CFGS DAW

Mòdul 6: Desenvolupament web en entorn client

INS Joan d'Àustria

Arrays

Podemos definir un array como una colección de valores a los que llamamos **elementos**. Esta colección almacena los elementos en base a una posición numérica denominada **índice**.

En realidad, un array es de tipo object, es decir, es un objeto con funcionalidad añadida

Arrays

Esta definición de Array es similar a la que se puede encontrar en cualquier lenguaje de programación. No obstante, podemos observar algunas diferencias:

- Como javascript es un lenguaje no tipado, un elemento de un array puede ser de cualquier tipo y dos elementos de un array pueden ser de dos tipos diferentes
- El tamaño de un array se ajusta dinámicamente, no es necesario indicar este tamaño en la declaración

Arrays: Declaración

- Mediante el constructor:

```
var  unArray = new Array();
```

Si sabemos el número de elementos, también podemos indicárselo en el momento de declararlo:

```
var  unArray = new Array(10);
```

También podemos darle valores iniciales directamente:

```
var  unArray = new Array("rojo", "verde", "azul");
```

Si sólo se le indica un valor y este es de tipo numérico se asume que es la longitud del Array!

El operador new se puede omitir

```
var  unArray = Array(10);
```

Arrays: Declaración

- Mediante la notación literal de arrays:

```
var  unArray = [“rojo”, “verde”, “azul”];
```

Hay que tener cuidado de no dejar comas sueltas. Por ejemplo, el código:

```
var  unArray = [1,2, ];
```

- En IE 8 y anteriores creará un array de 3 elementos con valores 1, 2 y undefined.
- En el resto de navegadores, el array tendrá 2 elementos

Arrays: Acceso a los elementos

- Mediante la posición de los elementos, podemos acceder para consultar su valor o para modificarlo

```
alert (unArray[2]);
unArray[1]="blanco";
```

- El primer elemento está en la posición 0.
- Si se utiliza un índice superior a la última posición para modificar un valor, el array crecerá dinámicamente y tendrá esa nueva longitud

```
unArray[25]="lila"; // el array tiene ahora longitud 26
```

Arrays: Acceso a los elementos

- La longitud de un array mide el número de elementos que tiene. Esta propiedad, a diferencia de en otros lenguajes no es de solo lectura.
- Podemos eliminar o añadir elementos modificando esta propiedad

```
var  unArray = [“rojo”, “verde”, “azul”];
unArray.length=2;
alert (unArray[2]);  //undefined!! Recordar que el acceso es a
                                  partir de la posición 0!
```

Arrays: Acceso a los elementos

- Mediante el uso de esta propiedad también podemos añadir elementos al final del array

Posición	0	1	2	...	Longitud-2	Longitud-1
Valor	A	B	C	...	R	S

```
unArray[unArray.length]="T";  //Añade el valor "T" al final en un nuevo elemento
```

La longitud máxima de un array es 2^{32}-1 elementos (= *4.294.967.295)*

Arrays: Acceso a los elementos

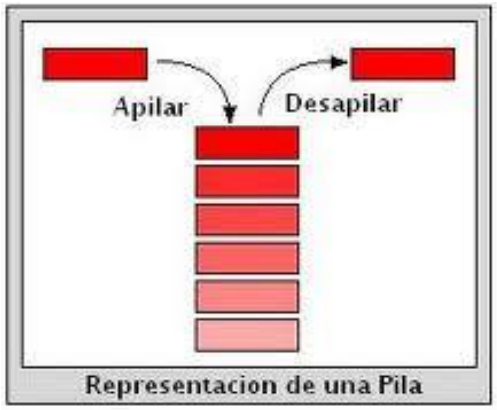
- O recorrer todo el vector para consultar o modificar valores

```
for (var i=0; i<unArray.length; i++){
        unArray[i]=0;
}
```

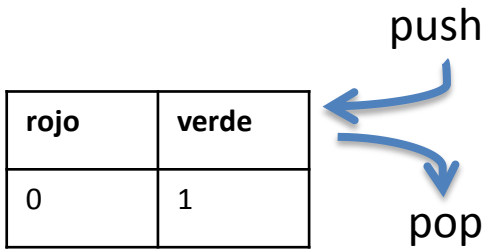
```
for (var i=0; i<unArray.length; i++){
        document.write(unArray[i] + “<br>“);
}
```

Arrays: Pilas

- Pila: Estructura que sigue un comportamiento LIFO (“Last In First Out” o el último que llega es el primero en salir)



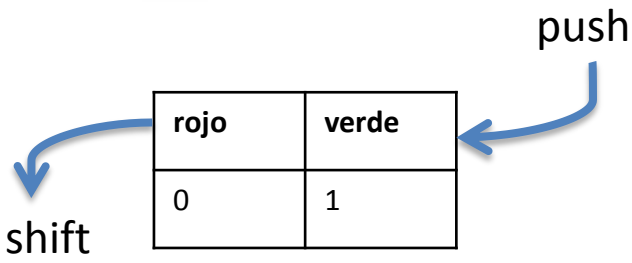
Métodos	
push	Recibe como parámetros los elementos a apilar y retorna el total de elementos que hay en la pila
pop	Retorna el elemento de la parte superior de la pila (el último)




```
var colores = new Array();
var x = colores.push(“rojo”, “verde”);
alert(x); //2
var unElem = colors.pop();
alert(unElem); //”verde”
alert(colores.length); //1
```

Arrays: Colas

- Cola: Estructura que sigue un comportamiento FIFO (“First In First Out” o el primero que llega es el primero en salir)



Métodos	
push	Recibe como parámetros los elementos a apilar y retorna el total de elementos que hay en la pila
shift	Retorna el primer elemento de la lista

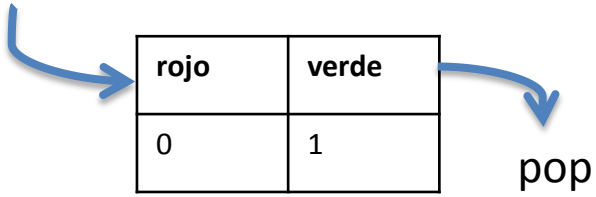
```
var colores = new Array();
var x = colores.push(“rojo”, “verde”);
alert(x); //2
var unElem = colors.shift();
alert(unElem); //”rojo”
alert(colores.length); //1
```

Arrays: Colas

- Cola: Podemos crear una cola invertida

Métodos	
unshift	Recibe como parámetros los elementos a añadir y retorna el total de elementos que hay en la lista
pop	Retorna el último elemento de la lista (el primero que se añadió)

unshift




```
var colores = new Array();
var count = colors.unshift("rojo", "verde");
alert(count); //2
var item = colors.pop();
alert(item); //"verde"
alert(colors.length); //1
```

Arrays: Manipulación

- Concatenar arrays (concat):
  - Sin parámetros: se hace una copia del array
  - Con parámetros: se hace una copia y se añaden los elementos al final del nuevo array

```
var colores = [“rojo”, “verde”, “azul”];
var colores2 = colores.concat(“amarillo”,”gris”);
```

colores

rojo	verde	azul
0	1	2

colores2

rojo	verde	azul	amarillo	gris
0	1	2	3	4

Arrays: Manipulación

- Trocear arrays (slice):
  - Con un parámetro: Se retornan los elementos desde esa posición hasta el final
  - Con dos parámetros: indican posición inicial y final (posición final no incluida)

```
var colores = [“rojo”, “verde”, “azul”, “amarillo”, “gris”];
var colores2 = colores.slice(1);
var colores3 = colores.slice(1,4);
```

colores

rojo	verde	azul	amarillo	gris
0	1	2	3	4

colores2

verde	azul	amarillo	gris
0	1	2	3

colores3

verde	azul	amarillo
0	1	2

Arrays: Manipulación

- Empalmar arrays (splice):
  - Dos parámetros (pos_ini, x): Se eliminan x elementos desde la posición inicial
  - Tres parámetros o más (pos_ini, reempl?, elems):
    - Si reempl? = 0, se insertan los elementos en la posición inicial indicada
    - Si reempl? > 0, se sustituirán los elementos indicados por este parámetro desde la posición inicial por los indicados como parámetro. Si indicamos un número mayor que nuevos elementos, se eliminarán los restantes. Si indicamos un número menor que nuevos elementos, se sustituirán los indicados y se insertarán los demás.

Arrays: Manipulación

- Empalmar arrays (splice):

```
var colores = [“rojo”, “verde”, “azul”];
var borrados= colores.splice(0,1);
```

colores

rojo	verde	azul
0	1	2

colores

verde	azul
0	1

borrados

rojo
0

```
Borrados = colores.splice(1, 0, “amarillo”, “naranja”);
```

colores

verde	azul
1	2

colores

verde	amarillo	naranja	azul
0	1	2	3

borrados


```
Borrados = colores.splice(1, 1, “rojo”, “lila”);
```

colores

verde	amarillo	naranja	azul
0	1	2	3

colores

verde	rojo	Lila	naranja	azul
0	1	2	3	4

borrados

amarillo
0

Arrays: Localización

- Se proporcionan dos mecanismos para localizar elementos dentro de un array:
  - indexOf
  - lastIndexOf
- Ambos métodos reciben 2 parámetros:
  - El elemento a buscar
  - [Opcional] Posición inicial de búsqueda
- indexOf empieza a buscar por el principio, mientras que lastIndexOf empieza por el final
- Ambos métodos retornan la posición en la que se situa el elemento buscado si se ha encontrado o el valor -1 si no se ha encontrado
- La búsqueda se realiza internamente con el operador estrictamente igual (===).

Arrays: Iteración

- Se proporcionan cinco métodos para iteración de arrays. Todos ellos aceptan dos parámetros: una función a ejecutar y el ámbito en el que se aplica.
- La función a ejecutar siempre va a recibir tres parámetros: item, index y array
- Dependiendo del método, se retornan unos valores u otros.

Sólo estan disponibles para IE 9+, Firefox 2+, Safari 3+, Opera 9.5+, y Chrome.

Arrays: Iteración

- Estos cinco métodos son:
  - every: Ejecuta la función en todos los elementos del array y retorna cierto si la función retorna cierto para cada elemento
  - filter: Ejecuta la función en todos los elementos del array y retorna un array con los elementos para los que la función retorna cierto
  - forEach: Ejecuta la función en todos los elementos del array. No retorna nada.
  - map: Ejecuta la función en todos los elementos del array y retorna el resultado de ejecutar la función para cada elemento
  - some: Ejecuta la función en todos los elementos del array y retorna cierto si la función retorna cierto para un elemento

Arrays: Iteración


```
var numeros = [1,2,3,4,5,4,3,2,1];

// Comprobar que todos los elementos sean mayores que 2
var everyResult = numeros.every(function(item, index, array){
return (item > 2);
});
alert(everyResult); //false

// Comprobar que algún elemento sea mayor que 2
var someResult = numeros.some(function(item, index, array){
return (item > 2);
});
alert(someResult); //true
```

Arrays: Iteración

```
var numeros = [1,2,3,4,5,4,3,2,1];

// Obtener los números mayores que 2
var filterResult = numeros.filter(function(item, index, array){
return (item > 2);
});
// filterResult = [3,4,5,4,3]

//Obtener un array resultante de multiplicar todos los elementos por 2
var mapResult = numeros.map(function(item, index, array){
return item * 2;
});
// mapResult = [2,4,6,8,10,8,6,4,2]

//iniciar el array con valores 0
numeros.forEach(function(item, index, array){
    array[index]=0;
});
```