

Если ваш сайт тормозит:

1. забудьте про память (*до поры*)
2. разгрузите процессор
3. прикиньтесь, что так и должно быть



**Никто не любит тормоза
в интерфейсе 😡**



Хороший сценарий



Хороший сценарий

- неприятные впечатления от сайта:



Хороший сценарий

- неприятные впечатления от сайта:
- неправильное прицеливание в играх



Хороший сценарий

- неприятные впечатления от сайта:
- неправильное прицеливание в играх
- заметные задержки в анимации



Плохой сценарий



Плохой сценарий

- ошибки при важных операциях



Плохой сценарий

- ошибки при важных операциях
- повторная отправка данных (*например, при оплате*)



Плохой сценарий

- ошибки при важных операциях
- повторная отправка данных (*например, при оплате*)
- неотправка данных



Плохой сценарий

- ошибки при важных операциях
- повторная отправка данных (*например, при оплате*)
- неотправка данных
- уход пользователей



Плохой сценарий

- ошибки при важных операциях
 - повторная отправка данных (*например, при оплате*)
 - неотправка данных
- уход пользователей
- потеря денег компанией



Плохой сценарий

- ошибки при важных операциях
 - повторная отправка данных (*например, при оплате*)
 - неотправка данных
- уход пользователей
- потеря денег компанией
- увольнение



Что такое тормоза?

(по-английски *lag* – задержка) Непредсказуемые задержки в обратной связи в интерфейсе: рывки и замирания



Взаимодействие с пользователем дискретно



Дискретное взаимодействие



Дискретное взаимодействие

- анимация



Дискретное взаимодействие

- анимация
- прокрутка



Дискретное взаимодействие

- анимация
- прокрутка
- взаимодействие с элементами страницы



Дискретное взаимодействие

- анимация
- прокрутка
- взаимодействие с элементами страницы
 - нажатия на кнопки



Дискретное взаимодействие

- анимация
- прокрутка
- взаимодействие с элементами страницы
 - нажатия на кнопки
 - ввод текста



Дискретное взаимодействие

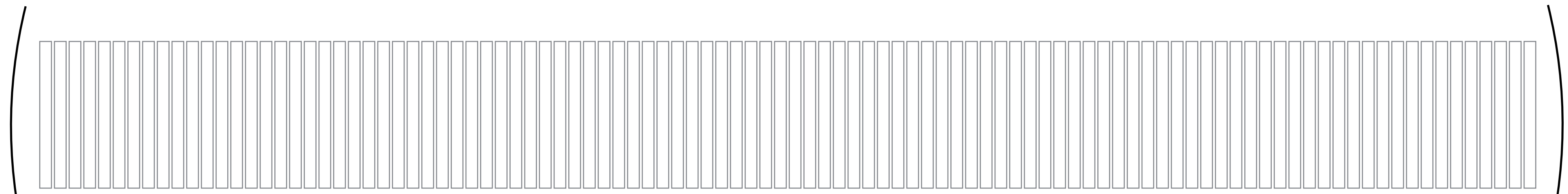
- анимация
- прокрутка
- взаимодействие с элементами страницы
 - нажатия на кнопки
 - ввод текста
 - перетаскивание



Event Loop

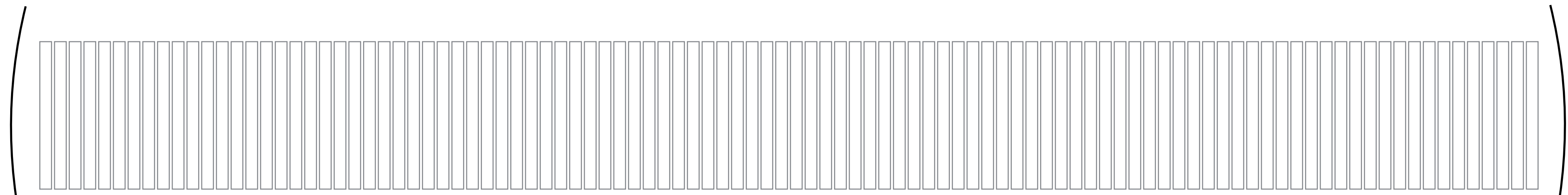


Event Loop



Event Loop

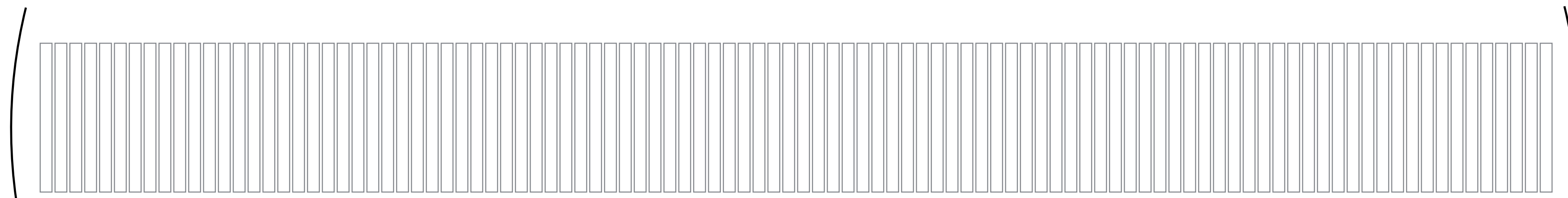
бесконечный цикл, который после выполнения всех инструкций продолжает работать и ждет поступления новых асинхронных команд



Event Loop

бесконечный цикл, который после выполнения всех инструкций продолжает работать и ждет поступления новых асинхронных команд

Благодаря EventLoop работает скролл, перерисовка DOM, все асинхронные операции и т. д.

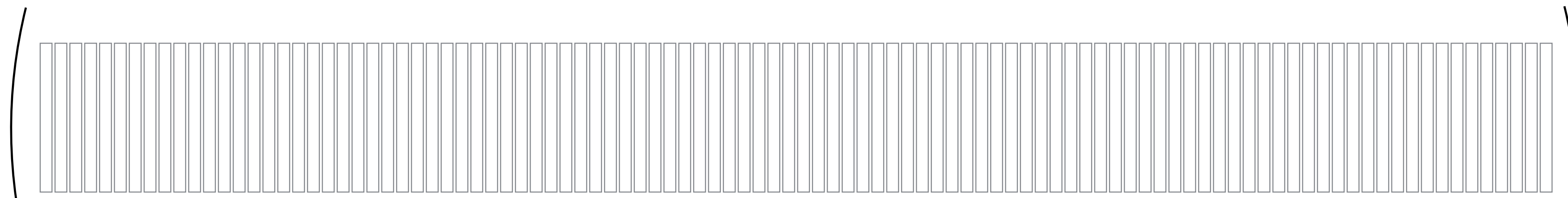


Event Loop

бесконечный цикл, который после выполнения всех инструкций продолжает работать и ждет поступления новых асинхронных команд

Благодаря EventLoop работает скролл, перерисовка DOM, все асинхронные операции и т. д.

Чтобы запустить код в начале кадра нужно передать его как коллбэк в функцию `requestAnimationFrame`

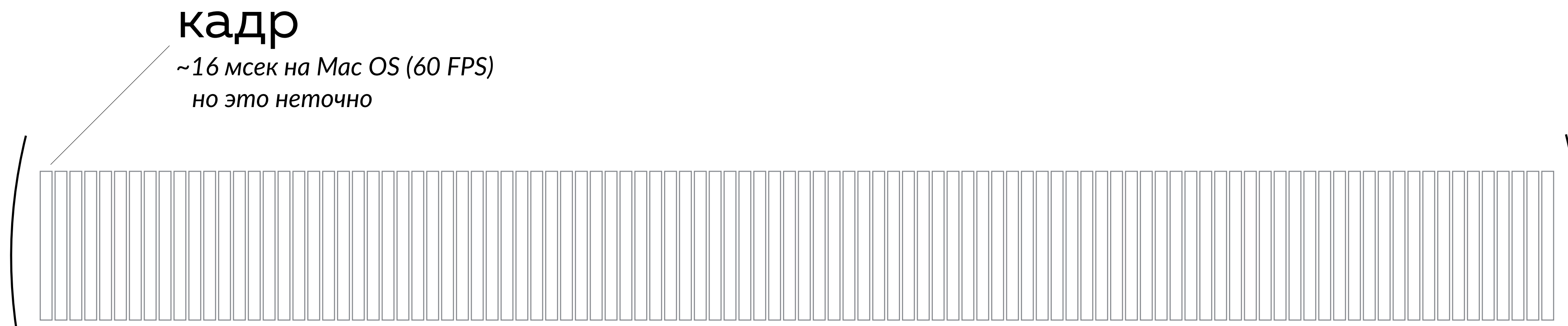


Event Loop

бесконечный цикл, который после выполнения всех инструкций продолжает работать и ждет поступления новых асинхронных команд

Благодаря EventLoop работает скролл, перерисовка DOM, все асинхронные операции и т. д.

Чтобы запустить код в начале кадра нужно передать его как коллбэк в функцию `requestAnimationFrame`

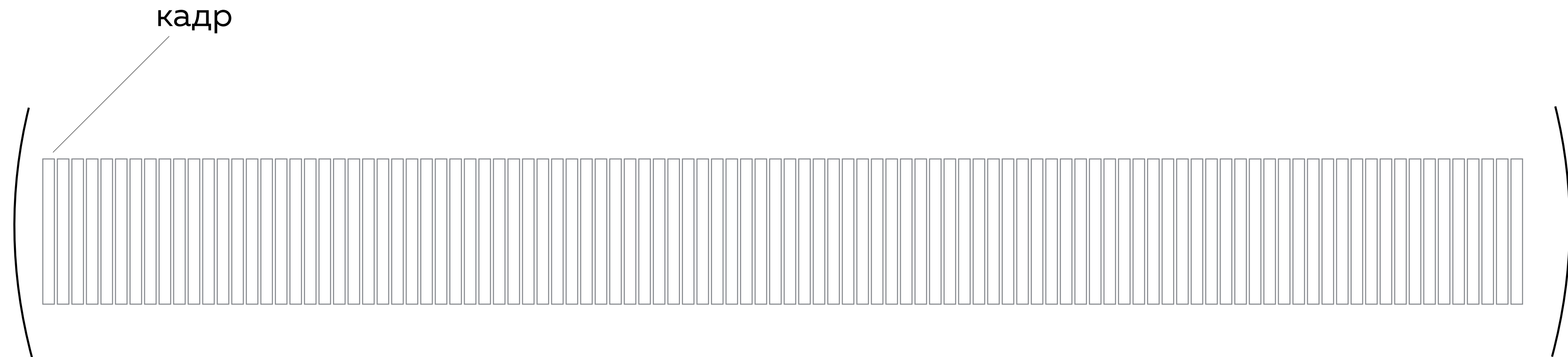


Откуда берутся тормоза

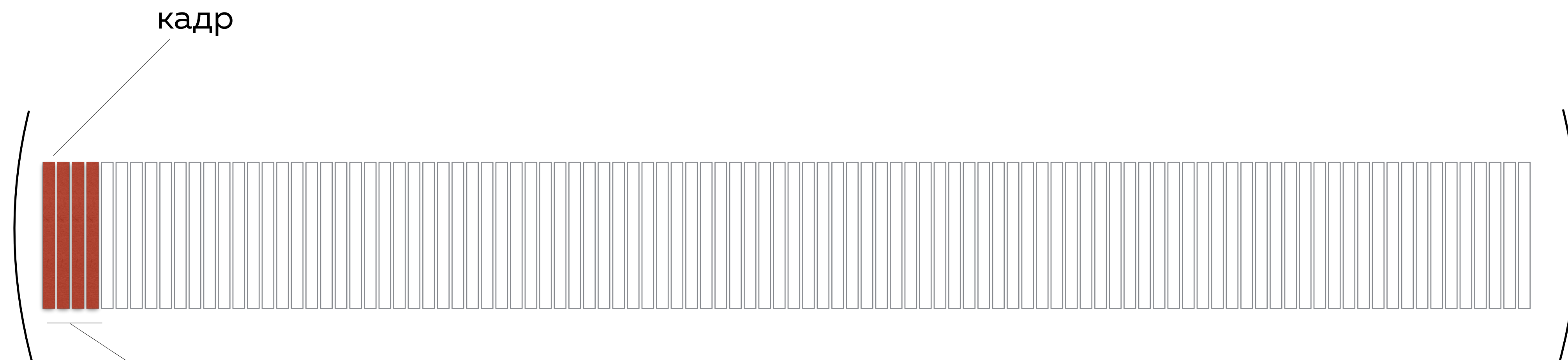
вычисления процессора занимают больше одного кадра
отрисовки и пользователь это замечает



Откуда берутся тормоза?



Откуда берутся тормоза?

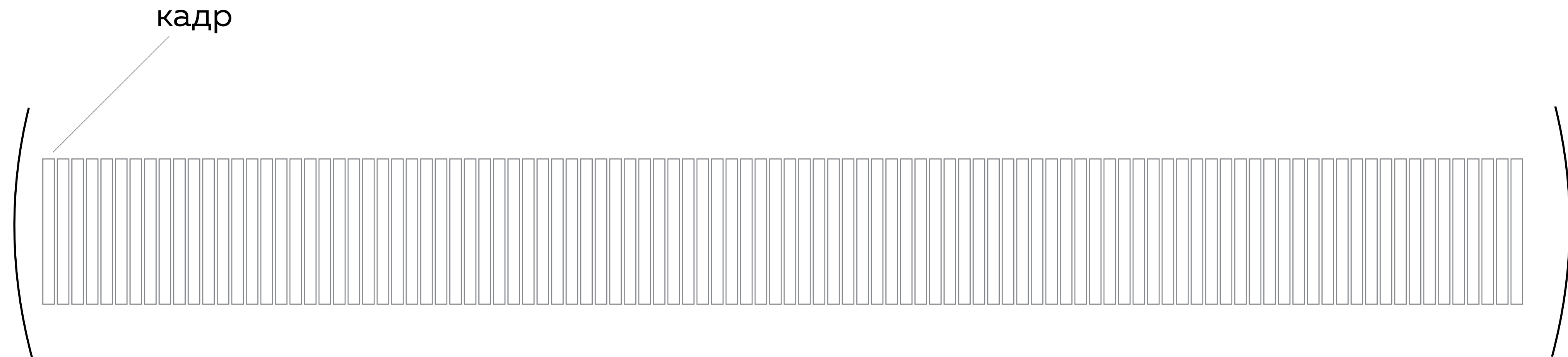


задержка (лаг, lag)

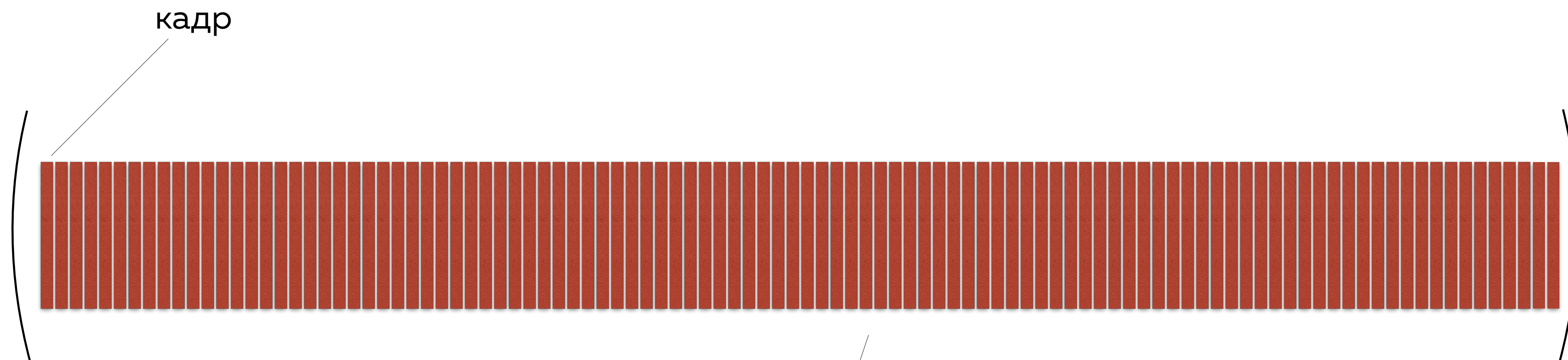
*вычисления занимают больше одного кадра
и при взаимодействии видны рывки*



Откуда берутся тормоза?



Откуда берутся тормоза?



зависание (фриз, freeze)

вычисления не прекращаются, взаимодействие невозможно



Процессор

*используется для мгновенных
вычислений*



Память

*используется для хранения
вычисленных значений*



Память



Запомнить результат вычислений



Запомнить результат вычислений

- объект `Math` помнит популярные результаты вычислений
(π , e , логарифм 10, корень из 2)



Запомнить результат вычислений

- объект `Math` помнит популярные результаты вычислений (*Пи, E, логарифм 10, корень из 2*)
- таблицы синусов и косинусов, рассчитанные на этапе компиляции в старых играх

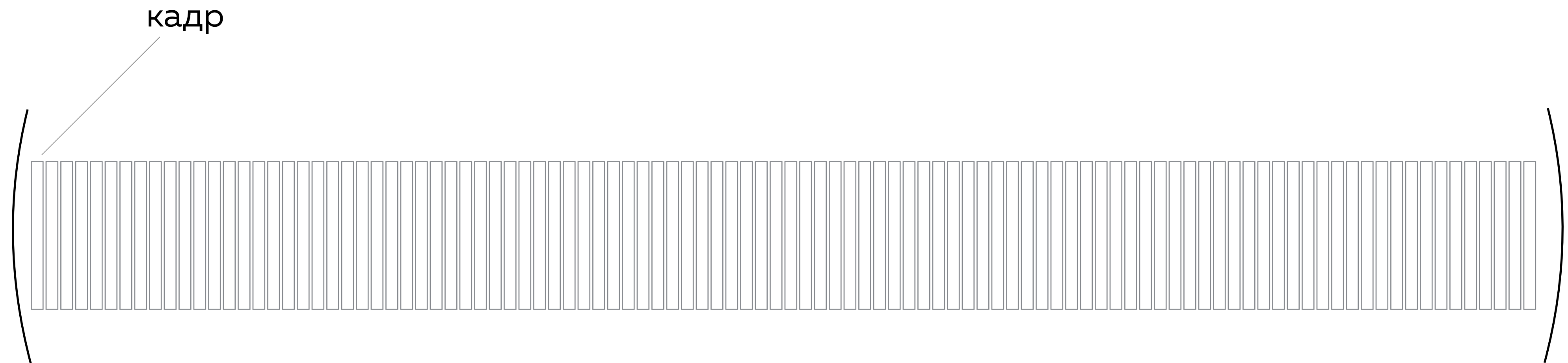


Запомнить результат вычислений

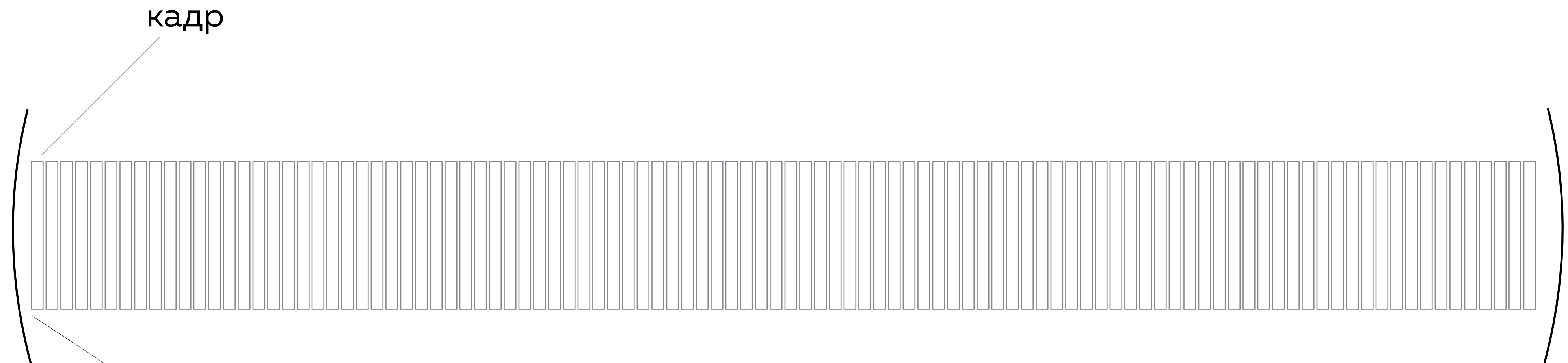
- объект `Math` помнит популярные результаты вычислений (*Пи, E, логарифм 10, корень из 2*)
- таблицы синусов и косинусов, рассчитанные на этапе компиляции в старых играх
- заранее просчитанные анимации



Запомнить результат вычислений



Запомнить результат вычислений



кадр

получение уже известного
результата почти
не занимает времени



Что хранится в памяти



Что хранится в памяти

- **все конструкции языка**
переменные, значения, функции



Что хранится в памяти

- **все конструкции языка**
переменные, значения, функции
- **данные программы**
загруженная информация, созданные объекты



Что хранится в памяти

- **все конструкции языка**
переменные, значения, функции
- **данные программы**
загруженная информация, созданные объекты
- **DOM-дерево**
для каждого элемента на странице создается соответствующий JS-объект (*даже для переносов и комментариев*)



Что хранится в памяти

- **все конструкции языка**
переменные, значения, функции
- **данные программы**
загруженная информация, созданные объекты
- **DOM-дерево**
для каждого элемента на странице создается соответствующий JS-объект (*даже для переносов и комментариев*)
- ...



Почему тормозит память

в неожиданные моменты времени происходит сборка мусора или происходит утечка памяти, которая засоряет память лишними объектами



Почему тормозит память



Почему тормозит память

- сборка мусора

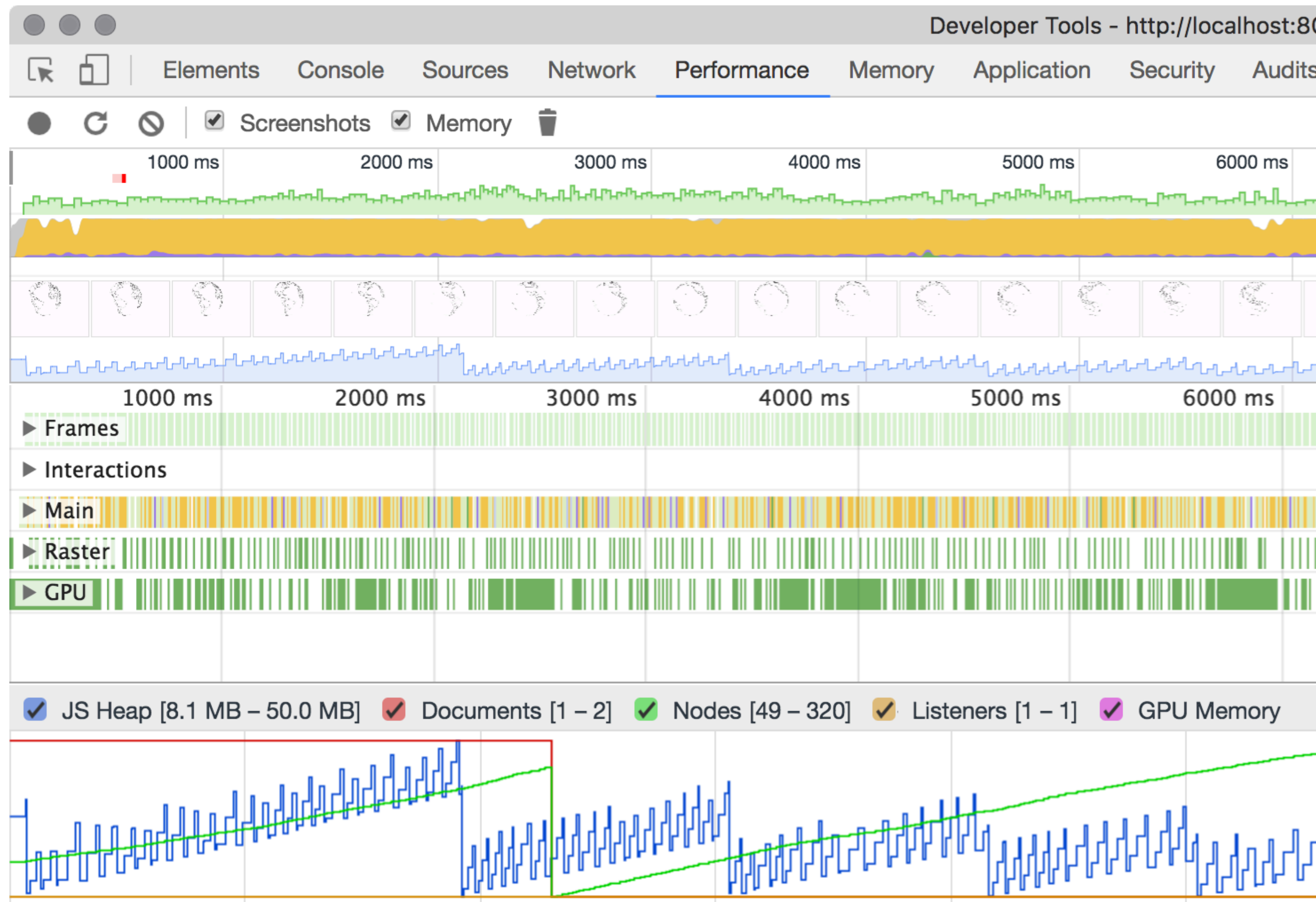


Почему тормозит память

- сборка мусора
- утечки



Профилирование



Профилирование

- таймлайн:



Профилирование

- таймлайн:
- частота кадров



Профилирование

- таймлайн:
- частота кадров
- загруженность процессора



Профилирование

- таймлайн:
- частота кадров
- загруженность процессора
- скриншот



Профилирование

- таймлайн:
- частота кадров
- загруженность процессора
- скриншот
- память

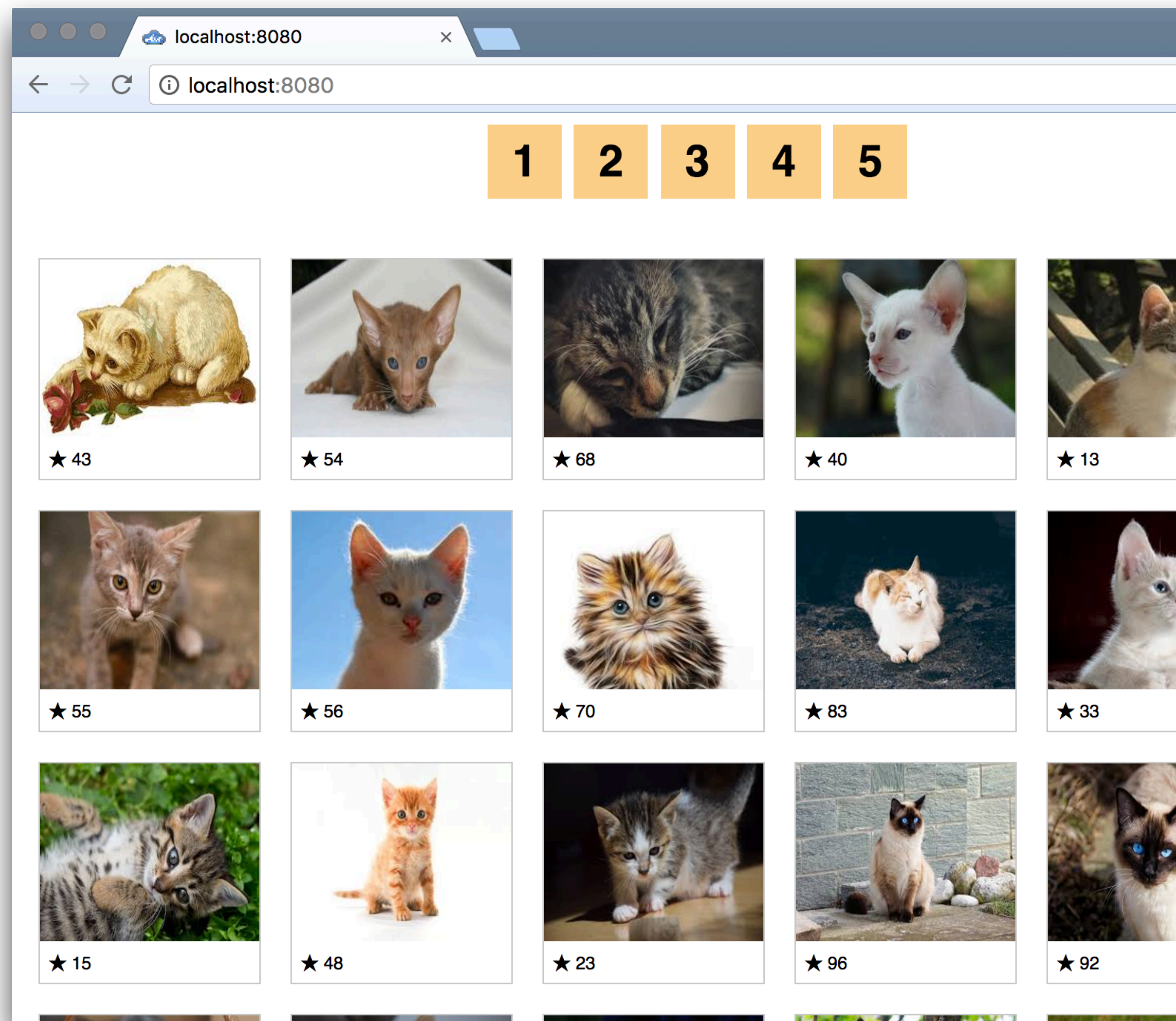


Профилирование

- таймлайн:
- частота кадров
- загруженность процессора
- скриншот
- память
- детальная статистика

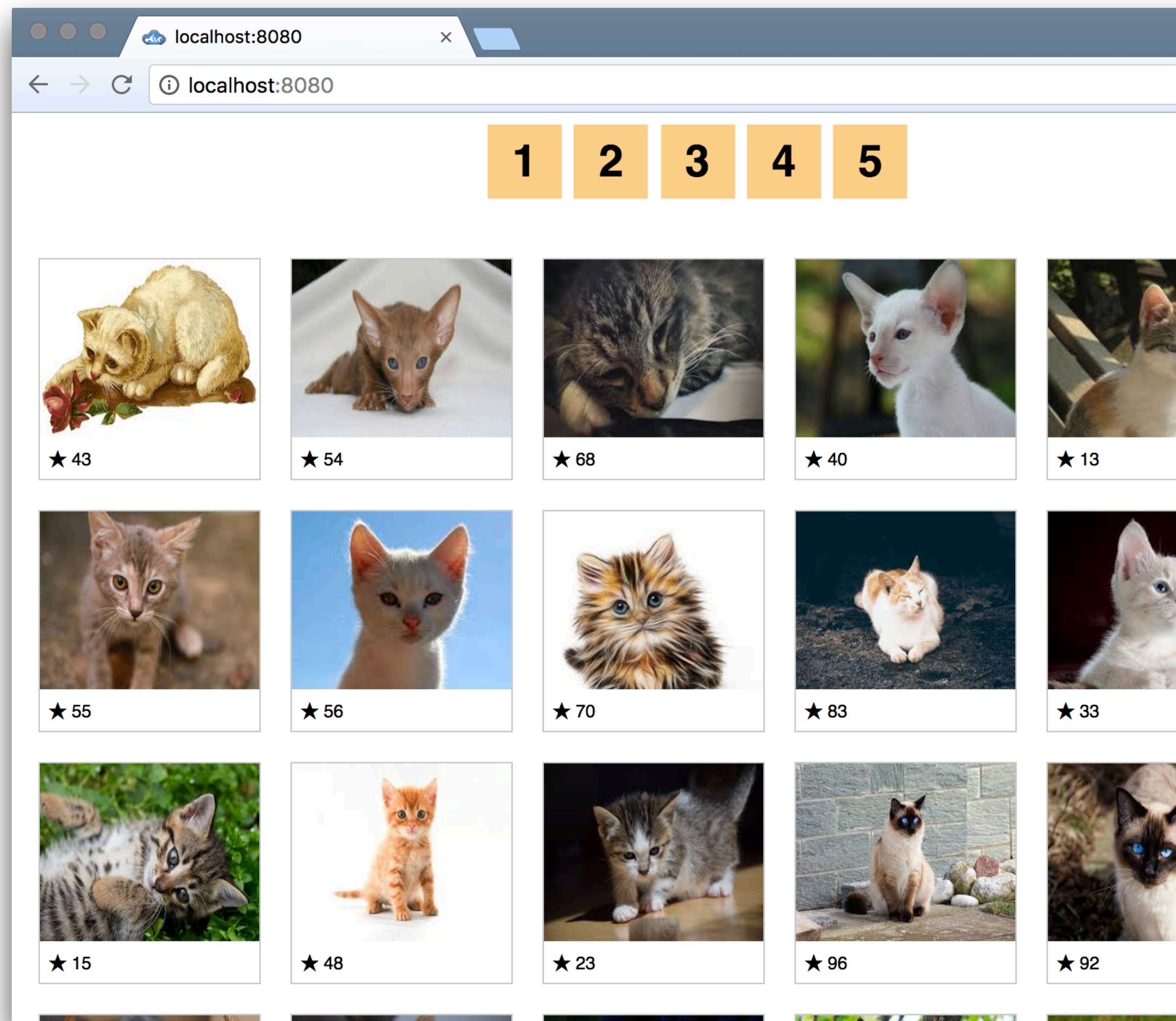


Тестовый DOM



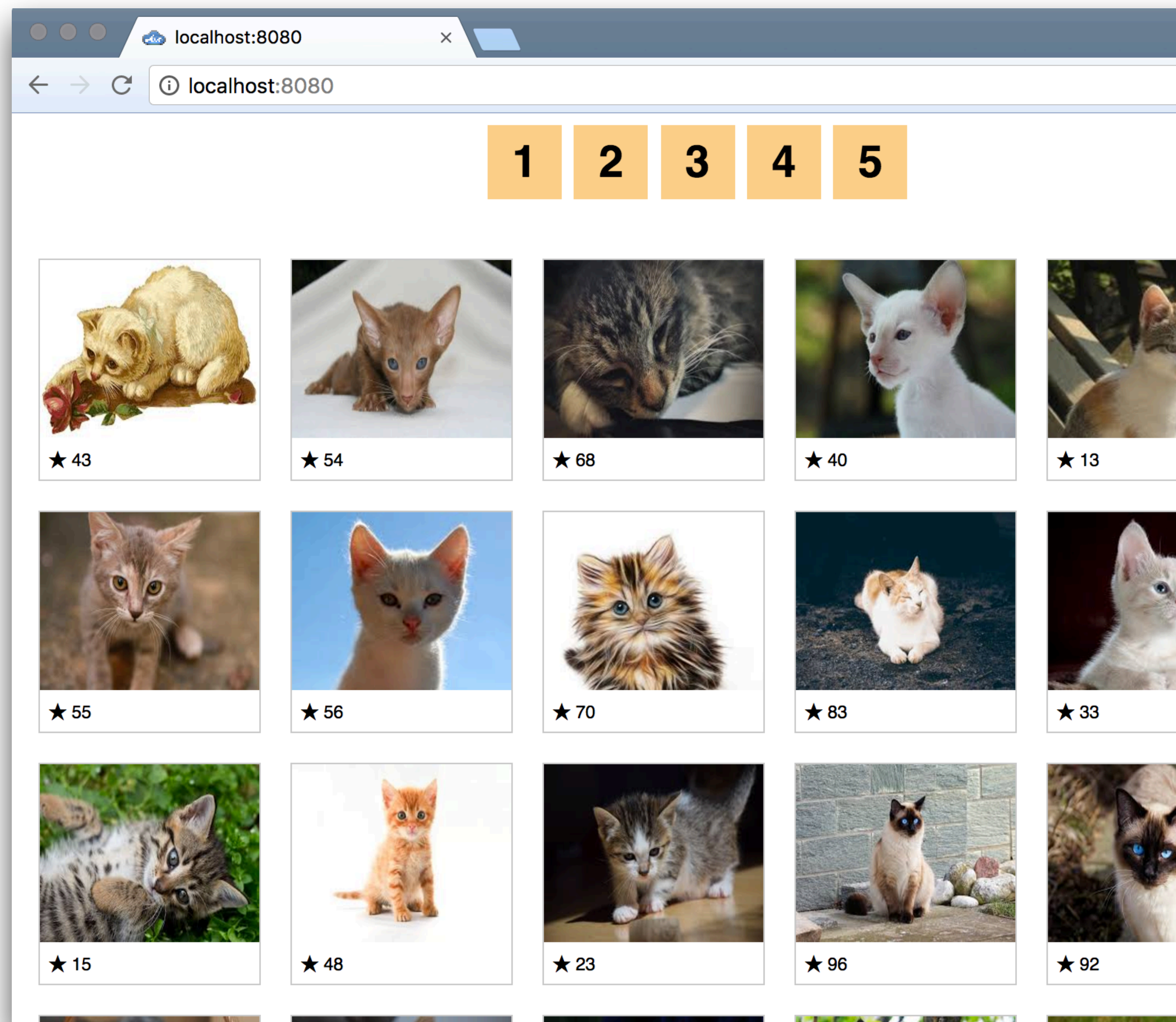
Тестовый DOM

- пять страниц по 5000 котиков



Тестовый DOM

- пять страниц по 5000 котиков
- страницы переключаются с помощью пагинатора



Тестовый DOM

```
let i = pageSize;
while (i--) {
  const el = templateElement.cloneNo
  el.querySelector(`.pic-thumb`).src
  el.querySelector(`.pic-like-counte
    Math.random() * 90) + 10;

  yield el;
}
```



Тестовый DOM

- элементы создаются из шаблона

```
let i = pageSize;
while (i-- > 0) {
  const el = templateElement.cloneNo
  el.querySelector(`.pic-thumb`).src
  el.querySelector(`.pic-like-counte
    Math.random() * 90) + 10;

  yield el;
}
```



Тестовый DOM

- элементы создаются из шаблона
- заполняются случайными данными (*нет кэширования*)

```
let i = pageSize;
while (i--) {
  const el = templateElement.cloneNo
  el.querySelector(`.pic-thumb`).src
  el.querySelector(`.pic-like-counte
    Math.random() * 90) + 10;

  yield el;
}
```



Тестовый DOM

- элементы создаются из шаблона
- заполняются случайными данными (*нет кеширования*)
- добавляются в один фрагмент (*DocumentFragment*)

```
for (const el of pageSwitcher.next())  
  fragment.appendChild(el);  
}
```



Тестовый DOM

страницы переключаются
простой очисткой контейнера

```
container.innerHTML = '';
```



Сборка мусора

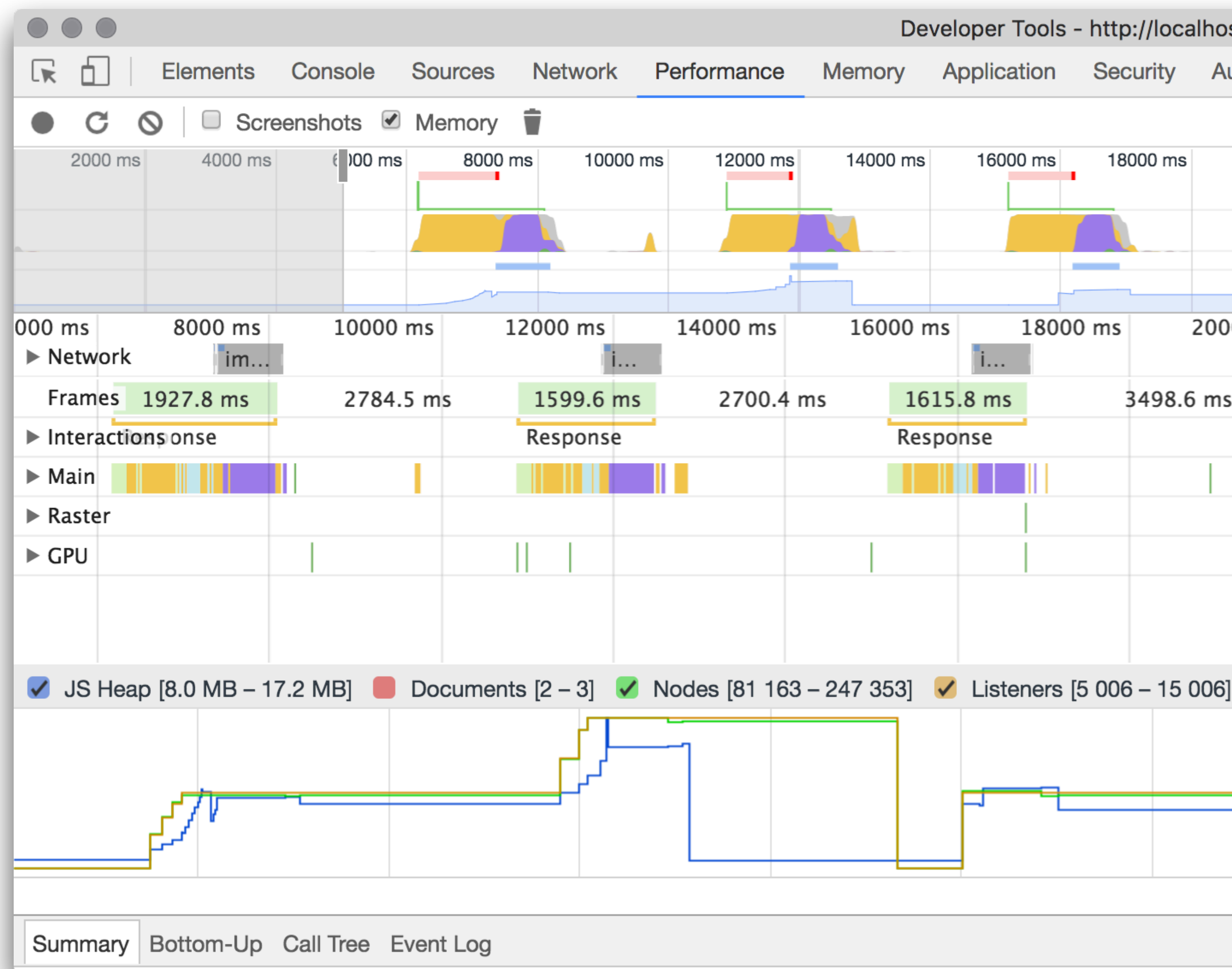


Сборка мусора —

(*Garbage Collection, GC*) процесс удаления неиспользуемых объектов из памяти. В JS начинается автоматически, когда движок понимает, что свободная память закончилась и нужно освободить место

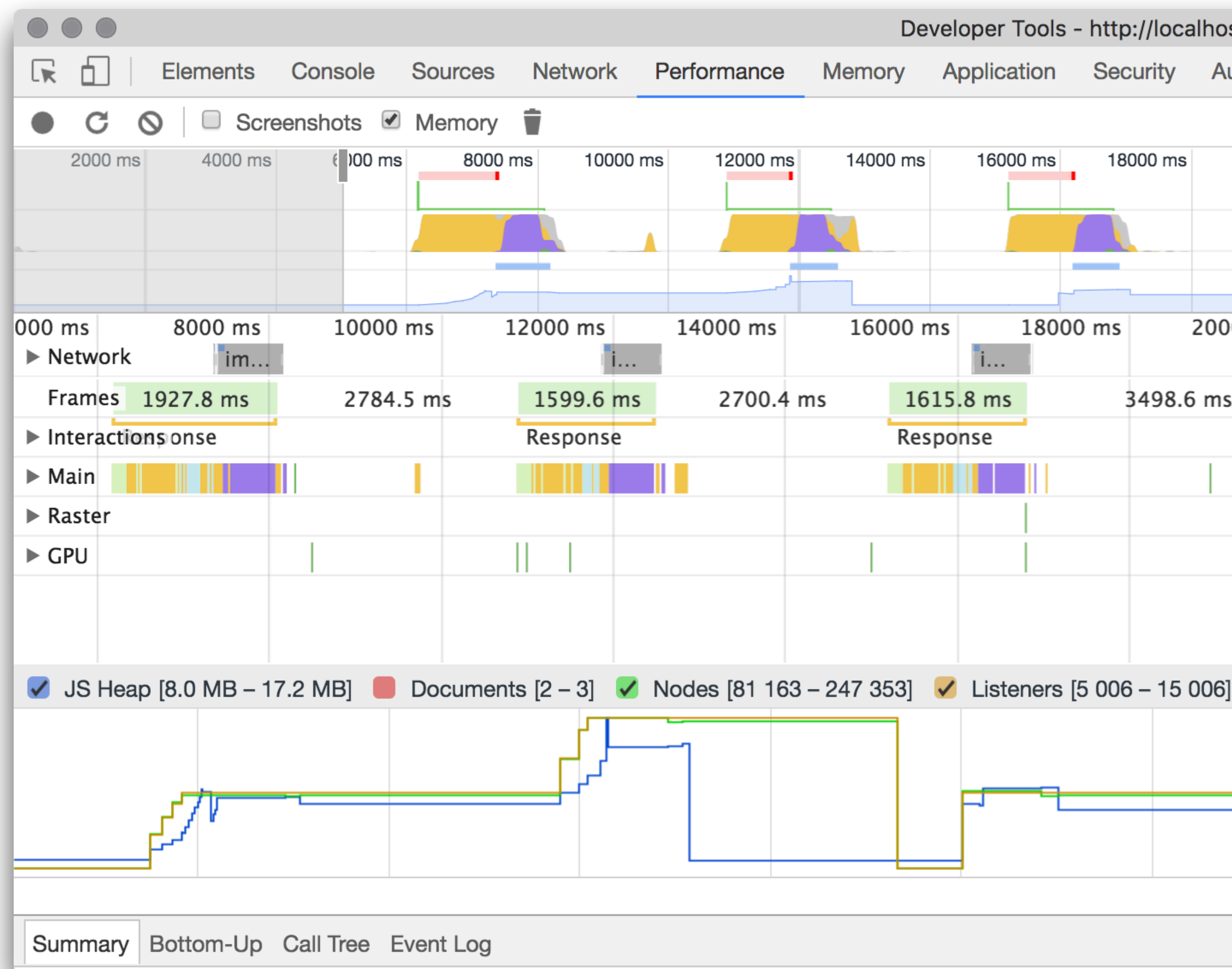


Сборка мусора



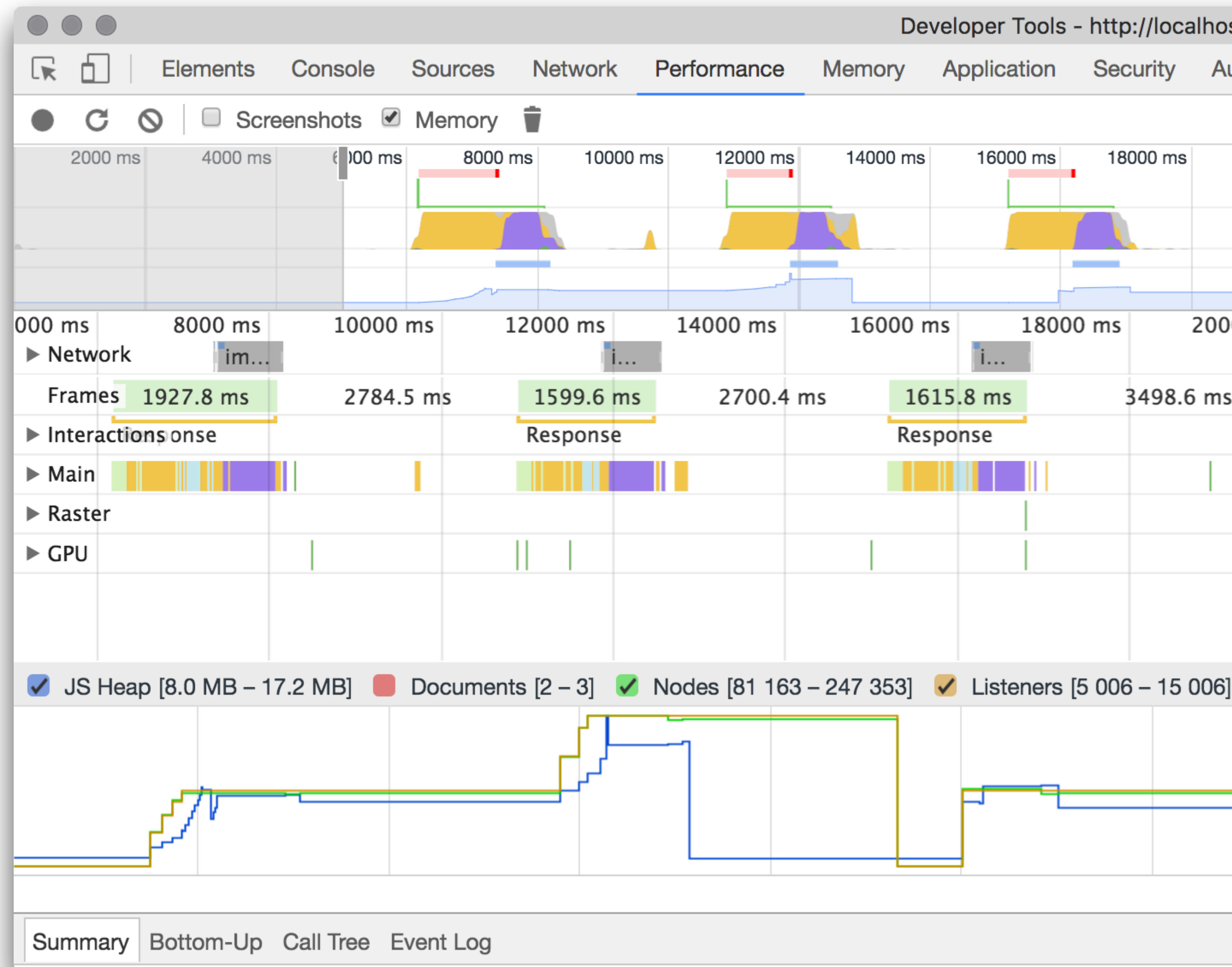
Сборка мусора

- когда память, выделенная под вкладку заканчивается, включается механизм GC



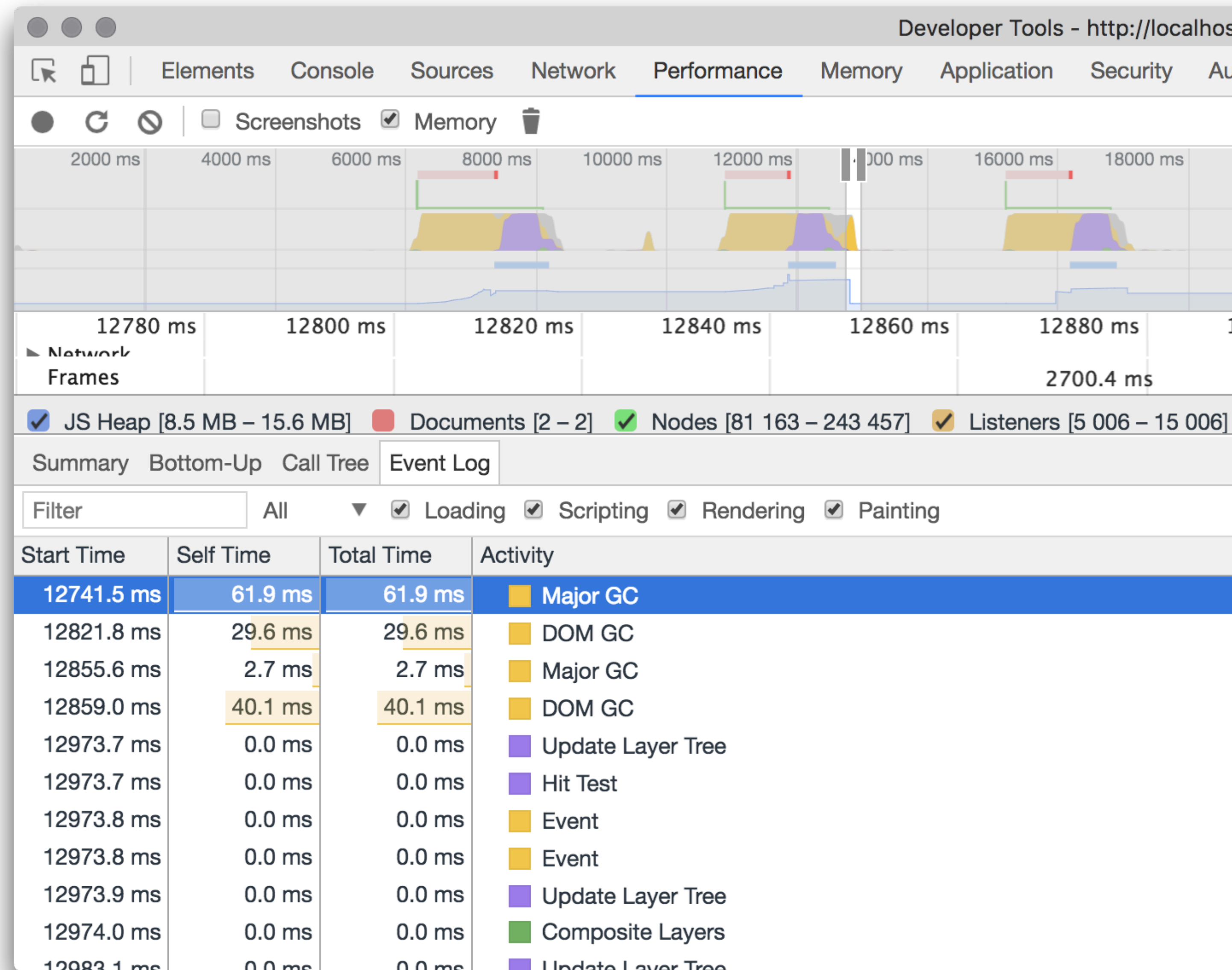
Сборка мусора

- когда память, выделенная под вкладку заканчивается, включается механизм GC
- процесс сборки мусора автоматический: предсказать его начало невозможно



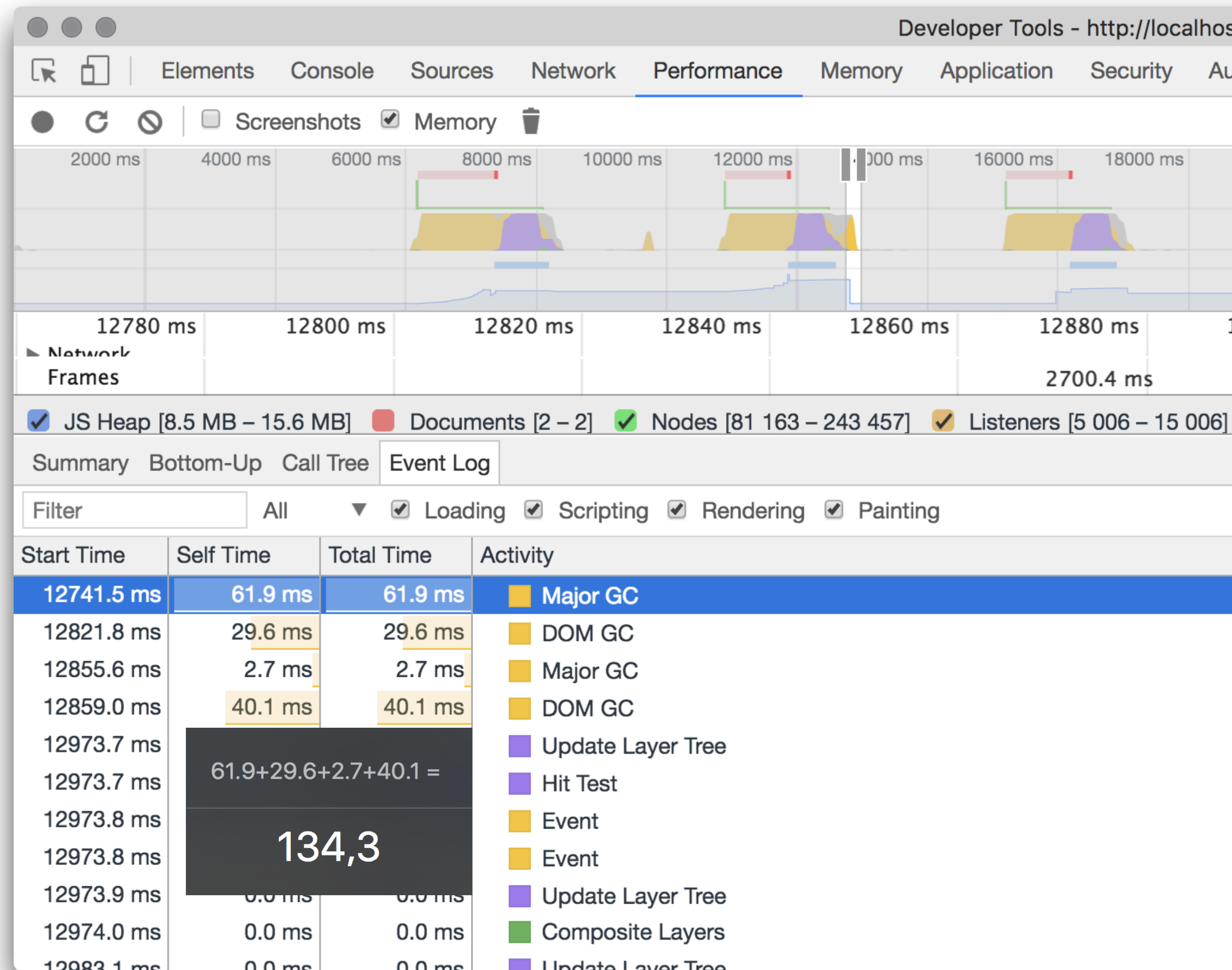
Сборка мусора

- когда память, выделенная под вкладку заканчивается, включается механизм GC
- процесс сборки мусора автоматический: предсказать его начало невозможно
- в некоторых случаях GC может работать очень медленно



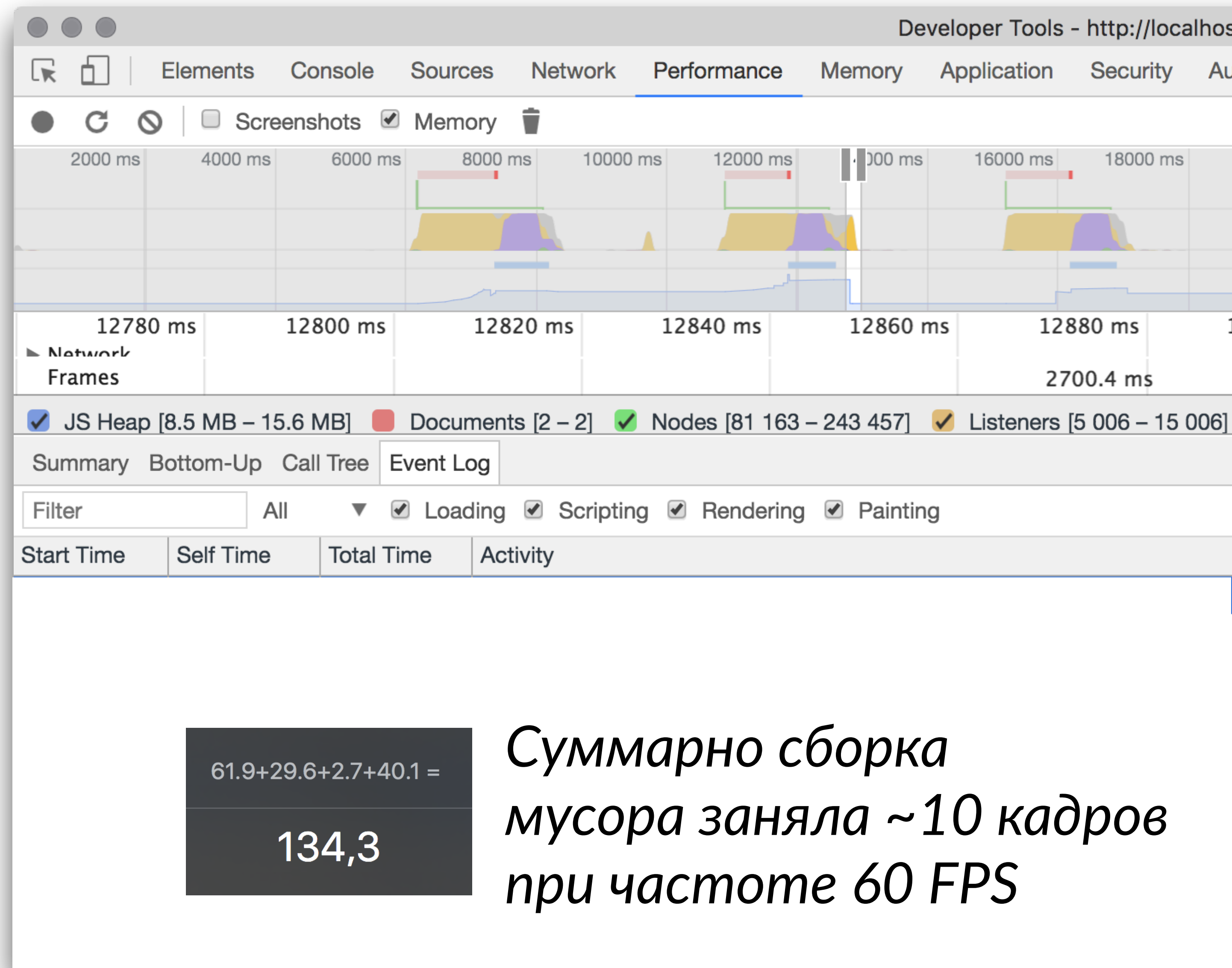
Сборка мусора

- когда память, выделенная под вкладку заканчивается, включается механизм GC
- процесс сборки мусора автоматический: предсказать его начало невозможно
- в некоторых случаях GC может работать очень медленно



Сборка мусора

- когда память, выделенная под вкладку заканчивается, включается механизм GC
- процесс сборки мусора автоматический: предсказать его начало невозможно
- в некоторых случаях GC может работать очень медленно



Суммарно сборка мусора заняла ~10 кадров при частоте 60 FPS



Утечка памяти



Утечка памяти —

ситуация, когда при сборке мусора, некоторые неиспользуемые объекты остаются в памяти, потому что сборщик мусора считает что они могут использоваться



Утечка памяти

```
const switchPage = (pageSwitcher, pageContainer) => {
  pageContainer.innerHTML = '';

  for (const el of pageSwitcher.nextElements()) {
    document.addEventListener('keydown', (evt) => {
      evt.stopPropagation();
      console.log(el, Math.random());
    });
    pageContainer.appendChild(el);
  }
};
```



Утечка памяти

- после удаления ноды, GC не может определить, что обработчик на документе не используется и не удаляет его

```
const switchPage = (pageSwitcher, page) => {
  container.innerHTML = '';

  for (const el of pageSwitcher.nextPageElements) {
    document.addEventListener('keydown', (evt) => {
      evt.stopPropagation();
      console.log(el, Math.random());
    });
    container.appendChild(el);
  }
};
```



Утечка памяти

- после удаления ноды, GC не может определить, что обработчик на документе не используется и не удаляет его
- обработчик через замыкание использует DOM-ноду, поэтому она тоже не удаляется из памяти

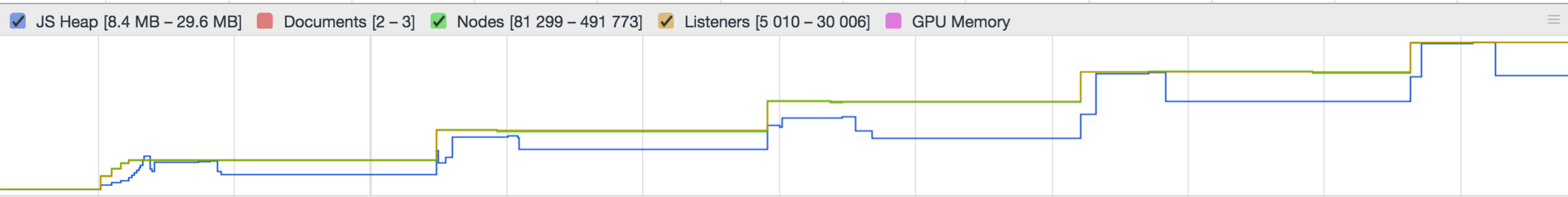
```
const switchPage = (pageSwitcher, page) => {
  container.innerHTML = '';

  for (const el of pageSwitcher.nextPageElements) {
    document.addEventListener('keydown', (evt) => {
      evt.stopPropagation();
      console.log(el, Math.random());
    });
    container.appendChild(el);
  }
};
```



Утечка памяти

- после удаления ноды, GC не может определить, что обработчик на документе не используется и не удаляет его
- обработчик через замыкание использует DOM-ноду, поэтому она тоже не удаляется из памяти



...it is a garbage garbage-collected language. It has performance uncertainty.

Performance unpredictability is one way to put it, where you may be giving something at 60 frames a second for a game and suddenly, you run out of real-time because of a garbage collection that has to happen to reclaim memory

—Брендан Айк, создатель JS в одном из недавних подкастов



Память ненадежна

тормоза, связанные с памятью могут происходить и при записи в неё значений и при её автоматической очистке. Прогнозировать момент возникновения тормозов очень сложно



Процессор



Как ускорить работу процессора



Как ускорить работу процессора

- уменьшить объем вычислений



Как ускорить работу процессора

- уменьшить объем вычислений
- затротлить



Как ускорить работу процессора

- уменьшить объем вычислений
- затротлить 



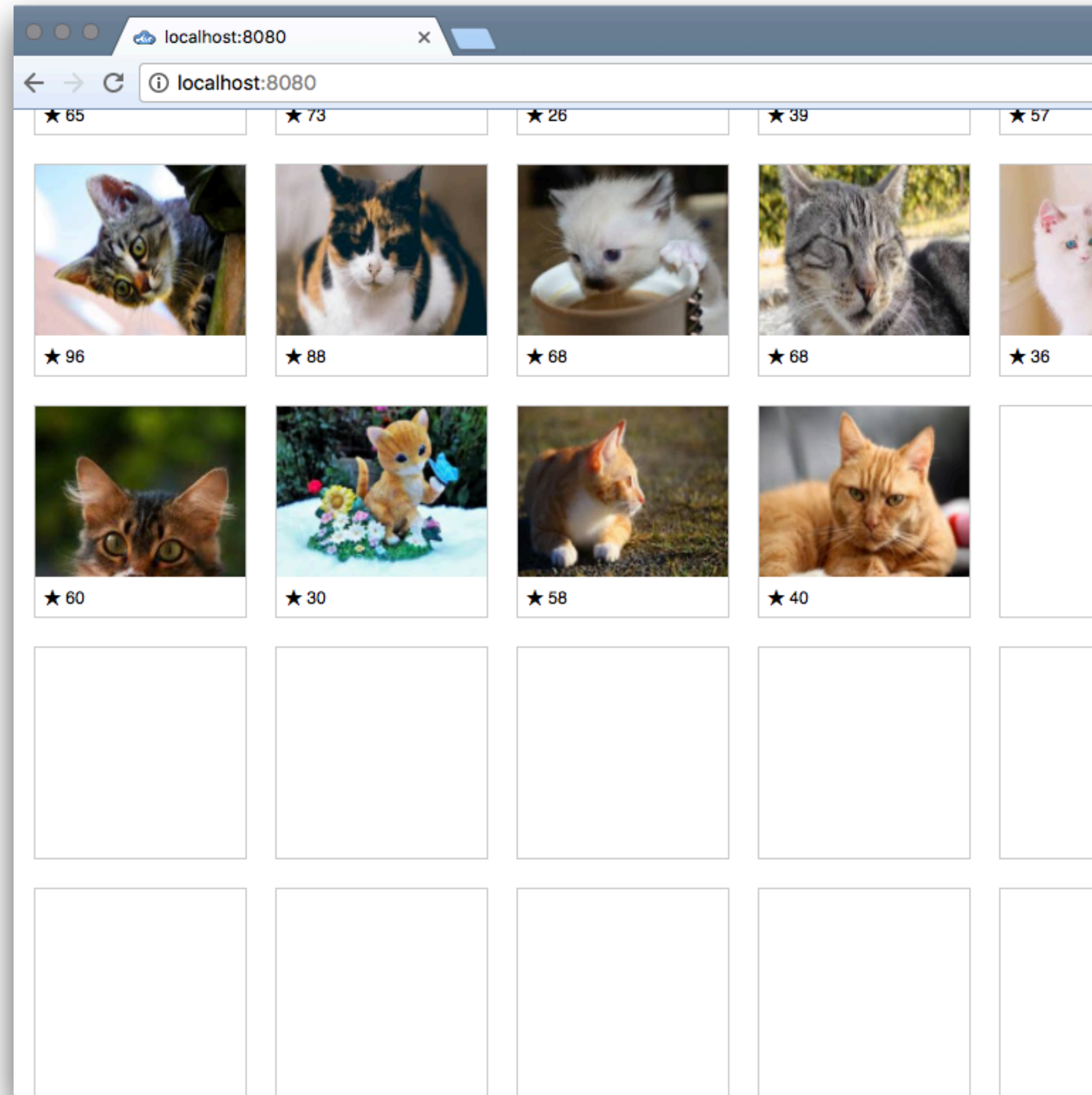
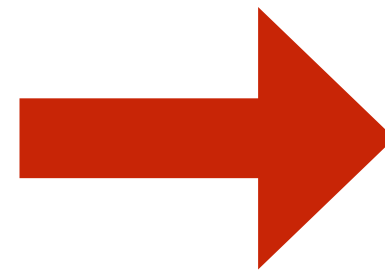
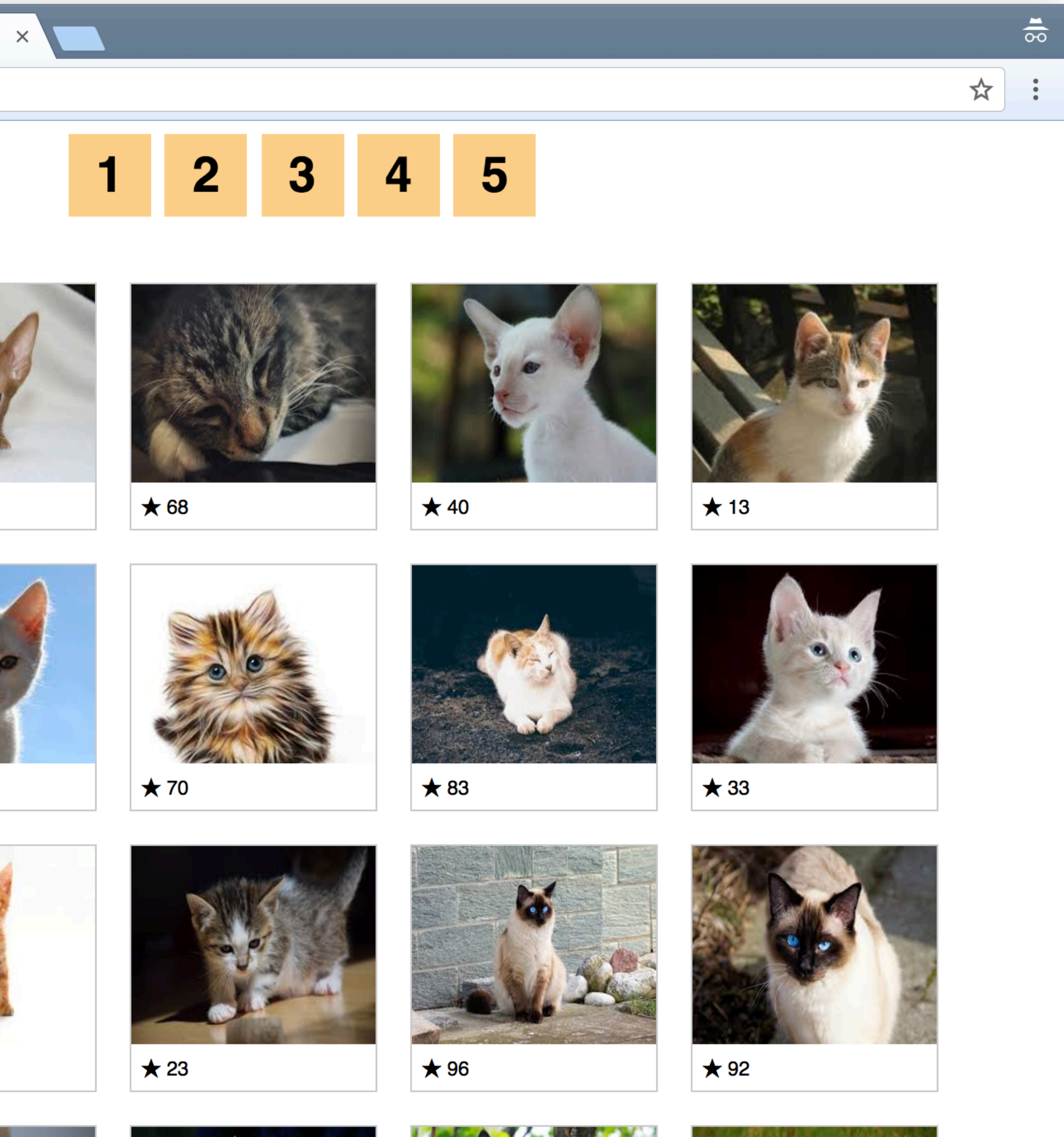
Как ускорить работу процессора

- уменьшить объем вычислений
- затротлить 
- воспользоваться другими инструментами расчёта



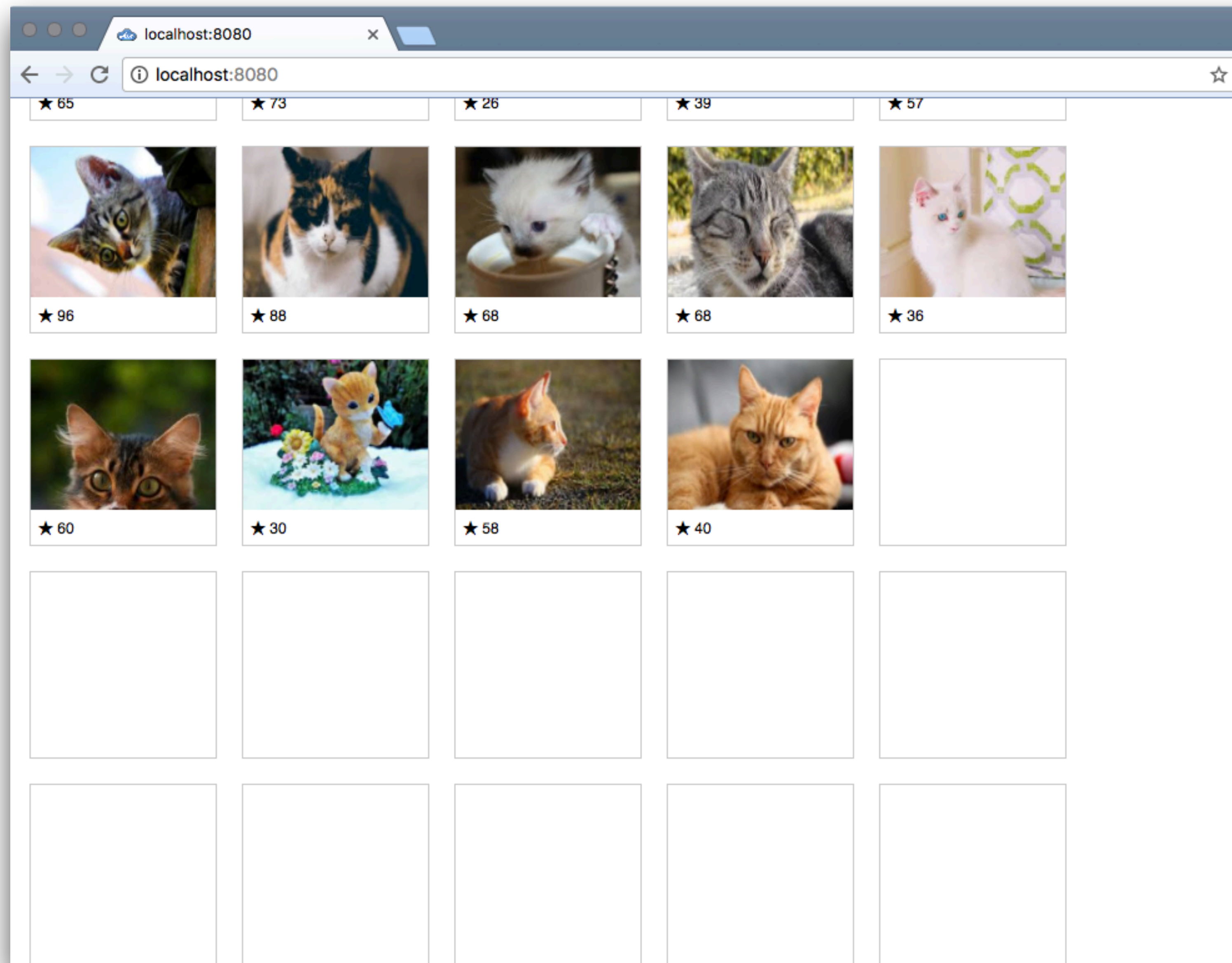
Уменьшение объема вычислений



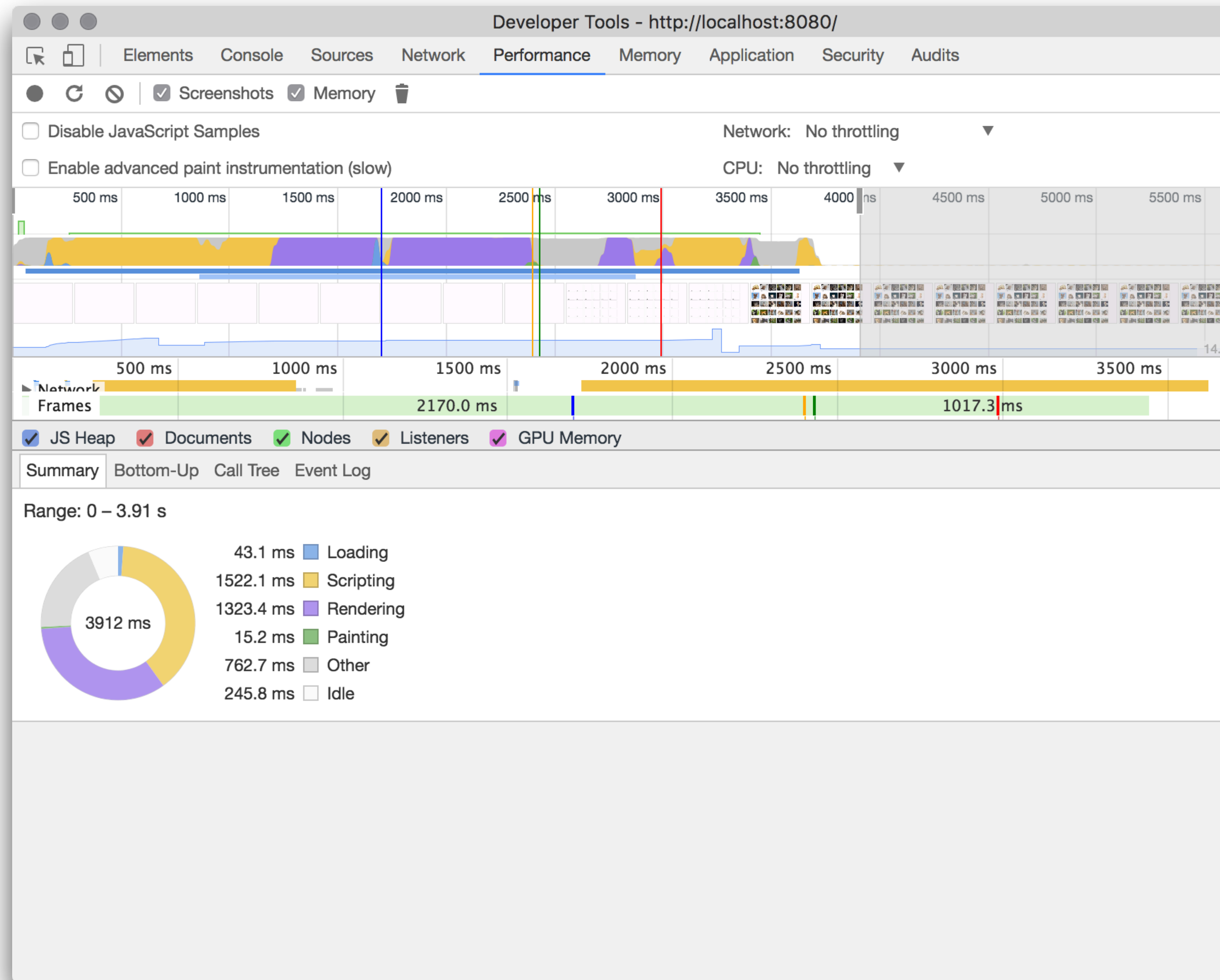


Динамическая прокрутка

показывать пользователю только те элементы, которые находятся в его поле зрения. Остальные элементы отрисовывать по мере необходимости

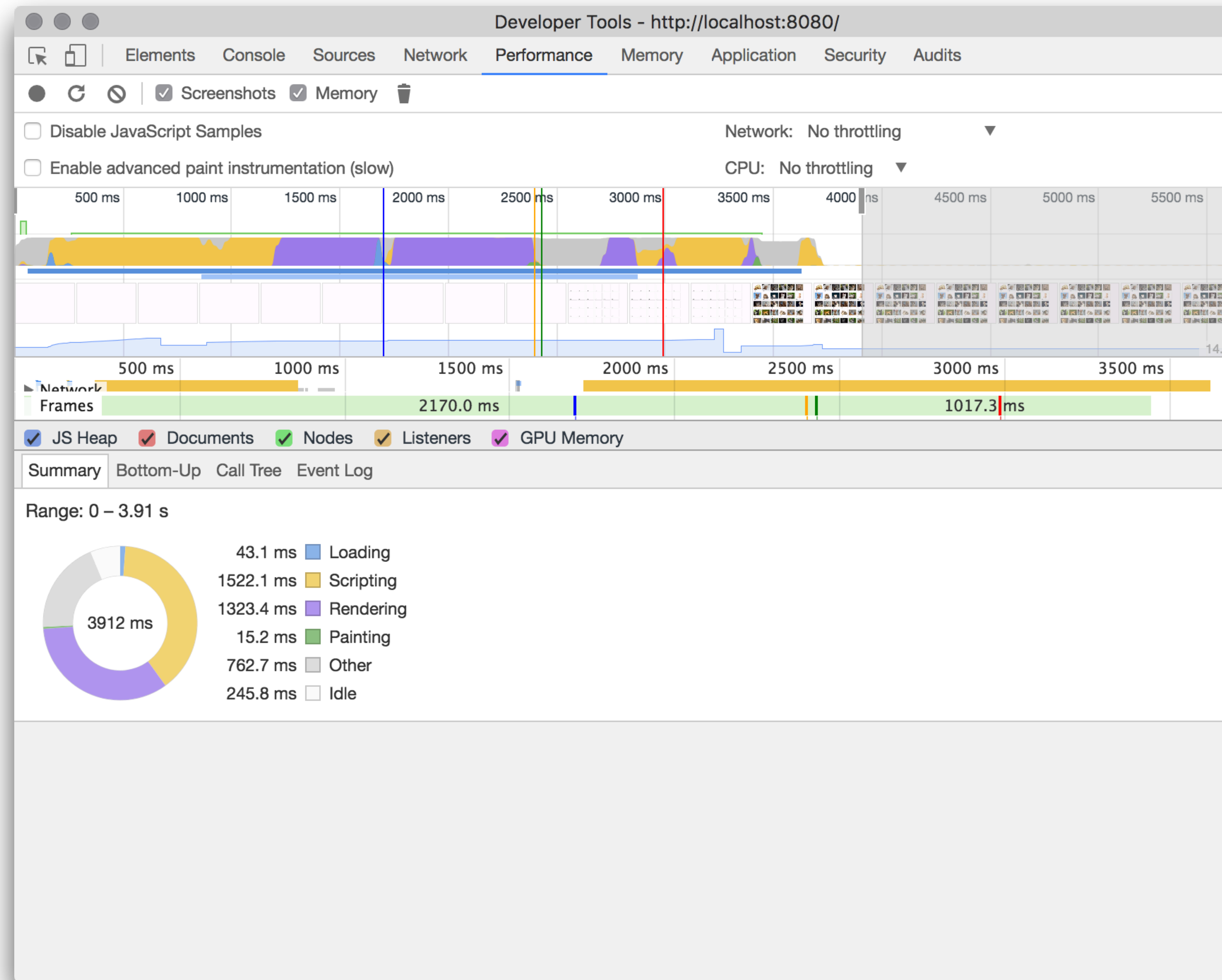


Первая страница на 5000 элементов



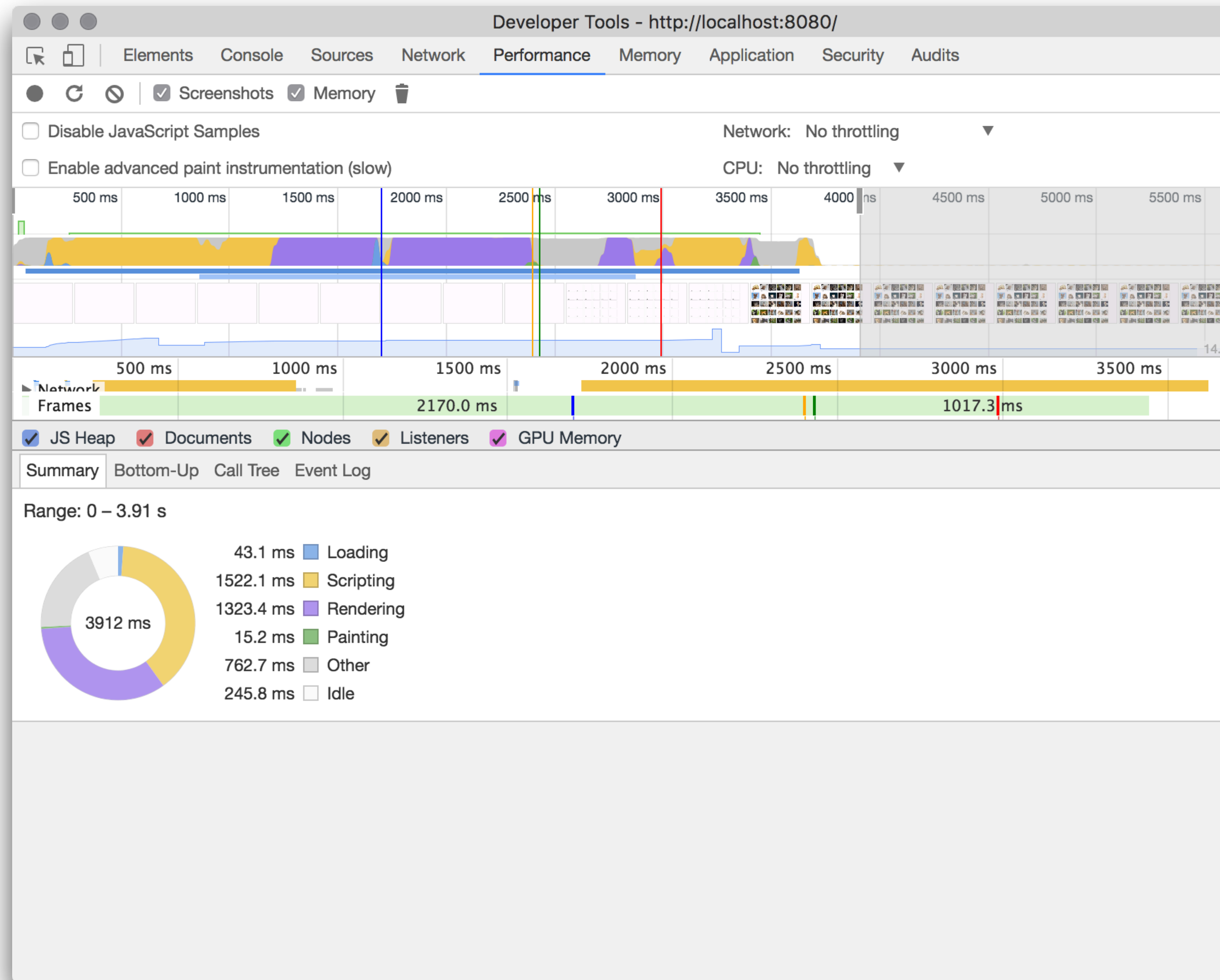
Первая страница на 5000 элементов

- полная отрисовка заняла четыре секунды



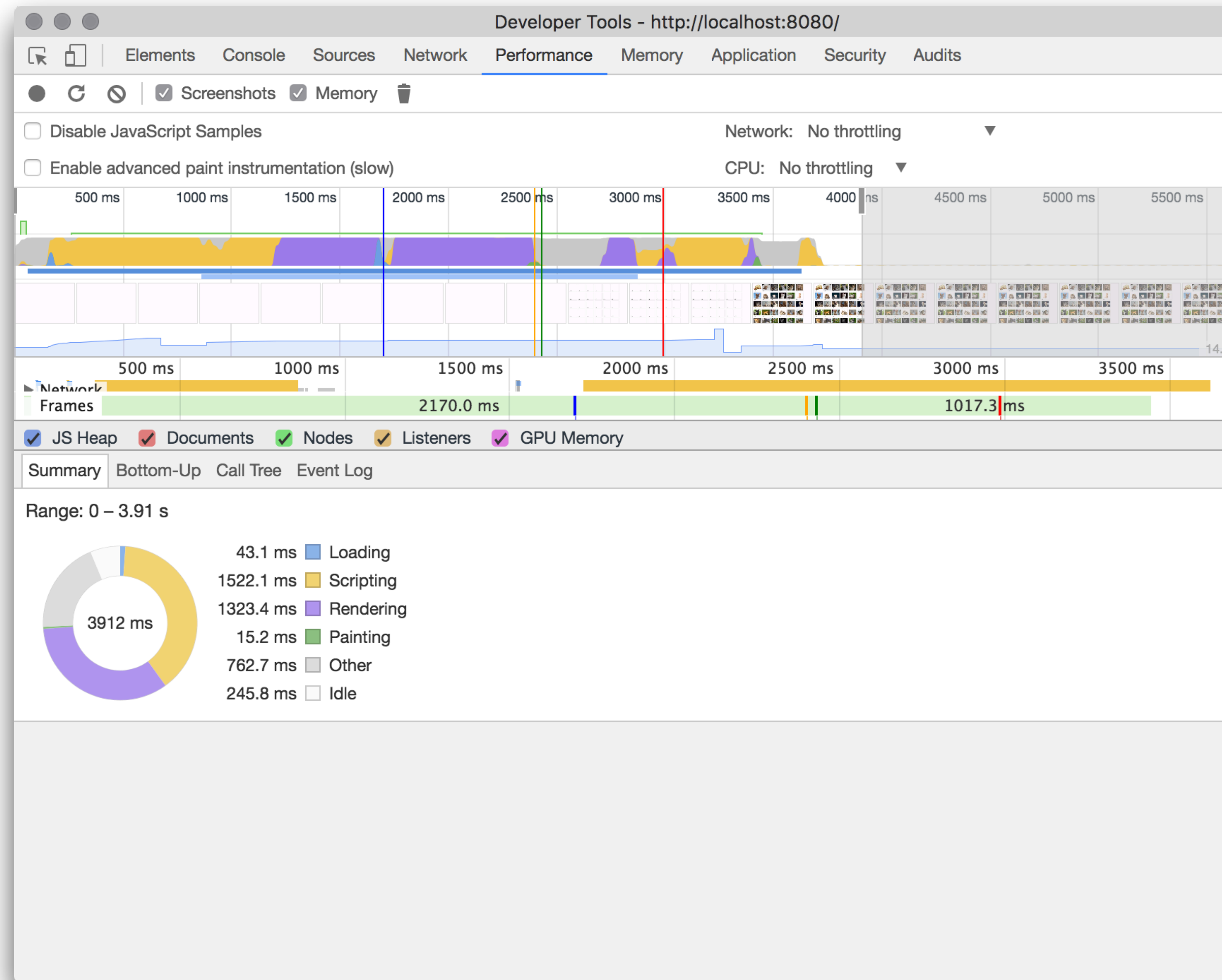
Первая страница на 5000 элементов

- полная отрисовка заняла четыре секунды
- пользователь увидел хоть что-то через две с половиной секунды

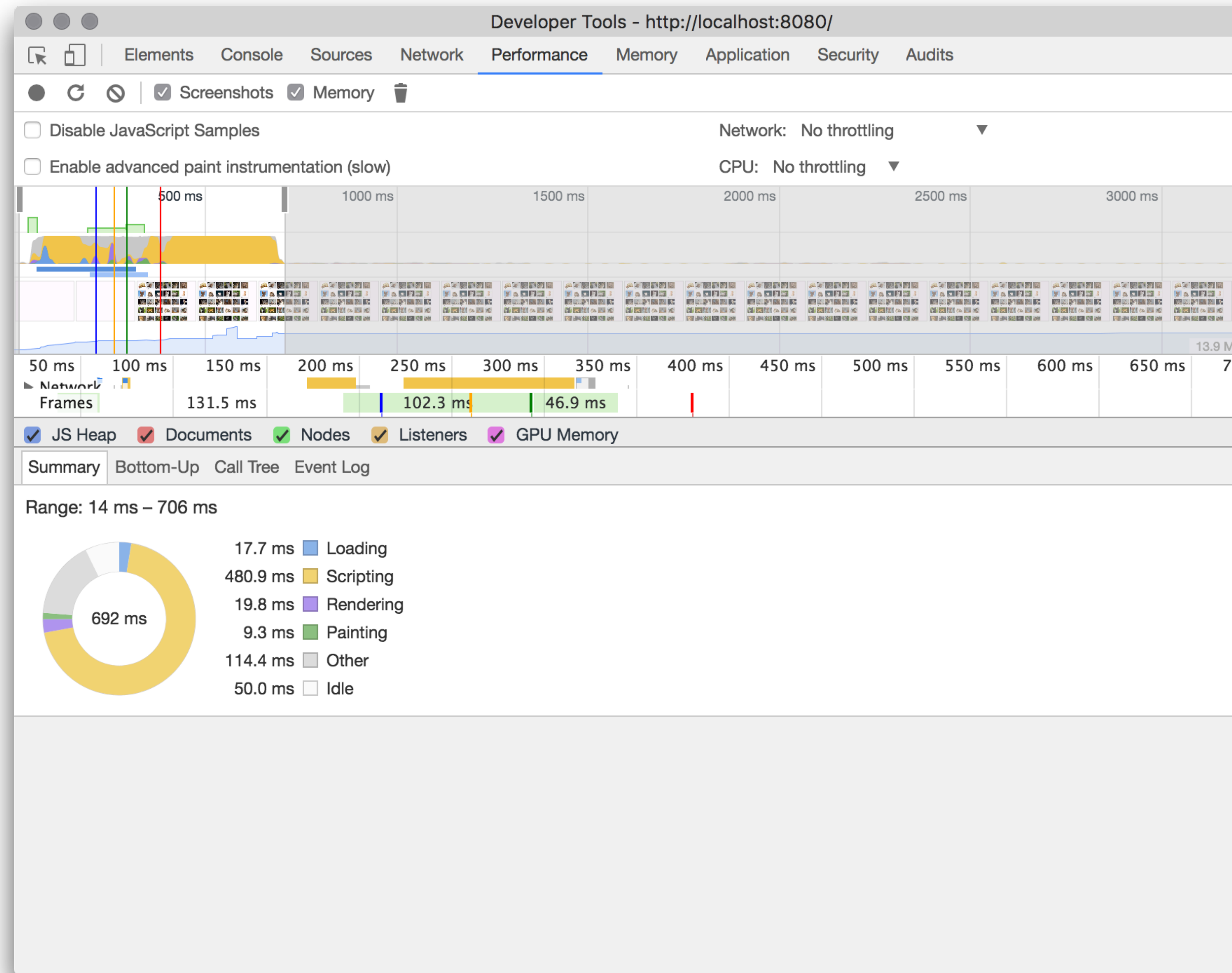


Первая страница на 5000 элементов

- полная отрисовка заняла четыре секунды
- пользователь увидел хоть что-то через две с половиной секунды
- пользователь увидел контент почти через три с половиной секунды

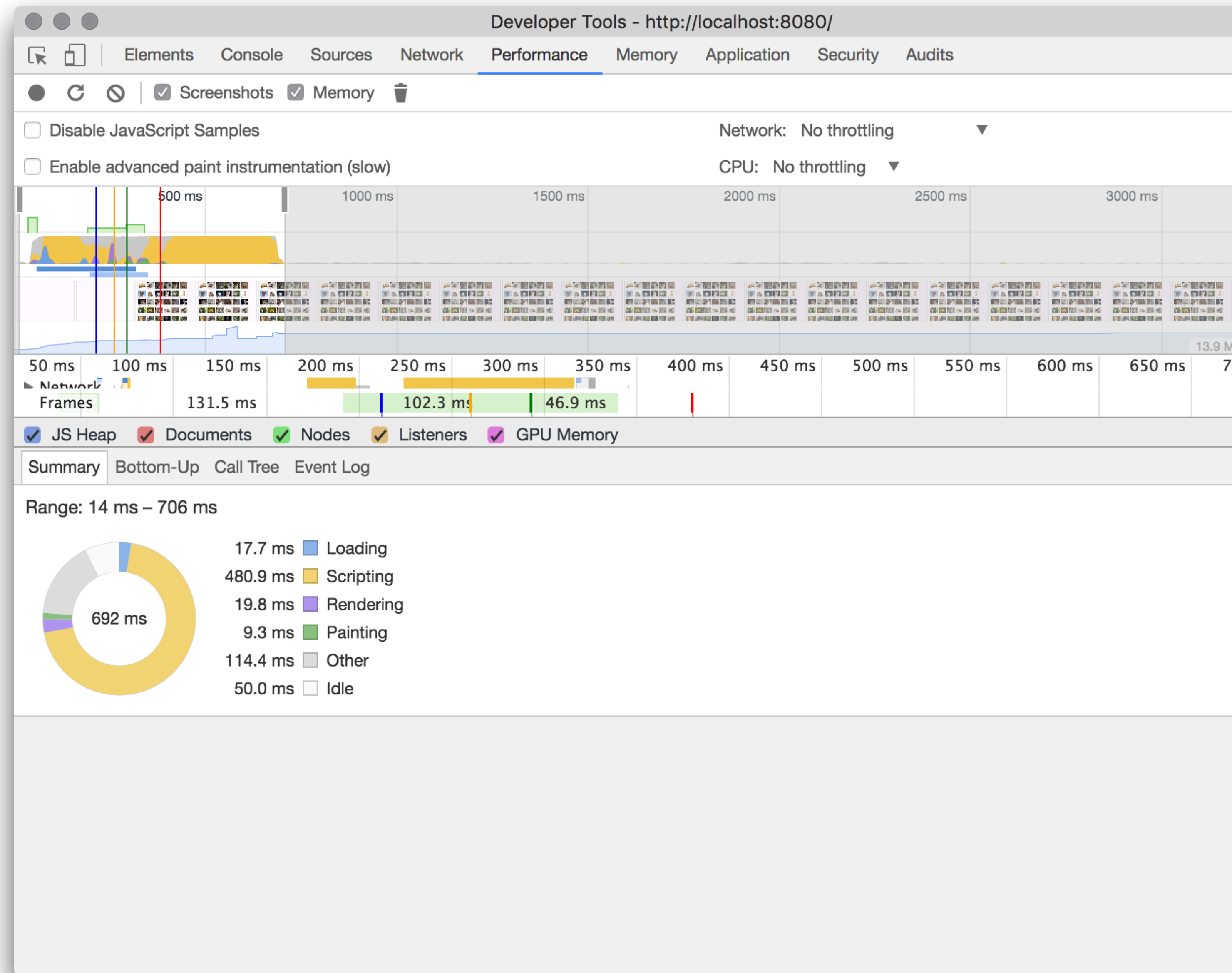


Первая страница на 100 элементов



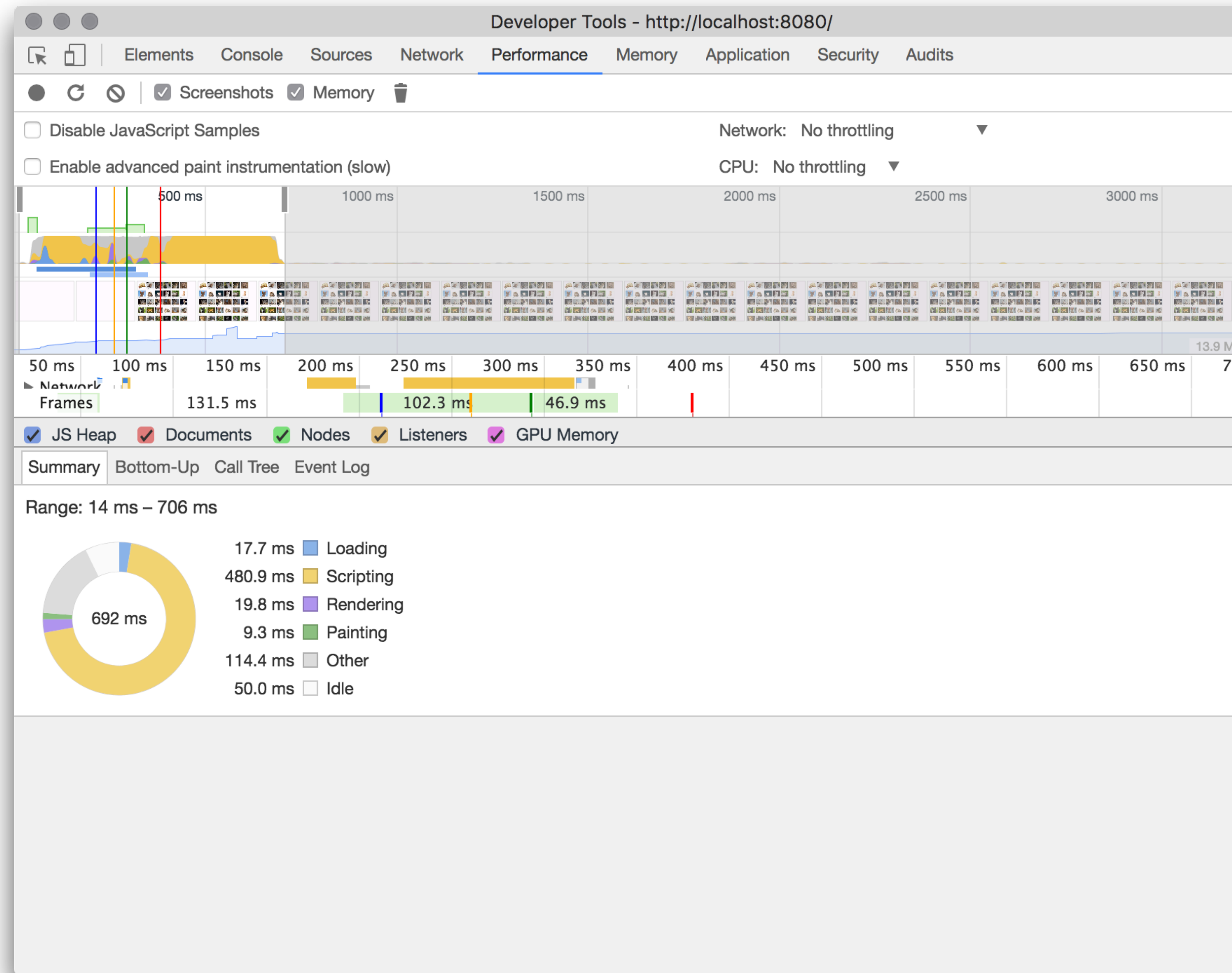
Первая страница на 100 элементов

- полная отрисовка произошла меньше чем за полсекунды



Первая страница на 100 элементов

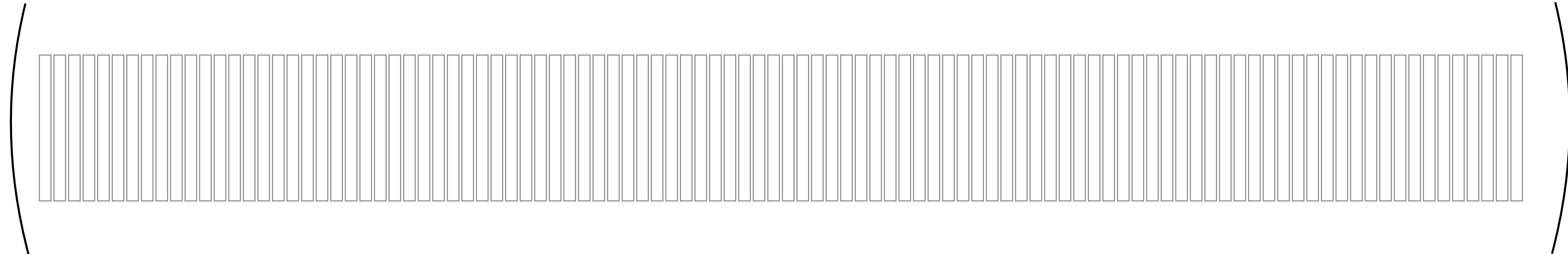
- полная отрисовка произошла меньше чем за полсекунды
- пользователь увидел результаты сразу как только появился контент, не было дополнительной загрузки, это заняло четверть секунды



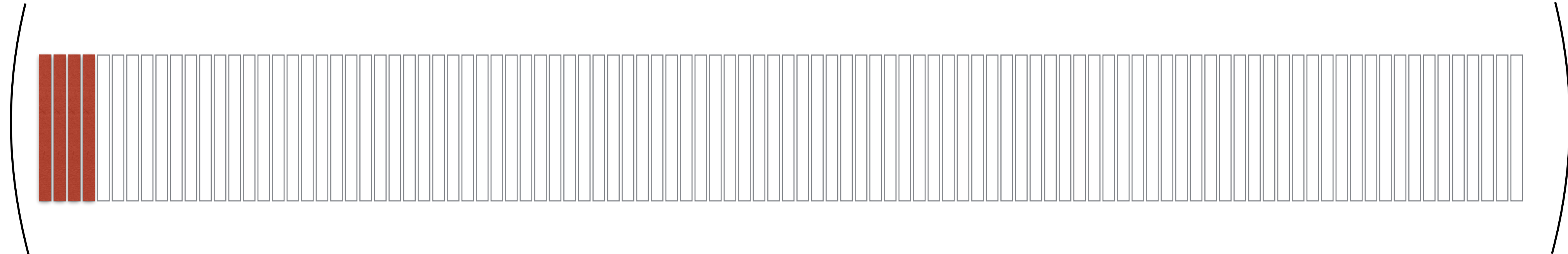
Тротлинг



Тротлинг (пропуск кадров)



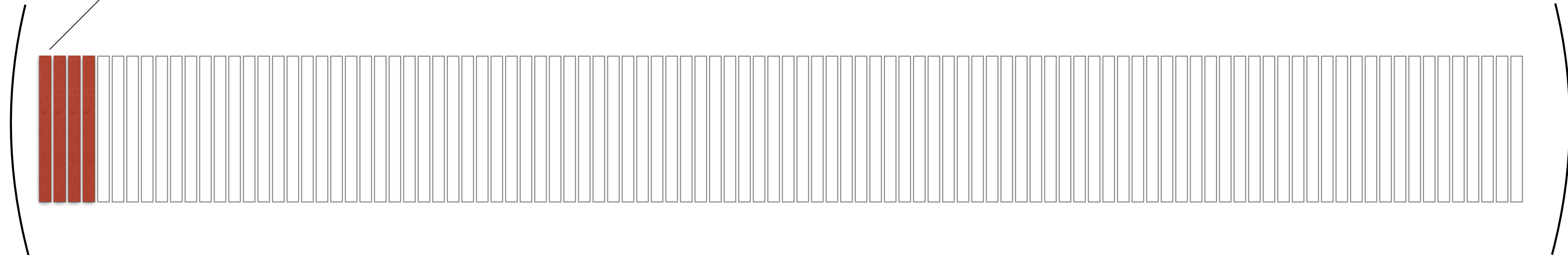
Тротлинг (пропуск кадров)



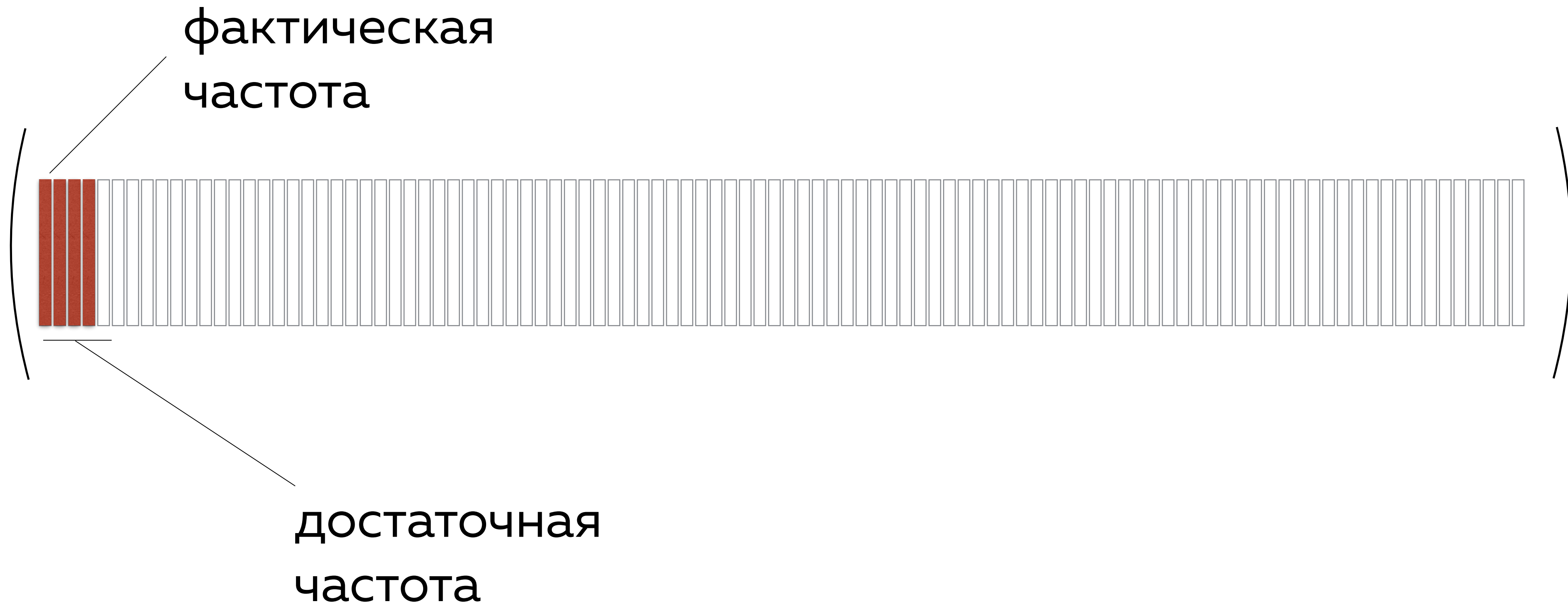
Тротлинг (пропуск кадров)



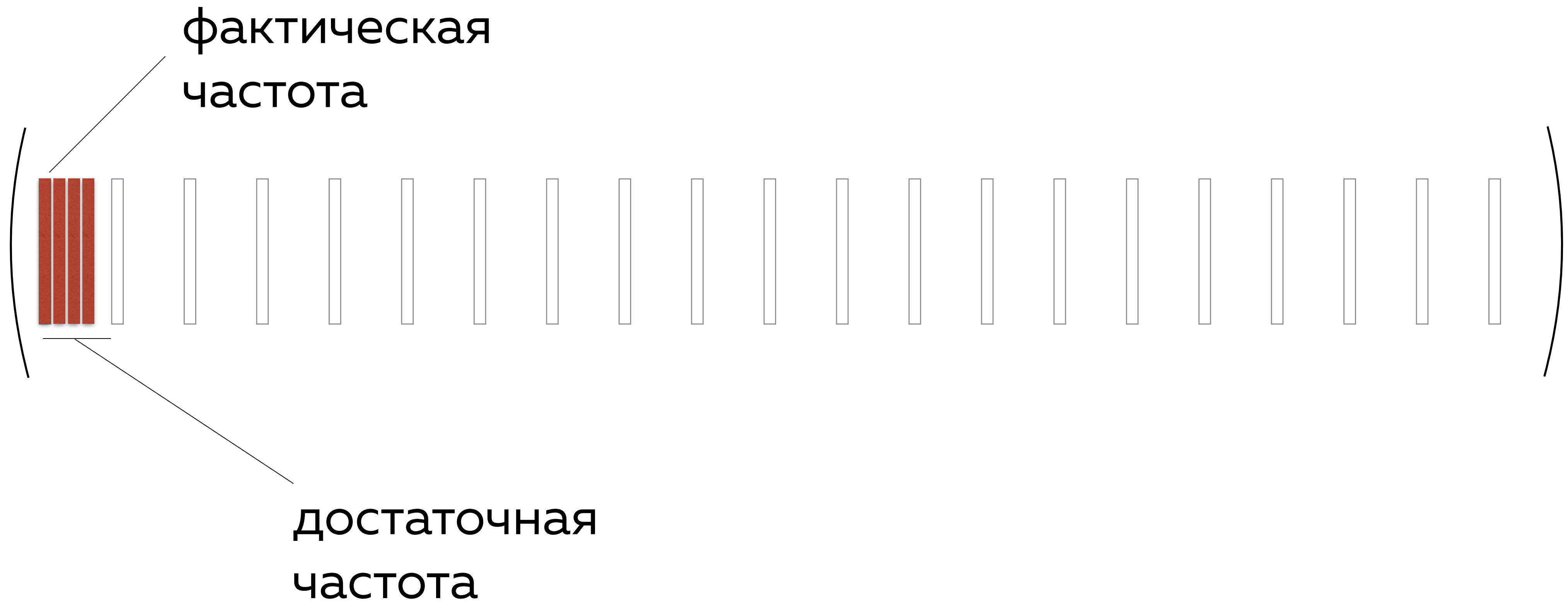
фактическая
частота



Тротлинг (пропуск кадров)



Тротлинг (пропуск кадров)



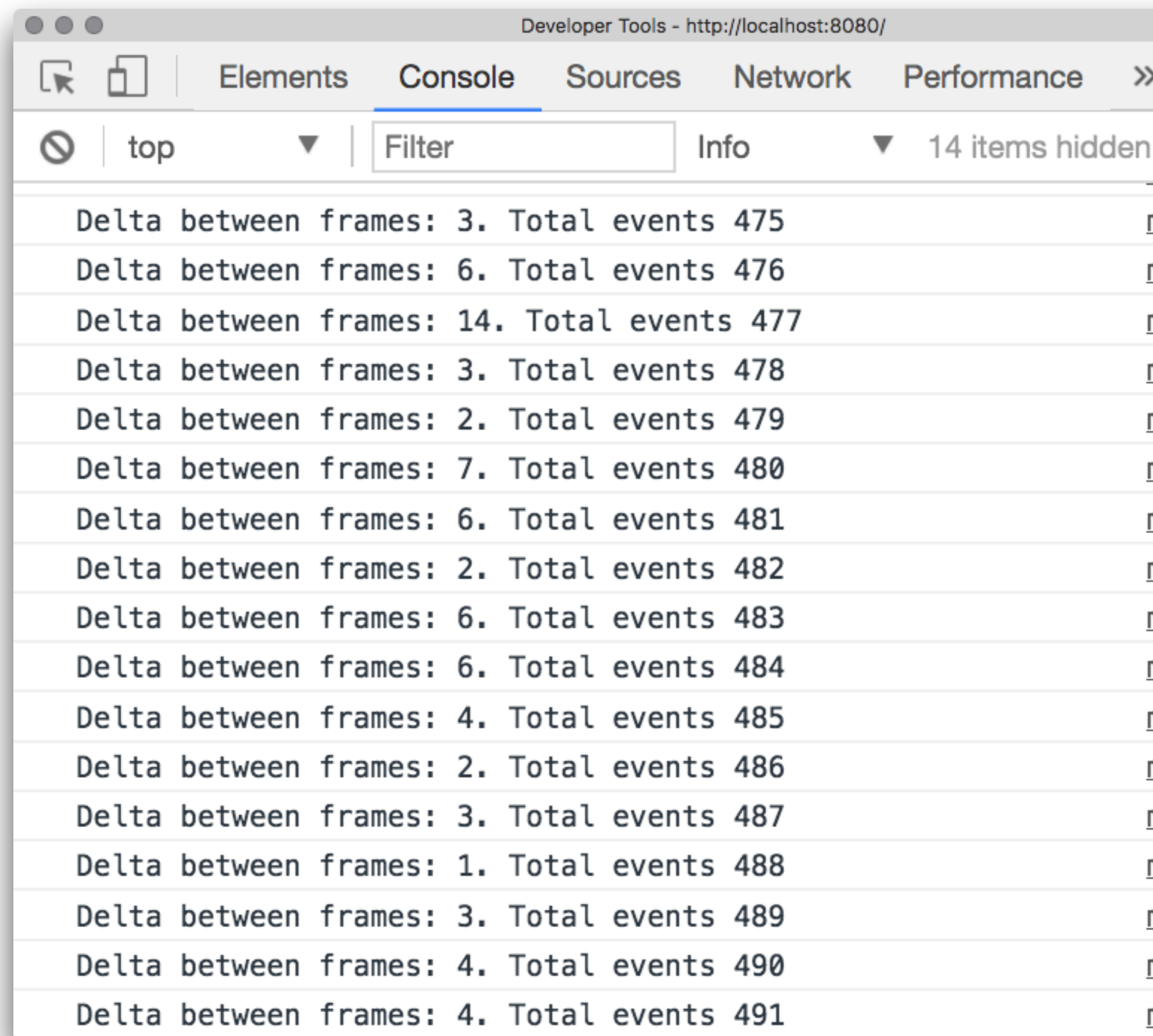
Проверка положения скролла

по каждому событию скролла проверяется, не находится ли пользователь в низу страницы (с небольшим запасом)

```
window.onscroll = () => {  
    if (document.body.scrollTop + window.innerHeight >= document.body.scrollHeight - S  
        switchPage(pageSwitcher, PAGE_S1)  
    }  
};
```

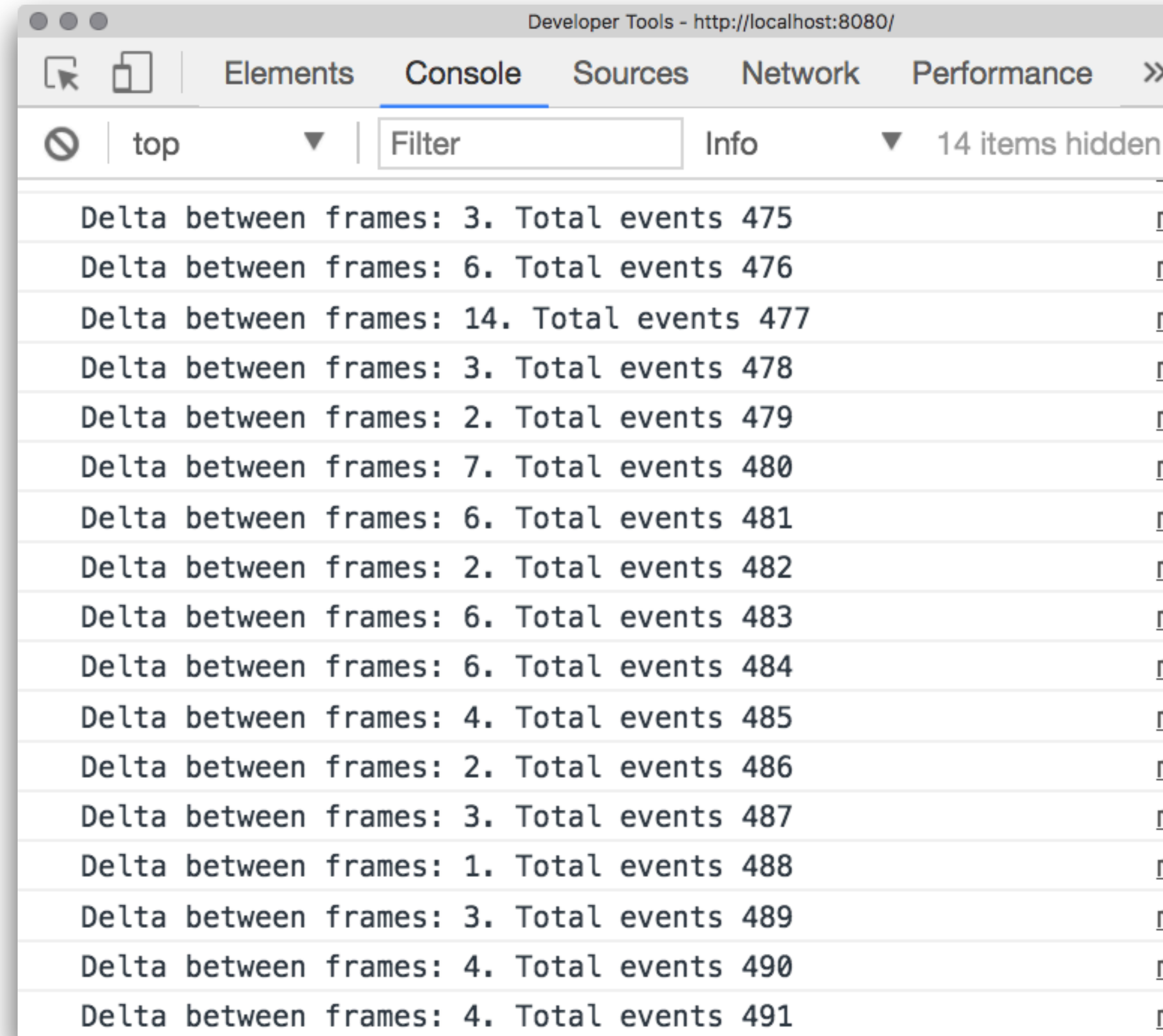


Прокрутка 100 фото



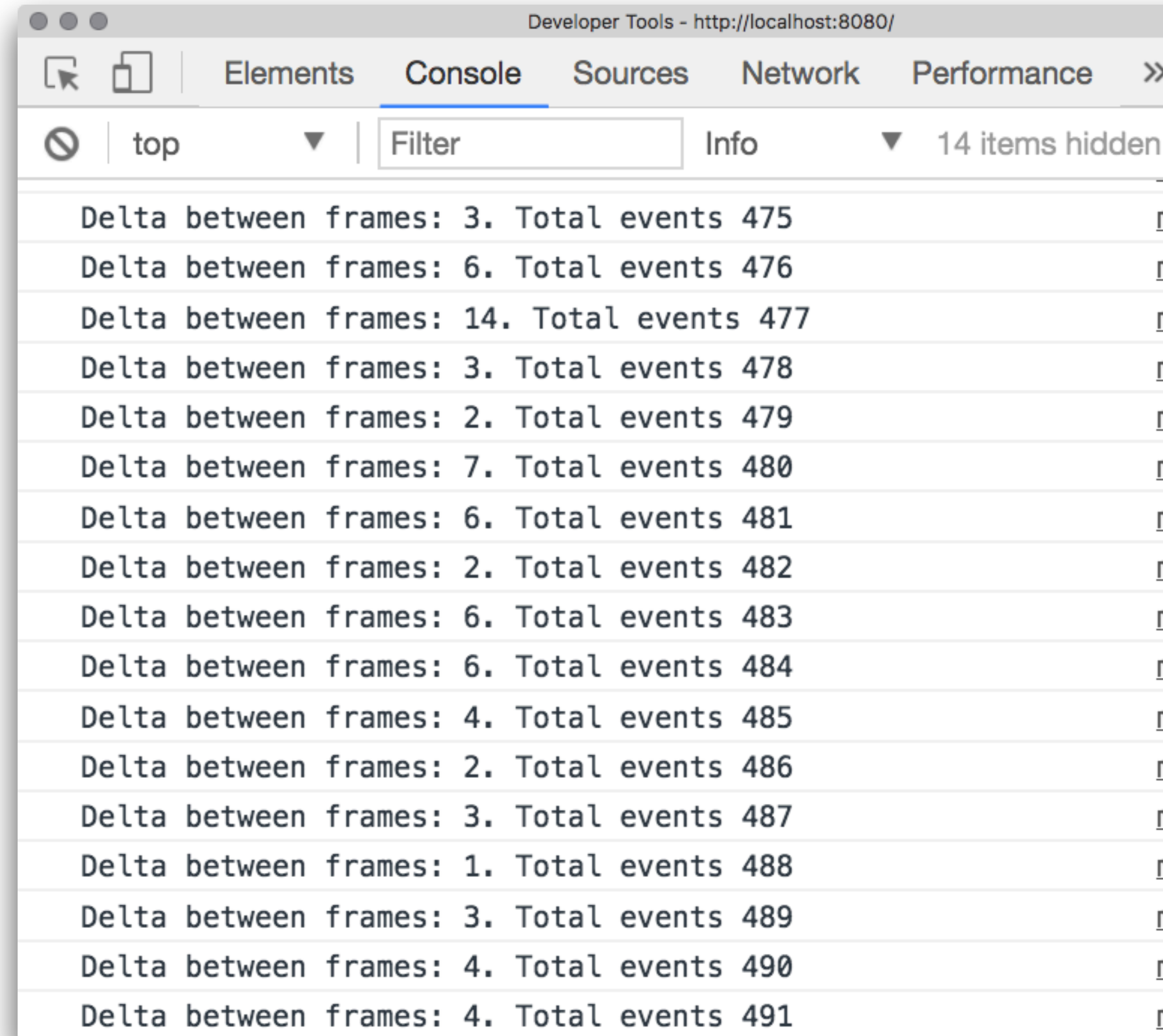
Прокрутка 100 фото

- событие скролла каждые **4 пикселя** при высоте страницы в 1000



Прокрутка 100 фото

- событие скролла каждые **4 пикселя** при высоте страницы в 1000
- от верха до низа — почти **500 проверок** где находится пользователь



Оптимизированная проверка

```
let prevComparison = performance.now()

window.onscroll = () => {
  const now = performance.now();
  if (now - prevComparison >= 100) {
    if (document.body.scrollTop + window.innerHeight >=
        document.body.scrollHeight -
        document.body.clientHeight) {
      switchPage(pageSwitcher, PAGE_);
    }
    prevComparison = now;
  }
};
```



Оптимизированная проверка

- событие скролла происходит с прежней частотой

```
let prevComparison = performance.now()

window.onscroll = () => {
  const now = performance.now();
  if (now - prevComparison >= 100) {
    if (document.body.scrollTop + window.innerHeight >=
        document.body.scrollHeight -
        window.innerHeight) {
      switchPage(pageSwitcher, PAGE_1);
    }
    prevComparison = now;
  }
};
```



Оптимизированная проверка

- событие скролла происходит с прежней частотой
- дополнительная нагрузка на скролл в виде сложных вычислений производится не чаще, чем раз в 100 мсек

```
let prevComparison = performance.now
```

```
window.onscroll = () => {
```

```
  const now = performance.now();
```

```
  if (now - prevComparison >= 100) {
```

```
    if (document.body.scrollTop + wi
```

```
        document.body.scrollHeight -
```

```
        switchPage(pageSwitcher, PAGE_
```

```
    }
```

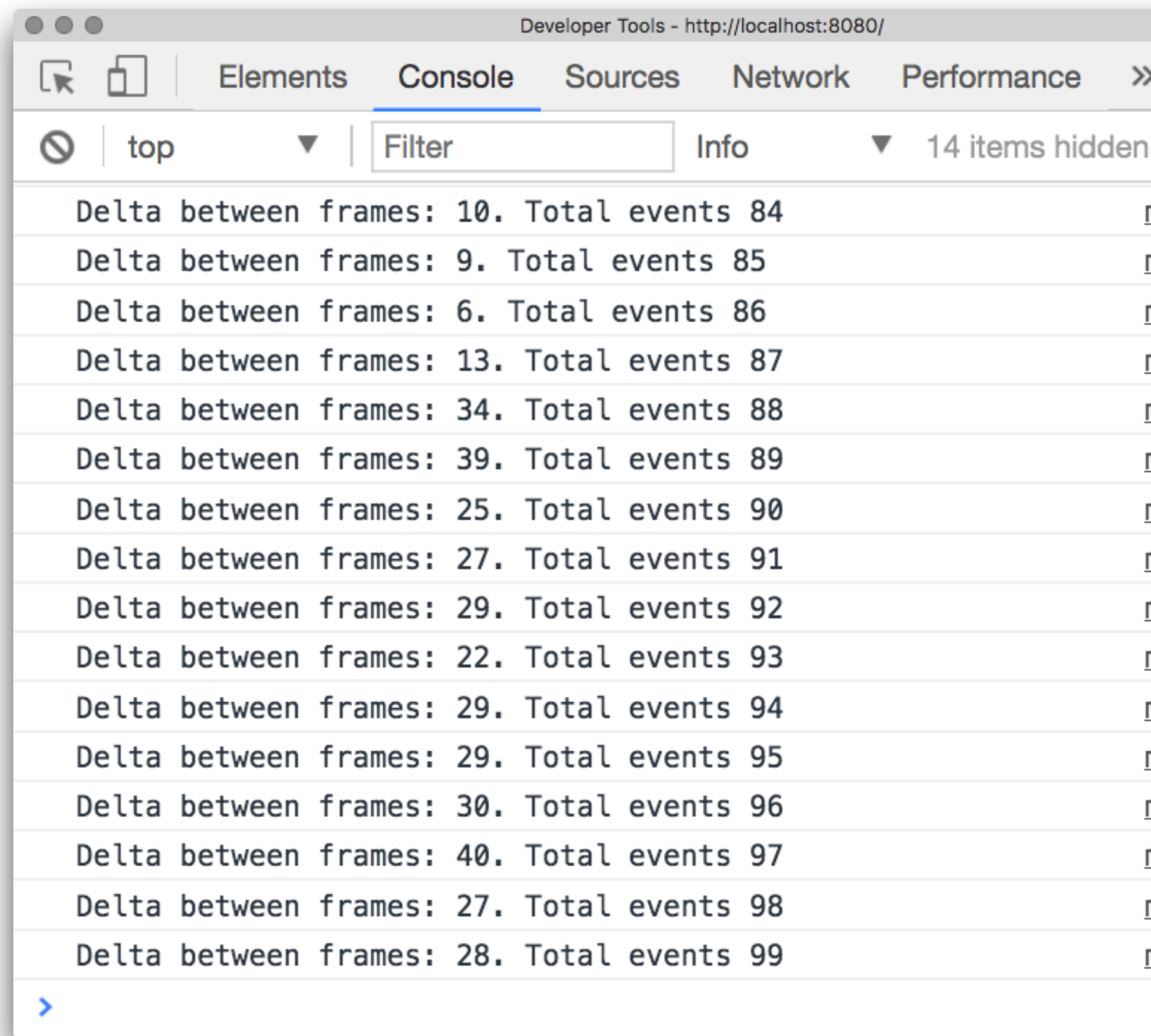
```
    prevComparison = now;
```

```
  }
```

```
};
```

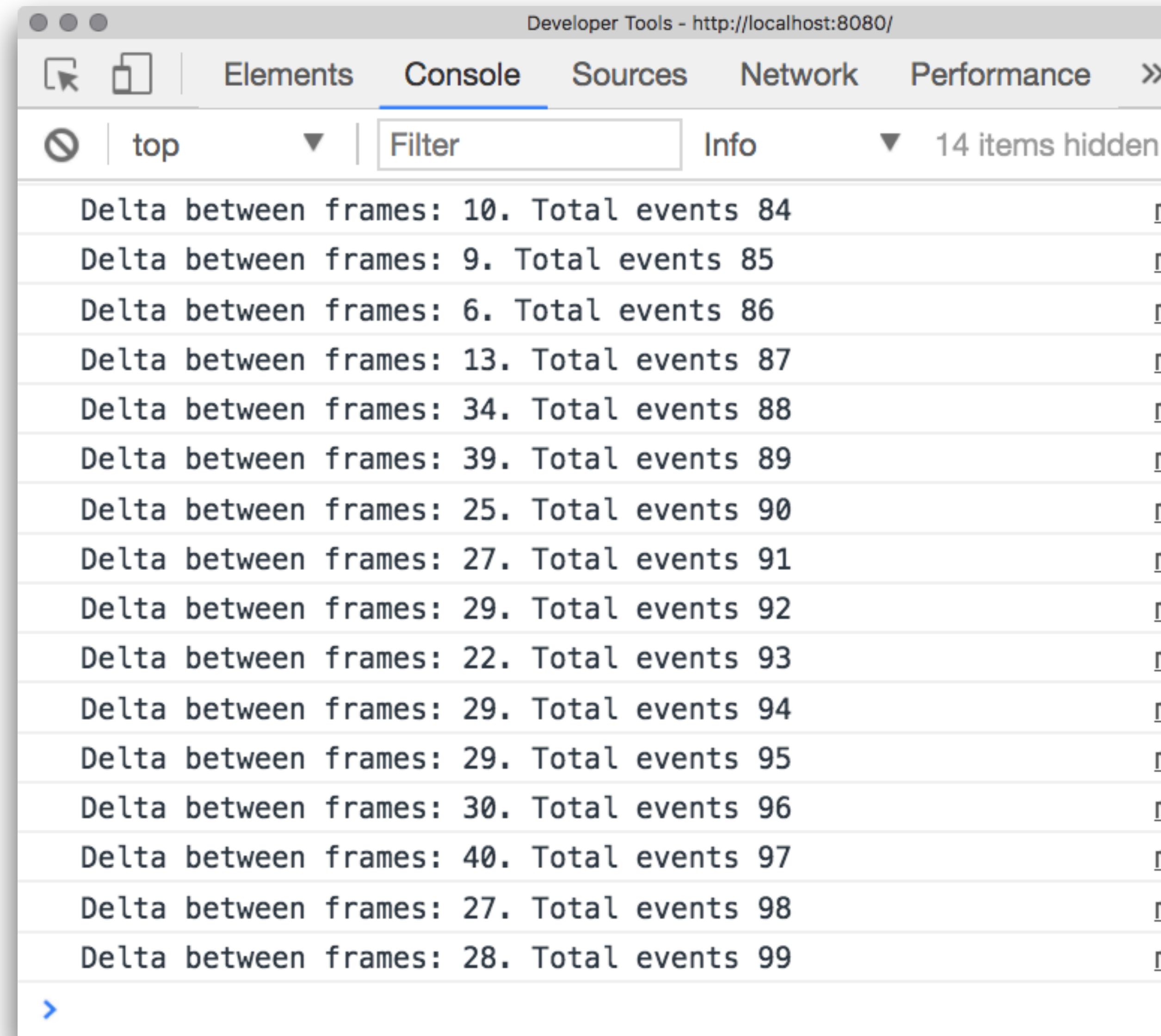


Прокрутка 100 фоток – тротл 100 мсек



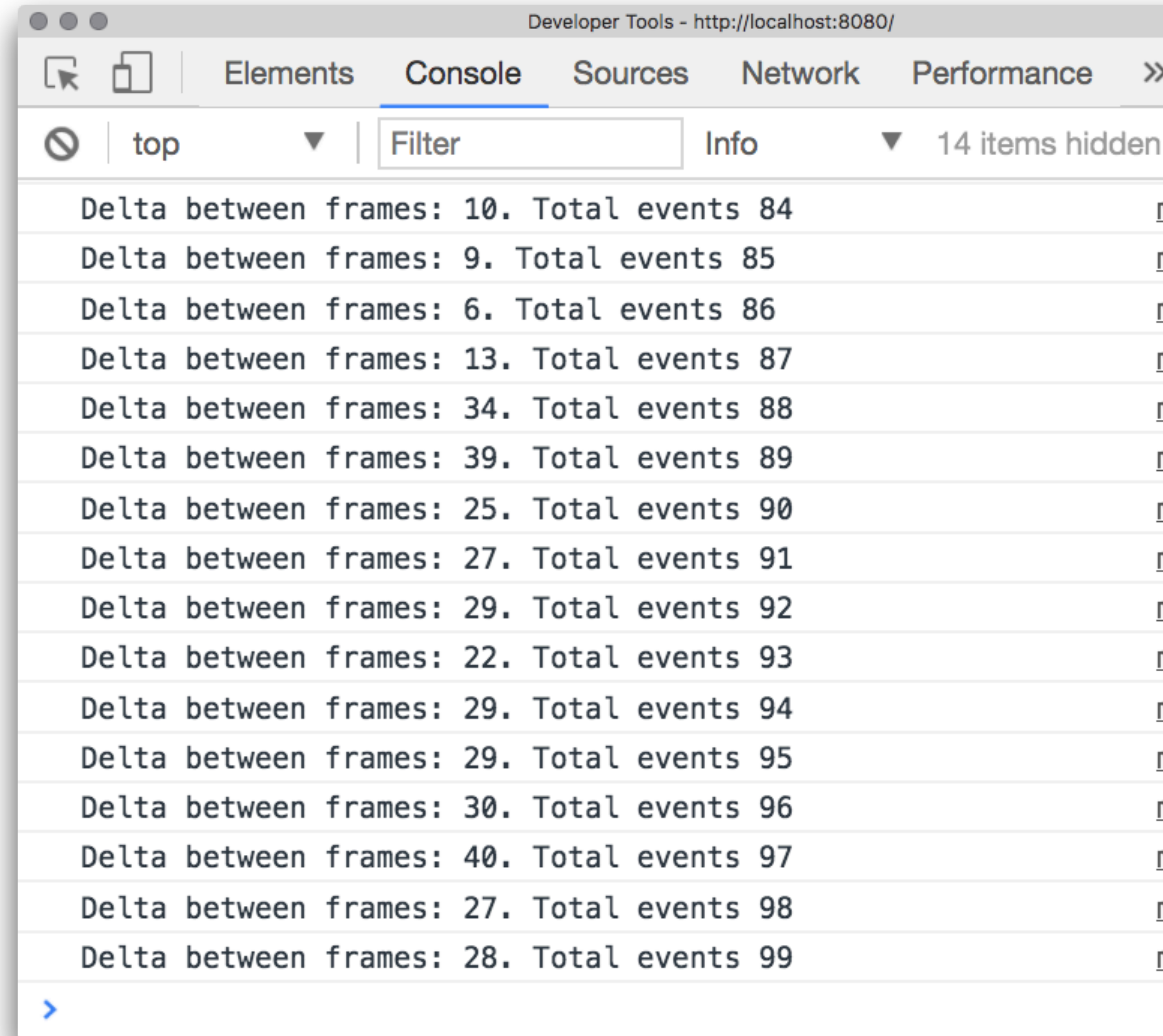
Прокрутка 100 фоток – тротл 100 мсек

- событие скролла каждые **20-30 пикселей** при высоте страницы в 1000



Прокрутка 100 фоток – тротл 100 мсек

- событие скролла каждые **20-30 пикселей** при высоте страницы в 1000
- **100 проверок** где находится пользователь



Отдать вычисления



Отдать вычисления



Отдать вычисления

- на видеокарту (*canvas, will-change*)



Отдать вычисления

- на видеокарту (*canvas, will-change*)
- на сервер



Отдать вычисления

- на видеокарту (*canvas, will-change*)
- на сервер
- в другой поток (*ServiceWorker*)



Отдать на видеокарту

почему браузерные игры делают не на SVG



SVG

canvas



SVG

- все элементы хранятся в памяти, даже незначительные

canvas



SVG

- все элементы хранятся в памяти, даже незначительные

canvas

- все отрисовывается сразу и все объекты превращаются в набор пикселей (*не используется память для долгосрочного хранения элементов*)



SVG

- все элементы хранятся в памяти, даже незначительные

canvas

- все отрисовывается сразу и все объекты превращаются в набор пикселей (*не используется память для долгосрочного хранения элементов*)
- для хранения данных нужно использовать отдельные структуры, которые можно оптимизировать под задачу



SVG

- все элементы хранятся в памяти, даже незначительные
- с каждым элементом можно взаимодействовать: изменять после отрисовки, анимировать, описывать пользовательское взаимодействие, обновлять по одиночке

canvas

- все отрисовывается сразу и все объекты превращаются в набор пикселей (*не используется память для долгосрочного хранения элементов*)
- для хранения данных нужно использовать отдельные структуры, которые можно оптимизировать под задачу



SVG

- все элементы хранятся в памяти, даже незначительные
- с каждым элементом можно взаимодействовать: изменять после отрисовки, анимировать, описывать пользовательское взаимодействие, обновлять по одиночке

canvas

- все отрисовывается сразу и все объекты превращаются в набор пикселей (*не используется память для долгосрочного хранения элементов*)
- для хранения данных нужно использовать отдельные структуры, которые можно оптимизировать под задачу
- низкоуровневое API: из коробки нет стилизации, взаимодействия с пользователем, анимации, работы с отдельными элементами



SVG

- все элементы хранятся в памяти, даже незначительные
- с каждым элементом можно взаимодействовать: изменять после отрисовки, анимировать, описывать пользовательское взаимодействие, обновлять по одиночке
- видеокарта используется не на полную: параметры элементы вычисляются на процессоре и только потом отрисовываются через видеокарту

canvas

- все отрисовывается сразу и все объекты превращаются в набор пикселей (*не используется память для долгосрочного хранения элементов*)
- для хранения данных нужно использовать отдельные структуры, которые можно оптимизировать под задачу
- низкоуровневое API: из коробки нет стилизации, взаимодействия с пользователем, анимации, работы с отдельными элементами



SVG

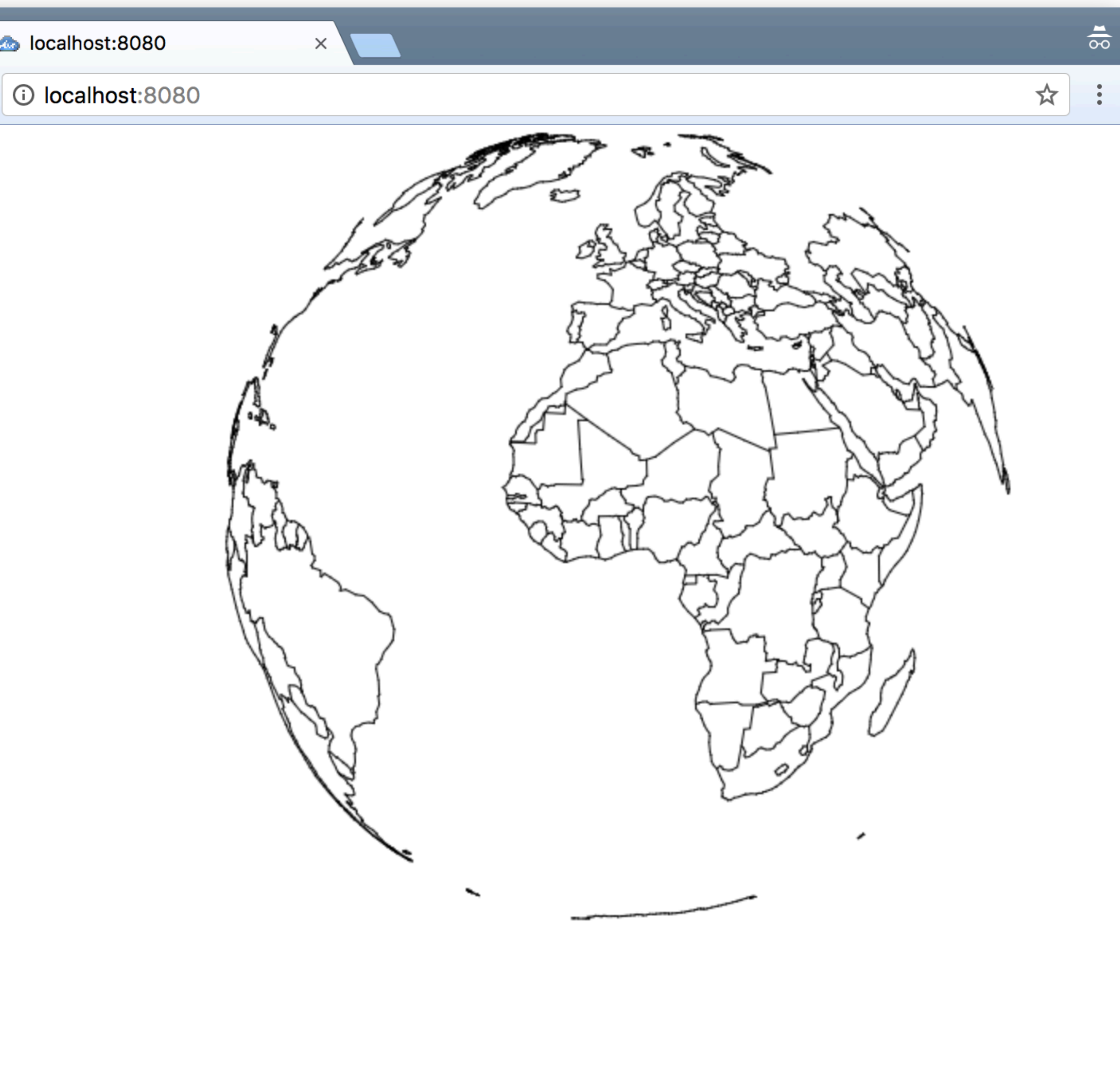
- все элементы хранятся в памяти, даже незначительные
- с каждым элементом можно взаимодействовать: изменять после отрисовки, анимировать, описывать пользовательское взаимодействие, обновлять по одиночке
- видеокарта используется не на полную: параметры элементы высчитываются на процессоре и только потом отрисовываются через видеокарту

canvas

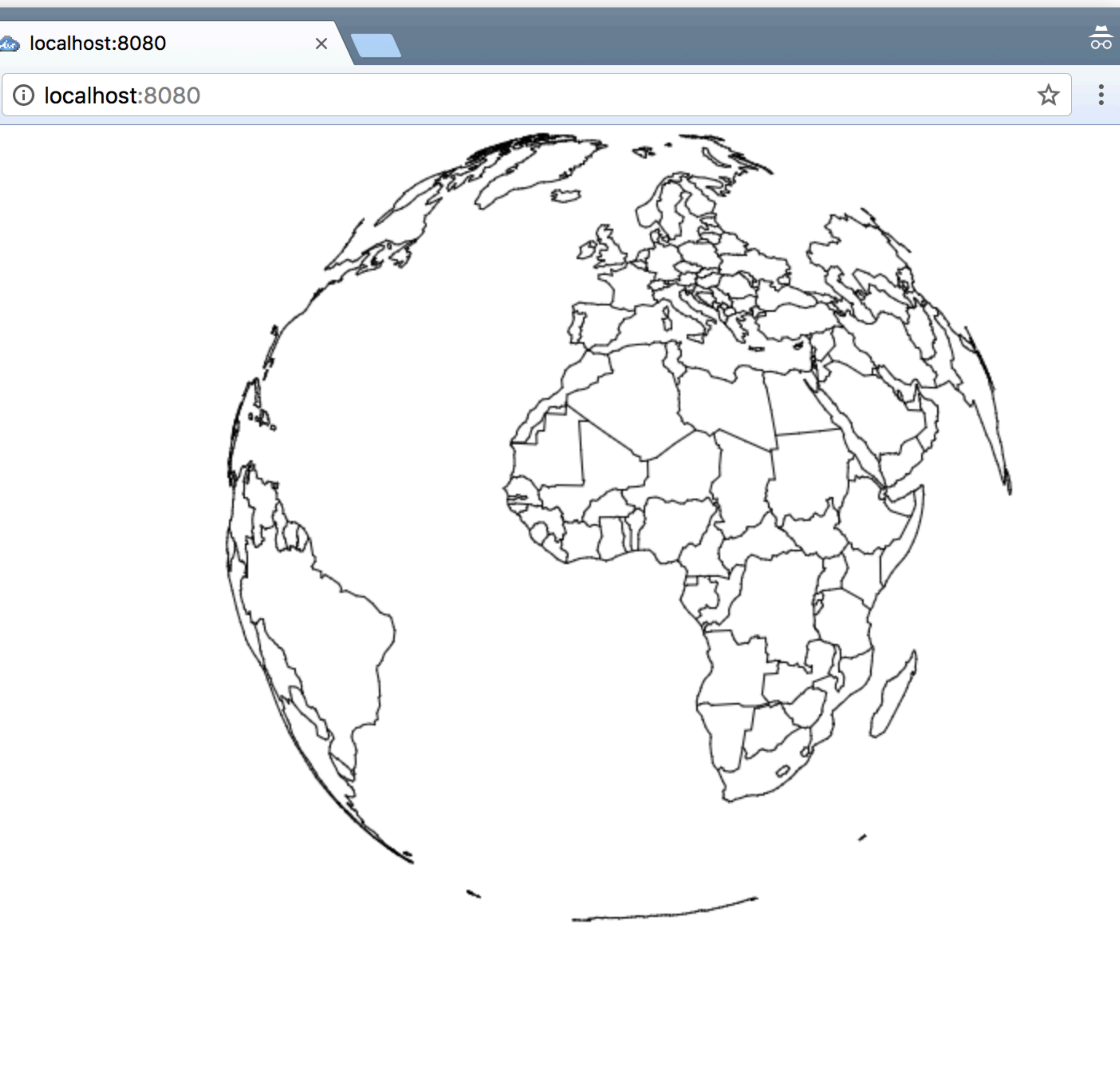
- все отрисовывается сразу и все объекты превращаются в набор пикселей (*не используется память для долгосрочного хранения элементов*)
- для хранения данных нужно использовать отдельные структуры, которые можно оптимизировать под задачу
- низкоуровневое API: из коробки нет стилизации, взаимодействия с пользователем, анимации, работы с отдельными элементами
- вся отрисовка происходит на видеокарте



Покрутим земной шар



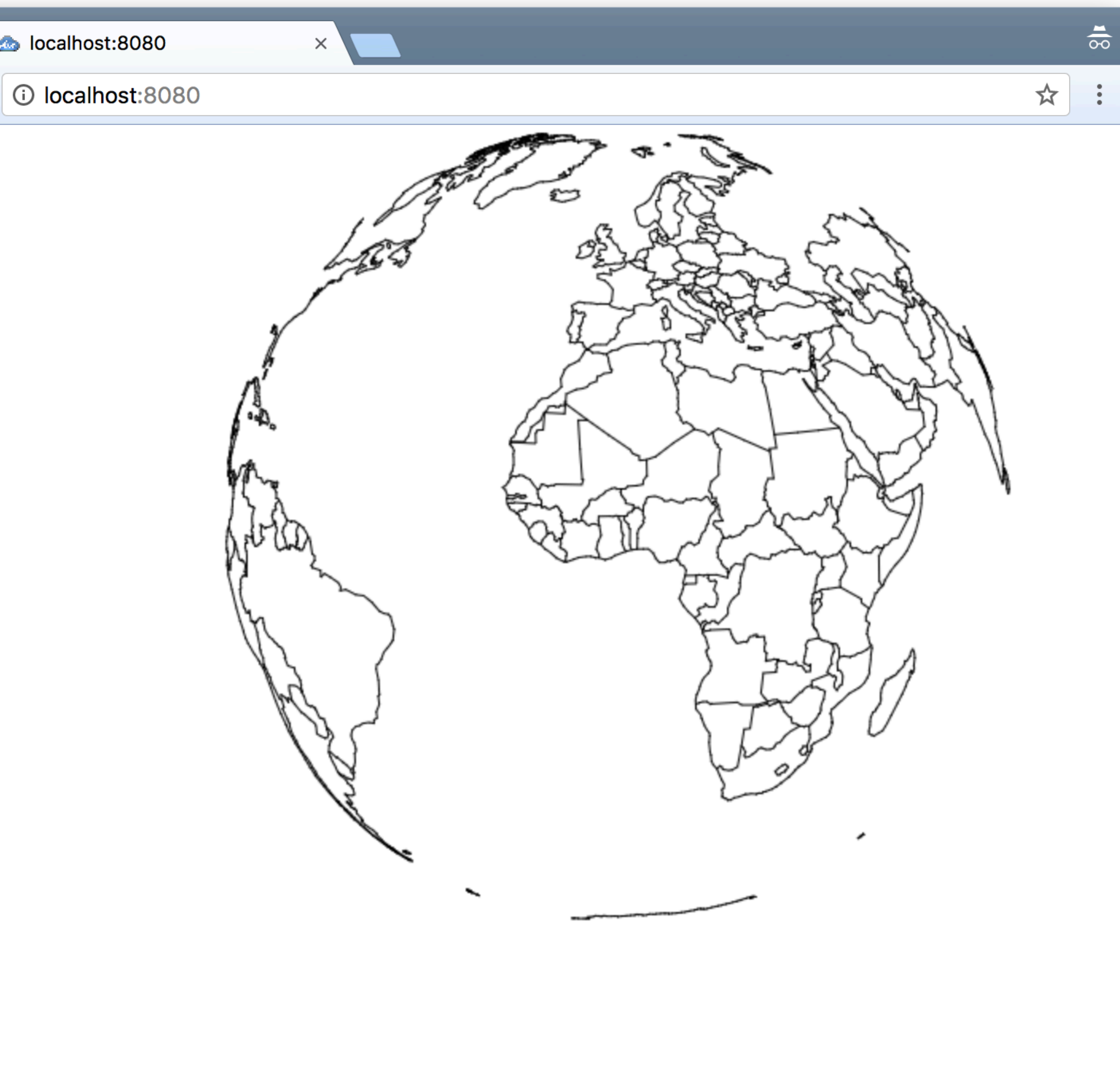
Покрутим земной шар



- 800×600



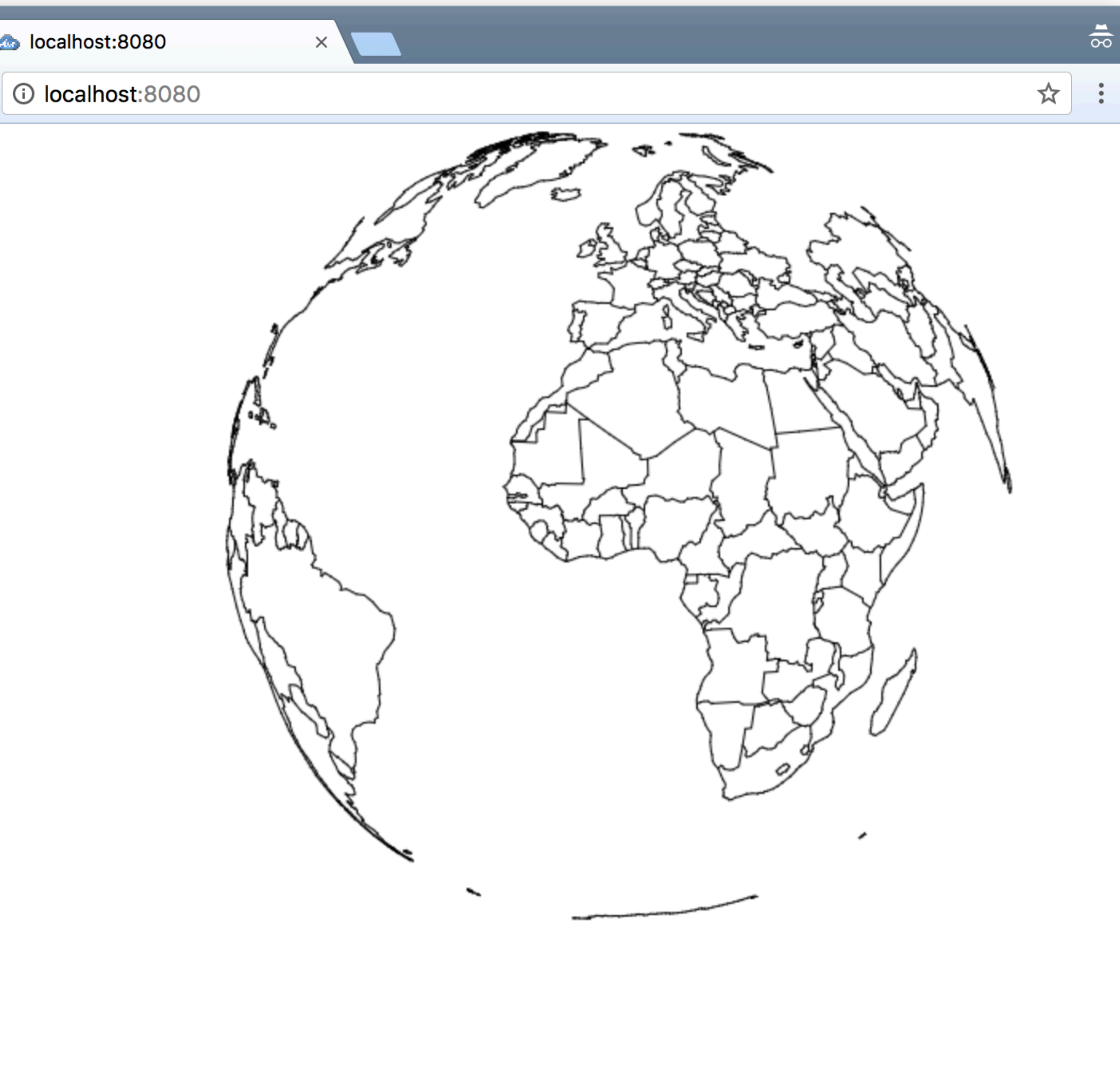
Покрутим земной шар



- 800×600
- все страны описаны через один контур (*path*)



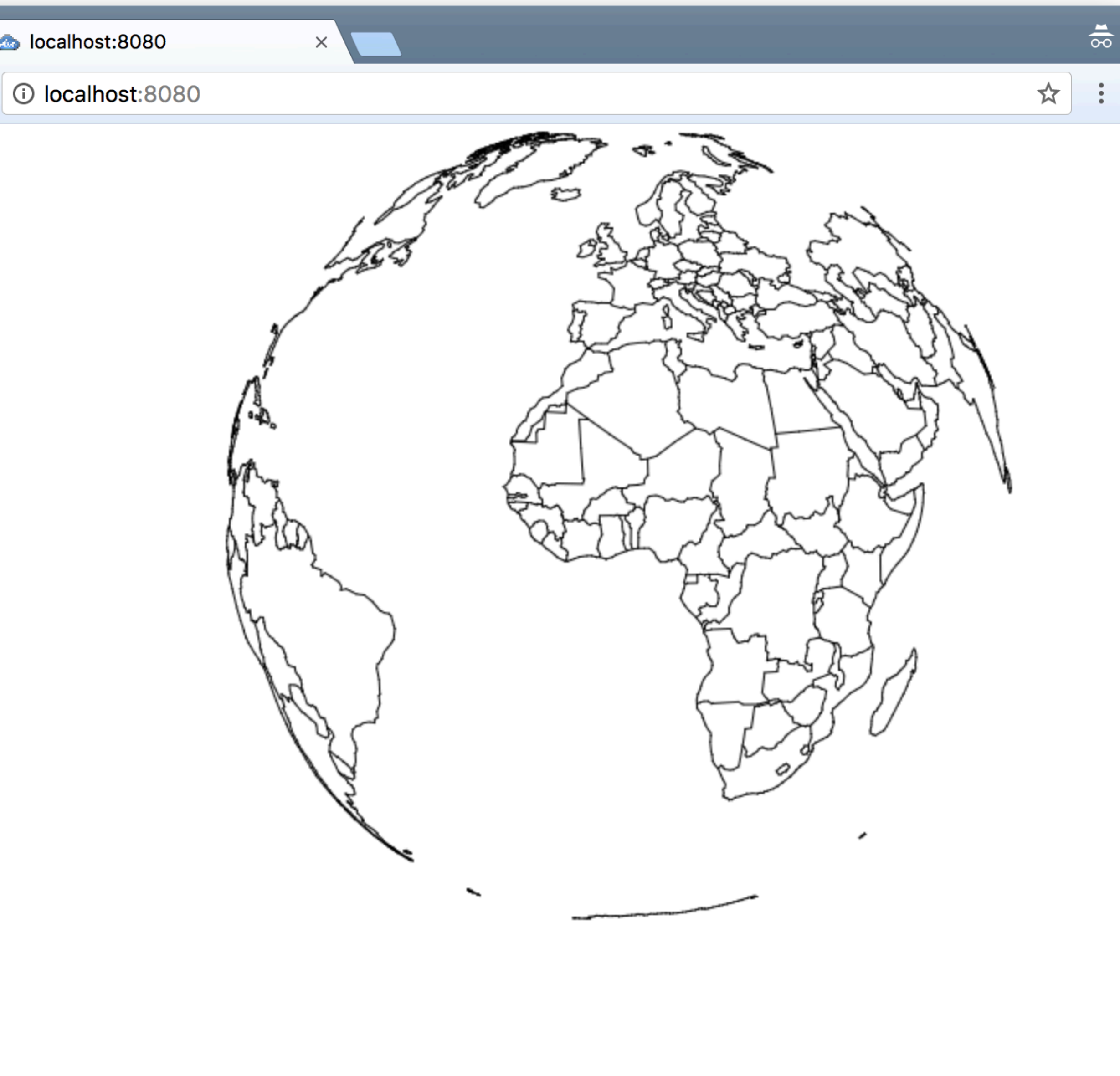
Покрутим земной шар



- 800×600
- все страны описаны через один контур (*path*)
- частоту кадров определяет сам браузер (*requestAnimationFrame*)



Покрутим земной шар

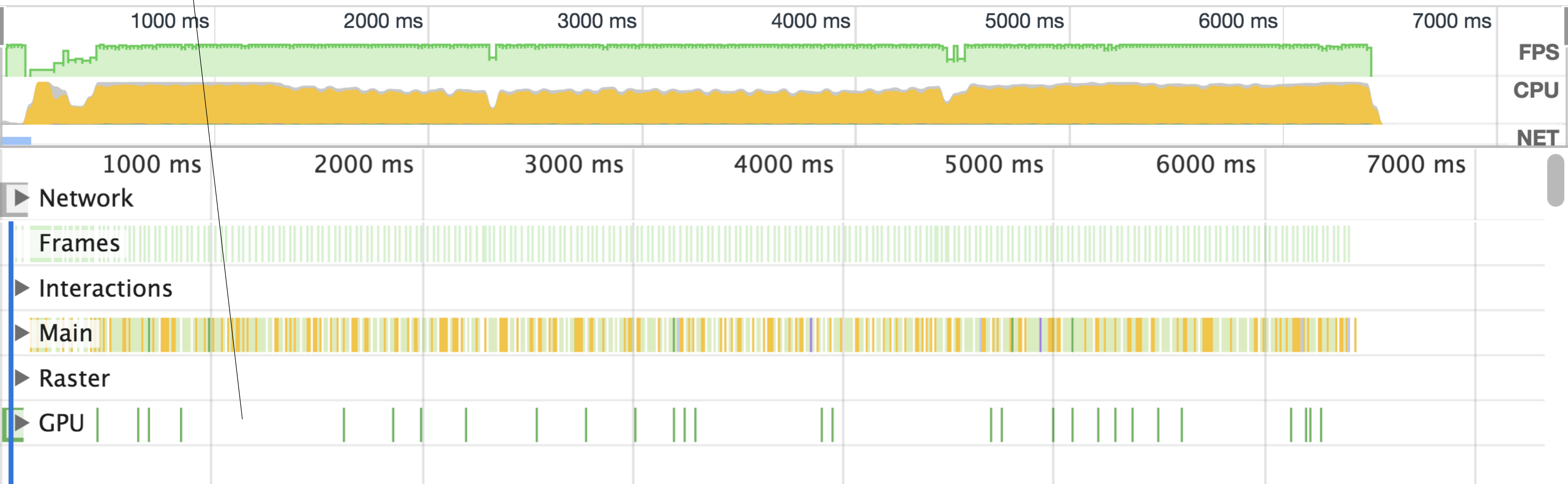


- 800×600
- все страны описаны через один контур (*path*)
- частоту кадров определяет сам браузер (*requestAnimationFrame*)
- один оборот на 360° по градусу за кадр



canvas

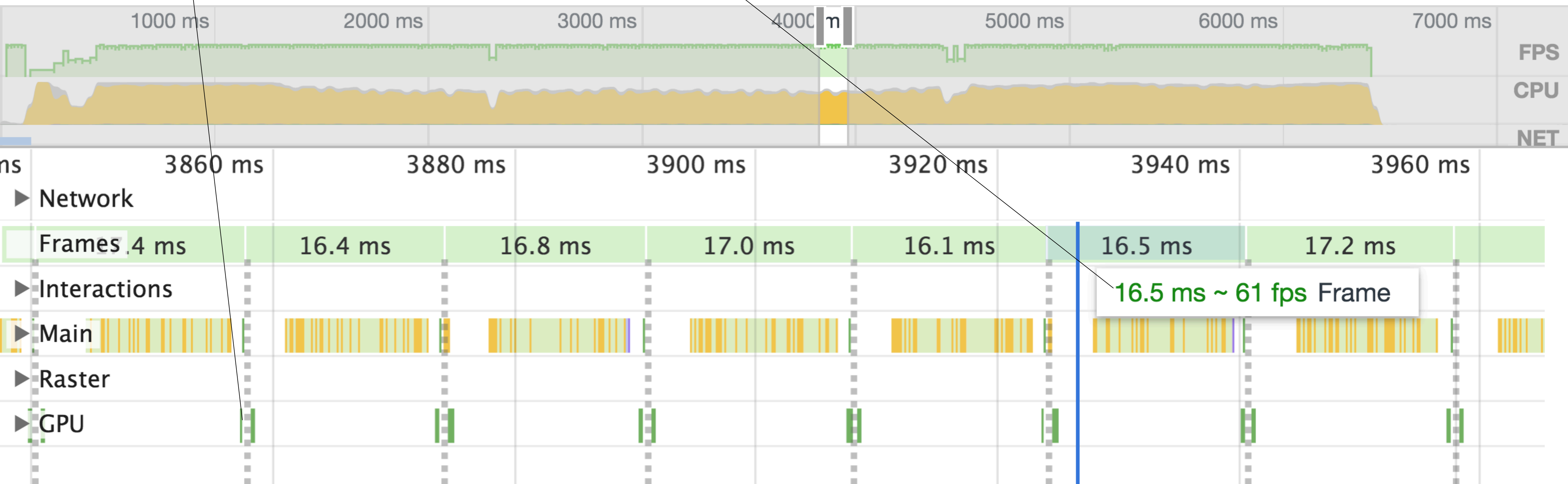
для отрисовки
используется
только GPU



canvas

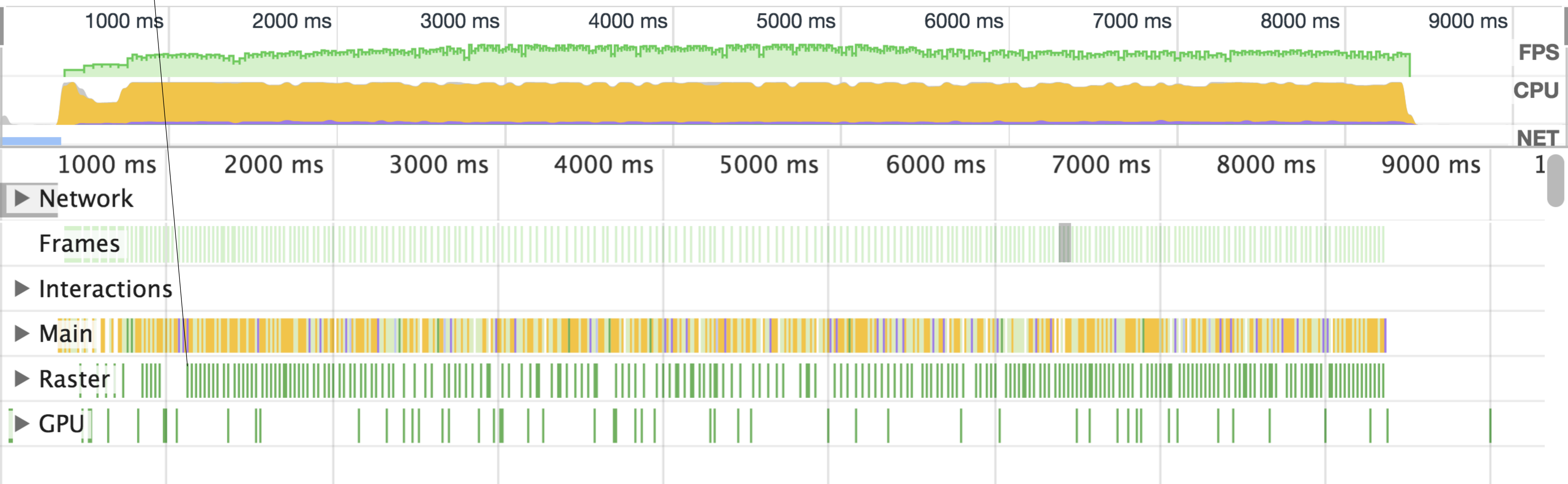
для отрисовки
используется
только GPU

в среднем время
выполнения 6–6,5 сек
($360 \text{ frames} / \sim 6 \text{ sec} = \sim 60 \text{ FPS}$)



SVG

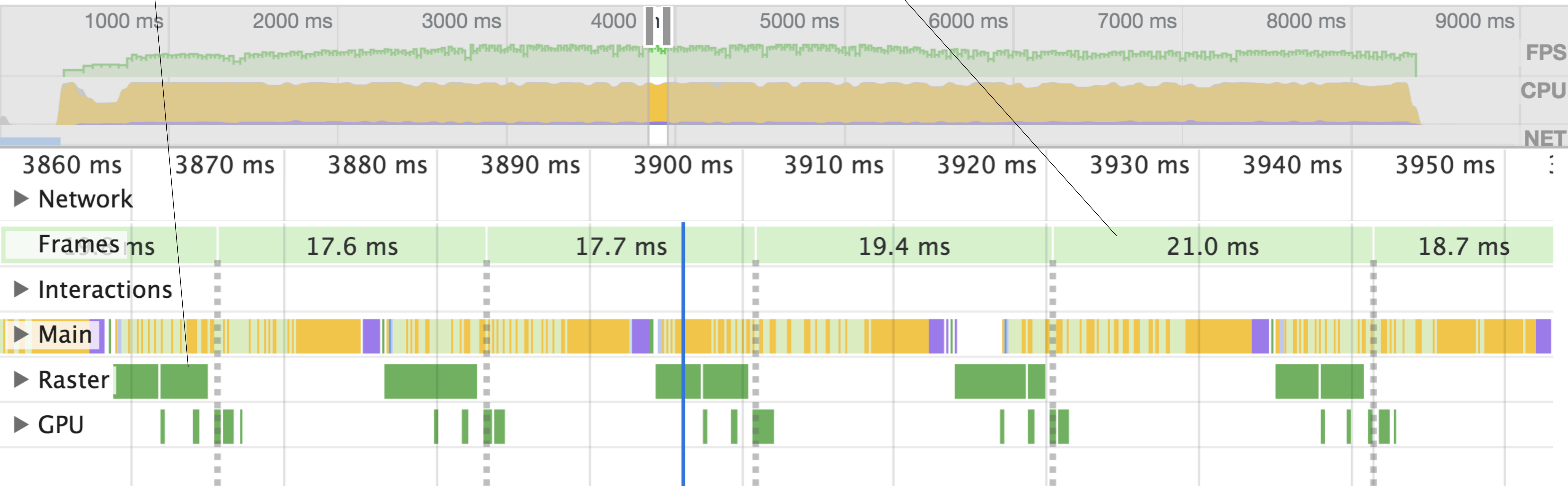
сначала параметры нод
рассчитываются процессором
как узлы DOM-дерева



SVG

сначала параметры нод
рассчитываются процессором
как узлы DOM-дерева

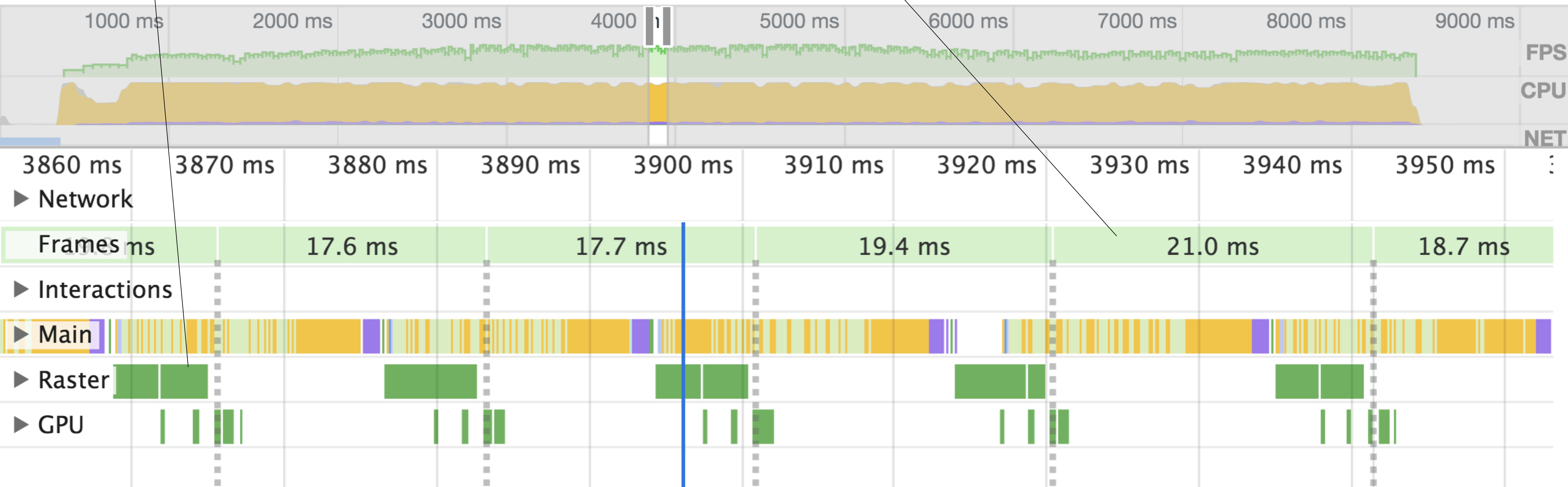
в среднем время
выполнения 8 сек
($360 \text{ frames} / \sim 8 \text{ sec} = \sim 45 \text{ FPS}$)

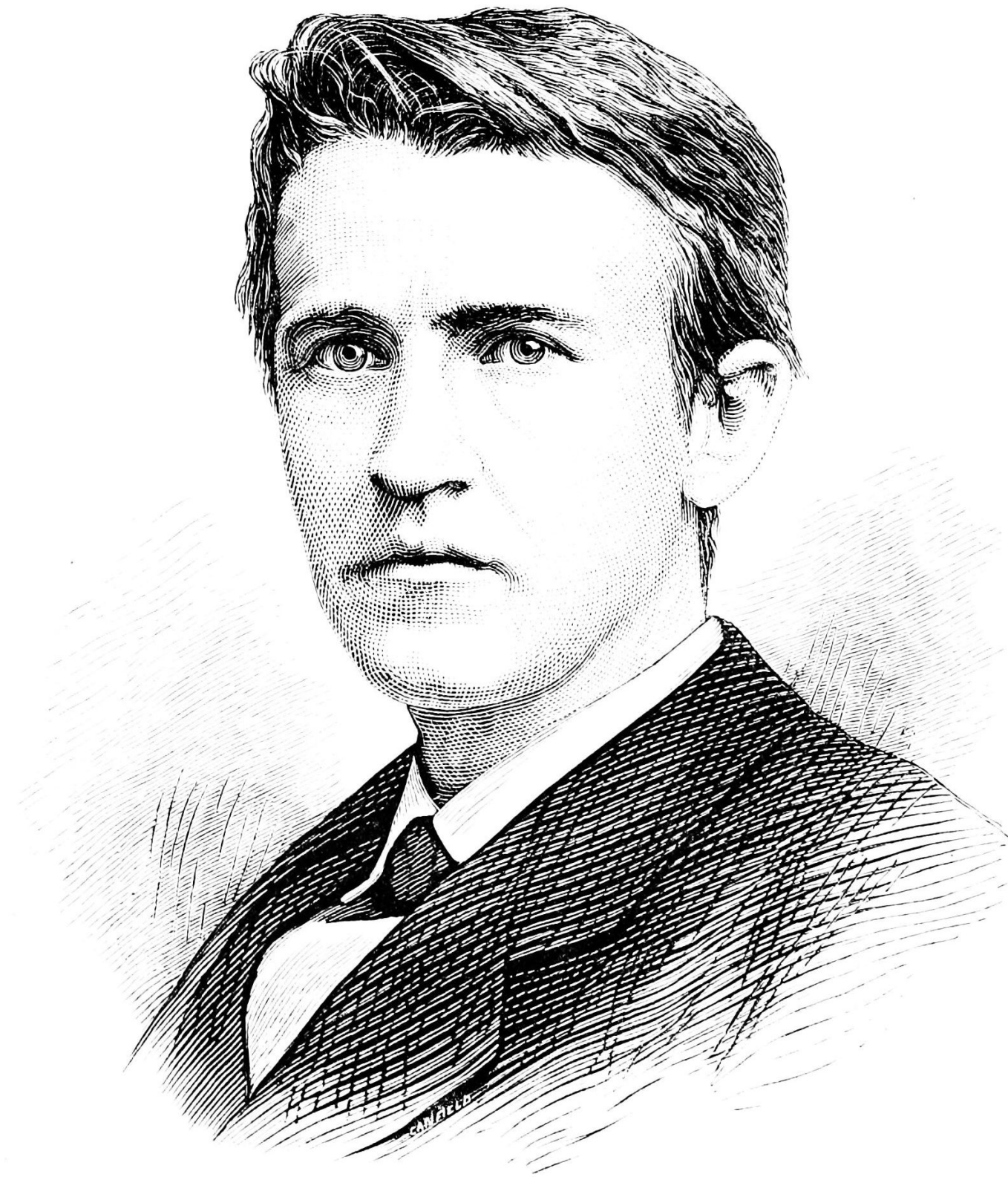


SVG

сначала параметры нод
рассчитываются процессором
как узлы DOM-дерева

в среднем время
выполнения 8 сек
($360 \text{ frames} / \sim 8 \text{ sec} = \sim 45 \text{ FPS}$)



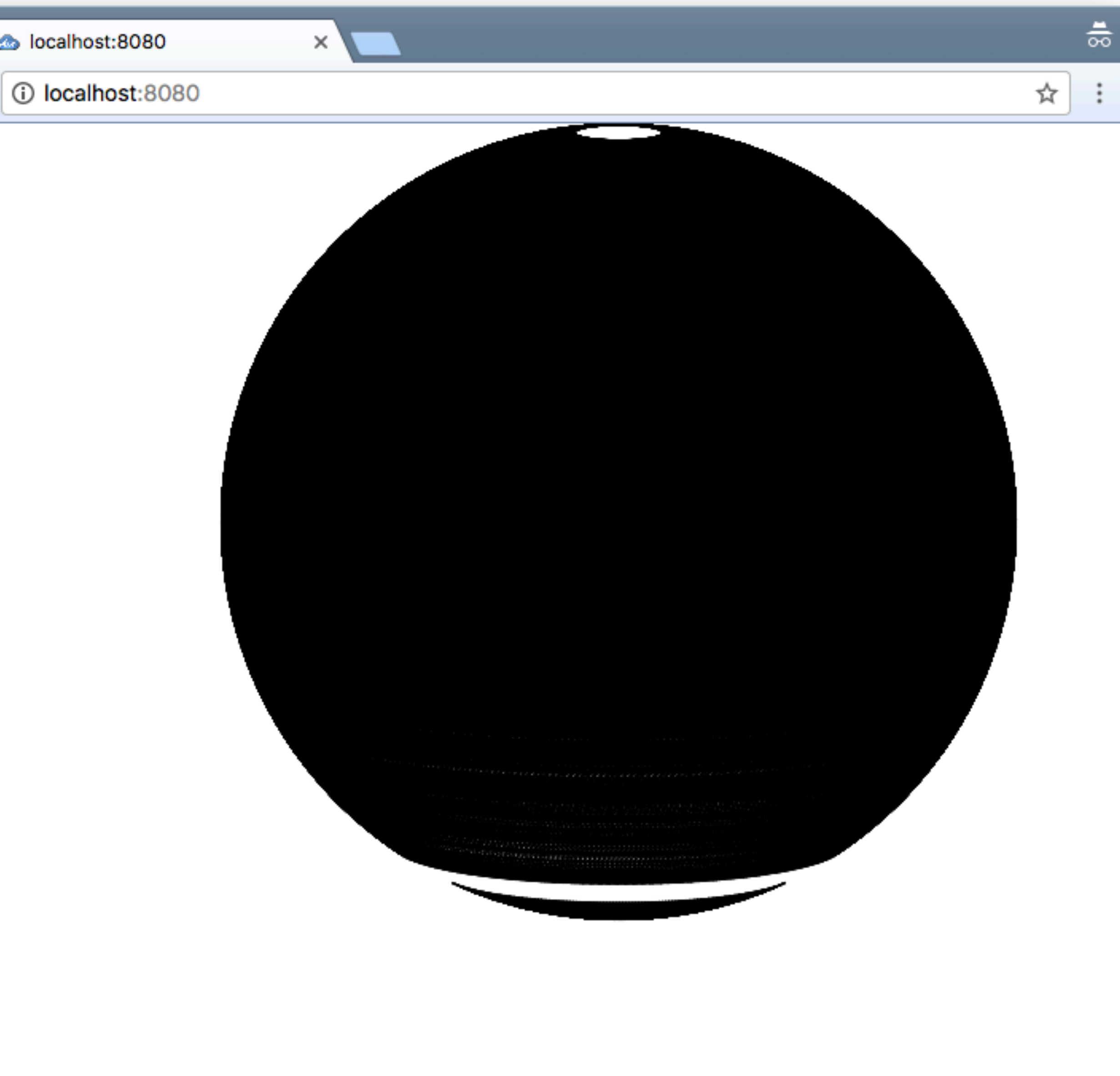


– *To obviate flicker from white light projected on a bright surface requires about 48 obscurations per second*

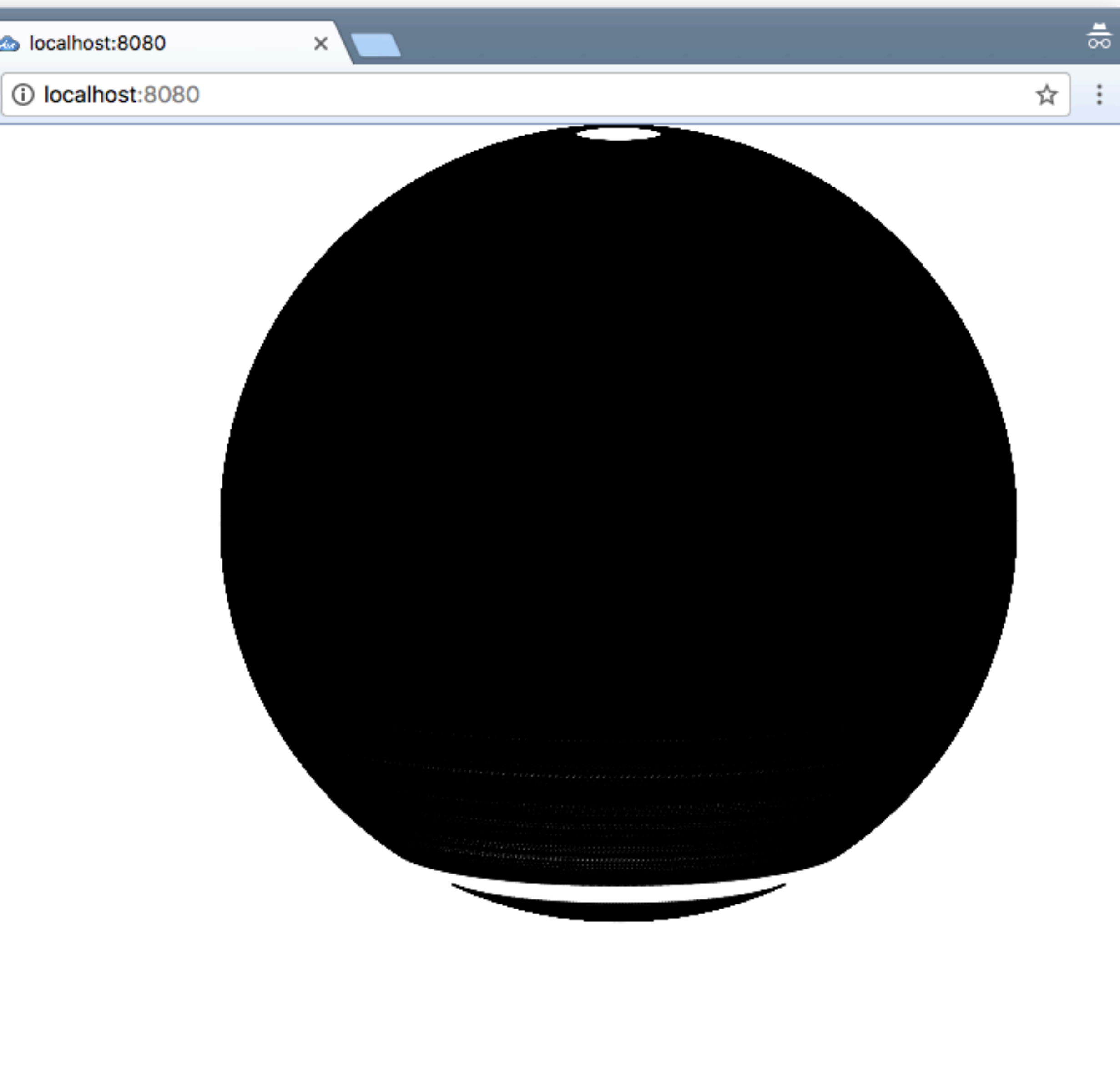
Эдисон,
который с лампочкой



Странный тест на утечку



Странный тест на утечку

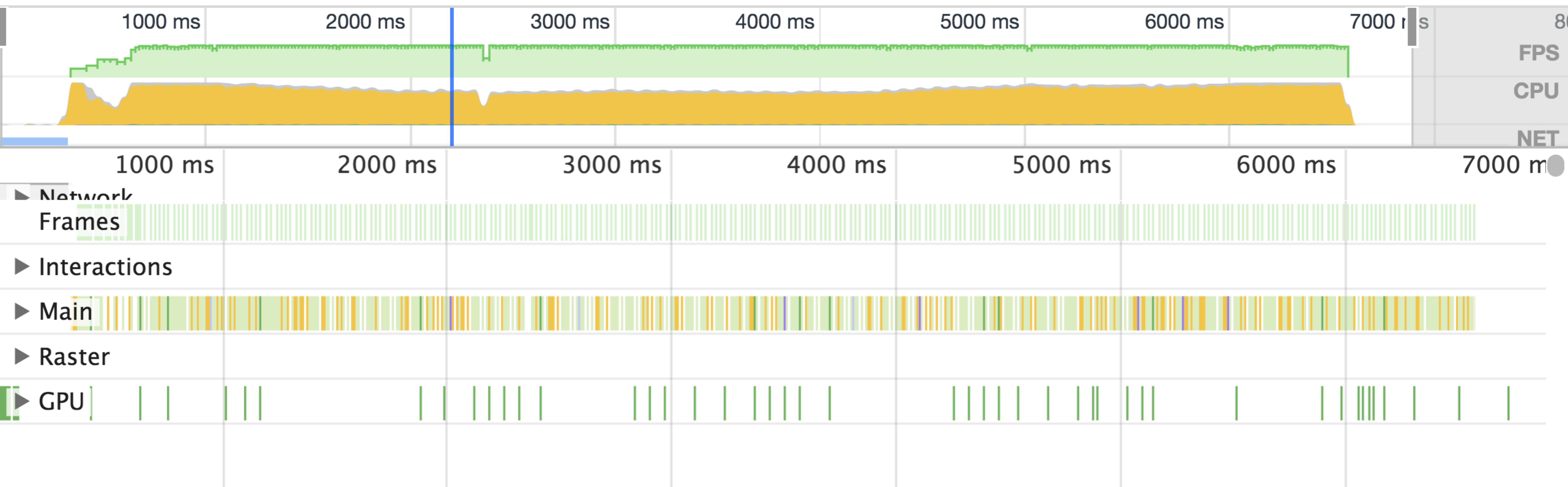


при вызове каждого
последующего кадра
«забыта» очистка
предыдущего: следующее
состояние рисуется поверх



canvas

результаты не
изменились – 6 сек,
~60 FPS



SVG



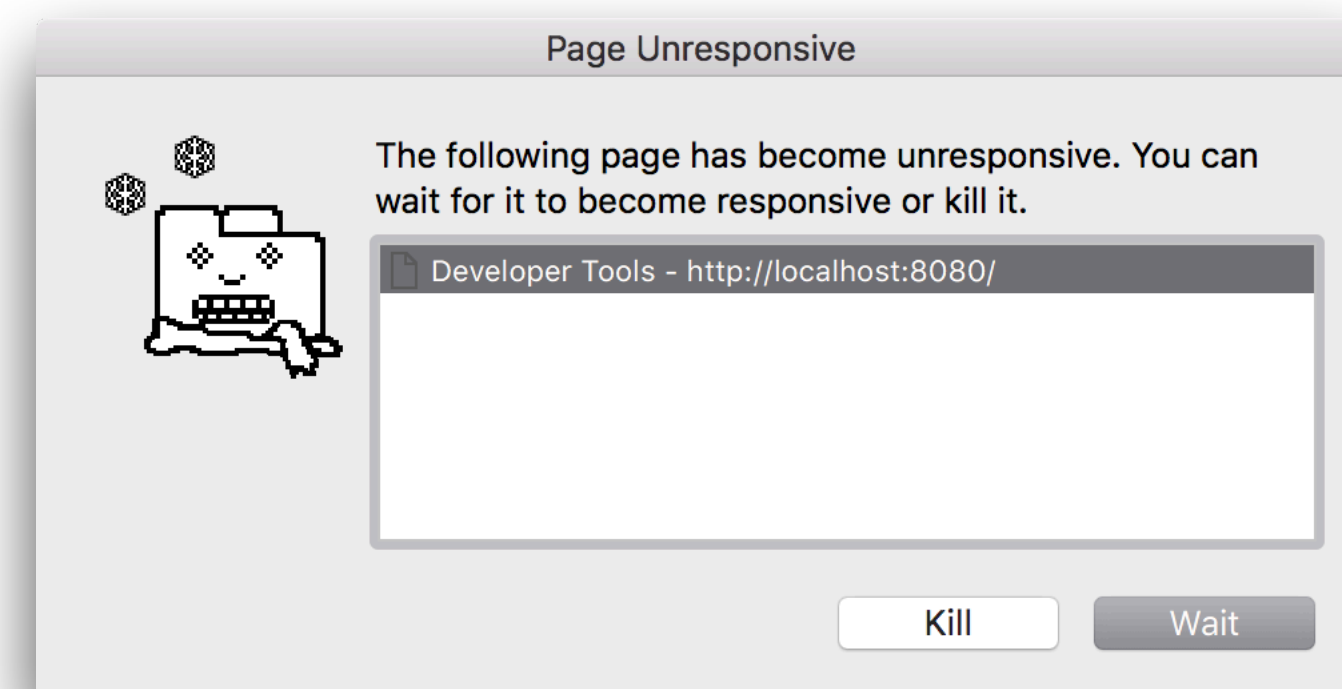
SVG

- полная длительность анимации составила 24 минуты



SVG

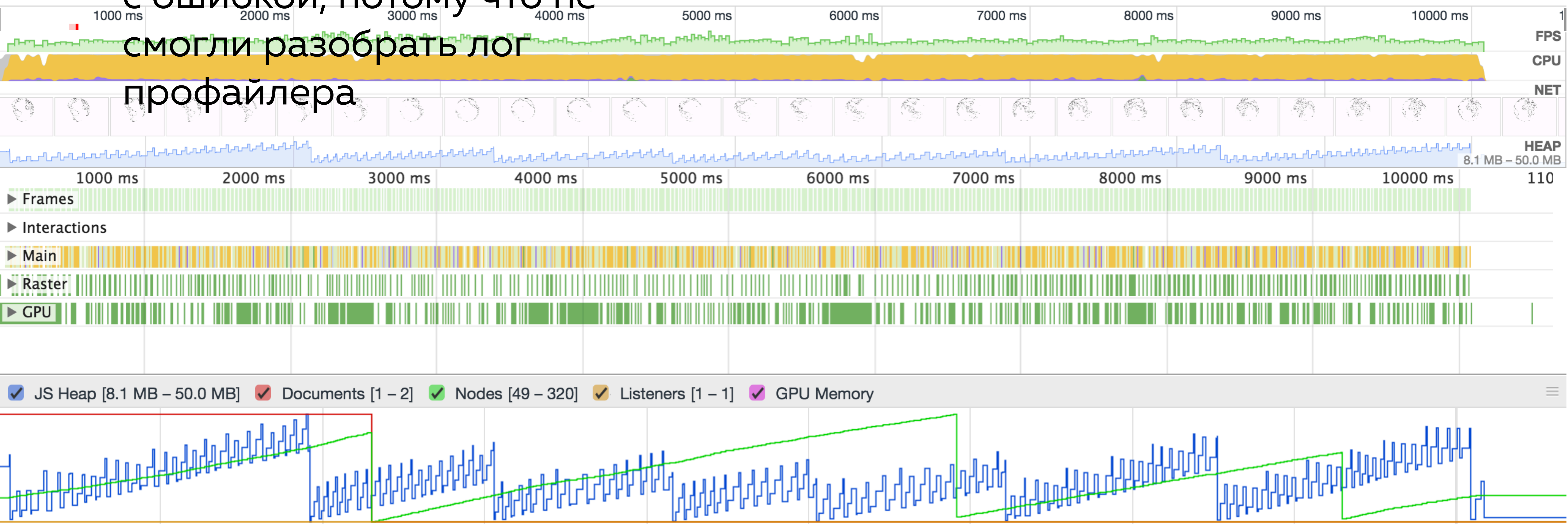
- полная длительность анимации составила 24 минуты
- DevTools хрома вылетели с ошибкой, потому что не смогли разобрать лог профайлера



SVG

- полная длительность анимации составила 24 минуты

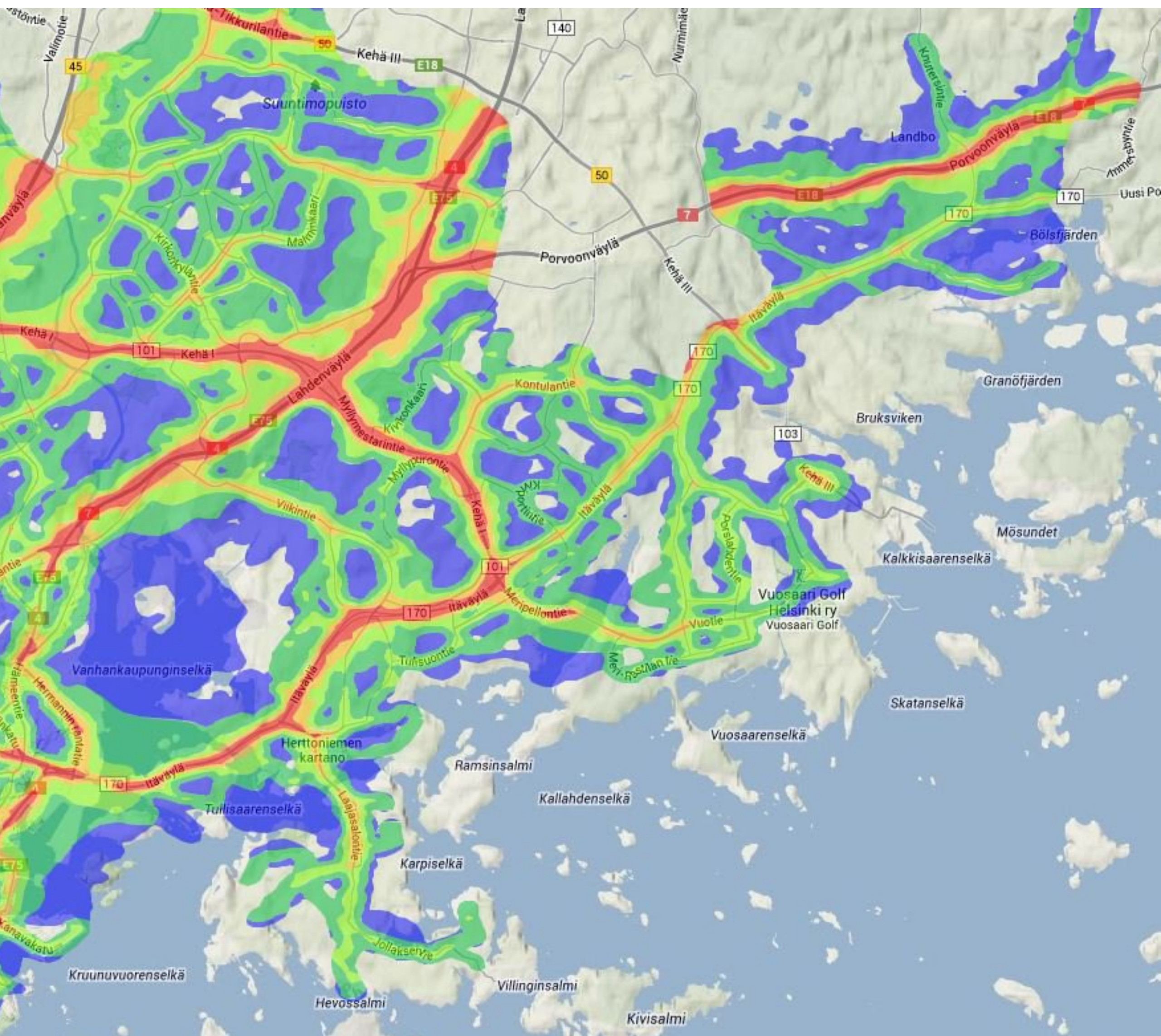
- DevTools хрома вылетели с ошибкой, потому что не смогли разобрать лог профайлера



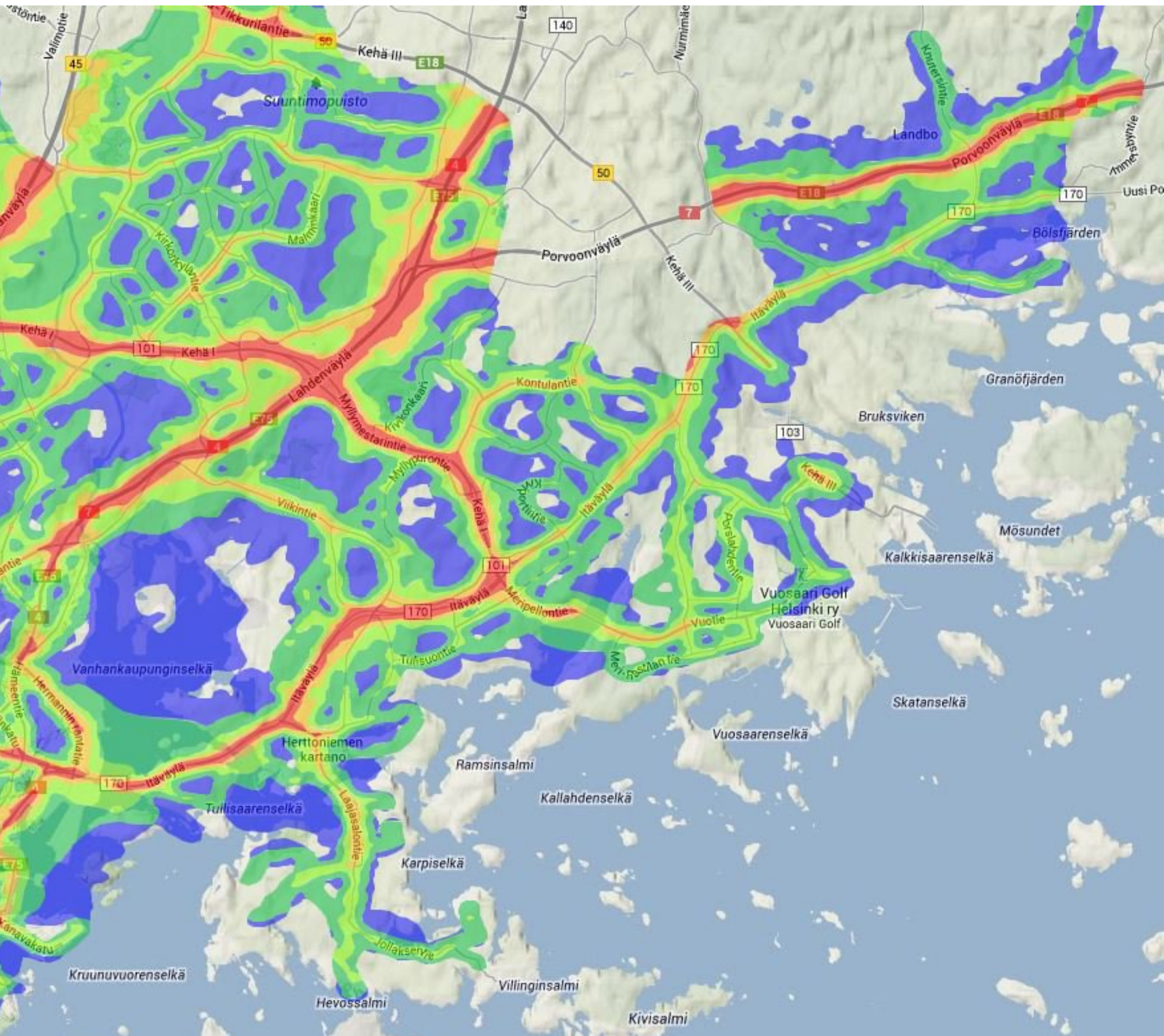
Отдать вычисления на сервер



Задача: тепловая карта



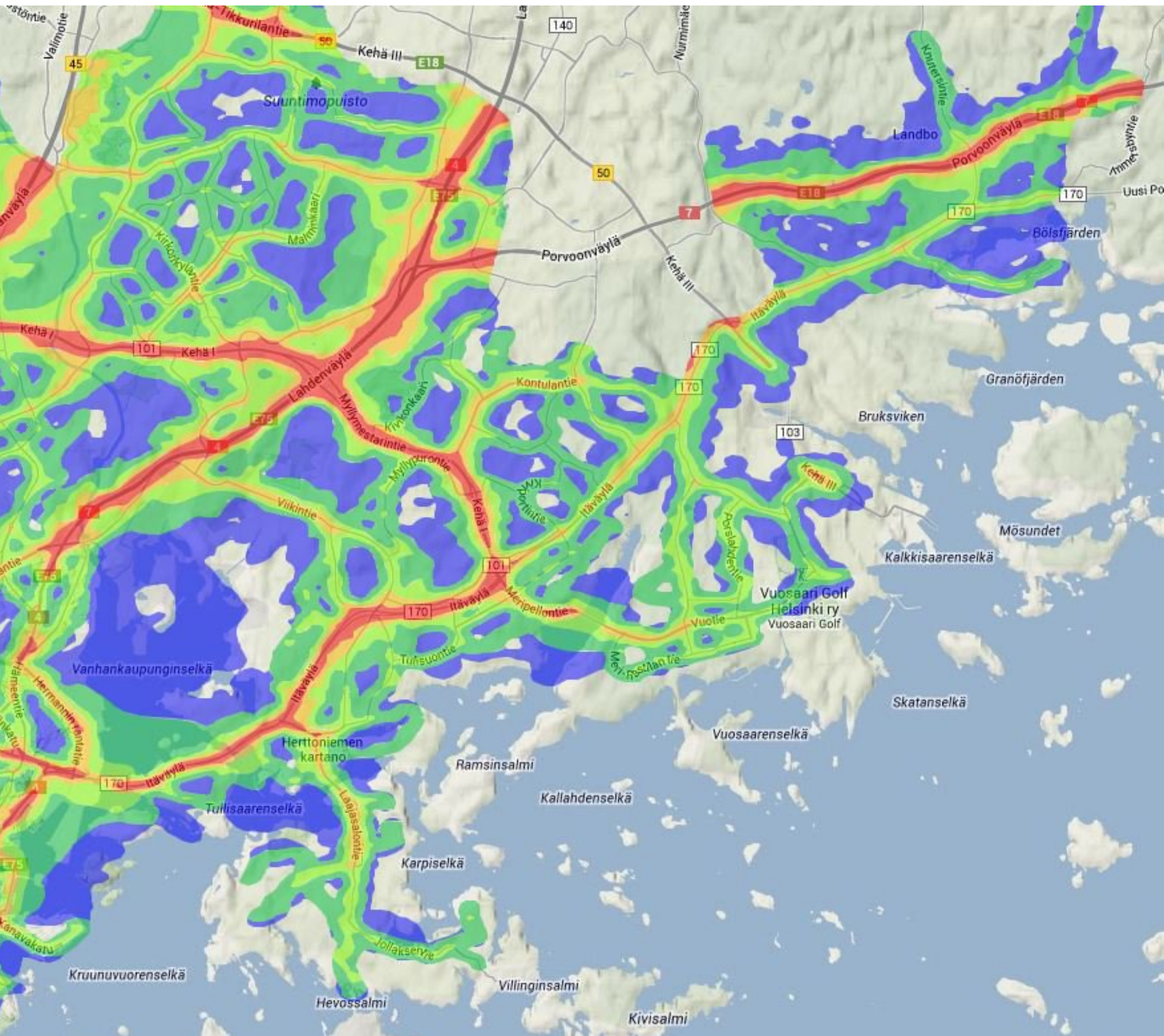
Задача: тепловая карта



- шаг обновления данных – 1рх



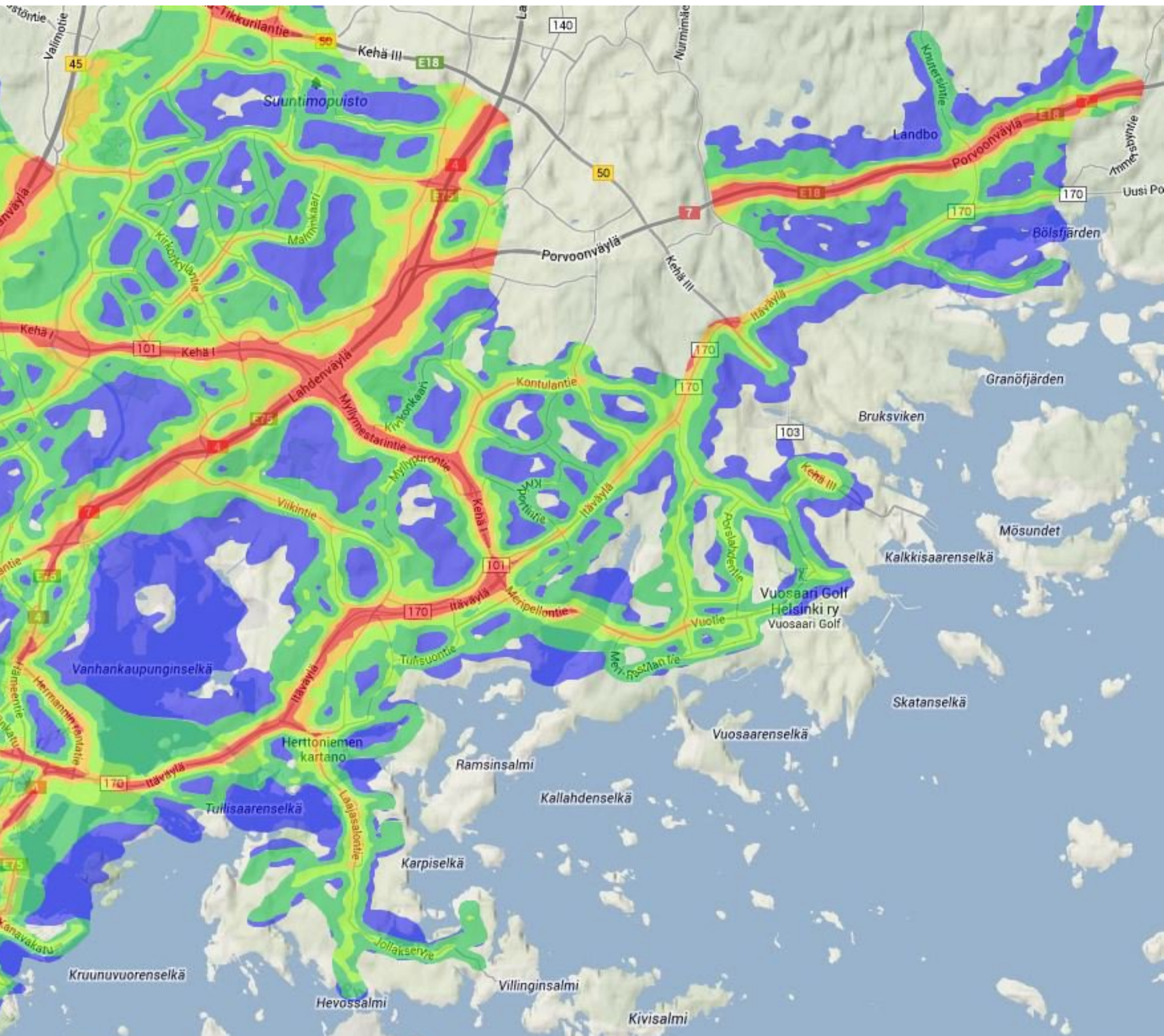
Задача: тепловая карта



- шаг обновления данных – 1рх
- полная перерисовка после каждого изменения состояния карты (заново создавать все canvas)



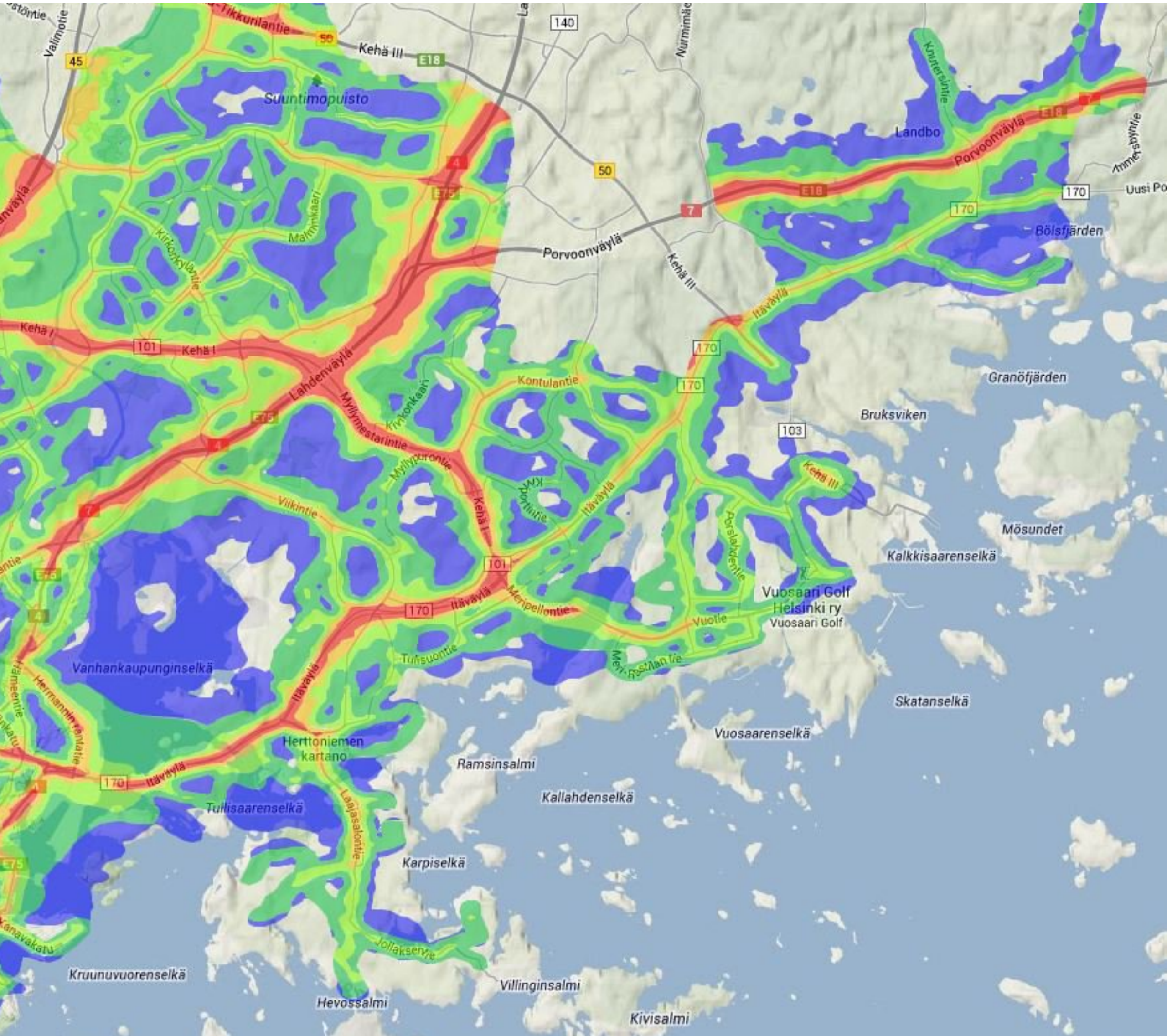
Задача: тепловая карта



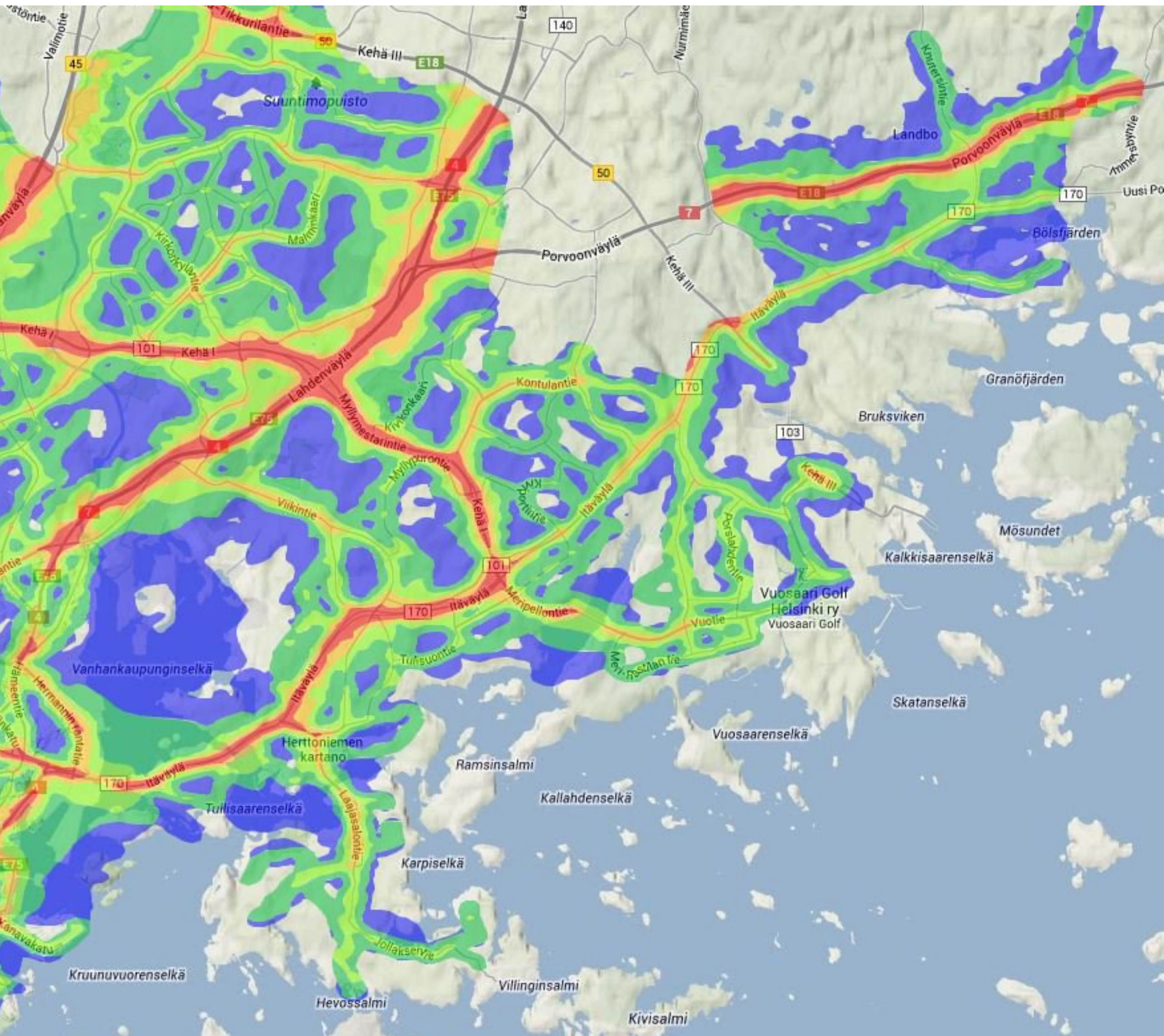
- шаг обновления данных – 1рх
- полная перерисовка после каждого изменения состояния карты (*заново создавать все canvas*)
- отсутствие интерактивного взаимодействия: данные статичны



Решение на canvas



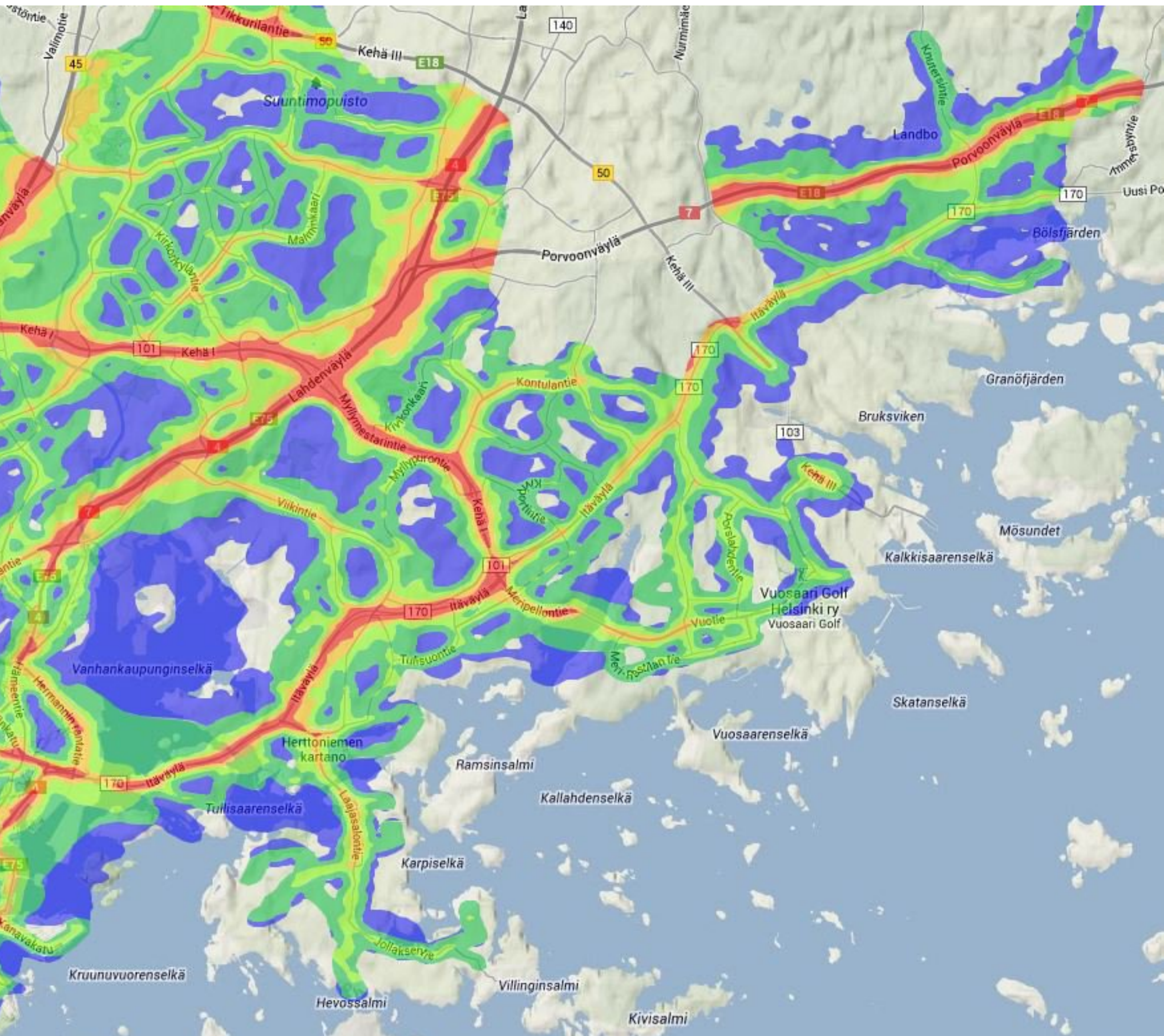
Решение на canvas



- с нуля написанный код — нет готовых хороших библиотек для создания тепловых карт



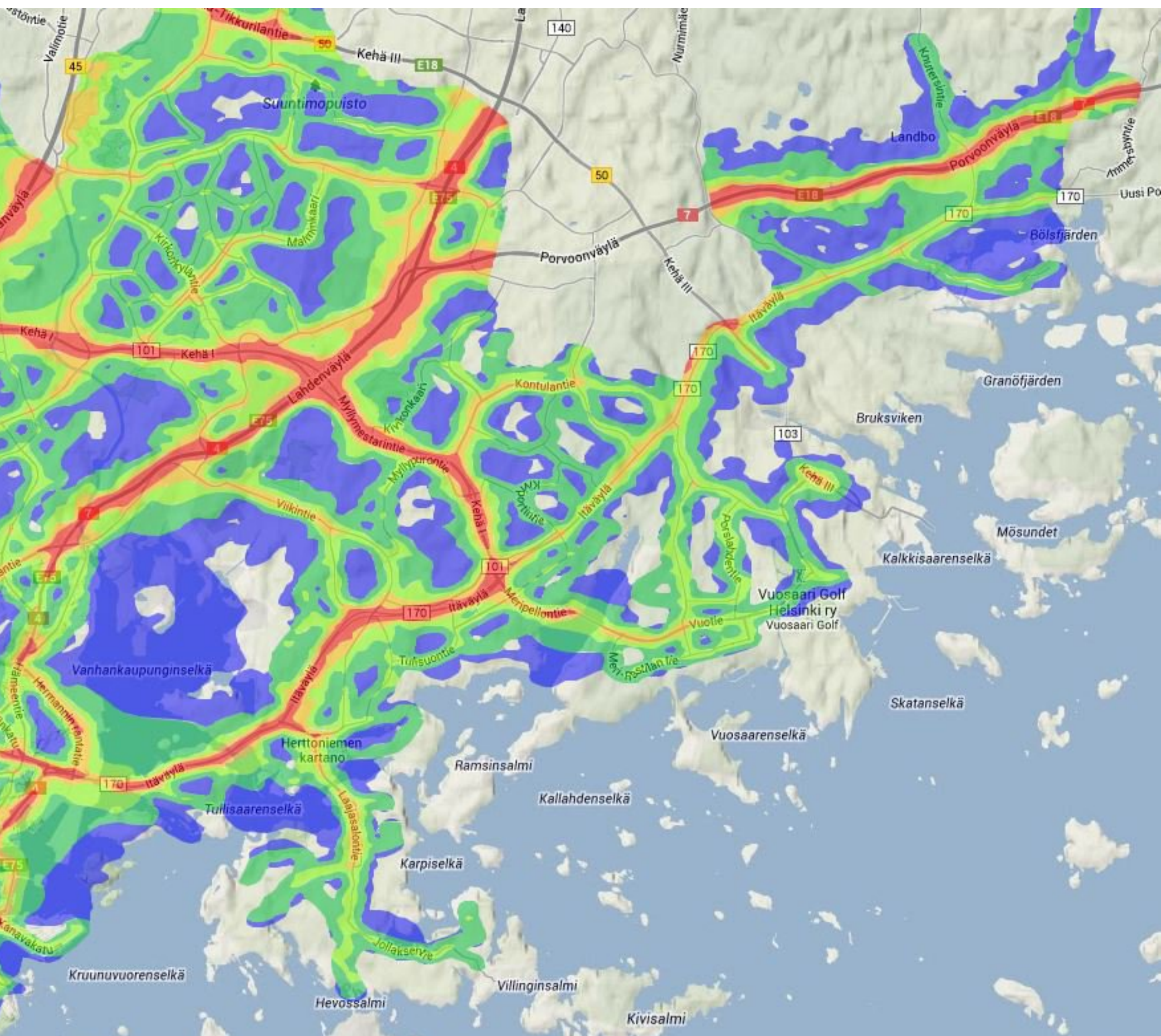
Решение на canvas



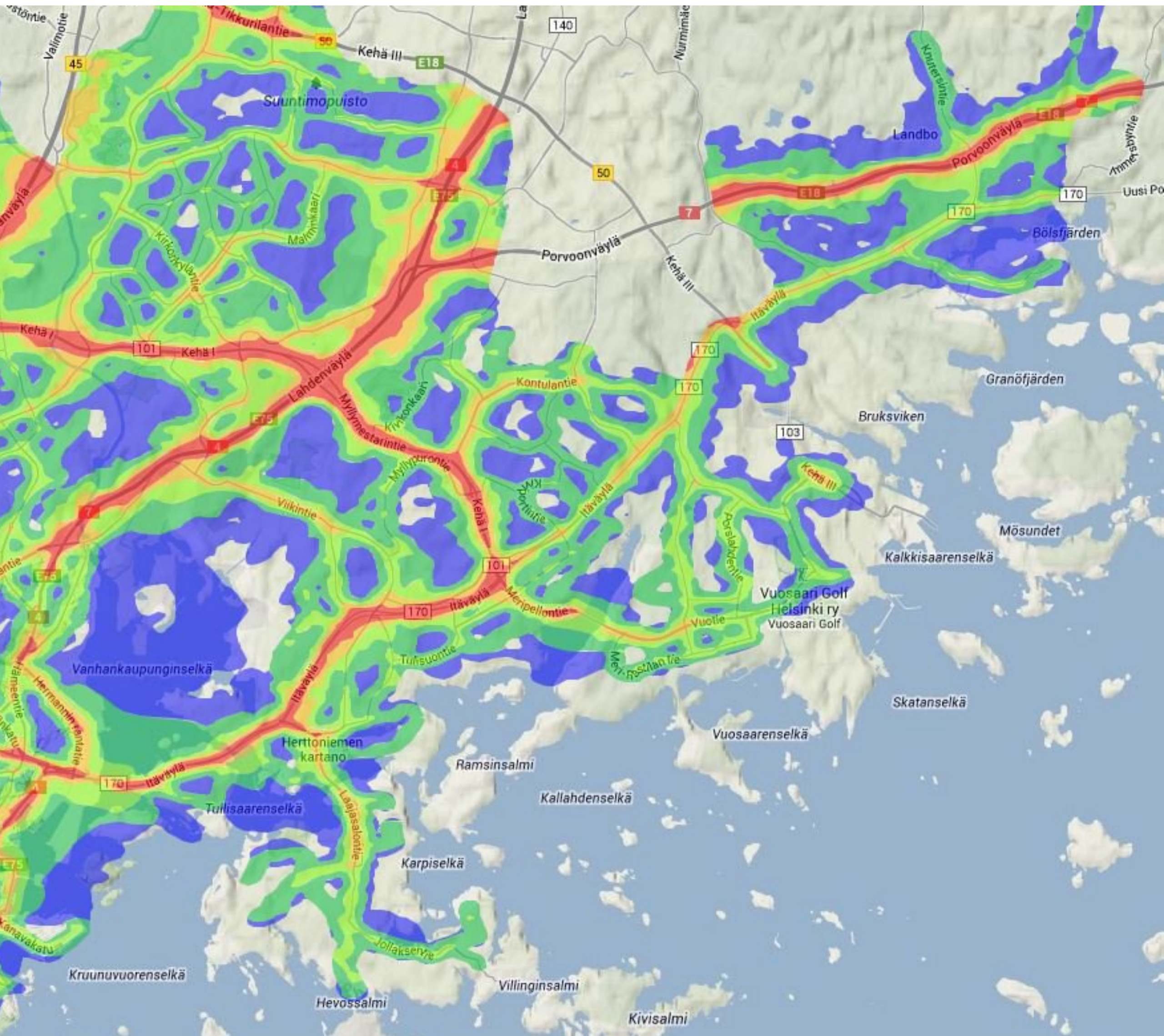
- с нуля написанный код — нет готовых хороших библиотек для создания тепловых карт
- большие вычисления приводят к одинаковому результату — тепловая карта не меняется со временем



Запрос изображений с сервера



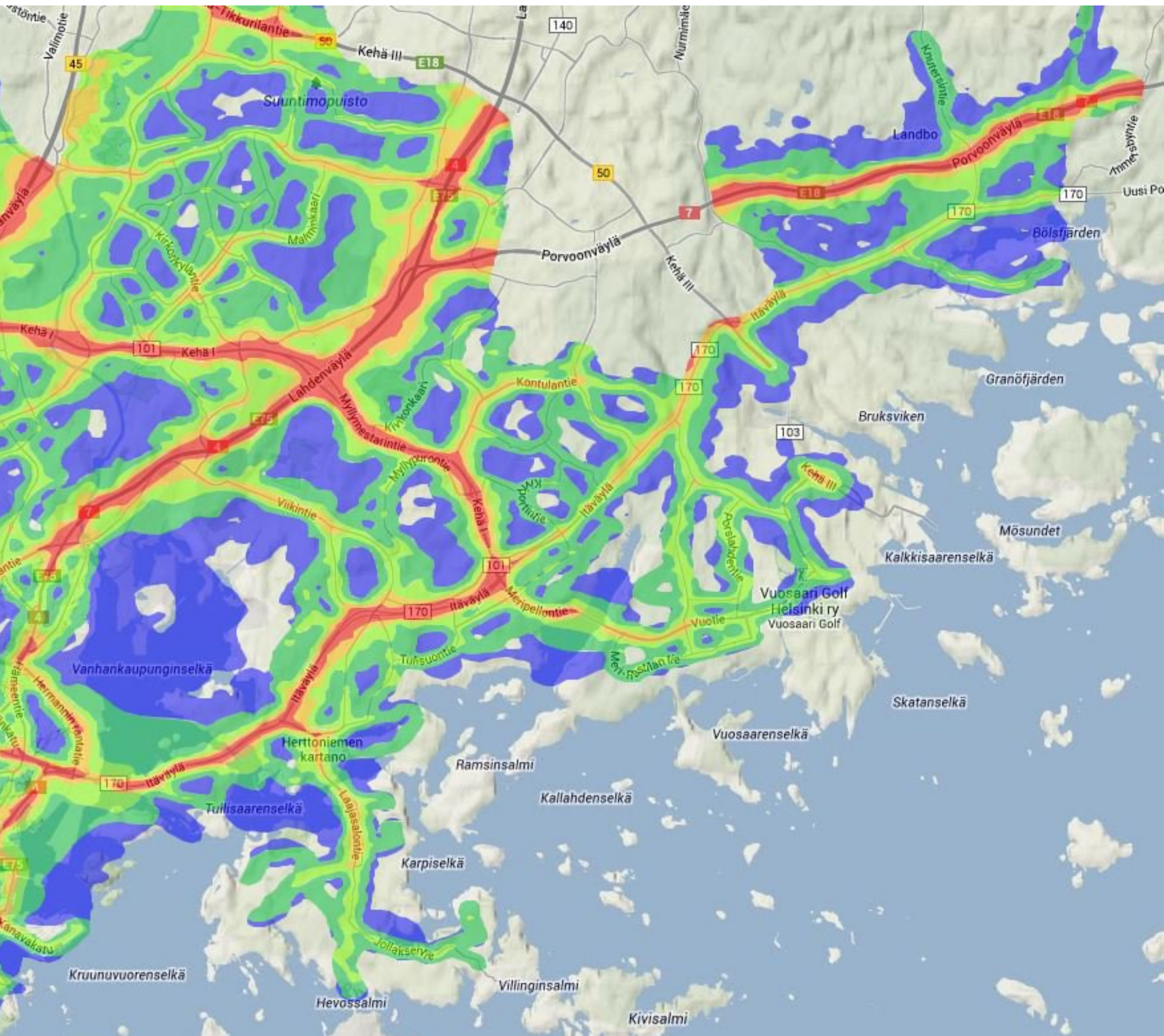
Запрос изображений с сервера



- изображения сгенерированы с помощью библиотеки визуализации для Python



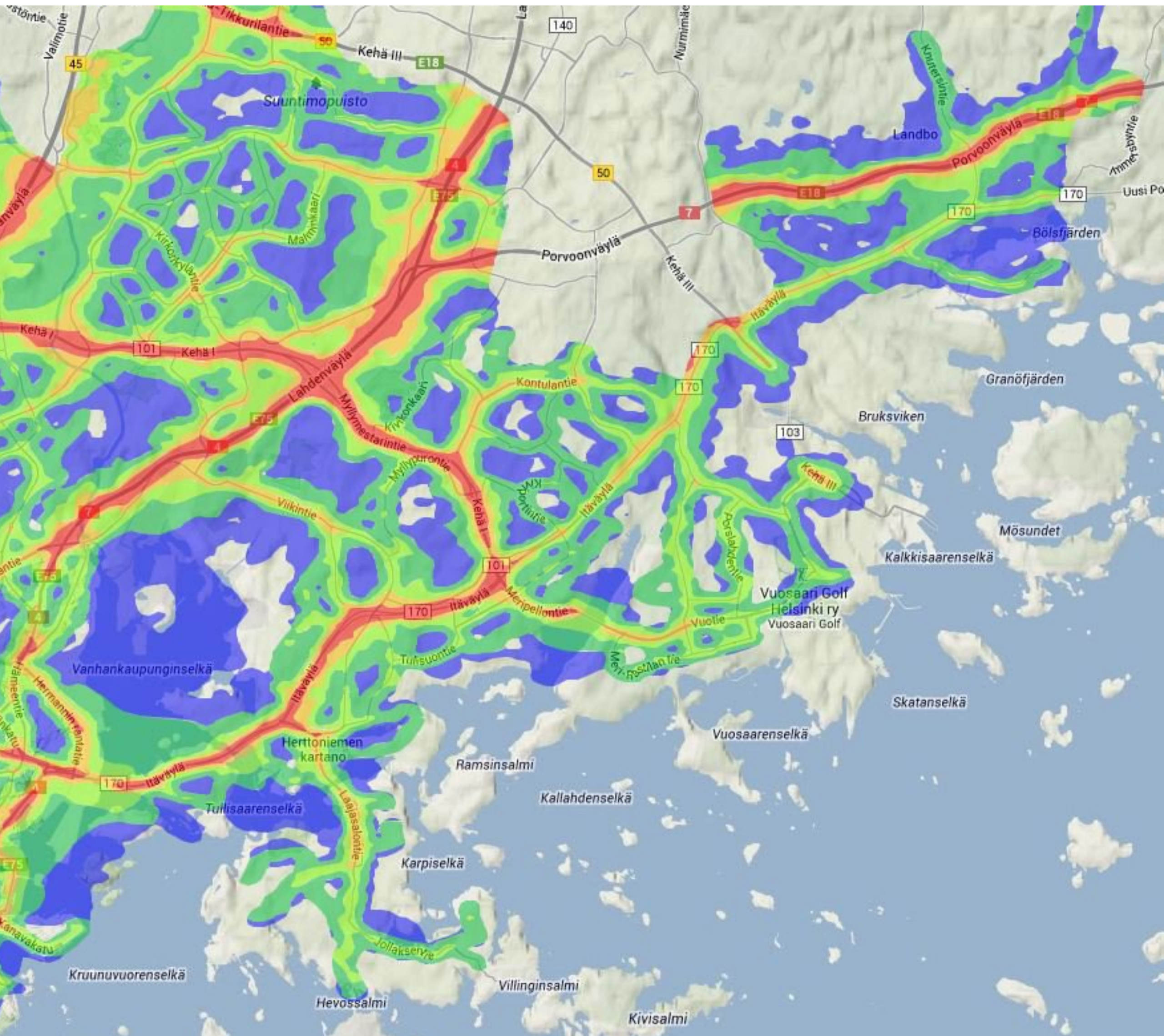
Запрос изображений с сервера



- изображения сгенерированы с помощью библиотеки визуализации для Python
- изображения хранятся на сервере и не создаются повторно для каждого использования



Запрос изображений с сервера



- изображения сгенерированы с помощью библиотеки визуализации для Python
- изображения хранятся на сервере и не создаются повторно для каждого использования
- кеширование уже загруженных изображений в браузере



Отдать в параллельный поток



Отдать в параллельный поток

редактор ace.js проводит проверку синтаксиса загруженного файла. Чтобы не тормозить основные вычисления (*работа с редактором*), проверка производится в параллельном потоке

```
6
7 - $('#submit').on('click', function() {
8     textarea.val(editor.getSession().getValue());
9 });
10
i 11 var a
12
x 13 var b -
```

Missing semicolon.
Unexpected early end of program.
Expected an identifier and instead saw '(end)'.



Отдать вычисления



Отдать вычисления

- измените рендерер d3.js на канвас



Отдать вычисления

- измените рендерер d3.js на канвас
- много запросов — не всегда плохо, иногда они работают быстрее, чем вычисления на клиенте



Отдать вычисления

- измените рендерер d3.js на канвас
- много запросов — не всегда плохо, иногда они работают быстрее, чем вычисления на клиенте
- сложные оверлеи на картах Google лучше сделать один раз на сервере картинками



Отдать вычисления

- измените рендерер d3.js на канвас
- много запросов — не всегда плохо, иногда они работают быстрее, чем вычисления на клиенте
- сложные оверлеи на картах Google лучше сделать один раз на сервере картинками
- то, что можно посчитать параллельно, отдайте в `ServiceWorker`



Что делать, если оптимизировать невозможно



Если оптимизировать невозможно

- использовать обратную связь (*спиннеры, заблокированные кнопки*)
- использовать особенности восприятия (*приветственный экран Apple – муляж, фотография последнего экрана*)



Правильная обратная связь



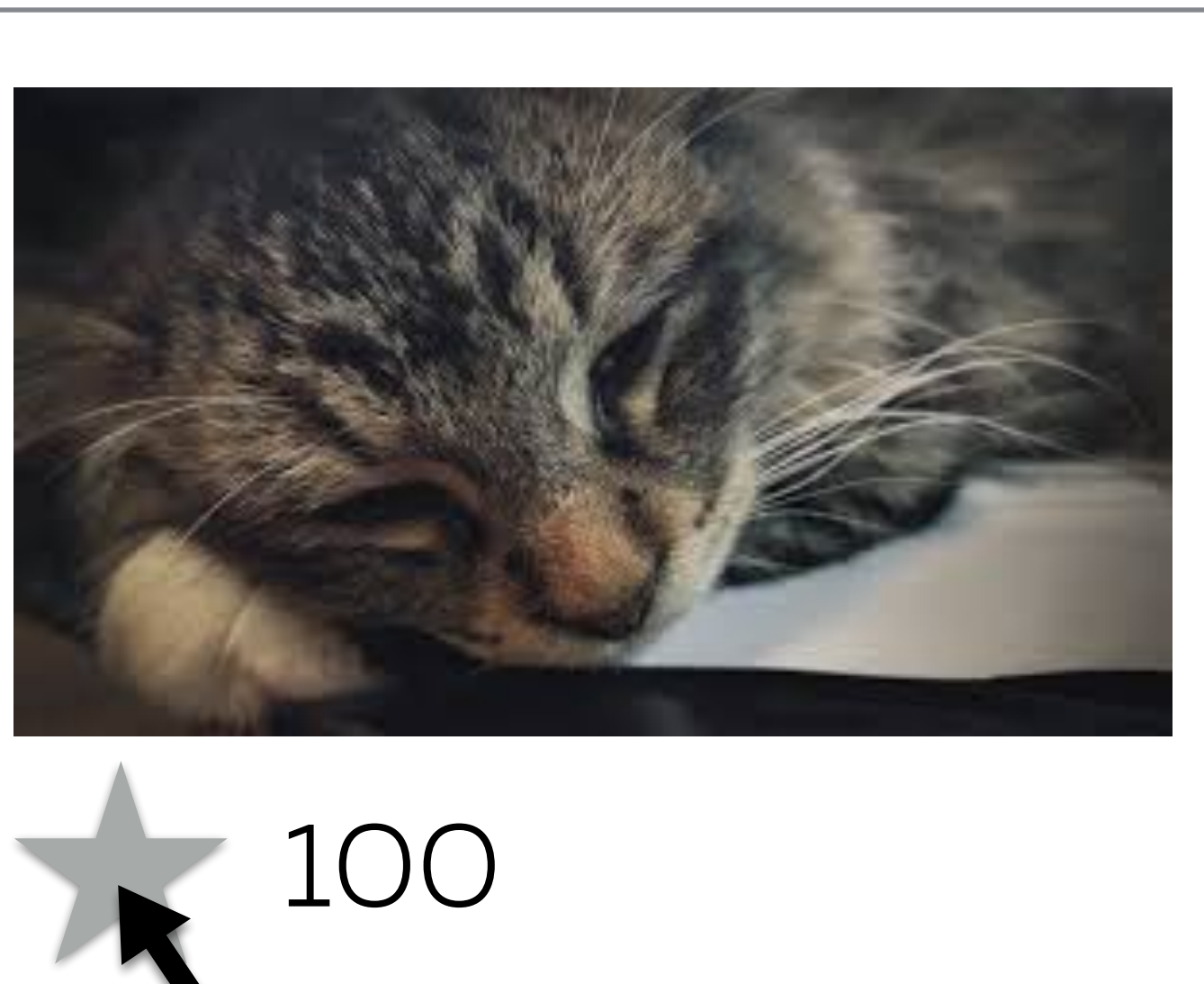
Обратная связь курильщика



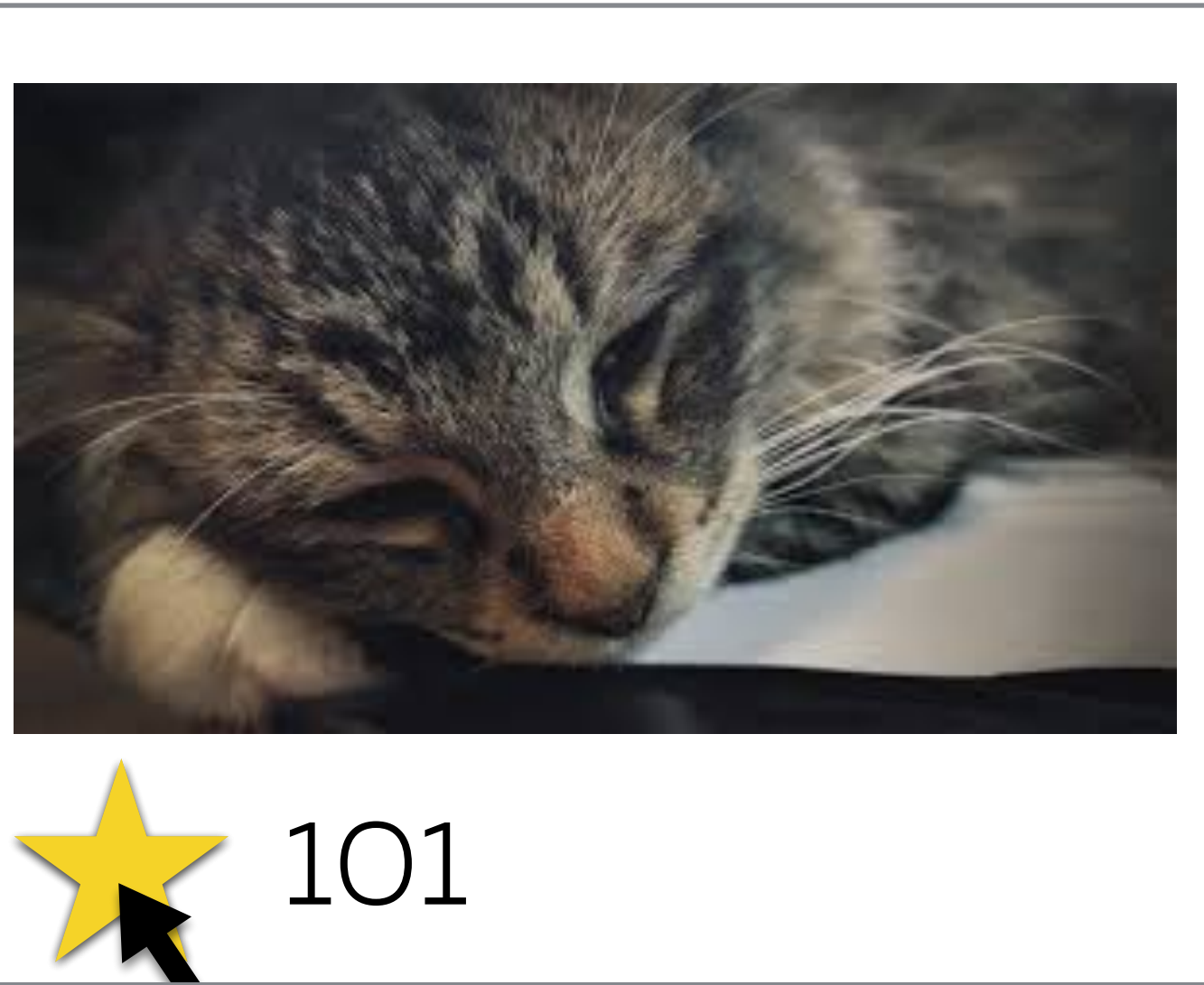
★ 100



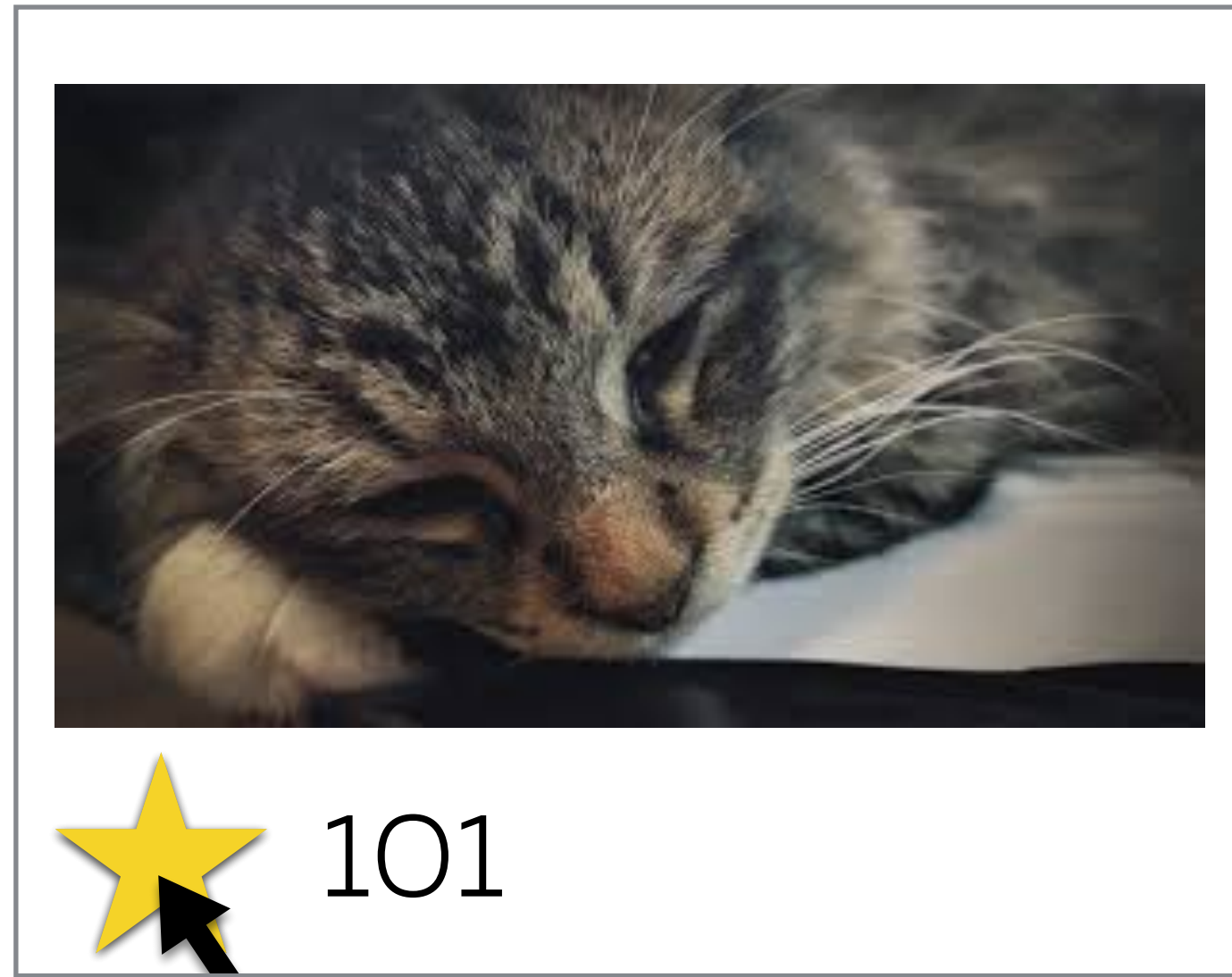
Обратная связь курильщика



Обратная связь курильщика



Обратная связь курильщика



GET <http://localhost/> 500 Internal Server Err...



Обратная связь курильщика



GET <http://localhost/> 500 Internal Server Err...

Some error happened

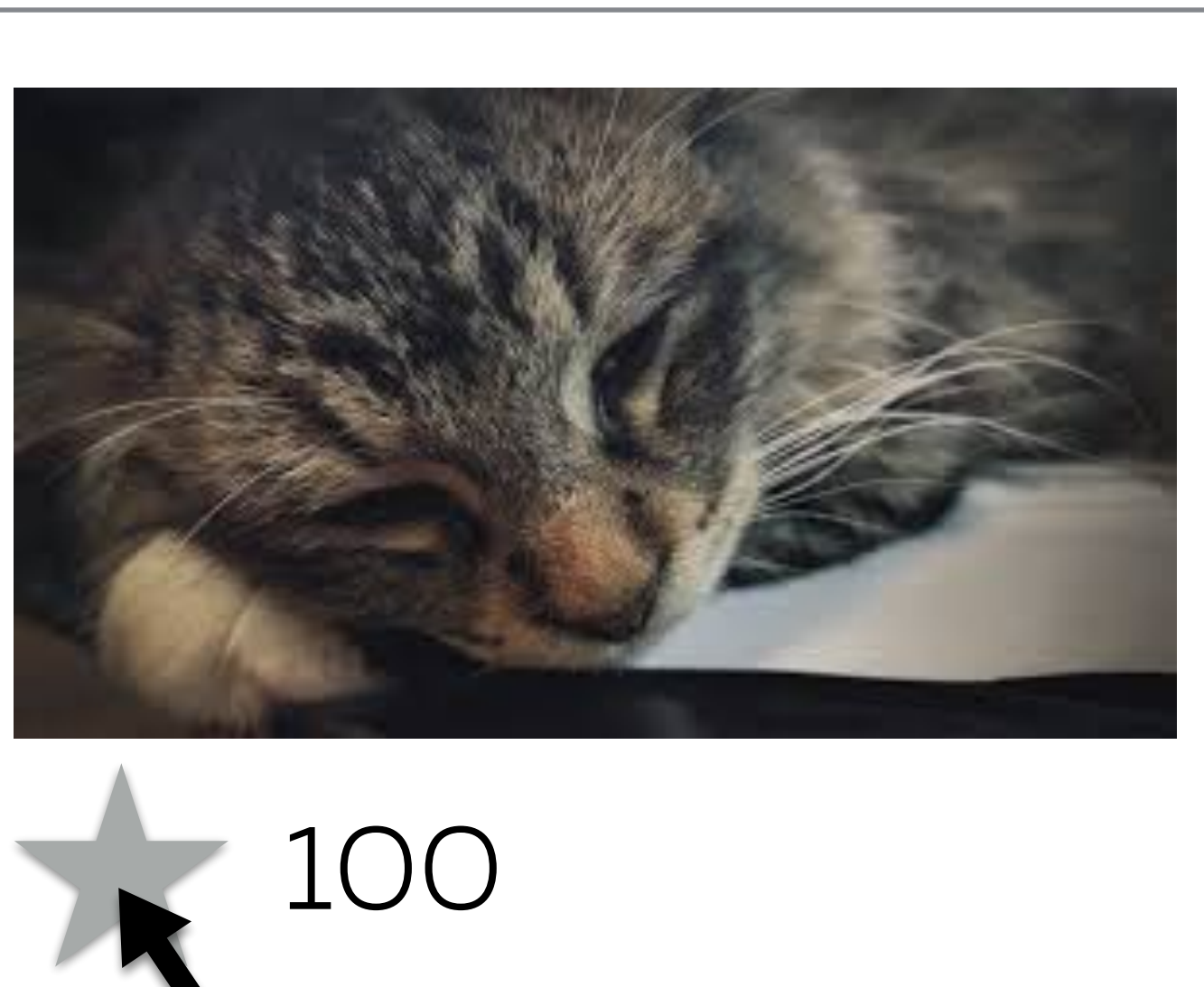


Обратная связь здорового человека



★ 100

Обратная связь здорового человека



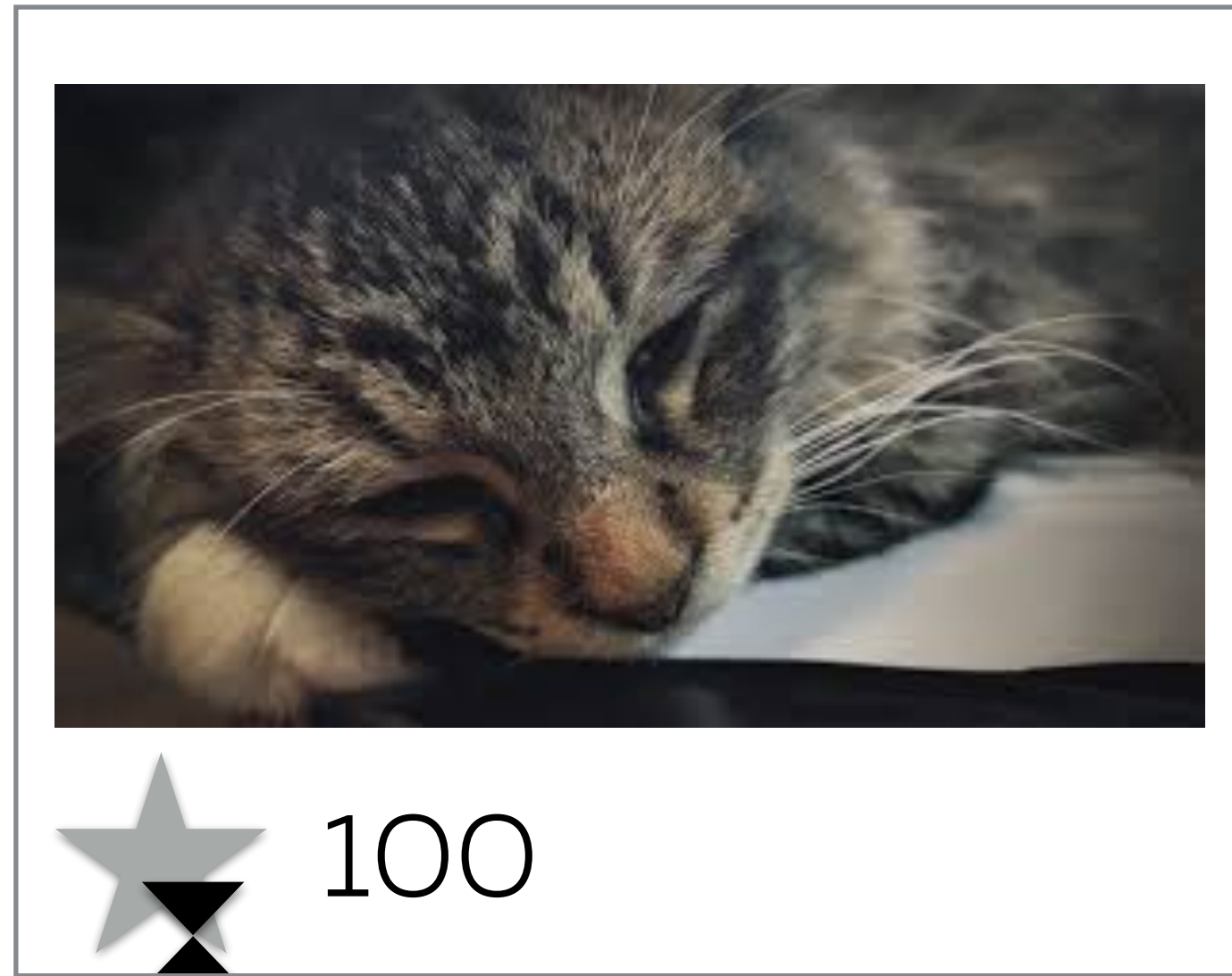
Обратная связь здорового человека



★ 100

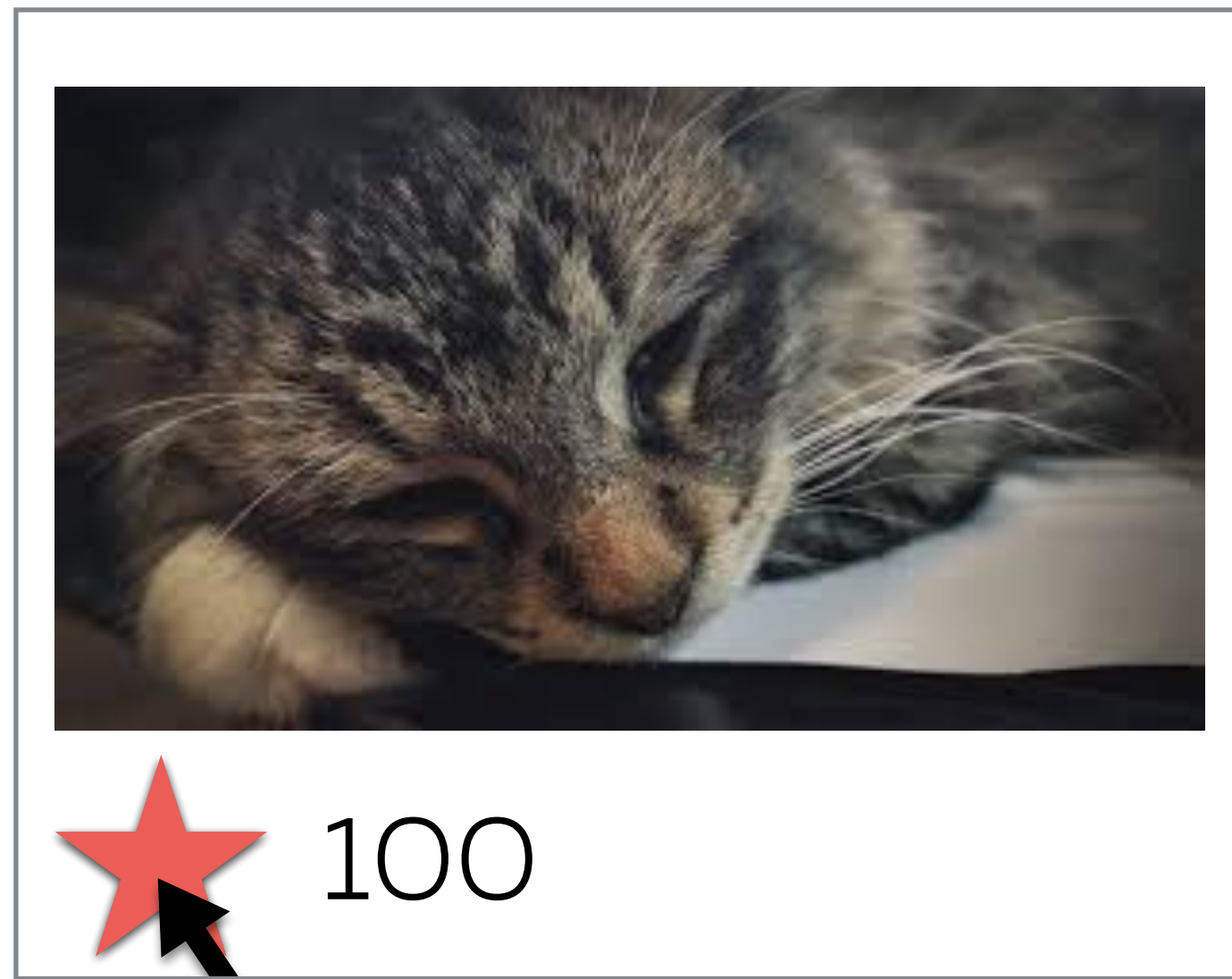


Обратная связь здорового человека



GET <http://localhost/> 500 Internal Server Err...

Обратная связь здорового человека



GET <http://localhost/> 500 Internal Server Err...

Обратная связь здорового человека



Что-то пошло не так и ваш голос не зачёлся

GET <http://localhost/> 500 Internal Server Err...

Обратная связь курильщика: 2

Месть обратной связи



Обратная связь курильщика: 2

Месть обратной связи

Очень остроумный коммент | Клик!

Обратная связь курильщика: 2

Месть обратной связи



Обратная связь курильщика: 2

Месть обратной связи

Очень остроумный коммент | – Клик!

Обратная связь курильщика: 2

Месть обратной связи



Обратная связь курильщика: 2

Месть обратной связи

Очень остроумный коммент

GET <http://localhost/> 204

Обратная связь курильщика: 2

Месть обратной связи

Очень остроумный коммент

Очень остроумный коммент

GET <http://localhost/> 204

GET <http://localhost/> 204

Обратная связь курильщика: 2

Месть обратной связи

Очень остроумный комментарий 👎

Очень остроумный комментарий 👎

GET <http://localhost/> 204

GET <http://localhost/> 204

Обратная связь здорового человека: 2

Новая надежда



Обратная связь здорового человека: 2

Новая надежда

Очень остроумный коммент | Клик!

Обратная связь здорового человека: 2

Новая надежда

Очень остроумный коммент |

Обратная связь здорового человека: 2

Новая надежда

Очень остроумный коммент

GET <http://localhost/> 204

Обратная связь здорового человека: 2

Новая надежда

Очень остроумный коммент 👍

GET <http://localhost/> 204

Обратная связь здорового человека: 2

Новая надежда

Очень остроумный коммент 👍

Очень остроумно!

GET <http://localhost/> 204

Скриншот



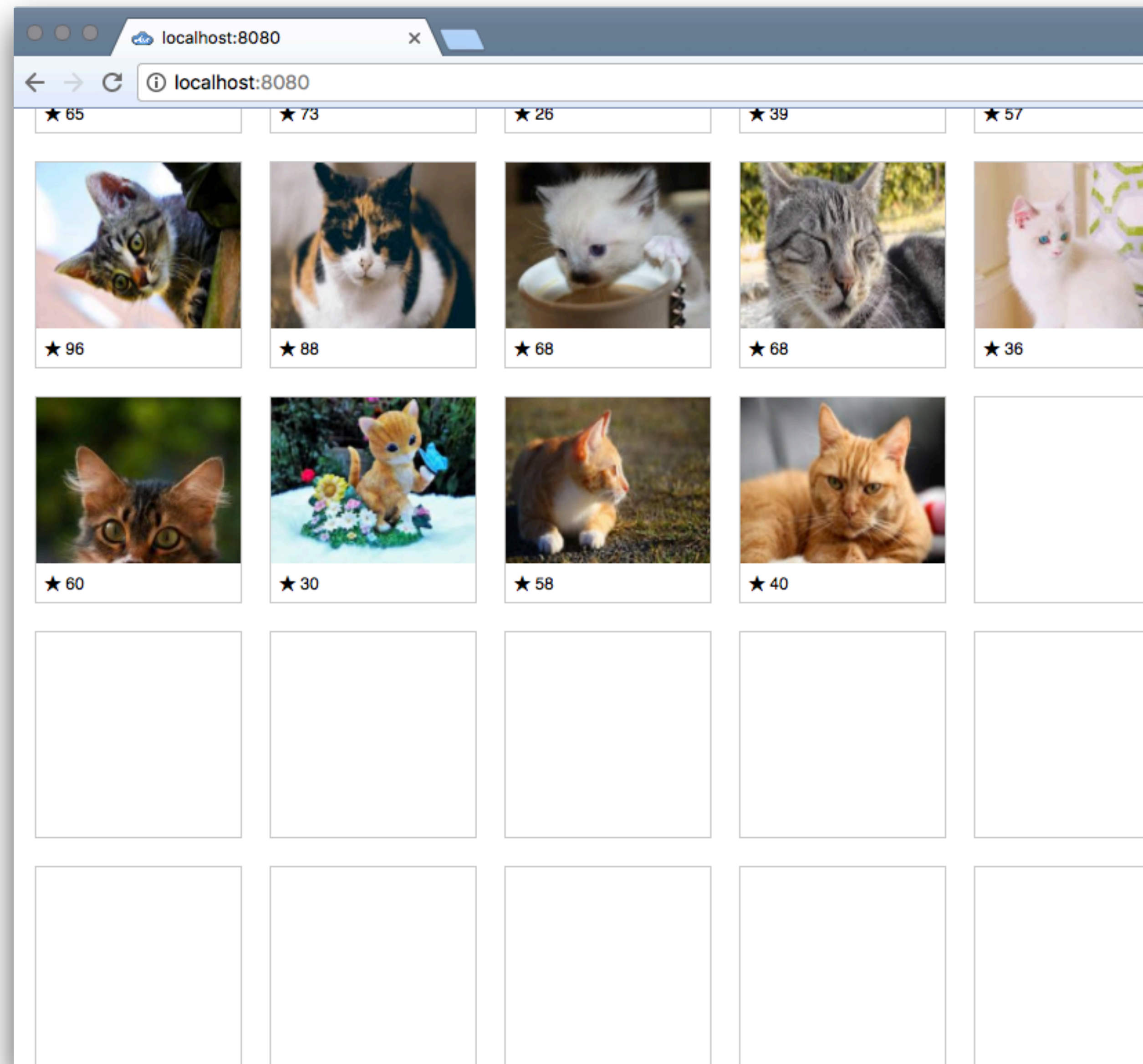
Mac OS во время загрузки

чтобы создать впечатление мгновенной инициализации ОС, Mac показывает скриншот последнего состояния страницы, а в фоне производит необходимые вычисления



Динамическая прокрутка

если элементы не успели прорисоваться при быстрой прокрутке, можно показать их муляжи, которые при остановке скролла заменятся на настоящие



Алгоритм оптимизации



Алгоритм оптимизации

1. Уменьшить нагрузку на процессор



Алгоритм оптимизации

1. Уменьшить нагрузку на процессор

1. использовать *меньше* памяти



Алгоритм оптимизации

1. Уменьшить нагрузку на процессор

1. использовать *меньше* памяти

2. проверить частоту кадров, вероятно её можно снизить



Алгоритм оптимизации

1. Уменьшить нагрузку на процессор
 1. использовать *меньше* памяти
 2. проверить частоту кадров, вероятно её можно снизить
 3. использовать все возможные ресурсы для расчётов



Алгоритм оптимизации

1. Уменьшить нагрузку на процессор
 1. использовать *меньше* памяти
 2. проверить частоту кадров, вероятно её можно снизить
 3. использовать все возможные ресурсы для расчётов
2. Добавить обратную связь в интерфейс



Алгоритм оптимизации

1. Уменьшить нагрузку на процессор
 1. использовать *меньше* памяти
 2. проверить частоту кадров, вероятно её можно снизить
 3. использовать все возможные ресурсы для расчётов
2. Добавить обратную связь в интерфейс
3. Начать оптимизацию памяти 🙌



 **Спасибо!**



o0.github.io

латинская «о», ноль

