



**ajax**

Asynchronous JavaScript and XML

- 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](https://www.w3schools.com/js/js_ajax_intro.asp)

- 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*

- 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

- 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();`

- 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

- 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

- 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  
}
```

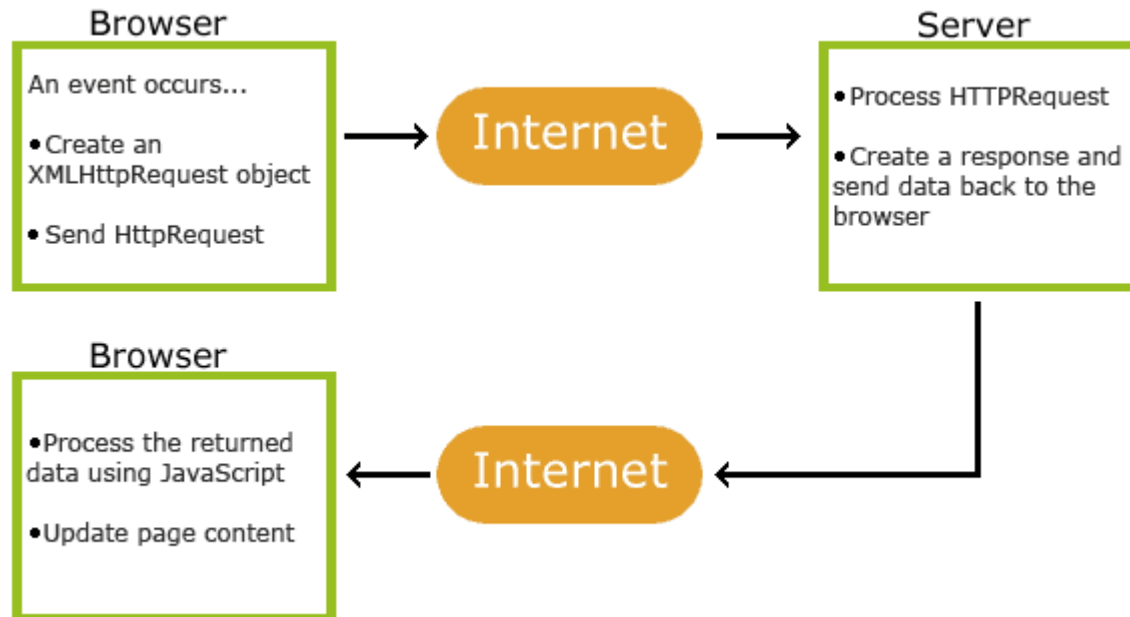
- necessario controllare anche il codice di stato della risposta http:  
il codice **200** è ok

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



- **`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.

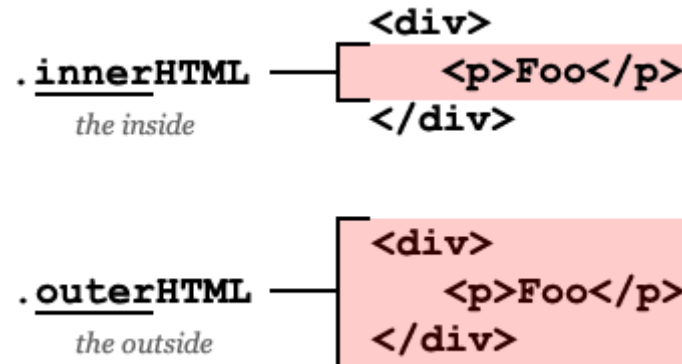
- selezionare un elemento della pagina
  - **getElementById(string)**
  - è un metodo del DOM che permette di selezionare un elemento mediante il suo **id**
- modificare il valore dell'elemento
  - **innerHTML**
  - proprietà che permette di ottenere o modificare il testo e le caratteristiche di un nodo



- *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



- Cambiare dinamicamente un link

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

- ***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