Corso base di Javascript

DOM, Document Object Model

DOM, Document Object Model

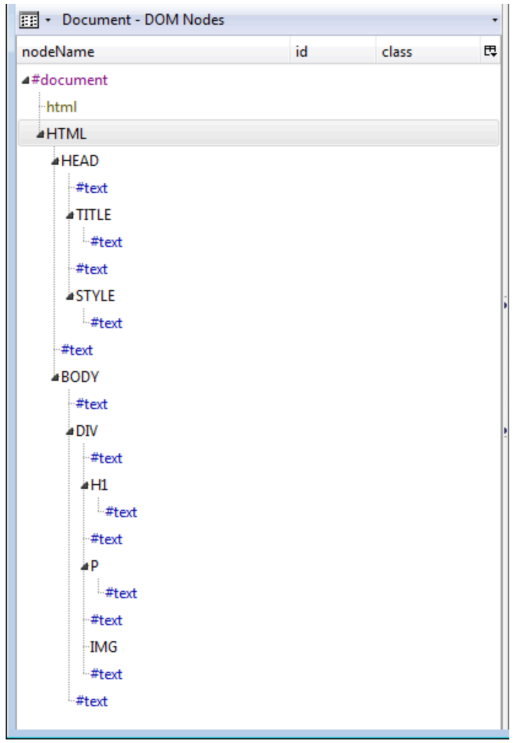
Una delle proprietà dell'oggetto `window` è `document`. Essa rappresenta il documento HTML caricato nella finestra corrente e la struttura di questo oggetto, nota con il nome di **DOM, Document Object Model.**

Il DOM fornisce una rappresentazione del documento come una composizione gerarchica di oggetti, chiamata **DOM tree**.

La radice di tale albero è l'**oggetto document** a cui sono collegati i diversi nodi corrispondenti agli elementi presenti nella pagina.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Pagina d'esempio</title>
  </head>
  <body>
    <div>
      <h1>Titolo</h1>
      <p>Questo &egrave; un paragrafo</p>
      <img src="img.jpg" alt="Immagine">
    </div>
  </body>
</html>
```

——>



Dom:
Tipi di nodi

Il DOM si presenta come un albero con nodi che hanno diversi tipi, esaminiamo brevemente i tipi di nodo di uso comune:

Tipo di nodo	Descrizione
documento	normalmente esiste un solo nodo di questo tipo nella rappresentazione di una pagina HTML e costituisce la **radice dell'albero**; tuttavia in presenza di pagine con frame ci sono più nodi di tipo documento
elemento	un nodo di tipo elemento individua in genere ad un tag HTML, come ad esempio `<body>` , `<div>` , `<p>`
attributo	un nodo di tipo attributo corrisponde ad un attributo di un tag HTML, come ad esempio l'attributo `src` del tag `<img>`
testo	un nodo di tipo testo corrisponde al contenuto testuale di un nodo, compresi eventuali spazi e caratteri speciali

Dom:
Selezionare gli elementi

Il DOM consente di manipolare gli oggetti di documento corrente tramite JavaScript, che ci permette diversi modi per accedere ai nodi che compongono il documento.

```
<p id="mioParagrafo">Questo &egrave; un paragrafo</p>
```

.getElementById()	restituisce un oggetto che rappresenta il nodo di tipo elemento che ha l'attributo `id` con il valore specificato.
.getElementByName()	che restituisce l'elenco dei nodi della pagina il cui valore dell'attributo `name` corrisponde a quello del parametro.
.getElementByTagName()	che restituisce l'elenco dei nodi della pagina in base al loro tag html

Dom: QuerySelector

Tra le novità più introdotte più di recente nelle specifiche del DOM c'è la possibilità di **selezionare gli elementi di una pagina utilizzando i selettori CSS**.

.querySelector()	restituisce il primo elemento trovato. `var p = document.querySelector("#mioParagrafo");`
.querySelectorAll()	restituisce l'elenco di tutti gli elementi trovati `var divList = document.querySelectorAll("div.messaggio");`

Dom: QuerySelector

Tra le novità più introdotte più di recente nelle specifiche del DOM c'è la possibilità di **selezionare gli elementi di una pagina utilizzando i selettori CSS**.

.querySelector()	restituisce il primo elemento trovato. `var p = document.querySelector("#mioParagrafo");`
.querySelectorAll()	restituisce l'elenco di tutti gli elementi trovati `var divList = document.querySelectorAll("div.messaggio");`

Dom:
Modificare gli elementi

Una volta individuato l'elemento o gli elementi presenti su una pagina, possiamo modificarne il contenuto o altre caratteristiche sfruttando proprietà e metodi specifiche dei nodi di tipo elemento.

Ad esempio, la proprietà `innerHTML` rappresenta il contenuto HTML di un elemento ed è accessibile sia in lettura che in scrittura.

```
var p = document.getElementById("mioParagrafo");
p.innerHTML = "Testo del paragrafo";
```

Dom:
Modificare gli elementi

Altri metodi dei nodi di tipo elemento ci consentono di analizzare e modificare i suoi attributi, come quelli riepilogati nella seguente tabella:

Metodo	Descrizione
`hasAttribute(attrName)`	restituisce `true` se l'elemento ha l'attributo specificato come parametro
`hasAttributes()`	restituisce `true` se l'elemento ha almeno un attributo valorizzato
`getAttribute(attrName)`	restituisce il valore dell'attributo specificato come parametro
`setAttribute(attrName, value)`	imposta un valore per un attributo

Dom:
Modificare gli elementi

Un elemento del DOM è sostanzialmente un oggetto JavaScript, possiamo interagire con esso come un qualsiasi altro oggetto.

L'aggiunta di una nuova proprietà all'elemento del DOM non ha alcuna ripercussione sull'HTML, ma può risultare utile per fare delle personalizzazioni tramite JavaScript.

```
var img = document.getElementById("miaImmagine");
img.miaProprieta = "nuovo valore";
```

Le proprietà e gli attributi di un elemento sono elementi distinti e differisco per alcuni comportamenti.

Una proprietà aggiunta ad un elemento del DOM non è gestibile tramite `getAttribute()` e `setAttribute()`, dal momento che non diventa un attributo HTML.

Inoltre, mentre gli attributi non sono `case-sensitive`, cioè le seguenti istruzioni sono equivalenti:

```
img.getAttribute("src");
img.getAttribute("SRC");
```

le proprietà di un oggetto JavaScript lo sono.

Dom:
Navigare i nodi

Alcuni metodi del DOM ci consentono di analizzare e muoverci all'interno della struttura di un documento.

.childNodes	Questa proprietà restituisce un array con tutti i figli di un elemento.
.firstChild()	Questo metodo restituisce il primo figlio dell'elemento.
.lastChild()	Questo metodo restituisce l'ultimo figlio dell'elemento.
.parentNode()	Questo metodo restituisce il genitore dell'elemento.
.nextSibling()	Questo metodo restituisce il fratello successivo dell'elemento.
.previousSibling()	Questo metodo restituisce il fratello precedente dell'elemento.

Dom:
Aggiungere e rimuove elementi

Oltre a modificare il contenuto HTML e gli attributi di un elemento, il DOM ci consente di modificare la struttura di un documento.

Consideriamo il seguente esempio:

```
var mainDiv = document.getElementById("mainDiv");
var img = document.createElement("img");
var srcAttr = document.createAttribute("src");
srcAttr.value = "default.png";
img.setAttributeNode(srcAttr);


mainDiv.appendChild(img);
```

- Abbiamo utilizzato il metodo **createElement()** dell'oggetto `document`, per creare un elemento `<img>`.
- Poi abbiamo creato l'attributo `src` con il metodo **createAttribute()** ed impostato il suo valore.
- Quindi abbiamo associato l'attributo appena creato all'elemento `<img>` tramite **setAttributeNode()**.
- Infine abbiamo aggiunto l'elemento in fondo all'elenco dei nodi figli del `div mainDiv` utilizzando il metodo **appendChild()**.

Dom:
Aggiungere e rimuove elementi

Oltre **appendChild()** abbiamo altre funzioni per gestire gli elementi

.insertBefore()	per inserire un nodo prima di un altro (ad esempio in un elenco). Il metodo prevede due parametri: il nodo da inserire ed il nodo prima del quale inserirlo nella lista. `document.body.insertBefore(img, mainDiv);`
.replaceChild()	Questo metodo sostituisce uno dei figli di un elemento con un nuovo nodo. Facendo riferimento all'esempio di prima, possiamo sostituire il primo un nodo figlio del `div mainDiv` con il nodo `<img>` appena creato `mainDiv.replaceChild(img, mainDiv.firstChild());`
.removeChild()	consentono di eliminare elementi ed attributi dal DOM. `mainDiv.removeChild(mainDiv.firstChild());`