Процесс отображения страниц можно разбить на следующие основные этапы:

1. **Начало разбора HTML** (Когда браузер начинает получать данные HTML страницы по сети, он немедленно запускает свой синтаксический анализатор parser для преобразования HTML в объектную модель документа (DOM))
2. **Получение внешних ресурсов** (Когда парсер встречает внешний ресурс, такой как файл CSS или JavaScript, он пытается, получить его. Синтаксический анализатор будет продолжать работу по мере загрузки файла CSS, но он **заблокирует рендеринг** до тех пор, пока файл не будет загружен и проанализирован.

Файлы JavaScript немного отличаются – по умолчанию они так же блокируют синтаксический анализ HTML, на время загрузки. Но у них есть два атрибута, которые могут быть добавлены в теги сценария, чтобы изменить это: **defer** и **async**. Оба позволяют синтаксическому анализатору продолжать работу, пока файл JavaScript загружается в фоновом режиме.

Чтобы пометить ресурс как важный и, следовательно, с большей вероятностью он должен быть загруженным на ранней стадии процесса рендеринга, можно использовать тег ссылки с**rel = “preload”**. <link href="style.css" rel="preload" as="style" />

1. **Разбор CSS и создание CSSOM** (Подобно файлам HTML и DOM, когда файлы CSS загружаются, они должны быть проанализированы и преобразованы в дерево – на этот раз CSSOM. Он описывает все селекторы CSS на странице, их иерархию и их свойства,)
2. **Выполнение JavaScript** (Как и когда ресурсы JavaScript будут загружены, определяет, в какой-то момент они будут проанализированы, скомпилированы и выполнены. В разных браузерах для выполнения этой задачи используются разные механизмы JavaScript. Анализ JavaScript может быть дорогостоящим процессом с точки зрения ресурсов компьютера, в большей степени, чем другие типы ресурсов, поэтому его оптимизация так важна для достижения хорошей производительности.)
3. **Объединение DOM и CSSOM, для построения дерево рендеринга** (Дерево рендеринга представляет собой комбинацию DOM и CSSOM и представляет все, что будет отображаться на странице. Это не обязательно означает, что все узлы в дереве рендеринга будут визуально присутствовать, например узлы со стилями opacity: 0 или visibility: hidden будут включены и могут быть прочитаны программой чтения с экрана и т. д., тогда как те, которые настроены на display: none будет исключены. Кроме того, такие теги, как <head>, не содержащие визуальной информации, всегда будут пропущены.)
4. **Расчет макета и отрисовка результата** (Теперь, когда у нас есть полное дерево рендеринга, браузер знает, что рендерить, но не знает, где рендерить. Следовательно, необходимо рассчитать макет страницы (то есть положение и размер каждого узла). Механизм рендеринга проходит дерево рендеринга, начиная с вершины и идя вниз, вычисляет координаты, в которых должен отображаться каждый узел.

Как только это будет сделано, последний шаг – используя эту информацию о макете отрисовать пиксели на экране.)