



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA

# Introduzione a Javascript Il parte

**Fabio Vitali**

Corsi di laurea in Informatica e  
Informatica per il Management  
Alma Mater – Università di Bologna

# Oggi parleremo di...

## Javascript

- Sintassi base (parte I)
- ***Modello oggetti del browser (parte II)***

## AJAX (parte III):

- Architettura di riferimento
- XMLHttpRequest

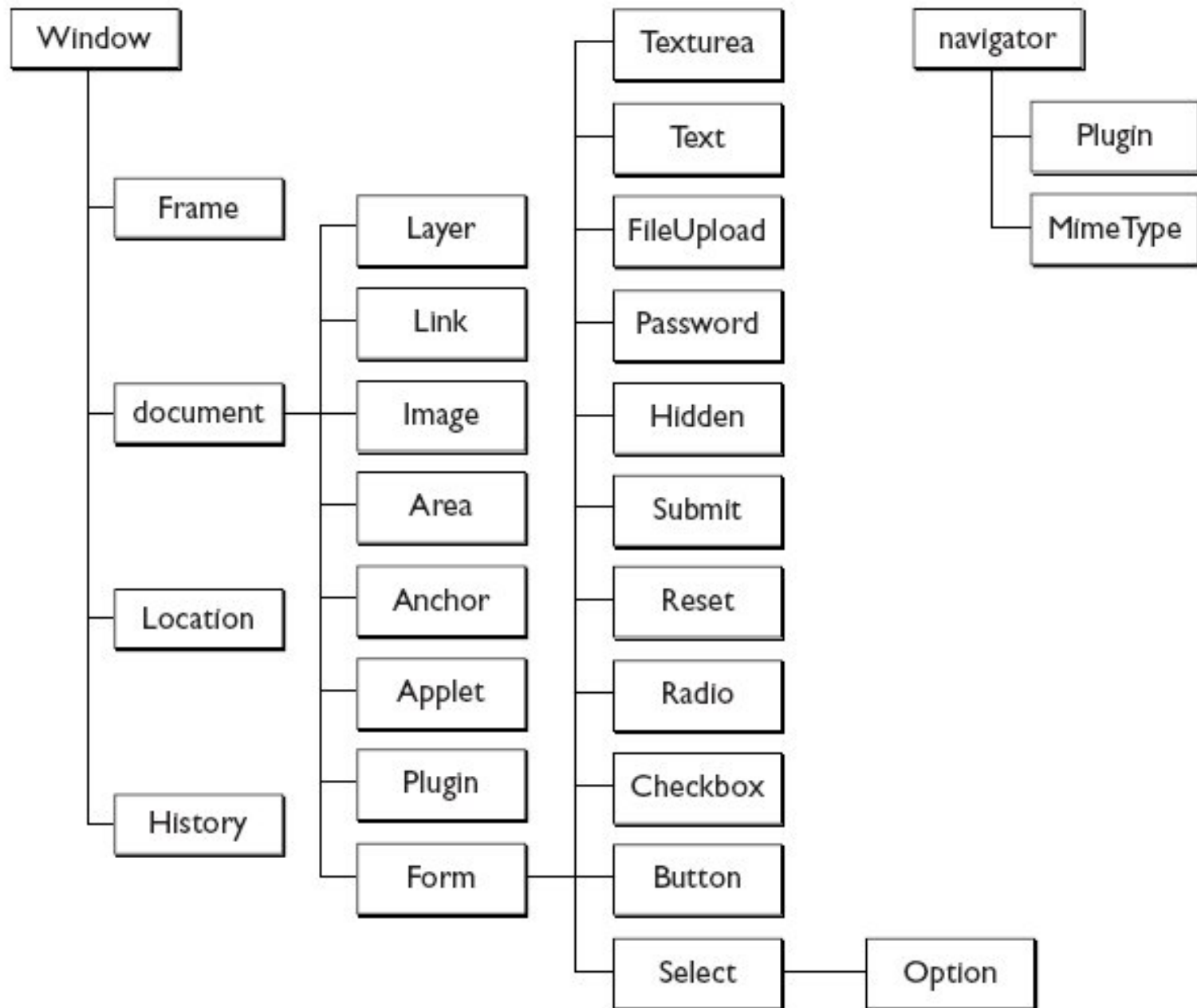




ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA

# Javascript client side

# Gli oggetti predefiniti del browser



# Gli oggetti principali:

## `window` e `navigator`

- **window**: è l'oggetto top-level con le proprietà e i metodi della finestra principale:
  - posizione: `moveBy(x,y)`, `moveTo(x,y)`, etc.
  - dimensioni: `resizeBy(x,y)`, `resizeTo(x,y)`, etc.
  - altre finestre: `open("URLname","Windowname",["opt"])`
- **navigator**: è l'oggetto con le proprietà del client come nome, numero di versione, plug-in installati, supporto per i cookie, etc.



# Gli oggetti principali:

## `location` e `history`

- **location:** l'URL del documento corrente. Modificando questa proprietà il client accede a un nuovo URL (redirect):
  - `window.location = "http://www.cs.unibo.it/";`
  - `window.location.href = "http://www.cs.unibo.it/";`
- **history:** l'array degli URL acceduti durante la navigazione. Possibile creare applicazioni client-side dinamiche che 'navigano la cronologia':
  - Proprietà: `length`, `current`, `next`
  - Metodi: `back()`, `forward()`, `go(int)`



# Gli oggetti principali:

## document

***document*** rappresenta il contenuto del documento, ed ha proprietà e metodi per accedere ad ogni elemento nella gerarchia:

- document.title: titolo del documento
- document.forms[0]: il primo form
- document.forms[0].checkbox[0]: la prima checkbox del primo form
- document.forms[0].check1: l'oggetto con nome "check1" nel primo form (non per forza una checkbox!)
- document.myform: l'oggetto "myform"
- document.images[0]: la prima immagine



# Modello di documento

Ogni oggetto nella gerarchia è caratterizzato da un insieme di proprietà, metodi ed eventi che permettono di accedervi, controllarlo, modificarlo.

```
function Verify() {  
    if (document.forms[0].elements[0].value == ""){  
        alert("Il nome è obbligatorio!")  
        document.forms[0].elements[0].focus();  
        return false;  
    }  
    return true;  
}
```

```
<FORM ACTION= "..." onSubmit="Verify()">  
<P>Name: <INPUT TYPE="text" NAME="nome" ...> </P>
```





# Javascript e DOM

Javascript implementa i metodi standard per accedere al DOM del documento.

```
var c = document.getElementById('c35');  
  
c.setAttribute('class', 'prova1');  
c.removeAttribute('align') ;  
  
var newP = document.createElement('p');  
  
var text = document.createTextNode('Ciao Mamma.');
```

```
newP.appendChild(text);  
  
c.appendChild(newP);
```



```
// Creazione elementi singoli
olist = document.createElement("ol");
voce1 = document.createElement("li");
voce2 = document.createElement("li");
testo1 = document.createTextNode("un po' di testo");
testo2 = document.createTextNode("altro testo - item 2");

// Creazione lista completa

voce1.appendChild(testo1);
voce2.appendChild(testo2);
olist.appendChild(voce1);
olist.appendChild(voce2);

// Inserimenti lista in una data posizione

div = document.getElementById("lista");
body = document.getElementsByTagName("body").item(0);
body.insertBefore(olist,div);
```



# Javascript ed eventi DOM

Javascript permette di associare callback di eventi ad oggetti (dichiarazione locale o globale)

```
...
<script language="JavaScript">
window.onkeypress= pressed;
window.document.onClick = clicked;

function pressed(e) { alert("Key pressed: " + e.which);}
function clicked() { alert("Mouse Click! "); }

</script>
</head>
<body>
  <p id="mypara">Puoi
    <a href="test.htm" onClick="alert('Link!');">
      cliccare qui
    </a>
    oppure qui.
  </p>
</body>
```



# innerHTML e outerHTML

Javascript (non DOM!) permette di leggere/scrivere interi elementi, trattandoli come stringhe:

- **innerHTML**: legge/scrive il contenuto di un sottoalbero (escluso il tag dell'elemento radice)
- **outerHTML**: legge/scrive il contenuto di un elemento (incluso il tag dell'elemento radice)

```
// HTML: <div id="d"><p>Paragrafo!</p></div>
d = document.getElementById("d");
alert(d.innerHTML);           // <p>Paragrafo!</p>
alert(d.outerHTML);          // <div id="d"><p>Paragrafo!</p></div>
d.innerHTML = "<ul><li>Lista!</li></ul>";
```



# Selettori in DOM

I metodi standard in DOM per accedere ai nodi di un documento sono essenzialmente:

- **getElementById**: solo ovviamente se l'elemento ha un id
- **getElementsByTagName**: se l'elemento ha un attributo name
- **getElementsByTagName**: tutti gli elementi con nome specificato

Il successo di JQuery ha portato nel tempo a proporre ed implementare in DOM HTML anche due nuovi selettori:

- **getElementsByTagName**; cerca tutti gli elementi di classe specificata
- **querySelector**: accetta un qualunque selettore CSS e restituisce il primo elemento trovato - del tutto equivalente a `$()[0]` in JQuery
- **querySelectorAll**: accetta un qualunque selettore CSS e restituisce tutti gli elementi trovati - del tutto equivalente a `$()` in JQuery



# Un piccolo esercizio

Andiamo a creare una calcolatrice digitale

Troviamo tutto a:

<http://www.fabioitali.it/TW/2021/calculator/>





ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA

**Fabio Vitali**

Dipartimento di Informatica – Scienze e Ingegneria  
Alma mater – Università di Bologna

Fabio.vitali@unibo.it

[www.unibo.it](http://www.unibo.it)