

# JavaScript: Array

## Sommario

- Introduzione
- Array
- Dichiarazione e Allocazione di Arrays
- Generazione Random di Immagini con Array
- Riferimenti e Parametri di riferimenti
- Passaggio di Array a Funzioni
- Ordinamento
- Ricerca lineare e binaria
- Array Multidimensionali

## Obiettivi

- Introdurre la struttura dati array
- Capire l'uso degli array per memorizzare, ordinare e cercare elenchi e tabelle di valori
- Capire come dichiarare e inizializzare un array e come riferire un elemento negli array
- Capire come passare array a funzioni
- Gestire array multidimensionali

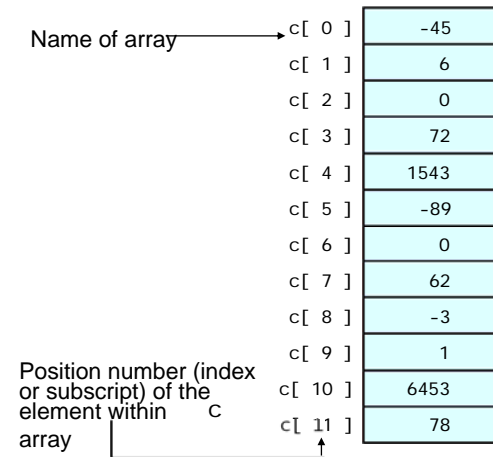
## Introduzione

- Array
  - È una struttura dati per la gestione di elementi omogenei
  - La struttura for è particolarmente utile per la loro scansione

## Array (1)

- Array in JavaScript
  - Ogni elemento è indicizzato da un numero
    - L'elemento iniziale è lo "zeresimo"
  - Per accedere a uno specifico elemento
    - nome dell'array
    - numero dell'elemento racchiuso tra parentesi quadre
  - Gli array conoscono la propria lunghezza
    - Proprietà length

## Array (2)



## Array (3)

Operators	Associativity	Type
() [] .	left to right	highest
++ -- !	right to left	unary
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
&&	left to right	logical AND
	left to right	logical OR
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment

Fig. 11.2 Precedence and associativity of the operators discussed so far.

## Dichiarazione e Allocazione di Array

- Array in memoria
  - Objects
  - Operatore new
    - Alloca memoria per gli oggetti
    - È un operatore di allocazione dinamica della memoria

```
var c;
c = new Array( 12 );
```

# Esempio

- Gli array crescono dinamicamente
  - Viene allocata maggiore memoria quando sono aggiunti nuovi elementi
- È necessario inizializzare gli elementi
  - Il valore di default è undefined
  - Riferirsi a elementi non inizializzati oppure a elementi esterni ai limiti dell'array produce un errore

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 11.3: InitArray.html -->
6 <!-- Initializing an Array -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10    <title>Initializing an Array</title>
11
12    <script type = "text/javascript">
13      <!--
14       // this function is called when the <body> element's
15       // onload event occurs
16       function InitializeArrays()
17       {
18         var n1 = new Array( 5 ); // allocate 5-element Array
19         var n2 = new Array(); // allocate empty Array
20
21         // assign values to each element of Array n1
22         for ( var i = 0; i < n1.length; ++i )
23           n1[ i ] = i;
```

Array n1 ha 5 elementi.

Array n2 è vuoto.

Il for inizializza gli elementi di n1 con un valore uguale al rispettivo indice (da 0 a 4).

```
24
25 // create and initialize five-elements in Array n2
26 for ( i = 0; i < 5; ++i )
27   n2[ i ] = i;
28
29 outputArray( "Array n1 contains", n1 );
30 outputArray( "Array n2 contains", n2 );
31 }
32
33 // output "header" followed by a two-column table
34 // containing subscripts and elements of "theArray"
35 function outputArray( header, theArray )
36 {
37   document.writeln( "<h2>" + header + "</h2>" );
38   document.writeln( "<table border = \"1\" width = " +
39     "\"100%\">" );
40
41   document.writeln( "<thead><th width = \"100\" +
42     "align = \"left\">Subscript</th> " +
43     "<th align = \"left\">Value</th></thead><tbody>" );
```

Il for aggiunge 5 elementi a Array n2 e inizializza ogni elemento al valore del rispettivo indice (da 0 a 4).

Ogni function mostra il contenuto dell'array passato come argomento in una tabella XHTML.

```
44
45   for ( var i = 0; i < theArray.length; i++ )
46     document.writeln( "<tr><td>" + i + "</td><td>" +
47       theArray[ i ] + "</td></tr>" );
48
49   document.writeln( "</tbody></table>" );
50 }
51 // -->
52 </script>
53
54 </head><body onload = "InitializeArrays()"></body>
55 </html >
```

# Esecuzione

Subscript	Value
0	0
1	1
2	2
3	3
4	4

Subscript	Value
0	
1	
2	
3	
4	

JavaScript: Array

13

# Dichiarazione e Inizializzazione

- È possibile dichiarare e inizializzare in un solo step
  - Si specifica l'elenco di valori  
`var n = [ 10, 20, 30, 40, 50 ];`  
`var n = new Array( 10, 20, 30, 40, 50 );`
  - È anche possibile inizializzare solo alcuni valori
    - Si lasciano vuoti gli elementi non inizializzati  
`var n = [ 10, 20, , 40, 50 ];`

JavaScript: Array

14

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 11.4: InitArray2.html -->
6 <!-- Initializing an Array with a Declaration -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10    <title>Initializing an Array with a Declaration</title>
11
12    <script type = "text/javascript">
13      <!--
14      function start()
15      {
16        // Initializer list specifies number of elements and
17        // value for each element.
18        var colors = new Array( "cyan", "magenta",
19          "yellow", "black" );
20        var Integers1 = [ 2, 4, 6, 8 ];
21        var Integers2 = [ 2, , , 8 ];
22
23        outputArray( "Array colors contains", colors );
24        outputArray( "Array Integers1 contains", Integers1 );
25        outputArray( "Array Integers2 contains", Integers2 );
26      }
27    </script>
28  </head>
29  <body onload = "start()">
30  </body>
31 </html>
```

JavaScript: Array

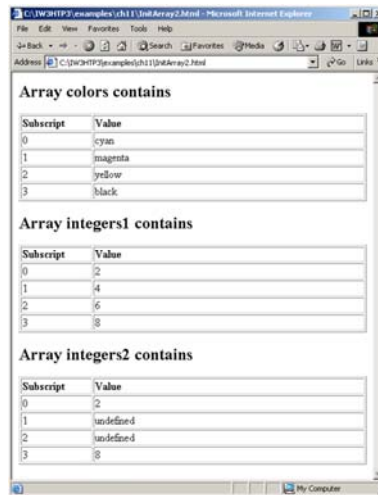
15

```
27
28 // output "header" followed by a two-column table
29 // containing subscripts and elements of "theArray"
30 function outputArray( header, theArray )
31 {
32   document.writeln( "<h2>" + header + "</h2>" );
33   document.writeln( "<table border = \"1\" +
34     \"width = \"100%\">" );
35   document.writeln( "<thead><th width = \"100%\" +
36     \"align = \"left\">Subscript</th> +
37     \"<th align = \"left\">Value</th></thead><tbody>" );
38
39   for ( var i = 0; i < theArray.length; i++ )
40     document.writeln( "<tr><td> " + i + "</td><td> " +
41       theArray[ i ] + "</td></tr>" );
42
43   document.writeln( "</tbody></table>" );
44 }
45 // -->
46 </script>
47
48 </head><body onload = "start()"></body>
49 </html >
```

JavaScript: Array

16

# Esecuzione



JavaScript: Array

17

# Operatore for ... in

- Svolge una azione per ogni elemento elemento dell'array
  - Svolge una iterazione su tutti gli elementi
  - Assegna ogni elemento a una variabile per volta
  - Ignora elementi che non esistono

JavaScript: Array

18

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 11.5: SumArray.html -->
6 <!-- Summing Elements of an Array -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9 <head>
10 <title>Sum the Elements of an Array</title>
11
12 <script type = "text/javascript">
13 <!--
14 function start()
15 {
16     var theArray = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ];
17     var total1 = 0, total2 = 0;
18
19     for ( var i = 0; i < theArray.length; i++ )
20         total1 += theArray[ i ];
21
22     document.writeln( "Total using subscripts: " + total1 );
23
```

JavaScript: Array

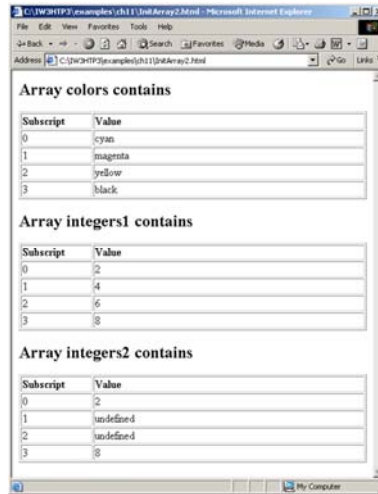
19

```
24     for ( var element In theArray )
25         total2 += theArray[ element ];
26
27     document.writeln( "<br />Total using for...in: " +
28         total2 );
29     }
30     // -->
31 </script>
32
33 </head><body onload = "start()"></body>
34 </html >
```

JavaScript: Array

20

# Esecuzione



JavaScript: Array

# Array come Sostituti dello switch

- Ogni elemento rappresenta un caso dello switch

JavaScript: Array

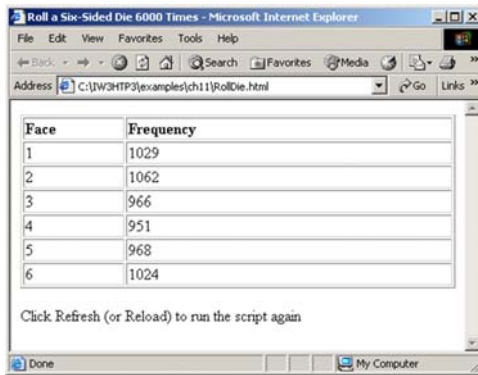
```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 11.6: RollDie.html -->
6 <!-- Roll a Six-Sided Die 6000 Times -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml" >
9 <head>
10 <title>Roll a Six-Sided Die 6000 Times</title>
11
12 <script type = "text/javascript">
13 <!--
14 var face, frequency = [ , 0, 0, 0, 0, 0 ];
15
16 // summarize results
17 for ( var roll = 1; roll <= 6000; ++roll ) {
18 face = Math.floor( 1 + Math.random() * 6 );
19 ++frequency[ face ];
20 }
21
```

JavaScript: Array

```
22 document.writeln( "<table border = \"1\" +
23 \"width = \"100%\"> );
24 document.writeln( "<thead<th width = \"100%\" +
25 \" align = \"left\">Face<th align = \"left\"> " +
26 "Frequency</th></thead></tbody>" );
27
28 for ( face = 1; face < frequency.length; ++face )
29 document.writeln( "<tr><td> " + face + "</td><td> " +
30 frequency[ face ] + "</td></tr>" );
31
32 document.writeln( "</tbody></table>" );
33 // -->
34 </script>
35
36 </head>
37 <body>
38 <p>Click Refresh (or Reload) to run the script again</p>
39 </body>
40 </html >
```

JavaScript: Array

# Esecuzione



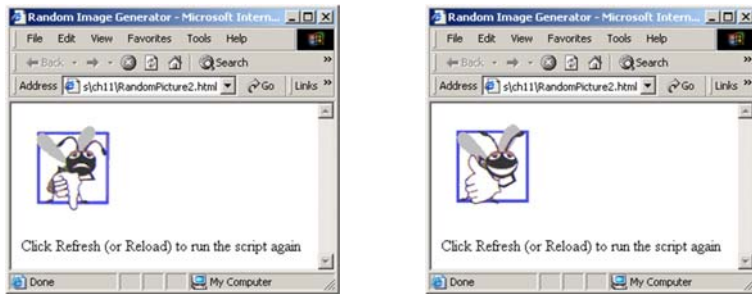
# Generazione casuale di immagini usando array

- Approccio diverso (più chiaro) rispetto a quanto visto precedentemente
  - Specifica il nome dei file anziché interi
  - Il risultato della chiamata a `Math.random()` è l'indice nell'array

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5 <!-- Fig. 11.7: RandomPicture2.html -->
6 <!-- Randomly displays one of 7 images -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml" >
9   <head>
10    <title>Random Image Generator</title>
11
12    <script type = "text/javascript">
13      <!--
14      var pictures =
15        [ "CPE", "EPT", "GPP", "GUI", "PERF", "PORT", "SEO" ];
```

```
16
17     document.write ( "<img src = \"\" +
18       pictures[ Math.floor( Math.random() * 7 ) ] +
19       ".gif\" width = \"105\" height = \"100\" /> " );
20     // -->
21   </script>
22
23 </head>
24
25 <body>
26   <p>Click Refresh (or Reload) to run the script again</p>
27 </body>
28 </html >
```

## Esecuzione



## Riferimenti e Parametri per Riferimento

- Due modalità per il passaggio di parametri
  - Per valore (**by-value**)
    - Si passa una copia del valore originale
    - È la modalità di default per i valori numerici e booleani
    - Il valore originale della variabile **non cambia**
  - Per riferimento (**by-reference**)
    - Si passa l'indirizzo di memoria del valore
    - Permette l'accesso diretto al valore originale
    - Migliora le prestazioni

## Passaggio di Array a Function

- Il nome dell'array è l'argomento
  - Non è necessario passare la dimensione dell'array
    - Gli array conoscono la propria dimensione
  - È passato by reference
    - Elementi singoli sono passati by value se sono valori numerici o booleani
- Array.join
  - Crea una stringa con tutti gli elementi dell'array

```
1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 11.8: PassArray.html -->
6 <!-- Passing Arrays -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10    <title>Passing Arrays and Individual Array
11      Elements to Functions</title>
12
13    <script type = "text/javascript">
14      <!--
15      function start()
16      {
17        var a = [ 1, 2, 3, 4, 5 ];
18
19        document.writeln( "<h2>Effects of passing entire " +
20          "array call-by-reference</h2>" );
21        outputArray(
22          "The values of the original array are: ", a );
23
24        modifyArray( a ); // array a passed call-by-reference
25      }
26    </script>
27  </head>
28  <body>
29  </body>
30 </html>
```

La prima chiamata a outputArray mostra il contenuto dell'array a prima della modifica.

La function modifyArray raddoppia ogni elemento.



```

26 outputArray(
27     "The values of the modified array are: ", a );
28
29 document.writeln( "<h2>Effects of passing array " +
30     "element call-by-value</h2>" +
31     "a[3] before modifyElement: " + a[ 3 ] );
32
33 modifyElement( a[ 3 ] );
34
35 document.writeln(
36     "<br />a[3] after modifyElement: " + a[ 3 ] );
37 }
38
39 // outputs "header" followed by the contents of "theArray"
40 function outputArray( header, theArray )
41 {
42     document.writeln(
43         header + theArray.join( " " ) + "<br />" );
44 }
45

```

JavaScript: Array

33

```

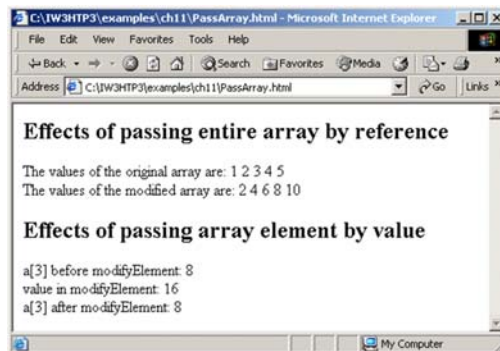
46 // function that modifies the elements of an array
47 function modifyArray( theArray )
48 {
49     for ( var j in theArray )
50         theArray[ j ] *= 2;
51 }
52
53 // function that attempts to modify the value passed
54 function modifyElement( e )
55 {
56     e *= 2;
57     document.writeln( "<br />value in modifyElement: " + e );
58 }
59 // -->
60 </script>
61
62 </head><body onload = "start()"></body>
63 </html >

```

JavaScript: Array

34

## Esecuzione



JavaScript: Array

35

## Ordinamento di Array

- Array.sort
  - Strumento per il confronto tra stringhe
  - Funzione che restituisce un valore
    - Negativo se il primo argomento è minore del secondo
    - Zero se gli argomenti sono uguali
    - Positivo se il primo argomento è maggiore del secondo

JavaScript: Array

36

```

1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 11.9: sort.html -->
6 <!-- Sorting an Array -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9   <head>
10    <title>Sorting an Array with Array Method sort</title>
11
12    <script type = "text/javascript">
13      <!--
14      function start()
15      {
16        var a = [ 10, 1, 9, 2, 8, 3, 7, 4, 6, 5 ];
17
18        document.writeln( "<h1>Sorting an Array</h1>" );
19        outputArray( "Data Items in original order: ", a );
20        a.sort( compareIntegers ); // sort the array
21        outputArray( "Data Items in ascending order: ", a );
22      }

```

Method sort takes as its optional argument the name of a function that compares two arguments and returns a value of -1, 0 or 1.

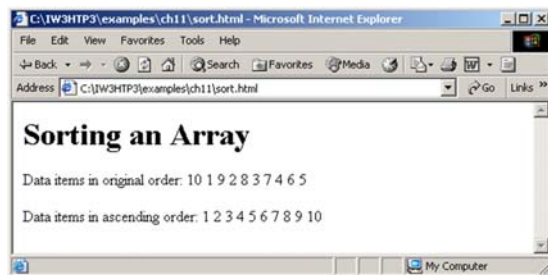
```

23
24 // outputs "header" followed by the contents of "theArray"
25 function outputArray( header, theArray )
26 {
27   document.writeln( "<p>" + header +
28     theArray.join( " " ) + "</p>" );
29 }
30
31 // comparison function for use with sort
32 function compareIntegers( value1, value2 )
33 {
34   return parseInt( value1 ) - parseInt( value2 );
35 }
36 // -->
37 </script>
38
39 </head><body onload = "start()"></body>
40 </html >

```

Function compareIntegers calculates the difference between the integer values of its arguments.

## Esecuzione



## Ricerca Elementi in un Array

- Ricerca Lineare
  - Si confrontano tutti gli elementi con quello da cercare, fino a quando non si trova corrispondenza
  - Nel caso peggiore si deve scandire l'intero array
- Ricerca Binaria
  - Gli elementi devono essere ordinati
  - Ad ogni iterazione l'intervallo di ricerca è dimezzato

```

1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 11.10: LinearSearch.html -->
6 <!-- Linear Search of an Array -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9 <head>
10 <title>Linear Search of an Array</title>
11
12 <script type = "text/javascript">
13 <!--
14 var a = new Array( 100 ); // create an Array
15
16 // fill Array with even Integer values from 0 to 198
17 for ( var i = 0; i < a.length; ++i )
18     a[ i ] = 2 * i;
19

```

JavaScript: Array

41

```

20 // function called when "Search" button is pressed
21 function buttonPressed()
22 {
23     var searchKey = searchForm.inputVal.value;
24
25     // Array a is passed to linearSearch even though it
26     // is a global variable. Normally an array will
27     // be passed to a method for searching.
28     var element = linearSearch( a, parseInt( searchKey ) );
29
30     if ( element != -1 )
31         searchForm.result.value =
32             "Found value in element " + element;
33     else
34         searchForm.result.value = "Value not found";
35 }
36

```

JavaScript: Array

42

```

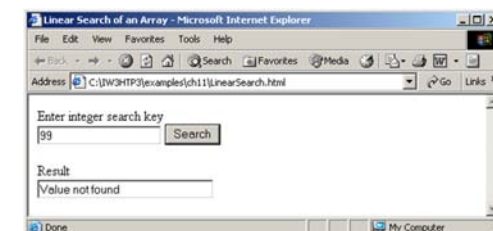
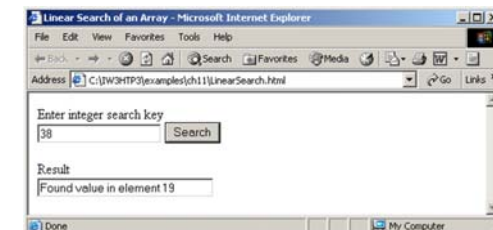
37 // Search "theArray" for the specified "key" value
38 function linearSearch( theArray, key )
39 {
40     for ( var n = 0; n < theArray.length; ++n )
41         if ( theArray[ n ] == key )
42             return n;
43
44     return -1;
45 }
46 // -->
47 </script>
48
49 </head>
50
51 <body>
52 <form name = "searchForm" action = "">
53 <p>Enter Integer search key<br />
54 <input name = "inputVal" type = "text" />
55 <input name = "search" type = "button" value = "Search"
56     onclick = "buttonPressed()" /><br /></p>
57
58 <p>Result<br />
59 <input name = "result" type = "text" size = "30" /></p>
60 </form>
61 </body>
62 </html>

```

JavaScript: Array

43

## Esecuzione



JavaScript: Array

44

```

1 <?xml version = "1.0"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4
5 <!-- Fig. 11.11 : BinarySearch.html -->
6 <!-- Binary search -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9 <head>
10 <title>Binary Search</title>
11
12 <script type = "text/javascript">
13 <!--
14 var a = new Array( 15 );
15
16 for ( var i = 0; i < a.length; ++i )
17     a[ i ] = 2 * i;
18

```

JavaScript: Array

45

```

19 // function called when "Search" button is pressed
20 function buttonPressed()
21 {
22     var searchKey = searchForm.inputVal.value;
23
24     searchForm.result.value =
25         "Portions of array searched\n";
26
27     // Array a is passed to binarySearch even though it
28     // is a global variable. This is done because
29     // normally an array is passed to a method
30     // for searching.
31     var element =
32         binarySearch( a, parseInt( searchKey ) );
33
34     if ( element != -1 )
35         searchForm.result.value +=
36             "\nFound value in element " + element;
37     else
38         searchForm.result.value += "\nValue not found";
39 }
40

```

JavaScript: Array

46

```

41 // Binary search
42 function binarySearch( theArray, key )
43 {
44     var low = 0; // low subscript
45     var high = theArray.length - 1; // high subscript
46     var middle; // middle subscript
47
48     while ( low <= high ) {
49         middle = ( low + high ) / 2;
50
51         // The following line is used to display the
52         // part of theArray currently being manipulated
53         // during each iteration of the binary
54         // search loop.
55         buildOutput( theArray, low, middle, high );
56
57         if ( key == theArray[ middle ] ) // match
58             return middle;
59         else if ( key < theArray[ middle ] )
60             high = middle - 1; // search low end of array
61         else
62             low = middle + 1; // search high end of array
63     }

```

JavaScript: Array

47

```

64
65     return -1; // searchKey not found
66 }
67
68 // Build one row of output showing the current
69 // part of the array being processed.
70 function buildOutput( theArray, low, mid, high )
71 {
72     for ( var i = 0; i < theArray.length; i++ ) {
73         if ( i < low || i > high )
74             searchForm.result.value += " ";
75         // mark middle element in output
76         else if ( i == mid )
77             searchForm.result.value += theArray[ i ] +
78                 ( theArray[ i ] < 10 ? " " : " " );
79         else
80             searchForm.result.value += theArray[ i ] +
81                 ( theArray[ i ] < 10 ? " " : " " );
82     }
83
84     searchForm.result.value += "\n";
85 }
86 // -->
87 </script>
88 </head>
89

```

JavaScript: Array

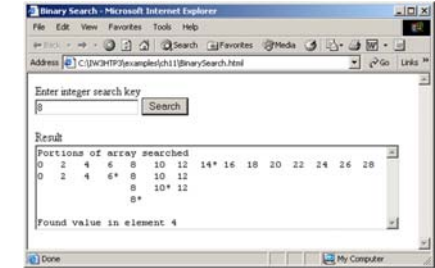
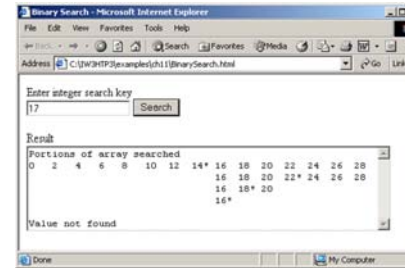
48

```

90 <body>
91   <form name = "searchForm" action = "">
92     <p>Enter Integer search key<br />
93     <input name = "inputVal" type = "text" />
94     <input name = "search" type = "button" value =
95       "Search" onclick = "buttonPressed()" /><br /></p>
96     <p>Result<br />
97     <textarea name = "result" rows = "7" cols = "60">
98   </textarea></p>
99 </form>
100 </body>
101 </html >

```

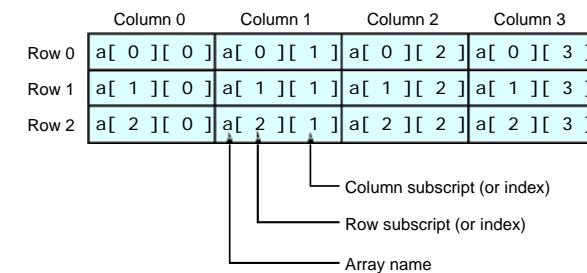
## Esecuzione



## Array Multidimensionali (1)

- Array bidimensionali sono analoghi alle matrici
  - Righe e colonne
    - Si specifica prima la riga e poi la colonna
  - Due indici

## Array Multidimensionali (2)



## Array Multidimensionali (3)

- Dichiarazione e inizializzazione di array multidimensionali
  - Elementi raggruppati per righe in parentesi quadre
  - Devono essere gestiti come array di array
  - Per creare l'array b con una riga di due elementi e una seconda riga di tre elementi:
    - `var b = [ [ 1, 2 ], [ 3, 4, 5 ] ];`

## Array Multidimensionali (4)

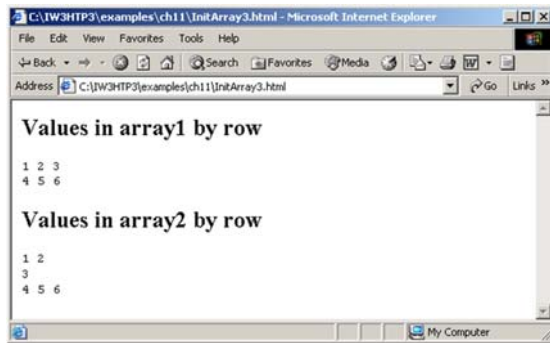
- È anche possibile usare l'operatore `new`
  - Crea l'array `b` con due righe, la prima con 5 colonne e la seconda con tre:

```
var b;  
b = new Array( 2 );  
b[ 0 ] = new Array( 5 );  
b[ 1 ] = new Array( 3 );
```

```
1 <?xml version = "1.0"?>  
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
4  
5 <!-- Fig. 11.13: Inizializzazione di array multidimensionali -->  
6 <!-- Inizializzazione di array multidimensionali -->  
7  
8 <html xmlns = "http://www.w3.org/1999/xhtml">  
9 <head>  
10 <title>Inizializzazione di array multidimensionali</title>  
11  
12 <script type = "text/javascript">  
13 <!--  
14 function start()  
15 {  
16     var array1 = [ [ 1, 2, 3 ], // first row  
17                  [ 4, 5, 6 ] ]; // second row  
18     var array2 = [ [ 1, 2 ], // first row  
19                  [ 3 ], // second row  
20                  [ 4, 5, 6 ] ]; // third row  
21  
22     outputArray( "Valori in array1 per riga", array1 );  
23     outputArray( "Valori in array2 per riga", array2 );  
24 }
```

```
25  
26 function outputArray( header, theArray )  
27 {  
28     document.writeln( "<h2>" + header + "</h2><tt>" );  
29  
30     for ( var i in theArray ) {  
31  
32         for ( var j in theArray[ i ] )  
33             document.writeln( theArray[ i ][ j ] + " " );  
34  
35         document.writeln( "<br />" );  
36     }  
37  
38     document.writeln( "</tt>" );  
39 }  
40 // -->  
41 </script>  
42  
43 </head><body onload = "start()"></body>  
44 </html >
```

# Esecuzione



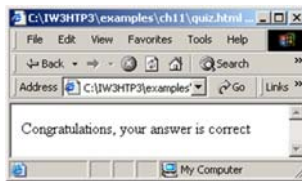
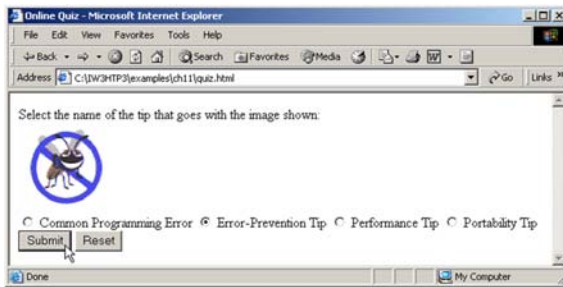
# Esempio: Quiz Online

- Radio button
  - Rappresentati da un array
    - Il nome dei radio button è il nome dell'array
    - Un elemento per ogni button
  - La proprietà checked è true quando quel button è selezionato
- Il form XHTML
  - Contiene controllo, tra cui i radio button
  - La proprietà action specifica cosa succede quando il form viene inviato

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
3   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
4
5 <!-- Fig. 11.14: quiz.html -->
6 <!-- Online Quiz -->
7
8 <html xmlns = "http://www.w3.org/1999/xhtml">
9 <head>
10 <title>Online Quiz</title>
11
12 <script type = "text/JavaScript">
13
14   function checkAnswers()
15   {
16     // determine whether the answer is correct
17     if ( myQuiz.radioButton[ 1 ].checked )
18       document.write( "Congratulations, your answer is correct" );
19     else // If the answer is incorrect
20       document.write( "Your answer is incorrect. Please try again" );
21   }
22
23 </script>
24
25 </head>
```

```
26
27 <body>
28   <form id = "myQuiz" action = "JavaScript:checkAnswers()">
29     <p>Select the name of the tip that goes with the image shown: <br />
30     
31     <br />
32
33     <input type = "radio" name = "radiobutton" value = "CPE" />
34     <label>Common Programming Error</label>
35
36     <input type = "radio" name = "radiobutton" value = "EPT" />
37     <label>Error-Prevention Tip</label>
38
39     <input type = "radio" name = "radiobutton" value = "PERF" />
40     <label>Performance Tip</label>
41
42     <input type = "radio" name = "radiobutton" value = "PORT" />
43     <label>Portability Tip</label><br />
44
45     <input type = "submit" name = "submit" value = "Submit" />
46     <input type = "reset" name = "reset" value = "Reset" />
47   </p>
48 </form>
49 </body>
50 </html>
```

# Esecuzione



JavaScript: Array