UNINOVE

Bootstrap - JavaScript: Revisão dos Conceitos

REVISAR OS PRINCIPAIS CONCEITOS JAVASCRIPT.

AUTOR(A): PROF. JORGE HENRIQUE PESSOTA

Apresentação e Fundamentos

JavaScript é uma linguagem de programação *client-side* voltada para o desenvolvimento web. Ela foi criada em 1995 por Brendan Eich e é utilizada para definir comportamentos em páginas web.

A linguagem JavaScript é orientada a objetos e pode ser empregada de três formas em um projeto web: inline, interno e externo.

JavaScript Inline

No estilo *inline*, a codificação JavaScript é inserida dentro da seção body do documento HTML. O código deve ser envolvido por um bloco de tags script. Na tag de abertura do bloco deve ser adicionado o atributo type="text/javascript" para que o navegador interprete adequadamente o código.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">

  <head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Modelagem de Interfaces - Bootstrap</title>
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/jquery-3.2.1.min.js"></script>
    <script src="js/bootstrap.min.js"></script>

  </head>

  <body>

    <h1>JavaScript Inline</h1>

    <script type="text/javascript">
      alert("javascript Inline");
    </script>

  </body>
</html>
```

JavaScript Interno

No estilo Interno, a codificação JavaScript é inserida dentro da seção head do documento HTML. O código deve ser envolvido por um bloco de tags script. Na tag de abertura do bloco deve ser adicionado o atributo type="text/javascript" para que o navegador interprete adequadamente o código.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">

  <head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Modelagem de Interfaces - Bootstrap</title>
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/jquery-3.2.1.min.js"></script>
    <script src="js/bootstrap.min.js"></script>

    <script type="text/javascript">
      alert("javascript Interno");
    </script>

  </head>

  <body>

   <h1>JavaScript Interno</h1>

  </body>

</html>
```

JavaScript Externo

No estilo Externo, a codificação JavaScript é inserida em um arquivo com a extensão .js e um link para este arquivo é inserido no documento HTML.

Para realizar o link, deve-se adicionar o elemento script contendo o atributo type e o atributo src na seção head do documento HTML. No atributo type deve ser indicado o tipo de arquivo a ser carregado e no atributo src deve ser informado o local e o nome do arquivo JavaScript.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">

  <head>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Modelagem de Interfaces - Bootstrap</title>
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <script src="js/jquery-3.2.1.min.js"></script>
    <script src="js/bootstrap.min.js"></script>

    <script type="text/javascript" src="Bootstrap_JavaScript_Externo.js"><

  </head>

  <body>

   <h1>JavaScript Externo</h1>

  </body>

</html>
```

Código presente no arquivo Bootstrap_JavaScript_Externo.js.

```
alert("JavaScript Externo");
```

Também é possível acessar arquivos JavaScript disponíveis externamente na web. Para isso, é necessário inserir a url no atributo src.

Exemplo:

```
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.mi
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap
```

O uso de JavaScript externo é o mais vantajoso, pois permite separar as marcações HTML dos códigos JavaScript. Isso facilita a organização e a manutenção do projeto. Além disso, os arquivos externos são carregados em *cache*, agilizando o processo de carga da página.

Apesar do uso de JavaScript externo ser mais vantajoso, as exemplificações dos tópicos serão apresentadas a partir dos métodos inline e interno, a fim de manter toda a codificação em um único arquivo.

JavaScript - Variáveis

As variáveis são utilizadas na programação para armazenar dados na memória do computador. São elas que possibilitam a manipulação dos dados.

Para utilizar variáveis é necessário definir uma identificação para elas. Esta identificação não pode conter caracteres especiais e espaços em branco. Os únicos caracteres especiais permitidos são underscore "_" e cifrão "$".

Destaca-se que a linguagem JavaScript é *case sensitive* e caracteres maiúsculos e minúsculos são diferenciados.

Para declarar uma variável em JavaScript utiliza-se a palavra reservada "var" (opcional) e o seu identificador.

Toda instrução deve ser finalizada com ponto e virgula ";".

Exemplo:

```
1. var x;
2. var nome;
3. idade;
```

Para atribuir dados a uma variável, utiliza-se o operador de atribuição "=". Podem ser atribuídos valores numéricos e valores textuais, que neste caso, devem estar entre aspas "".

Ao alterar uma variável, também é possível inicializá-la. Seguem exemplos:

```
1. var x = 10;
2. var nome = "Bootstrap";
3. idade = 24;
4. var web = ["HTML", "CSS", "JavaScript", "Bootstrap"];
```

JavaScript - Operadores

Para toda linguagem de programação é fundamental o uso de operadores, que podem ser aritméticos, lógicos e relacionais.

Os operadores aritméticos aceitos em JavaScript são:

+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão (mod)
++	Incremento
--	Decremento

Os operadores lógicos aceitos em JavaScript são:

&&	operador E
||	operador OR
!	operador de negação

Os operadores relacionais aceitos em JavaScript são:

==	igualdade
===	igualde de valor e de tipo de dados
!=	diferença
!==	diferença de valor e de tipo de dados
>	maior que
<	menor que
>=	maior ou igual que
<=	menor ou igual que
?	operador ternário

JavaScript - Controle de fluxos

Controle de fluxo é um recurso da lógica de programação que permite definir qual fluxo de execução o programa deve seguir, de acordo com condições pré-definidas.

Um controle de fluxo pode ser definido a partir das instruções if, if...else e switch.

Exemplos:

```
// controle de fluxo - if

if (condição desejada) {
    instrução;
    instrução;
    instrução;
}


 // controle de fluxo - if...else

if (condição desejada) {
    instrução;
    instrução;
    instrução;
}
else{
    instrução;
    instrução;
    instrução;
}


// controle de fluxo - if...if else...else

if (condição desejada) {
    instrução;
    instrução;
    instrução;
}
else if (condição desejada) {
    instrução;
    instrução;
    instrução;
}
else{
    instrução;
    instrução;
    instrução;
}
```

```
41.
42.
43. // controle de fluxo - switch
44.
45. switch (valor) {
46.   case condição1:
47.     instrução;
48.     break;
49.     case condição2:
50.     instrução;
51.     break;
52.   default:
53.     instrução;
54.     break;
55. }
```

JavaScript - Estruturas de Repetição

Estruturas de repetição são estruturas que permitem a execução de um determinado conjunto de instruções repetidas vezes, até que se atinja um ponto de parada.

A linguagem JavaScript suporta duas estruturas de repetição: for e while.

Estrutura de repetição for

A estrutura de repetição for é utilizada quando se deseja executar uma instrução por uma quantidade de vezes pré-determinada.

Exemplo:

```
1. for (inicialização; condição de parada; incremento/decremento) {
2.     instrução;
3.     instrução;
4.     instrução;
5. }
```

Estrutura de repetição while

A estrutura de repetição while é utilizada quando se deseja executar uma instrução repetidas vezes, enquanto uma condição for atendida. É possível emprega-la de duas formas. while e do...while.

A instrução while permite verificar se a condição desejada é atendida, antes mesmo de executar a primeira vez o conjunto de instruções (pré-teste).

A instrução do...while verifica se a condição desejada é atendida somente após a primeira execução das instruções (pós-teste).

Exemplo:

```
1. // Estrutura de repetição while
2.
3. while (condição desejada) {
4.     instrução;
5.     instrução;
6.     instrução;
7. }
8.
9.
10. // Estrutura de repetição do...while
11.
12. do {
13.     instrução;
14.     instrução;
15.     instrução;
16. } while (condição desejada);
```

JavaScript - Funções

Funções são blocos de instruções que realizam uma tarefa específica e que só são executadas quando invocadas.

Para criar uma função é necessário definir um identificador único, indicar os parâmetros de entrada (se houver) entre parênteses e incluir as instruções dentro de um bloco de chaves {}.

Exemplo:

```
1. minhaFuncao(parâmetros){
2.     instrução;
3.     instrução;
4.     instrução;
5. }
```

Uma função pode receber valores quando invocada, como também, pode retornar valores para quem à invoca.

Exemplo:

```
// função que não possui parâmetros de entrada e não retorna valor

minhaFuncao(){
    instrução;
    instrução;
    instrução;
}

// invocando a função
minhaFuncao();


// função que possui parâmetros de entrada mas não retorna valor

minhaFuncao(parametro1, parametro2){
    instrução;
    instrução;
    instrução;
}

// invocando a função
minhaFuncao(parametro1, parametro2);


// função que possui parâmetros de entrada e que retorna valor

minhaFuncao(parametro1, parametro2){
    instrução;
    instrução;
    instrução;
    return valor;
}

// invocando a função
var x = minhaFuncao(parametro1, parametro2);
```

JavaScript - Objetos

A linguagem JavaScript é orientada a objetos. Um objeto consiste na abstração de algum objeto do mundo real e é composto por propriedades e por métodos.

As propriedades de um objeto podem ser acessadas da seguinte maneira:

nome-do-objeto.nome-da-propriedade;
ou
nome-do-objeto["nome-da-propriedade"];

Os métodos de um objeto podem ser acessados da seguinte maneira:

nome-do-objeto.nome-do-metodo();

JavaScript - Apresentação de dados

É possível apresentar dados para o usuário de diversas maneiras com JavaScript.

Seguem detalhes:

document.getElementById(id).innerHTML

Este método possibilita apresentar um conteúdo em um elemento HTML. Neste caso, deve-se atribuir um id no elemento HTML alvo. Esse elemento é recuperado a partir do método document.getElementById(id) e o valor a ser apresentado é adicionado na propriedade "innerHTML".

Exemplo:

```
1. <h1 id="meuID"></h1>
2.
3. <script>
4.   document.getElementById("meuID").innerHTML = "JavaScript - innerHTML";
5. </script>
```

document.write()

Este método apresenta o conteúdo informado como parâmetro.

Exemplo:

```
1. <script>
2.    document.write("JavaScript - document.write()");
3. </script>
```

window.alert()

Este método apresenta o conteúdo informado como parâmetro em uma janela de alerta.

Exemplo:

```
1. <script>
2.    window.alert("JavaScript - window.alert()");
3. </script>
```

console.log()

Este método é útil para *debugar* o código. Ele apresenta o conteúdo informado como parâmetro do método na console, na opção de "Ferramentas do Desenvolvedor" do navegador.

Exemplo:

```
1. <script>
2.    console.log("JavaScript - console.log");
3. </script>
```

Resumo do Tópico

Neste tópico foram revisados os principais conceitos JavaScript.

Os seguintes tópicos foram abordados:

Apresentação e Fundamentos

JavaScript inline

JavaScript interno

JavaScript externo

Variáveis

Operadores

Controle de Fluxos

Estruturas de Repetição

Funções

Objetos

Apresentação de Dados

ATIVIDADE FINAL

Quais são as três formas de utilizar JavaScript em um projeto web?

A. inline, outline e externo.

B. inline, interno e externo.

C. outline, incorporado e externo

D. inline, incorporado e linkado.

JavaScript é *case sensitive*. Isso significa que:

A. Caracteres minúsculos e maiúsculos não são diferenciados pela linguagem.

B. A linguagem não permite o uso de caracteres especiais.

C. A linguagem não permite o uso de valores numéricos.

D. Caracteres minúsculos e maiúsculos são diferenciados pela linguagem.

Qual método JavaScript é indicado para realizar *debug* do código?

A. console.log()

B. window.alert()

C. document.write()

D. document.getElementById(id).innerHTML

REFERÊNCIA

SILVA, Maurício Samy. Bootstrap 3.3.5: Aprenda a usar o framework Bootstrap para criar layouts CSS complexos e responsivos. 1.ª ed. São Paulo: Novatec, 2015c.

TUTORIALS POINT. Bootstrap Overview, Bootstrap Tutorial. Disponível em: http://www.tutorialspoint.com/bootstrap/ (http://www.tutorialspoint.com/bootstrap/bootstrap_tutorial.pdf). Acesso em: 17 de dezembro de 2017.

W3SCHOOLS.COM. JavaScript Tutorial. Disponível em: https://www.w3schools.com/js/default.asp (https://www.w3schools.com/js/default.asp). Acesso em 17 de dezembro de 2017.