



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 de la función: saludo(psaludo)

Versión: 1.0

Descripción:

Función que saluda

saludo

Tipo de variable: Alfanumérico

Código:

```
function saludo(psaludo){  
    let saludar = psaludo;  
    return saludar;  
}
```

parametros: hola mundo

Nombre de la función: saludoExp(psaludo)

Versión: 2.0

Descripción:

Función que saluda

saludo

Tipo de variable: String

Código:

```
const saludoExp = function(psaludo){  
    let saludar = psaludo;  
    return saludar ;  
}
```

como una expresion : hola mundo



# Tecnólogo en Análisis y Desarrollo de Software

Ficha

## Funciones JS

Nombre de la función: suma (pnumUno, pnumDos)		Versión: 3.0
Descripción: Función que suma con una expresión(pnumUno , pnumDos)		
Sumar	Tipo de variable: int	
pnumUno	Tipo de variable: int	
pnumDos	Tipo de variable: int	
Código:		
<pre>let numUno; let numDos; const suma = function (pnumUno, pnumDos){   let sumar;   numUno = pnumUno;   numDos = pnumDos;   sumar = numUno + numDos;   return sumar; }</pre>		suma con parametro :10

Nombre de la función: suma1 (pnumUno, pnumDos)		Versión: 4.0
Descripción: Función que suma con parámetro (pnumUno , pnumDos)		
Sumar	Tipo de variable: int	
pnumUno	Tipo de variable: int	
pnumDos	Tipo de variable: int	
Código:		
<pre>function suma1(pnumUno , pnumDos){   let sumar;   numUno = pnumUno;   numDos = pnumDos;   sumar = numUno + numDos;   return sumar; }</pre>		suma con una expresion :13



# Tecnólogo en Análisis y Desarrollo de Software

## Ficha

### Funciones JS

Nombre de la función: suma (pnumUno, pnumDos)		Versión: 5.0
Descripción: Función que suma con una expresión (pnumUno , pnumDos)		
Sumar	Tipo de variable: int	
pnumUno	Tipo de variable: int	
pnumDos	Tipo de variable: int	
Código		
<pre>let numUno; let numDos;  const suma = function (pnumUno, pnumDos){   let sumar;   numUno = pnumUno;   numDos = pnumDos;   sumar = numUno + numDos;   return sumar; }</pre>		

<b>Nombre de la función:</b> restar (pnumUno, pnumDos)		<b>Versión:</b> 6.0
<b>Descripción:</b> Función que restar con una expresión (pnumUno , pnumDos)		
restar	Tipo de variable: int	
pnumUno	Tipo de variable: int	
pnumDos	Tipo de variable: int	

Código

```
const restar = function (pnumUno, pnumDos){  
  let resta;  
  numUno = pnumUno;  
  numDos = pnumDos;  
  resta = numUno + numDos;  
  return resta;  
}
```

<b>Nombre de la función:</b> multiplicación (pnumUno, pnumDos)		<b>Versión:</b> 7.0
<b>Descripción:</b> Función que multiplica con una expresión (pnumUno , pnumDos)		
multiplicacion	Tipo de variable: float	
pnumUno	Tipo de variable: int	
pnumDos	Tipo de variable: int	

Código

```
const multiplicacion = function (pnumUno, pnumDos){  
  let multiplicar;  
  numUno = pnumUno;  
  numDos = pnumDos;  
  multiplicar = numUno * numDos;  
  return multiplicar;  
}
```

<b>Nombre de la función:</b> división (pnumUno, pnumDos)		<b>Versión:</b> 8.0
<b>Descripción:</b> Función que división con una expresión (pnumUno, pnumDos)		
división	Tipo de variable: float	
pnumUno	Tipo de variable: int	
pnumDos	Tipo de variable: int	

## Código

```
const division = function (pnumUno, pnumDos){
  let dividir;
  numUno = pnumUno;
  numDos = pnumDos;
  dividir = numUno / numDos;
  return dividir;
}
```

**Nombre de la función: operaciones**  
(poperador, pnumUno, pnumDos)

**Versión: 9.0**

### Descripción:

Función que realiza las operaciones con una expresión (poperador, pnumUno, pnumDos)

operaciones	Tipo de variable: string
-------------	--------------------------

pnumUno	Tipo de variable: int
---------	-----------------------

pnumDos	Tipo de variable: int
---------	-----------------------

## Código

```
const Operaciones = function(poperador, pnumUno , pnumDos){
  let operador = poperador;
  numUno = pnumUno;
  numDos = pnumDos;
  if(operador == "suma"){
    return suma(numUno,numDos);
  }
  else if(operador == "resta"){
    return restar(numUno,numDos);
  }
  else if(operador == "multiplicacion"){
    return multiplicacion(numUno,numDos);
  }
  else if(operador == "division"){
    return division(numUno,numDos);
  }
  else {
    return "Error!!! no reconoce operador";
  }
}
```

suma con uno parametro :8

resta Uno parametro :20

multiplicacion Uno parametro :35

Suma Dos parametro :18

Division Uno parametro :5

suma Tres parametro :36

Multiplicacion Dos parametro :Error!!! no reconoce operador

suma Cuatro parametro: 80

Nombre de la función: suma1 (pnumUno, pnumDos)		Versión: 10.0
Descripción: Función que suma con un parámetro(pnumUno , pnumDos)		
Sumar1	Tipo de variable: int	
pnumUno	Tipo de variable: int	
pnumUno	Tipo de variable: int	
Código		
<pre>function suma1 (pnumUno, pnumDos){   let sumar1;   numUno = pnumUno;   numDos = pnumDos;   sumar1 = numUno + numDos;   return sumar1; }</pre>		


Nombre de la función: restar1 (pnumUno, pnumDos)		Versión: 11.0
Descripción: Función que restar con una parametro(pnumUno, pnumDos)		
Resta1	Tipo de variable: int	
pnumUno	Tipo de variable: int	
pnumDos	Tipo de variable: int	
Código		
<pre>function restar1 (pnumUno, pnumDos){   let resta1;   numUno = pnumUno;   numDos = pnumDos;   resta1 = numUno + numDos;   return resta1; }</pre>		

Nombre de la función: multiplicación1 (pnumUno, pnumDos)		Versión: 12.0
Descripción: Función que multiplica con un parámetro(pnumUno, pnumDos)		
multiplicacion1	Tipo de variable: int	
pnumUno	Tipo de variable: int	
pnumDos	Tipo de variable: int	
Código		
<pre>function multiplicacion1 (pnumUno, pnumDos){   let multiplicar1;   numUno = pnumUno;   numDos = pnumDos;   multiplicar1 = numUno * numDos;   return multiplicar1; }</pre>		

Nombre de la función: división (pnumUno, pnumDos)		Versión: 13.0
Descripción: Función de división con un parametro(pnumUno, pnumDos)		
division1	Tipo de variable: float	
pnumUno	Tipo de variable: int	
pnumDos	Tipo de variable: int	
Código		
<pre>function division1 (pnumUno, pnumDos){     let dividir1;     numUno = pnumUno;     numDos = pnumDos;     dividir1 = numUno / numDos;     return dividir1; }</pre>		




Nombre de la función: operaciones (poprador, pnumUno, pnumDos)		Versión: 14.0
Descripción: Función que realiza las operaciones con un parámetro (poprador, pnumUno, pnumDos)		
Operaciones1	Tipo de variable: string	
pnumUno	Tipo de variable: int	
pnumDos	Tipo de variable: int	
Código		
<pre>function Operaciones1(poperador, pnumUno , pnumDos){   let operador1 = poprador;   numUno = pnumUno;   numDos = pnumDos;   if(operador1 == "suma1"){     return suma1(numUno,numDos);   }   else if(operador1 == "restar1"){     return restar1(numUno,numDos);   }   else if(operador1 == "multiplicacion1"){     return multiplicacion1(numUno,numDos);   }   else if(operador1 == "division1"){     return division1(numUno,numDos);   }   else {     return "Error!!! no reconoce operador";   } }</pre>		
<pre>suma Uno con una expresion :9 resta Uno con una expresion :19 multiplicacion Uno con una expresion :25 Suma Dos con una expresion :18 Division Uno con una expresion :2 suma Tres con una expresion :45 Multiplicacion Dos con una expresion:Error!!! no reconoce operador suma Cuatro con una expresion:45</pre>		

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


<b>Nombre de la función:</b> porcentaje (pnumUno, pporce)		<b>Versión:</b> 15.0
<b>Descripción:</b> Función que saca porcentaje con una expresión(pnumUno , pporce)		
porcentaje	Tipo de variable: float	
nota	Tipo de variable: int	
total	Tipo de variable: float	
Código:		
<pre>const porcentaje = function (pnumUno , pporce ){     let nota = pnumUno;     let porcentaje = pporce;     let total;     total = nota / porcentaje;     return total; }</pre>		

Nombre de la función: porcentaje1		Versión: 16.0
Descripción: Función que saca porcentaje con parámetro (pnumUno )		
porcentaje	Tipo de variable: float	
nota	Tipo de variable: int	
porcentaje	Tipo de variable: float	
Código:		
<pre>function porcentaje1 (pnumUno , pporce){     let nota = pnumUno;     let porcentaje = pporce;     let total;     total = nota / porcentaje;     return total; }</pre>		

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

<b>Nombre de la función: promedio(pnumUno )</b>		<b>Versión: 17.0</b>
<b>Descripción:</b> Función que saca promedio con una expresión(pnumUno )		
nota	Tipo de variable: int	
Código:	<pre>const promedio = function (pnumUno ){     nota = pnumUno;     return nota; }</pre>	

<b>Nombre de la función: promedio1(pnumUno )</b>		<b>Versión: 18.0</b>
<b>Descripción:</b> Función que saca promedio con parámetro (pnumUno)		
nota	Tipo de variable: int	
Código:	<pre>function promedio1(pnumUno){     nota = pnumUno;     return nota; }</pre>	

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

<b>Nombre de la función: porNota(pnumUno, pporce)</b>		<b>Versión: 19.0</b>
<b>Descripción:</b> Función que saca 3 notas con una expresión(pnumUno , pporce)		
Nota	Tipo de variable: int	
porcentaje	Tipo de variable: float	
total	Tipo de variable: gloat	

Código:

```
const porNota = function (pnumUno , pporce ){
    let nota = pnumUno;
    let porcentaje = pporce;
    let total;
    total = nota * porcentaje;
    return total;
}
```

<b>Nombre de la función:</b> porNota1(pnumUno, pporce)		<b>Versión:</b> 20.0
<b>Descripción:</b> Función que saca 3 notas con parámetro (pnumUno, pporce)		
Notas	Tipo de variable: int	
porcentaje	Tipo de variable: float	
total	Tipo de variable: float	
Código:		
<pre>function porNota1 (pnumUno , pporce ){     let nota = pnumUno;     let porcentaje = pporce;     let total;     total = nota * porcentaje;     return total; }</pre>		



Tecnólogo en Análisis y Desarrollo de  
Software  
Ficha  
**Funciones JS**

<b>Nombre de la función:</b> AreaCuadrado(plado )		<b>Versión:</b> 21.0
<b>Descripción:</b> Función que saca el área de un cuadrado con parámetro (plado)		
AreaC	Tipo de variable: string	
lado	Tipo de variable: int	

Código:

```
function AreaCuadrado(plado){  
  let AreaC;  
  let lado = plado;  
  AreaC = lado * lado;  
  return AreaC  
}
```

<b>Nombre de la función:</b> AreaTriangulo(pbaseTri , palturaTri )		<b>Versión:</b> 22.0
<b>Descripción:</b> Función que saca el área de un triángulo con parámetro (pbaseTri , palturaTri )		
AreaTri	Tipo de variable: int	
BaseTri	Tipo de variable: int	
AlturaTri	Tipo de variable: float	
Código:		
<pre>function AreaTriangulo(pbaseTri , palturaTri){   let AreaTri;   let BaseTri = pbaseTri;   let AlturaTri = palturaTri   AreaTri = BaseTri * AlturaTri;   return AreaTri }</pre>		


<b>Nombre de la función:</b> <b>AreaRectangulo</b> (pbaseR , palturaR )		<b>Versión:</b> 23.0
<b>Descripción:</b> Función que saca el área de un rectangulo con parámetro (pbaseR, palturaR)		
AreaR	Tipo de variable: float	
BaseR	Tipo de variable: int	
AlturaR	Tipo de variable: int	
<b>Código:</b> <pre>function AreaRectangulo(pbaseR , palturaR){   let AreaR;   let BaseR = pbaseR;   let AlturaR = palturaR   AreaR = (BaseR * AlturaR) /2;   return AreaR }</pre>		

<b>Nombre de la función: AreaCuadrado1</b> (plado)		<b>Versión: 24.0</b>
<b>Descripción:</b> Función que saca el área de un triángulo con parámetro (pbasetri , palturaTri )		
AreaC1	Tipo de variable: int	
lado	Tipo de variable: int	
Código:	<pre>const AreaCuadrado1 = function (plado){   let AreaC1;   let lado = plado;   AreaC1 = lado * lado;   return AreaC1 }</pre>	

<b>Nombre de la función: AreaTriangulo1</b> pbasetri , palturaTri )		<b>Versión: 25.0</b>
<b>Descripción:</b> Función que saca el área de un cuadrado con parámetro (plado)		
AreaTri2	Tipo de variable: string	
baseTri	Tipo de variable: int	
AlturaTri2	Tipo de variable: int	
Código:	<pre>const AreaTriangulo2 = function (pbaseTri , palturaTri){   let AreaTri2;   let BaseTri = pbaseTri;   let AlturaTri = palturaTri   AreaTri2 = BaseTri * AlturaTri;   return AreaTri2 }</pre>	

<b>Nombre de la función: AreaRectangulo3</b> (pbaseR , palturaR)		<b>Versión: 26.0</b>
<b>Descripción:</b> Función que saca el área de un triángulo con parámetro (pbasetri , palturaTri )		
AreaR3	Tipo de variable: float	
BaseR	Tipo de variable: int	

AlturaR	Tipo de variable: int
Código: <pre>const AreaRectangulo3 = function (pbaseR , palturaR){     let AreaR3;     let BaseR = pbaseR;     let AlturaR = palturaR     AreaR3 = (BaseR * AlturaR) /2;     return AreaR3 }</pre>	

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

<b>Nombre de la función:</b> sueldo (pdiasT, pvalorD )		<b>Versión:</b> 27.0
<b>Descripción:</b> Función que saca el salario con un expresión (pdiasT, pvalorD )		
diasT	Tipo de variable: int	
valorD	Tipo de variable: int	
sueldoP	Tipo de variable: int	
Código:		
<pre>const sueldo = function(pdiasT, pvalorD){     let diasT = pdiasT;     let valorD = pvalorD;     let sueldoP     sueldoP = diasT * valorD;     return sueldoP; }</pre>		

<b>Nombre de la función:</b> <b>Salud</b> (sueldoP , pporceSalud )		<b>Versión: 29.0</b>
<b>Descripción:</b> Función que saca la salud con una expresion (sueldoP , pporceSalud )		
saludporc	Tipo de variable: float	
salurR	Tipo de variable: int	
Código:		
<pre>const salud = function(sueldoP , pporceSalud){     let saludPorc = pporceSalud;     let saludR;      saludR = sueldoP * saludPorc;      return saludR; }</pre>		

<b>Nombre de la función:</b> <b>Pension</b> (sueldoP , pporcePension )		<b>Versión: 30.0</b>
<b>Descripción:</b> Función que saca la pension con un expresión (sueldoP , pporcePension )		
pensionPorc	Tipo de variable: float	
pensionR	Tipo de variable: int	
Código:		
<pre>const pension = function(sueldoP , pporcePension){   let pensionPorc = pporcePension;   let pensionR;    pensionR = sueldoP * pensionPorc;    return pensionR; }</pre>		

<b>Nombre de la función:</b> <b>arl</b> (sueldoP , pporceArl)		<b>Versión: 31.0</b>
<b>Descripción:</b> Función que saca el Arl con una expresion (sueldoP , pporceArl)		
arlPorc	Tipo de variable: string	
arlR	Tipo de variable: int	



Código:

```
const arl = function(sueldoP , pporceArl){
  let arlPorc = pporceArl;
  let arlR;

  arlR = sueldoP * arlPorc;

  return arlR;
}
```

**Nombre de la función:** descuento (saludR ,  
pPensionR . ArlR )

**Versión: 32.0**

**Descripción:**

Función que saca el descuento con una expresion (saludR , pPensionR . ArlR )

saludP	Tipo de variable: int
--------	-----------------------

pensionP	Tipo de variable: int
----------	-----------------------

arlP	Tipo de variable: int
------	-----------------------

descuento	Tipo de variable: int
-----------	-----------------------

Código:

```
const descuento = function(saludR , pPensionR , arlR ){
  let saludP = saludR;
  let pensionP = pPensionR;
  let arlP = arlR;
  let descuentoP = saludP + pensionP + arlP;
  return descuentoP;
}
```

**Nombre de la función:** salario (sueldoP ,  
descuentoP )

**Versión: 33.0**

**Descripción:**

Función que saca el sueldo con un parametro (plado)

descuentoPers	Tipo de variable: int
---------------	-----------------------

sueldoPer	Tipo de variable: int
-----------	-----------------------

salarioP	Tipo de variable: int
----------	-----------------------

Código:

```
const salario = function(sueldoP ,descuentoP ){  
  let descuentoPers = descuentoP;  
  let sueldoPer = sueldoP;  
  let salarioP;  
  salarioP = sueldoPer - descuentoPers;  
  return salarioP  
}
```

```
sueldo de la persona (con una expresion) : 649500  
salud de la persona (con una expresion) : 77940  
pension de la persona (con una expresion) : 64950  
arl de la persona (con una expresion) : 33774  
total de la persona (con una expresion) : 472836
```

<b>Nombre de la función: sueldo1</b> (pdiasT, pvalorD )		<b>Versión: 34.0</b>
<b>Descripción:</b> Función que saca el salario con un parametro (pdiasT, pvalorD )		
diasT	Tipo de variable: int	
valorD	Tipo de variable: int	
sueldoP	Tipo de variable: int	
Código:		
<pre>function sueldo1(pdiasT, pvalorD){     let diasT = pdiasT;     let valorD = pvalorD;     let sueldoP     sueldoP = diasT * valorD;     return sueldoP; }</pre>		

<b>Nombre de la función:</b> <b>Salud1</b> (sueldoP , pporceSalud )		<b>Versión:</b> 35.0
<b>Descripción:</b> Función que saca la salud con una parametro (sueldoP , pporceSalud )		
saludporc	Tipo de variable: float	
salurR	Tipo de variable: int	

Código:

```
function salud1(sueldoP , pporceSalud){  
    let saludPorc = pporceSalud;  
    let saludR;  
  
    saludR = sueldoP * saludPorc;  
  
    return saludR;  
}
```

**Nombre de la función:**

**Pension1** (sueldoP , pporcePension )

**Versión: 36.0**

**Descripción:**

Función que saca la pension con un parametro (sueldoP , pporcePension )

pensionPorc

Tipo de variable: float

pensionR

Tipo de variable: int

Código:

```
function pension1(sueldoP , pporcePension){  
    let pensionPorc = pporcePension;  
    let pensionR;  
  
    pensionR = sueldoP * pensionPorc;  
  
    return pensionR;  
}
```

**Nombre de la función:** arl1 (sueldoP ,  
pporceArl)

**Versión: 37.0**

**Descripción:**

Función que saca el Arl con un parametro (sueldoP , pporceArl)

arlPorc

Tipo de variable: string

arlR

Tipo de variable: int

Código:

```
function arl1(sueldoP , pporceArl){  
    let arlPorc = pporceArl;  
    let arlR;  
  
    arlR = sueldoP * arlPorc;  
  
    return arlR;  
}
```

<b>Nombre de la función:</b> descuento1 (saludR , pPensionR . ArlR )		<b>Versión:</b> 38.0
<b>Descripción:</b> Función que saca el descuento con un parametro (saludR , pPensionR . ArlR )		
saludP	Tipo de variable: int	
pensionP	Tipo de variable: int	
arlP	Tipo de variable: int	
descuento	Tipo de variable: int	
Código:		
<pre>function descuento1(saludR , pPensionR , arlR ){     let saludP = saludR;     let pensionP = pPensionR;     let arlP = arlR;     let descuentoP = saludP + pensionP + arlP;     return descuentoP; }</pre>		

<b>Nombre de la función:</b> salario1 (sueldoP , descuentoP )		<b>Versión:</b> 39.0
<b>Descripción:</b> Función que saca el sueldo con un parametro (plado)		
descuentoPers	Tipo de variable: int	
sueldoPer	Tipo de variable: int	
salarioP	Tipo de variable: int	
Código:		
<pre>function salario1(sueldoP ,descuentoP ){   let descuentoPers = descuentoP;   let sueldoPer = sueldoP;   let salarioP;   salarioP = sueldoPer - descuentoPers;   return salarioP }</pre>		
<pre>suelo de la persona (con una expresion) : 1299000 salud de la persona (con una expresion) : 155880 pension de la persona (con una expresion) : 207840 arl de la persona (con una expresion) : 67548 total de la persona (con una expresion) : 867732</pre>		



# Tecnólogo en Análisis y Desarrollo de Software

## Ficha

### Funciones JS

<b>Nombre de la función:</b> numeros(pnumUno, pnumDos)		<b>Versión:</b> 40.0
<b>Descripción:</b> Función que calcula que numero es mayor con una expresión(pnumUno , pnumDos)		
numUno	Tipo de variable: int	
numDos	Tipo de variable: int	
<b>Código:</b> <pre>const numeros = function(pnumUno , pnumDos){   let numUno = pnumUno   let numDos = pnumDos   if(numUno &gt; numDos){     return "numero uno es mayor a numero dos"   }else{     return "numero dos es mayor a numero uno"   } }</pre>		

<b>Nombre de la función:</b> numeros1(pnumUno, pnumDos)		<b>Versión:</b> 41.0
<b>Descripción:</b> Función que calcula que numero es mayor con parámetro (pnumUno, pnumDos)		
numUno	Tipo de variable: int	
numDos	Tipo de variable: int	

Código:

```
function numeros1(pnumUno , pnumDos){  
    let numUno = pnumUno  
    let numDos = pnumDos  
    if(numUno > numDos){  
        return "numero uno es mayor a numero dos"  
    }else{  
        return "numero dos es mayor a numero uno"  
    }  
}
```

con parametros numero uno es mayor a numero dos



## Tecnólogo en Análisis y Desarrollo de Software

### Ficha


### Funciones JS

<b>Nombre de la función:</b> edad(pfechaActual, pfechaNacimiento)		<b>Versión:</b> 42.0
<b>Descripción:</b> Función que la edad de la persona con una expresión( <b>pfechaActual</b> , <b>pfechaNacimiento</b> )		
fechaActual	Tipo de variable: int	
fechaNacimiento	Tipo de variable: int	
edadPersona	Tipo de variable: int	
edad1	Tipo de variable: int	

Código:

```
const edad = function(pfechaActual , pfechaNacimiento){  
    let edadPersona  
    let fechaActual = pfechaActual  
    let fechaNacimiento = pfechaNacimiento  
    edadPersona = fechaActual - fechaNacimiento;  
    let edad1 = Math.floor(edadPersona / (1000 * 60 * 60 * 24 * 365.25));  
    return edad1  
}
```

<b>Nombre de la función:</b> edad1(pfechaActual, pfechaNacimiento)		<b>Versión:</b> 43.0
<b>Descripción:</b> Función que saca porcentaje con parámetro (pfechaActual, pfechaNacimiento)		
fechaActual		Tipo de variable: int
fechaNacimiento		Tipo de variable: int
edadPersona		Tipo de variable: int
edad1		Tipo de variable: int
Código:		
<pre>function edad1(pfechaActual , pfechaNacimiento){   let edadPersona   let fechaActual = pfechaActual   let fechaNacimiento = pfechaNacimiento   edadPersona = fechaActual - fechaNacimiento;   let edad1 = Math.floor(edadPersona / (1000 * 60 * 60 * 24 * 365.25));   return edad1 }</pre>		

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

<b>Nombre de la función:</b> mayor(pnumUno, pnumDos , pnumTres)		<b>Versión:</b> 44.0
<b>Descripción:</b> Función que realiza si los números son mayores o iguales con una expresión(pnumUno, pnumDos , pnumTres)		
numUno	Tipo de variable: int	
numDos	Tipo de variable: int	
numTres	Tipo de variable: int	

**Código:**

```
const mayor = function(pnumUno, pnumDos, pnumTres){
  let numUno = pnumUno
  let numDos = pnumDos
  let numTres = pnumTres

  if(numUno == numDos && numUno == numTres && numTres == numDos){
    return "los 3 son iguales"
  }else{
    if(numUno > numDos & numUno > numTres){
      return "el numero 1 es el mayor"
    }else{
      if(numDos > numUno & numDos > numTres){
        return "el numero 2 es el mayor"
      }else{
        return "el numero 3 es el mayor "
      }
    }
  }
}
```

el numero 2 es el mayor

**Nombre de la función:** mayor1: (pnumUno, pnumDos , pnumTres)

**Versión:** 45 .0

**Descripción:**

Función que realiza si los números son mayores o iguales con parámetro (pnumUno, pnumDos , pnumTres)

numUno      Tipo de variable: int

numDos      Tipo de variable: int

numTres      Tipo de variable: int

**Código:**

```
function mayor1(pnumUno, pnumDos, pnumTres){
  let numUno = pnumUno
  let numDos = pnumDos
  let numTres = pnumTres

  if(numUno == numDos && numUno == numTres && numTres == numDos){
    return "los 3 son iguales"
  }else{
    if(numUno > numDos & numUno > numTres){
      return "el numero 1 es el mayor"
    }else{
      if(numDos > numUno & numDos > numTres){
        return "el numero 2 es el mayor"
      }else{
        return "el numero 3 es el mayor "
      }
    }
  }
}
```

los 3 son iguales



Tecnólogo en Análisis y Desarrollo de  
Software  
Ficha

**Funciones JS**



Nombre de la función: figura1(plado1)		Versión: 46.0
Descripción: Función que saca el área de 3 cuadrados y muestra si es mayor o igual con una expresión (plado1)		
cuadrado1	Tipo de variable: int	
Lado1	Tipo de variable: int	
Código: <pre>const figura1 = function(plado1){   let cuadro1   lado1 = plado1   cuadro1 = lado1 * lado1   return cuadro1; }</pre>		

Nombre de la función: figura2(plado2)		Versión: 47.0
Descripción: Función que saca el área de 3 cuadrados y muestra si es mayor o igual con una expresión (plado2)		
Cudrado1	Tipo de variable: int	
Lado1	Tipo de variable: int	
Código:		
<pre>const figura2 = function(plado2){   let cuadro2   lado2 = plado2   cuadro2 = lado2 * lado2   return cuadro2; }</pre>		

<b>Nombre de la función: figura3(plado3)</b>		<b>Versión: 48.0</b>
<b>Descripción:</b> Función que saca el área de 3 cuadrados y muestra si es mayor o igual con una expresión (plado3)		
Cudrado1	Tipo de variable: int	
Lado1	Tipo de variable: int	

Código:	
<pre>const figura3 = function(plado3){   let cuadro3   lado3 = plado3   cuadro3 = lado3 * lado3   return cuadro3; }</pre>	

Nombre de la función: areaC(cuadro1 , cuadro2 , cuadro3)		Versión: 49.0
<b>Descripción:</b> Función que saca el área de 3 cuadrados y muestra si es mayor o igual con una expresión (cuadro1 , cuadro2 , cuadro3)		
areaF1	Tipo de variable: int	
areaF2	Tipo de variable: int	
areaF3	Tipo de variable: int	

Código:	
<pre>const areaC = function(cuadro1 , cuadro2 , cuadro3){   let areaF1 = cuadro1 ;   let areaF2 = cuadro2;   let areaF3 = cuadro3;    if(areaF1 == areaF2 &amp;&amp; areaF1 == areaF3 &amp;&amp; areaF3 == areaF2 ){     return "las 3 areas son iguales"   }else{     if(areaF1 &gt; areaF2 &amp; areaF1 &gt; areaF3){       return "el area del primer cuadrado es mayor"     }else{       if(areaF2 &gt; areaF1 &amp; areaF2 &gt; areaF3){         return "el area del segundo cuadrado es mayor"       }else{         return "el area del tercer cuadrado es mayor"       }     }   } }</pre>	
<pre>figura 1 con una expresion : 1600 figura 2 con una expresion : 1600 figura 3 con una expresion: 1600 las 3 areas son iguales</pre>	

Nombre de la función: figura4(plado1)		Versión: 50.0
<b>Descripción:</b> Función que saca el área de 3 cuadrados y muestra si es mayor o igual con una expresión (plado1)		
cuadrado1	Tipo de variable: int	

Lado1	Tipo de variable: int
Código: <pre>function figura4(plado1){   let cuadro1   lado1 = plado1   cuadro1 = lado1 * lado1   return cuadro1; }</pre>	

Nombre de la función: figura5(plado2)		Versión: 51.0
Descripción: Función que saca el área de 3 cuadrados y muestra si es mayor o igual con una expresión (plado2)		
Cudrado1	Tipo de variable: int	
Lado1	Tipo de variable: int	
Código: <pre>function figura5(plado2){   let cuadro2   lado2 = plado2   cuadro2 = lado2 * lado2   return cuadro2; }</pre>		

<b>Nombre de la función: figura6(plado3)</b>		<b>Versión: 52.0</b>
<b>Descripción:</b> Función que saca el área de 3 cuadrados y muestra si es mayor o igual con una expresión (plado3)		
Cudrado1	Tipo de variable: int	
Lado1	Tipo de variable: int	

Código:

```
function figura6(plado3){  
  let cuadro3  
  lado3 = plado3  
  cuadro3 = lado3 * lado3  
  return cuadro3;  
}
```

Nombre de la función: area1(cuadro1 , cuadro2 , cuadro3)	Versión: 53.0
--	---------------

**Descripción:**

Función que saca el área de 3 cuadrados y muestra si es mayor o igual con una expresión (cuadro1 , cuadro2 , cuadro3)

areaF1	Tipo de variable: int
areaF2	Tipo de variable: int
areaF3	Tipo de variable: int

Código:

```
function area1 (cuadro1 , cuadro2 , cuadro3){  
  let areaF1 = cuadro1 ;  
  let areaF2 = cuadro2;  
  let areaF3 = cuadro3;  
  
  if(areaF1 == areaF2 && areaF1 == areaF3 && areaF3 == areaF2 ){  
    return "las 3 areas son iguales"  
  }else{  
    if(areaF1 > areaF2 & areaF1 > areaF3){  
      return "el area del primer cuadrado es mayor"  
    }else{  
      if(areaF2 > areaF1 & areaF2 > areaF3){  
        return "el area del segundo cuadrado es mayor"  
      }else{  
        return "el area del tercer cuadrado es mayor"  
      }  
    }  
  }  
}
```

```
figura 1 con parametro : 400  
figura 2 con parametro : 400  
figura 3 con parametro: 400  
  
las 3 areas son iguales
```



Tecnólogo en Análisis y Desarrollo de  
Software  
Ficha

**Funciones JS**

<b>Nombre de la función:</b> contar(pcontador, pnumero)		<b>Versión:</b> 54.0
<b>Descripción:</b> Función que cuenta del hasta el número que digite con una expresión(pcontador, pnumero)		
resultado	Tipo de variable: string	
contador	Tipo de variable: int	
numero	Tipo de variable: int	
<b>Código:</b>		
<pre>const contar = function(pcontador , pnumero){   let resultado = "";   let contador = pcontador   let numero = pnumero    while(contador &lt;= numero){     resultado += `con una expresiom \${contador++}\n`   }   return resultado }</pre>		<pre>con una expresiom 2 con una expresiom 3 con una expresiom 4 con una expresiom 5 con una expresiom 6 con una expresiom 7 con una expresiom 8 con una expresiom 9</pre>

<b>Nombre de la función:</b> contar1(pcontador, pnumero)		<b>Versión:</b> 55.0
<b>Descripción:</b> Función cuenta del hasta el número que digite con parámetro(pcontador, pnumero)		
resultado	Tipo de variable: String	
Contador	Tipo de variable: int	
numero	Tipo de variable: int	
Código:		

```
function contar1(pcontador , pnumero){
    let resultado = "";
    let contador = pcontador
    let numero = pnumero

    while(contador <= numero){
        resultado += `con un parametro ${contador++}\n`
    }
    return resultado
}
```

```
con un parametro 0
con un parametro 1
con un parametro 2
con un parametro 3
con un parametro 4
con un parametro 5
con un parametro 6
con un parametro 7
con un parametro 8
con un parametro 9
con un parametro 10
```



# Tecnólogo en Análisis y Desarrollo de Software

## Ficha

### Funciones JS

Nombre de la función: factorial(pnumero, pfactorial)		Versión: 56.0
Descripción:		
Función que realiza el factorial de 5 con una expresión(pnumero, pfactorial)		
contador	Tipo de variable: string	
numero	Tipo de variable: int	
facto	Tipo de variable: int	
resultado	Tipo de variable: string	
Código:		
<pre>const factorial = function(pnumero , pfactorial){   let contador ;   let numero = pnumero;   let facto = pfactorial;   let resultado = "";    contador = 0;    while(contador &lt; numero){     contador++     facto = facto * contador     resultado += ` factorial de 5 con parametro : \${facto}\n`   }   return resultado }</pre>		
<pre>factorial de 5 con expresion : 1 factorial de 5 con expresion : 2 factorial de 5 con expresion : 6 factorial de 5 con expresion : 24 factorial de 5 con expresion : 120 factorial de 5 con expresion : 720</pre>		

<b>Nombre de la función:</b> factorial1(pnumero, pfactorial)	<b>Versión:</b> 28.0
<b>Descripción:</b> Función que realiza el factorial de 5 con parámetro (pnumero, pfactorial)	

contador	Tipo de variable: String
numero	Tipo de variable: int
facto	Tipo de variable: int
resultado	Tipo de variable: String

Código:


```
function factorial1(pnumero , pfactorial){
  let contador;
  let numero = pnumero;
  let facto = pfactorial;
  let resultado = "";

  contador = 0;

  while(contador < numero){

    contador++;
    facto = facto * contador
    resultado += ` factorial de 5 con expresion : ${facto}\n`
  }
  return resultado
}
```

```
factorial de 5 con parametro : 1
factorial de 5 con parametro : 2
factorial de 5 con parametro : 6
factorial de 5 con parametro : 24
factorial de 5 con parametro : 120
```

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

<p><b>Nombre de la función:</b> tablaMultiplicar(tabla, rango = 5)</p>		<p><b>Versión:</b> 57.0</p>
<p><b>Descripción:</b> Función que realiza la tabla del 5 con parametro(tabla, rango = 5)</p>		
tabla	Tipo de variable: int	
numero	Tipo de variable: int	
resultado	Tipo de variable: String	
resultadoFinish	Tipo de variable: String	
contadorTabla	Tipo de variable: String	
contador	Tipo de variable: String	

### Código:

```
function tablaMultiplicar(tabla, rango = 5 ){  
    let tablas = tabla; // tablas  
    let numero = rango; // rango multiplicacion 5 * n  
    let resultado;  
    let resultadoFinish = "";  
    let contadorTabla;  
    let contador;  
  
    contadorTabla = 0  
  
    while(contadorTabla < tablas){  
        contadorTabla++;  
        contador = 0;  
  
        while(contador < numero){  
            contador++;  
            resultado = contadorTabla * contador;  
  
            resultadoFinish += `Tablas De Multiplicar con parametro: ${contadorTabla} x ${contador} = ${resultado} \n`  
        }  
    }  
  
    return resultadoFinish;  
}
```

```
Tablas De Multiplicar con una expresion : 1 x 1 = 1  
Tablas De Multiplicar con una expresion : 1 x 2 = 2  
Tablas De Multiplicar con una expresion : 1 x 3 = 3  
Tablas De Multiplicar con una expresion : 1 x 4 = 4  
Tablas De Multiplicar con una expresion : 1 x 5 = 5  
Tablas De Multiplicar con una expresion : 2 x 1 = 2  
Tablas De Multiplicar con una expresion : 2 x 2 = 4  
Tablas De Multiplicar con una expresion : 2 x 3 = 6  
Tablas De Multiplicar con una expresion : 2 x 4 = 8  
Tablas De Multiplicar con una expresion : 2 x 5 = 10  
Tablas De Multiplicar con una expresion : 3 x 1 = 3  
Tablas De Multiplicar con una expresion : 3 x 2 = 6  
Tablas De Multiplicar con una expresion : 3 x 3 = 9  
Tablas De Multiplicar con una expresion : 3 x 4 = 12  
Tablas De Multiplicar con una expresion : 3 x 5 = 15  
Tablas De Multiplicar con una expresion : 4 x 1 = 4  
Tablas De Multiplicar con una expresion : 4 x 2 = 8  
Tablas De Multiplicar con una expresion : 4 x 3 = 12  
Tablas De Multiplicar con una expresion : 4 x 4 = 16  
Tablas De Multiplicar con una expresion : 4 x 5 = 20  
Tablas De Multiplicar con una expresion : 5 x 1 = 5  
Tablas De Multiplicar con una expresion : 5 x 2 = 10
```

**Nombre de la función:** tablaMultiplicar1(tabla, rango = 5)

**Versión:** 58.0

### Descripción:

Función que realiza la tabla del 5 con una expresión (tabla, rango = 5)

tabla	Tipo de variable: int
-------	-----------------------

numero	Tipo de variable: int
--------	-----------------------

resultado	Tipo de variable: String
-----------	--------------------------

resultadoFinish	Tipo de variable: String
-----------------	--------------------------



contadorTabla	Tipo de variable: String
contadorTabla	Tipo de variable: String

Código:

```
const tablaMultiplicar1 = function (tabla, rango = 5 ){

    let tablas = tabla; // tablas
    let numero = rango; // rango multiplicacion 5 * n
    let resultado;
    let resultadoFinish = "";
    let contadorTabla;
    let contador;

    contadorTabla = 0

    while(contadorTabla < tablas){
        contadorTabla++;
        contador = 0;

        while(contador < numero){
            contador++;
            resultado = contadorTabla * contador;

            resultadoFinish += `Tablas De Multiplicar con una expresion : ${contadorTabla} x ${contador} = ${resultado} \n`
        }

    }

    return resultadoFinish;
}
```

```
Tablas De Multiplicar con parametro: 1 x 1 = 1
Tablas De Multiplicar con parametro: 1 x 2 = 2
Tablas De Multiplicar con parametro: 1 x 3 = 3
Tablas De Multiplicar con parametro: 1 x 4 = 4
Tablas De Multiplicar con parametro: 1 x 5 = 5
Tablas De Multiplicar con parametro: 2 x 1 = 2
Tablas De Multiplicar con parametro: 2 x 2 = 4
Tablas De Multiplicar con parametro: 2 x 3 = 6
Tablas De Multiplicar con parametro: 2 x 4 = 8
Tablas De Multiplicar con parametro: 2 x 5 = 10
Tablas De Multiplicar con parametro: 3 x 1 = 3
Tablas De Multiplicar con parametro: 3 x 2 = 6
Tablas De Multiplicar con parametro: 3 x 3 = 9
Tablas De Multiplicar con parametro: 3 x 4 = 12
Tablas De Multiplicar con parametro: 3 x 5 = 15
Tablas De Multiplicar con parametro: 4 x 1 = 4
Tablas De Multiplicar con parametro: 4 x 2 = 8
Tablas De Multiplicar con parametro: 4 x 3 = 12
Tablas De Multiplicar con parametro: 4 x 4 = 16
Tablas De Multiplicar con parametro: 4 x 5 = 20
Tablas De Multiplicar con parametro: 5 x 1 = 5
Tablas De Multiplicar con parametro: 5 x 2 = 10
Tablas De Multiplicar con parametro: 5 x 3 = 15
```



# Tecnólogo en Análisis y Desarrollo de Software

## Ficha

### Funciones JS

<b>Nombre de la función:</b> tablaMultiplicar(tabla, rango = 5)		<b>Versión:</b> 59.0
<b>Descripción:</b> Función que realiza la tabla del 5 con parametro(tabla, rango = 5)		
tabla	Tipo de variable: int	
numero	Tipo de variable: int	
resultado	Tipo de variable: String	
resultadoFinish	Tipo de variable: String	
contadorTabla	Tipo de variable: String	
contador	Tipo de variable: String	
par	Tipo de variable: int	
impar	Tipo de variable: int	

## Código:

```
function tablaMultiplicar(tabla, rango = 5 ){

    let tablas = tabla; // tablas
    let numero = rango; // rango multiplicacion 5 * n
    let resultado;
    let resultadoFinish = "";
    let contadorTabla;
    let contador;
    let par = 0 ;
    let impar = 0;

    contadorTabla = 0

    while(contadorTabla < tablas){
        contadorTabla++;
        contador = 0;

        while(contador < numero){
            contador++
            resultado = contadorTabla * contador;

            if(resultado % 2 == 0){
                par++
                resultadoFinish += `multiplicacion con parametro ${contadorTabla} x ${contador} = ${resultado} buzz \n`
            }else{
                impar++
                resultadoFinish += `multiplicacion con parametro ${contadorTabla} x ${contador} = ${resultado} bass \n`
            }
        }

        resultadoFinish += `total numeros impares : ${impar}\n` ;
        resultadoFinish += `total numeros impares : ${impar}\n` ;
        resultadoFinish += `total numeros pares : ${par}\n` ;

        return resultadoFinish;
    }
}
```

```
multiplicacion con parametro 1 x 1 = 1 bass
multiplicacion con parametro 1 x 2 = 2 buzz
multiplicacion con parametro 1 x 3 = 3 bass
multiplicacion con parametro 1 x 4 = 4 buzz
multiplicacion con parametro 1 x 5 = 5 bass
multiplicacion con parametro 2 x 1 = 2 buzz
multiplicacion con parametro 2 x 2 = 4 buzz
multiplicacion con parametro 2 x 3 = 6 buzz
multiplicacion con parametro 2 x 4 = 8 buzz
multiplicacion con parametro 2 x 5 = 10 buzz
multiplicacion con parametro 3 x 1 = 3 bass
multiplicacion con parametro 3 x 2 = 6 buzz
multiplicacion con parametro 3 x 3 = 9 bass
multiplicacion con parametro 3 x 4 = 12 buzz
multiplicacion con parametro 3 x 5 = 15 bass
multiplicacion con parametro 4 x 1 = 4 buzz
multiplicacion con parametro 4 x 2 = 8 buzz
multiplicacion con parametro 4 x 3 = 12 buzz
multiplicacion con parametro 4 x 4 = 16 buzz
multiplicacion con parametro 4 x 5 = 20 buzz
```

<b>Nombre de la función: tablaMultiplica1r(tabla, rango = 5)</b>		<b>Versión: 60.0</b>
<b>Descripción:</b>		
Función que realiza la tabla del 5 con una expresion(tabla, rango = 5)		
tabla	Tipo de variable: int	
numero	Tipo de variable: int	
resultado	Tipo de variable: String	

resultadoFinish	Tipo de variable: String
contadorTabla	Tipo de variable: String
contador	Tipo de variable: String
par	Tipo de variable: int
impar	Tipo de variable: int

### Código:

```
const tablaMultiplicar1 = function(tabla, rango = 5 ){
  let tablas = tabla; // tablas
  let numero = rango; // rango multiplicacion 5 * n
  let resultado;
  let resultadoFinish = "";
  let contadorTabla;
  let contador;
  let par = 0 ;
  let impar = 0;

  contadorTabla = 0

  while(contadorTabla < tablas){
    contadorTabla++;
    contador = 0;

    while(contador < numero){
      contador++;
      resultado = contadorTabla * contador;

      if(resultado % 2 == 0){
        par++;
        resultadoFinish += `multiplicacion con una expresion ${contadorTabla} x ${contador} = ${resultado} buzz \n`;
      }else{
        impar++;
        resultadoFinish += `multiplicacion con una expresion ${contadorTabla} x ${contador} = ${resultado} bass \n`;
      }
    }

    resultadoFinish += `total numeros impares : ${impar}\n` ;
    resultadoFinish += `total numeros pares : ${par}\n` ;

    return resultadoFinish;
  }
}
```

```

multiplicacion con una expresion 1 x 1 = 1 bass
multiplicacion con una expresion 1 x 2 = 2 buzz
multiplicacion con una expresion 1 x 3 = 3 bass
multiplicacion con una expresion 1 x 4 = 4 buzz
multiplicacion con una expresion 1 x 5 = 5 bass
multiplicacion con una expresion 2 x 1 = 2 buzz
multiplicacion con una expresion 2 x 2 = 4 buzz
multiplicacion con una expresion 2 x 3 = 6 buzz
multiplicacion con una expresion 2 x 4 = 8 buzz
multiplicacion con una expresion 2 x 5 = 10 buzz
multiplicacion con una expresion 3 x 1 = 3 bass
multiplicacion con una expresion 3 x 2 = 6 buzz
multiplicacion con una expresion 3 x 3 = 9 bass
multiplicacion con una expresion 3 x 4 = 12 buzz
multiplicacion con una expresion 3 x 5 = 15 bass
multiplicacion con una expresion 4 x 1 = 4 buzz
multiplicacion con una expresion 4 x 2 = 8 buzz
multiplicacion con una expresion 4 x 3 = 12 buzz
multiplicacion con una expresion 4 x 4 = 16 buzz

```



# Tecnólogo en Análisis y Desarrollo de Software

## Ficha

### Funciones JS

Nombre de la función: numeros(pdigite)		Versión: 61.0
Descripción: Función del 1 al 5 con una expresión(pdigite)		
numeroDigitado	Tipo de variable: int	
contador	Tipo de variable: string	
mostrar	Tipo de variable: string	
Código:		
<pre>const numeros = function(Pdigite){   let numeroDigitado = Pdigite;   let contador;   let mostrar = "";   for(contador = 1; numeroDigitado &gt;=contador; contador++){     mostrar += ` numeros del 1 al 5 con expresion: \${contador}\n`   }   return mostrar }</pre>		
<pre>numeros del 1 al 5 con expresion: 1 numeros del 1 al 5 con expresion: 2 numeros del 1 al 5 con expresion: 3 numeros del 1 al 5 con expresion: 4 numeros del 1 al 5 con expresion: 5</pre>		

<b>Nombre de la función: contar1(pdigite)</b>		<b>Versión: 62.0</b>
<b>Descripción:</b> Función del 1 al 5 con una con parámetro(pdigite)		
resultado	Tipo de variable: String	
contador	Tipo de variable: int	
numero	Tipo de variable: int	

Código:

```
function contar1(pcontador , pnumero){
    let resultado = "";
    let contador = pcontador
    let numero = pnumero

    while(contador <= numero){
        resultado += `con un parametro ${contador++}\n`
    }
    return resultado
}
```

```
numeros del 1 al 5 con parametro: 1
numeros del 1 al 5 con parametro: 2
numeros del 1 al 5 con parametro: 3
numeros del 1 al 5 con parametro: 4
numeros del 1 al 5 con parametro: 5
```



## Tecnólogo en Análisis y Desarrollo de Software Ficha

### Funciones JS

<b>Nombre de la función: factorial (pdigite)</b>		<b>Versión: 63.0</b>
<b>Descripción:</b> Función que realiza el factorial de 5 con una expresión(pdigite)		
numeroDigitado	Tipo de variable: int	
contador	Tipo de variable: string	
mostrar	Tipo de variable: string	
factorial	Tipo de variable: int	
<b>Código:</b> <pre>const factorial = function(Pdigite){   let numeroDigitado = Pdigite;   let contador;   let factorial = 1;   let mostrar = "";   for(contador = 1; numeroDigitado &gt;=contador; contador++){     factorial = factorial * contador     mostrar += ` factorial de 5 con una expresion: \${factorial}\n`   }   return mostrar }</pre> factorial de 5 con una expresion: 1 factorial de 5 con una expresion: 2 factorial de 5 con una expresion: 6 factorial de 5 con una expresion: 24 factorial de 5 con una expresion: 120		


--

<b>Nombre de la función: : factorial1 (pdigite)</b>		<b>Versión: 64.0</b>
<b>Descripción:</b> Función que realiza el factorial de 5 con parámetro (pdigite)		
numeroDigite	Tipo de variable: int	
contador	Tipo de variable: String	
factorial	Tipo de variable: int	
mostrar	Tipo de variable: String	

Código:

```
function factorial1(Pdigite){  
  let numeroDigitado = Pdigite;  
  let contador;  
  let factorial = 1;  
  let mostrar = "";  
  for(contador = 1; numeroDigitado >=contador; contador++){  
    factorial = factorial * contador  
    mostrar += ` factorial de 5 con parametro: ${factorial}\n`  
  }  
  return mostrar  
}
```

factorial de 5 con parametro: 1  
factorial de 5 con parametro: 2  
factorial de 5 con parametro: 6  
factorial de 5 con parametro: 24  
factorial de 5 con parametro: 120

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

<b>Nombre de la función: multiplicacionNum1 (pdigite)</b>		<b>Versión: 65.0</b>
<b>Descripción:</b> Función que realiza la multiplicación del 5 con una expresión(pdigite)		
numDigitado	Tipo de variable: int	
contador	Tipo de variable: string	

multiplicación	Tipo de variable: string
resultado	Tipo de variable: string
<b>Código:</b> <pre>const multiplicacionNum1 = function(Pdigite){   let numDigitado = Pdigite   let contador   let multiplicacion   let resultado = ""    for(contador = 1; numDigitado &gt;= contador ; contador++){      multiplicacion = numDigitado * contador      resultado += `multiplicacion del 5 con una expresion \${numDigitado} x \${contador} = \${multiplicacion}\n`   }   return resultado }</pre> <p> multiplicacion del 5 con una expresion 5 x 1 = 5  multiplicacion del 5 con una expresion 5 x 2 = 10  multiplicacion del 5 con una expresion 5 x 3 = 15  multiplicacion del 5 con una expresion 5 x 4 = 20  multiplicacion del 5 con una expresion 5 x 5 = 25 </p>	

Nombre de la función: : mutiplicacion2 (pdigite)		Versión: 66.0
Descripción: Función que realiza la multiplicación del 5 con parámetro (pdigite)		
numDigitado	Tipo de variable: int	
contador	Tipo de variable: String	
multiplicación	Tipo de variable: String	
resultado	Tipo de variable: String	
Código:		
<pre>function multiplicacion2(Pdigite){   let numDigitado = Pdigite   let contador   let multiplicacion   let resultado = ""    for(contador = 1; numDigitado &gt;= contador ; contador++){      multiplicacion = numDigitado * contador      resultado += `multiplicacion del 5 con una expresion \${numDigitado} x \${contador} = \${multiplicacion}\n`   }    return resultado }</pre>		
multiplicacion del 5 con una expresion 5 x 1 = 5 multiplicacion del 5 con una expresion 5 x 2 = 10 multiplicacion del 5 con una expresion 5 x 3 = 15 multiplicacion del 5 con una expresion 5 x 4 = 20 multiplicacion del 5 con una expresion 5 x 5 = 25		





# Tecnólogo en Análisis y Desarrollo de Software

## Ficha

### Funciones JS

**Nombre de la función:** multiplicacionNum1 (pdigite)

**Versión:** 67.0

**Descripción:**

Función que realiza la multiplicación del 9 con una expresión (pdigite)

multiplicación

Tipo de variable: Alfanumérico

**Código:**

```
const multiplicacionNum1 = function(Pdigite){
  let numDigitado = Pdigite
  let contador
  let multiplicacion
  let resultado = ""
  let par = 0;
  let impar = 0;

  for(contador = 1; 5 >= contador ; contador++){

    multiplicacion = numDigitado * contador

    if (multiplicacion %2 == 0){
      par++

      resultado += `multiplicacion del 9 con una expresion ${numDigitado} x ${contador} = ${multiplicacion} par \n`
    } else {
      impar++

      resultado += `multiplicacion del 9 con una expresion ${numDigitado} x ${contador} = ${multiplicacion} impar \n`
    }
  }

  return resultado
}
```

```
multiplicacion del 9 con una expresion 9 x 1 = 9 impar
multiplicacion del 9 con una expresion 9 x 2 = 18 par
multiplicacion del 9 con una expresion 9 x 3 = 27 impar
multiplicacion del 9 con una expresion 9 x 4 = 36 par
multiplicacion del 9 con una expresion 9 x 5 = 45 impar
```

**Nombre de la función:** : multiplicacionNum2 (pdigite)

**Versión:** 68.0


**Descripción:**

Función que realiza la multiplicación del 9 con parámetro (pdigite)

multiplicación

Tipo de variable: String

Código:	
<pre>function multiplicacionNum2(Pdigite){     let numDigitado = Pdigite     let contador     let multiplicacion     let resultado = ""     let par = 0;     let impar = 0;      for(contador = 1; 5 &gt;= contador ; contador++){          multiplicacion = numDigitado * contador          if (multiplicacion %2 == 0){             par++              resultado += `multiplicacion del 9 con parametro \${numDigitado} x \${contador} = \${multiplicacion}   par \n`         } else {             impar++              resultado += `multiplicacion del 9 con parametro \${numDigitado} x \${contador} = \${multiplicacion}   impar \n`         }     }     return resultado }</pre>	
<pre>multiplicacion del 9 con parametro 9 x 1 = 9 impar multiplicacion del 9 con parametro 9 x 2 = 18 par multiplicacion del 9 con parametro 9 x 3 = 27 impar multiplicacion del 9 con parametro 9 x 4 = 36 par multiplicacion del 9 con parametro 9 x 5 = 45 impar</pre>	

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

<b>Nombre de la función: multiplicacionNum2 (pdigite , prango)</b>		<b>Versión: 69.0</b>
<b>Descripción:</b> Función que realiza la multiplicación del 5 con una expresión(pdigite)		
multiplicación	Tipo de variable: Alfanumérico	

Código:

```
const multiplicacionNum2 = function (Pdignite,prango){
  let rango = prango;
  let numDigitado = Pdignite;
  let contadorTabla;
  let contador ;
  let multiplicacion;
  let resultado = ""
  let par = 0;
  let impar = 0;

  for(contador = 1; numDigitado >= contador ; contador++){
    for(contadorTabla = 1; rango >= contadorTabla ; contadorTabla++){

      multiplicacion = contador * contadorTabla

      if (multiplicacion %2 == 0){
        par++

        resultado += ` con una expresion ${contador} x ${contadorTabla} = ${multiplicacion} buzz \n`
      } else {
        impar++

        resultado += ` con una expresion ${contador} x ${contadorTabla} = ${multiplicacion} bass \n`
      }
    }
  }
  resultado += `total numeros impares : ${impar}\n` ;
  resultado += `total numeros pares : ${par}\n` ;

  return resultado
}
```

con una expresion 1 x 1 = 1 bass  
con una expresion 1 x 2 = 2 buzz  
con una expresion 1 x 3 = 3 bass  
con una expresion 1 x 4 = 4 buzz  
con una expresion 1 x 5 = 5 bass  
con una expresion 2 x 1 = 2 buzz  
con una expresion 2 x 2 = 4 buzz  
con una expresion 2 x 3 = 6 buzz  
con una expresion 2 x 4 = 8 buzz  
con una expresion 2 x 5 = 10 buzz  
con una expresion 3 x 1 = 3 bass  
con una expresion 3 x 2 = 6 buzz  
con una expresion 3 x 3 = 9 bass  
con una expresion 3 x 4 = 12 buzz  
con una expresion 3 x 5 = 15 bass  
con una expresion 4 x 1 = 4 buzz  
con una expresion 4 x 2 = 8 buzz  
con una expresion 4 x 3 = 12 buzz  
con una expresion 4 x 4 = 16 buzz  
con una expresion 4 x 5 = 20 buzz  
con una expresion 5 x 1 = 5 bass  
con una expresion 5 x 2 = 10 buzz  
con una expresion 5 x 3 = 15 bass  
con una expresion 5 x 4 = 20 buzz  
con una expresion 5 x 5 = 25 bass  
total numeros impares : 9  
total numeros pares : 16

**Nombre de la función:** : multiplicacionNum1  
(pdignite , prango)

**Versión:** 70.0

**Descripción:**

Función que realiza la multiplicación del 5 con parámetro (pdignite)

multiplicación

Tipo de variable: String

Código:

```
function multiplicacionNum1(Pdigite,prango){
    let rango = prango;
    let numDigitado = Pdigite;
    let contadorTabla;
    let contador ;
    let multiplicacion;
    let resultado = ""
    let par = 0;
    let impar = 0;

    for(contador = 1; numDigitado >= contador ; contador++){
        for(contadorTabla = 1; rango >= contadorTabla ; contadorTabla++){
            multiplicacion = contador * contadorTabla

            if (multiplicacion %2 == 0){
                par++

                resultado += ` parametro ${contador} x ${contadorTabla} = ${multiplicacion} par \n`
            } else {
                impar++

                resultado += ` parametro ${contador} x ${contadorTabla} = ${multiplicacion} impar \n`
            }
        }
    }

    resultado += `total numeros impares : ${impar}\n` ;
    resultado += `total numeros pares : ${par}\n` ;

    return resultado
}
```

```
parametro 1 x 1 = 1 impar
parametro 1 x 2 = 2 par
parametro 1 x 3 = 3 impar
parametro 1 x 4 = 4 par
parametro 1 x 5 = 5 impar
parametro 2 x 1 = 2 par
parametro 2 x 2 = 4 par
parametro 2 x 3 = 6 par
parametro 2 x 4 = 8 par
parametro 2 x 5 = 10 par
parametro 3 x 1 = 3 impar
parametro 3 x 2 = 6 par
parametro 3 x 3 = 9 impar
parametro 3 x 4 = 12 par
parametro 3 x 5 = 15 impar
parametro 4 x 1 = 4 par
parametro 4 x 2 = 8 par
parametro 4 x 3 = 12 par
parametro 4 x 4 = 16 par
parametro 4 x 5 = 20 par
parametro 5 x 1 = 5 impar
parametro 5 x 2 = 10 par
parametro 5 x 3 = 15 impar
parametro 5 x 4 = 20 par
parametro 5 x 5 = 25 impar
total numeros impares : 9
total numeros pares : 16
```



# Tecnólogo en Análisis y Desarrollo de Software

## Ficha

### Funciones JS

Nombre de la función: multiplicacionNum2 (pdigite , prango)		Versión: 71.0
Descripción: Función que realiza la multiplicación del 5 con una expresión(pdigite)		
tablas	Tipo de variable: int	
numero	Tipo de variable: int	
resultado	Tipo de variable: string	
resultadoFinish	Tipo de variable: string	
contadorTabla	Tipo de variable: string	
contador	Tipo de variable: string	
par	Tipo de variable: int	
impar	Tipo de variable: int	

Código:

```
const multiplicacionNum2 = function (Pdigite,prango){
  let rango = prango;
  let numDigitado = Pdigite;
  let contadorTabla;
  let contador ;
  let multiplicacion;
  let resultado = ""
  let par = 0;
  let impar = 0;

  for(contador = 1; numDigitado >= contador ; contador++){
    for(contadorTabla = 1; rango >= contadorTabla ; contadorTabla++){

      multiplicacion = contador * contadorTabla

      if (multiplicacion %2 == 0){
        par++

        resultado += ` con una expresion ${contador} x ${contadorTabla} = ${multiplicacion} buzz \n`
      } else {
        impar++

        resultado += ` con una expresion ${contador} x ${contadorTabla} = ${multiplicacion} bass \n`
      }
    }
  }

  resultado += `total numeros impares : ${impar}\n` ;
  resultado += `total numeros pares : ${par}\n` ;

  return resultado
}
```

```

multiplicacion con una expresion 1 x 1 = 1 bass
multiplicacion con una expresion 1 x 2 = 2 buzz
multiplicacion con una expresion 1 x 3 = 3 bass
multiplicacion con una expresion 1 x 4 = 4 buzz
multiplicacion con una expresion 1 x 5 = 5 bass
multiplicacion con una expresion 2 x 1 = 2 buzz
multiplicacion con una expresion 2 x 2 = 4 buzz
multiplicacion con una expresion 2 x 3 = 6 buzz
multiplicacion con una expresion 2 x 4 = 8 buzz
multiplicacion con una expresion 2 x 5 = 10 buzz
multiplicacion con una expresion 3 x 1 = 3 bass
multiplicacion con una expresion 3 x 2 = 6 buzz
multiplicacion con una expresion 3 x 3 = 9 bass
multiplicacion con una expresion 3 x 4 = 12 buzz
multiplicacion con una expresion 3 x 5 = 15 bass
multiplicacion con una expresion 4 x 1 = 4 buzz
multiplicacion con una expresion 4 x 2 = 8 buzz
multiplicacion con una expresion 4 x 3 = 12 buzz
multiplicacion con una expresion 4 x 4 = 16 buzz
multiplicacion con una expresion 4 x 5 = 20 buzz
multiplicacion con una expresion 5 x 1 = 5 bass
multiplicacion con una expresion 5 x 2 = 10 buzz
multiplicacion con una expresion 5 x 3 = 15 bass
multiplicacion con una expresion 5 x 4 = 20 buzz
multiplicacion con una expresion 5 x 5 = 25 bass
total numeros impares : 9
total numeros pares : 16

```

Nombre de la función: : tablaMultiplicar (tabla , rango= 5)		Versión: 72.0
<b>Descripción:</b> Función que realiza la multiplicación del 5 con parámetro (tabla , rango= 5)		
tablas	Tipo de variable: int	
numero	Tipo de variable: int	
resultado	Tipo de variable: string	
resultadoFinish	Tipo de variable: string	
contadorTabla	Tipo de variable: string	
contador	Tipo de variable: string	
par	Tipo de variable: int	
impar	Tipo de variable: int	

### Código:

```
function tablaMultiplicar(tabla, rango = 5 ){  
  
    let tablas = tabla; // tablas  
    let numero = rango; // rango multiplicacion 5 * n  
    let resultado;  
    let resultadoFinish = "";  
    let contadorTabla;  
    let contador;  
    let par = 0 ;  
    let impar = 0;  
  
    contadorTabla = 0  
  
    while(contadorTabla < tablas){  
        contadorTabla++;  
        contador = 0;  
  
        while(contador < numero){  
            contador++  
            resultado = contadorTabla * contador;  
  
            if(resultado % 2 == 0){  
                par++  
                resultadoFinish += `multiplicacion con parametro ${contadorTabla} x ${contador} = ${resultado} buzz \n`  
            }else{  
                impar++  
                resultadoFinish += `multiplicacion con parametro ${contadorTabla} x ${contador} = ${resultado} bass \n`  
            }  
        }  
  
        resultadoFinish += `total numeros impares : ${impar}\n` ;  
        resultadoFinish += `total numeros pares : ${par}\n` ;  
  
        return resultadoFinish;  
    }  
}
```

```
multiplicacion con parametro 1 x 1 = 1 bass  
multiplicacion con parametro 1 x 2 = 2 buzz  
multiplicacion con parametro 1 x 3 = 3 bass  
multiplicacion con parametro 1 x 4 = 4 buzz  
multiplicacion con parametro 1 x 5 = 5 bass  
multiplicacion con parametro 2 x 1 = 2 buzz  
multiplicacion con parametro 2 x 2 = 4 buzz  
multiplicacion con parametro 2 x 3 = 6 buzz  
multiplicacion con parametro 2 x 4 = 8 buzz  
multiplicacion con parametro 2 x 5 = 10 buzz  
multiplicacion con parametro 3 x 1 = 3 bass  
multiplicacion con parametro 3 x 2 = 6 buzz  
multiplicacion con parametro 3 x 3 = 9 bass  
multiplicacion con parametro 3 x 4 = 12 buzz  
multiplicacion con parametro 3 x 5 = 15 bass  
multiplicacion con parametro 4 x 1 = 4 buzz  
multiplicacion con parametro 4 x 2 = 8 buzz  
multiplicacion con parametro 4 x 3 = 12 buzz  
multiplicacion con parametro 4 x 4 = 16 buzz  
multiplicacion con parametro 4 x 5 = 20 buzz  
multiplicacion con parametro 5 x 1 = 5 bass  
multiplicacion con parametro 5 x 2 = 10 buzz  
multiplicacion con parametro 5 x 3 = 15 bass  
multiplicacion con parametro 5 x 4 = 20 buzz  
multiplicacion con parametro 5 x 5 = 25 bass  
total numeros impares : 9  
total numeros pares : 16
```



# Tecnólogo en Análisis y Desarrollo de Software

## Ficha

### Funciones JS

<b>Nombre de la función:</b> : edad (pFechaActual , pFechaNacimiento)		<b>Versión:</b> 73.0
<b>Descripción:</b> Función que calcula la edad de 3 personas con una expresión(pFechaActual , pFechaNacimiento)		
edadPersona	Tipo de variable: string	
fechaActual	Tipo de variable: int	
fechaNacimiento	Tipo de variable: int	
edad1	Tipo de variable: Sting	
<b>Código:</b> <pre>let edad1 const edad = function(pFechaActual , pFechaNacimiento){   let edadPersona   let fechaActual = pFechaActual   let fechaNacimiento = pFechaNacimiento   edadPersona = fechaActual - fechaNacimiento;   edad1 = Math.floor(edadPersona / (1000 * 60 * 60 * 24 * 365.25));   return edad1 }</pre>		

<b>Nombre de la función:</b> : mayorEdad (edad1)		<b>Versión :</b> 74.0
<b>Descripción:</b> Función que verifica si la persona es mayor o menor de edad con una expresión(edad1 )		
edad1	Tipo de variable: Sting	



Código:

```
const mayorEdad = function(edad1 ){
  if(edad1 >= 18){
    return `es mayor de edad , la persona tiene : ${edad1}`
  }else{
    return `es menor de edad , la persona tiene : ${edad1}`
  }
}
```

Nombre de la función: : promedioEdad (persona1 , persona2 , persona 3)		Versión: 75.0
Descripción: Función que realiza el promedio de las edades con una expresión(persona1 , persona2 , persona3)		
edadPersona1	Tipo de variable: int	
edadPersona2	Tipo de variable: int	
edadPersona3	Tipo de variable: int	
resultado	Tipo de variable: float	

Código:

```
const promedioEdad = function(persona1 , persona2 , persona3 ){
  let edadPersona1
  let edadPersona2
  let edadPersona3
  edadPersona1 = persona1
  edadPersona2 = persona2
  edadPersona3 = persona3
  let resultado = (persona1 + persona2 + persona3)/3
  return resultado
}
```

Nombre de la función: : validar (resultado)		Versión: 76.0
Descripción: Función que valida si el promedio es mayor o menor a la mayoría de edad con una expresión(resultado)		
validarP	Tipo de variable: string	

Código:

```
const validar = function(resultado){
  let validarP = resultado
  if([validarP] >= 18){
    return `el promedio de la edad es superior a la mayoría de edad`
  }else{
    return `el promedio de la edad es inferior a la mayoría de edad`
  }
}
```

```
con una expresion
es mayor de edad , la persona tiene : 18
es menor de edad , la persona tiene : 17
es menor de edad , la persona tiene : 14
promedio de edades : 16.333333333333332
validacion : el promedio de la edad es inferior a la
mayoría de edad
```

Nombre de la función: : edad2 (pFechaActual , pFechaNacimiento)		Versión: 77.0
<b>Descripción:</b> Función que calcula la edad de 3 personas con un parámetro(pFechaActual , pFechaNacimiento)		
edadPersona	Tipo de variable: string	
fechaActual	Tipo de variable: int	
fechaNacimiento	Tipo de variable: int	
edad1	Tipo de variable: Sting	


Código:

```
function edad2(pFechaActual , pFechaNacimiento){
  let edadPersona
  let fechaActual = pFechaActual
  let fechaNacimiento = pFechaNacimiento
  edadPersona = fechaActual - fechaNacimiento;
  edad1 = Math.floor(edadPersona / (1000 * 60 * 60 * 24 * 365.25));
  return edad1
}
```

Nombre de la función: : mayorEdad1 (edad1)		Versión : 78.0
Descripción:		
Función que verifica si la persona es mayor o menor de edad con un parámetro(edad1 )		
edad1	Tipo de variable: Sting	
Código:		
<pre>function mayorEdad1 (edad1 ){     if(edad1 &gt;= 18){         return `es mayor de edad , la persona tiene : \${edad1}`     }else{         return `es menor de edad , la persona tiene : \${edad1}`     } }</pre>		

<b>Nombre de la función: : promedioEdad1 (persona1 , persona2 , persona 3)</b>		<b>Versión: 79.0</b>
<b>Descripción:</b> Función que realiza el promedio de las edades con un parámetro( <b>persona1 , persona2 , persona3</b> )		
edadPersona1	Tipo de variable: int	
edadPersona2	Tipo de variable: int	
edadPersona3	Tipo de variable: int	
resultado	Tipo de variable: float	
Código:		
<pre>function promedioEdad1 (persona1 , persona2 , persona3 ){     let edadPersona1     let edadPersona2     let edadPersona3     let resultado      edadPersona1 = persona1     edadPersona2 = persona2     edadPersona3 = persona3     resultado = (persona1 + persona2 + persona3)/3 }</pre>		

Nombre de la función: : validar1 (resultado)		Versión: 80.0
Descripción: Función que valida si el promedio es mayor o menor a la mayoría de edad con un parámetro (resultado)		
validarP	Tipo de variable: string	
Código:		
<pre>function validar1 (resultado){   let validarP = resultado   if([validarP] &gt;= 18){     return `el promedio de la edad es superior a la mayoría de edad`   }else{     return `el promedio de la edad es inferior a la mayoría de edad`   } }</pre>		
<pre>con parametro es mayor de edad , la persona tiene : 23 es menor de edad , la persona tiene : 17 es mayor de edad , la persona tiene : 21 promedio de edades : 20.33333333333332 validacion : el promedio de la edad es superior a la mayoría de edad</pre>		

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

<b>Nombre de la función: : sueldo (pdiasT , pvalorD)</b>		<b>Versión: 81.0</b>
<b>Descripción:</b> Función que calcula el salario de una persona con una expresión( <b>pdiasT , pvalorD</b> )		
diasT	Tipo de variable: int	
valorD	Tipo de variable: int	
sueldoP	Tipo de variable: int	

Código:

```
const sueldo = function(pdiasT, pvalorD){
  let diasT = pdiasT;
  let valorD = pvalorD;
  let sueldoP
  sueldoP = diasT * valorD;
  return sueldoP;
}
```

Nombre de la función: : salud (sueldoP )		Versión: 82.0
<b>Descripción:</b> Función que calcula el dinero por salud de una persona con una expresión(sueldoP )		
salarios	Tipo de variable: int	
saludR	Tipo de variable: float	
Código:		
<pre>const salud = function(sueldoP){     let salarioS = sueldoP;     let saludR;      saludR = salarioS * 0.12;      return saludR; }</pre>		

Nombre de la función: : pension (sueldoP )		Versión: 83.0
<b>Descripción:</b> Función que calcula la pensión de una persona con una expresión( <b>sueldoP</b> )		
salarioP	Tipo de variable: int	
pensionR	Tipo de variable: int	

Código:

```
const pension = function(sueldoP ){  
    let salarioP = sueldoP;  
    let pensionR;  
  
    pensionR = salarioP * 0.16;  
  
    return pensionR;  
}
```

**Nombre de la función:** : arl (sueldoP )

**Versión:** 84.0

**Descripción:**

Función que calcule el arl de una personas con una expresión(**sueldoP** )

salarioA	Tipo de variable: int
----------	-----------------------

arlR	Tipo de variable: int
------	-----------------------

Código:

```
const arl = function(sueldoP ){  
    let salarioA = sueldoP;  
    let arlR;  
  
    arlR = salarioA * 0.052;  
  
    return arlR;  
}
```

**Nombre de la función:** : retencion (sueldoP )

**Versión:** 85.0

**Descripción:**

Función que calcule la retención de una persona con una expresión(**sueldoP** )

retencionP	Tipo de variable: string
------------	--------------------------

salarioM	Tipo de variable: int
----------	-----------------------

Código:

```
const retencion = function(sueldoP){
  let retencionP;
  let salarioM = 1300000;
  if(sueldoP > 4*salarioM ){
    retencionP = sueldoP * 0.04;
  }else{
    retencionP = 0;
  }
  return retencionP;
}
```

Nombre de la función: : subTrasporte (sueldoP )		Versión: 86.0
Descripción: Función que calcule el subTrasporte de una persona con una expresión(sueldoP )		
subTras	Tipo de variable: string	
salarioM	Tipo de variable: int	
Código:		
<pre>const subTrasporte = function(sueldoP){   let subTras;   let salarioM = 1300000;   if(sueldoP &lt; 2 *salarioM ){     subTras = 114000;   }else{     subTras = 0;   }   return subTras }</pre>		

Nombre de la función: : sueldoTotal (sueldoP )		Versión: 44.0
Descripción: Función que calcule la el sueldo de una persona con una expresión(sueldoP )		
salario	Tipo de variable: int	
saludF	Tipo de variable: string	
pensionF	Tipo de variable: string	
arIF	Tipo de variable: string	

retencionF	Tipo de variable: string
subTrasporteF	Tipo de variable: string
descuento	Tipo de variable: int
totalSalario	Tipo de variable: float

Código:

```
const sueldoTotal = function(sueldoP){
  let salario = sueldoP;
  let saludF = salud(sueldoP);
  let pensionF = pension(sueldoP);
  let arlF = arl(sueldoP);
  let retencionF = retencion(sueldoP);
  let subTrasporteF = subTrasporte(sueldoP);
  let descuento = saludF + pensionF + arlF;
  let totalSalario = (salario + subTrasporteF) - (retencionF + descuento);
  return totalSalario
}
```

```
sueldo con expresion 1500000
salud con expresion 180000
pension con expresion 240000
arl con expresion 78000
retencion con expresion 0
subTrasporte con expresion 114000
sueldoTotal con expresion 1116000
```

<b>Nombre de la función: : sueldoParametro (pdiasT , pvalorD)</b>		<b>Versión: 87.0</b>
<b>Descripción:</b> Función que calcula el salario de una persona con un parametro( <b>pdiasT , pvalorD</b> )		
diasT	Tipo de variable: int	
valorD	Tipo de variable: int	
sueldoP	Tipo de variable: int	
Código:		
<pre>function sueldoParametro(pdiasT, pvalorD){   let diasT = pdiasT;   let valorD = pvalorD;   let sueldoP1   sueldoP1 = diasT * valorD;   return sueldoP1; }</pre>		



Nombre de la función: : saludParametro (sueldoP )		Versión: 88.0
Descripción: Función que calcula el dinero por salud de una persona con un parametro (sueldoP )		
salarios	Tipo de variable: int	
saludR	Tipo de variable: float	
Código:		
<pre>function saludParametro(sueldoP1){   let salarioS1 = sueldoP1;   let saludR1;    saludR1 = salarioS1 * 0.12;    return saludR1; }</pre>		

Nombre de la función: : pensionParametro (sueldoP )		Versión: 89.0
Descripción: Función que calcula la pensión de una persona con un parametro (sueldoP )		
salarioP	Tipo de variable: int	
pensionR	Tipo de variable: int	
Código:		
<pre>function pensionParametro(sueldoP1 ){     let salarioP = sueldoP1;     let pensionR;      pensionR = salarioP * 0.16;      return pensionR; }</pre>		

Nombre de la función: : arlParametro (sueldoP )		Versión: 90.0
Descripción: Función que calcule el arl de una personas con una expresión(sueldoP )		
salarioA	Tipo de variable: int	
arlR	Tipo de variable: int	
Código:		
<pre>function arlParametro(sueldoP1 ){     let salarioA = sueldoP1;     let arlR;      arlR = salarioA * 0.052;      return arlR; }</pre>		

<b>Nombre de la función: : subTrasporteParametro (sueldoP )</b>		<b>Versión: 91.0</b>
<b>Descripción:</b> Función que calcule el subTrasporte de una persona con un Parametro ( <b>sueldoP</b> )		
subTras	Tipo de variable: string	
salarioM	Tipo de variable: int	
Código:	<pre>function subTrasporteParametro(sueldoP1){     let subTras;     let salarioM = 1300000;     if(sueldoP1&gt;2 *salarioM ){         subTras = 114000;     }else{         subTras = 0;     }     return subTras }</pre>	

<b>Nombre de la función: : retencionParametro (sueldoP )</b>		<b>Versión: 92.0</b>
<b>Descripción:</b> Función que calcule la retención de una persona con un parametro( <b>sueldoP</b> )		
retencionP	Tipo de variable: string	
salarioM	Tipo de variable: int	
Código:	<pre>function retencionParametro(sueldoP1){   let retencionP;   let salarioM = 1300000;   if(sueldoP1&gt;4 *salarioM ){     retencionP = sueldoP1 * 0.04;   }else{     retencionP = 0;   }   return retencionP; }</pre>	

<b>Nombre de la función: : sueldoTotalParametro (sueldoP )</b>		<b>Versión: 93.0</b>
<b>Descripción:</b> Función que calcule la el sueldo de una persona con un parametro ( <b>sueldoP</b> )		
salario	Tipo de variable: int	
saludF	Tipo de variable: string	
pensionF	Tipo de variable: string	
arlF	Tipo de variable: string	
retencionF	Tipo de variable: string	
subTrasporteF	Tipo de variable: string	
descuento	Tipo de variable: int	
totalSalario	Tipo de variable: float	

Código:

```
function sueldoTotalParametro(sueldoP1){  
  let salario = sueldoP1;  
  let saludF = salud(sueldoP1);  
  let pensionF = pension(sueldoP1);  
  let arlF = arl(sueldoP1);  
  let retencionF = retencion(sueldoP1);  
  let subTrasporteF = subTrasporte(sueldoP1);  
  let descuento = saludF + pensionF + arlF;  
  let totalSalario = (salario + subTrasporteF) - (retencionF + descuento);  
  return totalSalario  
}
```

sueldo con parametro	3000000
salud con parametro	360000
pension con parametro	480000
arl con parametro	156000
retencion con parametro	0
subTrasporte con parametro	114000
sueldoTotal con parametro	2004000