodes): представляют собой атрибуты элементов, такие как id, class, href и т. д. Узлы атрибутов обычно являются дочерними элементами узлов элементов.

4. Узлы комментариев (Comment nodes): представляют собой комментарии, которые могут быть добавлены в HTML или XML для пояснения кода. Узлы комментариев не имеют дочерних элементов и игнорируются браузером при отображении страницы.

5. Узлы документа (Document nodes): представляют собой сам документ HTML или XML. Они могут содержать другие узлы, такие как узлы элементов, текстовые узлы, узлы атрибутов и узлы комментариев.

Все эти типы узлов в документной объектной модели (DOM) представляют структуру и содержимое веб-страницы, позволяя JavaScript и другим языкам программирования взаимодействовать с ними и менять их по своему усмотрению.

59. Условные операторы: if и '?' в JavaScript

Условные операторы if и ? в JavaScript используются для выполнения определенного блока кода, если определенное условие истинно. Оператор if позволяет выполнить блок кода, если условие истинно, в то время как оператор ? (тернарный оператор) используется для более краткой записи условных выражений.

Условный оператор ? в JavaScript, также известный как тернарный оператор, позволяет выполнить условное присваивание значения переменной в зависимости от некоторого условия.

Синтаксис выглядит следующим образом:

условие ? значение\_если\_истина : значение\_если\_ложь;

Например:

var возраст = 18;

var проверка = (возраст >= 18) ? "Совершеннолетний" : "Несовершеннолетний";

console.log(проверка);

.

60. Операторы '||' и '??' в JavaScript.

Операторы '||' (логическое ИЛИ) и '??' (nullish coalescing) в JavaScript используются для работы с логическими значениями и значениями null или undefined. Оператор '||' возвращает первый истинный операнд, в то время как оператор '??' возвращает первый определенный операнд (не null и не undefined).

Оператор ?? (nullish coalescing) используется для проверки на значение null или undefined. Если переменная равна null или undefined, то оператор возвращает значение по умолчанию.

Например:

let name = null;

let defaultName = "Гость";

let result = name ?? defaultName;

console.log(result); // выведет "Гость"

В этом примере, переменная name равна null, поэтому в результате будет использовано значение по умолчанию "Гость".

61. Цикл for..of для перебора элементов DOM-коллекции.

Цикл for..of используется для перебора элементов DOM-коллекции, таких как NodeList или HTMLCollection. Он позволяет пройти по каждому элементу коллекции и выполнить определенные действия с ними.

Цикл for..of представляет собой новый способ перебора элементов массивов, строк и других итерируемых объектов в JavaScript. Он также может использоваться для перебора элементов DOM-коллекций.

DOM-коллекции представляют собой набор HTML-элементов (например, все элементы с определенным тегом или классом), которые могут быть найдены в документе с помощью методов, таких как getElementByTagName, getElementByClassName и других.

Для перебора элементов DOM-коллекции с помощью цикла for..of необходимо сначала преобразить коллекцию в массив с помощью метода Array.from, а затем используя цикл for..of перебирать элементы этого массива.

Пример использования цикла for..of для перебора элементов DOM-коллекции:

let elements = document.getElementsByClassName('someClass'); // получаем коллекцию элементов по классу

let elementsArray = Array.from(elements); // преобразуем коллекцию в массив

for (let element of elementsArray) {

// выполнение операций над каждым элементом

console.log(element.textContent);

}

Этот код позволяет перебирать все элементы, найденные в DOM-коллекции по определенному классу, и выполнять над ними необходимые операции.

62. Использование атрибутов defer и async в JavaScript.

Атрибут defer указывает браузеру, что скрипт должен быть выполнен после загрузки страницы, но до события DOMContentLoaded. Атрибут async указывает браузеру, что скрипт можно загружать параллельно с обработкой страницы и выполнять сразу после загрузки.

63. Использование методов getElementsBy\*.

Методы getElementsBy\* используются для получения коллекции элементов по их классу, тегу, атрибуту или имени. Примеры таких методов: getElementsByClassName, getElementsByTagName, getElementsByName и другие.

64. Использование методов для вставки и удаления узлов в JavaScript.

В JavaScript существуют методы для вставки и удаления узлов. Например, методы appendChild, removeChild, insertBefore позволяют добавлять, удалять и перемещать DOM-узлы.

65. Использование методов для создания узлов в JavaScript.

С помощью методов createElement и createTextNode можно создавать новые узлы в JavaScript. Метод createElement создает новый элемент, а метод createTextNode создает текстовый узел.

66. Использование свойства className.

Свойство className позволяет получить или изменить атрибут class элемента. Это свойство удобно использовать для изменения стилей элементов через JavaScript.

67. Использование свойств innerHTML и outerHTML.

Использование свойства innerHTML позволяет получить или установить HTML содержимое элемента, включая его потомков. Свойство outerHTML позволяет получить HTML содержимое элемента, включая сам элемент.

68. Коллекция childNodes в документной объектной модели.

Коллекция childNodes представляет собой список всех дочерних узлов элемента в документной объектной модели. Это может включать элементы, текстовые узлы и комментарии.

69. Копирование функции в другую переменную в JavaScript.

В JavaScript можно скопировать функцию в другую переменную, просто присвоив ей значение существующей функции. Например:

let originalFunction = function() {

// код функции

}

let copyFunction = originalFunction;

Копирование функции в другую переменную в JavaScript означает создание новой переменной, которая будет содержать ссылку на существующую функцию. Это позволяет использовать функцию под различными именами, вызывать её через новую переменную и передавать в качестве аргумента другим функциям.

Пример:

// Создание функции

function sayHello(name) {

console.log('Привет, ' + name + '!');

}

// Копирование функции в другую переменную

let greet = sayHello;

// Вызов функции через новую переменную

greet('Миша'); // Выведет "Привет, Миша!"

В этом примере мы создали функцию sayHello, а затем скопировали её в переменную greet. Теперь мы можем использовать greet для вызова функции sayHello с теми же аргументами.

70. Методы для вставки узлов или строк в HTML-документ.

Для вставки узлов или строк в HTML-документ в JavaScript можно использовать методы:

- appendChild() для добавления узла в конец родительского элемента

- insertBefore() для вставки узла перед указанным элементом

- innerHTML для вставки строки HTML внутрь элемента

- createDocumentFragment() для создания временного контейнера, в который можно добавить узлы или строки

71. Доступ к вложенным объектам в JavaScript.

Доступ к вложенным объектам в JavaScript можно осуществить через использование точечной нотации. Например, если объект obj имеет свойство nested, к которому нужно получить доступ, можно сделать это следующим образом:

let value = obj.nested;

Доступ к вложенным объектам в JavaScript означает возможность обращаться к свойствам и методам объектов, которые являются членами другого объекта. В JavaScript объекты могут содержать другие объекты в качестве своих свойств, образуя так называемые вложенные объекты.

Для доступа к вложенным объектам можно использовать точечную нотацию, например:

let car = {

brand: "Toyota",

model: "Camry",

color: "black",

engine: {

type: "V6",

power: "300 hp"

}

};

console.log(car.brand); // "Toyota"

console.log(car.engine.type); // "V6"

В данном примере мы обращаемся к свойствам объекта car с помощью точечной нотации. Свойство engine является вложенным объектом, и мы можем обратиться к его свойствам также с помощью точечной нотации.

Также можно использовать квадратные скобки для доступа к вложенным объектам, но в этом случае необходимо указать путь к свойству в виде строки:

console.log(car["brand"]); // "Toyota"

console.log(car["engine"]["type"]); // "V6"

Таким образом, доступ к вложенным объектам в JavaScript позволяет удобно работать с данными, организованными в виде объектов, и обращаться к вложенным свойствам и методам.

72. Перебор узлов DOM-коллекции свойствами nextSibling и previousSibling.

Перебор узлов DOM-коллекции свойствами nextSibling и previousSibling позволяет обойти всех соседних узлов определенного элемента. Эти свойства содержат ссылки на следующий и предыдущий узлы того же уровня вложенности.