Record & Tuple

Stage 1 Update @ TC39 June 2020

Robin Ricard & Rick Button
Bloomberg

Advisor: Daniel Ehrenberg
Igalia

A recap of last update

Syntax

```
const contents = #[
    #{
        text: "Record and Tuple",
        font: "Comic Sans",
    },
    #{
        text: "An ECMA TC39 Stage 1 Proposal",
    },
];
```

```
// no methods in records
#{
    method() {
        // not allowed
    },
};

// no holes in tuples
#[1,,2]; // not allowed
```

Equality Semantics

-0 === +0

Issues #20 / #65
Actively being discussed

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

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

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

New since the last update

Disallowing symbols as property keys in records

```
// Insertion order must not matter for equality.
const one = #{ a: 0, b: 1 };
const two = #{ b: 1, a: 0 };
assert(one === two);

// Solution: Record keys are sorted.
Object.keys(#{ a: 0, b: 1 }); // ["a", "b"]
Object.keys(#{ b: 1, a: 0 }); // ["a", "b"]
```

```
// What about symbol keys?
const sym1 = Symbol();
const sym2 = Symbol();

const rec = #{
    [sym1]: "foo",
    [sym2]: "bar",
};

Object.getOwnPropertySymbols(rec) // ???
```

No way to order symbol keys without a global order for unregistered symbols

However, concerns with global symbol order introducing side-channel (see issue #15)

Thus, disallowing symbols as keys in Records is our solution.

```
// TypeError
const rec = #{ [Symbol("foo")]: "foo" };

// TypeError
const rec = #{ [Symbol.for("foo")]: "foo" };
```

Destructuring Syntax?

Briefly mentioned last meeting

```
// already possible
const { foo } = #{ foo: "foo" };
asserts(foo === "foo");

// rest properties work for Record/Tuple
const { foo, ...rest } = #{ foo: "foo", bar: "bar" };

// :(
assert(typeof rest === "object");
```

```
// rest properties/elements syntax for Record/Tuple?
const #{ foo, ...rest } = #{ foo: "foo", bar: "bar" };
assert(rest === #{ bar: "bar" });

const #[one, two, ...rest] = #[1,2,3,4,5];
assert(rest === #[3,4,5]);
```

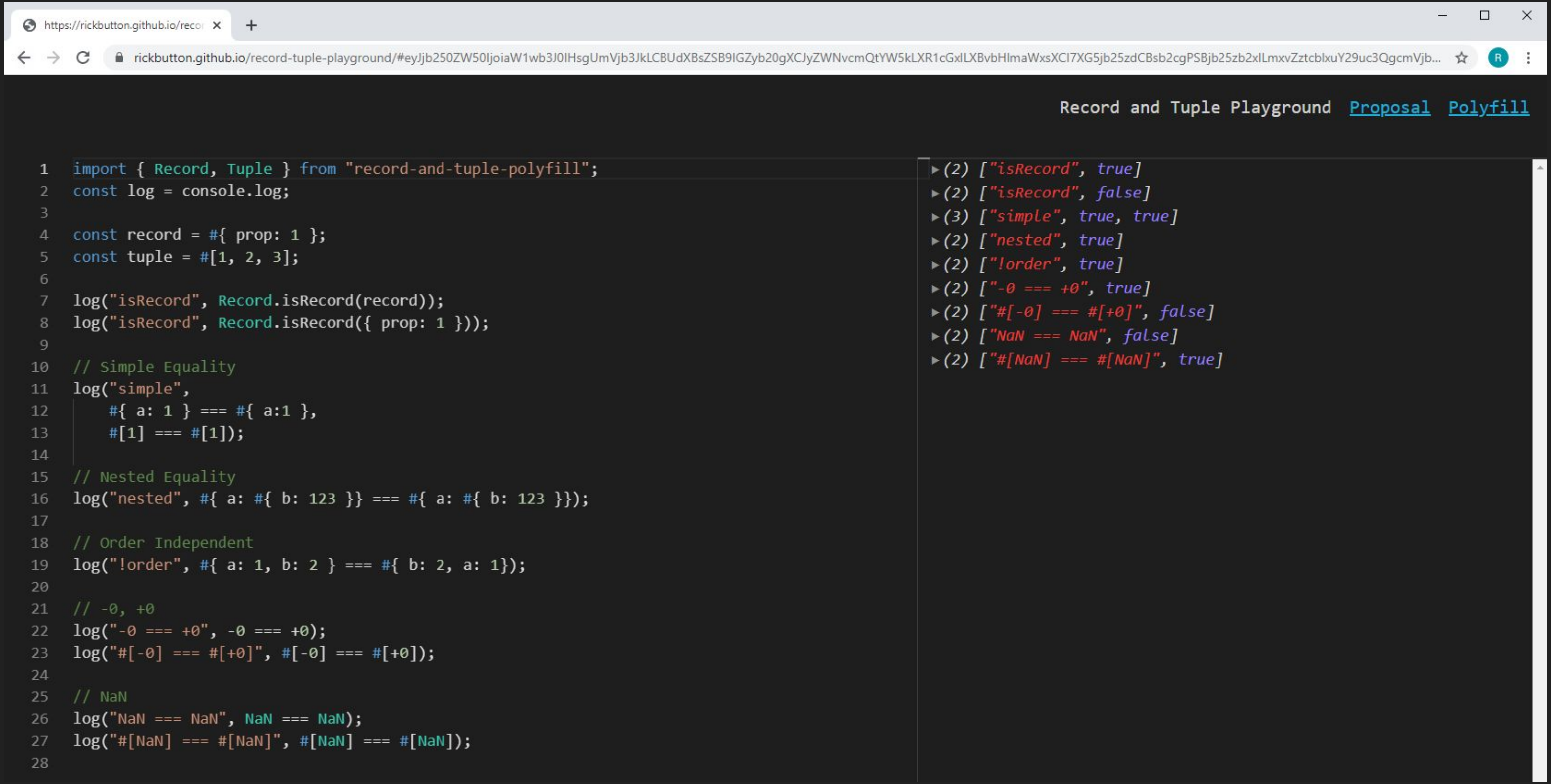
```
// tricky (but expected) behavior
// if destructuring an object with
// object property values into a "rest properties Record"
const #{ foo, ...rest } = { foo: 123, bar: {} };
// ^ TypeError, can't create Record containing object

const #{ bar, ...rest } = { foo: 123, bar: {} };
// works, but feels bad
```

Destructuring Syntax?

Resolution: Omit, investigate further in the future

Please try the Record and Tuple Playground!

https://tinyurl.com/RecordTupleFeedback


https://rickbutton.github.io/recor
rickbutton.github.io/record-tuple-playground/#eyJjb250ZW50IjoiaW1wb3J0IHsgUmVjb3JkLCBUdXBsZSB9IGZyb20gXCJyZWNvcmQtYW5kLXR1cGxlLXBvbHlmaWxsXCI7XG5jb25zdCBsb2cgPSBjb25zb2xlLmxvZztcblxuY29uc3QgcmVjb3JkID0gI3sgcHJvcDogMSB9O1xuY29uc3QgdHVwbGUgPSAjWzEsIDIsIDNdO1xuXG5sb2coXCJpc1JlY29yZFwiLCBSZWNvcmQuaXNSZWNvcmQocmVjb3JkKSk7XG5sb2coXCJpc1JlY29yZFwiLCBSZWNvcmQuaXNSZWNvcmQoeyBwcm9wOiAxIH0pKTs...
Record and Tuple Playground
Proposal
Polyfill

```javascript
import { Record, Tuple } from "record-and-tuple-polyfill";
const log = console.log;

const record = #{ prop: 1 };
const tuple = #[1, 2, 3];

log("isRecord", Record.isRecord(record));
log("isRecord", Record.isRecord({ prop: 1 }));

// Simple Equality
log("simple",
    #{ a: 1 } === #{ a:1 },
    #[1] === #[1]);

// Nested Equality
log("nested", #{ a: #{ b: 123 }} === #{ a: #{ b: 123 }});

// Order Independent
log("!order", #{ a: 1, b: 2 } === #{ b: 2, a: 1});

// -0, +0
log("-0 === +0", -0 === +0);
log("#[-0] === #[+0]", #[-0] === #[+0]);

// NaN
log("NaN === NaN", NaN === NaN);
log("#[NaN] === #[NaN]", #[NaN] === #[NaN]);

```

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

Seeking Stage 2 in July

- We believe we have a proper design for Record and Tuple
- Received good feedback on proposed solutions, hope to have more feedback on polyfill usage before Stage 2
- We have started spec work, intend to complete “first pass” before July meeting

Does anyone have additional considerations before Stage 2?

Discussion!

- Syntax/Grammar
- Equality Semantics
- Disallowing symbols as property keys in Records

Feedback before seeking Stage 2 in July?