UNIPAC
Barbacena

Centro Universitário Presidente Antônio Carlos
Programação para Internet

Introdução ao JavaScript

Felipe Roncalli de Paula Carneiro

felipecarneiro@unipac.br

O que vamos aprender nessa aula

- Introdução;
- Configuração de ambiente
- Variáveis e tipos de dados;
- Controle de Fluxo;
- Funções;
- Loops;
- DOM (Data Object Model)

Importância do JS no desenvolvimento web

Interatividade do usuário: JavaScript permite que os desenvolvedores criem interfaces de usuário interativas e responsivas. Isso inclui funcionalidades como validação de formulários, botões de clique, animações, carregamento dinâmico de conteúdo e muito mais.

Importância do JS no desenvolvimento web

Manipulação do DOM: Com JavaScript, é possível selecionar, modificar e manipular elementos HTML e CSS do Document Object Model (DOM). Isso significa que você pode alterar o conteúdo e o estilo de uma página em tempo real, sem a necessidade de recarregar a página.

Importância do JS no desenvolvimento web

Frameworks e Bibliotecas: JavaScript é a base para uma ampla gama de frameworks e bibliotecas, como jQuery, React, Angular e Vue.js. Essas ferramentas facilitam o desenvolvimento de aplicações web complexas e otimizam o processo de construção.

Importância do JS no desenvolvimento web

Validação de Dados: JavaScript permite a validação de dados no lado do cliente antes que sejam enviados ao servidor, melhorando a experiência do usuário ao fornecer feedback instantâneo sobre erros em formulários, por exemplo.

Importância do JS no desenvolvimento web

Desenvolvimento Web Full Stack: Muitos desenvolvedores trabalham com JavaScript tanto no front-end (interface do usuário) quanto no back-end (lógica do servidor). Isso torna o JavaScript uma escolha versátil para criar aplicativos web completos.

Importância do JS no desenvolvimento web

Acessibilidade e SEO: Através do JavaScript, é possível melhorar a acessibilidade das páginas web, tornando-as mais inclusivas para pessoas com deficiências. Além disso, técnicas adequadas de uso de JavaScript podem impactar positivamente o SEO (Search Engine Optimization) e a indexação de conteúdo pelos mecanismos de busca.

Importância do JS no desenvolvimento web

Comunidade Ativa: JavaScript possui uma comunidade vasta e ativa de desenvolvedores, o que significa que há uma grande quantidade de recursos, tutoriais, fóruns e exemplos disponíveis para aprender e aprimorar suas habilidades.

Configuração de Ambiente JavaScript

Editor de Texto ou IDE : Escolha um editor de texto ou uma IDE para escrever seu código JavaScript. Algumas opções populares incluem Visual Studio Code, Sublime Text, Atom e WebStorm.

Navegador Web: Qualquer navegador web moderno, como Google Chrome, Mozilla Firefox ou Microsoft Edge, já possui um console de desenvolvimento embutido que permite executar código JavaScript e depurar erros.

Configuração de Ambiente JavaScript

HTML: Crie um arquivo HTML básico que será a base para a execução do seu código JavaScript. Use um editor de texto para criar um novo arquivo chamado "index.html" e insira o seguinte código:

```
<!DOCTYPE html>
<html>
<head>
    <title>Minha Primeira Página JavaScript</title>
</head>
<body>
<!-- Seu código JavaScript será inserido aqui -->
</body>
</html>
```

Configuração de Ambiente JavaScript

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Minha Primeira Página JavaScript</title>
5  </head>
6  <body>
7  <!-- Seu código JavaScript será inserido aqui -->
8  <script>
9      console.log("Olá, mundo!");
10 </script>
11 </body>
12 </html>
```

Como incluir código JavaScript

Incorporar código JavaScript **inline** significa incluir o código diretamente dentro da marcação HTML da página. Você pode fazer isso usando a tag <script>... </script> dentro do elemento <head> ou <body> do seu arquivo HTML. Aqui está um exemplo:

Como incluir código JavaScript

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Incorporação Inline</title>
5      <script>
6          function saudacao() {
7              alert("Olimpia mandou um abraço!!!!");
8          }
9      </script>
10 </head>
11 <body>
12     <button onclick="saudacao()">Clique Aqui</button>
13 </body>
14 </html>
```

Como incluir código JavaScript

Incorporar código JavaScript em **arquivos externos** é uma prática recomendada para manter o código organizado e reutilizável. Você pode criar um arquivo separado com extensão .js contendo seu código JavaScript e, em seguida, fazer referência a esse arquivo em sua página HTML. Aqui está como fazer isso:

Como incluir código JavaScript

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Arquivos Externos</title>
    <script src="script.js"></script>
</head>
<body>
    <button onclick="saudacao()">Clique Aqui</button>
</body>
</html>
```

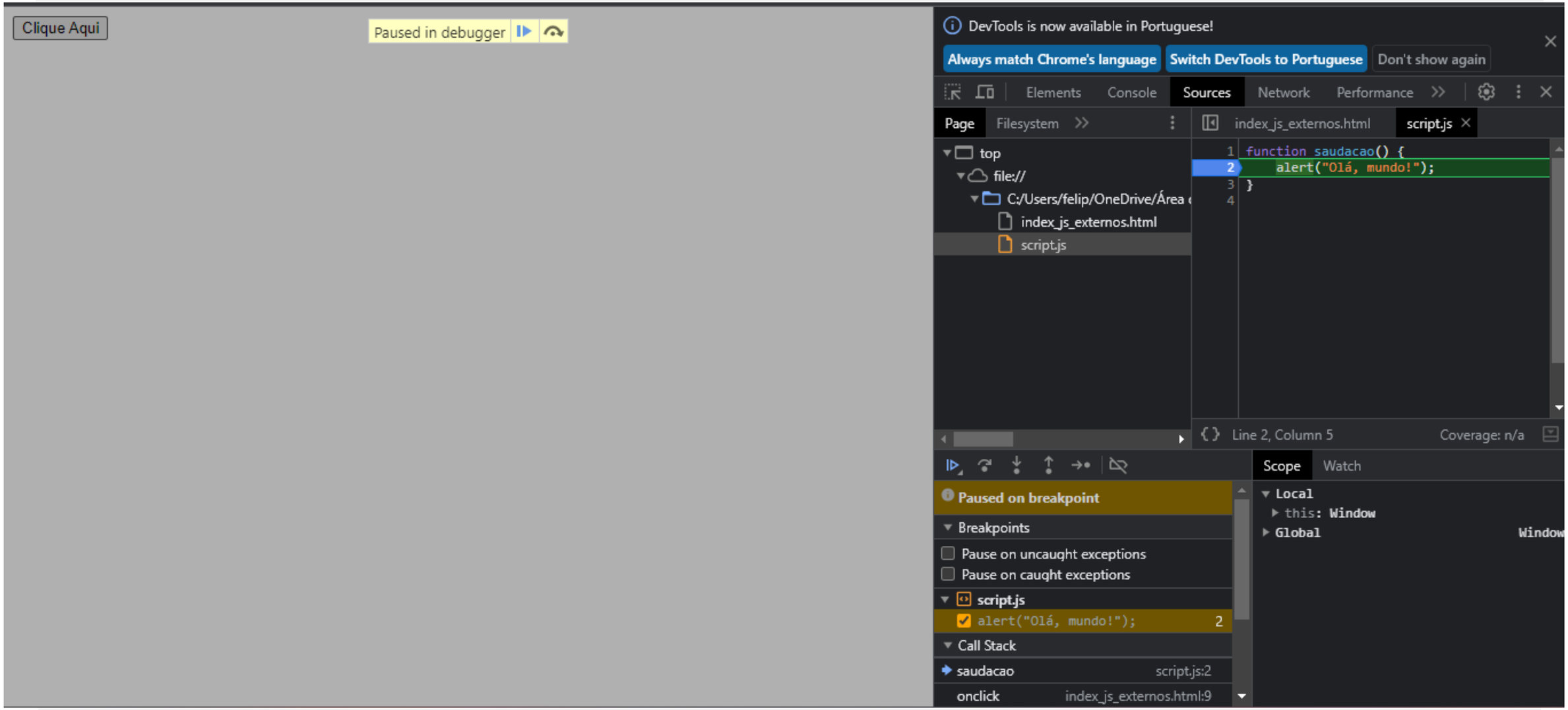
```javascript
function saudacao() {
    alert("Olá, mundo!");
}
```

Como incluir código JavaScript

Neste exemplo, o código JavaScript está no arquivo meu-script.js. A tag <script src="script.js"></script> no <head> da página HTML faz referência ao arquivo externo.

A incorporação de código JavaScript **inline** é útil para pequenos scripts ou experimentos rápidos, mas quando seu código começa a ficar mais complexo, é recomendável usar **arquivos externos** para manter uma estrutura organizada e facilitar a manutenção do código.

Como Depurar Código JavaScript no navegador



Variáveis e Constantes

Variáveis são contêineres nomeados que você pode usar para armazenar valores. Em JavaScript, você pode declarar uma variável usando a palavra-chave **var, let** ou declarar uma constante através da palavra-chave **const**. Podemos considerar a linguagem como **fracamente tipada**.

```
var idade = 25;      // Declarando uma variável chamada "idade"
let nome = "João";   // Declarando uma variável chamada "nome"
const PI = 3.14;     // Declarando uma constante chamada "PI" e
```

Variáveis - Tipos de Dados

JavaScript tem diversos tipos de dados que podem ser armazenados em variáveis. Alguns dos principais tipos de dados incluem:

Número (Number): Pode representar números inteiros ou decimais.

```
1   var idade = 25;          // Número inteiro
2   var preco = 19.99;       // Número decimal
```

Variáveis - Tipos de Dados

String: Representa texto. Deve estar entre aspas simples (' ') ou aspas duplas (" ").

```
1  var nome = "Maria";    // String com aspas duplas
2  var sobrenome = 'Silva'; // String com aspas simples
```

Booleano (Boolean): Pode ser true (verdadeiro) ou false (falso), usado para lógica condicional.

```
1  var estaChovendo = true;    // Verdadeiro
2  var diaEnsolarado = false; // Falso
```

Variáveis - Tipos de Dados

String: Representa texto. Deve estar entre aspas simples (' ') ou aspas duplas (" ").

```
1  var nome = "Maria";    // String com aspas duplas
2  var sobrenome = 'Silva'; // String com aspas simples
```

Booleano (Boolean): Pode ser true (verdadeiro) ou false (falso), usado para lógica condicional.

```
1  var estaChovendo = true;    // Verdadeiro
2  var diaEnsolarado = false; // Falso
```

Variáveis - Tipos de Dados

Array: Uma lista ordenada de valores, que podem ser de tipos diferentes.

```
var numeros = [1, 2, 3, 4, 5];     // Array de números
var frutas = ["maçã", "banana", "laranja"]; // Array de strings
```

Objeto (Object): Uma coleção de pares de chave-valor, onde os valores podem ser de tipos diferentes.

```
var pessoa = {
    nome: "Joana",
    idade: 30,
    casada: false
};
```

Variáveis - Tipos de Dados

Null: Representa a ausência intencional de valor.

```
var valor = null;   // Valor nulo
```

Undefined: Representa uma variável que foi declarada, mas não foi atribuída a nenhum valor.

```
var preco;   // Variável declarada, mas não atribuída (undefined)
```

Controle de Fluxo

O controle de fluxo em JavaScript é fundamental para tomar decisões e executar diferentes blocos de código com base em condições específicas. Existem várias estruturas de controle de fluxo, como if, else if, else e switch.

Controle de Fluxo

O **if** é usado para executar um bloco de código se a condição especificada for verdadeira.

```
1   var idade = 18;
2   if (idade >= 18) {
3       console.log("Você é maior de idade.");
4   }
```

Controle de Fluxo

O **else if** é usado para testar múltiplas condições encadeadas. Ele é executado se a condição anterior não for verdadeira.

```
var nota = 75;
if (nota >= 90) {
    console.log("Nota A");
} else if (nota >= 80) {
    console.log("Nota B");
} else if (nota >= 70) {
    console.log("Nota C");
} else {
    console.log("Nota abaixo de C");
}
```

Controle de Fluxo

O **else** é usado para executar um bloco de código quando a condição do **if** ou do **else if** não é verdadeira.

```
1   var idade = 15;
2   if (idade >= 18) {
3       console.log("Você é maior de idade.");
4   } else {
5       console.log("Você é menor de idade.");
6   }
```

Controle de Fluxo

O **switch** é usado para testar uma expressão em diferentes casos e executar o bloco de código correspondente ao caso encontrado.

```
var diaDaSemana = 3;
switch (diaDaSemana) {
    case 1:
        console.log("Domingo");
        break;
    case 2:
        console.log("Segunda-feira");
        break;
    case 3:
        console.log("Terça-feira");
        break;
    // ... outros casos ...
    default:
        console.log("Dia inválido");
}
```

Loop

Trabalhar com loops em JavaScript é essencial para executar um bloco de código várias vezes, seja para iterar sobre arrays, realizar operações repetidas ou processar conjuntos de dados. Existem três principais tipos de loops em JavaScript: **for**, **while** e **do-while**

Loop - For

O loop **for** é usado quando você sabe exatamente quantas vezes deseja executar um bloco de código.

```
1  for (var i = 0; i < 5; i++) {
2      console.log("Iteração #" + i);
3  }
```

Neste exemplo, o loop for executará o bloco de código cinco vezes, incrementando a variável i a cada iteração.

Loop - while

O loop **while** é usado quando você quer executar um bloco de código enquanto uma condição for verdadeira.

```
1   var contador = 0;
2   while (contador < 3) {
3       console.log("Contador: " + contador);
4       contador++;
5   }
```

O loop while executará o bloco de código até que a condição (contador < 3) seja falsa.

Loop - do-while

Semelhante ao **while**, o loop **do-while** executa um bloco de código enquanto uma condição é verdadeira, mas a verificação da condição ocorre após a primeira execução.

```
1   var num = 0;
2   do {
3       console.log(num);
4       num++;
5   } while (num < 3);
```

Looping por Arrays:

Para percorrer elementos em um **array**, você pode usar **loops**, como o **loop for**, ou a estrutura **for...of**, introduzida no ES6 (ECMAScript 2015).

```
1  var frutas = ["maçã", "banana", "laranja"];
2  for (var i = 0; i < frutas.length; i++) {
3      console.log(frutas[i]);
4  }
5  // ou usando for...of
6  for (var fruta of frutas) {
7      console.log(fruta);
8  }
```

Loop

Lembre-se de que loops também podem ser aninhados para criar lógica mais complexa. Use loops de acordo com o contexto e a necessidade do seu código para automatizar repetições de tarefas.

Funções

Trabalhar com funções é uma parte fundamental do desenvolvimento em JavaScript. As funções permitem que você defina blocos de código reutilizáveis que podem ser chamados várias vezes.

Funções - Definição

Você pode definir uma função usando a palavra-chave function, seguida pelo nome da função, parênteses () que podem conter parâmetros e um bloco de código {} com as instruções a serem executadas.

```
1  // Definindo uma função chamada "saudacao"
2  function saudacao(nome) {
3      console.log("Olá, " + nome + "!");
4  }
```

Funções - Chamada

Depois de definir a função, você pode chamá-la, passando os argumentos necessários entre os parênteses.

```
1  saudacao("Maria"); // Chama a função e passa o argumento "Maria"
2
```

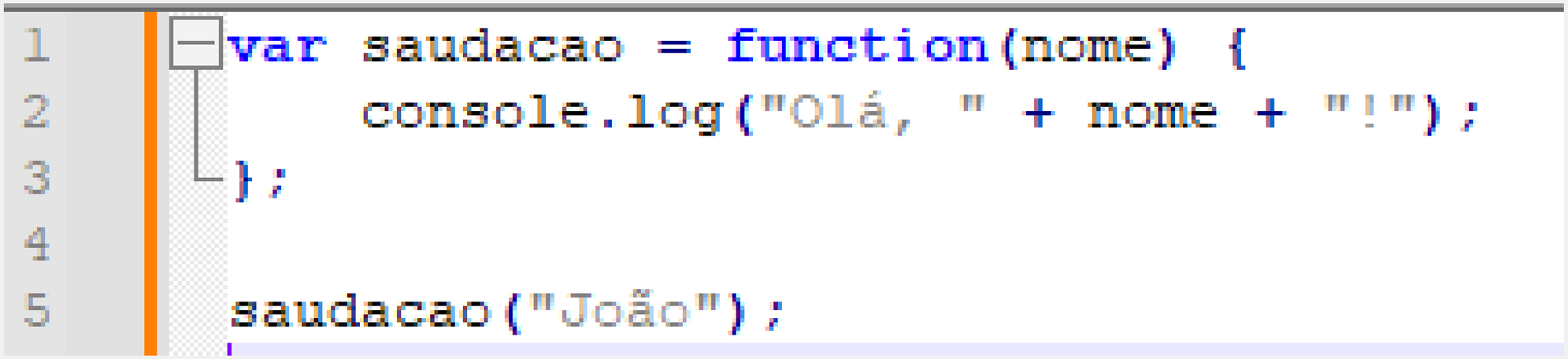
Funções - Retorno de Valores

As funções também podem retornar valores usando a palavra-chave return. Isso permite que você obtenha resultados calculados ou processados.

```
1  function soma(a, b) {
2      return a + b;
3  }
4
5  var resultado = soma(3, 5); // Atribui o resultado da função a "resultado"
6  console.log(resultado);     // Exibe o valor 8
```

Funções - Anônimas e Expressões de Função

Definir funções com nomes, você pode criar funções anônimas e atribuí-las a variáveis. Isso é comum ao usar callbacks em JavaScript.



Funções - Arrow Functions:

Uma forma mais concisa de escrever funções anônimas foi introduzida com as Arrow Functions (funções de seta).

```
1   const saudacao = nome => console.log("Olá, " + nome + "!");
2   saudacao("Maria");
```

DOM - Document Object Model

O **Document Object Model** (**DOM**) é uma representação em memória de um documento HTML ou XML, que permite a interação e manipulação dinâmica dos elementos presentes em uma página web. O **DOM** é uma estrutura hierárquica de objetos que reflete a estrutura do documento, permitindo que os programadores acessem e modifiquem elementos, atributos e conteúdo utilizando linguagens de programação, como JavaScript.

DOM - Características

- **Árvore de Elementos**: O DOM organiza os elementos do documento em uma estrutura de árvore, onde cada elemento é representado por um nó.
- **Acesso e Manipulação:** O DOM permite que você acesse, modifique e remova elementos, atributos e conteúdo de uma página web.
- **Interatividade**: Utilizando o DOM, você pode adicionar ou remover eventos (como cliques, hover etc.) aos elementos, tornando a página interativa e responsiva.

DOM - Características

- **Dinamismo**: O DOM possibilita a criação de efeitos visuais, animações e atualizações em tempo real, sem a necessidade de recarregar a página.
- **Cross-browser**: O DOM é uma API padronizada e suportada por todos os navegadores modernos, tornando o desenvolvimento consistente em diferentes plataformas.

DOM - Funcionamento

- **Parsing**: Quando um navegador carrega uma página web, ele analisa o HTML e constrói o DOM, criando uma representação em árvore dos elementos.

- **Árvore de Nós:** Cada elemento HTML é representado por um nó no DOM. Esses nós podem ser nós de elemento, nós de atributo, nós de texto etc.

DOM - Funcionamento

- **Acesso e Manipulação:** Você pode acessar e modificar elementos no DOM usando JavaScript. Isso inclui alterar o conteúdo, estilo, classes e até mesmo criar ou remover elementos.

- **Eventos**: Você pode adicionar ou remover eventos (como cliques, teclas pressionadas etc.) aos elementos para criar interatividade.

DOM - Exemplo

```html
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo DOM</title>
    <script src="script-dom.js"></script>
</head>
<body>
    <h1 id="titulo">Título</h1>
    <p id="paragrafo">Este é um parágrafo.</p>
    <button onclick="exemplodom()">Clique Aqui</button>

</body>
</html>
```

```javascript
function exemplodom(){
    var paragrafo = document.getElementById("paragrafo");
    paragrafo.textContent = "Texto modificado via DOM";
}
```

Exercício

Dúvidas??