Destructuring in Function Parameter Lists

The destructuring syntax explained in the previous lecture can also be used in **function parameter lists**.

For example, if a function accepts a parameter that will **contain an object** it can be destructured to *"pull out"* the object properties and make them available as **locally scoped variables** (i.e., variables only available inside the function body).

Here's an example:

```
1. function storeOrder(order) {
2.   localStorage.setItem('id', order.id);
3.   localStorage.setItem('currency', order.currency);
4. }
```

Instead of accessing the `order` properties via the *"dot notation"* inside the `storeOrder` function body, you could use destructuring like this:

```
1. function storeOrder({id, currency}) { // destructuring
2.   localStorage.setItem('id', id);
3.   localStorage.setItem('currency', currency);
4. }
```

The destructuring syntax is the same as taught in the previous lecture - just without creating a constant or variable manually.

Instead, `id` and `currency` are *"pulled out"* of the incoming object (i.e., the object passed as an argument to `storeOrder`).

It's very important to understand, that `storeOrder` **still only takes one parameter** in this example! It does **not** accept two parameters. Instead, it's one single parameter - an **object** which then just is destructured internally.

The function would still be called like this:

```
1. storeOrder({id: 5, currency: 'USD', amount: 15.99}); // one argument
   / value!
```