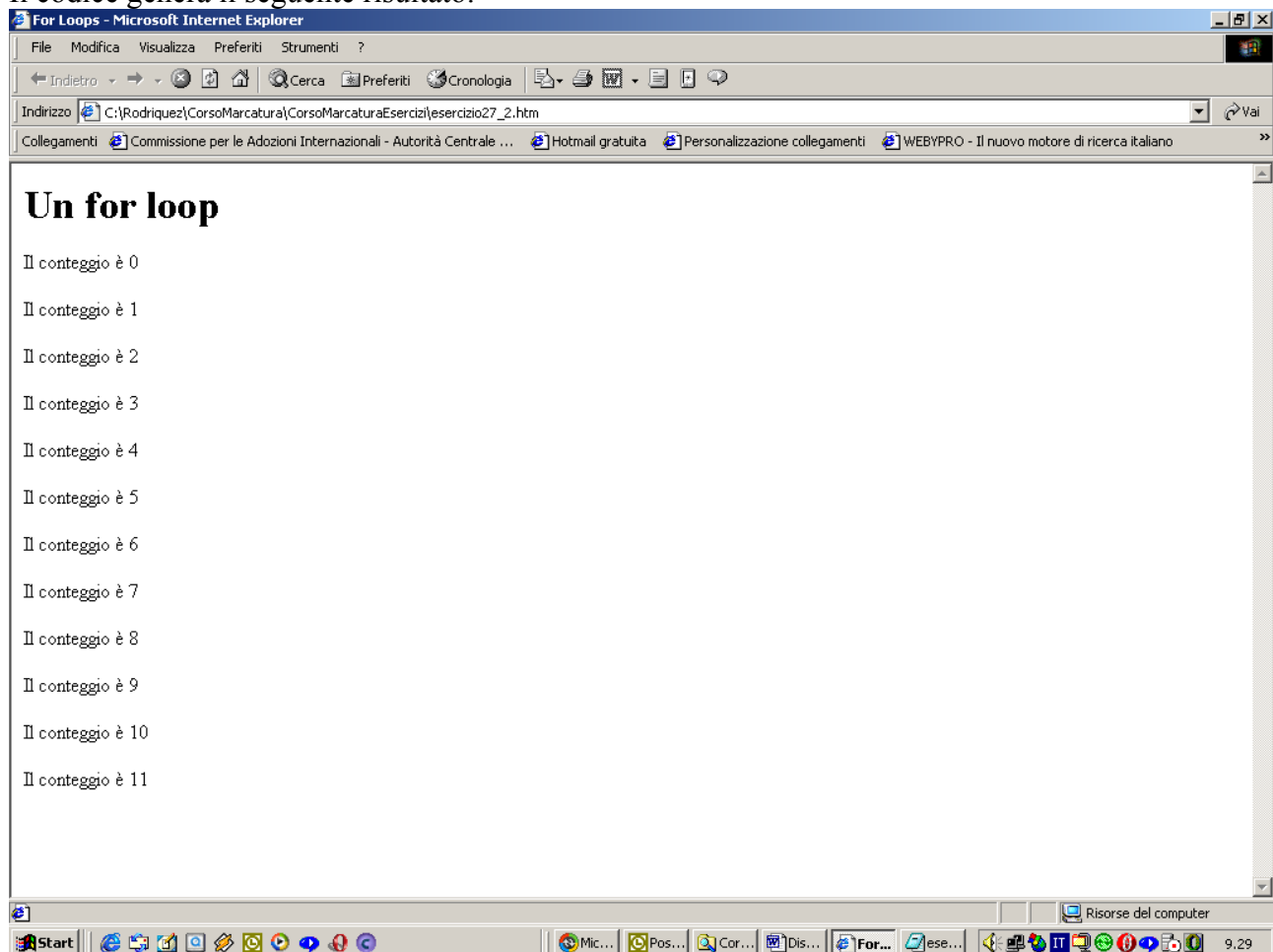


Esempio Loop For

```
<html>
  <head>
    <title>For Loops</title>
  </head>
  <body>
    <script language="Javascript">
      <!--
        document.write("<h1>Un for loop</h1>");
        for (var count=0;count<12;count++) {
          document.write("<p>Il conteggio è " + count);
          document.write("</p>");
        }
        document.close();
      //-->
    </script>
  </body>
</html>
```

Il codice genera il seguente risultato:



Il ciclo *for* è usato quando si conosce in anticipo quante volte deve essere eseguito il processo.

I tre statement contenuti nelle parentesi tonde e separate dal carattere “;” eseguono rispettivamente:

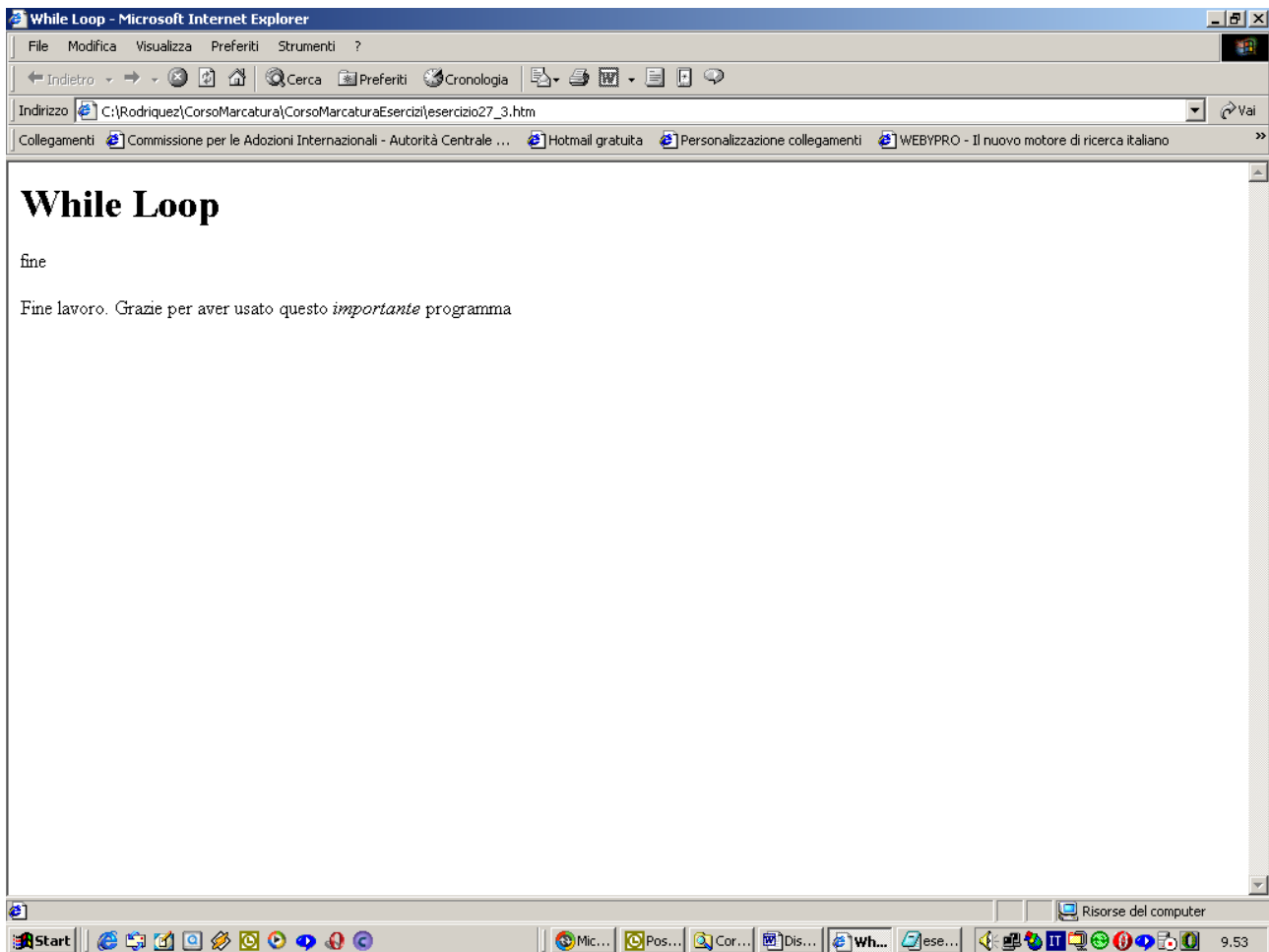
- inizializzazione della variabile di conteggio iterazioni
- indicazione della condizione in base alla quale il ciclo deve essere ripetuto
- incremento della variabile di conteggio iterazioni (nell’esempio l’incremento è di una unità)

Esempio Loop While

```
<html>
  <head>
    <title>While Loop</title>
  </head>
  <body>
    <script language="Javascript">
      <!--
        document.write("<h1>While Loop</h1>");
        var done=false;
        var msg;

        while (done==false) {
          msg = prompt ("Immettere una stringa","");
          document.write("<p>" + msg + "</p>");
          if (msg.toLowerCase()=="fine") {
            document.write("</p>Fine lavoro. Grazie per aver usato
questo") ;
            document.write ("<i> importante </i>programma</p>");
            done=true;
          } //if
        } // while
        document.close();
      //-->
    </script>
  </body>
</html>
```

Il codice produce il seguente risultato:



Il ciclo *while* è usato quando non si conosce a priori il numero delle iterazioni.

Tipicamente il processo continua sino a quando non si verifica un evento esterno, intercettato dal programma, che conduce al termine del ciclo. Esempi sono dati dalla presenza in una variabile di un particolare valore, dal raggiungimento dell'ultimo record di una tabella di un database, ecc.

Nel codice esemplificativo l'evento esterno è dato dalla digitazione da parte dell'utente della stringa "fine".

Si noti la trasformazione in caratteri minuscoli della stringa immessa, allo scopo di intercettare tutte le possibili varianti di caratteri maiuscoli/minuscoli con le quali la stringa potrebbe essere digitata.

Esempio Passaggio di parametri in una funzione

```
<html>
  <head>
    <title>Informazioni personali</title>
    <script language="javascript">
      <!--

      function pers_info(name, age, shoesize){
        document.write("<h1>Informazioni personali</h1>");
        document.write("<p><strong>Il tuo nome è: </strong>" + name);
        document.write("<p><strong>Hai: </strong> " + age + " anni");
        document.writeln("<p><strong>Porti scarpe numero: </strong>" + shoesize);
        document.close();
      }
      <!-->
    </script>
  </head>
  <body onLoad="pers_info('Mario', 50, 42)">

  </body>
</html>
```

Il codice produce il seguente risultato :



Al valore passato alla funzione come parametro viene attribuito un nome e tale nome sarà usato dalla funzione.

I parametri sono applicati nello stesso ordine con il quale sono passati alla funzione, per cui un ordine errato genererà errori.

Se non si desidera passare un determinato parametro dovrà essere passata al suo posto una stringa vuota ("").

Esempio Richiesta di conferma

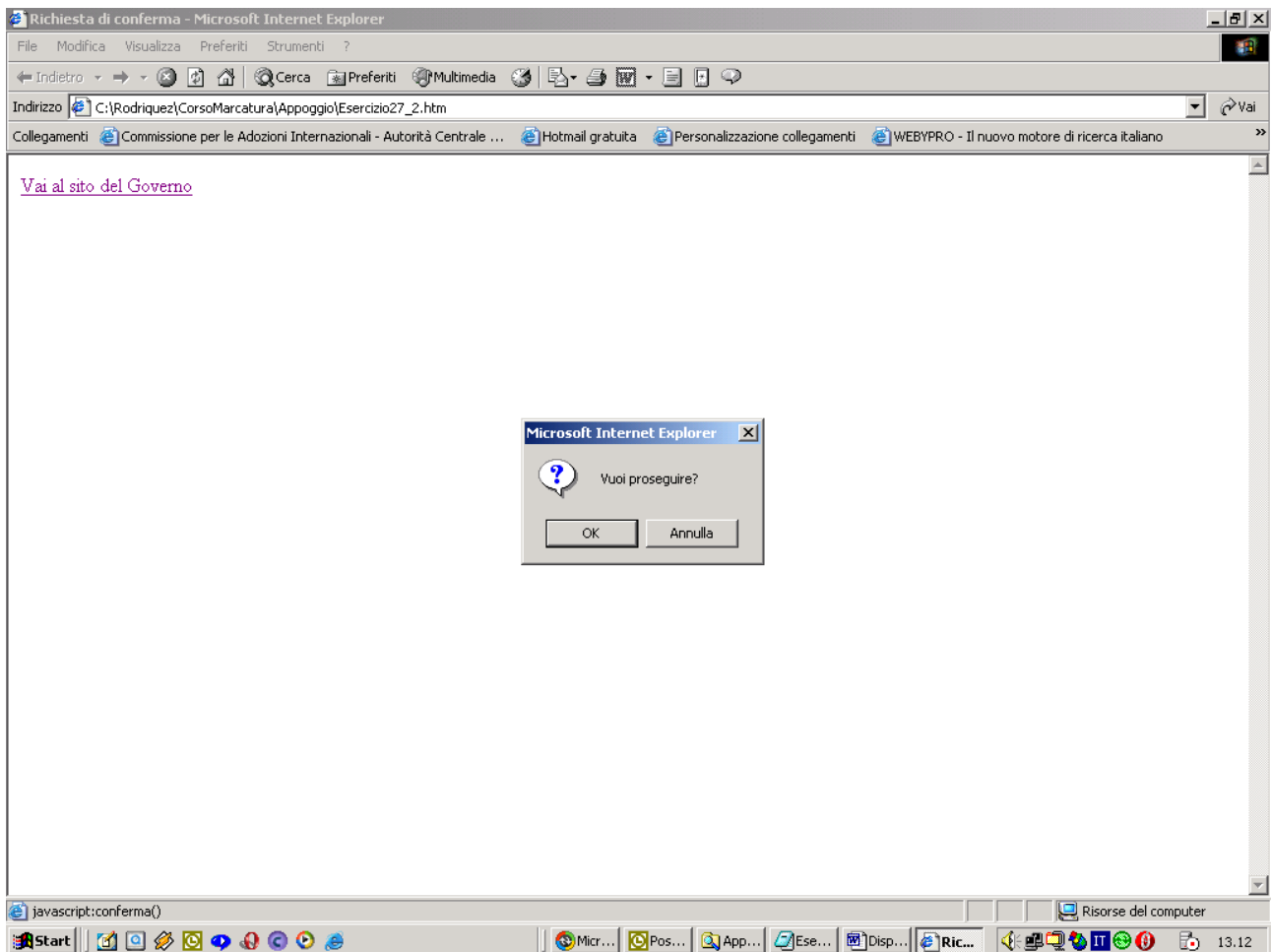
```
<<html>

<head>
<title>Richiesta di conferma</title>

<script language= "JavaScript" >
<!--
function conferma() {
    var continua = "Vuoi proseguire?";

    if (confirm(continua)) {
        location.replace("http://www.governo.it");
    }
}
//-->
</script>
</head>
<body>
<a href="javascript:conferma()">Vai al sito del Governo Italiano</a>
</body>
</html>
```

Il codice produce il seguente risultato



Nell'esempio si fa uso del metodo *confirm()*, metodo associato all'oggetto *window*, che consente all'utente di confermare o meno una determinata azione. Questo metodo è simile al metodo *alert()*, con la differenza che quest'ultimo visualizza solo l'opzione di conferma, senza dare la possibilità di annullare l'azione. Quando l'utente clicca sul bottone OK il valore restituito da JavaScript è *true*, quando clicca sul bottone Annulla il valore restituito è *false*.

Il messaggio che deve apparire nella finestra di dialogo può essere assegnato ad una variabile (nell'esempio la variabile è *continua*).

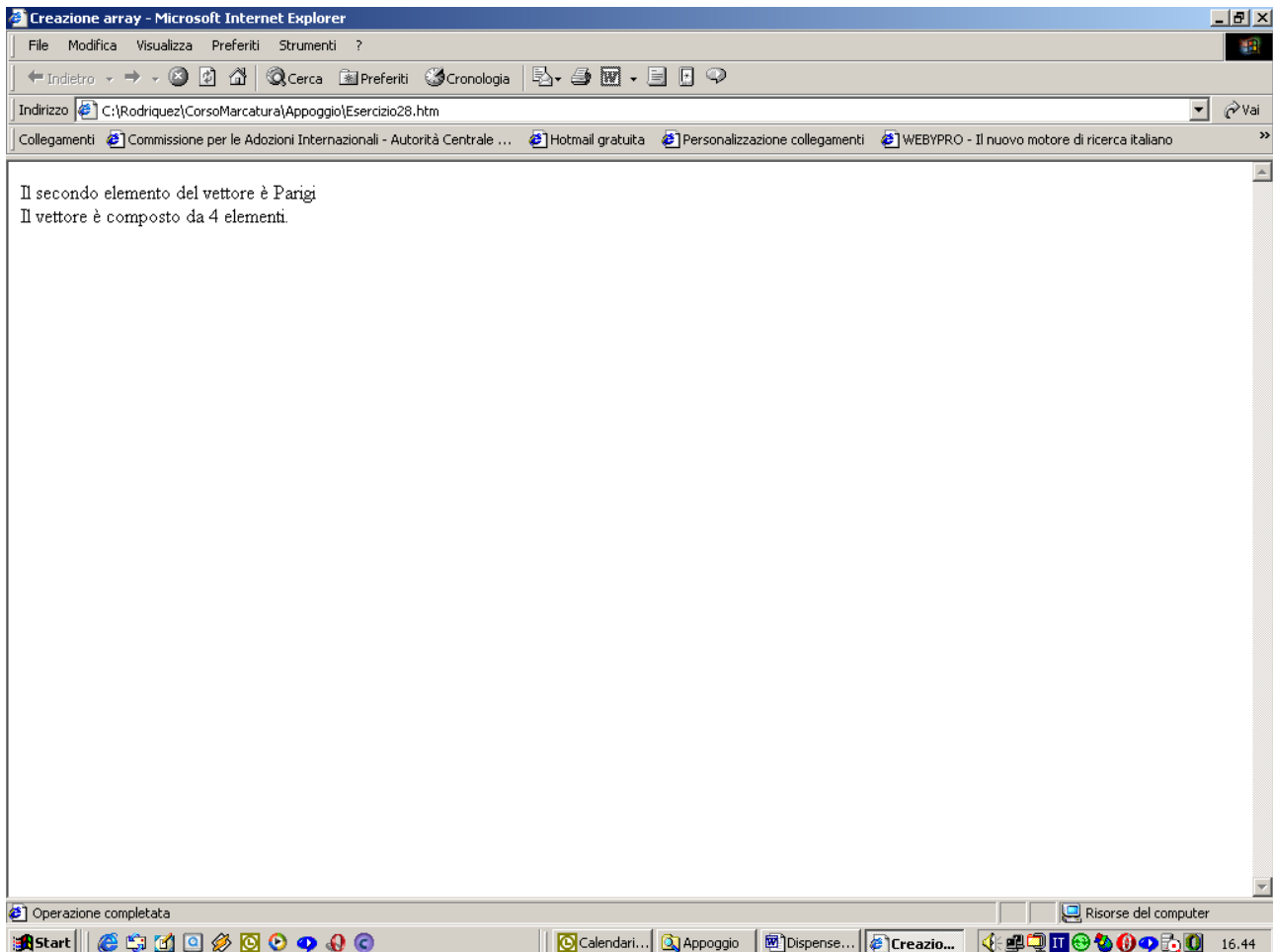
Va ricordato che le finestre di dialogo associate ai metodi *alert()*, *prompt()* e *confirm()* sono di tipo *modale*, ovvero bloccano l'esecuzione dello script sino a quando l'utente non fornisce la risposta e pertanto non possono essere eseguiti contemporaneamente altri script.

Esempio creazione array

```
<html>
  <head>
    <title>Creazione array</title>
  </head>
  <body>
    <script language="Javascript">
      <!--
        var metropolis = new Array("Roma","Parigi","Londra","Madrid");

        document.write("Il secondo elemento del vettore è " + metropolis[1]
+ "<br>");
        document.write("Il vettore è composto da " + metropolis.length + "
elementi.");
      <!-->
    </script>
  </body>
</html>
```

Il codice produce il seguente risultato:



Si tenga presente che:

- in Javascript non è necessario specificare la lunghezza dell'array nel momento della sua creazione
- in Javascript il primo elemento ha indice 0, il secondo ha indice 1 e così via

nel codice è usata la proprietà *length*, che restituisce il numero di elementi dell'array

Esempio Loop in un Array

```
<html>
  <head>
    <title>Loop in un array</title>
  </head>
  <body>
    <script language="Javascript">
      <!--
        document.write("<h1>Loop in un array</h1>");
        document.write("<p>");

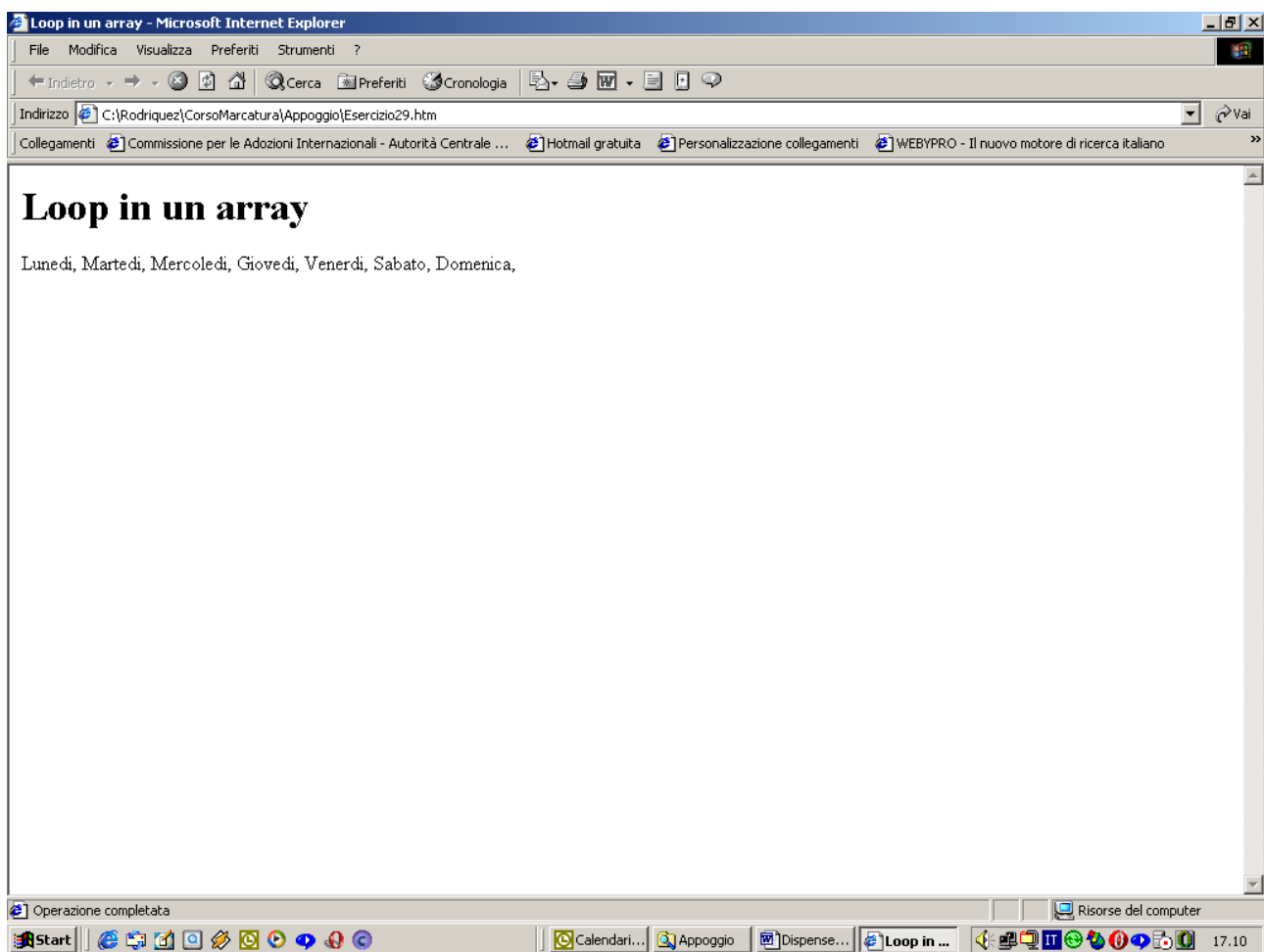
        var data=["Lunedì", "Martedì","Mercoledì","Giovedì",
"Venerdì","Sabato", "Domenica"];
        var lunghezza=data.length;

        for(var conteggio=0; conteggio < lunghezza; conteggio++)
        {
            document.write(data[conteggio] + ", ");
        }

        document.write("<p>");
        document.close();

        //-->
      </script>
    </body>
</html>
```

Il codice produce il seguente risultato:



Esempio concatenazione di Array

```
<html>
  <head>
    <title>Concatenazione di array</title>
  </head>
  <body>
    <script language="Javascript">
      <!--
      document.write("<h1>Concatenazione di 3 array</h1>");

      var prima=["Lunedì", "Martedì", 123, 3.14, "Mercoledì"];
      var seconda=["uno", "due", "tre", 55.55];
      var terza= new Array ("un", "oggetto", "Array");

      var risultato= prima.concat(seconda, terza);

      // visualizza l'array risultante

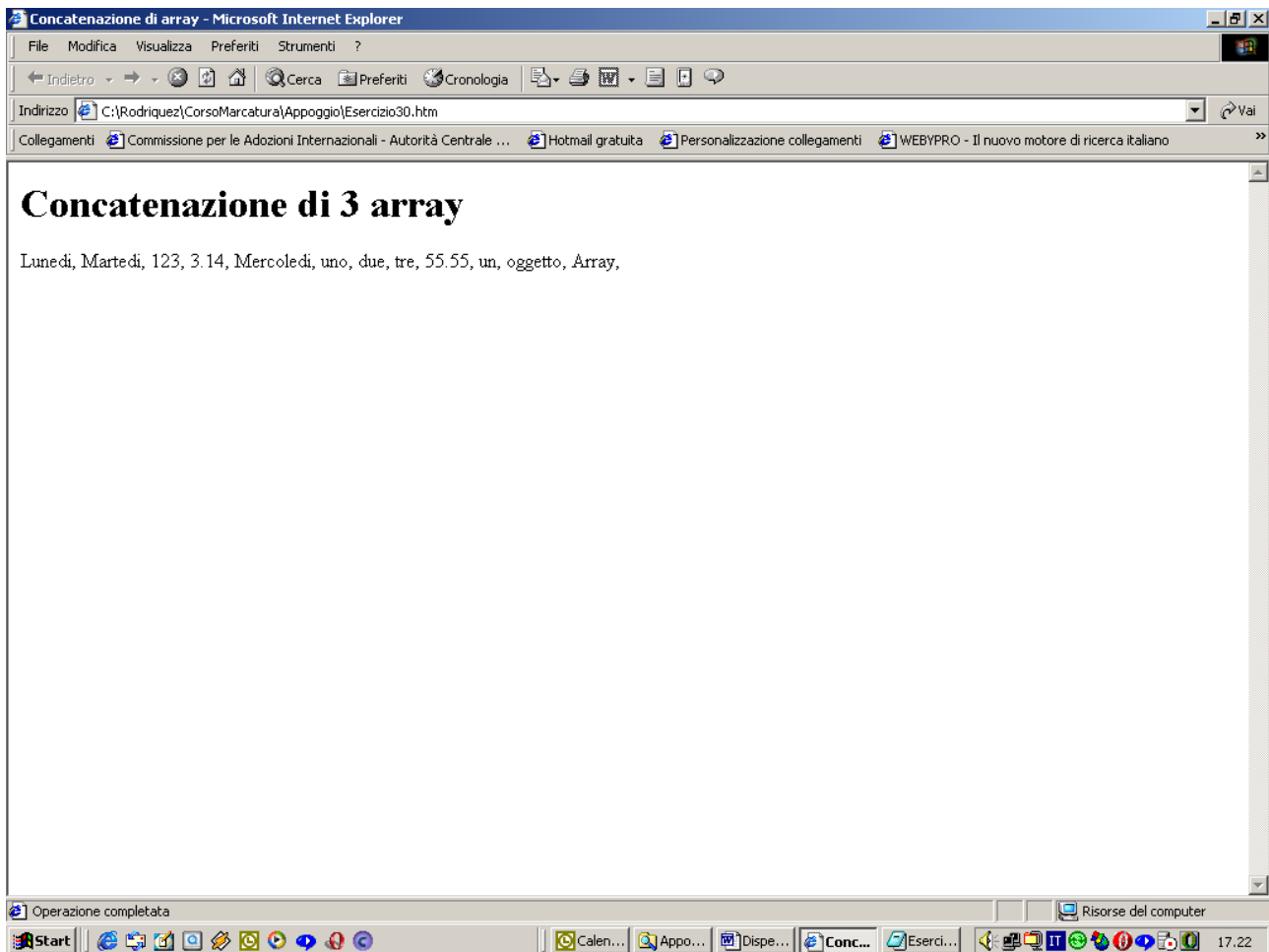
      document.write("<p>");
      var lunghezza = risultato.length;

      for(var conteggio=0;conteggio<lunghezza;conteggio++)
      {

        document.write(risultato[conteggio] + ", ");
      }

      document.write("<p>");
      document.close();
      //-->
    </script>
  </body>
</html>
```

Il codice produce il seguente risultato:



Nel codice esemplificativo è stata usata la funzione object-based *concat*, che concatena un elenco di array, uno dietro l'altro, restituendo un nuovo array. Si noti che gli array originali non vengono modificati dalla funzione.

Tutte le funzioni array sono usate allo stesso modo, ovvero indicando il nome dell'array sul quale eseguire l'elaborazione, un punto seguito dal nome della funzione e, tra parentesi tonde, i nomi dei parametri separati da virgole.

Esempio Funzioni con gli Array

```
<html>
  <head>
    <title>Operazioni con gli Array</title>
  </head>
  <body>
    <script language="javascript">
      <!--
        document.writeln("<h1>Funzioni con gli Array</h1>");

        var first = ["Lunedì", "Martedì", 123, 3.14, "Mercoledì"];
        document.write("<p>");

        document.write(first.join(", "));
        document.write("<br>");

        first.pop();
        document.write(first.join(", "));
        document.write("<br>");

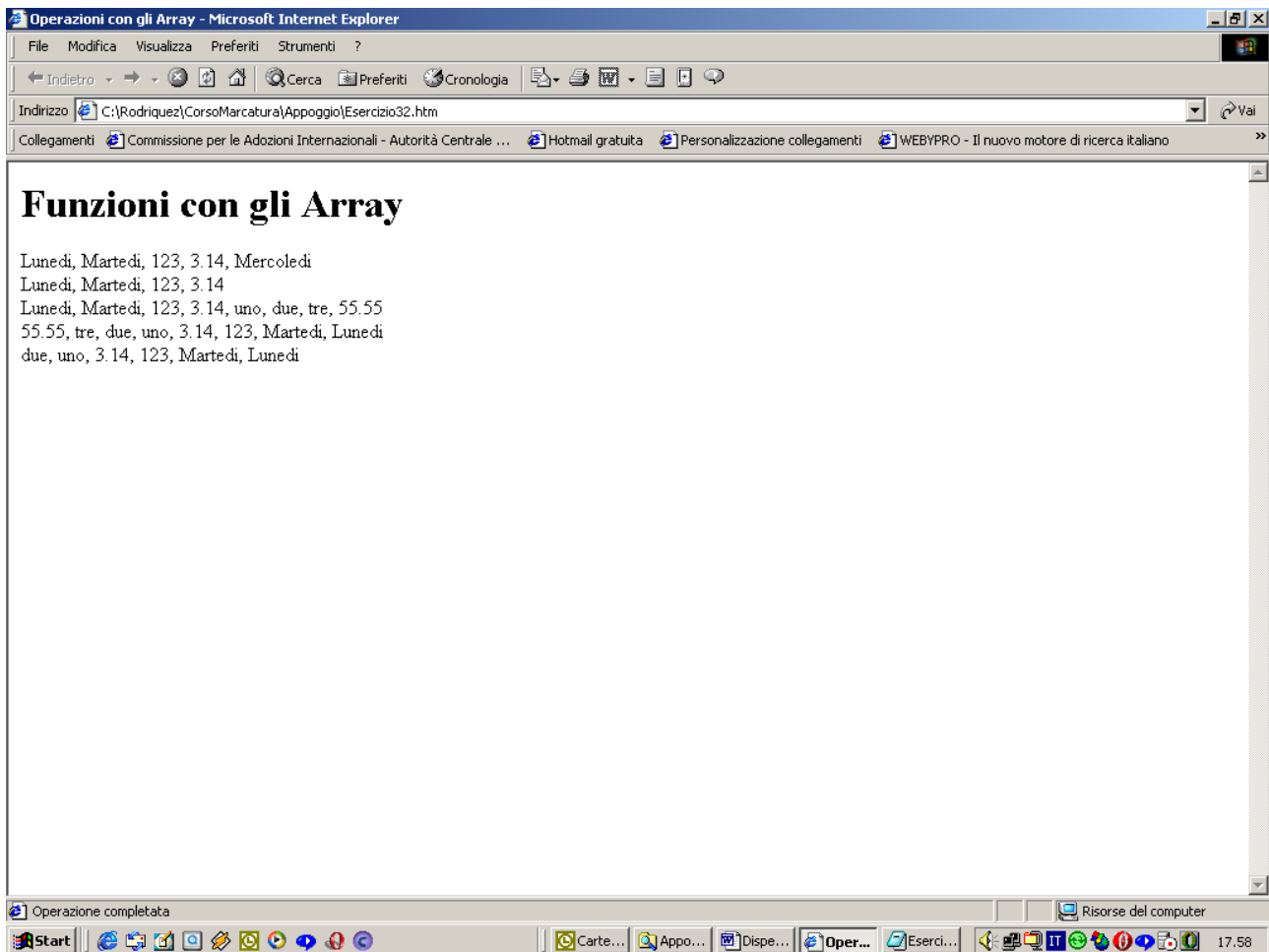
        first.push("uno", "due", "tre", 55.55);
        document.write(first.join(", "));
        document.write("<br>");

        first.reverse();
        document.write(first.join(", "));
        document.write("<br>");

        first.shift();
        first.shift();
        document.write(first.join(", "));

        document.writeln("</p>");
        document.close();
      //-->
    </script>
  </body>
</html>
```

Il codice produce il seguente risultato:

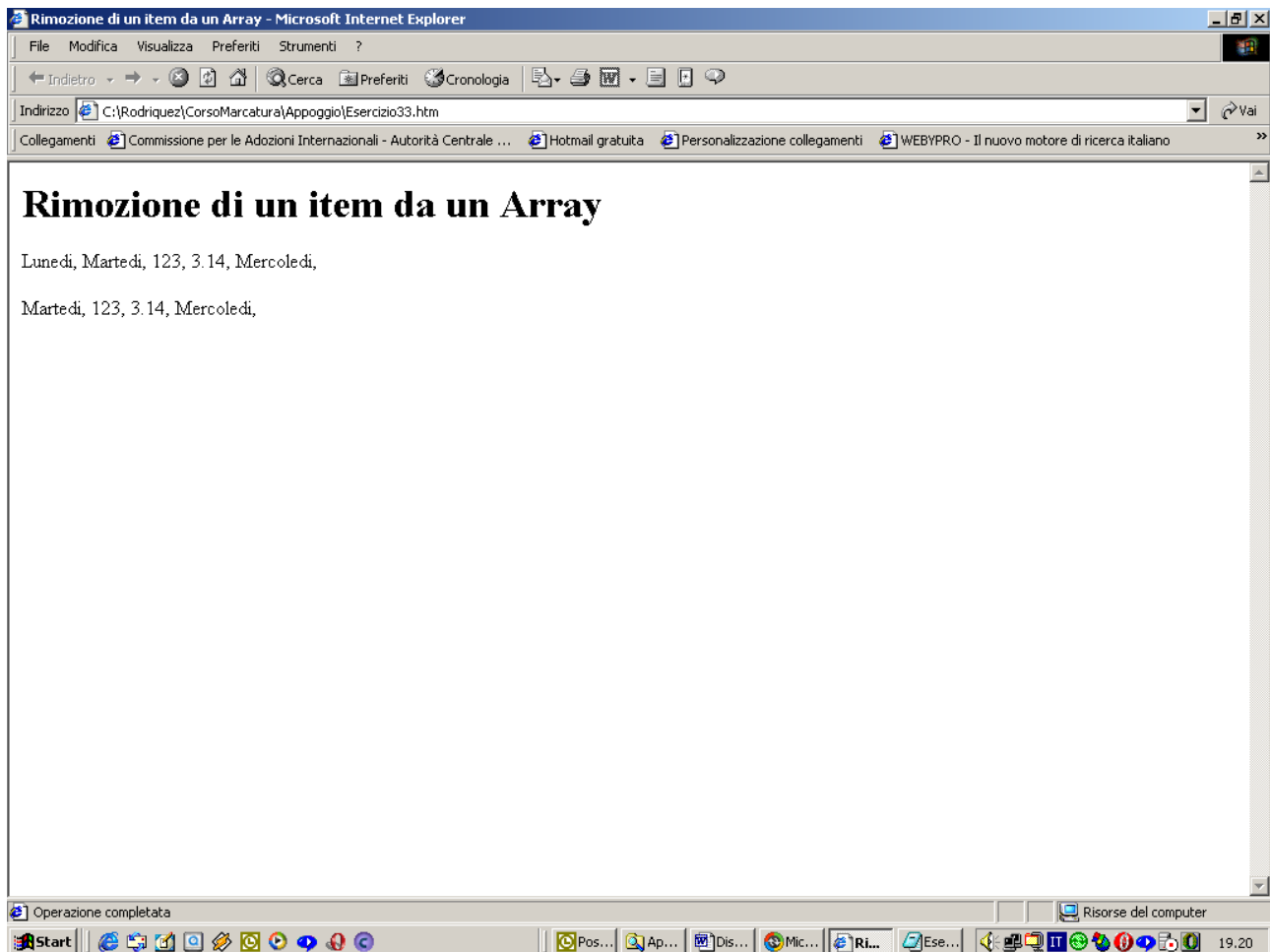


- la funzione *join* crea una stringa composta da tutti gli elementi dell'array
- la funzione *pop* elimina l'ultimo elemento dell'array riducendo di un'unità la lunghezza dell'array
- la funzione *push* inserisce un elenco di voci in coda all'array
- la funzione *reverse* inverte l'ordine degli elementi nell'array
- la funzione *shift* elimina il primo elemento dell'array riducendo di un'unità la lunghezza dell'array

Esempio Rimozione di un item da un Array

```
<html>
  <head>
    <title>Rimozione di un item da un Array</title>
  </head>
  <body>
    <script language="javascript">
      <!--
        document.writeln("<h1>Rimozione di un item da un Array</h1>");
        var data = ["Lunedì", "Martedì", 123, 3.14, "Mercoledì"];
        // Visualizza l'array di origine
        document.write("<p>");
        var len = data.length;
        for(var count = 0; count < len; count++) {
          document.write(data[count] + ", ");
        }
        document.writeln("</p>");
        // Prompt: quale item da rimuovere?
        var rem = prompt("Quale item devo rimuovere?", "");
        var tmp = new Array(data.length - 1);
        var count2 = 0;
        // Loop di ricerca e rimozione item
        for(var count = 0; count < len; count++) {
          if(data[count] == rem){
            // nessuna azione
            //document.write("Item trovato. data[count] = " + count);
            //document.write(": " + data[count] + "<br>");
          } else {
            tmp[count2] = data[count];
            //document.write("Copia. data[count] = " + count);
            //document.write(": " + data[count] + "<br>");
            //document.write("&nbsp;&nbsp;&nbsp;tmp[count2] = " + count2);
            //document.write(": " + tmp[count2] + "<br>");
            count2++;
          }
        }
        data = tmp;
        // Visualizza il nuovo Array
        document.write("<p>");
        var len = data.length;
        for(var count = 0; count < len; count++) {
          document.write(data[count] + ", ");
        }
        document.writeln("</p>");
        //document.writeln("<p>");
        //data.splice(2, 0, "elemento uno", "elemento due");
        //document.writeln(data.join(", ") + "<br>");
        //data.splice(3, 3, "elemento tre", "elemento quattro");
        //document.writeln(data.join(", "));
        //document.writeln("</p>");
        document.close();
      <!-->
    </script>
  </body>
</html>
```

Il codice produce il seguente risultato, indicando come item da rimuovere la stringa Lunedì



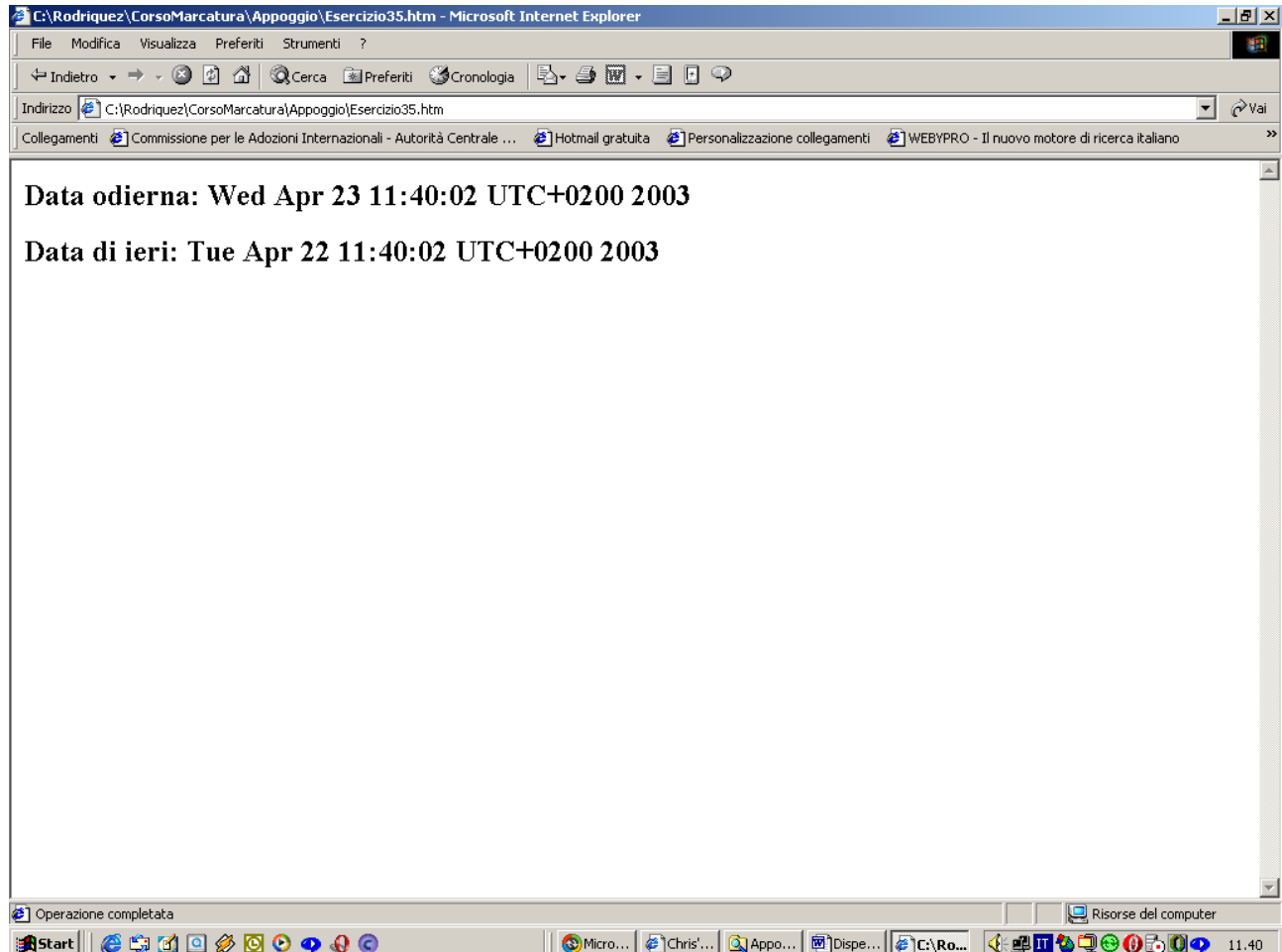
Esempio Operazioni sulle date

```
<html>
  <head>
    <title>Operazioni sulle date</title>
  </head>
  <body onLoad="Dater()">
    <script language="javascript">
      <!--
        function Dater() {
          var oggi = new Date();
          var ieri = new Date();
          var diff = oggi.getDate() - 1;

          ieri.setDate(diff);

          document.write("<h2>Data odierna: " + oggi + "</h2>");
          document.write("<h2>Data di ieri: ");
          document.writeln(ieri + "</h2>");
          document.close();
        }
      //-->
    </script>
  </body>
</html>
```

Il codice produce il seguente risultato:



Esempio Operazioni con gli eventi

```
<html>
  <head>
    <title>Operazioni con gli eventi</title>
  </head>
  <body ONLOAD="PaginaCaricata()" ONUNLOAD="Saluto()">
    <h1>Operazioni con gli eventi</h1>
    <p>
      <a href="#" ONMOUSEOVER="Mouse()">Un esempio di hyperlink</a>
    </p>

    <form>
      <input type="button"
        value="Clicca!"
        onClick="Cliccato()"
      </form>

  <script language="javascript">
    <!--
      function PaginaCaricata(){
        alert("La pagina è stata caricata");
      }

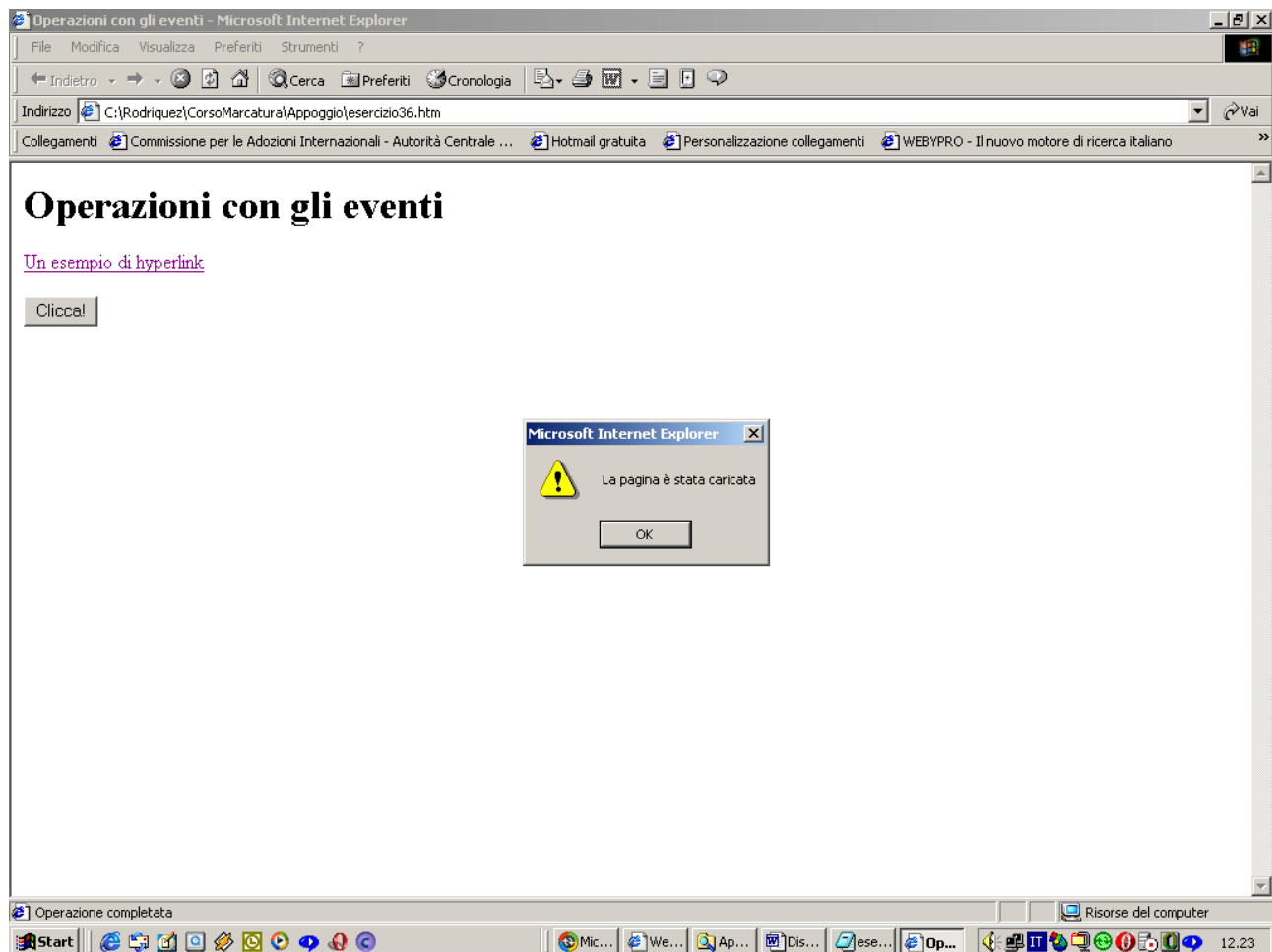
      function Saluto() {
        alert("Arrivederci, grazie per aver usato questa pagina");
      }

      function Cliccato() {
        alert("Cliccato sul bottone");
      }

      function Mouse() {
        alert("Il mouse è sul link");
      }

    //-->
  </script>
</body>
</html>
```

Il codice produce il seguente risultato



Gli eventi possono essere definiti, in questo contesto, come una qualsiasi *variazione* che l'utente determina nello *stato* del browser.

Quando viene rilevato un evento è possibile generare altre azioni attraverso l'implementazione di un gestore di eventi (*event handler*, script da elaborare quando l'evento viene rilevato), costituito da funzioni Javascript che si associano ad elementi HTML nel codice sorgente della pagina.

Nel codice esemplificativo tutti gli eventi eseguono la stessa azione, ovvero l'apertura di una finestra di alert contenente un diverso messaggio (stringa di caratteri).

Al caricamento della pagina si verifica l'evento OnLoad al quale è associata una funzione che presenta una finestra di alert contenente la stringa "La pagina è stata caricata".

Cliccando sul bottone o posrtando il cursore sul link si eseguono le rispettive funzioni previste nel codice della pagina.

Infine, chiudendo la pagina, si verifica l'evento OnUnLoad al quale è associata la funzione Saluto()).

Dal codice si può notare che i nomi degli eventi usati non sono case-sensitive (ad es. OnLoad è uguale ad ONLOAD).

Esempio Testo sulla barra di stato

```
<html>
  <head>
    <title>Testo sulla barra di stato</title>
  </head>
  <body>
    <a href="#" OnMouseOver="status='Messaggio personalizzato sulla barra del
testo';
    return true;" onMouseOut="status='';">Hyperlink</a>

  </body>
</html>
```

Il codice produce il seguente risultato

