Symbols complicated it all

by Robin Pokorny

Why talk about it?

Symbols

```
const sym = Symbol();

typeof sym;
// -> "symbol"

sym === Symbol();
// -> false
```

Primitive values

- boolean
- number
- string
- undefined
- null
- **symbol**

Primitive values

- boolean
- number
- string
- undefined
- null
- **symbol**

complication #1

What for?

```
const sym = Symbol()

const o = {
  name: "My key is a string"
  [sym]: "My key is a symbol"
}

o[sym1]
// -> "My key is a symbol"
```

Convertion

```
Boolean(sym);
// -> true
```

```
!sym;
// -> false
```

Convertion

```
Number(sym);
// -> TypeError
```

```
sym * 5;
// -> TypeError
```

complication #2

Convertion

```
String(sym);
// -> "Symbol()"
```

```
"" +
  sym`;
${sym}`;
// -> TypeError
```

complication #3

Object inspection

symbol vs enumerable

```
const c = Symbol();
const d = Symbol("d");

const o = { a: 1, [c]: 10 };

// non-enumerable by default
Object.defineProperty(o, "b", { value: 2 });
Object.defineProperty(o, d, { value: 20 });
```

```
> o
< ▼{a: 1, b: 2, Symbol(): 10, Symbol(d): 20} i
    a: 1
    Symbol(): 10
    b: 2
    Symbol(d): 20
  ▶ __proto__: Object
```

Access

```
o.a;
// -> 1
o.b;
// -> 2
o[c];
// -> 10
o[d];
// -> 20
```

hasOwnProperty

```
o.hasOwnProperty("a");
// -> true
o.hasOwnProperty("b");
// -> true
o.hasOwnProperty(c);
// -> true
o.hasOwnProperty(d);
// -> true
```

keys

```
Object.keys(o);
// -> ["a"]

Object.entries(o);
// -> [["a", 1]]
```

complication #4

Descriptors

```
Object.getOwnPropertyDescriptors(o);
// -> {
//    a: {…}, b: {…},
//    Symbol(): {…}, Symbol(d): {…}
// }
```

complication #5

getOwn

```
Object.getOwnPropertyNames(o);
// -> ["a", "b"]
Object.getOwnPropertySymbols(o);
// -> [Symbol(), Symbol(d)]
```

assign

```
Object.assign({}, o);
// -> {a: 1, Symbol(): 10}


{ ...o };
// -> {a: 1, Symbol(): 10}
```

JSON

```
JSON.stringify(o);
// -> `{"a":1}`
```

complication #6

Reflect

```
Reflect.ownKeys(o);
// -> ["a", "b", Symbol(), Symbol(d)]
```

Where to use

. Hidden properties

```
onst ROLE = Symbol();

onst createAdmin = name => ({
 name,
 [ROLE]: "admin"
);

onst isAdmin = ({ [ROLE]: role }) => role === "admin"
```

2. Meta-data

- Well-known symbols on Symbol object
- `Symbol.iterator`
- written as `@@iterator`
- Option for libraries

Some well-known symbols

- `@@iterator`
- `@@toPrimitive`
- `@@match`
- `@@isConcatSpreadable`

Symbols are your friends!

If you do not overuse them…

/robinpokorny

symbols-complicated-it-all.netlify.com