Linguaggi e tecnologie per il Web

prof. Riccardo Rosati
Corso di laurea in Ingegneria informatica e automatica
Sapienza Università di Roma, a.a. 2018/2019

Esercizi su JavaScript, DOM e Web Storage

Esercizio 1 Scrivere un documento HTML contenente una form contenente i seguenti campi:

- cognome (casella di testo editabile lunga 40 caratteri)
- nome (casella di testo editabile lunga 30 caratteri)
- matricola (casella di testo editabile lunga 12 caratteri)
- bottone di invio
- bottone di reset
- bottone di stampa della local storage degli utenti
- bottone di reset della local storage degli utenti

I dati relativi ad ogni utente devono essere memorizzati, all'atto del loro invio, nella local storage. Il bottone di stampa della local storage deve visualizzare, nella parte finale del documento HTML, gli utenti memorizzati nella local storage. Il bottone di reset della local storage deve azzerare la local storage degli utenti.

Soluzione

Documento HTML contenente la form richiesta:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1"/>
<script>
function inizializzaStorageUtenti(){
  if (typeof(localStorage.utenti) == "undefined") {
    localStorage.utenti="[]";
  }
}

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

function stampaStorageSemplice(){
  var u = JSON.parse(localStorage.utenti);
  var l = u.length;
  var s = new String("<h3>Stato di localStorage:</h3>");
  for (i=0;i<l;i++)
    s += JSON.stringify(u[i])+"<br/>";
```

```
  document.getElementById("vistaStorage").innerHTML=s;
  return true;
}

function stampaStorageTabella(){
  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</th><th>nome</th><th>matricola</th></tr>";
  for (i=0;i<l;i++)
    s += "<tr><td>"+u[i].c+"</td><td>"+u[i].n+"</td><td>"+u[i].m+"</td></tr>";
  s += "</table>";
  document.getElementById("vistaStorage").innerHTML = s;
  return true;
}

function uguali(u,o){
  if ((u.c==o.c)&&(u.n==o.n)&&(u.m==o.m))
    return true;
  return false;
}

function inserisciUtente() {
  if (document.registrazione.cognome.value=="") {
    alert("Inserire cognome");
    return false;
  }
  if (document.registrazione.nome.value=="") {
    alert("Inserire nome");
    return false;
  }
  if (document.registrazione.matricola.value=="") {
    alert("Inserire matricola");
    return false;
  }
  var u = JSON.parse(localStorage.utenti);
  var nextpos = u.length;
  var o = { c:document.registrazione.cognome.value,
            n:document.registrazione.nome.value,
            m:document.registrazione.matricola.value };
  for (i=0;i<nextpos;i++)
    if(uguali(u[i],o)) {
      alert("Utente gi inserito");
      return false;
    }
  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:
<input type="text" name="cognome" size="40" maxlength="40">
<br>
nome:
<input type="text" name="nome" size="30" maxlength="30">
<br>
matricola:
<input type="text" name="matricola" size="12" maxlength="12">
<br>
<input type="submit" value="Invia">
<input type="button" value="Stampa Storage Semplice" onclick="stampaStorageSemplice();">
<input type="button" value="Stampa Storage Tabella" onclick="stampaStorageTabella();">
<input type="button" value="Reset Storage" onclick="resetStorageUtenti();">
<input type="reset" value="Reset form">
</form>
<hr/>
<div id="vistaStorage"></div>
<hr/>
</body>
</html>
```

Si è scelto di rappresentare ogni utente come un oggetto contenente le tre proprietà `c`, `n`, `m` che rappresentano rispettivamente il cognome, il nome e il numero di matricola. Un array di tali oggetti (in forma serializzata) è rappresentato nella proprietà `utenti` dell'oggetto `localStorage` (si ricorda che l'oggetto `localStorage` può solo contenere proprietà di tipo stringa).

Si noti che sono stati inseriti due bottoni di stampa della local storage degli utenti: il primo attiva una funzione che stampa una riga per ogni utente, e ogni utente è rappresentato dalla sequenza delle tre coppie (proprietà, valore) che lo compongono; il secondo bottone attiva una funzione che visualizza le stesse informazioni usando una tabella HTML che presenta in ogni riga i dati relativi ad un utente.

Si noti anche l'uso della funzione `inizializzaStorageUtenti` che verifica, appena avvenuto il caricamento del documento HTML da parte del browser (evento `onload`), che sia definita la proprietà `localStorage.utenti` (altrimenti la crea inizializzandola all'array vuoto).

Esercizio 2 Scrivere uno script che funzioni su qualsiasi documento HTML e che sia tale che:

- ogni volta che il puntatore passa su un qualsiasi elemento `div`, questo cambia colore, ovvero se il colore attuale è quello di default, passa al colore rosso, e viceversa;
- ogni volta che l'utente clicca su un qualsiasi elemento di classe c1, questo diventa di colore blu;
- ogni volta che l'utente fa doppio click su un qualsiasi elemento di classe c1, questo sparisce;
- ogni volta che l'utente clicca su un qualsiasi elemento `span` di classe c2, questo elemento diventa di colore verde.

Soluzione

```
<!DOCTYPE html>
<html>
<script>
function cambiaColore(e){
  if (e.target.style.color=="red")
    e.target.style.color="initial";
  else e.target.style.color="red";
}
function nascondi(e){
  e.target.style.visibility="hidden";
}
function coloraVerde(e){
  e.target.style.color="green";
}
function coloraBlu(e){
  e.target.style.color="blue";
}
function ripristinaColore(e){
  e.target.style.color="initial";
}
function assegnaEventHandlers(){
  var divDoc=document.getElementsByTagName("div");
  for (i=0;i<divDoc.length;i++)
    divDoc[i].addEventListener("mouseenter",cambiaColore);
  var class1=document.getElementsByClassName("c1");
  for (i=0;i<class1.length;i++){
    class1[i].addEventListener("click",coloraBlu);
    class1[i].addEventListener("dblclick",nascondi);
  }
  var span2=document.querySelectorAll("span.c2");
  for (i=0;i<span2.length;i++){
    span2[i].addEventListener("mouseenter",coloraVerde);
    span2[i].addEventListener("mouseleave",ripristinaColore);
  }
}
</script>
</head>
<body onload="assegnaEventHandlers();">
<div>Div 1</div>
<div>Div 2</div>
<div>Div 3</div>
<span class="c1">Primo span classe 1</span><br/>
<span class="c2">Primo span classe 2</span><br/>
<span class="c1">Secondo span classe 1</span><br/>
```

```
<span class="c2">Secondo span classe 2</span><br/>
<span class="c1">Terzo span classe 1</span><br/>
<span class="c2">Terzo span classe 2</span>
<div class="c1">Questo  un div di classe 1</div>
<div class="c2">Questo  un div di classe 2</div>
</body>
</html>
```

Si noti l'uso di diversi metodi d'accesso agli elementi della pagina relativi all'oggetto `document`. In particolare:

- `document.getElementsByTagName("div")` restituisce una NodeList contenente tutti i nodi corrispondenti agli elementi `div` del documento;
- `document.getElementsByClassName("c1")` restituisce una NodeList contenente tutti i nodi corrispondenti agli elementi del documento che hanno valore dell'attributo `class` uguale a c1;
- `document.querySelectorAll("span.c2")` restituisce una NodeList contenente tutti i nodi corrispondenti agli elementi `span` del documento che hanno valore dell'attributo `class` uguale a c2.

Si noti inoltre l'uso del metodo `elem.addEventListener("nomeEvento",f)` che associa all'elemento `elem` la funzione `f` come gestore (event handler) dell'evento `nomeEvento`.

Molto importante: l'uso del metodo `elem.addEventListener` prevede che le funzioni utilizzate come event handler abbiano come unico argomento un oggetto di tipo evento (`Event`). L'oggetto passato a tali funzioni rappresenta ovviamente l'evento che si è verificato. Tale oggetto ha una proprietà `target` che contiene l'elemento su cui è avvenuto l'evento. Pertanto, negli event handler di questo esercizio, che usano tutte il parametro formale `e` per rappresentare l'evento passato a tali funzioni, vengono settate proprietà di formattazione dell'oggetto `e.target`.