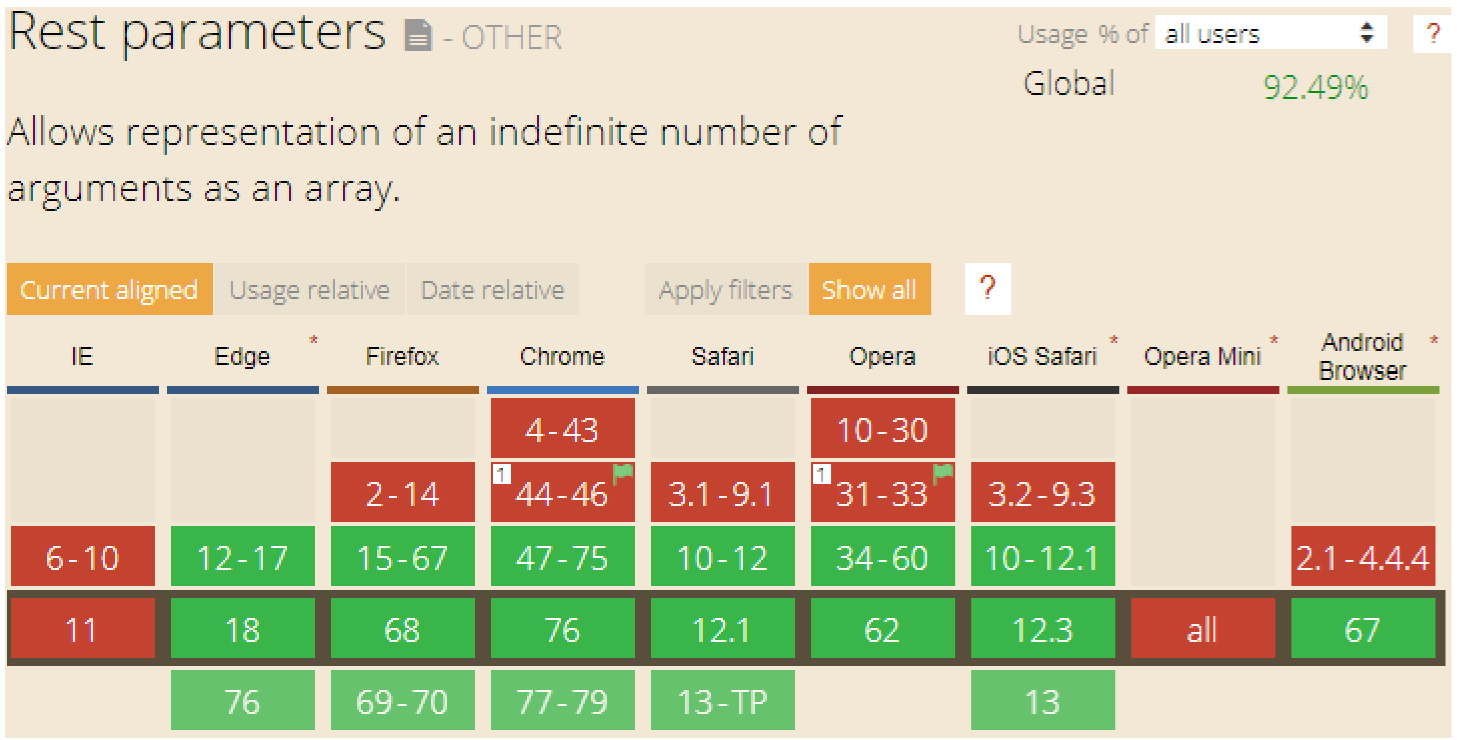
SPREAD AND REST

Andrew Sheehan
MET CS: Web Development

THE ‘3-DOTS’
ECMASCRIPT (ECMA6) FEATURE

...

REST PARAMETERS
NOT WITH INTERNET EXPLORER



Rest parameters - OTHER

Usage % of all users ?

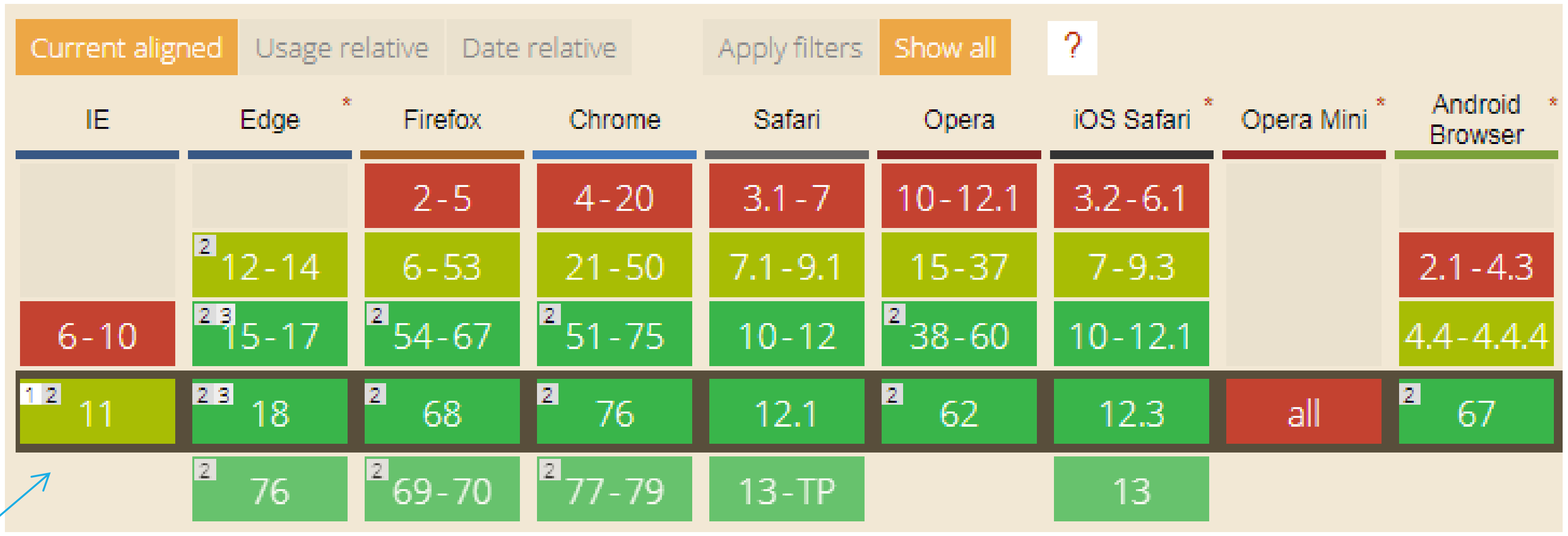
Global 92.49%

Allows representation of an indefinite number of arguments as an array.

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *
			🟥 4-43		🟥 10-30			
		🟥 2-14	🟥 [1] 44-46	🟥 3.1-9.1	🟥 [1] 31-33	🟥 3.2-9.3		
🟥 6-10	🟩 12-17	🟩 15-67	🟩 47-75	🟩 10-12	🟩 34-60	🟩 10-12.1		🟥 2.1-4.4.4
🟥 11	🟩 18	🟩 68	🟩 76	🟩 12.1	🟩 62	🟩 12.3	🟥 all	🟩 67
	🟩 76	🟩 69-70	🟩 77-79	🟩 13-TP		🟩 13		

SPREAD OPERATOR

DON’T TRUST INTERNET EXPLORER



IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *
		2-5	4-20	3.1-7	10-12.1	3.2-6.1		
	[2] 12-14	6-53	21-50	7.1-9.1	15-37	7-9.3		2.1-4.3
6-10	[2] [3] 15-17	[2] 54-67	[2] 51-75	10-12	[2] 38-60	10-12.1		4.4-4.4.4
[1] [2] 11	[2] [3] 18	[2] 68	[2] 76	12.1	[2] 62	12.3	all	[2] 67
	[2] 76	[2] 69-70	[2] 77-79	13-TP		13		

Partial support

WHAT IS THE DIFFERENCE

The … could be used either way

It depends on how you use it.

WHAT IS
THE DIFFERENCE

It can be used as

- ✓ A Rest Parameter
- ✓ Spread Operator

WHAT IS
THE DIFFERENCE

As Rest Parameter

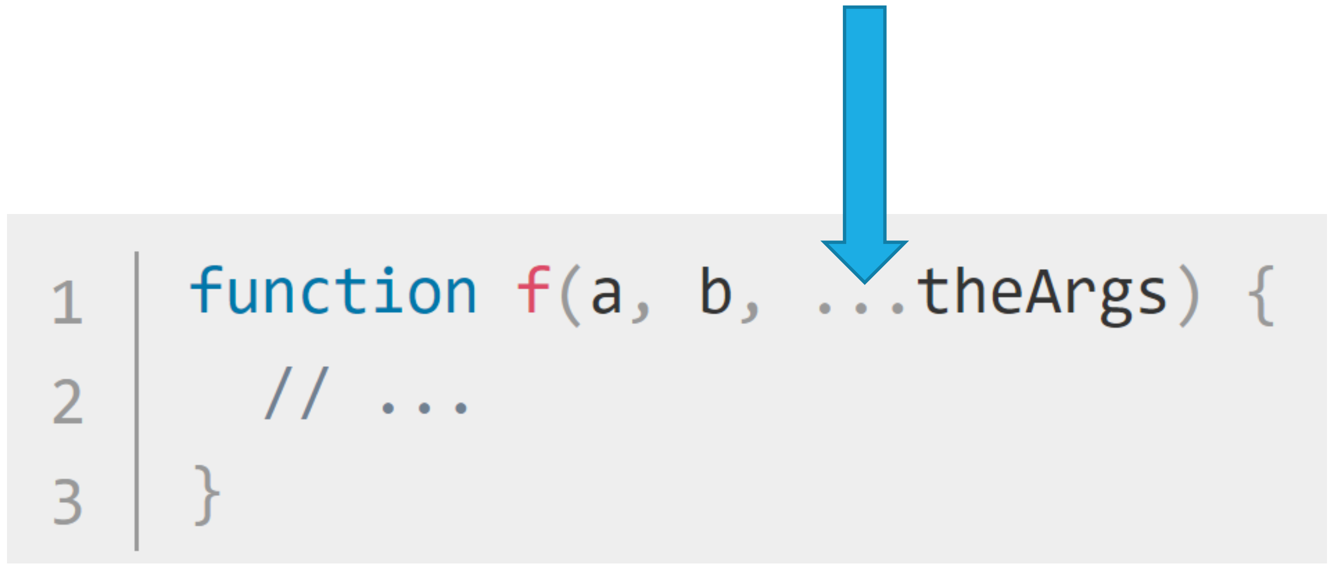
It’s use will collect all values into an array.

WHAT IS
THE DIFFERENCE

As a Spread Operator

Allows iterable things to be expanded into single elements

MORE ABOUT
THE REST OPERATOR



```
function f(a, b, ...theArgs) {
  // ...
}
```

Represents infinite arguments

MORE ABOUT
THE REST OPERATOR

Has to be the last parameter

```
const move = function(x, y, ... z) {
        // logic goes here...
}
```

MORE ABOUT
THE REST OPERATOR

Simple example

```
const  toArray =  function(...values)  {
        return values;
}
```

MORE ABOUT
THE REST OPERATOR

No values passed in?

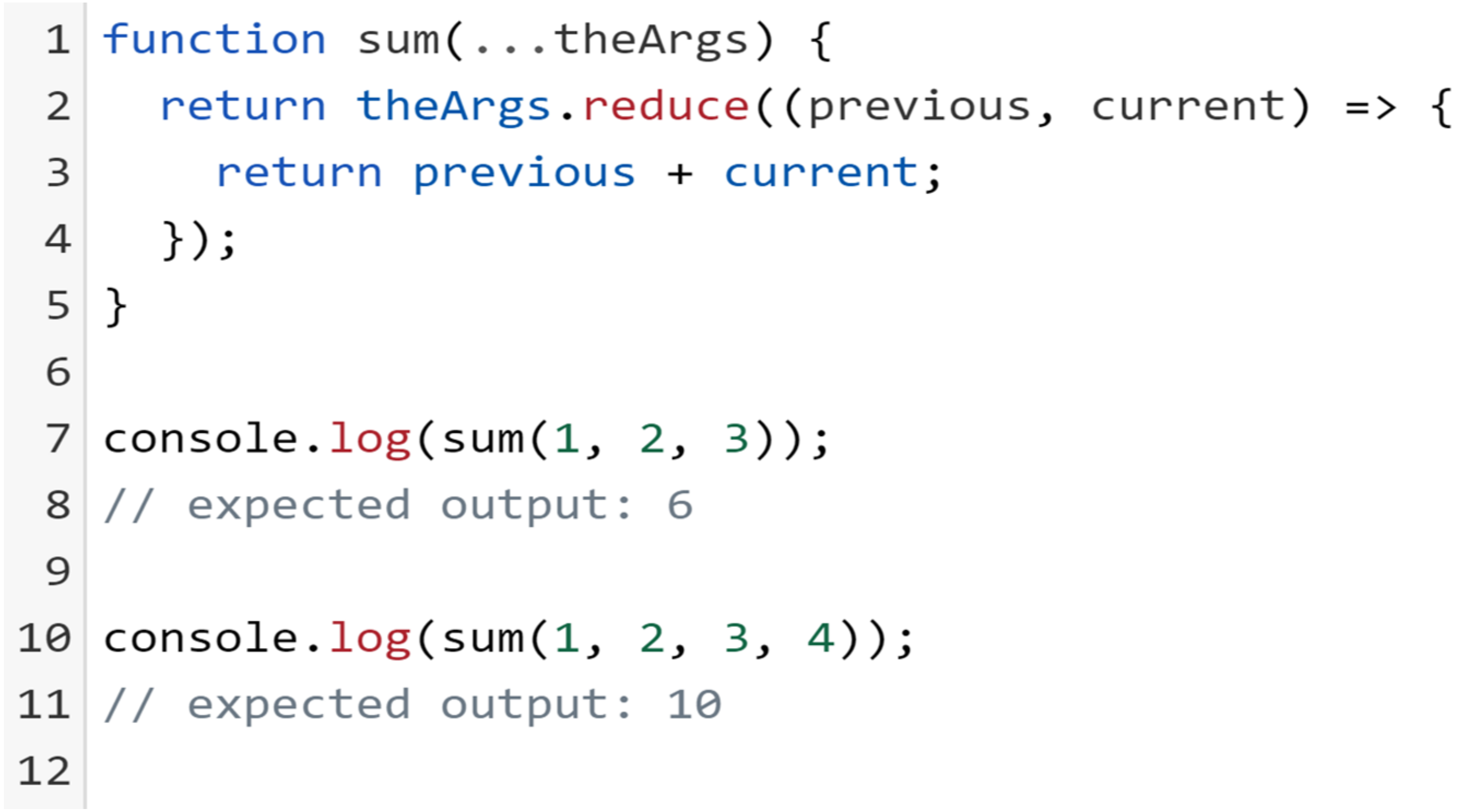
```
const  toArray =  function(...values)  {
        return values;   // returns empty [ ]
}
```

MORE ABOUT
THE REST OPERATOR

This will cause an error

```
const move = function(…x,  y, …z) {
        // logic goes here…
}
```

REST
EXAMPLE



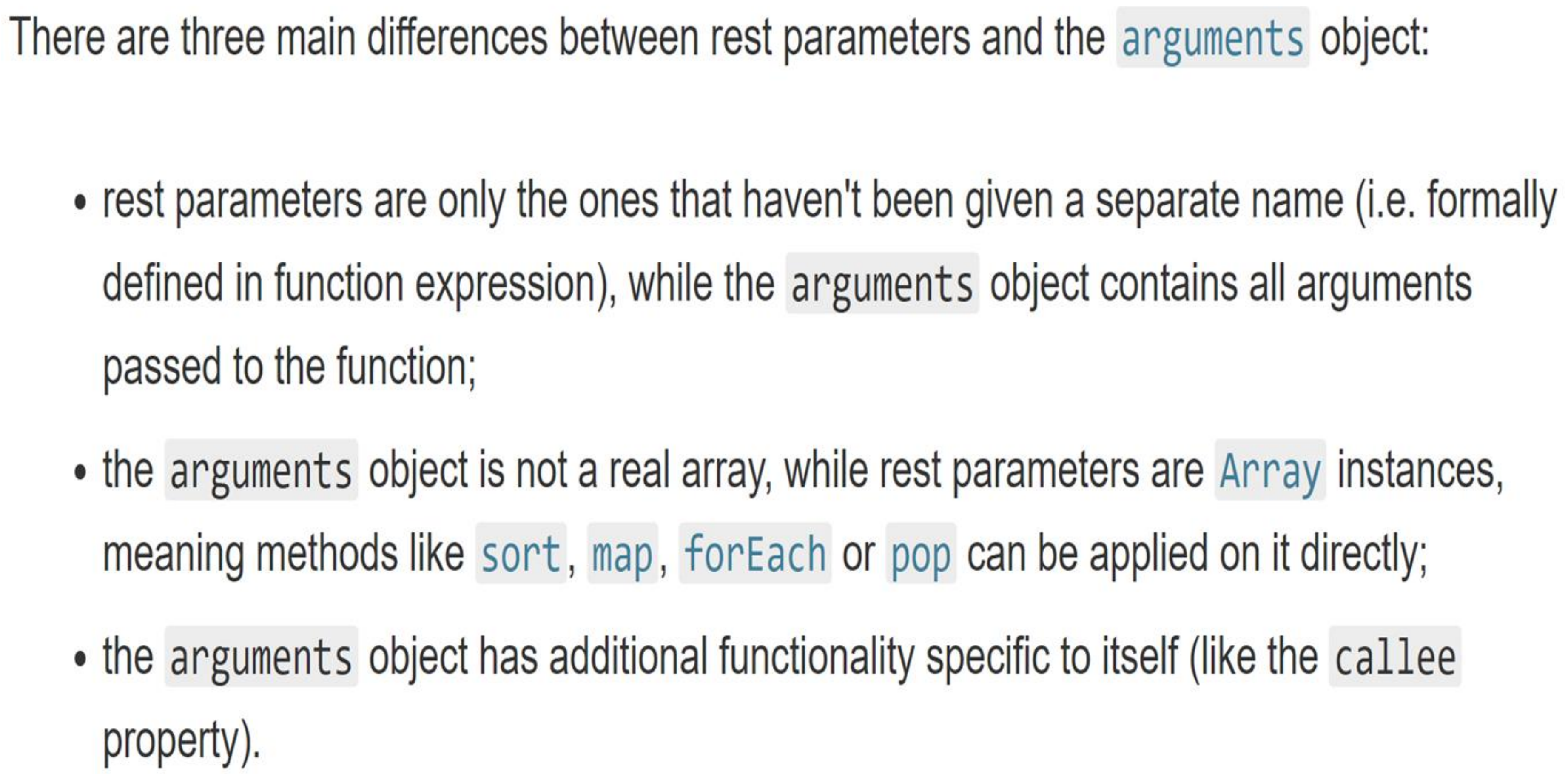
```
function sum(...theArgs) {
  return theArgs.reduce((previous, current) => {
    return previous + current;
  });
}

console.log(sum(1, 2, 3));
// expected output: 6

console.log(sum(1, 2, 3, 4));
// expected output: 10

```

REST VS .ARGUMENTS



There are three main differences between rest parameters and the `arguments` object:

- rest parameters are only the ones that haven't been given a separate name (i.e. formally defined in function expression), while the `arguments` object contains all arguments passed to the function;
- the `arguments` object is not a real array, while rest parameters are `Array` instances, meaning methods like `sort`, `map`, `forEach` or `pop` can be applied on it directly;
- the `arguments` object has additional functionality specific to itself (like the `callee` property).

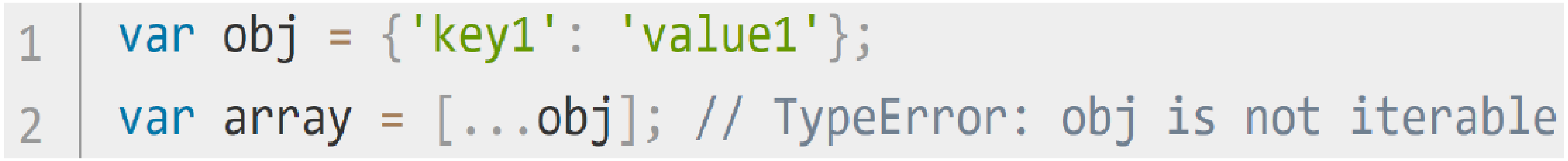
SPREAD EXPLAINED

Spread Operator

In a spread operation, you’re trying to take either an array or an object and spread it onto a new array or object

SPREAD
EXPLAINED

Only useful for Iterable things

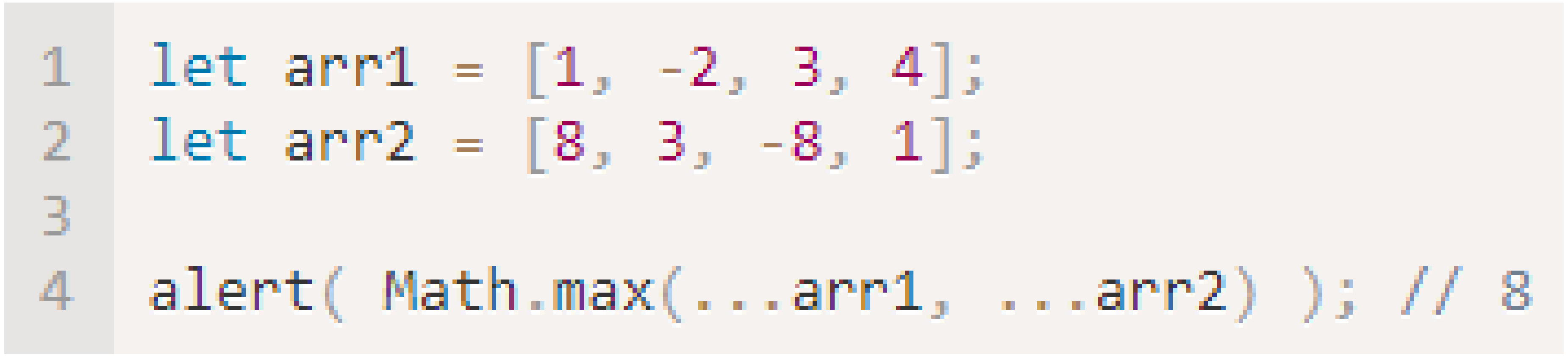


```
var obj = {'key1': 'value1'};
var array = [...obj]; // TypeError: obj is not iterable
```

SPREAD
EXAMPLE

```
const words = ['hi', 'apple', 'dart'];
const more_words = [...words, 'mom'];
```

SPREAD
USED MULTIPLE TIMES



```
let arr1 = [1, -2, 3, 4];
let arr2 = [8, 3, -8, 1];

alert( Math.max(...arr1, ...arr2) ); // 8
```

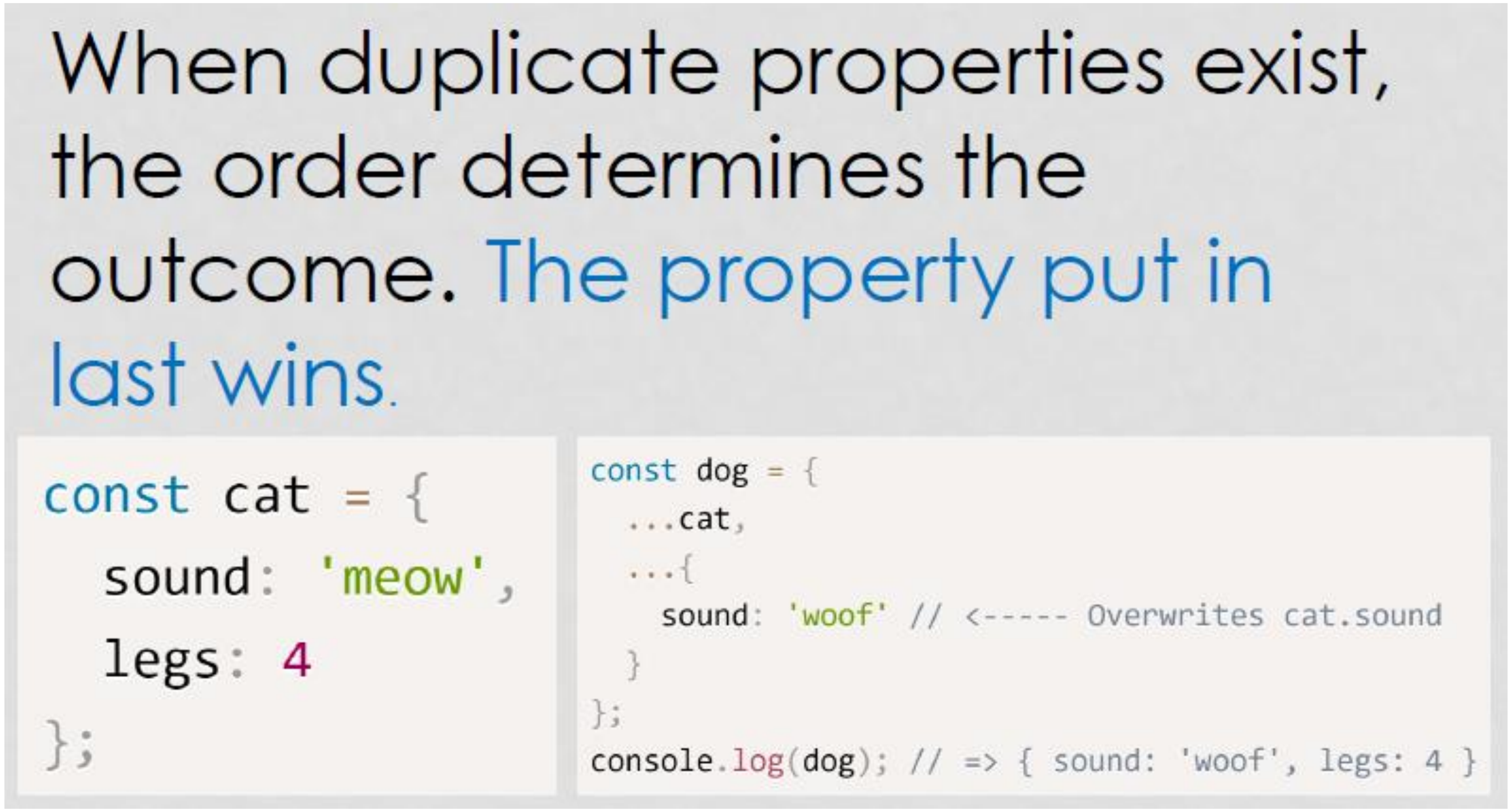
SPREAD
WITH OBJECT LITERALS

```
var obj1 = { foo: 'bar', x: 42 };
var obj2 = { foo: 'baz', y: 13 };

var clonedObj = { ...obj1 };
// Object { foo: "bar", x: 42 }

var mergedObj = { ...obj1, ...obj2 };
// Object { foo: "baz", x: 42, y: 13 }
```

SPREAD
WITH DUPLICATES

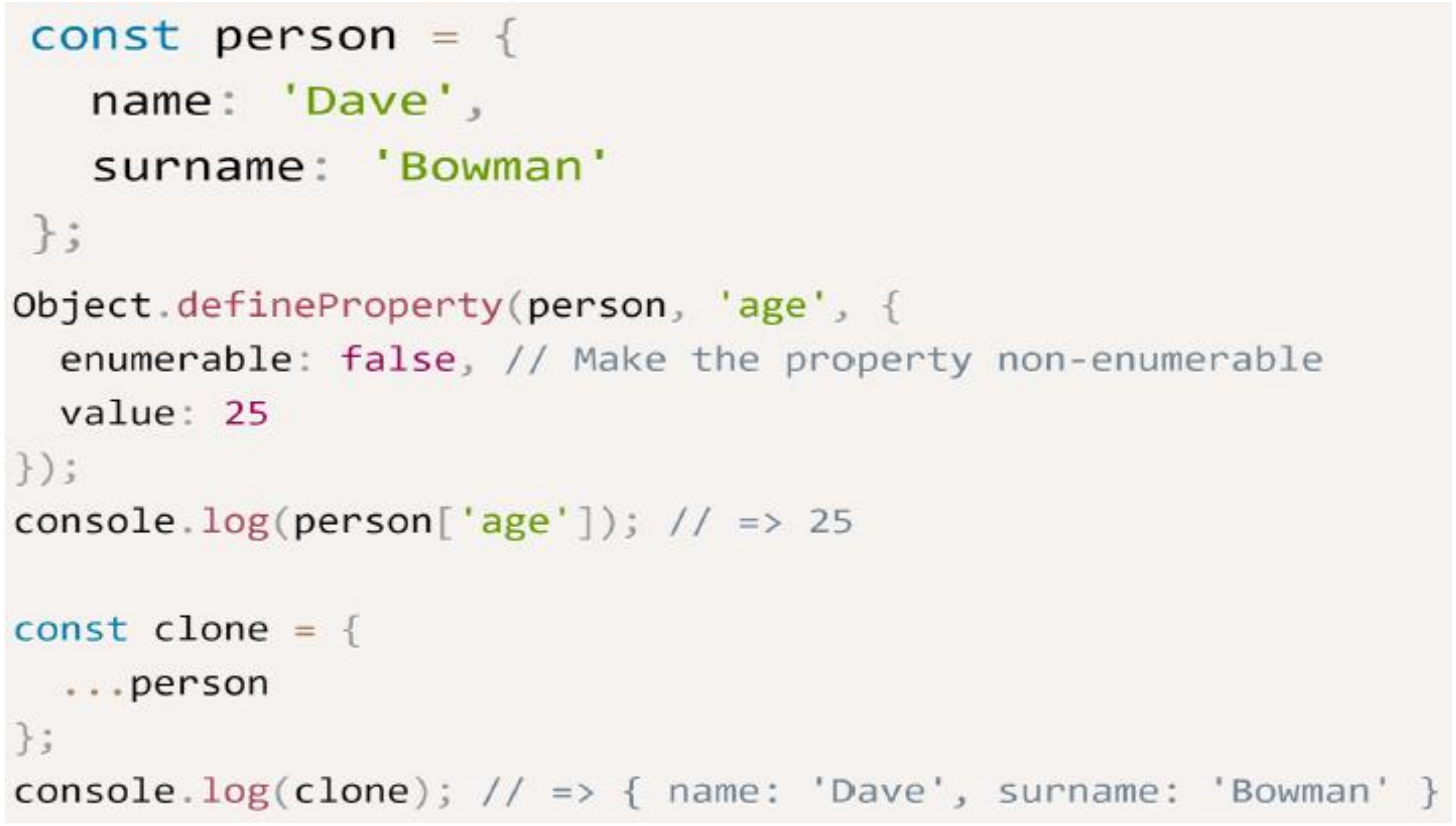


When duplicate properties exist, the order determines the outcome. The property put in last wins.

```
const cat = {
  sound: 'meow',
  legs: 4
};
```

```
const dog = {
  ...cat,
  ...{
    sound: 'woof' // <----- Overwrites cat.sound
  }
};
console.log(dog); // => { sound: 'woof', legs: 4 }
```

SPREAD
WITH NON-ENUMERABLE PROPERTIES



```
const person = {
  name: 'Dave',
  surname: 'Bowman'
};
Object.defineProperty(person, 'age', {
  enumerable: false, // Make the property non-enumerable
  value: 25
});
console.log(person['age']); // => 25

const clone = {
  ...person
};
console.log(clone); // => { name: 'Dave', surname: 'Bowman' }
```

SUMMARY

When … is at the end of function parameter list is it rest parameter

When … occurs in a function call , it’s being used in a spread operation