Funções

Resumo

Até agora, trabalhamos principalmente com estruturas de dados simples, o uso de loops e adquirimos mais conhecimento sobre Strings. Agora iremos avançar em nossos conhecimentos sobre o uso de funções para tornar nosso código mais organizado, legível e reutilizável.

Funções

Funções em JavaScript são blocos de código executável, nos quais podemos passar parâmetros e trabalhar com eles. Elas nos auxiliam na modularização dos nossos programas e na estruturação em blocos que executam tarefas específicas. Dessa forma, nosso código se torna mais legível e sustentável. Em outras palavras, as funções são componentes fundamentais na linguagem JavaScript.

Uma função é um procedimento em JavaScript, que consiste em um conjunto de instruções para executar uma tarefa ou calcular um valor. Para utilizar uma função, é necessário defini-la em algum lugar do escopo onde ela será chamada. Normalmente, as funções retornam um valor ao final da execução, que pode ser obtido utilizando a palavra reservada ‘return’. Elas são declaradas com a palavra-chave ‘function’ e, geralmente, possuem um nome para que possam ser invocadas posteriormente. Caso não possuam um nome, são chamadas de funções anônimas. Vejamos um exemplo de uma função:

```
function Multiplicar(a, b) {
  return a * b;             //retorna o produto de dois números
}
```

Fonte: Autoral

Você pode escrever esse bloco de código repetidamente sempre que precisar, ou você poderia escrevê-lo uma vez usando uma função, nomeando-a como 'Multiplicar' e passando os dois parâmetros 'a' e 'b', como fizemos acima. Após isso, este será o único código que você precisará chamar para executar todo o bloco.

A função mencionada tem o objetivo de calcular a multiplicação entre dois números. Perceba que podemos realizar esse cálculo várias vezes quando precisamos multiplicar dois números.

Para executar uma função, basta seguir estes passos:

```
var x = Multiplicar(4, 3);  //chama a função passando os argumentos
console.log(x);             //com valores e armazena em x
```

Fonte: Autoral

A definição da função (também chamada de declaração de função) consiste no uso da palavra-chave 'function', seguida por:

- Nome da Função.
- Lista de argumentos para a função, entre parênteses e separados por vírgulas.
- Declarações JavaScript que definem a função, entre chaves { }.

Como vimos, uma das coisas realmente úteis sobre as funções é que podemos passar dados para dentro delas. Se você colocar algu dados entre os parênteses, poderá passar esses dados para a função, que os utilizará quando for executada.

Agora, preenchemos os parênteses tanto na chamada da função quanto na sua definição. Os parênteses na chamada da função contêm um argumento. No exemplo, o argumento é o '4' e o '3'. E, como pode ser visto no exemplo acima, os parênteses da função agora contêm as duas variáveis 'a' e 'b', que podem ser utilizadas dentro da função.

O valor passado para uma função é utilizado no corpo da função para realizar alguma ação. Podemos usar o valor para especificar a mensagem de alerta ou realizar alguma operação dentro da função. No nosso caso, simplesmente realizamos o cálculo e retornamos o resultado utilizando a palavra-chave 'return'. Em seguida, o valor é atribuído à variável 'x'. Essa variável será exibida no console quando utilizarmos 'console.log(x)'.

A declaração de função não é a única maneira de definir uma nova função; você pode definir sua função dinamicamente usando o construtor **Function()** junto com o novo operador.

A seguir está a sintaxe para criar uma função usando o construtor Function () junto com o novo operador:

```
var variablename = new Function (Arg1, Arg2 ..., "corpo da função");
```

Fonte: Autoral

O construtor Function() espera qualquer número de argumentos de string. O último argumento é o corpo da função - este pode conter

instruções JavaScript arbitrárias, separadas umas das outras por ponto e vírgula.

É importante observar que o construtor Function() não recebe nenhum argumento que especifique um nome para a função criada.

```
1   var func = new Function("x","y", "return x * y");
2
3   function calculo(){
4       resultado = func(20,40)
5       console.log(resultado)
6   }
7
8   calculo();
```

Fonte: Autoral

Conteúdo Bônus

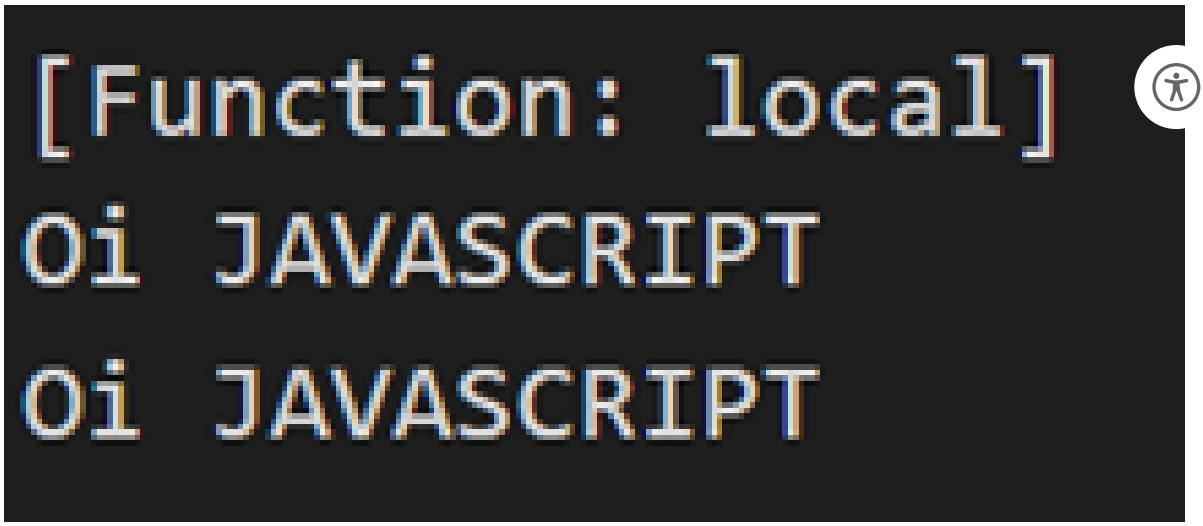
As funções podem ter outras funções dentro delas, o que cria novos escopos para as variáveis definidas dentro de cada uma. E, para acessar as funções internas de fora, é necessário invocá-las usando o operador de duplo parêntese ()(). Vamos analisar um exemplo:

```
var a = "Oi ";
function global() {
    var b = "JAVA";
    function local() {
        var c = "SCRIPT";
        return a + b + c;
    }
    return local;
}

console.log(global());
console.log(global()());
var testeFuncao = global();
console.log(testeFuncao());
```

Fonte: Autoral

O resultado ficaria assim:



Fonte: Autoral

Referência Bibliográfica

FLANAGAN, David. **JavaScript: O Guia Definitivo**. 6ª Ed. Porto Alegre: Bookman, 2013.

FREEMAN, Eric. **Use a cabeça!: programação JavaScript**. 1ª Ed. São Paulo: Alta Books, 2016.

ATIVIDADE PRÁTICA

Título da Prática: Como trabalhar com Funções?

Objetivos: Compreender o uso e os tipos de funções

Materiais, Métodos e Ferramentas: Para realizar esta prática vamos utilizar o Visual Studio Code

Prática

Bem-vindo ao programa de cálculo de média ponderada! Neste exercício, você será responsável por criar um script em Node.js para calcular a média ponderada de um aluno, com base em três notas fornecidas pelo usuário. A média ponderada é uma forma de calcular a média levando em consideração pesos diferentes para cada nota.

Você precisará utilizar a biblioteca `readline-sync` para permitir a interação com o usuário através do console. O programa solicitará que o usuário informe as três notas do aluno e, em seguida, calculará a média ponderada dessas notas, utilizando os seguintes pesos: 2, 3 e 5, respectivamente.

Após realizar os cálculos, o programa exibirá a nota da média ponderada no console. Lembre-se de que a média ponderada é uma medida importante para avaliar o desempenho dos alunos, uma vez que determinadas disciplinas ou avaliações podem ter maior peso na nota final.

Vamos lá! Mãos à obra para criar um script eficiente em Node.js que irá calcular a média ponderada do aluno com base nas notas fornecidas. Boa sorte!

Resolução

Aqui está um exemplo de script em Node.js que utiliza a biblioteca `readline-sync` para ler as notas do usuário, calcular a média ponderada e exibir a nota média no console:

```javascript
const readline = require('readline-sync');

function calcularMediaPonderada(notas) {
  const pesos = [2, 3, 5]; // Pesos das notas (exemplo: 2, 3 e 5)
  let somaPesos = 0;
  let somaNotas = 0;

  for (let i = 0; i < notas.length; i++) {
    somaPesos += pesos[i];
    somaNotas += notas[i] * pesos[i];
  }

  const media = somaNotas / somaPesos;
  return media;
}

function lerNotas() {
  const notas = [];

  for (let i = 0; i < 3; i++) {
    const nota = parseFloat(readline.question(`Digite a nota ${i + 1}: `));
    notas.push(nota);
  }

  return notas;
}

function main() {
  console.log('Calculadora de Média Ponderada\n');

  const notas = lerNotas();
  const media = calcularMediaPonderada(notas);

  console.log(`\nA média ponderada do aluno é: ${media.toFixed(2)}`);
}

main();
```

Neste exemplo, o usuário será solicitado a digitar as três notas do aluno, e a média ponderada será calculada usando os pesos 2, 3 e 5 para as notas respectivamente. Por fim, o resultado será exibido no console.

Certifique-se de ter a biblioteca `readline-sync` instalada. Para instalá-la, você pode executar o seguinte comando no terminal:

```
npm install readline-sync
```

Espero que isso possa te ajudar!

Ir para exercício