ESNext: Proposals to look forward to

@bramus

04/10/18 @ Fronteers (Jam Session)

CAUTION!

Changes Ahead

Everything I'm about to say is subject to change.

The features mentioned are still proposals.

They need to be approved by TC39 before they can become part of the ECMAScript Language Specification.

TC39?

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

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

TC39 Members

Its members are companies
(all major browser vendors, among others).

TC39 meets regularly, its meetings are attended by
delegates that members send, and by invited experts.

http://exploringjs.com/es6/ch_about-es6.html#sec_tc39

TC39's Development Process

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

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

TC39's Development Process

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

A free-form way of submitting ideas for evolving ECMAScript.

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

TC39's Development Process

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

A formal proposal for the feature.

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

TC39's Development Process

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

A first version of what will be in the specification.
Eventual inclusion of the feature in the standard is likely.

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

TC39's Development Process

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

The proposal is mostly finished and now needs feedback from implementations and users to progress further.

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

TC39's Development Process

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

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

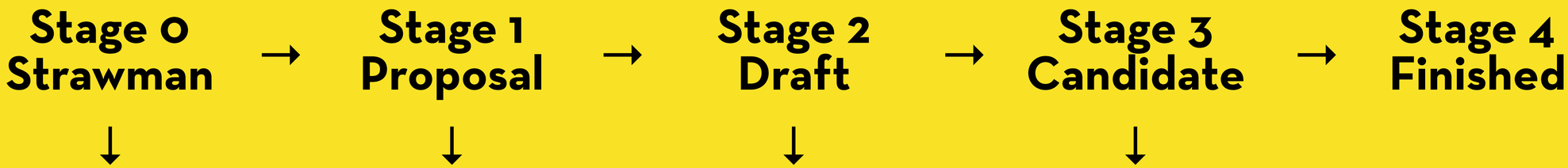
https://tc39.github.io/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

TC39's Development Process

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

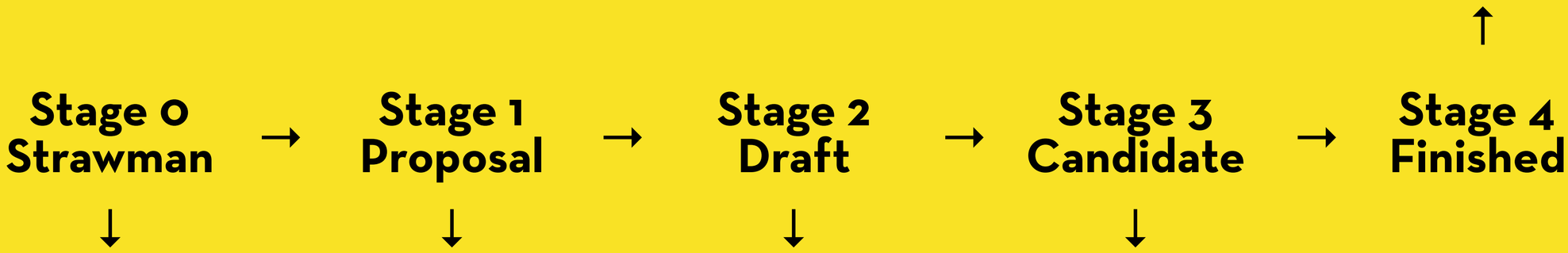
https://tc39.github.io/process-document/
http://exploringjs.com/es2016-es2017/ch_tc39-process.html

TC39's Development Process



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

TC39's Development Process



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

Next ESNext Release?

When the spec goes through its yearly ratification as a standard (mostly around June), all stage 4 proposals are ratified as part of it.

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

ESNext

A few of my favorite proposals

Stage 1

Optional Chaining

💔 Unbreak my ~~heart~~ code

https://github.com/tc39/proposal-optional-chaining
https://www.bram.us/2017/01/30/javascript-null-propagation-operator/

Optional Chaining (1/4)

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

Optional Chaining (1/4)

```
const message = {
    user: {
        firstName: 'Bramus',
        city: 'Vinkt',
        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',
       city: 'Vinkt',
       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',
       city: 'Vinkt',
       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',
       city: 'Vinkt',
       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();
//~> "2018-10-04T20:42:12.633Z"
```

Stage 1

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://github.com/tc39/proposal-optional-chaining
https://www.bram.us/2017/01/30/javascript-null-propagation-operator/

Stage 1

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://github.com/tc39/proposal-nullish-coalescing
https://www.bram.us/2018/02/05/esnext-javascript-nullish-coalescing-operator/

Null Coalescing (1/3)

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

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://github.com/tc39/proposal-nullish-coalescing
https://www.bram.us/2018/02/05/esnext-javascript-nullish-coalescing-operator/

Cancellation API

🤚 Pinky Swear … or not.

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

Cancellation API (1/2)

```
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`?

Cancellation API (2/2)

```
function doSomethingAsync(url, cancellationToken = CancellationToken.none) {
    return new Promise((resolve, reject) => {
        cancellationToken.throwIfCancellationRequested(); // [1]

        const registration = cancellationToken.register(function() { // [2]
            console.log('Promise Cancelled');
            reject(new CancelError());
        });

        console.log('Promise Started');
        const timeout = window.setTimeout(function() {
            console.log('Promise Resolved');
            registration.unregister(); // [3]
            resolve();
        }, 2500);
    });
}
```

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

Stage 1

Cancellation API (2/2)

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

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

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

… and that's just a selection

- Dynamic Imports
- Class Fields
- Private Methods and Fields
- Decorators
- `global`
- `BigInt`
- WeakRefs
- Pattern Matching
- New `Set` methods
- Array Slice Notation
- Realms
- Default exports

- `Promise.try`
- Temporal
- Function Bind Syntax
- Partial application
- `String#codePoints`
- `Object.fromEntries`
- `Object.freeze` & `Object.seal`
- `Array#lastItem`
- `deprecated;`
- Mixins
- Pipeline Operator
- …

… and that's just a selection

- Dynamic Imports
- Class Fields
- Private Methods and Fields
- Decorators
- `global`
- `BigInt`
- WebAssembly
- Pattern Matching
- New `Set` methods
- Array Slice Notation
- Realms
- Default exports

https://github.com/tc39/proposals

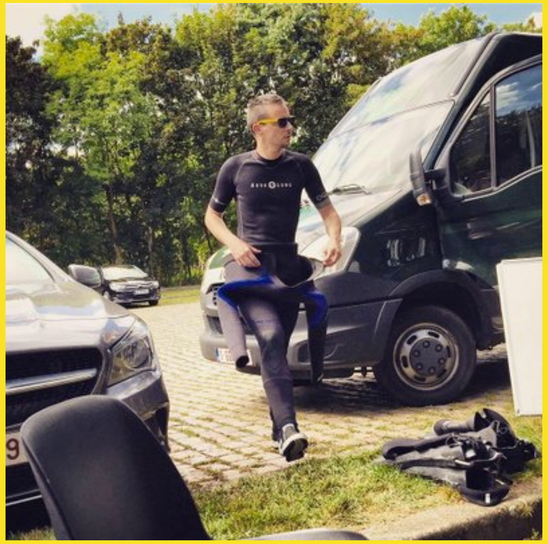
https://github.com/tc39/proposals/blob/master/stage-0-proposals.md

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

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

- `Promise.try`
- Temporal
- Function Bind Syntax
- Partial application
- String#codePoints
- `Object.fromEntries`
- Object.observe
- `Array#lastItem`
- Decorators
- Mixins
- Pipeline Operator
- …

Thanks!



@bramus

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

ESNext: Proposals to look forward to

@bramus

04/10/18 @ Fronteers (Jam Session)