JavaScript Orientado a Objectos

O bom, o mau e uma solução.

Artur Ventura

Índice

JavaScript

- Criado em 1995 na Netscape (originalmente chamava-se LiveScript)
- Standardizado através de um dialecto chamado ECMAScript
- Actualmente existe um motor de JavaScript em qualquer browser.
- Dinâmica, fracamente tipificada (duck typing), baseada em protótipos, com funções de primeira classe.
- Inspirado por Self, C, Scheme, Perl, Python, Java.

Exemplo

JavaScript

```
function Foo (x,y){
    var arry = [1,2,3];
    var hash = { bing:'bang', ting:"tong" };
    hash.bing = hash['bing'];
    var f = function (){ return arguments[0] };
    x = f(x);
    for(var i=0; i<arry.length; i++){
        arry[i]++
    }
    for(i in hash){
        var value = hash[i];
    }
    return x + y;
}
```

JavaScript Orientado a Objectos

- Uma "classe" cria-se definindo um **object** , chamada protótipo que representa o estado inicial das instancias
- Uma instancia dessa "classe" é um "copia" desse **object**.
- Os métodos são elementos chaves desse **object.**

JavaScript Orientado a Objectos

```
function Rectangle(w,h){
    this.width = w;
    this.height = h;
}
var x = new Rectangle(10,10);
```

JavaScript Orientado a Objectos

```
var x = new Rectangle(10,10);
Rectangle.prototype.getArea = function (){
    return this.height * this.width;
}
x.getArea()
```

JavaScript Orientado a Objectos

Visibilidade

- Não existe qualquer tipo controlo de visibilidade na linguagem.
- No entanto é possível simular o comportamento da visibilidade privada através de ambientes.

JavaScript Orientado a Objectos

Visibilidade

```
function ImmutableRectangle(w, h) {
    function privateArea() {
        return w * h;
    }
    this.getWidth = function() { return w; }
    this.getHeight = function() { return h; }
    this.getArea = function() { return privateArea(); }
}
```

JavaScript Orientado a Objectos

Herança

- A herança é simulada copiando o protótipo da classe pai para a filha.

JavaScript Orientado a Objectos

Herança

```
function PositionedRectangle(x, y, w, h) {
    Rectangle.call(this, w, h);

    this.x = x;
    this.y = y;
}
PositionedRectangle.prototype = new Rectangle();
```

JavaScript Orientado a Objectos

Problemas

- Hacks de linguagem
- Single Namespace
- Falta de programação estruturada

JavaScript Orientado a Objectos

Problemas

```
var i = 0

racio = function (){
    return 1 + i;
}
```

JavaScript Orientado a Objectos

Problemas

```
var i = 0

racio = function (){
    return 1 + i;
}
```

```
>> racio()
21 // ?????
>> i
0
>> racio.toString()
function (){
    return 1 + i;
}
>>
```

JavaScript Orientado a Objectos

Problemas

```
var i = 0

racio = function (){
    return 1 + i;
}
```

```
>> racio()
21 // ?????
>> i
0
>> racio.toString()
function (){
    return 1 + i;
}
>>
```

```
racio = (function (){
    var i=20;
    return function (){
        return 1 + i;
    }
})()
```

JavaScript Orientado a Objectos

Problemas

```
var mypackage = (function (){
    return {
        i: 0,
        racio: function(){
            return 1 + this.i;
        }
    }
})()
```

```
...

var racio = mypackage.racio

var x = racio()
var y = racio()
var z = racio()
alert (x + " " + y + " " + z);
```

JavaScript Orientado a Objectos

Problemas

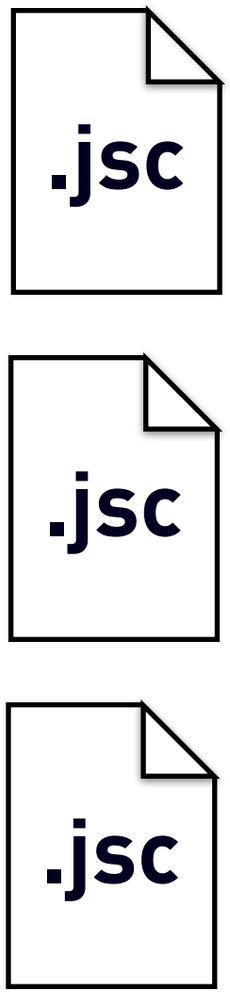
```
var mypackage = (function (){
    return {
        i: 0,
        racio: function(){
            return 1 + this.i;
        }
    }
})()

...

var racio = mypackage.racio

var x = racio()
var y = racio()
var z = racio()
alert (x + " " + y + " " + z);
```

NaN NaN NaN

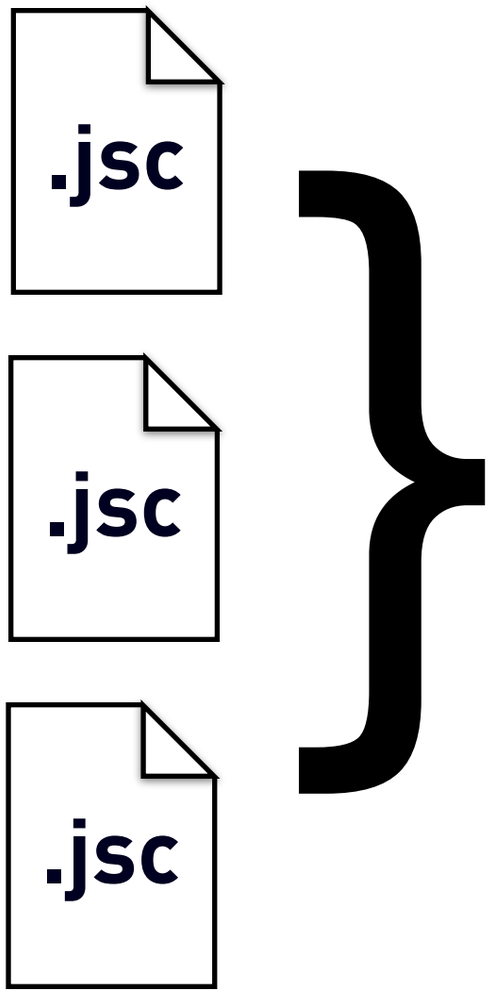
JSC

- Extensão ao JavaScript para permitir programação OO estruturada.
- Criar mecanismos de isolamento de código em pacotes e classes.
- Introspecção e de Intercepção mantendo o código organizado.
- Herança múltipla (baseada em protótipos).
- Acordos de implementação (interfaces do Java, ou protocolos do Objective-C).
- Simplificar o uso de RMI.

JSC

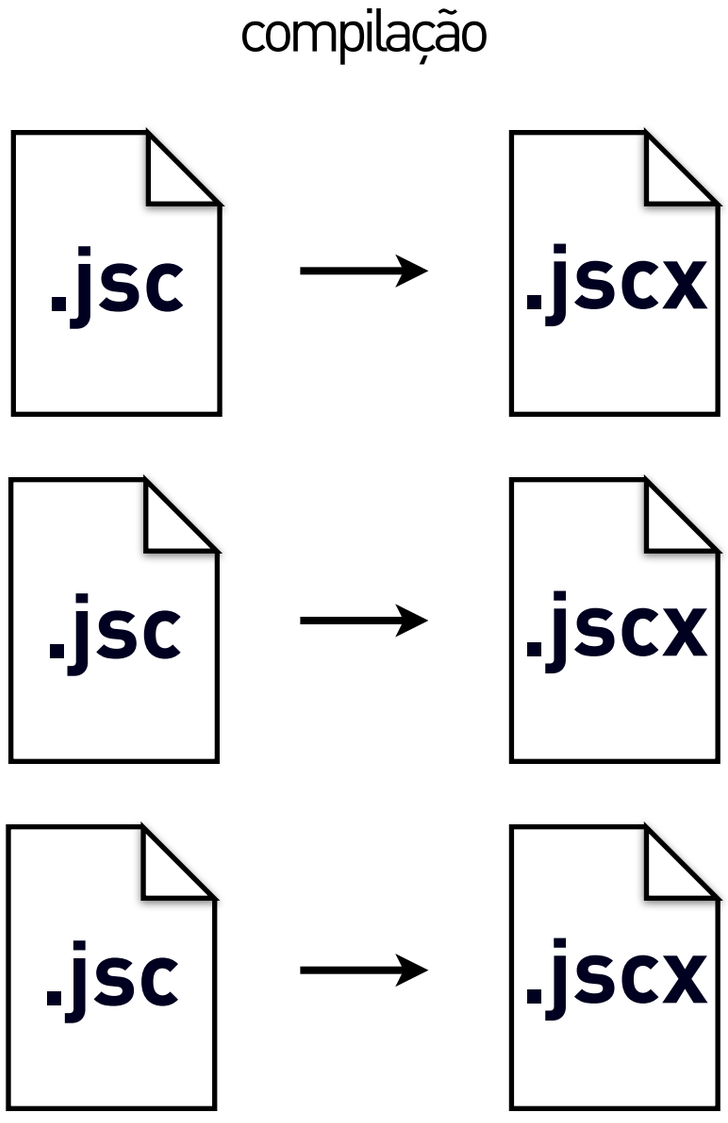


JSC

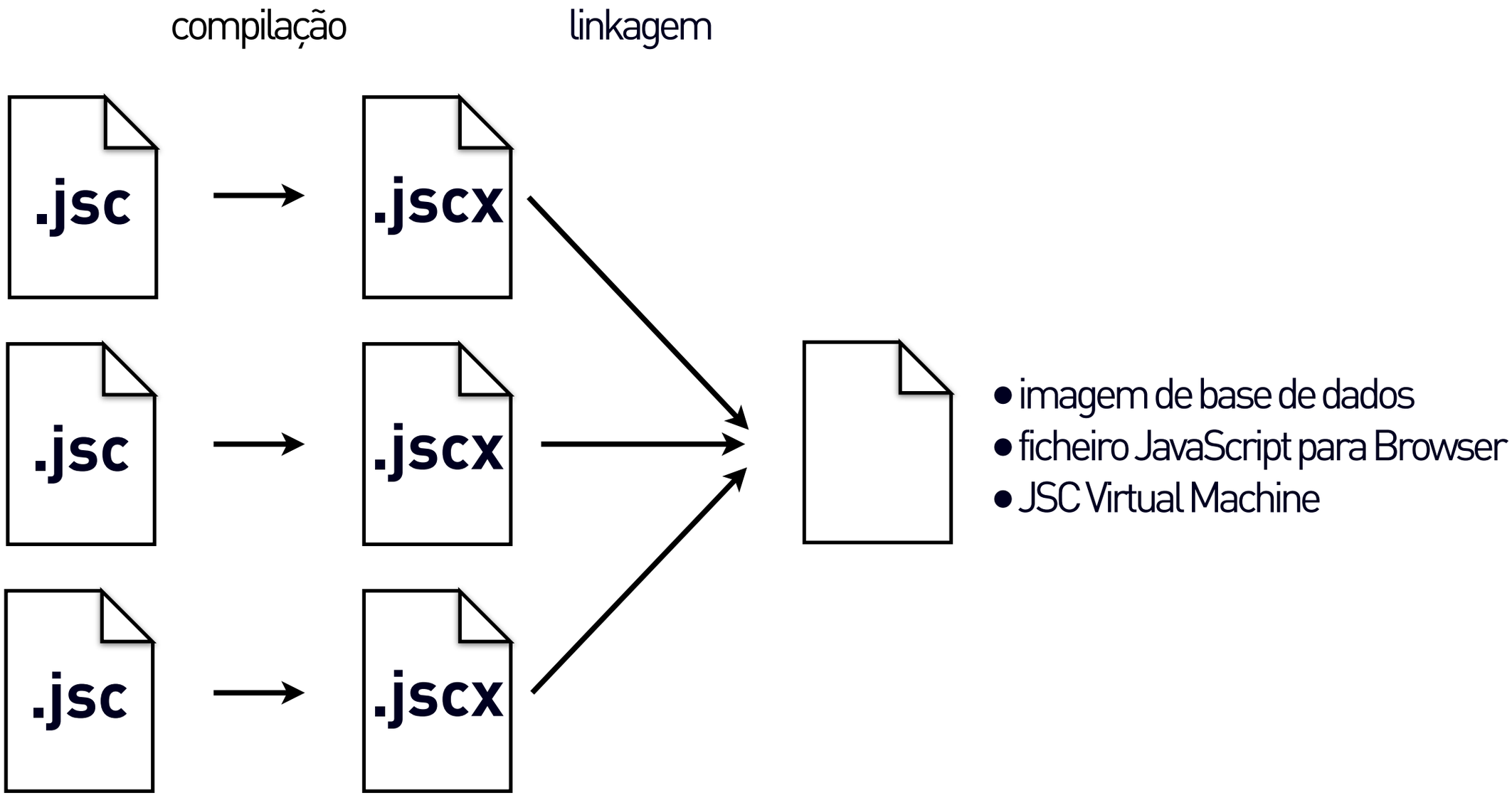


Cada ficheiro representa uma classe.

JSC



JSC


compilação
linkagem
.jsc
.jscx
.jsc
.jscx
.jsc
.jscx
• imagem de base de dados
• ficheiro JavaScript para Browser
• JSC Virtual Machine

JSC

Definição

```
package UI.Component;

class Rectangle{
   slots:[height,width],
   Rectangle: function (w,h){
      this.setHeight(h);
      this.setWidth(w);
    },
   getArea: function (){
   return this.getHeight() * this.getWidth();
}
```

JSC

Definição

```
package UI.Component;


class Rectangle{
   slots:[height,width],
   Rectangle: function (w,h){
      this.setHeight(h);
      this.setWidth(w);
    },
   getArea: function (){
   return this.getHeight() * this.getWidth();
}
```

JSC
Definição

```
package UI.Component;

class Rectangle{
   slots:[height,width],
   Rectangle: function (w,h){
      this.setHeight(h);
      this.setWidth(w);
    },
   getArea: function (){
   return this.getHeight() * this.getWidth();
}
```

JSC

Definição

```
package UI.Component;

class Rectangle{
   slots:[height,width],
   Rectangle: function (w,h){
      this.setHeight(h);
      this.setWidth(w);
    },
   getArea: function (){
   return this.getHeight() * this.getWidth();
}
```

JSC

Definição

```
package UI.Component;

class Rectangle{
   slots:[height,width],
   Rectangle: function (w,h){
      this.setHeight(h);
      this.setWidth(w);
    },
   getArea: function (){
   return this.getHeight() * this.getWidth();
}
```

JSC
Instanciação

```
Class("UI.Component.Rectangle").create(10,10)
```

JSC
Herança

```
package UI.Component;

class PositionedRectangle extends UI.Component.Rectangle{
   slots:[x,y],
   PositionedRectangle: function (x,y,w,h){
      Class("UI.Component.Rectangle").init(this,w,h);
      this.setX(x);
      this.setY(y);
   }
}
```

JSC
Herança

```
package UI.Component;

class PositionedRectangle extends UI.Component.Rectangle,
                                  UI.mixin.RectangleDimensions{
   slots:[x,y],
   PositionedRectangle: function (x,y,w,h){
      Class("UI.Component.Rectangle").init(this,w,h);

      this.setX(x);
      this.setY(y);
   }
}
```

JSC

Métodos de classe

- Estes métodos são invocados através da instancia da classe.
- Conceito semelhante aos métodos **static** do Java

JSC

Métodos de classe

```
package Main;

class App {
   static:{
      main: function (args){
      ...
      }
   }
}
```

JSC
Protocolos

- Os protocolos garantem a existência de um conjunto de métodos.
- Os métodos nos protocolos podem ser declarados como obrigatórios, que obrigam a classe a implementar esses métodos, ou facultativos em que é garantido a existência de um método caso um não seja declarado.

JSC
Protocolos

```
package UI.Component;

protocol Draggable {
   element: true,
   eventListener: false
}
```

JSC
Problemas

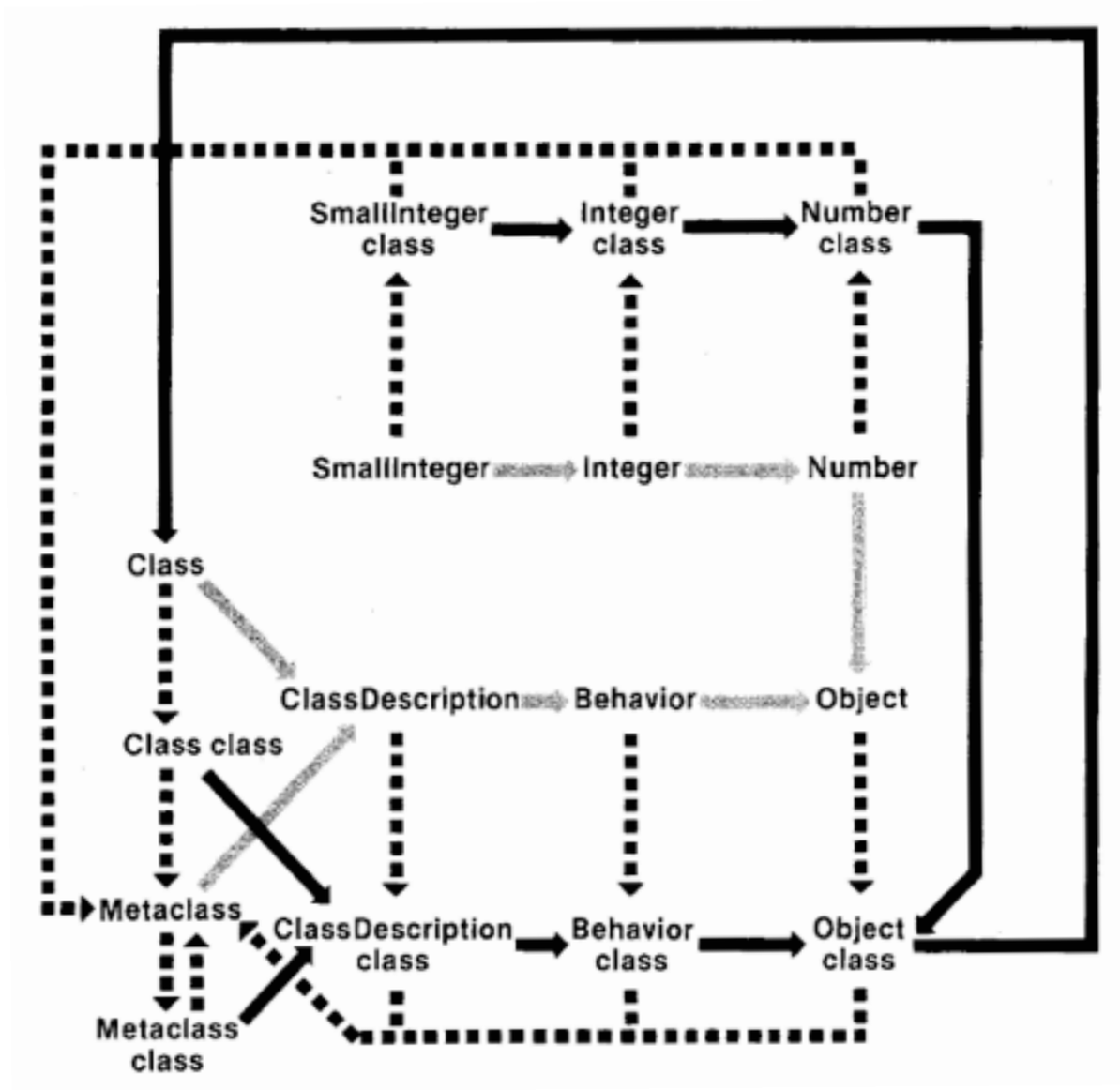
- Variáveis globais.

```
function Foo(){
   var local = 1;
   global = 1;
}
```

- Valor por omissão de um slot.

```
class Foo{
   slots:{
      aSlot:{ getter:"getSlot", setter:"setIt", default:1 },
      anotherSlot: { default: Class("Baz.Bing").create(1,2) }
   }
}
```

- Separação de classe / metaclasse.



JSC

Demo

Questões?