Цена пропущенного фрейма

Дмитрий Шуранов, Туту.ру

О себе

Дмитрий Шуранов

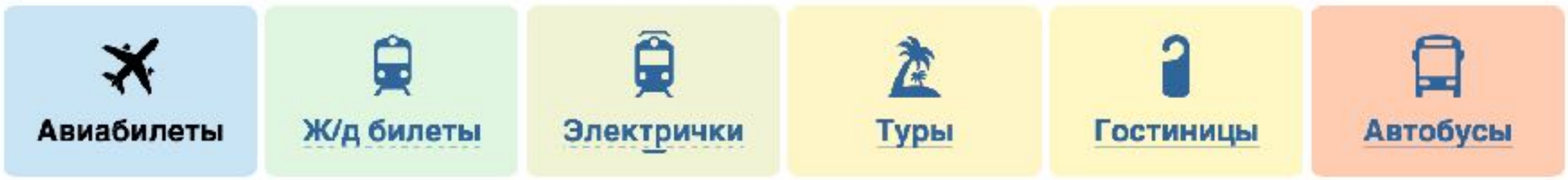
Фронтэнд-разработчик в Туту.ру

shuranov@tutu.ru

dvshur@gmail.com

Мы продаём ж/д, авиа и автобусные билеты, туры, бронируем отели и рассказываем о расписании

Самый посещаемый сервис туристических услуг в России (по версии comScore).



600 тыс.
посетителей в день

2003 г.
год основания

11 млн.
посетителей в месяц

280
сотрудников

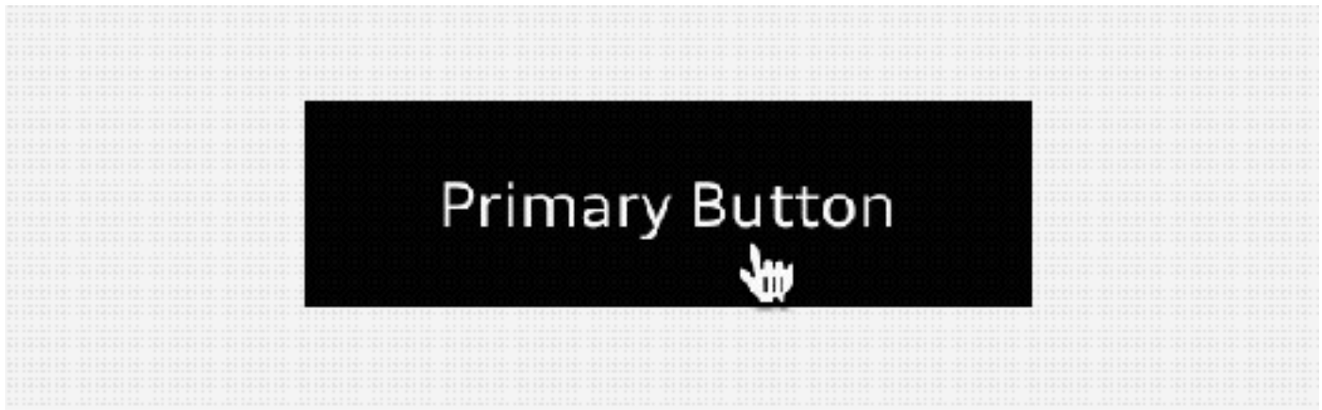
О чём пойдёт речь

1. Роль анимации в UX
2. Основы рендеринга
3. Видимая плавность и FPS
4. Что происходит при пропуске кадров
5. Подходы к анимации
6. Инструменты разработки и отладки анимаций

Роль анимации в UX

animation by Doretti Nicolas

Видимый отклик



Действие порождает движение, как и в материальном мире

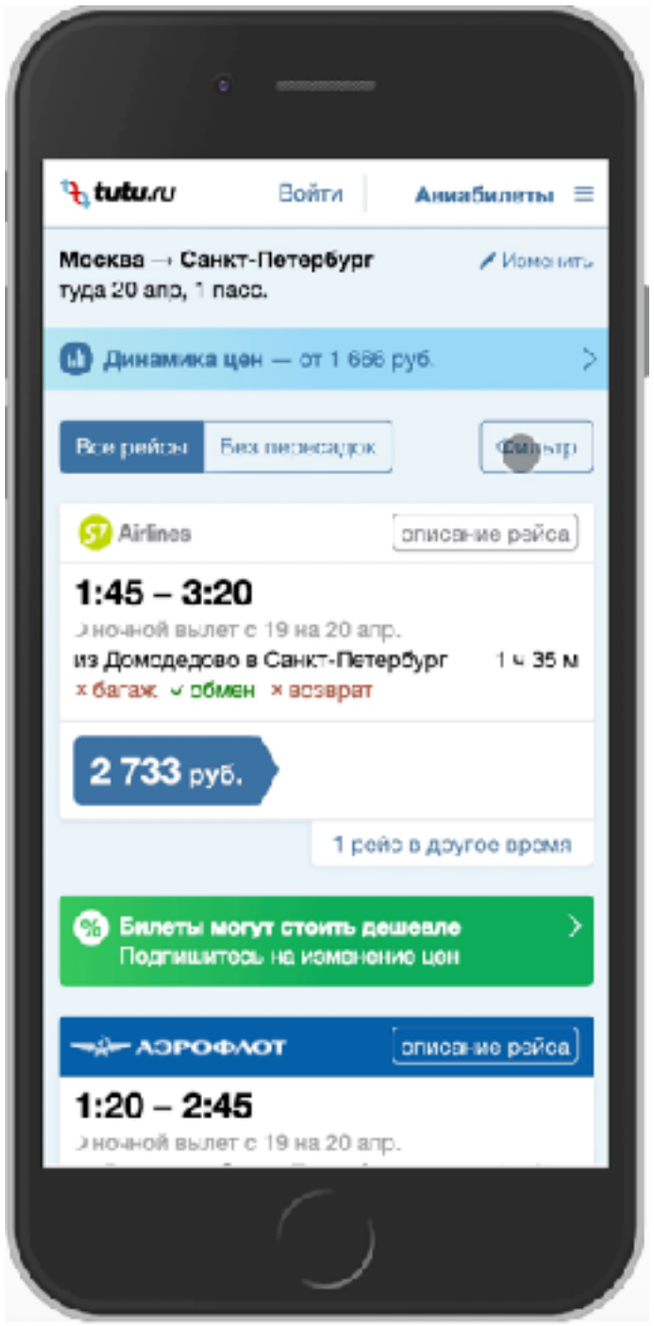
Интуитивное восприятие

Понятные жесты, передающие смысл без пояснений

Направление внимания

Управление вниманием пользователя

Видимость быстродействия



Маскировка фоновых процессов, вычислений

Плохая анимация не достигнет цели

И может вызвать негатив от «тормозов»

Продукты конкурируют на уровне UX

Хороший UX выделяет продукт среди конкурентов

Нужно знать меру

Слишком много анимаций — визуальный мусор

Нужна ли эта анимация?

1. Имеет конкретную UX-цель

2. Выиграла в АБ-кампании

3. Показала хорошие поведенческие метрики

Мобильные приложения

Touch-устройства больше

нуждаются в анимации

Веб-приложения могут конкурировать с нативными,

если их делать правильно

Сейчас клиенты и заказчики в это мало верят

К основам

К основам

К основам

- Layout и Paint — дорогие

К основам

- Layout и Paint — дорогие
- обычно **Composite** — на GPU

К основам

- Layout и Paint — дорогие
- обычно **Composite** — на GPU
- только `transform` и `opacity` — не задействуют layout/paint

К основам

На самом деле всё несколько сложнее

А ещё по-разному в разных браузерах

- Anatomy of a frame — bit.ly/2rrkUeX
- How browsers work — bit.ly/2ec5KmQ

Браузерные особенности

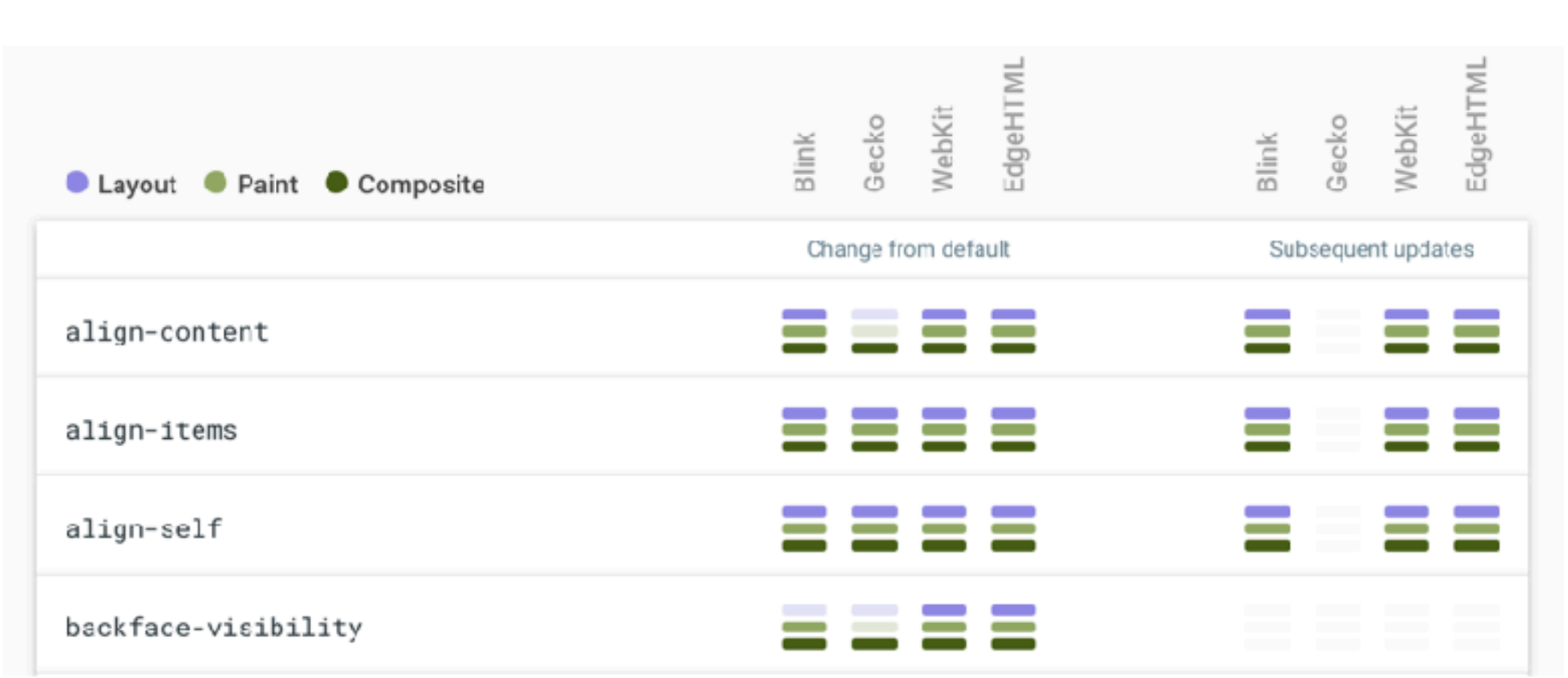
- Многие браузеры отдают Composite-only свойства на GPU (но не всегда, чаще из CSS)
- Хром не только Composite передаёт в отдельный тред, но и Paint
    - Полезно для box-shadow (material design) и пр.
- В Firefox в некоторых случаях scale вызывал repaint
- В Chrome scale иногда вызывает размазанные шрифты

К основам

- Все особенности (оптимизации) не удержать в голове, и они меняются постоянно
- Стоит делать придерживаться общих хороших практик, которые работают универсально
- Нужно профилировать и тестировать
  - хотя бы в одном браузере, кроме Chrome

К основам

- Какие CSS-свойства на что влияют
  - csstriggers.com



Плавность и FPS

Сколько FPS — достаточно?

- В фильмах — 24 FPS
- В компьютерных играх — 60+ FPS

Почему?

Сколько FPS — достаточно?

- В фильмах — 24 FPS
- В компьютерных играх — 60+ FPS

Почему?

Естественный **motion blur**

Motion blur

Without Motion Blur - 15FPS

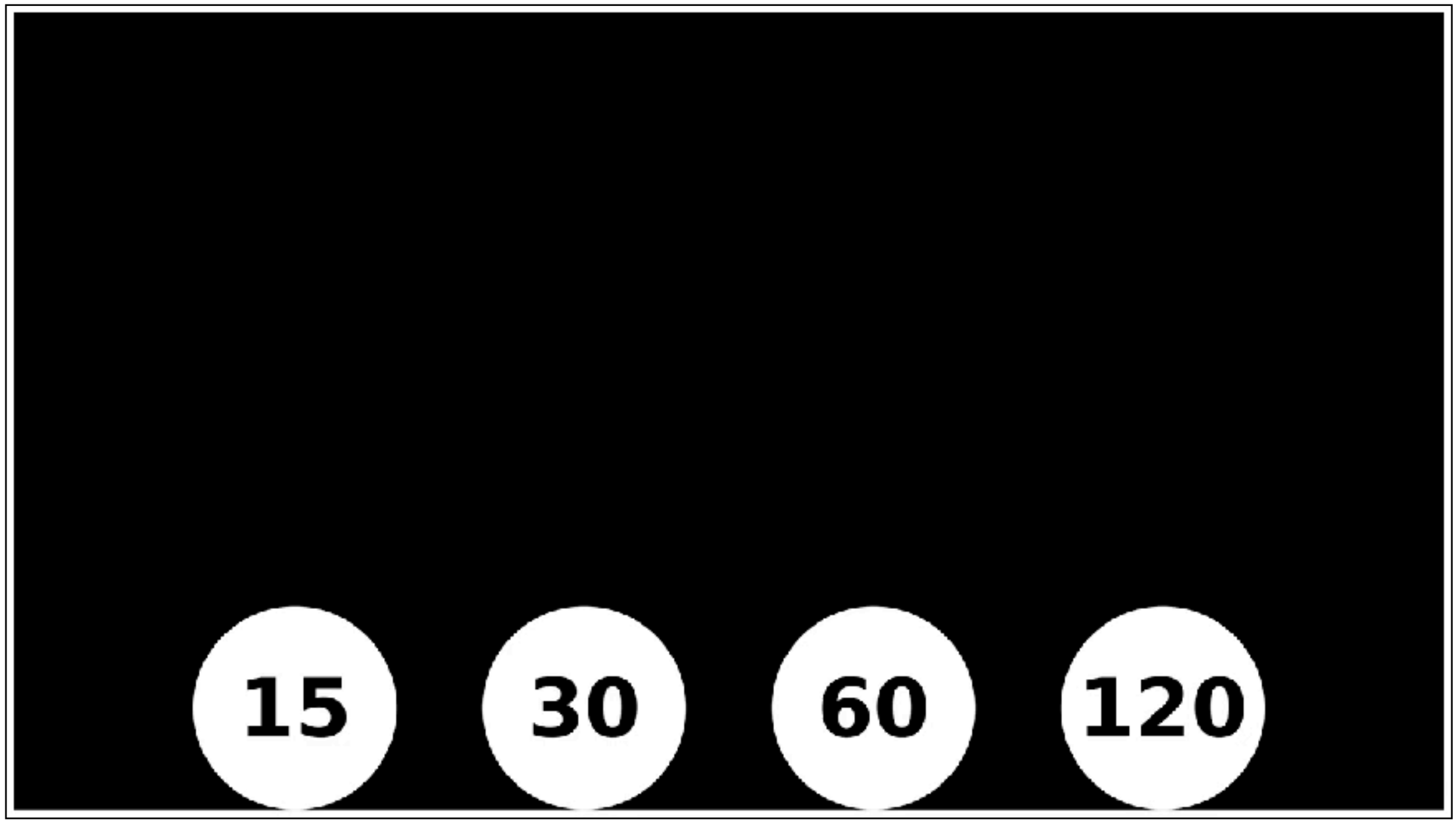
With Motion Blur - 15FPS

Плавность и FPS

- Чем резче переход, тем лучше его видит глаз
- В компьютерной анимации нет естественного motion blur

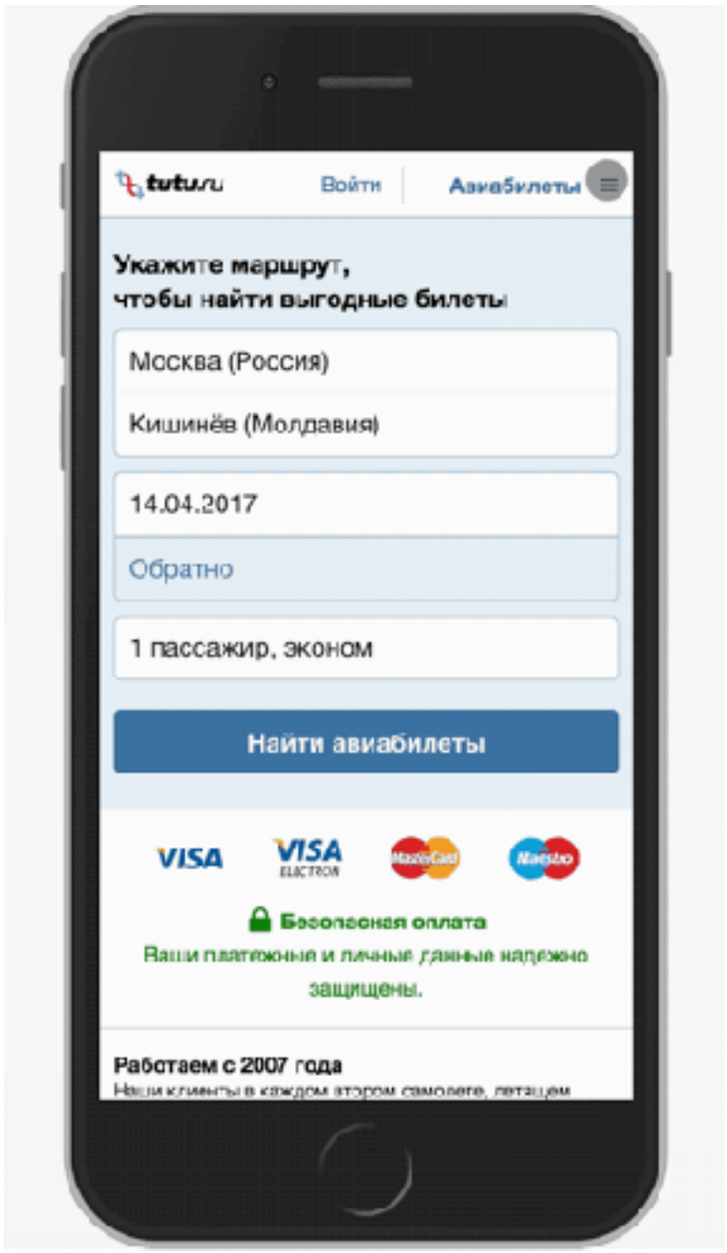
Стоит стремиться к максимуму FPS — сколько даёт устройство. Обычно 60 Hz -> **60 FPS**.

Плавность и FPS



Что происходит при пропуске фреймов

Цена пропущенного фрейма



Немного математики

300 px / 0.5 s / 60 Hz =

10px/frame

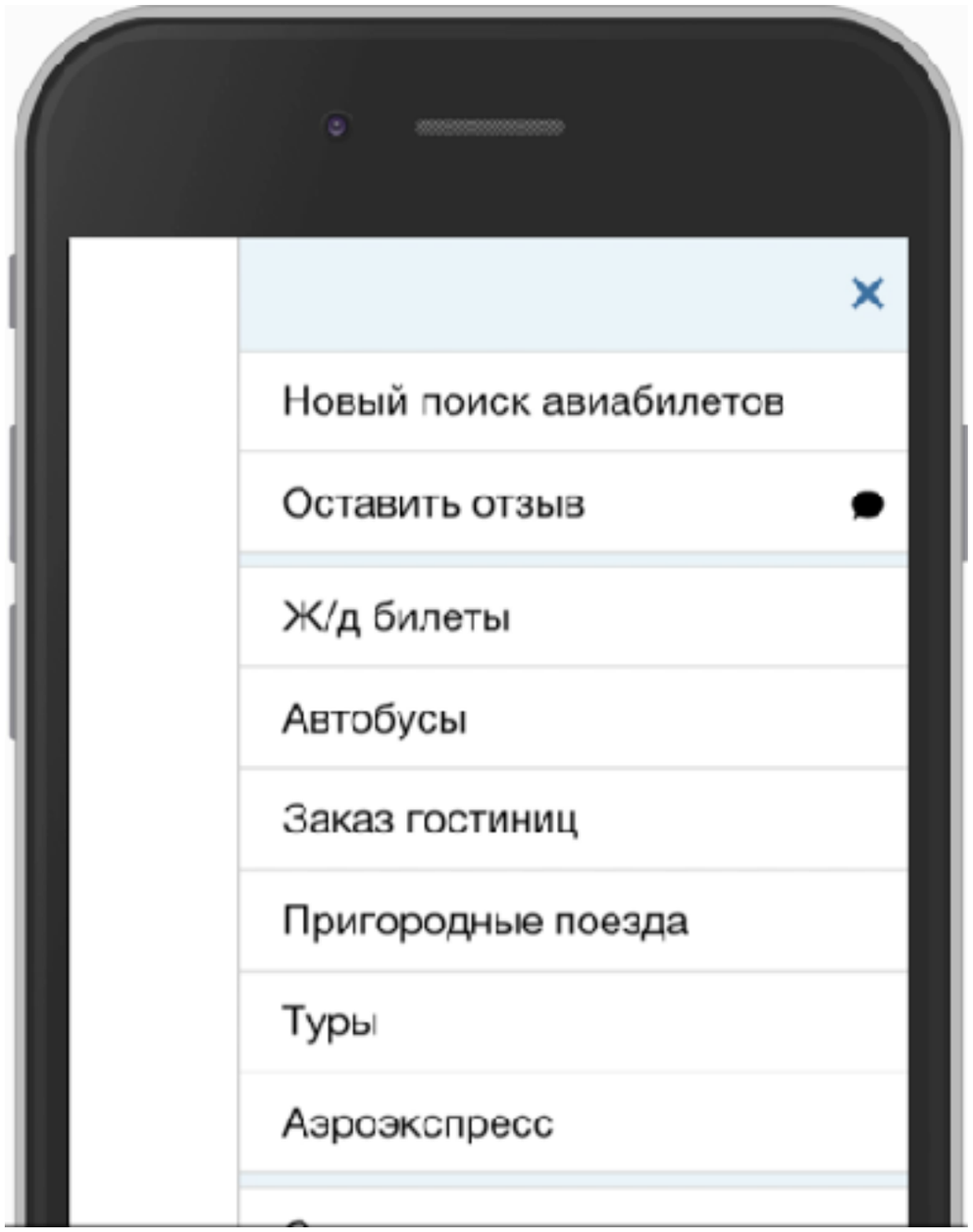
1 пропущенный фрейм — скачок на 20 пикселей

Цена пропущенного фрейма

Дёрганая анимация

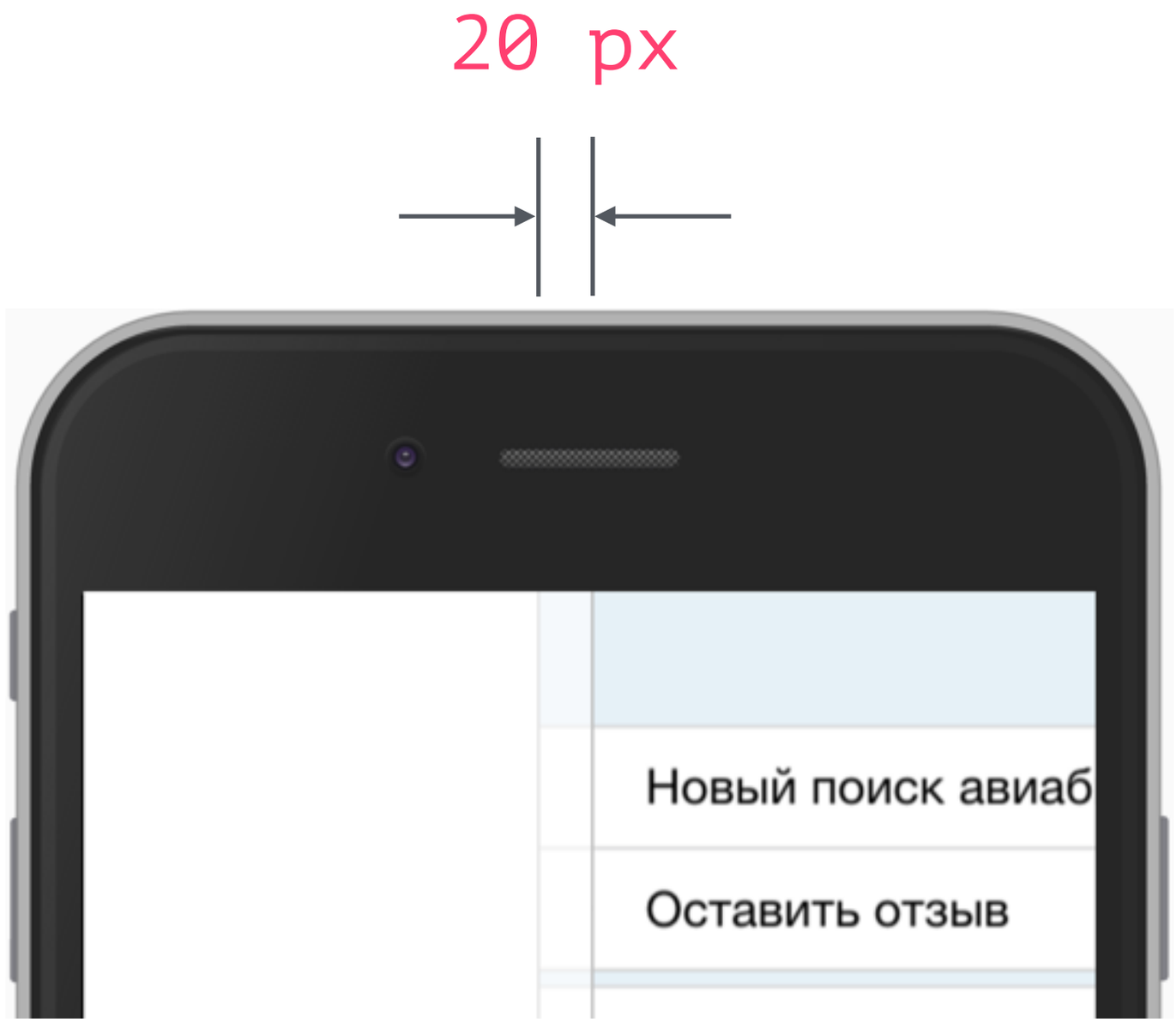
2 проседания FPS

Цена пропущенного фрейма

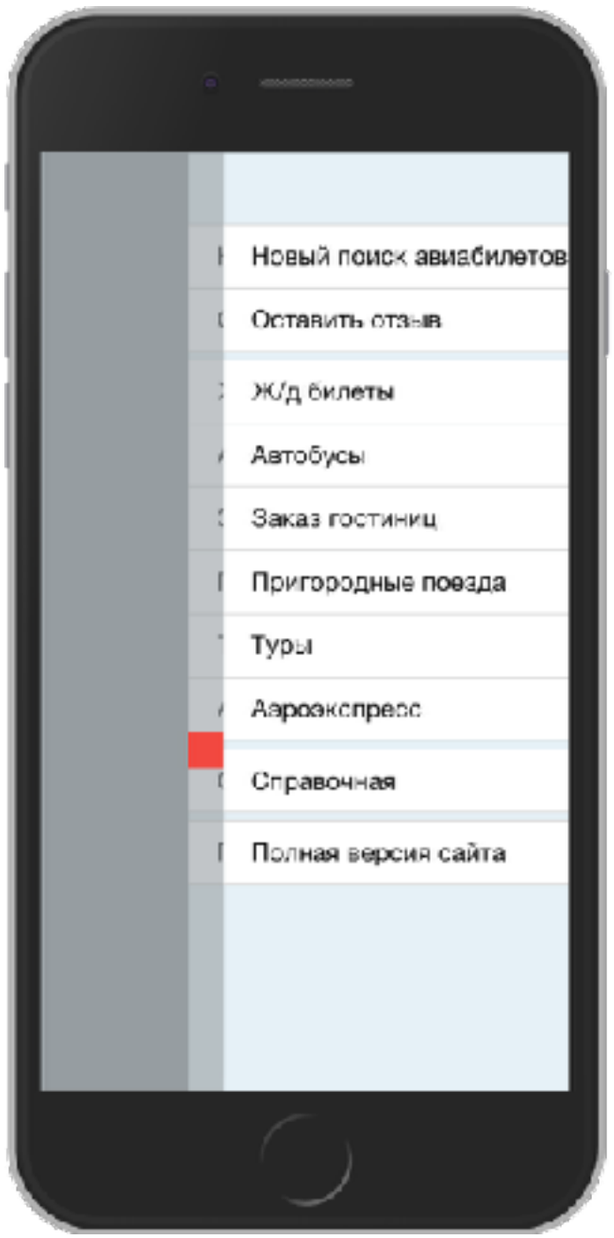


Замедлим для наглядности

Цена пропущенного фрейма

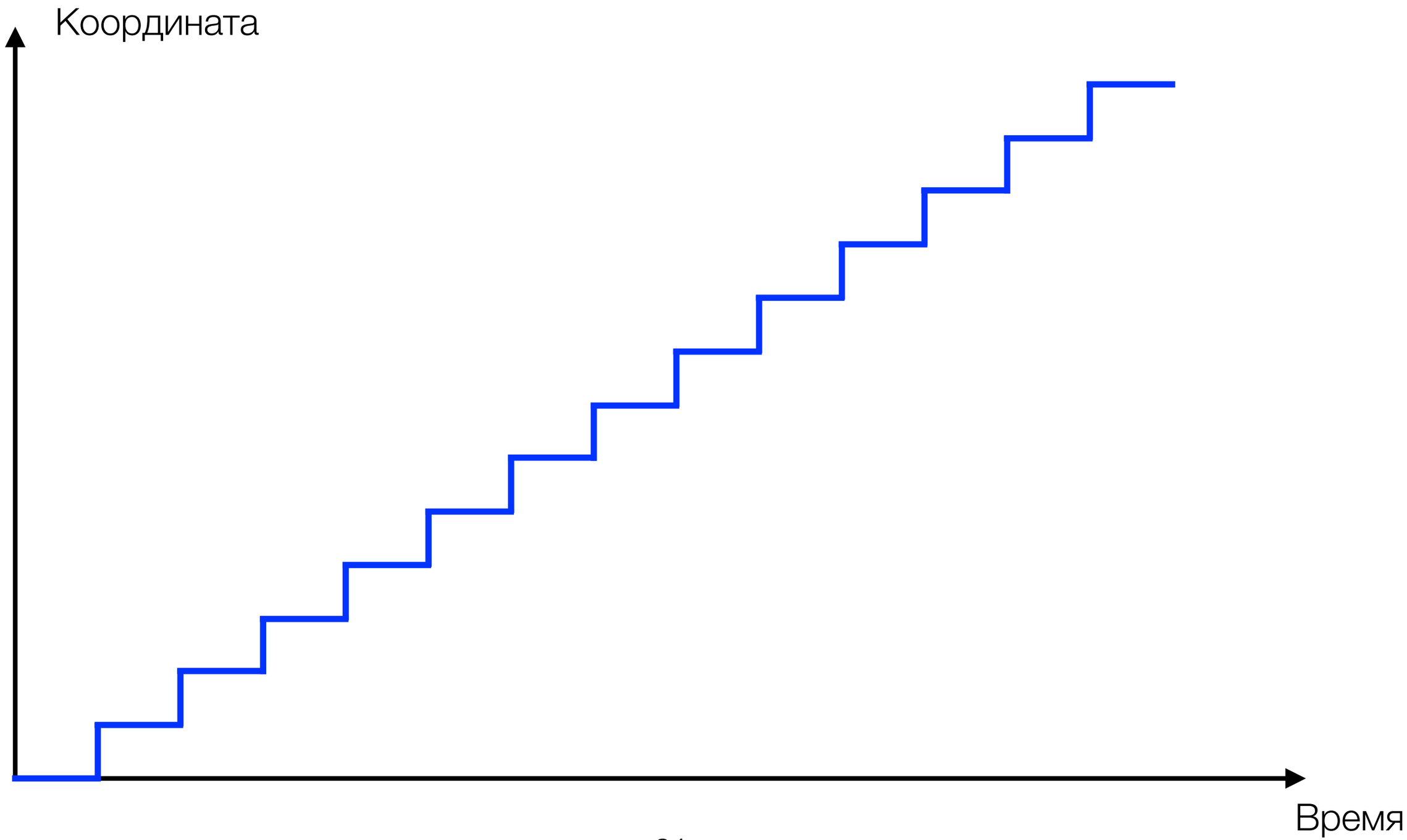


Цена пропущенного фрейма

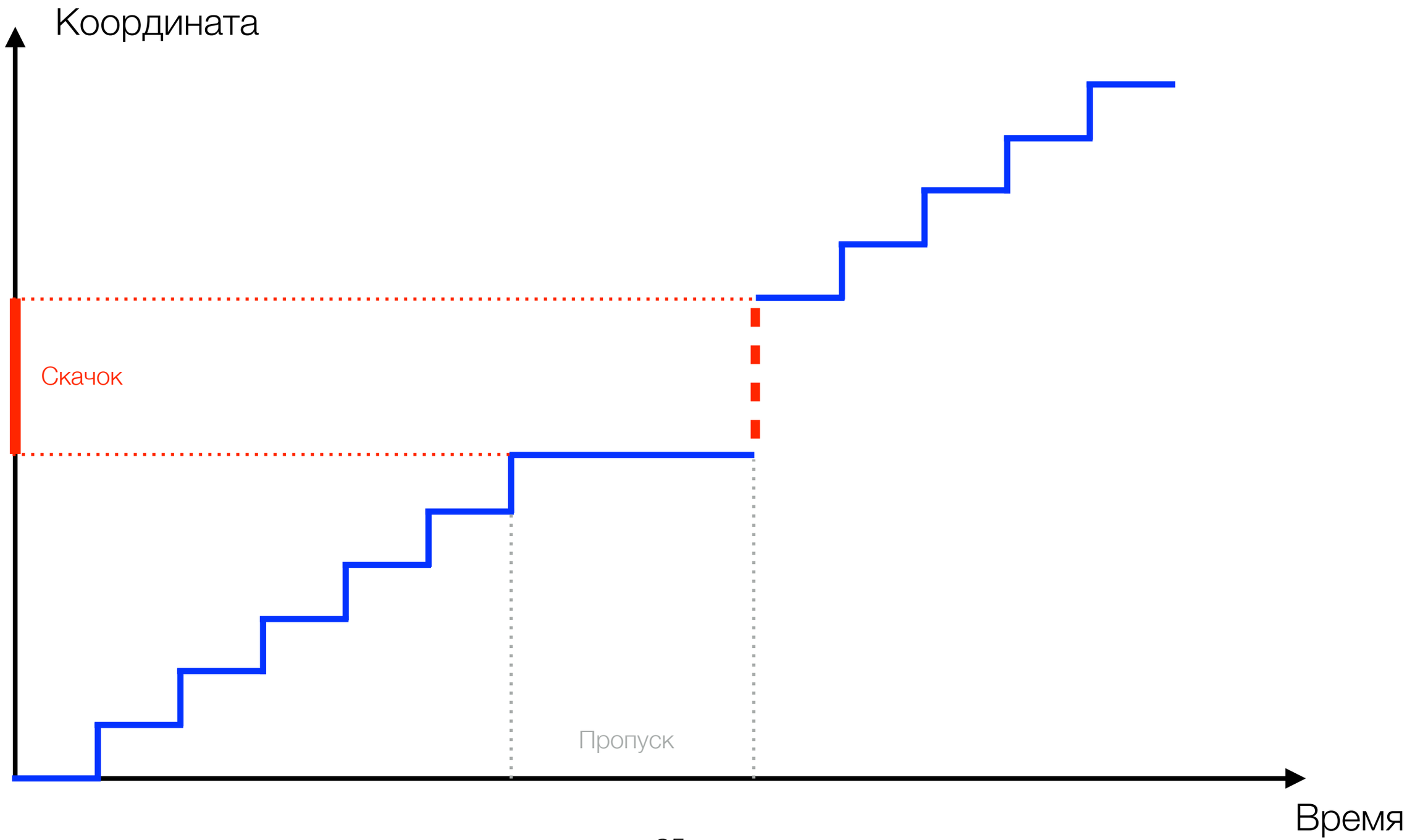


Даже 1 пропущенный фрейм может быть отчётливо виден

Цена пропущенного фрейма



Цена пропущенного фрейма



И так сойдёт!

Мы в 2017 году, у клиентов мощные процы,
тормозить не будет

Почему не сойдёт

- Куча процессов в фоне
  - Тяжёлая ОС
  - Десятки сторонних приложений

Почему не сойдёт

- Куча процессов в фоне
  - Тяжёлая ОС
  - Десятки сторонних приложений
- Энергосбережение
  - Мобильники, планшеты, ноутбуки — большинство клиентов

Почему не сойдёт

- Куча процессов в фоне
  - Тяжёлая ОС
  - Десятки сторонних приложений
- Энергосбережение
  - Мобильники, планшеты, ноутбуки — большинство клиентов
- Анимация выполняется в один поток, а он сам по себе слабый на мобильных устройствах

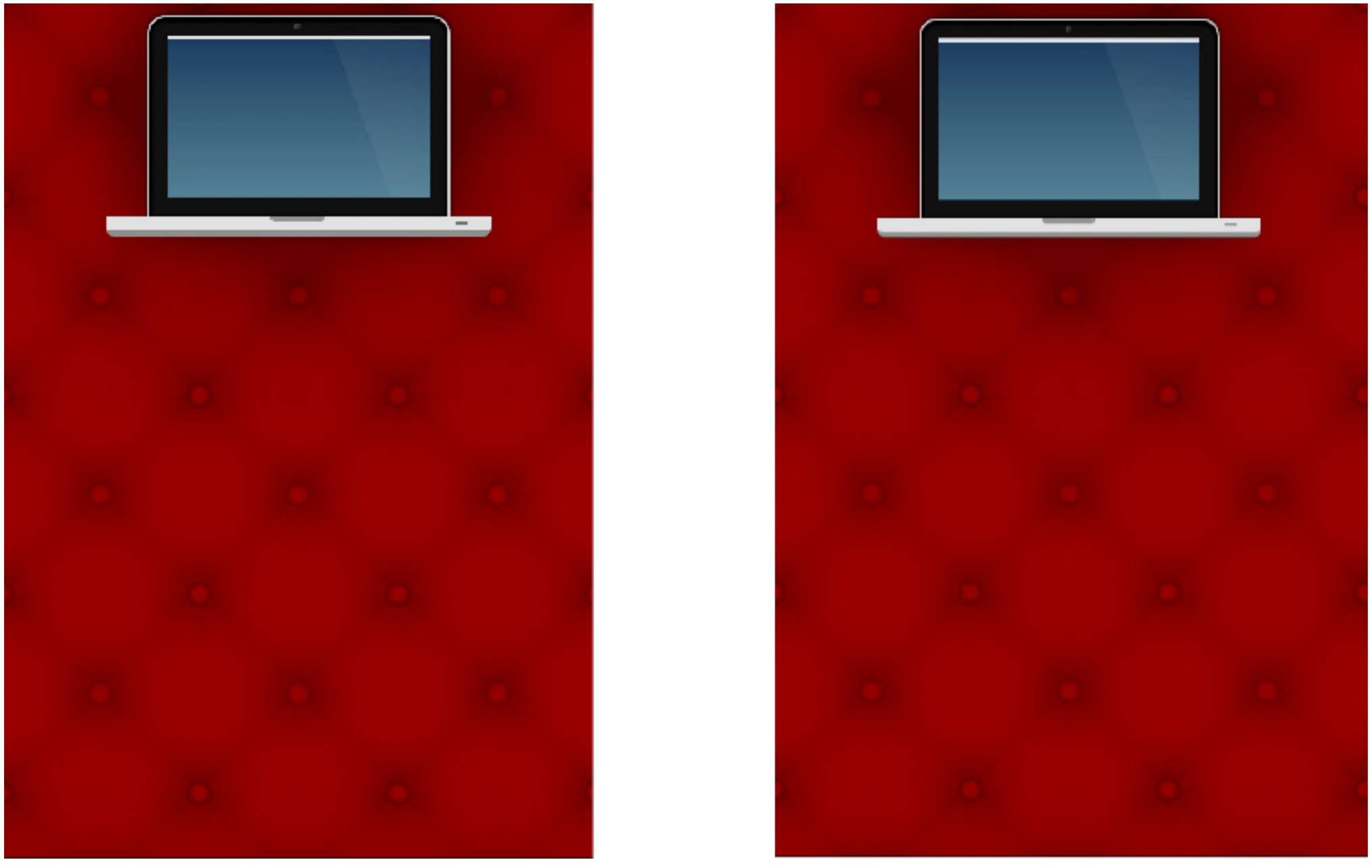
Подходы

Top/Left vs Translate

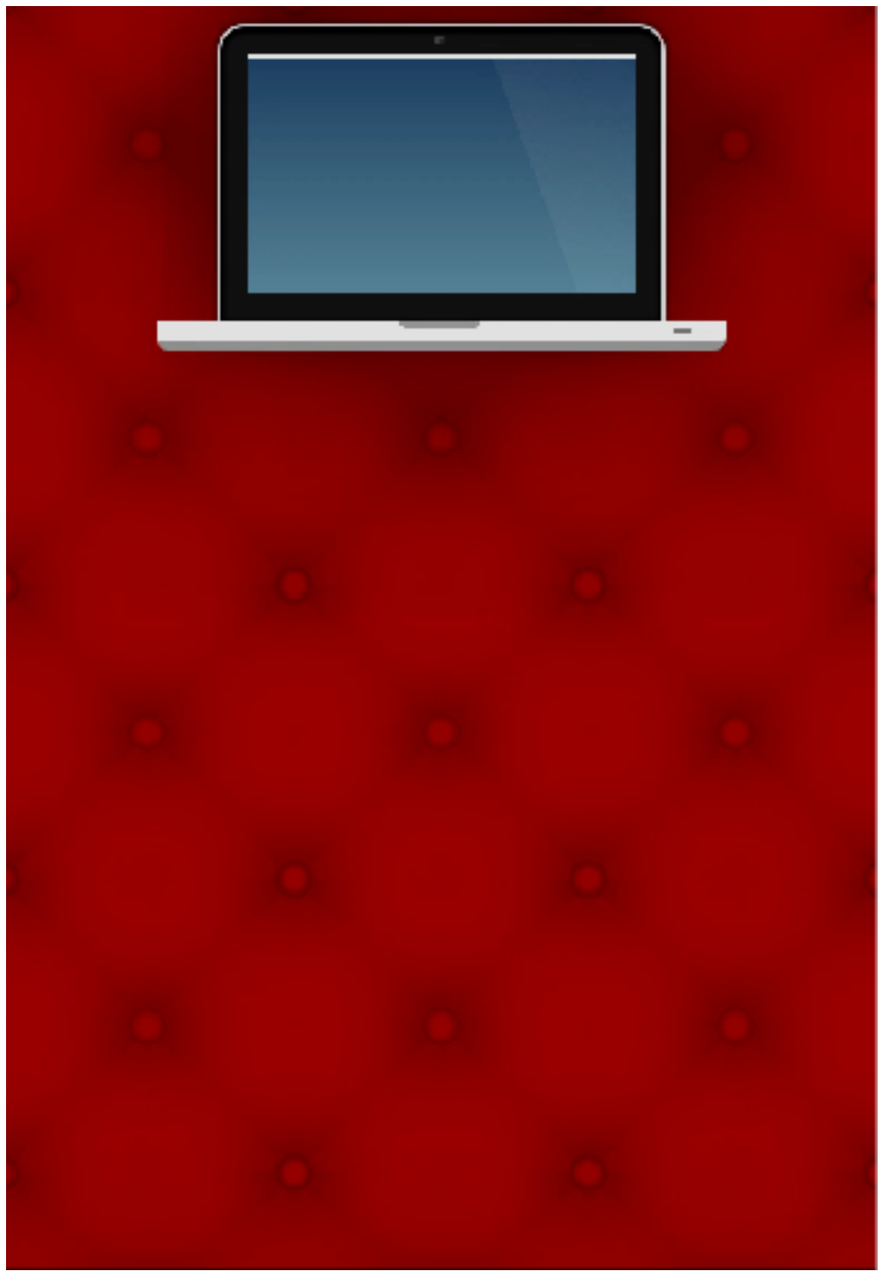
```
.start {
   position: absolute;
   top: 0;
   left: 0;
   transition: top .5s, left .
   5s;
}

.end {
   top: 100px;
   left: 100px;
}
```

```
.start {
   position: absolute;
   top: 0;
   left: 0;
   transition: transform .5s;
}

.end {
   transform: translate(100px,
   100px);
}
```

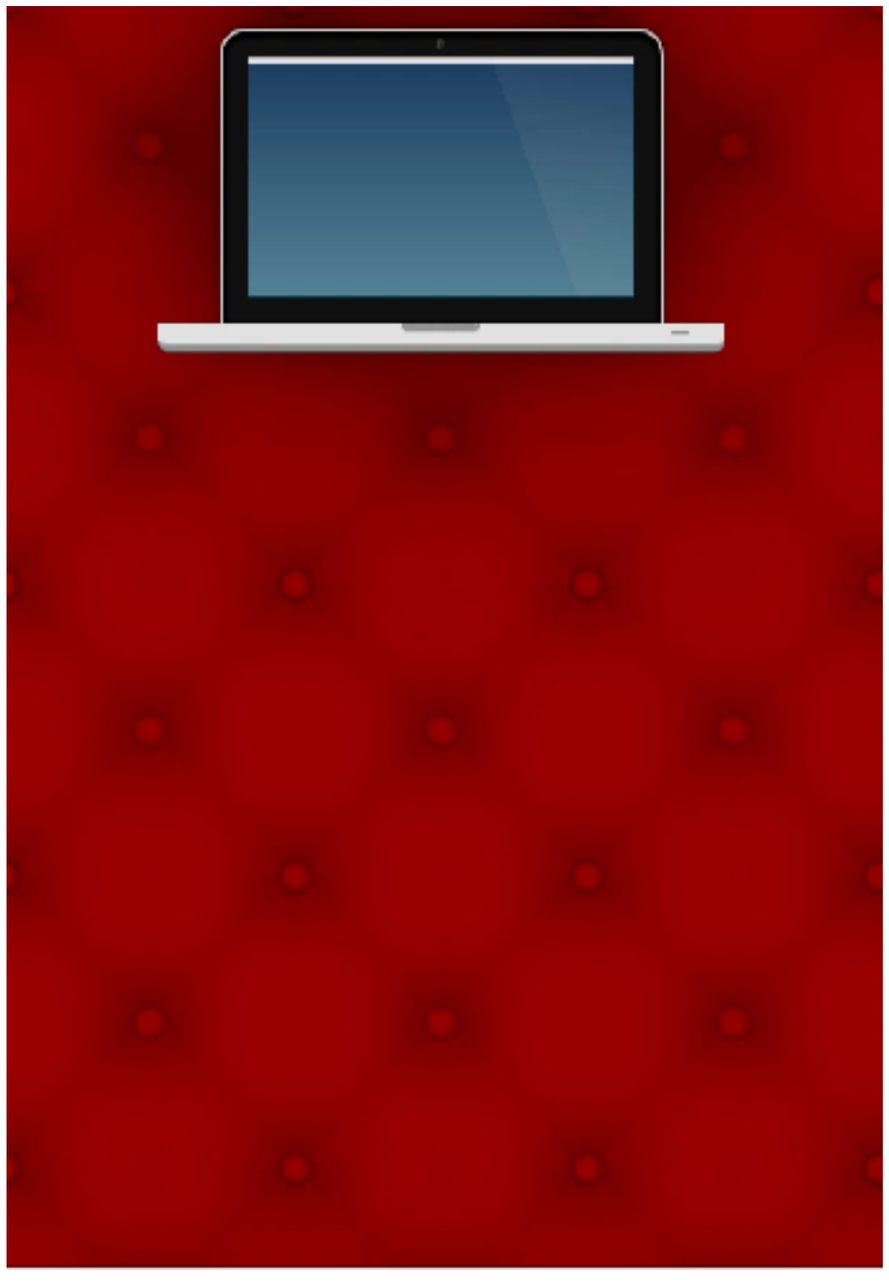
Top/Left vs Translate



Top/Left vs Translate



Top/Left



Translate

Top/Left vs Translate

Top/Left

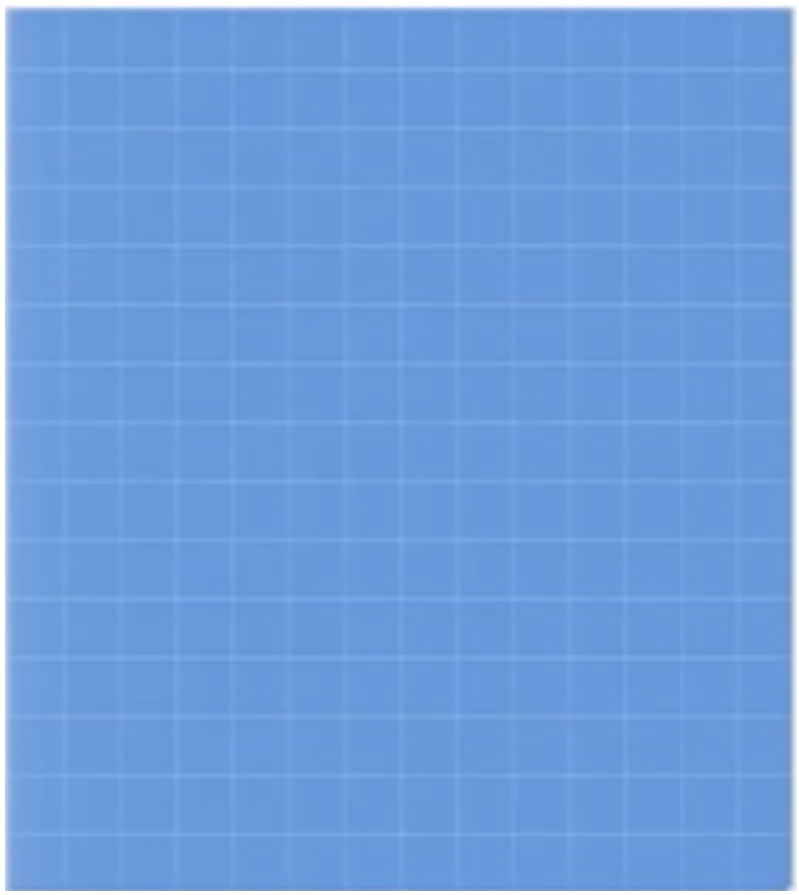
Пиксельная «лестница»

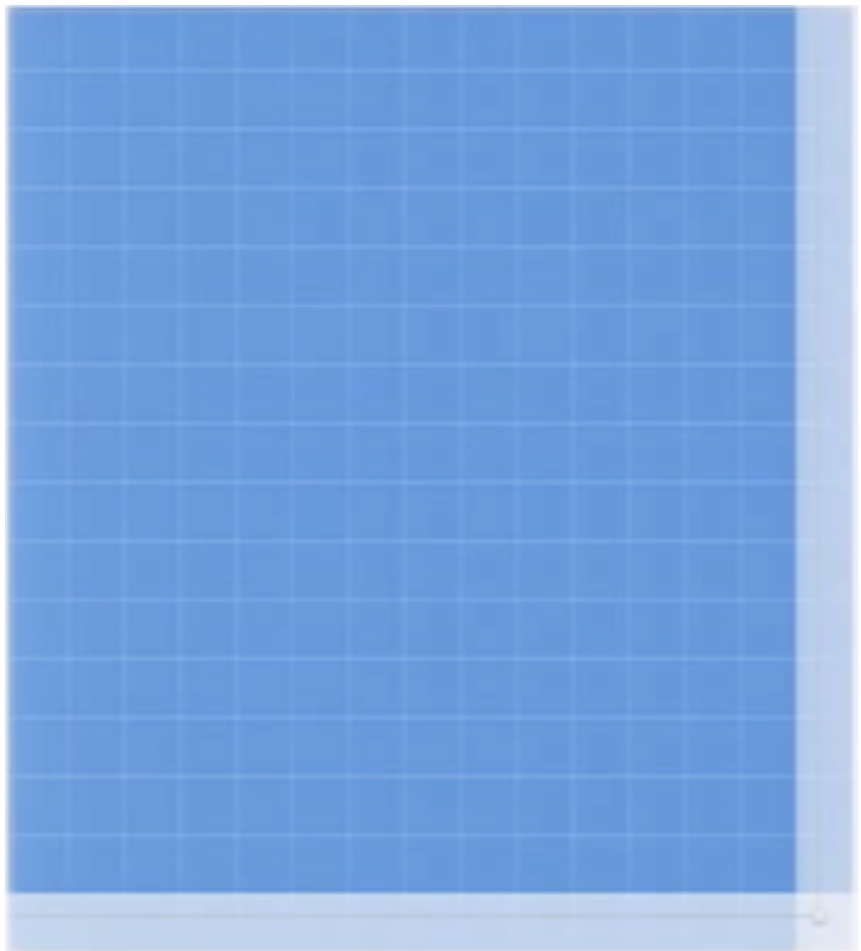
Transform

Субпиксельная интерполяция

Top/Left vs Translate

- Субпиксельная интерполяция





Top/Left vs Translate

Одновременные анимации (11 макбуков 😀)

5 FPS

60 FPS

Итак, transform:

- Composite-only
- Может задействовать GPU/compositor и не блокировать Main thread
- Субпиксельная интерполяция
- Выигрывает, если одновременно несколько анимаций

Подходы

Слои

```
will-change: transform;
```

Выносит элемент и его содержимое в отдельный слой.

- Дорогая операция
- Может происходить без вашего ведома

Если злоупотреблять, отрисовка сильно замедлится, и уйдёт много памяти.

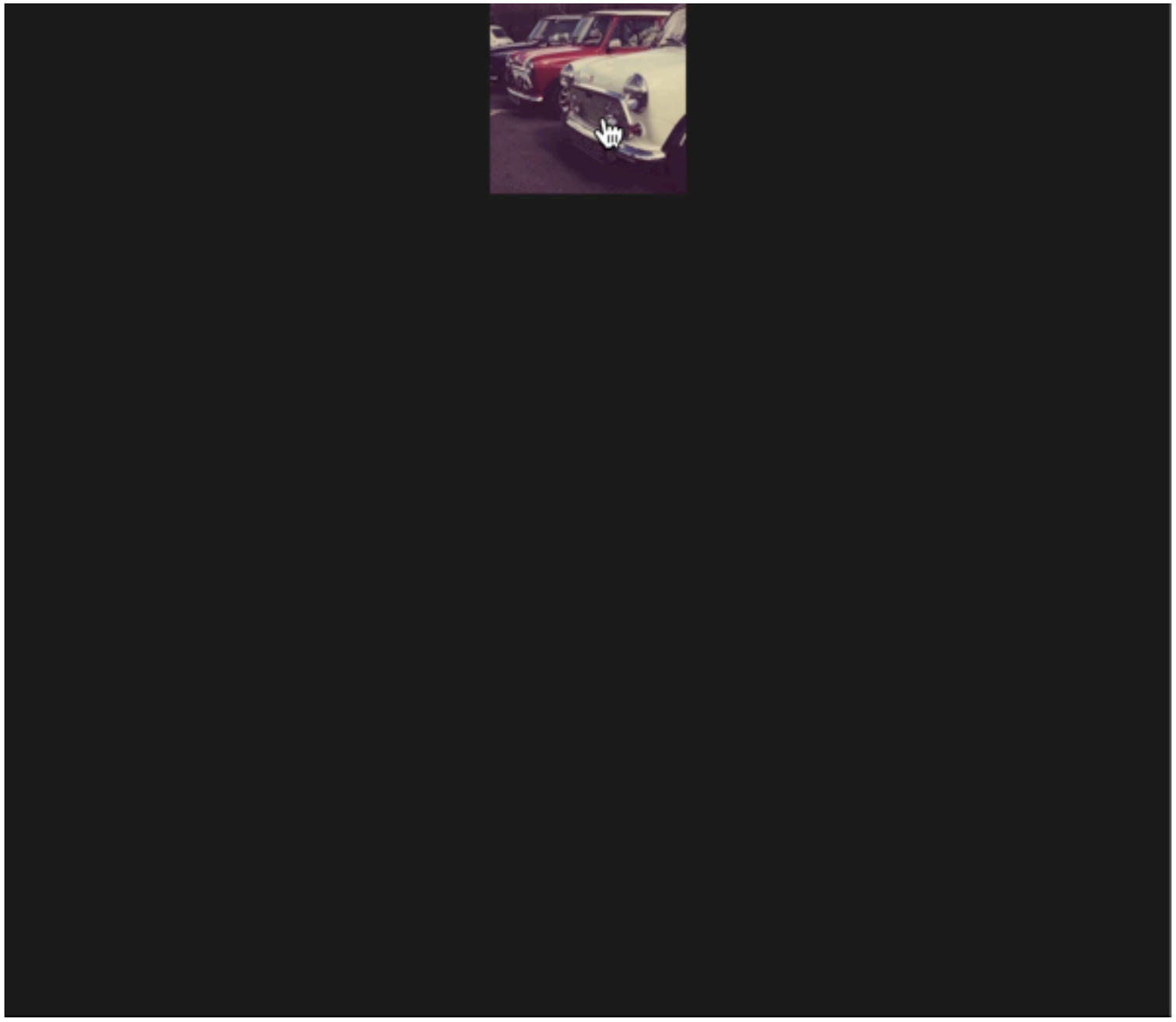
Подходы

Первые и последние 100 ms

Проседания FPS незаметны в это время

Подходящее время для вычислений, Layout/Style обновлений

А что, если анимация сложнее?



Подходы

FLIP (**F**irst, **L**ast, **I**nvert, **P**lay)

.some-css-class

Подходы

FLIP (**F**irst, **L**ast, **I**nvert, **P**lay)

```
.some-css-class
```

Подходы

FLIP (**F**irst, **L**ast, **I**nvert, **P**lay)

```
First -> Last
```

```
const before = el.getBoundingClientRect().left;

el.classList.add('some-css-class');

const after = el.getBoundingClientRect().left;

const offset = after - before;    // 100 px
```

Подходы

FLIP (**F**irst, **L**ast, **I**nvert, **P**lay)


```
.some-css-class
```

```
1. offset = 100 px
```

Подходы

FLIP (**F**irst, **L**ast, **I**nvert, **P**lay)

Invert

```
el.style.transition='none';
el.offsetHeight; // hack — force DOM update
el.style.transform=`translateX(-${offset}px)`;
el.style.transition=null;
```

Подходы

FLIP (**F**irst, **L**ast, **I**nvert, **P**lay)

.some-css-class

transform: translateX(-100px)

.some-css-class

1. offset = 100 px
2. transform: translateX(-100px)

Подходы

FLIP (**F**irst, **L**ast, **I**nvert, **P**lay)

Play

```
el.style.transform=null;
```

Подходы

FLIP (**F**irst, **L**ast, **I**nvert, **P**lay)

.some-css-class

transform:
translateX(-100px)

.some-css-class

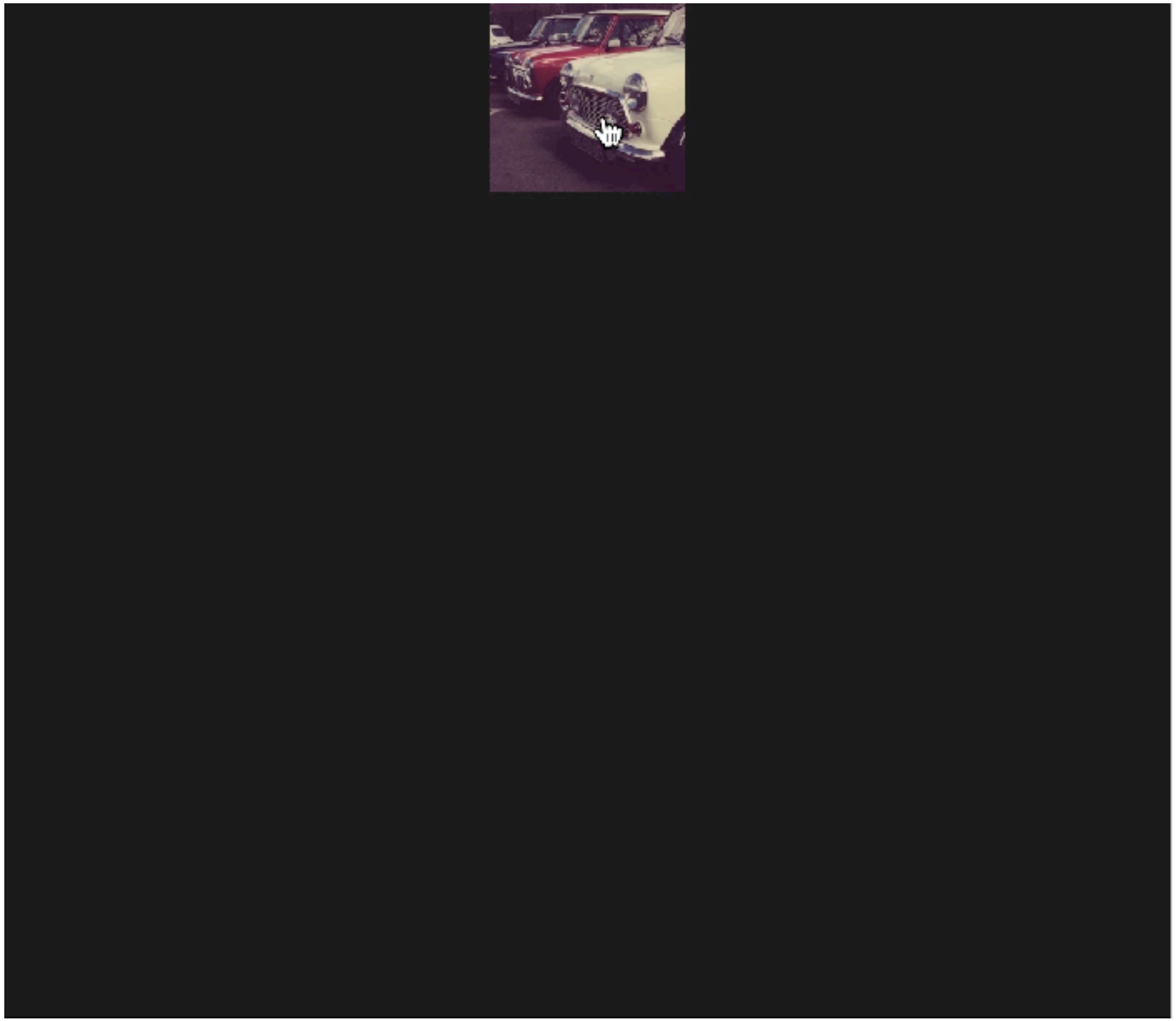
transform:
null

1. offset = 100 px
2. transform: translateX(-100px)
3. transform: null

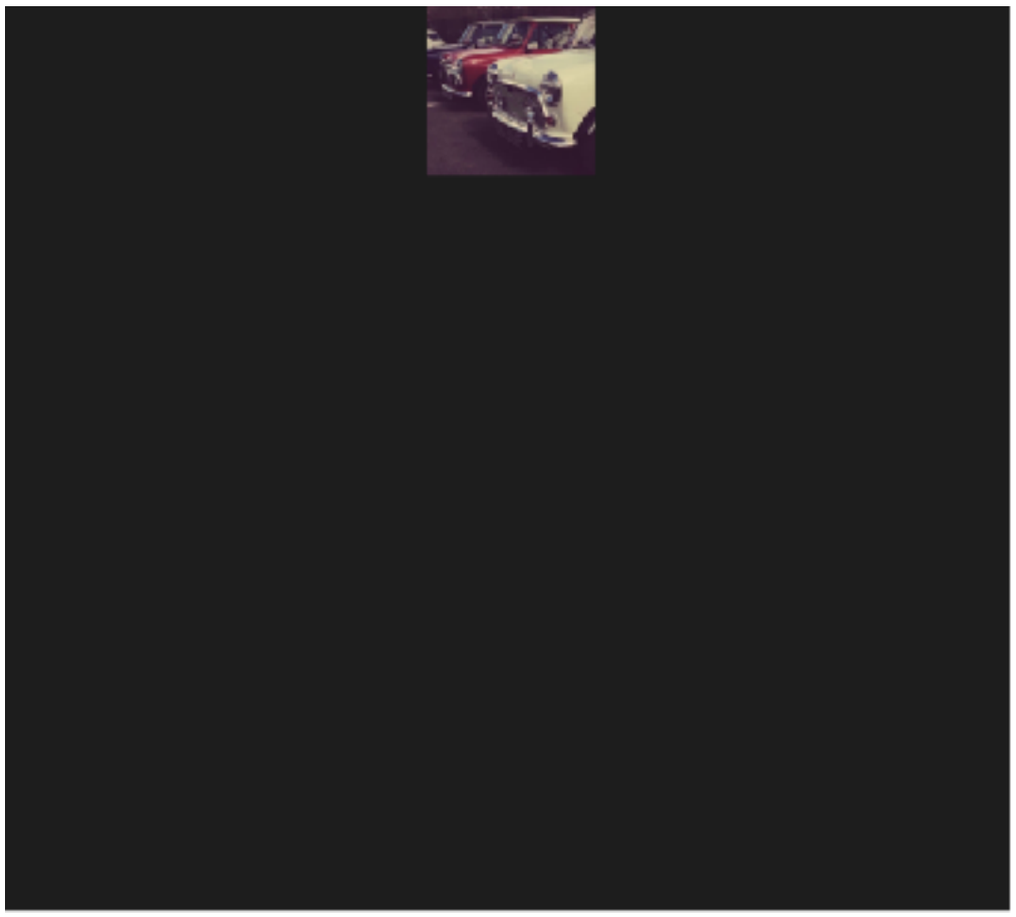
Подходы

FLIP (**F**irst, **L**ast, **I**nvert, **P**lay)

```
.some-css-class
```

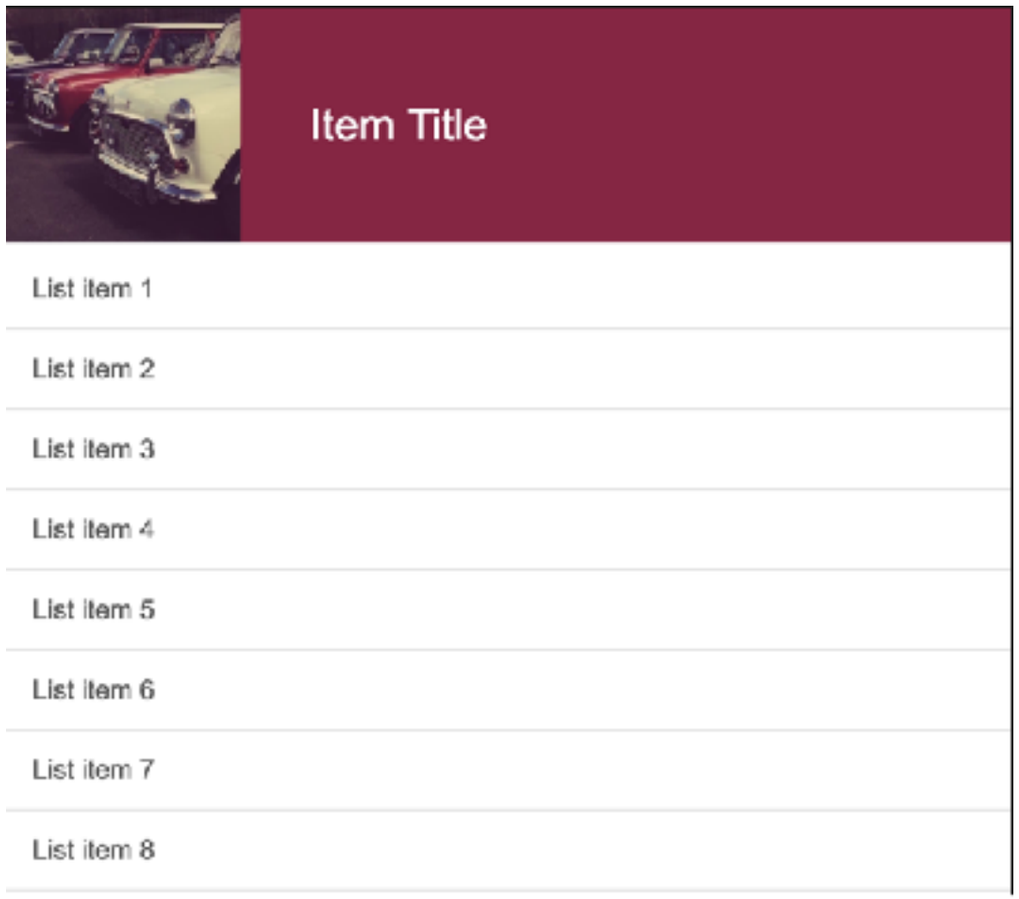


First -> Last



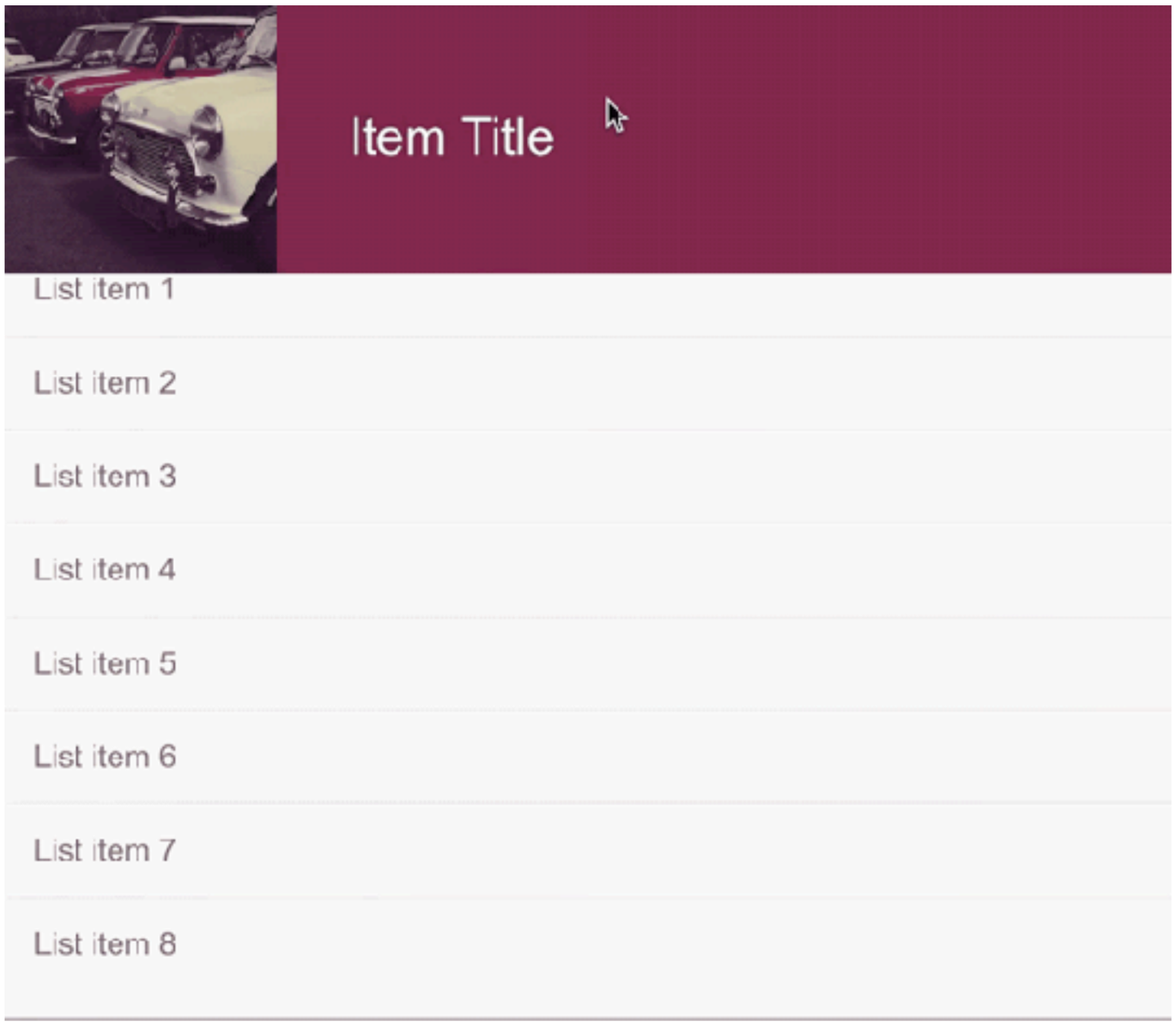
.card

>



.card .card--expanded

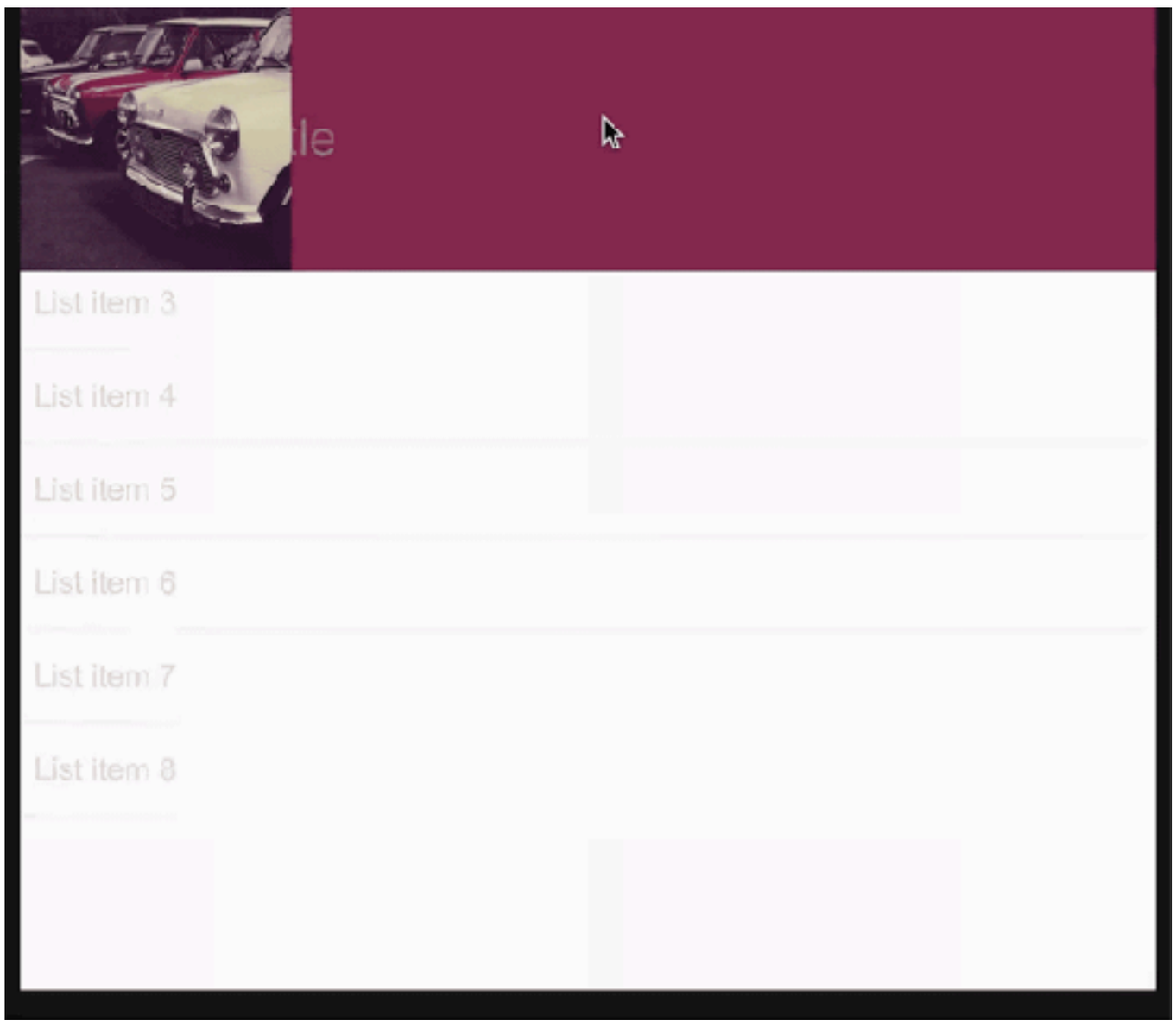
Invert




```
title:

translateX влево

opacity 0
```

```
list:

translateY вверх

opacity 0
```

Invert



card:

transform-origin

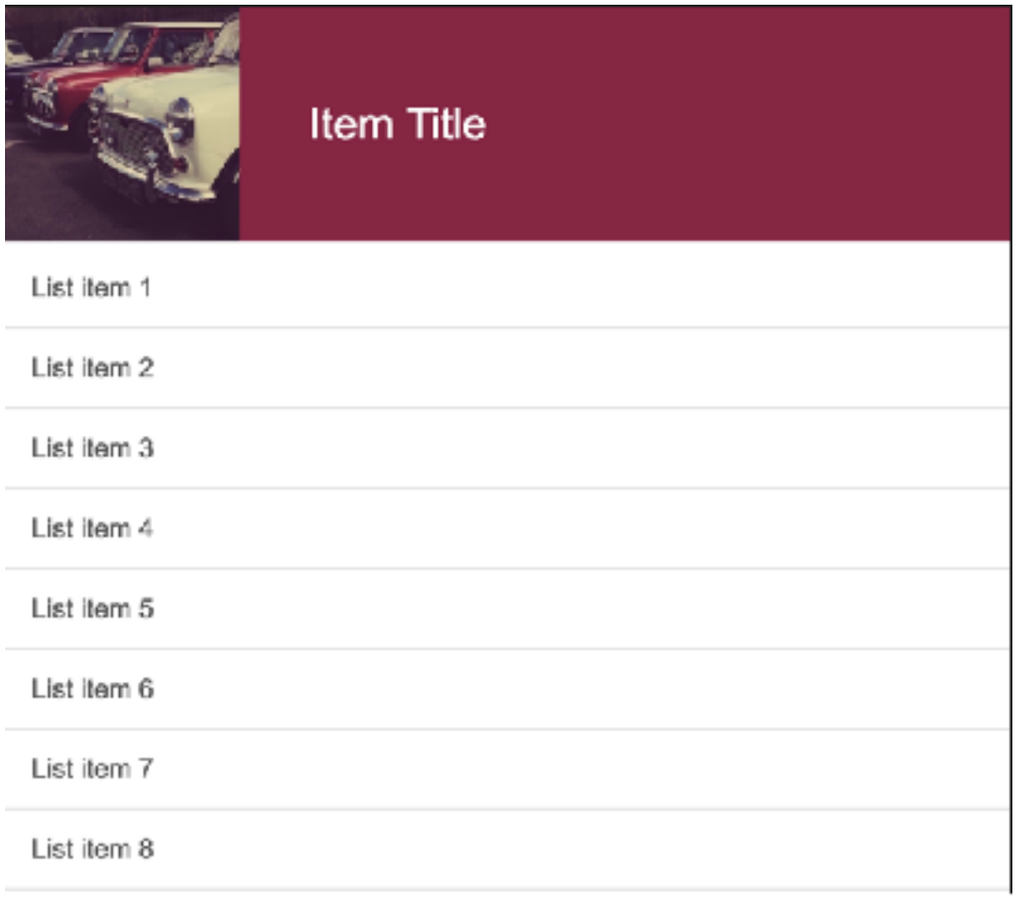
scale уменьшить

photo:

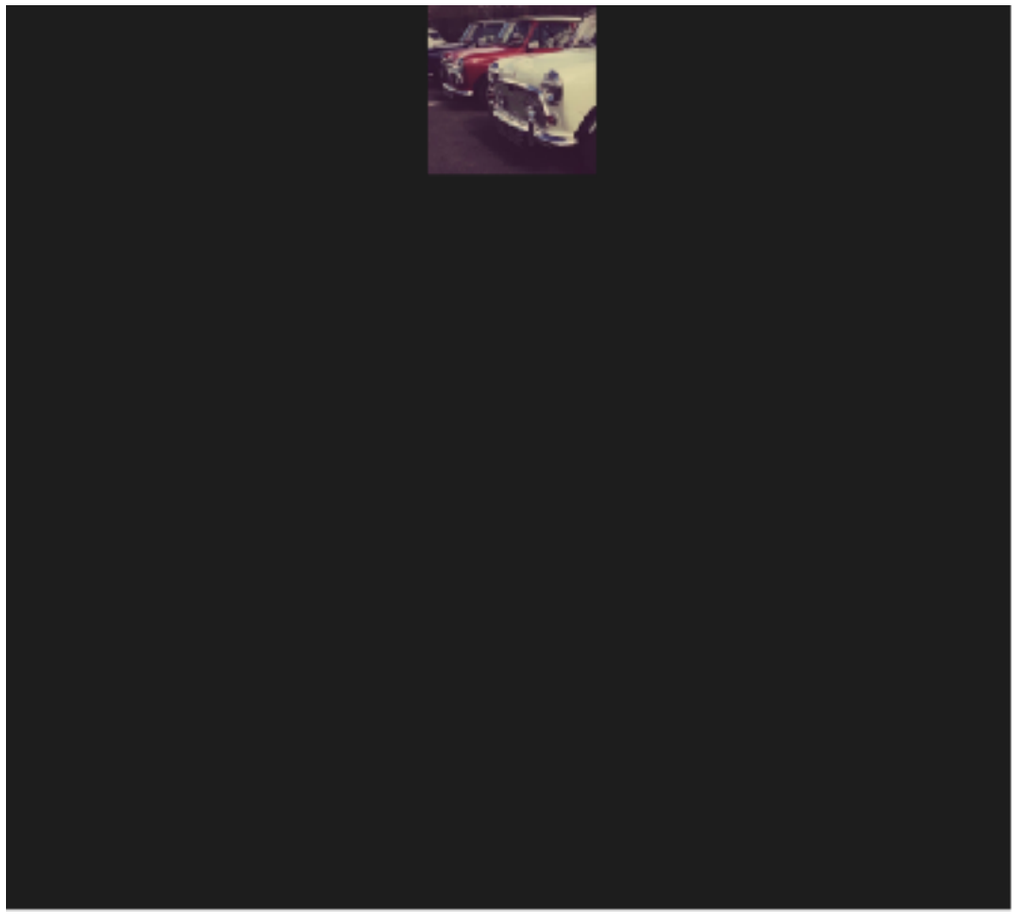
translateX вправо

scale уменьшить

Это происходит мгновенно



>

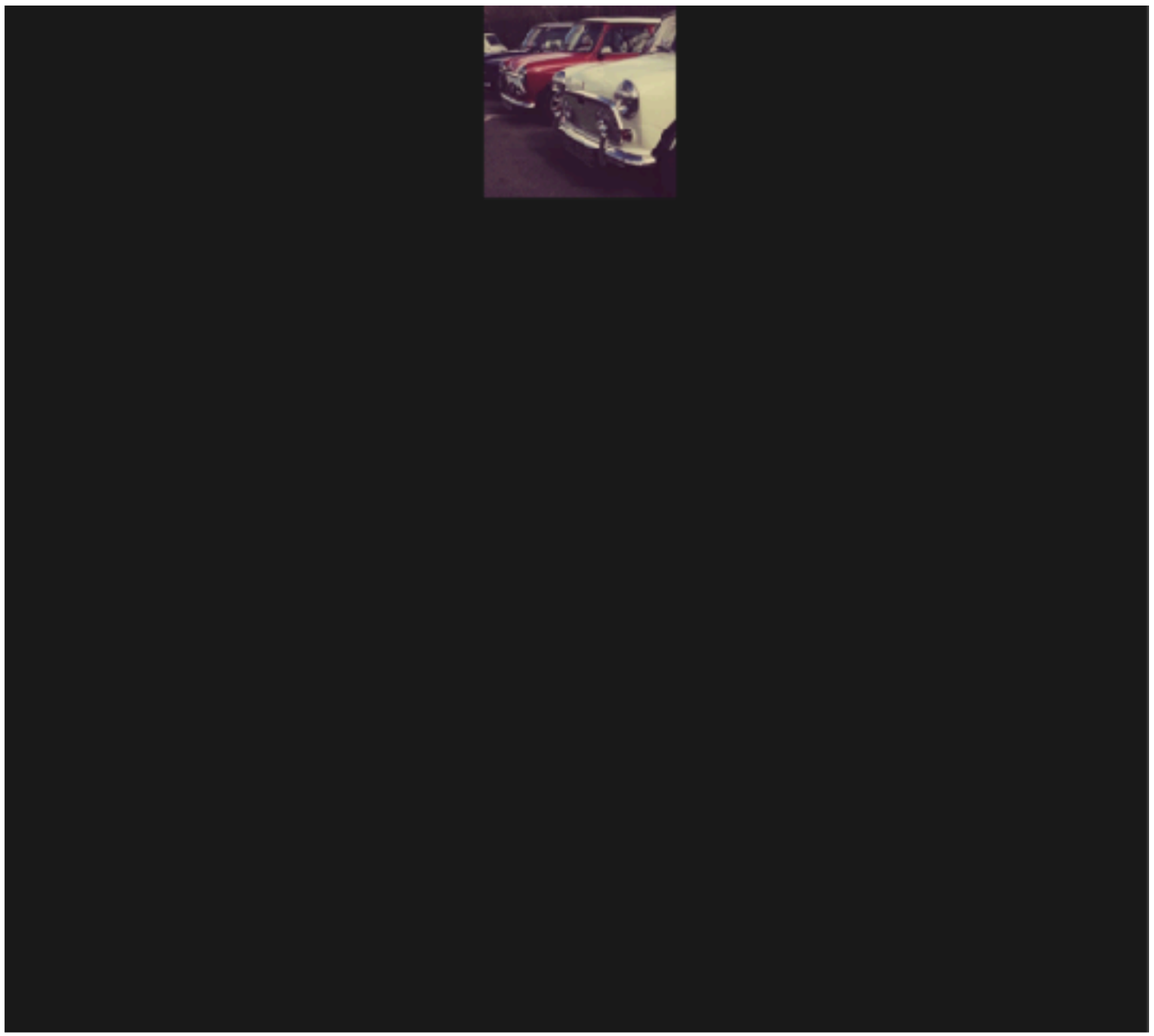


`.card .card--expanded`

`.card .card--expanded`

`+ transform, opacity`

Invert



`.card .card--expanded`

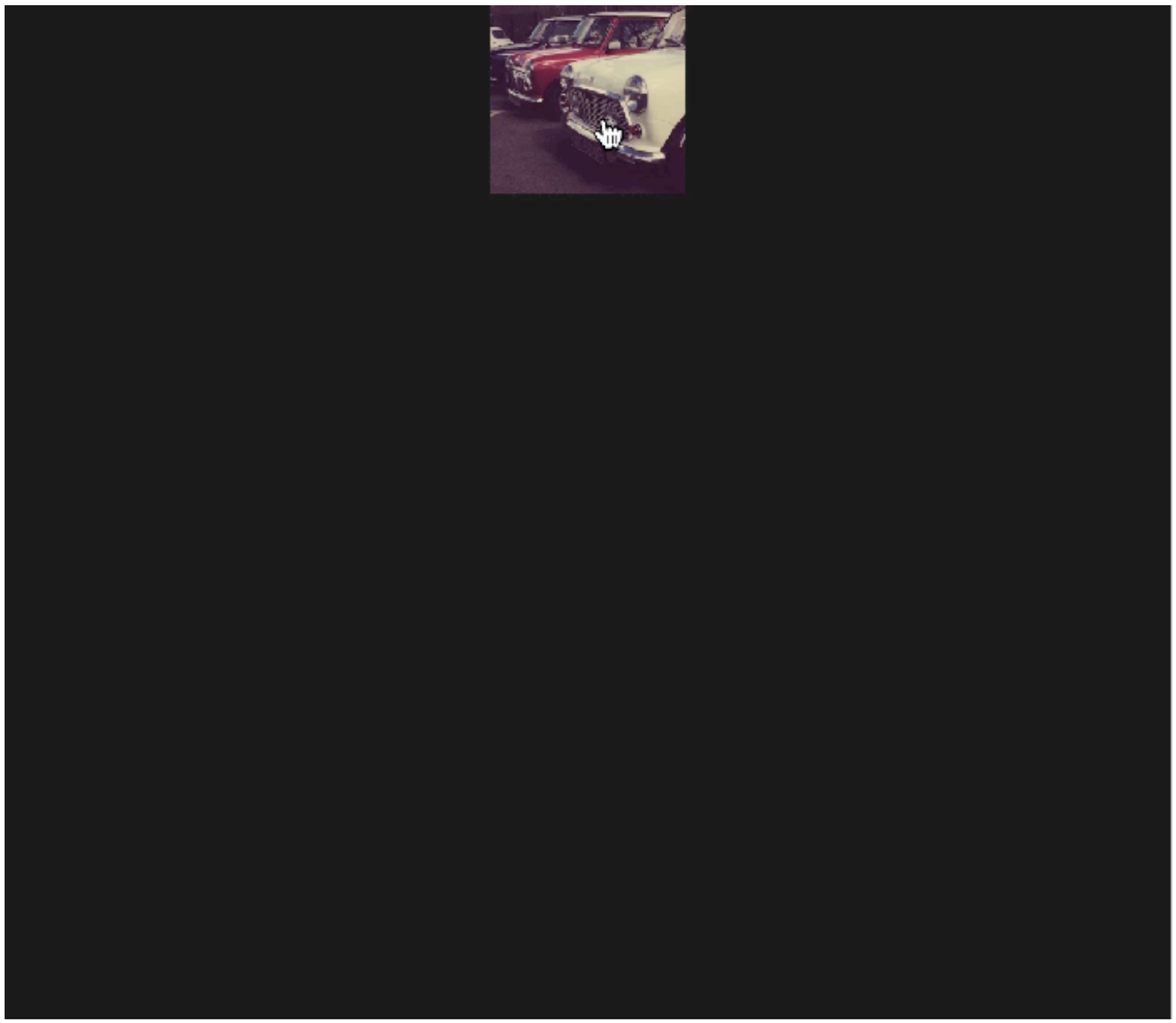
`card, title, photo,`

`list — на них`

`translate, scale,`

`opacity`

Play



`.card .card--expanded`

`.card--animated`

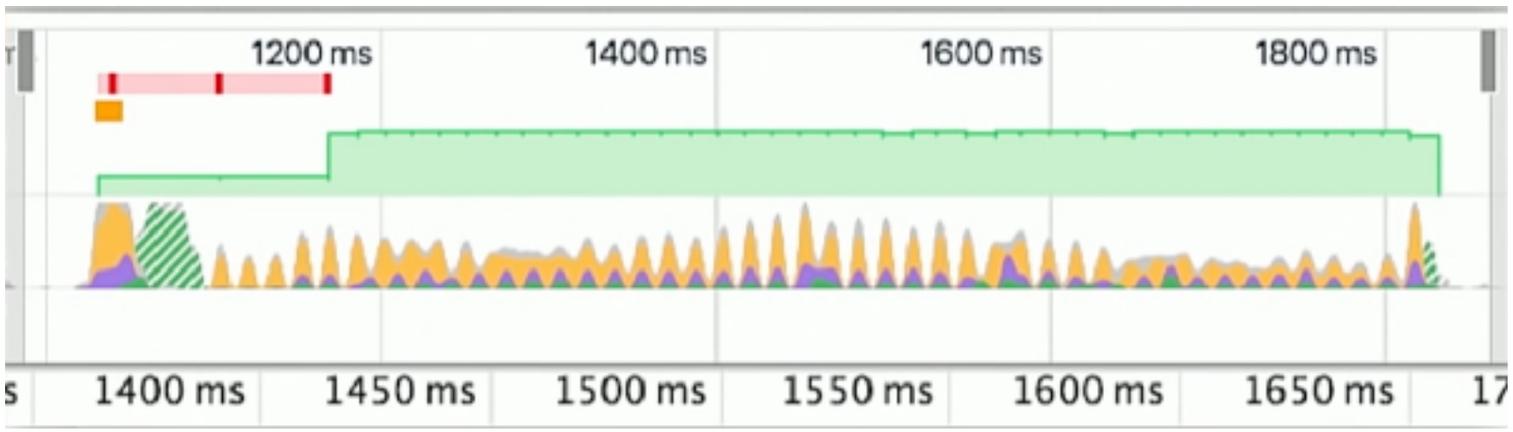
1. Добавляем transitions

(.card--animated)

2. Убираем все трансформы

Подходы

Первые и последние 100 ms



First

Last

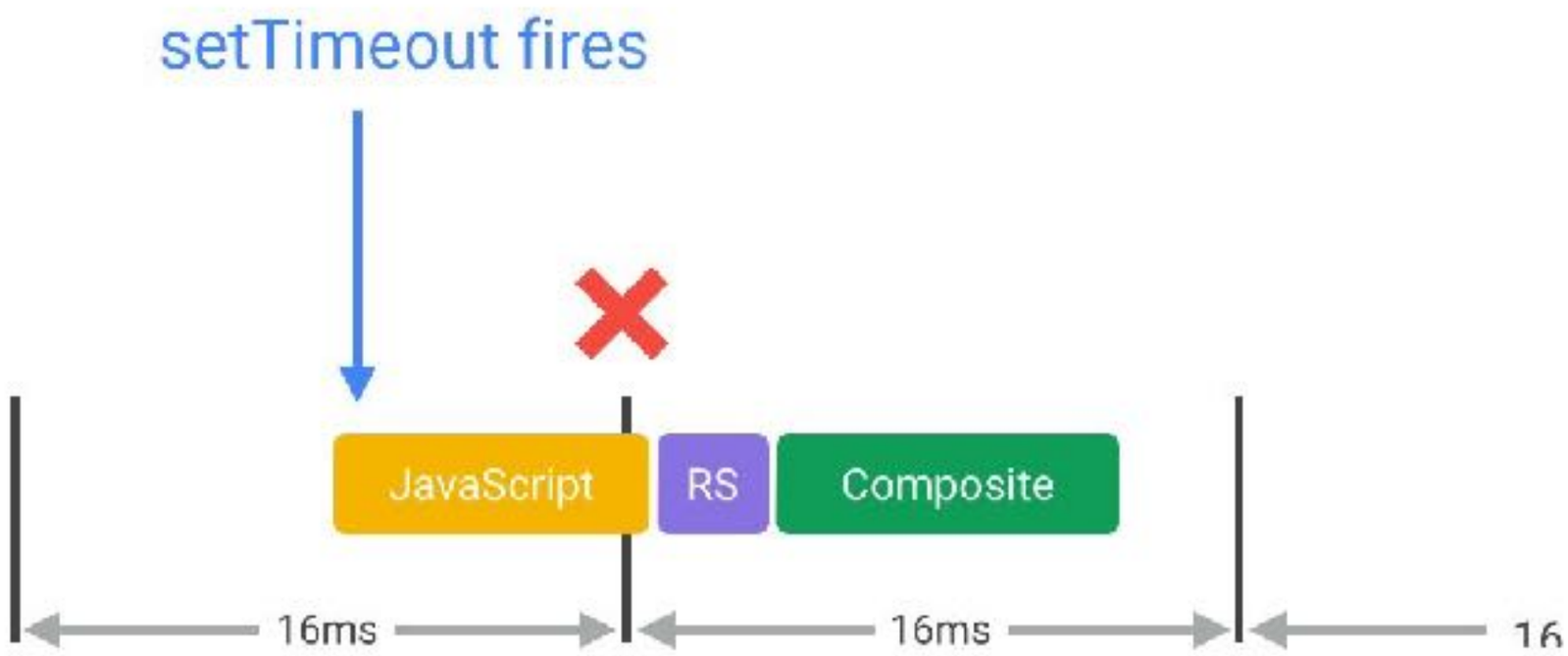
Invert

Play

Подходы

JS-анимация

У браузера есть собственная частота обновления.
Если не попасть в неё — фрейм будет пропущен



requestAnimationFrame

```
const updateScreen = time => { // do stuff };

requestAnimationFrame(updateScreen);
```

- Callback к requestAnimationFrame будет выполнен в начале фрейма
- В неактивной вкладке выполнение приостанавливается

requestAnimationFrame

Smooth scroll пример

```
const scrollALittle = time => {
    // делаем шаг скролла (с учётом ease-функции времени)
    window.scroll(0, ease(time, position));

    // если ещё не пора заканчивать, вызываем ещё
    if (time - timeStart < duration)
        requestAnimationFrame(scrollALittle);
};

// вызываем в первый раз
requestAnimationFrame(scrollALittle);
```

requestAnimationFrame

CPU time

Хорошо бы успевать делать вычисления хотя бы за **10 мс.** Запас времени.

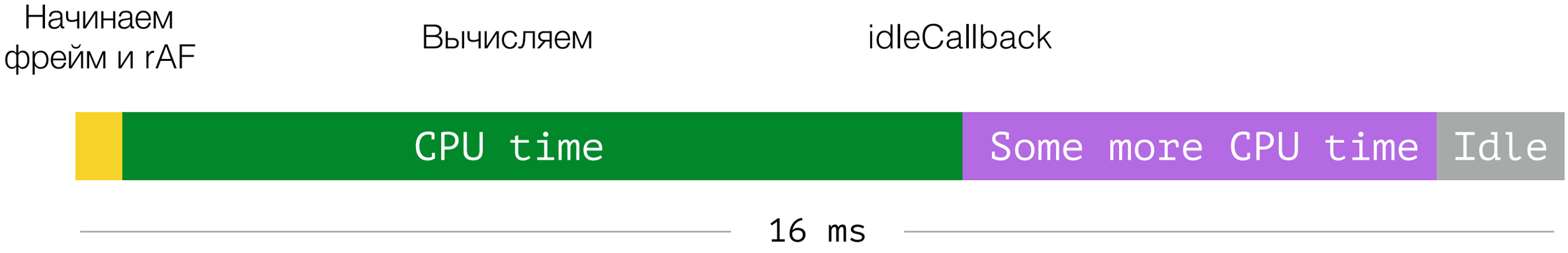
requestIdleCallback

Обратная ситуация:

Здесь можно делать что-нибудь ещё

requestIdleCallback

Обратная ситуация:



- Отправка данных для аналитики
- Манипуляции с DOM (вне текущей анимации)
- Другие несрочные операции

requestIdleCallback

```
requestIdleCallback(myNonEssentialWork);

function myNonEssentialWork (deadline) {
    while(deadline.timeRemaining() && tasks.length)
        doStuff(tasks);

    if (tasks.length > 0)
        requestIdleCallback(myNonEssentialWork);
}
// если обязательно нужно вызвать в течение 2 сек
requestIdleCallback(myNonEssentialWork, {timeout: 2000});
```

Using requestIdleCallback, Paul Lewis — http://bit.ly/2rLPpiC

Web Animations API

```
const player = HTMLElement.animate( /* ... */ )
console.log(player.playState); //"running"
```

Web Animations API

```
const player = HTMLElement.animate( /* ... */ )
console.log(player.playState); //"running"

player.pause(); //"paused"
player.play();  //"running"
player.cancel(); //"idle" – в начальное состояние
player.finish(); //"finished" – в конечное состояние

player.reverse(); проиграть анимацию задом наперёд
```

Web Animations API

```
const player = HTMLElement.animate( /* ... */ )
console.log(player.playState); //"running"

player.pause(); //"paused"
player.play();  //"running"
player.cancel(); //"idle" – в начальное состояние
player.finish(); //"finished" – в конечное состояние

player.reverse(); проиграть анимацию задом наперёд

console.log(player.playbackRate); //1
player.playbackRate = 2; // можно динамически менять скорость
```

Web Animations API

```
const player = HTMLElement.animate( /* ... */ )
console.log(player.playState); //"running"

player.pause(); //"paused"
player.play();  //"running"
player.cancel(); //"idle" – в начальное состояние
player.finish(); //"finished" – в конечное состояние

player.reverse(); проиграть анимацию задом наперёд

console.log(player.playbackRate); //1
player.playbackRate = 2; // можно динамически менять скорость

player.onfinish = function() {
  console.log(player.currentTime); // 3500 ms
};
```

Web Animations API

Поддержка:

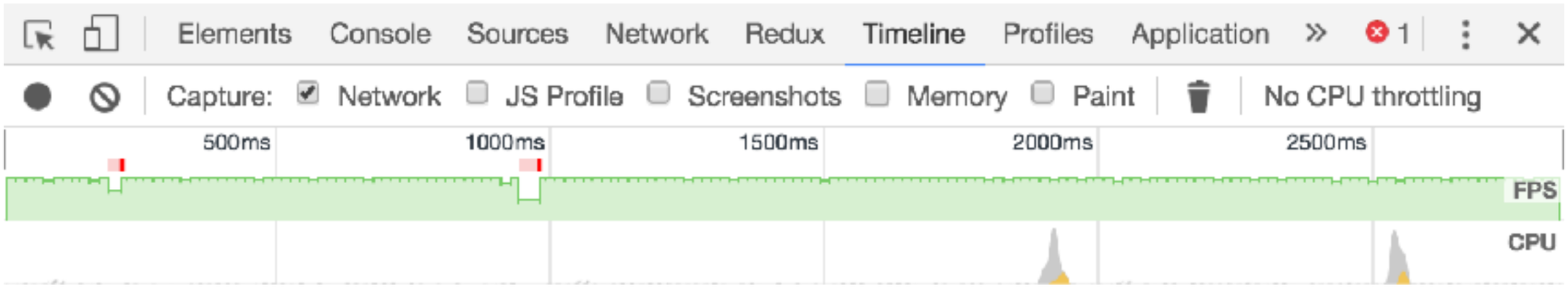
Polyfill — bit.ly/IWukam

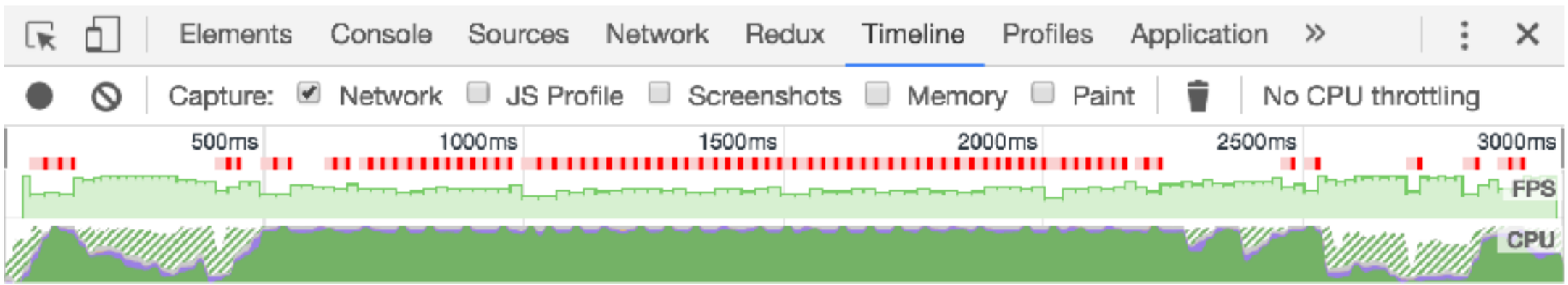
Web Animations API — Dan Wilson — bit.ly/2rxSr95

Инструменты

Инструменты

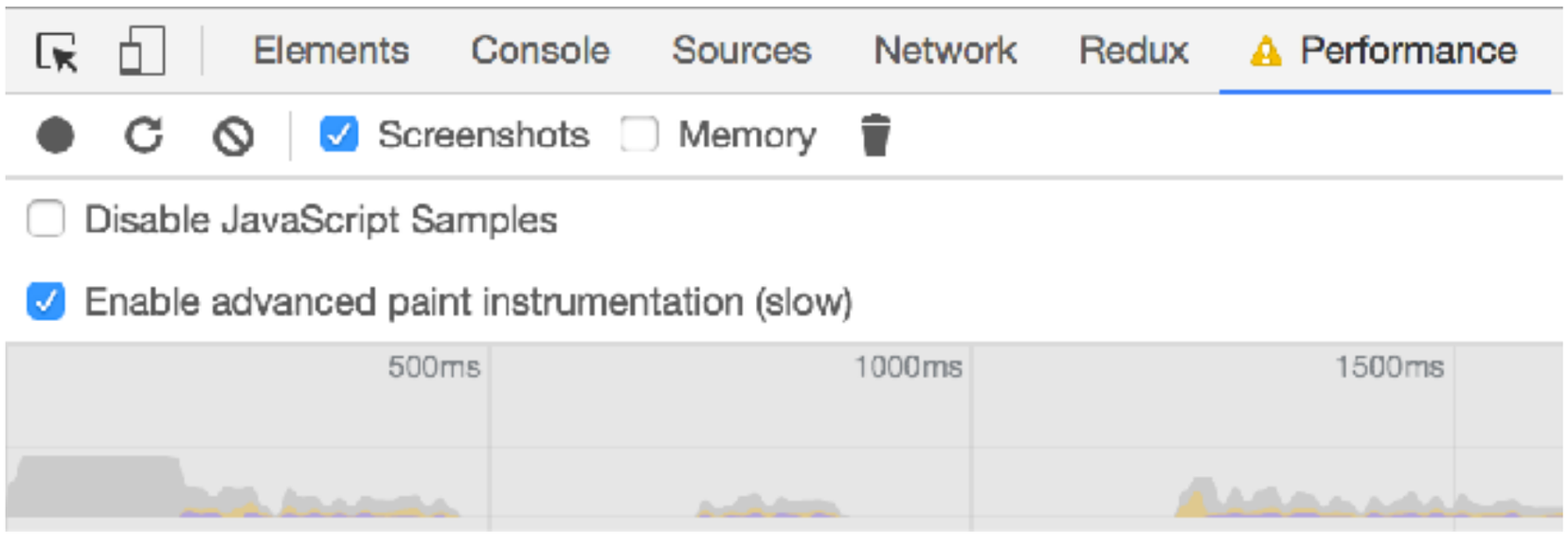
Performance





Инструменты

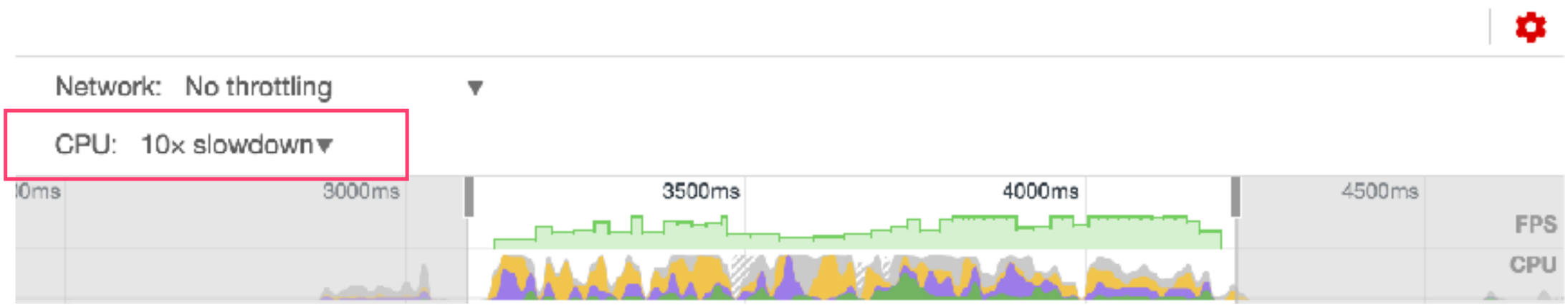
Performance ощутимо замедляют:



- Screenshots
- Enable advanced paint instrumentation

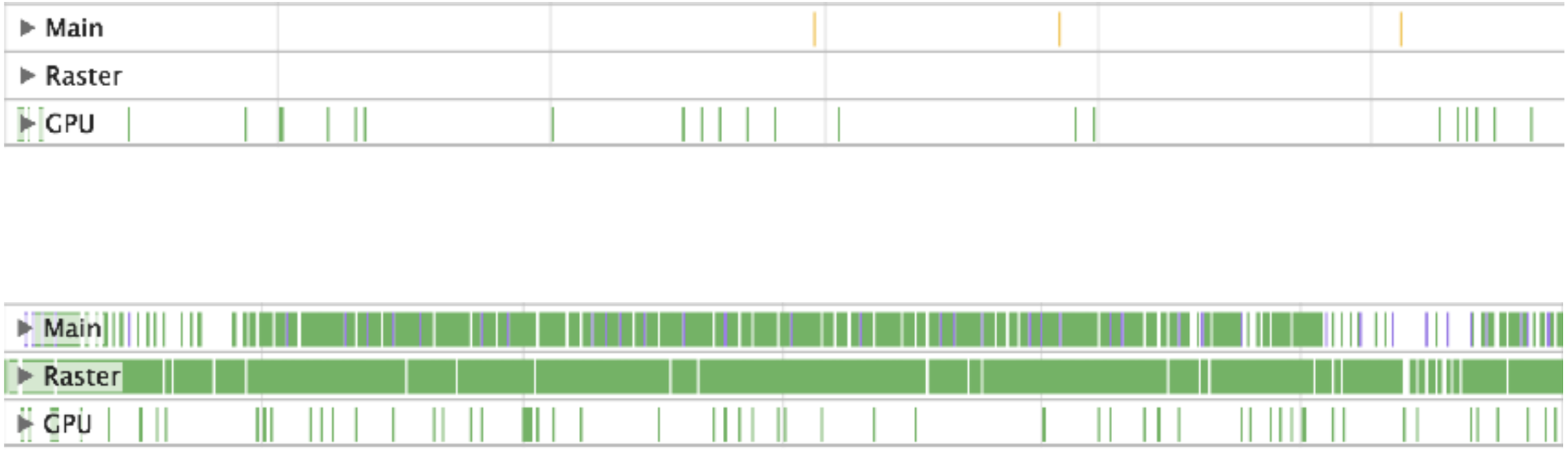
Инструменты

Performance CPU throttling



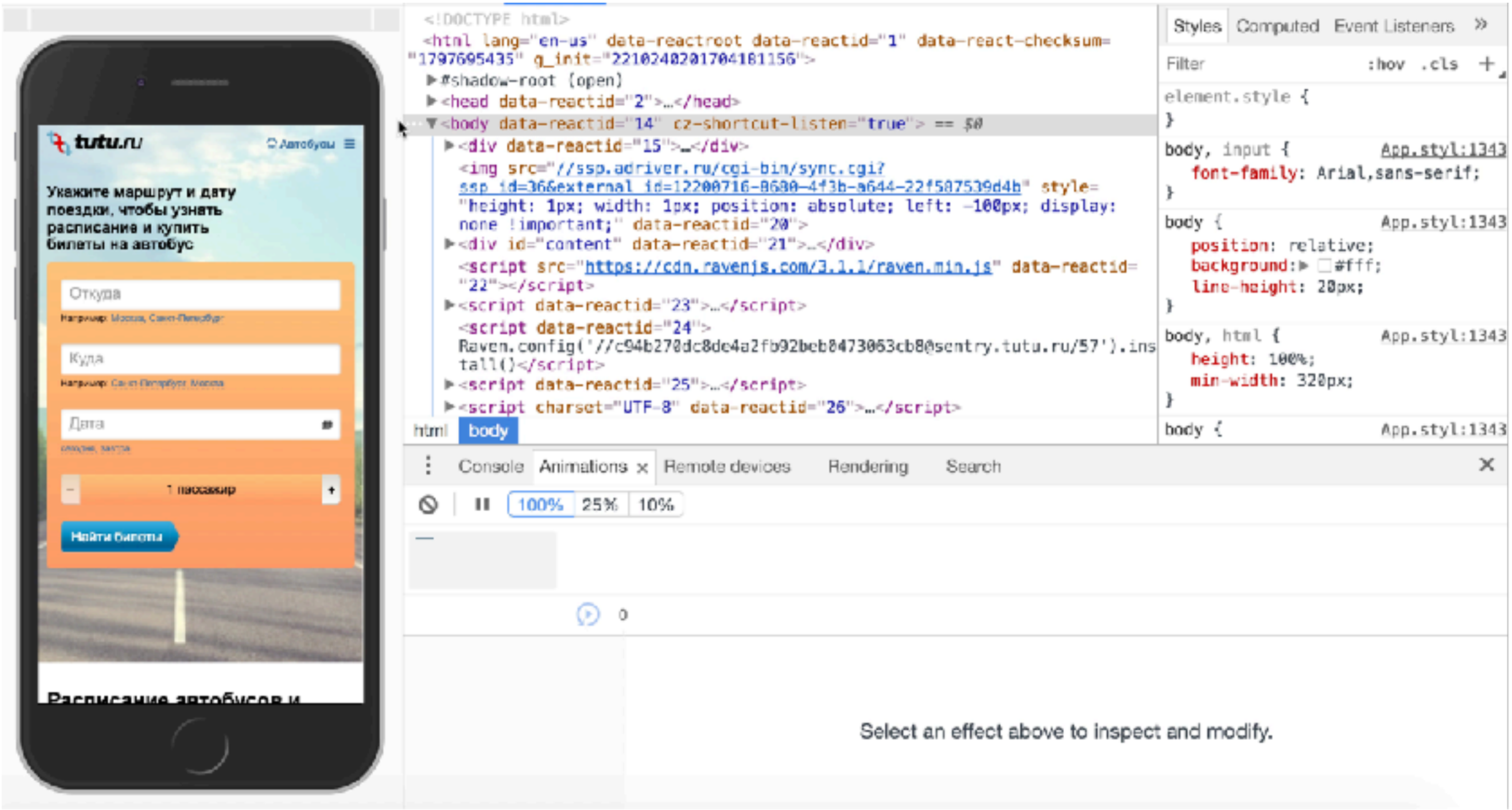
Инструменты

Загруженность CPU/GPU



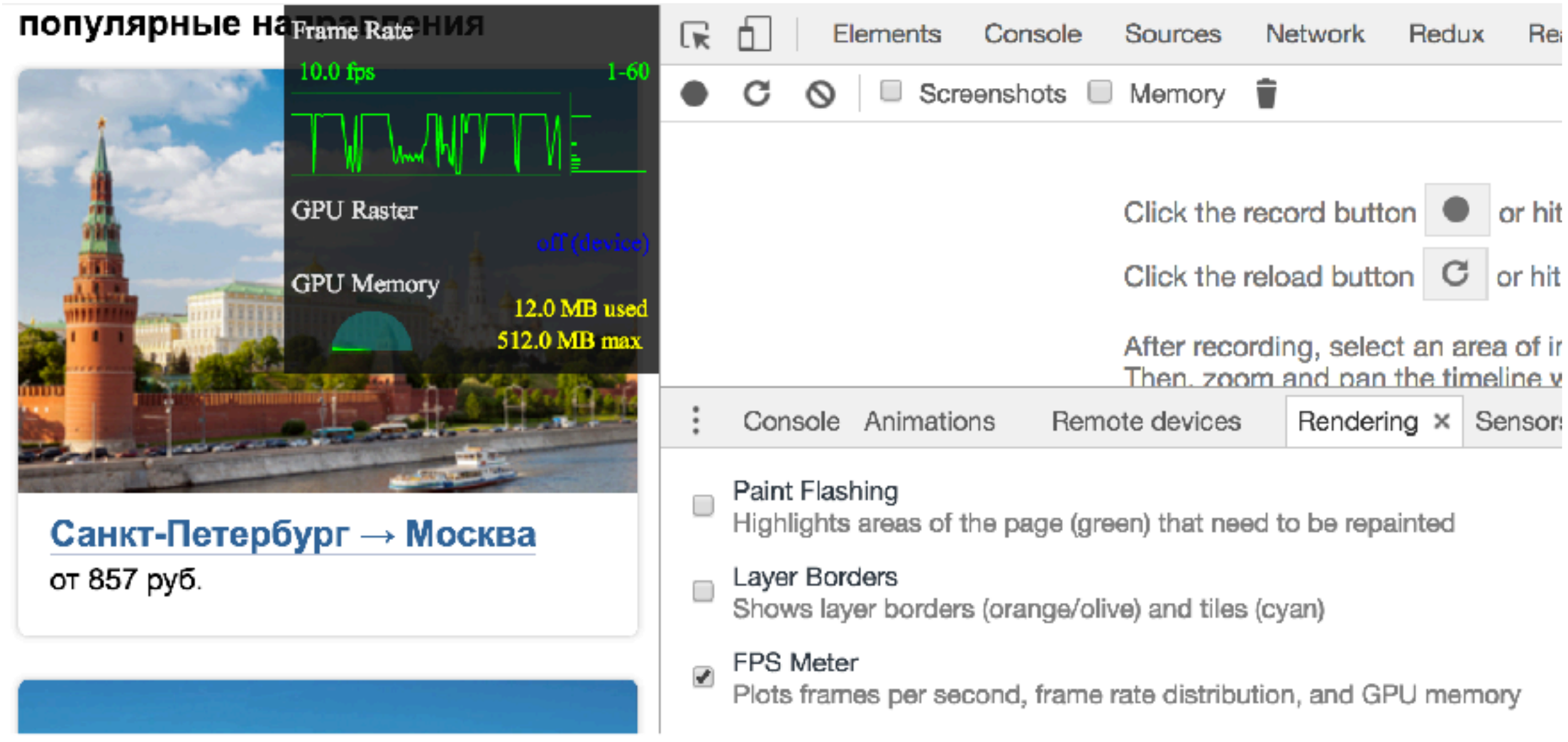
Инструменты

Анимация



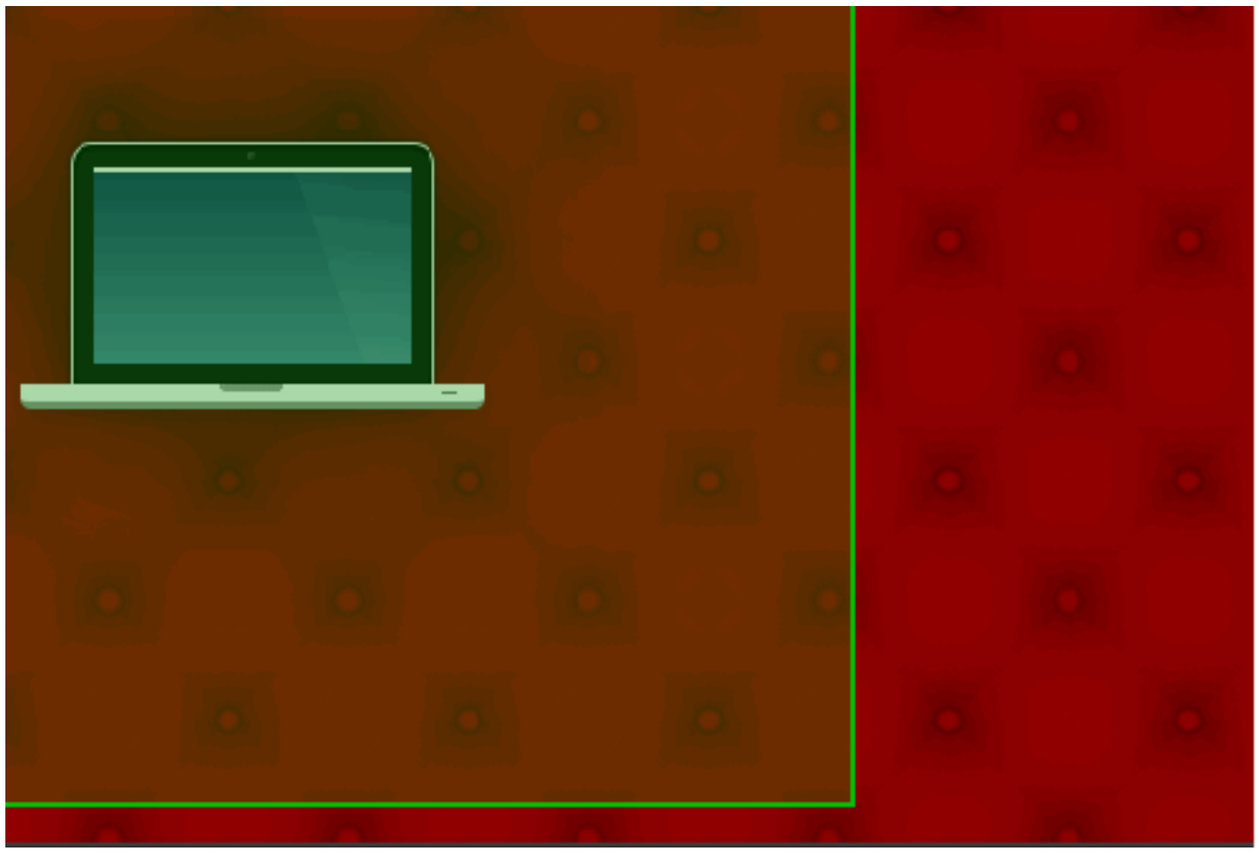
Инструменты

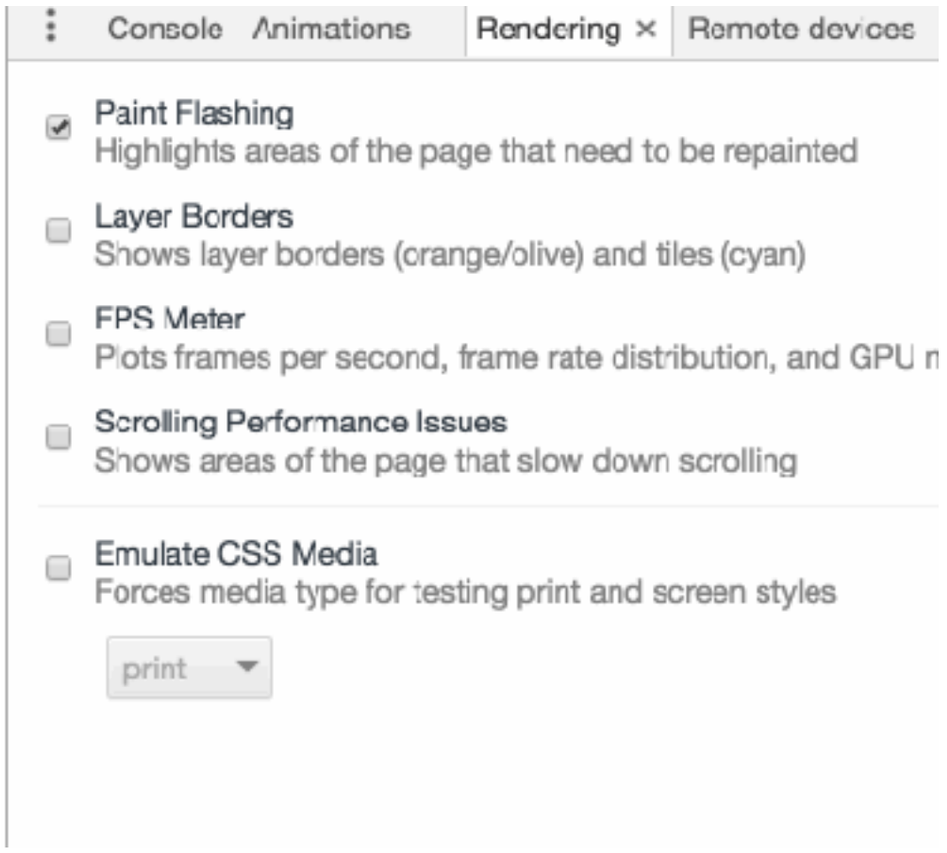
Счётчик FPS



Инструменты

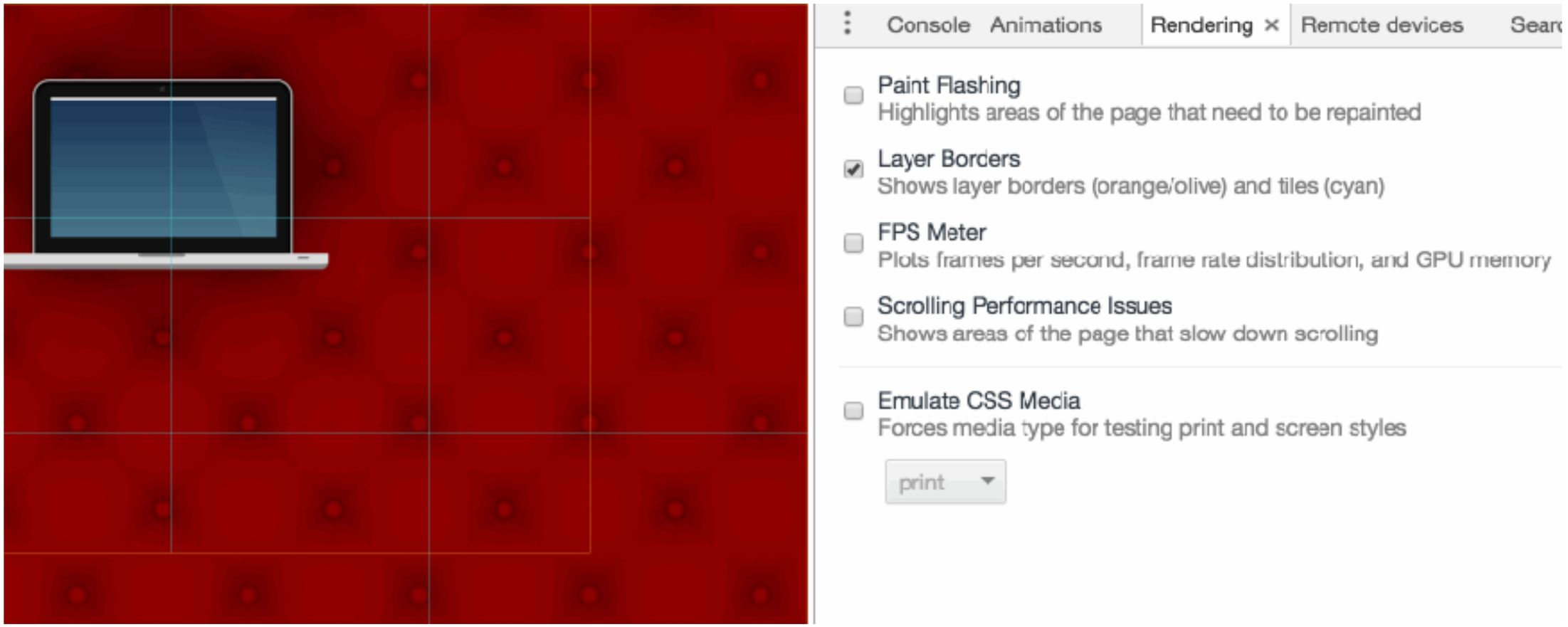
Paint flashing





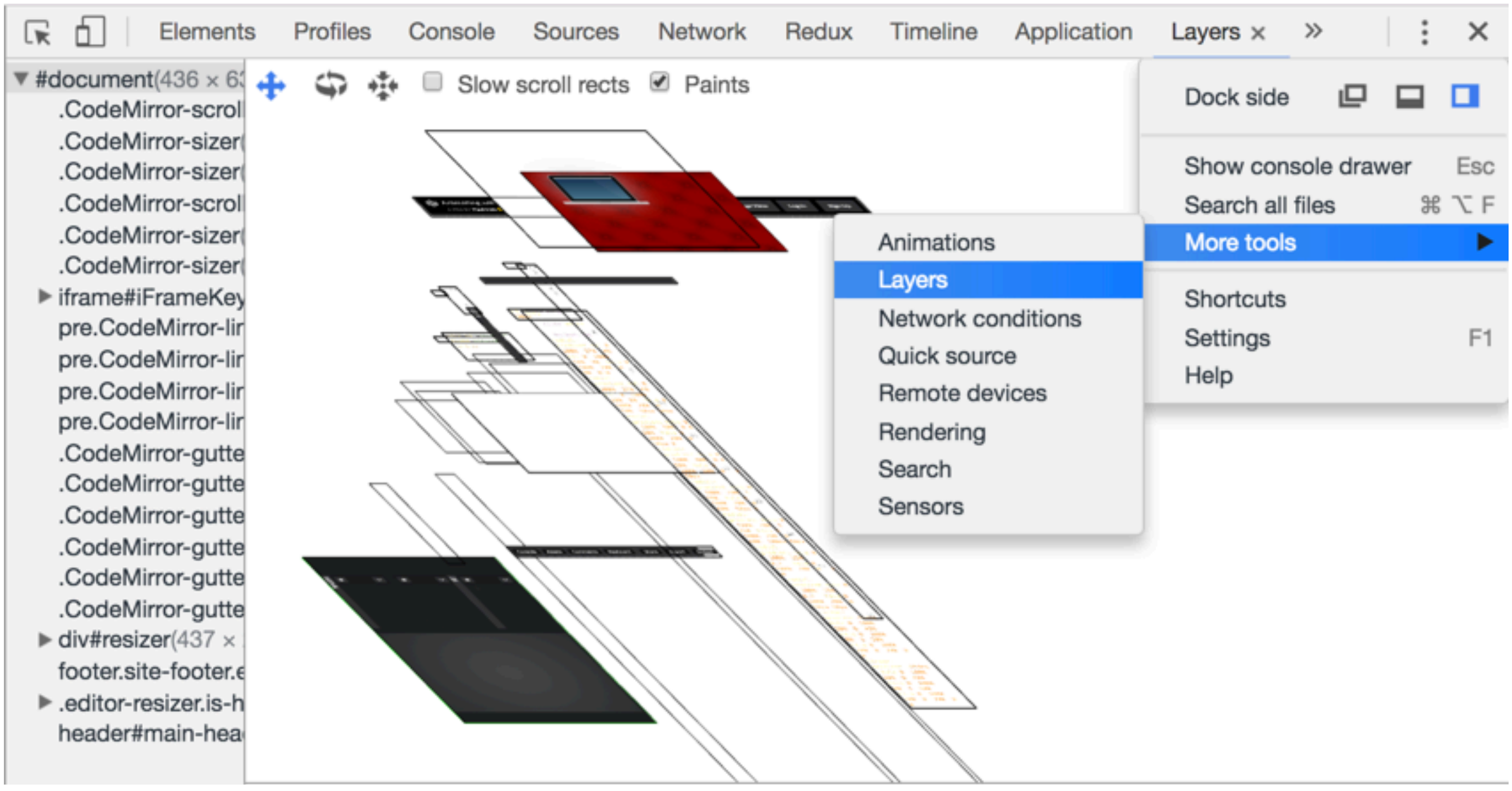
Инструменты

Layer borders



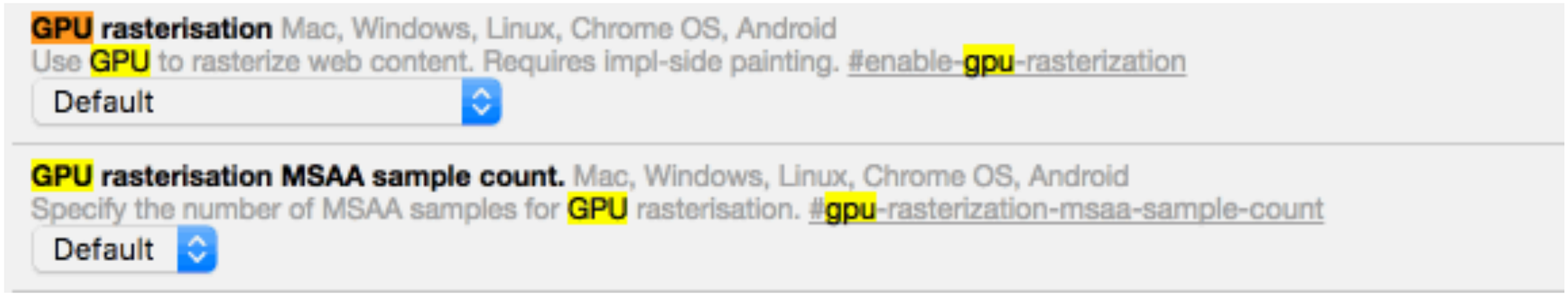
Инструменты

Layers



Подлый флаг в Chrome

Хром рендерит не только Composite на GPU, но ещё и Paint



Снимайте этот флаг при разработке анимаций

Ссылки на полезные ресурсы

- csstriggers.com
- Transform vs absolute — Paul Irish
- FLIP your animations — Aerotwist
- GPU animation: doing it right
- Paul Irish
- Paul Lewis (aerotwist)

Ссылки на полезные ресурсы

- Anatomy of a frame
- How browsers work
- Web Animations API — Dan Wilson
- Web Animations API — Polyfill

Спасибо!

bit.ly/2ryj5yX