

**Lo sviluppo delle
competenze di
sviluppatore front-
end**





Hello!

Alessandro Pasqualini

Full-stack developer, appassionato di programmazione e sviluppo software in generale, amo le serie tv e i gatti

1

jQuery



IL \$

jQuery ci mette a disposizione un alias per semplificare il suo utilizzo: **\$**.

\$ può essere usato come selettore degli elementi del DOM.

Funziona con gli stessi selettori di CSS e può usare anche le sue pseudoclassi.



Selezionare un elemento

```
<div id="mio-id">...</div>  
<div class="mia-classe">...</div>
```

```
<script>  
  $('#mio-id');  
  $('.mia-classe');  
</script>
```

\$ ci permette di selezionare uno o più elementi dal DOM e ottenere una referenza a loro semplicemente usando i classi selettori CSS.



Selettori CSS avanzati

div,p	Selects all <div> elements and all <p> elements
div > p	Selects all <p> elements where the parent is a <div> element
div + p	Selects all <p> elements that are placed immediately after <div> elements
p ~ ul	Selects every element that are preceded by a <p> element



Selettori CSS avanzati

<code>input[name="nome1"]</code>	Selects all <code><input></code> with attribute name equals to nome1
<code>input:checked</code>	Selects every checked <code><input></code> element
<code>:not(p)</code>	Selects every element that is not a <code><p></code> element
<code>:hidden</code>	Selects all elements which are hidden



Aggiungere/rimuovere classi

```
<div id="mio-id">...</div>
```

```
<script>  
    $('#mio-id').addClass('classe1');  
    $('#mio-id').removeClass('classe1');  
</script>
```

Per aggiungere una classe ad un elemento:

```
$('#...').addClass('classe');
```

Mentre per rimuovere una classe da un elemento:

```
$('#...').removeClass('classe');
```




Aggiungere/rimuovere classi

```
<div id="mio-id">...</div>
```

```
<script>
```

```
    $('#mio-id').toggleClass('classe1');
```

```
</script>
```

```
$('...').toggleClass('classe');
```

Aggiunge la classe 'classe' se non presente, mentre la toglie se già presente nell'elemento.

Aggiungere/rimuovere CSS inline



```
<div id="mio-id">...</div>
```

```
<script>  
    $('#mio-id').css('background-color', 'red');  
</script>
```

E' possibile aggiungere regole CSS inline semplicemente con;
`$('...').css('regola', 'valore');`

Mostrare/nascondere un elemento



```
<div id="mio-id">...</div>
```

```
<script>  
    $('#mio-id').hide();  
    $('#mio-id').show();  
</script>
```

Per nascondere un elemento, ovvero applicare `display: none;`

```
$('#...').hide();
```

Per mostrare un elemento, ovvero applicare `display: block;`

```
$('#...').show();
```



Testare la presenza di elementi

```
<div id="mio-id">...</div>
```

```
<script>  
  if ($('#mio-id').length) {  
    ...  
  }  
</script>
```

Per testare se uno o più elementi esistono all'interno della pagina:

`$('#...').length`

Sarà maggiore di zero, ovvero uguale al numero di elementi selezionati



Attributi VS proprietà

Un attributo aggiunge delle informazioni ad un elemento (es. href).

Una proprietà descrive le caratteristiche di un elemento (es. checked)

```
<a href="#">link</a> <!-- href è un attributo -->
```

```
<input type="checkbox" checked> <!-- checked è una proprietà -->
```



Attributi

```
<a id="mio-a" href="https://google.com">...</a>
```

```
<script>
```

```
    console.log($('#mio-a').attr('href'));
```

```
</script>
```

Per accedere agli attributi:

```
$('#...').attr('nome_attributo');
```

Per impostare un attributo

```
$('#...').attr('nome_attributo', 'valore_attributo');
```

× Expression

not available

⚠ A cookie associated with this site has been set. To deliver cookies with Chrome, you must visit this site again. See more details at

⚠ A cookie associated with this site has been set. To deliver cookies with Chrome, you must visit this site again. See more details at

<https://google.com>

> |



Proprietà

```
<input type="checkbox" id="check">  
<script>  
    $('#check').prop('checked', true);  
</script>
```

Per accedere ottenere il valore di una proprietà:

```
$('#...').prop('nome_ proprietà ');
```

Per impostare un proprietà

```
$('#...').prop('nome_proprietà', 'valore');
```



Attributi 'data-'

E' possibile usare degli attributi aggiuntivi **non standard** agli elementi (es. per contenere delle informazioni che ci fornisce il server) usando gli attributi data-.

```
<div data-mio-attributo="mio-valore"><div>
```




Attributi 'data-'

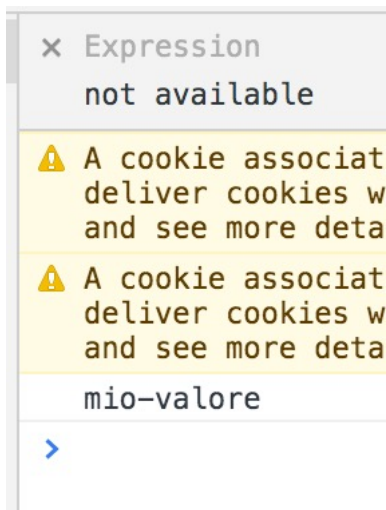
```
<div data-mio-attributo="mio-valore" id="mio-id"><div>  
<script>  
  console.log($('#mio-id').data('mio-attributo'));  
</script>
```

Per accedere ottenere il valore di un attributo data:

```
$('#...').data('nome-senza-data');
```

Per impostare un attributo data:

```
$('#...').data(nome-senza-data', 'valore');
```





Modificare il contenuto

```
<div id="mio-id"><div>  
<script>  
    $('#mio-id').html($('#<p>test</p>'));  
</script>
```

test

Per accedere al contenuto:

```
$('#...').html();
```

Per impostare il contenuto:

```
$('#...').html('valore');
```



Creare un elemento

```
<script>  
    $('<p>Test</p>').appendTo($('body'));  
</script>
```

test

Per creare un elemento è sufficiente passare html come stringa a \$

`$(..).appendTo(el)`

Permette di appendere un elemento ad un altro (oppure di muoverlo)

Accedere al contenuto di un input



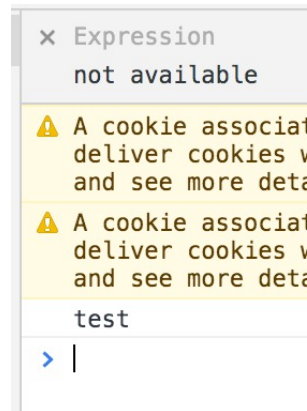
```
<input id="mio-i" value="test">
<script>
  console.log($('#mio-i').val());
</script>
```

Per accedere al contenuto:

```
$('#...').val();
```

Per impostare il contenuto:

```
$('#...').val('valore');
```





Eventi ed event handler

```
<button id="mio-id">Test</button>
<script>
    $('#mio-id').on('click', function () {
        alert('cliccato');
    });
</script>
```



Eventi ed event handler

jQuery permette di associare un event handler ad un evento, ed il click di un bottone. La sintassi base è:

```
$(..).on('evento', function () {  
  // Codice  
});
```

Allo stesso modo è possibile anche eliminare un event handler

```
$(..).off('evento');
```



Transizioni di jQuery

jQuery possiede alcune transizioni di base utili per mostrare o nascondere del contenuto:

- fadeIn /fadeOut
- slideDown/slideUp



`$(this)`

Quando usate jQuery e siete all'interno di una funzione (ad esempio perchè state scrivendo un event handler) a volte capita di dover accedere all'elemento che ha generato l'evento.

jQuery mette a disposizione questa sintassi:

`$(this)`



preventDefault();

Se provate a registrare un click event in un link il browser segue comunque il link dopo aver eseguito il vostro handler.

Per prevenire questo problema esiste una "funzione" capace di inibire il comportamento di default.

```
$('a').on('click', function (e) {  
    e.preventDefault();  
    // Codice  
});
```

Ajax (Asynchronous Javascript And Xml)



Ajax

Ajax è la tecnologia più usata per realizzare applicazioni interattive. Il concetto che sta alla base è quello di scambiare dati con il browser senza dover ricaricare la pagina.

jQuery ci permette di realizzare delle chiamate AJAX con pochissimo sforzo.

Le chiamate AJAX sono chiamate HTTP classiche (GET, POST, etc).

AJAX originariamente usava XML come formato di trasporto dei dati ma è stato quasi completamente sostituito da JSON (e da altri formati)



\$.load

\$.load è un metodo di jQuery (\$) di jQuery è un oggetto) che ci permette di caricare del contenuto (principalmente HTML) in modo asincrono.

```
<button id="cliccami">Cliccami</button>
<div id="contenuto"></div>
<script>
    $('#cliccami').click(function () {
        $('#contenuto').load('snippet.html', function () {
            console.log('Contenuto caricato');
        });
    })
</script>
```

`$.load`



Cliccami

Cliccami





\$.load



Il file snippet.html

```

```







\$.load

Name	Status	Type
 corso	200	document
 jquery.min.js	200	script

XHR significa
XMLHttpRequest.

XMLHttpRequest è
l'API (Application
Program Interface)
messa a
disposizione dal
browser per fare
richieste asincrone

Name	Status	Type
 corso	200	document
 jquery.min.js	200	script
 snippet.html	200	xhr
 cat.png	200	png



\$.get

\$.get è un metodo di jQuery che ci permette di effettuare una richiesta HTTP GET per prelevare del contenuto in maniera asincrona.

```
$(document).ready(function () {  
    $.get('users.json', function (usernames) {  
        console.log(usernames);  
    });  
});
```




\$.get

```
▼ (2) [{...}, {...}] ⓘ  
  ▶ 0: {username: "ilmiousername1", signupDate: 1578820209000}  
  ▶ 1: {username: "ilmiousername2", signupDate: 1578820209000}  
    length: 2  
  ▶ __proto__: Array(0)
```

```
[{  
  "username": "ilmiousername1",  
  "signupDate": 1578820209000  
}, {  
  "username": "ilmiousername2",  
  "signupDate": 1578820209000  
}]
```

Il file
users.json



`$(document).ready(...)`

`$(document).ready(...)` ci permette di registrare un callback (una funzione) da eseguire quando il browser emettere l'evento "pagina pronta".

Questo ci permette di ritardare l'esecuzione del codice fino a che la pagina non è pronta.

```
$(document).ready(function () {  
    console.log('La pagina è pronta');  
});  
console.log('Scrivi subito');
```

Scrivi subito
La pagina è pronta



`$(window).on('load', ...)`

`$(document).load(...)` ci permette di registrare un callback (una funzione) da eseguire quando il browser emettere l'evento "pagina completamente caricata" comprensiva di immagini, iframe etc.

```
$(window).on('load', function () {  
    console.log('La pagina è caricata');  
});  
$(document).ready(function () {  
    console.log('La pagina è pronta');  
});  
console.log('Scrivi subito');
```

Scrivi subito

La pagina è pronta

La pagina è caricata

.



`$(window).on('load', ...)`

Attenzione che **load** funziona solo su window e non document.

```
$(document).on('load', function () {  
    console.log('La pagina è caricata');  
});
```



```
$(window).on('load', function () {  
    console.log('La pagina è caricata');  
});
```

La pagina è caricata





\$.post

\$.post è un metodo di jQuery che ci permette di effettuare una richiesta HTTP POST per inviare del contenuto al server in modo asincrono.

\$.post accetta un oggetto Javascript contenente i dati da inviare, quindi il nostro compito è crearlo inserendo i dati da comunicare al server.

\$.post permette anche di impostare un il tipo dei dati che il server ci ritornerà (ad esempio json). Se non è impostato viene considerato testo e quindi l'eventuale json NON è automaticamente trasformato in un oggetto Javascript.



\$.post

```
$.post(  
  
    'url_della_pagina',  
  
    { ... },  
  
    function (risposta) {  
        ...  
    },  
  
    'tipo_dei_dati_in_risposta_dal_server'  
);
```



\$.ajax

jQuery mette a disposizione dei programmatori anche un metodo chiamato \$.ajax che permette di avere un controllo maggiore sulla richiesta asincrona.

\$.load, \$.get e \$.post sono solamente delle "scorciatoie" per \$.ajax.

Nella documentazione di jQuery è possibile trovare tutti i parametri accettati da questi metodi per modificarne il comportamento. Ad esempio è possibile inserire nella richiesta HTTP headers aggiuntivi.



Il problema degli event handler

jQuery ci permette di registrare delle funzioni da eseguire quando si verifica un certo evento. es. click

```
$('#id').click(function () { ... });
```

Questi event handler sono però registrati al caricamento della pagina sul contenuto attuale. Se carichiamo del nuovo contenuto HTML gli event handler NON vengono registrati nel il nuovo codice.



Il problema degli event handler

```
<button class="cliccami">Cliccami 1</button>
<script>
  $('.cliccami').click(function () {
    alert('ok');
  });
</script>
```

Se ora aggiungiamo un nuovo bottone attraverso del codice JS (magari caricandolo con \$.load)...



Il problema degli event handler

```
<button class="cliccami">Cliccami 1</button>
```

```
<button class="cliccami">Nuovo bottone</button>
```

```
<script>  
  $('cliccami').click(function () {  
    alert('ok');  
  });  
</script>
```

L'event handler non funziona per il nuovo bottone



Il problema degli event handler

La soluzione è registrare l'event handler a livello di **document**

```
$('.cllicami').click(function () {  
    alert('ok');  
});
```

```
$(document).click('.cllicami', function () {  
    alert('ok');  
});
```

3

CORS



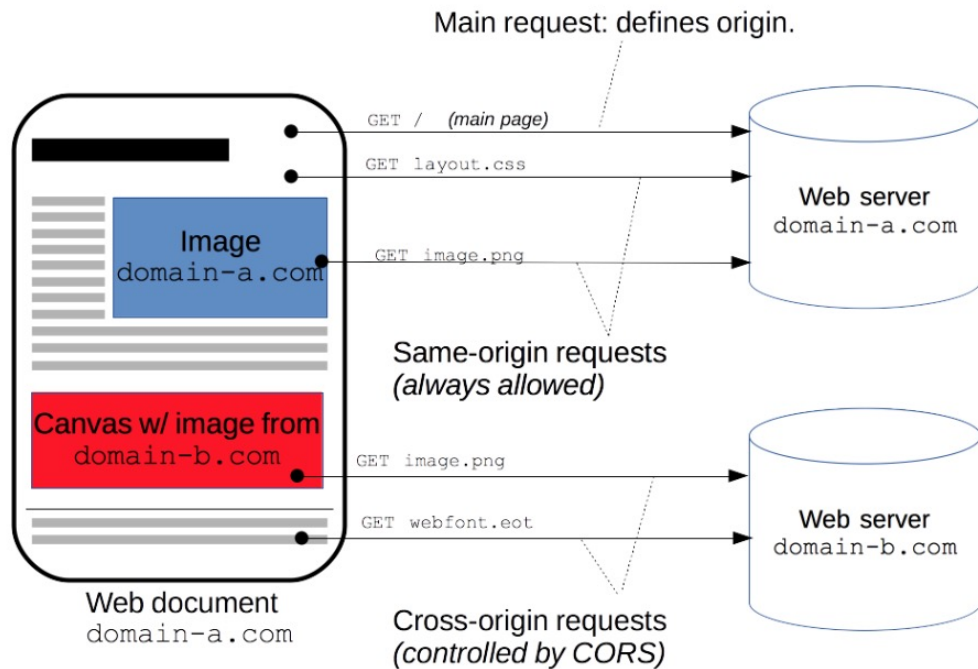
CORS

CORS (Cross Origin Resource Sharing) è un meccanismo usato per indicare al browser che l'applicazione in esecuzione su una specifica origine (il dominio) ha l'autorizzazione di accedere a risorse su un'altra origine (altro dominio).

Per ragioni di sicurezza i browser limitano l'utilizzo di CORS per proteggere l'utente da situazioni pericolose.

Tutte le richieste CORS che provengono da script (Javascript) vengono sottoposte al controllo del browser prima di consentire la richiesta.

CORS





CORS

Riprendendo l'esempio di \$.load ma caricandolo all'esterno del web server

```
<button id="cliccami">Cliccami</button>
<div id="contenuto"></div>
<script>
  $('#cliccami').click(function () {
    $('#contenuto').load('snippet.html', function () {
      console.log('Contenuto caricato');
    });
  });
}
</script>
```

CORS



Cliccami

✖ ▶ Access to XMLHttpRequest at 'file:///Users/alessandro/Desktop/Corso/Slides/20200113/test/snippet.html' from origin 'null' has been blocked by CORS policy: Cross origin requests are only supported for protocol schemes: http, data, chrome, chrome-extension, https.

>



CORS

Riprendendo l'esempio di \$.load ma caricandolo all'esterno del web server

```
<button id="cliccami">Cliccami</button>
<div id="contenuto"></div>
<script>
  $('#cliccami').click(function () {
    $('#contenuto').load('http://localhost/snippet.html', function () {
      console.log('Contenuto caricato');
    });
  });
}
</script>
```

CORS



Cliccami

✖ Access to XMLHttpRequest at '<http://localhost/snippet.html>' [sop.html:1](#) from origin '<http://corso>' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

Creare elementi con jQuery



Creare elementi con jQuery

Capita spesso di dover generare elementi HTML dinamicamente usando Javascript in risposta a certi eventi (callback, richieste/risposte con il server mediante Ajax, etc)

jQuery ci permette di generare elementi utilizzando un sintassi molto semplice:

```
var elemento = $('<div></div>');
```



Creare elementi con jQuery

```
var elemento = $('<div></div>');
```

jQuery ci ritorna un oggetto che rappresenta il nostro nuovo elemento.

A questo punto possiamo modificarlo ad esempio aggiungendo classi, css, event handler (es. click), etc

Successivamente dobbiamo ricordarci di includerlo all'interno della pagina.



Creare elementi con jQuery

```
var elemento = $('<div></div>');  
$('body').append(elemento);
```

Quando jQuery crea un elemento **non lo aggiunge** automaticamente al DOM (e quindi al documento) e di conseguenza l'elemento non viene reso (rendered) dal browser.

Per questo jQuery ci mette a disposizione un paio di metodi (funzioni) per aggiungere l'elemento al DOM.



Creare elementi con jQuery

```
var elemento = $('<div></div>');  
$('body').append(elemento);
```

Innanzitutto dobbiamo scegliere un elemento (ad esempio il body) come "punto di ingresso" per inserire il nuovo elemento. E poi chiamare una delle seguenti funzioni:

- **append** aggiunge l'elemento come **ultimo figlio** del "punto di ingresso".
- **prepend** aggiunge l'elemento come **primo figlio** del "punto di ingresso".



Creare elementi con jQuery

```
var elemento = $('<div></div>');  
$('#contenuto').insertBefore(elemento);
```

Oppure possiamo scegliere se inserire il nuovo elemento prima o dopo il nostro elemento "punto di ingresso", ovvero decidiamo di inserirlo come **sibling** (fratello).

- **insertBefore** aggiunge l'elemento sopra il "punto di ingresso".
- **insertAfter** aggiunge l'elemento sotto il "punto di ingresso".



Templating Javascript

Abbiamo visto che con jQuery possiamo creare nuovi elementi e aggiungerli al DOM con delle semplici chiamate.

Se però dobbiamo creare una struttura HTML "complessa" ciò risulta essere complesso con jQuery dovendo scrivere molto codice.

Javascript ha così introdotto il concetto di templating.



Templating Javascript

Le stringhe templating sono racchiuse dal carattere backtick (` `), che **non è** l'apice singolo) e possono comprendere più linee.

```
var tpl = `

Naturalmente le stringhe di templating sono solamente stringhe e non un elemento. Dovremo comunque fare ricorso a jQuery per costruire gli elementi ed inserirli nel DOM.



58


```



Templating Javascript

```
var tpl = `    <p>Sono un template</p>  
</div>`;
```

```
var elemento = $(tpl);  
$('body').append(elemento);
```

```
var div = $('<div></div>')  
    .addClass('myclass');  
  
var p = $('<p></p>')  
    .text('Sono un template');  
  
div.append(p);  
$('body').append(div);
```



Templating Javascript

La vera potenza delle stringhe di templating di Javascript è la possibilità di usare dei **segnaposto**.

Un segnaposto non è altro che un'espressione Javascript (del codice javascript).

```
var text = 'Sono un template';  
var tpl = `    <p>${text}</p>  
    </div>`;
```

```
var elemento = $(tpl);  
$('body').append(elemento);
```



Templating Javascript

I segnaposti sono composti in questo modo:

`${espressione}`

Possono essere inclusi direttamente nella stringa template:

```
var tpl = `1 + 1 = ${1+1}`;
```

```
var tpl = `1 + 1 = ${'1'+ '1'}`;
```

```
var tpl = `1 + 1 = 2`;
```

```
var tpl = `1 + 1 = 11`;
```

5

Esercizio



Esercizio

- Creare un form di registrazione con bootstrap. Il form deve avere nome, cognome, username, email, password e conferma password.
 - Il form deve essere inviato al file form.php
 - L'attributo name dei vari input deve essere firstname, lastname, username, email, password, confirm-password
- Intercettare con jQuery l'invio del form (o del click sul pulsante di registrazione)
- Validare il form di registrazione in modo che il nome e il cognome siano lunghi almeno 3 caratteri, lo username abbia lunghezza compresa tra 7 e 20 caratteri, le due password coincidano e la mail sia valida.

Per validare la mail:

<https://stackoverflow.com/questions/46155/how-to-validate-an-email-address-in-javascript>

Esercizio



Esercizio

- Continuando l'esercizio precedente, l'utente non può inserire uno username già utilizzato.
 - Quando l'utente inserisce lo username, validate "on the fly" lo username con le regole precedenti.
 - Inviare con ajax lo username inserito dall'utente al file username.php che vi dirà se lo username è già stato utilizzato o meno
 - Se lo username è valido mettere una spunta verde, altrimenti una X rossa
- Validare la password in modo che sia sicura (almeno un carattere maiuscolo, un carattere minuscolo, un numero e un carattere speciale)

Esercizio



Load more

- L'idea è creare un pulsante che vi permette di caricare altri post (immaginate la pagina categoria di un blog)
- Il file posts.php vi ritorna un array JSON contenente un numero il numero di post da voi richiesto
 - <http://localhost/post.php?n=10> vi ritorna 10 post
- Dovete creare un bottone che se premuto aggiunge sopra di lui il i post ritornati (dovete costruire anche gli elementi HTML)

Esercizio



Scroll infinito

- Il funzionamento è simile al pulsante Load more dell'esercizio precedente solo che questa volta non c'è bottone: ogni volta che scrollate la pagina vengono caricati nuovi post (l'idea è quella del feed di Facebook)

Esempio: la chat



Thanks!

Any questions?

Per qualsiasi domanda contattatemi:
alessandro.pasqualini.1105@gmail.com