JavaScript: Functions

Introdução

- **Para desenvolver e manter um programa grande**
  - **construí-la a partir de pequenos pedaços simples**
  - **dividir e conquistar**

Observação de Engenharia de Software

Evite reinventar a roda. Use objetos JavaScript, métodos e funções existentes em vez de escrever novos. Isso reduz o tempo de desenvolvimento e ajuda a evitar erros.

Programando Funções em Javascript

- **Você pode definir funções como programador e executar tarefas específicas em muitos pontos de um script**
  - **As demonstrações reais que definem a função são escritas apenas uma vez e estão escondidas de outras funções**
- **Funções são invocadas escrevendo seu nome, seguido por um parêntese esquerdo, seguido de uma lista de zero ou mais argumentos separados por vírgula, seguido por um parêntese direito**
- **Métodos são chamados da mesma maneira que as funções, mas requerem o nome do objeto para o qual o método pertence e um ponto antes do nome do método.**

Funções definidas pelo programador

- **Variáveis declaradas nas definições de funções são variáveis locais**
  - **elas só podem ser acessadas na função em que são definidas**
- **Parâmetros de uma função são considerados variáveis locais**
- **Quando uma função é chamada, os argumentos na chamada são atribuídos aos parâmetros correspondentes na definição da função**

As definições de função

- **declaração `return`**
  - **passa informações de dentro de uma função de volta para o ponto do programa onde foi chamado**
- **A função deve ser chamada explicitamente para o código em seu corpo para executar**
- **O formato da definição da função é**
- **função function-name (lista de parâmetros)**
- **{**
- **declarações**
- **}**

As definições de função

- **Três maneiras de retornar o controle para o ponto em que a função foi invocada**
  - **Atingindo a chave direita de fim de função**
  - **Executando a instrução return;**
  - **Executar o comando "return expressão;" para retornar o valor da expressão para o chamador**
- **Quando uma instrução de retorno é executada, o controle retorna imediatamente para o ponto em que a função foi invocada**

Função definida pelo programador square (Parte 1 de 2).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 9.2: SquareInt.html -->
6  <!-- Programmer-defined function square. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>A Programmer-Defined square Function</title>
10       <script type = "text/javascript">
11          <!--
12          document.writeln( "<h1>Square the numbers from 1 to 10</h1>" );
13
14          // square the numbers from 1 to 10
15          for ( var x = 1; x <= 10; x++ )
16             document.writeln( "The square of " + x + " is " +
17                square( x ) + "<br />" );
18
19          // The following square function definition is executed
20          // only when the function is explicitly called.
21
22          // square function definition
23          function square( y )
24          {
25             return y * y;
26          } // end function square
27          // -->
28       </script>
29    </head><body></body>
30 </html>
```

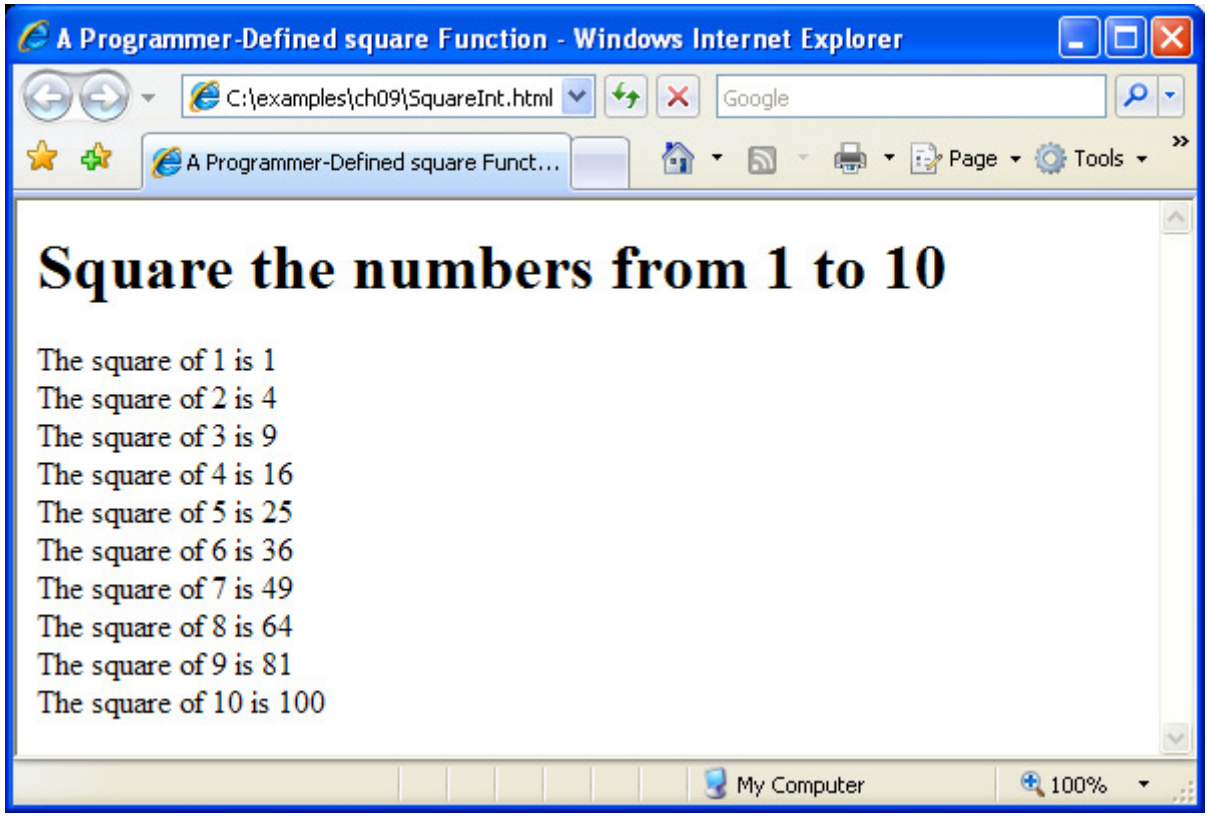
Chama a função square (quadrado) com x como argumento, que irá retornar o valor a ser inserido aqui

Nomeia o parâmetro `y`

Comece a função `square`

Final da função `square`

Retorna o valor de `y * y` (quadrado) para o chamador



Função definida pelo programador **square** (Parte 2 of 2).

Observação de Engenharia de Software

Declarações que são colocadas no corpo da definição da função não são executadas pelo interpretador Java-Script, a menos que a função seja chamada explicitamente.

Erro comum de programação

Esquecendo-se de retornar um valor de uma função que deve retornar um valor é um erro de lógica.

Erro comum de programação

Colocar um ponto e vírgula após o parêntese direito encerrando a lista de parâmetros de uma definição de função resulta em um erro de tempo de execução de JavaScript.

Erro comum de programação

Redefinir um parâmetro de função como uma variável local na função é um erro de lógica.

Erro comum de programação

Passando para uma função um argumento que não é compatível com o tipo esperado pelo parâmetro correspondente é um erro de lógica e pode resultar em um erro de execução de JavaScript.

Boa prática de programação

Embora não seja errado fazê-lo, não use o mesmo nome de um argumento passado para a função e o parâmetro correspondente na definição da função. Usando diferentes nomes evita-se a ambiguidade.

Observação de Engenharia de Software

Para promover a reutilização de software, cada função deve ser limitada a execução de uma tarefa única, bem definida, e o nome da função deve expressar essa tarefa de forma eficaz. Tais funções tornam os programas mais fáceis de escrever, depurar, manter e modificar.

Dica de prevenção de Erro

Uma pequena função que executa uma tarefa é mais fácil de testar e depurar do que uma função maior, que executa muitas tarefas.

Eventos

- **JavaScript pode executar ações em resposta à interação do usuário com um componente GUI XHTML. Isso é conhecido como manipulação de eventos GUI**
- **Atributo onclick de um elemento XHTML indica a ação a ser tomada quando o usuário do documento XHTML clica no elemento**
- **Na programação orientada a eventos**
  - **o usuário interage com um componente GUI**
  - **o roteiro é notificado sobre o evento**
  - **o script processa o evento**
- **A interação do usuário com a interface gráfica "dirige" o programa.**
- **A função que é chamada quando ocorre um evento é conhecida como uma função de tratamento de eventos ou manipulador de eventos.**

Eventos(Cont.)

- **O método getElementById, dado um id como um argumento, encontra o elemento XHTML com um atributo id correspondente e retorna um objeto JavaScript que representa o elemento**
- **O valor da propriedade de um objeto JavaScript que representa um elemento de entrada de texto em XHTML especifica o texto a ser exibido no campo de texto**
- **Um recipiente XHTML (por exemplo, div, button, p) a propriedade InnerHTML do elemento pode ser usado em um script para definir o conteúdo desse elemento**

Observação de Engenharia de Software

As variáveis que são declaradas no interior do corpo de uma função são conhecidas apenas pela função. Se os mesmos nomes de variáveis são usadas em outras partes do programa, elas serão variáveis completamente diferentes na memória.

Regras de escopo

- **Cada identificador de um programa tem um alcance**
- **O âmbito de um identificador para uma variável ou função é a parte do programa, em que o identificador pode ser referenciado**
- **As variáveis globais são acessíveis em qualquer parte de um programa e dizem ter escopo global**
  - **Assim, todas as funções do script podem potencialmente usar essas variáveis**

Regras de escopo (Cont.)

- **Identificadores declarados dentro de uma função tem escopo local e só podem ser utilizados nessa função**
- **Se uma variável local de uma função tem o mesmo nome de uma variável global, a variável global está "escondida" a partir do corpo da função.**

Boa prática de programação

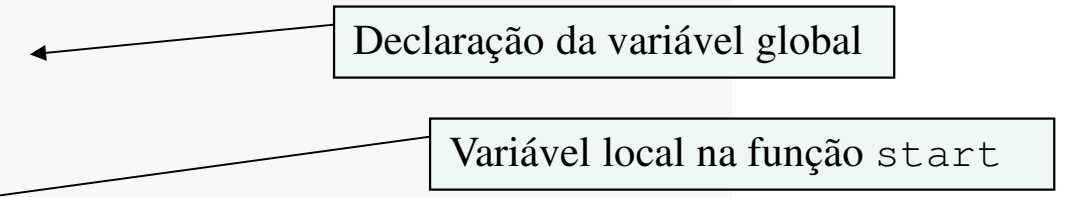
Evite nomes de variável local que escondem nomes de variáveis globais. Isto pode ser conseguido simplesmente por evitar a utilização de identificadores duplicados num script.

Regras de escopo (Cont.)

- A propriedade *onload* do elemento *body* pede um manipulador de eventos quando a tag <body> do documento XHTML é completamente carregada na janela do navegador

exemplo (Parte 1 de 3).

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 9.8: scoping.html -->
6  <!-- Scoping example. -->
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>A Scoping Example</title>
10       <script type = "text/javascript">
11          <!--
12          var x = 1; // global variable
13
14          function start()
15          {
16             var x = 5; // variable local to function start
17
18             document.writeln( "local x in start is " + x );
19
20             functionA(); // functionA has local x
21             functionB(); // functionB uses global variable x
22             functionA(); // functionA reinitializes local x
23             functionB(); // global variable x retains its value
24
25             document.writeln(
26                "<p>local x in start is " + x + "</p>" );
27          } // end function start
28
```

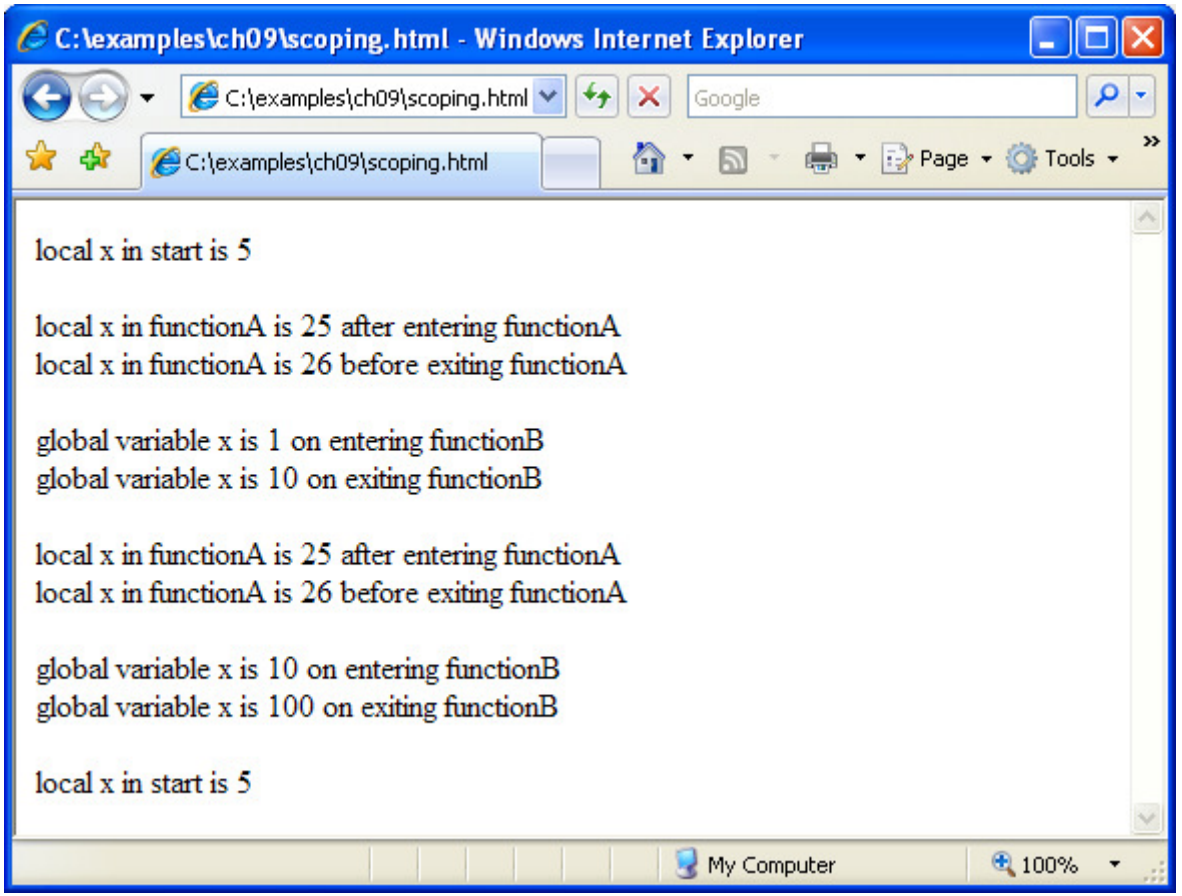


```
29          function functionA()
30          {
31             var x = 25; // initialized each time
32                         // functionA is called
33
34             document.writeln( "<p>local x in functionA is " +
35                               x + " after entering functionA" );
36             ++x;
37             document.writeln( "<br />local x in functionA is " +
38                x + " before exiting functionA" + "</p>" );
39          } // end functionA
40
41          function functionB()
42          {
43             document.writeln( "<p>global variable x is " + x +
44                " on entering functionB" );
45             x *= 10;
46             document.writeln( "<br />global variable x is " +
47                x + " on exiting functionB"  + "</p>" );
48          } // end functionB
49          // -->
50       </script>
51    </head>
52    <body onload = "start()"></body>
53 </html>
```

Varaiável local na função `functionA`, inicializada a cada vez que `functionA` é chamada

Chama a função `start` quando o corpo do document foi carregado na janela do navegador

exemplo(Parte 2 de 3).



Exemplo (Parte 3 de 3).