

# Programação na Internet

Turma i52d

Aula 3

Objectos e Funções

# Objectos

Em Java ou .Net

Objectos têm membros:

- Campos
- Métodos
- Propriedades (getters e setters => só em .Net)

Em Javascript

Objectos têm:

- Propriedades ( ⇔ campos)

Propriedades podem armazenar qq coisa:

Valores, Objectos, Funções

# Funções

1<sup>st</sup> class objects => têm todos os privilégios de um objecto

Função = Objecto + Capacidade de ser chamada.

# Untyped

Não existe um representante de tipo como **Type** em .Net ou **Class** em Java

**Prototype** => tem características “semelhantes”

`obj.constructor.prototype`

`constructor` e `prototype` são propriedades implícitas de `Object`

# Como partilhar um método?

```
function Student(nr) { this.nr = nr }  
  
var student = new Student(798)  
  
student.print = function() { console.log('I am a Studen with nr: ' + this.nr ) }  
student.print() // I am a Studen with nr: 798  
  
var student2 = new Student(666)  
student2.print() // ?????? Erro em tempo de execução => chamar um método sobre null
```

- Opção 1: definir na função construtora!

# E se não tivermos acesso ao construtor?

- Extension Methods ? E.g. em C# são processados em compilação
- Através do prototype
  - => partilhado por todos os objectos que tenham o mesmo prototype.

```
Student.prototype.print = function() { console.log('I am a Studen with nr: ' + this.nr ) }
```

Em tempo de execução!!!!

- ????? Imaginem fazer o mesmo em C# ??? ILGenerator.Emit(...)

“Mais do que reflexão” => Meta-programação (instrumentação)

# Reflexão / Introspecção em Javascript

Reflexão a capacidade de um AVE para inspecionar em TE

Introspecção é a acção feita com suporte de uma API de Reflexão

E.g. .Net System.Reflection e em Java java.lang.reflect

C#:

- GetFields() ou GetMethods() etc...
- **FieldInfo** representante do campo
- FieldInfo.GetValue(target) => valor
- Invoke(object target, object[] args)

Javascript - Não há distinção entre membros - são propriedades:

- for(var key in objRef) – o nome da propriedade
- key é uma **String** que é o nome da propriedade
- target[key] => valor
- target[key]()

# TPC

function inspect(obj):

- Escreve na console os pares nome valor de propriedades de obj:  
nome-prop = valor
- Se a propriedade for um método (função) apresentamos o resultado da chamada à função.
  - Adicional: só para métodos sem parametros. Os métodos com parametros são ignorados.

=> Colocar num repositório vosso Github e adicionar fmcavalho