Funciones

DEV.F:
DESARROLLAMOS(PERSONAS);

FUNCIONES

Reciben elementos de entrada y entregan un valor de salida

¿Para qué?

Envolver bloques de código que queramos repetir resulta ser muy útil

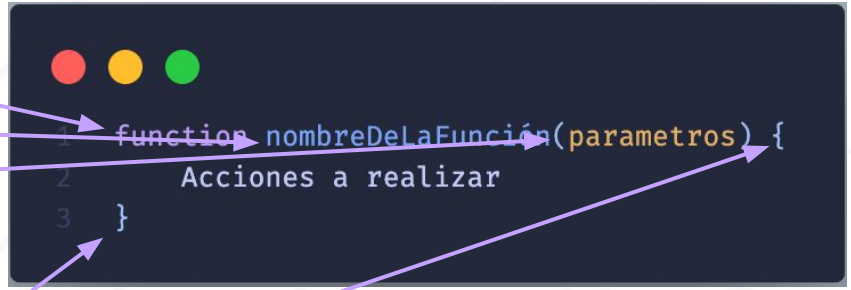
Nos ayudan a dar estructura a nuestro código y mantenerlo DRY (Do Not Repeat Yourself)

Introducimos nuevo vocabulario.

DEV.F

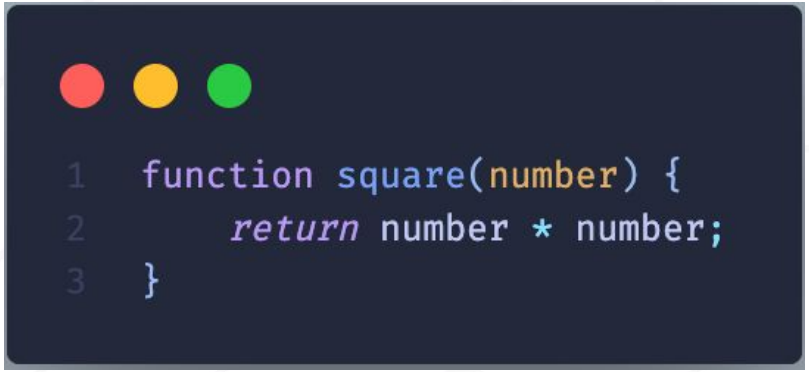
Consta de

- La palabra reservada `function`
- El nombre de la función
- Una lista de parámetros
  - Separados por comas
  - NO necesarios
- Las declaraciones que definen la función (body) van entre llaves { }



Function Declaration

Una función declarada de esta manera es una Function Declaration (en forma de declaración/sentencia)



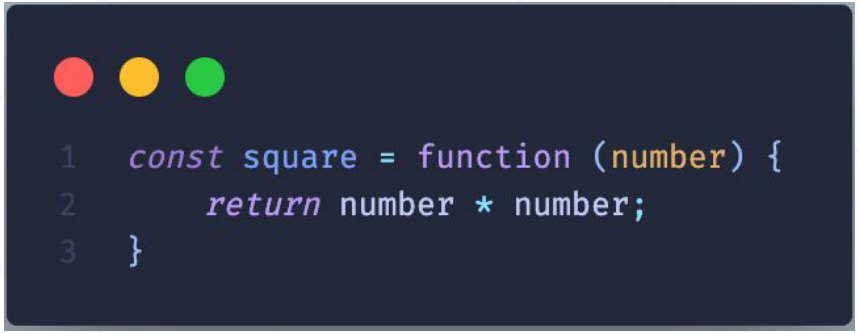
```
1  function square(number) {
2      return number * number;
3  }
```

Funciones anónimas

Todo lo que regrese un valor se puede guardar en una variable.

Las funciones anónimas no tienen un nombre y se guardan en variables.

Estas funciones son declaradas como expresiones (Function Expression)



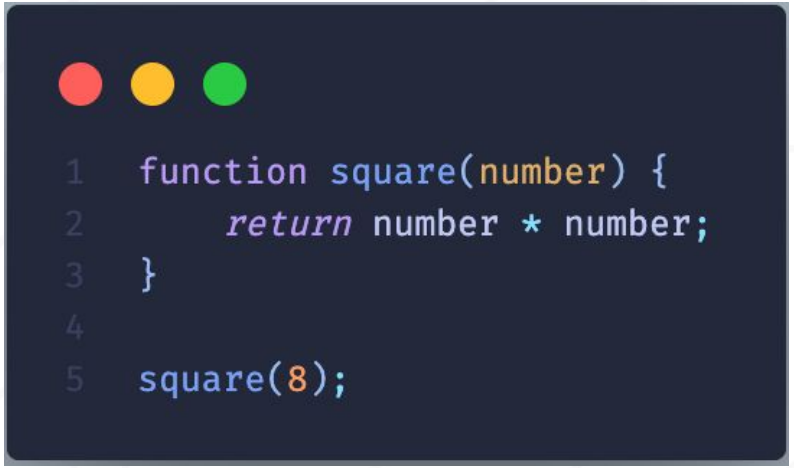
DEV.F

¿Cómo las invoco?

Las funciones se mandan llamar, invocar, o correr. *(calling, running, invoking)*

Pero ¿cómo?

Usamos su nombre seguido de paréntesis en donde van los argumentos.



```
function square(number) {
    return number * number;
}

square(8);
```

Demostración: ¿Cuántos años te faltan para el retiro?

Vamos a hacer un programa que nos indique cuántos años nos faltan para retirarnos. Supongamos que la edad de retiro es de 65 años.

Funception. Funciones dentro de funciones

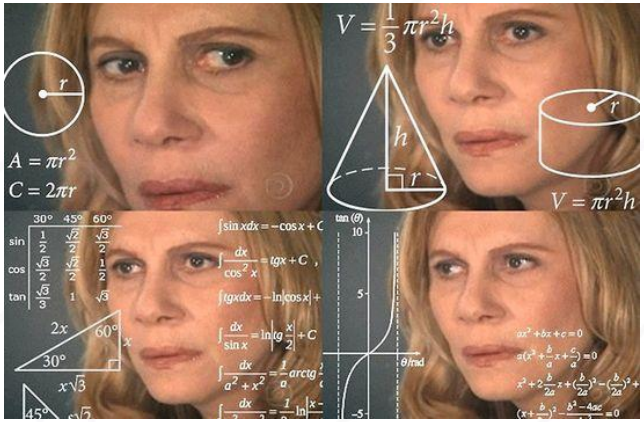
¿Podemos tener funciones dentro de funciones?

Por supuesto!

Continuemos con el ejemplo del retiro

Ejercicio en clase: Hola!

Crea una función que acepte un parámetro (string) que devuelva un saludo “Hola, NOMBRE” y lo imprima en la consola.



05:00

DEV.F

Actividad

Ejercicio Funciones

Challenge:

¿Te acuerdas de los partidos entre los Patriots y Broncos?

Requisitos:
1. Una función que calcule los promedios entre las 3 puntuaciones.
2. Guardar los promedios en las variables: “averagePatriots” y “averageBroncos”
3. Una función que tome los dos promedios, compare ambos para obtener al ganador y guarde quién es el ganador.
4. Imprimir quién es el ganador y los promedios usando los datos 1 y datos 2
   “Patriots ganan 🏆 (20 .vs. 17)” o “Broncos ganan 🏆 (18 .vs. 12)”

Datos1 :

Broncos: 15, 7, 10
Patriots: 5, 12, 24

Datos 2:

Broncos: 11, 18, 13
Patriots: 10, 17, 7

DEV.F

Scope
(Alcance)

DEV.F::
DESARROLLAMOS(PERSONAS);

Glosario

- **Scoping:** Cómo se organizan y accedemos a las variables. ¿Dónde viven? ¿Desde dónde puedo acceder a ellas?
- **Lexical Scoping:** Cuando el scoping está controlado por la ubicación de donde se declaran las funciones
- **Scope:** El Entorno en donde se declara una variable en específico.
    - Scope Global
    - Scope de Función
    - Scope de Bloque
- **Scope de una variable:** Región del código en donde se puede acceder a esa variable

Los 3 tipos de Scope

Scope Global

```
const firstName = 'Yaxche';
const job = 'sensei';
const year = 1993;
```

Fuera de cualquier función o bloque { }

Se puede acceder a las variables declaradas en el scope global desde donde queramos

Scope de función

```
function ageCalculator(birthYear) {
    const now = 2023;
    const age = now - birthYear;
    return age;
}
```

Estas variables solo son accesibles **dentro de la función**

También llamado scope local

Scope de bloque (ES06)

```
if (year ⩾ 1981 && year ⩽ 1986) {
    const millennial = true;
    const food = 'Aguacate 🥑';
}
```

Solo se pueden acceder desde **dentro del bloque**

Solo aplica a variables declaradas con `let` y `const`.

`var` puede ser consultada fuera del bloque, se 'sale' al scope de función más cercano

Cómo se ve el scope?

```
const firstName = 'Yaxche';

function first() {
    const age = 30;

    if (age ⩾ 30) {
        const decade = 3;
        var millenial = true;
    }

    function second() {
        const job = 'sensei';

        console.log
            (`${firstName} tiene ${age} y es ${job}`);
            // Yaxche tiene 30 años y es sensei
    }

    second();
}

first();
```