ESNext: Proposals to look forward to

@bramus

13/03/19 @ Full Stack Ghent

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.

ECMAScript?

ECMAScript is a general purpose, cross platform, vendor-neutral programming language upon which JavaScript is based.

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

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.

including browsers!

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

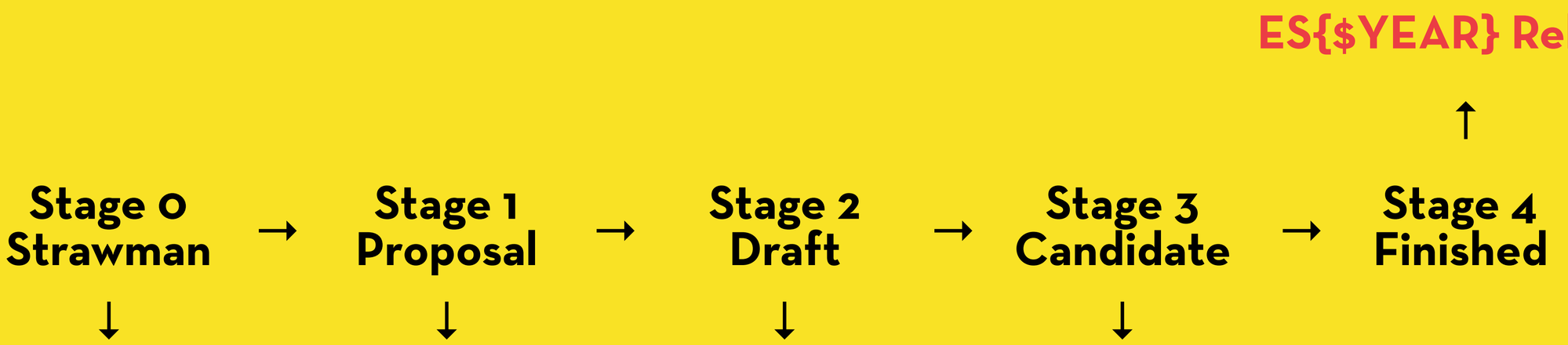
TC39's Development Process

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



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

Next ES{$YEAR} 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 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)

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

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

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

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>',
    },
};

// 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"
```

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)
```

Stage 1

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
    },
};

// 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/

Stage 1

Pipeline Operator

🔗 Chain it up!

https://github.com/tc39/proposal-pipeline-operator
https://www.bram.us/2017/09/02/esnext-proposal-the-pipeline-operator/

Stage 1

Pipeline Operator

```
// Classic Way
const result = exclaim(capitalize(doubleSay("hello")));

// With Pipeline Operator
const result = "hello"
    |> doubleSay
    |> capitalize
    |> exclaim;
```

https://github.com/tc39/proposal-pipeline-operator
https://www.bram.us/2017/09/02/esnext-proposal-the-pipeline-operator/

Stage 3

Dynamic Import

https://github.com/tc39/proposal-dynamic-import
https://www.bram.us/2017/11/27/dynamically-import-es-modules-with-dynamic-import/

Stage 3

Dynamic Import

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

https://github.com/tc39/proposal-dynamic-import
https://www.bram.us/2017/11/27/dynamically-import-es-modules-with-dynamic-import/

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.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 => {
    case (vector) {
        when { x, y, z } -> Math.sqrt(x ** 2 + y ** 2 + z ** 2),
        when { x, y } -> Math.sqrt(x ** 2 + y ** 2),
        when [...] -> vector.length,
        when _ -> throw new Error('Unknown vector type');
    }
}
```

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

Stage 1

Pattern Matching Syntax (3/3)

```
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);
    }
}
```

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

Promises

♻️ Pinky swear … a quick refresher

Promises

```
// Before: Callback Hell
callApi('https://example.com/endpoint1', function(result1) {
    callApi('https://…/endpoint2/' + result1.id + '/', function(result2) {
        callApi('https://…/endpoint3/' + result2.id + '/', function(result3) {
            // …
        });
    });
});
```

Promises

```
// After:
callApi('https://example.com/endpoint1')
    .then(result1 => {
        return callApi(`https://…/endpoint2/${result1.id}/`);
    })
    .then(result2 => {
        return callApi(`https://…/endpoint3/${result2.id}/`);
    })
    .then(result3 => {
        // …
    })
    .catch(err => {
       //  …
    });
```

Promises: Real world example

```
fetch(url, { headers: { 'Accept': 'application/json' } })
    .then(response => {
        if (!response.ok) {
            throw Error(response.statusText);
        }
        return response.json(); // <- Also a promise
    })
    .then(json => {
        // …
    })
    .catch(err => {
        // …
    });
```

Promises: Roll your own!

```
const load = () => new Promise((resolve, reject) => {
    const result = doSomething();
    if (result) {
        resolve(result);
    } else {
        reject('Oopz!');
    }
});

load()
    .then(result => {/* … */})
    .catch(err => {/* … */});
```

Stage 1

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`?

Stage 1

Cancellation API (2/2)

```
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 (2/2)

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

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

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

Stage 0

Function Bind Syntax

🤔 What is this?

https://github.com/tc39/proposal-bind-operator

Stage 0

Function bind Syntax (1/4)

```
// Won't work
class Foo extends React.Component {
    clickCount = 0;

    onClick(e) {
        e.preventDefault();
        this.clickCount++;
    }

    render() {
        return <a onClick={this.onClick} href="#">Click Me!</a>;
    }
}
```

https://github.com/tc39/proposal-bind-operator

Function bind Syntax (2/4)

```
// Oldskool way of fixing it
class Foo extends React.Component {
    clickCount = 0;

    constructor(props) {
        super(props);
        this.onClick = this.onClick.bind(this);
    }

    onClick(e) {
        e.preventDefault();
        this.clickCount++;
    }

    render() {
        return <a onClick={this.onClick} href="#">Click Me!</a>;
    }
}
```

Stage 0

Function bind Syntax (3/4)

```
// ES2015 way of fixing it
class Foo extends React.Component {
    clickCount = 0;

    onClick = (e) => {
        e.preventDefault();
        this.clickCount++;
    }

    render() {
        return <a onClick={this.onClick} href="#">Click Me!</a>;
    }
}
```

https://github.com/tc39/proposal-bind-operator

Function bind Syntax (4/4)

```
// With Function Bind Syntax
class Foo extends React.Component {
    clickCount = 0;

    constructor(props) {
        super(props);
        this.onClick = ::onClick;
    }

    onClick(e) {
        e.preventDefault();
        this.clickCount++;
    }

    render() {
        return <a onClick={this.onClick} href="#">Click Me!</a>;
    }
}
```

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/

ES2019

Object.fromEntries()

🔀 Is it worth it, let me work it.

https://github.com/tc39/proposal-object-from-entries

Object.values/Object.entries

```
const user = {
    firstName: 'Bramus',
    lastName: 'Van Damme',
    twitter: 'bramus',
};

Object.values(user);
// [ "Bramus", "Van Damme", "bramus" ]

Object.entries(user);
// [
//     [ "firstName", "Bramus" ],
//     [ "lastName", "Van Damme" ],
//     [ "twitter", "bramus" ],
// ]
```

http://exploringjs.com/es2016-es2017/pt_es2017.html
https://www.bram.us/2017/07/18/es2017-es8-language-features/

Object.fromEntries

```
const obj = { x: 42, y: 50 };

const entries = Object.entries(obj);
// → [['x', 42], ['y', 50]]

const result = Object.fromEntries(entries);
// → { x: 42, y: 50 }
```

https://github.com/tc39/proposal-object-from-entries

… and that's just a selection

- Static Instance Fields
- Decorators
- `global`
- `BigInt`
- WeakRefs
- New `Set` methods
- Realms
- Default exports
- `Promise.try`

- Temporal
- Partial application
- `String#codePoints`
- `Object.freeze` & `Object.seal`
- `Array#lastItem`
- `deprecated;`
- Mixins
- …

… and that's just a selection

- Static Instance Fields
- Decorators
- `global`
- `BigInt`
- WeakRefs
- New Set methods
- Realms
- Default exports
- Promise.try

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

- Temporal
- Partial application
- `String#codePoints`
- `Object.freeze` & `Object.seal`
- `Array#lastItem`
- `deprecated;`
- Pipeline
- …

ESNext

Welcome to the world of tomorrow!

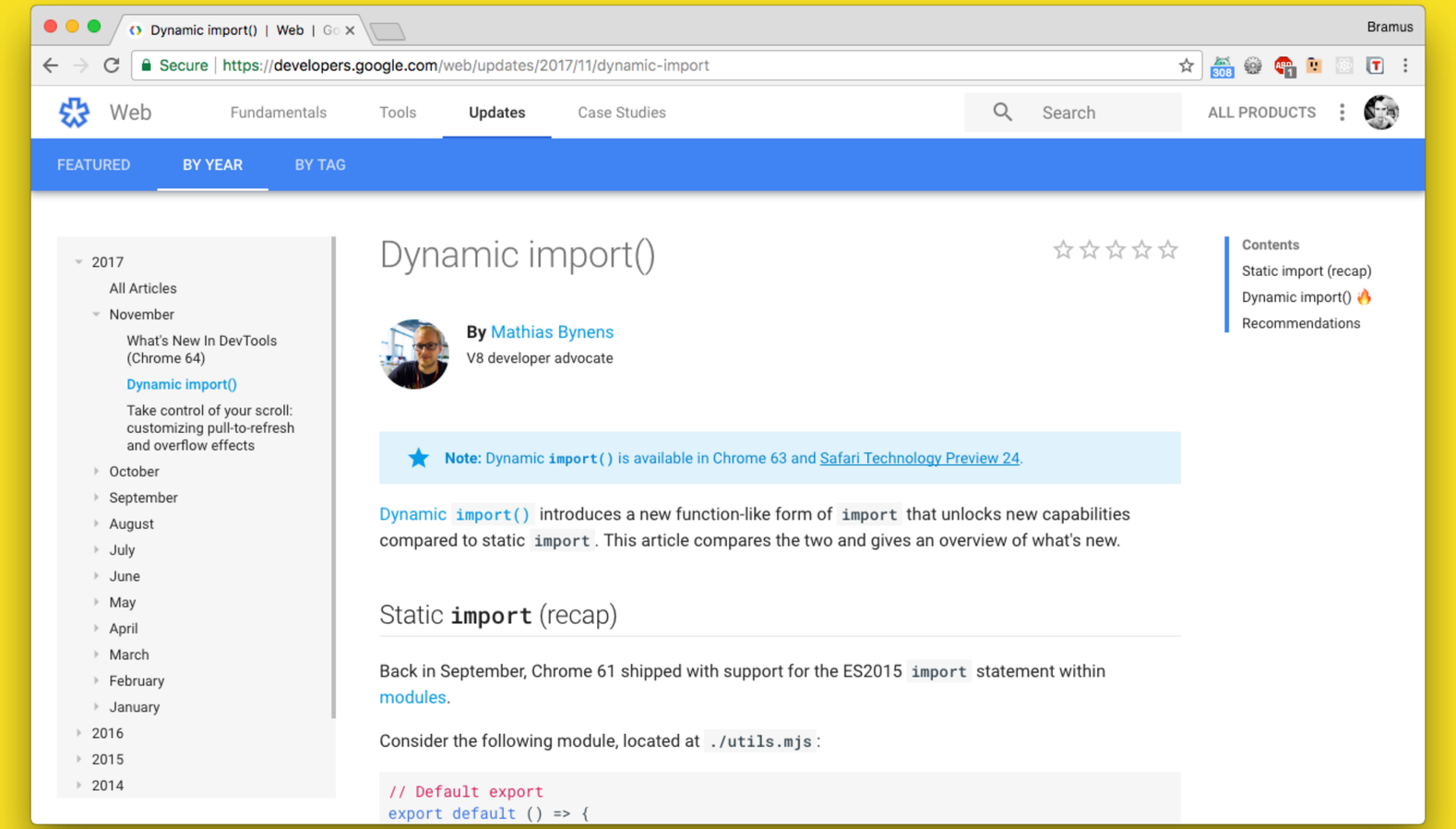
ESNext

Welcome to the world of ~~tomorrow~~!

ESNext

Welcome to the world of today!

ESNext, now



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

ESNext, now

https://github.com/babel/proposals

https://babeljs.io/

ESNext, soonish?

```
<!--
Imagine build tools were smart enough to produce all these
output bundles automatically:
📦  2019.mjs: no transpiled code, no polyfills 😍
📦  2018.mjs: some transpiled code and polyfills 🙂
📦  2017.mjs: slightly more transpiled code + polyfills 😐
📦 legacy.js: megabytes of transpiled polyfill code ☹️
-->
<script type="module"
        srcset="2018.mjs 2018, 2019.mjs 2019"
        src="2017.mjs"></script>
<script nomodule src="legacy.js"></script>
<!--
Imagine browsers were smart enough to download only the
smallest possible bundle based on the `srcset` attribute
value. 🤯
-->
```

https://github.com/whatwg/html/issues/4432

Thanks!

@bramus

3RDS
«Web & Mobile Development»

@3rds

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

Wait a minute!

Didn’t you forget lots of fancy new stuff?

Browser Features / APIs

- IntersectionObserver
- MutationObserver
- Abortable fetch()
- Network API: Quality Indicator
- Intl.PluralRules API
- WebVR
- Generic Sensor API
- Web Share API
- WebUSB API
- Paint Timing API
- Web Payment API
- WebRTC Improvements
- …

(Not included in this slidedeck, as they're no ECMAScript Features)