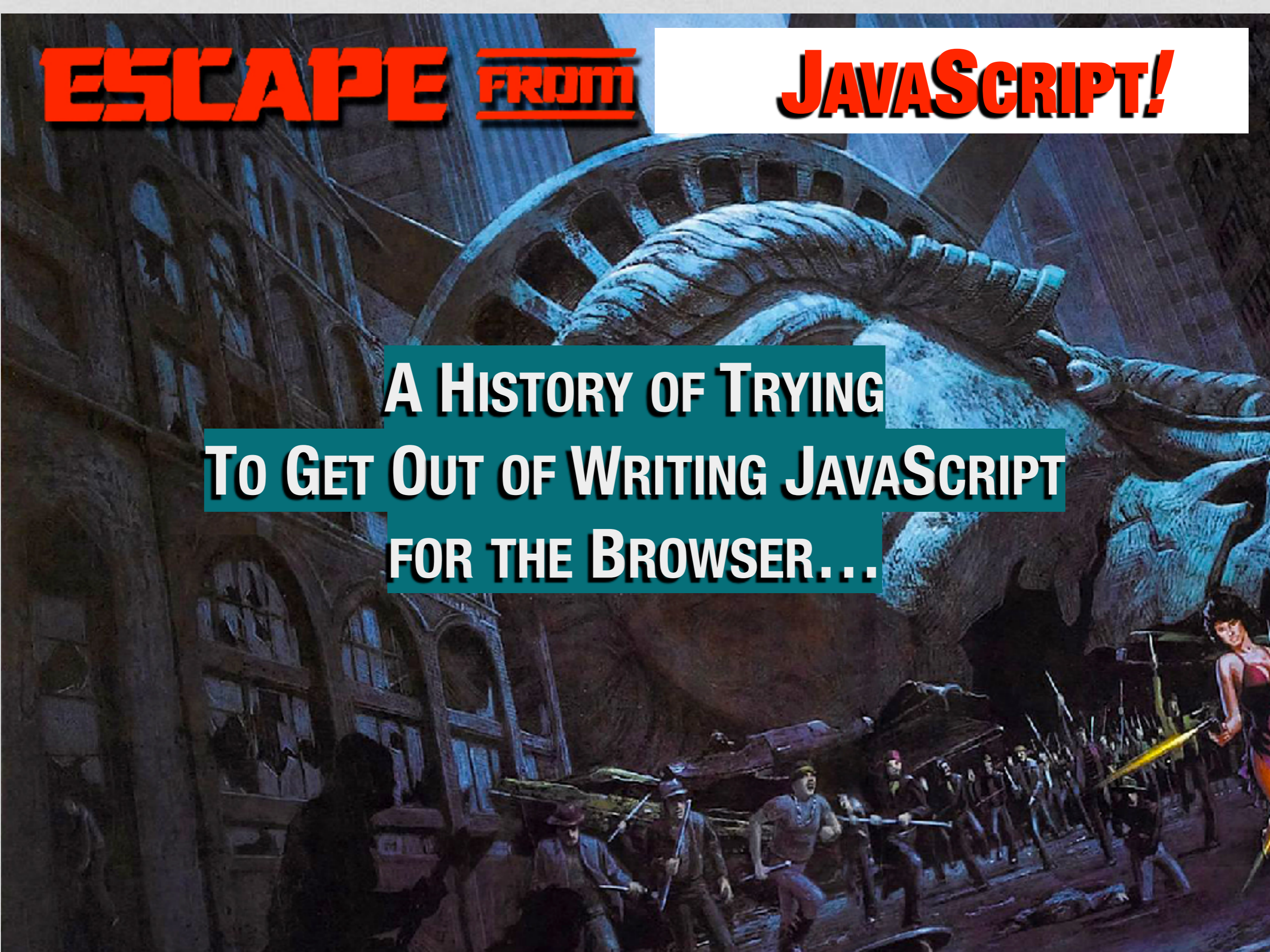


ESCAPE FROM

JAVASCRIPT!

**A HISTORY OF TRYING
TO GET OUT OF WRITING JAVASCRIPT
FOR THE BROWSER...**





```
failbowl:~(master!?) $ jsc
```

```
> [] + []
```

```
> [] + {}
```

```
[object Object]
```

```
> {} + []
```

```
0
```

```
> {} + {}
```

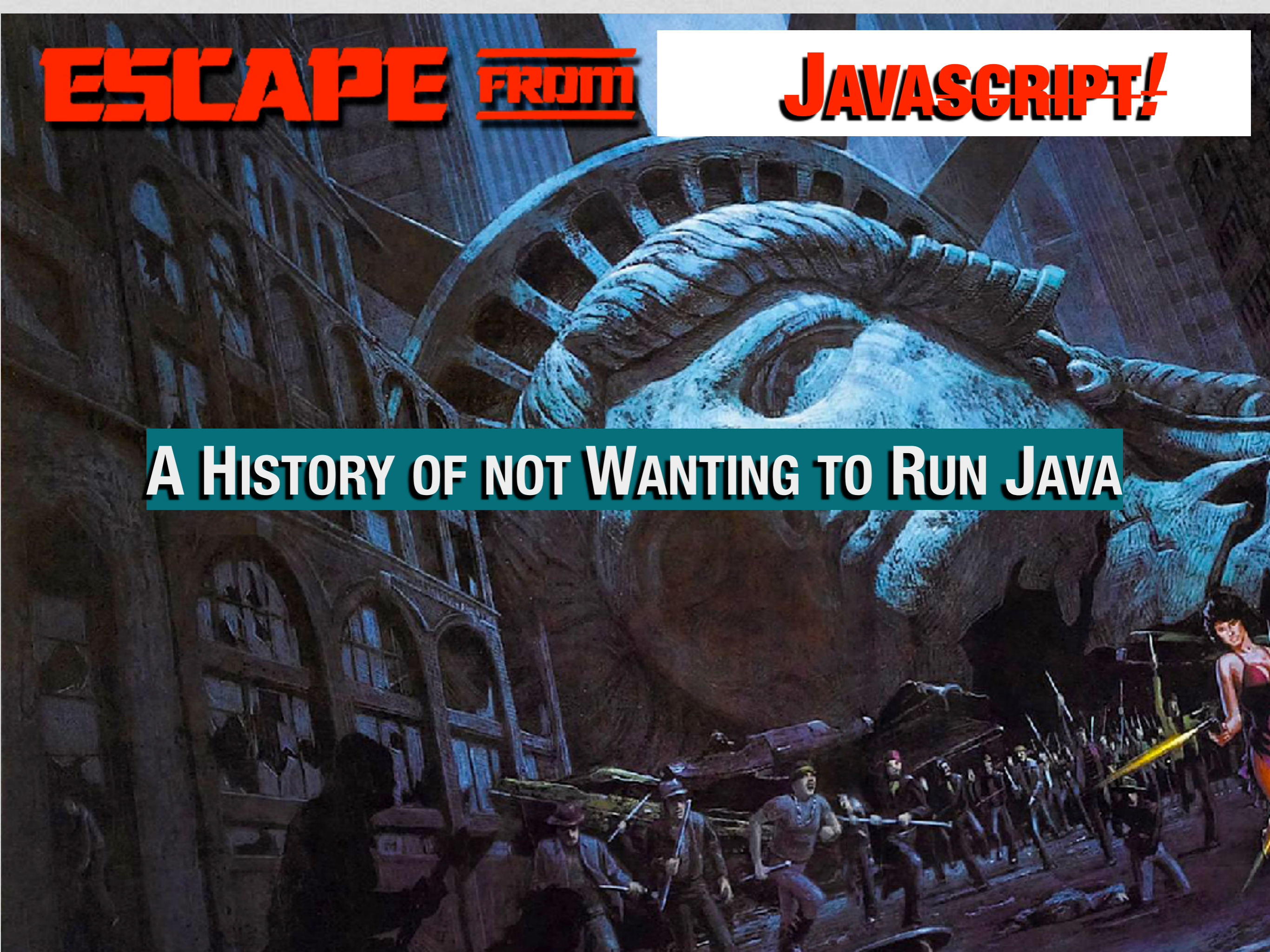
```
NaN
```

```
> █
```


ESCAPE FROM

JAVASCRIPT!

A HISTORY OF NOT WANTING TO RUN JAVA



- ❑ **1995: JavaScript (aka Mocha, LiveScript) invented**
- ❑ **1996: Microsoft releases JScript, ActiveX, and VBScript**
- ❑ **1997: ECMAScript version 1**

Search

 Go

[Advanced Search »](#)

Sponsored Developer Resources

[Inside Lightroom](#)

Web Columns

[Adobe GoLive](#)
[Essential JavaScript](#)
[Megnut](#)

Web Topics

[All Articles](#)
[Browsers](#)
[ColdFusion](#)
[CSS](#)
[Database](#)
[Flash](#)
[Graphics](#)
[HTML/XHTML/DHTML](#)
[Scripting Languages](#)
[Tools](#)
[Weblogs](#)

Sites

[Beautiful Code](#)
[Databases](#)
[Digital Media](#)
[Emerging Telephony](#)
[Inside Aperture](#)
[Inside Lightroom](#)
[Inside Port 25](#)
[InsideRIA.com](#)
[LinuxDevCenter.com](#)
[MacDevCenter.com](#)
[ONJava.com](#)
[ONLamp.com](#)
[Apache](#)
[BSD](#)
[MySQL](#)

JavaScript: How Did We Get Here?

by [Steve Champeon](#)
04/06/2001

JavaScript. Spawned in 1995 by the need to make Netscape Navigator's newly added support for Java applets more accessible to non-Java programmers and web designers, a powerful scripting language too often described as "simple."

Plagued in its early days by security flaws, crippled by a lack of powerful development tools such as integrated development environments, debuggers, and meaningful error messages, extended to contexts that range far beyond the initial intent of its designers, and saddled with the legacy of incompatible browser object models, JavaScript has suffered for years at the hands of those who would criticize it for being too unlike Java, or too much like Perl, or too often used by well-meaning but otherwise ignorant web designers, shoehorned into pages without thought of future compatibility, intelligent abstraction, or code reuse.

Yet it is by far the most popular language on the Web, the foundation for the next generation of dynamic client-side Web applications, a solid core with amazing potential and power. So why don't more programmers view JavaScript as an essential part of their toolbox? Let's look at the history of JavaScript (née LiveScript) and take a good, hard look at where it came from -- and where it is going.

Back to the early days of the web

Rewind to early 1995. Netscape had just hired Brendan Eich away from MicroUnity Systems Engineering, to take charge of the design and implementation of a new language. Tasked with making Navigator's newly added Java support more accessible to non-Java programmers, Eich eventually decided that a loosely-typed scripting language suited the environment and audience, namely the few thousand web designers and developers who needed to be able to tie into page elements (such as forms, or frames, or images) without a bytecode compiler or knowledge of object-oriented software design.

The language he created was christened "LiveScript," to reflect its dynamic nature, but was quickly (before the end of the Navigator 2.0 beta cycle) renamed JavaScript, a mistake driven by marketing that would plague web designers for years to come, as they confused the two incessantly on mailing lists and on Usenet. Netscape and Sun jointly announced the new language on December 4, 1995, calling it a "complement" to both HTML and Java.

[Print](#) [Subscribe to Newsletters](#)



[ShareThis](#)

Related Reading



[JavaScript: The Definitive Guide, 4th Edition](#)

By David Flanagan
[Table of Contents](#)
[Index](#)
[Simple Chapter](#)

framers of the W3C DOM). The desire to get the web back on track as an SGML-inspired platform, where document structure encodes semantics but not presentation, fueled by the relative simplicity and power of XML, led to another layer of abstraction and several years of incompatible or partial implementations.

The open source Mozilla project took years without a stable or widely distributed public release, while Microsoft tightened its hold on the browser market share. DHTML programmers retrenched, for the most part, in response to the complexity of cross-browser DHTML, a new emphasis on standards support, and the rise of other tools for providing interactive content, such as Flash.

Where does this leave us today?

Fast forward to 2001. JavaScript's dependence on context-specific object models is both the core of its strength and the fatal flaw in its implementation. Unlike Perl or Java or other languages, JavaScript's capabilities can't be extended or overridden by developers, who are instead utterly dependent on the caprice of the browser vendors. JavaScript is flexible enough to overcome many weaknesses, but not enough to overcome fundamental, context-enlaved object model incompatibilities.

Also, the inability to hide source continues to be an obstacle to the development of applications whose code the owners wish to protect as their intellectual property. Continuing lack of an IDE, debugger, and other standard development tools are still considered insurmountable problems by many programmers, who defend their Windows-only applications with the not unreasonable excuse that debugging is too hard on a platform that lacks a debugger. The WYSIWYG tools vendors caught up with some excellent cross-browser JavaScript libraries integrated into their page building tools, only to be caught out by Mozilla's abandonment of the old Netscape document.layers object model.

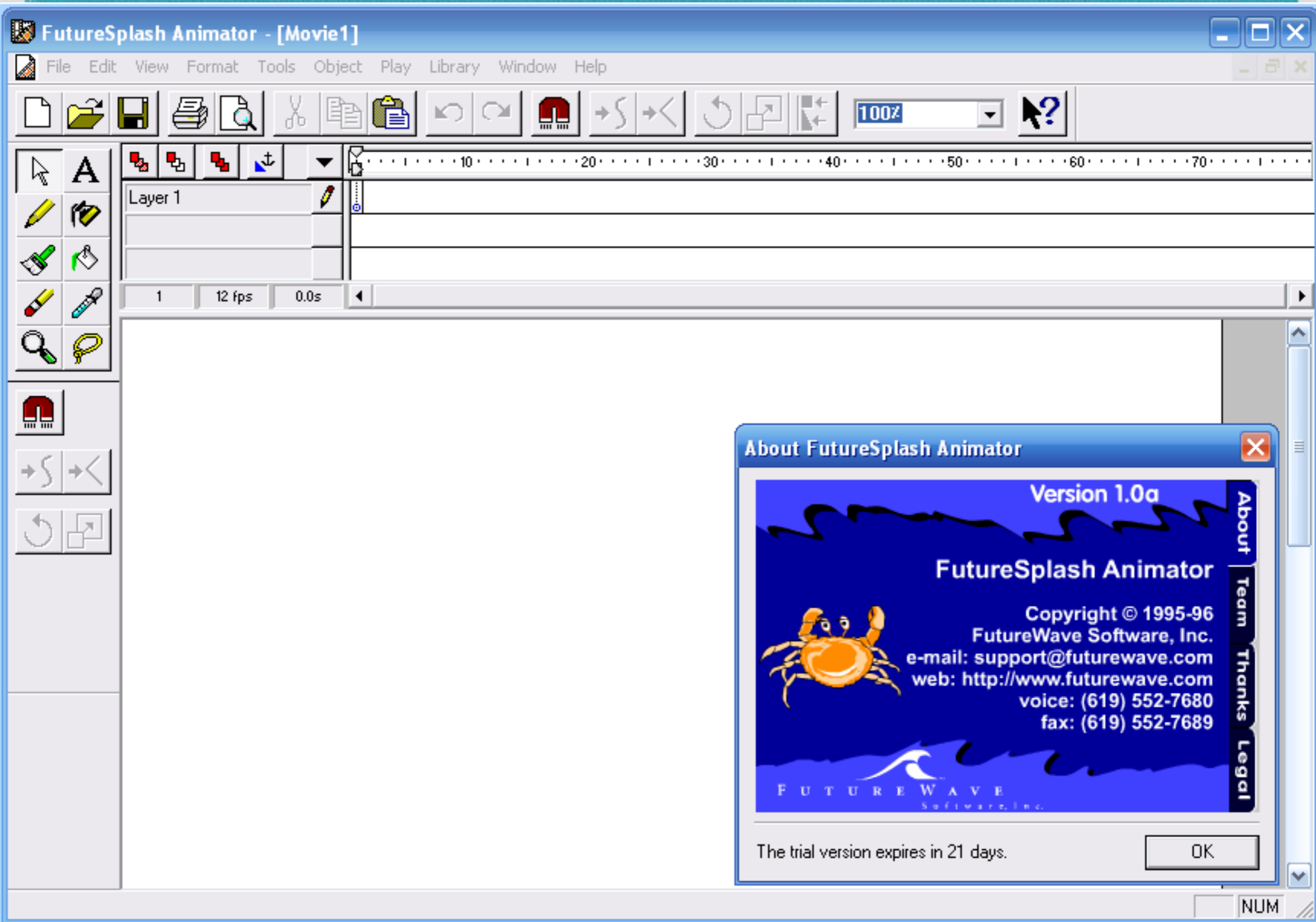
And yet, JavaScript has evolved into an incredibly powerful mélange of the best features of Perl (associative arrays, loosely typed variables, regular expressions), C/C++ and Java (clean, block-parsed syntax, objects and classes, highly evolved date, math, and string libraries), and TCL (widely ported application control environment), to name a few, with all the power of the W3C DOM in the latest browsers (Mozilla, Netscape 6, IE6/Windows and IE5/Mac).

Companies such as Blox, KnowNow, and others are using client-side JavaScript in combination with intelligent server-side tools to deliver the next generation of Web-based applications. Even Microsoft, with all the ballyhoo about C#, seems to have recognized the power of JavaScript: Pick any MSDN sample application or one-off script, and odds are it's written at least partly in JavaScript.

Java has suffered at the hands of internecine and highly publicized feuds between Sun and, well, pretty much everyone. VBScript is being abandoned by Microsoft, but the compiled C# can't be expected to fill the ever-needed role of scripting language. JavaScript has conquered the Web. So why don't you know more about it? What is there to know? How should you begin to address your long-standing neglect of this powerful and mature language? We'll address these issues and more in our next article.

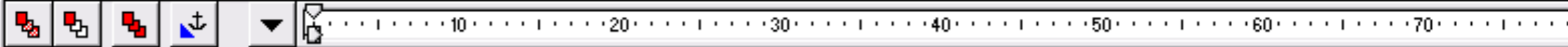
[Steve Champeon](#) is a recognized developer, author, and editor specializing in Web technologies. At his "day job," he serves as the CTO of [hesketh.com](#), a Web services firm in Raleigh, NC.

Return to the [JavaScript and CSS DevCenter](#).



FutureSplash Animator - [Movie1]

File Edit View Format Tools Object Play Library Window Help



About FutureSplash Animator [X]

Version 1.0a

FutureSplash Animator

Copyright © 1995-96
FutureWave Software, Inc.
e-mail: support@futurewave.com
web: http://www.futurewave.com
voice: (619) 552-7680
fax: (619) 552-7689

FutureWave Software, Inc.

About Team Thanks Legal

The trial version expires in 21 days. [OK]

NUM

- 1999: XMLHttpRequest released**
- 2002: Discovery of JSON by Douglas Crockford**
- 2004: Gmail released**
- 2005: Google Maps released, AJAX named**

The logo for the Prototype JavaScript framework, featuring the word "prototype" in a lowercase, sans-serif font. The letter "o" is orange, while the other letters are blue. The logo is centered within a white rounded rectangle with a subtle drop shadow.

Download

1.7.3 (September 22, 2015) & Git

Learn

Docs & tutorials

Discuss

Mailing lists & IRC

Contribute

Report bugs & patches

[Weblog](#)

[Prototype 1.7.3](#)

The new bugfix release of Prototype features lots of tiny fixes and one giant change under the hood.

A foundation for ambitious web user interfaces.

Prototype takes the complexity out of client-side web programming. Built to solve real-world problems, it adds useful extensions to the browser scripting environment and provides elegant APIs around the clumsy interfaces of Ajax and the Document Object Model.

Getting started: [Defining classes and inheritance](#) • [How Prototype extends the DOM](#) • [Introduction to Ajax](#) • [Using JSON](#) • [Event delegation](#) • [Using Element.Layout](#)

MooTools is a collection of JavaScript utilities designed for the intermediate to advanced JavaScript developer. It allows you to write powerful and flexible code with its elegant, well documented, and coherent APIs.

MooTools code is extensively documented and easy to read, enabling you to extend the functionality to match your requirements.

Open Source License

MooTools libraries are released under the [Open Source MIT license](#) which gives you the possibility to use them and modify them in every circumstance.

How to use?

MooTools Selectors

Selectors for DOM Elements

```
// get elements by class
$$('.foo'); // or even:
document.getElements('.foo');

// selector with different elements
$$('div.foo, div.bar, div.bar a');

// get a single element
document.getElement('div.foo');
```

Ajax!

MooTools uses a Class called Request.

```
// create a new Class instance
var myRequest = new Request({
  url: 'getMyText.php',
  method: 'get',
  onRequest: function(){
    myElement.set('text', 'loading...');
  },
  onSuccess: function(responseText){
    myElement.set('text', responseText);
  },
  onFailure: function(){
    myElement.set('text', 'Sorry, your request
```



Your donations help fund the continued development and growth of jQuery.

SUPPORT THE PROJECT



Lightweight Footprint

Only 32kB minified and gzipped. Can also be included as an AMD module



CSS3 Compliant

Supports CSS3 selectors to find elements as well as in style property manipulation



Cross-Browser

[Chrome, Edge, Firefox, IE, Safari, Android, iOS, and more](#)

 **Download jQuery**
v3.2.1

The 1.x and 2.x branches no longer receive patches.

[View Source on GitHub](#) →
[How jQuery Works](#) →

What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Corporate Members

Resources

- [jQuery Core API Documentation](#)
- [jQuery Learning Center](#)
- [jQuery Blog](#)
- [Contribute to jQuery](#)
- [About the jQuery Foundation](#)
- [Browse or Submit jQuery Bugs](#)

Underscore.js (1.8.3)

- » [GitHub Repository](#)
- » [Annotated Source](#)
- » [Underscore-contrib](#)

Filter

Introduction

Collections

- each
- map
- reduce
- reduceRight
- find
- filter
- where
- findWhere
- reject
- every
- some
- contains
- invoke
- pluck
- max
- min
- sortBy
- groupBy
- indexBy
- countBy
- shuffle
- sample
- toArray
- size
- partition

Arrays

- first
- initial
- last
- rest

UNDERSCORE.JS

Underscore is a JavaScript library that provides a whole mess of useful functional programming helpers without extending any built-in objects. It's the answer to the question: "If I sit down in front of a blank HTML page, and want to start being productive immediately, what do I need?" ... and the tie to go along with [jQuery's](#) tux and [Backbone's](#) suspenders.

Underscore provides over 100 functions that support both your favorite workaday functional helpers: **map**, **filter**, **invoke** — as well as more specialized goodies: function binding, javascript templating, creating quick indexes, deep equality testing, and so on.

A complete [Test Suite](#) is included for your perusal.

You may also read through the [annotated source code](#).

Enjoying Underscore, and want to *turn it up to 11*? Try [Underscore-contrib](#).

The project is [hosted on GitHub](#). You can report bugs and discuss features on the [issues page](#), on Freenode in the [#documentcloud](#) channel, or in our [Gitter](#) channel.

Underscore is an open-source component of [DocumentCloud](#).

Downloads *(Right-click, and use "Save As")*

[Development Version \(1.8.3\)](#) 52kb, Uncompressed with Plentiful Comments

[Production Version \(1.8.3\)](#) 5.7kb, Minified and Gzipped ([Source Map](#))

Edge Version *Unreleased* [current](#) [master](#) *use at your own risk*

Unearthing the Excellence in JavaScript



JavaScript: The Good Parts

JavaScript: The Good Parts

Crockford



YAHOO! PRESS

O'REILLY

JavaScript

The Definitive Guide

Flanagan

FIFTH EDITION

COVERS AJAX AND DOM SCRIPTING



O'REILLY

[Overview](#)[CoffeeScript 2](#)[What's New in CoffeeScript 2](#)[Compatibility](#)[Installation](#)[Usage](#)[Command Line](#)[Node.js](#)[Transpilation](#)[Language Reference](#)[Functions](#)[Strings](#)[Objects and Arrays](#)[Comments](#)[Lexical Scoping and Variable Safety](#)[If, Else, Unless, and](#)[Conditional Assignment](#)[Splats, or Rest](#)[Parameters/Spread Syntax](#)[Loops and Comprehensions](#)[Array Slicing and Splicing](#)

CoffeeScript

CoffeeScript is a little language that compiles into JavaScript. Underneath that awkward Java-esque patina, JavaScript has always had a gorgeous heart. CoffeeScript is an attempt to expose the good parts of JavaScript in a simple way.

The golden rule of CoffeeScript is: *"It's just JavaScript."* The code compiles one-to-one into the equivalent JS, and there is no interpretation at runtime. You can use any existing JavaScript library seamlessly from CoffeeScript (and vice-versa). The compiled output is readable, pretty-printed, and tends to run as fast or faster than the equivalent handwritten JavaScript.

Latest Version: [2.0.3](#)

```
# Install locally for a project:  
npm install --save-dev coffeescript
```

```
# Install globally to execute .coffee files anywhere:  
npm install --global coffeescript
```



- ↔ Overview
- » Get started

- ↔ Tutorials
- ↔ Docs
- ↔ Resources
- ↔ Make GWT Better
- ↔ Terms

⌵ Download

JD Java Doc



Creative Commons Attribution
3.0 License.

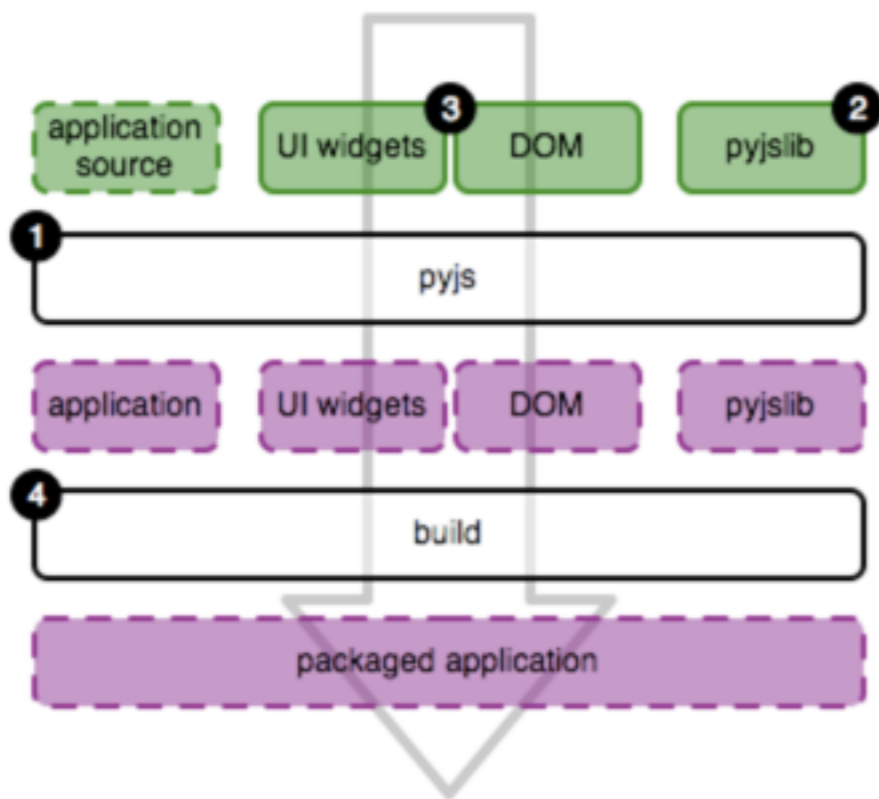
Productivity for developers, performance for users

GWT is used by many products at Google, including Google AdWords and Google Wallet. It's open source, completely free, and used by thousands of enthusiastic developers around the world.



Overview

Like GWT, pyjs translates a Python application and libraries (including UI widgets and DOM classes) into a JavaScript application and libraries, and packages it all up:



1. **pyjs** translates Python code to JavaScript by walking the Python abstract syntax tree and generating JavaScript.
2. **pyjslib** takes care of Python types that require a custom JavaScript implementation for pyjs to work. For example, even though Python lists are similar to JavaScript arrays, Python lists are converted to custom objects that implement methods such as `append`.
3. **UI widgets and a DOM library** are provided as a convenience. They are written in Python and, like everything else, they are translated to JavaScript for deployment. These are based on those in GWT.
4. **build** manages the overall translation of individual components and creates a set of `.html` and `.js` files that can be served up by a Web server.

Features

This is a list of the overall pyjs features, including the Python-to-JavaScript translator as well as the UI Widget Set. For specific language features supported by the Python-to-JavaScript translator, see the [translator](#) description.

About

About
Overview
Translator
Download
Getting Help

Documentation


Examples
UI Hierarchy
API Docs
Book
Wiki

Development

Develop
Optimize
Contribute
Roadmap


[Overview](#)[Closure Compiler Service UI
Getting Started](#)[Closure Compiler Service API
Getting Started
Communicating with the API
Compressing Files with the API](#)[Closure Compiler Application
Getting Started](#)[Advanced Topics
Compilation Levels
Restrictions Imposed by the Compiler
Advanced Compilation
Annotating JavaScript for the Compiler](#)**Contents**[What is the Closure Compiler?](#)
[How can I use the Closure Compiler?](#)
[What are the benefits of using Closure Compiler?](#)
[How do I start?](#)

What is the Closure Compiler?



The Closure Compiler is a tool for making JavaScript download and run faster. Instead of compiling from a source language to machine code, it compiles from JavaScript to better JavaScript. It parses your JavaScript, analyzes it, removes dead code and rewrites and minimizes what's left. It also checks syntax, variable references, and types, and warns about common JavaScript pitfalls.

How can I use the Closure Compiler?



You can use the Closure Compiler as:

- An open source Java application that you can run from the command line.
- A simple web application.
- A RESTful API.

To get started with the compiler, see "How do I start" below.



```
import 'dart:async';
import 'dart:math' show Random;

main() async {
  print('Compute  $\pi$  using the Monte Carlo method.');
```

```
  await for (var estimate in computePi().take(500)) {
    print('π ≈ $estimate');
  }
}

/// Generates a stream of increasingly accurate estimates of π.
Stream<double> computePi({int batch: 100000}) async* {
  var total = 0;
  var count = 0;
  while (true) {
    var points = generateRandom().take(batch);
    var inside = points.where((p) => p.isInsideUnitCircle);
    total += batch;
  }
}
```

[Open in DartPad](#)

Dart is an application programming language that's easy to learn, easy to scale, and deployable everywhere.

Google depends on Dart to make very large apps.

[Get Started](#)[Install Dart](#)

[**Click underlined text or code to learn more.**]

[News](#)

Follow the latest.

[API](#)

Browse core libraries.

[Pub](#)

Find packages.

[Dart webdev](#)

Build browser apps.

[Flutter](#)

Build mobile apps.

[Dart VM](#)

Build scripts, servers.

Core goals



[Home](#)
[Chromium](#)
[Chromium OS](#)

Quick links

[Report bugs](#)
[Discuss](#)
[Sitemap](#)

Other sites

[Chromium Blog](#)
[Google Chrome Extensions](#)
[Google Chrome Frame](#)

Except as otherwise [noted](#), the content of this page is licensed under a [Creative Commons Attribution 2.5 license](#), and examples are licensed under the [BSD License](#).

[Native Client](#) >

PNaCl

Subpage Listing

[Aligned bundling support in LLVM](#)

[Building and using PNaCl components for Chromium distribution packagers](#)

[Developing PNaCl](#)

[Experimenting with generated bitcode](#)

[Introduction to Portable Native Client](#)

[PNaCl Bitcode Reference Manual](#)

[Stability of the PNaCl bitcode ABI](#)

▼ [Subzero](#)

[Simple loop example](#)

Subpages (8): [Aligned bundling support in LLVM](#) [Building and using PNaCl components for Chromium distribution packagers](#) [Developing PNaCl](#) [Experimenting with generated bitcode](#) [Introduction to Portable Native Client](#) [PNaCl Bitcode Reference Manual](#) [Stability of the PNaCl bitcode ABI](#) [Subzero](#)

Comments

You do not have permission to add comments.

asm.js

Working Draft — 18 August 2014

Latest version:

<http://asmjs.org/spec/latest/>

Editors:

David Herman, Mozilla, <dherman@mozilla.com>

Luke Wagner, Mozilla, <luke@mozilla.com>

Alon Zakai, Mozilla, <azakai@mozilla.com>

Abstract

This specification defines **asm.js**, a strict subset of JavaScript that can be used as a low-level, efficient target language for compilers. This sublanguage effectively describes a sandboxed virtual machine for memory-unsafe languages like C or C++. A combination of static and dynamic validation allows JavaScript engines to employ an ahead-of-time (AOT) optimizing compilation strategy for valid asm.js code.

Status

This specification is working towards a candidate draft for asm.js version 1. Mozilla's SpiderMonkey JavaScript engine provides an optimizing implementation of this draft.

Changelog

ASM.JS

```
function strlen(ptr) { // calculate length of C string
  ptr = ptr|0;
  var curr = 0;
  curr = ptr;
  while (MEM8[curr]|0 != 0) {
    curr = (curr + 1)|0;
  }
  return (curr - ptr)|0;
}
```

- Ensure that ptr is always an integer
- Read an integer from address curr
- Additions and subtractions are all 32-bit

[Introducing Emscripten](#)[Getting Started](#)[Porting](#)[Optimizing Code](#)[Optimizing WebGL](#)[Profiling the Toolchain](#)[Compiling and Running Projects](#)[Building Emscripten from Source](#)[Contributing to Emscripten](#)[API Reference](#)[CyberDWARF Debugging](#)[Tools Reference](#)[About this site](#)[Index](#)[Home » Main](#)

emscripten

Emscripten is an LLVM-based project that compiles C and C++ into highly-optimizable JavaScript in asm.js format. This lets you run C and C++ on the web at near-native speed, without plugins.

Porting

Compile your existing projects written in C or C++ and run them on all modern browsers.

APIs

Emscripten converts OpenGL into WebGL, and lets you use familiar APIs like SDL, or HTML5 directly.

Fast

Thanks to LLVM, Emscripten and [asm.js](#), code runs at near-native speed.

Interested to learn more? Read our [About Page!](#)

Ready to get started? [Download and install the SDK](#) and then proceed to the [Tutorial!](#)

List of languages that compile to JS

Adrian Castravete edited this page Nov 2, 2017 · 565 revisions

CoffeeScript + Family & Friends

Name	Description
CoffeeScript	Unfancy JavaScript
CoffeeScript II: The Wrath of Khan	Rewrite of the CS compiler

Family (share genes with CoffeeScript)

Name	Description
Coco	A CoffeeScript dialect that aims to be more radical and practical, also acts as a test bed for features that get imported in CoffeeScript.
LiveScript	A fork of Coco that is much more compatible with CoffeeScript, more functional, and with more features.
IcedCoffeeScript	A CoffeeScript dialect that adds support for <code>await</code> and <code>defer</code> keywords which simplify async control flow.
Parsec CoffeeScript	CS based on parser combinators. The project's aim is to add static metaprogramming (i.e. macros + syntax extensibility) to Coffee Script (CS), similar to how Metalua adds such features to Lua. The resulting compiler, once merged with the official compiler, should

▼ Pages 14

[Home](#)

[\[HowTo\] Compiling and Setting Up Build Tools](#)

[\[HowTo\] Hacking on the CoffeeScript Compiler](#)

[\[HowTo\] How parsing works](#)

[\[HowTo\] Publish to NPM](#)

[Build tools](#)

[CoffeeScript 2 Honor Roll](#)

[Common Gotchas](#)

[FAQ](#)

[In The Wild](#)

[List of languages that compile to JS](#)

[Text editor plugins](#)

[Uniform Type Identifiers](#)

[Web framework plugins](#)

Friends (philosophically related)

Name	Description
NodeScript	JavaScript without the Variable Declarations and Semicolons
Bizubee	The World's Most Intense Programming Language!
Kaffeine	Enhanced Syntax for JavaScript.
Moescript (dead link (https://github.com/moescript/moescript))	Indent-based language
pogoscript	Language that emphasises readability, handles async control flow nicely, is friendly to domain specific languages and compiles to regular JavaScript
LispyScript	A JavaScript with Lispy syntax and Macros. <i>[o](#other "this item appears twice in this list; jump to its other instance")</i>
Hot Cocoa Lisp	A Lisp-like language that compiles to JavaScript. <i>[o](#other "this item appears twice in this list; jump to its other instance")</i>
Sibilant	JavaScript with a lisp. <i>[o](#other "this item appears twice in this list; jump to its other instance")</i>
	Clojure-like syntax, mori 's immutable data structures in a few sweet.js macros. Can

JavaScript Extensions

Security enforcing JavaScript

Name	Description
Caja	Compiles ES5/strict to ES3 and supports object-capabilities
ADsafe	Client-side static verifier and API, making third party scripts safe.
Jacaranda	Static verifier supporting object-capabilities.
Dojo Secure	Framework for building secure mashups.

Static typing

NOTE: Some of the projects listed below are also statically typed, such as [mobl](#), [GWT](#), [JSIL](#), [NS Basic](#), and [Haxe](#).

Name	Description
Dart	C/Java-like syntax with optional typing by Google.
TypeScript	Typed superset of JavaScript by Microsoft.
TeJaS	From Brown PLT. Types for JavaScript (itself).
asm.js	Subset of JavaScript that can be used as a low-level, efficient target language for compilers. Now included in Firefox.
JavaScript++	JavaScript superset with classes, type checking, among other features
Mascara	[commercial] Enhances JavaScript with powerful features like classes, namespaces and type-checking.
Roy	Tries to meld JavaScript semantics with some features common in static functional languages

Synchronous to Asynchronous JavaScript Compilers (CPS)

Name	Description
Streamline.js	Uses underscore (_) to stand for callbacks. This fork preserves line numbers for debugging.
mobl	The new language for programming the mobile web.
StratifiedJS	JavaScript + structured concurrency.
NarrativeJS	JavaScript extension with asynchronous futures and promises.
jwacs	JavaScript With Advanced Continuation Support.
Wind.js	Wind.js is an advanced library which enable us to control flow with plain JavaScript for asynchronous programming (and more) without additional pre-compiling steps.
TameJS	Adds new keywords 'await' and 'defer'
Continuation.js	A lightweight JIT compiler for simplifying asynchronous JavaScript programming with no runtime dependences. It supports both Node.js and browser-side JavaScript and is compatible with CoffeeScript (also TypeScript, and any other scripts compile to js).
Kal	Makes asynchronous programming easy and clean by allowing functions to pause and wait for I/O, replacing an awkward callback syntax with a clean, simple syntax
JSPipe	Provides JavaScript primitives to write async code without callbacks or chained functions. Inspired by Goroutines and Channels found in Go and in Clojure. For Web and NodeJS. ES5 and ES6.
promiseLand	PromiseLand is a very promising Language. It includes ES5 strict mode, uses the * dereferencing operator to access promise results and introduces additional features designed to let you focus on your

[Overview](#)[Demo](#)[Getting Started](#)[Docs](#)[Spec](#)[Community](#)[Roadmap](#)[FAQ](#)

WEBASSEMBLY

The initial version of WebAssembly has reached cross-browser consensus. [Learn more](#)

WebAssembly or *wasm* is a new portable, size- and load-time-efficient format suitable for compilation to the web.

WebAssembly is currently being designed as an open standard by a [W3C Community Group](#) that includes representatives from all major browsers.

Efficient and fast

The *wasm* [stack machine](#) is designed to be encoded in a size- and load-time-efficient [binary format](#). WebAssembly aims to execute at native speed by taking advantage of [common hardware capabilities](#) available on a wide range of platforms

Safe

WebAssembly describes a memory-safe, sandboxed [execution environment](#) that may even be implemented inside existing JavaScript virtual machines. When [embedded in the web](#), WebAssembly will enforce the same-origin and permissions security policies of the browser

GopherJS - A compiler from Go to JavaScript

* used by **244 projects** **PASSED**

GopherJS compiles Go code (golang.org) to pure JavaScript code. Its main purpose is to give you the opportunity to write front-end code in Go which will still run in all browsers.

Playground

Give GopherJS a try on the [GopherJS Playground](#).

What is supported?

Nearly everything, including Goroutines ([compatibility table](#)). Performance is quite good in most cases, see [HTML5 game engine benchmark](#). Cgo is not supported. Using a vendored copy of GopherJS is currently not supported, see [#415](#).

Installation and Usage

Get or update GopherJS and dependencies with:

```
go get -u github.com/gopherjs/gopherjs
```

Now you can use `gopherjs build [package]`, `gopherjs build [files]` or `gopherjs install [package]` which behave similar to the `go` tool. For `main` packages, these commands create a `.js` file and `.js.map` source map in the current directory or in `$GOPATH/bin`. The generated JavaScript file can be used as usual in a website. Use `gopherjs help [command]` to get a list of possible command line flags, e.g. for minification and automatically watching for changes.

Note: GopherJS will try to write compiled object files of the core packages to your `$GOROOT/pkg` directory. If that fails, it will fall back to `$GOPATH/pkg`.

gopherjs run, gopherjs test



ClojureScript is a **robust, practical, and fast** programming language with a set of useful features that together form a **simple, coherent, and powerful tool**.

[Get Started!](#)

ClojureScript is a compiler for [Clojure](#) that targets JavaScript. It emits JavaScript code which is compatible with the advanced compilation mode of the Google Closure optimizing compiler.

Why Clojure?

Clojure is a dynamic, general-purpose programming language supporting interactive development. Clojure is a functional programming language featuring a rich set of immutable, persistent data structures. As a dialect of Lisp, it has a code-as-data philosophy and a powerful macro system.

Why JavaScript?

The spread of the browser and web-based applications has given JavaScript an exclusive reach unmatched by any other language. Because of its importance, JavaScript engines are also receiving extensive research and optimization allowing JavaScript to compete for performance with many more established platforms like the JVM.

Why Google Closure?

Learn More

Release

Current version: 1.9.946

Rationale

A brief overview of ClojureScript

Reference

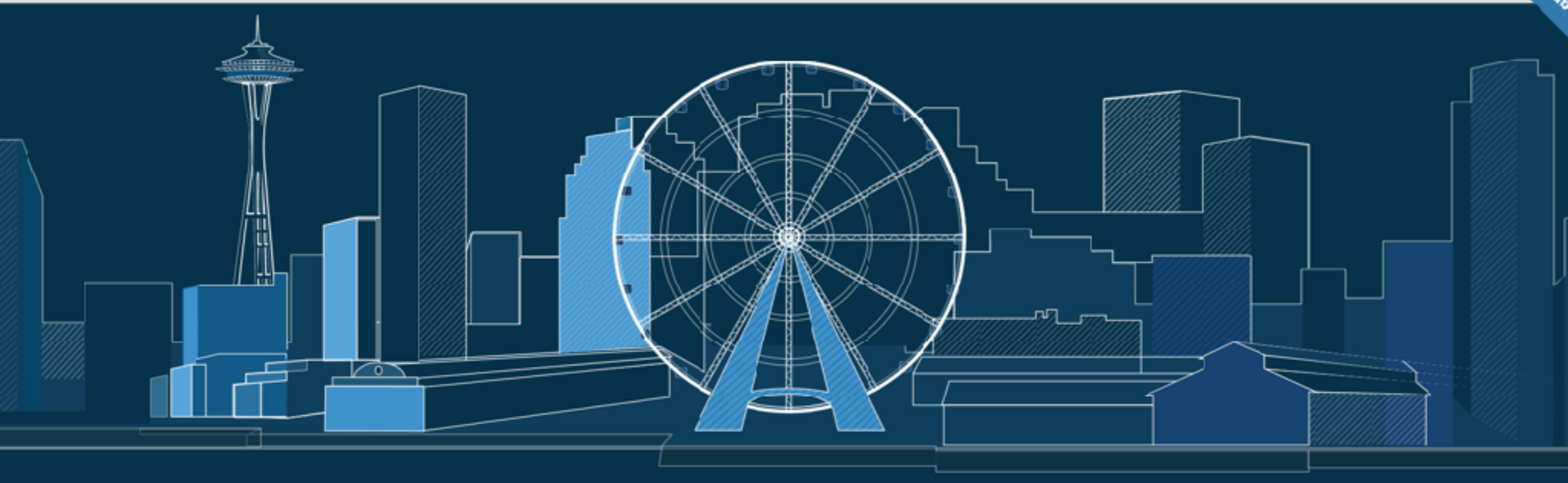
Detailed reference material

Guides

Walkthroughs to help you learn along the way

Community

TypeScript 2.6 is now available. [Download](#) our latest version today!



TypeScript

JavaScript that scales.

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.

Any browser. Any host. Any OS. Open source.

[Download](#)[Documentation](#)

Babel is a JavaScript compiler.

Use next generation JavaScript, today.

Put in next-gen JavaScript

```
[1, 2, 3].map(n => n ** 2);
```

Get browser-compatible JavaScript out

```
[1, 2, 3].map(function (n) {  
  return Math.pow(n, 2);  
});
```

[Check out our REPL to experiment more with Babel!](#)

[Latest From Our Blog: Babel Turns Three](#)

Introducing JSX

Consider this variable declaration:

```
const element = <h1>Hello, world!</h1>;
```

This funny tag syntax is neither a string nor HTML.

It is called JSX, and it is a syntax extension to JavaScript. We recommend using it with React to describe what the UI should look like. JSX may remind you of a template language, but it comes with the full power of JavaScript.

JSX produces React “elements”. We will explore rendering them to the DOM in the [next section](#). Below, you can find the basics of JSX necessary to get you started.

Embedding Expressions in JSX

You can embed any JavaScript expression in JSX by wrapping it in curly

QUICK START ^

Installation

Hello World

Introducing JSX

Rendering Elements

Components and Props

State and Lifecycle

Handling Events

Conditional Rendering

Lists and Keys

Forms

Lifting State Up

Composition vs Inheritance

Thinking In React

ADVANCED GUIDES v

REFERENCE v

CONTRIBUTING v

FAQ v

Rhino

[Languages](#)[Edit](#)

Rhino is an open-source implementation of [JavaScript](#) written entirely in Java. It is typically embedded into Java applications to provide scripting to end users. It is embedded in J2SE 6 as the default Java scripting engine.

Rhino downloads

How to [get source and binaries](#).

Rhino documentation

[Information on Rhino](#) for script writers and embedders.

Rhino help

[HOME](#)[ABOUT](#)[DOWNLOADS](#)[DOCS](#)[GET INVOLVED](#)[SECURITY](#)[NEWS](#)[FOUNDATION](#)

About

[Governance](#)[Community](#)[Working Groups](#)[Releases](#)[Resources](#)[Trademark](#)[Privacy Policy](#)

About Node.js®

As an asynchronous event driven JavaScript runtime, Node is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection the callback is fired, but if there is no work to be done, Node will sleep.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

This is in contrast to today's more common concurrency model where OS threads are



Thanks for listening!

Carl Johnson

Newsroom Product Engineer

Baltimore Sun

blog.carlmjohnson.net

@carlmjohnson