Algoritmos em Javascript

Sumário

Algoritmos

Um algoritmo é uma seqüência de instruções definidas para serem executadas mecanicamente. Todos os dias realizamos algoritmos e não percebemos. Basicamente todas as manhãs nós nos levantamos da cama, tomamos o café da manhã, nos arrumamos e vamos ao trabalho. Entre cada uma dessas ações descritas, realizamos várias outras ações. Nosso próprio corpo realiza algoritmos o tempo inteiro realizando decisões. Se nos machucamos, sentimos dor. Se o ambiente externo tem temperatura baixa, o corpo treme para tentar aumentar a temperatura novamente.

Todo algoritmo segue uma seqüência lógica. Geralmente um algoritmo é lido de cima pra baixo e da esquerda pra direita. Cada linha de código é lida e, caso não haja nenhum erro, executada e logo depois é lida a próxima linha e assim sucessivamente até terminarem as linhas do algoritmo.

Como exemplo de algoritmo podemos pensar em um ser humano andando. Observe:

1. Levantar a perna direita
2. Inclinar o corpo para frente
3. Baixar a perna direita
4. Apoiar o pé direito no chão
5. Levantar a perna esquerda
6. Baixar a perna esquerda
7. Apoiar o pé esquerdo no chão
8. Levantar a perna direita...

Outro exemplo de algoritmo seria fazer um calculo simples como a soma ou multiplicação de dois números como no exemplo abaixo:

1. Ler primeiro número
2. Ler segundo número
3. Somar os dois números lidos
4. Exibir o resultado

O que é um programa?

Um programa é um ou mais algoritmos escritos numa linguagem de programação. Existem diversas linguagens de programação atualmente. A grosso modo as linguagens mais fáceis são chamadas de “linguagens de alto nível” e as mais difíceis são chamadas de “linguagens de baixo nível”. As Linguagens de Baixo nível são assim chamadas por que você precisa trabalhar mandando comandos e recebendo respostas direto para o hardware. As Linguagens de Alto nível são assim chamadas por que elas “escondem” as tarefas difíceis como trabalhar direto com hardware, o que facilita o desenvolvimento do código e aumenta a velocidade de programação.

Exercícios:

1. Escreva um algoritmo para pentear o cabelo. Tente ser bem especifico e detalhado em cada ação realizada.

Entrada e saída de dados

Muitas vezes não percebemos, mas para haver o processamento de informações, é necessário ter informações para que elas sejam processadas. Um programa precisa que dados sejam inseridos nele para que eles sejam processados e retorne algum resultado daquele processamento. Esses dados podem ser inseridos pelo usuário do programa e algumas vezes o usuário não sabe que está inserindo dados. Digitar texto em um formulário eletrônico ou o simples ato de copiar um arquivo para o seu computador é considerado entrada de dados.

Exercício: Escreva alguns modos de entrada e saída de dados. Ao lado de cada um, diga se é entrada ou saída.

Programando

A partir de agora, utilizaremos uma linguagem de programação. Assim podemos ver melhor os resultados e realmente programar. Utilizaremos a linguagem Javascript por ser fácil e não ser necessário a instalação de nenhum programa na maioria dos sistemas operacionais atuais no mercado, entretanto é recomendado que você utilize um editor de textos que faça a coloração do seu código para que fique mais fácil o aprendizado. Nos exemplos da apostila será utilizado o GVim (http://www.vim.org/).

Antes de começarmos a escrever é necessário saber o que é o Javascript. Javascript foi inicialmente criada pela Netscape em 1995 para se desenvolver websites dinâmicos tendo em vista que no inicio da internet todos os sites eram estáticos, ou seja, não haviam mudanças em seu comportamento inicial. A linguagem tem total integração com o HTML por meio do DOM (Document Object Model), assim a mesma pode manipular qualquer elemento em uma página. Devido ao seu sucesso, o Javascript hoje é suportado pela maioria dos navegadores que seguem os padrões W3C () e outros no mercado.

O que é necessário para programar em Javascript?

Apenas um navegador, como o Mozilla Firefox, Internet Explorer, Google Chrome e Safari, e um editor de texto, como o Bloco de Notas (Microsoft Windows), GEdit (GNU/Linux com a interface Gnome) ou Vi (Utilizado em grande parte das distribuições GNU/Linux), pode ser utilizado para programar em Javascript. Recomendo que utilize um editor que tenha highlight (utiliza cores para mostrar diferentes elementos no programa), como o Notepad++ (http://notepad-plus.sourceforge.net/), para que a programação seja mais fácil e encontrar erros mais rapidamente. É necessário também salvar o arquivo com a extensão ".htm" ou ".html" para que o navegador execute o que está escrito no arquivo.

Variáveis

Variáveis são espaços em memória onde podemos alocar algum valor temporariamente. Como o nome diz, o valor de uma variável pode ser alterado.

```
<script>
        var num;
</script>
```

Acabamos de inicializar uma variável, mas a mesma ainda não tem valor algum. A seguir, um exemplo de uma variável recebendo valores.

```
<script>
        var num;
        num = 10;
        num = 20;
        document.write(num);
</script>
```

O código acima cria a variável “num”, atribui o número 10 como valor para ela e por fim imprime o valor da variável em tela com o comando “document.write”. Sempre que precisar imprimir um texto, um número ou o valor de uma variável utilize-o.

Após esses pequenos exemplos já podemos definir algumas regras para trabalhar com o Javascript com linguagem.

1 – Sempre que for definir criar uma variável, escreva “var” antes do nome da mesma. Isso será bastante útil e seu benefício será mostrado quando falarmos sobre escopo de variável mais à frente;

2 – Toda linha deve terminar com o sinal de “;” (Ponto e vírgula). Assim o interpretador saberá que aquela linha de código acabou.

3 – Um código em Javascript deve ser colocado entre as TAGS “<script>” e” </script>”. Isso é apenas para realizar nossos algoritmos. Caso você trabalhe com Html seguindo os padrões W3C, será necessário mais informações dentro da Tag “<script>”, mas esse não é o nosso foco no momento.

Tipos de Variáreis

O Javascript tem três tipos de variáveis: Numérico, Booleano e Cadeias de Caracteres.
As variáveis numéricas contem números, sejam eles inteiros ou de ponto flutuante e podem ser utilizadas para fazer cálculos de qualquer tipo.
Exemplo:

```
<script>
        var num;
        num = 10;
        var num2 = 20;
</script>
```

Os dois modos para criação de variáveis e atribuição de valores estão corretos. Podemos declarar uma variável e depois atribuir um valor à ela ou no ato da declaração da variável, definir um valor à ela.

As variáreis booleanas podem conter apenas dois valores: “verdadeiro” ou “falso”.
Exemplo:

```
<script>
        var verdadeiro = true;
        var falso = false;
</script>
```

As variáveis do tipo cadeia de caracteres podem receber qualquer tipo de texto. Para isso é necessário colocar aspas duplas (“) ou simples (‘) no início e no fim da cadeia de caracteres.

Exemplo:

```
<script>
        var nome = "Meu nome";
        var endereco = 'Meu endereco';
</script>
```

Exercícios:

1 – Esse exercício é apenas para fixação de conceitos. Crie variáveis com nomes diferentes e atribua valores a elas. Imprima o valor das variáveis com o comando “document.write”.

Obs.: Para fazer uma impressão por linha, você pode utilizar a tag de HTML “
”. Para isso, basta escrever o código como no exemplo abaixo.

```
<script>
        var nome = "João da Silva";
        var endereco = 'Rua do João';

        document.write(nome + "<br>");
        document.write(endereco + "<br>");

</script>
```

No exemplo acima, estamos utilizando o sinal “+” para concatenar as cadeias de caracteres, que agora chamaremos de String. Concatenar significa ligar, então utilizaremos a palavra “concatenar” para dizer que estamos ligando duas Strings.

Arrays

Traduzir “array” de um modo fácil seria dizer que ele é uma “variável que tem o valor de várias variáveis”. Dentro de um array é possível colocar vários valores diferentes.

```
<script>
    meuArray = new Array();
    meuArray[0] = 10;
    meuArray[1] = "José da Silva";

    document.write(meuArray[0]);
</script>
```

Para utilizar um array é necessário que antes de atribuir valores aos seus índices (Um índice é um espaço dentro do array onde podemos armazenar dados), você o declare com o comando “new Array()”. Após isso basta atribuir valores aos seus índices, que podem ser numéricos ou associativos.

Índices Numéricos

São os índices de um array representados por números.

```
<script>
    estados = new Array();

    estados[0] = "Rio de Janeiro";
    estados[1] = "São Paulo";
    estados[2] = "Minas Gerais";
    estados[3] = "Espirito Santo";
    estados[4] = "Bahia";

    document.write(estados[3]);
</script>
```

Do mesmo modo que você atribui valores aos índices, você pode acessar seus valores utilizando o nome do array e seu índice entre chaves como no exemplo acima. Se você atribuir dois valores a um mesmo índice, o mesmo será substituído.

Índices Associativos

Índices Associativos funcionam como os numéricos, mas ao invés de números, strings são utilizados para sinalizar um índice.

```
<script>
    estados = new Array();

    estados["rj"] = "Rio de Janeiro";
    estados["sp"] = "São Paulo";
    estados["mg"] = "Minas Gerais";
    estados["es"] = "Espirito Santo";
    estados["ba"] = "Bahia";

    document.write(estados["rj"]);
</script>
```

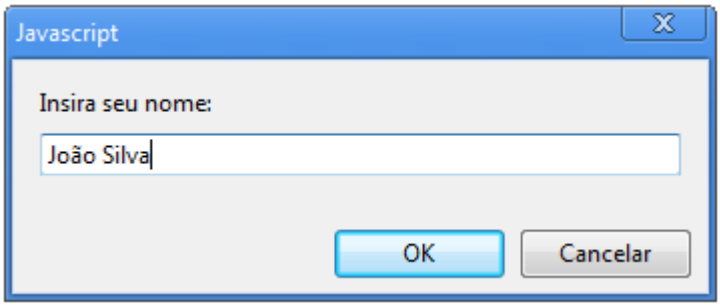
Pegando dados externos

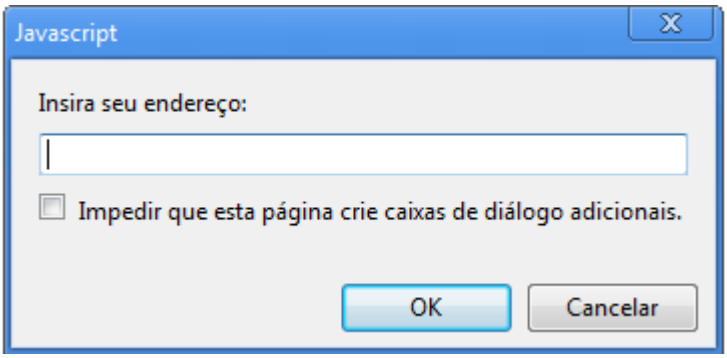
Como já falamos anteriormente, para um programa funcionar ele deve ter uma entrada de dados para que eles sejam processados e serem enviados para uma saída. Utilizando o comando “window.prompt” do seguinte modo:

```
<script>
        var nome = window.prompt("Insira seu nome:");
        var endereco = window.prompt("Insira seu endereço:");

        document.write(nome + "<br>");
        document.write(endereco + "<br>");
</script>
```

Ao executar o código acima, serão exibidas duas janelas no navegador, uma para cada comando “window.prompt” e o que for digitado será inserido na variável correspondente.





A opção “Impedir que esta página crie caixas de diálogo adicionais” foi adicionada pelo próprio navegador.
Agora que podemos pegar dados dos usuários, podemos utilizar esses dados para que nossos programas comecem a ter vida.

Exercício:
1 – Utilize as caixas de prompt (window.prompt) e faça um pequeno programa que peça alguns dados para o usuário e os imprima em tela, um por linha.

Operadores

Operador de Atribuição

Como já vimos anteriormente, para atribuir um valor a uma variável, é utilizado o sinal de “=”. Você pode atribuir valores numéricos, Strings e booleanos às suas variáveis.

Operadores Aritméticos

Operadores Aritméticos servem para calcular números e valores em variáveis. Em Javascript nós temos os seguintes Operadores Aritméticos:

Adição (+)
Subtração (-)
Multiplicação (*)
Divisão (/)
Resto da Divisão (%)

Utilizaremos o sinal de % quando precisarmos ter o resto de uma divisão, diferente de / que retorna o resultado de um número dividido pelo outro.

Exemplo:

```
<script>
        var num1 = 10;
        var num2 = 2;

        var num3 = num1 + num2;
        document.write(num3 + "<br>");

        num3 = num1 - num2;
        document.write(num3 + "<br>");

        num3 = num1 * num2;
        document.write(num3 + "<br>");

        num3 = num1 / num2;
        document.write(num3 + "<br>");


        num3 = num1 % num2;
        document.write(num3 + "<br>");
</script>
```

Operadores Relacionais

Os operadores relacionais são utilizados quando você precisa descobrir a relação entre um valor e outro. Os Operadores Relacionais são:

>	Maior que...
<	Menor que...
>=	Maior ou igual que...
<=	Menor ou igual que...
==	Igual a...
!=	Diferente de...

Os Operadores Relacionais retornam sempre um valor booleano, ou seja, true (verdadeiro) ou false (false). Podemos testar os operadores do seguinte modo:

```
<script>
        var retorno;

        retorno = 10 > 15;
        document.write(retorno + "<br>");

        retorno = 10 < 15;
        document.write(retorno + "<br>");

        retorno = 10 >= 15;
        document.write(retorno + "<br>");

        retorno = 10 <= 15;
        document.write(retorno + "<br>");

        retorno = 10 == 15;
        document.write(retorno + "<br>");

        retorno = 10 != 15;
        document.write(retorno + "<br>");
</script>
```

Operadores Lógicos

Os operadores lógicos servem para se utilizar várias operações relacionais em uma mesma linha.

&& E
|| Ou
! Não

Os Operadores Lógicos “&&” e “||” seguem a lógica da tabela verdade, descrita abaixo:

A	B	A && B
Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Falso
Falso	Verdadeiro	Falso
Falso	Falso	Falso

A	B	A || B
Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Verdadeiro
Falso	Verdadeiro	Verdadeiro
Falso	Falso	Falso

Incremento e Decremento

Para incrementar em 1 uma variável, basta utilizar duas vezes o sinal “+” após a variável. Para decrementar em um, faça o mesmo, porém utilizando o sinal de “-”.
Exemplo:

```
<script>
        var num = 1;

        document.write(num + "<br>");

        num++; //Aqui fizemos um incremento
        document.write(num + "<br>");

        num++;
        document.write(num + "<br>");

        num--; //Aqui fizemos um decremento
        document.write(num + "<br>");
</script>
```

Comentários

Comentários são utilizados quando você precisa descrever parte de um código ou apenas para sinalizar algo importante como um lembrete. Em Javascript existem comentários de linha e de bloco. É importante sempre fazer comentários quando for necessário.

```
<script>
    //Esse é um comentário de linha

    /*
        Esse é um comentário de bloco
    */
</script>
```

O interpretador do Javascript irá ignorar tudo o que estiver comentado. No comentário de linha, tudo o que estiver após as barras será ignorado. Já no comentário de bloco, tudo o que estiver entre o “/*” e o “*/” será ignorado.

Funções

Algumas vezes será necessário que você execute uma mesma rotina várias vezes. Para que você não precise reescrever o mesmo código várias vezes, existem as funções. Uma função permite que você passe parâmetros para ela. A função irá realizar o bloco de código definido dentro das chaves com ou sem os parâmetros passados.
Exemplo:

```
<script>
   function soma(num1,num2){
      return num1 + num2;
   }
</script>
```

No exemplo anterior nós temos a função “soma” que recebe dois números. Ela soma esses dois números e retorna o resultado da soma.
O comando *return* utilizado no final da função retorna a soma das duas variáveis que pode ser utilizado do lado de fora da função como mostra o exemplo baixo.

```
<script>
   function soma(num1,num2){
      return num1 + num2;
   }


   var retorno = soma(10,15);
   document.write("O resultado da soma é: " + retorno);
</script>
```

Estruturas de Controle

As estruturas de controles servem para, como o nome diz, controlar a execução do programa.

IF

Imagine que no sistema de uma loja virtual apenas maiores de 18 anos de idade podem realizar compras. Para isso é necessário checar se a idade do comprador é maior ou igual a 18 anos. Utilizaremos a estrutura de controle IF (Se) que serve para executar determinado código caso uma expressão retorne true.
Exemplo:

```
<script>
   var idade = 18;

   if(idade >= 18){
      document.write("O usuário pode comprar.");
   }
</script>
```

Nesse exemplo será testado se a idade do usuário é igual ou maior que 18 anos. Se o teste entre parêntesis retornar true, será executado o código entre as chaves. Esse exemplo é bastante limitado levando em consideração que nada acontecerá se o usuário tiver idade menor de 18 anos. Sempre que houver a necessidade de que um bloco de código seja executado se a expressão retornar false, basta adicionar a o bloco do ELSE (SENÃO) após os parêntesis do IF.

```
<script>
   var idade = 18;

   if(idade >= 18){
      document.write("O usuário pode comprar.");
   }
   else{
      document.write("O usuário não pode comprar");
   }
</script>
```

Se houver a necessidade de fazer mais de um teste, basta utilizar o ELSE IF.

```
<script>
   var valorProduto = 30;

   if(valorProduto <= 10){
      document.write("Permitido apenas compra à vista.");
   }
   else if(valorProduto >10 && valorProduto <= 20){
      document.write("Permitido financiamento em 3 vezes sem juros.");
   }
   else{
      document.write("Permitido financiamento em até 12 vezes sem juros.");
   }
</script>
```

Você pode perceber que utilizamos o operador lógico && para dizer ao ELSE IF que o valor do produto deve ser maior que 10 e menor ou igual a 20. Você pode utilizar quantos ELSE IF forem necessários.

Switch

O Switch é utilizado quando você precisa de comparações simples como, por exemplo, comparar se a letra digitada é uma vogal ou consoante.

```
<script>
   var letra = 'a';

   switch(letra){
      case 'a':
         document.write("A letra digitada é uma vogal.");
         break;
      case 'e':
         document.write("A letra digitada é uma vogal.");
         break;
      case 'i':
         document.write("A letra digitada é uma vogal.");
         break;
      case 'o':
         document.write("A letra digitada é uma vogal.");
         break;
      case 'u';
         document.write("A letra digitada é uma vogal.");
         break;
      default:
         document.write("A letra digitada é uma consoante.");
   }
</script>
```

Estruturas de Repetição

As estruturas de Repetição são utilizadas quando você precisa repetir um código várias vezes enquanto uma determinada condição não for satisfeita. Em Javascript existem 3 estruturas de repetição.

While

O while irá executar um bloco de código até que sua condição seja satisfeita. Veja o exemplo:

```
<script>
    var num = 10;

    while(num < 20){
        document.write("O valor de num é menor que 20.");
    }
</script>
```

Acima temos um problema. O valor de num nunca será igual ou maior que 20, então o programa entrará no que chamamos de loop infinito. Ele entrará nesse bloco de código e executará infinitamente, geralmente travando a o seu navegador. Para que isso não aconteça, é necessário que você faça com que a variável num receba um valor que não satisfaça a condição do while. Um exemplo simples seria incrementar a variável a cada loop.

```
<script>
    var num = 10;

    while(num < 20){
        document.write("O valor de num é menor que 20.");
        num++;
    }
</script>
```

Pronto. Agora o while executará o código um número limitado de vezes e sairá logo assim que a condição não mais o satisfizer.

For

O for funciona do mesmo modo que o while, mas utilizando ele fica mais fácil não esquecer declarar uma variável ou incrementá-la. O For apresenta uma sintaxe mais definida.

```
<script>
    for(num = 0; num < 10; num++){
        document.write("A variável agora vale " + num);
    }
</script>
```

Podemos ver que o for recebe 3 parâmetros. O primeiro é a variável com o seu valor inicial. O segundo é a condição que deve ser satisfeita e no terceiro parâmetro você pode incrementar a variável como eu fiz ou atualizar o valor dela.

Do/While

O Do/While é mais parecido com o while, porém ao invés de ele fazer o teste antes da execução do bloco, seu teste é feito apenas depois.

```
<script>
    var num = 10;
    do{
        document.write("O valor de num é menor que 10");
    }while(num < 10 )
</script>
```

A grande diferença entre o While e o Do/While é que no do/while o código no bloco será executado pelo menos uma vez, mesmo se a condição não for satisfeita.