

Sapienza Università di Roma  
corso di laurea in Ingegneria informatica e automatica

# **Linguaggi e tecnologie per il Web**

a.a. 2023/2024

## **Parte 4**

# **JavaScript nel World Wide Web**

Lorenzo Marconi

# Integrazione con i browser web

- La caratteristica principale di JavaScript è di essere integrabile all'interno delle pagine web
- In particolare consente di aggiungere una logica procedurale alle pagine rendendole "dinamiche"
- È stato introdotto nel Web per la **programmazione lato client**: il codice JavaScript viene eseguito dal web client (browser)
- In seguito è stato utilizzato diffusamente anche per la programmazione lato server (es Node.js)

# Integrazione con i browser web

- I campi di impiego tradizionali sono
  - Validazione dell'input dell'utente e controllo dell'interazione
  - Effetti visivi di presentazione
- Con l'avvento di HTML5 i potenziali usi di JavaScript sono notevolmente aumentati
- JavaScript è supportato da tutti i browser più diffusi

# JavaScript e HTML

- L'integrazione degli script all'interno di una pagina HTML avviene in due modi
  - Associando l'esecuzione di funzioni JavaScript agli **eventi** collegati alla pagina che si intende gestire
  - Accedendo dalle funzioni JavaScript alle proprietà degli oggetti che costituiscono la pagina

# JavaScript in documenti HTML

- Il codice JavaScript viene inserito all'interno di una pagina HTML delimitato dall'elemento `<script>...</script>`
- Oppure è memorizzato in una risorsa (file) a parte e viene linkato dalla pagina web sempre attraverso l'elemento `<script>`:  

```
<script type="application/javascript"
      src="myscript.js"></script>
```
- Il media type `application/javascript` viene assegnato per default

# JavaScript in documenti HTML

- Il codice incorporato nel documento solitamente viene incluso all'interno dell'intestazione (<HEAD>)
- Il codice viene eseguito **prima** della visualizzazione del documento
- In questa sezione si procede con la definizione delle funzioni e delle variabili globali

# JavaScript in documenti HTML

```
<HTML>
<HEAD>
  <SCRIPT TYPE="application/javascript">
    function f(...);
    var v=...;
    ...
  </SCRIPT>
  ...
</HEAD>
  ...
</HTML>
```

# Modello ad oggetti

- Un browser web esporta verso JavaScript un modello ad oggetti della pagina (DOM) e dell'"ambiente" (BOM) in cui la pagina è visualizzata
- Una funzione JavaScript adopera tali oggetti invocando i metodi e accedendo alle proprietà



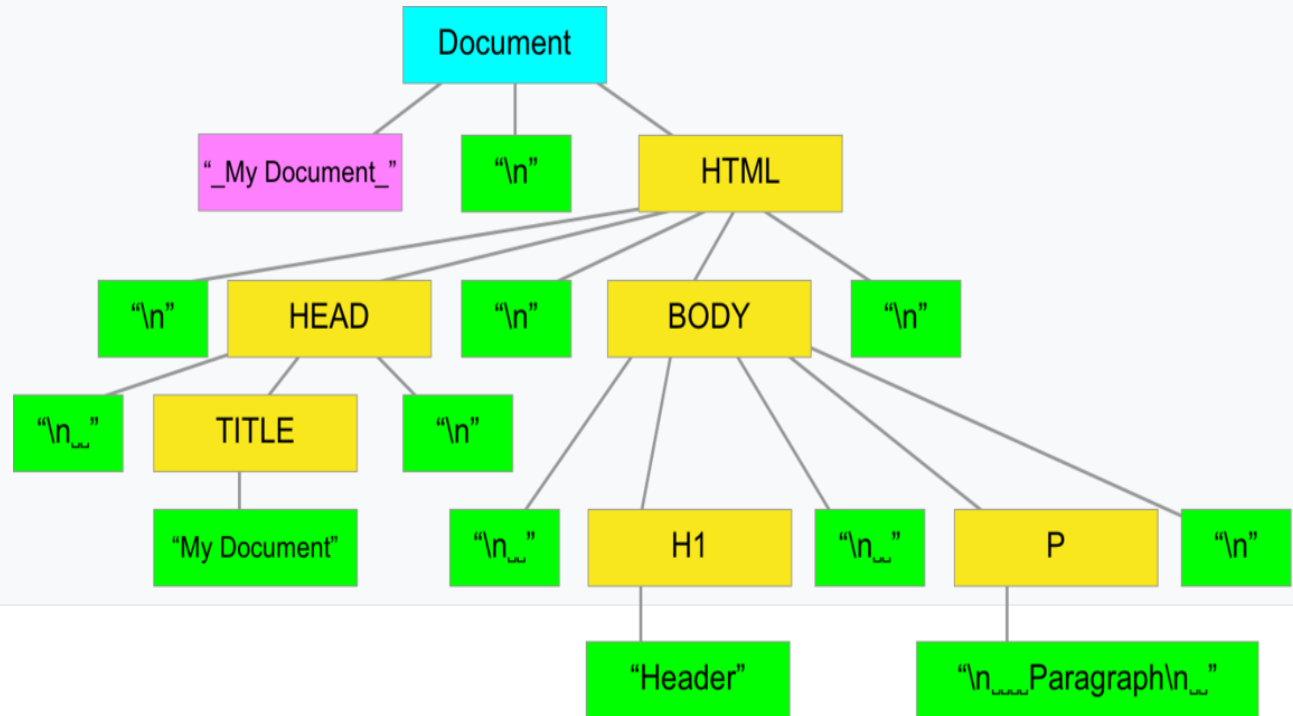
# DOM e BOM

- DOM = Document Object Model
  - Corrisponde in pratica all'oggetto `document` che vedremo nel seguito
  - Il DOM HTML è uno standard (è parte dello standard HTML)
- BOM = Browser Object Model
  - Corrisponde in pratica all'oggetto `window` che vedremo in seguito
  - Il BOM non è uno standard (ma i browser si basano essenzialmente sullo stesso BOM)

# Documento HTML = albero

- Nel DOM il documento HTML è rappresentato come un albero:

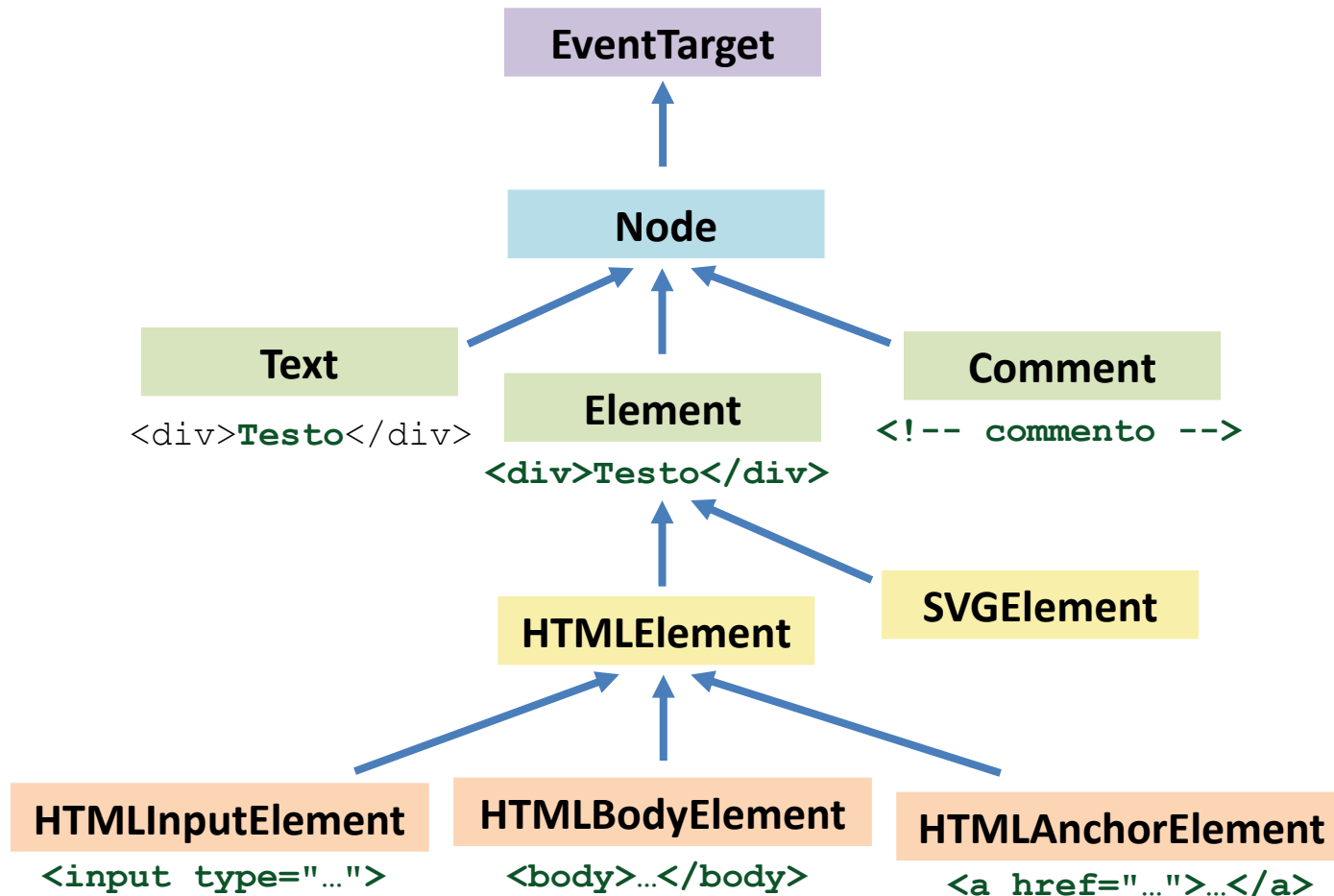
```
<!-- My document -->
<HTML>
<HEAD>
  <TITLE>My Document</TITLE>
</HEAD>
<BODY>
  <H1>Header</H1>
  <P>
    Paragraph
  </P>
</BODY>
</HTML>
```



# DOM: nodi ed elementi

- Nodi ed elementi NON sono sinonimi
- Un nodo può essere
  - un elemento (ad es. un elemento `div` o un elemento `img`)
  - un testo
  - un commento
- In JavaScript, i nodi sono rappresentati da oggetti di tipo `Node`

# Gerarchia dei tipi di nodo



# Oggetti di tipo Node

## Proprietà:

- `childNodes` / `children` ritornano l'elenco dei nodi/elementi figli
- `firstChild` / `lastChild` ritornano il primo/ultimo nodo figlio
- `parentNode` / `parentElement` ritornano il nodo/elemento genitore
- `nextSibling` / `previousSibling` ritornano il nodo precedente/successivo
- `textContent` ritorna (o imposta) il contenuto testuale di un nodo
- `nodeName` ritorna il nome del tag (se un elemento), oppure `"#text"`, `"#document"` o `"#comment"`, ecc.
- `nodeType` ritorna il tipo di nodo, ad es. `ELEMENT_NODE`, `TEXT_NODE`, `COMMENT_NODE`, ecc.

# Oggetti di tipo Node

## Metodi:

- `appendChild()` / `removeChild()` / `replaceChild()`
- `cloneNode()` clona un nodo
  - ricordarsi di modificare l'eventuale ID del clone!
  - `deep=true` permette di copiare il nodo "in profondità"
- `hasChildNodes()`
- `insertBefore()`

# Oggetti di tipo HTMLElement

- Gli oggetti di tipo `HTMLElement` permettono di accedere e modificare lo stato (ad es., lo stile o il contenuto) di qualunque elemento del documento
- Riguardo lo stile:
  - La proprietà `classList` ritorna la "lista" delle classi dell'elemento (modificabile tramite i metodi `add()`, `remove()` e `toggle()`)
  - La proprietà `style` ritorna un oggetto da cui si può accedere a tutte le proprietà CSS dell'elemento

# Oggetti di tipo HTMLElement

- Riguardo il contenuto:
  - `innerText`: proprietà contenente una stringa con tutto il testo contenuto nell'elemento
  - `innerHTML`: proprietà (ereditata da `Element`) contenente una stringa con tutto il codice HTML *contenuto* nell'elemento
- È possibile anche simulare eventi sugli elementi (es. metodo `click()`)



# Oggetto document

- L'oggetto **document** rappresenta il documento HTML che costituisce la pagina visualizzata
- È un *nodo* (ovvero istanza di Node)
- È il nodo radice: contiene tutti gli altri nodi del documento

# Oggetto document

- Alcune proprietà:
  - `body/head` ritornano i nodi corrispondenti a `body` e `head` del documento, rispettivamente
  - `forms`, `images`, `links` oggetti contenenti tutti i moduli/immagini/collegamenti presenti nella pagina
  - `cookie` (sia `getter` che `setter`)
  - `title` titolo del documento
  - `URL` indirizzo del documento

# Oggetto document

Qualche esempio:

- Modificare il titolo del documento corrente

```
document.title = "Questo è il nuovo  
titolo";
```

- Impostare un cookie

```
document.cookie = "username=Mario  
Rossi; expires=Thu, 05 Apr 2024  
23:00:00 UTC";
```

# Oggetto document

Principali metodi per selezionare elementi:

- `document.getElementById("titolo");`  
Restituisce l'elemento (se esiste) il cui attributo ID vale "titolo"
- `document.getElementsByClassName("c1");`  
Restituisce (in un oggetto `NodeList`) tutti gli elementi il cui attributo CLASS vale "c1"
- `document.getElementsByName("pluto");`  
Restituisce (in un oggetto `NodeList`) tutti gli elementi il cui attributo NAME vale "pluto"
- `document.getElementsByTagName("div");`  
Restituisce (in un oggetto `NodeList`) tutti gli elementi div

# Oggetto document

Principali metodi per selezionare elementi (segue):

- `document.querySelectorAll ( "div.c1" );`  
Restituisce (in un oggetto `NodeList`) tutti gli elementi selezionati dal selettore CSS `div.c1` (e cioè tutti gli elementi `div` di classe `c1`)
- `document.querySelector ( "div.c1" );`  
Restituisce il primo elemento selezionato dal selettore CSS `div.c1`

Altri metodi:

- `createElement (name) / createTextNode (text)`

# Oggetto document

- Le proprietà `forms` / `images` / `links` ritornano oggetti iterabili
- Esempi:

```
for(var i = 0; i < document.forms.length; i++) {  
    var f = document.forms[i];  
    ...  
}  
for(var img of document.images) {  
    // fai qualcosa con img  
    ...  
}
```

# Oggetto document

- Supponendo che nel documento HTML sia definito un modulo di nome *myForm*

```
<FORM ID="myForm"...>
```

```
...
```

```
</FORM>
```

- Si può accedere a tale oggetto in diversi modi:

```
document.forms[0]; (sconsigliato!)
```

```
document.forms["myForm"];
```

```
document.forms.myForm;
```

- Specificando NAME="*myForm*" funzionerà anche  

```
document.myForm;
```

# Oggetto Form

- Un oggetto di questo tipo corrisponde ad un modulo all'interno di una pagina HTML
- Tramite le proprietà di questo oggetto è possibile accedere ai diversi elementi (o controlli) del modulo (inputbox, listbox, checkbox, ecc.)



# Oggetto Form

- Proprietà
  - `action` valore dell'attributo ACTION
  - `elements` vettore contenente gli elementi del modulo
  - `length` numero di elementi del modulo
  - `method` valore dell'attributo METHOD
  - `target` valore dell'attributo TARGET

# Oggetto Form

- Metodi:
  - `reset()` azzera il modulo reimpostando i valori di default per i vari elementi
  - `submit()` invia il modulo

# Oggetto Form

- Supponendo che l'i-esimo elemento di un modulo `mod` sia denominato `nome_i` è possibile farvi riferimento in 3 modi diversi

```
document.mod.elements[i-1];
```

```
document.mod.elements["nome_i"];
```

```
document.mod.name_i;
```

# Elementi dei moduli

All'interno di un modulo possono comparire (oltre agli altri elementi HTML già noti) diversi elementi specifici delle form:

- `<input type="...">`
- `<select>`
- `<textarea>`
- `<button>`
- ...

# Elementi dei moduli

- Tutti questi elementi possiedono le seguenti proprietà
  - `name` nome dell'elemento
  - `value` valore corrente dell'elemento
- Gli elementi di tipo `Input` possiedono la proprietà `defaultValue` che contiene il valore predefinito del campo (attributo `VALUE` del tag HTML)

# Elementi dei moduli

- Gli elementi di tipo `Radio` e `Checkbox` possiedono la proprietà `checked` che indica se l'elemento è stato selezionato
- Gli elementi di tipo `Select` possiedono la proprietà `selectedIndex`, che contiene l'indice dell'elemento selezionato nella lista, e la proprietà `options`, che contiene il vettore delle scelte dell'elenco

# Elementi dei moduli

- È possibile modificare i valori contenuti negli elementi dei moduli
- Pertanto è possibile utilizzare questi elementi anche per fornire risultati all'utente
- Se un elemento ha scopi esclusivamente di rappresentazione può essere marcato come READONLY

# Esempio: modulo di iscrizione

- Il modulo deve raccogliere i dati su un utente che vuole sottoscrivere un certo servizio
- Per ogni utente deve richiedere
  - nominativo, età, sesso
  - se desidera ricevere informazioni commerciali
  - il servizio cui desidera iscriversi



# Esempio: modulo di iscrizione

- Il modulo deve suggerire un'età di 18 anni ed il consenso all'invio di informazioni commerciali
- I servizi disponibili sono denominati "Servizio 1", "Servizio 2" e "Servizio 3"
- Il modulo deve suggerire la sottoscrizione al primo servizio

# Esempio: modulo di iscrizione

- Per ogni dato si determina il tipo di elemento da inserire nel modulo
  - nominativo ed età con elementi di tipo Text
  - sesso con un elementi di tipo Radio
  - infoComm con un elemento di tipo Checkbox
  - servizio con un elemento di tipo Select
- Si deve, inoltre, inserire il pulsante di invio del modulo

# Esempio: modulo di iscrizione

- Per gestire la presentazione si adopera una tabella HTML che presenta sulla prima colonna il nome del campo e nella seconda gli elementi corrispondenti

# Esempio: modulo di iscrizione

```
<HTML>
```

```
<BODY>
```

```
<FORM NAME="iscrizione" ACTION="" METHOD="POST">
```

```
<TABLE>
```

```
...
```

# Esempio: modulo di iscrizione

```
...
<!-- Nominativo -->
<TR>
  <TD>Nominativo</TD>
  <TD>
    <INPUT NAME="nominativo" TYPE="text" SIZE="40">
  </TD>
</TR>
...
```

# Esempio: modulo di iscrizione

...

```
<!-- Età -->
```

```
<TR>
```

```
  <TD>Età;</TD>
```

```
  <TD>
```

```
    <INPUT NAME="eta" TYPE="text" SIZE="3" VALUE="18">
```

```
  </TD>
```

```
</TR>
```

...

# Esempio: modulo di iscrizione

```
...
<!-- Sesso -->
<TR>
  <TD>Sesso</TD>
  <TD>
    <INPUT NAME="sesso" TYPE="radio" VALUE="M"> M
    <INPUT NAME="sesso" TYPE="radio" VALUE="F"> F
  </TD>
</TR>
...
```

# Esempio: modulo di iscrizione

```
...  
<!-- Inform. commerciali -->  
<TR>  
  <TD>Inform. commerciali</TD>  
  <TD>  
    <INPUT NAME="infoComm" TYPE="checkbox" CHECKED>  
  </TD>  
</TR>  
...
```



# Esempio: modulo di iscrizione

```
...
<!-- Servizio -->
<TR>
  <TD>Servizio</TD>
  <TD>
    <SELECT NAME="servizio">
      <OPTION VALUE="s1" SELECTED>Servizio 1</OPTION>
      <OPTION VALUE="s2">Servizio 2</OPTION>
      <OPTION VALUE="s3">Servizio 3</OPTION>
    </SELECT>
  </TD>
</TR>
...
```

# Esempio: modulo di iscrizione

...

```
<!-- Invio del modulo -->
```

```
<TR>
```

```
  <TD>
```

```
    <INPUT TYPE="submit" VALUE="Invia">
```

```
  </TD>
```

```
</TR>
```

...

# Esempio: modulo di iscrizione

...

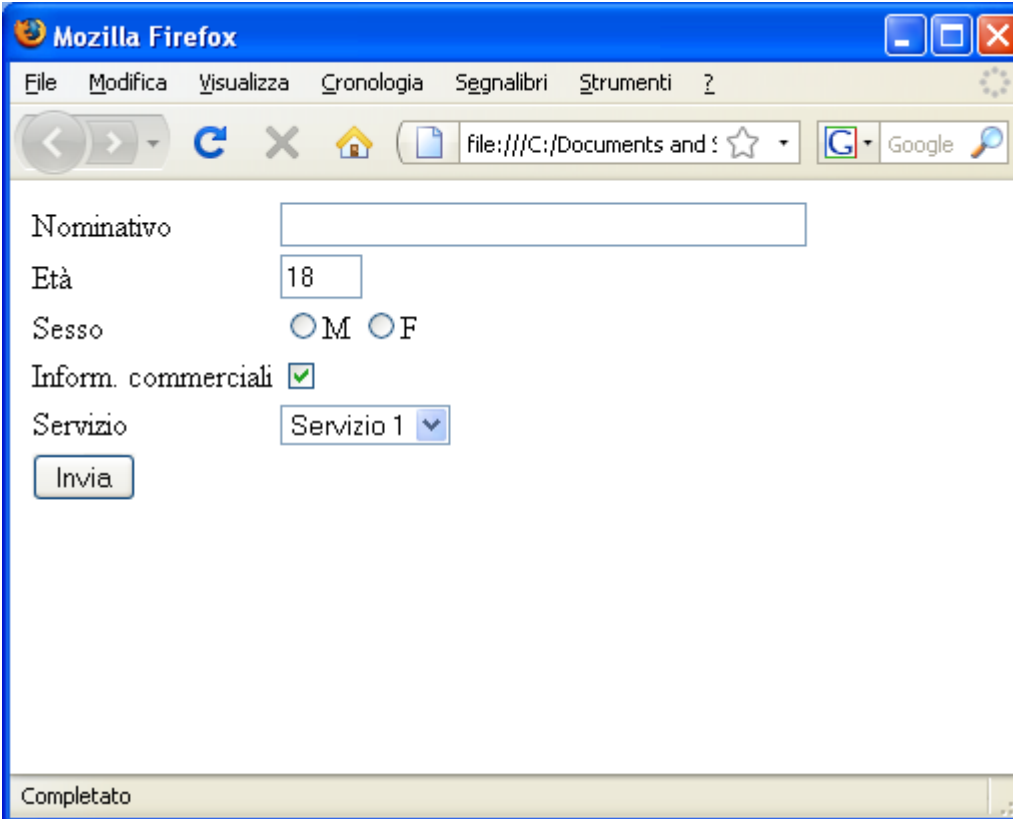
</TABLE>

</FORM>

</BODY>

</HTML>

# Esempio: modulo di iscrizione



The screenshot shows a Mozilla Firefox browser window with a registration form. The browser's address bar displays a file path: `file:///C:/Documents and S...`. The form contains the following fields and controls:

- Nominativo**: A text input field.
- Età**: A text input field containing the value `18`.
- Sesso**: Two radio buttons labeled `M` and `F`.
- Inform. commerciali**: A checked checkbox.
- Servizio**: A dropdown menu currently showing `Servizio 1`.
- Invia**: A button to submit the form.

The status bar at the bottom of the browser window indicates `Completato`.

# Esempio: modulo di iscrizione

- Per accedere al nominativo immesso

```
document.iscrizione.nominativo.value
```

- Per accedere all'età

```
document.iscrizione.eta.value
```

- Per accedere al valore numerico dell'età

```
parseInt(document.iscrizione.eta.value)
```

# Esempio: modulo di iscrizione

- Per visualizzare un messaggio relativo alla scelta di ricevere informazioni commerciali:

```
if (document.iscrizione.infoComm.checked)
    alert("Vuoi ricevere informazioni
        commerciali");
else
    alert("Non vuoi ricevere informazioni
        commerciali");
```

# Eventi

- Ogni oggetto di un documento HTML "genera" degli **eventi** in risposta alle azioni dell'utente
- Ad esempio, l'evento `click` corrisponde al click del puntatore sull'oggetto
- Per gestire l'interazione con l'utente si associano funzioni JavaScript a particolari eventi

# Eventi

- Esempi di eventi:
  - `onSubmit` invio del modulo
  - `onReset` azzeramento del modulo
  - `onClick` click del puntatore
  - `onChange` modifica del contenuto
  - `onFocus` selezione dell'elemento
  - `onKeyDown` pressione di un tasto sulla tastiera
  - `onLoad` caricamento della risorsa



# Intercettazione eventi

- Per intercettare l'evento  $E$  di un tag  $T$  ed associare l'esecuzione di una funzione  $f()$   
`<T onE="return f();">`
- Se il risultato della valutazione della funzione è `false` viene interrotta l'esecuzione del comando corrente, ad esempio l'invio di un modulo

# Intercettazione eventi

- Ad esempio, la funzione `valida()` verifica se i dati immessi nel modulo `modulo` sono corretti ed eventualmente procede con l'invio di quest'ultimo

```
<FORM NAME="modulo"  
  onsubmit="return valida();" ...>
```

...

```
</FORM>
```

# Validazione modulo di iscrizione

- Nel modulo di sottoscrizione del servizio è necessario indicare il nominativo del sottoscrittore
- Quindi il valore del campo corrispondente non deve essere una stringa vuota

# Validazione modulo di iscrizione

- Il modulo viene ridefinito come

```
<FORM NAME="iscrizione" ACTION=""  
  METHOD="POST" onSubmit="return  
  validaIscrizione();" >
```

- Nell'intestazione del documento viene definita una funzione `validaIscrizione()`

# Validazione modulo di iscrizione

```
function validaIscrizione() {  
    if (document.iscrizione.nominativo.value=="")  
    {  
        alert("Nominativo obbligatorio.  
        Impossibile procedere.");  
        return false;  
    }  
    ... //Altri controlli  
    return true;  
}
```

# Proprietà di visualizzazione

- Tra le proprietà degli elementi del modello ad oggetti del documento sono presenti alcune specifiche della modalità di presentazione all'utente degli elementi medesimi
- La possibilità di poter accedere e manipolare tali caratteristiche permette di utilizzare JavaScript per ottenere sofisticati effetti di presentazione visiva
- Purtroppo, queste proprietà sono specifiche dei singoli browser

# Proprietà di visualizzazione

- Le proprietà di visualizzazione sono connesse con l'uso dei CSS
- Infatti, alcune proprietà degli stili possono essere modificate dinamicamente da funzioni JavaScript
- Tra le proprietà più utili per la creazione di effetti visivi abbiamo la visualizzazione ed il posizionamento degli elementi

# BOM: Oggetto window

- Questo oggetto rappresenta la finestra in cui il documento corrente viene visualizzato
- Una funzione può accedere alle proprietà della finestra corrente, ma può creare e manipolare nuove finestre (pop-up)



# Oggetto window

- L'oggetto **window** è in realtà l'oggetto radice di tutto il BOM e DOM
- Infatti i seguenti oggetti (incluso l'oggetto `document`) sono tutti proprietà dell'oggetto `window`:
  - `document`
  - `navigator`
  - `screen`
  - `location`
  - `history`
  - `console`

# Oggetto window

- Altre proprietà dell'oggetto `window`:
  - `title` titolo della finestra
  - `closed` determina se una finestra è aperta o chiusa
  - `localStorage`, `sessionStorage`
  - `frames` contiene l'array dei frame
  - `opener` contiene l'oggetto che ha creato la finestra

# Oggetto window

- Accedere ad un nuovo documento

```
window.location =  
    "http://www.yahoo.com/";
```

- Calcolare l'area (in pixel) del viewport

```
var area = window.innerWidth *  
    window.innerHeight;
```

# Oggetto window

- Metodi
  - `open(location, title)` apre una nuova finestra o tab
  - `close()` chiude la finestra su cui viene invocato
  - `alert(message)` visualizza il messaggio in una finestra di dialogo
  - `confirm(message)` visualizza il messaggio e richiede una conferma all'utente
  - `prompt(message)` visualizza il messaggio e richiede un input testuale
  - `moveTo/By(x, y)` sposta la finestra
  - `resizeTo/By(w, h)` ridimensiona la finestra

# Oggetto window

- Creazione e posizionamento di una nuova finestra

```
var w = window.open("http://www.google.com/", "Google");  
w.moveTo(0, 0);
```

- Visualizzazione di un messaggio

```
window.alert("Attenzione si è  
verificato un errore");
```

# Oggetto window

- Richiesta conferma all'utente

```
if(confirm("Vuoi proseguire con  
l'operazione?")) {  
    //L'utente ha risposto SI  
    ...  
} else {  
    //L'utente ha risposto NO  
    ...  
}
```

# Oggetto navigator

- L'oggetto **navigator** è una proprietà dell'oggetto `window` e rappresenta l'istanza del browser in cui lo script è in esecuzione
- Alcune proprietà
  - `appName` codice identificativo del browser
  - `appName` nome del browser (sconsigliato)
  - `appVersion` numero di versione
  - `geolocation`
  - `language`
  - `userAgent`

# Oggetto history

- Proprietà dell'oggetto `window`
- Rappresenta la sequenza di pagine visitate dall'utente in una scheda
- Metodi
  - `back()` torna alla pagina precedente
  - `forward()` passa alla pagina successiva
  - `length` "lunghezza" della cronologia della scheda