Estudo sobre a Linguagem de Programação Java

Débora Bianca Taveira de Moura e Ewelly Fabiane Cunha de Sousa

Departamento de Ciência da Computação – Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brasil

lovellytm@gmail.com, Ewellyfab@gmail.com

JavaScript é uma linguagem de script orientada a objetos, multiplataforma. É uma linguagem pequena e leve. Dentro de um ambiente de host (por exemplo, um navegador web). O nosso aplicativo visa acelerar o atendimento da ambulância. O tempo recomendado de atendimento é de 8 a 10 min. (desde o telefonema até a chegada da ambulância ao local da ocorrência), porém, nem sempre esse processo ocorre dentro do tempo limite. Para auxiliar esse processo, na nossa aplicação o usuário poderá rapidamente chamar a ambulância. tanto pelo celular, quanto computador, tablet, entre outros. Além da chamada da ambulância, também será enviado à localização do usuário para a ambulância.

1. História do surgimento da linguagem

JavaScript foi desenvolvido por Brendan Eich enquanto trabalhava na Netscape (um dos precursores dos navegadores web). Essa década foi um período de revolução, pois os browsers ainda eram estáticos. Originalmente chamava-se LiveScript, mas logo seu nome foi mudado para JavaScript. Essa mudança de nome coincidiu com a época em que a Netscape adicionou suporte à tecnologia Java em seu navegador (Applets).

A escolha final do nome causou confusão dando a impressão de que a linguagem foi baseada em Java, sendo que tal escolha foi caracterizada por muitos como uma estratégia de marketing da Netscape para aproveitar a popularidade do recém-lançado Java. Assim, JavaScript rapidamente adquiriu ampla aceitação como linguagem de script client-side de páginas web.

JavaScript tem se transformado na linguagem de programação mais popular da web. O javaScript foi criado com uma implementação ao HTML, já que este não era suficiente para cumprir todos os objetivos que se esperava de uma página web. (Portal Educação, 2015) Inicialmente, no entanto, muitos profissionais denegriram a linguagem, pois ela tinha como alvo principal o público leigo. Com o advento do Ajax, JavaScript teve sua popularidade de volta e recebeu mais atenção profissional. O resultado foi a proliferação de frameworks e bibliotecas, práticas de programação melhoradas e o aumento no uso do JavaScript fora do ambiente de navegadores, bem como o uso de plataformas de JavaScript server-side.

O JavaScript tem um papel fundamental na evolução da web, pois a ideia central da Internet era ser uma plataforma aberta para todos. Foi super importante que a sua

principal linguagem de programação também fosse aberta, sem controle centralizado de corporações e de fácil aprendizado. (Leonardo Balter)

JavaScript tem uma biblioteca padrão de objetos, como: Array, Date, e Math, e um conjunto de elementos que formam o núcleo da linguagem, tais como: operadores, estruturas de controle e declarações. O núcleo do JavaScript pode ser estendido para uma variedade de propósitos, complementando assim a linguagem:

O **lado cliente do JavaScript** estende-se do núcleo linguagem, fornecendo objetos para controlar um navegador web e seu Document Object Model (DOM). Por exemplo, as extensões do lado do cliente permitem que uma aplicação coloque elementos em um formulário HTML e responda a eventos do usuário, como cliques do mouse, entrada de formulário e de navegação da página.

O **lado do servidor do JavaScript** estende-se do núcleo da linguagem, fornecendo objetos relevantes à execução do JavaScript em um servidor. Por exemplo, as extensões do lado do servidor permitem que uma aplicação comunique-se com um banco de dados, garantindo a continuidade de informações de uma chamada para a outra da aplicação, ou executar manipulações de arquivos em um servidor.

2. Domínios de aplicação

JavaScript é mais usada para criar scripts em páginas web. Exemplos de páginas que utilizam a linguagem: Google (Front End), Youtube (Front End), Facebook (Front End), Yahoo (Front End e Back End), Amazon (Front End), EBAY (Front End e Back End), entre outros.

3. Paradigmas suportados pela linguagem

É uma linguagem de script multi-paradigma, baseada em protótipo que é dinâmica, e suporta estilos de programação orientado a objetos, imperativo e funcional.

Exemplo Funcional: Função para executar somente se tiver algum campo interno.

```
function executaSeTemCampo(target, name) {
   return facaQuando(existy(target[name]), function() {
     var result = _.result(target, name);
     console.log(['O resultado é', result].join(' '));
     return result;
```

```
    });
   }
```

Exemplo Imperativa:

```
var visits = getCookie(“counter”);
 if (visits) {
     visits = paserInt(visits)+1;
     document.write ("By the way, you have been here" +
visits + "times");
 }
 else {
     visits=1;
       document.write("By the way, this is your first time
here.");
 }
setCookie("counter", visits);
}
```

Exemplo Orientado a Objetos:

```
<html>
 <head>
 <script language="JavaScript">
  function Carro(){
   var Marca = "Sem marca";
   var Modelo = "Sem modelo";
   this.SetMarca = SetMarca;
   this.SetModelo = SetModelo;
   this.ShowMarca = DisplayMarca;
   this.ShowModelo = DisplayModelo;

   function DisplayMarca(){
    alert(Marca);
   }

   function DisplayModelo(){
    alert(Modelo);
   }
```

```
    function SetMarca(_marca) {
     Marca = _marca;
    }

    function SetModelo(_modelo) {
       Modelo = _modelo;
    }

   }
   var carro = new Carro();
   carro.SetMarca("Ford");
   carro.SetModelo("Ka");
   carro.ShowMarca();
   carro.ShowModelo();
   </script>
   </head>
  </body>
 </html>
```

4. Variáveis e tipos de dados

Os tipos de dados de uma variável Javascript podem ser:

String: quase qualquer valor entre aspas simples ou aspas duplas.

Os tipo de dados "String" é uma cadeia de zero ou mais caracteres Unicode (letras, números e sinais de pontuação). Normalmente utilizamos este tipo de dado para representar textos em geral. Os caracteres de uma string devem ser envolvidos em aspas simples ou aspas duplas.

```
var emailCliente = 'contato@todoespacoonline.com/w';
```

Numérico: números;

Variáveis com tipos de dados numéricos só podem conter um tipo de valor, números (óbvio). Tais números podem, ou não, ter o ponto flutuante, por exemplo:

```
/* Números inteiros */
var numeroNegativo = -1420;
var zero = 0;
var numeroComum = 35842;
```

```
/* Números válidos de ponto flutuante */
var numeroFloat = 0.3555
var outroNumero = 144.006;
var negativoComDecimal = -2.3;
var outroNumero = 19.5e-2 // .195
var decimalZero = 12.0;
```

Booleano: true (verdadeiro) ou false (falso);

Variáveis com tipo booleano têm apenas duas possibilidades de valores, true (verdadeiro) ou false (falso). Estes valores não devem ser cercados por aspas, ou seja, "true" não é a mesma coisa que true.

```
var valorBooleano = true; // Verdadeiro
```

Normalmente utilizamos esses tipo de valores com condicionais para controlar o fluxo da nossa aplicação Javascript, por exemplo:

```
var valorBooleano = true; // Verdadeiro
// Verifica se valorBooleano é verdadeiro ou falso
if ( valorBooleano ) {
alert( 'Variável verdadeira' );
} else {
alert( 'Variável falsa' );
}
```

Null e undefined: representam variáveis que não têm um valor ou estão incompletas. Para criar uma variável nula, você deve especificar seu valor como "null", como no exemplo abaixo:

```
var semValor = null; // null
```

No entanto, em Javascript você pode declarar uma variável mas não a inicializar, exemplo:

```
var indefinida; // undefined
```

Outro método que não está entre nulo (null) nem indefinida (undefined), é um método que alguns desenvolvedores costumam utilizar para inicializar variáveis:

```
var semValor = ""; // Nenhum valor
```

Ao contrário do que muitos imaginam, a variável acima não tem valor, mas não está indefinida (undefined) e nem tem valor nulo (null).

Uma maneira boa para checar uma variável quando você não tem certeza se ela está definida e tem valor, é utilizando uma expressão condicional (if) com o nome da variável, exemplo:

```
// Checa se a variável está definida (undefined)
// Se ela tem valor
// E se ela não é nula (null)
if ( variavel ) {
    // Continua o código
}
```

Arrays: Referência a vários espaços na memória. Com o Array é possível armazenar um conjunto de quaisquer valores javascript, como números, caracteres ou textos ou uma mistura deles. Imagine o array como um gaveteiro onde você pode adicionar ou retirar gavetas e cada gaveta contém o objeto que quiser.

Exemplo: vamos criar aqui um gaveteiro onde a primeira gaveta contém o valor 10 a segunda 20 e a terceira 30.

```
var  gaveteiro = [10,20,30];
```

Objetos são como uma espécie de "super variáveis" que armazenam uma "coleção de valores" referenciados por nome, e que podem ser recuperados para serem utilizados em diversas outras partes de um programa. Em JavaScript praticamente qualquer tipo de dado é um objeto.

Cada item dessa "coleção de valores", é chamado de propriedade. Cada propriedade é composta por um par de "nome: valor". Quando uma propriedade armazena uma função, ela se torna o que chamamos de método.

Exemplo: Imagine que você vai criar um programa para organizar álbuns de vários cantores e bandas. Aqui vamos criar um objeto para armazenar informações sobre um álbum da banda Metallica, depois você pode praticar criando objetos com suas bandas favoritas.

```
var album = {
 title: "Metallica (Black Album)",
 released: 1991,
 showInfo: function() {
   alert("Título do álbum: " + this.title + "Lançado em: "
+ this.released);
 }
};
```

Regras de nomeação:

- Nomes de variáveis podem começar com uma letra, um underscore (_) ou um cifrão ($).
- Os caracteres seguintes podem ser letras, números, underscores e o cifrão.
- Também é possível usar letras acentuadas, caracteres escapados e outros caracteres Unicode, a partir da versão JavaScript 1.5.
- O nome de uma variável sempre deve passar uma boa ideia do que ela representa.
- Se um nome for composto (mais de uma palavra), recomendo usar a notação CamelCase (por ex.:, nomeCliente)
- Os nomes de variáveis são case-sensitive

5. Comandos de controle

Em JavaScript são usados quatro tipos de estruturas de controle:

For: Repete uma seção do código um determinado número de vezes. Consiste de uma declaração que define as condições da estrutura e marca seu início. Esta declaração é seguida por uma ou mais declarações executáveis, que representam o corpo da estrutura.

Estabelece um contador inicializando uma variável com um valor numérico. O contador é manipulado através da <ação> especificada no comando toda a vez que o loop alcança seu fim, permanecendo nesse loop até que a <condição> seja satisfeita ou a instrução Break seja executada.

Forma geral:

```
For (<inicialização>; <condição>; <ação>){
    Corpo da Estrutura
}
```

Exemplo: o bloco de instruções será executado 10 vezes, pois a variável Contador é inicializada com o valor 1 e o bloco de instruções será executado enquanto Contador for menor que 11. A cada execução do bloco de instruções Contador é incrementado.

```
For (var Contador = 1; Contador < 11; Contador++){
    document.write(Contador);
}
```

For...In: Este comando tem por objetivo, procurar a ocorrência de uma variável, dentro das propriedades de um objeto, ao encontrar a referida variável, um bloco de comandos pode ser executado.

Forma geral:

```
For (variavel In objeto){
    bloco de comandos
}
```

Exemplo: Esta função procura por uma propriedade do Objeto, cujo o nome esteja especificado pela variável Procura, onde Nome é uma string correspondendo ao nome do objeto.

```
Function SearchIn(Procura,Objeto,Nome){
    Var ResultadoDaBusca = ""
    For (Procura In Objeto){
          document.write (Nome + "." + Procura + " = " +
Objeto[Procura] + "<BR>");
    }
}
```

If...else... : A estrutura If executa uma porção de código se a condição especificada for verdadeira. A estrutura pode também ser especificada com código alternativo para ser executado se a condição for falsa (else).

Forma geral:

```
if (<condição>){
    bloco de comandos
}else{
```

```
        bloco de comandos
}
```

Exemplo: A função retornará 'Já pode votar!' se a condição após o if for verdadeira, caso seja falsa, a função retornará 'Ainda é muito cedo para votar!'.

```
Function VerificaIdade(anos){
    If anos >= 16{
        Return ('Já pode votar!')
    }else{
        Return ('Ainda é muito cedo para votar!')
    }
}
```

Uma alternativa para economizar If's seria a utilização de uma expressão condicional, que funciona para situações mais simples, o exemplo acima ficaria da seguinte forma:

```
VariavelDeRetorno=(anos>=16) ? 'Já pode votar!': 'Ainda é muito cedo para votar!'
```

While: Outro tipo de loop é aquele baseado numa condição ao invés de no número de repetições. Por exemplo, suponha que você necessita que um determinado processo seja repetido até que um determinado teste dê um resultado verdadeiro, ou seja, executada a instrução Break.

Forma geral:

```
while (<condição>){
    Corpo da Estrutura
}
```

Exemplo: o bloco de instruções será executado 10 vezes, pois a variável Contador é inicializada com o valor 1 e o bloco de instruções será executado enquanto Contador for menor que 11. A cada execução do bloco de instruções Contador é incrementado.

```
Var Contador=1;
While ( Contador < 11 ){
    document.write(Contador++);
}
```

6. Escopo (regras de visibilidade)

Em Javascript utilizamos a palavra var para dar escopo a uma variável, tal escopo pode ser local ou global.
Uma variável local pode ser acessada apenas pela função na qual ela foi criada, uma variável global pode ser acessada em qualquer parte do seu arquivo Javascript, incluindo outros arquivos e bibliotecas que você possa vir a utilizar no futuro.

```
/* Variável local */
var minhaMensagem = "Oi";
```

```
/* Variável Global*/
minhaMensagem = "Oi";
```

Não é incorreto utilizar variáveis globais (sem a palavra var), contudo, como elas podem ser acessadas em quase todas as partes do seu arquivo, não é recomendável que você as utilize. Isso pode gerar conflitos com outras bibliotecas e/ou seus próprios arquivos Javascript que você adicionar posteriormente na sua página.

Isso acontece porque você pode precisar utilizar o nome daquela variável para salvar algo que seja necessário para seu código, porém, se alguma outra biblioteca, ou outra função que você criar, modificar o valor daquela variável acidentalmente, seu código terá problemas. Neste caso, pode acontecer de você perder horas tentando encontrar um problema que você mesmo gerou.

Para resumir, sempre utilize a palavra "var" antes do nome de suas variáveis, a não ser que saiba exatamente o que está fazendo.

```
var numeroUm = 1; // O número um :)
```

7. Exemplo prático de uso da linguagem de programação

O nosso aplicativo tem o objetivo de salvar vidas, reduzindo o tempo de atendimento da ambulância utilizando plataformas eletrônicas como tablets, smartphones e notebooks como meios de comunicação para solicitar o serviço. O tempo recomendado de atendimento é de 8 a 10 min. (desde o telefonema até a chegada da ambulância ao local da ocorrência), porém, de acordo com a Secretaria de Saúde, o tempo médio atual se encontra com uma taxa de atraso superior a 30%.

Para auxiliar esse processo, na nossa aplicação o usuário pode rapidamente chamar a ambulância após um cadastro prévio.

O usuário deverá inserir seus dados uma única vez para cadastro de suas informações pessoais. Quando for necessário acionar o sistema de atendimento pré-hospitalar. Ao solicitar ajuda, as informações de localização do solicitante são salvas e enviadas ao Socorrista. Na tela do socorrista a aplicação lê a localização por meio das coordenadas de latitude e longitude passadas pelo solicitante, e por meio disto exibe o posicionamento do solicitante no mapa. Isso minimiza o tempo da chamada e da chegada de uma ambulância.

Para a implementação da aplicação nós usamos o paradigma orientado a objetos, já que este oferece uma divisão de código um pouco mais lógica e melhor encapsulada do que a empregada nos sistemas não orientados a objetos. Isto torna a manutenção e extensão do código mais fácil e com menos riscos de inserção de bugs.

8. Conclusões

A elaboração do objeto deste trabalho (e-pronto) permitiu adquirir um grande conhecimento técnico e científico a respeito da linguagem de programação JavaScript, uma vez que foi necessário um aprofundamento teórico, sobre o assunto, pelos integrantes do grupo. Além disso, a produção do aplicativo proporcionou, também, uma experiência prática com a linguagem de programação estudada, contribuindo com uma melhor formação acadêmica dos integrantes.

9. Referências

Developer Mozilla: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript> <Acessado em: 09/07/2017>

Tech Em Portugues: <http://www.techemportugues.com/2016/06/20/linguagens-usam-os-sites-conhecidos/> <Acessado em: 09/07/2017>

Tutorial JavaScript: <http://www.portaldoctordj.com/apostilas/javascript4/parte3.html> <Acessado em: 10/07/2017>

Curso Básico de JavaScript: <http://www.bosontreinamentos.com.br/programacao-em-javascript/curso-basico-de-javascript-variaveis-e-tipos-de-dados/> <Acessado em: 10/07/2017>

Todo espaço online: <https://www.todoespacoonline.com/w/2014/04/variaveis-em-javascript/> <Acessado em : 17/07/2017>

Google Maps APIs: <https://developers.google.com/maps/> <Acessado em: 17/07/2017>

W3Schools JavaScript Tutorial: <https://www.w3schools.com/js> <Acessado em: 10/07/2017>

GitHub debugout.js: <https://github.com/inorganik/debugout.js> <Acessado em 18/07/2017>

Free Geo IP: <http://freegeoip.net> <Acessado em: 19/07/2017>

Stackoverflow: <stackoverflow.com> <Acessado em: 15/07/2017>

JQuery: <https://jquery.com/> <Acessado em: 10/07/2017>