ITEC1208

Web Application Development I

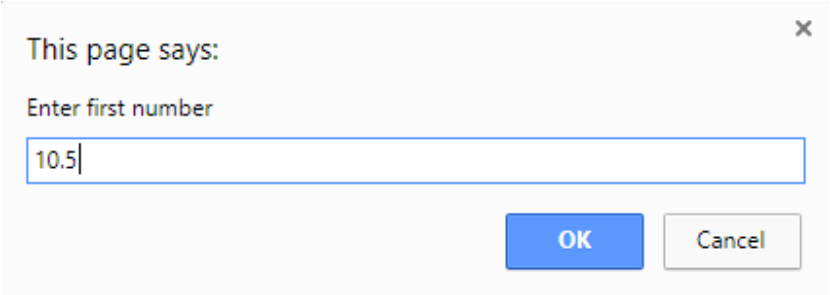
Lab6: Java Script functions

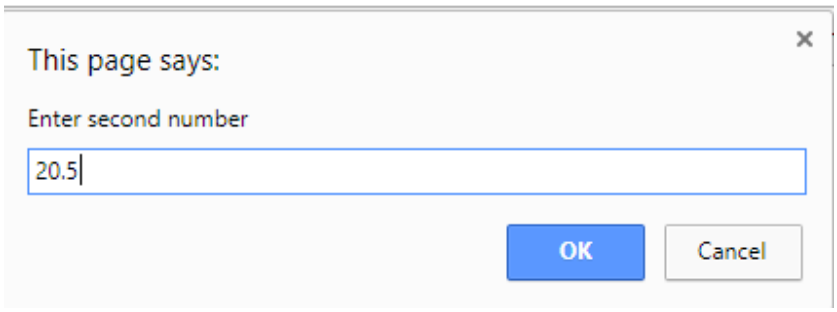
Instructor: Assoc.Prof.Dr.Mudarmeen Munlin รศ.ดร.หมัดอามีน หมันหลิน

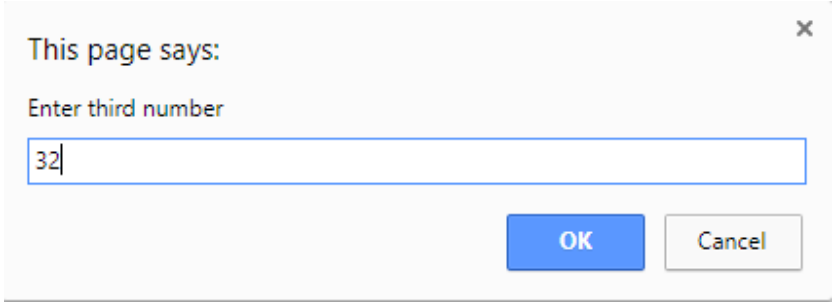
คำสั่ง

1. สร้างโปรเจกใช้ชื่อ รหัสนักศึกษา-Lab6 เช่น 25510090-Lab6
2. เพิ่มไฟล์ html ตามตัวอย่าง และเขียน HTML ให้ได้ผลลัพธ์ตามรูปตัวอย่างในเอกสารแลป
    - ForCounter.html
    - Sum.html
    - SwitchTest.html
    - DoWhileTest.html
    - BreakTest.html
    - ContinueTest.html
    - SquareInt.html
    - Maximum.html
    - Scoping.html
    - FactorialTest.html
3. **สร้างหน้ารวม main.html เพื่อเรียกใช้ทุกๆหน้า กลับไปกลับมาได้**
4. **เพิ่มไฟล์ Assignment6.html แล้วเขียน** HTML รับค่าตัวเลขจำนวนจริง 3 ตัว แล้วเขียนฟังก์ชัน Average() เพื่อหาค่าเฉลี่ย ฟังชัน Maximum() เพื่อหาค่าสูงสุด และฟังก์ชัน Minimum() เพื่อหาค่าต่ำสุด ให้ได้ผลลัพธ์ตามรูปตัวอย่างข้างล่างนี้ (ให้ดูตัวอย่างจาก Maximum.html) แล้วทำคล้ายๆกัน
5. **เพิ่มลิงค์ของ Assignment6.html ไปยังหน้ารวม main.html**

ผลลัพธ์ Assignment6.html







First number: 10.5
Second number: 20.5
Third number: 32
Average is: 21
Maximum is: 32
Mininum is: 10.5

Click Refresh (or Reload) to run the script again

JavaScript: Control Statements and Functions

Outline

- Introduction
- `for` Repetition Statement
- Examples Using the `for` Statement
- `switch` Multiple-Selection Statement
- `do…while` Repetition Statement
- `break` and `continue` Statements
- Program Modules in JavaScript
- Function Definitions
- Scope Rules
- Recursion

Objectives

- In this lesson, you will learn:
  - To be able to use the `for` and `do…while` repetition statements to execute statements in a program repeatedly.
  - To understand multiple selection using the `switch` selection statement.
  - To be able to use the `break` and `continue` program-control statements.
  - To understand how to construct programs modularly from small pieces called functions.
  - To be able to create new functions.
  - To understand the mechanisms used to pass information between functions.
  - To understand how the visibility of identifiers is limited to specific regions of programs.

`for` Repetition Statement

- Counter-controlled repetition
  - Initial value
  - Increment or decrement
  - Final value
- `for` repetition statement
  - Handles all the details of counter-controlled repetition
  - `for` structure header
    - The first line

`for` Repetition Statement

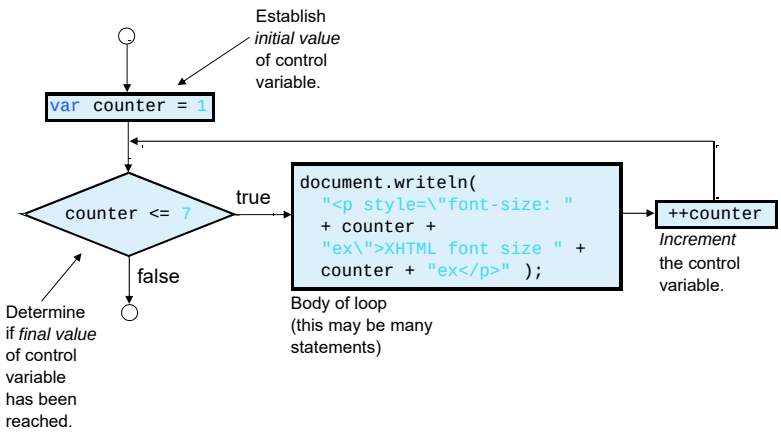


Fig. 9.4 `for` repetition structure flowchart.

for Repetition Statement

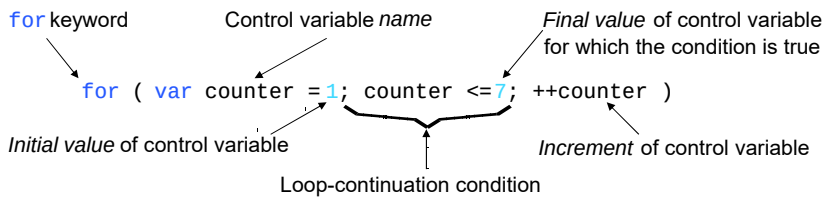
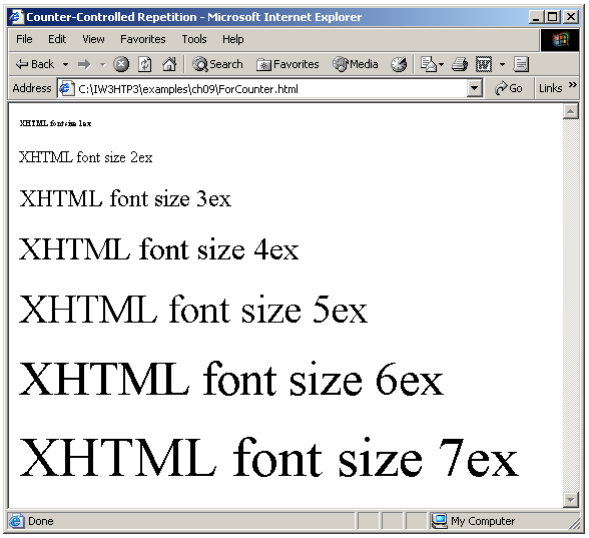


Fig. 9.3 `for` statement header components.

```
<?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. 9.2: ForCounter.html                        -->
<!-- Counter-Controlled Repetition with for statement -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Counter-Controlled Repetition</title>

      <script type = "text/javascript">
         <!--
         // Initialization, repetition condition and
         // incrementing are all included in the for
         // statement header.
         for ( var counter = 1; counter <= 7; ++counter )
            document.writeln( "<p style = \"font-size: " +
               counter + "ex\">XHTML font size " + counter +
               "ex</p>" );
         // -->
      </script>

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

ForCounter.html
(1 of 1)

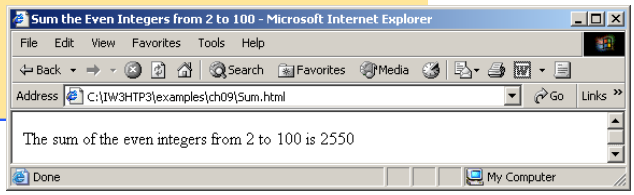


Examples Using the `for` Statement

- Summation with `for`
- Compound interest calculation with `for` loop
  - `Math` object
    - Method `pow`
    - Method `round`

```
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. 9.5: Sum.html                 -->
6  <!-- Using the for repetition statement -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Sum the Even Integers from 2 to 100</title>
11
12       <script type = "text/javascript">
13          <!--
14          var sum = 0;
15
16          for ( var number = 2; number <= 100; number += 2 )
17             sum += number;
18
19          document.writeln( "The sum of the even integers " +
20             "from 2 to 100 is " + sum );
21          // -->
22       </script>
23
24    </head><body></body>
25 </html>
```

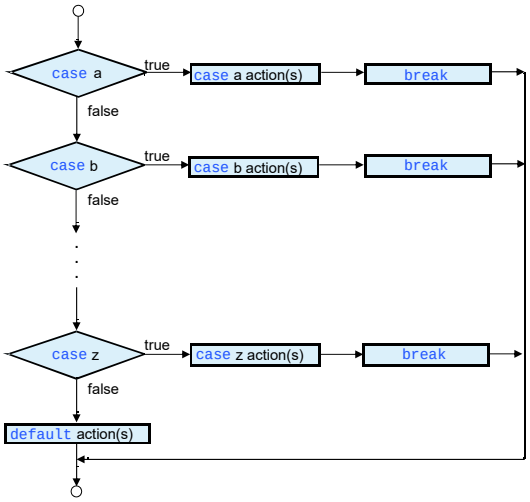
Sum.html
(1 of 1)



`switch` Multiple-Selection Statement

- Controlling expression
- Case labels
- Default case

switch Multiple-Selection Statement



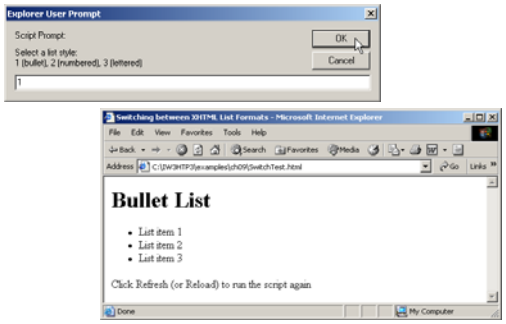

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 9.7: SwitchTest.html  -->
<!-- Using the switch statement -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Switching between XHTML List Formats</title>

      <script type = "text/javascript">
         <!--
         var choice,             // user's choice
             startTag,           // starting list item tag
             endTag,             // ending list item tag
             validInput = true,  // indicates if input is valid
             listType;           // list type as a string

         choice = window.prompt( "Select a list style:\n" +
             "1 (bullet), 2 (numbered), 3 (lettered)", "1" );

```

SwitchTest.html
(1 of 3)

```
23          switch ( choice ) {
24             case "1":
25                startTag = "<ul>";
26                endTag = "</ul>";
27                listType = "<h1>Bullet List</h1>";
28                break;
29             case "2":
30                startTag = "<ol>";
31                endTag = "</ol>";
32                listType = "<h1>Ordered List: Numbered</h1>";
33                break;
34             case "3":
35                startTag = "<ol type = \"A\">";
36                endTag = "</ol>";
37                listType = "<h1>Ordered List: Lettered</h1>";
38                break;
39           default:
40                validInput = false;
41          }
42
43          if ( validInput == true ) {
44             document.writeln( listType + startTag );
45
46             for ( var i = 1; i <= 3; ++i )
47                document.writeln( "<li>List item " + i + "</li>" );
```

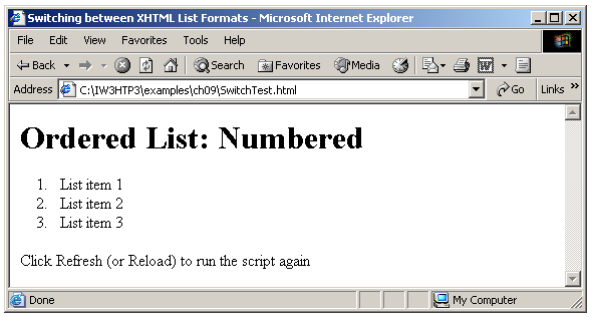
SwitchTest.html (2 of 3)

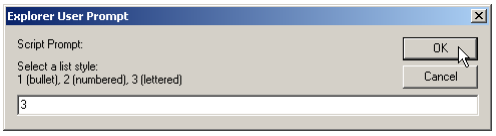
```
48
49             document.writeln( endTag );
50          }
51          else
52             document.writeln( "Invalid choice: " + choice );
53          // -->
54       </script>
55
56    </head>
57    <body>
58       <p>Click Refresh (or Reload) to run the script again</p>
59    </body>
60 </html>
```

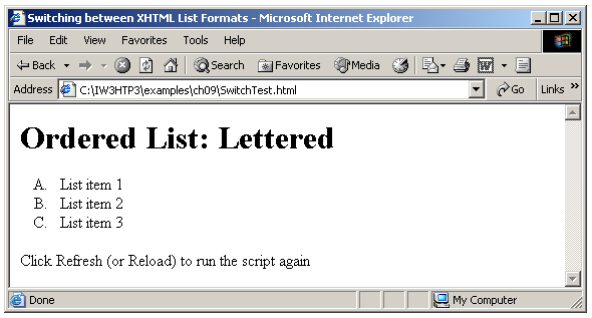


SwitchTest.html (3 of 3)


Explorer User Prompt
Script Prompt:
Select a list style:
1 (bullet), 2 (numbered), 3 (lettered)
OK
Cancel
2


Switching between XHTML List Formats - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch09\SwitchTest.html
Go Links
Ordered List: Numbered
1. List item 1
2. List item 2
3. List item 3
Click Refresh (or Reload) to run the script again
Done
My Computer


Explorer User Prompt
Script Prompt:
Select a list style:
1 (bullet), 2 (numbered), 3 (lettered)
OK
Cancel
3


Switching between XHTML List Formats - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch09\SwitchTest.html
Go Links
Ordered List: Lettered
A. List item 1
B. List item 2
C. List item 3
Click Refresh (or Reload) to run the script again
Done
My Computer

do…while Repetition Statement

- Similar to the `while` statement
- Tests the loop continuation condition after the loop body executes
- Loop body always executes at least once

do…while Repetition Structure

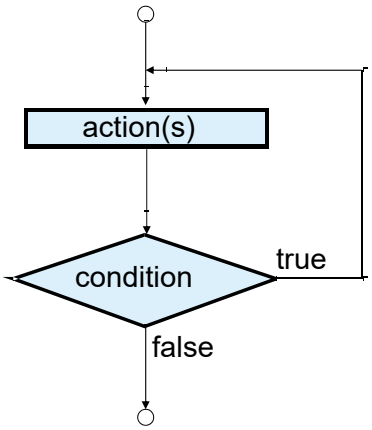
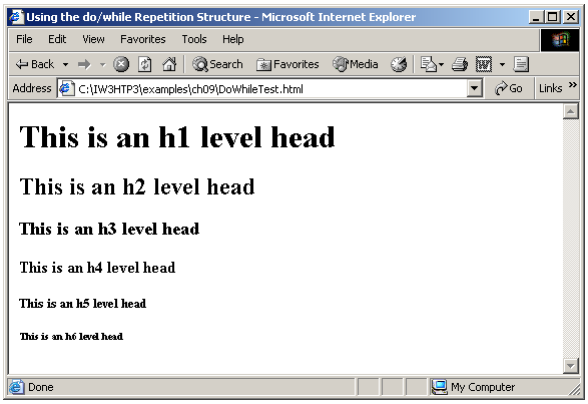


Fig. 9.10 `do…while` repetition statement flowchart.

```
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. 9.9: DoWhileTest.html     -->
6  <!-- Using the do...while statement -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Using the do...while Repetition Statement</title>
11
12       <script type = "text/javascript">
13          <!--
14          var counter = 1;
15
16          do {
17             document.writeln( "<h" + counter + ">This is " +
18                "an h" + counter + " level head" + "</h" +
19                counter + ">" );
20
21             ++counter;
22          } while ( counter <= 6 );
23          // -->
24       </script>
```

DoWhileTest.html (1 of 2)

```
25
26    </head><body></body>
27 </html>
```



DoWhileTest.html (2 of 2)

break and continue Statements

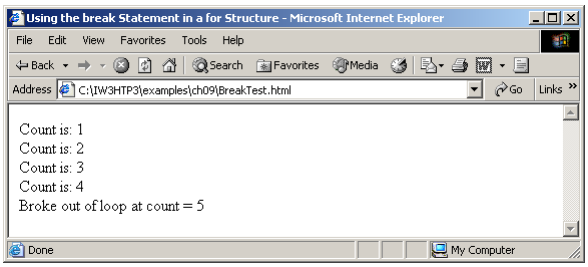
- `break`
  - Immediate exit from the structure
  - Used to escape early from a loop
  - Skip the remainder of a `switch` statement
- `continue`
  - Skips the remaining statements in the body of the structure
  - Proceeds with the next iteration of the loop

```
<?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. 9.11: BreakTest.html   -->
<!-- Using the break statement   -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>
         Using the break Statement in a for Structure
      </title>

      <script type = "text/javascript">
         <!--
         for ( var count = 1; count <= 10; ++count ) {
            if ( count == 5 )
               break;  // break loop only if count == 5

            document.writeln( "Count is: " + count + "<br />" );
         }

```

BreakTest.html
(1 of 2)

```
23          document.writeln(
24             "Broke out of loop at count = " + count );
25          // -->
26       </script>
27
28    </head><body></body>
29 </html>
```

BreakTest.html
(2 of 2)

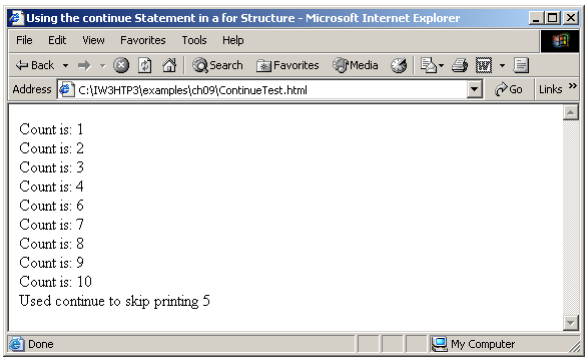



```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 9.12: ContinueTest.html  -->
6  <!-- Using the break statement     -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>
11          Using the continue Statement in a for Structure
12       </title>
13
14       <script type = "text/javascript">
15          <!--
16          for ( var count = 1; count <= 10; ++count ) {
17             if ( count == 5 )
18                continue;  // skip remaining code in loop
19                           // only if count == 5
20
21             document.writeln( "Count is: " + count + "<br />" );
22          }
23
```

ContinueTest.html
(1 of 2)

```
24          document.writeln( "Used continue to skip printing 5" );
25          // -->
26       </script>
27
28    </head><body></body>
29 </html>
```

ContinueTest.html (2 of 2)



Program Modules in JavaScript

- Modules in JavaScript
  - Functions
  - Methods
    - Belong to an object
  - JavaScript includes many useful pre-defined methods
    - Combine with programmer-defined methods to make a program

Program Modules in JavaScript

- Functions
  - Started by function call
  - Receive necessary information via arguments (parameters)
  - Boss-Worker relationship
    - Calling function
    - Called function
    - Return value when finished
    - Can have many tiers

Program Modules in JavaScript

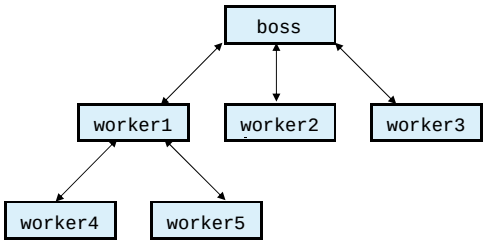


Fig. 10.1 Hierarchical boss-function/worker-function relationship.

Program Modules in JavaScript

- Function calls
  - Name
  - Left parenthesis
  - Arguments separated by commas
    - Constants, variables or expressions
  - Right parenthesis
  - Examples:

```
total += parseFloat( inputValue );
total += parseFloat( s1 + s2 );
```


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)

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

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

```
<?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`.

SquareInt.html
(1 of 2)


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

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

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

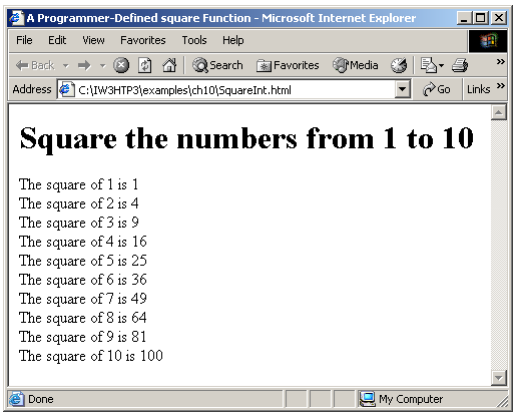
Variable `y` gets the value of variable `x`.

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

SquareInt.html
(2 of 2)

Function Definitions

Fig. 10.2 Using programmer-defined function `square`.



10.4 Function Definitions

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

Maximum.html
(1 of 2)

```
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. 10.3: maximum.html -->
6  <!-- Maximum function        -->
7
8  <html xmlns = "http://www.w3.org/1999/xhtml">
9     <head>
10       <title>Finding the Maximum of Three Values</title>
11
12       <script type = "text/javascript">
13          <!--
14          var input1 =
15             window.prompt( "Enter first number", "0" );
16          var input2 =
17             window.prompt( "Enter second number", "0" );
18          var input3 =
19             window.prompt( "Enter third number", "0" );
20
21          var value1 = parseFloat( input1 );
22          var value2 = parseFloat( input2 );
23          var value3 = parseFloat( input3 );
```

Prompt for the user to input three integers.


```
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>
```

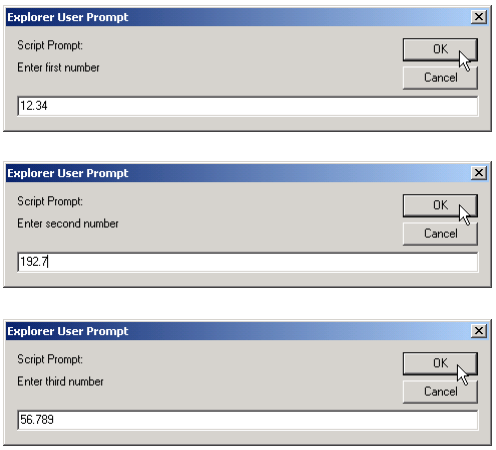
Call function `maximum` and pass it the value of variables `value1`, `value2` and `value3`.

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.

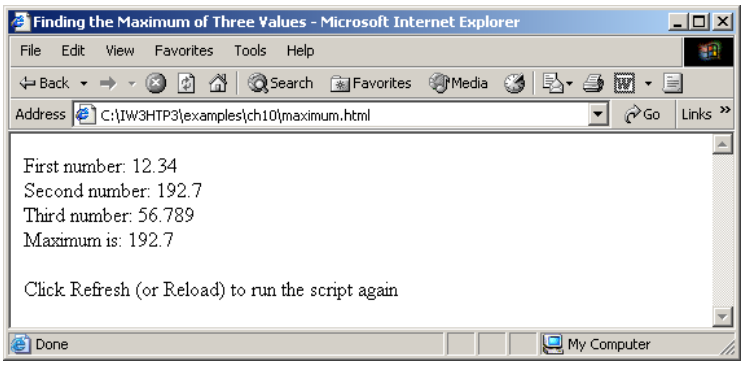
Function Definitions

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



Function Definitions

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



Scope Rules

- Scope
  - Portion of program where identifier can be referenced
  - Inside function is local or function scope
    - Identifiers exist only between opening and closing braces
    - Local variables hide global variables

Scope Rules

- Scope demonstration
  - Global variable `x` initialized to `1`
  - `start` has local variable `x` initialized to `5`
  - `functionA` has local variable `x` initialized to `25`
  - `functionB` has no local variable `x`
  - Observe output of each function

Scoping.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. 10.8: scoping.html    -->
<!-- Local and Global Variables -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>A Scoping Example</title>

      <script type = "text/javascript">
         <!--
         var x = 1;       // global variable

         function start()
         {
            var x = 5;  // variable local to function start

            document.writeln( "local x in start is " + x );

            functionA(); // functionA has local x
            functionB(); // functionB uses global variable x
            functionA(); // functionA reinitializes local x
            functionB(); // global variable x retains its value
```

To begin the program, variable x is initialized to 1.

Function `start` changes the value of x to 5.

Scoping.html (2 of 3)

```

            document.writeln(
               "<p>local x in start is " + x + "</p>" );
         }

         function functionA()
         {
            var x = 25;  // initialized each time
                         // functionA is called

            document.writeln( "<p>loc
                              x + " after entering functionA" );
            ++x;
            document.writeln( "<br />local x in functionA is " +
               x + " b                                          );
         }

```

Function `functionA` changes the value of x to 25.

The value of x is incremented.

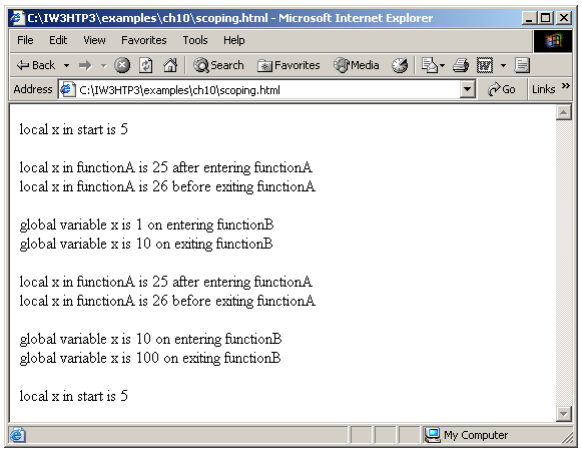
```
43         function functionB()
44         {
45            document.writeln( "<p>global variable x is " + x +
46               " on entering functionB" );
47            x *= 10;
48            document.writeln( "<br />global variable x is " +
49               x + " on exiting functionB"  + "</p>" );
50         }
51         // -->
52      </script>
53
54   </head>
55   <body onload = "start()"></body>
56 </html>
```

Function `functionB` multiplies the value of `x` by `10`.

Scoping.html (3 of 3)

Scope Rules

Fig. 10.8 Scoping example.



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

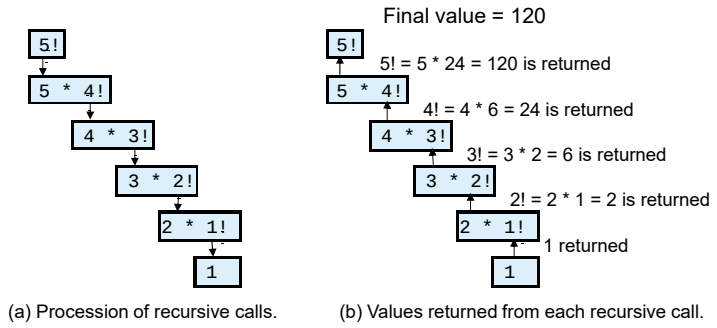
Recursion

- 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

Recursion



(a) Procession of recursive calls. (b) Values returned from each recursive call.

Fig. 10.10 Recursive evaluation of 5!.

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`.

```
23          // Recursive definition of function factorial
24          function factorial( number )
25          {
26             if ( number <= 1 )  // base c
27                return 1;
28             else
29                return number * factorial( number - 1 );
30          }
31       </script>
32    </head><body></body>
33 </html>
```

Variable number gets the value of variable i.

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

ialTest.html

(2 of 2)

Recursion

Fig. 10.11 Factorial calculation with a recursive function.