Lógica de Programação com Java Script

Othon Oliveira

SENAC - PROA

Sumário

Introdução a Lógica de Programação com Java Script

Locais quase obrigatírios do Java Script/CSS

Com organizar seu código

a. O Java Script deve ficar num arquivo separado do HTML

Introdução a Lógica de Programação com Java Script

Locais quase obrigatírios do Java Script/CSS

Com organizar seu código

a. O Java Script deve ficar num arquivo separado do HTML

b. Normalmente um "seu_arquivo.js" fica numa pasta separada

Introdução a Lógica de Programação com Java Script

Locais quase obrigatírios do Java Script/CSS

Com organizar seu código

a. O Java Script deve ficar num arquivo separado do HTML
b. Normalmente um "seu_arquivo.js" fica numa pasta separada
c. O arquivo CSS também segue a mesma lógica anterior

Introdução a Lógica de Programação com Java Script

Locais quase obrigatírios do Java Script/CSS

Com organizar seu código

a. O Java Script deve ficar num arquivo separado do HTML
b. Normalmente um "seu_arquivo.js" fica numa pasta separada
c. O arquivo CSS também segue a mesma lógica anterior
d. Dessa forma, seu código ficará organizado e fácil de ser encontrado

Exemplo de arquivo.css

Nome do aquivo: style.css

```
/* style.css */

body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
}

h1 {  color: blue; }

p {  font-size: 16px;
  margin-bottom: 10px;
}
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Exemplo de CSS com arquivo separado</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <h1>Título da Página</h1>
  <p>Este é um parágrafo com estilo definido.</p>
</body>
</html>
```

Vincular é o mesmo que "conectar, ligar,.."

Apresentando o seu projeto

Seu projeto ficará, mais ou menos assim:

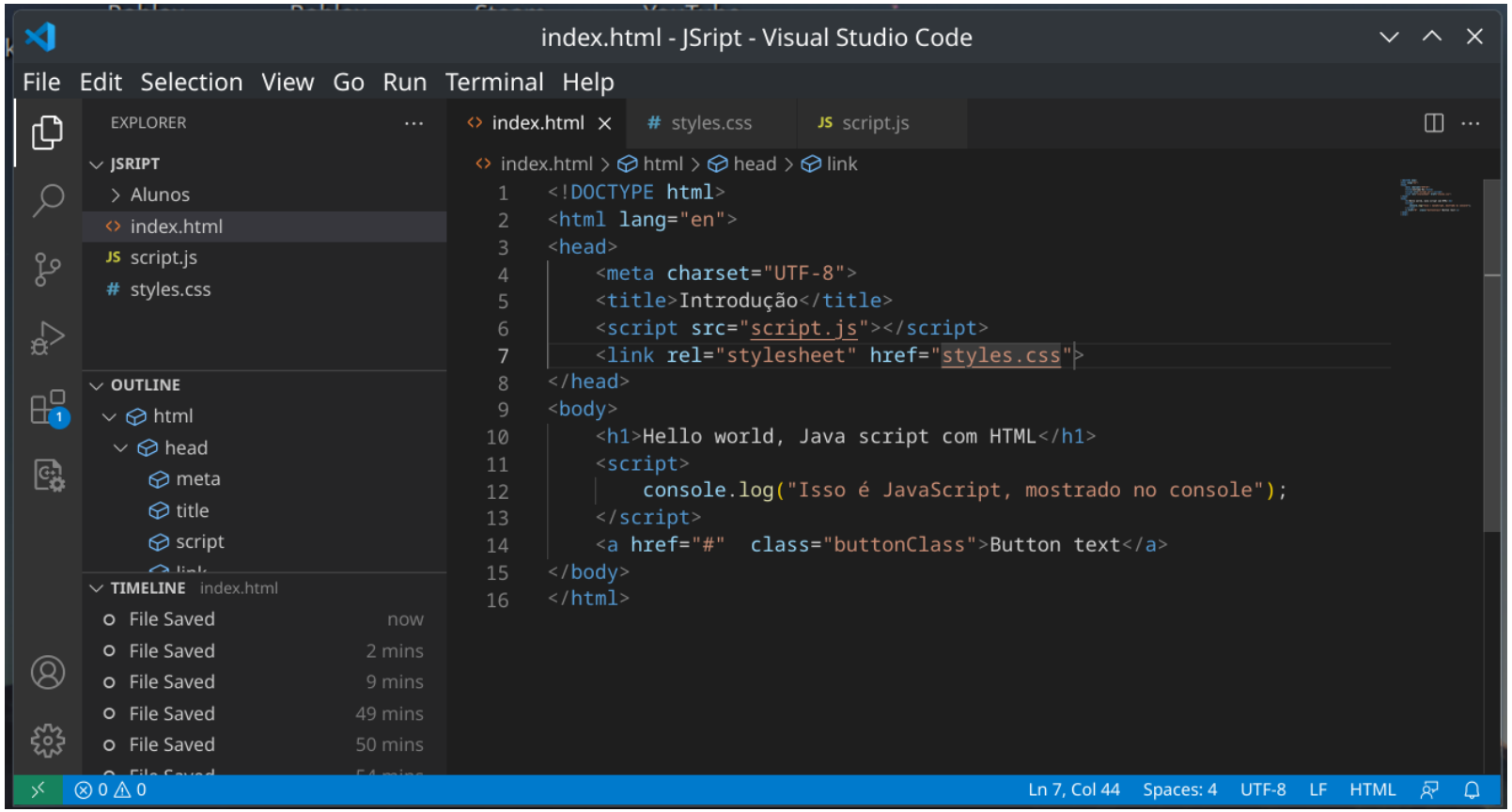

Figura: Estrutura do projeto

A página “index.html” geralemente é mandatória pelos servidores Web

Estruturas de Repetição em JavaScript

Estrutura "for"

```
for (inicialização; condição; atualização) {
  // Código a ser executado
}
```

- Inicialização: Define o valor inicial da variável de controle.
- Condição: Verifica se a repetição deve continuar.
- Atualização: Atualiza a variável de controle após cada iteração.

Estruturas de Repetição em JavaScript

Estrutura "while"

```
while (condição) {
  // Código a ser executado
}
```

- A repetição ocorre enquanto a condição for verdadeira.
- A condição é verificada antes de cada iteração.
- Certifique-se de que a condição seja eventualmente falsa para evitar loops infinitos.

Estruturas de Repetição em JavaScript

Estrutura "do-while"

```
do {
  // Código a ser executado
} while (condição);
```

- A repetição ocorre pelo menos uma vez, pois a condição é verificada após a execução do bloco.
- O código é executado primeiro e a condição é verificada depois.
- A condição determina se o loop continua ou não.

Estruturas de Repetição especiais em JavaScript

Loop For...in

```
const person = { name: "John", age: 30 };
for (let key in person) {
  console.log(key + ": " + person[key]);
}
```

- O loop "for...in" itera sobre as propriedades de um objeto.
- Em cada iteração, a variável "key" contém o nome da propriedade.
- Pode ser útil para percorrer propriedades de objetos ou elementos de um array associativo.

Estruturas de Repetição especiais em JavaScript

Loop For...of

```
const numeros = [1, 2, 3, 4, 5];
for (let num of numeros) {
  console.log(num);
}
```

- O loop "for...of" itera sobre os valores de um objeto iterável, como um array.
- Em cada iteração, a variável "num" contém o valor do elemento atual.
- É uma forma mais simples de percorrer os elementos de um array.

Declaração de Variáveis em JavaScript

let vs. var vs. const

- **let**: Introduz uma variável de escopo de bloco.
- **var**: Introduz uma variável de escopo de função ou global.
- **const**: Introduz uma constante de escopo de bloco.

```
// Exemplos:
let x = 10;          // Escopo de bloco
var y = 20;          // Escopo de função/global
const z = 30;        // Constante de bloco
```

- Use ‘let‘ para criar variáveis com escopo mais restrito.
- Use ‘var‘ com cautela, devido ao seu escopo menos restrito.
- Use ‘const‘ para criar constantes imutáveis.

Declaração de Variáveis em JavaScript

Problemas do uso de let e var

Problemas com "var":

- Escopo global ou de função pode causar bugs.
- Hoisting pode levar a comportamentos inesperados.

Problemas com "let":

- Escopo de bloco mais restrito ajuda a evitar bugs.
- Variáveis declaradas com "let" não são hoisted.
- Uso em loops pode causar problemas de closure.

Recomendação:

- Prefira usar "let" para evitar bugs de escopo e hoisting.
- Evite o uso de "var" para declarações de variáveis.

Hoisting em JavaScript

Hoisting

- Hoisting é o comportamento em que declarações de variáveis e funções são movidas para o topo de seu escopo antes da execução do código.
- Isso significa que você pode usar variáveis antes de declará-las.

Exemplo de Hoisting com "var":

```
console.log(x); // variável indefinida ainda
var x = 10;     // variável definia e atribuída
```

Exemplo de Hoisting com "let"e "const":

```
console.log(y); // ReferenceError (erro de referência
let y = 20;
```

Dica:

- Evite hoisting não intencional usando "let"e "const".

Estruturas de Repetição em JavaScript

1. Laço (ou loop)- For

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

Estruturas de Repetição em JavaScript

2. Laço (ou loop)- While

```
let counter = 0;
while (counter < 5) {
  console.log(counter);
  counter++;
}
```

Estruturas de Repetição em JavaScript

3. Laço (ou loop)- D̈o While¨

```
let number = 1;
do {
  console.log(number);
  number++;
} while (number <= 5);
```

Estruturas de Repetição em JavaScript

Usando o Break para sair da repetição

```
for (let i = 0; i < 10; i++) {
  if (i === 5) {
    break;
  }
  console.log(i);
}
```

Estruturas de Repetição em JavaScript

5. O comando: Continue

```
for (let i = 0; i < 5; i++) {
  if (i === 2) {
    continue;
  }
  console.log(i);
}
```

Estruturas de Repetição em JavaScript

6. Laço (ou loop)- For...in

```
const pessoas = { nome: "John", idade: 30 };
for (let chave in pessoas) {
  console.log(key + ": " + pessoas[chave]);
}
```

Estruturas de Repetição em JavaScript

7. Laço (ou loop)- For...of

```
const numbers = [1, 2, 3, 4, 5];
for (let num of numbers) {
  console.log(num);
}
```

Estruturas de Repetição em JavaScript

8. Laços (loops) aninhados

```
for (let i = 0; i < 3; i++) {
  for (let j = 0; j < 2; j++) {
    console.log(i, j);
  }
}
```

Estruturas de Repetição em JavaScript

9. laço While com Break

```
let num = 1;
while (true) {
  console.log(num);
  num++;
  if (num > 5) {
    break;
  }
}
```

Estruturas de Repetição em JavaScript

10. laço While com Continue

```
let num = 0;
while (num < 5) {
  num++;
  if (num === 3) {
    continue;
  }
  console.log(num);
}
```

Teste de números primos - primalidade

Função para verificar se um número é primo

```
function ehPrimo(numero) {
  if (numero <= 1) { return false; }
  if (numero <= 3) { return true;  }
  if (numero % 2 === 0 || numero % 3 === 0) {
    return false;
  }
  for (let i = 5; i * i <= number; i += 6) {
    if (number % i === 0 || number % (i + 2) === 0) {
      return false;
    }
  }
  return true;
}
```