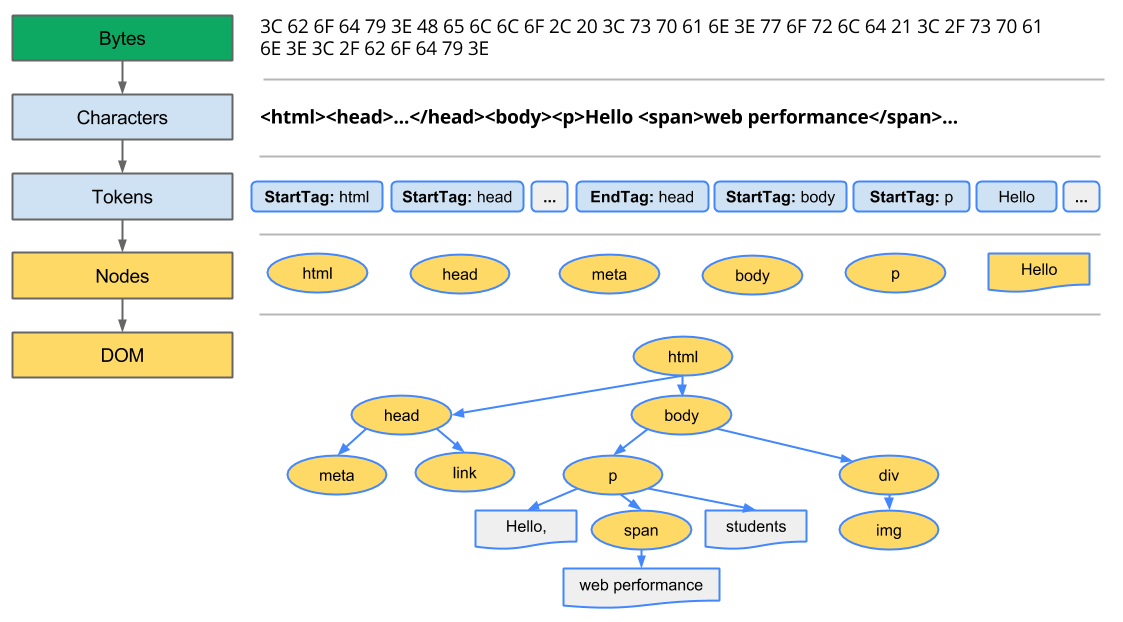
Tree model, browser specifics

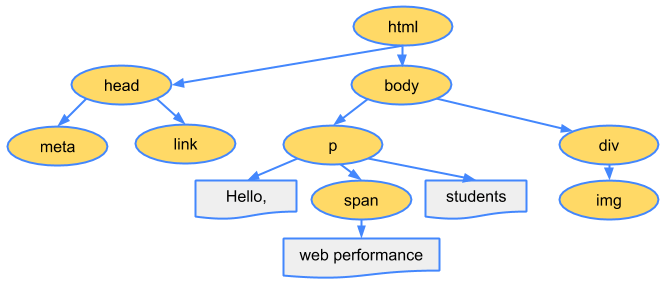
Перед тем как вывести страницу на экран, браузер создает модели DOM и CSSOM. Поэтому мы должны предоставить браузеру кратчайший путь к HTML и CSS.

Model DOM

Возьмем самую простую веб-станицу с текстом без форматирования и одним изображением и посмотрим, как она обрабатывается в браузере.



1. **Преобразование**. Браузер преобразует байты из HTML-файла, размещенного на диске или в сети, в символы, основываясь на приведенной в файле кодировке (например, UTF-8).
2. **Разметка**. На основании [стандарта W3C HTML5](http://www.w3.org/TR/html5/) браузер выделяет среди символов теги в угловых скобках, такие как <html>, <body> и другие. У каждого тега есть свое значение и свой набор правил.
3. **Создание объектов**. С помощью HTML-тегов браузер выделяет в документе объекты с определенными свойствами.
4. **Формирование DOM**. Объекты образуют древовидную структуру, повторяющую иерархию HTML-файла, в котором одни теги помещаются в другие. Так, объект *p* помещается под *body*, а объект *body*, в свою очередь, под *html*, и так далее.



В результате образуется объектная модель документа (DOM), с помощью которой браузер продолжает обрабатывать страницу.

Model CSSOM

При формировании DOM браузер обнаружил в документе ссылку на таблицу стилей (style.css). Поскольку она необходима для визуализации страницы, браузер мгновенно отправляет на сервер запрос и получает в ответ следующий код:

body { **font-size:** 16px }

p { **font-weight:** **bold** }

span { **color:** red }

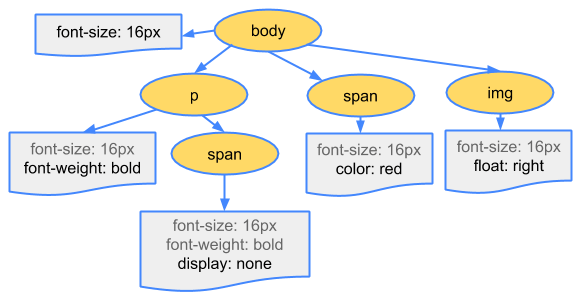
p span { **display:** **none** }

img { **float:** **right** }

Чтобы обработать данные из CSS-файла, браузер должен выполнить те же самые действия, что и с HTML-документом:



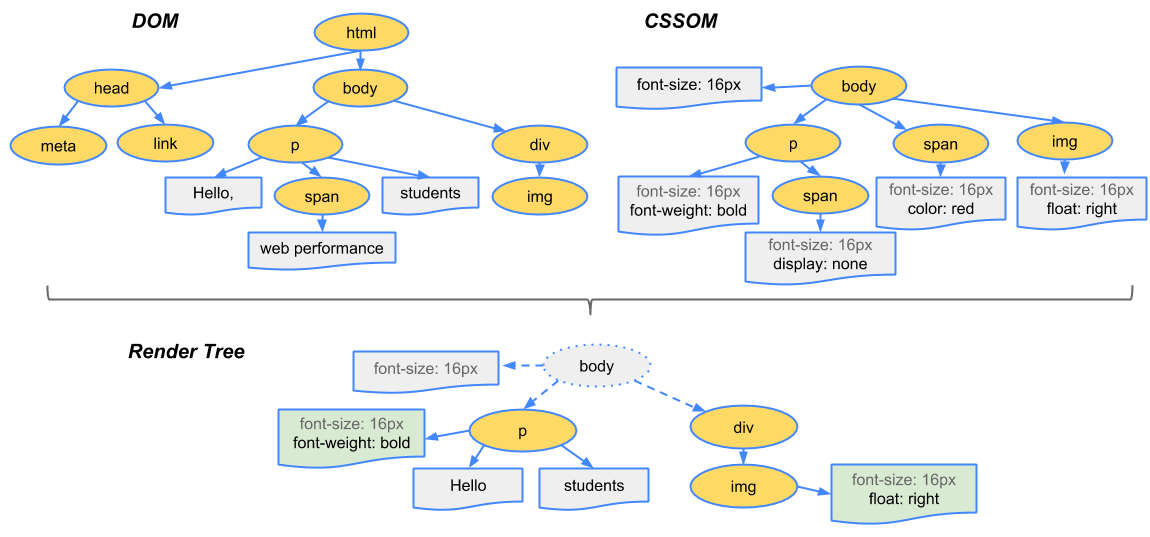
Байты из CSS-файла преобразуются в символы, символы - в теги, а теги - в объекты, которые образуют модель CSSOM:



Почему CSSOM имеет древовидную структуру? Сначала браузер присваивает объекту правила, характерные для его родительского элемента, а затем - характерные только для него. Таким образом получается каскадный набор стилей.

Обратите внимание, что схема отражает модель CSSOM не полностью. На ней представлены только правила, замещающие стили по умолчанию, которые применяются браузерами при отсутствии CSS-файла и соответствующей HTML-разметки.

DOM описывает контент, а CSSOM - стили, которые будут к нему применены. Рассмотрим, как браузер объединяет эти модели и выводит страницу на экран. Для начала браузер формирует модель визуализации, в которой всем видимым объектам из модели DOM присваиваются наборы стилей из модели CSSOM.



Для формирования модели визуализации браузер выполняет следующие действия:

1. Начиная с основания модели DOM, находит все видимые объекты.
   * Этот этап не затрагивает элементы, которые не будут видны на странице, например теги скриптов, метатеги и т. п.
   * Он также не затрагивает объекты, помеченные как невидимые с помощью CSS. Взгляните на приведенную выше схему: объект span отсутствует в модели визуализации, потому что ему присвоен параметр display: none.
2. Находит в CSSOM наборы стилей и присваивает их соответствующим объектам.
3. Формирует модель из видимых объектов, их содержания и стилей.

Браузер уже определил, какие объекты будут видны на странице и какие стили нужно им присвоить. Пришло время создать макет, т. е. выяснить, какого размера будут объекты и как их нужно расположить в [области просмотра](https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/set-the-viewport.html).

Для этого браузер вычисляет геометрическую форму объектов, анализируя модель визуализации с самого начала. Рассмотрим простой пример:

<html>

<head>

<meta name="viewport" content="width=device-width,initial-scale=1">

<title>Critial Path: Hello world!</title>

</head>

<body>

<div style="width: 50%">

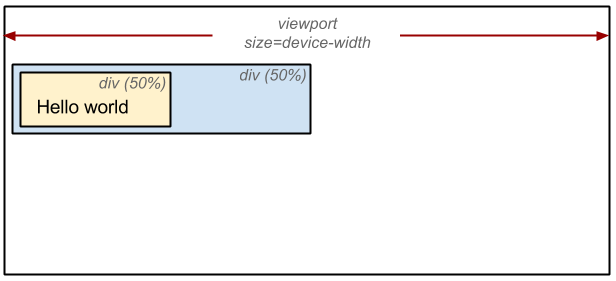
<div style="width: 50%">Hello world!</div>

</div>

</body>

</html>

В теле этой страницы есть два блока div. Ширина родительского блока - 50% от области просмотра, а вложенного - 50% от родительского, т. е. 25% экрана.



Сформировав макет, браузер получает блочную модель, точно отражающую расположение и размер каждого объекта в области просмотра. Все относительные показатели преобразуются в абсолютное положение пикселей на экране.

Наконец, когда браузеру известно, какие объекты будут отображаться на странице, где их разместить и какие стили им нужно присвоить, можно приступать к следующему этапу - выводу страницы на экран. Этот этап также называется визуализацией или растеризацией.