PROGRAMAÇÃO ORIENTADA A OBJETOS

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

POLITÉCNICO
DO PORTO
ESCOLA
SUPERIOR
DE MEDIA ARTES E
DESIGN

POLITÉCNICO DO PORTO
ESCOLA SUPERIOR DE MEDIA ARTES E DESIGN

P.PORTO

PROGRAMAÇÃO ORIENTADA A OBJETOS

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

2017-18

MÓDULO III – ORIENTAÇÃO A OBJETOS

ÍNDICE

1. OBJETOS

DEFINIÇÃO

- INSTÂNCIA QUE CONTÉM UM CONJUNTO DE PARES DE VALORES-CHAVE
- PODEM REPRESENTAR VALORES MÚLTIPLOS E MUDAR AO LONGO DE SEU TEMPO DE VIDA
- VALORES PODEM SER ESCALARES, FUNÇÕES OU MESMO ARRAYS DE OUTROS OBJETOS
- SINTAXE:

```
let identifier = {
  key1: value,
  key2: function () {
      // funções
  },
  key3: [“conteudo1”, ”conteudo2”]
}
```

- O CONTEÚDO DE UM OBJETO É CHAMADO DE **PROPRIEDADES** (OU MEMBROS)
- AS PROPRIEDADES CONSISTEM NUM **NOME (OU CHAVE) E VALOR**
  - **NOMES** DEVEM SER STRINGS OU SÍMBOLOS
  - **VALORES** PODEM SER DE QUALQUER TIPO (INCLUINDO OUTROS OBJETOS)

1. OBJETOS

DEFINIÇÃO

- EM JAVASCRIPT, TODOS OS VALORES (EXCETUANDO OS PRIMITIVOS) SÃO OBJETOS!
- VALORES PRIMITIVOS: STRINGS ("RUI SILVA"), NÚMEROS (3.14), BOOLEANOS (TRUE, FALSE), NULL, E UNDEFINED

```
let fullName = “Rui Silva”
```

- OBJETOS SÃO VARIÁVEIS, MAS
  - CONTÊM VÁRIAS **PROPRIEDADES** SEPARADAS POR VÍRGULAS (,)
  - CADA PROPRIEDADE ESCRITA COMO PAR **"NOME: VALOR"** SEPARADOS POR DOIS PONTOS (:)
  - INSTANCIAÇÃO DE OBJETOS:

```
let person = {
  firstName : “Rui”,
  age       : 50
}
```

  - POR EXEMPLO, **FIRSTNAME** É UMA PROPRIEDADE COM O VALOR **"RUI“**
  - OBJETOS PODEM SER VAZIOS

```
let person = { }
```

1. OBJETOS

COMPARAÇÃO DE OBJETOS

- EM JAVASCRIPT, OS OBJETOS SÃO UM **TIPO DE REFERÊNCIA**
- DOIS OBJETOS DISTINTOS **NUNCA SÃO IGUAIS**, MESMO QUE TENHAM AS MESMAS PROPRIEDADES
- ISSO É PORQUE, ELES APONTAM PARA UM ENDEREÇO DE MEMÓRIA COMPLETAMENTE DIFERENTE
- SOMENTE OS OBJETOS QUE PARTILHAM UMA REFERÊNCIA COMUM SÃO VERDADEIROS NA COMPARAÇÃO

```
let num = 2
let str = "2"

console.log(num == str)
console.log(num === str)

--------------------------
true
false
```

```
let val1 = {name: "Rui"}
let val2 = {name: "Rui"}

console.log(val1 == val2)
console.log(val1 === val2)

--------------------------
false
false
```

```
let val1 = {name: "Rui"}
let val2 = val1

console.log(val1 == val2)
console.log(val1 === val2)

--------------------------
true
true
```

1. OBJETOS

DESESTRUTURAÇÃO DE OBJETOS

- O TERMO **DESESTRUTURAÇÃO** REFERE-SE A QUEBRAR A ESTRUTURA DE UMA ENTIDADE
- USANDO DESESTRUTURAÇÃO PODE-SE EXTRAIR DADOS DE ARRAYS OU OBJETOS EM VARIÁVEIS DISTINTAS

```
let emp = {name: ”João”, age: 22}
let {name, age} = emp

console.log(name)
console.log(age)

----------------------------------
João
22
```

ÍNDICE

1. OBJETOS

CRIAÇÃO DE OBJETOS

- 3 FORMAS:
  1. CRIAR UM ÚNICO OBJETO ATRAVÉS DE UM **LITERAL OBJETO**
  2. CRIAR UM ÚNICO OBJETO ATRAVÉS DA PALAVRA-CHAVE **NEW**
  3. DEFINIR UM CONSTRUTOR DE OBJETO, E DEPOIS CRIAR OBJETOS DO TIPO DO CONSTRUTOR

1. OBJETOS

CRIAÇÃO DE OBJETOS

- 3 FORMAS:
  1. **CRIAR UM ÚNICO OBJETO ATRAVÉS DE UM LITERAL OBJETO**

  - LITERAL OBJETO É UMA LISTA DE PARES NOME:VALOR DENTRO DE {}
  - SIMPLES E LEGÍVEL
  - CRIAÇÃO DO OBJETO NUMA ÚNICA DECLARAÇÃO

```
let person = {firstName : “Rui”, lastName : “Silva”, age : 50, eyeColor : “azul” }
// OU
let person = {
  firstName : “Rui”,
  lastName  : “Silva”,
  age       : 50,
  eyeColor  : “azul”
}
```

  - CRIAÇÃO DE OBJETOS S PARTIR DE LITERAL OBJETO EXISTENTE

```
let person1 = Object.create(person)
let person2 = Object.create(person)
```

1. OBJETOS

CRIAÇÃO DE OBJETOS

- 3 FORMAS:
  1. **CRIAR UM ÚNICO OBJETO ATRAVÉS DE UM LITERAL OBJETO**

```
let firstName = “Rui”
let age = 50

let person = {
  firstName : firstName,
  age       : age
}
```


```
let firstName = “Rui”
let age = 50

let person = { firstName, age }
```

1. OBJETOS

CRIAÇÃO DE OBJETOS

- 3 FORMAS:
  2. **CRIAR UM ÚNICO OBJETO ATRAVÉS DA PALAVRA-CHAVE NEW**

  - USO DA PALAVRA-CHAVE **NEW**

```
let person = new Object()
person.firstName = “Rui”
person.lastName = “Silva”
person.age = 50
person.eyeColor = “azul”
```

1. OBJETOS

CRIAÇÃO DE OBJETOS

- 3 FORMAS:
  3. **DEFINIR UM CONSTRUTOR DE OBJETO, E DEPOIS CRIAR OBJETOS DO TIPO DO CONSTRUTOR**

    - "TIPO DE OBJETO" (CLASSE) QUE PODE SER USADO PARA CRIAR MUITOS OBJETOS DESSE TIPO
    - USO DE **FUNÇÃO CONSTRUTORA** DO OBJETO

```
// Função construtora
function Person(first, last, age, eye) {
  this.firstName = first
  this.lastName = last
  this.age = age
  this.eyeColor = eye
}

// Criação de 2 objetos a partir da função construtora
let myFriend = new Person("Rui", "Silva", 50, "azul")
let myBoss = new Person("Joana", "Vieira", 48, "verde")
```

ÍNDICE

1. OBJETOS

PROPRIEDADES

- UM OBJETO JAVASCRIPT É UMA COLEÇÃO DE **PROPRIEDADES** DESORDENADAS
- AS PROPRIEDADES GERALMENTE PODE SER ADICIONADAS, ALTERADAS E REMOVIDAS
- SINTAXE: **OBJETO.PROPRIEDADE**

```
function Person(first, last) {
  this.firstName = first,
  this.lastName = last
}
let myFriend = new Person("Rui", "Silva")

// Acesso à propriedade lastName
console.log(myFriend.lastName)
----------------------------------------
Silva
```

1. OBJETOS

PROPRIEDADES

- SINTAXES ALTERNATIVAS:
  - OBJETO.PROPRIEDADE
  - OBJETO[“PROPRIEDADE”]
  - OBJETO[EXPRESSÃO]

```
function Person(first, last) {
  this.firstName = first,
  this.lastName = last
}
let myFriend = new Person(“Rui”, “Silva”)

console.log(myFriend.lastName)
console.log(myFriend[“lastName”])
let x = “lastName”
console.log(myFriend[x])
------------------------------------------------------------
Silva
Silva
Silva
```

1. OBJETOS

PROPRIEDADES

- **ITERAÇÃO**
  - A DECLARAÇÃO **FOR ... IN** PERCORRE AS PROPRIEDADES DE UM OBJETO

```
for (variable in object) {
  // Iterar sobre as propriedades do objeto
}
```

  - O Nº DE ITERAÇÕES DO CICLO É IGUAL AO Nº DE PROPRIEDADES

```
let person = {firstName : "Rui", lastName : "Silva", age : 50}
let txt = ""
for (let x in person) {
  txt = "nome: " + x + " valor: " + person[x] + "\n"
}

console.log(txt)
-------------------------------------------------------------
nome: firstName valor: Rui
nome: lastName valor: Silva
nome: age valor: 50
```

1. OBJETOS

PROPRIEDADES

- **ADIÇÃO DE NOVAS PROPRIEDADES**
  - PODE-SE ADICIONAR NOVAS PROPRIEDADES A UM OBJETO EXISTENTE, BASTA DAR-LHE UM VALOR

```
function Person(first) {
  this.firstName = first
}

let myFriend1 = new Person("João")
let myFriend2 = new Person("Maria")

// Adiciona a propriedade age apenas ao objeto MyFriend1
myFriend1.age = 32

console.log(myFriend1.age)
console.log(myFriend2.age)
-------------------------------------------------------
32
undefined
```

1. OBJETOS

PROPRIEDADES

- **REMOÇÃO**
  - A PALAVRA-CHAVE **DELETE** ELIMINA UMA PROPRIEDADE DE UM OBJETO

```
let person = {firstName : "Rui", age : 50}
delete person.age
console.log(person.age)
-----------------------------------------
undefined
```

  - APÓS A REMOÇÃO, A PROPRIEDADE NÃO PODE SER UTILIZADA ANTES DE SER ADICIONADA NOVAMENTE

1. OBJETOS

PROPRIEDADES

- **COMPUTED PROPERTIES**
  - DEFINIÇÃO DE NOMES DE PROPRIEDADES À CUSTA DO VALOR DE UMA VARIÁVEL

```
let feature = “color”
let car = {
  plate : “12-SA-23”,
  [feature] : “azul”
}

console.log(car.color)
---------------------------
azul
```

ÍNDICE

1. OBJETOS

MÉTODOS

- MÉTODOS SÃO AÇÕES QUE PODEM SER EXECUTADAS EM OBJETOS
- UM MÉTODO JAVASCRIPT É UMA PROPRIEDADE QUE CONTÉM UMA DEFINIÇÃO DE FUNÇÃO

```
let person = {
  firstName : “Rui”,
  lastName  : “Silva”,
  fullName  : function() {
    return this.firstName + “ “ + this.lastName
  }
}

// Invocação do método
console.log(person.fullName())
-----------------------------------------------
Rui Silva
```

1. OBJETOS

MÉTODOS

- SINTAXE MAIS COMPACTA NA VERSÃO 6 DO JAVASCRIPT

```
let person = {
  firstName : “Rui”,
  lastName  : “Silva”,
  fullName  : function() {
    return this.firstName + “ “ + this.lastName
  }
}
```


```
let person = {
  firstName : “Rui”,
  lastName  : “Silva”,
  fullName () {
    return this.firstName + “ “ + this.lastName
  }
}
```

1. OBJETOS

MÉTODOS

- **ADIÇÃO DE NOVOS MÉTODOS**
  - EXEMPLO COM ATRIBUIÇÃO DE REFERÊNCIAS A MÉTODOS:

```
function personFullNameReversed() {
  return this.lastName + ', ' + this.firstName;
}

function Person (first, last) {
  this.firstName = first,
  this.lastName  = last,
  this.personFullNameReversed = personFullNameReversed
}

let myFriend = new Person("João", "Silva")

console.log(myFriend.personFullNameReversed())
----------------------------------------------------------
Silva, João
```

1. OBJETOS

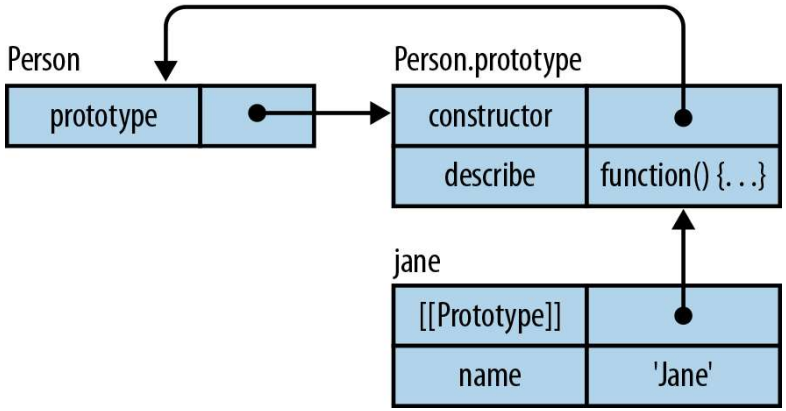
MÉTODOS

- **USO DO PROTOTYPE**
  - TODOS OS OBJETOS JAVASCRIPT HERDAM DE **OBJECT.PROTOTYPE**
  - USO DA PALAVRA-CHAVE **PROTOTYPE**

```
function Person (name) {
  this.name = name
}

Person.prototype.describe = function () {
  return “O meu nome é ” + this.name
}

let jane = new Person(“Jane”)
console.log(jane.decribe())
----------------------------------------
O meu nome é Jane
```



  - PODE-SE
    - MODIFICAR ALGO PROTOTIPADO EM QUALQUER MOMENTO DO PROGRAMA
    - ADICIONAR MÉTODOS EXTRA EM OBJETOS PREEXISTENTES, EM TEMPO DE EXECUÇÃO!

1. OBJETOS

MÉTODOS

- **USO DO PROTOTYPE**
  - POSSIBILIDADE DE ADICIONAR PROPRIEDADES/MÉTODOS AOS PROTÓTIPOS DOS OBJETOS BUILT-IN DO JS
  - EXEMPLO: ADICIONAR UM MÉTODO AO OBJETO STRING QUE DEVOLVE UMA STRING INVERTIDA

```
let str = “Rui”
str.reversed() // Erro: não existe esse método para o objeto String
String.prototype.reversed = function () {
  let r = “”
  for(let i = this.length - 1; i >= 0; i--) {
    r += this[i]
  }
  return r
}
console.log(str.reversed())
---------------------------------------------------------------------
iuR
```

ÍNDICE

1. OBJETOS

ITERAÇÃO SOBRE UM ARRAY DE OBJETOS


```
let library = [
  {
    title: 'The Road Ahead',
    author: 'Bill Gates',
    readingStatus: true
  },
  {
    title: 'Steve Jobs',
    author: 'Walter Isaacson',
    readingStatus: true
  },
  {
    title: 'Mockingjay: The Final Book of The Hunger Games',
    author: 'Suzanne Collins',
    readingStatus: false
  }
];
```

1. OBJETOS

ITERAÇÃO SOBRE UM ARRAY DE OBJETOS

- EXEMPLO (ESCREVER O TÍTULO DOS LIVROS LIDOS)

```
let library = [
  {
    title: 'The Road Ahead',
    author: 'Bill Gates',
    readingStatus: true
  },
  ...
]
let result = “”
for (let i = 0; i < library.length; i++) {
  if(library[i].readingStatus) {
    result += library[i].title + “\n";
  }
}

console.log(result)
-------------------------------------------
The Road Ahead
Steve Jobs
```

ÍNDICE

2. CLASSES

SINTAXE

- DESCRIÇÃO QUE ABSTRAI UM CONJUNTO DE OBJETOS COM CARACTERÍSTICAS SIMILARES
- EM JS6 PODE-SE USAR A KEYWORD **CLASS**

```
class MyClass {
  constructor(...) {
    // ...
  }
  method1(...) {}
  method2(...) {}
  get something(...) {}
  set something(...) {}
  static staticMethod(...) {}
  // ...
}
```

EXPRESSÃO DE CLASSE

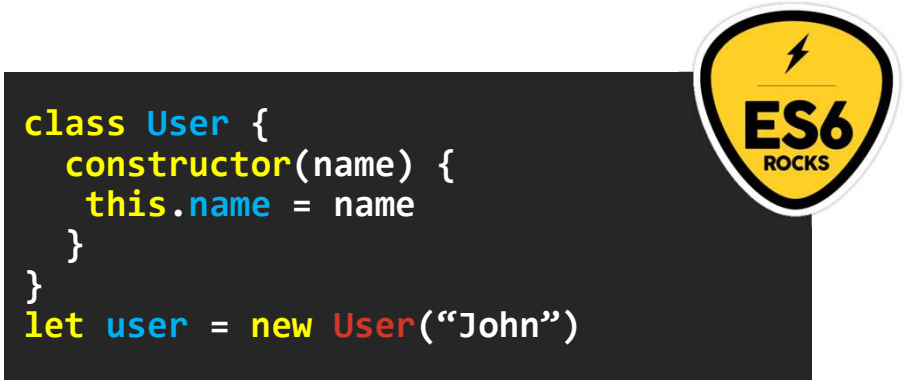
```
let MyClass = class {
  // ...
}
```

- CONSTRUÇÃO DE UM OBJETO (USANDO A PALAVRA-CHAVE **NEW**) CHAMA O MÉTODO ESPECIAL **CONSTRUCTOR**
- MÉTODOS LISTADOS NA CLASSE SÃO MEMBROS DE SEU PROTÓTIPO
- EXCEÇÃO PARA OS **MÉTODOS ESTÁTICOS**
  - ESCRITOS NA PRÓPRIA FUNÇÃO
  - INVOCADOS COMO **MYCLASS.STATICMETHOD()**
  - USADOS QUANDO SE PRECISA DE FUNÇÃO VINCULADA À CLASSE, MAS NÃO A UM OBJETO DESSA CLASSE

2. CLASSES

SINTAXE

- CLASSES ENCAPSULAM ABSTRAÇÕES DE DADOS E PROCEDIMENTOS QUE DESCREVEM O CONTEÚDO E O COMPORTAMENTO DE ENTIDADES DO MUNDO REAL, REPRESENTADAS POR OBJETOS
- COMPARAÇÃO ENTRE **FUNÇÕES CONSTRUTORAS** E **CLASSES** (IGUAIS EM TERMOS DE RESULTADO)



2. CLASSES

SINTAXE

- O OPERADOR **INSTANCEOF**
  - RETORNA **TRUE** SE O OBJETO PERTENCE AO TIPO ESPECIFICADO
  - CASO CONTRÁRIO RETORNA **FALSE**

```
class Person { }
let obj = new Person()
let isPerson = obj instanceof Person
console.log("obj é uma instância de Person? " + isPerson)
---------------------------------------------------------
obj é uma instância de Person? true
```

ÍNDICE

2. CLASSES

PROPRIEDADES

- PROPRIEDADES DEFINIDAS COM
  - FUNÇÕES ACESSORAS (GETTERS) – KEYWORD **GET**
  - FUNÇÕES MODIFICADORAS (SETTERS) – KEYWORD **SET**

```
class User {
  constructor(name) {
   // invoca o setter
   this._name = name
  }

  get name() {
   return this._name
  }

  set name(value) {
   if(value.length < 4) {
    console.log("nome muito curto")
    return
   }
   this._name = value
  }
}
```

```
let user = new User("John")
console.log(user.name)
---------------------------------------
John
```

```
let user = new User("")
---------------------------------------
nome muito curto
```

ÍNDICE

2. CLASSES

MÉTODOS

- MÉTODOS DEFINIDOS NO INTERIOR DA CLASSE
- NÃO É NECESSÁRIO EXPLICITAMENTE
  - ADICIONAR MÉTODOS AO **PROTOTYPE** (FAZ ISSO IMPLICITAMENTE!)
  - USAR A KEYWORD **FUNCTION**

JS

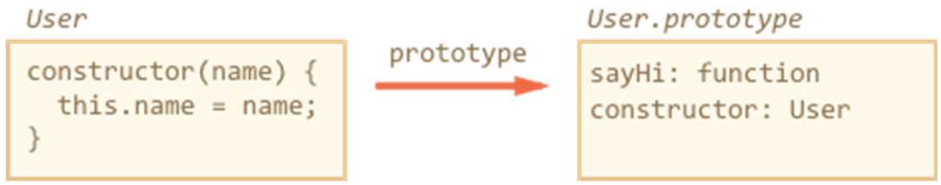
```
function User(name) {
  this.name = name
}

User.prototype.sayHi = function () {
  return this.name
}

let user = new User("John")
console.log(user.sayHi())
------------------------------------
John
```

ES6 ROCKS

```
class User {
  constructor(name) {
   this.name = name
  }
  sayHi() {
   return this.name
  }
}
let user = new User("John")
console.log(user.sayHi())
------------------------------------
John
```



ÍNDICE

2. CLASSES

MÉTODOS ESTÁTICOS

- NÃO REQUEREM UMA INSTÂNCIA DA CLASSE
- NÃO PODEM ACEDER IMPLICITAMENTE AOS DADOS (THIS) DE UMA QUALQUER INSTÂNCIA
- SINTAXE:
  - PALAVRA-CHAVE **STATIC** NO INÍCIO DA ASSINATURA DO MÉTODO
  - CHAMADA ATRAVÉS DE: **NOMEDACLASSE.MÉTODOESTÁTICO()**

```
class Article {
  constructor(title, date) {
   this.title = title
   this.date = date
  }
  static createTodays() {
   return new Article("Today's digest", new Date())
  }
}
let article = Article.createTodays()
console.log(article.title())
--------------------------------------------------
Today's digest
```

- PRINCIPAIS FUNÇÕES:
  - IMPLEMENTAR FUNÇÕES QUE PERTENCEM À CLASSE, MAS NÃO A NENHUM OBJETO PARTICULAR
  - MÉTODOS "FACTORY"