Estudo sobre a Linguagem de Programação <Java Script>

Aluno Angelo Almeida Ferro e Paulo Fabio dos Santos

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

Angelo.sonic@gmail.com, Paulofabyo@gmail.com

Nesse trabalho, iremos fazer uma abordagem sobre a Linguagem de programação JavaScript, apresentando quais os paradigmas que ela apresenta juntamente com alguns exemplos de como eles funcionam, além disso, será apresentado uma breve história de como ela surgiu, quais motivos levaram a ela ser criada e quais os benefícios que ela proporcionou. A linguagem JavaScript não deve ser confundida com Java, o JS é usado em conjunto com a Linguagem de marcação HTML e CSS e fica responsável pelas ações que são realizadas na página web.

# 1. História do surgimento da linguagem:

O Java Script foi originalmente desenvolvido sob o nome de mocha, posteriormente teve seu nome modificado para LiveScript e, por fim, JavaScript. LiveScript foi o nome oficial da linguagem quando ela foi lançada pela primeira vez na versão beta do navegador Netscape 2.0, em setembro de 1995, mas teve seu nome alterado em um anúncio conjunto com a Sun Microsystems, em dezembro do mesmo ano, quando foi implementado no navegador Netscape, versão 2.0B3.

O fundador do JavaScript foi Brendan Eich, que nasceu em 1961 nos Estados Unidos. O programador iniciou sua carreira na Silicon Graphics, trabalhando por sete anos em [sistemas operacionais](https://pt.wikipedia.org/wiki/Sistema_operacional) e sistemas de rede. Depois atuou por três anos na Micro Unity Systems Engineering, escrevendo microkernels e códigos para [DSPs](https://pt.wikipedia.org/w/index.php?title=Processador_de_sinais_digitais&action=edit&redlink=1), bolando a primeira portagem do compilador [GCC](https://pt.wikipedia.org/wiki/GNU_Compiler_Collection) para [MIPS](https://pt.wikipedia.org/wiki/Arquitetura_MIPS) R4000.

Eich ficou realmente conhecido por seu trabalho na Netscape e no Mozilla. Na Netscape Communications Corporation, começou a trabalhar em abril de 1995, justamente no [JavaScript](https://pt.wikipedia.org/wiki/JavaScript). Depois ajudou a fundar a Mozilla.org em 1998, atuando como arquiteto-chefe. Quando a AOL desativou a divisão para o navegador Netscape em julho de 2003, Eich ajudou a fomentar a Mozilla Foundation.

O JavaScript tem um papel fundamental na evolução da web, pois a ideia central da Internet era ser uma plataforma aberta para todos. Foi superimportante que a sua principal linguagem de programação também fosse aberta, sem controle centralizado de corporações e de fácil aprendizado, ressalta Leonardo Balter, que é engenheiro de software especialista em JavaScript.

Balter lembra que, nesses 20 anos, o JS conseguiu, inclusive, eliminar os temíveis “plug-ins de terceiros” de navegadores, quando vivíamos uma Internet cheia de ferramentas proprietárias e com a performance ruim. “A principal vantagem é que todos os navegadores competiam por uma implementação mais rápida de JavaScript; enquanto isso, os desenvolvedores e os usuários saíam ganhando, daí o JS ficou cada vez melhor por causa dessa concorrência em cima do mesmo produto”, lembra o especialista em Java Script.

Para Eduardo Mattos, programador Fullstack no Medicinia, as principais qualidades do Java Script são a possibilidade e a facilidade em testar a linguagem de forma rápida, o que contribui para o desenvolvedor visualizar o resultado do seu código. “Com Node.js, a linguagem rompeu a barreira da web, e isso é muito significativo na hora de escolher uma linguagem para trabalhar. Já negativamente eu vejo que ainda estamos discutindo compatibilidade de browsers por conta das políticas que estão por trás de cada grande empresa dona de navegadores. Isso impede de estarmos em um estágio mais avançado”, comenta Mattos.

Já Balter enxerga a ECMA como o principal lado positivo do Java Script, pois a fundação faz um trabalho de desenvolvimento contínuo e descentralizado. A ECMA coordena um grupo que tem participação colaborativa de empresas que implementam o run-time do JS, como Mozilla, Google, Microsoft e Apple, além da participação de desenvolvedores web que representam as comunidades de desenvolvimento.

“Um ponto negativo eu digo que é o nome. Primeiro porque o Java Script nunca foi Java, e hoje ele não é apenas um script, mas sim uma linguagem completa de programação. Apesar de ser a linguagem mais popular da web, muitos desenvolvedores subestimam o JS muitas vezes por ser realmente a única linguagem que tem como opção para o front-end”, explica Balter.

# 2. Domínios de aplicação:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Numbers</h1>

<p>Number can be written with or without decimals.</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = 10.50;

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<h1>My First Web Page</h1>

<p>My first paragraph.</p>

<script>

window.alert(5 + 6);

</script>

</body>

</html>

# 3. Paradigmas suportados pela linguagem:

Hoje qualquer linguagem mainstream (que todos conhecem e usam e que possui popularidade) usa [vários paradigmas](http://pt.stackoverflow.com/q/129597/101). O principal de todas elas (só entre as mainstream) é o [imperativo](https://en.wikipedia.org/wiki/Imperative_programming), JS não é diferente e até pode ser usada na forma [procedural](https://en.wikipedia.org/wiki/Procedural_programming) (obviamente de forma [estruturada](https://en.wikipedia.org/wiki/Structured_programming)). JavaScript é conhecida por também ser [orientada a objeto](https://en.wikipedia.org/wiki/Object-oriented_programming) através de [protótipos](https://en.wikipedia.org/wiki/Prototype-based_programming), mas como toda linguagem moderna, vem usando paradigma [funcional](https://en.wikipedia.org/wiki/Functional_programming) cada vez mais.

ORIENTADO A OBJETO:

List = function () {

this.dataStore = [];

};

List.prototype.listSize = 0;

List.prototype.pos = 0;

List.prototype.myMethod = function () {

alert('Este é um método');

};

var list = new List();

IMPERATIVO:

function join(array) {

var text = "";

for (var i = 0; i < array.length; i++) {

text += array[i];

}

return text;

}

FUNCIONAL:

function selecionarTudoAbaixoDeTresReaisFuncional(produtos) {

var itemMenorQueTres = function(item) { return item.preco < 3.00; };

var pegarSomenteNome = function(item) { return item.nome };

return produtos.filter(itemMenorQueTres).map(pegarSomenteNome);

}

PROCEDURAL:

var f = (function (){

var b = 10,

c = 20,

d = 50;

var e = function(){

return b + c + d;

};

return e();

}());

​console.log(f);​

**4. Variáveis e tipos de dados:**

Existem dois tipos de sintaxe de declaração de variável em JavaScript que são:

var nome-da-variável = valor-da-variável;

nome-da-variável = valor-da-variável;

<script>

    var valor1 = 10;

    var valor2 = 3.14;

    document.write(valor1 + " é um valor inteiro, já " + valor2 + " é um ponto flutuante!");

</script>

<script>

    var nome = "André";

    var cidade = "Rio Claro";

    document.write("Meu nome é " + nome + " e sou de " + cidade);

</script>

Perceba a diferença entre a declaração de um número para uma string:  
   
Número: var valor = 10;  
String: var nome = "André";

Quando declaramos uma string utilizamos aspas duplas (") ou simples ('). Você pode declarar número utilizando aspas, mas não é obrigatório.

Em JavaScript só temos os seguintes tipos de variáveis: booleano, constante, null, object e function.

Booleanos retorna verdadeiro ou falso (true/false). Exemplo:

<script>

    var nome = false;

    if (nome == false) {

        document.write("A variável booleana está <b>FALSA</b>.")

    }else {

        document.write("A variável booleana está <b>VERDADEIRA</b>.")

   }

</script>

Constante o valor que não se altera no decorrer do código. Exemplo:

<script>

    const valor = 10;

    document.write("O valor constante é " + valor);

</script>

Null sua variável é vazia. Exemplo:

<script>

    var valor = '';

    document.write("O valor é nulo: " + valor);

    var valor = 10;

    document.write("O valor era nulo agora é: " + valor);

</script>

Object são conhecidas como Object Array, que são uma lista de elementos. Exemplo:

<script>

    nome = new Array(3);

    nome[0]="Andre ";

    nome[1]="Mack ";

    nome[2]="Nardy";

    document.write("Meu nome é " + nome[0] + nome[1] + nome[2])

</script>

Function o valor retornado de uma expressão. Exemplo:

function exemplo(nome) {

    document.write("Meu nome é: " + nome);

}

<body>

    <script>exemplo("André");</script>

</body>

As variáveis são classificadas em dois tipos, em qualquer linguagem de programação, que são as globais e as locais. A diferença entre elas é:

* Variável Global: Criada ou declarada fora de uma função, portanto podem ser utilizadas a qualquer momento no seu script.
* Variável Local: Criada ou declarada dentro de uma função, portanto só podem ser utilizadas dentro da função criada. Variáveis Locais precisam da instrução **var**.

Exemplos:

<script>

    var nome = "João";

    document.write("Nome é " + nome + "");

function exemplo1() {

    var nome = "André";

    document.write("Nome é " + nome + "");

}

</script>

<body>

    <script>

        exemplo1();

        document.write(nome);

    </script>

</body>

Nos exemplos acima temos duas variáveis com o mesmo nome, **var nome**, uma fora e a outra na função. Veja que a variável que recebe o nome João é global, pois foi criada fora de uma função e depois é executa na tag **body**. Já a variável que recebe o nome André é local, pois foi criada dentro de uma função e pertence apenas a está função.

O mais recente padrão ECMAScript define sete tipos de dados:

Seis tipos de dados são os chamados [primitivos](https://developer.mozilla.org/pt-BR/docs/Glossario/Primitivo):

* [Boolean](https://developer.mozilla.org/pt-BR/docs/Glossario/Booleano): true e false.
* [null](https://developer.mozilla.org/pt-BR/docs/Glossario/Nulo)? Uma palavra-chave que indica valor nulo. Devido JavaScript ser case-sensitive,null não é o mesmo que Null, NULL, ou ainda outra variação.
* [undefined](https://developer.mozilla.org/pt-BR/docs/Glossario/undefined): Uma propriedade superior cujo valor é indefinido.
* [Number](https://developer.mozilla.org/pt-BR/docs/Glossary/Number): 42 ou 3.14159.
* [String](https://developer.mozilla.org/pt-BR/docs/Glossario/String): "Howdy"
* [Symbol](https://developer.mozilla.org/pt-BR/docs/Glossary/Symbol) (novo em ECMAScript 6): Um tipo de dado cuja as instâncias são únicas e imutáveis.
* e [Object](https://developer.mozilla.org/pt-BR/docs/Glossario/Objeto" \o "Object: Objeto refere-se a uma estrutura de dados contendo dados e instruções para se trabalhar com estes dados. Objetos algumas vezes se referem a coisas do mundo real, por exemplo um objeto de carro ou um mapa em um jogo de corrida. JavaScript, Java, C++, Py)

**5. Comandos de controle**

Existem algumas estruturas de controle que lhe permitem modificar o fluxo de execução de um programa ou seja elas possuem comandos que condicionam a execução de uma certa tarefa à veracidade ou não de uma determinada condição, ou enquanto determinada condição for verdadeira.

São eles:

Comando IF

if (condição)

{ /\*ação para condição satisfeita\*/ }

[ else

{ /\*ação para condição não satisfeita\*/ } ]

Exemplo :

if (Idade < 18){

alert( "Menor de idade" );

}

else{

alert ("Maior de idade");

}

Comando FOR

for ( [inicialização/criação de variável de controle ;]

[condição ;]

[incremento da variável de controle] )

{ ação }

Exemplo.

for (x = 0 ; x <= 10 ; x++){

alert ("X igual a " + x)

}

Comando WHILE

Executa uma ação enquanto determinada condição for verdadeira.

while (condição)

{ ação }

Exemplo.

var contador = 10

while (contador > 1)

{ contador-- }

Move condicional

receptor = ( (condição) ? verdadeiro : falso)

Exemplo.

NomeSexo = ((VarSexo == "M") ? "Masculino" : "Feminino")

Operadores Lógicos

Os principais operadores lógicos a serem utilizados em comandos condicionais são:

|  |  |
| --- | --- |
| OPERADORES LÓGICOS | SIGNIFICADO |
| = = | Igual |
| != | Diferente |
| > | Maior |
| >= | Maior ou Igual |
| < | Menor |
| <= | Menor ou Igual |
| && | E |
| || | Ou |

Operadores Matemáticos:

Os operadores Matemáticos a serem utilizados em cálculos, referências de indexadores e manuseio de strings são:

|  |  |
| --- | --- |
| OPERADORES MATEMÁTICOS | SIGNIFICADO |
| + | Adição de valor e concatenação de strings. |
| - | Subtração de valores |
| \* | Multiplicação de valores |
| / | Divisão de valores |
| % | Obtém o resto de uma divisão |

Controles Especiais:

|  |  |
| --- | --- |
| CONTROLES ESPECIAIS | SIGNIFICADO |
| \b | backspace |
| \f | form feed |
| \n | new line caracters |
| \r | carriage return |
| \t | tab characters |
| // | Linha de comentário |
| /\*....\*/ | Delimitadores para inserir um texto com mais de uma linha como comentário. |

**6. Escopo (regras de visibilidade)**

A linguagem JavaScript tem dois escopos: global e local. Uma variável declarada fora de uma definição de função é uma variável global, e seu valor será acessível e modificável em todo o seu programa. Uma variável declarada dentro de uma definição de função é local. Ela é criada e destruída sempre que a função é executada e não pode ser acessada por qualquer código fora da função. O JavaScript não suporta escopo de bloco (no qual um conjunto de chaves {. . .} define um novo escopo), exceto em caso especial de variáveis com escopo em bloco.

Uma variável local pode ter o mesmo nome que uma variável global, mas é totalmente separada. A alteração do valor de uma variável não afeta a outra.  Somente a versão local tem significado dentro da função na qual ela é declarada.

=============================================================

// Variável Global.

var aCentaur = "a horse with rider,";

// Variável local declarada na função.

function antiquities(){

//É criada uma variável local com o mesmo nome da variável global, porem seu valor não ira alterar a variável global.

var aCentaur = "A centaur is probably a mounted Scythian warrior";

}

antiquities();

aCentaur += " as seen from a distance by a naive innocent.";

document.write(aCentaur);

// Output: "a horse with rider, as seen from a distance by a naive innocent."

No JavaScript, as variáveis são avaliadas como se fossem declaradas no início do escopo no qual existam.  Às vezes, isso resulta em comportamentos inesperados, como mostrado aqui.

=============================================================

var aNumber = 100;

tweak();

function tweak(){

// Ira imprimir indefinido pois a atribuição está sendo feita depois do procedimento.

document.write(aNumber);

if (false)

{

var aNumber = 123;

}

}

Quando o JavaScript executa uma função, ele primeiro procura todas as declarações de variáveis, por exemplo, var someVariable;.  Ela cria as variáveis com um valor inicial de **undefined**.  Se a variável for declarada com um valor, por exemplo var someVariable = "something";, inicialmente ela ainda terá o valor **undefined** e assumirá o valor declarado somente quando a linha contendo a declaração for executada.

O JavaScript processa todas as declarações de variáveis antes de executar qualquer código, independentemente de a declaração estar dentro ou não de um bloco condicional ou de qualquer outra construção.  Depois de encontrar todas as variáveis, o JavaScript executa o código na função.  Se uma variável estiver implicitamente declarada dentro de uma função, ou seja, se ela aparecer no lado esquerdo de uma expressão de atribuição, mas não tiver sido declarada com **var**, ela será criada como uma variável global.

No JavaScript, uma função interna (aninhada) armazena referências a variáveis locais presentes no mesmo escopo que a função em si, mesmo após os retornos da função.  Este conjunto de referências é chamado um fechamento.  No exemplo a seguir, a segunda chamada para a função interna gera a mesma mensagem (“Hello Bill”) assim como a primeira chamada, porque o parâmetro de entrada para a função externa, *name*, é uma variável local que é armazenada no fechamento da função interna.

=============================================================

function send(name) {

// A variável local é armazenada no fechamento da função interna.

return function () {

sendHi(name);

}

}

function sendHi(msg) {

console.log('Hello ' + msg);

}

var func = send('Bill');

func();

// Output:

// Hello Bill

sendHi('Pete');

// Output:

// Hello Pete

func();

// Output:

// Hello Bill

=============================================================

**7. Exemplo prático de uso da linguagem de programação**

Uma página para cadastro de pessoas para serviço de entrega, onde consta os dados pessoais, endereço e telefone para contato com a mesma. Junto de uma área para o cliente cadastrar um login e senha para que ela possa ter acesso ao sistema. Foi usado o paradigma imperativo pois através dele se torna mais fácil definir as estruturas necessárias para realizar esse cadastro.

# 8. Conclusões:

Java Script é uma linguagem que trouxe grandes benefícios para o mundo atual. Ela teve um papel fundamental para que o desenvolvimento web chegasse ao nível que podemos usar hoje, ela trabalha em conjunto com HTML e CSS, sendo responsável pela parte de ações de uma página web oferecendo diversas ferramentas que auxiliam o desenvolvedor durante a criação dos sistemas.

# 9. Referencias:

<http://imasters.com.br/front-end/javascript/javascript-20-anos-de-historia-e-construcao-da-web/?trace=1519021197>

<http://pt.stackoverflow.com/questions/126886/qual-%C3%A9-o-paradigma-de-programa%C3%A7%C3%A3o-usado-pelo-javascript>

<http://www.w3schools.com/js/js_examples.asp>

<https://www.codigofonte.net/dicas/javascript/696_variaveis-em-javascript>

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Values,_variables,_and_literals>

<http://abrescript.blogspot.com.br/2009/03/javascript-principais-comandos-eventos.html>