Non-extensible Applies to Private

for stage 1 (or 2?, or 2.7?)

Mark S. Miller — Agoric

Shu-yu Guo — Google

Chip Morningstar — METAMASK

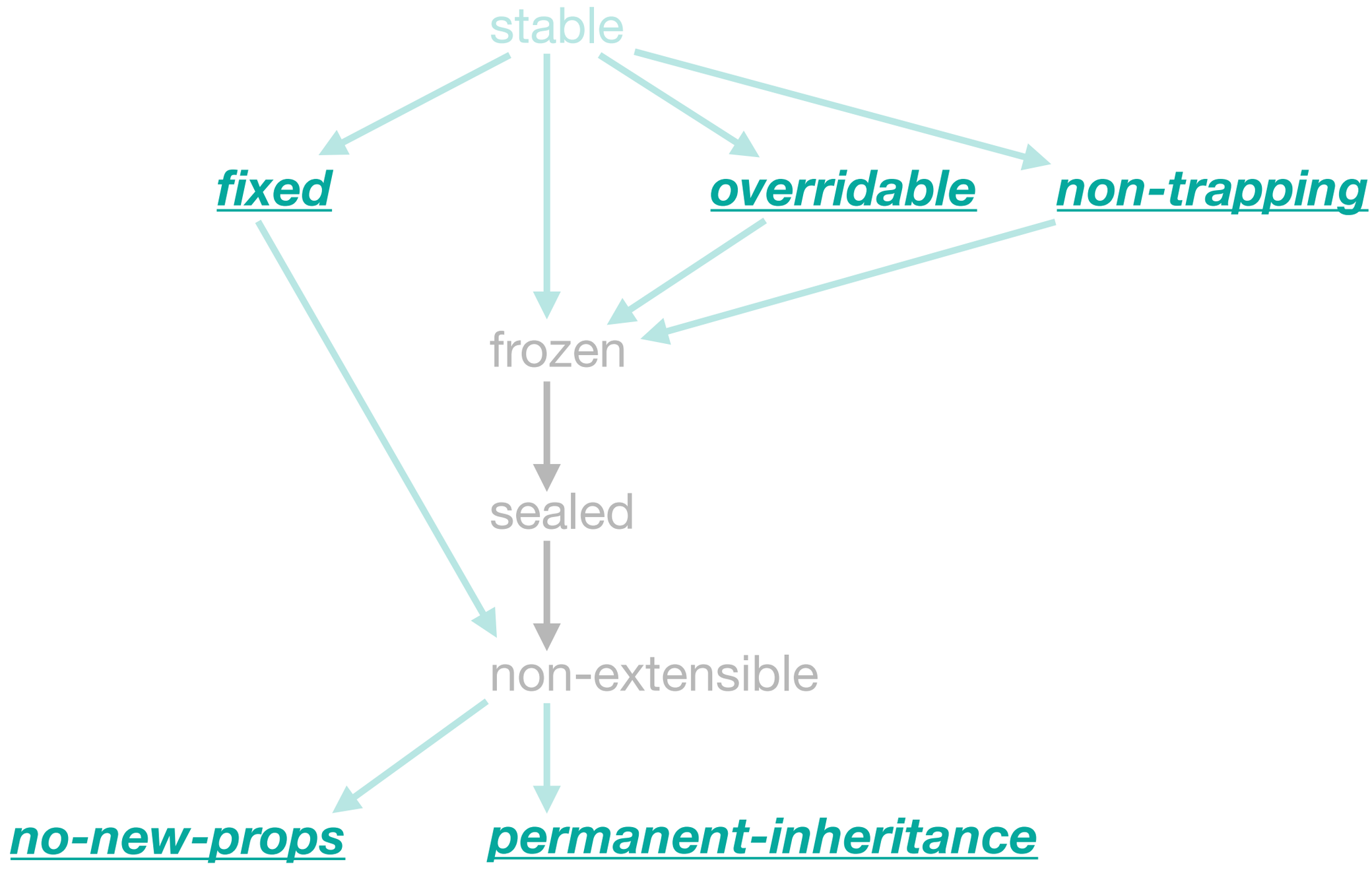
Erik Marks — METAMASK

107th Plenary

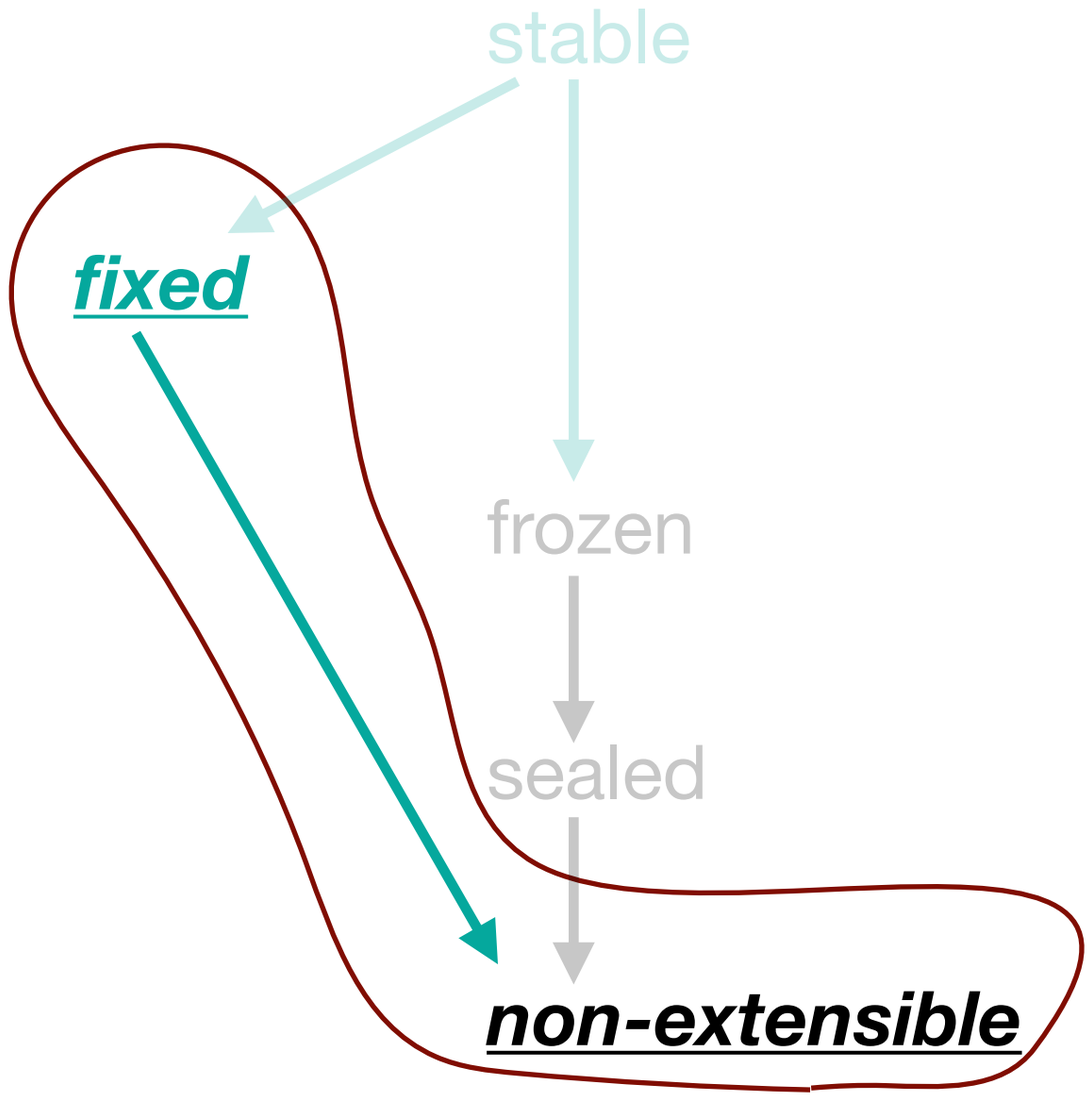
April 2025

TC 39

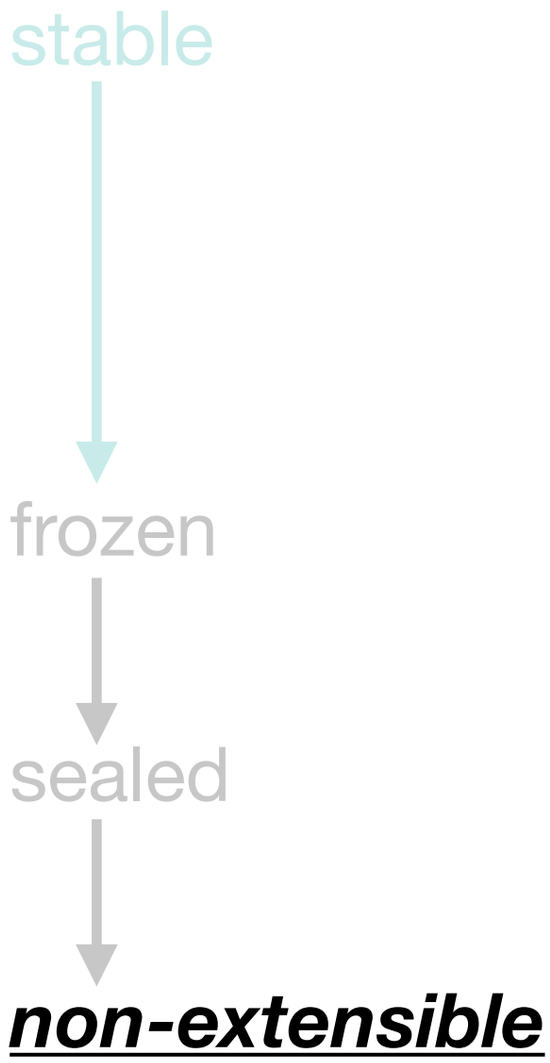
Recap: Stabilize Unbundled Integrity Traits



Recap: Hopes and Dreams



Recap: Hopes and Dreams



Counter-intuitive

```
class FrozenBase {
  constructor() {
    Object.freeze(this);
  }
}
```

```
class AddsProperty extends FrozenBase {
  _val;

  constructor(v) {
    super();
    this._val = v;
  }
}

// throws TypeError
new AddsProperty(42);
```

```
class AddsPrivateField extends FrozenBase {
  #val;

  constructor(v) {
    super();
    this.#val = v;
  }
}

// doesn't throw
new AddsPrivateField(42);
```

Arguably, Least Surprise

```
class FrozenBase {
  constructor() {
    Object.freeze(this);
  }
}
```

```
class AddsProperty extends FrozenBase {
  _val;

  constructor(v) {
    super();
    this._val = v;
  }
}

// throws TypeError
new AddsProperty(42);
```

```
class AddsPrivateField extends FrozenBase {
  #val;

  constructor(v) {
    super();
    this.#val = v;
  }
}

// throws TypeError
new AddsPrivateField(42);
```

Recap: Override Mistake —> structs change shape

```
class Trojan {
  constructor(key) { return key; }
}

class Tagger extends Trojan {
  #value;

  constructor(key, value) {
    super(key);
    this.#value = value;
  }
}

// struct instances born sealed
new Tagger(struct, 'a'); // adds #value to struct anyway
```

Recap: Mitigate Override Mistake —> structs fixed shape

```
class Trojan {
  constructor(key) { return key; }
}

class Tagger extends Trojan {
  #value;

  constructor(key, value) {
    super(key);
    this.#value = value;
  }
}

// struct instances born sealed
new Tagger(struct, 'a'); // throws TypeError
```

WeakMap reachable by syntax alone

```
class Trojan {
  constructor(key) {
    return key;
  }
}

const makeHiddenWeakMap = () => {
  const tombstone = Symbol('deleted');
  class PrivateTagger extends Trojan {
    #value;

    constructor(key, value) {
      super(key);
      this.#value = value;
    }

    static HiddenWeakMap = class {…}
  };
  return PrivateTagger.HiddenWeakMap;
}
```

```
const makeHiddenWeakMap = () => {
  const tombstone = Symbol('deleted');
  class PrivateTagger extends Trojan {
    …
    static HiddenWeakMap = class {
      set(key, value) { new PrivateTagger(key, value); }

      has(key) {
        try {
          return key.#value !== tombstone;
        } catch { return false; }
      }
      get(key) {
        try {
          const value = key.#value;
          return value === tombstone ? undefined : value;
        } catch { return undefined; }
      }
      delete(key) {
        if (this.has(key)) {
          key.#value = tombstone;
          return true;
        }
        return false;
      }
    }
  };
  return PrivateTagger.HiddenWeakMap;
}
```

1.1 PrivateFieldAdd (*O*, *P*, *value*)

The abstract operation PrivateFieldAdd takes arguments *O* (an Object), *P* (a Private Name), and *value* (an ECMAScript language value) and returns either a normal completion containing UNUSED or a throw completion. It performs the following steps when called:

1. If *O*.[[Extensible]] is **false**, throw a **TypeError** exception.
2. If the host is a web browser, then
    a. Perform ? HostEnsureCanAddPrivateElement(*O*).
3. Let *entry* be PrivateElementFind(*O*, *P*).
4. If *entry* is not EMPTY, throw a **TypeError** exception.
5. Append PrivateElement { [[Key]]: *P*, [[Kind]]: FIELD, [[Value]]: *value* } to *O*.[[PrivateElements]].
6. Return UNUSED.

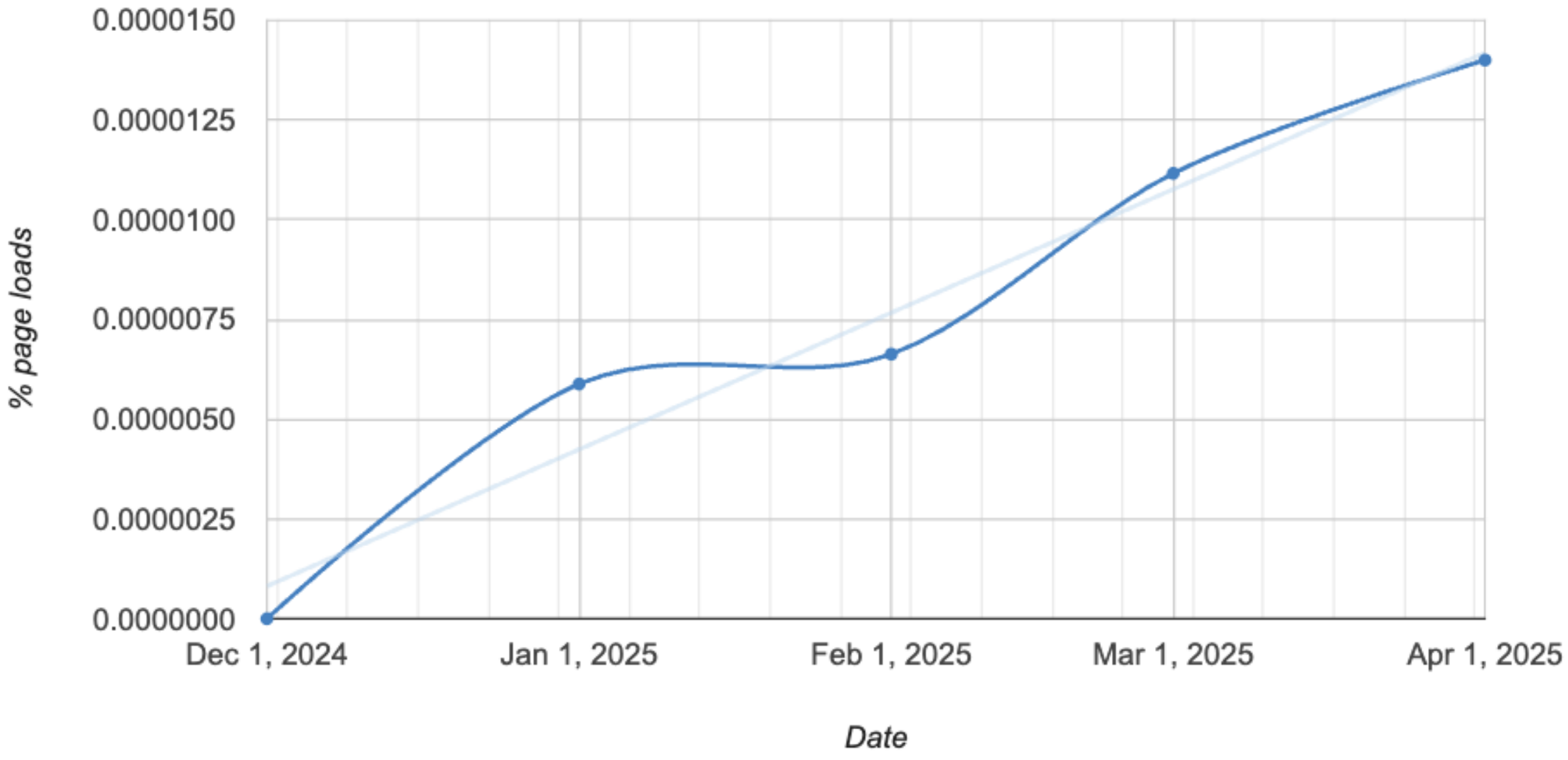
1.2 PrivateMethodOrAccessorAdd (*O*, *method*)

The abstract operation PrivateMethodOrAccessorAdd takes arguments *O* (an Object) and *method* (a PrivateElement) and returns either a normal completion containing UNUSED or a throw completion. It performs the following steps when called:

1. Assert: *method*.[[Kind]] is either METHOD or ACCESSOR.
2. If *O*.[[Extensible]] is **false**, throw a **TypeError** exception.
3. If the host is a web browser, then
    a. Perform ? HostEnsureCanAddPrivateElement(*O*).
4. Let *entry* be PrivateElementFind(*O*, *method*.[[Key]]).
5. If *entry* is not EMPTY, throw a **TypeError** exception.
6. Append *method* to *O*.[[PrivateElements]].
7. Return UNUSED.

Percentage of page loads over time

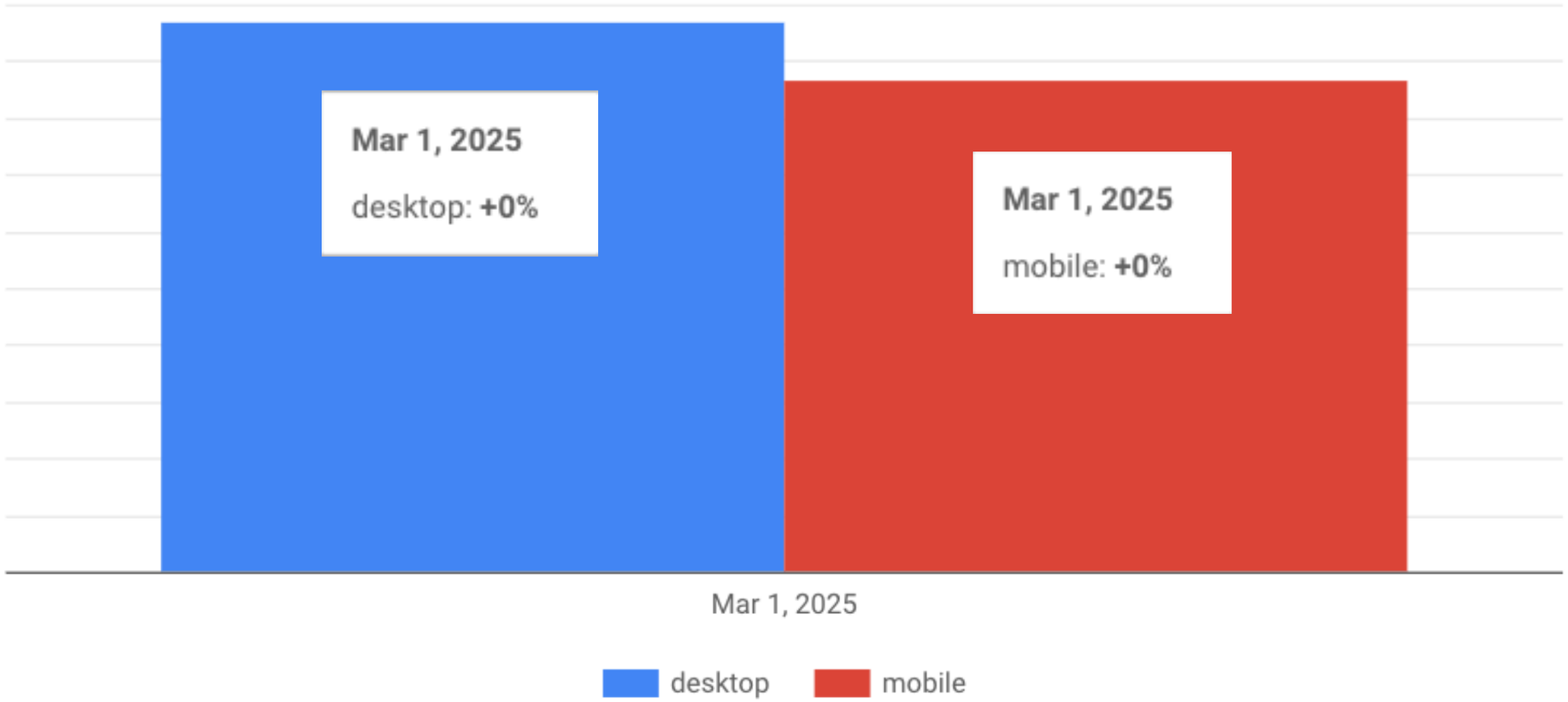
The chart below shows the percentage of page loads (in Chrome) that use this feature at least once. Data is across all channels and platforms. Newly added use counters that are not on Chrome stable yet only have data from the Chrome channels they're on.



Adoption of the feature on top sites

The chart below shows the adoption of the feature by the top URLs on the internet. Data from HTTP Archive.

Select date range | client | feature: V8ExtendingNonExtensibl... (1)



	Sample URLs of the most popular sites using this feature ordered alphabetically ▲
1.	https://gis.bvl.bund.de/
2.	https://heidebeat.de/
3.	https://tsis.fli.de/
4.	https://udo.lubw.baden-wuerttemberg.de/
5.	https://umweltdaten.lubw.baden-wuerttemberg.de/
6.	https://www.rt-strafverteidiger.de/

Web compat analysis #1

Open

syg opened 4 days ago

Chrome use counter data show there are indeed in-the-wild usage of extending non-extensible objects with private fields.

The percentage of page load is very low at time of this writing (0.000011%), and are found to be concentrated in two pieces of software.

disy Cadenza

Most breakages come from Cadenza, which seems to be a closed-source German GIS software. The pattern used is freezing a class with nothing but static fields using the RHS of a private field initializer. The minified version looks like the following.

```
class _ {
  static FOO = ...;
  static BAR = ...;
  static #t = void (Object.keys(_).forEach((t) => {
          _[t].type = t;
        }), Object.freeze(_));
}
```

The percentage of page load is very low at time of this writing (0.000011%), and are found to be concentrated in two pieces of software.

"Axial"

The remaining breakages (https://heidebeat.de/ and https://www.rt-strafverteidiger.de/) seem to be a UI framework. Both sites have identically structured source, with very similar JS, all referencing code with "Axial" in variable names. The pattern used is freezing `this` in a superclass constructor, but have many subclasses with private fields.

```
class t extends EventTarget {
  constructor() {
    super();
    Object.freeze(this);
  }
}

class h extends t {
  #foo;
  #bar;
  constructor() {
    super();
    this.#foo = ...;
  }
}
```

I cannot find references to this framework on GitHub or the web. If anyone knows how to contact the developers of this framework, please let me know.

Questions? Stage 1?, 2?, 2.7?

Stage 2.7

- [ ] committee approval
- [ ] spec editor signoff (@tc39/ecma262-editors)
- [ ] spec reviewer signoff
- [x] resolve all normative issues
  - None yet
- [ ] receive implementer feedback
  - [?] v8

Stage 2

- [ ] committee approval
- [ ] spec reviewers selected
- [x] spec text written

Stage 1

- [ ] committee approval