Linguaggio Javascript

5.Caselle Popup

JavaScript ha 3 tipi di caselle popup: Alert box, Confirm box, Prompt Box.

Alert Box

Una alert box si utilizza per assicurarsi che l'utente legga una informazione. Per chiudere la casella e continuare la navigazione bisogna per forza premere il tasto OK.

```
alert("attenzione");
```

Vediamo un esempio:

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
  alert("Occhio..!");
}
</script>
</head>
<body>
<input type="button" onclick="show_alert()" value="Show alert box" />
</body>
</html>
```

Confirm Box

Una confirm box viene utilizzata se si vuole che l'utente verifichi e accetti qualcosa. Per proseguire, l'utente può scegliere "OK" o "Cancel", che restituiscono rispettivamente Vero o Falso.

```
confirm("confermi?");
```

Vediamo un esempio:

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
  var r = confirm("Press a button");
  if (r==true)
  {
    document.write("You pressed OK!");
  }
  else
  {
    document.write("You pressed Cancel!");
  }
}
</script>
</head>
<body>
<input type="button" onclick="show_confirm()" value="Show confirm box" />
</body>
</html>
```

Prompt Box

Una prompt box viene utilizzata per permettere all'utente di inserire un valore prima di entrare in una pagina. Anche qui, per proseguire l'utente puà scegliere "OK", che restituisce il valore immesso e "Cancel", che restituisce la stringa nulla.

```
prompt("una domanda","risposta predefinita");
```

Anche qui vediamo un semplice esempio di utilizzo:

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
 var name=prompt("Inserisci il tuo nome","Ciccio");
 if (name!=null && name!="")
 {
 document.write("Ciao " + name + "! Come stai oggi?");
 }
}
</script>
</head>
<body>
<input type="button" onclick="show_prompt()" value="Show prompt box" />
</body>
</html>
```

Esercizi

1. Testare gli esempi presentati per Alert Box, Confirm Box e Prompt Box in nuove pagine web per prendere confidenza con la sintassi e le funzionalità delle finestre modali.
2. Dato un numero intero n, inserito dall'utente, fare la somma dei primi n numeri e visualizzarlo in un popup.
3. Permettere all'utente di inserire i suoi dati personali (nome, cognome, giorno/mese/anno di nascita) tramite confirm box e visualizzare una alertbox per ogni dato errato inserito (ad esempio, nome mancante, giorno più grande di 31, etc..) permettendo un nuovo inserimento. Alla fine dovranno essere visualizzati nella pagina tutti i dati inseriti dall'utente.

6. Funzioni JavaScript

Una funzione sarà eseguita in risposta ad un evento o ad una chiamata diretta. Le funzioni possono essere inserite comodamente nella sezione head (dove saranno caricate solo on-demand) o in un file separato e chiamate da ogni punto della pagina.

Vediamo la sintassi:

```
function functionname(var1,var2,...,varX)
{
  some code
}
```

I parametri var1, var2, etc. sono variabili o valori passati alla funzione. Anche in assenza di questi le funzioni richiedono le parentesi tonde () dopo il nome. Ricordiamo inoltre che Javascript è case-sensitive!

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
  alert("Hello World!");
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```

Se la linea alert(“Hello world!!”) non fosse inserita in una funzione, sarebbe eseguita subito all'inizio del caricamento. Dentro la funzione invece, viene eseguita in risposta all'evento onclick dell'input button.

L'istruzione return

L'istruzione return viene utilizzata per specificare il valore di uscita dalla funzione. Quindi, tutte le funzioni che restituiscono un valore devono utilizzare l'istruzione return.

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
  return a*b;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3));
</script>
</body>
</html>
```

Le variabili nelle funzioni Javascript

Quando si dichiara una variabile dentro una funzione, questa è visibile solo all'interno della stessa. Quando si esce dalla funzione la variabile è distrutta e il suo valore perso. Queste variabili si definiscono ***variabili locali***. Se si vuole definire una variabile che sia visibile da tutte le funzioni della pagina, occorre definirla fuori da qualsiasi funzione. Queste variabili si dicono ***variabili globali***. Queste variabili esistono da quando sono definite fino al momento in cui si chiude la pagina.

7. Istruzioni Break, Continue, For...In

L'istruzione ***break*** provoca l'uscita forzata da un ciclo, continuando l'esecuzione del codice dall'istruzione seguente.

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
{
  if (i==3)
  {
    break;
  }
  document.write("The number is " + i);
  document.write("<br />");
}
</script>
</body>
</html>
```

L'istruzione ***continue*** interrompe solo l'iterazione corrente di un ciclo, continuando con quella successiva.

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
{
  if (i==3)
  {
    continue;
  }
  document.write("The number is " + i);
  document.write("<br />");
}
</script>
</body>
</html>
```

L'istruzione ***for...in*** esegue un ciclo all'interno degli elementi di un array o attraverso le proprietà di un oggetto:

```
for (variable in object)
{
 code to be executed
}
```

Il codice da iterare viene eseguito una volta per ogni elemento, nel caso di un'array o per ogni proprietà nel caso di un oggetto.

La variabile *argument* nella sintassi può essere un nome di variabile, un elemento array o una proprietà di un oggetto.

```
<html>
<body>
<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";
for (x in mycars)
{
 document.write(mycars[x] + "<br />");
}
</script>
</body>
</html>
```

11. I principali oggetti del JavaScript

Introduzione

Il Javascript è un linguaggio di programmazione orientato agli oggetti (OOP). Un linguaggio OOP permette di definire degli oggetti propri e specificare dei tipi personali.

In ogni caso, la creazione di oggetti generici è una pratica molto poco comune rispetto all'utilizzo degli oggetti predefiniti disponibili nel linguaggio. Quindi la cosa più importante da fare è lo studio degli oggetti Javascript esistenti.

Un oggetto non è nient'altro che un tipo speciale di dato, che può contenere proprietà e metodi.

Proprietà

Le proprietà sono i valori associati ad un oggetto, che in qualche modo ne descrivono le caratteristiche. Ad esempio una proprietà sensata per un oggetto stringa è il numero di caratteri da cui è composta.

```
<script type="text/javascript">
var txt="Hello World!";
document.write(txt.length);
</script>
```

Metodi

I metodi sono le azioni che possono essere eseguite da un oggetto. Ad esempio un'azione che un oggetto stringa può compiere è quella di modificare il suo contenuto in caratteri maiuscoli.

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.toUpperCase());
</script>
```

Oggetto String

Visti gli esempi precedenti abbiamo già appreso dell'esistenza di un oggetto String, utilizzato per contenere e manipolare il testo. Vediamone l'elenco completo di metodi e proprietà:

Proprietà dell'oggetto String

Proprietà	Descrizione
constructor	Riferimento alla funzione che ha creato l'oggetto
length	Restituisce il numero di caratteri in una stringa
prototype	Permette di aggiungere proprietà e metodi ad un oggetto

Metodi dell'oggetto String

Metodo	Descrizione
anchor()	Crea una ancora HTML (tag <a>)
big()	Visualizza la stringa in un carattere grande
blink()	Visualizza la stringa in un carattere blink
bold()	Visualizza la stringa in grassetto
charAt()	Restituisce il carattere in una posizione specifica
charCodeAt()	Restituisce il carattere UNICODE in una posizione specifica
concat()	Unisce due o più stringhe
fixed()	Visualizza una stringa con font a dimensione fissa
fontcolor()	Visualizza una stringa con uno specificato colore
fontsize()	Visualizza una stringa con una specificata dimensione
fromCharCode()	Prende il valore UNICODE e visualizza la stringa corrispondente
indexOf()	Restituisce la posizione della prima occorrenza della stringa specificata in una stringa
italics()	Visualizza una stringa in corsivo
lastIndexOf()	Restituisce la posizione dell'ultima occorrenza della stringa specificata in una stringa
link()	Visualizza una stringa come link
match()	Cerca un valore specificato in una stringa
replace()	Sostituisce alcuni caratteri con altri in una stringa
search()	Cerca una stringa per un valore specificato
slice()	Estrae una parte di una stringa e restituisce la parte estratta in una nuova stringa
small()	Visualizza una stringa in un carattere piccolo
split()	Divide una stringa in un array di stringhe

strike()	Visualizza una stringa barrata
sub()	Visualizza una stringa come subscript
substr()	Estrae un numero specificato di caratteri in una stringa, da un indice iniziale specificato
substring()	Estrae i caratteri in una stringa fra due indici specificati
sup()	Visualizza una stringa come superscript
toLowerCase()	Visualizza una stringa in caratteri minuscoli
toUpperCase()	Visualizza una stringa in caratteri maiuscoli
toSource()	Visualizza il codice sorgente di un oggetto
valueOf()	Restituisce il valore primitivo di un oggetto String

Oggetto Date

L'oggetto Date è utilizzato per lavorare con date ed orari. Il seguente codice ne crea uno:

```
var myDate = new Date()
```

Un oggetto Date contiene la data e l'orario corrente al momento della creazione. Per impostare un'ora diversa si utilizzano i metodi dell'oggetto:

```
var myDate=new Date();
myDate.setFullYear(2009,7,26);
myDate.setDate(myDate.getDate()+6);
```

Nell'esempio proposto prima si modifica la data con la funzione *setFullYear().* Attenzione però: il conto dei mesi parte da Gennaio che è il mese 0, quindi la data è..

Nella terza riga si aggiungono 6 giorni alla data precedentemente impostata, restituita con la funzione getDate(). In caso ci sia un cambio di mese o di anno questo viene gestito automaticamente.

È inoltre possibile confrontare due date fra loro:

```
var myDate=new Date();
myDate.setFullYear(2010,0,14);
var today = new Date();

if (myDate>today)
{
```

```
  alert("Today is before 14th January 2010");
}
else
{
  alert("Today is after 14th January 2010");
}
```

Vediamo l'elenco completo di metodi e proprietà:

Proprietà dell'oggetto Date

Proprietà	Descrizione
constructor	Restituisce un riferimento alla funzione Date che ha creato l'oggetto
prototype	Permette di aggiungere proprietà e metodi all'oggetto

Metodi dell'oggetto Date

Metodo	Descrizione
Date()	Restituisce la data e l'ora di oggi
getDate()	Restituisce il giorno del mese di un oggetto Date (1-31)
getDay()	Restituisce il giorno della settimana da un oggetto Date (0-6)
getFullYear()	Restituisce l'anno, come numero a 4 cifre, da un oggetto Date
getHours()	Restituisce l'ora di un oggetto Date (0-23)
getMilliseconds()	Restituisce i millisecondi di un oggetto Date (0-999)
getMinutes()	Restituisce i minuti di un oggetto Date (0-59)
getMonth()	Restituisce i mesi di un oggetto Date (0-11)
getSeconds()	Restituisce i secondi di un oggetto Date (0-59)
getTime()	Restituisce il numero di millisecondi da mezzanotte del 1 gennaio 1970
getTimezoneOffset()	Restituisce la differenza in minuti tra l'ora locale e quella di Greenwich (GMT)
getUTCDate()	Restituisce il giorno del mese da un oggetto Date, secondo il tempo universale (1-31)
getUTCDay()	Restituisce il giorno della settimana da un oggetto Date, secondo il tempo universale (0-6)
getUTCMonth()	Restituisce il mese da un oggetto Date, secondo il tempo universale (0-11)
getUTCFullYear()	Restituisce le 4 cifre di un anno da un oggetto Date, secondo il tempo universale
getUTCHours()	Restituisce l'ora da un oggetto Date, secondo il tempo universale (0-23)
getUTCMinutes()	Restituisce i minuti da un oggetto Date, secondo il tempo universale (0-59)

getUTCSeconds()	Restituisce i secondi da un oggetto Date, secondo il tempo universale (0-59)
getUTCMilliseconds()	Restituisce i millisecondi da un oggetto Date, secondo il tempo universale (0-999)
parse()	Prende una data come stringa e restituisce il numero di millisecondi da mezzanotte del 1 Gennaio, 1970
setDate()	Imposta il giorno di un oggetto Date (1-31)
setFullYear()	Imposta l'anno di un oggetto Date (4 cifre)
setHours()	Imposta l'ora di un oggetto Date (0-23)
setMilliseconds()	Imposta i millisecondi di un oggetto Date (0-999)
setMinutes()	Imposta i minuti di un oggetto Date (0-59)
setMonth()	Imposta il mese di un oggetto Date (0-11)
setSeconds()	Imposta i secondi di un oggetto Date (0-59)
setTime()	Calcola una data aggiungendo un numeso specificato di millisecondi alla mezzanotte del 1 Gennaio, 1970
setUTCDate()	Imposta il giorno di un oggetto Date, secondo il tempo universale (1-31)
setUTCMonth()	Imposta il mese di un oggetto Date, secondo il tempo universale (0-11)
setUTCFullYear()	Imposta l'anno di un oggetto Date, secondo il tempo universale (4 cifre)
setUTCHours()	Imposta l'ora di un oggetto Date, secondo il tempo universale (0-23)
setUTCMinutes()	Imposta i minuti di un oggetto Date, secondo il tempo universale (0-59)
setUTCSeconds()	Imposta i secondi di un oggetto Date, secondo il tempo universale (0-59)
setUTCMilliseconds()	Imposta i millisecondi di un oggetto Date, secondo il tempo universale (0-999)
toDateString()	Restituisce come stringa formattata la data di un oggetto Date
toLocaleDateString()	Restituisce come stringa formattata la data di un oggetto Date, secondo il tempo locale
toLocaleTimeString()	Restituisce come stringa formattata l'ora di un oggetto Date, secondo il tempo locale
toLocaleString()	Restituisce come stringa formattata la data e l'ora di un oggetto Date, secondo il tempo locale
toSource()	Rappresenta il codice sorgente di un oggetto
toString()	Converte una data in una stringa
toTimeString()	Restituisce come stringa formattata l'ora di un oggetto Date
toUTCString()	Converte un oggetto Date, secondo il tempo universale, in una stringa.
UTC()	Prende una data e restituisce il numero di millisecondi dalla mezzanotte del 1 Gennaio, 1970, secondo il tempo universale
valueOf()	Restituisce il valore iniziale di un oggetto Date

Oggetto Array

L'oggetto Array viene utilizzato per memorizzare valori multipli in una singola variabile, come ad esempio una lista di cose, di numeri, di stringhe:

```
var myCars = new Array();
myCars[0] = "Saab";
myCars[1] = "Volvo";
myCars[2] = "BMW";
```

Tutte i valori memorizzati in questo modo sono presenti in una sola variabile e accessibili tramite il loro ID unico. Se si hanno a disposizione fin da subito i valori da memorizzare è possibile utilizzare la forma ristretta di definizione:

```
var myCars = new Array("Saab","Volvo","BMW");
```

Per accedere ai valori di un array bisogna utilizzare l'indice numerico identificativo di ognuno. Da notare che l'indice comincia da 0.

```
document.write(myCars[0]);
```

per modificare i valori di un array si utilizza la stessa semplice sintassi:

```
myCars[1]="Opel";
```

Proprietà dell'oggetto Array

Proprietà	Descrizione
constructor	Restituisce un riferimento alla funzione che ha creato l'oggetto
index	
input	
length	Imposta o restituisce il numero di elementi di un array
prototype	Permette di aggiungere proprietà e metodi a un oggetto

Metodi dell'oggetto Array

Metodo	Descrizione
concat()	Unisce due o più array, restituendo il risultato
join()	Mette tutti gli elementi di un array in una stringa, separati da un delimitatore specificato

pop()	Rimuove e restituisce l'ultimo elemento di un array
push()	Aggiunge un elemento alla fine di un array e restituisce la nuova lunghezza dell'array
reverse()	Rigira l'ordine degli elementi nell'array
shift()	Rimuove e restituisce il primo elemento di un array
slice()	Restituisce gli elementi selezionati da un array esistente
sort()	Ordina gli elementi di un array
splice()	Rimuove e aggiunge nuovi elementi ad un array
toSource()	Rappresenta il codice sorgente di un oggetto
toString()	Converte un array in una stringa e restituisce il risultato
unshift()	Aggiunge un elemento all'inizio dell'array e restituisce la nuova lunghezza
valueOf()	Restituisce i valori iniziale dell'oggetto Array

Oggetto Boolean

Gli oggetti Boolean sono utilizzati per lavorare con i valori booleani (vero o falso, solitamente “true” o “false”). La sintassi è la solita per gli oggetti:

```
var myBoolean=new Boolean();
```

Inoltre una delle loro funzioni più importanti è la conversione di un valore non booleano in uno booleano.

```
var myBoolean=new Boolean();
var myBoolean=new Boolean(0);
var myBoolean=new Boolean(null);
var myBoolean=new Boolean("");
var myBoolean=new Boolean(false);
var myBoolean=new Boolean(NaN);
```

In tutti i casi descritti (nessun valore iniziale, 0, null, “”, false, NaN) l'oggetto è inizializzato a false. Negli esempi seguenti invece (true, “true”, “false”, “Richard”) l'oggetto è inizializzato a true.

```
var myBoolean=new Boolean(true);
var myBoolean=new Boolean("true");
var myBoolean=new Boolean("false");
var myBoolean=new Boolean("Richard");
```

Proprietà dell'oggetto Boolean

Proprietà	Descrizione
constructor	Restituisce un riferimento alla funzione Boolean che ha creato l'oggetto
prototype	Permette di aggiungere proprietà e metodi all'oggetto

Metodi dell'oggetto Boolean

Metodo	Descrizione
toSource()	Restituisce il codice sorgente di un oggetto
toString()	Trasforma un valore Boolean in una stringa e restituisce il risultato
valueOf()	Restituisce il valore iniziale di un oggetto Boolean

Oggetto Math

L'oggetto Math permette di eseguire operazioni matematiche, oltre che contenere numerosi costanti matematiche e metodi pronti all'uso. Nell'esempio proposto si ottiene il valore di Pi Greco e la radice quadrata di un numero:

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

La particolarità dell'oggetto Math è che non ha bisogno di essere dichiarato, ma si può utilizzare subito tramite la parola chiave Math.

Proprietà dell'oggetto Math

Proprietà	Descrizione
E	Restituisce la costante di Eulero (approx. 2.718)
LN2	Restituisce il logaritmo naturale di 2(approx. 0.693)
LN10	Restituisce il logaritmo naturale di 10(approx. 2.302)
LOG2E	Restituisce il logaritmo di E base 2 (approx. 1.442)
LOG10E	Restituisce il logaritmo di E base 10 (approx. 0.434)
PI	Restituisce PI greco (approx. 3.14159)

SQRT1_2	Restituisce la radice quadrata di 1/2 (approx. 0.707)
SQRT2	Restituisce la radice quadrata di 2 (approx. 1.414)

Metodi dell'oggetto Math

Metodo	Descrizione
abs(x)	Restituisce il valore assoluto di un numero
acos(x)	Restituisce l'arcocoseno di un numero
asin(x)	Restituisce l'arcoseno di un numero
atan(x)	Restituisce l'arcotangente di un numero
ceil(x)	Restituisce il valore di un numero arrotondato al più vicino intero dall'alto
cos(x)	Restituisce il coseno di un angolo
exp(x)	Restituisce il valore di E elevato alla x
floor(x)	Restituisce il valore di un numero arrotondato al più vicino intero dal basso
log(x)	Restituisce il logaritmo naturale (base E) di un numero
max(x,y)	Restituisce il massimo fra x, y
min(x,y)	Restituisce il minimo fra x, y
pow(x,y)	Restituisce il valore di x elevato alla y
random()	Restituisce un numero casuale fra 0 e 1
round(x)	Arrotonda un numero all'intero più vicino
sin(x)	Restituisce il seno di un angolo
sqrt(x)	Restituisce la radice quadrata di un numero
tan(x)	Restituisce la tangente di un angolo
toSource()	Rappresenta il codice sorgente di un oggetto
valueOf()	Restituisce il valore iniziale di un oggetto Math

Oggetto RegExp

L'oggetto RegExp (espressione regolare) è utilizzato per specificare cosa cercare in un testo. Quando si cerca in un testo, si specifica solitamente un pattern da ricercare al suo interno. Un espressione regolare è quel pattern.

```
var patt1 = new RegExp("e");
```

L'oggetto RegExp ha a disposizione tre metodi: test(), exec(), compile().

Il metodo ***test()*** ricerca uno specifico valore in una stringa, restituendo un boolean. L'esempio seguente conclude una ricerca positiva.

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
```

Il metodo ***exec()*** ricerca uno specifico valore in una stringa, restituendo il testo del valore trovato, oppure *null*. Nell'esempio seguente si ottiene "e" come output del metodo exec().

```
var patt1=new RegExp("e");
document.write(patt1.exec("The best things in life are free"));
```

In questo modo il metodo exec() non sembra molto interessante, ma combinandolo con un'altra proprietà dell'oggetto RegExp si ottengono buoni risultati. Se si specifica un secondo parametro "g" (global) nella definizione dell'oggetto, il metodo exec() cerca la prima occorrenza del pattern memorizzandone la posizione e proseguendo da quella alla successiva iterazione:

```
var patt1=new RegExp("e","g");
do
{
  result = patt1.exec("The best things in life are free");
  document.write(result);
}
while (result!=null)
```

Il metodo ***compile()*** è utilizzato per modificare il RegExp.

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
patt1.compile("d");
document.write(patt1.test("The best things in life are free"));
```