

# Desarrollo Backend con NodeJS

Funciones y métodos

# Funciones

## Funciones

En Javascript una función es un procedimiento, **un conjunto de sentencias que realizan una tarea o calculan un valor**. Para usar una función se debe definir en algún lugar del ámbito desde el cual se desea llamarla.

# Declaración de una función

Para declarar funciones vamos a usar la siguiente sintaxis:



```
function nombreDeLaFunción(param1, param2, param3, param100) {  
    // instrucciones  
}
```



```
function sayHi(name) {  
    console.log('Hola ' + name)  
}
```

## return

La sentencia `return` finaliza la ejecución de la función y especifica un valor para ser **devuelto** a quien llama a la función.

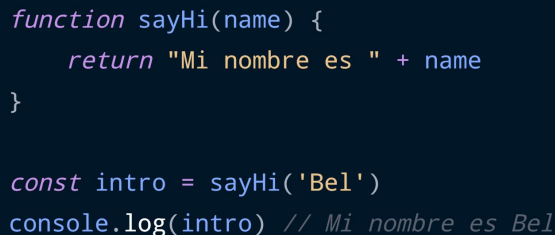


```
function sayHi(nombre) {  
    return "Mi nombre es " + name  
}
```

## Ejecución de funciones

Para ejecutar una función que anteriormente hayamos creado, basta con llamarla utilizando su nombre y pasando los valor por parámetro.

El valor de la función puede ser asignado a una variable para luego utilizarlo.



```
function sayHi(name) {  
    return "Mi nombre es " + name  
}  
  
const intro = sayHi('Bel')  
console.log(intro) // Mi nombre es Bel
```

# Scope

Las variables definidas dentro de una función no pueden ser accedidas desde ningún lugar fuera de la función, ya que la variable está definida sólo en el ámbito de la función. Sin embargo, **una función puede acceder a todas las variables y funciones definidas dentro del ámbito en el cual está definida**

```
function display(name) {  
  const intro = "Mi nombre es " + name;  
  
  function addToName() {  
    return intro + "It"  
  }  
  
  return addToName();  
}  
  
const greeting = sayHi('Course')  
console.log(greeting) // Mi nombre es CourseIt
```

```
const name = "Course";  
  
function display() {  
  const intro = "Mi nombre es " + name;  
  
  function addToName() {  
    return intro + "It";  
  }  
  
  return addToName();  
}  
  
const greeting = display("Course");  
console.log(greeting); // Mi nombre es CourseIt
```

## Expresiones de una función

Las funciones pueden también ser creadas por una expresión de función, la cual puede **ser anónima**, por ejemplo, tomando el caso anterior de la función de **display**, nos quedaría así:



```
const display = function(name) {  
  return "Mi nombre es " + name  
}
```

```
const name = display("CourseIt")  
console.log(name)
```



```
const display = () => {  
  return "Mi nombre es " + name  
}
```


```
// const display = () => return "Mi nombre es " + name
```

```
const name = display("CourseIt")  
console.log(name)
```



# Closures

Los closures o clausuras son funciones que manejan variables independientes. En otras palabras, la función definida en el closure "recuerda" el ámbito en el que se ha creado.



```
function makeCouples(person) {  
  return function(partner) {  
    return person + " y " + partner;  
  };  
}  
  
const thelma = makeCouples('Thelma');  
const bel = makeCouples('Bel');  
  
console.log(thelma('Louise')); // Thelma y Louise  
console.log(bel('Juani')); // Bel Y Juani
```

# Variables privadas con closures

Vamos a crear un closure donde la variable value sea privada y solo pueda ser accedida mediante métodos públicos

```
function counter() {  
  let value = 1;  
  
  return {  
    incrementValue: function() {  
      value++  
      // o value = value + 1  
    },  
    decreaseValue: function() {  
      value--  
    },  
    value: function() {  
      return value  
    }  
  };  
}  
  
const newCounter = new Counter();  
newCounter.value() // 1  
newCounter.incrementValue();  
newCounter.value() // 2  
newCounter.decrementValue();  
newCounter.value() // 1
```