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

# Selettori CSS avanzati

| | |
|---|---|
| input[name="nome1"] | Selects all <input> with attribute name equals to nome1 |
| input:checked | Selects every checked <input> element |
| :not(p) | Selects every element that is not a <p> element |
| :hidden | 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 prorietà

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

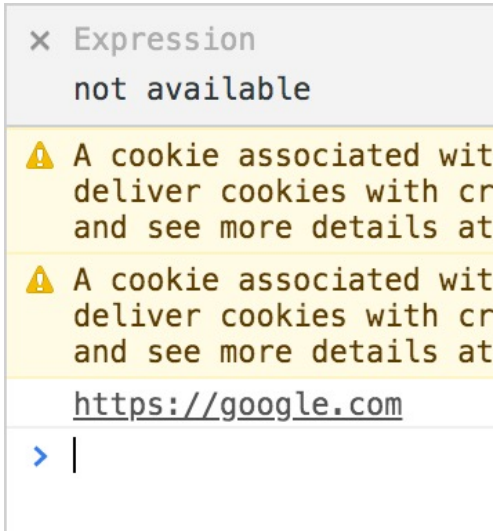
**Una proprietà descrive le caratteristiche di un elemento** (es. checked)

```
<a href="#">link</a> <!-- href è un attributo -->
<input type="checkbox" checked> <!-- checked è una proprieta'-->
```

# 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);
```

# 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');`

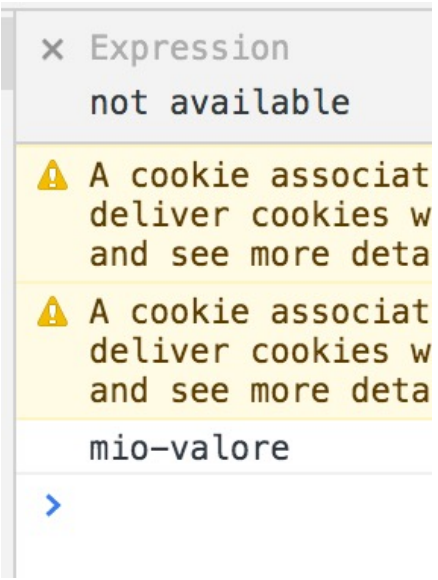
# Attributi 'data-'

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>
```

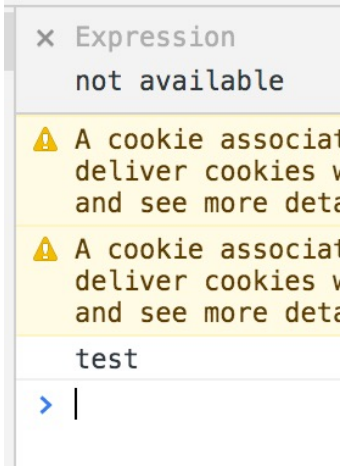

Per creare un elemento è sufficiente passare html come stringa a $

`$(..).appentTo(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 sinstassi:

```
$(this)
```