Boas-vindas!

Agora você faz parte da maior comunidade de aprendizagem online e ao vivo da América Latina!

Como você chega?

1

2

3

CODERHOUSE

Esta aula será

- gravada

Resumo
da aula anterior

- ✓ Operador if, else e suas variantes.
- ✓ Variáveis booleanas.
- ✓ Operações relacionais.
- ✓ Operações lógicas.
- ✓ Uso dos operadores nas condicionais.

Dúvidas?

Antes de iniciarmos...

CORREÇÃO ATIVIDADE EXTRA

Criar um algoritmo com uma condicional

Desenvolva um sistema que:

- ✓ Solicite um número através do prompt
- ✓ Verifica se está entre 10 e 50
- ✓ Em caso positivo, mostre uma mensagem indicativa em um alert
- ✓ Em caso negativo, mostre uma mensagem indicativa em um console

O tutor irá corrigir e tirar as dúvidas.

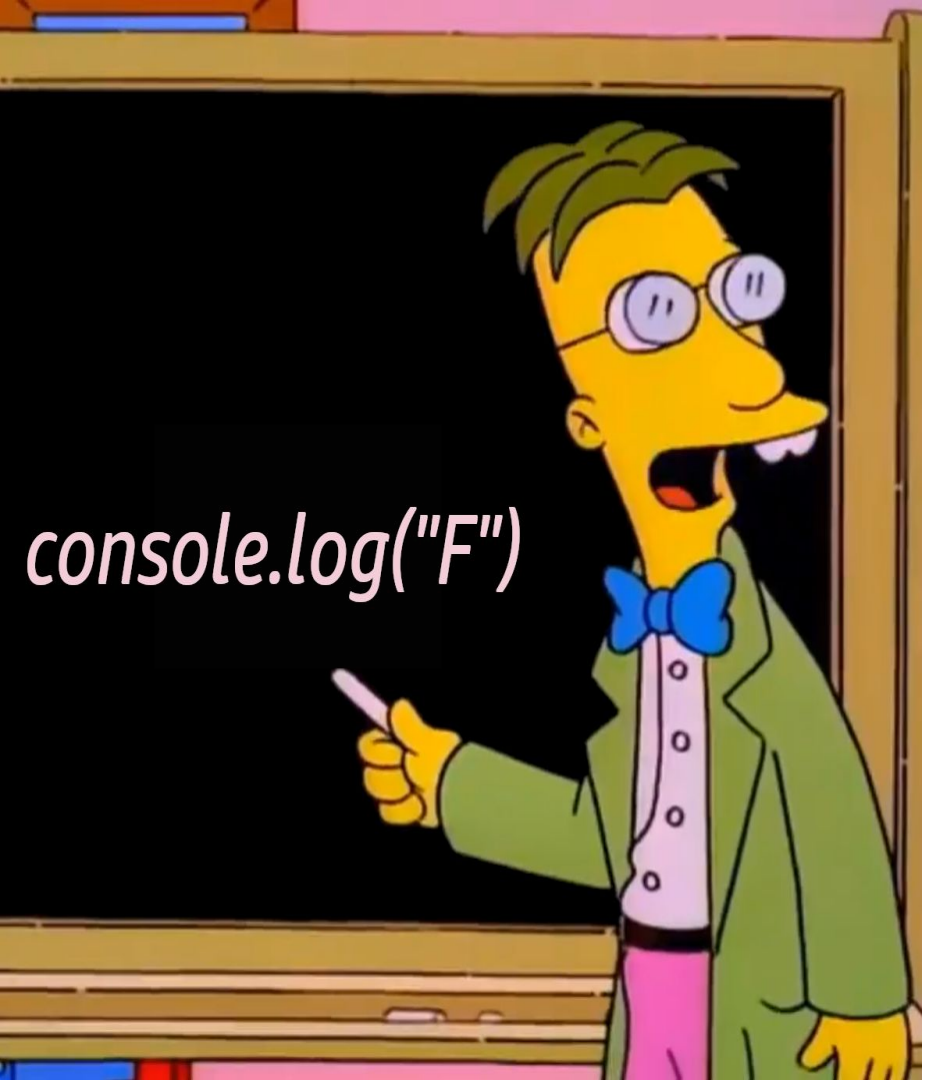
Aula 4. Javascript

Funções

Objetivos da aula

- O que é uma função e como nos ajuda a escrever menos código?
- O que são os argumentos e parâmetros
- O que é uma função anônima?
- O que é uma arrow function?

Funções e propriedades básicas



Funções

Quando se desenvolve uma aplicação ou site, é muito comum usar as mesmas instruções uma ou outra vez.

Na programação, uma **função é um conjunto de instruções que se agrupam para realizar uma tarefa concreta**, que depois pode ser reutilizada ao longo de diferentes instâncias do código.

CODERHOUSE

Quais as vantagens das Funções?

As principais vantagens do uso de funções são:

- ✓ Evitam instruções duplicadas (Princípio DRY)
- ✓ Solucionam um problema complexo utilizando tarefas simples (Princípio KISS)
- ✓ Focam nas tarefas prioritárias para o programa (Princípio YAGNI)
- ✓ Fornecem maior organização e clareza ao código
- ✓ Proporcionam facilidade e rapidez para fazer modificações

Declaração

É feita através da palavra reservada **function.** É recomendável que não tenha espaços e com os característicos parênteses **()** vindo logo em seguida.

O conteúdo da função é escrito entre as chaves.

O nome de uma função não pode ser repetido em outra função.

```
function saudacao() {
    console.log("Olá, estudantes!");
}
```

Declaração

Uma vez que declaramos a função, podemos usá-la em qualquer outra parte do código quantas vezes quisermos. Para executar uma função, é só escrever o nome dela e finalizar a sentença com os parênteses. Isso é conhecido como a **chamada da função.**

Onde fizermos a chamada, se interpretará as instruções definidas na função.

```
saudacao();
```

Exemplo ao vivo

Se precisarmos solicitar um nome ao usuário para depois mostrá-lo em um alert, normalmente poderíamos fazer este código:

```
var nome = prompt("Inserir nome")
alert("O nome inserido é " + nome)
```

Se quisermos repetir, podemos copiar e colar o código.

```
var nome1 = prompt("Inserir nome")
alert("O nome inserido é " + nome)

var nome2 = prompt("Inserir nome")
alert("O nome inserido é " + nome2)
```

Exemplo ao vivo

Poderíamos então criar uma função que se chame solicitarNome() para que solicite a informação ao usuário pela quantidade de vezes que precisarmos

```
function solicitarNome() {
    var nome = prompt("Inserir nome")
    alert("O nome inserido é " + nome)
}
```

Para chamar a função, a invocamos (chamamos) em outra parte do código quantas vezes forem necessárias:

```
solicitarNome();
solicitarNome();
solicitarNome();
solicitarNome();
```

Para praticar

Duração: **15 minutos**

CODERHOUSE

ATIVIDADE EM SALA

Para praticar

Crie uma função que:

- ✓ Receba um número do usuário
- ✓ Verifique se esse número é par
- ✓ Exiba um alert com o resultado
- ✓ Lembrando que números pares são aqueles cujo resto da divisão por 2 é 0.
- ✓ Para calcular o resto da divisão utilizamos a operação de módulo (representada pelo símbolo %)

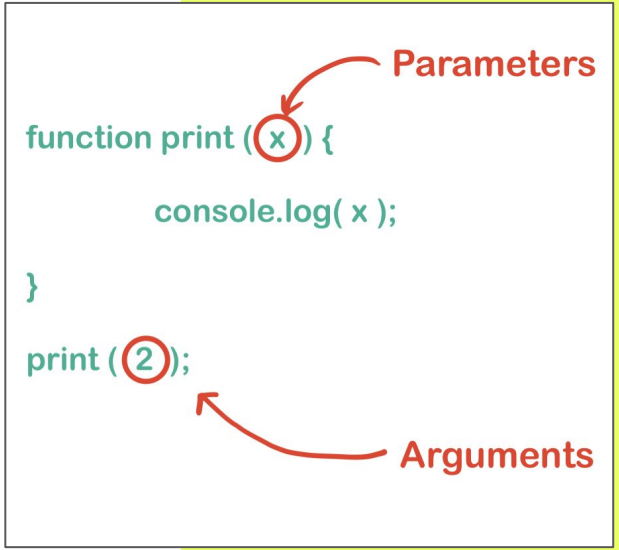
Execute sua função para 3 valores diferentes de entrada.

Argumentos e parâmetros

Uma função simples pode não precisar de nenhum dado para funcionar. Mas quando começamos a codificar **funções mais complexas**, nos deparamos com a necessidade de receber informações.

Quando enviamos à função um ou mais valores para serem utilizados em suas operações, estamos falando dos **argumentos da função.**

Para que a função receba esses valores, usamos os **parâmetros da função**, que ficam entre os parênteses, logo após seu nome.



CODERHOUSE

Argumentos e parâmetros

No exemplo temos uma função que trabalha com dois parâmetros (palavra1 e palavra2) para gerar seu resultado. Seus valores são definidos na chamada da função. Na primeira chamada os argumentos são: “Olá” e “Coder”. Na segunda chamada os argumentos são “Cursando” e “JS”.

Nesse caso, a primeira string que enviamos é atribuída em palavra1, e a segunda string em palavra2; configurando as saídas segundo a lógica definida na função.

```
function unirPalavras(palavra1, palavra2) {
    console.log(palavra1 + " " + palavra2);
}

unirPalavras("Olá", "Coder"); // -> “Olá Coder”
unirPalavras("Cursando", "JS"); // -> “Cursando JS”
```

CODERHOUSE

Exemplo ao vivo

Somar e exibir

```
//Declaração de variável para armazenar o resultado da soma
var resultado = 0;

//Função que soma dois números e atribui o resultado
function somar(primeiroNumero, segundoNumero) {
    resultado = primeiroNumero + segundoNumero
}

//Função que mostra o resultado no console
function mostrar(mensagem) {
    console.log(mensagem)
}

//Chamamos primeiro para somar e depois para mostrar
somar(6, 3);
mostrar(resultado);
```

Resultado de uma Função

Resultado de uma Função

No exemplo anterior, somamos dois números a uma variável declarada anteriormente. No entanto, as funções podem gerar um valor de retorno usando a palavra **return**, obtendo esse valor quando a função é chamada.

```
function somar(primeiroNumero, segundoNumero) {
    return primeiroNumero + segundoNumero;
}
var resultado = somar(5, 8);
```

CODERHOUSE

Resultado de uma Função

A função pode se comportar como uma operação que gera valores (como nas operações matemáticas e lógicas anteriores).

No espaço onde se chama a função, é gerado um novo valor:
esse valor é aquele definido pelo **return** da função.

```
var resultado = somar(5, 8);

console.log(resultado)  // ⇒ 13
```

Exemplo: Calculadora

```
function calculadora(primeiroNumero, segundoNumero, operacao) {
    if (operacao == "+") {
        return primeiroNumero + segundoNumero;
    }
    else if (operacao == "-") {
        return primeiroNumero - segundoNumero;
    }
    else if (operacao == "*") {
        return primeiroNumero * segundoNumero;
    }
    else if (operacao == "/") {
        return primeiroNumero / segundoNumero;
    }
    else {
        return 0;
    }
}
console.log(calculadora(10, 5, "*"));
```

Para praticar

Duração: **15 minutos**

CODERHOUSE

ATIVIDADE EM SALA

Para praticar

Crie uma função que faz subtrações:

- ✓ Receba dois números
- ✓ Verifique qual é o número maior
- ✓ Faça uma subtração do maior pelo menor
- ✓ Exiba o resultado com o console

Break

5 minutos e voltamos!

CODERHOUSE

Break

10 minutos e voltamos!

CODERHOUSE

Função anônima e Arrow Function

Funções Anônimas

Uma **função anônima** é uma função que é definida **sem nome** e é utilizada para ser enviada como parâmetro ou atribuída a uma variável.

No caso de atribuí-la a uma variável, ela pode ser chamada usando o identificador da variável declarada.

```
var soma = function (a, b) { return a + b }
var subtrair = function (a, b) { return a - b }

console.log(soma(15, 20))
console.log(subtrair(15, 5))
```

CODERHOUSE

Arrow Function(Funções seta)

Identificamos as arrow functions como **funções anônimas de sintaxe simplificada.** Estão disponíveis a partir da versão ES6 do JavaScript e não usam a palavra **function**, mas sim **=> (seta)** entre os parâmetros e o bloco.

```
var soma = (a, b) => { return a + b }
//Se for uma função de uma só linha com um retorno, não é
necessário escrever o corpo.
var subtrair = (a, b) => a - b;

console.log(soma(15, 20))
console.log(subtrair(20, 5))
```

CODERHOUSE

Exemplo: Calculadora de preço

```
var somar = (a, b) => a + b
var subtrair = (a, b) => a - b
var calcularICMS = x => x * 0.21

var precoProduto = 500
var desconto = 50

//Calculo o PrecoProduto + ICMS - Desconto
var icms = calcularICMS(precoProduto)
var soma = somar(precoProduto, icms)
var novoPreco = subtrair(soma, desconto)
console.log(novoPreco)
```

Para praticar

Duração: **10 minutos**

CODERHOUSE

ATIVIDADE EM SALA

Para praticar

- ✓ Refaça o exemplo da calculadora usando apenas arrow functions.
- ✓ Exiba o resultado com o alert.

Perguntas?

Como foi a aula?

1

Que bom

O que foi super legal na aula e podemos sempre trazer para as próximas?

2

Que pena

O que você acha que não funcionou bem e precisamos melhorar?

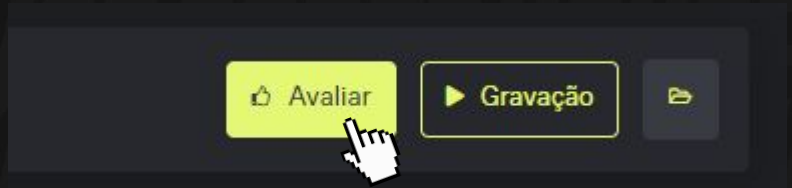
3

Que tal

Qual sugestão deveríamos tentar em próximas aulas?

CODERHOUSE

O que você achou da aula?



Seu feedback vale pontos para o Top 10!! 😎

Deixe sua opinião!

1. Acesse a plataforma
2. Vá na aula do dia
3. Clique em Avaliar

CODERHOUSE

Resolver um processo complexo

DESAFIO COMPLEMENTAR

CODERHOUSE

DESAFIO COMPLEMENTAR

Resolver um processo complexo

Instrução:

- ✓ Codifique ao menos três funções cujas instruções permitam resolver um problema particular, segmentado em tarefas.
- ✓ A informação a ser processada deve ser inserida pelo usuário, e o resultado do processamento será visualizado em uma saída (alert ou console.log).

Recomendação:

- ✓ Para realizar essa tarefa, sugerimos pensar em um processo complexo, decompô-lo em ao menos três partes e criar funções que se encarreguem de cada uma dessas partes.

Exemplos:

- ✓ Cálculo do ICMS:
  1) Inserir preço de custo
  2) Somar ICMS
  3) Mostrar preço calculado.

- ✓ Determinar se dois número são múltiplos:
  1) Inserir números a serem verificados
  2) Verificar se um é múltiplo do outro
  3) Mostrar resultado.

Formato:

- ✓ Link do código no GitHub e da publicação no GitHubPages

Resumo
da aula de hoje

- ✓ Construindo funções
- ✓ Argumentos, parâmetros e retorno
- ✓ Funções anônimas
- ✓ Arrow functions (Funções seta)

Você quer saber mais?
Deixamos material extra da aula

MATERIAL AMPLIADO

Recursos multimídia

Referências

- ✓ Funções | **Developer Mozilla**
- ✓ Entendendo as funções e suas formas de uso | **React Brasil**

CODERHOUSE

Obrigado por estudar conosco!