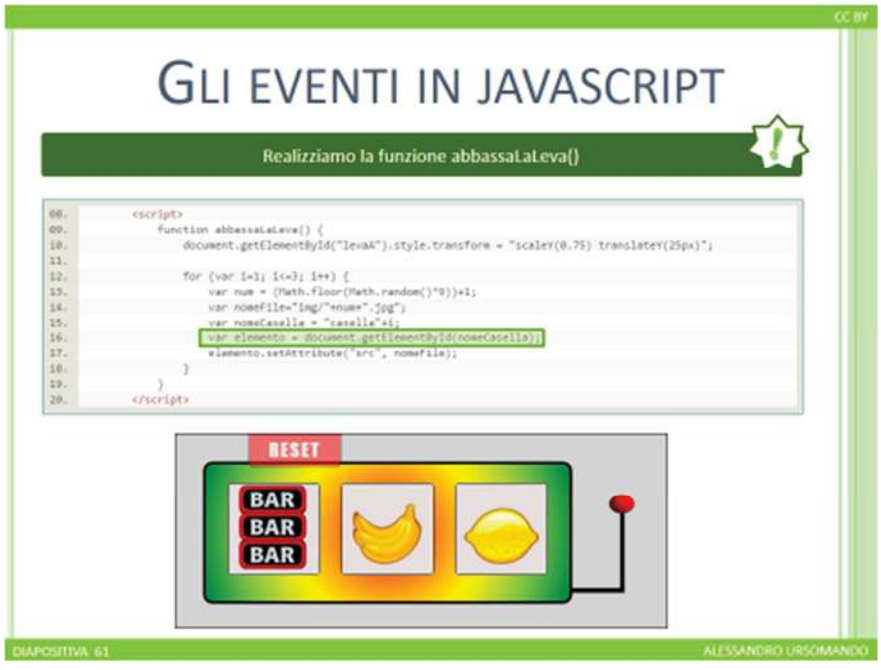
TECNOLOGIE WEB

Manipolazione di elementi di una pagina web con Javascript

INTRODUZIONE

Ricordate l'ultimo esercizio visto?

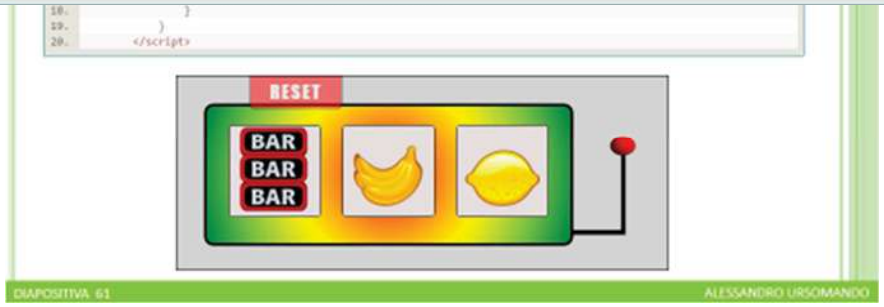


INTRODUZIONE

Ricordate l'ultimo esercizio visto?

Ricordate il codice?

```
08.         <script>
09.             function abbassaLaLeva() {
10.                 document.getElementById("levaA").style.transform = "scaleY(0.75) translateY(25px)";
11.
12.                 for (var i=1; i<=3; i++) {
13.                     var num = (Math.floor(Math.random()*9))+1;
14.                     var nomeFile="img/"+num+".jpg";
15.                     var nomeCasella = "casella"+i;
16.                     var elemento = document.getElementById(nomeCasella);
17.                     elemento.setAttribute("src", nomeFile);
18.                 }
19.             }
20.         </script>
```



CC BY

INTRODUZIONE

Ricordate l'ultimo esercizio visto?

Ricordate il codice?

```
08.         <script>
09.             function abbassaLaLeva() {
10.                 document.getElementById("levaA").style.transform = "scaleY(0.75) translateY(25px)";
11.
12.                 for (var i=1; i<=3; i++) {
13.                     var num = (Math.floor(Math.random()*9))+1;
14.                     var nomeFile="img/"+num+".jpg";
15.                     var nomeCasella = "casella"+i;
16.                     var elemento = document.getElementById(nomeCasella);
17.                     elemento.setAttribute("src", nomeFile);
18.                 }
19.             }
20.         </script>
```

Qui impostavamo l'attributo **src** di un **img** al volo

Bene, qui, in due modi diversi perseguivamo lo stesso obiettivo: un obiettivo molto comune.

Qui impostavamo l'attributo **style** di un **img** al volo

Modificare al volo una proprietà di un elemento

INTRODUZIONE

- Recuperare un **riferimento** all'elemento
- Decidere su quale **attributo** si vuole intervenire
- Che **intervento** fare

Cancellare, modificare, aggiungere..

src, class, style..

Modificare al volo una proprietà di un elemento

ACCEDERE AD UN ELEMENTO

getElementById (x)	Restituisce un riferimento all'elemento con id x

```
<!-- codice qualsiasi -->
<img src="logo.png" alt="img del logo" id="logo" />
<!-- codice qualsiasi -->
```

```
var elemento = document.getElementById("logo");
```

ACCEDERE AD UN ELEMENTO

getElementById (x)	Restituisce un riferimento all'elemento con id x
getElementsByName (x)	Restituisce un vettore di (riferimenti a) elementi di nome x

```
<!-- codice qualsiasi -->
<nav name="barraSuperiore">
    <!-- codice qualsiasi -->
</nav>
<!-- codice qualsiasi -->
```

```
var elemento = document.getElementsByName("barraSuperiore")[0];
```


ACCEDERE AD UN ELEMENTO

getElementById (x)	Restituisce un riferimento all'elemento con id x
getElementsByName (x)	Restituisce un vettore di (riferimenti a) elementi di nome x
getElementsByClassName (x)	Restituisce un vettore di (riferimenti a) elementi di classe x

```
<!-- codice qualsiasi -->
<li class="voceMaschile">Baritono</li>
<li class="voceMaschile">Basso</li>
<li class="voceFemminile">Contralto</li>
<li class="voceFemminile">Mezzosoprano</li>
<li class="voceFemminile">Soprano</li>
<li class="voceMaschile">Tenore</li>
<!-- codice qualsiasi -->
```

```
var elemento = document.getElementsByClassName("voceFemminile")[2];
```

ACCEDERE AD UN ELEMENTO

getElementById (x)	Restituisce un riferimento all'elemento con id x
getElementsByName (x)	Restituisce un vettore di (riferimenti a) elementi di nome x
getElementsByClassName (x)	Restituisce un vettore di (riferimenti a) elementi di classe x
getElementsByTagName (x)	Restituisce un vettore di (riferimenti a) elementi del tag specificato

```
<!-- codice qualsiasi -->
<li class="voceMaschile">Baritono</li>
<li class="voceMaschile">Basso</li>
<li class="voceFemminile">Contralto</li>
<li class="voceFemminile">Mezzosoprano</li>
<li class="voceFemminile">Soprano</li>
<li class="voceMaschile">Tenore</li>
<!-- codice qualsiasi -->
```

```
var elemento = document.getElementsByTagName("li")[2];
```


ACCEDERE AD UN ATTRIBUTO

Esistono due strategie: mediante il metodo apposito o mediante accesso diretto

```
// una volta recuperato il riferimento ad un elemento in qualche modo

// accesso in scrittura
elemento.setAttribute("src", nomeFile);
elemento.src = nomeFile;

// accesso in lettura
alert (elemento.getAttribute("src"));
alert (elemento.src)
```

Ovviamente al posto di **src**, posso usare **alt**, **name**, **style**, **start**..

Attenzione!
Al posto di **class** dovremo usare **className**!

```
elemento.setAttribute("className", "voce voceSelezionata nascosto");
alert(elemento.getAttribute("className")
```

Inoltre, l'attributo **style** merita una trattazione a parte

ACCEDERE ALL'ATTRIBUTO STYLE

Esistono due strategie: mediante il metodo apposito o mediante accesso diretto

```
elemento.setAttribute("style", "visibility: hidden; border: 1px solid black");
```

```
elemento.style.visibility = "hidden";
elemento.style.border = "1px solid black";
```

Vediamo come comportarci con le proprietà css composte da più parole (**font-size**, **letter-spacing**, ecc.)

```
elemento.setAttribute("style","font-size: 45px; letter-spacing: 4px;")
```

```
elemento.style.fontSize = "45px";
elemento.style.letterSpacing = 3px;
elemento.style.fontFamily = "Verdana";
```

È abbastanza chiaro che elemento è un'istanza di una certa classe.

ACCEDERE ALL'ATTRIBUTO PER FARE ALTRE COSE

Quali sono altri metodi e/o attributi interessanti della classe Element?

setAttribute(x,y)	Imposta l'attributo x al valore y
getAttribute(x)	Restituisce il valore dell'attributo x
hasAttribute(x)	Restituisce true se per l'elemento in questione è stato impostato l'attributo x
removeAttribute(x)	Elimina l'attributo x per l'elemento
innerHTML	L'attributo relativo al contenuto dell'elemento

```
<h1 id="titolo">Ciao Mondo</h1>
```

```
document.getElementById("titolo").innerHTML = "Hello World";
```

Esistono poi delle classi che specializzano ulteriormente un elemento

TECNOLOGIE WEB

Gli eventi

GLI EVENTI

La gestione degli eventi ci permette di stabilire cosa fare quando l'utente interagisce con la nostra pagina in un certo modo.

Ma quali sono esattamente gli eventi che possiamo gestire?

Ci sono eventi generali..

..ed eventi relativi ai form

GLI EVENTI PIÙ GENERALI

onclick	Si verifica quando l'utente fa click sull'elemento
ondblclick	Si verifica quando l'utente fa doppio click sull'elemento

onmouseover	Si verifica quando l'utente passa con il mouse sull'elemento
onmouseout	Si verifica quando il cursore lascia l'elemento

GLI EVENTI RELATIVI AI FORM

onchange	Si verifica quando l'utente modifica il valore di un elemento di input (cioè di un form)
onsubmit	Si verifica quando l'utente preme il pulsante di submit (il pulsante principale di un form)

onfocus	Si verifica quando il controllo riceve il focus (tab)
onblur	Si verifica quando il controllo perde il focus

HTML

I form: introduzione

I FORM
DAL PUNTO DI VISTA DELL'UTENTE

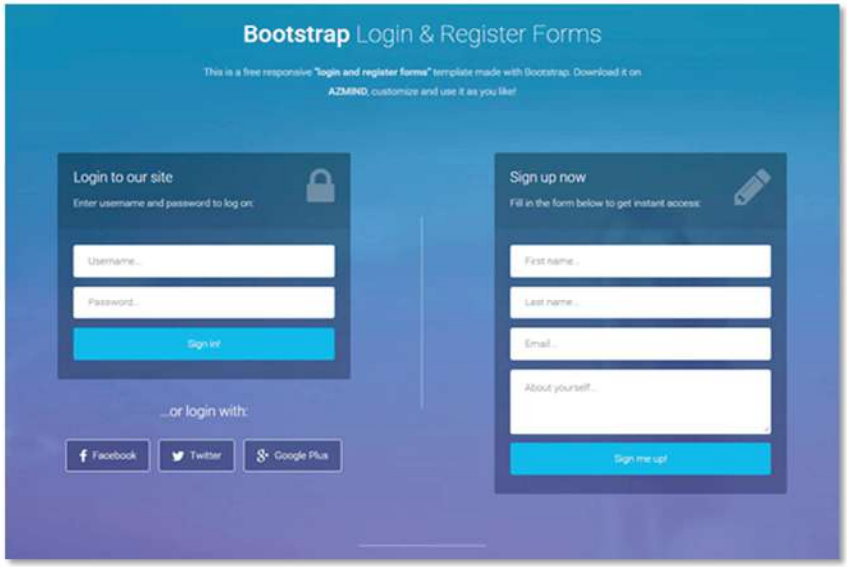
Un form (modulo) permette all'utente di interagire con la pagina WEB.

L'utente inserisce uno o più input (e spesso conferma premendo un tasto)

ALESSANDRO URSOMANDO

I FORM
DAL NOSTRO PUNTO DI VISTA

Una pagina WEB può avere **uno o più** elementi form.



Gli elementi form **non sono mai** annidati l'uno nell'altro.

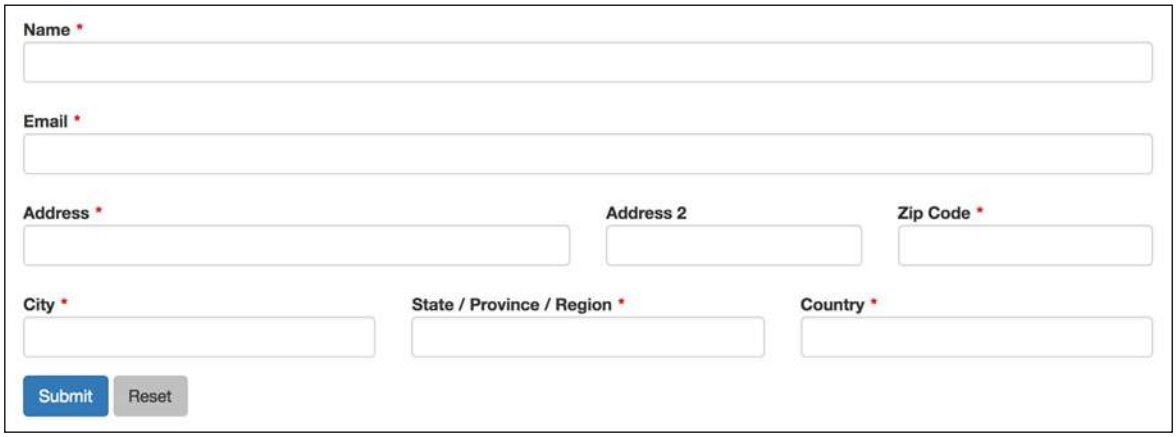
ALESSANDRO URSOMANDO

HTML5 → FORM
CC BY

I FORM
DAL NOSTRO PUNTO DI VISTA

Ogni form può contenere svariati elementi di interazione (**controlli**) e tra questi (di solito) c'è almeno **un pulsante** di conferma (**submit**).

Se c'è, l'altro pulsante serve per pulire il form (**reset**)



ALESSANDRO URSOMANDO

I FORM
DAL NOSTRO PUNTO DI VISTA

Cliccando sul pulsante **submit** si convalida il form

JavaScript

Questo genera l'evento **onsubmit** sul client

E avvia l'esecuzione di uno **script** sul server

Name *

Email *

Address * | Address 2 | Zip Code *

City * | State / Province / Region * | Country *

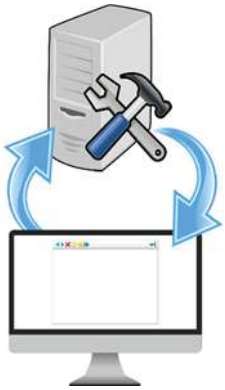
Submit Reset

I FORM
LATO CLIENT

Cliccando sul pulsante **submit** si convalida il form

JavaScript

Questo genera l'evento **onsubmit** sul client

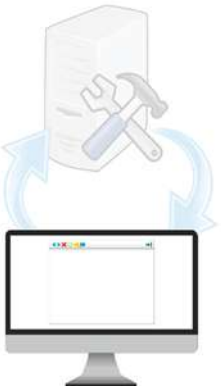


I FORM LATO CLIENT

Cliccando sul pulsante **submit** si convalida il form

Questo genera l'evento **onsubmit** sul client

Script lato client possono anche essere attivati dagli eventi **onchange**, **onfocus**, **onblur**.



I FORM LATO CLIENT

GLI EVENTI RELATIVI AI FORM

onchange	Si verifica quando l'utente modifica il valore di un elemento di input (cioè di un form)
onsubmit	Si verifica quando l'utente preme il pulsante di submit (il pulsante principale di un form)
onfocus	Si verifica quando il controllo riceve il focus (tab)
onblur	Si verifica quando il controllo perde il focus

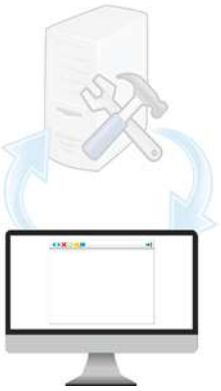
I FORM LATO CLIENT

Cliccando sul pulsante **submit** si convalida il form

Questo genera l'evento **onsubmit** sul client

Script lato client possono anche essere attivati dagli eventi **onchange**, **onfocus**, **onblur**.

Uno script fa riferimento ai controlli utilizzando gli strumenti visti poco fa.



I FORM LATO CLIENT

ACCEDERE AD UN ELEMENTO

getElementById (x)	Restituisce un riferimento all'elemento con id x
getElementsByName (x)	Restituisce un riferimento all'elemento di nome x
getElementsByClassName (x)	Restituisce un vettore di (riferimenti a) elementi di classe x
getElementsByTagName (x)	Restituisce un vettore di (riferimenti a) elementi del tag specificato

```
<!-- codice qualsiasi -->
<li class="voceMaschile">Baritono</li>
<li class="voceMaschile">Basso</li>
<li class="voceFemminile">Contralto</li>
<li class="voceFemminile">Mezzosoprano</li>
<li class="voceFemminile">Soprano</li>
<li class="voceMaschile">Tenore</li>
<!-- codice qualsiasi -->
```

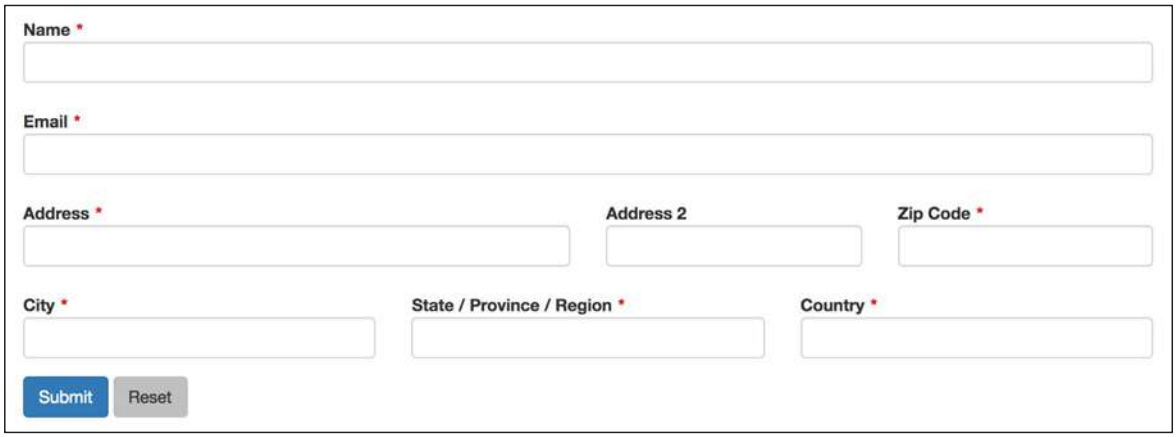
```
var elemento = document.getElementsByTagName("li")[2];
```

I FORM
DAL NOSTRO PUNTO DI VISTA

Cliccando sul pulsante **submit** si convalida il form

Questo genera l'evento **onsubmit** sul client

E avvia l'esecuzione di uno **script** sul server



I FORM
LATO SERVER

Cliccando sul pulsante **submit** si convalida il form

E avvia l'esecuzione di uno **script** sul server

I FORM LATO SERVER

Cliccando sul pulsante **submit** si convalida il form

L'elemento form dispone di due attributi speciali: **action** e **method**.

L'attributo **action** contiene il nome dello script da avviare. L'attributo **method** contiene il nome del vettore dove verranno resi disponibili i valori dei vari elementi del form (**post** o **get**)

Ogni controllo dispone dei valori **name** e **value**: nello script lato server saranno resi disponibili **name** e **value** di ogni controllo del form.

E avvia l'esecuzione di uno **script** sul server

I FORM LATO SERVER

Cliccando sul pulsante **submit** si convalida il form

L'elemento form dispone di due attributi speciali: **action** e **method**.

L'attributo **method** contiene il nome del vettore dove verranno resi disponibili i valori dei vari elementi del form (**post** o **get**)

Ogni controllo dispone dei valori **name** e **value**: nello script lato server saranno resi disponibili **name** e **value** di ogni controllo del form.

E avvia l'esecuzione di uno **script** sul server

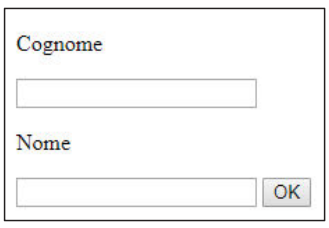
Lasciamo che questa dinamica, per ora resti un po' oscura, tra pochissimo sarà tutto molto chiaro

HTML

I form: gli elementi

TEXT E PULSANTI

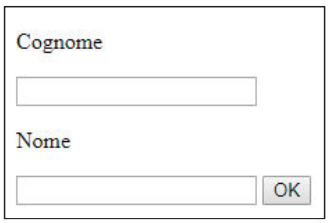
Cominciamo con una cosa semplice.




```
01. <form>
02.     <p>Cognome</p>
03.     <input type="text" />
04.     <p>Nome</p>
05.     <input type="text" />
06.     <input type="submit" value="OK"/>
07. <form>
```

TEXT E PULSANTI DA JAVASCRIPT

Adesso supponiamo di voler leggere i valori inseriti dall'utente da javascript



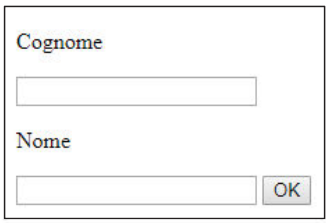

```
<form>
    <p>Cognome</p>
    <input type="text" />
    <p>Nome</p>
    <input type="text" />
    <input type="submit" value="OK"/>
<form>
```

La prima cosa che dobbiamo sapere è che
troveremo questi dati nell'attributo **value** dei controlli.

```
var cognome = document.getElementsByTagName("input")[0].value;
var nome = document.getElementsByTagName("input")[1].value;
alert( "Ciao " + nome + " " + cognome);
```


TEXT E PULSANTI DA JAVASCRIPT

Adesso supponiamo di voler leggere i valori inseriti dall'utente da javascript



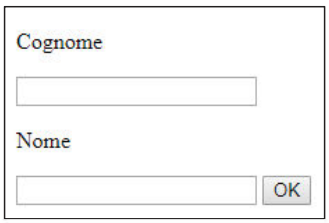

```
<form>
    <p>Cognome</p>
    <input type="text" id="cognome"/>
    <p>Nome</p>
    <input type="text" id="nome"/>
    <input type="submit" value="OK" />
<form>
```

La seconda cosa è che è abbastanza brutto riferirsi ai controlli con il nome del **tag**.
Sarebbe meglio farlo usando l'**id** o il **name**.
Poiché con php useremo il **name** vorremmo usare **name** anche qui ma
la verità è che l'**id** è l'unico valore che – per definizione – è unico nella pagina.
Quindi scegliamo l'**id**.

```
var cognome = document.getElementById("cognome").value;
var nome = document.getElementById("nome").value;
alert( "Ciao " + nome + " " + cognome);
```

TEXT E PULSANTI DA JAVASCRIPT

Adesso supponiamo di voler leggere i valori inseriti dall'utente da javascript



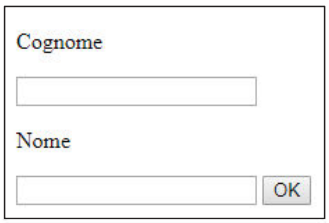

```
<form onsubmit="func()">
    <p>Cognome</p>
    <input type="text" id="cognome"/>
    <p>Nome</p>
    <input type="text" id="nome"/>
    <input type="submit" value="OK" />
<form>
```

Prima di vedere la terza cosa
dobbiamo collegare il pezzettino di codice qui sotto al pezzettino di codice in alto.
Ovvero dobbiamo invocare l'evento **onsubmit()**.

```
function func() {
    var cognome = document.getElementById("cognome").value;
    var nome = document.getElementById("nome").value;
    alert( "Ciao " + nome + " " + cognome);
}
```


TEXT E PULSANTI DA JAVASCRIPT

Adesso supponiamo di voler leggere i valori inseriti dall'utente da javascript



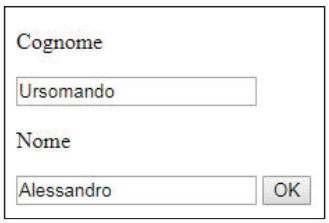

```
<form>
    <p>Cognome</p>
    <input type="text" id="cognome"/>
    <p>Nome</p>
    <input type="text" id="nome"/>
    <input type="button" value="OK" onclick="func()"/>
<form>
```

Ora possiamo vedere la terza cosa.
Dobbiamo sapere che alla pressione del tasto **submit** si avvia un'altra pagina.
Se il **form** ha gli attributi **action** e **method** si passa la palla ad una pagina **php**
altrimenti si ricarica la pagina corrente

Se volessimo evitare questa cosa dovremmo camuffare il pulsante **submit** con un pulsante normale, ovvero con un elemento **input** di tipo **button**.
E – ovviamente – cambiare evento per invocare **func**()

L'ATTRIBUTO VALUE

Adesso supponiamo di voler proporre dei valori preimpostati




```
01. <form>
02.     <p>Cognome</p>
03.     <input type="text" id="cognome" value="Ursomando"/>
04.     <p>Nome</p>
05.     <input type="text" id="nome" value="Alessandro" />
06.     <input type="button" value="OK" onclick="func()"/>
07. <form>
```

Basta dare un valore all'attributo **value** dei due controlli

L'ATTRIBUTO REQUIRED

Adesso supponiamo di voler obbligare l'utente ad inserire un valore


```
01. <form onsubmit="func()">
02.     <p>Cognome</p>
03.     <input type="text" id="cognome"/>
04.     <p>Nome</p>
05.     <input type="text" id="nome" required="required"/>
06.     <input type="submit" value="OK"/>
07. <form>
```

In questo caso aggiungiamo l'attributo **required**

Due cose: la prima.
Le specifiche HTML5 non prevedono che questo attributo venga valorizzato
ma noi – per seguire le regole di XML – dobbiamo dargli un valore.
Decidiamo di seguire la convenzione secondo la quale il valore è il nome dell'attributo.

La seconda. L'attributo **required** funziona solo quando si clicca il pulsante **submit**

L'ATTRIBUTO PLACEHOLDER

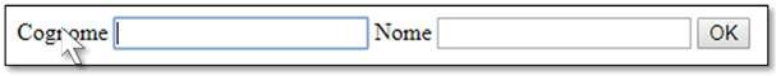
Adesso supponiamo di voler far comparire un'indicazione


```
01. <form>
02.     <p>Cognome</p>
03.     <input type="text"
04.            id="cognome"
05.            placeholder="il tuo cognome"/>
06.     <p>Nome</p>
07.     <input type="text"
08.            id="nome"
09.            placeholder="il tuo nome" />
10.     <input type="button" value="OK" onclick="func()"/>
11. <form>
```

Basta dare un valore all'attributo **placeholder** dei due controlli

ETICHETTE INVECE DI PARAGRAFI

Vogliamo che
cliccare sul testo relativo ad un certo controllo sia come cliccare sul controllo stesso.



L'elemento per ottenere questo effetto si chiama **label**:
ci sono due modi in cui possiamo usarlo.

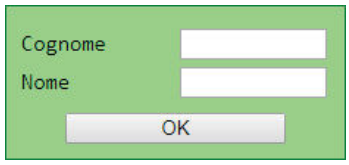
```
01. <form>
02.     <label>
03.         Cognome
04.         <input type="text" id="cognome"/>
05.     </label>
06.     <label for="nome">Nome</label>
07.     <input type="text" id="nome" required="required"/>
08.     <input type="button" value="OK" onclick="func()"/>
09. <form>
```

Tutto molto bello.. però i controlli tutti in fila sono inguardabili!

UN PO' D'ORDINE

Per ottenere un aspetto grafico gradevole
Come si imposta un **form** in html? e quali **specificità** ci sono in ambito **css**?

```css
form {
    width: 210px;
    background: lightgreen;
    padding: 10px;
    border: 1px solid green;
    font-family: Monospace;
    font-size: 15px;
}
div {
    margin: 5px 0px;
}
label {
    width: 100px;
    display: inline-block;
}
input {
    width: 100px;
}
input[type=button] {
    width: 150px;
    display: block;
    margin: 10px auto 0 auto;
}
```



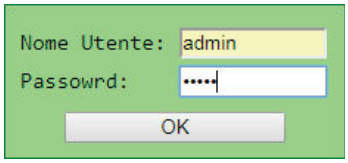

```html
<form>
    <div>
        <label for="cognome">Cognome</label>
        <input type="text" id="cognome"/>
    </div>
    <div>
        <label for="nome">Nome</label>
        <input type="text" id="nome"/>
    </div>
    <input type="button" value="OK"/>
<form>
```


ALESSANDRO URSOMANDO

PASSWORD

Vediamo adesso il controllo per inserire una password.

```html
<form>
    <div>
        <label for="username">Nome Utente:</label>
        <input type="text" id="username"/>
    </div>
    <div>
        <label for="passwd">Passowrd:</label>
        <input type="password" id="passwd"/>
    </div>
    <input type="submit" value="OK"/>
<form>
```



Attenzione!
Se questo form inviasse i dati raccolti ad un server
questi viaggerebbero in chiaro a meno che non si usi il protocollo **https**.

ALESSANDRO URSOMANDO

RESET

È dall'inizio che parliamo del pulsante di reset

```
01. <form>
02.     <div>
03.         <label for="username">Nome Utente:</label>
04.         <input type="text" id="username"/>
05.     </div>
06.     <div>
07.         <label for="passwd">Passowrd:</label>
08.         <input type="password" id="passwd"/>
09.     </div>
10.     <input type="submit" value="OK"/>
11.     <input type="reset" value="SVUOTA FORM" />
12. <form>
```

Nome Utente:
Passowrd:
OK
SVUOTA FORM

Attenzione!
Se questo form inviasse i dati raccolti ad un server
questi viaggerebbero in chiaro a meno che non si usi il protocollo **https**.

RADIO BUTTON

- Nero
- Rosso
- Verde
- Blu
- Giallo

OK

I **radio button** sono elementi **input** con **type** = **radio**.

Un radio button senza un **testo** accanto
non ha senso

Tipicamente sistemiamo l'elemento per il **radio button** dentro un tag **label**

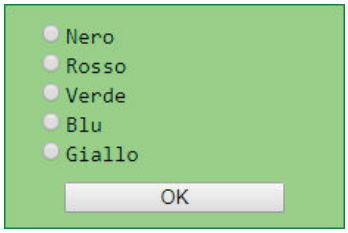
```
01. <form>
02.     <label><input type="radio"/>Nero</label>
03.     <label><input type="radio"/>Rosso</label>
04.     <label><input type="radio"/>Verde</label>
05.     <label><input type="radio"/>Blu</label>
06.     <label><input type="radio"/>Giallo</label>
07.     <input type="submit" value="OK"/>
08. <form>
```

- (•) Nero
- (•) Rosso
- ( ) Verde
- (•) Blu
- ( ) Giallo

OK

Le scelte non sono in mutua esclusione

RADIO BUTTON



I **radio button** sono elementi **input** con **type** = **radio**.

Per avere la mutua esclusione
diamo lo stesso **name** a tutti

```
01. <form>
02.     <label><input type="radio" name="colore" />Nero</label>
03.     <label><input type="radio" name="colore" />Rosso</label>
04.     <label><input type="radio" name="colore" />Verde</label>
05.     <label><input type="radio" name="colore" />Blu</label>
06.     <label><input type="radio" name="colore" />Giallo</label>
07.     <input type="submit" value="OK" />
08. <form>
```

Vediamo adesso come pre-impostare una voce di default

RADIO BUTTON

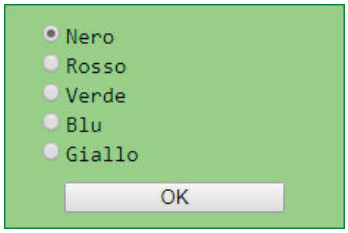
I **radio button** sono elementi **input** con **type** = **radio**.

Per avere la mutua esclusione
diamo lo stesso **name** a tutti

```
01. <form>
02.     <label><input type="radio" name="colore" checked="checked"/>Nero</label>
03.     <label><input type="radio" name="colore" />Rosso</label>
04.     <label><input type="radio" name="colore" />Verde</label>
05.     <label><input type="radio" name="colore" />Blu</label>
06.     <label><input type="radio" name="colore" />Giallo</label>
07.     <input type="submit" value="OK" />
08. <form>
```

Vediamo adesso come pre-impostare una voce di default

RADIO BUTTON



I **radio button** sono elementi **input** con **type** = **radio**.

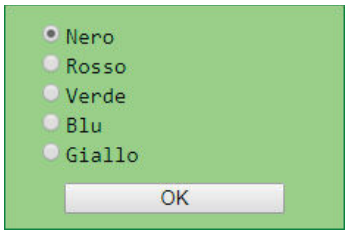
Per avere la mutua esclusione diamo lo stesso **name** a tutti

```
<form>
    <label><input type="radio" name="colore" checked="checked"/>Nero</label>
    <label><input type="radio" name="colore" />Rosso</label>
    <label><input type="radio" name="colore" />Verde</label>
    <label><input type="radio" name="colore" />Blu</label>
    <label><input type="radio" name="colore" />Giallo</label>
    <input type="submit" value="OK" />
<form>
```

Come faremo da **javascript** a sapere quale **voce** è stata **scelta**?

Notiamo che l'**etichetta** non è immediatamente riconducibile all'**elemento input**: bisognerà differenziarli con l'attributo **value**

RADIO BUTTON

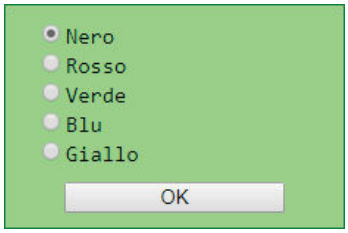


I **radio button** sono elementi **input** con **type** = **radio**.

I radio button di uno stesso gruppo condividono il valore dell'attributo **name** e si differenziano grazie a **value**.

```
<form>
    <label><input type="radio" name="colore" value="#000000" checked="checked"/>Nero</label>
    <label><input type="radio" name="colore" value="#FF0000"/>Rosso</label>
    <label><input type="radio" name="colore" value="#00FF00"/>Verde</label>
    <label><input type="radio" name="colore" value="#0000FF"/>Blu</label>
    <label><input type="radio" name="colore" value="#FFFF00"/>Giallo</label>
    <input type="submit" value="OK" />
<form>
```

RADIO BUTTON DA JAVASCRIPT



Arrivati a questo punto,
vediamo come si accede alla **voce scelta**: ci sono due possibilità

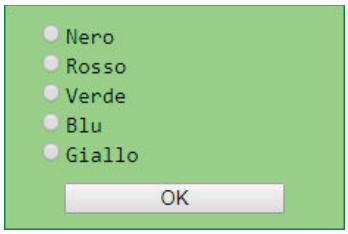
Agganciamo codice javascript al nostro form

```
<form>
    <label><input type="radio" name="colore" value="#000000" checked="checked"/>Nero</label>
    <label><input type="radio" name="colore" value="#FF0000"/>Rosso</label>
    <label><input type="radio" name="colore" value="#00FF00"/>Verde</label>
    <label><input type="radio" name="colore" value="#0000FF"/>Blu</label>
    <label><input type="radio" name="colore" value="#FFFF00"/>Giallo</label>
    <input type="submit" value="OK" onclick="myFunction()"/>
<form>
```

Scorriamo tutti i tag di nome **colore**

```
function myFunction() {
    var elencoRadioButton = document.getElementsByName("colore");
    for (var i=0; i<elencoRadioButton.length; i++) {
        if (elencoRadioButton[i].checked) {
            alert(elencoRadioButton[i].value);
}    }    }
```


RADIO BUTTON DA JAVASCRIPT



Arrivati a questo punto,
vediamo come si accede alla **voce scelta**: ci sono due possibilità

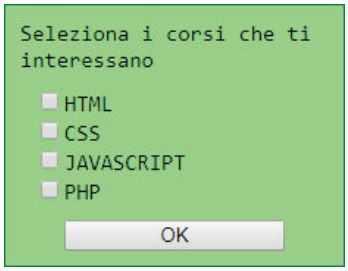
Agganciamo codice javascript al nostro form

```
<form id="formColore">
    <label><input type="radio" name="colore" value="#000000" checked="checked"/>Nero</label>
    <label><input type="radio" name="colore" value="#FF0000"/>Rosso</label>
    <label><input type="radio" name="colore" value="#00FF00"/>Verde</label>
    <label><input type="radio" name="colore" value="#0000FF"/>Blu</label>
    <label><input type="radio" name="colore" value="#FFFF00"/>Giallo</label>
    <input type="submit" value="OK" onclick="myFunction()"/>
<form>
```

Accediamo direttamente dal form

```
function myFunction() {
    alert(document.getElementById("formColore").elements["colore"].value);
}
```

CHECK BOX



Sono elementi di input con **type=checkbox**.

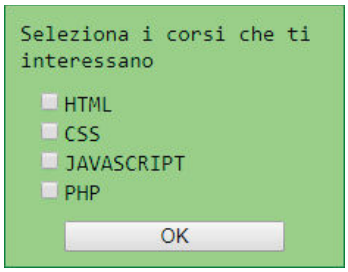
Anche loro sono caratterizzati da un testo
e anche stavolta
siamo soliti inserire il testo e il controllo in un elemento **label**

```
01. <form>
02.     <p>Seleziona i corsi che ti interessano</p>
03.     <label><input type="checkbox" />HTML</label>
04.     <label><input type="checkbox" />CSS</label>
05.     <label><input type="checkbox" />JAVASCRIPT</label>
06.     <label><input type="checkbox" />PHP</label>
07.     <input type="submit" value="OK"/>
08. </form>
```

Stavolta non abbiamo il problema della mutua esclusione

Anche qui è valido l'attributo **checked**

CHECK BOX DA JAVASCRIPT



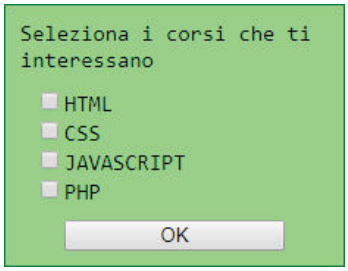
Per analizzare ciascuna voce possiamo decidere
se vogliamo farlo tramite **id** o tramite **name**

Una volta avuto un riferimento a una voce
valutiamo il suo attributo **checked**
e – se c'è – possiamo usare il suo attributo **value**

```
01. <form onsubmit="myFunction()">
02.     <p>Seleziona i corsi che ti interessano</p>
03.     <label><input type="checkbox" id="corsoHtml" />HTML</label>
04.     <label><input type="checkbox" id="corsoCss"  />CSS</label>
05.     <label><input type="checkbox" id="corsoJavascript" />JAVASCRIPT</label>
06.     <label><input type="checkbox" id="corsoPhp"  />PHP</label>
07.     <input type="submit" value="OK"/>
08. </form>
```

```
01. function myFunction() {
02.     if (document.getElementById("corsoHtml").checked) {
03.         window.open("http://www.bbuio.it/didattica/html/", "_blank");
04.     }
05.     if (document.getElementById("corsoCss").checked) {
06.         window.open("http://www.bbuio.it/didattica/css/", "_blank");
07.     }
08.     /* ecc. ecc. */
09. }
```

CHECK BOX DA JAVASCRIPT

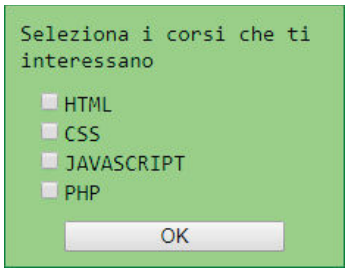


Per analizzare ciascuna voce possiamo decidere se vogliamo farlo tramite **id** o tramite **name**

Una volta avuto un riferimento a una voce valutiamo il suo attributo **checked** e – se c'è – possiamo usare il suo attributo **value**

```
<form  onsubmit="myFunction()">
    <p>Seleziona i corsi che ti interessano</p>
    <label>
        <input type="checkbox" name="corsi" value="http://www.bbuio.it/didattica/html/" />
        HTML
    </label>
    <label>
        <input type="checkbox" name="corsi" value="http://www.bbuio.it/didattica/css/" />
        CSS
    </label>
    <label>
        <input type="checkbox" name="corsi" value="http://www.bbuio.it/didattica/javascript/" />
        JAVASCRIPT
    </label>
    <label>
        <input type="checkbox" name="corsi" value="http://www.bbuio.it/didattica/php/" />
        PHP
    </label>
    <input type="submit" value="OK"/>
</form>
```


CHECK BOX DA JAVASCRIPT

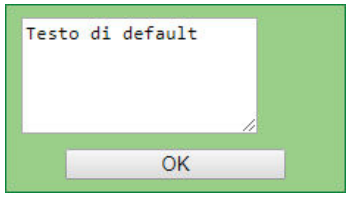



```
function myFunction() {
    var elencoCheckBox = document.getElementsByName("corsi");
    for (var i=0; i<elencoCheckBox.length; i++) {
        if (elencoCheckBox[i].checked) {
            window.open(elencoCheckBox[i].value);
        }
    }
}
```

```
<form  onsubmit="myFunction()">
    <p>Seleziona i corsi che ti interessano</p>
    <label>
        <input type="checkbox" name="corsi" value="http://www.bbuio.it/didattica/html/" />
        HTML
    </label>
    <label>
        <input type="checkbox" name="corsi" value="http://www.bbuio.it/didattica/css/" />
        CSS
    </label>
    <label>
        <input type="checkbox" name="corsi" value="http://www.bbuio.it/didattica/javascript/" />
        JAVASCRIPT
    </label>
    <label>
        <input type="checkbox" name="corsi" value="http://www.bbuio.it/didattica/php/" />
        PHP
    </label>
    <input type="submit" value="OK"/>
</form>
```

AREA DI TESTO

```
01. <form>
02.     <textarea cols="20" rows="5">
03.         Testo di default
04.     </textarea>
05.     <input type="submit" value="OK"/>
06. </form>
```

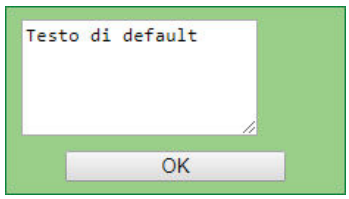


Notiamo che questo non è un elemento **input** con un qualche **type**
ma è un attributo a sé stante
il cui **contenuto** viene posto all'interno del controllo come testo di default.

Gli attributi **cols** e **rows** indicano la dimensione in termini di righe e colonne
ovviamente è sempre possibile (nonché consigliabile) dimensionare il box con **css**.

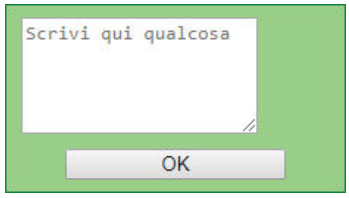
AREA DI TESTO

```
01. <form>
02.     <textarea cols="20" rows="5">
03.         Testo di default
04.     </textarea>
05.     <input type="submit" value="OK"/>
06. </form>
```



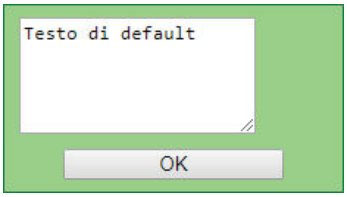
Anche qui è possibile usare **placeholder**

```
01. <form>
02.     <textarea placeholder="Scrivi qui qualcosa">
03.         <!-- niente testo di default -->
04.     </textarea>
05.     <input type="submit" value="OK"/>
06. </form>
```



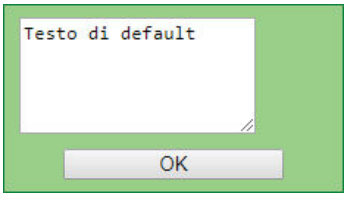
AREA DI TESTO DA JAVASCRIPT

```
<form>
    <textarea cols="20" rows="5">
        Testo di default
    </textarea>
    <input type="submit" value="OK"/>
</form>
```



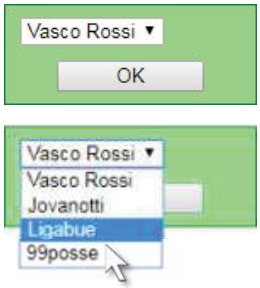
Per accedere al controllo da Javascript
gli forniamo un **id**
e poi accediamo al suo attributo **value**

```
<form onsubmit="myFunction()">
    <textarea id="area">
        Testo di default
    </textarea>
    <input type="submit" value="OK"/>
</form>
```




```
function myFunction() {
    alert(document.getElementById("area").value);
}
```

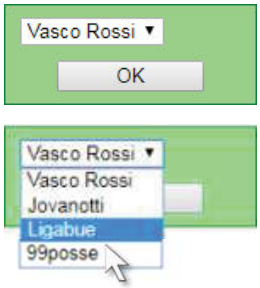

MENU A TENDINA




```
<form>
    <select id="sito">
        <option value="http://www.vascorossi.net">Vasco Rossi</option>
        <option value="http://www.soleluna.it">Jovanotti</option>
        <option value="http://www.ligachannel.com">Ligabue</option>
        <option value="http://www.novenove.it">99posse</option>
    </select>
    <input type="submit" value="OK"/>
</form>
```

L'elemento **select** produce un menu a tendina
con tante voci quanti elementi **option** contiene

MENU A TENDINA DA JAVASCRIPT




```html
<form onsubmit="myFunction()">
    <select id="sito">
        <option value="http://www.vascorossi.net">Vasco Rossi</option>
        <option value="http://www.soleluna.it">Jovanotti</option>
        <option value="http://www.ligachannel.com">Ligabue</option>
        <option value="http://www.novenove.it">99posse</option>
    </select>
    <input type="submit" value="OK"/>
</form>
```

La classe relativa all'elemento **select** ha un paio di attributi interessanti

```javascript
function myFunction() {
    var x = document.getElementById("sito");
    var i = x.selectedIndex;
    alert( x.options[i].value );
}
```

L'attributo **selectedIndex**, per esempio, ci dice qual è l'indice della voce selezionata

E ancora, l'attributo **options** è un vettore che contiene le voci del menu

CHECK BOX

Per inserire un **check box** nel nostro form useremo un elemento di tipo **input** con attributo **type** pari a **checkbox**.

Ogni casella avrà il suo nome (**name**) e il suo valore (**value**).

```
07. <form>
08.     <label><input type="checkbox" name="corso1" value="true" /> HTML </label>
09.     <label><input type="checkbox" name="corso2" value="true" /> CSS </label>
10.     <label><input type="checkbox" name="corso3" value="true" /> JAVASCRIPT </label>
11.     <label><input type="checkbox" name="corso4" value="true" /> PHP </label>
12.     <input class="btn" type="submit" value="Invia Dati" />
13. </form>
```

Allo script php arriverà una variabile **corso1** con valore **true** solo se l'utente avrà selezionato la casella relativa al corso **HTML**

Sarà quindi più facile gestire lo script php se il form sarà stato preparato in questo modo:

ALESSANDRO URSOMANDO

CHECK BOX

Per inserire un **check box** nel nostro form useremo un elemento di tipo **input** con attributo **type** pari a **checkbox**.

Ogni casella avrà il suo nome (**name**) e il suo valore (**value**).

```
07. <form>
08.     <label><input type="checkbox" name="corsi[]" value="html" />HTML</label>
09.     <label><input type="checkbox" name="corsi[]" value="css" />CSS</label>
10.     <label><input type="checkbox" name="corsi[]" value="javascript" />JAVASCRIPT</label>
11.     <label><input type="checkbox" name="corsi[]" value="php" />PHP</label>
12.     <input class="btn" type="submit" value="Invia Dati" />
13. </form>
```

In questo modo, allo script php arriverà un vettore **corsi** con al suo interno il valore delle voci selezionate

ALESSANDRO URSOMANDO

NUOVI TIPI DI INPUT

HTML5 ha rivoluzionato i form introducendo
sia nuovi **attributi** che nuovi **tipi** di input. Tralasciamo gli attributi.

Alcuni si sostituiscono
a un lavoro che in passato andava
fatto con **Javascript**:

- ❑ number
- ❑ range
- ❑ date
- ❑ time
- ❑ color

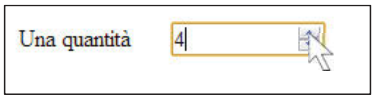
Altri sono propri
dell'ambito **mobile**:

- ❑ search
- ❑ email
- ❑ url

ALESSANDRO URSOMANDO

NUOVI TIPI DI INPUT: NUMBER

L'elemento **input** con type pari a **number**
viene reso come un textedit con i pulsantini per aumentare e diminuire



Qui possono essere utili
gli attributi **min**, **max** e **step.**

```
08. <div>
09.     <label for="number">Una quantità</label>
10.     <input id="number" type="number" name="number" />
11. </div>
```


ALESSANDRO URSOMANDO

NUOVI TIPI DI INPUT: RANGE

L'elemento **input** con type pari a **range**
viene reso con un selettore che scorre su un asse

Un intervallo

Anche qui possono essere utili
gli attributi **min**, **max** e **step.**

```
08. <div>
09.     <label for="range">Un intervallo</label>
10.     <input id="range" type="range" name="range" />
11. </div>
```


NUOVI TIPI DI INPUT: DATE

Una data gg/mm/aaaa

ottobre 2018

lun	mar	mer	gio	ven	sab	dom
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

L'elemento **input** con type pari a **date** disegna sulla pagina un textedit atto a contenere una data; con due pulsantini per aumentare o diminuire la data presente e con un pulsante che apre un intero calendario.

```
08. <div>
09.     <label for="date">Una data</label>
10.     <input id="date" type="date" name="date" />
11. </div>
```

NUOVI TIPI DI INPUT: TIME

L'elemento **input** con type pari a **time**
disegna sulla pagina un textedit atto a contenere un orario;
con due pulsantini per aumentare o diminuire la data presente

Anche qui possono essere utili
gli attributi **min**, **max.**

```
08. <div>
09.     <label for="time">Un orario</label>
10.     <input id="time" type="time" name="time" />
11. </div>
```


ALESSANDRO URSOMANDO

NUOVI TIPI DI INPUT: COLOR

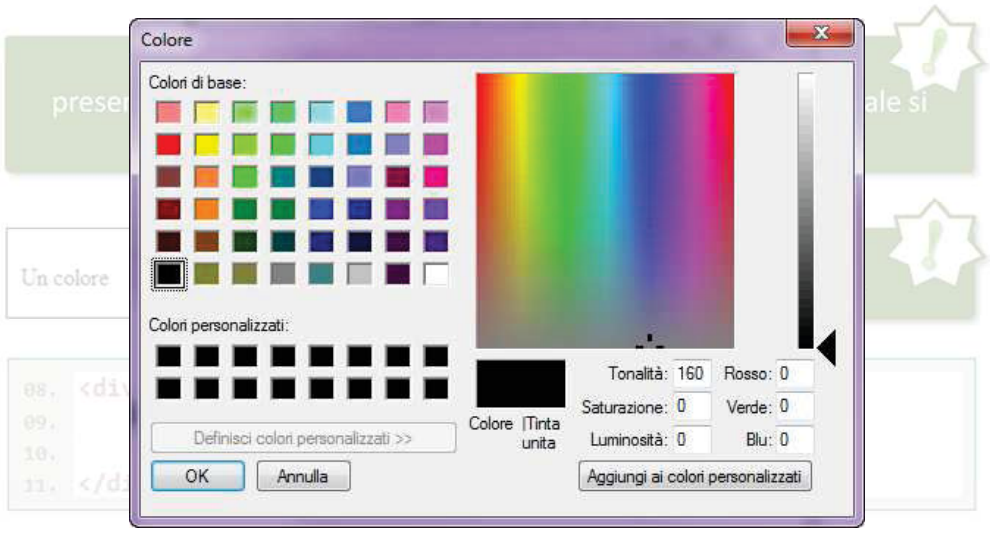
L'elemento **input** con type pari a **color**
presenta sulla pagina un pulsante con un colore, premendo il quale si
presenta la finestra di dialogo per la scelta di un colore.

Dando un valore a **value**
si configura il colore iniziale

```
08. <div>
09.     <label for="color">Un colore</label>
10.     <input id="color" type="color" name="color" />
11. </div>
```


ALESSANDRO URSOMANDO

NUOVI TIPI DI INPUT: COLOR



FIELDSET E LEGEND

Carta di credito

Numero

Data di Scadenza gg/mm/aaaa

Numero CSV

Per raggruppare esteticamente più controlli sotto un'unica etichetta possiamo usare gli elementi **fieldset** e **legend**.

```
08. <fieldset>
09.     <legend>Carta di credito</legend>
10.     <div>
11.         <label for="numero" >Numero</label>
12.         <input id="numero" type="text" name="numero"/>
13.     </div>
14.     <div>
15.         <label for="dataS">Data di Scadenza</label>
16.         <input id="dataS" type="date" name="dataS"/>
17.     </div>
18.     <div>
19.         <label for="csv" >Numero CSV</label>
20.         <input id="csv" type="text" name="csv"/>
21.     </div>
22. </fieldset>
```

HIDDEN

Presentiamo infine il controllo necessario ad aggiungere
ai dati inseriti dall'utente, eventuali dati inseriti dal programmatore.

```
07. <form>
08.     <input
09.         type="hidden"
10.         name="autore"
11.         value="Alessandro Ursomando"
12. form>
```

In questo modo per esempio, inviamo allo script php
una variabile che si chiama **autore** e che vale **«Alessandro Ursomando»**

TECNOLOGIE WEB

Variabili globali in Javascript

VARIABILI GLOBALI

Abbiamo visto come – con il tag **script** – sia possibile introdurre codice Javascript nel nostro documento html dove lo riteniamo più opportuno.

Abbiamo visto come sia possibile – nonché buona prassi – raccogliere tuto il codice javascript in un file esterno e linkarlo dalla sezione **head**

```
01.         <script src="script.js"></script>
```

Anche se non abbiamo ancora una grande esperienza è facile immaginare che tutto il codice javascript sia raccolto in funzioni.

Funzioni da invocare all'occorrenza di un certo evento secondo quanto specificato nel codice html

Bene. Ma come comunicano queste funzioni?
Come facciamo ad usare un valore prodotto al verificarsi di un certo evento in un evento che si verifica successivamente?

VARIABILI GLOBALI

Con le variabili globali.

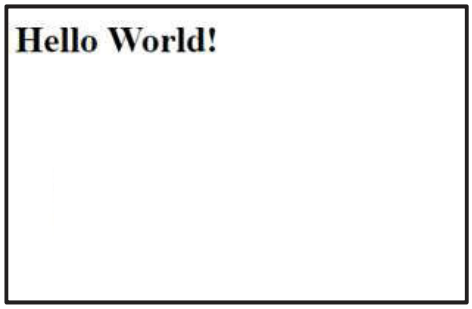
```
01. var varGlobale;
02.
03. function myFunction1() {
04.     // acesso in scrittura alla var globale
05. }
06.
07. function myFunction2() {
08.     // acesso in lettura alla var globale
09. }
10.
11. // ecc. ecc.
12.
```

TECNOLOGIE WEB

Manipolazione del DOM

MANIPOLAZIONE DEL DOM

```
01. <html>
02.     <head>
03.         <title>Saluti</title>
04.         <script src="script.js"></script>
05.     </head>
06.     <body>
07.         <h1 onclick="aggiungiUnImg()">
08.             Hello World!
09.         </h1>
10.     </body>
11. </html>
```



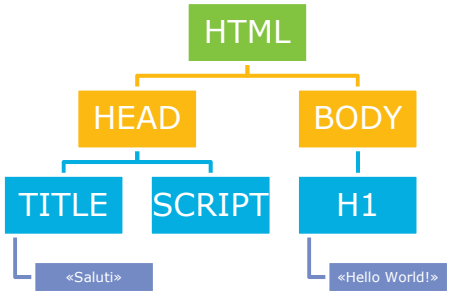
Ecco, stiamo per vedere una cosa che non capita proprio tutti i giorni.

Nello specifico stiamo per vedere come aggiungere elementi al DOM.

Cominciamo con il considerare il DOM della seguente pagina web

MANIPOLAZIONE DEL DOM

```
01. <html>
02.     <head>
03.         <title>Saluti</title>
04.         <script src="script.js"></script>
05.     </head>
06.     <body>
07.         <h1 onclick="aggiungiUnImg()">
08.             Hello World!
09.         </h1>
10.     </body>
11. </html>
```



Ecco, stiamo per vedere una cosa che non capita proprio tutti i giorni.

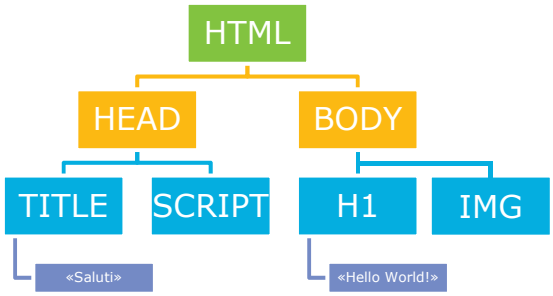
Nello specifico stiamo per vedere come aggiungere elementi al DOM.

Cominciamo con il considerare il DOM della seguente pagina web

Ora aggiungiamo un elemento **img** al **body** ogni volta che si fa click sull'elemento **h1**

MANIPOLAZIONE DEL DOM

```
01. <html>
02.     <head>
03.         <title>Saluti</title>
04.         <script src="script.js"></script>
05.     </head>
06.     <body>
07.         <h1 onclick="aggiungiUnImg()">
08.             Hello World!
09.         </h1>
10.     </body>
11. </html>
```

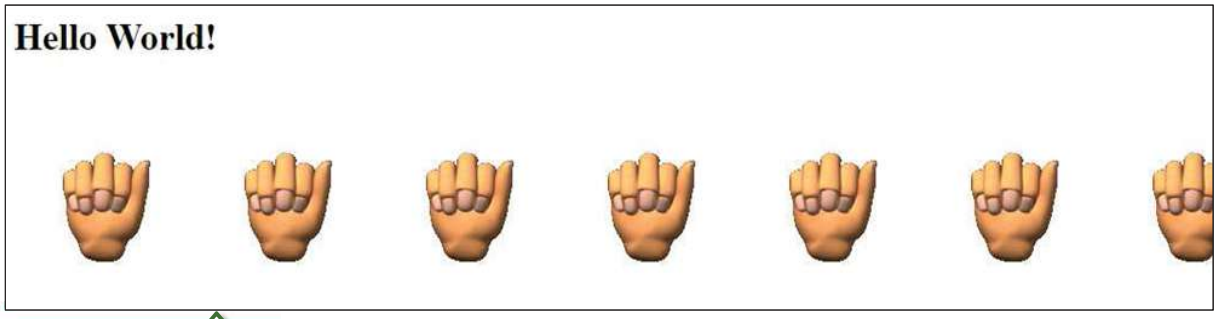


Al primo click sul testo si aggiungerà un tag **img** al **body**.

E poi tanti altri!

```
01. function aggiungiUnImg() {
02.     // creo un nuovo elemento img e ne prendo il riferimento
03.     var nuovoElemento = document.createElement("img");
04.
05.     // creo un nuovo attributo e ne prendo il riferimento
06.     var nuovoAttributo = document.createAttribute("src");
07.
08.     // aggancio questo attributo all'elemento
09.     nuovoElemento.setAttributeNode(nuovoAttributo);
10.
11.     // valorizzo questo nuovo attributo
12.     nuovoElemento.src = "manina.gif"
13.
14.     // aggancio questo nuovo elemento al body
15.     document.getElementsByTagName("body")[0].appendChild(nuovoElemento);
16. }
```

MANIPOLAZIONE DEL DOM



Al primo click sul testo si aggiungerà un tag **img** al **body**.

E poi tanti altri!

```
01. function aggiungiUnImg() {
02.     // creo un nuovo elemento img e ne prendo il riferimento
03.     var nuovoElemento = document.createElement("img");
04.
05.     // creo un nuovo attributo e ne prendo il riferimento
06.     var nuovoAttributo = document.createAttribute("src");
07.
08.     // aggancio questo attributo all'elemento
09.     nuovoElemento.setAttributeNode(nuovoAttributo);
10.
11.     // valorizzo questo nuovo attributo
12.     nuovoElemento.src = "manina.gif"
13.
14.     // aggancio questo nuovo elemento al body
15.     document.getElementsByTagName("body")[0].appendChild(nuovoElemento);
16. }
```


ALESSANDRO URSOMANDO

MANIPOLAZIONE DEL DOM

Allora introduciamo una variabile globale **flag** che poniamo a **true** non appena aggiunta una immagine.

Al primo click sul testo si aggiungerà un tag **img** al **body**.

E poi tanti altri!

```
01. function aggiungiUnImg() {
02.     // creo un nuovo elemento img e ne prendo il riferimento
03.     var nuovoElemento = document.createElement("img");
04.
05.     // creo un nuovo attributo e ne prendo il riferimento
06.     var nuovoAttributo = document.createAttribute("src");
07.
08.     // aggancio questo attributo all'elemento
09.     nuovoElemento.setAttributeNode(nuovoAttributo);
10.
11.     // valorizzo questo nuovo attributo
12.     nuovoElemento.src = "manina.gif"
13.
14.     // aggancio questo nuovo elemento al body
15.     document.getElementsByTagName("body")[0].appendChild(nuovoElemento);
16. }
```


ALESSANDRO URSOMANDO

MANIPOLAZIONE DEL DOM

Allora introduciamo una variabile globale **flag**
che poniamo a **true** non appena aggiunta una immagine.

```
01. var giaAggiunto = false;
02.
03. function aggiungiUnImg() {
04.     if (!giàAggiunto) {
05.         // creo un nuovo elemento img e ne prendo il riferimento
06.         var nuovoElemento = document.createElement("img");
07.
08.         // creo un nuovo attributo e ne prendo il riferimento
09.         var nuovoAttributo = document.createAttribute("src");
10.
11.         // aggancio questo attributo all'elemento
12.         nuovoElemento.setAttributeNode(nuovoAttributo);
13.
14.         // valorizzo questo nuovo attributo
15.         nuovoElemento.src = "manina.gif"
16.
17.         // aggancio questo nuovo elemento al body
18.         document.getElementsByTagName("body")[0].appendChild(nuovoElemento);
19.
20.         // tengo traccia del fatto che ho già aggiunto il tags
21.         giaAggiunto = true;
22.     }
23. }
```