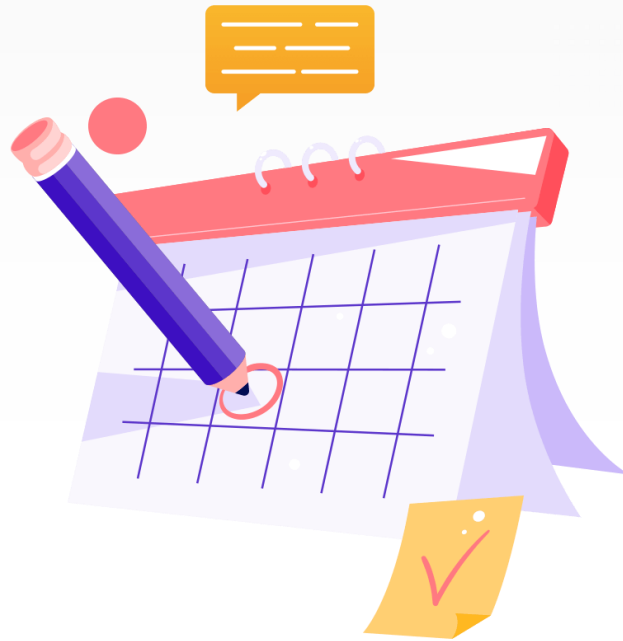


JavaScript

Módulo Básico





STRINGS E DATAS

Strings e Datas

- *Strings e Datas*

- Para realizar operações sobre *strings* e *datas*, as linguagens de programação dispõem de *métodos próprios*
- *Strings*: há métodos para obter cada uma das letras que compõem uma palavra, converter uma palavra para letras maiúsculas ou minúsculas ou, então, extrair partes de uma palavra
- *Datas*: há métodos para criar objetos do tipo *Date* e realizar cálculos sobre as datas, como adicionar ou subtrair dias, meses ou anos a uma data

Strings e Datas

- ***Strings e Datas***

- ***Exemplos de rotinas de programação que realizam operações sobre **cadeias** de **caracteres** e **datas**:***

- ***validar o preenchimento de um nome em um formulário de cadastro***
 - ***criar uma sugestão de e-mail com as iniciais de nome e sobrenome de um aluno***
 - ***criar um jogo em que o usuário deve acertar as letras que compõem uma palavra***
 - ***calcular a diferença de dias entre duas datas***
 - ***determinar a data de vencimento das parcelas de uma compra***
 - ***calcular a data limite para a obtenção de um determinado desconto***

Strings e Datas

- ***Strings e Datas***

- ***Exemplos de rotinas de programação que realizam operações sobre **cadeias** de **caracteres** e **datas**:***

- ***validação de senhas de usuários de um sistema também podem ser realizadas a partir dos métodos e propriedades de manipulação de string***



Strings e Datas

- **Strings e Datas**

- Com as **propriedades** e os **métodos** de manipulação de strings, é possível verificar se uma senha conta com um **número mínimo** de caracteres, se é **formada** por letras e números, se contém letras maiúsculas e minúsculas e, ainda, se utiliza algum **caractere especial** na sua composição




Strings e Datas

- ***Percorrer uma String***

- ***charAt(): método mais simples, retorna o caractere de uma posição da palavra. Similar aos vetores, a posição inicial da string é 0 (zero)***

```
const cidade = "Pelotas"
```

```
cidade.charAt(0)
cidade.charAt(1)
cidade.charAt(2)
...
```



Strings e Datas

- ***Percorrer uma String***

- Com a **propriedade** `length` o **método** `charAt()`, é possível, portanto, **percorrer todos os caracteres de uma string**
- Se quisermos **verificar**, por exemplo, **quantas palavras contém o texto de um anúncio**



Strings e Datas

- *Percorrer uma String*

- *Exemplo (01):*

- *Programa lê um anúncio e informa o número de palavras*

```
8      <title>Exemplo 01 | bkBank Academy</title>
9  </head>
10 <body>
11   <script>
12       const anuncio = prompt("Anúncio: ") // lê o anúncio
13       let numPalavras = 0 // inicializa contador
14       const tam = anuncio.length // obtém o tamanho
15       for (let i = 0; i < tam; i++) { // percorre os caracteres do anúncio
16           if (anuncio.charAt(i) == " ") { // se encontrou um espaço
17               numPalavras++ // incrementa contador
18           }
19       }
20       // exibe anúncio e número de palavras (que é o nº de espaços + 1)
21       alert(`Anúncio: ${anuncio}\nNº Palavras: ${numPalavras + 1}`)
22   </script>
23 </body>
24 </html>
```

Figura 1 – JS que utiliza o método charAt() para percorrer os caracteres de uma string

Strings e Datas

- ***Percorrer uma String***
 - ***Exemplo (01):***
 - ***Programa lê um anúncio e informa o número de palavras***

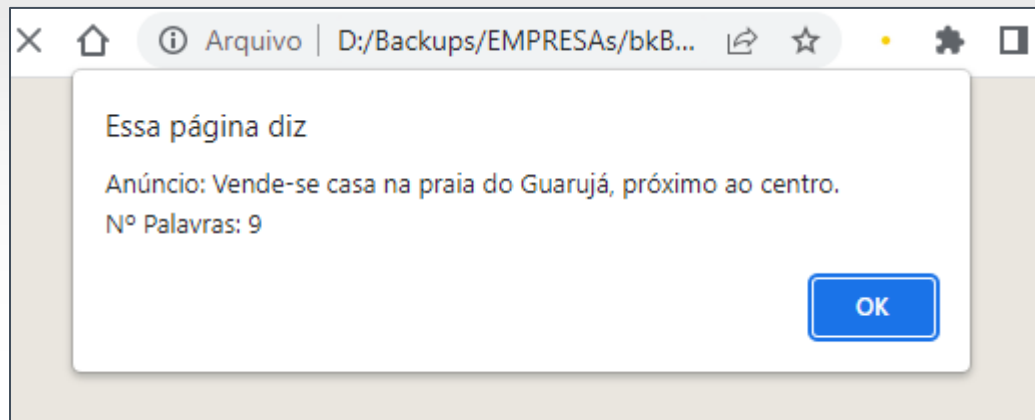


Figura 2 – Testando o método charAt()

Strings e Datas

- ***Percorrer uma String***

- ***Uma forma alternativa de percorrer um vetor é utilizando `for..of`***

```
let numPalavras = 0           // inicializa contador

for (const letra of anuncio) { // percorre os caracteres do anúncio
  if (letra == " ") {          // se encontrou um espaço...
    numPalavras++              // incrementa contador
  }
}
```

Strings e Datas

- ***Percorrer uma String***

- ***Crie uma pasta `css`, `img` e `js`***
- ***Crie um arquivo intitulado de `estilos.css` com o conteúdo abaixo, e salve dentro da pasta `css`:***

```
img { float: left; height: 300px; width: 300px; }  
img.exerc { float: left; height: 200px; width: 200px; }  
h1 { border-bottom-style: inset; }  
.entre-letras { letter-spacing: 0.5em; }  
.alinha-direita { text-align: right; }
```

Strings e Datas

- **Percorrer uma String**

- **Exemplo (02):**

- O programa "*Qual é a Fruta?*", deve ler uma **palavra** (*sugere-se uma fruta*) e **exibir**, após o **clique** no botão **Mostrar Dica**, a **letra inicial da fruta** e as **demaís ocorrências** dessa **letra** na **palavra**
 - As **outras letras** **não** devem ser **exibidas**, apenas um **sublinhado** (*underline*) "_" para **representar cada letra**
 - O **conteúdo** do **campo** de **entrada** deve ser **substituído** por **asteriscos**



Strings e Datas

- **Percorrer uma String**
 - **Exemplo (02):**
 - Programa "*Qual é a Fruta?*"



Figura 3 – Renderização do HTML | Exemplo 2

Strings e Datas

- *Percorrer uma String*
 - *Exemplo (02):*

```
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 02 | bkBank Academy</title>
9  </head>
10 <body>
11     
12     <h1> Programa Qual é a Fruta </h1>
13     <form>
14         <p> Fruta:
15             <input type="text" id="inFruta" autofocus required>
16             <input type="submit" value="Jogar">
17         </p>
18     </form>
19     <h3> Descubra: <span class="entre-letras"></span></h3>
20     <script src="../js/exemplo_2.js"></script>
21 </body>
22 </html>
```

Figura 4 – HTML do Exemplo 2

Strings e Datas

- *Percorrer uma String*
 - *Exemplo (02):*

```
1  const frm = document.querySelector("form") // obtém elementos da página
2  const resp = document.querySelector("span")
3
4  frm.addEventListener("submit", (e) => {      // "escuta" evento submit do form
5      e.preventDefault()                    // evita envio do form
6      const fruta = frm.inFruta.value.toUpperCase() // conteúdo do campo em maiúsculas
7      let resposta = ""                     // string que irá conter a resposta
8
9      for (const letra of fruta) {          // percorre todos os caracteres da fruta
10         if (letra == fruta.charAt(0)) {    // se letra igual a letra inicial da string...
11             resposta += fruta.charAt(0)    // adiciona a letra inicial
12         } else {                           // senão, ...
13             resposta += "_"               // adiciona o sublinhado
14         }
15     }
16
17     resp.innerText = resposta              // exibe a resposta
18     frm.inFruta.value = "*".repeat(fruta.length) // preenche com "*", conforme tamanho
19 }
```

Figura 5 – JS do Exemplo 2

Strings e Datas

- **Letras maiúsculas e minúsculas**
 - O uso dos métodos **toUpperCase()** e **toLowerCase()**, além de servir para **apresentar** uma **palavra** em **caixa alta** ou **baixa**, também é importante para **auxiliar** nas **condições envolvendo palavras ou letras**
 - As **linguagens diferenciam** as **letras maiúsculas** de suas **equivalentes minúsculas** em uma **comparação**
 - **Utilizar** esses **métodos** pode **simplificar** algumas **condições criadas** em um programa

Strings e Datas

- *Letras maiúsculas e minúsculas*

```
<script>
  while (true) {    // cria repetição
    // comandos ...
    const continua = prompt("Continuar (S/N)?") // lê uma entrada
    if (continua.toUpperCase() == "N") {        // converte em maiúscula
      break // sai da repetição
    }
  }
</script>
```

Strings e Datas

- ***Letras maiúsculas e minúsculas***

- ***Exemplo (03):***

- ***O programa deve ler uma palavra e exibi-la de forma invertida***
 - ***Um detalhe a ser observado, na inversão, o primeiro caractere deve ficar em letra maiúscula e os demais, em minúsculas***

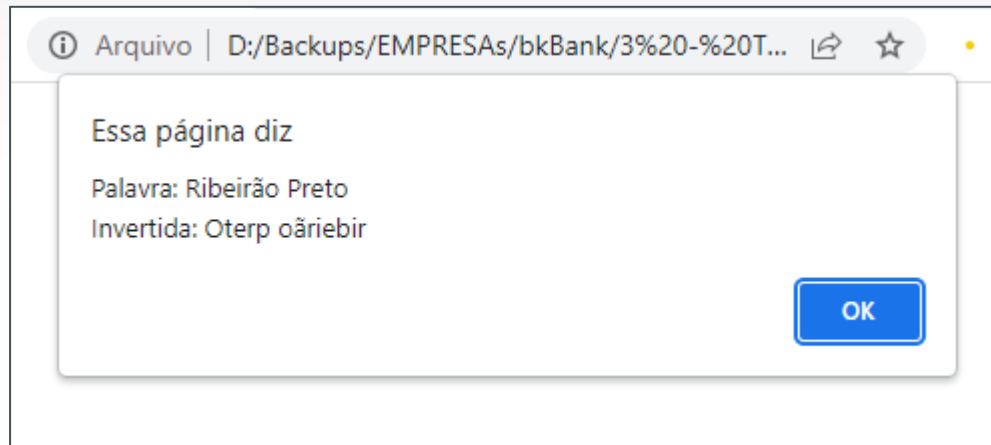


Figura 6 – Executando o Exemplo 3

Strings e Datas

- *Letras maiúsculas e minúsculas*
 - *Exemplo (03):*

```
10 <body>
11   <script>
12     const palavra = prompt("Palavra: ") // lê a palavra
13     const tam = palavra.length          // obtém o tamanho
14
15     // inverso inicia com a última letra da palavra em caixa alta
16     let inverso = palavra.charAt(tam - 1).toUpperCase()
17
18     // for decrescente percorre as demais letras e ...
19     for (let i = tam - 2; i >= 0; i--) {
20       inverso += palavra.charAt(i).toLowerCase() // converte-as em caixa baixa
21     }
22
23     // exibe palavra original e invertida
24     alert(`Palavra: ${palavra}\nInvertida: ${inverso}`)
25   </script>
26 </body>
```

Figura 7 – JS do Exemplo 3

Strings e Datas

- ***Manipulando Caracteres (cópia e remoção de espaços)***
 - O método ***substr()***, contém ***dois parâmetros: posição inicial da string e número de caracteres a serem copiados***
 - ***Caso apenas a posição inicial seja informada, todos os caracteres dessa posição até o final da string são copiados***

<script>

```
const palavra = "saladas"
```

```
const copia1 = palavra.substr(2)    // obtém "ladas"
```

```
const copia2 = palavra.substr(2, 4) // obtém "lada"
```

```
const copia3 = palavra.substr(0, palavra.length - 1) // obtém "salada"
```

```
const copia4 = palavra.substr(-2)   // obtém "as"
```

</script>

Strings e Datas

- **Localizar um ou mais caracteres na string**
 - Os métodos **indexOf()**, **lastIndexOf()** e **includes()** são utilizados para **pesquisar um conteúdo em uma lista de dados armazenada**
 - Também podem ser aplicados sobre um **string** e possuem a mesma finalidade: **localizar um caractere (ou mais caracteres) agora em uma string**
 - **indexOf()**: a pesquisa ocorre a partir do início da string
 - **lastIndexOf()**: a pesquisa ocorre da direita para a esquerda, caso o conteúdo **não** exista, o valor **-1** é retornado
 - **includes()**: retorna **true** ou **false** de acordo com a existência ou não dos caracteres

Strings e Datas

- *Localizar um ou mais caracteres na string*

```
<script>
```

```
const palavra = "saladas"
```

```
const posicao1 = palavra.indexOf("a")           // retorna 1
```

```
const posicao2 = palavra.lastIndexOf("a")       // retorna 5
```

```
const posicao3 = palavra.indexOf("sal")         // retorna 0
```

```
const posicao4 = palavra.indexOf("e")           // retorna -1
```

```
const existe = palavra.includes("d")           // retorna true
```

```
</script>
```

Strings e Datas

- **Localizar um ou mais caracteres na string**

- **Exemplo (04):**

- O programa “**Nome do Crachá**” deve ler o **nome completo** de um **participante** em um **evento** e **exibir apenas o seu nome e sobrenome**

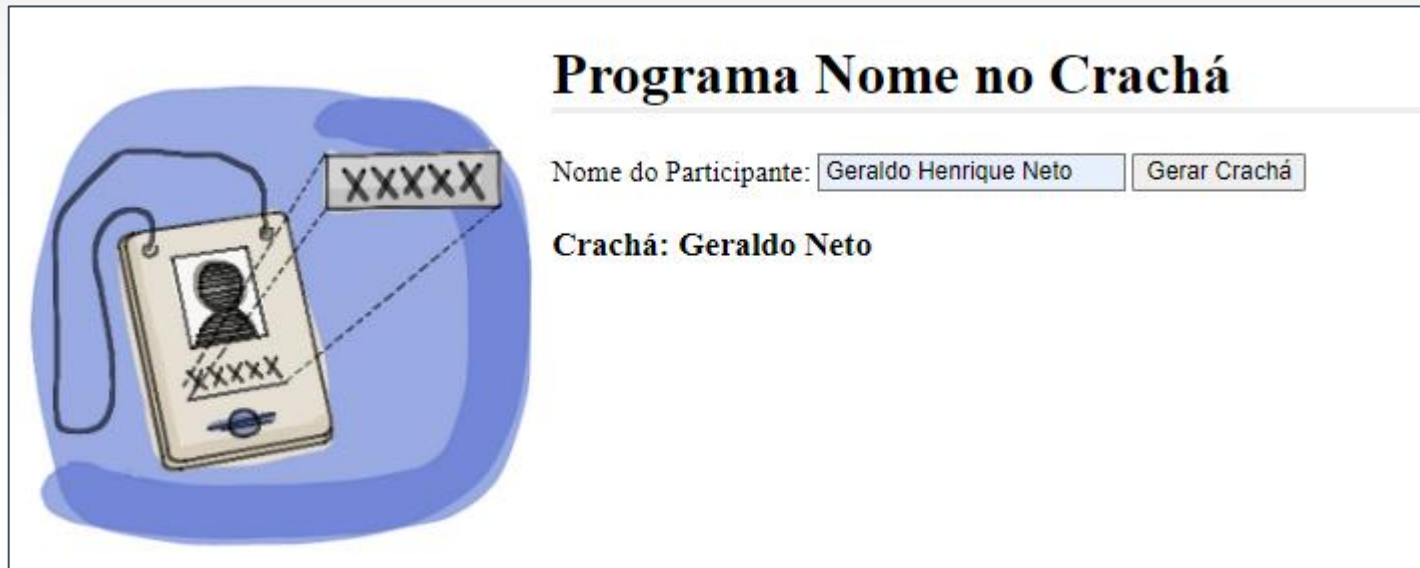


Figura 8 – Renderização do HTML | Exemplo 4

Strings e Datas

- *Localizar um ou mais caracteres na string*
 - *Exemplo (04):*

```
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 04 | bkBank Academy</title>
9  </head>
10 <body>
11     
12     <h1> Programa Nome no Crachá </h1>
13     <form>
14         <p> Nome do Participante:
15             <input type="text" id="inNome" autofocus required>
16             <input type="submit" value="Gerar Crachá">
17         </p>
18     </form>
19     <h3></h3>
20     <script src="../js/exemplo_3.js"></script>
21 </body>
22 </html>
```

Figura 9 – HTML do Exemplo 4

Strings e Datas

- *Localizar um ou mais caracteres na string*
 - *Exemplo (04):*

```
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    // obtém o nome informado e retira espaços em branco do início e final da string
7    const nome = frm.inNome.value.trim()
8
9    if (!nome.includes(" ")) { // se o nome não (!) possuir espaço
10      alert("Informe o nome completo...")
11      return
12    }
13
14    const priEspaco = nome.indexOf(" ") // posição do primeiro espaço
15    const ultEspaco = nome.lastIndexOf(" ") // posição do último espaço
16
17    // copia nome e sobrenome usando os parâmetros do substr()
18    const cracha = nome.substr(0, priEspaco) + nome.substr(ultEspaco)
19
20    resp.innerText = `Crachá: ${cracha}` // exibe a resposta
21  })
```

Figura 10 – JS do Exemplo 4
Aula 06 | Módulo Básico

Strings e Datas

- *Dividir a string em elementos de vetor*
 - ***split()**: converte a string em elemento de vetor a cada ocorrência de um determinado caractere*

```
const sabores = "calabresa, 4 queijos, atum, frango"  
const partes = sabores.split(",")
```

- *Como uma vírgula (",") foi passada como parâmetro para o método **split()**, o vetor **partes** conterá os seguintes elementos*

```
// partes[0] = "calabresa"  
// partes[1] = "4 queijos"  
// partes[2] = "atum"  
// partes[3] = "frango"
```

Strings e Datas

- **Dividir a string em elementos de vetor**
 - **Exemplo (05):**
 - Uma **empresa necessita** de um **programa** que **gere** um **e-mail institucional** para **todos** os seus **funcionários**
 - O **e-mail** deve ser **formado** pelas **letras iniciais** do **nome** do **funcionário** e de seu **sobrenome**, seguido pelo “@empresa.com.br”



Programa E-mail Institucional

Funcionário:

E-mail: **jgsampaio@empresa.com.br**

Figura 11 – Renderização do HTML | Exemplo 5

Strings e Datas

- *Dividir a string em elementos de vetor*
 - *Exemplo (05):*

```
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 05 | bkBank Academy</title>
9  </head>
10 <body>
11     
12     <h1> Programa E-mail Institucional </h1>
13     <form>
14         <p> Funcionário:
15             <input type="text" id="inFuncionario" autofocus required>
16             <input type="submit" value="Gerar E-mail">
17         </p>
18     </form>
19     <h3></h3>
20     <script src="../js/exemplo_4.js"></script>
21 </body>
22 </html>
```

Figura 12 – HTML | Exemplo 5
Aula 06 | Módulo Básico

Strings e Datas

- *Dividir a string em elementos de vetor*
 - *Exemplo (05):*

```
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 funcionario = frm.inFuncionario.value // obtém nome do funcionário
7
8      // divide o nome em itens de vetor, criados a cada ocorrência do espaço
9      const partes = funcionario.split(" ")
10     let email = "" // vai concatenar letras
11     const tam = partes.length // obtém nº de itens do vetor partes
12
13     for (let i = 0; i < tam - 1; i++) { // percorre itens do vetor (exceto o último)
14         email += partes[i].charAt(0) // e obtém a letra inicial de cada item
15     }
16
17     // concatena as letras iniciais com a última parte (sobrenome) e empresa
18     email += partes[tam - 1] + "@empresa.com.br"
19
20     resp.innerText = `E-mail: ${email.toLowerCase()}` // exibe e-mail em minúsculas
21 })
```

Figura 13 – JS | Exemplo 5
Aula 06 | Módulo Básico

Strings e Dados

- **Validar Senhas**

- **match()**: ótimo *método* para *implementar* uma *política* de *senhas* em um *sistema*
- *Permite verificar a existência de* **letras maiúsculas, minúsculas, números e símbolos** *em uma string*
- *Seu funcionamento utiliza o conceito de* **expressões regulares**
- *Uma* **expressão regular** *contém um conjunto de caracteres que indicam um padrão a ser pesquisado*

reg[ular]
expr[essio]n

Strings e Datas

- **Validar Senhas**

- **Sintaxe:**

- `const vetor = palavra.match(/[A-Z]/g)`

- `/[A-Z]/` é o **padrão** de **expressão regular** que se deseja encontrar na palavra
 - A opção **g** (**global**) indica que a pesquisa deve retornar **todas** as **ocorrências** dos caracteres na string
 - O retorno é um **vetor** contendo os **elementos encontrados** ou **null**, caso o **padrão não exista** na string fornecida

- ***Validar Senhas***

- ***Exemplos:***

```
<script>
  const palavra = "#SenhA_123!"
  const vetor1 = palavra.match(/[a-z]/g) // e, n, h
  const vetor2 = palavra.match(/[A-Z]/g) // S, A
  const vetor3 = palavra.match(/[0-9]/g) // 1, 2, 3
  const vetor4 = palavra.match(/\W|_/g) // #, _, !
  const vetor5 = palavra.match(/[T-Z]/g) // null
</script>
```

Strings e Dados

- **Validar Senhas**


- **Exemplo (06):**

- *Suponha que, para ser **válida**, uma **senha** deve possuir as seguintes regras de composição:*
 - *possuir entre 8 e 15 caracteres*
 - *possuir, no mínimo, 1 número*
 - *possuir, no mínimo, 1 letra minúscula*
 - *possuir, no mínimo, 2 letras maiúsculas*
 - *possuir, no mínimo, 1 símbolo*



Strings e Datas

- **Validar Senhas**
 - **Exemplo (06):**
 - **Programa para “Validar Senhas”**



Programa Valida Senha

Senha:

Erro... A senha deve possuir entre 8 e 15 caracteres, possuir letras maiúsculas (no mínimo, 2), possuir símbolos (no mínimo, 1)

Figura 14 – Renderização do HTML | Exemplo 6

Strings e Datas

- *Validar Senhas*
 - *Exemplo (06):*

```
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 Valida Senha </h1>
13     <form>
14         <p> Senha:
15             <input type="text" id="inSenha" required>
16             <input type="submit" value="Verifica Validade">
17         </p>
18     </form>
19     <h3></h3>
20     <script src="../js/exemplo_5.js"></script>
21 </body>
22 </html>
```

Figura 15 – HTML do Exemplo 6

Strings e Datas

- **Validar Senhas**
 - **Exemplo (06): - Parte 1**

```
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 senha = frm.inSenha.value // obtém senha informada pelo usuário
7    const erros = [] // vetor com erros
8
9    // verifica se o tamanho da senha é inválido
10   if (senha.length < 8 || senha.length > 15) {
11     erros.push("possuir entre 8 e 15 caracteres")
12   }
13
14   // verifica se não possui números
15   if (senha.match(/[0-9]/g) == null) {
16     erros.push("possuir números (no mínimo, 1)")
17   }
```

Figura 16 – JS do Exemplo 6 (Parte 1)

Strings e Datas

- *Validar Senhas*
 - *Exemplo (06): - Parte 2*

```
19 // verifica se não possui letras minúsculas
20 if (!senha.match(/[a-z]/g)) {
21     erros.push("possuir letras minúsculas (no mínimo, 1)")
22 }
23
24 // verifica se não possui letras maiúsculas ou se possui apenas 1
25 if (!senha.match(/[A-Z]/g) || senha.match(/[A-Z]/g).length == 1) {
26     erros.push("possuir letras maiúsculas (no mínimo, 2)")
27 }
28
29 // verifica se não possui símbolos ou "_"
30 if (!senha.match(/[\W|_]/g)) {
31     erros.push("possuir símbolos (no mínimo, 1)")
32 }
33
34 // se vetor está vazio (significa que não foram encontrados erros)
35 if (erros.length == 0) {
36     resp.innerText = "Ok! Senha Válida"
37 } else {
38     resp.innerText = `Erro... A senha deve ${erros.join(", ")}`
39 }
40 }
```

Figura 17 – JS do Exemplo 6 (Parte 2)

- **Substituição de Caracteres**
 - As **expressões regulares** são utilizadas como **parâmetros** do método **replace()**, quando desejarmos **substituir um caractere** (ou um **conjunto de caracteres**) por outro em uma **string**
 - **Por padrão, a substituição incide apenas sobre a primeira ocorrência do caractere na string**
 - Com o uso de uma **expressão regular**, com a opção **g** (**global**) indicada, a **troca ocorre em toda a string**
 - **Um detalhe importante é o de que a string mantém o seu conteúdo original**

Strings e Datas

- ***Substituição de Caracteres***

- ***Apenas a variável que recebe o retorno do método ou o conteúdo que é apresentado pelo programa contem as substituições dos caracteres indicados***

- ***Sintaxe:***

- `const novaStr = str.replace(caracterePesquisado, novoCaractere)`

- ***Considere a variável `senha`***

- `const senha = "ABACAD"`

- `const senha1 = senha.replace("A", "X") // XBACAD`

- `const senha2 = senha.replace(/A/g, "X") // XBXCXD`

Strings e Datas

- ***Substituição de Caracteres***
 - ***Também podemos utilizar o método `replace()` para retirar um caractere de uma string***

```
const app = "App Controle Financeiro"  
const app2 = app.replace(" ", "")           // AppControle Financeiro  
const app3 = app.replace(/ /g, "")           // AppContoleFinanceiro  
const app4 = app.replace(/ /g, "").toLowerCase() // appcontrolefinanceiro
```

Strings e Datas

- ***Manipulação de Datas***

- ***Declarar um objeto que recebe uma instância do objeto Date***

- ```
const hoje = new Date()
```

- ***Existe diversos métodos específicos para a manipulação de datas***

- ***Alguns são utilizados para extrair partes da data, outros, para modificar as partes que compõem uma data***

- ***Também é possível realizar operações matemáticas envolvendo datas***

# Strings e Datas

- ***Manipulação de Datas***

- Os métodos `getDate()`, `getMonth()` e `getFullYear()` são utilizados para obter, respectivamente, o dia, o mês e o ano de uma data
- Os métodos `setDate()`, `setMonth()` e `setFullYear()` permitem alterar o dia, o mês e o ano da data

```
<script>
```

```
const hoje = new Date()
```

```
const amanha = new Date()
```

```
const dia = amanha.getDate()
```

```
amanha.setDate(dia + 1)
```

```
console.log(`Hoje: ${hoje}\nAmanhã: ${amanha}`)
```

```
</script>
```

# Strings e Datas

- ***Manipulação de Datas***
  - ***Exemplo (07):***
    - ***Calcula ano de nascimento de uma pessoa***

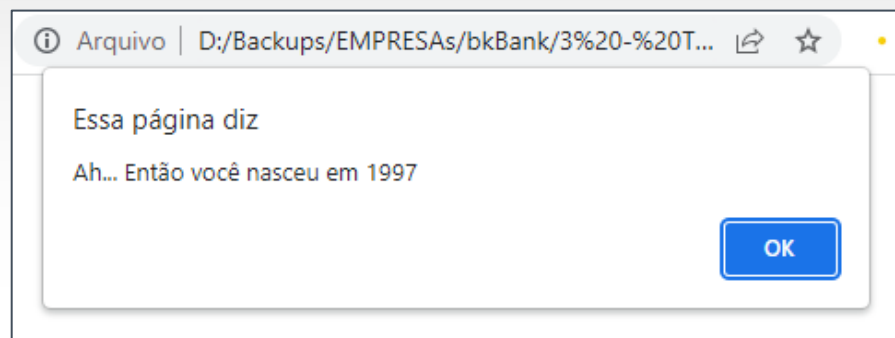
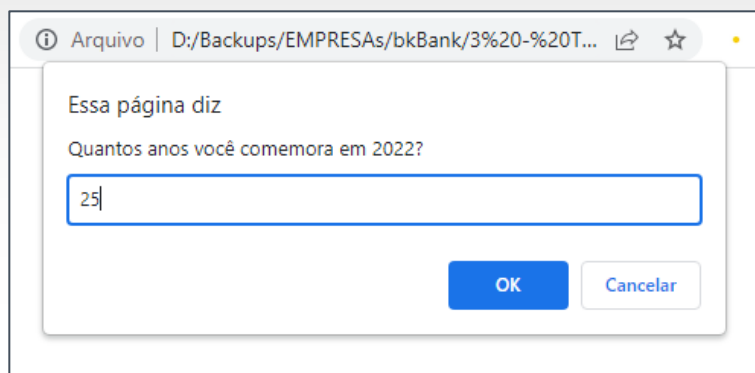


Figura 18 – Renderização do HTML | Exemplo 7

# Strings e Datas

- *Manipulação de Datas*
  - *Exemplo (07):*
    - *Calcula ano de nascimento de uma pessoa*

```
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>
10 <body>
11 <script>
12 const anoAtual = new Date().getFullYear()
13 const idade = prompt(`Quantos anos você comemora em ${anoAtual}?`)
14 const anoNasc = anoAtual - idade
15 alert(`Ah... Então você nasceu em ${anoNasc}`)
16 </script>
17 </body>
18 </html>
```

Figura 19 – JS do Exemplo 7

# Strings e Datas

- **Manipulação de Datas**

- **Exemplo (07):**


- Para calcular a **diferença** de dias entre **duas datas**, é necessário entender uma importante particularidade da linguagem JS em relação aos objetos **Date**: as datas JS são armazenadas internamente como um valor numérico
    - Uma data também pode ser criada ou calculada a partir de um número expresso em milissegundos, a contar do dia 1 de janeiro de 1970
    - Se quisermos obter a **diferença** de dias entre **duas datas**, podemos subtrair as datas e dividir o valor por **86400000**, que é o número de milissegundos em um dia: **24** horas x **60** minutos x **1000** milissegundos

# Strings e Datas

- **Manipulação de Datas**

- **Exemplo (08):**

- **O programa lê uma *data* de vencimento e o valor de uma conta**
    - **Caso a conta esteja em atraso, o programa deve calcular o valor da multa e dos juros a serem acrescentados ao valor total**



## Programa Caixa da Loja

Data de Vencimento:

Valor da Conta R\$:

Valor da Multa R\$:

Valor do Juros R\$:

Total a Pagar R\$...:

Figura 20 – Renderização do HTML | Exemplo 8

# Strings e Datas

- *Manipulação de Datas*
  - *Exemplo (08): - Parte 1*

```
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>
11
12 <h1> Programa Caixa da Loja </h1>
13 <form>
14 <p>Data de Vencimento:
15 <input type="date" id="inDataVenc" autofocus required>
16 </p>
17 <p>Valor da Conta R$:
18 <input type="number" id="inValor" min=0 step="0.01" class="alinha-direita" required>
19 <input type="submit" value="Calcular Multa e Juros">
20 </p>
```

Figura 21 – HTML | Exemplo 8 – Parte 1



# Strings e Datas

- *Manipulação de Datas*
  - *Exemplo (08): - Parte 2*

```
21 <p>Valor da Multa R$:
22 <input type="number" id="outMulta" class="alinha-direita" readonly>
23 </p>
24 <p>Valor do Juros R$.:
25 <input type="number" id="outJuros" class="alinha-direita" readonly>
26 </p>
27 <p>Total a Pagar R$...:
28 <input type="number" id="outTotal" class="alinha-direita" readonly>
29 <input type="reset" value="Nova Conta">
30 </p>
31 </form>
32 <script src="../../js/exemplo_6.js"></script>
33 </body>
34 </html>
```

Figura 22 – HTML | Exemplo 8 – Parte 2

# Strings e Datas

- *Manipulação de Datas*
  - *Exemplo (08): - Parte 1*

```
1 const frm = document.querySelector("form") // obtém elementos da página
2 const resp = document.querySelector("h3")
3
4 const TAXA_MULTA = 2 / 100 // multa por atraso
5 const TAXA_JUROS = 0.33 / 100 // juros por dia de atraso
6
7 frm.addEventListener("submit", (e) => {
8 e.preventDefault() // evita envio do form
9 const dataVenc = frm.inDataVenc.value
10 const valor = Number(frm.inValor.value)
11 const hoje = new Date() // cria variáveis (instancia objetos)
12 const vencto = new Date() // do tipo Date()
13
14 const partes = dataVenc.split("-") // data vem no formato aaaa-mm-dd
15 vencto.setDate(Number(partes[2]))
16 vencto.setMonth(Number(partes[1]) - 1)
17 vencto.setFullYear(Number(partes[0]))
```

Figura 23 – JS | Exemplo 8 – Parte 1

# Strings e Datas

- *Manipulação de Datas*
  - *Exemplo (08): - Parte 2*

```
19 const atraso = hoje - vencido // calcula a diferença de dias entre datas (em ms)
20 let multa = 0 // inicializa multa e juros com 0
21 let juros = 0
22
23 if (atraso > 0) { // se conta estiver em atraso ...
24 // converte ms do atraso em dias (1 dia = 24h x 60min x 60seg x 1000ms: 86400000)
25 const dias = atraso / 86400000
26 multa = valor * TAXA_MULTA // calcula multa e juros
27 juros = valor * TAXA_JUROS * dias
28 }
29 const total = valor + multa + juros // calcula o total
30
31 frm.outMulta.value = multa.toFixed(2) // exibe os valores com 2 decimais
32 frm.outJuros.value = juros.toFixed(2)
33 frm.outTotal.value = total.toFixed(2)
34 }
```

Figura 24 – JS | Exemplo 8 – Parte 2

# Strings e Datas

- **Manipulação de Datas**

- *Um cuidado especial com a manipulação de datas em JS é necessário quando quisermos exibir o mês de uma data no formato `dd/mm/aaaa`*
- *A variação do mês na linguagem JS se encontra entre o intervalo de `0` e `11`*
- *Para o dia e o ano, não há esse problema*

```
<script>
 const hoje = new Date()
 const dia = hoje.getDate()
 const mes = hoje.getMonth()
 const ano = hoje.getFullYear()
 console.log(hoje)
 console.log(`Data: ${dia}/${mes}/${ano}`)
</script>
```

Wed Jan 05 2022 21:52:58 GMT-0300 (Horário Padrão de Brasília)  
Data: 5/0/2022



# Strings e Datat

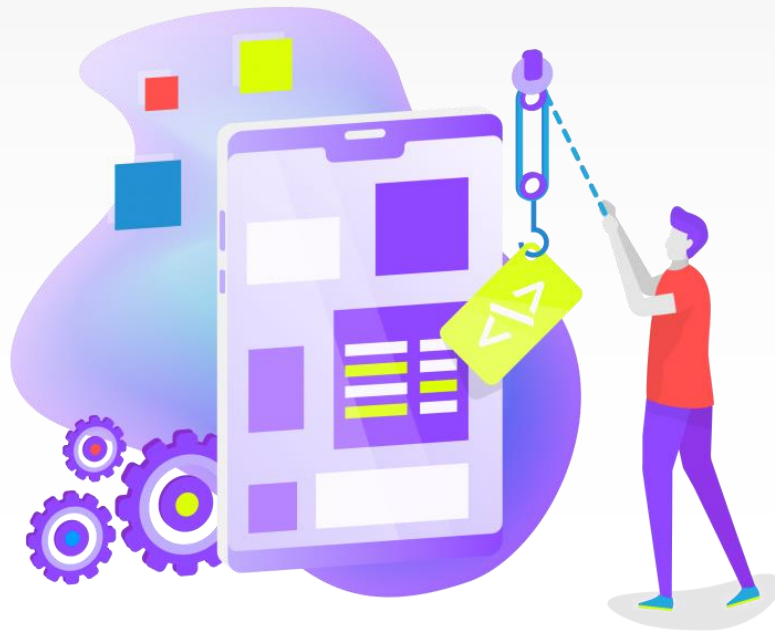
- **Manipulação de Datat**

- É possível **converter o número para string** e utilizar o método **padStart()** em uma mesma linha (**exibir a data atual no formato dd/mm/yyyy** com os **dois dígitos para dia e mês**)

**<script>**

```
const hoje = new Date()
const dia = hoje.getDate()
const mes = hoje.getMonth() + 1
const ano = hoje.getFullYear()
const dia2 = dia.toString().padStart(2, "0")
const mes2 = mes.toString().padStart(2, "0")
console.log(`Data: ${dia2}/${mes2}/${ano}`)
```

**</script>**



# 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.

