Instituto Federal de Educação Ciência e Tecnologia
Curso Técnico Integrado em Informática
Campus Currais Novos

Desenvolvimento Web

JavaScript – aula IV – *Funções*

Professor: Bruno E. G. Gomes

2013

Introdução

- Na aula de hoje iremos detalhar o conceito de funções em *JavaScript*
- Funções permitem dividir o código *JavaScript* em módulos
  - Dividido em partes menores, reutilizáveis
- É possível definir uma biblioteca de funções relacionadas e reutilizá-la em seus programas

Funções em *JavaScript*

- Uma função define um *bloco* reutilizável de código

- Criar funções permite uma melhor organização do seu código *JavaScript*
  - Código dentro de uma função não interfere em códigos externos
  - É possível criar uma biblioteca de funções relacionadas que você pode reutilizar em seus diversos *scripts*

- Palavra-chave ***function*** define a criação de uma nova função

Funções

- Código da função será executado de duas formas
  - Através de uma chamada à função dentro de <script></script>
  - Por um evento que, quando acionado, ative a execução da função
- Variáveis locais
  - Variáveis declaradas dentro de funções
  - Mantém seu valor apenas durante a execução da função
  - É uma boa prática utilizar somente variáveis locais em funções

Declarando uma nova função

function <*nome*> ([p_1, p_2,..., p_n])
{
    <*código da função*>
    return *[retorno]*
}

<*nome*> - nome da função;

<*código_da_função*> - código a ser executado quando da chamada à função;

[p_1, p_2,..., p_n] – parâmetros da função (opcional). São os valores que a ela recebe como entrada;

[retorno] – valor que a função retorna como saída (opcional). palavra-chave **return**.

Exemplos

```
function ola ()
{
        alert("Hello World!");
}
```

```
function  soma (a, b)
{
        return a + b;
}
```

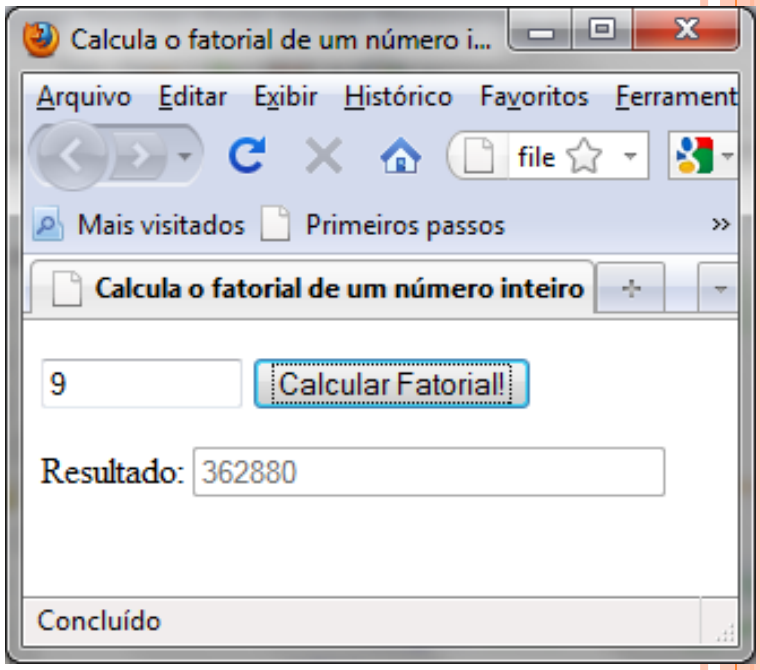
OBS.: No caso da função "soma" acima, se os argumentos recebidos pela função forem Strings, elas serão concatenadas.

Exercício

- Faça uma função par que receba um valor e retorne **verdadeiro** (*true*) se ele for par e **falso** (*false*) se ele for ímpar.

Exercício

- Implemente em *JavaScript* a função fatorial, que retorna o fatorial de um número inteiro. O fatorial é definido por:
  - *fat* (0) = 1
  - *fat* (1) = 1
  - *fat* (n) = *fat* (n-1) * n



- Receba as entradas da função de uma caixa de texto de formulário e exiba os resultados também em um formulário, conforme a figura ao lado:

Funções – o *array* "*arguments*"

- *Array arguments*:
  - Outra forma de acessar os valores dos atributos de uma função

- Mesmo que a função não receba parâmetros na sua declaração, podemos passar valores a ela e chamá-los com *arguments*

- Uso de *arguments* pode servir para definir funções que aceitam qualquer número de argumentos

Exemplo do uso de "arguments"

```
/* Função soma_n
 * Soma um número indefinido de argumentos
 */
function soma_n() {
    var soma = 0;
    var i = 0;

    for (i = 0; i < soma_n.arguments.length; i++) {
        soma += soma_n.arguments[i];
    }

    return soma;
}
```

Exemplo – XHTML Utilizando a função *soma_n*

```
/* Inclui o JavaScript externo (no caso, mat.js) contendo a
   definição da função soma_n.
<head>
...
<script type="text/javascript" src="mat.js">
</script>
/* outra tag script, contendo uma caixa de "alerta" exibindo o
   resultado da soma com vários argumentos passados à
   função.  */
<script type="text/javascript">
      window.alert(soma_n(3, 4, 5, 6));
</script>
...
<head>
```

Exercício

- Faça uma implementação da função *Math.max* (obviamente, sem usar *Math.max*), que retorna o maior número de uma lista de números fornecidos como argumento da função. Chame a sua função de *maximo*.

Exemplos:

maximo (5, 8, 5.6, 9.9) retorna como resultado 9.9

maximo (3, 2, 1) resulta em 3

Passagem de parâmetros

- Parâmetros pode ser passados para a função de duas formas

- Por valor
  - O valor da variável passada à função é copiado no parâmetro
  - Não altera a variável externa
  - Valores de tipos primitivos são passados por valor

- Por referência
  - Um apontador (referência) para o endereço da variável é copiado no parâmetro
  - Qualquer alteração por meio do parâmetro altera a variável externa
  - *Arrays* e Objetos são passados por referência

Exemplo – passagem de parâmetro por referência

Declare a função abaixo em um JavaScript externo.

```
function addDate(dt, d) {
    dt.setDate(dt.getDate() + d);
}
```

Exemplo – Script a ser inserido dentro do *XHTML* para testar *addDate*

```
<script type="text/javascript" src="func-ref.js">
</script>

    <script type="text/javascript">
        var d = new Date(2010, 7, 9);

        document.writeln("<p>" + d + "</p>");
        addDate(d, 2);
        document.writeln("<p>" + d + "</p>");
</script>
```

Exercício

- Faça uma função *multArray* que multiplique todos os números armazenados em um *array* por outro número fornecido. O *array* e o número para multiplicá-lo devem ser passados como parâmetros da função.

  - OBS.: Para passar um *array* para a função é preciso apenas passar o nome do *array* (sem colchetes)