




Lógica de Programação

JavaScript





 profalex.deus@fiap.com.br

Alex Sander Resende de Deus

Professor

Há 25 anos ensinando programação, é um apaixonado por tecnologia. Atualmente é coordenador de cursos na ETEC Albert Einstein do Centro Paula Souza.

E professor da FIAP School e FIAP Corporate, lecionando C#, Banco de Dados e Desenvolvimento Mobile.

AGENDA

1

AULA 1

Definição e histórico

Fundamentos Básicos da Programação – Tipos de dados, variáveis e estruturas básicas; Usando o GIT

2

AULA 2

Estruturas condicionais, operadores lógicos e switch case

3

AULA 3

Laços de Repetição: For, while,

4

AULA 4

Objetos nativos JavaScript – Date, String, Math

AGENDA

5

AULA 6

Arrays, Filter, Map

Reduce, Splite, forEach, for in, for of

6

AULA 6

Funções: tipos, declarações, escopo, retorno, clousers,
call-back, IIFE, factory

7

AULA 7

DOM – Document Object Model

8

AULA 8

DOM – Document Object Model

AGENDA

9

AULA 9

- Eventos
- Objetos: Criação, prototype, getters, setters

10

AULA 10

Classes JavaScript Assíncrono



Instalando o Visual Studio Code



<https://visualstudio.microsoft.com/pt-br/downloads/>

Qual é a diferença entre var e let x JavaScript Operators x Tryit Editor v3.7 x Baixar Ferramentas do Visual Studio x

<https://visualstudio.microsoft.com/pt-br/downloads/>

IDE (ambiente de desenvolvimento integrado) completo para Android, iOS, Windows, Web e nuvem

[Comparar as edições >](#)
[Como fazer a instalação offline >](#)

Download gratuito ↓

IDE avançada, gratuito para estudantes, colaboradores de software livre e indivíduos

IDE profissional mais indicado para equipes pequenas

Avaliação gratuita ↓

Solução pont-a-ponto escalonável para equipes de qualquer tamanho

Avaliação gratuita ↓

Visual Studio Code
[Notas de versão >](#)

O editor de código-fonte gratuito e rápido que se adapta às suas necessidades

Download grátis ↓

Ao baixar e usar o Visual Studio Code, você concorda com os [termos de licença](#) e [política de privacidade](#).

Visual Studio para Mac
Versão 8.10
[Release notes >](#)

Desenvolva aplicativos e jogos para os sistemas iOS e Android e para a Web usando o .NET

Download grátis ↓

[Como ativar sua licença](#): Selecione sua edição após a instalação. O Visual Studio para Mac requer uma conexão de Internet ativa para a instalação.

Me ajude a escolher!

Comentários

AULA 1

JavaScript

JavaScript é uma linguagem de programação que permite implementar itens complexos em páginas web — toda vez que uma página da web faz mais do que simplesmente mostrar a você informação estática — mostrando conteúdo que se atualiza em um intervalo de tempo, mapas interativos ou gráficos 2D/3D animados etc. — você pode apostar que o JavaScript provavelmente está envolvido. É a terceira camada do bolo das tecnologias padrões da web (HTML e CSS).



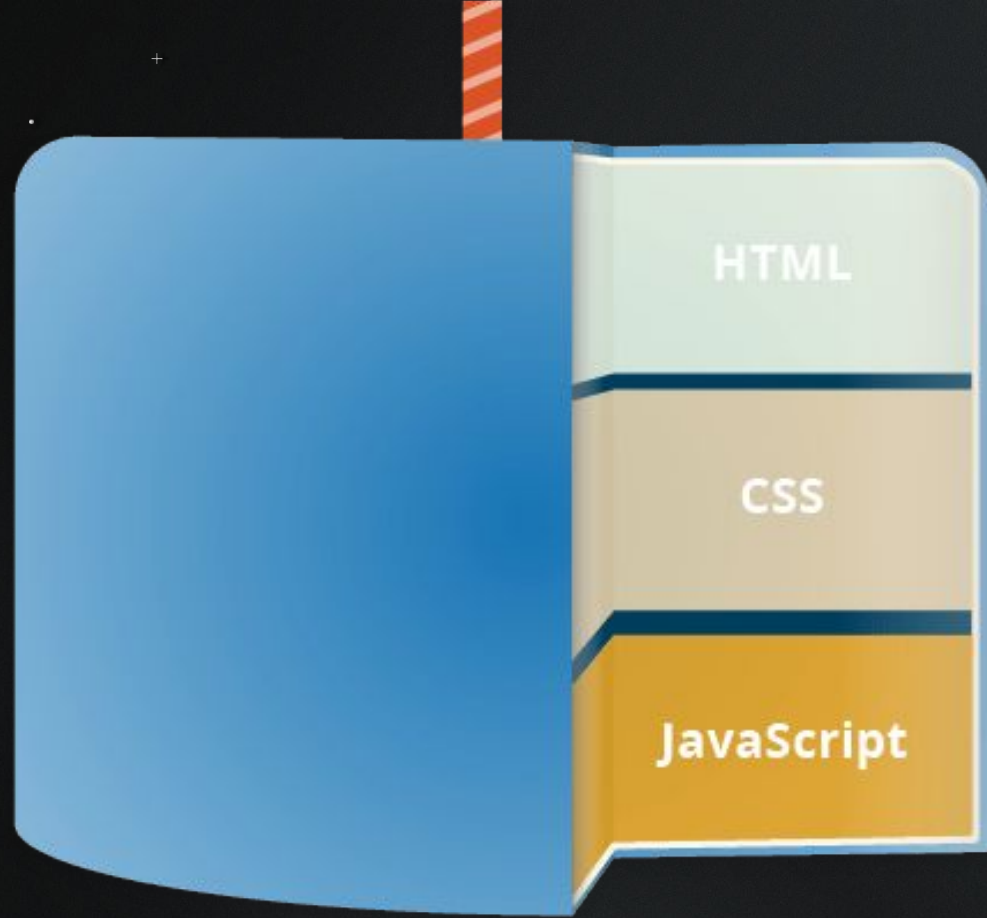
Um pouco de História



JavaScript é uma Linguagem de Programação, criada por Brendan Eich, a pedido da empresa Netscape, em meados de 1995. No início, o JavaScript foi batizado com outro nome: LiveScript. No entanto, a Netscape não ficou sozinha com o desenvolvimento do JavaScript. A empresa SUN Microsystems interessou-se por ela e entrou de cabeça no desenvolvimento desta nova linguagem, uma vez que acreditava na ideia inovadora que era o JavaScript.

Com o sucesso inicial do JavaScript, a mudança do nome de LiveScript para JavaScript foi inevitável, e, com certeza, veio por influência da própria SUN, que mantém uma Linguagem de Programação chamada JAVA. É claro que as Linguagens de Programação JAVA e JavaScript são parecidas somente no nome, já que se diferem no conceito e no uso.

O JavaScript, a princípio, foi criado com o objetivo principal de validar formulários HTML. Na verdade, o JavaScript foi concebido para rodar no servidor. No entanto, com o passar do tempo isso foi modificado e, hoje, o JavaScript é uma linguagem Client-side.



Tipos de Dados

Tipo "Boolean": Boolean representa uma entidade lógica e pode ter dois valores: verdadeiro (true) ou falso (false).

Tipo "Null": O tipo Null tem exatamente um valor: null (nulo).

Tipo "Undefined": Uma variável que não foi atribuída a um valor específico, assume o valor undefined (indefinido).

Tipo "Number": Qualquer valor numérico.

Tipo "String": O tipo String em JavaScript é usado para representar dados textuais.

Tipagem Dinâmica

- JavaScript é uma linguagem de tipagem dinâmica. Isso significa que você não necessita declarar o tipo de uma variável antes de sua atribuição. O tipo será automaticamente determinado quando o programa for processado. Isso também significa que você pode reatribuir uma mesma variável com um tipo diferente:

```
var foo = 42;    // foo é um Number agora  
foo      = "bar"; // foo é um String agora  
foo      = true;  // foo é um Boolean agora
```

let **vs** var

Uma resposta curta e simples para este questionamento é: “O que muda é o escopo”. Quando você declara uma variável com a palavra **var**, ela pode corresponder ao escopo global e local, já quando você faz isso com a palavra **let**, ela pode corresponder ao escopo global, local e de bloco. Caso não tenha ficado muito claro, vamos entender isso em detalhes agora.

Afinal, o que é **escopo**?

Em poucas palavras, escopo é a propriedade que determina onde uma variável pode ser utilizada dentro de um programa, por exemplo, se você declara uma variável dentro de uma função, só aquela função consegue utilizar a variável. Se você declara uma variável fora de uma função, ela pode ser acessada de qualquer parte daquele mesmo script, pois ela foi declarada globalmente. No Javascript existem 3 tipos de escopos, **o global, local e de bloco** (adicionado no ES6). Vamos ver em detalhes como cada um deles funciona para `var` e `let` a seguir.

Escopo Global

Quando você declara uma variável fora de qualquer função, seja ela **var** ou **let**, ela tem o escopo global, pois qualquer função no script consegue utilizar esta variável. Veja abaixo um exemplo de uma variável global declarada com var:

Note que, tanto com var como com let as variáveis são visíveis em qualquer parte do programa.

```
var minhaVariavelGlobal = 10;  
let minhaVariavelGlobal2=5
```

```
function func1(){  
  console.log(minhaVariavelGlobal);  
  console.log(minhaVariavelGlobal2);  
}
```

```
function func2(){  
  console.log(minhaVariavelGlobal);  
  console.log(minhaVariavelGlobal2);  
}
```

```
function func3(){  
  console.log(minhaVariavelGlobal);  
  console.log(minhaVariavelGlobal2);  
}  
  
console.log(minhaVariavelGlobal);  
console.log(minhaVariavelGlobal2);
```

Escopo Local

Uma variável é local quando ela é declarada dentro de alguma função, pois isso significa que apenas aquela função consegue enxergá-la. As demais funções do script não conseguem utilizá-la, pois a mesma foi declarada dentro de uma função específica. Neste ponto de vista, podemos dizer que o escopo local é completamente diferente do escopo global. A seguir vamos ver um exemplo de uma variável declarada localmente com `var`.

Neste exemplo,
usando o LET e o VAR
daria o mesmo
resultado. Até agora
nada de diferente.

// Consegue manipular a variável minhaVar

```
function func1(){  
  var minhaVar = "Teste";  
  console.log(minhaVar);  
}
```

// NÃO Consegue manipular a variável minhaVar

```
function func2(){  
  console.log(minhaVar);  
}
```

// NÃO Consegue manipular a variável minhaVar

```
function func3(){  
  console.log(minhaVar);  
}
```


Escopo de Bloco

Podemos dizer que um bloco é tudo aquilo em um código que está entre chaves (`{ }`), ou seja, estruturas condicionais, estruturas de repetição, entre outras entidades que trabalham com blocos. Então, em resumo, uma variável com escopo de bloco é uma variável que foi declarada dentro de um determinado bloco, e apenas pode ser manipulada dentro daquele bloco e nenhuma outra parte do código pode manipulá-la. Dentre todos os tipos de escopo este é o mais restrito.

Este tipo de escopo só funciona com variáveis declaradas com a palavra chave `let` e está é a grande diferença entre `let` e `var`. O escopo de bloco abaixo é um exemplo de uma variável `let` declarada dentro de um bloco `if`.

Note agora que com o
let a variável só
funciona no bloco
onde foi declarada.

```
if(condicao){  
  let minhaVariavel = 10;  
  // É possível manipular a let, pois ela foi declarada  
  // neste bloco  
  console.log(minhaVariavel);  
}
```

// ERRO, Não é possível acessar a variável, pois ela tem escopo de bloco.

```
function func1(){  
  console.log(minhaVariavel);  
}
```

// ERRO, Não é possível acessar a variável, pois ela tem escopo de bloco.

```
function func2(){  
  console.log(minhaVariavel);  
}
```

// ERRO, Não é possível acessar a variável, pois ela tem escopo de bloco.

```
console.log(minhaVariavel);
```

Já com o var, mesmo declarada dentro de um bloco a variável ainda é visível em todo o programa.

```
if(condicao){  
  var minhaVariavel = 10;  
  console.log(minhaVariavel);  
}
```

```
// Funciona, pois o escopo de bloco não existe para var  
function func1(){  
  console.log(minhaVariavel);  
}
```

```
// Funciona, pois o escopo de bloco não existe para var  
function func2(){  
  console.log(minhaVariavel);  
}
```


Constantes

Ao contrário das variáveis, as constantes recebem valores no momento de sua declaração e não podem ter seu valor alterado.

Exemplo:

```
const mensagem = 'teste';
```

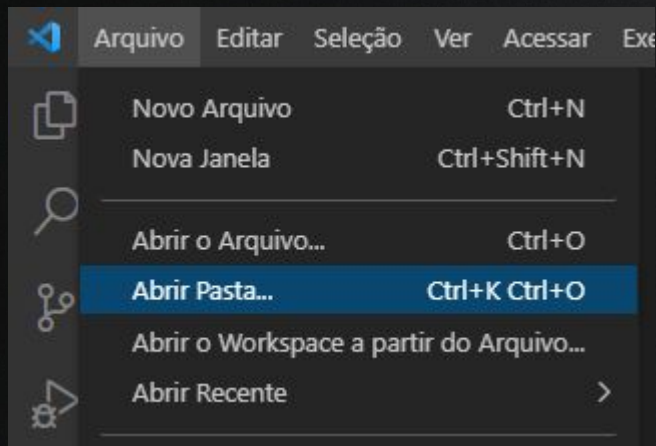
Obs.: as constantes tem escopo de bloco como variáveis declaradas com **let**.

OPERADORES ARITMÉTICOS

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
**	Exponenciação
/	Divisão
%	Módulo – Resto da Divisão
++	Incremento
--	Decremento

Projeto Olá Mundo!

- Abra o Visual Studio Code e siga a sequencia abaixo:

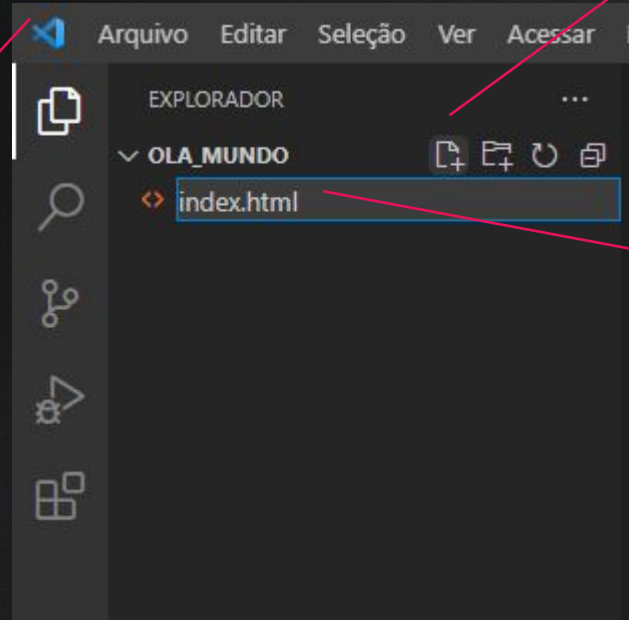


Vamos criar uma pasta para o nosso projeto.

Na unidade C, crie uma pasta chamada **modulo_logica** e dentro dela crie outra pasta chamada **ola_mundo**.

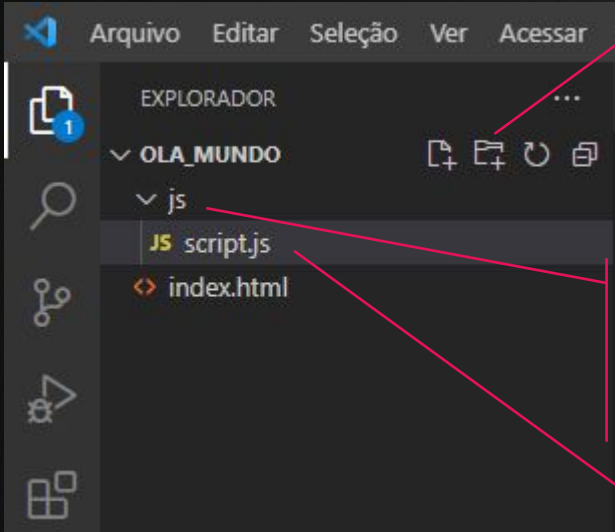
Na caixa de diálogo, clique em não salvar.

1 - Clique no
botão
EXPLORADOR.



2 – Clique no botão
NOVO ARQUIVO.

3 – Digite o nome do
novo arquivo com a
extensão HTML.



1 – Clique no botão
NOVA PASTA.

2 – Crie uma pasta
com o nome js para
acomodar o script.

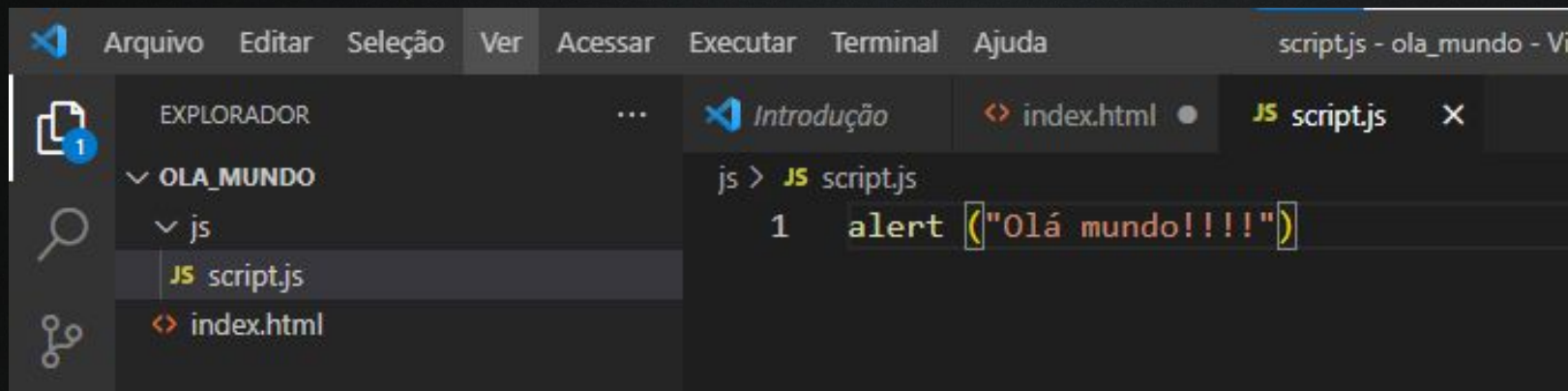
3 – Crie o arquivo
script.js dentro da
pasta js.

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h2>Introdução JavaScript</h2>
6
7 <p>Meu primeiro script</p>
8
9 <p id="saida"></p>
10
11 <script>
12
13 document.getElementById("saida").innerHTML ="Olá Mundo!!!!";
14 </script>
15
16 </body>
17 </html>
```

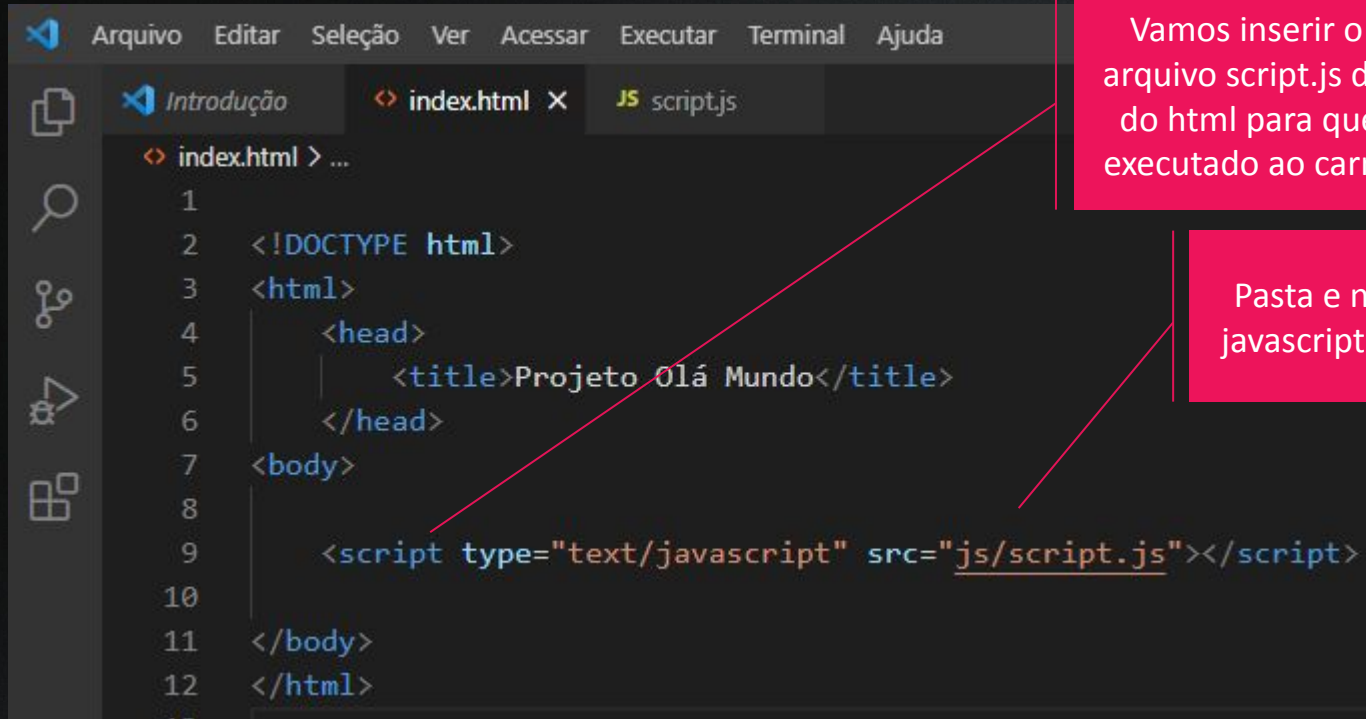
Para exibir o resultado, vamos utilizar em parágrafo identificado.

Aqui temos a saída no parágrafo identificado.

- Este é o código do arquivo script.js



É agora o código do arquivo html:

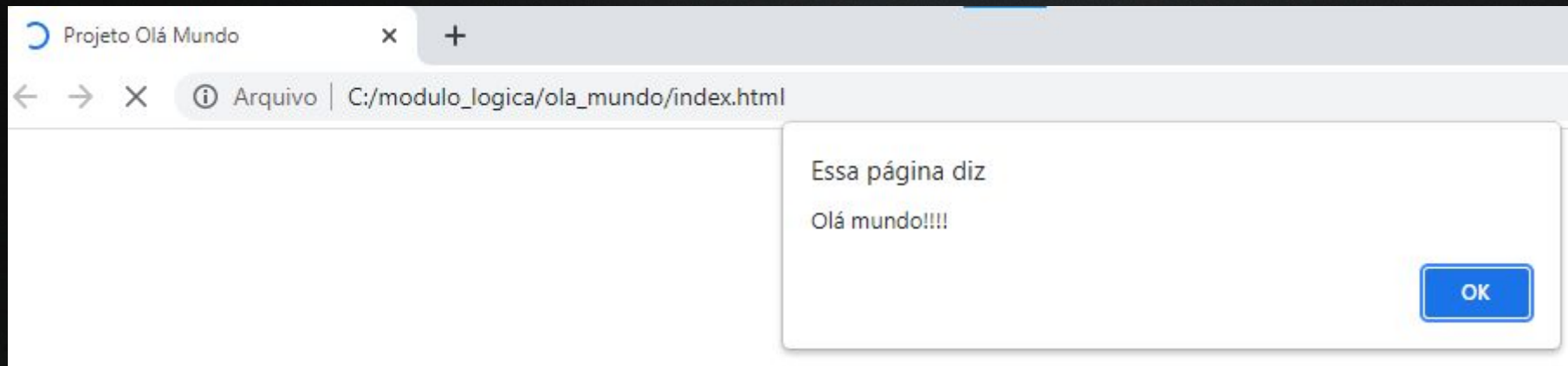


```
1
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <title>Projeto Olá Mundo</title>
6   </head>
7   <body>
8
9     <script type="text/javascript" src="js/script.js"></script>
10
11   </body>
12 </html>
```

Vamos inserir o chamado ao arquivo script.js dentro do body do html para que o script seja executado ao carregar a página.

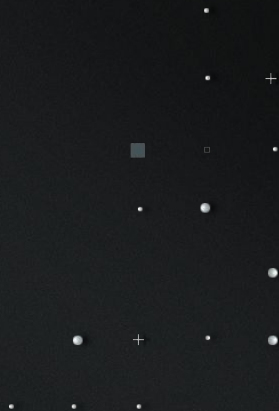
Pasta e nome do arquivo javascript a ser executado.

Eis a saída no navegador

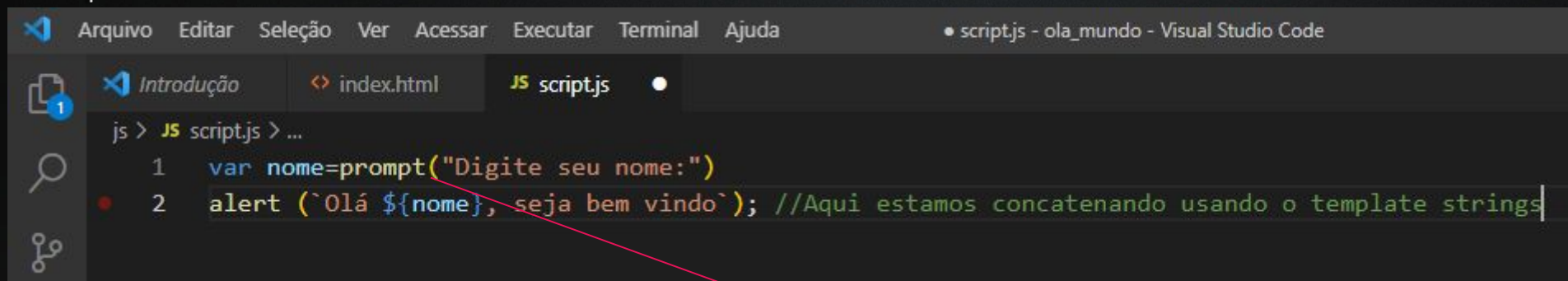




Melhorando o Olá Mundo



Experimente fazer esta alteração no seu script e veja o resultado no navegador:



The image shows a screenshot of the Visual Studio Code editor interface. The top menu bar includes 'Arquivo', 'Editar', 'Seleção', 'Ver', 'Acessar', 'Executar', 'Terminal', and 'Ajuda'. The title bar indicates the file is 'script.js - ola_mundo - Visual Studio Code'. The editor has three tabs: 'Introdução', 'index.html', and 'script.js'. The 'script.js' tab is active, showing the following code:

```
js > JS script.js > ...  
1  var nome=prompt("Digite seu nome:")  
2  alert (`Olá ${nome}, seja bem vindo`); //Aqui estamos concatenando usando o template strings
```

A red line points from the `prompt` function in line 1 to a red callout box on the right.

O prompt permite a entrada de dados.

Usando o GIT





- Ao acessar o link <https://git-scm.com/> o site oficial do git detecta automaticamente a versão do sistema operacional que você está rodando.
- Basta clicar no botão de download.
- Após a conclusão do download executaremos o instalador.
- As opções padrão do instalador nos dão exatamente a estrutura que precisaremos.

- Abra o console do Windows (pesquise por cmd)
- Navegue até a pasta do seu projeto
- Digite: `git config --global user.email "seu@email.com"` para configurar o email o usuário que estará publicando projetos
- Agora, digite `git config --global user.name "Seu Nome"` para configurar o nome do usuário
- Digite `git init` para criar um repositório para o projeto
- Digite `git add .` para versionar o projeto
- E por fim digite: `git commit -m "projeto da Aula 01 do curso de lógica"` (grava no git e insere uma mensagem identificando a versão)

Selecione o Prompt de Comando

```
c:\Treinamento C#\boasVindas>git config --global user.email "prof.alexresende@gmail.com"

c:\Treinamento C#\boasVindas>git config --global user.name "Alex Sander Resende de Deus"

c:\Treinamento C#\boasVindas>git init
Initialized empty Git repository in C:/Treinamento C#/boasVindas/.git/

c:\Treinamento C#\boasVindas>git add .

c:\Treinamento C#\boasVindas>git commit -m "projeto da aula 01 do curso de lógica"
[master (root-commit) 50d9cb9] projeto da aula 01 do curso de lógica
43 files changed, 807 insertions(+)
create mode 100644 .vs/boasVindas/v16/.suo
create mode 100644 boasVindas.sln
create mode 100644 boasVindas/App.xaml
create mode 100644 boasVindas/App.xaml.cs
create mode 100644 boasVindas/AssemblyInfo.cs
create mode 100644 boasVindas/MainWindow.xaml
create mode 100644 boasVindas/MainWindow.xaml.cs
create mode 100644 boasVindas/bin/Debug/netcoreapp3.1/boasVindas.deps.json
create mode 100644 boasVindas/bin/Debug/netcoreapp3.1/boasVindas.dll
create mode 100644 boasVindas/bin/Debug/netcoreapp3.1/boasVindas.exe
create mode 100644 boasVindas/bin/Debug/netcoreapp3.1/boasVindas.pdb
create mode 100644 boasVindas/bin/Debug/netcoreapp3.1/boasVindas.runtimeconfig.dev.json
create mode 100644 boasVindas/bin/Debug/netcoreapp3.1/boasVindas.runtimeconfig.json
create mode 100644 boasVindas/boasVindas.csproj
create mode 100644 boasVindas/boasVindas.csproj.user
create mode 100644 boasVindas/obj/Debug/netcoreapp3.1/.NETCoreApp,Version=v3.1.AssemblyAttributes.cs
create mode 100644 boasVindas/obj/Debug/netcoreapp3.1/App.g.cs
```


CRIANDO UM REPOSITÓRIO REMOTO

Owner *
profalexresende

Repository name *
boasVindas ✓

Great repository names are short and unique. boasVindas is available. Inspiration? How about sturdy-disco?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

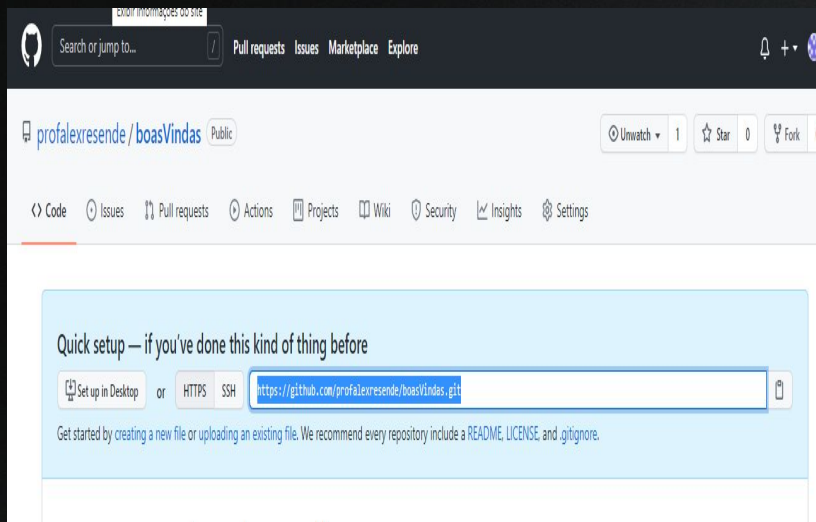
☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

- Uma vez que estiver logado no site do GitHub, acesse <https://github.com/new>.
- Crie um nome para o novo repositório (pode ser o mesmo nome do projeto criado no Visual Studio).
- Dê uma descrição para o seu repositório.
- Indique se ele é público ou privado.
- Ao final, clique em Create Repository.

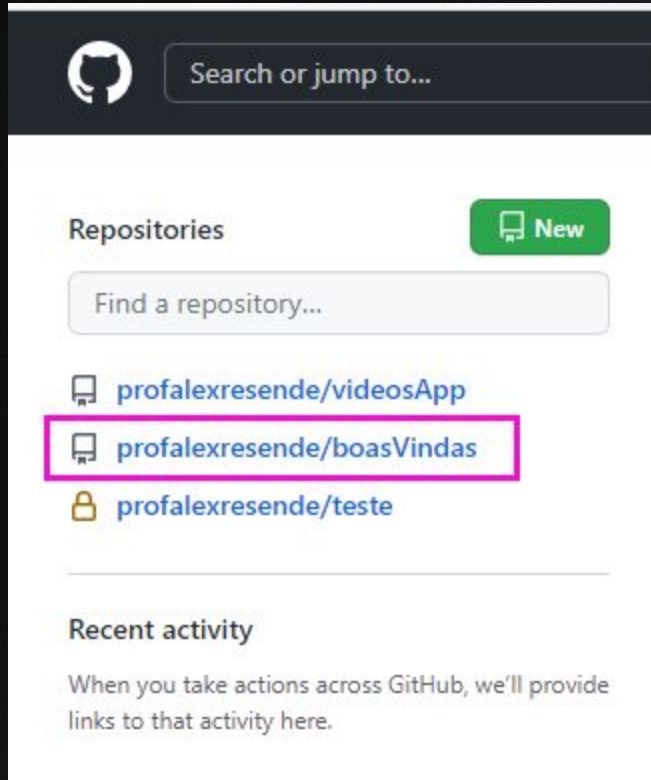
CRIANDO UM REPOSITÓRIO REMOTO



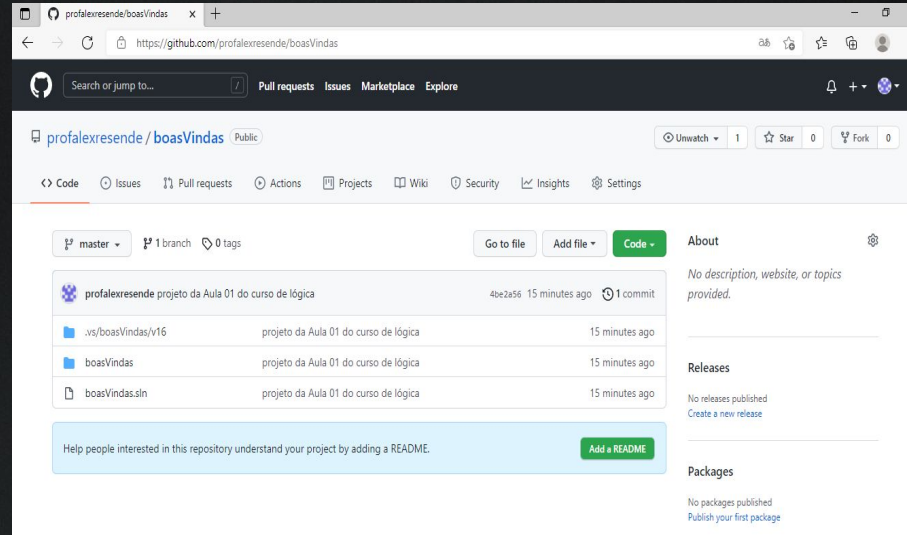
- Copie o link do seu repositório.
- Vamos utilizar o link gerado para incluir no terminal.

- Abra novamente o console do Windows (pesquise por cmd)
- Navegue até a pasta do seu projeto
- Digite: `git remote add origin link_que_você_copiou`
- Digite: `git remote -v` (este comando mostra o link dos seus repositórios)
- Digite: `git push -f origin master` (este comando envia o seu projeto para os repositórios remotos. Uma tela para inserir credenciais pode ser exibida)

CRIANDO UM REPOSITÓRIO REMOTO



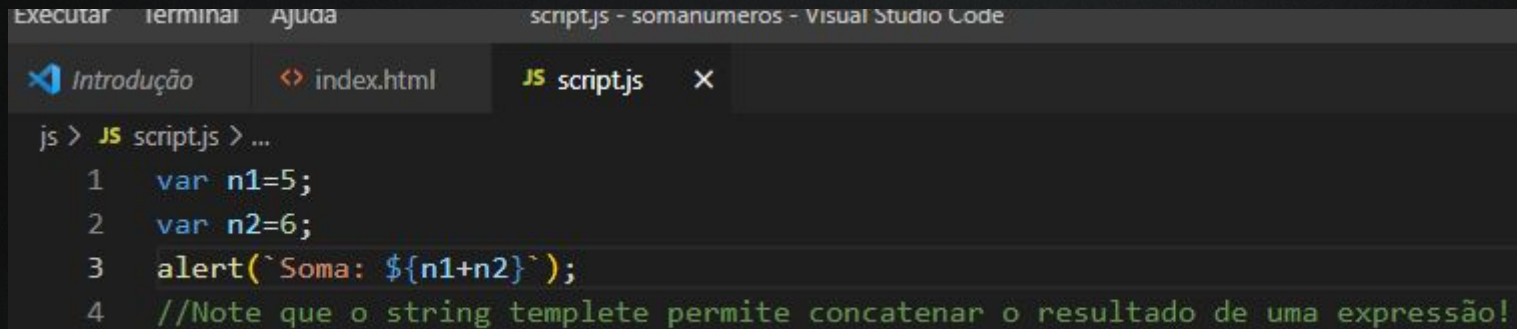
- Clicando no logo do git no canto superior esquerdo do site, podemos ver nossos repositórios.
- Clique no repositório para ver seus arquivos lá!



Trabalhando com Números

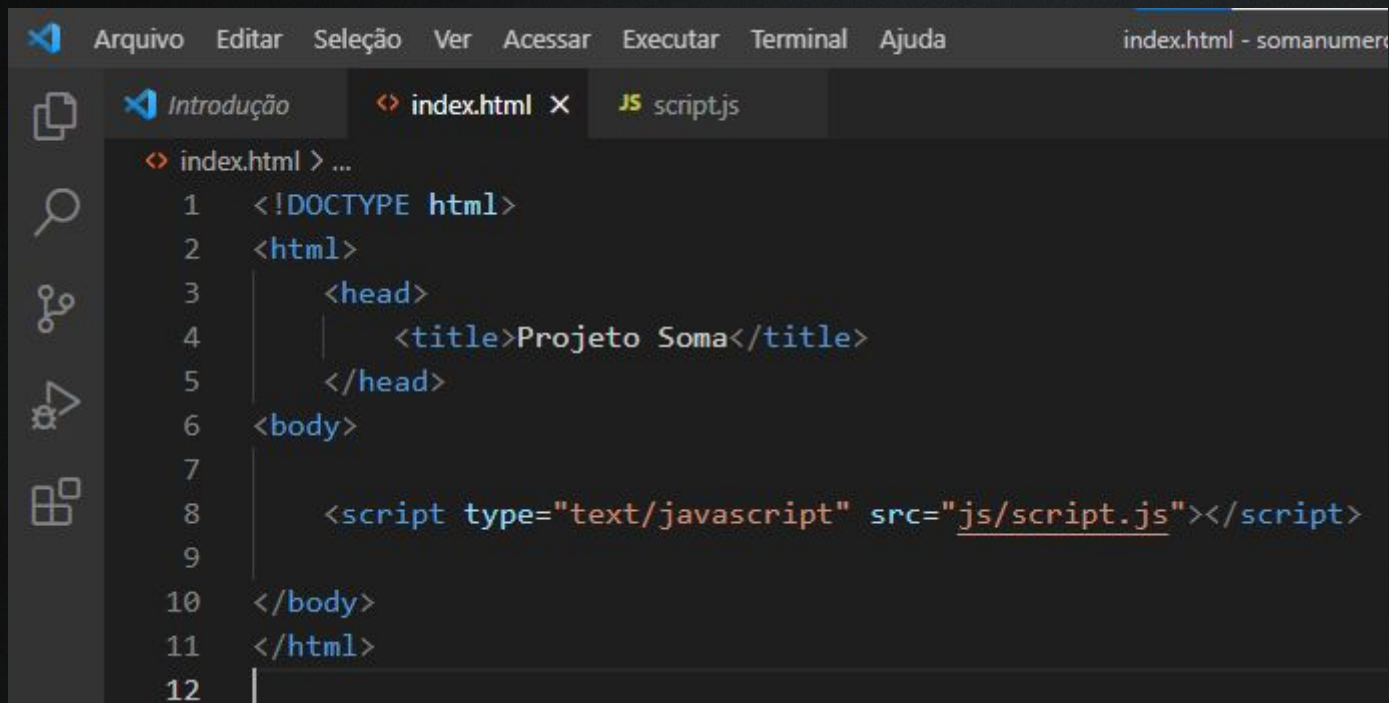
Utilizando o VS Code, crie agora a pasta somanumeros dentro da pasta modulo_logica que está em sua unidade C:.

Crie a pasta js e digite este script:



```
js > JS script.js > ...
1  var n1=5;
2  var n2=6;
3  alert(`Soma: ${n1+n2}`);
4  //Note que o string template permite concatenar o resultado de uma expressão!
```


• E agora o HTML do projeto:



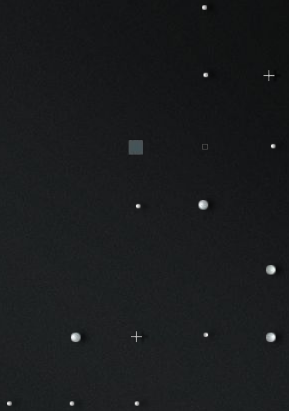
The screenshot shows the Visual Studio Code editor interface. The top menu bar includes 'Arquivo', 'Editar', 'Seleção', 'Ver', 'Acessar', 'Executar', 'Terminal', and 'Ajuda'. The title bar on the right says 'index.html - somanumeros'. The editor has three tabs: 'Introdução', 'index.html X', and 'JS script.js'. The 'index.html' tab is active, showing the following HTML code:

```
index.html > ...
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Projeto Soma</title>
5      </head>
6  <body>
7
8      <script type="text/javascript" src="js/script.js"></script>
9
10 </body>
11 </html>
12
```

Envie seu projeto para o **git** e compartilhe o link no
chat da reunião



Moment O Hands on



Ocasionalmente, a ONG para a qual você trabalha recebe doações em dólar e precisa saber qual é o valor em reais.

Crie um programa que permita que o usuário digite o valor da doação em dólares, converta esse valor para reais e exiba o resultado na tela.

- Um funcionário da ONG, do exercício anterior, realiza o trabalho de buscar alimentos diariamente no Mercado Municipal utilizando um carro.

É importante que a ONG saiba quantos quilômetros por litro esse carro faz.

Crie um programa em que o usuário digite quantos quilômetros o painel do carro mostra no início de uma viagem, quantos quilômetros ele mostra na chegada ao posto de gasolina e quantos litros foram reabastecidos. O programa deve calcular e exibir a média de quilômetros por litro que o veículo faz.

Em uma escola, os alunos recebem uma nota denominada AC, que vale 20% da média, uma nota denominada Avaliação Geral, que vale 10% da média e uma terceira nota denominada Avaliação Trimestral, que vale 70% da nota. Elabore um programa que leia essas três notas e calcule a média deste aluno.

Referências

- <https://developer.mozilla.org/pt-BR/docs/Aprender/JavaScript>
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array
- https://developer.mozilla.org/pt-BR/docs/DOM/Referencia_do_DOM
- <https://www.w3schools.com/>

OBRIGADO!



profalex.deus@fiap.com.br



/alexsanderresende

FIAP

Copyright © 2018 | Professor (a) Alex Sander Resende de Deus

Todos os direitos reservados. A reprodução ou divulgação total ou parcial deste documento é expressamente proibida sem o consentimento formal, por escrito, do professor/autor.

