



Кошелёк или деньги

вычислять или вспоминать

Игорь Алексеенко, училка из HTML Academy



**Никто не любит тормоза
в интерфейсе 😡**



Тормоза в интерфейсе



Тормоза в интерфейсе

- неприятные впечатления от сайта (*в лучшем случае*)



Тормоза в интерфейсе

- неприятные впечатления от сайта (*в лучшем случае*)
- ошибки при производстве операций (*в худшем случае*)



Тормоза в интерфейсе

- неприятные впечатления от сайта (*в лучшем случае*)
- ошибки при производстве операций (*в худшем случае*)
 - повторная отправка данных



Тормоза в интерфейсе

- неприятные впечатления от сайта (*в лучшем случае*)
- ошибки при производстве операций (*в худшем случае*)
 - повторная отправка данных
 - неправильное прицеливание (*в играх*)



Тормоза в интерфейсе

- неприятные впечатления от сайта (*в лучшем случае*)
- ошибки при производстве операций (*в худшем случае*)
 - повторная отправка данных
 - неправильное прицеливание (*в играх*)
- уход пользователей



Тормоза в интерфейсе

- неприятные впечатления от сайта (*в лучшем случае*)
- ошибки при производстве операций (*в худшем случае*)
 - повторная отправка данных
 - неправильное прицеливание (*в играх*)
- уход пользователей
- потеря денег



Что такое тормоза?

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



Динамическое взаимодействие с пользователем дискретно



Дискретное взаимодействие



Дискретное взаимодействие

- анимация



Дискретное взаимодействие

- анимация
- прокрутка



Дискретное взаимодействие

- анимация
- прокрутка
- нажатия на кнопки

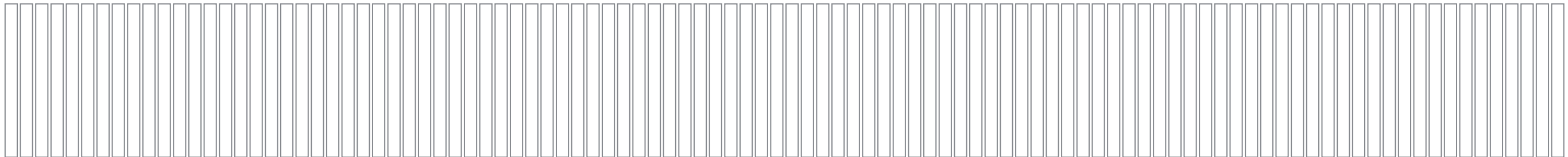


Дискретное взаимодействие

- анимация
- прокрутка
- нажатия на кнопки
- обработка асинхронных процедур

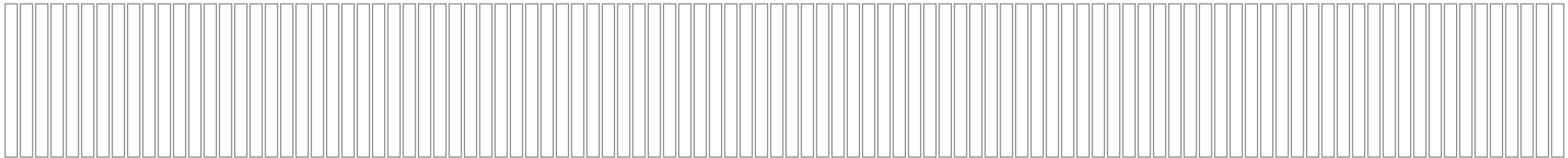


Откуда берутся тормоза?



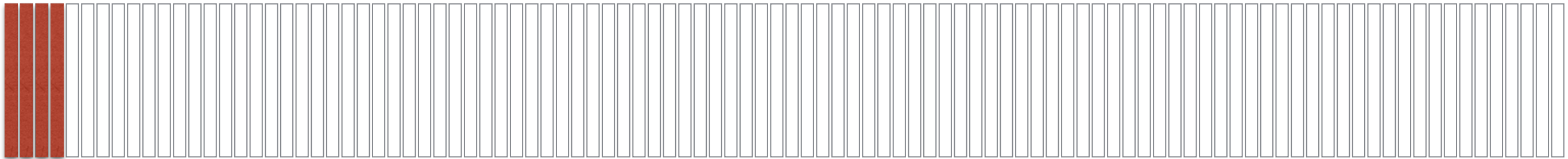
Откуда берутся тормоза?

кадр



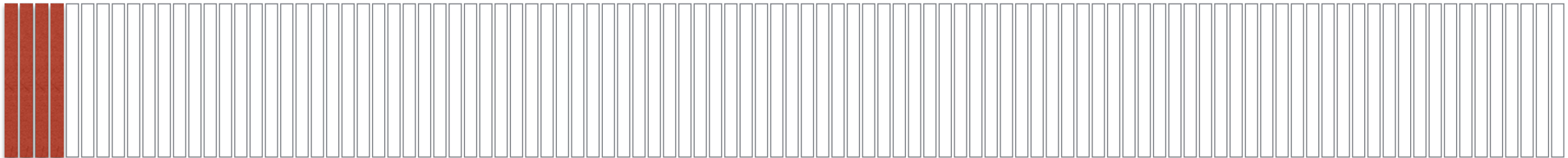
Откуда берутся тормоза?

кадр



Откуда берутся тормоза?

кадр

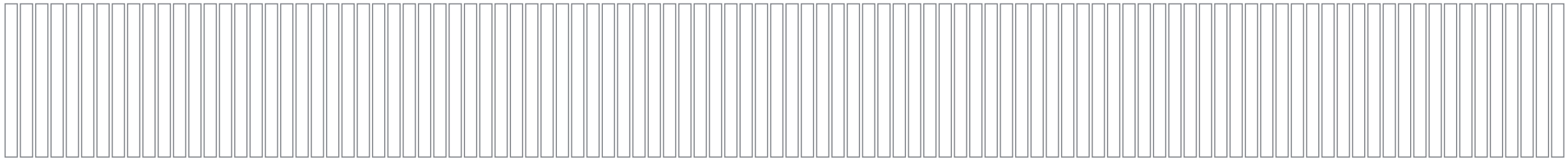


лаг



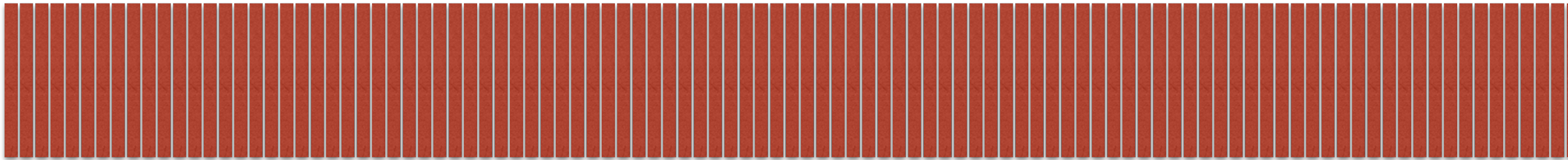
Откуда берутся тормоза?

кадр



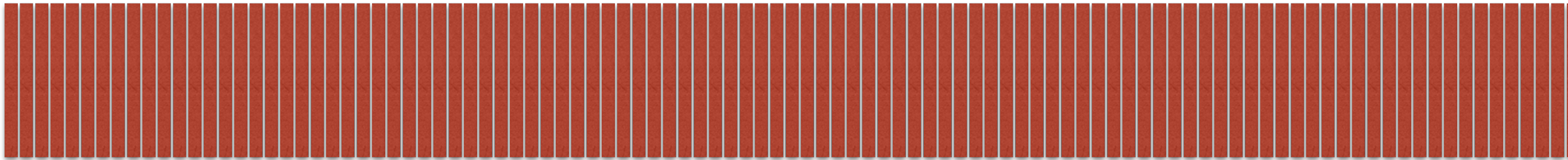
Откуда берутся тормоза?

кадр



Откуда берутся тормоза?

кадр



фриз



Откуда берутся тормоза

вычисления процессора занимают больше одного кадра отрисовки и пользователь это замечает





Процессор

используется для мгновенных
вычислений



Память

используется для хранения
вычисленных сложных значений



Память



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*)



Почему тормозит память

происходит слишком долгое чтение из памяти, гораздо медленнее чем происходили бы расчеты



Почему тормозит память



Почему тормозит память

- чтение из переполненной памяти (*большие объемы данных*)



Почему тормозит память

- чтение из переполненной памяти (*большие объемы данных*)
- сборка мусора (*происходит в произвольные моменты*)

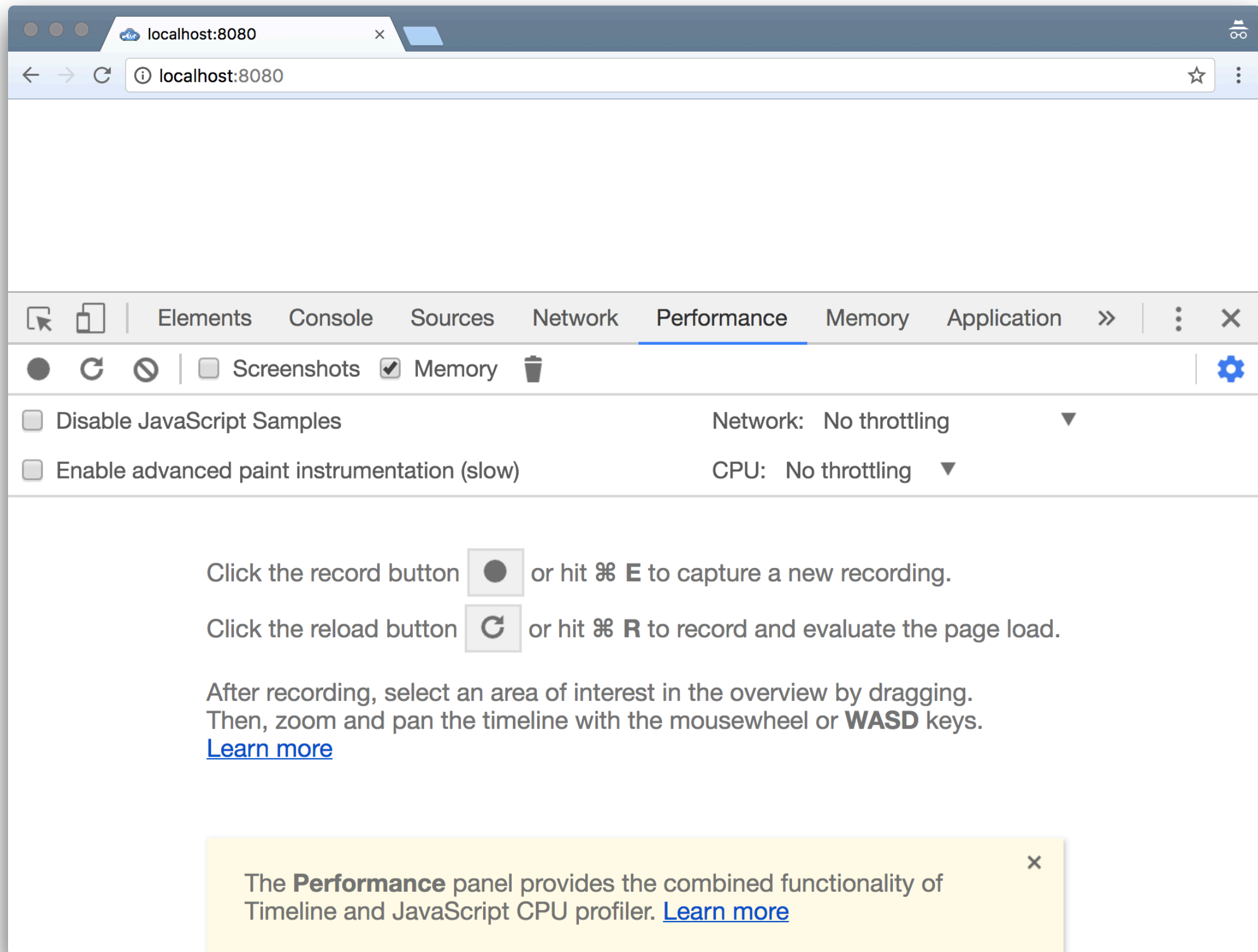


Почему тормозит память

- чтение из переполненной памяти (*большие объемы данных*)
- сборка мусора (*происходит в произвольные моменты*)
- утечки



Профилирование



The image shows a browser window with the address bar set to localhost:8080. The Chrome DevTools interface is open, with the Performance panel selected. The panel includes a toolbar with buttons for recording (a solid black circle), reloading (a circular arrow), and pausing (a circle with a slash). Below the toolbar are checkboxes for 'Screenshots' (unchecked) and 'Memory' (checked). There are also settings for 'Disable JavaScript Samples' (unchecked), 'Enable advanced paint instrumentation (slow)' (unchecked), 'Network: No throttling', and 'CPU: No throttling'. The main content area contains instructions on how to use the record and reload buttons, and a link to learn more. A yellow tooltip at the bottom explains that the Performance panel combines the functionality of the Timeline and JavaScript CPU profiler.

localhost:8080


localhost:8080


Elements Console Sources Network **Performance** Memory Application >> ⋮ ✕

● ↻ ⓧ | Screenshots Memory 🗑️ | ⚙️

Disable JavaScript Samples Network: No throttling ▼

Enable advanced paint instrumentation (slow) CPU: No throttling ▼

Click the record button  or hit ⌘ E to capture a new recording.

Click the reload button  or hit ⌘ R to record and evaluate the page load.

After recording, select an area of interest in the overview by dragging.
Then, zoom and pan the timeline with the mousewheel or **WASD** keys.
[Learn more](#)

The **Performance** panel provides the combined functionality of Timeline and JavaScript CPU profiler. [Learn more](#)



Тестовый DOM на 10 000 элементов

The screenshot shows a web browser window at localhost:8080. The page content consists of a grid of colored rectangles. A tooltip over one of the rectangles shows its dimensions: `div.layout | 972 x 174338`.

The developer tools are open, showing the DOM tree and the style panel for a selected element.

DOM Tree:

```
<html>
  <head>...</head>
  <body> == $0
    <div class="layout">
      <div style="background-color: rgb(247, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(252, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(237, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(173, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(155, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(136, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(40, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(126, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(158, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(121, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(36, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(122, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(107, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(122, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(86, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(173, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(155, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(150, 128, 0);">1492135593554</div>
      <div style="background-color: rgb(102, 128, 0);">1492135593554</div>
    </div>
  </body>
</html>
```

Style Panel:

```
element.style {
}

body {
  display: block;
  margin: 8px;
}
```

Diagram:

The diagram illustrates the box model for the selected element. The element has a width of 972 and a height of 174338. It has a margin of 8px, a border, and padding.



```
function* getPageSwitcher() {
  while (true) {
    yield function* (pageSize) {
      let i = pageSize;
      while (i--) {
        const el = document.createElement('div');
        el.textContent = Date.now();
        el.style.backgroundColor = `rgb(${Math.floor(Math.random()
yield el;
      }
    }
  }
}
```



Утечка памяти



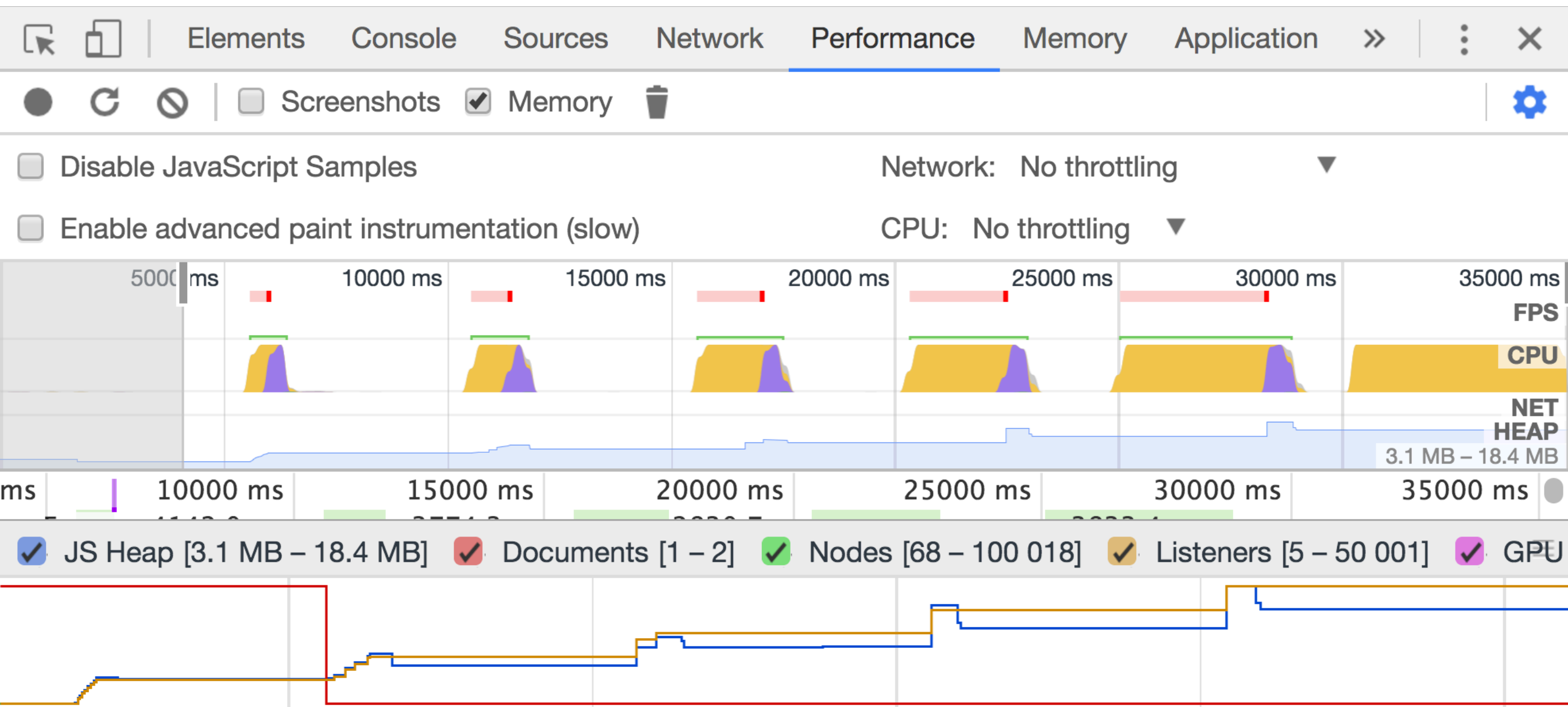
```
const switchPage = () => {  
  container.innerHTML = '';  
  
  for (const el of pageSwitcher.next().value(10000)) {  
    document.addEventListener('click', evt => {  
      evt.stopPropagation();  
      console.log(el, Math.random());  
    });  
    container.appendChild(el);  
  }  
};  
  
document.onclick = switchPage;
```



```
const switchPage = () => {  
  container.innerHTML = '';  
  
  for (const el of pageSwitcher.next().value(10000)) {  
    document.addEventListener('click', evt => {  
      evt.stopPropagation();  
      console.log(el, Math.random());  
    });  
    container.appendChild(el);  
  }  
};  
  
document.onclick = switchPage;
```



Утечка памяти



Сборка мусора



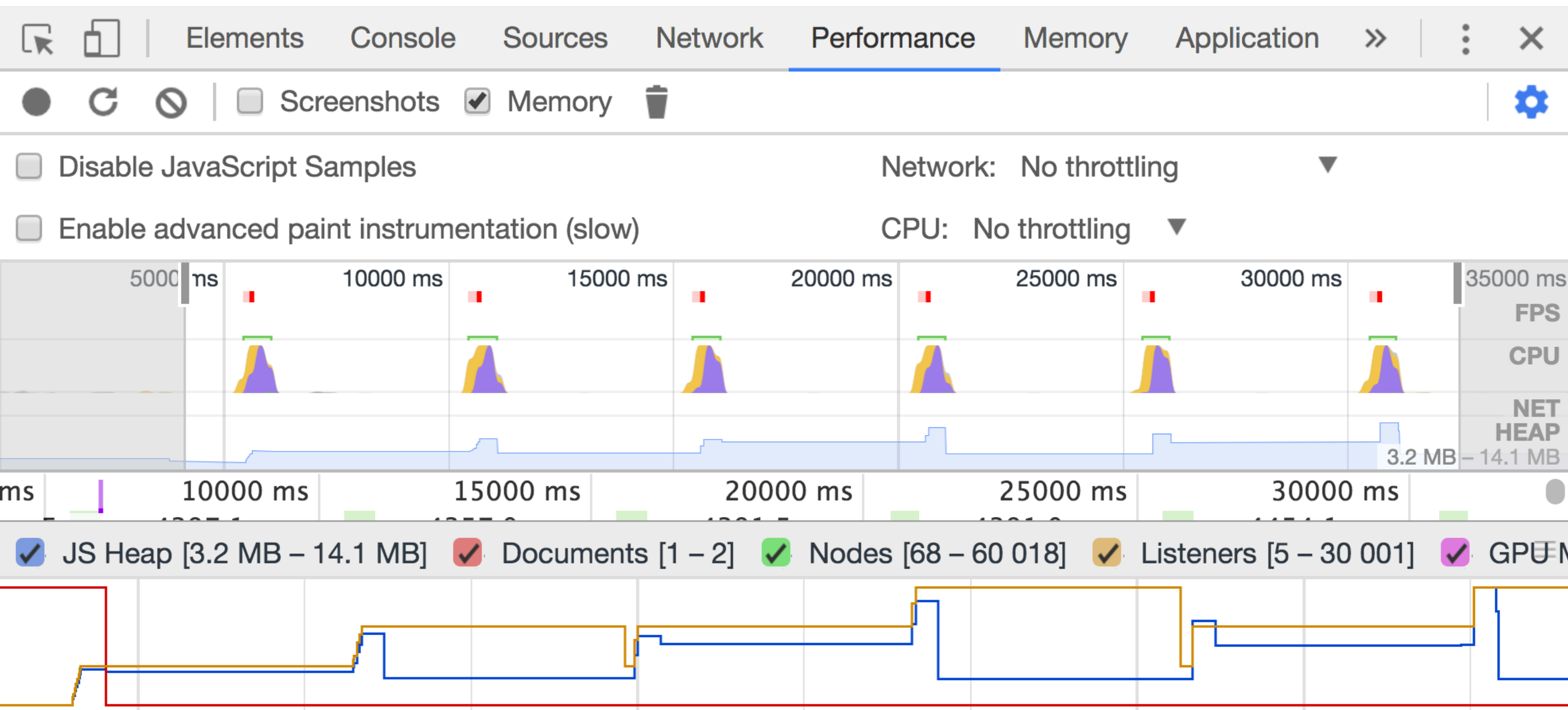
```
const switchPage = () => {  
  container.innerHTML = '';  
  
  for (const el of pageSwitcher.next().value(10000)) {  
    el.addEventListener('click', evt => {  
      evt.stopPropagation();  
      console.log(el, Math.random());  
    });  
    container.appendChild(el);  
  }  
};  
  
document.onclick = switchPage;
```



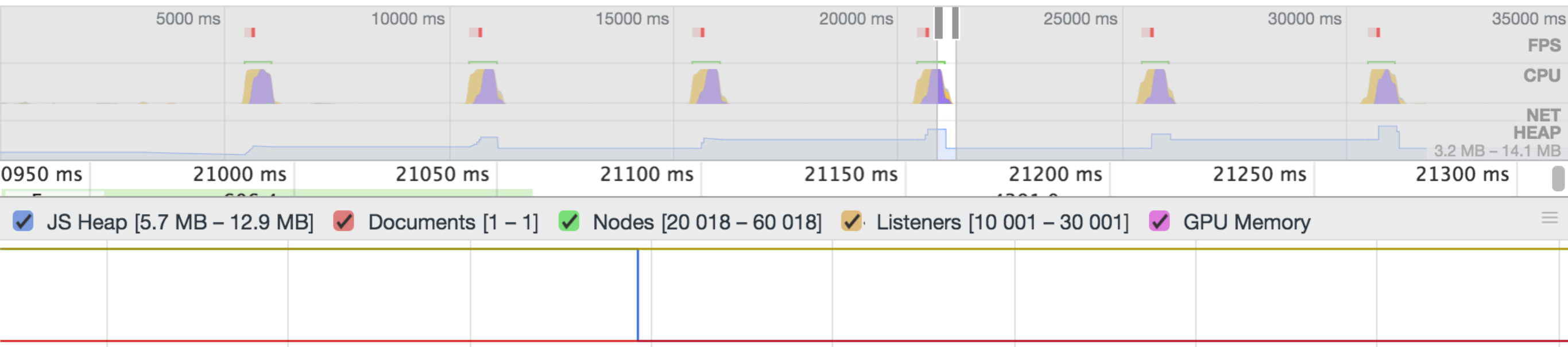

```
const switchPage = () => {  
  container.innerHTML = '';  
  
  for (const el of pageSwitcher.next().value(10000)) {  
    el.addEventListener('click', evt => {  
      evt.stopPropagation();  
      console.log(el, Math.random());  
    });  
    container.appendChild(el);  
  }  
};  
  
document.onclick = switchPage;
```



Сборка мусора



Сборка мусора 15 мсек (1 кадр 60 FPS)



JS Heap [5.7 MB - 12.9 MB] Documents [1 - 1] Nodes [20 018 - 60 018] Listeners [10 001 - 30 001] GPU Memory

Summary Bottom-Up Call Tree **Event Log**

Filter All Loading Scripting Rendering Painting

Start Time	Self Time	Total Time	Activity
21057.9 ms	0.2 ms	0.2 ms	Hit Test
21067.9 ms	15.0 ms	15.0 ms	Major GC
21083.0 ms	8.5 ms	8.5 ms	DOM GC
21091.7 ms	6.7 ms	6.7 ms	DOM GC

Select item for details.



...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 Айтк, 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



Процессор



Как ускорить работу процессора



Как ускорить работу процессора

- записать результаты в память



Как ускорить работу процессора

- ~~• записать результаты в память~~



Как ускорить работу процессора

- ~~• записать результаты в память~~
- уменьшить объем вычислений



Как ускорить работу процессора

- ~~• записать результаты в память~~
- уменьшить объем вычислений
- затротлить




Как ускорить работу процессора

- ~~• записать результаты в память~~
- уменьшить объем вычислений
- затротлить



Как ускорить работу процессора

- ~~записать результаты в память~~
- уменьшить объем вычислений
- затротлить 
- воспользоваться другими инструментами расчета (отдать на видеокарту)



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 10 000 элементов

The screenshot shows a web browser window at localhost:8080 displaying a page with 10,000 elements. The elements are represented as colored rectangles in a grid. The Chrome DevTools Performance tab is open, showing a timeline of the page load. The total load time is 618.8 ms. The performance breakdown is as follows:

Category	Time (ms)
Loading	5.1
Scripting	249.2
Rendering	379.4

The Performance tab also shows various settings and metrics:

- Disable JavaScript Samples:
- Enable advanced paint instrumentation (slow):
- Network: No throttling
- CPU: No throttling
- NET HEAP: 6.1 MB - 11.2 MB
- FPS: 80

The Performance tab is currently set to 'Summary' view, with other options like 'Bottom-Up', 'Call Tree', and 'Event Log' available. The range of the performance data is 4 ms - 773 ms.



Первая страница на 68 элементов

The screenshot shows a web browser window at localhost:8080 displaying a page with 68 elements. The elements are arranged in two rows of eight. The first row contains elements with IDs 1492140843173, and the second row contains elements with IDs 1492140843174. The elements are colored in a repeating pattern of green, orange, and brown.

The Chrome DevTools Performance tab is open, showing a timeline from 0 to 3.10 seconds. The timeline includes a 'Flames' section and a 'NET HEAP' section. The 'NET HEAP' section shows a memory usage of 7.5 MB - 7.9 MB. The 'Flames' section shows a total time of 1.9 ms for Loading, 18.6 ms for Scripting, and 6.7 ms for Rendering.

Performance settings are visible, including 'Disable JavaScript Samples', 'Enable advanced paint instrumentation (slow)', 'Network: No throttling', and 'CPU: No throttling'. The 'JS Heap' section is expanded, showing 'Documents', 'Nodes', 'Listeners', and 'GPU Memory'.

Summary: Bottom-Up Call Tree Event Log

Range: 0 - 3.10 s

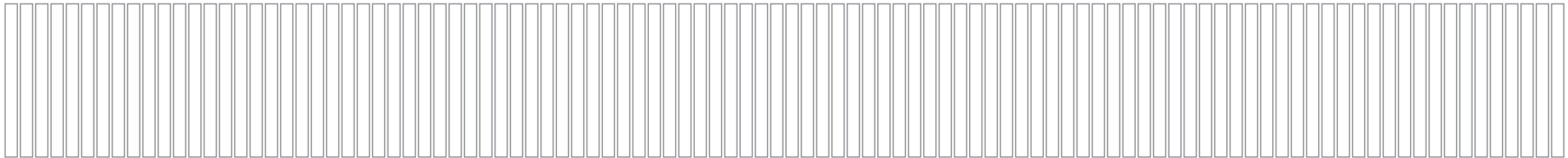
- 1.9 ms Loading
- 18.6 ms Scripting
- 6.7 ms Rendering



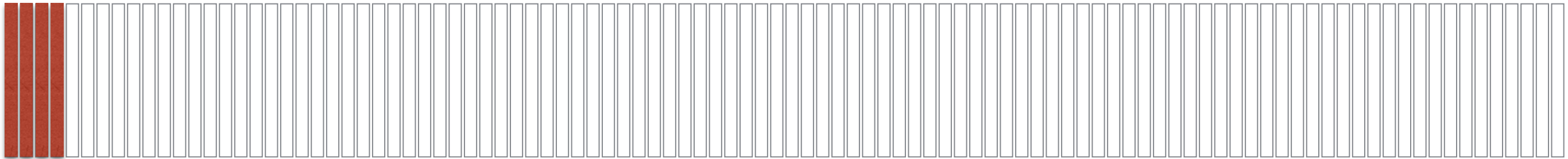
Тротлинг



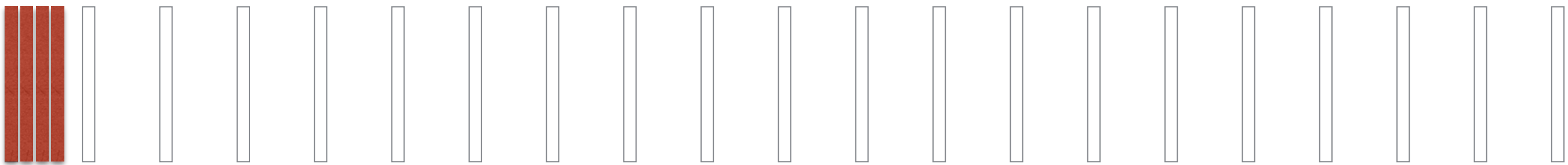
Тротлинг (пропуск кадров)



Тротлинг (пропуск кадров)



Тротлинг (пропуск кадров)



```
window.onscroll = (evt) => {  
    if (document.body.scrollTop + window.innerHeight ===  
        document.body.scrollHeight) {  
        switchPage();  
    }  
};
```



```
let prevPosition = document.body.scrollTop;
let prevComparison = 0;
let evtCounter = 0;

window.onscroll = (evt) => {
  const now = Date.now();

  console.log(`Time shift between scroll is ${now - prevComparison}`);
  console.log(`Delta is ${document.body.scrollTop - prevPosition}`);
  console.log(`Events happen ${evtCounter++}`);
  console.log(`-----`);

  prevPosition = document.body.scrollTop;
  prevComparison = now;
};
```



600px прокрутки – 10 событий

The image shows a browser window at localhost:8080 displaying a grid of 24 colored squares arranged in 3 rows and 8 columns. The colors of the squares vary, including shades of green, olive, orange, and brown. Below the grid, the Chrome DevTools console is open, showing 10 log entries. The console entries are as follows:

Log Entry	Source
Events happen 6	main.js:40
-----	main.js:41
Time shift between scroll is 16	main.js:38
Delta is 34. Viewport is 674	main.js:39
Events happen 7	main.js:40
-----	main.js:41
Time shift between scroll is 17	main.js:38
Delta is 63. Viewport is 674	main.js:39
Events happen 8	main.js:40
-----	main.js:41
Time shift between scroll is 34	main.js:38
Delta is 65. Viewport is 674	main.js:39
Events happen 9	main.js:40
-----	main.js:41
Time shift between scroll is 16	main.js:38
Delta is 41. Viewport is 674	main.js:39
Events happen 10	main.js:40
-----	main.js:41




```
let prevComparison = Date.now();
const throttleInterval = 100;

window.onscroll = (evt) => {
  const now = Date.now();

  if (now - prevComparison ≥ throttleInterval) {
    if (document.body.scrollTop + window.innerHeight ≡
        document.body.scrollHeight) {
      switchPage();
    }

    prevComparison = now;
  }
};
```



Тротл: 1 проверка на 5–6 событий

The screenshot shows a web browser window at localhost:8080. The page content consists of a 3x8 grid of colored squares. The colors of the squares in each row are: Row 1: Orange, Green, Green, Olive, Green, Olive, Green, Green. Row 2: Orange, Green, Orange, Green, Green, Olive, Green, Green. Row 3: Orange, Olive, Orange, Green, Green, Orange, Olive, Olive. The browser's developer console is open, showing a list of events. The events are: Scroll evt (main.js:36), Complex check (main.js:39), 5 Scroll evt (main.js:36), Complex check (main.js:39), 6 Scroll evt (main.js:36), Complex check (main.js:39), 6 Scroll evt (main.js:36), Complex check (main.js:39), 6 Scroll evt (main.js:36), Complex check (main.js:39), 6 Scroll evt (main.js:36), Complex check (main.js:39), 5 Scroll evt (main.js:36), Complex check (main.js:39), 3 Scroll evt (main.js:36). The numbers in blue circles (5, 6, 6, 6, 6, 6, 5, 3) indicate the throttling rate: one check for every 5-6 scroll events.



Отдать на видеокарту



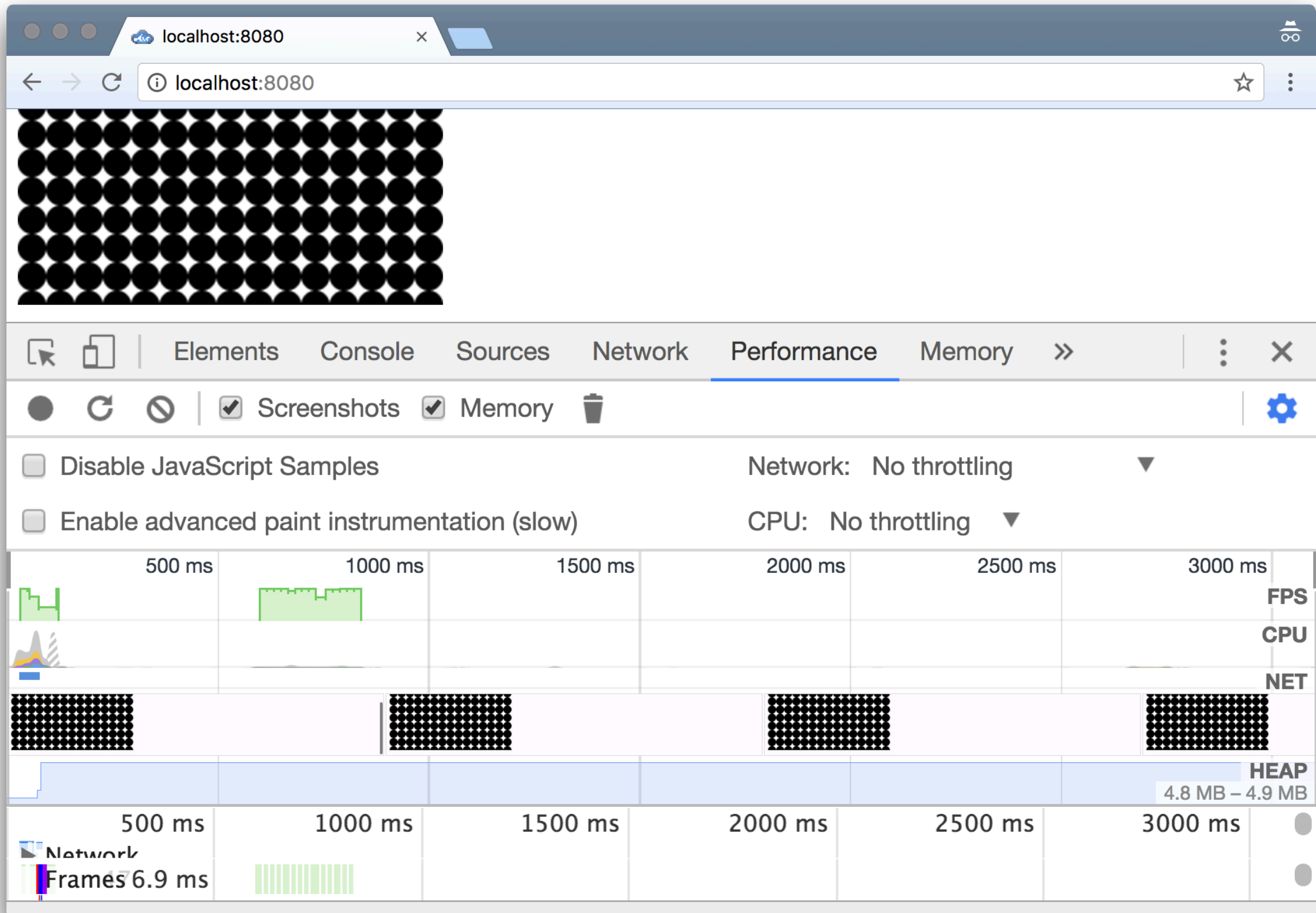
**Браузерные игры делают на
канвасе, а не на SVG**



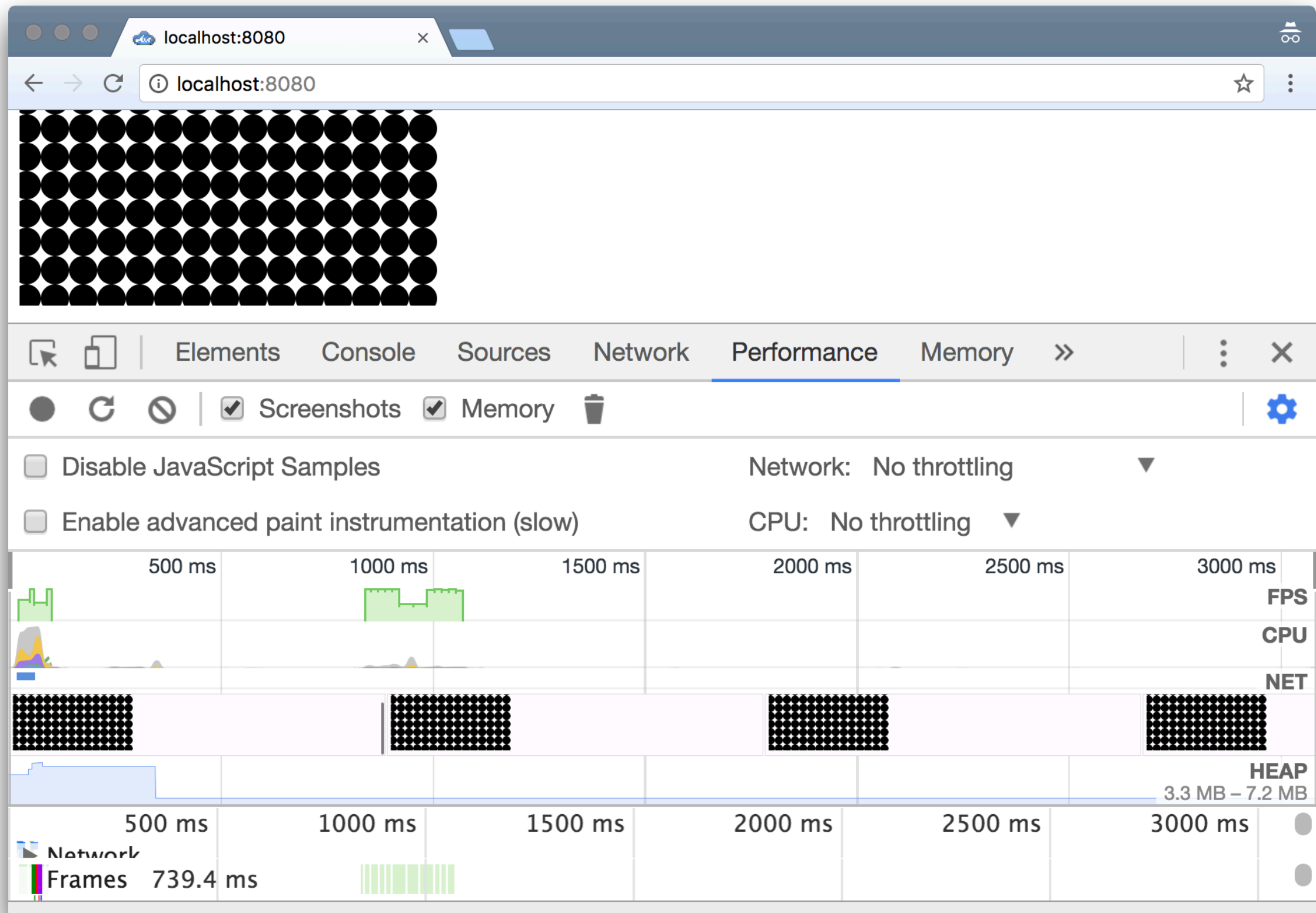
Заполним шариками объём



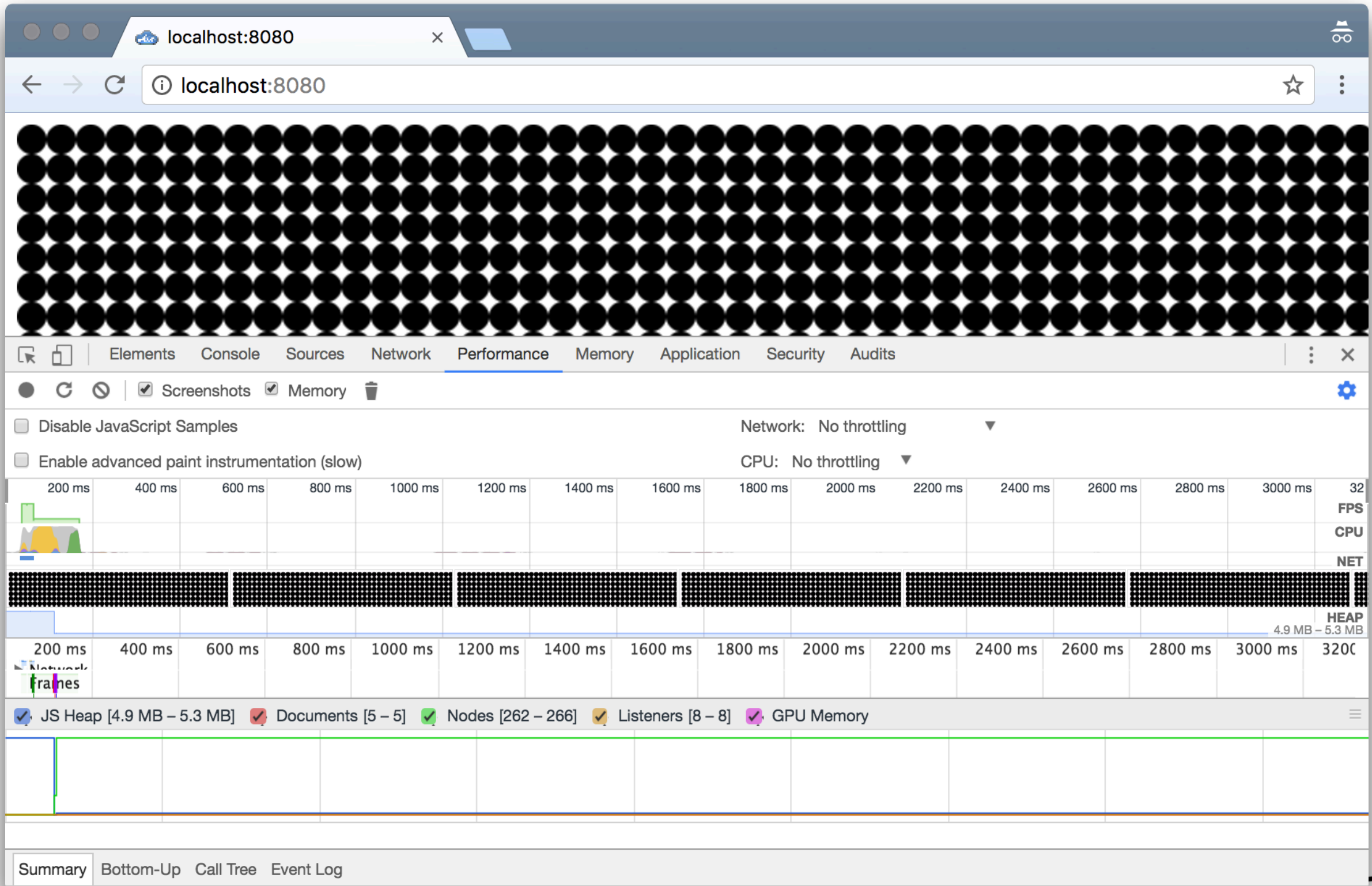
canvas 300x150 ~100 mcek



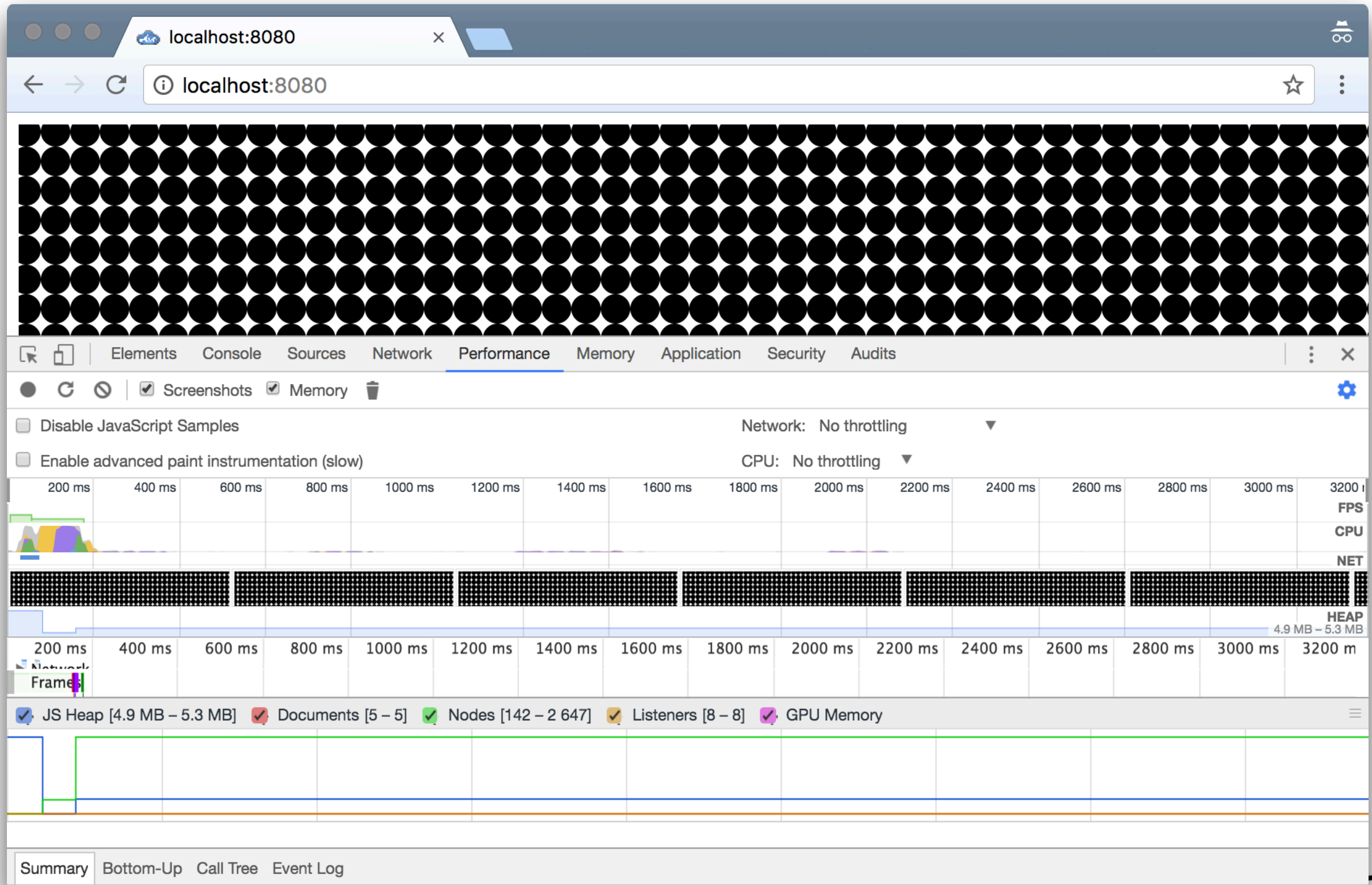
SVG 300x150 ~100 mcek



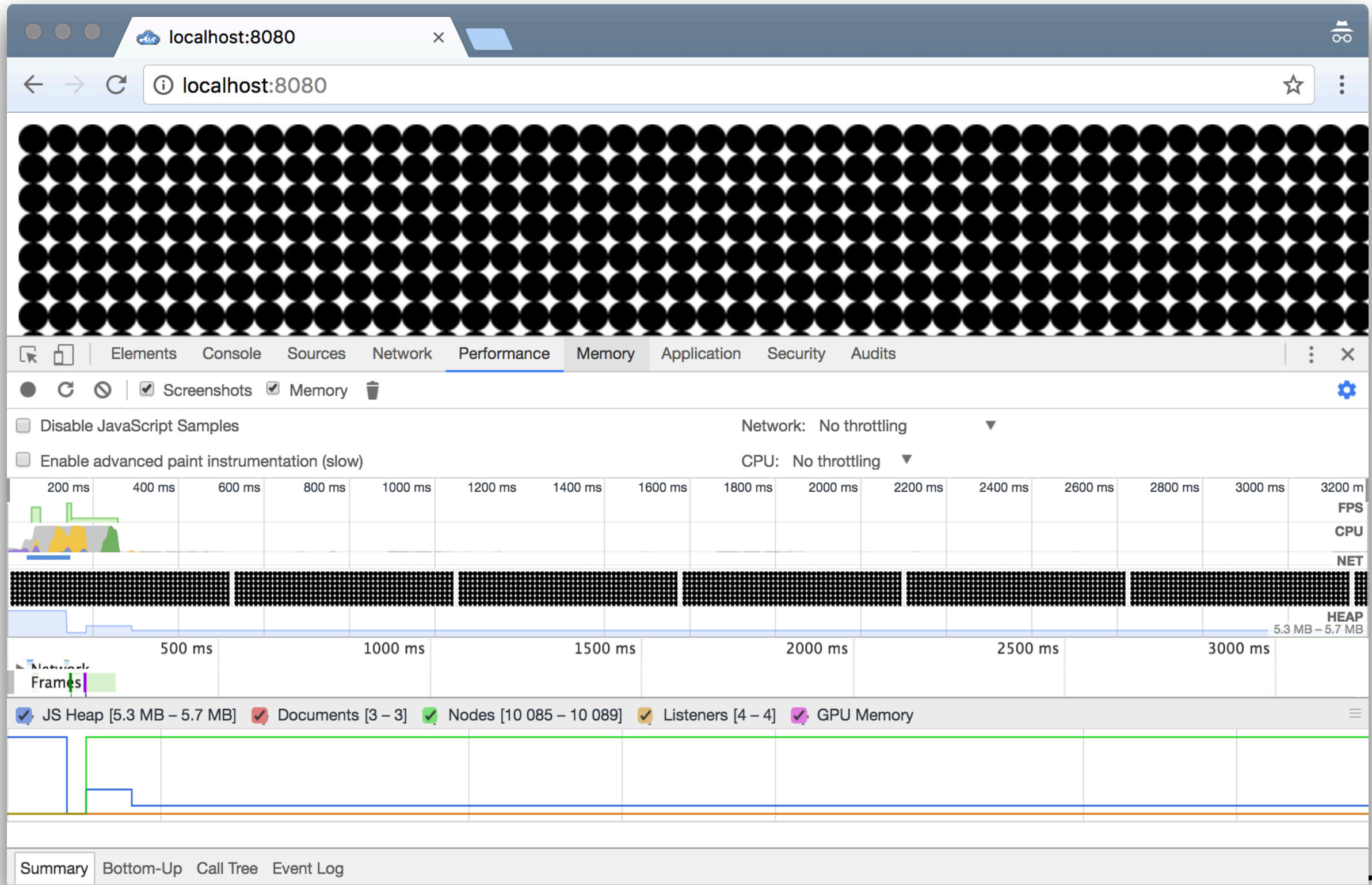
canvas 1000x1000 ~180 MCEK



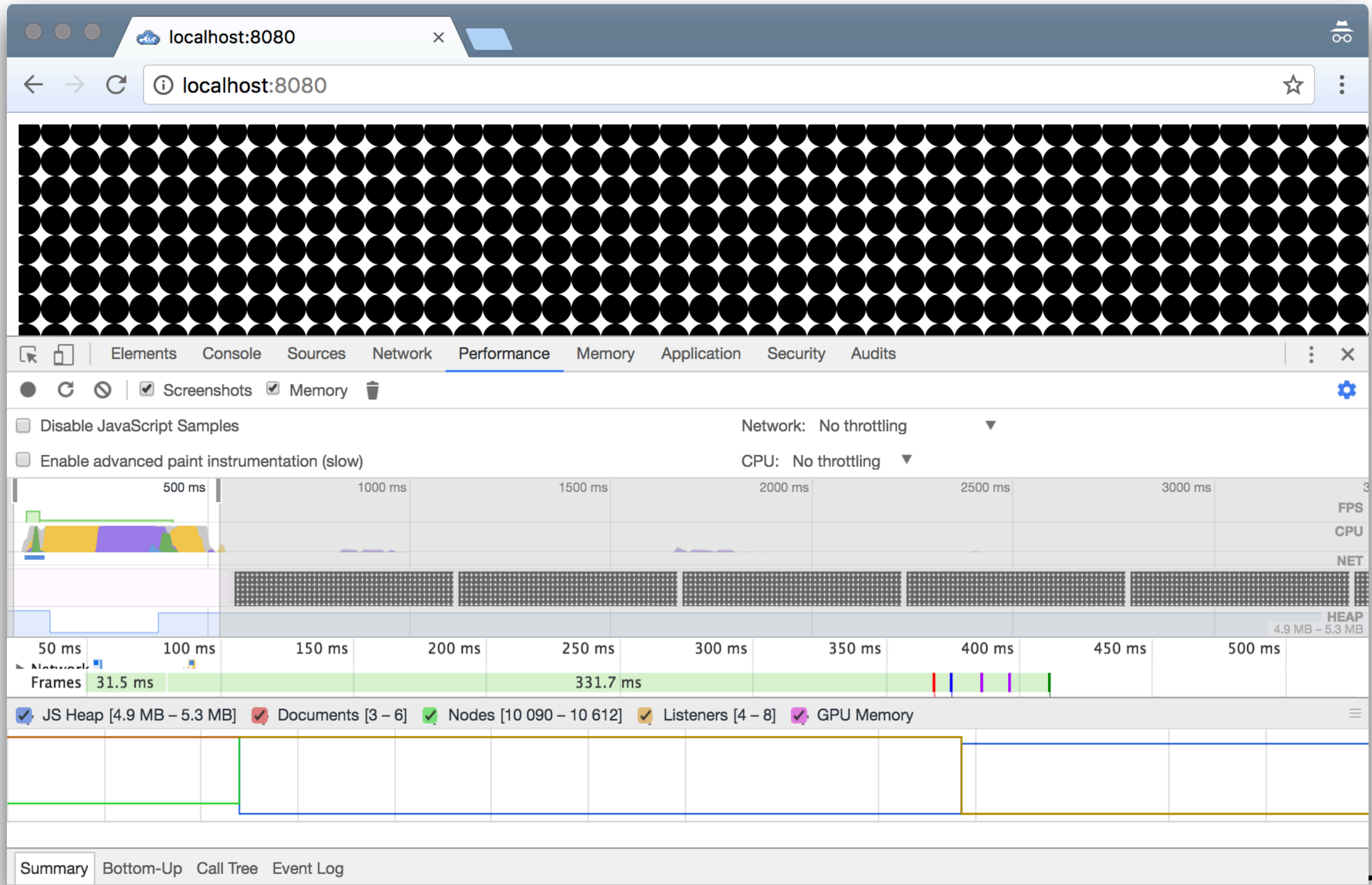
SVG 1000x1000 ~200 msec



canvas 2000x2000 ~300 msec



SVG 2000x2000 ~ 1/2 cek



Вы можете заменить d3js 🤔



**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



100



Обратная связь курильщика



101



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



100

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

Очень остроумный коммент |

OK



Обратная связь курильщика: 2

Месть обратной связи

Очень остроумный коммент |



Клик!

Обратная связь курильщика: 2

Месть обратной связи

Очень остроумный коммент |

OK



Обратная связь курильщика: 2

Месть обратной связи

Очень остроумный коммент |

OK

– Клик!

Обратная связь курильщика: 2

Месть обратной связи

Очень остроумный коммент |

OK



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 коммент 👎

Очень остроумный коммент 👎

GET <http://localhost/> 204

GET <http://localhost/> 204

Обратная связь здорового человека: 2

Новая надежда

Очень остроумный коммент |

OK



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. использовать все возможные ресурсы для расчётов



Алгоритм оптимизации

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



Алгоритм оптимизации

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



Алгоритм оптимизации

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



 **Спасибо!**



o0.github.io

латинская «о», ноль

