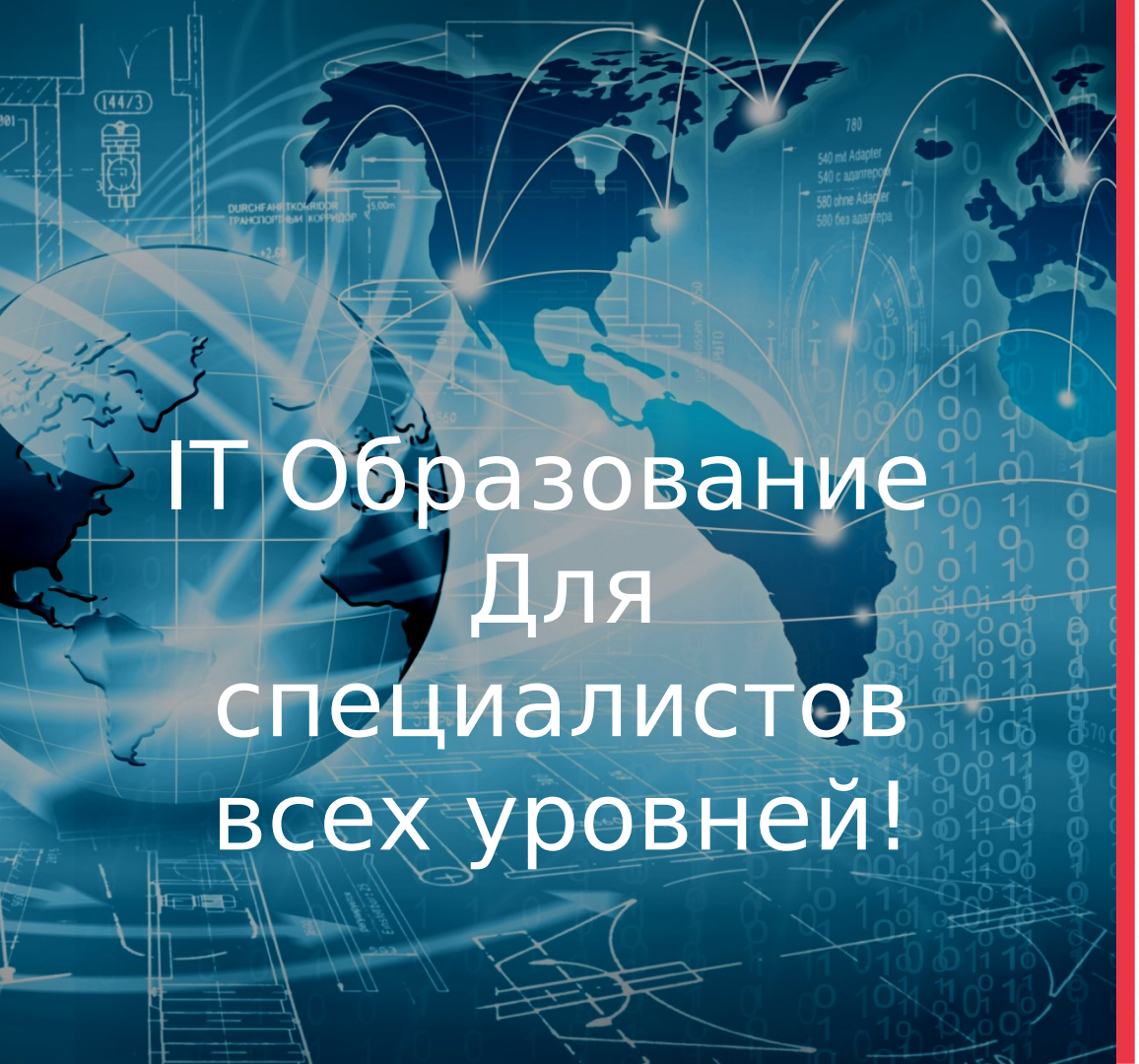


IT Образование Для специалистов всех уровней!

Education
Academy

WWW.ITEDUCATE.COM.UA

JavaScript Pro

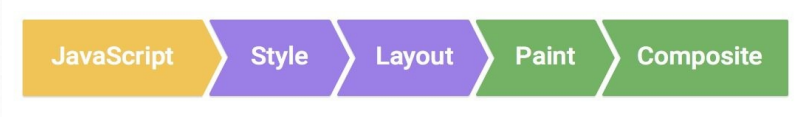
60 кадров в секунду

Конвейер пикселей. Оптимизация

Bad performance

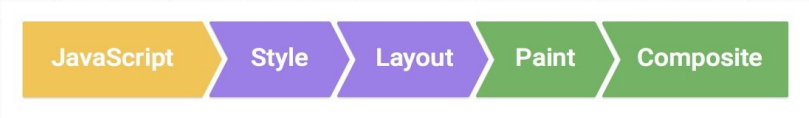
Производительность – это искусство избегать работы и делать любую работу как можно более эффективно.

Конвейер пикселей



- **JavaScript**. Обычно JavaScript используется для выполнения работы, результатом которой будут визуальные изменения, будь то функция jQueryanimate, 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: .headline или .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 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.** 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.

1. JS / CSS > Стиль > Расчет макета > Прорисовка > Компоновка



- Если изменить свойство, относящееся к макету, например такое свойство геометрии элемента, как ширина, высота или положение относительно левого верхнего угла, браузеру придется проверить все остальные элементы и перерасчитать дерево отрисовки страницы. Все затронутые области необходимо будет прорисовать заново, а итоговые элементы нужно будет снова скомпоновать.
- ***(Layout > Paint > Composite) - “reflow”***

2. JS / CSS > Стиль > Прорисовка > Компоновка

- Если изменить свойство, которое связано только с прорисовкой, например фоновое изображение, цвет текста или его тень, другими словами, свойство, которое не влияет на макет страницы, браузер оставит макет неизменным, но ему все равно придется выполнить прорисовку.
- ***Paint > Composite - “repaint”***

2. JS / CSS > Стиль > Прорисовка > Компоновка

- Если же изменить свойство, которое не требует ни перерасчета макета, ни прорисовки, браузер сразу же перейдет к компоновке.
- Последний вариант является самым простым и наиболее предпочтительным для точек высокой нагрузки в жизненном цикле приложения, например при анимации или прокрутке.
- http://csstriggers.com/

60 кадров в секунду и частота обновления экрана устройства

Сегодня большинство устройств обновляют свои экраны 60 раз в секунду. Если выполняется анимация или переход либо если пользователь прокручивает страницы, браузеру нужно соответствовать частоте обновления экрана устройства и выдавать по одной новой картинке (или кадру) при каждом обновлении экрана.

Каждый из этих кадров может длиться чуть более 16 мс (1 секунда / 60 = 16,66 мс). В реальности же браузеру нужно выполнить и еще кое-какие действия, потому вся ваша работа должна занимать не более 10 мс. Если не уложиться в эти рамки, частота кадров будет меньше, а контент начнет дергаться на экране. Часто эту ситуацию называют подвисанием, она отрицательно сказывается на восприятии пользователей.

Оптимизация выполнения JavaScript

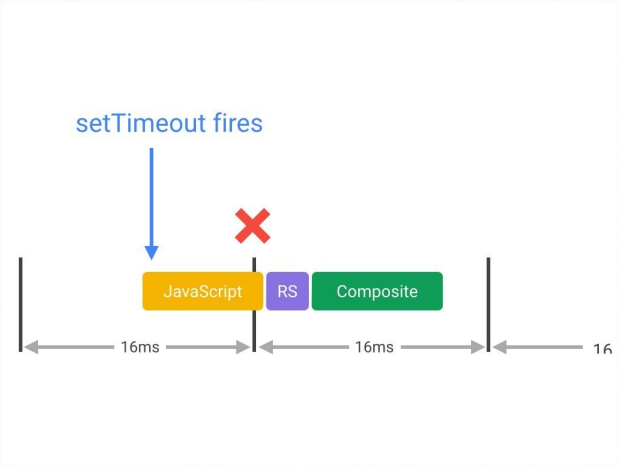
JavaScript часто используется для внесения визуальных изменений. Иногда это делается непосредственно путем переработки стилей, в других же случаях визуальные изменения являются результатом определенных вычислений, например поиска или сортировки тех или иных данных. Неправильно выбранные параметры времени и продолжительности выполнения JavaScript часто являются причиной проблем с производительностью, поэтому при любой возможности следует стараться свести влияние этого кода к минимуму

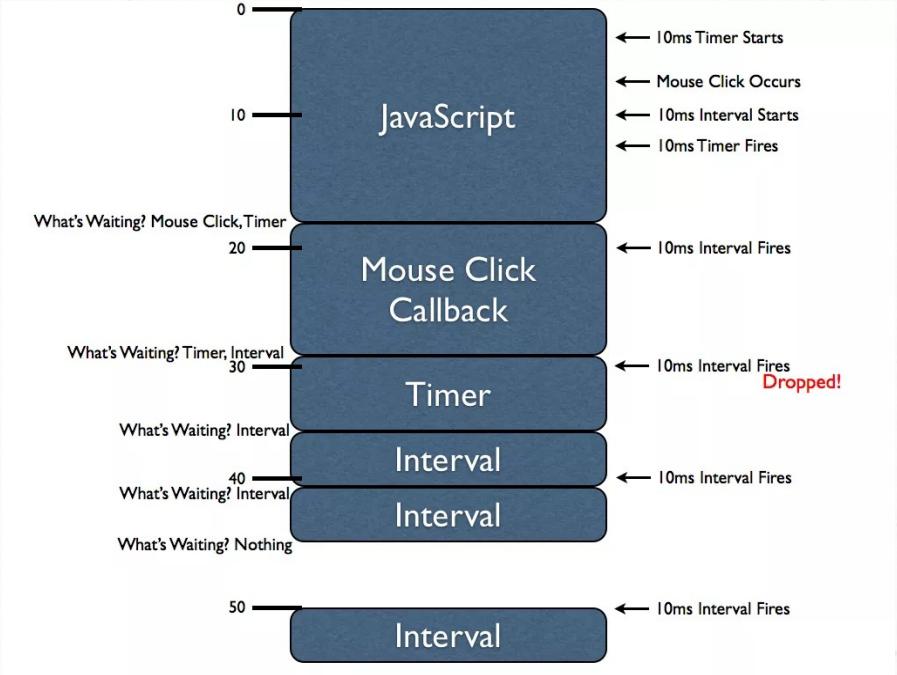
Оптимизация выполнения JavaScript

- Не используйте функции setTimeout или setInterval для внесения визуальных изменений; вместо этого всегда пользуйтесь функцией requestAnimationFrame.
- Перемещайте скрипты JavaScript, которые выполняются долго, за пределы основного потока в рабочие веб-процессы Web Worker.
- Для внесения изменений в DOM-элементы за несколько кадров используйте микрозадачи.
- Для оценки влияния JavaScript используйте шкалу времени и средство профилирования JavaScript из Chrome DevTools.

Оптимизация выполнения JavaScript

- **requestAnimationFrame**
- Когда на экране происходят визуальные изменения, свою работу нужно выполнять в подходящее время для браузера, а именно – в самом начале кадра. Единственным способом гарантировать выполнение кода JavaScript в начале кадра является использование функции requestAnimationFrame.

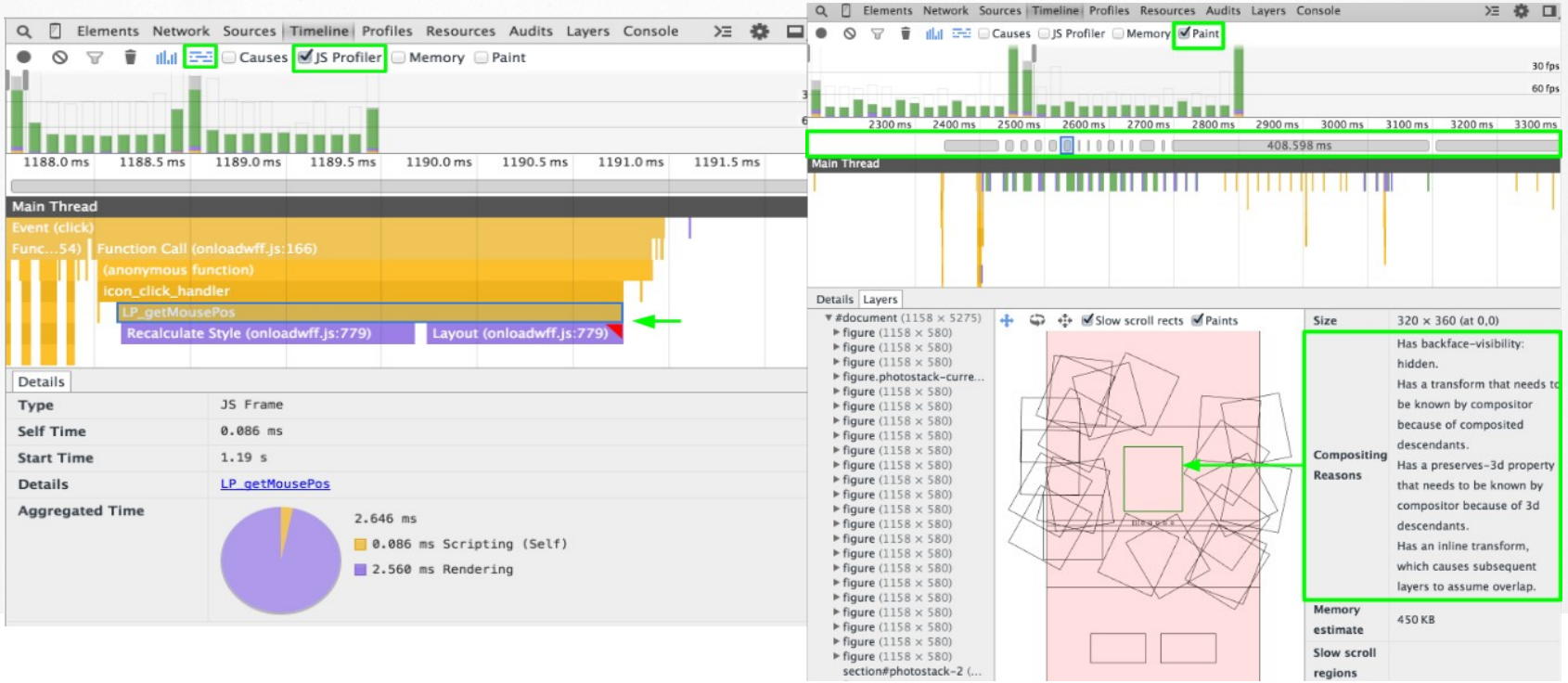



0
10
20
30
40
50
JavaScript
Mouse Click Callback
Timer
Interval
Interval
Interval
10ms Timer Starts
Mouse Click Occurs
10ms Interval Starts
10ms Timer Fires
10ms Interval Fires
10ms Interval Fires
Dropped!
10ms Interval Fires
10ms Interval Fires
What's Waiting? Mouse Click, Timer
What's Waiting? Timer, Interval
What's Waiting? Interval
What's Waiting? Interval
What's Waiting? Nothing

Перемещайте скрипты JavaScript, которые выполняются долго, за пределы основного потока в рабочие веб-процессы Web Worker.

- JavaScript выполняется в основном потоке браузера вместе с вычислением стилей, макета и, во многих случаях, прорисовкой. Если ваш код JavaScript выполняется в течение длительного времени, он заблокирует все эти задачи, что может привести к пропуску кадров.
- Следует тактически грамотно выбирать время и продолжительность выполнения JavaScript. Например, если выполняется такая анимация, как прокрутка, идеальным будет выполнение JavaScript в течение первых 3–4 мс. Чуть дольше – и вы рискуете занять слишком много времени. Если же в данный момент никаких действий не выполняется, то за временем работы можно позволить себе следить не так строго.
- Во многих случаях чисто вычислительную работу можно переместить в рабочие веб-процессы (Web Worker), если, например, для нее не требуется доступ к DOM. Обработка данных или такие промежуточные состояния, как сортировка или поиск, нередко хорошо подходят для этой модели, как и загрузка или формирование моделей.
- **Не вся работа годится для этой модели: у рабочих веб-процессов Web Worker нет доступа к DOM**

- Знайте, как ваш код JavaScript влияет на кадры



Ограничение объема и сложности вычисления стилей

Приблизительно 50 % времени, которое тратится на вычисление стиля элемента, уходит на сопоставление селекторов, а вторую половину времени занимает построение RenderStyle (представления стиля) на основе сопоставленных правил.

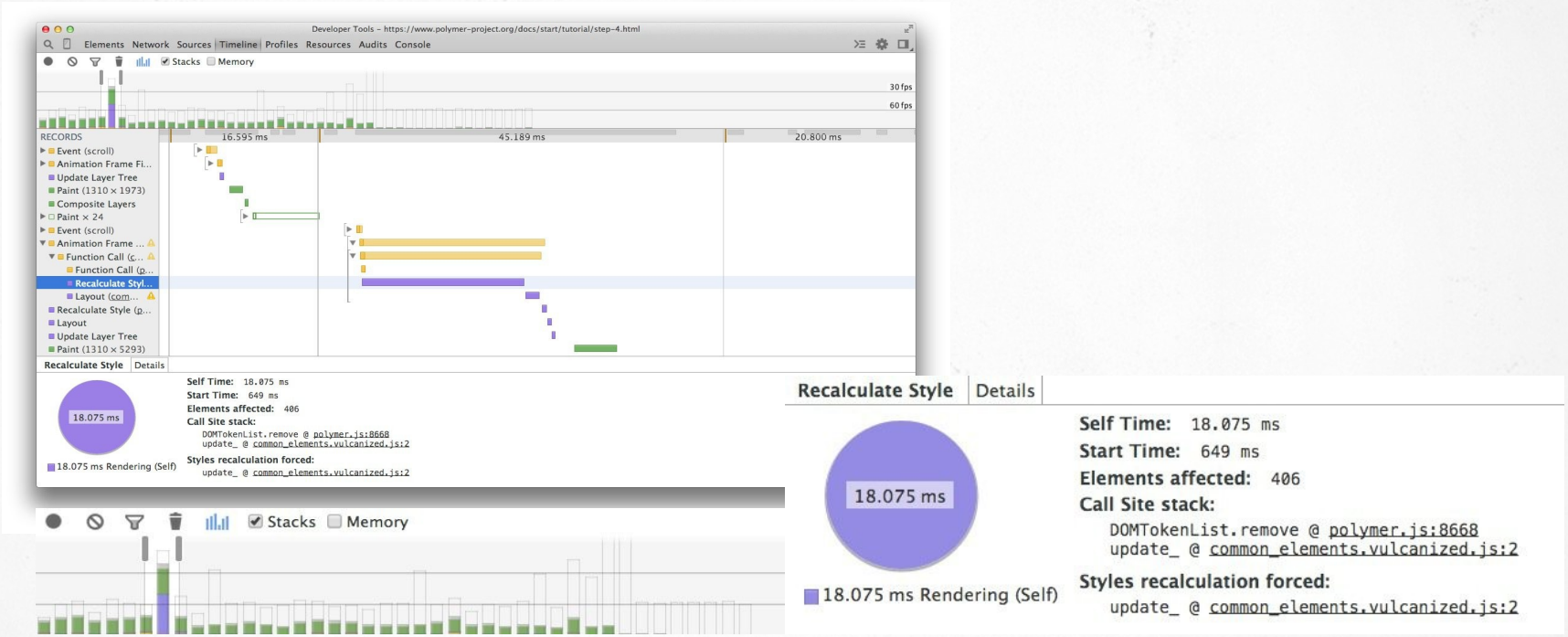
```
.box:nth-last-child(-n+1) .title {
  /* styles */
}
```

```
.final-box-title {
  /* styles */
}
```

BEM

```
.list__list-item--last-child {}
.list { }
.list__list-item { }
```

Ограничение объема и сложности вычисления стилей



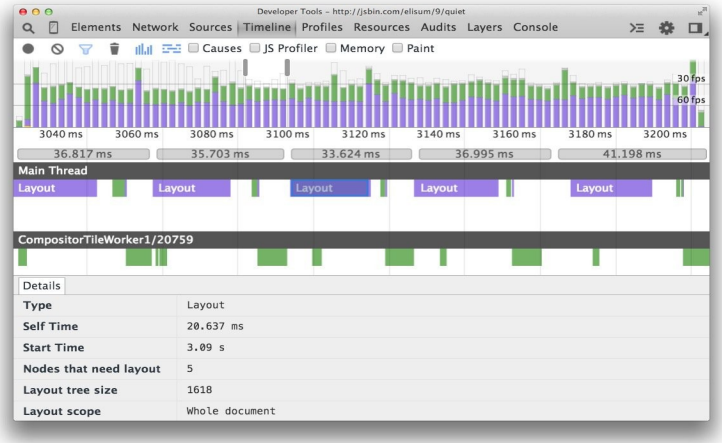
Не используйте большие сложные макеты и избегайте подтормаживания макетов

Макет — это источник, из которого браузеры получают информацию о геометрии элементов: об их размере и расположении на странице. Для каждого элемента будет предоставлена явная или скрытая информация о размере, основанная на использовавшемся CSS, содержимом элемента или о родительском элементе. В браузерах Chrome, Opera, Safari и в Internet Explorer этот процесс называется Layout (Перерасчет макета). В Firefox он называется Reflow (Перерасчет дерева отрисовки), но по сути это один и тот же процесс

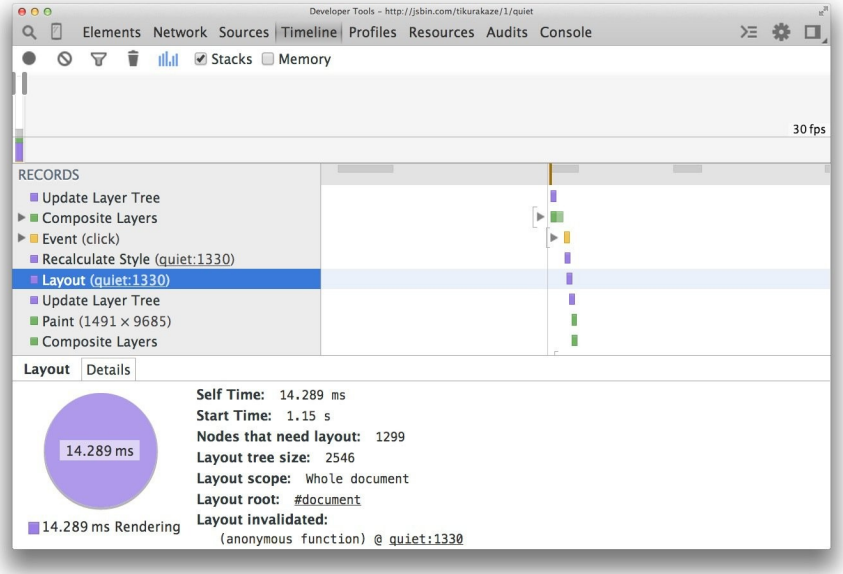
При изменении стилей браузер проверяет, требуется ли для какого-либо из изменений перерасчет дерева визуализации. Изменение всех “геометрических свойств”, таких как width, height, left, или top влечет за собой перерасчет макета.

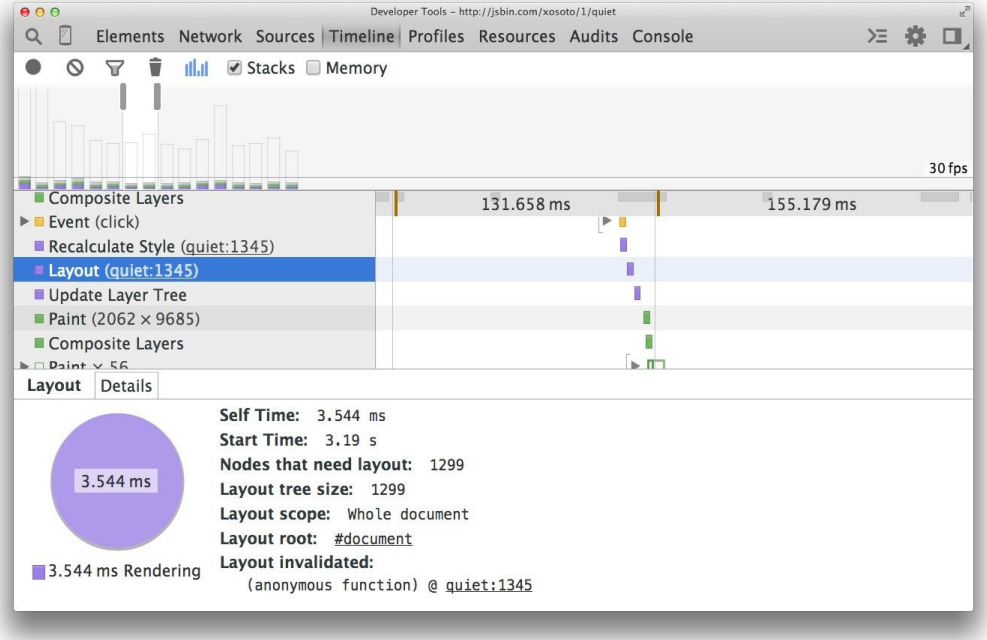
Не используйте большие сложные макеты и избегайте подтормаживания макетов

Перерасчет макета почти всегда выполняется для всего документа. Если элементов много, на определение мест расположения и размеров всех этих элементов потребуется длительное время. Если избежать перерасчета макета невозможно, то, опять же, необходимо с помощью Chrome DevTools посмотреть, сколько времени займет эта операция, и определить, является ли эта она причиной проблем с производительностью.

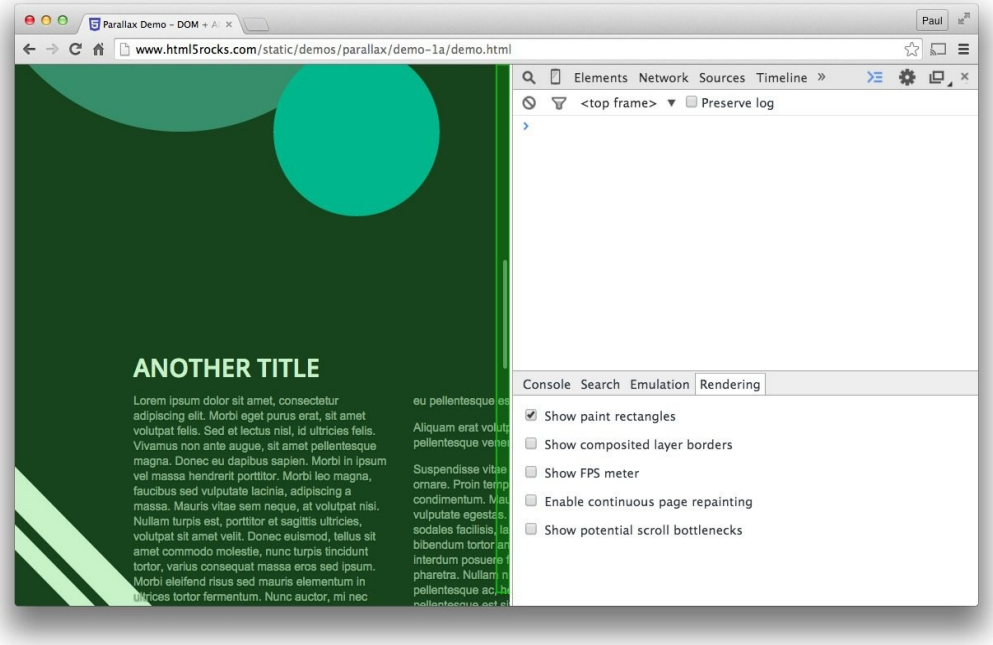


Используйте модуль flexbox для старых моделей макета





Для быстрого определения проблем с прорисовкой используйте Chrome DevTools



Перемещайте на отдельные слои элементы, которые двигаются или исчезают с экрана

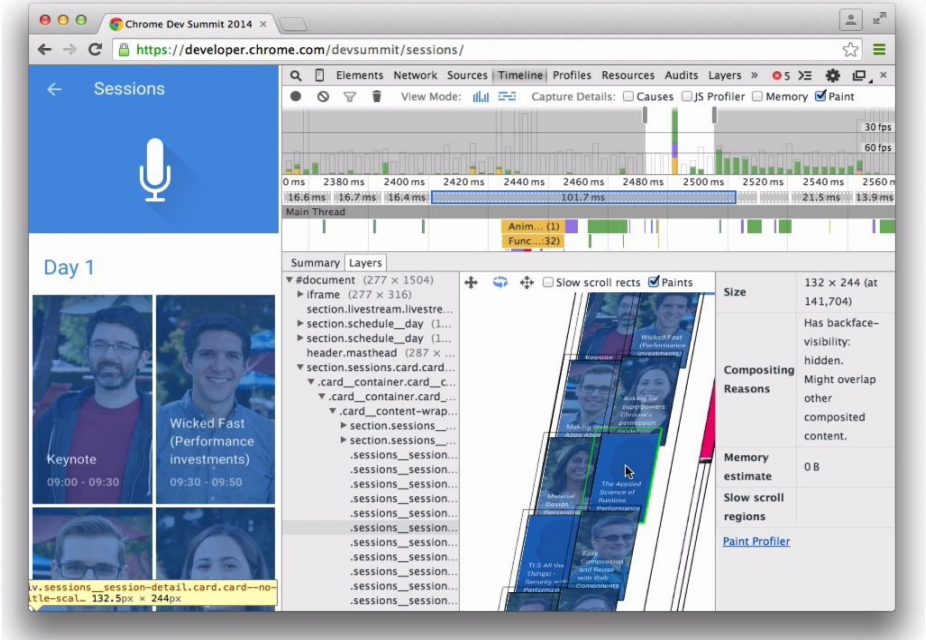
Используйте свойства, вызывающие только компоновку, и контролируйте количество слоев

Для достижения анимационного эффекта изменяйте свойства transform и opacity

Position	`transform: translate(npx, npx);`
Scale	`transform: scale(n);`
Rotation	`transform: rotate(ndeg);`
Skew	`transform: skew(X|Y)(ndeg);`
Matrix	`transform: matrix(3d)(...);`
Opacity	`opacity: 0...1;`

(The element will need to be on its own compositor layer.)

Используйте свойства, вызывающие только компоновку, и контролируйте количество слоев



Оптимизация обработчиков ввода

```
function onScroll (evt) {
  // запомните позицию скролла
  lastScrollY = window.scrollY;
  // Prevent rAF callbacks.
  if (scheduledAnimationFrame)
    return;
  scheduledAnimationFrame = true;
  requestAnimationFrame(readAndUpdatePage);
}
window.addEventListener('scroll', onScroll);
```