2. **presentación:**

esta monografía presenta diferentes programas el cual enseña el manejo de los arrays en JavaScript, se presentarán diferentes funciones y varias formas en que estas se pueden utilizar.



La siguiente función permitirá organizar de manera más optima el código de modo que si el texto tiene el símbolo ( : : ) dibujara varios iguales antes de colocar los datos que hay en la cadena, además también cuenta con una variable llamada lineas\_post el cual permite dar el número de saltos de líneas que uno quiera.

**AUTOR: ALEXANDER GONZALEZ ORTIZ**

**CÓDIGO: 1004681679**

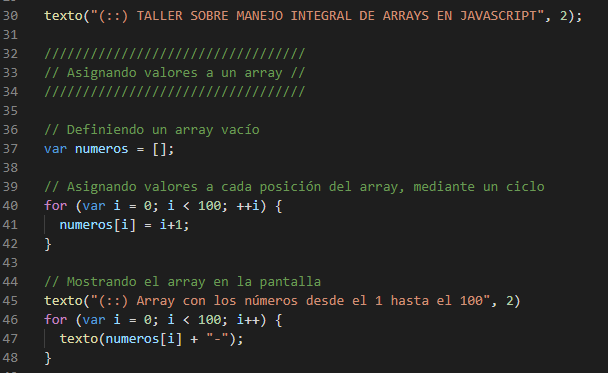
**CORREO:** [**alexander.gonzalez1@utp.edu.co**](mailto:alexander.gonzalez1@utp.edu.co)

**GITHUB:** [**https://github.com/AGO2001/tercera\_previa/tree/main/matrices**](https://github.com/AGO2001/tercera_previa/tree/main/matrices)

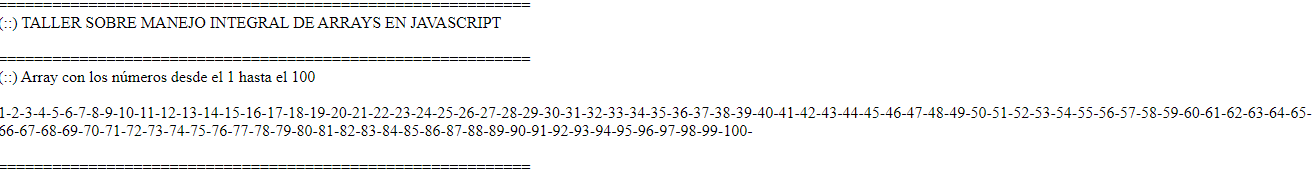
**3. ARRAYS PRIMERA PARTE:**

**3.1 ARRAYS CON LOS NUMEROS DEL 1 AL 100**

En este programa se crea un array vacío llamado números. Mediante un ciclo para se define una variable i el cual va a representar la posición que va a tener el numero en el array vacío, y al final mediante otro ciclo para se van mostrando los números introducidos en el array. A continuación se muestra el código

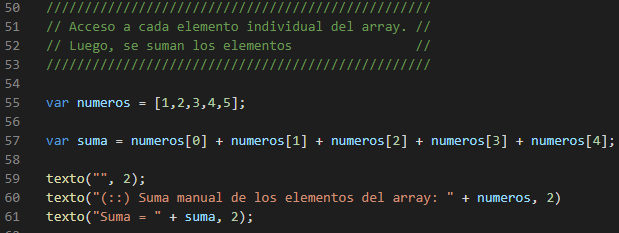


Y así se vería el código al ejecutarlo

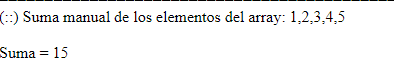


**3.2 SUMAR ELEMENTOS DE UN ARRAY**

En el siguiente código se crea un array con números del 1 al 5 y se suman los elementos de dicho array (var suma = numero [0] + numero [1] . . .) y luego se muestra en pantalla el resultado de la sumatoria, a continuación, se muestra el código.

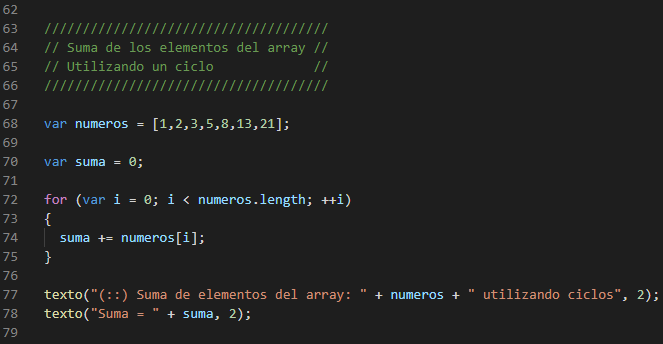


Y así se vería al ejecutarlo.

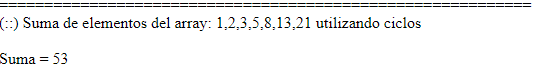


**3.3 SUMAR ELEMENTOS DE UN ARRAY CON CICLOS**

El siguiente programa cumple con la misma función que el anterior, pero en vez de sumar manualmente cada elemento del array, se utiliza un ciclo para repetir el proceso, el código es el siguiente.

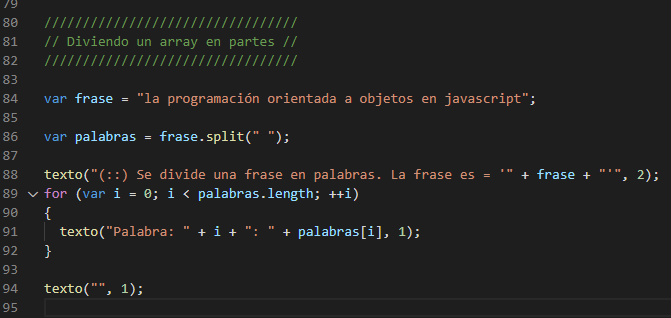


Y así se vería al ejecutarlo.

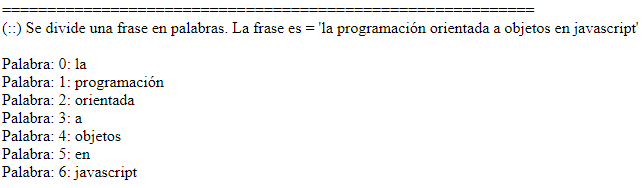


**3.4 DIVIDIR UN ARRAY EN PARTES**

El siguiente programa divide lo que haya adentro de un array en varias partes, por ejemplo, nos dan un array que dice “hola mundo”, mediante la función Split divide la palabra en dos partes, “hola” “mundo”. A continuación, se mostrará el código .

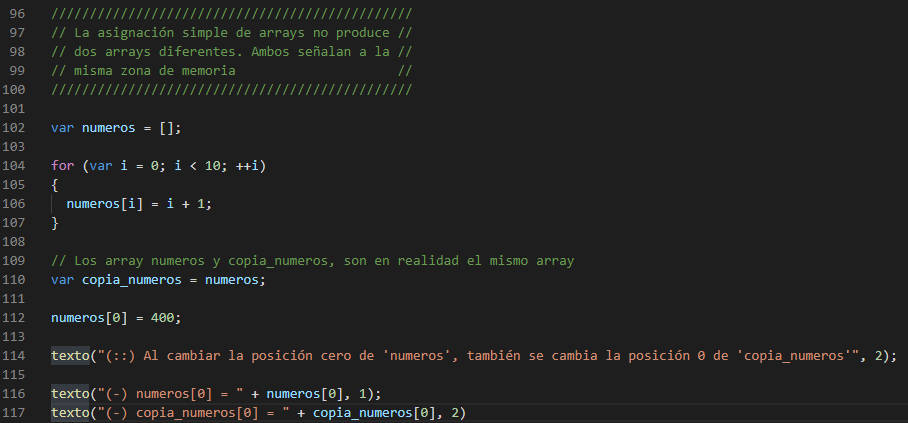


Así vería el código al ejecutarse.

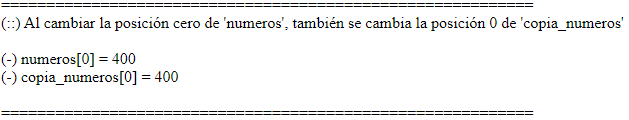


**3.5 COPIAR NUMEROS**

El siguiente código crea una copia de un array, pero ambos arrays se guardan en una misma memoria provocando que se modifique tanto el array el original como la copia. A continuación, se muestra el código.

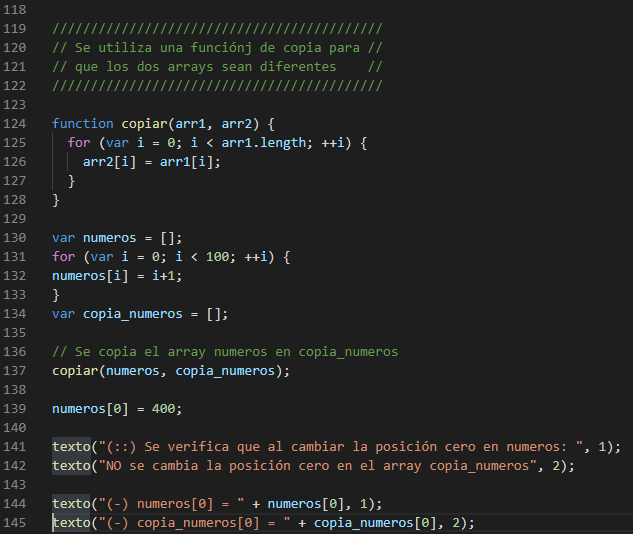


Así se vería el código al ejecutarse.

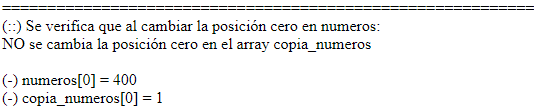


**3.6 COPIAR NUMEROS SIN MODIFICAR EL SEGUNDO ARRAY**

Este código permite copiar un el contenido de un array a otro sin el riesgo a que el contenido del segundo rey se modifique al cambiar algún dato del primero.

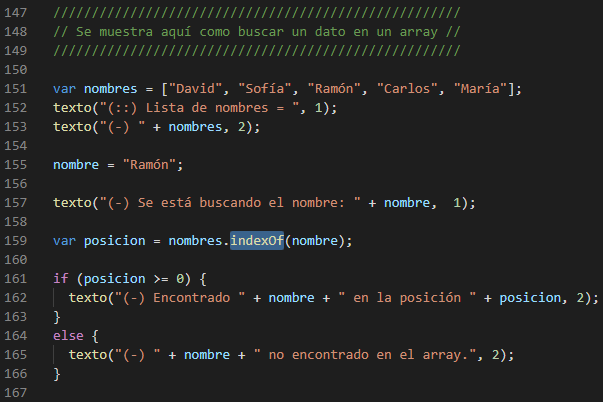


Así ve el código al ejecutarse.

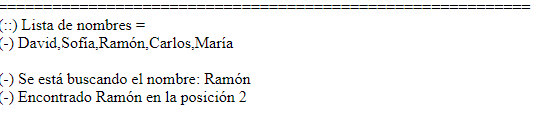


**3.7 BUSCAR UN DATO EN UN ARRAY**

El siguiente código permite buscar un dato dentro de un array teniendo en cuenta la posición en la que está dicho dato.

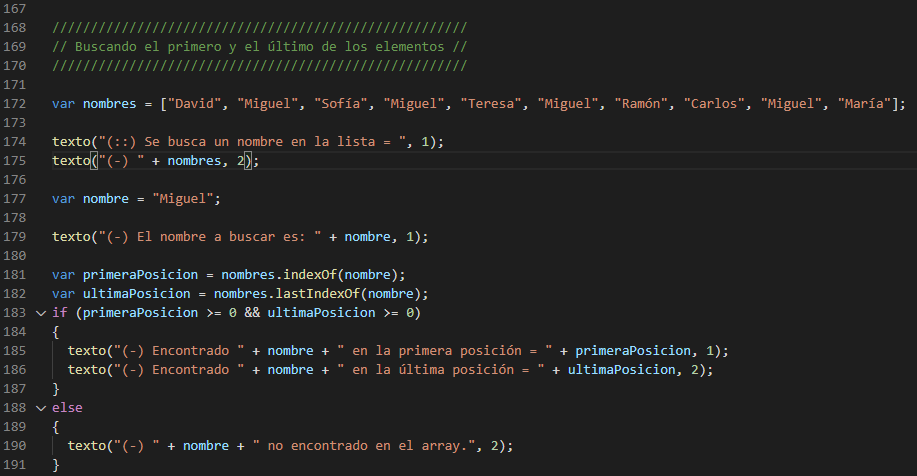


Se utiliza la función indexOf para buscar la posición del nombre que se quiere encontrar, y con un si entonces se dice si el nombre se encuentra incluida en la función o no. Así ve el código al ejecutarse.

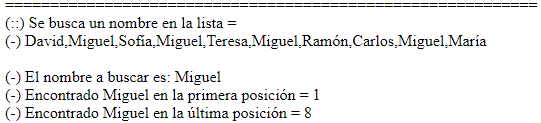


**3.8 BUSCAR EL PRIMER Y SEGUNDO DATO DE UN ARRAY**

El siguiente código permite buscar la primera y última vez que un dato se repite

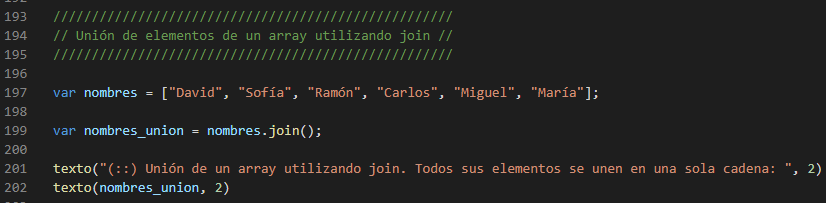


Mediante las funciones indexOf y lastIndexOf se puede buscar el primer y último dato de un array como se muestra en el código, así se ve al ejecutarse.

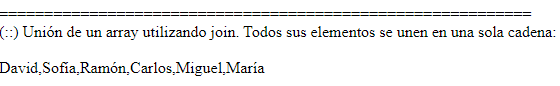


**3.9 UNION DE UN ARRAY UTILIZANDO JOIN**

En este código se utiliza la una función llamada join el cual sirve para unir los elementos de un array

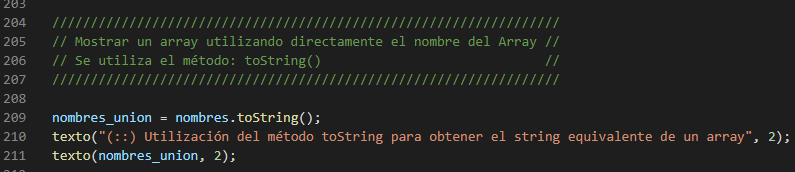


Su función es el de convertir el array en una sola cadena, como se muestra a continuación.

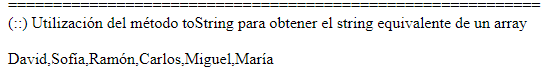


**3.10 UNION DE UN ARRAY UTILIZANDO TOSTRING**

Su función es igual a la de join. Con la diferencia de que trata cada elemento de la cadena como si fuera un elemento de tipo string

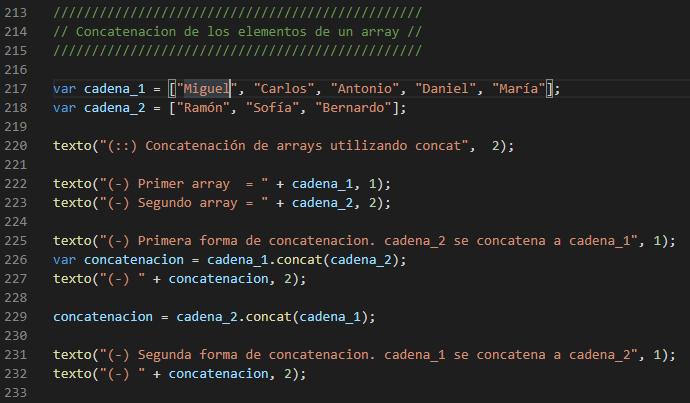


Así se ve el código al ejecutarse.

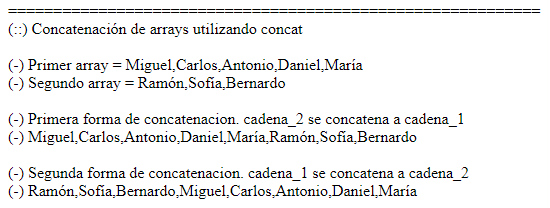


**3.11 FUNCION CONCAT**

El siguiente código permite unir un array a otro mediante la función concat



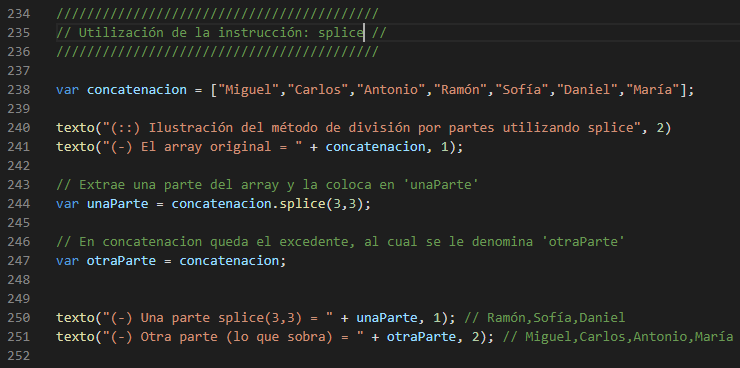
En el código se puede ver que se crean dos cadenas y mediante la función concat se puede unir de dos maneras.

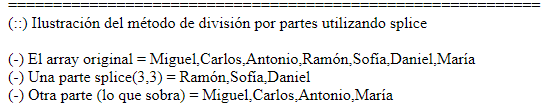


Cadena dos se pude concatenar con la cadena uno y la cadena uno se puede concatenar con la cadena dos.

**3.12 FUNCION SPLICE**

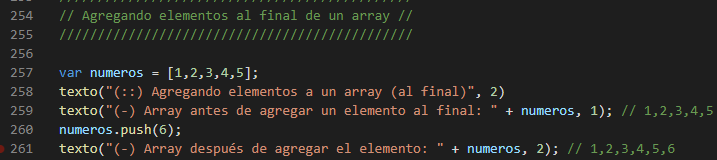
Esta función permite quitar una parte cualquiera de un array.

En el código se ve que se crea un array con varios nombres. La variable llamada unaparte cumple la función de quitar los nombres de la posición 3,3, o sea que va a quitar a Ramón, Sofia y Daniela de esa lista. Luego se crea otra variable llamada otraparte el cual va ser igual al array original. Pero ahora va aparecer solo con los nombres que no fueron removidos del array

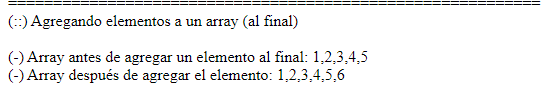


**3.13 FUNCION PUSH**

La siguiente función permite agregar uno o más elementos de un array al final de esta.

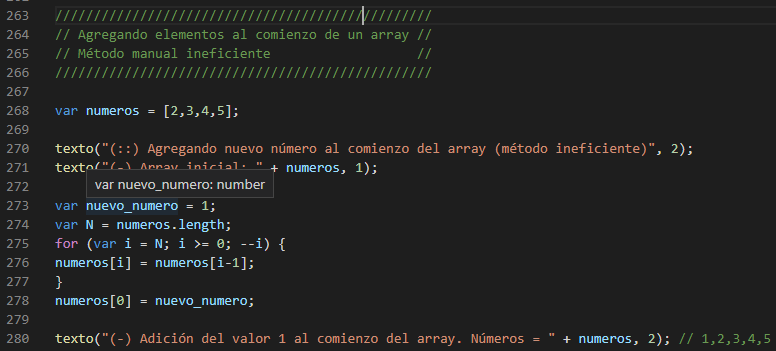


Así ve el código al ejecutarse.

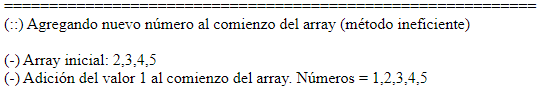


**3.14 AGREGANDO NUEVO ELEMENTO AL COMIENZO DEL ARRAY (METODO INEFICIENTE**

El siguiente código muestra un método para agregar un nuevo elemento al comienzo del array, sin embargo, este es un método ineficiente de hacerlo.

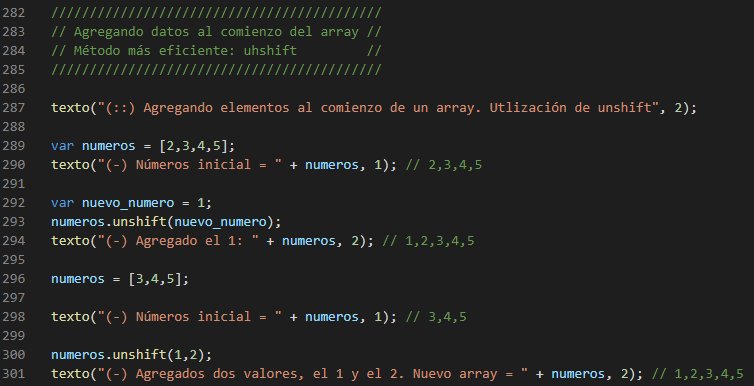


Como se ve en el código para incluir un nuevo elemento al comienzo del array se debe tomar la longitud del array para poder darle un espacio extra al inicio e ir poniendo nuevamente uno por uno cada elemento del array mediante un ciclo. Así se ve el código después de ser ejecutado.

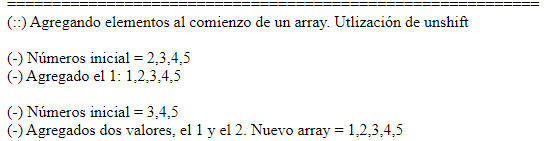


**3.15 FUNCION UNSHIFT**

Mediante esta función se pueden agregar uno o más elementos al comienzo de un array sin necesidad de utilizar el método anterior.

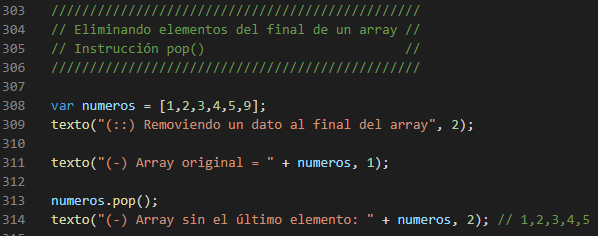


Así se ve el código al ejecutarse.

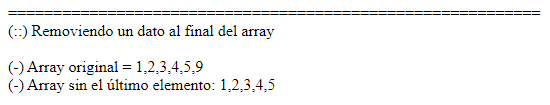


**3.16 FUNCION POP**

Este código permite quitar un elemento al final de un array mediante la función pop.

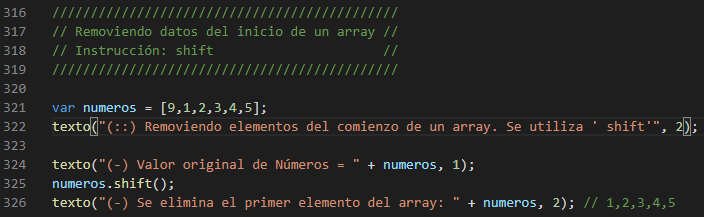


Así se ve el código al ejecutarse.

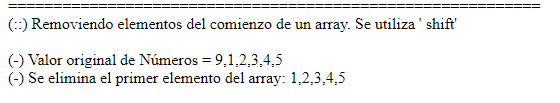


**3.17 FUNCION SHIFT**

La siguiente función permite quitar el primer elemento del array.

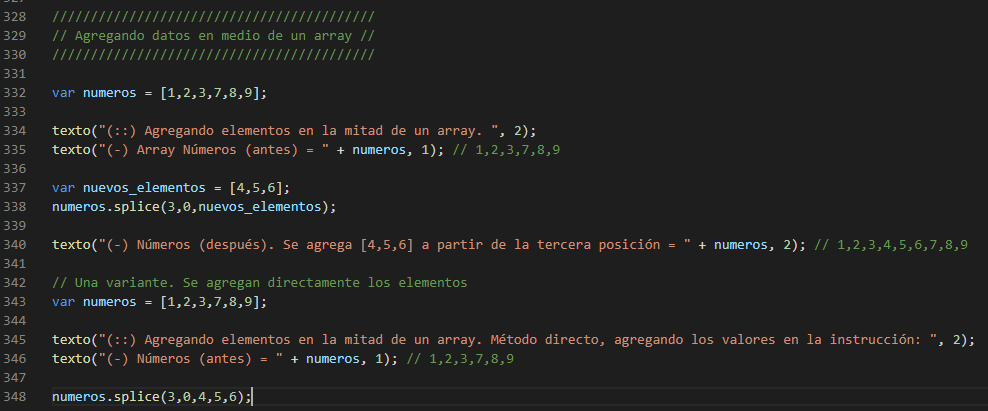


Así se ve el código al ejecutarse.

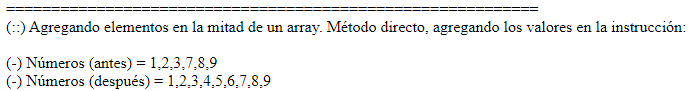


**3.18 AGREGANDO ELEMENTOS A LA MITAD DE UN ARRAY CON LA FUNCION SPLICE**

Aunque anterior mente se mostró que esta función quita elementos de un array, también se puede utilizar para agregar elementos en cualquier parte del array.

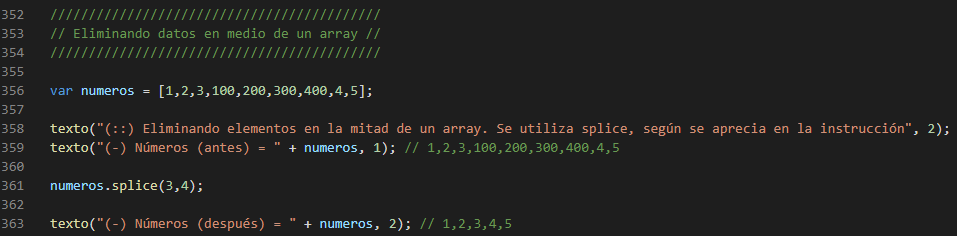


En el código se puede apreciar las dos maneras en las que se pude usar esta función.

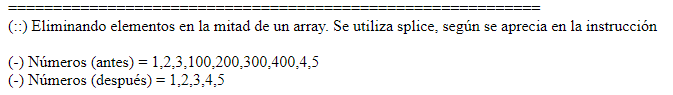


**3.19 ELIMINANDO ELEMENTOS DE UN ARRAY CON SPLICE**

Splice también se puede utilizar para eliminar uno o más elementos de un array

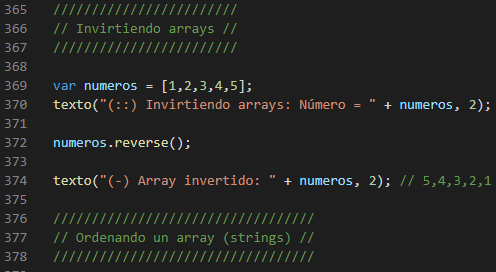


Así se ve el código al ejecutarse.

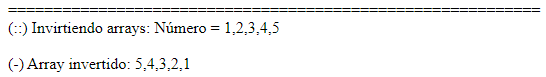


**3.20 FUNCION REVERSE**

Este código cumple la función de tomar un array e invertirla, por ejemplo, se tiene el array “1, 2, 3, 4, 5” al utilizar la función lanza lo siguiente “5, 4, 3, 2, 1”.

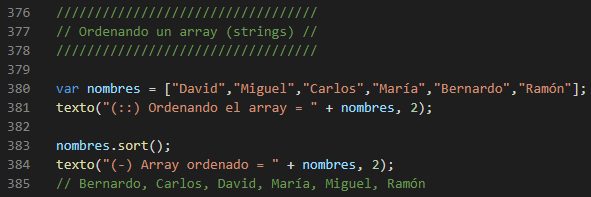


Así se ve el código al ejecutarse.

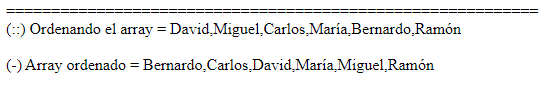


**3.21 FUNCION SORT**

Esta función permite ordenar un array desde la letra a hasta la z.

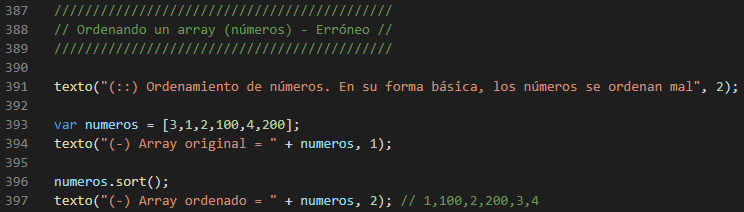


Así se ve el código al ejecutarse.

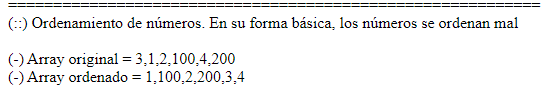


**3.22 ORDENAR UN ARRAY DE NUMEROS CON SORT (FORMA ERRONEA)**

La función sort también sirve para ordenar números, sin embargo, la forma en que se muestra a continuación es una forma errónea de hacerlos ya que solo toma en cuenta el primer digito de cada número.

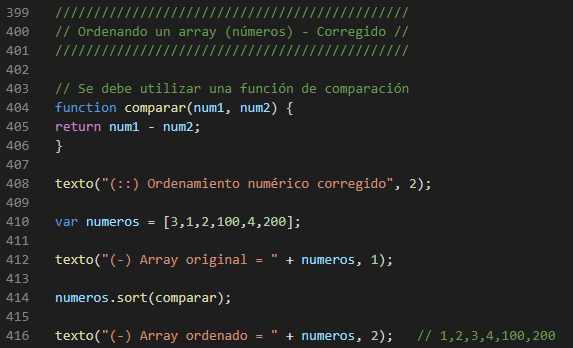


Así se ve le código al ejecutarse.

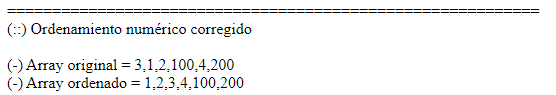


**3.23 ORDENAR UN ARRAY DE NUMEROS CON SORT (FORMA CORRECTA)**

Este código presenta la forma correcta de ordenar un array de números al crear una función que permite comparar un numero de otro.

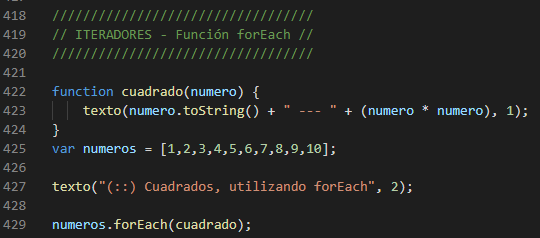


Así ve el código al ejecutarse.

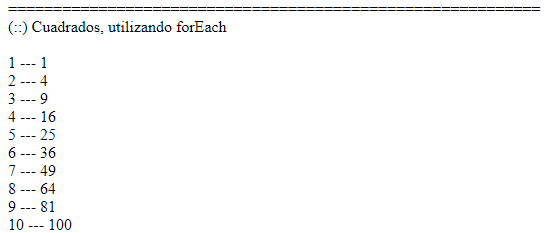


**3.24 ITERADOR FOREACH**

La siguiente función se puede decir que funciona a manera de ciclo ya que evalúa el contenido de un vector y lo va recorriendo, y según lo que haya entra al llamado de la función y realiza los procedimientos que estén dentro de la función para cada elemento del vector.

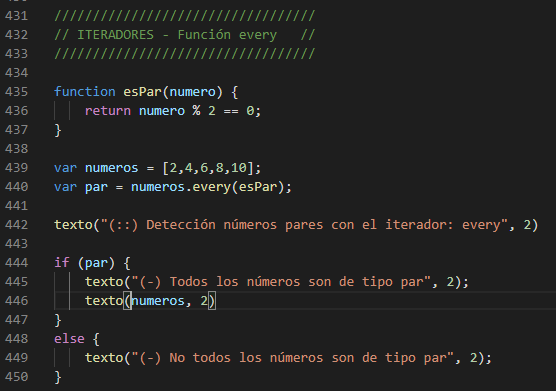


En el código también se puede ver que se utiliza una función ya vista antes para visualizar el número y su cuadrado. Así se ve el código al ejecutarse.

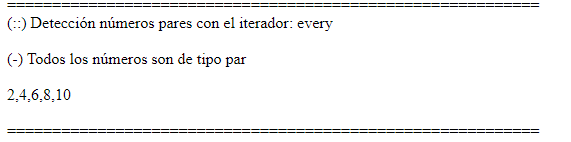


**3.25 ITERADOR EVERY**

Esta función es un iterador el cual evalúa cada elemento de un vector y devuelve un valor booleano al realizar los procedimientos que hallan dentro de la función.

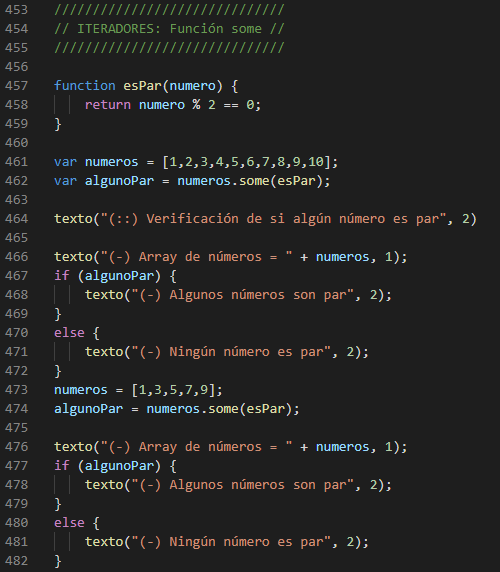


Este código evalúa si los elementos del vector son pares o no, si cada elemento del vector es igual a cero al dividirlo entre dos entonces la función es par retorna un true o verdadero lo que quiere decir que cada elemento del vector es par, pero si retorna un false o falso, quiere decir que no todos los elementos del vector son pares. Así se ve el código al ejecutarse

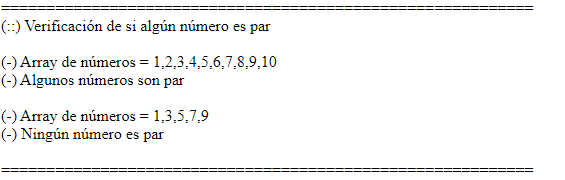


**3.26 ITERADOR SOME**

A diferencia del iterador every, este iterador devuelve un true si al menos alguno de los elementos del vector cumple con la condición de la función y devuelve un false si ninguno de los elementos del vector cumple con la condición.

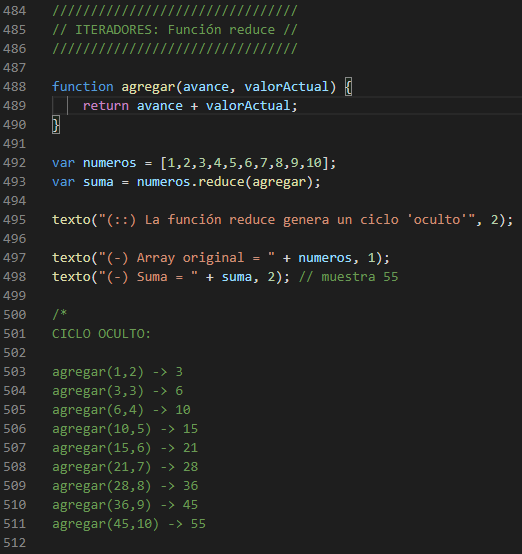


Así se ve el código al ejecutarse.

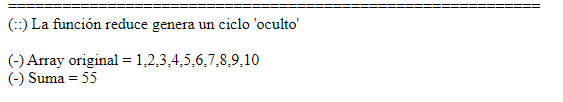


**3.27 FUNCION REDUCE**

La función reduce como su nombre lo dice, reduce un vector a un solo valor ya sea que la función este sumando o restando o haciendo cualquier otra operación.

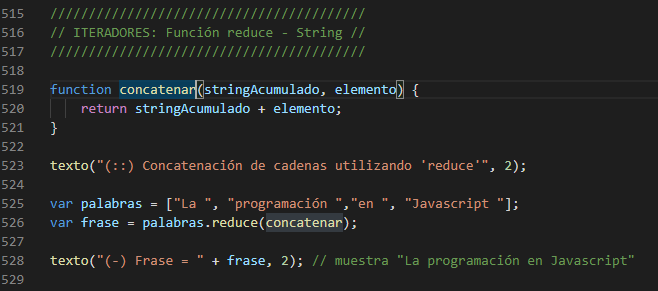


La función tiene dos valores de entrada, estos valores al utilizar reduce se convierten inicial mente en los valores de la posición 0 y 1 del vector y conforme va avanzando el procedimiento el valor actual va pasando de un valor a otro hasta llegar al valor nulo, también se puede apreciar en el código un ciclo oculto que son los resultados de cada suma.

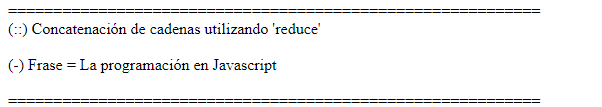


**3.28 CONCATENACION DE CADENAS CON REDUCE**

Reduce no solo sirve para reducir un vector de números, este también puede convertir varias palabras en una cadena.

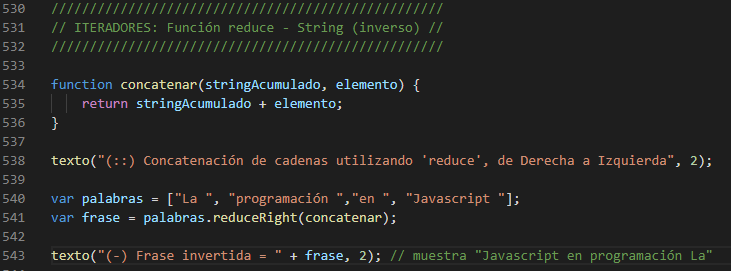


En el codigo se puede apreciar el vector de palabras y como reduce las une en una sola cadena.

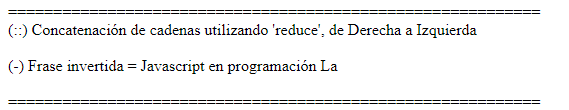


**3.29 CONCATENACION INVERTIDA DE CADENAS CON REDUCE**

La siguiente función concatena una cadena, pero invertida con la función reduceright.

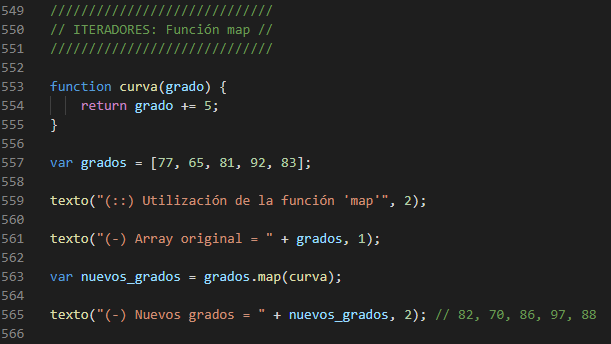


Esta función empieza a concatenar cada palabra desde la última posición de vector hasta la primera como se muestra en el código.

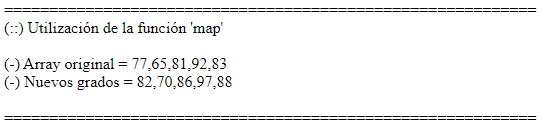


**3.30 FUNCION MAP**

La función map permite modificar cada elemento de un vector (numérico) dando como resultado un nuevo array.

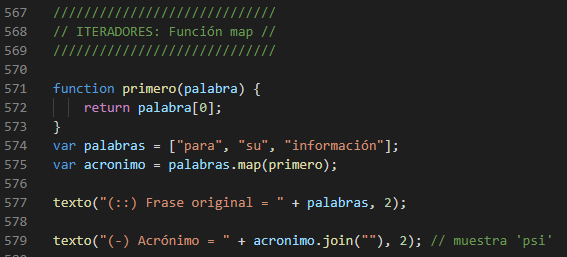


En el código se pude ver que nos dan un array de números llamados grados, mediante la funcion map, se puede recorrer cada uno de los elementos y de esta manera cambiar el valor de cada elemento.

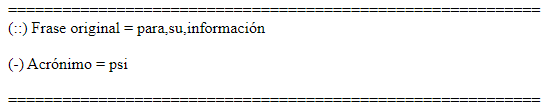


**3.31 CREANDO ACRONIMOS CON MAP**

Al utilizar esta función en un vector de palabras permite sacar una letra de cualquier posición de cada palabra.

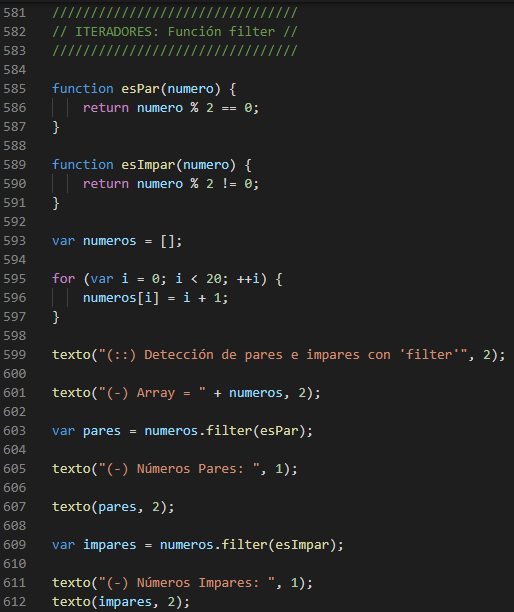


Esta función evalúa cada palabra i devuelve las letras que se encuentren en la posición 0 de cada una creando así el acrónimo, adicional mente se utiliza también la función jon con el objetivo de que el acrónimo no salgan separados por comas.

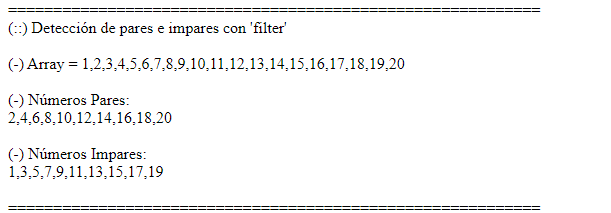


**3.32 FUNCION FILTER**

La función filter permite sacar un conjunto de elementos que cumplan una condición especifica.

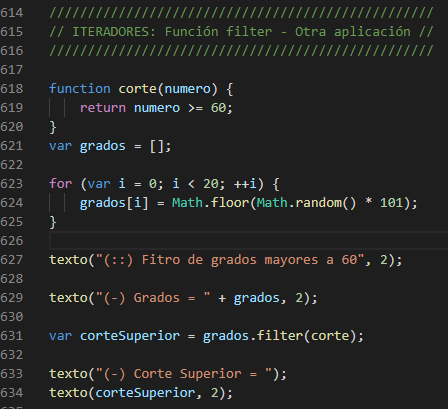


Como se puede ver en el código la función filter evalúa dos funciones llamadas espar y esimpar y devuelve dos cadenas, una con los números que son pares y otra con los números que son impares.

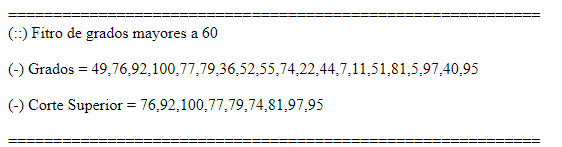


**3.33 OTRA APLICACION PARA FILTER**

La función filter también puede comparar si un número es mayor menor o igual a otro.

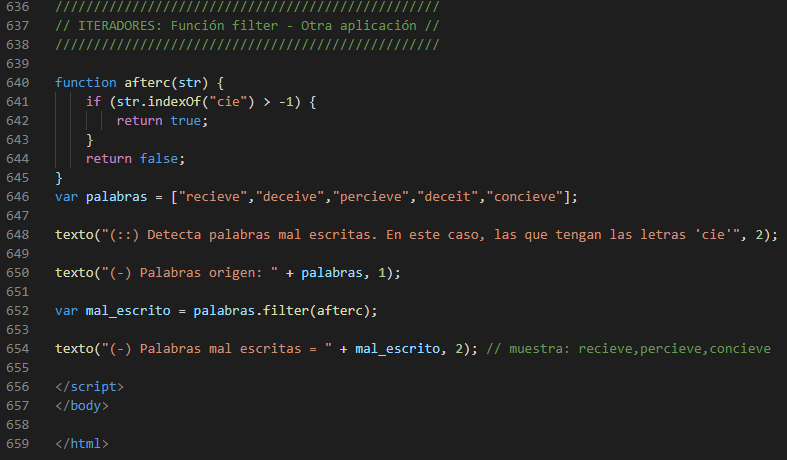


El código crea un vector de números de forma aleatoria para seguidamente buscar a aquellos que sean mayores o iguales a 60.

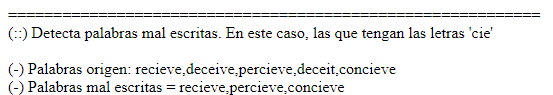


**3.34 OTRA APLICACION MAS PARA FILTER**

Otra aplicación para esta función es buscar palabras que estén mal escritas.



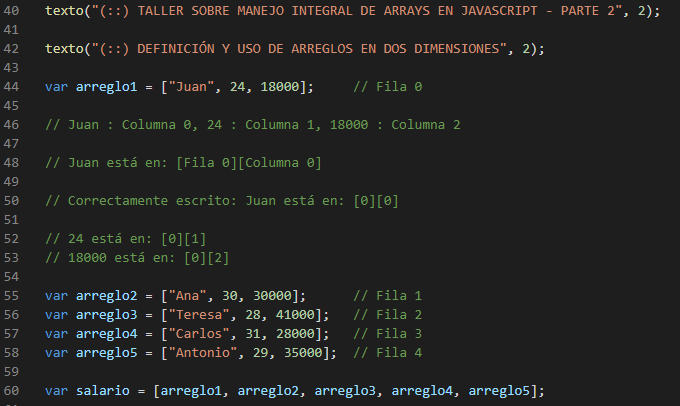
Como se puede ver en el código la función evalúa la cadena cie el cual se encuentran en las palabras del vector, si devuelve true pone la palabra en la variable llamada mal-escrito, si no devuelve nada entonces no la pone.



**4. ARRAYS SEGUNDA PARTE:**

**4.1 DEFINICION Y ARREGLOS DE DOS DIMENCIONES**

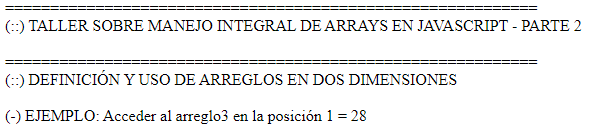
Esta segunda parte trata sobre el manejo de matrices el cual permite maneja información según un numero n de filas y columnas. En otras palabras, las filas son la posición en donde se encuentra dicha información y las columnas lo que contiene. Para entenderlo mejor veamos un ejemplo.



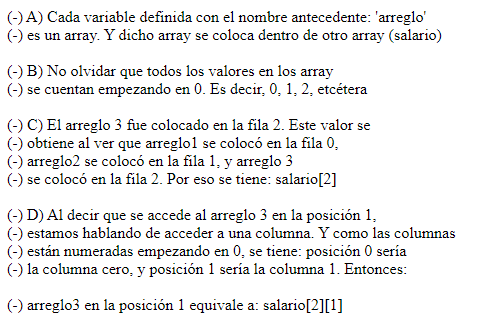
El siguiente código tenemos definidas 5 variables, cada uno con 3 columnas. Debajo de estas variables se puede ver otra llamada salario, en esta variable cada arreglo es una fila y las columnas representan la información que hay en esas filas.



La línea de código que se ve a continuación, muestra en pantalla la información en salario que se encuentra en la fila 2, columnas 1. tener en cuenta que tanto en filas como en columnas la posición empieza desde 0, así se ve el código al ejecutarse.

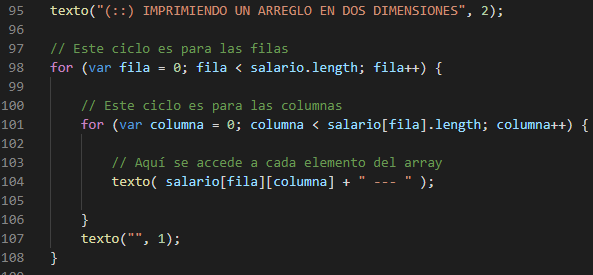


La siguiente parte de código es una explicación más detalla sobre los arreglos.

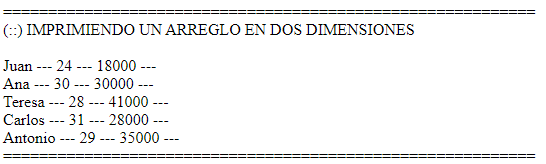


**4.2 IMPRIMIR UN ARREGLOS EN DOS DIMENSIOES**

El siguiente código utiliza un ciclo para con el fin de imprimir en pantalla cada uno de los arreglos.



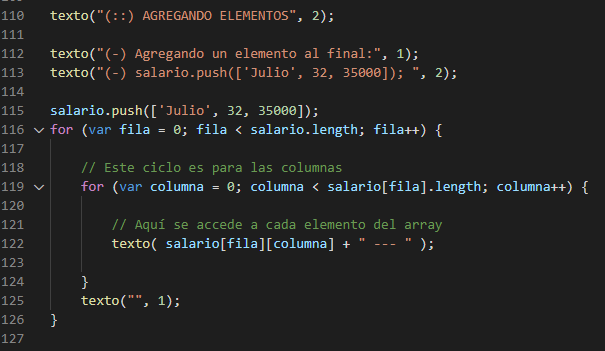
El primer ciclo para indica en que fila se encuentra salario y el segundo ciclo indica la columna, una vez que la variable columna es mayor que la longitud de salario en fila, se aumenta la posición de fila en uno e imprime la información que halla en esa posición hasta que columnas vuelva a ser mayor que el tamaño en fila, el código termina de ejecutarse cuando fila es mayor que el tamaño del salario. Así se ve le código al ejecutarse.



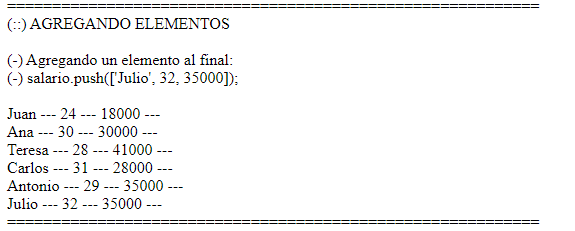
Ahí se puede ver que cada arreglo representa una fila, y la información que hay dentro de cada arreglo representa una columna.

**4.3 AGREGANDO ELEMENTOS A UN ARREGLO**

El siguiente código utiliza la función push para agregar un arreglo más al salario.

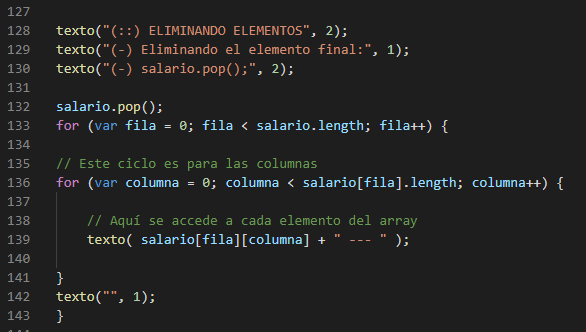


Al igual que antes, una vez se agrega el nuevo arreglo a la variable salario, se utiliza los ciclos para para ir imprimiendo la información.

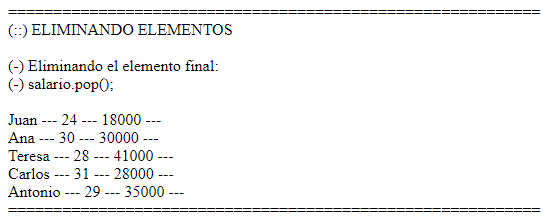


**4.4 ELIMINANDO ELEMENTOS DE UN ARREGLO**

A diferencia de push que agrega un arreglo al final de la matriz, la función pop quita el ultimo arreglo que este en la matriz.

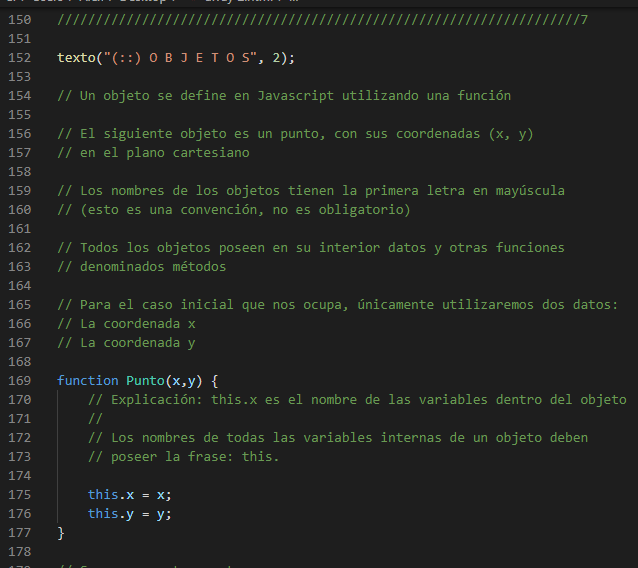


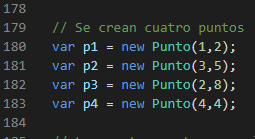
Una vez que se utiliza la función pop para quitar el ultimo arreglo, se vuelve a utilizar los ciclos para, para verificar que se haya quitado dicho elemento.



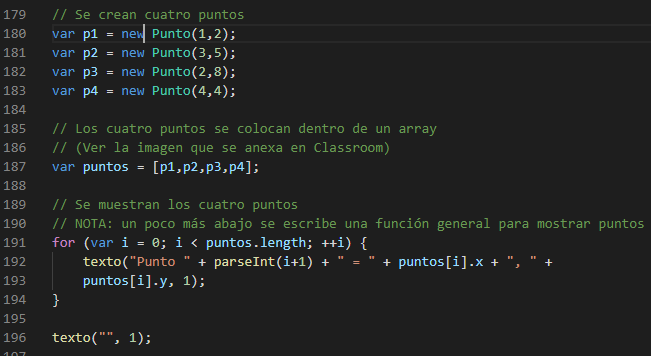
**4.5 OBJETOS FUNCION THIS**

Un objeto en JavaScript es prácticamente una función que adentro tiene datos y otras funciones llamadas métodos, en este caso solo vamos a centrarnos en dos datos, X y Y el cual con la ayuda de la función this se le pueden asignar diferentes valores.

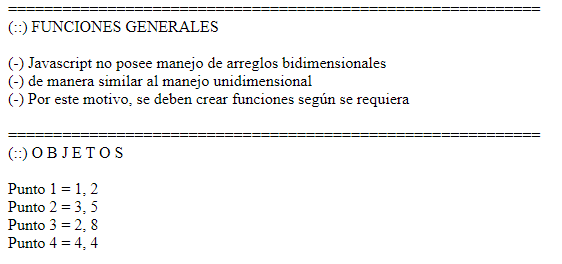




En el código se define una función llamada punto el cual con la ayuda de la función this se puede almacenar más de un dato en los putos X y Y, se puede observar que se crean cuatro variables en el cual se asignan diferentes puntos a la misma función, la función new permite que el código distinga entre un punto y otro.

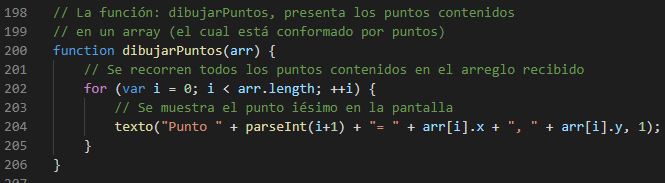


Aquí se crea un array llamada puntos el cual posterior mente se muestran con un ciclo para, aquí la manera en que se imprime los puntos es muy diferente a la de una matriz ya que hay que tener en cuenta que las variables X y Y son independientes por lo que se deben poner de forma separada como se muestra en el código.

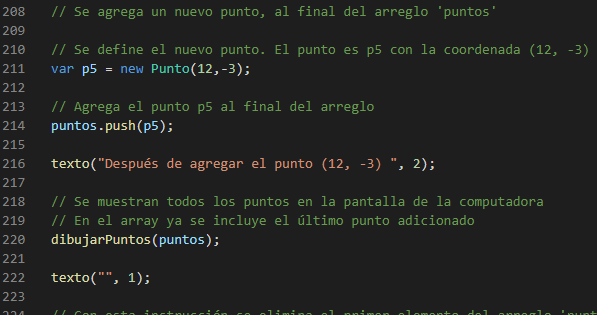


**4.5 AGREGAR Y QUITAR PUNTOS**

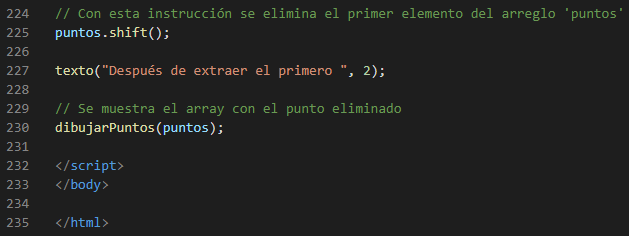
Para el siguiente código se crea una función llamada dibujar puntos para no tener que dibujar nuevamente el ciclo para cada vez que se requiera.



El siguiente código utiliza la función push para agregar un punto más a la función puntos.



Recordar que la función push agrega un punto siempre al final del array. Para quitar un punto al final del array se utiliza la función shift.



Así se ve el código después de ejecutarse.

