DAVE HERMAN

"USE MODULE";

MY GOALS

- Present proposal again
- Report on Node.js discussions
- Stage 2
- Discuss open issues

MODULE? SCRIPT? CJS?

```
var window = 'ohhh myyyyy';

console.log(typeof __dirname);

function foo() {
  console.log(this);
}
foo();
```

Is this file a *Script*?

Is this file a CJS module?

Is this file strict?

MOTIVATION #1: NODE ENTRY POINTS

- Application entry points

```
$ node my-app.js
```

- Package binaries

```
{ "bin": { "my-app": "./bin/my-app.js" } }
```

- Requiring submodules

```
var array = require("lodash/array.js");
```

MOTIVATION #2: PORTABLE/DEFENSIVE MODULES

```
<script src="jquery.mjs"></script>
```

PROPOSAL: OPTIONAL PRAGMA

```
"use module";

var window = 'ohhh myyyyy';

console.log(typeof __dirname);

function foo() {
  console.log(this);
}
foo();
```

SEMANTICS

- Early error if pragma doesn't match goal symbol
- Browser determines mode from attributes, not pragma
- But host environments *can* use pragma to disambiguate
- Without pragma, Node.js would of course default to CJS

ALTERNATIVE: .MJS

- Controversial
- Fragmentation: other ESM ecosystems use `.js`
- Upgrading is generally semver-incompatible
- Doesn't solve portable/defensive use case
- **But** not incompatible with pragmas

ALTERNATIVE: EMPTY EXPORT

```
export {};

var window = 'ohhh myyyyy';

console.log(typeof __dirname);

function foo() {
  console.log(this);
}
foo();
```

ALTERNATIVE: EMPTY EXPORT

- Obscure incantation: doesn't say what it means
- Easier to forget, e.g. when refactoring
- Pragma has to appear at the top

OPEN ISSUE: "USE SCRIPT"? "USE NOMODULE"?

- Do we need pragmas for other modes?
- `"use script";` at top-level doesn't work for CJS
- `"use nomodule";` at top-level and function body?

STAGE 2

- Spec draft: https://github.com/proposal-modules-pragma/
- Includes `"use script";` but easily modified/removed
- Includes nested pragmas but easily removed