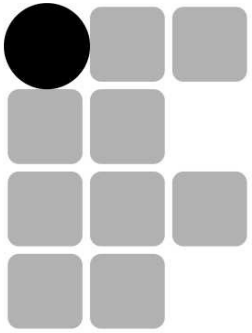

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO SUL
Câmpus Feliz

Linguagem JavaScript – Parte II

Prof. Moser Fagundes

Programação III
ADS

Sumário

- Eventos
- Funções
- Objetos
- Exercícios

Eventos

- Eventos são "coisas" que acontecem com os elementos HTML.
- Formato:

```
<elemento-HTML evento='JavaScript'>
```

- Eventos mais comuns:

```
onclick, onmouseover
onmouseout, onload, onfocus
onsubmit, onkeypress
```

Eventos

- Para reagir a estes eventos, usamos os chamados **atributos handlers** com a seguinte sintaxe:

```
<button onclick="minhaFuncao()">Texto<button>
```

- **Exemplo**:

  → ver **Exemplo-Slide-04A.html**

  → ver **Exemplo-Slide-04B.html**

Lista de eventos

https://www.w3schools.com/jsref/dom_obj_event.asp

Funções

- As funções no JavaScript são na verdade **objetos**.

- Podemos ter funções
  - *Declarativas*
  - *Anônimas*
  - *Literais*

Função Declarativa

Exemplo:

```
function multiplica(a, b) {
   return a * b;
}

var x = multiplica(2,3);

console.log(x);
```

→ ver **Exemplo-Slide-06.html**

Passagem por valor e por referência

- As variáveis baseadas em **tipos primitivos** (string, número e boolean) são passados por **valor**.
- Os **objetos** são passados por **referência**.

```
function altera(str,arr) {
   str = "Mudou";
   arr[1] = "dois";
}

var nome = "Feliz";
var numeros = ["one","two"];
altera(nome,numeros);
console.log(nome);
console.log(numeros);
```

→ ver **Exemplo-Slide-07.html**

Função Anônima

A função **anônima** pode ser criada com o construtor Function, onde o **último parâmetro** é o **corpo** da função, e os **demais parâmetros** são os **argumentos** da função.

Exemplo:

```
var f1 = new Function("x","y","return x*y");

var resultado = f1(4,3);

console.log(resultado);
```

→ ver **Exemplo-Slide-08.html**

Função Literal

É possível atribuir uma função a uma variável dado que uma função é um objeto em JavaScript.

Exemplo:

```
var x = function (a, b) {
   return a * b;
};
var y = x(4,3);
console.log(y);
```

→ ver **Exemplo-Slide-09.html**

Acesso a argumentos não declarados

Exemplo:

```
function find() {
    var i, mtemp = 0;
    for (i = 0; i < arguments.length; i++){
        if (arguments[i] > mtemp) {
            mtemp = arguments[i];
        }
    }
    return mtemp;
}
x = find(1, 123, 500, 115, 44, 88);
```

→ ver **Exemplo-Slide-10.html**

Exercício

- Considere uma função que recebe 3 parâmetros, **a, b** e **c**, que correspondem às medidas dos lados de um triângulo. A sua função deve determinar se um triângulo é **equilátero, isósceles** ou **escaleno**, ou se a figura geométrica **não é um triângulo**.

Objetos

- **Object** é o principal tipo de dados do JavaScript
- Em JavaScript "*quase tudo*" são objects.
- Um object é uma variável complexa com:
  - Propriedades
  - Métodos

Objetos

- Mesmo **tipos de dados primitivos** (exceto *null* e *undefined*) podem ser tratados como **objetos**:
  - Booleans podem ser objetos
  - Números podem ser objetos
  - Strings também são objetos
  - Dates (datas) são sempre objetos
  - Arrays são sempre objetos
  - Mesmo funções são sempre objetos

Propriedade de objetos

- **Propriedades** são valores associados com um objeto.
- Um **objeto** é coleção de propriedades não ordenadas.
- Cada propriedade possui **nome** e **valor**.
- A sintaxe para acessar uma propriedade é:

  `objeto.propriedade` ou `objeto["propriedade"]`

- **Por exemplo**, suponha um objeto `pessoa` com a propriedade `nome`

  `pessoa.nome` ou `pessoa["nome"]`

→ ver **`Exemplo-Slide-14.html`**

Métodos de objetos

- **Métodos** consistem em ações que podemos executar sobre os objetos.

- A sintaxe para chamar um método é:

```
objeto.metodo()
```

- **Por exemplo**, suponha um objeto `pessoa` que possui o método `imprimir`

```
passoa.imprimir()
```

→ ver **Exemplo-Slide-15.html**

Construção de objetos

- **Há três modos de criar objetos JavaScript**
  - Usando um objeto literal
  - Usando a palavra-chave **new**
  - Usando uma função como construtor de objetos

Construção de um objeto literal

- É a **maneira mais fácil** de criar um objeto.
- Usando um objeto literal, podemos **definir** e **criar** um objeto em uma única linha de código.
- Consiste em uma **lista de pares nome:valor**, como por exemplo idade:50, **entre parênteses** { }.
- **Por exemplo:**

```
var carro = {modelo:"Chevete", marca:"GM"};

var pessoa = {nome:"Luke", idade:20};
```

→ ver **Exemplo-Slide-17.html**

Construção de objetos com new

- Podemos criar objetos usando o operador **new**.

- **Exemplo:**

```
var jedi = new Object();
jedi.nome = "Obi-Wan";
jedi.sobrenome = "Kenobi";
jedi.poder = 90;
```

→ ver **Exemplo-Slide-18.html**

Usando uma função como construtor

- Os exemplos mostrados até agora são **limitados** no sentido que ele criam apenas **um objeto**.
- As vezes é mais interessante criar um **"tipo de objeto"** que possa ser usado muitas vezes

  "como se fosse uma classe"

- O modo padrão de criar um **"tipo de objeto"** é através de um **construtor** na forma de uma **função**.

- **Exemplo** no próximo slide...

Usando uma função como construtor

- **Exemplo:**

```
function jedi(n, s, p) {
   this.nome = n;
   this.sobrenome = s;
   this.poder = p;
}

var j1 = new jedi("Obi-Wan", "Kenobi", 90);
var j2 = new jedi("Aayla", "Secura", 88);
```

→ ver **Exemplo-Slide-20.html**

Palavra-chave this

- `this` **não** é uma variável, é uma palavra-chave.
- Você **não** pode modificar o valor de `this`.
- Quando usado em um objeto, o valor de `this` é o próprio objeto.
- A palavra-chave `this`, quando usada em um construtor, é uma substituta para o **novo objeto** que está sendo criado.

Exercícios

- Construa três objetos, cada um deles usando um modo de construção diferente.
  - Os objetos devem ter pelo menos três propriedades (atributos).
  - Para testar os seus objetos, imprima no Console os atributos dos mesmos.
- Construa um objeto que represente um Contrato de compra e venda. Esse objeto deve conter propriedades de um contrato. O comprador e o vendedor devem ser representados por objetos que contém dados como nome, sobrenome, etc.

Loop "for … in" com propriedades

- Em JavaScript, o laço **"for … in"** percorre as propriedades de um objeto. Sua sintaxe é:

```
for (propriedade in objeto) {
    código a ser executado
}
```

- **Exemplo:**

```
var h1 = {nome:"Wolverine", grupo:"XMen"};
for (x in h1) {
    console.log(x + " - " + h1[x]);
}
```

→ ver **Exemplo-Slide-23.html**

Adicionando novas propriedades

- Podemos **adicionar novas propriedades** a objetos já existentes simplesmente atribuindo novos valores.

- **Exemplo**:

```
jedi.raca = "Zabrak";
```

→ ver **Exemplo-Slide-24.html**

Removendo propriedades

- É possível **remover propriedades** de um objeto:

```
delete jedi.raca;
// ou
// delete jedi["raca"];
```

→ ver **Exemplo-Slide-25.html**

Adicionando métodos

- Podemos definir **métodos** dentro do **construtor**

```
function jedi(n, s, p) {
   this.nome = n;
   this.sobrenome = s;
   this.poder = p;

   this.mudarNome = function (novoNome) {
      this.nome = novoNome;
   }
}

var j2 = new jedi("No name", "Secura", 88);
j2.mudarNome("Aayla");
```

→ ver **Exemplo-Slide-26.html**

Exercícios

- Adicione métodos nos três objetos criados anteriormente. Teste os métodos executando os mesmos.
- Adicione um método no objeto que representa um contato. O método deve imprimir no Console um texto informando todos os dados contidos no contrato, bem como nos objetos comprador e vendedor.

Anexando eventos a formulários

- O principal evento associado a um formulário é o **`submit`**, e o manipulador de eventos é **`onsubmit`**.
- Podemos anexar o manipulador de eventos via código JavaScript (método tradicional):

```
document.getElementById("f1").onsubmit = func1;
```

- Ou podemos fazer inline usando o **`return`**:

```
<form name="f1" onsubmit="return func1();">
```

- Para **cancelar** o envio do formulário, basta com que o retorno de `func1` seja `false`.

→ ver **`Exemplo-Slide-28.html`**