# Tema 03 – Objetos, Funções e Eventos

## Introdução

Olá meus alunxs, sejam bem vindos do curso de Frontend do Instituto da Oportunidade Social. Nessa aula, você aprenderá sobre funções, objetos e eventos no JavaScript.

## Criando projeto inicial

Vamos criar um projeto para testar toda teoria que vimos nesse tema. Siga os passos para criar o projeto:

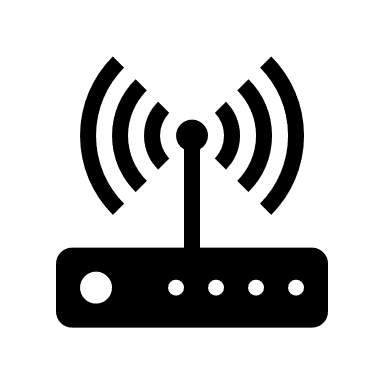
1. Abra o VS Code e escolha um diretório de trabalho para o seu projeto.
2. Crie um diretório para seu projeto com o nome representativo, por exemplo, 03\_JS\_Funcoes\_Objetos\_Eventos.
3. Crie um arquivo dentro do diretório do projeto com o nome index.html.
4. Insira o seguinte código no seu arquivo index.html.

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">      <head>          <meta charset="UTF-8" />  <link rel="shortcut icon" href="#" />          <title>Funções, Objetos, Eventos e Strings</title>      </head>      <body>          <script src="main.js"></script>      </body>  </html> |

1. Esse código mostra a marcação <script> sem nenhum código JavaScript entre a abertura e o fechamento da tag, apenas o atributo **src** com o valor main.js. Isso significa que o código JavaScript está em um arquivo externo. Portanto, temos que criar esse novo arquivo **main.js** dentro do diretório do projeto.
2. Ao criar o seu arquivo a hierarquia do diretório do projeto deverá ficar assim:

A picture containing logo

Description automatically generated

1. Vamos deixar o arquivo main.js vazio e a medida que vamos aprendendo coisas novas vamos completando a implementação do código JavaScript.
2. Abra o arquivo index.html, clique no botão **Go Live** da extensão **Live Server** e abra as Ferramentas de desenvolvimento do navegador web (Atalho **F12**).

## Funções

Funções são usadas em programação para executar ações que são rotineiramente executadas em um programa. Uma função é um bloco de código implementado para executar uma tarefa em particular. Para executar uma função devemos sempre invocá-la (chamá-la) dentro do código.

A sintaxe para criar funções em JS é:

|  |
| --- |
| function myFunc (valor1, valor2) {  return valor1 \* valor2;  } |

|  |
| --- |
| Atenção: O nome de funções segue as mesmas regras de variáveis deve começar por letras ou underline ( \_ ) e pode conter números e não pode conter caracteres especiais. O nome também não pode ser uma palavra-chave ou palavra reservada. Programadores geralmente usam somente nomes de funções como camelCase ou CamelCase, isto é, a primeira letra de cada palavra deve ser escrita em maiúscula. Isso padroniza o código e facilita identificar o que é variável e o que é função. |

No exemplo mostrado anteriormente, a função com o nome **myFunc** espera receber dois parâmetros (**valor1** e **valor2**) e retorna o resultado da multiplicação dos dois valores (**return valor1 \* valor2;**). Quando o JavaScript encontra a instrução **return**, a função para de ser executada. Se a função foi chamada a partir de uma instrução, o JavaScript "retornará" para executar o código após a instrução de chamada. As funções geralmente calculam um valor de retorno. O valor de retorno é "retornado" de volta ao objeto que a chamou.

### Vamos praticar

No projeto criado no início desse material, vamos implementar duas funções para você entender melhor como tudo funciona. Siga os passos:

1. Abra o arquivo **main.js** e insira o seguinte código:

|  |  |
| --- | --- |
| 1 | // Funções  function addNums(num1 = 1, num2 = 1) {      return num1 + num2;  }  let x = addNums(4, 5); // Chamada da função addNums  console.log(x);  let z = myFunc(4, 5); // Chamada da função myFunc  console.log(z);  function myFunc(num1, num2) {      return num1 \* num2;  } |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |

Podemos observar a declaração de duas funções: a primeira com o nome **addNums**, que está nas linhas 2 a 4, e a segunda com o nome **MyFunc**, que está nas linhas 12 a 14. Note, também, que no JS você pode invocar a função antes da sua declaração, por exemplo a chamada da função **myFunc** está na linha 9, antes da sua declaração.

Note, também, que os parâmetros da função **addNums** possui valores padrão (num1 = 1, num2 = 1), ou seja, se na chamada da função não passarmos nenhum valor, os argumentos num1 e num2 assumirão o valor 1.

1. Com a arquivo index.html aberto no seu navegador, você pode salvar o arquivo main.js e ver o resultado da execução do código JavaScript no console.

Graphical user interface, text, application, Word

Description automatically generated

### Arrow Function

O conceito de **Arrow Function** foi introduzido em 2015 no ES6. Acostume-se bem com essa forma mais prática de declarar funções, pois muitos exemplos que você encontrará na internet usará esse tipo de declaração. A sintaxe de uma **arrow function** é:

|  |
| --- |
| const hello = () => {  return "Olá Arrow Function!";  } |

Basicamente, você deve atribuir a função a uma variável declarada com a palavra **const** e utilizar o operador **=>** para indicar o bloco da função. A palavra **const** pode ser ocultada se você não estiver usando o **Strict Mode**, mas vamos sempre seguir as dicas de bons programadores e nunca deixar de declarar uma variável.

No exemplo mostrado anteriormente, a função não tem nenhum parâmetro de entrada (dentro dos parênteses da função está vazio). Toda função pode ter zero ou mais parâmetros de entrada. Mas também podemos fazer uma versão da função addNums como **arrow function**, vejamos o exemplo:

|  |
| --- |
| const addNums2 = (num1 = 1, num2 = 1) => {  return num1 + num2;  } |

### Vamos praticar

1. No arquivo **main.js**, insira o seguinte código:

|  |  |
| --- | --- |
| 16 | //Arrow functions  console.clear();  const hello = () => {      return 'Olá Arrow Function!';  };  console.log(hello);  // Retorna o objeto função  console.log(hello()); // Executa a função e imprime a string no return |
| 17 |
| 18 |
| 19 |
| 20 |
| 21 |
| 22 |
| 23 |

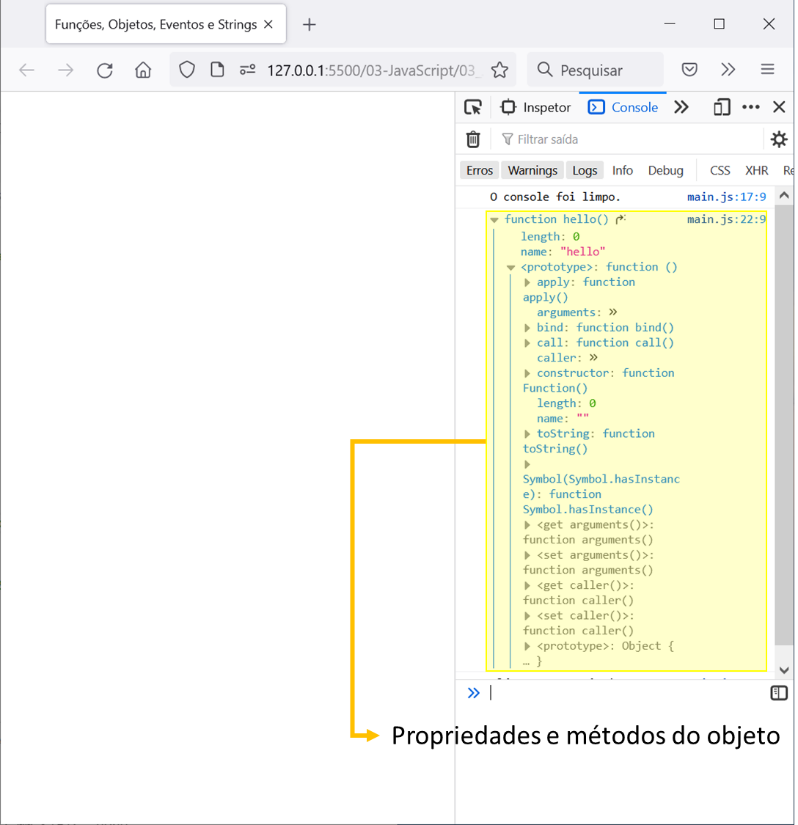
Observe na linha 22, quando usamos apenas o nome da função sem os parênteses, o console exibe o objeto **function**. Lembre-se disso, pois a medida que avançamos no estudo de JS, vamos fazer **debbug** para saber o tipo de objeto que estamos manipulando e não estaremos interessados somente no valor do objeto.

1. Ao salvar o arquivo **main.js**, é possível ver as informações impressas no console.

Graphical user interface, application, Word

Description automatically generated

1. Vamos dar uma olhada no objeto **function hello**. Clique na seta antes do objeto para expandir o objeto e também clique na seta do **prototype** para expandir completamente o objeto.



Observe que o objeto possui diversas, propriedade, métodos e protótipos. Não se preocupe com isso agora, mas isso será bem útil mais adiante e, também, é bom você saber que existe muita coisa legal para aprenderemos. E lembre-se da fala de Issac Newton ,que foi usada na série Dark:

|  |
| --- |
| “What we know is a drop, what we don't know is an ocean.” ― **Isaac Newton**. |
| “O que sabemos é uma gota, o que não sabemos é um oceano.” ― **Issac Newton.** |

1. Vamos inserir a versão da função addNum com **Arrow Function** no nosso código.

|  |  |
| --- | --- |
| 25 | console.clear();  const addNums2 = (num1 = 1, num2 = 1) => {      return num1 + num2;  };  let soma = addNums2(5, 10);  console.log(soma); |
| 26 |
| 27 |
| 28 |
| 29 |
| 30 |
| 31 |

1. O resultado pode ser visto no console do navegador:

Graphical user interface, application, Word

Description automatically generated

## Objetos

Na vida real um carro é um objeto e esse objeto possui propriedades como, por exemplo, nome, modelo e peso, e métodos (ações a serem executadas) tais como: ligar, dirigir, frear e parar. As propriedades de um carro podem ser as mesmas, mas os valores são diferentes de carro para carro. Os métodos de um carro podem também ser os mesmos, mas a execução pode ser ligeiramente diferente de carro para carro, por exemplo, um carro pode ligar utilizando a chave e outro pode ser com um botão.

Graphical user interface, application

Description automatically generated

No JS, um objeto é uma coleção de dados e/ou funcionalidades relacionadas (que geralmente consistem em diversas variáveis e funções — que são chamadas de propriedades e métodos quando estão dentro de objetos). Por enquanto, pense que toda declaração de variável ou função feita no JS é um objeto. As variáveis mostradas abaixo são objetos:

|  |  |  |
| --- | --- | --- |
| let marca = "Fiat"; | const carro = {  type:"Fiat",  model:"500",  color:"white"  }; | const pessoa = {  firstName: "John",  lastName: "Doe",  age: 50,  eyeColor: "blue"  }; |

Os objetos **carro** e **pessoa** são bem interessantes, pois eles são formados por um conjunto de valores ou o que chamamos de Array de objetos. Falaremos mais sobre array de objetos mais adiante. Vamos abordar cada assunto com calma e aprofundar gradativamente.

## Eventos

Eventos são ações executadas quando algo acontece na página web, ou seja, é a reação algum estímulo ou interação em elemento HTML. Esses eventos do HTML normalmente chamam funções do JavaScript. Eventos HTML são coisas que o navegador pode fazer ou algo que o usuário pode fazer. Os principais eventos do HTML são:

|  |  |
| --- | --- |
| **Event** | **Description** |
| onchange | Um elemento HTML é alterado. |
| onclick | O usuário clica em um elemento HTML. |
| onmouseover | O usuário move o mouse sobre o elemento HTML. |
| onmouseout | O usuário tira o mouse de cima do elemento HTML. |
| onkeydown | O usuário pressiona um Tecla do teclado. |
| onload | O navegador termina de carregar a página. |

|  |
| --- |
| **Para saber mais**: se você quiser saber mais sobre eventos HTML pode consultar o link do w3schools, que está disponível em: <https://www.w3schools.com/jsref/dom_obj_event.asp> |

### Vamos praticar

Vamos implementar nesse momento os eventos **onmouseover**, **onload** e **onclick**, pois esse eventos são mais legais quando estivermos acessando o DOM. Vamos ter uma aula somente sobre JavaScript e DOM, então aguardem. Vamos precisar modificar o arquivo index.html e o main.js. Siga os passos para implementar alguns eventos utilizando HTML e JavaScript.

1. Atualize o arquivo index.html com o código mostrado a seguir:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">      <head>          <meta charset="UTF-8" />          <link rel="shortcut icon" href="#" />          <title>Funções, Objetos, Eventos e Strings</title>      </head>      <body onload="boasVindas()">          <button onclick="eventClique()">Clique aqui</button>          <script src="main.js"></script>      </body>  </html> |

Nesse código fizemos duas atualizações, a primeira é na marcação <body> a inserção do evento **onload**, que chama a função **boasVindas()**, e a segunda alteração é a inserção de um botão com o evento **onclick**, que invoca a função **eventClique()**. Essas funções serão implementadas no arquiv **main.js** e você pode dar qualquer nome para elas desde que respeite as regras para nomes de funções.

1. Salve o arquivo index.html e vamos para o arquivo **main.js**. Insira o código abaixo com as declarações das funções.

|  |  |
| --- | --- |
| 33 | // Eventos  console.clear();  const boasVindas = () => {      alert('Bem vindo a nossa página');      console.log('Bem vindo a nossa página');  };  const eventClique = () => {      console.log('Você clicou no botão');  }; |
| 34 |
| 35 |
| 36 |
| 37 |
| 38 |
| 39 |
| 40 |
| 41 |
| 42 |

1. Salve o arquivo **main.js** e abra a página no navegador (arquivo **index.html**).

Graphical user interface, text, application, chat or text message

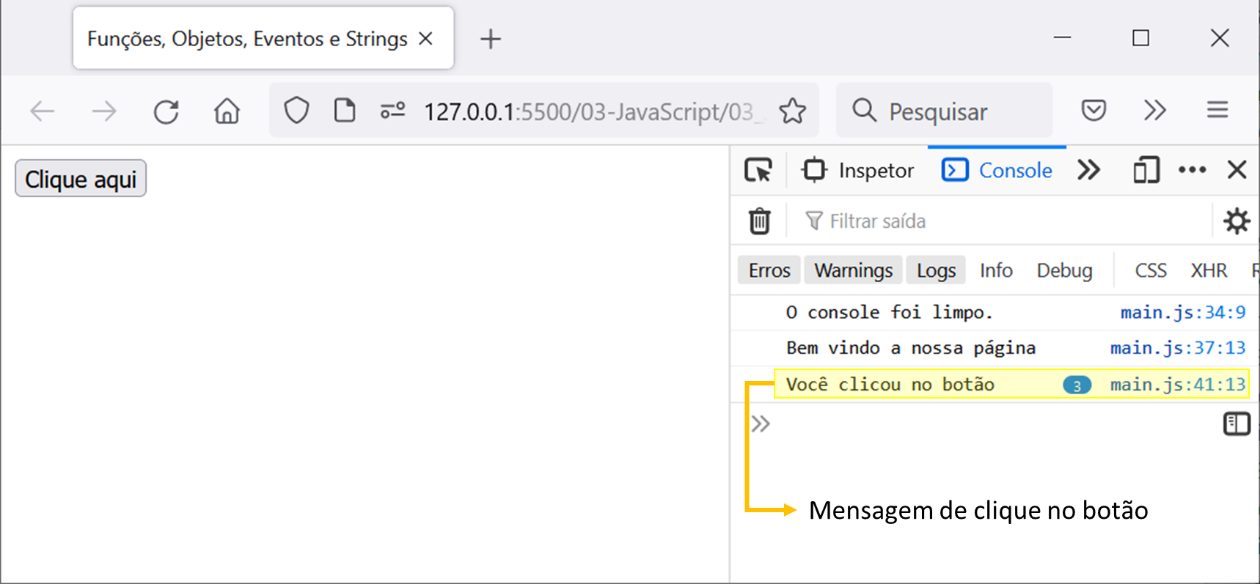
Description automatically generated

Ao carregar a página o evento **onload** é disparado e uma mensagem de alerta é disparada (instrução **alert()** na linha 36). Após você clicar no botão **OK**, a mesma mensagem irá aparecer no console (instrução **console.log()** na linha 37).

Graphical user interface, application, Word

Description automatically generated

1. Essas mensagens aparecem sempre que você recarregar a página.
2. Se você clicar no botão verá a mensagem no console com uma **badge** mostrando quantas vezes o botão foi clicado.



1. Agora, vamos fazer mais uma atualização no arquivo index.html como mostra o código a seguir:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">      <head>          <meta charset="UTF-8" />          <link rel="shortcut icon" href="#" />          <title>Funções, Objetos, Eventos e Strings</title>      </head>      <body onload="boasVindas()">          <h2 onmouseover="mouseEmCima()">Passe o mouse sobre esse título</h2>          <button onclick="eventClique()">Clique aqui</button>          <script src="main.js"></script>      </body>  </html> |

Nesse código, inserimos um elemento heading <h2> com o evento onmouseover, que chama a função mouseEmCima(). Vamos implementar a função mouseEmCima() no arquivo **main.js**.

1. Volte ao arquivo main.js e insira o código abaixo.

|  |  |
| --- | --- |
| 44 | const mouseEmCima = () => {      console.log('Mouse está em cima do título');  }; |
| 45 |
| 46 |

1. Passe o mouse por cima do título e veja a mensagem impressa no console.

Graphical user interface, text, application

Description automatically generated