JUGANDO CON ARRAYS

PROYECTO INTEGRADO

DESARROLLO WEB EN ENTORNO CLIENTE

ALUMNO: *Alberto García Sola*

CURSO: *2 DAW*

ÍNDICE

¿EN QUÉ CONSISTE Y A QUIÉN VA DIRIGIDA?

Jugando con arrays es una aplicación web que va dirigida a programadores con cualquier experiencia, ya sean juniors, semis o seniors.

Consiste en una interfaz gráfica a través de la cual podemos crear, modificar y eliminar matrices de manera gráfica.

OBJETIVO

El objetivo de la aplicación web es que podamos manejar matrices y observar como se comportan sin escribir una línea de código. Es por esto que va dirigida a cualquier perfil de programador, por ejemplo a un junior le puede venir bien para observar como se comportan las matrices y como se alteran dependiendo de que método seleccione a modo de aprendizaje, mientras que a un senior le puede venir bien por ejemplo para realizar una prueba rápida y ver el resultado sin necesidad de ponerse a escribir el código a mano.

¿CÓMO SE USA?

Jugando con arrays nos permite:

- CREAR MATRICES
    - Escribiendo el nombre de la matriz y los elementos deseados separados por comas en los campos de texto adecuados y haciendo clic en el botón **AÑADIR**.
- MODIFICAR MATRICES
    - Seleccionando la matriz y el método deseado en las lista desplegables y haciendo clic en el botón **MUTAR**.
- ELIMINAR MATRICES
    - Seleccionando la matriz deseada en la lista desplegable y haciendo clic en el botón **ELIMINAR**.
- OBTENER AYUDA DE UN MÉTODO
    - Haciendo clic sobre la etiqueta **MÉTODO** de color azul, habiendo seleccionado previamente el método en cuestión.

¿CÓMO FUNCIONA?

La mayor parte de la funcionalidad de la aplicación web recae sobre el objeto **Arrays**, en el cual tenemos todas las matrices que se van generando durante el uso de la aplicación.

Cuando un usuario añade una matriz lo que pasa en el programa es que se añaden el nombre de la matriz, sus elementos y su longitud en las propiedades **noms**, **arrs** y **lengths** del objeto **Arrays**. Como dichas propiedades son matrices, después de que el usuario haya introducido varias matrices y queramos trabajar con una de ellas, para referirnos por ejemplo a la primera matriz que introdujo el usuario necesitaríamos recoger el nombre de la matriz(Arrays.noms[0]), los elementos de la matriz(Arrays.arrs[0]), y la longitud de la matriz(Arrays.lengths[0]).

Cuando un usuario elimina una matriz primero se localiza la posición que hace referencia a todas las propiedades de la matriz seleccionada, como por ejemplo el nombre de la matriz(Arrays.noms[3]), los elementos de la matriz(Arrays.arrs[3]), y la longitud de la matriz(Arrays.lengths[3]). Una vez sabemos la posición que hace referencia a la matriz seleccionada por el usuario(en este caso la 3), como las propiedades del objeto Arrays son matrices, basta con eliminar dicha posición.

Cuando un usuario ejecuta un método se ejecuta la función **metodoArr()**, la cual está formada principalmente por una estructura *switch case* en la que cada *case* es un método de los que puede seleccionar el usuario, una vez dentro del *case* adecuado se recupera la matriz seleccionada por el usuario de la misma manera en la que se explica arriba para poder trabajar con ella y se ejecuta el código correspondiente al método seleccionado por el usuario.

Explicaré con detalle las propiedades y métodos que contiene el objeto Arrays en el apartado REQUISITOS posteriormente.

REQUISITOS

OBJETOS *NAVIGATOR* Y *SCREEN*

Uso del **objeto navigator** en la línea 594 de código del archivo arrays.js

El objeto navigator es usado para obtener el lenguaje del navegador mediante navigator.language, una vez obtenido el lenguaje, si está en español o inglés, el lenguaje de las etiquetas y botones principales de la aplicación web cambia al idioma del navegador automáticamente.

Uso del **objeto screen** en la línea 670 de código del archivo arrays.js

El objeto screen es usado para obtener el ancho de la pantalla del dispositivo desde el cual se visita la aplicación web y así advertir a los usuarios que entren a la aplicación web desde un dispositivo móvil de que dicha web no está optimizada para dispositivos móviles. Advierto a los usuarios con una anchura menor a 400 píxeles ya que la mayoría de móviles tienen un ancho entre los 300 y 380 píxeles.

VENTANAS SECUNDARIAS CON COMUNICACIÓN ENTRE ELLAS

En la línea 648 de código del archivo arrays.js podemos ver la función encargada de abrir una nueva ventana, mientras que en la linea 372 de código del archivo methods.html podemos observar como la ventana hija recupera información de la ventana padre. Desde la ventana secundaria se recupera el método seleccionado por el usuario y se muestra la explicación y ejemplo de como se usa el método.

ARRAYS BIDIMENSIONALES

La propiedad **arrs** del objeto Arrays es una matriz bidimensional, ya que en cada elemento de la matriz introduzco otra matriz con los elementos indicados por el usuario. Dicha propiedad empieza lógicamente estando vacía y se va rellenando a medida que el usuario va introduciendo matrices. Podemos encontrar la propiedad arrs en la línea 3 del archivo arrays.js.

También hay una segunda matriz bidimensional que corresponde con la propiedad **history** del objeto Arrays, la cual almacena en cada posición otra matriz con el nombre de los métodos que se han ejecutado para la matriz de esa posición en concreto. Propiedad history en la línea 5 del archivo arrays.js.

FUNCIONES

ARCHIVO	FUNCIÓN	DESCRIPCIÓN	LÍNEA
arrays.js	addArr()	Añade las matrices que introduces el usuario al objeto Arrays y va rellenando las listas desplegables para seleccionar las matrices.	53
	fillSelectArr2()	Encargada de rellenar la segunda lista desplegable que se usa para el método concat().	98
	changeListArr()	Muestra los elementos y longitud correspondientes a la matriz seleccionada por el usuario, tanto para la lista de matrices principal como para la lista de matrices secundaria(la que solo se usa para el método concat()), dependiendo de si le pasamos un parámetro o no.	108
	changeMethod()	Esconde o muestra determinados elementos HTML dependiendo del método seleccionado, por ejemplo para el método pop() se esconde el input donde se introducen argumentos, porque el método pop() no necesita argumentos.	121
	deleteArr()	Encargada de eliminar la matriz seleccionada por el usuario, tanto del objeto Arrays como de las listas seleccionables.	176
	metodoArr()	Es la función más grande, es la que se encarga de ejecutar el método que haya seleccionado el usuario, está construida con un gran switch case.	205
	printCodeHTML()	Se encarga de representar el código real que habría que escribir en JavaScript para obtener el resultado que hemos obtenido de manera gráfica. Esta función recibe el código que tiene que representar desde la función createCodeHTML(). El código se representa debajo del título **OBSERVA TU CÓDIGO.**	479

	createCodeHTML()	Encargada de crear el código HTML que representará el código JavaScript.	495
	refreshInputs()	Encargada de actualizar los inputs que muestran la matriz seleccionada, su longitud y sus argumentos.	580
	mutate()	Esta función únicamente se encarga de llamar a la función metodoArr() y de mantener siempre el rectángulo donde se muestra el código JavaScript en la última línea escrita.	586
	navegadorLanguage()	Esta función se encarga de cambiar el idioma de las etiquetas y botones principales según el idioma del navegador. Solo acepta ingles y español de momento.	593
	openWindowMethods()	Encargada de abrir la ventana secundaria y de advertir al usuario en caso de que no haya seleccionado ningún método para ver la ayuda.	648
	onPageLoad()	Encargada de limpiar todos los inputs al cargar la página y de advertir a los usuarios de dispositivos móviles que la aplicación web no está optimizada para dichos dispositivos.	656
methods.html	methodSelected()	Encargada de mostrar y ocultar el contenido HTML correspondiente dependiendo del método seleccionado por el usuario	370
	changeDisplay()	Función necesaria para que la función methodSelected() funcione de manera correcta.	374

BUCLES

Como casi cualquier programa, la aplicación web hace uso de diversos bucles a lo largo de su ejecución.

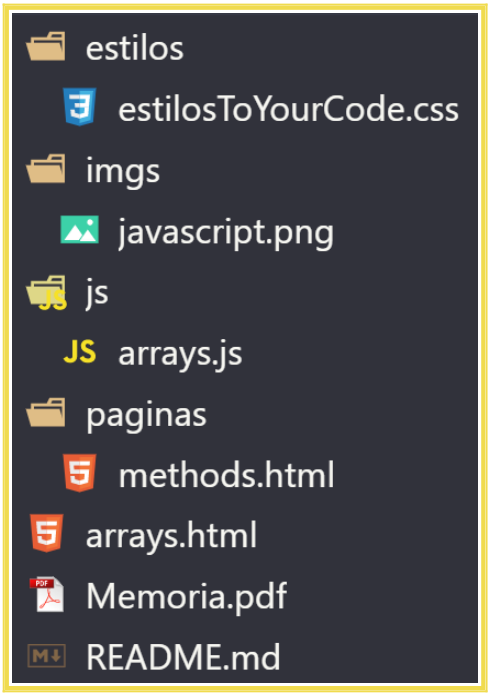
OBJETOS DEFINIDOS POR EL USUARIO

El único objeto que he creado para la aplicación web es el objeto Arrays, el cual es el encargado del mantenimiento de las matrices introducidas por el usuario. Este objeto está compuesto por los siguientes método y propiedades:

- **PROPIEDADES**
  - **noms**: Matriz que almacena los nombres de cada matriz.
  - **arrs**: Matriz que almacena los elementos de cada matriz, en cada posición de arrs tenemos una matriz de elementos pertenecietes a cada matriz que a introducido el usuario, haciendo de arrs una matriz bidimensional.
  - **lengths**: Matriz que almacena la longitud de cada matriz.
  - **history**: Matriz que almacena los métodos que ejecuta cada matriz, en cada posición almacena una matriz con el nombre de los métodos que ha ejecutado la matriz que corresponde con esa posición, haciendo de history también una matriz bidimensional.
- **MÉTODOS**
  - **addArray()**: Añade el nombre, los elementos y la longitud de la matriz introducida por el usuario, además añade una matriz vacía en la propiedad history.
  - **findArrByNom()**: Pasandole el nombre de una matriz nos devuelve sus elementos.
  - **findPosByNom()**: Pasandole el nombre de una matriz nos devuelve su posición.
  - **deleteArray()**: Pasandole el nombre de una matriz elimina su nombre, elementos, longitud e historial. La elimina.
  - **lengthOfArray()**: Pasandole el nombre de una matriz nos devuelve su longitud.
  - **historyOfArray()**: Se encarga de almacenar los métodos que ejecuta cada matriz a modo de historial.
  - **info()**: Muestra en la consola del navegador los nombres, elementos, longitudes e historiales de las matrices actuales.

ESTRUCTURA DE FICHEROS

En la carpeta del proyecto podemos encontrar los siguientes ficheros y directorios:



- **arrays.html** es la página principal y contiene la interfaz de la aplicación web.
- **methods.html** es la página que contiene las ayudas y ejemplos de cada método.
- **arrays.js** es el fichero que contiene todo el código JavaScript de la aplicación web, a excepción de dos funciones que se encuentran dentro del archivo methods.html.
- **estilosToYourCode.css** contiene el código css que da color al código de los ejemplos y de la ventana donde se va representando el código JavaScript, este archivo css se usa exclusivamente para este propósito, las demás modificaciones con respecto al css las realizo dentro del propio fichero html.
- **javascript.png** es el icono que uso para la pestaña del navegador.
- **Memoria.pdf**
- **README.md** no influye en el proyecto, es un fichero informativo para los usuarios que visiten este proyecto en GitHub.

DETALLES, LIMITACIONES Y OTROS DATOS DEL PROYECTO

- Casi todo el estilo de la aplicación está realizado con Bootstrap.
- Al obtener los elementos que el usuario quiere en la matriz desde un input obtengo una cadena de tipo string en lugar de una matriz de elementos, esto me obliga a separar la cadena por comas para poder obtener cada elemento que el usuario ha introducido y así crear una matriz de verdad. Debido a esto la aplicación web no permite que un elemento tenga una o varias comas en medio, de ser así pasaría a ser más de un elemento.
- El punto anterior está explicado en una pequeña ventana emergente llamada popover que se muestra al hacer clic en el botón que se encuentra al lado del input de la longitud de la matriz. Este popover es lo único que utiliza jquery en el proyecto, todo lo demás es JavaScript.
- Una de las cosas que quería hacer, y he hecho en la página web es que cualquier usuario pudiese "toquetear" el programa sin tener ni idea de programación. Para esto me pareció muy importante que el usuario pudiese introducir los elementos sin tener que escribirlos entre comillas. A mi parecer una persona cualquiera no tiene porque saber que los datos de tipo string que queramos introducir en una matriz deben ir entre comillas para no ser confundidos con variables sin declarar que provocarán un error. A causa de esto la aplicación web quedó limitada a interpretar siempre los datos de tipo numéricos únicamente como numéricos, es decir, un usuario nunca podrá introducir un "2" de tipo string, siempre será un 2 de tipo number.
- Al margen de la nota del proyecto estoy muy contento con el trabajo realizado.