

Record & Tuple

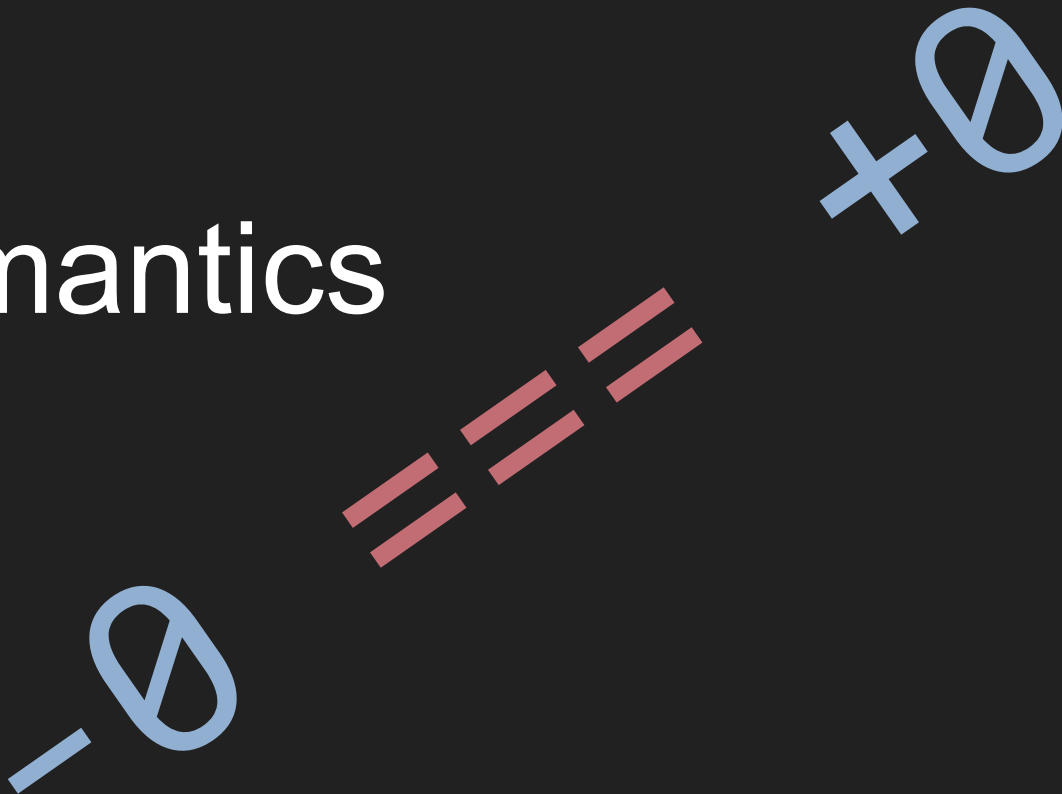
for Stage 2

Robin Ricard & Rick Button
Bloomberg

*Advisor: Daniel Ehrenberg
Igalia*

A recap of last update

Equality Semantics



Going with intermediary semantics for `==/===`:

- The one used for Map keys/Set values comparison.
- An unification of `+0` and `-0`.

`Object.is` compares to see if they are identical:
In that case `+0` and `-0` are different.

```
const s = new Set();  
s.add(#[+0]);  
s.has(#[ -0]) === true;  
s.add(#[NaN]);  
s.has(#[NaN]) === true;
```

```
#[ -0] === #[+0]  
#[NaN] === #[NaN]
```

```
#[ -0] == #[+0]  
#[NaN] == #[NaN]
```

```
Object.is(#[ -0], #[+0]) === false  
Object.is(#[NaN], #[NaN]) === true
```

Avoids “black-holing”
structures if a NaN appears in
any of them.

```
const measure = 42;
```

```
const computed = #{  
  name: "Computed Measurement",  
  value: pureComputeValue(measure),  
};
```

```
assert(computed === computed);  
// What if pureComputeValue returns NaN?
```

Avoids failing comparisons
when the structure potentially
has a -0 in it.

```
function isAtOrigin(c) {  
    return c === #{x: 0, y: 0};  
}
```

```
const coord = #{x: 0, y: 3};  
const coord2 = #{  
    x: coord.x * -4,  
    y: coord.y - 3,  
};
```

```
assert(isAtOrigin(coord));  
// We expect this one to be true!
```

In general, we're trying to make comparing records and tuples “trustworthy” for users and avoiding those subtle equality breakages helps in establishing this.

Still open for discussion!

- This is the equality we have in the Stage 2 spec
- This can change before we get to Stage 3
- The right decision will appear through more research:
 - Experimental implementations
 - Interviewing and surveying developers
 - Performance implications in implementations

State of the proposal

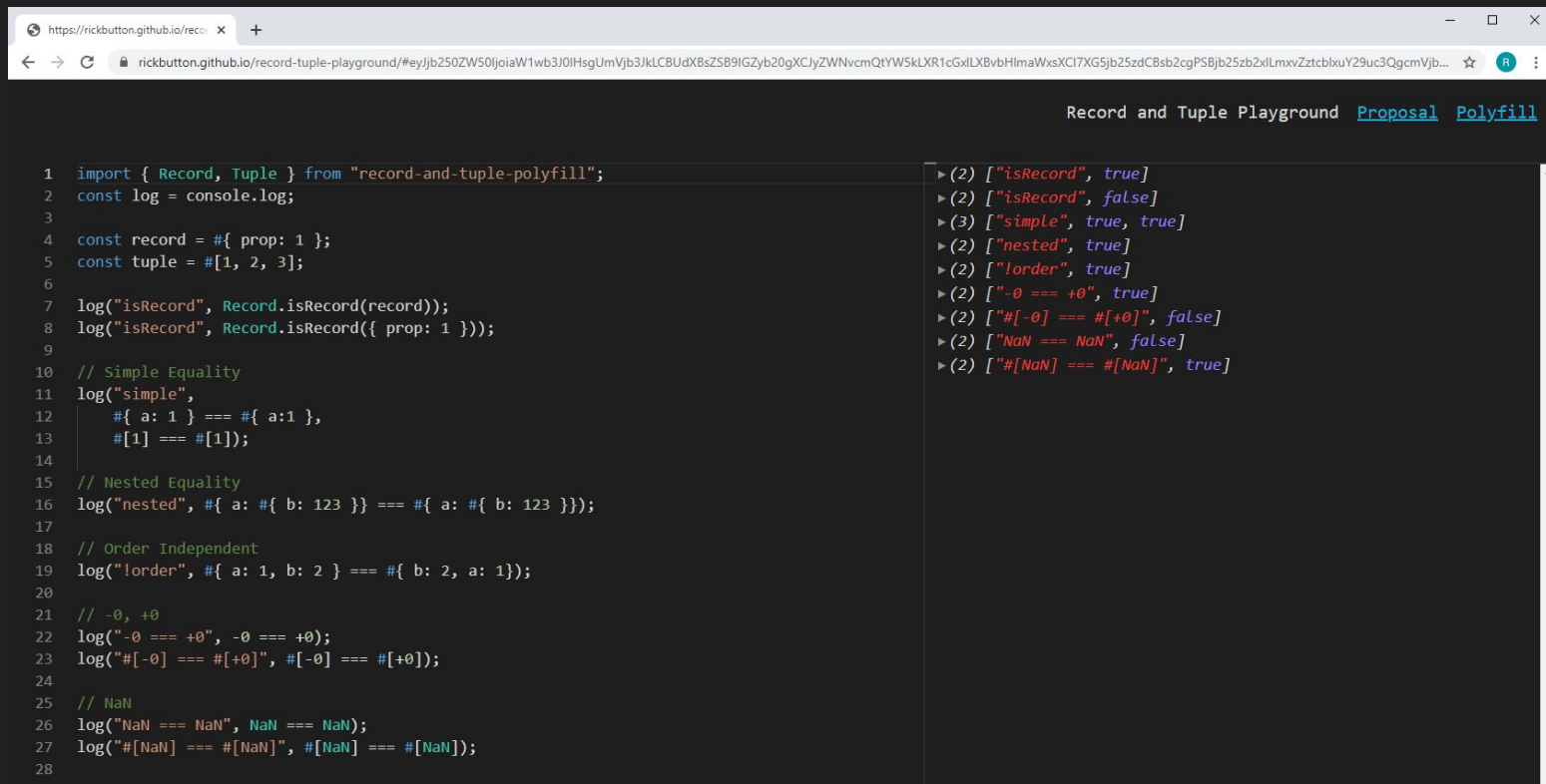
All decisions open until Stage 3

- Definitive equality semantics ([#65](#))
- Exact ToString behavior ([#136](#))
- Names and exact semantics of Tuple.prototype methods (e.g. pushed) ([#121](#))
- Spec-internal Record naming conflict in ECMA262 & WebIDL ([#96](#) & [#116](#))
- Syntax still open with a possibility to move to { | } and [|] ([#10](#))
- Should the wrapper objects be extensible ([#137](#))
- Should Record have a null prototype? ([#71](#))

Record and Tuple Toy Implementation & Playground

<https://github.com/bloomberg/record-tuple-polyfill>

<https://rickbutton.github.io/record-tuple-playground/>



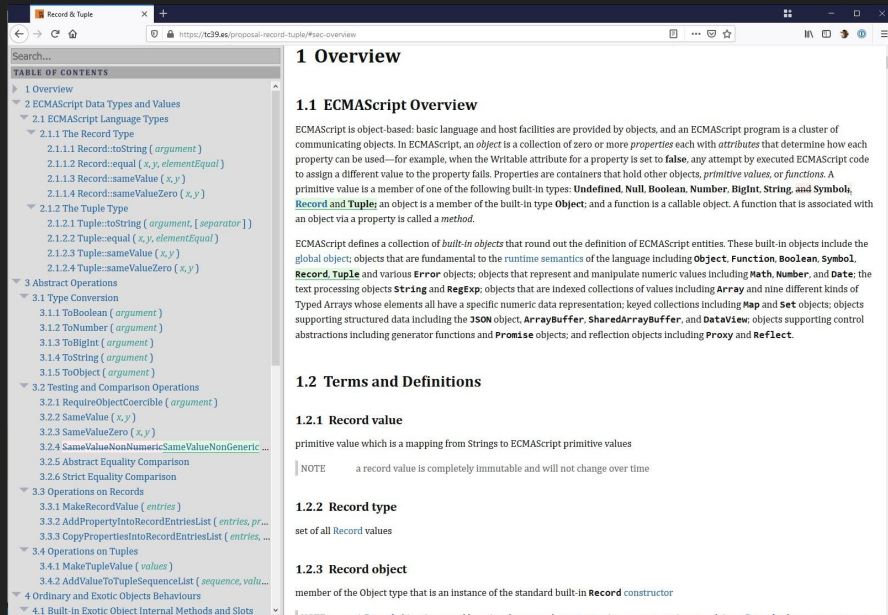
```
1 import { Record, Tuple } from "record-and-tuple-polyfill";
2 const log = console.log;
3
4 const record = #{ prop: 1 };
5 const tuple = #[1, 2, 3];
6
7 log("isRecord", Record.isRecord(record));
8 log("isRecord", Record.isRecord({ prop: 1 }));
9
10 // Simple Equality
11 log("simple",
12     #{ a: 1 } === #{ a: 1 },
13     #[1] === #[1]);
14
15 // Nested Equality
16 log("nested", #{ a: #{ b: 123 } } === #{ a: #{ b: 123 } });
17
18 // Order Independent
19 log("lorder", #{ a: 1, b: 2 } === #{ b: 2, a: 1 });
20
21 // -0, +0
22 log("-0 === +0", -0 === +0);
23 log("#[-0] === #[+0]", #[-0] === #[+0]);
24
25 // NaN
26 log("NaN === NaN", NaN === NaN);
27 log("#[NaN] === #[NaN]", #[NaN] === #[NaN]);
28
```

Record and Tuple Playground [Proposal](#) [Polyfill](#)

```
▶ (2) ["isRecord", true]
▶ (2) ["isRecord", false]
▶ (3) ["simple", true, true]
▶ (2) ["nested", true]
▶ (2) ["lorder", true]
▶ (2) ["-0 === +0", true]
▶ (2) ["#[-0] === #[+0]", false]
▶ (2) ["NaN === NaN", false]
▶ (2) ["#[NaN] === #[NaN]", true]
```

Record and Tuple Spec Text

<https://tc39.es/proposal-record-tuple>



Notable sections:

- [Record::equal](#) and [Tuple::equal](#)
- [Abstract Operations](#) updated
- [Record exotic object](#) wrapper
- [Tuple exotic object](#) wrapper
- [Record initializer](#) syntax & semantics
- [Tuple initializer](#) syntax & semantics
- [typeof unary expression](#)
- [Record](#) & [Tuple](#) objects...
- ... with the [Tuple prototype](#)

We also started reaching out to the W3C TAG for a preliminary review.
The review is now approved.

<https://github.com/w3ctag/design-reviews/issues/518>

Seeking Stage 2

- Last meeting's open questions are now solved.
- Toy Implementation & Spec Text written.
- Positive feedback in framework outreach calls.

We are now seeking for Stage 2 and reviewers.

Stage 2?

Last time: discussing issues
[#20](#) / [#65](#).

Resolution found during
hallway track discussions!

```
// SameValue
#[-0]    !== #[+0]
#[NaN]   === #[NaN]

// Strict Equality
#[-0]    === #[+0]
#[NaN]   !== #[NaN]

// Normalization
Object.is(
  #[-0][0],
  +0)
```