GUIDA JAVASCRIPT:

# Introduzione:

Il linguaggio di programmazione Javascript è sostanzialmente un linguaggio di scripting orientato agli oggetti lato client, ossia un linguaggio che permette al programmatore di scrivere script che rendono una pagina web dinamica. Tale definizione risulterà più chiara nel corso della guida analizzando nel dettaglio alcuni esempi. La sintassi di Javascript è molto simile alla sintassi del linguaggio C. Vediamo subito un banalissimo esempio in cui si può fin da subito evidenziare la struttura di base di una pagina web al cui interno è presente almeno uno script scritto in Javascript.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Primo test in Javascript</title>

**<script language="Javascript">**

**alert("Prima semplice pagina web con script JAVASCRIPT!!");**

**</script>**

</head>

<body>

Pagina di prova!!!

</body>

</html>

La parte interessante è la parte che riguarda il piccolissimo script scritto in Javascript. In sostanza, se si desidera inserire uno script in una pagina HTML è sufficiente nella intestazione della pagina stessa (quella parte racchiusa tra i due tag <HEAD>) inserire tramite il tag <SCRIPT> lo script desiderato, specificando ovviamente il linguaggio utilizzato (nel nostro caso specifico il Javascript. Ogni istruzione Javascript è terminata, come in C, con un punto e virgola finale. L’istruzione denominata **alert** permette di stampare a video un messaggio. Vediamo il seguente secondo esempio:

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"**

**"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head>**

**<title>Secondo esempio javascript</title>**

**<script type="text/javascript" src="esterno.js"></script>**

**</head>**

**<body>**

**</body>**

**</html>**

Come si può facilmente notare in questo secondo caso il codice javascript è memorizzato in un file esterno al codice HTML. Tutti gli script javascript esterni hanno un’estensione di tipo **.js**.

# Variabili e cicli

La dichiarazione dei dati in Javascript avviene in maniera totalmente automatica, ossia Javascript non è tipizzato. Per dichiarare una variabile in javascript è sufficiente utilizzare la parola chiave **var** seguita dal nome della variabile che si intende creare. Per esempio:

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"**

**"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head>**

**<title>Esempio variabili javascript</title>**

**<script language="javascript">**

**var prova;**

**prova = 2;**

**alert(prova);**

**</script>**

**</head>**

**<body>**

**</body>**

**</html>**

Il risultato di questo esempio viene mostrato di seguito:

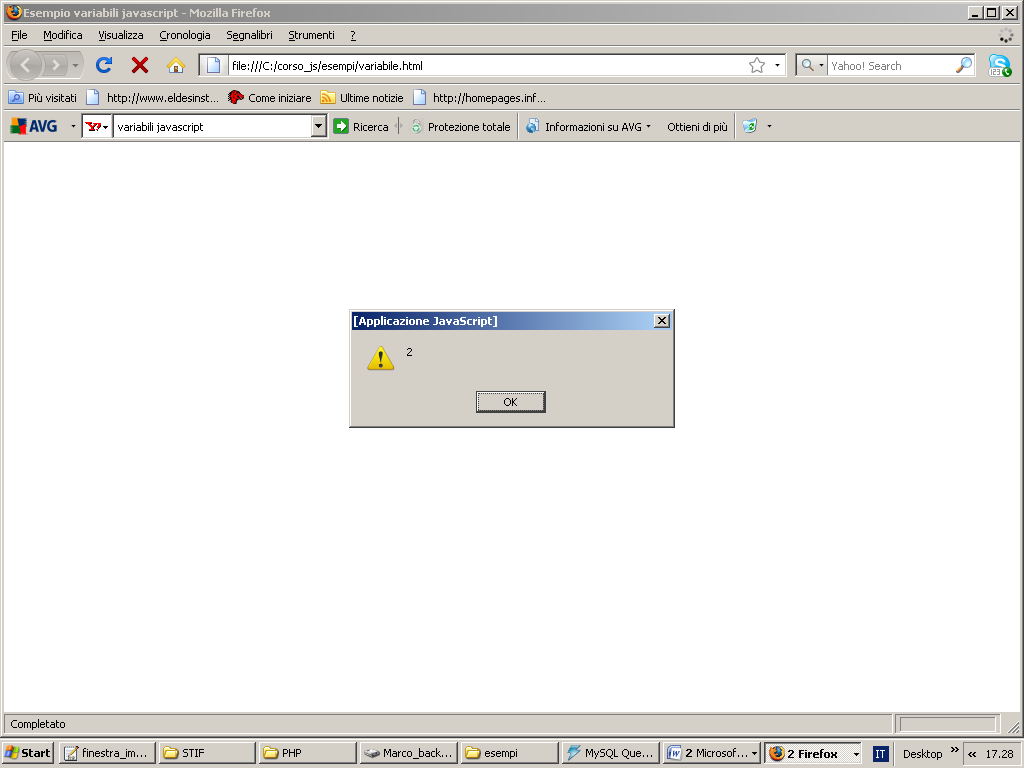


Figura 1

Per dichiarare invece una stringa basta utilizzare l’istruzione var seguita dal nome della stringa e dalla sua eventuale inizializzazione:

var str=”pippo”;

Ovviamente è d’obbligo l’uso dei doppi apici nella inizializzazione della stringa. Se invece si hanno a disposizione due stringhe e si desidera effettuare la loro concatenazione è sufficiente utilizzare l’operatore di concatenazione “+”, come viene mostrato di seguito:

stringa\_totale=str1+str2;

Per esempio:

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"**

**"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head>**

**<title>Esempio concatenazione stringhe</title>**

**<script language="javascript">**

**var str1="ciao";**

**var str2="Marco";**

**var str\_tot=str1+" "+str2;**

**alert(str\_tot);**

**</script>**

**</head>**

**<body>**

**</body>**

**</html>**

Il Javascript permette chiaramente di lavorare con i cicli. Innanzitutto vediamo i costrutti di base del linguaggio. Analizziamo per prima cosa l’istruzione if.

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"**

**"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head>**

**<title>Istruzione if</title>**

**<script language="javascript">**

**var x=2;**

**if (x==2)**

**{**

**alert("OK");**

**}**

**else**

**{**

**alert("STOP");**

**}**

**</script>**

**</head>**

**<body>**

**</body>**

**</html>**

L’esempio di per se è abbastanza auto esplicativo. In sostanza la sintassi dell’if è la seguente:

if (<espressione>) {<istruzione>}  
else (<espressione>) {istruzione}  
else (<espressione>) {istruzione}  
else (espressione) {istruzione}  
else <istruzione>;

Un altro concetto basilare di fondamentale importanza da analizzare è il concetto di ciclo. Vediamo i cicli fondamentali partendo dal ciclo **for**.

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"**

**"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head>**

**<title>Ciclo for</title>**

**</head>**

**<script language="javascript">**

**var index;**

**for(index=0; index<10; index++)**

**{**

**alert(index);**

**}**

**</script>**

**<body>**

**</body>**

**</html>**

Come si può facilmente notare la sintassi è simile alla sintassi del linguaggio C. Per questa ragione il PHP si dice anche linguaggio C-like. Il ciclo for è composto da tre condizioni: un condizione iniziale, una condizione di controllo ed una condizione di incremento. Si presti particolare attenzione al fatto che entrambe le condizioni possono cambiare. Per esempio, è possibile cambiare il valore iniziale del conteggio, oppure il limite massimo dello stesso, oppure l’incremento. Magari si desidera incrementare di 2 il valore dell’indice per ogni iterazione. Vediamo degli esempi anche degli altri cicli ossia del ciclo **while** e del ciclo **do while**. Partiamo dal ciclo while:

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"**

**"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head>**

**<title>Ciclo while</title>**

**</head>**

**<script language="javascript">**

**var index=0;**

**while(index < 5)**

**{**

**alert(index);**

**index=index+1;**

**}**

**</script>**

**<body>**

**</body>**

**</html>**

Il seguente esempio mostra l’utilizzo invece del ciclo do-while:

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"**

**"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head>**

**<title>Ciclo do-while</title>**

**</head>**

**<script language="javascript">**

**var index=0;**

**do**

**{**

**alert(index);**

**index=index+1;**

**}while(index <6);**

**</script>**

**<body>**

**</body>**

**</html>**

Un ruolo particolarmente importante lo assume l’istruzione **break** e l’istruzione **continue**. L’istruzione break posta all’interno di un ciclo permette l’uscita dal ciclo stesso. L’istruzione continue invece se viene inserita all’interno di un ciclo permette di saltare l’iterazione corrente e di passare all’iterazione successiva. Un altro costrutto molto utilizzato in javascript è lo **switch-case**. Vediamo subito un esempio:

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"**

**"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head>**

**<title>switch case</title>**

**</head>**

**<script language="javascript">**

**var carattere='A';**

**switch (carattere)**

**{**

**case 'A':**

**{**

**alert(carattere);**

**break;**

**}**

**case 'B':**

**{**

**alert("B");**

**break;**

**}**

**default:**

**{**

**alert("Errore");**

**}**

**}**

**</script>**

**<body>**

**</body>**

**</html>**

Per chiudere questo paragrafoè importante ricordarsi che in javascript i commenti su singola linea vengono inseriti tramite il simbolo ”//” mentre i commenti si più righe vengono inseriti in questo modo:

/\* commento su più

Righe \*/

# La gestione degli eventi

Un evento è un meccanismo di richiamo di uno script Javascript oppure di una determinata funzione dello script. Solitamente, Gli eventi, per poter interfacciare HTML con Javascript, non vengono definiti nel tag <script> (tranne che in qualche caso), ma sono inseriti all'interno dei tag <html>: il browser compatibile con Javascript incontrando un evento lo interpreta e lo attiva.

Il seguente esempio mostra l’uso di un evento (click del mouse):

<A href="test.html" onclick="alert('prova')">Link</A>

Cliccando sulla parola Link si aprirà la pagina denominate “test.html” e verrà visualizzato il messaggio “prova”.Un evento può essere di vario tipo. Per esempio gli eventi possono essere legati al click del mouse oppure alla digitazione di un tasto della tastiera oppure ancora ad eventi legati all’esecuzione di qualche altro programma. Possiamo definire un evento come il **verificarsi di una condizione**: dalla pressione di un pulsante, alla digitazione in una casella di testo. Quando si verifica una di queste condizioni, diciamo che il controllo genera (oppure lancia) un evento. Di seguito viene mostrata una lista dei principali titpi di eventi che spesso si possono incontrare:

1. **onClick**: attivato quando si clicca su un oggetto;
2. **onDblClick**: attivato con un doppio click;
3. **onMouseDown**: attivato quando si schiaccia il tasto sinistro del mouse;
4. **onMouseUp**: attivato quando si alza il tasto sinistro del mouse precedentemente schiacciato;
5. **onContextMenu**: attivato quando si clicca il tasto destro del mouse aprendo il ContextMenu.

Il seguente esempio mostra l’uso degli eventi:

<html>

<head>

<title></title>

<script language="Javascript1.3">

function test() {

window.alert("fase di test!!!");

}

</script>

</head>

<body>

<form>

<input type="button" value="TEST" onClick="test();">

</form>

</body>

</html>

L’esempio mostra chiaramente come viene richiamata la funzione quando si clicca su un bottone con il mouse. Un ruolo di primaria importanza riveste il concetto di evento legato al singolo oggetto di un browser. Ogni oggetto presente in un browser ha a disposizione tutta una serie di “metodi”. Per esempio l’oggetto **history** permette la navigazione a ritroso ossia la consultazione di tutte le pagine web visitate in precedenza. Tramite il metodo **back()** è possibile infatti tornare alla pagina visitata precedentemente. Il seguente esempio mostra l’uso del metodo appena menzionato:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"> <html>

<head>

<title>Esempio</title>

</head> <body>

<a href="#" onClick="history.back()">torna indietro</a>

</body>

</html>

Per informazioni più dettagliate sui vari tipi di eventi si rimanda il lettore a guide più specializzate.

# Gli array

Anche il Javascript, come del resto gli altri lunguaggi di programmazione, usano gli array. Un array è sostanzialmente una lista numerata di oggetti. Ogni oggetti è identificabile in modo univoco da un indice. Il seguente esempio mostra la dichiarazione ed il riempimento di un array:

<html>

<head>

<title></title>

<script language="Javascript1.3">

function test() {

prova = new Array(5)

prova[0] = "1";

prova[1] = "2";

prova[2] = "3";

prova[3] = "4";

prova[4] = "5";

//ciclo scansione array

for(i=0;i<prova.length;i++)

alert(prova[i]);

}

</script>

</head>

<body>

<form>

<input type="button" value="TEST" onClick="test();">

</form>

</body>

</html>

Il ciclo for visualizza il contenuto dell’array cella per cella. Si noti che l’uscita si ha quando viene raggiunta la dimensione dell’array (il numero di celle totali riempite dell’array). **“Lenght”** è un metodo applicato sul vettore che mi restituisce il numero di elementi presenti nel vettore medesimo.

# Le condizioni

Anche in Javascript è possibile controllare il flusso di esecuzione di uno script tramite l’uso dell’if. In questo paragrafo vediamo come gestire il controllo del flusso esecutivo. Il seguente esempio mostra l’uso dell’if:

<html>

<head>

<title></title>

<script language="Javascript1.3">

function test() {

i=1;

if(i==2)

alert("OK");

else

alert("NO");

}

</script>

</head>

<body>

<form>

<input type="button" value="TEST" onClick="test();">

</form>

</body>

</html>

Il programma di esempio è di una banalità sconcertante. Ad ogni modo siccome Javascript è un linguaggio C-LIKE ossia prende la sintassi del linguaggio C, per chi programma in C imparare Javascript è veramente un’operazione semplice. Il costrutto IF-ELSE è identico per molti linguaggi di programmazione. L’uguaglianza si indice con il doppio uguale (==), mentre la diversità si indica con !=. Il seguente esempio mostra un uso un pochino più complesso dell’if.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"> <html>

<head>

<title>Esempio</title>

<script language="javascript">

function test()

{

i=1;

if(i==2)||(i==1)

alert("OK");

else

alert("Errore");

</script>

</head>

<body>

<form>

<input type="button" value="TEST" onClick="test();">

</form>

</body>

</html>

Il precedente esempio mostra l’uso dell’OR come connettore logico. Un altro possibile connwettore logico è l’AND (&&).

# Il DOM

Il DOM sta per **Document Object Model** ed è il modello ad oggetti del documento. Abbiamo accennato al fatto che un browser viene visto come una collezione di oggetti in qualche modo correlati tra di loro. Per esempio, una finestra è un oggetto, un pulsante al suo interno è un altro oggetto. In generale il DOM prende come oggetto di riferimento il **document** ossia la pagina web xche si intende visualizzare su browser. Le specifiche DOM elaborate da W3C (World Wide Web) sono suddivise in livelli, ciascuno dei quali contiene moduli obbligatori o opzionali. Attualmente la specifica del DOM è al secondo livello. I livelli sono tre:

Livello 0

include tutto quello che viene fornito a DOM per la creazione del Livello 1, per esempio: document.images, document.forms, document.layers, e document.all

Livello 1

navigazione di un documento DOM e manipolazione del contenuto.

Livello 2

supporto a XML, viste filtrate e [Eventi DOM](http://it.wikipedia.org/w/index.php?title=Eventi_DOM&action=edit&redlink=1).

Livello 3

consiste in 6 specifiche differenti:

1. il nucleo del Livello 3;
2. caricamento e salvataggio del Livello 3;
3. XPath del Livello 3;
4. viste e formattazione del Livello 3;
5. requisiti del Livello 3;
6. validazione del Livello 3, che potenzia ulteriormente DOM.

Pertanto per poter capire come lavora il DOM è necessario comprendere bene come è la struttura di una pagina web (documento web). Vediamo un banalissimo esempio:

<html>

<head></head>

<body>

<h1>TEST</h1>

<p>ESEMPIO</p>

</body>

</html>

La sua struttura è la seguente:

HTML

BODY

HEAD

P

H1

Ogni elemento nel documento occupa una specifica posizione che nella struttura ad albero viene chiamato “nodo”. Pertanto ogni nodo è un elemento del documento web. Chiaramente ad ogni nodo è possibile associare un nodo chiamato “nodo genitore” tranne che per il nodo radice del documento (es: HTML). Ogni nodo può avere più figli. I nodi più importanti sono:

* **Document**, il quale è un oggetto radice del documento (es: window.document è un’istanza di questo oggetto).
* **Element**, il quale rappresenta un elemento HTML
* **Text**, il quale è un nodo di puro testo.

Quindi, per esempio, se abbiamo intenzione di identificare un determinato browser, tale identificazione può essere fatta sull’oggetto denominato **navigator**. Tale oggetto contiene il nome, la versione, ed altri dati inerenti al browser. L’oggetto racchiude le seguenti variabili:

|  |  |
| --- | --- |
| VARIABILI | DESCRIZIONE |
| navigator.appCodeName | Nome in codice del Browser che si sta utilizzando |
| navigator.appName | Nome del Browser |
| navigator.appVersion | Versione del Browser |
| navigator.userAgent | Titolo completo del Browser |
| navigator.platform | Sistema operativo dell’utente |

Il seguente esempio mostra l’uso dell’oggetto “navigator”.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"> <html>

<html>

<head>

<title>Estrai informazioni browser</Title>

<Script Language="JavaScript">

browsername=navigator.appName;

if (browsername.indexOf("Netscape")!=-1)

{

browsername="NS"

}

else

{

if (browsername.indexOf("Microsoft")!=-1)

{

browsername="MSIE"

}

else

{

browsername="N/A"

}

};

browser="0";

if (navigator.appVersion.indexOf("2.0")!=-1)

{

browser="2"

};

if (navigator.appVersion.indexOf("3.0")!=-1)

{

browser="3"

};

if (navigator.appVersion.indexOf("4.0")!=-1)

{

browser="4"

};

alert(browsername);

alert(browser);

</script>

</head>

<body>

</body>

</html>

# GLI OPERATORI

In javascript, come in qualsiasi linguaggio di programmazione sono presenti svariati tipi di operatori. Qui di seguito ne elenchiamo alcuni (i più usati) con qualche esempio chiarificatore.

* Operatore di assegnamento (=). Es: x=y, assegna a x il valore di y;
* Operatore di confronto (==). Es: x==y, verifica se x è uguale a y;
* Operatore +
* Operatore –
* Operatore \*
* Operatore /
* Operatore += (es: a+=c -> a=a+c)
* Operatore -= (es: a-=c -> a=a-c)
* Operatore \*= (es: a\*=c -> a=a\*c)
* Operatore /= (es: a/=c -> a=a/c)
* Operatore ++ di incremento (es: z++ -> z=z+1)
* Operatore - - di decremento (es: x-- -> x=x-1);
* Operatore && di AND logico
* Operatore || di OR logico
* Operatore ! di neazione

Vediamo qualche esempio:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"> <html>

<html>

<head>

<script type="text/javascript">

function and\_logico()

{

var x = false;

var y= true;

var z= true;

if(x&&y)

alert("X e Y sono uguali");

if(x&&z)

alert("X e Z sono uguali");

if(y&&z)

alert("Y e Z sono uguali");

}

</script>

</head>

<body>

<input type="button" onclick="and\_logico()" value="AND LOGICO" />

</body>

</html>

Come mostra l’esempio con l’operatore && è possibile effettuare l’AND logico tra variabili booleane o comunque tra condizioni più complesse. Medesimo discorso vale per l’OR logico. Si ricordi che è possibile

# LE CLASSI

Anche in Javascript come in molti altri linguaggi orientati agli oggetti il concetto di classe assume un ruolo fondamentale. Una classe, in prima approssimazione, è un modello che ci permette di rappresntare un qualcosa di astratto e di poterci lavorare sopra. Per esempio se vogliamo rappresetare una automobile possiamo costruire una classe denominata “auto” la quale ha delle proprietà (colore, cilindrata, tipologia,...) e dei metodi che si possono applicare su di essa (es: accelera, frena, ...). In javascript una classe viene creata tramite l’assegnazione di una variabile ad una determinata funzione. Vediamo la sintassi di seguito:

var Nome classe = function(parametri)

{

Lista istruzioni

}

Vediamone un esempio:

<html>

<script type="text/javascript">

function auto(targa, colore)

{

this.targa = targa;

this.colore = colore;

}

var car = new auto("xyz", "rosso");

alert("ww");

alert(car.targa);

</script>

</head>

<body>

</body>

</html>

Come si può facilmente notare la classe viene specificata tramite il costrutto function. S istanzia la classe, ossia si crea un oggetto ad essa associato tramite la parola chiave “new”. Come in molti altri linguaggi di programmazione per accedere alle proprietà della classe si usa la notazione puntata.

# 9. GLI OGGETTI

In javascript ci sono molti oggetti standard forniti dal linguaggio stesso. In questo paragrafo ne citiamo alcuni. Partiamo con la descrizione dell’oggetto **DATE**. Tale oggetto ci permette di lavorare con le date e con i tempi. Di seguito elenchiamo alcuni metodi più importanti:

|  |  |
| --- | --- |
| METODI | DESCRIZIONE |
| getDate() | Ottiene il giorno (1-31) usando il tempo locale |
| getDay() | Ritorno un numero (1-6) che rappresenta il giorno della settimana. |
| getFullYear() | Ritorna l’anno in quattro numeri |
| getHours() | Ritorna l’ora (0-23). |
| getMinutes() | Ritorna (0-59) i minuti dell’oggetto a cui è applicato. |
| getSeconds() | Ritorna i secondi (0-59). |
| getMonth() | Ritorna il mese (0-11). O rappresenta gennaio. |

Analogamente si ha per i metodi di settaggio di un particolare parametro. Per esempio, usando il metodo setData() imposto il giorno (1-31). Il segunte esempio mostra l’uso dell’oggetto DATE:

<html>

<script type="text/javascript">

var data = new Date();

var anno = data.getYear();

alert(anno);

</script>

</head>

<body>

</body>

</html>

Data è l’oggetto della classe Date(). Per vedere le sintassi più precise di ogni metodo riferirsi alla guida ufficiale (es: <http://www.java2s.com/Code/JavaScriptReference/Javascript-Methods/CatalogJavascript-Methods.htm>).

Un altro oggetto molto utile sopratutto per fare calcoli matematici è l’oggetto **MATH**. Tale oggetto è un oggetto che fornisce costanti e funzioni matematiche. Vediamo i principali metodi:

|  |  |
| --- | --- |
| METODI | DESCRIZIONE |
| Math.abs() | Ritorna il valore assoluto del numero specificato come parametro. |
| Math.cos() | Ritorna il coseno del numero specificato. |
| Math.sin() | Ritorna il seno del numero specificato. |
| Math.exp() | Ritorna il E elevato al numero specificato. |
| Math.log() | Ritorna il logaritmo naturale dell’espressione numerica specificata. |
| Math.max(num1, num2) | Ritorna il maggiore tra num1 e num2 |
| Math.min(num1, num2) | Ritorna il minore tra num1 e num2 |
| Math.Pow(base, esponente) | Ritorna l’elevazione della base per l’esponente |
| Math.Round() | Ritorna numero arrotondato in difetto se il parametro è inferiore a 0,5. |
| Math.sqrt() | Ritorna la radice quadrata del parametro specificato |
| Math.tan() | Ritorna la tangente del numero. |
| Math.Random() | Ritorna un numero casuale compreso tra 0 e 1. |

Vediamo un esempio:

<html>

<script type="text/javascript">

var numero = Math.exp(2);

alert(numero);

</script>

</head>

<body>

</body>

</html>

Accenniamo brevemente ora l’oggetto primitive OBJECT. Tale oggetto è l’oggetto da cui tutti gli altri discendono. Tutti gli oggetti Javascript ereditano metodi e poprietà di questo oggetto. Un altro oggetto importante è l’oggetto che ci permette di lavorare con le stringhe. Tale oggetto prende il nome di oggetto **STRING**. Qui di seguito riportiamo i metodi prencipali:

|  |  |
| --- | --- |
| METODI | DESCRIZIONE |
| Big() | Rende maiuscolo il testo |
| Bold() | Rende grassetto il testo |
| Small() | Rende piccolo il testo |
| Strike() | Rende sbarrato il testo |
| Split(separatore) | Ritorna la stringa |
| Slice(inizio, fine) | Ritorna una sottostringa che inizia da inizio e finisce in fine-1. |
|  |  |

Vediamo un esempio:

<html>

<script type="text/javascript">

var str ="prova";

alert(str.big());

var str2="prova";

alert(str2.bold());

</script>

</head>

<body>

</body>

</html>

In javascript esistono due tipi di oggetti principali:

* **Window**
* **Navigator**

Analizziamo l’oggetto “Window”. Tale oggetto è sostanzialmente una finestra del browser oppure un frame. L’oggetto Window è un oggetto globale e quindi non è strettamente necessario specificarlo per utilizzare i suo metodi. Un metodo che abbiamo già visto è il metodo **ALERT**(). Analizziamo brevemente i metodi più usati:

|  |  |
| --- | --- |
| METODO | DESCRIZIONE |
| Alert() | Visualizza messaggio a video |
| Prompt() | Permette di gestire l’input dell’utente |
| Confirm() | Per visualizzare messaggi di conferma |
| Print() | Stampa la pagina corrente senza richiedere parametri. |
| Home() | Torna all’home page settata di default nel browser |
| Blur() | Elimina il focus dalla finestra |
| Close() | Chiude la finestra corrente |

Vediamo un esempio:

<html>

<script type="text/javascript">

myWindow=window.open('','','width=200,height=100');

myWindow.document.write("<p>Nuova finestra.</p>");

myWindow.blur();

</script>

</head>

<body>

</body>

</html>

Questo esempio permette di aprire una finestra nuova ed abbandonare il focus da essa. Altri oggetti importanti legati all’oggetto WINDOW sono:

* LOCATION, il quale contiene informazioni sull’URL (**Uniform Resource Locator**) corrente.
* HISTORY, il quale è un array con le URL delle pagine visitate.

Altro oggetto che dipende da WINDOW molto usato è DOCUMENT. Per maggiori informazioni in merito a questi oggetti consultare le apposite guide più dettagliate su Javascript on line, come per esempio: <http://msdn.microsoft.com/en-us/library/yek4tbz0(v=vs.85).aspx>.