

# Desenvolvimento Web Javascript e Frameworks

Professor Especialista Ronie Cesar Tokumoto



T646d Tokumoto, Ronie Cesar

Desenvolvimento web Javascript e Frameworks / Ronie

Cesar Tokumoto. Paranavaí: EduFatecie, 2022.

119 p. : il. Color.

1. JavaScript (Linguagem de programação de computador) .

2. Aplicações Web. 3. Sites da Web – Desenvolvimento.

I. centro Universitário UniFatecie. II. Núcleo de Educação a Distância. II. Título.

CDD: 23 ed. 005.133

Catalogação na publicação: Zineide Pereira dos Santos – CRB 9/1577

**Reitor**

Prof. Ms. Gilmar de Oliveira

**Diretor de Ensino**

Prof. Ms. Daniel de Lima

**Diretor Financeiro**

Prof. Eduardo Luiz

Campano Santini

**Diretor Administrativo**

Prof. Ms. Renato Valença Correia

**Secretário Acadêmico**

Tiago Pereira da Silva

**Coord. de Ensino, Pesquisa e Extensão - CONPEX**

Prof. Dr. Hudson Sérgio de Souza

**Coordenação Adjunta de Ensino**

Profª. Dra. Nelma Sgarbosa Roman de Araújo

**Coordenação Adjunta de Pesquisa**

Prof. Dr. Flávio Ricardo Guilherme

**Coordenação Adjunta de Extensão**

Prof. Esp. Heider Jeferson Gonçalves

**Coordenador NEAD - Núcleo de Educação à Distância**

Prof. Me. Jorge Luiz Garcia Van Dal

**Web Designer**

Thiago Azenha

**Revisão Textual**

Beatriz Longen Rohling

Caroline da Silva Marques

Carolayne Beatriz da Silva Cavalcante

Geovane Vinícius da Broi Maciel

Jéssica Eugênio Azevedo

Kauê Berto

**Projeto Gráfico, Design e Diagramação**

André Dudatt

Carlos Firmino de Oliveira

**UNIFATECIE Unidade 1**

Rua Getúlio Vargas, 333

Centro, Paranavaí, PR

(44) 3045-9898

**UNIFATECIE Unidade 2**

Rua Cândido Bertier

Fortes, 2178, Centro,

Paranavaí, PR

(44) 3045-9898

**UNIFATECIE Unidade 3**

Rodovia BR - 376, KM

102, nº 1000 - Chácara

Jaraguá, Paranavaí, PR

(44) 3045-9898

[www.unifatecie.edu.br/site](http://www.unifatecie.edu.br/site)

**As imagens utilizadas neste livro foram obtidas a partir do site Shutterstock.**

# AUTOR

## **Professor Especialista Ronie Cesar Tokumoto**

- Graduado em Bacharelado em Informática pela Universidade Federal do Paraná.
- Graduado em Gastronomia pela Unicesumar.
- Pós-Graduação em Docência no Ensino Superior pela Unicesumar.
- Pós-Graduação em Tutoria em Educação à Distância pela Faculdade Eficaz.
- Pós-Graduação em Gestão Escolar Integrada e Práticas Pedagógicas pela Faculdade Eficaz.
- Experiência na área de educação desde 1994 sendo está em escolas profissionalizantes e colégios estaduais.
- Experiência em Coordenação de Curso Técnico no ano de 2013 e Participação em Palestras e Eventos Organizados para o Curso durante este ano de 2013.
- Atuei no setor EAD da Unicesumar desde abril de 2014 como professor mediador até 2017, e como Tutor Pedagógico até 2021.
- Professor no curso de Engenharia de Software da Unifamma de agosto de 2019 a dezembro de 2020.
- Coordenador e Professor no curso de Tecnologia em Análise e Desenvolvimento de Sistemas na Faculdade Senac de Maringá.

**CURRÍCULO LATTES:** <http://lattes.cnpq.br/2653232632701400>

# APRESENTAÇÃO DO MATERIAL

## O que significa produzir conteúdo para a Internet?

Conteúdo pode representar uma ideia vaga e bastante ampla, incluindo deste texto simples até conteúdo multimídia ou páginas interativas dinâmicas que têm como objetivo, oferecer experiência de uso variadas a usuários na web.

Para o desenvolvimento de todos estes tipos de conteúdo, existem tecnologias e ferramentas distintas que auxiliam no processo, mas para os estudos nesta disciplina, a ideia é a de proporcionar as bases para o aprendizado do desenvolvimento de conteúdo dinâmico que complementa os demais elementos citados em páginas web.

A linguagem Javascript é capaz de agregar muito em termos de recursos a páginas web, sendo uma complementação bastante poderosa que traz aspectos ligados ao chamado *back end* que permitem o tratamento de dados e outras funcionalidades adicionais ao *front end* essencial fornecido pelo HTML e CSS.

Conhecer os fundamentos da linguagem Javascript abre as portas para infinitas possibilidades de desenvolvimento de software, e permite também o acréscimo de outras ferramentas poderosas chamadas de frameworks que trazem muito conteúdo padronizado e funcionalidades que tornam o trabalho de desenvolvimento de software mais eficiente e otimizado para determinados tipos de soluções computacionais.

Frameworks como AngularJS, Node.JS, React e JQuery oferecem grandes possibilidades de melhoria no desenvolvimento de aplicações com qualidade e atendendo às necessidades do mercado.

Siga adiante e inicie seus estudos numa das mais importantes áreas da tecnologia da informação.

# SUMÁRIO

UNIDADE I .....	3
<b>A Base do Desenvolvimento Web com Javascript</b>	
UNIDADE II .....	24
<b>Recursos da Linguagem Javascript</b>	
UNIDADE III .....	50
<b>Estruturas de Dados e Estruturas de Controle Javascript</b>	
UNIDADE IV .....	81
<b>Frameworks Javascript</b>	

# UNIDADE I

# A Base do Desenvolvimento

# Web com Javascript

Professor Especialista Ronie Cesar Tokumoto



## Plano de Estudo:

- Introdução ao Javascript (Js);
- Um Primeiro Script;
- Avançando em Javascript;
- Completando o Primeiro Ciclo de Conceitos em Javascript.

## Objetivos da Aprendizagem:

- Conhecer as bases do desenvolvimento de scripts para criação de páginas web;
- Aprender como criar scripts básicos em HTML e como adicionar conteúdo JavaScript;
- Compreender como podem ser utilizados símbolos e operadores em JavaScript;
  - Conhecer tipos de dados utilizáveis em scripts.

# INTRODUÇÃO

Ao iniciar seus estudos em programação web, é importante estruturar uma base conceitual inicial para que possa compreender a utilidade do que será aprendido ao longo dos estudos da disciplina em si.

Nesta etapa dos estudos em desenvolvimento web, é importante compreender alguns fundamentos do conteúdo que pode ser produzido através da linguagem JavaScript, mostrando como este conteúdo se adequa ao padrão já estabelecido pela linguagem HTML que representa a base estrutural do desenvolvimento para a Internet.

Conhecer a semântica básica que permite estruturar páginas e a sintaxe dos recursos iniciais a serem apresentados é essencial e deve ser além de estudado, praticado à medida que se avança nos estudos em JavaScript.

Conheça os principais símbolos utilizados e aplicações para eles, assim como também terá a oportunidade de compreender os tipos de dados aceitos em páginas web e como se pode utilizá-los na implementação de scripts.

Lembre-se que os estudos não se resumem à leitura do que se apresenta no material, mas à prática de uso dos scripts disponibilizados e modificação e criação de novos scripts para que se possa validar o que está sendo estudado.

**Bons estudos!**



## 1. INTRODUÇÃO AO JAVASCRIPT (JS)

O desenvolvimento de conteúdo para web é uma atividade bastante relevante no mercado de TI e utiliza como base a linguagem de marcação HTML, criada no início da década de 1990, e a linguagem de estilos CSS (*Cascading Style Sheets*), criada nos meados da mesma década que se complementam e oferecem diversos recursos para simplificar a estruturação de páginas web.

Essas linguagens não possuem, entretanto, recursos para o processamento de dados necessário nas páginas web atuais, mais dinâmicas e interativas, ligadas a bancos de dados e capazes de realizar processos com dados obtidos e interações realizadas por usuários.

A linguagem HTML é utilizada para estruturar páginas web organizando elementos multimídia diversos a serem exibidos por página em navegadores, mas de forma limitada apenas no que é chamado de front-end, ou seja, a parte visual e interativa de uma página web, sem realizar processamento de dados ou algum tipo efetivo de programação do chamado *back-end*.

Com o tempo, as páginas que inicialmente eram estáticas e eram compostas basicamente de texto e imagens, foi evoluindo e conteúdo mais elaborado e com maior interatividade e dinamicidade das páginas web criou a necessidade de que surgisse algo complementar ao HTML para o desenvolvimento da parte lógica de uma página exibida em navegadores.

Pensando na evolução do conteúdo web, surgiram linguagens e complementos para as linguagens fundamentais da web como a linguagem JS e a linguagem ActionScript utilizada em *plugins Flash*, já descontinuados.

Segundo Silva (2010), a linguagem teve sua primeira versão disponibilizada também em meados da década de 1990, sendo implementada no navegador Netscape inicialmente, antes da leva de navegadores que ocorreu com a crescente popularização da Internet, criando um novo e promissor nicho de mercado de desenvolvimento de conteúdo para web.

Com a evolução dos conteúdos, a linguagem JS também evoluiu, assim como o HTML e o CSS que tiveram mudanças relevantes em novas versões que foram sendo desenvolvidas ao longo do tempo, adaptando o desenvolvimento de conteúdo às novas tendências do mercado como a inclusão de efeitos, animações, conteúdos dinâmicos como jogos interativos, som e vídeo, etc.

Outra evolução que ocorreu naturalmente foi a utilização da Internet como meio base para comércio e utilização de sistemas diversos, fazendo com que outros tipos de evoluções ocorressem para atender esta outra demanda que logo se tornou o foco de muitas empresas de desenvolvimento de software, abrindo a oportunidade para o surgimento de novas formas de desenvolvimento de software e implementação de novas linguagens e ferramentas para este nicho de mercado.

Linguagens como PHP surgiram para atender também demandas relacionadas ao desenvolvimento de software para web, tendo cada opção suas peculiaridades, como no caso de PHP, que necessita estar hospedado em servidores remotos ou locais configurados para executarem scripts PHP com bancos de dados, e JS que roda diretamente no navegador geralmente.

Silva (2010) cita que de forma complementar ao HTML, JS pode trabalhar de forma dinâmica uma página web, interferindo no que tenha sido originalmente estruturado em HTML, assim como também manipular os estilos CSS definidos pela própria para uma página.

Também é possível controlar o navegador com ações como abrir janelas extras do tipo pop-up para mensagens e anúncios complementares ao conteúdo original de uma página, ajuste de dimensões da janela do navegador, e elementos contidos na janela, como menus e barras de ferramentas, por exemplo.

Como complemento ao conteúdo de uma página web, a linguagem JS permite que dados inseridos em formulários inseridos em páginas web para que estes sejam utilizados em processos como de validação, cálculos, e outros tipos de ações relacionadas aos dados coletados.

O uso de linguagem JS deve ocorrer em conformidade com padrões web que implicam que uma página contendo conteúdo JS não seja obstrutivo, ou seja, mesmo que um navegador não tenha suporte à linguagem, o conteúdo essencial da página em HTML não deve ser impedido de ser visualizado.

Também é sugerido que os scripts em JS sejam apenas complementos para a melhoria da usabilidade em páginas e não apenas para geração de efeitos visuais desnecessários como ocorria no início de seu uso, e sejam escritos em um arquivo separado a ser vinculado ao script HTML para afetar minimamente o conteúdo original em sua estrutura.

Outra característica da programação web que influencia a programação JS, segundo Silva (2010), é a de desenvolvimento em três camadas de estruturação de conteúdos de marcação HTML, outra camada de folhas de estilo, e uma terceira de comportamentos que envolve os scripts JS, sendo que todas acabam sendo interligadas através de links.

Com o passar do tempo, a linguagem foi se atualizando e incorporando diversas inovações para atender as demandas mais atuais de cada época no mercado, resultando numa série de versões que geraram inclusive mudanças em sua nomenclatura.

Uma das evoluções permitiu que a linguagem pudesse ser executada em ambientes diferentes dos navegadores web apenas, tais como o Adobe Acrobat, Apache CouchDB (banco de dados de código-aberto), Node.js (software baseado na máquina virtual V8 da Google fora do navegador).

JS também evoluiu para aceitar programação dita multi-paradigma, ou seja, aceita a escrita de códigos em diferentes padrões como no tradicional imperativo, no orientado a objetos, ou até no declarativo paradigma funcional.

Em meados da década de 1990, também surgiu uma linguagem alternativa baseada em JS e JScript (variação de JS da Microsoft para o Internet Explorer), sob o nome de ECMAScript que em 1998 foi aprovado como norma internacional ISO/IEC 16262 sob o nome ECMA-262 que também foi sofrendo atualizações. É comum os navegadores atuais suportarem a versão 5.1 do ECMA-262, ou a versão 3 em navegadores mais antigos.

Com a padronização, aos poucos a linguagem JS foi se adequando ao padrão ECMA-262, e mesmo que a documentação utilize alguns termos distintos, JS suporta todas as funcionalidades na definição ECMA-262. Esta definição não descreve o chamado DOM (*Document Object Model*), que é padronizado pelo W3C (*World Wide Web Consortium*) que define a maneira na qual os objetos do documento HTML estão expostos no seu script.

A definição ECMAScript em si não representa uma linguagem autossuficiente computacionalmente, pois não contém recursos suficientes para gerar uma aplicação completa, tendo como objetivo, a realização de cálculos e outros tipos de ações em navegadores ou servidores web para hospedar elementos como janelas, caixas de diálogo, menus pop-up, cookies, etc.

## WELCOME TO OUR SITE

The lorem ipsum text is typically a scrambled section of De finibus bonorum et malorum, a .

[Read More](#)

## 2. UM PRIMEIRO SCRIPT

Para se criar scripts em JS é preciso que tenhamos em mente que o HTML estará presente para estruturar os elementos que irão compor uma página web como botões, imagens ou campos de formulários, por exemplo, e complementar este conteúdo realizando processos relacionados a estes elementos como validações, cálculos ou organização de dados obtidos.

A escrita de scripts para páginas web pode ser feita em um editor simples como o bloco de notas contido no Windows ou qualquer outro editor de texto de outros sistemas operacionais, ou pode ser escrito em editores especiais que possuem mais recursos de auxílio no trabalho de desenvolvimento como IDEs especializadas para programação como Eclipse, Netbeans, Visual Studio, dentre outros que possuem a capacidade de auxiliar na escrita de códigos com um assistente de sintaxe, compiladores e interpretadores agregados ou acionados de outros softwares como navegadores geralmente.

Para facilitar os estudos e reduzir os recursos necessários para executar scripts de forma geral, sugere-se que se utilize um editor simples textual que receberá os scripts, e depois, estes devem ser gravados como arquivos em formato HTML para serem interpretados e executados em navegadores web, mesmo que sem acesso à Internet.

Os scripts para páginas web podem ser escritos de diferentes formas, e ao longo dos estudos, serão experimentadas variações possíveis no desenvolvimento de páginas web utilizando scripts que mesclam HTML e JavaScript principalmente.

**TABELA 1 - EXEMPLO DE HTML COM JAVASCRIPT**

<pre>1 &lt;!DOCTYPE html&gt;   &lt;html&gt;     &lt;body&gt;       &lt;h2&gt;JavaScript - Exemplo 1&lt;/h2&gt;       &lt;hr&gt;       &lt;p id="ex1"&gt;Disciplina&lt;/p&gt;       &lt;button type="button" onclick='document.getElementById("ex1").innerHTML =       "Desenvolvimento Web Javascript e Frameworks"'&gt;Clique&lt;/button&gt;     &lt;/body&gt;   &lt;/html&gt;</pre>
---

**Fonte:** O autor (2022).

Observando o exemplo da tabela 1, temos um script completo que tem por base o padrão HTML 5 que possui o mínimo necessário para ser aceito por navegadores, e dentro deste script, vemos claramente a estrutura base do HTML sendo utilizada com as tags <!DOCTYPE html>, <html>, <body>, <h2>, <hr>, <p>, e <button> que representam os marcadores para estruturação do layout da página web.

As tags essenciais <!DOCTYPE html>, <html> e <body> definem a identificação da estrutura da página, e as demais tags estruturam elementos de composição do layout, sendo opcionais e ajustáveis para cada diferente página a ser criada.

O detalhe relacionado ao JS é o uso do método getElementById() que permite adicionar dinamicidade a scripts HTML, e no caso do exemplo, executa a ação de alterar o conteúdo da tag <p> identificada como ex1 para outro conteúdo indicado no método no momento em que o botão criado na tag que contém o método é clicado (onclick).

A criação de scripts mesclando HTML e JavaScript, além da inclusão de estilos CSS se desejado, é bastante comum atualmente no desenvolvimento de sites, e a forma como se pode adicionar código JS pode ser melhor organizada se forem utilizadas tags <script> e </script> para delimitar o que for JS ao invés de HTML.

Para que seja possível esta comunicação direta do script JS com o conteúdo do navegador, utiliza-se o método innerHTML = "" que exibe o conteúdo entre aspas no local da estrutura da página web indicado no restante da instrução.

**TABELA 2 - EXEMPLO DE HTML COM JAVASCRIPT**

1	<!DOCTYPE html> <html> <head>
2	<script> function funcao() { document.getElementById("ex2").innerHTML = " Desenvolvimento Web Javascript e Frameworks "; } </script>
3	</head> <body> <h2>JavaScript - Exemplo 2</h2> <hr> <p id="ex2">Disciplina</p> <button type="button" onclick="funcao()">Clique</button> </body> </html>

Fonte: O autor (2022).

Partindo da nova versão do script da Tabela 1, o script da Tabela 2 traz o uso de dois recursos importantes em JS que são o uso das tags `<script>` e `</script>` para delimitar o código e a declaração e uma função para conter a funcionalidade desejada em JS, permitindo que o script seja otimizado, e utilizando-se uma função, possa-se chamar a função quantas vezes desejado sem a necessidade de repetir todo o código contido na função dentro do script.

Outro aspecto relevante é o uso da tag `<head>` para que a função em JS seja inserida nesta área, sendo uma boa alternativa por ficar fora do corpo principal do script HTML, sem alterar a funcionalidade do código JS.

Alguns detalhes da chamada sintaxe, que representa o conjunto de regras para a escrita de instruções em cada linguagem de programação, e que também em JavaScript devem ser observados para facilitar os estudos ao longo do material.

É necessário o uso de ponto e vírgula ao final de cada instrução como indicativo do final da mesma, pois as quebras de linhas do editor não possuem essa função de encerramento de instruções como ocorre em algumas linguagens.

**TABELA 3 - EXEMPLO DE HTML COM JAVASCRIPT**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 3&lt;/h2&gt; &lt;hr&gt; &lt;p id="ex3"&gt;Disciplina&lt;/p&gt; &lt;button type="button" onclick="funcao()"&gt;Clique&lt;/button&gt;</pre>
2	<pre>&lt;script&gt; function funcao() {   document.getElementById("ex3").innerHTML = "Desenvolvimento Web Javascript e Frameworks"; } &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

Avaliando o exemplo da tabela 3, temos o retorno do código JS para a seção do corpo da página indicada pela tag `<body>`, mostrando que os scripts JS podem ser adicionados sem problemas nesta seção, ou na seção de cabeçalho `<head>` como mostrado na tabela 2.

As duas formas de inserção de scripts JS respeitam a chamada semântica de construção de scripts em HTML, e não geram problemas na exibição de páginas web, desde que inseridos adequadamente sem prejudicar os comandos HTML em si.



### 3. AVANÇANDO EM JAVASCRIPT

Outro aspecto da sintaxe da linguagem que deve ser respeitado e corretamente utilizado é a delimitação de instruções que pertencem a outra instrução através de chaves que indicam o bloco de instruções que deve ser associado a uma função, por exemplo.

Para a construção de instruções em JavaScript, assim como em todas as linguagens de programação, existe um conjunto de palavras reservadas que geram ações e podem ou não necessitar de dados complementares chamados de parâmetros para serem funcionais, reforçando que todos devem ser utilizados seguindo as regras de sintaxe e semântica de JS.

**TABELA 4 - EXEMPLO DE HTML COM JAVASCRIPT EM ARQUIVO EXTERNO**

	<!DOCTYPE html>
	<html>
	<body>
1	<h2>JavaScript - Exemplo 4</h2>
	<hr>
	<p id="ex4">JavaScript</p>
	<button type="button" onclick="funcao()">Clique</button>
2	<script src="extra.js"></script>
3	</body>
	</html>
4	function funcao() { document.getElementById("ex4").innerHTML = "Desenvolvimento Web Javascript e Frameworks"; }

Fonte: O autor (2022).

Neste outro exemplo trazido na tabela 4, o conteúdo da quarta linha da tabela representa um script JS que deve ser gravado em um arquivo chamado extra.js em algum editor de texto para que possa ser aberto pela tag `<script src="extra.js">` que busca pelo arquivo com este nome, sendo que o nome é apenas sugestivo e pode ser alterado sem problemas, desde que na indicação da tag e no nome do arquivo gravado propriamente dito.

Com isso, o arquivo original HTML fica menor, mais facilmente comprehensível e ainda seguindo as orientações do padrão web de utilização de scripts CSS e JS em arquivos externos ao arquivo HTML de marcação de layout de páginas.

É possível que os arquivos contendo scripts externos sejam utilizados a partir de fontes online remotas ao invés de terem que estar juntos ao arquivo HTML fazendo com que este conteúdo externo possa ser acessado por outros scripts HTML em qualquer lugar do mundo, por exemplo.

O HTML exibe conteúdo numa página utilizando tags específicas como `<p>` de forma bastante simples, mas a linguagem JS utiliza métodos específicos para a exibição de saídas de dados como `document.write()` e `window.alert()`.

**TABELA 5 - EXEMPLO DE HTML COM JAVASCRIPT**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 5&lt;/h2&gt; &lt;hr&gt; &lt;button type="button" onclick="funcao()"&gt;2 + 2&lt;/button&gt; &lt;p id="ex5"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; document.write (2 + 2); document.getElementById("ex5").innerHTML = 2 + 2;  function funcao() {   window.alert(2 + 2); }  &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

Fonte: O autor (2022).

O exemplo da tabela 5 traz três variantes que podem ser utilizadas para exibir conteúdo a usuários, todos inseridos na área de script (2), sendo a primeira a opção `document.write` que quando utilizada, limpa todo o conteúdo da página antes de realizar a exibição dos dados, sendo uma alternativa indicada em casos onde o conteúdo original da página não seja mais necessário.

A segunda forma se baseia na capacidade de alterar a propriedade `innerHTML` de um elemento HTML para exibir dados em HTML, sem eliminar o conteúdo da página neste caso, sendo em método simples e que não altera os demais elementos de uma página.

A terceira forma se baseia no uso do método `window.alert`, que permite que dados sejam exibidos a usuários através de janelas popup que surge sobre o navegador que exibe a página, trazendo os dados solicitados como se complementassem o conteúdo da página sem alterar seu conteúdo atual ou layout, usando uma caixa de mensagem apenas. Um detalhe é que se pode, em casos de métodos baseados em Windows, que esta seja omitida, e apenas o que deve ser escrito depois do método Windows seja necessário (`alert()`).

Para se construir páginas web, a linguagem HTML não necessita de muitos recursos de programação propriamente dita, e até por isso, acaba sendo desconsiderada como, mas conteúdos produzidos em HTML necessitam seguir determinada semântica na escrita de scripts e sintaxes para cada instrução que gere algum elemento para o layout de uma página.

Já a linguagem JavaScript, por ser mais completa, possui os elementos mais comuns utilizados em linguagens de programação como estruturas de dados, expressões lógicas e estruturas de controle de fluxo, dentre outros recursos relevantes.



#### 4. COMPLETANDO O PRIMEIRO CICLO DE CONCEITOS EM JAVASCRIPT

Para iniciar o aprendizado da programação em si na linguagem JS, é importante conhecer as bases da programação na linguagem como os recursos para construção de expressões e para declaração de estruturas de dados, para depois avançar nos recursos de como controlar o fluxo dos processos durante a execução de um script JS.

Expressões são utilizadas para a realização de cálculos matemáticos e obtenção de novos dados a partir de outros já disponíveis, ou para avaliação lógica como verdadeira ou falsa, uma expressão que tenha como base comparações entre dados, por exemplo, e para a construção de expressões assim, são utilizados diferentes tipos de operadores representados por símbolos que não devem ser utilizados para outras finalidades, sendo então reservados.

**TABELA 6 - OPERADORES MATEMÁTICOS EM JAVASCRIPT**

OPERADO R	FINALIDADE
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
**	Exponenciação (potência)
%	Módulo (resto de uma divisão)
++	Incremento (aumenta o valor de uma estrutura de dados em 1 unidade)
--	Decremento (reduz o valor de uma estrutura de dados em 1 unidade)

**Fonte:** Adaptado de: W3C School (online).

Os operadores trazidos na tabela 6 são utilizados para a realização de operações matemáticas em geral, podendo ser utilizados em expressões matemáticas comuns para atribuição para estruturas de dados, ou expressões lógicas que podem definir como deve ocorrer a execução de um script JS.

Existem os operadores mais comuns como os de soma, subtração, multiplicação e divisão, mas existem também operadores como a exponenciação e módulo que são operações menos comuns da matemática, e os operadores de incremento e decremento que são mais específicos da computação para controle de iterações, por exemplo.

**TABELA 7 - EXEMPLO DE HTML COM JAVASCRIPT USANDO OPERADORES MATEMÁTICOS**

1	<pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 6&lt;/h2&gt; &lt;hr&gt;  1. Soma padrão&lt;br&gt; &lt;p id="ex61"&gt;&lt;/p&gt; 2. Concatenação de texto &lt;br&gt; &lt;p id="ex62"&gt;&lt;/p&gt; 3. Soma com outra sintaxe &lt;br&gt; &lt;p id="ex63"&gt;&lt;/p&gt; </pre>
2	<pre> &lt;script&gt; let a, b, c; a = 2; b = 2; c = a + b; document.getElementById("ex61").innerHTML = "2 + 2 = " + c;  document.getElementById("ex62").innerHTML = "HTML" + " " + " " + "JavaScript";  var e = 3; var f = 3; var g = e + f; document.getElementById("ex63").innerHTML = "3 + 3 = " + g; &lt;/script&gt; </pre>
3	<pre> &lt;/body&gt; &lt;/html&gt; </pre>

**Fonte:** O autor (2022).

Tomando por base o exemplo da tabela 7, é importante destacar o papel estruturante de layout e indicação de elementos a serem posicionados em uma página, pois os scripts em JS para que possam ser exibidos na página, necessitam de ligação com elementos HTML, sendo no exemplo, indicados pelos nomes de referência ex61, ex62, ex63 e ex64 que funcionam como identificadores para ligação em tags `<p>` e instruções JS.

O exemplo traz três variações de uso de operadores para ilustrar um pouco do que pode ser feito com os operadores matemáticos e o primeiro exemplo declara três variáveis a, b, c que são utilizadas para receberem valores numéricos em a e b que são somados e o resultado armazenado em c para exibição na página.

O segundo trecho do script traz um caso especial que é o uso do operador + para concatenar textos de forma a se tornarem um apenas na exibição da página sem que sejam utilizados valores numéricos, e por fim, uma forma variante de declaração e atribuição de valores a variáveis é trazida na terceira parte do script usando a palavra reservada var.

#### 4.1 Tipos de Dados

Todos os dados que podem ser utilizados ou obtidos através de páginas web possuem algum tipo definido, sejam valores numéricos ou texto, por exemplo, e cabe ao desenvolvedor que escreve os scripts JS definir como serão definidos estes tipos pela forma como os dados são atribuídos a estruturas de dados.

Basicamente, a sintaxe como os dados são atribuídos definem quais seus tipos, e com isso, a correta indicação de como é realizado o processo em scripts gera o bom funcionamento dos mesmos ou problemas no uso de conteúdos em páginas web.

Até objetos podem ser declarados e já terem dados associados para definir os tipos determinados para cada um de seus chamados atributos ou campos que juntos, definem a estrutura de dados como um todo, que permite a mescla de dados de diferentes tipos sob um mesmo nome, como se pode observar no exemplo da tabela 17.

**TABELA 17 - EXEMPLO DE HTML COM JAVASCRIPT USANDO OPERADORES LÓGICOS**

1	<!DOCTYPE html> <html> <body> <h2>JavaScript - Exemplo 10</h2> <hr>  <p id="ex101"></p> <p id="ex102"></p> <p id="ex103"></p> <p id="ex104"></p>
2	<script> let altura = 1.60; var peso = 60; let cadastro = {Nome:"Fulano", Sobrenome:"de Souza"};  document.getElementById("ex101").innerHTML = cadastro.Nome + " " + cadastro.Sobrenome; document.getElementById("ex102").innerHTML = "Peso: " + peso; document.getElementById("ex103").innerHTML = "Altura: " + altura;  IMC = peso / (altura * altura); document.getElementById("ex104").innerHTML = "IMC: " + IMC; </script>
3	</body> </html>

Fonte: O autor (2022).

Neste exemplo da tabela 17, são criadas diversas estruturas de dados para exemplificar a declaração e atribuição de dados às estruturas, mostrando também que é possível declarar essas estruturas utilizando as palavras reservadas let ou var, ou ainda não utilizar palavra reservada alguma, indicando apenas o nome da estrutura, e em seguida, atribuindo dados a mesma, sejam puros ou objetos como mostrado no exemplo.

As estruturas de dados altura e cadastro são declaradas utilizando a palavra reservada let, sendo a primeira do tipo variável, e a segunda, um objeto por conter subestruturas Nome e Sobrenome, enquanto a variável peso usa a palavra var para mostrar uma variação de declaração de estruturas de dados.

Por fim, a variável IMC é declarada sem o uso de palavras reservadas, sendo possível identificá-la como variável por logo após seu nome ser atribuída uma expressão matemática que utiliza os dados de outras duas variáveis declaradas anteriormente, peso e altura.

**TABELA 18 - EXEMPLO DE HTML COM JAVASCRIPT USANDO OPERADORES LÓGICOS**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 11&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;Digite um valor para X: &lt;input id="x" value="0" /&gt; &lt;p&gt;Digite um valor para Y: &lt;input id="y" value="0" /&gt; &lt;br&gt; &lt;button onclick="funcao(x, y)"&gt;Multiplique&lt;/button&gt; &lt;p id="ex11"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; function funcao() {   let x = document.getElementById("x").value;   let y = document.getElementById("y").value;   document.getElementById("ex11").innerHTML = "Resultado = " + x * y; } &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

Fonte: O autor (2022).

Para completar o uso de estruturas de dados, o exemplo da tabela 18 traz um script que utiliza variáveis x e y para receber dados do formulário HTML, de forma que possam ser utilizados no processamento do script JS de forma a gerar um novo dado que representa o resultado da multiplicação entre ambos os dados que são exibidos pelo método innerHTML.

## SAIBA MAIS

Conhecer o que cada linguagem possui de recursos disponíveis para que se possa desenvolver soluções computacionais como softwares ou sites é algo que é construído ao longo do aprendizado e posterior prática, pois existe uma gama de recursos que são utilizados regularmente, mas existe um outro subconjunto de recursos que são utilizados apenas esporadicamente, em virtude de necessidades mais específicas e pouco usuais da rotina do desenvolvedor.

Assim, fica claro que não é preciso conhecer todos os recursos de uma linguagem prontamente, e sem necessidade de consultas, pois é algo desnecessário, ainda mais nos dias atuais, mas é essencial saber onde encontrar a ajuda necessária, seja em colegas de trabalho, ou fontes confiáveis de consulta.

Uma ótima fonte de referência para a linguagem JavaScript se encontra na seção de referência da linguagem no site da [w3schools.com](https://www.w3schools.com) que além de seções separadas para descrever cada detalhe relevante da linguagem JavaScript, possui uma página dedicada a unir toda a referência disponível para a linguagem no local.

**Fonte:** Adaptado de: W3Schools (online).

## REFLITA

A ética é fundamental para o desenvolvimento de conteúdo para web, e por isto, seja correto naquilo que se propuser a fazer, mesmo que lhe peçam o contrário.

**Fonte:** O autor (2022).

# CONSIDERAÇÕES FINAIS

Após ter tido contato com a base da linguagem JavaScript, é chegado o momento de praticar o que foi estudado, e para isto, não há método mais eficiente que praticar, seja revendo os scripts exemplo da unidade, alterando exemplos prontos, ou criando seus próprios scripts.

O importante ao final de um primeiro contato com a linguagem é a de não deixar passar a oportunidade de firmar uma boa base de conhecimento para avançar nos estudos e utilizar o que foi visto como conteúdo base para o que vier na sequência, criando um aprendizado que se complementa a cada nova etapa e não etapas isoladas que sejam estudadas isoladamente.

Por sorte, os estudos nesta disciplina facilitam muito esta dinâmica e o aprendizado deve ocorrer desta forma mais naturalmente que em alguns outros casos, pelas suas características práticas e de avanço progressivo e acumulativo nos conteúdos.

Siga firme em seus estudos na próxima etapa que trará conceitos e recursos complementares da linguagem JavaScript, que irão tornar mais completos os conhecimentos adquiridos ao final da disciplina.

## LEITURA COMPLEMENTAR

Ao iniciar os estudos em linguagem Java, nos deparamos com muita informação e nem sempre estas são diretamente relacionadas ao que se estuda em determinado momento, e o que ocorre, é que facilmente nos deparamos com informações paralelas relevantes ou não que podem complementar ou simplesmente prejudicar o foco no aprendizado.

Um tipo de situação que pode ocorrer facilmente ao longo dos estudos em JavaScript é imaginar que se esteja estudando Java na verdade, sendo que são duas linguagens de programação que podem atuar em segmentos parecidos dentro do desenvolvimento de soluções computacionais, mas são duas ferramentas alternativas de desenvolvimento que possuem muitas diferenças entre si, mantendo as duas em pontos distintos dentro do universo de linguagens de programação.

Uma diferença simples e que logo de início já mostra que não são a mesma linguagem é o fato de Javascript normalmente ser ligado ao HTML em páginas web, e o Java poder desenvolver softwares para web, mas sem necessitar do HTML.

**Fonte:** Java (online).

## MATERIAL COMPLEMENTAR



### LIVRO

**Título:** JavaScript - Guia do Programador: Guia completo das funcionalidades de linguagem JavaScript.

**Autor:** Maurício Samy Silva.

**Editora:** Novatec.

**Sinopse:** Livro que trata dos fundamentos da linguagem JavaScript, do básico para web à programação orientada a objetos, seguindo recomendações do W3C e ECMAScript.



### FILME/VÍDEO

**Título:** Helvetica.

**Ano:** 2007.

**Sinopse:** Um documentário sobre os conceitos de tipografia e design gráfico associados a cultura de forma geral.

# UNIDADE II

## Recursos da Linguagem Javascript

Professor Especialista Ronie Cesar Tokumoto



### Plano de Estudo:

- Conhecendo Mais Operadores;
- Regras Matemáticas para Uso de Operadores;
- Utilizando Funções;
- Objetos e Suas Propriedades.

### Objetivos da Aprendizagem:

- Compreender os diferentes tipos de operadores em JavaScript;
- Conhecer regras de precedência para o correto uso de operadores;
- Conceituar e contextualizar o uso de funções;
- Compreender e objetos em JavaScript.

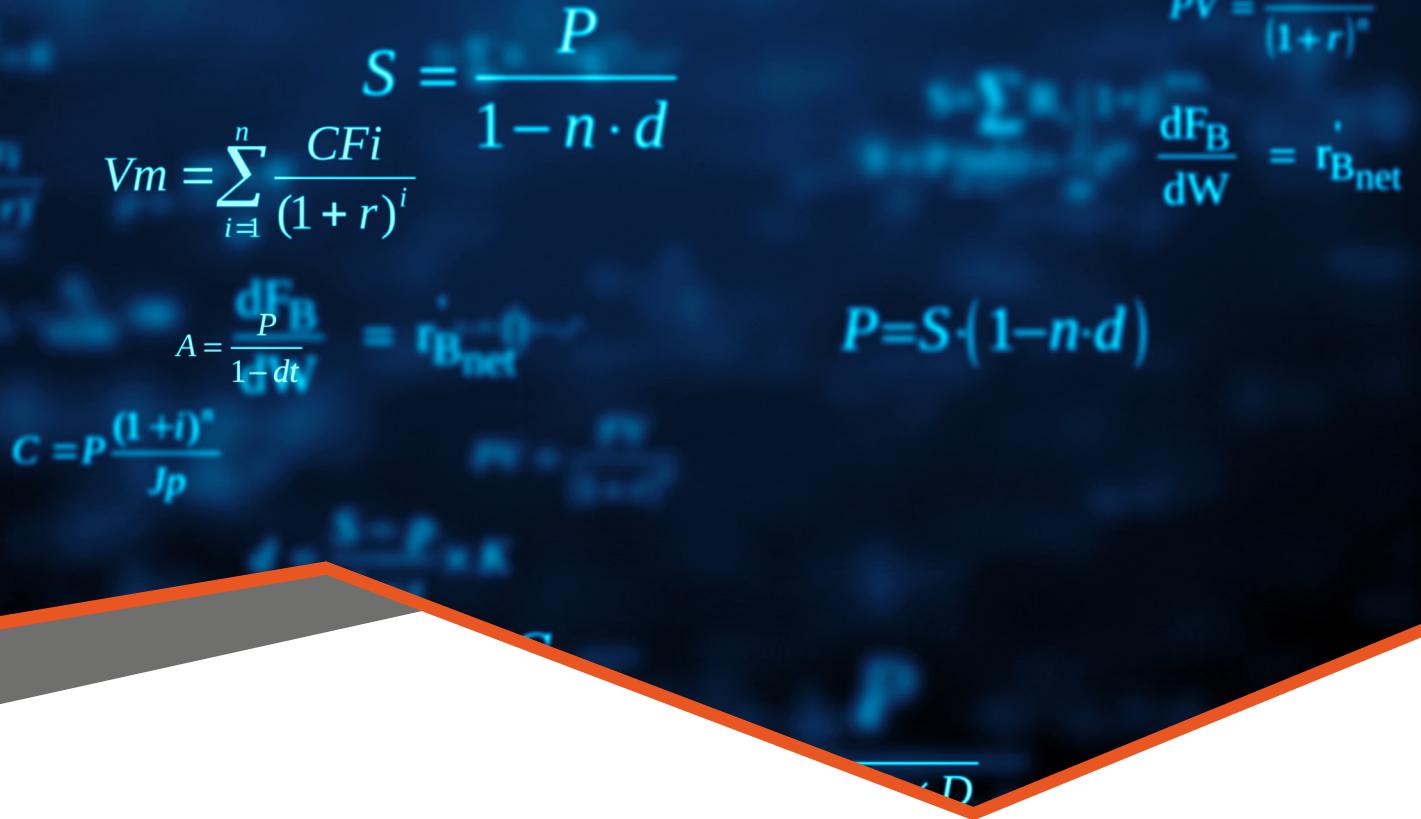
# INTRODUÇÃO

Após conhecer alguns conceitos e recursos iniciais da implementação de scripts JavaScript combinados com scripts de marcação de páginas web em HTML, é importante avançar nos estudos e se aprofundar em conceitos complementares mais complexos e também importantes para o aprendizado da programação.

O conteúdo que segue nesta nova unidade não difere do que foi estudado na unidade anterior, e sim, complementa o que foi estudado anteriormente, pois é um conteúdo acumulativo em função de sempre ser necessária a utilização de elementos estudados na unidade anterior nesta também.

São tratados diferentes tipos de operadores para a construção de scripts e sua adequada utilização, assim como reforçar a forma como se pode criar scripts estruturados em funções, e por fim, compreender como se pode criar objetos que podem conter estruturas de dados de variados tipos sob um mesmo nome.

Mantenha seu foco no aprendizado da linguagem JavaScript que é algo muito relevante para o mercado de desenvolvimento de software, e consequentemente, para sua formação profissional.



## 1. CONHECENDO MAIS OPERADORES

Operadores de atribuição são responsáveis por permitir que dados ou resultados de expressões possam ser diretamente inseridos em estruturas de dados como variáveis ou outros, sendo que valores anteriormente contidos nestas estruturas de dados são afetados pelo processo de atribuição.

**TABELA 1 - OPERADORES DE ATRIBUIÇÃO EM JAVASCRIPT**

OPERADOR	FINALIDADE
=	Atribuição direta
+=	Atribuição após soma
-=	Atribuição após subtração
*=	Atribuição após multiplicação
/=	Atribuição após divisão
%=	Atribuição após módulo
**=	Atribuição após exponenciação

**Fonte:** Adaptado de: W3C School (online).

Assim como os operadores matemáticos, os operadores de atribuição são muito importantes na programação, pois permitem que dados sejam associados a estruturas de dados e com as opções disponíveis em JS, a implementação de scripts é facilitada pela redução da quantidade de instruções necessárias em certos casos usando os operadores que realizam uma operação matemática, e depois atribuem o resultado da operação à variável ou outro tipo de estrutura de dado.

**TABELA 2 - EXEMPLO DE HTML COM JAVASCRIPT USANDO OPERADORES DE ATRIBUIÇÃO**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 7&lt;/h2&gt; &lt;hr&gt; &lt;p id="ex7"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt;   let texto = "Aprendendo ";   texto += "JavaScript";   document.getElementById("ex7").innerHTML = texto; &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

No exemplo da tabela 2, o operador `+=` realiza a concatenação do conteúdo que foi anteriormente atribuído à variável `texto` com o complemento da mensagem, para depois atribuir a frase completa à variável e este poder ser exibido na página em seguida.

Operadores relacionais são utilizados para a construção de expressões lógicas, essenciais para o desenvolvimento dos processos em um script JS, pois são responsáveis por permitir que decisões sejam tomadas automaticamente ou processos ocorram em função de escolhas proporcionadas por estas decisões.

**TABELA 3 - OPERADORES RELACIONAIS EM JAVASCRIPT**

OPERADOR	FINALIDADE
<code>==</code>	Igualdade (= matemático)
<code>===</code>	Igualdade em valor e tipo de dado
<code>!=</code>	Diferente (não igual)
<code>!==</code>	Diferente em valor e tipo
<code>&gt;</code>	Maior
<code>&lt;</code>	Menor
<code>&gt;=</code>	Maior ou igual
<code>&lt;=</code>	Menor ou igual
<code>?</code>	Operador ternário (<condição> ? <X> : <Y>;)

**Fonte:** Adaptado de: W3C School (online).

Conforme Silva (2010), a correta utilização dos operadores relacionais afeta diretamente a funcionalidade dos scripts, pois além da lógica que deve ser observada na escolha de operadores e construção de expressões, a sintaxe também é importante e bem padronizada em linguagens como JS em que um dado puro ou contido em uma estrutura de dados é comparado com outro dado puro ou contido em outra estrutura de dados geralmente.

**TABELA 4 - EXEMPLO DE HTML COM JAVASCRIPT USANDO OPERADORES RELACIONAIS**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 8&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;Digite sua idade atual:&lt;/p&gt; &lt;input id="idade" value="0" /&gt; &lt;br&gt; &lt;button onclick="funcao()"&gt;Clique&lt;/button&gt; &lt;p id="ex8"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; function funcao() {   let idade = document.getElementById("idade").value;   let maioridade = (idade &gt;= 18) ? "Maior": "Menor";   document.getElementById("ex8").innerHTML = maioridade + " de idade."; } &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

O exemplo da tabela 4 é bastante completo e traz uma série de conceitos importantes que podem ser tratados como o uso propriamente dito do operador relacional `>=` que é utilizado para comparar a idade digitada por um usuário no campo de entrada de um formulário e armazenada na variável `idade`.

Este valor é então comparado com o valor base 18, e caso seja realmente maior ou igual a 18 o valor digitado pelo usuário, uma mensagem informa o mesmo que ele é maior de idade, mas caso a comparação resulte falso pela idade digitada ser 17 anos ou menos, a mensagem exibida informa o usuário que ele é menor de idade.

Isso é possível pelo uso do operador ternário que é capaz de avaliar a condição e decidir por si só se deve ser atribuído a palavra Maior ou a palavra Menor para a variável maioridade que depois ainda terá a palavra armazenada concatenada com o texto complementar com base no uso do operador `+=` no método `innerHTML()`.

Além dos operadores relacionais que retornam resultados verdadeiros ou falsos em expressões lógicas, existem os próprios operadores lógicos que também retornam estes tipos de valores, mas não constroem expressões diretamente com dados geralmente, sendo na verdade utilizados mais como auxiliares em determinadas situações.

**TABELA 5 - OPERADORES LÓGICOS EM JAVASCRIPT**

OPERADOR	FINALIDADE
<code>&amp;&amp;</code>	E lógico
<code>  </code>	OU lógico
<code>!</code>	Negação lógica

**Fonte:** Adaptado de: W3C School (online).

Os operadores lógicos também são importantes na implementação de scripts que envolvam a avaliação de expressões lógicas, pois em muitos casos, é necessário que mais de uma avaliação seja utilizada para avaliar uma condição completa como em casos onde é preciso avaliar o conteúdo de mais de uma variável para que uma decisão seja tomada.

**TABELA 6 - EXEMPLO DE HTML COM JAVASCRIPT USANDO OPERADORES LÓGICOS**

1	<pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 9&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;Digite sua idade atual:&lt;/p&gt; &lt;input id="idade" value="0" /&gt; &lt;p&gt;Digite seu sexo (M ou F):&lt;/p&gt; &lt;input id="sexo" value="F" /&gt; &lt;br&gt; &lt;button onclick="funcao()"&gt;Clique&lt;/button&gt; &lt;p id="ex9"&gt;&lt;/p&gt; </pre>
2	<pre> &lt;script&gt; function funcao() {   let idade = document.getElementById("idade").value;   let sexo = document.getElementById("sexo").value;   let situacao = (idade &gt;= 18 &amp;&amp; sexo == "F") ? "Aprovado": "Reprovado";   document.getElementById("ex9").innerHTML = situacao + " na etapa inicial."; } </pre>
3	<pre> &lt;/body&gt; &lt;/html&gt; </pre>

**Fonte:** O autor (2022).

O exemplo da tabela 6 amplia o exemplo da tabela 5 e acrescenta uma condição extra para avaliação da condição que antes verificava apenas a idade do usuário, e agora verifica também o sexo do mesmo, sendo os dois critérios agrupados com o uso do operador **&&**, que permite que as duas condições possam ser avaliadas ao mesmo tempo, e um resultado único verdadeiro ou falso obtido a partir desta avaliação, direcionando o conteúdo a ser armazenado na variável situação que terá a informação de aprovação ou reprovação de um usuário neste exemplo.

Segundo Slilva (2010), Operadores de avaliação de tipos são ferramentas úteis para o desenvolvimento de scripts, pois através deles, é possível verificar quais tipos de dados são aceitos por cada estrutura de dado utilizável em JavaScript, permitindo ações diversas como validações de dados a serem atribuídos a estruturas de dados ou verificar entradas indevidas de dados em formulários.

**TABELA 7 - OPERADORES DE VERIFICAÇÃO DE TIPOS EM JAVASCRIPT**

OPERADOR	FINALIDADE
<b>typeof</b>	Devolve o tipo de uma estrutura de dados (variável)
<b>instanceof</b>	Devolve verdadeiro se um objeto é uma instância de um tipo de objeto

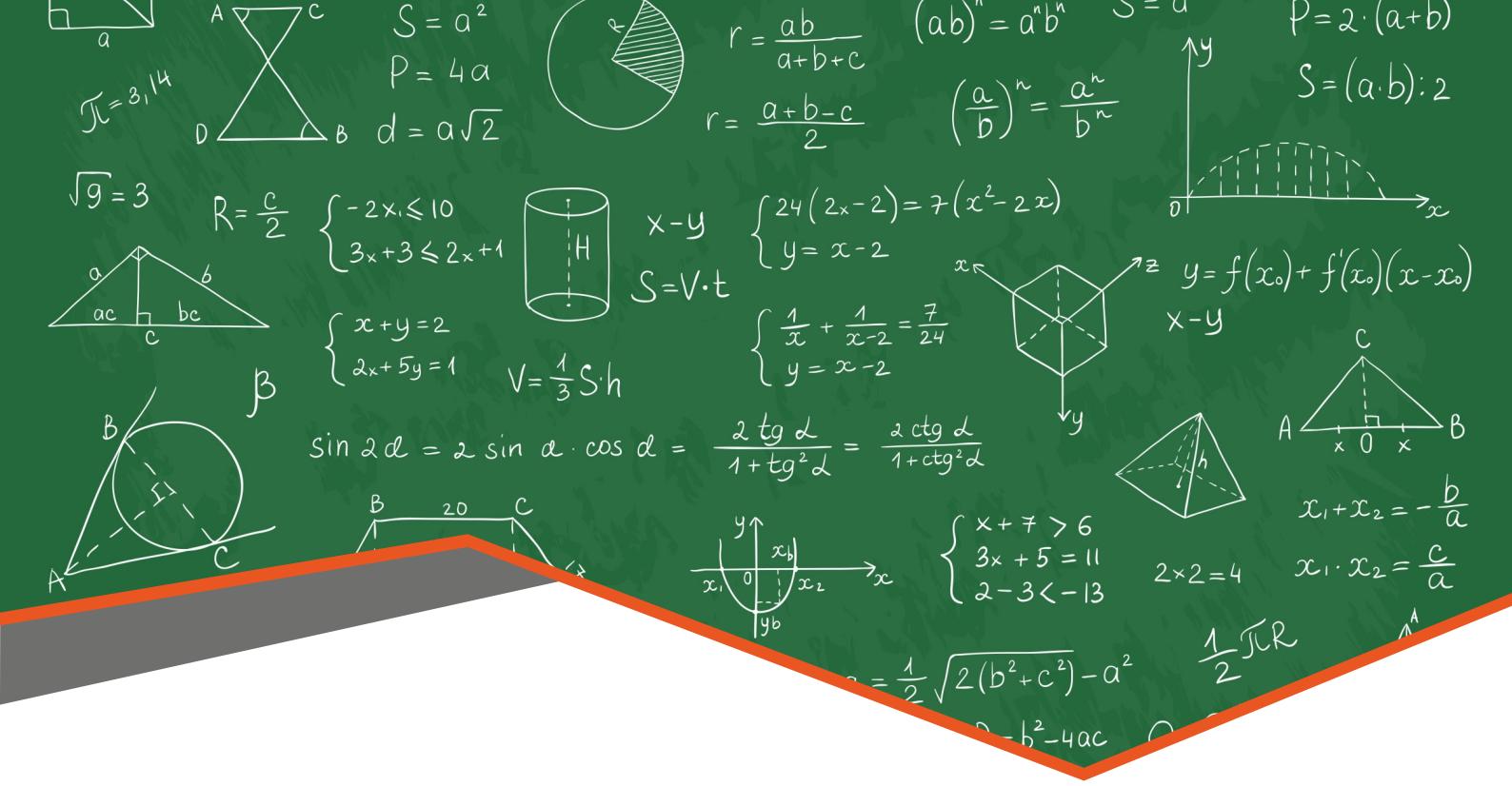
**Fonte:** O autor (2022).

Operadores binários de 32 bits representam tipos especiais de operadores que trabalham com dados a nível de bit, ou seja, da forma como estes são efetivamente armazenados em hardware, permitindo ajustes bastante específicos em dados para operações como comparação de sequências de bits transmitidas pela web, por exemplo.

**TABELA 8 - OPERADORES BINÁRIOS EM JAVASCRIPT**

OPERADOR	FINALIDADE
<b>&amp;</b>	E binário
<b> </b>	OU binário
<b>~</b>	Negação binária
<b>^</b>	OU Exclusivo binário
<b>&lt;&lt;</b>	Deslocamento (shift) para esquerda
<b>&gt;&gt;</b>	Deslocamento (shift) para direita
<b>&gt;&gt;&gt;</b>	Deslocamento (shift) para direita desconsiderando negativos

**Fonte:** Adaptado de W3C School (online).



## 2. REGRAS MATEMÁTICAS PARA USO DE OPERADORES

Após ter conhecido os principais tipos e opções de operadores disponíveis para desenvolvimento de scripts em JS, é preciso entender que existem regras para seu correto uso, e além de respeito pela sintaxe de construção de expressões, é preciso respeitar regras de precedência entre operadores, delimitadores e demais símbolos relevantes que influenciam diretamente os resultados que podem ser obtidos em expressões de todo tipo.

**TABELA 9- PRECEDÊNCIA DE ELEMENTOS EM JAVASCRIPT**

OPERADOR	ORDEM - FINALIDADE
( )	Delimitadores para agrupamento para expressões
.	Indicadores de atributos de objetos
[ ]	Delimitadores para elementos de listas
( )	Indicadores de parâmetros de funções
new	Criação de novas instâncias de objetos
++	Incremento pós-fixado
--	Decremento pós-fixado
++	Incremento pré-fixado

--	Decremento pré-fixado
!	Negação lógica
typeof	Tipo de estrutura de dado
**	Exponenciação
*	Multiplicação
/	Divisão
%	Módulo
+	Soma
-	Subtração
<<	Deslocamento para esquerda
>>	Deslocamento para direita
>>>	Deslocamento para direita sem sinal
<	Menor
<=	Menor ou igual
>	Maior
>=	Maior ou igual
in	
instanceof	Verifica se objeto é uma instância de um tipo de objeto (Object, Array)
==	Igual

==	Igual em valor e tipo
!=	Diferente
!==	Diferente em valor e tipo
&	E binário
^	OU Exclusivo binário
	OU binário
&&	E lógico
	OU lógico
??	Escolhe por um primeiro valor não nulo
? :	Operador ternário condicional
+=	Soma com atribuição
/=	Divisão com atribuição
-=	Subtração com atribuição
*=	Multiplicação com atribuição
%=	Módulo com atribuição
<<=	Deslocamento para esquerda com atribuição
>>=	Deslocamento para direita com atribuição
>>>=	Deslocamento para direita sem sinal com atribuição
&=	E binário com atribuição
^=	OU Exclusivo binário com atribuição
=	OU binário com atribuição
yield	Avalia retornos em funções
,	Múltiplos dados ou expressões em conjunto

**Fonte:** Adaptado de: W3C School (online).

Após ter acesso a uma lista bastante completa de operadores e outros símbolos importantes para a construção de scripts, é importante ter sempre em mente que todos possuem finalidades restritas e devem ser utilizados de forma adequada, seguindo as regras da linguagem em relação a sua semântica e sintaxe.

A escrita de scripts funcionais depende do bom uso de símbolos, operadores e palavras reservadas para que estruturas de dados sejam declaradas e utilizadas corretamente, assim como expressões sejam corretamente escritas e gerem resultados esperados, sendo estes, componentes importantes da implementação software em quaisquer linguagens de programação.



### 3. UTILIZANDO FUNÇÕES

A escrita de scripts em JS para implementação de soluções para web é algo que pode gerar duas situações comuns do desenvolvimento de software, sendo uma delas, a da geração de grandes quantidades de linhas contendo instruções em softwares mais complexos.

Além disso, escrever scripts pode também gerar uma imensa repetição de código desnecessária que pode ser reduzida com o uso de um mecanismo muito importante que permitiu uma grande evolução na programação como um todo quando foi desenvolvida a décadas.

A chamada programação com base no paradigma imperativo que representa a base da programação da maioria das linguagens de programação gera códigos inteiros seguindo uma ordem sequencial lógica que resolve problemas, sem permitir que o mesmo seja, por exemplo, fracionado.

Com o tempo, surgiu o conceito de programação estruturada que trouxe a possibilidade de fracionamento de códigos completos em subcódigos menores, permitindo que partes repetitivas pudessem ser inseridas como blocos de instruções em estruturas mais independentes do código principal, podendo ser acionadas e utilizadas livremente a qualquer momento da execução pelo código principal ou outras chamadas sub-rotinas.

**TABELA 10 - EXEMPLO DE HTML COM JAVASCRIPT USANDO FUNÇÕES**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 12&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;Digite um valor para a primeira nota:&lt;/p&gt; &lt;input id="n1" value="0" /&gt; &lt;p&gt;Digite um valor para a segunda nota:&lt;/p&gt; &lt;input id="n2" value="0" /&gt; &lt;p&gt;Digite um valor para a terceira nota:&lt;/p&gt; &lt;input id="n3" value="0" /&gt; &lt;br&gt; &lt;button onclick="calculaMedia (n1, n2, n3)"&gt;Média&lt;/button&gt; &lt;p id="ex12"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; function calculaMedia() {   let n1 = document.getElementById("n1").value;   let n2 = document.getElementById("n2").value;   let n3 = document.getElementById("n3").value;   document.getElementById("ex12").innerHTML = "Resultado = " + ( parseInt(n1) +   parseInt(n2) + parseInt(n3)) / 3; } &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

Fonte: O autor (2022).

O exemplo trazido na tabela 10 oferece uma ideia de como se pode estruturar um script com base na inserção de instruções em um bloco delimitado por chaves que define o conteúdo da função nomeada calculaMedia que ao receber seus três parâmetros, soma-os e depois divide o resultado da soma por 3, gerando assim o resultado esperado a ser exibido pelo atributo innerHTML do método getElementById().

A função é interessante de ser analisada, pois ela é responsável por realizar o processamento dos dados obtidos pelo preenchimento de valores nos campos de um pequeno formulário gerado em HTML através do uso da atribuição direta do dado contido em cada campo a uma variável n1, n2 ou n3.

Outro aspecto relevante e essencial para o funcionamento correto do script é a conversão dos dados dos campos que inicialmente se comportam como caracteres texto desde sua inserção nos campos até a atribuição às variáveis, e posterior conversão em valores numéricos inteiros através de outra função padrão da linguagem chamada parseInt.

Com isto, é importante observar que é possível utilizar funções prontas da própria linguagem JavaScript, assim como criar funções específicas para realizar processos necessários a partir de demandas específicas.

**TABELA 11 - EXEMPLO DE HTML COM JAVASCRIPT USANDO FUNÇÕES**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 13&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;Digite um valor para a primeira nota:&lt;/p&gt; &lt;input id="n1" value="0" /&gt; &lt;p&gt;Digite um valor para a segunda nota:&lt;/p&gt; &lt;input id="n2" value="0" /&gt; &lt;p&gt;Digite um valor para a terceira nota:&lt;/p&gt; &lt;input id="n3" value="0" /&gt; &lt;br&gt; &lt;button onclick="document.getElementById('ex13').innerHTML = 'Resultado = ' + calculaMedia()"&gt;Média&lt;/button&gt; &lt;p id="ex13"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; function calculaMedia() {   n1 = document.getElementById("n1").value;   n2 = document.getElementById("n2").value;   n3 = document.getElementById("n3").value;   return ((parseInt(n1) + parseInt(n2) + parseInt(n3)) / 3); } &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

Existem diferentes formas de se obter um mesmo resultado na programação, e uma alternativa ao que foi implementando no exemplo da tabela 10 é o script da tabela 11 onde ao invés da função realizar todo o processamento, inclusive da exibição do resultado do cálculo ao usuário, a função processa os dados e cálculos necessários apenas.

Ao final da função é utilizada a palavra reservada `return` para que o resultado do que foi processado pela função possa ser devolvido ao comando que ativou a função, aumentando a independência da função em relação ao restante do script, e tornando-a mais reutilizável em outros scripts por ser bastante previsível em sua funcionalidade.



#### 4. OBJETOS E SUAS PROPRIEDADES

O conceito de objeto é algo bastante comum na área de desenvolvimento de software nos dias atuais, e se baseia na forma como diferenciamos objetos do mundo real, avaliando as características mais relevantes que sejam necessárias para que se possa ter um nível suficiente de diferenciação entre cada objeto.

Este conceito vem de uma variante do paradigma imperativo de programação chamado de programação orientada a objetos onde é realizada uma chamada abstração em casos reais de forma a serem escolhidos e representados de forma digital, características relevantes que possam ser suficientes para diferenciar um objeto representado de outro.

Para isto, é necessário que se compreenda que a correta escolha das características e definição de um conjunto destas características é fundamental para que uma representação digital de identificação do objeto seja suficientemente completa para que todos os objetos existentes deste tipo possam ser diferenciados entre si e identificados o mais precisamente possível.

Partindo desta ideia base, podemos analisar um exemplo de forma a auxiliar a compreensão dos conceitos relacionados com a definição de objetos e demonstrar de forma sucinta, como poderia ser realizado o processo de seleção de características relevantes de um elemento real a ser convertido em um conjunto de dados.

Um dos exemplos mais comuns é o de abstração de meios de transporte, e o primeiro aspecto a ser levado em consideração, é o quanto se conhece do objeto real para que se possa abstrair características relevantes que possam se tornar atributos de um objeto representativo para os meios de transporte a serem utilizados.

Partindo de uma classe de meios de transporte mais conhecida, temos os carros que são um tipo bastante popular de meio de transporte e que facilmente pode ser analisado para que se possa abstrair um conjunto de dados adequado.

**FIGURA 1 - EXEMPLO DE CARRO**



**Fonte:** <https://pixabay.com/pt/photos/auto-carro-cadillac-velho-788747/>

Veículos possuem algumas características fundamentais que os diferenciam como marca e modelo, mas isso pode se repetir muito facilmente, pois é fácil termos num cadastro de uma loja, dois carros de mesma marca e modelo.

O exemplo da figura 1 traz um veículo já antigo e não mais fabricado regularmente, sendo mais facilmente diferenciado de outros veículos, mas mesmo assim, existem outras características que são necessárias para garantir uma diferenciação eficiente entre veículos, e ai surgem dados de identificação de veículos como chassis e placa que formam um conjunto único de identificação de veículos e nosso país.

Existem também muitas outras características que diferenciam veículos como cor, ano de fabricação, motorização e câmbio que são importantes na avaliação e diferenciação de veículos para venda, sendo geralmente relevantes para a venda de veículos, por exemplo.

Além de todas as características citadas até o momento, existem outras que podem ser importantes de acordo com especificidades dos softwares a serem implementados como acessórios contidos no veículo, estado de conservação de componentes, ocorrências que possam alterar aspectos legais do veículo, e itens específicos como material dos bancos.

**FIGURA 2 - EXEMPLO DE CARRO**



**Fonte:**<https://pixabay.com/pt/photos/auto-autom%c3%b3vel-automotivo-amg-2179220/>

Avaliando a figura 2, temos um veículo nitidamente muito mais moderno que o da figura 1, e é possível identificar outras características que podem diferenciar os dois modelos de veículos como dispositivos eletrônicos disponíveis em cada um, impostos e multas pendentes, ou quaisquer outros dados que possam ser mais comuns em um ou outro tipo de veículo.

Com base em todos estes aspectos, é possível observar que o que define quais são as características que devem ser utilizadas para definir os atributos de um objeto que represente algo real, são as demandas específicas de cada problema a ser solucionado.

Diferenciando situações reais, temos o exemplo de uma montadora de veículos que necessita de todos as características possíveis do veículo citadas até o momento para diferenciar os veículos, mas como estes ainda não foram vendidos, não possuem placa, sendo então necessária a diferenciação básica por chassis.

Além da identificação única dos chassis, para outras atividades como de logística e avaliação de recall (substituição ou reparo) de itens defeituosos, é importante que se tenha também dados sobre lote de produção e outras identificações específicas da linha de produção.

Já no caso de uma revenda, também existem diferenças entre revenda de veículos novos e veículos seminovos, pois a segunda categoria já possui emplacamento, maior depreciação do veículo e seus componentes, pode ter ocorrido algum sinistro que muda a avaliação do veículo, ou o mesmo pode até não ter mais condições de circulação.

Para uma empresa que adquire o veículo para uso, outras características podem ser necessárias como alguma numeração de identificação para a frota, assim como atualização da quilometragem rodada a cada utilização do mesmo, e para pessoas que adquirem o veículo apenas para uso quando necessário, detalhes como quilometragem para revisão e manutenções, datas de vencimento de parcelas e seguro, por exemplo.

Com base em tudo que foi citado, é fácil concluir que o processo de abstração de atributos não é tão simples quanto se pode imaginar, e quanto menos se conhece do objeto, mais complicado pode ser o processo, como no caso de barcos, trens ou aeronaves que possuem características desconhecidas de grande parte das pessoas e no caso de serem necessárias, é preciso realizar mais pesquisas e coleta de informações com quem conheça estes meios e possa auxiliar no processo.

A partir da definição dos objetos e seus atributos, é preciso também definir que tipos de dados são adequados para cada diferente atributo, pois alguns necessitam armazenar texto e outros valores numéricos, fora atributos mais específicos que podem receber imagens, sons, vídeos ou outros tipos de dados não primitivos.

**FIGURA 3 - ANIMAIS DIVERSOS**



**Fonte:** <https://pixabay.com/pt/photos/zebra-girafa-%c3%a1frica-namibia-1170177/>

Além dos atributos que identificam objetos, existem ações que podem ser realizadas para processar dados de atributos de forma a obtê-los, alterá-los ou excluí-los sendo igualmente importantes para o desenvolvimento de software, pois toda a manipulação de dados ocorre a partir destas ações que são implementadas em chamados métodos.

Os métodos contêm instruções especificamente implementadas para agir sobre atributos geralmente e funcionam como aquilo que pode ser realizado com os objetos reais que foram abstraídos de certa forma.

No caso do tipo de objeto veículo que vem sendo utilizado como exemplo, métodos que poderiam ser utilizados seriam para representar eventos que podem ocorrer com veículos como o cadastro de seus dados, consulta de dados cadastrados, a alteração destes dados quando necessário, e se necessário, sua exclusão, sendo este conjunto de eventos possíveis chamado de CRUD (Create, Read, Update and Delete ou Criar, Ler, Atualizar e Excluir).

Além do CRUD, existem outros métodos possíveis associados com atividades mais específicas de cada objeto como dirigir um veículo, vende-lo, realizar manutenção no mesmo, comunicar uma ocorrência ocorrida com o mesmo, ou até torna-lo sucata, fazendo com que deixe de ser considerado um veículo apto para uso.

A escolha dos métodos adequados a serem implementados para objetos em uma solução depende do que a solução necessita ser capaz de realizar e quais processos podem ser aplicados aos dados relativos aos atributos destes objetos, sendo importante destacar a importância de alinhamentos entre cliente que solicita a solução computacional e responsáveis pelo projeto a ser desenvolvido para alinhar quais seriam os objetos, atributos e métodos associados ao projeto e suas especificidades.

#### **4.1 Implementando objetos**

Aproveitando o que foi contextualizado sobre a programação com base na definição de objetos com seus atributos e métodos, é interessante exemplificar o uso desta forma de programação em scripts que permitam o uso deste tipo mais complexo de estrutura de dados.

**TABELA 12 - EXEMPLO DE HTML COM JAVASCRIPT USANDO OBJETOS E ATRIBUTOS**

1	<pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 14&lt;/h2&gt; &lt;hr&gt;  &lt;p&gt;Digite a marca do veículo:&lt;/p&gt; &lt;input id="n1" value="0" /&gt; &lt;p&gt;Digite o modelo do veículo:&lt;/p&gt; &lt;input id="n2" value="0" /&gt; &lt;p&gt;Digite a placa do veículo:&lt;/p&gt; &lt;input id="n3" value="0" /&gt; &lt;br&gt; &lt;button onclick="funcao()"&gt;Consulta Dados&lt;/button&gt; &lt;p id="ex141"&gt;&lt;/p&gt; &lt;p id="ex142"&gt;&lt;/p&gt; &lt;p id="ex143"&gt;&lt;/p&gt; </pre>
2	<pre> &lt;script&gt; function funcao() {   n1 = document.getElementById("n1").value;   n2 = document.getElementById("n2").value;   n3 = document.getElementById("n3").value;   const carro = {marca: n1, modelo: n2, placa: n3};   document.getElementById('ex141').innerHTML = 'Marca: ' + carro.marca;   document.getElementById('ex142').innerHTML = 'Modelo: ' + carro.modelo;   document.getElementById('ex143').innerHTML = 'Placa: ' + carro.placa; } &lt;/script&gt; </pre>
3	<pre> &lt;/body&gt; &lt;/html&gt; </pre>

Fonte: O autor (2022).

Observando o exemplo da tabela 12, temos uma situação típica da implementação de um script, onde após um clique no botão que aciona a função, é instanciado um objeto chamado carro que possui os atributos marca, modelo e placa que recebem os dados inseridos nos campos do formulário preenchido pelo usuário.

Segundo Silva (2010), é importante observar que o uso de objetos possui uma sintaxe própria que deve ser respeitada, indicando sempre o nome do objeto e depois o atributo desejado separados por um ponto como foi feito em carro.marca para especificar que seria atribuído um dado a este atributo e para que o dado pudesse ser exibido com o uso do atributo innerHTML.

A aplicabilidade dos objetos no desenvolvimento em JavaScript é bastante interessante, pois permite que informações que poderiam ser espalhadas em diversas variáveis desconectadas sejam agrupadas sob uma mesma estrutura que as torne pertencentes a um mesmo elemento chamado objeto, facilitando a interpretação e implementação de scripts.

Além da definição de atributos e seu uso, temos a possibilidade de definir métodos em objetos que permitem a realização do processamento dos dados contidos como atributos do próprio objeto, assim como dados de fora do objeto.

**TABELA 13 - EXEMPLO DE HTML COM JAVASCRIPT USANDO OBJETOS, ATRIBUTOS E MÉTODOS**

	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 14&lt;/h2&gt; &lt;hr&gt;  &lt;p&gt;Digite a marca do veículo:&lt;/p&gt; &lt;input id="n1" value="" /&gt; &lt;p&gt;Digite o modelo do veículo:&lt;/p&gt; &lt;input id="n2" value="" /&gt; 1 &lt;p&gt;Digite a placa do veículo:&lt;/p&gt; &lt;input id="n3" value="" /&gt; &lt;p&gt;Digite o ano de fabricação do veículo:&lt;/p&gt; &lt;input id="n4" value="0" /&gt; &lt;br&gt; &lt;button onclick="funcao()"&gt;Consulta Dados&lt;/button&gt; &lt;p id="ex141"&gt;&lt;/p&gt; &lt;p id="ex142"&gt;&lt;/p&gt; &lt;p id="ex143"&gt;&lt;/p&gt; &lt;p id="ex144"&gt;&lt;/p&gt; &lt;p id="ex145"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; function funcao() {   n1 = document.getElementById("n1").value;   n2 = document.getElementById("n2").value;   n3 = document.getElementById("n3").value;   n4 = document.getElementById("n4").value;   n5 = new Date().getFullYear();   const carro = {     marca: n1,</pre>

	<pre>         modelo: n2,         placa: n3,         anoFabricacao: n4,         ipva : function() {             if (n5 - this.anoFabricacao &gt;= 20)                 return this.anoFabricacao + " não paga IPVA.";             else                 return this.anoFabricacao + " paga IPVA.";         }     };     document.getElementById('ex141').innerHTML = 'Marca: ' + carro.marca;     document.getElementById('ex142').innerHTML = 'Modelo: ' + carro.modelo;     document.getElementById('ex143').innerHTML = 'Placa: ' + carro.placa;     document.getElementById('ex144').innerHTML = 'Ano: ' + carro.anoFabricacao;     document.getElementById('ex145').innerHTML = carro.ipva(); } &lt;/script&gt; </pre>
3	<pre> &lt;/body&gt; &lt;/html&gt; </pre>

Fonte: O autor (2022).

O exemplo trazido na tabela 13 contém elementos que foram agregados em relação ao exemplo da tabela 12, em que um método é adicionado ao objeto carro como aspecto mais importante, mas outros elementos podem ser analisados pela sua funcionalidade em conjunto com o método.

Primeiramente, observa-se que foi adicionado um campo adicional no formulário de entrada de dados para a indicação do ano de fabricação de um veículo e a adição de dois parágrafos extras para exibição do ano de fabricação e uma mensagem adicional que será gerada pelo método implementado no objeto.

Outro aspecto importante é a obtenção de um dado importante no contexto do problema que é o ano atual que servirá de base para o método, sendo atribuído à variável n5 pelo uso do método `getFullYear()` que pertence ao objeto padrão `Date()` da própria linguagem JavaScript.

Tendo disponível o ano atual armazenado na variável n5, o método associado ao atributo ipva realiza um processo de avaliação na quantidade de anos que o veículo já existe e a partir deste dado, pode avaliar se o mesmo, tendo ultrapassado 20 anos, pode estar isento do pagamento de IPVA (10 anos para motos).

O processo de análise faz parte de um conteúdo que será tratado mais adiante nesta unidade, e por isso será ignorado por enquanto, mas é extremamente importante para o processamento realizado no método, pois a mensagem adequada a ser exibida ao usuário pela execução do método depende desta avaliação automática realizada durante sua execução.

## SAIBA MAIS

Um dos importantes módulos para implementação e scripts JavaScript é chamado math e serve para que processamento de dados numéricos seja menos trabalhoso com o uso de métodos e constantes como PI.

Existem métodos para cálculos não tão convencionais como sqrt(n) para raiz quadrada e LN10 para calcular o chamado logaritmo natural com base 10, por exemplo, existem métodos que ajustam valores numéricos como round(n), ceil(n), floor(n) e trunc(n) que realizam o processo a partir de diferentes regras como arredondamento para cima, para baixo, ou até a simples exclusão das casas decimais.

Além destes métodos matemáticos citados, existem outros para cálculos trigonométricos como sin(n) e cos(n), min(n, m, ..., k) e max(n, m, ..., k) para menor ou maior valor de uma lista, pow(n,m) para exponenciação e por fim random() que gera um número aleatório entre 0 e 1 que pode depois ser multiplicado para que se obtenham outros valores.

**Fonte:** Adaptado de: JavaScript Math Object.

## REFLITA

Conhecer várias linguagens de programação não demonstram que a pessoa seja um ótimo programador, mas sim sua capacidade de compreender e criar algoritmos para resolver problemas.

**Fonte:** O autor (2022).

# CONSIDERAÇÕES FINAIS

Com o encerramento dos estudos nesta unidade, completou-se um ciclo de aprendizado básico da programação em linguagem JavaScript, lembrando que existem outros recursos disponíveis que podem valer um esforço extra de pesquisa e implementação de exemplos para que seja possível um contínuo aperfeiçoamento na linguagem JavaScript e seus recursos.

Houve a continuidade dos estudos de operadores complementares aos tradicionais matemáticos vistos na unidade anterior e a correta utilização de toda essa variedade de símbolos na escrita de scripts.

Foram vistos recursos essenciais de programação como estruturas variadas de dados simples como variáveis e mais complexas como objetos e vetores que podem ser amplamente utilizados na implementação de scripts.

Os estudos seguem ao longo do material que traz novos recursos que podem ser muito úteis no processo de desenvolvimento de conteúdo interativo e dinâmico para a Internet.

## LEITURA COMPLEMENTAR

A linguagem JavaScript oferece um mecanismo bastante funcional para a manipulação de dados do tipo data, utilizando um objeto chamado Date() que contém diversos métodos para a realização deste trabalho.

O formato padrão para os dados neste objeto são no formato Date (ano, mês, dia, horas, minutos, segundos, milissegundos) e o processo para se instanciar um objeto do tipo data é através de uma constante declarada como objeto deste tipo.

**CONST DATA = NEW DATE();**

ou

**CONST DATA = NEW DATE(2022, 08, 15, 21, 00, 00, 0);**

É possível indicar dados para instanciar a data parcialmente se desejado, bastando informar nos parênteses apenas ano, mês e dia, por exemplo, sendo que os demais dados que compõe a estrutura da data serão considerados zero por serem irrelevantes no momento da declaração da constante.

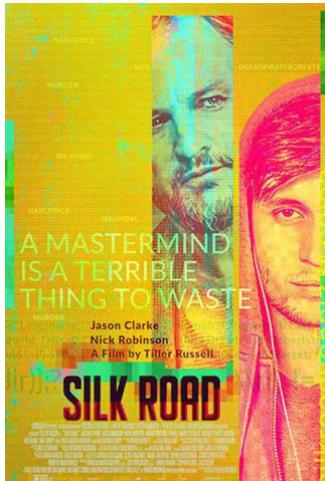
Utilizando o atributo innerHTML do método getElementById() é possível exibir a data atual armazenada na constante data de maneira atualizada, de acordo com dados atribuídos e atualizados pelo relógio interno do equipamento que executa o script.

Utilizando o método toDateString() com o objeto de data a ser exibido, o formato da data é alterado para dia da semana, dia, mês e ano de acordo com o padrão do país para exibição de datas.

Existem outros métodos disponibilizados pela linguagem JavaScript para ampliar as possibilidades de uso de datas como dados simples ou elementos para a realização de outros tipos de processamento, por exemplo.

**Fonte:** Adaptado de: JavaScript Date Objects.

## MATERIAL COMPLEMENTAR



### LIVRO

**Título:** Lógica de Programação e Algoritmos com JavaScript: uma Introdução à Programação de Computadores com Exemplos e Exercícios Para Iniciantes

**Autor:** Edécio Fernando Iepsen

**Editora:** Novatec

**Sinopse:** Livro que utiliza a linguagem JavaScript para demonstrar conceitos fundamentais de programação como entrada, processamento e saída de dados, estruturas de controle de fluxo e manipulação de estruturas de dados.



### FILME/VÍDEO

**Título:** SILK ROAD: MERCADO CLANDESTINO

**Ano:** 2021.

**Sinopse:** Pensando em uma forma alternativa de ganhar dinheiro, o protagonista utiliza um site para realizar a venda de narcóticos tentando não ser rastreado.

# UNIDADE III

# Estruturas de Dados e Estruturas de Controle Javascript

Professor Especialista Ronie Cesar Tokumoto



## Plano de Estudo:

- Conhecendo Vetores;
- Utilizando Estruturas de Dados;
- Estruturas de Controle;
- Apresentando JSON.

## Objetivos da Aprendizagem:

- Compreender os tipos de estruturas de dados simples e complexas;
- Estabelecer a importância de conhecer estruturas de controle de fluxo;
  - Conhecer o padrão JSON para desenvolvimento de scripts.

# INTRODUÇÃO

A partir dos conhecimentos adquiridos em unidades anteriores, é possível então avançar para conhecer a parte mais relacionada ao raciocínio lógico da programação em JavaScript que traz estruturas de controle que permitem oferecer meios de um script tomar decisões por si só automaticamente com base em instruções implementadas para oferecer mecanismos de avaliação de condições.

Primeiramente, devem ser estudadas formas complementares e mais complexas de armazenamento de dados como vetores que são capazes de armazenar maiores quantidades de dados sob um mesmo nome, diferentemente das variáveis que armazenam apenas um dado por vez.

As estruturas de controle são outro aspecto muito importante tratado na unidade que permite que os scripts evoluam de conteúdo interativo e com efeitos para conteúdos dinâmicos e capazes de realizar processos e tomadas de decisão em tempo real.

Esta unidade completa o ciclo de apresentação da linguagem JavaScript ao estudante, oferecendo conceitos e exemplos comentados sobre muitos dos principais fundamentos da linguagem JavaScript.



## 1. CONHECENDO VETORES

Chamamos de estruturas de dados todo tipo de meio de armazenamento de dados durante a execução de um script, sendo a variável, o tipo mais comum de estrutura de dados que permite que apenas um dado seja armazenado nesta, sendo possível a alteração do dado a qualquer momento, diferentemente do que ocorre em constantes que recebem dados como variáveis, mas permanecem inalteradas por toda a execução de um script.

Para diferenciar a declaração de uma variável ou constante, deve-se utilizar a palavra reservada `const` para constantes, e utiliza-se as palavras reservadas `var` ou `let` para variáveis, sendo `var` de uso mais global e `let` mais restrito a instruções ou blocos, além de que variáveis não obrigatoriamente precisam ter dados atribuídos em sua declaração, mas constantes precisam. Pode-se declarar também variáveis sem palavras reservadas antes dos nomes.

**TABELA 1 - EXEMPLO DE VARIÁVEIS E CONSTANTES EM JAVASCRIPT**

1	var idade; rg; let peso = 70; altura = 1,75;
2	const pi = 3,14; const pessoa = { nome: "João", sobrenome: "Silva" };

**Fonte:** O autor (2022).

Pelos exemplos contidos na tabela 1, podemos observar diferentes formas de declaração de variáveis, utilizando ou não palavras reservadas e inicializando-as com valores ou não, assim como exemplos de constante simples como pi e de um objeto mais complexo como pessoa que possui atributos, sendo que em ambos os exemplos, é preciso inicializar a constante pi e os atributos da constante objeto com dados.

Existem recursos para o processamento de texto (string) que são muito úteis, pois geralmente os dados utilizados na web são tratados como texto e assim, quanto mais métodos estiverem disponíveis para processar dados deste tipo, melhor para os desenvolvedores que necessitam criar menos scripts para realizar estas ações.

A tabela 2 traz alguns dos métodos padrões que são disponibilizados pela linguagem JavaScript para que se possa realizar ações como de contar caracteres de um texto, obter partes do texto, converter seus caracteres para maiúsculas ou minúsculas se forem letras do alfabeto, realizar buscas de conteúdos em textos, etc.

**TABELA 2 - MÉTODOS UTILIZÁVEIS COM TEXTO EM JAVASCRIPIT**

<b>toString(n)</b>	converte dado numérico em texto
<b>length ("X")</b>	retorna a quantidade de caracteres de um texto
<b>substr(n,m)</b>	retorna m caracteres após o n caractere de um texto
<b>replace(n,m)</b>	substitui um conjunto de caracteres por outro em uma variável /"X"/i – a diferença entre maiúsculas e minúsculas é ignorada /"X"/g – todas as ocorrências no texto serão substituídas
<b>toUpperCase()</b>	converte todos os caracteres de um texto em maiúsculas
<b>toLowerCase()</b>	converte todos os caracteres de um texto em minúsculas
<b>concat("X")</b>	une dois textos em um apenas
<b>trim()</b>	Elimina espaços em branco no início e final de um texto
<b>padStart(n,"X")</b>	complementa texto com n caracteres X antes do texto
<b>padEnd(n,"X")</b>	complementa texto com n caracteres X depois do texto
<b>charAt(0)</b>	retorna um determinado caractere de um texto iniciando em 0
<b>split(",")</b>	transforma partes de um texto em vetor de textos
<b>indexOf("X")</b>	retorna a posição onde um conjunto de caracteres se encontra em um texto
<b>search("X")</b>	realiza uma busca por um conjunto de caracteres em um texto

Fonte: O autor (2022).

Os métodos contidos na tabela não representam a totalidade de métodos disponíveis na linguagem JS, mas contém opções bastante úteis e comuns que podem ser utilizados na implementação de scripts que manipulam cadeias de caracteres e obtém novos dados a partir destes textos.

**TABELA 3 - EXEMPLO DE MANIPULAÇÃO DE TEXTO EM JAVASCRIPT**

1	<pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 15&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;Digite seu nome completo:&lt;/p&gt; &lt;input id="n1" value="XXXXXXXXXX" /&gt; &lt;p&gt;Digite sua data de nascimento no formato (dd/mm/aaaa):&lt;/p&gt; &lt;input id="n2" value="99/99/9999" /&gt; &lt;p&gt;Digite a cidade onde nasceu:&lt;/p&gt; &lt;input id="n3" value="XXXXXXXXXX" /&gt; &lt;/br&gt; &lt;button onclick="funcao()"&gt;Gera Dados&lt;/button&gt; &lt;p id="ex151"&gt;&lt;/p&gt; &lt;p id="ex152"&gt;&lt;/p&gt; &lt;p id="ex153"&gt;&lt;/p&gt; &lt;p id="ex154"&gt;&lt;/p&gt; </pre>
2	<pre> &lt;script&gt; function funcao() {   n1 = document.getElementById("n1").value;   document.getElementById('ex151').innerHTML = 'Quantidade de caracteres: ' + n1.length;   n2 = document.getElementById("n2").value;   document.getElementById('ex152').innerHTML = 'Ano de nascimento: ' + n2.substr(6,4); // poderia ser usado (-4) com o mesmo resultado   n3 = document.getElementById("n3").value;   document.getElementById('ex153').innerHTML = 'Nova cidade onde irá trabalhar: ' + n3.replace (n3, "Paranavaí");   document.getElementById('ex154').innerHTML = 'Nome ajustado: ' + n1.toUpperCase (n1); // toLowerCase() mudaria todas as letras para minúsculas } </pre>
3	<pre> &lt;/script&gt; &lt;/body&gt; &lt;/html&gt; </pre>

**Fonte:** O autor (2022).

O exemplo da tabela 3 traz a aplicação de alguns métodos para manipulação de texto para exemplificar seu uso e possibilidades práticas de aplicação destes métodos em situações comuns do cotidiano como a contagem de caracteres de um texto, a utilização de apenas uma parte do conteúdo de uma string, ou substituir parte específica do conteúdo de uma string por outro, e por fim, a conversão de todas as letras de uma string em maiúsculas se necessário.

O método `length()` retorna a quantidade de caracteres contidos na variável `n1` no exemplo, mas poderia ser de um texto específico indicado entre aspas, sendo uma escolha possível nos métodos que lidam com texto, pois podem ser utilizados dados de campos de um formulário armazenados em estruturas de dados ou textos fixos que devem ser processados pelos métodos a partir de demandas específicas como de verificação de senhas, por exemplo.

Outro método utilizado no exemplo, `substr()` possui duas formas de utilização, sendo a do exemplo mais detalhada, pois são indicados como parâmetros os valores 6 e 4 que indicam que a partir do início da string, deve-se mover 6 caracteres adiante, da esquerda para a direita, e depois, os próximos 4 caracteres devem ser devolvidos como retorno da função, mas poderia ser indicado apenas o parâmetro -4 que também traria os 4 últimos dígitos da string, mas isso só coincidiu pelo valor 6 andar exatamente a mesma quantidade de caracteres que é desconsiderada pelo parâmetro -4.

O método `replace()` simplesmente substitui o conteúdo da string contida na variável `n3` pelo texto Paranavaí, independentemente da quantidade de caracteres que o usuário havia digitado no formulário e que haviam sido armazenados na variável `n3`.

Por fim, o método `toUpperCase()` busca na string contida na variável `n1`, letras do alfabeto que sejam minúsculas e vai substituindo, uma a uma, pelas equivalentes maiúsculas, sendo que o método `toLowerCase()` poderia ser utilizado para realizar o processo oposto.

Existem também alguns métodos bastante simples para uso com valores numéricos como `toFixed(n)` que fixa a quantidade de casas decimais de números para o valor indicado pelo parâmetro `n`, e assim, `toFixed(2)` faria com que um valor como 1,23678 fosse convertido para 1,24 arredondando o valor de acordo com a regra padrão matemática onde casas decimais maiores ou iguais a 0,5 aumentam 1 unidade na casa anterior.

**TABELA 4 - EXEMPLO DE MANIPULAÇÃO DE NÚMEROS EM JAVASCRIPT**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 16&lt;/h2&gt; &lt;hr&gt; Experimento de soma de valores com conversão numérica. &lt;br&gt; Serão somados os valores: &lt;p id="ex161"&gt;&lt;/p&gt; Resultado: &lt;p id="ex162"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; valor = Number(true) + Number(false) + Number("5") + Number(" 5"); document.getElementById('ex161').innerHTML      =      "Number(true)      + Number(false) + Number('5') + Number(' 5')"; document.getElementById('ex162').innerHTML = valor; &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

No exemplo da tabela 4, temos um exemplo de uso de um método para tratamento de dados numéricos que procura dados que possam ser convertidos em numéricos, ignorando caracteres não numéricos ou situações que não possam ser convertidas.

Segundo Silva (2010), valores lógicos true (verdadeiro) e false (falso) são substituídos pelos valores 1 e 0 respectivamente, sendo então compreendidos pelo JavaScript como dados que possuem valores numéricos equivalentes, como ocorre em muitas linguagens de programação.

Outros métodos importantes para conversão numérica são parseInt() e parseFloat() que convertem, se possível, dados considerados texto em números inteiros ou com casas decimais de acordo com o uso de um dos dois métodos citados respectivamente.



## 2. UTILIZANDO ESTRUTURAS DE DADOS

Estruturas de dados como variáveis e constantes são importantes para o desenvolvimento de soluções computacionais para à Internet através de scripts JavaScript combinados com páginas web estruturadas em HTML.

Segundo Silva (2010), além destes tipos de estruturas, existe outro tipo relevante de estrutura de dados chamada de vetor que permite que uma maior quantidade de dados seja adicionado a uma única estrutura de dados, sendo este um tipo bastante útil de estrutura, pois é melhor utilizar um vetor com dez elementos que dez variáveis diferentes contendo apenas um dado cada.

Sua utilização é baseada na inicialização da estrutura com os dados em forma de lista que depois podem ser acessados através de índices indicados entre colchetes, sendo o primeiro elemento representado pelo índice 0.

TABELA 5 - EXEMPLO DE MANIPULAÇÃO DE NÚMEROS EM JAVASCRIPT

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 17&lt;/h2&gt; &lt;hr&gt;  &lt;b&gt;Vogais:&lt;/b&gt; &lt;p id="ex171"&gt;&lt;/p&gt; &lt;b&gt;Números:&lt;/b&gt; &lt;p id="ex172"&gt;&lt;/p&gt; &lt;b&gt;Segundo elemento da lista de números:&lt;/b&gt; &lt;p id="ex173"&gt;&lt;/p&gt; &lt;b&gt;Pessoas:&lt;/b&gt; &lt;p id="ex174"&gt;&lt;/p&gt; &lt;b&gt;Estados:&lt;/b&gt; &lt;p id="ex175"&gt;&lt;/p&gt; &lt;b&gt;Países:&lt;/b&gt; &lt;p id="ex176"&gt;&lt;/p&gt; &lt;b&gt;Adiciona país:&lt;/b&gt; &lt;p id="ex177"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; const vogais= [   "A",   "E",   "I",   "O",   "U" ];vogais; document.getElementById("ex171").innerHTML = vogais;  const numeros = [];</pre>

	<pre> numeros[0]= "1"; numeros[1]= "2"; numeros[2]= "3"; document.getElementById("ex172").innerHTML = numeros; document.getElementById("ex173").innerHTML = numeros[1];  const pessoas = ["João", "Maria", "Ana", "Pedro"]; document.getElementById("ex174").innerHTML = pessoas.length;  const estados = ["PR", "SC", "RS"]; document.getElementById("ex175").innerHTML = "&lt;li&gt; " + estados[0] + "&lt;li&gt; " + estados[1] + "&lt;li&gt; " + estados[2] + "&lt;/script&gt;";  const paises = ["Brasil", "Argentina", "Uruguai"]; document.getElementById("ex176").innerHTML = paises; paises.push("Paraguai"); document.getElementById("ex177").innerHTML = paises; </pre>
3	<pre> &lt;/body&gt; &lt;/html&gt; </pre>

Fonte: O autor (2022).

O script de exemplo da tabela 5 traz alguns exemplos de uso de vetores criados para demonstrar seu uso em diferentes situações, e na primeira e segunda opções são trazidas duas diferentes formas de se declarar vetores, onde a primeira declara uma constante do tipo vetor vogais e já insere os dados relacionados, e o segundo exemplo declara a constante vetor números vazios e em seguida atribui dados para cada posição desejada do vetor.

Também é exemplificado como exibir os dados contidos no vetor através do atributo innerHTML indicando apenas o nome do vetor para que todos os seus elementos sejam exibidos separados automaticamente por vírgulas, ou um elemento específico seja exibido através da indicação do índice [1] associado a ele no vetor, representando o segundo elemento.

Depois, um terceiro vetor pessoas é declarado e inicializado com dados para que depois seja exibida a quantidade de elementos através do método length(), e também, em outro exemplo na sequência do script é utilizado o método push() para adicionar um novo elemento ao final do vetor, pois a estrutura funciona como uma pilha, onde os novos elementos são adicionados sempre ao final da estrutura.

Existe um outro trecho do script que mostra como se pode adicionar tags HTML dentro de scripts através do uso do atributo innerHTML que permite que além de texto e estruturas de dados, sejam utilizadas tags HTML para formatar como tópicos a exibição dos elementos do vetor estados.

Compreender o correto uso de estruturas de dados e métodos que podem ser aplicados sobre eles é algo essencial para que se possa avançar nos estudos e desenvolver bons scripts em JavaScript, assim como em quaisquer linguagens de programação, e por isto, mantenha presente em sua rotina de estudos o seu aperfeiçoamento em como utilizar variadas estruturas de dados.



### 3. ESTRUTURAS DE CONTROLE

Após ter conhecido as bases de como estruturar os dados para uma aplicação com base em scripts JS, é hora de conhecer como se pode automatizar parte do processamento que pode ser realizado em um script através do que se chama de estruturas de controle de fluxo, ou seja, meios para que o script sendo executado tenha autonomia para decidir certas ações a serem realizadas, ou não.

As estruturas de controle se dividem em estruturas de decisão e estruturas de repetição que permitem que scripts se tornem muito mais eficientes e complexos, aumentando as variações de processamento que podem ser realizados, mas também a dificuldade de implementação de scripts, pois para uso destes tipos de estruturas, é preciso mais esforço lógico para a elaboração dos algoritmos que servem de base para a escrita de scripts.

Para poder utilizar estas estruturas de controle é preciso destacar que o uso de operadores para a elaboração de expressões lógicas é bastante comum, pois servem para avaliar situações que ocorrem durante a execução de um script e que necessitam de autonomia para a tomada de decisões visando a maior automatização de processos.

Inicialmente, trataremos das estruturas de decisão que são fundamentais para a automatização de processos, pois a partir de expressões lógicas que podem resultar em resultados verdadeiros ou falsos, podemos decidir os rumos da execução de scripts.

Existem estruturas de decisão simples, compostas e múltiplas, sendo que todos estes tipos de estruturas podem ser encadeadas, ou seja, utilizadas em conjunto para ampliar a possibilidade de escolhas a serem oferecidas como alternativas de execução.

As palavras reservadas if, else e switch são as bases para a elaboração de instruções que utilizando comandos baseados nestas palavras reservadas, são capazes de avaliar dados e decidir entre processar ou não outras instruções.

**TABELA 6 - EXEMPLO DE ESTRUTURA DE DECISÃO EM JAVASCRIPT**

1	<!DOCTYPE html> <html> <body> <h2>JavaScript - Exemplo 18</h2> <hr> <p>Digite sua idade: </p> <input id="idade" value="0" />   <button onclick="funcao()">Verifica Idade</button> <p id="ex18"></p>
2	<script> function funcao() { idade = document.getElementById("idade").value; if (idade >= 18){ document.getElementById('ex18').innerHTML = "Maior de idade"; } } </script>
3	</body> </html>

Fonte: O autor (2022).

O exemplo da tabela 6 mostra que o uso de uma estrutura de decisão permite que a exibição da mensagem de maioridade ao usuário esteja condicionada a análise do valor inserido no campo do formulário da página antes do clique no botão que ativa a função.

A análise verifica se o usuário seria considerado maior de idade em caso de sua idade ser maior ou igual ao valor base de 18 anos a partir de uma estrutura de decisão simples que compara o valor atribuído à variável idade a partir do conteúdo do campo preenchido com o valor 18, decidindo a partir do resultado da expressão se deve ou não mostrar o texto ao usuário que informa ser maior de idade.

**TABELA 7 - EXEMPLO DE ESTRUTURA DE DECISÃO EM JAVASCRIPT**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 19&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;Digite sua idade: &lt;/p&gt; &lt;input id="idade" value="0" /&gt; &lt;br&gt; &lt;button onclick="funcao()"&gt;Verifica Idade&lt;/button&gt; &lt;p id="ex19"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; function funcao() {     idade = document.getElementById("idade").value;     if (idade &gt;= 18){         document.getElementById('ex19').innerHTML = "Maior de idade";     } else {         document.getElementById('ex19').innerHTML = "Menor de idade";     } } &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

A pequena modificação realizada no exemplo da tabela 7 em relação ao exemplo da tabela 6 amplia a avaliação realizada na estrutura de decisão simples para composta, pois com a adição da palavra reservada else, existe a opção de que seja também informado ao usuário uma mensagem que informa ao mesmo ser menor de idade, tornando a instrução mais completa.

**TABELA 8 - EXEMPLO DE ESTRUTURA DE DECISÃO EM JAVASCRIPT**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 20&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;Digite sua idade: &lt;/p&gt; &lt;input id="idade" value="0" /&gt; &lt;br&gt; &lt;button onclick="funcao()"&gt;Verifica Idade&lt;/button&gt; &lt;p id="ex20"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; function funcao() {     idade = document.getElementById("idade").value;     if (idade &gt;= 18) {         mensagem = "Maior de idade e pode votar";     } else if (idade &gt;= 16) {         mensagem = "Menor de idade, mas pode votar";     } else {         mensagem = "Menor de idade, e não pode votar";     }     document.getElementById('ex20').innerHTML = mensagem; } &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

Para tornar o script ainda mais completo, utilizando a técnica de encadear ou aninhar instruções de decisão, podemos aumentar a quantidade de alternativas que podem ser abrangidas pela estrutura de decisão como um todo, e assim, com a inserção de uma instrução de decisão sempre dentro do comando else de uma instrução anterior, podemos realizar este encadeamento.

A primeira etapa da estrutura de decisão encadeada contida no exemplo da tabela 8 trata a condição `idade>=18` para avaliar se a pessoa é maior e pode votar, mas a possibilidade de votar entra numa faixa ampliada de idade e pessoas com 16 ou 17 anos também podem votar, mesmo sendo menores de idade.

Assim, encadeamos uma condição adicional para tratar desta particularidade adicionando uma nova estrutura de decisão composta que avalia se `idade>=16` para informar que se a pessoa estiver nesta faixa de idade de 16 a 17 anos é menor, mas pode votar, e abaixo dela, é menor e ainda não pode votar.

**TABELA 9 - EXEMPLO DE ESTRUTURA DE DECISÃO EM JAVASCRIPT**

1	<pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 21&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;&lt;b&gt;** MENU DE OPÇÕES **&lt;/b&gt;&lt;/p&gt; &lt;p&gt;1. Inserir Dados&lt;/p&gt; &lt;p&gt;2. Consultar Dados&lt;/p&gt; &lt;p&gt;3. Editar Dados&lt;/p&gt; &lt;p&gt;4. Excluir Dados&lt;/p&gt; &lt;p&gt;&lt;b&gt;Informe uma Opção:&lt;/b&gt;&lt;/p&gt; &lt;input id="opcao" value="0" /&gt; &lt;br&gt; &lt;button onclick="funcao()"&gt;Opção&lt;/button&gt; &lt;p id="ex21"&gt;&lt;/p&gt; </pre>
2	<pre> &lt;script&gt; function funcao() {   opcao = document.getElementById("opcao").value;   switch (opcao) {     case "1":       mensagem = "Escolhida a Inserção de Dados.";       break;     case "2":       mensagem = "Escolhida a Consulta de Dados.";       break;     case "3":       mensagem = "Escolhida a Edição de Dados.";       break;     case "4":       mensagem = "Escolhida a Exclusão de Dados.";       break;     default:       mensagem = "Opção Inválida.";   }   document.getElementById('ex21').innerHTML = mensagem; } &lt;/script&gt; </pre>
3	<pre> &lt;/body&gt; &lt;/html&gt; </pre>

**Fonte:** O autor (2022).

Mais especificamente, quando é preciso oferecer várias alternativas de escolhas a partir de um dado contido em uma variável, por exemplo, a complexidade para se utilizar estruturas baseadas no comando if se torna bastante complexa e as chances de erros ocorrerem em função de equívocos de lógica ou sintaxe são muito maiores.

Para casos como o do exemplo da tabela 9, uma instrução baseada na palavra reservada switch é uma opção interessante, pois sua sintaxe é bastante padronizada, reduzindo a dificuldade lógica e sua sintaxe é intuitiva, também contribuindo com uma menor complexidade de implementação.

No caso do exemplo citado, é criado um menu simulando opções de manipulação de dados padrão CRUD onde cada alternativa possível levaria a realização de processos diferentes associados a um conjunto de dados.

O comando definido pela palavra reservada switch avalia o valor contido na variável opção que recebeu o dado informado pelo usuário no campo do formulário da página e de acordo com o valor informado, opta entre mostrar uma das quatro alternativas de texto informativo de processos que poderiam ser realizados como inserção ou consulta de dados.

As opções válidas devem ser indicadas em linhas da instrução switch que se baseiam na palavra reservada case que deve ser associada a cada uma das alternativas válidas possíveis, e no caso do usuário informar um dado que não corresponda a nenhuma das alternativas válidas, é utilizada uma parte adicional da estrutura switch com base na palavra reservada default que realiza algum processo para quaisquer dados diferentes dos desejáveis.

Após conhecer os princípios das estruturas de decisão, é hora de avançar mais um pouco nos estudos e passar a estudar as estruturas de repetição que também oferecem importantes recursos para controle de fluxo na execução de scripts.

As estruturas de repetição permitem que instruções inteiras sejam repetidas por uma quantidade fixa ou variável de vezes a partir de um número de iterações (repetições) predeterminadas no comando da instrução, ou que a quantidade de repetições esteja condicionada a uma condição lógica que tem a responsabilidade de automaticamente decidir se as repetições devem continuar ocorrendo.

**TABELA 10 - EXEMPLO DE ESTRUTURA DE REPETIÇÃO EM JAVASCRIPT**

1	<pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 22&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;&lt;b&gt;Informe um Valor:&lt;/b&gt;&lt;/p&gt; &lt;input id="valor" value="7" /&gt; &lt;button onclick="funcao()"&gt;Tabuada&lt;/button&gt; &lt;p id="ex221"&gt;&lt;/p&gt; &lt;p id="ex222"&gt;&lt;/p&gt; </pre>
2	<pre> &lt;script&gt; function funcao() {   let tabuada = "";   valor = document.getElementById("valor").value;   document.getElementById('ex221').innerHTML = "Tabuada do " + valor;   for (let i = 1; i &lt;= 10; i++) {     tabuada += valor + " x " + i + " = " + valor * i + "&lt;br&gt;";   }   document.getElementById('ex222').innerHTML = tabuada; } &lt;/script&gt; </pre>
3	<pre> &lt;/body&gt; &lt;/html&gt; </pre>

**Fonte:** O autor (2022).

No exemplo da tabela 10 é utilizada uma instrução com base na palavra reservada for que necessita de três parâmetros para que possa ser executada, sendo a primeira um valor inicial para contagem atribuído a uma variável de controle, geralmente nomeada apenas por i de iteração.

Como segundo parâmetro é definida uma condição de parada das iterações, onde a escolha da regra define tanto quantas vezes devem ocorrer as repetições, quanto se devem ocorrer ou não, e até se as repetições devem ocorrer infinitamente.

No exemplo é atribuído o valor 1 para a variável i como valor inicial para contagem de repetições, e em seguida, a condição de parada é definida como i<=10, ou seja, enquanto as repetições não gerarem um resultado falso para esta condição, continuarão ocorrendo.

Para que seja possível o valor da variável de controle i avançar de forma a se aproximar da condição de parada, existe o terceiro parâmetro que contém a regra de modificação do valor da variável i de controle, indicando que a cada repetição, o valor da variável é incrementado em uma unidade com o uso de i++.

A construção da mensagem vai sendo realizada a medida que as iterações vão ocorrendo, e assim, a cada iteração, vai sendo incluída uma linha da tabuada concatenada a uma tag <br> para pular linha ao final, permitindo que cada linha da tabuada fique em uma linha diferente.

A medida que cada iteração ocorre, o valor da variável i inicialmente atribuída com 1 vai sendo incrementada em uma unidade até que atinja o valor 10 e realize a última iteração, pois na próxima verificação realizada pelo comando for, o valor de i será 11 e a estrutura de repetição será finalizada, seguindo com a execução normal do script.

**TABELA 11 - EXEMPLO DE ESTRUTURA DE REPETIÇÃO EM JAVASCRIPT**

	<!DOCTYPE html> <html> <body> <h2>JavaScript - Exemplo 23</h2>
1	<hr> <p><b>Informe o valor para busca na listagem:</b></p> <input id="valor" value="0" /> <button onclick="funcao()">Buscar</button> <p id="ex23"></p>
2	<script> function funcao() { const listagem = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]; let resultado = ""; let busca = document.getElementById("valor").value; for (let x in listagem) { if (busca == listagem[x]) { resultado += listagem[x] + " "; } } document.getElementById("ex23").innerHTML = resultado; }
3	</body> </html>

Fonte: O autor (2022).

O novo exemplo trazido na figura 11 realiza uma atividade bastante comum em soluções computacionais implementadas para as mais diversas atividades, permitindo que um dado informado pelo usuário no campo de pesquisa seja utilizado como base para uma pesquisa automática por todo um vetor de dados na busca por alguma ocorrência igual.

**TABELA 12 - EXEMPLO DE ESTRUTURA DE REPETIÇÃO EM JAVASCRIPT**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 24&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;&lt;b&gt;Exemplo de Exibição de Vetor&lt;/b&gt;&lt;/p&gt; &lt;button onclick="funcao()"&gt;Listagem&lt;/button&gt; &lt;p id="ex24"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; function funcao() {   const listagem = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];   let mensagem = "";   listagem.forEach(laco);   document.getElementById("ex24").innerHTML = mensagem;    function laco(n) {     mensagem += n + "&lt;br&gt;";   } } &lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

Existe um método em JavaScript que oferece a possibilidade de automatizar o processo de realização de processos para cada elemento de um vetor chamado `forEach()` que ativa uma função que deve ser especificamente implementada para realizar os processos deste método sobre todos os elementos do vetor desejado.

A implementação do que a função `laco()` faz no exemplo da tabela 12 é definida pelo desenvolvedor, de acordo com o que deve ser realizado com os elementos, oferecendo ampla liberdade de atuação sobre os elementos do vetor como estruturas de controle e atribuições de valor, por exemplo.

**TABELA 13 - EXEMPLO DE ESTRUTURA DE REPETIÇÃO EM JAVASCRIPT**

1	<pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 25&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;&lt;b&gt;Digite um valor:&lt;/b&gt;&lt;/p&gt; &lt;input id="valor" value="0" /&gt; &lt;button onclick="funcao()"&gt;Laço&lt;/button&gt; &lt;p id="ex25"&gt;&lt;/p&gt; </pre>
2	<pre> &lt;script&gt; function funcao() {   let valor = document.getElementById("valor").value;   let mensagem = "";   while (valor &gt;= 0) {     mensagem += valor + "&lt;br&gt;";     valor--;   }   document.getElementById("ex25").innerHTML = mensagem; } &lt;/script&gt; </pre>
3	<pre> &lt;/body&gt; &lt;/html&gt; </pre>

**Fonte:** O autor (2022).

Neste último exemplo contemplando estruturas de repetição da tabela 13, temos a estrutura que se baseia na palavra reservada while que oferece a oportunidade de relacionar a quantidade de iterações a serem realizadas a uma condição lógica indicada como é feito nos comandos de decisão.

O valor informado pelo usuário no campo do formulário é avaliado pela condição, e se for maior ou igual a zero, inicia a sequência de iterações que inicia um processo de contagem regressiva controlado pelo decremento do valor informado pelo usuário a cada repetição, armazenando o valor e uma quebra de linha <br> a cada iteração em uma variável a ser exibida ao final da execução estrutura de repetição.

O uso de estruturas de controle se mostrou muito importante para o desenvolvimento de scripts funcionais e capazes de interagir com formulário e páginas web, tornando estas muito mais completas e oferecendo recursos bastante úteis como a possibilidade de validação de dados, consultas e processamento em geral dos dados inseridos dos campos ou outros que possam ser obtidos e utilizados nos scripts.

Sigam se aperfeiçoando nos estudos desta parte do conteúdo, pois existe uma infinidade de possibilidades associadas ao uso de estruturas de controle e outras variantes de comandos que podem ser aprendidos e praticados para uso posterior.



#### 4. APRESENTANDO JSON

Para completar os estudos nesta unidade, vamos tratar de como se pode gerenciar dados que podem ser obtidos e utilizados por páginas web de forma que não fiquem restritos ao momento em que a página está ativa no navegador, sendo perdidos com uma mudança de página ou encerramento do navegador.

Utilizando mecanismos que possibilitam dados transitarem do equipamento onde se encontra um navegador ativo com conteúdo web sendo exibido para outros equipamentos através da comunicação de dados gera a possibilidade de armazenamento permanente de dados e a geração de soluções que trabalhem com bancos de dados.

JSON (JavaScript Object Notation) representa um padrão implementado para a comunicação de dados que se tornou muito comum para aplicações entre servidores de dados e aplicações e equipamentos que funcionam como clientes destes servidores para acessar recursos e dados.

O formato definido pelo padrão JSON é baseado na sintaxe utilizada para objetos em JavaScript, facilitando o aprendizado e mantendo uma padronização em texto que pode ser utilizada por outras linguagens de programação para desenvolvimento web, pois não é um recurso restrito ao JS.

Para a padronização do formato para dados, JSON utiliza pares nome e valor para identificação de campos de um conjunto de dados que são separados por vírgulas. Chaves delimitam estes conjuntos de dados que são representados como matrizes de objetos que são então englobados como um todo utilizando-se colchetes.

**TABELA 14 - EXEMPLO DE ESTRUTURA DE REPETIÇÃO EM JAVASCRIPT**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 26&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;&lt;b&gt;Consulta de Estoque de Veículos&lt;/b&gt;&lt;/p&gt; &lt;p&gt;&lt;b&gt;Digite a marca de veículo desejada:&lt;/b&gt;&lt;/p&gt; &lt;input id="carro" value="" /&gt; &lt;button onclick="funcao()"&gt;Laço&lt;/button&gt; &lt;p id="ex26"&gt;&lt;/p&gt;</pre>
2	<pre>&lt;script&gt; function funcao() {   let carro = document.getElementById("carro").value;   let dados = '{"veiculos":[' +     '{"marca":"GM", "modelo":"Onix LT 1.0", "ano":"2019"},' +     '{"marca":"VW", "modelo":"Gol 1.6 MSI", "ano":"2018"},' +     '{"marca":"Fiat", "modelo":"Argo Drive 1.0", "ano":"2020"}]};</pre> <p>2</p> <pre>const obj = JSON.parse(dados); for (let i=0; i&lt;=2; i++){   if (carro == obj.veiculos[i].marca){     document.getElementById("ex26").innerHTML =       obj.veiculos[i].modelo + " " + obj.veiculos[i].ano;   } }</pre> <p>3</p> <pre>&lt;/script&gt;</pre>
3	<pre>&lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

Avaliando o exemplo da tabela 14, temos a aplicação do formato JSON combinado com script JS para criar uma matriz de objetos do tipo texto que serve como base de dados para uma consulta associada ao dado inserido pelo usuário no campo do formulário.

O formato básico JSON é um pouco diferente do que é apresentado no exemplo, pois é preciso inserir os dados dos objetos e toda a sintaxe JSON dentro de aspas simples para que possa ser compreendido pelo script, mas abaixo podemos observar como seria o padrão JSON puro sem a parte relativa ao script JS.

```

        {"veiculos": [
            {"marca": "GM", "modelo": "Onix"},
            {"marca": "VW", "modelo": "Gol"},
            {"marca": "Fiat", "modelo": "Argo"}
        ]}
    
```

O formato padrão JSON é bastante similar ao que se utiliza para a definição de vetores de objetos e isso facilita a implementação do padrão em aplicações web, mas é preciso estar atento que para ser aceito em scripts JS, é preciso ajustar tanto a sintaxe quanto a forma com que os dados podem ser utilizados.

O método `JSON.parse()` converte as strings JSON em objetos JavaScript para que os dados e estruturas que os contém possam ser utilizados, sendo os dados definidos no próprio script JSON, quanto obtidos em um servidor web.

**TABELA 15 - EXEMPLO DE ESTRUTURA DE REPETIÇÃO EM JAVASCRIPT**

1	<pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 27&lt;/h2&gt; &lt;hr&gt; &lt;p id="ex271"&gt;&lt;/p&gt; &lt;p&gt;&lt;b&gt;Edição de Dados em Objeto&lt;/b&gt;&lt;/p&gt; &lt;input id="tel" value="3564421487" /&gt; &lt;p&gt;&lt;b&gt;Digite o novo telefone:&lt;/b&gt;&lt;/p&gt; &lt;button onclick="funcao()"&gt;Atualizar Dados&lt;/button&gt; &lt;p id="ex272"&gt;&lt;/p&gt; &lt;p id="ex273"&gt;&lt;/p&gt; </pre>
2	<pre> &lt;script&gt; const agenda = {nome:"Ana", fone: 32326565}; document.getElementById("ex271").innerHTML = agenda.nome + " - " + agenda.fone; function funcao() {     agenda["fone"] = document.getElementById("tel").value;     document.getElementById("ex272").innerHTML = "Dados atualizados";     document.getElementById("ex273").innerHTML = agenda.nome + " - " + agenda.fone; } &lt;/script&gt; </pre>
3	<pre> &lt;/body&gt; &lt;/html&gt; </pre>

Fonte: O autor (2022).

Neste novo exemplo da tabela 15 temos o uso do script para editar dado contido em um objeto de forma que a entrada de dados fornecida pelo usuário no campo de formulário seja armazenada no lugar do número de telefone anteriormente armazenado no objeto.

Após conhecer boa parte dos recursos da linguagem JavaScript, é interessante que se mantenha aprendendo mais, pesquisando conteúdo adicional e exercitando a implementação de muitos scripts variados para os conceitos tratados no material para adquirir experiência e maior habilidade.

## SAIBA MAIS

Quando desejamos enviar dados armazenados em estruturas do tipo objeto, a sintaxe padrão JavaScript deve ser convertida para texto, formando uma string única com todos os campos e seus respectivos dados indicados de acordo com um padrão de formato para a string.

O método `stringfy()` é responsável por realizar este ajuste, simplificando o processo, pois é preciso adicionar diversas aspas delimitadoras para identificar campos, além das já utilizadas em dados do tipo texto.

```
const obj = {nome: "Ana", fone: 32326565};  
const objetoJSON = JSON.stringify(obj);  
document.getElementById("id000").innerHTML = objetoJSON;
```

Um script como o exemplificado acima utiliza o método `stringfy()` para unir os campos nome e fone com seus respectivos dados em uma string única formatada como abaixo:

```
{"nome":"Ana","fone":32326565}
```

Com isto, percebe-se que o uso do método específico para esta finalidade se mostra bastante útil, simplificando o processo de implementação de scripts para conversão de dados de objetos JavaScript em dados padrão JSON para envio a servidores web.

**Fonte:** O autor (2022).

## REFLITA

Trabalhar a capacidade lógica não é fácil, pois não é algo que é aprendido geralmente em sala de aula e acaba sendo um divisor de águas na graduação muitas vezes, mas com esforço e persistência, pode ser trabalhada ao nível de se poder programar profissionalmente como os demais da área.

# CONSIDERAÇÕES FINAIS

Após encerrados os estudos nesta unidade, é possível dizer que uma base de estudo sobre a linguagem JavaScript foi concluída, tendo muito ainda o que conhecer e praticar para se atingir um alto grau de conhecimento na linguagem.

Nesta unidade foram trabalhados conceitos essenciais de estruturas de controle que permitem que inúmeras decisões e processos sejam automatizados e controlados pelo próprio script sem a necessidade de interação do usuário.

Estas chamadas estruturas de controle incluem estruturas de decisão que permitem que o script decida muito do que processar a partir de expressões lógicas, assim como estruturas de repetição que podem repetir processos indefinidamente até que uma condição seja atendida.

Por fim, foi apresentado o padrão JSON que pode ser combinado com JavaScript para oferecer meios de se enviar e receber dados através de comunicação entre equipamentos em rede ou pela web, seguindo padrões que facilitam o desenvolvimento de aplicações web.

## LEITURA COMPLEMENTAR

Um importante recurso da linguagem HTML é chamado de DOM (Document Object Model) que padroniza o acesso a páginas web através de uma estrutura que organiza o uso dos elementos HTML de forma geral que foi definido pelo W3C (World Wide Web Consortium).

A estrutura geral baseada em tags <html>, <head><title><body> e as demais tags de estruturação de páginas web possuem uma semântica e sintaxe padronizada que pode ser representada em forma de árvore definida pelo DOM que pode ser toda manipulada pela linguagem JavaScript.

Existem três padrões DOM que foram criados, sendo um Core DOM para todo tipo de página web, outro XML DOM para padrões XML e outro HTML DOM para o padrão HTML, sendo este relevante para os estudos neste material.

Este padrão funciona como um CRUD para HTML (elementos, propriedades, métodos e eventos), sendo que métodos HTML DOM representam ações que podem ser realizadas em elementos HTML, assim como propriedades estão relacionadas a dados que podem ser manipulados destes elementos.

Alguns métodos DOM são utilizados em exemplos neste material como os métodos getElementById() para buscar elementos HTML, e innerHTML para atribuir dados a elementos.

```
  
<script>  
document.getElementById("myImage").src = "landscape.jpg";  
</script>
```

O script acima traz como exemplo de método DOM, getElementById() que fornece a indicação do arquivo de imagem a ser utilizado como fonte para o elemento de imagem HTML representado pela tag <img>, sobrepondo o que havia sido originalmente na tag como uma imagem do tipo gif por outra do tipo jpg, mostrando como os métodos DOM aplicados em scripts podem influenciar os elementos HTML.

**Fonte:** Adaptado de JavaScript HTML DOM - Changing HTML (online).

# MATERIAL COMPLEMENTAR



## LIVRO

**Título:** JavaScript: O Guia Definitivo

**Autor:** David Flanagan.

**Editora:** O'Reilly

**Sinopse:** Livro bastante completo sobre a linguagem JavaScript e bibliotecas JavaScript mais utilizadas em soluções para uso em aplicações cliente-servidor, servindo como um guia de referência da linguagem e boa opção de material de consulta.



David Flanagan



## FILME/VÍDEO

**Título:** Jogador N° 1

**Ano:** 2018.

**Sinopse:** Filme com uma temática interessante, pois mostra um futuro fictício não tão distante de uma possível realidade onde todos vivem vidas intensas num mundo virtual para fugir de vidas frustrantes no mundo real, trazendo um paralelo com a temática do que é o chamado front end e o back end na web.

# UNIDADE IV

## Frameworks Javascript

Professor Especialista Ronie Cesar Tokumoto



### Plano de Estudo:

- Apresentando JQuery;
- Conhecendo Node.JS;
- Interfaces com React;
- Complementando HTML com AngularJS.

### Objetivos da Aprendizagem:

- Conceituar e contextualizar o uso de frameworks para HTML e JavaScript;
  - Conhecer diferentes frameworks e como podem ser utilizados;
  - Compreender as diferentes aplicações e sintaxe dos frameworks;
  - Saber como diferenciar frameworks com base em suas características.

# INTRODUÇÃO

Após conhecer as bases da linguagem JavaScript, é importante seguir avançando nos estudos da linguagem para desenvolver melhor suas habilidades com a linguagem, e aos poucos, estruturar um conhecimento mais completo em relação a como desenvolver conteúdo web de forma geral.

Páginas em HTML combinadas com JavaScript oferecem experiências de usuário mais dinâmicas e variadas, e adicionando o uso de frameworks ao desenvolvimento web, propicia-se maior padronização dos trabalhos e facilidade de reaproveitamento de código em futuros trabalhos.

Para esta unidade, a intenção é apresentar um pouco de alguns frameworks conhecidos e bastante populares no mercado de desenvolvimento como Node-JS, React, AngularJS e JQuery que representam uma boa fatia do que se utiliza mundialmente no mercado, lembrando que existem outros frameworks alternativos ou complementares a estes que podem ser estudados também.

O HTML citado nas unidades anteriores, assim como a linguagem JavaScript estudada ainda se manterão relevantes nesta unidade, pois são necessários para o uso dos frameworks de maneira geral.

# jQuery

## 1. APRESENTANDO JQuery

A partir do ponto em que se conhece um pouco sobre HTML e JavaScript, sendo CSS também relevante, pode-se avançar nos estudos e conhecer meios de simplificar o processo de implementação de scripts para diversas finalidades através do uso de soluções prontas implementadas com base na linguagem JavaScript.

Os chamados frameworks funcionam como pacotes completos de funcionalidades que podem ser utilizadas no desenvolvimento de conteúdo web, visando obter com o uso destes recursos, maior padronização na elaboração de páginas, muito código pronto para scripts, além de recursos complementares como efeitos para elementos HTML.

Um dos frameworks mais relevantes do mercado é JQuery que possui recursos em grande quantidade para complementar o que se pode produzir em HTML e CSS principalmente, trazendo recursos para manipulação do que é produzível por estas duas linguagens, além de recursos adicionais e plugins que são necessários para muitas atividades.

JQuery é uma biblioteca de livre uso que para ser utilizada em páginas web, necessita ser adicionada a um script através da indicação de um link para acesso direto da mesma através da web como é exibido no exemplo da tabela 39, por exemplo, ou baixando a biblioteca toda em uma pasta local na máquina para implementação e teste de scripts.

A pasta para inclusão da biblioteca normalmente é a mesma onde se localizam as páginas web desenvolvidas de forma que basta a simples indicação do arquivo da biblioteca baixada no comando src do script para que se possa acessar seus recursos.

Utiliza-se a linha <script src="jquery-3.6.0.min.js"></script> para a biblioteca contida na pasta das páginas web.

**TABELA 1 - EXEMPLO DE HTML COM JAVASCRIPT USANDO FUNÇÕES**

1	<!DOCTYPE html> <html> <head>
2	<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script> < <script> \$(document).ready(function(){ \$("button").click(function(){ \$(this).hide(); }); \$("p").click(function(){ \$(this).hide(); }); }); </script>
3	</head> <body> <h2>JavaScript - Exemplo 28</h2> <hr> <button>Clicar</button> <p><b>CLICAR</b></p> </body> </html>

**Fonte:** O autor (2022).

O exemplo da tabela 1 traz a aplicação simples de um método de ocultamento de elementos HTML chamado `hide()` como botões ou parágrafos representados pelas tags `<button>` e `<p>` respectivamente, associado ao evento de clique do mouse em um destes elementos.

O exemplo utiliza a biblioteca diretamente da versão hospedada na Google para acessar os recursos citados de forma direta sem a necessidade de que esteja a biblioteca baixada e disponibilizada na pasta onde fica a estrutura de páginas web como já citado.

Um detalhe relevante relacionado ao script do exemplo é a inserção de duas tags `<script>`, sendo a primeira apenas para indicar a localização da biblioteca JQuery, e a segunda para conter a parte de comandos JQuery propriamente dita.

Outro aspecto importante é a colocação dos scripts JQuery dentro da área de cabeçalho do HTML delimitada por tags `<head>`, antes do corpo principal da página delimitado por tags `<body>`, ao invés do JavaScript que nos exemplos estudados, vinham após o fechamento da tag `<body>`.

Assim como em Javascript e HTML, existe uma sintaxe que define a forma como os comandos devem ser implementados, e no caso de JQuery, a forma torna bastante intuitiva a programação, pois indica nos comandos, o elemento HTML sendo tratado por algum método da biblioteca.

A sintaxe em JQuery também necessita do uso do caractere `$` no início de cada linha de comando nos scripts, e este caractere é, por padrão da sintaxe, seguido de um elemento HTML entre parênteses, e após o ponto, um evento a ser processado de acordo com o método indicado.

Assim, seguindo a sintaxe padrão de `$(elemento HTML).método()` é utilizada como padrão em scripts JQuery e isto facilita bastante o aprendizado da biblioteca para uso imediato em aplicações web.

Algo da biblioteca JQuery que é interessante manter como hábito é a inserção da linha `$(document).ready(function())` como principal instrução do script, tendo todas as demais linhas de script de ações a serem realizadas contidas como bloco de instruções desta linha, sendo delimitadas por `{ }` como visto também no exemplo da tabela 2.

**TABELA 2 - EXEMPLO DE HTML COM JAVASCRIPT USANDO FUNÇÕES**

1	<!DOCTYPE html> <html> <head>
2	<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script> <script> \$(document).ready(function(){ \$("button").click(function(){ \$("#ex29").hide(); }); \$("p").click(function(){ \$(this).hide(); }); }); </script>
3	</head> <body> <h2>JavaScript - Exemplo 29</h2> <hr> <button>Clicar</button> <p id="ex29">CLICAR</p> </body> </html>

**Fonte:** O autor (2022).

O exemplo trazido na tabela 2 oferece uma pequena modificação em relação ao exemplo da tabela 1 que utiliza o elemento this para indicar que o próprio elemento botão deve ser oculto pelo método hide() para a indicação de um elemento ex29 específico indicado no lugar do termo this.

Além dos métodos que realizam ações, é importante conhecer os eventos relacionados à linguagem HTML disponíveis na biblioteca como click para mouse, keypress para teclado, e submit para formulários, sendo a lista de eventos da biblioteca bastante extensa.

**TABELA 3 - EXEMPLO DE HTML COM JAVASCRIPT USANDO FUNÇÕES**

1	<pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt; </pre>
2	<pre> &lt;script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"&gt;&lt;/script&gt; &lt;script&gt; \$(document).ready(function(){   \$("p").mouseenter(function(){     \$(this).css("background-color", "LightBlue");     alert("Soma realizada!");   });   \$("p").mouseleave(function(){     \$(this).css("background-color", "white");     alert("Subtração realizada!");   });   \$("input").focus(function(){     \$(this).css("background-color", "LightBlue");   });   \$("input").blur(function(){     \$(this).css("background-color", "white");   }); }); &lt;/script&gt; </pre>
3	<pre> &lt;/head&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 30&lt;/h2&gt; &lt;hr&gt; Primeiro valor: &lt;input id="v1" type="text" name="v1"&gt; &lt;br&gt; Segundo valor: &lt;input id="v2" type="text" name="v2"&gt; &lt;br&gt; &lt;button id="botao1" onclick="funcaoSomar()"&gt;Somar&lt;/button&gt; &lt;br&gt; &lt;button id="botao2" onclick="funcaoSubtrair()"&gt;Subtrair&lt;/button&gt; &lt;br&gt; Soma: &lt;p id="somar"&gt;&lt;/p&gt; Subtração: &lt;p id="subtrair"&gt;&lt;/p&gt; </pre>

	<pre> &lt;script&gt;  function funcaoSomar() {     let x = document.getElementById("v1").value;     let y = document.getElementById("v2").value;     document.getElementById("somar").innerHTML = parseInt(x)     + parseInt(y); }  function funcaoSubtrair() {     let x = document.getElementById("v1").value;     let y = document.getElementById("v2").value;     document.getElementById("subtrair").innerHTML = parseInt(x) - parseInt(y); }  &lt;/script&gt; </pre>
	<pre> &lt;/body&gt; &lt;/html&gt; </pre>

**Fonte:** O autor (2022).

Avançando no uso de JQuery, temos um exemplo mais completo na tabela 3 que foi dividido em mais partes que a forma como era feito anteriormente, onde se dividia o script em HTML, JS e HTML, e agora existe o elemento adicional da aplicação da biblioteca nos scripts também.

Isso fez com que o script todo pudesse ser dividido em mais partes, sendo três de elementos HTML, uma para o script JQuery e outra para o script JavaScript, sendo que a tag script aparece nas duas partes, e para diferenciar o tipo de script, JQuery utiliza o caractere \$ antes de cada par de elemento e método como base de sua sintaxe.

A biblioteca JQuery primeiramente é ativada no primeiro script, e em seguida, abre a parte efetiva do script com o método ready() aplicado à página de forma a ativar os demais comandos do script apenas após o completo carregamento da página e execução do HTML.

Estando a página carregada, são ajustados os elementos da página através da indicação destes entre aspas como a passagem do mouse sobre os conteúdos associados às tags <p> para mudar a cor de fundo quando o mouse está sobre o conteúdo, e após não estar mais, retornar a branco o fundo, assim como a mudança de cor de fundo da caixa de entrada quando se clica para digitação, retornando a cor original após sair do campo.

O outro script relativo ao JavaScript puro trabalha com os dados inseridos nos dois campos de forma que se o usuário clica no botão de soma, a operação é calculada e o resultado exibido, o mesmo ocorrendo ao se clicar no botão de subtração, realizando o outro tipo de cálculo e exibindo o resultado no local indicado da página.

**TABELA 4 - EXEMPLO DE HTML COM JAVASCRIPT USANDO FUNÇÕES**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;head&gt;</pre>
2	<pre>&lt;script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"&gt; &lt;/script&gt; &lt;script&gt; \$(document).ready(function(){   \$(".efeito").click(function(){     \$("p").fadeOut();     \$("p").fadeIn();     \$("#ex31").animate({height: "300px"});     \$("#ex31").animate({width: "300px"});     \$("#ex31").animate({opacity: "0.5"});     \$("#ex31").animate({height: "100px"});     \$("#ex31").animate({width: "100px"});     \$("#ex31").animate({opacity: "1"});   });   \$(".conteudo").click(function(){     \$("p").after("&lt;p&gt;CONTEÚDO ADICIONADO&lt;/p&gt;");   }); }); &lt;/script&gt;</pre>
3	<pre>&lt;/head&gt; &lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 31&lt;/h2&gt; &lt;hr&gt; &lt;p&gt;EXEMPLO DE CONTEÚDO&lt;/p&gt; &lt;button class="efeito"&gt;Efeito&lt;/button&gt; &lt;button class="conteudo"&gt;Conteúdo&lt;/button&gt; &lt;div id="ex31" style="background:#ffff00;height:100px;wid- th:100px;margin:6px;"&gt;&lt;/div&gt; &lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

Complementando os estudos da biblioteca JQuery, o exemplo da tabela 4 traz a aplicação de efeitos visuais mais voltados à aparência que ao processamento de dados, mostrando alternativas para incremento na experiência de usuário que pode ser oferecida em páginas web, lembrando que efeitos que não tenham relevância em termos de elevar a qualidade do conteúdo podem não ser muito bem aceitas em muitos tipos de aplicações, e por isto, devem ser dosadas adequadamente de acordo com o tipo de aplicação.

No exemplo são utilizados efeitos de mudança de tamanhos de um elemento do tipo container representado pela tag `<div>` que são ativados por um botão, que no caso funciona como elemento gráfico apenas, aumentando suas medidas e depois aplicando uma redução de opacidade, para depois, reduzir novamente suas medidas e depois retornando à opacidade para 100%.

Outro botão utiliza o método `after()` para adicionar conteúdo após o elemento parágrafo da tag `<p>`, sendo um ponto de atenção que no caso de novos cliques irem adicionando conteúdo a todos os já exibidos na página, gerando um efeito gradativamente maior, ou seja, duplicando o conteúdo do tipo parágrafo na página a cada clique.

Isso mostra que é importante avaliar cada recurso JQuery a ser adicionado a páginas HTML, testar seus efeitos e ponderar sua utilidade e capacidade de melhoria em termos de resultados obtidos e capacidade de melhorar a experiência do usuário (UX).

# Node.js

## 2. CONHECENDO Node.JS

Para executar scripts JavaScript em servidores, Node.js é uma opção bastante utilizada no mercado, sendo um ambiente para servidor gratuito que pode ser utilizado em diferentes plataformas como Windows, Linux e Mac OS X segundo o W3Schools (online).

É necessário baixar seu conteúdo no site oficial <https://nodejs.org>, observando a escolha da versão adequada para sua plataforma, para depois poder iniciar testes com a implementação de scripts de conteúdo dinâmico para servidores.

Segundo W3Schools (online a), por trabalhar de forma assíncrona, os scripts têm seus comandos executados sequencialmente, e mesmo que o processamento a ser realizado por um comando não tenha sido encerrado ainda, é possível que um próximo da sequência possa ser iniciado.

Um exemplo é na solicitação de abertura de um arquivo ou impressão de um documento. Em determinados casos, seria necessário o processo aguardar resposta dos dispositivos de hardware confirmando a possibilidade de execução da tarefa para que outros processos pudessem ser iniciados, mas o Node.js permite que o próximo processo seja adiantado enquanto o processo anterior aguarda resposta do hardware.

**TABELA 5 - EXEMPLO DE APLICAÇÃO NODE.JS.**

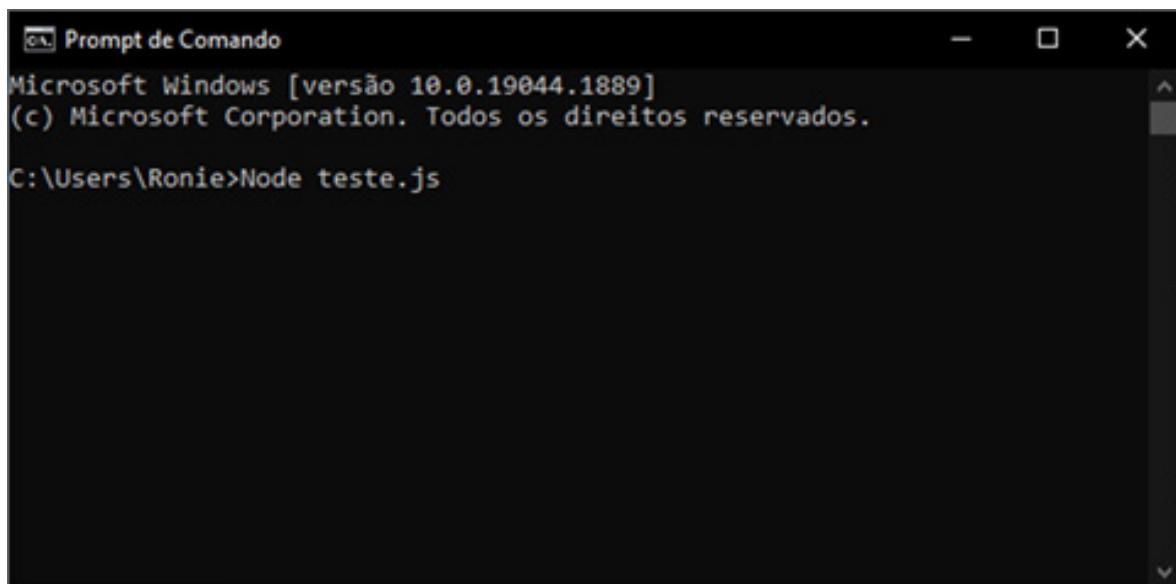
1	var http = require('http'); http.createServer(function (req, res) { res.writeHead(200, {'Content-Type': 'text/plain'}); res.end('Teste de Node.JS!'); }).listen(8080);
2	<b>Node teste.js</b>
3	<b>http://localhost:8080</b>

**Fonte:** Adaptado de: W3Schools (online b).

Neste exemplo de uso da Node.js da tabela 5, é considerado que o software já foi baixado e instalado no equipamento para teste dos scripts para então realizar as três etapas contidas no exemplo, sendo a primeira delas, utilizar um editor de textos simples para inserir a primeira parte do exemplo e gravar o arquivo em um local desejado com um nome desejado e a extensão js. Para o exemplo, foi utilizado o nome teste.js para o exemplo.

Depois de gravado o arquivo com o script, deve ser acionar o prompt de comandos do sistema operacional, acessar a pasta onde foi gravado o script do exemplo e utilizar o comando node teste.js para inicializar o node com base no script e deixá-lo ativo como mostrado na figura 1.

**FIGURA 1 - PROMPT DE COMANDO PARA INICIALIZAR NODE.JS**

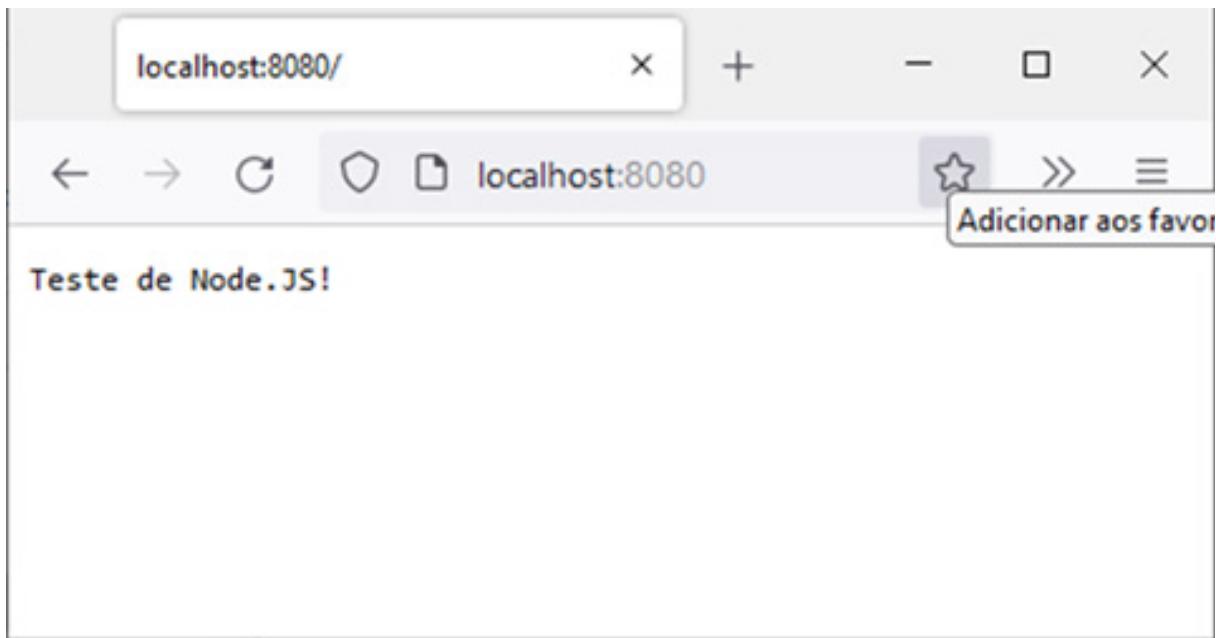


**Fonte:** O autor (2022).

Ao executar o comando, nada aparecerá no prompt de comandos, pois ocorreu apenas um processo de inicialização de um serviço que ficará em execução até ser interrompido com um comando CTRL+C, por exemplo.

Depois, utilizando o endereço http da terceira parte da tabela 5, é executado o script programado para responder solicitações realizadas pela porta 8080 que, entre outras finalidades, permite a execução de um servidor da Web por um usuário.

**FIGURA 2 - RESULTADO DO TESTE DA PORTA NO NAVEGADOR**



**Fonte:** O autor (2022).

Ao realizar o teste de comunicação com a porta 8080 utilizando o endereço indicado na tabela 5, o resultado exibido na figura 2 é obtido e serve para demonstrar como se pode utilizar o Node.js para lidar com conteúdo inserido em um servidor web.

É possível executar métodos JavaScript direto do servidor utilizando Node.js de forma que comandos possam ser executados remotamente e seus resultados trazidos para a solicitação da porta 8080, por exemplo.

**TABELA 6 - EXEMPLO DE APLICAÇÃO NODE.JS**

<b>1</b>	Arquivo biblioteca.js
<b>2</b>	<code>exports.myDateTime = function () {     return Date(); };</code>
<b>3</b>	Arquivo teste.js
<b>4</b>	<code>var http = require('http'); var dt = require('./biblioteca');  http.createServer(function (req, res) {     res.writeHead(200, {'Content-Type': 'text/html'});     res.write("Data e Hora atuais: " + dt.myDateTime() + "&lt;br&gt;");     res.end('Teste de uso de biblioteca externa!'); }).listen(8080);</code>
<b>5</b>	Node teste.js
<b>6</b>	<code>http://localhost:8080</code>

**Fonte:** Adaptado de W3Schools (online b).

O exemplo da tabela 6 cria duas situações importantes indicadas nas partes 1 a 4, onde nas partes 1 e 2, um script de apoio que funciona como uma biblioteca específica criada para o exemplo em JavaScript é gravada com o nome de biblioteca.js para implementar uma função que retorna a data e hora atuais do sistema pelo uso do Objeto Date().

Depois, nas partes 3 e 4 do exemplo, a variável dt é associada à biblioteca para poder ser utilizada no servidor para obtenção da data solicitada na função pelo método write() para ser exibida no navegador antes de uma mensagem simples para complementar o script apenas.

Por ser um framework voltado ao uso com a comunicação de dados em aplicações cliente-servidor, existem recursos implementados em Node.js que simplificam bastante os processos relacionados a comunicação via web, por exemplo.

Existe um módulo próprio para uso em aplicações que necessitam se comunicar pelo protocolo HTTP padrão da Internet que deve ser ativado logo no início de um script através da linha de comando `var http = require('http');` que incorpora o módulo ao script.

**TABELA 7 - EXEMPLO DE APLICAÇÃO NODE.JS**

	<pre>var http = require('http');  1 http.servidor(function (req, serv) {   serv.write('Teste de comunicação via HTTP');   serv.end(); }).listen(8080);</pre>
2	Node teste.js
3	http://localhost:8080

**Fonte:** Adaptado de: W3Schools (online f).

O script simples de exemplo de comunicação via HTTP contido na tabela 7 traz uma simples função que ativa um servidor web que fornece a partir da porta 8080, uma mensagem simples ilustrando a conexão como ativa e funcional.

O método write() permite a saída de dados para o navegador através da conexão com o servidor local, o método end() encerra finaliza o envio de dados pela conexão ativa momentaneamente, e o método listen() define o canal de comunicação através da porta 8080.

Outro módulo importante é o de gerenciamento de arquivos de dados em disco através de um módulo específico chamado de fs (file system) que é habilitado pela linha de comando var fs = require('fs'); também no início de um script.

**TABELA 8 - EXEMPLO DE APLICAÇÃO NODE.JS**

	<pre> var http = require('http'); var fs = require('fs');  http.createServer(function (req, servico) {   fs.appendFile('texto.txt', '\nExemplo de arquivo de dados.\n', function   (err){     if (err) throw err;     console.log('Arquivo gerado com sucesso.');   });   fs.readFile('texto.txt', function(err, data) {     servico.writeHead(200, {'Content-Type': 'text/html'});     servico.write(data);   });   fs.appendFile('texto.txt', '\nNovo conteúdo adicionado.\n', function (err) {     if (err) throw err;     console.log('Conteúdo adicionado.');   });   fs.readFile('texto.txt', function(err, data) {     servico.writeHead(200, {'Content-Type': 'text/html'});     servico.write(data);   });   fs.writeFile('texto.txt', '\nNovo texto.\n', function (err) {     if (err) throw err;     console.log('Alterações realizadas.');   });   fs.readFile('texto.txt', function(err, data) {     servico.writeHead(200, {'Content-Type': 'text/html'});     servico.write(data);     return servico.end();   }); }).listen(8080); </pre>
<b>2</b>	Node teste.js
<b>3</b>	http://localhost:8080

**Fonte:** Adaptado de W3Schools (online f).

O exemplo da tabela 8 traz um exemplo bastante completo de gerenciamento de arquivo em disco, onde cada função adicionada ao script realiza parte de um processo que ilustra o uso cotidiano de arquivos em aplicações em geral.

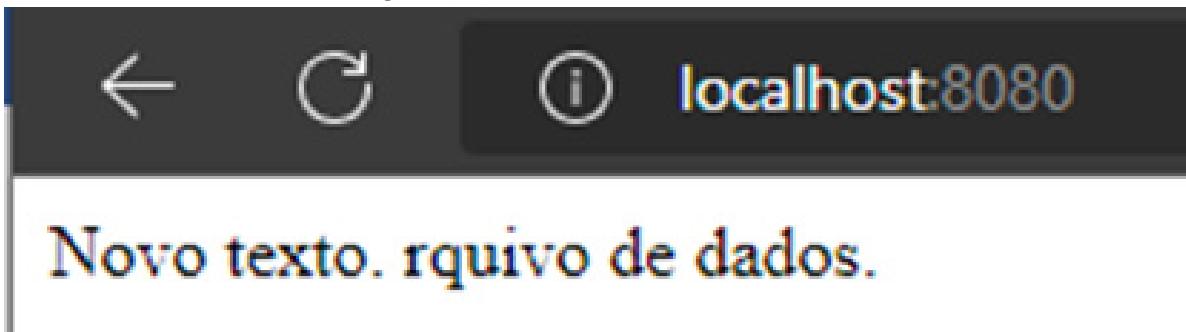
O método `createServer()` inicia o processo através a criação de um servidor que disponibiliza as ações implementadas dentro deste para comunicação via porta 8080 como indicado no método `listen(8080)`.

O método `appendFile()` da biblioteca `fs` permite que um novo arquivo do tipo texto ou contendo script HTML seja criado e determinado conteúdo inserido no mesmo, tendo como complemento uma mensagem de sucesso a ser exibida no prompt de comandos para registro do evento ocorrido.

O método `readFile()` é utilizado mais de uma vez para exibir no navegador o conteúdo do arquivo `texto.txt` criado pela aplicação, de forma a permitir um acompanhamento do que ocorre com o conteúdo do mesmo ao longo da execução do script.

Um detalhe importante do arquivo de dados e sua exibição no navegador, é que não há preocupações com a aparência do que é armazenado no arquivo ou sua exibição no navegador, e simplesmente o método exibe o que estiver no arquivo sobrepondo o que aparece no navegador, ficando com um visual aparentemente estranho como se vê na imagem 3.

**FIGURA 3 - EXIBIÇÃO DE CONTEÚDO DE ARQUIVO DE TEXTO**



Fonte: O autor (2022).

Uma ação final que poderia ser adicionada ao script para encerrar a demonstração da funcionalidade do gerenciamento de arquivos texto, seria um método chamado `unlink()` que permite a exclusão do arquivo gerado e seu conteúdo como é mostrado a seguir.

```
fs.unlink('texto.txt', function (err) {  
  if (err) throw err;  
  console.log('File deleted!');  
});  
}).listen(8080);
```

Por fim, é importante destacar que o framework também aceita a realização de conexões com gerenciadores de bancos de dados como MySQL e MongoDB, por exemplo, permitindo que bancos de dados sejam criados e gerenciados por aplicações web.

Para isto, é preciso instalar algum gerenciador de banco de dados compatível como os dois citados e depois, adicionar o módulo correspondente através do comando `npm install mysql` ou `npm install mongodb`, por exemplo.

Depois, nos scripts é preciso adicionar os módulos através de comandos no início dos scripts como foi feito com os demais módulos utilizando comandos como `var mysql = require('mysql');` ou `var mongo = require('mongodb');`; de acordo com a opção instalada.

**TABELA 9 - EXEMPLO DE APLICAÇÃO NODE.JS**

1	<pre> var mysql = require('mysql');  var con = mysql.createConnection({   host: "localhost",   user: "usuario",   password: "senha" });  con.connect(function(err) {   if (err) throw err;   console.log("Conexão ativa.");   con.query("CREATE DATABASE Agenda", function (err, result) {     if (err) throw err;     console.log("Banco de Dados Agenda criado.");   });   var sql = "CREATE TABLE contatos (nome VARCHAR(255), fone VARCHAR(255))";   con.query(sql, function (err, result) {     if (err) throw err;     console.log("Tabela Contatos criada.");     var sql = "INSERT INTO contatos (nome, fone) VALUES ('Ana', '32326565')";     con.query(sql, function (err, result) {       if (err) throw err;       console.log("Dados armazenados.");     });     con.query("SELECT * FROM contatos", function (err, result, fields) {       if (err) throw err;       console.log(result);     });   }); }); </pre>
2	Node teste.js
3	<a href="http://localhost:8080">http://localhost:8080</a>

**Fonte:** O autor (2022).

Nesse exemplo de gerenciamento de banco de dados usando Node.js da tabela 9, temos etapas que realizam um processo bastante comum em aplicações web que lidam com bases de dados, iniciando primeiramente com uma conexão controlada por login e senha de usuário ao servidor para gerenciar níveis de segurança e acesso.

Depois, o banco de dados Agenda para o exemplo é criado primeiramente, sendo apenas um arquivo vazio através do comando SQL de gerenciamento de bancos de dados CREATE DATABASE.

Com o banco de dados criado, é então criada uma tabela chamada Contatos que terá campos Nome e Fone para que dados possam ser vinculados a eles, assim como foi feito anteriormente em JavaScript onde um objeto Agenda pode ser implementado para que dados também pudessem ser inseridos em campos Nome e Fone.

No caso dos gerenciadores de bancos de dados, estes são bastante automatizados e simplificam o processo de inserção e tratamento de dados através de comandos específicos para estas atividades da linguagem SQL.

A tabela pode ser criada com o uso do comando SQL CREATE TABLE, assim como dados podem ser adicionados a registros que representam conjuntos de dados associados aos campos da tabela com INSERT INTO e consultados pelo comando SELECT.

Entra então neste caso, a necessidade de haver conhecimentos prévios de gerenciadores de bancos de dados como MySQL ou MongoDB e da linguagem de manipulação de bancos de dados SQL que não fazem parte do escopo da disciplina, mas merecem atenção se for do interesse se aprofundar neste tipo de desenvolvimento de software.

# REACT.JS

### 3. INTERFACES COM REACT

Avançando nos estudos, outro framework bastante popular no mercado de desenvolvimento é o React que possui recursos muito elogiados para construção de interfaces de interação de usuários com software.

Além de facilitar o desenvolvimento padronizado de interfaces, potencializa o reuso de código, pois quanto mais padronizado e independente um trecho de código ou função do restante da aplicação, mais facilmente pode ser exportado e reaproveitado em outras aplicações que necessitem da mesma funcionalidade.

Os scripts em React dependem da existência do pacote Node.js instalado, sendo utilizado o gerenciador de pacotes chamado npm que possibilita a instalação e uso de bibliotecas de terceiros através de comandos.

Scripts React podem ser escritos diretamente em HTML, mas para que possa ser utilizado efetivamente como recurso em aplicações reais, é preciso que o pacote Node com seu gerenciador de pacotes esteja instalado previamente, pois bibliotecas React são importadas nos scripts como mostrado no exemplo da tabela 10.

**TABELA 10 - EXEMPLO DE APLICAÇÃO NODE.JS**

1	import React from "react"; import ReactDOM from "react-dom/client"; function teste(props) { return <h1>REACT ATIVADO</h1>; } ReactDOM.render(<teste />, document.getElementById("root"));
2	<b>Node teste.js</b>
3	<b>http://localhost:8080</b>

**Fonte:** Adaptado de W3Schools (online b).

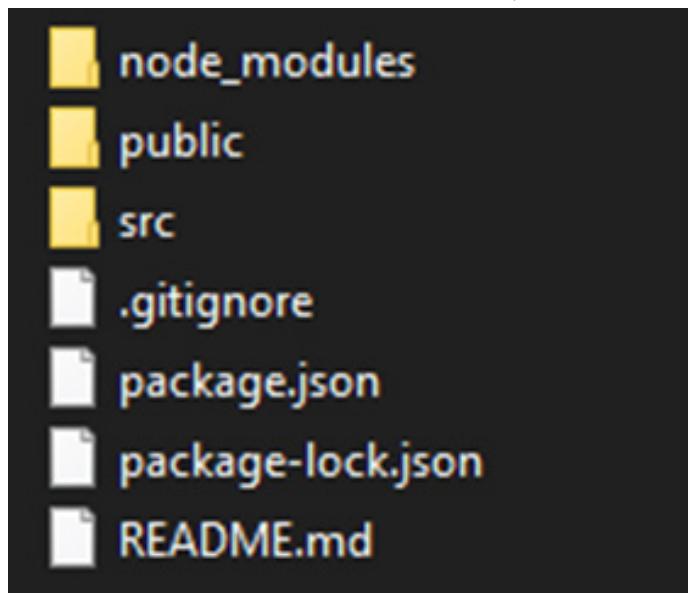
Pelo exemplo da tabela 10, duas bibliotecas são adicionadas por linhas de script utilizando a palavra reservada import, habilitando o uso do método render(), por exemplo que aciona a função teste() que retorna uma mensagem para ser exibida no navegador.

Após criado o script, ele pode ser convertido em uma aplicação real utilizando-se um comando chamado npx que permite através do parâmetro create-react-app gerar uma aplicação com o nome indicado após estes comandos como mostrado a seguir.

**npx create-react-app aplicativo1**

Este comando gera uma estrutura completa para a aplicação como a mostrada na figura 4 que ilustra a composição de pastas e arquivos visíveis na pasta da aplicação criada pelo comando.

**FIGURA 4 - ESTRUTURA DE PASTAS E ARQUIVOS DA APLICAÇÃO**



**Fonte:** O autor (2022).

Observando a estrutura organizada para a aplicação indicada na figura 4, existem arquivos no padrão JSON para definição de configurações da aplicação em relação a navegadores e scripts JS.

Dentro da pasta src existem arquivos JS que contém diferentes componentes da aplicação, mas o que normalmente é utilizado para a personalização de uma aplicação é App.js que contém o script base da aplicação, e para exemplificar a edição deste arquivo, segue um exemplo de script na tabela 11.

**TABELA 11 - EXEMPLO DE APLICAÇÃO REACT**

1	<pre>import logo from './logo.svg'; import './App.css';  function App() {   return (     &lt;div className="App"&gt;       &lt;header className="App-header"&gt;         &lt;img src={logo} className="App-logo" alt="logo" /&gt;         &lt;p&gt;           Para alterar o resultado, modifique o arquivo &lt;code&gt;src/App.js&lt;/code&gt;, grave e veja o resultado.         &lt;/p&gt;         &lt;a           className="App-link"           href="https://reactjs.org"           target="_blank"           rel="noopener noreferrer"         &gt;           Testando Scripts React!         &lt;/a&gt;       &lt;/header&gt;     &lt;/div&gt;   ); }</pre>
2	<pre>export default App;</pre>
4	<pre>http://localhost:8080</pre>

Fonte: Adaptado de: W3Schools (online c).

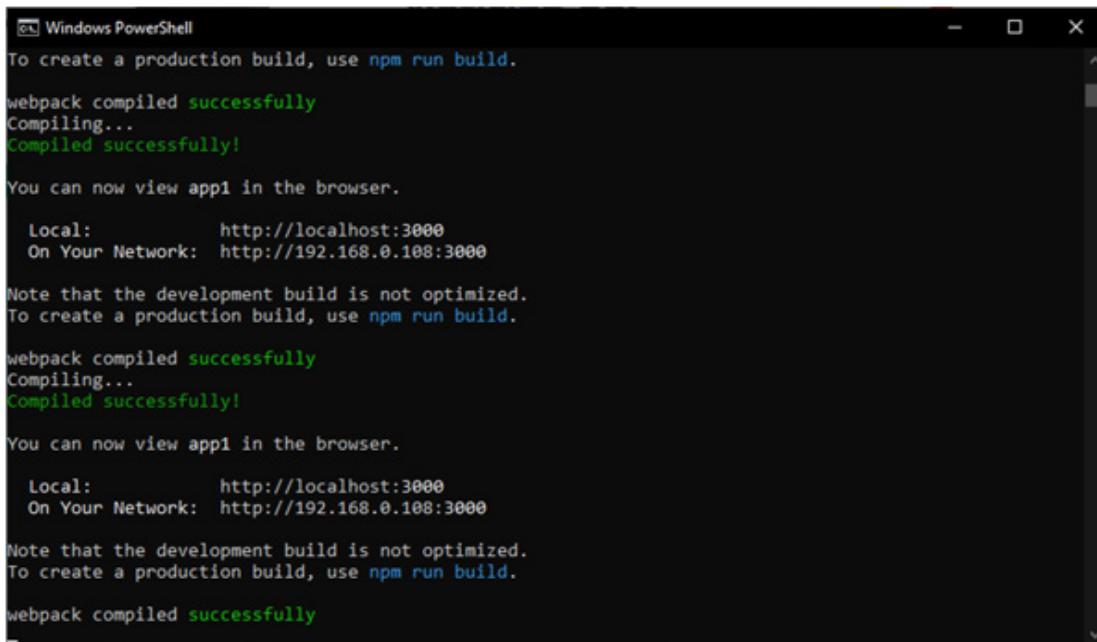
No exemplo da tabela 11, temos na primeira parte a importação de arquivos externos necessários para o funcionamento da aplicação, sendo o primeiro, um arquivo contendo conteúdo gráfico para exibição, e o segundo, um arquivo do tipo CSS de estilos para ajustar elementos HTML utilizados na página.

Na segunda parte, um script React implementa uma função App() com o mesmo nome do arquivo JS que retorna um script completo a ser executado quando houver comunicação com a porta 8080, exibindo elementos HTML estruturados em tags como <img> para mostrar a imagem importada, <p> para exibição de texto na página e <a> para gerar o link que direciona a página para o site da React.

A aplicação é posta em funcionamento através da execução do comando npm start no prompt de comandos, na pasta onde encontra a aplicação que inicia um processo de execução do script, indicando o início do processo logo em seguida com mensagens como > app1@0.1.0 start e react-scripts start que são exibidas no próprio prompt de comandos.

Ao final da preparação da execução, um conteúdo como o exibido na figura 2 é mostrado ao usuário, informando que o script está ativo e preparado para comunicação pela porta 8080 com a mensagem final webpack compiled successfully que indica que não foram encontrados erros que impeçam a execução do script e a compilação da aplicação para execução foi realizada com sucesso.

**FIGURA 5 - ILUSTRAÇÃO DA EXECUÇÃO DA APLICAÇÃO**



```
Windows PowerShell
To create a production build, use npm run build.
webpack compiled successfully
Compiling...
Compiled successfully!
You can now view app1 in the browser.
  Local:      http://localhost:3000
  On Your Network: http://192.168.0.108:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
Compiling...
Compiled successfully!
You can now view app1 in the browser.
  Local:      http://localhost:3000
  On Your Network: http://192.168.0.108:3000
Note that the development build is not optimized.
To create a production build, use npm run build.
webpack compiled successfully
```

Fonte: O autor (2022).

Depois de estar ativa a aplicação, basta acessar o endereço <http://localhost:3000/> em um navegador compatível que a aplicação exibirá seu conteúdo normalmente, com a imagem e as mensagens sendo exibidas na página do navegador.

O conteúdo do script a ser inserido no arquivo App.js é livre e deve apenas seguir a semântica e sintaxe aceitos por React, tanto para comandos específicos do framework, quanto para comandos HTML, por exemplo, de forma que aplicações diversas possam ser executadas e acessadas em navegadores como aplicações web.

**TABELA 12 - EXEMPLO DE APLICAÇÃO REACT**

1	<pre>import { useState } from 'react'; import ReactDOM from 'react-dom/client';</pre>
2	<pre>function App() {   const [nome, novoNome] = useState("");   const [idade, novaldade] = useState(0);    const handleSubmit = (event) =&gt; {     event.preventDefault();     alert(`Nome: \${nome} - Idade: \${idade}`)   }    return (     &lt;form onSubmit={handleSubmit}&gt;       &lt;label&gt;Digite seu nome:         &lt;input type="text" value={nome}           onChange={(e) =&gt; novoNome (e.target.value)}&gt;       &lt;/input&gt;     &lt;/label&gt;     &lt;p&gt; &lt;/p&gt;     &lt;label&gt;Digite sua idade:       &lt;input type="number" value={idade}         onChange={(e) =&gt; novaldade (e.target.value)}&gt;       &lt;/input&gt;     &lt;/label&gt;     &lt;input type="submit" /&gt;   &lt;/form&gt;   ) }  const root = ReactDOM.createRoot(document.getElementById('root')); root.render(&lt;App /&gt;);</pre>
3	<pre>export default App;</pre>
4	<pre>http://localhost:8080</pre>

**Fonte:** Adaptado de: W3Schools (online d).

Observando agora o exemplo da tabela 12, temos logo de início um detalhe relevante que é a utilização de um recurso chamado Hook que é representado pelo uso de `{ useState }` que reduz a necessidade de se implementar uma classe para coletar e tratar estados de um evento como no caso do exemplo, dados para inicialização dos campos nome e idade do formulário pelas linhas de declaração das duas constantes para os campos.

Em seguida, temos a implementação de controle para o que deve ser realizado quando os dados do formulário forem enviados pelo uso do botão Submit, que no exemplo, geram uma mensagem do tipo alerta do navegador, exibindo os dados coletados do formulário.

Um detalhe curioso e bastante diferente do que costumamos ver diretamente no JavaScript é a implementação do formulário todo como retorno da função que terá interação direta com o usuário para obtenção de dados.

**TABELA 13 - EXEMPLO DE APLICAÇÃO REACT**

1	import React from 'react'; import ReactDOM from 'react-dom/client';
2	<b>function App(props) {</b> <b>    alert(`\${valor}`);</b> <b>    if (valor == 7) {</b> <b>        return &lt;h2&gt;Acertou!&lt;/h2&gt;;</b> <b>    } else {</b> <b>        return &lt;h2&gt;Errou...&lt;/h2&gt;;</b> <b>    }</b> <b>}</b> <b>var valor = Math.random();</b> <b>valor = Math.floor(valor * 10);</b> <b>const root = ReactDOM.createRoot(document.getElementById('root'));</b> <b>root.render(&lt;App valor=&gt;);</b>
3	export default App;
4	<b>http://localhost:8080</b>

**Fonte:** Adaptado de: W3Schools (online e).

Por fim, para demonstrar um pouco do uso da lógica de programação usando React, o script da tabela 13 traz uma aplicação simples que utiliza um número entre 0 e 10 gerado aleatoriamente através do método random() da classe Math como parâmetro para a função principal App() para que seja avaliado dentro da condição de ser ou não igual ao número 7.

Uma mensagem de alerta informa o valor gerado aleatoriamente, e caso seja verdadeiro a condição do valor ser igual a 7, é exibida uma mensagem de acerto no navegador, e caso contrário, é informado que errou.

React é um framework bastante interessante para o trabalho com HTML por complementá-lo de forma simples e gradativa, à medida que seja necessário e de acordo com a complexidade do que se deseja implementar.

# Angular

## 4. COMPLEMENTANDO HTML COM ANGULARJS

Como último framework a ser estudado nesta unidade, AngularJS é uma outra opção bastante popular no mercado de desenvolvimento de aplicações web que amplia as funcionalidades através da disponibilização de diretivas com base no JavaScript, podendo ser livremente adicionado a páginas web como um script JS.

**TABELA 14 - EXEMPLO DE SCRIPT ANGULARJS**

1	<!DOCTYPE html> <html>
2	<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"> </script>
3	<body> <h2>JavaScript - Exemplo 32</h2> <hr> <div ng-app=""> <p>Nome: <input type="text" ng-model="name"></p> <p ng-bind="name"></p> </div> </body> </html>

**Fonte:** O autor (2022).

Analisando o exemplo da tabela 14, temos um exemplo de script HTML comum, com as tags essenciais de estruturação de páginas como `<html>` e `<body>`, mas existe um pequeno script JS que serve apenas para que seja carregado o framework AngularJS juntamente com a exibição da página pelo navegador.

Essa biblioteca adicional permite que novos recursos sejam disponibilizados ao desenvolvedor através de diretivas adicionais às tags comuns HTML como `<p>` e `<div>`, agregando maior dinamicidade ao HTML puro.

As diretivas do framework possuem finalidades específicas, sendo `ng-app` utilizada para que o elemento `<div>` possa utilizar recursos do AngularJS inicialmente, para depois `ng-model` atribuir dados de um elemento como `<input>` para uma estrutura de dados indicada, e por fim, `ng-bind` associa o elemento `<p>` à estrutura de dados que recebeu o valor atribuído pela diretiva `ng-model`.

Como padrão, as diretivas fornecidas por AngularJS iniciam todas por `ng` para padronizar nomes de diretivas e facilitar sua identificação em meio a outros comandos e elementos em scripts que podem unir HMTL, CSS, JavaScript e Node.Js, por exemplo.

**TABELA 15 - EXEMPLO DE SCRIPT ANGULARJS**

1	<code>&lt;!DOCTYPE html&gt;</code> <code>&lt;html&gt;</code>
2	<code>&lt;script</code> <code>src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"&gt;</code> <code>&lt;/script&gt;</code>
3	<code>&lt;body&gt;</code> <code>&lt;h2&gt;JavaScript - Exemplo 33&lt;/h2&gt;</code> <code>&lt;hr&gt;</code> <code>&lt;div ng-app="" ng-init="nome='Ana"&gt;</code> <code>&lt;p&gt;Nome: &lt;input type="text" ng-model="nome" value="nome"&gt;&lt;/p&gt;</code> <code>&lt;p&gt;Peso: &lt;input type="text" ng-model="peso"&gt;&lt;/p&gt;</code> <code>&lt;p&gt;Altura: &lt;input type="text" ng-model="altura"&gt;&lt;/p&gt;</code> <code>&lt;hr&gt;</code> <code>&lt;p&gt;IMC = {{peso / (altura*altura)}}&lt;/p&gt;</code> <code>&lt;/div&gt;</code> <code>&lt;/body&gt;</code> <code>&lt;/html&gt;</code>

Fonte: O autor (2022).

O exemplo da tabela 15 traz uma evolução do uso de diretivas e do padrão de uso do framework, oferecendo em tempo real a utilização de dados que vão sendo inseridos pelo usuário em campos de um formulário

Importante observar dois detalhes nesse exemplo que são a inicialização de um valor para o campo nome do formulário indicado como diretiva da própria tag `<div>` que contém o script AngularJS de forma que o campo não apareça em branco na página.

Além desse detalhe, existe a linha responsável por realizar o cálculo do IMC em tempo real onde dentro de uma tag `<p>` é concatenado um trecho de texto para iniciar o texto, e depois, é realizado o cálculo em si, que por definição da sintaxe AngularJS, deve ser escrita entre chaves duplas.

Uma variação possível para a expressão de cálculo do IMC poderia ser escrita como `<p>IMC = <span ng-bind="peso / (altura*altura)"></span></p>` que utiliza a diretiva `ng-bind` para que o resultado da expressão seja exibido no parágrafo, após o texto, sendo colocado dentro de uma tag `<span>` que gera uma espécie de contêiner apenas para delimitar a diretiva e a expressão, isolando do restante do conteúdo do parágrafo para evitar conflitos de sintaxe.

No caso de textos a serem exibidos utilizando a diretiva `ng-bind`, por exemplo, é possível a concatenação simples utilizando o operador `+`, e algo como `<span ng-bind="IMC = " + "peso / (altura*altura)"></span>` poderia ser utilizado se os dados da expressão não fossem de tipo diferente do texto em si.

**TABELA 16 - EXEMPLO DE SCRIPT ANGULARJS**

1	<code>&lt;!DOCTYPE html&gt;</code> <code>&lt;html&gt;</code>
2	<code>&lt;script</code> <code>src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"&gt;</code> <code>&lt;/script&gt;</code>
3	<code>&lt;body&gt;</code> <code>&lt;h2&gt;JavaScript - Exemplo 34&lt;/h2&gt;</code> <code>&lt;hr&gt;</code> <code>&lt;div ng-app="" ng-init="dados={nome:'Ana',peso:'70',altura:'1.72'}"&gt;</code> <code>&lt;p&gt;Nome: &lt;input type="text" ng-model="dados.nome" value=dados.nome&gt;&lt;/p&gt;</code> <code>&lt;p&gt;Peso: &lt;input type="text" ng-model="dados.peso" value=dados.peso&gt;&lt;/p&gt;</code> <code>&lt;p&gt;Altura: &lt;input type="text" ng-model="dados.altura" value=dados.altura&gt;&lt;/p&gt;</code> <code>&lt;hr&gt;</code> <code>&lt;p&gt;IMC = &lt;span ng-bind="dados.peso / (dados.altura*dados.altura)"&gt;&lt;/span&gt;&lt;/p&gt;</code> <code>&lt;/div&gt;</code> <code>&lt;/body&gt;</code> <code>&lt;/html&gt;</code>

Fonte: O autor (2022).

Complementando o exemplo da tabela 15, é possível utilizar outros tipos de estruturas de dados como objetos, por exemplo, tornando mais completo o framework para uso com formulários principalmente, pois objetos podem ser utilizados para implementar todos os dados que podem ser utilizados em formulários como mostrado na tabela 16.

Vetores também podem ser utilizados normalmente através da indicação da estrutura entre chaves duplas ou em contêiner `<span>` com a diretiva `ng-bind`, inicializando os elementos do vetor na declaração da tag `<div>` como feito nos exemplos anteriores, mas ajustando a sintaxe para algo como `<div ng-app="" ng-init="notas=[6,7,9,8]">` e `<p>Segunda nota {{ notas[1] }}</p>`.

**TABELA 17 - EXEMPLO DE SCRIPT ANGULARJS**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;</pre>
2	<pre>&lt;script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"&gt; &lt;/script&gt;</pre>
3	<pre>&lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 35&lt;/h2&gt; &lt;hr&gt; &lt;div ng-app="App" ng-controller="Controle"&gt; Peso = {{ peso }} &lt;br&gt; Altura = {{ altura }} &lt;br&gt; IMC = {{ peso / (altura * altura) }} &lt;/div&gt; &lt;script&gt; var app = angular.module("App", []); app.controller("Controle", function(\$dados) {     \$dados.peso = 70;     \$dados.altura = 1.72; }); &lt;/script&gt; &lt;/body&gt; &lt;/html&gt;</pre>

**Fonte:** O autor (2022).

Além da sintaxe distinta do framework AngularJS, existem diferenças na implementação de aplicações, e um exemplo que altera a forma como se pode desenvolver software é utilizando módulos AngularJS que funcionam como contêineres para conter os componentes de um software baseado no framework como é mostrado no exemplo da tabela 17.

Através da diretriz `ng-app` é possível indicar um nome para a aplicação dentro da tag `<div>` e depois, o método `module()` do objeto angular, e pelo método `ng-controller` também podemos indicar um chamado controlador para que alterações realizadas em dados como as padrões do CRUD, sejam imediatamente visualizadas na página se desejado após serem manipulados.

Depois, uma variável `app` é declarada para receber o próprio módulo e em seguida, o seu controlador receber uma inicialização contendo dados para uso no script, mesmo que a parte em HTML tenha sido implementada linhas antes no script, pois a parte relativa à aplicação e controlador são prioridade na execução.

**TABELA 18 - EXEMPLO DE SCRIPT ANGULARJS**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;</pre>
2	<pre>&lt;script src="https://ajax.googleapis.com/ajax/libs/angular- js/1.6.9/angular.min.js"&gt; &lt;/script&gt;</pre>
3	<pre>&lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 36&lt;/h2&gt; &lt;hr&gt; &lt;div ng-app=""&gt; &lt;form&gt;   Seleccione as Opções Desejadas do Menu:&lt;br&gt;   &lt;input type="checkbox" ng-model="opcao1"&gt;Bebidas   &lt;input type="checkbox" ng-model="opcao2"&gt;Pratos   &lt;input type="checkbox" ng-model="opcao3"&gt;Sobremesas &lt;/form&gt; &lt;hr&gt; &lt;h2 ng-show="opcao1"&gt;Bebidas&lt;/h2&gt;   &lt;h3 ng-show="opcao1"&gt;1 - Refrigerante&lt;/h3&gt;   &lt;h3 ng-show="opcao1"&gt;2 - Água&lt;/h3&gt; &lt;h2 ng-show="opcao2"&gt;Pratos&lt;/h2&gt;   &lt;h3 ng-show="opcao2"&gt;1 - Feijoada&lt;/h3&gt;   &lt;h3 ng-show="opcao2"&gt;2 - Massas&lt;/h3&gt; &lt;h2 ng-show="opcao3"&gt;Sobremesas&lt;/h2&gt;   &lt;h3 ng-show="opcao3"&gt;1 - Pudim&lt;/h3&gt;   &lt;h3 ng-show="opcao3"&gt;2 - Sorvete&lt;/h3&gt;&lt;br&gt; &lt;/div&gt; &lt;/body&gt; &lt;/html&gt;</pre>

Fonte: O autor (2022).

Trazendo mais possibilidades de construção de aplicações mesclando HTML com AngularJS, temos o exemplo da tabela 18 que simula um cardápio dinâmico que pode ter suas opções exibidas ou ocultadas a partir de caixas de seleção (checkbox) que estão associadas a diretrizes de exibição de conteúdo.

Assim, é possível utilizar este tipo de elemento HTML para ativar ou desativar eventos implementados em AngularJS de forma interativa com usuários, servindo para inúmeras atividades comuns em aplicações web como na escolha de pedidos em lanchonetes e a adição opcional de ingredientes extras ou a substituição destes, por exemplo.

**TABELA 19 - EXEMPLO DE SCRIPT ANGULARJS**

1	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;</pre>
2	<pre>&lt;script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/ angular.min.js"&gt; &lt;/script&gt;</pre>
3	<pre>&lt;body&gt; &lt;h2&gt;JavaScript - Exemplo 37&lt;/h2&gt; &lt;hr&gt; &lt;script&gt; var app = angular.module("pedidos", []); app.controller("controle", function(\$scope) {     \$scope.produtos = [         {tipo:"Água", preco:"R\$ 2,00"},         {tipo:"Refrigerante", preco:"R\$ 4,00"},         {tipo:"Suco", preco:"R\$ 5,00"}     ];     \$scope.removeltem = function (item) {         \$scope.produtos.splice(item, 1);     } }); &lt;/script&gt;  &lt;h2&gt;Lista de Itens do Pedido&lt;/h2&gt; &lt;div ng-app="pedidos"&gt; &lt;table ng-controller="controle" border=1&gt;     &lt;tr ng-repeat="item in produtos"&gt;         &lt;td&gt;{{item.tipo}}&lt;/td&gt;         &lt;td&gt;{{item.preco}}&lt;/td&gt;         &lt;td align=center&gt;&lt;span ng-click="removeltem(\$index)"&gt;X&lt;/span&gt;&lt;/td&gt;     &lt;/tr&gt;     &lt;tr&gt;     &lt;/table&gt; &lt;/div&gt; &lt;/body&gt; &lt;/html&gt;</pre>

Fonte: O autor (2022).

Como último exemplo de AngularJS, temos uma simulação de lista de produtos dinâmica que são listados em uma tabela que permite que dados sejam excluídos da mesma em tempo real utilizando recursos importantes como trabalho com vetores de objetos e a manipulação dos dados a partir de função associada à interatividade da página.

No exemplo da tabela 19, temos associados ao controlador da aplicação um objeto produtos que recebe os dados da aplicação em forma de vetor a serem exibidos na página de forma organizada em linhas de uma tabela dinâmica.

Uma função para remover itens é implementada de forma que a ação de exclusão da linha da tabela e do item do vetor de objetos esteja associada ao evento de clique associado à diretiva ng-click no X da segunda coluna de cada linha da tabela através do método splice().

Utilizando a diretiva ng-controller como parâmetro da definição da tabela pela tag <table> que faz com que o controlador mantenha atualizada a tabela constantemente, e a cada ação de clique em algum X da tabela, o vetor de objetos seja atualizado e reflita na exibição da tabela na página.

Assim, pudemos ver algumas boas funcionalidades disponibilizadas pelo framework AngularJS que podem aumentar a gama de recursos disponíveis para o desenvolvimento de aplicações web dinâmicas que agregam muito valor a scripts HTML mais estáticos.

## SAIBA MAIS

Após ter tido contato com diferentes formas de implementação de scripts para páginas web, é importante saber que é comum a combinação entre diferentes linguagens e frameworks de forma oferecer maiores facilidades e otimização de trabalho para empresas de desenvolvimento de soluções em TI de forma geral.

Quando se trabalha com bancos de dados, é preciso ter acesso a estes através de aplicações que sejam capazes de se comunicar com servidores e utilizando scripts em linguagem SQL, por exemplo, gerenciar estes para uso em aplicações diversas.

Os estudos para aplicação de tudo que foi estudado podem ser aprofundados e se estender por anos se desejado, pois são praticamente infinitas as possibilidades de utilizações na solução de problemas pela combinação de diferentes linguagens e frameworks baseados nelas.

**Fonte:** O autor (2022).

## REFLITA

A medida que progredimos em nossos estudos, sentimos muitas vezes que não vai ter fim, mas a cada etapa vencida, a cada novo conhecimento obtido, nos tornamos melhores pessoalmente e profissionalmente.

**Fonte:** O autor (2022).

# CONSIDERAÇÕES FINAIS

Após conhecer alguns frameworks muito populares no mercado de desenvolvimento de software para a web, é importante não perder de vista que todo aprendizado deve ocorrer em etapas, e desenvolver-se primeiramente em JavaScript é bastante importante, conhecer HTML é um pré-requisito fundamental.

Partindo dessa ideia, desenvolva-se no uso do HTML e JavaScript primeiramente para simplificar a compreensão e real aprendizado da programação web, para depois estudar métodos alternativos como os propostos por frameworks como os estudados nesta unidade.

Combinar recursos de diferentes frameworks em scripts é uma tarefa comum nas empresas, mas é preciso ter conhecimento da forma como se usa cada framework e saber como combinar os recursos de um framework com a programação pura realizada com JavaScript e HTML, por exemplo.

O importante é manter os estudos e aos poucos evoluir ao ponto de sentir segurança na ideia de desenvolver aplicações própria e poder então atuar no mercado de forma autônoma ou em alguma empresa de desenvolvimento, pois bons profissionais da área de desenvolvimento estão escassos e o mercado aquecido.

## LEITURA COMPLEMENTAR

Além dos frameworks estudados, existem outras alternativas de bibliotecas capazes de contribuir também com o desenvolvimento de aplicações web em JavaScript, HTML e CSS, sendo também relevantes ao mercado de forma geral também.

Um framework interessante se chama Vue.js que assim como React, é voltado para o desenvolvimento de interfaces, tendo bons recursos para o desenvolvimento front-end que permitem desde simples melhoria em conteúdo estático HTML a aplicações de página única (SPA – Single Page Application).

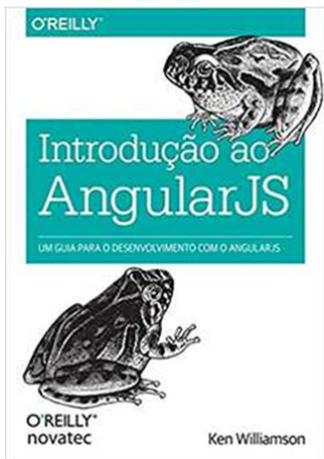
A execução de scripts Vue.js pode ser realizada de duas formas, sendo uma mais simples com a adição do script Vue.js a um outro script HTML diretamente, ou por um processo compilado que unifica todos os componentes HTML, CSS e JavaScript em arquivos do tipo SFC.

Um exemplo bastante simples de inserção de script Vue.js em script HTML é visto a seguir onde é indicado o framework inicialmente com `<script src="https://unpkg.com/vue@3"></script>`, e depois os comandos a serem executados em um script próprio para os comandos.

```
<!DOCTYPE html>
<html>
<script src="https://unpkg.com/vue@3"></script>
<div id="App">{{ message }}</div>
<script>
  const { createApp } = Vue
  createApp({
    data() {
      return {
        message: 'Teste do framework Vue.js'
      }
    }
  }).mount('#App')
</script>
<body>
</body>
</html>
```

**Fonte:** Adaptado de Vue.js (online)

## MATERIAL COMPLEMENTAR



### LIVRO

**Título:** Introdução ao Angular Js.

**Autor:** Ken Williamson

**Editora:** O'Reilly Novatec

**Sinopse:** Livro que tem como objetivo oferecer conhecimentos sobre a criação de aplicações para plataformas desktop ou móveis voltados para backend. Partindo de uma abordagem prática, deve ajudar no aprendizado de programação usando o framework.



### FILME/VÍDEO

**Título:** Silicon Valley

**Ano:** 2014 a 2019.

**Sinopse:** Série de comédia que mostra como uma startup no Vale do Silício tenta criar um produto de sucesso em meio às gigantes do ramo.



### WEB

Exemplificando o trabalho conjunto entre frameworks baseados em JavaScript acesse o site indicado abaixo.

Link de acesso: <https://www.devmedia.com.br/criando-aplicacoes-web-com-angularjs-node-e-concise-css/36777>.

## REFERÊNCIAS

JAVA. Qual é a diferença entre o JavaScript e o Java? Disponível em: [https://www.java.com/pt-BR/download/help/java\\_javascript\\_pt-br.html](https://www.java.com/pt-BR/download/help/java_javascript_pt-br.html). Acesso em: 13 jul. 2022.

JULIO. Criando aplicações web com AngularJS, Node e Concise CSS. DevMedia, 2016. Disponível em <https://www.devmedia.com.br/criando-aplicacoes-web-com-angularjs-node-e-concise-css/36777>. Acesso em: 20 ago. 2022.

Node.js File System Module. W3Schools, [s.d.]f. Disponível em [https://www.w3schools.com/nodejs/nodejs\\_filesystem.asp](https://www.w3schools.com/nodejs/nodejs_filesystem.asp). Acesso em: 20 ago. 2022.

Node.js Introduction. W3Schools, [s.d.]a. Disponível em [https://www.w3schools.com/nodejs/nodejs\\_intro.asp](https://www.w3schools.com/nodejs/nodejs_intro.asp). Acesso em: 18 ago. 2022.

Node.js Modules. W3Schools, [s.d.]b. Disponível em [https://www.w3schools.com/nodejs/nodejs\\_modules.asp](https://www.w3schools.com/nodejs/nodejs_modules.asp). Acesso em: 18 ago. 2022.

Quick Start. Vue.js, [s.d.]. Disponível em <https://vuejs.org/guide/quick-start.html>. Acesso em: 22 ago. 2022.

React Conditional Rendering. W3Schools, [s.d.]e. Disponível em [https://www.w3schools.com/react/react\\_conditional\\_rendering.asp](https://www.w3schools.com/react/react_conditional_rendering.asp). Acesso em: 20 ago. 2022.

React Forms. W3Schools, [s.d.]d. Disponível em [https://www.w3schools.com/react/react\\_forms.asp](https://www.w3schools.com/react/react_forms.asp). Acesso em: 20 ago. 2022.

React Introduction. W3Schools, [s.d.]c. Disponível em [https://www.w3schools.com/react/react\\_getstarted.asp](https://www.w3schools.com/react/react_getstarted.asp). Acesso em: 18 ago. 2022.

SILVA, Maurício Samy. JavaScript: guia do programador. São Paulo: Novatec Editora, 2010.

W3Schools. JavaScript Date Objects. Disponível em: [https://www.w3schools.com/js/js\\_dates.asp](https://www.w3schools.com/js/js_dates.asp). Acesso em: 15 ago. 2022.

W3Schools. JavaScript HTML DOM - Changing HTML. Disponível em: [https://www.w3schools.com/js/js\\_htmldom\\_html.asp](https://www.w3schools.com/js/js_htmldom_html.asp). Acesso em: 17 ago. 2022.

W3Schools. JavaScript Math Object. Disponível em: [https://www.w3schools.com/js/js\\_math.asp](https://www.w3schools.com/js/js_math.asp). Acesso em: 10 ago. 2022.

W3Schools. JavaScript Operators. Disponível em: [https://www.w3schools.com/js/js\\_operators.asp](https://www.w3schools.com/js/js_operators.asp). Acesso em: 15 jul. 2022.

W3Schools. JavaScript Reference. Disponível em: [https://www.w3schools.com/jsref/jsref\\_reference.asp](https://www.w3schools.com/jsref/jsref_reference.asp). Acesso em: 15 jul. 2022.

# CONCLUSÃO GERAL

Após ter tido contato com uma das mais importantes e populares linguagens de programação do mercado, é natural a ansiedade por tentar vagas no mercado e já iniciar o quanto antes, uma bela carreira profissional, mas lembre-se que o mercado está aquecido, mas necessita principalmente de profissionais qualificados.

Uma boa alternativa é praticar aquilo que foi estudado para assimilar ao máximo a base da programação JavaScript e conhecer outras funcionalidades que a linguagem oferece para complementar seus estudos.

Crie um portfólio de aplicações pequenas que mostrem que é capaz de produzir software e a medida que avança nos estudos, agregue novos recursos e aos poucos introduza frameworks como mecanismos auxiliares que o levem para mais perto do que é praticado no mercado.

Depois de algum tempo de treino, será capaz de mostrar realmente do que pode ser capaz e ai sim, buscar vagas no mercado amplo e com oportunidades em todo o mundo que oferecem vagas remotas inclusive.

Apenas ler e não gerar real conhecimento não lhe dará meios para disputar vagas no mercado com reais chances de uma boa colocação no mercado, e por isto, deixe a ansiedade de lado e continue com seus estudos!



+55 (44) 3045 9898  
Rua Getúlio Vargas, 333 - Centro  
CEP 87.702-200 - Paranavaí - PR  
[www.unifatecie.edu.br](http://www.unifatecie.edu.br)

