Object literal upgrades

This lesson covers the upgrade made to object literal in ES6.

WE'LL COVER THE FOLLOWING

- Deconstructing variables into keys and values
- Add functions to our Objects
- Dynamically define properties of an object

In this lessons, we'll look at the many upgrades brought by ES6 to the object literal notation.

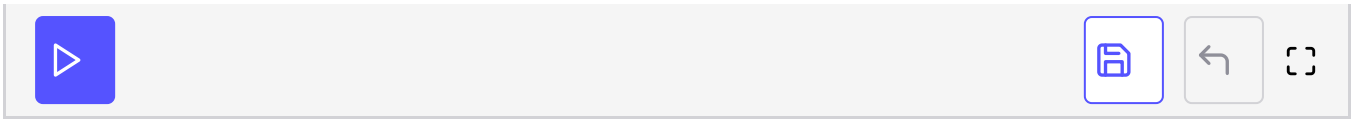
Deconstructing variables into keys and values

This is our initial situation:

```
const name = "Alberto";
const surname = "Montalesi";
const age = 25;
const nationality = "Italian";
```

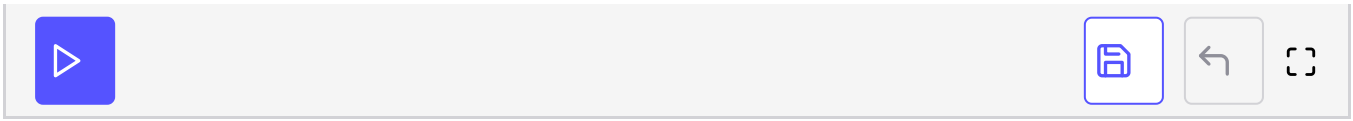
Now if we wanted to create an object literal this is what we would usually do:

```
const name = "Alberto";
const surname = "Montalesi";
const age = 25;
const nationality = "Italian";

const person = {
  name: name,
  surname: surname,
  age: age,
  nationality: nationality,
}

console.log(person);
```



In ES6 we can simplify like this:

```
const name = "Alberto";
const surname = "Montalesi";
const age = 25;
const nationality = "Italian";

const person = {
  name,
  surname,
  age,
  nationality,
}
console.log(person);
// {name: "Alberto", surname: "Montalesi", age: 25, nationality: "Italian"}
```

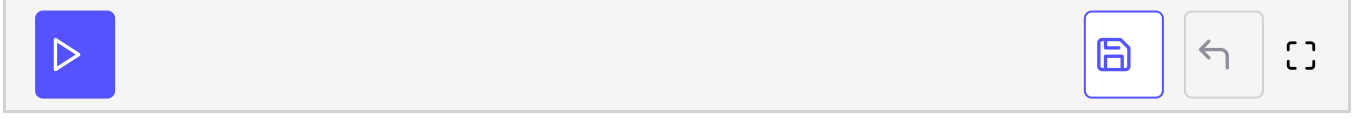


As our `const` are named the same way as the properties we are using, we can reduce our typing.

Add functions to our Objects

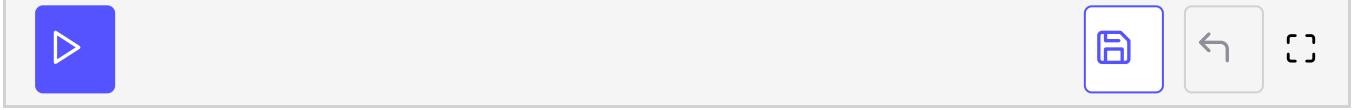
Let's look at an example from ES5:

```
const person = {
  name: "Alberto",
  greet: function(){
    console.log("Hello");
  },
}

person.greet();
// Hello
```



If we wanted to add a function to our object, we had to use the `function` keyword. In ES6 it got easier, look here:

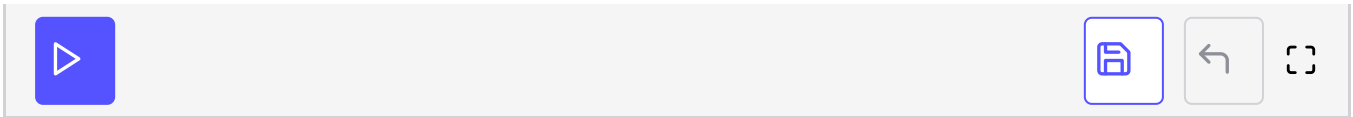
```
const person = {
  name: "Alberto",
  greet(){
    console.log("Hello");
  },
}

person.greet();
// Hello;
```



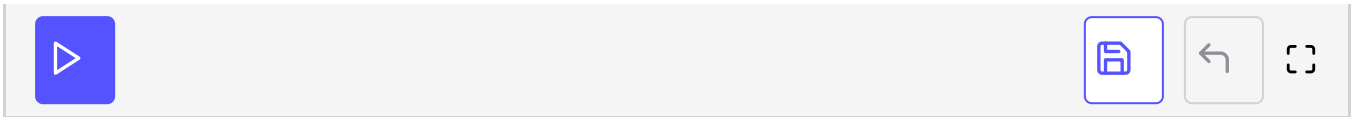
No more `function`, it's shorter and it behaves the same way.

Remember that **arrow functions** are anonymous, look at this example:

```
// this won't work, you need a key to access the function
const person1 = {
  () => console.log("Hello"),
};
```



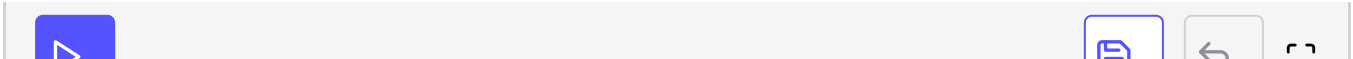
```
const person2 = {
  greet: () => console.log("Hello"),
}
person2.greet()
// Hello
```



Dynamically define properties of an object

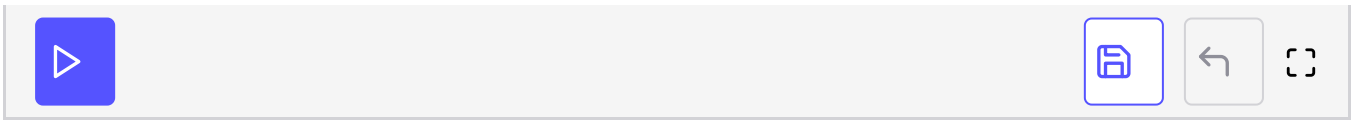
This is how we would dynamically define properties of an object in ES5:

```
var name = "myname";
// create empty object
var person = {}
// update the object
person[name] = "Alberto";
console.log(person.myname);
// Alberto
```



In the example given above, first we created the object and then we modified it. However, in ES6 we can do both things at the same time. Take a look at the following example:

```
const name = "myname";
const person = {
  [name]:"Alberto",
};
console.log(person.myname);
// Alberto
```



Now onto the quiz.