Chapter 10 - JavaScript: Functions

Outline

10.4 Function Definitions

- Format of a function definition

```
function function-name( parameter-list )
{
  declarations and statements
}
```

  - Function name any valid identifier
  - Parameter list names of variables that will receive arguments
    - Must have same number as function call
    - May be empty
  - Declarations and statements
    - Function body ("block" of code)

10.4 Function Definitions

- Returning control
  - return statement
  - Can return either nothing, or a value

    return *expression*;

  - No return statement same as return;
  - Not returning a value when expected is an error

10.4 Function Definitions

- Writing a function to square two numbers
  - for loop from 1 to 10
  - Pass each number as argument to square
  - return value of argument multiplied by itself
  - Display result

SquareInt.html
(1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 10.2: SquareInt.html -->
<!-- Square function           -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>A Programmer-Defined square Function</title>

      <script type = "text/javascript">
         <!--
         document.writeln(
            "<h1>Square the number

         // square the numbers from 1 to 10
         for ( var x = 1; x <= 10; ++x )
            document.writeln( "The square of " + x + " is " +
               square( x ) + "<br />" );

```

Calling function square and passing it the value of x.

```
      // The following square function's body is executed
      // only when the functi

      // square function definition
      function square( y )
      {
         return y * y;
      }
      // -->
    </script>

  </head><body></body>
</html>
```

Variable y gets the value of variable x.

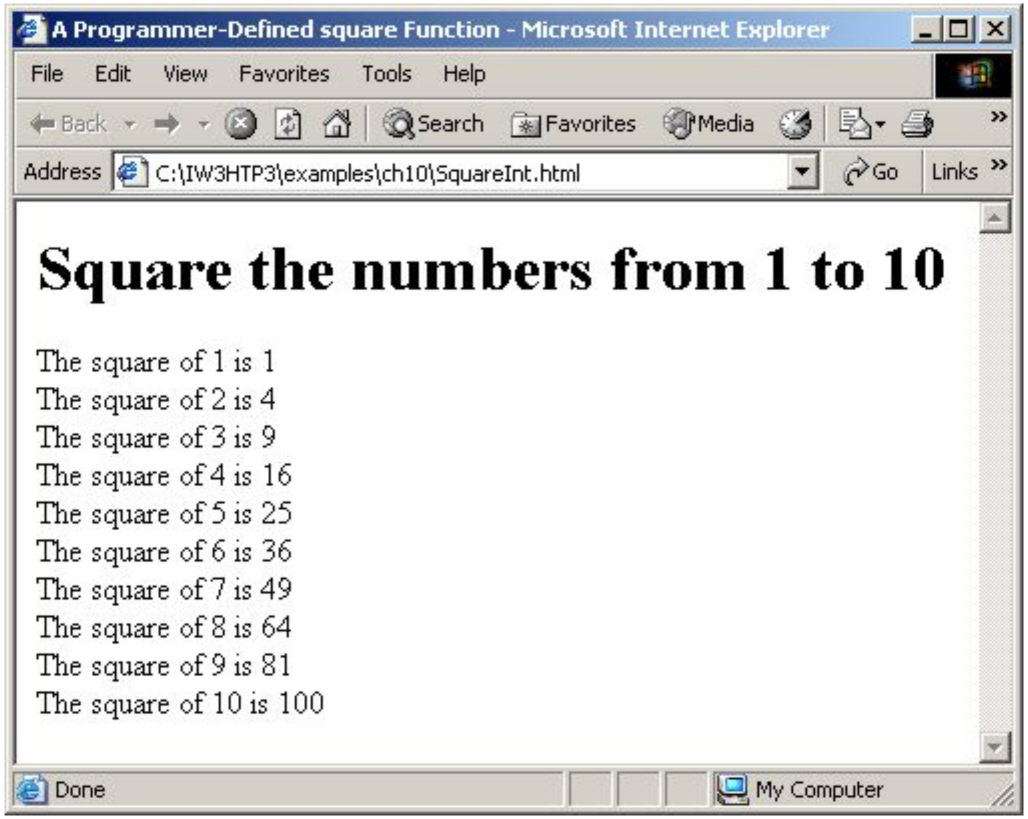
The return statement passes the value of y * y back to the calling function.

Outline

SquareInt.html (2 of 2)

10.4 Function Definitions

Fig. 10.2 Using programmer-defined function square.


A Programmer-Defined square Function - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch10\SquareInt.html
Go Links
Done
My Computer

```
Square the numbers from 1 to 10

The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
The square of 6 is 36
The square of 7 is 49
The square of 8 is 64
The square of 9 is 81
The square of 10 is 100
```

10.4 Function Definitions

- Finding the maximum of 3 numbers
  - Prompt for 3 inputs
  - Convert to numbers
  - Pass to maximum
  - Math.max

Outline

Maximum.html (1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 10.3: maximum.html -->
<!-- Maximum function        -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Finding the Maximum of Three Values</title>

      <script type = "text/javascript">
         <!--
         var input1 =
            window.prompt( "Enter first number", "0" );
         var input2 =
            window.prompt( "Enter second number", "0" );
         var input3 =
            window.prompt( "Enter third number", "0" );

         var value1 = parseFloat( input1 );
         var value2 = parseFloat( input2 );
         var value3 = parseFloat( input3 );
```

Prompt for the user to input three integers.

Outline

m.html

(2 of 2)

```
24
25          var maxValue = maximum( value1, value2, value3 );
26
27          document.writeln( "First number: "
28             "<br />Second number: " + value
29             "<br />Third number: " + value3 +
30             "<br />Maximum is: " + maxValue
31
32          // maximum method definition (called from line 25)
33          function maximum( x, y, z )
34          {
35             return Math.max( x, Math.max
36          }
37          // -->
38       </script>
39
40    </head>
41    <body>
42       <p>Click Refresh (or Reload) to run the script again</p>
43    </body>
44 </html>
```

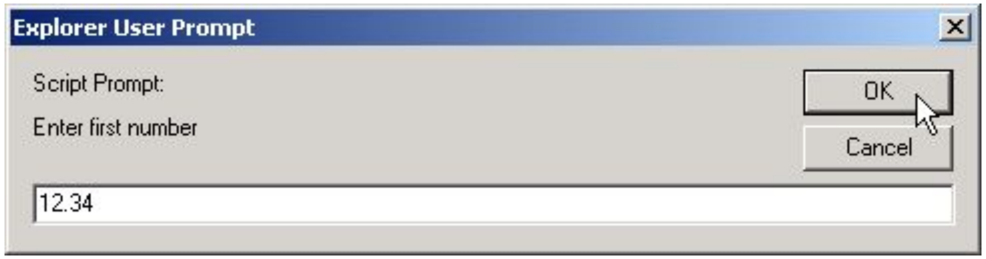
Call function maximum and pass it the value of variables value1, value2 and value3.

Method max returns the larger of the two integers passed to it.

Variables x, y and z get the value of variables value1, value2 and value3, respectively.

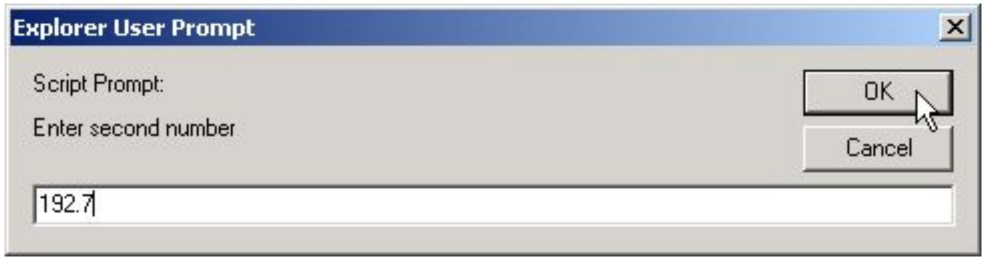
10.4 Function Definitions

Fig. 10.3 Programmer-defined maximum function (1 of 2).



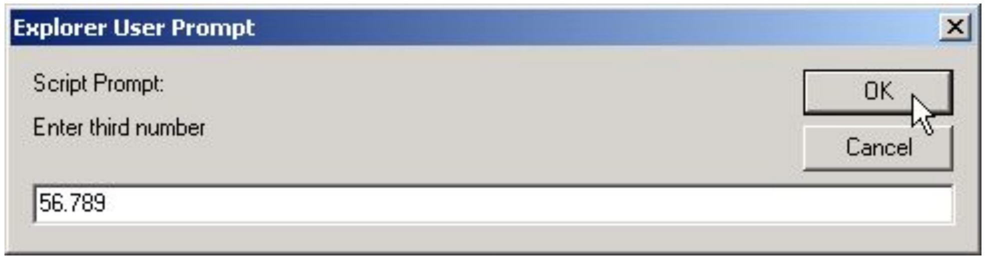
Script Prompt:
Enter first number

12.34



Script Prompt:
Enter second number

192.7

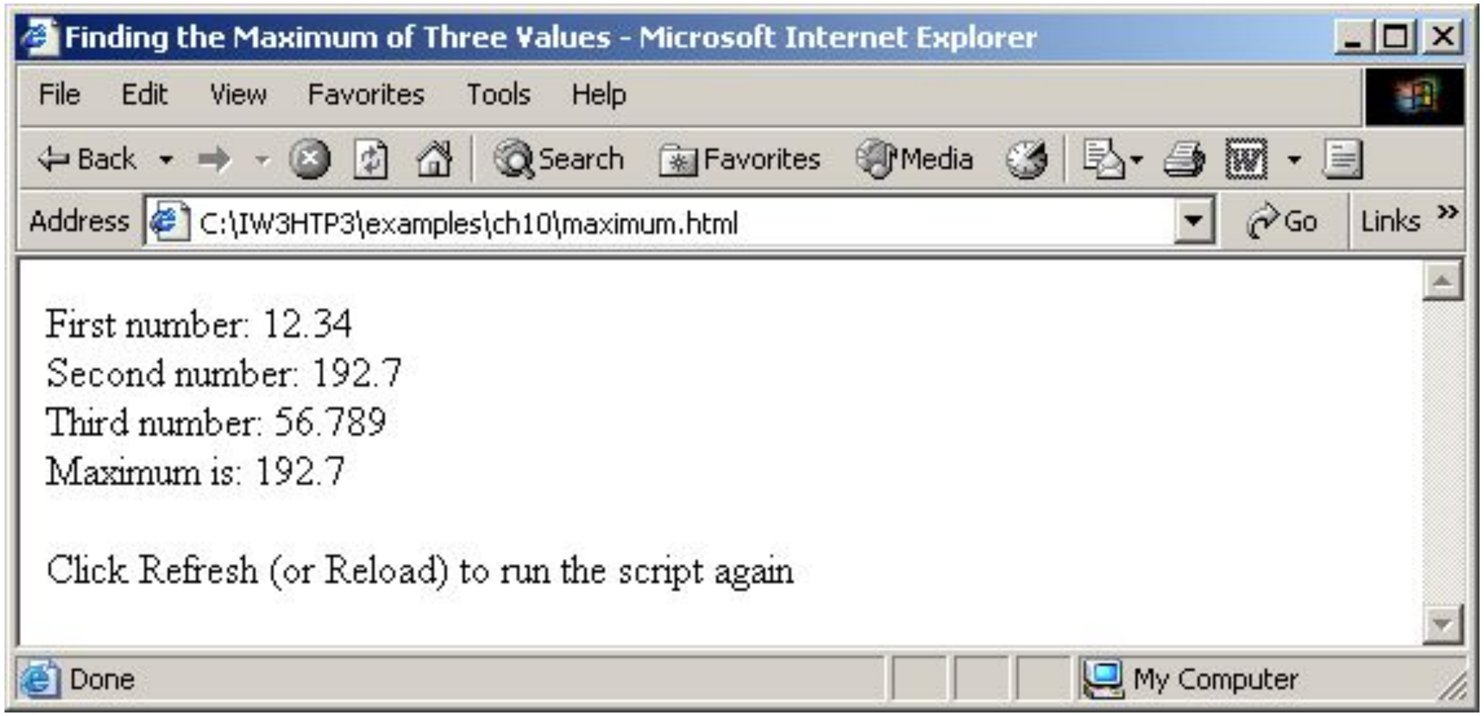


Script Prompt:
Enter third number

56.789

10.4 Function Definitions

Fig. 10.3 Programmer-defined maximum function (2 of 2).


Finding the Maximum of Three Values - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch10\maximum.html
Go Links
Done
My Computer

```
First number: 12.34
Second number: 192.7
Third number: 56.789
Maximum is: 192.7

Click Refresh (or Reload) to run the script again
```

10.6 Example: Game of Chance

- Craps
  - Click **Roll Dice**
  - Text fields show rolls, sum and point
  - Status bar displays results

10.6 Example: Game of Chance

- Uses XHTML forms
  - Gather multiple inputs at once
  - Empty action attribute
  - name attribute allows scripts to interact with form
- Event handling and event-driven programming
  - Assign a function to an event
  - Onclick
- Constants
  - Variable that cannot be modified
  - Part of many languages, not supported in JavaScript
    - Name "constant" variables with all capital letters
  - Make values easier to remember/change

10.6 Example: Game of Chance

- Changing properties
  - Access with dot (.) notation
  - value property of text fields
  - status property of window

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 10.6: Craps.html -->
<!-- Craps Program         -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Program that Simulates the Game of Craps</title>

      <script type = "text/javascript">
         <!--
         // variables used to test the state of the game
         var WON = 0, LOST = 1, CONTINUE_ROLLING = 2;

         // other variables used in program
         var firstRoll = true,            // true if first roll
             sumOfDice = 0,               // sum of the dice
             myPoint = 0, // point if no win/loss on first roll
             gameStatus = CONTINUE_ROLLING; // game not over yet

```

Outline

Craps.html
(1 of 5)

Outline

```
      // process one roll of the dice
      function play()
      {
         if ( firstRoll ) {              // fi
            sumOfDice = rollDice();

            switch ( sumOfDice ) {
               case 7: case 11:              // win on fi
                  gameStatus = WON;
                  // clear point field
                  document.craps.point.value = "";
                  break;
               case 2: case 3: case 12: // lose on first roll
                  gameStatus = LOST;
                  // clear point field
                  document.craps.point.value =
                  break;
               default:                       // re
                  gameStatus = CONTINUE_ROLLING;
                  myPoint = sumOfDice;
                  document.craps.point.value = myPoint;
            }
            firstRoll = false;
         }
      }
```

If the value of firstRoll is true, then function rollDice is called.

If function rollDice returns a value of 7 or 11, the player wins and the break statement causes program control proceeds to the first line after the switch structure.

If function rollDice returns a 2, 3 or 12, the player loses and the break statement causes control to proceed to first line after the switch structure.

Outline

tml

```
            else {
               sumOfDice = rollDice();

               if ( sumOfDice == myPoint ) // win by making point
                  gameStatus = WON;
               else
                  if ( sumOfDice == 7 )    // l
                     gameStatus = LOST;
            }

            if ( gameStatus == CONTINUE_ROLLING )
               window.status = "Roll again";
            else {
               if ( gameStatus == WON )
                  window.status = "Player wins. 
                     "Click Roll Dice to play ag
               else
                  window.status = "Player loses. " +
                     "Click Roll Dice to play again.";

               firstRoll = true;
            }
         }

```

If the value of firstRoll is false, function rollDice is called to see if the point has been reached.

If the values returned by function rollDice equals 7, the player loses.

If the value returned by function rollDice equals the value of variable myPoint, the player wins because the point has been reached.

window method status displays a message in the status bar of the browser.

```
      // roll the dice
      function rollDice()
      {
         var die1, die2, workSum;

         die1 = Math.floor( 1
         die2 = Math.floor( 1 + Math.random() * 6 );
         workSum = die1 + die2;

         document.craps.firstDie.value = d
         document.craps.secondDie.value =
         document.craps.sumvalue = workSum;

         return workSum;
      }
      // -->
   </script>

</head>
```

Function rollDice is called to simulate the rolling of two dice on the craps table.

Methods random and floor are used to generate the values for the two dice.

Referencing the names of form elements in the XHTML document, the values of the dice are placed in their respective form fields.

Outline

Craps.html (4 of 5)

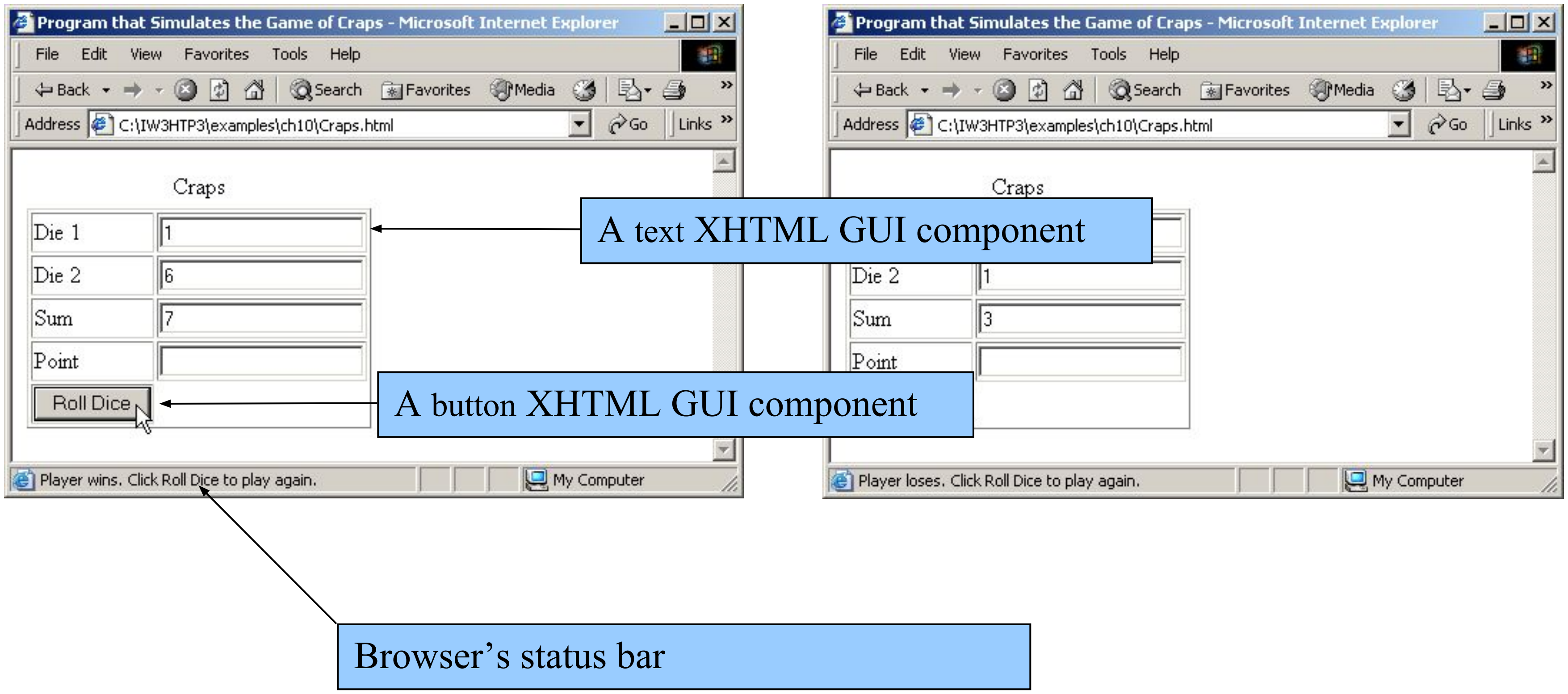
```
   <body>
      <form name = "craps" action = "">
         <table border = "1">
         <caption>Craps</caption>
         <tr><td>Die 1</td>
             <td><input name = "firstDie" type = "text" />
             </td></tr>
         <tr><td>Die 2</td>
             <td><input name = "secondDie" type = "text" />
             </td></tr>
         <tr><td>Sum</td>
             <td><input name = "sum" type = "text" />
             </td></tr>
         <tr><td>Point</td>
             <td><input name = "point" type = "text" />
             </td></tr>
         <tr><td><input type = "button" value = "Roll Dice"
             onclick = "play()" /></td></tr>
         </table>
      </form>
   </body>
</html>
```

Outline

Craps.html
(5 of 5)

10.6 Example: Game of Chance

Fig. 10.6 Craps game simulation.


Program that Simulates the Game of Craps - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch10\Craps.html
Go Links
Craps
A text XHTML GUI component
A button XHTML GUI component
Browser's status bar
My Computer

Die 1	1
Die 2	6
Sum	7
Point	

Roll Dice

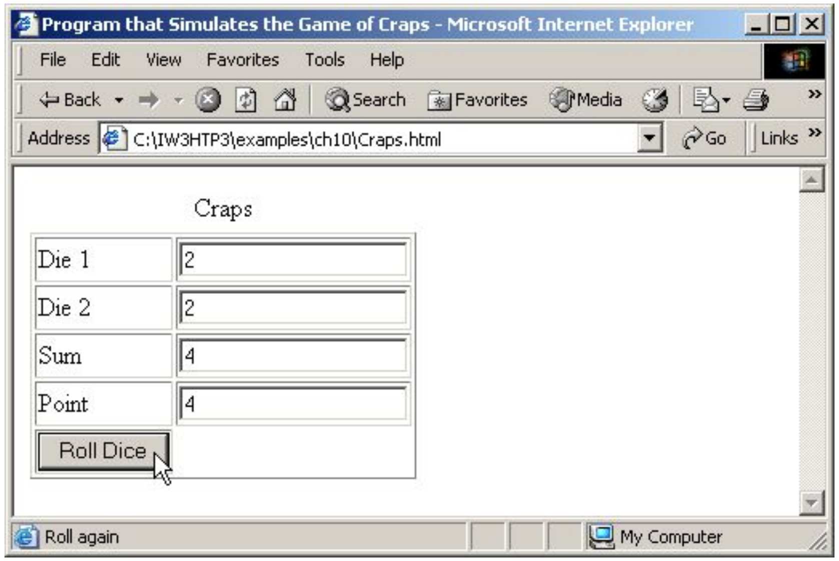
Player wins. Click Roll Dice to play again.

Die 2	1
Sum	3
Point	

Player loses. Click Roll Dice to play again.

10.6 Example: Game of Chance

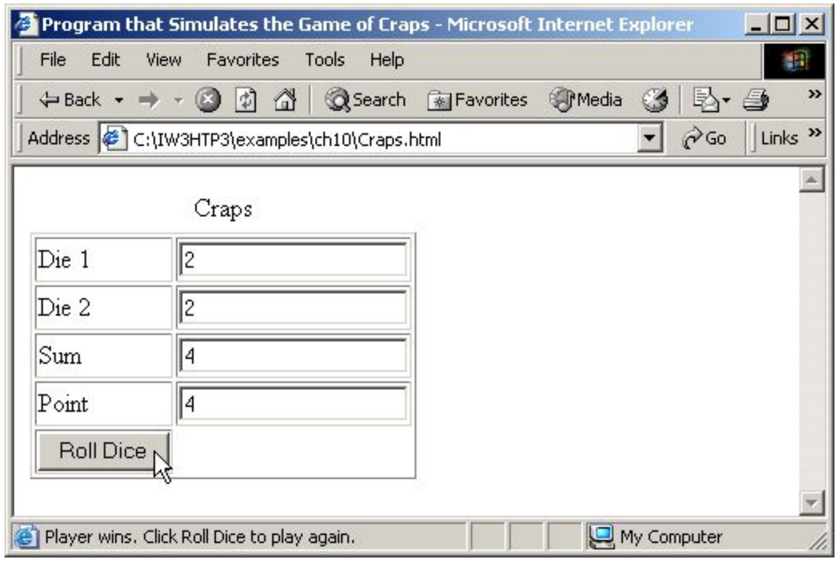
Fig. 10.6 Craps game simulation.


Program that Simulates the Game of Craps - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch10\Craps.html Go Links
Roll Dice
My Computer

Craps

Die 1	2
Die 2	2
Sum	4
Point	4

Roll again


Program that Simulates the Game of Craps - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch10\Craps.html Go Links
Roll Dice
My Computer

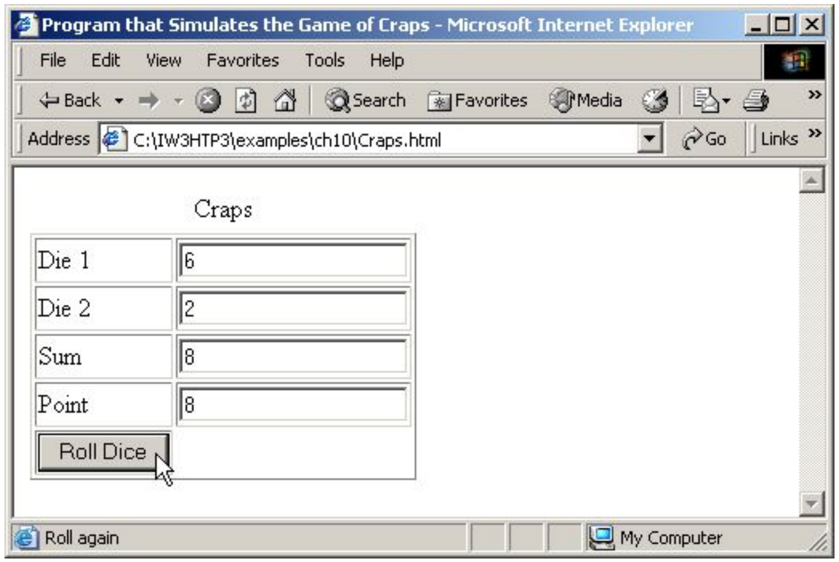
Craps

Die 1	2
Die 2	2
Sum	4
Point	4

Player wins. Click Roll Dice to play again.

10.6 Example: Game of Chance

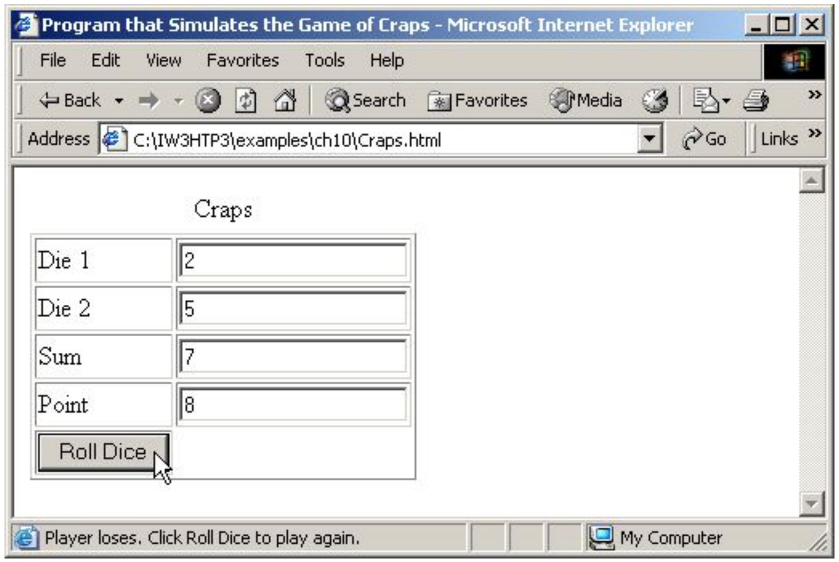
Fig. 10.6   Craps game simulation.


Program that Simulates the Game of Craps - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch10\Craps.html
Go Links
Roll Dice
My Computer

Craps

Die 1	6
Die 2	2
Sum	8
Point	8

Roll again


Program that Simulates the Game of Craps - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch10\Craps.html
Go Links
Roll Dice
My Computer

Craps

Die 1	2
Die 2	5
Sum	7
Point	8

Player loses. Click Roll Dice to play again.

10.9 JavaScript Global Functions

- Global object
  - Always available
  - Provides 7 methods
  - Do not need to explicitly reference Global before method call
  - Also holds all global variables, user defined functions

10.9 JavaScript Global Functions

Global function	Description
escape	This function takes a string argument and returns a string in which all spaces, punctuation, accent characters and any other character that is not in the ASCII character set (see Appendix D, ASCII Character Set) are encoded in a hexadecimal format (see Appendix E, Number Systems) that can be represented on all platforms.
eval	This function takes a string argument representing JavaScript code to execute. The JavaScript interpreter evaluates the code and executes it when the eval function is called. This function allows JavaScript code to be stored as strings and executed dynamically.
isFinite	This function takes a numeric argument and returns true if the value of the argument is not NaN, Number.POSITIVE_INFINITY or Number.NEGATIVE_INFINITY; otherwise, the function returns false.
isNaN	This function takes a numeric argument and returns true if the value of the argument is not a number; otherwise, it returns false. The function is commonly used with the return value of parseInt or parseFloat to determine whether the result is a proper numeric value.

Fig. 10.9 JavaScript global functions.

10.9 JavaScript Global Functions

Global function	Description
`parseFloat`	This function takes a string argument and attempts to convert the beginning of the string into a floating-point value. If the conversion is unsuccessful, the function returns `NaN`; otherwise, it returns the converted value (e.g., `parseFloat( "abc123.45" )` returns `NaN`, and `parseFloat( "123.45abc" )` returns the value `123.45`).
`parseInt`	This function takes a string argument and attempts to convert the beginning of the string into an integer value. If the conversion is unsuccessful, the function returns `NaN`; otherwise, it returns the converted value (e.g., `parseInt( "abc123" )` returns `NaN`, and `parseInt( "123abc" )` returns the integer value `123`). This function takes an optional second argument, from 2 to 36, specifying the **radix** (or **base**) of the number. Base 2 indicates that the first argument string is in **binary** format, base 8 indicates that the first argument string is in **octal** format and base 16 indicates that the first argument string is in **hexadecimal** format. See see Appendex E, Number Systems, for more information on binary, octal and hexadecimal numbers.
`unescape`	This function takes a string as its argument and returns a string in which all characters previously encoded with `escape` are decoded.

Fig. 10.9 JavaScript global functions.

10.10 Recursion

- Recursive functions
  - Call themselves
    - Recursion step or recursive call
    - Part of return statement
  - Must have base case
    - Simplest case of problem
    - Returns value rather than calling itself
  - Each recursive call simplifies input
    - When simplified to base case, functions return

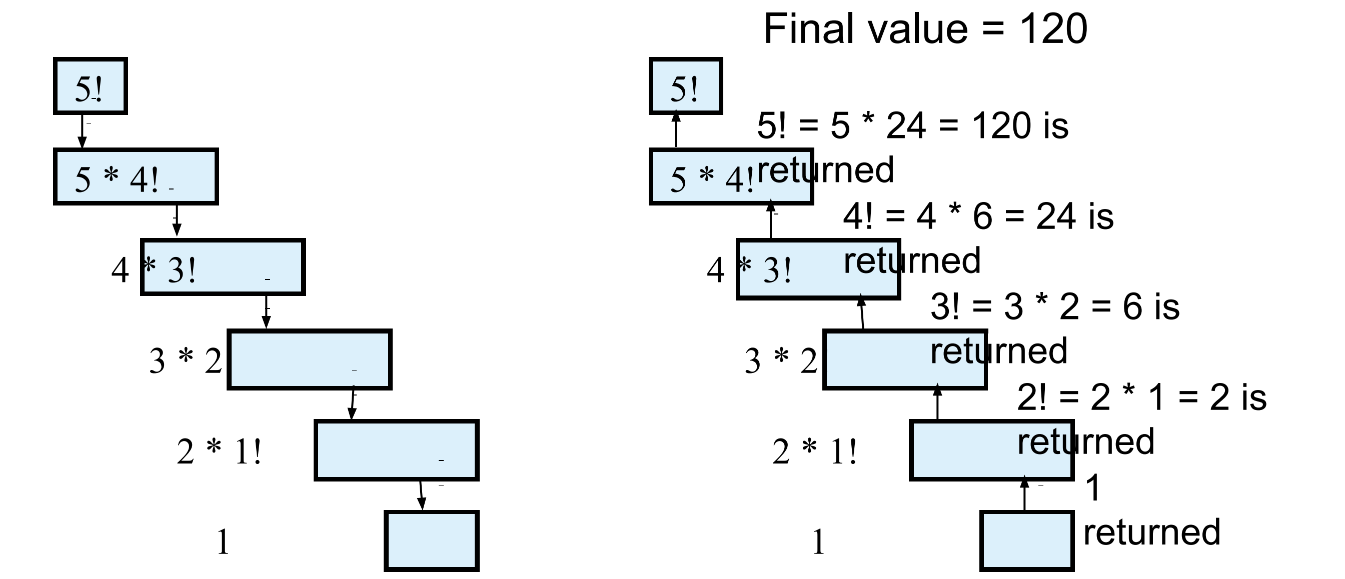
10.10 Recursion

- Factorials
  - Product of calculation $n \cdot (n - 1) \cdot (n - 2) \cdot \ldots \cdot 1$
  - Iterative approach:

```
var factorial = 1;

for ( var counter = number; counter >= 1; --counter )
   factorial *= counter;
```

  - Note each factor is one less than previous factor
    - Stops at 1: base case
    - Perfect candidate for recursive solution

10.10 Recursion



Final value = 120
5!
5 * 4!
4 * 3!
3 * 2
2 * 1!
1
5! = 5 * 24 = 120 is returned
4! = 4 * 6 = 24 is returned
3! = 3 * 2 = 6 is returned
2! = 2 * 1 = 2 is returned
1 returned

(a) Procession of recursive calls.

(b) Values returned from each recursive call.

Fig. 10.10 Recursive evaluation of 5!.

Outline

FactorialTest.html
(1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 10.11: FactorialTest.html -->
<!-- Recursive factorial example    -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Recursive Factorial Function</title>

      <script language = "javascript">
         document.writeln( "<h1>Factorials of
         document.writeln(
            "<table border = '1' width = '100%'>" );

         for ( var i = 0; i <= 10; i++ )
            document.writeln( "<tr><td>" + i + "!</td><td>" +
                        factorial( i ) + "</td></tr>" );

         document.writeln( "</table>" );

```

Calling function factorial and passing it the value of i.

```
        // Recursive definition of function factorial
        function factorial( number )
        {
           if ( number <= 1 )  // base ca
              return 1;
           else
              return number * factorial( number - 1 );
        }
      </script>
   </head><body></body>
</html>
```

Variable number gets the value of variable i.

Call to function factorial and passing it 1 less than the current value of number .

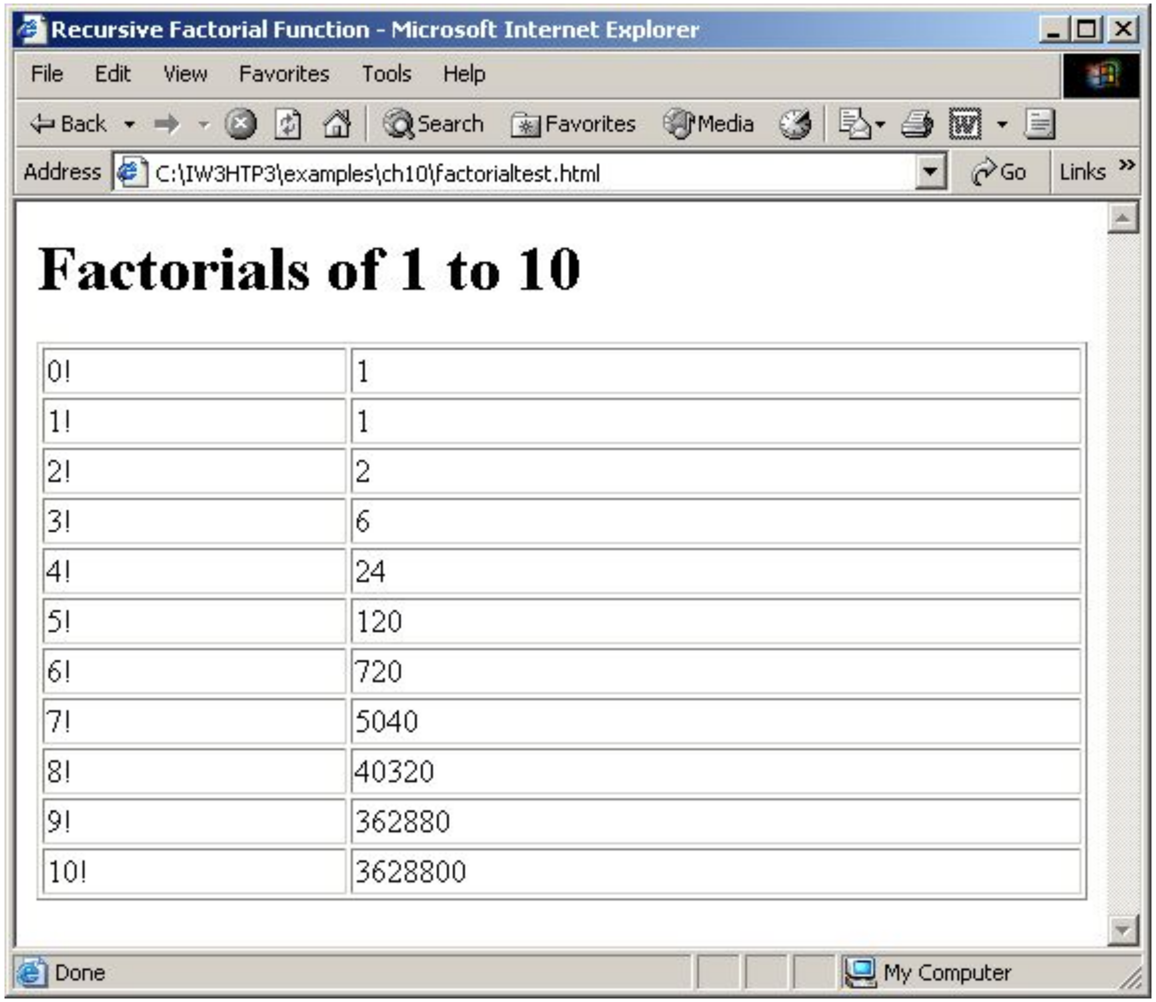
ne

ialTest.html

(2 of 2)

10.10 Recursion

Fig. 10.11 Factorial calculation with a recursive function.


Recursive Factorial Function - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch10\factorialtest.html
Go Links
Done
My Computer

Factorials of 1 to 10

0!	1
1!	1
2!	2
3!	6
4!	24
5!	120
6!	720
7!	5040
8!	40320
9!	362880
10!	3628800

10.11 Recursion vs. Iteration

- Iteration
  - Explicitly uses repetition structures to achieve result
  - Terminates when loop-continuation condition fails
  - Often faster than recursion
- Recursion
  - Repeats through function calls
  - Terminates when base case reached
  - Slower due to function call overhead
    - Each call generates new copy of local variables

Chapter 11 - JavaScript: Arrays

Outline

11.2 Arrays

- Arrays in JavaScript
  - Each element referenced by a number
    - Start at “zeroth element”
    - Subscript or index
  - Accessing a specific element
    - Name of array
    - Brackets
    - Number of element
  - Arrays know their length
    - length property

11.2 Arrays

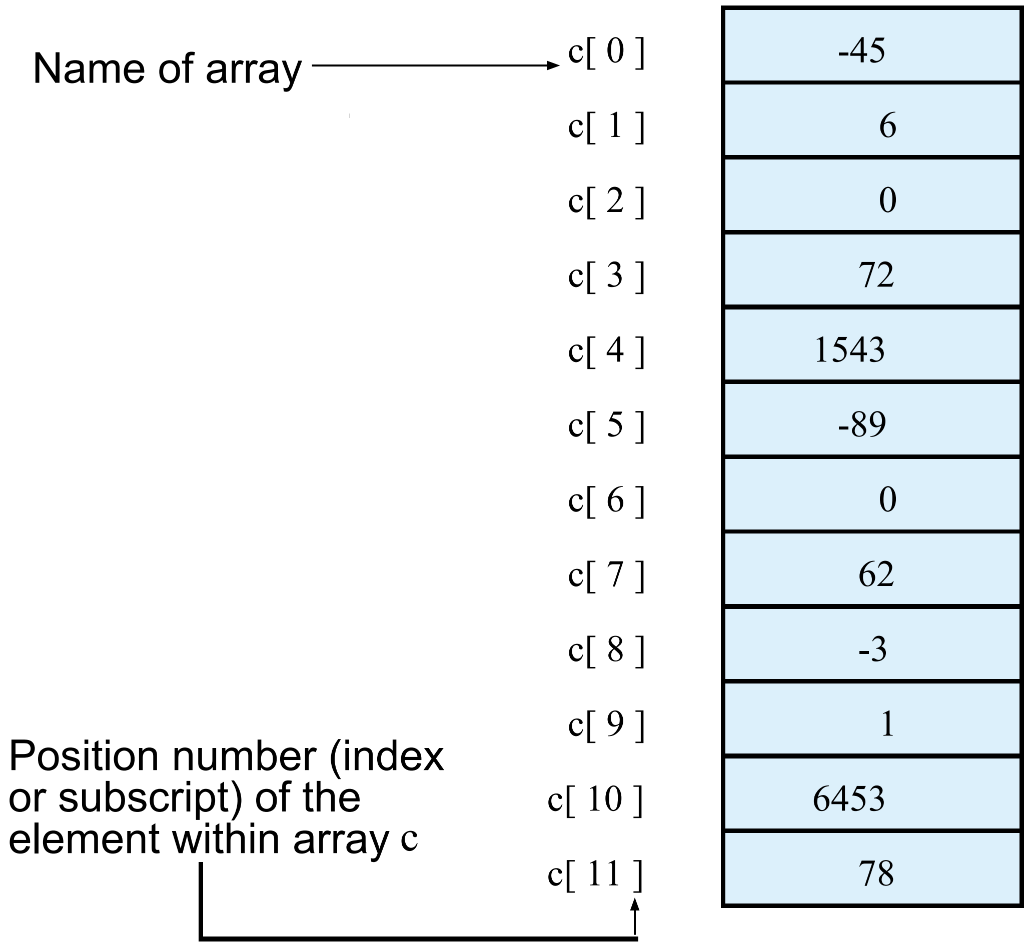


Fig. 11.1 A 12-element array.

11.3 Declaring and Allocating Arrays

- Arrays in memory
  - Objects
  - Operator new
    - Allocates memory for objects
    - Dynamic memory allocation operator

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

11.4 Examples Using Arrays

- Arrays grow dynamically
  - Allocate more space as items are added
- Must initialize array elements
  - Default value is undefined
  - for loops convenient
  - Referring to uninitialized elements or elements outside array bounds is an error

Outline

InitArray.html (1 of 3)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.3: InitArray.html -->
<!-- Initializing an Array     -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Initializing an Array</title>

      <script type = "text/javascript">
         <!--
         // this function is called when the <body> ele
         // onload event occurs
         function initializeArrays()
         {
            var n1 = new Array( 5 );    // allo
            var n2 = new Array();       // allo

            // assign values to each element of Array n1
            for ( var i = 0; i < n1.length; ++i )
               n1[ i ] = i;
```

Array n1 has five elements.

Array n2 is an empty array.

The for loop initializes the elements in n1 to their subscript numbers (0 to 4).

```

            // create and initialize five-elements in Array n2
            for ( i = 0; i < 5; ++i )
               n2[ i ] = i;

            outputArray( "Array n1 contains", n1 );
            outputArray( "Array n2 contains", n2 );
         }

         // output "header" followed by a two-column table
         // containing subscripts and elements of "theArray"
         function outputArray( header, theArray )
         {
            document.writeln( "<h2>" + header + "</h2>" );
            doc



            doc
               align = \"left\">Subscript</th>" +
               "<th align = \"left\">Value</th></thead><tbody>" );
```

The for loop adds fi
each element to its s

Each function displays the contents of its respective Array in an XHTML table.

The second time function ouputArray is called, variable header gets the value of “Array n2 contains” and variable theArray gets the value of n2.

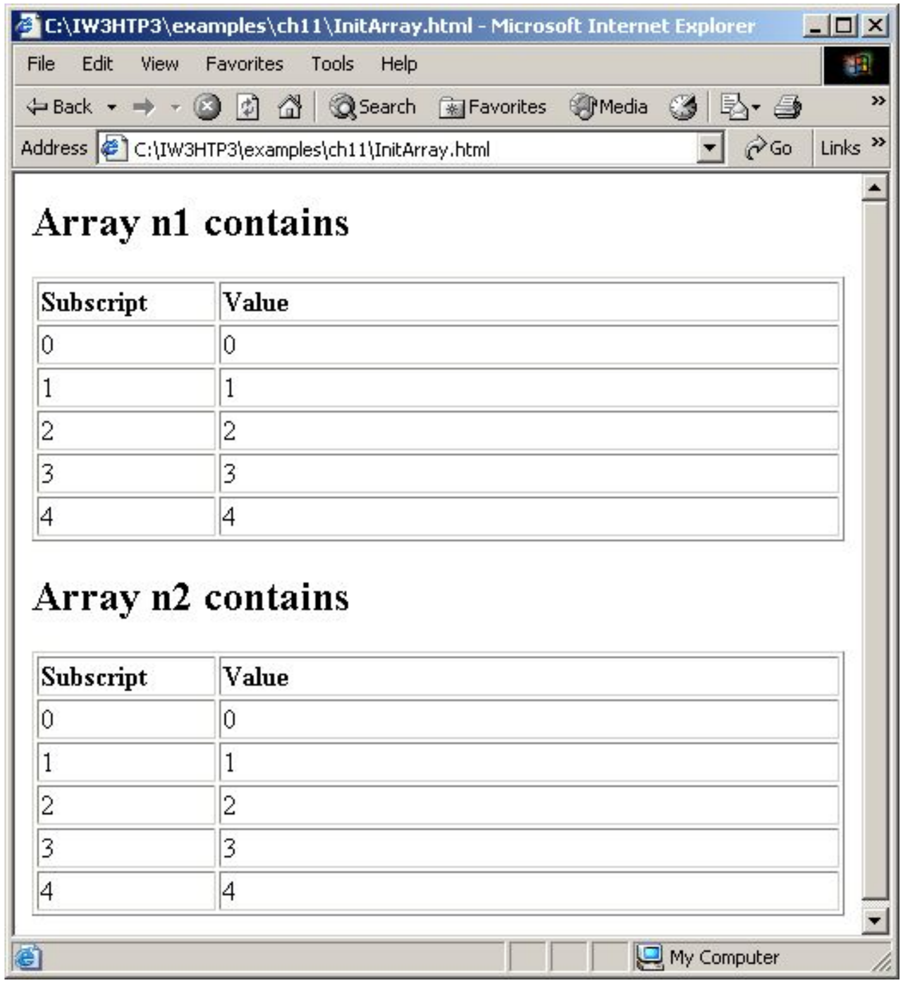
```

         for ( var i = 0; i < theArray.length; i++ )
            document.writeln( "<tr><td>" + i + "</td><td>" +
               theArray[ i ] + "</td></tr>" );

         document.writeln( "</tbody></table>" );
      }
      // -->
   </script>

   </head><body onload = "initializeArrays()"></body>
</html>
```

Outline

InitArray.html (1 of 3)

11.4 Examples Using Arrays

Fig. 11.3 Initializing the elements of an array.


C:\IW3HTP3\examples\ch11\InitArray.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\InitArray.html
Go Links
My Computer

Array n1 contains

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

Array n2 contains

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

11.4 Examples Using Arrays

- Possible to declare and initialize in one step
  - Specify list of values
    - Initializer list

```
  var n = [ 10, 20, 30, 40, 50 ];
var n = new Array( 10, 20, 30, 40, 50 );
```

  - Also possible to only initialize some values
    - Leave uninitialized elements blank
    - Uninitialized elements default to “undefined”

```
  var n = [ 10, 20, , 40, 50 ];
```

InitArray2.html
(1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.4: InitArray2.html                -->
<!-- Initializing an Array with a Declaration -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Initializing an Array with a Declaration</title>

      <script type = "text/javascript"
         <!--
         function start()
         {
            // Initializer list specifies n
            // value for each element.
            var colors = new Array( "cyan", "magenta",
               "yellow", "black" );
            var integers1 = [ 2, 4, 6, 8 ];
            var integers2 = [ 2, , , 8 ];

            outputArray( "Array colors contains", colors );
            outputArray( "Array integers1 contains", integers1 );
            outputArray( "Array integers2 contains", integers2 );
         }
```

Array integers1 is initialized using an initializer list.

Two values are not supplied for integers2, which will be displayed as undefined.

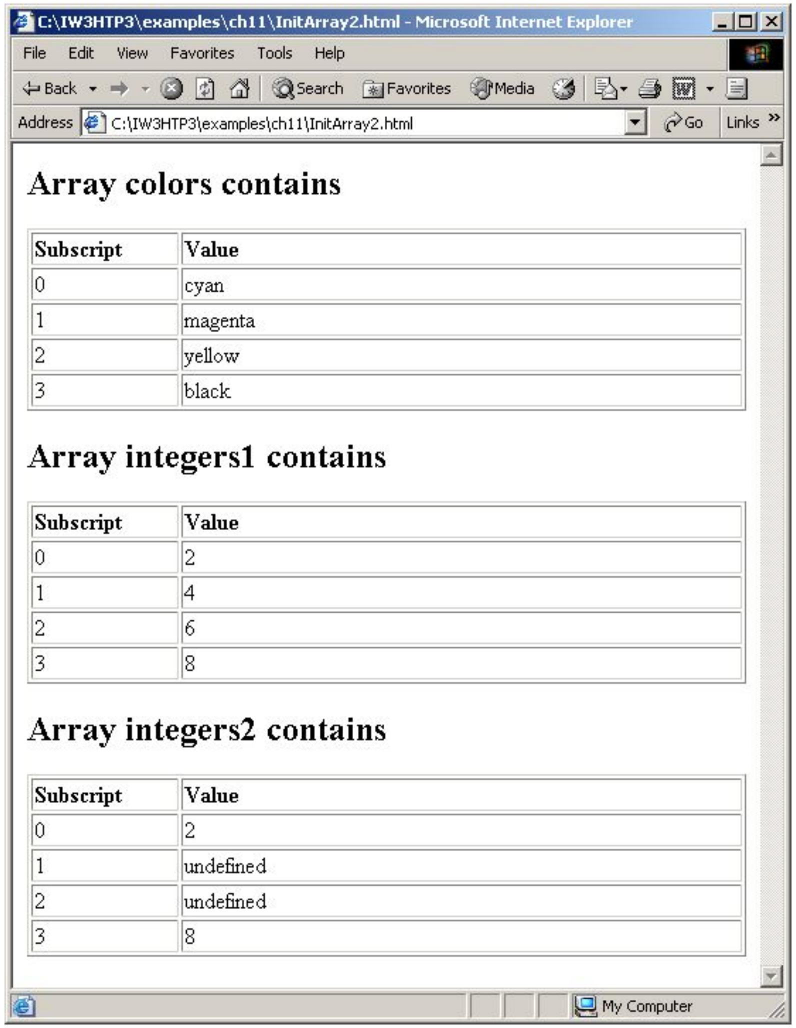
Outline

InitArray2.html
(2 of 2)

```

         // output "header" followed by a two-column table
         // containing subscripts and elements of "theArray"
         function outputArray( header, theArray )
         {
            document.writeln( "<h2>" + header + "</h2>" );
            document.writeln( "<table border = \"1\"" +
               "width = \"100%\">" );
            document.writeln( "<thead><th width = \"100\" " +
               "align = \"left\">Subscript</th>" +
               "<th align = \"left\">Value</th></thead><tbody>" );

            for ( var i = 0; i < theArray.length; i++ )
               document.writeln( "<tr><td>" + i + "</td><td>" +
                  theArray[ i ] + "</td></tr>" );

            document.writeln( "</tbody></table>" );
         }
         // -->
      </script>

   </head><body onload = "start()"></body>
</html>
```

11.4 Examples Using Arrays

Fig. 11.4 Initializing the elements of an array.


C:\IW3HTP3\examples\ch11\InitArray2.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\InitArray2.html
Go Links
My Computer

Array colors contains

Subscript	Value
0	cyan
1	magenta
2	yellow
3	black

Array integers1 contains

Subscript	Value
0	2
1	4
2	6
3	8

Array integers2 contains

Subscript	Value
0	2
1	undefined
2	undefined
3	8

11.4 Examples Using Arrays

- for…in statement
  - Perform an action for each element in an array
  - Iterates over array elements
    - Assigns each element to specified variable one at a time
  - Ignores non-existent elements

Outline

SumArray.html (1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.5: SumArray.html      -->
<!-- Summing Elements of an Array -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Sum the Elements of an Array</title>

      <script type = "text/javascript">
         <!--
         function start()
         {
            var theArray = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ];
            var total1 = 0, total2 = 0;

            for ( var i = 0; i < theArray.length; i++ )
               total1 += theArray[ i ];

            document.writeln( "Total using subscripts: " + total1 );

```

The for loop sums the values contained in the 10-element integer array called theArray.

Outline

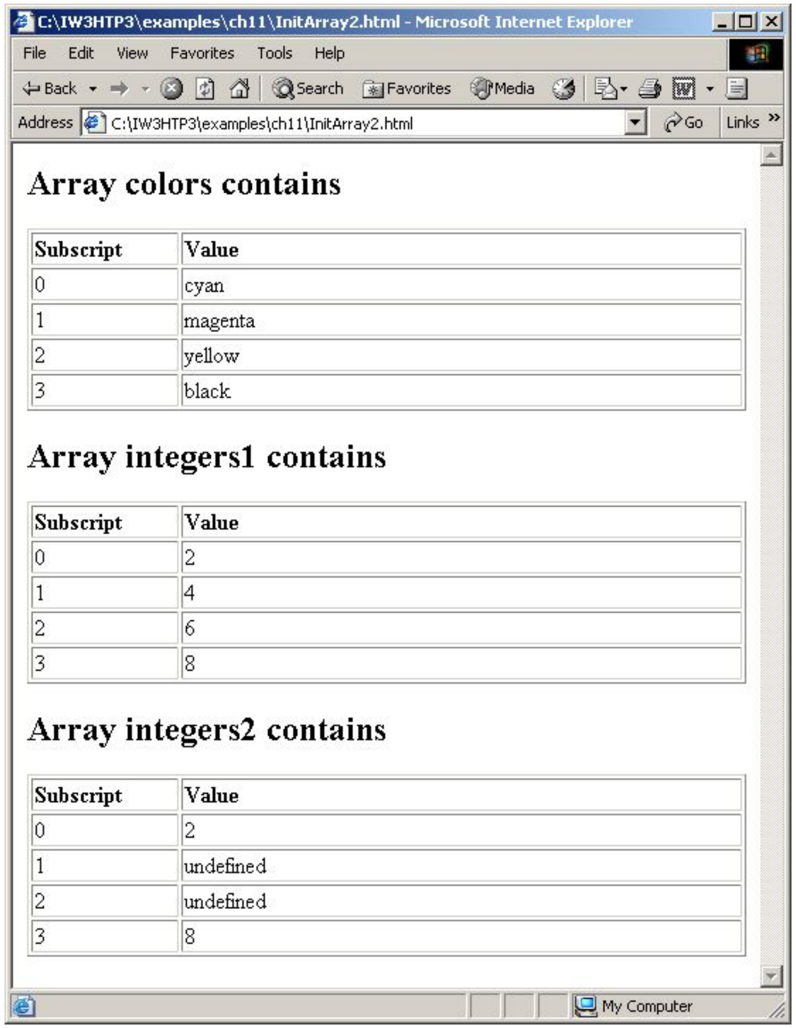
(2 of 2)

```
            for ( var element in theArray )
               total2 += theArray[ element ];

            document.writeln( "<br />Total using for...in
               total2 );
         }
         // -->
      </script>

   </head><body onload = "start()"></body>
</html>
```

Variable element is assigned a subscript in the range of 0 up to, but not including, theArray.length.

11.4 Examples Using Arrays

Fig. 11.5 Calculating the sum of the elements of an array.


C:\IW3HTP3\examples\ch11\InitArray2.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\InitArray2.html
Go Links
My Computer

Array colors contains

Subscript	Value
0	cyan
1	magenta
2	yellow
3	black

Array integers1 contains

Subscript	Value
0	2
1	4
2	6
3	8

Array integers2 contains

Subscript	Value
0	2
1	undefined
2	undefined
3	8

11.4 Examples Using Arrays

- Arrays can provide shorter and cleaner substitute for switch statements
  - Each element represents one case

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.6: RollDie.html          -->
<!-- Roll a Six-Sided Die 6000 Times -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Roll a Six-Sided Die 6000 Times</title>

      <script type = "text/javascr
         <!--
         var face, frequency = [ , 0, 0, 0, 0, 0, 0 ];

         // summarize results
         for ( var roll = 1; roll <= 6000; ++roll ) {
            face = Math.floor( 1 + Math.random() * 6 );
            ++frequency[ face ];
         }

```

Outline

RollDie.html (1 of 2)

Referencing Array frequency replaces the switch statement used in Chapter 10’s example.

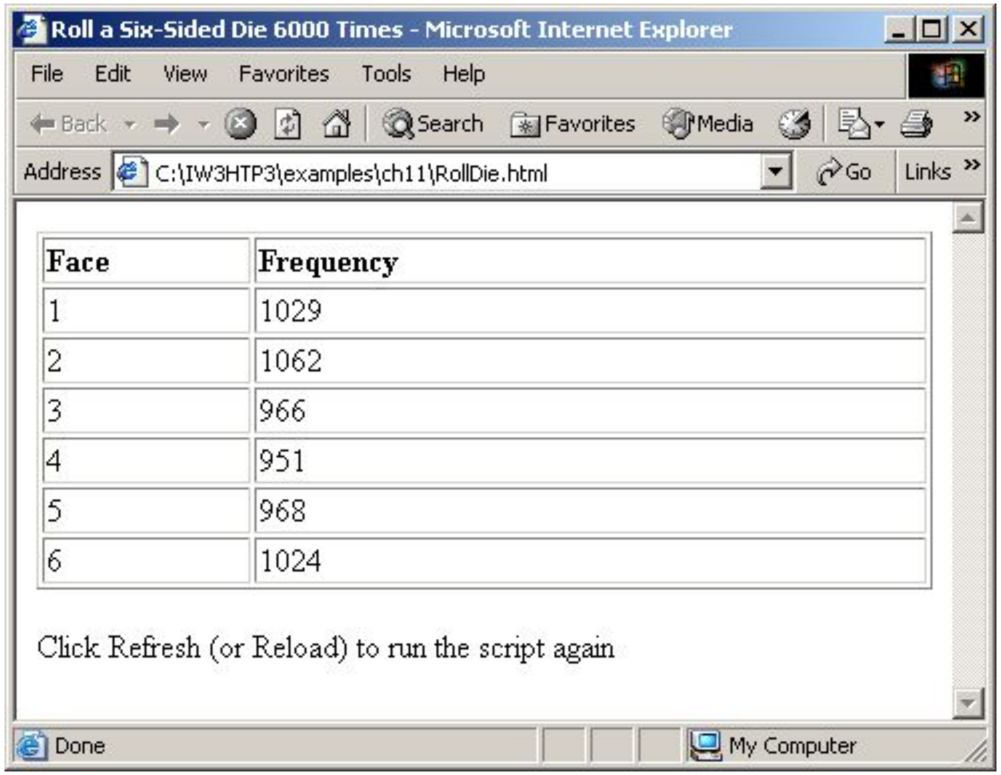
```
         document.writeln( "<table border = \"1\"" +
            "width = \"100%\">" );
         document.writeln( "<thead><th width = \"100\"" +
            " align = \"left\">Face<th align = \"left\">" +
            "Frequency</th></thead></tbody>" );

         for ( face = 1; face < frequency.length; ++face )
            document.writeln( "<tr><td>" + face + "</td><td>" +
               frequency[ face ] + "</td></tr>" );

         document.writeln( "</tbody></table>" );
         // -->
      </script>

   </head>
   <body>
      <p>Click Refresh (or Reload) to run the script again</p>
   </body>
</html>
```

Outline

RollDie.html (2 of 2)

11.4 Examples Using Arrays

Fig. 11.6 Dice-rolling program using arrays instead of a switch.


Roll a Six-Sided Die 6000 Times - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\RollDie.html
Go Links
Done
My Computer

Face	Frequency
1	1029
2	1062
3	966
4	951
5	968
6	1024

Click Refresh (or Reload) to run the script again

11.10 Multidimensional Arrays

- Two-dimensional arrays analogous to tables
  - Rows and columns
    - Specify row first, then column
  - Two subscripts

11.10 Multidimensional Arrays

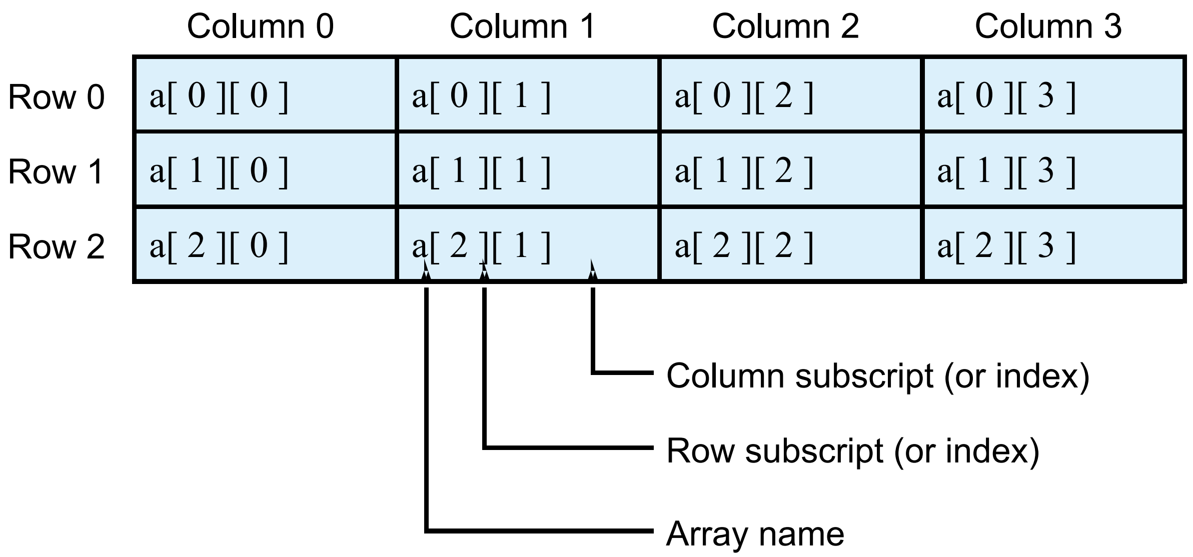


Fig. 11.12 Two-dimensional array with three rows and four columns.

11.10 Multidimensional Arrays

- Declaring and initializing multidimensional arrays
  - Group by row in square brackets
  - Treated as arrays of arrays
  - Creating array b with one row of two elements and a second row of three elements:

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

11.10 Multidimensional Arrays

- Also possible to use new operator
  - Create array b with two rows, first with five columns and second with three:

```
      var b;

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

InitArray3.html
(1 of 2)

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 11.13: InitArray3.html          -->
<!-- Initializing Multidimensional Arrays -->

<html xmlns = "http://www.w3.org/19
   <head>
      <title>Initializing Multidimensional Arrays</title>

      <script type = "text/javascri
         <!--
         function start()
         {
            var array1 = [ [ 1, 2, 3 ],      // first row
                           [ 4, 5, 6 ] ];    // second row
            var array2 = [

                           [ 4, 5, 6 ] ];    // third row

            outputArray( "Values in array1 by row", array1 );
            outputArray( "Values in array2 by row", array2 );
         }
```

Array array1 provides six initializers in two rows.

Array array2 provides six initializers in three rows.

Function outputArray displays each array’s elements in a Web page.

```

      function outputArray( header, theArray )
      {
         document.writeln( "<h2>" + header + "</h2><tt>" );

         for ( var i in theArray ) {

            for ( var j in theArray[ i ] )
               document.write( theArray[ i ][ j ] + " " );

            document.writeln( "<br />" );
         }

         document.writeln( "</tt>" );
      }
      // -->
   </script>

   </head><body onload = "start()"></body>
</html>
```

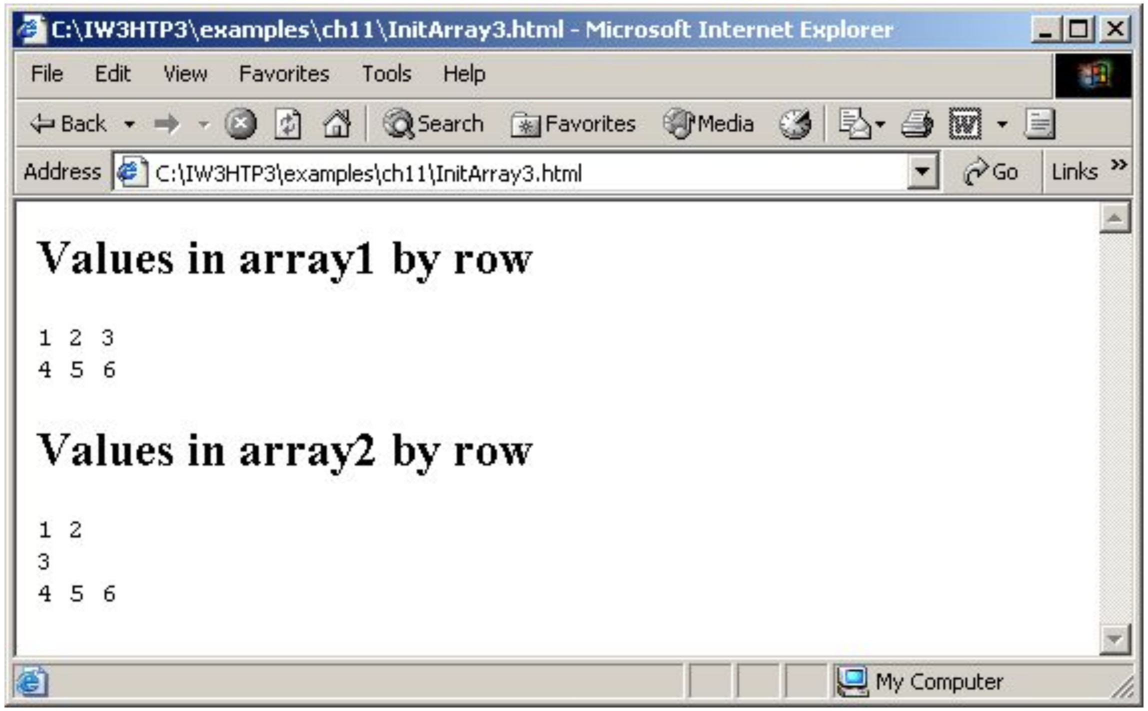
Outline

InitArray3.html (2 of 2)

Referencing the multidimensional array theArray.

11.10 Multidimensional Arrays

Fig. 11.13 Initializing multidimensional arrays.


C:\IW3HTP3\examples\ch11\InitArray3.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch11\InitArray3.html
Go Links
My Computer

Values in array1 by row

```
1 2 3
4 5 6
```

Values in array2 by row

```
1 2
3
4 5 6
```

- Read chapter 12 for different function