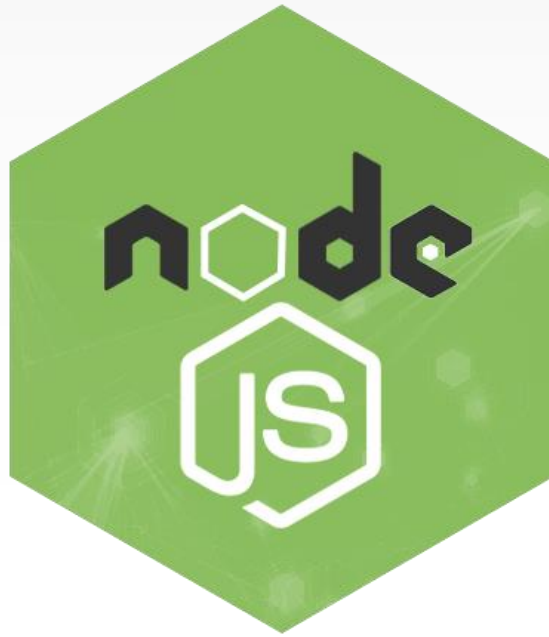


JavaScript

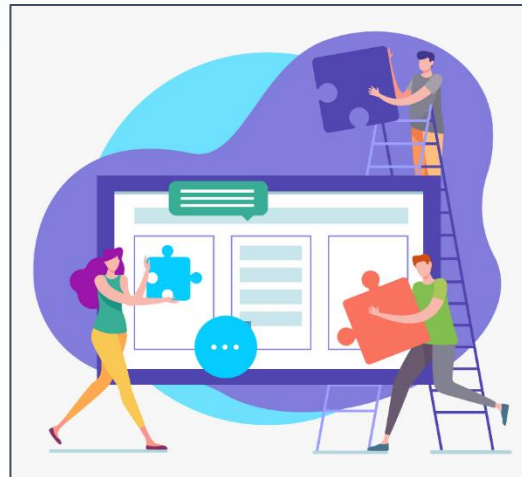
Módulo Básico





NODE.JS

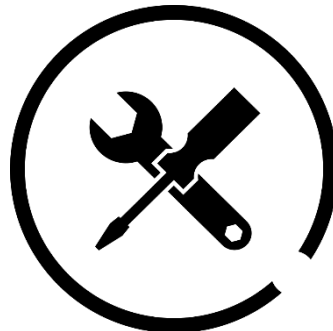
- **Node.js**
 - **Algoritmos em JS realizam a *interação* com o usuário e duas formas:**
 - **a partir de caixas de diálogo exibidas no browser**
 - **a partir de páginas web, onde o usuário preenche alguns campos de um formulário e obtém uma resposta exibida na própria página**



- **Node.js**
 - *Existem também outro modo de construir algoritmos com JS, que é utilizando o **Node.js***
 - *Nesse modo, semelhante ao que acontece com linguagens de programação tradicionais como **Python**, **Java** e **C#**, é necessário instalar a linguagem na máquina e executar os programas a partir de um **prompt** de linha de comandos*
 - **Node.js** é um ambiente de execução de programas JS gratuito, de **código aberto** e **multiplataforma** que permite aos desenvolvedores escrever programas de linhas de comando e scripts do **lado** do **servidor** fora de um navegador

Instalação

- *Instalação do Node.js*
 - Para instalar o **Node.js** acesse o site do ambiente (**nodejs.dev**) e selecione a opção **download**
 - Escolha a opção **LTS Recommended for most users** e baixe o instalador de acordo com o SO de sua máquina
 - Posterior a realização do download, **execute-o** conforme o **SO**



- *Instalação do Node.js*

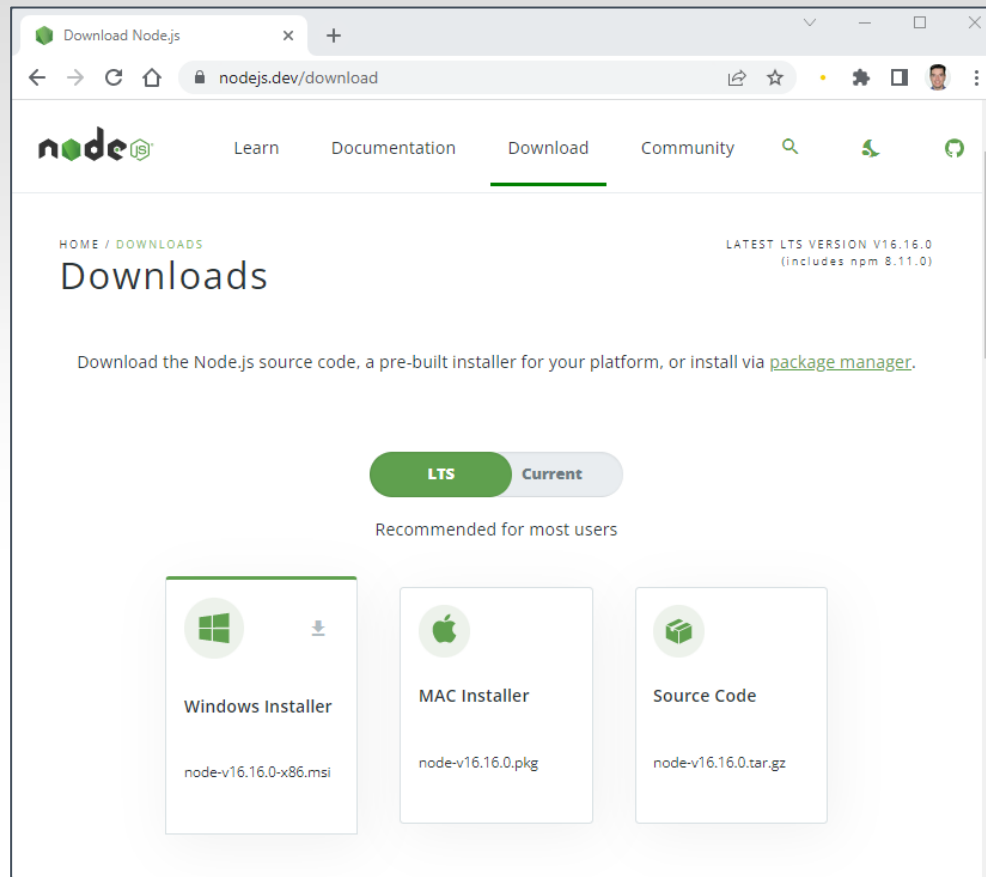
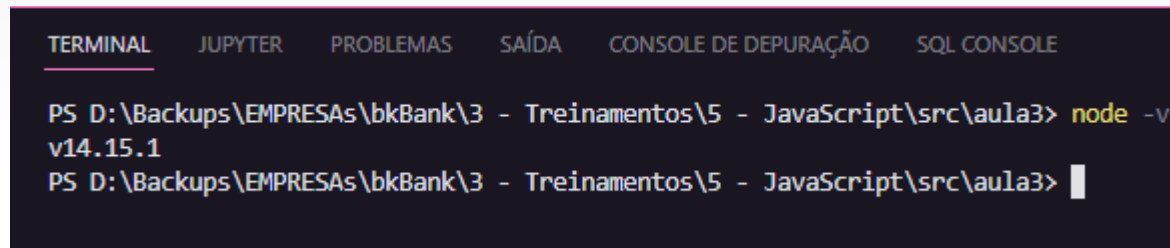


Figura 1 – Opções de download do Node.js

Instalação

- *Instalação do Node.js*
 - *Concluída a instalação, acesse o prompt de comandos e execute o comando **node -v***
 - *Exibe a versão do Node.js instalada, indicando que o processo de instalação foi realizado com sucesso*



```
TERMINAL  JUPYTER  PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  SQL CONSOLE

PS D:\Backups\EMPRESAS\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3> node -v
v14.15.1
PS D:\Backups\EMPRESAS\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3> |
```

Figura 2 – Versão e êxito da instalação

Input de Dados

- ***Pacote para Input de Dados***
 - ***Adicionar um pacote que permitirá aos programas realizar a **entrada** de **dados** via linha de comando, por meio do **prompt-sync()*****
 - ***Existem diversos outros, mas esse contém os recursos básicos necessários para o desenvolvimento***
 - ***Na sequência, criaremos uma pasta via linha de comandos***

```
D:\JavaScript\src\aula3> npm i prompt-sync
```

```
D:\JavaScript\src\aula3> md js
```

```
D:\JavaScript\src\aula3> cd js
```


Input de Dados

- *Pacote para Input de Dados*
 - *Foi adicionado ao diretório (pasta) **aula3** a pasta **node-modules***
 - ***node-modules** contém os pacotes que foram adicionados ao projeto*
 - *Também foram criados arquivos **.json***
 - *Os programas a serem desenvolvidos com o Node.js (utilizam o prompt-sync) precisam estar em pastas que ficam dentro de **aula3** (pacote prompt-sync foi instalado)*



- ***Criação e execução de programas com o Node.js***
 - ***Exemplo (01):***
 - ***Soma de 2 números***

```
js > JS exemplo_1.js > ...
1  const prompt = require("prompt-sync")()      // adiciona pacote para entrada de dados
2  const num1 = Number(prompt("1º Número: "))    // lê os números
3  const num2 = Number(prompt("2º Número: "))
4
5  const soma = num1 + num2                      // calcula a soma
6  console.log(`Soma é: ${soma}`)               // exibe o resultado
```

TERMINAL JUPYTER PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO SQL CONSOLE

```
PS D:\Backups\EMPRESAs\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3\js> node .\exemplo_1
1º Número: 10
2º Número: 5
Soma é: 15
PS D:\Backups\EMPRESAs\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3\js> 
```

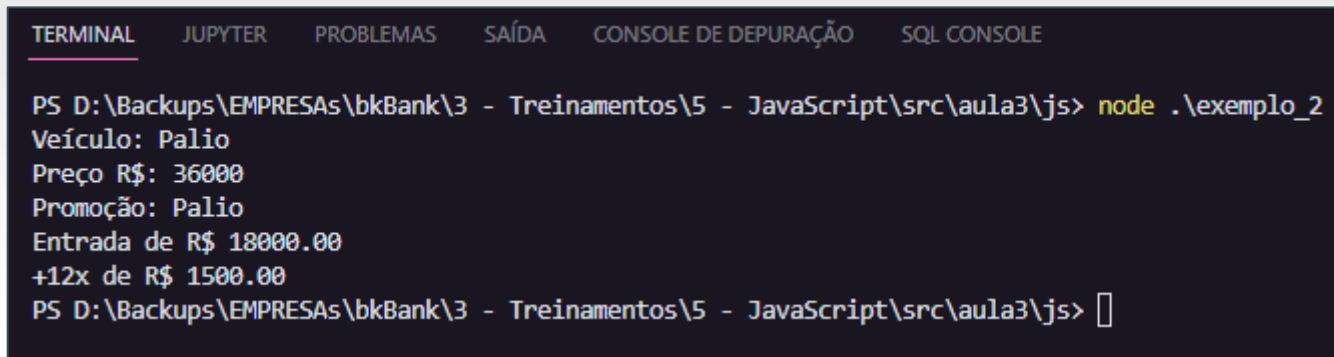
Figura 3 – Criação e execução do exemplo_1.js

- *Criação e execução de programas com o Node.js*
 - *Exemplo (02):*
 - *Revenda de veículos*

```
js > JS exemplo_2.js > ...
1  const prompt = require("prompt-sync")() // adiciona o pacote ao programa
2  const veiculo = prompt("Veículo: ")     // lê os dados de entrada
3  const preco = Number(prompt("Preço R$: "))
4
5  const entrada = preco * 0.50             // calcula o valor da entrada
6  const parcela = (preco * 0.50) / 12     // ... e das parcelas
7
8  console.log(`Promoção: ${veiculo}`)     // exibe as respostas
9  console.log(`Entrada de R$ ${entrada.toFixed(2)}`)
10 console.log(`+12x de R$ ${parcela.toFixed(2)}`)
```

Figura 4 – Criação do exemplo_2.js

- *Criação e execução de programas com o Node.js*
 - *Exemplo (02):*
 - *Revenda de veículos*

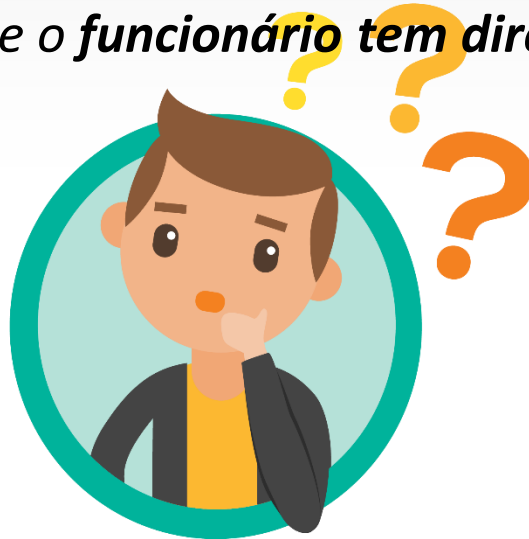


```
TERMINAL  JUPYTER  PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  SQL CONSOLE

PS D:\Backups\EMPRESAs\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3\js> node .\exemplo_2
Veículo: Palio
Preço R$: 36000
Promoção: Palio
Entrada de R$ 18000.00
+12x de R$ 1500.00
PS D:\Backups\EMPRESAs\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3\js> 
```

Figura 5 – Execução do exemplo_2.js

- ***Criação e execução de programas com o Node.js***
 - ***Exemplo (03):***
 - ***Elabore um programa para uma empresa que leia o salário e o tempo que um funcionário trabalha na empresa***
 - ***Sabendo que a cada 4 anos (quadriênio) o funcionário recebe um acréscimo de 1% no salário, calcule e informe o número de quadriênios a que o funcionário tem direito e o salário final***

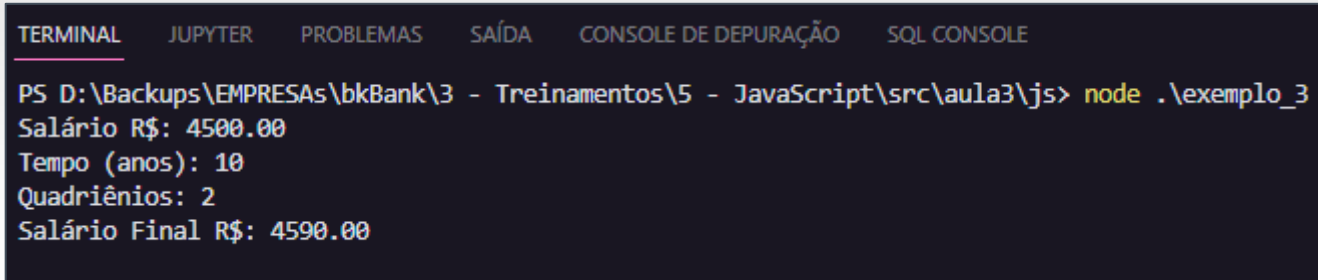


- *Criação e execução de programas com o Node.js*
 - *Exemplo (03):*
 - *Calculando os quadriênios*

```
js > JS exemplo_3.js > ...
1  const prompt = require("prompt-sync")()           // adiciona o pacote ao programa
2  const salario = Number(prompt("Salário R$: "))     // lê os dados de entrada
3  const tempo = Number(prompt("Tempo (anos): "))
4  const quadrienios = Math.floor(tempo / 4)          // calcular quadriênios
5  const acrescimo = salario * quadrienios / 100      // .. e acréscimo
6
7  console.log(`Quadriênios: ${quadrienios}`)         // exibe as respostas
8  console.log(`Salário Final R$: ${(salario + acrescimo).toFixed(2)}`)
```

Figura 6 – Criação do exemplo_3.js

- *Criação e execução de programas com o Node.js*
 - *Exemplo (03):*
 - *Calculando os quadriênios*



The image shows a terminal window with a dark background. At the top, there are several tabs: 'TERMINAL' (highlighted with a red underline), 'JUPYTER', 'PROBLEMAS', 'SAÍDA', 'CONSOLE DE DEPURAÇÃO', and 'SQL CONSOLE'. Below the tabs, the terminal displays the following text:

```
PS D:\Backups\EMPRESAS\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3\js> node .\exemplo_3
Salário R$: 4500.00
Tempo (anos): 10
Quadriênios: 2
Salário Final R$: 4590.00
```

Figura 7 – Execução do exemplo_3.js

- ***Criação e execução de programas com o Node.js***
 - ***Exemplo (04):***
 - ***Elabore um programa para uma veterinária, que leia o peso de uma ração em kg e o quanto um gato consome por dia da ração, em gramas. Informe quantos dias irá durar a ração e o quanto sobra da ração (em gramas).***

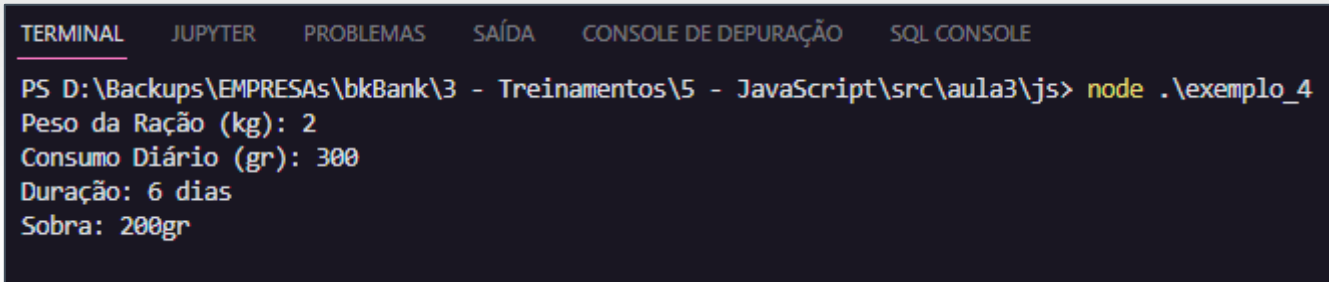


- ***Criação e execução de programas com o Node.js***
 - ***Exemplo (04):***
 - ***Solicitação da veterinária***

```
js > JS exemplo_4.js > ...
1  const prompt = require("prompt-sync")()           // adiciona o pacote ao programa
2  const pesoKg = Number(prompt("Peso da Ração (kg): ")) // lê dados de entrada
3  const consumo = Number(prompt("Consumo Diário (gr): "))
4
5  const pesoGr = pesoKg * 1000                       // cria variável auxiliar pesoGr
6  const duracao = Math.floor(pesoGr / consumo)       // cálculo das respostas
7  const sobra = pesoGr % consumo
8
9  console.log(`Duração: ${duracao} dias`)           // dados de saída
10 console.log(`Sobra: ${sobra}gr \n\n`)
```

Figura 8 – Criação do exemplo_4.js

- ***Criação e execução de programas com o Node.js***
 - ***Exemplo (04):***
 - ***Solicitação da veterinária***



The image shows a terminal window with a dark background and light-colored text. At the top, there are several tabs: 'TERMINAL' (which is active and underlined), 'JUPYTER', 'PROBLEMAS', 'SAÍDA', 'CONSOLE DE DEPURAÇÃO', and 'SQL CONSOLE'. Below the tabs, the terminal displays the following text: 'PS D:\Backups\EMPRESAs\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3\js> node .\exemplo_4'. This is followed by four lines of output: 'Peso da Ração (kg): 2', 'Consumo Diário (gr): 300', 'Duração: 6 dias', and 'Sobra: 200gr'.

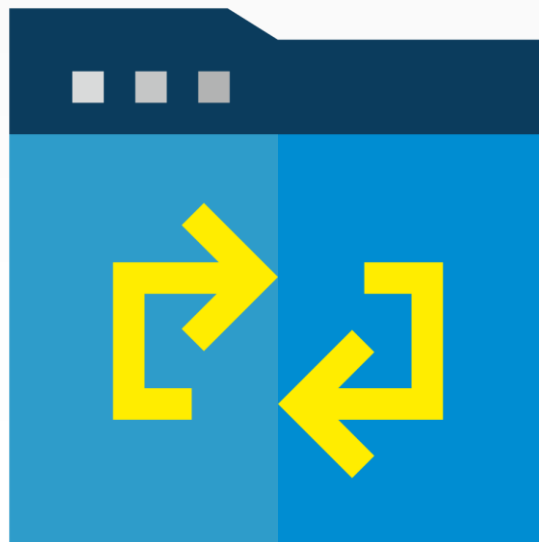
```
TERMINAL  JUPYTER  PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  SQL CONSOLE
PS D:\Backups\EMPRESAs\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3\js> node .\exemplo_4
Peso da Ração (kg): 2
Consumo Diário (gr): 300
Duração: 6 dias
Sobra: 200gr
```

Figura 9 – Execução do exemplo_4.js

- **Observações:**

- *Node.js* permite que **vários pacotes adicionais** sejam **acrescentados** aos **programas**, tornando-os **aptos a realizar tarefas** além das **básicas disponíveis na linguagem**, como **pacotes** que **permitem configurar e enviar e-mails**, **se comunicar com bancos de dados**, **manipular imagens etc**
- A **sintaxe das instruções em JS** é a **mesma (Node.js/browser)**





ESTRUTURAS CONDICIONAIS

- ***If...else***
 - *Para criar uma estrutura condicional básica, utilizamos os comandos **if** ... **else** (**se** ... **senão**)*
 - *Eles possuem algumas variações*
 - *É possível utilizar apenas o **if** (exemplo: para apresentar uma mensagem caso o cliente seja menor de idade)*
 - *Criar vários comandos **else** (verificar a classificação etária de um aluno de natação, que poderia ser infantil, juvenil ou adulto)*

- ***If...else***

- ***Sintaxe***

```
// define uma condição simples  
if (condição) {  
    comandos  
}
```

```
// define uma condição de if ... else  
if (condição) {  
    comandos (TRUE)  
} else {  
    comandos (FALSE)  
}
```

Condições

- ***If...else***

- ***Sintaxe***

```
// define múltiplas condições  
if (condição 1) {  
    comandos (1)  
} else if (condição 2) {  
    comandos (2)  
} else {  
    comandos (3)  
}
```

Quando houver apenas um comando que pertence à condição, o uso das chaves **não** é obrigatório.

Para facilitar a compreensão, recomenda-se utilizar as **{ }** em todas as ocorrências das estruturas condicionais de um programa.

Operadores

- **Operadores Relacionais**

- Para definir as condições utilizadas nas **estruturas condicionais**, deve-se fazer uso dos **operadores relacionais**
- Quando inseridas em um programa, cada comparação deve retornar **true** (verdadeiro) ou **false** (falso)

Símbolo	Significado
==	Igual. Retorna verdadeiro caso os dados contenham o mesmo conteúdo
!=	Diferente. Retorna verdadeiro caso os dados contenham conteúdos diferentes
>	Maior. Pode ser utilizado para comparar números ou palavras. Na comparação de palavras, a classificação alfabética é avaliada

Tabela 1 – Operadores relacionais

Operadores

- ***Operadores Relacionais***

Símbolo	Significado
<	Menor. Também podem ser realizadas comparações de números ou palavras
>=	Menor ou igual. Os símbolos devem estar nesta ordem (>=)
<=	Menor ou igual. Tenha cuidado com a ordem dos símbolos (<=)

Tabela 1 – Operadores relacionais

- ***Existem ainda os símbolos de === (estritamente igual) e !== (estritamente diferente)***
- ***Eles comparam também o tipo do dado em análise***
- ***'5' === 5 retorna false e '5' !== 5 retorna verdadeiro***

Operadores

- **Operadores Relacionais**

- **Exemplo (05):**

- *Realiza a leitura do nome e das notas de um aluno, apresenta a média e uma mensagem para o aluno: “Parabéns ... Você foi aprovado(a)!” ou, então, “Ops... Você foi reprovado(a)”*
 - *A situação de **aprovado** ou **reprovado** é definida pela média das notas, que deve ser **7.0** ou superior para aprovação*
 - *Caso a nota seja inferior a **7.0**, a mensagem indicando a reprovação deve ser exibida*
 - *Aplicaremos um estilo na mensagem que indica a situação do aluno (a mensagem de aprovação é exibida em **azul** e de reprovação em **vermelho**)*

Operadores

- *Operadores Relacionais*
 - *Exemplo (05):*

```
html > exemplo_5.html > html > body > form > input
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Exemplo 05 | bkBank Academy</title>
8  </head>
9  <body>
10     <h1> Situação do Aluno </h1>
11     <form>
12         <p> Nome do Aluno:
13             <input type="text" id="inNome" required>
14         </p>
15         <p> 1ª Nota:
16             <input type="number" step="0.1" min="0" max="10" id="inNota1" required>
17         </p>
18         <p> 2ª Nota:
19             <input type="number" step="0.1" min="0" max="10" id="inNota2" required>
20         </p>
21         <input type="submit" value="Exibir Média e Situação">
22     </form>
23     <h3></h3>
24     <h4></h4>
25     <script src="../../js/exemplo_5.js"></script>
26 </body>
27 </html>
```

Figura 10 – HTML do Exemplo (05)

Operadores

- ***Operadores Relacionais***
 - ***Exemplo (05):***



A screenshot of a web browser window titled "Exemplo 05 | bkBank Academy". The address bar shows a local file path: "D:/Backups/EMPRESAs/bkBank/3%20-...". The page content is titled "Situação do Aluno". It features a form with the following elements:

- A label "Nome do Aluno:" followed by a text input field containing the name "Rafael".
- A label "1ª Nota:" followed by a text input field containing the value "7.5".
- A label "2ª Nota:" followed by a text input field containing the value "8.0".
- A button labeled "Exibir Média e Situação".
- Below the button, the text "Média das Notas 7.75" is displayed.
- At the bottom, a blue message reads "Parabéns Rafael! Você foi aprovado(a)".

Figura 11 – Renderização do HTML

Operadores

- *Operadores Relacionais*
 - *Exemplo (05): - Parte 1*

```
js > JS exemplo_5.js > frm.addEventListener("submit") callback
1 // cria referência ao form e elementos de resposta do programa
2 const frm = document.querySelector("form")
3 const resp1 = document.querySelector("h3")
4 const resp2 = document.querySelector("h4")
5
6 // cria um "ouvinte" de evento, acionado quando o btn submit for acionado
7 frm.addEventListener("submit", (e) => {
8     e.preventDefault() // evita envio do form
9     const nome = frm.inNome.value // obtém valores do form
10    const nota1 = Number(frm.inNota1.value)
11    const nota2 = Number(frm.inNota2.value)
12
13    const media = (nota1 + nota2) / 2 // calcula a média das notas
14    resp1.innerHTML = `Média das Notas ${media.toFixed(2)}`
15
```

Figura 12 – JS do Exemplo (05) – Parte 1

Operadores

- *Operadores Relacionais*
 - *Exemplo (05): - Parte 2*

```
16 // cria as condições
17 if (media >= 7){
18     // altera o texto e estilo da cor do elemento resp2
19     resp2.innerText = `Parabéns ${nome}! Você foi aprovado(a)`
20     resp2.style.color = "blue"
21 } else {
22     resp2.innerText = `Ops ${nome}... Você foi reprovado(a)`
23     resp2.style.color = "red"
24 }
25 })
```

Figura 13 – JS do Exemplo (05) – Parte 2

Operadores

- **Operadores Lógicos**

- ***Há situações em que mais do que uma condição deve ser analisada***

- a) ***um cliente quer um carro da cor azul ou cinza;***
 - b) ***o preço do carro deve ser inferior a R\$ 20.000,00 e o ano maior ou igual a 2010;***
 - c) ***o modelo do carro deve ser “fusca” e o preço menor que R\$ 8.000,00***



Operadores

- ***Operadores Lógicos***

- ***Para definir mais de uma condição em um programa, devemos utilizar os operadores lógicos***

Símbolo	Significado
!	Not. Indica negação. Inverte o resultado de uma comparação
&&	And. Indica conjunção. Retorna verdadeiro quando todas as comparações forem verdadeiras
	Or. Indica disjunção. Retorna verdadeiro se, no mínimo, uma das condições definidas for verdadeira

Tabela 2 – Operadores lógicos

Operadores

- **Operadores Lógicos**

- *Apresentaremos a seguir, duas variáveis, **cor** e **ano**, e um valor que ela pode assumir na execução do programa*

```
const cor = "Azul"  
const ano = 2017
```

- *A **negação** é o mais simples dos operadores relacionais. Ela inverte o resultado (**verdadeiro** ou **falso**) de uma condição. Equivale ao sinal de (**!=**) quando puder ser aplicado*

```
if (!cor == "Azul") { ... }  
if (cor != "Azul") { ... }
```

Operadores

- **Operadores Lógicos**

- A **conjunção**, representada pelos símbolos **&&**, reflete a ideia da **simultaneidade**
- Na **conjunção**, a expressão só retorna verdadeiro se todas as comparações forem **verdadeiras**
- Se um cliente quer um carro azul e de 2017, ele só será atendido se as duas condições forem satisfeitas

```
if (cor == "Azul" && ano == 2017) { ... }  
if (cor == "Cinza" && ano < 2017) { ... }  
if (ano >= 2012 && ano <= 2017) { ... }  
if (cor != "Azul" && cor != "Vermelho") { ... }
```

Operadores

- **Operadores Lógicos**

- Na **disjunção**, símbolos **||**, no mínimo uma das condições deve ser **verdadeira**
- Agora, nosso **cliente** quer um carro de **cor azul** ou de **2017**
- **Qualquer** carro em que uma dessas condições for **verdadeira** serve para esse **cliente**

```
if (cor == "Azul" || ano == 2017) { ... }  
if (cor == "Azul" || cor == "Branco") { ... }  
if ((cor == "Azul" || cor == "Branco") && ano == 2017) { ... }  
if (cor == "Azul" && (ano == 2016 || ano == 2017)) { ... }
```

- ***Operadores Relacionais***

- ***Exemplo (06):***

- ***Calcular o peso ideal de uma pessoa***
 - ***Para isso, foram pesquisados alguns sites sobre o assunto***
 - ***Em um deles, há a indicação de que o peso ideal de um adulto pode ser calculado a partir das fórmulas:***

22 * altura² (para homens)

21 * altura² (para mulheres)

Operadores

- **Operadores Relacionais**
 - **Exemplo (06):**
 - **Cálculo do Peso Ideal**



Programa Cálculo do Peso Ideal

Nome:

Sexo: ☒ Masculino ☐ Feminino

Altura:

Geraldo Henrique: Seu peso ideal é 69.70 kg

Figura 14 – Renderização do HTML

- ***Operadores Relacionais***

- ***Exemplo (06):***

- ***Inclua as seguintes linhas no arquivo estilos.css***

```
img { float: left; height: 300px; width: 300px; }  
h1 { border-bottom-style: inset; }
```

Operadores

- *Operadores Relacionais*
 - *Exemplo (06): - Parte 1*
 - *Cálculo do Peso Ideal*

```
html > exemplo_6.html > html > body > form > p > input#inFeminino
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="stylesheet" href="../css/estilos.css">
8      <title>Exemplo 06 | bkBank Academy</title>
9  </head>
10 <body>
11     
12     <h1> Programa Cálculo do Peso Ideal </h1>
13     <form>
14         <p> Nome:
15             <input type="text" id="inNome" required>
16         </p>
```

Figura 15 – HTML do Exemplo (06) – Parte 1

Operadores

- ***Operadores Relacionais***
 - ***Exemplo (06): - Parte 2***
 - ***Cálculo do Peso Ideal***

```
17     <p> Sexo:  
18         <input type="radio" name="sexo" id="inMasculino" required> Masculino  
19         <input type="radio" name="sexo" id="inFeminino" required> Feminino  
20     </p>  
21     <p> Altura:  
22         <input type="number" step="0.01" min="0" id="inAltura" required>  
23     </p>  
24     <input type="submit" value="Calcular Peso Ideal">  
25     <input type="reset" value="Limpar Campos">  
26 </form>  
27 </h3></h3>  
28 <script src="../../js/exemplo_6.js"></script>  
29 </body>  
30 </html>
```

Figura 16 – HTML do Exemplo (06) – Parte 2

Operadores

- *Operadores Relacionais*
 - *Exemplo (06): - Parte 1*
 - *Cálculo do Peso Ideal*

```
js > JS exemplo_6.js > frm.addEventListener("submit") callback
1 // cria referência ao form e elemento onde será exibida a resposta
2 const frm = document.querySelector("form")
3 const resp = document.querySelector("h3")
4
5 // "ouvinte" de evento, acionado quando o botão submit for clicado
6 frm.addEventListener("submit", (e) => {
7   e.preventDefault(); // evita envio do form
8
9   const nome = frm.inNome.value // obtém valores do form
10  const masculino = frm.inMasculino.checked
11  const altura = Number(frm.inAltura.value)
12
```

Figura 17 – JS do Exemplo (06) – Parte 1

Operadores

- *Operadores Relacionais*
 - *Exemplo (06): - Parte 1*
 - *Cálculo do Peso Ideal*

```
13 let peso // declara a variável peso
14 if (masculino) { // se masculino (ou, if masculino == true)
15     peso = 22 * Math.pow(altura, 2) // Math.pow eleva ao quadrado
16 } else {
17     peso = 21 * Math.pow(altura, 2)
18 }
19
20 // apresenta a resposta (altera o conteúdo do elemento h3 da página)
21 resp.innerText = `${nome}: Seu peso ideal é ${peso.toFixed(2)} kg`
22 })
```

Figura 17 – JS do Exemplo (06) – Parte 2

Operadores

- *Operadores Relacionais*



Acionar o botão “*Limpar Campos*” limpa o conteúdo dos campos do formulário, mas não limpa o espaço onde é exibida a resposta do programa

Operadores

- **Operador Ternário**

- *Forma abreviada para criar as instruções **if .. else***
- *Consiste em realizar uma **atribuição** para uma **variável** com **base na análise de uma condição***
- **Exemplo:**

```
const categoria = idade >= 18 ? "Adulto" : "Juvenil"
```

condição

true

false

Operadores

- ***Switch ... Case***

- *Útil quando tivermos várias condições definidas a partir do conteúdo de uma variável*
- ***Exemplo: - Parte 1***
 - *Informar a taxa de entrega de um medicamento em uma farmácia, conforme o bairro do cliente*

```
<script>  
  const bairro = prompt("Bairro de Entrega: ")  
  let taxaEntrega  
  
  ...
```

Operadores

- ***Switch ... Case***
 - ***Exemplo: - Parte 2***

```
switch(bairro) {  
  case "Centro":  
    taxaEntrega = 5.00  
    break  
  case "Iguatemi":  
  case "Nova Aliança":  
    taxaEntrega = 7.00  
    break  
  case "Sumarezinho"  
    taxaEntrega = 10.00  
    break  
  default:  
    taxaEntrega = 8.00  
}  
alert(`Taxa R$: ${taxaEntrega.toFixed(2)}`)  
</script>
```

Operadores

- ***Operadores Relacionais***

- ***Exemplo (07):***

- ***Sabendo que o fuso horário da França em relação ao Brasil é de + 5 horas (no horário de verão na França), elabore um programa que leia a hora no Brasil e informe a hora na França***



Operadores

- **Operadores Relacionais**
 - **Exemplo (07):**
 - **Fuso Horário**



Figura 18 – Renderização do HTML

Operadores

- ***Operadores Relacionais***
 - ***Exemplo (07): - Parte 1***
 - ***Fuso Horário***

```
html > exemplo_7.html > html
1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <link rel="stylesheet" href="../css/estilos.css">
8     <title>Exemplo 07 | bkBank Academy</title>
9 </head>
```

Figura 19 – HTML do Exemplo (07) – Parte 1

Operadores

- *Operadores Relacionais*
 - *Exemplo (07): - Parte 2*
 - *Fuso Horário*

```
10 <body>
11   
12   <h1> Programa Fuso Horário </h1>
13   <form>
14     <p> Hora do Brasil (h.m):
15       <input type="number" id="inHoraBrasil" min="0" max="23.59" step="0.01" required>
16     </p>
17     <input type="submit" value="Exibir Hora na França">
18   </form>
19   <h3></h3>
20   <script src="../../js/exemplo_7.js"></script>
21 </body>
22 </html>
```

Figura 20 – HTML do Exemplo (07) – Parte 2

Operadores

- *Operadores Relacionais*

- *Exemplo (07):*

- *Fuso Horário*

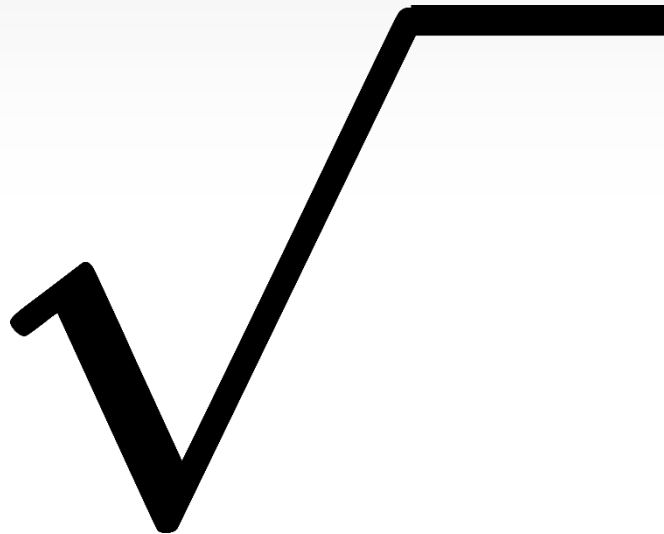
```
js > JS exemplo_7js > ...
1 // cria referência ao form e elemento onde será exibida a resposta
2 const frm = document.querySelector("form")
3 const resp = document.querySelector("h3")
4
5 // "ouvinte" de evento, acionado quando o botão submit for clicado
6 frm.addEventListener("submit", (e) => {
7     e.preventDefault() // evita envio do form
8
9     // obtém e converte o conteúdo do campo inHoraBrasil
10    const horaBrasil = Number(frm.inHoraBrasil.value)
11    let horaFranca = horaBrasil + 5 // calcula o horário na França
12
13    if (horaFranca > 24){ // se passar das 24 horas na França
14        horaFranca = horaFranca - 24 // ... subtrai 24
15    }
16    // exibe a resposta (altera o conteúdo do elemento h3 da página)
17    resp.innerText = `Hora na França ${horaFranca.toFixed(2)}`
18 })
```

Figura 21 – JS do Exemplo (07)

- ***Operadores Relacionais***

- ***Exemplo (08):***

- ***Elabore um programa que leia um número e calcule sua raiz quadrada. Caso a raiz seja exata (quadrados perfeitos), informa-la, caso contrário, informe: “Não há raiz exata para...”***



Operadores

- ***Operadores Relacionais***
 - ***Exemplo (08):***
 - ***Raiz Quadrada***



Programa Raiz

Número:

Raiz: 5

Figura 22 – Renderização do HTML

Operadores

- *Operadores Relacionais*
 - *Exemplo (08): - Parte 1*
 - *Raiz Quadrada*

```
html > S exemplo_8.html > html > body > h3
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="stylesheet" href="../css/estilos.css">
8      <title>Exemplo 08 | bkBank Academy</title>
9  </head>
10 <body>
```

Figura 23 – HTML do Exemplo (08) – Parte 1

Operadores

- *Operadores Relacionais*
 - *Exemplo (08): - Parte 2*
 - *Raiz Quadrada*

```
10 <body>
11   
12   <h1> Programa Raiz </h1>
13   <form>
14     <p> Número:
15       <input type="number" id="inNumero" required>
16     </p>
17     <input type="submit" value="Exibir Raiz Quadrada">
18   </form>
19   <h3></h3>
20   <script src="../js/exemplo_8.js"></script>
21 </body>
22 </html>
```

Figura 24 – HTML do Exemplo (08) – Parte 2

Operadores

- *Operadores Relacionais*

- *Exemplo (08):*

- *Raiz Quadrada*

```
js > JS exemplo_8js > ...
1  const frm = document.querySelector("form") // obtém elementos da página
2  const resp = document.querySelector("h3")
3
4  frm.addEventListener("submit", (e) => {
5      e.preventDefault() // evita envio do form
6      const numero = Number(frm.inNumero.value) // obtém número digitado no form
7      const raiz = Math.sqrt(numero) // calcula raiz quadrada do número
8
9      if (Number.isInteger(raiz)) { // se valor da raiz for um número inteiro
10         resp.innerText = `Raiz: ${raiz}` // ...mostra a raiz
11     } else {
12         resp.innerText = `Não há raiz exata para ${numero}` // ... mostra mensagem
13     }
14 }
```

Figura 25 – JS do Exemplo (08)

Operadores

- **Operadores Relacionais**

- **Exemplo (09):**

- *Em um determinado momento do dia, apenas notas de 10, 50 e 100 estão disponíveis em um terminal de caixa eletrônico. Elaborar um programa que leia um valor de saque de um cliente, verifique sua validade (ou seja, de pode ser pago com as notas disponíveis) e informe o número mínimo de notas de 100, 50 e 10 necessárias para pagar esse saque*



Operadores

- ***Operadores Relacionais***
 - ***Exemplo (09):***
 - ***Caixa Eletrônico***



Programa Caixa Eletrônico

Valor do Saque R\$:

Notas de RS 100: 2

Notas de RS 50: 1

Notas de RS 10: 2

Figura 26 – Renderização do HTML

Operadores

- *Operadores Relacionais*
 - *Exemplo (09): - Parte 1*
 - *Caixa Eletrônico*

```
html > exemplo_9.html > html > body
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="stylesheet" href="../css/estilos.css">
8      <title>Exemplo 09 | bkBank Academy</title>
9  </head>
```

Figura 27 – HTML do Exemplo (09) – Parte 1

Operadores

- *Operadores Relacionais*
 - *Exemplo (09): - Parte 2*
 - *Caixa Eletrônico*

```
10 <body>
11   
12   <h1> Programa Caixa Eletrônico </h1>
13   <form>
14     <p> Valor do Saque R$:
15     |   <input type="number" id="inSaque" min="10" required>
16     </p>
17     <input type="submit" value="Exibir Notas para Saque">
18   </form>
19   <h3 id="outResp1"></h3>
20   <h3 id="outResp2"></h3>
21   <h3 id="outResp3"></h3>
22   <script src="../js/exemplo_9.js"></script>
23 </body>
24 </html>
```

Figura 28 – HTML do Exemplo (09) – Parte 2

Operadores

- *Operadores Relacionais*
 - *Exemplo (09): - Parte 1*
 - *Caixa Eletrônico*

```
js > JS exemplo_9.js > frm.addEventListener("submit") callback
1  const frm = document.querySelector("form")           // obtém elementos da página
2  const resp1 = document.querySelector("#outResp1")
3  const resp2 = document.querySelector("#outResp2")
4  const resp3 = document.querySelector("#outResp3")
5
6  frm.addEventListener("submit", (e) => {              // "escuta" evento submit do form
7      e.preventDefault()                               // evita envio do form
8      const saque = Number(frm.inSaque.value)          // obtém valor do saque
9
10     if (saque % 10 !== 0){                             // se saque não é múltiplo de 10
11         alert("Valor inválido para notas disponíveis (R$ 10, 50, 100)")
12         frm.inSaque.focus()
13         return
14     }
```

Figura 29 – JS do Exemplo (09) – Parte 1

Operadores

- *Operadores Relacionais*
 - *Exemplo (09): - Parte 2*
 - *Caixa Eletrônico*

```
16     const notasCem = Math.floor(saque / 100) // calcula notas de 100
17     let resto = saque % 100                  // quanto sobra para pagar
18
19     const notasCinquenta = Math.floor(resto / 50) // calcula notas de 50
20     resto = resto % 50                        // quanto ainda sobra
21
22     const notasDez = Math.floor(resto / 10) // calcula notas de 10
23     if (notasCem > 0) {
24         resp1.innerHTML = `Notas de R$ 100: ${notasCem}`
25     }
26     if (notasCinquenta > 0) {
27         resp2.innerHTML = `Notas de R$ 50: ${notasCinquenta}`
28     }
29     if (notasDez > 0) {
30         resp3.innerHTML = `Notas de R$ 10: ${notasDez}`
31     }
32 })
```

Figura 30 – JS do Exemplo (09) – Parte 2

Operadores

- **Operadores Relacionais**

- **Exemplo (09): - Parte 2**

- Se o valor solicitado **não** for **múltiplo** de R\$ 10,00, o resto da divisão do valor por **10** produzirá um valor diferente de zero, e a mensagem de advertência será exibida
 - A realização de testes de validação contendo o comando **return** são uma forma de evitar a criação de diversos comandos **else** no programa. Assim, caso algum campo apresente um **erro** de validação, a mensagem é exibida e o programa retorna à página. Após as validações, a programação é realizada sem os possíveis problemas que dados inválidos poderiam causar, como uma divisão por zero

Operadores

- **Operadores Relacionais**

- **Exemplo (09): - Parte 2**

- Para calcular o número mínimo de notas de **100**, **50** e **10** necessárias para pagar um saque, iniciamos pelo cálculo do número de notas de **100**. Utilizamos a função **Math.floor()** para arredondar para baixo o resultado da divisão do valor solicitado por **100** ($490/100 \rightarrow 4.9$, com o **Math.floor()** resulta **4**; $1240/100 \rightarrow 12.4$, com **Math.floor()** resulta **12**)
 - O próximo passo é obter o valor que ainda **não** foi pago com as notas de **100**



Operadores

- **Operadores Relacionais**

- **Exemplo (09): - Parte 2**

- Para isso, **podemos utilizar o operador módulo (%)** outra vez. **Por exemplo:** $490 \% 100$, resulta **90**; $1240 \% 100$, resulta **40**, ou seja, os valores que **precisam ser pagos** com as **notas de 50 e 10**
 - E o **processo para o cálculo das notas de 100** e para o **cálculo do resto é então aplicado novamente para as notas de 50 e 10** (poderíamos aplicar para outras notas (20 ou 5))



- **Exemplos com Node.js**

- **Exemplo (10):**

- A entrada para um clube de pesca custa R\$ 20,00 por pessoa e cada pessoa tem direito a levar um peixe. Peixes extras custam R\$ 12,00. Elabore um programa que leia o número de pessoas de uma família que foram ao clube e o número de peixes obtidos na pescaria. Informe o valor a pagar.



- *Exemplos com Node.js*

- *Exemplo (10):*

- *Clube de Pesca*

```
js > JS exemplo_10.js > ...
1  const prompt = require("prompt-sync")() // adiciona pacote prompt-sync
2  const pessoas = Number(prompt("Nº de Pessoas: ")) // lê dados de entrada
3  const peixes = Number(prompt("Nº de Peixes: "))
4
5  let pagar // declara variável pagar
6  if (peixes <= pessoas) { // condição definida no enunciado
7      pagar = pessoas * 20
8  } else {
9      pagar = (pessoas * 20) + ((peixes - pessoas) * 12)
10 }
11 console.log(`Pagar R$: ${pagar.toFixed(2)} \n\n`) // exibe o valor a ser pago
```

TERMINAL

JUPYTER

PROBLEMAS

SAÍDA

CONSOLE DE DEPURACÃO

SQL CONSOLE

```
PS D:\Backups\EMPRESAS\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3\js> node .\exemplo_10
Nº de Pessoas: 4
Nº de Peixes: 5
Pagar R$: 92.00
```

Figura 31 – JS do Exemplo (10)
Aula 03 | Módulo Básico

- ***Exemplos com Node.js***

- ***Exemplo (11):***

- ***Uma farmácia necessita de um programa que leia o total de uma compra. Exiba como resposta o nº máximo de vezes que o cliente pode parcelar essa compra e o valor de cada parcela. Considere as seguintes condições:***

- a) cada parcela deve ser de, no mínimo, R\$ 20,00***

- b) o número máximo de parcelas permitido é 6***



- *Exemplos com Node.js*

- *Exemplo (11):*

- *Farmácia*

```
js > JS exemplo_11.js > ...
1  const prompt = require("prompt-sync")() // adiciona pacote prompt-sync
2  const valor = Number(prompt("Valor da Compra R$: ")) // lê valor da compra
3  const aux = Math.floor(valor / 20) // aux = nº de parcelas não permitida
4  const parcelas = aux == 0 ? 1 : aux > 6 ? 6 : aux // operador ternário
5  const valorParcela = valor / parcelas // cálculo do valor da parcela
6  console.log(`Pode pagar em ${parcelas}x de R$ ${valorParcela.toFixed(2)} \n\n`)
```

TERMINAL JUPYTER PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO SQL CONSOLE

```
PS D:\Backups\EMPRESAS\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3\js> node .\exemplo_11
Valor da Compra R$: 90.00
Pode pagar em 4x de R$ 22.50
```

Figura 32 – JS do Exemplo (11)

- ***Exemplos com Node.js***

- ***Exemplo (12):***

- ***Elabore um programa que leia um número – que deve ser uma centena. Calcule e exiba a centena invertida. Caso o número **não** seja uma centena, exiba uma mensagem.***



- *Exemplos com Node.js*

- *Exemplo (12):*

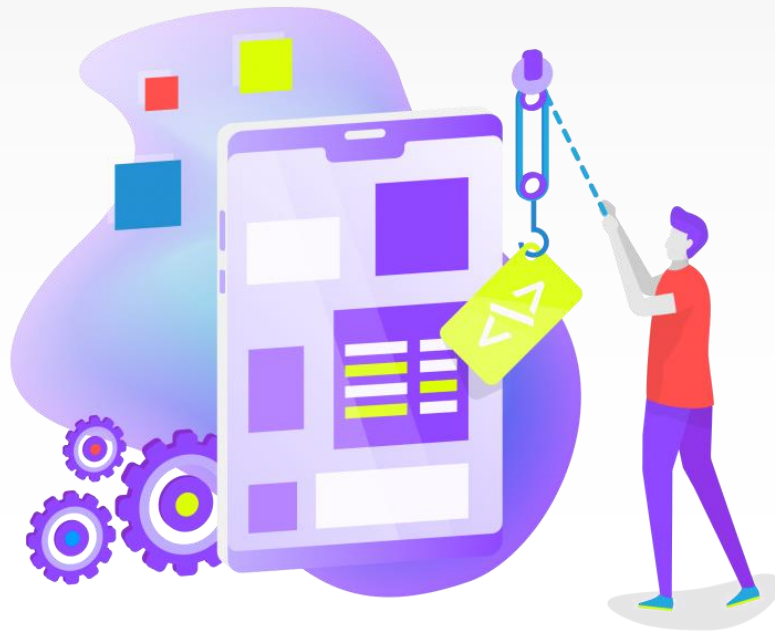
- *Centena Invertida*

```
js > JS exemplo_12.js > ...
1  const prompt = require("prompt-sync")()    // adiciona pacote prompt-sync
2  const numero = Number(prompt("Numero (centena): "))    // lê o número
3
4  if (numero < 100 || numero >= 1000){
5      console.log("Erro... deve ser uma centena")
6      return
7  }
8  const num1 = Math.floor(numero / 100)    // obtém 1º número
9  const sobra = numero % 100    // o que sobra (dezena)
10 const num2 = Math.floor(sobra / 10)    // obtém 2º número
11 const num3 = sobra % 10    // obtém 3º número
12 console.log(`Invertido: ${num3}${num2}${num1} \n\n`) // exibe o número invertido
```

TERMINAL JUPYTER PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO SQL CONSOLE

```
PS D:\Backups\EMPRESAS\bkBank\3 - Treinamentos\5 - JavaScript\src\aula3\js> node .\exemplo_12
Numero (centena): 278
Invertido: 872
```

Figura 33 – JS do Exemplo (12)



EXERCÍCIOS

Referências

Duckett, J.; Javascript e JQuery - Desenvolvimento de interfaces web interativas. Alta Books, 2018.

Flanagan, D.; JavaScript: The Definitive Guide, 7th Edition. O'Reilly Media, Inc. 2020.

Scott A. D., MacDonald M., Powers S.; JavaScript Cookbook, 3rd Edition. O'Reilly Media, Inc. 2021.

