Model Tree

<https://developers.google.com/web/fundamentals/performance/rendering/?hl=ru>

<https://habrahabr.ru/post/224187/>

<https://habrahabr.ru/post/262239/>



* **JavaScript**. Обычно JavaScript используется для выполнения работы, результатом которой будут визуальные изменения, будь то функция jQuery animate, сортировка набора данных или добавление DOM-элементов на страницу. Однако вызывать визуальное изменение можно не только с помощью JavaScript: Также часто используются анимация CSS, переходы и API-интерфейс веб-анимации.
* **Вычисление стилей**. В процессе вычисления стилей определяется, какие правила CSS к каким элементам применяются с учетом соответствующих селекторов, например: .headlineили .nav > .nav\_\_item. Отсюда, после того как правила определены, они применяются и вычисляются итоговые стили для каждого элемента.
* **Расчет макета**. Как только браузер будет знать, какие правила применяются к элементу, он может начать вычислять, сколько места он займет, и где он находится на экране. Модель макета для Интернета означает, что один элемент может влиять на другие, например: ширина элемента <body> обычно влияет на значения ширины дочерних элементов и так далее по всему дереву, поэтому этот процесс для браузера может быть довольно сложным.
* **Прорисовка**. Прорисовка – это процесс заполнения пикселей. Он подразумевает вывод текста, цветов, изображений, границ и теней, по сути – всех визуальных частей элементов. Прорисовка обычно выполняется на нескольких поверхностях, которые называются слоями.
* **Компоновка**. Поскольку части страницы потенциально были прорисованы на нескольких слоях, они должны быть выведены на экран в надлежащем порядке, с тем чтобы страница отображалась правильно. Это особенно важно для элементов, которые перекрывают другие элементы, поскольку ошибка может привести к тому, что один элемент будет неправильно показан поверх другого элемента.

 Из полученного от сервера HTML-документа формируется DOM (Document Object Model).

 Загружаются и распознаются стили, формируется CSSOM (CSS Object Model).

 На основе DOM и CSSOM формируется дерево рендеринга, или render tree — набор объектов рендеринга (Webkit использует термин «renderer», или «render object», а Gecko — «frame»). Render tree дублирует структуру DOM, но сюда не попадают невидимые элементы (например — <head>, или элементы со стилем display:none;). Также, каждая строка текста представлена в дереве рендеринга как отдельный renderer. Каждый объект рендеринга содержит соответствующий ему объект DOM (или блок текста), и рассчитанный для этого объекта стиль. Проще говоря, render tree описывает визуальное представление DOM.

 Для каждого элемента render tree рассчитывается положение на странице — происходит layout. Браузеры используют поточный метод (flow), при котором в большинстве случаев достаточно одного прохода для размещения всех элементов (для таблиц проходов требуется больше).

 Наконец, происходит отрисовка всего этого добра в браузере — painting.

1. Получить HTML-документ.
2. Провести парсинг HTML на предмет включенных ресурсов.
3. Построить DOM tree (document object model).
4. Отправить запросы критических элементов.
5. Получить весь CSS-код (также запустить запросы на JS-файлы).
6. Построить CSSOM tree.
7. Выполнить весь полученный JS-код.
8. Перестроить DOM tree (при необходимости).
9. Построить Render tree и начать отрисовку страницы.

Советы:

* Пишите валидный HTML и CSS, с указанием кодировки. Стили лучше включать в <head>, а скрипты — в конце <body>.
* Стремитесь упрощать и оптимизировать селекторы CSS (этим часто пренебрегают разработчики, использующие препроцессоры). Чем меньше вложенность — тем лучше. По эффективности обработки селекторы можно расположить в следующем порядке (начиная с наиболее быстрого):  
  1. Идентификатор: #id
  2. Класс: .class
  3. Тэг: div
  4. Соседний селектор: a + i
  5. Дочерний селектор: ul > li
  6. Универсальный селектор: \*
  7. Селектор атрибутов: input[type="text"]
  8. Всевдоэлементы и псевдоклассы: a:hover

Следует помнить, что браузер обрабатывает селекторы справа налево, поэтому в качестве ключевого (крайнего правого) селектора лучше использовать наиболее эффективные — идентификатор и класс.

**div** \* {...} // плохо

.list **li** {...} // плохо

.list-item {...} // хорошо

**#list** .list-item {...} // хорошо

* В скриптах минимизируйте любую работу с DOM. Кэшируйте всё: свойства, объекты, если подразумевается повторное их использование. При сложных манипуляциях разумно работать с «offline» элементом (т.е. который находится не в DOM, а в памяти), с последующим помещением его в DOM.
* При использовании jQuery для выборки элементов придерживайтесь[рекомендаций по составлению селекторов](http://learn.jquery.com/performance/optimize-selectors/).
* Для изменения стилей элементов лучше модифицировать только атрибут «class», и как можно глубже в дереве DOM, это и более грамотно с точки зрения разработки и поддержки (отделение логики от представления), и менее затратно для браузера.
* Анимировать желательно только абсолютно и фиксировано спозиционированные элементы.
* Можно отключать сложные :hover анимации во время скроллинга (например, добавляя к body класс «no-hover»)