**Ajax e Chiamate Asincrone in JavaScript**

**Cos'è Ajax?**

Ajax (Asynchronous JavaScript and XML) è una tecnologia che permette di invocare un indirizzo remoto, simile a come avviene con un link o una form, ma attraverso JavaScript. La principale differenza rispetto alle richieste tradizionali è che le chiamate Ajax sono **asincrone**. Quando si invia una richiesta tramite Ajax, l'utente non è costretto ad aspettare il caricamento della pagina. Questo significa che l'utente può continuare ad interagire con la pagina mentre la richiesta HTTP viene inviata al server e la risposta arriva successivamente.

**Chiamata asincrona:**

In una richiesta tradizionale, quando si clicca su un link, si attende che il server risponda prima di poter interagire ulteriormente con la pagina. Al contrario, in una chiamata asincrona, la pagina non viene bloccata e l'utente può continuare ad interagire con essa mentre la richiesta viene elaborata in background.

**Programmazione Non Bloccante**

La **programmazione non bloccante** è un concetto fondamentale quando si parla di chiamate asincrone. In modo tradizionale, quando si invoca una funzione, il flusso del programma si blocca fino a quando quella funzione non termina la sua esecuzione. In una chiamata asincrona, invece, l'esecuzione del programma continua senza attendere la risposta della funzione invocata. Quando la funzione asincrona riceve una risposta, viene eseguita una callback function (una funzione di callback) che gestisce il risultato.

**Esempio pratico:** In JavaScript, quando invochiamo una funzione asincrona, non dobbiamo aspettare che finisca per proseguire con altre operazioni. Questo approccio è molto utile quando dobbiamo gestire operazioni lente, come la richiesta di dati a un server, senza bloccare l'interfaccia utente.

**JavaScript: Linguaggio Lato Client**

JavaScript è un linguaggio di programmazione **lato client** che si differenzia da Java, nonostante alcune somiglianze nella sintassi. In JavaScript, la programmazione è orientata principalmente alle **funzioni**, a differenza di Java che si basa più sulle classi. Le funzioni in JavaScript sono definite tramite la parola chiave function, seguita dal nome della funzione e dai parametri.

**Esempio di dichiarazione di funzione:**

|  |
| --- |
| function nomeFunzione(parametri) {   // corpo della funzione  }  ​ |

Le variabili in JavaScript si dichiarano utilizzando la parola chiave var. Il tipo della variabile viene determinato automaticamente dal valore che le viene assegnato. Ad esempio, se scriviamo:

|  |
| --- |
| var età = 10; // variabile di tipo numerico  ​  ​ |

La variabile età avrà tipo numerico. Se invece facciamo:

|  |
| --- |
| var nome = "Mario"; // variabile di tipo stringa  ​ |

nome sarà una stringa.

**Differenza con Java:** In JavaScript, possiamo modificare dinamicamente il tipo di dato di una variabile. Ad esempio, una variabile che inizialmente è un numero può essere successivamente trattata come una stringa o un altro tipo di dato. In Java, invece, il tipo di una variabile è fissato alla dichiarazione e non può cambiare.

**Utilizzo di XMLHttpRequest in JavaScript**

Per fare una chiamata asincrona in JavaScript, si utilizza l'oggetto **XMLHttpRequest**. Questo oggetto permette di inviare richieste HTTP al server e di ricevere risposte senza ricaricare la pagina.

**Come si usa XMLHttpRequest:**

1. Si crea un'istanza dell'oggetto XMLHttpRequest.
2. Si imposta una funzione di callback tramite il parametro onload, che verrà chiamata quando la risposta dal server sarà ricevuta.

Ecco un esempio base:

|  |
| --- |
| var xhr = **new** XMLHttpRequest();  xhr.open("GET", "url-del-server", true);  xhr.onload = function() {  **if** (xhr.status === 200) {     console.log(xhr.responseText);  }  };  xhr.send();  ​ |

**Filtri e Servlet in un'applicazione Web**

**Cosa sono i Filtri?**

I **filtri** sono componenti che vengono eseguiti prima delle servlet. La differenza principale tra filtri e servlet è che i filtri sono invocati **prima** della servlet e possono essere utilizzati per modificare la richiesta o la risposta, o addirittura per bloccare l'accesso a determinate risorse. Ad esempio, un filtro può essere usato per autenticare un utente, aggiungere delle informazioni alla richiesta, o addirittura bloccare l'accesso a una risorsa specifica.

**Configurazione dei Filtri:**

I filtri possono essere configurati nel file **web.xml** di un'applicazione web, ma possono anche essere configurati tramite **annotazioni** nelle classi Java. I filtri vengono invocati tramite l'URL dell'applicazione, proprio come le servlet.