P – Aulas de JavaScrip

**H1 - JavaScript**

**H2 – Aulas**

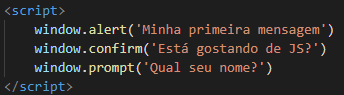
**H3 – de**

***H4 – JS***

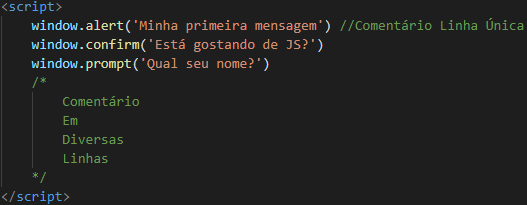
1. **Softwares Necessários**

* Navegador (Chrome ou Edge)
* Node JS
* Visual Studio Code

1. **Primeiros Comandos JavaScrip**

  
>> <https://github.com/viniciusschnoor/JavaScript-Learning/blob/main/04/ex001.html>

1. **Comentando o Código**

  
>> <https://github.com/viniciusschnoor/JavaScript-Learning/blob/main/04/ex001.html>

// - Para comentário de linha única

/\*\*/ - Para comentários de diversas linhas

1. **Variáveis**
2. Delimitar
3. Nomear
4. Definir tamanhos

Exemplificando, delimitar as vagas em um estacionamento, nomear as vagas para encontrar onde um carro específico está e definir o tamanho para diversas categorias de veículos, como vagas para carros, caminhões e motos.

Para colocar o carro, definir qual vaga receberá que carro com “=”. O sinal de igual sozinho, em JavaScript, chamará “recebe”. vaga01 = carro01

Cada vaga receberá apenas um carro, mas podemos definir que nenhum veículo entrará naquela vaga usando “null”. vaga01 = null

Agora com o mesmo raciocínio, dentro da memória do computador (estacionamento), delimite espaços da memória (delimitando as vagas) que receberão as informações (carros).

As vagas passam a se chamar variáveis e, ao invés de definir vagas, definimos var (ou let, no JavaScript também é permitido usar esta palavra).

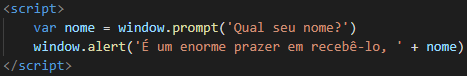
Os carros passam a ser os valores, podemos definir quaisquer valores utilizando o sinal de igual (=). var n1 = 5 >> Neste caso, definimos a variável n1 (o nome da variável é chamado de identificador) e indicamos que ela receberá o valor 5.

* 1. **Identificadores**

Os nomes dos identificadores devem seguir algumas regras, são elas:

1. Podem começar com letras, $ ou \_;
2. Não podem começar com números;
3. É possível usar letras ou números;
4. É possível usar acentos e símbolos;
5. Não podem conter espaços, eles normalmente são substituídos pelo sinal de sublinhado (\_);
6. Não podem ser palavras reservadas (palavras que o JavaScript usa como comandos, por exemplo, var
7. **Guardando Informações**

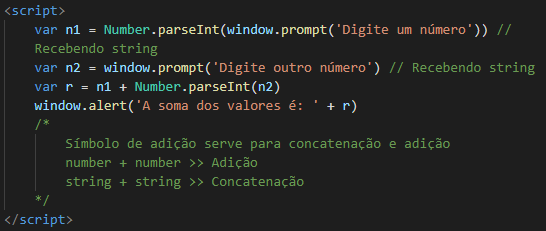
Simplesmente, para guardar uma informação, basta definir uma variável antes da informação que será armazenada. Por exemplo, o comando window.prompt(‘Qual seu nome?’) se torrnará var nome = window.prompt(‘Qual seu nome?’).

  
>> <https://github.com/viniciusschnoor/JavaScript-Learning/blob/main/06/ex002.html>

* 1. **Tratamento de Dados**
     1. **Inteiros e Reais**

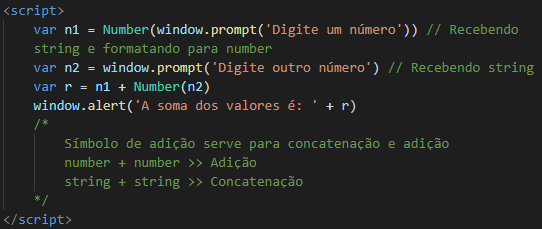
Number.parseInt(n) >> Converte número para número inteiro

Number.parseFloat(n) >> Converte número em número real

  
>> <https://github.com/viniciusschnoor/JavaScript-Learning/blob/main/06/ex003.html>

Pode-se verificar que temos duas maneiras de converter o dado, antes de colocar na variável (neste caso em n1) ou antes de realizar qualquer atividade com a variável (neste caso, usar n2 que é uma string, formata-la para inteiro e somar).

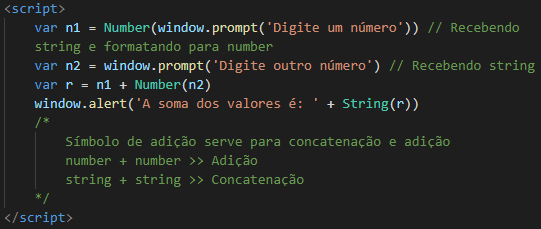
Number(n) >> Identifica automaticamente qual tipo de número e o converte

  
>> <https://github.com/viniciusschnoor/JavaScript-Learning/blob/main/06/ex003.html>

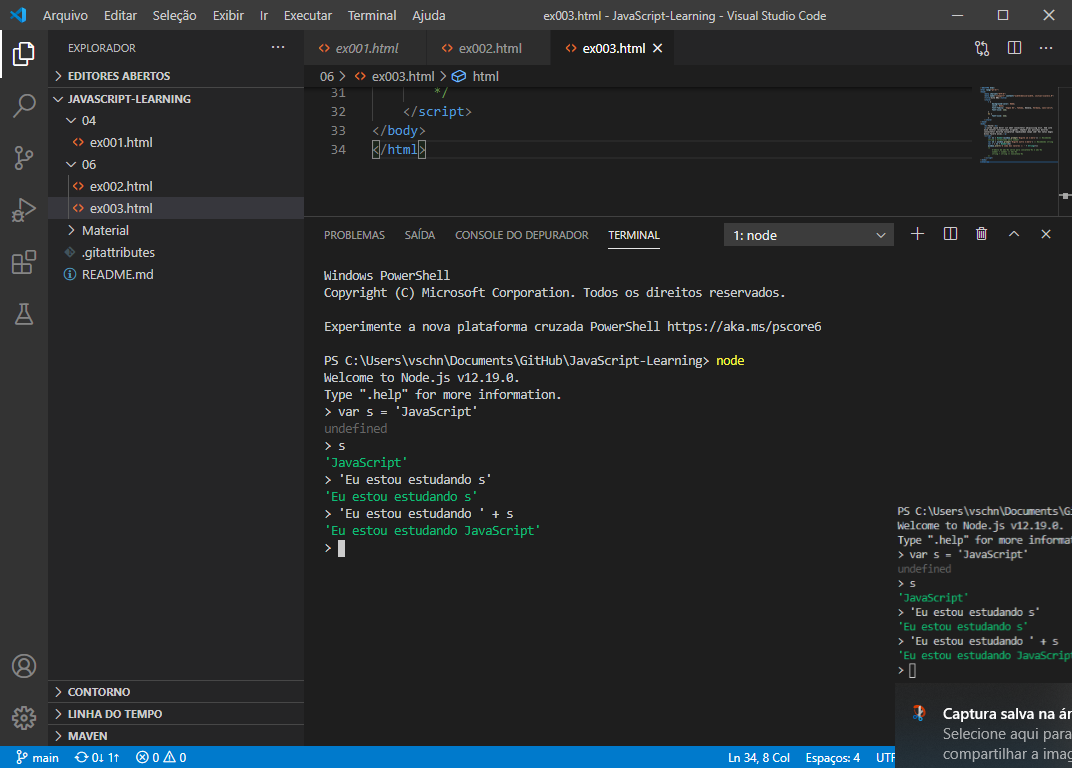
* + 1. **Strings**

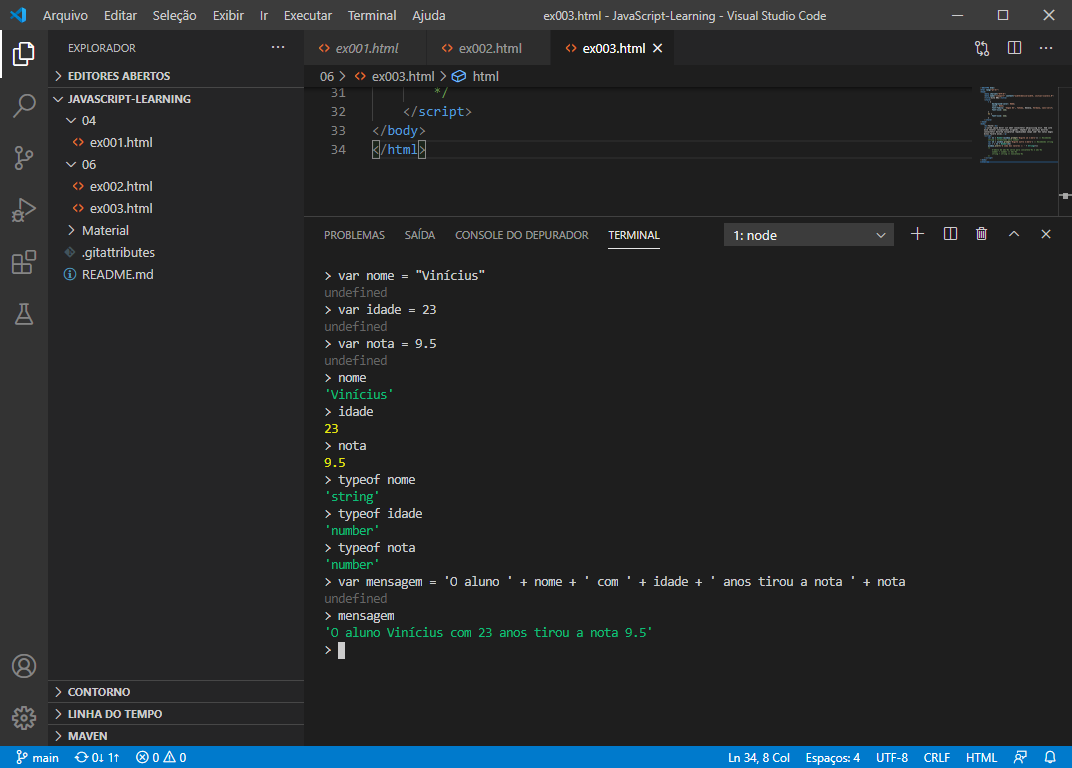
String(n) >> Converte número em string

n.toString() >> Converte número em string

  
>> <https://github.com/viniciusschnoor/JavaScript-Learning/blob/main/06/ex003.html>

Neste caso a formatação para string não se fazia necessária, foi feita apenas para efeito de exemplificação. Veja o mesmo exemplo usando o terminal:

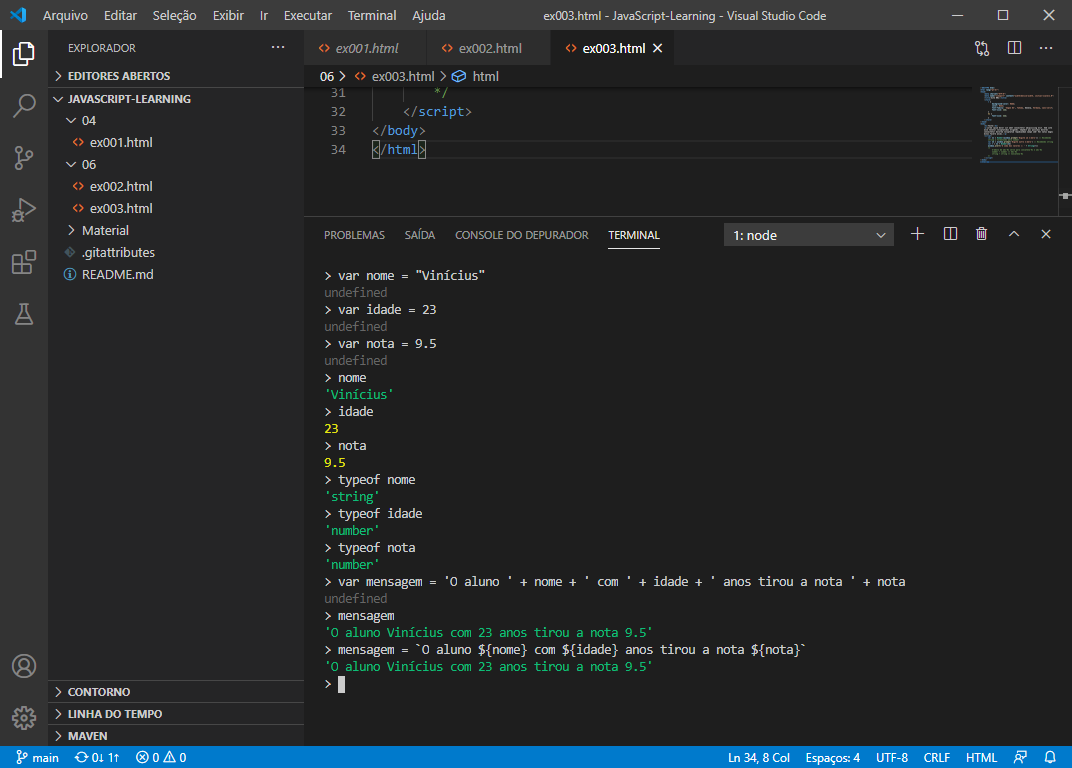
  
>> Ctrl + Shift + ‘ para abrir o Terminal do Visual Studio Code

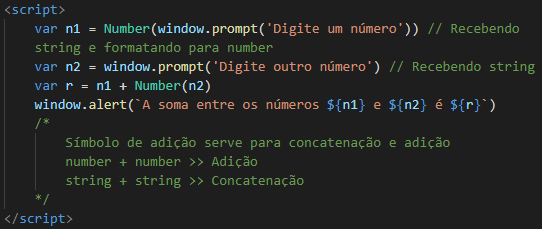
  
>> Repare a concatenação, precisa separar por aspas simples e o sinal de adição a cada alteração entre string e variável

Para resolver o problema acima, podemos utilizar da Template String utilizando crases (``) e o Place Holder (${}).

Por exemplo:

|  |
| --- |
| var s = ‘JavaScript’  ‘Eu estou usando s’ // Não faz interpolação  ‘Eu estou usando ‘ + s // Usando concatenação  `Eu estou usando ${s}` // Usando Template String |

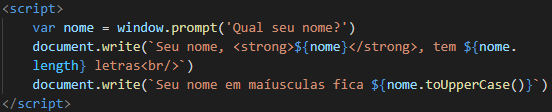
  
>> Repare na diferença entre os comandos

  
>> <https://github.com/viniciusschnoor/JavaScript-Learning/blob/main/06/ex003.html>

s.length >> Mostra quantidade de caracteres de uma string (sem parêntese, um atributo)

s.toUpperCase() >> Coloca caracteres em MAIÚSCULAS (com parêntese, um método)

s.toLowerCase() >> Coloca caracteres em minúsculas

  
>> <https://github.com/viniciusschnoor/JavaScript-Learning/blob/main/06/ex004.html>  
>> Repare na inclusão de tags HTML dentro da Script  
>> Repare também o uso do atributo .length e do método .toUpperCase

* + 1. **Números**

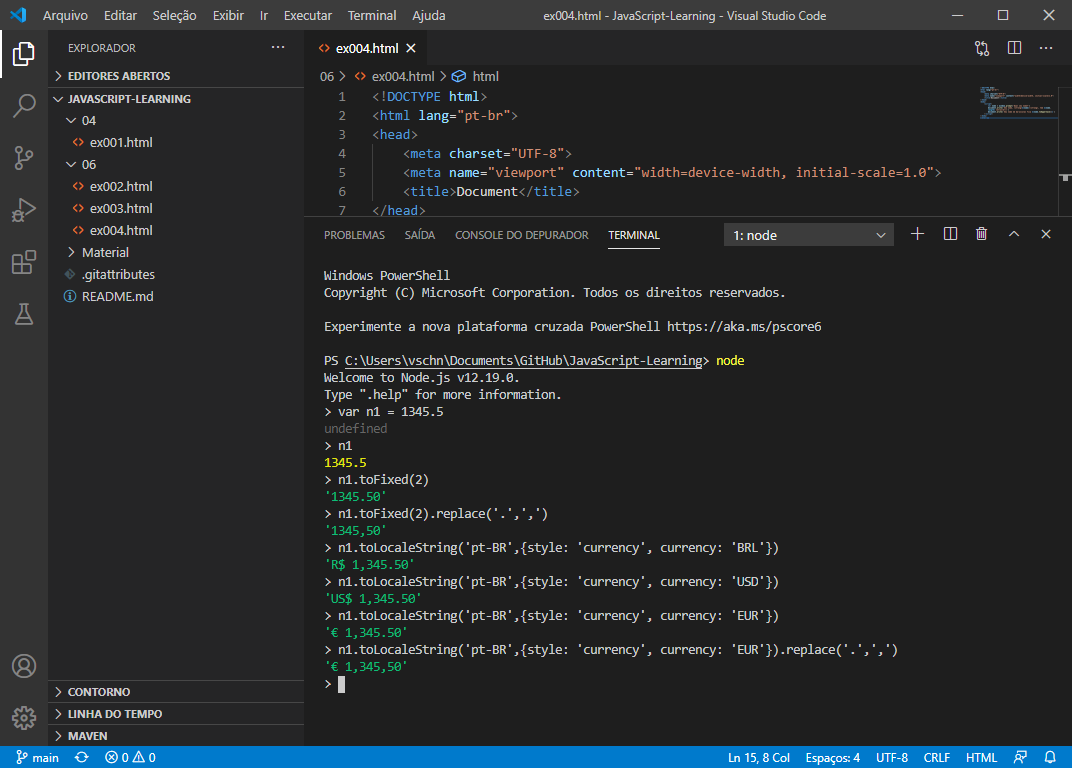
n1.toFixed(2) >> Define duas casas decimais

n1.toFixed(2).replace( ‘.’ , ’,’ ) >> Substitui ponto por vírgula

n1.toLocaleString(‘pt-BR’, {style: ‘currency’, currency: ‘BRL’}) >> Define como moeda Real

n1.toLocaleString(‘pt-BR’, {style: ‘currency’, currency: ‘USD’}) >> Define como moeda Dólar

n1.toLocaleString(‘pt-BR’, {style: ‘currency’, currency: ‘EUR’}) >> Define como moeda Euro

  
>> Terminal do Visual Studio Code  
>> Não é necessário o uso do .replace() no último caso, no navegador será convertido automaticamente

1. **Operadores**

Tipos de Operadores:

1. Aritméticos
2. Atribuição
3. Relacionais
4. Lógicos
5. Ternário
   1. **Aritméticos**

|  |  |  |  |
| --- | --- | --- | --- |
|  | OPERADOR |  | RESULTADO |
| 5 | **+** | 2 | 7 |
| **-** | 3 |
| **\*** | 10 |
| **/** | 2.5 |
| **%** | 1 |
| **\*\*** | 25 |

>> Operador de Exponenciação não existia em versões anteriores

Cuidado!

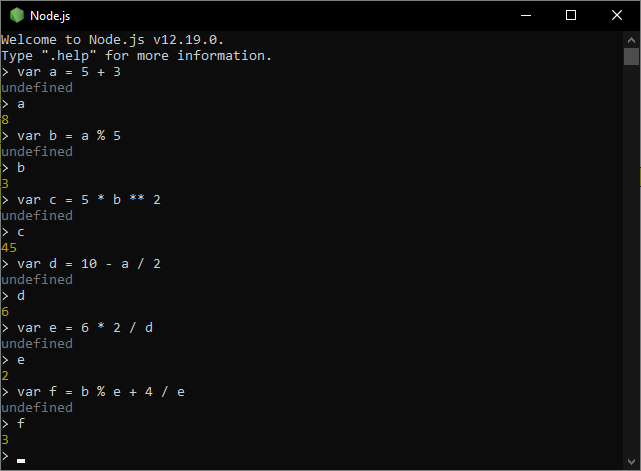
5 + 3 / 2 = 6.5 >> Importância da Divisão no cálculo

( 5 + 3 ) / 2 = 4.0 >> Parênteses altera Importância do cálculo

|  |  |  |
| --- | --- | --- |
| () |  |  |
| \*\* |  |  |
| \* | / | % |
| + | - |  |

>> Ordem de Importância de cima para baixo

Exemplo aplicado no Terminal do Node.js:

  
>> Terminal do Node.js

* + 1. **Auto Atribuições**

Relembrando a comparação com estacionamento e vagas para veículos. Para um carro estacionar em uma vaga já ocupada, é necessário retirar o carro que está ocupando a vaga para podermos colocar o outro carro.

No caso das variáveis é a mesma coisa! Se fizermos uma auto atribuição (atribuir a uma variável o valor já presente nela realizado uma operação qualquer com outro número, a variável deixa o valor anterior e passa a assumir o novo resultado.

  
>> Repare que demonstro o valor de n após cada cálculo e ele se altera a cada nova auto atribuição

* + 1. **Auto Atribuições (forma encurtada)**

|  |  |
| --- | --- |
| var n = 3 |  |
| n = n + 4 | n += 4 |
| n = n – 5 | n -= 5 |
| n = n \* 4 | n \*= 4 |
| n = n / 2 | n /= 2 |
| n = n \*\* 2 | n \*\*= 2 |
| n = n % 5 | n %= 2 |

>> Apenas funciona caso seja um cálculo de um valor já atribuído a ele mesmo

Não é apenas o JavaScript que utiliza esta forma de auto atribuição, Java, C, Python, php, por exemplo, também aceitam essa forma de auto atribuição.