JAVASCRIPT: GLI ARRAY E LE FUNZIONI

GLI ARRAY

INTRODUZIONE AGLI ARRAY.
Nella lezione precedente è stato introdotto il concetto di variabile e le modalità di utilizzo all'interno degli script. A tal proposito sono stati introdotti e analizzati codici che operavano su un limitato numero di dati.
Spesso però si ha esigenza di istanziare e gestire un numero maggiore di variabili, ad esempio per mantener traccia di più valori numerici o stringhe.
Un array, detto anche vettore, è una lista di variabili. Possiamo immaginare un array come una **sequenza di celle**, dove ogni cella contiene un valore.

Grazie ad un array è possibile utilizzare un numero infinito di variabili senza esser costretti a istanziare e gestire molteplici variabili singole.

GLI ARRAY IN JAVASCRIPT.
Come tutti i linguaggi di programmazione moderni anche Javascript permette di utilizzare gli array. La sintassi per istanziare un nuovo array è:
var nome_array = new Array();
La parola chiave **new** indica all'interprete Javascript l'inizializzazione di un nuovo oggetto il cui tipo è **Array()**.
Una volta definito un array è necessario aggiungere i valori alle singole celle grazie ad un puntatore che indichi di volta in volta la cella da elaborare.
Se ad esempio volessimo definire un valore alla cella i, la sintassi è la seguente:

nome_array[i] = valore;
Volendo tradurre l'istruzione diremo: **imposta valore alla cella i-esima dell'array**. Se immaginiamo di dover scorrere tutti gli *n* elementi di un array, allora l'indice (i) varrà di volta in volta 1,2,3 e così via sino all'elemento n.
Per chiarire l'utilizzo degli array, analizziamo un esempio pratico:

```
<script type='text/javascript'>
        // Istanziamo un array
        var numeri = new Array();
        // Riempiamo l'array con i numeri da 1 a 10
        numeri[0] = 1;
        numeri[1] = 2;
        numeri[2] = 3;
        numeri[3] = 4;
        numeri[4] = 5;
        numeri[5] = 6;
        numeri[6] = 7;
        numeri[7] = 8;
        numeri[8] = 9;
        numeri[9] = 10;
        // Stampiamo a video un valore dell'array
        window.document.write(numeri[4]);
</script>
```

Nell'esempio abbiamo istanziato un nuovo array e quindi avvalorato con i numeri da 1 a 10. Da notare che il puntatore **i** inizia da 0 e termina a 9.
Infine stampiamo a video un valore tra quelli presenti nell'array.

CICLARE TRA GLI ELEMENTI DI UN ARRAY

Un modo efficiente e veloce per manipolare gli array è fornito dai cicli for o while.
Nell'esempio visto precedentemente, il puntatore ad ogni avvaloramento dell'array viene incrementato di 1, passando da 0 a 9. Avremmo potuto utilizzare un ciclo for per aggiungere i valori all'array:

```
<script type='text/javascript'>
        // Istanziamo un array
        var numeri = new Array();
        // Riempiamo l'array con i numeri da 1 a 10
        for(var i=0; i<10; i++){
                numeri[i] = i+1;
        }
</script>
```

Nel for la variabile i funge da indice per l'array. Analizzando ad esempio per i=0 avremo:
numeri[0] = 0+1;
Per i = 1 : numeri[1] = 1+1.
Per i = 2 : numeri[2] = 2+1. E così via.
Il vantaggio si misura in termini di efficienza e pulizia del codice, infatti abbiamo riscritto lo script passando da 10 a sole 3 righe di codice. Un ulteriore vantaggio è nel momento in cui ci troviamo a leggere un array:

LUNGHEZZA DI UN ARRAY.

In fase di inizializzazione di un array in Javascript è possibile specificare il numero massimo di elementi dalle sarà composto:

var nome_array = new Array(lunghezza);

Utilizzando questa sintassi, l'interprete Javascript realizzerà un array con un numero di elementi specificato avvalorati a **null** (valore nullo). Utilizzare questo metodo risulta utile quando si è certi del numero di variabili da istanziare.

Nonostante ciò nella maggior parte dei casi risulterà più utile la sintassi generale non sapendo a priori il numero di elementi che utilizzeremo durante lo sviluppo del codice. Di conseguenza a volte è indispensabile calcolare il numero di elementi presenti in un array.

In aiuto ci viene la seguente funzione:

var lunghezza = nome_array.length;

Nella variabile lunghezza otterremo il numero di elementi presenti.

Riprendendo il codice Javascript che inserisce i numeri da 1 a 10 all'interno di un array, calcoliamo il numero di elementi:

```
<script type='text/javascript'>
        // Istanziamo un array
        var numeri = new Array();
        // Riempiamo l'array con i numeri da 1 a 10
        for(var i=0; i<10; i++){
                numeri[i] = i+1;
        }

        // Stampo a video il numero di elementi
        var lunghezza = numeri.length;
        window.document.write('Il numero di elementi è logicamente: ' + lunghezza);
</script>
```

Per verificare la funzione length si consiglia, per prova, di stampare la lunghezza dell'array ad ogni iterazione del ciclo for di inserimento valori.

ARRAY MISTI

Un array in Javascript può contenere valori di tipo diverso.

In altre parole è possibile utilizzare numeri e stringhe insieme(come mostrato nell'immagine successiva):

```
numeri[0] = 1;

numeri[1] = 'uno';
```

Questo aspetto ci fornisce un grado di flessibilità maggiore non essendo legati ad un tipo di dato per l'array. È possibile raggruppare le diverse informazioni in un singolo array: sarà poi dovere dello sviluppatore controllare i valori e gestire al meglio il flusso dei dati evitando confusioni.

GLI ARRAY INCORPORATI DI JAVASCRIPT

Gli array rivestono un ruolo essenziale in Javascript, difatti quando un browser web legge una pagina html, crea automaticamente vari array per contenere gli elementi che compongono la pagina.

Per capire meglio questo concetto immaginiamo una pagina web nella quale sono presenti, oltre a testo e altri elementi, tre immagini:

```
...
<img src='immagine1.jpg' alt='Immagine 1' />
<img src='immagine2.jpg' alt='Immagine 2' />
<img src='immagine3.jpg' alt='Immagine 3' />
...
```

Nel momento in cui il browser elabora il codice html e crea la pagina, realizza al contempo un array contenente le immagini visualizzate a video.
In ugual maniera realizzerà un array per tutte le form che incontra nonché per i campi di input e altri elementi che compongono la pagina.
In definitiva avremo ad esempio i seguenti array:
Per le immagini : window.document.images[];
Per le forms : window.document.forms[];
Per i links : window.document.anchors[];
e così via per molti elementi della pagina.
Questo modo di mantenere la struttura della pagina, permette a Javascript di **accedere** in maniera piuttosto veloce ai singoli elementi che la compongono.
Potremo ad esempio immaginare di **sostituire un'immagine** della pagina dopo un determinato evento, oppure **accedere ai dati immessi in una form**.
In DHTML verrà approfondita la manipolazione, tramite codice Javascript, degli elementi che costituiscono una pagina.
Prendiamo confidenza con questi array:

```
<script type='text/javascript'>
        // Preleviamo il riferimento alle 3 immagini
        var immagine1 = window.document.images[0];
        var immagine2 = window.document.images[1];
        var immagine3 = window.document.images[2];
        // Scambiamo l'immagine 1 con la 2 tramite l'attributo src
        var url_img1 = immagine1.src;
        immagine1.src = immagine2.src;
        immagine2.src = url_img1;
</script>
```

Nel codice l'esempio, legato ad una pagina contenente 3 immagini in sequenza, abbiamo **scambiato** la prima con la seconda immagine grazie alla gestione dell'attributo **src** che indica l'url delle immagini.
Analizzando il codice abbiamo innanzitutto prelevato i riferimenti alle 3 immagini tramite l'array **images[]**, appoggiato quindi in una variabile (**url_img**) il valore dell'attributo src della prima immagine, quindi scambiato tra loro le immagini tramite src.
Per far sì che il codice funzioni adeguatamente è necessario che venga posto **dopo il codice html** che imposta le tre immagini. Difatti essendo l'html interpretato dall'alto verso il basso, gli array incorporati vengono avvalorati man mano che vengono incontrati gli elementi del codice html. Posizionando quindi il codice nell'head le 3 immagini **non saranno ancora viste** perché non ancora processate dal browser. Questo esempio, oltre a dare un primo assaggio delle potenzialità di Javascript in termini di **dinamicità del codice html**, fornisce una chiara metodologia per la gestione degli array incorporati.

LE FUNZIONI.

SCRIVERE FUNZIONI.

Abbiamo già visto alcune **funzioni incorporate** di Javascript: alert(), prompt() e altre ancora.
E' possibile realizzare **funzioni personalizzate**, partendo da zero, per risolvere un determinato problema o per raggruppare istruzioni.
Nella sua forma più semplice una funzione è una **scorciatoia** per eseguire una serie di istruzioni Javascript. Quando ad esempio chiamate la funzione alert(), Javascript la interpreta come un comando per svolgere determinate attività quali aprire una finestra che include un pulsante OK, un pulsante di chiusura e visualizzare un testo nella finestra stessa.
Le funzioni create dallo sviluppatore operano anch'esse come scorciatoie.
Una dichiarazione di funzione inizia con la parola **function** seguita quindi da un nome che la identifichi:

```
function nome_funzione(){
        ...
}
```

Le regole per la denominazione delle funzioni sono simili a quelle per le variabili. Il primo carattere deve essere una lettera e gli altri possono essere lettere, numeri e trattini. Non sono consentiti altri caratteri o spazi.
Il nome della funzione è seguito da una coppia di parentesi tonde **()**. Possiamo non inserire niente tra queste, oppure elencare, separati da virgole, una serie di parametri (variabili) di input che la funzione utilizzerà.
Analizziamo il codice di questa prima semplice funzione Javascript:

```
<script type='text/javascript'>
// Definiamo una funzione chiamata somma
// Acquisisce 2 numeri, li somma e li mostra in un alert.
function somma(){
        var a = prompt('Inserisci il primo numero','');
        var b = prompt('Inserisci il secondo numero','');
        var s = parseInt(a) + parseInt(b);
        alert('La somma è: ' + s);
}
</script>
```

La funzione acquisisce due valori numeri e li somma, quindi utilizza un alert() per visualizzare il risultato.
E' buona norma inserire le funzioni nell'head dell'html o in un file esterno in modo da poterla riutilizzare anche su altre pagine.
Il passo successivo è **richiamare la funzione** quando necessario. In qualunque posto della pagina html, dopo la definizione della funzione, possiamo così scrivere:

```
<script type='text/javascript'>
        somma();
</script>
```

La funzione può essere inoltre invocata più volte all'interno dell'html.

UTILIZZARE PARAMETRI.

La funzione vista in precedenza è perfezionabile per essere resa più flessibile e performante. Si ricordi che la funzione alert() utilizza un parametro, le parole che vogliamo visualizzare nella finestra di

avviso. E' possibile riscrivere la funzione somma() in modo tale che utilizzi uno o più parametri.
In questo caso i parametri saranno i due valori numerici da sommare.
In generale i parametri di una funzione sono qualsiasi informazione di cui la funzione stessa ha bisogno per fare il suo lavoro.
Il codice della funzione somma è così riscritto:

```
<script type='text/javascript'>
// Definiamo una funzione chiamata somma
// Ha 2 numeri come parametri, li somma e li mostra in un alert.
function somma(a,b){
        var s = parseInt(a) + parseInt(b);
        alert('La somma è: ' + s);
}
</script>
```

Alla funzione vengono passati due valori (a e b) evitando quindi che questi vengano acquisiti all'interno della funzione.
Se ad esempio vogliamo sommare 10 e 3, chiameremo la funzione nel seguente modo:
somma(10,3);
Il vantaggio nell'uso dei parametri è evidente, un codice più pulito e adattabile alle diverse esigenze.
Di conseguenza quando richiameremo la funzione all'interno del codice dovremo apportare delle modifiche:

```
<script type='text/javascript'>
        var n1 = prompt('Inserisci il primo numero: ','');
        var n2 = prompt('Inserisci il secondo numero: ','');
        somma(n1, n2);
</script>
```

OTTENERE INFORMAZIONI DALLE FUNZIONI.
E' possibile scrivere anche funzioni che restituiscono delle informazioni. Consideriamo ad esempio la funzione prompt():

```
var n1 = prompt('Inserisci il primo numero: ','');
```

Quando l'utente digita il numero nella casella e fa clic su OK, il numero viene inserito nella variabile n1.
Nel gergo della programmazione si direbbe che la funzione prompt() **restituisce** i numeri e le parole digitate nella casella.
Anche le funzioni personalizzate possono restituire valori:

```
<script type='text/javascript'>
// Definiamo una funzione chiamata somma
// Ha 2 numeri come parametri, li somma e restituisce il risultato.
function somma(a,b){
        var s = parseInt(a) + parseInt(b);
        return s;
}
</script>
```

La funzione **somma(a, b)** dopo aver valorizzato la variabile **s** con la somma appunto di **a** e **b,** non effettua più l'alert del risultato bensì lo restituisce con l'istruzione **return s**.
Quando ad esempio invocheremo la funzione dobbiamo predisporre una variabile che conterrà la somma:

```
<script type='text/javascript'>
        var n1 = prompt('Inserisci il primo numero: ','');
        var n2 = prompt('Inserisci il secondo numero: ','');
        var som = somma(n1, n2);
        alert(som);
</script>
```

Con quest'ultimo perfezionamento abbiamo ottenuto una funzione minimale, adibita esclusivamente al ruolo per il quale è stata creata: **sommare due valori**. Non sono più presenti operazioni di input o istruzioni di stampa all'interno della funzione.

LEGARE UNA FUNZIONE AD UN EVENTO.
Gli ***eventi*** sono utilizzati per richiamare delle istruzioni.
Infatti il codice Javascript va eseguito in maniera sequenziale, ma per fare in modo di inserire la dinamicità e l'interattività occorre che questo resti caricato in memoria e venga attivato o richiamato solo quando si verificano particolari situazioni, come il passaggio del mouse, il caricamento di un documento, il clic su un bottone, ecc..
Gli eventi sono essenzialmente legati ai tag che compongono il codice HTML.
Ad esempio se definiamo un bottone:
<input type='button' value='Bottone' />
possiamo associare una funzione al clic del mouse da parte dell'utente. A tale scopo utilizziamo uno dei più classici eventi Javascript: onClick().
La sintassi sarà la seguente:
<input type='button' value='Bottone' onClick='nome_funzione()' />
Il browser quando interpreta il codice html lega all'evento click la funzione specificata.
Esistono decine di eventi, e costantemente in aumento, tanto che non è facile elencarli tutti; nonostante ciò è possibile raggrupparli in sezioni omogenee:

- **eventi attivabili dai tasti del mouse;**
- **eventi attivabili dai movimenti del mouse;**
- **eventi attivabili dal trascinamento del mouse (drag and drop);**
- **eventi attivabili dall'utente con la tastiera;**
- **eventi attivabili dalle modifiche dell'utente;**
- **eventi legati al "fuoco";**
- **eventi attivabili dal caricamento degli oggetti;**
- **eventi attivabili dai movimenti delle finestre;**
- **eventi legati a particolari bottoni;**
- **altri e nuovi tipi di eventi.**

Nei primi due gruppi sono inseriti quegli eventi tipici del mouse o della tastiera, come il movimento o la pressione, negli altri sono inseriti gli eventi strettamente correlati agli oggetti.
Tutti gli eventi hanno la propria sintassi composta sintatticamente dal loro nome col prefisso **on**, ad esempio l'evento **click** è stato richiamato con **onclick**.

Se passiamo col mouse su un elemento possiamo attivare **onMouseOver** e **onMouseOut**

rispettivamente per controllare l'ingresso del puntatore sull'elemento e l'uscita da esso.
Se premiamo un tasto della tastiera in un campo di testo possiamo ad esempio attivare **onKeyUp** e **onKeyDown** e così via**.**

LA FUNZIONE PER LO SCAMBIO DI IMMAGINI.

Nella sezione precedente di questa lezione si è visto come manipolare le immagini di una pagina html grazie all'array images[]. A tale scopo riproponiamo il codice:

```
<script type='text/javascript'>
        // Preleviamo il riferimento alle 3 immagini
        var immagine1 = window.document.images[0];
        var immagine2 = window.document.images[1];
        var immagine3 = window.document.images[2];
        // Scambiamo l'immagine 1 con la 2 tramite l'attributo src
        var url_img1 = immagine1.src;
        immagine1.src = immagine2.src;
        immagine2.src = url_img1;
</script>
```

Ci eravamo ripromessi di migliorarla e di legarla ad un evento.
Lo scopo da raggiungere è fornire un bottone che venga visualizzato dopo le tre immagini e che una volta premuto scambi la prima immagine con la seconda.
Il primo passo da compiere è racchiudere il codice in una funzione che chiameremo **scambioImmagini()**:

```
<script type='text/javascript'>
        function scambioImmagini(){
                // Preleviamo il riferimento alle 3 immagini
                var immagine1 = window.document.images[0];
                var immagine2 = window.document.images[1];
                var immagine3 = window.document.images[2];
                // Scambiamo l'immagine 1 con la 2 tramite l'attributo src
                var url_img1 = immagine1.src;
                immagine1.src = immagine2.src;
                immagine2.src = url_img1;
        }
</script>
```

Questo ci permette poi il richiamo del codice al momento opportuno.
Vediamo quindi la parte di codice HTML:

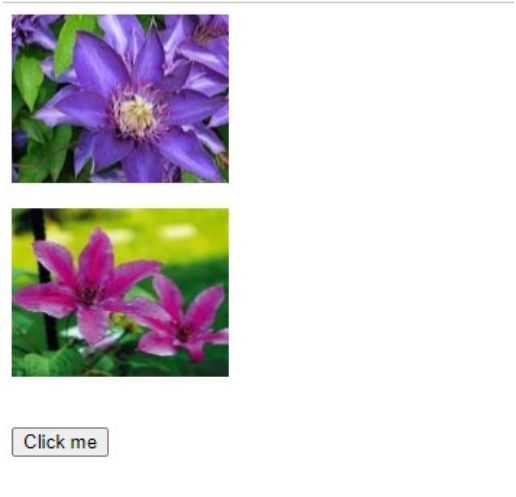
```
<body>
        <img src='img/uno.jpg' alt='Immagine'/><br/>
        <img src='img/due.jpg' alt='Immagine'/><br/>
        <img src='img/tre.jpg' alt='Immagine'/><br/>
        <input type='button' value='Scambia' onClick='scambioImmagini()'/>
</body>
```

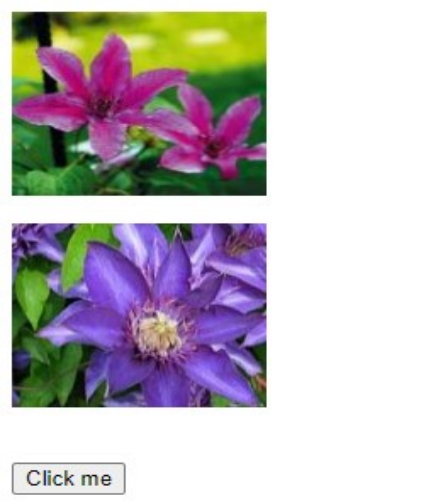
Una volta definite le tre immagini segue un bottone al quale è stato associata la funzione scambioImmagini() all'evento click (**onClick**).

Mettendo insieme la parte Javascript con la parte HTML si ottiene il codice completo:

```
<html>
<head>
<title>EVENTO</title>
<script type='text/javascript'>
        function scambioImmagini(){
                // Preleviamo il riferimento alle 3 immagini
                var immagine1 = window.document.images[0];
                var immagine2 = window.document.images[1];
                var immagine3 = window.document.images[2];
                // Scambiamo l'immagine 1 con la 2 tramite l'attributo src
                var url_img1 = immagine1.src;
                immagine1.src = immagine2.src;
                immagine2.src = url_img1;
        }
</script>
</head>
<body>
<img src='img/uno.jpg' alt='Immagine'/><br/>
<img src='img/due.jpg' alt='Immagine'/><br/>
<img src='img/tre.jpg' alt='Immagine'/><br/>
<input type='button' value='Scambia' onClick='scambioImmagini()'/>
</body>
</html>
```

Siamo pronti per testare il nostro codice. Cliccando il pulsante con etichetta '**Scambia**' vedremo sostituirsi la prima immagine con la seconda e viceversa.
Inoltre una seconda pressione del tasto effettuerà lo scambio inverso riportando le immagini al loro stato iniziale.





La funzione, legata all'evento, è solo un esempio che rende l'idea del concetto espresso all'inizio delle lezioni riguardanti Javascript, quando si è detto che **Javascript permette all' utente di interagire con la pagina html al punto tale che questa diventi dinamica.**

GESTIRE I FORM CON JAVASCRIPT.

JAVASCRIPT E I FORM HTML.
Si è mostrato fino ad ora come acquisire dati da parte dell'utente grazie alla funzione prompt(), ma indubbiamente il modo migliore per immettere dati è utilizzare i **form html**.
In precedenza abbiamo visto come realizzare in html un form e come aggiungere a questo bottoni per l'invio e per il reset, e abbiamo anche accennato a come questi dati vengono trattati dal server una volta inviati.
Javascript risulta molto utile per **validare i dati** immessi in un modulo. In altre parole sarebbe sempre preferibile controllare le informazioni immesse prima che questi vengano inviati al programma lato server che li elaborerà.
Così facendo si risparmieranno eventuali connessioni inutili per l'invio di dati errati e il modulo ne acquisterà in efficienza.
Oltre che a elaborare i dati presenti nei form, Javascript può riempire in automatico un form e gestire gli eventi legati ai singoli campi che lo compongono.

DENOMINARE GLI ELEMENTI DEI FORM.
Per poter leggere o scrivere un elemento di un modulo è necessario che Javascript possa identificarlo in maniera univoca.
Per far ciò esistono varie possibilità, la più semplice e immediata è assegnare un nome ad ogni elemento, form compreso.
L'attributo da utilizzare nell'HTML è **name**. Se ad esempio realizziamo un form con un singolo campo di testo e un bottone di submit avremo:

```
<form name='my_form' action='pagina.php' method='post'>
        Inserisci il tuo nome : <input type='text' name='nome' size='10'/>
        <input type='submit' name='invio' value='Invia' />
</form>
```

Così facendo in fase di visualizzazione nel browser il risultato non cambia se si omette o meno l'attributo name, ma per Javascript tale attributo può risultare indispensabile.
In alternativa all'attributo name è possibile utilizzare anche un metodo ancor più efficiente, cioè l'attributo **ID** (utilizzato anche nelle lezioni sul CSS):

```
<form id='my_form' action='pagina.php' method='post'>
        Inserisci il tuo nome : <input type='text' id='nome' size='10'/>
        <input type='submit' id='invio' value='Invia' />
</form>
```

L'unica cosa da tener presente quando si usa id è che ogni id elemento deve essere univoco all'interno della pagina.

LEGGERE UN CAMPO DI TESTO.
Una volta che i form e gli elementi hanno il loro nome o il loro id, Javascript può leggere facilmente il testo contenuto all'interno di un campo.
E' necessario dire a Javascript di quale form e di quale elemento si desiderano le informazioni:
window.document.nome_form.nome_elemento.value

Questa riga di codice dice a Javascript di guardare in **window**, individuare il suo **document**, trovare il **form** denominato nome_form in document, trovare l'**elemento** di form denominato *nome_elemento* in quel form e leggere il suo valore (**value**).
Riprendendo il codice HTML visto in precedenza, creiamo una funzione che legga il nome inserito e lo stampi a video tramite un alert.
Il codice completo della pagina è il seguente:

```
<html>
<head><title>Leggere campo di input</title>
<script type='text/javascript'>
        function leggiNome(){
                var nome = window.document.my_form.nome.value;
                alert('Il tuo nome è: ' + nome);
        }
</script>
</head>
<body>
        <form name='my_form' action='pagina.php' method='post'>
                Inserisci il tuo nome : <input type='text' name='nome' size='10'/>
                <input type='button' name='invio' value='Invia'
        onClick='leggiNome()' />
        </form>
</body>
</html>
```

Da notare il legame tra l'onClick e la funzione leggiNome().
Una volta inserito il nome cliccando il bottone verrà richiamata la funzione.

La stessa sintassi è da utilizzare per le textarea o i campi password.
Qualora si utilizzasse l'id per raggiungere un campo la sintassi da utilizzare nel codice Javascript è leggermente diversa:

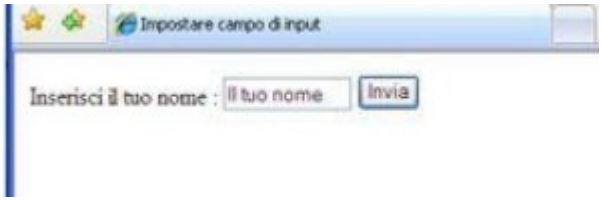
var valore_campo = window.document.getElementById('id_campo').value;

Con questa sintassi stiamo facendo riferimento direttamente al documento senza passare per il form, quindi stiamo chiedendo a Javascript di identificare l'elemento il cui id è uguale a quello passatogli come parametro.
Operare con gli id sta diventando sempre più di uso comune sia perché tale standard è supportato egregiamente da tutti i browser, sia perché spesso chi utilizza il CSS (prima di scrivere il Javascript) già imposta un id per ogni elemento e questo può essere utilizzato anche nel codice Javascript.
Un altro vantaggio, non di poco conto, è la possibilità di usare tale sintassi per far riferimento a **qualsiasi elemento** della pagina html, paragrafi, div, span, titoli etc, a condizione che ogni elemento possieda un suo id.

IMPOSTARE IL VALORE DI UN CAMPO DI TESTO.
Tramite codice Javascript è possibile non solo leggere il testo di un campo ma anche impostarlo.
Il codice è del tutto simile alla lettura:
window.document.nome_form.nome_elemento.value = valore;
Nel codice che segue impostiamo, tramite Javascript, un nome di default per il campo:

```
<script type='text/javascript'>
        function impostaNome(){
                window.document.my_form.nome.value = "Il tuo nome";
        }
</script>
```

La funzione va richiamata dopo aver definito il form nel codice html:



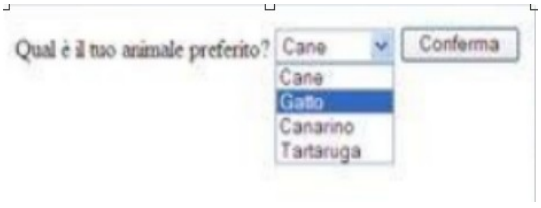
LEGGERE LA SCELTA UTENTE IN UNA SELECT.
Per accedere ad una select da Javascript la sintassi è un po' più complessa dovendo lavorare sull'array incorporato options[] legato a ciascuna select.
Ricordiamo a tal proposito che una select è un elenco di opzioni e quindi di tag option. Ognuna di queste opzioni è un elemento dell'array incorporato options[].
La sintassi per accedere ad una select è la seguente:
window.document.nome_form.nome_select.options[];
Con questa sintassi stiamo accedendo all'array delle opzioni della select **nome_select** all'interno del form **nome_form**.
Per capire quale opzioni di una select l'utente ha scelto è necessario scorrere l'intero array options[] e verificare per quale delle opzioni sia **vera** la proprietà **selected**.
Prendiamo in esempio il seguente codice html che definisce una select:

```
<form name='form' mathod='post' action='pagina.php'>
        Qual è il tuo animale preferito?
        <select name='animali'>
                <option value='Cane'>Cane</option>
                <option value='Gatto'>Gatto</option>
                <option value='Canarino'>Canarino</option>
                <option value='Tartaruga'>Tartaruga</option>
        </select>
        <input type='button' value='Conferma' onClick='animaleScelto()' />
</form>
```

Nel momento in cui l'utente seleziona una voce dal menu e clicca sul pulsante di 'Conferma', viene invocata una funzione che chiameremo **animaleScelto()**:

```
<script type='text/javascript'>
	function animaleScelto(){
		var array_opt = window.document.form.animali.options;
		for(var i=0; i<array_opt.length; i++){
			if(array_opt[i].selected==true){
				alert('Il tuo animale preferito è : ' +
					array_opt[i].value);
			}
		}
	}
</script>
```

Il codice della funzione inizia con l'avvaloramento di una variabile (**array_opt**) che conterrà l'array delle opzioni legate alla select **animali**.
Di seguito un **for** che inizia dal primo elemento e si conclude all'ultimo (**array_opt.length**) che ad ogni iterazione controlla se la proprietà **selected** per quell'opzione in esame è **true**. Nel momento in cui la condizione è vera viene stampato il **value** dell'opzione scelta.



VERIFICARE IL CLIC SU UN CHECKBOX.
Per verificare se l'utente ha spuntato un checkbox è richiesto esclusivamente il controllo dell'attributo booleano **checked**.

```
if(window.document.nome_form.nome_checkbox.checked==true){
	...
}
```

LEGGERE LA SCELTA UTENTE IN UNA SEQUENZA DI RADIO-BUTTON.
Per verificare quale radio-button di una lista sia stato spuntato dall'utente è necessario utilizzare una metodologia simile a quella descritta per le select.
Le differenze sono sostanzialmente due; innanzitutto non bisogna far riferimento all'array options[] ma ad un **array di radio-button** ed invece di testare la proprietà selected, verificheremo quella chiamata **checked**.
La sintassi è la seguente:

```
if(window.document.nome_form.nome_radiobutton[i].checked==true){
	...
}
```

Anche in questo caso avremo bisogno di un ciclo for per iterare sui radio-button.

CONTROLLARE I DATI IMMESSI IN UN FORM.
Concludiamo questa lezione con un **esempio pratico** di controllo dei dati immessi in un form.
Realizzeremo una funzione di controllo che una volta invocata verificherà che ciascun campo presente nel form risponda a dei **requisiti minimi**. Se la funzione va a buon fine, allora il dati verranno inviati al server, in caso contrario un messaggio indicherà quale errore è stato commesso.

Il modulo che implementeremo, sarà un classico modulo di **inserimento dati anagrafici**, composto da:

- nome e cognome;
- sesso (tramite radio-button);
- data di nascita;
- luogo di nascita;
- informazioni personali (in textarea).

I **controlli** da effettuare sono:

- ciascun campo deve essere avvalorato;
- la data di nascita deve contenere un valore numerico espresso come aaaammgg;
- le informazioni personali nella textarea non devono superare i 200 caratteri.

Ecco il codice del form html:

```
<form name='anagrafica' mathod='post' action='pagina.php'>
        Nome: <input type='text' name='nome' /><br/>
        Cognome: <input type='text' name='cognome' /><br/>
        Sesso: M<input type='radio' name='sesso' value='M' /> F<input type='radio'
        name='sesso' value='F' /><br/>
        Data di nascita: <input type='text' name='data' /><br/>
        Luogo di nascita: <input type='text' name='luogo' /><br/>
        Informazioni personali:<br/>
        <textarea name='info' rows='4' cols='40'></textarea><br/>
        <input type='button' value='Invia' onClick='controllo()' valign='top'/>
</form>
```

Il form è così rappresentato nella finestra del browser:

Al clic sul bottone di Invio viene invocata la procedura di controllo il cui codice è il seguente:

```
<script type='text/javascript'>
	function controllo(){
		// Prelevo i dati inseriti in ciascun campo
		var nome = window.document.anagrafica.nome.value;
		var cognome = window.document.anagrafica.cognome.value;
		var data = window.document.anagrafica.data.value;
		var luogo = window.document.anagrafica.luogo.value;
		var info = window.document.anagrafica.info.value;
		// Prelevo il riferimento ai radio-button
		var sesso_obj = window.document.anagrafica.sesso;

		// Controllo un campo alla volta
		if(nome==''){
			alert('Inserire il nome!');
			return false;
		}
		if(cognome==''){
			alert('Inserire il cognome!');
			return false;
		}
		if(sesso_obj[0].checked==false && sesso_obj[1].checked==false){
			alert('Selezionare il proprio sesso!');
			return false;
		}
		if(data==''){
			alert('Inserire la data di nascita! (aaaammgg)');
			return false;
		}else{
			if(isNaN(data)==true || parseInt(data)<19000000 ||
			parseInt(data)>30000000){
				alert('La data inserita non è corretta! (aaaammgg)');
				return false;
			}
		}
		if(luogo==''){
			alert('Inserire il luogo di nascita!');
			return false;
		}
		if(info==''){
			alert('Inserire le informazioni personali!');
			return false;
		}else{
			if(info.length>200){
				alert('Per le info personali sono consentiti massimo 200 caratteri!');
				return false;
			}
		}
		alert('Il modulo è validato correttamente');
		return true;
	}
</script>
```

Analizziamo per punti il codice della funzione controllo():

- **preleviamo i dati** inseriti in ciascun campo di testo;
- **preleviamo il riferimento ai radio-button** per la selezione del sesso;
- verifichiamo per ciascun campo di testo che questo non sia uguale a **‘’** (**campo vuoto**);
- verifichiamo che uno dei due **radio-button sia stato selezionato** tramite l’attributo checked;
- verifichiamo che la data sia innanzitutto un **valore numerico** e che sia **compreso tra due soglie;**
- verifichiamo che la **lunghezza massima del testo** nella textarea non superi i 200 caratteri, tramite l’attributo length.

Inoltre in alcune condizioni abbiamo utilizzato gli operatori logici dell’algebra booleana, per verificare più condizioni contemporaneamente. Il simbolo **||** indica l’**OR**, mentre il simbolo ***&&*** indica l’**AND**. Nella condizione in cui controlliamo il sesso verifichiamo ad esempio se entrambi i checked sono

impostati a false **(if(sesso_obj[0].checked == false && sesso_obj[1].checked == false).**
Ugualmente per la data di nascita abbiamo verificato una condizione tripla tramite l'operatore OR. In questo caso **se almeno una delle condizione è falsa** allora viene stampato il messaggio di errore data.

Infine è da notare che dopo un eventuale messaggio di errore viene effettuato il **return false**, che permette di interrompere l'esecuzione della funzione di controllo.
Qualora nessun return false venga attivato, l'esecuzione arriva fino in fondo alla funzione dove un messaggio di convalida form precede un **return true**.