JavaScript

Umas das linguagens de programação mais utilizadas no mundo é o Javascript. Essa linguagem é muito versátil e pode ser utilizada nos mais diversos cenários, desde programação para servidores até para a criação de páginas Web mais dinâmicas, sendo executada no navegador. É usado, portanto, para tornar suas páginas mais interativas para o usuário.

O JavaScript é uma linguagem de script que, quando incorporada a uma página Web, permite incrementar a apresentação e interatividade da mesma. Essa linguagem permite ao programador ter acesso a elementos de uma página web, como imagens, campos de um formulário, links etc. Todo esse controle é executado no lado do cliente (navegador Web), sem precisar necessariamente entrar em contato com o servidor. Evitando essa comunicação, diminuímos o tempo de resposta do sistema e também o tráfego na rede. Não é legal?

Sendo assim, os elementos de uma página HTML podem ser modificados via programação, mesmo após a página ter sido carregada e exibida para o usuário. O JavaScript permite, por exemplo,

associar código a eventos como o click do mouse ou uma tecla pressionada de seu teclado.

O JavaScript foi desenvolvido inicialmente pela Netscape e era conhecido pelo nome de LiveScript. Adotado no fim do ano de 1995 pela Sun (que também desenvolveu o Java), ela recebeu o seu nome atual: JavaScript. Apesar de ter sido criado por uma empresa criadora de um navegador Web (o Nestcape), a linguagem JavaScript não ficou exclusiva do Netscape. No decorrer dos tempos, outros navegadores também incorporaram o JavaScript, de forma que atualmente a maioria dos navegadores Web dão suporte à mesma.

Apesar disso, podem existir alguns problemas de compatibilidade entre diferentes navegadores Web. Isso porque existem pequenas variações na linguagem. Códigos específicos que rodam em determinados tipos de navegadores e que não são suportados por outros.

A sintaxe do JavaScript é baseada na linguagem Java, mas a similaridade entre as duas não é muito grande, visto que elas possuem objetivos e características diferentes, como pode ser visto no Quadro 1:

Javascript	**Java**
Interpretada pelo navegador enquanto ele lê a página HTML.	Compilada em bytecode e executada normalmente fora do navegador pela máquina virtual Java.

Javascript	Java
Programação simples com poder limitado (indicado para tarefas simples e pontuais, ou seja, para o desenvolvimento de pequenos scripts).	Linguagem de programação mais complexa, porém mais poderosa, de uso geral.
Baseada em objetos (possui objetos, mas não suporta todos os conceitos de orientação a objetos).	Orientada a objetos.
Referências a variáveis são checadas em tempo de execução.	Referências a variáveis são checadas em tempo de compilação.
Os tipos das variáveis não são declarados.	Fortemente tipada (os tipos das variáveis devem ser definidos na declaração da mesma)

Diferenças entre JavaScript e Java.

Uma exceção de código Java rodando no navegador Web são os Applets que, ainda assim, não são embutidos junto ao código HTML, como é o caso de JavaScript.

Primeiros Passos

Para começar a desenvolver os primeiros scripts em JavaScript, poucos requisitos são necessários,como descritos a seguir:

- **Conhecimento de HTML.**
    - Como o JavaScript é embutido dentro de uma página HTML e manipula os elementos desta, é indispensável o conhecimento de HTML para construir páginas Web interativas com JavaScript.
- **Navegador Web que suporte JavaScript.**
    - Os navegadores atuais mais populares dão esse suporte, como é o caso do Internet Explorer, Firefox e Chrome.
- **Editor de Texto.**
    - Assim como código HTML, qualquer editor pode ser usado para desenvolver código JavaScript. Além do mais, existem diversos editores específicos para HTML que também facilitam o desenvolvimento de código JavaScript, como o Visual Code por exemplo.

Como já foi comentado antes, o código JavaScript é embutido dentro de uma página HTML, mais especificamente, ele é colocado dentro da tag <SCRIPT> do HTML. O trecho de código a seguir demonstra como o JavaScript funciona integrado ao código HTML.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Primeiro Exemplo</title>
    </head>
    <body>
        <script language="javascript">
            alert("Olá Mundo!!!");
        </script>
    </body>
</html>
```

Essa página, ao ser carregada, vai exibir uma caixa de diálogo com o texto “Olá mundo” e um botão OK. Isso é feito através da função alert() do JavaScript.

Outro exemplo simples do JavaScript é imprimir uma mensagem diretamente na página HTML. Para isso, você pode usar o comando documento.write() para adicionar textos dentro da página HTML, utilizando inclusive tags HTML para incrementar o texto, como pode ser visto no exemplo a seguir:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Primeiro Exemplo</title>
    </head>
    <body>
        <script language="javascript">
            document.write("<h1>Olá Mundo</h1>");
        </script>
    </body>
</html>
```

Nos dois exemplos, a tag <script> foi incluída dentro do corpo da página (tag <body>). Entretanto, isso não é uma regra, pois às vezes o JavaScript é incluído dentro da tag <head>. Dentro da tag <body> ela pode aparecer em qualquer ordem em relação às outras tags, tais como: <img>, <form> etc., bastando respeitar a sintaxe do HTML (regras de aninhamento de tags).

Uma coisa que você deve notar é que o browser carrega uma página Web lendo seu conteúdo de cima para baixo, assim como uma pessoa geralmente faz ao ler o arquivo. Muitas vezes, instruções HTML carregadas no início do arquivo só poderão ser executadas após o navegador ter carregado todos os elementos necessários para sua execução. Isso significa dizer que os elementos a serem manipulados

por uma instrução JavaScript devem ser carregados antes para que a instrução possa ser executada.

Por exemplo, o código a seguir pode não funcionar corretamente, visto que o código JavaScript, representado pela instrução entre as linhas 7 e 9, acessa a imagem com id=”ordem”, que está definida somente na linha 10.

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo Errado</title>
</head>
<body>
    <script language="javascript">
    //Código que manipula a imagem com id="ordem"
    </script>
    <img src="ordem.gif" id="ordem" />
</body>
</html>
```

Como era de se esperar, o mesmo problema ocorreria se um script tentasse acessar uma variável ou função que só está declarada depois dela. Por essa razão, para assegurar que scripts funcionem corretamente, é uma boa prática declarar funções e variáveis que são usadas em vários lugares dentro da tag <head>. Como essa tag vem antes do <body>, seus elementos serão naturalmente carregados antes de qualquer elemento que estiver do <body>, como é apresentado no trecho a seguir:

```
<!DOCTYPE html>
<html>
<head>
    <title>Script no head</title>
          <script language="javascript">
                    var mensagem = "Alô Mundo!!!";
          </script>
     </head>
     <body>
```

```
            <script language="javascript">
                        document.write(mensagem);
            </script>
        </body>
    </html>
```

Como pode ser visto, a instrução da linha 10 usa uma variável chamada mensagem, que foi definida na linha 6. Apesar do exemplo mostrar somente um bloco de <script> dentro do <head> e outro dentro do <body>, é possível colocar vários blocos de <script> nos dois casos, como pode ser visto no código abaixo.

```
<!DOCTYPE html >
<html>
    <head>
          <title>Script no head</title>
        <script language="javascript">
              var mensagem = "Alô Mundo!!!"";
        </script>
        <script language="javascript">
              var idade = 3;
        </script>
    </head>
    <body>
        <script language="javascript">
              alert(mensagem);
         </script>
        <script language="javascript">
              alert(idade);
         </script>
    </body>
</html>
```

Uma outra maneira de usar código JavaScript sem ter que necessariamente misturar os comandos HTML com JavaScript é utilizar referências para arquivos externos que possuam código JavaScript.

No exemplo a seguir dois arquivos são criados, um contendo somente código HTML e outro contendo somente código JavaScript.

Arquivo HTML:

```
<html>
   <head>
     <title>Script no head</title>

     <script language="javascript" src="./codigo.js" ></script>
   </head>
    <body>
    </body>
</html>
```

Arquivo JavaScript (codigo.js):

```
var descricao = "Esta é uma descricao";
alert(descricao);
```

Sintaxe

Como já foi comentado anteriormente, a sintaxe de JavaScript é bem familiar para quem já conhece Java. Iremos descrevê-la em detalhes.

Em particular, os seguintes elementos serão abordados:

- Variáveis e constantes;
- Operadores;
- Comandos;
- Funções;
- Eventos;

Variáveis e constantes

Em JavaScript, uma variável é declarada usando-se a palavra *var*, seguida do nome da variável. O nome de uma variável deve começar por uma letra ou o símbolo “_”, seguido por qualquer quantidade de letras, números e de caracteres “_” e $. Além disso,

os nomes de variáveis são case sensitive, ou seja, o JavaScript diferencia letras maiúsculas e minúsculas, assim como acontece na linguagem Java. O trecho que segue apresenta algumas declarações válidas de variáveis.

```
<script language="javascript">
    var mensagem = "Alô Mundo!!!";
    var idade = 23;
    var _altura = 2.32;
    var esta_contido = true;
</script>
```

Diferentemente da linguagem Java, em JavaScript não é necessário haver a especificação do tipo da variável. Nessa linguagem a declaração de variáveis é feita apenas como indicado anteriormente, com o nome e a palavra-chave *var*. Isso cria a possibilidade de uma variável assumir valores de tipos diferentes ao longo da execução de um script! É isso mesmo! Em alguns momentos do programa uma variável pode estar armazenando um número, depois pode estar armazenando uma string e assim por diante.

No exemplo anterior, a variável mensagem é do tipo string, enquanto as variáveis idade e altura são do tipo inteiro e decimal, respectivamente.

Veja a seguir um exemplo de código JavaScript que muda o tipo do dado armazenado em uma variável:

```
var num = 1;
num = “agora armazeno uma string”;
```

A tabela a seguir faz um resumo dos tipos de dados primitivos que as variáveis JavaScript podem assumir.

Tipo	Descrição
Números	Podem ser números inteiros ou números decimais (valores com ponto).
Cadeias de caracteres	Qualquer cadeia de caractere entre ' (aspas simples) ou “ (aspas duplas).
Booleanos	As palavras true para verdadeiro e false para falso.
Nulo	Valor null que representa a ausência de um valor.

As constantes possuem a mesma função das variáveis, mas seu valor não pode ser alterado. Para se declarar constantes, usa-se a palavra const antes do nome da constante.

```
<script language="javascript">
     const maximo = 100;
</script>
```

Assim como outras linguagens de programação, as variáveis podem ser **locais** ou **globais**. Variáveis globais podem ser usadas por qualquer trecho de código JavaScript presente na página Web (desde que o código venha depois da declaração da variável). Elas são declaradas fora e antes de todas as funções que a usam, mas podem estar contidas em qualquer local da página.

Já as variáveis locais são declaradas e invocadas somente dentro da função. Apesar de vermos o assunto de funções em detalhes mais adiante, a seguir temos um trecho de código JavaScript que

exemplifica o uso de variáveis locais e globais. Como pode ser visto, a variável mensagem é declarada na linha 2 e usada nas linhas 6 e 11, enquanto a variável numero1 e numero2 são locais, sendo definidas dentro das duas funções nas linhas 5 e 10, respectivamente.

```
<script language="javascript">
    var mensagem = "O número é: ";

    function imprime1(){
        var numero1 = 1;
        alert(mensagem);
        alert(numero1);
    }

    function imprime2(){
        var numero2 = 2;
        alert(mensagem);
        alert(numero2);
    }
</script>
```

Operadores

Os operadores são úteis para manipular as variáveis, em JavaScript existem diversos operadores, veja quais são eles.

a) Operadores Matemáticos

São os operadores que permitem a realização de operações aritméticas sob as variáveis.

Símbolo	Significado	Exemplo
+	Adição	x + 7
-	Subtração	x - 7
*	Multiplicação	x * 8

Símbolo	Significado	Exemplo
/	Divisão	x / 8
%	Resto	x % 8
=	Atribuição	x = 8

b) Operadores lógicos

São os operadores que permitem a realização de operações lógicas sob as variáveis. São normalmente utilizados em comandos condicional (como o if) e repetições (como o while). São aplicáveis a valores booleanos (true e false) e seu resultado também é um booleano.

Símbolo	**Nome**	**Descrição**	**Exemplo**
&&	E	Verdadeiro se as duas condições são verdadeiras	x && y
||	Ou	Verdadeiro se pelo menos uma das condições é verdadeira	x || y
!	Negação	Retorna o valor booleano inverso	!x

c) Operadores de Comparação

São os operadores que permitem a realização de operações de comparação sob variáveis. São muito utilizados em comandos condicional (como o if) e repetições (como o while), pois o resultado de sua avaliação é um valor booleano.

Símbolo	Significado	Exemplo
==	Verifica igualdade	x == 11
<	Verifica se valor menor que outro	x < 11
<=	Verifica se valor menor ou igual a outro	x <= 11
>	Verifica se valor maior que outro	x > 11
>=	Verifica se valor maior ou igual a outro	x >= 11
!=	Verifica se valor diferente de outro	x != 11

d) Operadores combinados

São operadores que simplificam a sintaxe de operações aritméticas bem comuns, servindo como atalho mais compacto da operação.

Símbolo	Significado	Exemplo	Alternativa
+=	Incrementa valor	x += y	x = x + y
-=	Decrementa valor	x -= y	x = x - y
*=	Multiplica por	x *= y	x = x * y
/=	Divide por	x /= y	x = x / y
x++	Incrementa em uma unidade	x = y++	y = y + 1 e x = y

Símbolo	Significado	Exemplo	Alternativa
x--	Decrementa em uma unidade	x = y--	y = y - 1 e x = y

Comandos JavaScript

Para começar a programar em JavaScript é preciso que você estude os comandos básicos disponíveis na linguagem, mostrados a seguir.

IF

O comando IF é usado simplesmente para testar uma condição. O exemplo que segue ilustra o uso do if para decidir o que será adicionado como conteúdo da página.

A função prompt() exibe uma mensagem para o usuário e aguarda a digitação de um texto e confirmação do usuário. O trecho abaixo, por exemplo, solicita que o usuário digite um número e depois avalia se esse número é maior ou menor que 10, exibindo uma mensagem diferente para cada caso.

```
<script>
    var i = prompt("digite um número");
    if (i > 10) {
        document.write("é maior que dez");
    }else{
        document.write("é menor ou igual a dez");
    }
</script>
```

WHILE

O comando while, assim como em Java, é usado como comando de repetição dependente de uma condição. Na maioria das vezes, ele é aplicado quando não podemos determinar a prioridade e a quantidade de repetições que nosso laço vai ter.

No próximo exemplo é apresentado um script que adiciona
 (quebra de linha) enquanto i<=10.

```
<script Language="javascript">
    var i=1;
    while(i <= 10){
        i++;
        document.write(i +'<br/>');
    }
</script>
```

FOR

A expressão for permite executar um bloco de instruções determinado número de vezes, de acordo com certo critério.

Neste exemplo é apresentado um script que exibe uma lista não numerada, contendo os números de 1 a 10, um em cada item.

```
<script>
    document.write("<ul>");
    for(count=1; count <= 10; count++){
        document.write( "<li>" + count + "</li>");
    }
    document.write("</ul>");
</script>
```

As instruções break e continue podem ser usadas nos dois tipos de laços. O break permite interromper prematuramente um laço, enquanto o continue permite suspender o ciclo atual e continuar o próximo ciclo do laço.

SWITCH/CASE

A expressão switch serve para avaliar o valor de uma variável e executar um código associado, de acordo com o valor da mesma (o que é definido em cada case). Também é possível definir um default, que é um valor padrão para caso a avaliação não se encaixe em nenhum dos case. O exemplo abaixo demonstra o uso do switch/case.

```
<script>
    var numero = prompt("digite um valor de 1 até 5");
    switch(numero){
        case '1':
            alert('seu valor foi 1');
            break;
        case '2':
            alert('seu valor foi 2');
            break;
        case '3':
            alert('seu valor foi 3');
            break;
        case '4':
            alert('seu valor foi 4');
            break;
        case '5':
            alert('seu valor foi 5');
            break;
        default:
            alert('seu valor não foi nenhum valor entre 1
e 5');
    }
</script>
```

Funções

Assim como nas principais linguagens de programação, JavaScript também possui uma maneira de se criar funções. Uma função nada mais é do que um pedaço de programa destinado a uma tarefa bem específica e que pode ser utilizado várias vezes, em diferentes pontos do programa principal. Em JavaScript, uma função é definida usando-se a palavra function.

O trecho abaixo apresenta a declaração da função imprimeLista(), que escreve dez itens numa lista não ordenada

de HTML. Nesse caso, a função foi declarada dentro do <HEAD>, mas ela poderia ser declarada em qualquer local da página, desde que seja antes de ser usada

```
<html>
<head>
     <title>Script no head</title>
     <script>
          function imprimeLista(){
               document.write("<ul>");
               for(count=1; count <= 10; count++){
                    document.write( "<li>" + count +
"</li>");
               }
               document.write("</ul>");
          }
     </script>
</head>
<body>
     <script>
          imprimeLista();
     </script>
</body>
</html>
```

As funções em JavaScript também podem receber valores como parâmetros. Observe o exemplo abaixo. Nesse caso, a função imprimeLista() foi melhorada para receber a quantidade de itens da lista como parâmetro. Assim como nas variáveis, nos parâmetros de funções o tipo não é definido.

```
<html>
<head>
     <title>Script no head</title>
     <script>
          function imprimeLista(quantidade){
               document.write("<ul>");
               for(count=1; count <= quantidade; count++){
                    document.write( "<li>" + count + "</li>");
               }
               document.write("</ul>");
          }
```

```
    </script>
</head>
<body>
    <script>
        imprimeLista(10);
    </script>
</body>
</html>
```

Por fim, uma função também pode retornar um valor após sua execução. Nesse caso, basta usar o comando return para retornar o valor. O exemplo a seguir ilustra uma função que recebe um número como parâmetro e retorna o dobro de seu valor.

```
<script>
  function dobro(valor){
    return valor * 2;
  }
</script>
```

Tratamento de Eventos

Durante o carregamento e exibição de uma página HTML, diversos eventos podem acontecer. Esses eventos podem ser cliques e movimentos do mouse ou teclas digitadas pelo usuário, além de muitos outros eventos.

Esses eventos ocorrem em uma diversidade de situações diferentes e alguns deles são específicos de alguns tipos de componentes. Por exemplo, o evento de tecla digitada pode ser produzido quando o usuário digita algo sobre um componente de texto (como o textarea ou input text).

Já em elementos como imagens ou listas não ordenadas, eventos de digitação, como o exemplo anterior, não existem e nem teria muito sentido, já que esses tipos de elementos não interagem com o teclado, do ponto de vista de usabilidade. Porém mesmo para esses

tipos de elementos outros eventos podem ocorrer, como por exemplo quando o usuário passa o ponteiro do mouse sobre eles ou quando o carregamento de uma imagem é finalizado. Os eventos são produzidos normalmente em vários elementos da página Web ao longo do seu carregamento e utilização, assim como em diversos momentos onde o usuário navega pela página. Mas e o que o JavaScript tem a ver com os eventos produzidos por elementos HTML?

Finalizada essa visão geral sobre eventos em HTML, resta a questão de saber o que o JavaScript tem a ver com os eventos produzidos por elementos HTML.

Apesar de não parecer claro em um primeiro momento, essa é uma das partes mais importantes da utilização de JavaScript em páginas Web. A grande resposta para essa charada é que esses eventos podem ser tratados pelo programador ou, em outras palavras, o programador pode definir o que o programa deve fazer quando um determinado evento acontecer, e isso é especificado em JavaScript!

Por exemplo, o programador pode definir uma ação (um código JavaScript) que deve ser executada ao se passar o mouse por cima de uma imagem específica. O trecho de código que segue, por exemplo, define que a função abreAlerta() deve ser executada quando o usuário passar o mouse por cima da imagem (evento onMouseOver).

```
<html>
    <head>
       <title>Script no head</title>
       <script>
            function abreAlerta(){
                 alert('Passei por cima da imagem!');
            }
       </script>
    </head>
    <body>
       <img id='exemplo1' src='okgo.gif' onMouseOver='abreAlerta()'
/>
```

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

Cada tag HTML suporta diferentes tipos de eventos. A maioria dos eventos é precedida da palavra ON, como foi o caso do exemplo (Evento de MouseOver, ou seja, passar o mouse por cima, é tratado informando a propriedade onMouseOver da tag IMG).

Outro evento que pode ser tratado na tag IMG é o evento de MouseOut, através da propriedade onMouseOut. Esse evento é gerado quando o usuário sai com o mouse de cima da imagem. O exemplo a seguir ilustra o tratamento de múltiplos eventos de um mesmo elemento HTML.

```
<html>
    <head>
        <title>Script no head</title>
        <script>
            var over = 0;
            var out = 0;
            function contOver(){
                 over++;
            }
            function contOut(){
                 out++;
            }
            function imprimeAlerta(){
                 alert('Entradas:'+over+' Saidas:'+out);
            }
        </script>
    </head>
    <body onClick='imprimeAlerta()'>
        <img id='exemplo1' src='okgo.gif' onMouseOver='contOver()'
onMouseOut='contOut()' />
    </body>
</html>
```

Nesse caso, os eventos de entradas e saídas de mouse da área imagem são tratados pelas funções contOver() e contOut(), enquanto o clique em qualquer local da página executa a função imprimeAlerta().

Existe uma infinidade de eventos a serem tratados. Alguns eventos são comuns a vários tipos de elementos, enquanto outros só existem para determinados elementos. O Quadro 2 contém alguns eventos que são comuns a vários tipos de elementos. Para saber sobre outros eventos existentes por tipo de elemento HTML acesse:

Evento	**Descrição do Evento**
Click	Ativado pelo clique do usuário sobre o elemento.
MouseOver	Ativado quando o usuário coloca o mouse sobre o elemento.
MouseOut	Ativado quando o mouse sai de cima do elemento.
Focus	Ativado quando o elemento recebe o foco, como no caso do cursor em campos de formulários.
Blur	Ativado quando o elemento perde o foco.
Change	Ativado quando o elemento é modificado, como no caso de alteração do valor de campos de formulários.

Exemplos de eventos comuns

E, assim, chegamos ao fim da nossa primeira aula sobre JavaScript, mas você já pode e deve ir além.

Saiba mais:

https://www.w3schools.com/js/js_intro.asp

https://youtu.be/1-w1RfGIov4