Object rest and spread properties

By Mathias Bynens
V8 JavaScript whisperer

Before discussing *object rest and spread properties*, let's take a trip down memory lane and remind ourselves of a very similar feature.

ES2015 array rest and spread elements

Good ol' ECMAScript 2015 introduced *rest elements* for array destructuring assignment and *spread elements* for array literals.

```
// Rest elements for array destructuring assignment:
const primes = [2, 3, 5, 7, 11];
const [first, second, ...rest] = primes;
console.log(first); // 2
console.log(second); // 3
console.log(rest); // [5, 7, 11]

// Spread elements for array literals:
const primesCopy = [first, second, ...rest];
console.log(primesCopy); // [2, 3, 5, 7, 11]
```

These ES2015 features have been supported since Chrome 46 and Chrome 47, respectively.

ES.next: object rest and spread properties 🆕

So what's new, then? Well, a stage 3 proposal enables rest and spread properties for object literals, too.

```
// Rest properties for object destructuring assignment:
const person = {
    firstName: 'Sebastian',
    lastName: 'Markbåge',
    country: 'USA',
    state: 'CA',
};
```

```
const { firstName, lastName, ...rest } = person;
console.log(firstName); // Sebastian
console.log(lastName); // Markbåge
console.log(rest); // { country: 'USA', state: 'CA' }

// Spread properties for object literals:
const personCopy = { firstName, lastName, ...rest };
console.log(personCopy);
// { firstName: 'Sebastian', lastName: 'Markbåge', country: 'USA', state: 'CA' }
```

Spread properties offer a more elegant alternative to **`Object.assign()`** in many situations:

```
// Shallow-clone an object:
const data = { x: 42, y: 27, label: 'Treasure' };
// The old way:
const clone1 = Object.assign({}, data);
// The new way:
const clone2 = { ...data };
// Either results in:
// { x: 42, y: 27, label: 'Treasure' }

// Merge two objects:
const defaultSettings = { logWarnings: false, logErrors: false };
const userSettings = { logErrors: true };
// The old way:
const settings1 = Object.assign({}, defaultSettings, userSettings);
// The new way:
const settings2 = { ...defaultSettings, ...userSettings };
// Either results in:
// { logWarnings: false, logErrors: true }
```

However, there are some subtle differences in how spreading handles setters:

1. **`Object.assign()`** triggers setters; spread doesn't.
2. You can stop **`Object.assign()`** from creating own properties via inherited read-only properties, but not the spread operator.

Axel Rauschmayer's write-up explains these gotchas in more detail.

Object rest and spread properties are supported by default in V8 v6.0.75+ and Chrome 60+. Consider transpiling your code until this feature is more widely supported across engines.

Last updated July 2, 2018.