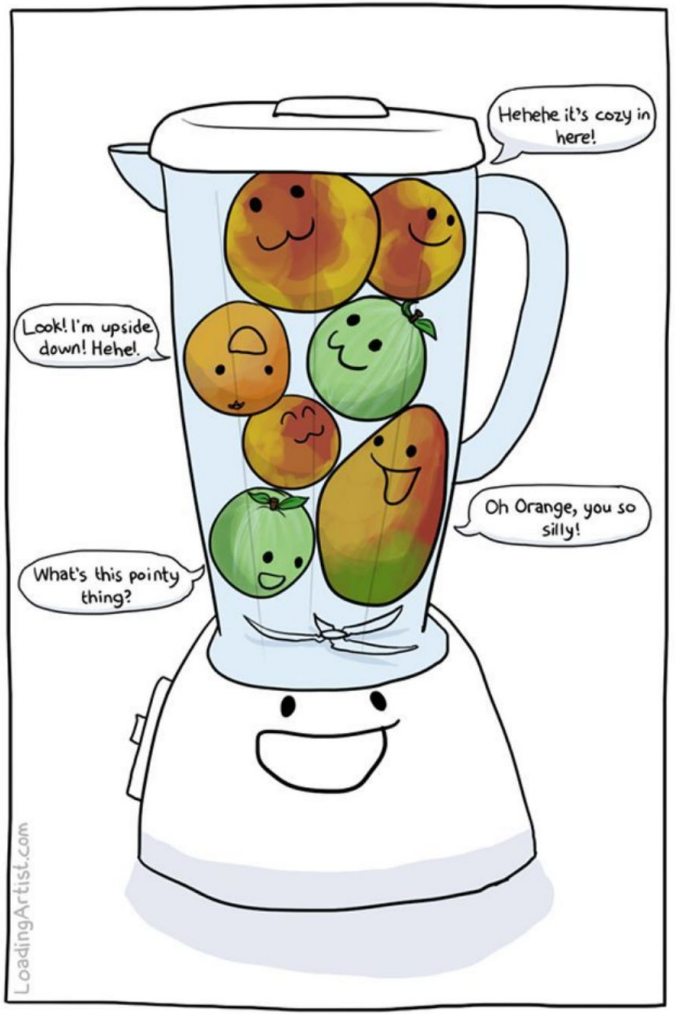
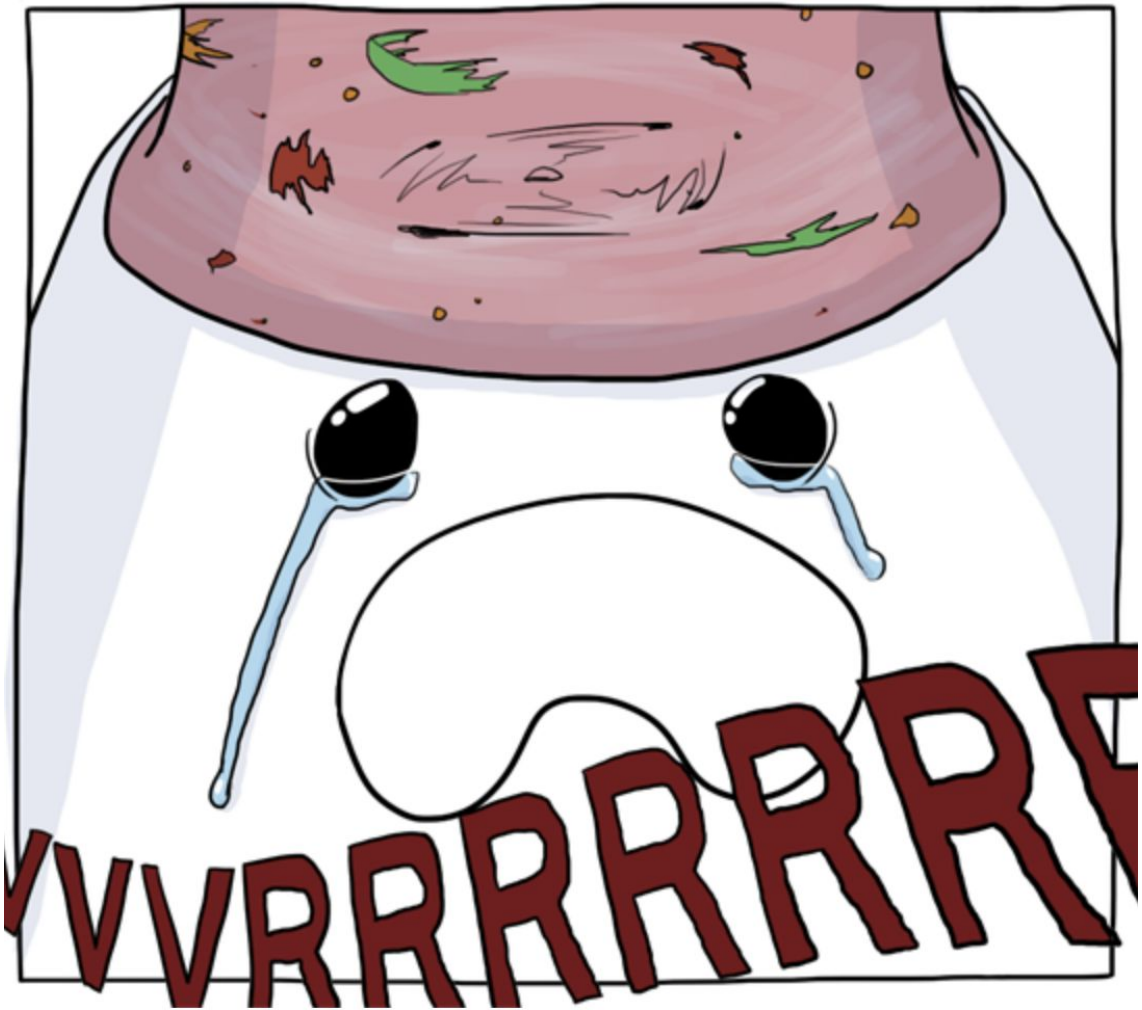
TC39
Language Design in the Open

Yulia Startsev • Berlin.js 2019

```
class ChildNetworkResponseLoader {
  constructor(context, requestId) {
    this.context = context;
    this.requestId = requestId;
  }

  api() {
    const {context, requestId} = this;
    return {
      getContent(callback) {
        return context.childManager.callParentAsyncFunction(
          "devtools.network.Request.getContent",
          [requestId],
          callback);
      },
    };
  }
}
```


Hehehe it's cozy in here!
Look! I'm upside down! Hehe!
Oh Orange, you so silly!
What's this pointy thing?
LoadingArtist.com


VVVRRRRRRR

Hi. My name is Yulia.
I work at Mozilla.

I am also a co-chair on TC39

Overview

- A bit of history
- Issues from ES4
- The Proposal Process
- How is JavaScript designed today?
- Current Work
- Getting involved

What *was* JavaScript?

The Requirements

- A scripting language for web APIs
- Intended to be easy for both professionals and amateurs
- Developed by Brendan Eich in 10 days
- Originally planned as a Scheme-like language[*]
- It should look like Java

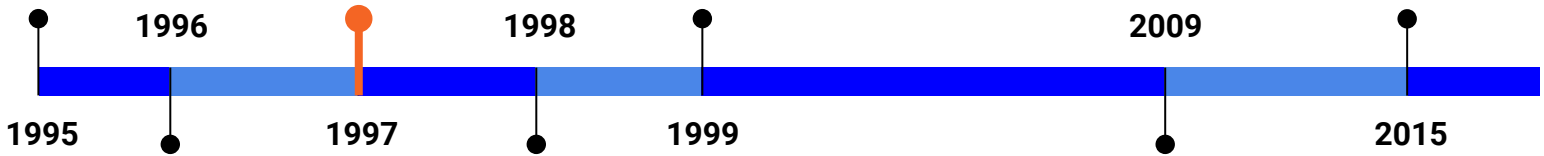
“It was deceptive in intent and effect”

We aimed to provide a “glue language” for the Web designers and part time programmers who were building Web content from components such as images, plugins, and Java applets. We saw Java as the “component language” used by higher-priced programmers, where the glue programmers—the Web page designers—would assemble components and automate their interactions using [a scripting language]. *

The Timeline

A prototype named Mocha

Written in 10 days by Brendan Eich, this is the first “JavaScript” implementation from netscape



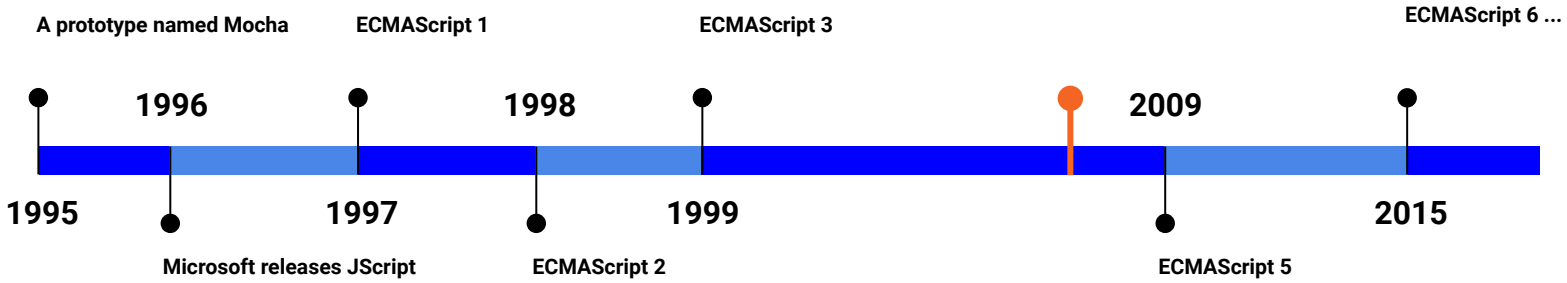
JScript was a reverse engineered version of JavaScript. To keep MicroSoft in check, Netscape moved to standardize JavaScript

TC39 and its structure

- Part of Ecma International
- Technical Committee 39 of Ecma International
- Takes care of several standards aside from JavaScript, including ECMA-402, ECMA-404, ECMA-414

- Operates via “consensus”

The Timeline



Issues with ES4[*]

- ES4 was a proposed standard that was debated for 10 years
- Proposals were added to the language without implementations
- It became too large, vendors began to wonder how much of it should be implemented
- ES3.1 was introduced as an incremental change that was a subset of ES4
- These two specifications eventually came into conflict

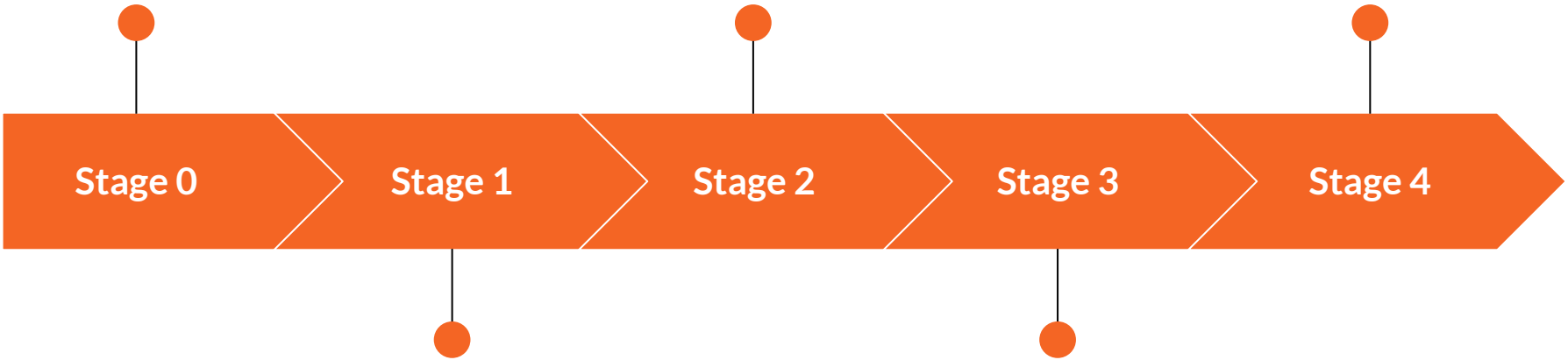
Moving forward with “Harmony”

- The project codenamed “Harmony” would lead to first to ES3.1, then ES5 and ES6[*]
- Addresses the issue of having working implementations for proposals, called the Harmony Process
- Introduces a structure of champions, using ESDiscuss, and advancing proposals in stages

Someone has an idea and they write it up

Committee discusses if this feature “should be in the language”

Proposal is included in the specification



The idea is presented to the committee, committee makes comments

Polyfill and browser implementations, final form of the proposal takes shape

Stage 0

- Allow input into the specification

Example: Pattern matching

Matching `fetch()` responses:

```
const res = await fetch(jsonService)
case (res) {
  when {status: 200, headers: {'Content-Length': s}} -> {
    console.log(`size is ${s}`)
  }
  when {status: 404} -> {
    console.log('JSON not found')
  }
  when {status} if (status >= 400) -> {
    throw new RequestError(res)
  }
}
```

Stage 1

- Make the case for the addition
- Describe the shape of a solution
- Identify potential challenges

Requirements

- Identified “champion” who will advance the addition
- Prose outlining the problem or need and the general shape of a solution
- Illustrative examples of usage
- High-level API
- Discussion of key algorithms, abstractions and semantics
- Identification of potential “cross-cutting” concerns and implementation challenges/complexity

Stage 2

Precisely describe the syntax and semantics using formal spec language

The committee expects the feature to be developed and eventually included in the standard

Requirements

- Initial spec text
- all *major* semantics, syntax and API are covered, but TODOs, placeholders and editorial issues are expected

Stage 3

Indicate that further refinement will require feedback from implementations and users

The solution is complete and no further work is possible without implementation experience, significant usage and external feedback.

Requirements

- Complete spec text
- Designated reviewers have signed off on the current spec text
- All ECMAScript editors have signed off on the current spec text
- All semantics, syntax and API are completed described

Stage 4

Indicate that the addition is ready for inclusion in the formal ECMAScript standard

Requirements

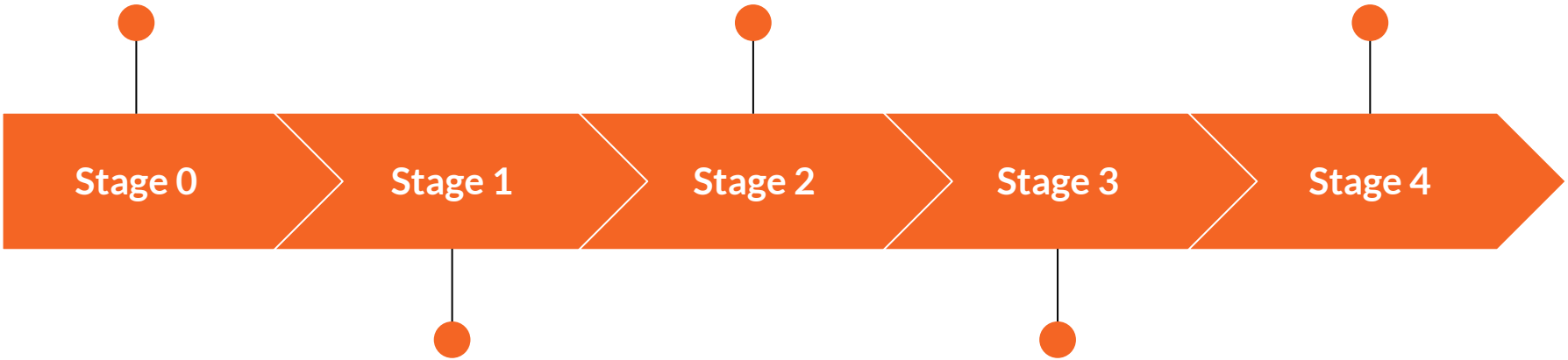
- Test262 acceptance tests have been written for mainline usage scenarios, and merged
- Two compatible implementations which pass the acceptance tests
- Significant in-the-field experience with shipping implementations, such as that provided by two independent VMs
- A pull request has been sent to tc39/ecma262 with the integrated spec text
- All ECMAScript editors have signed off on the pull request

How is JavaScript Designed Today?

Someone has an idea and they write it up

Committee discusses if this feature “should be in the language”

Proposal is included in the specification



The idea is presented to the committee, committee makes comments

Polyfill and browser implementations, final form of the proposal takes shape

TC39 Meetings

- 6 meetings a year
- 89 delegates from members
- Usual attendance of ~60
- Proposals are presented followed by debate
- Session ends with a decision
- Everything is recorded and published at https://github.com/tc39/tc39-notes

Ecma TC39

Ecma International, Technical Committee 39 - ECMAScript

The web https://www.ecma-international.org/me...

Repositories 112 People 129 Teams 11 Projects 0

Pinned repositories

ecma262

Status, process, and documents for ECMA262

HTML ★ 6.8k 492

proposals

Tracking ECMAScript Proposals

★ 5.1k 183

test262

Official ECMAScript Conformance Test Suite

JavaScript ★ 822 217

agendas

TC39 meeting agendas

JavaScript ★ 359 108

tc39-notes

Forked from rwaldron/tc39-notes

TC39 Meeting Notes

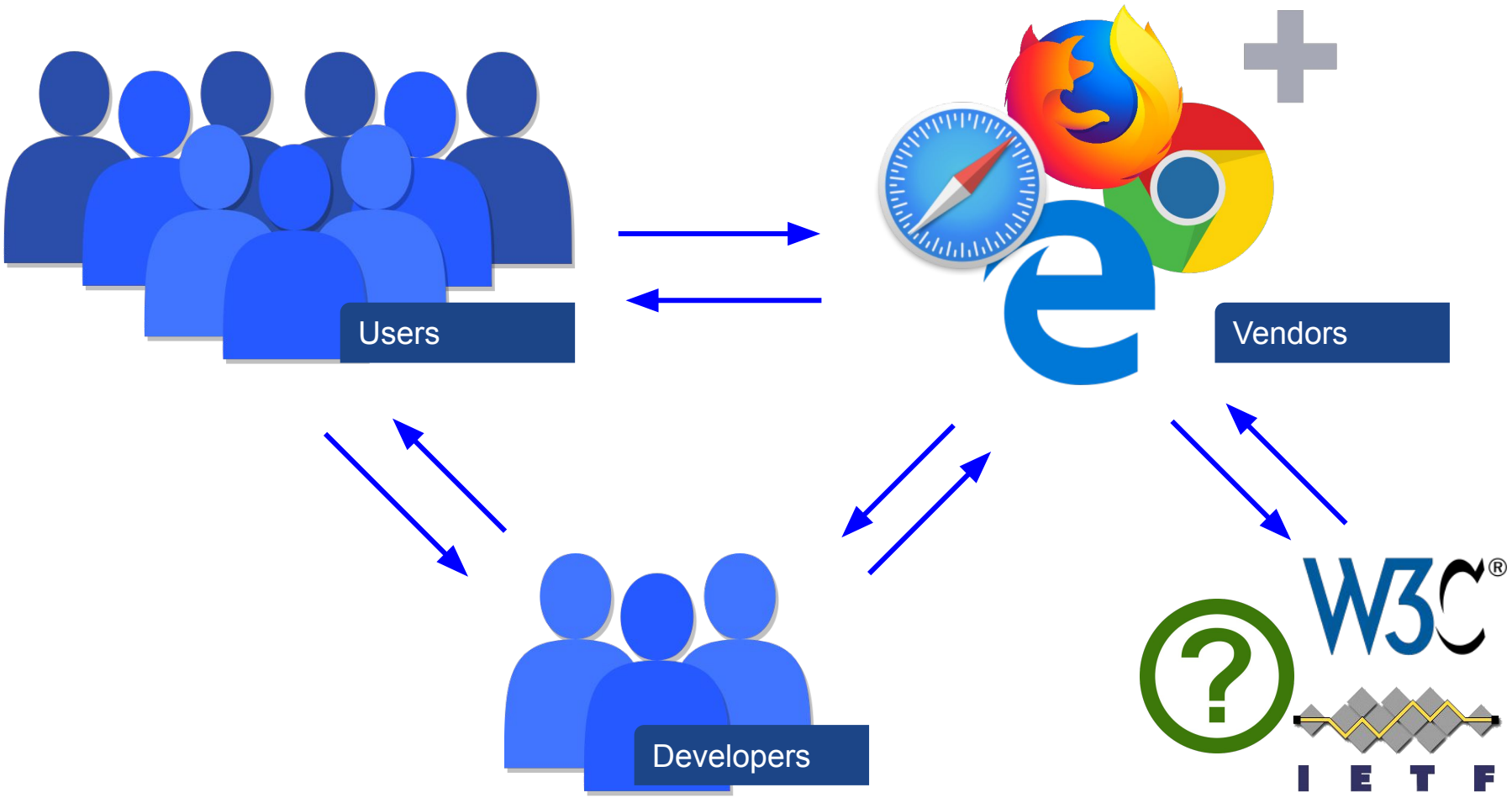
JavaScript ★ 315 30

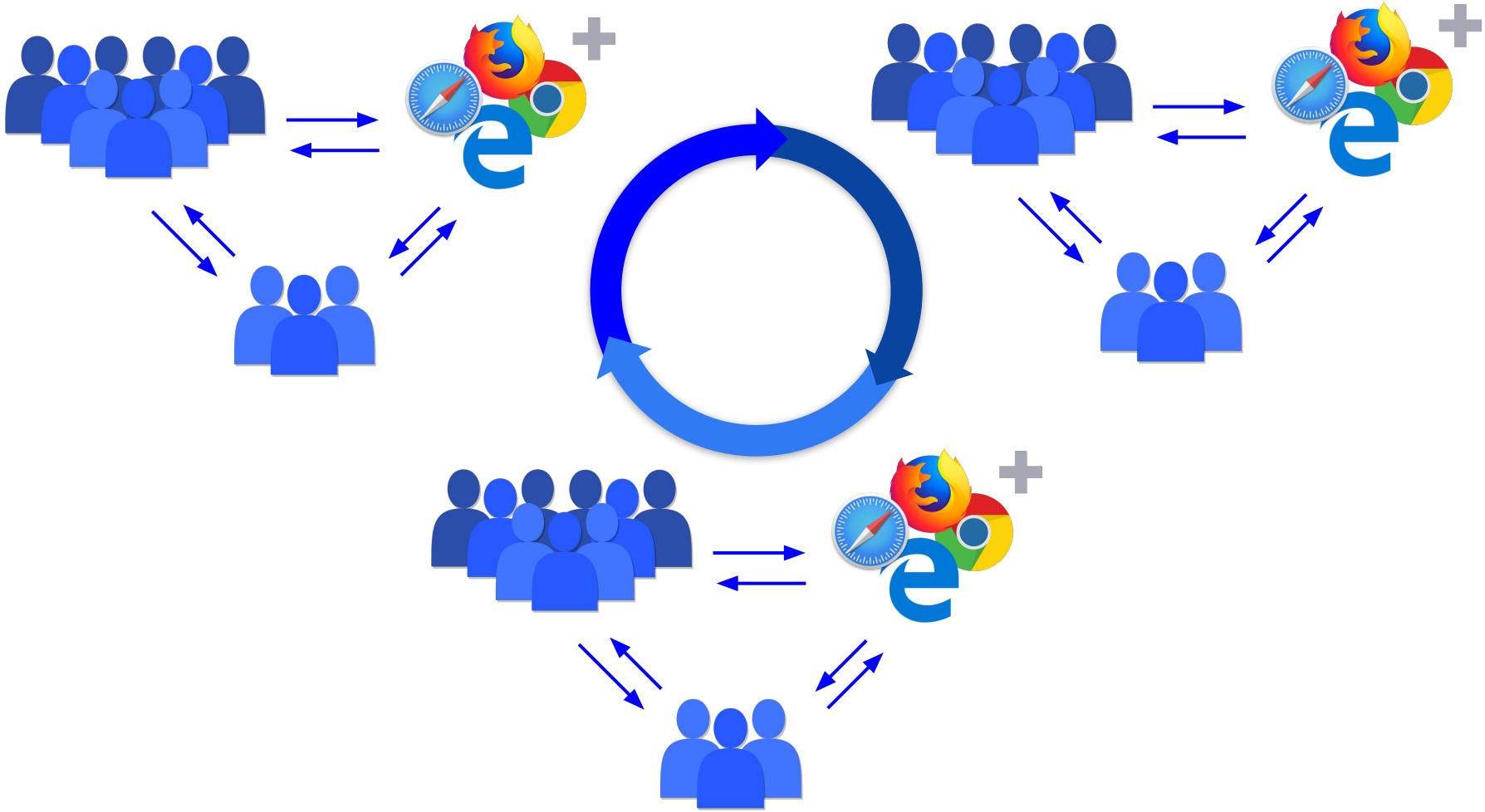
ecma402

Status, process, and documents for ECMA 402

HTML ★ 155 37

What is at stake?


Users
Vendors
Developers
W3C®
IETF

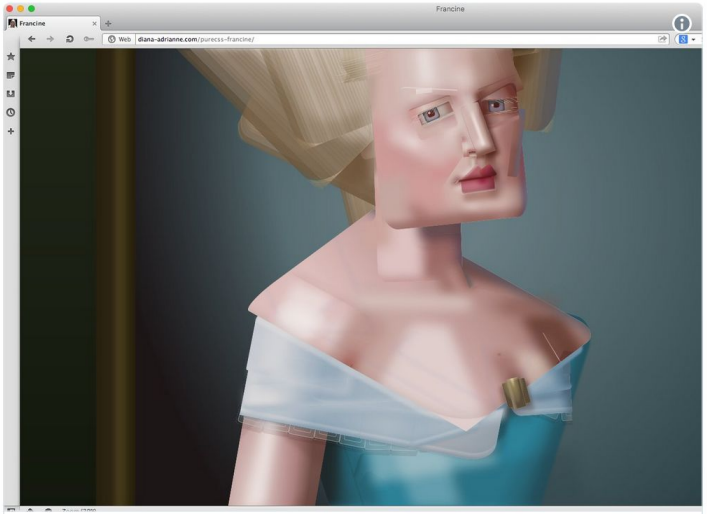


The Extensible Web Manifesto[*]

- The standards process should focus on adding ***new low-level capabilities*** to the web platform that are secure and efficient.
- The web platform should expose low-level capabilities that ****explain existing features***, such as HTML and CSS, allowing authors to understand and replicate them.
- The web platform should develop, describe and test new high-level features in JavaScript, and allow web developers to iterate on them before they become standardized. This creates a ***virtuous cycle*** between standards and developers.
- The standards process should ***prioritize efforts*** that follow these recommendations and deprioritize and refocus those which do not.

Backwards Compatibility

- Many people rely on the web for their livelihood
- Many people rely on services provided by the web
- Not all users have the capability to upgrade their operating system or their browsers.



David Zhou ✓
@dz

i tried @cyanharlow's incredible pure css portrait in an old version of opera and well, the disclaimer wasn't lying: "so the live preview will most likely look laughable in anything other than chrome"github.com/cyanharlow/pur…

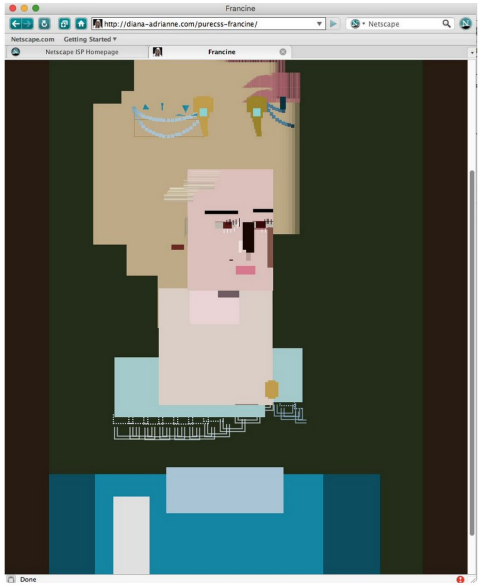
7:17 PM - May 1, 2018

♡ 1,269 💬 424 people are talking about this

Mayowa Tomori
@mdotslash

And Netscape Navigator for the true romantics amongst you.
pic.twitter.com/hO12KvVoJg

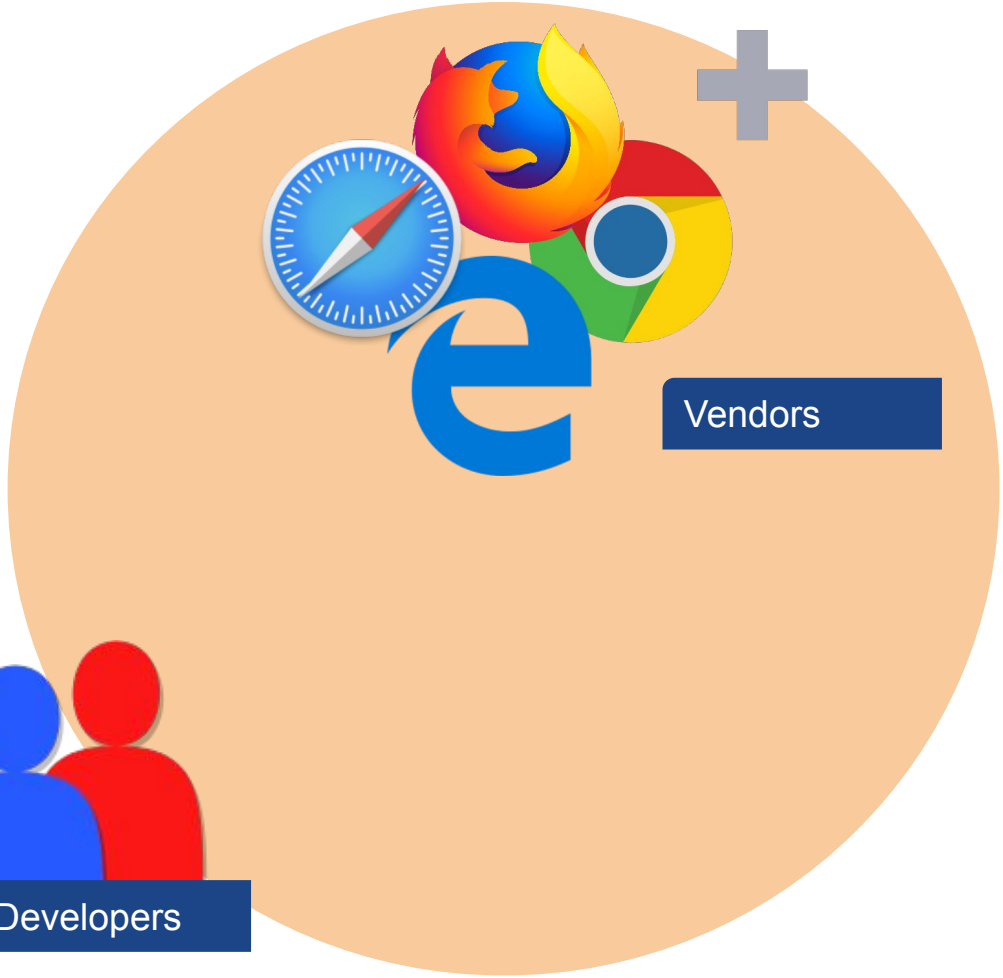
4:50 AM - May 2, 2018



♡ 238 💬 54 people are talking about this

[Pure CSS Francine](#) by Diana Smith


Users


Vendors

Developers

Why not break the web?

- The costs are impossible to estimate
- It is easy to rename something.

```
Array.prototype.contains -> Array.prototype.includes
```

- Breaking the Web affects users disproportionately compared to developers. Making the web worse for users pushes them away from the web.

Current work

- WASM and JavaScript
- Class improvements
- Realms
- Lots more!

README.md

WeakReferences TC39 proposal

Introduction

The WeakRef proposal encompasses two major new pieces of functionality:

1. creating *weak references* to objects with the `WeakRef` class
2. running user-defined *finalizers* after objects are garbage-collected, with the `FinalizationGroup` class

These interfaces can be used independently or together, depending on the use case.

A note of caution

Garbage collectors are complicated. If an application or library depends on GC cleaning up a WeakRef or calling a finalizer in a timely, predictable manner, it's likely to be disappointed: the cleanup may happen much later than expected, or not at all. Sources of variability include:

- One object might be garbage-collected much sooner than another object, even if they become unreachable at the same time, e.g., due to generational collection.
- Garbage collection work can be split up over time using incremental and concurrent techniques.
- Various runtime heuristics can be used to balance memory usage, responsiveness.
- The JavaScript engine may hold references to things which look like they are unreachable (e.g., in closures, or inline caches).
- Different JavaScript engines may do these things differently, or the same engine may change its algorithms across versions.

For this reason, the W3C TAG Design Principles recommend against creating APIs that expose garbage collection. It's

README.md

JavaScript Decorators

Stage 2

Status

Decorators are a JavaScript language feature, proposed for standardization at TC39. Decorators are currently at Stage 2 in TC39's process, indicating that the committee expects them to eventually be included in the standard JavaScript programming language. The decorators champion group is considering a redesign of the proposal as "static decorators", which the rest of this document describes.

The idea of this proposal

This decorators proposal aims to improve on past proposals by working towards twin goals:

- It should be easy not just to use decorators, but also to write your own.
- Decorators should be fast, both generating good code in transpilers, and executing fast in native JS implementations.

This proposal enables the basic functionality of the JavaScript original decorators proposal (e.g., most of what is available in TypeScript decorators), as well as two additional capabilities of the previous Stage 2 proposal which were especially important: access to private fields and methods, and registering callbacks which are called during the constructor.

Core elements:

- There's a set of **built-in decorators** that serve as the basic building blocks.
  - `@wrap`: Replace a method or the entire class with the return value of a given function

README.md

ECMAScript Pattern Matching

Status

Stage: 1

Author: Kat Marchán (npm, @maybekatz)

Champions: Brian Terlson (Microsoft, @bterlson), Sebastian Markbåge (Facebook, @sebmarkbage), Kat Marchán (npm, @maybekatz)

Introduction

This proposal adds a pattern matching expression to the language, based on the existing Destructuring Binding Patterns.

There's many proposals potentially related to this one, and other proposals might mention interaction with this. This file includes casual, example-based discussion of the proposal, and there's also a document describing the core semantics in more formal language, which will be iterated over into the final Spec-ese.

There's also a document including suggestions for other future proposals, which are dependent on this one, but do not directly affect the main behavior of the feature.

This proposal was approved for Stage 1 in the May 2018 TC39 meeting, and slides for that presentation are available.

This proposal draws heavily from corresponding features in Rust, F#, Scala, and Elixir/Erlang.

Motivating Examples

Getting Involved!

- Community groups
- The website
- Discourse
- Participating on github
- What to watch for news

Community Groups

- Community groups have been organized around tooling, frameworks and teaching.
- Groups meet once a month on average
- We are planning an “open house” community call after each plenary
- If you want to get involved, talk to me

The website

- The website lives at http://tc39.github.io
- You can participate in the work on the website at https://github.com/tc39/tc39.github.io
- You can also check the website for change as the spec evolves.

Participating on Github

- You will need to sign the IPR form
- You can find all of the proposals under https://github.com/tc39/proposals
- You can also work on tests at https://github.com/tc39/test-262

News? What to watch

- The website
- ES discourse
- Proposals directory
- Notes summaries
- Follow committee members, we often post news

Upcoming Events in Berlin

- JSConf EU Panel - June 1-2
  - Ask us Questions!
  - Use the twitter tag #tc39panel
  - We will answer on stage and the session will be recorded
- Meet the TC39 - June 6th
  - At fullstack.js meetup
  - https://t.co/dPrqyrJSzh

Seriously. ask us questions #tc39panel

?

Thank you.

Please get in touch

@ioctaptceb

yulia@mozilla.com