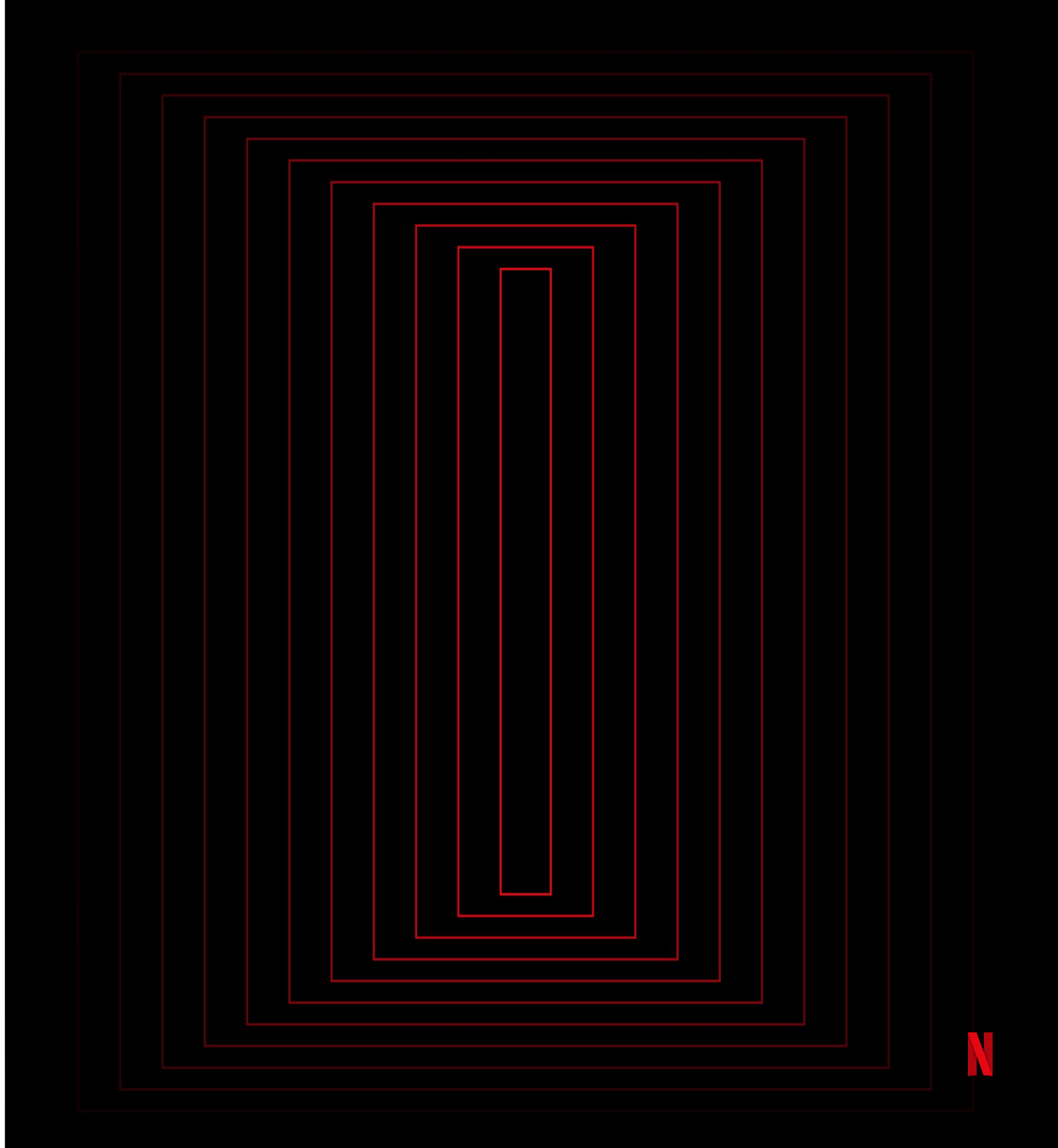
Приложение Netflix
Переезд с JavaScriptCore на V8

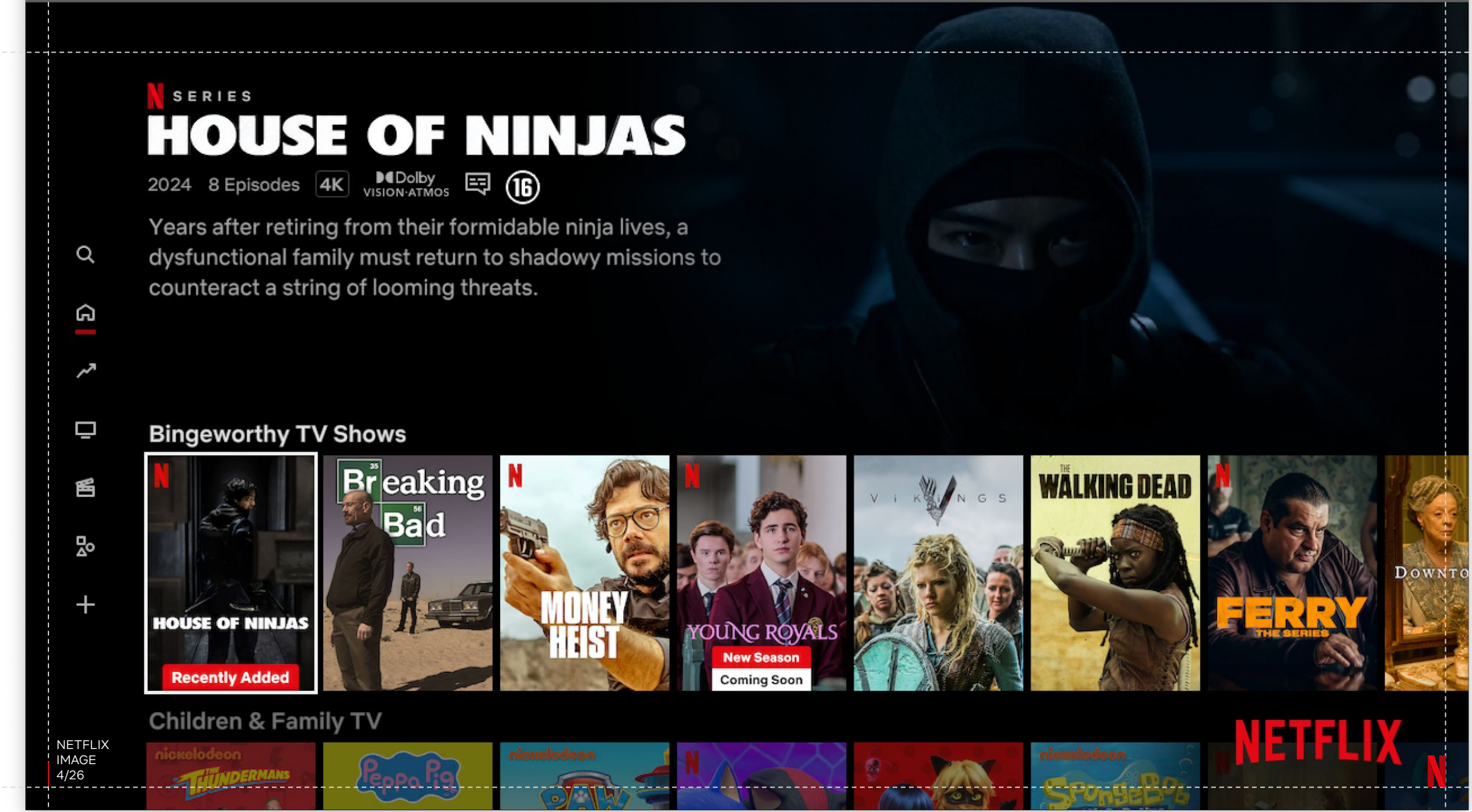
Алексей Козятинский
ak239spb@gmail.com
https://twitter.com/the_kozy

План

— Как устроено приложение Netflix?
— Зачем что-то менять?
— Как быстро исполнять JavaScript?
— Вопрос к аудитории!
— Как эффективно выносить мусор?
— Современные JavaScript движки
— Результаты переезда

Введение





NETFLIX
IMAGE
4/26

Web ≠ TV

— Очень разные устройства
— Отсутствие экосистем
— Медленные устройства
— Другие ожидания пользователей
— Целый подкаст на тему

Свой браузер!

— React Native
— JavaScript - никаких CSS/HTML
— Компактный, но быстрый API
— Свой сетевой стек (два)
— Доступ к железу (DRM и друзья)

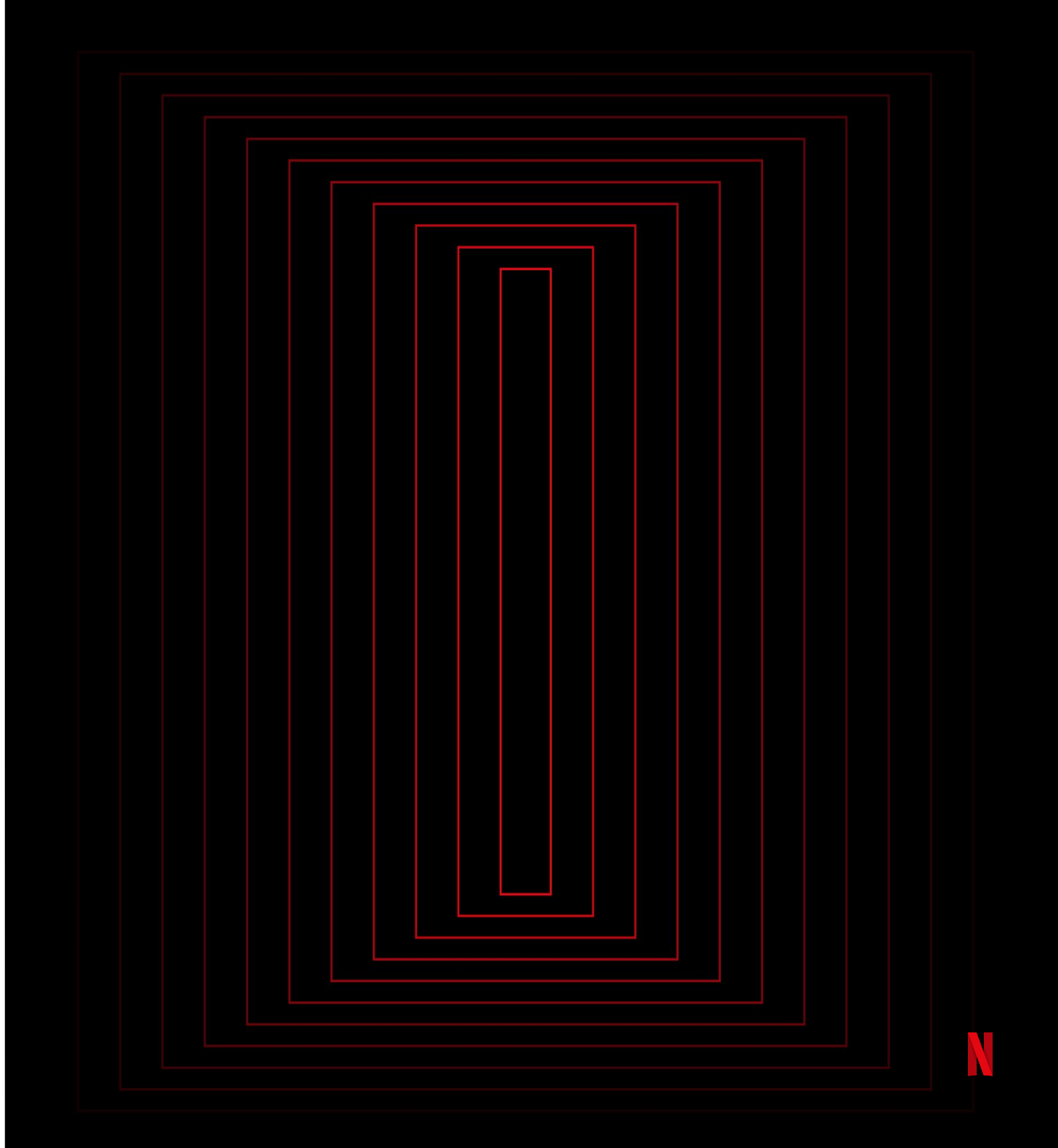
JavaScriptCore (~2014)

- EcmaScript 5
- Интерпретатор
- Сборка мусора при полной остановке
- Минимум встроенных инструментов
- Медленно и прожорливо

Зачем?

— быстрей
— сэкономить память
— стать более гибкими
— инструменты
— современный JavaScript

Трюки JavaScript движков



Как быстрей?

— JavaScript на входе
— больше компилируем - быстрее работает
— больше информации - лучше оптимизируем
— необходимо отбить затраты!

Компиляторы

- JavaScript → байткод
- байткод → машинный код
  - быстро
  - подольше
  - еще дольше
  - бесконечно долго
- NP полная задача

Оптимизации?

```
1   function slow() {
2       const a = { a: 1, b: 2 };
3       const b = { c: 42 };
4       let sum = 0;
5       for (let i = 0; i < 1000; ++i) {
6           sum += a.a;
7           sum += a.b;
8       }
9       const c = { a: 1, b: 2 };
10      return sum + slowInner(a, c);
11      a.a++;
12  }
13
14  function slowInner(a, b) {
15      return a.a + a.b;
16  }
```

Пример#1
Inline Cache

— очень дорого читать поля
— очень часто поля остаются на месте
— “форма объекта”
— все виртуальные машины

```
1   const object = {a: 1, b: 2, __proto__: { c: 3 }};
2
3   let sum = 0;
4   for (let i = 0; i < 10000; ++i) {
5       sum += object.a + object.b + object.c;
6   }
```

Пример#2 Inlining

— NP проблема - очень медленно
— все не выйдет - закончатся регистры
— стоимость компиляции не линейна

```
1   function foo() {
2       const a = { prop: 239 };
3       const b = { prop: 42 };
4       return boo(a, b);
5   }
6
7   function boo(a, b) {
8       return a.prop;
9   }
```

```
1   function foo() {
2       const a = { prop: 239 };
3       const b = { prop: 42 };
4       return a.prop;
5   }
```

Как экономить память?

— во время выносить мусор
— неявная работа с памятью
— постоянно новый мусор
— больше собираем -
меньше тратим на пике -
меньше времени на остальное
— количество собранных байт на затраченное время

Как собирать?

— пометить все живое
— вынести все остальное

— наивная реализация с полной остановкой
— помечать параллельно (memory barriers)
— помечать меньше (поколения)

JavaScriptCore (новый)

- Ferrari
- очень быстро на Apple
- не работает на arm32
- побольше памяти
- хорошие инструменты
- Bun.js, Safari, WebKit

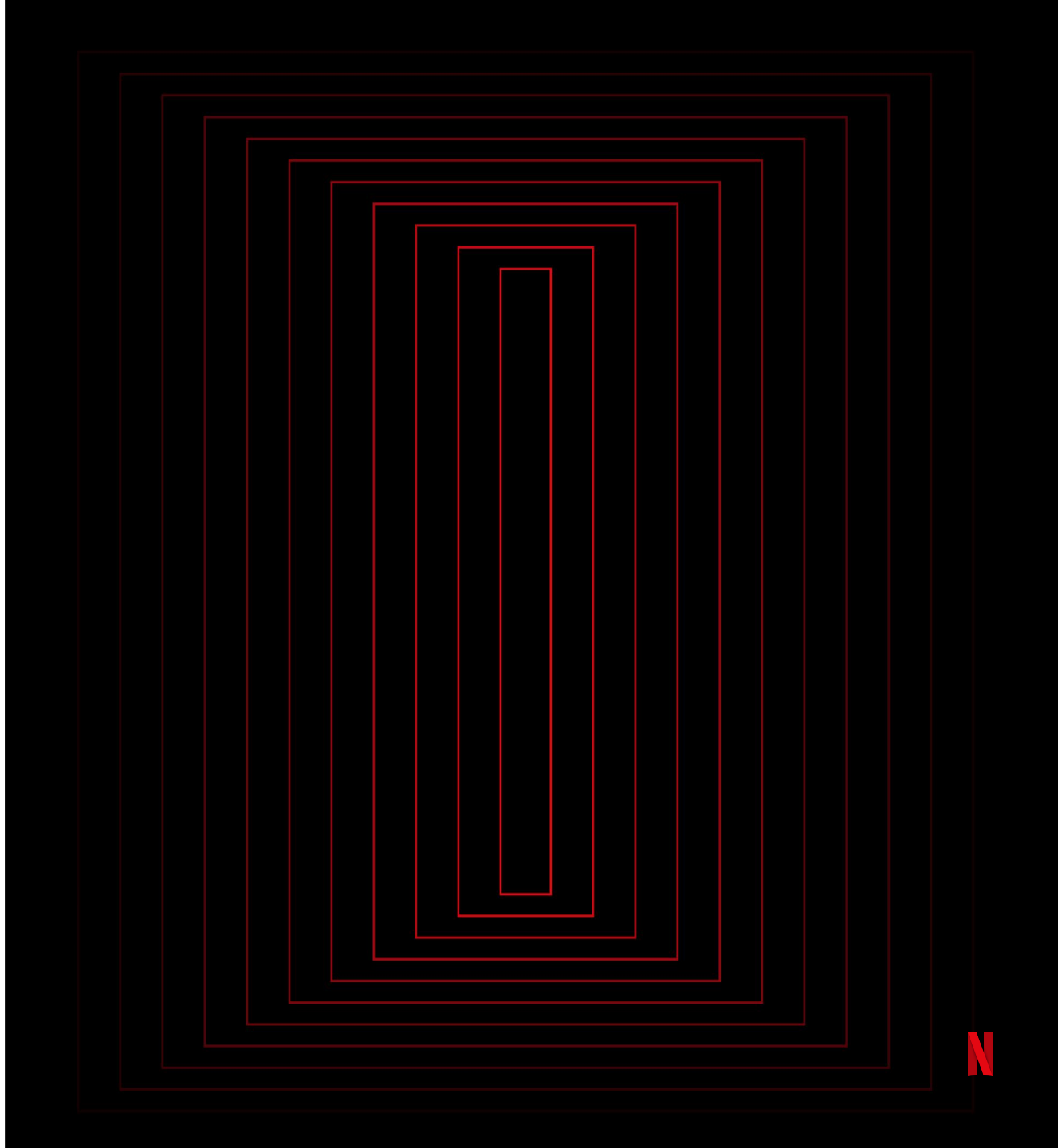
V8

- работает практически на всем
- может быть быстрым, либо экономным
- сравним по скорости с JavaScriptCore
- отличные инструменты
- Node.js, Chromium, Electron

Hermes

- JavaScript → байткод на сервере
- Был медленней и прожорливей
- Становится быстрей
- Минимум инструментов
- Идеальный если пять релизов в день
- React Native

Результаты



Результаты

- 40% (!!!) меньше памяти на пике
  - ленивый V8
  - все более компактное
  - эффективный вынос мусора
- 5-10% медленней
  - ???

Object.assign

— очень горяч в React приложении
— по стандарту - копируем всё!
— но ведь можно быстрей...

```
1   function simpleAssign(target, ...sources) {
2       sources.forEach(source => {
3           if (source) {
4               Object.keys(source).forEach(key => {
5                   target[key] = source[key];
6               });
7           }
8       });
9       return target;
10  }
```

Object.assign

```
1   function unsafeObjectMerge(object1, object2) {
2       /* eslint-disable tvui/no-direct-proto-assignment */
3
4       // @ts-expect-error - `__proto__` hack is not supported
5       object2.__proto__ = object1;
6
7       /* eslint-enable */
8       return object2 as
9           Omit<typeof object1, keyof typeof object2> & typeof object2;
10  };
```

Результаты

- 40% (!!!) меньше памяти на пике
  - ленивый V8
  - все более компактное
  - эффективный вынос мусора
- 10-20% быстрей
  - компиляторы

Каждый человек должен уверенно следовать по той стезе, которую выбрал. Должен быть предан своему делу. Инженер не должен быть лишен честолюбия в хорошем смысле этого слова, чтобы не было стыдно сказать: "Это моя разработка, моя конструкция"

Юрий Николаевич Козятинский

Спасибо!

https://c.cloudpayments.ru/payments/8e95d0d1c360432f9222d5e0d55073c5

Алексей Козятинский
ak239spb@gmail.com
https://x.com/the_kozy