ajax

Asynchronous JavaScript and XML

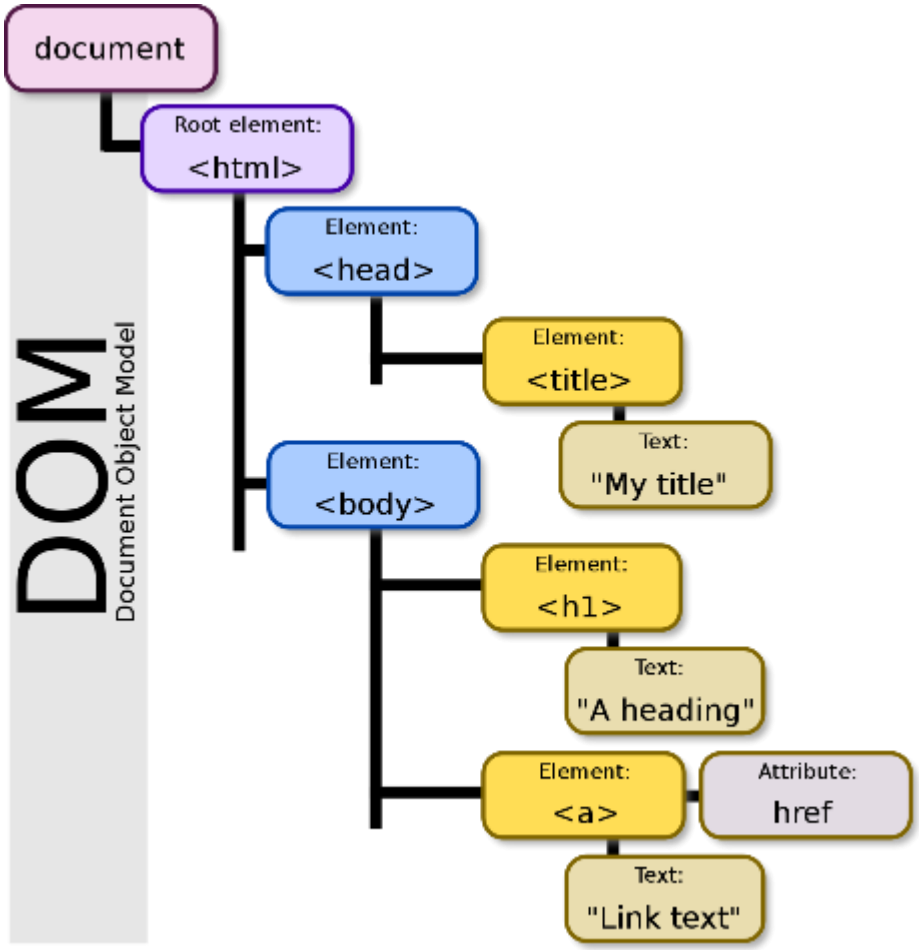
definizione

- Asynchronous JavaScript and XML (AJAX)
  - non è un linguaggio
  - non è una tecnologia
  - è un termine che descrive un "***nuovo***" approccio all'utilizzo di diverse tecnologie esistenti, compresi: HTML, CSS, JavaScript, DOM, XML, XSLT e l'oggetto ***XMLHttpRequest***
- con AJAX, le applicazioni web possono eseguire ***aggiornamenti rapidi e incrementali*** dell'interfaccia utente ***senza ricaricare*** nel browser l'intera pagina
- questo rende l'applicazione ***più performante*** e ***più reattiva*** alle azioni dell'utente

https://www.w3schools.com/js/js_ajax_intro.asp

DOM
(Document Object Model)

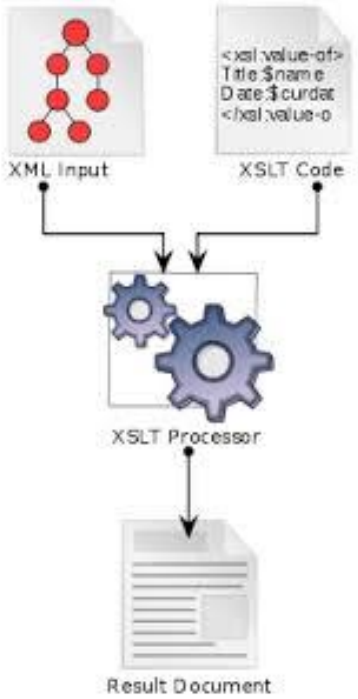
A. Ferrari



- **D**ocument **O**bject **M**odel
  - rappresentazione dei documenti strutturati attraverso una gerarchia di oggetti
- standard W3C per la rappresentazione di documenti in modo indipendente dalla piattaforma e dal linguaggio
- nel nostro caso interessa il document object che rappresenta la pagina html

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

- XML (eXtensible Markup Language) metalinguaggio per la definizione di linguaggi di markup
  - linguaggio di marcatura che consente di definire il significato degli elementi contenuti in un testo
- XSLT (e**X**tensible **S**tylesheet **L**anguage **T**ransformations)
  - un linguaggio di trasformazione dell'XML
  - nel nostro caso lo utilizziamo implicitamente per trasformazioni da XML a HTML



- AJAX è uno strumento per la realizzazione di **applicazioni web interattive** (Rich Internet Application)
- la tecnologia AJAX si basa su uno ***scambio di dati in background*** fra web browser e server, che consente l'***aggiornamento dinamico*** di una pagina web senza esplicito ricaricamento da parte dell'utente
- la richiesta è ***asincrona***: non si deve attendere che sia ultimata per effettuare altre operazioni
- normalmente le funzioni richiamate sono scritte in ***JavaScript***
- AJAX è una tecnica ***multi-piattaforma***

caratteristiche

- AJAX sfrutta l'oggetto ***XMLHttpRequest***
- offre la possibilità di interpretare e lavorare con i documenti XML
- reso popolare da Google, a partire dal 2005
  - utilizzato in Google Maps, Gmail, Amazon ...
- il vantaggio principale di AJAX è la grande ***velocità*** alla quale un'applicazione ***risponde*** agli input dell'utente

richiesta HTTP

- la richiesta HTTP al server utilizza un'istanza di una classe
  - questa classe è stata introdotta originariamente in Internet Explorer come oggetto ActiveX e si chiamava XMLHTTP
  - successivamente Mozilla, Safari e altri browser hanno implementato la classe ***XMLHttpRequest***, che supporta gli stessi metodi e le stesse proprietà della classe di Microsoft
- come fare:
  - `http_request = new XMLHttpRequest();`

definire come gestire la risposta inviata dal server

- è necessario comunicare all'oggetto XMLHttpRequest quale ***funzione*** JavaScript elaborerà il codice XML
- si assegna alla proprietà ***onreadystatechange*** dell'oggetto la funzione JavaScript
- **`http_request.onreadystatechange = nomeFunzione;`**
- in questo modo si sta assegnando un ***riferimento*** alla funzione, non la si sta ancora chiamando

inviare la richiesta al server

A. Ferrari

- per inviare la richiesta bisogna chiamare i metodi **`open()`** e **`send()`**

```
http_request.open('GET', 'http://www.nomeserver.org/qualsiasi.file', true);
http_request.send(null);
```

- ***open(-,-,-)***
  - parametro 1: ***metodo*** : GET, POST … (in lettere MAIUSCOLE)
  - parametro 2: ***URL*** (*non è possibile chiamare pagine che si trovino su un dominio differente da quello in cui si trova la pagina corrente*)
  - parametro 3: ***true*** per richiesta ***asincrona***, false sincrona
- ***send()***
  - il parametro è costituito dai dati che si vogliono inviare al server se la richiesta è di tipo POST

gestire la risposta del server

A. Ferrari

- la funzione deve controllare lo stato della richiesta
  - se lo stato ha un valore di 4, significa che la risposta è stata interamente ricevuta

```
if (http_request.readyState == 4) {
    // tutto ok, la risposta è stata ricevuta
} else {
    // non ancora pronto
}
```

- necessario controllare anche il codice di stato della risposta http: il codice ***200*** è ok

```
if (http_request.status == 200) {
    // bene!
} else {
    // problema nella richiesta
}
```

accedere ai dati ricevuti

A. Ferrari

- **`http_request.responseText`**
  - restituisce la risposta sotto forma di stringa di testo
- **`http_request.responseXML`**
  - restituisce la risposta sotto forma di oggetto XMLDocument che si può navigare tramite le funzioni DOM.

XMLHttpRequest

- **proprietà**
  - **onreadystatechange** definisce la funzione da richiamare
  - **readyState** stato della richiesta
    - **0** non inviata
    - **1** stabilita connessione con il server
    - **2** richiesta ricevuta dal server
    - **3** richiesta in fase di processo
    - **4** richiesta conclusa (risposta pronta)
  - **status** stato della richiesta
    - 200 ok
    - 404 file not found
  - **responseText** risposta in formato stringa
  - **responseXML** risposta in formato XML

XMLHttpRequest

- **metodi**
  - **new XMLHttpRequest()** costruttore
  - **abort()** cancella richiesta
  - **open(metodo,url,asincrono)** specifiche della richiesta
    - metodo GET/POST
    - url riferimento al file che contiene/elabora la risposta (.txt, .hml, .php)
    - asincrono true/false
  - **send()** invio richiesta

https://www.w3schools.com/xml/dom_http.asp

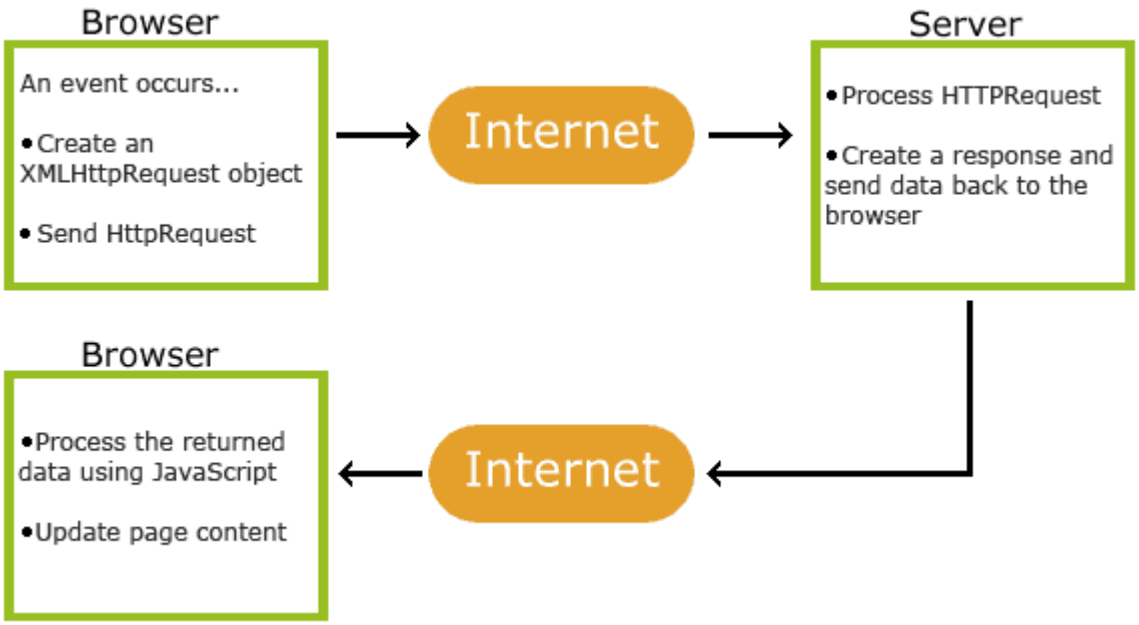
cosa fare con i dati ricevuti

A. Ferrari

- selezionare un elemento della pagina
  - **`getElementById(string)`**
  - è un metodo del DOM che permette di selezionare un elemento mediante il suo **id**
- modificare il valore dell'elemento
  - **`innerHTML`**
  - proprietà che permette di ottenere o modificare il testo e le caratteristiche di un nodo

client <> AJAX <> server

A. Ferrari



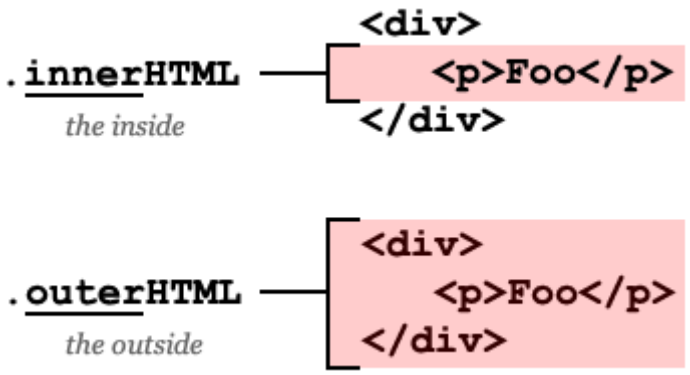
getElementById esempio d'uso

A. Ferrari

- *esempio*

```
function cambiaColore(nuovoColore)
{
 elem = document.getElementById("paragrafo1");
 elem.style.color = nuovoColore;
}
```

- la proprietà innerHTML permette di ottenere o modificare il testo e le caratteristiche di un nodo



innerHTML - esempio

A. Ferrari

- Cambiare dinamicamente un link

```
function cambiaLink()
{
 document.getElementById('mioLink').innerHTML="sito";
 document.getElementById(‘mioLink').href="http://www.sito.com";
 document.getElementById(mioLink').target="_blank";
}
```

esempi

- ***esempio 01***
  - visualizzazione di dati presenti sul server in un file di testo
- ***esempio 02***
  - analogo al precedente ma richiedendo un file XML
- ***esempio 03***
  - uso di innerHTML
- ***esempio 04***
  - tooltip