ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Framework Javascript

Prima parte

Fabio Vitali

Corsi di laurea in Informatica e
Informatica per il Management
Alma Mater – Università di Bologna

Oggi parleremo di…

Alcuni framework:

- JQuery

- *Il problema dei template*
- *Moustache e Handlebar*
- *Angular*
- *React*
- *Vue*

- *Node + Express*

I framework Ajax

Sono librerie Javascript che semplificano la vita nella creazione di applicazioni Ajax anche complesse.

Hanno tre scopi fondamentali

- ***Astrazione***: gestiscono le differenze tra un browser e l'altro e forniscono un modello di programmazione unico (o quasi) che funziona MOLTO PROBABILMENTE su tutti o molti browser.
- ***Struttura dell'applicazione***: forniscono un modello di progetto dell'applicazione omogeneo, indicando con esattezza come e dove fornire le caratteristiche individuali dell'applicazione
- ***Libreria di widget***: forniscono una (più o meno) ricca collezione di elementi di interfaccia liberamente assemblabili per creare velocemente interfacce sofisticate e modulari

Caratteristiche comuni (1/2)

Accesso al DOM

- Navigazione sull'albero del documento
- Selezione di elementi per nome, id, classe, path
- Modifiche dinamiche al contenuto degli elementi
- Modifiche dinamiche allo stile CSS degli elementi
  - Posizione
  - Comparsa / sparizione
  - Caratteristiche tipografiche

Comunicazioni asincrone con il server (Ajax)

- Gestione asincronicità e callback
- Gestione successo e errori
- Conversione da e per JSON

Caratteristiche comuni (2/2)

Gestione eventi

- Selezione dell'evento e associazione ad elementi arbitrari
- Bubbling degli eventi
- Ricca struttura dati dell'evento

Libreria di widget

- Layout complessivo della pagina
- Elementi dei form
- Animazioni
- Tabelle intelligenti e alberi
- Templating

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

jQuery

jQuery

http://www.jquery.com

In assoluto il framework Ajax più usato al mondo. Nel 2017 l'89% del milione di siti web con maggiore traffico include jQuery nelle sue pagine.

Nato nel 2006, ha una licenza open source molto aperta. E' molto leggero (78 Kb) e veloce.

Fornisce supporto per la navigazione e la modifica del DOM, gestione di eventi e l'uso di comunicazioni asincrone con il server (Ajax).

Estendibile attraverso la creazione di plug-in.

Non ha librerie di widget proprie, ma il progetto parallelo jQuery UI fornisce una prima libreria di widget (con l'aggiunta di altre librerie)

jQuery: l'oggetto $

jQuery ha una peculiarità sintattica, l'uso della keyword $ per ogni comando della libreria:

```
$("a").click(function() { alert("Hello world!"); })
```

Il carattere $ in Javascript è usabile negli identificatori.

jQuery definisce allora una funzione $() che è un alias del costruttore jQuery(), e ha come parametro un selettore DOM.

Il codice jQuery è immediatamente identificabile come tale grazie a questo ausilio sintattico.

jQuery: usare la libreria (1)

```
<html>
  <head>
    <script type="text/javascript" src="jquery.js"></script>
    <script type="text/javascript”>
        // codice personale
    </script>
  </head>
  <body>
    <p> parte HTML </p>
  </body>
</html>
```

jQuery: usare la libreria (2)

```
<html>
  <head>
    <script type="text/javascript" src="jquery.js"></script>
    <script type="text/javascript” src="mycode.js"></script>
  </head>
  <body>
    <p> parte HTML </p>
  </body>
</html>
```

Quando vengono eseguiti gli script?

Immediatamente!

- Tutti gli script vengono eseguiti subito dopo il loro caricamento.
  - Ad esempio, gli script stanno tipicamente dentro all'head di HTML, che viene caricato prima del body e molto prima di tutte le risorse esterne come le immagini.
- Ci sono due momenti specifici a partire dai quali si possono attivare gli script:
  - document.ready: il DOM è caricato, pronto e inizializzato. Eventuali risorse esterne (come le immagini) possono ancora non essere pronte, ma il loro posto nel DOM è già riservato.
  - window.load: il DOM è caricato e pronto, e tutte le risorse esterne (come le immagini sono pronte e caricate.
- Per la maggior parte delle situazioni, lo script può partire al ready. Se avete bisogno di informazioni aggiuntive non presenti nel DOM (ad esempio, la dimensione in byte di un'immagine) allora bisogna aspettare il load.

jQuery, eventi ready e load

- jQuery mette a disposizione due callback associate agli eventi di ready e load.
  - ```
    $(document).ready(function(){
      // codice da eseguire al ready
    })
    ```
  - ```
    $(window).load(function(){
      // codice da eseguire al load
    })
    ```

Scorciatoia

```
$(function() {
    // codice del ready
}
```

- Tipicamente si mette tutto il codice dentro a `$(document).ready`.
- N.B.: E' equivalente scrivere:
  - `$(document).ready( function() { … } )`
  - ```
    $(document).ready = inizio;
    …
    function inizio() { … }
    ```
    ```

jQuery: accedere al DOM

Il parametro del costruttore ***$()*** è un selettore jQuery. Sono selettori jQuery tutti i selettori CSS 3, più alcuni specifici di jQuery.

- ***$('p')*** restituisce un array di tutti i p del documento
- ***$('.pippo')*** restituisce un array di tutti gli elementi di classe 'pippo'
- ***$('#pluto')*** restituisce un array che contiene l'elemento di id='pluto'.
- ***$(':selected')*** restituisce un array di tutti gli elementi *selected*
- ***$(':visible')*** restituisce un array di tutti gli elementi visibili
- ***$('p[id^="basic"]')*** restituisce un array di tutti i p il cui id ***inizia con*** la parola 'basic'
- …

Ad esempio:

```
$("a").click(function() {
  alert("Hello world!");
});
```

jQuery: modificare il DOM

jQuery mette a disposizione molte funzioni di modifica del DOM.

Esse agiscono o su tutto l'array identificato dal selettore, o sul primo elemento di questo array

- ***$("p").addClass("importante"), $("p").removeClass("importante")***: aggiunge o toglie la classe CSS a ogni elemento dell'array
- ***$('h2').before(' Capitolo '), $('h2').after('<hr/>')***: aggiunge contenuto prima o dopo ogni elemento dell'array.
- ***$('h2').css("background-color"), $('h2').css("background-color","#FF0000")***: restituisce il valore della proprietà CSS specificata del primo elemento dell'array selezionato/ cambia il valore della proprietà CSS specificata in ogni elemento dell'array selezionato.
- ***$('h2').html(), $('h2').text()***: restituisce il codice HTML o il testo contenuto in ogni elemento dell'array selezionato.
- ***$('h2').html('Un titolo'), $('h2').text('Un nuovo titolo')***: sostituisce il codice HTML o il testo contenuto in ogni elemento dell'array selezionato.

jQuery: effetti e animazioni

E’ possibile attivare effetti e animazioni agli elementi del documento. Questi effetti vengono eseguiti subito o dopo un delay.

Per attivare gli effetti in seguito ad un evento (ad esempio un click su un bottone) è necessario specificare l’animazione dentro all’handler dell’evento desiderato.

- ***$(’#target').hide(); $(’#target').show()***; nasconde e mostra l’elemento con id=‘target’, assegnando la proprietà CSS specifica (che è diversa nei vari browser).
- ***$(’img.mostra').fadeIn('slow’); $(’img.nascondi').fadeOut('slow’)***: fa lentamente apparire o sparire ogni elemento dell’array selezionato.
- ***$(’img.mostra').slideDown('slow’); $(’img.nascondi').slideUp('slow’)***: fa lentamente apparire, con movimento dall’alto al basso o dal basso in alto, ogni elemento dell’array selezionato.

jQuery: eventi (1/2)

jQuery permette di specificare una funzione di callback da associare agli eventi del DOM.

- ```
  $('input').mouseover(function() {
      alert('mouseover');
  })
  ```
- ```
  $('input').keypress(function(e) {
      alert(e.which);
  })
  ```
- ```
  $('a').click( function(e) {
    document.href="http://www.google.com";
    e.preventDefault() ;
  })
  ```

e.preventDefault() blocca l'esecuzione del comportamento di default (in questo caso, la navigazione verso il contenuto originario dell'href).
  ```

jQuery: eventi (2/2)

In realtà, click(), mouseover(), ecc. derivano da un solo metodo:

`$('input').mouseover(f1)` è equivalente a `$('input').on('mouseover',f1)`

`$('a').click(f3)` è equivalente a `$('a').on('click',f3)`

Questo porta ad un'immediata e comodissima estensione:

`$('body').on('click', 'a', f3)` estende `$('a').on('click',f3)`

- Il secondo parametro, facoltativo, è un secondo selettore CSS, che indica a quali discendenti del selettore primario associare la callback.
- In entrambi i casi, la callback si applica agli elementi <a>, e non a <body>.
- Normalmente, il binding della callback all'evento avviene solo su elementi già esistenti nel DOM. Quindi `$('a').on('click',f3)` si applica solo agli elementi del DOM presenti, ad esempio, al document.ready.
- Se voglio associare un evento a elementi che potrebbero essere creati in seguito al binding, debbo delegare ad un elemento sempre esistente (ad esempio, <body>) di associare il callback all'elemento generato dinamicamente.

jQuery: Ajax

Una singola funzione si occupa di tutta la comunicazione asincrona usando XMLHttpRequest (XHR):

```
$.ajax({
    url: 'app.php',
    success: function(data) {
            $('.result').html(data);
            alert('Caricamento effettuato');
    },
    error: function(data) {
            alert('Caricamento impossibile');
    }
})
```

Il parametro unico della funzione è un oggetto Javascript di cui i membri url e success sono obbligatori. Alcune funzioni equivalenti ($.get(), $.post() e $.put()) sono disponibili per maggiore rapidità.

Le funzioni success() e error() vengono chiamate appena la connessione HTTP si è conclusa e dunque readystate = 4 (***loaded***).

jQuery: asincronia e promesse

Una alternativa al call-back è usare le promesse.

Il metodo $.ajax() è già una promessa, per cui si possono creare catene di funzioni then() con due parametri, la funzione da chiamare in caso di successo e quella in caso di insuccesso:

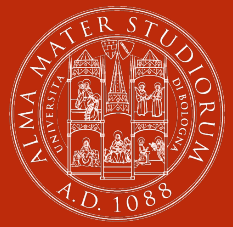
```
var good = function(data) {
    $('.result').html(data);
     alert('Caricamento effettuato');
};
var bad = function(data) {
     alert('Caricamento impossibile');
};
$.get('http://www.server.it/server').then(good, bad);
```

jQuery: conclusioni

jQuery si comporta in maniera molto conservativa nei confronti del documento HTML di partenza, aggiungendo dinamicamente comportamenti e stili ad elementi che rimangono quelli originali.

La semplicità d'uso e la leggerezza della libreria la rendono ideale per piccoli compiti di animazione e attivazione di elementi in pagine che sono già ben strutturate e graficamente complete.

jQuery non è fatto per progettare interfacce, ma per descrivere comportamenti dinamici su alcuni elementi particolari di pagine web tradizionali (form, link, menu, ecc.).



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fabio Vitali

Dipartimento di Informatica – Scienze e Ingegneria
Alma mater – Università di Bologna

Fabio.vitali@unibo.it

www.unibo.it