Linguaggi e tecnologie per il Web 2018/2019
Facsimile di un compito d'esame

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 funzioni JavaScript che eseguono i seguenti controlli:

(b1) verifica che il cognome e nome non sia vuoto. Questa verifica va fatta ogni volta che l'utente cambia il valore del campo cognome e nome;

(b2) verifica che sia stata selezionata una regione. Questa verifica va fatta all'atto dell'invio della form;

(b3) verifica che l'anno di corso sia un numero compreso tra 1 e 6 oppure sia la stringa "FC". Questa verifica va fatta ogni volta che l'utente cambia il valore del campo anno di corso;

(b4) verifica che o l'email o il telefono siano non vuoti. Questa verifica va fatta all'atto dell'invio della 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 e la funzione Javascript richiesta:

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" language="javascript">
    function controllaCognomeNome() {
      if (document.registr.cognome.value=="") {
        alert("Inserire cognome");
        return false;
      }
      return true;
    }
    function controllaAnno() {
      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;
          }
        }
      }
      return true;
```

```
  }
  function validaForm() {
    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;
    }
    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" onChange="return controllaCognomeNome();">
    <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" onChange="return controllaAnno();">
    <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) I primi tre controlli del punto precedente sono realizzabili in HTML5 senza includere script. In particolare:

(a1) verifica che il cognome e nome non sia vuoto:

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

(a2) verifica che sia stata selezionata una regione;

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

(a3) 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 (a4) (verifica che o l'email o il telefono siano non vuoti) non può essere codificato in HTML5 senza l'uso di script.

Esercizio 2 Scrivere un documento HTML contenente una form contenente i seguenti campi:

- matricola (casella di testo editabile lunga 12 caratteri)
- cognome e nome (casella di testo editabile lunga 60 caratteri)
- data di nascita (casella di tipo date)
- sesso (menu a due opzioni, F e M)
- corso frequentato (menu a tendina contenente una lista di corsi)
- bottone di invio
- bottone di stampa della local storage degli utenti

Aggiungere funzioni JavaScript e un foglio di stile CSS tali che:

1. i dati relativi ad ogni utente devono essere memorizzati, all'atto del loro invio, nella local storage;
2. il bottone di stampa della local storage deve visualizzare, nella parte finale del documento HTML, gli utenti memorizzati nella local storage;
3. il foglio di stile CSS deve fare in modo che i dati relativi agli studenti maschi siano visualizzati con colore e font diverso dai dati relativi alle studentesse femmine. Non si possono usare CSS inlined (solo CSS incorporati).

Soluzione

Documento HTML contenente la form richiesta:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      tr.M {color:blue; font-family: "Courier New";}
      tr.F {color:red; font-family: Arial;}
    </style>
    <script>
      function inizializzaStorageUtenti(){
        if (typeof(localStorage.utenti) == "undefined") {
          localStorage.utenti="[]";
        }
      }
```

```
      function resetStorageUtenti(){
        localStorage.utenti="[]";
      }

      function stampaStorage(){
        var u = JSON.parse(localStorage.utenti);
        var l = u.length;
        var s = new String("<h3>Stato di localStorage:</h3>");
        s +="<table border=1><tr><th>cognome e nome</th><th>data di nascita</th><th>sesso</th></tr>";
        for (i=0;i<l;i++)
          s += "<tr class="+u[i].sesso+"><td>"+u[i].cogn+"</td><td>"+u[i].data+"</td><td>"+u[i].sesso+"</td></tr>";
        s += "</table>";
        document.getElementById("vistaStorage").innerHTML = s;
        return true;
      }

      function inserisciUtente() {
        if (document.registrazione.cognome.value=="") {
          alert("Inserire cognome");
          return false;
        }
        if (document.registrazione.dataNascita.value=="") {
          alert("Inserire data di nascita");
          return false;
        }
        var u = JSON.parse(localStorage.utenti);
        var nextpos = u.length;
        var o = { cogn:document.registrazione.cognome.value,
                  data:document.registrazione.dataNascita.value,
                  sesso:document.registrazione.sesso.value };
        alert("Dati inseriti correttamente");
        u[nextpos] = o;
        localStorage.utenti = JSON.stringify(u);
        return true;
      }
    </script>
  </head>
  <body onload="inizializzaStorageUtenti()">
    <form action="" method="post" name="registrazione" onSubmit="return inserisciUtente();" accept-charset="utf-8">
      cognome e nome:
      <input type="text" name="cognome" size="60" maxlength="60"/>
      <br/>
      data di nascita:
      <input type="date" name="dataNascita" size="10" maxlength="10"/>
      <br/>
      sesso:
      <select name="sesso" id="sesso">
        <option value="F">F</option>
        <option value="M">M</option>
      </select>
      <br/>
      <input type="submit" value="Invia"/>
      <input type="button" value="Stampa Storage" onclick="stampaStorage();"/>
      <input type="button" value="Reset Storage" onclick="resetStorageUtenti();"/>
    </form>
    <hr/>
    <div id="vistaStorage"></div>
    <hr/>
  </body>
</html>
```

Esercizio 3 Dato il seguente documento HTML:

```
<html>
  <body>
```

```
    inizio documento
    <hr/>
    <div>
      prima zona di visualizzazione dei documenti
    </div>
    <hr/>
    <div>
      seconda zona di visualizzazione dei documenti
    </div>
    <hr/>
    fine documento
  </body>
</html>
```

Modificare il documento HTML in modo da poter selezionare, tramite un menu con 8 opzioni, il caricamento asincrono di 8 diversi documenti HTML (nomi dei documenti: `doc1.htm`, `doc2.htm`, `doc3.htm`, `doc4.htm`, `doc5.htm`, `doc6.htm`, `doc7.htm`, `doc8.htm`): i primi quattro documenti vanno visualizzati all'interno del primo elemento `div`, mentre gli altri quattro documenti vanno visualizzati all'interno del secondo elemento `div`.

Soluzione

```
<html>
  <head>
    <meta charset="UTF-8">
    <title>AJAX: esercizio esame</title>
  </head>
  <body>
    inizio documento
    <hr/>
    <div>
      <button>doc1</button>
      <button>doc2</button>
      <button>doc3</button>
      <button>doc4</button>
      <button>doc5</button>
      <button>doc6</button>
      <button>doc7</button>
      <button>doc8</button>
    </div>
    <hr/>
    <div id="zona1">
      prima zona di visualizzazione dei documenti
    </div>
    <hr/>
    <div id="zona2">
      seconda zona di visualizzazione dei documenti
    </div>
    <hr/>
    fine documento
    <script>
      var bottoni = document.getElementsByTagName("button");
      for (var i = 0; i < bottoni.length; i++) {
        bottoni[i].onclick = caricaDocumento;
      }
      function caricaDocumento(e) {
        var httpRequest = new XMLHttpRequest();
        httpRequest.prevTarget = e.target;
        httpRequest.onreadystatechange = gestisciResponse;
        httpRequest.open("GET", e.target.innerHTML + ".htm", true);
        httpRequest.send();
      }
      function gestisciResponse(e) {
        if (e.target.readyState == 4 && e.target.status == 200) {
          var c=e.target.prevTarget.innerHTML.charAt(3);
```

```
          if (c<="4") c="1"; else c="2";
          document.getElementById("zona"+c).innerHTML = e.target.responseText;
        }
      }
    </script>
  </body>
</html>
```

Esercizio 4 Scrivere un documento HTML che risolve il precedente esercizio 5 utilizzando JQuery.

Soluzione

```
<html>
  <head>
    <meta charset="UTF-8">
    <title>JQuery: esercizio esame</title>
  </head>
  <body>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js">
    </script>
    inizio documento
    <hr/>
    <div>
      <button>doc1</button>
      <button>doc2</button>
      <button>doc3</button>
      <button>doc4</button>
      <button>doc5</button>
      <button>doc6</button>
      <button>doc7</button>
      <button>doc8</button>
    </div>
    <hr/>
    <div id="zona1">
      prima zona di visualizzazione dei documenti
    </div>
    <hr/>
    <div id="zona2">
      seconda zona di visualizzazione dei documenti
    </div>
    <hr/>
    fine documento
    <script>
      $(document).ready(function(){
        $("button").click(function(){
        var c=this.innerHTML.charAt(3);
        if (c<="4") c="1"; else c="2";
          $("#zona"+c).load(this.innerHTML+".htm",
            function(responseTxt, statusTxt, xhr){
               if(statusTxt == "error") alert("Errore" + xhr.status + ":" + xhr.statusText);
            });
        });
      });
    </script>
  </body>
</html>
```

Esercizio 5 Data la seguente DTD:

```
<!DOCTYPE r [
  <!ELEMENT r ((t,x)|(s,y))+>
  <!ELEMENT s (y,u)+>
  <!ELEMENT t (x,v)>
```

```
  <!ELEMENT u ((t|s|v)?,((r|x|y)+,y)*)>
  <!ELEMENT v ((u|x)+,(x|y)+,(y|u)+)>
  <!ELEMENT x (#PCDATA|t|x)*>
  <!ELEMENT y (EMPTY)>
  <!ATTLIST u attru CDATA #IMPLIED>
  <!ATTLIST v attrv CDATA #REQUIRED
              attrvv CDATA #IMPLIED>
]>
```

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 almeno una occorrenza di ogni elemento dichiarato nella DTD.

Soluzione

1) La DTD è corretta.

2) Un documento XML valido rispetto alla DTD è il seguente:

```
<?xml version="1.0"?>
<r>
  <t>
    <x>ciao</x>
    <v attrv="abcde">
      <u/>
      <x/>
      <y/>
    </v>
  </t>
  <x>ciao ciao</x>
  <s>
    <y/>
    <u/>
  </s>
  <y/>
</r>
```

Esercizio 6 Scrivere un foglio di stile XSL che, dato un documento XML, restituisce il documento tale che:

1. l'elemento radice di input viene trasformato in un elemento `root`, e viene creato per tale elemento un attributo che ha per nome `nomeRoot` e che ha per valore il nome dell'elemento di input corrente. Inoltre, il contenuto dell'elemento di input viene ricorsivamente trasformato;
2. ogni elemento che è figlio dell'elemento radice viene copiato in output, e il contenuto di tale elemento viene ricorsivamente trasformato;
3. per ogni elemento che è figlio di un figlio dell'elemento radice viene creato in output un elemento `nipote` come figlio della radice, e il contenuto di tale elemento viene ricorsivamente trasformato;
4. per ogni elemento che è figlio di un figlio di un figlio dell'elemento radice viene creato in output un elemento `pronipote` come figlio della radice, e viene aggiunto a tale elemento un sottoelemento il cui nome è `nomePronipote` e il cui contenuto testuale è il nome dell'elemento di input corrente. Inoltre il contenuto di tale elemento viene ricorsivamente trasformato;
5. ogni altro elemento viene copiato in output come figlio dell'elemento radice. Inoltre il contenuto di tale elemento viene ricorsivamente trasformato;
6. ogni nodo di tipo testo del documento non viene copiato in output.

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

```
<a>
  <x>testo 1</x>
  <b>
    <c><w>testo 2</w></c>
```

```
  </b>
  <w>
    <y>testo 3
      <d><z>testo 4</z></d>
    </y>
  </w>
</a>
```

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

```
<root nomeRoot="a">
  <x/>
  <b/>
  <nipote/>
  <pronipote><nomePronipote>w</nomePronipote></pronipote>
  <w/>
  <nipote/>
  <pronipote><nomePronipote>d</nomePronipote></pronipote>
  <z/>
</root>
```

Soluzione

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml"/>

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

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

  <xsl:template match="/*/*/*">
    <xsl:element name="nipote">
    </xsl:element>
    <xsl:apply-templates/>
  </xsl:template>

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

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

  <xsl:template match="text()"/>

</xsl:stylesheet>
```