Record & Tuple

Immutable data structures in JS?

Bloomberg Engineering

Web Engines Hackfest 2021
May 6th, 2021

Rick Button
JavaScript Infrastructure Engineer
TC39 Delegate

TechAtBloomberg.com

A Tour of Record & Tuple

Bloomberg

Engineering

Record Syntax

```
const record = #{
    name: "Record & Tuple",
    stage: 2,
};
```

Bloomberg

Engineering

Tuple Syntax

```
const tuple = #["Record & Tuple", 2];
```

Bloomberg

Engineering

Nested structures

```
const proposals = #[
    #{  name: "Record & Tuple",
        stage: 2, },
    #{  name: "Change Array by Copy",
        stage: 1, },
];
```

Bloomberg

Engineering

Immutability

```
const rt = #{ name: "Record & Tuple",
              stage: 2, };

rt.name = "Record & Toople"; // ❌
```

Bloomberg

Engineering

Deep immutability

```
const proposals = #[
    #{  name: "Record & Tuple",
        stage: 2, },
    #{  name: "Change Array by Copy",
        stage: 0, },
];

proposals[0].name = "Record & Toople"; // ❌
```

Bloomberg

Engineering

Deep immutability / Object.freeze?

```
const config = {
    db: { driver: "postgres",
          host: "pg0", },
    // ...
};
Object.freeze(config);
await initDrivers(config);
assert(config.db.host === "pg0"); // ❓
```

Bloomberg

Engineering

Deep immutability / **Defensive cloning**

```
const config = {
    db: { driver: "postgres",
          host: "pg0", },
    // ...
};
const initConfig =
    JSON.parse(JSON.stringify(config)); // 🐌
await initDrivers(initConfig);
assert(config.db.host === "pg0"); // ✅
```

Bloomberg

Engineering

Deep immutability / **No cloning, no changes!**

```
const config = #{
    db: #{ driver: "postgres",
           host: "pg0", },
    // ...
};
await initDrivers(initConfig);
assert(config.db.host === "pg0"); // ✅
```

Bloomberg

Engineering

Deep immutability / **No objects in R&T!**

```
const config = #{
    db: { driver: "postgres", // ❌
          host: "pg0", },
    // ...
};
```

Bloomberg

Engineering

Deep immutability / **Boxes: explicit interior identity**

⚠️ Functions have identity!

```
const config = #{
    db: #{
        driver: "postgres",
        host: "pg0",
        onConnect: Box(() => {
            // ...
        }),
    },
};
config.db.onConnect.unbox()();
```

Bloomberg

Engineering

Equality

```
[1, 2, 3] !== [1, 2, 3]
 { a: 1 } !== { a: 1 }

#{ a: 1 } === #{ a: 1 }
#[1, 2, 3] === #[1, 2, 3]
```

Bloomberg

Engineering

Equality / Identity

```
const srcPath = ["src", "index.ts"];
const distPath = ["dist", "index.js"];

assert(srcPath !== distPath);
assert(srcPath === ["src", "index.ts"]); // ❌
```

Bloomberg

Engineering

Equality / **Identity-less-ness**

```
const srcPath = #["src", "index.ts"];
const distPath = #["dist", "index.js"];

assert(srcPath !== distPath);
assert(srcPath === #["src", "index.ts"]);
```

Bloomberg

Engineering

Equality / Indexing by identity

```
const utilPath = ["dist", "util.js"];
const sourceMapping = new Map();
sourceMapping.set(["dist", "index.js"],
                  ["src", "index.ts"]);
sourceMapping.set([utilPath, ["src", "util.ts"]]);

sourceMapping.get(utilPath);
// => ["src", "util.ts"]
sourceMapping.get(["dist", "util.js"]);
// => undefined
```

Bloomberg

Engineering

Equality / Indexing by value

👍 Records can be looked up too!

```
const sourceMapping = new Map();
sourceMapping.set(#["dist", "index.js"],
                  #["src", "index.ts"]);
sourceMapping.set(#["dist", "util.js"],
                  #["src", "util.ts"]);

sourceMapping.get(#["dist", "util.js"]);
// => #["src", "util.ts"]
```

Bloomberg

Engineering

Update by copy

```
const root = #["C:", "dev", "rt-project"];
const rel = #["src", "index.ts"];

const abs = // ?
```

Bloomberg

Engineering

Update by copy / **Spread**

👍 Records can be spread too!

```
const root = #["C:", "dev", "rt-project"];
const rel = #["src", "index.ts"];

const abs = #[...root, ...rel];
// => #["C:", "dev", "rt-project", "src", "index.ts"]
```

Bloomberg

Engineering

Update by copy / New Methods

```
const root = #["C:", "dev", "rt-project"];

const abs = root.pushed("src", "index.ts");
// => #["C:", "dev", "rt-project", "src", "index.ts"]
const rev = abs.reversed();
// => #["index.ts", "src", "rt-project", "dev", "C:"]
```

Bloomberg

Engineering

The TC39 Committee: Advancement of Record & Tuple

TechAtBloomberg.com

Bloomberg

Engineering

TC39 / The Stage Process

- **Stage 0**: Proposals are ideas
- **Stage 1**: The committee is interested by the proposed idea
- ***Stage 2****: The committee intends to specify the proposal*
- **Stage 3**: The proposal has a spec and should land in the language with minor changes
- **Stage 4**: The proposal is implemented in major browsers and will ship in the next yearly specification

Record & Tuple

Record & Tuple Status

- ✅ Spec Text Draft
- ✅ Babel Syntax Parser
- ✅ Babel Syntax Transform
- ✅ Polyfill
- 🚧 Tuple toy implementation in SpiderMonkey (Firefox)
  by Nicolò Ribaudo (Babel Maintainer Team)

Record & Tuple Status / **Open for experimentation!**

- **Experimentation is encouraged**
- Production use **is not advised**
- **Things will change** in the future **thanks to experimentation**

Record & Tuple Playground
https://rickbutton.github.io/record-tuple-playground/

Resources

Proposal	https://github.com/tc39/proposal-record-tuple
Spec text	https://tc39.es/proposal-record-tuple/
Playground	https://rickbutton.github.io/record-tuple-playground/

Performance

Comparison semantics are linear time, but there are some strategies for improving comparison performance in-engine, for example:

Interning - only one **“engine value”** for a given **“language value”:**

- Comparison results in a quick pointer check
- Can be performed at construction time or comparison time

Performance / **Complications?**

- Only some constructed records/tuples will be compared
- Records/Tuples containing **-0** require additional overhead or fallback to linear time **(performance cliffs!)**
  — **-0 === 0, #[-0] === #[0]**
- In theory, most Record/Tuple comparisons are between small values (**Map** keys/**Set** values)

Performance / **Linear Time Comparison?**

- Less implementation complexity
  - No need to worry about **whether to / when to intern**
  - No need to handle **-0/+0** complexity

- More performance consistency

- Matches existing linear time comparison expectation for userland deep equality

- Keeps the door open for future optimizations as Record/Tuple is observed in the wild

Thank You!

Rick Button - JS Infrastructure Engineer & TC39 Delegate

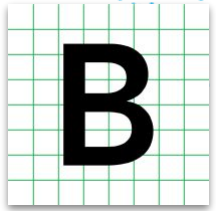
- @rickbutton
- @rickbutton

Bloomberg Engineering

- @TechAtBloomberg
- @bloomberg

TechAtBloomberg.com

https://careers.bloomberg.com/job/search?fd=Engineering



TechAtBloomberg.com

Bloomberg

Engineering