Polymer Summit

Copenhagen // 2017

SYD

Use The Platform

ES6 Modules
...& Polymer

Sam Thorogood

@samthor

PolymerSYD

ES Modules

ES Modules

JavaScript Modules

ES Modules

JavaScript Modules

ECMAScript Modules

ES Modules

JavaScript Modules

ECMAScript Modules

~~commonJS modules / require()~~

```
<!-- code.js will run as a module -->
<script type="module" src="code.js"></script>

<!-- an inline script tag as as module -->
<script type="module">
    console.info(`It's module time!`);
</script>
```

```
<!-- code.js will run as a module -->
<script type="module" src="code.js"></script>

<!-- an inline script tag as as module -->
<script type="module">
    console.info(`It's module time!`);
</script>
```

```html
<!-- index.html -->
<script type="module" src="code.js"></script>
```

```js
// code.js
import {answer} from './question.js';
console.info(`It's ${answer()} time!`);
```

```js
// question.js
export function answer() { return 'module'; }
```

```html
<!-- index.html -->
<script type="module" src="code.js"></script>
```

```js
// code.js
import {answer} from './question.js';
console.info(`It's ${answer()} time!`);
```

```js
// question.js
export function answer() { return 'module'; }
```

🛠️💕




🏴 ~May 2018

🛠️💕




🏴 ~May 2018

What is a module?

The spec definition_

- JS run with **`<script type="module">`**
- Supported* in all evergreen browsers
- Gives your code *superpowers*—to **`import`** or **`export`**

JS in Modules

Righting some wrongs_

- Always in strict mode
- Can't pollute global scope
- Implicit **`defer`**
- Modules are singletons

PolymerSYD

```
'use strict';
(function() {



/* module code goes here */



}).call(undefined);
```

```
'use strict';
(function() {

with (foo) {}      // syntax error, 'with' considered harmful
q = 123;           // error, variable not declared

var x = 123;               // ok
console.info(window.x);    // will log undefined
console.info(this);        // will log undefined
buttonId.click();          // still works on <button id="buttonId">...

}).call(undefined);
```

Trade restrictions  

Shipping code

Or: how to run an import/export company_

- Exported values can be *variable*

```
export var foo = callMethod();
```

- Imports and exports cannot be *conditional*

```
if (true) { export var bar = ...; }
```

Dynamic imports

Responding to change_

- The top-level **`import`** cannot be dynamic

```
import './' + path + '.js';
```

- **`import()`** to the rescue—*in Chrome/Safari*

```
import(`./${path}.js`).then(...)
```

Relative imports

Qualify yo paths_

- **`import`** must target a relative or absolute path

```
import 'foo/bar.js';
```

- *...more on this later*

The actual syntax

Is actually boring and well documented_

- 🔎 Google it

Takeaways

The relevant CS bits_

- Module spec is *simple*
- Easy to work with (cf. Rollup)
- Static analysis possible

Pre-modules browsers

What's the catch?_

- Older browsers ignore **`type="module"`**
- Use the **`nomodule`** attribute to load transpiled code on legacy browsers—*ignored by modern browsers**

Pre-modules browsers

What's the catch?_

- Older browsers ignore **`type="module"`**
- Use the **`nomodule`** attribute to load transpiled code on legacy browsers—*ignored by modern browsers**

```
<script nomodule src="old.js">
```


```
<!-- for ES module browsers -->
<script src="new-hotness.js" type="module"></script>

<!-- for older browsers -->
<script src="old-built-coldness.js" defer nomodule></script>


<!--
  Caveat: Safari 10.1 (<1.5% global) shipped without nomodule support!
  Fix:    https://tinyurl.com/safari-nomodule
-->
```

The Benefit

Why ship modules at all?_

- ES modules are a 🌊 **high water mark**
- **Every** browser supporting modules supports...

Modules	Chrome*	Edge	Safari	Firefox
Shipped	🏎💨	🏎💨	🏎💨	➡ SOON 60
`<script type="module">`	61	16	10.1	🏁 54
`<script nomodule>`	61	16	11	🏁 55
Other Features...	**Chrome**	**Edge**	**Safari**	**Firefox**
ES6 support (Kangax)	97%	96%	99%	98%
Promises	✓	✓	✓	✓
`window.fetch`	✓	✓	✓	✓
Async/Await	✓	✓	✓	✓
CSS Variables	✓	✓	✓	✓
Custom Elements	✓	Polyfill	✓	🏁
Shadow DOM	✓	Polyfill	✓	🏁

Modules	Chrome*	Edge	Safari	Firefox
Shipped	☁	☁	☁	SOON 60
`<script type="module">`	61	16	10.1	54
`<script nomodule>`	61	16	11	55
Other Features...	**Chrome**	**Edge**	**Safari**	**Firefox**
ES6 support (Kangax)	97%	96%	99%	98%
Promises	✓	✓	✓	✓
`window.fetch`	✓	✓	✓	✓
Async/Await	✓	✓	✓	✓
CSS Variables	✓	✓	✓	✓
Custom Elements	✓	Polyfill	✓	
Shadow DOM	✓	Polyfill	✓	

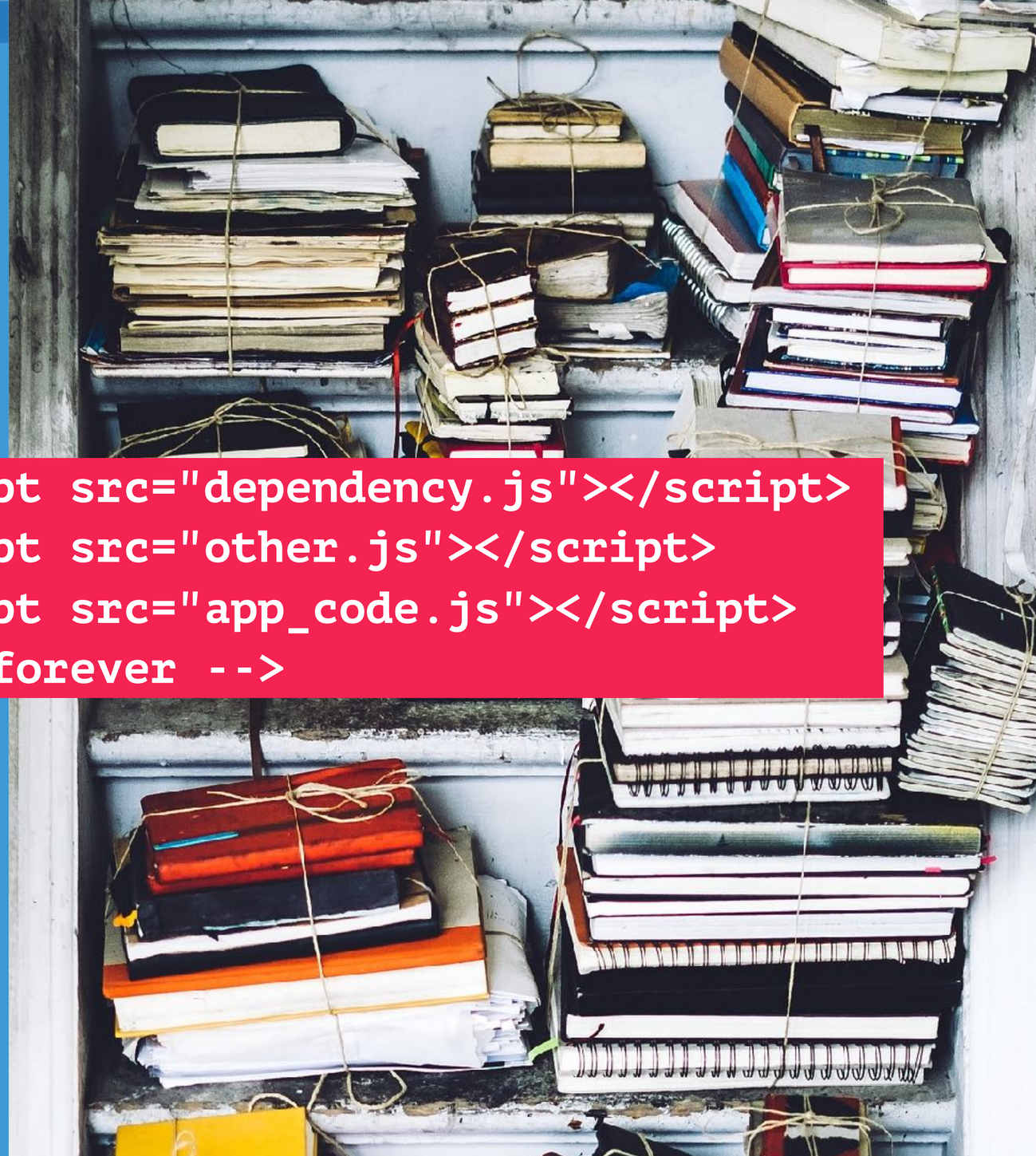
Modules	Chrome*	Edge	Safari	Firefox
Shipped				SOON 60
`<script type="module">`	61	16	10.1	54
`<script nomodul`	61	16	11	55
Other Features…	**Chro**	**Edg**	**ri**	**Firefox**
ES6 support (Kanga	97	96%	%	98%
Promises	✓	✓	✓	✓
`window.fetch`	✓	✓	✓	✓
Async/Await	✓	✓	✓	✓
CSS Variables	✓	✓	✓	✓
Custom Elements	✓	Polyfill	✓	
Shadow DOM	✓	Polyfill	✓	

And the obvious...

Stop doing this_

- Avoid anti-patterns, e.g. loosely-coupled code
- We want to **#UseThePlatform**

```
<script src="dependency.js"></script>
<script src="other.js"></script>
<script src="app_code.js"></script>
<!-- forever -->
```



Build modern +
Compile for legacy

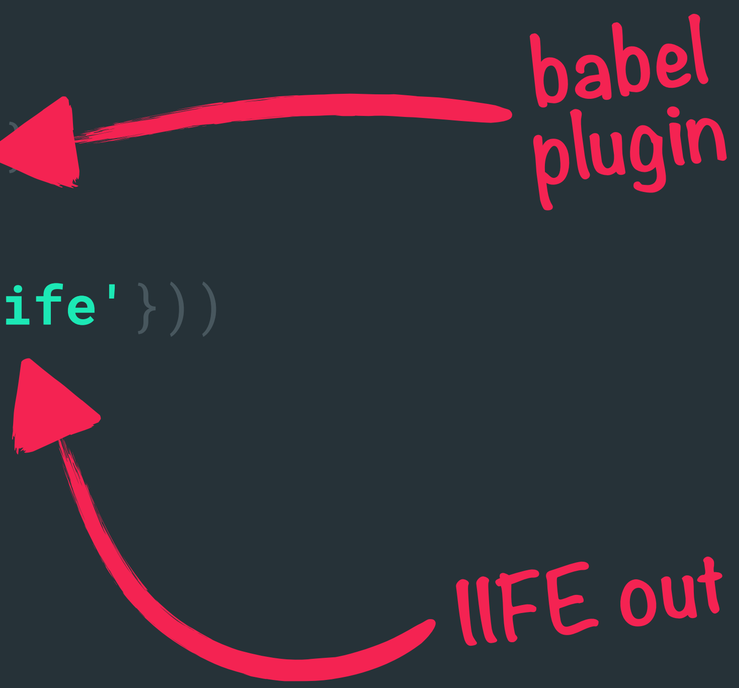
rollup.js

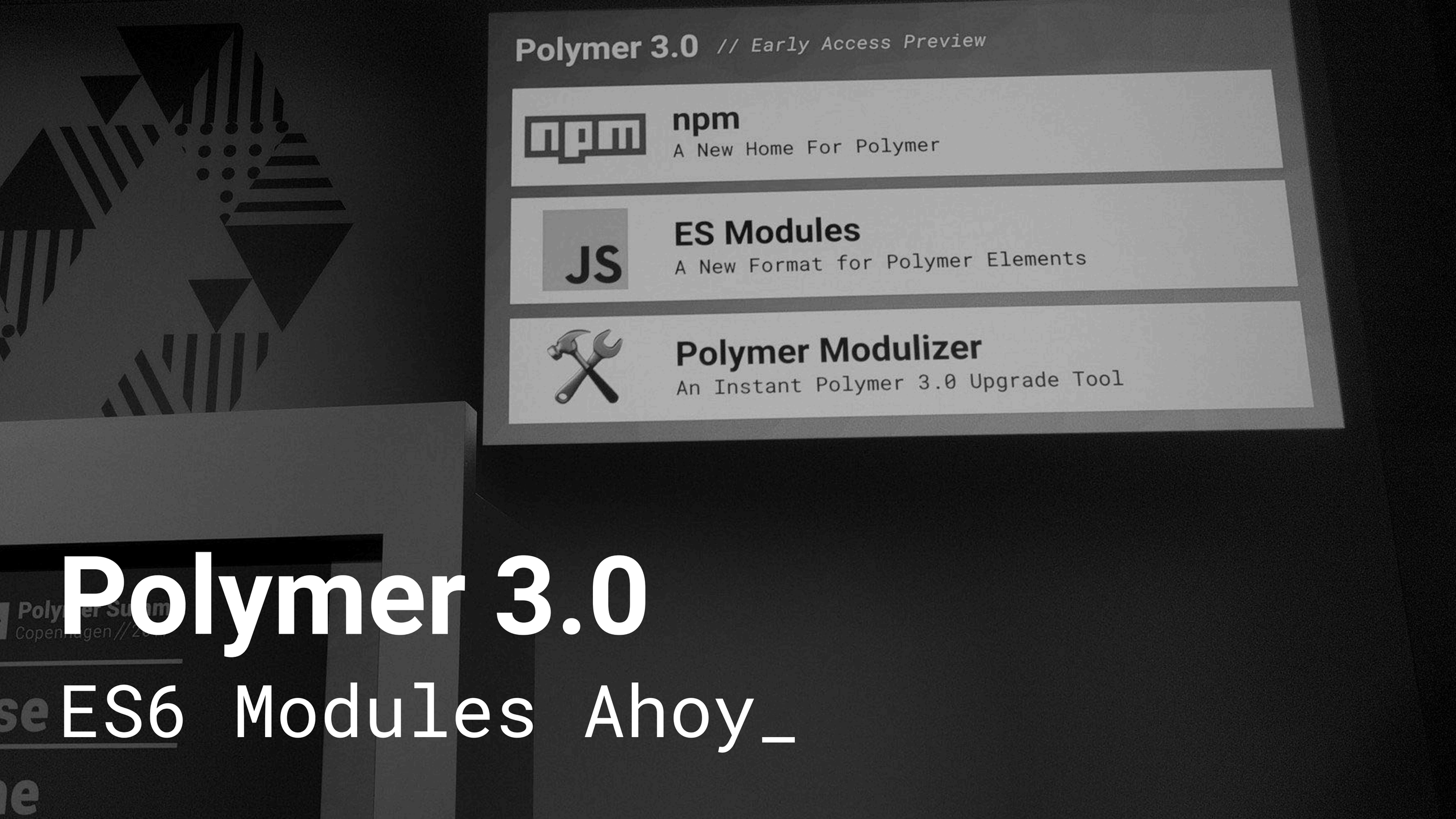
+

BABEL

```
const babel = require('rollup-plugin-babel');

gulp.task('rollup', function() {
  const options = {plugins: [babel()]};
  return gulp.src(['entrypoint.js'])
    .pipe(rollup(options, {format: 'iife'}))
    .pipe(gulp.dest('./dist'));
});
```

```
const babel = require('rollup-plugin-babel');

gulp.task('rollup', function() {
  const options = {plugins: [babel()]};
  return gulp.src(['entrypoint.js'])
    .pipe(rollup(options, {format: 'iife'}))
    .pipe(gulp.dest('./dist'));
});
```





Polymer 3.0

ES6 Modules Ahoy_

Big changes

Just a few_

- ES6 Modules
- Published on NPM—not bower (...but use Yarn?)
- Mechanical rewrite of 2.0—*no material differences*

```
<!-- Polymer 2.0 -->
<dom-module id="paper-icon-button">
  <template strip-whitespace>
    <style>...</style>
    <div><!-- button stuff --></div>
  </template>
  <script>
    Polymer({ /* as expected */ });
  </script>
</dom-module>
```

```
import '../polymer/polymer.js';
import '../iron-icon/iron-icon.js';
import { Polymer } from '../polymer/lib/legacy/polymer-fn.js';
const $_documentContainer = document.createElement('div');
$_documentContainer.setAttribute('style', 'display: none;');

$_documentContainer.innerHTML = `<dom-module id="paper-icon-button">
  <template strip-whitespace="">
    <style>...</style>
    <div><!-- button stuff --></div>
  </template>
</dom-module>`;

Polymer({ /* as expected */ });
```

```
import '../polymer/polymer.js';
import '../iron-icon/iron-icon.js';
import { Polymer } from '../polymer/lib/legacy/polymer-fn.js';
const $_documentContainer = document.createElement('div');
$_documentContainer.setAttribute('style', 'display: none;');

$_documentContainer.innerHTML = `<dom-module id="paper-icon-button">
  <template strip-whitespace="">
    <style>...</style>
    <div><!-- button stuff --></div>
  </template>
</dom-module>`;

Polymer({ /* as expected */ });
```

Not part of 3.0

But interesting anyway_

- **`class`**-based elements
  ...although superclass needs be imported
- **`lit-html`**

```
class MyAppElement extends PolymerElement {
  static get template() {
    return html`<div>I'm a template</div>
                <div>[[withBindings]]</div>
                <button on-click="clickHandler">Click me!</button>`
  }
  ...
}
customElements.define('my-app-element', MyAppElement);
```

```
import {PolymerElement} from '@polymer/polymer/polymer-element.js';

class MyAppElement extends PolymerElement {
  static get template() {
    return html`<div>I'm a template</div>
                <div>[[withBindings]]</div>
                <button on-click="clickHandler">Click me!</button>`
  }
  ...
}
customElements.define('my-app-element', MyAppElement);
```

Deprecated features

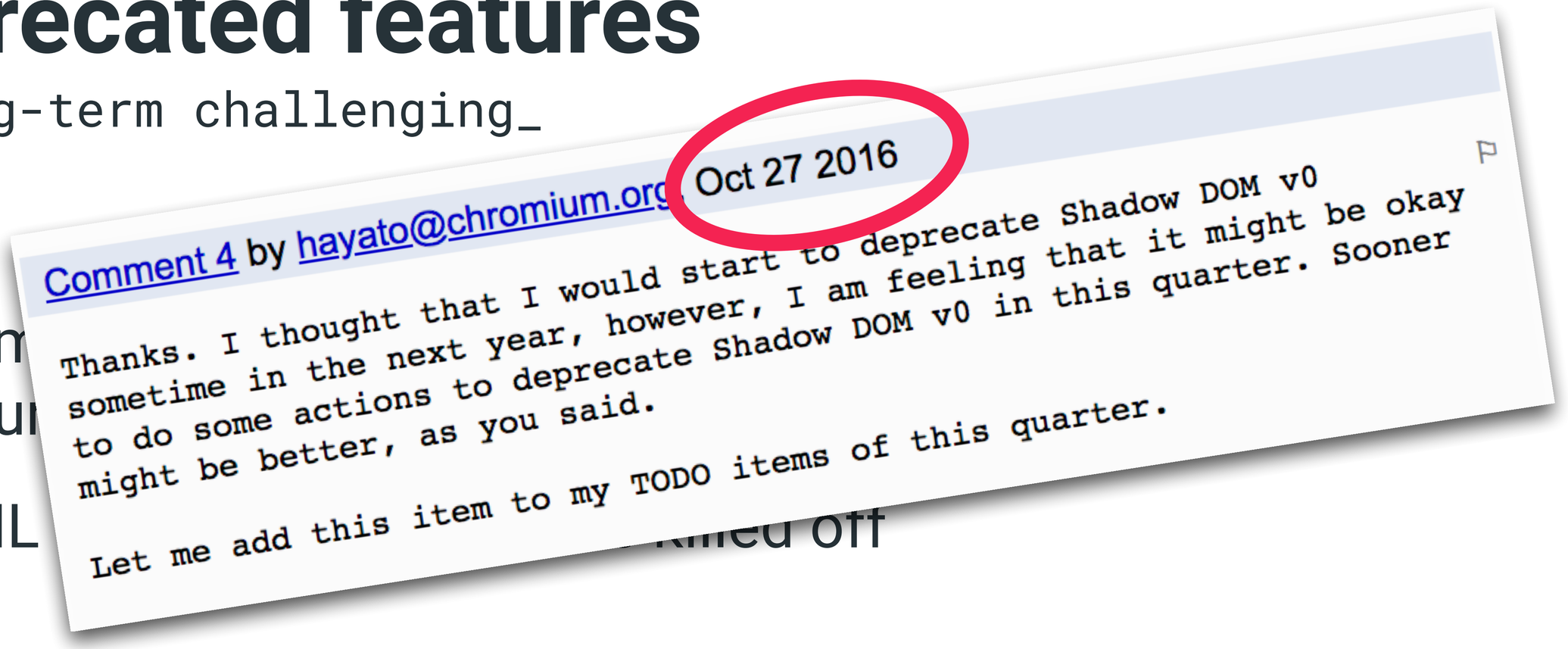
Or long-term challenging_

- Polymer 1.0—**`createShadowRoot`** is going away in favour of **`attachShadow`**
- HTML imports will slowly be killed off

Deprecated features

Or long-term challenging_

- Polym
  favou
- HTML
  illed off



```
⚠ [Deprecation] Styling master document from stylesheets defined in HTML Imports is deprecated, and is planned to
be removed in M65, around March 2018. Please refer to https://goo.gl/EGXzpw for possible migration paths.

⊗ ▶Uncaught DOMException: Failed to execute 'createShadowRoot' on 'Element': Shadow root cannot   polymer.js:10032
be created on a host which already hosts a shadow tree.
    at HTMLElement.shadowFromTemplate (https://web-animations.github.io/web-animations-demos/components/polymer/polymer.js:10032:25)
    at HTMLElement.parseDeclaration (https://web-animations.github.io/web-animations-demos/components/polymer/polymer.js:10004:25)
    at HTMLElement.parseDeclarations (https://web-animations.github.io/web-animations-demos/components/polymer/polymer.js:9985:28)
    at HTMLElement.makeElementReady (https://web-animations.github.io/web-animations-demos/components/polymer/polymer.js:9885:12)
    at HTMLElement.bindFinished (https://web-animations.github.io/web-animations-demos/components/polymer/polymer.js:9656:12)
    at processBindings (https://web-animations.github.io/web-animations-demos/components/polymer/polymer.js:7015:10)
    at cloneAndBindInstance (https://web-animations.github.io/web-animations-demos/components/polymer/polymer.js:7112:5)
    at cloneAndBindInstance (https://web-animations.github.io/web-animations-demos/components/polymer/polymer.js:7099:7)
    at cloneAndBindInstance (https://web-animations.github.io/web-animations-demos/components/polymer/polymer.js:7099:7)
    at HTMLTemplateElement.createInstance (https://web-animations.github.io/web-animations-demos/components/polymer/polymer.js:6763:21)
```


Bare specifiers

Forgot about those?_

- ES6 modules spec doesn't define this behaviour— import from **`'raw_string'`**
- Nor importing non-JS

```
import {thing}
  from 'something';

import {stuff}
  from 'un/qualified.js';

import html
  from './htmlfile.html';
```


Build tools help us

`Undefined behaviour => party time_`

- Rollup and other tools will do the sensible thing—import bare specifiers from **`node_modules`** and switch modes

```
import leftPad from 'left-pad';
```

- *...but this is unlikely to come to browsers anytime soon*

Polymer 3.0 uses bare specifiers

The Pragmatic Choice®_

- As of March 2018, Polymer takes this approach

```
import {PolymerElement}
  from '@polymer/polymer/polymer-element.js';
```

- Polymer's tooling will support this out-of-the-box
  ...but tooling is required

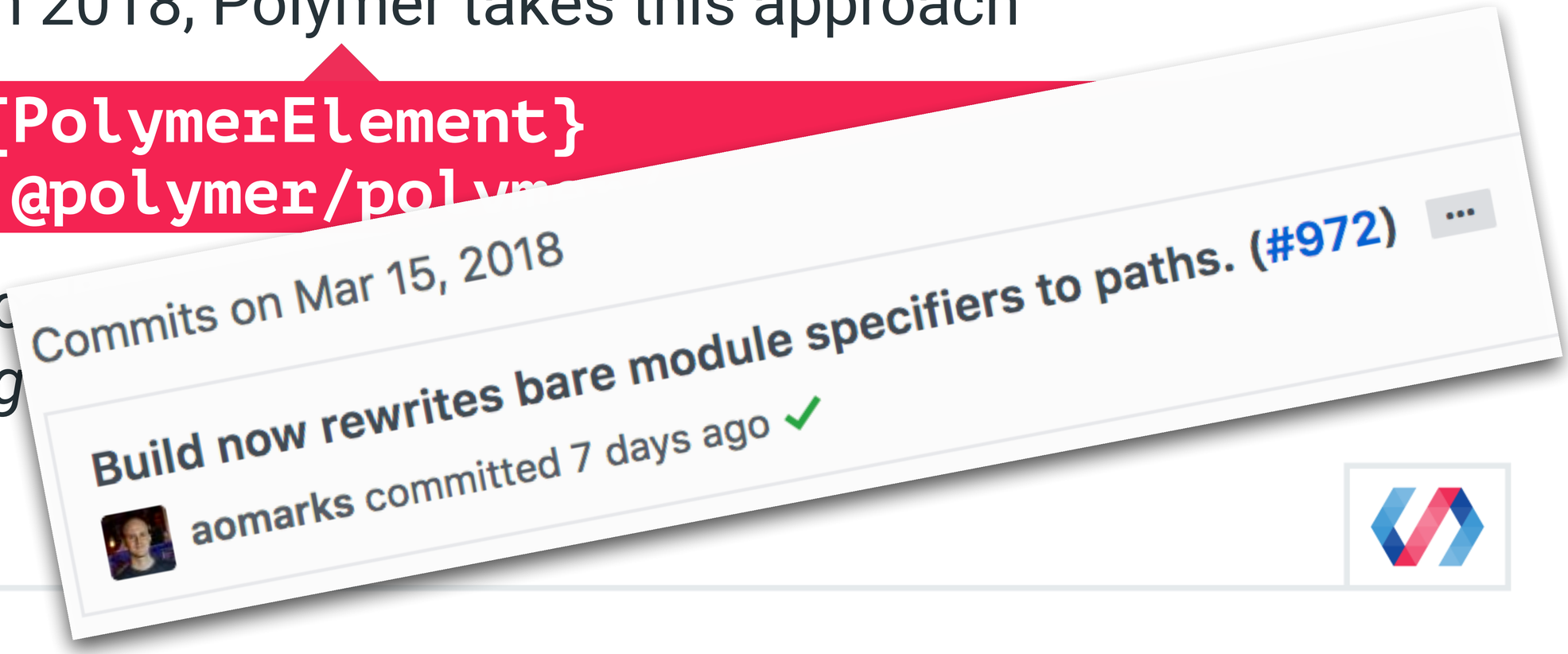
Polymer 3.0 uses bare specifiers

The Pragmatic Choice®_

- As of March 2018, Polymer takes this approach

```
import {PolymerElement}
  from '@polymer/polym
```

- Polymer's to
  ...but tooling



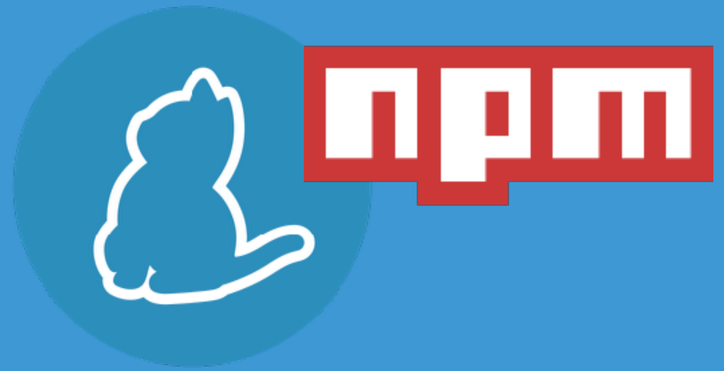
Try it out today

I just did!_

- You can run—
  `yarn add @polymer/polymer@^3.0.0-pre.11`
- But this is hilariously *before* the bare specifiers change

```
import {Element as PolymerElement}
  from "./node_modules/@polymer/polymer/polymer-element.js"

class MyAppElement extends PolymerElement {
  static get template() {
    return html`<div>I'm a template</div>
                <div>[[withBindings]]</div>
                <button on-click="clickHandler">Click me!</button>`
  }
  ...
}
customElements.define('my-app-element', MyAppElement);
```


JS


npm

Further reading

Take advice home with you_

- Jake Archibald

  https://jakearchibald.com/2017/es-modules-in-browsers/

- Sam Thorogood

  https://github.com/samthor/srcgraph
  https://tinyurl.com/samthor-modules

Thank You!

@samthor

Sam Thorogood

@samthor

PolymerSYD

Attribution

Legalese_

- Polymer Summit photo
  https://twitter.com/JorgeCasar/status/899915056535785472
- Other photos from Pixabay
- Emoji by Emojityper

Appendix: build modern … even with modules

Idea: Ship Rolled Up

For simple apps_

- Use Rollup to combine all ES modules into a single file
- **Don't transpile** for modern browsers
  ...they get modern, smaller JS—but with new features!
  ...tree shaking and fewer requests

Idea: Ship Rolled Up

For simple apps_

`rollup -f es`

- Use Rollup to combine all ES modules into a single file
- **Don't transpile** for modern browsers
  ...they get modern, smaller JS—but with new features!
  ...tree shaking and fewer requests

Appendix: Module nuances

Export vs side-effects

`Both options_`

- Modules don't *have* to **`export`** anything
  ... but may have *side-effects*, such as exporting on **`window`**

- Fits traditional polyfills or minified, single-file JS libraries

```
// base64.min.js
window.base64lib = {encode: function() { ... }, decode: ... };
```

```
// base64.min.js
window.base64lib = {encode: function() { ... }, decode: ... };

// main.js
import './node_modules/base64/base64.min.js';
console.info(base64lib.encode('hello!'));
```

Modules are deferred

No page-blocking_

- Modules are implicit **`defer`**
  ...might import lots of deps
  ...script blocking was a mistake™

- You can add **`async`** to run before page load

Modules only import once

Even if popular_

- Modules are singletons and are only run once
  ...no matter how they are imported—directly or via **`import`**

Modules only import once

Even if popular_

- Modules are singletons and are only run once
  ...no matter how they are imported—directly or via **`import`**

```
<script src="a.js" type="module"></script>
<script type="module">
  import './a.js';
</script>
```


Hoisted imports

Going all the way up_

- Imports are hoisted
  ...like a top-level **`function`**
- Executed in graph order

```
// call method at top
excitingMethod();

// import can happen later
import {excitingMethod}
    from './excitingFile.js';
```


Hoisted imports

Going all the way up_

- Imports are hoisted
  ...like a top-level **`function`**
- Executed in graph order

```
// effectively at top of file
import {excitingMethod}
    from './excitingFile.js';

// call method at top
excitingMethod();

// import can happen later
import {excitingMethod}
    from './excitingFile.js';
```

Modules, hoisted

Going all the way up_

- Imports are hoisted
  ...like a top-level **`function`**
- Executed in graph order


```
// new emoji polyfill
window.emojiPolyfill = {...};

// we need it for emojineer
import './node_modules/🤔/emoji.js';
```

Modules, hoisted

Going all the way up_

- Imports are hoisted
  ...like a top-level **`function`**
- Executed in graph order


```
// include as an import
import './emojiPolyfill.js';

// we need it for emojineer
import './node_modules/🤔/emoji.js';
```

Circular dependencies are allowed*

*with some caveats_

- Modules can import each other safely
  ...but some exports aren't available until completion

- Rule of thumb: don't do any work in top-level of module

```
// vehicle.js


let id = 0;
export class Vehicle {
  constructor() {
    this.id = ++id;
  }
}
```

```
// vehicle.js


let id = 0;
export class Vehicle {
  constructor() {
    this.id = ++id;
  }
}
```

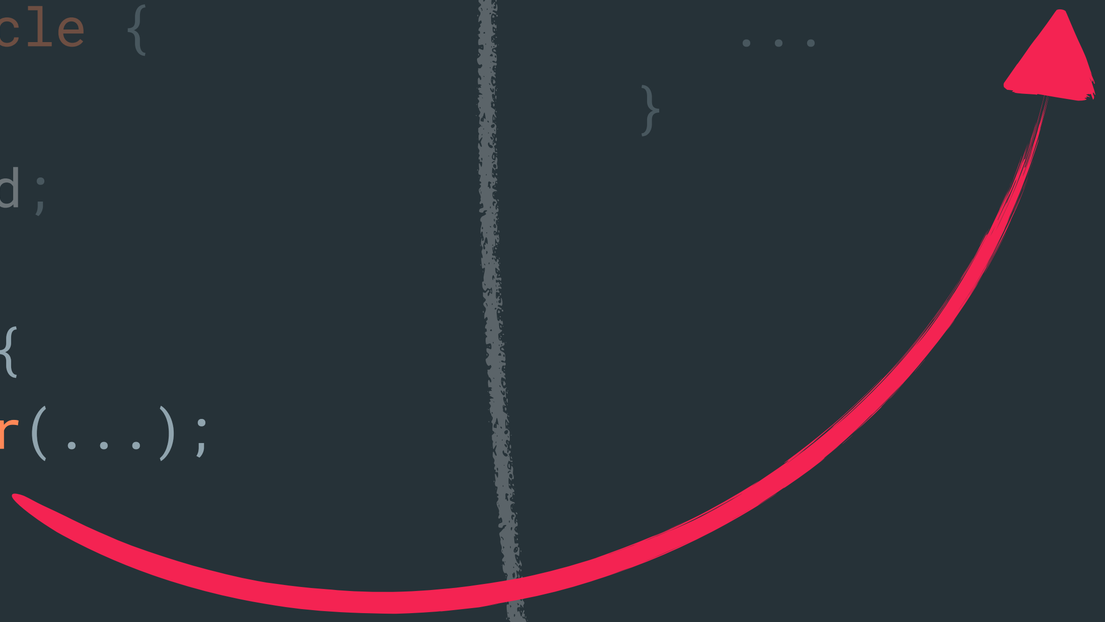
```
// car.js
import {Vehicle} from './vehicle.js';

export class Car extends Vehicle {
    ...
}
```

```
// vehicle.js
import {Car} from './car.js';

let id = 0;
export class Vehicle {
  constructor() {
    this.id = ++id;
  }
  static build() {
    return new Car(...);
  }
}
```

```
// car.js
import {Vehicle} from './vehicle.js';

export class Car extends Vehicle {
    ...
}
```

```
// vehicle.js
import {Car} from './car.js';

let id = 0;
export class Vehicle {
  constructor() {
    this.id = ++id;
  }
  static build() {
    return new Car(...);
  }
}
```

```
// car.js
import {Vehicle} from './vehicle.js';

export class Car extends Vehicle {
    ...
}
```



```
// vehicle.js
import {Car} from './car.js';

let id = 0;
export class Vehicle {
  constructor() {
    this.id = ++id;
  }
  static build() {
    return new Car(...);
  }
}
```

```
// car.js
import {Vehicle} from './vehicle.js';

export class Car extends Vehicle {
    ...
}
```

```
// vehicle.js
import {Car} from './car.js';

let id = 0;
export class Vehicle {
  constructor() {
    this.id = ++id;
  }
  static build() {
    return new Car(...);
  }
}
```

```
// car.js
import {Vehicle} from './vehicle.js';

export class Car extends Vehicle {
    ...
}

let testCar = new Car();
```

```
export class Vehicle {
  // id not in scope
}

// vehicle.js
import {Car} from './car.js';

let id = 0;
// Vehicle gets hoisted
```

```
// car.js
import {Vehicle} from './vehicle.js';

export class Car extends Vehicle {
    ...
}

let testCar = new Car();
```

```
export class Vehicle {
  // id not in scope
}

// vehicle.js
import {Car} from './car.js';

let id = 0;
// Vehicle gets hoisted
```

```
// car.js
import {Vehicle} from './vehicle.js';

export class Car extends Vehicle {
    ...
}

let testCar = new Car();
```