



`<CODE>`

JavaScript

Material Complementar

Ricardo Alexandre Bontempo

Formado em Análise de Sistemas, Mestre em Administração, Comunicação e Educação.

É Analista de Sistemas no desenvolvido de Front-End e Back-End | Grupo Dass, é professor na Gama Academy, também atua como Professor Universitário no Senac de Osasco em Sistemas e na Universidade Metodista em SBC.



[Ricardo Alexandre Bontempo | LinkedIn](#)



#PraCegoVer: Fotografia do autor Ricardo Alexandre Bontempo.



Índice

- + Introdução ao JavaScript
- + O que é linguagem de programação
- + Porque aprender JavaScript?
- + Frameworks e Libraries
- + Instalação e Extensões do VSCode
- + Introdução aos Protocolos
- + O que é DOM
- + Elementos e Eventos
- + Usando o getElementById
- + Usando o getElementsByTagName

- + Usando o getElementByClassName
- + Usando o queryselector
- + Usando o CreateElement e InsertBefore
- + Usando o AppendChild
- + Usando o ParentNode
- + Usando o get e setAttribute
- + Alterando o estilo CSS com JS
- + Pegando todos os determinados elementos
- + Criando e removendo elementos
- + Eventos de Clique



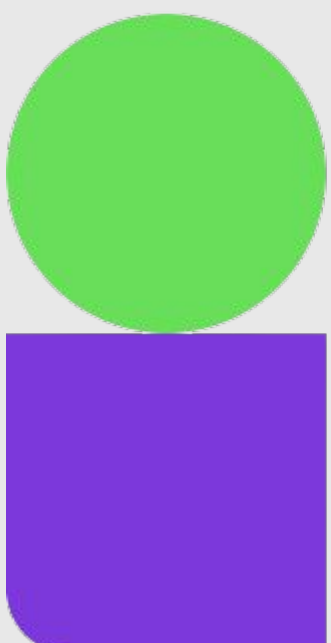
Índice

- + Capturando movimentos do Mouse
- + Capturando Eventos do scroll
- + Definição de Estrutura de dados
- + Instalando o Node com NVM
- + Tipos de dado em JavaScript
- + Estrutura de Dados
- + JavaScript Boolean
- + JavaScript Operadores Lógicos
- + JavaScript Number
- + JavaScript String

- + JavaScript Operadores Lógicos
- + JavaScript Operadores Relacionais
- + JavaScript Operadores de comparação
- + Exemplo de Operadores de comparação
- + Estrutura de programação: Variáveis var
- + Estrutura de programação: Variáveis left
- + Estrutura de programação: Variáveis const
- + Palavras reservadas do JavaScript
- + Declarando Funções em JavaScript
- + Estrutura de repetição JavaScript

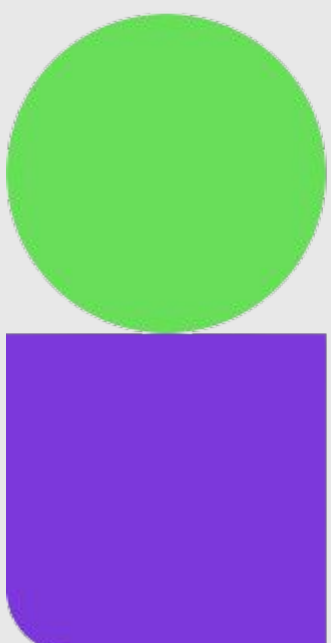


Índice

- + Estrutura de repetição For JavaScript
 - + Estrutura de repetição While JavaScript
 - + Estrutura condicionais JavaScript
 - + Estrutura condicional if JavaScript
 - + Objetos e Arrays no JavaScript
 - + Como Acessar os elementos de um Array JavaScript
 - + Como acessar um Array completo Em JavaScript
 - + Como trabalhar com os objetos Array Em JavaScript
 - + Analisar as propriedades de um Array JS
 - + Utilizando a propriedade length Array JS
 - + Interagindo com um Array JS
 - + Inserindo registro em um Array JS
 - + Como fazer o reconhecimento de uma variável do tipo Array Em JavaScript
 - + Como criar uma função para reconhecer um variável do tipo Array Em J
 - + Criando uma Desestruturação de Array Em JavaScript
 - + O que é JSON?
- 



Índice

- + Vantagens do JSON
 - + Fechamento
 - + Referência Bibliográfica
- 

Introdução ao JavaScript

O JavaScript (frequentemente abreviado como JS) é uma linguagem de programação interpretada e estruturada que permite a implementação de itens complexos em páginas web. Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da World Wide Web.

As três principais tecnologias da World Wide Web.

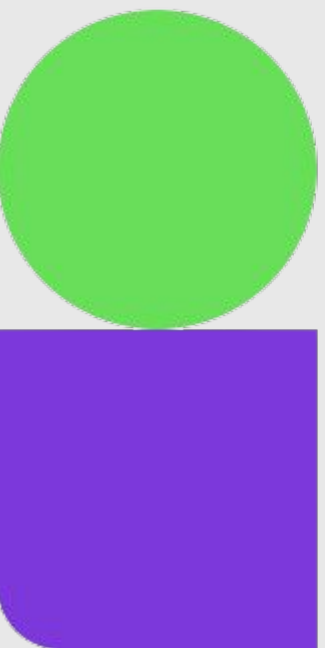


Fonte: developer.mozilla.org

#PraCegoVer: Imagem ilustrando as camadas de um bolo para representar o níveis de conhecimento das principais tecnologias.

O que é Linguagem de Programação?

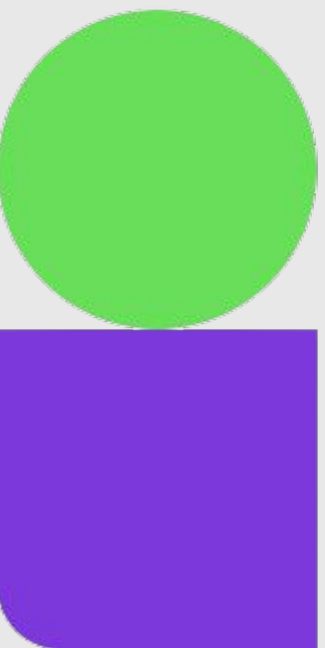
A Linguagem de Programação é uma linguagem escrita e formal é um método padronizado, formado por um conjunto instruções e regras sintáticas e semânticas, de implementação de um código fonte para gerar programas (software).



Porque aprender JavaScript?

Importância do JavaScript:

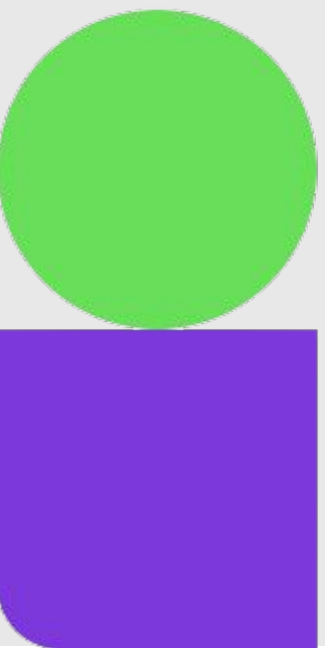
- Tem suporte em todos os navegadores;
- Scripts JS e plugins são utilizados em todos os lugares e por todo mundo;
- Possui muitos recursos;
- E especialistas que saibam programar em JS são sempre requisitados;



Frameworks e Libraries

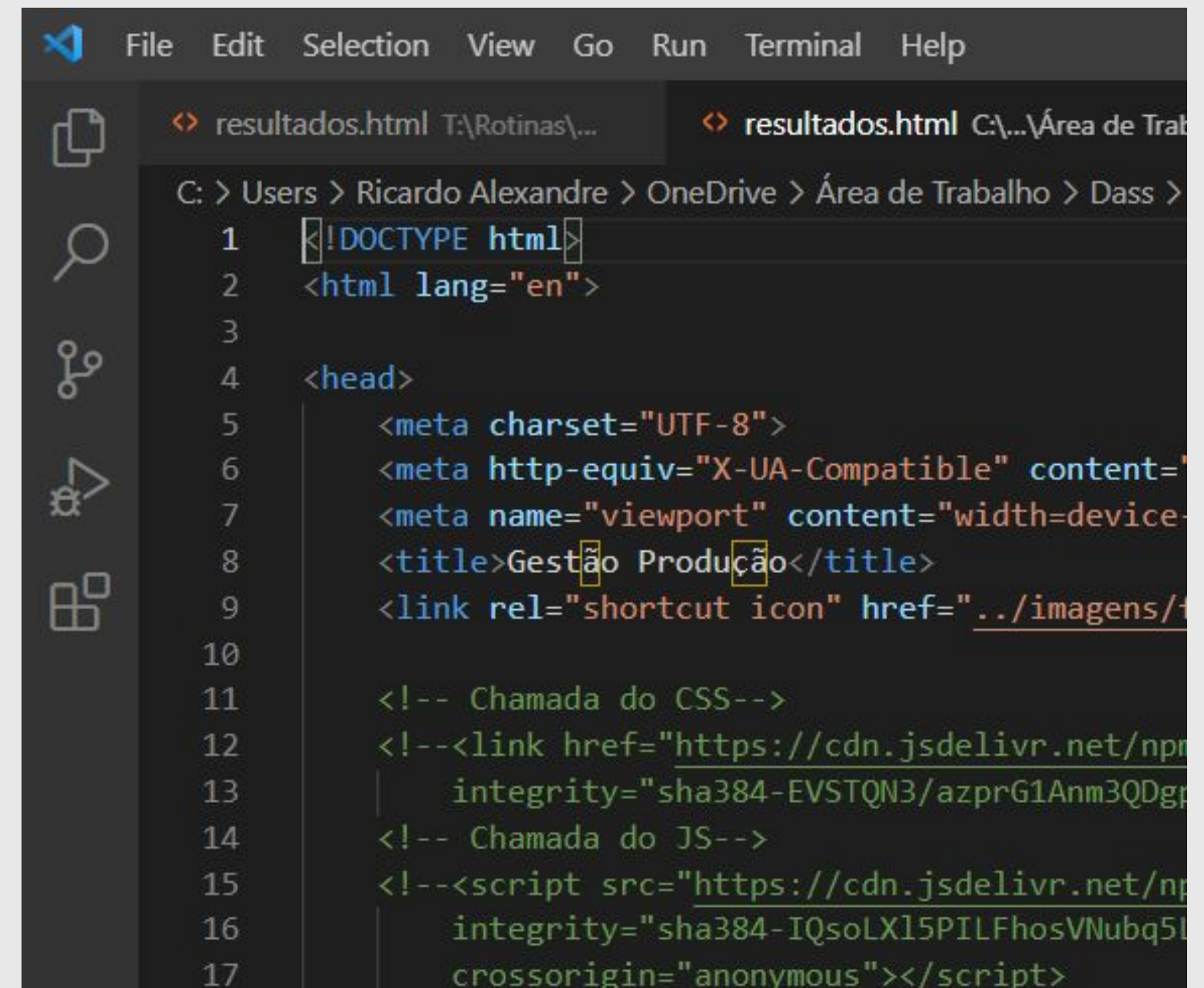
Um framework é um conjunto de biblioteca captura a funcionalidade comum a várias aplicações, oferecendo não apenas funcionalidades, mas também uma arquitetura para o trabalho de desenvolvimento.

As libraries, traduzindo bibliotecas permite você criar processos, eventos que você pode utilizá-las em diferentes projetos.



Instalação e Extensões VSCode

O VS Code - Visual Studio Code é um editor de código de código aberto desenvolvido pela Microsoft. Permite a criação de softwares Desktop com HTML, CSS, JavaScript entre outros.



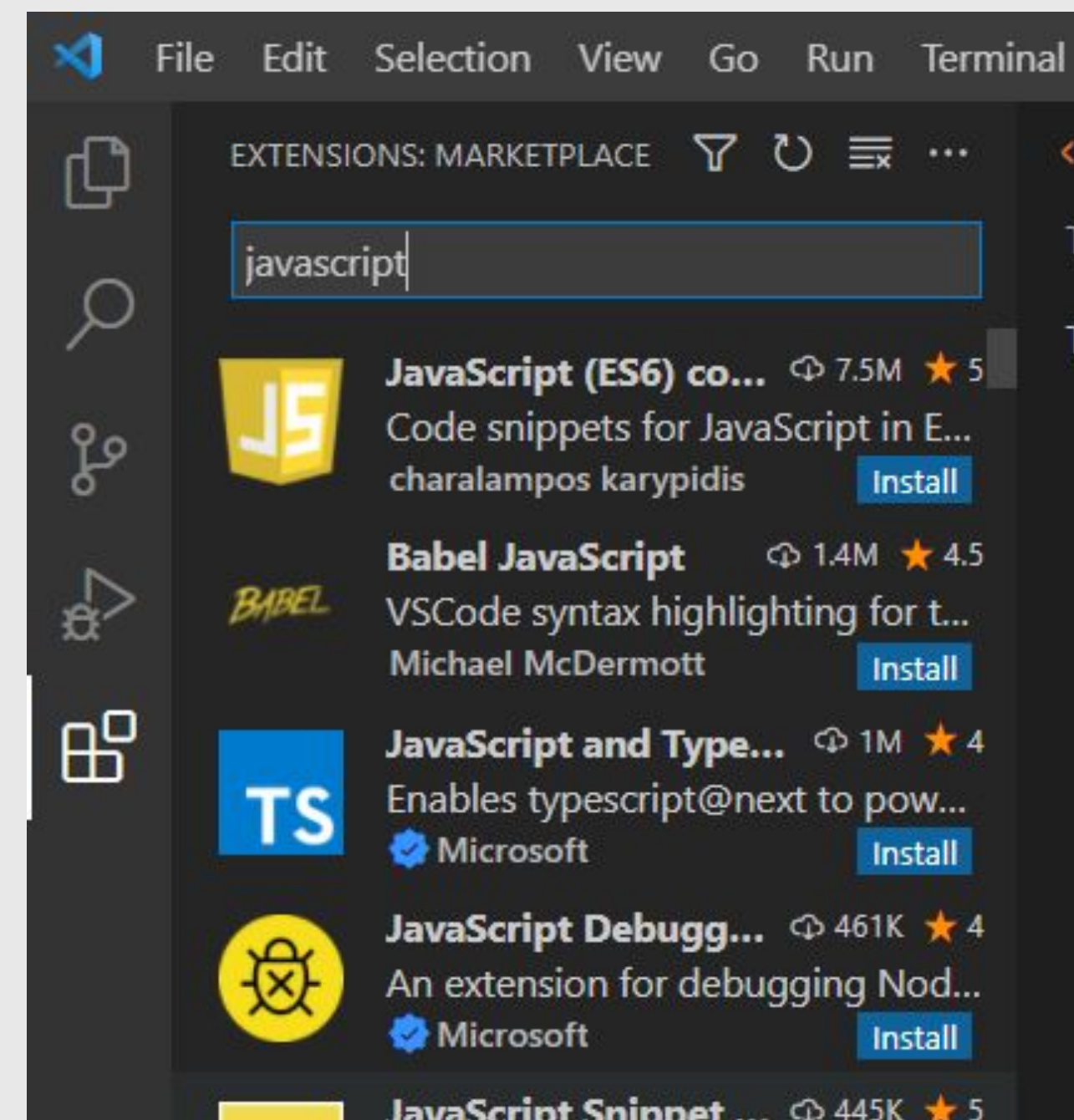
```
File Edit Selection View Go Run Terminal Help
<> resultados.html T:\Rotinas\... <> resultados.html C:\...\Área de Trab
C: > Users > Ricardo Alexandre > OneDrive > Área de Trabalho > Dass >
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="
7      <meta name="viewport" content="width=device-
8      <title>Gestão Produção</title>
9      <link rel="shortcut icon" href=" ../imagens/t
10
11      <!-- Chamada do CSS-->
12      <!--<link href="https://cdn.jsdelivrivr.net/npr
13          integrity="sha384-EVSTQN3/azprG1Anm3QDgp
14      <!-- Chamada do JS-->
15      <!--<script src="https://cdn.jsdelivrivr.net/np
16          integrity="sha384-IQsoLX15PILFhosVNubq5l
17          crossorigin="anonymous"></script>
```

#PraCegoVer: Imagem da Tela de Trabalho do VSCode

Instalação e Extensões VSCode

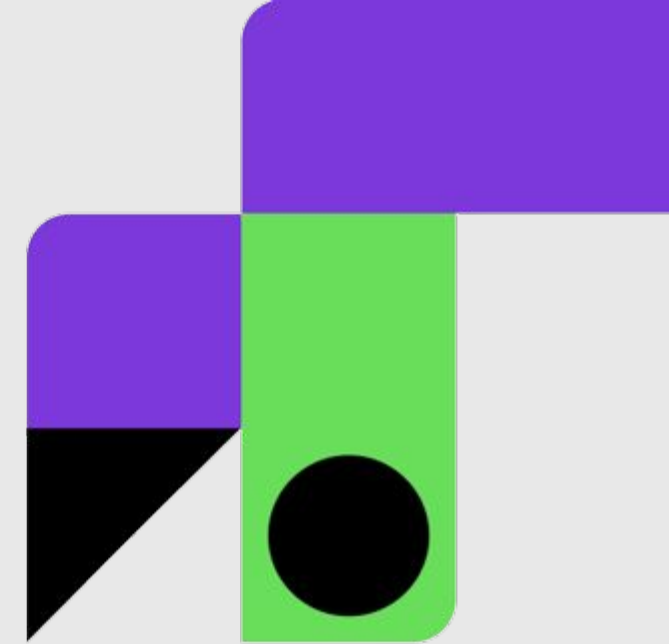
Para instalar extensões de dentro do Visual Studio, siga os passos abaixo:

1. Em ferramentas > extensões e atualizações, localize a extensão que você deseja instalar. Se você souber o nome ou parte do nome da extensão, poderá pesquisar na janela de pesquisa . Exemplo: javascript
2. Selecione Baixar. Esta extensão está agendada para instalação.



#PraCegoVer: Imagem da Tela de Instalações de Extensão do VSCode

Introdução aos Protocolos



Para iniciar, vou usar Visual Studio Code, mas você pode usar outros editores de textos . Dentro do fonte em HTML vamos criar a tag <Script> e ao término fechar a tag com </Script>

Dentro do Script, vamos inserir o código em JavaScript como o exemplo ao lado.

```
<html>
<head>
<title>Introdução - Olá World!</title>
<script language="javascript"
type="text/javascript">
alert('Olá World!');
</script>
</head>
<body>
</body>

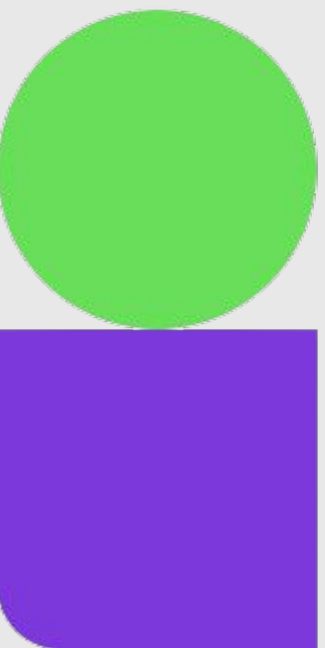
</html>
```

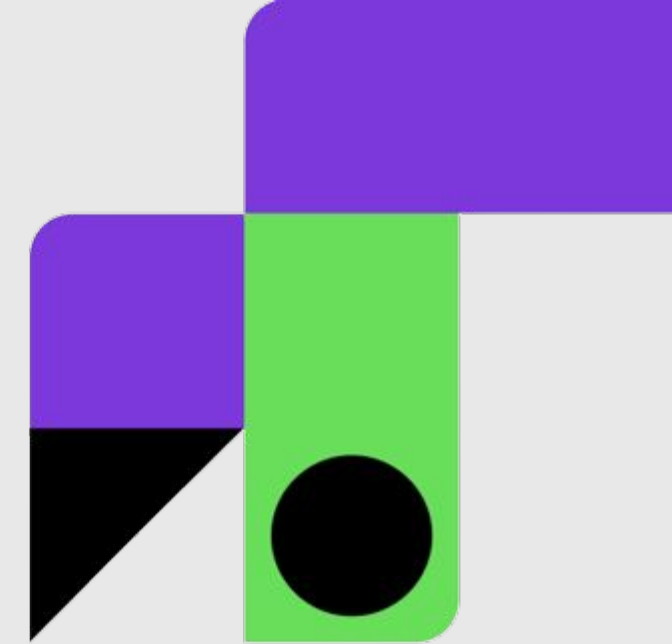
#PraCegoVer: Imagem da Tela de Instalações de Extensão do VSCode

O que é o DOM?

O DOM Document Object Model, traduzindo Modelo de objeto de documento, é utilizado pelo navegador Web para representar a sua página Web. Quando altera-se esse modelo com o uso do Javascript altera-se também a página Web.

O DOM é basicamente ele é uma representação de uma página HTML no mundo Javascript.



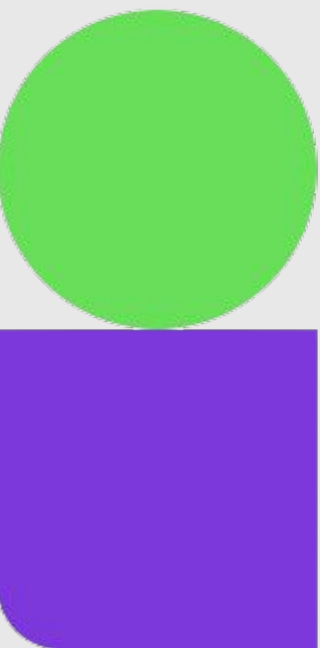


Elementos e Eventos

Elementos e Eventos

Elementos e Eventos em Javascript são categorizados em três áreas distintas sendo:

- a) Interface de usuário (teclado e mouse)
- b) Lógica (resultante de um processo qualquer)
- c) Mutação (ação que altera determinada área do documento).

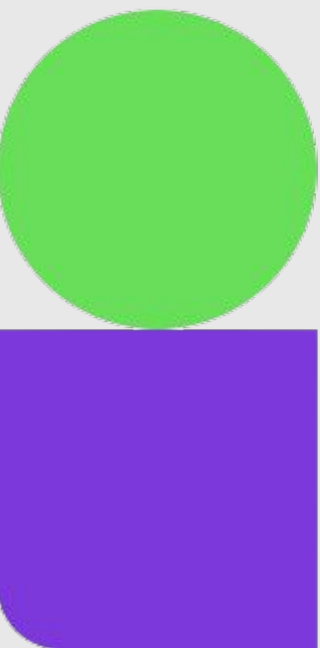


Usando o getElementById

O Método `getElementById()`, permite com que o programador possa obter um elemento do documento a partir de seu atributo ID especificado.

Este método é importante e muito empregado para obter informações ou manipular um elemento em um documento HTML.

Sua Sintaxe é: `document.getElementById("idNome").value;`



Usando o getElementById

Exemplo da Aplicação

```
<html>

<body>
<h4>Usando em JavaScript o Elemento getElementById</h4>

  <p id="demo"></p>

  <script>
    document.getElementById("demo").innerHTML = "Olá Mundo";
  </script>
</body>
</html>
```

Imagem do resultado:

Usando em JavaScript o Elemento getElementById

Olá Mundo

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com frase 'Ola Mundo' em preto.

Usando o getElementsByTagName

O método `getElementsByTagName` da interface `Document` permite retorna um `HTMLCollection` de elementos com o nome da tag fornecido, no momento do desenvolvimento.

Sua Sintaxe é: `document.getElementById("idNome").value;`

Usando o getElementsByTagName

Exemplo da Aplicação

```
<!DOCTYPE html>
<html>
<body>

<h4>Usando em JavaScript o Elemento getElementsByTagName</h4>

<p>Isso é um Parágrafo</p>
<p>Isso é um Parágrafo</p>

<script>
document.getElementsByTagName("p")[0].innerHTML = "Olá Mundo";
</script>

</body>
</html>
```

Imagem do resultado:

Usando em JavaScript o Elemento getElementsByTagName

Olá Mundo

Isso é um Parágrafo

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com frase 'Ola Mundo' e 'Isso é um parágrafo' em preto.

Usando o getElementByClassName

O getElementByClassName tem como principal objetivo, buscar todos os elementos que tenham uma dada classe e retorna uma coleção de elementos. O Detalhe importante: Este método retorna uma coleção.

Sua Sintaxe é: `document.getElementsByClassName().value;`

Usando o getElementByClassName

Exemplo da Aplicação

```
<!DOCTYPE html>
<html>
<body>

<h4>Usando em JavaScript o Elemento getElementByClassName</h4>

<p>Altere o texto do primeiro elemento com class="example" :</p>

<div class="example">Element01</div>
<div class="example">Element02</div>

<script>
const collection = document.getElementsByClassName("example");
collection[0].innerHTML = "Olá Mundo";
</script>

</body>
</html>
```

Imagem do resultado:

Usando em JavaScript o Elemento getElementByClassName

Altere o texto do primeiro elemento com class="example" :

Olá Mundo
Element02

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com frase 'Ola Mundo' e "Elemento02"

Usando o `document.querySelector`

O Elemento `document.querySelector (selectors);` É utilizado no JavaScript, onde: `element` é um objeto `Element`. E o `selectors` é uma string que contém um ou mais seletores CSS separados por vírgulas..

Sua Sintaxe é: `element.document.querySelector();`

Usando o document.querySelector

Exemplo da Aplicação

```
<html>
<body>

<h4>Usando em JavaScript o Elemento querySelector:</h4>
<h3>Adicione uma cor de fundo ao primeiro elemento p:</h3>

<p>Este é um elemento p.</p>
<p>Este é um elemento p.</p>

<script>
document.querySelector("p").style.backgroundColor = "red";
</script>

</body>
</html>
```

Imagem do resultado:

Usando em JavaScript o Elemento querySelector:

Adicione uma cor de fundo ao primeiro elemento p:

Este é um elemento p.

Este é um elemento p.

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com destaque em vermelho pra frase 'esse é um elemento p'

Usando o createElement

A função createElement() tem como principal função criar um elemento HTML para ser, posteriormente, inserido em um documento HTML.

Sua Sintaxe é: **var** element = document.createElement(tagName);

Usando o createElement

Exemplo da Aplicação

```
<!DOCTYPE html>
<html>
<body>

<h2>O método createElement() </h2>

<p>Crie um elemento p com algum texto:</p>

<script>
// Create element:
const p = document.createElement("p");
p.innerText = "Inserindo um conteúdo no Parágrafo.";

// Append to body:
document.body.appendChild(p);
</script>

</body>
</html>
```

Imagem do resultado:

O método createElement()

Crie um elemento p com algum texto::

Inserindo um conteúdo no Parágrafo.

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com as frases 'crie um elemento p com algum texto:' e 'inserindo um conteúdo no parágrafo'

Usando o InsertBefore

A função insertBefore tem como principal objetivo inserir um elemento antes (before) de um determinado elemento.

Por tanto, o uso desta função é indicado para trabalhar normalmente com listas, tabelas, itens e etc... ou seja, elementos que representam uma listagem.

Sua Sintaxe é: `elemento_pai.insertBefore(novoElem, elemDeReferencia);`

Usando o InsertBefore

Exemplo da Aplicação

```
<!DOCTYPE html>
<html>
<body>
<ul>
  <li>primeiro item</li>
  <li>segundo item</li>
  <li>quarto item</li>
</ul>
<script>
// Criando o terceiro elemento
var novoElem = document.createElement('li');
var texto    = document.createTextNode('terceiro item');
novoElem.appendChild(texto);
// Recuperando a lista
var lista = document.getElementsByTagName('ul')[0];
// Recuperando os itens
var itens = document.getElementsByTagName('li');
// Inserindo com insertBefore()
lista.insertBefore(novoElem, itens[0]);
</script>
</body>
</html>
```

Imagem do resultado:

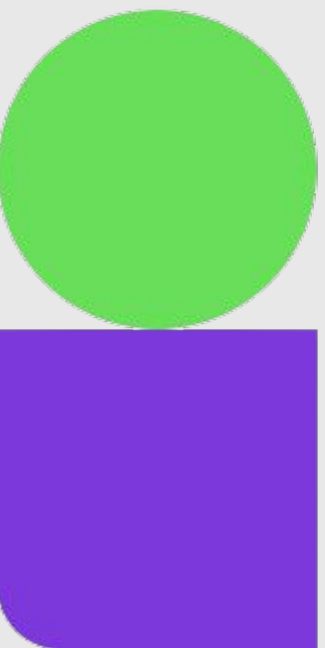
- terceiro item
- primeiro item
- segundo item
- quarto item

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com frases em preto no formato de tópicos. Terceiro item, primeiro item, segundo item, quarto item.

Usando o AppendChild

Neste processo, se filho é uma referência a um nó existente no documento, appendChild o evento irá movê-lo de sua posição atual para a nova posição, não tendo a necessário remover o nó de seu pai atual antes de adicioná-lo a outro nó.

Sua Sintaxe é: `document.body.appendChild(p);`



Usando o AppendChild

Exemplo da Aplicação

```
<!DOCTYPE html>
<html>
<body>
  <script>
    var elemento_pai = document.body;
    var titulo = document.createElement('h1');
    // Criar o nó de texto
    var texto = document.createTextNode("Um Novo Título");
    // Anexar o nó de texto ao elemento h1
    titulo.appendChild(texto);
    elemento_pai.appendChild(titulo);
  </script>
</body>
</html>
```

Imagem do resultado:

Um Novo Título

Um título qualquer

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela utilizando H1. Tela branca com as frases 'Um novo Título' e "um título qualquer".

Usando o parentNode

Neste processo esta propriedade contém o elemento pai do elemento informado, em nosso código de exemplo buscamos o elemento pai que é o do elemento informado, permite fazer a troca das ordens.

Sua Sintaxe é: `var elemento=document.getElementById("item1").parentNode.nodeName;`

Usando o parentNode

```
<!doctype html>
<html lang="pt-br">
  <head>
    <title>Curso de Javascript</title>
    <meta charset="UTF-8">
    <script>

      function primeiroElemento(){
        var elemento=document.getElementById("item1").parentNode.nodeName;
        alert(elemento);
      }

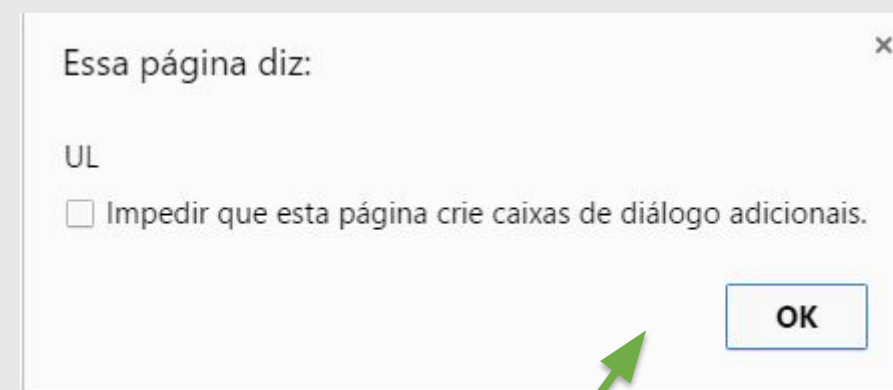
    </script>
  </head>
  <body>

    <ul id="info">
      <li id="item1">Monitor</li>
      <li id="item2">Mouse</li>
      <li id="item3">Teclado</li>
      <li id="item4">Impressora</li>
    </ul>

    <button onclick="primeiroElemento()">Elemento Pai</button>

  </body>
</html>
```

Imagem do resultado:



#PraCegoVer: imagem mostrando como o JavaScript aparece em tela utilizando li. Tela branca com os tópicos:
Scanner
Mouse
Impressora
Teclado
E um botão Elemento Pai que chama um Pop Up.

Usando o get e setAttribute

Obtendo o valor do atributo do elemento O método `getAttribute()` é usado para obter o valor atual de um atributo no elemento. Se o atributo especificado não existir no elemento, ele retornará null. Aqui está um exemplo:

Sua Sintaxe é: `var elemento=document.getElementById("item1").parentNode.nodeName;`

Usando o get e setAttribute

Exemplo da Aplicação

```
<html>

<a href="https://www.google.com/" target="_blank" id="myLink">Google</a>

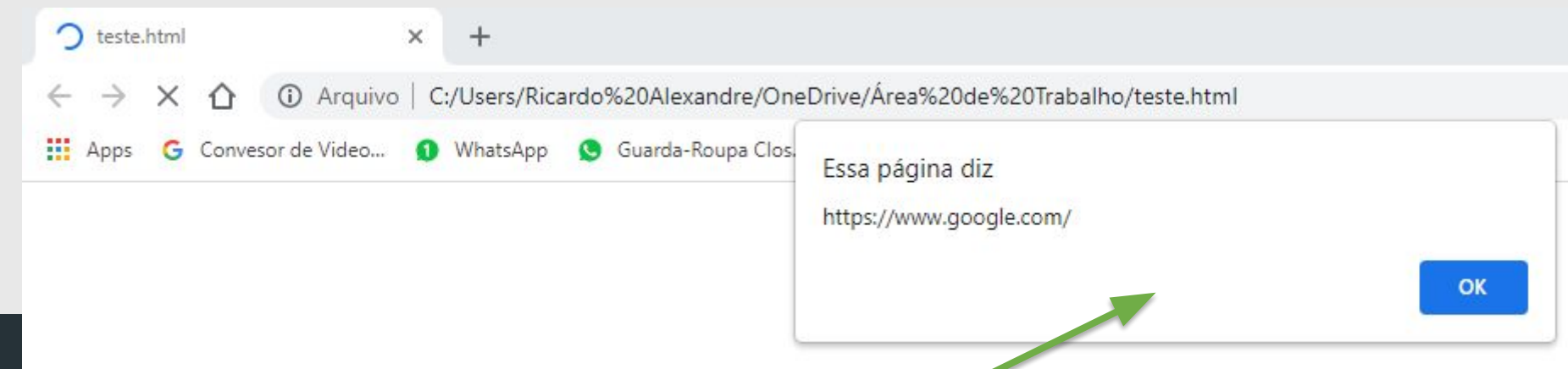
<script>
  // Selecting the element by ID attribute
  var link = document.getElementById("myLink");

  // Obtendo os valores dos atributos
  var href = link.getAttribute("href");
  alert(href); // Saídas : https://www.google.com/

  var target = link.getAttribute("target");
  alert(target); //Saídas : _blank
</script>

</html>

</body>
</html>
```



#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com as frases no Pop Up, ao clicar demonstra mais uma tela de alert e o link do Google.

Alterando o estilo CSS com JS

A ideia por trás deste procedimento é muito simples, antes de tudo, precisamos selecionar o elemento alvo.

Como vimos nos exemplos anteriores, seja por classe, id ou query selector, para isso você pode utilizar os métodos:

- `getElementById`
- `getElementsByClassName`
- `querySelector` ou `querySelectorAll`

Após este procedimento, você precisa acessar a propriedade `style` do elemento selecionado, e fazer as alterações no CSS.

Lembrando que regras como `background-color` que tem duas palavras, você deve substituir pelo `camelCase` desta maneira: `backgroundColor`

Alterando o estilo CSS com JS

Exemplo da Aplicação

```
<!DOCTYPE html>
<html>
<head>
  <title>Como usar JavaScript para mudar propriedades CSS</title>
  <meta charset="utf-8">
</head>
<body>
  <p id="paragrafo">Este parágrafo vai ter o CSS alterado!</p>

  <script>
let el = document.getElementById('paragrafo');
// alterando uma propriedade
el.style.color = 'red';
// varias propriedades
el.style.cssText =
  'color: blue;' +
  'background-color: yellow;' +
  'border: 1px solid magenta';
</script>
</body>
</html>
```

Este parágrafo vai ter o CSS alterado!

Este parágrafo vai ter o CSS alterado!

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com a frase 'este parágrafo vai ter o CSS alterado', após execução a frase estará recebendo um formato em CSS deixando em amarelo a frase.

Pegando todos os determinados elementos

Para ilustrar melhor este processo iremos pegar todos os elementos de um array com JavaScript, lembrando um array é uma matriz.

Métodos dos arrays

- 1.concat() une dois arrays e retorna um novo array. ...
- 2.join(delimitador = ',') une todos os elementos de um array dentro de um string.
...
- 3.push() adiciona um ou mais elementos no fim de um array e retorna o comprimento resultante do array. ...
- 4.pop() remove o último elemento de um array e retorna esse elemento.

Pegando todos os determinados elementos

Exemplo da Aplicação

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript - Arrays</h2>
<p>O objeto Array é usado para armazenar vários valores em uma
única variável: </p>
<p id="demo"></p>
<script>
const cars = ["Gol", "HB20", "Tiggo"];
document.getElementById("demo").innerHTML = cars;
</script>
</body>
</html>
```

JavaScript - Arrays

O objeto Array é usado para armazenar vários valores em uma única variável:

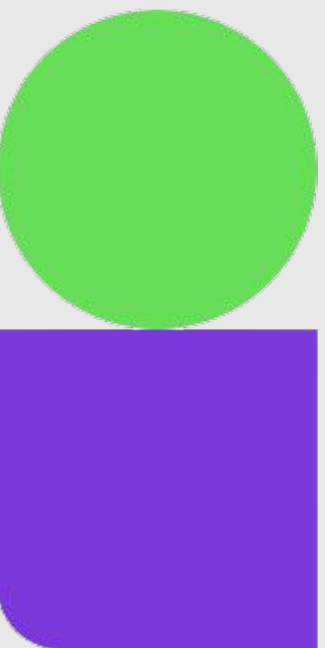
Gol,HB20,Tiggo

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela utilizando Array. Tela branca com a frase “O objeto Array é usado para armazenar vários valores em uma única variável:”

Criando e removendo elementos

Para se remover um elemento antes de tudo temos que ter criado um elemento, após isso precisamos acessar o elemento pai, do elemento que precisa ser removido.

Para isso, poderemos aplicar o método `removeChild` no nó alvo.



Criando e removendo elementos

Exemplo da Aplicação

```
<!DOCTYPE html>
<html>
<body>
  <div>
    <p id="elemento-para-remover">Remover este parágrafo</p>
  </div>
<script>
var elemento = document.querySelector("#elemento-para-remover");
elemento.parentNode.removeChild(elemento);
</script>
</body>
</html>
```

Remover este parágrafo

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela
Tela branca com a frase Remover este parágrafo e depois executando o JS deixando a tela em branco.

Evento Click

O evento `onclick` permite executar uma determinada funcionalidade quando um botão é clicado. Isso pode ocorrer quando um usuário envia dados de um formulário, ou quando você altera um determinado conteúdo na página da web e assim por diante. Você coloca a função em JavaScript que você quer executar dentro da tag de abertura do botão.

Capturando movimentos do Mouse

A captura de movimentos do mouse, pode ser definida por dois contextos:

onMouseOver:

define ação quando o usuário passa o mouse sobre o elemento

onMouseOut:

define ação quando o usuário retira o mouse sobre o elemento

Capturando Eventos do scroll

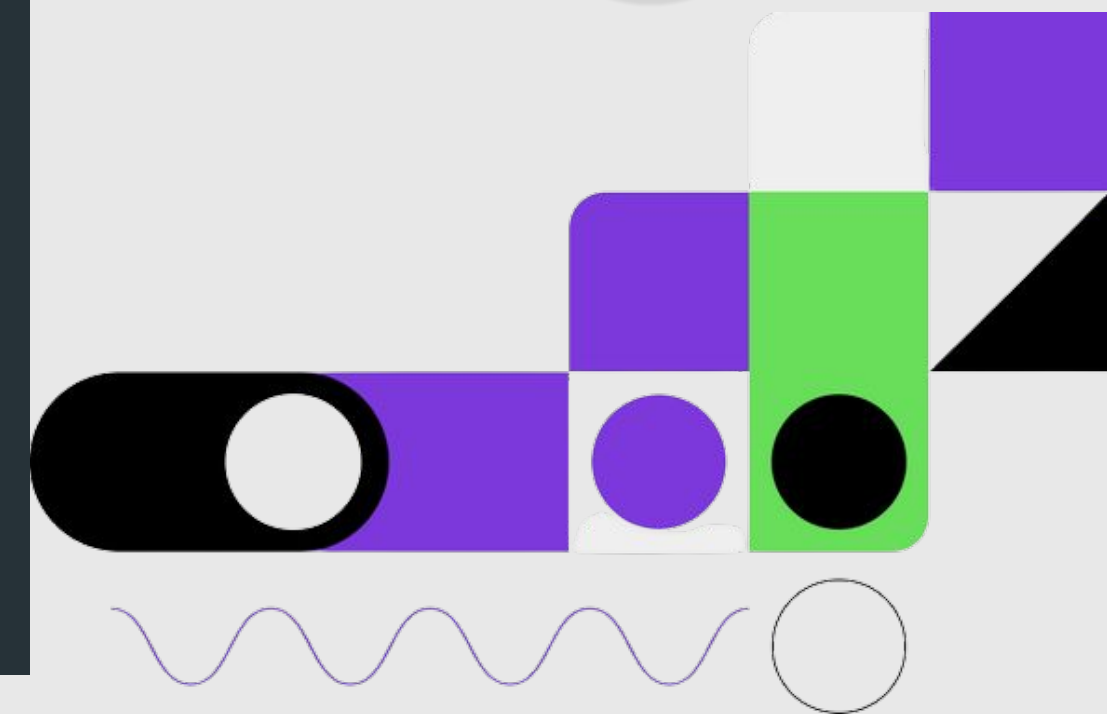
O evento `onscroll` ocorre quando a barra de rolagem de um elemento está sendo rolada, através da roda de rolagem do mouse.

Dica: use a propriedade CSS `overflow style` para criar uma barra de rolagem para um elemento.

Exemplos dos Eventos

```
<!DOCTYPE html>
<html>
<head>
<title>Usando eventos no Javascript, onmousedown, onmouseup e onclick</title>
</head>
<body>
// Evento de movimento do mouse para baixo
<div onmousedown="mDown(this)" onmouseup="mUp(this)"
style="background-color:#D94A38;width:90px;height:20px;padding:40px;">Clique
aqui</div>
// Evento de movimento do mouse para cima
<script>
function mDown(obj)
{
obj.style.backgroundColor="#1ec5e5";
obj.innerHTML="Solte o clique"
}
// Evento de movimento do mouse para cima
function mUp(obj)
{
obj.style.backgroundColor="#D94A38";
obj.innerHTML="Obrigado"
}
</script>
```

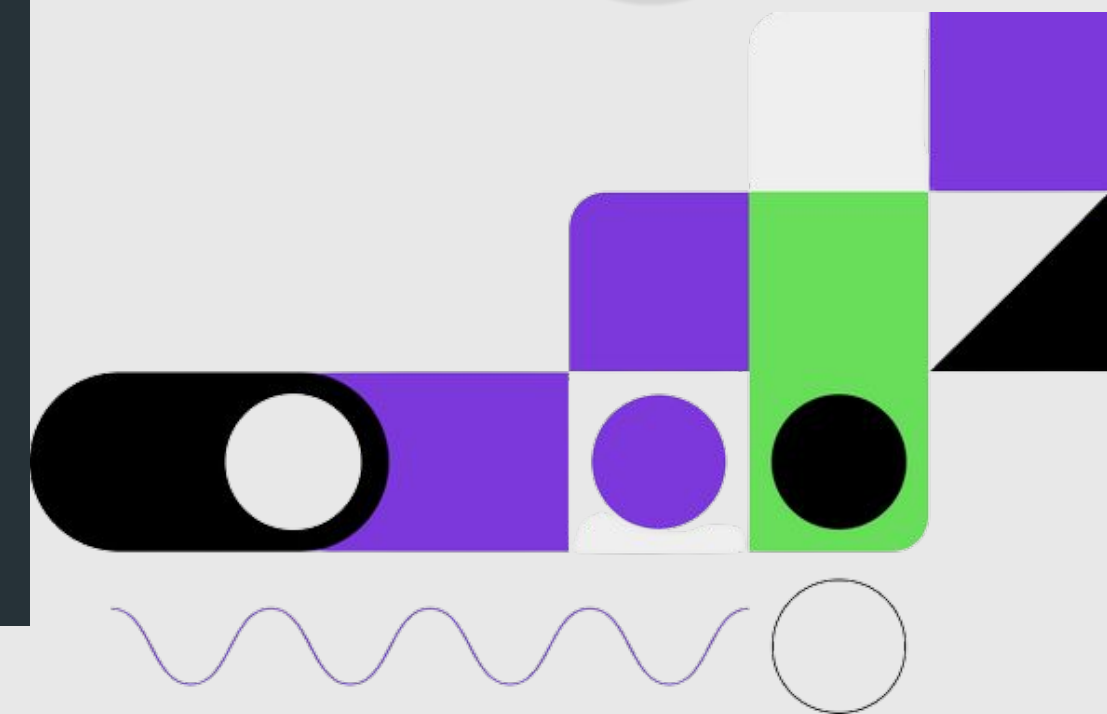
*Evento click e movimentos
do mouse!*



Exemplos dos Eventos

```
<!DOCTYPE html>
<html>
<style>
.test {
  background-color: yellow;
}
</style>
<body style="font-size:20px;height:1500px">
<h1>0 evento onscroll </h1>
<p id="myP" style="position:fixed">
  Se você rolar 50 pixels do topo desta página, a classe "teste" será adicionada. Role
  para cima novamente para remover a classe.
</p>
<script>
// Capturando o evento scroll com a função
window.onscroll = function() {myFunction()};
function myFunction() {
  if (document.body.scrollTop > 50 || document.documentElement.scrollTop > 50) {
    document.getElementById("myP").className = "test";
  } else {
    document.getElementById("myP").className = "";
  }
}
</script>
</body>
</html>
```

*Capturando
Eventos do scroll*



Definição de Estrutura de dados

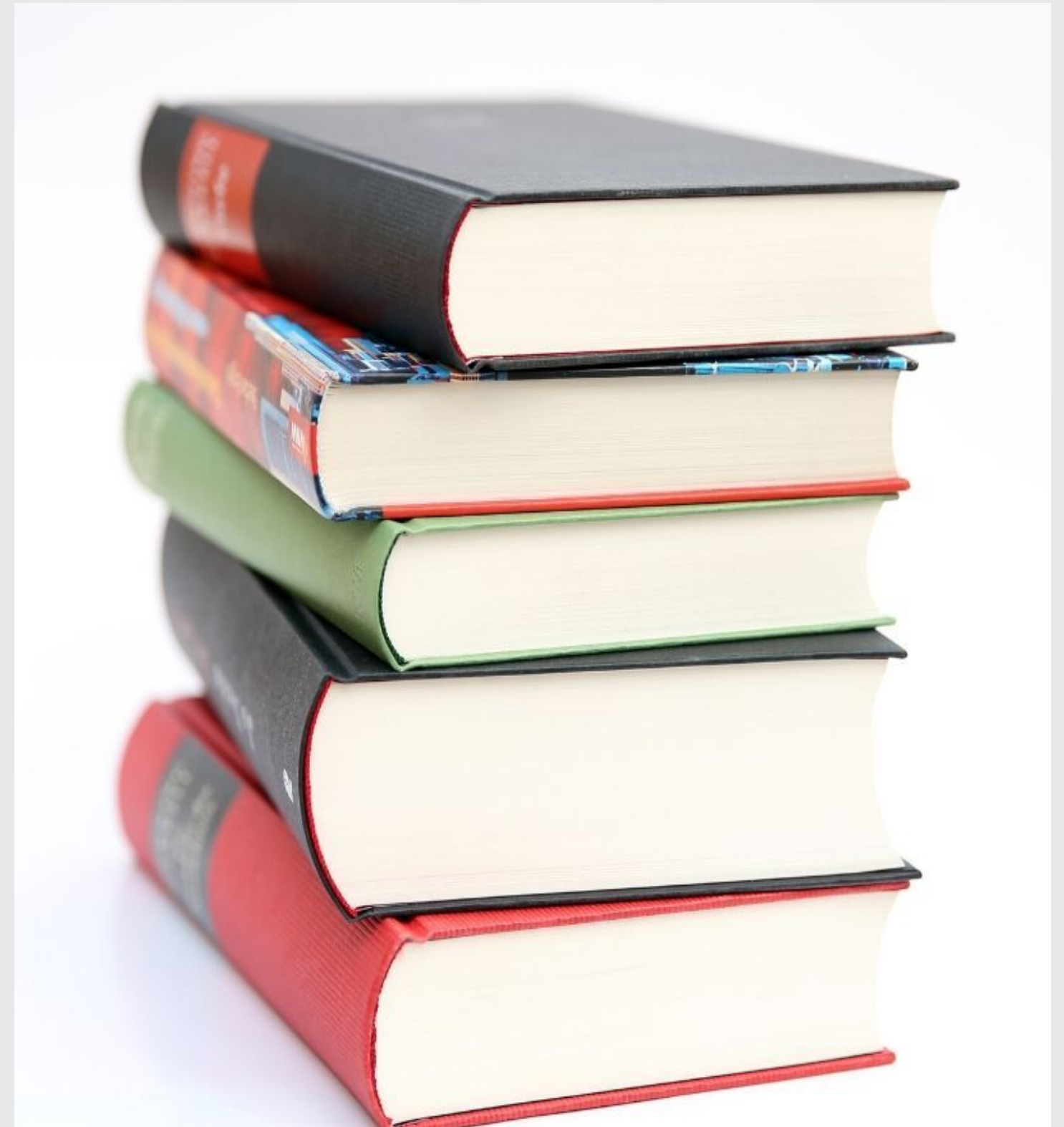
Por padrão, todas as linguagens de programação, têm sua própria estrutura de dados embutida, mas essa estrutura frequentemente difere uma da outra. Os tipos de dados disponíveis na linguagem JavaScript são muito parecidos em outras linguagens, vamos ver as propriedades que elas possuem.





Instalando o Node com NVM

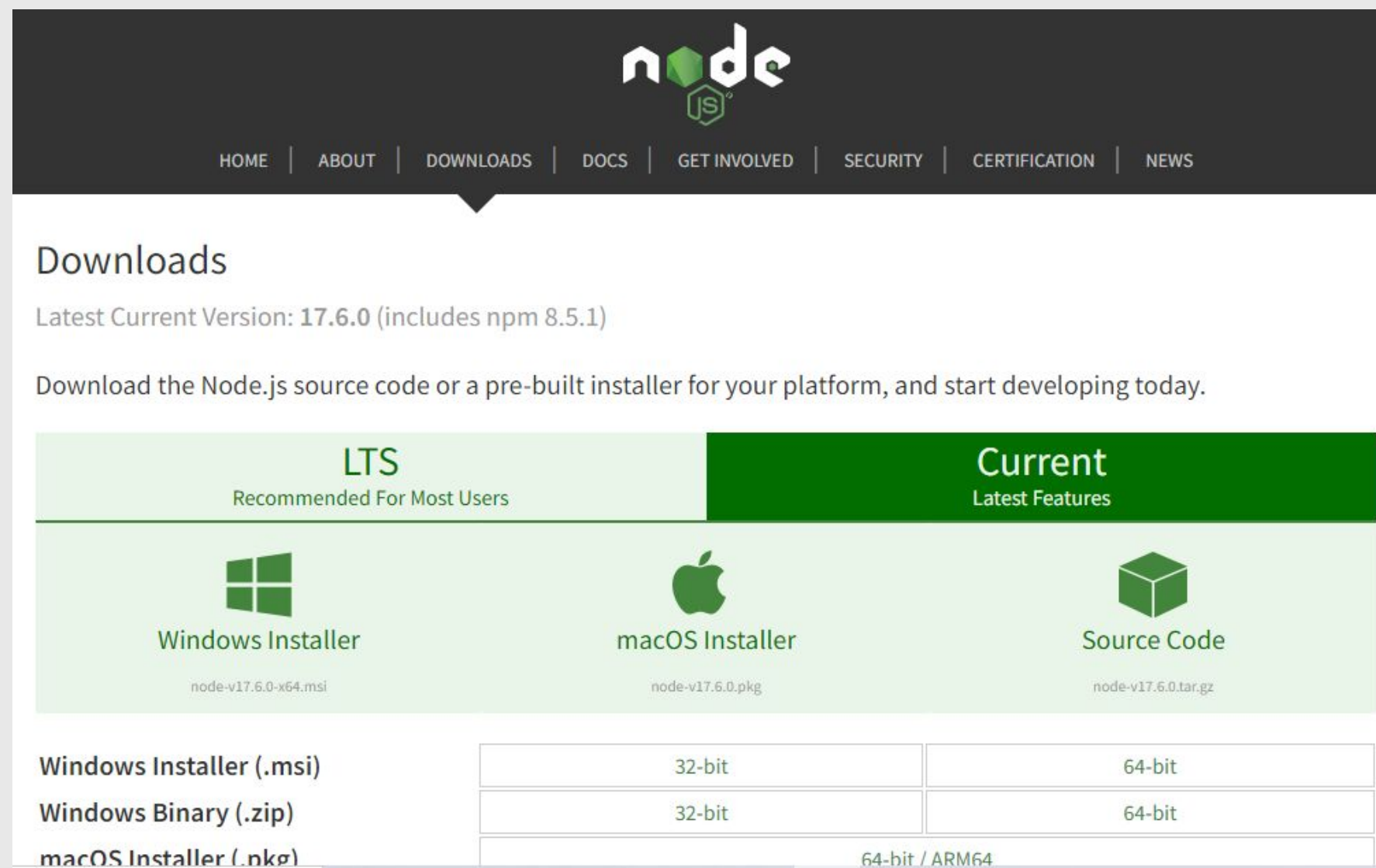
Com o NVM, (Node Version Manager) deixa tudo bem simples, você pode alterar a versão do Node.js para aquela que precisar executando apenas um comando. Para instalar o NVM, o processo pode ser feito pelo Homebrew ou você pode instalar seguindo as recomendações oficiais da documentação.



Instalação do NodeJS e NPM no Windows

Para instalar o nodejs e o npm no Windows, primeiro você precisa acessar a página de downloads do site oficial para baixar o instalador.

Download | Node.js (nodejs.org)

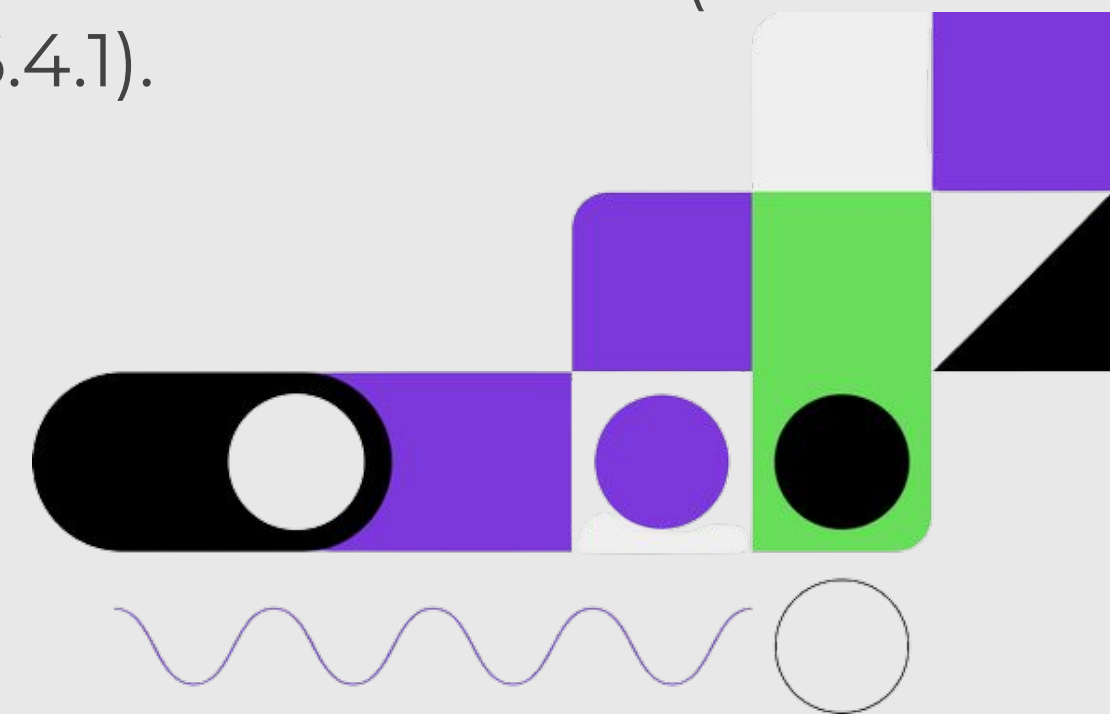


The screenshot shows the Node.js Downloads page. At the top, there's a dark header with the Node.js logo and navigation links: HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, and NEWS. Below the header, the 'Downloads' section is highlighted. It states 'Latest Current Version: 17.6.0 (includes npm 8.5.1)' and 'Download the Node.js source code or a pre-built installer for your platform, and start developing today.' There are two main tabs: 'LTS' (Recommended For Most Users) and 'Current' (Latest Features). Under the 'LTS' tab, there are three options: 'Windows Installer' (node-v17.6.0-x64.msi), 'macOS Installer' (node-v17.6.0.pkg), and 'Source Code' (node-v17.6.0.tar.gz). Below these, there's a table for Windows and macOS installers.

| | 32-bit | 64-bit |
|--------------------------|--------|----------------|
| Windows Installer (.msi) | | |
| Windows Binary (.zip) | | |
| macOS Installer (.pkg) | | 64-bit / ARM64 |

Geralmente tem duas versões a LTS que tem suporte de longo prazo e a Current que é a última versão. Neste momento a versão LTS é a 10.13.0 (com npm 6.4.1) e a última versão é a 11.0.0 (com npm 6.4.1).

#PraCegoVer: Tela de Instalação
Tela da máquina do Gerador de Conteúdo

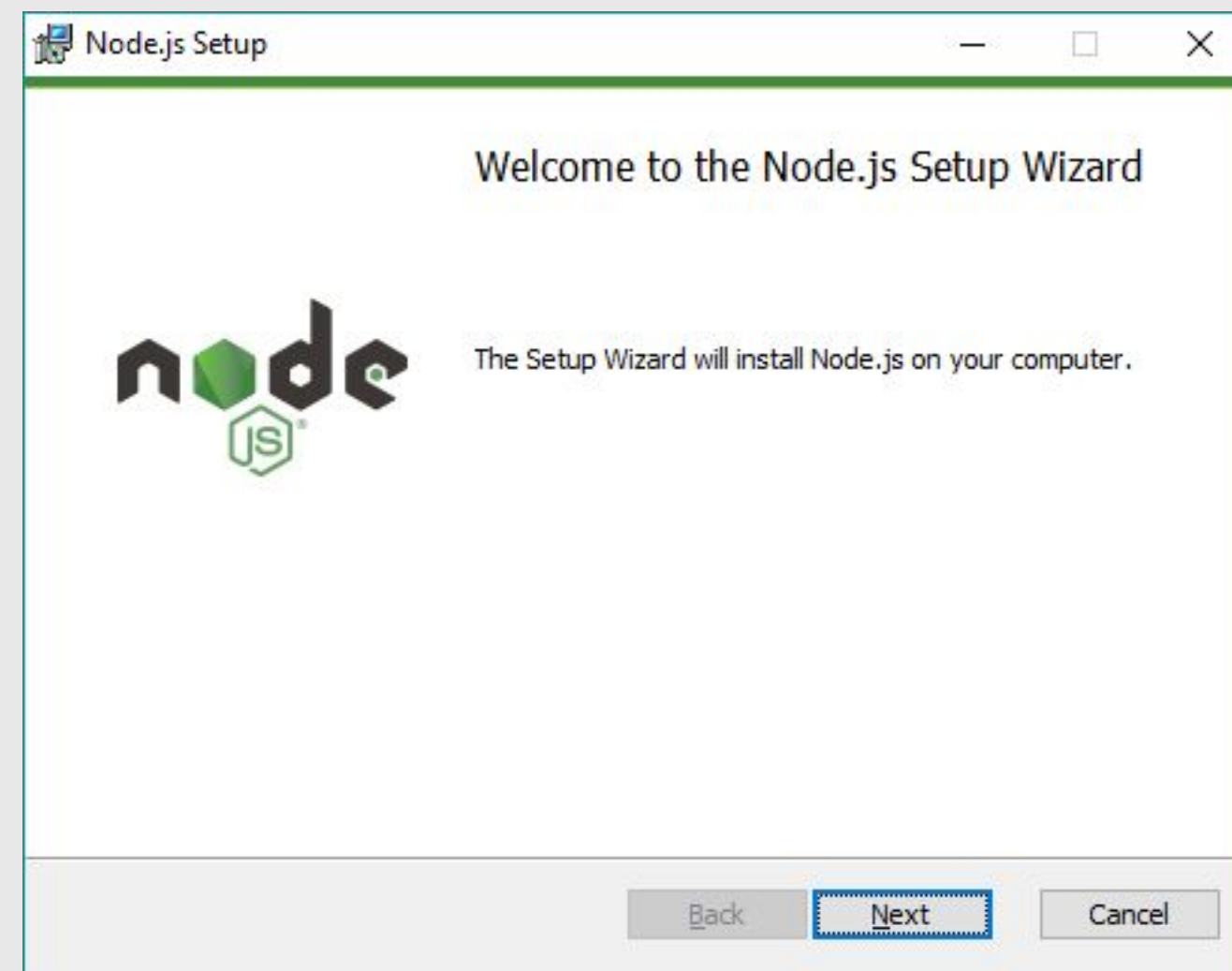


Instalação do NodeJS e NPM no Windows

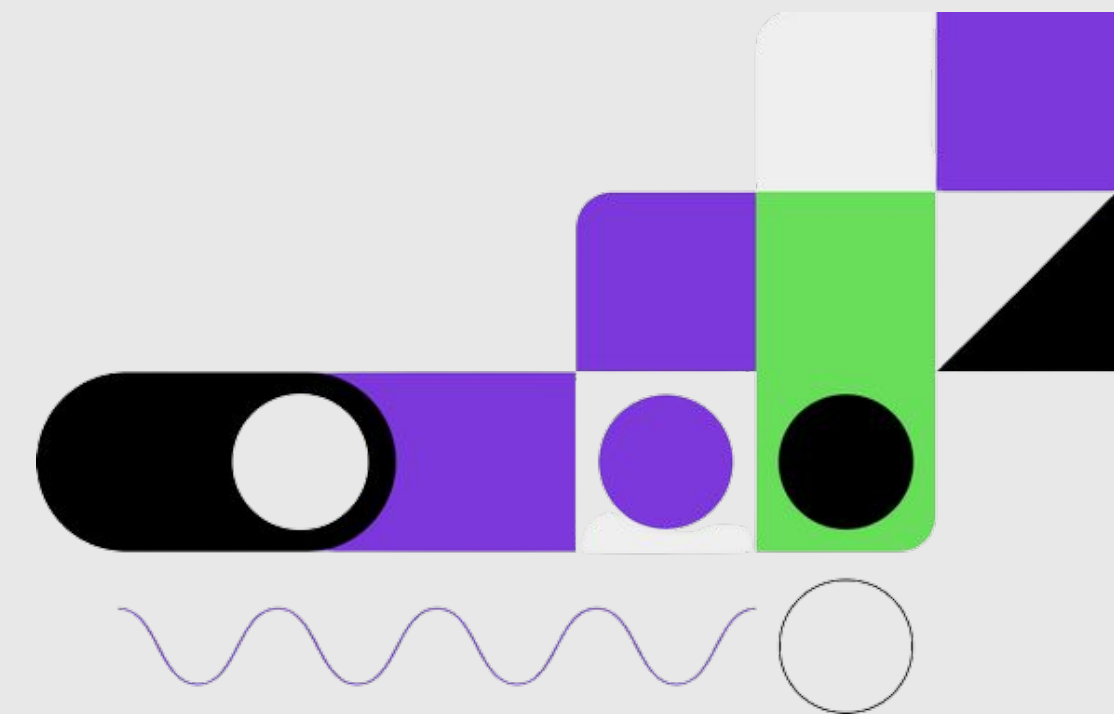
Escolha a versão de sua preferência e baixe o instalador para o seu sistema operacional, no meu caso neste vou usar o Windows 64 bits.

Após baixar, execute o instalador.

A primeira tela do instalador do NodeJS apresenta uma mensagem de boas vindas do instalador informando que será instalado o Node.js no seu computador, basta clicar em Next para continuar.

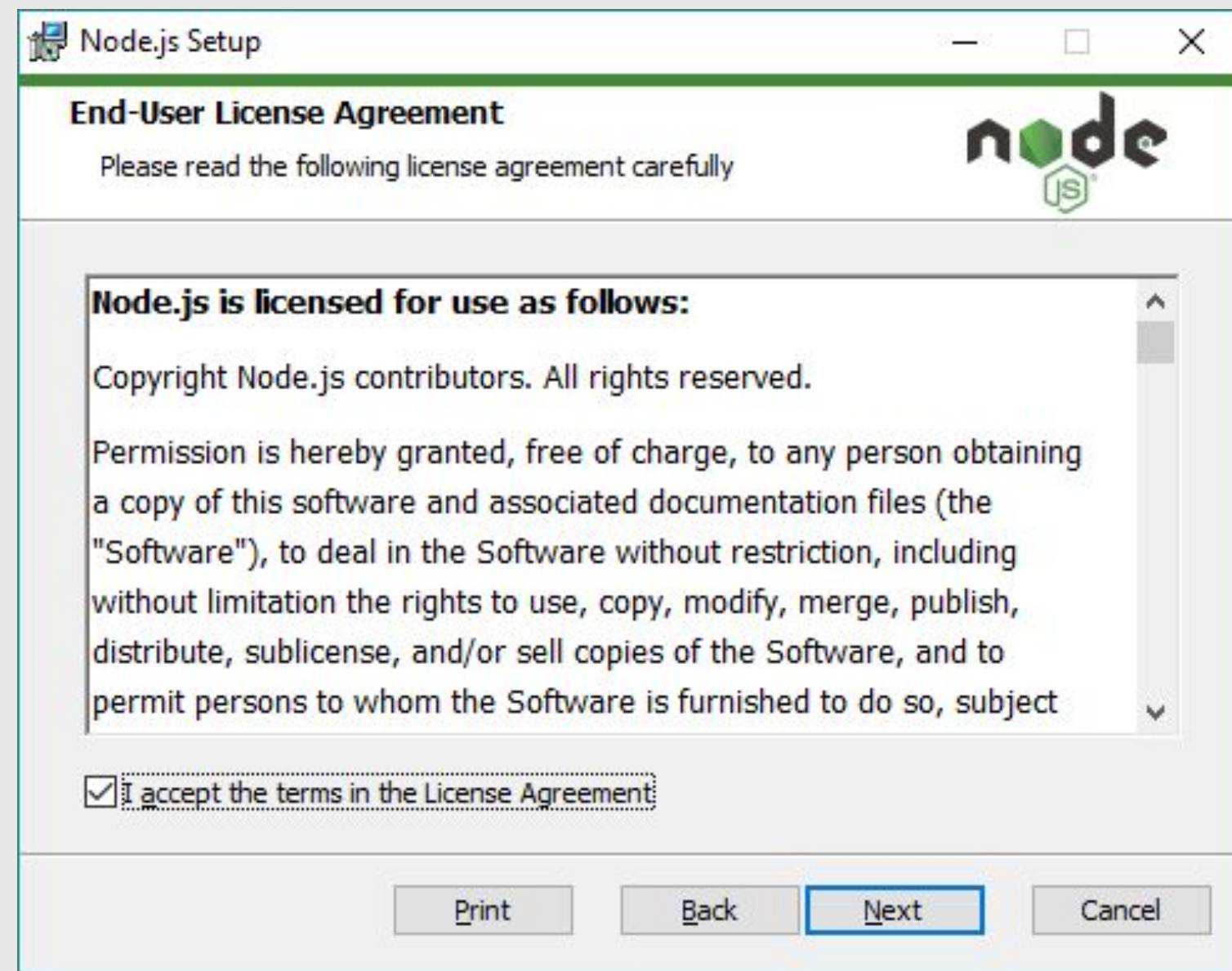


#PraCegoVer: Tela de Instalação
Tela da máquina do Gerador de Conteúdo

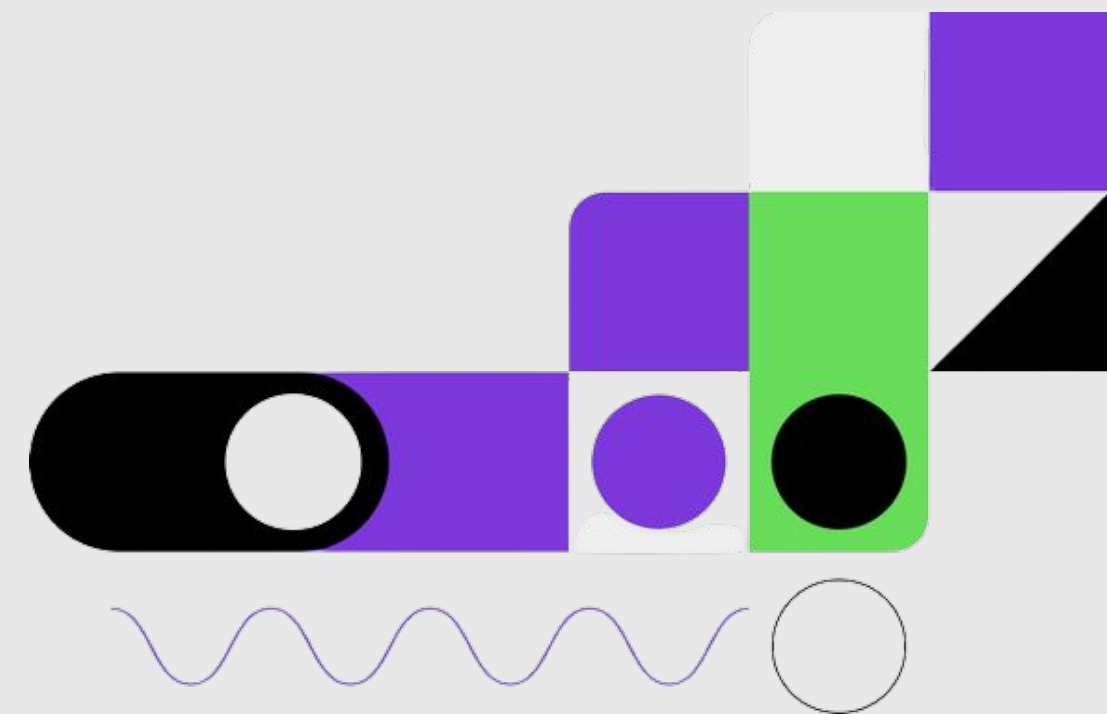


Instalação do NodeJS e NPM no Windows

A segunda tela do instalador do NodeJS pede para ler a licença de uso e aceitar os termos. Se você concordar com os termos, marque a caixinha **I accept ...** e clique em **Next** para continuar.

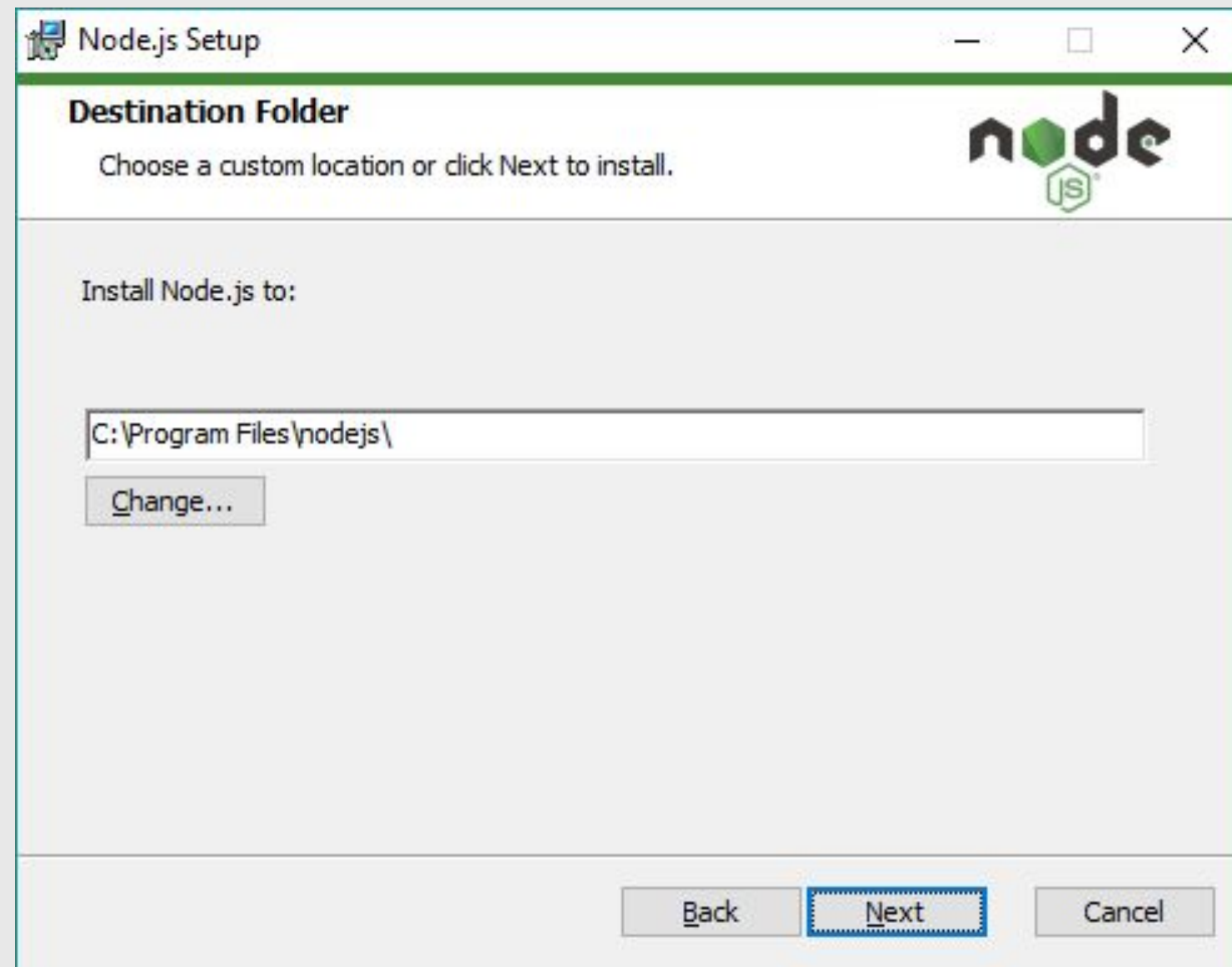


#PraCegoVer: Tela de Instalação
Tela da máquina do Gerador de Conteúdo

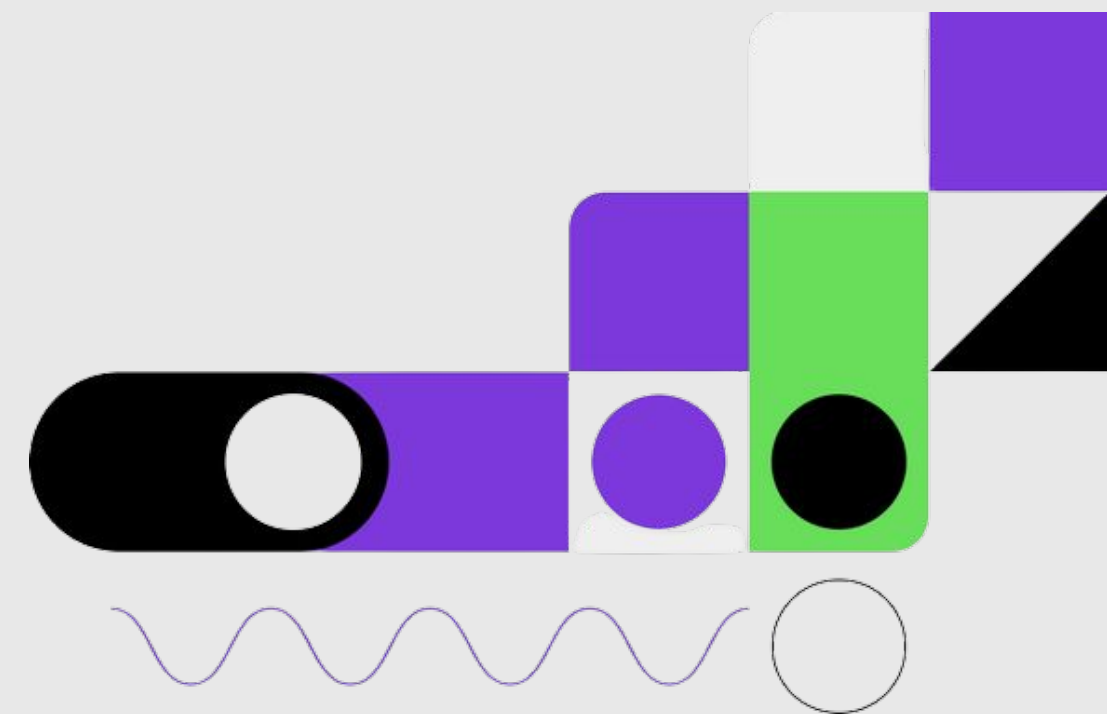


Instalação do NodeJS e NPM no Windows

A tela seguinte oferece a opção de alterar a pasta onde o NodeJS será instalado, escolha o local da instalação e clique em Next para continuar.

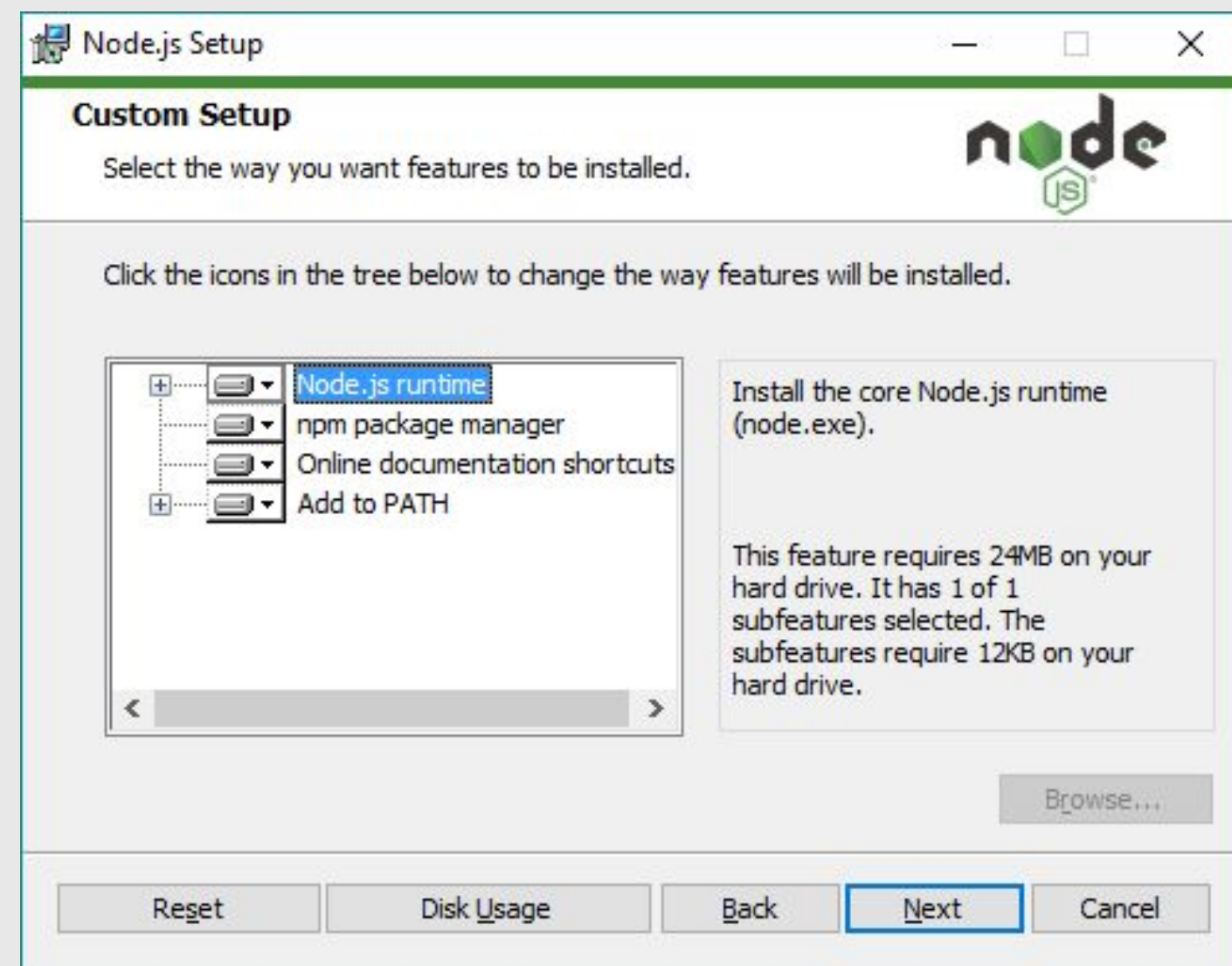


#PraCegoVer: Tela de Instalação
Tela da máquina do Gerador de Conteúdo

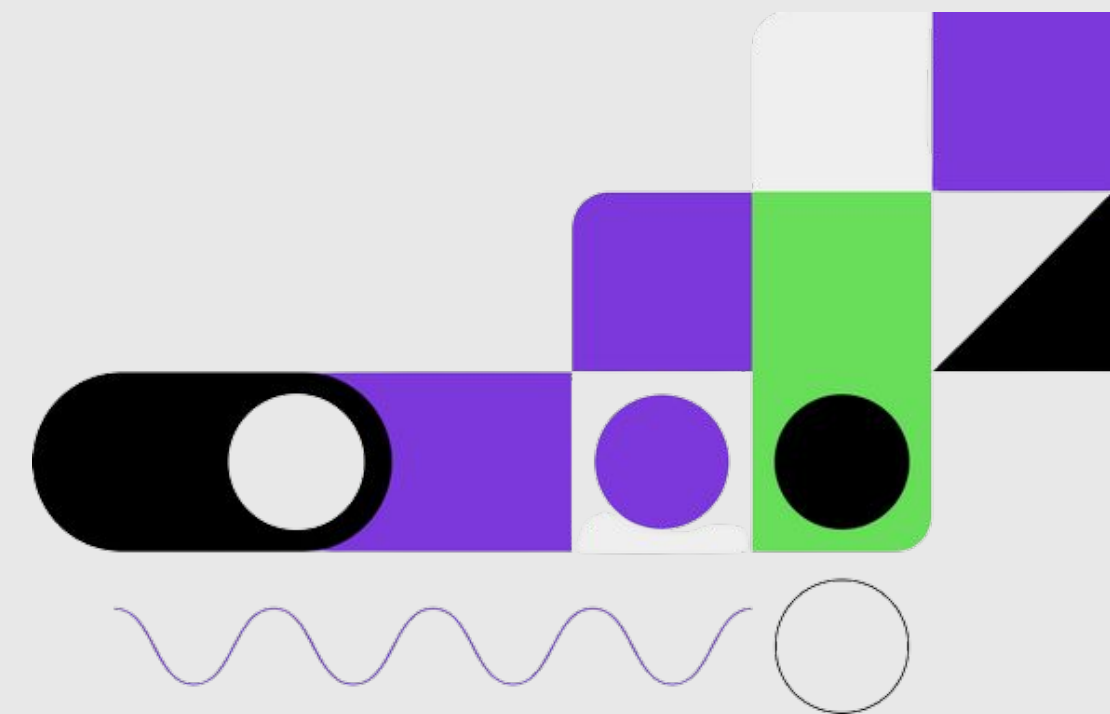


Instalação do NodeJS e NPM no Windows

Em seguida o instalador permite personalizar algumas opções da instalação como adicionar ou não o caminho do node e npm na variável PATH, etc. Eu deixo todas as opções marcadas. Clique em Next para continuar.

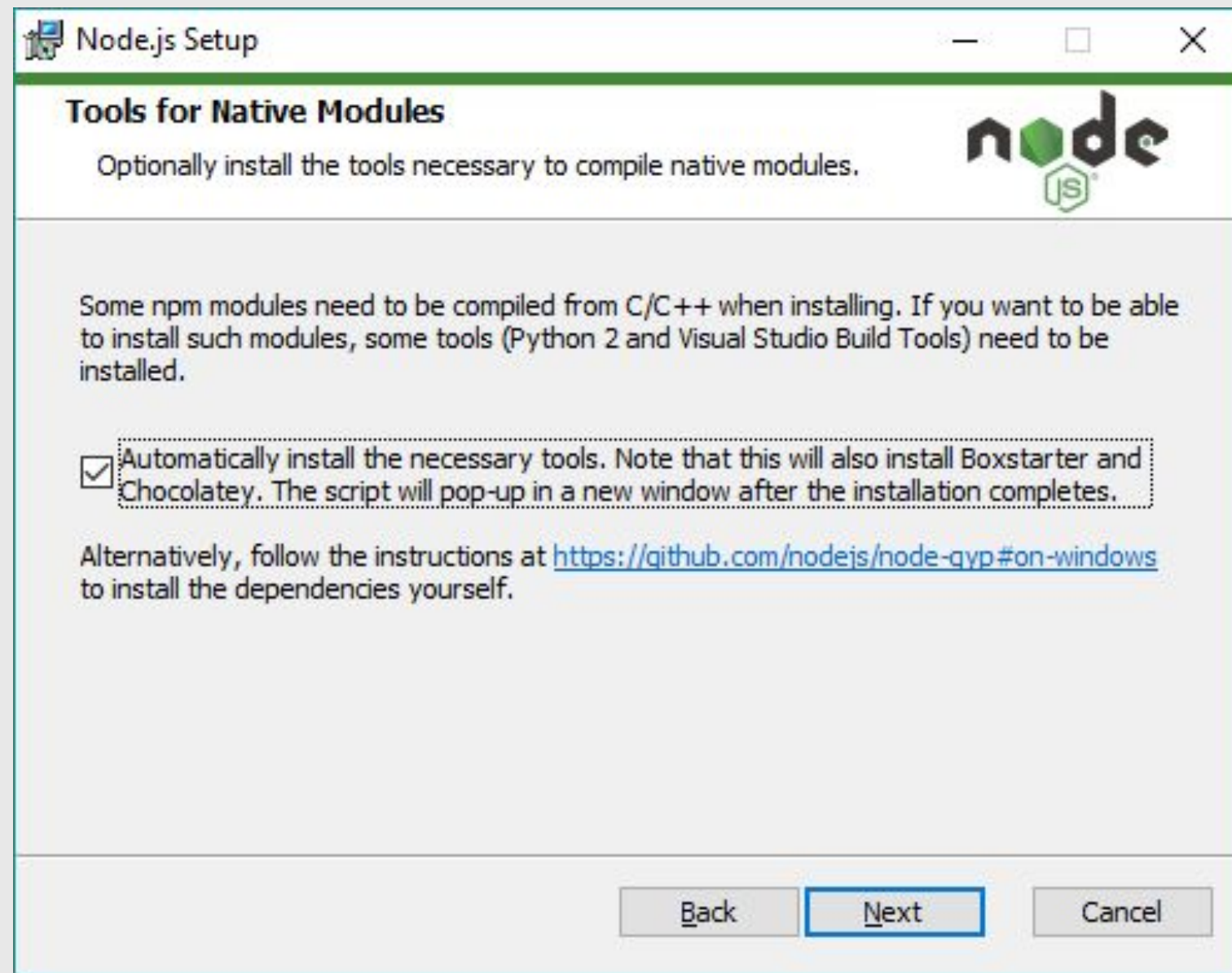


#PraCegoVer: Tela de Instalação
Tela da máquina do Gerador de Conteúdo

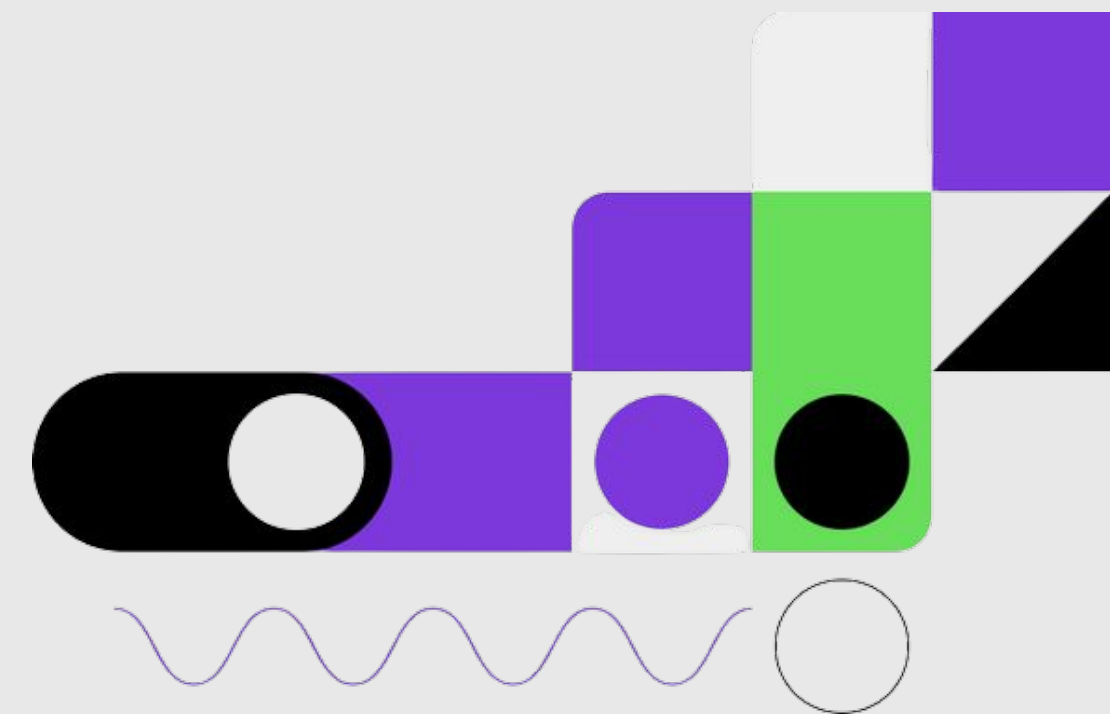


Instalação do NodeJS e NPM no Windows

Na sequência o instalador pergunta se queremos instalar as ferramentas necessárias para compilar módulos nativos. Se você marcar a caixinha para instalar essas ferramentas, o instalador vai iniciar outra janela para instalar essas ferramentas. Eu marquei, escolha se você deseja essa opção e Clique em Next para continuar.

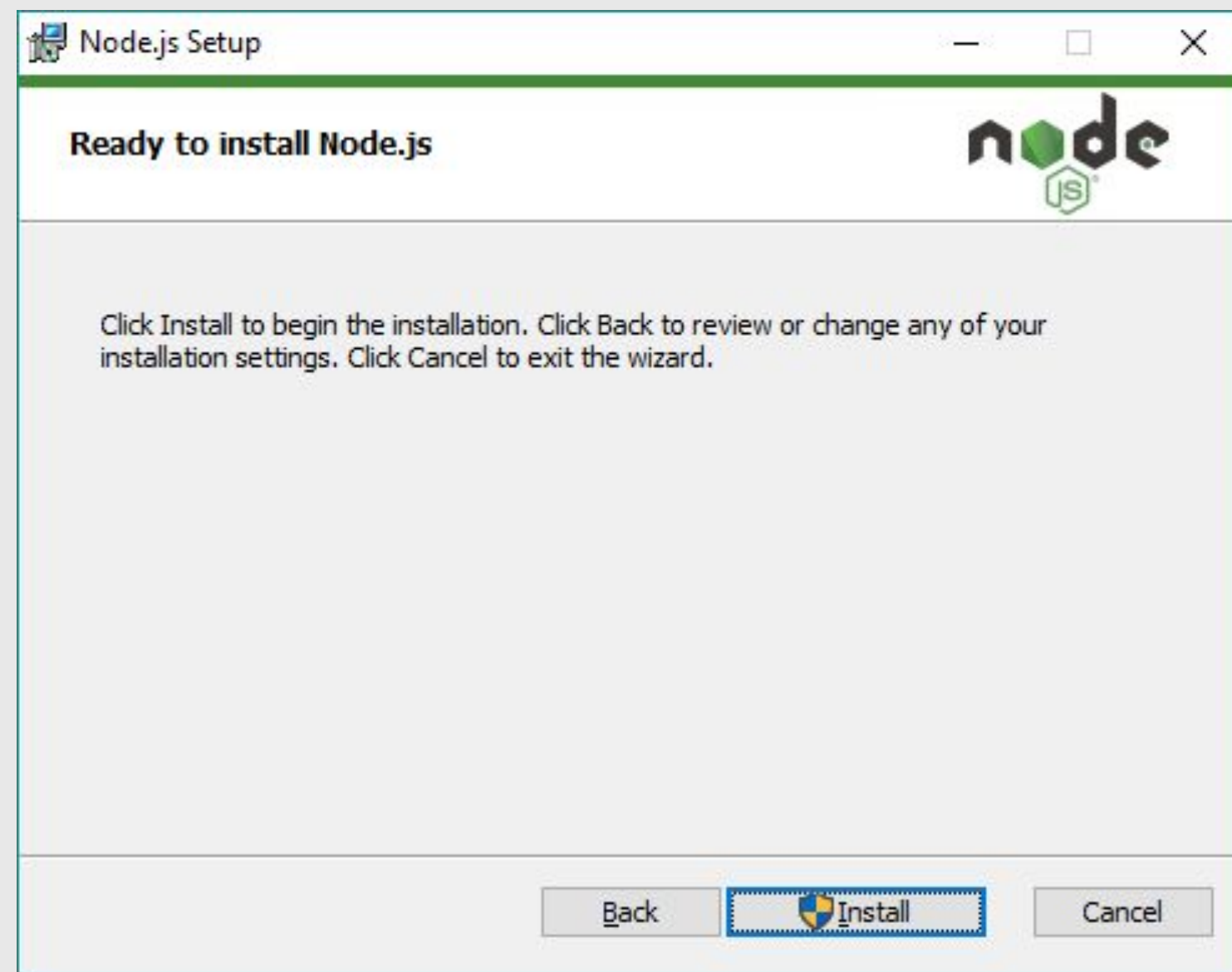


#PraCegoVer: Tela de Instalação
Tela da máquina do Gerador de Conteúdo

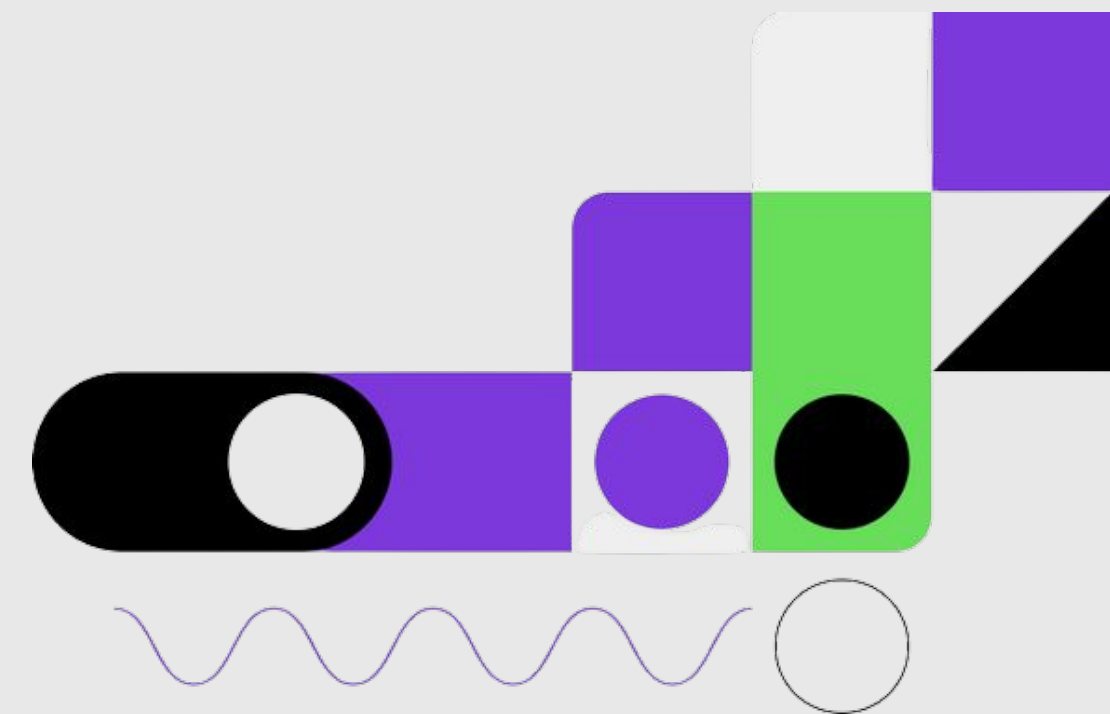


Instalação do NodeJS e NPM no Windows

Agora basta clicar em Install para iniciar a instalação. O Windows vai pedir permissão de administrador para instalar o NodeJS, Terminada a instalação clique em Finish e pronto.



#PraCegoVer: Tela de Instalação
Tela da máquina do Gerador de Conteúdo





Tipos de dado em JavaScript



Por padrão no JavaScript, as instruções são chamadas de declaração e são separadas por um ponto e vírgula (;).

Espaços, tabulação e uma nova linha são chamados de espaços em branco. O código fonte desenvolvido nos scripts em JavaScript são lidos da esquerda para a direita por padrão e são convertidos em uma sequência de elementos de entrada como símbolos, caracteres de controle, terminadores de linha, comentários ou espaço em branco.

Estrutura de Dados

Em JavaScript temos por padrão sete tipo de dados:
Seis tipos de dados são os chamados primitivos:

| Tipo de Dados | Descrição |
|---------------------|--|
| ✓ Boolean. | true e false. |
| ✓ null. | Uma palavra-chave que indica valor nulo. Devido JavaScript ser case-sensitive, null não é o mesmo que Null, NULL, ou ainda outra variação. |
| ✓ undefined. | Uma propriedade superior cujo valor é indefinido. |
| ✓ Number | . 42 ou 3.14159. |
| ✓ String. | "Howdy" |
| ✓ Symbol | (novo em ECMAScript 6). |
| ✓ Object | Um tipo de dado cuja as instâncias são únicas e imutáveis |

JavaScript Boolean

Um JavaScript Boolean representa um de dois valores: true ou false.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Booleans</h2>
<p>Exibe o valor de Boolean (10 > 9):</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
Boolean(10 > 9);
</script>

</body>
</html>
```

JavaScript Booleans

Exibe o valor de Boolean (10 > 9):

true

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela
Tela branca com as frases
JavaScript Booleans, Exibe o valor de Boolean (10 > 9):
como verdadeiro

JavaScript Operadores Lógicos

Operadores lógicos aritméticos

| Operador | Descrição |
|----------|-----------------------|
| + => | SOMA |
| - => | SUBTRAÇÃO |
| / => | DIVISÃO |
| * => | MULTIPLICAÇÃO |
| % => | MÓDULO DE UMA DIVISÃO |
| ** => | POTÊNCIA |

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Operadores</h2>
<h3>Adição</h3>
<p>O operador + concatena (une) strings:
</p>
<p id="demo"></p>
<script>
let text1 = "Bom ";
let text2 = "Dia";
let text3 = text1 + text2;
document.getElementById("demo").innerHTML =
text3;
</script>
</body>
</html>
```

JavaScript Operadores

Adição

O operador + concatena (adiciona) strings:

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela
Tela branca com as frases
JavaScript Operadores
Adição Exibe uma frase concatenado

JavaScript Numbers

JavaScript trabalhando com casas decimais e ponto flutuantes

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Numbers</h2>
<p>A aritmética de ponto flutuante nem sempre é
100% precisa:</p>
<p id="demo1"></p>
<p>Mas ajuda a multiplicar e dividir:</p>
<p id="demo2"></p>
<script>
let x = 0.2 + 0.1;
document.getElementById("demo1").innerHTML = "0.2 +
0.1 = " + x;
let y = (0.2*10 + 0.1*10) / 10;
document.getElementById("demo2").innerHTML = "0.2 +
0.1 = " + y;
</script>
</body>
</html>
```

JavaScript Numbers

A aritmética de ponto flutuante nem sempre é 100% precisa:

$0.2 + 0.1 = 0.30000000000000004$

Mas ajuda a multiplicar e dividir:

$0.2 + 0.1 = 0.3$

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela
Tela branca com as frases JavaScript Numbers, Exibe o valor A aritmética de ponto flutuante nem sempre é 100% precisa e seus resultados.

JavaScript Number

JavaScript tem apenas um tipo de número. Os números podem ser escritos com ou sem decimais.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Numbers</h2>

<p>Os números podem ser escritos com ou sem
decimais:</p>

<p id="demo"></p>

<script>
let x = 3.14;
let y = 3;
document.getElementById("demo").innerHTML = x
+ "<br>" + y;
</script>

</body>
</html>
```

JavaScript Numbers

Os números podem ser escritos com ou sem decimais:

3.14
3

#PraCegoVer: Imagem mostrando
como o JavaScript aparece em tela
Tela branca com as frases JavaScript
Numbers, Exibe Os números podem
ser escritos com ou sem decimais:
3:14
3

JavaScript String

O tipo de dados string em JavaScript, é utilizado para armazenar e manipular texto.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript String</h2>

<p id="demo"></p>

<script>
let text = "Ricardo Alexandre Bontempo";
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

JavaScript String

Ricardo Alexandre Bontempo

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela
Tela branca com o nome Ricardo Alexandre Bontempo

JavaScript Operadores Lógicos

Os operadores lógicos são importantes no JavaScript porque permitem comparar variáveis e fazer algo com base no resultado dessa comparação.

O JavaScript fornece três operadores lógicos

| Operador | Descrição | em Inglês |
|----------|---------------------|---------------|
| ! | Operador lógico NÃO | (Logical NOT) |
| | Operador lógico OU | (Logical OR) |
| && | Operador lógico E | (Logical AND) |

JavaScript Operadores Relacionais

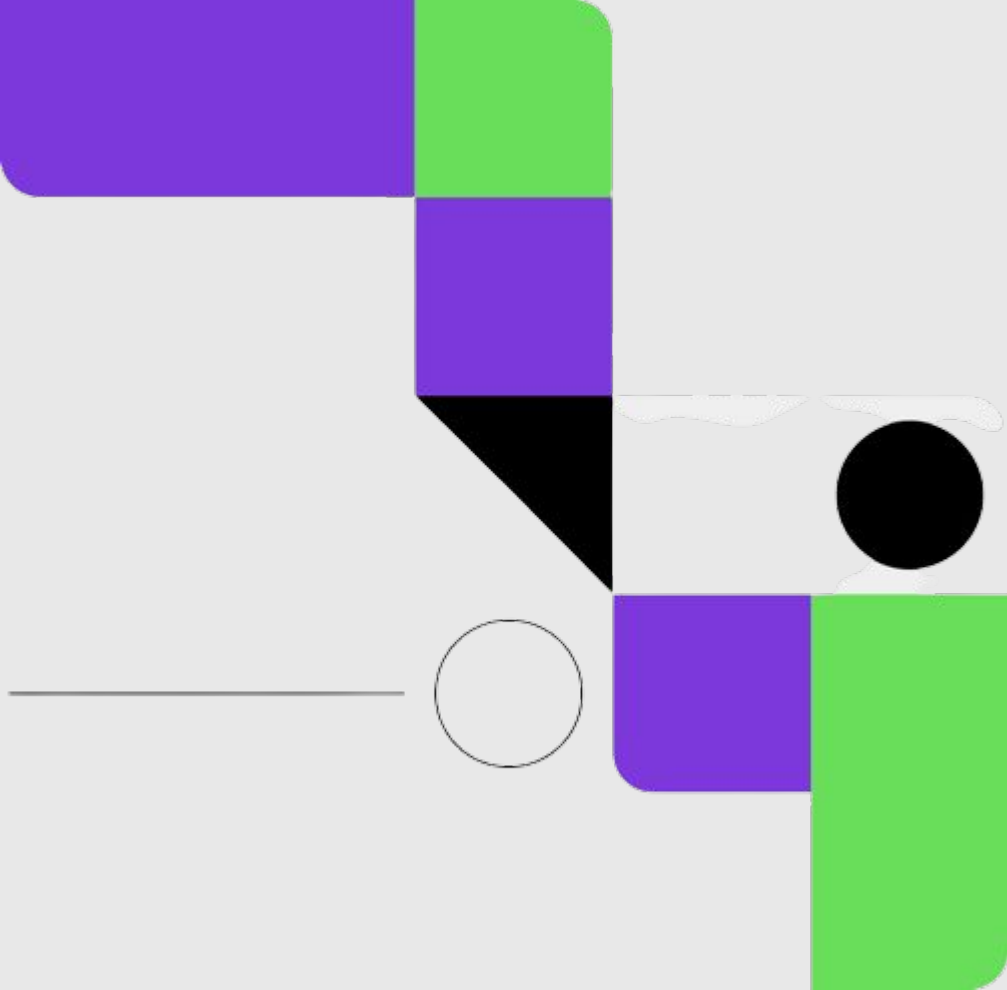
Os operadores relacionais, em JavaScript possui tanto operadores binários quanto unários e um operador ternário e o operador condicional. Um operador binário exige dois operandos, um antes do operador e outro depois: `operando_1 operador operando_2`. Por exemplo, `5+4` ou `x*y`.

| Operadores | Descritivo |
|--------------------------|----------------|
| <code>> =></code> | maior |
| <code>< =></code> | menor |
| <code>>= =></code> | maior ou igual |
| <code><= =></code> | menor ou igual |
| <code>!= =></code> | diferente |
| <code>== =></code> | igualdade |

JavaScript Operadores de comparação

Os tipos de operadores de operações em JavaScript

| Operador | Descrição | Comparação | Retorno |
|----------|---------------------------------------|------------|------------|
| == | igual a | x == 8 | FALSO |
| | | x == 5 | VERDADEIRO |
| | | x == "5" | VERDADEIRO |
| === | igual valor e igual tipo | x === 5 | VERDADEIRO |
| | | x === "5" | FALSO |
| != | diferente de | x != 8 | VERDADEIRO |
| !== | valor diferente ou diferente do tipo | x !== 5 | FALSO |
| | | x !== "5" | VERDADEIRO |
| | | x !== 8 | VERDADEIRO |
| > | maior que | x > 8 | FALSO |
| < | menor que | x < 8 | VERDADEIRO |
| >= | greater than or equal tmaior ou igual | x >= 8 | FALSO |
| <= | menor ou igual | x <= 8 | VERDADEIRO |



JavaScript exemplo de Operadores de comparação

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Comparação</h2>
<p>Atribua 5 a x e exiba o valor da comparação (x == 8):</p>
<p id="demo"></p>
<script>
  let x = 5;
  document.getElementById("demo").innerHTML = (x == 8);
</script>
</body>
</html>
```

JavaScript Comparação

Atribua 5 a x e exiba o valor da comparação (x == 8):

false

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela Tela branca com as frases JavaScript Comparação, Exibe Atribua 5 a x e exiba o valor da comparação (x == 8)

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Comparação</h2>
<p>Atribua 5 a x e exiba o valor da comparação (x === 5):</p>
<p id="demo"></p>
<script>
  let x = 5;
  document.getElementById("demo").innerHTML = (x === 5);
</script>
</body>
</html>
```

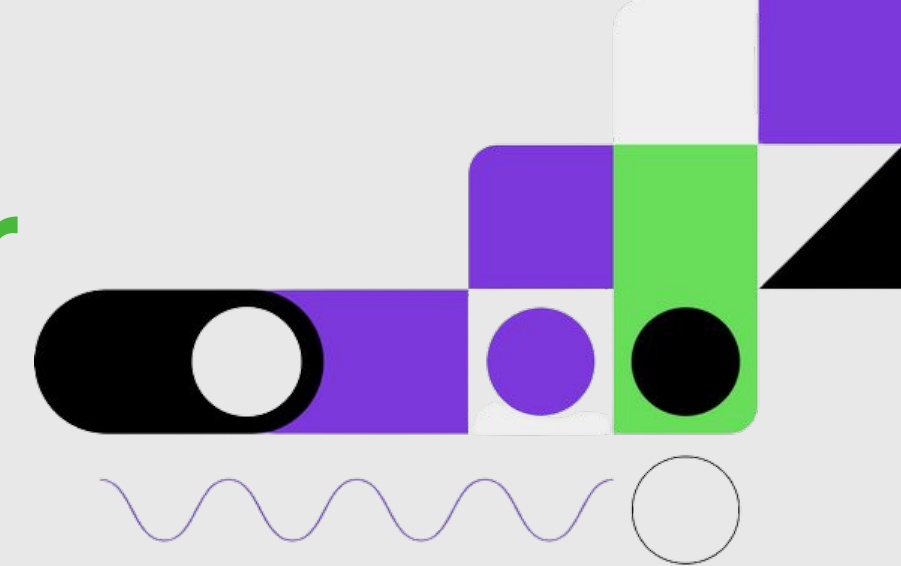
JavaScript Comparação

Atribua 5 a x e exiba o valor da comparação (x === 5):

true

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela Tela branca com as frases JavaScript Comparação, Exibe atribua 5 a x e exiba o valor da comparação (x === 5)

Estrutura de programação: Variáveis var



O que são Variáveis? Variáveis são contêineres para armazenar dados (armazenando valores de dados). Neste exemplo, x, y e z são variáveis, declaradas com a palavra-chave var:

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Variáveis</h1>
<p>Neste exemplo, x, y e z são variáveis.</p>
<p id="demo"></p>
<script>
  var x = 5;
  var y = 6;
  var z = x + y;
  document.getElementById("demo").innerHTML =
    "O valor de z é: " + z;
</script>
</body>
</html>
```

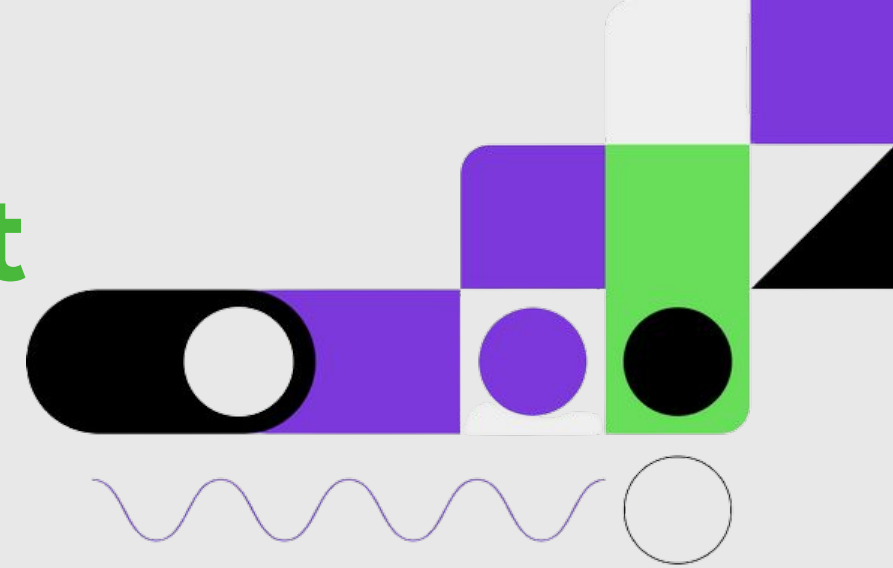
JavaScript Variables

Neste exemplo, x, y e z são variáveis.

O valor de z é: 11

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com as frases JavaScript Variables, Neste exemplo, x, y e z são variáveis e o resultado 11.

Estrutura de programação: Variáveis left



Variáveis definidas com `let` não podem ser redeclaradas. Você não pode redeclarar acidentalmente uma variável.

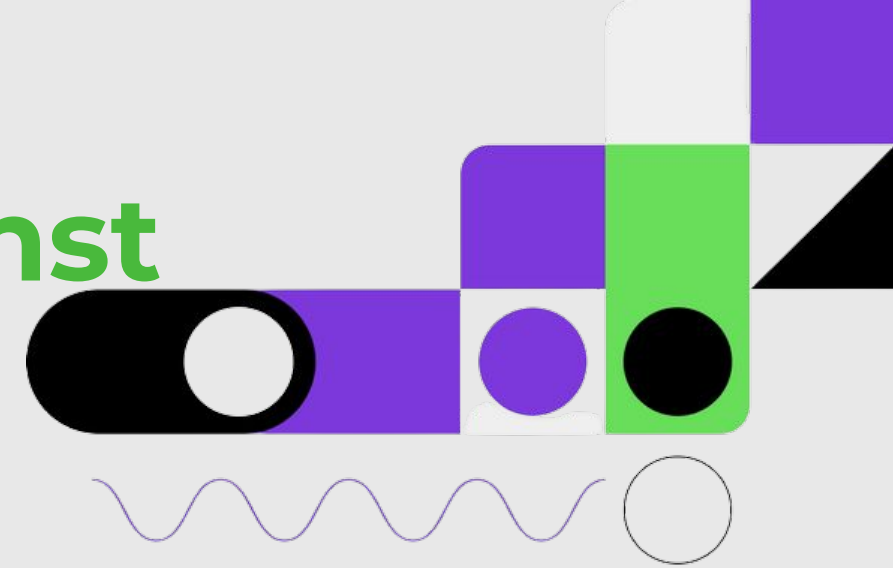
```
<!DOCTYPE html>
<html>
<body>
<h2>Redeclarando uma variável usando let </h2>
<p id="demo"></p>
<script>
let x = 10;
{
  let x = 2;
}
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

Redeclarando uma variável usando let

10

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com a frase Redeclarando uma variável usando let. Neste exemplo, retornamos o valor 10.

Estrutura de programação: Variáveis const



A declaração `const` permite criar uma variável cujo o valor é fixo, ou seja, uma constante somente leitura. Isso não quer dizer que o valor é imutável, ou apenas que a variável constante não pode ser alterada ou retribuída.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript const</h2>
<p id="demo"></p>
<script>
try {
  const PI = 3.141592653589793;
  PI = 3.14;
}
catch (err) {
  document.getElementById("demo").innerHTML = err;
}
</script>
</body>
</html>
```

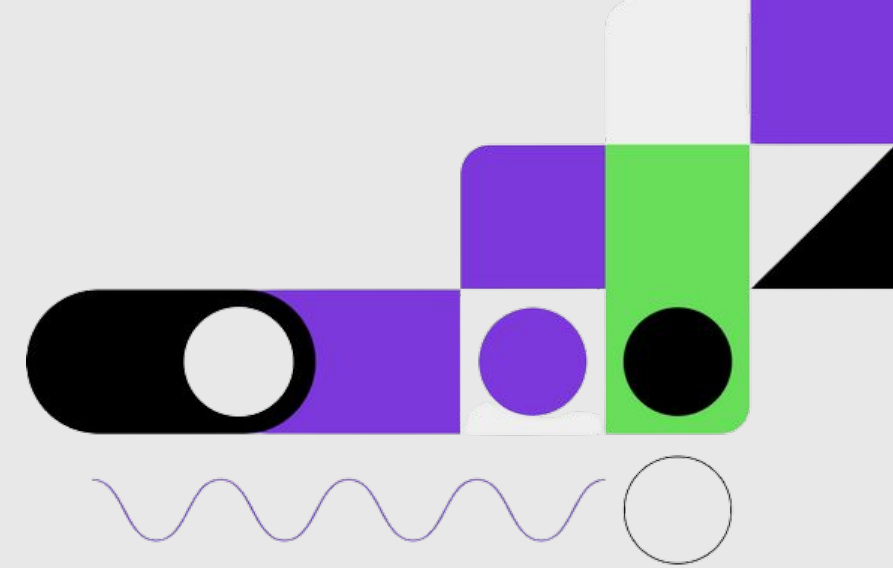
Palavras reservadas do JavaScript

As palavras reservadas, também conhecidas como palavras chaves da linguagem, Estas palavras não podem ser usadas com outro propósito não podemos usar essas palavras reservadas como variáveis, labels ou nomes de função:



| | | | |
|----------|------------|--------------|-----------|
| abstract | arguments | await | boolean |
| break | byte | case | catch |
| char | class | const | continue |
| debugger | default | delete | do |
| double | else | enum | eval |
| export | extends | false | final |
| finally | float | for | function |
| goto | if | implements | import |
| in | instanceof | int | interface |
| let | long | native | new |
| null | package | private | protected |
| public | return | short | static |
| super | switch | synchronized | this |
| throw | throws | transient | true |
| try | typeof | var | void |
| volatile | while | with | yield |

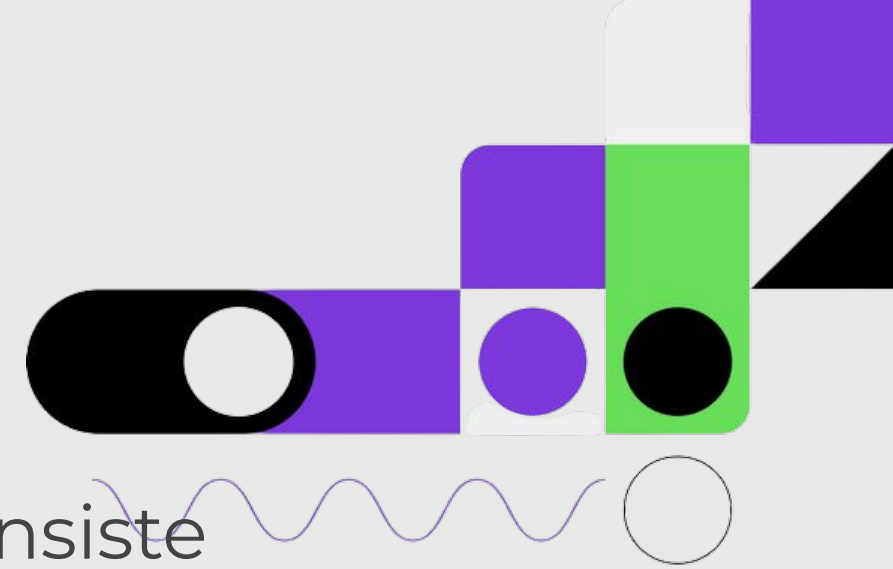
Palavras reservadas do JavaScript



Devemos evitar também usar o nome de objetos, propriedades e métodos integrados do JavaScript.

| | | | |
|----------------|----------|-----------|-----------|
| Array | Date | eval | function |
| hasOwnProperty | Infinity | isFinite | isNaN |
| isPrototypeOf | length | Math | NaN |
| name | Number | Object | prototype |
| String | toString | Undefined | valueOf |

Declarando Funções em JavaScript



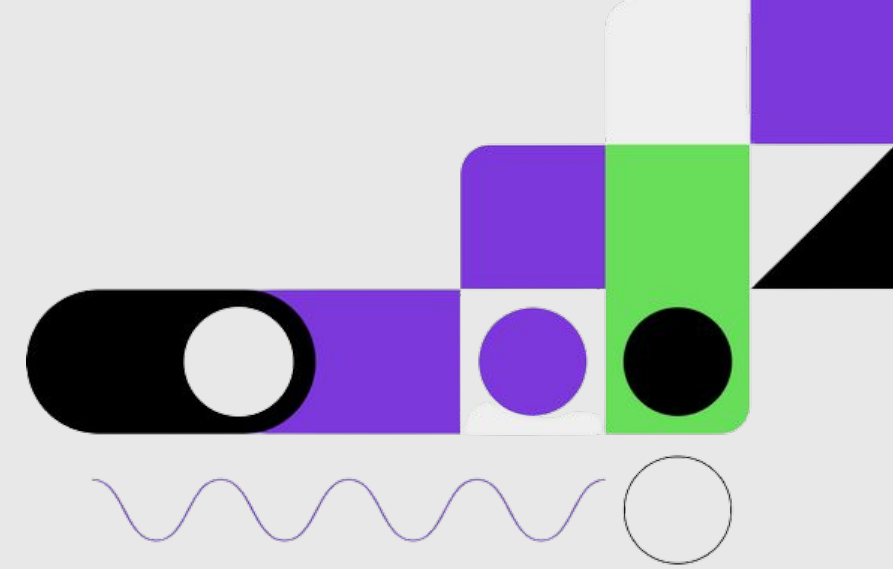
A definição da função (também chamada de declaração de função) consiste no uso da palavra chave function (en-US), seguida por:

- Nome da Função.
- Lista de argumentos para a função, entre parênteses e separados por vírgulas.
- Declarações JavaScript que definem a função, entre chaves { }.

OBS: Veja que dentro da função temos o return, ou seja o que a função calcular irá retornar como resposta
Outra dica função sem nada digitado nos parenteses é sem parâmetros, com conteúdos a função recebe valores

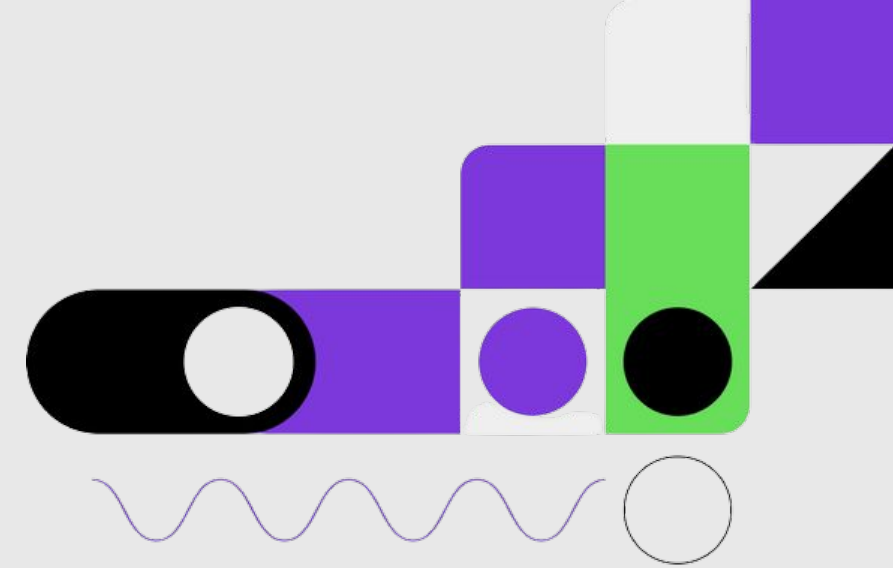
```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Functions</h2>
<p>Este exemplo chama uma função que realiza um
cálculo e retorna o resultado:</p>
<p id="demo"></p>
<script>
// Função de retorno do cálculo
function myFunction(p1, p2) {
  return p1 * p2;
}
document.getElementById("demo").innerHTML =
myFunction(4, 3);
</script>
</body>
</html>
```

Estrutura de repetição JavaScript



As estruturas de repetição, são úteis quando precisamos repetir N vezes a execução de um bloco de comandos até que uma condição seja atendida.
Caso você não esteja familiarizado com os conceitos de estruturas condicionais e estruturas de repetição.

Estrutura de repetição For JavaScript



Os loops são úteis, se você quiser executar o mesmo código várias vezes, cada vez com um valor diferente.

String do comando for

```
for (let i = 0 ; i <= 9 ; i++){  
    bloco que será executado...  
}
```

```
<!DOCTYPE html>  
<html>  
<body>  
<h2>JavaScript For Loop</h2>  
<p id="demo"></p>  
<script>  
const cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat",  
"Audi"];  
let text = "";  
for (let i = 0; i < cars.length; i++) {  
    text += cars[i] + "<br>";  
}  
document.getElementById("demo").innerHTML = text;  
</script>  
</body>  
</html>
```

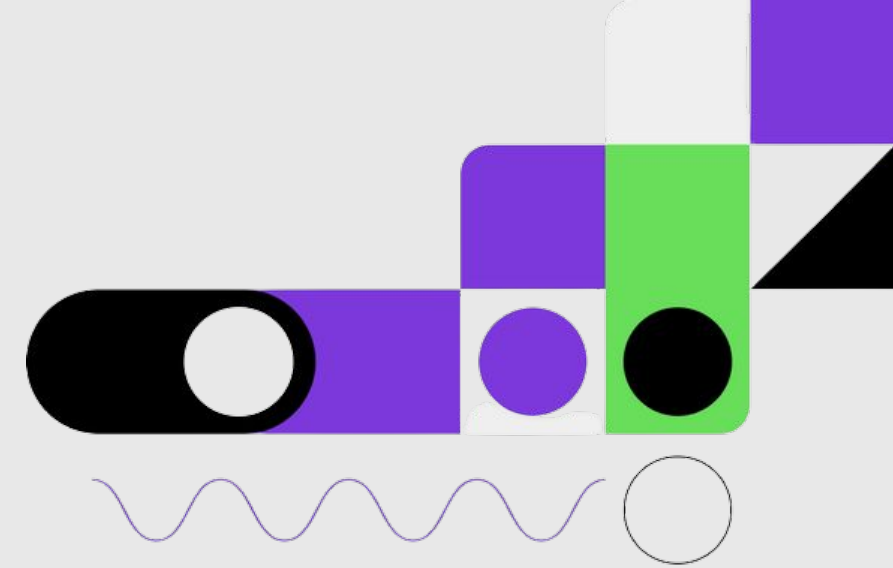
JavaScript For Loop

BMW
Volvo
Saab
Ford
Fiat
Audi

#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela. Tela branca com as frases JavaScript For Loop. Neste exemplo, serão mostrados várias marcas:

BMW
Volvo
Saab
Ford
Fiat

Estrutura de repetição While JavaScript



O loop while percorre um bloco de código enquanto uma condição especificada for verdadeira.

String do comando while

```
while (condicionamento) {  
    bloco que será executado...  
    pode ter vários comandos...  
}
```

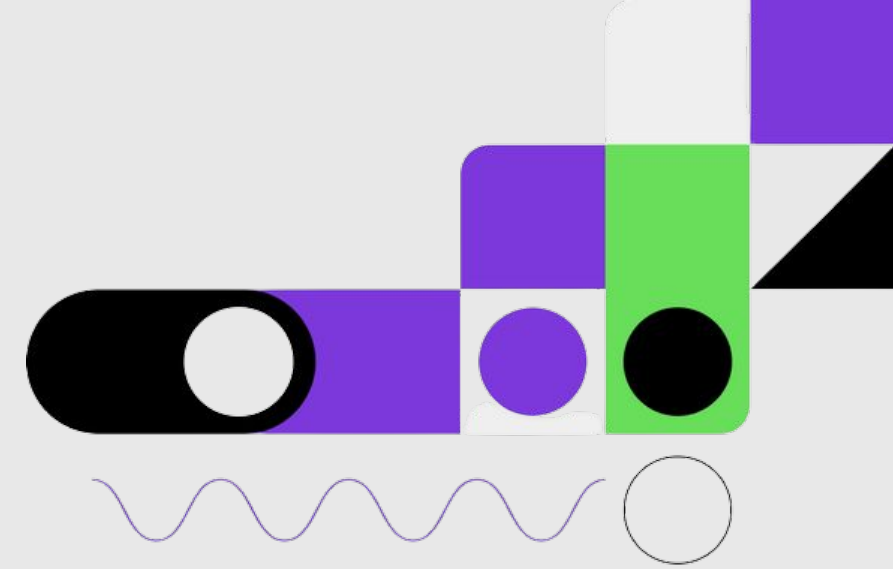
```
<!DOCTYPE html>  
<html>  
<body>  
<h2>JavaScript While Loop</h2>  
<p id="demo"></p>  
<script>  
let text = "";  
let i = 0;  
while (i < 10) {  
    text += "<br>O Número é " + i;  
    i++;  
}  
document.getElementById("demo").innerHTML = text;  
</script>  
</body>  
</html>
```

JavaScript While Loop

O Número é 0
O Número é 1
O Número é 2
O Número é 3
O Número é 4
O Número é 5
O Número é 6
O Número é 7
O Número é 8
O Número é 9

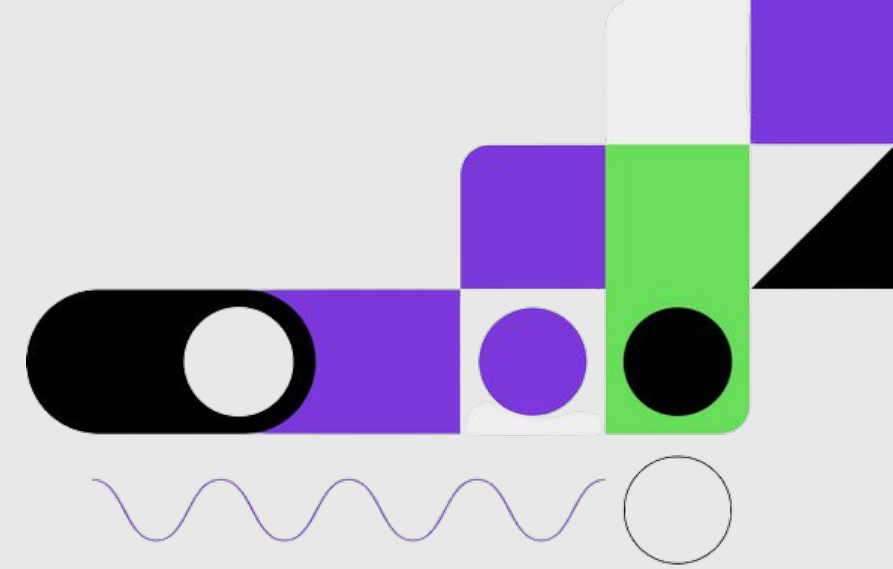
#PraCegoVer: Imagem mostrando como o JavaScript aparece em tela
Tela branca com as frases
JavaScript While Loop
Neste exemplo, O número é 0 até 9.

Estrutura condicionais JavaScript



As estruturas condicionais em JavaScript, estão ligadas à tomada de decisão de um algoritmo. Ao utilizar este modelo de expressões que retornam verdadeiro ou falso, o algoritmo executa o bloco de comandos relativos a este resultado.

Estrutura condicional if JavaScript



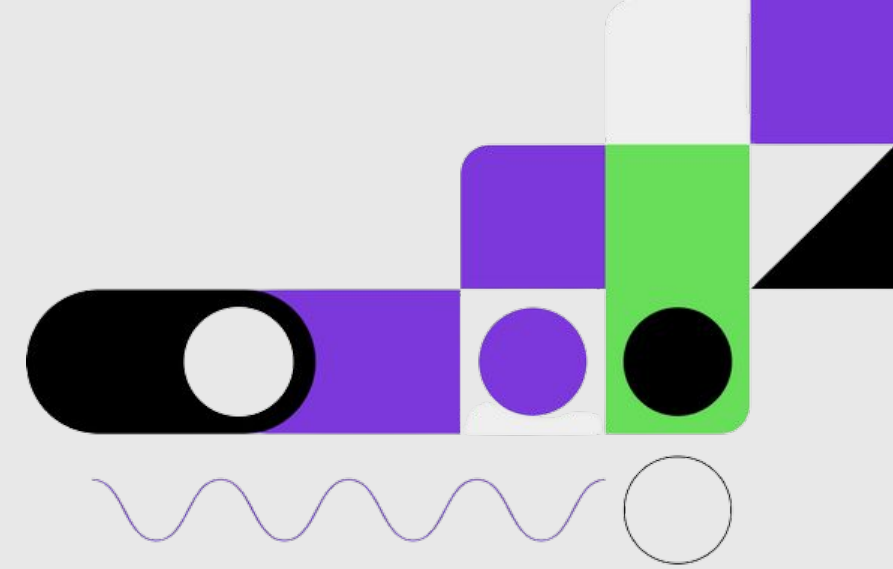
A condicional if é uma estrutura condicional que executa a afirmação, dentro do bloco, se determinada condição for verdadeira. Se for falsa, executa as afirmações dentro de else.

String do comando if

```
if (teste lógico){  
    condição verdadeira  
}  
else {  
    condição falsa  
}
```

```
<script>  
    function verificaNumero(){  
        var numero=prompt("Digite um número");  
        var resultado=parseInt(numero)%2;  
        // Condição para analisar se o número é par ou  
        impar  
        if (parseInt(resultado)==0){  
            alert("Número é par");  
        }else{  
            alert("Número é Impar");  
        }  
    }  
</script>
```

Objetos e Arrays no JavaScript



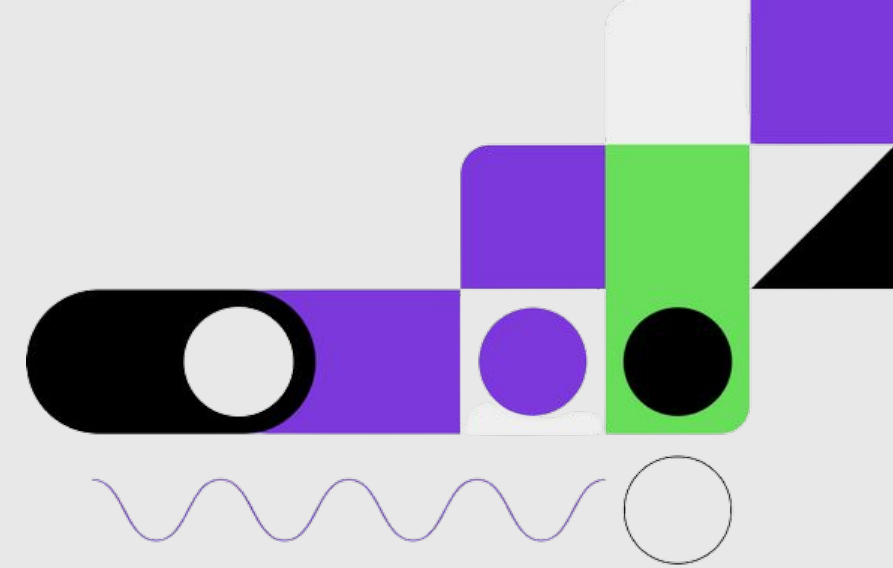
A definição de um Array no JavaScript é um objeto com um construtor único, com uma sintaxe literal e com um conjunto adicional de propriedades e de métodos herdados de um protótipo de Array. Arrays.

Normalmente são usados para armazenar vários valores em uma única variável, ou seja um Array é uma variável especial, que pode conter mais de um valor por vez.

Por exemplo se você criar uma lista de itens com nomes de carros, armazenar os carros em uma única variável.

```
var carros = [ "HB20", "Sander",  
               "Gol", "Onix" ];
```

Como Acessar os elementos de um Array JavaScript



Para você acessar um elemento do Array referente ao número de índice. Esta declaração acessa o valor do primeiro elemento em carros:

```
var name = carros[0];
```

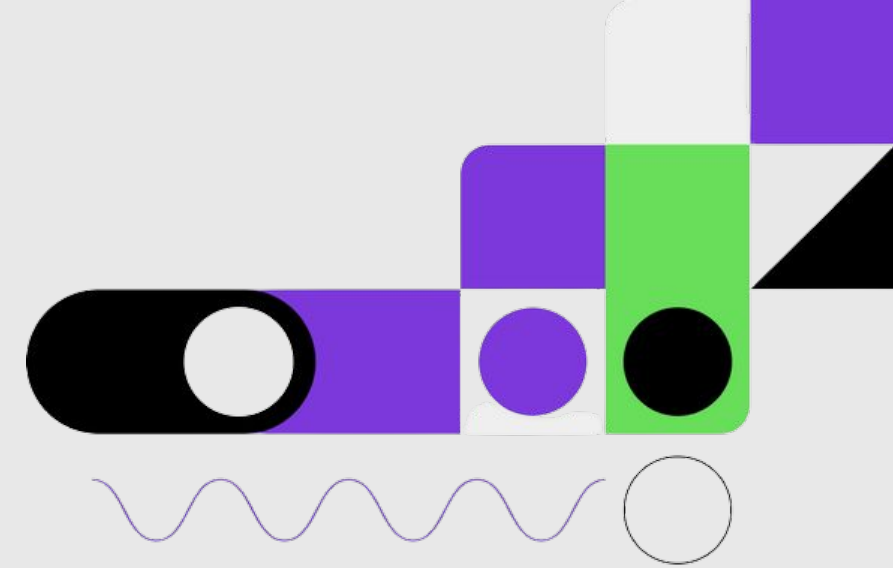
A declaração abaixo, modifica o primeiro elemento, registro na matriz carros

```
carros[0] = "Voyage";
```

Exemplo

```
var carros = [ "HB20", "Sander", "Gol", "Onix" ];  
document.getElementById("demo").innerHTML = carros[0];
```

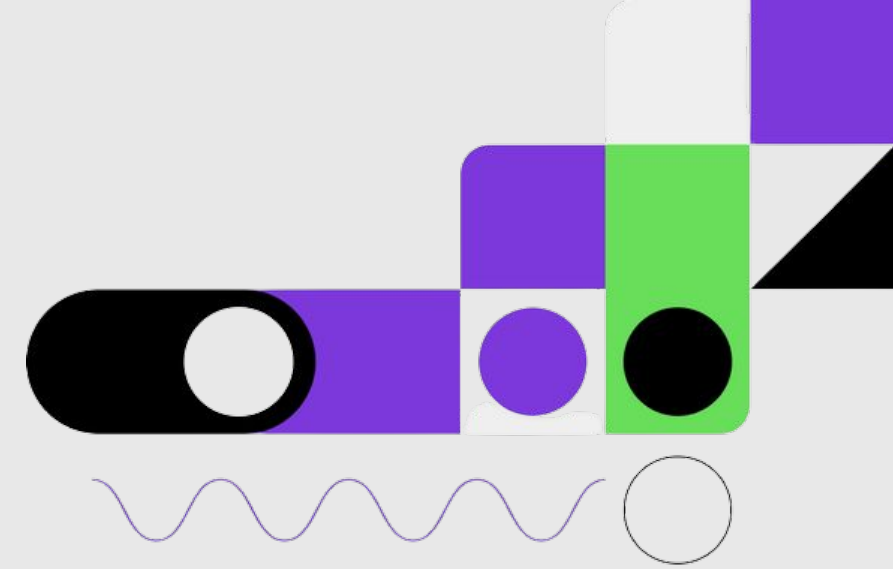

Como acessar um Array completo Em JavaScript



Para se acessar a matriz completa consultando pelo o nome do Array:

```
<script>  
// Validação dos valores inseridos dentro da variável matriz  
var carros = [ "HB20", "Sander", "Gol", "Onix" ];  
document.getElementById("demo").innerHTML = carros;  
</script>
```

Como trabalhar com os objetos Array Em JavaScript



Os Arrays são um tipo especial de objetos, por padrão o tipo de operador em JavaScript retorna "objeto" para arrays.

Todavia os arrays de JavaScript são melhor descritos como arrays.

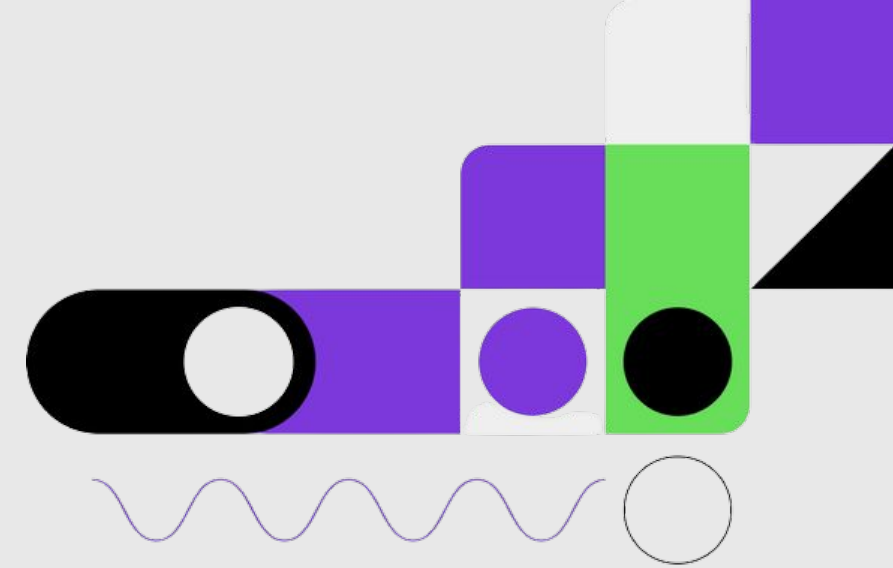
Os Arrays usam números para controlar e acessar seus "elementos". Neste exemplo, a `pessoa[0]` retorna Ricardo:

```
var pessoa = [ "Ricardo", "Bontempo", 47 ];
```

Por padrão, os objetos usam nomes para acessar seus "membros". Neste exemplo abaixo, `pessoa.primeiroNome` retorna Ricardo:

```
var pessoa = {primeiroNome:"Ricardo", lastName:"Bontempo", age:47};
```

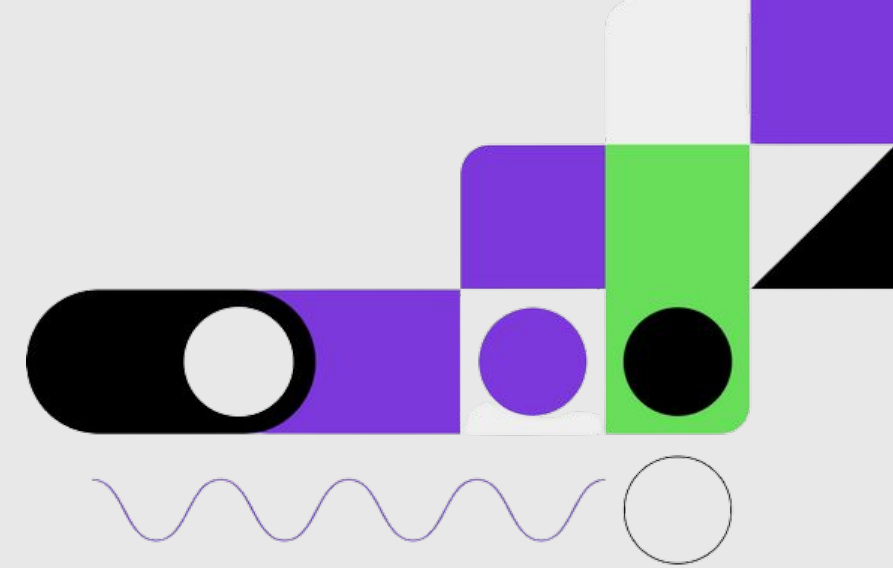
Como trabalhar com os objetos Array Em JavaScript



Como vimos anteriormente, as variáveis de JavaScript podem ser objetos. Os Arrays são tipos especiais de objetos. Por isso, você pode ter inúmeras variáveis de diferentes tipos no mesmo Array, como demonstrado no exemplo abaixo:

```
myArray[0] = Date.now;  
myArray[1] = myFunction;  
myArray[2] = mycarros;
```

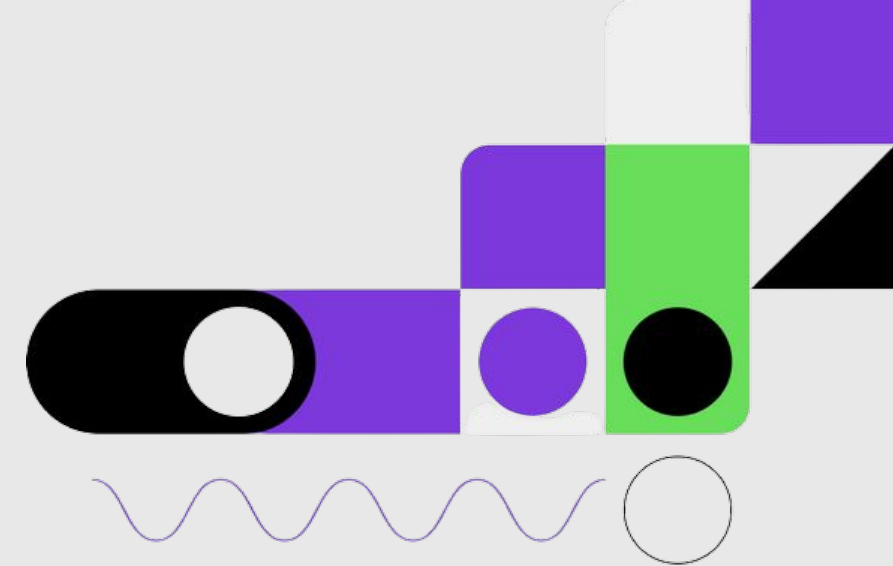
Como analisar as propriedades de um Array Em JavaScript



As propriedades nos Arrays de JavaScript vem incorporadas nos métodos do Array :

```
var x = carros.length; // A propriedade length retorna o número de elementos  
var y = carros.sort(); // O método sort() classifica os arrays
```

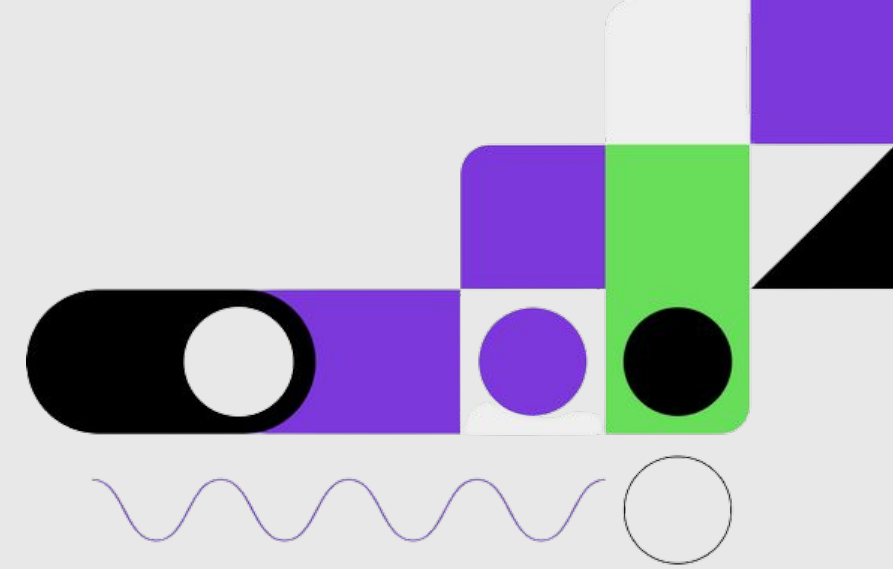
Utilizando a propriedade length no Array Em JavaScript



A propriedade length no Array retorna o comprimento de um Array (o número de elementos do Array), ou se você preferir os registros.

```
// Contexto abaixo mostra os registros atribuídos a matriz  
var carros = ["Uno", "Gol", "Ka", "HB20"];  
carros.length; // o length de carros é 4
```

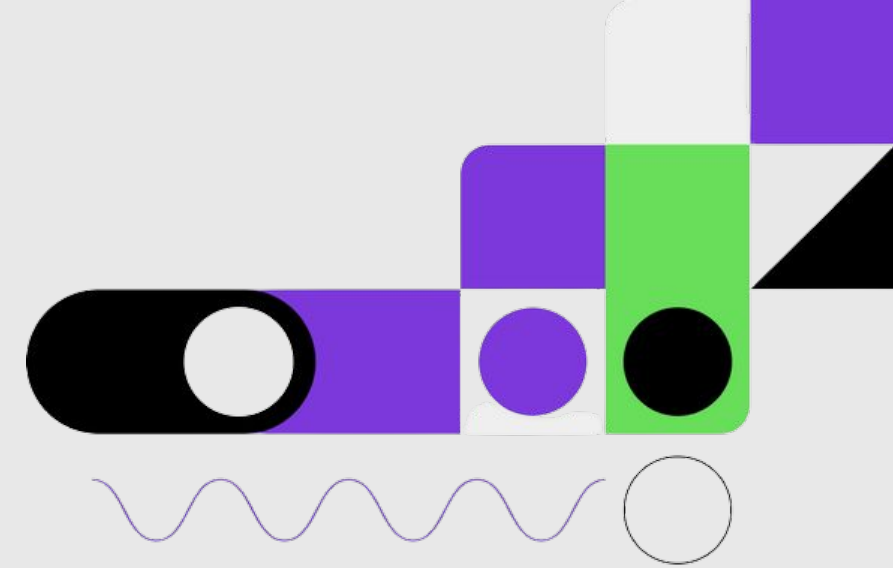
Interagindo com um Array Em JavaScript



A melhor maneira de listar, percorrer um Array é usando um loop como por exemplo o "for":

```
// Declaração das variáveis
var carros, text, cLen, i;
carros = ["Uno", "Gol", "Ka", "HB20"];
// Identificação dos registros na variável
cLen = carros.length; text = "<ul>";
# Loop de repetição do processo
for (i = 0; i < cLen; i++)
{
text += "<li>" + carros[i] + "</li>";
}
```


Inserindo registro em um Array Em JavaScript



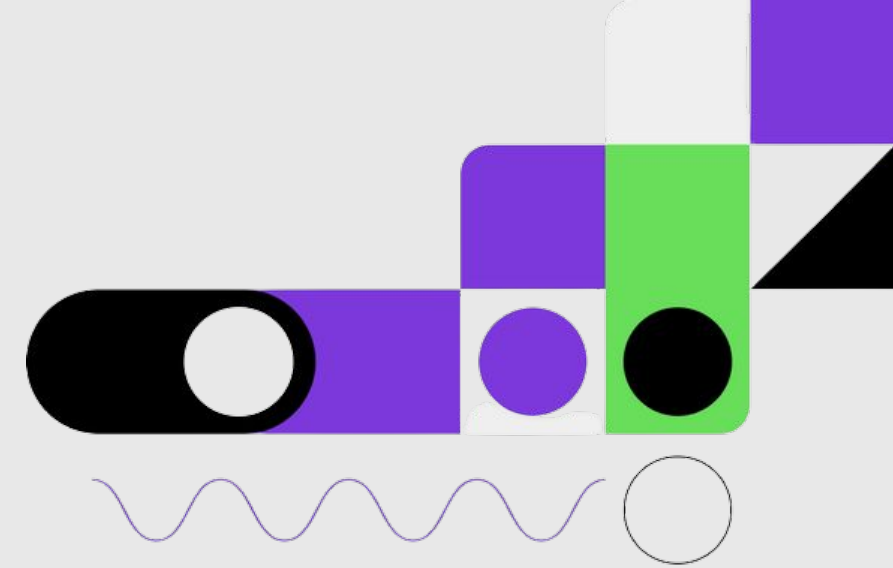
A forma mais fácil de adicionar um novo elemento em uma variável do tipo Array é usando o método push:

```
var carros = ["Uno", "Gol", "Ka", "HB20"];  
carros.push("Sander");  
// acrescenta o novo elemento (Sander) na variável carro
```

Além disso, novos elementos também podem ser inseridos a um array usando a propriedade length, que vimos anteriormente.

```
var carros = ["Uno", "Gol", "Ka", "HB20"];  
carros[carros.length] = "Sander";  
// Adiciona o novo elemento (Limão) a frutas
```

Como fazer o reconhecimento de uma variável do tipo Array Em JavaScript

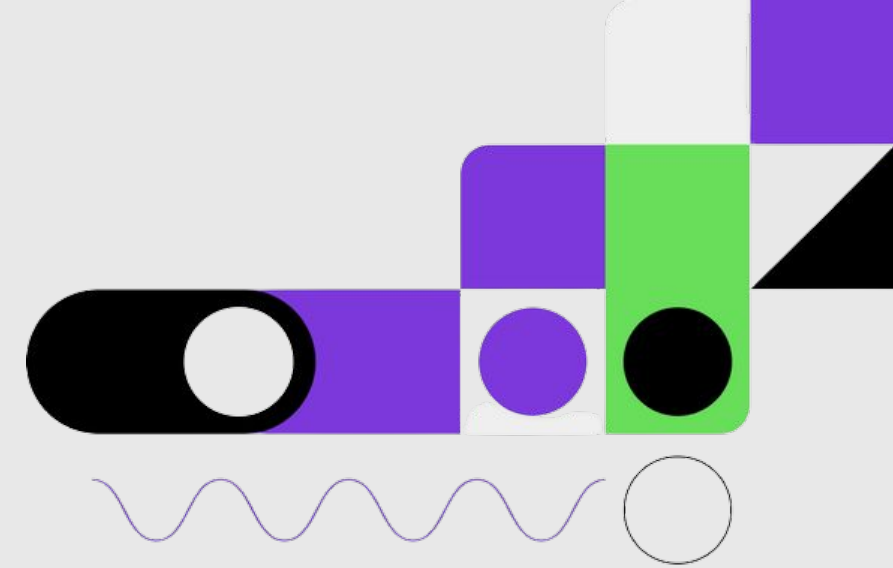


Normalmente quando programamos em JavaScript surge a seguinte pergunta comum: Como eu sei se uma variável é um Array?

O problema é que o operador JavaScript `typeof` retorna o "objeto", pois o Array é um objeto, Exemplo:

```
var carros = ["Uno", "Gol", "Ka", "HB20"];  
typeof carros; // retorna o object
```

Como criar uma função para reconhecer um variável do tipo Array Em JavaScript



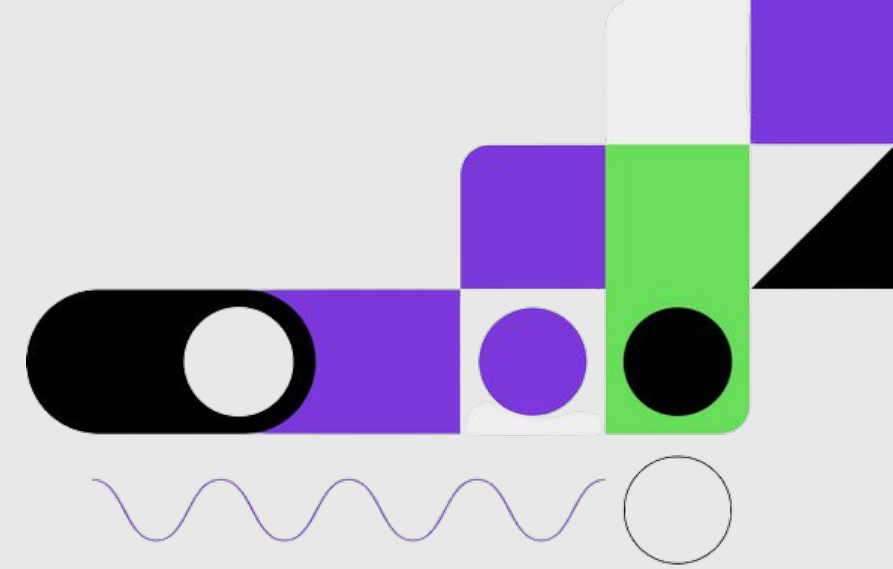
Podemos desenvolver uma função simples, utilizando a própria função isArray()

```
function isArray(x) {  
  return x.constructor.toString().indexOf("Array") > -1;  
}
```

A função criada acima, sempre retorna verdadeiro se o argumento for um Array. Ou mais precisamente: ele retorna se o protótipo do objeto contiver a palavra "Array". O instanceof operador retorna verdadeiro se um objeto for criado por um determinado construtor:

```
var carros = ["Uno", "Gol", "Ka", "HB20"];  
carros instanceof Array  
  
// retorna true
```

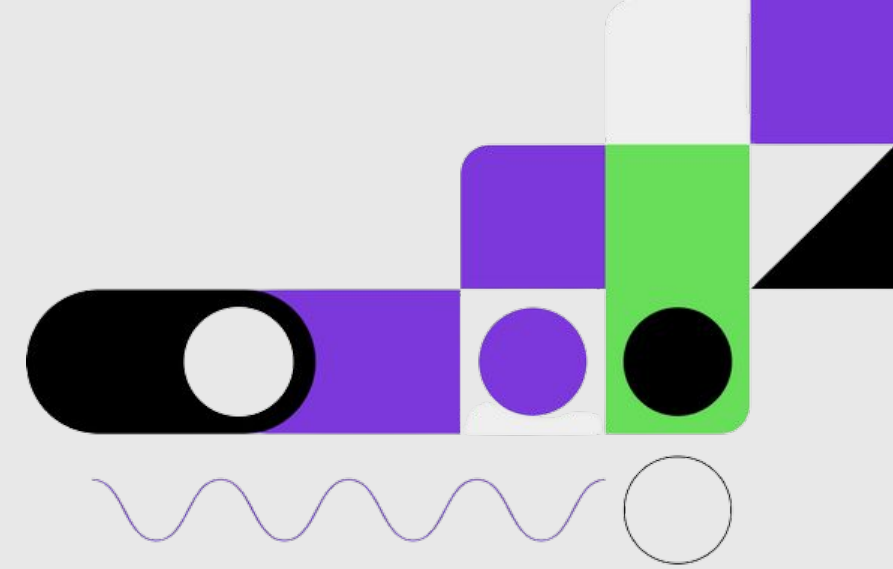
Criando uma Deseestruturação de Array Em JavaScript



```
var numeros = ["um", "dois", "tres"];  
var [um, dois, tres] = numeros;  
  console.log(um); // "um"  
  console.log(dois); // "dois"  
  console.log(tres); // "tres"
```

No exemplo acima criamos uma atribuição padrão de valores para o array.

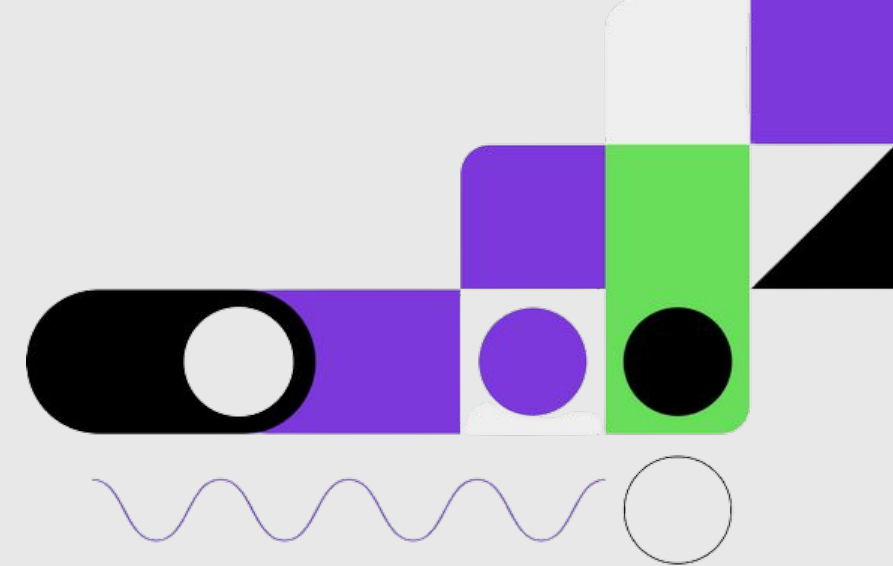
Criando uma Desestruturação de Array Em JavaScript



```
var numeros = ["um", "dois", "tres"];  
var [um, dois, tres] = numeros;  
  console.log(um); // "um"  
  console.log(dois); // "dois"  
  console.log(tres); // "tres"
```

No exemplo acima, criamos uma atribuição padrão de valores para o array.

Criando uma Desestruturação de Array Em JavaScript



```
var a, b;  
  
[a, b] = [1, 2];  
console.log(a); // 1  
console.log(b); // 2
```

No exemplo acima, criamos uma atribuição via desestruturação separadamente da declaração dela

O que é JSON?



JSON ou JavaScript Object Notation é um formato leve de troca de informações/dados entre sistemas.

O JSON além de ser um formato leve é muito simples de ler. Mas quando dizemos que algo é simples, precisamos ter cuidado em compará-lo com algo mais complexo para entendermos tal simplicidade

Para ficar mais claro o nosso exemplo, podemos comparar o JSON com o formato XML.

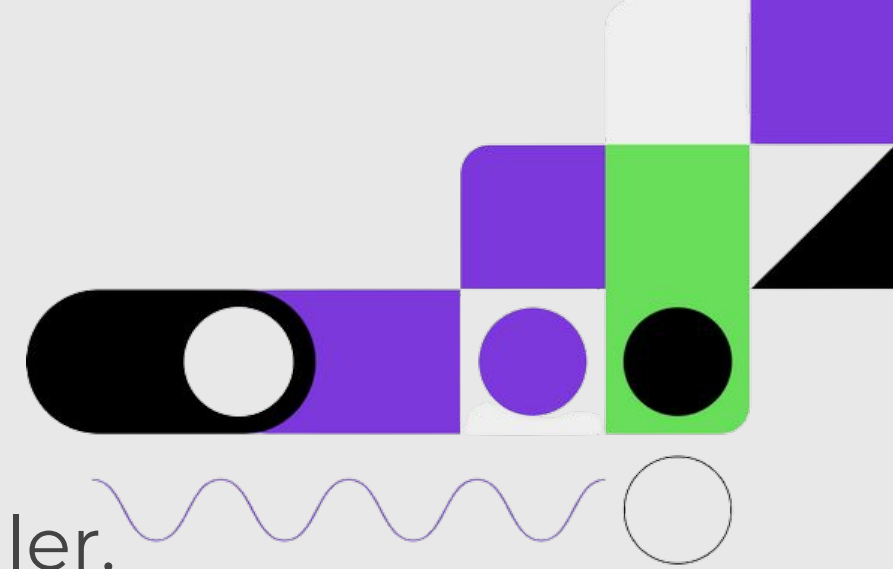
Exemplo em XML

```
<note>
<to>Ricardo</to>
<from>Luciana</from>
<heading>Lembrete</heading>
<body>Não se esqueça de mim neste fim de semana!
</body>
</note>
```

Exemplo em Json

```
{
  "id":1,
  "nome":"Ricardo Alexandre",
  "endereco":"Rua Catão"
}
```

Vantagens do JSON



A diferença visual é nítida, o segundo trecho (em JSON) é mais fácil de ler.

- ❑ Leitura mais simples
- ❑ JSON suporta objetos! Sim, ele é tipado!
- ❑ Quem utiliza? Google, Facebook, Yahoo!, Twitter...
- ❑ Velocidade maior na execução e transporte de dados
- ❑ Analisador(parsing) mais fácil
- ❑ Arquivo com tamanho reduzido

A seguir vamos ver um exemplo de JSON para Web

Vantagens do JSON



Existem várias bibliotecas para trabalharmos com JSON. No nosso estudo o json.jar que iremos utilizar

Neste nosso exemplo, teremos uma classe Carro que será a nossa classe POJO e a classe EstudoJSON que terá o nosso famoso método main.

Início do Fonte

```
package br.com.json;
public class Carro {
    private Long id; private String modelo;
    private String placa;
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getModelo() {
        return modelo; }
    public void setModelo(String modelo) {
        this.modelo = modelo; }
```

Fim do Fonte

```
public String getPlaca() {
    return placa; }
public void setPlaca(String placa) { this.placa = placa; }
//Aqui fizemos o Override do método toString() para
//visualizar a impressão com o System.out.println() @Override
public String toString() {
    return "[id=" + id + ", modelo=" + modelo + ", placa=" + placa + "]; } }
```



Fechamento

Chegamos ao final de mais um curso da nossa apostila de JavaScript. Espero que tenha aproveitado o máximo os nossos conteúdos. Deixo como dica de leitura o site developer.mozilla.org, caso queira ler mais sobre o assunto, [clique aqui e acesse.](#)



Bem turma espero que tenha gostado da nossa suuuuper apostila de JavaScript, agora é pôr a mão na massa e praticar muito.

*Grande Abraço.
Prof. Ricardo Alexandre
Bontempo*

Referência Bibliográfica

Javascript – O Guia definitivo

David Flanagan – trad. Edson FurmankiewiczBookman
Companhia Editora2004

Iniciando em Javascript 1.5

Adrian Kingsley-Hughes e Kathie Kingsley-HughesEditora
Makron Books2001

Javascript 1.3. - Aprenda em 24 Horas

Michael MoncurEditora Campus1999

