

# *JavaScript*

*Módulo Básico*





# INTEGRAÇÃO COM HTML

- **Integração com HTML**
  - Os códigos de programas JS são desenvolvidos para adicionar um **comportamento** à página
  - **Não** é preciso **compilar** o programa ou outra ação adicional. O próprio navegador web contém um interpretador para os programas JS
  - Eles são inseridos nas páginas web em uma seção delimitada pelas tags **<script>** e **</script>** ou em um arquivo **.js** que deve ser referenciado pelo documento HTML

- ***Integração com HTML***

- Com **JS** podemos **interagir** de **diversas formas** com os **usuários** de **páginas web**
- Os **conceitos** serão agora **empregados** para **recuperar informações digitadas** em **campos de formulários** de uma **página** e para **exibir** os **resultados** em **parágrafos** do **documento HTML**



- ***Estrutura básica de um documento HTML***

- ***Visual Studio Code:***

- ***Inicie um novo arquivo (**Arquivo** > **Novo Arquivo**), salve o documento como sendo do tipo HTML e, posteriormente digite !***
    - ***O editor apresentará um recurso que permite inserir um modelo de códigos no documento (**Emmet Abbreviation**)***
    - ***Pressione **Tab** ou **Enter** para que os comandos básicos de uma página HTML sejam inseridos***
    - ***Observe que a linha **2** é especificado o idioma da página***
    - ***Troque o "**en**" (**English**) para "**pt-br**" (**Português** do **Brasil**)***

- *Estrutura básica de um documento HTML*
  - *Visual Studio Code:*

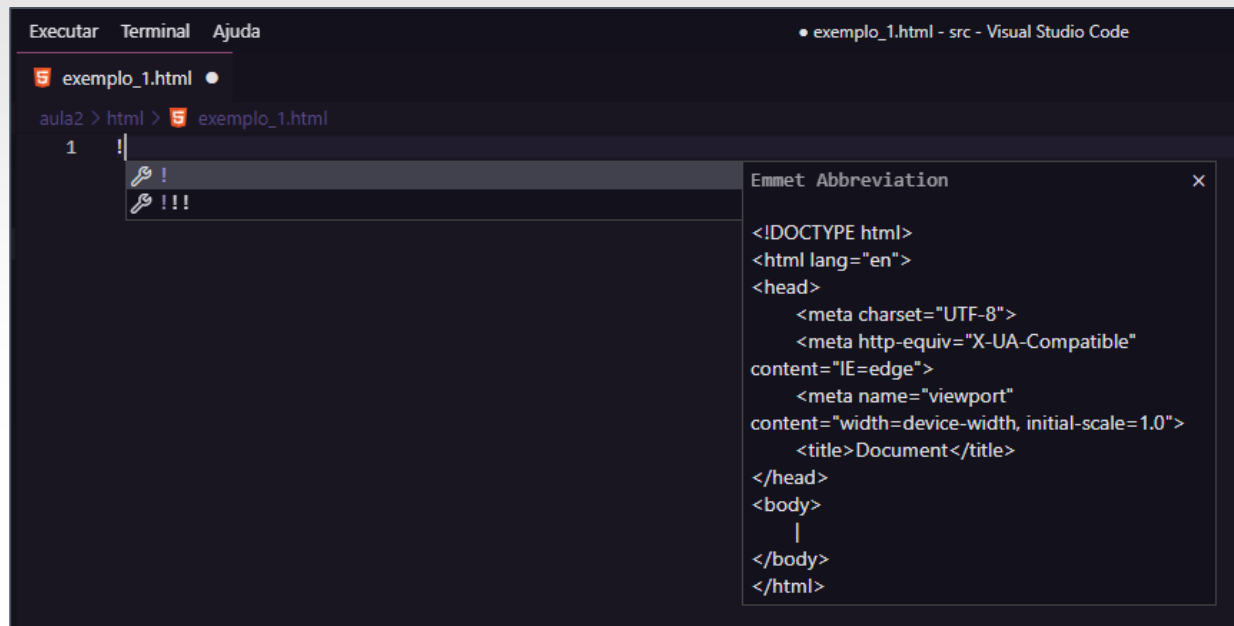
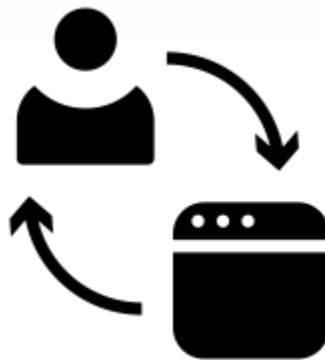


Figura 1 – Exemplo do uso do Emmet Abbreviation

- ***Estrutura básica de um documento HTML***

- ***Visual Studio Code:***

- Definir corretamente o **idioma** do documento é importante por diversos aspectos, como permitir uma **melhor pronúncia** por um software de leitura de tela (**portadores de necessidades especiais**) e indicar ao browser o dicionário a ser utilizado para a **correção gramatical** de textos digitados em campos de formulários



- *Estrutura básica de um documento HTML*
  - *Visual Studio Code:*



```
aula2 > html > exemplo_1.html > html > head > meta
1  <!DOCTYPE html>
2  <html lang="en">
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>Document</title>
8  </head>
9  <body>
10
11 </body>
12 </html>
```

Figura 2 – Ao pressionar *Tab* ou *Enter*, os comandos básicos de um documento HTML são inseridos



- ***Estrutura básica de um documento HTML***
  - ***Visual Studio Code:***
    - As **tags HTML** são **geralmente declaradas em pares**
    - Há `<html>` e `</html>`, `<head>` e `</head>`, `<body>` e `</body>`
    - As **tags** `<head>` e `<body>` definem as **seções principais** da página
    - Na **seção de cabeçalho (head)**, foram inseridas **três metatags** e o **título do documento** (factível de alteração)
    - O **título define o texto a ser exibido em uma aba na barra superior do navegador**
    - `<meta charset="utf-8">` define os **códigos suportados pelo documento**

- ***Estrutura básica de um documento HTML***

- ***Visual Studio Code:***

- ***<meta name="viewport" ...> está relacionada ao processo de criação de páginas responsivas (desktops, tablets e smartphones)***
- ***<meta http-equiv ...> tem relação com os aspectos de compatibilidade entre navegadores***



- *Cabeçalho, parágrafos e campos de formulários*
  - *A acrescentar outras tags HTML ao corpo (body) do documento*
  - *Cada campo deve possuir um **identificador** (**id**) a ser utilizado no código JS para obter o conteúdo do campo*
  - *<input type="**submit**" ...> utilizado para enviar dados para um destino específico*

- *Cabeçalho, parágrafos e campos de formulários*
  - *Exemplo (01):*
    - *Documento HTML com campos de formulário*

```
aula2 > html > exemplo_1.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      <title>Exemplo 01 | bkBank Academy</title>
8  </head>
9  <body>
10     <h1> Programa Olá Você! </h1>
11     <form>
12         <p> Nome:
13             <input type="text" id="inNome">
14             <input type="submit" value="Mostrar">
15         </p>
16     </form>
17     <h3></h3>
18     <script src="../js/exemplo_1.js"></script>
19 </body>
20 </html>
```

Figura 3 – HTML do Exemplo (01)  
Aula 02 | Módulo Básico

- ***Cabeçalho, parágrafos e campos de formulários***
  - ***Exemplo (01):***
    - ***Documento HTML com campos de formulário***

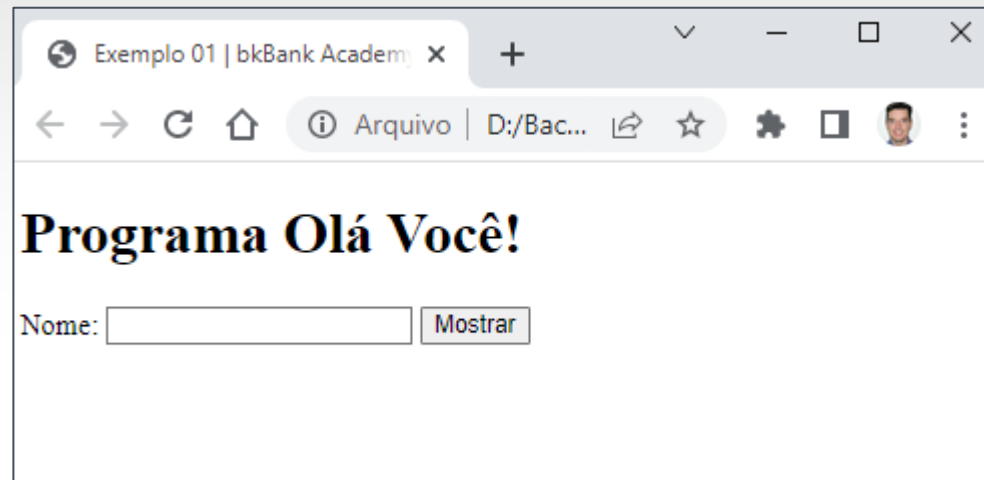


Figura 4 – Renderização do HTML com campos de formulário

- ***Criação do programa JS***
  - ***Concluído o código HTML, responsável por renderizar a página no browser, criaremos o arquivo JS, o qual será responsável por adicionar um comportamento à página***
  - ***Observe que o HTML já contém um link para esse programa JS na linha***

```
<script src="../../js/exemplo_1.js"></script>
```

- ***Criação do programa JS***
  - ***Exemplo (01):***
    - ***Exibe o nome informado pelo usuário no campo de edição***

```
aula2 > js > JS exemplo_1.js > frm.addEventListener("submit") callback
1 // cria referência ao form e ao elemento h3 (onde será exibida a resposta)
2 const frm = document.querySelector("form")
3 const resp = document.querySelector("h3")
4
5 // cria um "ouvinte" de evento, acionado quando o botão submit for clicado
6 frm.addEventListener("submit", (e) => {
7   const nome = frm.inNome.value // obtém o nome digitado no form
8   resp.innerHTML = `Olá ${nome}` // exibe a resposta do programa
9   e.preventDefault()           // evita o envio do form
10 })
```

Figura 5 – JS do Exemplo (01)

# Métodos

- **Métodos `querySelector()` e `getElementById()`**
  - Permitem **referenciar qualquer elemento** da página – como um campo de formulário, um parágrafo, um botão, uma imagem, etc.
  - **`getElementById()`:**
    - Para que um elemento HTML seja referenciado, ele **precisa conter um atributo `id`**
  - **`querySelector()`:**
    - Permite criar uma referência a um elemento HTML pela sua tag **`name`, `id` ou `classe`**

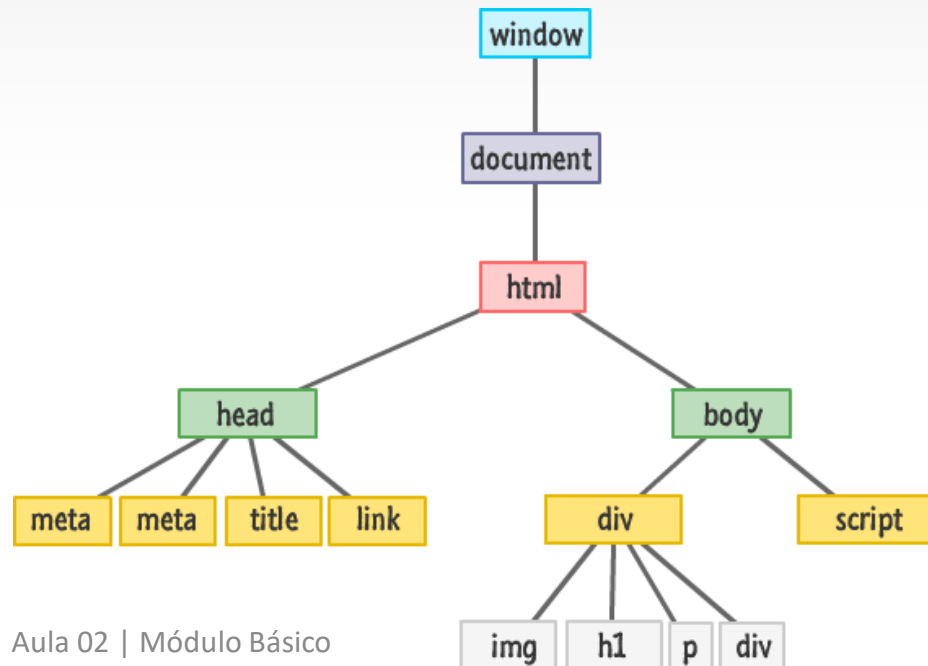
```
const resp = document.querySelector("h3") // primeiro elemento h3 da página
const cor = document.querySelector("#inCor") // elemento com id=inCor
const lista = document.querySelector(".lista") // elemento da class=lista
```



- **Métodos `querySelector()` e `getElementById()`**
  - `getSelector()`: **Não** é suportado por **versões antigas** dos navegadores
  - Para **solucionar** esse **problema**, ou seja, utilizar **recursos modernos de programação JS**, sem preocupar com essa questão, surgiram os chamados “**transpiladores**” de **código**
  - **Caso o programa precise funcionar em **versões antigas** dos browsers**, utilize o **transpiler** o qual **converterá o código moderno para instruções equivalentes compatíveis com esses browsers**
  - O **Babel** (**[babeljs.io](https://babeljs.io)**) é um dos **principais softwares desse segmento para JS**

# Métodos

- **Métodos `querySelector()` e `getElementById()`**
  - Além do `querySelector()`, que **cria uma referência ao primeiro elemento** da **página** que corresponda ao **seletor**, existe também o **método `querySelectorAll()`**, o qual **cria uma lista com todos os elementos que correspondam ao seletor**



- **Métodos `querySelector()` e `getElementById()`**
  - **É possível armazenar a *referência* a um elemento em uma variável e depois obter a sua propriedade**

```
const frm = document.querySelector("form")  
const nome = frm.inNome.value
```

- **Podemos ainda utilizar um único comando, acessando diretamente a propriedade**

```
const nome = document.querySelector("form").inNome.value
```

- ***Eventos e Funções***

- ***Muito da programação JS construída em páginas web é desenvolvida desta forma:***

- ***Elas são acionadas a partir da ocorrência de um evento***
    - ***Quando o usuário executa uma ação, o programa responde ao evento do usuário com uma ou mais ações***
    - ***O evento mais comum de ser implementado para um formulário é o clique no botão submit***
    - ***Há diversos outros, como modificar o conteúdo de um campo, clicar sobre um elemento da página, sair de um campo, carregar a página, etc.***

- **Eventos e Funções**

- Para **criar um evento** e definir o que será **executado** quando esse **evento ocorrer**, deve-se utilizar uma **palavra reservada** para **indicar** para qual **evento** a **linguagem** ficará “**na escuta**”
- A **palavra reservada** pode ser, por exemplo, **submit**, **change**, **click**, **blur** ou **load**
- Para **adicionar** um **ouvinte** de **evento** a um elemento da **página**, utiliza-se o **método** **addEventListener()**, com o **evento** e o **nome** de uma **função** ou uma **arrow function** (**função de seta**) com os **comandos** a serem **executados**

- *Eventos e Funções*

- *Sintaxe:*

```
form.addEventListener("submit", (e) => { comandos })
```



# Propriedades

- **Propriedades `innerText`, `innerHTML` e `value`**
  - **`value`:** *obtém* ou *altera* o conteúdo de uma campo de formulário HTML
  - **`innerText`:** permite *alterar* ou *obter* o conteúdo de elementos de texto do documento identificados no código HTML (*alterar* o texto de qualquer parágrafo ou texto de cabeçalho)
  - **`innerHTML`:** *similar* ao *innerText* quanto aos elementos em que *atua*, porém *renderiza* os códigos HTML existentes no seu conteúdo

# Propriedades

- **Propriedades *innerText*, *innerHTML* e *value***
  - A **Tabela 1**, a seguir, destaca a **diferença** entre as propriedades ***innerText***, ***innerHTML*** e ***value***

innerText	Consulta ou altera o texto exibido por elementos HTML como parágrafos (p), cabeçalhos (h1, h2, ...) ou containers (span, div)
innerHTML	Consulta ou altera o conteúdo de elementos HTML como parágrafos (p), cabeçalhos (h1, h2, ...) ou containers (span, div). Códigos HTML presentes no conteúdo são renderizados pelo navegador
value	Consulta ou altera o conteúdo de campos de formulário

Tabela 1 – Comparativo entre as propriedades *innerText*, *innerHTML* e *value*

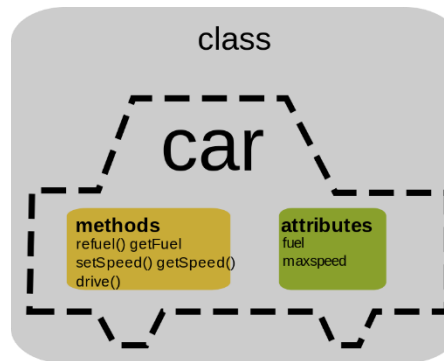


# Propriedades

- *Propriedades `innerText`, `innerHTML` e `value`*
  - *`innerHTML` pode apresentar algum **risco** relacionado à **segurança** na construção de páginas web em um tipo de ataque denominado XSS (*Cross-Site Scripting*)*
  - *Essa vulnerabilidade explora a exibição de dados contendo códigos que poderiam ser enviados por **usuários maliciosos***
  - *Para **evitar** esse **problema**, é necessário **filtrar** os **dados** de entrada de um site*
  - *Caso o conteúdo a ser exibido na página pelo programa **não** contenha **dados** informados pelo usuário, **não há riscos** em utilizar o `innerHTML`*

- **Método `preventDefault()`**
  - Por padrão, quando o **usuário clica** sobre o **botão submit** de um formulário, uma ação de envio dos dados desse **form** é executada
  - Isso faz um **reload** da **página**, e tanto o **conteúdo** dos **campos** do **form** quanto das **respostas exibidas** pelo **programa** são **perdidas**
  - **`preventDefault()`** previne esse comportamento **default** do **botão submit** (aplicado sobre um **event**, no exemplo anterior, pelo **parâmetro (e)** na **construção** da **arrow function**)

- *Método preventDefault()*
  - *Novos termos foram utilizados na descrição dos programas, como **objeto**, **método** e **propriedade***
    - *Objeto: representa uma instância de uma classe*
    - *Método: representa uma instrução ou um conjunto de instruções que executam uma tarefa*
    - *Propriedade: representa uma característica (**atributo**) de um objeto*



- **Método `preventDefault()`**
  - No exemplo (01), utilizamos o objeto `document`, que a partir da execução do método `querySelector()` pode referenciar as tags `form` e `h3` da página
  - A propriedade `value` foi utilizada para obter o conteúdo digitado no campo de formulário
  - A propriedade `innerText` alterou o atributo do documento, que permitiu que a resposta fosse exibida na página

- ***Método preventDefault()***
  - ***Exemplo (01):***
    - ***Exibe o nome informado pelo usuário no campo de edição***

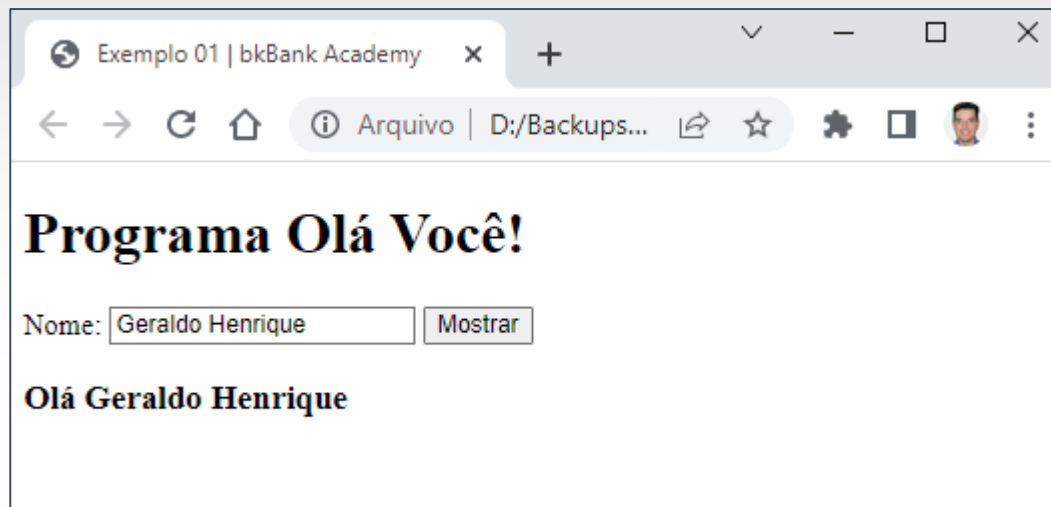


Figura 6 – Ao clicar no botão *Mostrar*, a mensagem *Olá* seguida do *nome* é exibida

# Operadores

- ***Operadores Aritméticos e Funções Matemáticas***
  - Além dos **tradicionais operadores** de **adição (+)**, **subtração (-)**, **multiplicação (\*)**, **divisão (/)** e **exponenciação (\*\*)**, as **linguagens de programação dispõem também do operador módulo (%)**
  - **O módulo é utilizado para obter o resto da divisão entre dois números**

```
const a = 5 % 2 // a = 1  
const b = 7 % 4 // b = 3
```

# Operadores

- ***Operadores Aritméticos e Funções Matemáticas***
  - ***Outros cálculos como raiz quadrada, seno e cosseno podem ser obtidos em JS como uso das funções matemáticas da classe **Math*****

Math.abs(num)	Retorna o valor absoluto de um número, ou seja, se o valor for negativo, ele será convertido para positivo. Se positivo, o valor permanece o mesmo. Exemplo: Math.abs(-3) → 3
Math.ceil(num)	Arredonda o valor para cima. Dessa forma, se o valor possuir decimais, retorna o próximo número inteiro do valor analisado. Exemplo: Math.ceil(4.2) → 5

Tabela 2 – Principais funções matemáticas da classe Math

# Operadores

- ***Operadores Aritméticos e Funções Matemáticas***
  - ***Outros cálculos como raiz quadrada, seno e cosseno podem ser obtidos em JS como uso das funções matemáticas da classe **Math*****

Math.floor(num)	Arredonda o valor para baixo, retornando a parte inteira do número. Exemplo: Math.floor(7.9) → 7
Math.pow(base, exp)	Retorna a base elevada ao expoente. Exemplo: Math.pow(3, 2) → 9
Math.random()	Retorna um número aleatório entre 0 e 1, com várias casas decimais. O número aleatório possível inicia em 0 e vai até um valor inferior a 1. Exemplo: Math.random() → 0.6501314074022906

Tabela 2 – Principais funções matemáticas da classe Math



# Operadores

- ***Operadores Aritméticos e Funções Matemáticas***
  - ***Outros cálculos como raiz quadrada, seno e cosseno podem ser obtidos em JS como uso das funções matemáticas da classe **Math*****

Math.round(num)	Arredonda o valor para o inteiro mais próximo. A partir de .5 na parte fracionária, o valor é arredondado para cima. Anterior a .5 é arredondado para baixo. Exemplo: Math.round(2.7) → 3
Math.sqrt(num)	Retorna a raiz quadrada do número ( <i>square root</i> ). Exemplo: Math.sqrt(16) → 4

Tabela 2 – Principais funções matemáticas da classe Math

# Operadores

- ***Operadores Aritméticos e Funções Matemáticas***
  - ***Ao criar expressões matemáticas, devemos ter o cuidado com a ordem de precedência dos operadores***

```
const media1 = (nota1 + nota2) / 2  
const media2 = nota1 + nota2 / 2
```

- ***O valor das variáveis **media1** e **media2** será o mesmo?***



# Operadores

- ***Operadores Aritméticos e Funções Matemáticas***
  - Ao montar uma **expressão matemática**, fique atento à **ordem hierárquica de execução dos operadores**
  - **Exemplo (A):**
    - Os **parênteses redefinem a ordem das prioridades**. Podem ser utilizados vários conjuntos de parênteses em uma mesma expressão

Exemplo...:  $10 * (6 - (2 * 2))$

Cálculo(1):  $10 * (6 - 4)$

Cálculo(2):  $10 * 2$

Resultado: 20

# Operadores

- ***Operadores Aritméticos e Funções Matemáticas***

- ***Exemplo (B):***

- ***As funções matemáticas ou funções criadas pelo usuário têm prioridades sobre os demais operadores aritméticos***

**Exemplo..:** `Math.sqrt(9) * 8 / 2`

**Cálculo(1):** `3 * 8 / 2`

**Cálculo(2):** `24 / 2`

**Resultado:** 12

# Operadores

- ***Operadores Aritméticos e Funções Matemáticas***

- ***Exemplo (C):***

- ***Os operadores de multiplicação, subtração e módulo têm prioridade sobre os operadores de adição e subtração***

**Exemplo...:**  $2 + 5 * 2$

**Cálculo(1):**  $2 + 10$

**Resultado:** 12

# Operadores

- ***Operadores Aritméticos e Funções Matemáticas***

- ***Exemplo (D):***

- ***Caso uma expressão contenha operadores de mesmo nível de hierarquia, o resultado é calculado da esquerda para a direita***

**Exemplo...:**  $5 / 2 * 3$

**Cálculo(1):**  $2.5 * 3$

**Resultado:** 7.5

# Operadores

- ***Operadores Aritméticos e Funções Matemáticas***
  - *Você pode modificar a ordem de execução de qualquer fórmula com a inserção de parênteses*
  - *Os parênteses também podem ser utilizados em algumas expressões para auxiliar na compreensão do cálculo*

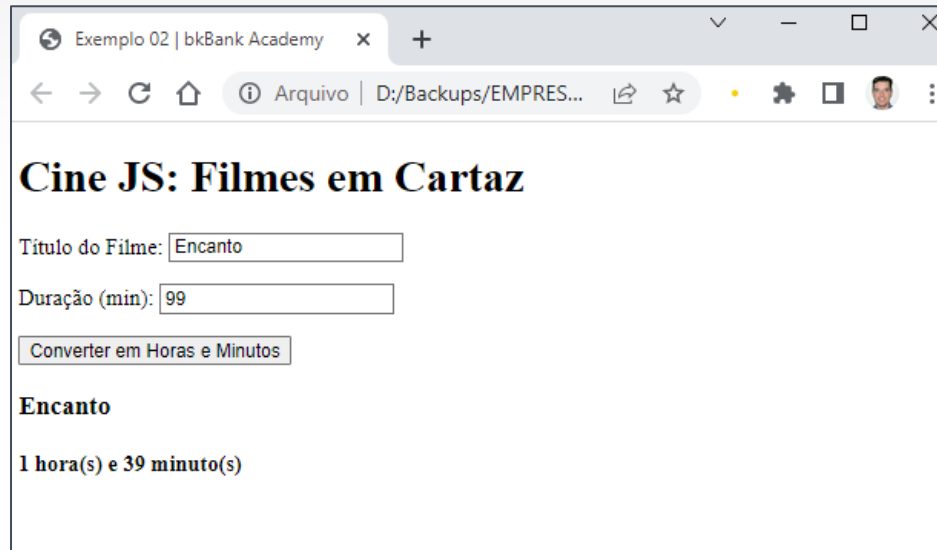
`const total = (dias * 24) + horas`



- ***Operadores Aritméticos e Funções Matemáticas***

- ***Exemplo (02):***

- ***Elabore um programa para um Cinema, que leia o título e a duração de um filme em minutos***
    - ***Exibir o título do filme e converta a duração para horas e minutos***



Exemplo 02 | bkBank Academy

Arquivo | D:/Backups/EMPRES...

### Cine JS: Filmes em Cartaz

Título do Filme:

Duração (min):

**Encanto**

1 hora(s) e 39 minuto(s)

Figura 7 – Programa deve converter a duração do filme



- *Operadores Aritméticos e Funções Matemáticas*
  - *Exemplo (02):*

```
aula2 > html > exemplo_2.html > html > body > script
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 02 | bkBank Academy</title>
8  </head>
9  <body>
10     <h1> Cine JS: Filmes em Cartaz </h1>
11     <form>
12         <p> Título do Filme:
13             <input type="text" id="inTitulo" required>
14         </p>
15         <p> Duração (min):
16             <input type="text" id="inDuracao" required>
17         </p>
18         <input type="submit" value="Converter em Horas e Minutos">
19     </form>
20     <h3></h3>
21     <h4></h4>
22     <script src="../../js/exemplo_2.js"></script>
23 </body>
24 </html>
```

Figura 8 – HTML do Exemplo (02)

- *Operadores Aritméticos e Funções Matemáticas*
  - *Exemplo (02):*

```
aula2 > js > JS exemplo_2.js > frm.addEventListener("submit") callback > horas
1 // cria referência ao form e aos elementos h3 e h4 (respostas)
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 botão submit for clicado
7 frm.addEventListener("submit", (e) => {
8     const titulo = frm.inTitulo.value // obtém conteúdo dos campos
9     const duracao = Number(frm.inDuracao.value)
10
11     const horas = Math.floor(duracao / 60) // arredonda para baixo o resultado
12     const minutos = duracao % 60 // obtém o resto da divisão
13
14     resp1.innerText = titulo // exibe as respostas
15     resp2.innerText = `${horas} hora(s) e ${minutos} minuto(s)`
16
17     e.preventDefault() // evita envio do form
18 })
```

Figura 9 – JS do Exemplo (02)

- ***Operadores Aritméticos e Funções Matemáticas***
  - ***Exemplo (03):***
    - ***Elabore um programa para uma revendedora de veículos. O programa deve ler modelo e preço do veículo***
    - ***Exibir como resposta o valor da entrada (50%) e o saldo em 12x***



Exemplo 03 | bkBank Academy

Arquivo | D:/Backups/EMPRES...

### Revenda de Veículos JS

Veículo: Sander 2018

Preço: 48000

Ver Promoção

Promoção: Sander 2018

Entrada de R\$: 24000.00

12 x de R\$ 2000.00

Figura 10 – Exemplo de dados do programa revenda de veículos

- *Operadores Aritméticos e Funções Matemáticas*
  - *Exemplo (03):*

```
aula2 > html > exemplo_3.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      <title>Exemplo 03 | bkBank Academy</title>
8  </head>
9  <body>
10     <h1> Revenda de Veículos JS </h1>
11     <form>
12         <p> Veículo:
13             <input type="text" id="inVeiculo" required>
14         </p>
15         <p> Preço:
16             <input type="text" id="inPreco" required>
17         </p>
18         <input type="submit" value="Ver Promoção">
19     </form>
20     <h3 id="outResp1"></h3>
21     <h3 id="outResp2"></h3>
22     <h3 id="outResp3"></h3>
23     <script src="../../js/exemplo_3.js"></script>
24 </body>
25 </html>
```

Figura 11 – HTML do Exemplo (03)

- *Operadores Aritméticos e Funções Matemáticas*
  - *Exemplo (03):*

```
aula2 > js > JS exemplo_3.js > ...
1 // cria referência ao form e aos elementos de resposta (pelo seu id)
2 const frm = document.querySelector("form")
3 const resp1 = document.querySelector("#outResp1")
4 const resp2 = document.querySelector("#outResp2")
5 const resp3 = document.querySelector("#outResp3")
6
7 // cria um "ouvinte" de evento, acionado quando o botão submit for clicado
8 frm.addEventListener("submit", (e) => {
9     const veiculo = frm.inVeiculo.value // obtém o conteúdo dos campos
10    const preco = Number(frm.inPreco.value)
11
12    const entrada = preco * 0.50 // calcula valor da entrada
13    const parcela = (preco * 0.50) / 12 // ... e das parcelas
14
15    resp1.innerText = `Promoção: ${veiculo}` // exibe as respostas
16    resp2.innerText = `Entrada de R$: ${entrada.toFixed(2)}`
17    resp3.innerText = `12 x de R$ ${parcela.toFixed(2)}`
18
19    e.preventDefault() // evita envio do form
20 })
```

Figura 12 – JS do Exemplo (03)

- ***Operadores Aritméticos e Funções Matemáticas***
  - ***Exemplo (04):***
    - ***Elabore um programa para um restaurante que leia o preço por kg e o consumo (em gramas) de um cliente. Exibir o valor a ser pago***

Exemplo 04 | bkBank Academy

Arquivo | D:/Backups/EMPRES...

## Restaurante JS

Buffet por Quilo R\$:

Consumo do Cliente (gr):

**Valor a pagar R\$: 29.25**

Figura 11 – Programa Restaurante | dados fictícios

- *Operadores Aritméticos e Funções Matemáticas*
  - *Exemplo (04):*

```
aula2 > html > exemplo_4.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      <title>Exemplo 04 | bkBank Academy</title>
8  </head>
9  <body>
10     <h1> Restaurante JS </h1>
11     <form>
12         <p> Buffet por Quilo R$:
13         <input type="number" id="inQuilo" min="0" step="0.01" required>
14         </p>
15         <p> Consumo do Cliente (gr):
16         <input type="number" id="inConsumo" min="0" required>
17         </p>
18         <input type="submit" value="Calcular Preço">
19     </form>
20     <h3></h3>
21     <script src="../js/exemplo_4.js"></script>
22 </body>
23 </html>
```

Figura 12 – HTML do Exemplo (04)

- *Operadores Aritméticos e Funções Matemáticas*
  - *Exemplo (04):*

```
aula2 > js > JS exemplo_4.js > frm.addEventListener("submit") callback > [e] valor
1  // cria referência ao form e ao elemento h3 (onde será exibida a resposta)
2  const frm = document.querySelector("form")
3  const resp = document.querySelector("h3")
4
5  // cria um "ouvinte" de evento, acionado quando o botão submit for clicado
6  frm.addEventListener("submit", (e) => {
7      const quilo = Number(frm.inQuilo.value) // obtém o conteúdo dos campos
8      const consumo = Number(frm.inConsumo.value)
9
10     const valor = (quilo / 1000) * consumo // calcula valor a ser pago
11
12     resp.innerText = `Valor a pagar R$: ${valor.toFixed(2)}` // exibe resposta
13
14     e.preventDefault() // evita envio do form
15 })
```

Figura 13 – JS do Exemplo (04)



- **Observações**

- *Para criar um documento HTML, é necessário inserir algumas tags básicas que definem a estrutura da página. Compete ao HTML determinar o conteúdo e a semântica dos elementos que compõem um site*
- *A programa JS em página web é geralmente acionada a partir da ocorrência de um evento, como carregar a página, alterar um campo de formulário, pressionar uma determinada tecla. Um evento comumente utilizado para executar um programa é o **click** sobre o botão exibido na página*

- **Observações**

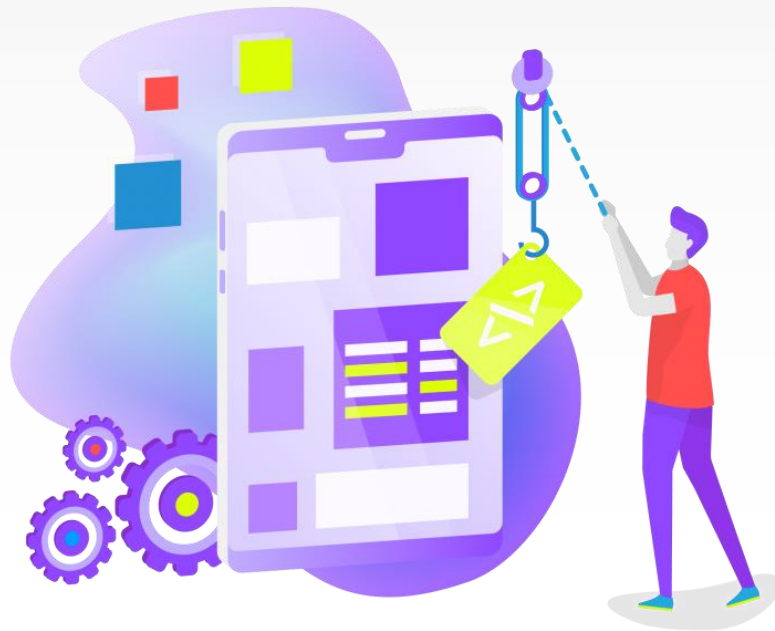
- *Os programas, por sua vez, devem ser criados contendo uma **função** declarada pelo **programador**. Assim, quando o **evento ocorre**, a **função** contendo um **conjunto de ações** a serem **executadas** é **acionada***
- *Para **referenciar um elemento HTML** da página, seja um **campo de formulário** ou um **parágrafo de texto**, é necessário **identifica-lo** no **código HTML** e, em seguida, **utilizar os métodos `querySelector()` ou `getElementById()`** no **programa JS***

- **Observações**

- ***A integração do programa JS com o documento HTML pode ocorrer a partir de três formas:***

- ***Inserir uma seção <script> no próprio documento HTML***
    - ***Criar um novo arquivo JS e usar as rotinas de tratamento de eventos DOM***
    - ***Criar um novo arquivo JS e registrar um ouvinte de evento, também chamado modelo de eventos DOM nível 2 (recomendado)***

- **Observações**
  - *Separar o código HTML do código JS é considerado uma boa prática de programação, pois HTML e JS têm papéis diferentes no processo de construção de um site*



# 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.

