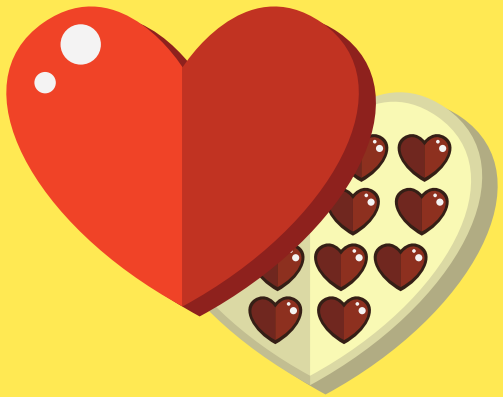
# OTHER USEFUL THINGS

A JS Assortment

# NEW-ISH FEATURES

## My goal was to include as many new JS features as early as possible...

- ✅ Arrow Functions
- ✅ String Template Literals
- ✅ Let & Const
- ✅ For...of
- ✅ For...in
- ✅ Exponent Operator
- ✅ String.includes()
- ✅ Array.includes()
- ✅ Object.values()
- ☑ Rest & Spread
- ☑ Destructuring
- ☑ Default Function Params
- ☐ Object Enhancements
- ☐ Classes
- ☐ Async Functions



# NEW-ISH FEATURES

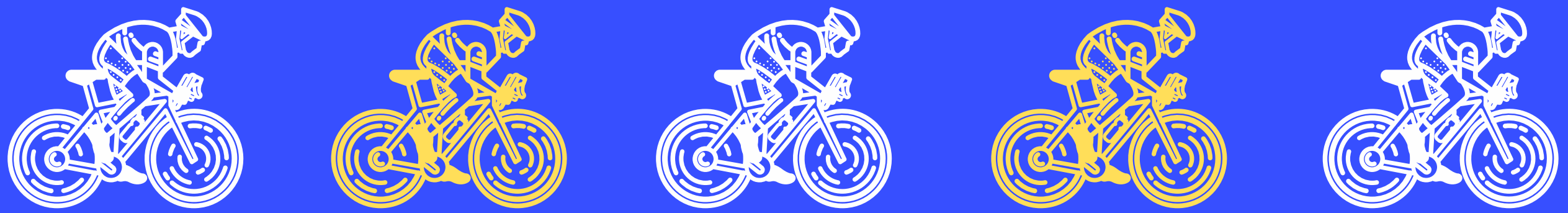
My goal was to include as many new JS features as early as possible...

- ✅ Arrow Functions
- ✅ String Template Literals
- ✅ Let & Const
- ✅ For...of
- ✅ For...in
- ✅ Exponent Operator
- ✅ String.includes()
- ✅ Array.includes()
- ✅ Object.values()

- ☑ Rest & Spread
- ☑ Destructuring
- ☑ Default Function Params
- ☐ Object Enhancements
- ☐ C
- ☐ As

# GOALS

- Work with rest & spread
- Add default function parameters
- Understand 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
```

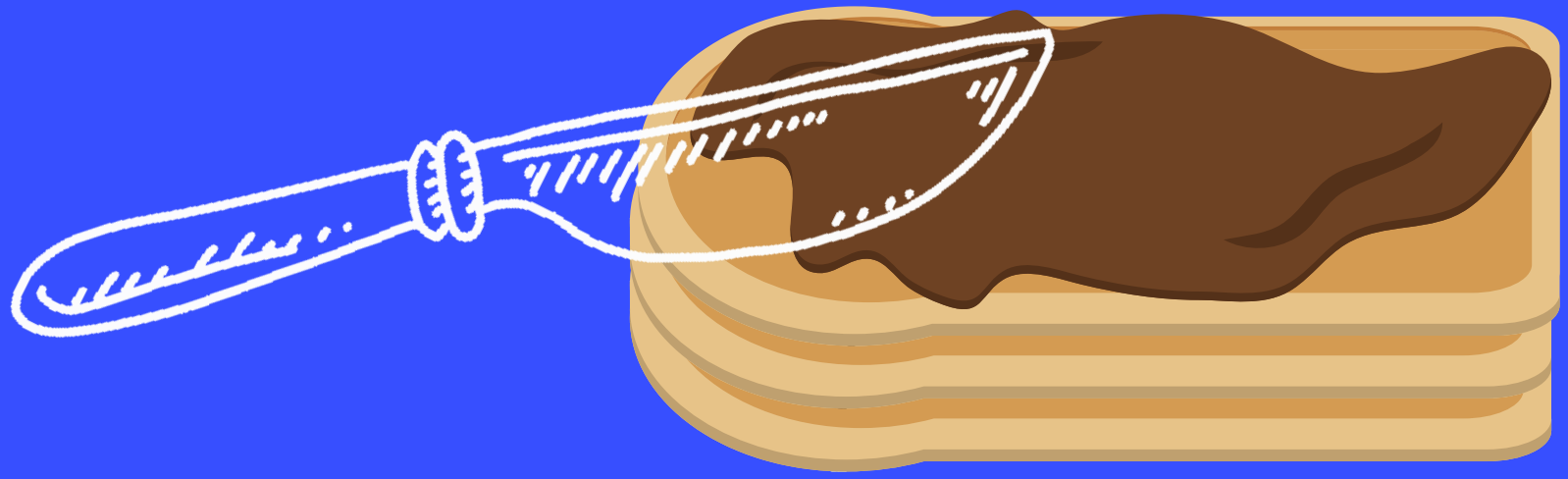
# DEFAULT PARAMS

## The New Way

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

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

# SPREAD



# 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.

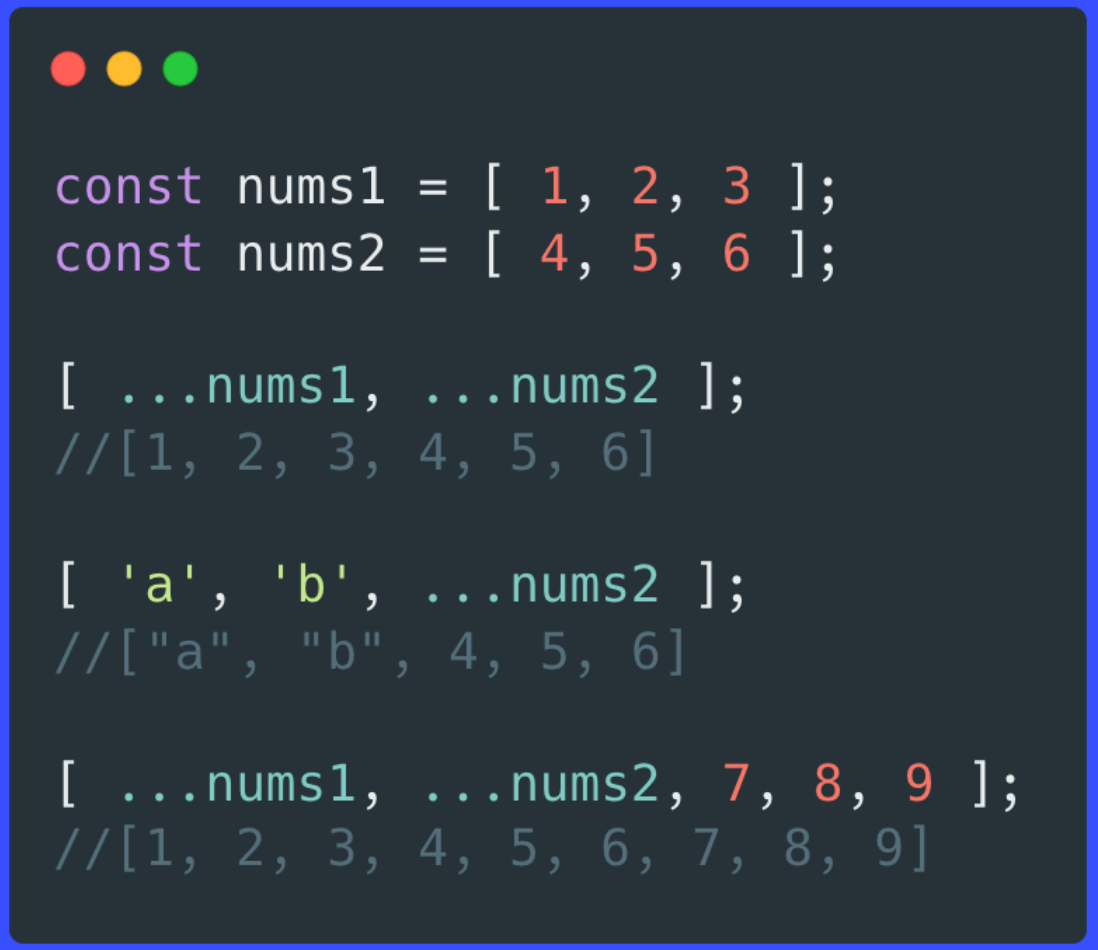

WHAT!?

# SPREAD

## For Function Calls

```
const nums = [ 9, 3, 2, 8 ];
Math.max(nums); //NaN
// Use spread!
Math.max(...nums); //67
// Same as calling:
// Math.max(9,3,2,8)
```

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




# SPREAD

## In Array Literals

Create a new array using an existing array. Spreads the elements from one array into a new array.

# SPREAD In Object Literals

```
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}
```

Copies properties from one object into another object literal.

# REST

It looks like spread,
but it's not!

# 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
```

Using it is very... MEH

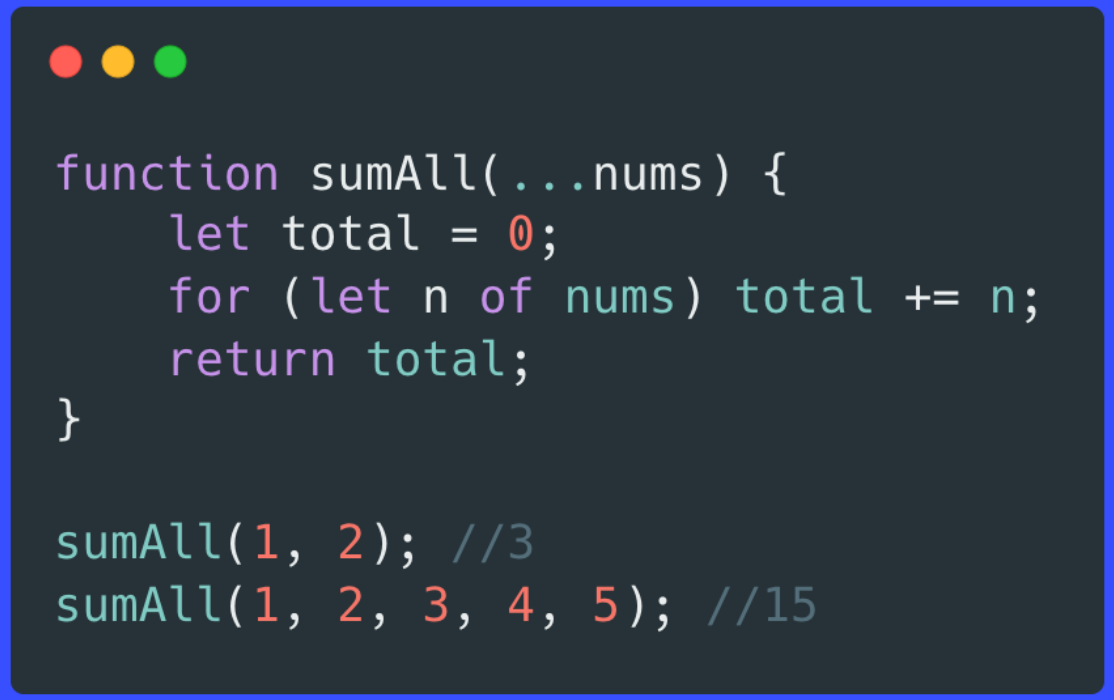
- Available inside every function.
- It's an **array-like** object
  - Has a length property
  - Does not have array methods like push/pop
- Contains all the arguments passed to the function
- Not available inside of arrow functions!

Thankfully...

# WE HAVE REST

# 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

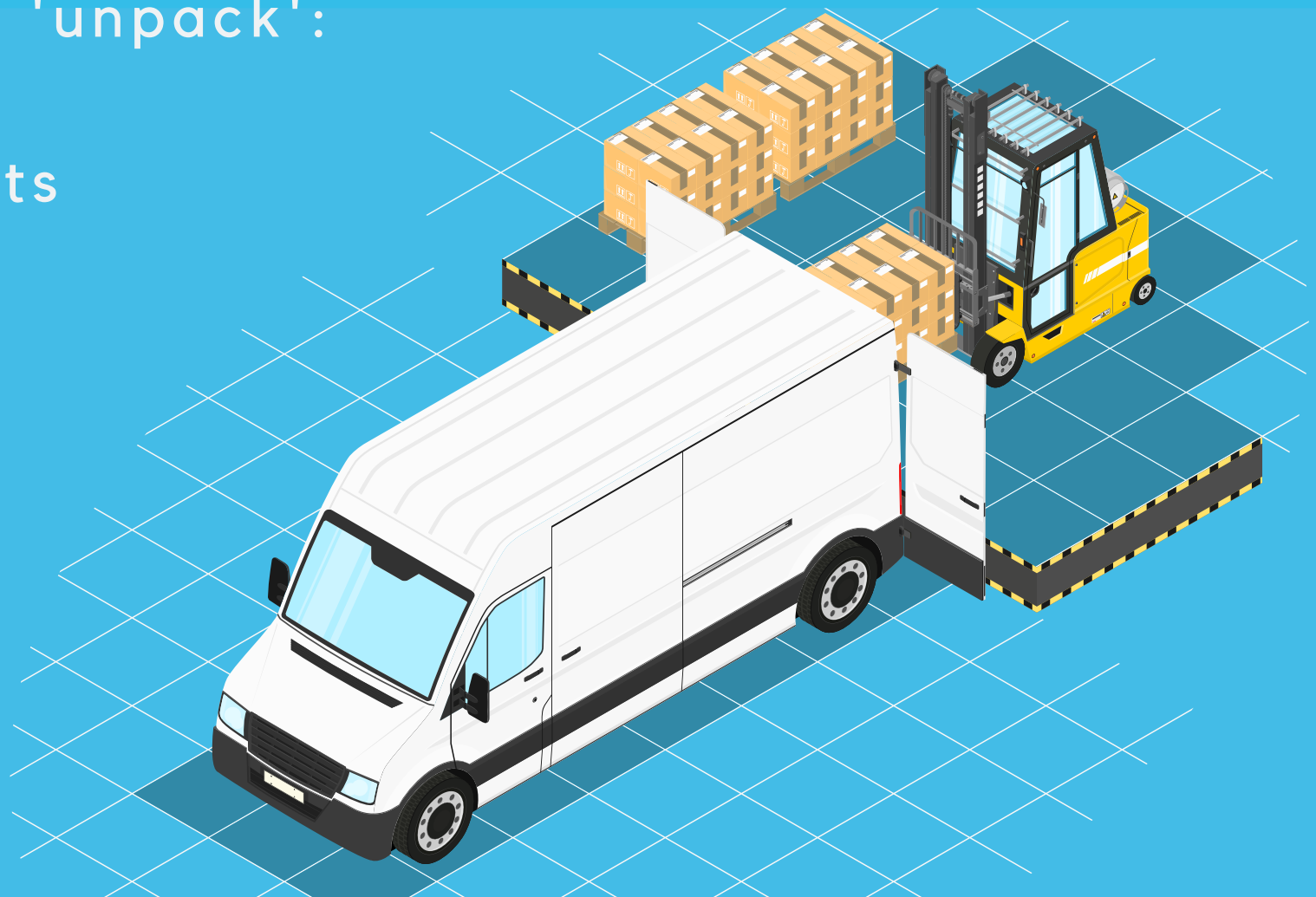
Into distinct variables.

# DESTRUCTURING

A short, clean syntax to 'unpack':

- Values from arrays
- Properties from objects

Into distinct variables.



# ARRAY
## Destructuring

```
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"]
```

# OBJECT
## Destructuring

```
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"
```

# PARAM

## Destructuring

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

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