Universitat Oberta de Catalunya

Aula

M4.250 - Introducción a la programación en JavaScript I aula 1

Arrays

Inicio: **22/04/20**

Fin: **05/05/20**

Introducción

En este apartado veremos el tema Colecciones con índice del material de Mozilla.

Un detalle antes de empezar. Suele pasar que las traducciones al español varían según las preferencias del traductor. Así, Array, se suele traducir cómo Matriz, cómo tabla o cómo Vector dependiendo del traductor. En Mozilla han escogido Matriz, en la wikipedia lo podréis encontrar cómo vector . Aunque el autor de estas líneas lo aprendió como tabla y, por tanto, prefiere esta palabra para designarlos, para simplificar vamos a usar la palabra inglesa array.

Un array es un conjunto de datos, habitualmente del mismo tipo, a los que podemos acceder mediante un índice.

- Conjunto de datos del mismo tipo: Conjunto de números, conjunto de cadenas de caracteres.
- Podemos acceder mediante un índice: el dato 0 será el primer número o cadena, el dato 1, el segundo, el dato 2, el tercero, y así sucesivamente.

Creando un Array
Crear un array con contenido
Crear un array vacío

Podemos crear un array de diferentes maneras.

Crear un array con contenido

Primero veremos tres maneras de crear arrays con contenido:

```
var arr = new Array(23,445,43,77,23);
var arr = Array("cero","uno","dos");
```

```
var arr = [true,false,false,true,true];
```
Aquí hemos creado tres arrays, con datos de tres tipos distintos. Las tres maneras de crear arrays son equivalentes, por lo que habitualmente se usa la tercera manera:

```
var arr = [23,445,43,77,23];
```
Crear un array vacío
Los arrays que hemos creado hasta ahora, tienen ya un contenido predefinido. Podemos crear arrays vacíos, sin un contenido previo. De nuevo podemos hacerlo de tres maneras distintas:

```
var arr = new Array(10);
var arr = Array(10);
var arr = []; arr.length = 10;
```
Estas tres maneras de crear un array vacío son equivalentes. En los tres casos se crea un array vacío de 10 posiciones. En los dos primeros casos lo hacemos con una sola instrucción, en la tercera con dos.

Sin embargo, si vamos a crear un array vacío, no tiene mucho sentido indicar que tamaño tendrá. Los arrays crecen conforme se les añaden elementos, con lo cual, para crear un array vacío, lo más fácil será usar la siguiente instrucción:

```
var arr = [];
```
¿Dudas? Sigue con el siguiente tema y verás algunos ejemplos y un vídeo que te ayudarán a entenderlo.

Trabajando con los Array
Llenar un array
Consultar un elemento de un array
Borrar un dato de un array

Trabajar con los arrays significará guardar datos en ellos, consultarlos y borrarlos. Vamos a ver cómo:

Llenar un array
Recuerda que podemos guardar datos en un array en el momento de crearlo.

Si tenemos un array vacío, guardar datos en él es tan fácil como indicar en qué posición del array queremos guardar el dato:

```
var arr = []; // Creamos un array vacío
arr[0] = "Un dato";
```
Si ahora ejecutásemos la instrucción

```
console.log(arr);
```
Lo que veríamos sería esto:

```
["Un dato"]
```
Vamos a añadir un dato más:

```
arr[3] = "Otro dato";
```
Nos hemos saltado dos posiciones (la 1 y la 2). ¿Qué pasará?

Esto:

```
["Un dato", undefined, undefined, "Otro dato"]
```

El array ha crecido para dar cabida al elemento de la posición 3 y los elementos que no tienen contenido se rellenan con **undefined** .

Consultar un elemento de un array

Para consultar un elemento del array, el funcionamiento es similar a cuando guardamos un dato. Ponemos el nombre del array y entre corchetes la posición que queremos consultar:

```
var arr = ["Un dato", "Otro dato", "Otro dato más", "¡Y otro!"];
var a = arr[2];
```

¿Qué estamos guardando en la variable **a** ?

Hemos de tener en cuenta que el índice del array empieza en 0. Por tanto estaremos guardando "Otro dato más".

Borrar un dato de un array

Para borrar un dato de un array lo único que debemos hacer es guardar otro en su posición. Si no queremos guardar ninguno, siempre podemos guardar un **undefined** . También podemos reducir el tamaño de un array. En este caso, todos los elementos que queden fuera del nuevo tamaño desaparecerán.

Veámoslo con un ejemplo:

```
var arr = [33,44,51,34,67,88];
console.log(arr); // Devolverá [33,44,51,34,67,88]
arr.length = 3;   // El tamaño del array será 3
console.log(arr); // Devolverá [33,44,51]
```

Los elementos 34,67, y 88 se habrán perdido y no se podrán recuperar.

Ejercicios y ejemplos

Vamos a ver un par de ejemplos.

Ejemplo 1

Supongamos que tenemos un array donde vamos guardando la temperatura que hace en nuestra ciudad cada día y queremos que, en un momento determinado podamos calcular la temperatura máxima y la media de todos los datos acumulados.

El array de temperaturas con el que vamos a hacer pruebas es éste:

```
var temperatura = [25, 24, 20, 23, 22, 25, 27, 26, 23, 27];
```

Crearemos dos funciones, cada una para hacer una de las operaciones.

Calcular la media

```
function media(temp) {
  let suma = 0;
  for (let i=0; i
```

Calcular la temperatura máxima

```
function max(temp) {
  let vMax = temp[0];
  for (let i=1;i
```

Un detalle. Cuando tengamos que calcular el máximo (o el mínimo) de una secuencia, la mejor opción para inicializar la variable donde guardaremos el máximo es el primer elemento de la secuencia.

El programa será entonces muy sencillo:

```
console.log(media(temperatura));
console.log(max(temperatura));
```

El programa se puede probar en uno de estos dos editores online:

- Python Tutor: https://goo.gl/LXMZkt
- CodePen: https://codepen.io/ccasadom/pen/BbOeBd?editors=0012

Ejemplo 2

Vamos a hacer un programa que nos guarde en una tabla todos los números primos que encuentre entre 2 y un número dado.

La solución está disponible en:

https://codepen.io/ccasadom/pen/WmgBYL?editors=0012

Recursos de aprendizaje

Fuentes de información