Санкт-Петербургский Академический университет - научно-образовательный центр нанотехнологий РАН

Оптимизация JavaScript-кода, генерируемого компилятором Kotlin

Руководитель: Бреслав А.А.
Выполнил: Башоров З.А.

Санкт-Петербург
2013

Цель

Улучшить компилятор Kotlin в JavaScript так, чтобы он генерировал более эффективный, в плане производительности, код.

Задачи

- Изучить существующие аналоги.
- Изучить существующие виртуальные. машины JavaScript.
- Выбрать и реализовать тесты производительности для сравнения (бенчмарки).
- Проанализировать результаты тестов и выделить "узкие места".
- Предложить и реализовать способы их оптимизации.

Тесты производительности

- **Richards**
  Симулирует работу ядра ОС.
  Основной фокус: чтение и запись данных, вызов функций.
- **DeltaBlue**
  Constraint solver.
  Основной фокус: полиморфизм, нагрузка на GC.
- **Havlak**
  Используется алгоритм поиска и преобразования циклов в исходном коде.
  Основной фокус: контейнеры (List, Map), нагрузка на GC

С кем сравниваемся?

- **JavaScript**

  Целевая "платформа" для многих современных языков[1].

- **Dart**

  Новый язык для разработки веб-приложений.

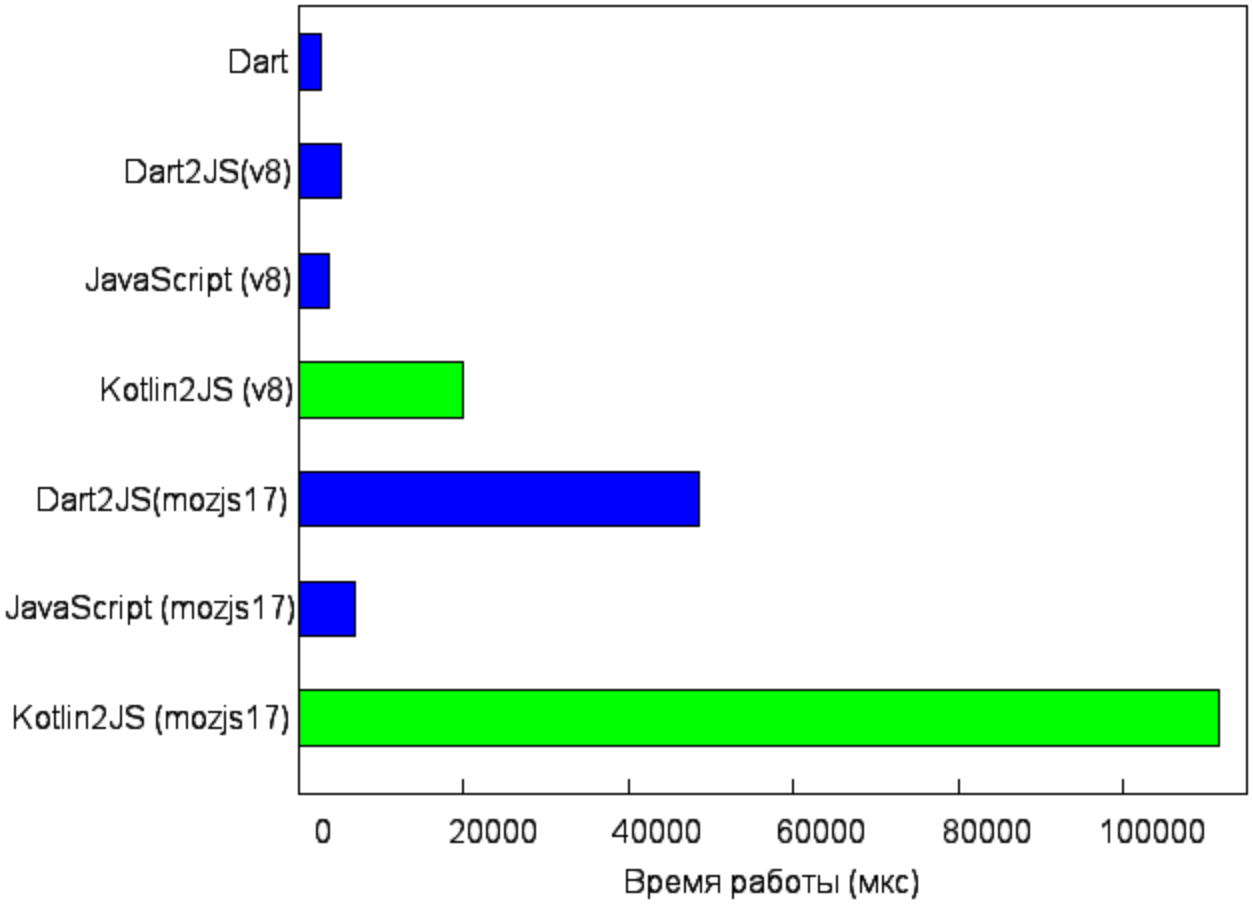
- **Dart2js**

  Компилятор из языка Dart в JavaScript.

[1] altjs.org

Richards

Результаты до оптимизации

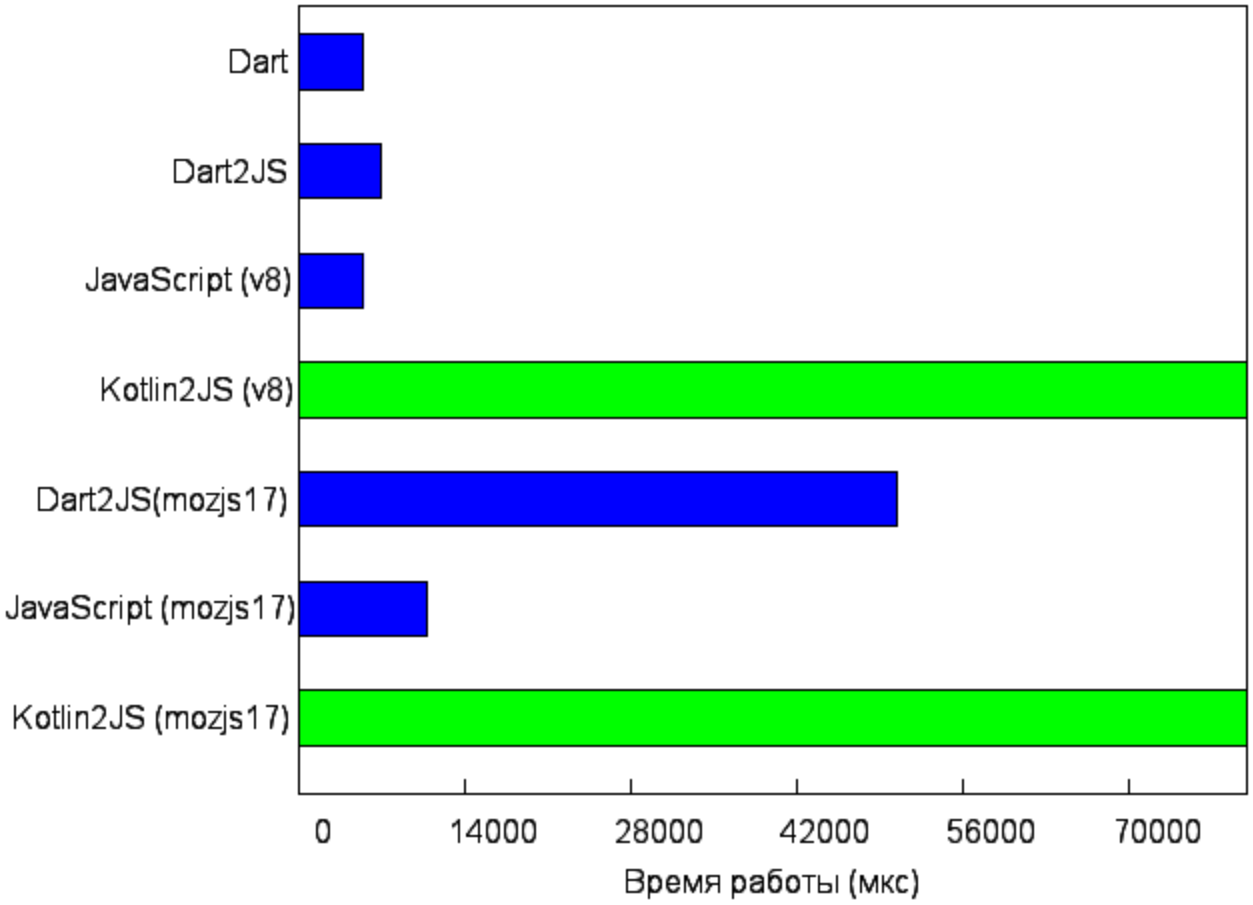


Richards: Выводы

- Оптимизировать доступ к полям/свойствам, по возможности, сохранив "бинарную" совместимость.
- Сделать аналог импорта, предоставив тем самым возможность кратко обращаться к объектам и их полям.
- Необходимо упростить структуру генерируемого кода.

DeltaBlue

Результаты до оптимизации

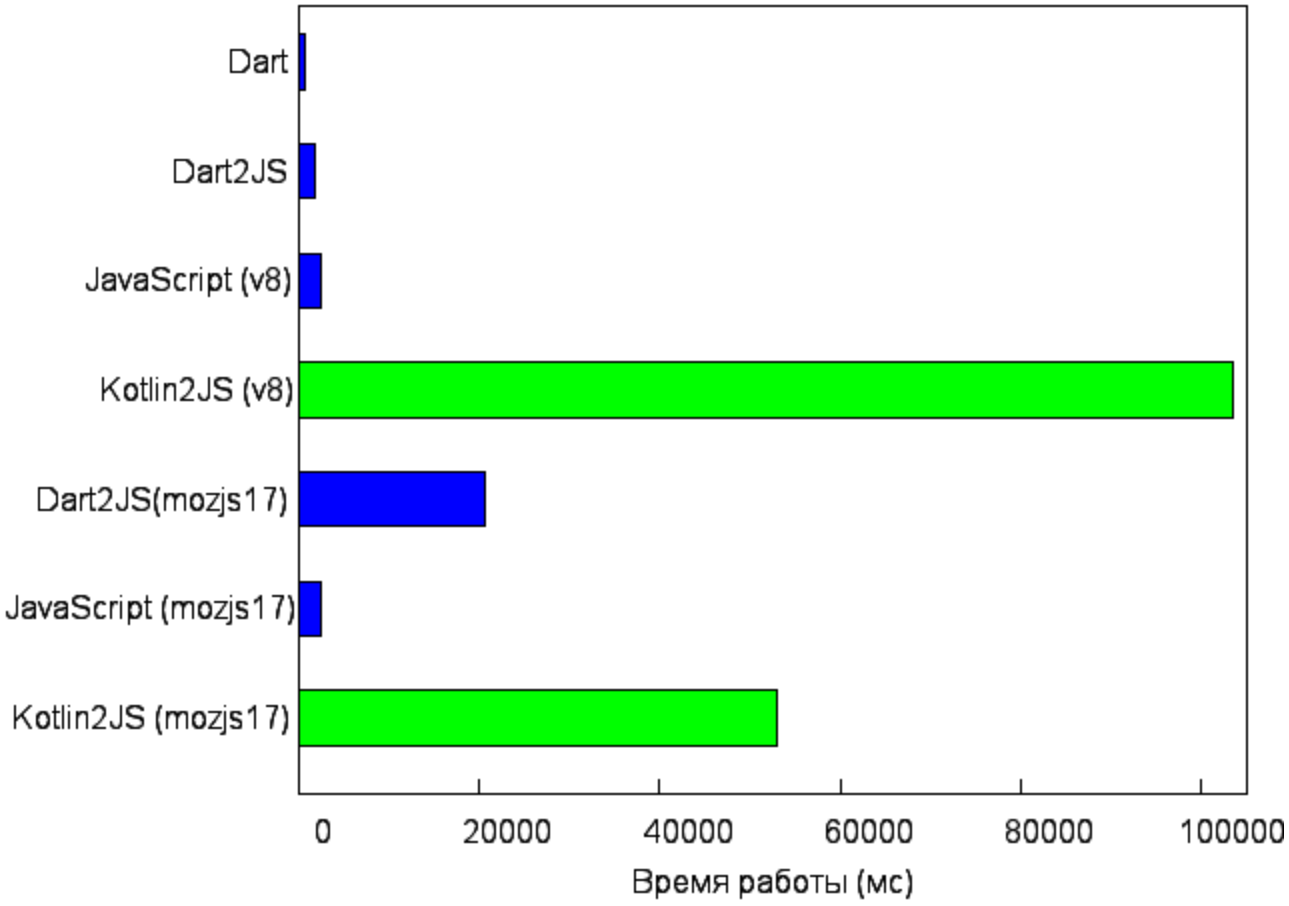


DeltaBlue: Выводы

- Упростить реализацию наследование.
- Заменить все контейнеры на родные для JavaScript аналоги.
- Заменить или оптимизировать Kotlin. equals.

Havlak

Результаты до оптимизации



Havlak: Выводы

- Оптимизировать реализацию получения хэш-кода у объектов.
- Использовать PrimitiveHashMap когда это возможно.
- Упростить реализацию наследование.

Пример оптимизации

```
1  open class A(val name: String)
2
3  class B(name: String) : A(name)
4
```

Пример оптимизации

```kotlin
open class A(val name: String)

class B(name: String) : A(name)

```

```javascript
(function () {
  'use strict';
  var classes = function () {
    var c0 = Kotlin.createClass({
      initialize: function (name) {
        this.$name = name;
      },
      get_name: function () {
        return this.$name;
      }
    });
    return {c0: c0};
  }()
  , _ = {
    A: classes.c0,
    B: Kotlin.createClass(classes.c0, {
      initialize: function (name) {
        this.super_init(name);
      }
    }),
  };
  Kotlin.defineModule('JS_TESTS', _);
}());
```

Пример оптимизации

```
16      B: Kotlin.createClass(classes.c0, {
17        initialize: function (name) {
18          this.super_init(name);
19        }
20      }),
```

Пример оптимизации

```
16      B: Kotlin.createClass(classes.c0, {
17        initialize: function (name) {
18          this.super_init(name);
19        }
20      }),
```

```
1  function klass() {
2      this.initializing = klass;
3      if (this.initialize) {
4          this.initialize.apply(this, arguments);
5      }
6  }
7
8  function super_init() {
9      this.initializing = this.initializing.superclass;
10     this.initializing.prototype.initialize.apply(this, arguments)
11 };
```

Пример оптимизации

```
16      B: Kotlin.createClass(classes.c0, {
17        initialize: function (name) {
18          this.super_init(name);
19        }
20      }),
```

```
B = B.prototype.initialize
```

Пример оптимизации

```
16    B: Kotlin.createClass(classes.c0, {
17      initialize: function (name) {
18        this.super_init(name);
19      }
20    }),
```

```
B = B.prototype.initialize
```

```
16    B: Kotlin.createClass(classes.c0, {
17      initialize: function (name) {
18        A.call(this, name);
19      }
20    }),
```

Результаты (1/2)

- Изучены существующие аналоги.
- Изучены существующие JS виртуальные машины.
- Написаны тесты производительности.
- Выделены "узкие места" и предложены. способы их оптимизации.

Результаты (2/2)

Реализованные оптимизации:

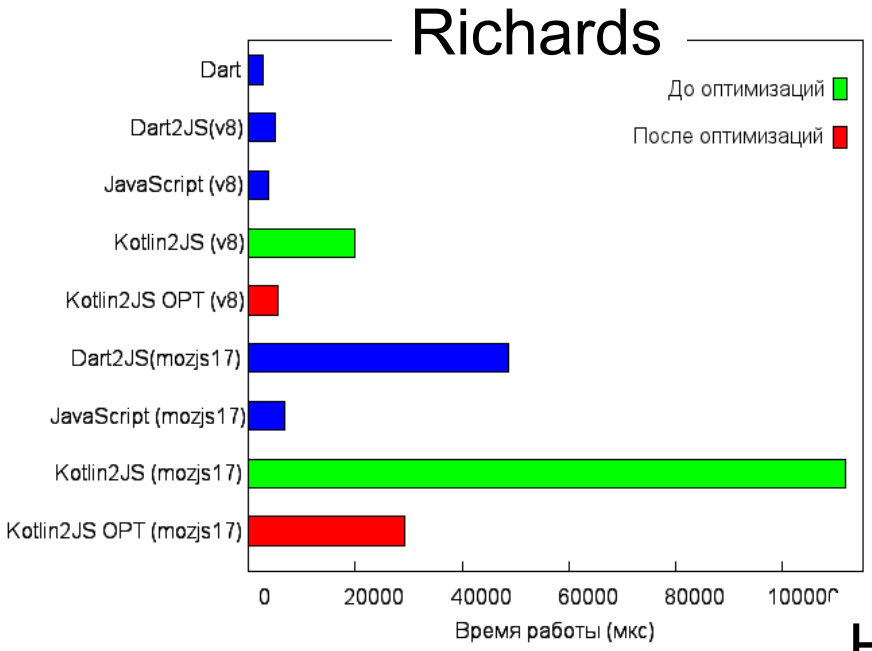
- Оптимизирован доступ к свойствам.
- Улучшена реализация наследования.
- Улучшено сравнивание объектов.
- Оптимизировано вычисление хеш-кода
- Для примитивных типов используется PrimitiveHashMap.
- Оптимизированы реализации HashMap и PrimitiveHashMap.

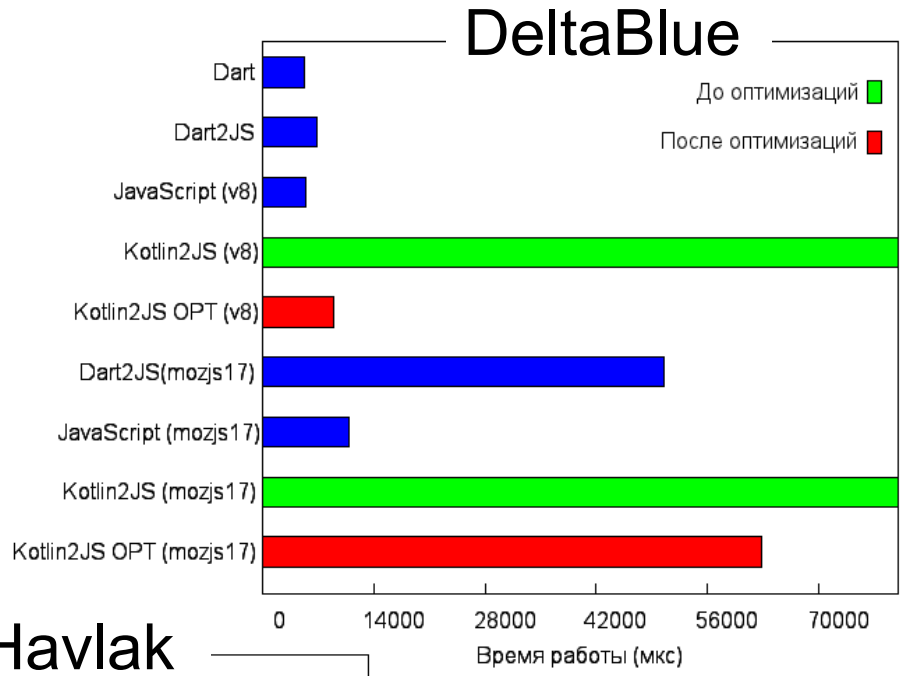
Результаты тестов (1/2)

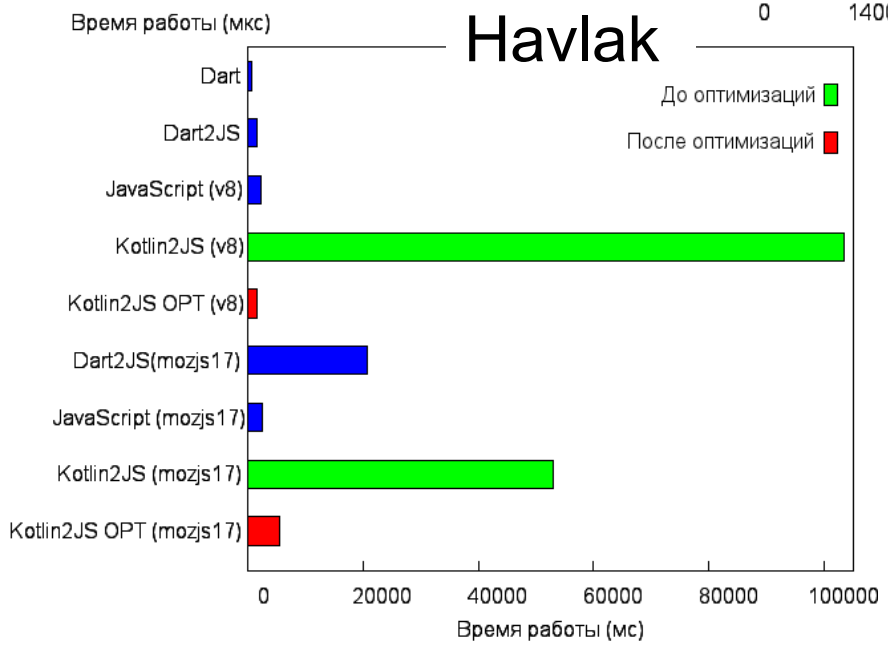
Сводная таблица

Тест	До опт. (мкс)	После опт. (мкс)	Ускорение (раз)
Richards (v8)	19 900.99	5 647.88	3.52
Richards (mozjs17)	111 722.22	29 188.4	3.83
DeltaBlue (v8)	1 347 000	8 950.89	150.49
DeltaBlue (mozjs17)	563 000.65	62 750	8.97
Havlak (v8)	103 371 000	1 641 000	62.99
Havlak (mozjs17)	52 893 000	5 615 000	9.42

Результаты тестов (1/2)







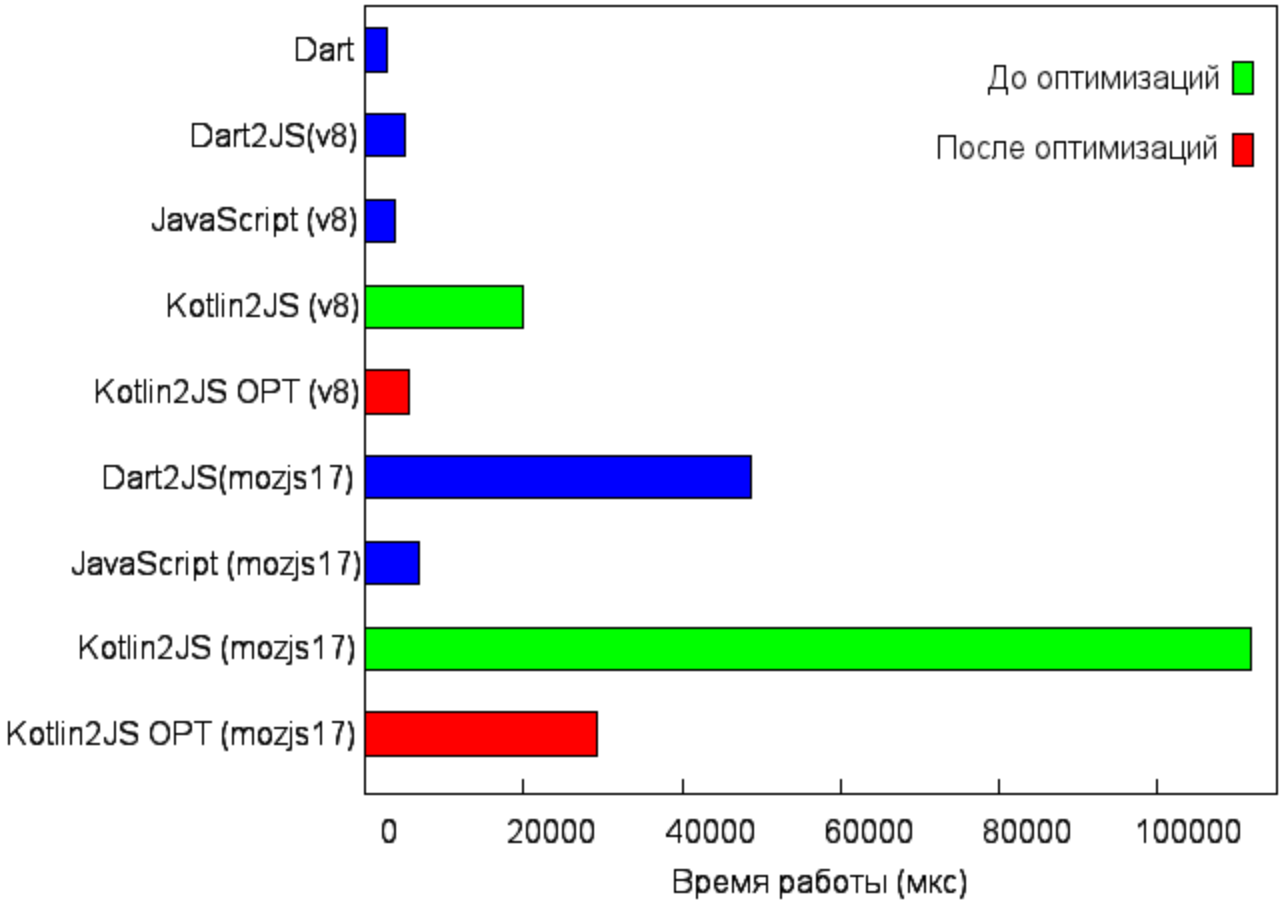
Спасибо за внимание!

kotlin.jetbrains.org

Залим Башоров (bashorov@gmail.com)

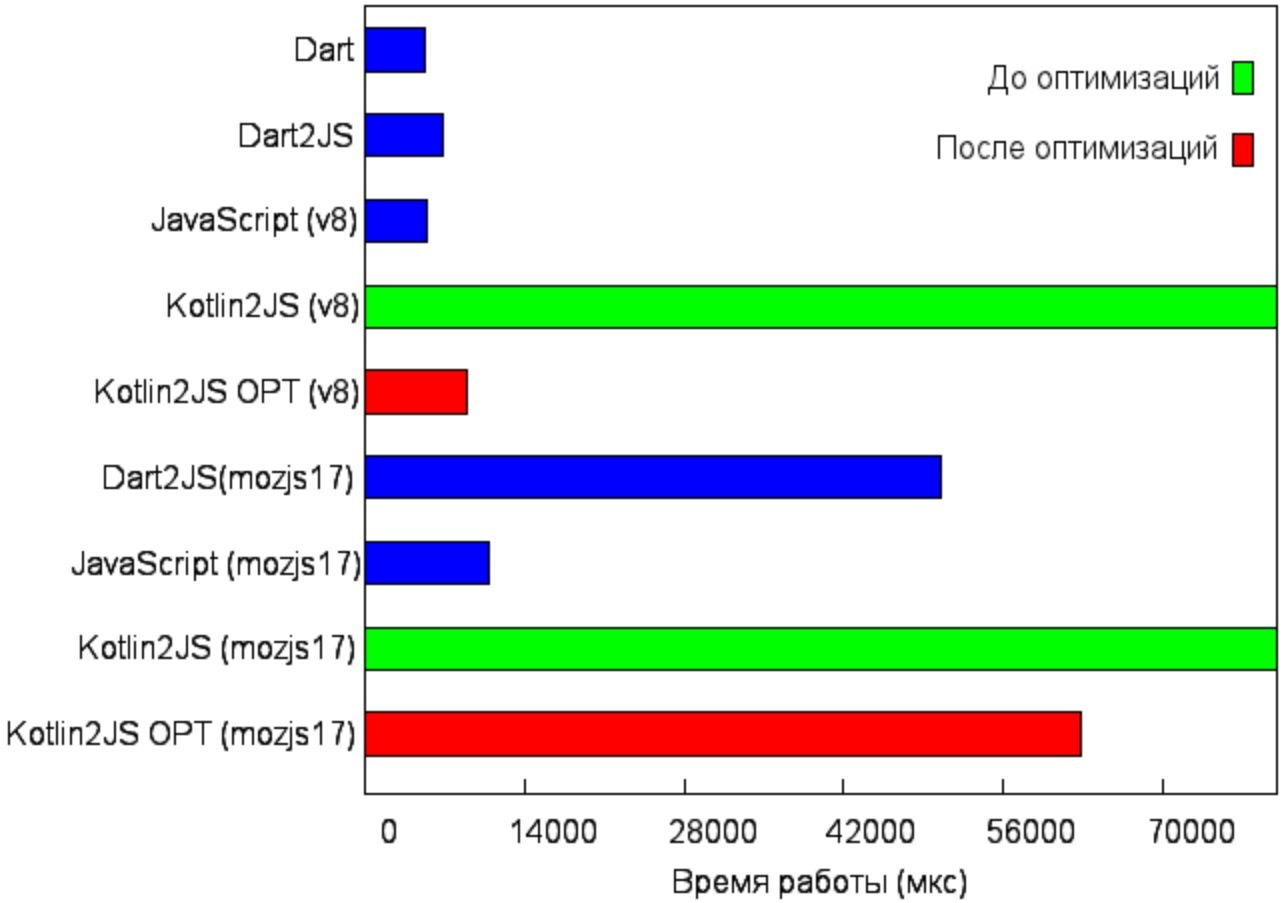
Результаты тестов (2/4)

Richards



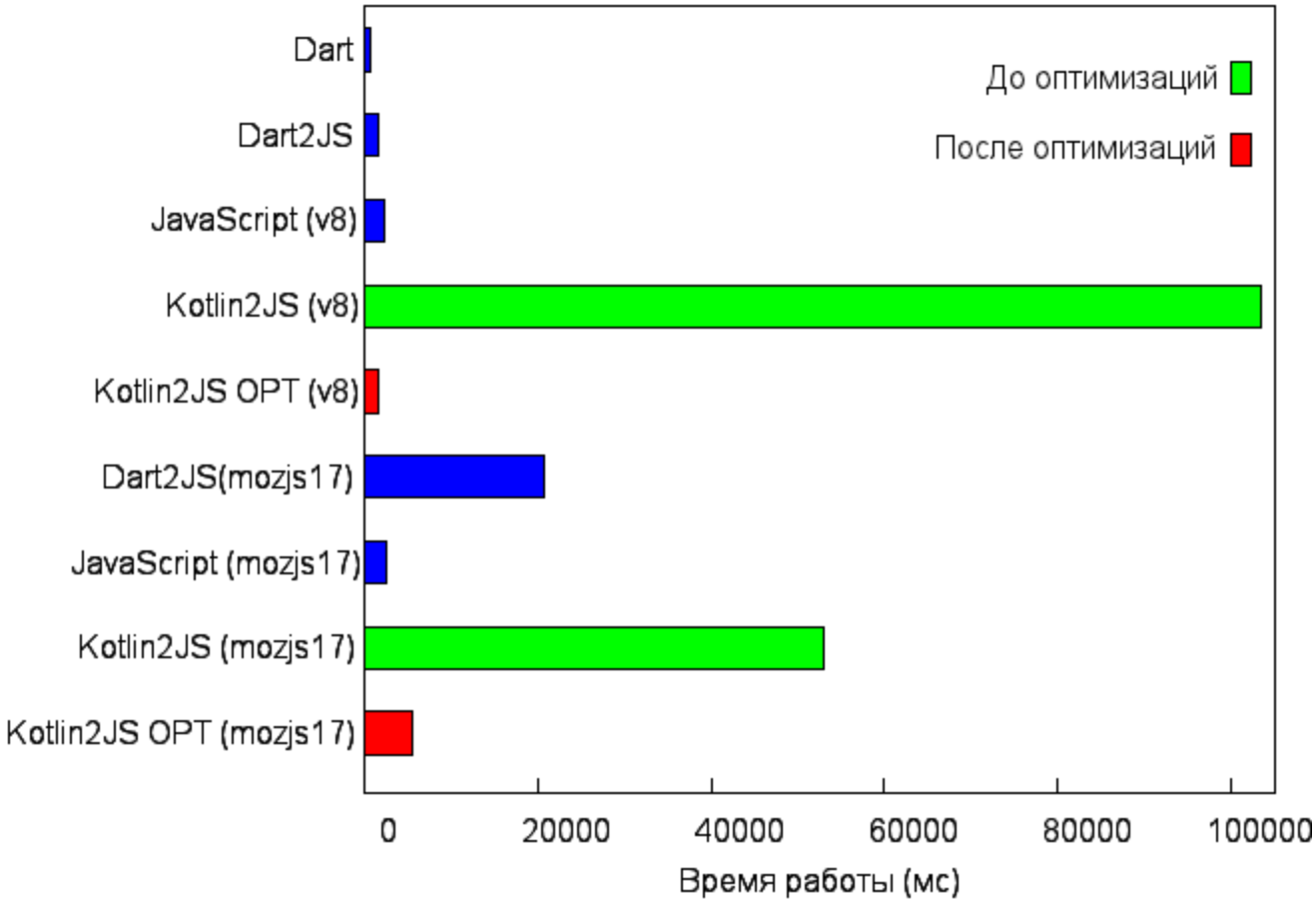
Результаты тестов (3/4)

DeltaBlue



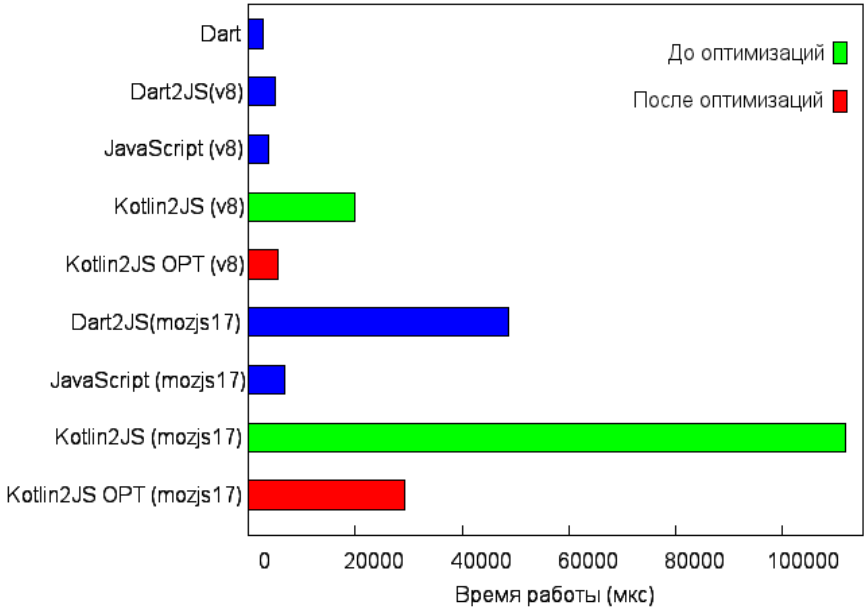
Результаты тестов (4/4)

Havlak

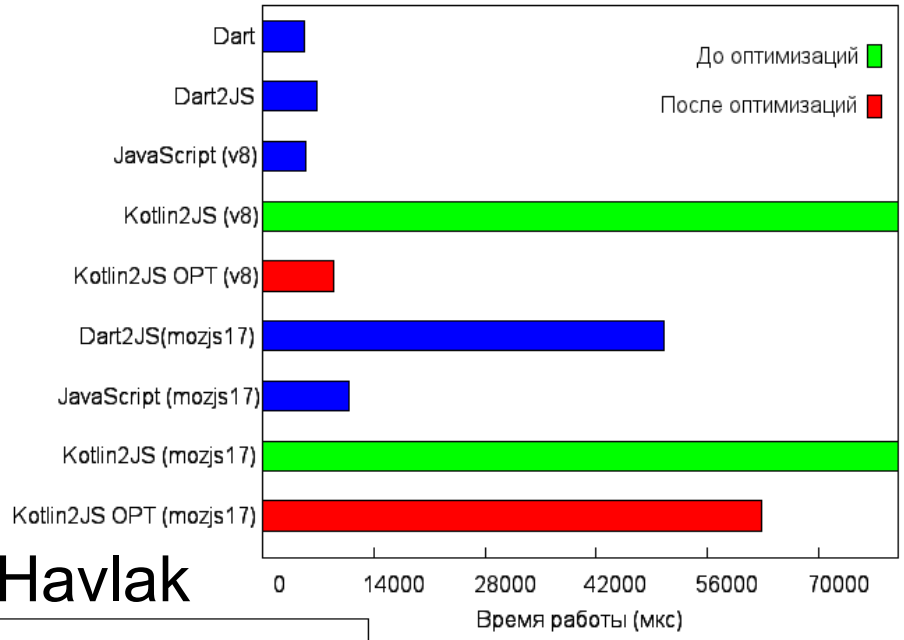


Результаты тестов (1/2)

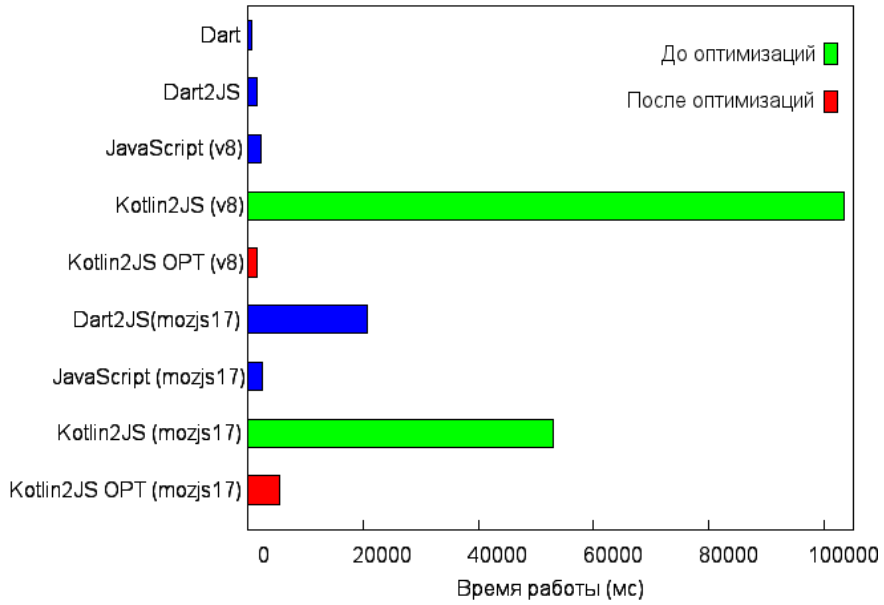
Richards



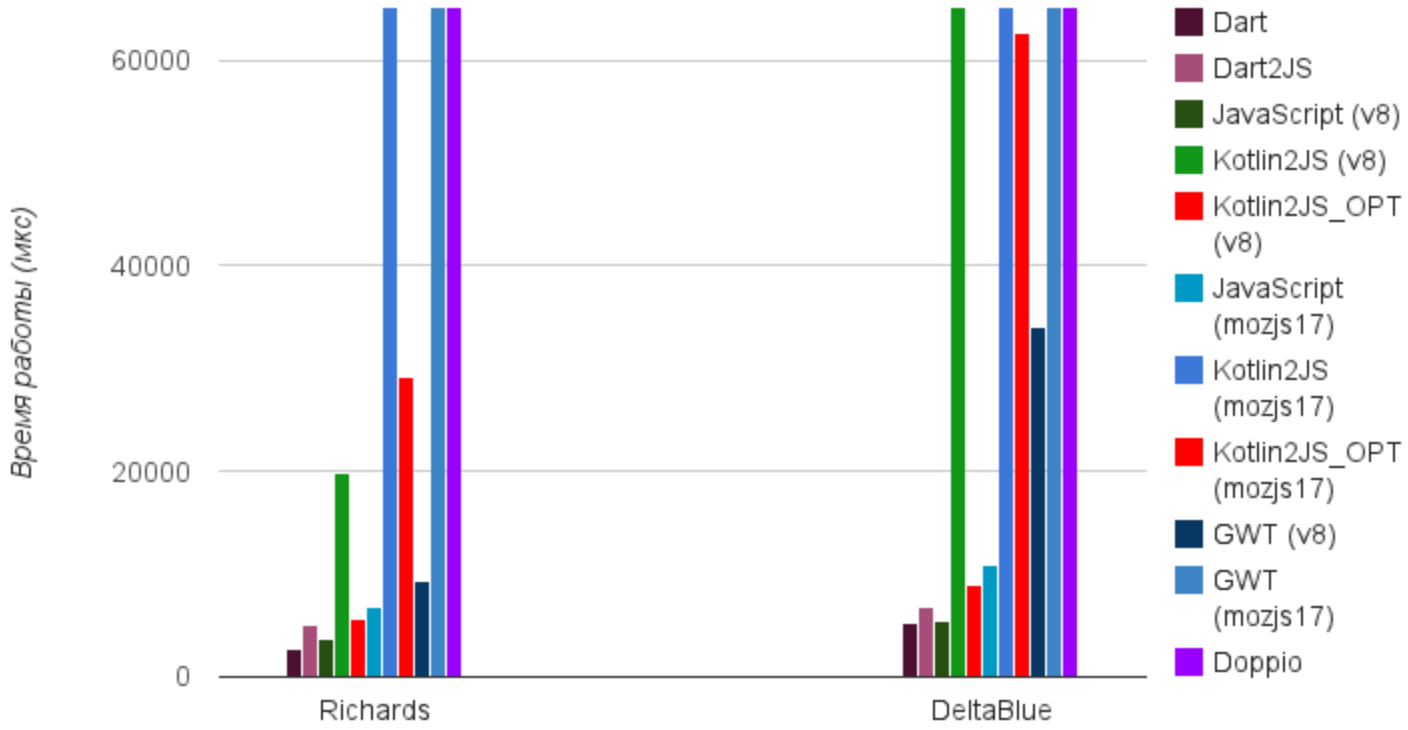
DeltaBlue



Havlak



Результаты тестов 2:



Введение (1/2)

JavaScript — прототипно-ориентированный скриптовый язык программирования с динамической типизацией. Является диалектом языка ECMAScript.

Kotlin — статически типизированный объектно-ориентированный язык, разрабатываемый в компании JetBrains. Компилируется в Java байт-код и в JavaScript.

Введение (2/2)

JavaScript is the x86 of the web

Brendan Eich (создатель JavaScript)

...JavaScript is the VM of the web. ....

Douglas Crockford (известный JS-разработчик, создатель JSON)

JavaScript is an assembly language for the Web. ...

Erik Meijer (language designer из Microsoft)

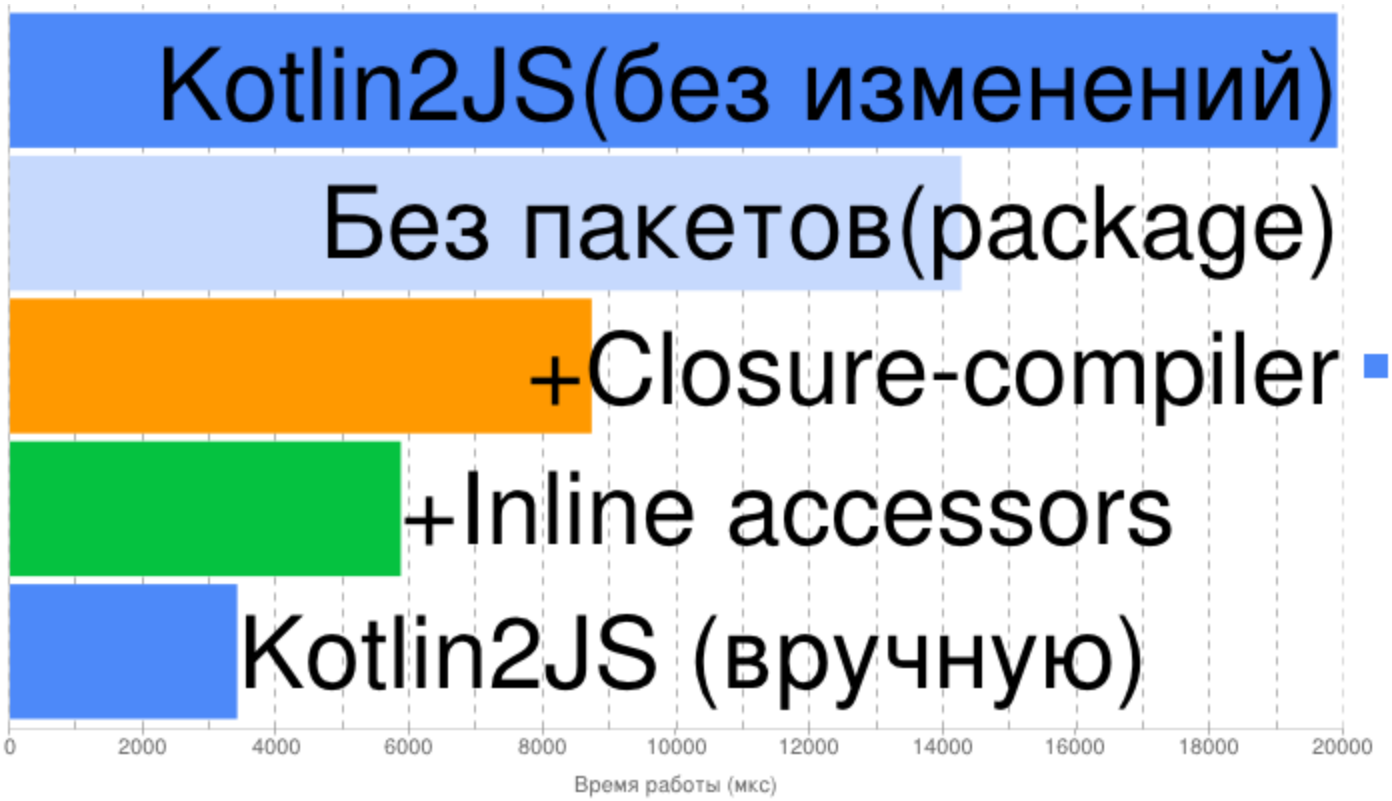
Почему JavaScript?

- Интерпретатор JS есть почти везде
- Удобно писать все части системы на одном языке
- JS не очень удобен для разработки больших проектов
- ...

Зачем изобретать велосипед?

- Компилятор должен быть быстрым
- В результате компиляции должен получаться читаемый код
- Полученный код должен быть быстрым

Richards: Оптимизация



"+" означает что изменения накладываются на предыдущее

DeltaBlue: Оптимизация

"+" означает что изменения накладываются на предыдущее

JS виртуальные машины

	Разработчик	Где используется?
v8	Google	Chrome, Node.js, Android, ...
SpiderMonkey	Mozilla	Firefox, Thunderbird, ...
Chakra	Microsoft	IE ver. ≥ 9
JavaScriptCore	WebKit	Safari, iOS ...
Rhino	Mozilla	

Аналоги (1/2)

- **Dart** — язык программирования, созданный Google, позиционируется в качестве замены JavaScript
- **Dart2js** — компилятор Dart в JavaScript. Разрабатывается той же командой что и Dart
- **CoffeeScript** — иной взгляд на JavaScript. Многое взято из Ruby, Python, Haskell и Erlang
- **TypeScript** — язык от Microsoft, позиционируемый как средство разработки веб-приложений

Аналоги (2/2)

- **Ceylon** — язык, разрабатываемый в RedHat. Компилируется в Java байт-код и JavaScript
- **Google Web Toolkit** — Java-фреймворк для создания веб-приложений. Включает в себя компилятор из Java в JavaScript
- **Closure-Compiler** — оптимизатор JavaScript - кода
- **Pyjs, qb.js, ghcjs, ClojureScript, ...**

Большой обзор на **altJs.org**