Optional: IIFEs

In JavaScript - especially in older scripts - you sometimes find a pattern described as "**IIFEs**". IIFE stands for "**I**mmediately **I**nvoked **F**unction **E**xpression" and the pattern you might find looks like this (directly in a script file):

```
(function() {
   var age = 30;
   console.log(age); // 30
})()

console.log(age); // Error: "age is not defined"
```

What's that?

We see a function expression which calls itself (please note the `()` right after the function).

It's NOT a function declaration because it's wrapped in `()` - that happens on purpose since you can't immediately execute function declarations.

But why would you write some code?

Please note that the snippet uses `var`, NOT `let` or `const`. Remember that `var` does **NOT use block scope** but only differ between global and function scope.

As a consequence, it was hard to control where variables were available - variables outside of function always were available globally. Well, IIFEs solve that problem since the script (or parts of it) essentially are wrapped in a function => Function scope is used.

Nowadays, this is not really required anymore. With `let` and `const` we got block scope and if you want to restrict where variables are available (outside of functions, `if` statements, `for` loops etc - where you automatically have scoped variables since these structures create blocks), you can simply wrap the code that should have scoped variables with `{}`.

```
{
        const age = 30;
        console.log(age); // 30
    }

    console.log(age); // Error: "age is not defined"
```
Not something you see too often but something that is possible.