Какой тег используется для создания ссылки?

<a>

Этот тег используется для определения гиперссылок. Атрибут href указывает адрес, на который ведёт ссылка. Например, Перейти на сайт создаст ссылку на указанный адрес.

Какой тег определяет основное содержимое веб-страницы?

<body>

Этот тег содержит все видимые элементы на странице, такие как текст, изображения, ссылки и другие компоненты. Все, что вы видите в браузере, находится внутри тега <body>.

Как подключить внешний файл script.js к HTML?

<script src="script.js"></script>

Этот тег используется для подключения JavaScript-файлов. Атрибут src указывает путь к внешнему скрипту, который будет загружен и выполнен. Например, вы можете подключить файл script.js, добавив этот тег в секцию <head> или перед закрывающим тегом </body>.

Как сделать, чтобы элемент занимал всю доступную ширину, но не превышал 1200px?

width: 100%; max-width: 1200px;.

Это CSS-свойство позволяет элементу увеличиваться до 100% ширины своего родительского контейнера, но при этом не превышать значение в 1200px. Идеально подходит для адаптивного дизайна.

Какое CSS-свойство позволяет анимировать изменения значений других свойств?

transition

Это свойство позволяет плавно изменять значение CSS-свойств, когда происходит изменение, например, при наведении курсора на элемент. Вы можете задать, какие свойства будут анимироваться, а также длительность анимации.

Как применить стили только к элементам <p>, которые являются прямыми потомками ?

div > p

div > p: Этот селектор применяется только к тем элементам <p>, которые являются непосредственными (прямыми) потомками . То есть он будет выбирать только те <p>, которые непосредственно находятся внутри без промежуточных элементов.

Что вернет этот код?

Promise.resolve(1).then(x => { throw x + 1 }).catch(x => x + 2).then(x => x + 3);

5

- Promise.resolve(1) создаёт промис, который сразу же разрешается со значением 1.
- .then(x => { throw x + 1 }) - когда промис разрешается, он передает значение 1 в этот обработчик. Здесь x становится 1, и поскольку происходит выброс ошибки с помощью throw, код завершает выполнение этого обработчика и передаёт управление в следующий блок catch.
- .catch(x => x + 2) - этот блок catch перехватывает ошибку, которая произошла в предыдущем then. Поскольку выброшенное значение — это 1 + 1, то x в этом обработчике будет 2. Этот блок вернет значение 2.
- .then(x => x + 3) - на этом этапе x станет 2 (значение, возвращаемое из catch). Этот блок then будет выполнен, и тут мы добавляем 3, получая 2 + 3, что равно 5.

Какой метод класса Array изменит исходный массив, удалив элементы и вставив новые?

splice()

- .slice(): Этот метод создает новый массив, содержащий копию части исходного массива, начиная с указанного индекса и заканчивая (но не включая) другим указанным индексом. Он не изменяет исходный массив.
- .splice(): Данный метод изменяет исходный массив, начиная с указанной позиции. Он может удалять, добавлять или заменять элементы в массиве. Например, если вызвать array.splice(1, 2, 'new'), он удалит два элемента, начиная с индекса 1, и вставит 'new' на то же место. Таким образом, этот метод действительно изменяет исходный массив.
- .map(): Этот метод создает новый массив, в котором каждый элемент является результатом вызова переданной функции для каждого элемента исходного массива. Он также не изменяет исходный массив, а просто создает новую копию.
- .reduce(): Этот метод выполняет указанную функцию на каждом элементе массива и сводит его к одному значению. Результатом также является новое значение, а исходный массив не изменяется.

Что будет в result?

const arr = [1, 2, 3]; const result = [...arr, ...arr]; [1, 2, 3] [1, 2, 3, 1, 2, 3] [[1, 2, 3], [1, 2, 3]] "1,2,31,2,3"

[1, 2, 3, 1, 2, 3].

В данном случае, в выражении const result = [...arr, ...arr]; используется оператор расширения (spread operator) для массива arr, который равен [1, 2, 3]. Оператор расширения позволяет "распаковать" элементы массива и использовать их в другом массиве или контексте.
Когда мы применяем ...arr дважды, фактически мы собираем все элементы массива arr и добавляем их в новый массив. Это значит, что сначала будет добавлен первый ...arr, который создает [1, 2, 3], а затем второй ...arr, который также создает [1, 2, 3]. Таким образом, результатом объединения этих двух массивов будет новый массив, содержащий все элементы из обоих arr, что в итоге даст:
[1, 2, 3, 1, 2, 3].