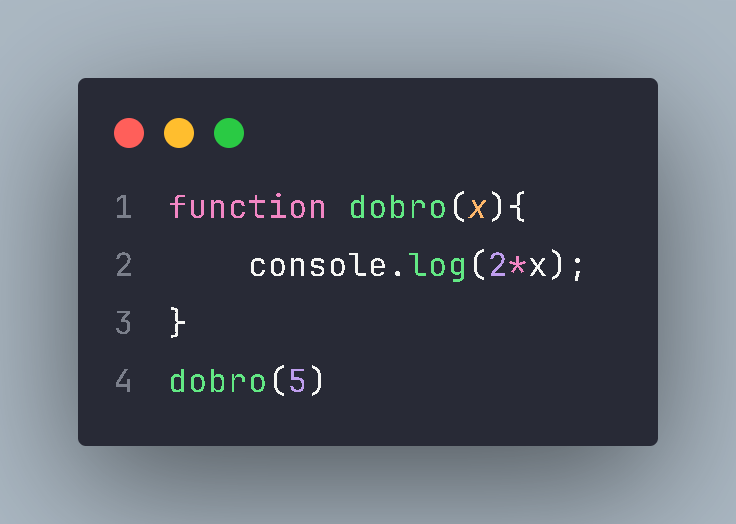
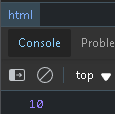
**Modulo 18 – JS Avançado**

**AULA 4 - Arrow Functions**

Uma função do tipo arrow tem como a única finalidade encurtar a quantidade de linhas que é necessário para escrever um mesmo código, vamos verificar um exemplo onde declaramos um função e ao mostrá-la em tela execute uma operação de multiplicação.



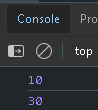
Podemos observar aqui, que o parâmetro “x” levara o valor de 5 e assim que a função receber esse valor irá por meio do “console.log” realizar uma operação e exibir o resultado “10” no console do navegador.



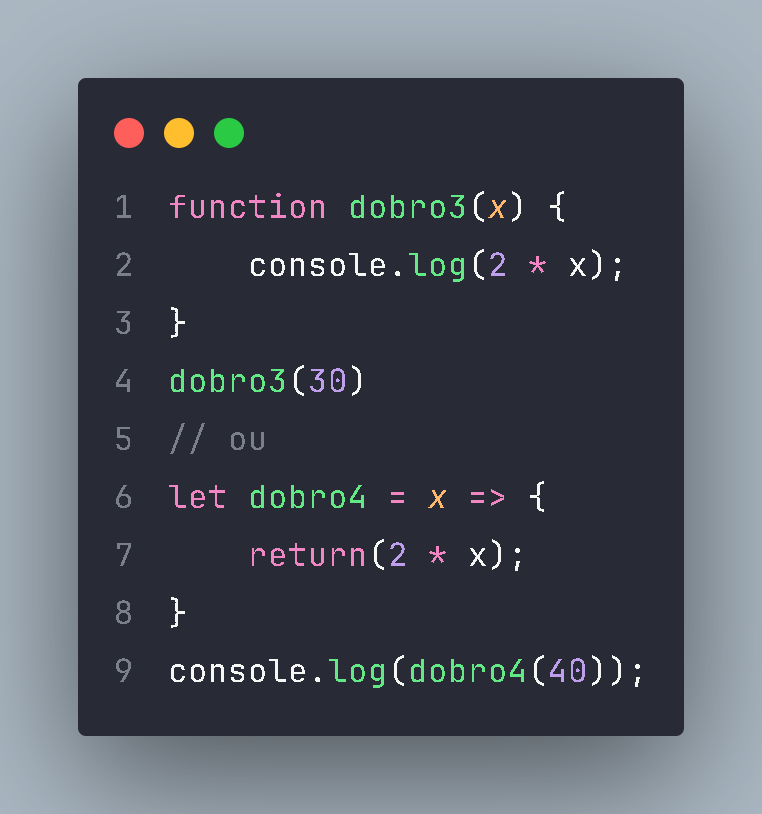
Temos outra forma de declarar, passando a função para uma variável.



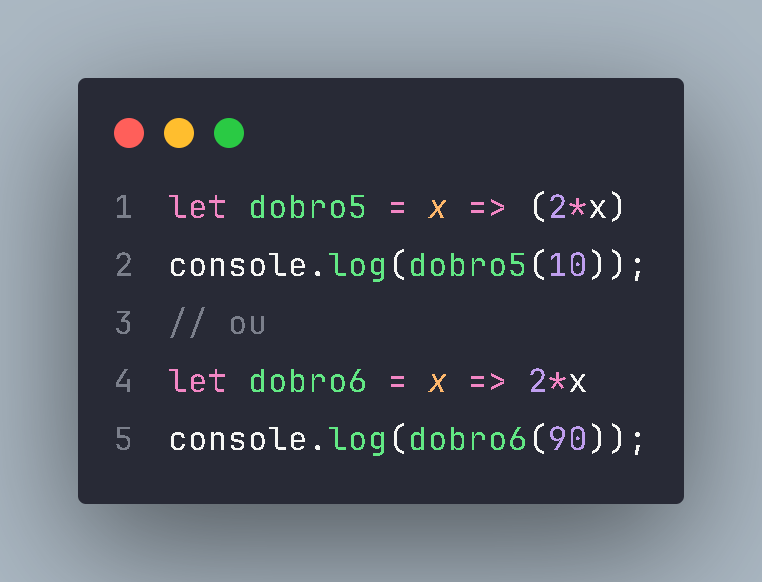
Vamos sempre alterar o valor para percebermos que o resultado em tela irá ser alterado também, nesse caso deixamos as duas funções no mesmo arquivo para notarmos se houve mudança no console.



Agora usando uma arrow function, vamos tentar perceber a diferença entre um function normal.

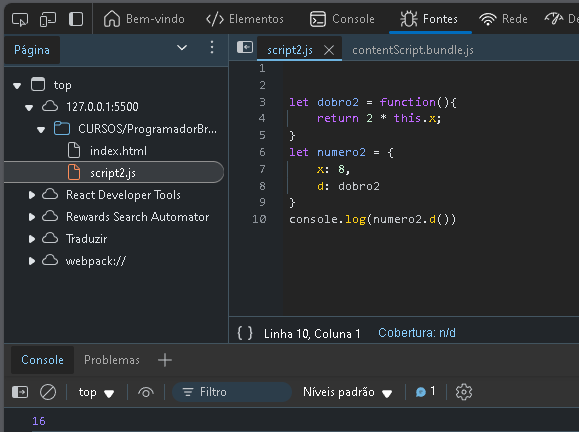
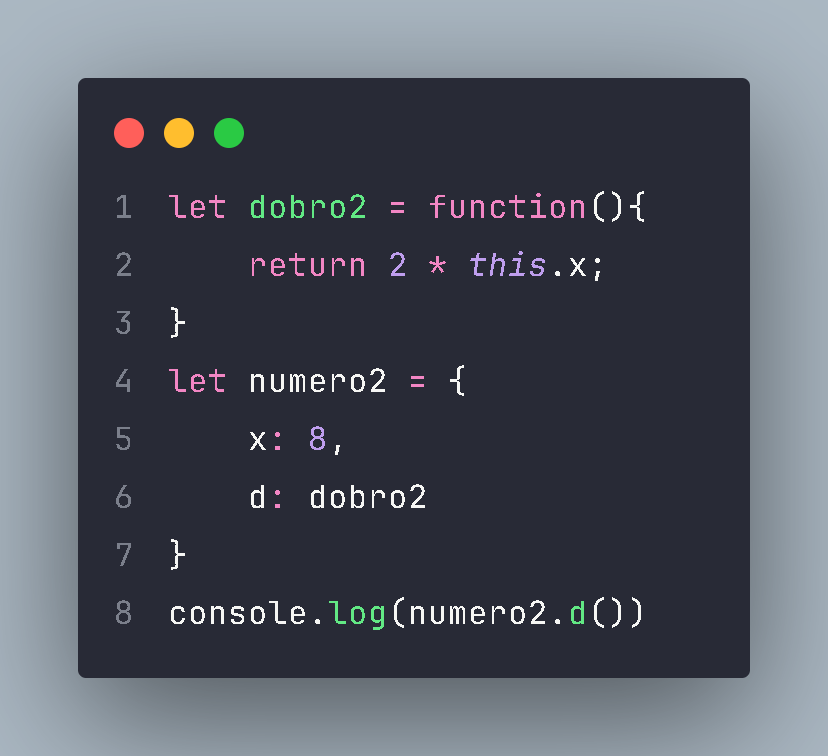


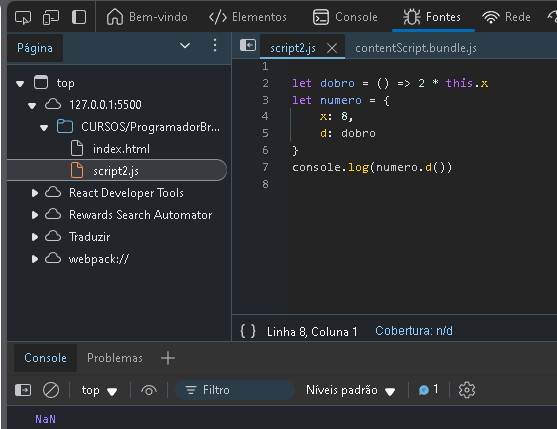
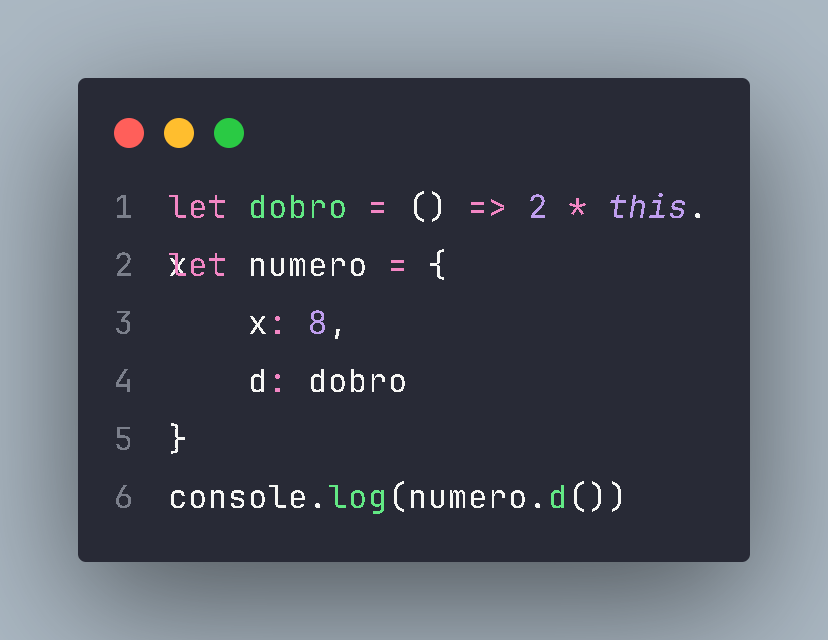
Notamos que não há mais o uso da palavra reservada “*function*” para declarar a função em si, e para demostrar como o código pode mudar, nesse caso colocamos o “*console.log*” do lado de fora da função e alteramos internamente com um “*return*”.



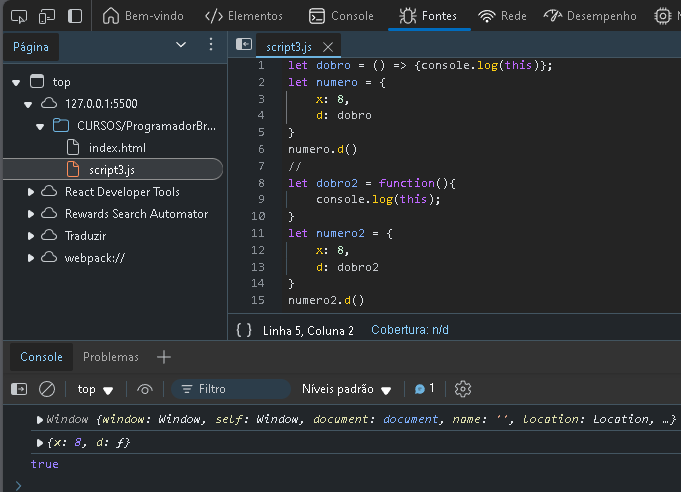
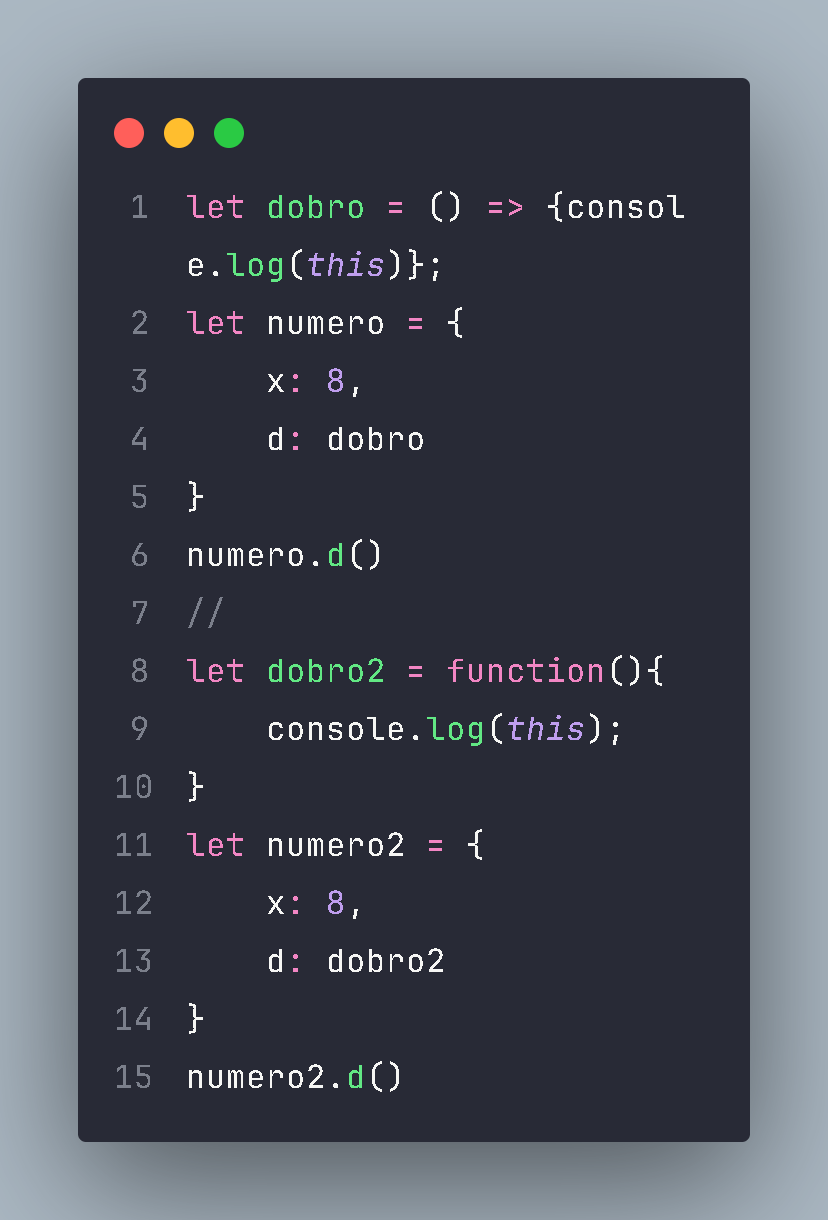
Agora podemos perceber o quão pequeno fica nosso código quando colocamos uma arrow function em ação.

Não podemos deixar de pontuar o ponto negativo, normalmente quase nunca iremos trabalhar com isso, mas o atributo “this” não funciona com uma arrow function. Veja no exemplo abaixo:





Podemos entender aqui então que o “*this*” perde seu contexto quando utilizamos ele em uma arrow function, podemos mandar imprimir o “*this*” nas duas funções e iriamos perceber que na função “normal” teríamos o retorno do objeto no console, porém ao fazer com o “*this*” o retorno no console seria o objeto global “*windown*”.



**AULA 5 - Callback**