O que é Javascript?

- Javascript, como o nome já diz, é uma linguagem de script.
  - Entenda linguagem de script com uma linguagem de programação leve e mais simples.
- O principal objetivo do Javascript é adicionar interatividade nas páginas HTML.
- É uma linguagem interpretada.
- Bastante simples e prática, é uma linguagem baseada em objetos e eventos.
- Suportada pela maioria dos navegadores atuais: Firefox, Internet Explorer, Opera, Safari...
- Seu nome real é ECMAScript. Este nome é baseado no orgão que padroniza a linguagem.
- Não confundir com JScript. Este é um padrão da Microsoft e com baixa adoção.

O que Javascript pode fazer?

- Javascript é uma ferramenta de programação para páginas web.
- Javascript pode ler e escrever HTML dentro de uma página.
- Javascript pode ficar aguardando um evento ocorrer para disparar uma ação em resposta. (listener)
- Javascript pode ser usado para validar informações antes que elas sejam enviadas ao servidor.
- Javascript pode trabalhar com cookies (não comestíveis). Cookies permitem armazenar informações do usuário entre páginas ou sessões.
- Javascript possui suporte a expressões regulares.

O que Javascript não pode fazer?

Dentro do nosso escopo de estudo:

- Acessar ou definir as preferências do usuário para impressão, aparência e opções gerais do navegador.
- Executar um aplicativo (programa) no computador do usuário.
- Ler ou escrever em arquivos ou diretórios no computador do usuário.
- Enviar e-mails.

Nosso primeiro exemplo

```
<html>
<head>
  <title>Meu primeiro Javascript</title>
</head>
<body>
  <script type="text/javascript">
    window.alert( 'Olá mundo!' );
  </script>
</body>
</html>
```

Comentários

- Comentário de uma linha:

```
// este é um comentário de linha
```

- Comentário de bloco:

```
/*
   Este é um comentário de bloco.
   Ele pode ter múltiplas linhas.
*/
```

Localização do Javascript

- Dentro da página (incorporado).
- Externo (anexado):

```
<script type="text/javascript" src="arquivo.js"></script>
```

- A extensão padrão de um arquivo Javascript é "js".

Variáveis

- Tipagem dinâmica
- Tipagem fraca
- Exemplos:

```
var nome; // declaração de uma variável
var idade = "vinte e sete"; // declaração e atribuição
nome = "Christiane"; // atribuição de uma variável
idade = 27; // tipagem dinâmica

window.alert( nome + idade ); // tipagem fraca
```

Tipos básicos

- Sete tipos básicos:
  - String
  - Boolean
  - Number
  - Function
  - Object
  - Null
  - Undefined

Operadores aritméticos

- + (adição e concatenação)
- - (subtração)
- * (multiplicação)
- / (divisão)
- % (módulo)
- ++ e -- (incremento e decremento)
  - Podendo ser pré-incremento ou pós-incremento.
    O mesmo vale para o decremento.

O operador +

- Usado também para concatenar strings
- Você pode somar (concatenar) um número com uma string e o resultado será uma string.

Operadores de comparação

- == (igual)
- === (exatamente igual, idêntico)
- != (diferente)
- > (maior que)
- < (menor que)
- >= (maior ou igual que)
- <= (menor ou igual que)

Operadores lógicos

- && (E lógico)
- || (OU lógico)
- ^ (XOR binário)
- ! (negação)

- Operador ternário:
  - (condição ? caso verdadeiro : caso falso)

Estrutura: if ... else if ... else

- Similar ao que temos em Java e C.

```
if( condição )
{
  // instruções
}
else if( condição )
{
  // mais instruções
}
else
{
  // ainda mais instruções
}
```

Estrutura: switch ... case ... case ... default

- Também similar ao que temos em C e Java.

```
switch( variável )
{
  case "homem" :
    mensagem = "Bem-vindo!";
    break;
  case "mulher" :
    mensagem = "Bem-vinda!";
    break;
  default :
    mensagem = "Olá!";
}
```

Caixas de diálogo

- O Javascript possui três tipos de caixas de diálogo que permitem a interação com o usuário.
  - alert(mensagem)
    - Usada para exibir uma informação. Esta caixa de diálogo não retorna informação alguma, apenas exibe a informação.
  - confirm(mensagem)
    - Usada para pedir confirmação do usuário. A confirmação é sempre sim ou não (verdadeiro ou falso).
  - prompt(mensagem[, texto_padrão])
    - Permite a entrada de uma string pelo usuário.

Funções

- Funções em Javascript são compostas, basicamente, por quatro elementos.
  - Nome
  - Parâmetros
  - Instruções
  - Valor de retorno
- Nenhum deles é obrigatório.

```
function nome( parametros )
{
  // uma instrução
  // outra instrução
  return algum_valor;
}
```

- Uma função sem nome é chamada de anônima.

Funções (continuação - exemplo)

```
// definindo uma função de soma
function soma( a, b )
{
  return a + b;
}

// agora vamos usar a função soma
var x = 10;
var y = 5;
z = soma( x, y );

// e vamos usar novamente
w = 5;
w = soma( z, w );
```

Escopo de variáveis

- Uma variável declarada dentro de uma função existe enquanto a função está sendo executada.
- Esta variável, limitada pelo escopo da função, recebe o nome de **variável local**.

Estrutura: for

- Similar ao existente no Java e C.

```
for( instrucao_inicio, condicao_parada, pos_iteracao )
{
  // instruções aqui
  // mais instruções
  // podemos passar para a próxima iteração usando:
  continue;
  // ou podemos forçar a parada da repetição usando:
  break;
}
```

- Cada uma das repetições recebe o nome de iteração.

Estrutura: for (continuação - exemplo)

```
var soma = 0;
for( int i = 0; i < 10; i++ )
{
  soma += i;
}
alert( soma );
```

Estrutura: while

```
while( condição_parada )
{
  // instruções
  // outras instruções
  // podemos passar para a próxima iteração com um:
  continue;
  // ou interromper de vez usando:
  break;
}
```

- Também temos o do ... while disponível.

Estrutura: while (continuação - exemplo)

```
while( 18 > prompt( "Você deve ter mais que 18 anos
  para acessar esta página!", "" ) )
{
  // he he he he
}
```

Vetores e Matrizes

- Vetores e Matrizes são conjuntos de variáveis.
- Podem ter somente uma dimensão (vetor) ou várias dimensões (matriz).
- Exemplos:

```
var cores = new Array();
cores[0] = "blue";
cores[1] = "red";
cores[2] = "green";

var cidades = new Array( "São Paulo", "Rio de Janeiro",
"Porto Alegre" );
```

Vetores e Matrizes (continuação)

```
var cores = new Array();
cores[0] = "blue";
cores[1] = "red";
cores[2] = "green";
```

- O acesso aos itens dos vetores é feito usando seus índices:

```
alert( cores[0] ); // o primeiro elemento das cores
alert( cores[2] ); // o terceiro elemento das cores
```

Vetores e Matrizes (continuação)

- A criação de matrizes é similar. Para criar uma matriz de duas dimensões e tamanho 2x3, por exemplo:

```
var matriz = [];
matriz[0] = [ "um", "dois", "três" ];
matriz[1] = [ "quatro", "cinco", "seis" ];
// matriz agora é
// [ ["um", "dois", "três"], ["quatro", "cinco",
  "seis"] ]
```

Vetores e Matrizes (continuação)

- Os índices de um vetor não são obrigatoriamente números. Podem ser, por exemplo, strings:

```
var notas = new Array();
notas["Carlos"] = 7;
notas["Shirley"] = 2;
notas["Maria"] = 8.5;
```

Estrutura: for ... in

- Esta estrutura permite iterar nos elementos de um vetor ou nas propriedades de um objeto qualquer.
- Pode ser aplicado em um vetor que possui strings como índices, para descobrir quais são os índices usados.

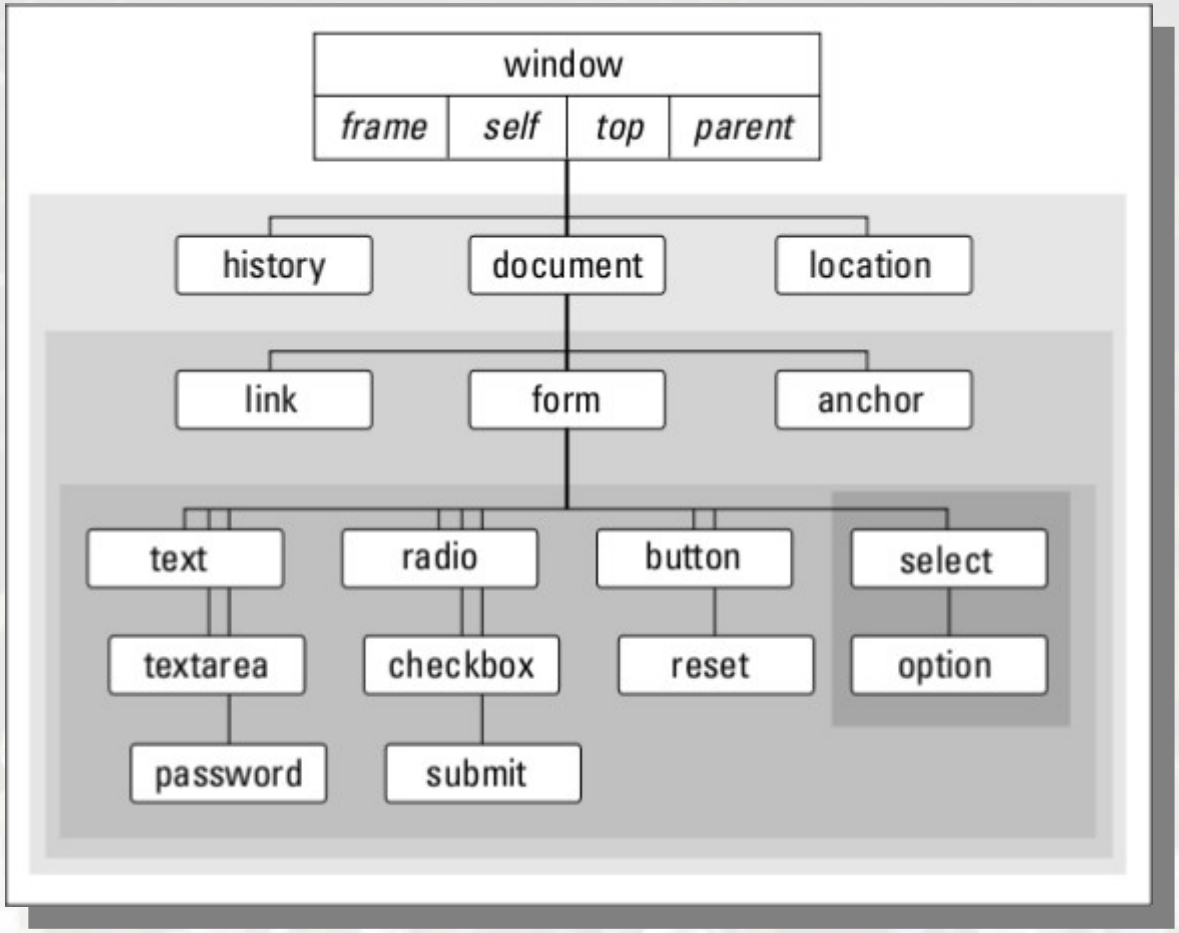
```
for( indice in vetor_ou_objeto )
{
  // instruções aqui
  alert( "índice: " + indice );
  alert( "valor: " + vetor_ou_objeto[indice] );
}
```

Estrutura: for ... in (continuação)

- Exemplo:

```
var versao_atual = [];
versao_atual["IE"] = 7;
versao_atual["Firefox"] = 3;
versao_atual["Safari"] = 3.1;
versao_atual["Opera"] = 9.5;
for( browser in versao_atual )
{
  alert( browser + ": " + versao_atual[browser]
}
```

DOM (Document Object Model)



(Imagem do livro Javascript Bible, Gold Edition - página 30)

Eventos

- Eventos são ações geradas pelo navegador ou pelo usuário.
- Você pode especificar funções para responder aos eventos disparados. (listeners)
- Exemplos de eventos:
  - blur
  - focus
  - change
  - click
  - mouseover
  - mouseout
  - load
  - unload