



PROGRAMA:

TECNOLOGO EN ANALISIS Y
DESARROLLO DE SOFTWARE

FICHA TECNICA 2900177

Presentado por:

Camilo Andrés Losada
Ramírez

Instructor:

ANDRES MORENO

Neiva Huila



Tecnólogo en Análisis y Desarrollo de Software


Ficha

Funciones JS

Nombre del arreglo: factura()		Versión: 1.0
Descripción: Función que permite realizar un saludo		
factura	tipo de variable: array	
valorTotalProducto	Tipo de variable: string	
iteracion	Tipo de variable: string	
totalPagar	Tipo de variable: string	
totalPagoProducto	Tipo de variable: string	
Código:		
<pre>let factura =[]; let valorTotalProducto; let iteracion; let totalPagar = []; let totalPagoProducto; factura =[{condigo : 1, nombreProducto : 'malteada' , cantidad: 2, valorUnidad :12000}, {condigo : 2, nombreProducto : 'picada' , cantidad: 3, valorUnidad :12000}, {condigo : 3, nombreProducto : 'hamburguesa mixta' , cantidad: 4, valorUnidad :1600}, {condigo : 4, nombreProducto : 'churrasco' , cantidad: 1, valorUnidad :25000}, {condigo : 5, nombreProducto : 'gaseosa' , cantidad: 5, valorUnidad :5000}, {condigo : 6, nombreProducto : 'limonada' , cantidad: 5, valorUnidad :6000}]; /** valorTotalProducto = factura[3].cantidad * factura[3].valorUnidad; console.log(factura[4].nombreProducto) console.log("Total Pagar : " +valorTotalProducto) */ for(iteracion = 0; iteracion<factura.length; iteracion++){ totalPagoProducto = factura[iteracion].cantidad * factura[iteracion].valorUnidad; totalPagar.push({nombreProducto : factura[iteracion].nombreProducto, cantidad : factura[iteracion].cantidad, valorUnidad: factura[iteracion].valorUnidad,totalPagar: totalPagoPro } console.table(totalPagar);</pre>		
<pre><!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>tabla</title> <script src="js/factura.js"></script> </head> <body> </body> </html></pre>		

(index)	nombreProducto	cantidad	valorUnidad	totalPagar
0	'malteada'	2	12000	24000
1	'picada'	3	12000	36000
2	'hamburguesa mixta'	4	1600	6400
3	'churrasco'	1	25000	25000
4	'gaseosa'	5	5000	25000
5	'limonada'	5	6000	30000

factura.js:32
▶ Array(6)

	<p>Tecnólogo en Análisis y Desarrollo de Software</p> <p>Ficha</p>
	<p>Funciones JS</p>

Nombre del arreglo: nomina()		Versión: 1.0
Descripción:		
Función que permite realizar un saludo		
totalRetencion	tipo de variable: int	
EstratoP	Tipo de variable: int	
Estrato	Tipo de variable: int	
subtrasP	Tipo de variable: int	
retencionP	Tipo de variable: int	
arIP	Tipo de variable: int	
pensionP	Tipo de variable: int	
saludP	Tipo de variable: int	
dias	Tipo de variable: int	
valor	Tipo de variable: int	
nomina	Tipo de variable: array	
iteracion	Tipo de variable: int	
totalPagar	Tipo de variable: int	
TotalPagarSueldo	Tipo de variable: string	

```
let totalRetencion;
let EstratoP;
let Estrato;
let subTrasP;
let retencionP;
let arlP;
let pensionP;
let saludP;
let dias;
let valor;
let nomina = [];
let iteracion;
let totalPagar = [];
let totalPagarSueldo;

nomina= [
  {Cedula : 1077724121, Nombre : 'Camilo andres ', Apellido: 'losada Ramirez', edad : 21, Estrato : 2, valorDia : 10000, DiasTrabajados : 30},
  {Cedula : 1075225114, Nombre : 'yerson stiven ', Apellido: 'cuellar rubiano', edad : 15, Estrato : 2, valorDia : 30000, DiasTrabajados : 30},
  {Cedula : 1080291867, Nombre : 'Ingrid ', Apellido: 'Medina Esquivel', edad : 18, Estrato : 1, valorDia : 250000, DiasTrabajados : 30},
  {Cedula : 1075793094, Nombre : 'Karol Natalia ', Apellido: 'Osorio Poveda', edad : 35, Estrato : 2, valorDia : 400000, DiasTrabajados : 30},
  {Cedula : 1077723426, Nombre : 'Brayan Santiago ', Apellido: 'Guerrero Mendez', edad : 19, Estrato : 2, valorDia : 60000, DiasTrabajados : 30},
  {Cedula : 1138274089, Nombre : 'Daniel ', Apellido: 'Caicedo Trujillo', edad : 25, Estrato : 4, valorDia : 700000, DiasTrabajados : 30},
  {Cedula : 1084331856, Nombre : 'Jesus David ', Apellido: 'Fierro', edad : 18, Estrato : 6, valorDia : 1000000, DiasTrabajados : 30},
  {Cedula : 1075548543, Nombre : 'Marcos ', Apellido: 'Rojas Alvarez', edad : 17, Estrato : 7, valorDia : 120000, DiasTrabajados : 30},
  {Cedula : 1005256532, Nombre : 'Andres Felipe ', Apellido: 'tresPalacios Perez', edad : 3, Estrato : 3, valorDia : 10000, DiasTrabajados : 30},
  {Cedula : 1075231111, Nombre : 'Daniel Felipe ', Apellido: 'Bata', edad : 17, Estrato : 1, valorDia : 12000, DiasTrabajados : 30}
]

for(iteracion = 0; iteracion<nomina.length; iteracion++){
  valor = nomina[iteracion].valorDia;
  dias = nomina[iteracion].DiasTrabajados;
  Estrato = nomina[iteracion].Estrato
  totalSueldo = sueldo(valor,dias);
  saludP = salud(totalSueldo);
  pensionP = pension(totalSueldo);
  arlP = arl(totalSueldo);
  subTrasP = subTras(totalSueldo)
  EstratoP = bonos(totalSueldo, Estrato)
  retencionP = reten(totalSueldo, Estrato)
  totalRetencion = totalSueldo * retencionP
  saldoTotal = (totalSueldo + subTrasP + EstratoP) - (retencionP + saludP + pensionP + arlP)
  totalPagar.push({Nombre : nomina[iteracion].Nombre, Apellido : nomina[iteracion].Apellido, Edad : nomina[iteracion].edad, Estrato : nomina[iteracion].Estrato, valorDia : valor,
})
}

console.table(totalPagar);
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>nomina</title>
  <script src="js/nomina.js"></script>
</head>
<body>

</body>
</html>
```

nomina.js:110															
(index)	Nombre	Apellido	Edad	Estrato	valorDia	DiasTra...	sueldo	salud	pension	arl	subTras...	bonific...	retenci...	DineroR...	sueldoT...
0	'Camilo...	'losada...	21	2	10000	30	300000	36000	48000	15600	114000	100000	0	0	414400
1	'yerson...	'cuella...	15	2	30000	30	900000	108000	144000	46800	114000	100000	0	0	815200
2	'Ingrid...	'Medina...	18	1	250000	30	7500000	900000	1200000	390000	0	0	0.03	225000	5009999...
3	'Karol ...	'Osorio...	35	2	400000	30	12000000	1440000	1920000	624000	0	0	0.04	480000	8015999...
4	'Brayan...	'Guerre...	19	2	60000	30	1800000	216000	288000	93600	114000	0	0	0	1316400
5	'Daniel...	'Caiced...	25	4	700000	30	21000000	2520000	3360000	1092000	0	0	0.04	840000	1402799...
6	'Jesus ...	'Fierro'	18	6	1000000	30	30000000	3600000	4800000	1560000	0	0	0.05	1500000	2003999...
7	'Marcos...	'Rojas ...	17	7	120000	30	3600000	432000	576000	187200	0	0	0	0	2404800
8	'Andres...	'tresPa...	3	3	10000	30	300000	36000	48000	15600	114000	0	0	0	314400
9	'Daniel...	'Bata'	17	1	12000	30	360000	43200	57600	18720	114000	100000	0	0	454480

► Array(10)

Nombre de la función: sueldo(pdiasT , pvalorD)		Versión: 1.0
Descripción: Función que calcula el sueldo de una persona		
diasT	Tipo de variable: int	
valorD	Tipo de variable: int	
pago	Tipo de variable: int	
<pre>function sueldo(pdiasT, pvalorD) { let diaT; //Dias Trabajados let valorD; //Valor por día let pago; diaT = pdiasT; valorD = pvalorD; pago = diaT * valorD; return pago; }</pre>		

Nombre de la función: salud(pago)		Versión: 1.0
Descripción: Función que calcula cuando dinero debera dar para la salud		
salud	Tipo de variable: int	
<pre>function salud(pago) { let saludP = pago * 0.12; return saludP; }</pre>		

Nombre de la función: pension(pago)		Versión: 1.0
Descripción: Función que calcula cuando dinero debera dar para la pension		
pensionP	Tipo de variable: int	
<pre>function pension(pago) { let pensionP = pago * 0.16; return pensionP; }</pre>		

Nombre de la función: arl(pago)		Versión: 1.0
Descripción: Función que calcula cuando dinero debera dar para la arl		
arlP	Tipo de variable: int	
<pre>function arl(pago) { let arlP = pago * 0.052; return arlP; }</pre>		

Nombre de la función: subtras(pago)		Versión: 1.0
Descripción: Función que calcula si la persona recibe o no recibe subtrasporte		
salarioM	Tipo de variable: int	
trans	Tipo de variable: int	
	<pre>function subTras(pago) { let salarioM = 1300000; let trans; if (pago <= salarioM * 2) { trans = 114000; } else { trans = 0; } return trans; }</pre>	

Nombre de la función: reten(pago)		Versión: 1.0
Descripción: Función que calcula si a la persona se le deberá retener una parte del sueldo		
salarioM	Tipo de variable: int	
retencion	Tipo de variable: int	
	<pre>function reten(pago) { let salarioM = 1300000; let retencion; if (pago > salarioM * 4) { retencion = pago * 0.04; } else { retencion = 0; } return retencion; }</pre>	

Nombre de la función: bonos(pago , EstratoP)		Versión: 1.0
Descripción: Función que valida si la persona es de estrato 1 y 2 en caso de ser así se le dará un bono de 100000		
bonificacion	Tipo de variable: int	
EstratoPersonas	Tipo de variable: int	
retencion	Tipo de variable: int	
	<pre>function bonos(pago , EstratoP){ let bonificacion; let EstratoPersonas = EstratoP let sueldo = pago if(sueldo <= 1300000 && (EstratoPersonas === 1 EstratoPersonas === 2)){ bonificacion = 100000 }else{ bonificacion = 0 } return bonificacion; }</pre>	



Tecnólogo en Análisis y Desarrollo de Software

Ficha

Funciones JS

Nombre del arreglo: bingo()		Versión: 1.0
Descripción: Función que permite mostrar y seleccionar algunos números del bingo		
bingo	Tipo de variable: array	
letraB	Tipo de variable: array	
letraI	Tipo de variable: array	
letraN	Tipo de variable: array	
letraG	Tipo de variable: array	
letraD	Tipo de variable: array	
X1	Tipo de variable: array	
X2	Tipo de variable: array	
X3	Tipo de variable: array	
X4	Tipo de variable: array	
Iteracion1	Tipo de variable: int	
Iteracion2	Tipo de variable: int	
contador	Tipo de variable: int	
pares	Tipo de variable: int	
impares	Tipo de variable: int	
tabla	Tipo de variable: int	


```

let bingo = [];
let letraB = [];
let letraI = [];
let letraN = [];
let letraG = [];
let letraO = [];
let x1 = [];
let x2 = [];
let x3 = [];
let x4 = [];

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

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.log(bingo);
  console.log("");

  //Asigne un arreglo a cada letra
  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);

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

  x1.sort(function (a, b) {
    return a - b;
  });

  console.log("xGrande " + mostrar(x1));

  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));
  console.log("");

  //Sacar los numeros pares e impares
  for (iteracion1 = 0; iteracion1 < 5; iteracion1++) {
    let interno = [];
    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");
}

```

```

<!DOCTYPE html>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Bingo</title>
  <script src="js/bingo.js"></script>
</head>
<body>

</body>
</html>

```

```

▼ (5) [Array(5), Array(5), Array(5), Array(5), Array(5)] ⓘ
  ▶ 0: (5) [3, 6, 9, 12, 15]
  ▶ 1: (5) [18, 21, 24, 27, 30]
  ▶ 2: (5) [33, 36, 39, 42, 45]
  ▶ 3: (5) [48, 51, 54, 57, 60]
  ▶ 4: (5) [63, 66, 69, 72, 75]
    length: 5
  ▶ [[Prototype]]: Array(0)

```

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

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

xMedia 6,12,24,36,42

xChica 33,39,51,63,69

xMini 39,45,57,69,75

Tiene 12 pares

Tiene 13 impares

Nombre de la función: mostrar(pago)		Versión: 1.0
Descripción: Función que permite seleccionar y mostrar cada una de los numeros		
org	Tipo de variable: int	
imp	Tipo de variable: int	
result	Tipo de variable: array	
<pre>function mostrar(xp) { let org = xp; let imp = new Set(org); let result = [...imp]; return result; }</pre>		