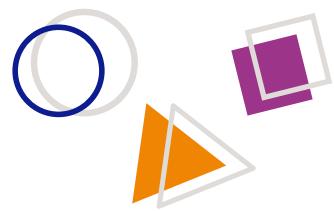


Mundo Tech



■ ■ ■

PROGRAMAÇÃO BACK-END FUNDAMENTOS DA LINGUAGEM JAVASCRIPT

SUMÁRIO

Linguagem JavaScript	3
Fundamentos da linguagem JavaScript	3
Configuração do ambiente de desenvolvimento utilizando a plataforma Codesandbox	4
Introdução à linguagem JavaScript.....	7
Tipos de dados.....	15
Operadores aritméticos, relacionais e lógicos.....	16
Operadores aritméticos	16
Operadores relacionais	17
Operadores lógicos	18
Expressões lógicas e aritméticas	18
Referências.....	20



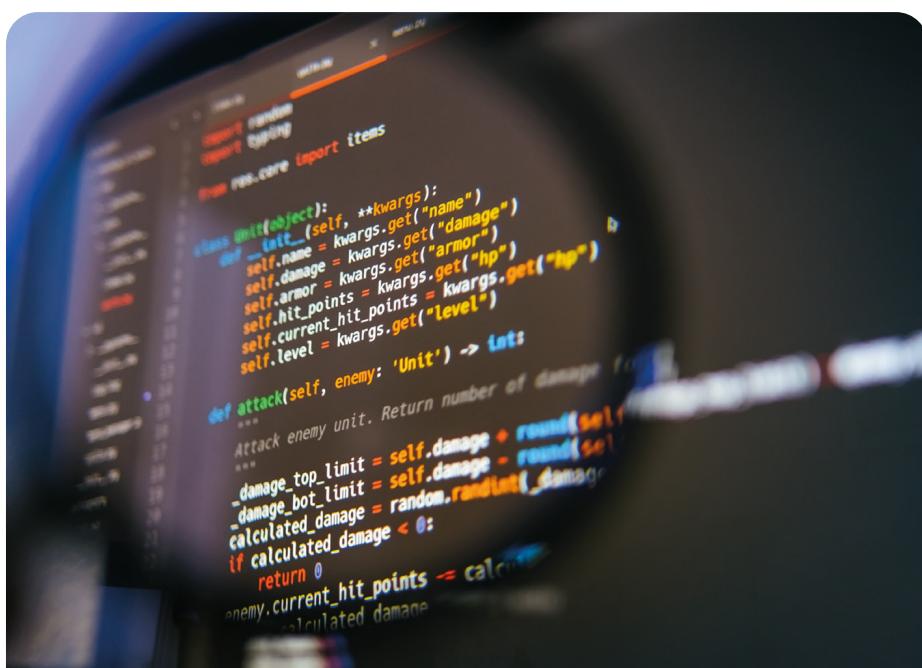
LINGUAGEM JAVASCRIPT

O JavaScript é uma linguagem de programação de script que é usada em páginas da web junto com as linguagens de marcação HTML (Linguagem de Marcação de HiperTexto) e CSS (Folhas de Estilo em Cascata). O JavaScript é muito popular e adotado universalmente por todos os navegadores da web por seu suporte, o que permite que o conteúdo dinâmico seja executado em uma página da web. É por isso que o uso de JavaScript no desenvolvimento web está aumentando, tanto no *back-end*, quanto no *front-end*, pois facilita bastante o procedimento de desenvolvimento.

Assim, neste estudo será apresentada a configuração do ambiente de desenvolvimento de projetos utilizando a plataforma CodeSandbox. Você terá uma introdução à linguagem JavaScript para que possa compreender os fundamentos básicos dessa linguagem. Durante seus estudos, você terá compreensão dos tipos de dados existentes, como números, cadeias de caracteres, símbolos e booleanos. Serão apresentados os operadores aritméticos, relacionais e lógicos. E, por fim, você terá dicas de como montar expressões lógicas e aritméticas. Então, este é o momento de você avançar seus estudos!

Fundamentos da linguagem JavaScript

O JavaScript é uma das linguagens de programação mais populares do mundo. É tão popular que tudo o que for lançado para a web vai ter algum tipo de integração com JavaScript em algum momento. O JavaScript é utilizado, principalmente, para criar websites, aplicações web e aplicações do lado do servidor usando a plataforma Node.js. Mas, o JavaScript não está limitado a esses fins. Ele também pode ser usado para criar aplicações móveis usando ferramentas como o React Native.



JavaScript é uma linguagem de programação que traz diversas características, por exemplo (MORRISON, 2008; FLANAGAN, 2013; HAVERBEKE, 2018; CAELUM, 2022):

- Alto nível: fornece abstrações que permitem ignorar os detalhes da máquina onde ela está funcionando. Essa linguagem gerencia a memória automaticamente para que você possa se concentrar no código.
- Tipagem dinâmica: variáveis fracamente tipadas são um tipo de variável que não impõe um tipo. Assim, você pode reatribuir qualquer tipo a uma variável, por exemplo, atribuindo um inteiro a uma variável que contenha uma cadeia de caracteres.
- Interpretada: isso significa que o JavaScript não precisa de uma etapa de compilação antes que um programa possa ser executado, ao contrário do Java, por exemplo. Na prática, os navegadores compilam JavaScript antes de executá-lo, por razões de desempenho, mas não há nenhuma etapa adicional envolvida.

Às vezes é difícil separar o JavaScript das características do ambiente em que ele é usado. Por exemplo, o comando “console.log()” que você pode encontrar em muitos exemplos de código não é JavaScript. Ao invés disso, faz parte da vasta biblioteca de APIs fornecida no navegador. Da mesma forma, no servidor, às vezes, pode ser difícil separar as características da linguagem JavaScript dos recursos fornecidos pelo Node.js.

Configuração do ambiente de desenvolvimento utilizando a plataforma Codesandbox

O IDE (Ambiente de Desenvolvimento Integrado) deste estudo será a plataforma CodeSandbox (<https://codesandbox.io/>), um ambiente de desenvolvimento integrado on-line, fácil de usar e compartilhável que está em alta com os desenvolvedores de JavaScript e Node.js.

Os IDEs online aproveitaram os recursos de ferramentas baseadas em nuvem, crescendo em poder nos últimos anos. CodeSandbox é uma das opções mais populares nesse espaço, e seu uso vem aumentando recentemente. O CodeSandbox está ganhando popularidade na codificação social por sua facilidade de uso, suporte tecnológico simplificado e estrutura de custo razoável. Além disso, CodeSandbox é um projeto de código aberto.

Você pode utilizar sua conta do GitHub ou do Google para acessar esta plataforma de forma gratuita. Assim, não há necessidade de instalar nada em seu computador. Tudo pode ser feito diretamente do navegador. Isso vai facilitar todo o seu trabalho! Após realizar a autenticação na plataforma, vá até a opção “Home”, depois selecione “New Sandbox” e, posteriormente, “Create Sandbox” e nas opções sugeridas escolha o ambiente pré-definido “static”, conforme apresentado na imagem seguinte.

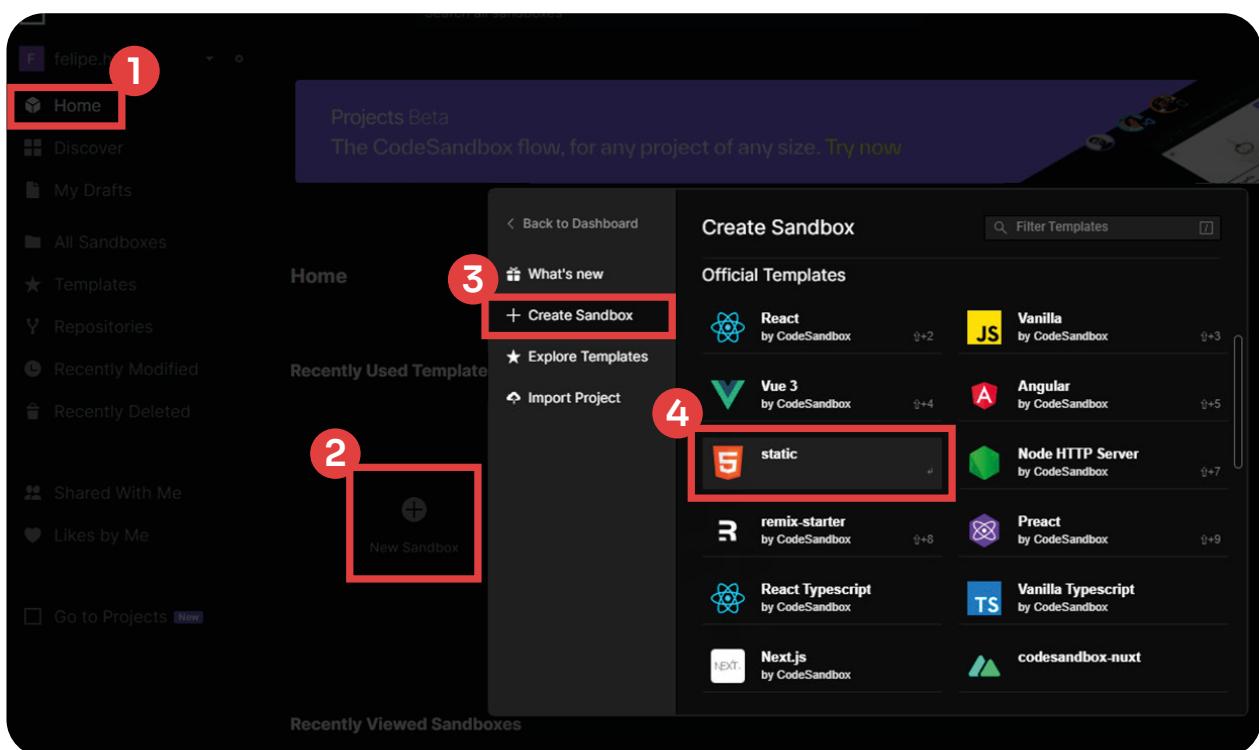
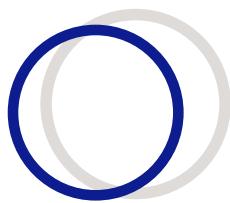


Figura 1 - Configuração de um ambiente na plataforma CodeSandbox

Fonte: elaborada pelo Autor (2022)





O CodeSandbox é mais conhecido como um ambiente para ativar e compartilhar rapidamente aplicativos JavaScript de *front-end*. Isso soa semelhante ao CodePen, visto em estudos anteriores, mas na verdade o CodeSandbox é um projeto mais ambicioso, com suporte full-stack quase comparável a um IDE online completo, embora apenas para JavaScript.

Confira na imagem seguinte um exemplo de um projeto criado utilizando a plataforma CodeSandbox. Nela, é possível ver claramente três áreas principais, que serão explicadas na sequência.



Figura 2 - Plataforma CodeSandbox

Fonte: elaborada pelo Autor (2022)

A primeira exibe os arquivos existentes no projeto. A segunda área apresenta o arquivo aberto atualmente. A terceira exibe, em tempo real, o resultado da renderização do arquivo em HTML no browser. Observe que bem na parte inferior da aba que renderiza o conteúdo do browser há uma outra área chamada “Console”. Nesta aba podemos ver os resultados exibidos utilizando o método “console.log()”. Este recurso será explicado posteriormente.



Não se esqueça de que todas as alterações realizadas nos arquivos criados na plataforma CodeSandbox precisam ser salvas antes, utilizando o comando Ctrl + S. Caso contrário, as mudanças não serão refletidas.

Introdução à linguagem JavaScript

Para iniciar seus estudos, você precisa conhecer alguns fundamentos básicos do JavaScript. Esses fundamentos estão relacionados às formas de inserção de códigos desenvolvidos em JavaScript em arquivos HTML, uso do método “`console.log()`”, utilização de janelas de diálogo, formas de inserção de comentários em trechos de código e conceitos de variáveis.

a) Formas de inclusão do JavaScript no HTML

Ao trabalhar com arquivos para a web, o JavaScript precisa ser carregado e executado junto com a marcação HTML. Isso pode ser feito online dentro de um documento HTML ou em um arquivo separado que o navegador baixará junto com o documento HTML.

Você pode adicionar código JavaScript em um documento HTML empregando a tag HTML dedicada `<script>` que envolve o código JavaScript. A tag `<script>` pode ser colocada na seção `<head>` do seu HTML ou na seção `<body>`, dependendo de quando você deseja que o JavaScript seja carregado. Confira um exemplo:



```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <title>Exemplo</title>
    <script>
      alert("Alô mundo!");
    </script>
  </head>
  <body></body>
</html>
```

Para acomodar scripts maiores ou scripts que serão usados em várias páginas, o código JavaScript geralmente reside em um ou mais arquivos ".js" que são referenciados em documentos HTML, de forma semelhante como ativos externos, como CSS, são referenciados. Confira o arquivo desenvolvido em HTML:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <title>Exemplo</title>
    <script src="script.js"></script>
  </head>
  <body></body>
</html>
<body></body>
</html>
```

Agora, confira o arquivo desenvolvido em JavaScript:

```
...  
...  
...  
  
alert("Alô mundo!");
```

Assim, você terá dois arquivos: um contendo código HTML com extensão “.html”, e outro contendo código JavaScript com extensão “.js.”. Haverá ainda uma tag HTML chamada <script> que “unirá” os arquivos.

Dica

O JavaScript é sensível a maiúsculas e minúsculas (*case sensitive*). Logo, uma variável chamada de “teste” é diferente de “Teste”. O mesmo vale para qualquer identificador. Um identificador é uma sequência de caracteres que pode ser usada para identificar uma variável, uma função ou um objeto. Ele pode começar com uma letra, um cífrão “\$” ou um sublinhado “_”, e pode conter dígitos. Alguns nomes são reservados para uso interno em JavaScript, e não podemos usá-los como identificadores.

b) Método “console.log()”

O método “console.log()” é usado para exibir mensagens ao usuário. Para isso, basta criar um arquivo JavaScript na plataforma CodeSandbox separado ou vinculá-lo ao arquivo HTML. Não se esqueça de que para ver os resultados deve-se utilizar a aba “Console” da plataforma. Lembre-se: todas as alterações precisam ser salvas utilizando o comando Ctrl + S. Confira a imagem seguinte que apresenta o resultado dessa impressão:

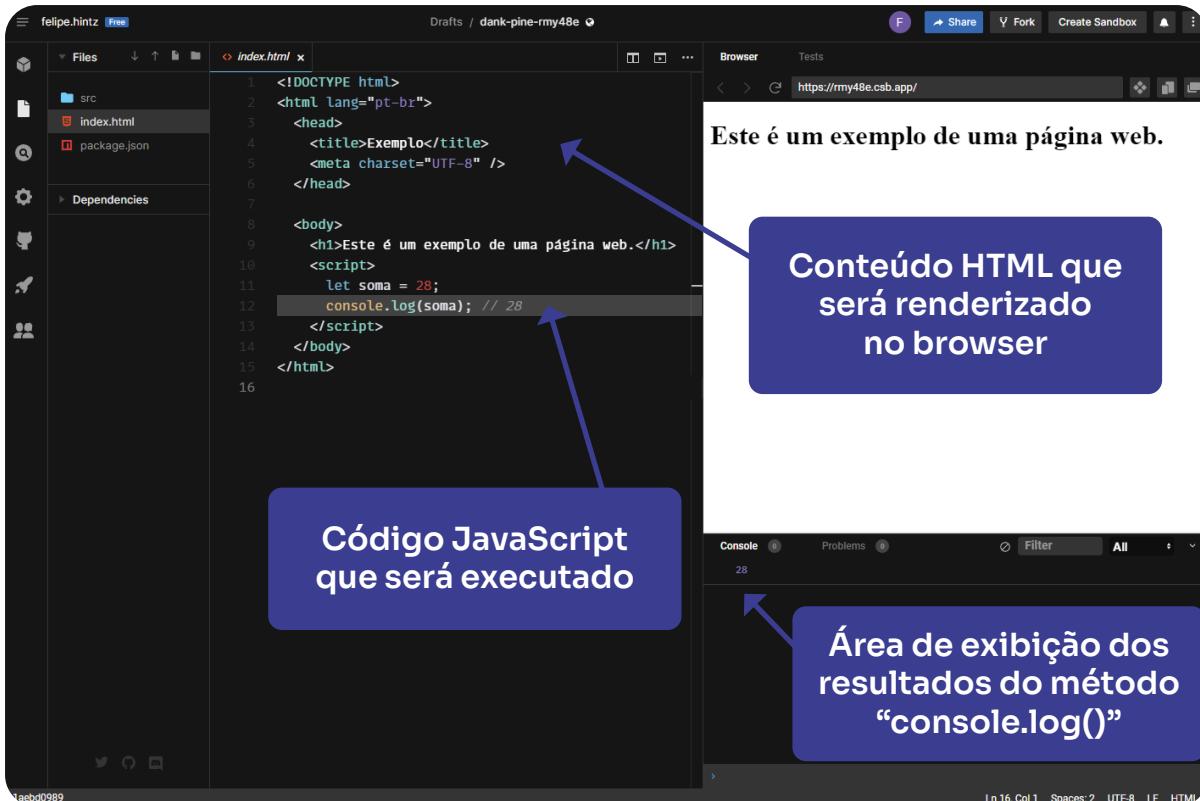


Figura 3 - Acesso à aba “Console” do CondeSandbox
Fonte: elaborada pelo Autor (2022)

Quando você executa o código acima, o valor 28 é impresso no console. Confira outro exemplo que imprime uma cadeia de caracteres.

```

// Programa para imprimir uma frase
// Passando a string
let ano = 2022;
console.log("Eu estou aprendendo JavaScript em" + ano);

```

A saída ao executar o código acima será “Eu estou aprendendo JavaScript em 2022”.

Como você pode ver nesses exemplos, ao utilizar “console.log()” fica mais fácil ver o valor dentro de uma variável. É por isso que é comumente usado para testar/depurar um código.

c) Janelas de diálogo

Existem principalmente três categorias de caixas de diálogo em JavaScript. Esses tipos são utilizados para exibir mensagens de confirmação, exibir um erro ou exibir uma mensagem de aviso ou notificação. Nesta seção serão explicadas as seguintes caixas de diálogo: “alert”, “prompt” e “confirm”.

Caixas do tipo “alert” são usadas principalmente para validação. Ele exibe uma mensagem na caixa de diálogo. Esta caixa de diálogo bloqueará o navegador. Você não pode fazer nada na página do navegador sem pressionar o botão “ok” desta caixa de diálogo, então ela é fechada. É usado para exibir mensagens de erro (MORRISON, 2008). Confira um exemplo:

```
● ● ●  
alert( "Olá, estou aprendendo JavaScript!" );
```

As janelas do tipo “confirm” permitem exibir uma janela *pop-up* com dois botões: “ok” e “cancelar”. Funciona como uma função: se o usuário clicar em “ok”, ela retorna “true”; em “cancelar” retorna false. Confira um exemplo:

```
● ● ●  
var valor = confirm("Você come muitos alimentos gordurosos?");  
if (valor == true) {  
    alert("Procure um nutricionista.");  
} else {  
    alert("Você está no caminho certo.");  
}
```

Já as janelas do tipo “prompt” permitem exibir uma janela *pop-up* com dois botões (“ok” e “cancelar”) e uma caixa de texto. Funciona como uma função: se o usuário clicar em “ok” e preencher a caixa de texto, ela retorna o valor do texto; em “cancelar” retorna “null” para o campo que deveria ser coletado. O segundo parâmetro pode ser preenchido como uma sugestão de preenchimento. Confira um exemplo:



```
var email = prompt("Digite seu e-mail", "");  
alert("O e-mail" + email + " foi coletado com sucesso.");
```

O que é lido da janela console é uma “string”. Pode-se converter “string” para inteiro utilizando as funções pré-definidas “parseInt” e “parseFloat”.

- › parseInt(valor, base): converte uma “string” para inteiro. O valor será convertido para inteiro e base é o número da base (base 10).
- › parseFloat(valor): converte uma “string” para um valor real/ponto flutuante.

```
var prova = parseFloat(prompt("Qual é a nota da prova?", ""));  
var trabalho = parseFloat(prompt("Qual é a nota do trabalho?", ""));  
  
nota = prova + trabalho;  
  
alert("Sua nota é: " + nota);
```

d) Inclusão de comentários

Os comentários são uma das partes mais importantes de qualquer programa e em qualquer linguagem de programação. Eles são importantes porque nos permitem anotar o código e acrescentar informações importantes que de outra forma não estariam disponíveis para outras pessoas (ou para nós mesmos) lendo o código.

Em JavaScript, podemos escrever um comentário em uma única linha usando “//”. Tudo após “//” não é considerado como código pelo intérprete do JavaScript. Assim:

```
// Aqui vem um comentário de uma linha...  
teste = 89; // Aqui vem outro comentário
```

Outro tipo de comentário é um comentário com várias linhas. Ele começa com “`/*`” e termina com “`*/`”. Tudo no meio não é considerado como código:

```
/*
Aqui vem um comentário de várias linhas...
*/
```

e) Variáveis

Uma variável é um valor atribuído a um identificador, para que você possa referenciá-lo e utilizá-lo mais tarde no programa. Uma variável deve ser declarada antes que você possa utilizá-la. Temos duas formas principais de declarar variáveis. A primeira é usar “`const`”:

```
const valor = 85;
```

A segunda maneira é usar “`let`”:

```
let valor = 85;
```

O tipo “`const`” define uma referência constante a um valor. Isso significa que a referência não pode ser alterada. Não se pode reatribuir um novo valor a ela. Usando “`let`” você pode atribuir um novo valor a ele.

Por exemplo, você não pode fazer isso:

```
const valor = 85;
valor = 34;
```

Porque você receberá um erro de atribuição à variável constante. Por outro lado, você pode fazer isso usando “let”:

```
let valor = 85;  
valor = 34;
```

As variáveis do tipo “const” devem ser inicializadas no momento da declaração, mas as variáveis do tipo “let” podem ser inicializadas mais tarde.

```
const ano = 1989;  
let taxa;  
taxa = 5.5;
```

Não se esqueça de que não se pode redeclarar a mesma variável mais de uma vez, caso contrário você receberá um erro de declaração duplicada.



Atenção

Até 2015, o tipo “var” era a única maneira de declarar uma variável em JavaScript. Hoje, uma base de código moderna muito provavelmente usará apenas “const” e “let”. Caso queira saber mais informações sobre a diferença entre estes usos, acesse o seguinte site:

<https://dev.to/ananopaisdojavascript/var-let-e-const-qual-e-a-diferenca-473o>



Tipos de dados

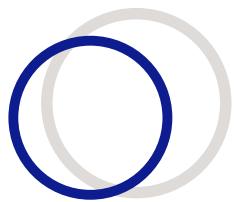
As variáveis no JavaScript não têm nenhum tipo anexado. Elas não são tipadas. Uma vez que você atribui um valor com algum tipo a uma variável, você pode posteriormente reatribuir a variável para receber um valor de qualquer outro tipo, sem qualquer problema.

No JavaScript temos dois tipos principais de tipos: tipos primitivos e tipos de objetos. Os tipos primitivos são: números (number), cadeias de caracteres (string), símbolos (symbol), e lógicos (boolean). Além disso, o JavaScript conta com dois tipos especiais: nulo (null) e indefinido (undefined).

Vamos passar por uma breve visão geral de cada tipo de dados e seus possíveis usos (MORRISON, 2008; FLANAGAN, 2013; HAVERBEKE, 2018; CAELUM, 2022).

- Number representa a categoria de números, incluindo inteiros e decimais. Os computadores costumam usar números para realizar operações matemáticas.
- String representa coleções de caracteres alfanuméricos e símbolos. É assim que vamos armazenar letras e palavras. Coisas como endereços.
- Symbol é um tipo de dado muito peculiar. Depois de criar um símbolo, seu valor é mantido privado e para uso interno. Símbolos são frequentemente usados para identificar propriedades de objetos. Muitas vezes para evitar conflitos de nomes entre propriedades, pois nenhum símbolo é igual a outro. Ou adicionar propriedades que o usuário não pode substituir, intencionalmente ou sem perceber.
- Boolean só pode ter dois valores lógicos: verdadeiro ou falso ("true" ou "false"). Eles representam todos os dados que possuem apenas dois estados, como um interruptor de luz: ligado ou desligado.
- Null é semelhante a undefined, exceto porque deve ser definido intencionalmente. Também significa vazio ou nada, mas é assim porque um desenvolvedor determinou.
- Undefined significa que a variável foi criada, mas nunca recebeu um valor. Está assim porque o desenvolvedor jamais se preocupou em dizer qual valor deveria ter.

Qualquer valor que não seja de um tipo primitivo é um objeto. Os tipos de objetos têm propriedades e também têm métodos que podem agir sobre essas propriedades.



Operadores aritméticos, relacionais e lógicos

Operadores são símbolos que dizem ao compilador para realizar alguns cálculos matemáticos ou operações lógicas nas variáveis e constantes. A linguagem de programação JavaScript possui grande variedade de operadores que são categorizados em diferentes grupos com base no tipo de operações que realizam. Confira agora os operadores aritméticos, relacionais e lógicos.

Operadores aritméticos

Você já pôde notar que operadores são símbolos usados para a construção de expressões. Eles trabalham em conjunto com as constantes e variáveis. Em JavaScript, temos os seguintes operadores aritméticos:

Operador	Símbolo	Exemplo
Soma	+	<pre>const operando1 = 34; const operando2 = 32; console.log(operando1 + operando2); // Retorna 66</pre>
Subtração	-	<pre>const operando1 = 56; const operando2 = 45; console.log(operando1 - operando2); // Retorna 11</pre>
Multiplicação	*	<pre>const operando1 = 85; const operando2 = 41; console.log(operando1 * operando2); // Retorna 44</pre>
Divisão	/	<pre>const operando1 = 45; const operando2 = 15; console.log(operando1 / operando2); // Retorna 3</pre>
Resto da divisão	%	<pre>const operando1 = 20; const operando2 = 10; console.log(operando1 % operando2); // Retorna 2</pre>

Há um outro operador muito utilizado nas linguagens de programação: o operador de atribuição representado pelo símbolo “=” . Ele simplesmente atribui um valor a uma constante ou variável. Esse valor pode ser um valor inteiro, real, caractere, vazio ou nulo.

Operadores relacionais

Ainda existem os operadores relacionais, que são utilizados para realizar comparações de valores. Os operadores de comparação sempre retornam um valor booleano, ou seja, um valor que é verdadeiro ou falso. Acompanhe no quadro a seguir a relação de operadores relacionais na linguagem JavaScript:

Operador	Símbolo
Igualdade sem considerar o tipo	<code>==</code>
Igualdade considerando o tipo	<code>===</code>
Diferente	<code>!=</code>
Maior	<code>></code>
Menor	<code><</code>
Maior ou igual	<code>>=</code>
Menor ou igual	<code><=</code>

Confira um exemplo de aplicação e análise dos operadores apresentados anteriormente:

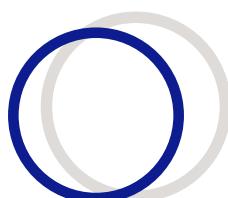
```
let valorA = 5, valorB = 10, valorC = "5";  
let valorX = valorA;  
console.log(valorA == valorC); // Retorna true  
console.log(valorA === valorC); // Retorna false  
console.log(valorA == valorX); // Retorna true  
console.log(valorA != valorB); // Retorna true  
console.log(valorA > valorB); // Retorna false  
console.log(valorA < valorB); // Retorna true  
console.log(valorA >= valorB); // Retorna false  
console.log(valorA <= valorB); // Retorna true
```

Operadores lógicos

Por fim, temos os operadores lógicos. Os operadores lógicos são usados para representar situações em que os operadores aritméticos não são suficientes. A união de operações simples por meio dos operadores lógicos resolve os cenários em que se deve comparar duas ou mais expressões simples. Os operadores lógicos são os seguintes:

Operador	Símbolo
E	&&
Ou	
Não	!

```
var valorA = 5, valorB = 10;  
console.log(valorA != valorB && valorA < valorB); // Retorna  
true  
  
console.log(valorA > valorB || valorA == valorB); // Retorna  
false  
  
console.log(valorA < valorB || valorA == valorB); // Retorna  
true  
  
console.log(!(valorA < valorB)); // Retorna false  
console.log(!(valorA > valorB)); // Retorna true
```



Expressões lógicas e aritméticas

Uma expressão é uma única unidade de código JavaScript que o motor JavaScript pode avaliar, e devolver um valor. As expressões podem variar em complexidade. Partimos das muito simples, chamadas expressões primárias:

```
20;  
0.05;  
"Algum texto;  
true;  
false;  
valor; // Onde valor é uma variável ou uma constante
```

Expressões aritméticas são expressões que tomam uma variável e um operador e resultam em um número:

```
20 / 20;  
contador++;  
valor * 2;
```

Expressões de *string* são expressões que resultam em uma cadeia de caracteres:

```
'Estou aprendendo' + 'JavaScript'; // Estou aprendendo JavaScript
```

Expressões lógicas fazem uso de operadores lógicos e resolvem a um valor booleano:

```
valorA && valorB; // O símbolo && representa o operador E  
valorA || valorB; // O símbolo || representa o operador Ou  
!valorA // O símbolo ! representa o operação Não
```

Cada linha em um programa JavaScript é opcionalmente terminada usando ponto e vírgula. Assim, o interpretador JavaScript é inteligente o suficiente para introduzir ponto e vírgula para você. Na maioria dos casos, você pode omitir os pontos e vírgulas completamente de seus programas.

Durante seus estudos você pôde perceber que o JavaScript é uma das linguagens mais fáceis e eficazes da web. É a linguagem mais favorita, popular e admirada dos desenvolvedores da web em todo o mundo! JavaScript tem a capacidade de criar uma das páginas da web mais dinâmicas e responsivas e pode proporcionar uma experiência de usuário magnífica para os internautas!

Como complemento do seu aprendizado, você conheceu os procedimentos necessários para realizar a configuração do ambiente de desenvolvimento utilizando a plataforma CodeSandbox. Assim, seus projetos puderam ser criados de forma mais prática e rápida. Você teve contato com os fundamentos básicos da linguagem JavaScript, permitindo que as execuções de diversos exemplos pudessem auxiliar na sua compreensão. Você compreendeu a importância e a correta utilização dos tipos de dados. E, por fim, você conheceu os operadores aritméticos, relacionais e lógicos, bem como aprendeu a montar expressões lógicas e aritméticas utilizando estes operadores. Esta etapa se encerra por aqui. Mas não se preocupe, ainda existem diversos tópicos para serem explorados! Avance em seus estudos!

REFERÊNCIAS

DESENVOLVIMENTO web com HTML, CSS e JavaScript: curso WD-43. **Caelum**, [s.d.]. Disponível em: <https://www.caelum.com.br/download/caelum-html-css-javascript.pdf>. Acesso em: 15 jul. 2022.

FLANAGAN, D. **JavaScript**: o guia definitivo. 6. ed. Porto Alegre: Bookman, 2013.

HAVERBEKE, M. **Eloquent JavaScript**. 3. ed. San Francisco. 2018.

MORRISON, M. **Use a Cabeça! JavaScript**. Rio de Janeiro: Alta Books, 2008.



