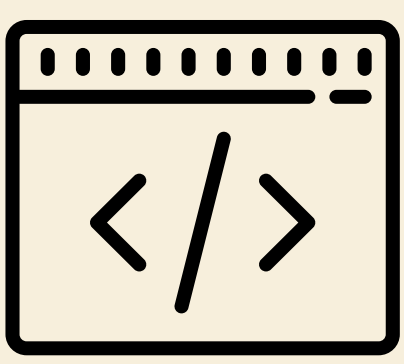
VARIABLES Y FUNCIONES EN JAVASCRIPT

GLOBAL SCOPE

Se dice que una variable se encuentra en el Scope global cuando está declarada fuera de una función o de un bloque.

```
var alfajor = "postre";
```



LOCAL SCOPE

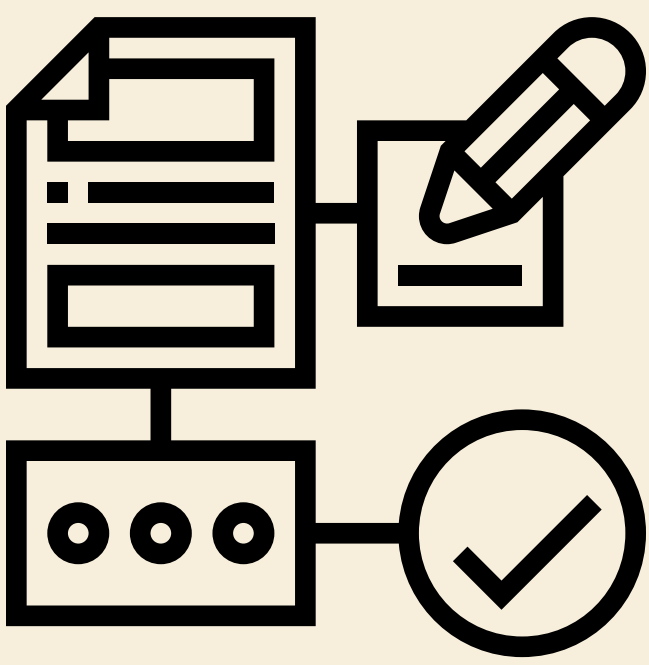
Las variables que definimos dentro de una función son variables locales, y solamente van a existir dentro del proceso correspondiente.
-si intentamos accederlas fuera de ella, dichas variables no van a estar definidas

```
function myFunction() {
 var alfajor= "Postre";}
```

Block Scope

Para ello, contamos con los keyword let y const los cuales nos permiten tener un scope de bloque, esto quiere decir que las variables solo van a vivir dentro del bloque de código correspondiente

```
      if (true) {
 // la variable nombre es global por el uso de la keyword 'var'
       var nombre = 'Juan';
 // preferencias se encuentra en el scope local por el uso de la keyword 'let'
       let preferencias = 'Codear';
 // skills también es una scope local por el uso de la keyword 'const'
       const skills = 'Java';
}
      console.log(nombre);
      // logs 'Juan'
      console.log(preferencias);
      // Uncaught ReferenceError:
      preferencias is not defined
      console.log(skills);
      // Uncaught ReferenceError: skills is
      not defined
```



Emmanuel Yepes Conde
Jonathan Betancur Espinosa
Ficha 2255102

Funciones y formas de declararlas

Una función en JavaScript es similar a un procedimiento (un conjunto de instrucciones que realiza una tarea o calcula un valor), pero para que un procedimiento califique como función, debe tomar alguna entrada y devolver una salida donde hay alguna relación entre la entrada y la salida

Función declarada

Este tipo de función se creará con la palabra reservada function, seguido obligatoriamente por un nombre, que identificará a nuestra función, una lista de parámetros entre paréntesis, y el símbolo de las llaves {}. Qué será el que delimite el contenido de nuestro conjunto de sentencias.

```
function hola(nombre){
console.log(`Hola ${nombre}.`)
}

hola('Victor');    // => Hola Victor
```

Si queremos que nuestra función devuelva algún valor debemos utilizar la instrucción return en nuestro bloque

este tipo de funciones son compatibles con el hoisting. El hoisting es una característica de Javascript por la cual las definiciones se ejecutan al principio de la ejecución del código

Función de expresión

Es similar a las de declaración, La única diferencia es que la definición de nuestra nueva función no comienza por instrucción function, no es compatible con hoisting, y el nombre de la función es opcional.

Ejemplo: función que sumará dos al argumento que enviemos a la función.

```
const SUMADOS = function sumaDos(valor) {
return valor + 2;
}

console.log(SUMADOS(5));    // => 7
```

Funciones arrow:

Es una forma más corta que se creó con el ES6 para escribir funciones, Primero definiremos la lista de parámetros, en caso de ser necesario, entre paréntesis seguido del símbolo => y las {} para indicar las instrucciones que se van a realizar.

Antes:

```
hello = function() {
return "Hello
World!";
}
```

Ahora:

```
hello = () => {
return "Hello World!";
}
```

Si la función tiene solo una declaración y la declaración devuelve un valor, puede eliminar los corchetes y la palabra clave return:

```
hello = () => "Hello World!";
```

Si se necesitan agregar parámetros, se pasan entre paréntesis

Ejemplo:

```
<script>
var hello;

hello = (val) => "Hello " + val;

document.getElementById("demo").innerHTML = hello("Universe!");
</script>
```

Emmanuel Yepes Conde
Jonathan Betancur Espinosa
Ficha 2255102