Javascript – Array

Javascript – array

In Javascript gli array sono collezioni che contengono elementi anche di tipo differente.

Ogni elemento ha una posizione all'interno dell'array e la numerazione parte da 0.

In Javascript gli array sono dinamici, ovvero non hanno una dimensione predeterminata. Aumentano o diminuiscono in base al numero di elementi presenti.

Ci sono molteplici modi per creare un array:

```
var array = [elem1, elem2, … , elemN];

var array = [ ];

var array = new Array();

var array = new Array(10);

var array = new Array(elem1, elem2, … , elemN);
```

Javascript – array

L'accesso ad un elemento dell'array, sia in lettura che in scrittura, avviene specificando l'indice dell'elemento tra parentesi quadre:

```
var array = [elem1, elem2, … , elemN];

array[2] = '12';

var elem1 = array[0];
```

L'aggiunta di un nuovo elemento può avvenire valorizzando un indice precedentemente non esistente, oppure utilizzando uno dei metodi messi a disposizione (vedi prossima slide).

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

array[2] = 3; //ora l'array ha 3 elementi

array[4] = 5; //ora l'array ha 5 elementi. Il quarto vale undefined
```

Javascript – array

Anche gli array sono oggetti e come tali hanno proprietà e metodi. Tra le proprietà troviamo:

- `length` – restituisce il numero di elementi presenti nell'array

Tra i metodi troviamo invece:

- `concat()` – concatena due o più array restituendone uno nuovo
- `push()` – accoda uno o più elementi ad un array esistente
- `unshift()` – inserisce uno o più elementi all'inizio di un array esistente
- `pop()` – rimuove l'ultimo elemento dell'array e lo restituisce
- `shift()` – rimuove il primo elemento dell'array e lo restituisce
- `splice()` – rimuove elementi dal centro dell'array con la possibilità anche di inserirne

Javascript – array

Tra i metodi troviamo invece:

- `toString()` – Converte l'array in stringa separando gli elementi con la virgola
- `join()` – Converte un array in stringa specificando un separatore
- `slice()` – Estrae una porzione di un array creandone uno nuovo
- `indexOf()` – Restituisce l'indice della prima occorrenza dell'elemento passato come parametro. -1 se non esiste.
- `lastIndexOf()` – Restituisce l'indice dell'ultima occorrenza dell'elemento passato come parametro. -1 se non esiste.
- `includes()` – restituisce true se l'elemento passato come parametro esiste nell'array

NB: Esistono anche altri metodi che prevedono il passaggio di funzioni. Li vedremo in seguito

Javascript – array – Spread Operator

La concatenazione di array può essere scritta più agevolmente usando l'operatore spread, composto da tre punti prima del nome di un array.

L'operatore ha l'effetto di sostituire al nome dell'array tutti i suoi elementi.

```
const array1 = [1,2,3];
const array2 = [4,5,6];

const array3 = [...array1, ...array2];
```

E' equivalente a:

```
const array3 = array1.concat(array2);
```

Javascript – for of

L'operatore for of permette di eseguire del codice su tutti gli elementi di oggetti che rappresentano collezioni (per esempio Array).

```
for(let variabile of object) { … }
```

Per esempio:

```
var array=[1,2,3,4];

for(let elemento of array) { console.log(elemento); }
```

Javascript – Funzioni

Javascript – funzioni

Le funzioni sono blocchi di codice, ai quali è associato un nome, richiamabili da altri punti del codice.

Le funzioni possono avere parametri in ingresso e possono avere valori di ritorno. In javascript si definiscono con la parola chiave function:

```
function nomeFunzione( param1, param2,…, paramN)
{
      istruzioni;
      return retVal;
}
```

Javascript – funzioni

I parametri in ingresso sono da considerarsi come variabili locali alla funzione stessa. Sono cioè utilizzabili all'interno della funzione e all'interno di eventuali funzioni contenute.

Il valore che la funzione restituisce al suo chiamante viene identificato con la parola chiave return.

Return ha anche l'effetto di terminare immediatamente l'esecuzione della funzione

La chiamata di una funzione avviene specificandone il nome e gli eventuali parametri.

```
function somma( x, y)
{
      return x+y;
}
var s = somma(10,7);
```

Javascript – funzioni - parametri

In linea di principio il passaggio dei parametri alle funzioni viene fatto per "valore". Il valore viene cioè copiato all'interno della variabile locale della funzione e le modifiche che questa può subire non si ripercuotono sul chiamante. Esempio:

```
function doppio( x)
{
      x = x*2;
      return x;
}
var n = 10;
var r = doppio(n);
//Quanto vale r? e quanto n?
```

Javascript – funzioni - parametri

Javascript non effettua particolari controlli sui parametri passati ad una funzione.

- Se vengono passati meno parametri di quelli previsti, i mancanti avranno il valore undefined
- Se vengono passati più parametri di quelli previsti, quelli in eccesso vengono ignorati.
- Se vengono passati valori di un tipo non previsto non si genera nessun errore. Eventualmente l'errore nascerà in fase di utilizzo del parametro

Javascript – funzioni - parametri

E' possibile definire dei valori di default per alcuni parametri. In questo modo se il parametro non viene passato assumerà il valore di default anziché undefined.

Si definisce un valore di default semplicemente specificandolo nella firma della funzione:

```
function nomeFunzione( param1, param2 = 10)
```

Se non specificato il param2 varrà 10. I parametri con valore di default devono necessariamente essere gli ultimi. Un parametro senza valore di default non può seguire uno che ce l'ha.

La sintassi è equivalente a scrivere:

```
function nomeFunzione( param1, param2) {
        param2 = param2 || 10;
}
```