INSTITUTO FEDERAL DE MATO GROSSO DO SUL
CAMPUS NAVIRAÍ

NOTAS DE AULA DE

LINGUAGEM DE APRESENTAÇÃO E ESTRUTURAÇÃO DE CONTEÚDOS II

Prof. MSc. Luiz F. Picolo

NAVIRAÍ - MS

1 Introdução

Aprender é a única coisa que a mente nunca se cansa,
nunca tem medo e nunca se arrepende
Autor desconhecido

1.1 O que é JavaScript

O JavaScript foi criado na década de 90 por **Brendan Eich** a serviço da Netscape (uma empresa de serviços de computadores nos EUA a qual era mais conhecida pelo seu navegador, o Netscape). Essa década foi um período de "revolução", pois os navegadores ainda eram estáticos sendo o mais popular dessa época o Mosaic, da NCSA.

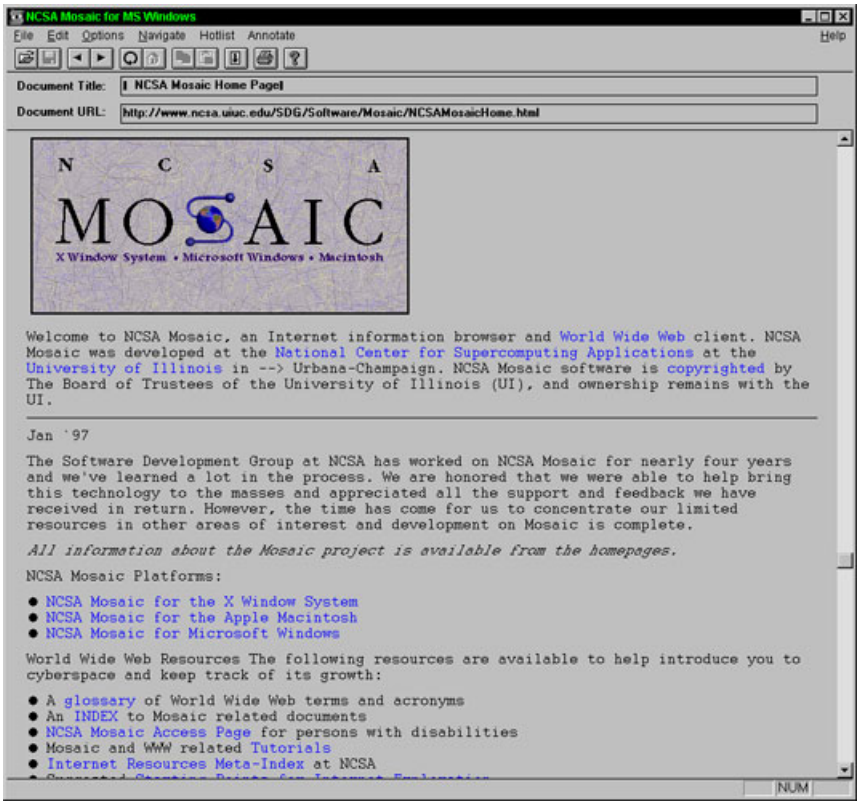

Figura 1 – NCSA Mosaic versão beta

Fonte: [History Computer]

O JavaScript foi introduzido em 1995 como uma forma de adicionar programas às páginas web do navegador Netscape. A linguagem, desde então, foi adotada por todos os outros grandes navegadores web que possuem interfaces gráficas. Ele tornou as aplicações

modernas possíveis, fazendo com que você não tenha que recarregar a página inteira quando for necessário realizar interações diretas com a aplicação. Além disso, ele é usado em páginas web mais tradicionais, fornecendo diferentes maneiras de criar interatividade e inteligência [Haverbeke 2014].

1.2 ECMAScript ou JavaScript?

Depois que o JavaScript foi adotado fora do Netscape, um documento padrão foi escrito para descrever a maneira na qual a linguagem deveria funcionar, garantindo que as diferentes partes dos softwares que afirmavam suportar JavaScript estavam, de fato, falando sobre a mesma linguagem. Esse documento é chamado de padrão ECMAScript, nomeado pela organização internacional Ecma, que foi responsável pela padronização. Na prática, os termos ECMAScript e JavaScript podem ser usados como sinônimos, pois são dois nomes para a mesma linguagem.

Na prática, existem diversos softwares que suportam JavaScript e possuem seu comportamente semelhante. Os navegadores, ou browsers, são exemplos destes tipos de software os quais implementam a linguagem por meio das especificações regidas pelo ECMAScript. Outro mais atual é o NodeJS ou simplesmente Node (https://nodejs.org/) que busca executar o JavaScript diretamente no servidor (Node será aprofundado em capítulos posteriores).

1.3 Executando JavaScript

Para constatar o fato citado na sessão anterior, vamos executar o seguinte código no console em diferentes navegadores. Para tanto, crie um arquivo **index.html** e um outro com o nome de **javascript.js**. No arquivo index.html adicione o seguinte código:

```
1  <!DOCTYPE HTML>
2  <html lang="pt-br">
3  <head>
4      <meta charset="utf-8">
5      <title>JS Exemplos</title>
6  </head>
7  <body>
8    <script type="text/javascript" src="javascript.js"></script>
9  </body>
10 </html>
```

E no arquivo **javascript.js** adicione o seguinte:

```
1 alert('Bem vindo(a) ao JavaScript')
```

O código acima possui o mesmo comportamento? Sim, o comportamento é o mesmo em todos os navegadores. Contudo, a forma gráfica com que é apresentado faz parte da implementação feita. Portanto, o desenvolvedor pode utilizar o JavaScript nos navegadores sem muito problema, pois eles seguem não ideias da equipe que o criou mas uma especificação que dita as regras de como determinadas funções devem se comportar.

1.4 Versões do JavaScript ou edições ECMAScript

Como foi visto, o ECMAScript é apenas a especificação. Contudo, ao estudar a linguagem é muito comum escutar a abreviação de ECMAScript, ou seja, ES. Assim, sempre que se ler ES seguido de um número, esse está fazendo referência a uma edição do ECMAScript. Atualmente, existem oito edições do ECMAScript publicadas, sendo que, a partir de 2015, as edições começaram a receber o ano e não mais o número da edição.

1. ECMAScript 1 (1997)
2. ECMAScript 2 (1998)
3. ECMAScript 3 (1999)
4. ECMAScript 4 Nunca foi lançada.
5. ECMAScript 5 (2009)
    5.1. ECMAScript 5.1 (2011)
6. ECMAScript 2015
7. ECMAScript 2016
8. ECMAScript 2017
9. ECMAScript 2018

1.5 Conclusão

Portanto, JavaScript se tornou a linguagem de programação mais popular no desenvolvimento Web sendo suportada por todos os navegadores e responsável por praticamente qualquer tipo de dinamismos em páginas web. Ao se usar todo o poder que ela tem para oferecer, pode-se chegar a resultados impressionantes. Alguns excelentes exemplos disso são aplicações Web complexas como Gmail, Google Maps e Google Docs.

2 Tipos de dados, Variáveis e Constantes

Aquele que não luta pelo futuro que quer,
deve aceitar o futuro que vier
Autor desconhecido

Definir o que são variáveis ou constantes de forma geral é extremamente simples. Uma variável nada mais é do que um espaço que o sistemas operacional (SO) aloca na memória de seu computador para que seja armazenado algo. Ou seja, quando precisamos armazenar um determinado valor nosso SO cria um espaço e permite ao usuário determinar o que será adicionado. A medida que não será utilizado este espaço com um valor, ele será dissipado, fazendo com que seus disponibilizando-os para serem usados na construção de outros novos valores [Haverbeke 2014].

Mas como é criado este valores? Eu preciso entender de memoria, sistemas operacionais, entre outros para poder alocar este valores? A resposta é não, não é necessário a compreenção de tão baixo nível, basta que se compreenda como é criar variáveis ou constantes em um determinada linguagem, que, neste caso, é JavaScript.

2.1 Tipos de dados

Diferente das linguagens com tipagem forte, ou seja, que levam em consideração o tipo para executar suas operações, o JavaScript possui tipagem fraca, a qual é dado o tipo dos dados que serão alocados quando o código é executado. Outro ponto é a tipagem dinâmica. Logo, não é necessário tipar uma variável quando inicia-se o programa como é feito em Java, C, C++, entre outras linguagens, mas esta responsabilidade é repassada ao compilador, que, caso do JavaScript em questão, esse processo é dado o nome Inferência de Tipo.

A inferência de tipos é a capacidade do compilador entender/'adivinhar' qual é o tipo de dados de determinada variável sem ela ter sido declarada no código escrito.

Segundo a MDN Web Docs Mozilla 2019, JavaScript reconhece os seguintes tipos de valores:

- Números (en), como 42 ou 3,14159
- Valores lógicos (Booleanos) (en), true ou false

- Strings (en), tais como "Howdy!"
- null, um palavra chave especial denotando um valor nulo; null também é um valor primitivo. Como JavaScript é sensível a maiúsculas, null não é a mesma coisa que Null, NULL, ou qualquer outra variante
- undefined (en), uma propriedade de alto nível a qual possui o valor indefinido; undefined também é um valor primitivo.

Este conjunto de tipos de valores, relativamente pequeno, permite realizar funções úteis nas aplicações. Não há distinção explícita entre números inteiros e reais, para JavaScript, todos os números são tratados como *Number*.

2.2 Variáveis

O JavaScript é uma linguagem de tipagem dinâmica. Isto significa que não é necessário especificar o tipo de dado de uma variável quando ela for declarada, e tipos de dados são automaticamento convertidos conforme necessário durante a execução do script. Então, por exemplo, pode-se definir uma variável como:

```
1  var idade = 30
2  var instituicao = "IFMS"
```

Para que possamos saber o tipo que foi atribuido para cada variável, podemos utilizar a função **typeof** da seguinte forma:

```
1  var idade = 30
2  console.log(typeof idade) // Retornará number
```

Outra forma de declarar variáveis é utilizando a palavra reservada **let**.

```
1  let idade = 30
2  let instituicao = "IFMS"
```

Mas qual a diferença entre **var** e **let**? Neste momento não possuimos o conhecimento necessário para diferenciar as duas palavras reservadas. Contudo, veremos nos próximos capítulos como as duas se diferenciam e quando utilizar uma ou outra.

2.2.1 Hoinsting

Segundo a MDN Web Docs Mozilla 2019, em JavaScript, funções (a qual será vista em um outro capítulo) e variáveis são hoisted (ou "levados ao topo"). Hoisting é um

comportamento do JavaScript de mover declarações para o topo de um escopo (o escopo global ou da função em que se encontra). Isso significa que nós somos capazes de usar uma função ou variável antes mesmo de tê-las declaradas, ou em outras palavras: uma função ou variável podem ser declaradas depois de já terem sido utilizadas.

```
1  foo = 2
2  var foo;
3
4  // é implicitamente entendido como:
5  var foo;
6  foo = 2;
```

Assim, as variáveis quando são declaradas sem um valor, recebem o valor **undefined**, ou seja, sem tipo.

2.3 Constantes

Podemos criar elementos "somente leitura", nomeados constantes com a palavra chave **const**. A sintaxe de um identificador constante é a mesma para um identificador de variáveis: deve começar com uma letra ou sublinhado e pode conter caracteres alfabéticos, numéricos ou sublinhado. Porém, uma constante não pode ter seu valor mudado por meio de uma atribuição ou ser declarada novamente enquanto o *script* estiver rodando.

```
1  const idade = 30
2  console.log(typeof idade) // Retornará number
```

2.4 Tipo básico de dados

Essa sessão introduz os elementos que representam os tipos de valores e os operadores que podem atuar sobre eles presentes na linguagem que estamos estudanto.

2.4.1 Números

Em JavaScript todos números são tratados como *Number*. Ao criar uma variável contendo um número inteiro ou decimal qualquer, esta variável em algumas linguagem seriam tratadas como Integer ou Float por exemplo. Contudo,em JavaScript, esta recebe o tipo *Number* independente do seu valor.

```
1  var idade = 30
2  console.log(typeof idade) // Retornará number
```

```
3
4   var peso = 65.5
5   console.log(typeof peso) // Retornará number
```

2.4.1.1 Números Especiais

Existem três valores especiais no JavaScript que são considerados números, mas não se comportam como números normais.

Os dois primeiros são Infinity e -Infinity, que são usados para representar os infinitos positivo e negativo. O cálculo Infinity - 1 continua sendo Infinity, assim como qualquer outra variação dessa conta. Entretanto, não confie muito em cálculos baseados no valor infinito, pois esse valor não é matematicamente sólido e rapidamente nos levará ao próximo número especial: NaN [Haverbeke 2014].

NaN é a abreviação de "not a number" (não é um número), mesmo sabendo que ele é um valor do tipo número. Você receberá esse valor como resultado quando, por exemplo, tentar calcular 0 / 0 (zero dividido por zero), Infinity - Infinity ou, então, realizar quaisquer outras operações numéricas que não resultem em um número preciso e significativo [Haverbeke 2014].

2.4.2 Strings

O próximo tipo básico de dado é a String. Strings são usadas para representar texto, e são escritas delimitando o seu conteúdo entre aspas. Ambas as aspas simples e duplas podem ser usadas para representar Strings, contanto que as aspas abertas sejam iguais no início e no fim.

```
1   var nome = "Genoveva"
2   console.log(typeof nome) // Retornará String
```

Existem outras maneiras de manipular as Strings, as quais serão discutidas nos próximos capitulos

2.4.3 Valores Booleanos

Você frequentemente precisará de um valor para distinguir entre duas possibilidades, como por exemplo "sim" e "não", ou "ligado" e "desligado". Para isso, o JavaScript possui o tipo Booleano, que tem apenas dois valores: verdadeiro e falso (que são escritos como true e false respectivamente).

```
1   console.log(3 > 2) // Retornará true
```

```
2  console.log(3 < 2) // Retornará false
```

2.4.4 Valores Indefinidos

Existem dois valores especiais, null e undefined, que são usados para indicar a ausência de um valor com significado. Eles são valores por si sós, mas não carregam nenhum tipo de informação. A diferença de significado entre undefined e null é um acidente que foi criado no design do JavaScript, e não faz muita diferença na maioria das vezes. Podemos comprovar isso comparando ambos

```
1  console.log(null == undefined); // Retornará True
```

2.5 Aritmética

Para que possamos realizar os cálculos básicos em JavaScript, podemos usar os operadores que já estamos acostumados

```
1  2 + 2 // Soma
2  5 - 4 // Subtrai
3  2 * 3 // Multiplica
4  4 / 2 // Divide
5  144 % 12 // Retorna o Resto da divisão
```

Mas o que acontece quando adicionamos os operados acima a Strings?

```
1  "Java" + "Script" // JavaScript
2  "Java" - "Script" // NaN
3  "Java" * "Script" // NaN
4  "Java" / "Script" // NaN
5  "Java" % "Script" // NaN
```

Ou seja, com o valor Aritmético + as Strings são concatenadas.

2.6 Exercicios 001

1. Crie três variáveis contendo três notas. Depois, calcule a média das três notas e mostre-a na tela por meio de um **Alert**.

2. Crie cinco variáveis contendo cinco números inteiros. Logo após imprima o quadrado de cada número.

Referências

HAVERBEKE, M. *Eloquent javascript: A modern introduction to programming*. [S.l.]: No Starch Press, 2014. Citado 3 vezes nas páginas 3, 5 e 8.

HISTORY Computer. Disponível em: <https://history-computer.com/Internet/Conquering/Mosaic.html>. Acesso em: 09 julho. 2019. Citado na página 2.

MDN Web Docs Mozilla. 2019. Disponível em: <https://developer.mozilla.org/pt-PT/docs/Web/JavaScript/Guia/Valores,_Variáveis_e_Literais>. Citado 2 vezes nas páginas 5 e 6.