

Javascript

Trabalhando com Objetos

Estruturação de Objetos

- A partir de funções

```
1 var MyClass = function (param){
2   this.attr1 = 0; //publico
3   var attr2 = 1; //privado
4 }
5
```

- De forma literal

```
6 var myLiteralObject = {
7   attr1: 0, //publico
8   attr2: 1 //publico
9 }
10
```

- Usando o **Prototype** de um classes definida

```
10
11 //inclusão de um atributo publico
12 MyClass.prototype.attr3 = 2
13
```

Instanciação de Objetos 1/4

- A partir das funções

```
1
2  var MyClass = function (param){
3      this.attr1 = 0; //publico
4      var attr2 = 1; //privado
5  }
6
7  //instanciação
8  var myClass = new MyClass();
9  // imprime 0
10 console.info(myClass.attr1);
11
```

Instanciação de Objetos 2/4

- A partir da estruturação literal

```
12
13 // estruturação com
14 // auto instanciação
15 var myLiteralObject = {
16     attr1: 0, //publico
17     attr2: 1 //publico
18 }
19
20 // imprime 0
21 console.info(myLiteralObject.attr1);
22
```

Instanciação de Objetos 3/4

- A partir de classes prototipadas

```
23
24  var MyClass = function (){
25  }
26
27  //inclusão de um atributo publico
28  MyClass.prototype.attr1 = 0;
29  //instanciação
30  var myClass = new MyClass();
31  // imprime 0
32  console.info(myClass.attr1);
33
```

Instanciação de Objetos 4/4

- A partir da auto-invocação de função

```
33
34 // definição da classe
35 var MyClass = function (){}
36 // instanciação de MyClass
37 var myClass = new MyClass();
38 // reestruturando myClass
39 // e auto instanciando
40 var myObject = (function(obj){
41     obj.attr1 = 0;
42     return obj;
43 })(myClass);
44 // imprime 0
45 console.info(myObject.attr1);
46
```

Definição de Métodos 1/2

- Os métodos podem ser adicionados em tempo de estruturação ou em tempo de execução.

Exemplo #1:

```
48
49  var MyClass = function (param){
50      this.method1 = function(){}; //publico
51      var method2 = function(){}; //privado
52  }
53
```

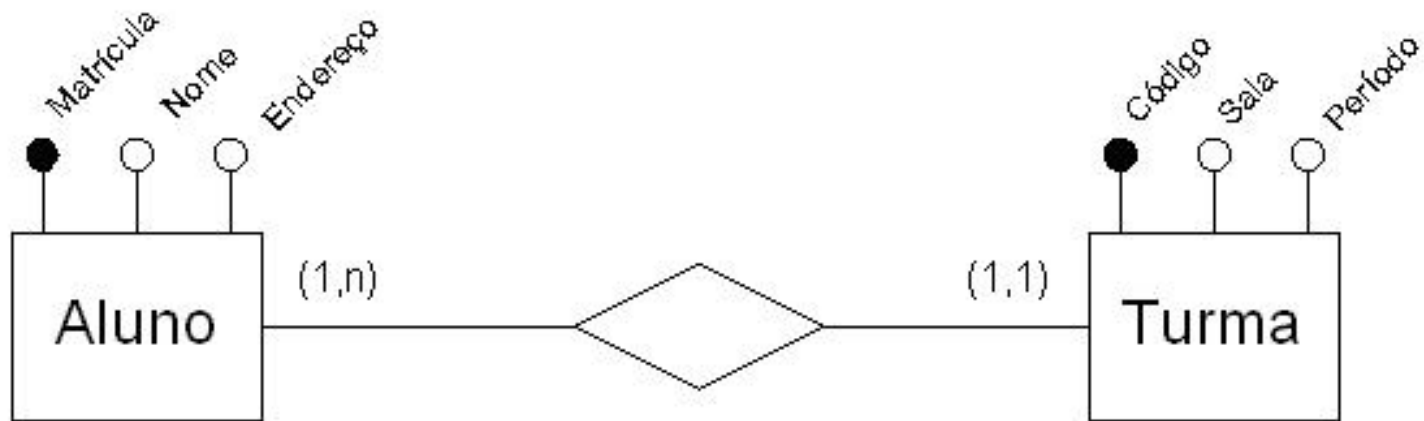
Definição de Métodos 2/2

- Exemplo #2:

```
55
56 // estruturação com
57 // auto instanciação
58 var myLiteralObject = {
59     attr1: 0, //publico
60     method: function(){
61         return "my first method"
62     }//publico
63 }
64
65 // imprime my first method
66 console.info(myLiteralObject.method());
67
68
```


Exercício #1

- Considerando as entidades representadas no MER abaixo, crie as respectivas estruturas de objetos Javascript utilizando o método de estruturação a partir de função.




Exercício #2

- Considerando as classes Java ao lado, crie as respectivas estruturas de objetos Javascript utilizando o método literal.

```
1
2 public class MyClass1{
3     private String attr1 = "0";
4     private Integer attr2 = 1;
5     public String getAttr1(){
6         return attr1;
7     }
8     public String getAttr2(){
9         return String.valueOf(attr2);
10    }
11 }
12
13 public class MyClass2{
14     private MyClass1 attr1 = new MyClass1();
15     public String getAttr1(){
16         return attr1.getAttr1();
17     }
18 }
19
```

Exercício #3

- Crie um objeto Javascript (com estruturação por função ou literal) que recupere o primeiro e segundo número ao clicar no botão e devolva o valor como uma mensagem de alerta.



The image shows a web form titled "Determinação do maior de dois números" in teal. It contains two input fields for numbers, labeled "Primeiro número" and "Segundo número". To the right of these fields is a button labeled "Encontrar o maior número".

Determinação do maior de dois números

Primeiro número

Segundo número

Encontrar o maior número

Resposta para Exercício #3 (parte 1)

```
3      <script type="text/javascript">
4          var Formulario = function(form){
5              //guardando referencia interna
6              var _this = this;
7              //metodo para calcular
8              this.calcula = function(a, b){
9                  if (a > b)
10                     return a;
11                 else
12                     return b;
13             };
14             //evento de submissão
15             this.onsubmit = function(event){
16                 event.preventDefault();
17                 var a = form.primeiro.value;
18                 var b = form.segundo.value;
19                 var r = _this.calcula(a, b);
20                 alert(r);
21             };
22             //informa que está preparado
23             this.done = function(){
24                 form.onsubmit = _this.onsubmit;
25                 console.info("Pronto para usar")
26             };
27         }
28     </script>
```

Resposta para Exercício #3 (parte 2)

```
31     <form>
32         <div>
33             <label>Primeiro número: </label>
34             <input name="primeiro">
35         </div>
36         <div>
37             <label>Segundo número: </label>
38             <input name="segundo">
39         </div>
40         <div>
41             <button type="submit">Encontrar o maior número</button>
42         </div>
43     </form>
44     <script type="text/javascript">
45         // recuperar o elemento
46         var fel = document.getElementsByTagName("form")[0];
47         // criar um objeto Formulário
48         var fobj = new Formulário(fel);
49         fobj.done();
50     </script>
```

Exercício #4

- Crie objetos Javascript (com qualquer estruturação) que possam capturar informações de um formulário e adicionar em uma lista, conforme a UI abaixo.

Cadastro de Clientes

Nome:	<input type="text"/>
Cidade:	<input type="text"/>
Endereço:	<input type="text"/>
Bairro:	<input type="text"/>
<input type="button" value="Cadastrar"/> <input type="button" value="Limpar"/>	

Código	Nome	Cidade	Endereço	Bairro	Excluir	Alterar
28	cvds	cs	cs	cs	Excluir	Alterar

AngularJS em 5 minutos

- Trabalha com o conceito de MVC
 - (view) a interface do usuário é considerada no formato de Template
 - (controller) responsável por coordenar a interação entre a visão (view) e o modelo (model)
 - (model) nível mais baixo da arquitetura, responsável por manipular os dados

AngularJS em 5 minutos

- AngularJS usa diretivas para controlar e estender HTML
- Usa atributos especiais
- Exemplos de diretivas:
 - **ng-app**: inicia um aplicativo angularjs.
 - **ng-init**: inicializa os dados do aplicativo.
 - **ng-model**: define a variável do modelo.
 - **ng-controller**: define a variável do controlador.
 - **ng-repeat**: repete elementos HTML para cada item em uma coleção.

AngularJS em 5 minutos

- Exemplo de uso:

```
<div ng-app="">
...
  <p>List of Countries with locale:</p>
  <ol>
    <li ng-repeat="country in countries">
      {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}
    </li>
  </ol>
</div>
```

AngularJS em 5 minutos (exemplo)

```
<html>
<title>AngularJS Expressions</title> <body>
  <h1>Sample Application</h1>
  <div ng-app=""
    ng-init="quantity=1;cost=30;student={firstname:'Mahesh',
      lastname:'Parashar',rollno:101};marks=[80,90,75 ,73,60]"
  >
    <p>Hello {{student.firstname + " " + student.lastname}}!</p>
    <p>Expense on Books : {{cost * quantity}} Rs</p>
    <p>Roll No: {{student.rollno}}</p>
    <p>Marks(Math): {{marks[3]}}</p>
  </div>
  <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.
    angular.min.js"> </script>
</body>
</html>
```

AngularJS em 5 minutos (resultado)

Sample Application

Hello Mahesh Parashar!

Expense on Books : 30 Rs

Roll No: 101

Marks(Math): 73