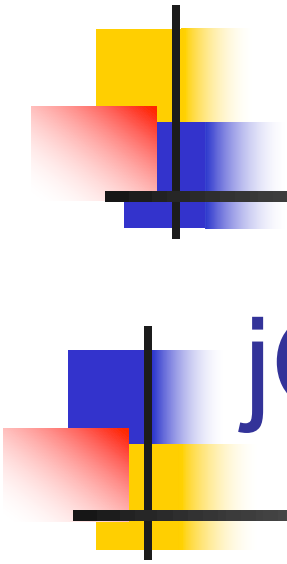
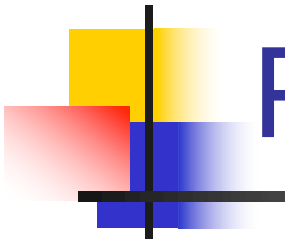


jQuery

Introduzione

Cos'è jQuery

- jQuery core: libreria JavaScript nella quale sono definiti operatori per:
  - La selezione di parti di un documento;
  - La manipolazione degli elementi del DOM;
  - La gestione degli eventi;
  - La definizione di eventi visuali sugli elementi del documento;
  - La gestione dell'interazione con il server tramite AJAX
- jQuery UI: libreria di componenti scritta in jQuery core articolata in categorie (che non vedremo) tra le quali:
  - Effetti (in aggiunta a quelli del core)
  - Interazioni (drag & drop, sorting, …)
  - Widgets (progress bar, sliders, dialog box, …)



Perché jQuery

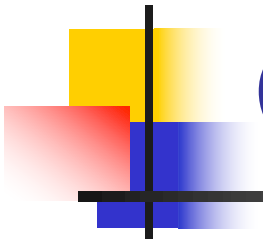
“Write less. Do more”

- Senza jQuery

```
var table = document.getElementsByTagName('table');
  for(var t = 0; t < table.length; t++){
    var rows = table[t].getElementsByTagName('tr');
    for (var i = 1; i < rows.length; i +=2) {
        rows[i].className += 'striped';
    }
}
```

- Con jQuery

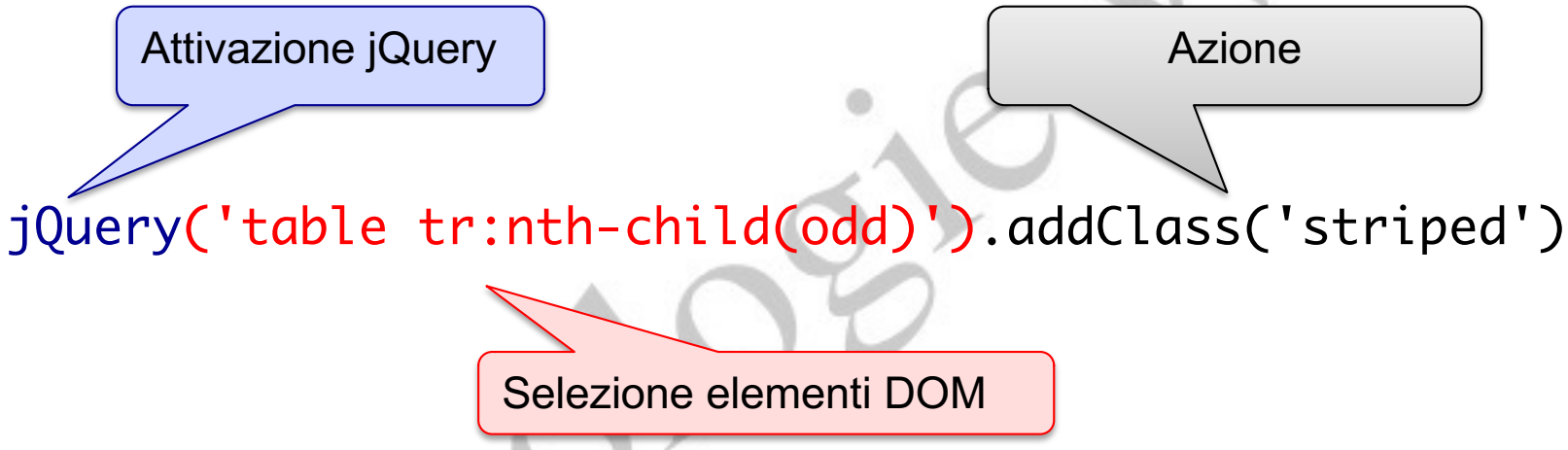
```
$('table tr:nth-child(odd)').addClass('striped');
```



Caratteristiche

- Primo rilascio: 2006.
- Piccole dimensioni
  - ~30Kb nella versione minima compressa
- Ricca di selettori HTML DOM veloci e di facile uso
- Compatibile con tutti i browser principali
  - Elimina problemi di codifica cross-browser
- È open-source
- È ben documentata
- È estensibile
  - Continuamente arricchita di componenti (plugin)
- È integrabile con altre librerie analoghe (Prototype, MooTools)
- È una delle librerie più utilizzate JavaScript

Uso di jQuery



- La chiamata alla funzione jQuery ritorna un oggetto JavaScript (wrapped set) che contiene:
  - un array con gli elementi del DOM selezionati, nell'ordine in cui compaiono nel documento;
  - una serie di metodi predefiniti che operano sull'oggetto

Uso di jQuery

- Notazione alternativa per la chiamata: jQuery --> $

```
$('table tr:nth-child(even)').addClass('striped')
```

- "Method chaining": la maggior parte dei metodi jQuery (wrapped methods) ritorna come risultato gli stessi elementi ricevuti in input. Questo permette l'attivazione in cascata di azioni sugli elementi selezionati

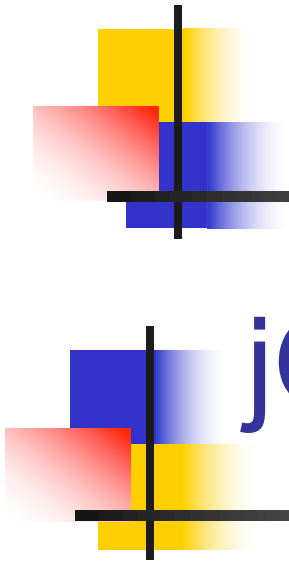
```
$('table tr:nth-child(even)').hide().addClass('removed')
```

- Attivazione di jQuery: al termine del caricamento del DOM

```
$(document).ready(function(){<Codice jQuery>})
```

alternativa

```
$(function(){<Codice jQuery>})
```



jQuery

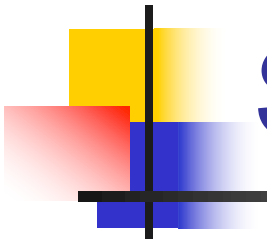
Selettori

Selettori CSS (lista parziale)

Selettore	Descrizione
*	Tutti gli elementi
E	Elementi con tag E
E F	Elementi con tag F discendenti di elemento con tag E
E>F	Elementi con tag F figli diretti di elemento con tag E
E.F	Elementi con tag E e classe F
E#I	Elementi con tag E e id I
E[A]	Elementi con tag E e attributo A (senza condizioni sul valore)
E[A=V]	Elementi con tag E e attributo A di valore V
E[A^=V]	Elementi con tag E e attributo A con valore che inizia con V
E[A$=V]	Elementi con tag E e attributo A con valore che termina con V
E[A!=V]	Elementi con tag E e attributo A di valore diverso da V o senza A
E[A*=V]	Elementi con tag E e attributo A con valore che contiene V

Selettori CSS – Esempi

Esempio	Selezione
`$('div, span')`	Gli elementi `'div'` o `'spawn'` del documento
`$('ul li img')`	Le immagini negli item delle liste non ordinate
`$('ul>li img')`	Le immagini negli item di primo livello delle liste non ordinate
`$('ul.miaClasse')`	Le liste non ordinate con classe `miaClasse`
`$('#intestazione')`	Gli elementi con `id='intestazione'`
`$('form[method]')`	Le form con l'attributo `method` definito
`$('input[type=checkbox][checked]')`	Gli elementi input di tipo `checkbox` selezionati
`$('img[src^='dog']')`	Le immagini con valore di `src` che inizia per `'dog'`
`$('img[src$='dog']')`	Le immagini con valore di `src` che finisce per `'dog'`
`$('img[src!='dog']')`	Le immagini con valore di `src` diverso da `'dog'`
`$('img[src*='dog']')`	Le immagini con valore di `src` che contiene `'dog'`



Selettori del Wrapped-Set

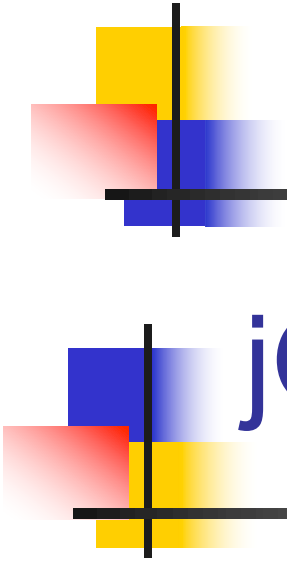
Selettore	Descrizione
`:parent`	Elementi che hanno almeno un figlio
`:empty`	Elementi che non hanno figli
`:first-child`	Elementi che sono primi figli del loro genitore
`:last-child`	Elementi che sono ultimi figli del loro genitore
`:only-child`	Elementi che non hanno fratelli
`:nth-child(n)`	Figli *n-esimi* del loro genitore
`:contains('text')`	Elementi che contengono 'text'
…	…

Selettori del WS – Esempi

Esempio	Selezione
`$('td:parent')`	Gli elementi *td* che hanno un contenuto (un testo o altre TAG HTML)
`$('td:empty')`	Gli elementi *td* che non hanno un contenuto (*<td></td>*)
`$('td:first-child)`	Gli elementi *td* che compaiono per primi all'interno dei *td* della pagina
`$('li:last-child')`	Gli ultimi elementi delle liste della pagina
`$('li:only-child')`	Gli elementi delle liste composte di un solo elemento
`$('li:nth-child(even)')`	Gli elementi di ordine pari delle liste
`$('p:contains(Bob)')`	I paragrafi che contengono la parola 'Bob'
…	…

Selettori generici

Selettore	Descrizione
`:button`	Elementi di tipo `button`, `submit`, `reset`
`:input`	Elementi di una form (`input`, `select`, `textarea`, `button`)
`:hidden`	Elementi `hidden`
`:header`	Elementi h1…h6
`:not(selector)`	Elementi non corrispondenti al `selector`
`:selected`	Elementi `option` selezionati
`:submit`	Elementi `<input type="submit">` o `<button type="submit">`
`:visible`	Elementi visibili
…	…



jQuery

Metodi Core

Manipolazione del Wrapped Set

get()	
`get(index)`	Ritorna l'elemento di posizione `index` nel set a cui è applicato. Se `index` è omesso, il risultato è l'intero set
`- var imgElement = $('img[alt]').get(0)`	

toArray()	
`toArray()`	Crea un array con tutti gli elementi del set a cui è applicato
`- $('div').toArray()`	

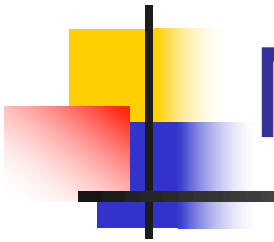
Manipolazione del Wrapped Set

not()	
`not(expression)`	Crea una copia del wrapped set a cui è applicato ed elimina da essa tutti gli elementi che corrispondono ai criteri definiti da `expression`

```
- $('img').not("[alt='eliminami']")
- $('img').not(function(){return !$(this).hasClass('tienimi');})
```

each()	
`each(iterator)`	Scandisce tutti gli elementi del wrapped set a cui è applicato ed attiva su ognuno la funzione `iterator`

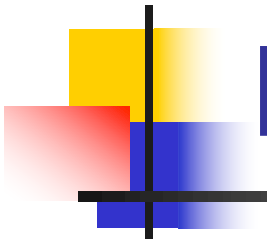
```
- $('img').each(function(n){
   this.alt="Questa è l'immagine n. " + n + " con id=" + this.id
  });
```



Manipolazione degli Attributi

attr()	
`attr(name)`	Ritorna il valore dell'attributo `name` del primo elemento del set a cui è applicato o `undefined` se l'attributo è assente
`attr(name,value)`	Assegna il valore `value` all'attributo `name` di tutti gli elementi set a cui è applicato
- `$('#myImage').attr('src')` - `$(':input').attr('title', 'Inserisci un valore')`	

removeAttr()	
`removeAttr(name)`	Rimuove l'attributo `name` da tutti gli elementi del set a cui è applicato
- `$(':input').removeAttr('target')`	



Manipolazione delle Classi

addClass()	
`addClass(names)`	Aggiunge la classe (o le classi) `names` agli elementi del set a cui è applicato. `names` può essere una funzione
`- $('img').addClass('mediumSize','highlight')`	

removeClass()	
`removeClass(names)`	Rimuove la classe (o le classi) `names` agli elementi del set a cui è applicato. `names` può essere una funzione
`- $('img').removeClass('mediumSize')`	

Manipolazione delle Classi

toggleClass()	
`toggleClass(names)`	Aggiunge la classe (o le classi) `names` agli elementi del set a cui è applicato che non la posseggono e la rimuove da quelli che la posseggono. `names` può essere una funzione
`- $('img').toggleClass('mediumSize')`	

hasClass()	
`hasClass(name)`	Ritorna `true` se tutti gli elementi del set a cui è applicato posseggono la classe `name`, `false` altrimenti
`- $('img').hasClass('mediumSize')`	

Manipolazione degli Stili

css()	
`css(name,value)`	Assegna alla proprietà di stile `name` il valore `value` per tutti gli elementi del set a cui è applicato
`css(name)`	Ritorna il valore `value` della proprietà di stile `name` del primo elemento del set a cui è applicato
`css(properties)`	Assegna le proprietà `properties` a tutti gli elementi del set a cui è applicato

```
-$('div.expandable').css('width','500')
-$('div.expandable').css('width')
-$('div.expandable').css({
                      width:'500',
                      height:'300',
                      backgroundColor: 'white'
                     })
```

Manipolazione dei Contenuti

html()	
`html()`	Estrae il codice HTML contenuto nel primo elemento del set a cui è applicato
`html(content)`	Assegna il codice HTML in `content` a tutti gli elementi del set a cui è applicato

```
<ul id="lista">
  <li>Uno</li>
  <li>Due</li>
  <li>Tre</li>
</ul>

-$('#lista').html() --> '<li>Uno</li><li>Due</li><li>Tre</li>

                                                <ul id="lista”>
-$('#lista').html('<li>Quattro</li>) -->          <li>Quattro</li>
                                                </ul>
```

Manipolazione dei Contenuti

text()	
`text()`	Estrae il testo contenuto negli elementi del set a cui è applicato
`text(content)`	Assegna il testo contenuto in `content` a tutti gli elementi del set a cui è applicato. I caratteri < > & vengono convertiti in codici UNICODE

```
<ul id="lista">
  <li>Uno</li>
  <li>Due</li>
  <li>Tre</li>
</ul>

-$('#lista').text() --> 'UnoDueTre'

                                          <ul id="lista">
-$('#lista').text('<li>Quattro</li> ') -->    &lt;li&gt;Quattro&lt;/li&gt;
                                          </ul>
```

Manipolazione dei Contenuti

append()	
`append(content)`	Opera come `html()`, con la differenza che il codice html in `content` viene aggiunto a quello degli elementi del set a cui il metodo è applicato. Il parametro può essere anche una funzione, alla quale vengono passati l'indice e il contenuto precedente di ogni elemento del set
- `$('p').append('<b>un qualsiasi testo</b>')` - `$('p.appedToMe').append($('a.appendMe'))` – equivale ad una move - `$('p.appedToMe').append(someElement)`	

- Metodi analoghi:
  - `prepend(content)`, `before(content)`, `after(content)`

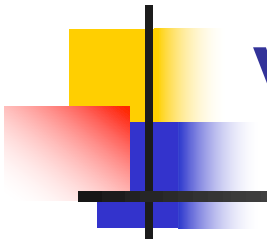
Manipolazione dei Contenuti

remove()	
`remove(selector)`	Rimuove gli elementi del set a cui il metodo è applicato. `selector` definisce un ulteriore selettore di un sottoinsieme del set da eliminare

```
- $('p').remove()
- $('p').remove(.miaClass)
```

replaceWith()	
`replaceWith(content)`	Sostituisce gli elementi del set a cui il metodo è applicato con `content`.

```
- $('img[alt]').each(function() {
    $(this).replaceWith('<div>' + $(this).attr('alt') + '</div>')
  });
```



Valori delle Form

val()	
`val()`	Ritorna il valore dell'attributo `value` del primo elemento del set a cui è applicato. Se è un elemento `select`, ritorna il valore dell'elemento selezionato, o un array di valori per `select` multipli.
`val(valore)`	Assegna `valore` all'attributo `value` di tutti gli elementi del set a cui è applicato. `value` può essere una funzione
`val(values)`	Marca come selezionati tutti gli elementi `checkbox`, `radio` e `option` del set a cui è applicato se il loro valore corrisponde ad uno degli valori dell'array `values`

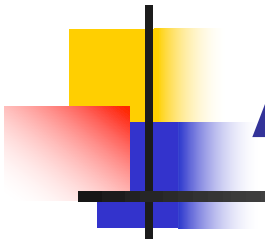
`- $('[name="radioGroup"]:checked').val()`
`- $('[name="radioGroup"]:checked').val('set')`
`- $('input,select').val(['uno','due','tre'])`	

Animazioni

show()	
`show(speed, callback)`	Rende visibile ogni elemento hidden del set a cui il metodo è applicato. `speed` definisce la velocità dell'effetto (in ms. o `slow`, `normal`, `fast`) e `callback` la funzione da attivare al termine dell'animazione

hide()	
`hide(speed, callback)`	Opera in maniera analoga a `show()`, rendendo invisibili gli elementi a cui è applicato

toggle()	
`toggle(speed, callback)`	Attiva il metodo `show()` sugli elementi hidden e quello `hide()` sugli elementi non-hidden del wrapped set



Animazioni

fadeIn()	
`fadeIn(speed, callback)`	Opera in maniera analoga a `show()`, usando un effetto di dissolvenza

fadeOut()	
`fadeOut(speed, callback)`	Opera in maniera analoga a `hide()`, usando un effetto di dissolvenza

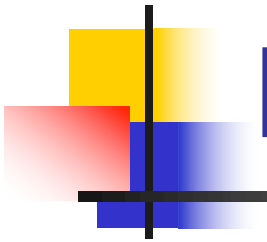
jQuery

Gestione degli Eventi

Eventi

- Gli eventi sono azioni generate dall'utente o dal browser durante la visualizzazione di un documento
- Modello operativo del Browser

- La reazioni agli eventi viene gestita dagli event handlers:
  - Definiti nel Browser (attivazione di ancore, submit di form);
  - Definiti da programmi (JavaScript) e codificati nei documenti.



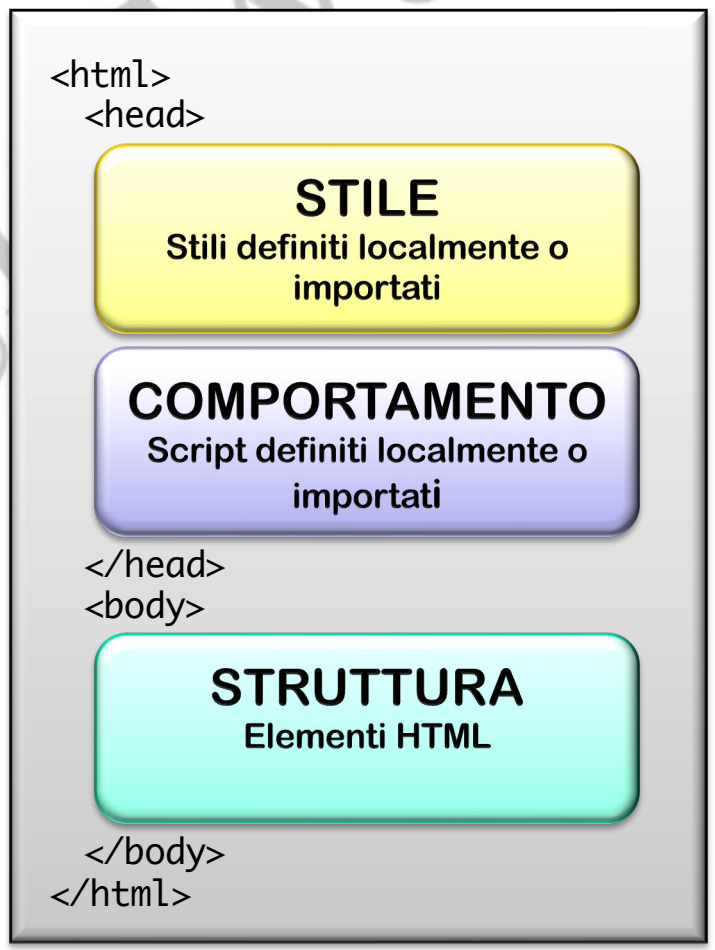
Eventi

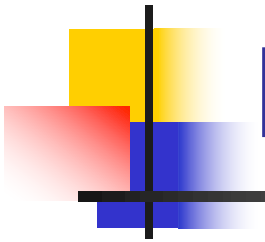
- Al verificarsi di un evento:
  - Un oggetto con le informazioni ad esso relative (elemento del documento che l'ha generato, tasto premuto …) viene passato a tutti gli handler definiti per l'evento, attivati in una sequenza che rispecchia la gerarchia di definizione (event bubbling)
    - Vengono attivati tutti gli handler dell'evento definiti negli elementi del DOM che contengono quello in cui l'evento si è verificato
- Problema: ogni browser ha un suo modello degli eventi, che determina modalità diverse di gestione da parte degli event handlers
- Soluzione: usare una libreria che fornisce funzionalità cross-browser per la definizione degli event handlers.

JavaScript Non-Intrusivo

- Problema: separare la definizione di un elemento della pagina HTML dal codice JS che ne descrive il comportamento
  - Analogia con HTML – CSS
- Soluzione: localizzare tutto il codice JS nella sezione HEAD del documento, selezionando dinamicamente gli elementi HTML ai quali applicarlo

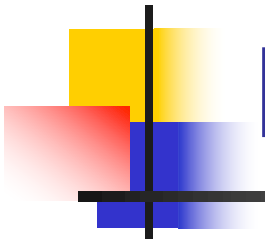
jQuery jQ1-jQ3





Definizione di Event Handler

on()	
`on(event,selector,data,handler)`	Definisce la funzione di data handling dell'evento indicato per tutti gli elementi del set selezionato
`event`	Nome dell'evento a cui associare l'handler (`blur, change, click, focus, ...`)
`selector`	Parametro opzionale che seleziona gli elementi figli (nel DOM) di quelli a cui il metodo **on()** è applicato che attivano l'handler
`data`	Parametro opzionale passato all'event handler attraverso la proprietà `data` dell'oggetto `Event`
`handler`	Funzione che implementa l'event handler. L'istanza dell'evento che la attiva viene passato come parametro alla funzione



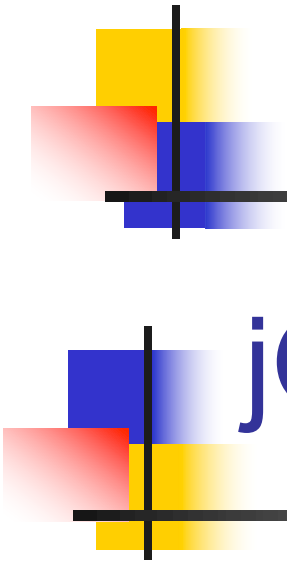
Definizione di Event Handler

hover()	
`hover(enterHandler, leaveHandler)`	Definisce le azioni da attivare al verificarsi degli eventi `mouseenter` e `mouseleave` sugli elementi a cui è applicato

- Definizione sintetica di event handlers (non permette il passaggio del parametro `data`):
  - `blur(handler)`, `change(handler)`, `click(handler)`, `focus(handler)` …
- Proprietà dell'oggetto `event`:
  - `altKey`, `ctrlKey`, `currentTarget`, `data`, `metaKey`, `pageX`, `pageY`, `relatedTarget`, `screenX`, `screenY`, `shiftKey` …

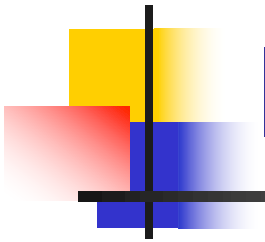
Esempi

- DOM Livello 0 — jQ4
- DOM Livello 2 — jQ5
- Modifica degli attributi degli elementi — jQ6
- Show/Hide di un elemento — jQ7
- Uso metodo slideToggle() — jQ8
- Uso metodo hover() — jQ9
- Validazione dati Form — jQ10



jQuery

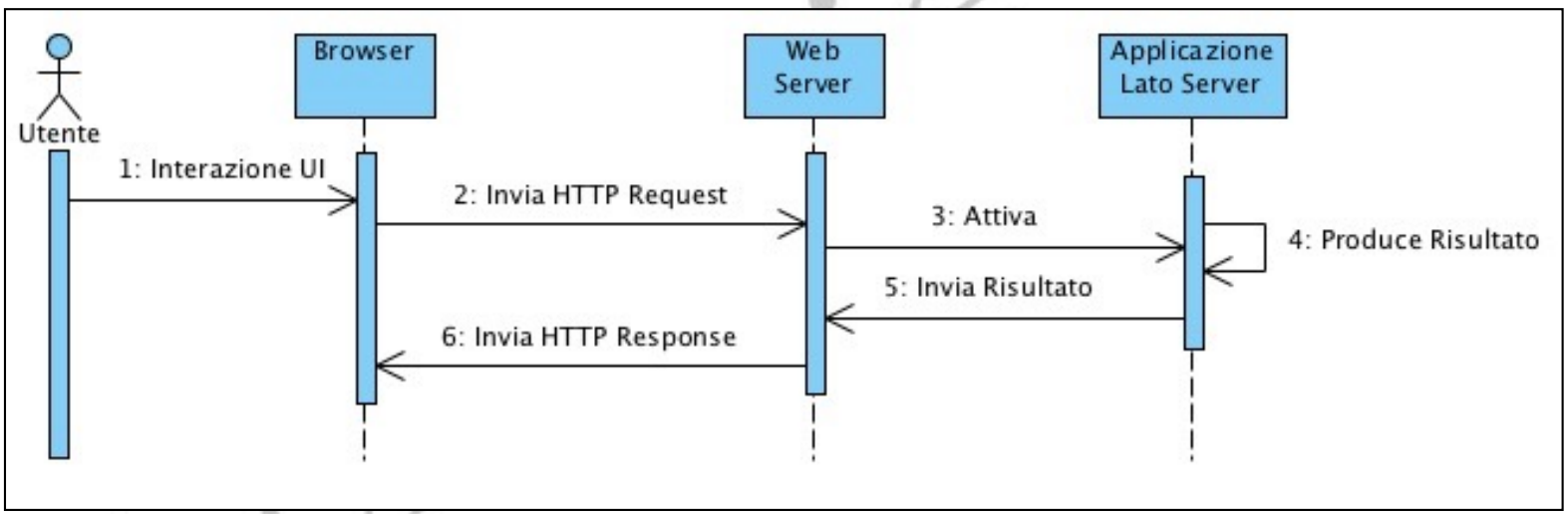
JQuery ed AJAX

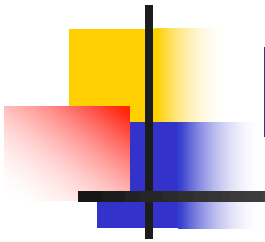


Interazione Client-Server

- JavaScript consente di modificare il contenuto di un documento visualizzato dal browser tramite manipolazione del DOM
- Possiamo "iniettare" in un documento informazioni provenienti dal server a seguito di una richiesta (GET o POST) fatta dal client?
  - NO, usando il protocollo HTTP: ad ogni richiesta del client corrisponde un processo di acquisizione di un intero documento HTML che viene visualizzato al posto del precedente
- Il protocollo HTTP è sincrono: ad ogni richiesta il client aspetta che il server risponda per poi visualizzare la risposta
  - Nell'attesa, nessun evento viene rilevato (e gestito) dal browser

HTTP Sequence Diagram

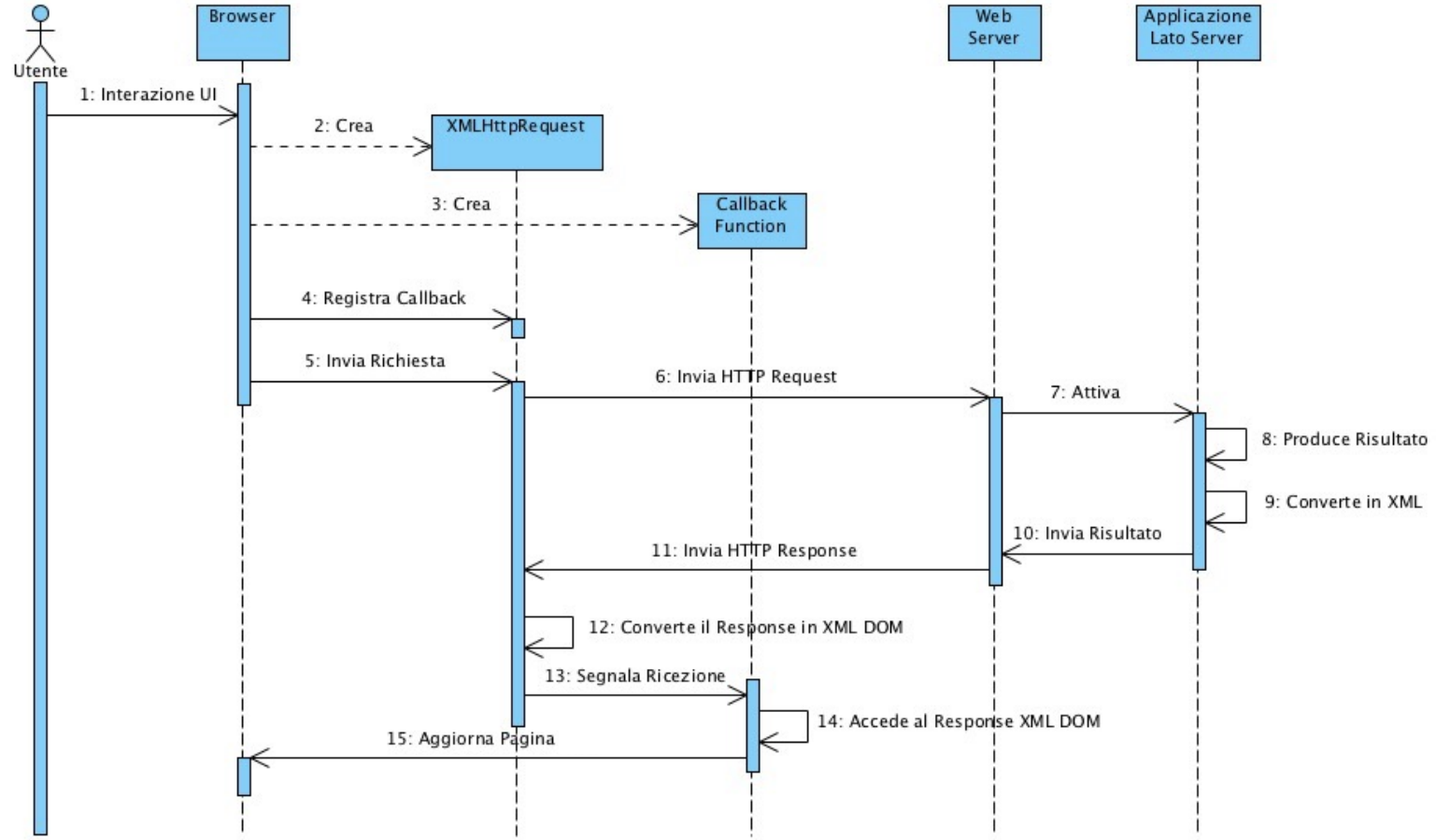


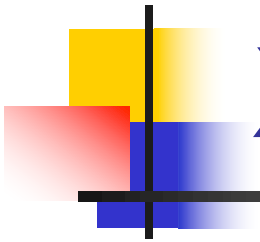


Interazione Asincrona: AJAX

- 1998: Microsoft introduce un modello di comunicazione asincrona tra client e server basata sulla definizione di un nuovo componente ActiveX per la sua applicazione Outlook Web Access (OWA)
- 2005: un modello di processo per comunicazione asincrona viene standardizzato e reso pubblico: AJAX (Asynchronous JavaScript and XML)
  - Interazione C/S gestita tramite un oggetto istanza dalla classe XMLHTTPRequest
  - Dati scambiati utilizzando, di norma, la codifica XML
- Risultato: possibilità di fare una richiesta al server ed acquisire dati che possono essere iniettati nel documento visualizzato senza doverlo ricaricare

AJAX Sequence Diagram



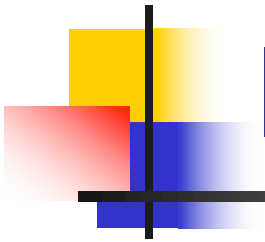


XML

- L'XML (eXtensible Markup Language) è un meta-linguaggio derivato dall'SGML e sviluppato a partire dal 1996 per strutturare documenti
- I Tag XML hanno stessa struttura di quelli definiti nell'HTML (sono parole racchiuse tra <> ed ammettono attributi con sintassi nome="valore")
- A differenza dell'HTML, i Tag XML non sono predefiniti, e quindi la loro interpretazione non è definita all'interno del linguaggio
- I Tag non definiscono in alcun modo il formato di visualizzazione dell'oggetto che racchiudono

XML - Esempio

```xml
<?xml version="1.0" encoding="UTF-8"?>
<oggetto>
  <province regione="Marche">
    <provincia>Ascoli Piceno</provincia>
    <provincia>Ancona</provincia>
    <provincia>Fermo</provincia>
    <provincia>Pesaro Urbino</provincia>
    <provincia>Macerata</provincia>
  </province>
  <province regione="Abruzzo">
    <provincia>Chieti</provincia>
    <provincia>L'Aquila</provincia>
    <provincia>Pescara</provincia>
    <provincia>Teramo</provincia>
  </province>
</oggetto>
```



Interazione AJAX

- Basata su due moduli software:
  - L'oggetto istanza della classe XMLHTTPRequest lato client
  - Un programma lato server che acquisisce la richiesta e fornisce i dati nel formato richiesto (XML)
- Processo lato client
  - Creazione istanza XMLHTTPRequest
  - Inizializzazione dell'oggetto: metodo di comunicazione (GET o POST), definizione del modulo da attivare sul server, definizione della funzione di callback per l'elaborazione dei dati provenienti dal server
  - Invio della richiesta al server

jQ11.html/php

Interazione AJAX con jQuery

ajax() – Utility Function	
`$.ajax(options)`	Invia una richiesta AJAX secondo le modalità specificate dall'oggetto `options`.
`options (set non completo)`	
`url`	L'url del programma sul server che riceve la richiesta
`type`	Il metodo HTTP (**GET** o POST) da usare per l'inoltro della richiesta.
`data`	I dati inviati nella richiesta al server (Types: `String|Object|Array`)
`dataType`	Codifica attesa dei dati ricevuti dal server `xml|html|json|text|…`
`success`	Nome della funzione di callback

Codifica JSON

- JSON (JavaScript Object Notation): notazione JS che consente di rappresentare in maniera semplice un oggetto per mezzo di una stringa che racchiude tra parentesi graffe le coppie nome:valore delle proprietà dell'oggetto, separate da virgola.
- Notazione ricorsiva: se un oggetto ne contiene un altro, quest'ultimo viene rappresentato allo stesso modo.

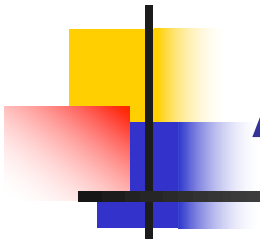
```
var proprietario = new Object();
proprietario.nome = 'Mario';
proprietario.cognome = 'Rossi';

var moto = new Object();
moto.modello = 'Ducati Diavel';
moto.anno = 2011;
moto.telaio = 156298876;
moto.proprietario = proprietario;
```

Notazione classica

```
var moto = {
    modello: 'Ducati Diavel',
    anno: 2011,
    telaio:156298876,
    proprietario {
        nome: 'Mario',
        cognome: 'Rossi'
    }
}
```

JSON



AJAX JSON

- Il formato JSON risulta più efficiente nello scambio di dati tra client e server in una sessione AJAX perché:
  - La codifica dei dati è più sintetica
  - Il parser JSON lato client è più efficiente
- JSON è diventato lo standard di fatto nelle interazioni AJAX
  - Variante al formato standard: sia i nomi che i valori delle componenti degli oggetti sono racchiusi tra apici doppi (")

AJAX – Laravel App

- Progetto laraProj6
  - Estensione di laraProj5
  - Validazione incrementale valori FORM inserimento prodotti lato server
- Obiettivi didattici: illustrare
  - Uso AJAX-JSON via jQuery integrato in Laravel

laraProj6: routing