	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

TECNÓLOGO EN ANÁLISIS Y DESARROLLO DE SOFTWARE

Ficha: 2900177

Aprendiz: Brayan Santiago Guerrero Mendez

Instructor: Andrés Moreno

Neiva-Huila

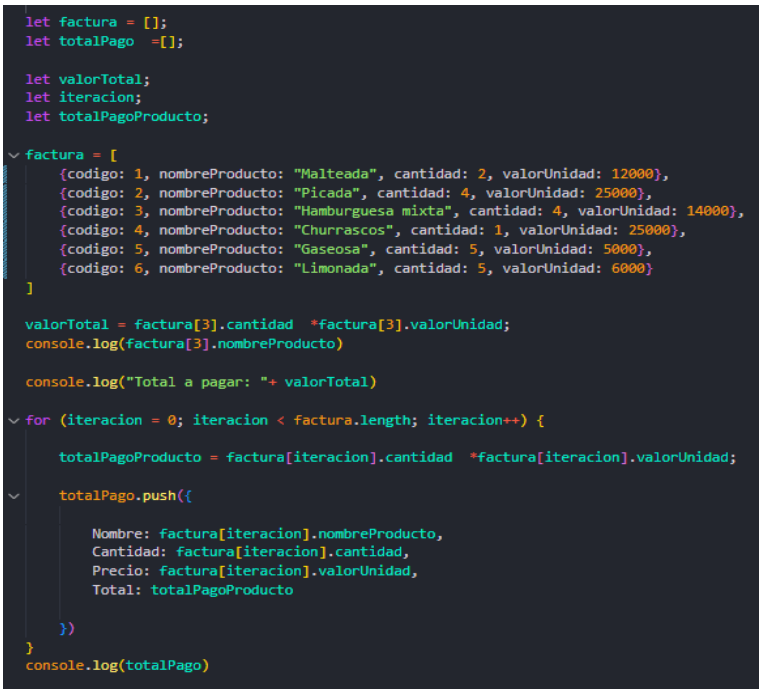
	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

Contenido

	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

Nombre del arreglo: factura	**Versión: 1.0**
Descripción: EL arreglo factura almacena detalles de los productos como lo son el codigo, el nombre, la cantidad y el valor, lo cual nos ayuda a calcular el precio total que será almacenado en totalPago	
valorTotal	Tipo de variable: Float
iteracion	Tipo de variable: Int
totalPagoProducto	Tipo de variable: Float

Código:



```
let factura = [];
let totalPago  =[];

let valorTotal;
let iteracion;
let totalPagoProducto;

factura = [
    {codigo: 1, nombreProducto: "Malteada", cantidad: 2, valorUnidad: 12000},
    {codigo: 2, nombreProducto: "Picada", cantidad: 4, valorUnidad: 25000},
    {codigo: 3, nombreProducto: "Hamburguesa mixta", cantidad: 4, valorUnidad: 14000},
    {codigo: 4, nombreProducto: "Churrascos", cantidad: 1, valorUnidad: 25000},
    {codigo: 5, nombreProducto: "Gaseosa", cantidad: 5, valorUnidad: 5000},
    {codigo: 6, nombreProducto: "Limonada", cantidad: 5, valorUnidad: 6000}
]

valorTotal = factura[3].cantidad  *factura[3].valorUnidad;
console.log(factura[3].nombreProducto)

console.log("Total a pagar: "+ valorTotal)

for (iteracion = 0; iteracion < factura.length; iteracion++) {

    totalPagoProducto = factura[iteracion].cantidad  *factura[iteracion].valorUnidad;

    totalPago.push({

        Nombre: factura[iteracion].nombreProducto,
        Cantidad: factura[iteracion].cantidad,
        Precio: factura[iteracion].valorUnidad,
        Total: totalPagoProducto

    })
}
console.log(totalPago)
```

	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

Churrascos

Total a pagar: 25000

(index)	Nombre	Cantidad	Precio	Total
0	'Malteada'	2	12000	24000
1	'Picada'	4	25000	100000
2	'Hamburguesa mixta'	4	14000	56000
3	'Churrascos'	1	25000	25000
4	'Gaseosa'	5	5000	25000
5	'Limonada'	5	6000	30000

▸ Array(6)

Nombre del arreglo: nomina	**Versión: 2.0**

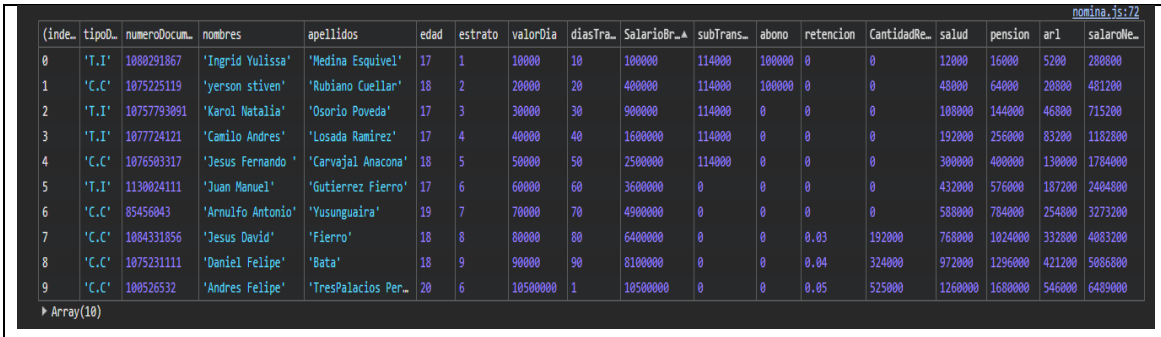
Descripción:
El arreglo nomina contiene información detallada sobre varios personas, incluyendo tipo y número de documento, nombres, apellidos, edad, estrato, valor diario, y días trabajados.

Código:

```
let nomina = [];
let pagoNomina = [];

nomina = [
  {tipoDocumento: "T.I", documento: 1080291867, nombres: "Ingrid Yulissa", apellidos: "Medina Esquivel", edad: 17, estrato: 1, valorDia: 10000, diasTrabajados: 10 },
  { tipoDocumento: "C.C", documento: 1075225119, nombres: "yerson stiven", apellidos: "Rubiano Cuellar", edad: 18, estrato: 2, valorDia: 20000, diasTrabajados: 20 },
  { tipoDocumento: "T.I", documento: 10757793091, nombres: "Karol Natalia", apellidos: "Osorio Poveda", edad: 17, estrato: 3, valorDia: 30000, diasTrabajados: 30 },
  { tipoDocumento: "T.I", documento: 1077724121, nombres: "Camilo Andres", apellidos: "Losada Ramirez", edad: 17, estrato: 4, valorDia: 40000, diasTrabajados: 40 },
  { tipoDocumento: "C.C", documento: 1076503317, nombres: "Jesus Fernando ", apellidos: "Carvajal Anacona", edad: 18, estrato: 5, valorDia: 50000, diasTrabajados: 50 },
  { tipoDocumento: "T.I", documento: 1130024111, nombres: "Juan Manuel", apellidos: "Gutierrez Fierro", edad: 17, estrato: 6, valorDia: 60000, diasTrabajados: 60},
  { tipoDocumento: "C.C", documento: 85456043, nombres: "Arnulfo Antonio", apellidos: "Yusunguaira", edad: 19, estrato: 7, valorDia: 70000, diasTrabajados: 70 },
  { tipoDocumento: "C.C", documento: 1084331856, nombres: "Jesus David", apellidos: "Fierro", edad: 18, estrato: 8, valorDia: 80000, diasTrabajados: 80 },
  { tipoDocumento: "C.C", documento: 1075231111, nombres: "Daniel Felipe", apellidos: "Bata", edad: 18, estrato: 9, valorDia: 90000, diasTrabajados: 90 },
  { tipoDocumento: "C.C", documento: 100526532, nombres: "Andres Felipe", apellidos: "TresPalacios Perez", edad: 20, estrato: 6, valorDia: 10500000, diasTrabajados: 1 }
];
```

	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS



nomina.js:72

(inde…	tipoD…	numeroDocum…	nombres	apellidos	edad	estrato	valorDia	diasTra…	SalarioBr…A	subTrans…	abono	retencion	CantidadRe…	salud	pension	arl	salaroNe…
0	'T.I'	1080291867	'Ingrid Yulissa'	'Medina Esquivel'	17	1	10000	10	100000	114000	100000	0	0	12000	16000	5200	280800
1	'C.C'	1075225119	'yerson stiven'	'Rubiano Cuellar'	18	2	20000	20	400000	114000	100000	0	0	48000	64000	20800	481200
2	'T.I'	10757793091	'Karol Natalia'	'Osorio Poveda'	17	3	30000	30	900000	114000	0	0	0	108000	144000	46800	715200
3	'T.I'	1077724121	'Camilo Andres'	'Losada Ramirez'	17	4	40000	40	1600000	114000	0	0	0	192000	256000	83200	1182800
4	'C.C'	1076503317	'Jesus Fernando '	'Carvajal Anacona'	18	5	50000	50	2500000	114000	0	0	0	300000	400000	130000	1784000
5	'T.I'	1130024111	'Juan Manuel'	'Gutierrez Fierro'	17	6	60000	60	3600000	0	0	0	0	432000	576000	187200	2404800
6	'C.C'	85456043	'Arnulfo Antonio'	'Yusunguaira'	19	7	70000	70	4900000	0	0	0	0	588000	784000	254800	3273200
7	'C.C'	1084331856	'Jesus David'	'Fierro'	18	8	80000	80	6400000	0	0	0.03	192000	768000	1024000	332800	4083200
8	'C.C'	1075231111	'Daniel Felipe'	'Bata'	18	9	90000	90	8100000	0	0	0.04	324000	972000	1296000	421200	5086800
9	'C.C'	100526532	'Andres Felipe'	'TresPalacios Per…	20	6	10500000	1	10500000	0	0	0.05	525000	1260000	1680000	546000	6489000

▸ Array(10)

Nombre del arreglo: pagoNomina	**Versión: 2.0**
Descripción: El arreglo pagoNomina la información de varias personas para calcular sus salarios netos. Primero, se calcula el salario bruto multiplicando los días trabajados por el valor diario. Luego, se aplican beneficios como el subsidio de transporte y bonificaciones según el estrato del empleado. Después, se deducen los costos de salud, pensión, ARL y retención de impuestos. Finalmente, se muestra en la consola una tabla con todos los detalles de los salarios netos de los empleados.	
iteración	Tipo de variable: Int
pago	Tipo de variable: Float
salarioM	Tipo de variable: Float
subTransporte	Tipo de variable: Float
abono	Tipo de variable: Float
retención	Tipo de variable: Float
reten	Tipo de variable: Float
pension	Tipo de variable: Float
arl	Tipo de variable: Float
deducibles	Tipo de variable: Float
salario	Tipo de variable: Float
Código:	

SENA	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

```
//Creamos un nuevo arreglo
let iteracion;
let pago;
let salarioM = 1300000;
let subTransporte;
let abono;
let retencion;
let reten;
let pension;
let arl;
let salud;
let deducibles;
let salario;

for (iteracion = 0; iteracion < nomina.length; iteracion++) {
  pago = nomina[iteracion].diasTrabajados * nomina[iteracion].valorDia;

  subT(pago);
  bono(pago, nomina[iteracion].estrato);
  saludNomina(pago);
  pensionNomina(pago);
  arlNomina(pago);
  retencionNomina(pago, nomina[iteracion].estrato);

  reten  = pago * retencion

  salario = ( pago + subTransporte + abono)-(salud+pension+arl+reten)
  pagoNomina.push({
    tipoDocumento: nomina[iteracion].tipoDocumento,
    numeroDocumento: nomina[iteracion].documento,
    nombres: nomina[iteracion].nombres,
    apellidos: nomina[iteracion].apellidos,
    edad: nomina[iteracion].edad,
    estrato: nomina[iteracion].estrato,
    valorDia: nomina[iteracion].valorDia,
    diasTrabajados: nomina[iteracion].diasTrabajados,
    SalarioBruto: pago,
    subTransporte: subTransporte,
    abono: abono,
    retencion: retencion,
    CantidadRetenida: reten,
    salud: salud,
    pension: pension,
    arl: arl,
    salaroNeto: salario,
  });
}
console.table(pagoNomina)
```

Nombre de la función: SubT(pago)	**Versión: 2.0**
Descripción: Función que calcula la asignación para transporte basada en el salario de la persona.	
Código:	

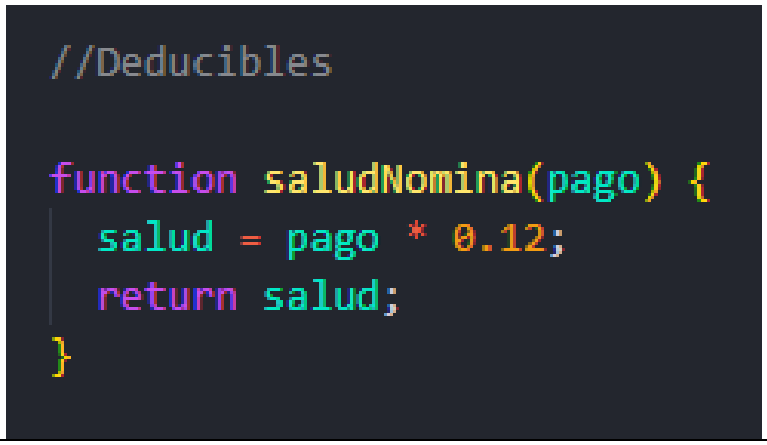
```
//subsidio de transporte

function subT(pago) {
  if (pago < 2 * salarioM) {
    subTransporte = 114000;
  } else {
    subTransporte = 0;
  }
  return subTransporte;
}
```

	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

Nombre de la función: bono(pago,estrato)	**Versión: 2.0**
Descripción: Función que calcula si el salario y el estrato de la persona cumplen ciertas condiciones para darles un bono.	
Descripción: Función que saluda	
Código: 	

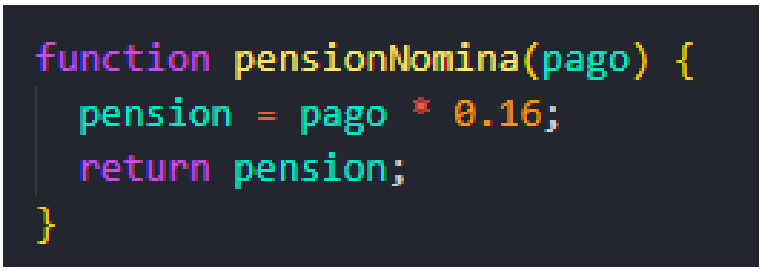
```
//Abono
function bono(pago, estrato) {
  if (pago < salarioM && estrato == 1 ||
      pago < salarioM && estrato == 2){
    abono = 100000;
  } else {
    abono = 0;
  }
  return abono;
}
```

Nombre de la función: saludNomina(pago)	**Versión: 2.0**
Descripción: Función que calcula el valor de una contribución a la salud, basada en el pago ingresado como parámetro.	
Código: 	

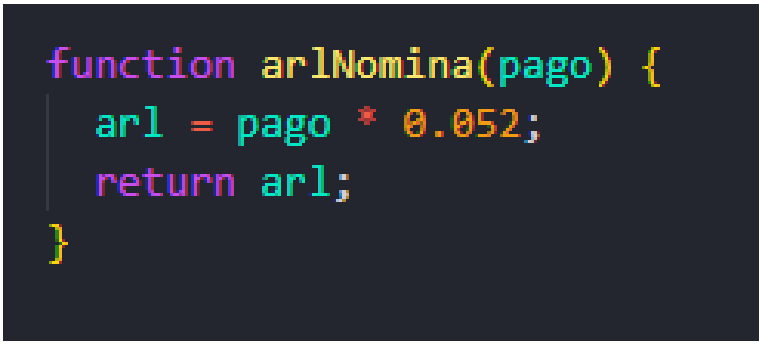
```
//Deducibles

function saludNomina(pago) {
  salud = pago * 0.12;
  return salud;
}
```

	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

Nombre de la función: pensionNomina(pago)	**Versión: 2.0**
Descripción: Función que calcula el valor de una contribución a la pensión, basada en el pago ingresado como parámetro.	
Código: 	

```
function pensionNomina(pago) {
  pension = pago * 0.16;
  return pension;
}
```

Nombre de la función: arlNomina(pago)	**Versión: 2.0**
Descripción: Función que calcula el valor de una contribución a la ARL, basada en el pago ingresado como parámetro.	
Código: 	

```
function arlNomina(pago) {
  arl = pago * 0.052;
  return arl;
}
```

SENA	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

Nombre de la función: retencionNomina(pago,estato)	**Versión: 2.0**
Descripción: Función que calcula la asignación para retención basada en el salario de la persona.	

Código:
`//Retencion`
`function retencionNomina(pago, estrato) {`
`  if (pago > 8 * salarioM && estrato == 6 ) {`
`  retencion = 0.05;`
`  } else if (pago > salarioM * 6) {`
`  retencion = 0.04;`
`  } else if (pago > salarioM * 4) {`
`  retencion = 0.03;`
`  } else {`
`  retencion = 0;`
`  }`
`  return retencion;`
`}`	

Nombre del arreglo: bingo	**Versión: 2.0**
Descripción: El arreglo bingo crea una matriz de 5x5, donde cada número se calcula multiplicando su posición por 3. Utiliza bucles anidados para llenar la matriz y luego muestra el resultado en forma de tabla en la consola.	
iteracion1	Tipo de variable: Int
iteracion2	Tipo de variable: Int
contador	Tipo de variable: Int
tabla	Tipo de variable: Int
Código:	

	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

```
//Arreglos
let bingo = [];

let iteracion1;
let iteracion2;
let contador = 0;
let tabla;

for (iteracion1 = 0; iteracion1 < 5; iteracion1++) {
    let interno = [];
    for (iteracion2 = 0; iteracion2 < 5; iteracion2++) {
    contador = contador + 1;
    tabla = contador * 3;
    interno.push(tabla);
    }

    bingo.push(interno);
}

console.table(bingo);
```

3	6	9	12	15
18	21	24	27	30
33	36	39	42	45
48	51	54	57	60
63	66	69	72	75

Nombre del arreglo: letras	**Versión: 2.0**
Descripción: El arreglo letras, organiza en arreglos separados las letras letraB, letraI, letraN, letraG, letra, en donde cada arreglo contiene valores de una columna de la matriz.	
Código:	

	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

```
//Asignale un arreglo a cada letra
let letraB = [];
let letraI = [];
let letraN = [];
let letraG = [];
let letraO = [];

for (iteracion1 = 0; iteracion1 < 5; iteracion1++) {
    letraB.push(bingo[iteracion1][0]);
    letraI.push(bingo[iteracion1][1]);
    letraN.push(bingo[iteracion1][2]);
    letraG.push(bingo[iteracion1][3]);
    letraO.push(bingo[iteracion1][4]);
}

console.log("Letra B " + letraB);
console.log("Letra I " + letraI);
console.log("Letra N " + letraN);
console.log("Letra G " + letraG);
console.log("Letra O " + letraO);
```

```
Letra B 3,18,33,48,63
Letra I 6,21,36,51,66
Letra N 9,24,39,54,69
Letra G 12,27,42,57,72
Letra O 15,30,45,60,75
```

Nombre del arreglo: x	**Versión: 2.0**
Descripción: El arreglo trae los elementos de las diagonales opuestas de una matriz bingo y los organiza en un arreglo x1. Luego, ordena y muestra los valores únicos en la consola, eliminando duplicados.	
saludo	Tipo de variable: String

	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

Código:

```
let x1 = [];

function mostrar(xp) {
    let org = xp;

    let imp = new Set(org);
    let result = [...imp];
    return result;
}

for (iteracion1 = 0; iteracion1 < 5; iteracion1++) {
    x1.push(bingo[iteracion1][iteracion1]);
    x1.push(bingo[iteracion1][4 - iteracion1]);
}

x1.sort((a,b) => a - b); // quita lo numeros repetidos
console.log("xGrande " + mostrar(x1));
```

```
xGrande 3,15,21,27,39,51,57,63,75
```

Nombre del arreglo: x	**Versión: 2.0**
Descripción:	

	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

El código extrae elementos de las diagonales opuestas de una matriz bingo y los organiza en un arreglo x2, x3, x4. Luego, ordena y muestra los valores únicos en la consola, eliminando duplicados.

saludo	Tipo de variable: String

Código:

```
let x2 = [];
let x3 = [];
let x4 = [];

for (iteracion1 = 0; iteracion1 < 3; iteracion1++) {

    //Sacar x pequeña
    x2.push(bingo[iteracion1][1 + iteracion1]);
    x2.push(bingo[iteracion1][3 - iteracion1]);

    //Sacar x Mediana
    x3.push(bingo[iteracion1 + 2][iteracion1]);
    x3.push(bingo[iteracion1 + 2][2 - iteracion1]);

    //Sacar x Mini
    x4.push(bingo[2 + iteracion1][2 + iteracion1]);
    x4.push(bingo[2 + iteracion1][4 - iteracion1]);
}

x2.sort((a,b) => a - b);
x3.sort((a,b) => a - b);
x4.sort((a,b) => a - b);



console.log('xMedia '+mostrar(x2));

console.log("xChica " + mostrar(x3));

console.log("xMini " + mostrar(x4));
```

```
xMedia 6,12,24,36,42
xChica 33,39,51,63,69
xMini 39,45,57,69,75
```

Nombre de la función: pares e impares	**Versión: 2.0**

Descripción:
Función que cuenta y muestra la cantidad de números pares e impares en una matriz 5x5 llamada bingo.

	Tecnólogo en Análisis y Desarrollo de Software Ficha
	Funciones JS

saludo	Tipo de variable: String

Código:

```
//Sacar los numeros pares e impares
let pares = 0;
let impares = 0;

for (iteracion1 = 0; iteracion1 < 5; iteracion1++) {

    for (iteracion2 = 0; iteracion2 < 5; iteracion2++) {
    if (bingo[iteracion1][iteracion2] % 2 == 0) {
        pares = pares + 1;
    } else {
        impares = impares + 1;
    }
    }
}

console.log("Tiene " + pares + " pares");
console.log("Tiene " + impares + " impares");
```

```
Tiene 12 pares
Tiene 13 impares
>
```