

TECNOLOGIA EM SISTEMAS PARA INTERNET
UNIVALI / CTTMAR

PROGRAMAÇÃO WEB

JavaScript

Professor: Carlos Henrique Bughi

Introdução

- Conteúdo estático (HTML)
- Programação Script lado Servidor (CGI)
- Plugins (ActiveX, Flash, Applets Java)
- JavaScript

Histórico

- Criada para permitir conteúdo dinâmico no lado cliente;
- Derivada da linguagem C e C++;
- Começou a ser desenvolvida em 1995 pela Netscape com o nome de LiveScript;
- Inicialmente deveria “rodar” no lado cliente e no lado servidor;

Histórico

- Lado servidor:
 - Conexão com outros serviços como banco de dados, motores de pesquisa, entre outros;
- Lado cliente:
 - Validação de formulários;

Histórico

- LiveScript vira JavaScript
 - Avanço da linguagem Java
 - Comunicação entre o HTML e o Applet Java;
 - Apesar do nome, o JavaScript tem mais **diferenças** do que **semelhança** com a linguagem Java;

Histórico

- Microsoft X Resto do Mundo
 - Concorria com a linguagem VBScript;
 - Passou a suportar JavaScript, porém com o nome JScript e com baixa compatibilidade;
 - Atualmente, ambos estão convergindo para a utilização de padrões; (Graças a Deus) ;-)

O que pode e o que não pode

- Em geral, use o JavaScript para os seguintes tipos de soluções:
 - Interação do usuário com elementos do formulário;
 - Controlar a navegação por vários frames, plugins ou Applets java a partir de escolhas do usuário;
 - Pré-processar dados no cliente antes do envio a um servidor;
 - Alterar o conteúdo e os estilos dinamicamente e instantaneamente em resposta a interação com o usuário;

O que pode e o que não pode

- O JavaScript não poderá executar:
 - Configurar ou obter configurações de preferência do navegador;
 - Iniciar uma aplicação no computador cliente;
 - Ler ou gravar arquivos ou diretórios no computador cliente ou servidor;
 - Enviar e-mails secretos dos visitantes de site da Web.

O que é necessário para programar?

- Editor de textos;
- Navegador com suporte ao JavaScript.

Maneiras de executar um script

- Execução direta:
 - Instruções JS dentro da tag `<SCRIPT>` no documento HTML;
- Resposta ao evento:
 - As instruções JS são executadas em resposta as ações que o usuário realiza;

Mais sobre a tag <SCRIPT>

- Definindo a linguagem:
 - <SCRIPT type="text/javascript">
 - //codigo fonte aqui
 - </SCRIPT>
- Utilizando arquivos externos:
 - <SCRIPT type="text/javascript"
src="arquivo_externo.js"></SCRIPT>

Valores possíveis:

- text/javascript
- text/jscript
- text/livescript

Dicas para começar....

- Abrir uma janela:

```
<script>  
window.open("http://www.seuendereco.com.br", "",  
"width=400, height=400")  
</script>
```

- Enviar mensagem:

```
<script>  
window.alert("sua mensagem")  
</script>
```

Dicas de começar....

- Escrevendo no documento:

```
<script>  
document.write("olá mundo");  
</script>
```

- Botão chamando uma função JavaScript:

```
<script>  
function ola(){  
    window.alert("sua mensagem");  
}  
</script>  
  
<input type="button" onclick="ola()">
```

Fazendo comentários

- Comentário de linha

// este é um comentário de linha

- Comentário de bloco

/* comentando um bloco
de código

*/

Separação de instruções

- O JS tem duas maneiras de separar instruções:
 - Através do caractere ponto e vírgula (;)
 - Através de uma quebra de linha
- Duas instruções na mesma linha devem ser separadas por ponto e vírgula;

```
<script>  
x = 5  
y = x*2  
document.write("resultado: "+y)  
</script>
```

```
<script>  
x = 5; y = x*2  
document.write("resultado: "+y)  
</script>
```

Declarações de variáveis

- Em JS não é necessário declarar variáveis, mas isso é possível:

```
var operando1  
var operando2
```

- Também pode-se atribuir um valor à variável quando se está declarando

```
var operando1 = 23  
var operando2 = 33
```

- Também é possível declarar várias variáveis na mesma linha, separadas por vírgulas.

```
var operando1,operando2
```


Escopo das variáveis

- Globais

```
<SCRIPT>
var variávelGlobal
</SCRIPT>
```

- Locais

```
<SCRIPT>
function minhaFuncao() {
    var variavelLocal
}
</SCRIPT>
```

```
<SCRIPT>
var numero = 2
function minhaFuncao () {
    numero = 19
    document.write(numero)
}
document.write(numero) //imprime 2
minhaFuncao()
document.write(numero) //imprime 19
</SCRIPT>
```

Tipos de dados

- Números
 - Base 10, qualquer número, por padrão, se entende que está escrito em base 10.
 - Base 8, para escrever um número em octal basta escrever o número precedido de um 0, ex: 045.
 - Base 16, para escrever um número em hexadecimal devemos escrevê-lo precedido de um zero e um xis, por exemplo, 0x3EF.
- String
 - Tudo o que se coloca entre aspas, independente do seu conteúdo, é considerado texto. Ex: meuTexto = “Ola mundo”
- Booleanos
 - Utilizado para realizar operações lógicas, armazena true ou false. Ex: passouDeAno = true

Operadores aritméticos

+	Soma de dois valores
-	Diferença de dois valores, também se pode utilizar para mudar o sinal de um número se o utilizamos com um só operando. Ex: -23
*	Multiplicação de dois valores
/	Divisão de dois valores
%	O resto da divisão de dois números
++	Incremento em uma unidade
--	Decremento em uma unidade

Operadores de atribuição

=	Atribuição. Atribui a parte da direita do igual à parte da esquerda. À direita se colocam os valores finais e à esquerda geralmente se coloca uma variável onde queremos salvar o dado.
+=	Atribuição com soma. Realiza a soma da parte da direita com a da esquerda e salva o resultado na parte da esquerda.
-=	Atribuição com diferença
*=	Atribuição da multiplicação
/=	Atribuição da divisão
%=	Obtém o resto e atribui

Exemplos

poupança = 7000 //atribui um 7000 à variável poupança

poupança += 3500 //incrementa em 3500 a variável poupança, agora vale 10500

poupança /= 2 //divide entre 2 minha poupança, agora ficam 5250

Operadores lógicos

!	Negação, se é true passa para false e vice-versa.
&&	AND (E), se dois são verdadeiros, vale o verdadeiro.
	OR (OU), vale verdadeiro se pelo menos um for verdadeiro.

Operadores condicionais

==	Comprova se dois valores são iguais
!=	Comprova se dois valores são distintos
>	Maior que, devolve true se o primeiro operador for maior que o segundo
<	Menor que, é true quando o elemento da esquerda for menor que o da direita
>=	Maior igual
<=	Menor igual
===	Comprova se dois valores são iguais e se suas variáveis são do mesmo tipo.

Controle de tipos

- Algumas vezes é importante determinar o tipo de dado de uma determinada variável.
- A partir da versão 1.1 do JavaScript, é possível determinar o tipo de dados através do operador `typeof`

```
<script>
var boleano = true
var numerico = 22
var numerico_flutuante = 13.56
var texto = "meu texto"
var data = new Date()
document.write("<br>O tipo de boleano é: " + typeof boleano)
document.write("<br>O tipo de numerico é: " + typeof numerico)
document.write("<br>O tipo de numerico_flutuante é: " + typeof numerico_flutuante)
document.write("<br>O tipo de texto é: " + typeof texto)
document.write("<br>O tipo de data é: " + typeof data)
</script>
```

Estruturas de controle

- Condicional



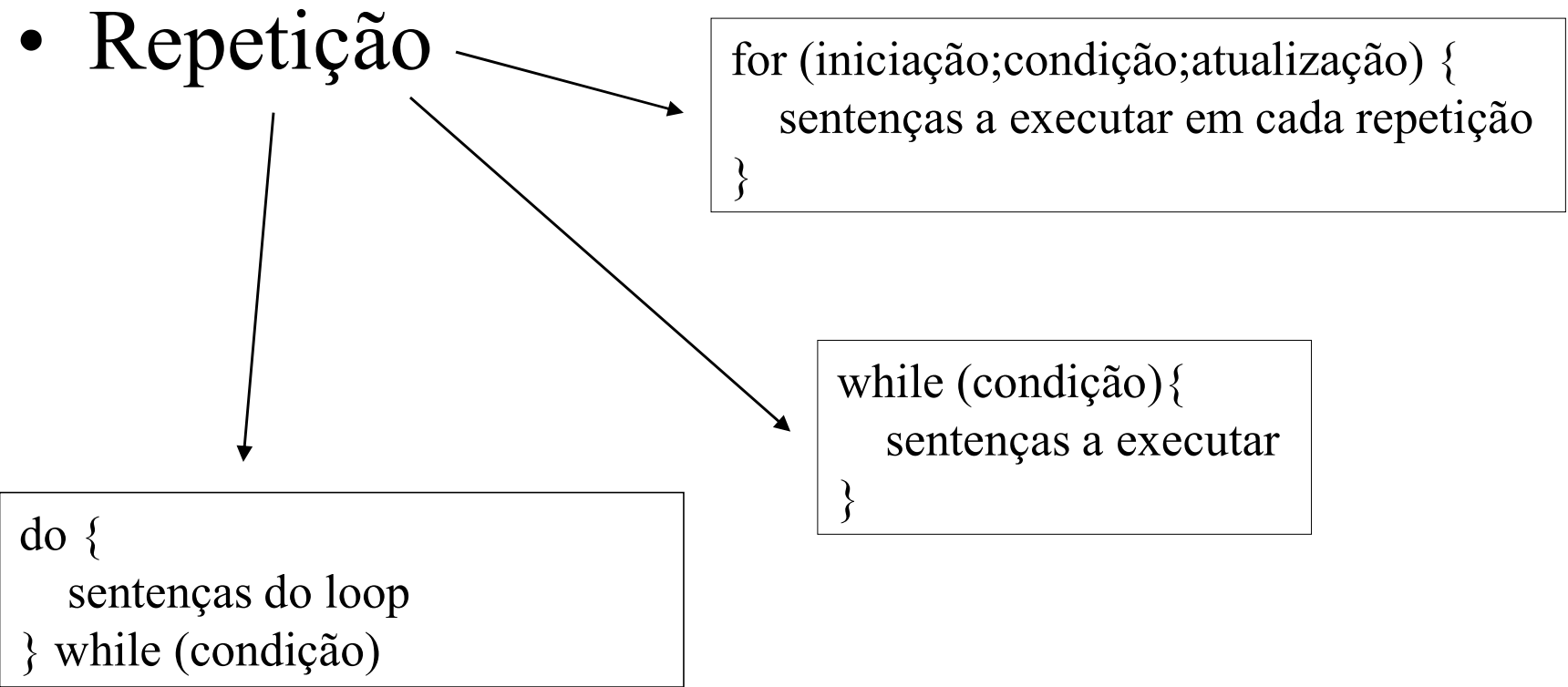
```
Variável = (condição) ? valor1 : valor2
```

```
if (expressão) {  
    ações a realizar em caso positivo  
} else {  
    ações a realizar em caso negativo  
}
```

```
switch (expressão) {  
    case valor1:  
        Sentenças a executar para valor1  
        break  
    case valor2:  
        Sentenças a executar para valor2  
        break  
    default:  
        Sentenças a executar se o valor não é  
        nenhum dos anteriores  
}
```


Estruturas de controle

- Repetição



```
graph TD; Repetição --> For["for (iniciação;condição;atualização) {  
    sentenças a executar em cada repetição  
}"]; Repetição --> While["while (condição){  
    sentenças a executar  
}"]; Repetição --> DoWhile["do {  
    sentenças do loop  
} while (condição)"]
```

for (iniciação;condição;atualização) {
 sentenças a executar em cada repetição
}

while (condição){
 sentenças a executar
}

do {
 sentenças do loop
} while (condição)

Estruturas de controle

- De maneira adicional ao uso das distintas estruturas de loop pode-se utilizar duas instruções para:
 - Deter a execução de um loop e sair dele (**Break**)
 - Deter a repetição atual e voltar ao princípio do loop (**Continue**).

```
for (x = 0; x < 10; x++){  
    if (x == 5){  
        break;  
    }  
}
```

```
var i=0  
while (i<7){  
    incrementar = confirm("incrementar i, atualmente em "+i+"?")  
    if (!incrementar)  
        continue  
    i++  
}
```