

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



Discussed in issues [#20](#) /
[#65](#)

Chosen in order to not
introduce more values for
which `===` and `Object.is()`
differ

`==` equivalent to `===`

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

`#[NaN] === #[NaN]`

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

`#[NaN] == #[NaN]`

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" };
```

Please try the Record and Tuple Playground!

<https://tinyurl.com/RecordTupleFeedback>

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

```
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
```

▶ (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]

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?