

# Los closures

**DigitalHouse** >  
Coding School



**Certified Tech  
Developer**  
The Ultimate Degree



Los closures son **funciones anidadas**,  
es decir, funciones que viven **dentro**  
**de otras funciones**.



# Estructura de un closure

Un closure vive dentro de una **función contenedora** y existirá **únicamente** cuando esa función sea ejecutada.

Una vez terminada la ejecución, el closure **dejará de existir**, logrando con esto optimizar el rendimiento general al dejar disponible su espacio en memoria para otras peticiones.

```
{  
  function contenedora(){  
    function interna(){  
      //código a ejecutar  
    }  
    //código a ejecutar  
  }  
}
```

“

Un closure tiene acceso a **todos** los **parámetros** y **variables** que la función contenedora esté **recibiendo** o **declarando** sin necesidad de tener que pasárselos como parámetro.

”



## {código}

```
function saludoPersonalizado(nombre){  
  let saludoGenerico = 'Hola';  
  function saludar(){  
    return saludoGenerico + ' ' + nombre;  
  }  
  return saludar();  
}
```

## {código}

```
function saludoPersonalizado(nombre){  
  let saludoGenerico = 'Hola';  
  function saludar(){  
    return saludoGenerico + ' ' + nombre;  
  }  
  return saludar();  
}
```

Definimos la función **saludoPersonalizado** la cual recibe un parámetro.

## {código}

```
function saludoPersonalizado(nombre){  
  let saludoGenerico = 'Hola';  
  function saludar(){  
    return saludoGenerico + ' ' + nombre;  
  }  
  return saludar();  
}
```

Definimos la variable local, **saludoGenerico** y le asignamos el string *'Hola'*.

# {código}

```
function saludoPersonalizado(nombre){  
  let saludoGenerico = 'Hola';  
  function saludar(){  
    return saludoGenerico + ' ' + nombre;  
  }  
  return saludar();  
}
```

Definimos nuestro  
closure: **saludar**.



## {código}

```
function saludoPersonalizado(nombre){  
  let saludoGenerico = 'Hola';  
  function saludar(){  
    return saludoGenerico + ' ' + nombre;  
  }  
  return saludar();  
}
```

El mismo podrá hacer uso del parámetro **nombre** que recibe su función contenedora sin la necesidad de pasárselo como parámetro.

También podrá hacer uso de la variable local **saludoGenerico**.

## {código}

```
function saludoPersonalizado(nombre){  
  let saludoGenerico = 'Hola';  
  function saludar(){  
    return saludoGenerico + ' ' + nombre;  
  }  
  return saludar();  
}
```

Finalmente, hacemos un **return** de la ejecución de nuestro closure.

El llamado a su ejecución se debe dar siempre **dentro** de su función contenedora.

DigitalHouse>  
Coding School