Incorporando um arquivo JavaScript em uma página HTML

HTML

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

  <h1>Algoritmo</h1>
  <h3>Lógica de programação</h3>

  <p>
    Fazendo o link entre um arquivo <strong>JavaScript</strong> e a página
     <strong>HTML</strong>.
  </p>

  <script src="arquivoJavaScript.js"></script>
</body>
</html>
```

Exibindo informações no browser

Document Object Model (DOM)

O **DOM**, ou **Modelo de Objeto de Documento** (em inglês, ***Document Object Model***), é uma representação da estrutura de um documento HTML ou XML, que pode ser manipulada e acessada por meio de linguagens de programação, como JavaScript.

O **DOM** permite que os desenvolvedores acessem, modifiquem e interajam com os elementos de uma página web de forma dinâmica.

Aqui estão alguns pontos importantes sobre o **DOM** em programação web:

1º) Árvore de Elementos:

O **DOM** representa uma página web como uma árvore de elementos, onde cada elemento (como tags HTML) é representado por um nó (node) na árvore.

A raiz da árvore é o próprio documento (normalmente representado por **document** em JavaScript), e os nós filhos são os elementos da página, como elementos HTML (tags), atributos, texto e assim por diante.

2º) Acesso e Manipulação:

O **DOM** oferece uma interface de programação que permite aos desenvolvedores acessar e manipular elementos HTML de uma página.

Isso significa que você pode usar JavaScript para encontrar elementos, ler ou modificar seus atributos, conteúdo e estilos, bem como adicionar ou remover elementos dinamicamente em uma página web.

3º) Interatividade:

O DOM é fundamental para a interatividade em páginas da web. Ele permite que você responda a eventos do usuário, como cliques e pressionamentos de teclas, para alterar a aparência ou o conteúdo da página em tempo real.

4º) Estrutura Hierárquica:

O DOM reflete a estrutura hierárquica dos elementos em uma página da web. Isso significa que você pode navegar de um elemento pai para seus filhos e vice-versa, o que é útil ao buscar elementos específicos em uma página complexa.

Aqui está um exemplo simples de manipulação do DOM usando JavaScript para alterar o conteúdo de um elemento HTML:

```
<!DOCTYPE html>
<html>
<body>

<p id="meuParagrafo">Este é um parágrafo.</p>

<script>
        // Acessa o elemento pelo seu id
        const paragrafo = document.getElementById("meuParagrafo");

        // Modifica o conteúdo do parágrafo
        paragrafo.textContent = "Novo conteúdo do parágrafo";
</script>

</body>
</html>
```

O DOM desempenha um papel fundamental no desenvolvimento web, permitindo a criação de páginas interativas e dinâmicas. É uma parte essencial para a manipulação e renderização de conteúdo em resposta às ações dos usuários e eventos do navegador.

document.getElementById

O código document.getElementById() é uma função em JavaScript que é usada para obter uma referência a um elemento HTML com base no valor do atributo id desse elemento.

Essa função é muito comum e amplamente usada na manipulação do DOM em páginas da web.

Aqui está uma explicação mais detalhada do document.getElementById():

- document: O objeto document representa o documento HTML atual em um navegador da web. Ele é uma parte essencial do DOM e fornece uma interface para interagir com todos os elementos na página.
- getElementById(): Este é um método do objeto document. Ele é usado para buscar um elemento específico com base em seu id.
- id: O atributo id é um identificador exclusivo atribuído a um elemento HTML. Cada elemento deve ter um id exclusivo dentro de uma página da web.

Quando você chama document.getElementById() e fornece o id do elemento que deseja selecionar como argumento, a função retorna uma referência para esse elemento. Isso permite que você acesse e manipule o elemento de várias maneiras, como alterar seu conteúdo, estilo, eventos ou até mesmo remover o elemento.

Aqui está um exemplo de como usar document.getElementById():

Suponha que você tenha o seguinte elemento HTML em sua página:

```
<div id="minhaDiv">Este é um elemento de teste.</div>
```

Você pode acessar esse elemento com JavaScript da seguinte maneira:

```
const elemento = document.getElementById('minhaDiv');
```

Agora, a variável elemento contém uma referência ao com o id "minhaDiv". Você pode, então, usar essa referência para realizar diversas operações no elemento, como alterar seu conteúdo, estilo ou manipular eventos associados a ele.

Por exemplo:

```
elemento.textContent = 'Texto modificado'; // Altera o conteúdo do elemento
elemento.style.backgroundColor = 'blue'; // Altera o estilo do elemento
elemento.addEventListener('click', function() {
  alert('Clicou na div!');
}); // Adiciona um evento de clique ao elemento
```

Essa é uma maneira fundamental de interagir com elementos HTML em uma página da web usando JavaScript e é amplamente utilizada no desenvolvimento web.

innerHTML

O innerHTML é uma propriedade em JavaScript que permite acessar e modificar o conteúdo HTML de um elemento do DOM.

Essa propriedade fornece uma maneira conveniente de manipular o conteúdo interno de um elemento HTML, incluindo texto e elementos filhos. Aqui está uma explicação sobre o innerHTML:

Acesso ao Conteúdo HTML:

A propriedade innerHTML permite que você acesse o conteúdo HTML de um elemento, incluindo seu texto e outros elementos HTML que estão dentro dele.

Leitura do Conteúdo:

Você pode ler o conteúdo atual de um elemento usando innerHTML. Por exemplo:

```
const elemento = document.getElementById('meuElemento');
const conteudoHTML = elemento.innerHTML;
console.log(conteudoHTML);
```

Isso retornará o conteúdo HTML do elemento com o ID "meuElemento."

Modificação do Conteúdo:

Você também pode usar innerHTML para modificar o conteúdo de um elemento, substituindo-o por uma nova string HTML. Isso pode ser útil para adicionar, atualizar ou remover elementos dentro de um elemento pai.

Exemplo de modificação do conteúdo:

```
const elemento = document.getElementById('meuElemento');
elemento.innerHTML = '<p>Novo conteúdo HTML</p>';
```

Isso substituirá todo o conteúdo dentro do elemento com o novo conteúdo HTML.

Cuidados de Segurança:

É importante ter cuidado ao usar innerHTML, pois ele pode abrir brechas de segurança se o conteúdo inserido não for confiável. Isso pode resultar em ataques de injeção de código. É recomendável usar outras técnicas, como textContent, quando você precisa manipular conteúdo de texto simples e confiável.

Aninhamento de Elementos:

O innerHTML permite o aninhamento de elementos HTML. Isso significa que você pode inserir elementos HTML dentro de outros elementos usando essa propriedade.

Exemplo de aninhamento de elementos:

```
const elemento = document.getElementById('meuElemento');
elemento.innerHTML = '<p>Parágrafo <span>aninhado</span></p>';
```

Nesse caso, um parágrafo (<p>) contém um elemento span aninhado.

O uso de innerHTML pode ser muito conveniente para tarefas específicas de manipulação de conteúdo HTML, mas lembre-se de usar com responsabilidade e cuidado, especialmente quando a entrada do usuário ou dados externos não confiáveis estiverem envolvidos, para evitar vulnerabilidades de segurança. Em muitos casos, dependendo da tarefa, pode ser mais seguro usar métodos como textContent para manipular apenas o texto do elemento.

Entrada de dados

Prompt

O comando prompt em JavaScript é uma função que permite que você exiba uma caixa de diálogo de entrada ao usuário em um navegador da web. Essa caixa de diálogo geralmente é usada para coletar dados inseridos pelo usuário, como texto, números ou informações diversas.

A sintaxe básica da função prompt é a seguinte:

```
let resultado = prompt("Texto da mensagem para o usuário", "Valor Padrão");
```

Aqui estão os componentes principais:

Texto da mensagem para o usuário: Esta é a mensagem ou pergunta que será exibida para o usuário na caixa de diálogo. É uma string que fornece contexto sobre o que o usuário deve inserir.

Valor Padrão (opcional): Este é um valor padrão que pode ser pré-preenchido na caixa de entrada. O usuário pode aceitar esse valor padrão ou substituí-lo.

O prompt retorna o valor inserido pelo usuário como uma string. Se o usuário pressionar "Cancelar" na caixa de diálogo, o resultado será null.

Exemplo de uso do prompt:

```
let nome = prompt("Digite seu nome:");
if (nome !== null) {
  console.log("Olá, " + nome + "!");
} else {
  console.log("Você cancelou a entrada.");
}
```

Neste exemplo, uma caixa de diálogo será exibida solicitando que o usuário insira seu nome. Se o usuário inserir um nome e clicar em "OK", o nome será armazenado na variável nome e saudado na próxima linha. Se o usuário clicar em "Cancelar", o prompt retorna null, e uma mensagem de cancelamento será exibida.

Tenha em mente que, como o **prompt** retorna uma string, você pode precisar fazer conversões de tipo (casting) se desejar que o valor inserido seja tratado como um número, por exemplo. Certifique-se também de validar os dados inseridos pelo usuário para garantir que sejam adequados ao que você espera em seu código.

Operador de atribuição " = "

O operador de atribuição = (um único sinal de igual) é um elemento fundamental para as linguagens de programação, incluindo JavaScript.

Sua função é atribuir um valor a uma variável.

Aqui estão os principais pontos a serem considerados sobre o operador de atribuição:

1º) Atribuição de Valor:

O operador = é usado para atribuir um valor a uma variável. O valor à direita do operador é atribuído à variável à esquerda.

Exemplo:

```
let x = 10; // Atribui o valor 10 à variável x
```

2º) Atualização de Variáveis:

O operador de atribuição é comumente usado para atualizar o valor de uma variável. Você pode atribuir um novo valor à variável, substituindo o valor existente.

Exemplo:

```
let y = 5;
y = y + 3; // Atualiza o valor de y para 8
```

Isso pode ser simplificado usando a notação de atribuição composta:

```
let y = 5;
y += 3; // Também atualiza o valor de y para 8
```

3º) Expressões à Direita:

A expressão à direita do operador de atribuição é resolvida e seu resultado é atribuído à variável à esquerda.

Exemplo:

```
let a = 5 + 3; // efetua a soma (5 + 3) e atribui o resultado (8) à variável 'a'
```

4º) Atribuição em Cadeia:

Você pode encadear múltiplas atribuições em uma única linha, atribuindo valores a várias variáveis.

Exemplo:

```
let p = 10, q = 20, r = 30; // Atribui 10 a p, 20 a q e 30 a r em uma única linha
```

5º) Operador de Atribuição em Destructuring:

O operador de atribuição também é usado em recursos de JavaScript, como atribuição de desestruturação, para extrair valores de objetos ou arrays e atribuí-los a variáveis.

Exemplo:

```
const person = { name: 'Alice', age: 30 };
const { name, age } = person; // Atribui 'Alice' a name e 30 a age
```

Em resumo, o operador de atribuição = é essencial para a criação, atualização e manipulação de variáveis em JavaScript e em muitas outras linguagens de programação. Ele permite que você associe valores a variáveis, o que é uma parte fundamental da programação.

Incremento e decremento

Em JavaScript, existem operadores de pré incremento (++variável), pós-incremento (variável++) e operadores de pré-decremento (--variável) e pós-decremento (variável--).

Esses operadores são usados para aumentar ou diminuir o valor de uma variável numérica em 1. No entanto, eles diferem na ordem em que o valor é modificado e quando o valor original é usado em uma expressão.

Vamos explicar cada um deles com exemplos:

Pré-incremento (++variável):

O operador pré-incremento aumenta o valor da variável em 1 e, em seguida, usa o novo valor.

Isso significa que primeiro a variável é incrementada e, em seguida, o valor atualizado é retornado podendo ser utilizado.

Exemplo:

```
let x = 5;
let y = ++x;
console.log(x); // x agora é 6
console.log(y); // y é 6, pois foi incrementado antes de ser atribuído a y
```

Pós-incremento (variável++):

O operador pós-incremento usa o valor atual da variável em uma expressão e, em seguida, aumenta o valor em 1.

Isso significa que primeiro o valor atual é retornado (podendo ser utilizado) e, em seguida, a variável é incrementada.

Exemplo:

```
let x = 5;
let y = x++;
console.log(x); // x agora é 6
console.log(y); // y é 5, pois o valor original de x é usado antes do incremento
```

Pré-decremento (--variável):

O operador pré-decremento diminui o valor da variável em 1 e, em seguida, usa o novo valor.

Isso significa que primeiro a variável é decrementada e, em seguida, o valor atualizado é retornado (podendo ser utilizado).

Exemplo:

```
let x = 5;
let y = --x;
console.log(x); // x agora é 4
console.log(y); // y é 4, pois foi decrementado antes de ser atribuído a y
```

Pós-decremento (variável--):

O operador pós-decremento usa o valor atual da variável em uma expressão e, em seguida, diminui o valor em 1.

Isso significa que primeiro o valor atual é retornado (podendo ser utilizado) e, em seguida, a variável é decrementada.

Exemplo:

```
let x = 5;
let y = x--;
console.log(x); // x agora é 4
console.log(y); // y é 5, pois o valor original de x é usado antes do decremento
```

Lembre-se de que esses operadores podem ser usados para incrementar ou decrementar variáveis numéricas e podem ser úteis em loops, cálculos matemáticos e outras situações em que você precise alterar o valor de uma variável de forma controlada.

Mas por que utilizar o ++ e não o += ou 'n = n+1'

Utilizar o operador **++** em vez de **+= 1** ou **n = n + 1** é uma questão de conveniência e legibilidade do código.

Todos esses métodos podem ser usados para incrementar o valor de uma variável em 1, mas eles têm suas vantagens e desvantagens:

1º) ++ é mais conciso:

O operador ++ é uma forma mais concisa de incrementar uma variável em 1. Isso torna o código mais limpo e mais fácil de ler, especialmente quando você está realizando incrementos simples.

Exemplo:

```
let x = 5;
x++; // Mais conciso
```

Em vez de:

```
let x = 5;
x += 1; // Menos conciso
// ou
x = x + 1; // Menos conciso
```

2º) ++ é uma operação única:

O operador ++ é uma única operação, enquanto += 1 ou n = n + 1 envolvem duas operações (uma adição e uma atribuição). Em muitos casos, uma única operação pode ser mais eficiente.

3º) ++ é amplamente reconhecido:

O uso de ++ é uma convenção amplamente reconhecida em muitas linguagens de programação, o que torna o código mais legível para outros desenvolvedores.

A maioria dos programadores está familiarizada com esse operador e entende imediatamente seu propósito.

No entanto, a escolha entre essas opções depende do contexto e das preferências de codificação da equipe.

O uso do operador ++ é recomendado quando você deseja realizar um incremento simples de 1. Para incrementos maiores do que 1 ou para situações mais complexas, como incrementar por uma variável diferente de 1, o uso de += ou n = n + 1 pode ser mais apropriado.

Exercícios:

Exercício 33:

Crie uma aplicação que receba um número e imprima sua raiz quadrada.
SAÍDA:

A raiz quadrada de no número .... é ...

Exercício 34:

Crie uma aplicação que receba um número e imprima seu valor elevado a 2, elevado a 3, elevado a 4 e elevado a 5.
SAÍDA:

O número digitado foi ... E seu valor elevado a 2 é ..., elevado a 3 é ..., ...

Exercício 35:

Crie uma aplicação que receba um número e imprima sua raiz quadrada e sua raiz cúbica.
SAÍDA:

O número digitado foi ...
Sua raiz cúbica é ...
Sua raiz quadrada é ...

Exercício 36:

Crie uma aplicação que receba com quatro números e imprimir a média ponderada, sabendo-se que os pesos são respectivamente 1, 2, 3 e 4.

Exercício 37:

Crie uma aplicação que receba o raio de uma circunferência e calcule a área e o perímetro do círculo correspondente.

A fórmula para se calcular a área da circunferência é : $\mathrm{A} = \pi * \mathrm{raio}^2$

A fórmula para se calcular o perímetro da circunferência é : $P = 2 * \pi * \mathrm{r}$

Exercício 38:

Crie uma aplicação que receba os lados A, B e C de um paralelepípedo.

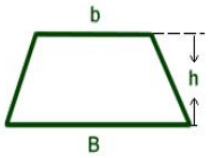
Calcular e imprimir o volume.

- Volume = A * B * C

Exercício 39:

Crie uma aplicação para calcular a área de um trapézio qualquer (figura meramente ilustrativa).

Área = ((b + B) * h) / 2

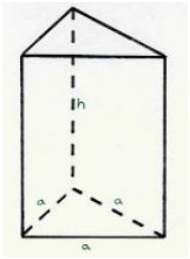


Exercício 40:

Crie uma aplicação que possa calcular o volume de um prisma de base triangular (figura meramente ilustrativa).

Para calcular a área de um triângulo utilize a fórmula "(base * altura)/2".

Volume = área da base x altura.

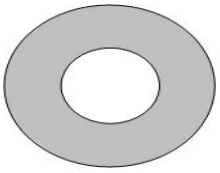


Exercício 41:

Crie uma aplicação que possa calcular a área de uma coroa de forma circular (figura meramente ilustrativa).

Área da circunferência = PI * raio^2

Área = (Área da circunferência Maior) – (Área da circunferência menor)



Exercício 42:

Crie uma aplicação que possa calcular o volume de um cilindro (figura meramente ilustrativa).

Área da base = área da circunferência

Volume = área da base x altura

Exercício 43:

Crie uma aplicação para calcular o volume e a área de uma esfera (figura meramente ilustrativa).

Área = 4 * PI * r^2

Volume = (4 * pi * r^3) / 3