# Module pattern

Projetos grandes precisam ser organizados em unidades coesas para manter a organização. Existem várias formas de se fazer isso, mas a mais aceitável é o **module pattern**.

Basicamente para se criar um módulo precisamos de uma função anônima que é imediatamente executada após a sua declaração:

|  |  |
| --- | --- |
| 1 2 3 4 5 | var nomeDoModulo = (function () {   /\* TODO \*/  })(); // executa imediatamente após a declaração |

O module pattern usa o conceito de [object literals](http://rmurphey.com/blog/2009/10/15/using-objects-to-organize-your-code/) para organizar o código e, dessa forma, usando [closures](http://en.wikipedia.org/wiki/Closure_%28computer_programming%29) (funções encadeadas) é possível criar o encapsulamento de propriedades e métodos.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 | var moduloEstacionamento = (function () {   // objetos privados  var patio = [];  var deletaCarro = function (carro) {  index = patio.indexOf(carro);  if (index > -1) {  patio.splice(index, 1);  }  };    // objetos públicos  return {  lista: patio,  entrar: function (carro) {  patio.push(carro);  },  sair: function (carro) {  deletaCarro(carro);  }  }; })(); |

No module pattern, como mostrado no exemplo acima, deixamos todos os objetos **públicos** dentro do return, fazendo com que tudo o que esteja fora dele seja **privado**. Além do mais, criamos um **alias** para a propriedade privada patio, que pode ser acessada publicamente pelo nome lista.

Dessa forma podemos usar o módulo moduloEstacionamento do seguinte modo:

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 | belina = {  "modelo" : "belina",  "placa" : "abc-1234" };  // mostra os objetos públicos do moduloEstacionamento console.log( moduloEstacionamento );  // estaciono a belina moduloEstacionamento.entrar( belina );  // mostra [{ "modelo" : "belina", "placa" : "abc-1234" }] console.log( moduloEstacionamento.lista );  // removo a belina moduloEstacionamento.sair( belina );  // mostra [] console.log( moduloEstacionamento.lista );  // mostra undefined console.log( moduloEstacionamento.patio ); |

##### As desvantagens de se usar o Module Pattern são:

* é um pouco desgastante mudar a privacidade de um objeto
* não é possível realizar testes unitários em objetos privados