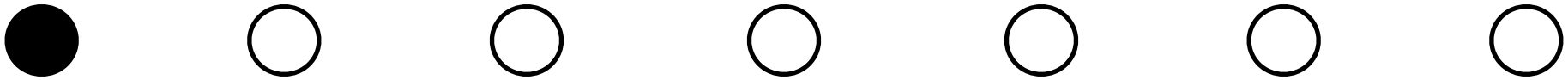
Rodrigo Branas – @rodrigobranas - http://www.agilecode.com.br

JavaScript Masterclass

Surgimento da linguagem e o caminho até a primeira especificação

ECMAScript 1
(110 páginas)



ES1
1997

Oficialização do que já havia sido feito
até o momento

Unicode Consortium. The Unicode Standard, Version 2.0. Addison-Wesley, Reading, Massachusetts (1996).

4 Overview

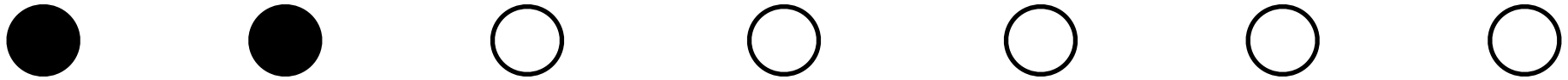
EMCAScript is an object-oriented programming language for performing computations and manipulating computational objects within a host environment. ECMAScript as defined here is not intended to be computationally self-sufficient; indeed, there are no provisions in this specification for input of external data or output of computed results. Instead, it is expected that the computational environment of an ECMAScript program will provide not only the objects and other facilities described in this specification but also certain environment-specific *host* objects, whose description and behavior are beyond the scope of this specification except to indicate that they may provide certain properties that can be accessed and certain functions that can be called from an ECMAScript program.

A ***scripting language*** is a programming language that is used to manipulate, customize, and automate the facilities of an existing system. In such systems, useful functionality is already available through a user interface, and the scripting language is a mechanism for exposing that functionality to program control. In this way, the existing system is said to provide a host environment of objects and facilities which completes the capabilities of the scripting language. A scripting language is intended for use by both professional and non-professional programmers, and therefore there may be a number of informalities and built into the language.

ECMAScript was originally designed to be a ***Web scripting language***, providing a mechanism to enliven Web pages in browsers and to perform server computation as part of a Web-based client-server architecture. ECMAScript can

provide core scripting capabilities for a variety of host environments, and therefore the core scripting language is specified in this document apart from any particular host environment.

ECMAScript 2
(117 páginas)

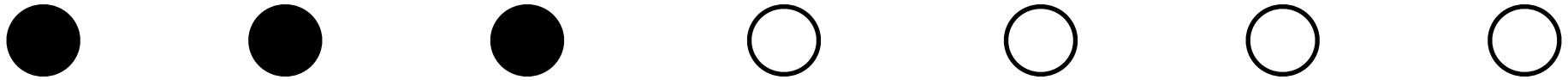


ES1
1997

ES2
1998

Adequação com a normativa ISO/IEC 16262

ECMAScript 3
(188 páginas)

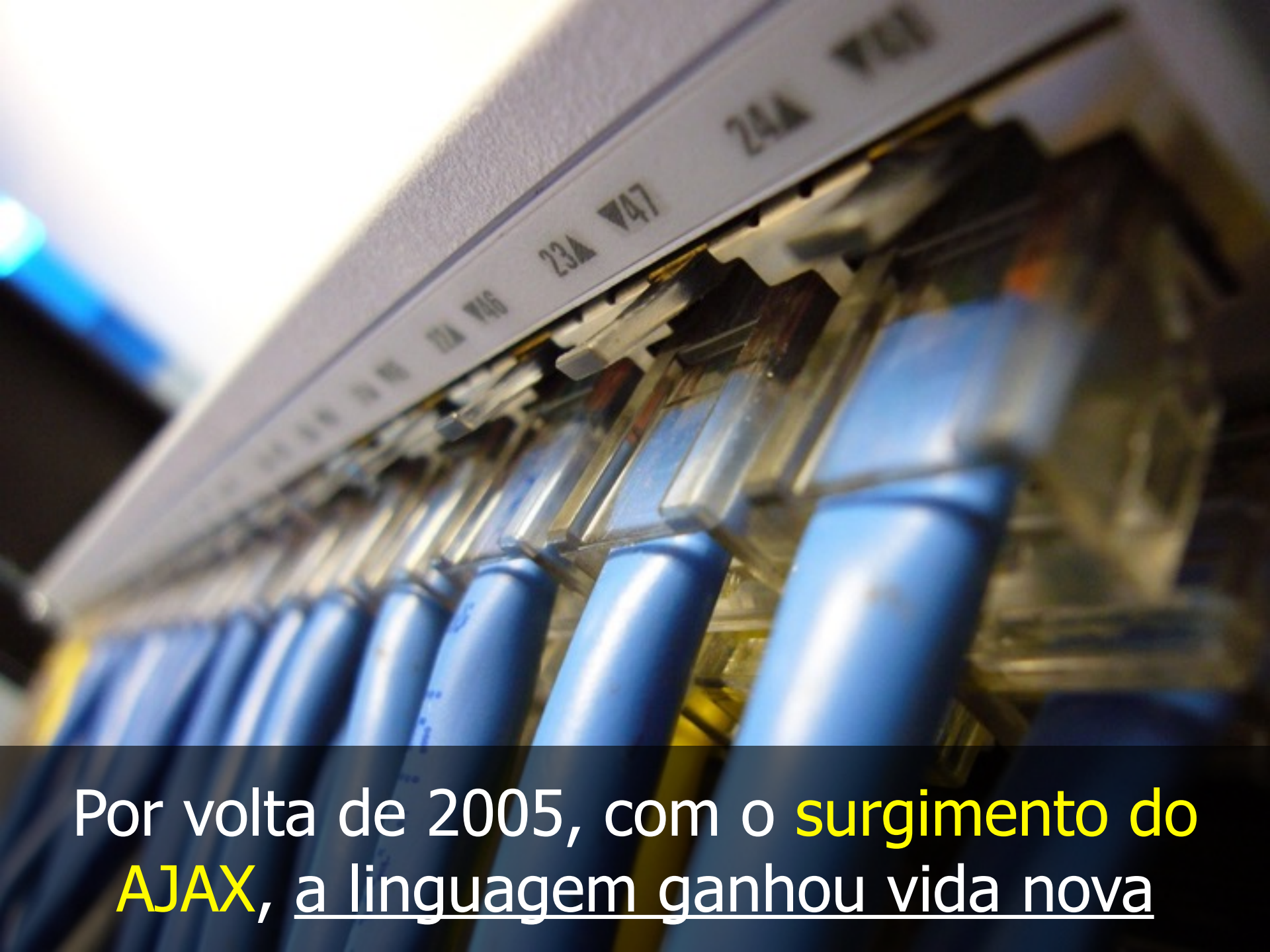


ES1
1997

ES2
1998

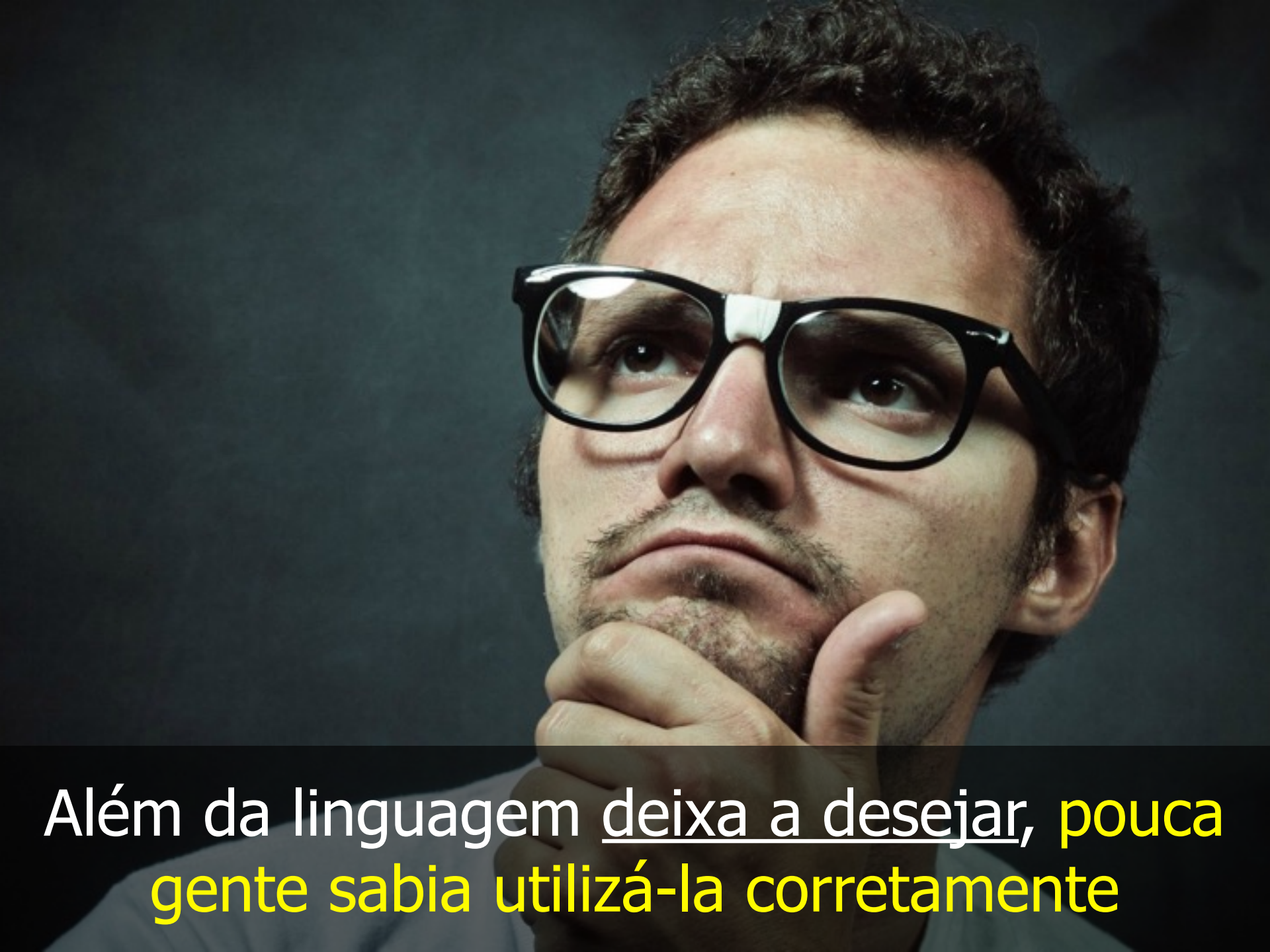
ES3
1999

Exception Handling (throw/try/catch), Regular Expression, switch, do-while, ...



Por volta de 2005, com o surgimento do AJAX, a linguagem ganhou vida nova

Esse período também foi conhecido como a era das trevas e do ódio


Além da linguagem deixa a desejar, pouca gente sabia utilizá-la corretamente

ECMAScript 5
(252 páginas)

ES1
1997

ES2
1998

ES3
1999

ES5
2009

JSON, strict mode, reserved words as property keys, multiline string, Object API, Array.prototype.*

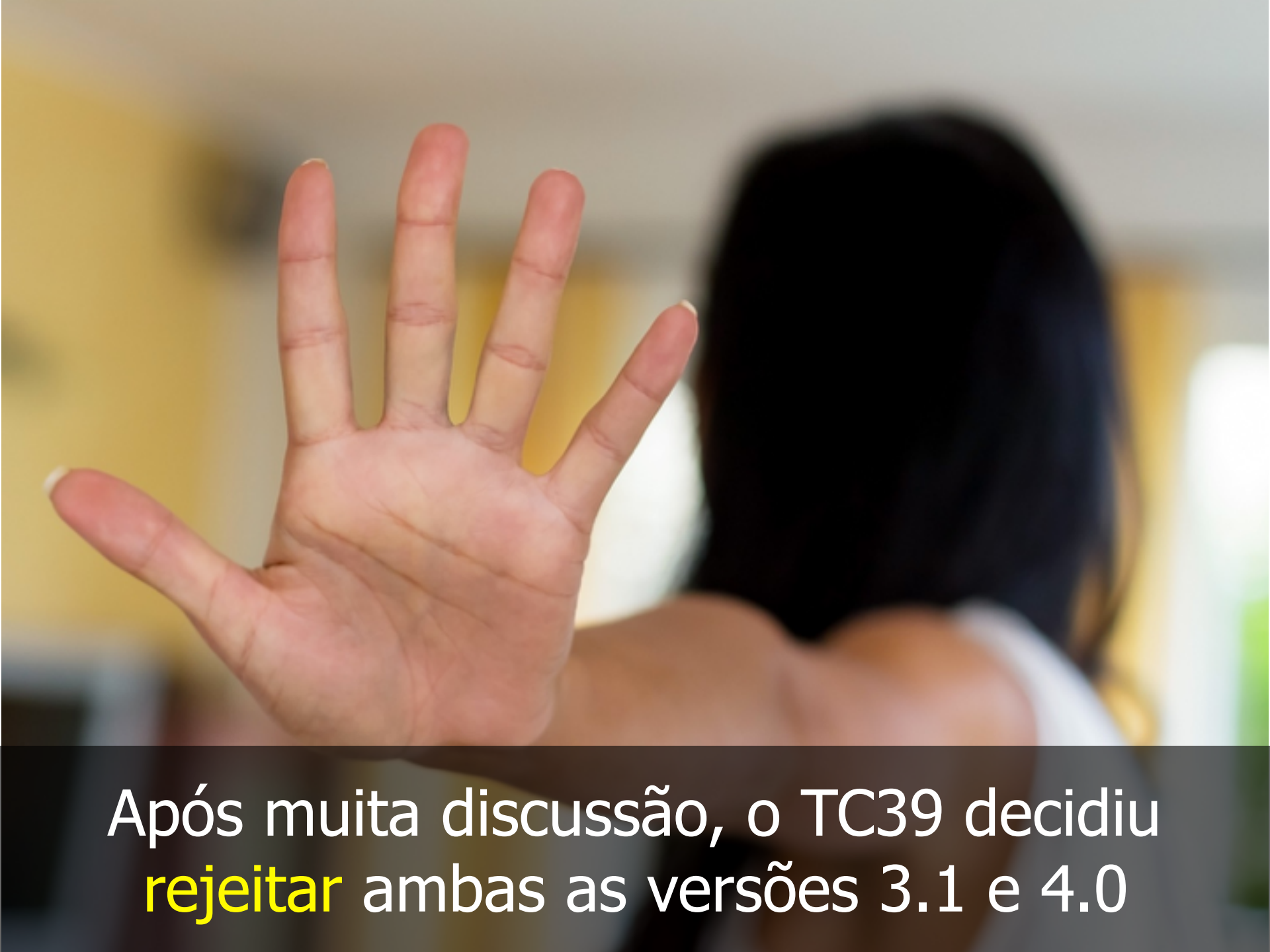
Se passaram 10 anos sem o lançamento de novas versões


Porque tanto tempo se passou, onde foi parar a versão 4?

Houve uma separação dos grupos que especificavam nas versões 3.1 e 4.0

Microsoft e Yahoo estavam na 3.1 e Adobe, Mozilla, Opera e Google na 4.0

Após mais de um ano de trabalho em paralelo, as versões não eram compatíveis



Após muita discussão, o TC39 decidiu rejeitar ambas as versões 3.1 e 4.0

ECMAScript Harmony

Brendan Eich brendan at mozilla.org
Wed Aug 13 14:26:56 PDT 2008

- Previous message: Old documents marked "New"
- Next message: ECMAScript Harmony
- **Messages sorted by:** [date] [thread] [subject] [author]

It's no secret that the JavaScript standards body, Ecma's Technical Committee 39, has been split for over a year, with some members favoring ES4, a major fourth edition to ECMA-262, and others advocating ES3.1 based on the existing ECMA-262 Edition 3 (ES3) specification. Now, I'm happy to report, the split is over.

The Ecma TC39 meeting in Oslo at the end of July was very productive, and if we keep working together, it will be seen as seminal when we look back in a couple of years. Before this meeting, I worked with John Neumann, TC39 chair, and ES3.1 and ES4 principals, especially Lars Hansen (Adobe), Mark Miller (Google), and Allen Wirfs-Brock (Microsoft), to unify the committee around shared values and a common roadmap. This message is my attempt to announce the main result of the meeting, which I've labeled "Harmony".

Executive Summary

The committee has resolved in favor of these tasks and conclusions:

1. Focus work on ES3.1 with full collaboration of all parties, and target two interoperable implementations by early next year.
2. Collaborate on the next step beyond ES3.1, which will include syntactic extensions but which will be more modest than ES4 in both semantic and syntactic innovation.

ECMAScript 5.1

(258 páginas)

ES1
1997

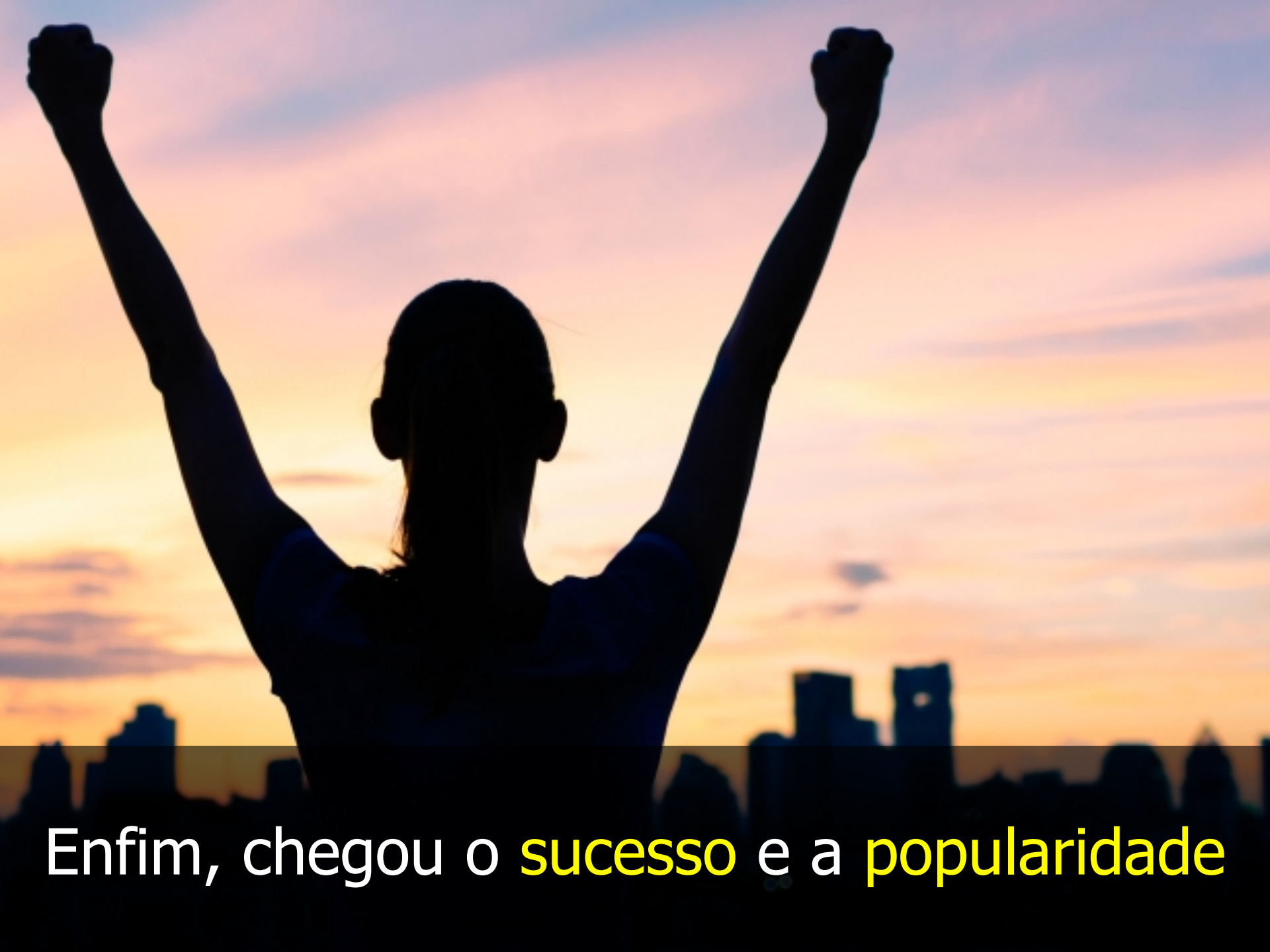
ES2
1998

ES3
1999

ES5
2009

ES5.1
2011

Adequação com a normativa ISO/IEC 16262


Enfim, chegou o sucesso e a popularidade


JS
REQUIRE·JS
npm
TS
BABEL
JS
REQUIRE·JS
npm
TS
REQUIRE·JS

ECMAScript 6 / ECMA2015

(566 páginas)

ES1	ES2	ES3	ES5	ES5.1	ES6	
1997	1998	1999	2009	2011	2015	

Class, Arrow Function, Proxy, Reflect, Map, Set, Destructuring, Rest Parameter, Default Value, Template Literal, Spread Operator, Generators, Promises, Modules

ECMAScript 7 / ECMA2016

(586 páginas)

ES1 1997 | ES2 1998 | ES3 1999 | ES5 2009 | ES5.1 2011 | ES6 2015 | ES7 2016

Async/Await, Object.values, Object.entries, String.prototype.padStart, String.prototype.padEnd, ...

ES.next

Qual é o drama com relação a questão da compatibilidade?

			Compilers/polyfills											
		97%	56%	71%	48%	59%	18%	5%	11%	83%	93%	95%	86%	
Feature name ▸	Current browser	Traceur	Babel + core-js[2]	Closure	Type-Script + core-js	es6-shim	KQ 4.14[3]	IE 11	Edge 13[4]	Edge 14[4]	Edge 15[4]	FF 45 ESR	FF
Optimisation													
proper tail calls (tail call optimisation) ▸	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0
Syntax														
default function parameters ▸	7/7	4/7	4/7	4/7	5/7	0/7	0/7	0/7	0/7	7/7	7/7	4/7	6	
rest parameters ▸	5/5	4/5	3/5	2/5	4/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5	
spread (...) operator ▸	15/15	15/15	13/15	12/15	4/15	0/15	0/15	0/15	15/15	15/15	15/15	15/15	15	
object literal extensions ▸	6/6	6/6	6/6	4/6	6/6	0/6	0/6	0/6	6/6	6/6	6/6	6/6	6	
for..of loops ▸	9/9	9/9	9/9	6/9	3/9	0/9	0/9	0/9	7/9	7/9	9/9	7/9	7	
octal and binary literals ▸	4/4	2/4	4/4	4/4	4/4	2/4	0/4	0/4	4/4	4/4	4/4	4/4	4	
template literals ▸	5/5	4/5	4/5	3/5	3/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5	
RegExp "y" and "u" flags ▸	5/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	5/5	5/5	5/5	2/5	5	
destructuring, declarations ▸	22/22	20/22	21/22	19/22	15/22	0/22	0/22	0/22	0/22	21/22	22/22	19/22	21	
destructuring, assignment ▸	24/24	23/24	24/24	17/24	19/24	0/24	0/24	0/24	0/24	23/24	24/24	21/24	23	
destructuring, parameters ▸	23/23	19/23	20/23	18/23	15/23	0/23	0/23	0/23	0/23	22/23	23/23	18/23	20	
Unicode code point escapes ▸	2/2	1/2	1/2	1/2	1/2	0/2	0/2	0/2	2/2	2/2	2/2	1/2	1	
new.target ▸	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	1/2	2/2	2/2	2/2	2	
Bindings														
const ▸	16/16	14/16	14/16	14/16	14/16	0/16	2/16	12/16	12/16	16/16	16/16	12/16	16	
let ▸	12/12	10/12	10/12	10/12	10/12	0/12	0/12	10/12	10/12	12/12	12/12	10/12	12	
block-level function declaration[12]	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	No	Y	
Functions														
arrow functions ▸	13/13	11/13	9/13	10/13	9/13	0/13	0/13	0/13	13/13	13/13	13/13	13/13	13	
class ▸	24/24	17/24	19/24	13/24	19/24	0/24	0/24	0/24	24/24	24/24	24/24	24/24	24	
super ▸	8/8	7/8	4/8	5/8	7/8	0/8	0/8	0/8	8/8	8/8	8/8	8/8	8	
generators ▸	27/27	24/27	24/27	16/27	0/27	0/27	0/27	0/27	27/27	27/27	27/27	25/27	25	