Sobre Este Curso

Público Alvo

Qualquer pessoa que deseja aprender sobre Lógica de Programação.

Pré-Requisitos

Não há.

Desenvolvido por DJF Treinamentos e Consultoria e licenciado para
Apex Treinamentos de Alta Performance

Índice

Anotações

Anotações

Copyright

Equipe

Conteúdos

- Felipe de Oliveira

Diagramação

- Fernanda Pereira

Revisão

- Fernanda Pereira

Histórico das Edições

Edição	Idioma	Edição
1ª	Português	Janeiro de 2018

Anotações

Capítulo 01 – Introdução à Lógica de Programação

Objetivos:

Neste capítulo você irá aprender:

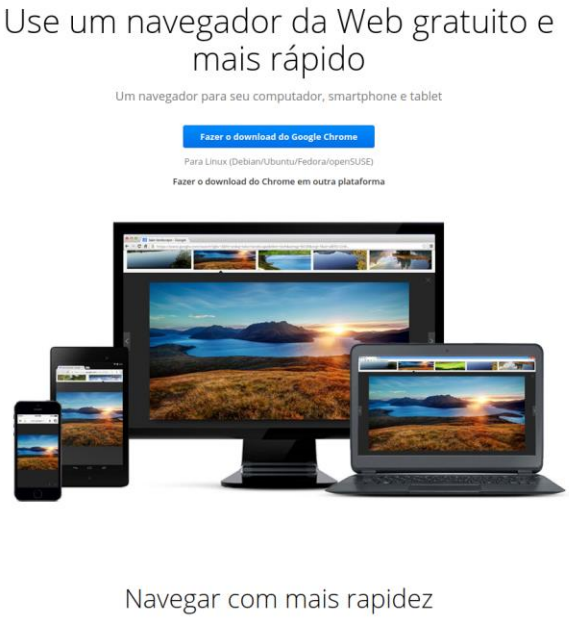
- Preparando o ambiente
- O que é algoritmo
- Html básico
- Criando o seu primeiro script

Anotações	

Preparando o ambiente

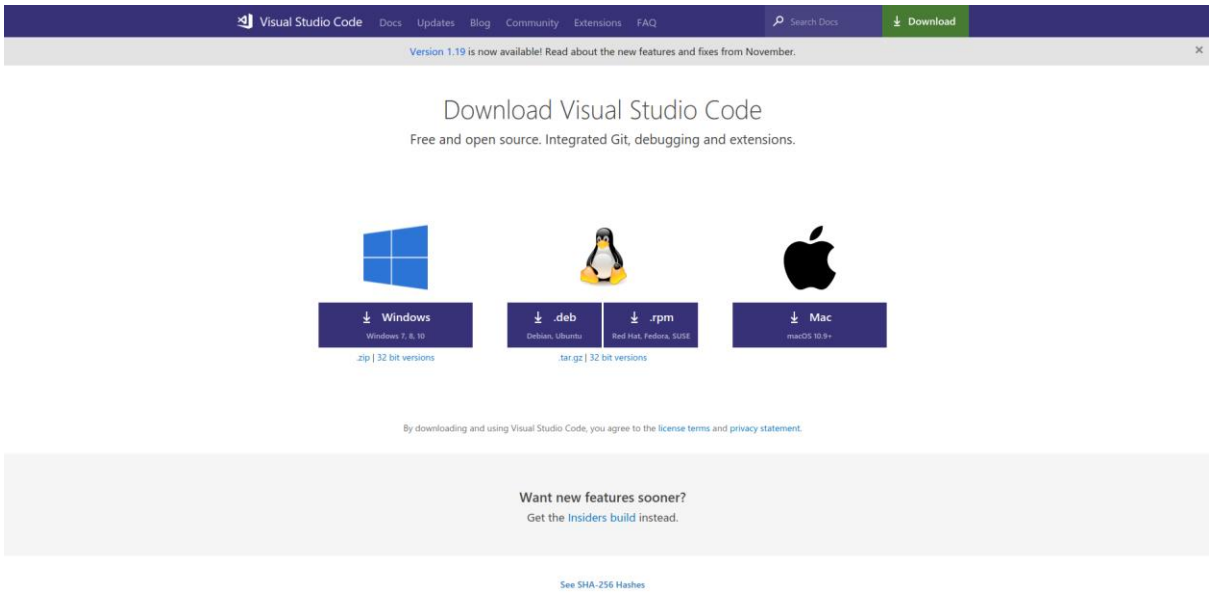
Neste treinamento iremos utilizar o navegador Google Chrome e o editor de textos Microsoft Visual Code.

Para realizar a instalação do Google Chrome acesse a site https://www.google.com.br/chrome/browser/desktop/index.html e clique no botão fazer o download do Google chrome conforme imagem abaixo.



Para realizar a instalação do editor Microsoft Visual Code acesse o link https://code.visualstudio.com/download e selecione a versão desejada conforme o sistema operacional instalado em seu computador.

	Anotações



O que é Algoritmo

O dicionário Michaelis nos traz as seguintes definições para a palavra algoritmo:

algoritmo

al·go·rit·mo

sm

1 MAT, OBSOL Sistema de notação aritmética com algarismos arábicos.

2 MAT Processo de cálculo que, por meio de uma sequência finita de regras, raciocínios e operações, aplicada a um número finito de dados, leva à resolução de grupos análogos de problemas.

3 MAT Operação ou processo de cálculo; sequência de etapas articuladas que produz a solução de um problema; procedimento sequenciado que leva ao cumprimento de uma tarefa.

4 LÓG Conjunto das regras de operação (conjunto de raciocínios) cuja aplicação permite resolver um problema enunciado por meio de um número finito de operações; pode ser traduzido em um programa executado por um computador, detectável nos mecanismos gramaticais de uma língua ou no sistema de procedimentos racionais finito, utilizado em outras ciências, para resolução de problemas semelhantes.

5 INFORM Conjunto de regras e operações e procedimentos, definidos e ordenados usados na solução de um problema, ou de classe de problemas, em um número finito de etapas.

Anotações	

Realizando uma reflexão sobre os termos apresentados podemos verificar que Algoritmo está ligado a execução de uma sequência de passos finitos que ao serem executados nos auxiliam na resolução de um determinado problema.

O que iremos aprender neste treinamento é como organizar o nosso pensamento de forma lógica e racional para determinarmos a sequência de passos correta para a resolução de um problema qualquer.

HTML Básico

Neste treinamento, visando um melhor aproveitamento para os alunos, iremos trabalhar os conceitos de Lógica de Programação inserindo os alunos dentro do contexto de documentos e aplicações web. Para facilitar esse tipo de aprendizado apresentamos abaixo alguns conceitos básicos da sintaxe de marcação de documentos HTML (HyperText Markup Language).

O exemplo abaixo apresenta um documento html com as principais tags que utilizaremos no decorrer do nosso treinamento.

```
1  <!DOCTYPE html>
2  <html lang="pt_BR">
3
4  <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=<device-width>,
   initial-scale=1.0">
7     <title>Exemplo documento Html</title>
8  </head>
9
10 <body>
11
12    <h1>Bem vindo a WEB</h1>
13    <h1>Tipos de títulos (cabeçalhos)</h1>
14    <h1>h1 Título Principal</h1>
15    <h2>h2 Título Secundário</h2>
16    <h3>h3 Título Secundário com menor ênfase</h3>
17    <h4>h4 Utilizado como subtítulo</h4>
18    <h5>h5 Utilizado como subtítulo</h5>
19    <h6>h6 Utilizado como subtítulo</h6>
20
21    <h1>Parágrafos &lt;p&gt;&lt;/p&gt;</h1>
22    <p>Primeiro parágrafo</p>
23    <p>Segundo  parágrafo. Aprender HTML é simples, basta
   conhecer o significado das tags de marcação e tudo vai dar
```

Anotações

```
    certo!</p>
24      <p>Outro parágrafo</p>
25
26      <h1>Lista não ordenada &lt;ul&gt;
    &lt;li&gt;&lt;/li&gt;&lt;/ul&gt;</h1>
27
28      <ul>
29          <li>Maça</li>
30          <li>Goiaba</li>
31          <li>Laranja</li>
32          <li>Mamão</li>
33          <li>Abacaxi</li>
34      </ul>
35
36      <h1>Lista  ordenada &lt;ol&gt;
    &lt;li&gt;&lt;/li&gt;&lt;/ol&gt;</h1>
37
38      <ol>
39          <li>Maça</li>
40          <li>Goiaba</li>
41          <li>Laranja</li>
42          <li>Mamão</li>
43          <li>Abacaxi</li>
44      </ol>
45
46      <h1>Tabela &lt;table&gt;&lt;/table&gt;</h1>
47      <table>
48          <thead>
49              <tr>
50                  <th>código</th>
51                  <th>descrição</th>
52                  <th>valor</th>
53              </tr>
54          </thead>
55          <tbody>
56              <tr>
57                  <td>000001</td>
58                  <td>Televisor 32 pol. smart LG</td>
59                  <td>R$1300,00</td>
60              </tr>
61              <tr>
62                  <td>000002</td>
63                  <td>Cadeira de praia Mor</td>
64                  <td>R$39,90</td>
65              </tr>
66              <tr>
67                  <td>000003</td>
68                  <td>Playstation 4 Sony com 2 controles + jogo
```

Anotações

```
    Fifa 2018</td>
69                  <td>R$1390,90</td>
70              </tr>
71          </tbody>
72      </table>
73
74      <h1>Formulário &lt;form&gt;&lt;/form&gt;</h1>
75      <form>
76          <fieldset>
77              <legend>Informações do Candidato</legend>
78              <div>
79                  <label>Nome Completo</label><br>
80                  <input type="text"><br>
81              </div>
82              <div>
83                  <label>Email</label><br>
84                  <input type="email"><br>
85              </div>
86              <div>
87                  <label>Gênero</label><br>
88                  <input type="radio"
    name="genero">Masculino<br>
89                  <input type="radio" name="genero">Feminino<br>
90              </div>
91              <div>
92                  <label>Benefícios</label><br>
93                  <input type="checkbox" >Vale transporte<br>
94                  <input type="checkbox" >Vale refeição<br>
95                  <input type="checkbox" >Plano de saúde<br>
96                  <input type="checkbox" >Plano odontológico<br>
97              </div>
98              <button type="submit">Salvar</button>
99
100         </fieldset>
101     </form>
102
103     <script>
104         alert('Alerta, executando código javascript')
105     </script>
106 </body>
107 </html>
```

Anotações

Doctype

Todo documento html deve começar pela tag doctype, essa tag indica aos browsers qual é a versão do html utilizado no documento.

HTML

Todo o restante do documento html deve estar envolvido dentro da tag html. Esta tag por sua vez possui duas seções principais head e body.

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <head>
4     <title>Documento base</title>
5  </head>
6  <body>
7  </body>
8  </html>
```

Head

Dentro do head (cabeçalho do documento) colocamos as meta informações do documento. Estas meta informações são informações não direcionadas ao usuário final (leitor do documento), mas são utilizadas pelo browser para configurar como as informações devem ser processadas e apresentadas para o usuário.

Body

A tag body (corpo) contém internamente todo o conteúdo que deve ser apresentado ao usuário final. Nela são colocadas todas as informações do documento propriamente dito.

Cabeçalhos h1-h6

Anotações	

As tags h1,h2,h3,h4,h5 e h6 são tags utilizadas para a marcação de cabeçalhos em diferentes níveis de tamanho.

```
1 <h1>h1 Título Principal</h1>
2 <h2>h2 Título Secundário</h2>
3 <h3>h3 Título Secundário com menor ênfase</h3>
4 <h4>h4 Utilizado como subtítulo</h4>
5 <h5>h5 Utilizado como subtítulo</h5>
6 <h6>h6 Utilizado como subtítulo</h6>
```

Parágrafos

Para realizarmos a marcação de um parágrafo dentro do documento html devemos utilizar a tag p. Para o html um parágrafo indica que o texto inserido dentro da tag deve ser apresentado em uma nova linha, por isso a tag p adiciona uma nova linha antes e após o conteúdo dentro dela.

```
1 <p>Primeiro parágrafo</p>
2 <p>Segundo  parágrafo. Aprender HTML é simples, basta
  conhecer o significado das tags de marcação e tudo vai dar
  certo!</p>
3 <p>Outro parágrafo</p>
```

Listas não ordenadas

Listas não ordenadas são marcadas através da tag ul. Cada item da lista deve ser incluído através da tag li.

```
1 <ul>
```

Anotações

```
2     <li>Maçã</li>
3     <li>Goiaba</li>
4     <li>Laranja</li>
5     <li>Mamão</li>
6     <li>Abacaxi</li>
7 </ul>
```

Listas Ordenadas

Listas ordenadas são marcadas através da tag ol. Cada item da lista também deve ser incluído através da tag li.

```
1 <ol>
2     <li>Maçã</li>
3     <li>Goiaba</li>
4     <li>Laranja</li>
5     <li>Mamão</li>
6     <li>Abacaxi</li>
7 </ol>
```

Tabelas

Anotações	

A marcação de tabelas no html é feita através da tag table. Uma tabela pode conter um cabeçalho demarcado através da tag thead, um corpo, demarcado através da tag tbody, e um rodapé, demarcado pela tag tfoot.

As linhas tabela são demarcadas através da tag tr. Já as colunas utilizadas no cabeçalho devem ser demarcadas com a tag th, e as colunas utilizadas no corpo da tabela devem ser demarcadas com a tag td.

```
<table>
  <thead>
      <tr>
          <th>código</th>
          <th>descrição</th>
          <th>valor</th>
      </tr>
  </thead>
  <tbody>
      <tr>
          <td>000001</td>
          <td>Televisor 32 pol. smart LG</td>
          <td>R$1300,00</td>
      </tr>
      <tr>
          <td>000002</td>
          <td>Cadeira de praia Mor</td>
          <td>R$39,90</td>
      </tr>
      <tr>
          <td>000003</td>
          <td>Playstation 4 Sony com 2 controles + jogo Fifa 2018</td>
          <td>R$1390,90</td>
      </tr>
  </tbody>
</table>
```

Formulários

Todo formulário é demarcado com a tag form. Dentro do formulário normalmente adicionamos rótulos através da tag label, e caixas de entrada de texto através da tag input. Também podemos adicionar botões através da tag button.

Anotações

A tag input pode representar diferentes tipos de caixas de entrada, para isso devemos utilizar a propriedade type da tag input para indicar qual o tipo que queremos utilizar. Alguns valores possíveis para a propriedade type são: text, email, checkbox, radio, password, entre outros.

Para dividir um formulário em seções podemos utilizar a tag fielset. Para adicionarmos uma legenda para cada seção através da tag legend.

```
1  <form>
2          <fieldset>
3              <legend>Informações do Candidato</legend>
4              <div>
5                  <label>Nome Completo</label><br>
6                  <input type="text"><br>
7              </div>
8              <div>
9                  <label>Email</label><br>
10                 <input type="email"><br>
11             </div>
12             <div>
13                 <label>Gênero</label><br>
14                 <input type="radio" name="genero">Masculino<br>
15                 <input type="radio" name="genero">Feminino<br>
16             </div>
17             <div>
18                 <label>Benefícios</label><br>
19                 <input type="checkbox" >Vale transporte<br>
20                 <input type="checkbox" >Vale refeição<br>
21                 <input type="checkbox" >Plano de saúde<br>
22                 <input type="checkbox" >Plano odontológico<br>
```

Anotações

```
23              </div>
24              <button type="submit">Salvar</button>
25
26          </fieldset>
27      </form>
```

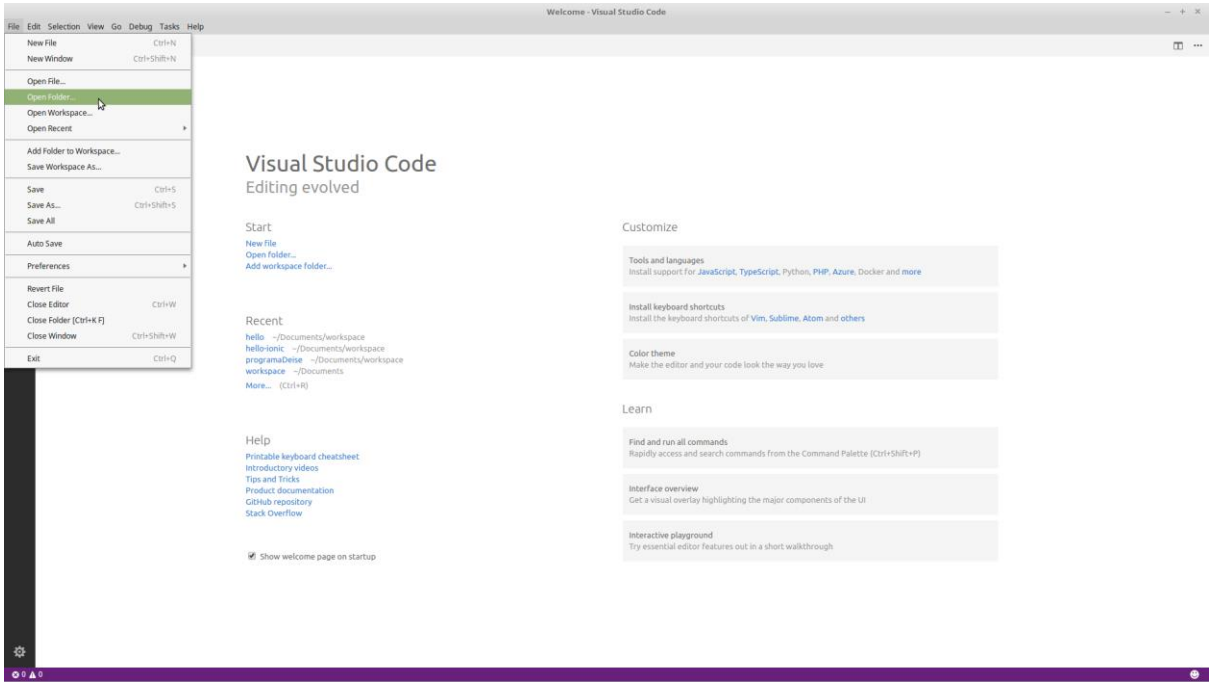
Criando seu primeiro Script

Antes de criarmos o nosso primeiro script, devemos criar uma estrutura de projeto onde iremos criar e manter nossos arquivos de script. Para isso abra o Microsoft Visual Code e crie uma nova pasta de trabalho.

As imagens abaixo apresentam a criação de uma pasta chamada logica-aula01 e dentro dessa pasta iremos criar nosso primeiro script.

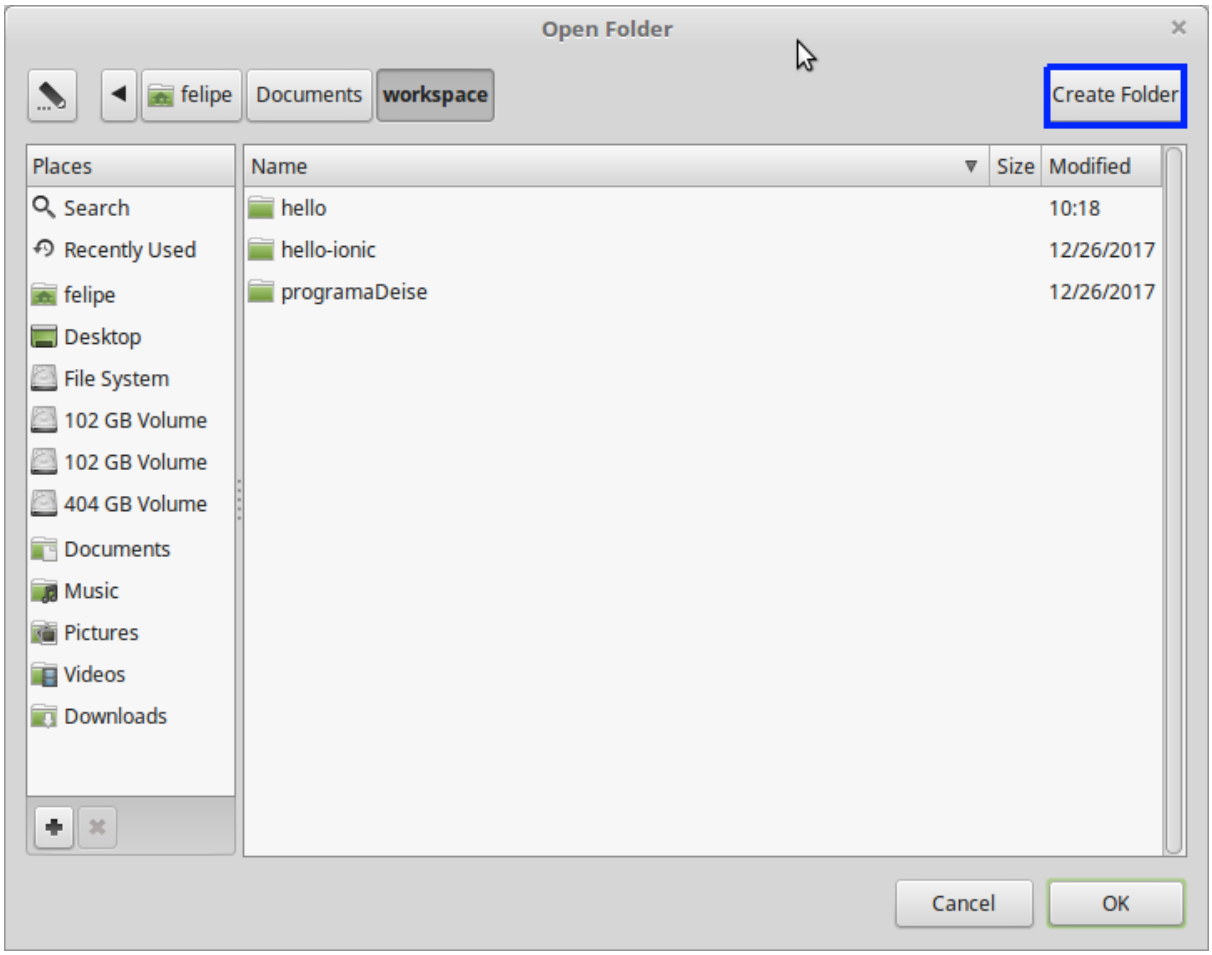
Selecione o menu File e escolha a opção Open Folder

Anotações



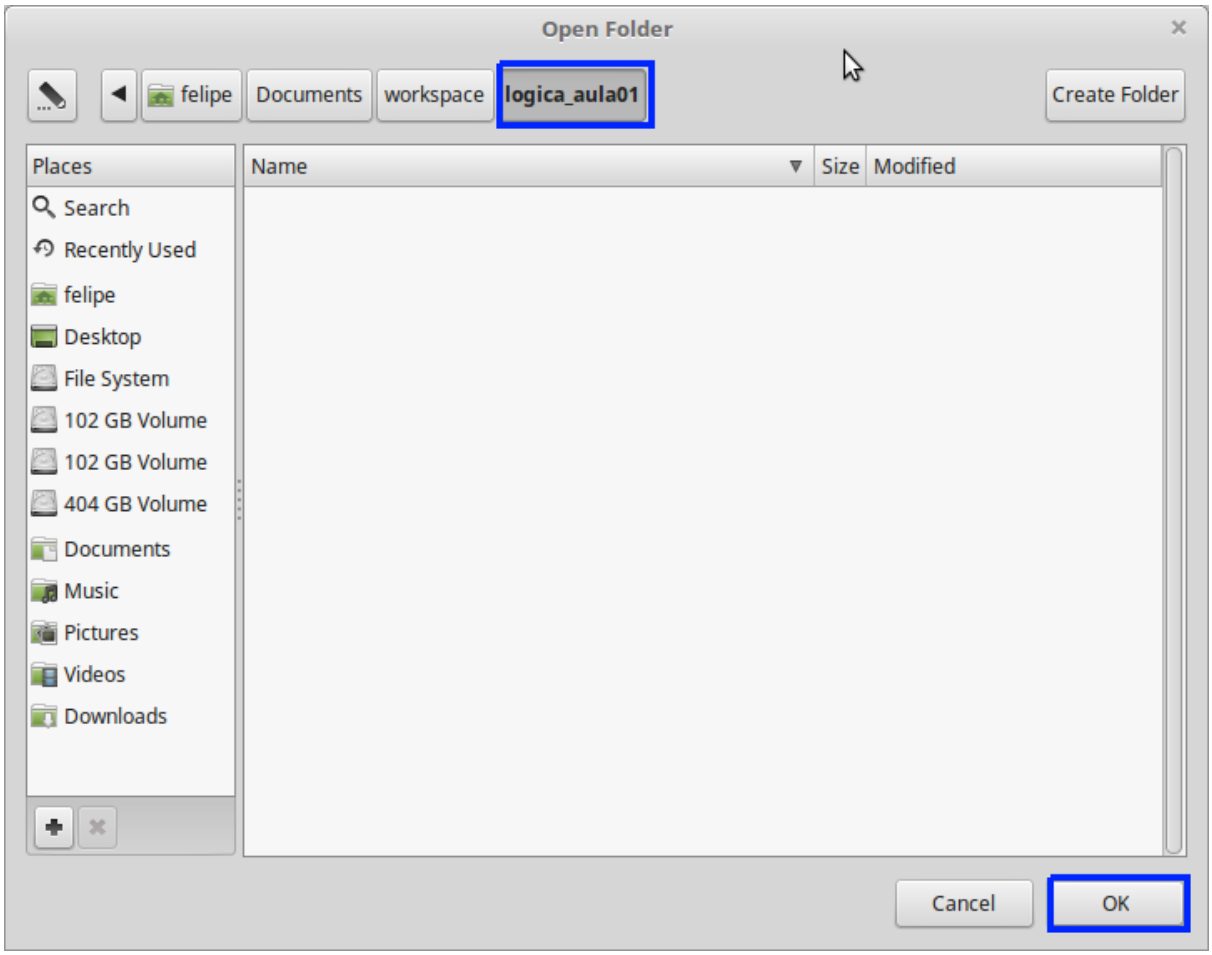
Na tela que se abre escolha o local onde serão armazenados os projetos e em seguida crie uma nova pasta com o nome logica_aula01 através do botão Create Folder.

Anotações	



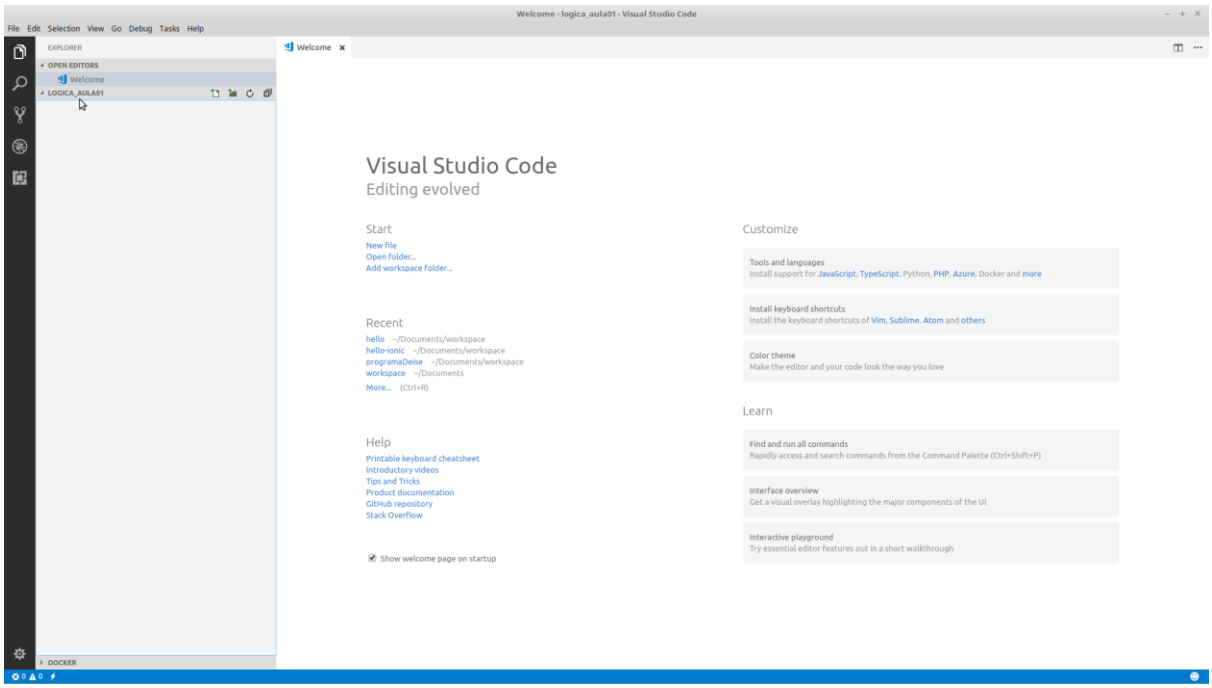
Com a pasta já selecionada clique no botão ok para confirmar a seleção da pasta.

Anotações

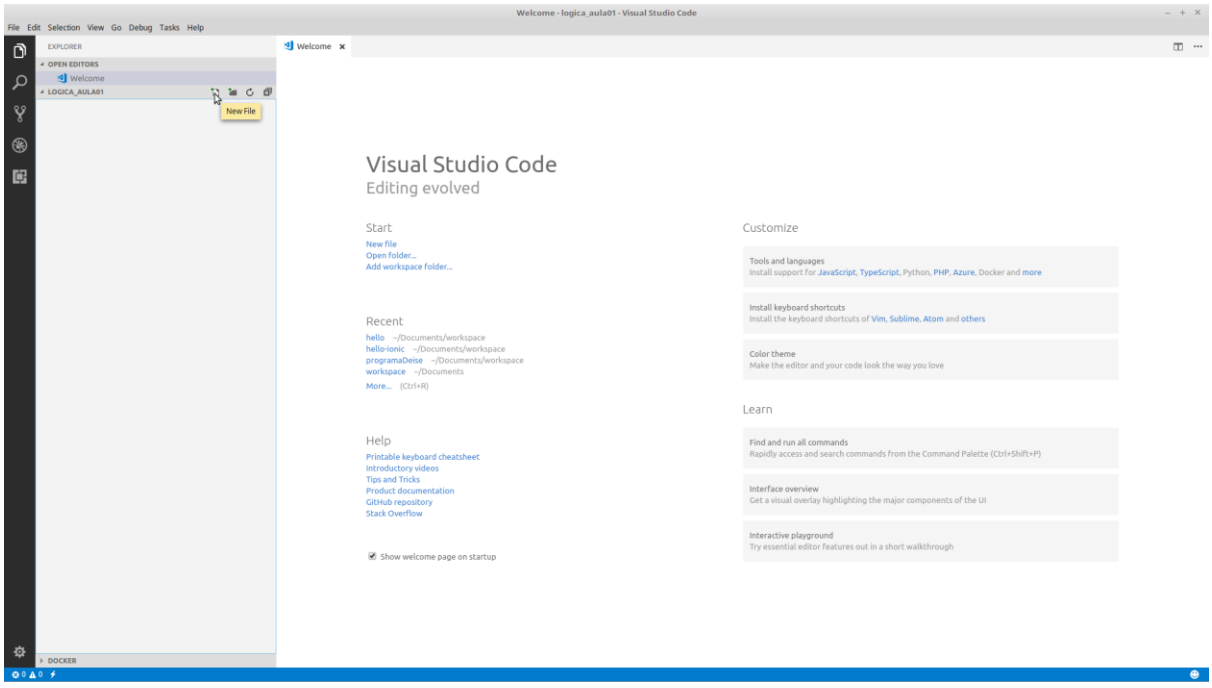


Após a seleção da pasta o Visual code deve estar conforme a imagem abaixo:

Anotações	

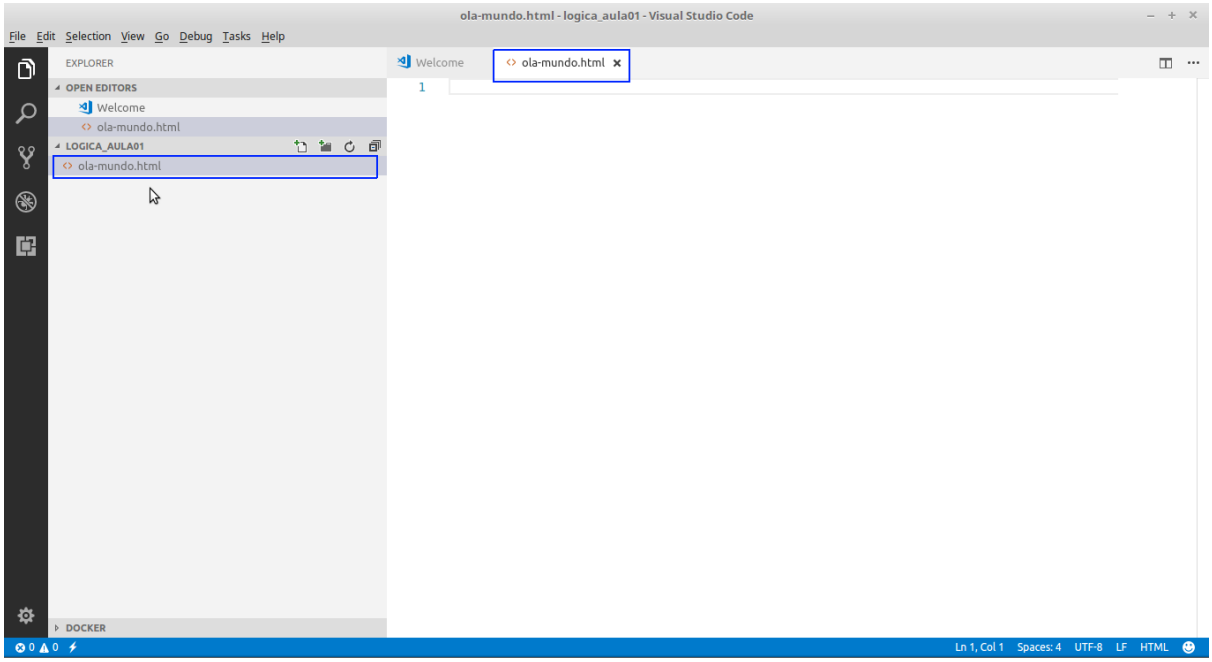


Para criar o nosso primeiro script selecione a opção new file conforme imagem abaixo.



Em seguida nomeie o arquivo como ola-mundo.html.

	Anotações



No arquivo criado vamos adicionar o seguinte código:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <body>
4     <script>
5        document.write('<h1>Bem vindo ao mundo da
   programação!</h1>');
6        document.write('<p>Aprendendo Lógica com Javascript e
   HTML</p>');
7     </script>
8  </body>
9  </html>
```

Entre as linhas 4 e 7 criamos um script utilizando a tag de mesmo nome. É dentro desta tag que escreveremos nossos códigos.

Nas linhas 5 e 6 executamos uma chamada para o comando document.write e passamos como argumento para esse comando o texto que queremos que seja apresentado na tela.

Anotações	

	Anotações

Capítulo 02 – Começando a Jornada

Objetivos:

Neste capítulo você irá aprender:

- Escrevendo mensagens para o usuário
- Tipos de dados
- Declaração de variáveis
- Regras de nomenclatura de um identificador
- Lendo dados do usuário
- Operações matemáticas
- Comparando valores
- Operações lógicas

Anotações	

Escrevendo mensagens para o usuário

Para escrever mensagens para usuário podemos utilizar o comando document.write().

Este comando recebe como argumento um texto delimitado entre aspas simples (‘texto aqui’) ou aspas duplas (“texto aqui”) e em seguida escreve no documento o texto informado pelo programador.

O exemplo abaixo cria um script que apresenta duas mensagens ao documento:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <body>
4     <script>
5        document.write('<p>escrevendo o texto com Aspas
   Simples</p>');
6        document.write("<p>escrevendo o texto com Aspas
   Duplas</p>");
7     </script>
8  </body>
9  </html>
```

Hora de Praticar

1) Crie um novo script que escreva as seguintes mensagens:
   a) Seu nome completo.
   b) O nome de seu time de futebol preferido.
   c) O que você espera atingir nesse treinamento.
   d)
2) Crie um novo script que apresente a seguinte mensagem:
   “Seu trabalho vai preencher uma parte grande da sua vida, e a única maneira de ficar realmente satisfeito é fazer o que você acredita ser um ótimo trabalho. E a única maneira de fazer um excelente trabalho é amar o que você faz”. Steve Jobs
3) Crie um novo script que apresente uma listagem com o nome de 5 cidades que você gostaria de visitar ou morar.

Anotações

Tipos de Dados

Os tipos de dados em javascript são divididos basicamente em 2 grupos denominados Tipos Primitivos e Tipos Objeto.

Tipos primitivos

São tipos imutáveis, ou seja, que não mudam. São considerados tipos primitivos as estruturas de dados que não se enquadram como Objetos ou Função. Em Javascript temos 6 tipos de dados primitivos:

boolean

Representa um valor lógico verdadeiro(true) ou falso(false).

Exemplo:

var continua = true;

null

Representa a atribuição de um valor nulo.

Exemplo:

var valorInicial= null;

undefined

Representa um valor não definido.

Exemplo:

var totalDePontos= undefined;

number

Representa um valor numérico inteiro (int) ou real(float). podem ser atribuídos valores reais entre -(25^3 -1) and 25^3 -1) .

Anotações	

Exemplo:

var pi= 3.1416;

var idade = 18;

string

Representa um valor do tipo texto.

Exemplo:

var nomeCurso = ‘Academia Frontend’;

Symbol

Tipo de dado utilizado para representar propriedades de um objeto. Este tipo de dado foi incluído na especificação ES6 (EcmaScript 2015) .

Exemplo:

var Obj= {};

Obj[Symbol(‘nomePropriedade’)] = valor;

Tipos de dados Objeto

Objetos são estruturas de dados que possuem propriedades e comportamentos. Essas estruturas são utilizadas para realizarmos o mapeamento de objetos da vida real para o mundo computacional.

Em um capítulo posterior entraremos em mais detalhes quanto a criação e utilização de objetos.

Operador typeof

Para identificarmos o tipo de dado de um determinado valor podemos utilizar o operador typeof.

Exemplo:

typeof ‘teste’ // ‘string’

	Anotações

typeof 25 // ‘number’

typeof true // ‘boolean’

Declarações de Variáveis

Variáveis são identificadores utilizados para manipulação de endereços de memória utilizados para o armazenamento de dados de um sistema computacional.

Javascript realiza uma inferência dinâmica em tempo de execução para verificar o tipo de dado de uma variável. Isto significa que no momento da atribuição ele infere o tipo conforme o valor atribuído para a variável.

Em Javascript podemos declarar uma variável utilizando os símbolos **var**, **let** e **const** mais o nome do identificador.

Símbolo var

O símbolo **var** foi utilizado desde o início da linguagem para a definição de variáveis. Variáveis declaradas com o símbolo **var** possuem seu escopo definido dentro do contexto de execução e sua sintaxe é definida conforme exemplo abaixo:

Exemplo:

var nomeCurso=”Academia Frontend”

Símbolo let

O símbolo **let** foi introduzido na linguagem na especificação ES6 (EcmaScript 2015). Variáveis declaradas com o símbolo **let** possuem um escopo de bloco e devem ser preferencialmente utilizadas durante a definição de variáveis.

Exemplo:

let nomeCurso=”Academia Frontend”

Símbolo const

Anotações	

O símbolo **const** foi introduzido na linguagem na especificação ES6 (EcmaScript 2015). Variáveis declaradas com o símbolo **const** não podem ser alteradas e devem declarar o seu valor no momento da definição da variável.

Exemplo:

cons PI=3.1416;

Regras de nomenclatura de um identificador

Uma variável deve começar com uma letra, sublinhado(_) ou cifrão($), caracteres subsequentes podem ser letras de “a” até “z” ,”A” até “Z” ou dígitos(0-9). Javascript é Case Sensitive , ou seja diferencia entre letras minúsculas e maiúsculas.

O exemplo abaixo apresenta a declaração e inicialização de variáveis:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <body>
4     <script>
5          let nome = 'João';
6          let sobrenome = ' da Silva Brasil';
7          let idade = 25;
8          document.write('<h3> Dados do Usuário</h3>');
9          document.write('<p><strong>Nome completo:</strong>' +
   nome + sobrenome +'</p>')
10         document.write('<p><strong>idade:</strong>' + idade +
   '</p>');
11    </script>
12 </body>
13 </html>
```

Lendo dados do usuário

Para lermos informações do usuário podemos utilizar o comando window.prompt(textoDemensagem).

Este comando apresenta uma caixa de mensagem com o título informado no parâmetro textoDeMensagem ao usuário e retorna o valor digitado pelo usuário após a confirmação, caso o usuário cancele a caixa de mensagem o resultado será null.

O exemplo abaixo apresenta um script com a utilização do comando window.prompt():

Anotações

```
<!DOCTYPE html>
<html lang="pt">
<body>
   <script>
        let nome = window.prompt('Informe seu nome');
        let sobrenome = window.prompt('Informe seu sobrenome');
        let idade = window.prompt('Informe sua idade');
        document.write('<h3> Dados do Usuário</h3>');
        document.write('<p><strong>Nome completo:</strong>' +
nome +' '+ sobrenome +'</p>')
        document.write('<p><strong>idade:</strong>' + idade +
'</p>');
   </script>
</body>
</html>
```

Observação:

O comando window.prompt() sempre terá como resultado um valor do tipo string ou um valor null. Caso desejarmos realizar a leitura de algum valor numérico devemos realizar a conversão para o tipo apropriado. Para os valores inteiros e reais podemos utilizar os comandos parseInt() ou parseFloat() conforme exemplo abaixo:

```
<!DOCTYPE html>
<html lang="pt">
<body>
   <script>
        let numero = window.prompt('digite sua idade');
        numero = parseInt(numero);
        let idadeMais20 = numero + 20;
        document.write('<p> sua idade daqui a 20 anos será
'+idadeMais20+'</p>');

        let preco = window.prompt('Informe o valor do
produto');
        preco = parseFloat(preco);
        let desconto = window.prompt('Informe o percentual do
desconto (0-100)');
        desconto = parseFloat(desconto);
        let valorTotal = preco -(preco * desconto/100);
        document.write('<p>valor do produto
<strong>R$'+preco.toFixed(2)+ '</strong></p>');
        document.write('<p>Percentual de desconto
<strong>'+desconto.toFixed(2)+ '%</strong></p>');
        document.write('<p>valor total da compra
```

Anotações

```
<strong>R$'+valorTotal.toFixed(2)+ '</strong></p>');
   </script>
</body>
</html
```

Operações Matemáticas

Os operadores matemáticos em javascript são:

+ Adição

Exemplo:

```
<!DOCTYPE html>
<html lang="pt">
<body>
   <script>
        let numero1 = 20;
        let numero2 = 30;
        let soma = numero1 + numero2;
        document.write('O resultado da soma é ' + soma);
   </script>
</body>
</html>
```

- Subtração

Anotações

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <body>
4     <script>
5      let numero1 = 30;
6      let numero2 = 10;
7      let resultado = numero1 - numero2;
8      document.write('O resultado da subtração é ' + resultado);
9     </script>
10 </body>
11 </html>
```

*** Multiplicação**

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <body>
4     <script>
5      let numero1 = 30;
6      let numero2 = 10;
7      let resultado = numero1 * numero2;
8      document.write('O resultado da multiplicação é ' +
resultado);
9     </script>
</body>
```

Anotações

```
10 </html>
11
```

/ Divisão

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <body>
4     <script>
5      let numero1 = 10;
6      let numero2 = 3;
7      let resultado = numero1 / numero2;
8      document.write('O resultado da divisão é ' + resultado);
       </script>
9  </body>
10 </html>
```

% Resto da Divisão

O operador resto de divisão retorna o valor restante de uma divisão entre dois números inteiros.

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <body>
4     <script>
5      let numero1 = 10;
```

Anotações

```
    let numero2 = 3;
    let resultado = numero1 % numero2;
    document.write('O resto da divisão é  ' + resultado);
   </script>
</body>
</html>
```

++ Incremento

O operador de incremento unário incrementa o valor de uma variável em uma unidade.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt">
<body>
   <script>
    let numero = 10;
    numero++;
    document.write('O valor atual de número é ' + numero);
    ++numero;
    document.write('O valor atual de número é ' + numero);
   </script>
</body>
</html>
```

-- Decremento

O operador de decremento unário decrementa o valor de uma variável em uma unidade.

Exemplo:

Anotações

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <body>
4     <script>
5      let numero = 10;
6      numero--;
7      document.write('O valor atual de número é ' + numero);
8      --numero;
9      document.write('O valor atual de número é ' + numero);
10    </script>
11 </body>
12 </html>
```

Precedência entre Operadores

A precedência de operadores determina a ordem em que os operadores são processados. Operadores com maior precedência são processados primeiro.

Em Javascript os operadores matemáticos seguem a seguinte precedência, da esquerda para a direita:

*** / % + -**

Caso seja necessário alterarmos a precedência dos operadores em um determinado cálculo, podemos utilizar o parênteses para indicar quais operações devem ser executadas primeiro.

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="pt">
```

Anotações

```
3  <body>
4     <script>
5      document.write('<p> 2 + 2 * 3 =  '+(2 + 2 * 3)+'</p>');
6      document.write('<p>(2 + 2) * 3 =  '+((2 + 2) * 3)+'</p>');
7     </script>
8  </body>
9  </html>
```

Comparando Valores

Os operadores de comparação utilizados em javascript são:

< Menor que

<= Menor ou igual que

> Maior

>= maior ou igual a

!= diferente de

== igual a

Toda a comparação em javascript retorna um valor boolean true (verdadeiro) ou false (false).

Exemplo:

2 < 10; // resulta em um valor true

219 < 10; // resulta em um valor false

10 > 2; // resulta em um valor true

5 != 10; // resulta em um valor true

5 == 5; // resulta em um valor true

Anotações

Devemos ter um cuidado especial quando fazer a comparação entre valores do tipo String e Number, pois este tipo de comparação também pode ser realizado.

2 == “2”; //resulta em valor true

10 > “100”; resulta em um valor false;

Os operadores diferente de (!=) e igual a (==) possuem uma outra versão que além de comparar os valores também fazem a comparação de seu tipos.

!== Estritamente diferente de

=== Estritamente igual ª

Exemplo:

2 === 2; //resulta um valor true

2 === “2”;// resulta em um valor false

Hora de Praticar

1. Crie um script que solicite um número ao usuário e em seguida escreva o resultado das comparações abaixo com o número informado.
    a. número < 10;
    b. número <= 100;
    c. número == 20;
    d. número != 50;
    e. número > 30;
    f. número >= 25;
2. Crie um script que solicite o nome do time de futebol preferido do usuário em seguida escreva o resultado da comparação do nome informado com o nome do seu time.

Operações Lógicas

Os operadores lógicos servem para realizar a comparação entre duas expressões que retornam um valor boolean true ou false. O resultado desta comparação sempre será um valor lógico true ou false.

&& Operador E

	Anotações

Realiza a avaliação de duas proposições e retorna verdadeiro apenas se as duas proposições forem verdadeiras.

Exemplo:

var x = true;

var y = true;

var z = false;

x && y; // resulta em um valor true;

x && z; // resulta em um valor false;

|| Operador OU

Realiza a avaliação de duas proposições e retorna verdadeiro se uma das proposições forem verdadeiras.

Exemplo:

var x = true;

var y = true;

var z = false;

x || y; // resulta em um valor true;

x || z; // resulta em um valor true;

z || z; // resulta em um valor false;

! Operador NAO

Inverte o valor da proposição.

Exemplo:

var x = true;

Anotações	

var z = false;

!x; // resulta em um valor false;

!z; // resulta em um valor true;

^ Operador OU Exclusivo

Realiza a avaliação de duas proposições e retorna verdadeiro se apenas uma das duas proposições forem verdadeiras.

Exemplo:

var x = true;

var y = true;

var z = false;

x ^ y; // resulta em um valor false;

x ^ z; // resulta em um valor true;

z ^ z; // resulta em um valor false;

Hora de Praticar

1. Dado o script abaixo.

```
1  <!DOCTYPE html>
```

Anotações

```
<html lang="pt">
   <head>
       <meta charset="UTF-8">
   </head>
<body>
   <script>
      let valorA = true;
      let valorB = true;
      let valorC = false;
      document.write(valorA && valorA);
      //complete o script conforme solicitado.
   </script>
</body>
</html>
```

Escreva o resultado das seguintes comparações:

a. valorA E valorB
b. valorA E valorB E valorC
c. valorA E valorC
d. valorB E valorC
e. valorA OU valorB
f. valorB OU ValorC
g. valorA OU valorB ou valorC
h. valorC OU valorC

Anotações

Capítulo 03 – Faça suas Escolhas

Objetivos:

Neste capítulo você irá aprender:

- Operador if/else
- Operador switch
- Operador ternário

	Anotações

Operador if/else

Quando queremos realizar algum desvio do fluxo conforme alguma condição podemos utilizar o comando if, este como recebe como parâmetro qualquer expressão que retorne true ou false.

Caso o valor seja true ele será executado.

var desconto = true;

if(desconto){

    //código de desconto aqui

}

Abaixo temos o exemplo de um script que utiliza o comando if.

```
<!DOCTYPE html>
<html lang="pt">
   <head>
       <meta charset="UTF-8">
   </head>
<body>
   <script>
     let numeroSecreto = Math.floor(Math.random() * (50+1) );
     let numeroDigitado = window.prompt('Digite um número
entre 0 e 50');
     numeroDigitado = parseInt(numeroDigitado);

     if(numeroDigitado && numeroSecreto === numeroDigitado){
         document.write('<h1>Parabéns você acertou! O número
secreto era '+numeroSecreto+'</h1>')
     }
   </script>
</body>
</html>
```

Anotações

Podemos adicionar a instrução else que será executada caso o resultado da comparação do if seja falso.

let desconto = false;

if(desconto){

    //código de desconto aqui

} else{

    // código sem desconto aqui

}

```
<!DOCTYPE html>
<html lang="pt">
   <head>
       <meta charset="UTF-8">
   </head>
<body>
   <script>
     let numeroSecreto = Math.floor(Math.random() * (50+1) );
     let numeroDigitado = window.prompt('Digite um número
entre 0 e 50');
     numeroDigitado = parseInt(numeroDigitado);

     if(numeroDigitado && numeroSecreto === numeroDigitado){
         document.write('<h1>Parabéns você acertou! O número
secreto era '+numeroSecreto+'</h1>')
     } else{
         document.write('Que pena você não acertou. |:(|');
     }

   </script>
</body>
</html>
```

Caso o if receba uma expressão que retorne um valor numérico, null ou undefined ele irá inferir os seguintes resultados;

0 →false

qualquer valor numérico != 0 → true

null → false

Anotações

undefined → false;

Hora de Praticar

1. Escreva um script que solicite um número para o usuário e em seguida escreva uma mensagem informando se o número informado é múltiplo de 5. Para que um número seja múltiplo de 5, ele deve ser maior ou igual a 5 e o resto da divisão por 5 deve ser igual a zero.
2. Escreva um script que leia um número informado pelo usuário e em seguida escreva uma mensagem indicando se o número é par ou impar. (Considere número par o número cujo resto da divisão por 2 seja igual a Zero)
3. Escreva um script que solicite a idade do usuário e em seguida escreva a mensagem 'Você é adulto!' caso a idade informada seja maior ou igual a 18.
4. Escreva um programa que solicite a nota de um aluno e em seguida escreva a mensagem "Aprovado" caso a nota seja igual ou maior que 6, senão escreva a mensagem "Reprovado".
5. Escreva um script que solicite as 3 notas de um aluno e em seguida calcule o valor da média das notas. Caso a média seja maior ou igual a 9 escreva a mensagem "Conceito A", caso a nota seja menor que 9 e maior ou igual a 7 escreva a mensagem "Conceito B", caso a nota seja menor que 7 e maior ou igual a 5 escreva a mensagem "Conceito C", se a nota for menor que 5 escreva a mensagem "Conceito Insuficiente";

Operador Switch

O operador **switch** avalia uma expressão combinando o valor da expressão para uma cláusula **case**, executando as instruções associadas ao **case**.

Sintaxe:

```
switch(expressão){
   case valor1:
   //instruções
   break;
   case valor2:
   //instruções
```

Anotações	

```
    break;
    default:
    //instruções;
}
```

Exemplo:

```
<!DOCTYPE html>
<html lang="pt">
<head>
    <meta charset="UTF-8">
</head>
<body>
    <script>
        let dia = window.prompt('Digite um número entre 1 e 7');
        dia = parseInt(dia);
        switch (dia) {
            case 1:
                document.write('<h3>Domingo</h3>')
                break;
            case 2:
                document.write('<h3>Domingo</h3>')
                break;
            case 3:
                document.write('<h3>Domingo</h3>')
                break;
            case 4:
                document.write('<h3>Domingo</h3>')
                break;
            case 5:
                document.write('<h3>Domingo</h3>')
                break;
            case 6:
                document.write('<h3>Domingo</h3>')
                break;
            case 7:
                document.write('<h3>Domingo</h3>')
                break;
            default:
                document.write('<h3>Número inválido</h3>')
                break;
        }
```

Anotações

```
    </script>
</body>
</html>
```

Hora de Praticar

1. Escreva um script que apresente uma lista com as opções ‘1 - Sacar’, ‘2 - Depositar’, ‘3 - verificar saldo’. Em seguida solicite que o usuário informe um número entre 1 e 3. Caso o usuário tenha informado o número 1 apresente a mensagem “você escolheu a opção sacar”, caso o usuário tenha informado o número 2 apresente a mensagem “você escolheu a opção depositar”, caso o usuário tenha informado o número 3, apresente a mensagem ‘você escolheu a opção verificar saldo’. Caso o usuário tenha informado um número fora do range(1,2,3) apresente a mensagem ‘Opção inválida’.
2. Escreva um script que solicite o nome de um dos três países com mais títulos da copa do mundo, em seguida escreva uma mensagem ao usuário informando se a resposta está correta ou errada. Considere os três países com mais títulos como sendo BRASIL, ITÁLIA E ALEMANHA. (Dica: tente utilizar um switch sem breaks).

Operador Ternário

O operador ternário é muito utilizado quando queremos atribuir um determinado valor a uma variável caso uma condição seja verdadeira ou um valor diferente caso a condição seja falsa.

let variavel = condição ? valor_caso_positivo : valor_caso_falso;

Exemplo:

```
<!DOCTYPE html>
<html lang="pt">
<head>
   <meta charset="UTF-8">
</head>
<body>
   <script>
      let idade = window.prompt('Informe sua idade:');
      idade = parseInt(idade);
      let mensagem = idade >=18 ? 'Você é Maior' : 'Voce é
Menor';
```

Anotações

```
11          document.write('<h3>'+mensagem+'</h3>');
12      </script>
13  </body>
14  </html>
```

Anotações

Anotações	

Capítulo 04 – Repetindo Passos

Objetivos:

Neste capítulo você irá aprender:

- Operador while
- Operador for

	Anotações

Operador While

O operador while é utilizado quando queremos repetir algum trecho de código enquanto uma determinada condição seja verdadeira. Ou seja, este operador recebe um valor lógico, avalia esse valor e executa um bloco de código caso o valor lógico seja verdadeiro.

Sintaxe:

while(expressão){

//bloco de código

}

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <head>
4     <meta charset="UTF-8">
5  </head>
6  <body>
7     <script>
8       let contador = 1;
9       while(contador <= 10){
10           document.write('<p>'+contador+'</p>');
11           contador++;
12      }
13    </script>
14 </body>
15 </html>
```

Anotações

Hora de Praticar

1. Escreva um script que escreva os números entre 1 e 1000.
2. Escreva um script que solicite dois números distintos para o usuário, e em seguida, escreva todos os número entre o intervalo dos números informados.
3. Escreva todos os números pares entre o número 101 e o número 200.
4. Escreva um script que solicite um número ao usuário e em seguida escreva o valor do número informado multiplicado por 5. O script deve continuar solicitando novos números até que o usuário digite um número negativo.

Operador For

A estrutura de controle for é utilizada quando queremos repetir algum trecho de código um número determinado de vezes.

Sua sintaxe é :

for(bloco de inicialização; bloco de validação; bloco de finalização){ //corpo }

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <head>
4     <meta charset="UTF-8">
5  </head>
6  <body>
7     <script>
8       for(let index = 0; index< 10; index++){
9            document.write('<p>'+index+'</p>')
10      }
```

Anotações

```
11      </script>
12  </body>
13  </html>
```

Hora de Praticar

1. Escreva um script que mostre todos os números entre 1 e 100;
2. Escreva um script que apresente todos os números ímpares entre 1 e 100.
3. Escreva um script que mostre na tela o desenho abaixo

   X

   XX

   XXX

   XXXX

   XXXXX

4. Escreva um script que mostre na tela o desenho abaixo:

   $$$$$

   $$$$

   $$$

   $$

   $

5. Escreva um script que calcule e escreva o resultado da soma dos números entre 1 e 50.

Anotações

	Anotações

Capítulo 05 – Evite Códigos Duplicados

Objetivos:

Neste capítulo você irá aprender:

- Criando uma função
- Utilizando uma função

Anotações	

Criando uma Função

Em javascript funções são objetos de primeira classe, ou seja, eles são objetos e podem ser manipulados e repassados como qualquer outro objeto. Especificamente, eles são objetos de função.

Toda a função retorna algum valor. Caso a função possua a instrução return a função retornará o valor informado por esta instrução, caso a função seja uma função construtora ela retornará uma nova instância do objeto, caso seja uma função simples retornará o valor undefined;

Para declararmos uma função devemos utilizar a seguinte sintaxe:

```
function nome_da_funcao(lista_de_parametros){
        //corpo da função
}
```

onde:

function → palavra reservada que indica que queremos criar uma função.

nome_da_funcao → identificador da função

lista_de_parametros → lista de parâmetros que a função espera receber

Anotações

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <head>
4     <meta charset="UTF-8">
5  </head>
6  <body>
7     <script>
8       function soma(a,b){
9           return a + b;
10      }
11    </script>
12 </body>
13 </html>
```

Utilizando uma Função

Para utilizarmos uma função basta chamar a função desejada passando para a sua lista de argumentos os valores necessários. Caso a função a ser chamada não possua argumentos de entrada, a lista de argumentos deve ficar vazia.

Anotações

O exemplo abaixo cria uma função e em seguida a executa repetidas vezes.

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <head>
4     <meta charset="UTF-8">
5  </head>
6  <body>
7     <script>
8         function soma(a,b){
9             return a + b;
10        }
11      function escreverSoma(a,b){
12          document.write('<p>'+a +' + '+b + ' = '+ soma(a,b));
13      }
14
15      escreverSoma(2,2);
16      escreverSoma(3,2);
17      escreverSoma(50,50);
18
19    </script>
20 </body>
21 </html>
```

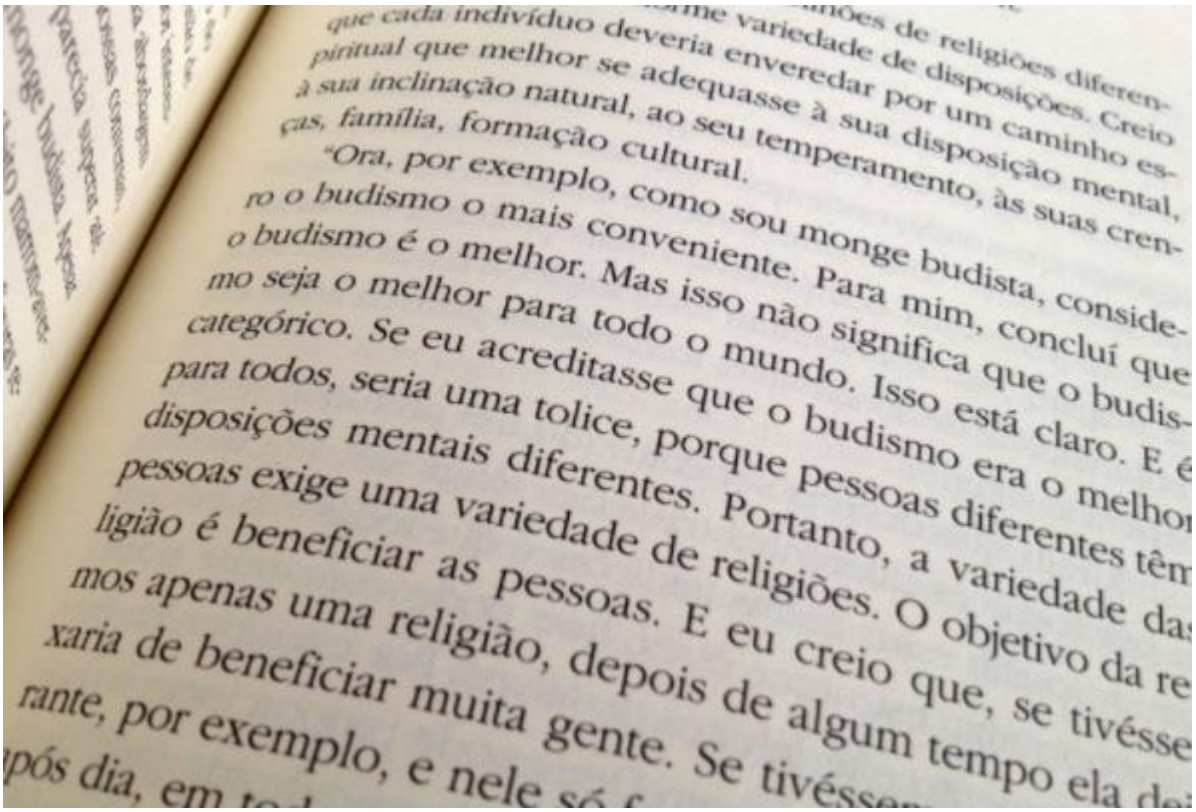
Anotações

Hora de Praticar

1. Escreva uma função que calcule o valor da média de 3 números.
2. Escreva uma função que calcule a área de um retângulo (base X altura);
3. Escreva uma função que calcule a área de um triângulo (base x altura /2) ;
4. Escreva uma função que escreva uma mensagem na tela 100 vezes;
5. Escreva uma função para a impressão de mensagens, esta função deve receber como parâmetros uma mensagem e o número de vezes que a mensagem deve aparecer.

Anotações	

Capítulo 06 – Manipulação de Texto



Objetivos:

Neste capítulo você irá aprender:

- Forma literal
- Caracteres Especiais
- Template String
- Contando letras
- Trocando letras
- Acrescentando letras
- Procurando letras em uma frase
- Dividindo frases
- Como separar os caracteres de um texto
- Mudando a caixa de Texto

	Anotações

Forma Literal

Para declarar um texto em javascript de forma literal podemos utilizar aspas simples ou aspas duplas. No exemplo abaixo as tuas declarações estão corretas.

Exemplo:

let nome = 'João'

let email = “joao@teste.com”

Caracteres Especiais

Caracteres especiais são os caracteres que não podemos escrever diretamente em um texto mas ao invés disso devemos utilizar a notação de escape:

\o	caractere NULL
\'	aspas simples
\"	aspas duplas
\\	barra invertida
\n	nova linha
\r	carriage return
\v	tab vertical
\t	tab

Anotações	

\b backspace

\f form feed

Template String

Template string é um recurso que serve para criarmos templates através de textos que podem ser completados com valores de variáveis.

Para criarmos um template precisamos colocar o texto desejado entre dois sinais crase ``. Para incluirmos os valores de uma variável no template devemos utilizar o sinal ${}.

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <head>
4     <meta charset="UTF-8">
5  </head>
6  <body>
7     <script>
8     function escreverDadosContato(nome,email,telefone){
9         let dados =  `<p>Nome:${nome}</p>
10                    <p>Email:${email}</p>
11                    <p>telefone:${telefone}</p>`;
12        document.write(dados);
13    }
14    escreverDadosContato('João','joao@teste.com','33221100');
15   escreverDadosContato('Maria','maria@teste.com','33221111');
16    escreverDadosContato('Adão','adão@teste.com','33221122');
17    escreverDadosContato('Eva','eva@teste.com','33221133');
```

Anotações

```
18      </script>
19  </body>
20  </html>
```

Contando Letras

Todo o objeto de texto (string) tem uma propriedade chamada length. Esta propriedade retorna a quantidade de caracteres que a string possui.

Exemplo:

```
1   <!DOCTYPE html>
2   <html lang="pt">
3   <head>
4       <meta charset="UTF-8">
5   </head>
6   <body>
7       <script>
8           let  titulo = 'Lógica de programação';
9           document.write(`<h3> A frase '${titulo}' possui
    ${titulo.length} caracteres</h3>`);
        </script>
10  </body>
11  </html>
12
```

Anotações

Trocando Letras

Strings em javascript são objetos imutáveis (que não podem ser modificados), entretanto os objetos strings possuem uma função chamada replace(old,new) que é utilizada para criar uma nova string trocando alguns caracteres conforme necessário.

A função replace possui 2 argumentos. O primeiro argumento pode ser uma String ou uma expressão regular e serve para indicar quais caracteres queremos alterar. Já o segundo argumento serve para indicar quais serão os novos caracteres substitutos.

Para passar uma expressão regular utilizamos o sinal de barras antes e após os caracteres que queremos utilizar em nossa expressão. As expressões regulares também podem conter um modificador ao final para indicar como devem operar.

Os modificadores que usaremos neste treinamento são:

g

corresponder globalmente

i

ignorar maiúsc./minúsc.

O exemplo abaixo apresenta a utilização da função replace juntamente com uma expressão regular.

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <head>
4      <meta charset="UTF-8">
5  </head>
6  <body>
7      <script>
8          let palavra = 'Felipe';
9          document.write(`<h3> ${palavra.replace(/e/g,'i')} -
   ${palavra} </h3>`);
10     </script>
11 </body>
12 </html>
```

Anotações

Acrescentando Letras

Para criarmos novas palavras resultantes da adição de novos caracteres podemos utilizar o operador + ou a função concat(novosCaracteres). A função concat recebe como parâmetro o texto que deve ser adicionado e retorna um novo texto contendo a concatenação do texto original com o texto passado no parâmetro da função.

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <head>
4     <meta charset="UTF-8">
5  </head>
6  <body>
7     <script>
8        let curso = 'Curso '+ 'de ';
9        curso = curso.concat('Lógica de
   ').concat('Programação');
10       document.write(`<h3> ${curso} </h3>`);
11    </script>
12 </body>
13 </html>
```

Procurando Letras em uma Frase

Para procurarmos por um caracter ou por uma combinação de caracteres em um texto, podemos utilizar a função indexOf(valor,apartirDoIndice).

Essa função recebe dois parâmetros em sua lista de argumentos. O primeiro parâmetro a ser informado é o valor a ser pesquisado no texto. Já o segundo valor é opcional, e serve para indicar a partir de qual posição do texto se deseja iniciar a busca.

O resultado da função será o índice da primeira ocorrência do valor pesquisado. Caso o valor pesquisado não seja encontrado, a função irá retornar o valor -1.

Anotações

O exemplo abaixo apresenta o uso da função indexOf.

```
<!DOCTYPE html>
<html lang="pt">
<head>
   <meta charset="UTF-8">
</head>
<body>
   <script>
       function verificarPalavra(frase,palavra){
           let index = frase.indexOf(palavra);
           if(index > -1){
               document.write(`<p>A frase '${frase}' contém a palavra '${palavra}'</p>`)
           }else{
               document.write(`<p>A frase '${frase}' não contém a palavra '${palavra}'</p>`)
           }
       }
       let frase = 'Brasil acima de tudo! abaixo somente de Deus!';
       verificarPalavra(frase,'Deus');
       verificarPalavra(frase,'Brasil');
       verificarPalavra(frase,'Argentina');
   </script>
</body>
</html>
```

Dividindo Frases

Podemos dividir uma String em partes menores utilizando a função substring(inicio,fim). A função substring recebe dois parâmetros em sua lista de argumentos.

O primeiro parâmetro indica qual a posição inicial da nova string a ser criada. Já o segundo parâmetro é opcional e serve para indicar qual será a posição final. Caso o segundo parâmetro não seja informado será considerado o tamanho total da string.

Anotações

O exemplo abaixo apresenta o uso da função substring.

```
<!DOCTYPE html>
<html lang="pt">
<head>
   <meta charset="UTF-8">
</head>
<body>
   <script>
       let frase = 'Brasil acima de tudo! Abaixo somente de
Deus!';
       document.write(`<p>${frase.substring(0,22)}</p>`)
       document.write(`<p>${frase.substring(22)}</p>`)
   </script>
</body>
</html>
```

Como separar os caracteres de um Texto

Para apresentarmos apenas um caracter de um texto podemos utilizar a função charAt(index). Esta função recebe como parâmetro o índice de um dos caracteres da string e retorna o caracter correspondente a posição informada.

O exemplo abaixo demonstra o uso da função charAt.

```
<!DOCTYPE html>
<html lang="pt">
<head>
   <meta charset="UTF-8">
</head>
<body>
   <script>
       let texto = 'Brasil';
       for(let index = 0; index < texto.length; index++){
           document.write(`<p>${texto.charAt(index)}</p>`)
       }
   </script>
</body>
</html>
```

Anotações

Mudando a caixa de Texto

Algumas vezes é necessário apresentar todo um texto em caixa baixa (letras minúscula) ou caixa alta (letra maiúscula). Para apresentar o texto em caixa baixa podemos utilizar a função toLowerCase(), esta função retorna o texto com todas as letras minúsculas, já se desejarmos o efeito contrário. Podemos utilizar a função toUpperCase(), esta função retorna a string com todos os caracteres maiúsculos.

O exemplo abaixo apresenta o uso das funções toLowerCase e toUpperCase()

```
<!DOCTYPE html>
<html lang="pt">
<head>
    <meta charset="UTF-8">
</head>
<body>
    <script>
        let pais = 'Brasil';
        let nomeCompleto = 'JOÃO DA SILVA';
        document.write(`<p>${pais.toUpperCase()}</p>`);
        document.write(`<p>${nomeCompleto.toLowerCase()}</p>`);
    </script>
</body>
</html>
```

Hora de Praticar

1. Escreva um script que solicite uma palavra ao usuário e em seguida apresente a quantidade de caracteres da palavra digitada.
2. Escreva um script que solicite uma palavra ao usuário e em seguida escreva a palavra digitada conforme a quantidade de caracteres da palavra.
3. Escreva um script que solicite uma frase ao usuário e em seguida apresente a frase digitada separada em duas partes.
4. Escreva um script que solicite uma frase para o usuário e em seguida escreva a frase invertida.
5. Escreva um script que solicite quatro palavras ao usuário e em seguida apresente todas as palavras separadas por um traço(‘ - ’).

Anotações

Anotações	

Capítulo 07 – Crie Objetos

Objetivos:

Neste capítulo você irá aprender:

- Objetos literais
- Funções construtoras

	Anotações

Objetos

Em javascript podemos declarar um objeto de várias formas aqui será apresentado a definição de um objeto de forma literal e utilizando uma função construtora.

Objetos Literais

Para declararmos um objeto de forma literal basta utilizamos o operador Chave {}, sempre que atribuímos este operador a uma variável estamos criando um novo objeto.

Exemplo:

```
var Pessoa = {};

Pessoa.nome = 'João'; // adiciona a propriedade nome ao Objeto criado
```

Também podemos definir propriedades para um objeto no momento de sua criação;

Exemplo:

```
var Pessoa = {nome: 'Maria'};

console.log(Pessoa.nome);
```

Funções Construtoras

Para criarmos um objeto através de uma função construtora basta criarmos uma função e em seguida chamá-la utilizando o operador new.

Exemplo:

```
function Contato(nome){
        this.nome = nome;
}

var contato1 = new Contato('Manoel');

console.log(contato1);
```

Anotações	

Observe que quando criamos uma função construtora, devemos utilizar a referência ‘this’ como mantenedora das propriedades e funções do objeto. A referência ‘this’ é uma referência para a própria instância do objeto criado.

Anotações

Capítulo 08 – Armazenando Dados em Listas

Objetivos:

Neste capítulo você irá aprender:

- O que é um array
- Criando a primeira lista
- Adicionando itens em uma lista
- Mostrando todos os itens da lista
- Trocando itens em uma lista
- Removendo itens da lista

Anotações	

O que é um Array

Arrays podem ser descritos como containers que armazenam outros containers. Em outras palavras, em javascript são objetos que armazenam uma lista de outros objetos.

Para acessarmos os elementos que estão contidos em um array devemos utilizar o índice da posição em que o elemento está armazenado.

A indexação dos arrays inicia-se na posição o e vai até o tamanho do array -1.

Para sabermos a quantidade de elementos que um array possui podemos utilizar a propridade length.

Criando a primeira Lista

Em javascript podemos criar arrays de forma literal utilizando o operador colchetes, ou criando uma nova instância do Objeto Array:

```
var nomes = []; // os colhcetes indicam que a variável deve apontar para um array
```

ou

```
var contatos = new Array();
```

Adicionando itens em uma Lista

Podemos utilizar a função push para incluir um elemento no final do array

```
var nomes = ['Eva','Maria'];

nomes.push('Adão'); //['Eva','Maria','Adão'];
```

Podemos utilizar a função unshift para incluir um elemento no inicio do array

```
var nomes = ['Eva','Maria'];

nomes.unshift('Adão'); // ['Adao','Eva','Maria'];
```

Podemos ainda incluir um elemento no array utilizando um índice.

```
var nomes = [];

nomes[0] = 'Adão';
```

	Anotações

Mostrando todos os itens da Lista

Para percorrermos todos os itens de uma lista, podemos utilizar o comando for para indexar as posições da lista e assim percorrermos todos os seus elementos.

Para identificarmos a quantidade de elementos da lista devemos utilizar a propriedade length. Esta propriedade está presente em todas as listas e indica a quantidade de elementos que a lista possui.

O exemplo abaixo demonstra como realizar essa tarefa.

```
1  <!DOCTYPE html>
2  <html lang="pt">
3  <body>
4      <script>
5          let lista = ['João','Maria','Adão','Eva'];
6          for(let indice = 0; indice < lista.length; indice++){
7              document.write(`<p>O item atual é ${lista[indice]}
   e está na posição ${indice}</p>`);
8          }
9      </script>
10 </body>
11 </html>
```

Trocando itens em uma Lista

Para trocar um determinado item em uma lista basta acessarmos o índice onde queremos alterar o valor e em seguida atribuirmos um novo valor.

Caso seja preciso primeiramente verificar se um elemento está presente na lista podemos utilizar a indexOf para descobrir em qual posição o elemento procurado se encontra.

A função indexOf retorna a posição do elemento na lista, caso o elemento procurado não seja encontrado, a função retornará a posição -1.

Anotações

O exemplo abaixo demonstra o uso da função indexOf em uma lista.

```
<!DOCTYPE html>
<html lang="pt">
    <head>
        <meta charset="UTF-8">
    </head>
<body>
    <script>
        let lista = ['João','Maria','Adão','Eva'];
        let posicao = lista.indexOf('Eva');
        if(posicao > -1){
            lista[posicao] = 'Berenice';
        }
        document.write(lista.join(', '))
    </script>
</body>
</html>
```

Removendo itens da Lista

Para remover um elemento do início do array podemos utilizar a função shift().

var nomes = ['Adão','Eva'];

nomes.shift(); // remove Adão. ['Eva']

Para remover um elemento do final do array podemos utilizar a função pop();

var nomes = ['Adão','Eva'];

nomes.pop(); // remove Eva. ['Adão']

Para remover um elemento do meio do array podemos utilizar a função splice(index,qt)

var nomes = ['Adão','Eva','José','Maria'];

nomes.splice(2,1); // remove José. ['Adão','Eva','Maria']

Anotações

Hora de Praticar

1. Escreva um script que solicite o nome e a nota de 20 alunos e os armazene em uma lista. Em seguida escreva a nota e o nome de todos os alunos.
2. Escreva um programa que solicite 10 notas e em seguida apresente a menor nota digitada e a maior nota digitada.
3. Escreva um programa que solicite o nome e o valor de 10 produtos. Em seguida calcule a média de preço de todos os produtos e apresente duas listas. A primeira lista deve conter todos os produtos com o valor abaixo da média, já a segunda lista deve conter todos os produtos com valor igual ou maior que a média.

Anotações	

Capítulo 09 – Tópicos Avançados

Objetivos:

Neste capítulo você irá aprender:

- Conheça as Funções anônimas
- Conheça as Arrow Functions
- Filtrando listas com o método filter
- Mapeando novas listas com o método map

Anotações

Conheça as Funções Anônimas

Funções anônimas são funções criadas sem a definição de nome, normalmente utilizada como callback de outras funções.

Exemplo:

```
var soma = function(a,b){
        return a+b;
}
console.log(soma(2+3))
```

Conheça as Arrow Functions

Arrow function foi introduzida na versão ES6, e possui uma sintaxe mais curta que a declaração de uma função, outra diferença é que ela vincula o valor de this de maneira léxica.

Sintaxe

```
([param],[param])=>{
//bloco de execução
}
```

Ou

```
param => comando_de_execução
```

Onde:

param: são os parâmetros necessários para a execução

Exemplos:

```
function callbackExec(callback){
callback('Frontend nota 10');
}
```

Anotações	

callbackExec(mensagem =>console.log(mensagem));

Filtrando Listas com o Método Filter

O objeto array possui um método chamado filter, esse método recebe como parâmetro uma função que será executa para cada item do array. A função passada no parâmetro do método filter recebe em seu parâmetro o item atual do array e deve retornar um valor true, caso o item corresponda ao filtro aplicado, ou false, caso o item não corresponda ao filtro.

O resultado do método filter será um novo array contendo apenas os itens que foram avaliados como verdadeiros no filtro.

O exemplo abaixo demonstra a utilização do método filter.

```
<!DOCTYPE html>
<html lang="pt">
   <head>
       <meta charset="UTF-8">
   </head>
<body>
   <script>

       function imprimirListaProdutos(titulo,produtos){
           document.write(`<h3>${titulo}</h3>`);
           document.write('<ul>');
           produtos.forEach(produto=>{
               document.write(`<li>${produto.nome} - R$
${produto.valor.toFixed(2)}</li>`)
           })
           document.write('</ul>')
       }

       let produtos = [
           {nome: 'produto 01', valor:999.90},
           {nome: 'produto 02', valor:430.00},
           {nome: 'produto 03', valor:1999.90},
           {nome: 'produto 04', valor:2799.70},
           {nome: 'produto 05', valor:99.90}
       ];

       let filtroProdutosValorMenorQueMil = (produto)=>{
           return produto.valor < 1000;
       };

       let filtroProdutosValorMaiorIgualMil = (produto)=>{
```

Anotações

```
31              return produto.valor >= 1000;
32          };
33
34          let produtosMenorQueMil =
    produtos.filter(filtroProdutosValorMenorQueMil);
35          let produtosMaiorIgualMil =
    produtos.filter(filtroProdutosValorMaiorIgualMil);
36          imprimirListaProdutos('Produtos Valor abaixo de
    R$1000,00',produtosMenorQueMil);
37          imprimirListaProdutos('Produtos Valor a partir de
    R$1000,00',produtosMaiorIgualMil);
38
39      </script>
40  </body>
41  </html>
```

Mapeando novas Listas com o Método MAP

O Método MAP pode ser utilizado quando queremos gerar uma nova lista a partir dos valores iniciais de uma lista já existente. Esse método recebe como parâmetro uma função que é responsável por retornar o novo item para a lista que será criada. A função que é passada como parâmetro recebe em seu argumento o item atual da lista original e a partir deste item podemos gerar o novo valor a ser retornado.

O exemplo abaixo demonstra a utilização do Método MAP.

```
1   <!DOCTYPE html>
2   <html lang="pt">
3       <head>
4           <meta charset="UTF-8">
5       </head>
6   <body>
7       <script>
8           function
    gerarProdutoComPrecoDeVenda(produto,indiceMultiplicador){
9               return  {
10                  nome:produto.nome,
11                  precoCusto:produto.precoCusto,
12                  precoVenda: produto.precoCusto *
    indiceMultiplicador
13              }
14          }
15
```

Anotações

```
16          function
    gerarListaProdutosComPrecoDeVenda(produtos,indiceMultiplicador
    ){
17              return produtos.map(produto=>{
18                  return
    gerarProdutoComPrecoDeVenda(produto,indiceMultiplicador);
19              })
20          }
21
22          function imprimirListaComPrecoVenda(produtos,
    indiceMultiplicador){
23              let produtosComPrecoDeVenda =
    gerarListaProdutosComPrecoDeVenda(produtos,indiceMultiplicador
    );
24           document.write('<ul>');
25           produtosComPrecoDeVenda.forEach(produto => {
26           document.write(`<li>
27               <strong>nome:</strong>${produto.nome}
28               <strong>custo:</strong>R$
    ${produto.precoCusto.toFixed(2)}
29               <strong>valorVenda:</strong>R$
    ${produto.precoVenda.toFixed(2)}</li>`)
30              })
31          document.write('</ul>')
32          }
33          let produtos = [
34              {nome: 'produto 01', precoCusto:25.90},
35              {nome: 'produto 02', precoCusto:215.00},
36              {nome: 'produto 03', precoCusto:999.90},
37              {nome: 'produto 04', precoCusto:799.70},
38              {nome: 'produto 05', precoCusto:35.90}
39          ];
40
41          imprimirListaComPrecoVenda(produtos,2.5)
42      </script>
43  </body>
44  </html>
```

Anotações