ESNext: Proposals to look forward to

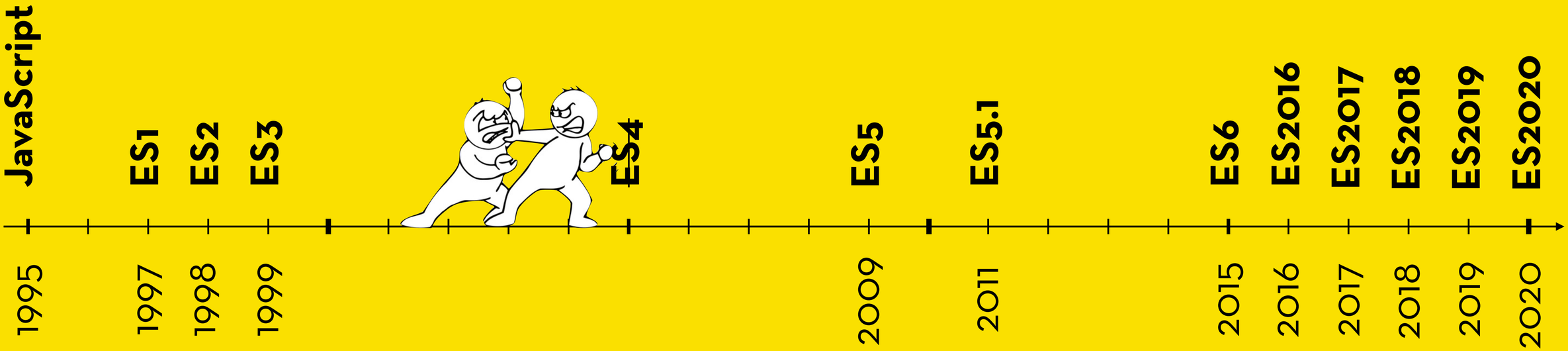
@bramus

July 15 – 16, 2021 @ JSCAMP VIRTUAL

@bramus

https://www.bram.us/
@bramusblog

ECMAScript® Timeline

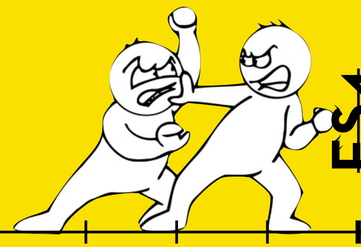


ECMAScript® Timeline

JavaScript	1995
ES1	1997
ES2	1998
ES3	1999
ES4	
ES5	2009
ES5.1	2011
ES6	2015
ES2016	2016
ES2017	2017
ES2018	2018
ES2019	2019
ES2020	2020

ECMAScript® Timeline

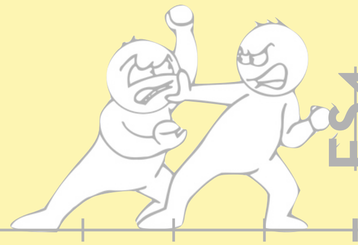
The next ECMAScript version will have features X, Y, and Z. Let's work on it until they are finished, and then tag a new version.



JavaScript 1995
ES1 1997
ES2 1998
ES3 1999
ES4
ES5 2009
ES5.1 2011
ES6 2015
ES2016 2016
ES2017 2017
ES2018 2018
ES2019 2019
ES2020 2020

ECMAScript® Timeline

Release a new version every year, and include the features that are finished.



JavaScript		ES1	ES2	ES3				ES4					ES5	ES5.1			ES6	ES2016	ES2017	ES2018	ES2019	ES2020
1995		1997	1998	1999									2009	2011			2015	2016	2017	2018	2019	2020

TC39

Technical Committee within Ecma International®
which is concerned with the "standardization of the
general purpose, cross platform, vendor-neutral
programming language ECMAScript®".

https://tc39.es/
https://www.ecma-international.org/memento/tc39.htm

ECMAScript?

ECMAScript® is a general purpose, cross platform, vendor-neutral programming language.

JavaScript is an implementation of ECMAScript®

https://www.ecma-international.org/memento/tc39.htm

TC39 Members

Developers
Implementers
Academics
Invited Experts

browsers

https://tc39.es/
https://ecma-international.org/memento/tc39-rf-taskgroup-members.htm

TC39 Meetings

Every two months

Meeting agendas and minutes shared online

Prososals are discussed, following the TC39 Development Process

https://tc39.es/
https://github.com/tc39/agendas/
https://github.com/tc39/tc39-notes/

TC39's Development Process

A 5 stage process by which it evolves the definition of ECMAScript®

https://tc39.es/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

TC39's Development Process

Stage 0 ~~Strawperson~~ → Stage 1 ~~Proposal~~ → Stage 2 ~~Draft~~ → Stage 3 ~~Candidate~~ → Stage 4 ~~Finished~~

https://tc39.es/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

TC39's Development Process

Stage 0 Strawperson → Stage 1 Proposal → Stage 2 Draft → Stage 3 Candidate → Stage 4 Finished

A free-form way of submitting ideas for evolving ECMAScript®, intended to spark the discussion.

No entrance criteria.

https://tc39.es/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

TC39's Development Process

Stage 0 Strawperson → **Stage 1 Proposal** → Stage 2 Draft → Stage 3 Candidate → Stage 4 Finished

A formal proposal described in prose and illustrated with examples, a high-level API and a discussion of semantics and algorithms.

Major changes allowed.

https://tc39.es/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

TC39's Development Process

Stage 0 Strawperson → Stage 1 Proposal → **Stage 2 Draft** → Stage 3 Candidate → Stage 4 Finished

A formal description precisely describing the semantics, syntax, and API in spec language.

Experimental implementation expected.
Eventual inclusion in the standard is likely.

browsers, Babel, ...

https://tc39.es/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

TC39's Development Process

Stage 0 Strawperson → Stage 1 Proposal → Stage 2 Draft → **Stage 3 Candidate** → Stage 4 Finished

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

Changes only in response to critical issues.

browsers

you

https://tc39.es/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

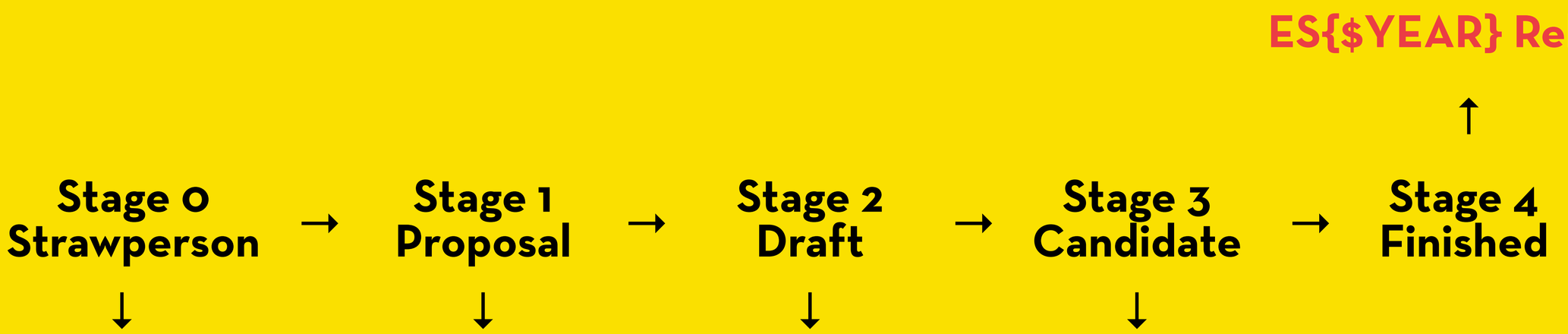
TC39's Development Process

Stage 0 Strawperson → Stage 1 Proposal → Stage 2 Draft → Stage 3 Candidate → **Stage 4 Finished**

The proposal is finished and ready to be included in the standard.

https://tc39.es/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

TC39's Development Process



https://tc39.es/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

Next ES{$YEAR} Release?

January
All Stage 4 proposals are marked for inclusion

March
Stage 4 proposals are incorporated into a draft spec and final semantics are approved.

July
Approval of new standard by the ECMA GA

https://tc39.es/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

ECMAScript® Timeline

JavaScript	1995
ES1	1997
ES2	1998
ES3	1999
ES4	
ES5	2009
ES5.1	2011
ES2015	2015
ES2016	2016
ES2017	2017
ES2018	2018
ES2019	2019
ES2020	2020

ESNext

A few of my favorite proposals / features

Stage 3

Field Declarations

👞 How classy!

https://github.com/tc39/proposal-class-fields

Stage 3

Field Declarations (1/3)

```
// ES2015-style:
class Counter extends HTMLElement {
    constructor() {
        super();
        this.clickCount = 0;
    }

    increment = () => {
        this.clickCount++;
    }
}
```

https://github.com/tc39/proposal-class-fields

Stage 3

Field Declarations (2/3)

```
// Public Fields
class Counter extends HTMLElement {
    clickCount = 0;

    increment = () => {
        this.clickCount++;
    }
}
```

https://github.com/tc39/proposal-class-fields

Stage 3

Field Declarations (3/3)

```
// What about Private?
class Counter extends HTMLElement {
    _clickCount = 0; // ❌ Still Public!

    increment = () => {
        this._clickCount++;
    }
}
```

https://github.com/tc39/proposal-class-fields

Stage 3

Field Declarations (3/3)

```
// Private Fields
class Counter extends HTMLElement {
    #clickCount = 0;

    increment = () => {
        this.#clickCount++;
    }
}
```

https://github.com/tc39/proposal-class-fields

ES2020

Optional Chaining

💔 Unbreak my ~~heart~~ code

https://brm.us/esnext-optional-chaining

Optional Chaining (1/4)

```
const message = {
    user: {
        firstName: 'Bramus',
        location: 'Belgium',
        twitter: '@bramus',
    },
    content: {
        body: '<p>…</p>',
    },
};

// Let's select some stuff …
console.log(message.user.firstName);
//~> "Bramus"
```

Optional Chaining (1/4)

```
const message = {
    user: {
       firstName: 'Bramus',
       location: 'Belgium',
       twitter: '@bramus',
    },
    content: {
       body: '<p>…</p>',
    },
};

// Let's select some inexistent stuff …
console.log(message.user.lastName);
//~> undefined
```

Optional Chaining (1/4)

```
const message = {
    user: {
       firstName: 'Bramus',
       location: 'Belgium',
       twitter: '@bramus',
    },
    content: {
       body: '<p>…</p>',
    },
};

// Let's select some inexistent stuff with a fallback value…
console.log(message.user.lastName || 'Anonymous');
//~> "Anonymous"
```

Optional Chaining (1/4)

```
const message = {
    user: {
       firstName: 'Bramus',
       location: 'Belgium',
       twitter: '@bramus',
    },
    content: {
       body: '<p>…</p>',
    },
};

// Let's select some inexistent-inexistent stuff with a fallback value…
console.log(message.meta.publicationDate || (new Date()).toISOString());
//~> ❌ “Cannot read property 'publicationDate' of undefined”
```

Optional Chaining (2/4)

```
// The Problem
console.log(message.meta.publicationDate || (new Date()).toISOString());
//~> ❌ “Cannot read property 'publicationDate' of undefined”

// A hacky workaround
const title = message && message.meta && message.meta.publicationDate || (…);

// Another hacky workaround
const title = (((message || {}).meta || {}).publicationDate || (…);

// The Non-Hacky Solution: The Optional Chaining Operator ?.
const title = message?.meta?.publicationDate || (new Date()).toISOString();
//~> "2021-07-13T15:08:12.633Z"
```

Optional Chaining (3/4)

"If the operand at the left-hand side of the optional chaining operator evaluates to `undefined` or `null`, the expression evaluates to `undefined`.

Otherwise the targeted property access, method or function call is triggered normally."

```
const title = message?.meta?.publicationDate || (new Date()).toISOString();
```

https://brm.us/esnext-optional-chaining

Optional Chaining (4/4)

The Optional Chaining Operator is spelled ***?.***
and can appear in three positions.

```
// optional static property access
obj?.prop

// optional dynamic property access
obj?.[expr]

// optional function or method call
func?.(...args)
```

Null Coalescing

😖 Damn you Optional Chaining!

https://brm.us/esnext-null-coalescing

Null Coalescing (1/3)

```
const message = {
    settings: {
        animationDuration: 0,
        showSplashScreen: false
    },
};

// Let's select some stuff … with a default value … in safe way …
const showSplashScreen = message?.settings?.showSplashScreen || true;
//~> true
```

Null Coalescing (2/3)

```
const message = {
    settings: {
        animationDuration: 0,
        showSplashScreen: false
    },
};

// The Solution: “Null Coalescing”
const showSplashScreen = message.settings?.showSplashScreen ?? true;
//~> false
```

Null Coalescing Operator (3/3)

"The nullary coalescing operator serves as an equality check against nullary values (`null` or `undefined`)"

```
const showSplashScreen = message?.settings?.showSplashScreen ?? true;
```

https://brm.us/esnext-null-coalescing

ES2021

Logical Assignment

... or else!

https://brm.us/esnext-logical-assignment

Stage 3

Logical Assignment (1/3)

```
function example(opts) {
  // Ok, but could trigger setter.
  opts.foo = opts.foo ?? 'bar'

  // No setter, but 'feels wrong' to write.
  opts.baz ?? (opts.baz = 'qux');
}

example({ foo: 'foo' })
```

https://brm.us/esnext-logical-assignment

Stage 3

Logical Assignment (2/3)

```
function example(opts) {
  // Setters are not needlessly called.
  opts.foo ??= 'bar'

  // No repetition of `opts.baz`.
  opts.baz ??= 'qux';
}

example({ foo: 'foo' })
```

https://brm.us/esnext-logical-assignment

Logical Assignment (3/3)

```
// Only assign if a is falsy (“Mallet Operator”)
a ||= b;

// Only assign if a is truthy
a &&= b;

// Only assign if a is null or undefined
a ??= b;
```

https://brm.us/esnext-logical-assignment

Dynamic Import

https://brm.us/esnext-dynamic-import

“Lazy Loading”

Without Dynamic Import

```
import lightbox from 'my-lightbox-module';

// …

lightboxLink.addEventListener('click', async (event) => {
    try {
        lightbox.open(event.target.href);
    } catch (error) {
        // …
    }
});
```

https://brm.us/esnext-dynamic-import

Dynamic Import

```
lightboxLink.addEventListener('click', async (event) => {
    try {
        const lightbox = await import('./lightbox.mjs');
        lightbox.open(event.target.href);
    } catch (error) {
        // …
    }
});
```

https://brm.us/esnext-dynamic-import

Sidenote: Native Image Lazy Loading

```
<!-- Lazy-load an offscreen image when the user scrolls near it -->
<img src="unicorn.jpg" loading="lazy" alt=".."/>

<!-- Load an image right away instead of lazy-loading -->
<img src="unicorn.jpg" loading="eager" alt=".."/>

<!-- Browser decides whether or not to lazy-load the image -->
<img src="unicorn.jpg" loading="auto" alt=".."/>
```

https://brm.us/lazy-loading

Stage 1

Decimal

 Let's count to 0.3!

https://github.com/tc39/proposal-decimal

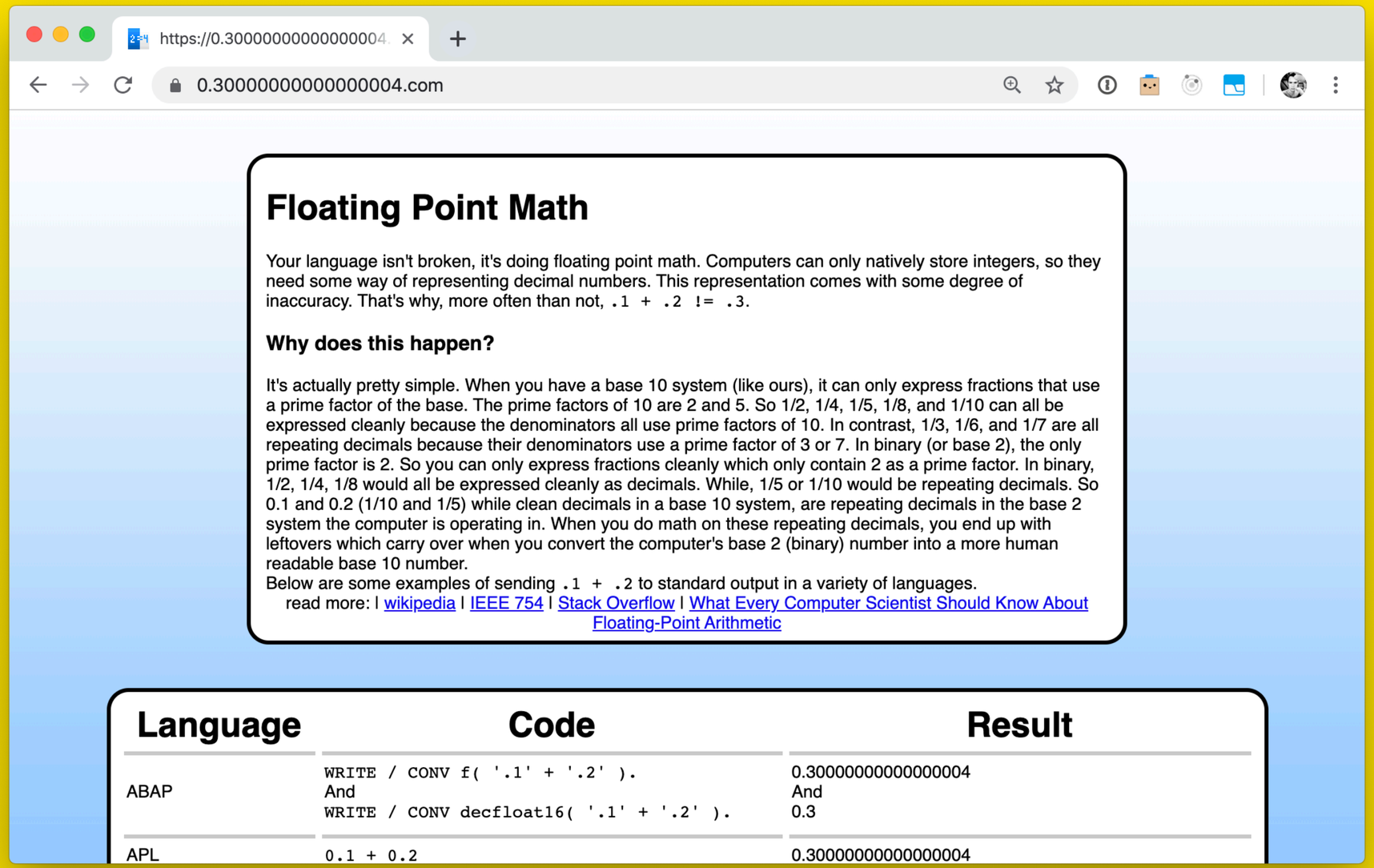
Stage 1

Decimal (1/2)

```
const result = 0.1 + 0.2;
console.log(result);

// ~> 0.30000000000000004
```

https://github.com/tc39/proposal-decimal



https://0.30000000000000004.com/

Stage 1

Decimal (2/2)

```
const result = 0.1m + 0.2m;
console.log(result);

// ~> 0.3m
```

https://github.com/tc39/proposal-decimal

Stage 1

Cancellation API

✋ Pinky Swear … or not.

https://github.com/tc39/proposal-cancellation/

Cancellation API (1/3)

```
function doSomethingAsync(url) {
    return new Promise(function(resolve, reject) {
        console.log('Promise Started');
        const timeout = window.setTimeout(function() {
            console.log('Promise Resolved');
            resolve();
        }, 2500);
    });
}

doSomethingAsync('https://www.bram.us/');
//~> "Promise Started"
//~> "Promise Resolved"
```

How can we cancel the execution of `doSomethingAsync`?

Stage 1

Cancellation API (2/3)

```
function doSomethingAsync(url, cancellationToken = CancellationToken.none) {
    return new Promise(function(resolve, reject) {
        cancellationToken.throwIfCancellationRequested();

        const registration = cancellationToken.register(function() {
            console.log('Promise Cancelled');
            reject(new CancelError());
        });

        console.log('Promise Started');
        const timeout = window.setTimeout(function() {
            console.log('Promise Resolved');
            registration.unregister();
            resolve();
        }, 2500);
    });
}
```

https://github.com/tc39/proposal-cancellation/

Stage 1

Cancellation API (2/3)

```
const source = new CancellationTokenSource();
setTimeout(() => source.cancel(), 1000);
doSomethingAsync(url, source.token);

//~> "Promise Started"
//~> "Promise Cancelled"
```

https://github.com/tc39/proposal-cancellation/

Stage 1

Cancellation API (3/3)

```
document.getElementById('#livesearch').addEventListener('change', (e) => {
    cancelPreviousSearchRequestIfAny();

    performSearchRequest({
        url: e.target.getAttribute('data-url'),
        q: e.target.value,
    });
});
```

https://github.com/tc39/proposal-cancellation/

Vendor specific solution

Sidenote: AbortController

```
function doSomethingAsync({signal}) {
    if (signal.aborted) {
        return Promise.reject(new DOMException('Aborted', 'AbortError'));
    }
    return new Promise((resolve, reject) => {
        console.log('Promise Started');
        const timeout = window.setTimeout(resolve, 2500, 'Promise Resolved')
        signal.addEventListener('abort', () => {
            window.clearTimeout(timeout);
            reject(new DOMException('Aborted', 'AbortError'));
        });
    });
}

const controller = new AbortController();
doSomethingAsync({ signal: controller.signal });
```

https://brm.us/abortcontroller

ES2021

String#replaceAll()

#️⃣ No More Regexes!

https://brm.us/string-replaceall

String#replaceAll()

```
'the quick brown fox jumps over the lazy dog'.replace('o', '_')
// ~> "the quick br_wn fox jumps over the lazy dog"

'the quick brown fox jumps over the lazy dog'.replace(/o/g, '_')
// ~> "the quick br_wn f_x jumps _ver the lazy d_g"

'the quick brown fox jumps over the lazy dog'.replaceAll('o', '_')
// ~> "the quick br_wn f_x jumps _ver the lazy d_g"
```

https://brm.us/string-replaceall

Stage 1

Declarations in Conditionals

https://github.com/tc39/proposal-Declarations-in-Conditionals

Stage 1

Declarations in Conditionals (1/2)

```
if (foo.data) {
    for (let item of foo.data) {
        // …
    }
}
```

https://github.com/tc39/proposal-Declarations-in-Conditionals

Stage 1

Declarations in Conditionals (1/2)

```
let data = foo.data;
if (data) {
    for (let item of data) {
        // …
    }
}
```

https://github.com/tc39/proposal-Declarations-in-Conditionals

Stage 1

Declarations in Conditionals (2/2)

```
if (let data = foo.data) {
    for (let item of data) {
        // …
    }
}
```

https://github.com/tc39/proposal-Declarations-in-Conditionals

Stage 3

Temporal

 A modern Date/Time API …

https://tc39.es/proposal-temporal/docs/

Stage 3

Instant vs. PlainDateTime vs. ZonedDateTime

```
const nowInstant = Temporal.now.instant();
console.log(nowInstant.toString());
// ~> 2021-07-12T20:09:24.212564211Z

const nowPlainDateTimeISO = Temporal.now.plainDateTimeISO();
console.log(nowPlainDateTimeISO.toString());
// ~> 2021-07-12T22:09:24.219564212

const zonedDateTimeISO = Temporal.now.zonedDateTimeISO();
console.log(zonedDateTimeISO.toString());
// ~> 2021-07-12T22:09:24.219564212+02:00[Europe/Brussels]
```

https://tc39.es/proposal-temporal/docs/

Stage 3

Time Manipulation

```
console.log(
    Temporal.ZonedDateTime.from('2020-01-09T00:00[America/Chicago]')
    .add({ hours: 8, minutes: 30 })
    .withTimeZone(Temporal.now.timeZone())
    .toString()
);
// ~> 2020-01-09T15:30:00+01:00[Europe/Brussels]
```

https://tc39.es/proposal-temporal/docs/

Stage 1

Slice Notation

🔪 Chop it up!

https://github.com/gsathya/proposal-slice-notation/

Stage 1

Slice Notation

```
const arr = ['a', 'b', 'c', 'd'];

// Old way
arr.slice(1, 3);
// ~> ['b', 'c']

// New way
arr[1:3];
```

https://github.com/gsathya/proposal-slice-notation/

Stage 1

Slice Notation

```
// Works on strings too!
const str = 'Bramus';
str[3:];
```

https://github.com/gsathya/proposal-slice-notation/

Stage 1

Pattern Matching

https://github.com/tc39/proposal-pattern-matching

Pattern Matching Syntax (1/3)

```
const getLength = vector => {
    if (vector.hasOwnProperty('x') && vector.hasOwnProperty('y') &&
        vector.hasOwnProperty('z')
    ) {
        return Math.sqrt(x ** 2 + y ** 2 + z ** 2);
    }
    if (vector.hasOwnProperty('x') && vector.hasOwnProperty('y')) {
        return Math.sqrt(x ** 2 + y ** 2);
    }
    if (vector.hasOwnProperty('length')) {
        return vector.length;
    }
    throw new Error('Unknown vector type');
}
```

https://github.com/tc39/proposal-pattern-matching

Stage 1

Pattern Matching Syntax (2/3)

```
const getLength = vector => {
    return match (vector) {
        when ({ x, y, z }) Math.sqrt(x ** 2 + y ** 2 + z ** 2);
        when ({ x, y }) Math.sqrt(x ** 2 + y ** 2);
        when ([...etc]) vector.length;
    }
};
```

https://github.com/tc39/proposal-pattern-matching

Stage 1

Pattern Matching Syntax (3/3)

```
const res = await fetch(jsonService)
match (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);
  }
};
```

https://github.com/tc39/proposal-pattern-matching

… and that's just a selection

- Static Instance Fields
- Decorators
- `global`
- `BigInt`
- WeakRefs
- New `Set` methods
- Matching & Destructuring Aliasing
- Pipeline Operator
- `Object.freeze()` & `Object.seal()`
- Default exports
- `Promise.try`
- Partial application
- `Array#lastItem` & `Array#lastIndex`
- `Array#filterOut()`
- `deprecated;`
- Mixins
- Export Default From
- `String#codePoints()`
- Numeric Separators
- Records & Tuples
- Function Implementation Hiding
- URL
- `Date.parse()` vs. RFC3339
- `Number.range()` & `BigInt.range()`
- …

… and that's just a selection

- Static Instance Fields
- Decorators
- `global`
- `BigInt`
- WeakRefs
- New Set methods
- Matching & Destructuring/Aliasing
- Pipeline Operator
- `Object.fromEntries()` & `Object.entries()`
- Default exports
- `Promise.try`
- Partial application
- `Array#lastItem` & `Array#lastIndex`
- `Array#filterOut()`
- deprecated:
- Mixins
- Export Default From
- `String#codePoints()`
- Numeric Separators
- Records & Tuples
- Function Implementation Hiding
- UUIDs
- `Date.parse()` vs. RFC3339
- `Number.range()` & `BigInt.range()`
- …

https://github.com/tc39/proposals

https://github.com/tc39/proposals/blob/master/stage-0-proposals.md

https://github.com/tc39/proposals/blob/master/finished-proposals.md

https://github.com/tc39/proposals/blob/master/inactive-proposals.md

… and that's just a selection

- Static Instance Fields
- Decorators
- `global`
- `BigInt`
- WeakRefs
- New `Set` methods
- Matching & Destructuring Aliasing
- Pipeline Operator
- `Object.freeze()` & `Object.seal()`
- Default exports
- `Promise.try`
- Partial application
- `Array#lastItem` & `Array#lastIndex`

https://www.bram.us/

https://brm.us/esnext-proposals

- `Array#filterOut()`
- `deprecated;`
- Mixins
- Export Default From
- `String#codePoints()`
- Numeric Separators
- Records & Tuples
- Function Implementation Hiding
- URL
- `Date.parse()` vs. RFC3339
- `Number.range()` & `BigInt.range()`
- …

ESNext

Welcome to the world of tomorrow!

ESNext

Welcome to the world of ~~tomorrow~~!

ESNext

Welcome to the world of today!

TC39's Development Process

Stage 0 Strawperson → Stage 1 Proposal → Stage 2 Draft → **Stage 3 Candidate** → Stage 4 Finished

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

Changes only in response to critical issues.

browsers

you

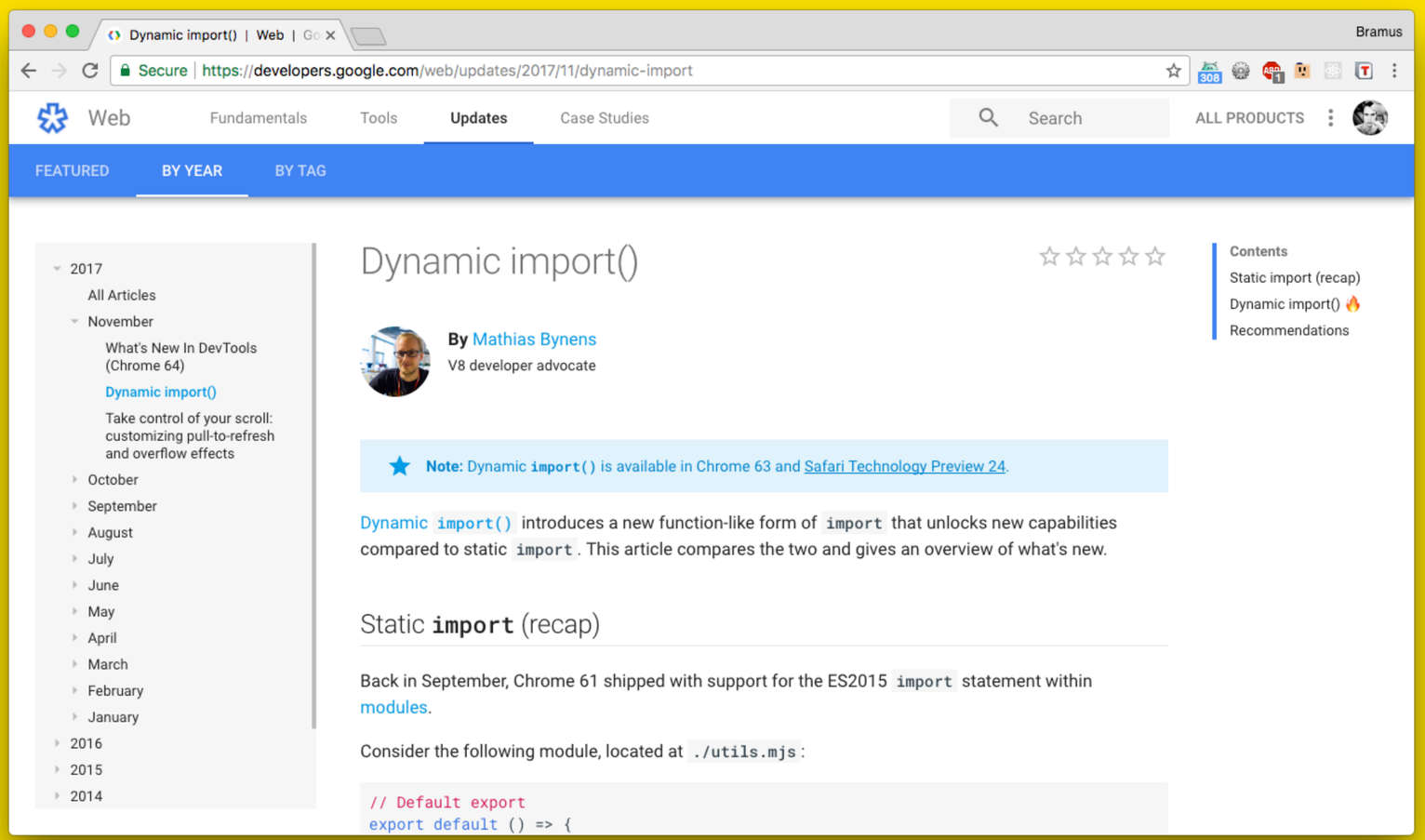
https://tc39.es/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

ESNext, now

https://github.com/babel/proposals

https://babeljs.io/

ESNext, now



https://developers.google.com/web/updates/2017/11/dynamic-import

Thanks!

@bramus

https://www.bram.us/
@bramusblog

ESNext: Proposals to look forward to

@bramus