


JavaScript



Ombretta Gaggi
Università di Padova

Nozioni di base

- ❑ JavaScript viene sviluppato nel 1995 da Netscape con il nome LiveScript. In seguito si unisce anche Sun Microsystems, creatrice del linguaggio Java
- ❑ Si basa sulle specifiche standard della European Computer Manufactures Association
 - www.ecma.ch
- ❑ Permette di definire degli script, anche frammentati all'interno del documento

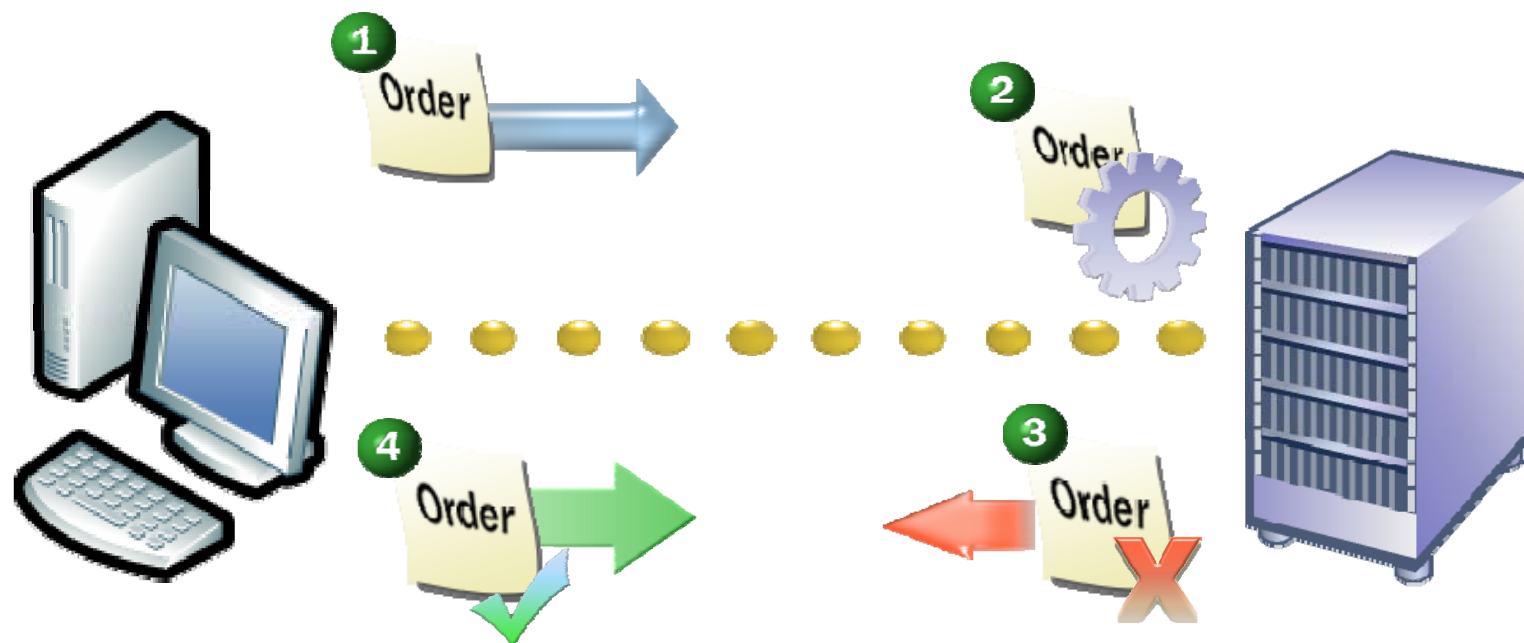


JavaScript vs. Java

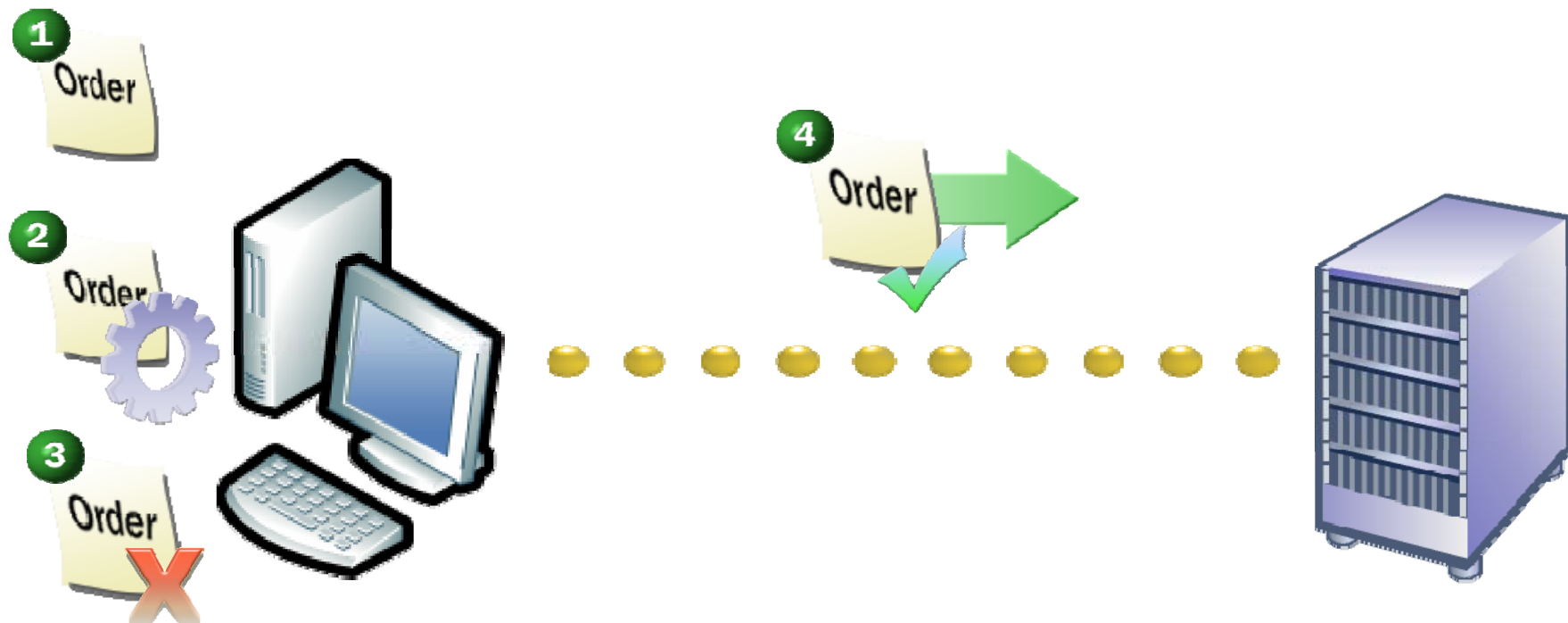
- ❑ Nonostante il nome, **JavaScript** non è un sottoinsieme del linguaggio **Java**
- ❑ Java è un linguaggio orientato agli oggetti, JavaScript no
 - supporta gli oggetti ma non le classi
- ❑ Non supporta l'ereditarietà
- ❑ A differenza di Java, JavaScript non richiede la dichiarazioni delle variabili e permette una tipizzazione dinamica
 - È sempre comunque opportuno dichiarare le variabili esplicitamente



Programmazione "Server Side"



Programmazione "Client Side"



Scripting non intrusivo

- ❑ Lo scripting non intrusivo è focalizzato sull'utente ed è progettato per migliorare una struttura di markup già di per sé semantica ed accessibile
- ❑ In particolare:
 - non attira l'attenzione dell'utente, è un'aggiunta funzionale al sito ovvia (**migliora usabilità**)
 - non attira l'attenzione dell'utente quando non funziona (**degrado aggraziato**)
 - non modifica le funzionalità della pagina, se non funziona l'utente non deve accorgersi che manca (**accessibilità**)
 - non modifica la struttura della pagina (**separazione struttura - comportamento**)



JavaScript

- ❑ È un linguaggio piuttosto semplice, che permette di creare documenti dinamici, in grado di interagire con l'utente
 - Ex. Dare un messaggio se l'utente fa un click con il tasto destro del mouse
- ❑ A differenza di PHP, non supporta il networking e le operazioni sui file anche se queste ultime trovano parziale supporto con le File API di HTML5
 - cookie
- ❑ Document Object Model (DOM): permette agli script JavaScript di avere accesso ai contenuti e ai widget del documento HTML in cui sono contenuti
- ❑ Calcoli basati sugli eventi



Effetto Gmail (beta 31 marzo 2004, 7 luglio 2009)

Gmail [Calendar](#) [Documenti](#) [Web](#) [Reader](#) [altro ▼](#)

ombretta.gaggi@gmail.com | [Impostazioni](#) | [Guida](#) | [Es](#)



Cerca nella posta

Cerca sul Web

[Mostra opzioni di ricerca](#)
[Crea un filtro](#)

[Scrivi messaggio](#)

Posta in arrivo

[Buzz](#)

[Speciali](#)

[Posta inviata](#)

[Bozze](#)

[Personale](#)

[Viaggio](#)

[Altre 6 ▼](#)

[Contatti](#)

[Attività](#)

Chat

Cerca, aggiungi o invita

Gazzetta.it - Venus, vecchietta terribile "Mi sento come agli inizi" - 17 ore fa

Clip web

Archivia

Segnala come spam

Elimina

Sposta in ▼

Etichette ▼

Altre azioni ▼

[Aggiorna](#)

1 - 4 di 4

Seleziona: Tutti, Nessuno, Già letti, Da leggere, Speciali, Non speciali

	URGENT! Berlusconi 03	URGENT! - Camera Paolo Berlusconi - Dear G. Contatti 03	12 gen
	URGENT! re, giuliano (3)	URGENT! 2010: Anna Berlusconi - Dato Berlusconi, la re	25/12/09
	URGENT! re (3)	URGENT! - Paper Berlusconi - Forwarded message From: M	06/11/09
	URGENT! Berlusconi infuoca	URGENT! Berlusconi - Dear Camera Paolo Berlusconi	06/11/09

Seleziona: Tutti, Nessuno, Già letti, Da leggere, Speciali, Non speciali

Archivia

Segnala come spam

Elimina

Sposta in ▼

Etichette ▼

Altre azioni ▼

[Aggiorna](#)

1 - 4 di 4



Inserire gli script in pagine web

- Possono apparire sia nell'header di un file HTML che nel corpo, con funzioni molto diverse:
 - **header** → servono per produrre contenuto su richiesta o si occupano dell'interazione con l'utente. In generale, definizioni di funzioni che vengono riutilizzate più volte
 - Ex. codice associato agli elementi di un form
 - **body** → script da interpretare una volta sola
 - Ex. controllo su un dato specifico
- Come per il CSS, gli script inseriti nell'intestazione sono inseriti tra commenti

<!--

codice JavaScript

//-->

- Commenti JavaScript: // oppure /* commento */



Esempio script all'interno del body

```
<html>
<head>
  <title>Pagina di Esempio</title>
</head>

<body onLoad="alert('Messaggio di apertura');">
  <p>Pagina di esempio con un alert.</p>
</body>
</html>
```



Browser che non supportano gli script

```
<script type="text/javascript">
```

```
<!--
```

```
    codice dello script
```

```
//-->
```

```
</script>
```

```
<noscript>
```

```
    <meta http-equiv="refresh" content="0;  
                                             url=altrapagina.html">
```

```
</noscript>
```



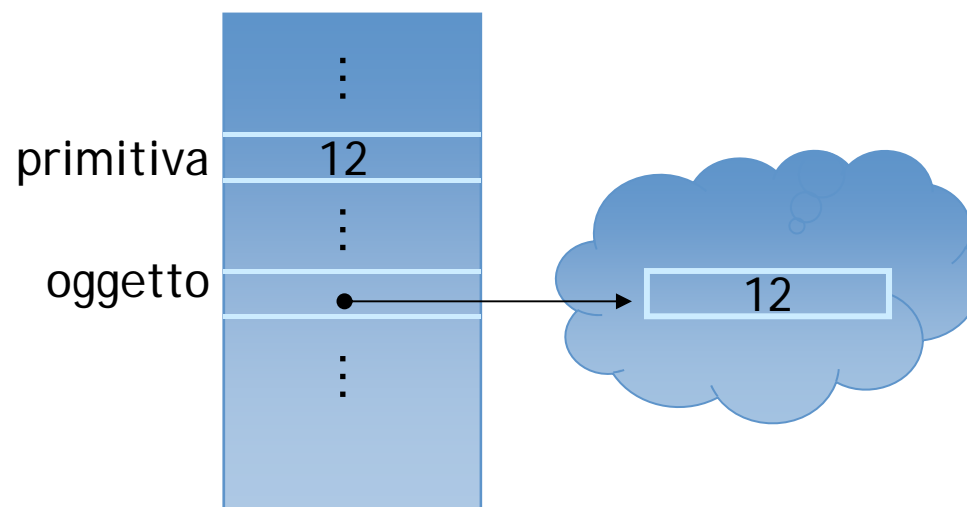
Gli oggetti e variabili JavaScript

- ❑ Ogni oggetto ha un insieme di proprietà
 - proprietà di **dati**
 - proprietà di **metodi**
- ❑ I tipi che non sono oggetti vengono chiamati primitive
- ❑ Per riferirsi alle proprietà di un oggetto si usa la forma **nome_variabile.nome_proprietà**
 - `automobile.modello`
 - `automobile.gira(90)`
- ❑ I nomi di variabili possono contenere lettere, cifre (non al primo posto), `_`, `$`, e non devono essere uguali alle stringhe utilizzate per i comandi (parole riservate)
 - per convenzioni non si usano lettere maiuscole e il segno del dollaro



Primitive ed oggetti

- ❑ number (Number)
- ❑ string (String)
- ❑ boolean (Boolean)
- ❑ undefined
- ❑ null/NaN



- ❑ Variabili con `var`
- ❑ Letterali numerici
 - 12 .12 12.12 ... come in PHP
- ❑ Letterali stringa
 - "questa è una stringa", 'anche questa e\' una stringa'
- ❑ Operatori numerici
 - + - * / ++ --
- ❑ Oggetti specifici includono una serie di operazioni e costanti di uso frequente
 - Oggetti Math e Number



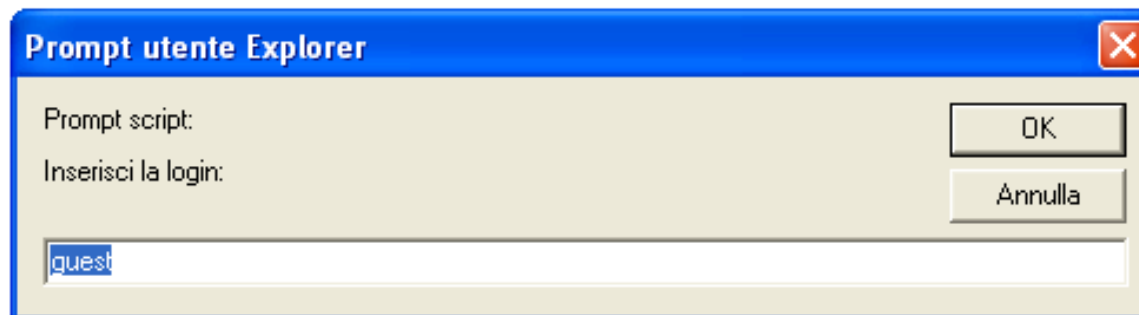
Tipi

- JavaScript supporta la tipizzazione dinamica
 - `1 + "Aprile" + 2005`
 - `14 * "3"`
 - `1 * "Aprile" → NaN`
- `.toString` converte numeri in stringa, ove è necessario
- `typeof` restituisce stringa descrittiva del tipo
- `instanceof` verifica se l'operando è stato creato con funzione costruttore



Output

- ❑ `document.write("<p>Testo paragrafo</p>");`
- ❑ `alert("Messaggio \n su più righe");`
- ❑ `var question =`
`confirm("Salvare il file?");`
- ❑ `input = prompt("Inserisci la login:",`
`"guest");`



Istruzioni condizionali

- ❑ Sono uguali a quelle di **PHP**, ma non è obbligatorio l'uso dei blocchi
 - if (espressione di controllo) istruzione
 - nome_variabile=(condizione)?valore_se_vero:valore_se_falso
- ❑ **switch (espressione) {**
 case valore1:
 ...
 case valore2:
 ...
 [default:
 ...]
}

Operatori booleani

&&, and

||, or

!, not

Operatori relazionali

==, !=

===, !==

>, >=

<, <=



Cicli

- ▣ while (espressione di controllo){
 #istruzioni del ciclo
}
- ▣ for (inizializzazione; espr. controllo; incremento){
 #istruzioni del ciclo
}
- ▣ do {
 #istruzioni del ciclo
} while (espressione di controllo)



Creazione e modifica di oggetti

- ❑ Quando viene creato, un oggetto è vuoto e privo di proprietà
 - `var occhiale = new Object();`
- ❑ Le proprietà vengono istanziate dinamicamente
 - `occhiale.tipo = "solari";`
 - `occhiale.marca = "Rayban";`
- ❑ Accesso alle proprietà:
 - `for (var prop in occhiale)`
istruzione
- ❑ Le proprietà possono anche essere eliminate
 - `delete occhiale.marca;`

```
function marca_occhiale(){document.write(this.marca);}  
function occhiale(ntipo,nmarca){  
    this.tipo =ntipo; this.marca = nmarca;  
    this.print_marca = marca_occhiale;}  

```



Array

- Sono oggetti che svolgono alcune funzioni speciali
 - `var lista = new Array(1, 2, "tre", "quattro");`
 - `var lista_vuota = new Array(100);`
 - `var lista_spesa = ["pane", "latte", "birra"];`
 - `lista_spesa[1] → "latte"`
 - `lista.length → 4`
- Altri metodi:
 - `lista_spesa.join(";"); → "pane;latte;birra"`
 - `lista_spesa.sort(); → ["birra", "latte", "pane"];`
 - `var nuova_lista = lista_spesa.concat(5,6);`
 - `slice`: come substring per le stringhe
 - `pop, push, shift, unshift`



Array associativi

- JavaScript prevede anche la definizione di array associativi:

```
voti = new Array();  
voti["Mario"] = 7;  
voti["Gianni"] = 4;  
voti["Monica"] = 4;
```

oppure

```
var voti = { "Mario": "7", "Gianni": "4", "Monica": "4" };
```



Scope delle variabili

- ❑ Le variabili vengono dichiarate con la parola chiave **var**
 - `var x=5, y=7, mese = 'Aprile';`
- ❑ Lo scope di una variabile è legato alle funzioni. Se definita all'interno di una funzione, indipendentemente da dove è definito lo scope è l'intera funzione
- ❑ Se definito fuori da una funzione la variabile è globale
- ❑ Se una variabile non viene dichiarata (tramite la parola chiave `var`) questa è automaticamente un variabile globale



Funzioni

- ❑ **function** scriviNome([arg1, ..., arg2]) {
 // inizializzo le variabili all'interno delle funzioni
 var nome=prompt("inserisci qui il tuo nome","il tuo nome");
}

scriviNome();

nome="Gianni"; // in assenza di var questa è una (nuova)
 // variabile globale

- ❑ I parametri di una funzione possono variare nel numero
 - array **arguments**



Corrispondenza dei pattern

- ❑ Ripresi dal linguaggio **PHP** ma utilizzando i metodi dell'oggetto **String**
 - **search**
 - **replace**
 - I modificatori vengono usati come parametri per i metodi
- ❑ Esempi:
 - `var stringa = "Tecnologie Web";`
 - `var pos = stringa.search(/c/);` → `pos = 2`
 - `stringa.replace(/e/, "E");` → `stringa = "TEcnologie Web"`
 - `stringa.replace(/e/g, "E");` → `stringa = "TEcnologiE WEb"`
 - `var parole = stringa.split(" ");` → `["Tecnologie", "Web"]`



Browser Object Model (BOM)

- Il BOM è un modello ad oggetti, **non standardizzato** e privo di specifica che consente di interagire con il browser

Elemento	Oggetto
Browser	navigator
Finestra	window
Frame	window.frames["ID Frame"]
Barra indirizzi	location
Barra di stato	status



Metodi BOM

- ❑ L'oggetto window rappresenta la finestra (o scheda) del browser. Può essere omesso nella chiamata alle sue proprietà o metodi perché usato implicitamente
- ❑ La funzione **open** permette di aprire una nuova finestra



Apertura di una nuova finestra - 1

- ❑ Questa soluzione si trasforma elegantemente perchè se javascript non è abilitato apre il link nella stessa pagina

```
<a href="http://www.example.com/"  
onclick="popUp(this.href); return false;">Example</a>
```

```
function popUp(winURL) {  
    window.open(winURL,"popup","width=320,height=480");  
}
```

```
window.open(url, nome, lista_di_features);
```

- ❑ Per dispositivi touch: ontouchend



Apertura di una nuova finestra - 2

- ❑ Questa soluzione preserva la separazione comportamento - struttura

```
<a href="http://www.example.com/"  
      class="popup">Example</a>
```

```
var links = document.getElementsByTagName("a");  
for (var i=0; i<links.length; i++) {  
    if (links[i].getAttribute("class") == "popup") {  
        links[i].onclick = function() {  
            popUp(this.getAttribute("href"));  
            return false;  
        }  
    }  
}
```



Connessione script - pagina HTML

```
window.onload = linkNuovaFinestra;
```

```
function linkNuovaFinestra(){  
    var links = document.getElementsByTagName("a");  
    for (var i=0; i<links.length; i++) {  
        if (links[i].getAttribute("class") == "popup") {  
            links[i].onclick = function() {  
                popUp(this.getAttribute("href"));  
                return false;  
            }  
        }  
    }  
}  
  
function popUp(url){  
    window.open(url, "nuovaFinestra", "width=320,height=480");  
}
```



Document Object Model (DOM)

- ❑ Il DOM permette di accedere ai diversi elementi di una pagina web. È uno standard, con supporto ormai completo.
- ❑ La pagina è divisa in vari elementi in relazione tra loro

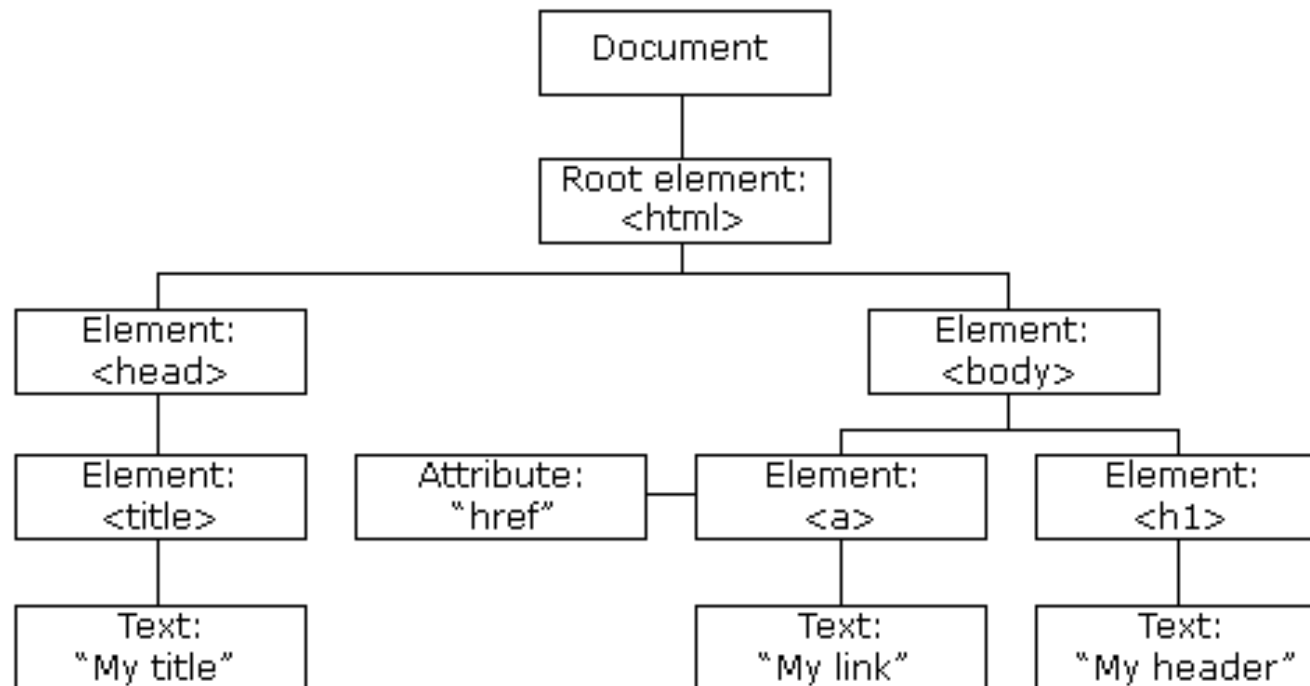
Elemento	Oggetto
Pagina web	window.document o document
Form	document.forms["ID form"]
Immagini	document.images["ID immagine"]

<http://www.quirksmode.org/dom/core/>



Document Object Model (DOM)

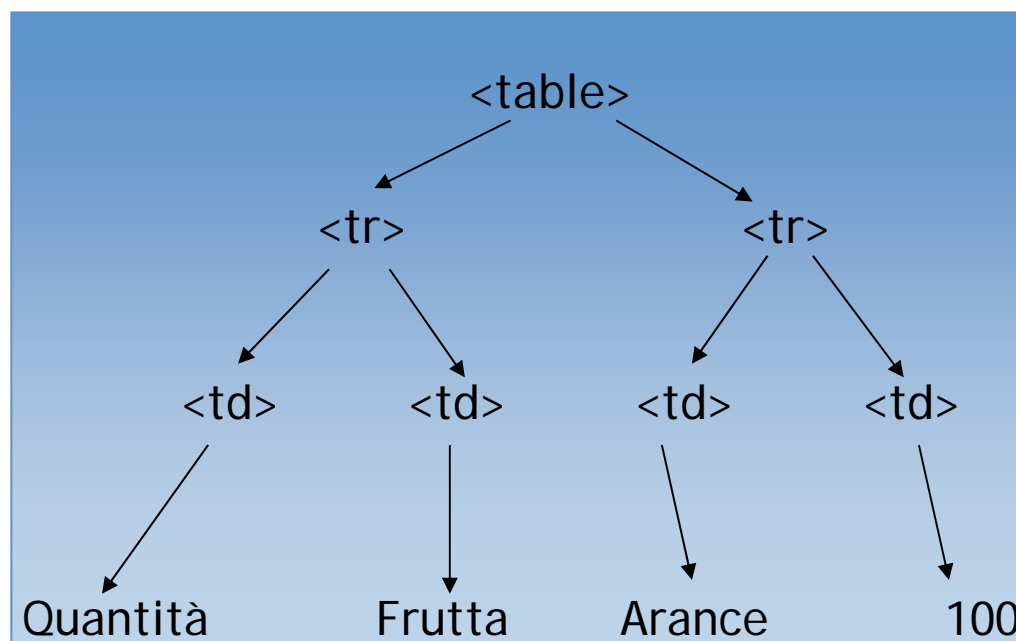
- Il DOM HTML permette di modificare, aggiungere o rimuovere elementi HTML in modo standard.



JavaScript e HTML

- ❑ Posso accedere ad un tag tramite `getElementsByTagName` o `getElementById`
- ❑ Ogni elemento dell'array `forms`, contiene un array `elements` con gli elementi del form (pulsanti, caselle di testo, etc.)
- ❑ Un documento DOM ha una struttura ad albero

```
<table>
  <tr>
    <td> Frutta </td>
    <td> Quantità </td>
  </tr>
  <tr>
    <td> Arance </td>
    <td> 100 </td>
  </tr>
</table>
```



Proprietà e metodi

□ Proprietà

- `x.innerHTML` → testo contenuto nell'elemento `x`
- `x.nodeName`, `x.nodeValue` → nome/valore del nodo `x`
- `x.parentNode` → nodo padre del nodo `x`
- `x.childNodes` → array contenente i figli di `x`
- `x.firstChild`, `x.lastChild` → primo/ultimo figlio di `x`
- `x.attributes`, `x.getAttribute(y)` → attributi di `x`, attributo `y`
- `x.nextSibling`, `x.previousSibling`

□ Metodi

- `x.getElementById(id)`
- `x.getElementsByTagName(name)` → restituisce tutti i tag di un certo tipo
- `x.appendChild(node)` → inserisce un figlio in coda
- `x.removeChild(node)`



Gli eventi

- ❑ Nella programmazione tradizionale il codice contiene in sé l'ordine in cui viene scritto
- ❑ Nella programmazione ad eventi si specificano funzioni (**event handler**) da eseguire all'occorrenza di un determinato evento
 - simile alla gestione delle eccezioni
 - ```
function unload_saluto(){
 alert("Grazie per aver visitato il nostro sito!");
}
```
- ❑ È importante scrivere correttamente l'evento a cui si vuole rispondere
- ❑ event handler o chiamata di funzione su di un attributo specifico
  - `onclick="alert('Stai uscendo da questo sito.');"`



# Attributi ed eventi

---

| Attributo   | Tag                                        |
|-------------|--------------------------------------------|
| onabort     | <img>                                      |
| onblur      | <body>, <form>, <frameset>, <frame>, etc.  |
| onchange    | <input>, <textarea>, <select>              |
| onclick     | <a>, <input>                               |
| onerror     | <img>, <body>, <frameset>                  |
| onfocus     | <body>, <frameset>, <frame>, <input>, etc. |
| onload      | <img>, <body>, <frameset>                  |
| onmouseover | <a>, <area>                                |



# Eventi JavaScript

---



## Mouse

mousedown  
mousemove  
mouseup  
mouseover  
mouseout



## Keyboard

keydown  
keypress  
keyup  
focus  
blur



## Touch

touchstart  
touchmove  
touchend  
–  
–



## Esempio: event handler

---

```
<script type="text/javascript">
 <!--
 function voto(){
 if (document.getElementById('trump').checked)
 alert('Questa mi sembra una buona scelta!');
 if (document.getElementById('biden').checked) {
 alert('Sei davvero sicuro della tua scelta?');
 document.getElementById('biden').checked = true;
 }
 }
 //-->
</script>
```

<h1> Vota per il presidente degli Stati Uniti. </h1>

```
<input type="radio" name="vote" value="trump" id="trump"
onclick="voto();" />
<input type="radio" name="vote" value="biden" id="biden"
onclick="voto();" />
```



## Esempio: controllo dell'input di un form

```
function check() {
 if (document.getElementById('cf').value==""){
 alert('Inserisci il tuo codice fiscale, grazie.'); document.getElementById('cf').focus(); return false;
 ...oppure...
 document.forms['id_form']['cf'].focus();
 }
 var elem = document.getElementById('cf').value;
 var pos =elem.search(/^[A-Z]{6}\d{2}[A-Z]\d{2}[A-Z]\d{3}[A-Z]$/);
 if (pos != 0){
 alert('Codice Fiscale non inserito correttamente; riprova.'); document.getElementById('cf').focus();
 document.getElementById('cf').select();
 return false;
 } else return true;
}
```

...

```
<input type="text" id="cf" name="cf" onchange="check();" />
```



# Gestire diversi dispositivi di input

---

- Gli event pointer sono un'astrazione per le interazioni basate su puntatore (mouse, stilo, tocco, altri futuri) che utilizzano eventi generici indipendenti dal puntatore

```
window.addEventListener('pointerdown', detectInput, false)
```

```
function detectInput(event){
 switch(event.pointerType){
 case 'mouse':
 ...
 break;
 case 'touch':
 ...
 break;
 }
}
```

<https://www.w3.org/TR/pointerevents/>



## Pagine dinamiche con JavaScript

---

- ❑ Il linguaggio JavaScript permette di posizionare e dimensionare gli oggetti contenuti nel documento HTML
- ❑ Può quindi essere usato per creare dinamicamente il documento in fase di caricamento (**onload**)
- ❑ La modifica dinamica lato client delle pagine web deve avvenire attraverso script non intrusivo



# Cosa si può rendere dinamico?

---

- Sulla base delle:
  - caratteristiche del browser
  - dimensioni della pagina
  - dell'input dell'utente
  - degli spostamenti del mouse e degli eventi in genere
- JavaScript è in grado di modificare
  - la posizione e la dimensione degli elementi
  - le caratteristiche di stile (colore, font disponibili, etc.)
  - il contenuto e la sua struttura
- Oppure può dare dei messaggi di aiuto e/o avviso





## Esempio

---

- ❑ `<p>Questo <a style = "color:green"  
onmouseover="this.style.color = 'orange';  
this.style.font = 'normal 20pt Verdana'; "  
onmouseout="this.style.color = 'green';  
this.style.font = 'normal 12pt Times';">  
link</a>.  
cambia colore e font quando vi è sopra il mouse.</p>`
- ❑ **Nota:** questo stesso comportamento si può realizzare utilizzando semplicemente i CSS. In questi casi è in generale scorretto utilizzare Javascript



# L'oggetto Navigator

---

- ❑ Indica quale browser sta utilizzando l'utente.
  - appName indica il nome del browser
  - appVersion indica la versione
- ❑ Esempio:
  - `alert("Il browser usato è: "+navigator.appName + "\n" + navigator.appVersion + "\n");`
- ❑ Conoscere quale browser si sta utilizzando è utile perché in alcuni casi bisogna predisporre codice differente per i diversi browser
  - *code forking*: da non utilizzare, ma diventa inevitabile se gli oggetti non sono definiti in modo comune nel DOM



## Identificare il browser: un'altra possibilità

---

```
var op = navigator.userAgent.indexOf("Opera");
if (document.layers){// se il browser e' Netscape 4xx
 document.write('<link rel=')
}else if(op >-1) {// se il browser e' Opera
 document.write('<link rel="stylesheet" href=... >')
}else if (document.all){// se il browser e ' Msie
 document.write('<link rel="stylesheet" href=...>')
}else if (document.getElementById){
 // se il browser e' Netscape 6xx
 document.write('<link rel="stylesheet" href="CDSstilen7.css"
 type="text/css">')
}
```



# Differenze tra i diversi browser

---

- ❑ Accesso agli elementi
- ❑ Ereditarietà degli attributi di carattere
  - Ex. lo stile specificato per il body non viene ereditato dalle tabelle
  - Netscape 4.X disegna i font più piccoli di circa un pixel
- ❑ Fogli di stile predefiniti di browser diversi sono diversi

```
<script language="javascript">
 var link='<link rel="stylesheet" type="text/css" href="';
 var css;
 if (ie4) { css = link + 'ie4.css">'};
 if (ie5) { css = link + 'ie5.css">'};
 if (ns4) { css = link + 'ns4.css">'};
 document.write(link+"\n");
</script>
<noscript> <link rel="stylesheet" type="text/css" href="gen.css">
</noscript>
```



## La strada giusta

---

- ❑ Individuare di quale browser si tratta vuol dire richiedere un continuo aggiornamento dello script
- ❑ Uno script che testa il supporto al DOM non richiede aggiornamento

```
if (!document.getElementById){
 window.location = "http://www.sito.it/altra_pagina.html";
}
```



## Libreria Modernizr

---

- ❑ Il supporto ad HTML5 e CSS3 non è ancora completo da parte di tutti i browser, inoltre c'è il problema dei browser non aggiornati (IE supporta HTML5 a partire dalla versione 9)
- ❑ Modernizr è una libreria opensource che aiuta a testare le funzionalità e non il tipo di browser utilizzato

```
if (Modernizr.video) {
 //video supportato
} else { //video non supportato }
```

```
function isTagVideoSupported() {
 return !!document.createElement("video").canPlayType;
}
```



# Spedire una mail con JavaScript

---

```
function Email() {
 var email = document.getElementById('email').value;
 var oggetto = document.getElementById('oggetto').value;
 var testo = document.getElementById('testo').value;
 var pos = email.search(/^([\w\-\+\.\.])+@([\w\-\+\.\.])+([\w\-\+\.\.])+$\/);
 if (pos != 0) {
 alert('Inserire un indirizzo Email valido!');
 document.getElementById('email').value = "";
 document.getElementById('email').focus();
 }
 else if (testo == "") {
 alert('Il campo \"Messaggio\" è obbligatorio!');
 document.getElementById('testo').focus()
 }
 else {
 location.href = 'mailto:' + email + '?Subject=' + oggetto +
 '&Body=' + testo;
 }
}
```



## Ordinare un array

---

```
<button onclick="myFunction()">Ordina in modo ascendente</button>
```

```
<p id="demo"></p>
```

```
<script>
```

```
 var points = [40, 100, 1, 5, 25, 10];
```

```
 document.getElementById("demo").innerHTML = points;
```

```
 function myFunction() {
```

```
 points.sort(function(a, b){return a-b});
```

```
 document.getElementById("demo").innerHTML = points;
```

```
 }
```

```
</script>
```





# Attenzione all'uso di JavaScript - 1

---

```
<script language="text/javascript">
 <!--
 var aiuti = ["Inserisci il nome in questo modo:\n \t nome cognome",
 "L'email deve avere questa forma: \n \t login@dominio",
 "La login deve avere almeno 6 caratteri",
 "La password deve contenere almeno 6 caratteri e deve
 contenere almeno un valore numerico",
 "Qui ci sono i messaggi di aiuto per compilare la form.
 Metti il mouse sopra un campo per ottenere l'aiuto"];

 function messages(n_messaggio){
 document.getElementById("aiuti").value=aiuti[n_messaggio];
 }
 //-->
</script>
```



## Attenzione all'uso di JavaScript - 2

---

```
<form id="f_email" action="#" >
 <fieldset>
 <label for="nome"> Nome:</label> <input type="text" id="nome"
 onmouseover="messages(0)" onmouseout="messages(4)" />
 <label for="email">Email:</label> <input type="text" id="email"
 onmouseover="messages(1)" onmouseout="messages(4)" />
 <label for="login">Login:</label> <input type="text" id="login"
 onmouseover="messages(2)" onmouseout="messages(4)" />
 <label for="passwd">Password:</label> <input type="text" id="passwd"
 onmouseover="messages(3)" onmouseout="messages(4)" />
 <label for="aiuti">Istruzioni:</label>
 <textarea id="aiuti" rows="4" cols="50" ></textarea>
 </fieldset>
</form>
```



## Un esempio più complesso - 1

---

- Si vuole popolare questa tabella con le informazioni sul browser

```
<table id="table">
 <tr><td>appName</td><td></td></tr>
 <tr><td>appVersion</td><td></td></tr>
 <tr><td>userAgent</td><td></td></tr>
 <tr><td>platform</td><td></td></tr>
 <tr><td>onLine</td><td></td></tr>
 <tr><td>geolocation</td><td></td></tr>
 <tr><td>javaEnabled()</td><td></td></tr>
 <tr><td>cookieEnabled</td><td></td></tr>
</table>
```



## Un esempio più complesso - 2

---

```
function giveInformations(){
 var table = document.getElementById("table");
 var i;
 var _n = window.navigator; // oggetto navigator
 // array delle info del browser
 var b_infos = [_n.appName, _n.appVersion, _n.userAgent,
 _n.platform, _n.onLine, _n.geolocation,
 _n.javaEnabled(), _n.cookieEnabled];
 // tr della tabella
 var trs = table.getElementsByTagName("tr");
 for(i = 0; i < trs.length; i++) {
 // popola ogni seconda cella
 var td_2 = trs[i].getElementsByTagName("td")[1];
 td_2.textContent = b_infos[i];
 }
}
```



# I cookie

---

- ❑ I cookie sono piccoli file di testo memorizzati sul computer dell'utente, e scambiati tra client e server, che contengono informazioni salvate dai siti web.
- ❑ Sono usati per memorizzare in modo permanente delle informazioni univoche rispetto ad un utente, in modo da poterlo riconoscere e/o poterle riusare
  - problemi di privacy
- ❑ Si deve far attenzione al fatto che i cookie possono essere eliminati o disabilitati dall'utente.
- ❑ Ogni cookie contiene dei parametri, tra cui:
  - **nome**: un nome identificativo per il cookie
  - **valore**: il valore da memorizzare
  - **scadenza** (*expiration date*): è opzionale, stabilisce la data di scadenza del cookie, cioè la data dopo la quale questi vengono eliminati dal disco rigido dell'utente.



# Creazione e distruzione di un cookie

---

// imposta il cookie con nome = valore per la durata di giorni

```
function setCookie(nome, valore, giorni) {
 var oggi = new Date();
 var scadenza= new Date();
 scadenza.setTime(oggi.getTime() + 24 * giorni * 3600000);
 document.cookie = nome + "=" + escape(valore) + "; expires=" +
 scadenza.toGMTString();
}
```

// rimuove un cookie

```
function delCookie(nome) {
 setCookie(nome, "");
}
```



## Accesso ad un cookie

---

```
// restituisce il valore del cookie nome
function getCookie(nome) {
 // genera un array di coppie "Nome = Valore" separate da ';'
 var asCookies = document.cookie.split("; ");
 var stringa="";
 // ciclo su tutti i cookies
 for (var i = 0; i < asCookies.length; i++){
 // leggo singolo cookie "Nome = Valore"
 var info = asCookies[i].split("=");
 if (nome == info[0]) {
 stringa = unescape(info[1]);
 }
 }
 return stringa;} //stringa="" se il cookie non esiste
```



# Utilizzo dei cookie per riconoscere un utente



## TECNOLOGIE WEB

dott. ssa Ombretta Gaggi  
Dipartimento di Matematica Pura ed Applicata  
Università di Padova

Bentornato Ombretta, ti trovi in: Home

- [Programma del Corso](#)
- [Materiale didattico](#)
- [Iscrizione al corso](#)
- [Esami](#)

---

[Crea un cookie con il tuo utente](#)

[Cancella il cookie con il](#)

### ORARIO DELLE LEZIONI

lunedì: 11.30 - 13.30, Aula P200  
martedì: 11.30 - 13.30, Aula P200  
mercoledì: 11.30 - 13.30, Aula P200

### LEZIONI DI LABORATORIO

venerdì 30/01/2009 11.30 - 13.30 laboratorio C (Paolotti)  
lunedì 9/03/2009 11.30 - 13.30 laboratorio C (Paolotti)





# Utilizzo dei cookie per riconoscere un utente

---

```
function legginome(){
 var nome=prompt("Inserisci il nome");
 setCookie("utente",nome,2);
 leggiutente();
}
function leggiutente(){
 var utente=getCookie("utente");
 if (utente!="") stringa_path="Bentornato " + utente + "...";
 document.getElementById("path").innerHTML =stringa_path;
}
```

```
<body onload="leggiutente();">
```

...

```
Crea un cookie con il tuo utente
```



## Altro modo di memorizzare i dati sul client

---

- ❑ Il sempre più frequente uso dei telefonini impone di non fare sempre affidamento sulla rete
  - L'utente si sposta, anche in zone dove non è presente il segnale
- ❑ I cookie hanno un limite di 4093KB per tutti i cookie di un dominio
- ❑ Soluzione: `localStorage`, `sessionStorage`, `indexedDB`

```
If ('localStorage' in window && window.localStorage !==null){
 //possiamo usare localStorage
 localStorage.setItem('key', 'value');
 localStorage.setItem('myObject', JSON.stringify(myObject));
 ...
 var myValue = localStorage.getItem('key');
 var myObject = JSON.parse(localStorage.getItem('myObject'));
}
```



# AJAX

---

- AJAX è l'acronimo di *Asynchronous JavaScript and XML* e realizza uno scambio asincrono di dati
  - lo scambio di dati avviene in background
  - non è richiesto di ricaricare la pagina
- La pagina HTML chiamante deve contenere
  - la chiamata allo script JavaScript (pulsante o evento)
  - un elemento (ex. div, select, etc) dove inserire il contenuto



# AJAX - creazione oggetto XMLHttpRequest

---

```
function getXMLHttpRequest(){
 var xmlHttp;
 try{
 xmlHttp = new XMLHttpRequest();
 }catch(e){ //Internet Explorer usa un oggetto ActiveX
 try{
 xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
 }catch(e){
 try{
 xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
 }catch(e){
 alert("Messaggio di errore per i vecchi browsers");return false;
 }
 }
 }
 return xmlHttp;}
}
```



## AJAX - esempio

---

```
function AjaxRequest(){
 var xmlHttp = getXMLHttp();
 xmlHttp.onreadystatechange =
 function(){ //quando l'operazione è completata
 if(xmlHttp.readyState ==4){
 inserisciTesto(xmlHttp.responseText);
 }
 }
 xmlHttp.open("GET", "ScriptPHP.php", true);
 xmlHttp.send(null);
}
function inserisciTesto(response){
 document.getElementById('IDdoveInserire').innerHTML = response;
}
```



# jQuery

---

- ❑ Library JavaScript per manipolazione di HTML, CSS ed eventi
- ❑ \$('qualcosa') restituisce un array di elementi ([n])
  - #id
  - .classe
  - Elemento (li, a, div)
- ❑ \$('#formInput').on(evento, function() {})
- ❑ Modifica dello stile tramite style o con addClass/removeClass
- ❑ \$(document).ready() eseguito solo quando il DOM è pronto per essere manipolato



## Utilizzo di librerie esterne

---

- ❑ È molto importante assicurarsi che le librerie esterne siano disponibili perché ci sono stati diversi esempi nella storia del web di blackout di siti dovuti all'indisponibilità di una libreria esterna
  - Gawker Media, 2011
  - Sky Broadband, 2014
  - Morale: ci sono troppi fattori su cui non abbiamo il controllo quando eseguiamo codice su browser

```
<script src="http://ajax.googleapis.com/path/jquery.js"></script>
<script>windows.jQuery || document.write('<script
 src="mioserver/jquery.js"></script>')</script>
```

```
If(typeof(jQuery) == 'undefined'){ return;}
```



# Alcune regole per il progressive enhancement

---

- ❑ Progettare una versione base che funzioni senza JavaScript
  - Anche nel caso di errore più catastrofico gli utenti saranno in grado di eseguire le funzionalità di base
- ❑ Programmare sulla difensiva
  - A differenza di HTML o CSS, JavaScript non è tollerante ai guasti, basta un errore e si ferma l'esecuzione
  - Guardare prima di agire: non si può dare per scontata la presenza di altri elementi oltre a `html`, `head` e `body`, si deve cercare un elemento prima di usarlo
  - Delegare i comportamenti: l'ascolto degli eventi può essere fatto sull'elemento `body` che poi cerca l'elemento a cui siamo interessati
  - Testare il supporto alle caratteristiche
  - Controllare la presenza delle librerie
  - Stabilire requisiti minimi
  - Tagliare le perdite





# Event delegation

---

- ❑ *Event delegation* è un pattern molto utile per gestire il DOM
- ❑ Consiste nel delegare la gestione di una risposta ad un evento uguale per molti elementi ad un elemento contenitore:
  - Si mette l'handler nell'elemento contenitore
  - Si verifica che elemento ha scatenato l'evento (*event.target*)
  - Se l'evento si è verificato all'interno di un elemento che dobbiamo gestire, si esegue la funzione associata con l'handler



## Esempio

---

```
container.onclick = function(event) {
 if (event.target.className != 'remove-button') return;

 let pane = event.target.closest('.pane');
 pane.remove();
};
```

### Horse

[x]

The horse is one of two extant subspecies of *Equus ferus*. It is an odd-toed ungulate mammal belonging to the taxonomic family Equidae. The horse has evolved over the past 45 to 55 million years from a small multi-toed creature, *Eohippus*, into the large, single-toed animal of today.

### Donkey

[x]

The donkey or ass (*Equus africanus asinus*) is a domesticated member of the horse family, Equidae. The wild ancestor of the donkey is the African wild ass, *E. africanus*. The donkey has been used as a working animal for at least 5000 years.

### Cat

[x]

The domestic cat (Latin: *Felis catus*) is a small, typically furry, carnivorous mammal. They are often called house cats when kept as indoor pets or simply cats when there is no need to distinguish them from other felids and felines. Cats are often valued by humans for companionship and for their ability to hunt vermin.



# Pro e contro

---

## □ Pro

- Semplifica e diminuisce i bit: si usa un solo handler
- Maggiore flessibilità: se si aggiungono o rimuovono componenti, non c'è necessità di modifiche al codice

## □ Contro

- Maggiore richiesta di CPU perché l'handler sul container potrebbe attivarsi anche per eventi, a qualunque livello, che in realtà non andrebbero gestiti



# Controllare i requisiti minimi

---

```
If ('querySelector' in document &&
 'localStorage' in windows &&
 'addEventListener' in window) {
```

```
 //browser HTML5
```

```
}
```



# Bibliografia

---

- ❑ Siti ufficiale (in inglese)
  - <http://www.javascript.com/>
  - <http://modernizr.com/>
  - <http://jquery.com/>
- ❑ Tutorial W3C
  - <http://www.w3schools.com/js/default.asp>
  - <http://www.w3schools.com/html/default.asp>
  - [http://www.w3schools.com/xml/ajax\\_intro.asp](http://www.w3schools.com/xml/ajax_intro.asp)
- ❑ Tutorial in italiano (datati e non sempre corretti)
  - <http://javascript.html.it/guide/leggi/25/guida-javascript-di-base/>
  - <http://javascript.html.it/guide/leggi/26/guida-javascript-per-esempi/>

