

Introducción a Javascript

Clase 4

Arrays I

Arrays: introducción

¿Qué es un array o matriz?

- Es un espacio de la memoria con un nombre de referencia para identificarlo, que se **divide** en sub-espacios.
- Cada **sub-espacio** se identifica por su **índice numérico**, que contendrá un valor.
- El **primer índice** siempre será el número **0**.
 - Ejemplo:
 - `let nombre = 'Carlitos', apellido = 'Tevez';`
 - `let materias = ['Historia', 'Física'];`

Memoria RAM			
nombre	apellido	materias	
Carlitos	Tevez	0	1
		Historia	Física

Arrays: introducción

¿Para qué sirven los arrays o matrices?

- Los arrays **reducen** la cantidad de variables a utilizar en un programa.
- Los arrays son **dinámicos**, durante el programa se le pueden agregar o quitar elementos (con las variables comunes no se puede).
- Los arrays se pueden **recorrer** de principio a fin, o como se lo desee (con las variables comunes no se puede).
- Los arrays son una manera de tener en un solo elemento una **fuentes de múltiples datos**.

Arrays: Creación

Arrays: creación

¿Cómo se crea un array o matriz?

- Existen dos formas de crear un array:
 - Usando la versión constructora:
 - `let datosConstructor = new Array();`
 - Usando la versión literal:
 - `let datosConstructor = [];`
- **ATENCIÓN**
 - ¡Si no se le indica a la variable que será un array, estaremos trabajando con una variable común ... por lo tanto, toda operación del tipo array a realizar posteriormente, causará errores en la ejecución del código!

Arrays: Contenido

Arrays: contenido

¿Cómo se le guardan valores a un array o matriz?

- Existen tres formas de agregar contenido a un array, sin importar si fue creado con la versión constructora o literal:
 - Separando con coma los valores entre los corchetes o paréntesis

```
let datosConstructor = new Array('Cadena', 21, true, 'ABC', [], null);  
let datosLiteral = ['Cadena', 21, true, 'ABC', [], null];
```

- Como se puede apreciar, un array permite guardar distintos tipos de datos en la misma variable.

Arrays: contenido

¿Cómo se le guardan valores a un array o matriz?

- Existen tres formas de agregar contenido a un array, sin importar si fue creado con la versión constructora o literal:
 - Creando un array vacío e ir guardando valores en cada índice indicado

```
let datos = [ ];  
datos[0] = 'Cadena';  
datos[1] = 21;  
datos[3] = true;  
datos[4] = [ ];  
datos[5] = null;
```

- Es indistinto si se crea con la versión constructora o literal.
- Se podrían saltar índices si se deseara. Los índices saltados se crean automáticamente.
- Por ser sub-espacios declarados sin inicializar, su valor será ***undefined***.

Arrays: contenido

¿Cómo se le guardan valores a un array o matriz?

- Existen tres formas de agregar contenido a un array, sin importar si fue creado con la versión constructora o literal:

- Creando un array vacío y cargando su contenido mediante el método ***push***

```
let datos = [ ];  
datos.push('Cadena');  
datos.push(21);  
datos.push(true);  
datos.push([ ]);  
datos.push(null);
```

- Es indistinto si se crea con la versión constructora o literal
- El método crea automáticamente un nuevo índice al final de array y guarda el valor pasado entre paréntesis (más adelante veremos este método en detalle).

Arrays: contenido

¿Cómo se puede mostrar el contenido de un array o matriz?

- Se puede mostrar el contenido particular, seleccionando cada índice que se desea "imprimir"

```
let datos = ['ABC', 1999, false];  
console.log(datos)           // ABC, 1999, false  
console.log(datos[0])        // ABC  
console.log(datos[2])        // false  
console.log(datos[3])        // undefined
```

- Veamos un ejemplo para comprender mejor ([ejemplo-1-1.js](#))

Arrays: diferencia entre constructor y literal

Arrays: diferencia entre constructor y literal

- Se sabe que si se colocan valores separados por coma entre los paréntesis o corchetes, cada uno se almacenará en un índice, comenzando desde el 0.
- Pero si al **crear un array de la manera constructora**, solamente se le coloca **un** número **entre sus paréntesis**, se estará definiendo la **longitud inicial** del array (es decir, cuántos índices ya debe tener reservados)
- Hasta que esos espacios no se inicializan, cada índice tendrá como valor **undefined**.
- Esto se suele utilizar en los casos que se sabe la longitud a la hora de trabajar con un array para evitar consumo de memoria
- La realidad es que hoy en día esto no es necesario.

Arrays: equivocaciones más comunes al comenzar

¿En qué nos solemos equivocar ni bien comenzamos a trabajar con arrays?

- A la hora de crearlos

- `let datos[];` **Falta el igual** \Rightarrow `let datos = [];`
- `let datos = Array ();` **Falta el new** \Rightarrow `let datos = new Array();`
- `let datos[] = [];` **Sobran corchetes** \Rightarrow `let datos [] = [];`

- A la hora de guardar contenido

- `let datos; datos[0] = 'valor';` **Nunca inicializamos la variable como un array**
 \Rightarrow `let datos = [];`

Arrays: Recorrido

Arrays: recorrido

Ciclo for:

- Sirve para recorrer un array numérico, utilizando una variable para recorrer cada índice
- Generalmente, se debe ir desde el índice 0 hasta la longitud (**length**) del array exclusive.

Ciclo for in:

- Recorre cada índice, que será lo que se guarda en la variable declarada entre paréntesis.

Ciclo for of:

- A diferencia del anterior, recorre cada valor, que será lo que se guarda en la variable declarada entre paréntesis. Sirve para recorrer valores en lugar de índices. Su sintaxis es:
- Veamos un ejemplo para comprender mejor [\(ejemplo-2-1.js\)](#)

Arrays: sintaxis de recorridos

Ciclo for in:

```
for (let indice in array) {  
    console.info(indice, array[indice]);  
}
```

Ciclo for of:

```
for (let valor of array) {  
    console.info(valor);  
}
```


Arrays II

Array o matriz multidimensional: introducción

¿Qué es un array o matriz multidimensional?

- Es un array que tiene en sus subdivisiones otros arrays
- Se suelen utilizar cuando por cada elemento del array principal se tienen muchos datos diferentes
- Podríamos pensar a los arrays multidimensionales como el juego "Batalla naval".

	1	2	3	4	5	6	7	8
A								
B								
C								
D								
E								
F								
G								
H								

- ¿Cómo se "dispara" aquí? ¿cuáles serían las coordenadas?
- En Javascript para indicar los índices de un array multidimensional es igual.
- Para acceder a un elemento de un array, que está en un índice de otro array, se deben tener dos coordenadas.
- Cada coordenada será un conjunto de corchetes a continuación del anterior.
- La ruta se hace desde el array de más afuera al de más adentro (en el ejemplo, fila, luego columna).

Array o matriz multidimensional: introducción

Del tablero al código

	0	1	2	3	4	5	6	7
0	N	N	A	N	N	A	N	A
1	A	A	A	A	A	A	N	A
2	A	N	N	N	N	A	N	A
3	A	A	A	A	A	A	N	A
4	A	A	N	N	N	A	N	A
5	A	A	A	A	A	A	A	A
6	N	A	N	N	N	A	A	N
7	N	A	A	A	A	A	A	N

- Pensemos un array tablero, que tiene 8 posiciones (del 0 al 7)
- Por cada posición (o fila) tenemos un array con otras 8 posiciones (cada uno sería una celda)
- Si contiene una N hay una nave, si contiene una A, es agua.
- ¿Cómo se "dispara" a la celda roja?
- ¿Cómo disparamos a la celda violeta?
- Veamos un ejemplo ([ejemplo-1-1.js](#))

Array o matriz multidimensional: armado

Otra forma de armar un array multidimensional

- Vimos que se puede armar una matriz de la siguiente manera
let matriz = [];
matriz[0] = ['Valor 1', 1, false];
matriz[1] = ['Valor 2', 2, true];
matriz[2] = ['Valor 3', 3, false];
- Pero sabiendo que cada índice de un array tiene un valor ...
- Y que cada valor puede ser un nuevo array dentro de dicho índice ...
- Se puede inicializar una matriz de la siguiente manera:
- let matriz = [
 ['Valor 1', 1, false],
 ['Valor 2', 2, true],
 ['Valor 3', 3, false],
];

Array o matriz multidimensional: recorrida

Iterar un array o matriz multidimensional

- Para recorrer un array dentro de otro array (una matriz de dos niveles) solamente haría falta anidar un ciclo for dentro de otro for
 - El **primer for** recorre el **primer nivel** de la matriz
 - El **segundo for** (el anidado) recorre el **segundo nivel** de la matriz, por cada índice del primer nivel
- Deben tenerse presente las siguientes consideraciones
 - Las **variables** de cada for deben ser **diferentes**
 - El primer for recorre el primer nivel, es decir
 - `matriz[indice_nivel_1]`
 - El segundo for (el anidado) recorre el segundo nivel, dentro del primero, es decir:
 - `matriz[indice_nivel_1] [indice_nivel_2]`
- Veamos un ejemplo para comprender mejor ([ejemplo-2-1.js](#))

Array o matriz multidimensional: recorrida

Iterar un array o matriz multidimensional

- En el ejemplo anterior, se visualiza cada valor del segundo nivel de la misma forma
- Partiendo de la base de que uno conoce la estructura del array:
 - Es muy probable que no todos los datos deban visualizarse de la misma manera, por ejemplo, una imagen como mínimo debería mostrarse mediante una etiqueta img
 - Entonces, podríamos utilizar un condicional para ver en qué posición del segundo nivel se está para visualizar el valor correctamente.
 - Veamos un ejemplo para comprender mejor [\(ejemplo-3-1.js\)](#)
 - Otra forma de visualizar la información sería utilizando un for que solo recorre el primer nivel
 - El segundo nivel se arma sin necesidad de otro for
 - Veamos un ejemplo para comprender mejor [\(ejemplo-4-1.js\)](#)
- En conclusión, las dimensiones de un array no tienen límite, la forma de recorrer o trabajar con sus niveles internos siempre es igual (sean 2 o más niveles).



Estoy Cansado, Jefe.

Array asociativo

¿Qué es un array asociativo?

- Son aquellos arrays que en lugar de tener índices numéricos, tienen **índices del tipo cadena**.
- Su objetivo es salir de la abstracción de posiciones numéricas (0, 1, 2, 3 ...) para utilizar palabras y hacerlo de **más fácil lectura**.
- Los índices de un array asociativo son cadenas, por ende **van entre comillas** (de lo contrario haremos referencia a una variable y no funcionará correctamente)

Analicemos la siguiente información

- Tenemos un array con 4 posiciones numéricas

```
let alumno = [ ];
```

alumno[0] = 'Franco';	=>	¿Este índice contiene el nombre?
-----------------------	----	----------------------------------

alumno[1] = 'Martín';	=>	¿Este índice contiene el apellido?
-----------------------	----	------------------------------------

alumno[2] = true;	=>	¿Qué significa el true?
-------------------	----	-------------------------

alumno[3] = 21061985;	=>	¿Es el DNI?
-----------------------	----	-------------

Array asociativo

¿Qué es un array asociativo?

- Son aquellos arrays que en lugar de tener índices numéricos, tienen **índices del tipo cadena**.
- Su objetivo es salir de la abstracción de posiciones numéricas (0, 1, 2, 3 ...) para utilizar palabras y hacerlo de **más fácil lectura**.
- Los índices de un array asociativo son cadenas, por ende **van entre comillas** (de lo contrario haremos referencia a una variable y no funcionará correctamente)

Analicemos la siguiente información

- Tenemos un array con 4 posiciones numéricas

```
let alumno = [ ];
```

```
alumno[0] = 'Franco';
```

=>

```
alumno['apellido'] = 'Franco';
```

```
alumno[1] = 'Martín';
```

=>

```
alumno['nombre'] = 'Martín';
```

```
alumno[2] = true;
```

=>

```
alumno['egresado'] = true;
```

```
alumno[3] = 21061985;
```

=>

```
alumno['DDMMAAAA'] = 21061985;
```

Array asociativo: recorrida

Iterar un array asociativo

- ¿Cómo ir de "apellido" a "DDMMAAAA"?
- Claramente para recorrer un array asociativo ya no podemos usar el for común.
- Vamos a utilizar el for in
- Tiene la ventaja de que se puede aprovechar también el índice como información.

```
for (let indice in Array) {  
    // Recorre cada índice del array (índice es una variable, puede ser i)  
    console.info(Array[indice]);  
}
```

Fin de la clase

Este es el espacio para dudas y/o preguntas existenciales