# Modules in JavaScript

→ Highly self contained and have their own dedicated functionality

→ We generally have multiple JS files for our whole project

→ So we need to be able to import functionali-ty that we want and export the same

Before diving in let's see what ways we already know?

① We can have functions (they can have their own variables), but for shared data we will need global variables (which is bad right?)

② We can have multiple script files (if they have global variables with the same name, it's going to override it)

@codeWithSimran

Modules give us a better way to organise variables and functions so when we group these together it makes sense

Where do modules come?

@codeWithSimran

$$\begin{tikzcd}
\text{Global scope} \arrow[dr] & & & \\
& \text{Module scope} \arrow[dr] & & \\
& & \text{Function scope} \arrow[dr] & \\
& & & \text{Block Scope}
\end{tikzcd}$$

## Ways to achieve module scope

Before

① IIFE (Immediatly invoked function expression)

Let's say want to wrap your entire JS content inside an IIFE

```
var module 1 = (function() {
        var name = 'Simran';
          :
        return {
            name : name
        }
}();
```

anything you want other files to have access, you can return and keep the rest private to the IIFE

This is called the revealing module pattern

You only reveal (<u>return</u>) what you want from the module

The other file that now wants to use variable name can simply do

<u>module1.name</u>

@codeWithSimran

<u>Note</u>: after wrapping your code in an IIFE it's no longer a part of the global scope and we're not polluting the global scope.

But what about the variable <u>module1</u> ?

Well it's still a global variable

- ☆ So we can still override module1 in another file (problem 1)
- ☆ We need to keep track of all the dependencies.

For example

@codewithSimran

```
<script src='./script1.js>
<script src = './script2.js>
<script src= './script3.js>
```

Now if script1 and script3 use a function sayName that is defined inside script-2, script1 wont have access to it (because script 2 is included after script 1) and we'll get an error.

Summary

@codewithSimran

2 problems

① Still a global variable (module variable)
② need to maintain the dependencies