Tipos de funções

Existem 3 formas de utilizar funções. Temos a **Function Declaration, Function Expression e Arrow Functions.**

Function Declaration: A declaração de função é declarada como uma instrução separada, no fluxo de código principal.

```
// Declaração da função
function sum(a, b) {
return a + b;
}
const resultado = soma(3, 7); // chamando a função
```

Tipos de funções

Function Expression: são criadas dentro de outra expressão ou instrução, como uma declaração de variável. A função abaixo é tratada como um valor que é atribuído a uma variável — essa variável pode então ser chamada como uma função.

```
// Function Expression
let sum = function(a, b) {
return a + b;
};
const result = sum(1, 3) // chamando a função
```

Tipos de funções

Arrow Functions: oferece uma sintaxe mais objetiva.

seja soma = (a, b) => a + b; // observe que não requer uma palavra-chave "return", porque está implícita.

Funções de Primeira Classe

Entenda-se que uma linguagem de programação tem **Function First-class** quando as funções são tratadas como qualquer outra variável. Por exemplo, numa linguagem desse tipo, a função pode ser passada como argumento pra outras funções, ser retornada por outra função e pode ser atribuída como um valor a uma variável.

- Atribuir uma função a uma variável
- Passar uma função como um argumento
- Retornar uma função

Atribuir uma função

```
const foo = function() {
    console.log("foobar");
}
// Chamar a função usando a variável
foo();
```

Passar uma função

```
function sayHello() {
    return "Hello, ";
}
function greeting(helloMessage, name) {
  console.log(helloMessage() + name);
}
// Passar `sayHello` como um argumento pra função `greeting`
greeting(sayHello, "JavaScript!");
```

Retornar uma função

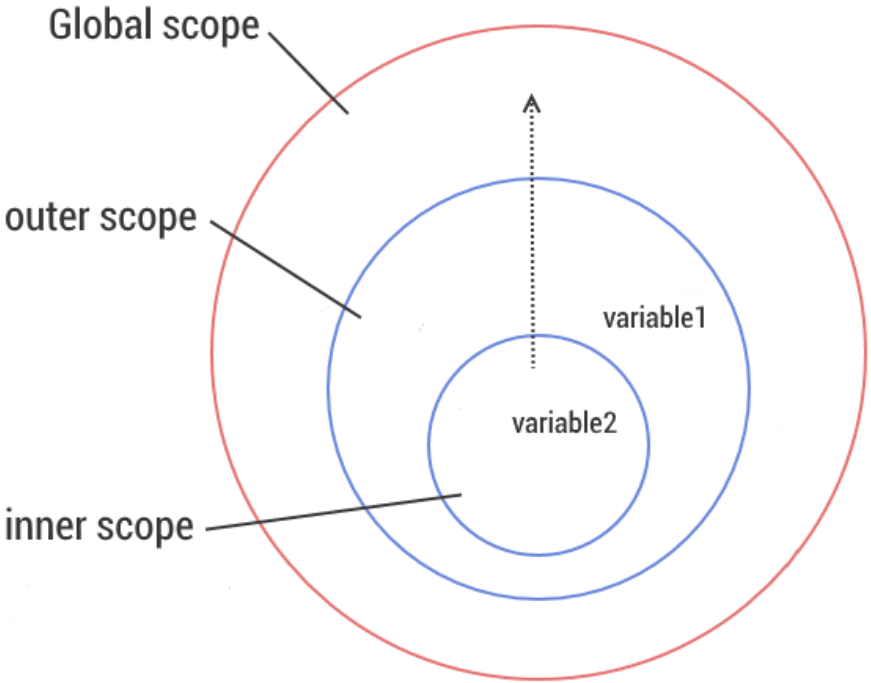
```
function sayHello() {
    return function() {
        console.log("Hello!");
    }
}
```

Escopo & hoisting

O escopo em JavaScript se refere à **visibilidade de variáveis e funções dentro de um programa**. Quando uma variável ou função é declarada em um determinado escopo, ela só pode ser acessada dentro daquele escopo ou de escopos internos a ele. Podeos determinar que no JavaScript tem dois tipos de escopo, o global e local.

Hoisting é o termo que explica essa situação, em português ele **significa "içamento", ou "elevação"** e foi citado pela primeira vez no ECMAScript® 2015 Language Specification. O Hoisting permite que você execute funções antes das suas declarações.

Escopo & hoisting



Funções Imediatas

Funções imediatas são funções que são automaticamente executadas imediatamente após sua declaração, sem necessidade de chamadas.

```
(function() {
  // Everything here will be self executed
  console.log('foo');
})();

// foo
```

```
var person = (function() {
  return {
    sayHello: function(name) {
      console.log('Hello' + name);
    }
  }
})();

person.sayHello('Natacsha'); // Olá Natacsha
```

Closures

é a combinação de uma função agrupada (incluída) com referências ao seu estado circundante (o **ambiente léxico**).

Em outras palavras, um encerramento fornece acesso ao escopo de uma função externa a partir de uma função interna.

Em JavaScript, os encerramentos são criados toda vez que uma função é criada, no momento da criação da função.

Closures

```
function init() {
  var name = "Mozilla"; // name is a local variable created by
init
  function displayName() {
    // displayName() is the inner function, that forms the
closure
    console.log(name); // use variable declared in the parent
function
  }
  displayName();
}
init();
```

Obrigada!

Perguntas?

E-mail: natacsha@icomp.ufam.edu.br