**Manual de funciones en JavaScript**



**Aprendiz:**

**Juan Felipe Guerrero**

**Instructor:  
Andrés Moreno Collazos**

**Centro de la Industria, la Empresa y los Servicios**

**Servicio Nacional de Aprendizaje – SENA**

**ADSO 2900177**

**2024**

**EJERCICIO 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre de la función: saludo(psaludo)** | | | **Versión: 1.0** | |
| **Descripción:**  Función que saluda | | | | |
| **Variables:**   1. saludar | **Tipo de variable:**   1. caracteres | | | |
| **Código:**  function **saludo**(*psaludar*) {      let saludar = psaludar;  ***return*** `${saludar} Parametros`;  } | | | | |
| **Consola:** | | | | |
| **Nombre de la función: saludoExp(psaludo)** | | | | **Versión: 2.0** |
| **Descripción:**  Función que saluda | | | | |
| **Variables:**   1. saludar | | **Tipo de variable:**   1. caracteres | | |
| **Código:**  const **saludoExp** = function (*psaludar*) {      let saludar = psaludar;  ***return*** `${saludar} Expresion`;  } | | | | |
| **Consola:** | | | | |

**EJERCICIO 2**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: suma(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que suma dos números | | |
| **Variables:**   1. num1 2. num2 3. sumar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **suma**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let sumar;      sumar = num1 + num2;  ***return*** sumar;  } | | |
| **Consola:** | | |
| **Nombre de la función: sumaExp(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que suma dos números | | |
| **Variables:**   1. num1 2. num2 3. sumar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **sumaExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let sumar;      sumar = num1 + num2;  ***return*** sumar;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: resta(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que resta dos números | | |
| **Variables:**   1. num1 2. num2 3. restar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **resta**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let restar;      restar = num1 - num2;  ***return*** restar;  } | | |
| **Consola:** | | |
| **Nombre de la función: restaExp(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que resta dos números | | |
| **Variables:**   1. num1 2. num2 3. restar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **restaExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let restar;      restar = num1 - num2;  ***return*** restar;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: multiplicacion(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que multiplica dos números | | |
| **Variables:**   1. num1 2. num2 3. multiplicar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **multiplicacion**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let multiplicar;      multiplicar = num1 \* num2;  ***return*** multiplicar;  } | | |
| **Consola:** | | |
| **Nombre de la función: multiplicacionExp(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que multiplica dos números | | |
| **Variables:**   1. num1 2. num2 3. multiplicar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **multiplicacionExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let multiplicar;      multiplicar = num1 \* num2;  ***return*** multiplicar;  } | | |
| **Consola:** | | |

|  |  |
| --- | --- |
| **Nombre de la función: division(pnum1, pnum2)** | **Versión: 1.0** |
| **Descripción:**  Función que divide dos números | |
| **Variables:**   1. num1 2. num2 3. dividir | **Tipo de variable:**   1. numérico 2. numérico 3. numérico |
| **Código:**  function **division**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let dividir;      dividir = num1 / num2;  ***return*** dividir;  } | |
| **Consola:** | |
| **Nombre de la función: divisionExp(pnum1, pnum2)** | **Versión: 2.0** |
| **Descripción:**  Función que divide dos números | |
| **Variables:**   1. num1 2. num2 3. dividir | **Tipo de variable:**   1. numérico 2. numérico 3. numérico |
| **Código:**  const **divisionExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let dividir;      dividir = num1 / num2;  ***return*** dividir;  } | |
| **Consola:** | |

**EJERCICIO 3**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: operations(poperation,pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que realiza la operación de dos números según el operador o la operación deseada | | |
| **Variables:**   1. operation 2. num1 3. num2 4. result | **Tipo de variable:**   1. alfanumérico 2. numérico 3. numérico 4. alfanumérico | |
| **Código:**  function **operations**(*poperation*, *pnum1*, *pnum2*) {      let operation = poperation.**toLowerCase**();      let num1 = pnum1;      let num2 = pnum2;      let result;  ***if*** (operation == "suma" ***||*** operation == '+') {          result = **suma**(num1, num2);      }  ***else*** ***if*** (operation == "resta" ***||*** operation == '-') {          result = **resta**(num1, num2);      }  ***else*** ***if*** (operation == "multiplicacion" ***||*** operation == '\*' ***||*** operation == 'x') {          result = **multiplicacion**(num1, num2);      }  ***else*** ***if*** (operation == "division" ***||*** operation == '/') {          result = **division**(num1, num2);      }  ***return*** result;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: operationsExp(poperation,pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que realiza la operación de dos números según el operador o la operación deseada | | |
| **Variables:**   1. operation 2. num1 3. num2 4. result | **Tipo de variable:**   1. alfanumérico 2. numérico 3. numérico 4. alfanumérico | |
| **Código:**  const **operationsExp** = function (*poperation*, *pnum1*, *pnum2*) {      let operation = poperation.**toLowerCase**();      let num1 = pnum1;      let num2 = pnum2;      let result;  ***if*** (operation == "suma" ***||*** operation == '+') {          result = **suma**(num1, num2);      }  ***else*** ***if*** (operation == "resta" ***||*** operation == '-') {          result = **resta**(num1, num2);      }  ***else*** ***if*** (operation == "multiplicacion" ***||*** operation == '\*' ***||*** operation == 'x') {          result = **multiplicacion**(num1, num2);      }  ***else*** ***if*** (poperation == "division" ***||*** operation == '/') {          result = **division**(num1, num2);      }  ***else*** {          result = 'Operacion no valida';      }  ***return*** result;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: suma(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que suma dos números | | |
| **Variables:**   1. num1 2. num2 3. sumar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **suma**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let sumar;      sumar = num1 + num2;  ***return*** `Suma: ${sumar}`;  } | | |
| **Consola:** | | |
| **Nombre de la función: sumaExp(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que suma dos números | | |
| **Variables:**   1. num1 2. num2 3. sumar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **sumaExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let sumar;      sumar = num1 + num2;  ***return*** `Suma: ${sumar}`;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: resta(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que resta dos números | | |
| **Variables:**   1. num1 2. num2 3. restar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **resta**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let restar;      restar = num1 - num2;  ***return*** `Resta: ${restar}`;  } | | |
| **Consola:** | | |
| **Nombre de la función: restaExp(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que resta dos números | | |
| **Variables:**   1. num1 2. num2 3. restar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **restaExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let restar;      restar = num1 - num2;  ***return*** `Resta: ${restar}`;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: multiplicacion(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que multiplica dos números | | |
| **Variables:**   1. num1 2. num2 3. multiplicar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **multiplicacion**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let multiplicar;      multiplicar = num1 \* num2;  ***return*** `Multiplicacion: ${multiplicar}`;  } | | |
| **Consola:** | | |
| **Nombre de la función: multiplicacionExp(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que multiplica dos números | | |
| **Variables:**   1. num1 2. num2 3. multiplicar | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **multiplicacionExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let multiplicar;      multiplicar = num1 \* num2;  ***return*** `Multiplicacion: ${multiplicar}`;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: division(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que divide dos números | | |
| **Variables:**   1. num1 2. num2 3. dividir | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **division**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let dividir;      dividir = num1 / num2;  ***return*** `Division: ${dividir}`;  } | | |
| **Consola:** | | |
| **Nombre de la función: divisionExp(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que divide dos números | | |
| **Variables:**   1. num1 2. num2 3. dividir | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **divisionExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let dividir;      dividir = num1 / num2;  ***return*** `Division: ${dividir}`;  } | | |
| **Consola:** | | |

**EJERCICIO 4**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: porcentajes(pnum)** | | **Versión: 1.0** |
| **Descripción:**  Función que obtiene el porcentaje de un número | | |
| **Variables:**   1. num 2. porcentaje | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  function **porcentajes**(*pnum*) {      let num = pnum;      let porcentaje = num / 100;  ***return*** porcentaje;  } | | |
| **Consola:** | | |
| **Nombre de la función: porcentajesExp(pnum)** | | **Versión: 2.0** |
| **Descripción:**  Función que obtiene el porcentaje de un número | | |
| **Variables:**   1. num 2. porcentaje | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  const **porcentajesExp** = function (*pnum*) {      let num = pnum;      let porcentaje = num / 100;  ***return*** porcentaje;  } | | |
| **Consola:** | | |

**EJERCICIO 5**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: promedio(pnota1, pnota2, pnota3)** | | **Versión: 1.0** |
| **Descripción:**  Función que obtiene el promedio de tres notas | | |
| **Variables:**   1. nota1 2. nota2 3. nota3 4. promedio | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico | |
| **Código:**  function **promedio**(*pnota1*, *pnota2*, *pnota3*) {      let nota1 = pnota1;      let nota2 = pnota2;      let nota3 = pnota3;      let promedio = (nota1 + nota2 + nota3) / 3;  ***return*** promedio;  } | | |
| **Consola:** | | |
| **Nombre de la función: promedioExp(pnota1, pnota2, pnota3)** | | **Versión: 2.0** |
| **Descripción:**  Función que obtiene el promedio de tres notas | | |
| **Variables:**   1. nota1 2. nota2 3. nota3 4. promedio | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico | |
| **Código:**  const **promedioExp** = function (*pnota1*, *pnota2*, *pnota3*) {      let nota1 = pnota1;      let nota2 = pnota2;      let nota3 = pnota3;      let promedio = (nota1 + nota2 + nota3) / 3;  ***return*** promedio;  } | | |
| **Consola:** | | |

**EJERCICIO 6**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: notas(pnota, pporcent)** | | **Versión: 1.0** |
| **Descripción:**  Función que saca el valor de la nota según su porcentaje | | |
| **Variables:**   1. nota 2. porcent 3. result | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **notas**(*pnota*, *pporcent*) {      let nota = pnota;      let porcent = pporcent;      let result = nota \* porcent;  ***return*** result;  } | | |
| **Consola:** | | |
| **Nombre de la función: notasExp(pnota, pporcent)** | | **Versión: 2.0** |
| **Descripción:**  Función que saca el valor de la nota según su porcentaje | | |
| **Variables:**   1. nota 2. porcent 3. result | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **notasExp** = function (*pnota*, *pporcent*) {      let nota = pnota;      let porcent = pporcent;      let result = nota \* porcent;  ***return*** result;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: sumar(pnota1, pnota2, pnota3)** | | **Versión: 1.0** |
| **Descripción:**  Función que suma tres notas | | |
| **Variables:**   1. nota1 2. nota2 3. nota3 4. suma | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico | |
| **Código:**  function **sumar**(*pnota1*, *pnota2*, *pnota3*) {      let nota1 = pnota1;      let nota2 = pnota2;      let nota3 = pnota3;      let suma = nota1 + nota2 + nota3;  ***return*** suma;  } | | |
| **Consola:** | | |
| **Nombre de la función: sumarExp(pnota1, pnota2, pnota3)** | | **Versión: 2.0** |
| **Descripción:**  Función que suma tres notas | | |
| **Variables:**   1. nota1 2. nota2 3. nota3 4. suma | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico | |
| **Código:**  const **sumarExp** = function (*pnota1*, *pnota2*, *pnota3*) {      let nota1 = pnota1;      let nota2 = pnota2;      let nota3 = pnota3;      let suma = nota1 + nota2 + nota3;  ***return*** suma;  } | | |
| **Consola:** | | |

**EJERCICIO 7**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: areas(pfigura, pbase, paltura)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el área de la figura geométrica elegida | | |
| **Variables:**   1. figura 2. base 3. altura 4. area | **Tipo de variable:**   1. caracteres 2. numérico 3. numérico 4. numérico | |
| **Código:**  function **areas**(*pfigura*, *pbase*, *paltura*) {      let figura = pfigura.**toLowerCase**();      let base = pbase;      let altura = paltura;      let area;      area = base \* altura;  ***if*** (figura == 'cuadrado') {  ***return*** area;      } ***else*** ***if*** (figura == 'rectangulo') {  ***return*** area;      } ***else*** ***if*** (figura == 'triangulo') {          area /= 2;  ***return*** area;      } ***else*** {  ***return*** 'Figura no valida';      }  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: areasExp(pfigura, pbase, paltura)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el área de la figura geométrica elegida | | |
| **Variables:**   1. figura 2. base 3. altura 4. area | **Tipo de variable:**   1. caracteres 2. numérico 3. numérico 4. numérico | |
| **Código:**  const **areasExp** = function (*pfigura*, *pbase*, *paltura*) {      let figura = pfigura.**toLowerCase**();      let base = pbase;      let altura = paltura;      let area;      area = base \* altura;  ***if*** (figura == 'cuadrado') {  ***return*** area;      } ***else*** ***if*** (figura == 'rectangulo') {  ***return*** area;      } ***else*** ***if*** (figura == 'triangulo') {          area /= 2;  ***return*** area;      } ***else*** {  ***return*** 'Figura no valida';      }  } | | |
| **Consola:** | | |

**EJERCICIO 8**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: salario(pdias, pvalDias)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el salario de una persona | | |
| **Variables:**   1. dias 2. valDias 3. salario 4. salud 5. pension 6. arl 7. desc | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. numérico 6. numérico 7. numérico | |
| **Código:**  function **salario**(*pdias*, *pvalDias*) {      let dias = pdias;      let valDias = pvalDias;      let salario = dias \* valDias;      let salud = salario \* 0.12;      let pension = salario \* 0.16;      let arl = salario \* 0.052;      let desc = salud + pension + arl;      salario -= desc;  ***return*** salario;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: salarioExp(pdias, pvalDias)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el salario de una persona | | |
| **Variables:**   1. dias 2. valDias 3. salario 4. salud 5. pension 6. arl 7. desc | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. numérico 6. numérico 7. numérico | |
| **Código:**  const **salarioExp** = function (*pdias*, *pvalDias*) {      let dias = pdias;      let valDias = pvalDias;      let salario = dias \* valDias;      let salud = salario \* 0.12;      let pension = salario \* 0.16;      let arl = salario \* 0.052;      let desc = salud + pension + arl;      salario -= desc;  ***return*** salario;  } | | |
| **Consola:** | | |

**EJERCICIO 9**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: edades(pedad)** | | **Versión: 1.0** |
| **Descripción:**  Función que dice si es mayor o menor de edad | | |
| **Variables:**   1. edad | **Tipo de variable:**   1. numérico | |
| **Código:**  function **edades**(*pedad*) {      let edad = pedad;  ***if*** (edad >= 18) {  ***return*** 'Mayor de edad';      } ***else*** {  ***return*** 'Menor de edad';      }  } | | |
| **Consola:** | | |
| **Nombre de la función: edadesExp(pedad)** | | **Versión: 2.0** |
| **Descripción:**  Función que dice si es mayor o menor de edad | | |
| **Variables:**   1. edad | **Tipo de variable:**   1. numérico | |
| **Código:**  const **edadesExp** = function (*pedad*) {      let edad = pedad;  ***if*** (edad >= 18) {  ***return*** 'Mayor de edad';      } ***else*** {  ***return*** 'Menor de edad';      }  } | | |
| **Consola:** | | |

**EJERCICIO 10**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: edades(panoNa, panoAc)** | | **Versión: 1.0** |
| **Descripción:**  Función que dice si es mayor o menor de edad según el año de nacimiento y año actual | | |
| **Variables:**   1. anoNa 2. anoAc 3. edad | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **edades**(*panoNa*, *panoAc*) {      let anoNa = panoNa;      let anoAc = panoAc;      let edad = anoAc - anoNa;  ***if*** (edad > 17) {  ***return*** 'Mayor de edad';      } ***else*** {  ***return*** 'Menor de edad';      }  } | | |
| **Consola:** | | |
| **Nombre de la función: edadesExp(panoNa, panoAc)** | | **Versión: 2.0** |
| **Descripción:**  Función que dice si es mayor o menor de edad según el año de nacimiento y año actual | | |
| **Variables:**   1. anoNa 2. anoAc 3. edad | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **edadesExp** = function (*panoNa*, *panoAc*) {      let anoNa = panoNa;      let anoAc = panoAc;      let edad = anoAc - anoNa;  ***if*** (edad > 17) {  ***return*** 'Mayor de edad';      } ***else*** {  ***return*** 'Menor de edad';      }  } | | |
| **Consola:** | | |

**EJERCICIO 11**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que dice cual número es mayor | | |
| **Variables:**   1. num1 2. num2 | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  function **numeros**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;  ***if*** (num1 == num2) {  ***return*** 'Son iguales';      } ***else*** ***if*** (num1 > num2) {  ***return*** `${num1} es mayor`;      } ***else*** {  ***return*** `${num2} es mayor`;      }  } | | |
| **Consola:** | | |
| **Nombre de la función** **numerosExp(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que dice cual número es mayor | | |
| **Variables:**   1. num1 2. num2 | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  const **numerosExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;  ***if*** (num1 == num2) {  ***return*** 'Son iguales';      } ***else*** ***if*** (num1 > num2) {  ***return*** `${num1} es mayor`;      } ***else*** {  ***return*** `${num2} es mayor`;      }  } | | |
| **Consola:** | | |

**EJERCICIO 12**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: areas(plado)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el área de un cuadrado | | |
| **Variables:**   1. lado 2. area | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  function **areas**(*plado*) {      let lado = plado;      let area = lado \*\* 2;  ***return*** area;  } | | |
| **Consola:** | | |
| **Nombre de la función:** **areasExp(plado)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el área de un cuadrado | | |
| **Variables:**   1. lado 2. area | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  const **areasExp** = function (*plado*) {      let lado = plado;      let area = lado \*\* 2;  ***return*** area;  } | | |
| **Consola:** | | |
| **Resultado:** | | |

**EJERCICIO 13**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: edades(panoNa, panoAc)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula la edad según el año de nacimiento y dice si es mayor o menor de edad | | |
| **Variables:**   1. anoNa 2. anoAc 3. edad 4. cont | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico | |
| **Código:**  function **edades**(*panoNa*, *panoAc*) {      let anoNa = panoNa;      let anoAc = panoAc;      let edad = anoAc - anoNa;  ***if*** (edad > 17) {          console.**log**(`La persona ${cont} es mayor de edad`);      } ***else*** {          console.**log**(`La persona ${cont} es menor de edad`);      }      cont += 1;  ***return*** edad;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función:** **edadesExp(panoNa, panoAc)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula la edad según el año de nacimiento y dice si es mayor o menor de edad | | |
| **Variables:**   1. anoNa 2. anoAc 3. edad 4. cont | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico | |
| **Código:**  const **edadesExp** = function (*panoNa*, *panoAc*) {      let anoNa = panoNa;      let anoAc = panoAc;      let edad = anoAc - anoNa;  ***if*** (edad > 17) {          console.**log**(`La persona ${cont} es mayor de edad`);      } ***else*** {          console.**log**(`La persona ${cont} es menor de edad`);      }      cont += 1;  ***return*** edad;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: promedio(psuma, pnum)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el promedio de una suma de numeros | | |
| **Variables:**   1. suma 2. num 3. result | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **promedio**(*psuma*, *pnum*) {      let suma = psuma;      let num = pnum;      let result = suma / num;  ***return*** result;  } | | |
| **Consola:** | | |
| **Nombre de la función: promedioExp(psuma, pnum)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el promedio de una suma de numeros | | |
| **Variables:**   1. suma 2. num 3. result | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **promedioExp** = function (*psuma*, *pnum*) {      let suma = psuma;      let num = pnum;      let result = suma / num;  ***return*** result;  } | | |
| **Consola:** | | |
| **Resultado:** | | |

**EJERCICIO 14**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: salario(pdias, pvalDias)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el salario de una persona | | |
| **Variables:**   1. dias 2. valDias 3. valSalario | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **salario**(*pdias*, *pvalDias*) {      let dias = pdias;      let valDias = pvalDias;      let valSalario = dias \* valDias;  ***return*** valSalario;  } | | |
| **Consola:** | | |
| **Nombre de la función: salarioExp(pdias, pvalDias)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el salario de una persona | | |
| **Variables:**   1. dias 2. valDias 3. valSalario | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **salarioExp** = function (*pdias*, *pvalDias*) {      let dias = pdias;      let valDias = pvalDias;      let valSalario = dias \* valDias;  ***return*** valSalario;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: subTrans(pvalSalario)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el valor del subsidio de transporte según una condicion | | |
| **Variables:**   1. valSalario 2. salMin 3. valSubTrans | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **subTrans**(*pvalSalario*) {      let valSalario = pvalSalario;      let salMin = 1600000;      let valSubTrans;  ***if*** (valSalario <= salMin \* 2) {          valSubTrans = 114000;      } ***else*** {          valSubTrans = 0;      }  ***return*** valSubTrans;  } | | |
| **Consola:** | | |
| **Nombre de la función: subTransExp(pvalSalario)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el valor del subsidio de transporte según una condicion | | |
| **Variables:**   1. valSalario 2. salMin 3. valSubTrans | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **subTransExp** = function (*pvalSalario*) {      let valSalario = pvalSalario;      let salMin = 1600000;      let valSubTrans;  ***if*** (valSalario <= salMin \* 2) {          valSubTrans = 114000;      } ***else*** {          valSubTrans = 0;      }  ***return*** valSubTrans;  } | | |
| **Consola:** | | |
| **Nombre de la función: retencion(pvalSalario)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula la retención según una condición | | |
| **Variables:**   1. valSalario 2. salMin 3. valRetencion | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **retencion**(*pvalSalario*) {      let valSalario = pvalSalario;      let salMin = 1600000;      let valRetencion;  ***if*** (valSalario >= salMin \* 4) {          valRetencion = salario \* 0.4;      } ***else*** {          valRetencion = 0;      }  ***return*** valRetencion;  } | | |
| **Nombre de la función: retencionExp(pvalSalario)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula la retención según una condición | | |
| **Variables:**   1. valSalario 2. salMin 3. valRetencion | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **retencionExp** = function (*pvalSalario*) {      let valSalario = pvalSalario;      let salMin = 1600000;      let valRetencion;  ***if*** (valSalario >= salMin \* 4) {          valRetencion = salario \* 0.4;      } ***else*** {          valRetencion = 0;      }  ***return*** valRetencion;  } | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: deducible(pvalSalario)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el deducible del salario | | |
| **Variables:**   1. valSalud 2. valPension 3. valArl 4. valDeducible | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico | |
| **Código:**  function **deducible**(*pvalSalario*) {      let valSalud = **salud**(pvalSalario);      let valPension = **pension**(pvalSalario);      let valArl = **arl**(pvalSalario);      console.**log**(`Salud: ${valSalud}\nPension: ${valPension}\narl: ${valArl}`);      let valDeducible = valSalud + valPension + valArl;  ***return*** valDeducible;  } | | |
| **Consola:** | | |
| **Nombre de la función: deducibleExp(pvalSalario)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el deducible del salario | | |
| **Variables:**   1. valSalud 2. valPension 3. valArl 4. valDeducible | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico | |
| **Código:**  const **deducibleExp** = function (*pvalSalario*) {      let valSalud = **salud**(pvalSalario);      let valPension = **pension**(pvalSalario);      let valArl = **arl**(pvalSalario);      console.**log**(`Salud: ${valSalud}\nPension: ${valPension}\narl: ${valArl}`);      let valDeducible = valSalud + valPension + valArl;  ***return*** valDeducible;  } | | |
| **Consola:** | | |
| **Nombre de la función: salud(pvalSalario)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el valor de la salud según un porcentaje del salario | | |
| **Variables:**   1. valSalario 2. valSalud | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  function **salud**(*pvalSalario*) {      let valSalario = pvalSalario;      let valSalud = valSalario \* 0.12;  ***return*** valSalud;  } | | |
| **Nombre de la función: saludExp(pvalSalario)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el valor de la salud según un porcentaje del salario | | |
| **Variables:**   1. valSalario 2. valSalud | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  const **saludExp** = function (*pvalSalario*) {      let valSalario = pvalSalario;      let valSalud = valSalario \* 0.12;  ***return*** valSalud;  } | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: pension(pvalSalario)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el valor de la pensión según un porcentaje del salario | | |
| **Variables:**   1. valSalario 2. valPension | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  function **pension**(*pvalSalario*) {      let valSalario = pvalSalario;      let valPension = valSalario \* 0.16;  ***return*** valPension;  } | | |
| **Nombre de la función: pensionExp(pvalSalario)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el valor de la pensión según un porcentaje del salario | | |
| **Variables:**   1. valSalario 2. valPension | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  const **pensionExp** = function (*pvalSalario*) {      let valSalario = pvalSalario;      let valPension = valSalario \* 0.16;  ***return*** valPension;  } | | |
| **Nombre de la función: arl(pvalSalario)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el valor del arl según un porcentaje del salario | | |
| **Variables:**   1. valSalario 2. valArl | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  function **arl**(*pvalSalario*) {      let valSalario = pvalSalario;      let valArl = valSalario \* 0.052;  ***return*** valArl;  } | | |
| **Nombre de la función: arlExp(pvalSalario)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el valor del arl según un porcentaje del salario | | |
| **Variables:**   1. valSalario 2. valArl | **Tipo de variable:**   1. numérico 2. numérico | |
| **Código:**  const **arlExp** = function (*pvalSalario*) {      let valSalario = pvalSalario;      let valArl = valSalario \* 0.052;  ***return*** valArl;  } | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: pagoTot(pvalSalario, pvalSubTrans, pvalRetencion, pvalDeducible)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el pago total de una persona | | |
| **Variables:**   1. valSalario 2. valSubTrans 3. valRetencion 4. valDeducible 5. valPagoTot | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. numérico | |
| **Código:**  function **pagoTot**(*pvalSalario*, *pvalSubTrans*, *pvalRetencion*, *pvalDeducible*) {      let valSalario = pvalSalario;      let valSubTrans = pvalSubTrans;      let valRetencion = pvalRetencion;      let valDeducible = pvalDeducible;      let valPagoTot = valSalario + valSubTrans - valRetencion - valDeducible;  ***return*** valPagoTot;  } | | |
| **Consola:** | | |
| **Nombre de la función: pagoTotExp(pvalSalario, pvalSubTrans, pvalRetencion, pvalDeducible)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el pago total de una persona | | |
| **Variables:**   1. valSalud 2. valPension 3. valArl 4. valDeducible | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico | |
| **Código:**  const **deducibleExp** = function (*pvalSalario*) {      let valSalud = **salud**(pvalSalario);      let valPension = **pension**(pvalSalario);      let valArl = **arl**(pvalSalario);      console.**log**(`Salud: ${valSalud}\nPension: ${valPension}\narl: ${valArl}`);      let valDeducible = valSalud + valPension + valArl;  ***return*** valDeducible;  } | | |
| **Consola:** | | |

**EJERCICIO 15**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: notas(pnota, pporcent)** | | **Versión: 1.0** |
| **Descripción:**  Función que calcula el porcentaje de una nota | | |
| **Variables:**   1. nota 2. porcent 3. result | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  function **notas**(*pnota*, *pporcent*) {      let nota = pnota;      let porcent = pporcent;      let result = nota \* porcent;  ***return*** result;  } | | |
| **Consola:** | | |
| **Nombre de la función: notasExp(pnota, pporcent)** | | **Versión: 2.0** |
| **Descripción:**  Función que calcula el porcentaje de una nota | | |
| **Variables:**   1. nota 2. porcent 3. result | **Tipo de variable:**   1. numérico 2. numérico 3. numérico | |
| **Código:**  const **notasExp** = function (*pnota*, *pporcent*) {      let nota = pnota;      let porcent = pporcent;      let result = nota \* porcent;  ***return*** result;  } | | |
| **Consola:** | | |
| **Resultado:** | | |

**EJERCICIO 16.1**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnumFin)** | | **Versión: 1.0** |
| **Descripción:**  Función que genera los números del 1 hasta el que sea indicado | | |
| **Variables:**   1. numFin 2. cont 3. nums | **Tipo de variable:**   1. numérico 2. numérico 3. alfanumérico | |
| **Código:**  function **numeros**(*pnumFin*) {      let numFin = pnumFin;      let cont = 0;      let nums = '';  ***while*** (cont < numFin) {          cont += 1;          nums += `${cont}\n`;      }  ***return*** nums;  } | | |
| **Consola:** | | |
| **Nombre de la función: numerosExp(pnumFin)** | | **Versión: 2.0** |
| **Descripción:**  Función que genera los números del 1 hasta el que sea indicado | | |
| **Variables:**   1. numFin 2. cont 3. nums | **Tipo de variable:**   1. numérico 2. numérico 3. alfanumérico | |
| **Código:**  const **numerosExp** = function (*pnumFin*) {      let numFin = pnumFin;      let cont = 0;      let nums = '';  ***while*** (cont < numFin) {          cont += 1;          nums += `${cont}\n`;      }  ***return*** nums;  } | | |
| **Consola:** | | |

**EJERCICIO 16.2**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnumFin)** | | **Versión: 1.0** |
| **Descripción:**  Función que genera los números del 1 hasta el que sea indicado | | |
| **Variables:**   1. numFin 2. cont 3. nums | **Tipo de variable:**   1. numérico 2. numérico 3. alfanumérico | |
| **Código:**  function **numeros**(*pnumFin*) {      let numFin = pnumFin;      let nums = '';  ***for*** (let cont = 1; cont <= numFin; cont++) {          nums += `${cont}\n`;      }  ***return*** nums;  } | | |
| **Consola:** | | |
| **Nombre de la función: numerosExp(pnumFin)** | | **Versión: 2.0** |
| **Descripción:**  Función que genera los números del 1 hasta el que sea indicado | | |
| **Variables:**   1. numFin 2. cont 3. nums | **Tipo de variable:**   1. numérico 2. numérico 3. alfanumérico | |
| **Código:**  const **numerosExp** = function (*pnumFin*) {      let numFin = pnumFin;      let nums = '';  ***for*** (let cont = 1; cont <= numFin; cont++) {          nums += `${cont}\n`;      }  ***return*** nums;  } | | |
| **Consola:** | | |

**EJERCICIO 17.1**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnumFin)** | | **Versión: 1.0** |
| **Descripción:**  Función que genera los números del 1 hasta el que sea indicado diciendo si es par o impar | | |
| **Variables:**   1. numFin 2. cont 3. result | **Tipo de variable:**   1. numérico 2. numérico 3. alfanumérico | |
| **Código:**  function **numeros**(*pnumFin*) {      let numFin = pnumFin;      let cont = 0;      let result = '';  ***while*** (cont < numFin) {          cont = cont + 1;  ***if*** (cont % 2 == 0) {              result += `${cont} es par\n`;          } ***else*** {              result += `${cont} es impar\n`;          }      }  ***return*** result;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numerosExp(pnumFin)** | | **Versión: 2.0** |
| **Descripción:**  Función que genera los números del 1 hasta el que sea indicado diciendo si es par o impar | | |
| **Variables:**   1. numFin 2. cont 3. result | **Tipo de variable:**   1. numérico 2. numérico 3. alfanumérico | |
| **Código:**  const **numerosExp** = function (*pnumFin*) {      let numFin = pnumFin;      let cont = 0;      let result = '';  ***while*** (cont < numFin) {          cont = cont + 1;  ***if*** (cont % 2 == 0) {              result += `${cont} es par\n`;          } ***else*** {              result += `${cont} es impar\n`;          }      }  ***return*** result;  } | | |
| **Consola:** | | |

**EJERCICIO 17.2**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnumFin)** | | **Versión: 1.0** |
| **Descripción:**  Función que genera los números del 1 hasta el que sea indicado diciendo si es par o impar | | |
| **Variables:**   1. numFin 2. cont 3. result | **Tipo de variable:**   1. numérico 2. numérico 3. alfanumérico | |
| **Código:**  function **numeros**(*pnumFin*) {      let numFin = pnumFin;      let result = '';  ***for*** (let cont = 1; cont <= numFin; cont++) {          result += `${cont} `;  ***if*** (cont % 2 == 0) {              result += 'es par\n';          } ***else*** {              result += 'es impar\n';          }      }  ***return*** result;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numerosExp(pnumFin)** | | **Versión: 2.0** |
| **Descripción:**  Función que genera los números del 1 hasta el que sea indicado diciendo si es par o impar | | |
| **Variables:**   1. numFin 2. cont 3. result | **Tipo de variable:**   1. numérico 2. numérico 3. alfanumérico | |
| **Código:**  const **numerosExp** = function (*pnumFin*) {      let numFin = pnumFin;      let result = '';  ***for*** (let cont = 1; cont <= numFin; cont++) {          result += `${cont} `;  ***if*** (cont % 2 == 0) {              result += 'es par\n';          } ***else*** {              result += 'es impar\n';          }      }  ***return*** result;  } | | |
| **Consola:** | | |

**EJERCICIO 18.1**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnum, pnumFin)** | | **Versión: 1.0** |
| **Descripción:**  Función que genera la tabla de multiplicar de un numero e imprimir de la siguiente manera: a \* b = c | | |
| **Variables:**   1. numFin 2. num 3. cont 4. result 5. screen | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. alfanumérico | |
| **Código:**  function **numeros**(*pnum*, *pnumFin*) {      let numFin = pnumFin;      let num = pnum;      let cont = 0;      let result;      let screen = '';  ***while*** (cont < numFin) {          cont += 1;          result = num \* cont;          screen += `${num} \* ${cont} = ${result}\n`;      }  ***return*** screen;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numerosExp(pnum, pnumFin)** | | **Versión: 2.0** |
| **Descripción:**  Función que genera la tabla de multiplicar de un numero e imprimir de la siguiente manera: a \* b = c | | |
| **Variables:**   1. numFin 2. num 3. cont 4. result 5. screen | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. alfanumérico | |
| **Código:**  const **numerosExp** = function (*pnum*, *pnumFin*) {      let numFin = pnumFin;      let num = pnum;      let cont = 0;      let result;      let screen = '';  ***while*** (cont < numFin) {          cont += 1;          result = num \* cont;          screen += `${num} \* ${cont} = ${result}\n`;      }  ***return*** screen;  } | | |
| **Consola:** | | |

**EJERCICIO 18.2**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnum, pnumFin)** | | **Versión: 1.0** |
| **Descripción:**  Función que genera la tabla de multiplicar de un numero e imprimir de la siguiente manera: a \* b = c | | |
| **Variables:**   1. numFin 2. num 3. cont 4. result 5. screen | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. alfanumérico | |
| **Código:**  function **numeros**(*pnum*, *pnumFin*) {      let numFin = pnumFin;      let num = pnum;      let result;      let screen = '';  ***for*** (let cont = 1; cont <= numFin; cont++) {          result = num \* cont;          screen += `${num} \* ${cont} = ${result}\n`;      }  ***return*** screen;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numerosExp(pnum, pnumFin)** | | **Versión: 2.0** |
| **Descripción:**  Función que genera la tabla de multiplicar de un numero e imprimir de la siguiente manera: a \* b = c | | |
| **Variables:**   1. numFin 2. num 3. cont 4. result 5. screen | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. alfanumérico | |
| **Código:**  const **numerosExp** = function (*pnum*, *pnumFin*) {      let numFin = pnumFin;      let num = pnum;      let result;      let screen = '';  ***for*** (let cont = 1; cont <= numFin; cont++) {          result = num \* cont;          screen += `${num} \* ${cont} = ${result}\n`;      }  ***return*** screen;  } | | |
| **Consola:** | | |

**EJERCICIO 19.1**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que genera los múltiplos del nueve hasta el quinto múltiplo y dice si cada numero es par o impar | | |
| **Variables:**   1. num1 2. num2 3. cont 4. result 5. screen | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. alfanumérico | |
| **Código:**  function **numeros**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let cont = 0;      let result;      let screen = '';  ***while*** (cont < num2) {          cont += 1;          result = num1 \* cont;          screen += `${result} `;  ***if*** (result % 2 == 0) {              screen += 'es par\n';          } ***else*** {              screen += 'es impar\n';          }      }  ***return*** screen;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numerosExp(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que genera los múltiplos del nueve hasta el quinto múltiplo y dice si cada numero es par o impar | | |
| **Variables:**   1. num1 2. num2 3. cont 4. result 5. screen | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. alfanumérico | |
| **Código:**  const **numerosExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let cont = 0;      let result;      let screen = '';  ***while*** (cont < num2) {          cont += 1;          result = num1 \* cont;          screen += `${result} `;  ***if*** (result % 2 == 0) {              screen += 'es par\n';          } ***else*** {              screen += 'es impar\n';          }      }  ***return*** screen;  } | | |
| **Consola:** | | |

**EJERCICIO 19.2**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que genera los múltiplos del nueve hasta el quinto múltiplo y dice si cada numero es par o impar | | |
| **Variables:**   1. num1 2. num2 3. cont 4. result 5. screen | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. alfanumérico | |
| **Código:**  function **numeros**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let result;      let screen = '';  ***for*** (let cont = 1; cont <= num2; cont++) {          result = num1 \* cont;          screen += `${result} `;  ***if*** (result % 2 == 0) {              screen += `es par\n`;          } ***else*** {              screen += `es impar\n`;          }      }  ***return*** screen;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numerosExp(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que genera los múltiplos del nueve hasta el quinto múltiplo y dice si cada numero es par o impar | | |
| **Variables:**   1. num1 2. num2 3. cont 4. result 5. screen | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. alfanumérico | |
| **Código:**  const **numerosExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let result;      let screen = '';  ***for*** (let cont = 1; cont <= num2; cont++) {          result = num1 \* cont;          screen += `${result} `;  ***if*** (result % 2 == 0) {              screen += `es par\n`;          } ***else*** {              screen += `es impar\n`;          }      }  ***return*** screen;  } | | |
| **Consola:** | | |

**EJERCICIO 20.1**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que hace la tabla de multiplicar del 1 al 5, multiplicando hasta el 5 cada tabla e imprimirlo de la siguiente manera a \* b = c, y que en frente si es número par poner buzz, de lo contrario poner bass. | | |
| **Variables:**   1. num1 2. num2 3. cont1 4. cont2 5. contar 6. contImpar 7. result 8. print | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. numérico 6. numérico 7. numérico 8. alfanumérico | |
| **Código:**  function **numeros**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let cont1 = 0;      let contPar = 0;      let contImpar = 0;      let print = '';      let result;  ***while*** (cont1 < num1) {          cont1 += 1          let cont2 = 0;  ***while*** (cont2 < num2) {              cont2 += 1;              result = cont1 \* cont2;              print += `${cont1} \* ${cont2} = ${result} `;  ***if*** (result % 2 == 0) {                  print += 'buzz\n';                  contPar += 1;              } ***else*** {                  print += 'bass\n';                  contImpar += 1;              }          }      }      print += `Par: ${contPar}\nImpar: ${contImpar}`;  ***return*** print;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que hace la tabla de multiplicar del 1 al 5, multiplicando hasta el 5 cada tabla e imprimirlo de la siguiente manera a \* b = c, y que en frente si es número par poner buzz, de lo contrario poner bass. | | |
| **Variables:**   1. num1 2. num2 3. cont1 4. cont2 5. contar 6. contImpar 7. result 8. print | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. numérico 6. numérico 7. numérico 8. alfanumérico | |
| **Código:**  const **numerosExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let cont1 = 0;      let contPar = 0;      let contImpar = 0;      let print = '';      let result;  ***while*** (cont1 < num1) {          cont1 += 1          let cont2 = 0;  ***while*** (cont2 < num2) {              cont2 += 1;              result = cont1 \* cont2;              print += `${cont1} \* ${cont2} = ${result} `;  ***if*** (result % 2 == 0) {                  print += 'buzz\n';                  contPar += 1;              } ***else*** {                  print += 'bass\n';                  contImpar += 1;              }          }      }      print += `Par: ${contPar}\nImpar: ${contImpar}`;  ***return*** print;  } | | |
| **Consola:** | | |

**EJERCICIO 20.2**

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnum1, pnum2)** | | **Versión: 1.0** |
| **Descripción:**  Función que hace la tabla de multiplicar del 1 al 5, multiplicando hasta el 5 cada tabla e imprimirlo de la siguiente manera a \* b = c, y que en frente si es número par poner buzz, de lo contrario poner bass. | | |
| **Variables:**   1. num1 2. num2 3. cont1 4. cont2 5. contar 6. contImpar 7. result 8. print | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. numérico 6. numérico 7. numérico 8. alfanumérico | |
| **Código:**  function **numeros**(*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let contPar = 0;      let contImpar = 0;      let print = '';      let result;  ***for*** (let cont1 = 1; cont1 <= num1; cont1++) {  ***for*** (let cont2 = 1; cont2 <= num2; cont2++) {              result = cont1 \* cont2;              print += `${cont1} \* ${cont2} = ${result} `;  ***if*** (result % 2 == 0) {                  print += 'buzz\n';                  contPar += 1;              } ***else*** {                  print += 'bass\n';                  contImpar += 1;              }          }      }      print += `Par: ${contPar}\nImpar: ${contImpar}`;  ***return*** print;  } | | |
| **Consola:** | | |

|  |  |  |
| --- | --- | --- |
| **Nombre de la función: numeros(pnum1, pnum2)** | | **Versión: 2.0** |
| **Descripción:**  Función que hace la tabla de multiplicar del 1 al 5, multiplicando hasta el 5 cada tabla e imprimirlo de la siguiente manera a \* b = c, y que en frente si es número par poner buzz, de lo contrario poner bass. | | |
| **Variables:**   1. num1 2. num2 3. cont1 4. cont2 5. contar 6. contImpar 7. result 8. print | **Tipo de variable:**   1. numérico 2. numérico 3. numérico 4. numérico 5. numérico 6. numérico 7. numérico 8. alfanumérico | |
| **Código:**  const **numerosExp** = function (*pnum1*, *pnum2*) {      let num1 = pnum1;      let num2 = pnum2;      let contPar = 0;      let contImpar = 0;      let print = '';      let result;  ***for*** (let cont1 = 1; cont1 <= num1; cont1++) {  ***for*** (let cont2 = 1; cont2 <= num2; cont2++) {              result = cont1 \* cont2;              print += `${cont1} \* ${cont2} = ${result} `;  ***if*** (result % 2 == 0) {                  print += 'buzz\n';                  contPar += 1;              } ***else*** {                  print += 'bass\n';                  contImpar += 1;              }          }      }      print += `Par: ${contPar}\nImpar: ${contImpar}`;  ***return*** print;  } | | |
| **Consola:** | | |