A recurring trend in ES6 is to remove unnecessary repetition in your code. By removing unnecessary repetition, your code becomes easier to read and more concise. This trend continues with the introduction of new *shorthand* ways for initializing objects and adding methods to objects.

Let’s see what those look like.

**Object literal shorthand**

You’ve probably written code where an object is being initialized using the same property names as the variable names being assigned to them.

But just in case you haven’t, here’s an example.

**let** type = 'quartz';

**let** color = 'rose';

**let** carat = 21.29;

**const** gemstone = {

type: type,

color: color,

carat: carat

};

console.log(gemstone);

***Prints:****Object {type: "quartz", color: "rose", carat: 21.29}*

Do you see the repetition? Doesn't type: type, color: color, and carat:carat seem redundant?

The good news is that you can remove those duplicate variables names from object properties \_if\_ the properties have the same name as the variables being assigned to them.

Check it out!

Replay

Mute

Loaded: 100.00%

Remaining Time -0:00

1x

Playback Rate

Picture-in-PictureFullscreen

If object properties have the same name as the variables being assigned to them, then you can drop the duplicate variable names.  
  
This video does not have audio. It was created as a visual to aid learning.

Speaking of shorthand, there’s also a shorthand way to add methods to objects.

To see how that looks, let’s start by adding a calculateWorth() method to our gemstone object. The calculateWorth() method will tell us how much our gemstone costs based on its type, color, and carat.

**let** type = 'quartz';

**let** color = 'rose';

**let** carat = 21.29;

**const** gemstone = {

type,

color,

carat,

calculateWorth: **function**() {

*// will calculate worth of gemstone based on type, color, and carat*

}

};

In this example, an anonymous function is being assigned to the property calculateWorth, but is the **function** keyword *really* needed? In ES6, it’s not!

**Shorthand method names**

Since you only need to reference the gemstone’s calculateWorth property in order to call the function, having the function keyword is redundant, so it can be dropped.

**let** gemstone = {

type,

color,

carat,

calculateWorth() { ... }

};