

1 . JavaScript

1. JavaScript (http://www.w3schools.com/js/default.asp).....	02
1.1 JavaScript Como Fazer.....	03
1.2 Onde colocar seu JavaScript.....	04
1.3 Variáveis JavaScript.....	05
1.4 Operadores JavaScript.....	06
1.5 Funções JavaScript.....	08
1.6 Instrução Condicional em JavaScript.....	10
1.7 Repetição JavaScript.....	12

1. JavaScript

JavaScript é usado por milhões de páginas web implementando design, validação de formulários e muito mais. JavaScript foi criado pela Netscape e é a linguagem script mais popular da Internet.

- O que é JavaScript?

- * JavaScript é uma linguagem script.
- * Um JavaScript é geralmente introduzido em páginas HTML.
- * JavaScript é uma linguagem interpretada, não precisando de compilação pré-execução.
- * Não é preciso licença para usar JavaScript.
- * A maioria dos navegadores web tem suporte nativo ao JavaScript.

- Java e JavaScript são iguais?

Não!
Java e JavaScript são duas linguagens completamente diferentes!
Java (desenvolvida pela Sun Microsystems) é uma poderosa e complexa linguagem - na mesma categoria de C e C++.

- O que pode ser feito com JavaScript?

- * JavaScript fornece a designers HTML uma ferramenta de programação.
- * JavaScript pode colocar textos dinâmicos em sua página HTML como isto:
`document.write("<h1>" + name + "</h1>")`.
- * JavaScript pode reagir a eventos - Um JavaScript pode ser setado a executar quando algo acontecer, como um click de botão.
- * JavaScript pode ler e escrever elementos HTML - Um JavaScript pode ler e mudar características de um elemento HTML.
- * JavaScript pode ser usado para validação de dados - Um JavaScript pode ser usado para validar um formulário antes de ser submetido ao servidor, isto livra o servidor de um processamento extra.

1.1 JavaScript

A tag HTML <script> é usada para inserir um JavaScript em uma página HTML.

- Como inserir um JavaScript em uma página HTML

```
<html>
  <body>
    <script type="text/javascript">
      document.write("Olá Mundo!")
    </script>
  </body>
</html>
```

O código acima produzirá esta saída na página HTML:

```
Olá Mundo!
```

- Explicação do Exemplo

Para inserir um script em uma página HTML, nós usamos a tag <script>. Usamos o atributo type para definir a linguagem script a ser utilizada, caso não seja utilizado o navegador assume o padrão.

```
<script type="text/javascript">
```

Então o JavaScript deve ser inicializado: O comando JavaScript para escrever algo em uma página é document.write

```
document.write("Hello World!")
```

Então a tag <script> deve ser fechada

```
</script>
```

1.2 Onde colocar seu JavaScript

Scripts na seção de corpo (body) serão executados ENQUANTO a página é carregada

Scripts na seção de cabeçalho (head) serão executados quando ocorrer uma CHAMADA.

- Onde colocar o JavaScript

Executar Scripts quando a página for carregada! Geralmente isto não é o que queremos. Algumas vezes nós queremos executar um script quando a página carrega, outras vezes quando ocorre um evento.

Scripts na seção de cabeçalho: Script a ser executado quando ocorrer uma chamada, ou quando ocorrer um evento, vá para a seção de cabeçalho. Quando você coloca o seu script na seção de cabeçalho, você vai verificar que o script será carregado antes que qualquer um o use.

```
<html>
  <head>
    <script type="text/javascript">
      ...
    </script>
  </head>
```

Scripts na seção de corpo: Script a ser executado quando a página é carregada, vá a seção de corpo. Quando você coloca o seu script na seção de corpo ele é gerado junto com a execução da página.

```
<html>
  <head>
  </head>
  <body>
    <script type="text/javascript">
      ...
    </script>
  </body>
```

Scripts em ambas seções podem ser utilizados: Você pode colocar um ilimitado número de scripts em seu documento, variando entre scripts na seção de corpo e na seção de cabeçalho.

1.3 Variáveis JavaScript

Uma variável é um "container" para informações que você quer guardar. O valor das variáveis pode ser mudado durante o script. Você pode referenciar uma variável pelo nome para ver o seu valor ou troca-lo.

- Regras para nomes de variáveis

- * O nome de variáveis são caso sensitivo
- * Precisam ser inicializadas com letras ou underscore

- Declarando uma variável

Você pode criar uma variável com a instrução var:

```
var minha_variavel = algum valor
```

Você também pode criar variáveis se a instrução var:

```
minha_variavel = algum valor
```

- Determinar um valor para uma variável

Você determina um valor para uma variável assim:

```
var minha_variavel = "algum valor"
```

Ou assim:

```
minha_variavel = "algum valor"
```

O nome da variável é o do lado esquerdo da expressão e o valor que você quer determinar é o do lado direito. Agora a variável "minha_variavel" tem o valor "algum valor".

- Vida de uma variável

Quando você declara uma variável dentro de uma função, a variável só pode ser acessada dentro daquela função. Quando você sai da função a variável é destruída. Estas variáveis são chamadas de variáveis locais. Você pode ter variáveis locais com o mesmo nome e diferentes funcionalidades, pois cada uma é reconhecida pela função que a declarou.

Se você declara uma variável fora de uma função, todas as funções de sua página podem acessa-la. A vida desta variável começa quando ela é declarada, e termina quando a página é fechada.

1.4 Operadores JavaScript

- Operadores Aritméticos

Operador	Descrição	Exemplo	Resultado
+	Adição	x=2 x+2	4
-	Subtração	x=2 5-x	3
*	Multiplicação	x=4 x*5	20
/	Divisão	15/5 5/2	3 2.5
%	Módulo (resto da divisão)	5%2 10%8 10%2	1 2 0
++	Incremento	x=5 x++	x=6
--	Decremento	x=5 x--	x=4

- Operadores de Tarefa

Operador	Exemplo	É igual a
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
=	x=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

- Operadores de Comparação

Operador	Descrição	Exemplo
==	é igual a	5==8 retorna false
!=	não é igual a	5!=8 retorna true
>	é maior que	5>8 retorna false
<	é menor que	5<8 retorna true
>=	é maior que e igual a	5>=8 retorna false
<=	é menor que e igual a	5<=8 retorna true

- Operadores Lógicos

Operador	Descrição	Exemplo
&&	and	x=6 y=3 (x < 10 && y > 1) retorna true
	or	x=6 y=3 (x==5 y==5) retorna false
!	not	x=6 y=3 !(x==y) retorna true

- String

Uma string é mais do que um texto, por exemplo, "Olá Mundo!". Para concatenar mais de duas strings, use o operador +.

```
txt1 = "Hoje está "
txt2 = "um belo dia!"
txt3 = txt1 + txt2
```

A variável txt3 agora contém "Hoje está um belo dia!".

- Propriedades do objeto String

Propriedade	Descrição
length	Contém o número de caracteres que compõem a string

- Métodos do objeto String

Método	Descrição
charAt(índice)	Devolve o caractere que ocupa a posição índice na string
charCodeAt(índice)	Devolve o código (conjunto Unicode) do carácter que ocupa a posição índice na string
indexOf(string_busca, índice_opcional)	Devolve a posição em que se encontra a primeira ocorrência de string_busca ou -1 se essa ocorrência não existir. Se não fornecermos um índice_opcional a busca inicia-se na posição zero, mas se o fornecermos é nesse ponto que se inicia a busca.
lastIndexOf(string_busca, índice_opcional)	Devolve a posição em que se encontra a última ocorrência de string_busca ou -1 se essa ocorrência não existir. A busca faz-se a partir do fim e caminha progressivamente para o início. Se não fornecermos um índice_opcional a busca inicia-se no fim, mas se o fornecermos é nesse ponto que se inicia a busca.
split(string_separador, limite_opcional)	Divide uma string em partes usando as ocorrências de string_separador como pontos de divisão. Devolve um array com todas as divisões (substrings) encontradas. Cada elemento do array é uma substring da string original. O limite_opcional indica o número máximo de partes a incluir no array que é devolvido. A string_separador é excluída das divisões e o objecto String sobre o qual foi invocado este método não sofre alterações.
substring(início, fim)	Devolve uma secção da string composta pelos caracteres que ocupam as posições com índices entre início (incluída) e fim (excluída).
toLowerCase()	Devolve uma versão da string com todos os caracteres convertidos para letra pequena
toUpperCase()	Devolve uma versão da string com todos os caracteres convertidos para letra grande

- Métodos estáticos do objeto String

Método	Descrição
String.fromCharCode()	Devolve o caractere correspondente ao código Unicode fornecido. Este objecto é muito importante em diversas tarefas que podemos realizar em JavaScript: validar formulários, trabalhar com cookies, etc.

1.5 Funções JavaScript

Uma função é um bloco reutilizável que será executado a partir de um evento, ou quando ocorrer uma chamada.

Uma função é setada por uma instrução. Você pode reutilizar funções dentro do mesmo script, ou em outros documentos. Você define funções no início do arquivo (na seção de cabeçalho), e chama ela mais tarde no documento.

- É hora de aprendermos algo sobre "caixas de alerta"

Este é um método Java Script para alertar um usuário.

```
alert("This is a message")
```

Este é um método Java Script para pedir confirmação a um usuário. Retorna verdadeiro ou falso.

```
confirm("Deseja fazer alterações")
```

- Como definir uma Função

Para criar uma função você define seu nome, seu parâmetros, e alguma instrução:

```
function minha_funcao(parametro_1, parametro_2, etc)
{
    alguma instrução
}
```

Uma função sem parâmetros necessita obrigatoriamente incluir os parênteses:

```
function minha_funcao()
{
    alguma instrução
}
```

Parâmetros são variáveis usadas na função. Os valores dos parâmetros são passadas na chamada da função.

Para colocar uma função na seção de cabeçalho do documento, você tem que ter certeza que todo o código a que a função se refere foi carregado antes que a função seja chamada.

Algumas funções podem retornar valores para a expressão que a chamou.

```
function resultado(a, b)
{
    c = a + b
    return c
}
```

- Como chamar uma função

Uma função não é executada antes de ser chamada.
Você pode chamar uma função passando parâmetros:

```
minha_funcao(parametro_1, parametro_2, etc)
```

ou sem parâmetros:

```
minha_funcao()
```

- Instrução de retorno

Funções que retornam um resultado necessitam da instrução "return". Esta instrução especifica que cada valor será retornado para onde a função foi chamada. Digamos que temos uma função que retorna a soma de dois números:

```
function total(a, b)
{
    resultado = a + b
    return resultado
}
```

Quando você chama esta função você necessita mandar dois parâmetros para ela:

```
soma = total(2, 3)
```

O valor retornado da função (5) será guardado em uma variável chamada soma.

1.6 Instrução Condicional em JavaScript

Instruções condicional em JavaScript são usadas para executar ações diferentes baseadas em diferentes condições.

- Em JavaScript nós temos três instruções condicionais

* if - use esta instrução se você quer executar um bloco de código quando uma condição for verdadeira

* if ... else - use esta instrução se você quer selecionar um ou dois blocos de código para executar

* switch - use esta instrução se você quer selecionar um ou muitos blocos de código para executar

- Instrução - If e If ... else

Você pode usar a instrução if se você quer executar um bloco de código quando uma condição for verdadeira.

Sintaxe:

```
if (condição)
{
bloco a de código a ser executado se a condição for verdadeira
}
```

Exemplo:

```
<script type="text/javascript">
  var texto = 'meu texto'

  if (texto != '')
  {
      document.write("Seu texto é <b>" + texto + "</b>")
  }
</script>
```

Repare que não existe ..else.. nesta sintaxe. Você só descobrirá o código que deverá executar se a condição for verdadeira.

Se você quiser executar algum código que a condição é verdadeira e outro código que a condição é falsa, use a instrução if...else.

Sintaxe:

```
if (condição)
{
    código a executar se a condição for verdadeira
} else {
    código a executar se a condição for falsa
}
```

Exemplo:

```
<script type="text/javascript">
  var texto = 'meu texto'

  if (texto != '')
  {
    document.write("Seu texto é <b>" + texto + "</b>!")
  } else {
    document.write("Seu texto é vazio!")
  }
</script>
```

- Instrução Switch

Você pode usar a instrução Switch se você quer selecionar um ou muitos blocos de código para executar

Sintaxe:

```
switch (expressão)
{
  case condição_1:
    código a ser executado se a expressão == condição_1
    break
  case condição_2:
    código a ser executado se a expressão == condição_2
    break
  default:
    código a ser executado se a expressão != das outras
}
```

Funciona assim: Primeiro nos temos uma expressão simples (maiorias das variáveis), que é avaliada uma vez. O valor da expressão é então comparado com os valores de cada caso no na estrutura. Se existir, o bloco de código associado é executado. Use o break para previne que o código dos próximos casos sejam executados.

Exemplo:

```
<script type="text/javascript">
  var numero=2

  switch (numero)
  {
    case 1:
      document.write("Numero UM")
      break
    case 2:
      document.write("Numero DOIS")
      break
    default:
      document.write("Não é UM nem DOIS")
  }
</script>
```

1.7 Repetição JavaScript

Instruções de repetição em JavaScript são usadas para executar o mesmo bloco de código um número específico de vezes.

- Em JavaScript nós temos as seguintes instruções de repetição

- * while - repete um bloco de código enquanto a condição for verdadeira
- * do...while - repete um bloco de código uma vez, e repete enquanto a condição for verdadeira
- * for - roda a instrução um determinado número de vezes

- while

O while repete um bloco de código enquanto a condição for verdadeira

```
while (condição)
{
    código a ser executado
}
```

- do...while

O do...while repete um bloco de código uma vez, e repete enquanto a condição for verdadeira

```
do
{
    código a ser executado
}
while (condição)
```

- for

O for roda a instrução um determinado número de vezes

```
for (inicialização; condição; incremento)
{
    código a ser executado
}
```