JavaScript завтра

Рубанов Сергей
Exante Limited

FrontendConf 2015

План доклада

- JavaScript вчера
- JavaScript сегодня
- JavaScript завтра

JavaScript вчера

Интерпретируемый язык программирования для браузера Netscape Navigator

- валидация форм
- динамическая замена изображений
- управление Java-аплетами извне

Брендан Айк, создатель JS

JavaScript

- интерпретируемый основанный на прототипном программировании
- с динамической типизацией

Экскурс в историю

- разработан в мае 1995 Бренданом Айком и получил название Mocha

- в сентябре 1995 был интегрирован в бета-версию браузера Netscape Navigator 2.0 уже под именем LiveScript

- в декабре 1995 года была выпущена третья бета-версия Netscape Navigator 2.0, в которой язык был переименован в JavaScript (с разрешения компании Sun)

- одновременно Netscape выпустила серверную реализацию JavaScript для Netscape Enterprise Server

Экскурс в историю

- в 1996 Microsoft выпускает IE 3 с JScript

- в 1996 Netscape, не добившись поддержки консорциума W3C, обращается к компании ECMA, при которой создается TC39 и выпускаются спецификации ES1 (июнь 1996) и ES2 (июнь 1998)

- в декабре 1997 выходит редакция ES3 (обработка исключений, регулярные выражения, switch, do-while и другие улучшения)

- 18 февраля 2005 Джесси Джеймс Гаррет выпускает статью "Ajax: A New Approach to Web Applications"

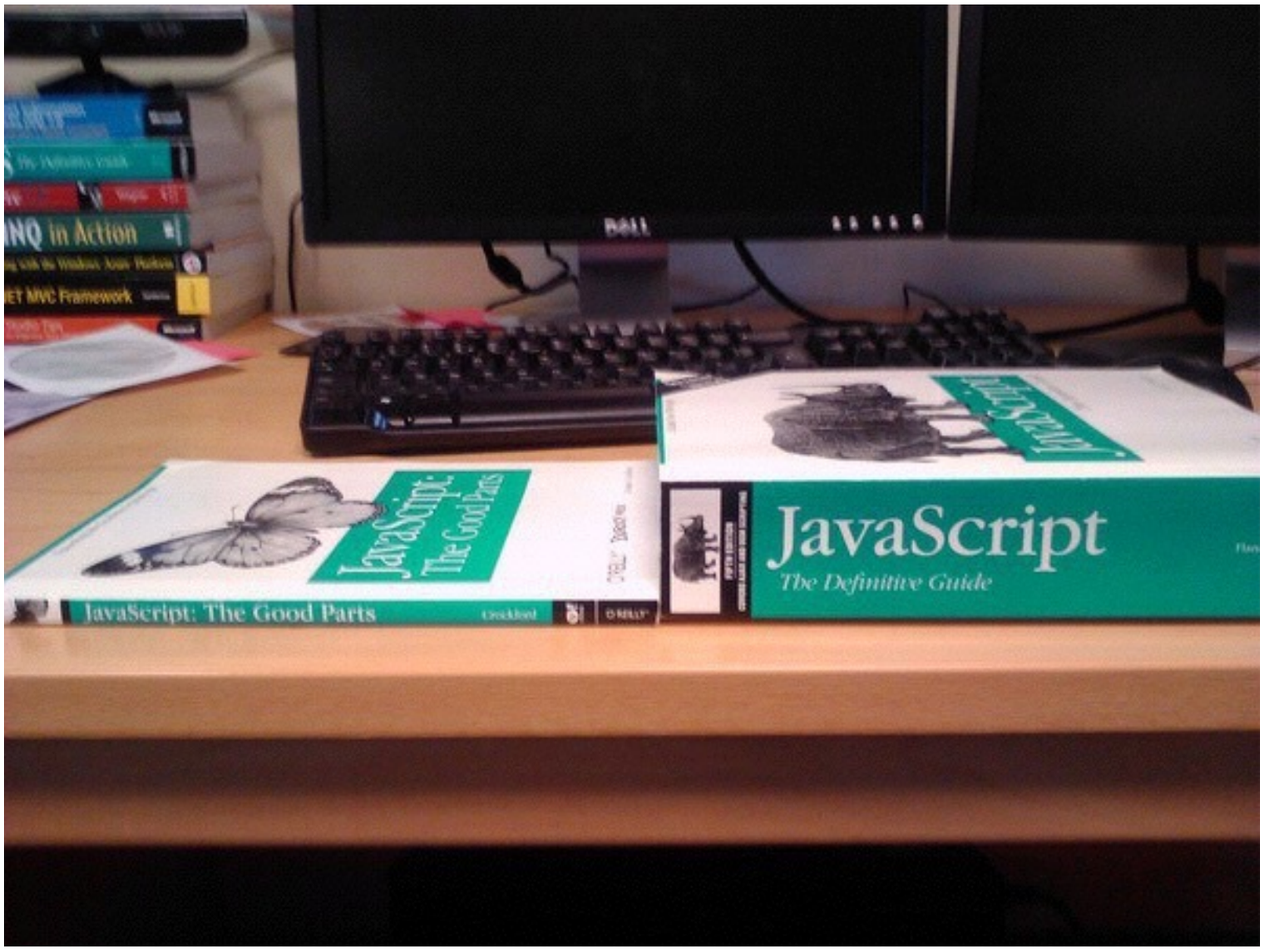
Экскурс в историю

- в 2008 прекращается работа над ES4 и начинается разработка Harmony

- декабрь 2009 — выход ES5 (strict mode, getters, setters, JSON, Object.create и т.д.)

- июнь 2011 — выход ES5.1 (приведение спецификации в соответствие стандартам ISO/IEC)

JavaScript сегодня



JavaScript: The Good Parts vs
JavaScript: The Definitive Guide

- стабильная версия 1.0.0 вышла 24 декабря 2010
- Золотое правило CoffeeScript: "It's just JavaScript"
- добавляет синтаксический сахар вдохновленный Python, Ruby и Haskell
- конструкции if, switch, for являются выражениями
- используется компаниями Dropbox, GitHub и многими другими, добавлен в Ruby on Rails 3.1+

Плюсы:

- позволяет не писать точки с запятой

Минусы:

- не заботится о скорости исполнения
- не совместим с ES6
- вызывает зависимость
- никогда не был "just JavaScript"

Популяризовал:

- rest parameters / spread operators (splats)
- destructuring assignment
- интерполяция строк
- классы
- =>
- аргументы по умолчанию
- for-of

Dart

- Создан 10 октября 2011.
- Позиционировался как замена JS, страдающего от фундаментальных изъянов, которые невозможно исправить
- 4 июля 2014 стал стандартом ECMA-408
- 25 марта 2015 создатели языка [объявили об остановке разработки VM]

Брендан Айк vs Dart team

news.ycombinator.com/item?id=9264531

Dart

Плюсы:

- добавляет опциональную типизацию
- аннотации
- более продвинутая система типов
- если верить разработчикам, то Dart VM незначительно обходит V8 по некоторым бенчмаркам

Dart

Минусы:

- маленькое коммьюнити
- добавляет относительно немного по сравнению с ES6
- заставляет грустить Брендана Айка

PNaCl

- представлен 12 ноября 2013
- основан на NaCl
  - компилирует C и C++ код в промежуточное представление подмножества LLVM с помощью AOT-компилятора
  - исполняется в песочнице браузера
- позволяет исполнять нативный код для процессоров архитектур Intel x86, ARM и MIPS

asm.js

- представлен 21 марта 2013
- представляет собой промежуточный язык программирования из кода на языках со статической типизацией и ручным управлением памятью
- является строгим подмножеством JavaScript

asm.js

- не предназначен для написания программ человеком, получается с помощью компиляторов исходного кода в исходный код, таких как Emscripten

- околонативная скорость исполнения достигается за счет AOT-оптимизации (в V8 применяется JIT)

- с 7 мая поддерживается браузером Microsoft Edge

Некоторые другие попытки улучшить JS

- ClojureScript
- scala.js
- LiveScript
- PureScript
- Elm
- TypeScript
- и т.д.

JavaScript

- интерпретируемый основанный на прототипном программировании
- с динамической типизацией

JavaScript

- динамически компилируемый
- основанный на прототипном программировании
- с динамической типизацией

JavaScript завтра

- релиз ES6 в июне 2015
  - то же самое касается ECMA 402 (Intl)
- уже частично поддерживается браузерами и серверными реализациями
- ECMAScript переходит на новую нумерацию версий

JavaScript завтра

- ECMAScript 2015
  - github.com/lukehoban/es6features
    - за исключением Module Loader API
    - черновик (Release Candidate #4, 3.04.2015)
- ECMAScript 2016+
  - github.com/tc39/ecma262
  - github.com/tc39/ecma262/blob/master/stage0.md

Транскомпиляторы

- Traceur
- Babel
- TypeScript 1.5+
- JSTransform
- и другие

Опциональная статическая типизация

- Closure Compiler (JSDoc)
- Flow
- TypeScript
- AtScript (был поглощен TypeScript)

TypeScript

- playground
- спецификация

Safe TypeScript

- обзор
- исследовательская работа
- playground

Типизация в ECMAScript

- 28 января 2015 TC-39 обсудил некоторые вопросы типизации на уровне JavaScript VM (Sane Mode и SoundScript)

- 1 февраля 2015 Дмитрий Ломов представил Stricter Mode и SoundScript на конференции The Rolling Scopes в Минске

    - SoundScript как альтернатива аннотаций asm.js

- реализация в Traceur и Chrome Canary

- страница V8 experiments (Strong Mode и SoundScript)

- strawman proposal

Sane/Stricter/Strong mode

- запрещено расширение объектов и классов
- var и необъявленные переменные запрещены
- запрещен доступ к несуществующим свойствам
- запрещены дыры в массивах
- количество параметров функций строго определено, доступ к объекту arguments запрещен

Sane/Stricter/Strong mode

- undefined не может быть переопределен
- ==, !=, for-in, + для случаев кроме пар строк и пар числе запрещен, switch ограничен
- eval запрещен
- интероперабелен с "weak mode" (и наоборот)

SoundScript

- типы используются для ранней и агрессивной оптимизации кода
- IDE могут использовать аннотации типов для раннего обнаружения ошибок
- использование strong mode и gradual (sound) typing позволяет сократить количество проверок во время исполнения, а также оптимизировать их
- предлагает резервацию синтаксиса TypeScript для последующего использования в ECMAScript

SoundScript challenges

- типизация должна быть эффективной, потому что время компиляции — это время исполнения

- должна поддерживаться ленивая компиляция — необходимо знать тип, возвращаемый функцией до анализа ее тела

- интероперабельность, т.к. новый код может быть добавлен в любой момент

Always bet on JS

- First they said JS couldn't be useful for building "rich Internet apps"
- Then they said it couldn't be fast
- Then they said it couldn't be fixed
- Then it couldn't do multicore/GPU
- Wrong every time!
- My advice: always bet on JS

Всегда ставьте на JavaScript
(Брендан Айк)

Вопросы?

github.com/chicoxyzzy

twitter.com/chicoxyzzy

ru.linkedin.com/in/chicoxyzzy

слайды: bit.ly/JS_frontendconf2015