TC39: взгляд изнутри

Сергей Рубанов

- BeerJS Moscow
- WebAssembly Moscow meetup
- t.me/juliarderity
- github.com/chicoxyzzy

JavaScript Kamikaze в

О чем доклад

- чем занимается TC39
- состав, структура
- процесс
- как принять участие

Почему я думаю, что это важно

- понимать как и почему принимаются те или иные решения, что они принимаются не просто так
- знать как и кому донести обратную связь
- участие в опенсорсе
- самообразование и обучение других


Harmony

ECMAScript 3.1

vs

ECMAScript 4


спорами ничего не решишь

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

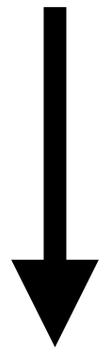
- Previous message: Old documents marked "New"
- Next message: ECMAScript Harmony
- **Messages sorted by:** [date] [thread] [subject] [author]

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".

https://mail.mozilla.org/pipermail/es-discuss/2008-August/003400.html

ECMAScript 3.1



ECMAScript 5

ECMAScript 6
aka
Harmony

Проблемы процесса

- долгая доставка новых версий (годы разработки спецификаций и имплементаций перед релизом)
- относительная закрытость (js коммьюнити сложно повлиять на процесс и дать обратную связь)

ES2016+

Technical committee 39

Кто из нижеперечисленных является членом TC39?

A: Mathias Bynens

B: Dr. Axel Rauschmayer

C: Вадим Макеев

D: Никто из перечисленных

Никто!

Члены Ecma International

Ordinary (Full)

Google
KONICA MINOLTA
HITACHI Inspire the Next
Microsoft
hp
PayPal
IBM
stripe
intel

Associate

airbnb
GoDaddy
NETFLIX
Bloomberg
360 www.360.cn
Canon
RICOH
salesforce
FUJITSU

SME Members

Evernote Corporation
METEOR
HEAD acoustics
npm
igalia
SHAPE

SPC Members

AGORIC
QUADRAC
bocoup
TILDE
moddable

Члены Ecma International

Not For Profit Members

Archive Disc Test Center

IT R&D Global Leader ETRI

SFW

General Incorporated Association Small Fan Workshop

Indian Institute of Technology Delhi

JBMIA

Kahu Research

OpenJS Foundation

TURING

BAM Bundesanstalt für Materialforschung und -prüfung

INGENIEURS DU MONDE

慶應義塾 Keio University

公立大学法人 大阪府立大学 Osaka Prefecture University

UC Santa Cruz

法政大学 HOSEI University

Inria

大阪産業大学 Osaka Sangyo University

Ben-Gurion University of the Negev

humanitarian TOOLBOX

KOREA UNIVERSITY

USF University of South Florida

ICT

PURDUE UNIVERSITY

Dr. G.R. Damodaran College of Science

imec

BERKELEY LAB

UTS University of Technology Sydney

IT University of Copenhagen

McGill

The LIBRARY of CONGRESS

University of Victoria

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Imperial College London

mozilla

UEC TOKYO

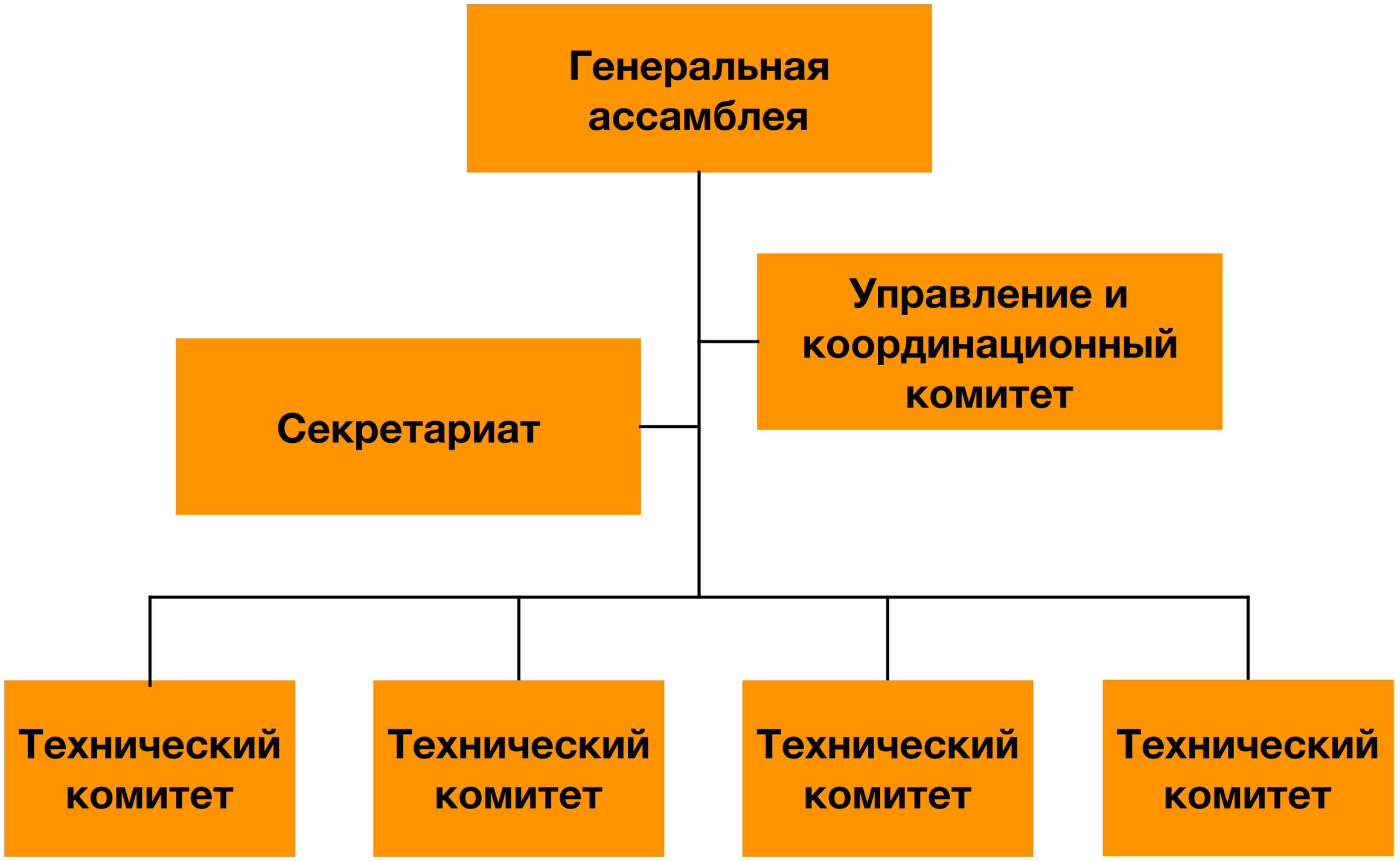
ETH

INDIANA UNIVERSITY

NHK Japan Broadcasting Corporation

OITDA

VCI


Генеральная ассамблея
Управление и координационный комитет
Секретариат
Технический комитет
Технический комитет
Технический комитет
Технический комитет

Члены Ecma International

Права и обязанности	Полноправные члены	Другие группы (включая некоммерческие организации)	Мелкие частные компании
Право голоса в ГА	✅		
Участие в ГА	✅	✅	
Участие в управлении	✅		
Участие в технических комитетах	✅	✅	1
Возможность номинировать председателя ТК	✅		
Право голоса в ТК	✅	✅	1

Технические комитеты в составе Ecma

- TC12 (Product Safety)
- TC20 (EMC and EMF)
- TC26 (Acoustic)
- TC31(Information Storage)
- TC32 (Multimedia Coding and Communications)
- TC38 (Product-related Environmental Attributes)
- **TC39 (ECMAScript)**
- TC45 (Office Open XML Formats)
- TC46 (Open XML Paper Specification (OpenXPS))
- TC49 (Programming Languages)
- TC50 (Close Proximity Electric Induction Data Transfer)
- TC51 (Access Systems)
- TC52 (Dart)
- TC53 (Smart wearable systems and sensor-based devices)

Стандарты, над которыми работает TC39

- ECMA-262 - ECMAScript® Language Specification
- ECMA-402 - ECMAScript® Internationalization API Specification
- ECMA-404 (aka ISO/IEC 21778) - The JSON Data Interchange Syntax
- ECMA-TR/104 - ECMAScript® Test Suit
- ECMA-414 - ECMAScript® Specification Suite

Роли

- делегаты компаний членов
- приглашенные эксперты
- наблюдатели

Председатели комитета

Aki Braun
(PayPal)

Brian Terlson
(Microsoft)

Yulia Startsev
(Mozilla)

Группа редакторов ES2019

Brian Terlson (Microsoft)

Jordan Harband (ex-Airbnb)

Bradley Farias (GoDaddy)

Группа редакторов ES2020

Jordan Harband
(ex-Airbnb)

Kevin Smith
(Microsoft)

Планирование и ревизии спецификаций

- февраль: черновики спецификаций
- март: в спецификации влиты все завершенные фичи
- апрель-июнь: ревью спецификаций Генеральной Ассамблеей
- июль: выпуск новой версии спецификации

🎉 **ES2019 уже выпущен!** 🎉

Процесс стандартизации предложений

Основные этапы

Stage 0 (Strawperson)

- идеи в любой форме
- без каких-либо приемочных требований
- авторы могут не являться делегатами компаний, входящих в Ecma

Stage 1 (Proposal)

- определен чемпион
- описана проблема или необходимость предложения и общая форма решения
- описаны примеры использования
- описано высокоуровневое API
- обсуждения ключевых алгоритмов, абстракций и семантики
- определены потенциальные возможности и сложности имплементации

Stage 2 (Draft)

- есть начальный текст спецификации

Stage 3 (Candidate)

- есть полный текст спецификации
- рецензенты одобрили текст спецификации
- все редакторы одобрили текст спецификации

Stage 4 (Finished)

- написаны приёмочные тесты
- есть как минимум две имплементации, которые проходят тесты
- есть PR для включения в стандарт
- все редакторы одобрили PR для включения в стандарт

Как проходят встречи

- перед встречей составляется повестка встречи
    - предложения с более высокими stage имеют более высокий приоритет и обсуждаются первыми
    - дедлайн для добавления предложений, собирающихся продвигаться на следующую ступень — 10 дней до начала встречи TC39
- встречи проходят раз в два месяца на протяжении трех дней
- в начале встречи проходит доклад от секретариата Ecma
- для продвижения фичи на следующий этап необходим **общий консенсус**
- в последний день встречи проходит newcomers event
- председатели комитета следят за соблюдением правил с помощью TCQ

AGENDA ITEM

Class Access Expressions for Stage 1

Ron Buckton (Microsoft) 30 minutes

TOPIC

I like this

Waldemar Horwat

SPEAKING

Waldemar Horwat

I like this

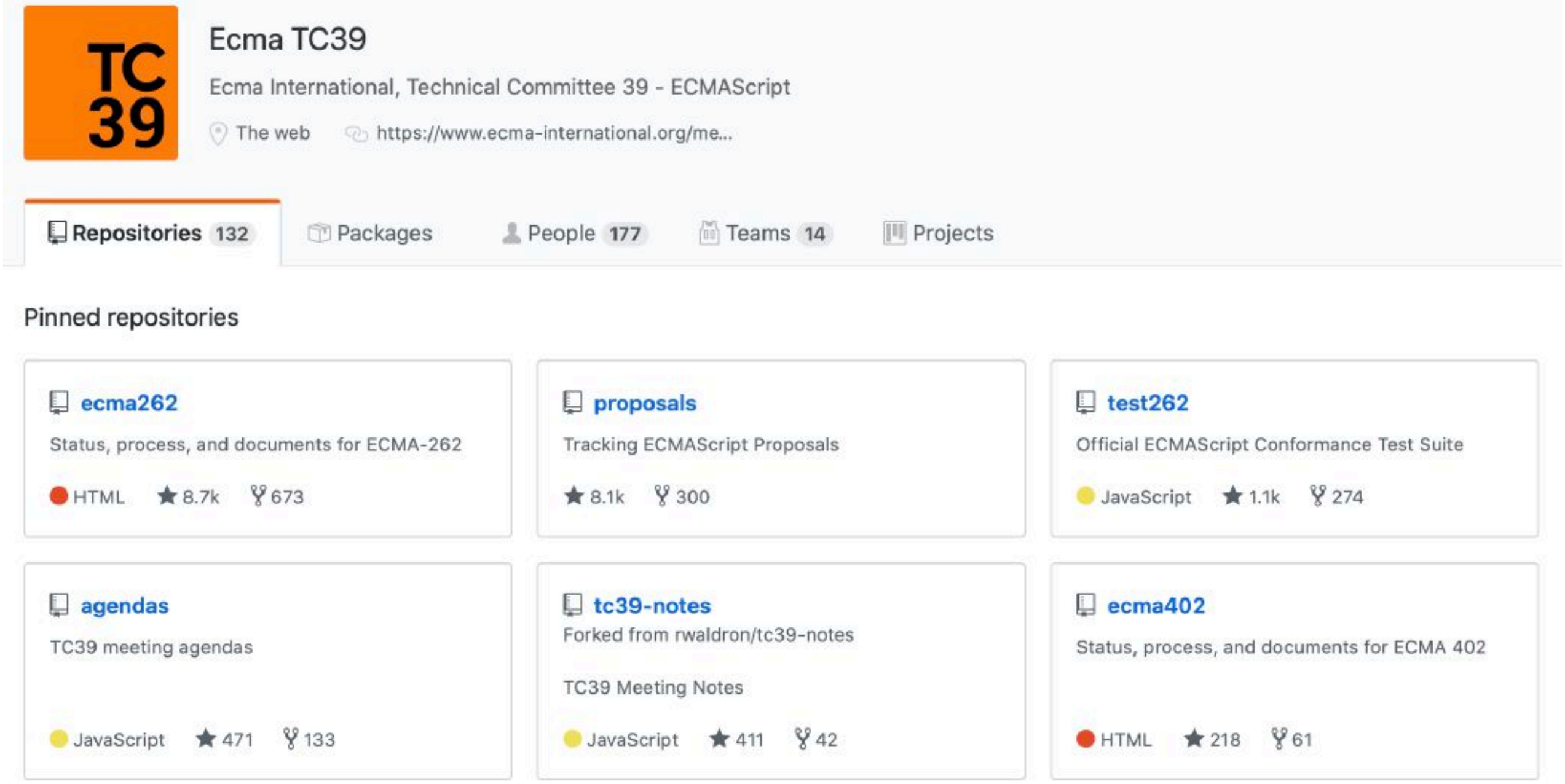
New Topic | Discuss Current Topic | Clarifying Question | Point of Order

SPEAKER QUEUE

1. Reply: I think you do have to learn this vs. class if you want this distinction
   Yehuda Katz (Tilde, Inc.)
2. New Topic: Reply to Jordan's reasons for not using class names
   @efaust
3. New Topic: Concur with YK. This doesn't solve the footgun if `this` isn't base class.
   Justin Ridgewell (Google)
4. New Topic: class.method.call(SubClass) is much more clear.
   Justin Ridgewell (Google)

Как поучаствовать

Всё самое главное — на GitHub



Обратная связь

- github
- discourse
- IRC: #tc39 на Freenode

Собственное предложение

- подумать имеет ли смысл
- поискать существующие (в списке предложений, в записях со встреч, на esdiscuss)
- оформить
- найти чемпиона, который будет
  - представлять предложение комитету
  - работать с имплементорами
  - собирать фидбек

PR в спецификацию

- normative (в т.ч. добавление stage 4 фич)
- editorial
- layering
- markup
- meta

Примеры моих PR

Editorial: remove `await` from FutureReservedWords in A.1. #926

Edit

Merged **bterlson** merged 1 commit into `tc39:master` from `chicoxyzzy:remove_await_from_futurereservedwords_a1` on Jun 5, 2017

Conversation 0 · Commits 1 · Checks 0 · Files changed 1 · +0 −1

Changes from **all commits** · File filter... · Jump to... · 0 / 1 files viewed · Review changes

1 spec.html · Viewed

```
...   ...
37852 37852       <emu-prodref name=ReservedWord></emu-prodref>
37853 37853       <emu-prodref name=Keyword></emu-prodref>
37854 37854       <emu-prodref name=FutureReservedWord></emu-prodref>
37855       -     <p>`await` is only treated as a |FutureReservedWord| when |Module| is the goal symbol of the syntactic grammar.<
37856 37855       <p>The following tokens are also considered to be |FutureReservedWord|s when parsing strict mode code:</p>
37857 37856       <p><emu-t>implements</emu-t>  <emu-t>package</emu-t>  <emu-t>protected</emu-t><br>
37858 37857         <emu-t>interface</emu-t>  <emu-t>private</emu-t>  <emu-t>public</emu-t>
```

2 Open ✓ 6 Closed Author Labels Projects Milestones

Editorial: Add ! and ? before CreateBuiltinFunction and CreateArrayFromList

editorial change

#1607 opened 11 days ago by chicoxyzzy • Approved

Normative: Redefine CatchParameter as FormalParameter needs consensus needs tests

normative change

#1126 opened on Mar 3, 2018 by chicoxyzzy • Review required

chicoxyzzy commented on Feb 26, 2018 • edited ▾ Member

There is some inconsistency between `CatchParameter` and `BindingElement` (i.e. function parameters).

Example from babel/babel#7434:

```
try {
  throw undefined;
} catch ({foo = 'bar'} = {}) {
  console.log(foo);
}
```

This is Syntax Error according to current spec grammar, but are there any reasons why assigning default value is not allowed here while destructuring (and default parameters inside of it) work fine?

👍 7

Normative: Redefine CatchParameter as FormalParameter #1126

Edit

Open **chicoxyzzy** wants to merge 1 commit into `tc39:master` from `chicoxyzzy:catchparameter_as_formalparameter`

Conversation 19 | Commits 1 | Checks 0 | Files changed 1 | +1 −2

Changes from **all commits** | File filter... | Jump to... | 0 / 1 files viewed | Review changes

3 spec.html — Viewed

```
@@ -18287,8 +18287,7 @@ <h2>Syntax</h2>
18287 18287          `finally` Block[?Yield, ?Await, ?Return]
18288 18288
18289 18289        CatchParameter[Yield, Await] :
18290       -          BindingIdentifier[?Yield, ?Await]
18291       -          BindingPattern[?Yield, ?Await]
      18290 +          FormalParameter[?Yield, ?Await]
18292 18291      </emu-grammar>
18293 18292      <emu-note>
18294 18293        <p>The `try` statement encloses a block of code in which an exceptional condition can occur, such as a runtime
```

ECMAScript proposal: `Promise.any`

Author: Mathias Bynens, Kevin Gibbons, Sergey Rubanov

Champion: Mathias Bynens

Stage: Stage 1 of the TC39 process.

Motivation

There are four main combinators in the `Promise` landscape.

name	description	
`Promise.allSettled`	does not short-circuit	separate proposal ➡ SOON
`Promise.all`	short-circuits when an input value is rejected	added in ES2015 ✅
`Promise.race`	short-circuits when an input value is settled	added in ES2015 ✅
`Promise.any`	short-circuits when an input value is fulfilled	this proposal 🆕

These are all commonly available in userland promise libraries, and they're all independently useful, each one serving different use cases.

Illustrative examples

This snippet checks which endpoint responds the fastest, and then logs it.

```
Promise.any([
  fetch('https://v8.dev/').then(() => 'home'),
  fetch('https://v8.dev/blog').then(() => 'blog'),
  fetch('https://v8.dev/docs').then(() => 'docs')
]).then((first) => {
  // Any of the promises was fulfilled.
  console.log(first);
  // → 'home'
}).catch((error) => {
  // All of the promises were rejected.
  console.log(error);
});
```

Search...

TABLE OF CONTENTS

2 Promise.any (*iterable*)

The **any** function returns a promise that is fulfilled by the first given promise to be fulfilled, or rejected with an array of rejection reasons if all of the given promises are rejected. It resolves all elements of the passed iterable to promises as it runs this algorithm.

1. Let *C* be the **this** value.
2. Let *promiseCapability* be ? NewPromiseCapability(*C*).
3. Let *iteratorRecord* be GetIterator(*iterable*).
4. IfAbruptRejectPromise(*iteratorRecord*, *promiseCapability*).
5. Let *result* be PerformPromiseAny(*iteratorRecord*, *C*, *promiseCapability*).
6. If *result* is an abrupt completion, then
    a. If *iteratorRecord*.[[Done]] is **false**, set *result* to IteratorClose(*iteratorRecord*, *result*).
    b. IfAbruptRejectPromise(*result*, *promiseCapability*).
7. Return Completion(*result*).

NOTE The **any** function requires its **this** value to be a constructor function that supports the parameter conventions of the **Promise** constructor.

Permalink Pin References (1)

2.1 Runtime Semantics: PerformPromiseAny (*iteratorRecord*, *constructor*, *resultCapability*)

When the PerformPromiseAny abstract operation is called with arguments *iteratorRecord*, *constructor*, and *resultCapability*, the following steps are taken:

1. Assert: ! IsConstructor(*constructor*) is **true**.

Другие способы помочь

- вебсайт и другие документы на GitHub
- Outreach groups
- Babel
- тесты

Так что же помогло стать приглашенным экспертом?

- работа над Babel?
- обратная связь в репозиториях пропозалов
- помощь с сайтом и документацией
- работа над спецификацией
- знакомства и знания, полученные на конференциях?

Так что же помогло стать приглашенным экспертом?

¯_(ツ)_/¯

Социализируйтесь, исследуйте, развлекайтесь!

Спасибо за внимание!

Сергей Рубанов

@chicoxyzzy



Exantech