Programmazione Web - JavaScript 2

4.1 - Eventi

Cos'è un Evento?

Un evento si scatena quando qualcosa accade sugli elementi della pagina HTML. Esempi includono:

- Un click del mouse in un punto della pagina
- Il caricamento completo della pagina
- La pressione del tasto "Invia" in un modulo

Gestione degli Eventi

Se gli eventi non vengono gestiti, il browser ne gestisce solo alcuni secondo lo standard. Tuttavia, è possibile aggiungere "gestori di evento", che sono funzioni JavaScript. Gli attributi `onEvent` (dove Event indica un evento generico) hanno questo scopo.

Esempio di Gestione Evento

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

Funzione Anonima

L'attributo `onClick` specifica il gestore dell'evento. Il suo valore diventa una funzione anonima:

```
function () {
    return fReazione();
}
```

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

Tipi di Eventi

Eventi Specifici per Elementi HTML

- La maggior parte degli elementi supporta `onClick`
- L'elemento `BODY` prevede `onLoad`, che viene invocato quando la pagina è completamente caricata
- L'elemento `FORM` prevede `onSubmit`, generato quando l'utente preme il pulsante "Submit"

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

Scenario Tipico

- La form deve consentire all'utente di inviare un'email al server
- Prima dell'invio, la funzione `controlla()` verifica che il valore del campo sia un indirizzo email valido
- Se non lo è, segnala l'errore e ferma l'invio (restituendo `false`)

4.2 - Eccezioni

Le versioni più recenti di JavaScript hanno introdotto il concetto di Eccezione:

- Una porzione di codice viene monitorata durante l'esecuzione
- Se si verifica un errore, viene generata un'eccezione

Gestione delle Eccezioni

L'eccezione può essere catturata per evitare l'interruzione inaspettata del programma. Quando catturata, si esegue una procedura di gestione.

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

Proprietà dell'Eccezione

- `name` : nome dell'eccezione
- `message` : messaggio dell'eccezione

Esempio

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

Tipi di Eccezione

- `EvalError` : errore nella funzione `eval()`
- `RangeError` : valore numerico oltre il limite di rappresentazione
- `ReferenceError` : riferimento sbagliato
- `SyntaxError` : errore di sintassi nel codice
- `TypeError` : errore di type checking
- `URIError` : errore nelle funzioni `encodeURI()` e `decodeURI()`

Verifica del Tipo di Eccezione

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

4.3 - Oggetti Predefiniti

L'interprete JavaScript del browser fornisce oggetti predefiniti che consentono di accedere a:

- Browser
- Pagina
- Navigazione

Oggetto Window

Oggetto di più alto livello che contiene tutti gli altri oggetti.

Campi Principali

- `document` : oggetto che descrive il contenuto del documento
- `frames[]` : array dei frame definiti nella finestra
- `history` : oggetto che rappresenta la storia della navigazione
- `name` : nome della finestra
- `self` : riferimento all'oggetto stesso
- `window` : sinonimo di `self`

Metodi Principali

- `alert(msg)` : mostra una finestra con un messaggio
- `confirm(msg)` : mostra un messaggio con pulsanti OK/Cancel
- `prompt(msg)` : mostra una dialog box per inserimento valore
- `open(URL, name, features, replace)` : apre una nuova finestra
- `print()` : stampa il contenuto della finestra

```
win = window.open(
    "http://www.esempio.com",
    "",
    "width=800px, height=600px, resizable"
);
```

Oggetto Document

Rappresenta la pagina HTML:

- Contenuto in formato DOM
- Modificabile tramite JavaScript

Campi Principali

- `all[]` : array di tutti gli elementi (solo IE)
- `anchors[]` : array di tutti i link
- `bgColor` : colore di sfondo
- `cookie` : stringa con coppie nome/valore dei cookie
- `domain` : dominio del server

- `forms[]` : array di tutti i form
- `images[]` : array di tutte le immagini
- `title` : titolo del documento
- `URL` : URL completo della pagina

Metodi Principali

- `getElementById(ID)` : restituisce l'elemento con l'ID specificato
- `getElementsByTagName(name)` : restituisce gli elementi con quel tag
- `write(string)` : scrive una stringa nel documento
- `writeln(string)` : scrive una stringa e va a capo

4.4 - DOM e HTML

HTML 5 e DOM

- HTML 5 è aderente allo standard XML
- DOM è stato usato fin da subito per rappresentare la pagina HTML

HTMLElement

Estensione della classe `Element` del DOM con:

- 54 sottoclassi (una per ogni elemento HTML)
- Campi e metodi specifici per elementi HTML

Campi Principali

- `innerHTML` : contenuto HTML dell'elemento
- `className` : classe di stile
- `id` : identificatore dell'elemento
- `style` : specifica dello stile CSS

Metodi

- `insertAdjacentHTML()` : inserisce HTML in posizioni specifiche
- Metodi ereditati: `getAttribute()` , `setAttribute()` , `appendChild()` , etc.

Form e Accesso ai Campi

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

Accesso al campo:

```
document.forms.myform.email.value
```

Conversioni

Da Stringa a Numero

- `parseInt(s)` : converte in numero intero
- `parseFloat(s)` : converte in numero in virgola mobile
- `isNaN(v)` : verifica se il valore non è un numero

Funzione eval()

- Consente di eseguire codice JavaScript da una stringa
- **Attenzione**: rischio di JavaScript injection
- **Consiglio**: evitare l'uso di `eval()`