# Decorator/export Ordering (continued)

Daniel Rosenwasser, Ron Buckton


REC

# Topic Recap: Ordering motivated by `toString` on the original class

- `Function.prototype.toString` is fragile.
  - Understood issues with unresolved bindings.
  - Plain `eval` doesn't work for `await` or `yield` in computed properties.
- Decorators not included on a method's `toString`, but are on classes.
  - *Kind of a weird current state actually.*
- Information from `toString` might be better-served by the decorator context object.

# Statement with Class Expression

vs.

# Class Declaration

- A class declaration really is its own thing.
  - And the `export` keyword is part of the declaration.
- The `return` statement example feels "off"
  - Class expressions exist under a different syntactic space
  - Otherwise, we'd just call them class declarations.
  - That's why `export default` doesn't get followed with a class *expression*.

# Future Space for

- A class declaration really is its own thing.
  - And the `export` keyword is part of the declaration.
- The `return` statement example feels "off"
  - Class expressions exist under a different syntactic space.
  - That's why they're not just class declarations.
  - That's why export default doesn't parse our a class expression.

We would like to make one of the following changes

- Option 1: Decorators are placed **before** the `export` keyword.
  - *Our preference*
- Option 2: Decorators can be placed **before or after** the `export` and `export default` keyword.
  - *Preference for exclusive-or*
- Option 3: Status quo – decorators continue to come after `export` and `export default`