Introdução ao JavaScript

Tiago Lopes Telecken

telecken@gmail.com

Introdução - JavaScript

- **Java Script é uma linguagem para criação de scripts em clientes Web;**
  - **Criação de interações entre usuários e páginas HTML**
  - **Acesso e manipulação de recursos de navegadores Web e páginas HTML**
  - **Criação de animações e recursos visuais**
  - **Validação e teste de formulários**
  - **Realização de processamentos e cálculos diversos em navegadores Web**

Características

- Linguagem compacta baseada em objetos, específica para aplicações web.
- Escrita diretamente no HTML através de qualquer editor de textos simples, como por exemplo, Bloco de Notas.
- Reconhece situações (eventos) como cliques, movimentos do mouse, entrada de dados, etc.
- Sintaxe parecida com a linguagem Java, porém é mais simples e flexível. Por outro lado tem menos recursos

Características

- **Linguagem Segura-Limitada**
  - **Não possui autorização para gravar dados no disco rígido.**
- **Linguagem interpretada**
  - **Não há a necessidade de compilar o código**
  - **É só escrever o código e executar**
  - **Agiliza o desenvolvimento**
  - **Desempenho mais lento**
- **Possue objetos e funções**

Características

- **Procuraremos seguir o “modern mode” apresentado a partir do ES2015.**
- **Também utilizaremos recursos dos ECMAScript 2019 em diante**

Comandos

- **É case-sensitive (diferencia maiúsculas de minúsculas).**
- **Comandos terminam com “;”**
- **Comentários //   /* */**

Primeiros programas

- **Inserindo um código JavaScript em uma página html. Existem 3 maneiras**
  - **1 Utilizar a tag <script>**

```
<html>
   <head></head>
   <body>
    <script language="JavaScript">
        document.write("Ola mundo!");
    </script>
   </body>
</html>
```

Primeiros programas.

```
<html>    <head></head>
   <body>
     Um <br>
     <script language="JavaScript">
              document.write("Ola mundo!");
     </script>
<br>dois
     <script language="JavaScript">
              document.write("<br><B>Ola
   mundo!</B>");
     </script>
   </body>
</html>
```

Comando document.write

- document.write(“”)
- Insere um código na página HTML, pode ser uma string simples ou pode conter tags

```
document.write(“Ola mundo!”);
document.write(“<I><B>Ola mundo!</B></I>”);
```

Primeiros programas

- **Usando arquivos externos**
  - **2 Utilizar o atributo “src” da tag <script>**

```
<html>
   <head></head>
   <body>
    <script language="JavaScript"
   src=“../file/ola.js”>
    </script>
   </body>
</html>
```

  - **3 Utilizar atributos de alguns elementos do HTML**
    - **Será visto quando trabalharmos com eventos**

Objetos JavaScript

- Objetos, propriedades, métodos
  - carro
  - carro.ano
  - carro.dono.nome
  - document.write(“ola”) // window.document.write(“ola”)
  - navigator.appName; //window.navigator.appName;
- Objetos pré-existentes, criados pelo usuário

Propriedades de Objetos JavaScript.

```
<html>
    <head>
        <title>Untitled</title>
    </head>
    <body>
        <Script Language="JavaScript">
          document.write ("<br>Navegador: " + navigator.appName);
         document.write ("<br> Versão do Navegador: "+ navigator.appVersion);
          document.write ("<br> Sistema operacional : "+ navigator.platform);
           document.write ("<br> location.href : "+ location.href);
          document.write ("<br>Tela Largura: "+ screen.width);
          document.write ("<br> Tela altura : "+ screen.height);
   // document.write (`<br> Navegador: ${screen.height}`);
  //notação alternativa – template string – delimitado pela crase
         </Script>
    </body>
</html>
```

Métodos de Objetos JavaScript.

- **Além de uma página**

```
<html>
<head>
    <script>
        alert("Mensagem para o  usuário");
        confirm("Aceita as condições?");
        prompt("Qual o seu nome?","fulano");
        window.open("http://www.google.com","janela2");
    </script>
</head>
<body>   </body>
</html>
```

Métodos de Objetos JavaScript

```
<html>
<head>
    </head>
<body>
    <script>
        let nome= prompt(“Qual o seu nome?”,”fulano”);
        document.write(“Seja bem vindo ”+nome);
    </script>
</body>
</html>
```

Funções matemáticas

- Math.abs(número) - retorna o valor absoluto do número (ponto flutuante)
- Math.ceil(número) - retorna o próximo valor inteiro maior que o número
- Math.floor(número) - retorna o próximo valor inteiro menor que o número
- Math.round(número) - retorna o valor inteiro, arredondado, do número
- Math.pow(base, expoente) - retorna o cálculo do exponencial
- Math.max(número1, número2) - retorna o maior número dos dois fornecidos
- Math.min(número1, número2) - retorna o menor número dos dois fornecidos
- Math.sqrt(número) - retorna a raiz quadrada do número

- alert (Math.sqrt(16))

Manipulando Strings

- As variáveis de tipo texto são objetos da classe String. Possuem propriedades e métodos.
- Propriedade Length: armazena o número de caracteres do String.
- Métodos
- charAt(índice): Devolve o caractere que há na posição indicada como índice. As posições de um string começam em 0.
- substring(início,fim): Devolve o substring que começa no caractere de início e termina no caractere de fim.
- toLowerCase(): Coloca todos os caracteres de um string em minúsculas.
- toUpperCase(): Coloca todos os caracteres de um string em maiúsculas.
- toString(): converte numeros e outros tipos de dados em strings.

Manipulando Strings

- <html>
- <body>

- <script type="text/javascript">
- let txt="Hello World!";
- document.write(txt.length);
- </script>

- </body>
- </html>

- txt.charAt(3) // l
- txt.substring(2,5) // llo

Tipos de dados

- String
  - let A=“ola”;
  - let B=‘ ola2 \n \t ’;
- Booleano
  - true //ou qualquer string diferente de 0 ou string vazia
  - false //ou 0 ou string vazia
- Número
  - Inteiro, ponto flutuante, decimal, octal, hexadecimal
  - 1; 1.2, 1.54e5, 0045, 0x66f
  - NaN, infinito
- undefined- propriedade que nao existe,variavel sem valor atribuido
- null – variável sem valor

Variáveis

- As variáveis são declaradas utilizando-se o operador let
  - Pode-se declarar e atribuir valores à uma variável em uma mesma linha.
  - Os tipos das variáveis são deduzidos pelo interpretador JavaScript
- Exemplos

```
let Valor = 30;          int
let Nome="Fulano";       string
let Peso= 20.5           float
let altura;
```

Variáveis

- Os tipos das variáveis podem ser alterados durante a execução de um código JavaScript

```
let Carga = 30;                    // tipo int , valor 30
let Peso= 20.5;                    // tipo float, valor 20.5
Carga= Carga + Peso;               // tipo float, valor 50.5
Carga=Carga + ”oi”;                // tipo string, valor “50.5oi”
```

- Existem funções que convertem tipos
  - X= parseInt(“10”); //10
  - Y= parseFloat(“44.6”); //44.6
  - Z=toString(10); //”10”

Conversão de tipos

- valor=String(valor); //true → ”true”
- valor=Number(valor); // ’12’ → 12
- valor=Boolean(valor); // ’12’ → true , 0 → false

Nomes de variáveis

- Não pode iniciar com número (2lugar --X)
- Só pode ter Alfanuméricos, _ ou $ (u-name --X)
- Boas Práticas
  - Nomes auto-explicativos (a, u, s --X) (userName)
  - camelCase (userName)
  - Para algumas constantes UPPER_CASE (MAX_TIME)
  - Funções também usam camelCase e geralmente tem um verbo no nome (isInteger, getName)
  - Classes são escritas em PascalCase (MainUser)

Operadores Aritméticos

+ adição de valor e concatenação de Strings;
- subtração de valores;
* multiplicação de valores;
/ divisão de valores;
=, += atribuição

- Utilizados em cálculos e manuseio de variáveis.
  - Peso= Peso+20*4
  - Nome=“fulano”+”de Tal”

Operadores lógicos

- São operadores a serem utilizados em estruturas condicionais e de repetição

===	Igual (valor e tipo, 5 === “5” false)
==	Igual (valor, 5 == “5” true)
!=	Diferente
>	Maior
>=	Maior ou Igual
<	Menor
<=	Menor ou Igual
&&	E
||	Ou

- Peso == Carga //retorna verdadeiro ou falso

Estruturas Condicionais

- São comandos que condicionam a execução de certa tarefa à veracidade ou não de uma determinada condição
- If, Else

```
if (condição) {
     ação para condição satisfeita
}
else {
    ação para condição não satisfeita
 }
```

```
if (Idade < 18) {
    let Categoria = "Menor“;
}
else {
    let Categoria = "Maior“;
}
```

Estruturas de repetição

- **Estruturas que permitem que uma parte do código seja executada várias vezes**
- **While**
  - **Executa uma ação enquanto determinada condição for verdadeira.**

```
while (condição)
 {
   ação
}
```

```
let x=0;
while (x<3)
 {
    alert ("X igual a " +
x);
    x=x+1;
}
```

Estruturas de repetição

- For

```
for ( [inicialização de variável de controle ;]
    [condição ;] [incremento da variável de
    controle] ) {
        comandos
}
```

```
for (let x = 0 ; x < 3 ; x++)
{
    alert ("X igual a " + x);
}
```

Estruturas de repetição

- **Estruturas que permitem que uma parte do código seja executada várias vezes**
- **While**
  - **Executa uma ação enquanto determinada condição for verdadeira.**

```
while (condição)
 {
   ação
}
```

```
let x=0;
while (x<3)
 {
    alert ("X igual a " +
x);
    x=x+1;
}
```

Declaração de Funções (Function Declaration)

- Funções modularizam os programas
- Programa modularizado em funções torna-se mais fácil de manter;

```
function nomeDaFunção ( parametro1,... parametroN ){
        instruções;
        return valor;
}

function soma (a, b){
       return a+b;
}
```

Funções

```
<html>
<head>
  <script>
     function soma (a, b) {
        return a+b;
     }
  </script>
</head>
<body>
  <script>
     alert (soma(1, 2) );
  </script>
</body>
</html>
```

Funções – Function Expression

- Function Declaration (vista anteriormente)
- Function Expression (anônima)

```
let soma = function (a, b) {
return a+b;
    }
…
alert(soma(3,6));     //9

...

let sum = soma;       // cópia
alert(sum(3,6));      //9
```

Function Expression

```
function consulta(questao, sim, nao) {
    if (confirm(questao)) sim()
    else nao();
}

consulta(
    "Vai comprar?" ,
    function() { alert("Você comprou."); } ,
    function() { alert("Você cancelou a compra."); }
);
```

- Uma função pode receber outras funções como parâmetro
- Os parametros “sim” e “nao” são funções passadas para consulta. Estas funções poderão ser executados conforme a lógica da consulta

Funções – Arrow functions

- **sintaxe mais compacta que as expression functions**
- **let soma = (a,b) => { return a+b}**
- **…**
- **alert(soma(2,3)); *//5***

==========

```
consulta(
  "Vai comprar?",
  () => alert ("Você comprou” ),
  () => alert ("Você cancelou a compra.")
);
```

/ forma compacta que pode ser usada quando a func. Tem só uma linha de comando*/*

Escopo - let

- **Variáveis declaradas em um bloco tem visibilidade no bloco**
- **Variáveis declaradas dentro de uma função tem visibilidade na função**
- **Variáveis declaradas fora de uma função tem visibilidade global (boa prática: declarar variáveis globais no inicio do arquivo)**
- **Permite redefinição mas não redeclaração**
  - let a=1;
  - a =2; // ok redefinição

Const, Var

- const: uso e escopo semelhante ao let. Porém a variável declarada com const não pode ter o seu valor alterado, é uma constante
- var: não tem escopo de bloco, permite redeclaração (dar preferência ao let). Nas demais situações similar ao let
- Variáveis sem prefixo tem visibilidade global (não importa aonde sejam declaradas -- evitar)
- As variáveis/funções não são visíveis de uma página para outra

Escopo funções

- **Funções declaradas fora de funções tem visibilidade global**
- **funções podem ser atribuídas a variáveis e neste caso sua visibilidade é a visibilidade da variável**
- **Funções podem ser criadas dentro de funções. As funções internas podem ver os valores das variáveis da função pai**

```
function imprimeNome() {
let nome=”fulano”;
return function () {return nome}; }
```

- **Funções podem ser passadas por parâmetro e retornadas**

Escopo funções

- **Funções declaradas tem hoisting**
  - console.log(soma(10,2)); //retorna 12
  - function soma(a,b) {return a+b;}
- **O código acima funciona por que a engine JS deixa todas declarações de funções visíveis ao seu escopo antes de serem executadas**
- **Utilizando-se o prefixo var em variáveis também ocorre o hoisting. Porém é uma má prática.**
- **O valor de uma variável declarada sem atribuição é undefined**
  - console.log(a); //undefined
  - var a=2;