Introdução ao Javascript

Vamos relembrar alguns conceitos básicos de programação com Javascript.

O que vamos ver?

- Variáveis
- Funções
- Estruturas condicionais
- Manipulação de Listas(Arrays) e Objetos

Variáveis (let, const, var)

Em JavaScript, let, const, e var são palavras-chave usadas para declarar variáveis, mas elas têm diferenças importantes em relação ao escopo e à mutabilidade.

var

- Tem escopo de função (function-scoped) e não de bloco.
- Pode ser redeclarada dentro do mesmo escopo.
- É elevada (hoisted) ao topo do seu contexto de execução.

Exemplo

```
function exemploVar() {
  if (true) {
    var x = 10;
    console.log(x); // Output: 10
  }
  console.log(x); // Output: 10
}
```

let

- Introduzido no ECMAScript 6 (ES6).
- Tem escopo de bloco (block-scoped).
- Não pode ser redeclarada no mesmo escopo.
- Não é elevada (hoisted) ao topo do seu contexto de execução.

Exemplo

```
function exemploLet() {
  if (true) {
    let y = 20;
    console.log(y); // Output: 20
  }
  // console.log(y); // Error: y is not defined (não pode ser acessada fora do bloco)
}
```

const

- Introduzido no ECMAScript 6 (ES6).
- Tem escopo de bloco (block-scoped).
- Não pode ser reatribuída após a atribuição inicial.
- Não é elevada (hoisted) ao topo do seu contexto de execução.

Exemplo

```
function exemploConst() {
  const z = 30;
  // z = 40; // Error: Assignment to constant variable (não pode ser reatribuída)
  console.log(z); // Output: 30
}
```

Dicas:

- Use `const` sempre que possível, pois isso ajuda a evitar acidentalmente reatribuir variáveis.
- Use `let` quando precisar reatribuir variáveis.
- Evite o uso de `var` em código moderno, a menos que haja uma razão específica para usá-lo em versões mais antigas do JavaScript

Funções

As funções são blocos de código reutilizáveis que podem ser definidos e chamados em qualquer parte do programa. Elas ajudam a organizar o código, promovem a reutilização e facilitam a compreensão do programa.

Declaração de Função:

- Forma tradicional de definir uma função.
- A função pode ser declarada antes ou depois de sua chamada.

```
function saudacao(nome) {
  console.log('Olá, ' + nome + '!');
}

// Chamando a função
saudacao('Alice'); // Output: Olá, Alice!
```

Expressão de Função:

- Definindo uma função como uma expressão.
- A função pode ser anônima ou ter um nome.

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

console.log(soma(5, 3)); // Output: 8
```

Arrow Functions:

- Introduzido no ECMAScript 6 (ES6).
- Sintaxe mais concisa para funções anônimas.

```
const quadrado = (x) => x * x;

console.log(quadrado(4)); // Output: 16
```

Parâmetros e Retorno:

- Funções podem ter parâmetros e retornar valores.
- Parâmetros são variáveis que recebem valores quando a função é chamada.

```
function multiplicacao(a, b) {
  return a * b;
}

let resultado = multiplicacao(3, 4);
console.log(resultado);
```

Escopo de Variáveis:

- Variáveis declaradas dentro de uma função têm escopo local.
- Variáveis fora da função não são acessíveis de dentro da função (a menos que passadas como parâmetros).

```
function escopoLocal() {
  let mensagem = 'Variável local';
  console.log(mensagem);
}
```

Funções como Argumentos:

- Funções podem ser passadas como argumentos para outras funções.

```
function operacaoMatematica(a, b, operacao) {
  return operacao(a, b);
}

const soma = (x, y) => x + y;
console.log(operacaoMatematica(5, 3, soma));
// Output: 8
```

Funções Recursivas

- Uma função que chama a si mesma.
- Útil para resolver problemas que podem ser divididos em subproblemas semelhantes.

```
function fatorial(n) {
  if (n === 0 || n === 1) {
    return 1;
  }
  return n * fatorial(n - 1);
}

console.log(fatorial(5)); // Output: 120
```

Conclusão

As funções são uma parte fundamental do JavaScript, permitindo a modularização e organização do código. Entender como declarar, chamar e utilizar funções é essencial para desenvolver aplicações web eficientes e escaláveis.