

Fundamentos do Desenvolvimento de Software

Programação Web com JavaScript I

Agenda

Etapas 2: Ambientes de programação com Javascript.

- Legibilidade de Código.
- Exercitar Códigos Simples.



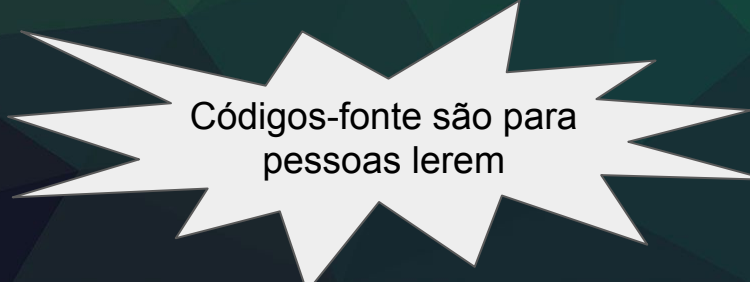
Legibilidade de Código

Se você tiver um arquivo longo com muitas linhas de código e não seguir algumas regras básicas de formatação, será difícil para entender o que você mesmo escreveu.

As duas regras mais importantes por enquanto são: **recuos** e **ponto e vírgula**.

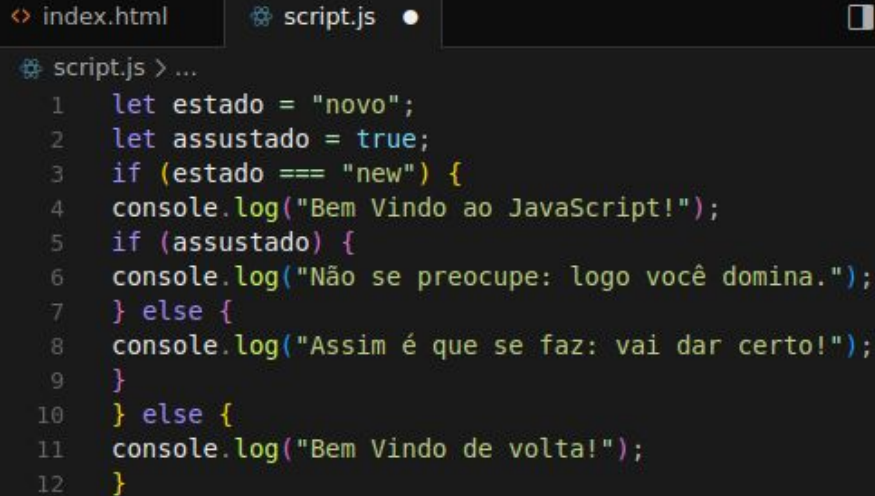
Quando você está escrevendo código, muitas vezes uma linha pertence a um determinado bloco (código entre chaves).

Se for esse o caso, você dá ao código neste bloco um recuo para ter certeza de que pode ver facilmente o que faz parte do bloco e quando um novo bloco é iniciado.

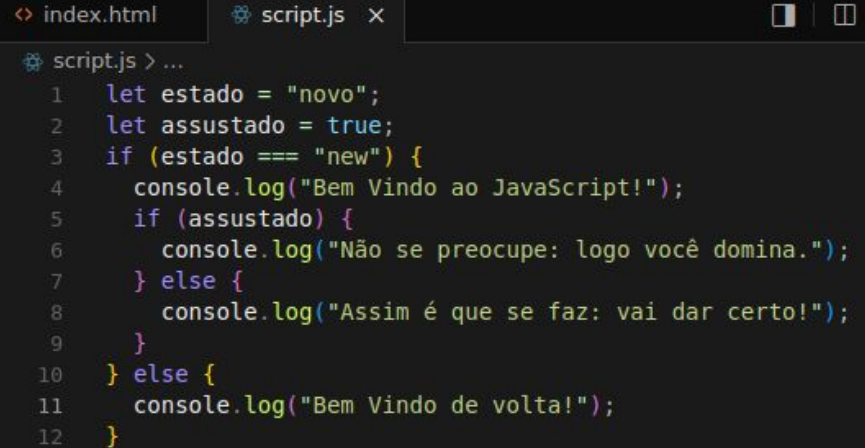


Códigos-fonte são para
pessoas lerem

Endentação ou **Indentação** é o espaço colocado no início de determinadas linhas para melhorar a legibilidade.



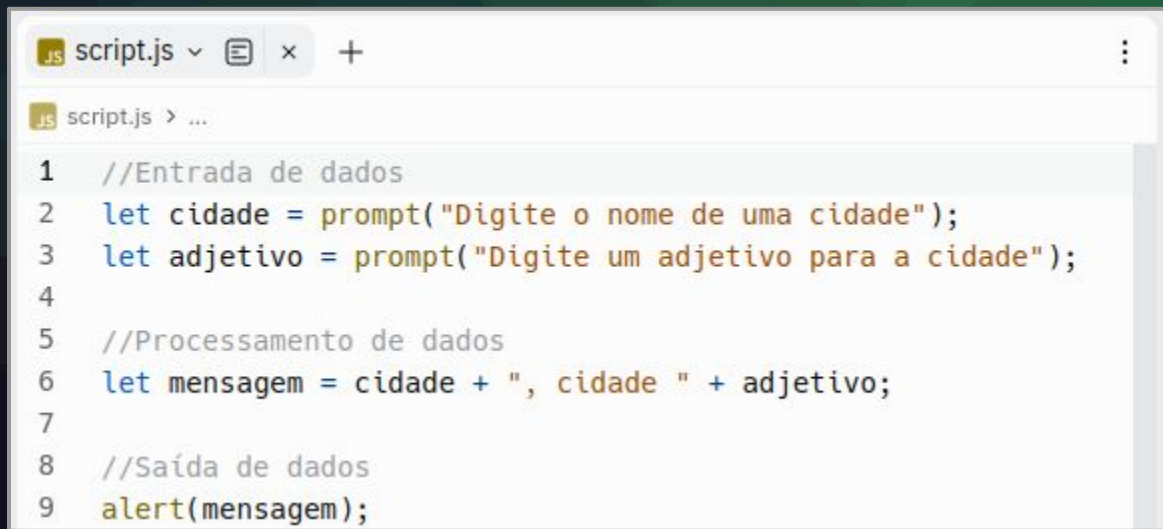
```
script.js > ...
1 let estado = "novo";
2 let assustado = true;
3 if (estado === "new") {
4   console.log("Bem Vindo ao JavaScript!");
5   if (assustado) {
6     console.log("Não se preocupe: logo você domina.");
7   } else {
8     console.log("Assim é que se faz: vai dar certo!");
9   }
10 } else {
11   console.log("Bem Vindo de volta!");
12 }
```



```
script.js > ...
1 let estado = "novo";
2 let assustado = true;
3 if (estado === "new") {
4   console.log("Bem Vindo ao JavaScript!");
5   if (assustado) {
6     console.log("Não se preocupe: logo você domina.");
7   } else {
8     console.log("Assim é que se faz: vai dar certo!");
9   }
10 } else {
11   console.log("Bem Vindo de volta!");
12 }
```


Após cada declaração, você deve inserir um **ponto e vírgula**.

JavaScript é muito tolerante e entenderá muitas situações em que você esqueceu um, mas **desenvolva o hábito de adicionar um ponto e vírgula** após cada linha de código antecipadamente.



```
script.js v [icon] x +  
script.js > ...  
1 //Entrada de dados  
2 let cidade = prompt("Digite o nome de uma cidade");  
3 let adjetivo = prompt("Digite um adjetivo para a cidade");  
4  
5 //Processamento de dados  
6 let mensagem = cidade + ", cidade " + adjetivo;  
7  
8 //Saída de dados  
9 alert(mensagem);
```

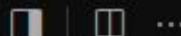
Com **comentários**, você pode dizer ao interpretador para ignorar algumas partes do código.

Isso pode ser pelos seguintes motivos:

- Você não deseja executar um pedaço de código enquanto executa o script, então você o comenta para que seja ignorado pelo interpretador.
- Adicionando algum contexto ao código, como o autor, e uma descrição do que o arquivo abrange.
- Adicionar comentários a partes específicas do código para explicar o que está acontecendo ou por que uma determinada escolha foi feita.

<> index.html

script.js x



script.js > ...

```
1  /*
2      Código de exemplo de endentação
3      Versão 0.1 alfa
4  */
5  let estado = "novo";
6  let assustado = true;
7  //Aqui testamos o estado para saber se o aluno é novo
8  if (estado === "new") {
9      console.log("Bem Vindo ao JavaScript!");
10     //Depois testamos se o aluno está assustado ou não
11     if (assustado) {
12         console.log("Não se preocupe: logo você domina.");
13     } else {
14         console.log("Assim é que se faz: vai dar certo!");
15     }
16 } else {
17     //Essa mensagem será exibida caso o aluno não seja novo
18     console.log("Bem Vindo de volta!");
19 }
```


Exercitar Códigos Simples

String → cadeias de caracteres: **palavras e frases**. São expressos entre aspas simples 'Abc' ou duplas "Abc".

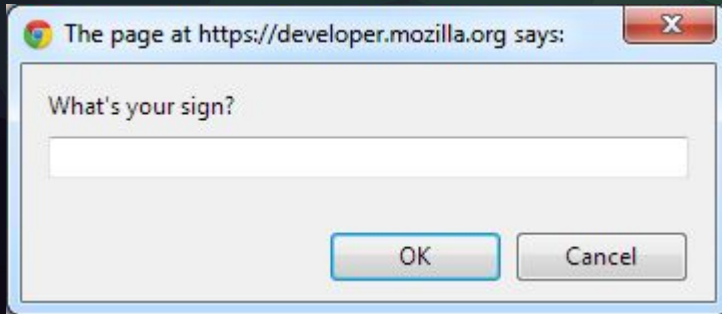
Number → numerais inteiros ou fracionários. São expressos por algarismos entre 0 e 9 com ou sem sinal, com ou sem fração.

Em **JavaScript** você não precisa especificar o tipo de dado de uma variável quando fizer a sua declaração.

Os tipos de dados são convertidos automaticamente conforme a necessidade durante a execução do script.

prompt → exibe uma caixa de diálogo com uma mensagem opcional solicitando ao usuário a entrada de algum texto.

alert → mostra uma caixa de diálogo de aviso com o conteúdo opcionalmente especificado e um botão OK.



let → declara uma variável local com escopo de bloco, inicializando-a opcionalmente com um valor.

variável → é um espaço na memória do computador destinado a um dado que pode ser alterado durante a execução do programa. Deve-se usar nomes sugestivos para facilitar o entendimento do programa.

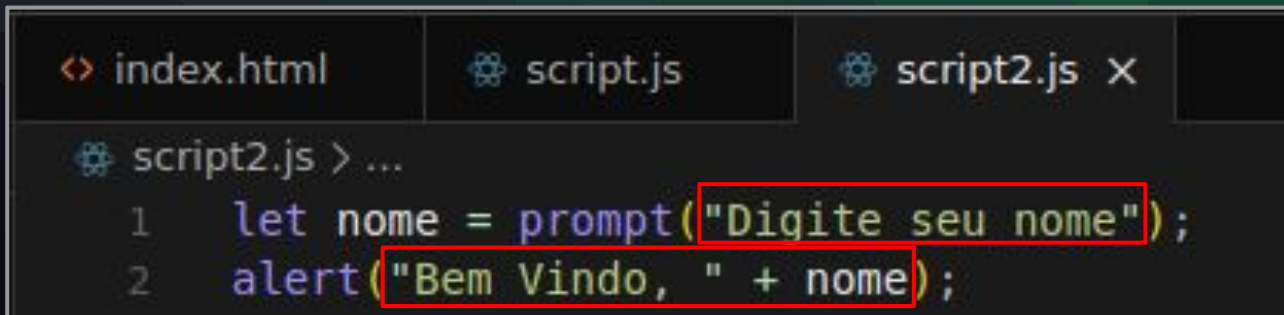
```
1 // Entrada de dados
2 let nome = prompt("Digite seu nome");
```

Parâmetros são variáveis que você lista como parte de uma definição de função.

Usamos parâmetros para dar às funções a capacidade de receber entrada e executar ações com base nessa entrada.

Isso torna as funções mais versáteis e reutilizáveis, pois nos permite criar funções que podem executar a mesma operação em diferentes conjuntos de entradas.

Aparecem nas chamadas entre parênteses.



```
<> index.html  script.js  script2.js x
script2.js > ...
1  let nome = prompt("Digite seu nome");
2  alert("Bem Vindo, " + nome);
```

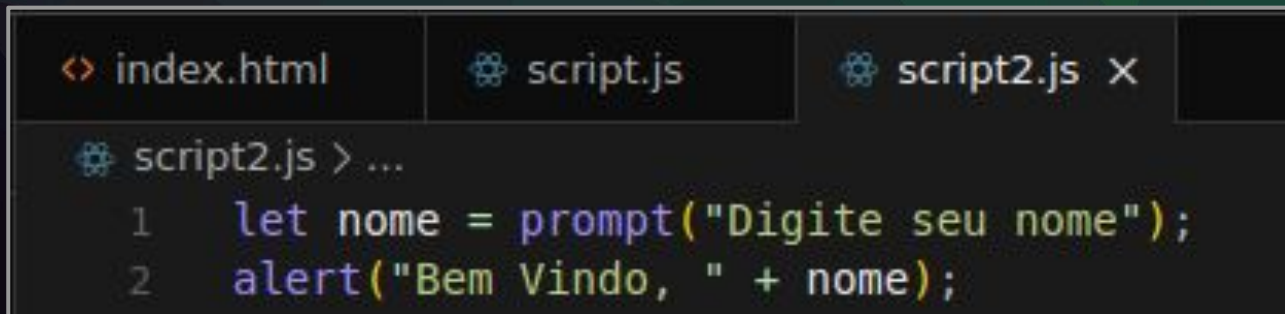
The screenshot shows a code editor with three tabs: 'index.html', 'script.js', and 'script2.js'. The 'script2.js' tab is active, showing two lines of JavaScript code. The first line is 'let nome = prompt("Digite seu nome");' and the second line is 'alert("Bem Vindo, " + nome);'. Both the string arguments in the 'prompt' and 'alert' functions are highlighted with red rectangular boxes, illustrating how parameters are passed to functions.

Valores de retorno são exatamente o que parecem — os valores que uma função retorna quando é concluída.

Você já encontrou valores de retorno várias vezes, embora possa não ter pensado neles explicitamente.

Aqui **prompt** retorna o que foi digitado no campo como **String** - você usa uma variável para obter esse dado digitado.

Não confunda executar uma ação com retorno: **alert** simplesmente apresenta uma mensagem e não retorna dado nenhum.



```
<> index.html  script.js  script2.js x
script2.js > ...
1  let nome = prompt("Digite seu nome");
2  alert("Bem Vindo, " + nome);
```

Crie um programa que receba na entrada o valor de três notas de um aluno - com valor entre 0 e 10 e informe a média entre elas.

Neste momento, não é necessário validar se a nota está dentro do intervalo válido!



index.html

script.js



script.js > ...

```
1  /*
2      Crie um programa que receba na entrada o valor de três
3      notas de um aluno - com valor entre 0 e 10 e informe a
4      média entre elas.
5  */
6  //Entrada de dados
7  let nota1 = prompt("Digite a primeira nota");
8  let nota2 = prompt("Digite a segunda nota");
9  let nota3 = prompt("Digite a terceira nota");
10
11 //Processamento de dados
12 let media = (nota1 + nota2 + nota3) / 3;
13
14 //Saída de dados
15 alert(media);
```

NaN quer dizer "Not
a Number"

<> index.html

script.js x



script.js > ...

```
1  /*
2      Crie um programa que receba na entrada o valor de três
3      notas de um aluno - com valor entre 0 e 10 e informe a
4      média entre elas.
5  */
6  //Entrada de dados
7  let nota1 = Number(prompt("Digite a primeira nota"));
8  let nota2 = Number(prompt("Digite a segunda nota"));
9  let nota3 = Number(prompt("Digite a terceira nota"));
10
11  //Processamento de dados
12  let media = (nota1 + nota2 + nota3) / 3;
13
14  //Saída de dados
15  alert(media);
```



Crie um programa que receba o valor da altura e do peso de uma pessoa e retorne o seu IMC - Índice de Massa Corporal.

$$\text{IMC} = \text{peso} / \text{altura}^2$$

