Università degli Studi di Bologna

Facoltà di Ingegneria

Tecnologie Web T

A.A. 2019 – 2020

Esercitazione 05

DHTML, DOM, Javascript

Agenda

- Alcune premesse “pratiche”
  - progetto d'esempio
  - risorse statiche e cache dei browser
- Javascript e le applicazioni Web
  - semplici esempi di utilizzi ricorrenti
  - alcuni esercizi da svolgere in autonomia
  - accesso alle stesse risorse tramite Web Server
  - esempi di utilizzo delle API JQuery come alternativa al linguaggio nativo JavaScript e al framework Bootstrap per facilitare e velocizzare il design e l’implementazione di pagine Web
  - esercizio guidato tratto dall’appello d’esame del 26 gennaio 2018: HTML, CSS e Javascript

Progetto di esempio

- All'interno del file ***05_TecWeb.zip*** trovate lo scheletro di un semplice progetto di esempio
  - creato con Eclipse, contiene già tutti i descrittori necessari a essere riconosciuto e configurato correttamente

- Non importare il progetto in Eclipse, ma scompattare il file zip direttamente su filesystem
  - nella directory *web/pages* le pagine html con codice Javascript da testare
  - nella directory *web/scripts* un po' di codice Javascript

Prima di iniziare

- I browser “fanno cache” del contenuto statico
  - pagine html, script js, fogli di stile css, immagini gif/png/jpeg/jpg/...
  - infatti, normalmente non accade che tali risorse cambino tra due invocazioni successive

- Quando si sviluppa una applicazione, questo comportamento è invece fastidioso e bisogna cercare di ovviare
  - aggiungere i seguenti meta attributi nell'header delle pagine che si stanno modificando:

    <meta http-equiv="Pragma" content="no-cache"/>

    <meta http-equiv="Expires" content="-1"/>
  - tenersi sempre pronti a svuotare la cache del browser quando non si osservano gli effetti o le modifiche sperati (anche se si usano i meta attributi)
  - impostare (se possibile) il browser affinché NON usi la versione in cache delle risorse già scaricate

Browser detection - *whoYouAre.html*

- La pagina *whoYouAre.html* effettua browser detection
  - apritela con semplice
    - ad esempio, dalla libreria Javascript *jQuery* , di cui vedremo alcuni esempi di funzionamento a breve… ☺
- Disattivate ora l'uso di Javascript nelle preferenze del browser e aggiornate la pagina
  - differenze?
  - ora riattivate Javascript

Definizione e invocazione di funzioni - *helloworld.html*

- Invocazione di funzioni man mano che il browser processa, effettua il rendering e interpreta il sorgente della pagina
  - le funzioni invocate, `alert()` e `confirm()`, sono predefinite
  - la loro definizione e implementazione è fornita dall'interprete Javascript presente nel browser
- La definizione di una funzione non ne comporta l'immediata esecuzione, ma la rende disponibile nel proseguimento della pagina, ai fini dell'invocazione:
  - dall'interno di uno script *in linea* nella pagina
  - dall'interno di altro codice Javascript
  - associata ad eventi DHTML
- in evidenza:
  - i messaggi popup interrompono il rendering
    - diversi browser, tuttavia, tentano di schedulare il rendering e l'esecuzione di codice Javascript come processi paralleli, quando possibile
    - provate (a casa su) Firefox per farvi un'idea delle differenze di comportamento che emergono
  - l'uso dell'alternanza tra apice singolo ' e virgolette “ permette di scrivere HTML ben formato anche all'interno del codice Javascript, senza utilizzare escaping
    - apice singolo e virgolette sono entrambi delimitatori di stringa validi in Javascript e **all'interno di una coppia di delimitatori di un tipo si possono usare quelli dell'altro tipo** senza bisogno di escaping

Interazione tra Javascript e il DOM - *javascript-dom.html*

- L'esecuzione di una funzione che interagisce con il contenuto visualizzato deve...

attendere il completamento

del caricamento del contenuto

...da parte del browser

- Sembra banale, eppure...

Reazione ad eventi DHTML - *javascript-events.html*

- Nella pagina *javascript-events.html*
  - l'evento ***onload*** è sfruttato per agganciare funzioni Javascript a parti del DOM, in corrispondenza degli eventi DHTML da essi supportati
  - un'altra funzione (definita nella pagina stessa) viene associata ad un evento DHTML direttamente nel tag interessato

- In evidenza
  - all'interno del codice di una funzione Javascript è possibile definire nuove funzioni ed assegnarle

In Javascript, infatti, le funzioni sono oggetti di prima classe

  - non c'è alcuna differenza in Javascript tra un numero, una stringa e una funzione
  - tutti questi tipi di dati possono essere passati come argomento, memorizzati in variabili e restituiti come valore di ritorno di una funzione

Primi problemi (1) - *rollover.html*

- L'effetto di evidenziazione al passaggio del mouse non sempre è ottenuto modificando il colore di sfondo e/o del font
  - spesso si usano piccole immagini, sostituite tra loro in corrispondenza degli eventi relativi al mouse
  - l'effetto prende il nome di 'roll over'
- Nella pagina rollover.html sono presenti 4 modi diversi di ottenere lo stesso effetto
  - tramite CSS e pseudoclassi
  - tramite DOM e DHTML
  - tramite funzioni Javascript
- ma.... l'ultima funzione non va!
  - a differenza del codice JSP (compilato), quello Javascript è interpretato e la presenza di errori si scopre solo al momento di eseguire il codice stesso
  - infatti, non ci siamo accorti del problema finché non abbiamo eseguito (o, almeno, provato ad eseguire) la funzione incriminata
  - inoltre, il Servlet Container fallisce la traduzione e compilazione in servlet e ci restituisce un errore (500) e il relativo stackTrace dove guardare: il browser, invece, prosegue “a testa bassa” se qualcosa non va e semplicemente "inibisce" le parti di codice non corrette

Primi problemi (2) - *rollover.html*

- Sì, ma... e adesso? un indizio sul problema?
  - ore e ore a spulciare il codice
  - uso di milioni di `alert()` per mostrare popup in cui effettuare il log dello stato di variabili e oggetti per capire dove nasce il problema
  - usare un tool di sviluppo più avanzato!
- È il momento di Chrome Developer Tools (…o Firebug):
  - logging dei messaggi di errore Javascript su una specie di console di stderr
  - esecuzione del codice Javascript in modalità debug
  - ...e molto di più
- Ctrl + Shift + I per attivare Chrome Developer Tools
  - interagire con la pagina per lanciare l'esecuzione della funzione sbagliata e leggere i messaggi di errore nella console
  - cosa correggere?
  - una volta che la funzione esegue, collocare breakpoint al suo interno e seguirne l'esecuzione in modalità debug
    - Per collocare un breakpoint nello script Javascript è necessario accedere al codice tramite la scheda “Sources” di Chrome Developer Tools

Validazione di form lato client - *validate.html*

- Tipico utilizzo di Javascript per evitare di...
  - formulare HTTP request (magari anche pesanti) destinate a fallire
  - evitare di consumare banda
  - scrivere pagine più “responsive” e migliorare l'esperienza dell'utente
- Accesso agli elementi di un form via DOM e verifica di correttezza: seguiamo insieme l'esecuzione in modalità debug
  - apertura di Chrome Developer Tools, tab “Sources”, selezione del sorgente Javascript che interessa
  - collocazione dei breakpoint
  - interazione con la pagina per scatenare l'invocazione del codice Javascript (click sul pulsante, nel nostro caso)
- In evidenza
  - "esternalizzazione" delle funzioni Javascript in appositi file ***.js***, richiamabili da più pagine
  - è possibile (e anzi è prassi comune) utilizzare i valori booleani restituiti dalle funzioni Javascript per bloccare o permettere l'effettiva submission di un form

Modifica della visibilità del contenuto - *hideAndSeek.html*

- Un tipico effetto grafico per ottenere contenuto dinamico all'interno di una pagina
  - tutto il contenuto informativo è scaricato all'atto della richiesta, ma solo una parte è immediatamente visibile
  - attraverso gli eventi DHTML scatenati dalle azioni dell'utente si aggancia l'esecuzione di codice Javascript che modifica gli attributi di stile degli elementi interessati
    - *display: block;* → l'elemento è mostrato E occupa spazio nella pagina
    - *display: none;* → l'elemento NON è mostrato E NON occupa spazio
  - intervenendo sull'attributo
    - *visibility: hidden;*

    si sarebbe invece ottenuto l'effetto di nascondere l'elemento, ma lasciare vuoto lo spazio che esso avrebbe occupato

Stringhe ed interi - *calculator.html*

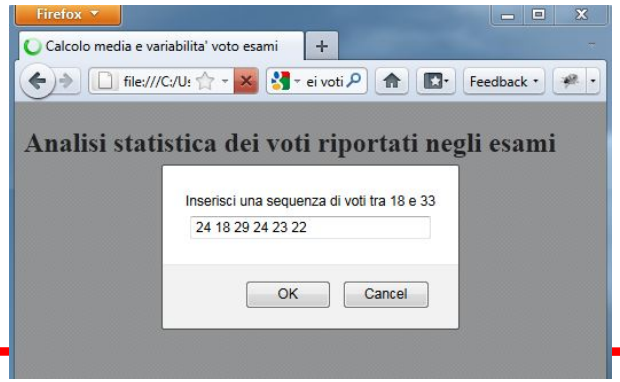
- Simile all'esempio presente nelle slide di teoria, serve qui a imprimere il concetto che **il browser e l'interprete Javascript che ne legge il DOM ragionano entrambi a stringhe**
  - non è specificato che un campo di input testuale sia destinato all'immissione di numeri
  - se non viene forzato a “valutare” l'espressione rappresentata dall'input testuale, l'interprete Javascript ne considera il valore come stringa

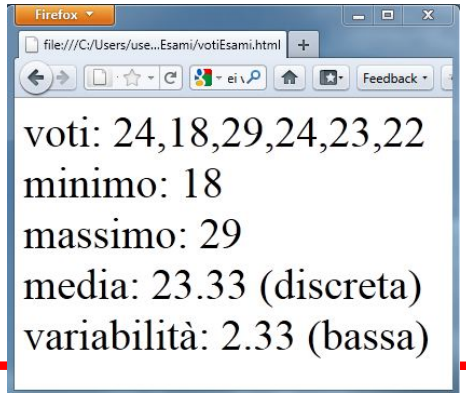
- Come sistemare?
  - sbirciate nel codice...
  - ovviamente non aprendo i file, ma con Chrome Developer Tools

Calcolo media e variabilità voto esami

Realizzare una pagina HTML dinamica basata su Javascript che, data una sequenza di voti di lunghezza non nota a priori, ne restituisca la media e la variabilità

- Ciascun voto deve essere un numero compreso tra 18 e 33: controllare opportunamente i dati in input
- La pagina restituita deve specificare
  - elenco dei voti inseriti
  - voto minimo e voto massimo
  - media aritmetica e giudizio sintetico (6 fasce tra 18 e 33)
  - variabilità come “errore medio” e giudizio sintetico (4 fasce tra 0 e 7.5)
- N.B. La soluzione deve basarsi sulla definizione di un oggetto *Statistica* contenente proprietà e metodi necessari alla computazione richiesta





Gestione dinamica componenti HTML

Realizzare una pagina HTML + Javascript che presenti

- In alto un **l'orario** che si aggiorna una volta al secondo
  - *onLoad()* per invocare ogni 1000ms una funzione di aggiornamento

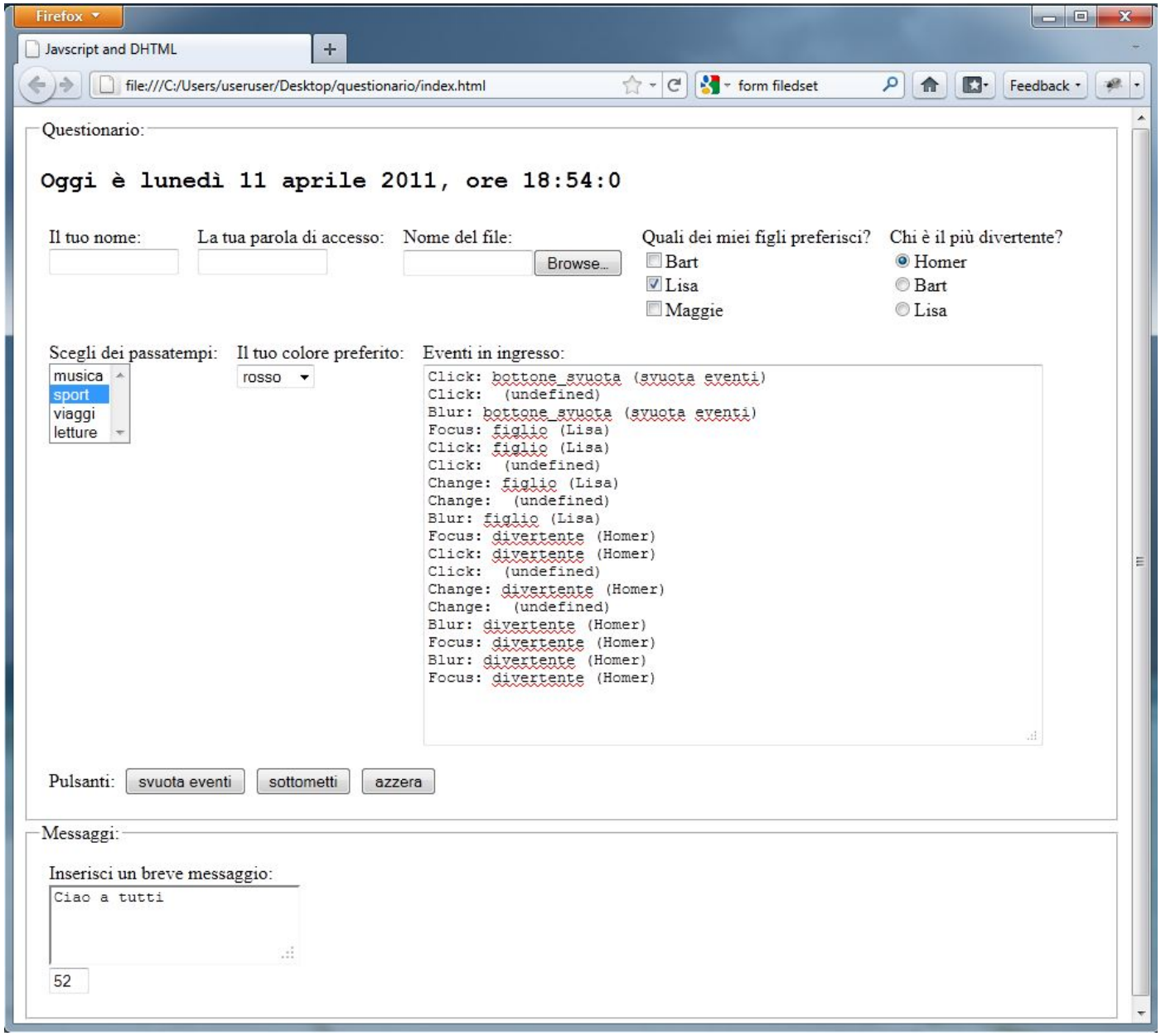
- In mezzo un **form** coi seguenti elementi HTML: input di testo, password, file, checkbox, radio, tendina scelta multipla, tendina scelta singola, button/submit/reset
  - realizzare una funzione Javascript che visualizzi in un'area di testo informazioni relative agli eventi associati ai precedenti elementi

    1) recuperare **dinamicamente** l'elenco degli elementi del form

    2) per ciascun elemento registrare una funzione agli eventi di tipo *onclick*, *onchange*, *onfocus*, *onblur*, *onselect* e *ondblclick*

- In basso un'**area di testo** con al più 64 caratteri
  - il colore dell'area cambia quando il mouse ci passa sopra (*onmouseover*/*onmouseout*)
  - un altro campo di testo specifica i caratteri ancora a disposizione (*onKeyUp*)
  - se si supera il limite, i caratteri in eccesso vengono eliminati

Gestione dinamica componenti HTML: esempio



Deployment su Tomcat

- Come dicevamo... all'interno del file ***05_TecWeb.zip*** trovate lo scheletro di un semplice progetto di esempio
  - creato con Eclipse, contiene già tutti i descrittori necessari a essere riconosciuto e configurato correttamente

- Ora importare il progetto come visto nelle precedenti esercitazioni
  - *File → Import → General → Existing Projects into Workspace → Next → Select archive file*

- Eseguirne il deployment su Tomcat ed accedere all'applicazione
  - *http:/localhost:8080/05_TecWeb/*

Pagina di benvenuto – *welcome.jsp*

- Struttura a *frame*...
  - giusto per sapere che esistono: tutti i maggiori browser li supportano, **ma il loro uso è deprecato**
  - non così l'uso di ***iframe***, invece, data la necessità di poter eseguire richieste cross-domain

– Il codice di *welcome.jsp* comporta la visualizzazione della pagine *whoYouAre.html* nel frame con nome *“content”*

Differenze con gli esempi precedenti

- Ora lavoriamo sul progetto d'esempio utilizzando il **browser**, il **Web Server** ed il protocollo **HTTP**

- Necessario in quanto
  - ...alcune pagine del progetto (i menu, la pagina con i frameset, ...) sono basate su tecnologia JSP e richiedono un Servlet Container per funzionare
  - ...ma **le pagine oggetto degli esempi Javascript** (cioè quelle all'interno della directory *pages*) **richiedono soltanto la presenza dell'interprete Javascript** per poter essere visualizzate
    - sono infatti risorse statiche con estensione **.html**
    - richiamano altre risorse statiche (immagini **.gif/.png**, script **.js**, fogli di stile **.css**) attraverso path relativi da cui il browser ottiene URL completi basandosi sull'URL della pagina visualizzata
    - **ecco perché potete aprire ciascuna di esse direttamente nel browser**, utilizzandone l'URL su file system (file://...), senza lanciare il Web Server e accedere via protocollo http (http://...)

APPENDICE

(esempi guidati sull'uso delle API JQuery ☺)

JQuery in a nutshell

- Breve introduzione a JQuery
  - JQuery vs. JavaScript
  - JQuery: Oggetti
  - JQuery: Selettori
  - JQuery: Css, Eventi e modifica del DOM

- Esempi d'uso delle API JQuery
  - Color
  - TODO List
  - Events Delegation

JavaScript vs. JQuery

- …come già sappiamo dalla teoria, JQuery è:
  - una libreria JavaScript (API)
  - essendo scritto in JavaScript tutto quello che permette di fare JQuery è possibile farlo anche solo con JavaScript
  - favorisce la compatibilità cross-browser che spesso JavaScript fatica a trovare
  - permette la manipolazione di CSS, del DOM, l'aggiunta dinamica di eventi e tante altre cose…

- JQuery è finalizzato a:
  - Rendere più semplice lo sviluppo di interfacce grafiche
  - Sviluppo semplice di applicativi lato Client
  - Aggiungere con minimo sforzo effetti altrimenti complicati da ottenere
  - Snellire il codice JavaScript e renderlo più sintetico

Per un pugno di $

- In JQuery tutto ruota attorno all'oggetto/funzione **$**, abbreviazione o alias di ***JQuery***
  - l $ è l'operatore principe di selezione, in particolare si fa quasi tutto attraverso i parametri della funzione **$()**
  - La caratteristica più utile ed apprezzata di JQuery è il suo motore di selezione
    - *$('nome_tag') restituisce un oggetto JQuery contenente tutti i tag di tipo nome_tag*
    - *$('.nome_class') restituisce un oggetto JQuery contenente tutti gli oggetti del DOM appartenenti alla classe nome_class*
    - *$('#nome_id') restituisce un oggetto JQuery contenente tutti gli oggetti del DOM con id nome_id*
    - *$('[nome_attr=val_attr]') restituisce un oggetto JQuery contenente tutti gli oggetti del DOM aventi l'attributo nome_attr che vale val_attr*
  - Come vedremo negli esempi, ci sono molti tipi di selettori, per gerarchie, componibili e combinabili, per posizione, ecc.. mentre in JavaScript solo document.getElementById, document.getElementByTagName, document.querySelector (recente)

JQuery: selettori (i)

- Selettori principali offerti da JQuery

Selector	Example	Selects
*	$("*")	All elements
#id	$("#lastname")	The element with id="lastname"
.class	$(".intro")	All elements with class="intro"
.class,.class	$(".intro,.demo")	All elements with the class "intro" or "demo"
element	$("p")	All <p> elements
el1,el2,el3	$("h1,div,p")	All <h1>, and <p> elements

:first	$("p:first")	The first <p> element
:last	$("p:last")	The last <p> element
:even	$("tr:even")	All even <tr> elements
:odd	$("tr:odd")	All odd <tr> elements

JQuery: selettori (ii)

[*attribute*]	$("[href]")	All elements with a href attribute
[*attribute=value*]	$("[href='default.htm']")	All elements with a href attribute value equal to "default.htm"
[*attribute!=value*]	$("[href!='default.htm']")	All elements with a href attribute value not equal to "default.htm"
[*attribute$=value*]	$("[href$='.jpg']")	All elements with a href attribute value ending with ".jpg"
[*attribute|=value*]	$("[title|='Tomorrow']")	All elements with a title attribute value equal to 'Tomorrow', or starting with 'Tomorrow' followed by a hyphen
[*attribute^=value*]	$("[title^='Tom']")	All elements with a title attribute value starting with "Tom"
[*attribute~=value*]	$("[title~='hello']")	All elements with a title attribute value containing the specific word "hello"
[*attribute*=value*]	$("[title*='hello']")	All elements with a title attribute value containing the word "hello"

parent > child	$("div > p")	All <p> elements that are a direct child of a element
parent descendant	$("div p")	All <p> elements that are descendants of a element
element + next	$("div + p")	The <p> element that are next to each elements
element ~ siblings	$("div ~ p")	All <p> elements that are siblings of a element
:eq(*index*)	$("ul li:eq(3)")	The fourth element in a list (index starts at 0)
:gt(*no*)	$("ul li:gt(3)")	List elements with an index greater than 3
:lt(*no*)	$("ul li:lt(3)")	List elements with an index less than 3
:not(*selector*)	$("input:not(:empty)")	All input elements that are not empty

JQuery: metodi … alcuni

- JQuery essendo una Libreria, offre metodi e proprietà
  - $(#menu li).size();
  - $(#menu li).lenght; (sintassi array, più veloce)
- Per ottenere elementi da una lista:
  - $("#menu li").get(0);
  - In JavaScript -> document.getElementById("menu").getElementsByTagName("li");
- Per ottenere l'html:
  - $("#menu li").get(0).innerHTML; // con JavaScript nativo
  - $("#menu li").eq(0).html(); // con metodo jQuery
- Per lavorare con gli attributi:
  - $("a#mioLink").attr("href"); restituisce il valore di href
  - $("a#mioLink").attr("href","http://www.html.it"); imposta il valore di href
  - $("a#mioLink").attr("href",function () { ... }); imposta il valore di href in base alla funzione
  - $("#menu li a").removeAttr("target"); rimuove un attributo
- Per lavorare con testi ed html
  - $("p").text("Nuovo testo");
  - $("p").html("Nuovo testo con HTML");

JQuery: impostare CSS

- Con JQuery, una volta ottenuti degli oggetti, è possibile impostargli dei css, anche qui con la stessa semantica di getter e setter:
  - $("a").css("color"); restituisce il colore esadecimale del primo elemento linkla soluzione alla seconda esercitazione (*seconda-esercitazione.html*)
  - $("a").css("color","#FF0000"); //imposta il colore dei link
  - $("a").css({
    "color" : "#FF0000", //imposta il colore
    "display" : "block" // imposta la visualizzazione
    });
- Impostare il colore di sfondo:
  - $("a").css("background-color","#FF0000");
  - $("a").css("backgroundColor","#FF0000");

Metodi	Descrizione
.width() **.height()**	permette di trovare o impostare larghezza e altezza complessiva di un elemento in pixel
.innerWidth() **.innerHeight()**	larghezza e altezza interne di un elemento (include il padding, non conta bordi e margini)
.outerWidth() **.outerHeight()**	larghezza e altezza esterne di un elemento (conta bordi e padding, opzionalmente i margini passando l'argomento `true`)
.offset()	ritorna un oggetto con la distanza da sinistra e dall'alto dell'emento rispetto al documento
.position()	come `.offset()` ma le distanze sono relative al contenitore più prossimo
.scrollTop() **.scrollLeft()**	trova o imposta scroll dell'elemento rispetto al documento

JQuery: binding e gestione eventi

- JQuery permette all'associazione dinamica di eventi agli oggetti selezionati

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

- Ci sono due modi per fare il binding di eventi:
  - $("p").click(function(){// action goes here!!});
  - $("p").on("evento", (function(){// action goes here!!});
- Esempio di *multiple event assignment*

```
$("p").on({
    mouseenter: function(){
        $(this).css("background-color", "lightgray");
    },
    mouseleave: function(){
        $(this).css("background-color", "lightblue");
    },
    click: function(){
        $(this).css("background-color", "yellow");
    }
});
```

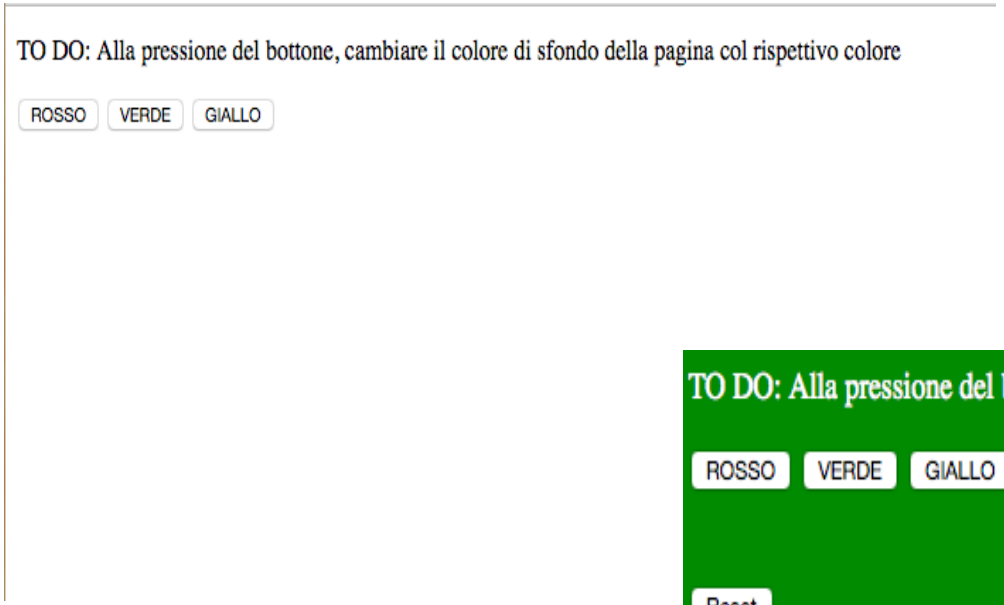
JQuery: modifica del DOM ed Event Delegation

- JQuery permette di modificare il DOM
  - $('body').append('<p ”>Testo a casa '); fa l’append del tag html al body
  - Esiste anche prepend che inserisce il contenuto prima dell’oggetto selezionato
- Altri metodi per la modifica del DOM
  - $(" Nuovo paragrafo "); *crea un nuovo elemento nell’oggetto jquery*
  - $("p").html("Testo del paragrafo"); fa la stessa cosa
  - $("
lista
").appendTo("#menu"); inserisco un nuovo nodo nell’elemento selezionato
  - ce ne sono molti altri…
- JQuery permette di fare una cosa molto comoda quando si aggiungono dinamicamente gli oggetti al DOM, ovvero la «delegazione di eventi»
- Si assegna un evento ad un nodo padre, ma lo si scatena solo quando l’utente interagisce con un nodo figlio. Esempio:
  - $(‘body’).on(‘click’, ‘button’, function(e){}); assegno la funzione di gestione dell’evento click al body, ma la scateno quando l’utente clicca bottoni contenuti nel body

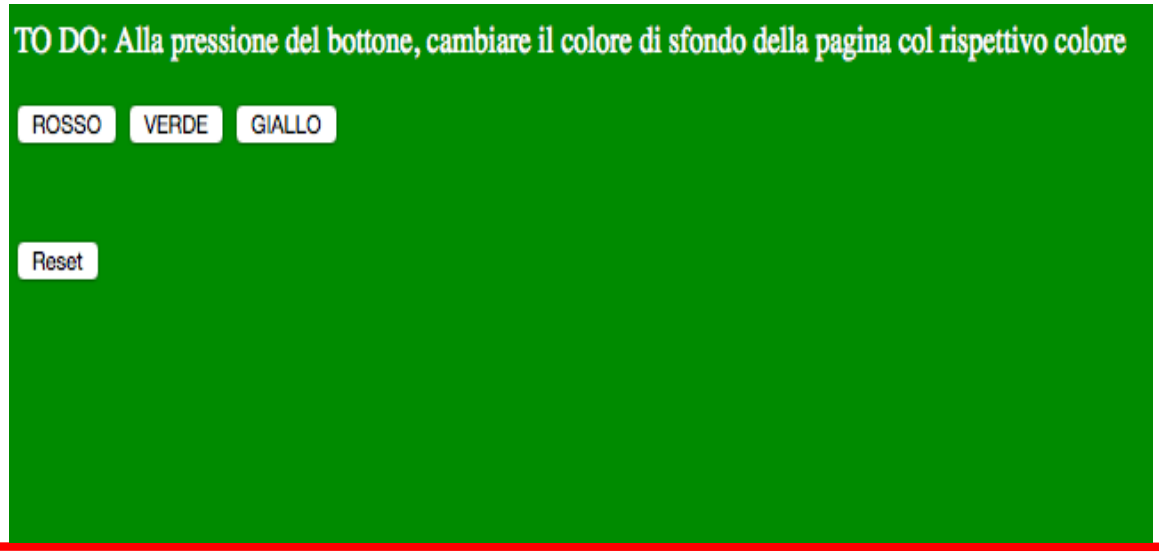
```
$('body').on('click', '[mio_attr="attr_val"]:first-of-type', function(e){
    // fai qualcosa
});
```

Esempio 1: «Colorize me»

- Il file color.html contiene un paragrafo con le istruzioni da implementare e 3 bottoni
- Utilizzando JQuery, alla pressione di uno di questi bottoni colorare lo sfondo del Body con il colore indicato da questi.

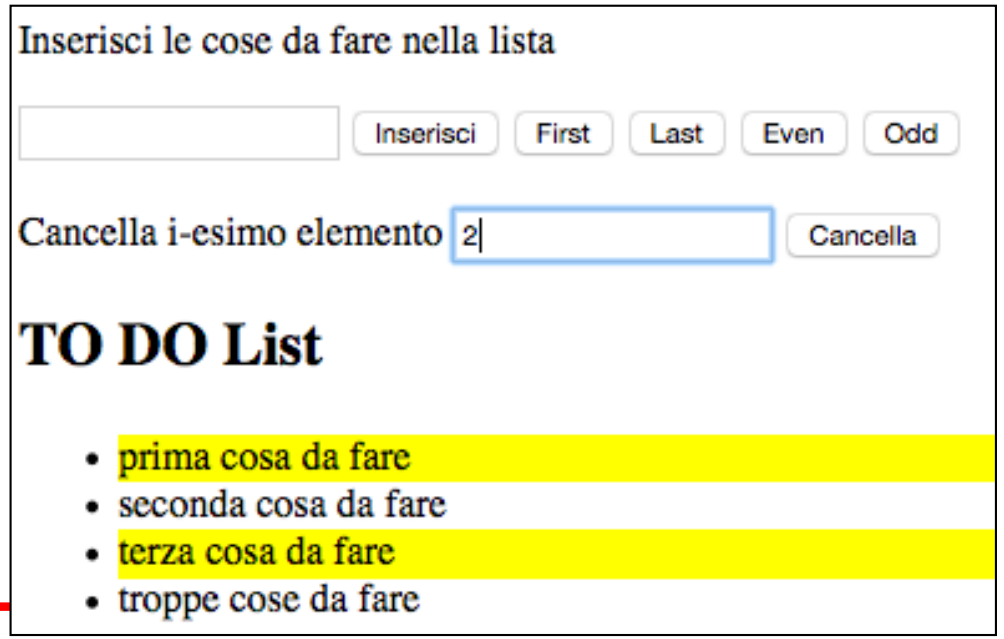


- Il risultato…



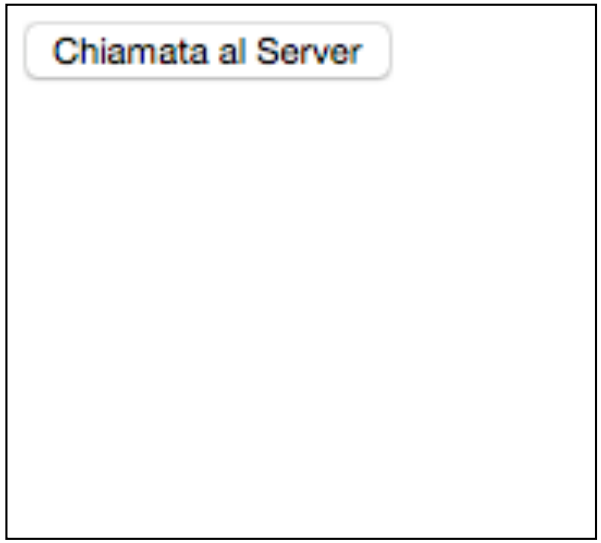
Esempio 2: «TODO list»

- File todo.html: creare una TODO list dinamica
- Attraverso JQuery inserire dinamicamente nella lista sottostante le cose da fare
- Aggiungere dei bottoni per la selezione:
  - del primo elemento della lista
  - degli elementi pari
  - degli elementi dispari
  - dell'ultimo elemento
- Infine inserire nel secondo campo di testo l'indice dell'elemento da eliminare ed eliminarlo
- Il risultato…



Esempio 3: «Randomize Color»

- File index.html: inserire dinamicamente alla pressione del bottone un paragrafo con scritto ‘Click’
- Al click si uno di questi paragrafi generare un colore casuale ed impostarlo come colore di sfondo
- Per generare un colore casuale
  - '#'+Math.floor(Math.random()*16777215).toString(16);



- ! ATTENZIONE: «Event Delegation»
- Il risultato…

APPENDICE

(introduzione a Bootstrap: facilitare e velocizzare design e implementazione di pagine Web ☺)

Cos’è Bootstrap

Bootstrap è un Framework per lo sviluppo di interfacce web, nato con l’obiettivo di facilitare e velocizzare il design e l’implementazione delle pagine web.

- Sviluppato originariamente da un gruppo di sviluppatori di Twitter per uniformare le interfacce e fornire modelli di design comuni all’interno della piattaforma
- Reso open source nel 2011
- Si basa su tecnologie **HTML**, **CSS**, e **JavaScript**
- Latest version: 4.4.1

Bootstrap: Highlights

- Bootstrap è compatibile con tutti i browser web nelle loro versioni più recenti
- Fornisce supporto nativo per sviluppare pagine fortemente **responsive**
- Essendo diventato usatissimo ed open source, ha una grande community alle spalle per il supporto
  - Ottima documentazione, tradotta in svariate lingue
- Tipico Page Design a griglia
- Sviluppo Live from scratch con BootstrapCDN (Content Delivery Network)
  - Inclusione dei link alle librerie remote direttamente nel codice, senza dover scaricare compilati o codice sorgente
    - Progetti più leggeri
    - Richiesta connessione Internet sempre attiva in fase di sviluppo

Struttura di Bootstrap 4

…Bootstrap non è solo un insieme di librerie CSS e JavaScript da includere nei nostri progetti..

Bootstrap è un Framework con una propria struttura sulla quale permette allo sviluppatore di creare le proprie pagine web

Bootstrap ha una **struttura a griglia**

- Permette un'impaginazione modulare e precisa (modello carta stampata)
- Si basa sul modello **Flexbox**
  - Specifica CSS mirata espressamente al layout
- La griglia è composta da 12 colonne
- Cinque misure differenti possibili
- Consente alta adattabilità
  - Responsiveness per dispositivi mobili (tablet e smartphone)

BOOTSTRAP 4

Device	Ems	Pixels	Class
Extra Small	Less than 34em	Less than 544px	col-xs
Small	34em and up	544px and up	col-sm
Medium	48em and up	768px and up	col-md
Large	62em and up	992px and up	col-lg
Extra-Large	75em and up	1200px and up	col-xl

Bootstrap 4: Grid System

Il Grid System di Bootstrap 4 è pensato per costruire ogni tipo di layout per forma e dimensione, assemblando un serie di tre tipologie di elementi:

- Container
- Row

One of three columns	One of three columns	One of three columns

Copy

```
<div class="container">
  <div class="row">
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
  </div>
</div>
```

Grid System: un po' di dettaglio (1)

- I Container permettono di centrare e riempire orizzontalmente i contenuti della pagina
  - .container: larghezza in pixel responsive a seconda dello schermo
  - .container-fluid: per un riempimento della pagina al 100% su qualsiasi dimensione di schermo
- Le Row fanno da wrapper per le classi Column
- Ogni Column ha:
  - Padding orizzontale (gutter): Serve per controllare lo spazio tra colonne. Il gutter viene poi mitigato da margini negativi sulle Row, allineando tutto il contenuto delle colonne sul lato sinistro
- Gli elementi della pagina devono essere messi dentro le Column
  - Solo le Column possono essere figlie delle Row
- Il Grid System fornisce l'auto-layout: Column senza larghezza specificata saranno larghe uguali
- La classe delle Column indica il numero di colonne che si vogliono creare su un massimo di 12. Per esempio se si vogliono creare 3 colonne di ugual misura si dovrà usare .col-4.
- La larghezza delle Column son in percentuale, fanno riferimento all'elemento padre

Grid System: un po’ di dettaglio (2)

La resposiveness della pagina è realizzata grazie a dei **Grid Responsive Breakpoint**

- Sono 5: extra-small, small, medium, large, extra-large
- Sono dei punti di interruzione «sensibili» del nostro layout che ci permettono di gestire al meglio il ridimensionamento dello schermo
- Sono basate su larghezze minime di schermo, e vengono applicate al layout a seconda della dimensione della vista della pagina

Esempio: Se si assegna ad un elemento la classe .col-sm-4, la dimensione verrà mantenuta per una dimensione small, medium, large, ed extra large dello schermo, ma non per gli schermi extra-small

Grid System: un po’ di dettaglio (3)

In Bootstrap vengono usati le unità di misura «tipografiche» per la maggior parte delle cose (em, rem). Per i container e i breakpoint si usano invece i pixel (px), per conformità con viste delle pagine e dimensioni degli schermi.

Di default viene usata la classe .col-sm-6

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

Grid System: Esempi (1) Equal-width

Griglia con colonne di larghezza uguale

1 of 2	2 of 2

1 of 3	2 of 3	3 of 3

Copy

```
<div class="container">
  <div class="row">
    <div class="col">
      1 of 2
    </div>
    <div class="col">
      2 of 2
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col">
      2 of 3
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
</div>
```

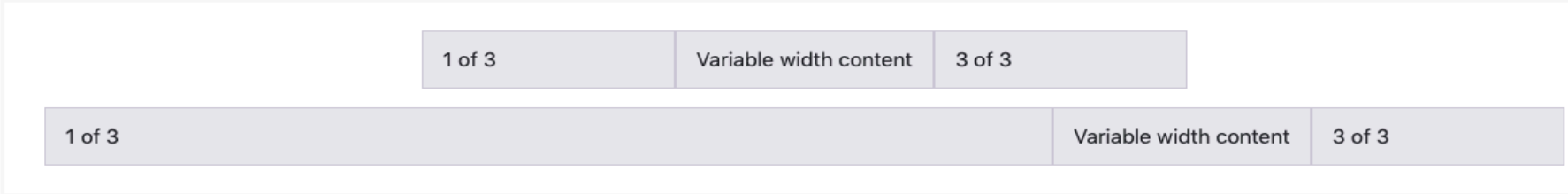
Grid System: Esempi (2) Wider Column

1 of 3	2 of 3 (wider)	3 of 3
1 of 3	2 of 3 (wider)	3 of 3

```
<div class="container">
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-6">
      2 of 3 (wider)
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-5">
      2 of 3 (wider)
    </div>
    <div class="col">
      3 of 3
    </div>
  </div>
</div>
```

Il layout automatico fornito dal modello Flexbox fa si che cambiando le dimensioni di una colonna le altre si ridimensionino automaticamente senza alcuna direttiva esplicita (al pari livello)

Grid System: Esempi (3) Content-size Column Width



Copy

```
<div class="container">
  <div class="row justify-content-md-center">
    <div class="col col-lg-2">
      1 of 3
    </div>
    <div class="col-md-auto">
      Variable width content
    </div>
    <div class="col col-lg-2">
      3 of 3
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 of 3
    </div>
    <div class="col-md-auto">
      Variable width content
    </div>
    <div class="col col-lg-2">
      3 of 3
    </div>
  </div>
</div>
```

La classe .col-{breakpoint}-auto fa si che la dimensione della colonna si adatti al suo contenuto

Bootstrap: Da Dove Partire (1)

Per prima cosa si devono includere le librerie CSS e Javascript nella pagine html.

Si può partire dal template vuoto fornito direttamente sul sito di Bootstrap

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3
+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj
1yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9I
OYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4
Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
  </body>
</html>
```

Bootstrap: Da Dove Partire (2)

E’ possibile partire dal template mostrato nella slide precedente oppure modificare uno dei tanti template ready-to-use messi a disposizioni da Bootstrap

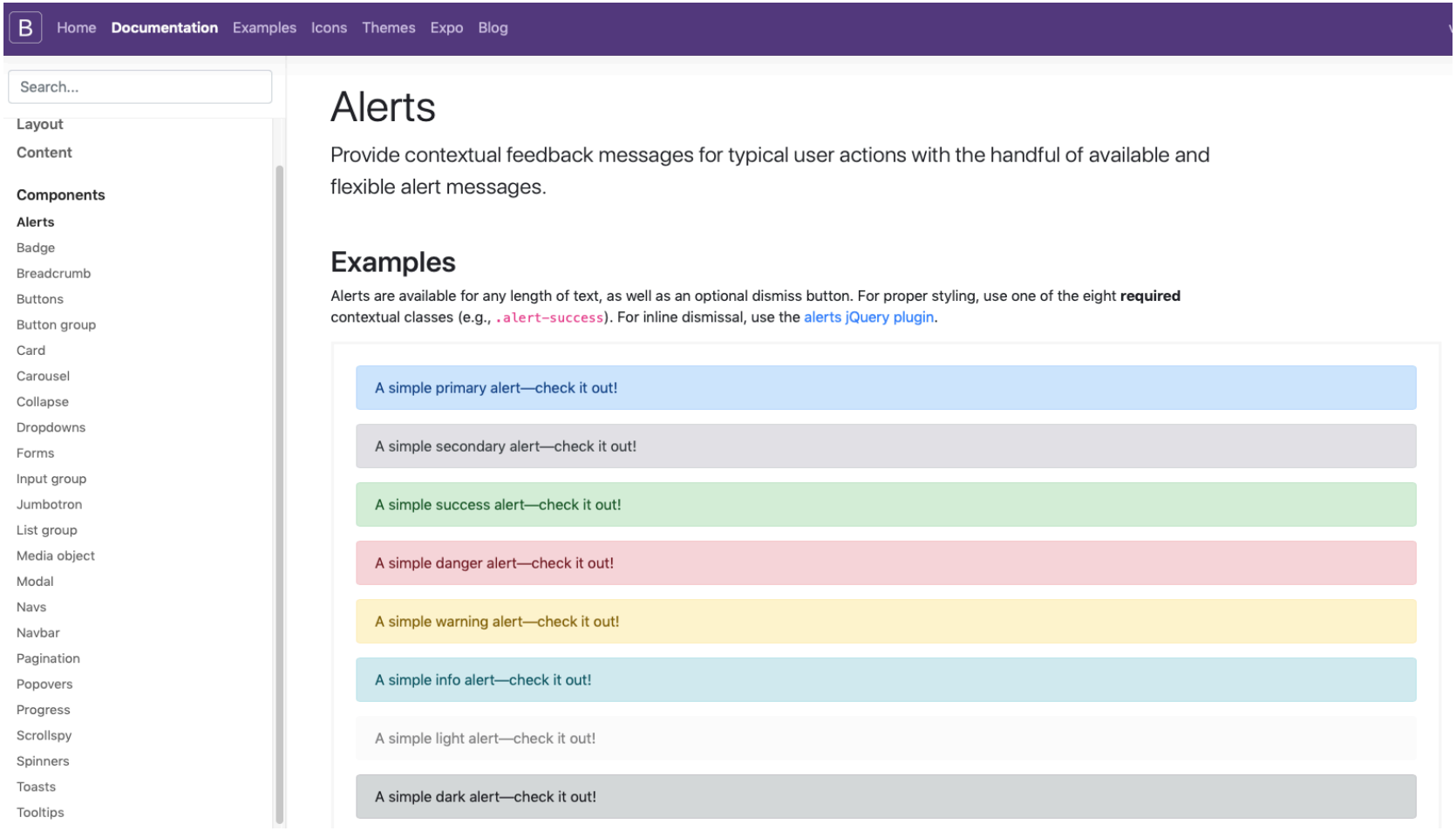
Una volta scaricato il template è possibile riempirlo con tutti i componenti già pronti messi a disposizione, oppure utilizzare le classi di Bootstrap per personalizzare un nostro progetto personale, o uno già pronto

Bootstrap ha riconosciuto alcuni «elementi notevoli», o componenti che vengono usati più spesso di altri nello sviluppo di un’applicazione web, e li mette a disposizione già pronti all’uso

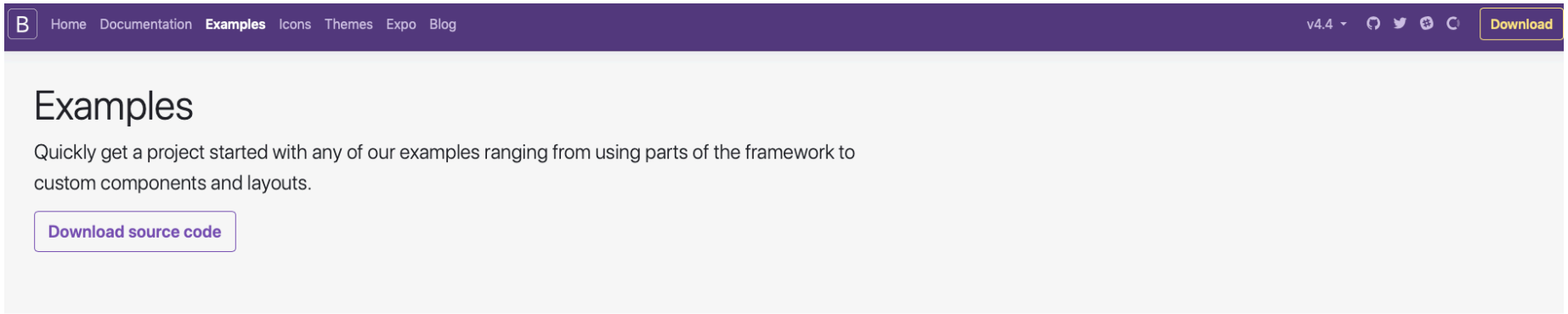
Bootstrap offre:

- Stili per ogni tipo di componente
- Stili per la tipografia dei testi
- Esempi open source di siti web
- Temi
- Icone

Bootstrap: Esempio Componenti

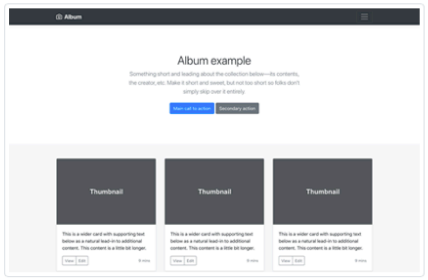


Bootstrap: Template



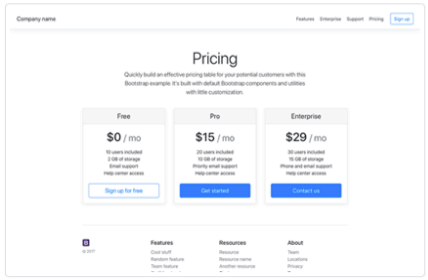
Custom components

Brand new components and templates to help folks quickly get started with Bootstrap and demonstrate best practices for adding onto the framework.



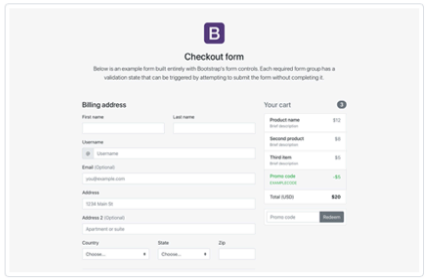
Album

Simple one-page template for photo galleries, portfolios, and more.



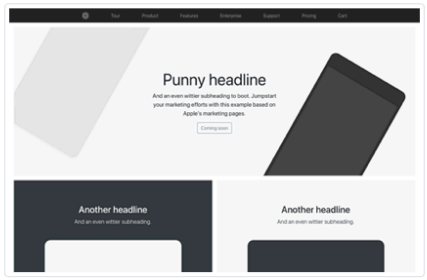
Pricing

Example pricing page built with Cards and featuring a custom header and footer.



Checkout

Custom checkout form showing our form components and their validation features.



Product

Lean product-focused marketing page with extensive grid and image work.

Bootstrap: Non è tutto rose e fiori

Tanta popolarità e tanti vantaggi per lo sviluppatore non significano, automaticamente, che Bootstrap sia la soluzione ottimale per ogni contesto e progetto.

Per esempio se la flessibilità e la manutenibilità del codice stesso sono priorità assolute, se il design del sito sul quale si sta lavorando deve avere i crismi dell'originalità, se deve essere alternativo e innovativo, allora non è bene usare Bootstrap

Se invece il design del sito che si sta o si vuole sviluppare è più convenzionale, se si cerca uno strumento da cui partire o un template solido per la strutturazione del progetto, allora Bootstrap è una delle scelte migliori.

Bootstrap Reference

Tutto il materiale mostrato è open source e disponibile al sito:

https://getbootstrap.com

Il sito fornisce un ampia e molto ben fatta documentazione contenente tutti i componenti, i template, i temi e le icone offerte dal Framework:

https://getbootstrap.com/docs/4.4/getting-started/introduction/

Il Framework è scaricabile da qui:

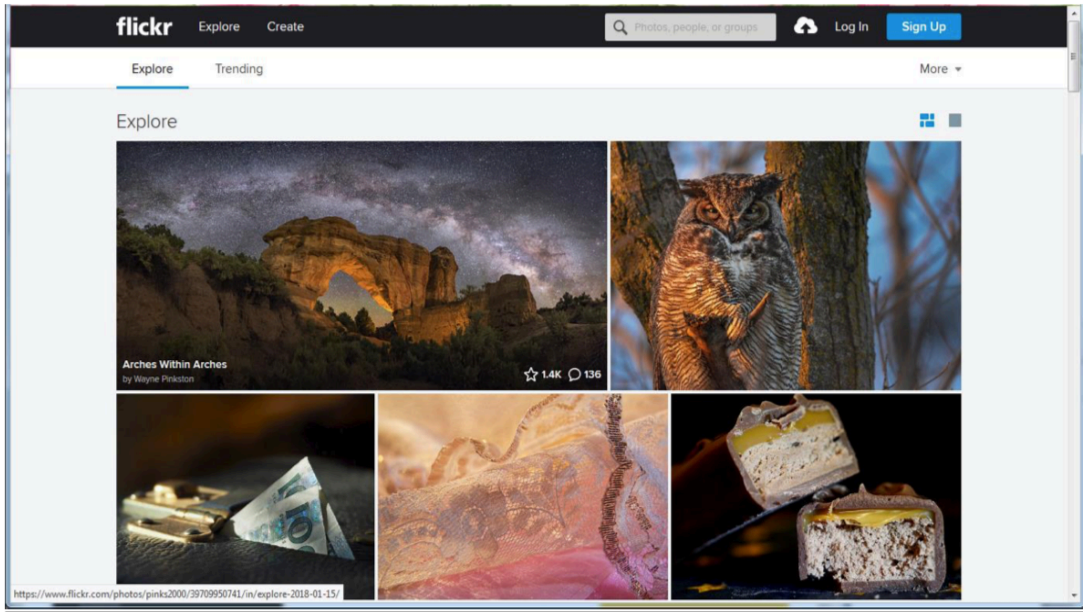
https://getbootstrap.com/docs/4.4/getting-started/download/

APPENDICE

(esercizio guidato tratto dall'appello d'esame del 26 gennaio 2018: HTML, CSS e Javascript)

Agenda

- Esercizio tratto **dall'appello d'esame del 26 gennaio 2018**
- Riproduzione secondo specifiche di una pagina Web, partendo da uno "snapshot" di riferimento:
  - **sito Web Flickr**
- Tecnologie da utilizzare:
  - HTML
  - CSS
  - Javascript



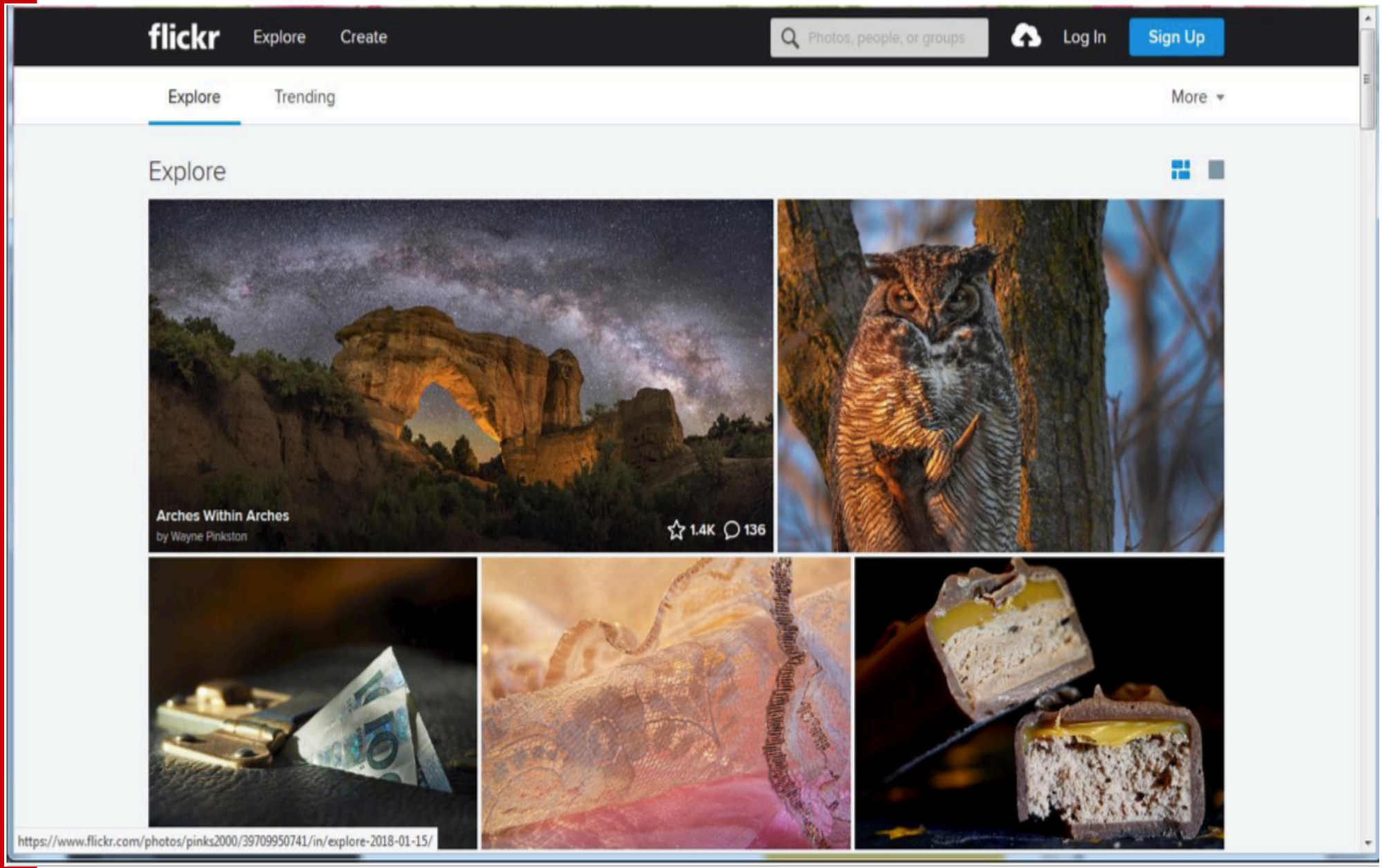
Esame 26/01/2018: Esercizio 2

ESERCIZIO 2 (11 punti)

Si realizzino le **pagine Web dinamiche** (basate su **tecnologia HTML**, **CSS**, e **Javascript**) in grado di **riprodurre il contenuto e il layout grafico** dello "snapshot" del **sito Web flickr** di seguito riportato. Nello specifico, la pagina visualizza il contenuto relativo alla voce "*Explore*" del Menù "orizzontale" nella barra grigio scuro in alto a sinistra, e ancora "*Explore*" tra le sotto-opzioni messe a disposizione nella barra bianca (la scelta è evidenziata dalla linea di sottolineatura azzurra della voce "Explore"), optando per la modalità di visualizzazione di immagini "a mosaico" piuttosto che di una singola immagine alla volta (evidenziata dal color azzurro della prima icona posta sul lato destro della finestra principale di sfondo grigio chiaro). In particolare, si deve prevedere che al passaggio del mouse su ogni immagine visualizzata nella finestra principale, si mostri nell'immagine stessa anche il titolo della fotografia assieme al nome dell'autore, sul lato sinistro, mentre sul lato destro si riporti il numero di *preferenze* ricevute (icona "stella") e il numero di *commenti* inseriti (icona "nuvola") dagli utenti, come evidenziato per la prima immagine della finestra principale.
Infine, si preveda che il campo di ricerca nella barra grigio scuro in alto possa ammettere solo stringhe alfabetiche (tutte maiuscole o minuscole) composte da una sola parola.

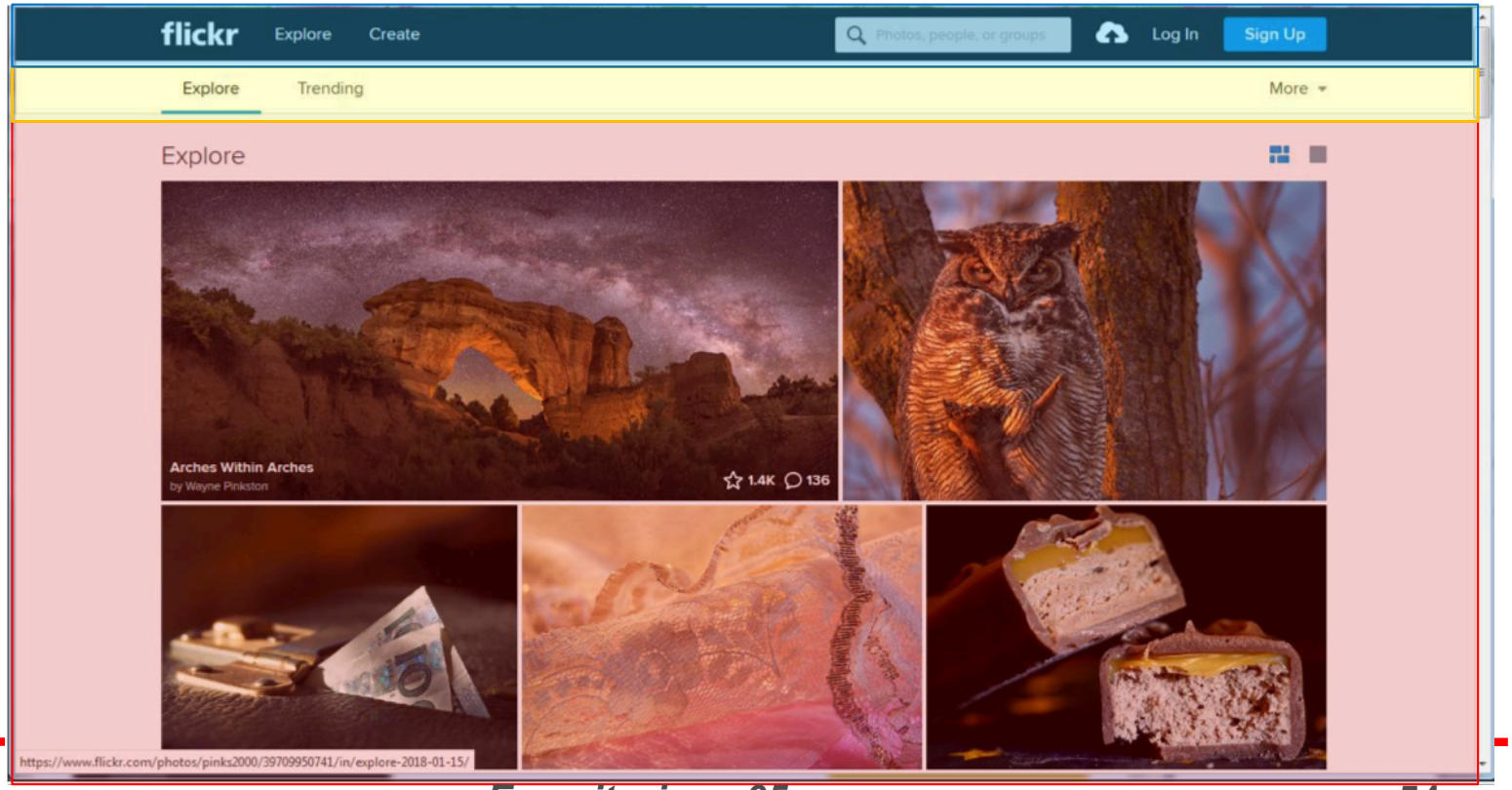
N.B. La soluzione NON deve far uso del costrutto HTML `frame`. Per rappresentare le immagini/icone riportate nello snapshot, si utilizzino figure di esempio a piacere.

“snapshot” del sito Web Flickr



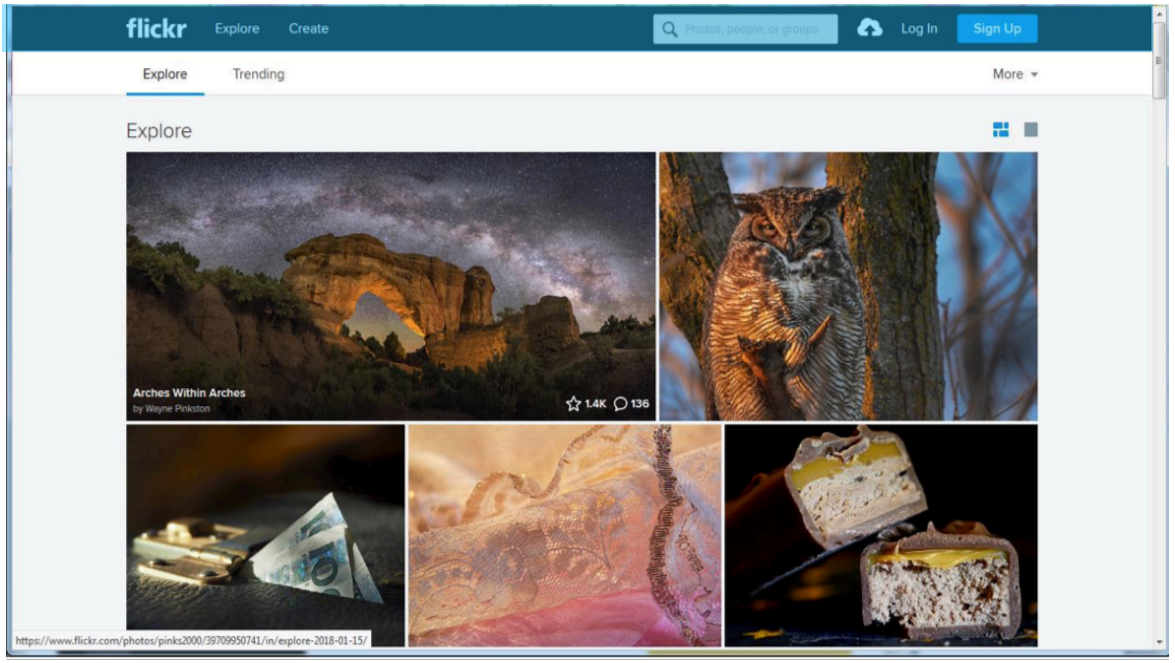
Step 1: Divide et Impera! (1)

- Il primo passo per affrontare questo tipo di esercizo è individuare la struttura dell'informazione da modellare, ovvero le "macro-aree" all'interno della pagina da riprodurre
  - Ripetere fino ad arrivare ai semplici componenti… un po' come il gioco delle scatole cinesi
    - Come vedremo, ci troveremo a fare `div` dentro `div`, dentro `div`, ecc.



Step 1: Divide et Impera! (2)

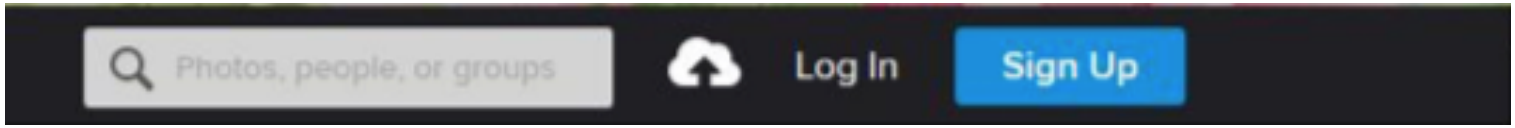
- Dividiamo ancora…



- Quali macro-aree possiamo individuare qui?

Step 1: Divide et Impera! (3)

- E ancora…



- A questo punto siamo arrivati a livello dei singoli componenti e abbiamo nell'ordine:
  - Un input di tipo text (sicuri? …lo vediamo fra un po')
  - Un bottone con un'immagine
  - Un bottone "invisibile"
  - Ed un bottone blu
- Stessa cosa andrà fatta per l'altra metà, e per le altre macro-aree individuate nella pagina

Hands-on

- Partiamo dall'inizio…

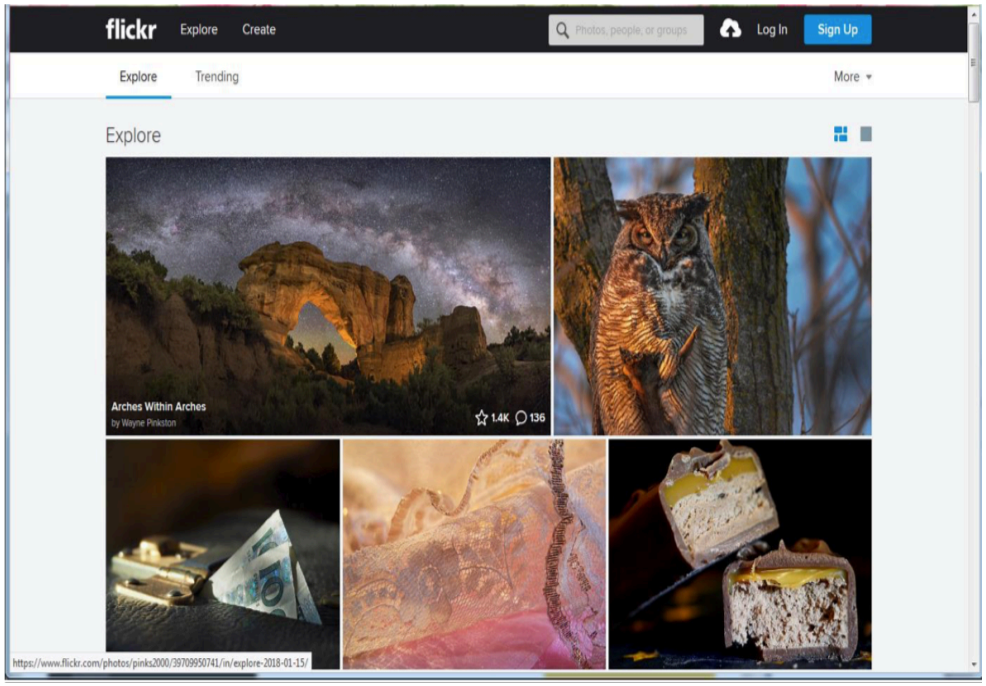
```
<body>
    <div id="container">

        <div class="header"> ··· </div> <!-- header -->

        <div class="topic"> ··· </div>


            <div class="main"> ··· </div> <!-- main -->


    </div> <!-- container -->
</body>
```




```
#container{
    width: 100%;
    height: 100%;
}
.header{
    width: 100%;
    height: 5%;
    background-color: #556675;
}
.topic{
    background-color: white;
    width: 100%;
    height: 5%;
}
.main
{
    width: 100%;
    height: 90%;
    background-color: rgb(243,245,246);
}
```

Hands-on: Div header (1)

- Dividiamo a sua volta la div header in due sotto div
  - Entrambe con altezza 100% del parent e come width il 50%
  - Le disponiamo una a destra e l'altra a sinistra
- Riempiamo le div appena create con I relativi componenti
  - La div di sinistra
    - Il logo di flickr
    - Un menù orizzontale con due voci
  - La div di destra
    - Un campo che sembra un input text con un'immagine…
    - Un bottone con immagine
    - Un bottone "invisibile"
    - Un bottone classico

Hands-on: Div header (2)

- Andiamo con ordine

```
<div class="header">
    <div class="navigation">
        <img class="imageSpace"  src="images/flickr_logo.png" alt="flickr logo" width="15%" height="70%">
        <ul class="menu" id="navMenu" style="padding-top: 0.5%;">
            <li ><a href="#">Explore</a></li>
            <li ><a href="#">Create</a></li>
        </ul>
    </div><!--#navigation-->
    <div class="search"> ··· </div> <!-- search -->
```

```
.menu {
    margin-top: 0px;
    margin-bottom: 0px;
}
.menu li {
    height: 100%;
    margin-top: 10px;
    display: inline;
    float: left;
    list-style: none;
    padding-right: 15px;
    padding-left: 15px;
}
li a {
    text-decoration: none;
    color: #FFFAF0;
    padding-bottom: 29%;
}
```

```
.imageSpace {
        padding-top: 0.8%;
        float: left;
}
```

! IMPORTANTE:

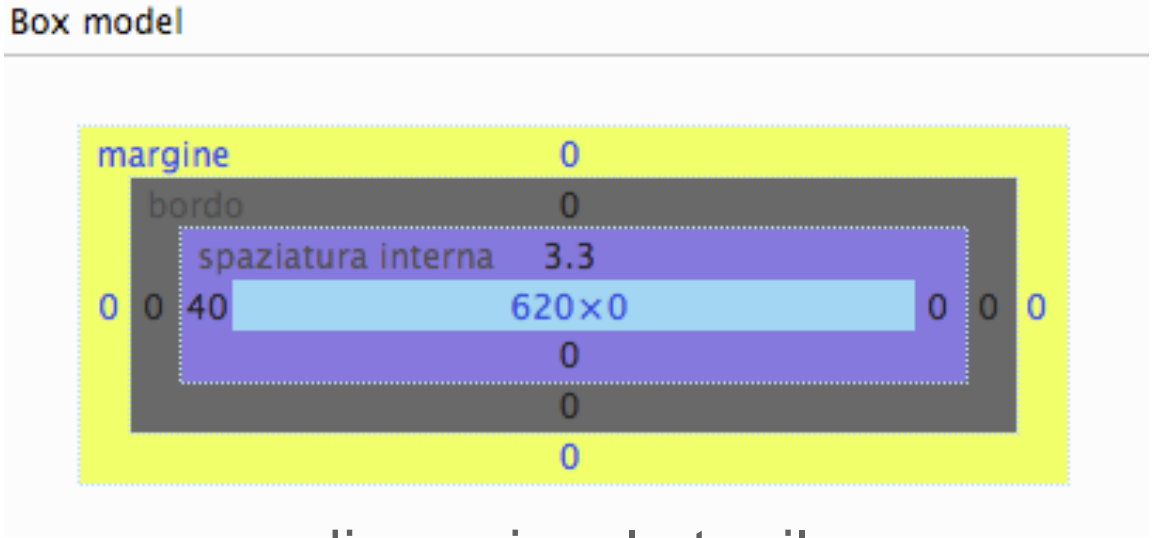
Proprietà Float

Con questa proprietà è possibile rimuovere un elemento dal normale flusso del documento e spostarlo su uno dei lati (destro o sinistro) del suo elemento contenitore

Proprietà display:inline sugli elementi **li**

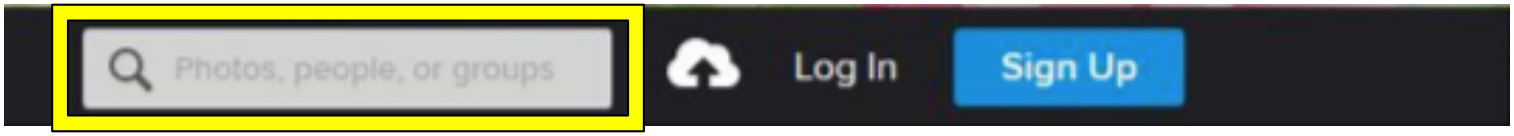
Padding e Margin

- Tutte le pagine web fanno riferimento al **Box Model**
- Per ottenere un effetto
- grafico efficace, occorre
- sfruttare le proprietà **margin** e **padding**,
- ma **attenzione**…



- Padding: modifica la distanza mono-dimenzionale tra il content ed il bordo dell'elemento (area lilla)
- Margin: modifica la distanza mono-dimenzionale tra il bordo dell'elemento e l'elemento successivo, o il container
- padding/margin-[left|top|right|bottom]: [px|%|preset]

Hands-on: Div header – right (1)



- Un input text non input text… in realtà è una `div` con un immagine ed un campo di testo
  - l'effetto "elemento unico" è ottenuto giocando con le dimensioni degli elementi, eliminando i

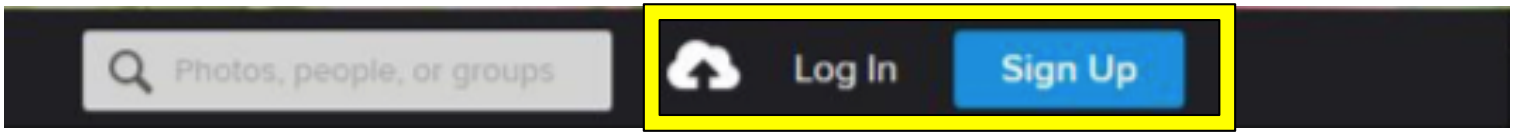
```
<div class="search">
    <div class="searchbar">
        <input type="image" class="searchImg" src="images/searchImg.png" >
        <input class="searchInput" id="searchStr" type="text" placeholder="Photo, People or group" onkeyup="checkString(event,this.value);"/>
    </div>
    <div class="barbuttons">
        <input type="image" class="upimg" src="images/upload.png" >
        <input type="button" class="loginButton" value="Log In"/>
        <input type="button" class="signupButton" value="Sign Up"/>
    </div>
</div> <!-- search -->
```

```
.search{
    height: 100%;
    width: 50%;
    float: right;
}
```

```
.searchbar{
    background: white;
    height: 55%;
    width: 40%;
    float: left;
    padding-top: 1%;
    margin-top: 1%;
}
```

```
.searchInput{
    height: 100%;
    width: 90%;
    padding-left: 0px;
    border-top-width: 0px;
    border-top-style: solid;
    padding-top: 0px;
    padding-bottom: 0px;
    padding-right: 0px;
    border-right-width: 0px;
    border-left-width: 0px;
    border-right-style: solid;
    border-bottom-width: 0px;
    border-bottom-style: solid;
    float: right;
}
```

Hands-on: Div header – right (2)




```
<div class="barbuttons">
    <input type="image" class="upimg" src="images/upload.png" >
    <input type="button" class="loginButton" value="Log In"/>
    <input type="button" class="signupButton" value="Sign Up"/>
</div>
```

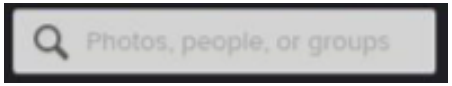
```
.barbuttons{
    height: 100%;
    width: 60%;
    float: right;
}
```

```
.upimg{
    height: 100%;
    width: 13%;
    margin-left: 3%;
    float: left;
}
```

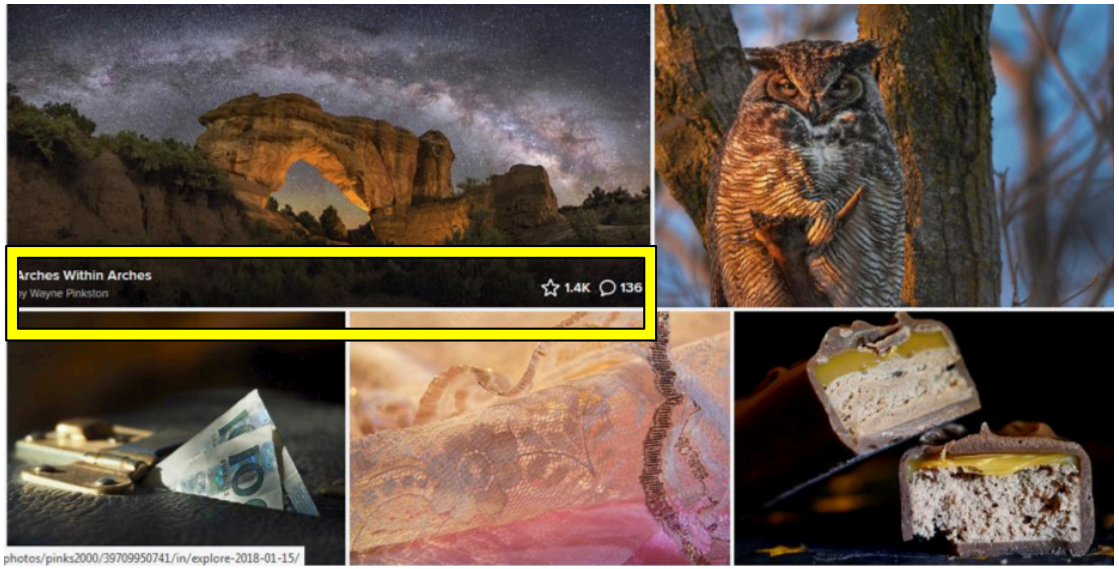
```
.loginButton{
    background-color: Transparent;
    color: white;
    height: 80%;
    width: 20%;
    margin-left: 3%;
    margin-top: 1%;
    border: 0;
}
```

```
.signupButton{
    background-color: rgb(0,145,214);
    color: white;
    height: 80%;
    width: 20%;
    margin-left: 3%;
    margin-top: 1%;
    border: 0;
}
```

Tricky elements

- La pagina da riprodurre presenta alcune parti da ricreare non banali:
  - L'input text della ricerca nella barra in alto (già visto) 

  - Il radio button con le icone per la disposizione delle foto 
  - La disposizione a mosaico delle immagini



  - La sovrapposizione dei dettagli dell'immane al `mouseover`

Tricky elements: Radio Button (1)

- Sono radio button particolari realizzati con icone personalizzate

```
<input type="radio" name="disposition" id="single" class="input-hidden" />
<label for="single">
    <img src="images/singleIconNoSel.png" alt="Single" class="dispositionImg" id="singleImg"/>
</label>
<input type="radio" name="disposition" id="grid" class="input-hidden" />
<label for="grid">
    <img src="images/gridIconNoSel.png" alt="grid" class="dispositionImg" id="gridImg"/>
</label>
```

- Ovviamente i due radio devono avere lo stesso **name** per operare in modalità esclusiva
- Utilizzeremo una **label** che contiene un'immagine per le icone
- Il collegamento tra immagine e radio button è realizzato tramite l'attributo **for** dell'elemento label

Attribute	Value	Description
for	*element_id*	Specifies which form element a label is bound to

- Siamo obbligati a dare degli id ai radio button, ovviamente diversi

Tricky elements: Radio Button (2)

- Ora che abbiamo modellato la “struttura” dell’informazione in html, usiamo css per la parte di layout grafico

```css
.input-hidden {
  position: absolute;
  left: -9999px;
}

input[type=radio]:checked + label>img {
  border: 1px solid rgb(0,145,214);
  box-shadow: 0 0 1px 1px rgb(0,145,214);
}
```

- Dobbiamo nascondere i veri radio button
  - Possiamo spostarli in modo absolute molto fuori lo schermo per esempio a -9999px
  - Oppure in modo più elegante e coinciso possiamo usare la proprietà Display: none;
- La seconda parte del css fa comparire i bordi dell’immagine dentro la label collegata al radio button checked

Tricky elements: visualizzazione mosaico (1)

- Ci sono diversi modi per realizzare un effetto del genere
- Il migliore in termini di pulizia ed eleganza è, come ci si può aspettare, la modellazione tramite `div` innestate
  - Abbiamo già visto come lavorare con le div e le loro dimensioni
  - Vediamo un altro approccio…
- L'approccio a tabella
  - L'idea è creare una tabella con il numero di righe richiesto e modellare con css il numero di colonne e la loro larghezza per ogni riga

Problema: In una table è si possibile impostare la larghezza delle colonne per riga, ma una volta sola; in altre parole la prima riga è personalizzabile, le altre ereditano lo stesso style

Escamotage: Più tabelle con una riga ciascuna

Tricky elements: visualizzazione mosaico (2)

```
<table class="mainTableDiv" style="height: 25%; width: 100%">
        <tr class="tableRow">
            <td class="tableRowContent" style="width: 70%;">
                <div class="tableContent" onmouseover="displayOn('overlap1');" onmouseout="displayoff('overlap1');" style="background-image:
                url('images/bologna.png'); background-repeat: no-repeat;">
                    <div class="filler">
                    </div> <!-- filler -->
                    <div class="overlap" id="overlap1"> ··· </div> <!-- overlap -->
                </div>  <!-- tableContent -->


            </td>

            <td class="tableRowContent" style="width: 30%;"> ··· </td>
        </tr>
</table>
```

Tricky elements: display singola immagine (1)

- Anche qui i modi per realizzare l'effetto sono molteplici

- La soluzione mostrata è realizzata con:
  - Una `div` che occupa tutta la cella della tabella con l'immagine come background
  - All'interno di questa altre due `div`
    - Una di filling, vuota, usata come riempimento
    - L'altra contenente tutti i componenti dei dettagli: nome, autore, commenti e likes
  - Al `mouseover` la seconda `div` compare
  - Al `mouseout` scompare
  - Le `div` interne hanno il background trasparente, per rendere l'immagine sempre visibile

Tricky elements: display singola immagine (2)

```
<div class="tableContent" onmouseover="displayOn('overlap1');" onmouseout="displayoff('overlap1');" style="background-image:
url('images/bologna.png'); background-repeat: no-repeat;">
    <div class="filler">
    </div> <!-- filler -->
    <div class="overlap" id="overlap1">
        <div class="photoText">
            <p class="pTitolo">werwerwerwewe</p><p class="pAutore">asdasdasdasd</p>
        </div> <!-- photoText -->
        <div class="photoLikeComment">
            <div class="like">
                <input type="image" class="likeImg" src="images/starIcon.png" ><p class="pLike">12</p>
            </div> <!-- like -->
            <div class="comment">
                <input type="image" class="likeImg" src="images/comicIcon.png" > <p class="pLike">54</p>
            </div> <!-- like -->
        </div> <!-- photoLikeComment -->
    </div> <!-- overlap -->
</div>  <!-- tableContent -->
```

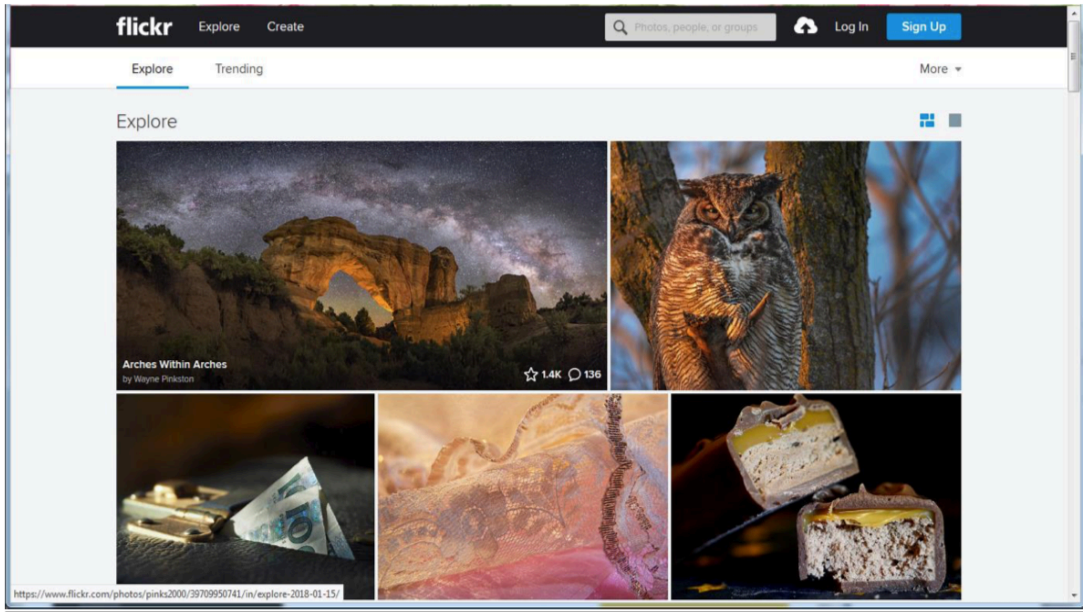
```
.tableContent{
    width: 100%;
    height: 100%;
    background-size: 100%;
}
.filler{
    height:80%;
    width: 100%;
    background-color: Transparent;


}
```

```
.overlap{
    height:20%;
    width: 100%;
    background-color: Transparent;
    display: none;
}
.photoText{
    height:100%;
    width: 50%;
    float: left;
}
.photoLikeComment{
    height:100%;
    width: 50%;
    float: right;
}
```

Agenda

- Esercizio tratto **dall'appello d'esame del 26 gennaio 2018**
- Riproduzione secondo specifiche di una pagina Web, partendo da uno "snapshot" di riferimento:
  - **sito Web Flickr**



- Tecnologie da utilizzare:
  - HTML
  - CSS
  - Javascript

Esame 26/01/2018: Esercizio 2

ESERCIZIO 2 (11 punti)

Si realizzino le **pagine Web dinamiche** (basate su **tecnologia HTML**, **CSS**, e **Javascript**) in grado di **riprodurre il contenuto e il layout grafico** dello "snapshot" del **sito Web flickr** di seguito riportato. Nello specifico, la pagina visualizza il contenuto relativo alla voce "*Explore*" del Menù "orizzontale" nella barra grigio scuro in alto a sinistra, e ancora "*Explore*" tra le sotto-opzioni messe a disposizione nella barra bianca (la scelta è evidenziata dalla linea di sottolineatura azzurra della voce "Explore"), optando per la modalità di visualizzazione di immagini "a mosaico" piuttosto che di una singola immagine alla volta (evidenziata dal color azzurro della prima icona posta sul lato destro della finestra principale di sfondo grigio chiaro). In particolare, si deve prevedere che al passaggio del mouse su ogni immagine visualizzata nella finestra principale, si mostri nell'immagine stessa anche il titolo della fotografia assieme al nome dell'autore, sul lato sinistro, mentre sul lato destro si riporti il numero di *preferenze* ricevute (icona "stella") e il numero di *commenti* inseriti (icona "nuvola") dagli utenti, come evidenziato per la prima immagine della finestra principale.

Infine, si preveda che il campo di ricerca nella barra grigio scuro in alto possa ammettere solo stringhe alfabetiche (tutte maiuscole o minuscole) composte da una sola parola.

N.B. La soluzione NON deve far uso del costrutto HTML `frame`. Per rappresentare le immagini/icone riportate nello snapshot, si utilizzino figure di esempio a piacere.

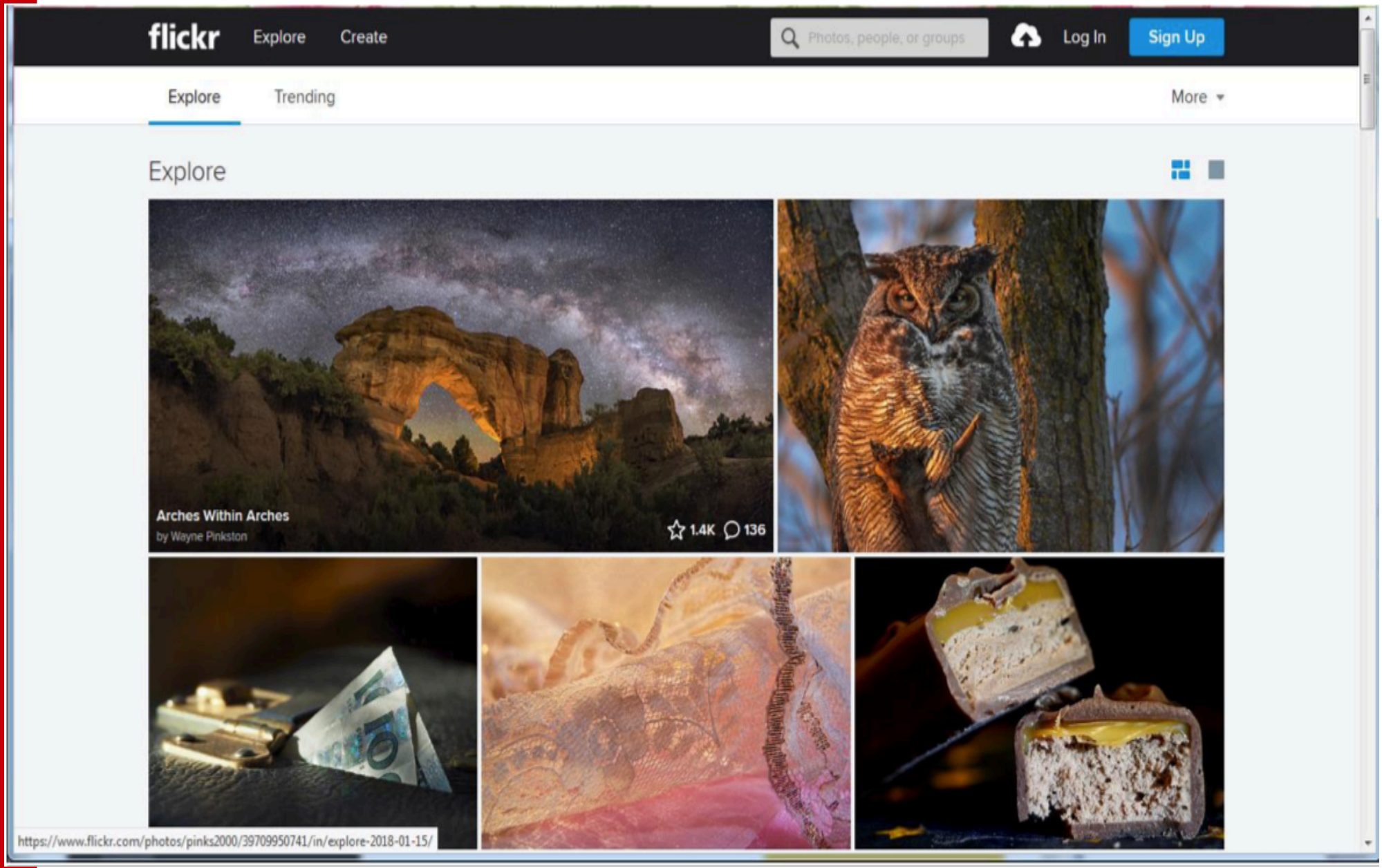
Oltre HTML e CSS: Javascript

- Oltre alla parte strutturale dell'informazione della pagina (HTML), e relativa layout grafico (CSS), è richiesto di realizzare anche una parte "comportamentale" della pagina basata su Javascript

- In particolare viene richiesto di:
  - Far comparire i dettagli delle foto quando si passa col mouse sopra la stessa, e farli scomparire all'uscita.
  - Controllare che nella barra di ricerca venga inserita solo UNA stringa alfabetica
  - Dettaglio nascosto
    - Evidenziare il bottom-border alla selezione del menù

- Anche qui, andiamo con ordine…

“snapshot” del sito Web Flickr



Javascript: Details onMouseOver

- Facendo riferimento alla struttura delle `div` per le foto illustrata nelle slide precedenti, concentriamoci sulla parte Javascript
- Gli eventi Javascritp che si occupano del trigger dell'effetto comparsa/scomparsa della `div` sono:
  - Comparsa: onmouseover
  - Scomparsa: onmouseout

  Da notare che è necessario anche il secondo altrimenti, una volta comparsa la `div` con i dettagli dell'immagine, questa non scompare più rimanendo fissa

```
<div class="tableContent" onmouseover="displayOn('overlap1');"
onmouseout="displayoff('overlap1');" style="background-image: url('images/bologna.png');
background-repeat: no-repeat;">
    <div class="filler">
    </div> <!-- filler -->
    <div class="overlap" id="overlap1"> ··· </div> <!-- overlap -->
</div>  <!-- tableContent -->
```

… e queste sono le funzioni Javascript

```
function displayOn(element)
{
    var ele = myGetElementById(element);
    ele.style.display = "block";
}

function displayoff(element)
{
    var ele = myGetElementById(element);
    ele.style.display = "none";
}
```

Javascript: Check Input String

- Per realizzare il controllo sulla stringa nell'input di ricerca abbiamo bisogno di combinare alcuni elementi:
  - Trigger dell'evento: non avendo un pulsalte apposito useremo l'evento javascript onkeyup
  - Riconoscere la pressione del pulsante INVIO per sapere quando effettuare l'effettivo controllo sulla stringa
  - Controllo della stringa: attraverso pattern regular expression

```
function checkString(event, stringa){
    if(event.keyCode !== 13)
        {
            return;
        }
    var words = stringa.split(" ");
    if(words.length >1 )
        {
            alert("E' consentita solo una parola con caratteri alfabetici");
            return;
        }
    if(stringa != stringa.toUpperCase() && stringa != stringa.toLowerCase())
        {
            alert("la parola dev'essere o tutta maiuscola o tutta minuscola");
            return;
        }
    stringa = stringa.toUpperCase()
    var alphaTest = /^[A-Z]*$/;
            if(!alphaTest.test(stringa))
            {
                alert("Si possono inserire solo caratteri alfabetici per la ricerca");
                return;
            }
    alert("Ricerca Effettuata Correttamente, the flickr is proud of you!!");
}
```

```
<input class="searchInput" id="searchStr" type="text" placeholder="Photo, People or group"
onkeyup="checkString(event,this.value);"/>
```

Javascript: Altri Dettagli

- Vediamo ora gli ultimi dettagli javascript “nascosti” nel testo
- Sottolineare la voce del menù selezionata…

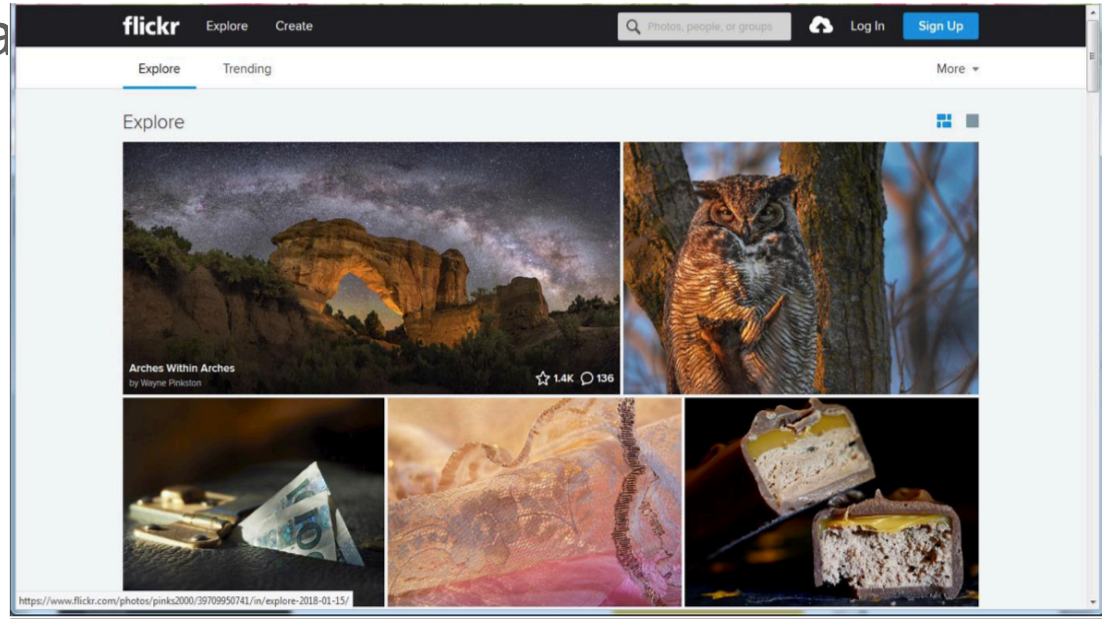
```
<ul class="menu" id="exploreMenu" style="padding-top: 0.5%; margin-left: 0px; padding-left: 0px;">
    <li onclick="underlineOnOff('exploreMenu','Explore');" ><a href="#"  id="Explore"
    style="color:black;">Explore</a></li>
    <li onclick="underlineOnOff('exploreMenu','Trending');"><a href="#" id="Trending"
    style="color:black;">Trending</a></li>
</ul>
```

…e questo è il codice Javascript

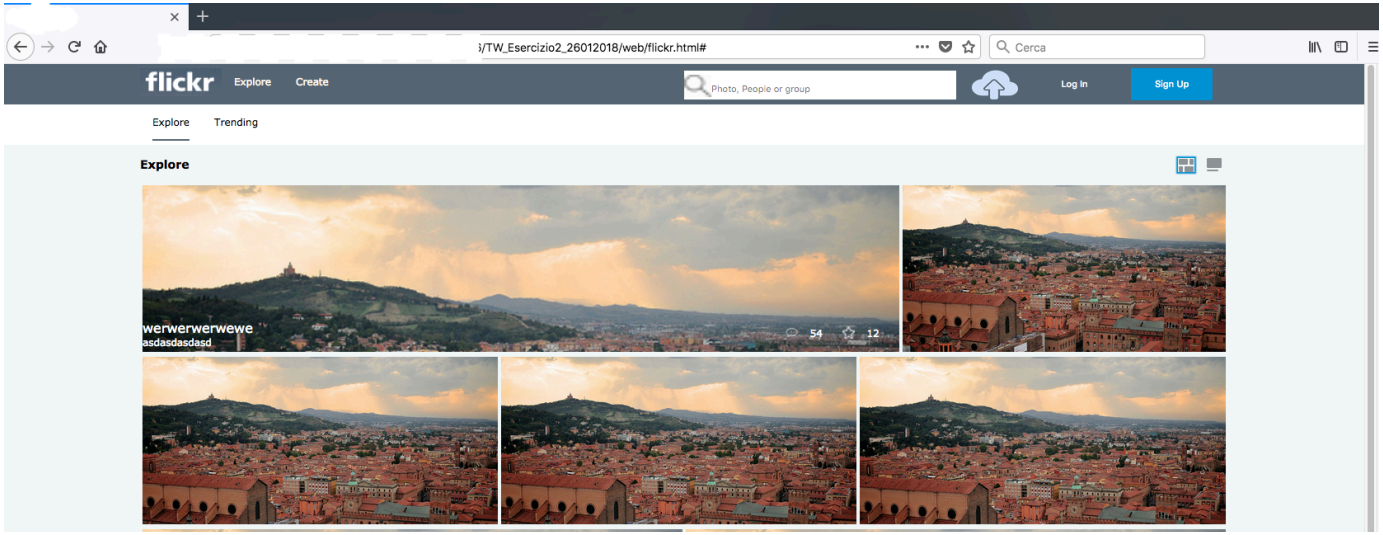
```
function underlineOnOff(lista, element)
{
    var menuItems = myGetElementById(lista).childNodes;
    for(var i=0; i< menuItems.length; i++ )
        {
            if(menuItems[i].firstChild != null)
                {
                    var a = menuItems[i].childNodes[0];
                    if(a.innerText == element)
                        {
                            var ele = myGetElementById(element);
                            ele.style.borderBottom = "2px solid #556675";
                        }
                    else
                        {
                            var ele = myGetElementById(a.innerText);
                            ele.style.borderBottom = "none";
                        }
                }
        }
}
```

Risultato finale

- Pagina richiesta



Pagina Risultato



Close enough ☺