

Técnico em Desenvolvimento de Sistemas



**Disciplina de
Fundamentos de Programação Orientada a Objetos**

INSTRUTORES

FUNDAMENTOS DE PROGRAMAÇÃO ORIENTADA A OBJETO



Rodrigo Alves Nunes

rodrigo.alves17@senaisp.edu.org.br

Matheus Souza de Oliveira

matheus.oliveira238@senaisp.edu.br



Funções de callback são funções que são passadas como argumentos para outras funções.

São executadas após a **conclusão de uma operação assíncrona** ou **quando um evento específico ocorre**.

As funções de callback são fundamentais em JavaScript, especialmente quando se trata de **operações assíncronas**, como **carregar dados de um servidor** ou lidar com **eventos de usuário**.

FUND. DE PROG. ORIENTADA A OBJETO



FUNÇÕES DE CALLBACK

Trabalhando com Funções de Callback

em eventos disparado pelo usuário na interface:

Você também pode usar funções de callback para lidar com **eventos de usuário**, como eventos de clique:

Insira o script no final
do elemento "body"

```
const botao = document.getElementById("id_botao");

function evento_de_click() {
  console.log("O botão foi clicado!");
}

botao.addEventListener("click", evento_de_click);
```

FUND. DE PROG. ORIENTADA A OBJETO



FUNÇÕES DE CALLBACK

Trabalhando com Funções de Callback

em eventos de tempo:

Você também pode usar funções de callback para serem executadas depois de um intervalo específico de tempo (**setTimeout**)

Ex1:

```
function minhaCallback() {  
  console.log("A função de callback foi chamada"  
    + " após o tempo especificado.");  
}  
  
// A função de callback será chamada após 2 segundos.  
setTimeout(minhaCallback, 2000);
```

FUND. DE PROG. ORIENTADA A OBJETO



FUNÇÕES DE CALLBACK

Trabalhando com Funções de Callback

em eventos de tempo:

Você também pode usar funções de callback para serem executadas a cada intervalo específico de tempo (**setInterval**)

Ex2:

```
function minhaCallback() {  
  console.log("A função de callback foi chamada"  
    + " após o tempo especificado.");  
}  
  
// A função de callback será chamada a cada 2 segundos.  
setInterval(minhaCallback, 2000);
```

FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Existem várias formas de declarar as funções. Até o momento usamos apenas a forma [declarativa / nomeada](#).

Forma 1 – Declaração de função

Nesse formato são **atribuídos nomes as funções** e elas **poderão ser chamadas** de por meio de seu nome **em vários pontos do código** e **sempre pelo mesmo nome** e **quantas vezes forem necessárias**. Possibilita ser reutilizada diversas vezes.

```
function minhaFuncao(parametro1, parametro2) {  
  // Corpo da função  
  // ...  
  let valor = parametro1 + parametro2;  
  
  return valor;  
}
```

FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 1 - Declaração de função (EXEMPLO)

```
function soma(a, b) {  
    return a + b;  
}  
  
let resultado = soma(5, 3);  
console.log(resultado); // Saída: 8
```


FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 2 – Expressão de função ou função anônima

Você também pode criar funções e atribuir a variáveis como expressões de função. Nesse formato as funções são criadas sem **nome** (**funções anônimas**) e atribuídas a **variáveis**. Só poderão ser utilizadas após serem criadas.

```
let variavel_func = function (parametro1, parametro2) {  
    // Corpo da função  
    // ...  
    let valor = parametro1 + parametro2;  
  
    return valor;  
}
```

FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 2 – Expressão de função ou função anônima (EXEMPLO)

```
let var_func_soma = function (a, b) {  
    return a + b;  
};
```

```
let resultado = var_func_soma(5, 3);  
console.log(resultado); // Saída: 8
```

FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 2 – Expressão de função ou função anônima

Qual a vantagem de se utilizar esse tipo de função?

São frequentemente usadas como **callbacks**, especialmente quando você precisa de uma **função simples e única para uma tarefa específica**. Em momentos que a função a ser criada não será reutilizada em outro trecho do código eliminando a necessidade de ser declarada.

Aqui está um exemplo usando uma **função anônima** como **callback** com **setTimeout**:

```
setTimeout(function () {  
    console.log("Essa função anônima é um callback.");  
}, 2000);
```

FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 2 – Expressão de função ou função anônima

Qual a vantagem de se utilizar esse tipo de função?

Podemos utilizar funções anônimas adicionar funções de **callback** em eventos.

```
const botao = document.getElementById("id_botao");  
  
botao.addEventListener("click", function () {  
    console.log("O botão foi clicado!");  
});
```


FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 2 – Expressão de função ou função anônima

Podemos deixar mais compacto ainda em uma linha somente o exemplo anterior.

```
const botao = document.getElementById("id_botao");  
  
botao.addEventListener("click", function () {  
    console.log("O botão foi clicado!");  
});
```

```
document.getElementById("id_botao").addEventListener("click",  
    function () {  
        console.log("O botão foi clicado!");  
    }  
);
```



FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 3 – Arrow Functions ou Funções Seta

As **arrow functions** são uma forma mais concisa de escrever funções em JavaScript. Elas foram introduzidas na versão **ECMAScript 6 (ES6)** e são especialmente úteis para **funções anônimas** e **pequenas**.

Possuem uma sintaxe mais enxuta em comparação com as funções tradicionais.

```
// Sintaxe básica
const minhaArrowFunction = (param1, param2) => {
  // Corpo da função
  return resultado;
};
```

FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 3 – Arrow Functions ou Funções Seta

Vamos **comparar** as **arrow functions** com as **funções anônimas**:

```
// Função Anônima
const minhaFuncAnonima = function (param1, param2) {
  // Corpo da função
  return resultado;
};
```

```
// Arrow function
const minhaArrowFunction = (param1, param2) => {
  // Corpo da função
  return resultado;
};
```

FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 3 – Arrow Functions ou Funções Seta

As **arrow functions** são frequentemente usadas como alternativa às funções tradicionais **quando a função é simples** e/ou envolve **uma única expressão de retorno**.

Aqui está um exemplo:



```
// Arrow function
const minhaArrowFunction = (param1, param2) => {
  // Corpo da função
  return resultado;
};
```

```
// Arrow function
const minhaArrowFunction = (param1, param2) => resultado;
```



FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 3 – Arrow Functions ou Funções Seta

Vamos **comparar** as **exemplos**:

```
// Função Declarativa  
function soma(a, b) {  
  return a + b;  
}
```

```
// Função Anônima atribuída a uma variável  
let var_func_soma = function (a, b) {  
  return a + b;  
};
```

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



```
// Arrow function  
const soma = (a, b) => a + b;
```

FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 3 – Arrow Functions ou Funções Seta

Vamos **comparar** as **exemplos**:

```
// Função Declarativa  
function soma(a, b) {  
  return a + b;  
}
```

```
// Função Anônima atribuída a uma variável  
let var_func_soma = function (a, b) {  
  return a + b;  
};
```

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



```
// Arrow function  
const soma = (a, b) => a + b;
```

FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS


Forma 3 – Arrow Functions ou Funções Seta

Usando **Arrow Functions** **com eventos**:

```
const botao = document.getElementById("id_botao");

function evento_de_click() {
  console.log("O botão foi clicado!");
}

botao.addEventListener("click", evento_de_click);
```



```
document.getElementById("id_botao").addEventListener("click", () => {
  console.log("O botão foi clicado!");
});
```

FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS


Forma 3 – Arrow Functions ou Funções Seta

Usando **Arrow Functions** **com eventos**:

```
const botao = document.getElementById("id_botao");

function evento_de_click() {
  console.log("O botão foi clicado!");
}

botao.addEventListener("click", evento_de_click);
```



```
document.getElementById("id_botao").addEventListener("click", () => {
  console.log("O botão foi clicado!");
});
```


FUND. DE PROG. ORIENTADA A OBJETO



DECLARAÇÃO DE FUNÇÕES - TIPOS

Forma 3 – Arrow Functions ou Funções Seta

Usando **Arrow Functions** *com eventos de tempo*:

```
setTimeout(function () {  
  console.log("Essa função anônima é um callback.");  
}, 2000);
```



```
setTimeout(() => {  
  console.log("Essa função anônima é um callback.");  
}, 2000);
```