Programação Web Responsiva

JavaScript

Prof. Bruno Azevedo

UNIP – Universidade Paulista

Introdução ao JavaScript

- O `JavaScript` é uma linguagem de programação interpretada, executada diretamente pelos navegadores modernos.
- É amplamente utilizada para adicionar interatividade a páginas web, como validação de formulários, animações, e manipulação de elementos HTML.
- Foi criada originalmente pela Netscape com o nome de `LiveScript`, mas foi renomeada para `JavaScript` por motivos de marketing, associando-se ao sucesso do `Java` na época.
- O `JavaScript` também pode ser usado no lado do servidor com plataformas como `Node.js`.

Java vs JavaScript

- Apesar do nome semelhante, `Java` e `JavaScript` são linguagens distintas.
- `Java` é híbrida, fortemente tipada e orientada a objetos, usada em aplicações desktop, web e mobile (como Android).
- `JavaScript` é interpretada, fracamente tipada e voltada principalmente para desenvolvimento web front-end.
- A semelhança entre os nomes é apenas superficial; a escolha de nome foi motivada por marketing, e não por relação técnica.

A Importância do JavaScript

- O JavaScript é frequentemente utilizado para adicionar interatividade a uma página web.
- A camada estrutural de uma página é definida pelo HTML, enquanto a camada de apresentação é gerenciada pelo CSS.
- A camada de **comportamento** é responsabilidade do JavaScript.
- Todos os elementos de uma página podem ser acessados e manipulados por meio de scripts.
- Podemos criar scripts que reagem à interação do usuário, permitindo a alteração dinâmica do conteúdo, dos estilos CSS ou do comportamento do navegador.

A Tag <script>

- O JavaScript é inserido em páginas HTML através da Tag <script>.
- Essa Tag pode ser colocada no <head> ou no <body> da página.
- ▷ Exemplo de uso:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      // JavaScript interno ou externo
    </script>
  </head>
  <body>
    <h1>Minha Página</h1>
  </body>
</html>
```

Primeiro Comando JavaScript: `alert`

- O comando `alert()` exibe uma janela de alerta no navegador.
- Pode ser usado para mostrar mensagens simples ao usuário.
- É útil para testes e notificações.
- Sintaxe básica:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      alert("Olá mundo!");
    </script>
  </head>
  <body>
    <h1>Minha Página</h1>
  </body>
</html>
```

Formas de Adicionar JavaScript

- O `JavaScript` pode ser adicionado a páginas HTML de três maneiras principais:
  - Código inline: inserido diretamente na Tag HTML.
  - Código interno (embutido): inserido no HTML utilizando a tag `<script>`.
  - Código externo: também utiliza a tag `<script>`, mas o código é escrito em um arquivo separado com a extensão `.js`, que é referenciado no HTML por meio do atributo `src`.

JavaScript *Inline*

- Um código `JavaScript` inline envolve a inserção do código diretamente nos atributos dos elementos HTML.
- Esse método é ideal para scripts simples e pequenos ou manipuladores de eventos.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <button onclick="alert('Script inline funcionando!');">
      Clique aqui!
    </button>
  </body>
</html>
```

- No HTML, ao usar aspas dentro de um atributo (como `onclick`), é necessário usar aspas simples (') ou então escapar as aspas duplas (\") para evitar erros de sintaxe.

JavaScript Interno

- Um código JavaScript interno é aquele que está diretamente incluído no arquivo HTML.
- Utiliza a Tag <script> contendo o código dentro.
- Ideal para scripts curtos e específicos da página.

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      alert("Script interno funcionando!");
    </script>
  </head>
  <body>
  </body>
</html>
```

JavaScript Externo

- Um código JavaScript externo é armazenado em um arquivo separado com extensão .js.
- A Tag <script> utiliza o atributo src para referenciar o arquivo.
- É mais organizado e facilita a reutilização do código em várias páginas.

```
<!-- Arquivo HTML -->
<!DOCTYPE html>
<html>
  <head>
    <script src="script.js" defer></script>
  </head>
  <body>
  </body>
</html>

// Arquivo script.js
alert("Script externo funcionando!");
```

Sempre Use o Atributo `defer`

- Ao usar `defer`, o script é carregado em paralelo com o conteúdo HTML e é executado somente quando o DOM estiver completamente carregado.
- Garante que o JavaScript tenha acesso completo à estrutura da página, evitando erros com a manipulação de elementos HTML que ainda não foram carregados.
- Sem o `defer`, o script pode bloquear o carregamento da página, fazendo com que o conteúdo HTML demore mais para ser exibido.
- A alternativa seria mover o script para o final da tag `<body>`. No entanto, isso não garante que o script será carregado em paralelo com o HTML e pode não ser tão eficiente.

Eventos

- Em JavaScript, eventos são ações que ocorrem na página, geralmente provocadas pelo usuário.
  - Exemplos de eventos incluem: cliques de mouse, pressionar teclas, mover o cursor, focar em campos, enviar formulários, entre outros.
- Podemos capturar esses eventos e executar códigos JavaScript quando eles ocorrem.

Eventos Comuns

- `onclick`: quando o usuário clica com o mouse.
- `ondblclick`: quando o usuário dá dois cliques.
- `onmouseover`: quando o ponteiro do mouse passa sobre o elemento.
- `onmousedown`: quando o botão do mouse é pressionado.
- `onmouseup`: quando o botão do mouse é solto.
- `onkeypress`: quando uma tecla é pressionada e solta.
- `onkeydown`: quando uma tecla é pressionada.
- `onkeyup`: quando uma tecla é solta.
- `onsubmit`: antes de enviar um formulário.

Evento onclick

- O evento `onclick` ocorre quando o usuário clica sobre um elemento.
- Pode ser usado para responder a interações como botões ou imagens clicáveis.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Exemplo onclick</title>
  </head>
  <body>
    <button onclick="alert('Você clicou no Botão!')">Clique aqui</button>
  </body>
</html>
```

Evento ondblclick

- O evento `ondblclick` é disparado quando o usuário clica duas vezes rapidamente sobre um elemento.
- Pode ser usado para ações que requerem confirmação, como abrir um item.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo ondblclick</title>
  </head>
  <body>
    <p ondblclick="alert('Clique duplo detectado!')">
      Dê dois cliques aqui
    </p>
  </body>
</html>
```

Evento onmouseover

- O evento onmouseover ocorre quando o ponteiro do mouse passa sobre um elemento.
- Pode ser usado para destacar elementos, mostrar dicas ou iniciar animações.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo onmouseover</title>
  </head>
  <body>
    <p onmouseover="alert('Mouse passou sobre o texto!')">
      Passe o mouse aqui.
    </p>
  </body>
</html>
```

Evento onmouseout

- O evento onmouseout ocorre quando o ponteiro do mouse sai de cima de um elemento.
- Pode ser usado para desfazer destaques, esconder dicas ou interromper animações.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo onmouseout</title>
  </head>
  <body>
    <p onmouseout="alert('Mouse saiu do texto!')">
      Passe o mouse aqui e depois saia.
    </p>
  </body>
</html>
```

Evento onmousedown

- O evento onmousedown é acionado quando o botão do mouse é pressionado sobre um elemento.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo onmousedown</title>
  </head>
  <body>
    <div onmousedown="alert('Mouse pressionado!')">
      Pressione aqui
    </div>
  </body>
</html>
```

Evento onmouseup

- O evento onmouseup é disparado quando o botão do mouse é solto sobre um elemento.
- É útil para detectar o fim de uma ação de clique.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo onmouseup</title>
  </head>
  <body>
    <div onmouseup="alert('Mouse solto!')">
      Clique e solte o botão aqui
    </div>
  </body>
</html>
```

Evento onkeypress

- O evento `onkeypress` ocorre quando o usuário pressiona uma tecla que gera um caractere.
- Não é acionado por teclas como `Shift`, `Ctrl` ou `Alt`.
- Utilizado para capturar letras, números e símbolos digitados.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo onkeypress</title>
  </head>
  <body>
    <input type="text" onkeypress="alert('Tecla pressionada!')">
  </body>
</html>
```

Evento onkeydown

- O evento `onkeydown` ocorre quando o usuário pressiona qualquer tecla do teclado.
- É acionado no momento em que a tecla é pressionada, antes mesmo de ser liberada.
- Pode detectar todas as teclas, incluindo `Shift`, `Ctrl`, `Alt`, `Setas`, etc.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo onkeydown</title>
  </head>
  <body>
    <input type="text" onkeydown="alert('Tecla para baixo!')">
  </body>
</html>
```

Evento onkeyup

- O evento onkeyup ocorre quando o usuário solta uma tecla do teclado.
- É acionado depois que a tecla foi pressionada e liberada.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo onkeyup</title>
  </head>
  <body>
    <input type="text" onkeyup="alert('Tecla liberada!')">
  </body>
</html>
```

Evento onsubmit

- O evento onsubmit ocorre quando um formulário é enviado.
- Pode ser usado para validar dados, impedir o envio ou mostrar mensagens antes do envio.

▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
      <title>Exemplo onsubmit</title>
  </head>
  <body>
    <form onsubmit="alert('Formulário enviado!')">
      <input type="text" placeholder="Seu nome" required>
      <button type="submit">Enviar</button>
    </form>
  </body>
</html>
```

Evento oninput

- O evento oninput ocorre sempre que o valor de um campo de entrada é alterado pelo usuário.
- É útil para responder a mudanças em tempo real, como validação ou atualização de conteúdo dinâmico.

▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo oninput</title>
    <script defer>
      function mostraTexto(valor) {
        alert("Você digitou: " + valor); }
    </script>
  </head>
  <body>
    <input type="text" oninput="mostraTexto(this.value)">
  </body>
</html>
```

Funções

- Uma `função` é um bloco de código que pode ser reutilizado em diferentes partes do programa.
- Pode receber parâmetros e retornar valores.
- Em JavaScript, as funções podem ser declaradas de várias formas: tradicionais, anônimas e arrow functions (funções seta).
- As funções podem ser definidas de forma interna (dentro do código HTML) ou externa (em arquivos JavaScript separados).

Vantagens de Usar Funções

- Evita repetição de código.
- Facilita a manutenção e testes.
- Permite dividir tarefas complexas em partes menores.
- Torna o código mais legível.

Declaração e Chamada de Funções

- Para declarar uma função em JavaScript, utiliza-se a seguinte sintaxe:

```
function nomeDaFuncao(parâmetros) {
  // corpo da função
}
```

- A função pode ou não retornar um valor.
- Parâmetros são os valores passados para a função no momento de sua chamada.
  - Os parâmetros podem ser nenhum ou múltiplos, separados por vírgulas.
- Exemplo de uma função que soma dois números e retorna o resultado:

```
function soma(a, b) {
  return a + b;
}
```

Declaração e Chamada de Funções

- Funções podem ser chamadas em qualquer lugar no código, desde que já tenham sido declaradas.
  - ▷ Exemplo de chamada de função:

```
soma(1024, 128);
```

- Em JavaScript, também é possível criar funções anônimas, ou seja, funções sem nome.
  - ▷ Exemplo de uma função anônima sendo definida e sua referência sendo atribuída a uma variável:

```
multiplicar = function(x, y) {
  return x * y;
};
```

- Posteriormente, podemos usar essa referência para chamar a função.

Exemplo de Declaração e Chamada de função

- Exemplo completo de declaração e chamada:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function saudacao() {
        alert("Olá, seja bem-vindo!");
      }
    </script>
  </head>
  <body>
    <button onclick="saudacao()">Clique aqui</button>
  </body>
</html>
```

Parâmetros em Funções

- Funções podem receber valores como parâmetros para realizar cálculos ou operações.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function soma(a, b) {
        alert("Resultado: " + (a + b));
      }
    </script>
  </head>
  <body>
    <button onclick="soma(256, 128)">Soma 256 + 128</button>
  </body>
</html>
```

Variáveis

- Uma variável pode ser vista como um contêiner para armazenar informações.
- Você define um nome para a variável e atribui um valor a ela, que pode ser um número, uma string, entre outros tipos de dados.
- Para declarar uma variável, usamos a palavra-chave **let**.

```
let minhaVariavel = 5;
```

- Vamos analisar o que foi feito.
- A palavra-chave **let** é usada para declarar a variável, seguida do nome da variável e do operador de atribuição (=).
- Finalmente, atribuímos o valor cinco à variável. Finalizamos a instrução com o ponto-e-vírgula, que indica o fim da declaração.

Variáveis

- Vamos aplicar o que aprendemos em um exemplo prático:

```
<!DOCTYPE html>
<html>
  <body>
    <script defer>
      let x = 2322;
      let y = 197;
      let resultado = x - (3*y - 47);
      alert("O resultado é: " + resultado);
    </script>
  </body>
</html>
```

- Neste código, declaramos duas variáveis, realizamos uma operação matemática e mostramos o resultado em uma caixa de alerta.

Tipos de Dados

- As variáveis podem armazenar diferentes tipos de dados.
- ▷ Exemplos comuns incluem números inteiros, números reais, strings (sequências de caracteres) e valores booleanos.

```
let numeroInteiro = 42; // Tipo número inteiro
let numeroReal = 3.14;  // Tipo número real
let nome = "Alice";     // Tipo string
let ligado = true;       // Tipo booleano
```

Incorporando Variáveis em Strings

- É possível incorporar variáveis dentro de strings de diferentes maneiras.
- Uma forma comum é utilizando o operador de concatenação +.

```
let nome = "Alice";
let saudacao = "Olá, " + nome + "!";
```

- Nesse exemplo, a variável `nome` é concatenada à string "Olá, " e à string "!".
- O resultado é a saudação completa.

DOM (Document Object Model)

- O DOM (Document Object Model) é uma interface de programação para documentos HTML e XML.
- Cada elemento HTML, atributo e texto são tratados como objetos JavaScript, que possuem propriedades e métodos.
- **Esses objetos podem ser manipulados usando JavaScript**.
- O DOM permite manipulações dinâmicas, como alterar conteúdo ou adicionar novos elementos sem recarregar a página.

Estrutura Hierárquica

- O DOM representa a estrutura de um documento como uma árvore de nós.
- Cada nó é um elemento, atributo ou texto.
- A raiz da árvore é o próprio documento (representado pelo nó `document`), e todos os outros elementos da página são representados como nós filhos dessa raiz.
- ▷ Exemplo:

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo de DOM</title>
  </head>
  <body>
    <h1 id="titulo">Olá, Mundo!</h1>
    <p>Este é um parágrafo.</p>
    <button id="btn">Clique aqui</button>
  </body>
</html>
```

```
document
  |--- html
  |     |--- head
  |     |     |--- title
  |     |--- body
  |           |--- h1 (id="titulo")
  |           |--- p
  |           |--- button (id="btn")
```

- Você pode acessar e manipular qualquer parte dessa estrutura usando JavaScript.

Métodos e Propriedades do DOM

- O DOM oferece uma grande variedade de métodos e propriedades para manipular os elementos HTML com JavaScript.
- Há muitos recursos disponíveis para estilização, conteúdo, estrutura, eventos, entre outros.
- Precisamos conhecer os métodos e as propriedades, de modo que possamos trabalhar com elas.
- Aprenderemos alguns métodos e propriedades tipicamente usados.
- Para aprofundar, consulte:
  - `https://www.w3schools.com/js/js_htmldom.asp`

Métodos para Acessar Elementos no DOM

- JavaScript oferece vários métodos para acessar elementos no DOM.
- Alguns exemplos:
  - Por ID: `document.getElementById()`: obtém a referência do elemento com o id especificado.
  - ▷ Exemplo:

```
let varTitulo = document.getElementById("titulo");
```

  - Por Tag: `document.getElementsByTagName()`: obtém uma lista de referências de todos os elementos de uma determinada Tag.
  - ▷ Exemplo:

```
let paragrafos = document.getElementsByTagName("p");
```

  - Por Classe: `document.getElementsByClassName()`: obtém uma lista de referências de todos os elementos de uma determinada classe.
  - ▷ Exemplo:

```
let botoes = document.getElementsByClassName("btn");
```

getElementById() e innerHTML

- O método `getElementById()` retorna uma referência para o elemento.
- A propriedade `innerHTML` representa o conteúdo HTML interno de um elemento.
- Pode ser usada tanto para obter como para alterar o conteúdo de um elemento.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function exibeTexto() {
        let elemento = document.getElementById("mensagem");
        alert("Texto acessado: " + elemento.innerHTML);
      }
    </script>
  </head>
  <body>
    <p id="mensagem">Texto original</p>
    <button onclick="exibeTexto()">Acessar e exibir texto</button>
  </body>
</html>
```

- Neste exemplo, usamos `getElementById()` para acessar o elemento com

getElementById() e textContent

- A propriedade `textContent` representa o conteúdo textual interno de um elemento.
- É mais segura que `innerHTML`, pois evita a inserção acidental de código HTML ou scripts.
- Usada para obter ou modificar texto simples dentro de elementos HTML.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function exibeTexto() {
        let elemento = document.getElementById("mensagem");
        alert("Texto acessado: " + elemento.textContent);
      }
    </script>
  </head>
  <body>
    <p id="mensagem">Texto original</p>
    <button onclick="exibeTexto()">Acessar e exibir texto</button>
  </body>
</html>
```

getElementsByTagName()

- O método `getElementsByTagName()` retorna uma coleção de elementos com base na tag informada.
- O resultado é uma coleção (tipo `HTMLCollection`), semelhante a um vetor.
- Podemos acessar os elementos individualmente usando índice, como `paragrafos[0]`.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo getElementsByTagName</title>
    <script defer>
      function estilizarParagrafos() {
        let paragrafos = document.getElementsByTagName("p");
        for (let i = 0; i < paragrafos.length; i+=2) {
          paragrafos[i].style.color = "blue";
          paragrafos[i+1].style.color = "red";
        }
      }
    </script>
  </head>
  <body>
    <p>Parágrafo 1</p>
    <p>Parágrafo 2</p>
    <p>Parágrafo 3</p>
    <p>Parágrafo 4</p>
    <p>Parágrafo 5</p>
    <p>Parágrafo 6</p>
    <button onclick="estilizarParagrafos()">Estilizar textos</button>
  </body>
</html>
```

Manipulando Elementos no DOM

- Após obtermos acesso a um elemento, podemos modificá-lo utilizando diversos métodos e propriedades disponíveis.
- Podemos modificar seu conteúdo, atributos, estilos, entre outros.
- Vamos conhecer.

Modificando um Parágrafo

- Podemos alterar o **conteúdo** de um parágrafo usando a propriedade `innerHTML`.
- Podemos alterar diferentes estilos, como a cor do texto.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function modificarTexto() {
        let p = document.getElementById("meuPar");
        p.innerHTML = "Este parágrafo foi modificado!";
      }
      function modificarCor() {
        let p = document.getElementById("meuPar");
        p.style.color = "blue";
      }
    </script>
  </head>
  <body>
    <p id="meuPar">Parágrafo original.</p>
    <button onclick="modificarTexto()">Modificar Texto</button>
    <button onclick="modificarCor()">Modificar Cor</button>
  </body>
</html>
```

Modificando uma Imagem

- Podemos modificar o atributo `src` de uma imagem.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function trocarImagem() {
        let img = document.getElementById("foto");
        img.src = "img_chania.jpg";
      }
    </script>
  </head>
  <body>
    <img id="foto" src="pic_trulli.jpg">
    <br>
    <button onclick="trocarImagem()">Trocar imagem</button>
  </body>
</html>
```

- A imagem exibida na tela é trocada dinamicamente ao clicar no botão.

Propriedade style

- Podemos alterar dinamicamente o estilo (CSS) de um elemento HTML com JavaScript usando a propriedade `style`.

▷ Exemplo completo:

```
<!DOCTYPE html>
<html>
 <head>
  <script defer>
    function estilizar() {
      let elemento = document.getElementById("meuElemento");
      elemento.style.color = "red";                // Cor do texto
      elemento.style.fontSize = "20px";            // Tamanho da fonte
      elemento.style.backgroundColor = "yellow"; // Cor de fundo
      elemento.style.padding = "10px";             // Espaçamento interno
    }
  </script>
 </head>
 <body>
  <p id="meuElemento">Este texto será estilizado.</p>
  <button onclick="estilizar()">Aplicar Estilo</button>
 </body>
</html>
```

Exemplo contendo onmouseDown, onmouseup e style

- xemplo que utiliza os eventos onmousedown e onmouseup, juntamente com a propriedade style.
- Modificaremos a estilização do elemento com JavaScript.

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function mouseDown(){
        document.getElementById("meuPar").style.color = "red";
      }
      function mouseUp(){
        document.getElementById("meuPar").style.color = "green";
      }
    </script>
  <body>
    <p id="meuPar" onmousedown="mouseDown()" onmouseup="mouseUp()">
      Clique neste texto e mantenha o clique pressionado.
      Repare que o texto mudará de cor Preto para Vermelho.
      Ao soltar o clique, o texto ficará na cor Verde.
    </p>
  </body>
</html>
```

- A propriedade style é usada para acessar ou modificar os estilos CSS de um

Exemplo com onmouseover, onmouseout e style

- Exemplo que utiliza os eventos `onmouseover` e `onmouseout`, juntamente com a propriedade `style`.

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function destacar() {
        document.getElementById("texto").style.fontWeight = "bold";
      }
      function removerDestaque() {
        document.getElementById("texto").style.fontWeight = "normal";
      }
    </script>
  </head>
  <body>
    <p id="texto" onmouseover="destacar()" onmouseout="removerDestaque()">
      Passe o mouse sobre este texto para destacá-lo.
    </p>
  </body>
</html>
```

- A propriedade `fontWeight` é usada para definir o peso da fonte (normal, bold etc.).

Propriedade value

- A propriedade `value` representa o valor atual de um campo de formulário, como `<input>` ou `<textarea>`.
- Pode ser usada tanto para obter quanto para alterar o valor desses campos.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function alterarValor() {
        // Altera o texto do input
        document.getElementById("entrada").value = "Novo texto!";
      }
    </script>
  </head>
  <body>
    <input type="text" id="entrada" value="Texto inicial">
    <button onclick="alterarValor()">Alterar Valor</button>
  </body>
</html>
```

Validação com `value`

- A propriedade `value` pode ser usada para validar o conteúdo de um campo.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function validarCampo() {
        let nome = document.getElementById("nome").value;
        if (nome === "") {
          alert("O campo nome é obrigatório.");
        } else {
          alert("Olá, " + nome + "!");
        }
      }
    </script>
  </head>
  <body>
    <input type="text" id="nome" placeholder="Digite seu nome">
    <button onclick="validarCampo()">Enviar</button>
  </body>
</html>
```

- Utilizamos a estrutura condicional `if` de JavaScript para verificar se uma condição é verdadeira.
- Quando a condição não é satisfeita, o bloco `else` pode ser usado para executar uma ação alternativa.

Usando this em Eventos HTML

- Usar `this` permite acessar ou modificar diretamente o elemento sem precisar usar funções como `getElementById`, `getElementsByClassName` ou `getElementsByTagName`.
- Isso simplifica o código e torna as funções mais reutilizáveis para múltiplos elementos.
- ▷ Exemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <script defer>
      function mudarCor(elemento) {
        elemento.style.color = "blue";
      }
    </script>
  </head>
  <body>
    <p onmouseover="mudarCor(this)">Passe o mouse aqui para mudar minha co
  </body>
</html>
```

- A função recebe o elemento via `this` e altera sua cor, sem buscar o elemento pelo `id` ou outro seletor.

Atividades Práticas 13 e 14

- Façam as atividades 13 e 14.