**FUNÇÃO ANÔNIMA:**

- Uma variável pode guardar uma função, pois não tem nome. seu relacionamento é com uma variável.

const soma = function (a, b){

return a + b;

}

soma(1, 2);

**## FUNÇÃO AUTOINVOCÁVEL ##**

(

function(){

let name = "Jorge"

return name;

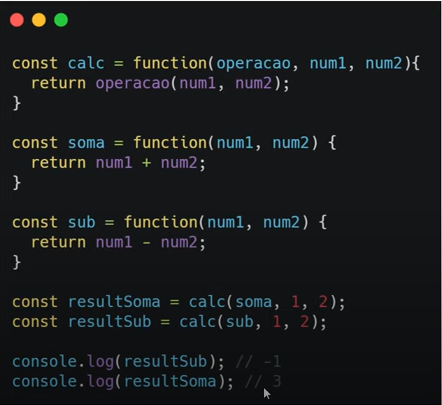
}

)();

Da mesma forma podemos passar parâmetros para a função autoinvocável.

**## CALLBACKS:** função passada como argumento para outra; com ela temos maior controle da ordem de chamadas.

]]



Objeto “arguments”: um array com todos os parametros passados quando a função foi invocada.

Array - Spread: forma de lidar com elementos separados de uma array. As partes de um array se torna um elemento independente. Como usar:

Function(x, y, z){

Return x + y + z;

}

Const numbers = [1, 2, 3];

Console.log(sum(...numbers));

Rest: combina os argumentos em um array. O elemtno independente se torna um array.

OBJETOS:

Destructuring:entre chaves podemos filtrar somente os dados que nos interessam em um objeto;

Const user = {

Id = 42;

displayName: ‘HH’;

fullName: {

Fisrt: ‘j’;

}

Function userID({id}){

Return id;

}

For...in: para propriedades enumeráveis;

For...of: loop entre estruturas iteráveis como array e string.

THIS: referência pro contexto que está. Retorna a propriedade do objeto atual.

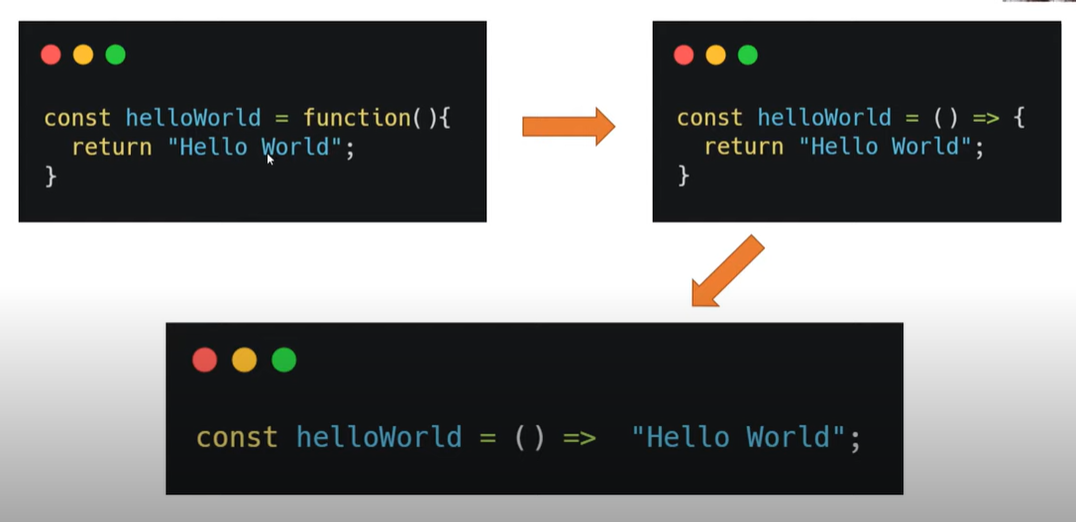


Apply: na função eu digo qual eu desejo, e no apply passo o ojbeto que desejo a informação.

É passado como array.

Bind: clona a estrutura da função onde foi chamado e aplica o valor do ojbeto passado como parâmetro. Clona a função e executa na variável de referência.

FUNÇÕES ARRAYS/arrow function:



Ela está sempre alinhada a uma constante. Caso só tenha uma linha pode dispensar as chaves e o return.

Ela não faz HOISTING, que é chamar a variável antes de declará-la. Ela não faz porque está sempre associada a uma constante que sempre precisa ser inicializada.

O this será sempre global, ou seja, modificar seu valor não irá funcionar. Nem o ojbeto arguments irá existir nem construtor será utilizado.