Appunti Javascript - Parte 2

Javascript client-side

Oggetti predefiniti del browser

- **`window`** : Oggetto top-level che rappresenta la finestra del browser.
  - Proprietà e metodi per gestire posizione, dimensioni e altre finestre (es. `open()`).
- **`navigator`** : Informazioni sul client (browser) come nome, versione, plugin, cookie, ecc.
- **`location`** : URL del documento corrente.
  - Modificabile per effettuare redirect (es. `window.location = "url"`).
- **`history`** : Array degli URL visitati.
  - Proprietà: `length` , `current` , `next` .
  - Metodi: `back()` , `forward()` , `go()` .
- **`document`** : Rappresenta il contenuto del documento.
  - Permette l'accesso a tutti gli elementi tramite proprietà e metodi (es. `document.title` , `document.forms[0]`).
  - Rappresenta l'oggetto `DOMDocument` del DOM.

Modello di documento (DOM)

- Ogni elemento nella gerarchia ha proprietà, metodi ed eventi per interazione. *Esempio di utilizzo per validare un form:

```
function verify() {
  if (document.forms[0].elements[0].value == "") {
    alert("Il nome è obbligatorio!");
    document.forms[0].elements[0].focus();
    return false;
  }
  return true;
}
```

Il Document Object Model (DOM)

- **Definizione**: API per documenti HTML e XML. Definisce la struttura logica e come accedere/manipolare il documento.
- **Scopo**: Permette di costruire, navigare, aggiungere, modificare o cancellare elementi del documento.
- **Parsing HTML5:**
  - Il WHATWG ha definito l'algoritmo di parsing di HTML, anche per documenti mal formati (ma validi secondo "HTML Living Standard").
  - L'importante è arrivare a una struttura dati in memoria, il **DOM**.

Struttura di un DOM

- Esempio di struttura ad albero:

```
<table id="tbl-01" class="mytable">
  <tbody>
```

```
    <tr class="first">
      <td>12 maggio</td>
      <td>Mario Rossi</td>
    </tr>
    <tr>
      <td>14 maggio</td>
      <td>Ugo Neri</td>
    </tr>
  </tbody>
</table>
```

Viene rappresentato come un albero di nodi:

- Tipi di nodo:
  1. Elemento.
  2. Attributo.
  3. Testo.

Oggetti del DOM

- **`DOMNode`** : Classe base. Definisce metodi per accedere a tutti i tipi di nodi.
  - Membri e metodi principali:
    - `nodeName` (stringa in maiuscolo)
    - `nodeType` (numero)
    - `children` (array di elementi)
    - `childNodes` (array di tutti i nodi figli)
    - `parentNode` (nodo genitore)
    - `attributes` (array di attributi)
    - `insertBefore()`, `replaceChild()`, `removeChild()`, `appendChild()`, `hasChildNodes()`, `hasAttributes()`
- **`DOMDocument`** : Rappresenta l'intero documento (radice dell'albero).
  - Membri e metodi principali:
    - `docType`
    - `documentElement`
    - `createElement()`, `createAttribute()`, `createTextNode()`
    - `getElementsByTagName()`, `getElementById()`
- **`DOMElement`** : Rappresenta un singolo elemento.
  - Membri e metodi principali:
    - `nodeName`
    - `getAttribute()`, `setAttribute()`, `removeAttribute()`

Javascript e DOM

- Esempi di manipolazione del DOM con Javascript:

```
// Ottenere un elemento per ID e modificarne gli attributi
var c = document.getElementById('c35');
c.setAttribute('class', 'prova1');
c.removeAttribute('align');
```

```
// Creare un nuovo elemento e aggiungerlo al DOM
var newP = document.createElement('p');
var text = document.createTextNode('Ciao Mamma.');
newP.appendChild(text);
c.appendChild(newP);

//Creare lista ordinata
Olist= document.createElement("ol");
voce1 = document.createElement("li");
testo1 = document.createTextNode(“un po’ di testo");
voce1.appendChild(testo1);
Olist.appendChild(voce1);

// Inserire la lista in una data posizione.
div = document.getElementById("lista"); //lista è un ipotetico div.
body = document.getElementsByTagName("body").item(0);
body.insertBefore(Olist,div);
```

`innerHTML` e `outerHTML`

- Permettono di leggere/scrivere il contenuto di elementi come stringhe.
  - `innerHTML` : Contenuto del sottoalbero (escluso il tag radice).
  - `outerHTML` : Contenuto dell'elemento (incluso il tag radice). *Esempi

```
// Dato: <div id="p1"><p>Paragrafo!</p></div>
let a = document.getElementById("p1");
let b = a.innerHTML; // -> <p>Paragrafo!</p>
let c = a.outerHTML; // -> <div id="p1"><p>Paragrafo!</p></div>
a.innerHTML = "<ul><li>Lista!</li></ul>"; // Modifica il contenuto
```

Selettori in DOM

- Metodi standard:
  - `getElementById(id)`
  - `getElementsByName(name)`
  - `getElementsByTagName(tagName)`
- Nuovi selettori (introdotti grazie a JQuery):
  - `getElementsByClassName(className)`
  - `querySelector(cssSelector)` : Restituisce il *primo* elemento che corrisponde al selettore CSS.
  - `querySelectorAll(cssSelector)` : Restituisce *tutti* gli elementi che corrispondono al selettore CSS.

Javascript ed eventi DOM

- Possibilità di associare funzioni (callback) a eventi di oggetti.
- Esempio:

```
// Dichiarazione globale
window.onkeypress = pressed;
window.document.onclick = clicked;

function pressed(e) {
  alert("Key pressed: " + e.which);
}

function clicked() {
  alert("Mouse Click! ");
}

//Dichiarazione locale, all'interno dell'HTML.
<a href="test.htm" onClick="alert('Link!');">cliccarequi</a>
```