Немного шалим со свежими WeakRef и FinalizationGroup API

Андрей Печкуров

О докладчике

- Пишу на Java (10+ лет), Node.js (5+ лет)
- Интересы: веб, архитектура, распределенные системы, производительность
- Можно найти тут:
  - https://twitter.com/AndreyPechkurov
  - https://github.com/puzpuzpuz
  - https://medium.com/@apechkurov

План на сегодня

- История вопроса
- Знакомство с WeakRef и FinalizationGroup API
- Простые примеры использования
- Шалости: Buffer pool для Node.js

История вопроса

Чисто там, где не мусорят

- Первый garbage collector - Lisp, 1959 (лень-матушка)
- Задачи любого GC:
  - Отследить объекты, более недоступные в программе
  - Освободить память в куче под новые объекты
  - Дефрагментировать память (опционально)

Особенности GC

- GC не подразумевает VM, но часто идет в связке
- GC отличаются стратегиями:
  - Tracing (самая популярная)
  - Reference counting
  - Escape analysis (compile-time, стоит особняком)
- Разновидностей конкретных алгоритмов GC - весьма много
- Кроме GC есть такие compile-time штуки, как automatic reference counter (ARC)

Что у нас в JS? (V8)

- Конечно, tracing стратегия
- Комбинирует различные подходы к сборке мусора, чтобы минимизировать stop-the-world паузы
- Кому интересны подробности: https://v8.dev/blog/trash-talk

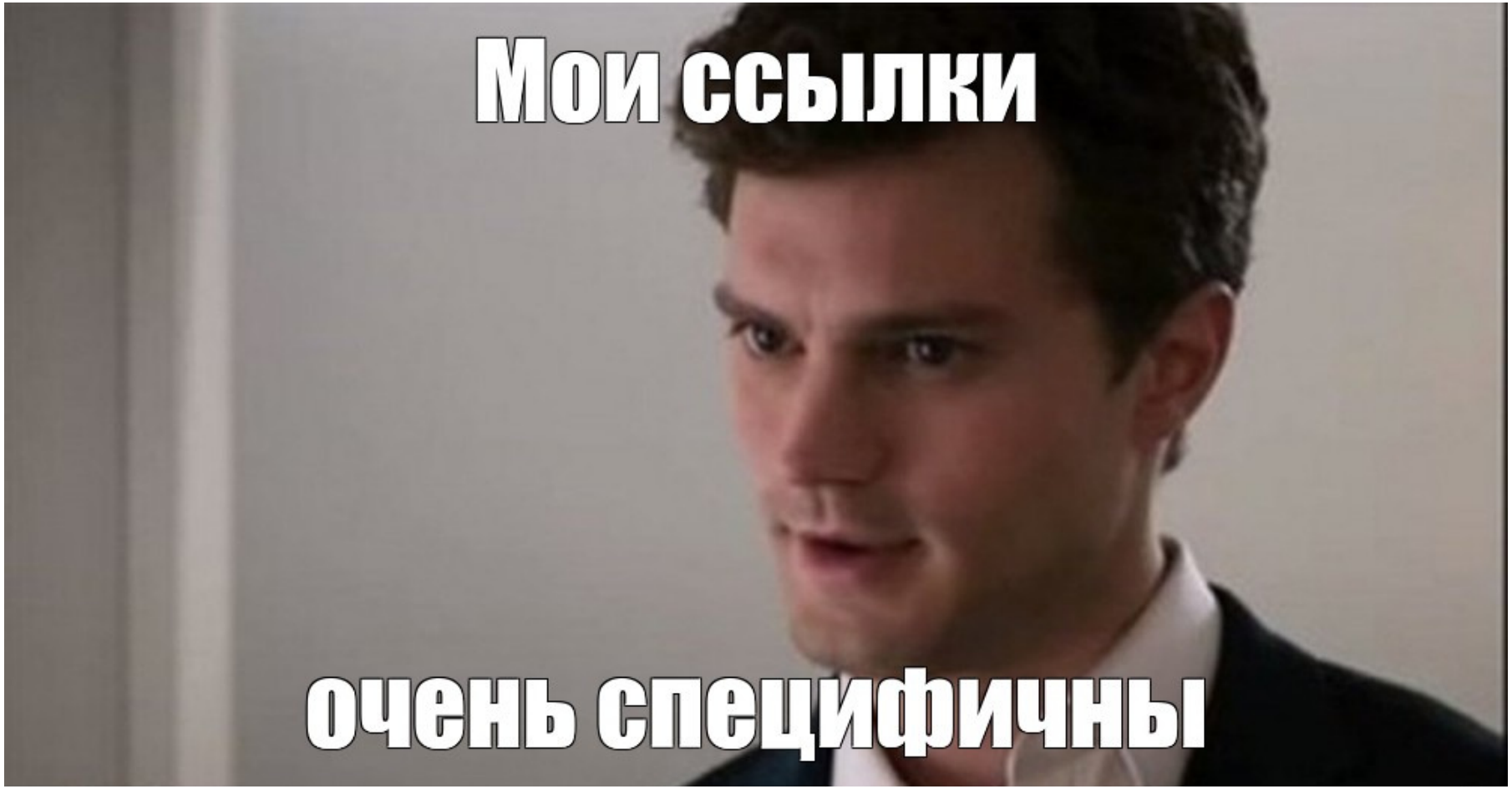
Ссылка есть? А если найду?

- GC оперирует обычными ссылками (strong reference)
- Конечно же, GC учитывает графы зависимостей (и не боится циклов в них)

```
let baz = { answer: 42 };
const foo = { bar: baz };

baz = null;
// a few moments later...
// baz жил, baz жив, baz будет жить
console.log('baz: ', foo.bar);
```

Но что если хочется "необычных" ссылок?



Ссылка есть? А куда дел?

- Во многих языках есть другие виды ссылок
- Например, слабые ссылки (weak reference)
- *Оффтопик*. В ARC слабые ссылки особенно важны

```
let baz = { answer: 42 };
const foo = { bar: new WeakRef(baz) };

baz = null;
// a few moments later...
// baz приказал долго жить
console.log('baz: ', foo.bar.deref());
```

История появления weak reference

- Диалекты Lisp
- Haskell
- Java 1.2, 1998
- Perl 5.xxx, 1999
- Python 2.1, 2001
- .NET Framework 1.1, 2002
- И много чего еще

- Знакомство с WeakRef и FinalizationGroup API

WeakRef

```
// поддерживает только объектные типы
const validRef = new WeakRef({foo: 'bar'});
//const invalidRef = new WeakRef(1); // TypeError

// имеет ровно один метод
const fooBar = validRef.deref();
// в fooBar будет или наш объект, или undefined
if (fooBar !== undefined) {
  console.log('жив курилка!');
}
```

WeakRef + WeakMap/WeakSet

- API WeakMap/WeakSet не связаны с WeakRef
- Конечно, WeakRef не препятствует очистке элементов в WeakMap/WeakSet
- Map + WeakRef !== WeakMap (проблема в ссылках из значений на ключи)
- WeakMap основан на ephemeron, а не на "классических" слабых ссылках

FinalizationGroup

TODO

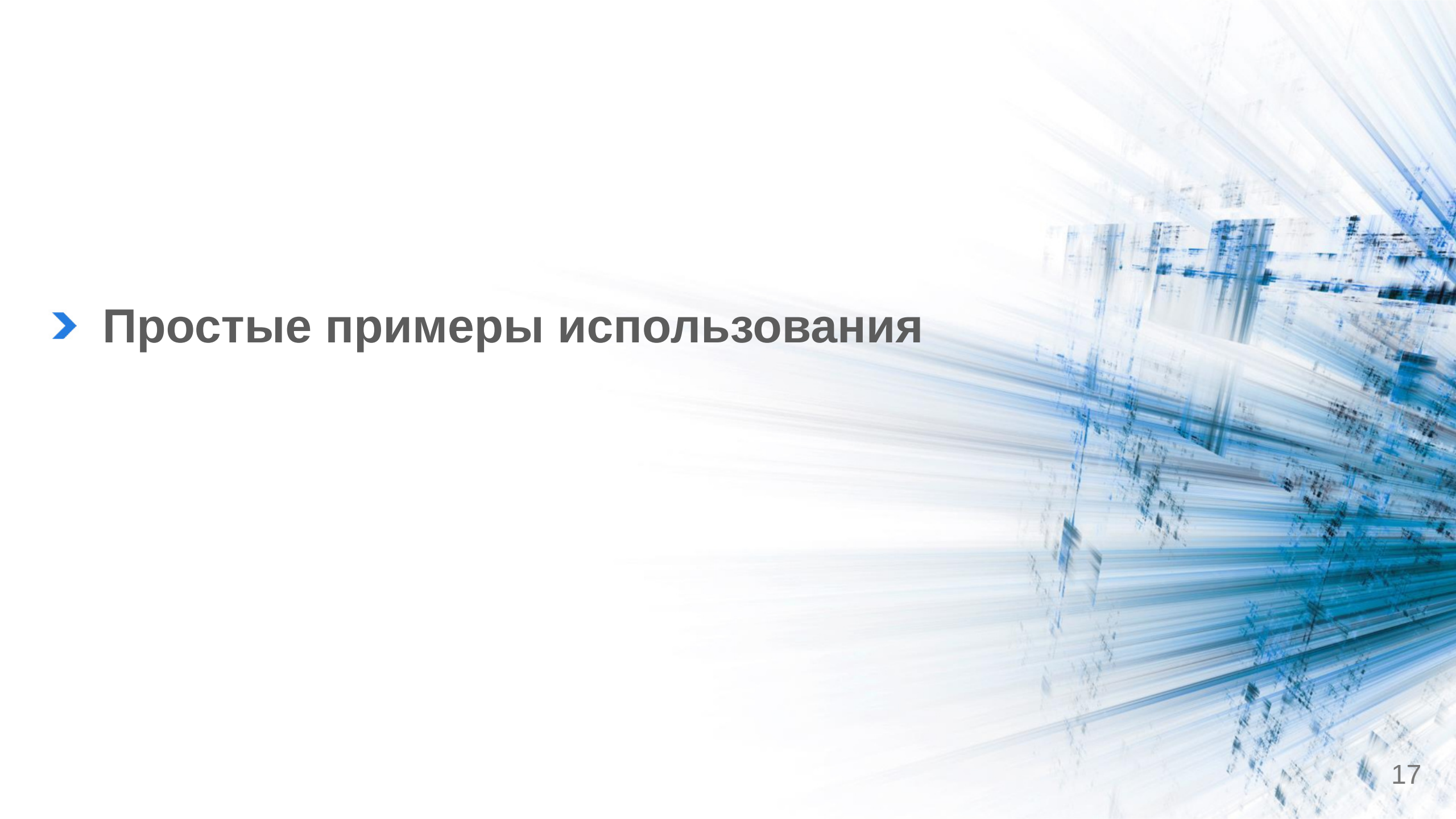
https://github.com/tc39/proposal-weakrefs#another-note-of-caution
https://github.com/tc39/proposal-weakrefs#scheduling-of-finalizers-and-consistency-of-multiple-deref-calls

КО подсказывает: время сборки мусора непредсказуемо

Когда ждать?

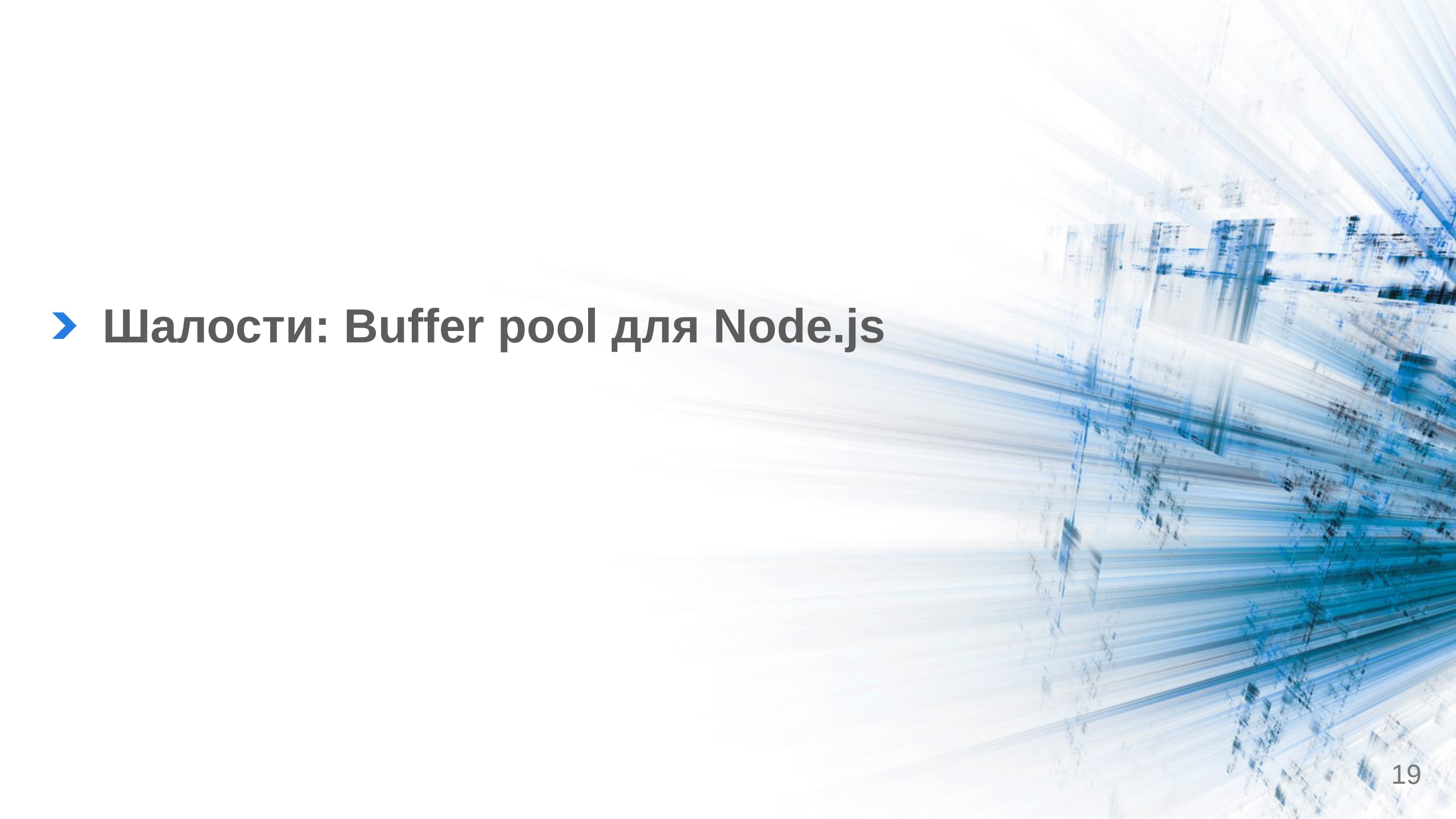
- Сейчас спецификация на stage 3 в TC39 (предпоследний шаг)
- https://github.com/tc39/proposal-weakrefs
- Можно щупать в Node.js v12+ (и V8) с флагом `--harmony-weak-refs`

Простые примеры использования



TODO

Шалости: Buffer pool для Node.js



TODO

- Hazelcast In-Memory Data Grid (IMDG)
- Большой набор распределенных структур данных
- Показательный пример - `Map`, который часто используют как кэш
- Написана на Java, умеет embedded и standalone режимы
- Хорошо масштабируется вертикально и горизонтально
- Часто используется в high-load и low-latency приложениях
- Области применения: IoT, in-memory stream processing, payment processing, fraud detection и т.д.

Hazelcast IMDG Node.js client

- https://github.com/hazelcast/hazelcast-nodejs-client
- Доклад про историю оптимизаций
  - Видео: https://youtu.be/CSnmpbZsVD4
  - Слайды: https://github.com/puzpuzpuz/talks/tree/master/2019-ru-nodejs-library-optimization

Спасибо за внимание!

Полезные ссылки

- https://github.com/tc39/proposal-weakrefs
- https://github.com/tc39/proposal-weakrefs/blob/master/history/weakrefs.md
- http://www.cs.bu.edu/techreports/pdf/2005-031-weak-refs.pdf
- http://www.jucs.org/jucs_14_21/eliminating_cycles_in_weak/jucs_14_21_3481_3497_barros.pdf