SYMBOLS: OBJECT OR PRIMITIVE?

Dave Herman
September 18, 2013
DESIDERATA
1. STATELESSNESS

- Sharing a symbol should not share state.
- Encapsulates a property key and nothing else.
2. CROSS-FRAME COMPAT

```javascript
obj[iterator] = function*() { ... };
let w = new Window(...);
w.shared = obj;
```
3. METHODS

alert.call()

Math.sin(0)

document.getElementById("body")

(1.2).toFixed()

"tc39".toUpperCase()

true.toString()
4. MUTABLE PROTOTYPES

Yes, monkey-patching *in general* is bad.

But monkey-patching *standard* methods is a **best practice**.

The evolution of the Web platform depends on it.
NON-ANSWERS
SHALLOW-FROZEN OBJECTS

0.gP0(iterator).foo = 12;

Fails Desideratum #1: stateful

Fails Desideratum #2: distinct xframe iterators
DEEP-FROZEN OBJECTS

0.gPO(iterator).foo = 12 // strict error

Fails Desideratum #4: no evolution
PROTOTYPE-FREE OBJECTS

0.gPO(iterator) === null

Fails Desideratum #3: no methods
NON-WRAPPING PRIMITIVES

iterator.valueOf() // error

Fails Desideratum #3: no methods
CONCLUSION:
• JS already has an answer for this!

• `typeof iterator === "symbol"`

• Get/call operations auto-wrap

• Prototype state is global per-frame

• Sending across frames doesn’t share state
I know people think auto-wrapping is gross.

Here’s my positive spin:

- Provides a uniform OO surface for all values.
- Does so without ruining value immutability.
- Does so without ruining API patchability.

Going forward: we need a solution for value types.
REMAINING ISSUES
FOOTGUNS?

[[ToPropertyKey]] of Symbol objects: auto-unwrap? Does it really matter in practice?

Worry about toString for symbols and Symbol objects? Again, does it matter in practice?
Do we know it won’t break the Web?

MSIE "unknown" type may simply be rare enough to be undiscovered.

**Fallback:** "object" with [[Get]] et al that behave like auto-wrappers? (plus `Object.isValue()`?)
18 SEPT 13 TC39 RESOLUTIONS

• Yes to primitives with auto-wrapping

• No auto-unwrapping of Symbol objects

• `typeof iterator === "symbol"`

• `Symbol.prototype.toString` should throw to help catch bugs in code evolution; `Object.prototype.toString` usable for infallible string coercion

• `Symbol()` creates primitive, new `Symbol` throws