Curso de JavaScript

Instituto Federal de São Paulo (IFSP)

Professor: Heder Dorneles Soares

31 de outubro de 2023

Introdução à História do JavaScript

- JavaScript: Uma linguagem de programação essencial para a web.
- Criada por Brendan Eich em 1995.
- Inicialmente chamada de LiveScript, renomeada posteriormente para JavaScript.

Tipos de Valores em JavaScript

- Números: Inteiros e de ponto flutuante.
- Strings: Sequências de caracteres.
- Booleanos: True (verdadeiro) e False (falso).
- Undefined e Null: Representando valores ausentes.

Operadores em JavaScript

- Operadores Aritméticos: $+, -, *, /, \%$, etc.
- Operadores de Comparação: $==, ===, <, >, <=, >=$, etc.
- Operadores Lógicos: $\&\&, ||, !$.
- Operadores de Atribuição: $=$, $+=$, -=, *=, $/=$, etc.

A Tag <script>

Em HTML, o código JavaScript é inserido entre as tags <script> e </script>. Exemplo:

```
<script>
document.getElementById("demo").innerHTML =
"Meu Primeiro JavaScript";
</script>
```

Neste exemplo, o código JavaScript é colocado entre as tags <script> e </script>, permitindo a execução do código no navegador.

Scripts no Elemento <head>

Quando você coloca seus scripts no elemento <head>, eles são carregados antes do conteúdo da página. Isso pode ser útil para scripts que precisam ser carregados antes da renderização da página, como folhas de estilo ou bibliotecas JavaScript.

Scripts no Elemento <body>

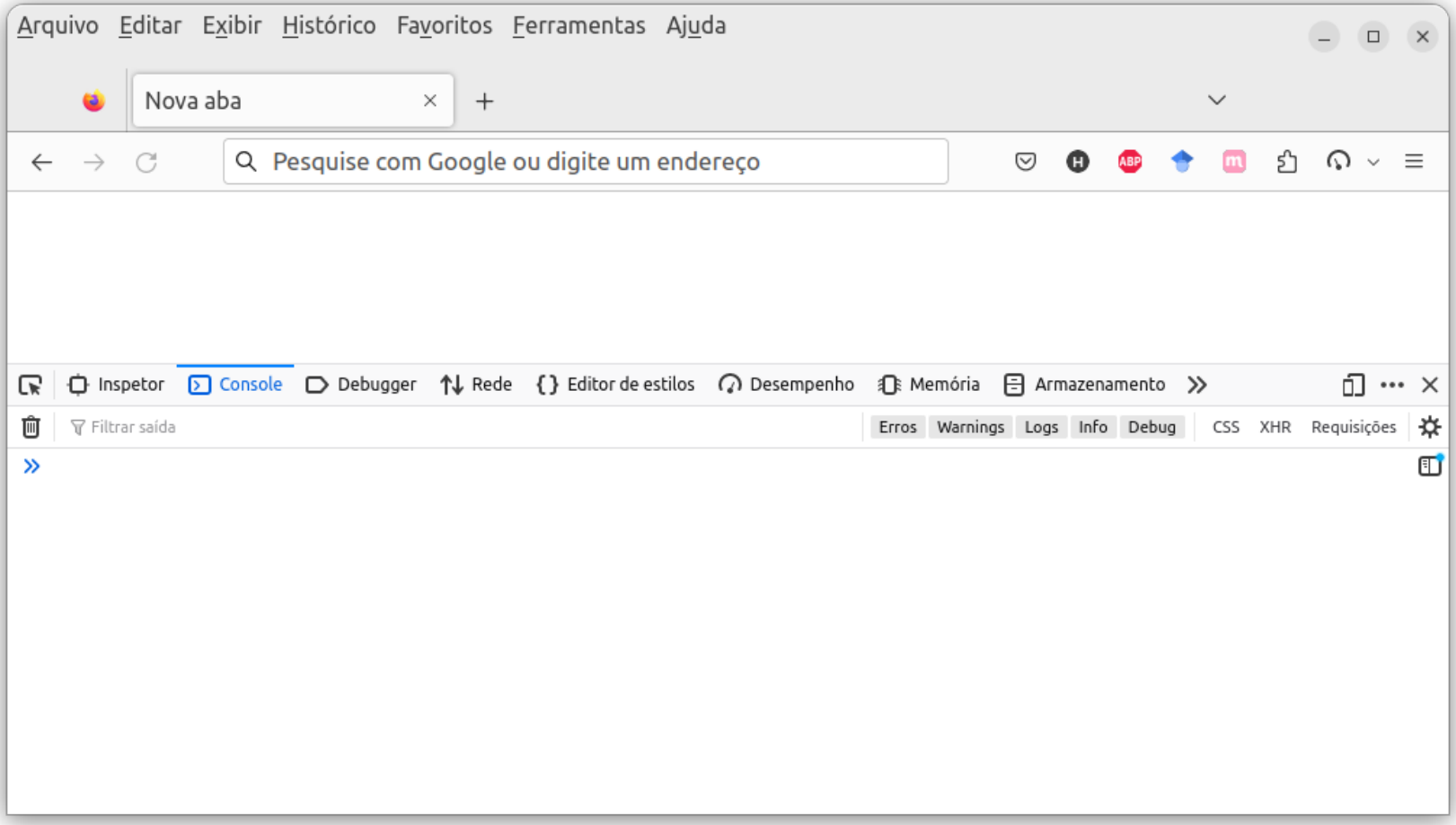
Quando você coloca seus scripts no elemento <head>, eles são carregados antes do conteúdo da página. Isso pode ser útil para scripts que precisam ser carregados antes da renderização da página, como folhas de estilo ou bibliotecas JavaScript.
Colocar seus scripts no final do elemento <body> faz com que sejam carregados após o conteúdo da página ter sido renderizado. Isso é útil para scripts que não precisam ser carregados imediatamente, melhorando a velocidade de carregamento da página.

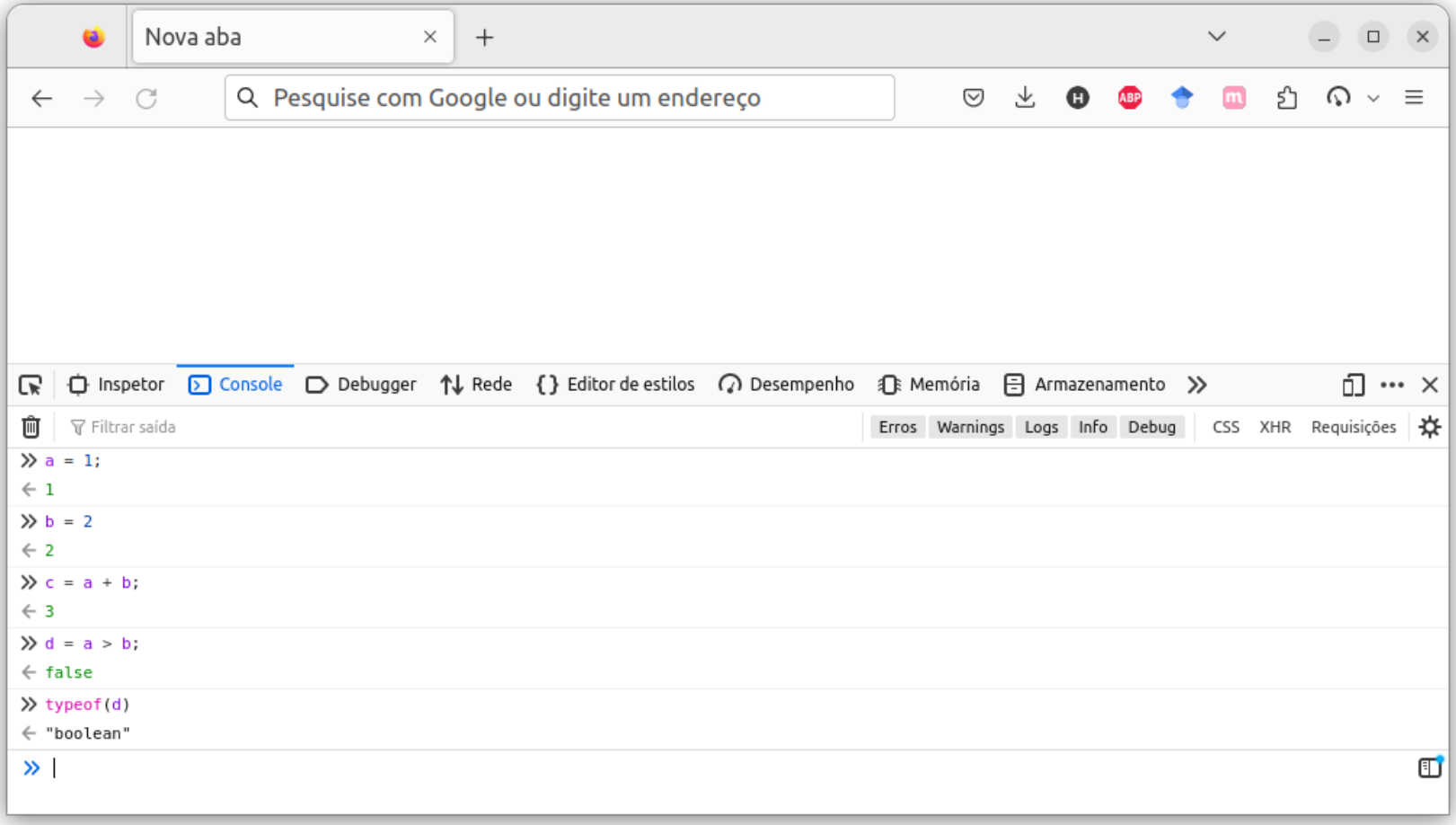
Sintaxe para Incluir um Script Externo

Use a seguinte sintaxe para incluir um script de um arquivo externo:

```
<script src="caminho-para-o-seu-arquivo.js"></script>
```

Substitua "caminho-para-o-seu-arquivo.js" pelo caminho relativo ou absoluto para o arquivo JavaScript externo.


Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
Nova aba
Pesquise com Google ou digite um endereço
Inspetor
Console
Debugger
Rede
Editor de estilos
Desempenho
Memória
Armazenamento
Filtrar saída
Erros
Warnings
Logs
Info
Debug
CSS
XHR
Requisições


Nova aba
Pesquise com Google ou digite um endereço
Inspetor
Console
Debugger
Rede
Editor de estilos
Desempenho
Memória
Armazenamento
Filtrar saída
Erros
Warnings
Logs
Info
Debug
CSS
XHR
Requisições
a = 1;
1
b = 2
2
c = a + b;
3
d = a > b;
false
typeof(d)
"boolean"

Variáveis em JavaScript

Variáveis são elementos fundamentais em JavaScript para armazenar e manipular dados. Elas são usadas para guardar informações que podem ser utilizadas ao longo do programa.

Declaração de Variáveis

Em JavaScript, você pode declarar variáveis usando as palavras-chave `var`, `let`, ou `const`. A escolha da palavra-chave afeta o escopo da variável.

Exemplos de Declaração de Variáveis

- `var` - Escopo de função.
- `let` - Escopo de bloco.
- `const` - Escopo de bloco e valor constante.

Inicialização de Variáveis

Você pode inicializar uma variável no momento da declaração. Exemplo:

```
var nome = "João";
let idade = 25;
const pi = 3.14;
```

Atribuição de Valores

Em JavaScript, você pode atribuir novos valores a variáveis já declaradas. Exemplo:

```
nome = "Maria";
idade = 30;
// pi = 3.14159; (Não é permitido com const)
```

Tipos de Variáveis

Variáveis em JavaScript podem conter diferentes tipos de dados, como números, strings, booleanos, objetos, funções e muito mais. O tipo de dado é determinado dinamicamente.

Strings em JavaScript

As strings são usadas para representar texto em JavaScript. Elas são coleções de caracteres, como letras, números, símbolos e espaços.
Exemplos de strings:

- "Olá, Mundo!"
- 'JavaScript é incrível!'
- "12345"(uma sequência de caracteres, não um número)

As strings podem ser delimitadas por aspas simples ('), aspas duplas () ou crases (‘).
Exemplo:

```
let saudacao = "Olá, Mundo!";
let nome = 'Alice';
let numero = "12345";
```

Operações Aritméticas em JavaScript

JavaScript oferece uma variedade de operadores aritméticos para realizar cálculos matemáticos em números. Alguns dos operadores mais comuns incluem:

- Adição (+) para somar números.
- Subtração (-) para subtrair números.
- Multiplicação (*) para multiplicar números.
- Divisão (/) para dividir números.
- Módulo (%) para obter o resto da divisão.

Exemplo de uso:

```
let resultado = 10 + 5; // resultado contém 15
let diferenca = 20 - 8; // diferença contém 12
let produto = 6 * 4;   // produto contém 24
let quociente = 15 / 3; // quociente contém 5
let resto = 10 % 3;     // resto contém 1
```

Lendo Valores em JavaScript

Para ler valores em JavaScript, você pode usar a função ‘prompt()‘ para solicitar entrada do usuário. A função ‘prompt()‘ exibe uma caixa de diálogo na qual o usuário pode inserir um valor. Veja um exemplo:

```
let nome = prompt("Digite seu nome:");
let idade = prompt("Digite sua idade:");
```

Neste exemplo, o usuário insere seu nome e idade, que são armazenados nas variáveis ‘nome‘ e ‘idade‘, respectivamente. Lembre-se de que os valores lidos com ‘prompt()‘ são tratados como strings. Você pode precisar convertê-los para outros tipos, se necessário.

Exibindo Valores em JavaScript

Para exibir valores em JavaScript, você pode usar várias abordagens, incluindo:

- `console.log()`: Usado para exibir valores no console do navegador, útil para depuração e desenvolvimento.
- `document.write()`: Utilizado para inserir conteúdo diretamente no documento HTML, sobrescrevendo o conteúdo existente.
- `alert()`: Cria uma caixa de diálogo pop-up exibindo uma mensagem ao usuário.

Exemplo de uso:

```
let mensagem = "Olá, mundo!";

console.log(mensagem); // Exibe no console
document.write(mensagem); // Insere no documento HTML
alert(mensagem); // Exibe uma janela de alerta
```

Exercício: Calculadora Simples

Crie um programa em JavaScript que funcione como uma calculadora simples. Siga as instruções abaixo:

1. Solicite ao usuário que insira dois números.
2. Realize a soma dos valores.
3. Exiba o resultado na página web.

Exemplo de saída na página:

```
Primeiro número: [número]
Segundo número: [campo]
Resultado: [resultado]
```

Utilize ‘prompt()‘, ‘alert()‘, ‘document.write()‘, ou qualquer outro método para interagir com o usuário e exibir o resultado.

Exemplo de Estrutura "if"

- Vamos ver um exemplo simples:
- Suponha que queremos exibir uma mensagem se uma variável `idade` for maior ou igual a 18.

Exemplo

```
var idade = 20;
if (idade >= 18) {
  document.write("Você é maior de idade.");
}
```

Operadores de Comparação em JavaScript

- Vamos analisar algumas instruções que usam operadores de comparação em JavaScript.
- Primeiro, vamos corrigir os erros nas instruções fornecidas:

```
var num1 = 10;
var num2 = 10;
document.write(num1 == num2); // imprime true
document.write(num1 != num2); // imprime false
```

- Agora, ambas as instruções retornam valores booleanos corretamente.
- Além dos operadores ‘<‘, ‘<=‘, ‘>‘, ‘>=‘, podemos usar também ‘==‘ para realizar comparações numéricas.
- Vamos responder à pergunta: Qual o retorno da instrução abaixo?

```
document.write(1 == "1"); // O que isso imprimirá?
```

Conversões em Comparações em JavaScript

- No JavaScript, ocorrem conversões automáticas em comparações.
- Quando uma comparação envolve uma string e um número, o JavaScript converte a string em um número.
- O mesmo ocorre quando um número é comparado com um valor booleano, ou seja, o booleano é convertido em um número.
- Isso também se aplica quando testamos se um número é maior que o outro:

```
document.write(2 > "1"); // retorna true
```

- No exemplo acima, a string "1"é automaticamente convertida em um número para que a comparação seja realizada.

Conversões em Comparações em JavaScript

- No JavaScript, ocorrem conversões automáticas em comparações.
- Quando uma comparação envolve uma string e um número, o JavaScript converte a string em um número.
- O mesmo ocorre quando um número é comparado com um valor booleano, ou seja, o booleano é convertido em um número.
- Isso também se aplica quando testamos se um número é maior que o outro:

Exemplo

```
document.write(2 > "1"); // retorna true
```

- No exemplo acima, a string "1"é automaticamente convertida em um número para que a comparação seja realizada.

Blocos Condicionais em JavaScript

Blocos condicionais são comuns em JavaScript, o bloco mais utilizado é o "if", que recebe um valor booleano como parâmetro:

```
var idade = 18;
var temCarteira = true;

if (idade >= 18) {
  document.write("Pode dirigir");
} else {
  document.write("Proibido dirigir");
}
```

- No exemplo acima, o bloco "if" verifica se a variável "idade" é maior ou igual a 18. Se a condição for verdadeira, ele exibe "Pode dirigir", caso contrário, exibe "Proibido dirigir".

Blocos Condicionais em JavaScript

Também é possível usar os operadores && (E) e || (OU) em blocos condicionais em JavaScript:

```
var idade = 18;
var temCarteira = false;

if (idade >= 18 || temCarteira) {
  document.write("Pode dirigir");
} else {
  document.write("Proibido dirigir");
}
```

Blocos Condicionais em JavaScript

Também é possível usar os operadores && (E) e || (OU) em blocos condicionais em JavaScript:

```
var idade = 18;
var temCarteira = false;

if (idade >= 18 || temCarteira) {
  document.write("Pode dirigir");
} else {
  document.write("Proibido dirigir");
}
```

O código acima possui um erro lógico: permite que um maior de idade sem carteira possa dirigir!

Blocos Condicionais em JavaScript

Correção

```
if (idade >= 18 && temCarteira) {
  document.write("Pode dirigir");
} else {
  document.write("Proibido dirigir");
}
```

Estruturas de Repetição em JavaScript

- As estruturas de repetição são usadas para executar um bloco de código repetidamente.
- Um exemplo comum é o loop "for".
- Aqui está um exemplo de como usar um loop "for" em JavaScript:

Exemplo de Loop "for"

```
for (var i = 0; i < 5; i++) {
  document.write("Número: " + i + "<br>");
}
```

Trabalhando com Vetores em JavaScript

- Vetores (ou arrays) são estruturas de dados que permitem armazenar vários valores em uma única variável.
- Você pode acessar os elementos de um vetor por meio de índices.
- Aqui está um exemplo de como criar e acessar um vetor em JavaScript:

Exemplo de Uso de Vetor

```
var frutas = ["Maçã", "Banana", "Laranja", "Pera"];
for (var i = 0; i < frutas.length; i++) {
  document.write("Índice "+ i + ": " + frutas[i] +"<br>");
}
```

Funções em JavaScript

- Muitas vezes é necessário definir um comportamento que será executado posteriormente em JavaScript.
- Para isso, existem as funções.

Estrutura de uma Função

```
function nomeDaFuncao(parametro) {
  // Corpo da função
}
```

Uma função criada dessa forma é chamada de "function declaration".

Funções em JavaScript - Exemplo

Vejamos um exemplo prático de função em JavaScript:

Exemplo de Função em JavaScript

```
function exibeMensagem() {
  alert("Attenzione pickpocket");
}
exibeMensagem(); // Chamando a função declarada
exibeMensagem(); // Chamando a função novamente
```

- No exemplo acima, a função ‘exibeMensagem‘ exibe um alerta com a mensagem ”Atenção”.
- É possível chamar a função quantas vezes forem necessárias.
- **Observação**: Não é necessário inserir o ponto-e-vírgula no final da declaração da função, mas pode ser colocado.

Exercício Calculadora

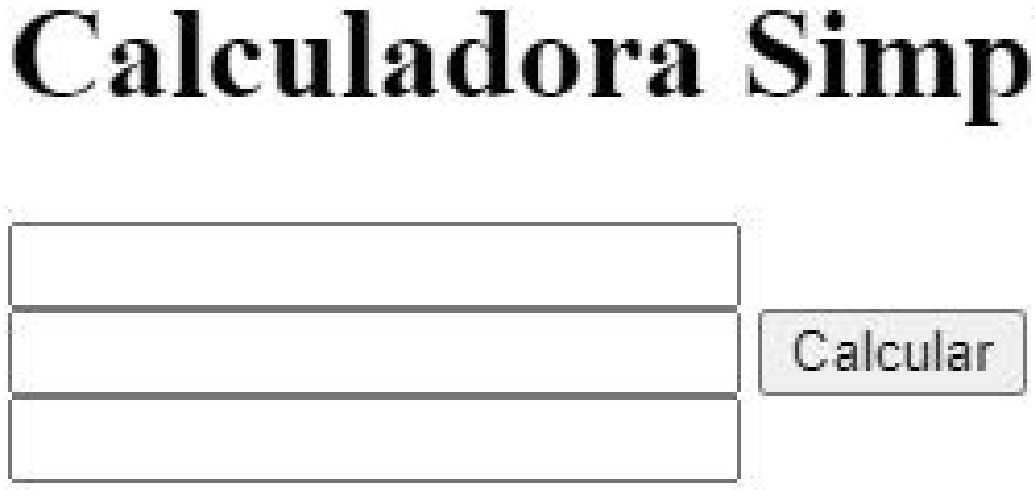


Figura: Calculadora HTML

Calculadora Simples em HTML

Código HTML da Calculadora

```
<h1>Calculadora Simples</h1>

<input type="text" id="numero1">
<input type="text" id="numero2">
<button onclick="calcular()">Calcular</button>
<input type="text" id="resultado" readonly>
```

Calculadora Simples em HTML

Código HTML da Calculadora

```
<script>
function calcular() {
  var num1 = parseFloat(document.
                  getElementById("numero1").value);
  var num2 = parseFloat(document.
                  getElementById("numero2").value);
  var resultado = num1 + num2;
  document.getElementById("resultado").value = resultado;
}
</script>
```