LEZIONE 8

I FRAMEWORK

I FRAMEWORK JAVASCRIPT

- Nella produzione del software, il framework è una struttura di supporto su cui un software può essere organizzato e progettato.
- Lo scopo di un framework è di risparmiare allo sviluppatore la riscrittura di codice già steso in precedenza per compiti simili.
- In altre parole utilizzando un framework lo sviluppatore può dedicare meno tempo alla scrittura del codice e più tempo alla progettazione e al raggiungimento degli obiettivi.

FRAMEWORK PIÙ DIFFUSI

Dojo
- dojotoolkit.org

ExtJs
- www.sencha.com

jQuery
- jquery.com

jQuery UI
- jqueryui.com

MooTools
- mootools.net

Prototype
- prototypejs.org

script.aculo.us
- script.aculo.us
- dipende da prototypr

Progettato per farti cambiare il modo in cui scrivi Javascript

VANTAGGI

- jQuery ha due vantaggi principali:
  - superare uno dei problemi che maggiormente complica la vita agli sviluppatori: la compatibilità tra le varie versioni dei browser
  - rendere lo script più compatto: scrivi di meno, fai di più.

INSERIRE JQUERY IN UNA PAGINA

- jQuery viene rilasciata in due versioni:
  - Compressa (che permette di avere file di dimensioni notevolmente più piccole)
  - Non compresso (versione leggibile e con commenti adatta per il debug, per fini didattici e per lo sviluppo).
- La versione compressa (minified) è contraddistista dal suffisso .min.

JQUERY SU CDN

- Un certo numero di grandi imprese mettono a disposizioni copie di jQuery su CDN (Content Deployment Network) pubblici:
  - Google Ajax API CDN (Disponibile anche download sicuro SSL via HTTPS)
    - http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js
  - Microsoft CDN (Disponibile anche download sicuro SSL via HTTPS)
    - http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.11.1.min.js
  - jQuery CDN
    - http://code.jquery.com/jquery-1.11.1.min.js (Minified version)
    - http://code.jquery.com/jquery-1.11.1.js (Development version)

JQUERY SU CDN

- Per caricare jQuery (come qualsiasi altra libreria) si usa il tag script:

```
<script type="text/javascript"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js">
</script>
```

- Se si usa un CDN per caricare jQuery (soluzione consigliata) può essere un buona idea preparare un'alternativa di caricamento così:

```
<script type="text/javascript">
   var scriptpath = //inserisci l'url di jQuery sul tuo sito
   if (!jQuery){
        document.write('<script type="text/javascript" src="' +
                scriptpath + '" </script>');
   }
</script>
```

L'INIZIO

- La libreria jQuery è costituita da codice che viene eseguito non appena caricata e che:
  - Crea un oggetto jQuery che è il namespace in cui ci muoveremo usando jQuery
  - Crea una funzione globale jQuery che è il cuore della libreria
  - Crea un alias per entrambe che è il segno del dollaro

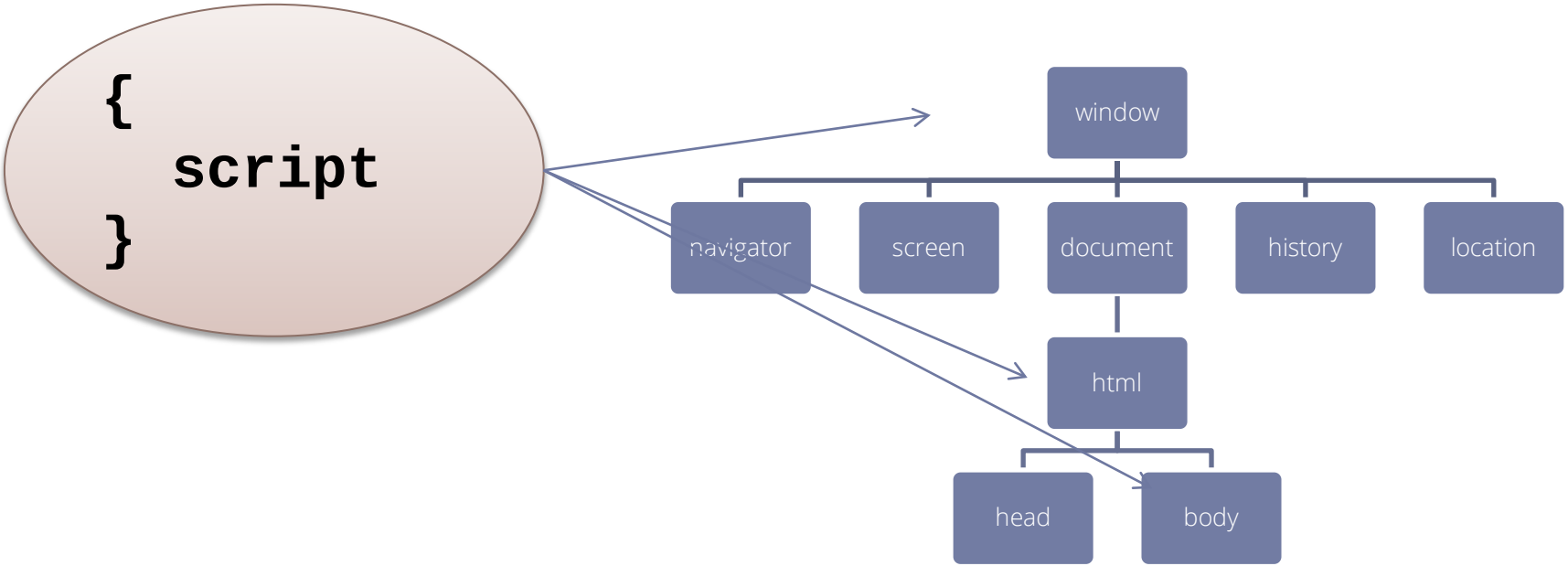
jQuery = $

A COSA SERVE jQuery

- Modifica degli elementi DOM:
  - Sostituisce i metodi e le proprietà native degli elementi che compongono il DOM per modificarli con maggiore facilità
  - Aggiunge metodi che consentono di scrivere meno codice
- Interattività:
  - aggiunge nuovi eventi
  - rende più flessibile la gestione degli eventi
- Creazione di elementi
- Utilizzo dei plugin

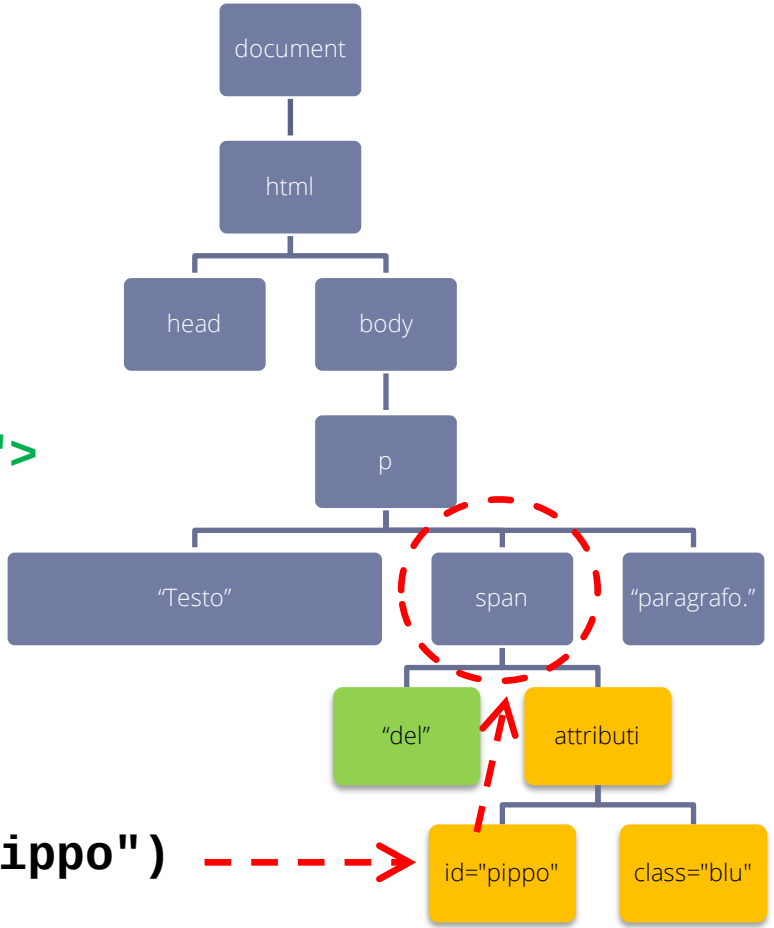
Come agisce javascript sulla tua pagina?

JAVASCRIPT AGISCE SUL DOM

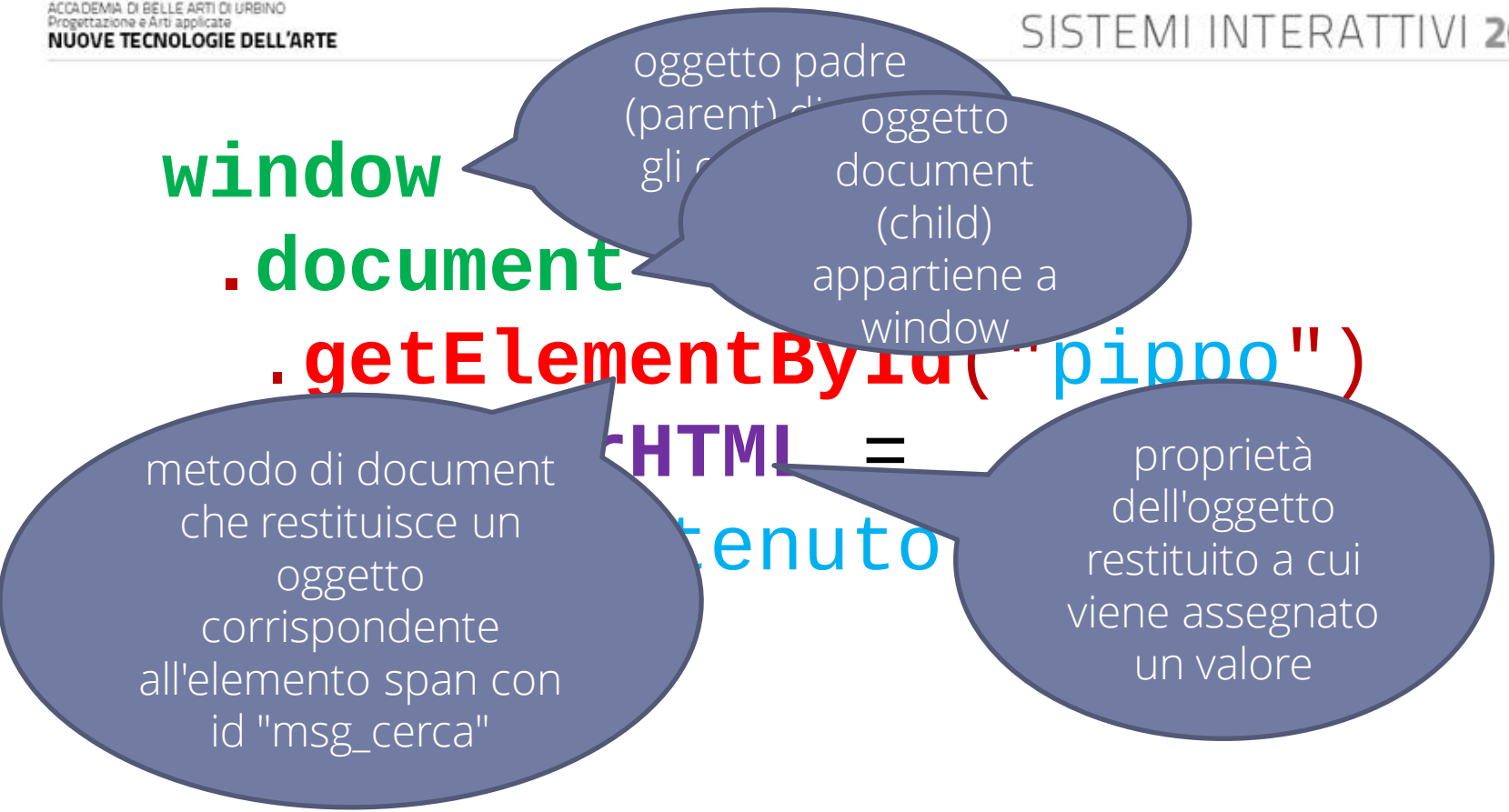


javascrip agisce sul DOM

```
<html>
 <head></head>
   <body>
    <p> Testo
    <span class="blu" id="pippo">
       del</span>
       paragrafo.</p>
   </body>
</html>
```

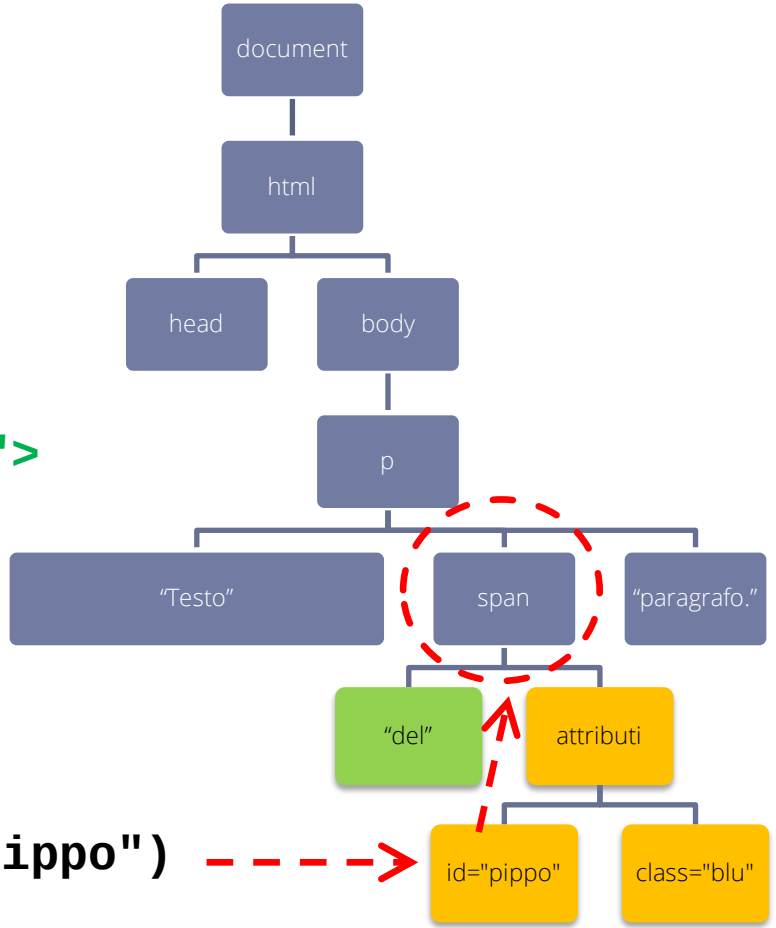


document.getElementById("pippo") - - - ->

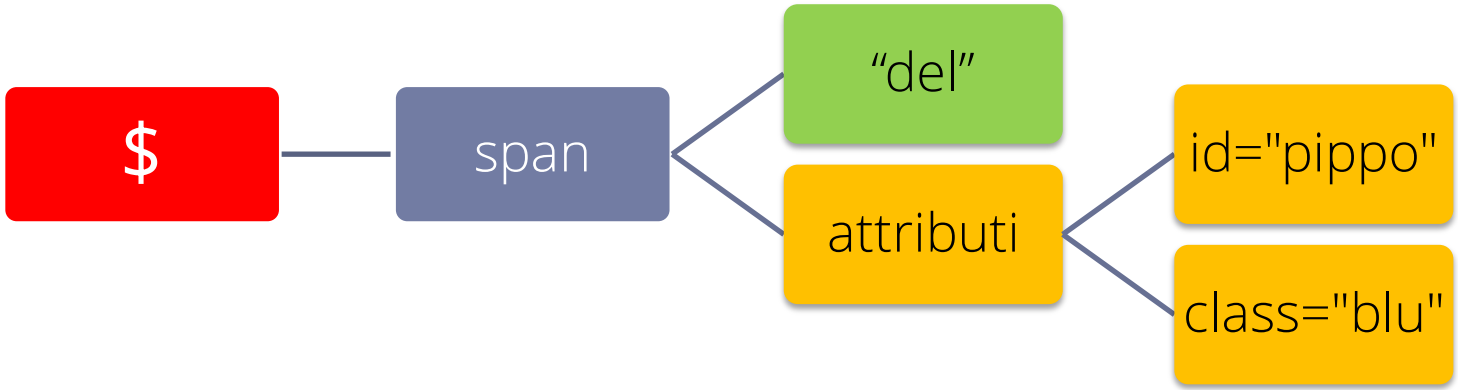

oggetto padre
(parent) di
gli
oggetto
document
(child)
appartiene a
window
window
.document
.getElementById("pippo")
.HTML =
tenuto
metodo di document
che restituisce un
oggetto
corrispondente
all'elemento span con
id "msg_cerca"
proprietà
dell'oggetto
restituito a cui
viene assegnato
un valore

javascrip agisce sul DOM

```
<html>
 <head></head>
   <body>
    <p> Testo
    <span class="blu" id="pippo">
       del</span>
       paragrafo.</p>
   </body>
</html>
```

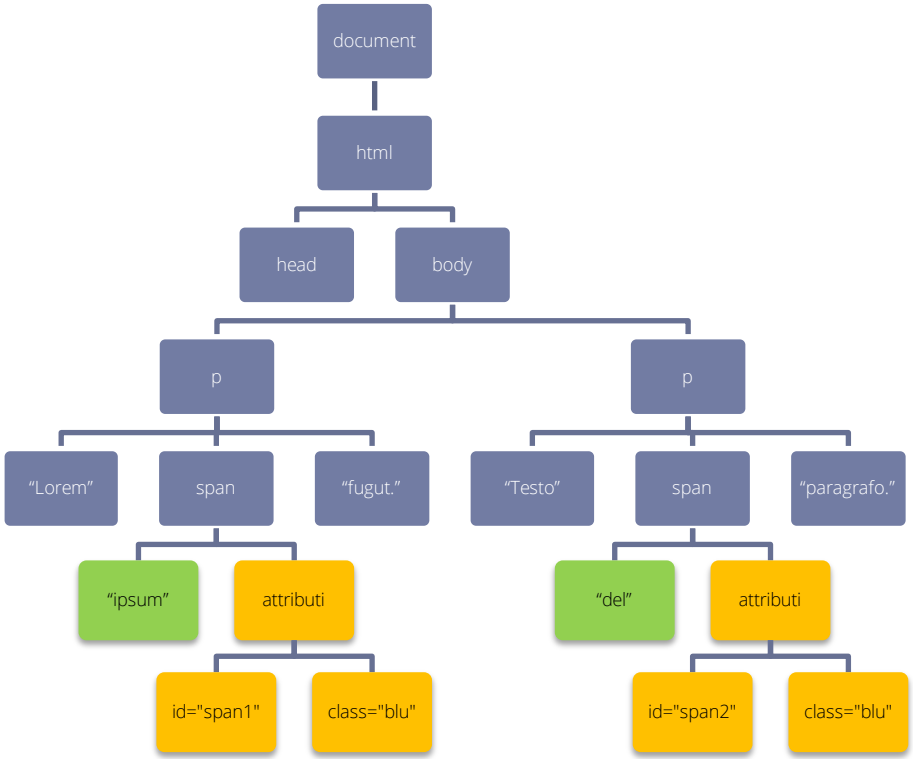


```
var mioOggetto = $("pippo");
```

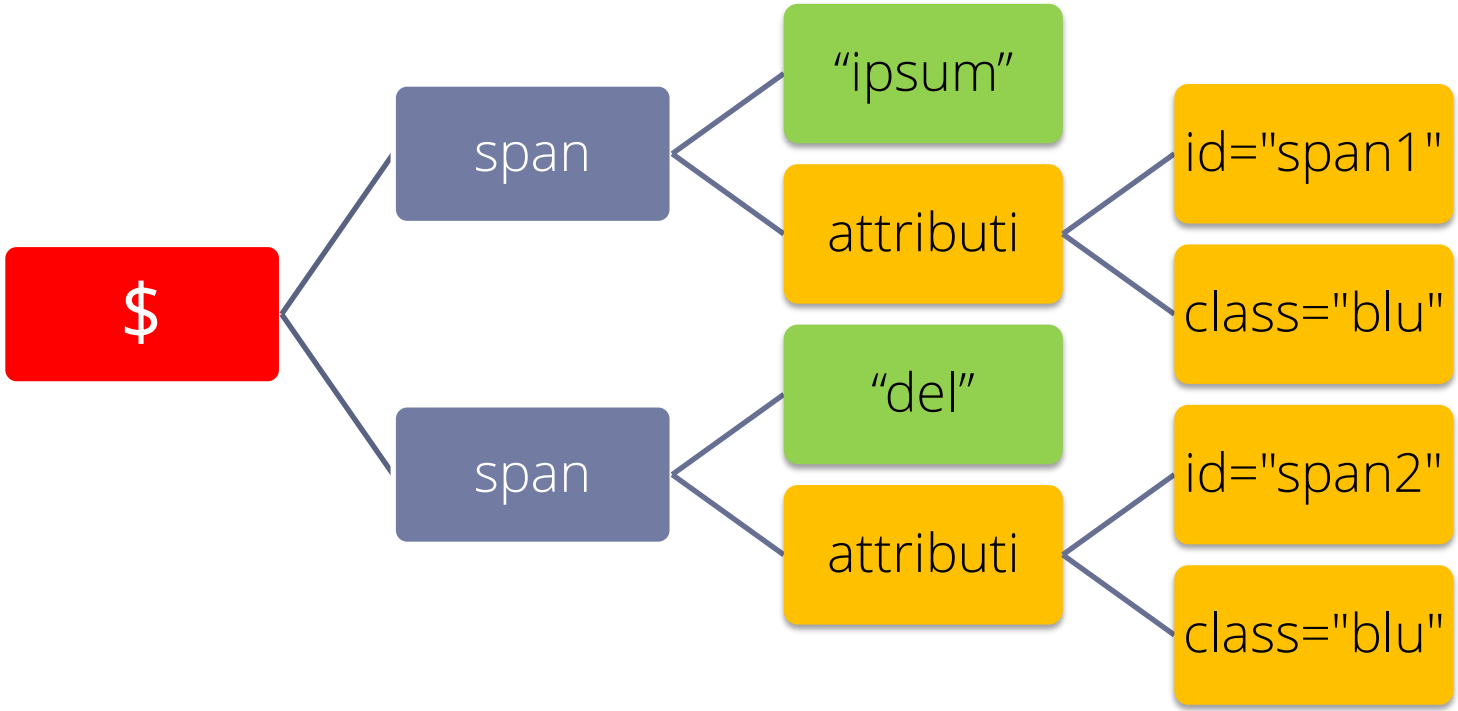


javascrip agisce sul DOM

```
<html>
 <head></head>
   <body>
    <p> Lorem
    <span class="blu" id="span2">
        ipsum</span>
       fugit.</p>
    <p> Testo
    <span class="blu" id="span1">
        del</span>
       paragrafo.</p>
   </body>
</html>
```



var mioOggetto = $(".blu");



VALORI E RIFERIMENTI

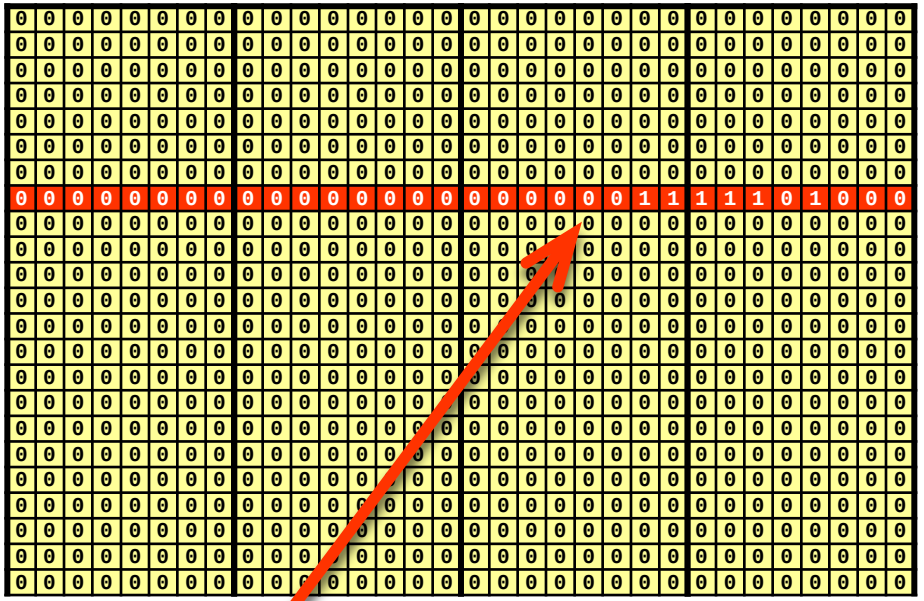
- Quando assegno un valore a una variabile l'interprete javascript riserva uno spazio di memoria per quella variabile.
- Possiamo dire che ad ogni variabile corrisponde una cella della memoria fisica del computer.
- Ognuna di queste celle è raggiungibile per l'elaborazione attraverso un riferimento anch'esso espresso in bit.
- Quando scrivo:

```
var a = 1000;
```

- Dico che a corrisponde ad una ben determinata cella di memoria composta da 32 bit in cui è scritto il formato binario il numero 1000.

VALORI E RIFERIMENTI

- La variabile a è associata a una cella di memoria.
- La cella contiene il valore di a in formato binario.

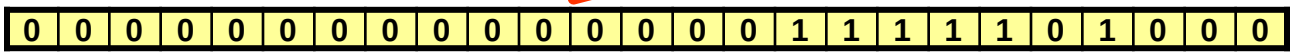


```
var a = 1000;
```

VALORI E RIFERIMENTI

- Se assegno ad **a** un numero intero stabilisco due cose
  - Che ad a vengono riservati 32 bit in memoria
  - Che il valore contenuto nella cella viene interpretato come numero intero

```
a = 1000;
a = -1;
```



VALORI E RIFERIMENTI

- Quando la casella che la variabile rappresenta contiene direttamente il dato si dice che la varibile contiene un valore.
- Se scrivo

```
var a = 10;
var b = a;
```

il valore di a viene copiato nella casella di memoria rappresentata da b e i due valori rimangono indipendenti.

VALORI E RIFERIMENTI

- Quando il valore assegnato a una variabile è un oggetto l'interprete javascript fa un operazione un po' più complessa. Lo spazio di 32 bit riservato alla variabile viene usato per memorizzare l'indirizzo di memoria in cui è collocato l'oggetto.
- In questo caso la variabile contiene il **riferimento** all'oggetto..
- Se scrivo:

```
var elemento = document.createElement("div");
```

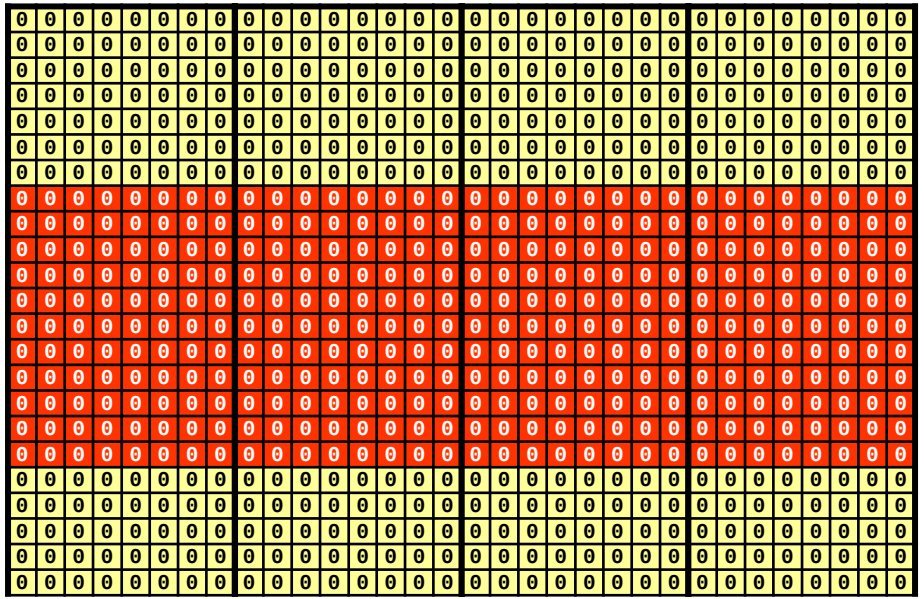
La cella di memoria di 32 bit rappresentata da elemento non conterrà l'elemento html creato ma l'indirizzo fisico di memoria in cui è memorizzato.

VALORI E PUNTATORI

```
var elemento = document.createElement("div");
```

Elemento contiene l'indirizzo alla zona di memoria in cui l'oggetto è memorizzato.

che punta a...



VALORI E RIFERIMENTI

- Quando la casella che la variabile rappresenta contiene l'indirizzo di memoria a partire dal quale e memorizzato l'oggetto si dice che la variabile, contiene il riferimento all'oggetto.
- L'interprete si occuperà automaticamente di risolvere il riferimento.

```
var elemento = document.createElement("div");
elemento.setAttribute("class", "articolo");
```

- Se però scrivo

```
var e = elemento;
```

quello che viene copiato in **e** è il riferimento all'oggetto ed entrambe le variabili si riferiranno allo stesso elemento.

DIFFERENZE

- In javascript questo processo non è obbligatorio in certi casi posso eseguire il codice prima che la pagina sia caricata o assegnare direttamente agli tag HTML gli eventi
- In jQuery tutti i comandi devono essere eseguiti quando il DOM è completamente caricato e jQuery è correttamente inizializzato.

$('document').ready()

- Per ottenere questo ogni comando di jQuery va inserito in questo blocco:

```
$(document).ready( function() {


});
```

$('document').ready()

- O, in versione compatta, semplicemente:

```
$(function() {

});
```

LO STILE DI SCRITTURA

- Per capire meglio. Ho una funzione globale:

```
$(); //jQuery();
```

- e un metodo della classe jQuery:

```
.ready();
```

- Che sono sinonimi. In entrambi casi passo come parametro una funzione anonima. Cioè l'intero blocco di codice di cui è composta la funzione preceduta da **function():**

```
$(function(){//blocco comandi});
```

LO STILE DI SCRITTURA

- Ma posso usare anche una funzione con nome:

```
function documentoPronto() {
  //corpo funzione
}
```

- E passare come parametro il nome della funzione:

```
$(document).ready(documentoPronto);
```

jQuery
write less, do more.

DIFFERENZE

- L'evento **windows.onload** viene sparato quando l'intero documento html è stato caricato, comprese le immagini.
- L'evento **.ready()** viene sparato quando il DOM è caricato (la sola struttura del documento).

jQuery
write less, do more.

DIFFERENZE

- **windows.onload** è una proprietà:

```
windows.onload = function() { //comandi}
```

- **.ready()** è un metodo:

```
$(document).ready(function() { //comandi});
```

jQuery
write less, do more.

DIFFERENZE

- Con l'evento **windows.onload** **assegno un'unica** funzione all'evento.
- Con l'evento **.ready()** **aggiungo** una funzione all'evento.

IL DOLLARO $

- la funzione $() (che sostituisce per concisione jQuery()) è la funzione principale. Può avere varie combinazioni di parametri:
  - **`$(funzione);`**
  - **`$(elemento);`**
  - **`$(selettore css);`**
  - **`$(selettore css, contesto);`**
  - **`$(codice html);`**

RICERCA DI ELEMENTI

- Il modo più classo di procedere di jQuery è quello di "selezionare alcuni elementi ed eseguire azioni su di essi."
- La selezione avviene passando alla funzione $():
  - Un stringa che rappresenta un selettore CSS
  - Un elemento del DOM (ad esempio document o window)

jQuery
write less, do more.

ESEMPI

- Selezione degli elementi in base alla loro ID
  - `$('#myid') / / L'ID deve essere univoco`
- Selezione degli elementi in base al nome della classe
  - `$('div.myClass')`
- Selezione degli elementi in base a un attributo
  - `$ ( 'input [name = first_name]' )`
- Selezionare gli elementi in base a un selettore CSS
  - `$ ( '# contenuti ul.people li' );`

PSEUDO-SELETTORI

- `$('a.external:first');` *// primo elemento <a>*
  // con classe 'external'
- `$('tr:odd');` *// elementi <tr>* dispari *in*
  //una tabella
- `$('#myForm :text');` *// tutti gli elementi input di*
  // tipo text in #myForm
- `$('div:visible');` *// tutte le div visibli*
- `$('div:gt(2)');` *// selecciona tutte le div eccetto*
  // le prime tre
- `$('div:animated');` *// tutte le div animate*

RISULTATO

- La funzione $() restituisce un oggetto di tipo jQuery.
- L'oggetto jQuery può rappresentare il set degli elementi trovati o un unico elemento.
- Per controllare se la ricerca ha prodotti risultato deve controllare la proprietà length dell'oggetto jQuery restituito.

```
if ($('div.foo').length > 0) { ... }
```

jQuery PSEUDO SELETTORI FORM

- :button Seleziona elementi <input> con l'attributo type='button'
- :checkbox Seleziona elementi <input> con l'attributo type='checkbox'
- :checked Seleziona elementi <input> selezionati
- :disabled Seleziona elementi disabilitati
- :enabled Seleziona elementi abilitati
- :file Seleziona elementi <input> con type='file'
- :image Seleziona elementi <input> con type='image'
- :input Seleziona elementi <input>, <textarea> y <select>
- :password Seleziona elementi <input> con type='password'
- :radio Seleziona elementi <input> con type='radio'
- :reset Seleziona elementi <input> con type='reset'
- :selected Seleziona elementi <options> selezionati
- :submit Seleziona elementi <input> con type='submit'
- :text Seleziona elementi <input> con type='text'

LAVORARE CON LE SELEZIONI

- Una volta ottenuto un set di componenti in base alla selezione, si possono utilizzare i metodi dell'oggetto jQuery.
- Gli oggetti jQuery non hanno proprietà direttamente accessibili escluso length
- I metodi si dividono in due categorie: getter e setter.
  - i metodi getter restituiscono una proprietà dell'elemento selezionato,
  - i metodi setter di impostano una proprietà di tutti gli elementi del set restituito.

CHAINING

- Ogni metodo jQuery restituisce l'oggetto jQuery su cui il metodo ha operato. Questo rende possibilie il concatenamento tipico della scrittura javascript di jQuery:

```
$('#content')
  .find('h3')
  .eq(2)
  .html('nuovo testo per il terzo h3');
```

GETTERS e SETTERS

- I metodi per impostare un valore hanno lo stesso nome dei metodi per ottenere un valore.
- Ciò che differenzia il metodo setter dal corrispondente getter è il parametro in più costituito dal valore a da impostare:

```
$ ('H1').html( 'ciao mondo' );
$ ('H1').html();
```

STILI CSS

- Per ottenere o modificare lo stile css di un elemento (o di un set di elementi) ho il metodo .css:
- Getter;

```
$ ( 'H1' ). css ( 'fontSize' )        / / restituisce"19px"
$ ( 'H1' ). css ( 'font-size' )       / / funziona
```

- Setter

```
$('h1').css('fontSize', '100px');
$('h1').css({ 'fontSize' : '100px', 'color' : 'red' });
```

CLASSI CSS

- Anche se molto utile, il metodo .css non dovrebbe essere usato per applicare direttamente stili agli elementi (si può fare direttamente da CSS). È meglio usare CSS per definire classi e applicare queste agli elementi a secondo delle nostre necessità.

```
var $h1 = $('h1');
$h1.addClass('big');
$h1.removeClass('big');
$h1.toggleClass('big');
if ($h1.hasClass('big')) { ... };
```

DIMENSIONI

- jQuery offre una varietà di metodi per ottenere e impostare le dimensioni e la posizione di un elemento.

```
$('h1').width('50px');    // imposta la larghezza di tutti gli elementi H1
$('h1').width();          // ottiene la larghezza di tutti gli elementi H1
$('h1').height('50px');   // imposta l'altezza di tutti gli elementi H1
$('h1').height();          // ottienel'altezza di tutti gli elementi H1
$('h1').position();   // restituisce un oggetto contenente
                      // informazioni sulla posizione
                      // del primo elemento H1 relativo all'offsett
                      // del suo elemento padre
```

ATTRIBUTI

- Il metodo attr() ha una sintassi simile a css ma ottiene e imposta gli attributi di un elemento anziché lo stile:
- Setter

```
$('a').attr('href', 'hrefTuttiUguali.html');
$('a').attr({
                    'title' : 'tutti lo stesso title',
                    'href' : 'laStessaUrl.html'
              });
```

- Getter

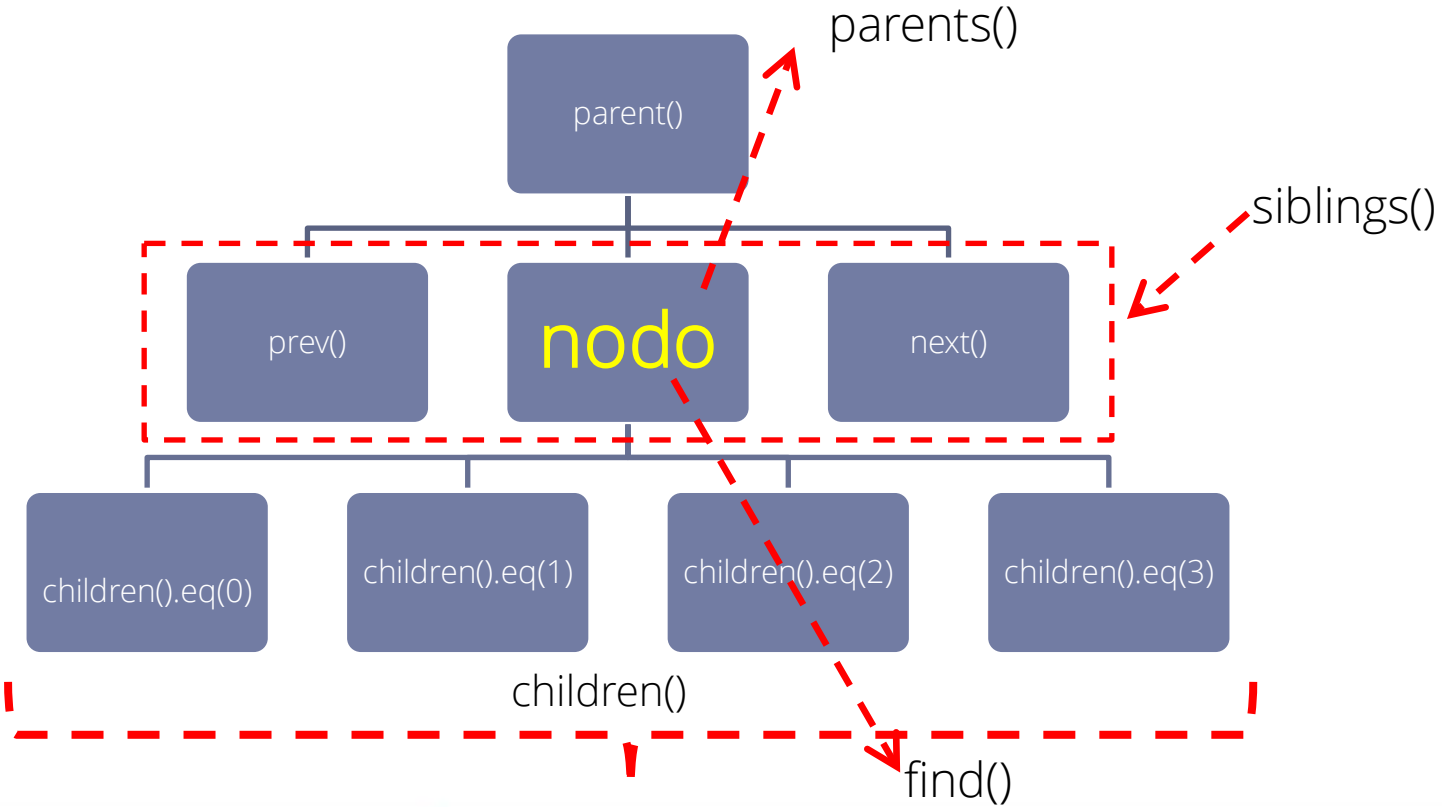
```
$('a').attr('href'); //restituisce href del primo elemento
```

ALTRI GETTER-SETTER

- .html()
  Ottiene o imposta il contenuto di un elemento HTML.
- .text()
  Ottiene o imposta il contenuto testuale dell'elemento, nel caso un testo con tag HTML, questi viene rimossi.
- .val()
  Ottiene o imposta il valore (*value*) di elementi di un form.

PROPRIETÀ DEI NODI

RELAZIONE TRA I NODI



jQuery CREARE NUOVI ELEMENTI

- Si può utilizzare $() anche per creare un nuovo oggetto jQuery (quindi un nuovo elemento)
  - Stringa HTML

```
$('<p>Un nuovo paragrafo</p>');
$('<li class="new">nuovo elemento della lista</li>');
```

  - Stringa HTML + oggetto che rappresenta gli attributi

```
$('<a/>', {
  html : 'Un <strong>nuovo</strong> enlace',
  'class' : 'new',
  href : 'foo.html'
});
```

INSERIRE, SPOSTARE...

- Ci sono diversi modi per collocare gli elementi nel DOM:
  - Posizionare nuovi elementi o selezioni rispetto ad un altro elemento
  - Inserire nuovi elementi o selezioni come figli di un altro elemento

```
var $myNewElement = $('<p>Nuovo elemento</p>');
$myNewElement.appendTo('#content');
$myNewElement.insertAfter('ul:last');
        // eliminerá l'elemento <p>
        // esistente in #content
$('ul').last().after($myNewElement.clone());
        // copia l'elemento <p>
        // e lo duplica
```

EVENTI

- Contrariamente a javascript gli eventi si gestiscono con metodi non con proprietà

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

ON

- *$(selector)*.on("evento1 evento2", funzione)
  1° parametro: Uno o più tipi di eventi separati da uno spazio.

  funzione: Una funzione che viene associata all'evento per ciascuno elemento del set

Evento	Descrizione
blur	Si verifica quando l'elemento perde il focus
change	Si verifica quando l'elemento cambia valore
click	Si verifica quando fai clic colmouse
dblclick	Si verifica quando un doppio clic del mouse
error	Si verifica quando c'è un errore
focus	Si verifica quando l'elemento riceve il focus
keydown	Si verifica quando si preme un tasto
keypress	Si verifica quando il tasto viene premuto e rilasciato
keyup	Si verifica quando il tasto viene rilasciato
load	Si verifica quando il documento viene caricato
mousedown	Si verifica quando si preme il pulsante del mouse
mouseenter	Si verifica quando il puntatore del mouse entra nell'area dell'elemento
mouseleave	Si verifica quando il puntatore del mouse esce dall'area dell'elemento
mousemove	Si verifica quando il puntatore del mouse si sposta
mouseout	Si verifica quando il puntatore del mouse viene spostato all'esterno di un elemento
mouseover	Si verifica quando il puntatore del mouse viene spostato sopra un elemento
mouseup	Si verifica quando il pulsante del mouse viene rilasciato
resize	Si verifica quando la finestra viene ridimensionata
scroll	Si verifica quando un elemento viene fatto scorrere con la barra di scorrimento
select	Si verifica quando un testo viene selezionato
submit	Si verifica quando un form viene inviato
unload	Si verifica quando il documenti viene abbandonato