Let's work together on standards to advance JS and the Web

Daniel Ehrenberg
Igalia, in partnership with Bloomberg
Shenzhen, 2019-12-19

igalia
littledan@igalia.com

我

- Daniel Ehrenberg
- @littledan
- Delegate in TC39


家
Les Roquetes del Garraf, Europe

工作

igalia

Embedded WebKit and Chromium development

Mesa and GStreamer drivers

CSS, ARIA, WebAssembly, MathML, JavaScript standards+implementation in web browsers

igalia

littledan@igalia.com

Who is TC39?

- A committee of Ecma, with…
- JS developers
- JavaScript engines
- Transpilers
- Frameworks, libraries
- Academics
- Major websites/app platforms
- Now, some Chinese companies!
- etc

Search...

TABLE OF CONTENTS

- Introduction
- 1 Scope
- 2 Conformance
- 3 Normative References
- 4 Overview
- 5 Notational Conventions
- 6 ECMAScript Data Types and Values
- 7 Abstract Operations
- 8 Executable Code and Execution Contexts
- 9 Ordinary and Exotic Objects Behaviours
- 10 ECMAScript Language: Source Code
- 11 ECMAScript Language: Lexical Grammar
- 12 ECMAScript Language: Expressions
- 13 ECMAScript Language: Statements and Declarati...
- 14 ECMAScript Language: Functions and Classes
- 15 ECMAScript Language: Scripts and Modules
- 16 Error Handling and Language Extensions
- 17 ECMAScript Standard Built-in Objects
- 18 The Global Object
- 19 Fundamental Objects
- 20 Numbers and Dates
- 21 Text Processing
- 22 Indexed Collections
- 23 Keyed Collections
- 24 Structured Data

Draft ECMA-262 / April 23, 2018

ECMAScript® 2019 Language Specification

ecma INTERNATIONAL

Contributing to this Specification

This specification is developed on GitHub with the help of the ECMAScript community. There are a number of ways to contribute to the development of this specification:

GitHub Repository: https://github.com/tc39/ecma262
Issues: All Issues, File a New Issue
Pull Requests: All Pull Requests, Create a New Pull Request
Test Suite: Test262
Editor: Brian Terlson (E-mail, Twitter, GitHub)

Community:

- Mailing list: es-discuss
- IRC: #tc39 on freenode

tc39 / ecma262

Status, process, and documents for ECMA262 https://tc39.github.io/ecma262/

Edit

ecmascript javascript Manage topics

1,191 commits 10 branches 10 releases 85 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

jmdyck and bterlson Editorial: Misc Editorial (#1182) ... Latest commit cbe4679 20 hours ago

img	Editorial: Store and rethrow module instantiation/evaluation errors	9 months ago
scripts	Meta: Build ES2016 in a subfolder.	2 years ago
workingdocs	Editorial: Use HTTPS for ecma-international.org URLs (#1017)	6 months ago
.editorconfig	Add .editorconfig	3 years ago
.gitignore	Editorial: reword InstanceofOperator copy and rename vars (closes #894)	11 months ago
.travis.yml	meta: require node 8 to build the spec	4 months ago
CODE_OF_CONDUCT.md	Add CoC.md reference file (#1168)	13 days ago
CONTRIBUTING.md	Meta: Add paragraph about resolving conflicts with master. (#1099)	2 months ago
FAQ.md	Fix example file name in FAQ.	3 years ago
README.md	meta: Add Community section to readme.md	a year ago
figure-2.uxf	Editorial: fix size of figure 2.	2 years ago

Meetings

- Every two months
- For three days
- Discuss language changes
- Seek consensus on proposal
  Stage advancement

igalia

6. Web compatibility issues / Needs Consensus PRs

✓	timebox	topic
✓	30m	Normative: Fix extending null
✓	5m	Normative: Make super() throw after evaluating args
✓	10m	Normative: make async iterators next/return/throw not p `undefined` when value is absent
✓	25m	Normative: Eliminate extra environment for eval in para initializers redux (slides)
✓	25m	Normative: TypedArray on prototypes web reality (low pr (slides)
✓	15m	Normative: make EnumerableOwnPropertyNames ordere

7. Overflow from previous meeting

✓	timebox	topic	presenter

8. Short (≤30m) Timeboxed Discussions

✓	timebox	topic
✓	20m	async-of grammar ambiguity
✓	30m	RegExp match indices performance feedback (slides)
~	30m	Policy on published code/polyfills in proposal repos
✓	30m (could be 10m)	JSExplain demo (code, demo)

9. Proposals

✓ represents an agenda item which has been presented, and does not indic

✓	stage	timebox	topic	present
✓	3	15m	Intl.RelativeTimeFormat for stage 4 (slides)	Zibi Bra
✓	3	15m	For-in order for Stage 4 (PR, tests, slides)	Kevin Gi

TC39 stages

- Stage 1: An idea under discussion
- Stage 2: We want to do this, and we have a first draft
- Stage 3: Basically final draft; ready to go
- Stage 4: 2+ implementations, tests ⇒ standard

littledan@igalia.com

Consensus-based decision-making

- TC39 doesn't vote on what the language will be
  - (rarely vote on technicalities/procedural matters)
- TC39 is *consensus-seeking*
  - We work together to meet everyone's goals
  - "Does anyone object to stage advancement?"
  - Objections must have rationale, appropriate to stage

- Goals: Listen to everyone engaging in process;
  Don't choose one stakeholder over another as the "decider";
  Avoid standardizing things which aren't ready yet

littledan@igalia.com

Consensus-based decision-making

- Consensus is *established* at some point, but could always be revisited with new consensus.
  - Established consensus is not open to being "vetoed" later
  - Technical concerns should be raised as early as possible

- We assume *good faith*
  - We don't hide motives:
    We all assume that the true reasons are given for objections
- All TC39 delegates are considered equal

littledan@igalia.com

Stage 4 features

2+ implementations, tests ⇒ standard

littledan@igalia.com

BigInt: Stage 4!

littledan@igalia.com

```
const x = 2 ** 53;
```

⇒ x === 9007199254740992

```
const y = x + 1;
```

⇒ y === 9007199254740992

littledan@igalia.com

```
const x = 2n ** 53n;
      ⇒ x === 9007199254740992n
const y = x + 1n;
      ⇒ y === 9007199254740993n
```


littledan@igalia.com

```
const x = 2n ** 53n;
      ⇒ x === 9007199254740992n
const y = x + 1n;
      ⇒ y === 9007199254740993n
```

n stands for BigIntn

littledan@igalia.com

67 

68 

Beta 

10.4 

Flag 

Igalia's role in BigInt

- Wrote BigInt specification and some tests
- Standardized BigInt in TC39
- Implemented BigInt in SpiderMonkey (shipping) and JSC (in progress)

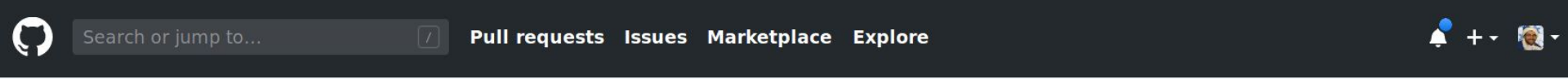
littledan@igalia.com

Dynamic import(): Stage 4

Domenic Denicola

igalia

littledan@igalia.com



tc39 / **proposal-dynamic-import**

Watch 110 | Star 1.5k | Fork 48

Code | Issues 11 | Pull requests 1 | Actions | Projects 0 | Security | Insights

Example

```
<script>
  const main = document.querySelector("main");
  for (const link of document.querySelectorAll("nav > a")) {
    link.addEventListener("click", e => {
      e.preventDefault();

      import(`./section-modules/${link.dataset.entryModule}.js`)
        .then(module => {
          module.loadPageInto(main);
        })
        .catch(err => {
          main.textContent = err.message;
        });
    });
  }
</script>
```

@littledan

igalia

Optional chaining: Stage 4

Daniel Rosenwasser
Claude Pache
Gabriel Isenberg
Dustin Savery

littledan@igalia.com

Optional Property Access

```
let x = foo?.bar;

// Equivalent to...

let x = (foo !== null && foo !== undefined) ?
  foo.bar :
  undefined;
```

Nullish coalescing: Stage 4

Daniel Rosenwasser
Gabriel Isenberg

littledan@igalia.com

Nullish Coalescing

```
let x = foo() ?? bar();

// Equivalent to...

let tmp = foo();
let x = (tmp !== null && tmp !== undefined) ?
  tmp :
  bar();
```

Stage 3 features

Basically final draft; ready to go

littledan@igalia.com

WeakRefs: Stage 3

Sathya Gunasekaran
Till Schneidereit
Mark Miller
Dean Tribble

littledan@igalia.com

Read consistency

```
const w = new WeakRef(someObject);

...

if (w.deref()) {
  w.deref().foo(); // w.deref() here can not fail
}
```

Private fields and methods: Stage 3

littledan@igalia.com

Why?

```
class Counter extends HTMLElement {
  #x = 0;

  connectedCallback() {
    this.#render();
  }
  #render() {
    this.textContent =
        this.#x.toString();
  }
}
```

- Private methods encapsulate behavior
- You can access private fields inside private methods

is the new _

for strong encapsulation

```
class PublicCounter {
  _x = 0;
}

let c = new PublicCounter();
console.log(c._x);    // 0
```

```
class PrivateCounter {
  #x = 0;
}

let p = new PrivateCounter();
console.log(p.#x);    // SyntaxError
```

Stage 3

littledan@igalia.com

Stage 2 features

We want to do this, and we have a first draft

littledan@igalia.com

Decorators: Stage 2

Yehuda Katz
Ron Buckton

littledan@igalia.com

Syntax abstraction for attrs/props in Web Components

```
// Polymer abstraction
class XCustom extends PolymerElement {
  @property({ type: String, reflect: true
})
  address = '';
}
```

```
// Salesforce abstraction
import { api } from '@salesforce';

class InputAddress extends HTMLElement {
    @api address = '';
}
```

Example of using decorators to improve ergonomics.

Temporal: Stage 2

Maggie Pint
Philipp Dunkel

igalia

littledan@igalia.com

When did this presentation start in Berlin?

```
let startDateTime =           // Temporal.DateTime
    Temporal.now.dateTime().with({ hour: 19, minute: 00 });

let startAbsolute =           // Temporal.Absolute
    startDateTime.inTimeZone(Temporal.now.timeZone());

let localizedToBerlin =       // Temporal.DateTime
    startAbsolute.inTimeZone("Europe/Berlin");

Intl.DateTimeFormat("zh", { hour: "numeric", minute: "numeric" })
    .format(localizedToBerlin);  // "下午2:00"
```


littledan@igalia.com

Stage 1 features

An idea under discussion

littledan@igalia.com

Pipeline operator: Stage 1

Gilbert Garza
J.S. Choi
James DiGioia

littledan@igalia.com

library.js

```
export function doubleSay(str) {
  return str + ", " + str;
}
export function capitalize(str) {
  return str[0].toUpperCase() +
      str.substring(1);
}
export function exclaim(str) {
  return str + "!";
}
```

ordinary.js

```
import { doubleSay, capitalize, exclaim }
         from "./library.js";
let result =
  exclaim(capitalize(doubleSay("hello")));
// ===> "Hello, hello!"
```

with-pipeline.js

```
import { doubleSay, capitalize, exclaim }
         from "./library.js";
let result = "hello"
  |> doubleSay
  |> capitalize
  |> exclaim;
// ===> "Hello, hello!"
```


littledan@igalia.com

Records and Tuples: Stage 1

Robin Ricard
Richard Button

littledan@igalia.com

```
const marketData = #[
    #{ ticker: "AAPL", lastPrice: 195.855 },
    #{ ticker: "SPY", lastPrice: 286.53 },
];
```

Operator overloading: Stage 1

littledan@igalia.com

Vector overloading: Usage

```
// Usage example
import { Vector } from "./vector.mjs";
with operators from Vector;

new Vector([1, 2, 3]) + new Vector([4, 5, 6])   // ==> new Vector([5, 7, 9])
3 * new Vector([1, 2, 3])                       // ==> new Vector([3, 6, 9])
new Vector([1, 2, 3]) == new Vector([1, 2, 3])  // ==> true
(new Vector([1, 2, 3]))[1]                      // ==> 2
```

Stage 0 features

Not even really at a stage!

littledan@igalia.com

BigDecimal: Stage 0

Andrew Paprocki

igalia

littledan@igalia.com

"Why are Numbers broken in JS?"

littledan@igalia.com

Problem and solution (?)

```
// Number (binary 64-bit floating point)

js> 0.1 + 0.2

                  =>          0.30000000000000004

// BigDecimal (???)

js> 0.1d + 0.2d

                  =>          0.3d
```


littledan@igalia.com

Other standards bodies

littledan@igalia.com

W3C WebAssembly CG

- Defines core WebAssembly bytecode features
- Working mode:
  - TC39-style stage process (5 Phases)
  - Call every two weeks
  - 1-3 in-person meetings a year
- Some current efforts: improve performance and JS integration
  - Exception handling
  - Interface types
  - GC proposal (starting with anyref, typed references, etc)
  - SIMD

littledan@igalia.com

W3C CSS WG

- Defines CSS (almost 100 specifications)
- Working mode:
  - Several levels of specifications published (some speculative, some implemented in browsers), according to W3C process
  - Weekly calls
  - 3-4 in-person meetings a year
- Some current efforts:
  - CSS Grid and subgrid
  - Color spaces
  - Line breaking options
  - Focus and spatial navigation
  - Houdini: Programmable paint, layout, etc
  - Logical writing modes

littledan@igalia.com

WHATWG: HTML, DOM, fetch, etc

- Defines core of processing HTML, network, etc
- Working mode:
  - No meetings
  - All on GitHub, in https://github.com/whatwg
  - PRs require support from 2 of 3 browsers, web-platform-tests
- Some current efforts
  - Style-able input elements
  - Web Components
  - Improvements to prefetching
  - Integration and consistency across the web platform
  - HTML, CSS, JSON module types

littledan@igalia.com

Many more

- WebAppSec
- WebPerf
- ARIA and a11y
- SVG
- WebFonts
- WebGPU
- WebXR
- MathML
- …

littledan@igalia.com

Why build on the open, standard web platform?

- Avoid control/dependence on a single vendor
- Compatibility over time
- Lower maintenance costs
- Large developer pool, documentation, etc
- Browsers are constantly improving

- Many groups face similar problems:
  Let's build infrastructure for shared solutions

littledan@igalia.com

Intl

littledan@igalia.com

Intl.NumberFormat

在此页面
语法
描述
NumberFormat 实例
例子
规范
浏览器兼容性
参见

相关主题
JavaScript 标准库
NumberFormat
属性
Intl.NumberFormat.prototype

Intl.NumberFormat是对语言敏感的格式化数字类的构造器类

语法

```
new Intl.NumberFormat([locales[, options]])
Intl.NumberFormat.call(this[, locales[, options]])
```

参数

locales

可选.缩写语言代码(BCP 47 language tag,例如:cmn-Hans-CN)的字符串或者这些字符串组成的数组. 关于参数locales的一般形式和解释请参见Intl page. 下面的这些Unicode扩展键也是被允许的:

Intl.DateTimeFormat

在此页面
- 语法
- 描述
- DateTimeFormat 实例
- 实例
- 规范
- 浏览器兼容性
- 相关链接

相关主题
JavaScript 标准库
DateTimeFormat
属性

您正在阅读此内容的英文版本，因为该语系尚未翻译。 帮助我们翻译此文章吧！

Intl.DateTimeFormat 是根据语言来格式化日期和时间的对象的构造器。

JavaScript Demo: Intl.DateTimeFormat

```
1  const date = new Date(Date.UTC(2012, 11, 20, 3, 0, 0));
2  // Results below assume UTC timezone - your results may vary
3
4  console.log(new Intl.DateTimeFormat('en-US').format(date));
5  // expected output: "12/20/2012"
6
7  console.log(new Intl.DateTimeFormat('en-GB').format(date));
8  // expected output: "20/12/2012"
9
10 // Include a fallback language, in this case Indonesian
11 console.log(new Intl.DateTimeFormat(['ban', 'id']).format(date));
12 // expected output: "20/12/2012"
13
```

Intl.Collator

在此页面
语法
描述
Collator 实例
例子
规范
浏览器兼容性
参见

相关主题
JavaScript 标准库
Collator
属性

`Intl.Collator` 是用于语言敏感字符串比较的 `collators`构造函数。

JavaScript Demo: Intl.Collator

```
1  function letterSort(lang, letters) {
2    letters.sort(new Intl.Collator(lang).compare);
3    return letters;
4  }
5
6  console.log(letterSort('de', ['a','z','ä']));
7  // expected output: Array ["a", "ä", "z"]
8
9  console.log(letterSort('sv', ['a','z','ä']));
10 // expected output: Array ["a", "z", "ä"]
11
```

Run ›

Browser compatibility

Update compatibility data on GitHub

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet	Node.js
Intl	24	12	29	11	15	10	4.4	25	56	Yes	10	1.5	Yes
Collator	24	12	29	11	15	10	≤37	25	56	Yes	10	1.5	?
DateTimeFormat	24	12	29	11	15	10	4.4	26	56	14	10	1.5	Yes
ListFormat	72	No	No	No	60	No	72	72	No	?	No	No	No
Locale	74	No	No	No	No	No	74	74	No	No	No	No	No
NumberFormat	24	12	29	11	15	10	Yes	26	56	14	10	1.5	?
PluralRules	63	18	58	No	50	No	63	63	58	46	No	8.0	10.0.0
RelativeTimeFormat	71	No	65	No	58	No	71	71	65	50	No	10.0	12.0.0
getCanonicalLocales	54	16	48	No	No	11	No	No	56	No	11	No	No

Intl standard: ECMA-402

- Specification parallel to ECMA-262 (ECMAScript)
- Use TC39 stage process
- Monthly, 2-hour calls in addition to TC39 meetings

- Igalia's work in Intl sponsored by Google, formerly by Mozilla

littledan@igalia.com

Advantages of using Intl

- Smaller bundles: Data already shipped in browser
- High quality: best practices from Unicode/CLDR
- Less effort for coders/translators

littledan@igalia.com

Newer Intl standards and proposals

Stage 4:

- formatToParts
- Intl.PluralRules
- Intl.RelativeTimeFormat

Stage 3:

- Intl.ListFormat
- Intl.Locale
- Intl.NumberFormat options
- Intl.DisplayNames
- dateStyle/timeStyle
- formatRange

Stage 2:

- Intl.Segmenter

Stage 0:

- Unicode Char Database API
- MessageFormat

littledan@igalia.com

Intl.RelativeTimeFormat: Stage 4

Zibi Braniecki

littledan@igalia.com

Intl.RelativeTimeFormat

- Shipped in Chrome and Firefox

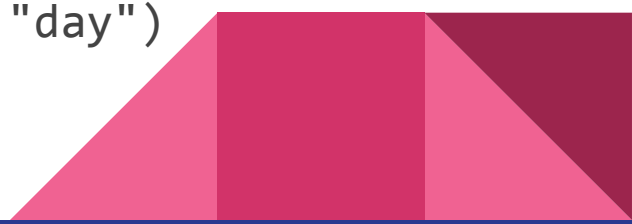
```
let rtf = new Intl.RelativeTimeFormat("en");

rtf.format(100, "day");
// "in 100 days"



new Intl.RelativeTimeFormat("zh").format(100, "day")

// "100天后"
```



Intl.DisplayNames: Stage 3

Frank Tang

littledan@igalia.com

Examples: Get Region Names in English

```
// Display names in English
let regionNames = new Intl.DisplayNames(
    ['en'], {type: 'region'});
regionNames.of('US'); // => "United States"
regionNames.of('419'); // => "Latin America"
regionNames.of('MM'); // => "Myanmar (Burma)"
```

Examples: Get Region Names in Traditional Chinese

```
// Display names in Traditional Chinese
let regionNames = new Intl.DisplayNames(
    ['zh-Hant'], {type: 'region'});
regionNames.of('US'); // => "美國"
regionNames.of('419'); // => "拉丁美洲"
regionNames.of('MM'); // => "緬甸"
```

Intl.Segmenter: Stage 2

Richard Gibson

igalia

littledan@igalia.com

```
// Create a locale-specific word segmenter
let segmenter = new Intl.Segmenter("fr", {granularity: "word"});

// Use it to get an iterator for a string
let input = "Moi?  N'est-ce pas.";
let segments = segmenter.segment(input);

// Use that for segmentation!
for (let {segment, index, isWordLike} of segments) {
  console.log("segment at code units [%d, %d): «%s»%s",
    index, index + segment.length,
    segment,
    isWordLike ? " (word-like)" : ""
  );
}
```

My suggestions for contributing to TC39 at different stages

littledan@igalia.com

Stage 0/1

- Document use cases
- Begin seeking feedback, and keep doing it the whole time!
- Discuss the problem space/big-picture questions
- Prototype implementations:
  unstable, rough, maybe downstream

littledan@igalia.com

Stage 2

- Nail down the proposal details, from discussion and prototyping
- Draft documentation and tests in the proposal repo
- Prototype implementations:
  Ideally nearing completion, but still considered unstable
- Distribute the prototype more broadly to gather more solid feedback

littledan@igalia.com

Stage 3

- Upstream tests into test262 and fill in any gaps.
- Place documentation in MDN and fill in the gaps
- Implement in engines and consider shipping
- Integrate usage into environments (e.g., the Web, Node.js)
- Distribute high-level explanations more broadly to developers

littledan@igalia.com

Scope of changes during Stage 3

- Changes are expected to be based on implementation feedback, e.g.,
  - Compatibility issues
  - Difficulty implementing/optimizing the feature
- TC39 consensus process used to review proposed changes
  - Committee unlikely to agree on big design changes.
- Please give design feedback earlier
- At Stage 3, proposals are implemented and ship in some browsers

littledan@igalia.com

Stage 4

- Ideally nothing!
- Tie up any loose ends in the spec language
- Implement in trailing engines if needed
- Further changes: separate PRs/proposals

littledan@igalia.com

Participating in TC39

littledan@igalia.com

Participating internationally

- Many TC39 members are in a similar situation to you all:
  - English is a second language
    - You can participate with mostly written communication
  - Live outside the US
    - Many delegates live in Europe
  - Unable to attend most TC39 meetings in person
    - Instead, join by video call
- TC39's code of conduct prohibits discrimination on nationality
- Many TC39 members want to increase international participation
- TC39 participants represent ideas and organizations, not countries

littledan@igalia.com

Participating asynchronously on GitHub

- Most tasks don't take place in meetings:
  Most important technical work happens on GitHub
- Work on GitHub:
  - Some of the discussion about design
  - Specification text
  - Documentation
  - Tests
  - Implementations
- Non-members can contribute on GitHub! We encourage it.
  - Just sign non-member IPR form
- Meeting notes are published to GitHub
- Only members can take part in meetings and consensus process

littledan@igalia.com

Joining TC39

- Join TC39 by joining Ecma
- Joining Ecma requires:
  - Signing IPR forms
    - Typical Ecma RAND policy
    - TC39-specific royalty-free agreement
  - Paying membership fee
  - Technically, a vote (typically a formality)
- TC39 *delegates* represent member *organizations*
- Companies considering joining can provisionally attend TC39 as "prospective members"

littledan@igalia.com

Representing member organizations

- Many member organizations send a few to represent many different internal stakeholders, including:
  - Front-end application developers
  - JavaScript infrastructure
    (bundlers, minifiers, type checkers, transpilers, etc)
  - Back-end Node.js and serverless
  - JS engine maintainers
- Common pattern: Internal committee to discuss concerns, meeting ahead of TC39 meetings to ensure company is well-represented

littledan@igalia.com

Good faith and building consensus

- Good qualities to demonstrate to colleagues in TC39:
  - Technical skill
  - Teamwork
  - Good faith
- Tactics for good relations:
  - Discuss ideas openly and calmly
  - Help with proposals
  - Respect committee process and consensus
- Good relations can build influence over the language

littledan@igalia.com

TC39 changes over time

littledan@igalia.com

Older TC39 mode

- Specification: Big MS Word doc
- Communication: Meetings and es-discuss list
- Large, occasional specification; no stages

littledan@igalia.com

ECMAScript 2015 Language Specification

1 Scope

This Standard defines the ECMAScript 2015 general purpose programming language.

2 Conformance

A conforming implementation of ECMAScript must provide and support all the types, values, objects, properties, functions, and program syntax and semantics described in this specification.

A conforming implementation of ECMAScript must interpret source text input in conformance with the Unicode Standard, Version 5.1.0 or later and ISO/IEC 10646. If the adopted ISO/IEC 10646-1 subset is not otherwise specified, it is presumed to be the Unicode set, collection 10646.

A conforming implementation of ECMAScript that provides an application programming interface that supports

Search...

TABLE OF CONTENTS

- Introduction
- 1 Scope
- 2 Conformance
- 3 Normative References
- 4 Overview
- 5 Notational Conventions
- 6 ECMAScript Data Types and Values
- 7 Abstract Operations
- 8 Executable Code and Execution Contexts
- 9 Ordinary and Exotic Objects Behaviours
- 10 ECMAScript Language: Source Code
- 11 ECMAScript Language: Lexical Grammar
- 12 ECMAScript Language: Expressions
- 13 ECMAScript Language: Statements and Declarati...
- 14 ECMAScript Language: Functions and Classes
- 15 ECMAScript Language: Scripts and Modules
- 16 Error Handling and Language Extensions
- 17 ECMAScript Standard Built-in Objects
- 18 The Global Object
- 19 Fundamental Objects
- 20 Numbers and Dates
- 21 Text Processing
- 22 Indexed Collections
- 23 Keyed Collections
- 24 Structured Data

Draft ECMA-262 / April 23, 2018

ECMAScript® 2019 Language Specification

ecma INTERNATIONAL

Contributing to this Specification

This specification is developed on GitHub with the help of the ECMAScript community. There are a number of ways to contribute to the development of this specification:

GitHub Repository: https://github.com/tc39/ecma262
Issues: All Issues, File a New Issue
Pull Requests: All Pull Requests, Create a New Pull Request
Test Suite: Test262
Editor: Brian Terlson (E-mail, Twitter, GitHub)

Community:

- Mailing list: es-discuss
- IRC: #tc39 on freenode

tc39 / ecma262

Status, process, and documents for ECMA262 https://tc39.github.io/ecma262/

Edit

ecmascript javascript Manage topics

1,191 commits 10 branches 10 releases 85 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

jmdyck and bterlson Editorial: Misc Editorial (#1182) ... Latest commit cbe4679 20 hours ago

img	Editorial: Store and rethrow module instantiation/evaluation errors	9 months ago
scripts	Meta: Build ES2016 in a subfolder.	2 years ago
workingdocs	Editorial: Use HTTPS for ecma-international.org URLs (#1017)	6 months ago
.editorconfig	Add .editorconfig	3 years ago
.gitignore	Editorial: reword InstanceofOperator copy and rename vars (closes #894)	11 months ago
.travis.yml	meta: require node 8 to build the spec	4 months ago
CODE_OF_CONDUCT.md	Add CoC.md reference file (#1168)	13 days ago
CONTRIBUTING.md	Meta: Add paragraph about resolving conflicts with master. (#1099)	2 months ago
FAQ.md	Fix example file name in FAQ.	3 years ago
README.md	meta: Add Community section to readme.md	a year ago
figure-2.uxf	Editorial: fix size of figure 2.	2 years ago

TC 39

Sign Up | Log In

all categories | all tags | Categories | Latest | Top

Category	Topics	Top
Announcements Welcome to TC39's focal communication hub.	0	
Proposals Discuss existing ECMAScript proposals.	2	
Ideas New ideas, proposals looking for a champion, and proposal mentoring. Get feedback on your new ideas here: what's been done, where to find prior art, the polish your proposal needs to be considered by the committee.	27	
I have questions Specifying a language is a complex affair, filled with confusing language and sometimes bewildering processes. It's okay not to understand, and we encourage you to ask questions! Spec Reading Delegates AMA	9	

Top		
	What should the categories be? Meta	18 Jun 7
	bigint enhancements. Proposals proposal	11 Oct 15
	Try-catch oneliner Ideas proposal	14 14d
	☑ Force HTTPS usage Meta	9 Apr 5
	New well-known Symbol: Symbol.typeofTag Ideas proposal	12 17d
G	JSON: Add Datetime, Timedelta and Binary Data Types	9 9d

Introduction

Like the technical community as a whole, TC39 is made up of a mixture of professionals and volunteers from all over the world. To ensure a fair and balanced standards process, to avoid communication issues and unhappiness, and to promote inclusiveness, we have a few ground rules that we ask people to adhere to.

This isn't an exhaustive list of things that you can't do. Rather, take it in the spirit in which it's intended a guide to make it easier to enrich all of us and the technical communities in which we participate and empower others to speak.

This Code of Conduct is enforced within all spaces managed by TC39. This includes IRC channels moderated by TC39, mailing lists such as esdiscuss, issue trackers on projects hosted by TC39, and TC39 events and meetings.

If you believe someone is violating the Code of Conduct, we ask that you report it by emailing tc39-conduct-reports@googlegroups.com. For more details, please see our Reporting Guidelines.

Be respectful

Respect is a fundamental value of the standardization work. Not all of us will agree all the time, but disagreement is no excuse for poor behavior and poor manners. We might all

README.md

Ecma International's TC39 is the standards committee which defines ECMAScript (JavaScript). This repository is intended to provide documentation into how TC39 works.

This repository is an early work in progress!

For an introduction to getting involved in TC39, see CONTRIBUTING.md.

Table of Contents

- Meta
  - Introduction and Table of Contents for this repo (this file)
  - List of suggested additions for this repo
- Proposals
  - Championing a proposal at TC39
  - How to write a good explainer
  - How to make a Pull Request against the ECMAScript specification

TC39

TC39 | Specs | Contribute | State of Proposals

Specifying JavaScript.

TC39

Ecma International's TC39 is a group of JavaScript developers, implementers, academics, and more, collaborating with the community to maintain and evolve the definition of JavaScript.

We are part of ecma INTERNATIONAL

Contribute

TC39 welcomes contributions. You can help by giving feedback on proposals, improving documentation, writing tests or implementations, or suggesting language feature ideas. See our contributor guide for details.

To participate in TC39 meetings as a member, join Ecma.

Specs

We develop the JavaScript (formally, ECMAScript) specification on GitHub and meet every two months to discuss proposals. To learn more about the process, please take a look at the four stages for new language feature proposals. See our meeting agendas and minutes to learn more.

State of Proposals

Current Candidates for the Specification

JavaScript and evidence-based language design

By Yulia Startsev

Posted on May 29, 2019 in Featured Article, JavaScript, and Standards

Author's note: Hi, I'm an engineer at Mozilla working on the Firefox DevTools server. I'm also a TC39 representative. This post focuses on some of the experiments I am trying out at the TC39, the standards body that manages the JavaScript specification. A follow up post will follow...

Integrating traditional and new values

- First-principles reasoning
- "The future is bigger than the past"
- Long, complete design cycles
- Rigorous debate
- Welcoming participants who join
- Language specialists/theorists

- Practicality
- Integration into the existing ecosystem
- Quick, incremental iteration
- Letting points be made without interruption
- Actively seeking out feedback
- More JS/frontend dev participation

All of these are useful,
complementary perspectives

littledan@igalia.com

谢谢!

https://tc39.es

Please write me any questions!
littledan@igalia.com

Add me on WeChat

Bonus slides

littledan@igalia.com

Igalia/Google AMP collaboration in WebKit

- Igalia is a consultancy; Google is a client
- AMP is a JavaScript framework for news/content sites
- Business case: Allow AMP to be fast and reliable on Safari
- Igalia's role (partial list):
  - Implement ResizeObserver in Safari
  - Advanced image preloading and lazy-loading
  - Assisted with IntersectionObserver in Safari
  - Scrolling API improvements and bug fixes
  - Improve iframes: fix flickering cases and add sandbox features
  - Web compatibility fixes in DOM, XMLHttpRequest and fetch
- WebKit is an open source project--possible to contribute upstream!

littledan@igalia.com

Igalia MathML crowdfunding effort

- Igalia's work sponsored by: NISO, Pearson, APS
  - We plan to extend crowdfunding to more small web platform features
- Business case: Make MathML universally available so that mathematics may be communicated on the web
- Igalia's role:
  - Refound MathML standards group, MathML Core specification
  - Define MathML interaction with CSS, ARIA, HTML, extensibly
  - Write 2000 conformance tests in web-platform-tests
  - Implement MathML Core in Chromium in LayoutNG
    - Based on TeX/OpenType layout
  - Improved interoperability with fixes in Firefox and Safari

littledan@igalia.com

PAST IMPLEMENTATIONS

Firefox/Gecko 2008

$$\Gamma(t) = \int_0^{+\infty} x^{t-1} e^{-x} dx = \frac{1}{t} \prod_{n=1}^{\infty} \frac{\left(1+\frac{1}{n}\right)^t}{1+\frac{t}{n}} \sim \sqrt{\frac{2\pi}{t}} \left(\frac{t}{e}\right)^t$$

Chrome/WebKit 2013

$$\Gamma(t) = \int_0^{+\infty} x^{t-1} e^{-x} dx = \frac{1}{t} \prod_{n=1}^{\infty} \frac{\left(1+\frac{1}{n}\right)^t}{1+\frac{t}{n}} \sim \sqrt{\frac{2\pi}{t}} \left(\frac{t}{e}\right)^t$$

Opera/Presto 2007

$$\Gamma(t) = \int_0^{+\infty} x^{t-1} e^{-x} dx = \frac{1}{t} \prod_{n=1}^{\infty} \frac{\left(1+\frac{1}{n}\right)^t}{1+\frac{t}{n}} \sim \sqrt{\frac{2\pi}{t}} \left(\frac{t}{e}\right)^t$$

2019 IMPLEMENTATIONS

Firefox/Gecko
July release

$$\Gamma(t) = \int_0^{+\infty} x^{t-1} e^{-x}\,dx = \frac{1}{t}\prod_{n=1}^{\infty} \frac{\left(1+\frac{1}{n}\right)^t}{1+\frac{t}{n}} \sim \sqrt{\frac{2\pi}{t}}\left(\frac{t}{e}\right)^t$$

Epiphany/WebKit
Build r249360

$$\Gamma(t) = \int_0^{+\infty} x^{t-1} e^{-x}\,dx = \frac{1}{t}\prod_{n=1}^{\infty} \frac{\left(1+\frac{1}{n}\right)^t}{1+\frac{t}{n}} \sim \sqrt{\frac{2\pi}{t}}\left(\frac{t}{e}\right)^t$$

Chromium/Blink
Igalia's Branch

$$\Gamma(t) = \int_0^{+\infty} x^{t-1} e^{-x}\,dx = \frac{1}{t}\prod_{n=1}^{\infty} \frac{\left(1+\frac{1}{n}\right)^t}{1+\frac{t}{n}} \sim \sqrt{\frac{2\pi}{t}}\left(\frac{t}{e}\right)^t$$

Metro analogy for stage process

- A TC39 proposal is like building a metro line
- Before Stage 3, we are designing the line and station plan
- When Stage 3 is reached, we decided on the line plan.
- After Stage 3, we start building the metro line.
- Stage 4 is declared once the line is completely built.
- If we discover ancient ruins in the path of the line, an adjustment will be forced to be made, even though the plan was decided.
- If a real estate developer asks for the line to be moved during Stage 3, he might not be able to convince the designers.
- If you want to influence where the line goes, tell the designers before the plan is set! Before Stage 3

littledan@igalia.com