



Desenvolvimento Web Completo 2022


Por Jorge Sant Ana, Programador
Jamilton Damasceno, Analista de Sistema e Professor



Seção 4: JavaScript





Aula 20: Funções Anônimas e a Técnica de Wrapper

- 
- Iremos aprender a como podemos criar funções anônimas e como podemos combinar essas funções com a técnica de wrapper.
 - Como o nome sugere, funções anônimas são funções que não possuem nome. A sintaxe e o funcionamento de uma função anônima é idêntica a uma função comum, porém, a única diferença é realmente a ausência do nome da função.

Exemplo


```
//Função Anônima  
function() {  
    document.write("Executou");  
}
```

- 
- Porém, por ser uma função anônima, como que podemos chamar essa função em algum momento do script?
 - O JavaScript tem um recurso bem interessante, que é a capacidade de associar a uma variável, não apenas strings, valores booleanos ou valores numéricos, mas também funções.

- 
- Então, podemos, por exemplo, criar uma variável qualquer, atribuir a essa variável, a função anônima e exibir o que foi feito nessa função, de tal modo, que a chamada dessa função pode ser feita a partir da utilização da variável em conjunto com abre e fecha parênteses.


Exemplo


```
//Função Anônima  
var teste = function() {  
    document.write("Executou");  
}  
  
teste();
```

Funções Anônimas

Executou

- 
- Então, perceba que a nossa variável passou a aguardar uma referência para a função, ou seja, ela está embrulhando a nossa função de tal modo que a função é uma função anônima, internamente ela faz alguma coisa.
 - Então, assim como como uma função comum, podemos receber aqui algum parâmetro. E se a função espera receber uma função, precisamos encaminhar esse parâmetro para a função.

- 
- Embora essa construção possa parecer pouco convencional, já que é uma função comum resolveria do mesmo jeito, funções anônimas e wrapper, ou seja, a técnica de embrulhar funções são recursos constantemente utilizados dentro do JavaScript.
 - Isso porque o JavaScript que incorpora um conceito muito forte de utilização de funções de callback, que nada mais é do que a passagem de funções como parâmetro para outras funções.