Record & Tuple

Unified equality, an important invariant.

Unified equality, an important invariant.

Concern: Records and Tuples need to be compared, but by which semantics should they be compared? **Identity? Contents? Does the language decide this, or does the developer?**

Question: Should Record & Tuple comparisons be implemented with a function/method or the === operator?

“Unified equality” is an important invariant to uphold for records/tuples.

Unified equality, an important invariant.

What does “unified equality” mean?

- One view of equality
- Common operations share semantics
- “Hiding” value identity important for coherence.

Invariant: one view of equality.

ECMA262 provides a single, coherent model of equality for every value.

- Objects only define equality by “identity”
- Primitives only define equality by “contents”

Unlike Java/C#/etc, JavaScript does not describe other methods of determining “equality”; no operator overloading for ===.

Developers have come to rely on the “reliability” of equality.

Invariant: common ops share equality semantics.

For example: strings are compared via the same equality semantics in all circumstances.

```
const a = "foo bar";
const b = "foo bar";
a === b; // true

const set = new Set();
set.add(a);
set.has(b); // true
```

There is no accidental way to mess this up. This is intentional.

Invariant: “Hiding” value identity important for coherence.

Developers are often confused by accidental identity comparisons in other languages: https://stackoverflow.com/questions/513832/how-do-i-compare-strings-in-java/513839#513839

- Objects are “owned” by the things that point to them.
- Nothing “owns” a String, it’s existence is self-describing.

These two models should remain distinct, each having different uses.