

AULA 4 – VARIÁVEIS, COMANDOS DE ENTRADA/SAÍDA E OPERADORES

OBJETIVO DA AULA

Conhecer os comandos básicos do JavaScript.

APRESENTAÇÃO

Nesta aula aprofundaremos nosso conhecimento na linguagem JavaScript. Aprenderemos o conceito de variáveis e os comandos básicos de entrada e saída da linguagem. Além disso, conheceremos também os seus operadores básicos: aritméticos, relacionais e lógicos. Os operadores nos permitem criar expressões que serão avaliadas ao longo do desenvolvimento. É importante lembrar que nesta linguagem as instruções são chamadas de declaração e são sempre separadas por um ponto e vírgula (;). Além disso, JavaScript é **case sensitive**, isso significa que a linguagem diferencia maiúsculo de minúsculo (num1 é diferente de Num1), precisamos ficar atentos a este detalhe.

VARIÁVEIS

O que é uma variável? De acordo com Flanagan (2013):

(...) quando um programa precisa manter um valor para uso futuro, ele atribui o valor (ou “armazena” o valor) a uma variável. A variável define um nome simbólico para um valor e permite que o valor seja referido pelo nome.

As variáveis são espaços de memória RAM, vale ressaltar que, no caso do JavaScript, uma variável só existe no espaço de memória ocupado por uma página aberta no navegador. Quando a página exibida na janela do navegador é modificada, todas as variáveis criadas pelos scripts desta página são perdidas.

Temos a liberdade para criar os nomes das variáveis, mas precisamos ter atenção as seguintes regras:

- Os nomes de variável podem incluir letras do alfabeto, tanto minúsculas como maiúsculas, lembrando que JavaScript é *case sensitive*;
- Devem começar com uma letra, underline (_), ou cifrão (\$);
- Elas podem conter números, porém não pode começar por eles;
- Os nomes de variável não podem incluir espaços nem quaisquer outros caracteres de pontuação.

Livro Eletrônico

Podemos declarar uma variável de três formas diferentes, são elas:

- **var**: muito utilizada antes da versão EcmaScript 6. Por conta de alguns problemas com o **var**, a nova versão trouxe novas maneiras de declarar variáveis (*let* e *const*). Declarações com **var** tem escopo global ou escopo de função/local. Além disso, variáveis declaradas como **var** podem ser declaradas de novo e atualizadas. Vamos entender o que isso significa.... No exemplo abaixo, podemos observar que a variável **hello** tem o seu escopo global, pois está fora de uma função, enquanto a variável **curso** tem o seu escopo de função/local, ou seja, não pode ser acessada fora da função. Vejamos:

```

1  var hello = "Hello World";
2
3  function newfunction()
4  {
5      var curso = "Gran Cursos Online";
6  }
7  console.log(curso);

```

Uncaught ReferenceError: curso is not defined

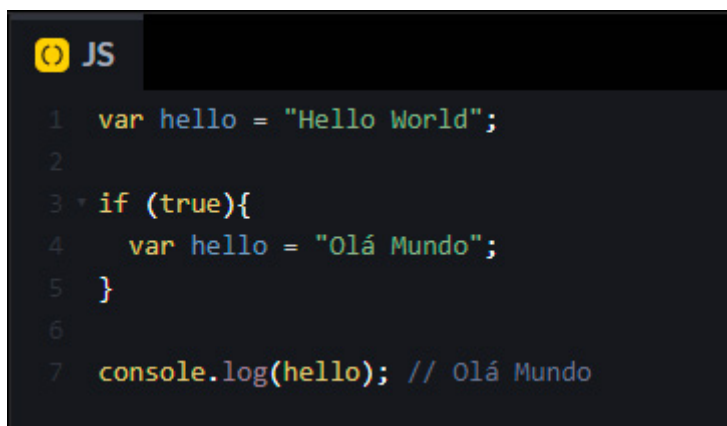
Observe que neste exemplo obtivemos um erro, que nos diz que a variável **curso** não está disponível fora da função *newfunction*. Além disso, variáveis declaradas como **var** podem ser declaradas novamente e atualizadas. Vejamos:

```

1  var hello = "Hello World";
2  var hello = "Olá"; //declarada novamente
3  hello = "Olá" //atualizada
4
5  console.log(hello); // Olá

```

O fato dela poder ser atualizada em qualquer parte do escopo e não gerar erro pode trazer alguns problemas. Vejamos:



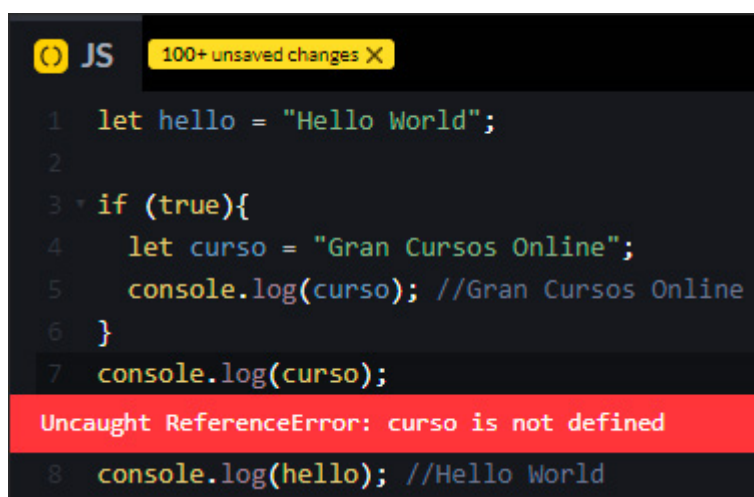
```

JS
1  var hello = "Hello World";
2
3  if (true){
4    var hello = "Olá Mundo";
5  }
6
7  console.log(hello); // Olá Mundo

```

Uma vez que a condição for verdadeira, a variável **hello** é redefinida como “Olá Mundo”. Isso pode não ser um problema, caso você queira que a variável **hello** seja redefinida. Agora, se você já usou a variável **hello** em outras partes do seu código, poderá ser surpreendido com o resultado que poderá obter. Portanto, tome cuidado! Isso pode causar muitos bugs no seu código, por esse motivo foram criadas outras maneiras de declarar como *let* e *const*.

- **let**: usada para declarar uma variável local de escopo de bloco. O que é um bloco? Um bloco nada mais é do que um porção de código envolto por {}. Sendo assim, uma variável declarada como **let** em um bloco, só estará disponível dentro daquele bloco. Vejamos:



```

JS 100+ unsaved changes X
1  let hello = "Hello World";
2
3  if (true){
4    let curso = "Gran Cursos Online";
5    console.log(curso); //Gran Cursos Online
6  }
7  console.log(curso);
Uncaught ReferenceError: curso is not defined
8  console.log(hello); //Hello World

```

Podemos observar que o uso da variável **curso** fora de seu bloco {} retornou um erro e o uso da variável **hello** não apresentou erro porque se encontra dentro do seu escopo de bloco. Assim como no **var**, a variável declara como **let** pode ser atualizada, porém, não pode ser declarada novamente.

```
JS
1 let hello = "Hello World";
2 hello = "Olá Mundo";
3 console.log(hello);
```

```
JS 100+ unsaved changes X
1 let hello = "Hello World";
2 let hello = "Olá Mundo";
Uncaught SyntaxError: Identifier 'hello' has already been declared
3 console.log(hello);
```

Diferentemente do uso do **var**, com o **let** se a mesma variável for definida em escopos diferentes, serão consideradas variáveis diferentes, já que são de escopos diferentes. Vejamos:

```
JS
1 let hello = "Hello World";
2
3 if (true){
4   let hello = "Olá Mundo";
5   console.log(hello); //Olá Mundo
6 }
7 console.log(hello); // Hello World
```

Isso faz com que o uso do **let** seja mais apropriado do que o uso do **var**, tendo em vista que você não precisa se preocupar se usou este nome de variável em algum outro momento, já que a variável existe somente dentro daquele escopo. Além disso, como uma variável não pode ser declarada mais de uma vez em um escopo, o problema que ocorre com **var** mencionado anteriormente, não acontece.

- **const**: a variável declarada como *const* possui seu valor constante, ou seja, seu valor não pode ser modificado. “Uma constante se comporta em relação ao escopo onde foi declarada da mesma forma que uma variável declarada como **let**”, ou seja, só pode ser acessada dentro do bloco que foi declarada (DEV MEDIA, 2020). A diferença é que uma constante não pode ser atualizada e nem declarada novamente.

```
JS 100+ unsaved changes X
1 const hello = "Hello World";
2 hello = "Olá";
Uncaught TypeError: Assignment to constant variable.
```

```
JS 100+ unsaved changes X
1 const hello = "Hello World";
2 const hello = "Olá";
Uncaught SyntaxError: Identifier 'hello' has already been declared
```

DESTAQUE

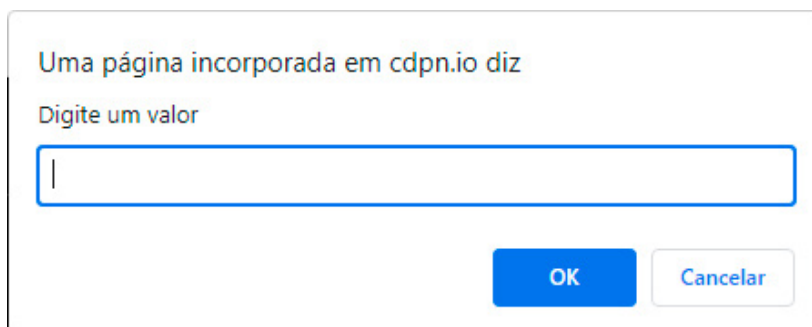
Resumindo:

(...) as declarações de **var** têm escopo global ou escopo de função enquanto **let** e **const** tem escopo de bloco. Variáveis **var** podem ser atualizadas e declaradas novamente dentro do seu escopo. Variáveis **let** podem ser atualizadas, mas não declaradas novamente. Variáveis **const** não podem ser atualizadas nem declaradas novamente (JESUS, 2021).

COMANDOS DE ENTRADA/SAÍDA

O objeto *document*, que representa a página propriamente dita, possui o método *write*, que juntos formam a instrução **document.write()** e possui a função de escrever informações na página para o usuário. Nela é possível escrever strings, conteúdo de variáveis e tags HTML. Qualquer expressão válida em JavaScript pode ser escrita com a utilização desse método. Logo, **documento.write** pode ser utilizado como método de saída. Além disso, podemos utilizar o **console.log()**, como visto nos exemplos anteriores, para acompanhar os valores de variáveis durante a execução de algum código.

Podemos utilizar o método **prompt()** como método de entrada. Este método faz exibir uma janela com uma mensagem e um campo aberto para que o usuário possa digitar informações.



A janela aberta pelo **prompt()** possui dois botões, *Ok* e *Cancelar*, usados para confirmar e cancelar a entrada de dados, respectivamente. A informação digitada pode ser armazenada em uma variável.

OPERADORES

“Os operadores são utilizados em JavaScript para expressões aritméticas, expressões de comparação, expressões lógicas, expressões de atribuição e muito mais” (FLANAGAN, 2013). Vejamos:

- **Operadores Aritméticos:** retornam o resultado da operação aritmética entre os valores.

| Operador | Descrição | Exemplo |
|----------|--|----------|
| + | retorna a soma entre dois valores ou concatena strings | $a + b$ |
| - | retorna a diferença entre o primeiro e o segundo valor | $a - b$ |
| * | retorna o produto entre dois valores | $a * b$ |
| / | retorna o quociente da divisão do primeiro valor pelo segundo | a / b |
| % | retorna o módulo (resto) da divisão inteira do primeiro valor pelo segundo | $a \% b$ |
| - | retorna o inverso de uma variável | -a |

- **Operadores Relacionais (comparação):** retornam um resultado booleano, ou seja, *true* ou *false*.

| Operador | Descrição | Exemplo |
|----------|--|---------|
| == | verifica se dois valores são iguais entre si | a == b |
| != | verifica se dois valores são diferentes entre si | a != b |
| > | verifica se o primeiro valor é maior que o segundo | a > b |
| >= | verifica se o primeiro valor é maior ou igual ao segundo | a >= b |
| < | verifica se o primeiro valor é menor que o segundo | a < b |
| <= | verifica se o primeiro valor é menor ou igual ao segundo | a <= b |

- **Operadores lógicos:** efetuam comparações lógicas entre dois valores.

| Operador | Descrição | Exemplo |
|----------|---|---------|
| && | comparação do tipo “E”, retornando true caso os dois valores sejam verdadeiros e false caso pelo menos um dos valores seja falso | a && b |
| | comparação do tipo “OU”, retornando true caso pelo menos um dos valores seja verdadeiro e false caso os dois valores sejam falsos | a b |
| ! | comparação do tipo “NÃO”, retornando true caso o valor seja falso e false caso o valor seja verdadeiro | !a |

- **Operadores de atribuição:** são utilizados como forma de abreviar determinadas expressões.

| Operador | Descrição | Exemplo |
|----------|---|------------|
| += | atribui a primeira variável o valor da soma dela mesma com a segunda variável | a += b |
| -= | atribui a primeira variável o valor da subtração dela mesma com a segunda variável | a -= b |
| *= | atribui a primeira variável o valor do produto dela mesma com a segunda variável | a *= b |
| /= | atribui a primeira variável o valor da divisão dela mesma pela segunda variável | a /= b |
| %= | atribui a primeira variável o valor do módulo (resto) da divisão inteira dela mesma pela segunda variável | a %= b |
| ++ | acrescenta 1 a variável | a++ ou ++a |
| -- | diminui 1 da variável | a-- ou --a |

CONSIDERAÇÕES FINAIS

Nesta unidade aprendemos os principais conceitos de JavaScript: variáveis, comandos de entrada/saída e operadores. Vimos que variáveis podem ser declaradas de três formas diferentes. Variáveis declaradas como **var** podem ser atualizadas e declaradas novamente, desde que seja dentro do seu escopo. Variáveis declaradas como **let** podem ser atualizadas, porém, não podem ser declaradas novamente, e ao contrário do **var** e do **let**, as variáveis declaradas como **const** não podem ser atualizadas e muito menos declaradas novamente. Com esses conceitos já conseguimos fazer uns scripts básicos. Na próxima aula colocaremos esses conceitos em prática.

MATERIAIS COMPLEMENTARES

Operadores Aritméticos:

<https://youtu.be/wkradRZd93g>

Operadores Relacionais (comparação):

<https://youtu.be/hoPy34YaR4U>

Operadores Lógicos:

<https://youtu.be/ivGxVPnWSCA>

Var, Let e Const:

<https://youtu.be/ZOx7iTnBqFQ>

REFERÊNCIAS

FLANAGAN, David. *JavaScript: o guia definitivo*. Porto Alegre, RS: Bookman, 2013.

JAVASCRIPT: Variáveis e constantes. *DevMedia*, 2020. Disponível em: <<https://www.devmedia.com.br/javascript-variaveis-e-constantes/41012>> Acesso em: 05 de nov. de 2022.

JESUS, Gabriel. *Qual a diferença entre var, let e const no JavaScript?* 2021. Disponível em <<https://pt.linkedin.com/pulse/qual-diferen%C3%A7a-entre-var-let-e-const-javascript-gabriel-de-jesus>> Acesso em: 05 de nov. de 2022.