Синхронное и асинхронное подключение JavaScript к HTML.

Независимо от того, как JavaScript-код добавляется на страницу (встроенным, внешним способом), элементы <script> обрабатываются в том порядке, в котором они расположены в HTML-коде, при условии, что у них нет атрибутов defer и async. Код первого элемента <script> должен быть полностью интерпретирован, чтобы можно было приступить ко второму элементу <script>, второй элемент должен быть полностью обработан перед третьим, и так далее. Если у тега <script> есть один из двух указанных атрибутов, порядок другой.

Синхронное подключение.

Браузер читает HTML-документ сверху вниз и, начинает отображать страницу, показывая часть документа до тега <script>. Когда он встречает тег <script>, переключается в JavaScript-режим, рассматривает текст программы как сценарий и выполняет его. Остальной контент страницы не загружается и не отображается, пока не будет выполнен весь код в элементе <script> . Закончив выполнение, браузер возвращается обратно в HTML-режим и отображает оставшуюся часть документа.

Пример, в котором элемент <script> расположен где-то в середине страницы:

```
<html>
<body>
Текст до скрипта
<script>Текст самого скрипта</script>
Текст после скрипта не будет показан, пока браузер не выполнит script.js
</body>
</html>
```

Несколько элементов <script> при синхронном подключении JavaScript к HTML интерпретируются браузером в том порядке, в котором они расположены в HTML-документе. Сначала интерпретируется код первого элемента <script> , затем браузер приступает ко второму элементу <script> и далее последовательно.

Такое поведение браузера называется "синхронным". Недостаток в том, что когда по умолчанию прерывается синтаксический анализ (парсинг) HTML-документа, это приводит к увеличению промежутка времени до первой отрисовки страницы. Если мы загружаем несколько JavaScript-файлов на странице, это увеличивает время её отрисовки значительно.

Асинхронное подключение.

Асинхронно - значит непоследовательно загружается, открывается, выполняется HTML и JavaScript, а по правилам атрибутов async и defer. Эти 2 атрибута обеспечивают асинхронное подключение. Поддерживаются async и defer только для внешних файлов сценариев, то есть работают при наличии атрибута src . Асинхронная загрузка – наиболее подходящий вариант, когда JavaScript независим от другого файла JavaScript, поскольку нам не важно, когда скрипт будет исполнен.

Атрибут async.
Атрибут async используется, если нужно разрешить браузеру продолжить загрузку страницы, не дожидаясь завершения загрузки и выполнения сценария.

При обнаружении тега <script async src="..."> браузер не останавливает обработку HTML-документа для загрузки и выполнения скрипта. Выполнение произойдёт после того, как скрипт будет получен параллельно с разбором документа. Когда скрипт будет загружен – он выполнится сразу, даже если HTML-документ ещё не полностью готов.

Для сценариев с атрибутом async не гарантируется выполнение скриптов в порядке их расположения в HTML. Первым начнёт выполняться тот сценарий, который раньше загрузится. Поскольку второй скрипт может быть выполнен перед первым, поэтому важно, чтобы между этими сценариями не было зависимостей.

Атрибут defer.
Атрибут defer используется, если нужно разрешить браузеру продолжить загрузку страницы, не дожидаясь завершения загрузки и выполнения сценария.

При обнаружении тега <script defer src="..."> браузер не останавливает обработку HTML-документа для загрузки и выполнения скрипта. Атрибут defer откладывает выполнение скрипта до тех пор, пока вся HTML-страница не будет загружена полностью. Сам JS-файл может быть загружен, в то время как HTML-документ ещё грузится. Однако, даже если скрипт будет полностью загружен ещё до того, как браузер закончит обработку страницы, он не будет выполнен до тех пор, пока HTML-документ не обработается до конца.

Несмотря на то, что тег <script defer src="..."> может быть включен в элемент <head> HTML-документа, выполнение сценария не начнется, пока браузер не дойдет до закрывающего тега </html>.

Если в HTML несколько скриптов, то сценарии будут выполняться в том порядке, в котором они расположены в HTML-коде. Первым выполнится тот сценарий, который расположен в коде раньше. Это в отличии от async, атрибут defer сохраняет последовательность выполнения скриптов.

Применение атрибута defer бывает полезным, когда в коде скрипта предусматривается работа с HTML-документом, и разработчик должен быть уверен, что страница полностью получена.

При асинхронном подключении отображение кода HTML не останалівлается, но при большом количестве скриптов на странице сайта или большом размере применяемых скриптов, могут снова возникнуть длительные задержки при загрузке, в течение которых пользователь видит пустое окно браузера. Поэтому считается хорошей практикой все **ссылки** на JavaScript-сценарии указывать после контента страницы перед закрывающим тегом <body>. Ведь скрипты всё-равно отдельными файлами, так как async и defer работают при внешнем подключении. Связь скриптов с телом HTML, например, через глобальный атрибут id или атрибут name. Можно в

конец кода, поэтому.

Пример, в котором ссылки на <script> расположены в конце страницы:

```
<html>
<body>
<!-- Контент страницы -->
.....
<script src="script1.js" async></script>
<script src="script2.js" async></script>
</body>
</html>
```

Такое расположение сценариев позволяет браузеру сначала загрузить контент страницы, а потом будет загружаться код сценария. Для пользователей это предпочтительнее, потому что страница полностью визуализируется в браузере до обработки JavaScript-кода.