Interactive Web Design & Development

Prototipazione di pagine e interfacce web interattive con programmazione in Javascript. Utilizzo della libreria JQuery

Unità Didattica UD02: Oggetti e librerie; introduzione a JQuery

prof. Giovanni Borga

Gli oggetti in Javascript

Esempio di definizione di un oggetto Javascript

Sintassi:

```
var nomeoggetto = {proprieta: valore, … … … metodo: definizione}
```

Esempio:

```
<script>
  var oggetto1 = {
    prop_numero: 12,                              // proprietà di tipo numerico intero
    prop_testo: 'Questa è una stringa’,           // proprietà di tipo stringa
    metodo1() {alert('Hai invocato il metodo 1')} // questo è un metodo
  };
</script>
```

In sostanza la definizione di un oggetto Javascript ha quasi la stessa sintassi di una regola CSS; l’unica differenza è il **separatore delle proprietà/eventi** che, anziché punto e virgola **è la virgola semplice**.

Esempio di utilizzo di un oggetto Javascript

Lettura del valore di una proprietà:

```
var dato = oggetto1.prop_numero;

document.getElementById('textbox1').value = oggetto1.prop_testo;
```

Utilizzo di un metodo:

```
<input type="button" onclick="oggetto1.metodo1()" value="premi">
```

Librerie Javascript

Come accade con i fogli di stile CSS, anche con Javascript è possibile:

- **Realizzare propri insiemi** di elementi e funzioni
- **Incorporare librerie precompilate** più o meno complesse.

L'utilizzo efficace di librerie precompilate è ovviamente subordinato alla disponibilità di

- **Documentazione** esaustiva
- **Tutorial** di testo o video
- **Esempi funzionanti** da personalizzare.

Esempi di applicazione di librerie precompilate: griglia «masonry»

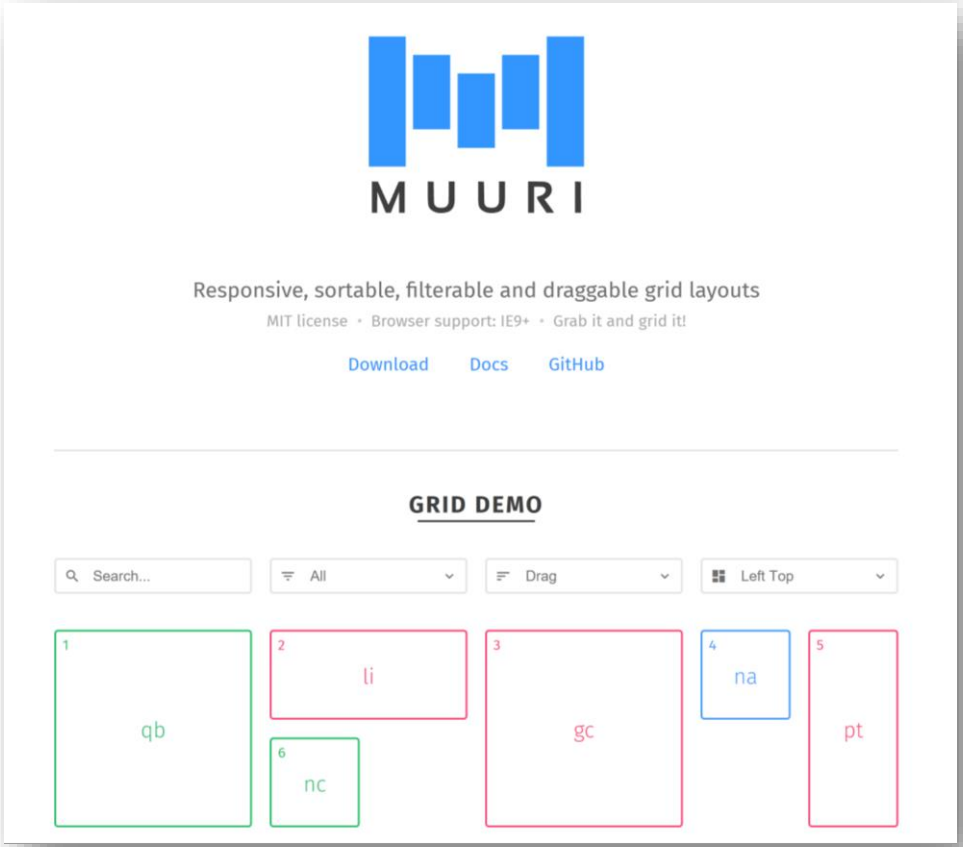
Masonry grid

Il termine «masonry» allude alla composizione delle **murature in mattoni o blocchi** ad incastro dove elementi rettangolari vengono apposti in modo da riempire l'area senza lasciare spazi.

Le «griglie masonry» sono utilizzate nelle pagine web per realizzare **«patchwork» di elementi quadrangolari** anche diversi ottimizzando lo spazio disponibile. Con Javascript è possibile anche rendere dinamiche le composizioni permettendo animazioni e interazioni.

Una libreria pronta per realizzare layout di questo tipo è MUURI (https://haltu.github.io/muuri/).

La libreria è gratuita e open-source.



Utilizzo di MUURI

Elementi in HEAD

Riferimenti a librerie Javascript ausiliarie esterne

```
<script src="velocity.min.js"></script>
<script src="hammer.min.js"></script>
```

Foglio di stile interno

```
<style>
	.grid {
		position: relative;
		width: 80%;
		margin: auto;
		border: 1px solid blue;
	}
	.item {
		display: block;
		position: absolute;
		margin: 1px;
		padding: 5px;
		z-index: 1;
		border: 0px solid red;
		background-color: silver;
	}
… … …
```

Utilizzo di MUURI

Elementi in BODY

Riferimento alla libreria esterna principale MUURI

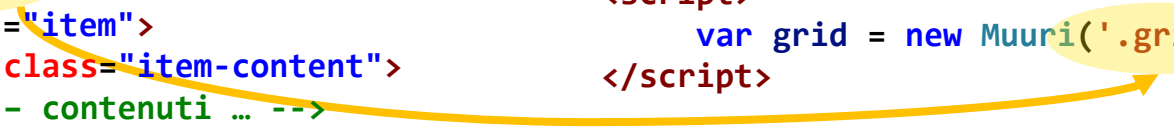
```
<!-- Muuri needs to have access to document.body when initiated -->
<script src="muuri.min.js"></script>
```

Strutturazione dei contenuti

```
<div class="grid">
     <div class="item">
          <div class="item-content">
             <!- contenuti … -->
          </div>
     </div>
     <div class="item">
          <div class="item-content">
             <!- contenuti … -->
          </div>
     </div>
</div>
```

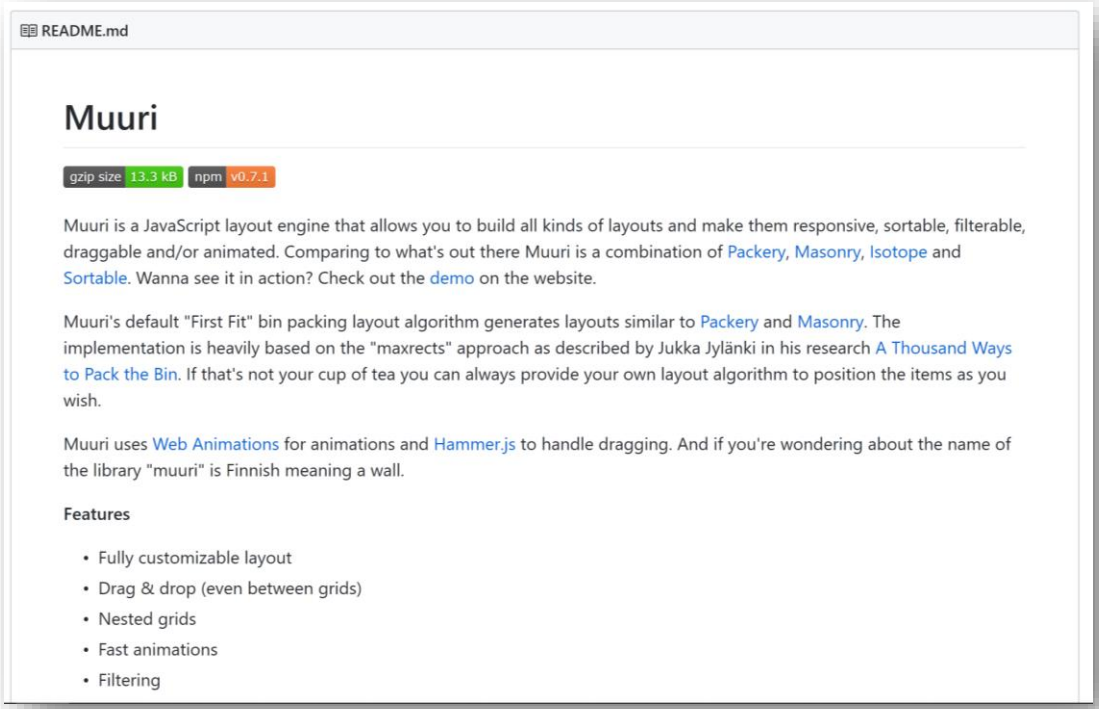
Script di applicazione delle funzioni al tag

```
<script>
     var grid = new Muuri('.grid');
</script>
```



Documentazione di MUURI

https://github.com/haltu/muuri#table-of-contents



README.md

Muuri

gzip size 13.3 kB | npm v0.7.1

Muuri is a JavaScript layout engine that allows you to build all kinds of layouts and make them responsive, sortable, filterable, draggable and/or animated. Comparing to what's out there Muuri is a combination of Packery, Masonry, Isotope and Sortable. Wanna see it in action? Check out the demo on the website.

Muuri's default "First Fit" bin packing layout algorithm generates layouts similar to Packery and Masonry. The implementation is heavily based on the "maxrects" approach as described by Jukka Jylänki in his research A Thousand Ways to Pack the Bin. If that's not your cup of tea you can always provide your own layout algorithm to position the items as you wish.

Muuri uses Web Animations for animations and Hammer.js to handle dragging. And if you're wondering about the name of the library "muuri" is Finnish meaning a wall.

Features

- Fully customizable layout
- Drag & drop (even between grids)
- Nested grids
- Fast animations
- Filtering

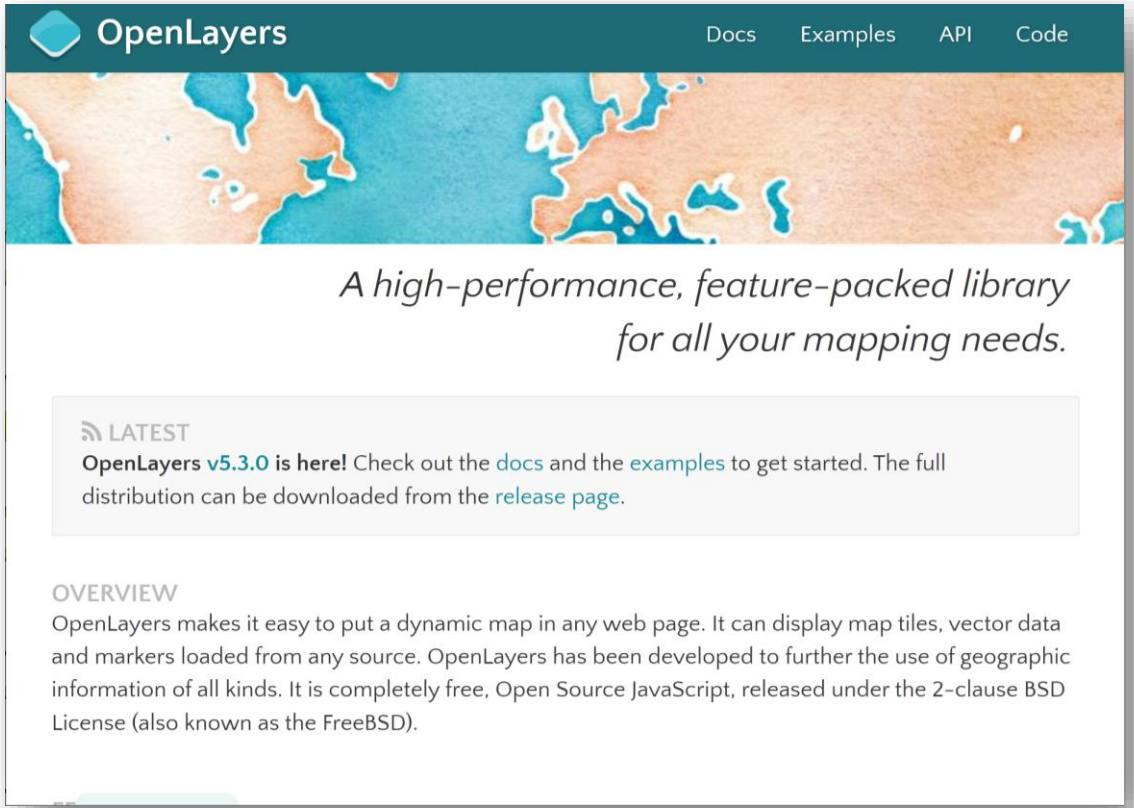
Esempi di applicazione di librerie precompilate: mappa geografica interattiva

OpenLayers

Siamo ormai abituati a trovare su siti web di vario tipo delle mappe geografiche interattive, ovvero **zoomabili e navigabili a diversi livelli** e con diversi «strati» di informazioni che possono essere cliccate per ottenere dettagli informativi.

Anche per questo tipo di tool non esiste un elemento predefinito in HTML; occorre sviluppare l'interfaccia in Javascript.

OpenLayers (www.openlayers.org) è **una ricca libreria di oggetti e funzioni open-source** sviluppata per agevolare i creatori di siti web nell'incorporare mappe interattive nelle proprie pagine.



OpenLayers quick start

Gli «ingredienti» per iniziare ad utilizzarla sono i seguenti:

La libreria:

```
<script src="ol.js"></script>
```

Un foglio di stile precompilato da abbinare alla libreria:

```
<link rel="stylesheet" href="ol.css" type="text/css">
```

Un **tag DIV** nel BODY opportunamente dimensionato (es: 500x300px) con ID e CLASS definiti (es: `map`) nel punto in cui si vuole che venga posizionata la mappa.

Questo script nel BODY per attivare la mappa (l'opzione target coincide con il valore dell'ID del DIV):

```
var map = new ol.Map({
            target: 'map',
            layers: [
                new ol.layer.Tile({
                    source: new ol.source.OSM()
                }),
                new ol.layer.Vector({
                    source: new ol.source.Vector({
                        url: 'layer1.geojson',
                        format: new ol.format.GeoJSON()
                    })
                })
            ],
            view: new ol.View({
                center: ol.proj.fromLonLat([12.32, 45.44]),
                zoom: 14
            })
      });
```

Analisi dello script

```
var map = new ol.Map({
            target: 'map',
            layers: [
                new ol.layer.Tile({
                    source: new ol.source.OSM()
                }),
                new ol.layer.Vector({
                    source: new ol.source.Vector({
                        url: 'layer1.geojson',
                        format: new ol.format.GeoJSON()
                    })
                })
            ],
            view: new ol.View({
                center: ol.proj.fromLonLat([12.32, 45.44]),
                zoom: 14
            })
      });
```

Creazione dell'oggetto mappa

Associazione della mappa al DIV

Configurazione dei layers:
1. OpenStreetMap
2. Vettoriale *geojson*

Definizione vista iniziale

Introduzione a JQuery

Cos'è jQuery?

jQuery è una libreria JavaScript veloce e leggera per la realizzazione di **effetti dinamici sulle pagine HTML**.

jQuery semplifica la manipolazione, la gestione degli eventi, l'animazione e le funzioni Ajax degli elementi HTML attraverso apposite **API cross-browser semplici e intuitive**.

Combinando **versatilità ed estensibilità**, jQuery ha modificato significativamente lo sviluppo di pagine web comprendenti animazioni ed effetti client-side.

Logica di fondo in jQuery

Il nome di jQuery deriva dalla sua logica di sviluppo, ovvero dal fatto che le azioni vengono applicate ad un **set di elementi HTML selezionati con un criterio di ricerca** *CSS-based*.

In sostanza la logica di jQuery può essere riassunta in due punti:

1. Gli elementi da «animare» sono selezionati tramite i selettori CSS (type selector, class, id e loro combinazioni)

2. Gli effetti vengono associati alla selezione con il carattere del punto (.)

La sintassi di base per l'applicazione di questa logica è la seguente:

```
$(" selezione ").effetto(opzioni);
```

Concetti di base per l'utilizzo di jQuery

Tutti i comandi di jQuery iniziano con il simbolo $

$ è in pratica un alias di jQuery

Tutto il codice di utilizzo della libreria è associato all'evento READY del documento

Questa tecnica prevede semplicemente che il nostro codice vada SEMPRE contenuto nella seguente frase:

```
$(document).ready ( function() {
        …
        …
});
```

Primi esempi di utilizzo di jQuery

Per usare il comando «hide» di jQuery, che ha la funzione di nascondere un elemento, possiamo scrivere:

`$(this).hide()`	nasconde l'elemento corrente.
`$("p").hide()`	nasconde tutti gli elementi <p>.
`$(".test").hide()`	nasconde tutti gli elementi con class="test".
`$("#test").hide()`	nasconde l'elemento con id="test".

Le «query» di selezione degli elementi sono:

- **this** (elemento corrente)
- **"p"** (type selector)
- **".test"** (class selector)
- **"#test"** (id selector)

NB: tutti i selettori CSS sono racchiusi da doppi apici.

Associazione degli effetti jQuery agli eventi

Molte azioni sono associate agli eventi della pagina che sono i seguenti:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

Il codice per associare effetti da azionare quando si clicca sopra un qualsiasi paragrafo è il seguente:

```
$("p").click(function() {
  ...
  ... codice da eseguire ...
  ...
});
```

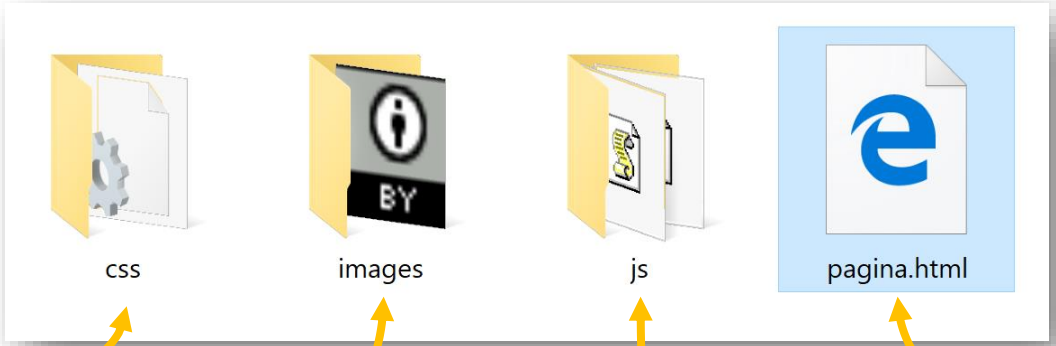
Documentazione

I principali riferimenti per l'utilizzo di jQuery sono:

- **La documentazione principale:** http://api.jquery.com
- **Il portale per l'autoformazione:** http://learn.jquery.com
- **Il tutorial fornito dal W3C:** http://www.w3schools.com/jquery

Primo esempio di applicazione di Jquery funzione mostra/nascondi

Strutturazione tipica delle cartelle di un sito web



Fogli di stile CSS

Immagini GIF, JPG, PNG

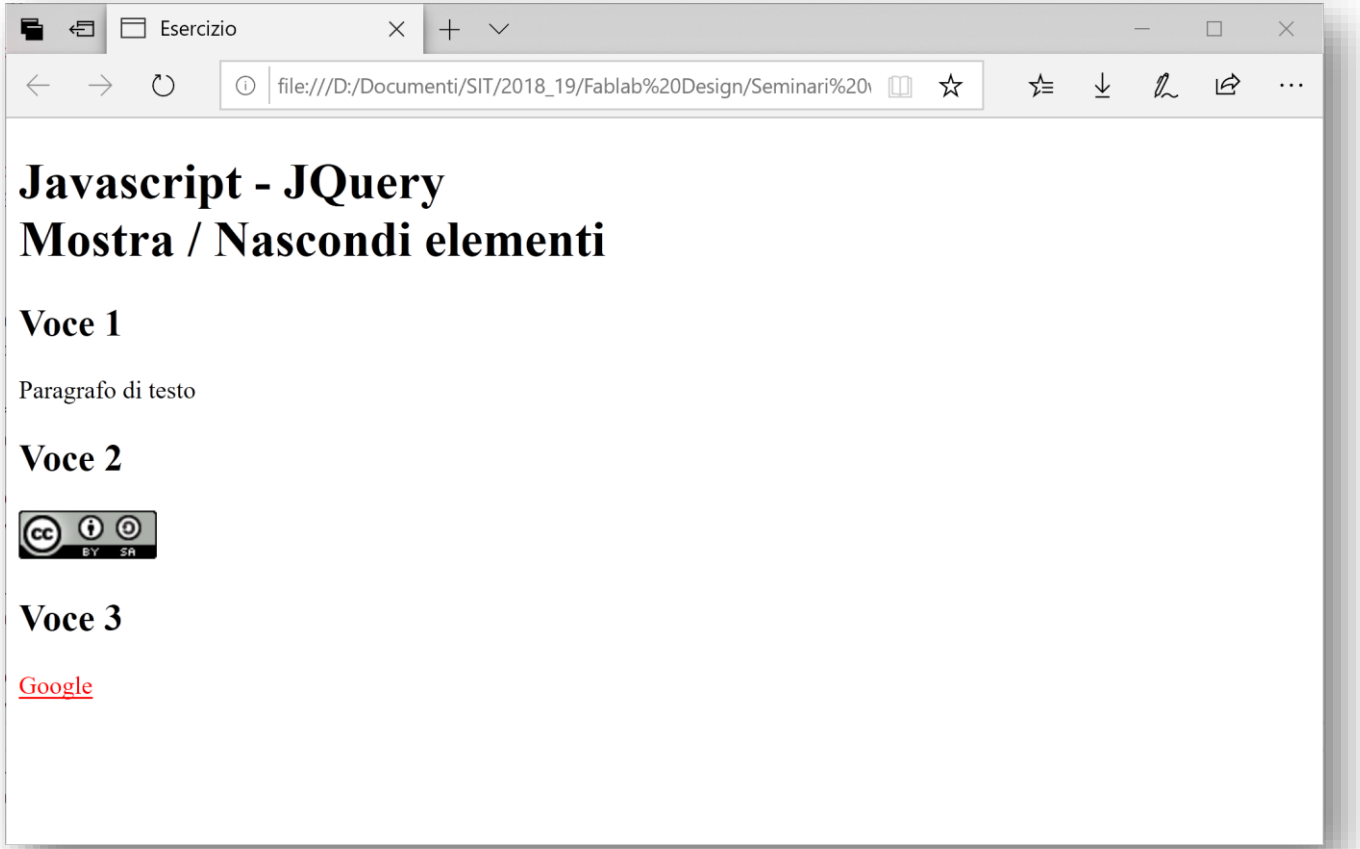
Librerie e scripts JS

Pagine HTML (nella cartella radice)

Markup della pagina

```
<!DOCTYPE html>
<html>
	<head>
		<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
		<title>Esercizio</title>
	</head>
	<body>
		<h1>Javascript - JQuery<br />
		Mostra / Nascondi elementi
		</h1>
		<div id="menu">
		   <span class="interruttore"><h2>Voce 1</h2></span>
		   <div class="contenuto">
		      <p>Paragrafo di testo</p>
		   </div>
		   <span class="interruttore"><h2>Voce 2</h2></span>
		   <div class="contenuto">
		      <p><img src="images/cc.png"></p>
		   </div>
		   <span class="interruttore"><h2>Voce 3</h2></span>
		   <div class="contenuto">
		      <a style="text-align:center; color:red;" href="http://www.google.it" title="Google"
		      target="_blank">Google</a>
		   </div>
		</div>
	</body>
</html>
```

Prima versione della pagina



Creazione del CSS

```css
body {
	padding: 0;
	margin: auto;
	background-color: silver;
}

/*Stile delle intestazioni*/

h1 {
	text-align: center;
	line-height: 35px;
}

h2 {
	width: 300px;
	height: 40px;
	margin: 0;
}
```

```css
/*Stile delle intestazioni del menu*/

#menu {
	width: 350px;
	margin: 0 auto;
}

span.interruttore {
	background-color: blue;
	display: block;
}

.contenuto {
	display: none;
	background-color: #ADD8E6;
}
```

Il file va salvato nella cartella «css» (ad es. col nome «stylesheet.css») e va aggiunto al markup il riferimento:

```
<link rel="stylesheet" type="text/css" href="css/stylesheet.css" />
```

Pagina con gli stili

Creazione dello script

1) Seleziona tutti gli elementi con classe «interruttore» e intercetta l'evento click di ciascuno.
2) Associa ad essi una specifica funzione.

```
$(document).ready(function () {
    $(".interruttore").click(function () {
        $(this).next().toggle();
    });
});
```

3) Seleziona l'elemento corrente.
4) Individua il primo elemento al suo interno.
5) Mostra/Nascondi l'elemento (comando *toggle* di jQuery).

Il file va salvato nella cartella «js» (ad es. col nome «toggle.js») e va aggiunto al markup il riferimento:

```
<script src="js/toggle.js"></script>
```

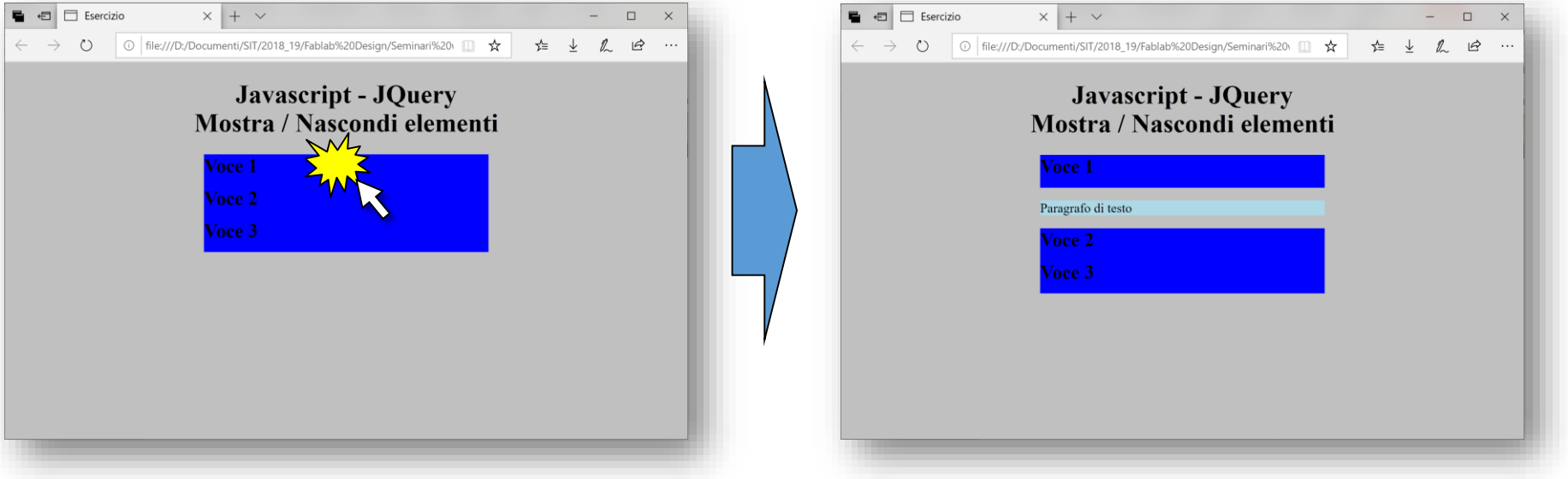
Inclusione di jQuery

Occorre infine scaricare la libreria ufficiale, salvarla nella cartella «js» e linkarla nella pagina dentro al tag HEAD:

```
<script type="text/javascript" src="js/jquery-3.4.1.min.js"></script>
```

NB: ATTENZIONE! Il link alla libreria jQuery deve precedere tutti gli altri in quanto contiene tutte le definizioni da utilizzare.
In caso contrario i nostri script tenteranno di utilizzare oggetti e funzioni non ancora definiti.

Pagina finale



… si provi ad inserire 500 come argomento del comando *toggle* …