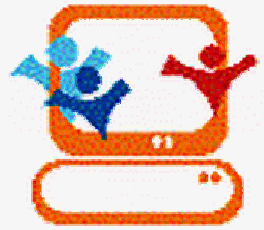
I linguaggi interpretati nei programmi utente



L'accessibilità degli script lato client

Commissione interministeriale permanente
per l'impiego delle ICT a favore delle categorie deboli o svantaggiate
Segreteria tecnico scientifica

Roberto Castaldo
r.castaldo@iol.it

I linguaggi di script

- La maggior parte dei linguaggi di scripting si basa su una serie di istruzioni eseguite al momento della lettura o di una chiamata da parte del browser

I linguaggi di script

- Queste istruzioni sono strutturate in
  - Blocchi di istruzioni eseguibili
  - Blocchi funzionali (con un nome univoco che può essere richiamato)
  - Blocchi ciclici
  - Blocchi condizionali

I linguaggi di script

- Ogni linguaggio utilizza vari tipi di dati *(stringhe, array, operatori booleani, numeri e così via)* e implementa diversi comandi per il loro trattamento

I linguaggi di script

- Questi linguaggi si sono evoluti di pari passo con i browser
- Condividono una base comune: lo standard **ECMASCRIPT**
  - Lo stesso ECMASCRIPT è stato sottoposto a 3 revisioni con successivi perfezionamenti, che sono stati assorbiti dai singoli linguaggi

I linguaggi di script

- Gli script sono collocati nella struttura HTML delle pagine:
  - All'interno dell'elemento <*script*> oppure
  - Direttamente in linea all'interno dei singoli tag

JacaScript ed altri...

- 1994: Netscape Corporation crea **JavaScript 1.0**
  - Permette di aggregare stringhe, fare calcoli, ordinare array e di poter creare dinamicamente pagine e contenuti HTML
- Microsoft: IE implementa...
  - **JScript**: una variante di JavaScript,
  - **VBScript**: una semplificazione di Visual Basic
- **JavaScript** è stato implementato in modo quasi uniforme nelle sue varie versioni all'interno di pressoché tutti i browser

I linguaggi di script

- Nelle prime versioni di JavaScript nessuno pensò a una standardizzazione
  - Questo per molto tempo ha prodotto grandi diversità di implementazione pur rivolgendosi alle stesse funzionalità nei vari browser

Gli script e il DOM

- *Il DOM è un interfaccia indipendente da browser, piattaforma e linguaggi di programmazione che consente a script e programmi di accedere dinamicamente a contenuto, struttura e stile di un documento*

Gli script e il DOM

- Proprio il DOM oggi rappresenta la vera forza di JavaScript
- Permette la creazione di vere applicazioni dinamiche on-line, nel browser, senza software aggiuntivi
  - Le potenzialità di JavaScript in combinazione al DOM sono enormi

Script: consigli

- Una sorta di “linee guida” che permettono di creare script funzionanti su tutte le piattaforme (anche quelle meno usuali o potenti) e in tutti i browser (anche quelli di nicchia)

Script: consigli

- Adottare le diverse versioni del linguaggio all'interno di blocchi dopo avere identificato il corretto supporto delle istruzioni utilizzate nello script
  - Si utilizzeranno blocchi condizionali
  - Le tecniche d'identificazione, o sniffing, permettono di identificare il supporto a singoli browser o sistemi operativi oppure, più semplicemente, il supporto ai singoli comandi

Script: consigli

- Ottimizzare il codice evitando i cicli ricorsivi o la saturazione della memoria del browser
  - Questo significa orientare le strutture alla programmazione a oggetti e strutturare il codice

Script: consigli

- Gestire gli errori nei browser
  - Se e dove possibile
- Evitare lunghi cicli di animazione o trasformazione del documento
- Non basare l'esecuzione del codice da eventi che ricorrono senza interruzione da parte del browser

Script: consigli

- Verificare la gestione degli eventi nella programmazione
  - Essa risulta diversa per quasi tutti i linguaggi e i browser disponibili
- Verificare sempre l'esistenza di un determinato oggetto o elemento HTML nel DOM del documento prima di utilizzarlo

Script: consigli

- Non richiedere al browser l'esecuzione di alcuna operazione prima che l'intero documento sia stato caricato
  - Gestore di eventi onload in HTML
- Testare *fino allo sfinimento* il codice su tutte le piattaforme e tutti i browser

JavaScript: struttura sintattica

```
<script type="text/javascript">

<!--
//questo è un blocco funzione
function dimmiciao(){
//questa è un istruzione
alert('ciao');
}
//questo è il richiamo alla funzione che ne
permette l'esecuzione
dimmiciao();
//-->

</script>
```

Un’applicazione JavaScript

Il DOM all'opera:
trasformare i pixel in em

Idea e realizzazione a cura di
Luca Mascaro e Livio Mondini

http://www.lucamascaro.info/projects/pixeltrasformer/
www.tiuvizeta.it/provapixel.htm

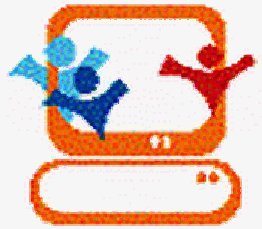
Trasformare i Pixel in Em

- IE ha un grave problema di accessibilità:
  - Non permette di rimensionare i caratteri la cui misura sia espressa in *pixel* con i comandi del menu *Visualizza>Caratteri*
- La soluzione?

  Trasformiamo i pixel in em!

Pixel _ Em Transformer.html

Gli script e le WCAG



Le linee guida del WGAC per la creazione di script accessibili

6.3 WCAG 1.0 (p1)

- *Garantire che le pagine siano utilizzabili quando script, applet, o altri oggetti di programmazione sono disabilitati oppure non supportati*
- *Se questo non è possibile, fornire informazione equivalente in una pagina accessibile alternativa*

6.3 WCAG 1.0 (p1)

- Si deve inserire il tag <*noscript*> nel *body* del documento
  - Interviene in tutti i browser quando JavaScript non è supportato o abilitato.
- All'interno di *noscript* dovranno essere forniti i contenuti necessari a sostituire completamente le funzionalità dello script

Tag *noscript*: esempio

```
<script type="text/javascript">
<!--
// questo comando scrive un normale testo
all'interno del documento, attenzione non
utilizza DOM
document.write('<p>ciao</p>');
//-->
</script>
…
<noscript>
<p>ciao</p>
</noscript>
```

6.4 WCAG 1.0 (p2)

- *Garantire che i gestori di eventi per gli script e le applet siano indipendenti dai dispositivi di input*

Gestori di eventi

- Non utilizzare gestori di eventi che siano legati a un particolare metodo di input
  - Utilizzare gestori pressoché universali per tutte le tipologie di navigazione, oppure
  - Sommare più gestori di eventi specifici ma che si completino →

Gestori di eventi

Un’altra soluzione più flessibile:

- “Agganciare" a un oggetto HTML una funzionalità JavaScript tramite DOM, ricorrendo alla gestione degli eventi prevista dagli stessi linguaggi di script, solo dopo aver identificato con precisione che quel particolare mezzo è supportato e può essere utilizzato
  - Molti sviluppatori agganciano l'evento onmouseover senza verificare che effettivamente il mouse si sposti sul monitor e che quindi venga realmente utilizzato
  - Stiamo però oltrepassando gli scopi di questo corso

Gestori di eventi: esempio

```
<script type="text/javascript">
<!--
//attenzione manca la parte di
identificazione che va personalizzata
di volta in volta
document.getElementById('para').onmouseove
r = alert('ciao');
//-->
</script>
…
<p id="para">ciao</p>
```

8.1 WCAG 1.0 (p1/2)

- *Fare in modo che elementi di programmi come script e applet siano direttamente accessibili o compatibili con le tecnologie assistive*

8.1 WCAG 1.0 (p1/2)

- E’ un requisito spesso assai complesso da soddisfare
  - Si parla di funzionalità rese compatibili con le tecnologie assistive

Alcuni esempi:

Esempi:

Problema:

- Un utente che naviga tramite tastiera, attiva un evento su un link all'inizio del documento, che inserisce un paragrafo alla fine del documento. L'utente si accorge di tale mutamento, ma per raggiungere tale paragrafo dovrà scorrere l'intera pagina

Soluzione:

- Si deve fornire un collegamento diretto (magari tramite ancore legate a delle accesskey) vicino all'area che ha subito una modifica

Esempi:

Problema:

- Un utente non vedente aziona un evento su un link all'inizio del documento, che produce l'inserimento di un paragrafo alla fine del documento. L'utente non può accorgersi di tale mutamento

Soluzione:

- Bisogna comunicare il mutamento subito dalla pagina e fornire un collegamento diretto (magari tramite ancore legate a delle accesskey) vicino all'area che ha subito una modifica

9.2 WCAG 1.0 (p2)

- *Garantire che ogni elemento dotato di una sua specifica interfaccia possa essere gestito in una modalità indipendente dal dispositivo.*

9.2 WCAG 1.0 (p2)

- Fornire le eventuali interfacce create con JavaScript in maniere indipendente dal dispositivo
- Essa dovrà essere utilizzabile con qualunque dispositivo che vi acceda
  - Tecniche di sniffing
  - CSS
    - JavaScript produce comunque dell'HTML, soggetto alle normali regole dei fogli di stile

9.3 WCAG 1.0 (p2)

- *Negli script, specificare gestori di evento logici piuttosto che gestori di evento dipendenti dal dispositivo.*
  - Questo punto di controllo specifica meglio l'uso all'interno della priorità 6.4 dei gestori d'eventi universali