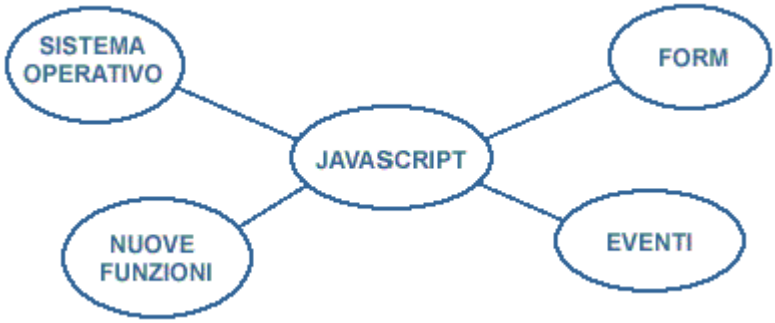
JavaScript
JavaScript

JavaScript è un **linguaggio di scripting**, ed sicuramente il più usato. Gli *script* realizzati tramite questo linguaggio possono essere incapsulati nel codice HTML. Tramite *JavaScript* è possibile rispondere alle azioni dell'utente. Ad esempio si possono convalidare i *form* prima che questi vengano trasmessi al ***server*** per una elaborazione magari errata. Infatti quando un utente inserisce un dato in un *form* questo deve essere inviato al ***server***, che dopo averlo elaborato spedisce una risposta. Grazie a *JavaScript* il *form* può essere inviato solo dopo che sia stato controllato.



Ma la potenza di *JavaScript* non si limita solo ai *form*: con esso si possono effettuare tantissimi tipi di *script* che si possono occupare dalla grafica alla *utility* più disparate. *JavaScript* è un linguaggio interpretato, infatti questo viene inviato al *client* in *file* ASCII, e quindi in chiaro, che vengono elaborati riga per riga nei *browser* in modalità *runtime*.

- **Pro.** Il **linguaggio di scripting** è più sicuro e affidabile perché in chiaro. Il codice *JavaScript* viene eseguito sul *client*, per cui i ***server*** sono sollecitati molto di meno. Codice visibile e leggibile da chiunque, ho inserito questa voce nei pro perché penso che solamente condividendo le risorse si possa avanzare tecnologicamente.
- **Contro.** Occorre scaricare completamente il codice dello *script* per essere eseguito. Con *JavaScript* non è possibile fare tutto, questo per ragioni di sicurezza, per cui per effettuare alcune operazioni occorre ricorrere a linguaggi più potenti tipo ***Java*** (*JavaScript* e ***Java*** non sono affatto la stessa cosa).

Le versioni

Nel corso degli anni la *NetScape Communications Corporation* ha dato vita a sempre più aggiornate versioni di *JavaScript*. La prima versione standardizzata di *JavaScript* fu riconosciuta nel giugno dell'1997 dall'ECMA, era la versione 1.1 e fu definita *ECMaScript* o *ECMA-262*. Per dovere di cronaca l'*ECMA* è una associazione internazionale di industrie basate sull'Europa dedicata alla standardizzazione di sistemi di comunicazioni e informazioni.

A *JavaScript* della *NetScape*, disponibile dalla release 2.0, rispose la *Microsoft* con *JScript* disponibile su *Internet Explorer* 3.0 simile a *JavaScript*.

	JavaScript				***JScript***		
Versioni	1.0	1.1	1.2	1.3	1.0	3.0	5.0
NS 2.0	*						
NS 3.0	*	*					
NS 4.0	*	*	*				

NS 4.06	*	*	*	*			
MSIE 3.0	*				*		
MSIE 4.0	*	*	*		*	*	
MSIE 5.0	*	*	*	*	*	*	*

NS sta per *NetSCape*, MSIE per *Internet Explorer*.

Tag Script e NoScript

Per inserire uno *script* all'interno di una pagina HTML occorre utilizzare il ***tag*** `<SCRIPT>`. Questo ***tag*** è possibile inserirlo in qualsiasi posizione della pagina, l'importante è chiuderlo.

Gli *script* possono essere posizionati tra i ***tag*** `<HEAD>` in modo che siano caricati per primi, importante se si utilizzano delle variabili per gestire la pagina e per inserirvi delle funzioni che vengono avviati da eventi attivati sulla pagina, oppure in qualsiasi parte della pagina.

Se ne possono inserire in una misura qualsiasi l'importante è chiuderli. Il *browser*, infatti, legge la pagina dall'alto verso il basso, quando incontra il *Tag* `<SCRIPT>` continua a leggere sempre nello stesso verso ma interpreta le righe in maniera diversa, per cui se il ***tag*** non viene chiuso con l'apposito ***tag*** `</SCRIPT>` anche la restante parte della pagina viene interpretata come codice *JavaScript* con conseguente errore nella esecuzione.

In caso di errore nella fase di esecuzione si possono verificare due comportamenti diversi da parte del *browser*:

- viene visualizzata la pagina ma il codice errato non viene eseguito.
- Se lo *script* genera un *loop* la pagina può o restare bianca o essere parzialmente visualizzata perché l'esecuzione del codice HTML è stato interrotto, e quindi verrà visualizzato solo il codice antecedente allo *script* che ha generato il *loop*.

Per inserire uno *script* in una pagina occorrono queste righe:

```
<SCRIPT>
<!-- Istruzioni JavaScript --!>
</SCRIPT>
```

Dato che i linguaggi di *scripting* sono diversi occorre specificare a quale linguaggio associare lo *script*, e quindi evitare che si utilizzi quello non voluto:

```
<SCRIPT language=JavaScript>
<!-- Istruzioni JavaScript --!>
</SCRIPT>
```

In questo modo si indica che lo *script* è in codice *JavaScript*. Se l'utente utilizza un *browser* che non supporta *JavaScript*, oppure è disabilitato, esiste un ***tag*** grazie al quale è possibile impedire la visualizzazione errata della pagina.

```
<NOSCRITP>
sezione per browser che non supportano JS
</NOSCRIPT>
```

Richiamo degli Script

Uno *script* può essere inserito in due modi all'interno di una pagina HTML:

- inserendo il codice nel documento.

- Caricando il codice da un *file* esterno.

Per inserire il codice *JavaScript* direttamente nel documento occorre inserire le istruzioni tra i ***tag*** `<SCRIPT>` e `</SCRIPT>` come spiegato nella **pagina precedente**.

```
<HTML>
<HEAD>
<TITLE>Script interno </TITLE>
</HEAD>
<BODY>
<BR> Questa stringa è scritta con l'Html <BR>
<SCRIPT language=JavaScript>
<!-- document.write (Questa con JavaScript); //-->
</SCRIPT>
<BR> Di nuovo HTML.
</BODY>
</HTML>
```

Caricare uno *script* da un *file* esterno può essere utile quando questo deve essere utilizzato su più pagine. Il *file* esterno può essere richiamato tramite un *file* ASCII che avrà estensione .js, la sintassi da inserire all'interno della pagina HTML è la seguente:

```
<SCRIPT language=JavaScript src=nome_del_file.js>
<!-- //-->
</SCRIPT>
```

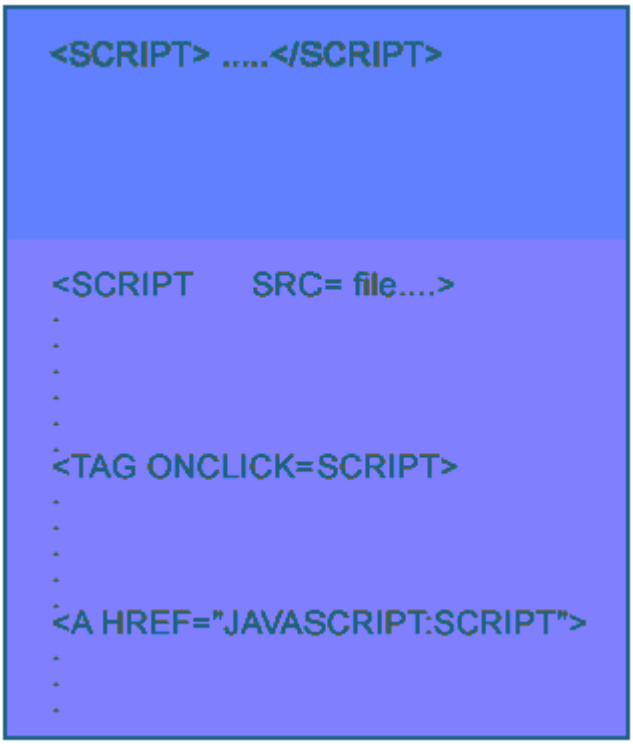
Il *file* può essere scritto con qualsiasi *editor* testuale, ma è importante che non contenga ***tag*** di apertura e chiusura degli *script*. In alternativa lo *script* può essere esterno. Scrivere con il blocco note la riga seguente e salvare il documento dandogli il nome prova.js.

```
<HTML>
<HEAD>
<TITLE>Script interno </TITLE>
</HEAD>
<BODY>
<BR> Questa stringa è scritta con l'Html <BR>
<SCRIPT language=JavaScript src=prova.js>
<!-- //-->
</SCRIPT>
<BR> Altro codice HTML.
</BODY>
</HTML>
```

Modalità di esecuzioni

Le istruzioni *JavaScript* possono essere eseguite anche diversamente rispetto ai casi trattati finora, infatti, facendo anche un riferimento anche alle precedenti lezioni, le istruzioni *JavaScript* possono essere eseguite:

- all'interno degli *script* (tra i **tag** `<SCRIPT>`);
- caricandole da un *file* esterno;
- in seguito all'attivazione di un evento;
- in luogo di un *link*: (da NS 3.0) nella forma `<A HREF: javascript:comando>;`;
- valori *JavaScript* possono essere richiamati dall'HTML includendoli tra i caratteri & { e };%.



Si può creare uno *script* che preleva l'ora d'inizio del caricamento della pagina sul *client* e la conservi nella **variabile** `orainizio`. Mettiamo il caso che vogliamo scrivere questo valore in un *textbox*, che può essere visibile all'utente anche dopo molto tempo dall'inizio del caricamento della pagina, per fare ciò ci occorre inserire queste semplici righe:

```
<INPUT type=text size=10 value=&{orainizio};%></INPUT>
```

in questo modo il campo `visualizzare` ha il valore della **variabile** `orainizio`.

Eventi

Gli eventi sono utilizzati per richiamare le istruzioni. Dato che l'esecuzione degli *script* è sequenziale per inserire della dinamicità all'interno delle pagine occorre che alcune funzioni vengono lanciate solo quando l'utente compie una particolare azione tipo cliccare su un pulsante, completare il *download* di un immagine e così via.

Ad un evento può essere associata un'unica istruzione, ma di solito l'associazione viene fatta con un blocco di istruzioni, le funzioni, che prendono il nome di *handler* o gestori di eventi. Per interfacciare HTML e *JavaScript* gli eventi non sono inseriti nei ***tag*** `<SCRIPT>` ma nei ***tag*** dell'HTML. Quando un *browser* compatibile con ***Javascript*** incontra un evento lo interpreta e lo attiva.

Questa è la sintassi generale per creare un *handler* per i ***tag*** HTML:

```
<TAG onEvento=JavaScript Code>
```

dove `TAG` è un ***tag*** dell'HTML compatibile con l'evento, `onEvento` è il nome dell'evento, e `JavaScript Code` è la sequenza *JavaScript* che si vuole attivare. Per esempio:

```
<FORM name=prova>
<INPUT type=Text size=15></INPUT>
```

```
<INPUT type=Button value=Controlla onClick=Controlla(text.value)></INPUT>
</FORM>
```

Gli eventi si possono attivare anche all'interno degli *script*, come se fossero proprietà dell'oggetto:

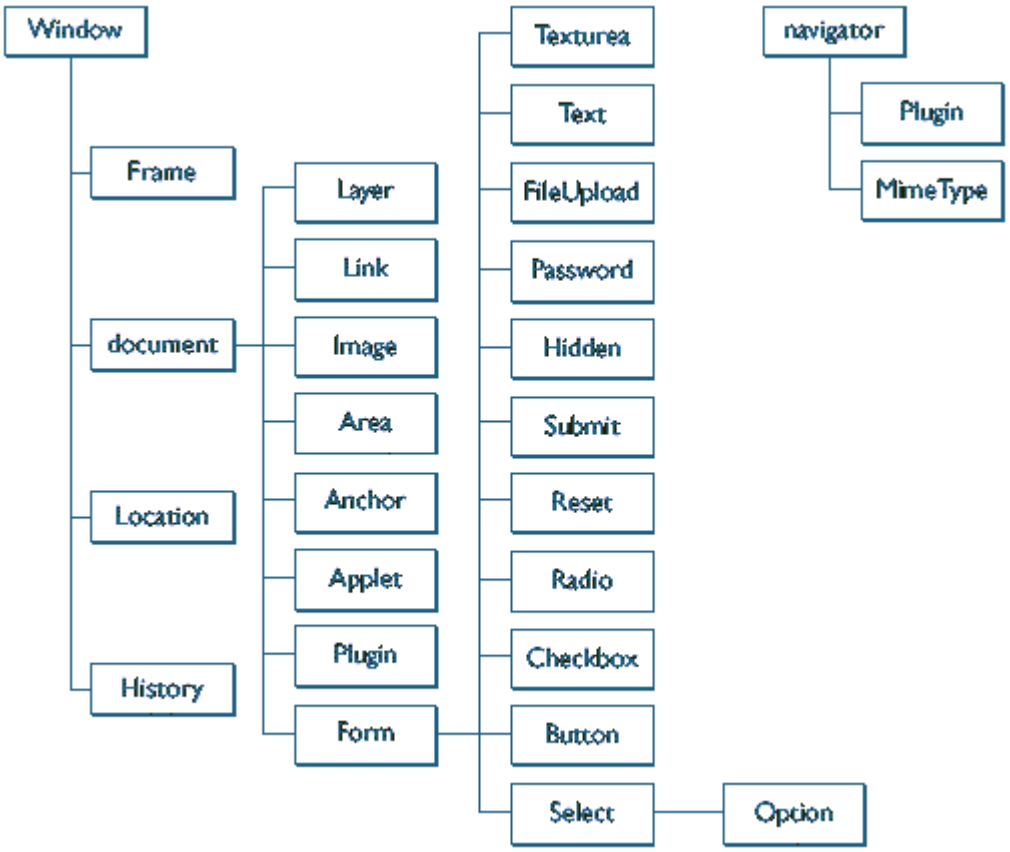
```
Oggetto.evento=handler;
```

Eventi disponibili

Evento	Si verifica quando	TAG	Versione
onAbort	quando l'utente clicca un *link* o si preme *Stop* nella barra dei comandi del *browser*	IMG	1.1
onBlur	l'oggetto sulla pagina perde il *focus*	SELECT, TEXTAREA, INPUT (TEXT)	1.0
onChange	il contenuto di un campo di un *form* è modificato e non più selezionato	SELECT, TEXTAREA, INPUT (TEXT)	1.0
onClick	*click* su un oggetto o su un *link*.	A, INPUT (tutti)	1.0
onDblClick	doppio *click* del *mouse*	BODY, A	1.2
onDragDrop	*drag & drop* sulla finestra	*Window*	1.2
onError	il caricamento dà un errore	*MG and Window*	1.1
onFocus	un oggetto sulla pagina acquisisce il *focus*	SELECT, TEXTAREA, INPUT (TEXT)	1.0
onKeyDown	viene premuto un tasto	BODY, IMG, A, INPUT (TEXTAREA)	1.2
onKeyPress	si preme e poi rilascia un tasto o lo si tiene premuto	BODY, IMG, A, INPUT (TEXTAREA)	1.2
onKeyUp	tasto precedentemente premuto è stato rilasciato	BODY, IMG, A, INPUT (TEXTAREA)	1.2
onLoad	una pagina o un'immagine finisce il suo caricamento	BODY, FRAMESET	1.0
onMouseDown	si preme un pulsante del *mouse*	BODY, A e i Bottoni	1.2
onMouseMove	si muove il *mouse*	nessuno per *default*	1.2
onMouseOut	il *mouse* esce fuori dall'oggetto	A, Mappe Cliccabili	1.1
onMouseOver	il *mouse* si muove su un oggetto	A, Mappe Cliccabili	1.1
MouseUp	si rilascia un pulsante del *mouse*	A, Mappe Cliccabili	1.1
onMove	si muove una finestra o un *frame*	*Window*	1.2
onReset	il tasto annulla di un *form*	FORM	1.1
onResize	si ridimensiona una finestra	*Window*	1.1
onSelect	selezione di testo	INPUT (TEXT)	1.0
onSubmit	è abbinato al tasto invio del *form*	FORM	1.0
onUnload	si rilascia una finestra	*Window*	1.0

Oggetti Navigatore

Quando viene caricata una pagina nel Navigatore del *Browser* vengono creati un numero di oggetti *JavaScript* settati in base all'HTML e ad altre informazioni pertinenti. La gerarchia di questi oggetti, che rispecchia la struttura di una pagina HTML, è la seguente:



In questa gerarchia, un oggetto discendente è una proprietà dell'oggetto da cui discende. Per esempio una *form* chiamata *theForm* è un oggetto così come una proprietà di *document*, ed è referenziata in questo modo: *document.theForm*

Ogni pagina ha i seguenti oggetti:

- ***navigator*:** è utilizzato per acquisire informazioni sul *browser* utilizzato dall'utente, per utilizzare i *plug-in* installati, e per i *MIME* supportati dal *client*;
- ***window*:** le sue proprietà sono destinate completamente alle finestre;
- ***document*:** contiene le proprietà basate sul contenuto del documento come il titolo, *links*, *form*;
- ***location*:** le proprietà basate sull'*URL* corrente;
- ***history*:** contiene la storia di navigazione del *client*.

Per riferirsi ad una specifica proprietà bisogna specificare il nome dell'oggetto e tutti i suoi antenati:

```
document.theForm.text1.value=prova
```

Nella precedente stringa si referenzia la proprietà *value* di un campo testo, *text1*, contenuto nel *form theForm* del documento corrente.

Definizione e chiamata di funzioni

Le funzioni in *JavaScript* sono l'elemento portante del linguaggio. Una funzione non è altro che una procedura *JavaScript* capace di compiere una azione specifica. Per definire una funzione occorrono quattro componenti:

- la *Keyword function*.
- Il nome della funzione.

- Gli argomenti della funzione compresi tra le parentesi tonde e separati dalle virgole.
- Le istruzioni *JavaScript* comprese tra le parentesi graffe.

Grazie alle funzioni è possibile scrivere codice più conciso, infatti, si può scrivere un gruppo di istruzioni, assegnarvi un nome e quindi eseguire l'intero gruppo in qualsiasi momento richiamandolo e specificando le informazioni necessarie. Le informazioni da passare alla funzione devono essere specificate tra parentesi tonde dopo il nome della funzione. Di solito le funzioni vengono inserite all'interno del documento nella sezione *HEAD* in modo che questa possa essere caricata subito e resa sempre disponibile all'interno della pagina.

```
<HEAD>
<SCRIPT LANGUAGE=JavaScript>
<!--
function quadrato(x) {
return x*x;
}
//-->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT>
document.write (La funzione ritorna , quadrato(5), .);
</SCRIPT>
<P>OK?</P>
</BODY>
```

La funzione definita prende il nome di quadrato, possiede un unico parametro x, è composta da un unica istruzione: *return* x*x. La funzione poi viene richiamata all'interno della sezione `BODY`, semplicemente specificando il nome della funzione e il valore del parametro (il parametro passato può essere anche una **variabile**).

JavaScript possiede una serie di funzioni predefinite che sono le seguenti:

- *Escape*.
- *Eval*.
- *isFinite*.
- *isNaN*.
- *Number*.
- *parseFloat*.
- *parseInt*.
- *String*.
- *Taint*.
- *Unescape*.
- *UnTaint*.

Variabili, valori e letterali

JavaScript riconosce i seguenti tipi di valore:

- **Numeri** (interi o decimali).
- **Valori Logici** (*Boolean*). Può avere due stati: *true* e *false*. Nei confronti le espressioni con risultati 0 vengono considerate false, mentre le istruzioni che danno come risultato un numero diverso da 0 sono considerate vere.
- **Stringhe**, come Pippo. Una stringa contiene zero o più caratteri racchiusi tra virgolette semplici (' ') o doppie(''). La stringa deve essere delimitata dallo stesso tipo di virgoletta.
- **Null**.

JavaScript è *case-sensitive*, per cui *null* è diverso da *Null*, come *myvar* è diversa da *myVar*. Quando si dichiarano le variabili non occorre specificare il tipo di dato che andrà a contenere: verrà fatto a seconda dell'assegnamento:

```
var myVar=24
```

Assegna a *myVar* il valore 24 e questa **variabile** viene definita automaticamente di tipo numerica. Se nel corso dello *script*, contenente l'istruzione si aggiunge la riga:

```
myVar=Questa è una stringa
```

Non viene generato nessun errore: la **variabile** *myVar* generata per prima (*var myVar*=24) sarà considerata di tipo numerico, mentre la seconda sarà considerata di tipo *String*, e la prima istanza viene persa. Non tutti i nomi possono essere assegnati ad una **variabile**, infatti, si devono rispettare alcune direttive:

- il nome della **variabile** deve iniziare con una lettera o con il carattere _ (trattino basso).
- può contenere le cifre numeriche (0-9).
- può includere le lettere comprese da A fino Z e le lettere comprese da a fino z estremi esclusi, c'è differenza tra la prima serie e la seconda in quanto *JavaScript* è *case-sensitive*.

Operatori

Gli operatori possono essere unari o binari, i primi richiedono solamente un operando al contrario dei binari che ne vogliono due. ***Javascript*** ha questi tipi di operatori:

- Assegnamento.
- Confronto.
- Aritmetici.
- *Bitwise*.
- Logici.
- Stringhe.

Operatori di Assegnamento

Assegna il valore dell'operando a destra dell'operatore all'operando presente alla sinistra dell'operatore. L'operatore di assegnamento base è l'uguale (=). La tabella seguente contiene gli altri operatori di assegnamento e sono riportati sia con la forma abbreviata che con quella integra.

Forma abbrevviata	**Forma integra**
x+=y	x=x+y
x-=y	x=x-y
x*=y	x=x*y
x%=y	x=x%y
x<<=y	x=x<<y
x>>=y	x=x>>y
x>>>=y	x=x>>>y
x&=y	x=x&y
x^=y	x=x^y

x|=y x=x|y

Operatori di confronto

Compara due operandi e ritorna un valore logico a seconda dell'esito del confronto. Se l'esito è positivo ritorna 1, altrimenti 0. Gli operatori di confronto sono i seguenti:

- == vero se gli operandi sono uguali.
- != vero se gli operando non sono uguali.
- > vero se l'operando di sinistra è maggiore di quello destro.
- >= vero se l'operando di sinistra è maggiore o uguale di quello destro.
- < vero se l'operando di sinistra è minore di quello destro.
- <= vero se l'operando di sinistra è minore o uguale di quello destro.

Operatori di Aritmetrici

Prelevano dei valori numerici per elaborarli e ritornare un singolo valore numerico.

- + (somma).
- - (sottrazione).
- * (moltiplicazione).
- / (divisione).
- % (resto intero).

Altri operandi aritmetici, sono l'incremento e il decremento e il meno unario. Questi al contrario dei precedenti sono operatori unari.

- ++ Incrementa di un unità.
- -- Decrementa di un unità.
- - Rende negativo un numero.

Operatori Logici

Ritornano due valori: 0 se l'espressione logica è vera, 1 se l'espressione è falsa. Gli operatori logici sono:

- ***expr1 && expr2*** *(and)* vero solo se entrambi gli operandi sono veri.
- ***expr1*** || ***expr2*** *(or)* vero se almeno uno degli operandi è vero.
- ***!expr*** *(not)* è la negazione dell'argomento.