**Symbols**

A **symbol** is a unique and immutable data type that is often used to identify object properties.

To create a symbol, you write Symbol() with an optional string as its **description**.

**const** sym1 = Symbol('apple');

console.log(sym1);

*Symbol(apple)*

This will create a unique symbol and store it in sym1. The description "apple" is just a way to describe the symbol, but it can’t be used to access the symbol itself.

And just to show you how this works, if you compare two symbols with the same description…

**const** sym2 = Symbol('banana');

**const** sym3 = Symbol('banana');

console.log(sym2 === sym3);

*false*

…then the result is false because the description is *only* used to describe the symbol. It’s not used as part of the symbol itself—each time a new symbol is created, regardless of the description.

Still, this can be hard to wrap your head around, so let’s use the example from the previous video to see how symbols can be useful. Here’s the code to represent the bowl from the example.

**const** bowl = {

'apple': { color: 'red', weight: 136.078 },

'banana': { color: 'yellow', weight: 183.15 },

'orange': { color: 'orange', weight: 170.097 }

};

The bowl contains fruit which are objects that are properties of the bowl. But, we run into a problem when the second banana gets added.

**const** bowl = {

'apple': { color: 'red', weight: 136.078 },

'banana': { color: 'yellow', weight: 183.151 },

'orange': { color: 'orange', weight: 170.097 },

'banana': { color: 'yellow', weight: 176.845 }

};

console.log(bowl);

*Object {apple: Object, banana: Object, orange: Object}*

Instead of adding another banana to the bowl, our previous banana is overwritten by the new banana being added to the bowl. To fix this problem, we can use symbols.

**const** bowl = {

[Symbol('apple')]: { color: 'red', weight: 136.078 },

[Symbol('banana')]: { color: 'yellow', weight: 183.15 },

[Symbol('orange')]: { color: 'orange', weight: 170.097 },

[Symbol('banana')]: { color: 'yellow', weight: 176.845 }

};

console.log(bowl);

*Object {Symbol(apple): Object, Symbol(banana): Object, Symbol(orange): Object, Symbol(banana): Object}*

By changing the bowl’s properties to use symbols, each property is a unique Symbol and the first banana doesn’t get overwritten by the second banana.