Immutable Data with Readonly

This lesson explains the 'readonly' mapped type.

```
WE'LL COVER THE FOLLOWING
The Object.freeze function
Adding generic using mapped
```

The Object.freeze function

One well-known JavaScript function that has a mapped type already backed inside TypeScript is the <code>Object.freeze</code>. The role of this function is to take a type and return everything as <code>read-only</code>. The function will have a return type of <code>Readonly<T></code> where <code>T</code> is the interface to freeze. The name 'mapped' comes from the fact that within the type there is an instruction with the keyword <code>in</code> which will loop through all the object properties and prototype chains.

```
// The interface that we want to have readonly without defining another interface
interface OriginalInterface {
    x: number;
    y: string;
}

// The mapped type that map a generic type with the readonly constraint
type ReadonlyInterface<T> = { readonly [P in keyof T]: T[P] };
```

The return type uses the lookup type which is a semantic way to indicate that the return type is not the original generic type T passed, but an aggregate of all properties.

Adding generic using mapped

Here is a simplistic version of the same example as before with read-only but generic using mapped type.

```
// The interface that we want to have readonly without defining another interface
interface OriginalInterface {
    x: number;
    y: string;
}

// The mapped type that map a generic type with the readonly constraint
type ReadonlyInterface<T> = { readonly [P in keyof T]: T[P] };

// Function that convert the object from one type to another
function genericInterfaceToReadOnly<T>(o: T): ReadonlyInterface<T> {
    return Object.freeze(o);
}

const original: OriginalInterface = { x: 0, y: "1" };
const originalReadonly = genericInterfaceToReadOnly(original);
// originalReadonly.x = 3; // error TS2540: Cannot assign to 'x'
```







[]

The mapped type <code>ReadonlyInterface<T></code> is a one-line definition that loops all its properties and returns a <code>readonly</code> version of the property. We do not need to create this interface because <code>TypeScript</code> comes with a set of predefined mapped types. One of these maps already in the toolbox is "Readonly". Actually, the <code>Object.freeze</code> of the previous object returns a <code>Readonly<T></code>. The code above works because our mapped map is an exact copy of the <code>Readonly</code>.

In the next mapped type lesson, we will see how to use a pre-built mapped type that comes with TypeScript.