WEBDEVTEACHER.COM

Newer JavaScript Functions

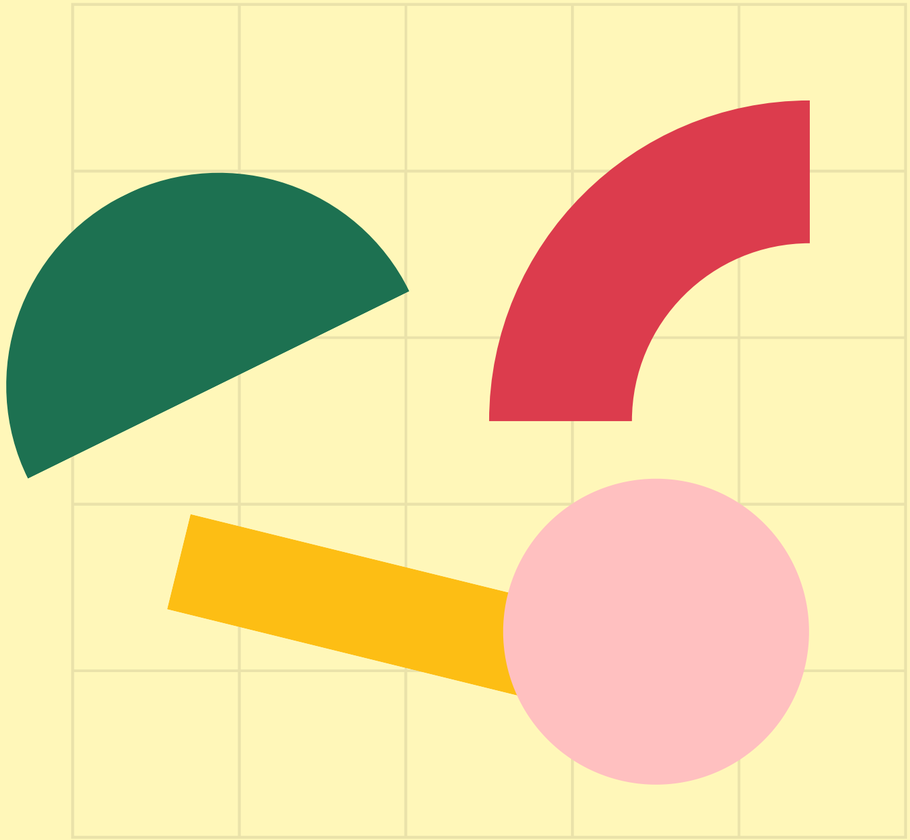
New Features

WEB DEVELOPMENT

Unit Goals

what we'll cover

- default params
- spread
- rest
- destructuring



DEFAULT PARAMS

THE OLD WAY

```
function multiply(a, b) {
    b = typeof b !== 'undefined' ? b : 1;
    return a * b;
}

multiply(7); //7
multiply(7, 3); //21
```

DEFAULT PARAMS

THE OLD WAY

```
function multiply(a, b = 1) {
    return a * b;
}

multiply(4); //4
multiply(4, 5); //20
```

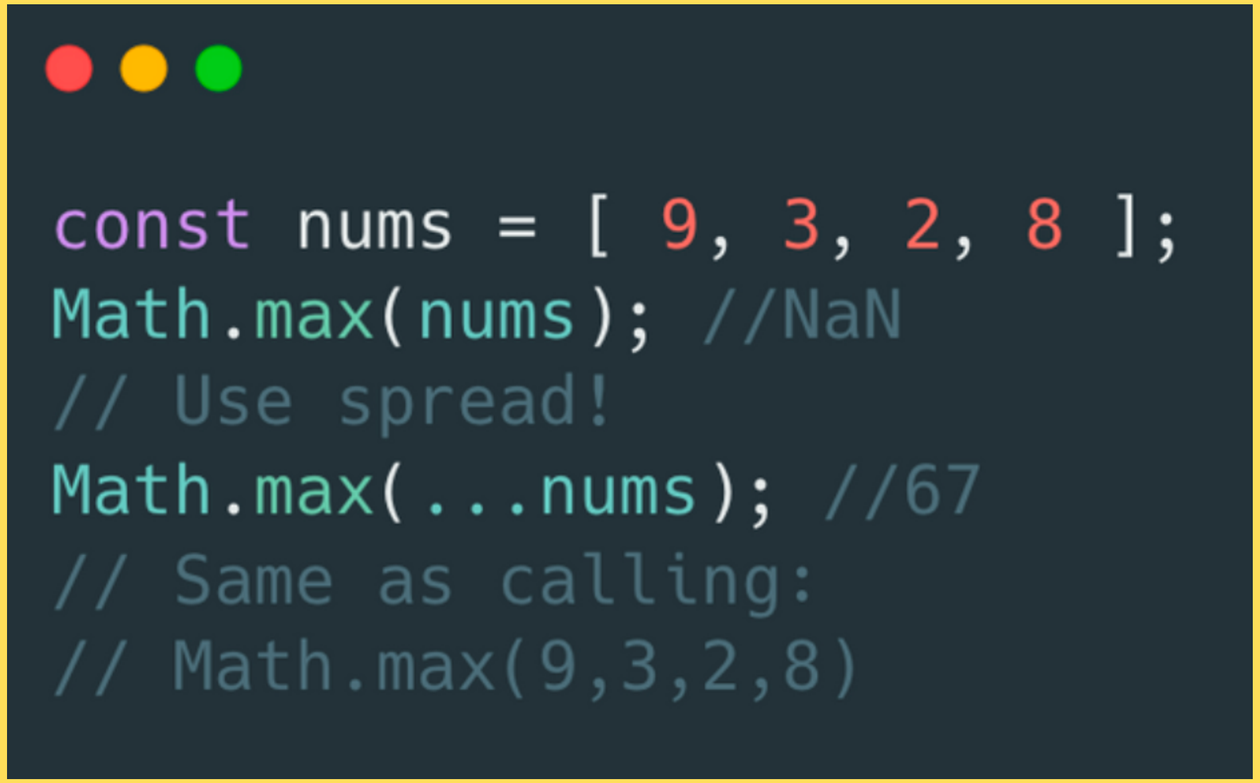
SPREAD

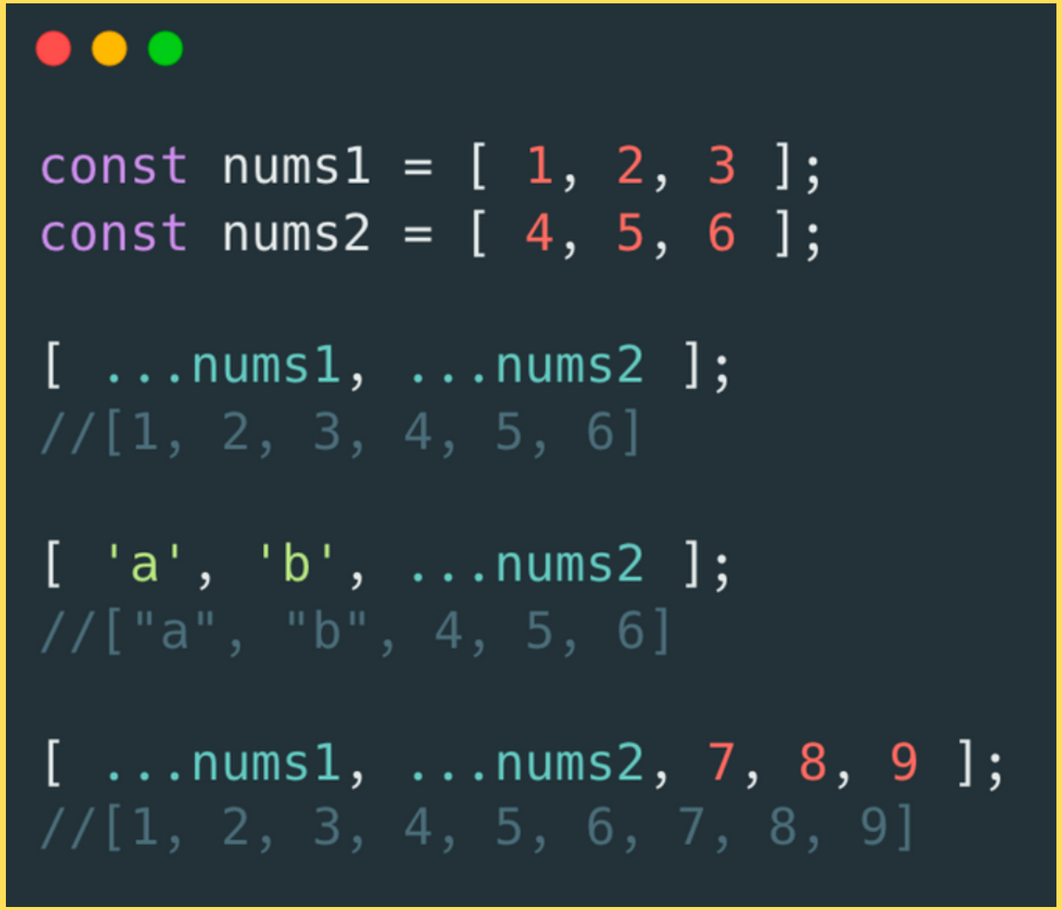
Spread syntax allows an iterable such as an array to be **expanded** in places where zero or more arguments (for function calls) or elements (for array literals) are expected, or an object expression to be expanded in places where zero or more key-value pairs (for object literals) are expected.

SPREAD

FOR FUNCTION CALLS

Expands an iterable
(array, string, etc.) into
a list of arguments





```
const nums1 = [ 1, 2, 3 ];
const nums2 = [ 4, 5, 6 ];

[ ...nums1, ...nums2 ];
//[1, 2, 3, 4, 5, 6]

[ 'a', 'b', ...nums2 ];
//["a", "b", 4, 5, 6]

[ ...nums1, ...nums2, 7, 8, 9 ];
//[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

SPREAD

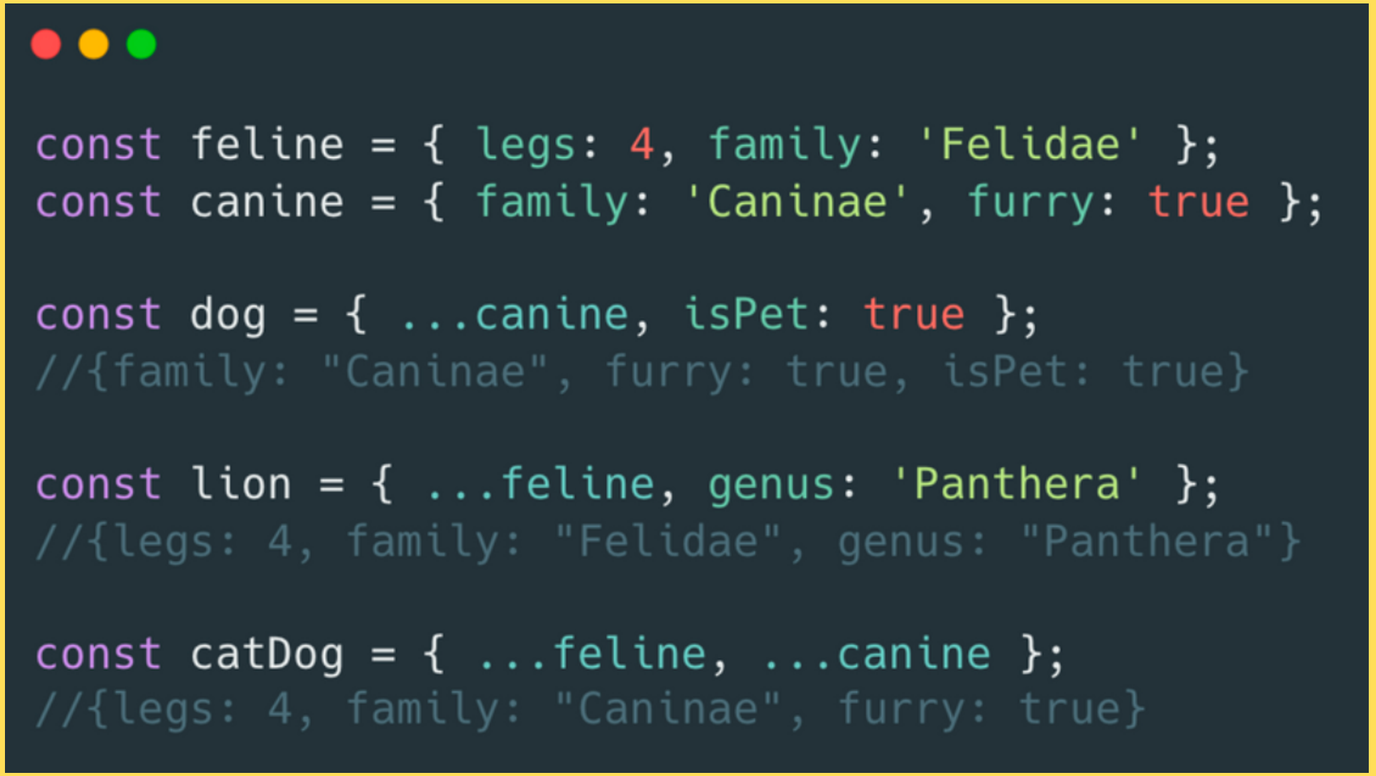
IN ARRAY LITERALS

It creates a new array using an existing array. It spreads the elements from one array into a new array.

SPREAD

IN OBJECT LITERALS

Copies properties from one object into another object literal.



```
const feline = { legs: 4, family: 'Felidae' };
const canine = { family: 'Caninae', furry: true };

const dog = { ...canine, isPet: true };
//{family: "Caninae", furry: true, isPet: true}

const lion = { ...feline, genus: 'Panthera' };
//{legs: 4, family: "Felidae", genus: "Panthera"}

const catDog = { ...feline, ...canine };
//{legs: 4, family: "Caninae", furry: true}
```

REST

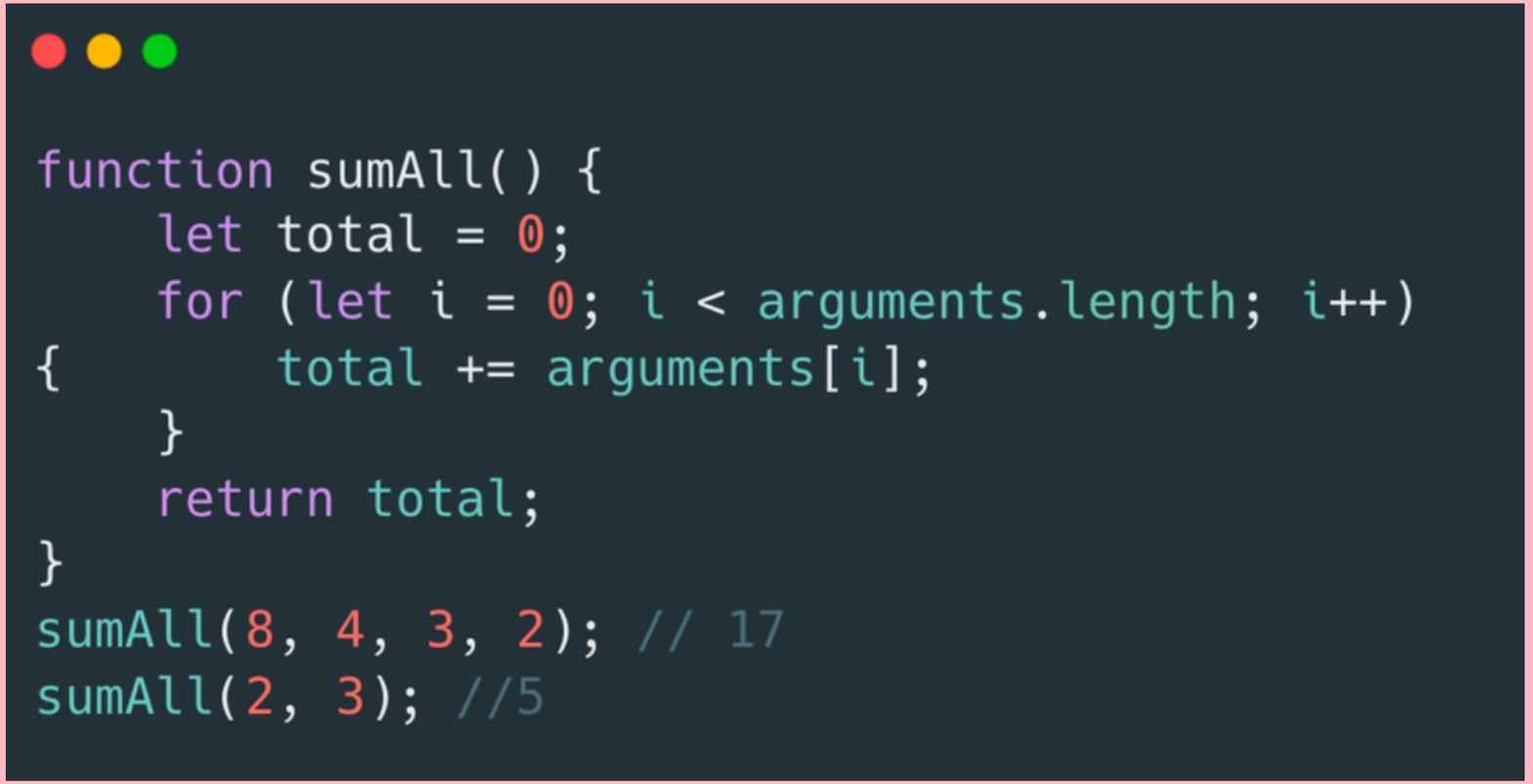
THE ARGUMENTS OBJECT

LOOKS LIKE SPREAD BUT IT'S NOT

- Available inside every function.
- It's an array-like object:
  - has a length property
  - does not have array methods like push/pop
- Contains all the arguments passed to the function.
- Not available inside of arrow functions!

REST

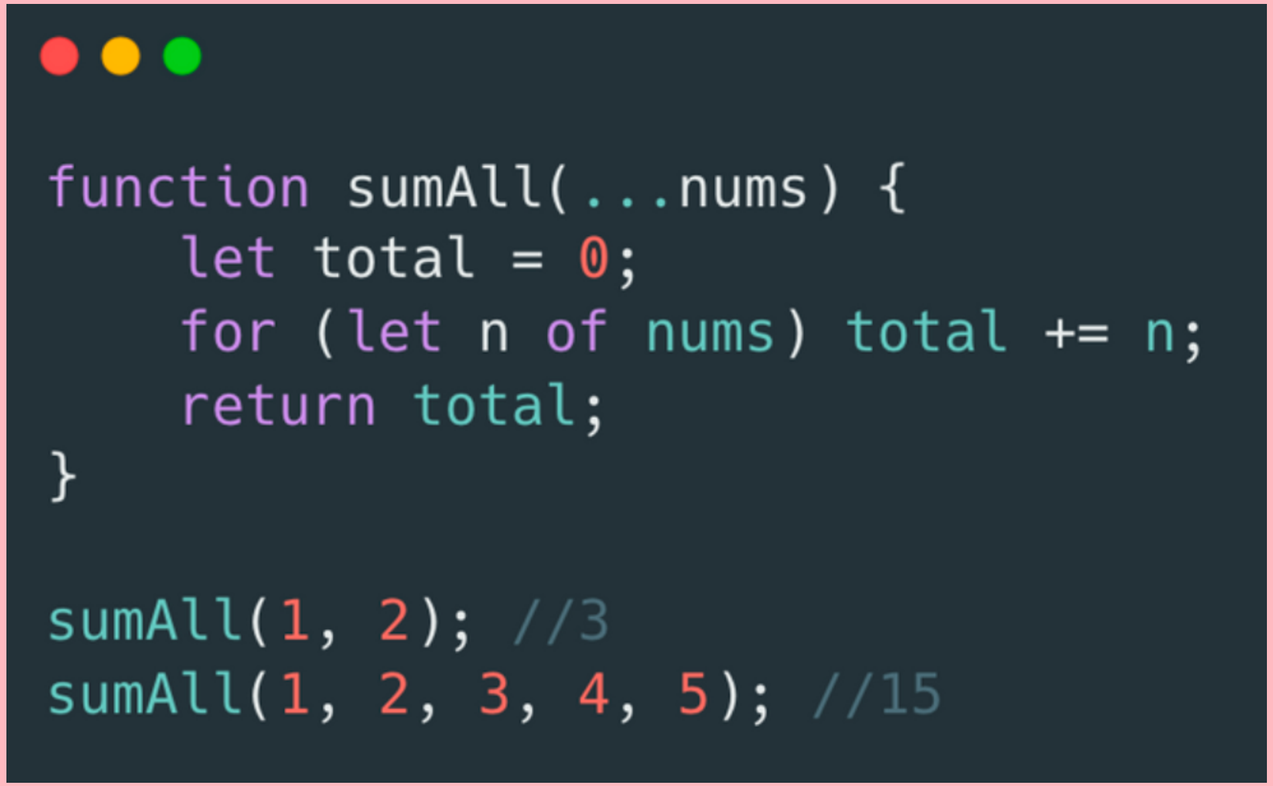
THE ARGUMENTS OBJECT



```
function sumAll() {
    let total = 0;
    for (let i = 0; i < arguments.length; i++)
{       total += arguments[i];
    }
    return total;
}
sumAll(8, 4, 3, 2); // 17
sumAll(2, 3); //5
```

REST PARAMS

Collects all remaining arguments into an actual array .



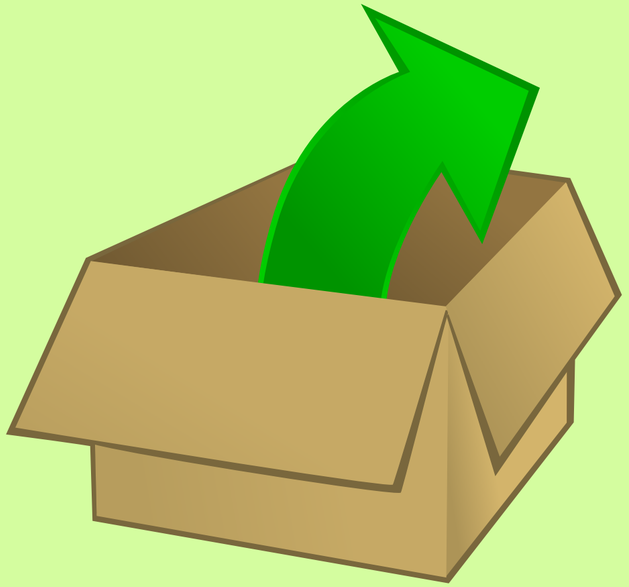
```
function sumAll(...nums) {
    let total = 0;
    for (let n of nums) total += n;
    return total;
}

sumAll(1, 2); //3
sumAll(1, 2, 3, 4, 5); //15
```

DESTRUCTURING

A short, clean syntax to 'unpack'

- values from arrays,
- properties from objects

into distinct variables.



DESTRUCTURING

ARRAY

```
const raceResults = [ 'Eliud Kipchoge', 'Feyisa Lelisa', 'Galen Rupp' ];

const [ gold, silver, bronze ] = raceResults;
gold; //"Eliud Kipchoge"
silver; //"Feyisa Lelisa"
bronze; //"Galen Rupp"

const [ fastest, ...everyoneElse ] = raceResults;
fastest; //"Eliud Kipchoge"
everyoneElse; //["Feyisa Lelisa", "Galen Rupp"]
```

DESTRUCTURING

OBJECT

```
const runner = {
  first: "Eliud",
  last: "Kipchoge",
  country: "Kenya",
  title: "Elder of the Order of the Golden Heart of Kenya"
}
const {first,last,country} = runner;

first; //"Eliud"
last; //"Kipchoge"
country; //"Kenya"
```

DESTRUCTURING

PARAM

```
const fullName = ({first, last}) => {
  return `${first}  ${last}`
}
const runner = {
  first: "Eliud",
  last: "Kipchoge",
  country: "Kenya",
}

fullName(runner); //"Eliud  Kipchoge"
```