AJAX con jQuery in Programmazione Web

6.1 - jQuery e i Framework JavaScript

jQuery è un framework JavaScript che si è affermato come uno degli strumenti più utilizzati nello sviluppo web. Come tutti i framework JavaScript, jQuery è stato sviluppato con l'obiettivo di fornire funzionalità di alto livello che, pur basandosi sull'ambiente JavaScript nativo, offrono capacità più sofisticate e complesse rispetto a quanto disponibile nell'ambiente base.

Il nome "jQuery" deriva dalla sua funzione primaria: interrogare (querying) il Document Object Model (DOM) con lo scopo di manipolarlo. Grazie a costrutti semplici ma potenti, jQuery riduce notevolmente lo sforzo richiesto al programmatore per interagire con il DOM e modificare dinamicamente il contenuto delle pagine web.

Nel corso del tempo, jQuery è stato esteso con funzionalità aggiuntive, tra cui una gestione robusta delle chiamate AJAX, molto più affidabile rispetto a quanto si possa ottenere con funzioni come `makeAjaxRequest` viste nelle lezioni precedenti.

6.2 - jQuery UI

Parallelamente a jQuery è stato sviluppato jQuery UI, un progetto che fornisce "gadget" o "widget" pronti all'uso nelle pagine web. Tra questi troviamo:

- Date picker, per inserire le date senza doverle digitare manualmente
- Progress bar, per indicare che il codice sta elaborando dati

Per maggiori informazioni sul progetto jQuery, il sito ufficiale è https://jquery.com/.

6.3 - Libreria jQuery

Per utilizzare jQuery, è necessario innanzitutto scaricare la libreria e includerla nella pagina HTML:

```
<script type="text/javascript" src="jquery-2.0.0.js"></script>
```

Questo tag script carica un file contenente il codice JavaScript della libreria jQuery.

L'Oggetto/Funzione $ o jQuery

Il cuore del framework jQuery è l'oggetto `jquery`, comunemente utilizzato attraverso l'alias `$`. Questa è una delle particolarità più interessanti di jQuery: sfrutta la doppia natura degli oggetti/funzioni in JavaScript, essendo contemporaneamente:

- Un oggetto che fornisce metodi (come quelli per AJAX)
- Una funzione che interroga il DOM

AJAX in jQuery

jQuery fornisce un supporto nativo per le chiamate AJAX, con diverse caratteristiche avanzate:

- Supporto diretto per le chiamate cross-domain
- Gestione di molteplici formati di risposta
- Controllo dettagliato della chiamata AJAX

L'Oggetto jqXHR

jQuery utilizza l'oggetto `jqXHR` che incorpora ed estende l'oggetto nativo `XMLHttpRequest`. Questo oggetto:

- Consente di controllare lo stato della comunicazione
- Memorizza i dati ricevuti
- Gestisce le funzioni di callback

I metodi principali dell'oggetto `jqXHR` sono:

- `done(funzione)`: specifica la funzione da chiamare in caso di successo della chiamata
- `fail(funzione)`: specifica la funzione da chiamare in caso di fallimento
- `always(funzione)`: specifica una funzione che sarà chiamata in ogni caso al termine della chiamata

Le interfacce delle funzioni di callback sono strutturate come segue:

```
// Interfaccia per done
jqXHR.done(function(data, textStatus, jqXHR) {
    // data: il dato ricevuto dalla risposta
    // textStatus: testo che indica lo stato finale della chiamata HTTP
    // jqXHR: l'oggetto che gestisce la sessione AJAX
});

// Interfaccia per fail
jqXHR.fail(function(jqXHR, textStatus, errorThrown) {
    // jqXHR: l'oggetto che gestisce la sessione AJAX
    // textStatus: testo che indica lo stato del processo
    // errorThrown: l'errore restituito
});

// Interfaccia per always
jqXHR.always(function(data, textStatus, jqXHR) {
    // Parametri identici a done
});
```

Una caratteristica interessante è che i metodi `done`, `fail` e `always` restituiscono ancora l'oggetto `jqXHR`, permettendo di creare catene di chiamate:

```
jqReq.done(...).fail(...).always(...);
```

Accesso all'Oggetto XMLHttpRequest

L'oggetto `jqXHR` incorpora l'oggetto JavaScript nativo `XMLHttpRequest` e fornisce metodi per accedervi. Generalmente non è necessario utilizzare direttamente questi metodi, ma sono disponibili per completezza:

- Proprietà: `readyState`, `status`, `statusText`, `responseXML`, `responseText`
- Metodi: `setRequestHeader(name, value)`, `getAllResponseHeaders()`, `getResponseHeader()`, `statusCode()`, `abort()`

Chiamata AJAX

L'oggetto `$` (jQuery) fornisce il metodo `ajax` che riceve come parametro un oggetto di configurazione contenente le impostazioni della chiamata. Questo metodo restituisce l'oggetto `jqXHR`.

L'oggetto di configurazione può contenere numerose opzioni. Il modo migliore per crearlo è utilizzare la notazione letterale degli oggetti JavaScript:

```
var ajaxConf = {
    url: "/AJAX_00/calculate_all.jsp?v1=" + n1 + "&v2=" + n2 + "&op=" + oper,
    dataType: "xml",
    type: "GET"
};
```

Le principali proprietà dell'oggetto di configurazione sono:

- `url` : l'URL (anche cross-domain) della richiesta AJAX
- `type` : il metodo della richiesta (GET, POST)
- `crossDomain` : indicatore per richieste cross-domain (default: false)
- `dataType` : il tipo di dato da ricevere (xml, html, json, jsonp, text)
- `data` : il contenuto della richiesta da inviare (oggetto serializzabile o stringa)
- `headers` : un oggetto con gli header della richiesta HTTP
- `mimeType` : stringa con il MIME type della richiesta
- `username` e `password` : credenziali per l'autenticazione (opzionali)

Una chiamata AJAX completa può essere scritta in questo modo:

```
// Definizione dell'oggetto di configurazione
var ajaxConf = {
    url: "/AJAX_00/calculate_all.jsp?v1=" + n1 + "&v2=" + n2 + "&op=" + oper,
    dataType: "xml",
    type: "GET"
};

// Funzione di successo
var success = function(data, textStatus, jqXHR) {
    risposta(data, "xml"); // Non tutti i parametri vengono utilizzati
};

// Funzione di fallimento
var failure = function(jqXHR, textStatus, errorThrown) {
    alert(errorThrown); // Non tutti i parametri vengono utilizzati
};

// Chiamata AJAX
$.ajax(ajaxConf)
    .done(success)
    .fail(failure);
```

Il Formato JSONP

JSONP (JSON with Padding) è un formato particolare in cui la risposta contiene una chiamata a una funzione il cui parametro è l'oggetto JSON.
Tecnica utilizzata per aggirare le restrizioni di sicurezza del same-origin policy, che impedisce a una pagina web di fare richieste AJAX a un dominio diverso da quello da cui è stata caricata. JSONP funziona sfruttando l'elemento `<script>` di HTML, che non è soggetto alla stessa restrizione delle richieste AJAX normali. Invece di eseguire una richiesta AJAX tradizionale, il browser carica un file di script scaricato dal server remoto che contiene una chiamata a una funzione locale.

Ad esempio:

```
Process({"result": 5})
```

La funzione chiamata (`Process` nell'esempio) deve essere presente nel codice JavaScript.
Tipicamente, nelle chiamate in modalità GET, la funzione con cui fare il "padding" viene specificata con il parametro `jsonp` . Data la pericolosita', JSONP non è più consigliato, e oggi si usa CORS (Cross-Origin Resource Sharing) per gestire le richieste cross-domain in modo sicuro.

Altri Metodi AJAX

Oltre al metodo `ajax` , jQuery offre metodi specializzati per casi d'uso specifici:

- `$.get` : per chiamate con metodo GET
- `$.getJSON` : per chiamate GET che ottengono dati JSON
- `$.getScript` : per chiamate GET che ricevono uno script
- `$.post` : per chiamate con metodo POST

6.4 - Accesso al DOM

Una delle funzionalità più potenti di jQuery è la capacità di selezionare e manipolare elementi del DOM in modo semplice ed efficace.

Selettori

Quando l'oggetto `$` viene utilizzato come funzione, il parametro contiene "selettori" dei nodi del DOM, tipicamente basati sui selettori CSS:

```
// Seleziona l'elemento con id="mioBlocco"
$('#mioBlocco')
```

Gli oggetti restituiti da queste selezioni sono estensioni del DOM proprie di jQuery, che forniscono metodi per modificarli:

```
// Modifica lo stile CSS dell'elemento
$('#mioBlocco').css("border", "2px solid green");
```

Anche le classi CSS possono essere utilizzate come selettori:

```
// Seleziona tutti gli elementi con class="miaClasse"
$('.miaClasse')
```

jQuery supporta selezioni più sofisticate:

- Tutti gli elementi in un blocco: `$('#mioBlocco *')`
- Tutti gli elementi di un certo tipo in un blocco: `$('#mioBlocco li')`
- Solo i figli diretti: `$('#mioBlocco > li')`

Creazione di Blocchi DOM

jQuery permette anche di creare nuovi blocchi DOM:

```
// Crea un nuovo blocco DOM
var blocco = $("<div><p>Esempio</p></div>");

// Aggiunge il blocco a un elemento esistente
blocco.appendTo(document.body);
// oppure
blocco.appendTo('#mioBlocco');

// In alternativa
$('body').append(blocco);
$('#mioBlocco').append(blocco);
```

Liste di Nodi

La funzione `$` (o jQuery) restituisce "liste di nodi". Sfruttando la programmazione a oggetti, si possono creare catene di azioni su queste liste utilizzando i metodi forniti.

Per conoscere la dimensione di una selezione:

```
$('#menu li').size()  // Deprecato in versioni recenti
$('#menu li').length  // Preferibile
```

Per ottenere la rappresentazione DOM base di JavaScript, invece di quella estesa di jQuery:

- `get()` : restituisce un NodeList del DOM
- `get(indice)` : fornisce l'elemento nella posizione indicata (DOM puro)
- `eq(indice)` : fornisce l'elemento nella posizione indicata (DOM jQuery)

Manipolazione degli Elementi

jQuery fornisce numerosi metodi per manipolare gli attributi e il contenuto degli elementi:

Per gli attributi:

- `.attr(nomeattr)` : restituisce il valore dell'attributo
- `.attr(nomeattr, valore)` : imposta il valore dell'attributo
- `.removeAttr(nomeattr)` : rimuove l'attributo

Per il contenuto testuale:

- `.text()` : restituisce il testo dell'elemento
- `.text(stringa)` : imposta il testo dell'elemento

Per il contenuto HTML:

- `.html()` : restituisce il contenuto HTML dell'elemento
- `.html(stringa)` : imposta il contenuto HTML dell'elemento

Per le classi di stile:

- `.hasClass(classe)` : verifica se l'elemento ha la classe indicata
- `.addClass(classe)` : aggiunge la classe di stile
- `.removeClass(classe)` : rimuove la classe di stile

Processare le Liste

Per elaborare ogni elemento di una selezione, jQuery fornisce il metodo `each()` :

```
$("#menu li").each(function(i, el) {
    var index = i;  // posizione nella lista (0, 1, 2, ecc.)
    var id = el.id;  // id dell'elemento (DOM puro)
    alert("(index,id)=(" + index + "," + id + ")");
});
```

Il parametro `el` rappresenta il nodo DOM puro. Per applicare i metodi di jQuery su questo elemento, è necessario avvolgerlo nuovamente con la funzione `$`:

```
$("#menu li").each(function(i, el) {
    var index = i;
    var id = $(el).attr("id");  // Utilizzo dei metodi jQuery
    alert("(index,id)=(" + index + "," + id + ")");
});
```

Gestione dei Campi dei Form

Per accedere e manipolare i campi dei form, jQuery offre il metodo `val()`:

- `.val()`: restituisce il valore del campo (singolo o array)
- `.val(valoresingolo)`: imposta il valore del campo
- `.val(array)`: imposta l'array di valori (per campi che ammettono valori multipli)

Eventi

jQuery semplifica la gestione degli eventi con il metodo `.bind(evento, funzione)`:

```
$("a").bind("click", function(event) {
    var target = event.target;  // nodo DOM puro su cui è avvenuto l'evento
    alert(target.tagName);  // nome del tag su cui abbiamo cliccato
});
```

L'oggetto `event` fornisce metodi e proprietà utili:

- `event.preventDefault()`: inibisce l'azione predefinita dell'evento
- `event.type`: tipo dell'evento (es. "click")
- `event.target`: l'elemento oggetto dell'evento (DOM puro)

jQuery fornisce anche una funzione di inizializzazione che viene eseguita quando il DOM è pronto:

```
$(function() {
    alert("DOM caricato!");
});
```

Questa funzione viene eseguita quando il DOM è pronto, ma prima che stili e immagini siano stati caricati completamente.

6.5 - jQuery UI

jQuery UI è un sottoprogetto di jQuery che fornisce widget predefiniti da inserire nelle pagine HTML. Questi widget sono accompagnati da codice JavaScript che manipola stili e DOM, permettendo ai programmatori di sfruttare funzionalità avanzate con poco sforzo.

Il sito web ufficiale di jQuery UI è www.jqueryui.com.

Tra i widget più utilizzati troviamo:

- Autocomplete
- Menu
- Progress Bar
- Spinner
- Tabs
- Date Picker

Utilizzo dei Widget

I widget hanno generalmente un'impostazione predefinita. In molti casi, è sufficiente attivarli sull'elemento DOM selezionato. Se necessario, è possibile personalizzarli attraverso un oggetto di configurazione.

Esempio di utilizzo del DatePicker:

```
$(function() {
    $('#mydate').datepicker();
});
```

Dove `mydate` è l'id di un campo di input di tipo text. Cliccando sul campo, il DatePicker apparirà.

Per personalizzare il formato della data:

```
$(function() {
    $('#mydate').datepicker({
        dateFormat: "dd-mm-yyyy"
    });
});
```

In questo caso, stiamo creando un oggetto di configurazione al volo per specificare il formato della data desiderato.