Programação Web

Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

Material Teórico

JavaScript

Responsável pelo Conteúdo:
Prof. Esp. Alexander Albuquerque Gobbato

Revisão Textual:
Profa. Ms. Luciene Oliveira da Costa Santos

UNIDADE JavaScript

- JavaScript
- Evento
- Arquivo JavaScript
- Comandos e Funções
- Operadores Lógicos
- Vetores

Objetivo de APRENDIZADO

Na unidade anterior, vimos como o css pode nos auxiliar a criar um visual mais profissional, vimos que o css é utilizado para a apresentação do HTML. Mas como faremos para validar as informações que estão sendo inseridas no formulário?

Para realizar esse tratamento de dados do lado cliente, existe a linguagem de script chamada JAVASCRIPT. Essa linguagem poderá auxiliar o usuário informando o que está sendo feito de errado ou simplesmente informar quais dados estão sendo utilizados.

Atenção

Para um bom aproveitamento do curso, leia o material teórico atentamente antes de realizar as atividades. É importante também respeitar os prazos estabelecidos no cronograma.

Contextualização

Por que é importante aprender javascript?

Com javascript podemos ter acesso a todos os elementos e funcionalidades do HTML. Por meio do javascript podemos auxiliar os usuários informando quais tipos de acesso eles podem ter, quais verificações devem ser feitas do lado cliente, ou seja, antes de enviarmos as informações ao servidor.

Por isso e por mais coisas que veremos adiante é extremamente importante aprendermos e entendermos o funcionamento do script do lado cliente.

JavaScript

- Linguagem integrada ao navegador.
- Usa o chamado "modelo de execução controlado por eventos", ou seja, o código JavaScript só é executado quando o evento ao qual está associado é acionado (em alguns casos, podemos inserir o código sem a associação a eventos e o mesmo é executado conforme o navegador interpreta a página, mas isso é menos usual).
- Processamento do lado cliente, no navegador.

É uma linguagem interpretada, diferente de outras linguagens que fazem o processo de compilação antes de executar o programa como, por exemplo, a linguagem Java.

JavaScript – para que serve?

Javascript é uma linguagem desenvolvida para a internet, com ela é possível realizarmos validações do lado cliente, interações entre páginas.

Com javascript é possível alterar dinamicamente estilos e elementos da página HTML, quando juntamos javascript com css temos então o DHTML, ou criação de HTML dinamicamente.

O que podemos fazer com JavaScript?

Usos mais comuns de JavaScript:

- Fazer algumas animações (como menus e submenus)
- Validar formulários antes do envio
- "Conversar" com o servidor usando XML
- Identificar o contexto (navegador, tamanho de tela etc.) e definir os comportamentos em função dele.

Quando é executado um programa em Javascript?

Peguemos o site do Google como exemplo:

- Quando muda a cor de contorno do campo?
- Quando se abre o autocomplete da procura?
- Quando se abrem os detalhes de um link? (aquilo que aparece do lado direito do link).

"Um programa é uma lista de comandos (ou ações) que devem ser executados em ordem".

Evento

Todo evento acontece em um elemento da página.

E as reações aos eventos (programadas em JavaScript) são sempre registradas como propriedades dos elementos

Elementos	Evento	Propriedade que registra reação
Todos elementos que aparecem na página	Clique no elemento	onclick
	Mouse passando sobre	onmouseover
input, select, textarea	onclick	onfocus
	onmouseover	onblur
Form	Formulário sendo enviado	onsubmit
Body	Página terminou de ser montada	onload

Como inserir um código JavaScript na página?

Podemos inserir nossos códigos JavaScript das seguintes formas:

- Dentro de um marcador html (como fizémos no exercício)
- Dentro do corpo da página <body>....</body>
- Dentro do cabeçalho <head>....</head>

Em um arquivo separado do html, este arquivo deve ter a extensão .js.

JavaScript em uma tag

Quando inserimos um código JavaScript em uma tag, este é sempre associado a um evento.

Note que, se o JavaScript estiver envolvido por aspas, só é possível usar apóstrofes no código (que, no JavaScript, tem a mesma função).

Ex:

```
<html>
<head>
<title>Exemplo JavaScript</title>
</head>
<body>
<input type=”button” value=”Clique aqui” onclick=”window.status=’Teste de javascript...’”>
</body>
</html>
```

JavaScript dentro do body e head

Se fizermos uma comparação com CSS, esse modo de inserir JavaScript em uma página, seria o modo incorporado.

Dentro do body ou do head, utilizamos a tag <script>. Note que os comandos podem ser executados sem a necessidade de eventos. Isso não é o mais comum, mas é um recurso que usaremos em nossos estudos.

Ex:

```
<html>
<head>
<title>Exemplo JavaScript</title>
<script language=”javascript”>
alert(“Eu estou no cabeçalho.”);
</script>
</head>
<body>
<script language=”javascript”>
document.write(“Eu estou no corpo do documento.”);
</script>
</body>
</html>
```

Arquivo JavaScript

Assim como em CSS, também podemos criar um arquivo separado do html com nossos códigos em JavaScript, esse arquivo deve ser salvo com a extensão .js e é chamado no cabeçalho da página com a tag <script>

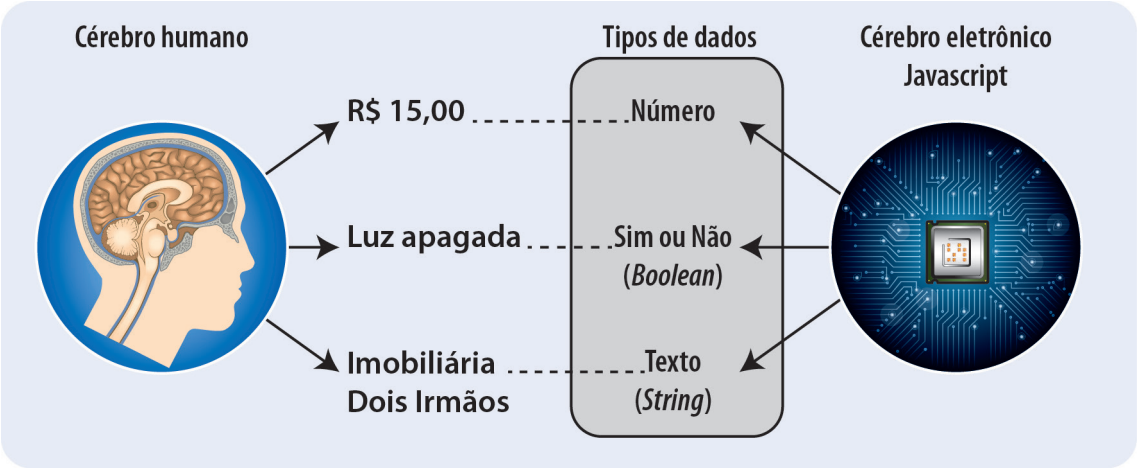
Ex:

```
<html>
<head>
<title>Exemplo JavaScript</title>
<script language=”javascript” src=”ex1.js”></script>
</head>
<body>
```

Conteúdo da página. Também podemos fazer aqui chamadas para blocos de códigos do arquivo .js.

```
</body>
</html>
```

Informação: dados e mais dados



Onde se armazenam os dados?

Sempre em um lugar na memória do computador, que pode ser do tipo:

Constante ou Variável.

- **Constante** – seu conteúdo não muda (ou seja, é constante)
- **Variável** – seu conteúdo pode ser trocado (ou seja, é variável)

Em JavaScript apenas estudaremos VARIÁVEIS, pois nem todos os navegadores reconhecem constantes.

Tipos de variáveis

JavaScript é uma linguagem de tipos de variáveis flexíveis, ao contrário de outras linguagens que exigem a declaração de uma variável com um tipo de dado definido.

Logo, não precisamos definir o tipo de uma variável. Com isso podemos mudar seu valor de texto para número, por exemplo, durante a execução do script (prática que deve ser evitada!).

Ex:

```
var nome="Fulano";
ou
nome="Fulano";
ou
var nome;
```

Nomes de variáveis

- **Variáveis**
  - Todo nome de variável ou constante deve começar com uma letra ou um underscore ("_");
  - Caracteres subsequentes devem ser letras ou números;
  - Não deve conter espaço em branco ou caracteres especiais;
  - Não deve ser uma palavra reservada.

- **JavaScript é Case-Sensitive**, logo as variáveis abaixo são todas diferentes:
  - quantidade
  - QUANTIDADE
  - Quantidade
  - QUantidade

Como trabalhar com dados

Por exemplo, e se eu quiser transferir um dado de um lugar para outro?

Operação de atribuição

```
x = y;
a = b + c
```

A variável x recebe o conteúdo da variável y

A variável a recebe o **RESULTADO** da operação b + c

Operadores Aritméticos para dados do tipo Número

- Para você verificar a tabela abaixo, declaramos uma variável x igual a 10
- var x=10;

Operador		Exemplo	Resultado
+	Adição	z = x + 10;	z = 20
-	Subtração	z = x - 10;	z = 0
*	Multiplicação	z = x * 10;	z = 100
/	Divisão	z = x / 10;	z = 1
%	Módulo (resto de uma operação de divisão)	z = x % 3;	z = 1
++	Incremento antes	z = ++x;	z = 11
	Incremento depois	z = x++;	z = 10
--	Decremento antes	z = --x;	z = 9
	Decremento depois	z = x--;	z = 10

Juntando Textos - Concatenação de Strings

Também usamos o operador "+" para concatenar Strings (textos) em JavaScript

Ex:

```
var x="Bom";
var y="dia";
var z = x + y;
// após a execução o resultado será: z=Bom dia
```

Cuidado, quando usamos o operador "+" com números e textos, o resultado é sempre texto (o número é convertido automaticamente);

```
var x=4;
var y="a";
var z=x+y; // o resultado será z="4a"

var x="4";
var y="4";
var z=x+y; //o resultado será z="44"
```

Por que saber sobre tipos é importante?

Apesar de você não declarar os tipos das variáveis, o JavaScript sabe sempre o tipo de dado com que está trabalhando. E para ele, tudo que o usuário escreve é TEXTO, mesmo que o usuário escreva algarismos. Como para TEXTO o sinal de + significa uma coisa diferente do que para os NÚMEROS, isso pode ficar complicado.

Por isso, se numa operação existe um NÚMERO e um TEXTO, o NÚMERO é sempre convertido e a operação ocorre como se tudo fosse TEXTO.

Conversão de tipos

Como comentado anteriormente, quando o usuário entra com dados de qualquer forma (veremos uma forma mais a frente), os valores lidos são sempre textos. Caso seja necessário realizar alguma operação matemática com os valores lidos, precisamos inicialmente convertê-los em números. Para isso temos duas funções em JavaScript que convertem para inteiro ou para *float* (número com casas decimais).

Para converter um texto em número inteiro utilizado a função parseInt.

Sintaxe:

variável = parseInt (valor [, base]);

Para converter um texto em número real (float), utilizamos a função parseFloat.

Sintaxe:

variável = parseFloat (valor);

É uma boa prática inserirmos comentários em nossos códigos, em JavaScript podemos fazer isso de duas formas:

Para Comentários de várias linhas:

/* Inserimos aqui
Nossos comentários
Em várias linhas
*/

Para Comentários de uma única linha:

//Aqui inserimos somente uma linha de comentário

Comandos e Funções

Os Comandos sempre chamam Funções (por isso os Comandos também são chamados de 'Chamadas de Funções'). E essas Funções são criadas assim:

function <nome da função> ([<lista de parâmetros>])

<bloco de instruções>

Instruções Simples X Bloco de Instruções

Para entender a função, primeiro precisamos saber o que significa essa [<lista de parâmetros>]

Uma Instrução Simples sempre termina com ";"

var x = 12;

Ex:

Um Bloco de Instruções começa com um "{" e termina com um "}". E dentro dele vai uma série de instruções:

```
{
var x = 12;
alert(x);
}
```

Para que servem os Blocos de Instrução?

- Rotina de um comando – Função
- Execução condicional
- Repetição

Estruturas de decisão – execução condicional

São blocos de código que serão executados somente se uma dada condição for satisfeita.

Na nossa rotina, estamos sempre tomando decisões, por exemplo:

- se estiver chovendo levo o guarda-chuva.
- se for final de semana e estiver sol irei para a praia.

Temos disponível em JavaScript as seguintes estruturas de decisões:

- if
- if ... else
- if ... else if ... else...
- switch

Operador	Nome	Exemplo	Resultado
==	Igual	a == b	Verdadeiro se a for igual a b
!=	Diferente	a != b	Verdadeiro se a não for igual a b
===	Idêntico	a === b	Verdadeiro se a for igual a b e for do mesmo tipo
<	Menor que	a < b	Verdadeiro se a for menor que b
>	Maior que	a > b	Verdadeiro se a for maior que b
<=	Menor ou igual	a <= b	Verdadeiro se a for menor ou igual a b.
>=	Maior ou igual	a >= b	Verdadeiro se a for maior ou igual a b.

Operadores Lógicos

Operador	Descrição	Exemplo(s), supondo a=3 e b=5

&&	E lógico: retorna verdadeiro se ambas as expressões são verdadeiras e falso nos demais casos	a==3 && b<10 // retorna verdadeiro
		a!=3 && b==5 // retorna falso

||	OU lógico: retorna verdadeiro se pelo menos uma das expressões é verdadeira e falso se todas são falsas	a==3 || b<10 // retorna verdadeiro
		a!=3 || b==5 // retorna verdadeiro
		a==1 || b==3 // retorna falso

!	NÃO lógico: retorna verdadeiro se o operando é falso e vice-versa	! (a==3) // retorna falso ! (a!=3) // retorna verdadeiro

O resultado destas operações sempre será FALSO ou VERDADEIRO

expressão1	expressão2	resultado && - expressão1 && expressão2
FALSO	FALSO	FALSO
FALSO	VERDADEIRO	FALSO
VERDADEIRO	FALSO	FALSO
VERDADEIRO	VERDADEIRO	VERDADEIRO

expressão	!expressão
FALSO	VERDADEIRO
VERDADEIRO	FALSO

Estrutura de decisão if

Decisão Simples (if)

Sintaxe:

```
if (condição)
{
//instruções caso a condição seja verdadeira
}
```

Exemplo:

```
<script language=”javascript”>
var data_hora = new Date();
var hora = data_hora.getHours();
if (hora < 10)
{
alert(“Bom dia!!!”);
}
</script>
```

Na condição da estrutura, utilizamos os operadores de comparação e os operadores lógicos. Se a condição for verdadeira, o bloco é executado.

Estrutura de decisão if ... else ...

Decisão Composta (if ... else ...)

Sintaxe:

```
if (condição)
{
//instruções caso a condição seja verdadeira
}
else
{
//instruções caso a condição seja falsa
}
```

Estrutura de decisão if ... else ...

Exemplo:

```
<script language=”javascript”>
var data_hora = new Date();
var hora = data_hora.getHours();
if (hora < 10){
alert(“Bom dia!!!”);
}
else{
alert(“Olá, tenha um bom dia!!!”);
}
</script>
```

* Se a condição for verdadeira, o bloco do if é executado, senão, se a condição for falsa, o bloco do else é que será executado.

Estrutura de decisão if (encadeados)

Decisão Encadeada (if ... else ... if ... else ...)

Sintaxe:

```
if (condição 1){
//instruções 1
}
else if (condição 2){
//instruções 2
}
else{
//instruções 3
}
```

A decisão encadeada seria um conjunto de if's, para cada IF devemos ter uma condição e poderemos caso necessário colocar o else (senão). Também podemos ter um bloco if dentro de um outro bloco if, ou dentro de um bloco else.

```
<script language="javascript">
var data_hora = new Date()
var hora = data_hora.getHours()
var previsao_tempo = "chuvoso"
if (hora < 10){
  alert("Bom dia...");
  if (previsao_tempo == "chuvoso"){
alert("Está chovendo, leve o guarda-chuva")
  }
}
else if (hora > 12 && hora < 18){
  alert("Boa tarde...");
}
else{
  alert("Boa noite...");
}
</script>
```

Estrutura de decisão switch ... case

Utilizado quanto temos várias condições simples

```
switch (valor) {
    case valor1:
    //instruções 1
    break

case valor2:
    //instruções 2
    break

case valor3:
    //instruções 3
    break

default:
    //instruções padrão
}
```

Para cada caso devemos colocar o comando break, este comando irá finalizar o caso e evitar que o caso posterior seja executado.

O default é opcional, ele é executado quando nenhum caso anterior é acionado. Não podemos fazer comparações no case, como x > y, por exemplo.

```
<script language=”javascript”>
var data_hora = new Date();
dia_semana = data_hora.getDay();
switch (dia_semana){
    case 0:
      alert(“Domingo de descanso merecido.”);
      break;
    case 5:
      alert(“Obaaa, sexta-feira.”);
      break;
    case 6:
      alert(“Maravilha, sabadão!!”)
      break;
    default:
      alert(“Semana longaaaa.”);
}
</script>
```

Estruturas de repetição

Utilizadas quando necessitamos repetir um bloco de instruções. Podemos executar um laço de repetição com um número específico de vezes, ou enquanto uma condição for verdadeira. Essas estruturas também são conhecidas como laço de repetição ou loop. Em JavaScript veremos for e while.

Estrutura de repetição for

Utilizada quando sabemos o número de repetições que serão feitas.

Sintaxe:

```
for (valor inicial; condição; incremento/decremento)
{
//instruções que serão repetidas
}
```

Exemplo:

```
<script language=”javascript”>
var cont;
for (cont = 0; cont < 10 ; cont++){
alert(“Número: “ + cont + “<br />”);
}
</script>
```

Estrutura de repetição while

Executa um bloco de instruções enquanto uma certa condição for verdadeira

Sintaxe:

```
while (condição)
{
//instruções
//alteração do valor da condição
}
```

Exemplo:

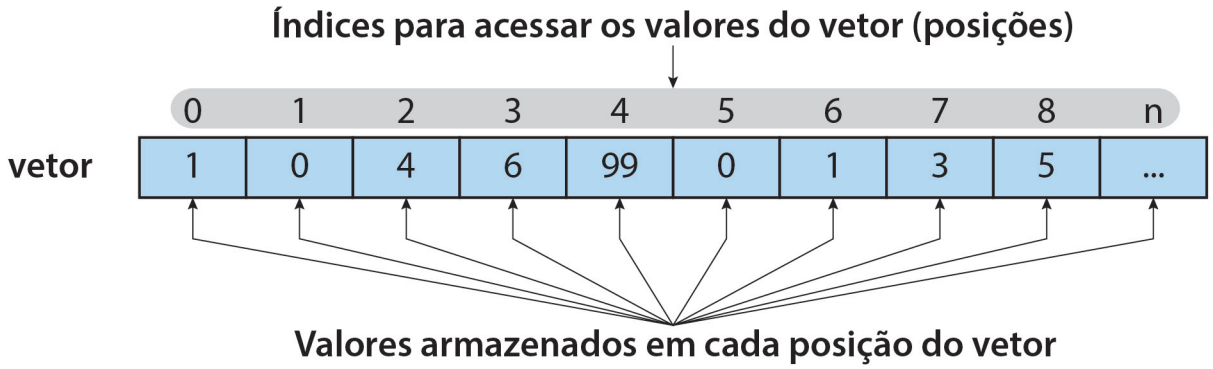
```
<script language=”javascript”>
var cont = 0;
while (cont < 10)
{
alert(“Número: “ + cont + “<br />”);
cont = cont + 1; //ou cont++;
}
</script>
```

Vetores

Um Vetor é uma coleção de espaços na memória, com posições definidas. Podemos atribuí-lo a uma variável, assim ele ganha um nome.

Em uma representação gráfica podemos visualizar um vetor da seguinte forma:

Índices para acessar os valores do vetor (posições)



Vetores em JavaScript

Para criar um vetor em JavaScript utilizamos o objeto Array, podemos criar um vetor de 4 formas:

Sem tamanho fixo (obviamente vazio, sem valores);

Com tamanho fixo (cada posição será vazia também);

Com valores (indicamos os dados que irão ser armazenados em cada posição);

Com valores de forma abreviada;

Criando vetores de JS

```
<script language=“javascript”>
//vetor sem tamanho definido
var vetor1 = new Array();
//vetor com tamanho definido
var vetor2=new Array(3);
//vetor com posições e valores definidos
var vetor3=new Array(“maça”,“banana”,“morango”);
//vetor – declaração abreviada
var vetor4 = [“limão”, “pera”, “uva”, “kiwi”];
</script>
```

OBS: em todos os casos acima, podemos inserir mais posições sempre que necessário.

Acessando um vetor

Para visualizarmos ou atribuirmos um valor para um vetor devemos acessar cada posição através do índice, este índice deve ficar entre colchetes "[...]".

Exemplo:

```
<script language="javascript">
        var frutas = new Array();
        frutas[0]="maça";
        frutas[1]="banana";
        frutas[2]="morango";
        alert(frutas[2]);
    </script>
```

Em quase todos os casos em que trabalhamos com vetores, utilizamos uma estrutura de repetição para acessar suas posições, a mais comum seria a estrutura for, porém também podemos usar o while.

Exemplo

```
<script language="javascript">
    var frutas = new Array();
    frutas[0]="maça";
    frutas[1]="banana";
    frutas[2]="morango";
    document.write(frutas[0] +"<br />");
    document.write(frutas[1] +"<br />");
    document.write(frutas[2] +"<br />");
</script>

script language="javascript">
var i;
var frutas = new Array();
frutas[0]="maça";
frutas[1]="banana";
frutas[2]="morango";
for (i=0;i<=2;i++)
{
    document.write(frutas[i] +"<br />");
}
</script>
```

O tamanho de um vetor

Todo vetor tem a propriedade length, que guarda o número de seus elementos e para interromper o laço, utiliza-se a instrução break.

Material Complementar

Professional JavaScript for Web Developers (Wrox Professional Guides).

http://www.w3schools.com/js/

(site com explicações e funcionalidades em javascript).

Referências

HTML5 – **Guia Prático** – Editora Érica – 2011.

HTML5 – **A linguagem de marcação que revolucionou a web** – Novatec – Maurício Samy Silva.

Anotações


Educação a Distância
Cruzeiro do Sul Educacional
Campus Virtual

www.cruzeirodosulvirtual.com.br
Campus Liberdade
Rua Galvão Bueno, 868
CEP 01506-000
São Paulo SP Brasil
Tel: (55 11) 3385-3000

Universidade Cruzeiro do Sul

UNICID Universidade Cidade de S. Paulo

UNIFRAN Universidade de Franca

UDF Centro Universitário

Módulo Centro Universitário