

High performance web applications

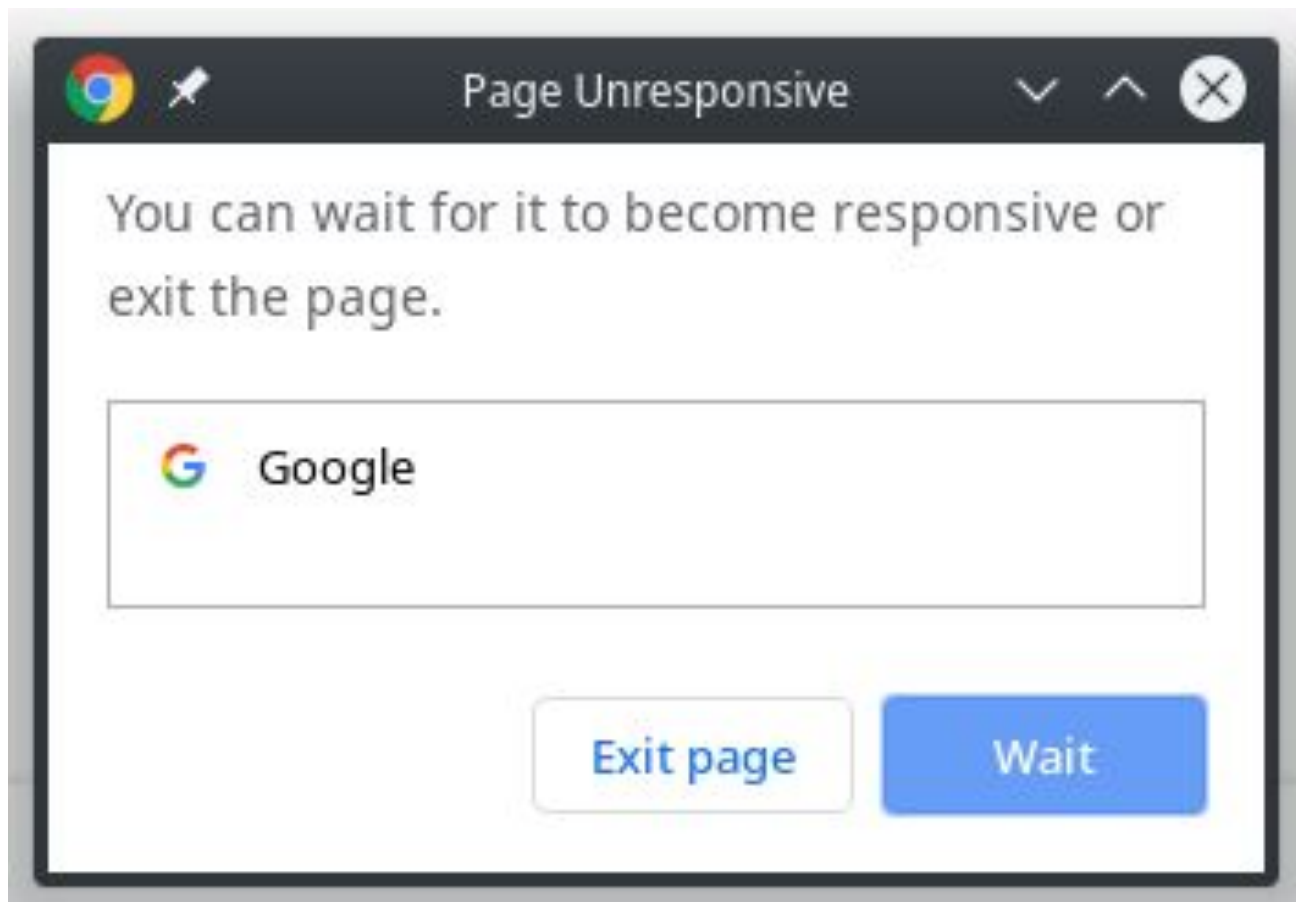
О авторе

Затравкин Иван

- Lead fullstack developer в SEMrush
- Профессионально занимаюсь разработкой последние ~7 лет
- High-performance computing интересовался в студенческие годы для физических симуляций
- Выжать максимальную скорость - всегда увлекательный сложный пазл <3



Почему производительность важна?



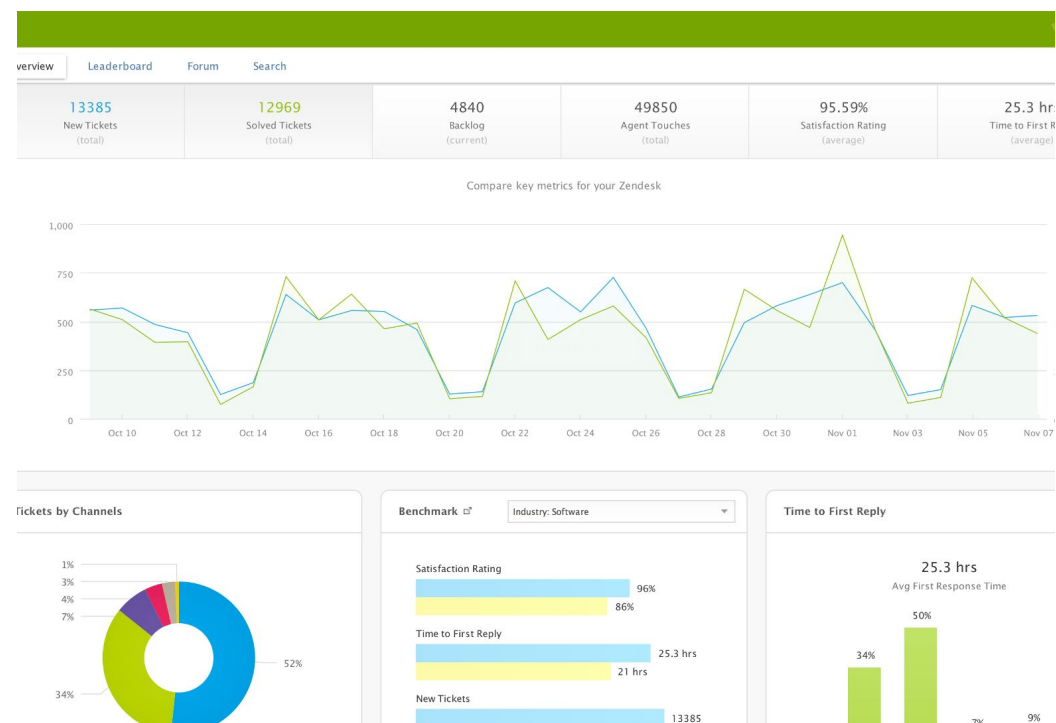
Почему производительность важна?

Yahoo - A Guide to WWW

[[What's New?](#) | [What's Cool?](#) | [What's Popular?](#) | [Stats](#) | [A Random Link](#)]

[Y Top](#) | [Up](#) | [Search](#) | [Mail](#) | [Add](#) | [Help](#)

- [Art\(466\)](#) NEW
- [Business\(6426\)](#) NEW
- [Computers\(2609\)](#) NEW
- [Economy\(743\)](#) NEW
- [Education\(1487\)](#) NEW
- [Entertainment\(6199\)](#) NEW
- [Environment and Nature\(193\)](#) NEW
- [Events\(53\)](#) NEW
- [Government\(1031\)](#) NEW
- [Health\(367\)](#) NEW
- [Humanities\(163\)](#) NEW
- [Law\(163\)](#) NEW
- [News\(185\)](#)



Что значит “быстро”?

- $>0.1s$ - “моментально”
- $0.1-1s$ - не прерывает работу пользователя, но нужен индикатор
- $1-10s$ - максимальная продолжительность концентрации внимания
- $>10s$ потеря интереса к происходящему

Зачем нужно “быстро”?

- Согласно A/B тестам Amazon, увеличение времени загрузки страницы на **100ms** снижает revenue на **1%**
- Согласно A/B тестам Microsoft, снижение времени загрузки Bing на **100ms** снижает revenue на **0.6%**

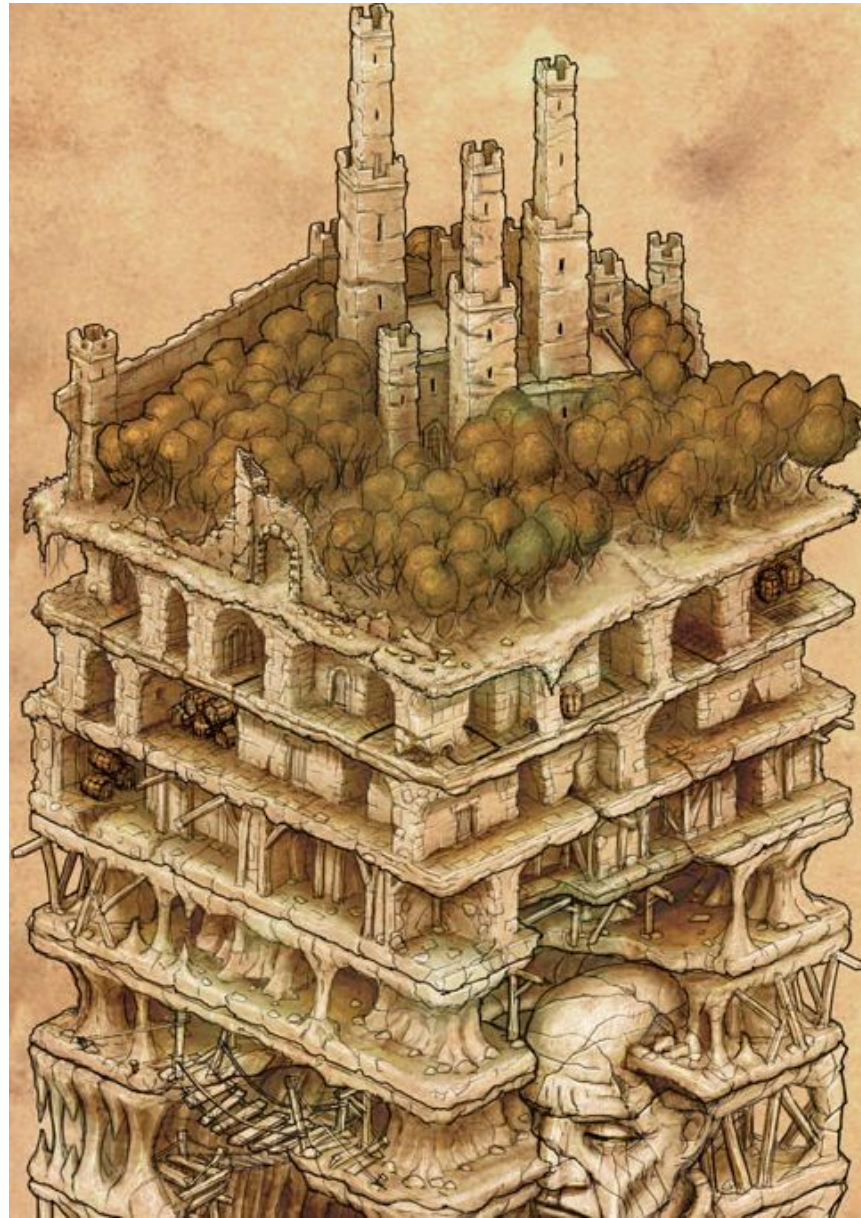
Источники: <http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html>
<https://exp-platform.com/Documents/2014%20experimentersRulesOfThumb.pdf>

Быстро на Frontend

- Реакция на действие пользователя - до **100ms**
- Визуальная плавность интерфейса - поток исполнения не должен быть занят дольше **16.6ms** (60 fps)

Roadmap техник и инструментов

Просто, быстро,
эффективно



Сложно, непонятно,
но очень интересно

Perceived performance

Самое простое, что можно сделать - не делать ничего.
Просто добавь спиннер!

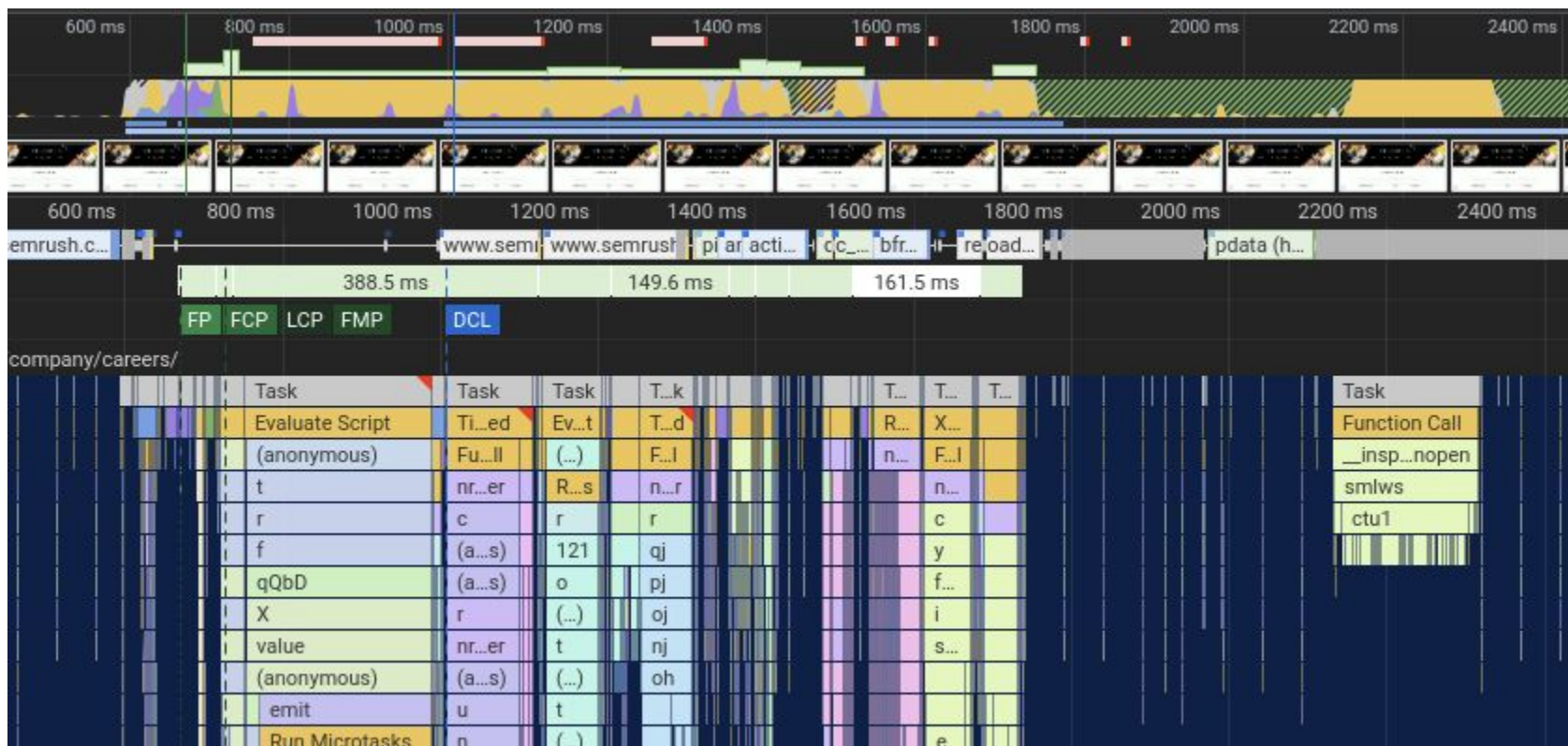


Совет

Анимировать спиннер можно с помощью CSS.

но не все свойства ускоряются железно

Неоптимальный код







HET

Проблема всё-таки есть

technology name

priority value integrated

Next

technology name

priority value integrated

Next

fadfawefewf

Детали реализации

Мемоизация

Чистые функции: результат зависит только от аргументов

При одних и тех же параметрах - один и тот же результат. Кэширование!

Совет

Не стоит запоминать всё подряд,
иногда проще пересчитать.

Детали реализации

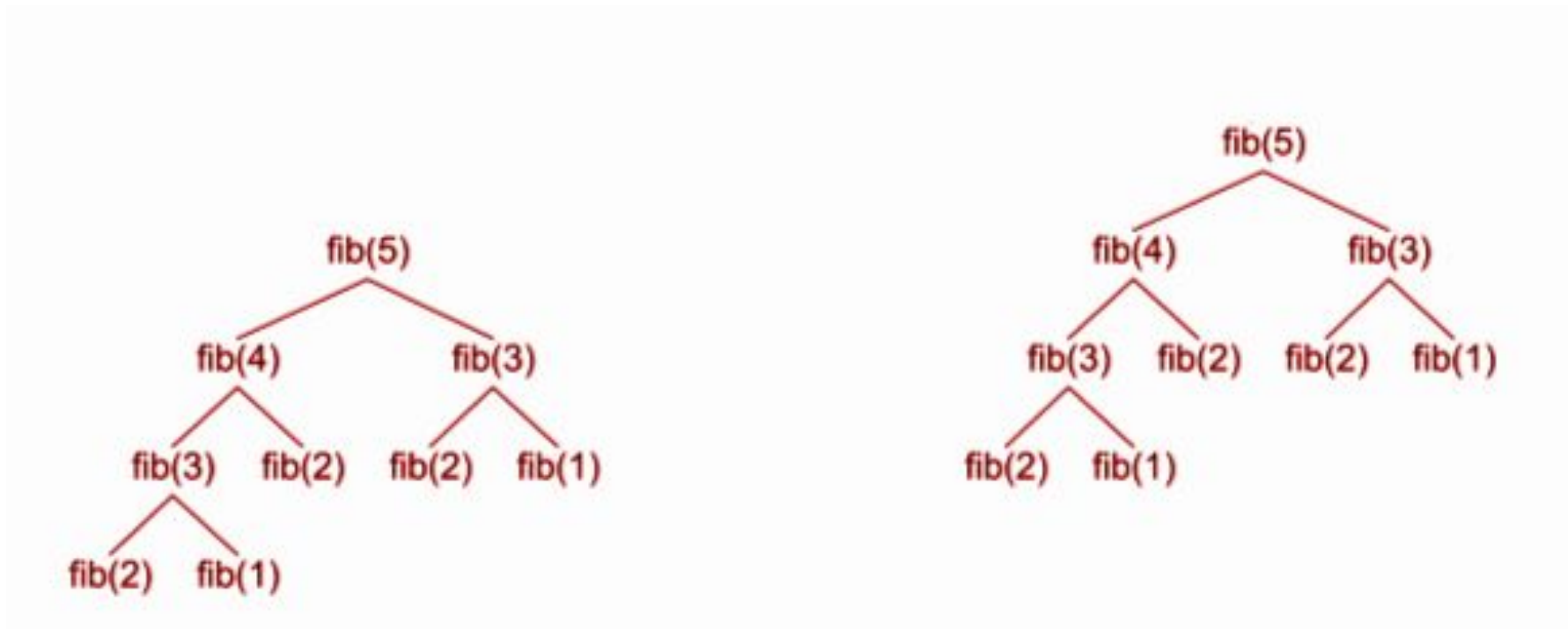
Мемоизация

```
const memoize = fn => {  
  let prevArs;  
  let prevRes;  
  return (...args) => {  
    if (deepEqual(prevArs, args)) {  
      return prevRes  
    } else {  
      prevRes = fn(...args);  
      prevArs = args;  
      return prevRes;  
    }  
  }  
}
```

Детали реализации

Мемоизация

```
const fibonacci = n => n > 1 ? fibonacci(n-1) + fibonacci(n-2) : n
```



Детали реализации

Иммутабельные объекты

Не мутировать объекты!

Неизменяемые объекты:

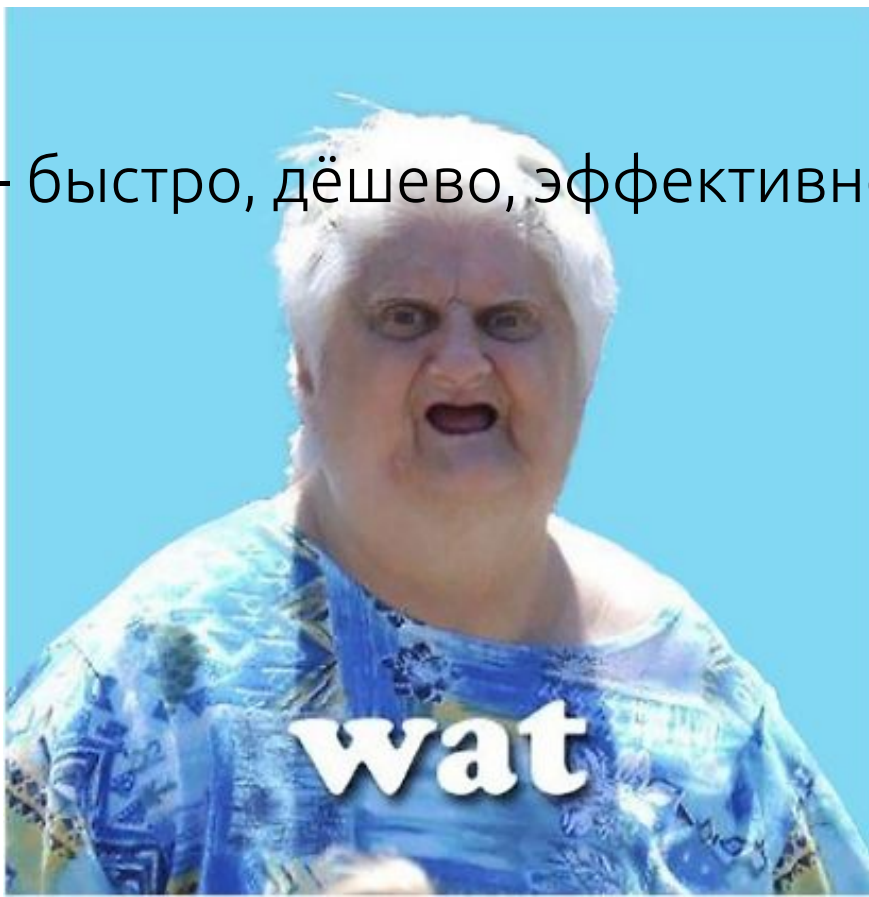
- Сравнение $O(1)$: $x === y$
- Хорошо работают с мемоизацией
- Меньше багов (никто не поменяет ваш объект)

Детали реализации

Мутация объектов

Мутировать объекты!

Мутирование объекта - быстро, дешево, эффективнее использует память



Детали реализации

Мутация объектов

Обход дерева

```
type ITree = { val: any, left?: ITree, right?: ITree }
```

220 ms

```
const pure = (tree: ITree) => {  
  return [  
    ...(tree.left ? pure(tree.left) : []),  
    tree.val,  
    ...(tree.right ? pure(tree.right) : [])  
  ]  
}
```

70ms

```
const mut = (tree: ITree, res: any[] = []) =>  
{  
  if (tree.left) {  
    mutate(tree.left, res)  
  }  
  res.push(tree.val)  
  if (tree.right) {  
    mutate(tree.right, res)  
  }  
  return res;  
}
```

Всё равно слишком медленно

Прячем проблему под ковёр

Асинхронные вычисления и WebWorker

Timeslicing

```
const data = new Array(1e8)

function reduce(data) {
  //do something with data
}
reduce(data)
```

```
const data = new Array(1e8);
const step = 5e7;
function reduce(data, start, end, acc, resolve) {
  ///do something with data
  if (not_done) {
    setTimeout(reduce(...args))
  } else {
    resolve(result)
  }
}
```

Прячем проблему под ковёр

Асинхронные вычисления и WebWorker

WebWorker

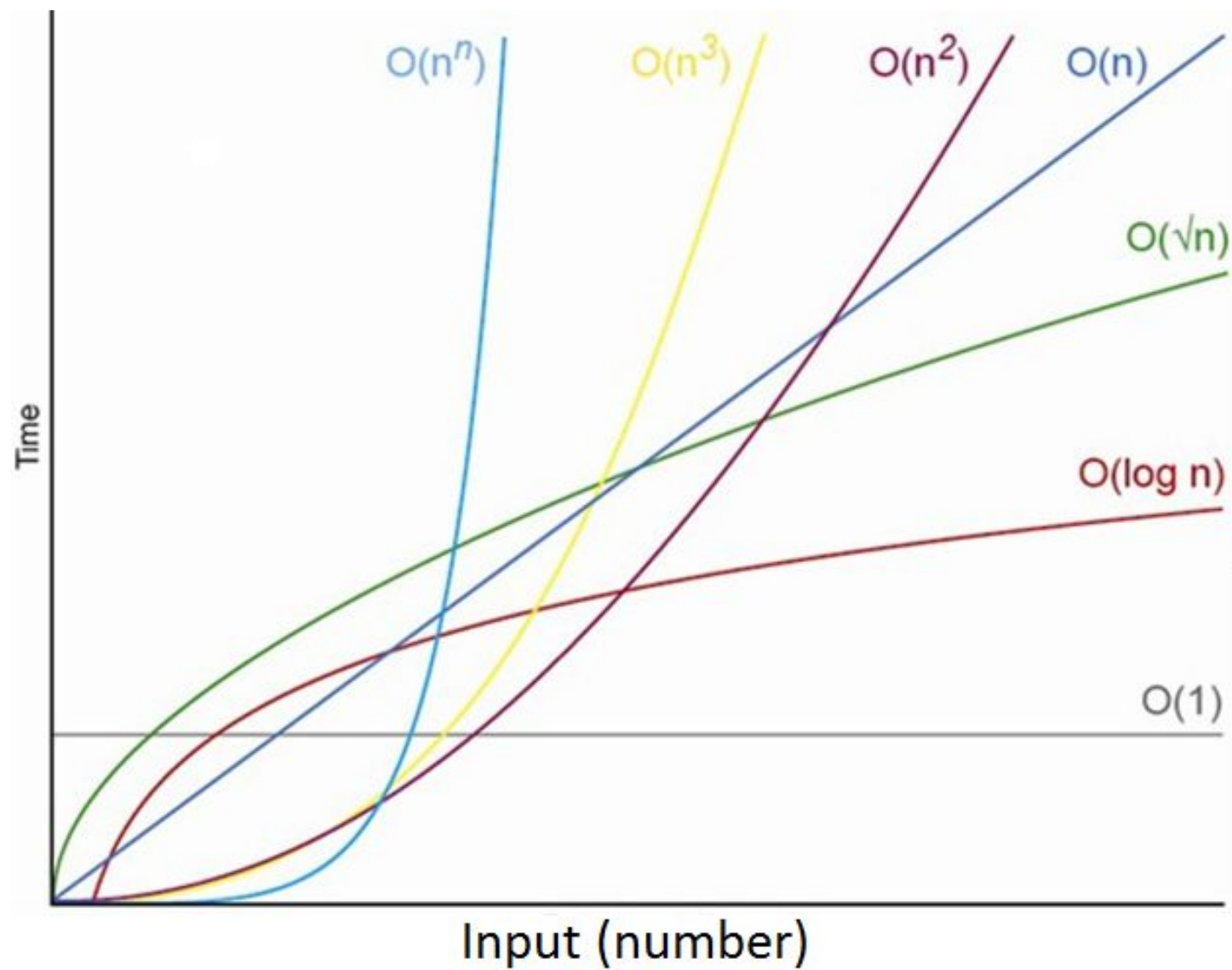
```
const worker = new Worker(source);  
worker.onmessage = e => process(e);  
worker.postMessage(data);
```

Совет

offscreenCanvas может использоваться
из WebWorker

Проблема не уходит :(

Смена алгоритма



Смена алгоритма

Пример #1

База данных продаж

Все проданные товары в диапазоне цен от 1\$ до M\$

Продаж очень много

Нужно: посчитать объём продаж в диапазоне цен

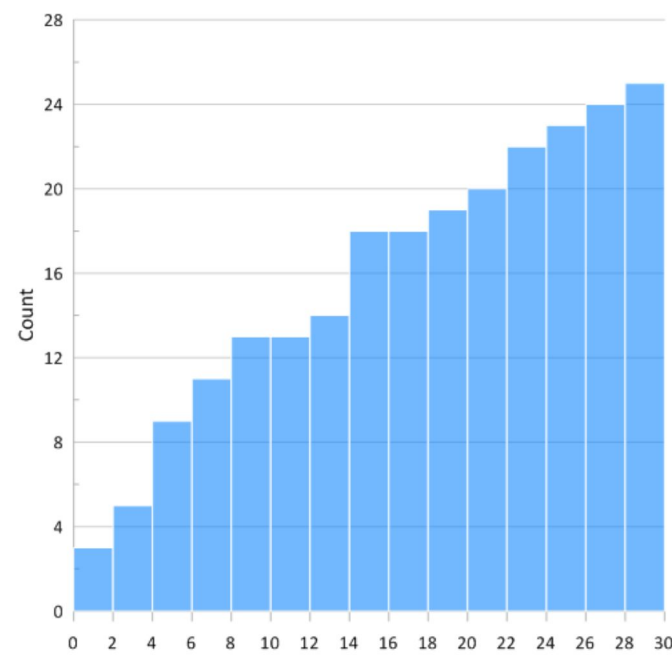
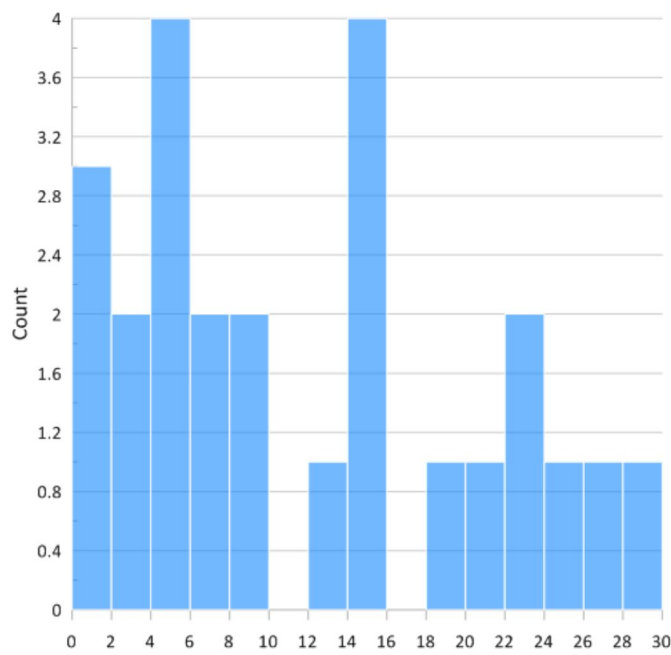
Смена алгоритма

Пример #1

Наивное решение, $O(n)$: обход всех продаж

Лучшее решение, $O(m)$: хранить гистограмму, обходить её

Идеально, $O(1)$: накопительная гистограмма



Смена алгоритма

Пример #2

Поиск по множеству

Наивный подход: сканирование массива, $O(N)$

Отсортированный массив: $O(\log N)$, но вставки $O(N)$

Дерево: $O(\log N)$

Карта ключей, если мы так можем: $O(1)$

Marketing Calendar Best team ever s...












Timezone: UTC +03:00 Europe/Moscow | Now: April 24, 2019 7:11 PM



Calendar Campaigns Activities My Tasks

Day Week **Month** Year

< Jul 2018 > Today

Mon	Tue	Wed	Thu
25	26	27	28
<div>1:00 PM  Soldes ✓ 2/4</div> <div>3:00 PM  Write 3 backlinks press releases</div> <div>3:00 PM  Collect voyage/destinations research ✓ 0/3</div>	<div>3:00 PM  Write a blogpost about Backlink research</div> <div>3:00 PM  Write take aways for "Why?" and "How?" research</div>	<div>3:00 PM  Prepare Soical Media Promo of FIFA (posts from 1/8 final)</div> <div>3:00 PM  Finalize and publish Backlink Research on blog</div>	<div>3:00 PM  Host 1st TwitterChat with @RaphSEO</div> <div>3:00 PM  Publish product article ("How to get ideas for your Content Marketing with SEMrush")</div> <div>3:00 PM  Publish external posts on Medium</div> <div>3:00 PM  Publish external posts on Webmarketing.com</div>

Notifications



Olivier AMICI • 4 months ago

Has joined your "[Blog planning](#)" calendar with read-write permissions.



Natalia • 4 months ago

A status for the activity [80 mistakes of e-commerce article](#) has been changed to ✓ Done



Natalia • 4 months ago

A status for the activity [Publish article with comments to promote the e-book](#) has been changed to ✓ Done



Natalia • 4 months ago

A status for the activity [Discuss and prepare the Top of Blogs redesign with designers and sky team](#) has been changed to 🕒 In progress



Natalia • 4 months ago

A status for the activity [Find a blog columnist for Social Media](#) has been changed to 🕒 In progress

New activity

SV

Filter

Sun

0 PM

/destinations
h

Смена алгоритма

```
type IActivity = { date: string };
const acts IActivity[] = [....];
days.map(day => {
    const aInDay = acts.filter(({date}) => date == day));
    return aInDay.map(a => <Activity {...a}/>)
})
```

```
type IActivity = { date: string };
const acts {[date: string]: IActivity[]} = {....};
days.map(day => {
    const aInDay = acts[day];
    return aInDay.map(a => <Activity {...a}/>)
})
```

Параллелизм в JS

WE NEED TO GO FASTER!

Параллелизм в JS

Основные инструменты:

- **WebWorker** - создание тредов
- **SharedArrayBuffer** и **TypedArray** - разделяемая память
- **Atoms** - синхронизация тредов и потоковая безопасность

Параллелизм в JS

WebWorker

```
const workers = [];  
for (let i = navigator.hardwareConcurrency; i > 0; i++) {  
    const worker = new Worker(source);  
    worker.onmessage = e => process(e);  
    workers.push(worker);  
}
```


Параллелизм в JS

SharedArrayBuffer и TypedArray

Поддерживаемые типы массивов:

```
Int8, Int16, Int32, Float32, Float64, Int64  
const int8Arr = new Int8Array(1000)
```

SharedArrayBuffer - разделяемая между потоками память*

*Пока только в Chrome и Node.js

Параллелизм в JS

Atoms

```
const futexes = new Int32Array(new SharedArrayBuffer(4))  
Atoms.wait(futexes, 0, val)
```

ЕЩЁ БЫСТРЕЕ!

Микрооптимизации!

```
const slow = (x, n) => Math.ceil(x/(2**n))
```

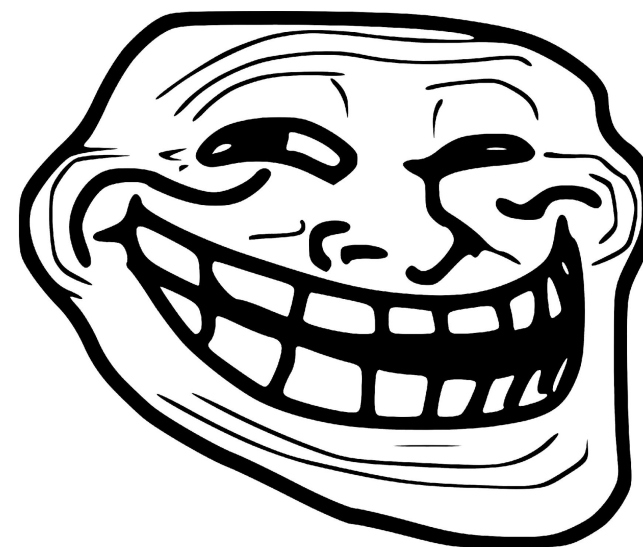
```
const fast = (x, n) => (x >> n) + ((x & (1 << n) - 1)) === 0 ? 0 : 1)
```

129 ms vs 26 ms

Микрооптимизации!

Второй прогон:

26 ms vs 26 ms



Микрооптимизации!

На что можно обратить внимание:

- ~~`if (slow_function() || a > b)`~~ -> `if (a > b || slow_function())`
- какие объекты попадут в closure (garbage collector)
- `requestAnimationFrame` для анимаций
- `TypedArray` для больших массивов

STILL WANNA FASTER!

WebAssembly

Плюсы WebAssembly

- AOT vs JIT
- Множество языков (**и существующего кода**): Go, C, C++, Rust, AssemblyScript(subset TypeScript) и другие
- Близкая к нативной производительность
- Работает на любых платформах

Нет доступа к DOM.

На простом коде менее эффективен, чем JS

WASM

Бенчмарки:

<https://takahirox.github.io/WebAssembly-benchmark/>

Fibonacci: 740ms JS vs 297ms WASM

<https://pspdfkit.com/webassembly-benchmark/>

3841ms JS vs 1917ms WASM

А ещё быстрее слабо?

MADNESS? THIS. IS. WEBDEV!

Интринсики

Scalar Operation

$$\begin{array}{l} A_1 \times B_1 = C_1 \\ A_2 \times B_2 = C_2 \\ A_3 \times B_3 = C_3 \\ A_4 \times B_4 = C_4 \end{array}$$

SIMD Operation

$$\begin{array}{l} A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \times \begin{array}{l} B_1 \\ B_2 \\ B_3 \\ B_4 \end{array} = \begin{array}{l} C_1 \\ C_2 \\ C_3 \\ C_4 \end{array}$$

Интринсики

128-битные векторные инструкции

Как это работает: 16 x 8 bit || 8 x 16 bit || 4 x 32 bit || 2 x 64 bit

<https://github.com/WebAssembly/simd/blob/master/proposals/simd/SIMD.md>

Производительность ~ нативным бинарникам.

Пока поддержка только у Chrome

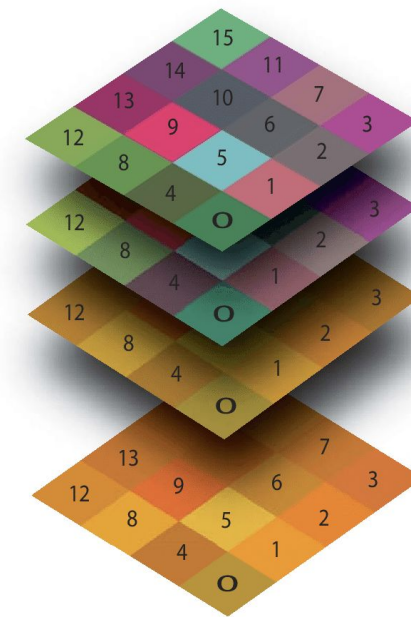
Конец?

GPU Compute

Значительно быстрее CPU на высокопараллельных задачах

Обработка массивов, перемножение матриц, etc

Применение по назначению: для 3D графики



GPU Compute

Основные инструменты: WebGL, shaders, compute shaders

Познакомиться подробнее:

<https://developers.google.com/web/updates/2019/08/get-started-with-gpu-compute-on-the-web>

GPU Compute: ComputeShaders

<https://developers.google.com/web/updates/2019/08/get-started-with-gpu-compute-on-the-web>

То же самое, что предыдущее, но с нормальным API

Есть поддержка TensorFlow - нейросети на клиенте

Конец

Вопросы?