

Objetivo:

- I. Linguagem de programação;
- II. Ambiente de desenvolvimento integrado – IDE
- III. Projeto JavaScript
- IV. Principais tipos de dados;
- V. Estrutura de uma função;
- VI. Variável;
- VII. Comentários.

I. Linguagem de programação

Qualquer tipo de programa de computador, aplicativo de celular ou página de internet foi construída utilizando alguma linguagem de programação. Logo, é impossível pensar em computação sem considerar as linguagens de programação.

Existem várias linguagens de programação, cada uma com suas características próprias. Atualmente as principais linguagens de programação são (<https://survey.stackoverflow.co/2022/#section-most-popular-technologies-programming-scripting-and-markup-languages>):

1. JavaScript: é uma das principais linguagens de programação na internet. Todos os navegadores suportam JavaScript. Atualmente, ela é usada para desenvolvimento de aplicações web (front e back-end) e mobile, por este motivo ela é a linguagem de programação mais utilizada;
2. Python: é uma linguagem de programação considerada de fácil aprendizado e possui grande inserção em comunidades de não-desenvolvedores, assim como engenheiros e cientistas. Por ter a capacidade de processar grandes volumes de dados, acabou sendo adotada na computação científica, ciência de dados e inteligência artificial;
3. Java: é uma linguagem robusta que pode ser utilizada para o desenvolvimento de grandes aplicações na internet e desktop.

Um programa é uma sequência de instruções (comandos) escritos seguindo uma lógica. De forma paralela, as instruções são como palavras em um texto que só fazem sentido se forem colocadas numa sequência lógica.

Um programa é uma sequência de instruções (comandos) escritas usando uma linguagem de programação de alto nível (JavaScript, Python, Java etc.) também chamado de código fonte. O programa é executado pelo processador, que por sua vez, só entende a sua própria linguagem de programação, que é uma linguagem muito difícil de ser

programada (linguagem de baixo nível também chamada de linguagem de máquina). Para fazer a conversão de código fonte para código de máquina precisamos fazer um processo chamado de compilação ou interpretação. Em outras palavras, escreveremos o nosso programa em JavaScript e ele precisa ser interpretado, por um interpretador, que por sua vez traduzirá ele para a linguagem de máquina do computador ou dispositivo. Todos os navegadores possuem um ambiente de linha de comando que suporta instruções JavaScript, pressione a tecla <F12> para abrir o console do navegador, assim como é mostrado na Figura 1.

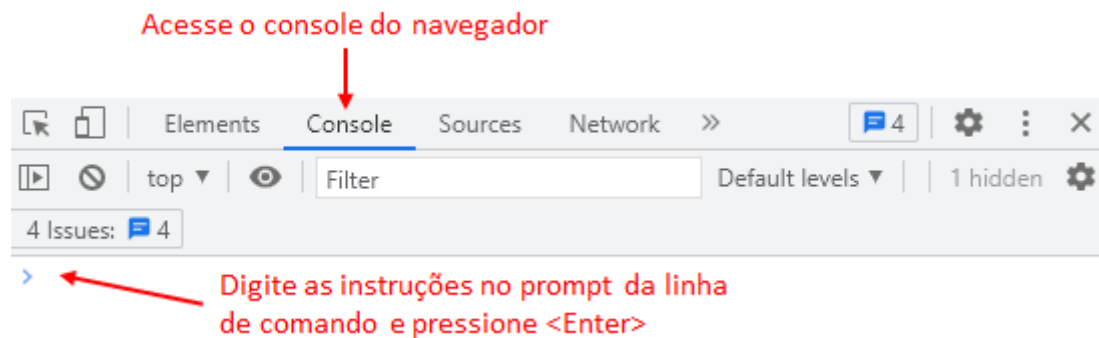


Figura 1 – Console do Google Chrome.

II. Ambiente de desenvolvimento integrado - IDE

Para escrever um programa usamos um editor de textos para digitar as instruções. Esse editor de textos é chamado de IDE (Integrated Development Environment, em português, Ambiente de Desenvolvimento Integrado). O termo integrado vem do fato dele: apontar os nossos erros de digitação e chamar o compilador ou interpretador da linguagem de programação quando pedimos para executar o programa.

Existem vários IDEs que podem ser instalados no seu computador ou usadas na internet. O IDE online Replit (<https://replit.com>) suporta várias linguagens de programação, para rodar JavaScript precisamos do runtime (em português, tempo de execução) Node.JS.

No Replit os projetos são chamados de Repl, a Figura 2 mostra a tela para criar um projeto. Observação: antes de criar um projeto é bom estar logado para que os seus Repls fiquem salvos na nuvem e poderão ser acessados de qualquer computador, ou seja, recomenda-se criar um conta no Replit antes de fazer os passos da Figura 2.

Após criar o seu Repl (projeto no IDE Replit), podemos digitar o nosso código no arquivo index.js, assim como é mostrado na Figura 3.

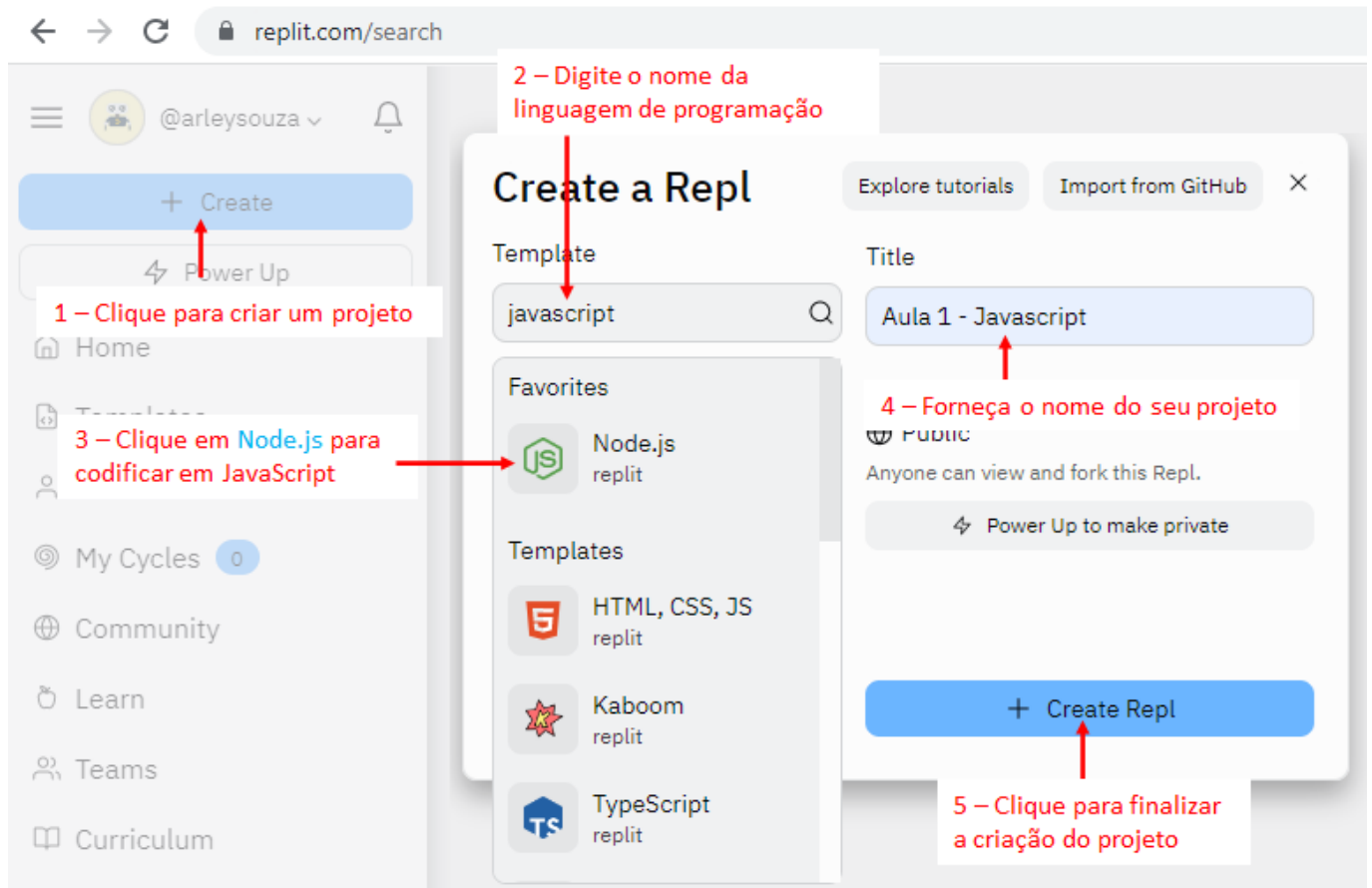


Figura 2 – Criar um projeto para escrever programa JavaScript no IDE online Replit.

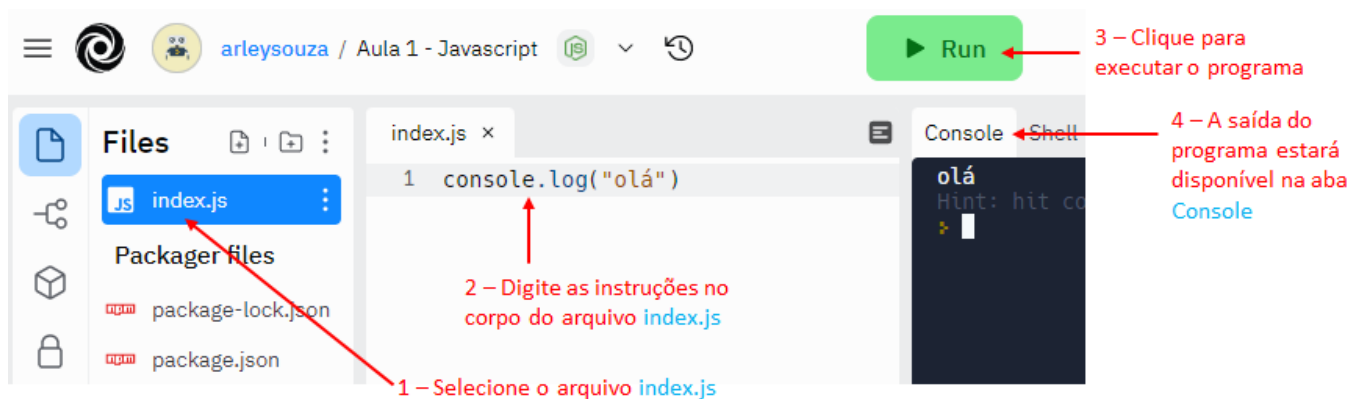


Figura 3 – Digitar e executar programas no IDE Replit.

No curso utilizaremos o IDE Visual Studio Code da Microsoft, também conhecido apenas por VS Code (<https://code.visualstudio.com>). Ele é o IDE mais utilizado pelos desenvolvedores web e mobile, talvez o grande sucesso do VS Code esteja no fato dele ser apenas uma interface para digitarmos os nossos programas (código fonte), para interpretar e executar o código fonte em JavaScript precisaremos do runtime (em português, tempo de execução) Node.JS (<https://nodejs.org/en>).

Como instalar o Node.JS <https://www.youtube.com/watch?v=-cLzUDOTQY0>

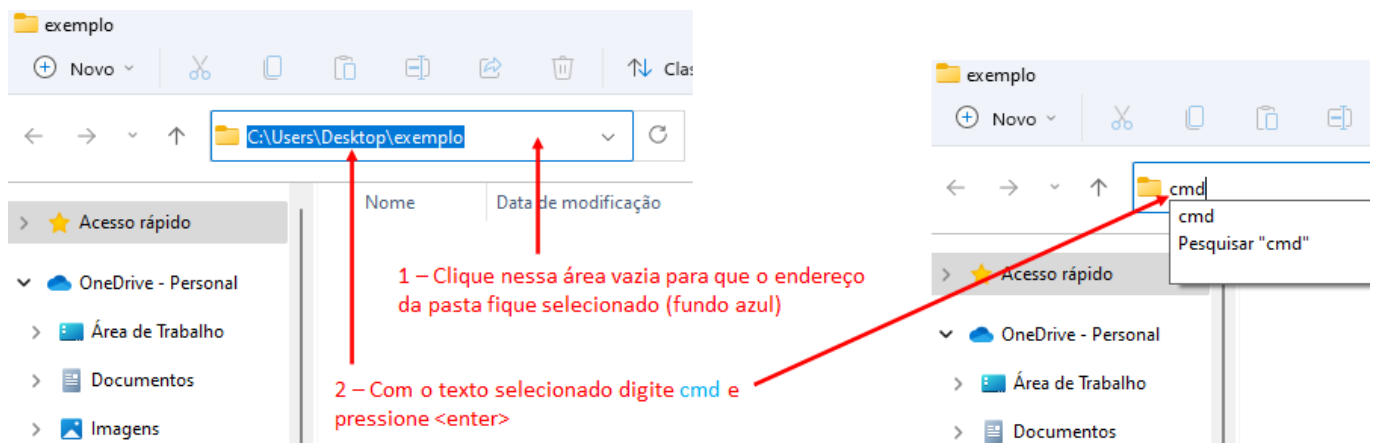
Como instalar o VS Code <https://www.youtube.com/watch?v=R6YslWRUFk>

III. Projeto JavaScript

Um projeto é formado pelos arquivos com as instruções que nós programamos e os arquivos de configuração para executar o nosso programa.

Siga os passos para criar um projeto JavaScript:

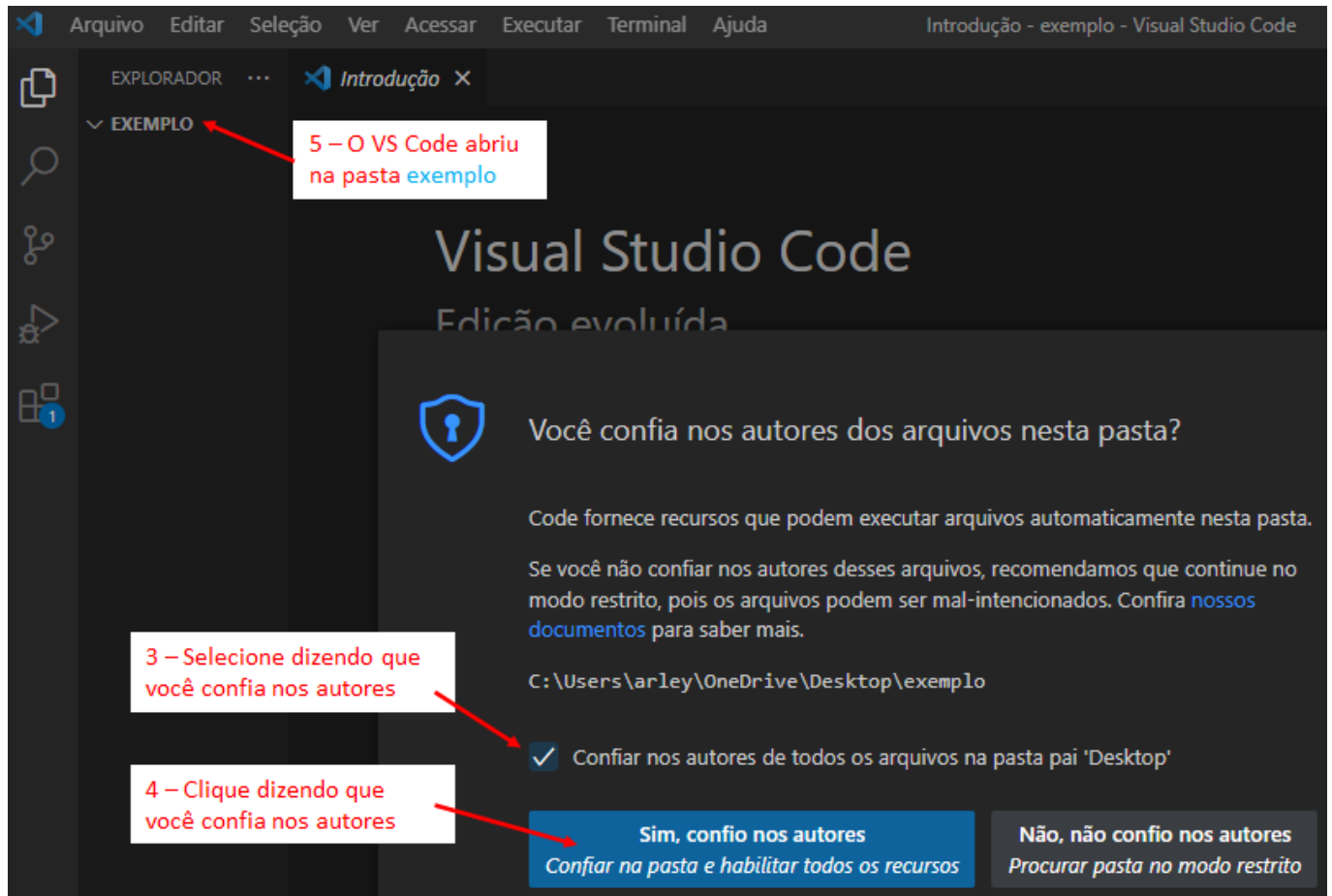
1. Crie uma pasta de nome **exemplo** em qualquer local do seu computador, por exemplo, na área de trabalho. Aqui foi sugerido o nome **exemplo**, mas poderia ser qualquer nome de pasta sem espaços e caracteres especiais (acentos e cedilha);
2. O prompt de comando do Windows (cmd) é um ambiente que nos permite executar os programas sem o recurso do clique do ambiente janelado do Windows. A seguir estão os passos para abrir o CMD (prompt de comando do Windows) na pasta **exemplo**:



3. Veja que o prompt de comando abriu na pasta **exemplo**. Digite code espaço ponto (**code .**) e pressione <Enter> para abrir o VS Code na pasta **exemplo**:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [versão 10.0.22000.1335]
(c) Microsoft Corporation. Todos os direitos reservados.
C:\Users\Desktop\exemplo>code .
```

4. O VS Code será aberto na pasta **exemplo**. Existem outras formas de abrir o VS Code numa pasta, porém esta é a sequência adotada pelos desenvolvedores da comunidade JavaScript:

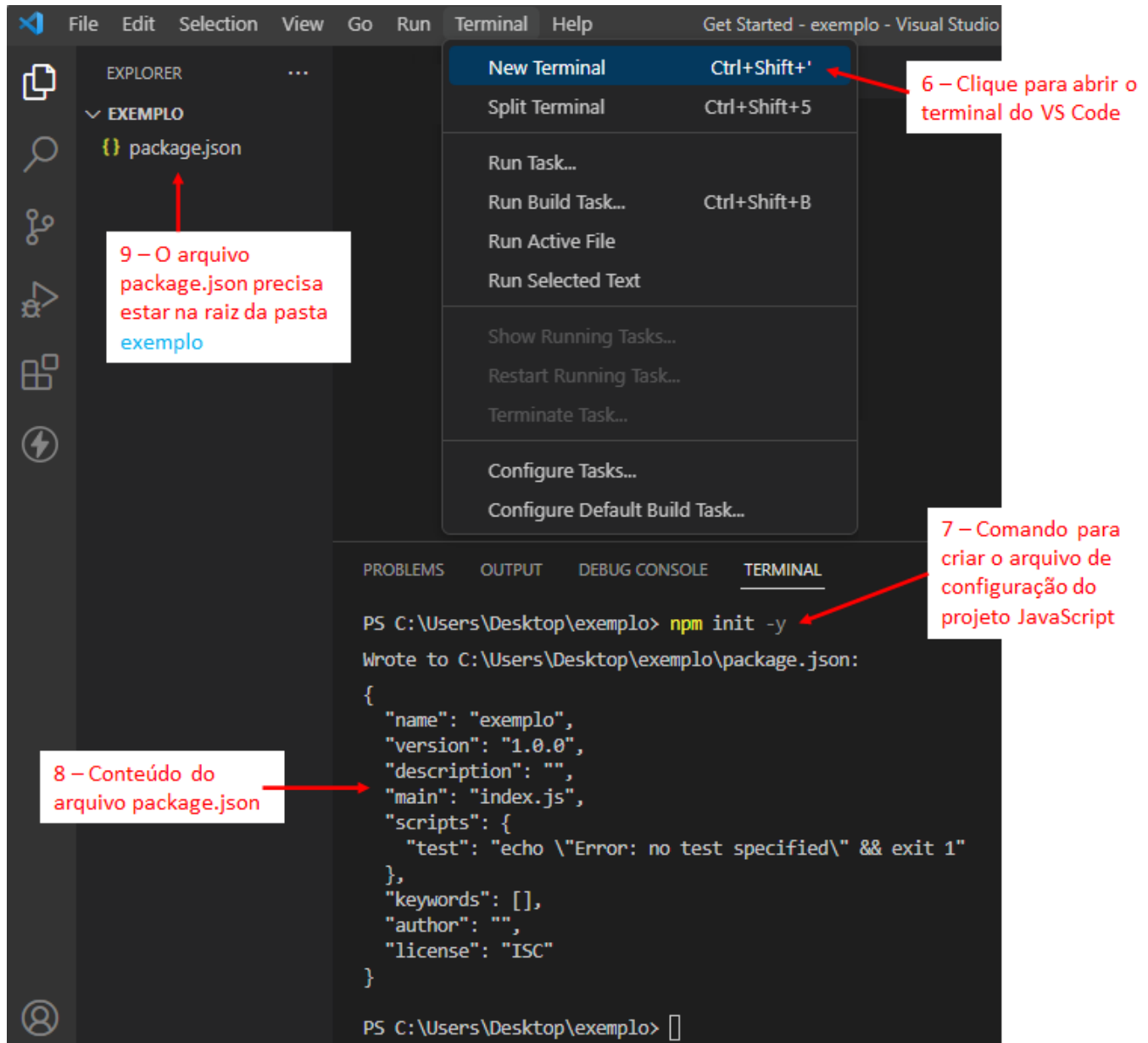


- O terminal do VS Code é um ambiente para digitarmos comandos, assim como no CMD. Siga os passos da figura a seguir para abrir o terminal do VS Code.

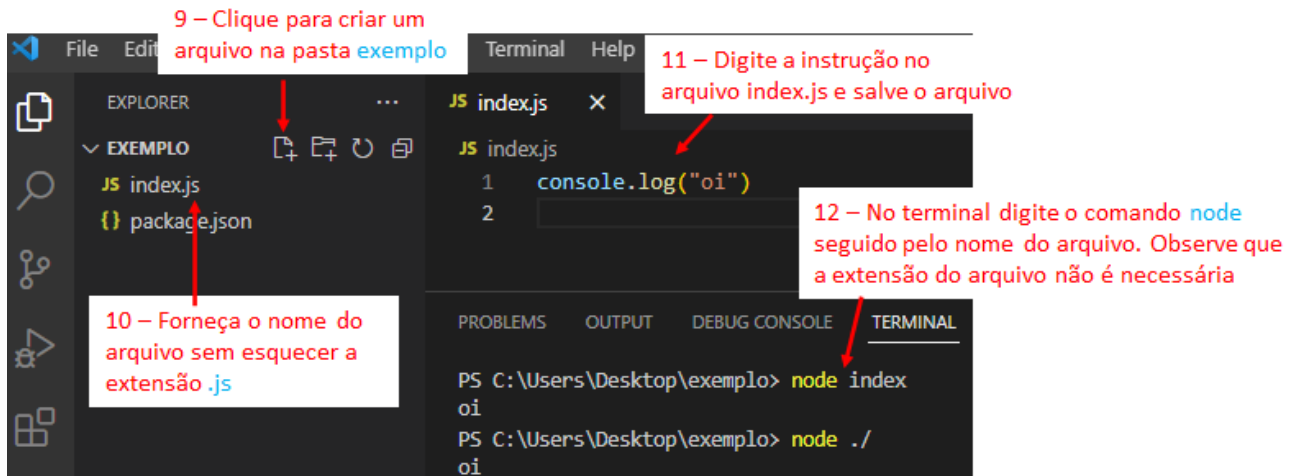
O comando `npm init` é usado para criar o arquivo `package.json` de configuração do projeto Node. O parâmetro `-y` é usado para responder sim (yes) para todas as perguntas. As perguntas são name, version, description, ...

O comando `npm` é um programa instalado com o Node.js, se você não conseguiu criar o arquivo `package.json` isso aconteceu pelo fato de o Node.js não ter sido instalado. Digite os comandos `node -v` e `npm -v` para verificar as versões dos softwares instalados. No caso de não aparecer algum número de versão, então você precisará instalar o Node.js.

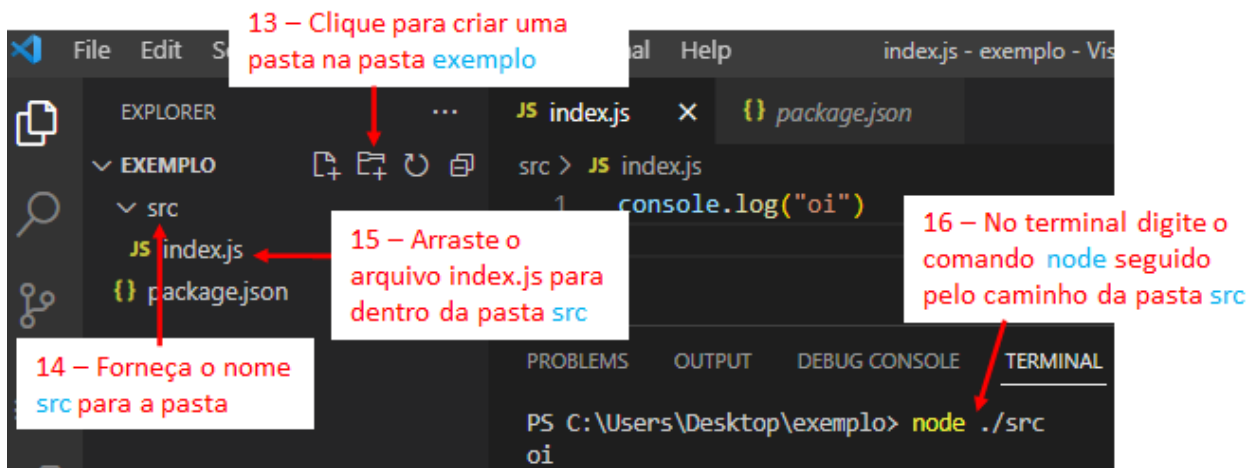
```
C:\Windows\System32\cmd.exe
C:\Users\Desktop\exemplo>node -v
v16.14.2
C:\Users\Desktop\exemplo>npm -v
8.5.0
```



- Após criar o arquivo package.json de configuração do projeto Node podemos criar os nossos arquivos de código. Crie o arquivo `index.js` na pasta `exemplo`. O nome do arquivo não é relevante, porém qualquer arquivo de nome `index.js` poderá ser executado endereçando apenas a pasta na qual ele se encontra. Como exemplo, `./` está endereçando a pasta atual, desta forma, o comando `node ./` executará o arquivo `index.js` que está dentro da pasta atual. A pasta atual neste exemplo é `C:\Users\Desktop\exemplo`.



7. Os programadores JavaScript geralmente gostam de colocar os arquivos de programa dentro de uma pasta de nome `src`, abreviação de source (fonte em português). Siga os passos da figura para criar a pasta `src` e mover o arquivo `index.js` para dentro da pasta `src`;



8. O comando `node ./src` pode ser colocado na propriedade `scripts` do arquivo `package.json`. Este recurso não faz diferença, porém, é muito usado pelos programadores JavaScript na organização do projeto.



IV. Principais tipos de dados

Na programação cada valor possui um tipo de dado, os principais tipos são:

- **string:** o tipo de dado string é usado para representar valores no formato de texto. Um texto precisa estar entre aspas simples ou dupla. Exemplos:

`"chuva"`, `'domingo'`, `"12.34"` e `"false"`

Ao colocar o valor entre aspas ele deixa de ser número ou booleano e passa a ser string;

- **number:** o tipo de dado number é usado para representar números inteiros e reais. Exemplos: -8 e 12.23. Observe que o ponto é o separador decimal;
- **boolean:** o tipo de dado booleano é usado para representar os valores `true` e `false` somente. Esses valores não podem estar entre aspas, se colocarmos aspas eles passam a ser string.

Cada tipo de dado ocupa uma certa quantidade de bits e possui uma organização na memória do computador. Os tipos de dados são utilizados para otimizar o uso da memória, pois na prática, tudo são dados nos dispositivos de armazenamento. Como exemplo, até mesmo um vírus de computador é um conjunto de instruções que ocupa um espaço no dispositivo de armazenamento.

V. Estrutura de uma função

Uma função é uma instrução usada para executar um conjunto de instruções. Para usar uma função precisamos colocar o nome da função seguido por um par de parênteses. A seguir tem-se três funções:

- `console.log()` neste exemplo `console.log` é o nome da função;
- `prompt()` neste exemplo `prompt` é o nome da função;
- `parseInt()` neste exemplo `parseInt` é o nome da função.

As boas práticas dizem que a 1ª letra do nome da função é sempre minúscula e só existe letra maiúscula onde deveria existir um espaço. Como exemplo, `parse int` são duas palavras, como não pode haver espaço no nome da função, então retira-se o espaço e capitaliza a letra seguinte, por este motivo escreve-se `parseInt`.

Cada função recebe uma certa quantidade de parâmetros, temos de consultar o help para sabermos quais são os parâmetros de uma função. Os parâmetros são valores passados dentro dos parênteses. Exemplos:

- `console.log("olá")` a função `console.log` pode receber um parâmetro do tipo de dado string, number ou boolean. A funcionalidade da função `console.log` é imprimir no console o valor passado como parâmetro. A Figura 4 mostra o uso da função, veja que os valores passados como parâmetro são exibidos no console;

- `prompt("Entre com o nome:")` a função `prompt` recebe um texto como parâmetro. Ela imprime na tela este texto e fica esperando o usuário digitar algo e pressionar <Enter>. A Figura 5 mostra o uso da função;
- `parseInt("2")` a função recebe um texto como parâmetro – dentro desse texto precisa ter um número inteiro – e retorna o número inteiro (Figura 6);
- `parseFloat("1.5")` a função recebe um texto como parâmetro – dentro desse texto precisa ter um número real – e retorna o número real (Figura 6);

No exemplo da Figura 6 a função `console.log` receberá como parâmetro o resultado da operação matemática $2 + 1.5$.

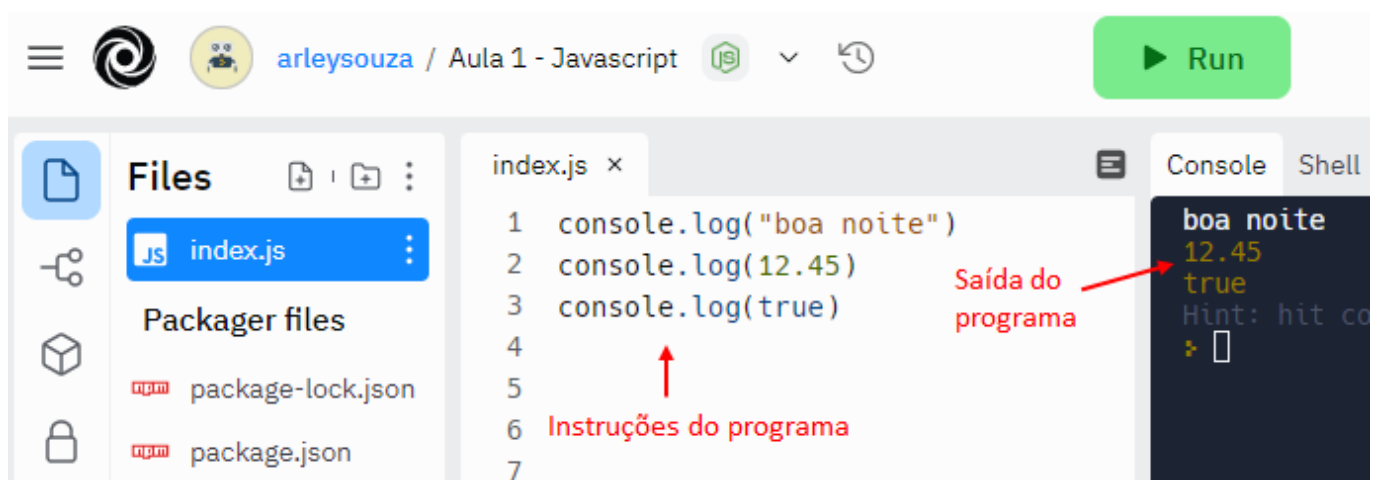


Figura 4 – Exemplo de uso da função `console.log`.

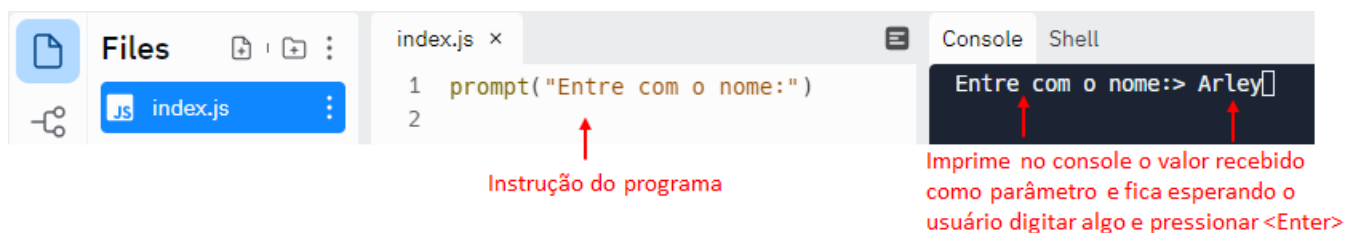


Figura 5 – Exemplo de uso da função `prompt`.

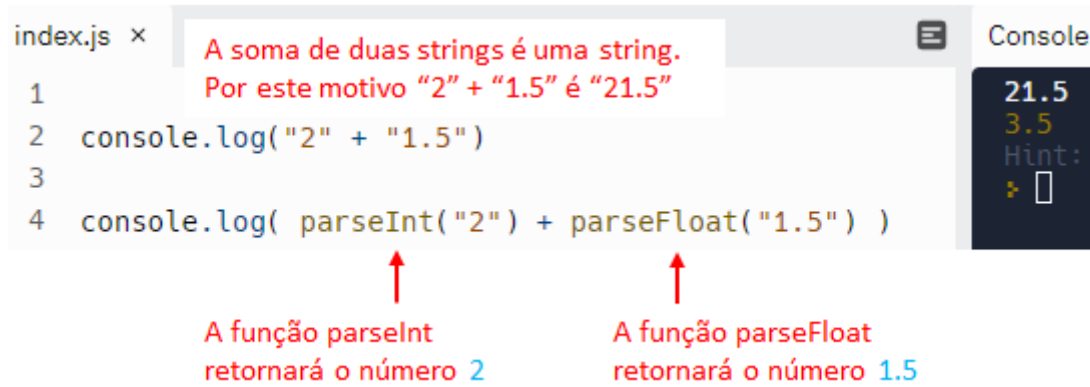


Figura 6 – Exemplo de uso das funções parseInt e parseFloat.

VI. Variável

É um espaço na memória do computador. Uma variável é formada por:

- nome: identificador do espaço de memória;
- conteúdo: valor guardado no espaço de memória;
- tipo de dado: tipo de dado do valor guardado no espaço de memória.

Notação para criar variáveis:

- `dia = "domingo"`
 - objetivo: criar um espaço na memória e dar o nome de `dia` a ele. A vantagem de dar um nome para um espaço de memória é poder chamar ele posteriormente;
 - `dia` é o nome do espaço na memória;
 - `"domingo"` é o conteúdo da variável;
 - `string` é o tipo de dado do conteúdo da variável.
- `idade = 21`
 - `idade` é o nome do espaço na memória;
 - `21` é o conteúdo da variável;
 - `number` é o tipo de dado do conteúdo da variável.
- `peso = 58.9`
 - `peso` é o nome do espaço na memória;
 - `58.9` é o conteúdo da variável;
 - `number` é o tipo de dado do conteúdo da variável.
- `doador = true`
 - `doador` é o nome do espaço na memória;
 - `true` é o conteúdo da variável;
 - `boolean` é o tipo de dado do conteúdo da variável.

A Figura 7 ilustra as variáveis na memória de um computador. O tipo de dado é usado internamente pela linguagem de programação para dimensionar o tamanho do espaço de memória. Nesta ilustração as variáveis

estão em locais aleatórios, na verdade, não temos como controlar o local onde as variáveis são colocadas na memória, pois o local caberá ao sistema operacional do computador.

As variáveis são usadas para armazenar dados durante a execução do programa, no exemplo da Figura 8 as variáveis `base`, `altura`, `b`, `a` e `area` (sem acento) são usadas para armazenar os conteúdos durante a execução do programa. Cada linha desse programa possui uma instrução e os conteúdos são perdidos de uma instrução para outra se eles não forem guardados numa variável.

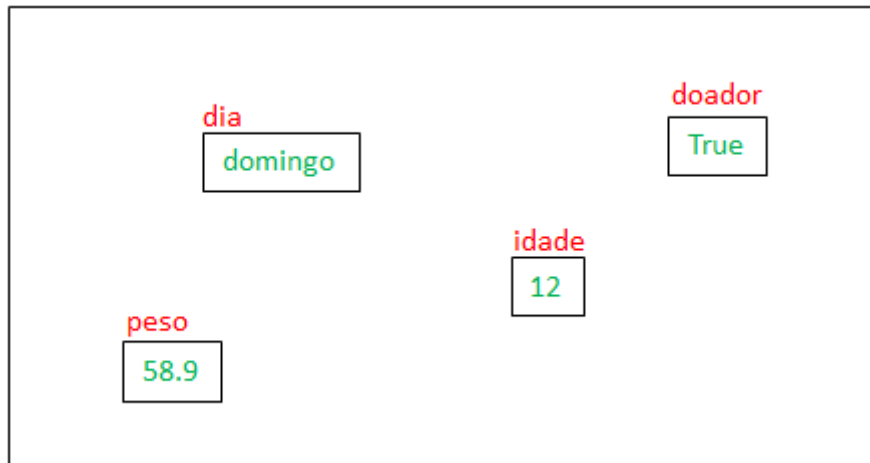


Figura 7 – Representação das variáveis na memória do computador.

A função `prompt` retorna o valor digitado pelo usuário. Para que esse valor não seja perdido, teremos de guardar ele numa variável. Aqui usamos o nome de variável `base`. A função `prompt` sempre retorna um texto, então a variável `base` terá a string `"2.5"`

A função `prompt` imprime no console o texto recebido como parâmetro e fica esperando o usuário digitar algo e pressionar `<Enter>`. Neste exemplo, o usuário digitou `2.5`

Uma variável armazena somente um conteúdo, por este motivo tivemos de criar a variável `altura` para guarda a outra entrada do usuário

A variável `area` recebe o resultado da multiplicação de `2.5 x 4`

A função `console.log` recebe como parâmetro o conteúdo da variável `area` e imprime ele no console

A função `parseFloat` recebe como entrada o conteúdo da variável `base`, ou seja, neste exemplo a função receberá o texto `"2.5"` e retornará o number `2.5`. O number `2.5` será guardado na variável `b`

```

index.js x
base = prompt("Entre com a base:")
altura = prompt("Entre com a altura:")
b = parseFloat(base)
a = parseFloat(altura)
area = b * a
console.log(area)
    
```

Console Shell

Entre com a base:> 2.5

Entre com a altura:> 4

10

Hint: hit control+c any

Figura 8 – Exemplo de uso de variáveis para guardar valores durante a execução do programa.

Uma variável será acessada para escrita se o nome da variável estiver à esquerda do símbolo de atribuição (representado no código pelo símbolo de `=`). Em todas as demais situações a variável será acessada para leitura.

Uma variável é um espaço na memória que pode ser acessado para leitura ou escrita. Na instrução `idade = 21` o espaço de memória nomeado por `idade` está sendo acessado para escrita. Na instrução `nova = idade + 1` o espaço de memória nomeado por `idade` está sendo acessado para leitura.

VII. Comentários

Os comentários são textos colocados no código com o intuito de documentar o programa para os programadores. Os textos adicionados à direita da `//` e entre os pares de `/* */` na Figura 9 serão ignorados pelo interpretador do JavaScript, ou seja, eles não serão executados pelo computador.

No JavaScript existem duas formas de inserir comentários no código:

- `//` tudo que estiver à direita das duas barras será ignorado. Observação, não pode haver espaço entre as duas barras;
- `/*` tudo que estiver entre a abertura e fechamento da barra e asterisco será ignorado `*/`. O par de barra e asterisco é usada para comentar mais de uma linha do código. Observação, constitui erro colocar somente a barra e asterisco de abertura.

```
/* a variável base recebe
   a string retornada pela função prompt */
base = prompt("Entre com a base:")
// recebe a altura do retângulo
altura = prompt("Entre com a altura:")
// a variável b recebe o valor convertido para number
b = parseFloat(base)
// a variável a recebe o valor convertido para number
a = parseFloat(altura)
// a variável area recebe o resultado da multiplicação
area = b * a
/* a função console.log imprime no console
   o conteúdo da variável area */
console.log(area)
```

Figura 9 – Comentários no programa.

Exercícios

Veja o vídeo se tiver dúvidas nos exercícios: https://youtu.be/ch_KohZ9Ql0

Para fazer os exercícios 1 a 6 crie um projeto de nome `aula1` no VS Code assim como é mostrado a seguir. Cada programa deverá estar num arquivo separado da pasta `src`. Crie uma propriedade para cada exercício na propriedade `scripts` do arquivo `package.json`. Para rodar o arquivo use: `npm run` propriedade, onde propriedade será um, dois, tres (sem acento), quatro, cinco ou seis.

```

package.json
{
  "name": "aula1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "um": "node ./src/um",
    "dois": "node ./src/dois",
    "tres": "node ./src/tres",
    "quatro": "node ./src/quatro",
    "cinco": "node ./src/cinco",
    "seis": "node ./src/seis"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

```

```

PS D:\Algoritmos\aula1> npm run um

> aula1@1.0.0 um
> node ./src/um

Boa noite!

```

Um programa se caracteriza por entrada de dados, processamento (transformação de dados) e saída de dados. Nos exercícios 1 a 3 trabalharemos apenas a saída de dados.

Exercício 1: Fazer um programa que imprime no console o texto **Boa noite!**.

Dica: use a função `console.log`.

Exemplo de saída:

Boa noite!

Exercício 2: Fazer um programa que imprime no console o número inteiro **100**.

Dica: use a função `console.log`.

Exemplo de saída:

100

Exercício 3: Fazer um programa que imprime no console o valor booleano

Exemplo de saída:

true.



Dica: use a função console.log.

Nos exercícios 4 a 6 trabalharemos com declaração de variáveis e saída de dados.

Exercício 4: Fazer um programa com as variáveis x e y. A variável x deverá ser iniciada com o valor 10 e variável y com o valor 20. Na sequência o seu programa deverá imprimir no console o resultado da soma das variáveis x e y.

Dica: use a função console.log para imprimir o resultado.

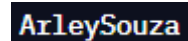
Exemplo de saída:



Exercício 5: Fazer um programa com as variáveis nome e sobrenome. A variável nome deverá ser iniciada com o seu 1º nome e variável sobrenome com o seu último sobrenome. Na sequência o seu programa deverá imprimir no console o resultado da soma dos conteúdos das variáveis nome e sobrenome.

Dica: use a função console.log para imprimir o resultado.

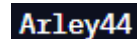
Exemplo de saída:



Exercício 6: Fazer um programa com as variáveis nome e idade. A variável nome deverá ser iniciada com o seu 1º nome e variável idade com a sua idade. Na sequência o seu programa deverá imprimir no console o resultado da soma das variáveis nome e idade. Lembre-se que o nome é string e idade é number.

Dica: use a função console.log para imprimir o resultado.

Exemplo de saída:



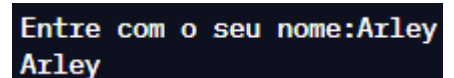
Nos exercícios 7 a 11 trabalharemos com declaração de variáveis, entrada de dados e saída de dados.

Observação: utilize o ambiente <https://replit.com> para fazer os programas que requerem entrada do usuário.

Exercício 7: Fazer um programa que pede para o usuário entrar com o seu 1º nome e na sequência o programa imprime no console o nome fornecido pelo usuário.

Dica: use a função prompt para ler o teclado e console.log para imprimir o resultado.

Exemplo de saída:



Exercício 8: Fazer um programa que pede para o usuário entrar com a idade e na sequência o programa imprime o dobro da idade fornecida pelo usuário.

Dica: use a função `prompt` para ler o teclado, a função `parseInt` para converter de string para number, e a função `console.log` para imprimir o resultado.

Exemplo de saída:

```
Entre com a sua idade:18
36
```

Exercício 9: Fazer um programa que pede para o usuário entrar com o peso e na sequência o programa imprime a metade do peso fornecido pelo usuário.

Dica: use a função `prompt` para ler o teclado, a função `parseFloat` para converter de string para number, e a função `console.log` para imprimir o resultado.

Exemplo de saída:

```
Entre com o seu peso:59
29.5
```

Exercício 10: O IMC (Índice de Massa Corporal) é calculado usando $\text{peso}/(\text{altura} \times \text{altura})$. Fazer um programa que pede para o usuário entrar com o peso (em Kg) e altura (em metros) e na sequência o programa imprime no console o IMC.

Dica: use a função `prompt` para ler o teclado, a função `parseFloat` para converter de string para number, e a função `console.log` para imprimir o resultado.

Exemplo de saída:

```
Entre com o peso (Kg):71.5
Entre com a altura (m):1.72
24.168469442942133
```

Exercício 11: Fazer um programa que pede para o usuário entrar com três números reais. Sendo um número de cada vez. Após fornecer os três números o programa deve imprimir na tela o valor médio.

Dica: use a função `prompt` para ler o teclado, a função `parseFloat` para converter de string para number, e a função `console.log` para imprimir o resultado.

Exemplo de saída:

```
Entre com o 1o número:8.5
Entre com o 2o número:1.5
Entre com o 3o número:6.5
5.5
```