
CÓDIGOS DE ALTA
PERFORMANCE - WEB

# HELLO!

## Sou a profa, Márcia!

## Formação

- Graduação: Analise e desenvolvimento de sistemas
- Graduação: Pedagogia
- Pós Graduada: Docência do Ensino Superior
- Pós Graduada: Projetos de aplicações Móveis Multiplataformas
- MBA em Segurança da Informação

# UNIDADE I

- PROTOCOLOS ENVOLVIDOS NA COMUNICAÇÃO NA WEB (HTTP, SMTP, POP3)
- ARQUITETURA DE APLICAÇÕES WEB
- WEB DESIGN E UI/UX DESIGN
- UTILIZAÇÃO DE FRAMEWORKS E FERRAMENTAS PARA DESENVOLVIMENTO FRONT-END
- MODELO DE COMUNICAÇÃO ENTRE NAVEGADOR (CLIENTE) E SERVIDOR WEB

# UNIDADE II

- PROGRAMAÇÃO FRONT-END: HTML, CSS, JAVASCRIPT
- MANIPULAÇÃO DO DOM
- FRAMEWORK DE DESENVOLVIMENTO FRONT-END

# UNIDADE III

- DESENVOLVIMENTO PROGRESSIVO DE APLICAÇÕES WEB
- SINGLE-PAGE
- AMBIENTE DE EXECUÇÃO JAVASCRIPT SERVER-SIDE (NODE.JS)
- SEGURANÇA WEB EM NUVEM

# UNIDADE IV

- COMUNICAÇÃO ENTRE FRONT-END E BACK-END
- ARMAZENAMENTO DE DADOS NO NAVEGADOR (HTML5)
- PROJETO PRATICO EM LABORATÓRIO

# O QUE VAMOS USAR?

- Navegador: Chrome, Edge, etc;
- Editor HTML: Visual Studio Code;
- Banco: Mysql;
- Linguagens de programação: JavaScript, Node.js.
- Frameworks: Bootstrap, React.
- Repositório: GitHub

# PLANO DE ENTREGAS - AVALIAÇÕES

**AV1 - Trabalho (Total: 2,0 pontos) + Prova (Total: 8,0 pontos)**

Parte 1 - Escrita do Trabalho (Valendo 1,0 pontos)

Parte 2 - Apresentação do Pitch (Valendo 1,0 pontos)

**Critérios de Avaliação:**

- Clareza na escrita e organização do conteúdo.
- Adequação ao tema e profundidade do conteúdo.
- Coerência entre objetivos, metodologia e resultados apresentados.
- Qualidade da redação (gramática, estrutura e clareza).

**Pesos das Avaliações:** AV1 (Total: 10,0 pontos)

# PLANO DE ENTREGAS - AVALIAÇÕES

**AV2 - Projeto Final (Total: 5,0 pontos) + Prova (Total: 5,0 pontos)**

Parte 1 - Relatório Escrito do Projeto: 2,5 pontos

Parte 2 - Apresentação do Projeto já desenvolvido: 2,5 pontos

**Critérios de Avaliação:**

- Complexidade e profundidade do projeto.
- Qualidade técnica e inovação.
- Documentação e justificativa das escolhas feitas durante o desenvolvimento.
- Funcionamento e apresentação do projeto.

**Pesos das Avaliações:** AV2 (Total: 10,0 pontos)

# PLANO DE ENTREGAS - AVALIAÇÕES

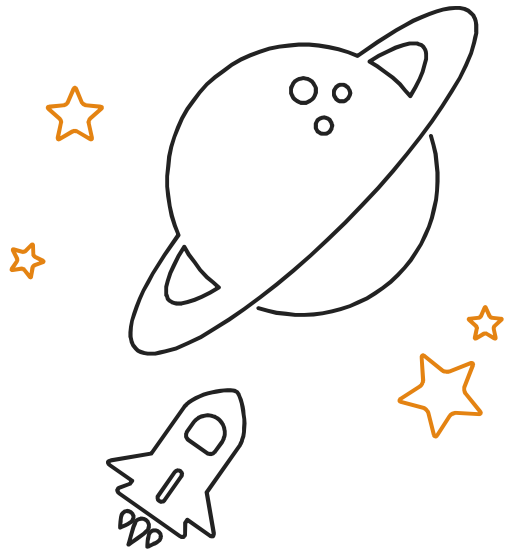
**Observações:**

- O projeto final será uma aplicação web funcional que deve ser entregue com o código-fonte completo e documentação.
- Os alunos devem demonstrar a aplicação ao vivo durante a apresentação ou fornecer um link de acesso para a avaliação da aplicação.
- O relatório escrito deve ser entregue antes da apresentação, e a aplicação web deve estar funcionando corretamente no momento da apresentação.
- A avaliação levará em conta tanto os aspectos técnicos quanto a clareza na comunicação do trabalho realizado.

# 1. PROTOCOLOS ENVOLVIDOS NA COMUNICAÇÃO NA WEB

# FUNCIONAMENTO DE UM SOFTWARE WEB

| CLIENT SIDE | SERVER SIDE |
|---|---|
| HTML | PHP |
| CSS | MYSQL |
| Javascript | Node.js |

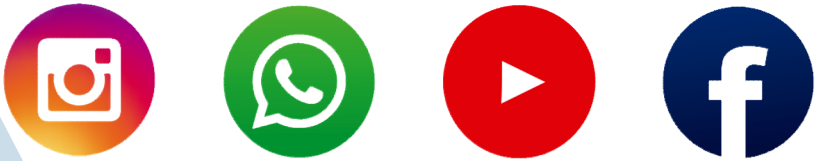


**Segundo o autor Luiz Duarte.**

*“O front-end é responsável por coletar a entrada do usuário em várias formas e processá-las para adequá-las a uma especificação em que o back-end possa utilizar.”*

# O QUE SÃO PROTOCOLOS DE INTERNET?

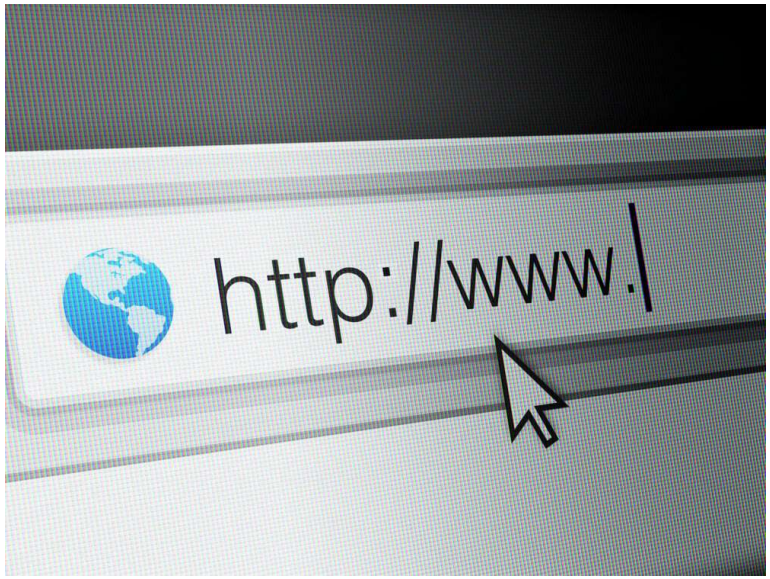
- Protocolos de rede são um conjunto de normas que permitem que qualquer máquina conectada à internet possa se comunicar com outra também já conectada na rede.



# PROTOCOLOS DE HTTP

# O QUE SÃO PROTOCOLOS DE HTTP?

- HTTP é um protocolo de transferência que possibilita que as pessoas que inserem a URL do seu site na Web possam ver os conteúdos e dados que nele existem. A sigla vem do inglês Hypertext Transfer Protocol.




**URL é o Localizador Padrão de Recursos, o endereço eletrônico de um site ou página que permite a sua localização na internet.**

# QUAL É A ORIGEM DO HTTP?

- 1989 - Tim Berners-Lee e sua equipe foram os responsáveis por inventar o HTTP original, juntamente com o HTML e a tecnologia associada para um servidor da Web e um navegador da Web baseado em texto.
- 1996 - O HTTP foi rapidamente adotado pelos principais desenvolvedores de navegadores.
- 1997 - O padrão HTTP / 1.1, como definido no RFC 2068.

# QUAL É A ORIGEM DO HTTP?

- 2007 - O HTTPbis Working Group foi formado, em parte, para revisar e esclarecer a especificação HTTP / 1.1.

- 2014 - O WG divulgou uma especificação atualizada de seis partes da obsoleta RFC 2616.

# COMO É O FUNCIONAMENTO DO HTTP?

- HTTP é um protocolo baseado em texto sem conexão.
- Isso significa que as pessoas que acessam o site da sua empresa enviam solicitações a servidores que as exibem na forma do seu site em formato de texto, imagens, e outros tipos de mídia. Depois que a solicitação é atendida por um servidor, a conexão entre o usuário e o servidor é desconectada.

# COMO É O FUNCIONAMENTO DO HTTP?

1. Se a URL pertencer a um domínio próprio, o navegador primeiro se conecta a um servidor e recuperará o endereço IP correspondente ao servidor (site);

2. O navegador se conecta ao servidor e envia uma solicitação HTTP para a página da web desejada (que, neste exemplo, é o seu site);

# COMO É O FUNCIONAMENTO DO HTTP?

3. O servidor recebe a solicitação e verifica a página desejada. Se a página existir, o servidor a mostrará. Se o servidor não conseguir encontrar a página solicitada, ele enviará uma mensagem de erro HTTP 404, ou seja, página não encontrada;

4. O navegador, então, recebe a página de volta e a conexão é fechada;

# COMO É O FUNCIONAMENTO DO HTTP?

5. Caso a página exista (e é isso que se espera), o navegador a analisa e procura outros elementos necessários para concluir a sua exibição, o que inclui seus textos, imagens e afins;

6. Para cada um desses elementos, o navegador faz conexões adicionais e solicitações HTTP para o servidor para cada elemento;

# COMO É O FUNCIONAMENTO DO HTTP?

7. Quando o navegador terminar de carregar todos os elementos, a página será carregada na janela do navegador.

Portanto, uma nova conexão deve ser feita para cada solicitação, isto é, cada vez que alguém acessa o seu site.

# QUAL É A DIFERENÇA ENTRE HTTP E HTTPS?

- O HTTP foca em apresentar a informação, se preocupando menos com a maneira como essa informação é transmitida de um ponto para outro.

- Isso quer dizer que o HTTP pode ser invadido e alterado, tornando as informações trocadas entre o site da sua empresa e a pessoa que o visita vulneráveis. É por isso que lojas virtuais não devem ter páginas construídas em HTTP.

# QUAL É A DIFERENÇA ENTRE HTTP E HTTPS?

- O HTTPS é uma extensão do HTTP. O “S” vem da palavra “secure” (“segurança” em inglês) e costuma ser oferecido pelo certificado SSL, oferecidos pela maioria dos servidores. Ele oferece uma conexão criptografada entre o servidor e o navegador, de maneira que o usuário possa inserir informações com mais segurança.

- A maioria dos e-commerces têm sites em HTTPS e o próprio navegador informa que é seguro digitar dados pessoais caso deseje realizar alguma ação, como uma compra online.

# QUAIS SOLICITAÇÕES HTTP EXISTEM?

- **GET:** essa requisição é usada para ler ou entregar dados de um servidor web. Quando realizada com sucesso, o servidor retorna o código de status 200.

- **HEAD:** uma requisição HEAD solicita ao servidor somente informações sobre o cabeçalho da página requisitada. O cabeçalho possuí diversas informações que podem ser úteis, como tamanho da página, cookies, tags e muito mais.

# QUAIS SOLICITAÇÕES HTTP EXISTEM?

- **POST:** é usado para transferir dados para o servidor (arquivos, formulários, etc.). Quando ocorre com sucesso, é retornado o código de status 201.
- **PUT:** essa requisição é utilizada quando se deseja modificar algum dado no servidor ou caso não exista nenhum dado para se atualizar, será gerado um.
- **PATCH:** essa requisição aplica modificações parciais em um dado do servidor.

# QUAIS SOLICITAÇÕES HTTP EXISTEM?

- **DELETE:** essa requisição deleta um dado especificado no servidor
- **CONNECT:** essa requisição estabelece um túnel TCP/IP com o servidor, geralmente para facilitar a comunicação criptografada com SSL (HTTPS).
- **OPTIONS:** descreve as opções de comunicações de um determinado recurso do servidor

# QUAIS SOLICITAÇÕES HTTP EXISTEM?

- **TRACE:** essa requisição envia um teste de loopback mostrando o caminho que uma requisição faz para chegar até o recurso especificado no servidor destinado. Pode ser útil para fazer debug caso alguma requisição esteja resultando em erro.

# PROTOCOLOS DE SMTP

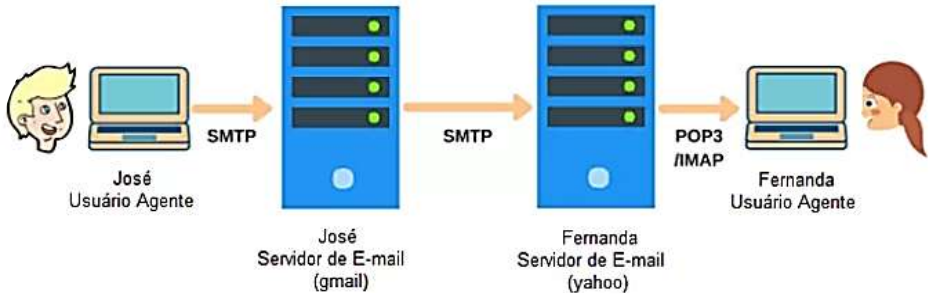
# O QUE SÃO PROTOCOLOS DE SMTP?

- O protocolo SMTP é o responsável por garantir que os e-mails que você envia diariamente cheguem aos destinatários. O objetivo principal é configurar regras de comunicação entre servidores, já que eles têm uma forma específica de se identificarem e anunciarem que tipo de comunicação estão tentando realizar.

- O Simple Mail Transfer Protocol (SMTP), ou seja, Protocolo de Transferência de Correio Simples é a tecnologia que permite que e-mails sejam enviados de um servidor para outro até serem entregues na caixa de correio.

# COMO FUNCIONA O PROTOCOLO SMTP?

- Vamos supor que José faz parte da equipe de marketing de uma empresa de Florianópolis da qual Fernanda é cliente. O e-mail profissional é a principal ferramenta de comunicação entre eles. Assim, diariamente, eles precisam trocar informações via e-mail. O e-mail de José é jose@empresa1.com, e o de Fernanda é fernanda2@empresa.com.



# COMO FUNCIONA O PROTOCOLO SMTP?

- Servidor de correio de José

José tem uma conta de e-mail no Gmail. Isso significa que há uma máquina remota no domínio gmail.com que gerencia todas as mensagens de e-mail enviadas para ele. Esta máquina é encarregada de enviar mensagens de e-mail de José para outros usuários em outros servidores. O aplicativo em execução nessa máquina remota, o Gmail, é o servidor de correio.

# COMO FUNCIONA O PROTOCOLO SMTP?

- Servidor de correio de Fernanda

Fernanda tem uma conta no Yahoo e envia suas mensagens por meio dessa máquina remota, da mesma forma que José. Dessa forma, o Yahoo é seu servidor de correio.

SMTP é o protocolo que rege a comunicação entre esses dois servidores de e-mail. Em nosso cenário particular, o servidor de correio de José desempenhará a função de um cliente SMTP, enquanto o servidor de correio de Fernanda desempenhará a função de um servidor SMTP.

# A IMPORTÂNCIA DO PROTOCOLO SMTP?

O servidor SMTP é usado para entregar e-mails transacionais e em massa de forma confiável e segura. Existem muitos provedores de serviço SMTP no mercado. Algumas das vantagens são:

# PROTOCOLOS DE POP3

# O QUE SÃO PROTOCOLOS DE POP3?

- O protocolo POP3 é um protocolo padrão da Internet usado por clientes de software de correio eletrônico local a fim de recuperar mensagens de um servidor de e-mail remoto por meio de uma conexão TCP/IP.

- Nas implementações mais simples do POP3, o servidor realmente mantém uma coleção de arquivos de texto – uma a cada conta de e-mail. Chegando uma mensagem, o servidor POP3 simplesmente a anexa ao final do arquivo do destinatário.

# COMO FUNCIONA POP3?

- O protocolo POP3 é responsável por baixar e-mails em seu computador local, mesmo offline. Por padrão, o protocolo POP3 funciona em duas portas: a Porta 110, não criptografada e POP3 padrão; e a Porta 995, usada caso queira se conectar usando POP3 com segurança.
- Toda vez que você verifica seu e-mail, o cliente de e-mail se conecta ao servidor POP3 usando a porta 110. O servidor POP3 pede um nome de conta, uma senha e, após o login, abre o arquivo de texto de modo que você o acesse.

# COMO FUNCIONA POP3?

O funcionamento do protocolo POP3 é:

1. O usuário abre sua caixa de entrada e verifica se há um novo e-mail;
2. O cliente de e-mail (Outlook, gmail e tantos outros) se conecta ao servidor POP3;
3. O software fornece nome de usuário e senha ao servidor para autenticação;

# COMO FUNCIONA POP3?

4. Conectados, o software de e-mail emite os comandos a fim de recuperar as mensagens;
5. O protocolo POP3 faz o download delas no sistema local do usuário como novos e-mails, exclui as cópias do servidor e se desconecta dele.

# VANTAGENS DO PROTOCOLO POP3

- Os e-mails são baixados para o computador do usuário;
- As mensagens podem ser lidas quando o usuário está offline;
- Abrir anexos é rápido e fácil, pois eles já foram baixados;
- Requer menos espaço de armazenamento no servidor, pois os e-mails são armazenados na máquina local;
- Muito popular, fácil de configurar e usar.

# DESVANTAGENS DO PROTOCOLO POP3

- Os e-mails não podem ser acessados de outras máquinas (a menos que configurado para isso);
- Exportar a pasta de correio local a outro cliente de e-mail ou máquina física pode ser difícil;
- As pastas de mensagens podem ser corrompidas.

# QUEM USA PROTOCOLO POP3

- **Usuários individuais:** Pessoas que preferem armazenar e-mails localmente para economizar espaço no servidor. Usuários que acessam e-mails apenas de um único dispositivo.
- **Pequenas empresas:** Empresas que não precisam manter cópias dos e-mails no servidor e desejam evitar custos com armazenamento.
- **Provedores de e-mail antigos:** Alguns provedores ainda oferecem suporte ao POP3, como Gmail, Yahoo Mail e Outlook.com, mas geralmente recomendam IMAP.

# QUEM NÃO USA PROTOCOLO POP3

- Usuários que precisam acessar e-mails em vários dispositivos (preferem IMAP).
- Empresas modernas que utilizam serviços baseados na nuvem (como Google Workspace ou Microsoft 365).
- Usuários que querem manter cópias dos e-mails no servidor e sincronizá-los automaticamente.

# A DIFERENÇA DO PROTOCOLO POP3 E O IMAP

- **POP3:** Baixa os e-mails para o dispositivo e os remove do servidor. Funciona offline, mas não sincroniza entre dispositivos.
- **IMAP:** Mantém os e-mails no servidor e sincroniza com todos os dispositivos. Requer internet para acessar e-mails, mas permite acesso de qualquer lugar.
- **Use POP3** se quiser acesso offline e economizar espaço no servidor.
- **Use IMAP** se precisar acessar os e-mails em vários dispositivos e manter tudo sincronizado.

# REFERÊNCIAS

## hostgator.com.br/blog/


1. https://www.hostgator.com.br/blog/o-que-e-protocolo-http/
2. https://www.hostgator.com.br/blog/o-que-e-protocolo-smtp/
3. https://www.hostgator.com.br/blog/entenda-protocolo-pop3/

# O QUE É JAVASCRIPT?

- JavaScript é uma linguagem de programação amplamente utilizada para criar interatividade em páginas web. Ela é executada no lado do cliente (no navegador), o que significa que permite a atualização de conteúdo, a manipulação de elementos da página e a interação com o usuário sem a necessidade de recarregar a página inteira.

# O QUE É JAVASCRIPT?

JavaScript é uma linguagem dinâmica, orientada a objetos, e pode ser usada para uma ampla variedade de tarefas, como:

- Manipulação de conteúdo HTML e CSS.
- Validação de formulários.
- Criação de animações e efeitos visuais.
- Interação com o servidor através de requisições assíncronas (AJAX).
- Desenvolvimento de aplicativos web modernos, como SPAs (Single Page Applications).

# COMO O JAVASCRIPT INTERAGE COM PROTOCOLOS

JavaScript é uma linguagem de programação essencial para interagir com esses protocolos, especialmente na comunicação do cliente com o servidor. Vamos ver como ele pode interagir com alguns desses protocolos:

**1. HTTP/HTTPS com JavaScript AJAX (Asynchronous JavaScript and XML):** AJAX é uma técnica que permite a troca de dados com o servidor de maneira assíncrona (sem recarregar a página).

# COMO O JAVASCRIPT INTERAGE COM PROTOCOLOS

Exemplo: Você pode usar o XMLHttpRequest ou a API fetch para fazer requisições HTTP/HTTPS e obter ou enviar dados para o servidor.

**Exemplo de requisição AJAX usando fetch:**

```
fetch('https://api.exemplo.com/dados')
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Erro:', error));
```

# COMO O JAVASCRIPT INTERAGE COM PROTOCOLOS

**2. Interação com API's (HTTP):** JavaScript frequentemente interage com APIs via requisições HTTP para obter ou enviar dados, como em RESTful APIs. Isso pode ser feito utilizando fetch ou bibliotecas como Axios.

**Exemplo: Enviar um POST request com JSON para uma API.**

```
fetch('https://api.exemplo.com/usuarios', {
method: 'POST',
headers: {    'Content-Type': 'application/json',  },
body: JSON.stringify({ nome: 'Márcia', idade: 15 })})
.then(response => response.json())
.then(data => console.log('Usuário criado:', data))
.catch(error => console.error('Erro:', error));
```

# ESTRUTURA BÁSICA DE UM SCRIPT JAVASCRIPT

Um script JavaScript pode ser inserido diretamente em uma página HTML ou em um arquivo separado.

1. Incorporando o JavaScript em uma Página HTML:

- No Cabeçalho (Head): Você pode colocar o script dentro da tag <head>, mas isso pode afetar a performance da página porque o script será carregado antes de qualquer outro conteúdo da página.

- Antes do Fechamento da Tag </body>: Esta é a maneira recomendada de incluir o JavaScript, pois o script será carregado após o conteúdo da página (HTML), melhorando a performance da página.

# ESTRUTURA BÁSICA DE UM SCRIPT JAVASCRIPT

## Exemplo: No Cabeçalho (Head)

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo de JavaScript</title>
    <script>
        // Código JavaScript aqui
        console.log('Olá, Mundo!');
    </script>
</head>
<body>
    <h1>Exemplo de JavaScript no Cabeçalho</h1>
</body>
</html>
```

# ESTRUTURA BÁSICA DE UM SCRIPT JAVASCRIPT

## Exemplo: Antes do Fechamento da Tag </body>

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo de JavaScript</title>
</head>
<body>
    <h1>Exemplo de JavaScript no final do corpo</h1>
    <script>
        // Código JavaScript aqui
        console.log('Olá, Mundo!');
    </script>
</body>
</html>
```

# ESTRUTURA BÁSICA DE UM SCRIPT JAVASCRIPT

O código JavaScript dentro da tag <script> pode conter diversas funcionalidades, como declarações de variáveis, funções, estruturas condicionais e loops.

```
// Declaração de variáveis
let nome = 'Márcia';
let idade = 35;

// Exibição de valores no console
console.log('Olá, meu nome é ' + nome);
console.log('Minha idade é ' + idade);

// Estrutura condicional
if (idade >= 18) {
    console.log(nome + ' é maior de idade.');
} else {
    console.log(nome + ' é menor de idade.');
}
```

# ESTRUTURA BÁSICA DE UM SCRIPT JAVASCRIPT

Declaração de variáveis: Usamos let para declarar variáveis que podem ser alteradas e armazenar dados.

Exibição de dados: O console.log() é usado para exibir informações no console do navegador. Neste caso, mostramos o nome e a idade da pessoa.

Estrutura condicional: A estrutura if é utilizada para verificar se a idade é maior ou igual a 18 e, em seguida, exibir uma mensagem correspondente.

# ESTRUTURA BÁSICA DE UM SCRIPT JAVASCRIPT

2. JavaScript Externo: Você também pode escrever o código JavaScript em um arquivo separado com a extensão .js e vinculá-lo à página HTML. Isso ajuda a manter o código organizado e facilita a reutilização.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo de JavaScript Externo</title>
</head>
<body>
    <h1>Exemplo de JavaScript Externo</h1>

    <script src="script.js"></script> <!-- Vinculando o arquivo externo -->
</body>
</html>
```

# ESTRUTURA BÁSICA DE UM SCRIPT JAVASCRIPT

- Arquivo JavaScript Externo (script.js):

```
// Código JavaScript no arquivo externo
console.log('Olá, Mundo do JavaScript Externo!');
```

- <script src="script.js"></script>: O atributo src dentro da tag <script> especifica o caminho para o arquivo JavaScript externo. Quando a página HTML é carregada, o script é automaticamente carregado e executado.

- O código JavaScript no arquivo script.js será executado da mesma forma como se estivesse dentro da tag <script> na própria página HTML.

# EXERCICIO 1

# EXERCÍCIO 1: SOMA DE DOIS NÚMEROS

**Objetivo**: Criar uma função que leia dois números fornecidos pelo usuário e exiba a soma desses números na página web.

**Passos para o exercício:**

**1.Criar uma função que receba dois números como parâmetros.**

**2.Calcular a soma desses números.**

**3.Exibir o resultado na página web.**

# EXERCÍCIO 1: SOMA DE DOIS NÚMEROS

```
<body>
    <h1>Calculadora de Soma</h1>

    <!-- Campos de entrada para os números -->
    <label for="num1">Digite o primeiro número:</label>
    <input type="number" id="num1" /><br /><br />

    <label for="num2">Digite o segundo número:</label>
    <input type="number" id="num2" /><br /><br />

    <!-- Botão para realizar a soma -->
    <button type=“button" onclick="calcularSoma()">Calcular Soma</button><br
/><br />

    <!-- Área para exibir o resultado -->
    <h2 id="resultado"></h2>
```

# EXERCÍCIO 1: SOMA DE DOIS NÚMEROS

```
<script>
  // Função que calcula a soma de dois números
  function calcularSoma() {
    // Obter os valores dos campos de entrada
    let num1 = parseFloat(document.getElementById("num1").value);
    let num2 = parseFloat(document.getElementById("num2").value);

    // Calcular a soma
    let soma = num1 + num2;

    // Exibir o resultado na página
    document.getElementById("resultado").textContent = "A soma é: " + soma;
  }
</script>
</body>
</html>
```

# DESAFIO: VERIFICAR FAIXA ETÁRIA

**Objetivo**: Criar uma função que leia a idade fornecida pelo usuário e exiba na página se a pessoa é menor de idade (0-17 anos), adulta (18-59 anos) ou idosa (60 anos ou mais).

**Passos para o exercício:**

**1.Criar uma função que receba a idade como parâmetro.**

**2.Verificar a faixa etária e definir a classificação:**

- Menor de idade: 0 a 17 anos
- Adulto: 18 a 59 anos
- Idoso: 60 anos ou mais

**3. Exibir a classificação na página web.**