Linguaggi e tecnologie per il Web

prof. Riccardo Rosati
corso di Laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma, a.a. 2015/2016

Esercitazione di riepilogo

Esercizio 1

(a) Scrivere un documento HTML contenente una form contenente i seguenti campi:

- cognome e nome (casella di testo editabile lunga 40 caratteri)
- sesso (selezionabile tramite due bottoni radio)
- matricola (casella di testo editabile lunga 12 caratteri)
- regione di residenza (da scegliere da un menu che riporta le 20 regioni italiane)
- email (casella di testo editabile lunga 30 caratteri)
- telefono (casella di testo editabile lunga 15 caratteri)
- anno di corso (casella di testo editabile lunga 2 caratteri)
- richieste particolari (area di testo editabile di 12 righe per 60 colonne)
- bottone di invio
- bottone di reset

(b) Aggiungere al documento HTML una funzione JavaScript che esegue i seguenti controlli:

(b1) verifica che il cognome e nome non sia vuoto;

(b2) verifica che sia stata selezionata una regione;

(b3) verifica che l'anno di corso sia un numero compreso tra 1 e 6 oppure sia la stringa "FC";

(b4) verifica che o l'email o il telefono siano non vuoti.

Inoltre, fare in modo che, nel documento HTML, tale funzione JavaScript venga eseguita quando l'utente invia la form.

(c) Per ognuno dei controlli specificati al punto (b), dire se è realizzabile in HTML5 senza utilizzare codice JavaScript, e in caso positivo, spiegare come.

Soluzione

(a) Documento HTML contenente la form richiesta:

```
<html>
<head>
</head>
<body>
<form action="" method="post" name="registr">
```

```
cognome:
<input type="text" name="cognome" size="40" maxlength="40">
<br>
sesso:
<input type="radio" name="sesso" value="M">M
<input type="radio" name="sesso" value="F">F
<br>
matricola:
<input type="text" name="matricola" size="12" maxlength="12">
<br>
regione:
<select name="regione">
<option value="nessuna" selected></option>
<option value="valdaosta">Val d'Aosta</option>
<option value="piemonte">Piemonte</option>
<option value="liguria">Liguria</option>
<option value="lombardia">Lombardia</option>
<option value="veneto">Veneto</option>
<option value="trentino">Trentino Alto Adige</option>
<option value="friuli">Friuli Venezia-Giulia</option>
<option value="emilia">Emilia-Romagna</option>
<option value="toscana">Toscana</option>
<option value="marche">Marche</option>
<option value="umbria">Umbria</option>
<option value="lazio">Lazio</option>
<option value="abruzzo">Abruzzo</option>
<option value="molise">Molise</option>
<option value="campania">Campania</option>
<option value="basilicata">Basilicata</option>
<option value="puglia">Puglia</option>
<option value="calabria">Calabria</option>
<option value="sicilia">Sicilia</option>
<option value="sardegna">Sardegna</option>
</select>
<br>
email:
<input type="text" name="email" size="30" maxlength="30">
<br>
telefono:
<input type="text" name="tel" size="15" maxlength="15">
<br>
anno di corso:
<input type="text" name="anno" size="2" maxlength="2">
<br>
richieste particolari:
<br>
<textarea name="richieste" cols="60" rows="12"></textarea>
```

```
<br>
<input type="submit" value="Invia">
<input type="reset" value="Reset">
</form>
</body>
</html>
```

(b) Documento HTML contenente, oltre alla form, la funzione Javascript richiesta:

```
<html>
<head>
<script type="text/javascript" language="javascript">
function validaForm() {
  if (document.registr.cognome.value=="") {
    alert("Inserire cognome");
    return false;
  }
  if (document.registr.matricola.value=="") {
    alert("Inserire matricola");
    return false;
  }
  if (document.registr.regione.value=="nessuna") {
    alert("Selezionare una regione");
    return false;
  }
  if ((document.registr.email.value=="")&&(document.registr.tel.value=="")) {
    alert("Inserire o l'email o il numero di telefono");
    return false;
  }
  if (document.registr.anno.value!="FC") {
    if (isNaN(document.registr.anno.value)||document.registr.anno.value=="") {
      alert("Anno di corso errato");
      return false;
    }
    else {
      var v=parseInt(document.registr.anno.value);
      if ((v<1)||(v>6)) {
        alert("Anno di corso errato");
        return false;
      }
    }
  }
  alert("Dati inseriti correttamente");
  return true;
}
</script>
</head>
```

```
<body>
<form action="" method="post" name="registr" onSubmit="return validaForm();">
cognome:
<input type="text" name="cognome" size="40" maxlength="40">
<br>
sesso:
<input type="radio" name="sesso" value="M">M
<input type="radio" name="sesso" value="F">F
<br>
matricola:
<input type="text" name="matricola" size="12" maxlength="12">
<br>
regione:
<select name="regione">
<option value="nessuna" selected></option>
<option value="valdaosta">Val d'Aosta</option>
<option value="piemonte">Piemonte</option>
<option value="liguria">Liguria</option>
<option value="lombardia">Lombardia</option>
<option value="veneto">Veneto</option>
<option value="trentino">Trentino Alto Adige</option>
<option value="friuli">Friuli Venezia-Giulia</option>
<option value="emilia">Emilia-Romagna</option>
<option value="toscana">Toscana</option>
<option value="marche">Marche</option>
<option value="umbria">Umbria</option>
<option value="lazio">Lazio</option>
<option value="abruzzo">Abruzzo</option>
<option value="molise">Molise</option>
<option value="campania">Campania</option>
<option value="basilicata">Basilicata</option>
<option value="puglia">Puglia</option>
<option value="calabria">Calabria</option>
<option value="sicilia">Sicilia</option>
<option value="sardegna">Sardegna</option>
</select>
<br>
email:
<input type="text" name="email" size="30" maxlength="30">
<br>
telefono:
<input type="text" name="tel" size="15" maxlength="15">
<br>
anno di corso:
<input type="text" name="anno" size="2" maxlength="2">
<br>
richieste particolari:
```

```
<br>
<textarea name="richieste" cols="60" rows="12"></textarea>
<br>
<input type="submit" value="Invia">
<input type="reset" value="Reset">
</form>
</body>
</html>
```

(c) I primi tre controlli del punto precedente sono realizzabili in HTML5 senza includere script. In particolare:

(b1) verifica che il cognome e nome non sia vuoto:

```
<input type="text" name="cognome" size="40" maxlength="40" required>
```

(b2) verifica che sia stata selezionata una regione;

```
<select name="regione" required>
```

(b3) verifica che l'anno di corso sia un numero compreso tra 1 e 6 oppure sia la stringa "FC";

```
<input type="text" name="anno" size="2" maxlength="2" pattern="1|2|3|4|5|6|FC">
```

Infine, il controllo (b4) (verifica che o l'email o il telefono siano non vuoti) non può essere codificato in HTML5 senza l'uso di script.

Esercizio 2a Dato il seguente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE a [
  <!ELEMENT a (b?, (c | d)*, e+)>
  <!ELEMENT b (#PCDATA)>
  <!ELEMENT c EMPTY>
  <!ELEMENT d (#PCDATA | a)*>
  <!ELEMENT e EMPTY>
  <!ATTLIST a prop CDATA #REQUIRED>
  <!ATTLIST b prop CDATA #IMPLIED>
  <!ATTLIST c attrc CDATA #IMPLIED>
  <!ATTLIST e attre CDATA #REQUIRED>
]>
<a prop="v1">
  <b prop="1">riga 1</b>
  <b prop="pippo">riga 2</b>
  <c/>
  <b x="3">riga 3</b>
  <d>riga 4
```

```
    <a prop="v2">
      <c prop="xyz"/>
      <e/>
    </a>
  </d>
  <e attre="38"/>
</a>
```

dire se il documento è valido. In caso contrario, evidenziare le violazioni della DTD da parte del documento.

Esercizio 2b Data la seguente DTD:

```
<!DOCTYPE a [
  <!ELEMENT a (a*,(u|s|z),a*,y,((t,y)|(w,z)))>
  <!ELEMENT s (#PCDATA|w)*>
  <!ELEMENT t (x?,w*,(t|a|s)?)>
  <!ELEMENT u (a?,(u|s)*,(z,y)+)>
  <!ELEMENT x (#PCDATA|x)*>
  <!ELEMENT y (#PCDATA|x)*>
  <!ELEMENT w EMPTY>
  <!ELEMENT z ((z,y)*,(u|s)+)>
  <!ATTLIST s attrs CDATA #REQUIRED>
  <!ATTLIST u attru CDATA #IMPLIED>
  <!ATTLIST x attrx CDATA #REQUIRED>
]>
```

1. dire se la DTD è corretta ed in caso negativo evidenziare gli errori presenti e correggerli;
2. scrivere un documento XML che sia valido rispetto alla DTD (eventualmente corretta) e che contenga tutti gli elementi dichiarati nella DTD.

Esercizio 2c Dato il seguente documento XML:

```
<a prop="v1">
  <b prop="1">riga 1</b>
  <b prop="pippo">riga 2</b>
  <c/>
  <b x="3">riga 3</b>
  <d>riga 4
    <a prop="v2">
      <c prop="xyz"/>
      <e/>
    </a>
  </d>
  <e attre="38"/>
</a>
```

1. scrivere l'albero DOM corrispondente a tale documento;

2. scrivere una sequenza di chiamate a metodi della API DOM che genera l'albero DOM corrispondente al documento.

Esercizio 2d

1. Scrivere una DTD che formalizza le seguenti regole:

   - sono ammessi solo gli elementi `<a>`, `<b>`, `<c>`, `<d>`, `<e>`;
   - `<a>` è l'elemento radice;
   - `<a>` può contenere solo elementi di tipo `<b>` o di tipo `<c>`, deve contenere almeno un elemento di tipo `<c>`, e tutti gli elementi `<b>` devono precedere gli elementi `<c>`;
   - `<b>` può contenere solo elementi di tipo `<d>` o di tipo `<e>`, deve contenere almeno un elemento `<e>`, e tutti gli elementi `<d>` devono precedere gli elementi `<e>`;
   - `<c>` può contenere solo elementi di tipo `<d>` o di tipo `<a>`, deve contenere almeno tre elementi `<d>`, e tutti gli elementi `<d>` devono precedere gli elementi `<a>`;
   - gli elementi `<d>` possono contenere ogni tipo di elemento (compreso `#PCDATA`);
   - `<e>` è un elemento vuoto;
   - `<a>` ha un attributo `attrx` obbligatorio di tipo `CDATA` e un attributo `attry` che può assumere solo i valori s1, s2, o s3, e s1 è il valore di default;
   - `<e>` ha un attributo `attrz` obbligatorio di tipo `CDATA` e un attributo `attrw` opzionale di tipo `NMTOKEN`.

2. scrivere una grammatica non contestuale G tale che $\mathcal{L}(G)$ corrisponde all'insieme dei documenti XML validi rispetto alla DTD del punto precedente.

Esercizio 3 Data la seguente DTD:

```
<!DOCTYPE a [
  <!ELEMENT a (b?, c*, (d|e)+)>
  <!ELEMENT b (#PCDATA)>
  <!ELEMENT c (#PCDATA)>
  <!ELEMENT d (#PCDATA)>
  <!ELEMENT e EMPTY>
  <!ATTLIST c attr CDATA #IMPLIED>
]>
```

scrivere un XML Schema corrispondente.

Soluzione

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="a">
    <xsd:complexType>
      <xsd:sequence>
```

```
        <xsd:element name="b" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="c" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                <xsd:attribute name="attr" type="xsd:string" use="optional"/>
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:choice minOccurs="1" maxOccurs="unbounded">
          <xsd:element name="d" type="xsd:string"/>
          <xsd:element name="e">
            <xsd:complexType/>
          </xsd:element>
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Esercizio 4a Scrivere un foglio di stile XSL che, dato un documento XML, restituisce il documento tale che: 1) l'elemento root è uguale all'elemento root del documento di input; 2) ogni elemento `x` diverso dall'elemento radice viene trasformato in un elemento `nuovo`, copiando il nome dell'elemento `x` in un elemento `nome` contenuto nell'elemento `nuovo`.

Ad esempio, se il documento XML di input è il seguente:

```
<root>
  <b>
    <a>testo1</a>
    <c>
      <a/>
      testo2
      <d>
        <a>testo3</a>
      </d>
    </c>
    testo4
    <a/>
  </b>
</root>
```

il foglio di stile applicato al documento deve restituire il documento seguente:

```
<root>
  <nuovo>
    <nome>b</nome>
```

```
    <nuovo>
      <nome>a</nome>
       testo1
    </nuovo>
    <nuovo>
      <nome>c</nome>
      <nuovo>
        <nome>a</nome>
      </nuovo>
      testo2
      <nuovo>
        <nome>d</nome>
        <nuovo>
          <nome>a</nome>
          testo3
        </nuovo>
      </nuovo>
    </nuovo>
    testo4
    <nuovo>
      <nome>a</nome>
    </nuovo>
  </nuovo>
</root>
```

Soluzione

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
  <xsl:output method="xml"/>

  <xsl:template match="/*">
    <xsl:element name="{name()}">
      <xsl:apply-templates/>
    </xsl:element>
  </xsl:template>

  <xsl:template match="*">
    <nuovo>
      <nome>
        <xsl:value-of select="name()"/>
      </nome>
      <xsl:apply-templates/>
    </nuovo>
  </xsl:template>

</xsl:stylesheet>
```

Esercizio 4b Scrivere un foglio di stile XSL che, dato un documento XML, restituisce il documento tale che: 1) l'elemento radice è uguale all'elemento radice del documento di input; 2) ogni elemento figlio dell'elemento radice viene trasformato in un elemento `<z>`, e il suo contenuto viene ricorsivamente trasformato; 3) ogni elemento che non è né la radice né un figlio dell'elemento radice viene trasformato in un elemento `nuovo`, con valore dell'attributo `elem` uguale al nome dell'elemento, e il suo contenuto viene ricorsivamente trasformato; 4) tutte le parti testuali vengono copiate nel corrispondente elemento in output.

Ad esempio, se il documento XML di input è il seguente:

```
<c>
  <b>
    <a>testo1</a>
    <c>
      <a/>
      testo2
      <d>
        <a>testo3</a>
      </d>
    </c>
    testo4
    <a/>
  </b>
  <a>
    <b>testo1</b>
    testo5
    <d/>
  </a>
</c>
```

il foglio di stile applicato al documento deve restituire il documento seguente:

```
<c>
  <z>
    <nuovo elem="a">testo1</nuovo>
    <nuovo elem="c">
      <nuovo elem="a"/>
      testo2
      <nuovo elem="d">
        <nuovo elem="a">testo3</nuovo>
      </nuovo>
    </nuovo>
    testo4
    <nuovo elem="a"/>
  </z>
  <z>
    <nuovo elem="b">testo1</nuovo>
    testo5
```

```
    <nuovo elem="d"/>
  </z>
</c>
```

Soluzione

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
  <xsl:output method="xml"/>

  <xsl:template match="/*">
    <xsl:element name="{name()}">
      <xsl:apply-templates/>
    </xsl:element>
  </xsl:template>

  <xsl:template match="/*/*">
    <z>
      <xsl:apply-templates/>
    </z>
  </xsl:template>

  <xsl:template match="/*/*//*">
    <xsl:element name="nuovo">
      <xsl:attribute name="elem">
        <xsl:value-of select="name()"/>
      </xsl:attribute>
      <xsl:apply-templates/>
    </xsl:element>
  </xsl:template>

</xsl:stylesheet>
```

Esercizio 5 Scrivere un documento HTML che permette di selezionare, tramite un menu a 3 opzioni, il caricamento asincrono di 3 diversi documenti HTML in una zona del documento stesso.

Soluzione

```
<!DOCTYPE HTML>
<html>
  <body>
    <form name="modulo" action="">
      Scegli documento:
      <select name="scelte" onChange="caricaDocumento;">
        <option>Documento_1</option>
```

```
        <option>Documento_2</option>
        <option>Documento_3</option>
      </select>
    </form>
    <hr/>
    <div id="zonaDinamica">
      Seleziona il documento da visualizzare
    </div>
    <hr/>
    Resto del documento<br/>
    ...
    <script>
      function caricaDocumento(e) {
        var httpRequest = new XMLHttpRequest();
        httpRequest.onreadystatechange = gestisciResponse;
        httpRequest.open("GET",
          "http://www.dis.uniroma1.it/~rosati/lw/esercizi-AJAX/" +
          this.value + ".htm", true);
        httpRequest.send();
      }
      function gestisciResponse(e) {
        if (e.target.readyState == XMLHttpRequest.DONE &&
        e.target.status == 200) {
          document.getElementById("zonaDinamica").innerHTML
          = e.target.responseText;
        }
      }
    </script>
  </body>
</html>
```

Esercizio 6 Scrivere un documento HTML che utilizza JQuery per risolvere il precedente esercizio 5.

Soluzione

```
<html>
  <body>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js">
    </script>
    <script>
    $(document).ready(function(){
      $("#scelte").change(function(e){
        $("#zonaDinamica").load("http://www.dis.uniroma1.it/~rosati/lw/esercizi-finali/"
        +e.target.value+".htm",
        function(responseTxt, statusTxt, xhr){
```

```
          if(statusTxt == "error") alert("Errore"+xhr.status+":"+xhr.statusText);
        });
      });
    });
    </script>
    <form name="modulo" action="">
      Scegli documento:
      <select name="scelte" id="scelte">
        <option>Documento_1</option>
        <option>Documento_2</option>
        <option>Documento_3</option>
      </select>
    </form>
    <hr/>
    <div id="zonaDinamica">
      Seleziona il documento da visualizzare
    </div>
    <hr/>
    Resto del documento<br/>
    ...
  </body>
</html>
```