Tecnologie Cloud e Mobile

Lez. 04

JavaScript 2

Giuseppe Psaila

Università di Bergamo

giuseppe.psaila@unibg.it

Eventi, Eccezioni e Pagina HTML

Completiamo l’introduzione del linguaggio JavaScript

- Eventi
- Eccezioni
- Oggetti predefiniti Window e Document
- Manipolazione della pagina HTML attraverso il DOM
- Gestione delle Form

Eventi

Che cosa è un Evento?

- Un evento si scatena quando qualcosa accade sugli elementi della pagina HTML
- Per esempio:
  - Un click del mouse in un punto della pagina
  - La pagina è stata caricata tutta
  - L’utente ha premuto il tasto «Invia» in una form

Che cosa è un Evento?

- Se gli eventi non vengono gestiti, il browser ne gestisce solo alcuni, secondo quanto previsto dallo standard
- Ma è possibile aggiungere dei «gestori di evento», che sono funzioni JavaScript
- Gli attributi *onEvent* (dove *Event* indica un evento generico) hanno questo scopo

Esempio

- si vuole intercettare il click su «Clicca qui»

```
<p   onClick="return fReazione();" >>Clicca qui
</p>
```

Unnamed Function

- L’attributo onClick specifica il gestore dell’evento
- Il suo valore è

  return fReazione();
- Diventa una unnamed function

  function ()

  { return fReazione(); }

Convenzione degli eventi

Il gestore dell'evento deve restituire un valore

- false
  l'evento viene fermato, il browser non prosegue
- true
  l'evento non viene fermato, il browser prosegue con la gestione dell'evento

Quali Eventi?

- Ogni elemento HTML ha eventi specifici che possono essere specificati.
- In genere, onClick è previsto dalla maggior parte degli elementi

Quali Eventi?

- L’elemento BODY prevede onLoad, che viene invocato quando tutta la pagina è stata caricata
- Consente di svolgere attività di inizializzazione di vario tipo.

Quali Eventi?

- L’elemento FORM prevede l’evento onSubmit
- Questo evento viene generato quando l’utente preme il pulsante «Submit»
- Se la gestione dell’evento non viene fermata, parte la chiamata HTTP

Eventi e Form

- Ma perché dovremmo fermare la chiamata HTTP?
- Per esempio, perché ci sono errori nella form
- Quindi non facciamo partire la chiamata HTTP

Esempio di Form

```
<form action="…"
   onSubmit="return controlla();">
 <input type="text" name="email"/>
 <input type="submit" value="Invia"/>
</form>
```

Esempio di Form

- L’attributo **`acion`** in **`form`** contiene l’indirizzo del programma sul server al quale mandare il contenuto della form
- Il primo elemento **`input`** definisce un campo di testo con nome **`email`**
- Il secondo elemento **`input`** definisce il pulsante di invio

Esempio di Form

- Che comportamento vogliamo ottenere?
- La form deve consentire all'utente di inviare la sua email al server
- Prima di inviarla, la funzione «**`controlla`**» verifica che il valore del campo sia effettivamente un indirizzo email
- Se non lo è, segnala l'errore e ferma l'invio (restituendo false)

Eccezioni

Eccezioni

- Le versioni più recenti di JavaScript hanno introdotto il concetto di *Eccezione*
- L'idea è semplice: una porzione di codice viene *monitorata* durante la sua esecuzione
- Se un errore occorre, un'*eccezione* viene generata

Eccezioni

- L’eccezione può essere catturata, evitando che il programma si interrompa in modo inaspettato per l’utente
- Quando un’eccezione viene catturata, si esegue una procedura che gestisce la situazione inaspettata
- Mantenendo vivo il programma

Strittira

```
try
{   …  /* Codice monitorato */  }
catch(e)
{ … /* Gestione dell'errore */ }
finally(e)    // opzionale
{ … /* sempre eseguito */ }
```

Proprietà dell'Eccezione

Proprietà dell'eccezione «e»

- name, il nome dell'eccecczione
- message, il messaggio dell'eccezione

- Attenzione: non si possono definire nuove eccezioni

Esempio

```
try { colours[2] = "red"; }
catch (e) {
 alert("An exception occurred." +
   "Error name: " + e.name +
   ". Error message: " +
    e.message);
}
```

Esempio

Il messaggio stampato è:

```
An exception occurred.
 Error name: ReferenceError
 Error message: colours is not
 defined
```

Esempio

Se il blocco catch è vuoto, l'eccezione viene ignorata (ma non fate così, **ORRORE**)

```
try { colours[2] = "red"; }
catch (e) { }
```

Tipo di Eccezione

Se necessario, si può sapere anche il tipo di eccezione

```
catch (e) {
   if (e instanceof TypeError)
 {
 alert("Bad or undefined variable!");
 }
}
```

Tipi di Eccezione

- EvalError
  la funzione eval() ha valutato un comando scorretto
- RangeError
  un valore numerico eccede il limite di rappresentazione
- ReferenceError
  un riferimento sbagliato è usato

Tipi di eccezione

- SyntaxError
  un errore di sintassi è stato trovato nel codice
- TypeError
  un errore di type checking si è verificato
- URIError
  le funzioni encodeURI() e decodeURI() sono state invocate su un URI scorretto

Oggetti Predefiniti

Oggetti Predefiniti

L’interprete JavaScript del Browser fornisce una serie di oggetti predefiniti che consentono di accedere al browser, alla pagina, alla navigazione, ecc.

Window

- L’oggetto predefinito Window è l’oggetto di più alto livello
- Contiene al suo interno tutti gli altri oggetti
- Ha molti campi e metodi. Vediamo quelli più utili

Window

Campi

- **document** L’oggetto che descrive il contenuto del documento mostrato nella finestra
- **frames[]** un Array che descrive tutti i frame definiti nella finestra
- **history** l’oggettto che rappresenta la storia della navigazione

Window

Campi

- **name** Il nome della finestra
- **self** Un riferimento all’oggetto stesso
- **window** Un sinonimo di self

Window

Metodi

- **alert(msg)** Mostra una finestra con il messaggio specificato (stringa) e il pulsante OK
- **blur()** Toglie il focus dalla finestra
- **close()** Chiude la finestra
- **focus()** Dà il focus alla finestra

Window

Metodi

- **Boolean confirm(msg)** Mostra il messaggio (stringa) con i pulsanti OK e Cancel. Restituisce true se il pulsante OK è stato premuto
- **print()** Stampa il contenuto della finestra

Window

Metodi

- **String prompt(msg)** Mostra una dialog box che richiede un valore, che viene restituito.

```
var yourname=window.prompt(
   "please enter your name");
alert(yourname);
```

Window

Metodi

- **Window open(URL, [name], [features], [replace])**
  Apre una nuova finestra, con il nome indicato (stringa), il documento nell'URL indicato, le caratteristiche (stringa features) indicate. replace (booleano) indica se il nuovo URL va aggiunto alla history

Esempio di Nuova Window

```
Win = window.open(
"http://www.dynamicdrive.com",
 "",  // nessun nome
"width=800px, height=600px, " +
 "resizable");
```

Proprietà delle Window

- **height**, altezza
- **location**, se "yes" la barra degli indirizzi viene mostrata
- **menubar**, se "yes" la barra del menu viene mostrata
- **resizable**, se "yes" la finestra può essere ridimensionata

Proprietà delle Window

- **scrollbars**, se "yes" le barre di scorrimento sono abilitate
- **status**, se "yes" la barra di stato viene mostrata
- **toolbar**, se "yes" la barra degli strumenti viene mostrata
- **width**, larghezza

Document

- L’oggetto document rappresenta la pagina HTML su cui si sta lavorando
- Il contenuto è rappresentato in formato DOM (vedi dopo)
- Il contenuto può essere modificato dal codice JavaScript
- Fornisce molti campi e metodi

Document

Campi

- **alinkColor** specifica il colore del link attivo nel documento (attributo Alink)
- **all[]** Soltanto in IE (dalla versione 4) un array che contiene tutti gli elementi nel documento. Si può usare come **`document.all["elementID"]`** oppure **`document.all.elementID`** per accedere ad un elemento di cui si conosce l'ID

Document

Campi

- **anchors[]** Un array che contiene tutte le ancore (link) nel documento
- **applets[]** Un array che contiene tutti gli applet nel documento
- **bgColor** Una stringa che specifica il colore di sfondo

Document

Campi

- **cookie** Una stringa che contiene le coppie “nome/valore” dei cookie
- **domain** Contiene il dominio del server dal quale il documento proviene
- **embeds[]** Un array che raccoglie tutti gli oggetti embedded nel documento (marcatore EMBED)

Document

Campi

- **fgColor** Una stringa che specifica il colore di default del testo
- **forms[]** Un array contenente tutte le form del documento
- **images[]** Un array contenente tutte le immagini del documento
- **lastModified** La data di ultima modifica (dal Web Server)

Document

Campi

- **linkColor** Specifica il colore dei link non visitati
- **links[]** Un array con tutti i link della pagina
- **plugins[]** Lo stesso di embed[]
- **referrer** L'URL della pagina dalla quale si è giunti

Document

Campi

- **title** il titolo del documento (può essere modificato nei browser moderni)
- **URL** l'URL completo della pagina
- **vlinkColor** Il colore dei link visitati

Document

Metodi

- **getElementById(ID)** Restituisce l'elemento con l'ID (Stringa) specificato
- **getElementsByTagName(name)** Restituisce gli elementi con il nome del tag (marcatore) specificato

Document

Metodi

- **open([mineType])** Apre uno stream che consente di aggiornare il documento. Se specificato, mimeType specifica il tipo mime
- **write(string)** Scrive una stringa nello stream del documento
- **writeln(string)** scrive una stringa nello stream del documento e va a capo
- **close()** chiude lo stream del documento

Esempio

```
win2=window.open("")
win2.document.open()
win2.document.write(
"<b>Some text</b>")
win2.document.close()
```

DOM e HTML

HTML 5

- HTML 5 è a tutti gli effetti aderente allo standard XML.
- Ma per quanto riguarda la struttura DOM, i dettagli sintattici sono irrilevanti
- Quindi DOM è stato usato fin da subito per rappresentare la pagina HTML

Document

- L’oggetto predefinito Document contiene la rappresentazione DOM della pagina
- Quando il DOM viene modificato, il browser rigenera la visualizzazione (rendering) della pagina
- In questo modo, il codice JavaScript controlla ciò che è visualizzato

HTMLElement

- La classe Element del DOM
- viene ulteriormente estesa
- dalla classe HTMLElement
- che aggiunge campi e metodi specifici per gli elementi HTML
- A sua volta, ha 54 sotto-classi (una per ognuno degli elementi HTML)

Gerarchia

HTMLElement

Campi (più importanti):

- Readonly NodeList **childNodes**
- Array **children**
- String **innerHTML**
  decrive il contenuto HTML dell'elemento; cambiando il suo valore, si cambia il codice HTML del contenuto e la pagina viene aggiornata

HTMLElement

- String **className**
  la classe di stile
- String **id**
  l'identificatore dell'elemento
- String **lang**
  la lingua del contenuto dell'elemento
- String **style**
  la specifica dello stile in CSS

HTMLElement

Campi (ereditati da Element):

- attributes, childNodes, firstChild, lastChild,
- nextSibling, nodeName, nodeType,
- nodeValue, parentElement, parentNode,
- previousSibling

HTMLElement

Metodi:

- Void **insertAdjacentHTML**(String where,
  String HTMLText)

where vale

- BeforeBegin,
- AfterBegin,
- BeforeEnd,
- AfterEnd.

HTMLElement

Metodi :

- (ereditati da Element) getAttribute, getElementsByTagName, removeAttribute, setAttribute
- (ereditati da Node) appendChild, cloneNode, hasChildNodes, insertBefore, removeChild, removeNode, replaceChild

L’attributo id

- In HTML, tutti gli elementi possono avere un attributo di nome «id»
- Il suo ruolo è identificare univocamente l’elemento
- In questo modo, attraverso il metodo getElementById dell’oggetto predefinito document si può ottenere facilmente il nodo che lo descrive

Esempio

```
<html>
 <body onLoad="prepara();">
  <div id="area"></div>
 </body>
</html>
```

Codice

```
function prepara()
{
 var nodo =
    document.getElementById("area");
 nodo.innerHTML = "<p>CIAO</p>";
}
```

Form

- Per accedere ai campi di una form è meglio non passare dal DOM
- È meglio sfruttare la proprietà «**`name`**»di **`form`** e **`input`**

Esempio di Form

```
<form action="…" name="myform"
  onSubmit="return controlla();">
   <input type="text" name="email"/>
   <input type="submit"
          value="Invia"/>
</form>
```

Accesso al Campo

`document.forms.myform.email.value`

- **`forms`** è il campo che contiene tutte le form
- Il nome della form identifica un oggetto che descrive tutta la form
- A sua volta, ogni nome di campo corrisponde ad un campo nell'oggetto
- La proprietà **`value`** è il valore del campo della form

Conversione da Stringa a Numero

- Come convertire una stringa in un numero?
- **`parseInt( s )`**
  interpreta la stringa come un numero intero e lo restituisce
- **`parseFloat( s )`**
  interpreta la stringa come un numero in virgola mobile e lo restituisce

Conversione da Stringa a Numero

- E se la stringa non contiene un numero?
- Il valore restituito è **`NaN`** (Not a Number)
- È una sorta di valore neutro
- Come verificare se il valore è **`NaN`**?
- Esiste una funzione booleana **`isNaN( v )`**

La funzione eval

Valutazione dinamica del codice

- La funzione **`eval`** consente di eseguire (valutare) la stringa ricevuta come parametro attuale
- Un qualsiasi codice JavaScript può essere contenuto nella stringa
- Quindi, un programma JavaScript potrebbe eseguire codice proveniente dall'esterno

JavaScript Injection

- Se il codice interagisce con il server (vedi modalità AJAX)
- potrebbe ricevere del codice maligno, che potrebbe fare danni
- Evitare il rischio di **JavaScript injection**: non usate la funzione eval