DOM

Riccardo Cattaneo

DOM

Il DOM fornisce una rappresentazione del documento come una composizione gerarchica di oggetti.

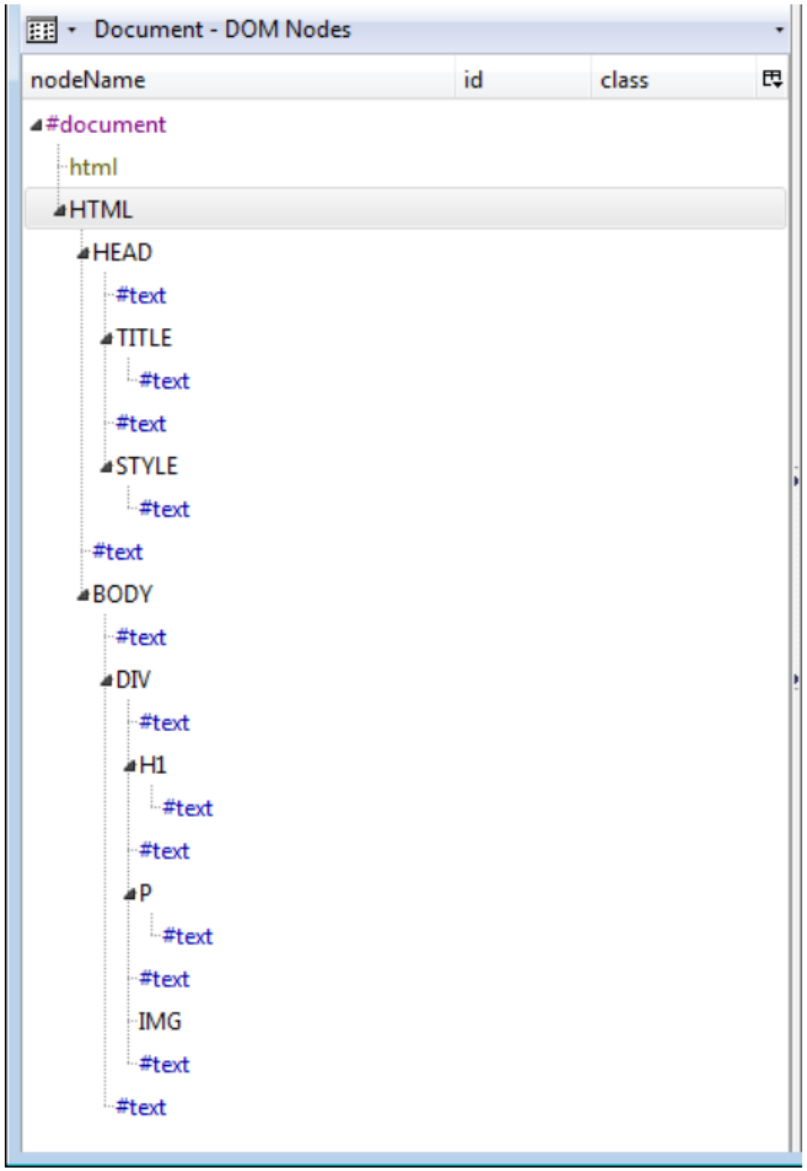
Questo albero di oggetti è pensato per essere facilmente accessibile tramite JavaScript non soltanto in lettura, ma anche per poter cambiare dinamicamente la sua struttura, il contenuto e lo stile.

In altre parole il DOM è un insieme di funzioni, metodi e proprietà, che i programmi possono richiamare al fine di delegare il lavoro al sistema sottostante.

Il DOM non è una parte di JavaScript. Il JavaScript è solo un modo per accedere al DOM, e non è neanche l'unico, ad esempio anche il linguaggio di scripting VBScript di Microsoft è in grado di interpellare ed intervenire sul DOM.

```
<!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>
```

viene visualizzato dal *DOM del browser* in questo modo:



Queste sono le caratteristiche…

1. **Document** (recupera l'elemento o gli elementi. Ad esempio uno specifico div)
2. **Elements** (decido cosa applicare all'elemento recuperato. Ad esempio cambio uno stile)
3. **Attributes** (decido quale attributo applicare)
4. **Events** (decido quale evento utilizzare)
5. **Style** (decido quale stile applicare)

Per completezza visitare il sito :

http://www.w3bai.com/it/jsref/dom_obj_document.html

Oggetto document

"**document**" è molto importante, perché è l'elemento che contiene tutti gli altri elementi del DOM. Di conseguenza, anche "document" corrisponde ad un tag, quello che contiene tutti gli altri e che identifica una pagina per il Web: il tag <html>.

Vediamo i principali metodi di document :

- **getElementById**
- **getElementsByTagName**
- **getElementsByClassName**
- **getElementsByName**

Javascript ci aiuta…

Per individuare una parte della pagina javascript ci viene in aiuto tramite dei comandi dell'oggetto document. Ecco i più importanti…

- **document.url**
- **document.head**
- **document.body**
- **document.links**

Per avere un elenco completo visitare la pagina del sito w3school:

https://www.w3schools.com/jsref/dom_obj_document.asp

getElementById

Per accedere agli elementi della pagina utilizziamo il criterio di selezione **getElementById**

Questo è un paragrafo

Questo metodo, uno dei più utilizzati per la gestione del DOM, restituisce un oggetto che rappresenta il nodo di tipo elemento che ha l'attributo id con il valore specificato.

Se l'elemento non esiste viene restituito il valore **null**, mentre se esistono più elementi con lo stesso id viene restituito il **primo individuato**.

getElementsByTagName

Questo metodo permette di recuperare l'insieme degli elementi caratterizzati dallo stesso tag. In particolare ritorna un array di tutti gli elementi del tag considerato, nell'ordine in cui compaiono nel codice della pagina. La sintassi è:

```
document.getElementsByTagName(nome_TAG)
```

nome_TAG è il **nome del Tag** di cui si vuole recuperare la lista.

Soffermiamoci sul fatto che quest'ultimo metodo ritorna un array. L'array può essere scandito con la consueta sintassi, ovvero usando le parentesi quadre: array[indice].

getElementsByClassName

Questo metodo è simile al precedente, ma fornisce un vettore di tutti gli elementi che appartengono a una medesima classe. La sintassi è:

```
document.getElementsByClassName(nome_classe)
```

nome_classe è il **nome della classe** di cui si vuole recuperare la lista.

getElementsByName

Questo metodo è simile al precedente, ma fornisce un vettore di tutti gli elementi con un nome specificato. La sintassi è:

```
document.getElementsByName(nome_elemento)
```

nome_elemento è il **nome dell'elemento** di cui si vuole recuperare la lista.

Eventi

Gli eventi DOM consentono a JavaScript di registrare diversi gestori di eventi su elementi in un documento HTML.

Gli eventi sono normalmente utilizzati in combinazione con le funzioni, e la funzione non saranno eseguite prima che si verifichi l'evento

Andiamo per gradi

Scegliamo intanto un evento che, al verificarsi… fa qualcosa. Il più semplice è **onclick**… proviamolo subito su un tag :

<button onclick=‘’alert(‘ciao’);’’> Salutami </button>

Se inseriamo questo codice all’interno del body della nostra pagina html possiamo notare il risultato. Anche se molto banale, ma abbiamo visto come posso collegare un evento della mia pagina ad un comando javascript.

Secondo passo

Possiamo eseguire anche funzioni al verificarsi dell'evento. Per farlo bisogna inserire il nome della funzione come valore dell'evento. Ad esempio :

```
<script>
        function saluta(){
          alert('ciao');
        }
</script>

<button onclick="saluta();"> Salutami </button>
```

Terzo passo

Invece di gestire l'evento all'interno del tag, posso gestire tutto all'interno dello script, facendo riferimento al tag tramite un id. Ad esempio :

```
<button id="miobottone"> Salutami </button>

<script>
document.getElementById("miobottone").onclick=function(){
        alert('ciao');
}
</script>
```

Quarto passo

Se voglio gestire il codice in modo più «pulito» posso anche scrivere :

```
<button id="miobottone"> Salutami </button>

<script>
        var mio = document.getElementById("miobottone");

        mio.onclick=function(){
                alert('ciao');
        }
</script>
```

Quarto passo… modificato

Se invece voglio ad esempio modificare lo stile di una pagina ?

```
<h2 id="papero">prova dom</h2>
   <button id="miobottone"> Salutami </button>
   <script>
       var mio = document.getElementById("miobottone");
       mio.onclick=function(){

        document.getElementById("papero").style.color = "red";

       }
    </script>
```

addEventListener

È un metodo per gestire più eventi in questo modo :

```
<h2 id="paperino">dom</h2>
<button id="miobottone"> Salutami </button>
<script>
        var mio = document.getElementById("miobottone");

        mio.addEventListener("mouseover", primafunzione);
        mio.addEventListener("click", secondafunzione);
        mio.addEventListener("mouseout", terzafunzione);

        function primafunzione(){
            document.getElementById("paperino").innerHTML ='ho generato un mouseover';
        }
        function secondafunzione(){
            document.getElementById("paperino").innerHTML ='ho generato un click';
        }
        function terzafunzione(){
            document.getElementById("paperino").innerHTML ='ho generato un mouseout';
        }
</script>
```

Esercizio 1

Scrivere un documento HTML contenente un form con i seguenti campi:

- cognome (casella di testo editabile lunga 40 caratteri)
- nome (casella di testo editabile lunga 30 caratteri)
- matricola (casella di testo editabile lunga 12 caratteri)
- regione di residenza (da scegliere da un menu che riporta le 20 regioni italiane)
- email (casella di testo editabile lunga 30 caratteri)
- telefono (casella di testo editabile lunga 15 caratteri)
- richieste particolari (area di testo editabile di 12 righe per 60 colonne)
- bottone di invio
- bottone di reset

Aggiungere al documento HTML una funzione JavaScript (non usare required) che esegue i seguenti controlli:

- verifica che il cognome non sia vuoto;
- verifica che il nome non sia vuoto;
- verifica che la matricola sia un numero;
- verifica che sia stata selezionata una regione;
- verifica che o l'email o il telefono siano non vuoti.

Inoltre, fare in modo che, nel documento HTML, tale funzione JavaScript venga eseguita quando l'utente invia la form