HTML

FORMATTAZIONE...

- **HTML** è l'acronimo di **H**yper**T**ext **M**arkup **L**anguage ("Linguaggio a marcatori per gli Ipertesti").
- Non è un linguaggio di programmazione non ha, cioè, meccanismi che consentono di prendere delle decisioni ("in questa situazione fai questo, in quest'altra fai quest'altro"), e non è in grado di compiere delle iterazioni ("ripeti questa cosa, finché non succede questo"), né ha altri costrutti propri della programmazione.
- Si tratta invece di un linguaggio di contrassegno (o 'di marcatura'), che permette di articolare gli elementi di una pagina in blocchi le cui caratteristiche vengono definite attraverso degli appositi marcatori, detti "**tag**".

... E SEMANTICA

- In origine HTML è stato concepito principalmente per formattare il testo: elementi con un preciso significato semantico si mischiavano con elementi di pura formattazione.
- Con l'evoluzione di html in xhtml e in html 5 i tag hanno sempre più assunto il compito di articolare la pagina i blocchi semantici:
  - Tutti i tag di pura formattazione (grassetto, corsivo, centrato) vanno considerati elementi deprecati.
  - Al contrario con html 5 sono stati introdotti tag che hanno l'unica funzione di definire in maniere più robusta le parti in cui è articolata una pagina Web (testata, piè di pagina, barra di navigazione, ecc.) dal punto di vista dell'organizzazione dei contenuti e dalla navigazione..
- Il compito di definire l'aspetto di una pagina è affidato ai fogli di stile.

I MARCATORI (TAG)

- I **tag** vanno inseriti tra parentesi uncinate: **<TAG>**
- La chiusura del tag viene indicata con una barra: **</TAG>**
- Il contenuto che il tag modifica va inserito tra l'apertura e la chiusura del tag medesimo:

```
Questa <span style="font-wieight:bold">parola</span> è in
                                grassetto.
```

- che nel rendering verrà reso:

Questa **parola** è in grassetto.

- Alcuni tag non hanno (o possono non avere) contenuto (**empty tag**) . Ad esempio l'interruzione di linea la indico così:

```
<br />
```

GLI ATTRIBUTI

- Le caratteristiche [illegible] del tag. Ogni tag ha [illegible] so modificare:

  <TAG attri[illegible]TAG>

- Alcuni attribu[illegible].), altri sono specifici

  <IMG widt[illegible] />

- Una caratteristica importante del codice HTML è che i tag possono essere annidati l'uno dentro l'altro.
- È quindi opportuno usare l'indentazione. Grazie ad essa il codice HTML risulta più leggibile.

```
<TAG1 attributo = "valore">
        contenuto 1
        <TAG2>
                contenuto 2
        </TAG2>
</TAG1>
```

CSS

REGOLE

- Un foglio di stile è costituito da una serie di regole che stabiliscono come un elemento (identificato da un selettore) viene reso su un media.

- Esempio:

```
p{
        font-family: Verdana, sans-serif;
        font-size:16px;
  }
```

SELETTORI

ELEMENTI

CSS 1.0

È il più semplice dei selettori. È costituito da uno qualunque degli elementi di (X)HTML.

```
h1 {color: #000000;}
p {background: white; font: 12px Verdana, arial, sans-serif;}
table {width: 200px;}
```

CSS 1.0

È possibile nei CSS raggruppare diversi elementi al fine di semplificare il codice.

```
h1 {background: white;}
h2 {background: white;}
h3 {background: white;}
h1, h2, h3 {background: white;}
```

CSS 2.1

```
* { color: black; }
```

CLASSI E ID

- In questa pagina abbiamo assegnato al paragrafo l'attributo class="testorosso":

  `<p class="testorosso">....</p>`

- Possiamo ora creare un regola e assegnargli il nome testorosso:

```
.testorosso {
   font: 12px arial, Helvetica, sans-serif;
   color: #FF0000;
}
```

- **In un documento potrò avere senza problemi questa situazione:**

  `<p class="testorosso">....</p>`
  `<div class="testorosso">....</div>`
  `<table class="testorosso">...</table>`
  `<p class="testorosso">....</p>`

- **E l'elemento seguirà la regola definita nella classe testorosso**

CLASSI E ID

CSS 1.0

La sintassi di un selettore ID è semplicissima. Basta far precedere il nome dal simbolo di cancelletto #:

```
#titolo {
  color: blue;
}
```

- assegniamo il colore blue all'elemento che presenti questa definizione:

```
<h1 id="titolo">...</h1>
```

CSS 1.0

Come per le classi è possibile usare una sintassi con elemento:

```
p#nome_id {
  color: red;
}
```

- Ma non ha senso perché l'id per sua natura dovrebbe essere unico.

PSEUDO-CLASSI

- Una pseudo-classe non definisce un elemento ma un particolare stato di quest'ultimo. In buona sostanza imposta uno stile per un elemento al verificarsi di certe condizioni.
- A livello sintattico le pseuso-classi non possono essere mai dichiarate da sole, ma per la loro stessa natura devono sempre appoggiarsi ad un selettore.

```
a:link {color: blue;}
```

  - La regola vuol dire: i collegamenti ipertestuali (<a>) che non siano stati visitati (:link) avranno il colore blue.

JAVASCRIPT

STORIA

1992 **Mosaic**, il primo browser, permette di visualizzare la grafica oltre al testo.

1994 Parte degli sviluppatori di Mosaic fondarono la **Netscape Communications Corporation**

1995 La **Sun Microsystems Inc**. presenta **Java**, il linguaggio evoluto che si propone di diventare uno standard nella comunicazione in rete.

1995 **Javascript** viene implementato per la prima volta sulla versione beta di Netscape Navigator 2.0

1995 Microsoft entra nel mercato di Internet. Acquista Mosaic e lancia **Internet Explorer 2.0** e tenta di contrapporre a Javascript una versione ridotta del Visual Basic che prende il nome di **VBScript**.

1996 La Microsoft con Internet Explorer 3.0 decide di affiancare a **VBScript** di un linguaggio che di fatto èmolto simile a Javascript, ma che per esigenza di copyright, non può avere lo stesso nome, per cui fu definito **JScript**.

COSA È JAVASCRIPT

- **Script** in inglese significa "copione" o "sceneggiatura",.
- Il browser legge una riga, la interpreta e la esegue, poi passa alla successiva e fa la stessa cosa, e così di seguito fino alla fine dello script.
- Javascript è un linguaggio interpretato: Cosa significa?

LINGUAGGI COMPILATI

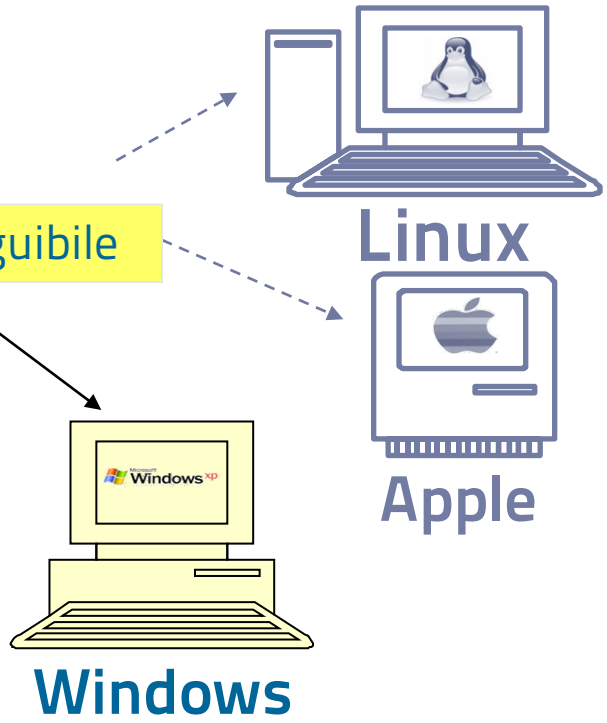
- Linguaggi abbastanza complessi in cui il sorgente (un file di testo con le operazioni da eseguire) viene **compilato** in **codice macchina** e viene impacchettato in un particolare file detto eseguibile che il computer è in grado di eseguire;
- È specifico di un determinato sistema operativo e la compatibilità tra sistemi diversi può essere garantita solo dal fatto che esistano compilatori specifici per ogni sistema.

LINGUAGGI COMPILATI

Script e risorse: (che variano a seconda del linguaggio usato)

Compilatore specifico del Sistema Operativo (Di solito in più passaggi)

File eseguibile

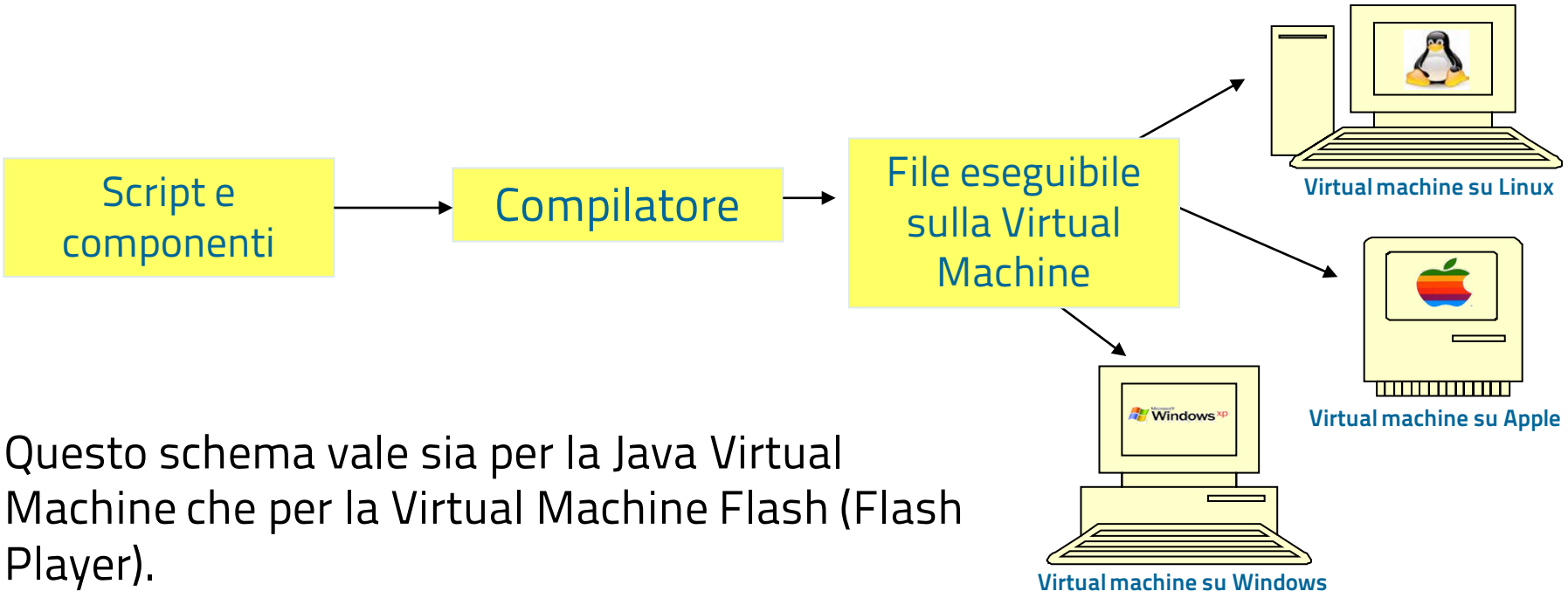


Un file di testo (file sorgente) viene trasformato in una serie di istruzioni semplici che il processore del computer è in grado di eseguire direttamente (codice macchiana)

LINGUAGGI SEMICOMPILATI

- Per semplificare la gestione della compatibilità non si compila il sorgente per uno specifico ambiente ma per una macchina virtuale.
- Per essere eseguiti i programmi presuppongono che la macchiana virtuale sia installata.

LINGUAGGI COMPILATI



Questo schema vale sia per la Java Virtual Machine che per la Virtual Machine Flash (Flash Player).

LINGUAGGI INTERPRETATI

Lo script viene eseguito immediatamente: uno script javascript viene interpretato dal browser e da un output sul monitor, sulla stampante, un output audio, ecc.

JAVASCRIPT NON È JAVA

- Con **java** si realizzano:
  - programmi (come StarOffice e OpenOffice)
  - applet (ma sono in disuso)
  - applicazioni lato server (J2EE, servlet, jsp...)
- Con JavaScript potete intervenire "solo" sulle vostre pagine web.

IL TAG SCRIPT

- Possiamo inserire il codice JavaScript in qualsiasi parte del documento (nella head oppure nel body) a seconda delle nostre esigenze.
- Per scrivere la sintassi è sufficiente aprire il tag **<SCRIPT>**. Il codice JavaScript va inserito tra l'apertura e la chiusura del tag. Così:

```
<script type="text/javascript">
  alert("ciao");
</script>
```

IL TAG SCRIPT

- La sintassi cha abbiamo appena visto può violare le regole di XHTML strict. In un codice possono essere infatti presenti costrutti illegali per il testo XML.
- Si può evitare così:

```
<script type="text/javascript">
/* <![CDATA[ */
  alert("ciao");
/*]]>*/
</script>
```

NO SCRIPT

- All'interno del tag noscript può essere utilizzata la sintassi HTML per visualizzare messaggi:

```
<noscript>
<div align="center">
  <h3><font face="Verdana,Arial,Helvetica,sans-serif">
    Per visualzzare correttamente il contenuto della
    pagina occorre avere JavaScript abilitato.
  </font></h3>
</div>
</noscript>
```

FILE ESTERNO

- Quando si scrive codice di una certa lunghezza e/o che potrebbe essere ripetuto su più pagine
- Quando si utilizza un libreria Javascript esistente:

```
<script type="text/javascript" src="miofile.js"></script>
```

- In questo caso il tag **script** va sempre inserito nella **head**

COMMENTI

JavaScript ha due tipi di commenti:

tag di apertura	tag di chiusura	descrizione
//	non si chiude	è un commento 'veloce", che deve essere espresso in una sola riga senza andare a capo
/*	*/	si usa per scrivere commenti su più righe

```
<script type="text/javascript">
     // questo è un commento su una sola riga
    /*
    questo è un commento che sta su più righe,  serve
    nel caso in cui ci siano commenti particolarmente
    lunghi
    */
    alert("ciao");
</script>
```

DOCUMENT.WRITE

- Il metodo **write** che si riferisce all'oggetto **document** (la pagina) consente di scrivere all'interno di una pagina HTML usando

```
<body>
 <script type="text/javascript">

  //Visualizza la scritta “Ciao gente“
  document.write(“<h1>Ciao gente</h1>");

 </script>
</body>
```

FINESTRE DI DIALOGO

- L'oggetto window ci fornisce, tre metodi che ci consentono di fornire o di chiedere informazioni all'utente utilizzando delle finestre di dialogo:

Metodo	Spiegazione	Esempio
alert	Presenta un messaggio all'utente e mostra il bottone **Ok**	window.alert('messaggio');
confirm	Richiede una conferma all'utente. Mostra i bottoni **Ok** e **Annulla**	var risposta; risposta = window.confirm('Vuoi continuare?');
prompt	Richiede all'utente di inserire un valore. Mostra un campo di testo e il bottone Ok	var nome; nome = window.prompt('Come ti chiami?', 'Inserisci qui il tuo nome');

GLI ELEMENTI DEL LINGUAGGIO

INTRODUZIONE

- **Costante**: quantità nota a priori che non dipende dall'input dell'utente e non cambia durante l'esecuzione del programma.
- **Variabile**: nome simbolico a cui è associato un valore che può dipendere dall'input dell'utente e cambiare durante l'esecuzione del programma.
- **Espressione**: sequenza di variabili, costanti, espressioni collegate tra loro da operatori.

Sintassi

- Ora prenderemo in esame questi elementi in termini grammaticali.
- Quello che diremo di JavaScript è sostanzialmente applicabile (salva variazioni di grammatica appunto) a tutti i linguaggi.

Elementi di un linguaggio

- Le unità semantiche di base di ogni linguaggio sono:
  - *Parole chiave*
  - *Operatori e separatori*
  - *Letterali* (o *Costanti*)
  - *Nomi* (o *Identificatori*)

Parole chiave

- Le parole chiave sono i termini (composti da caratteri alfanumerici), riservati al linguaggio di programmazione.
- Il creatore del linguaggio di programmazione stabilisce a priori quali termini riservare e quale sarà la loro funzione, il compito del programmatore è quello di impararle ed usarle in maniera appropriata.
- L'uso improprio di tali termini viene generalmente rilevato durante la fase di compilazione di un programma.

Parole chiave in JavaScript

```
abstract boolean break byte case catch char
class const continue debugger default delete
do double else enum export extends false
final finally float for function goto if
implements import in instanceof int
interface long native new null package
private protected public return short static
super switch synchronized this throw throws
transient true try typeof var void while with
```

Parole chiave in JAVA

abstract, assert, boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, enum, extends, final, finally, float, for, goto, if, implements, import, instanceof, int, interface, long, native, new, package, private, protected, public, return, short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while

Operatori

- Gli operatori sono token composti di uno o più caratteri speciali che servono a controllare il flusso delle operazioni che dobbiamo eseguire o a costruire ***espressioni***
- Operatori usati sia in *JavaScript* che in *JAVA*:

```
++  !   !=  !==  %   %=   &   &&   &=  (  )  -   *   *=
,   .   ?:  /   //  /*   /=  []  ^   ^=  {}  |   ||  |=
~   +   +=  <   <<  <<=  <=  <>  =   -=  ==
===  >   >=  >>  >>=  >>>  >>>=
```

Proprietà degli operatori

- **Precedenza (o Priorità)**
  Indica l'ordine con il quale verranno eseguite le operazioni. Ad esempio in **4+7*5** verrà prima eseguita la moltiplicazione poi l'addizione.
- **Associtività**
  Un operatore può essere associativo a **sinistra** oppure associativo a **destra**. Indica quale operazione viene fatta prima a parità di priorità.

Separatori

- I separatori sono simboli di interpunzione che permettono di chiudere un'istruzione o di raggruppare degli elementi.
- Il separatore principale è lo *spazio* che separa i *termini* tra di loro quando non ci sono altri separatori. Gli altri separatori sono:

```
( ) { } , ;
```

Letterali (o costanti)

- Le *costanti* (o letterali) sono quantità note a priori il cui valore non dipende dai dati d'ingresso e non cambia durante l' esecuzione del programma.
- La sintassi con cui le costanti sono descritte dipende dal tipo di dati che rappresentano.
- Le costanti servono:
  - A dare un valore iniziale ad una variabile
  - A confrontare un valore variable con un valore di riferimento

Costanti numeriche

- Le ***costanti numeriche*** iniziano sempre con un carattere numerico: il fatto che un *token* inizi con un numero basterà ad indicare al compilatore che si tratta di una costante numerica. Se il compilatore non potrà valutare quel *token* come numero segnalerà un errore.
- Il segno che separa la parte intera di un numero dalla parte decimale è il punto.
- È possibile inserire numeri in formato decimale, binario, ottale o esadecimale.
- Per segnalare al compilatore che un numero non è decimale si fa precedere il numero da un prefisso. Per i numeri esadecimali questo prefisso è 0x.
- Gli altri *termini* (*parole chiave* e *nomi*) NON possono iniziare con un numero.

Esempi di costanti numeriche

```
1
2433
1000000000
3.14
.33333333333
0.5
2345.675
0xFF0088
0x5500ff
0xff.00aa
```

Costanti stringa

- Una stringa è una sequenza di caratteri che permette di rappresentare testi. Un *costante* stringa è una sequenza (anche vuota) di caratteri racchiusi tra apici singoli o apici doppi.
- Per inserire ritorni a capo, tabulazioni, particolari caratteri o informazioni di formattazione si utilizzano speciali sequenze di caratteri dette *sequenze di escape*. Una sequenza di escape è formata da un carattere preceduto dal simbolo '\" (*backslash*). La sequenza di escape inserisce un carattere che non sarebbe altrimenti rappresentabile in un letterale stringa.

Principali sequenze di escape

\n nuova riga;

\r ritorno a capo;

\t tabulazione orizzontale;

\' apostrofo (o apice singolo);

\" doppio apice;

**** backslash(essendo un carattere speciale deve essere inserito con una sequenza di escape).

Esempi di costanti stringa

```
// Stringa racchiusa da apici singoli
'Ciao a tutti'
// Stringa racchiusa tra apici doppi
"Ciao"
/* La sequenza di escape risolve l'ambiguità tra l'apostrofo
inserito nella stringa e gli apici singoli che la
racchiudono */
'Questo è l\'esempio corretto'
/* In questo caso non c'è ambiguità perché la stringa è
racchiusa tra doppi apici */
"Anche questo è l'esempio corretto"
/* Per inserire un ritorno a capo si usano le sequenze
di escape */
"Questa è una stringa valida\rdi due righe"
```

Costanti booleane

- Le costanti booleane, poiché rappresentano valori logici, possono avere solo due valori: vero (rappresentato dal letterale ***true***) e falso (rappresentato dal letterale ***false***).

Costanti di tipo Array

- Il letterale ***Array*** è costituito da una serie di elementi separati da virgole compresa tra due parentesi quadre:

```
// array che contiene i mesi dell'anno
["January", "February", "March", "April"];
```

Costanti di tipo Object

- Il letterale *Object* è invece compreso tra parentesi graffe ed è costituito da una serie di coppie '**chiave:valore**" separate da virgole:

```
//record di una rubrica telefonica in formato Object
{name:"Irving",age:32,phone:"555-1234"};
```

Identificatori (o Nomi)

- Un identificatore è un nome definito dal programmatore. Gli identificatori si usano per dare nomi alle variabili e alle funzioni.

Regole per gli Identificatori

- il primo carattere deve essere una lettera o il simbolo '_" (ricordiamo che nel caso la prima lettera fosse un numero il compilatore tenterebbe di interpretare il nome come costante numerica);
- i caratteri successivi possono essere lettere, numeri o '_".
- Gli identificatori non possono inoltre coincidere con le parole riservate del linguaggio.

VARIABILI

Pensiamo a quando salviamo un numero di telefono del nostro amico Mario sul cellulare; se vogliamo chiamare il nostro amico, basterà inserire il suo nome (Mario, nome della **variabile**) ed il cellulare comporrà automaticamente il numero di telefono (**valore** della variabile). Se per qualche ragione Mario cambierà numero di telefono, modificherò il contenuto della mia rubrica (cambierò il valore della variabile). In questa maniera senza modificare le mie abitudini (inserirò sempre Mario) il mio cellulare comporrà il nuovo numero.

VARIABILI

- Una variabile è composta da due elementi: il suo **nome** e il suo **valore**; come ho visto nell'esempio del cellulare in un programma posso usare i nomi delle variabili al posto dei valori che rappresentano.
- Ho la possibilità di usare simboli mnemonici al posto di numeri e stringhe di grande entità o difficili da ricordare.
- Ho la possibilità di usare il nome della variabile al posto del suo valore per eseguirvi sopra delle operazioni, e generalizzare l'elaborazione.

VARIABILI

- Prima di usare una variabile la dichiaro usando l'istruzione **var.**
- Per assegnare alla variabile un valore utilizzo l'operatore di assegnazione ('=').

```
// creo una variabile che si chiama "mioNome"
var mioNome;

//assegno a mioNome il contenuto “Pippo"
mioNome="Pippo";
```

TIPI IN JAVASCRIPT

Tipo di dati	Spiegazione	Esempio
Number	Qualsiasi valore numerico	miaVariabile=300;
Number	Numeri con virgola	miaVariabile=12.5;
String	Qualsiasi valore letterale. È una sequenza di caratteri, racchiusa tra virgolette.	miaVariabile="Wolfgang";
Null	È uno speciale tipo di dato che indica l'assenza di alcun valore ('è il nulla"). Non è lo zero.	miaVariabile=null;

Boolean	È uno tipo di dato che indica uno stato. Di fatto un valore booleano può assumere solo due valori: acceso (vero), spento (falso). È il classico 'interruttore della luce".	//Vero:
		miaVariabile=true;
		
		//Falso:
		miaVariabile=false;

Object	Array (Elenco di valor)i	miaVariabile=['lunedì', 'martedì', 'mercoledì', 'giovedì', 'venerdì', 'sabato', 'domenica']
Object	Informazione complessa	miaVariabile = {nome:"Mario", cognome:"Rossi", eta:25}

EVENTI

evento	si applica a...	esempio
onload	<body>, <img>	<body onload="alert('ciao');">
onunload	<body>	<body onunload="alert('ciao');'>
onmouseover	<a>, <area>, <input> (submit, button, ecc.)	<a onmouseover="alert('ciao');" href="pagina.html">
onmouseout	<a>, <area>, <input>	<a onmouseout="alert('ciao');" href="pagina.html">
onclick	<a>, <area>, <input>	<a onclick="alert('ciao');" href="pagina.html">
onkeypress	<a>, <area>, <input>,	<textarea onkeypress="alert('ciao');'></textarea>
onchange	<select>	<select onchange="alert('ciao');'> <option>uno </option> ..... </select>
onsubmit	<form>	<form name="mioform" action="http://...." onsubmit="alert('ciao');">
onfocus	<a>, <input>, <body>	<body onfocus="alert('ciao');'>
onblur	<a>, <input>, <body>	<body onblur="alert('ciao');">

ISTRUZIONI

- Definire o inizializzare una variabile
- Assegnare un valore a una proprietà o variabile
- Modificare il valore di una proprietà o variabile
- Invocare il metodo di un oggetto
- Richiamare una routine di funzione
- Prendere una decisione

DEFINIRE UNA VARIABILE

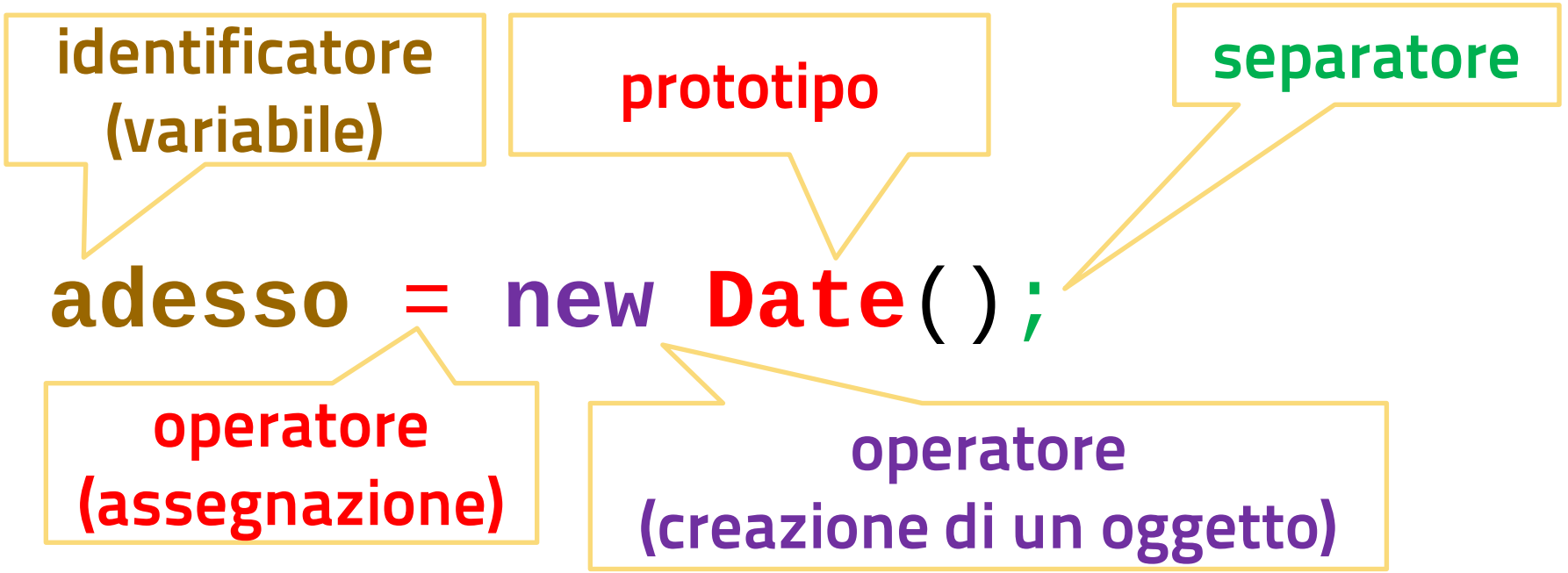
parola chiave (direttiva)

separatore

```
var adesso;
```

Identificatore (variabile)

ASSEGNARE UN VALORE



RICHIAMARE UN METODO

oggetto predefinito

parametro

oggetto date

```
document.getElementById("oggi_data").innerHTML = adesso.getDate();
```

metodo che restituisce un oggetto

proprietà dell'oggetto restituito

metodo che restituisce un valore

FUNZIONI E METODI

COSA È UNA FUNZIONE

- Una funzione (o metodo) è una costrutto presente in tutti i linguaggi di programmazione che consente di associare un gruppo di comandi ad un identificatore.
- Quando nel programma scriverò l'identificatore saranno eseguiti tutti i comandi che compongona la funzione

UTILITÀ DELLE FUNZIONI

- L'uso di funzioni ha due vantaggi:
  - evitare di scrivere codice ripetitivo
  - rendere il mio programma modulare facilitando così modifiche e correzioni.

IN JAVASCRIPT

- Le *funzioni* sono blocchi di codice *JavaScript* riutilizzabili in qualsiasi punto della pagina in cui sono inserite.
- I *metodi* sono semplicemente funzioni che sono associati a un oggetto.

DEFINIZIONE

- Una funzione deve essere **dichiarata** e **definita**;
  - cioè vanno specificati il nome e il numero di parametri che verranno utilizzati nel corpo della funzione
  - e successivamente dovremo scrivere il **corpo** della funzione vera e propria.
  - all'interno del corpo della funzione potrò definire un **valore di ritorno**.

ESEMPIO

```
function somma(n1, n2) {
   return (n1 + n2);
}
```

- Questo codice dichiara la funzione somma che accetta due parametri che devono essere numeri e restituisce un numero.
- La funzione viene poi definita dal blocco di codice tra le due parentesi graffe. Il comando fa che la funzioni ritorni la somma dei due numeri passati come parametri. Se scrivo:

```
var a;
a = somma(5, 7);
```

a conterrà 12.

FUNZIONI INCORPORATE

- Nel linguaggio *JavaScript* sono incorporate numerose funzioni e metodi che consentono di eseguire determinate attività e di accedere alle informazioni.
- Nella maggior parte dei casi si tratta di metodi incorporati in oggetti predefiniti.
- Le funzioni appartenenti a un oggetto sono denominate *metodi*.

SCRITTURA DI FUNZIONI CON NOME

```
function numefunzione (parametro1, parametro2, ….) {
   // Blocco di istruzioni
}
```

- nomefunzione è il nome univoco della funzione. Tutti i nomi di funzione in un documento devono essere univoci.
- parametro1, parametro2, … uno o più parametri che vengono passati alla funzione. I parametri sono detti anche *argomenti*.
- Blocco di istruzioni contiene tutto il codice *JavascriptScript* relativo alla funzione. Questa parte contiene le istruzioni che eseguono le azioni, ovvero il codice che si desidera eseguire. Il commento *// Blocco di* istruzioni è un segnaposto che indica dove deve essere inserito il blocco della funzione.

SCRITTURA DI FUNZIONI ANONIME

```
var nomevariabile = function (parametro1, parametro2, ….)
    {
    // Blocco di istruzioni
}
```

- nomevaribile è il nome di una variabile.
- parametro1, parametro2, … uno o più parametri che vengono passati alla funzione. I parametri sono detti anche *argomenti*.
- Blocco di istruzioni contiene tutto il codice *ActionScript* relativo alla funzione. Questa parte contiene le istruzioni che eseguono le azioni, ovvero il codice che si desidera eseguire.

PASSAGGIO DI PARAMETRI

- Si possono passare più parametri ad una funzione separandoli con delle virgole.
- Talvolta i parametri sono obbligatori e talvolta sono facoltativi. In una funzione potrebbero essere presenti sia parametri obbligatori che opzionali.
- In ogni caso se si passa alla funzione un numero di parametri inferiore a quelli dichiarati, questi conterranno il valore convenzionale *undefined*. Questo può provocare risultati imprevisti.

RESTITUZIONE DI VALORI

- Una funzione può restituire un valore che di norma è il risultato dell'operazione compiuta. Per compiere questa operazione si utilizza l'istruzione *return* che specifica il valore che verrà restituito dalla funzione.
- L'istruzione return ha anche l'effetto di interrompere immediatamente il codice in esecuzione nel corpo della funzione e restituire immediatamente il controllo del flusso di programma al codice chiamante.

JAVASCRIPT

- Javascript serve per programmare il browser.
- Lo studio di Javascript è strettamente legato allo studio del Document Object Model (DOM)

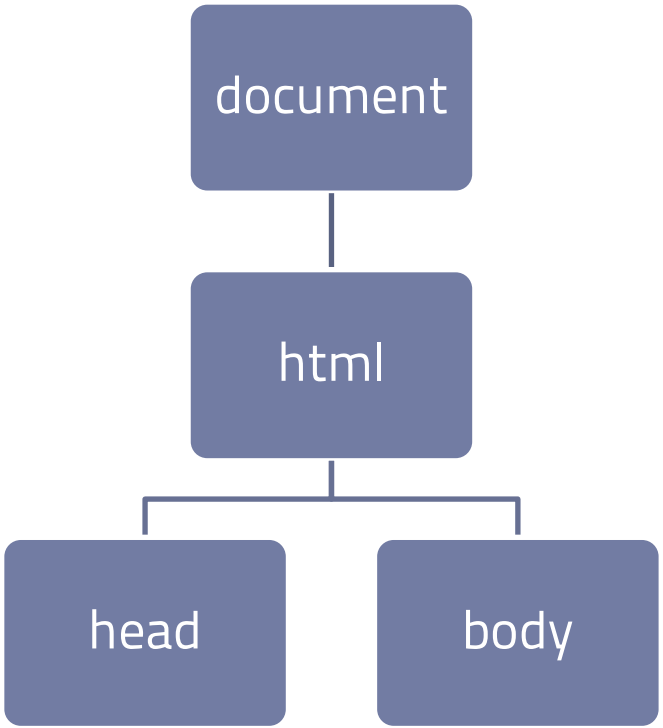
DOCUMENT OBJECT MODEL

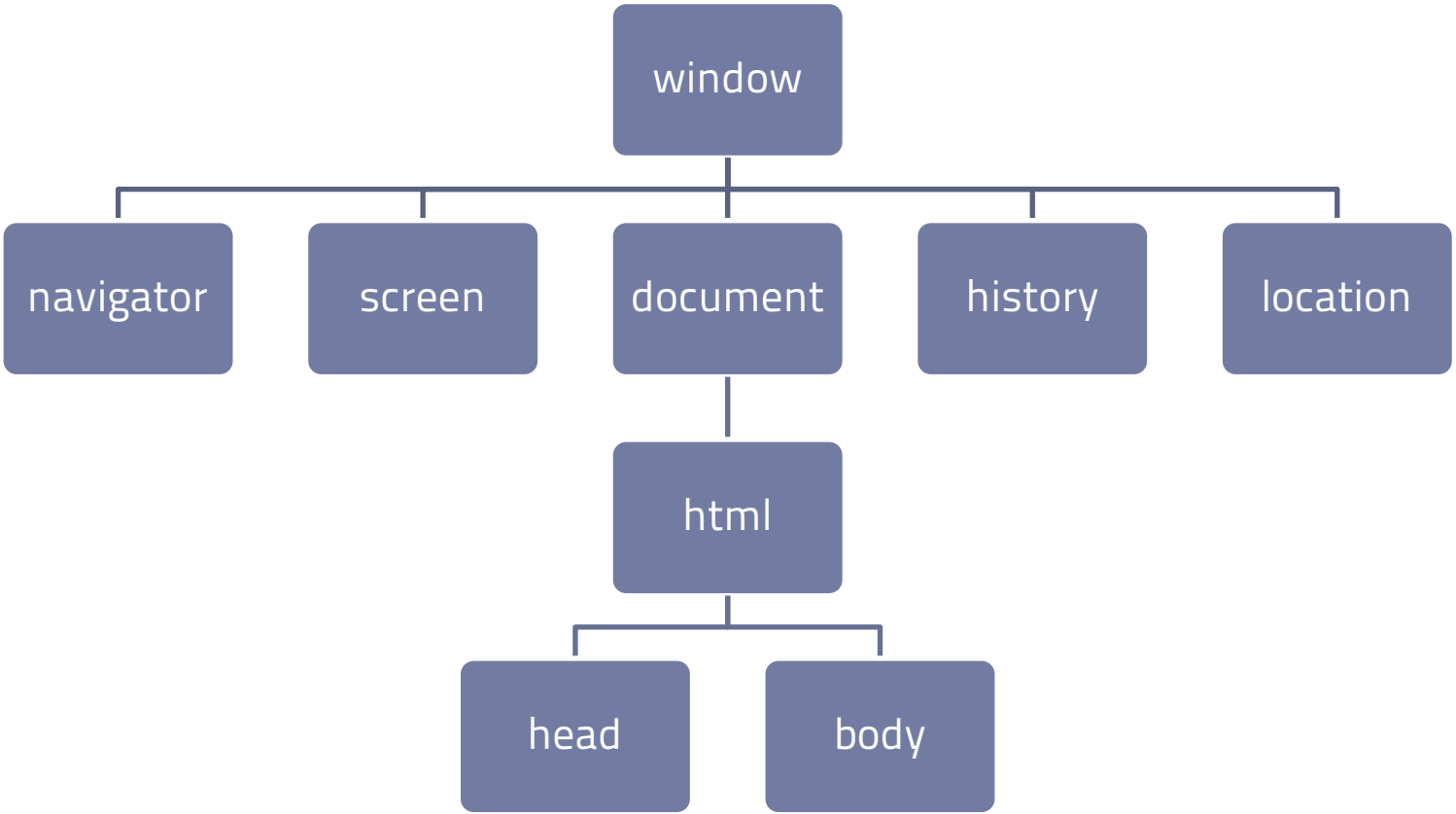
DOM

- HTML (e XHTML) hanno la funzione di strutturare in una rigida gerarchia i contenuti di una pagina WEB
- Quando i browser moderni caricano il contenuto di una pagina organizzano quindi questi contenuti in memoria in una struttura gerarchica ben definita
- Questa struttura gerarchica è il Document Object Model.
- Javascript consente di intervenire su questa struttura aggiungendo, togliendo o modificando gli elementi di cui è composta.

STRUTTURA MINIMA DI UNA PAGINA HTML

```
<html>
  <head></head>
  <body></body>
</html>
```




window
navigator
screen
document
history
location
html
head
body

window

- L'oggetto window è al vertice della gerarchia degli oggetti.
- Rappresenta il la finestra del browser in cui appaiono i documenti HTML. In un ambiente multiframe, anche ogni frame è un oggetto window.
- Dato che ogni azione sul documento si svolge all'interno della finestra, la finestra è il contenitore più esterno della gerarchia di oggetti. I suoi confini fisici contengono il documento.

NAVIGATOR

- L'oggetto navigator rappresente il browser.
- Utilizzando questo oggetto gli script posso accedere alle informazioni sul browser che sta eseguendo il vostro script (marca, versione sistemo operativo).
- E' un oggetto a sola lettura, e il suo uso è limitato per ragioni di sicurezza.

SCREEN

- L'oggetto screen rappresente lo schermo del computer su cui il browser è in esecuzione.
- E' un oggetto a sola lettura che consente allo script conoscere l'ambiente fisico in cui il browser è in esecuzione.
- Ad esempio, questo oggetto fornisce informazioni sulla risoluzione del monitor.

HISTORY

- L'oggetto history rappresenta l'oggetto che in memoria tiene traccia della navigazione e presiede al funzionamento dei bottoni back e forward e alla cronologia del browser.
- Per ragioni di sicurezza e di privacy gli script non hanno accesso a informazioni dettagliate sulla history e l'oggetto di fatto consente solo di simulare i bottoni back e forward.

LOCATION

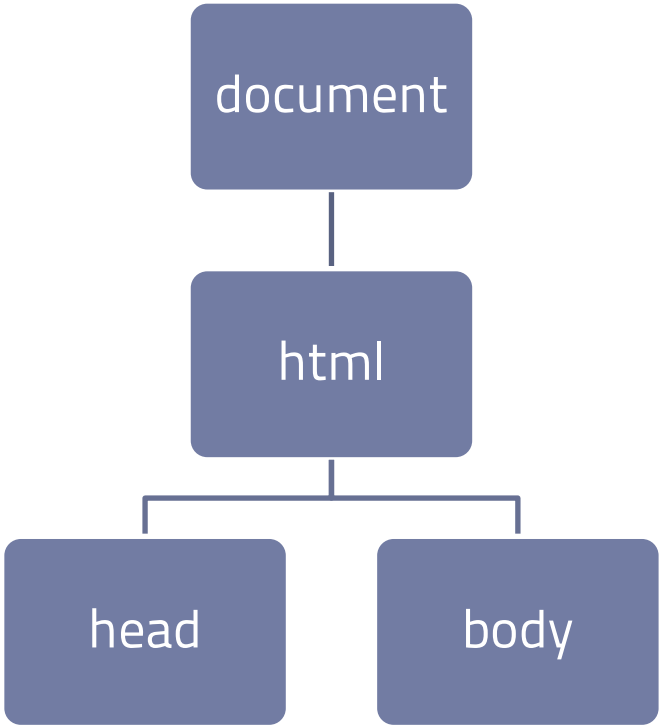
- L'oggetto location rappresenta l'url da cui è stata caricata la pagina
- La sua funzione principale è quella di caricare una pagina diversa nella corrente finestra o frame.
- Allo script è consentito di accedere ad informazioni solo sulla url da cui è stato caricato.

DOCUMENT

- Ogni documento HTML che viene caricato in una finestra diventa un oggetto document.
- L'oggetto document contiene il contenuto strutturato della pagina web.
- Tranne che per gli html, head e body, oggetti che si trovano in ogni documento HTML, la precisa struttura gerarchica dell'oggetto document dipende dal contenuto del documento.

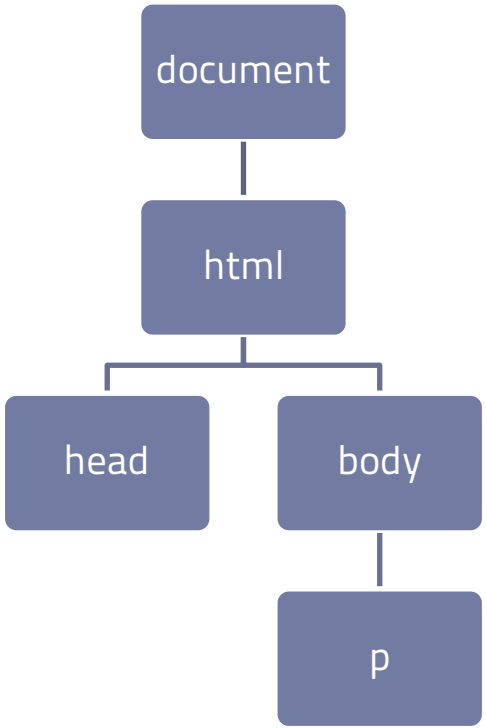
DOCUMENTO VUOTO

```
<html>
  <head></head>
  <body></body>
</html>
```



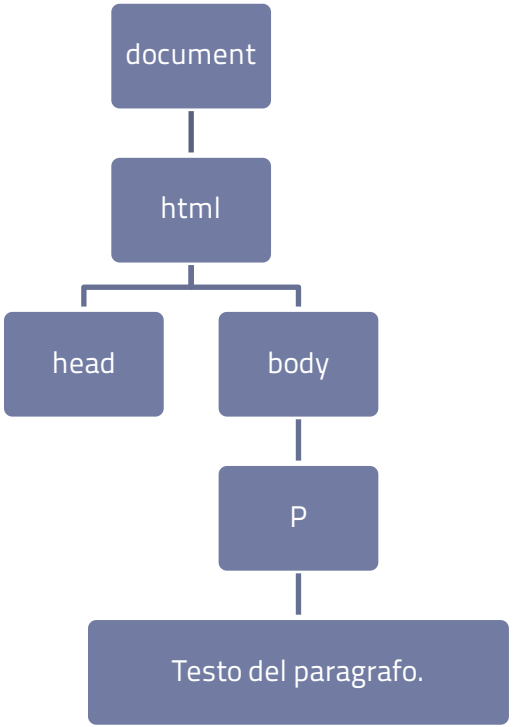
AGGIUNTA DI UN PARAGRAFO VUOTO

```
<html>
  <head></head>
  <body>
     <p></p>
  </body>
</html>
```



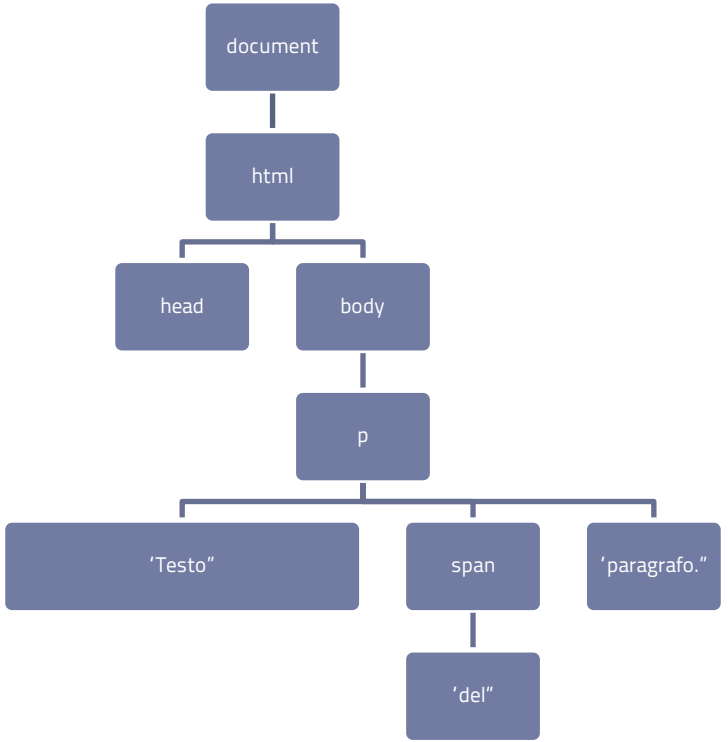
AGGIUNTA DI TESTO AL PARAGRAFO

```
<html>
  <head></head>
  <body>
     <p>Testo del
     paragrafo.</p>
  </body>
</html>
```



AGGIUNTA DI UN ELEMENTO

```
<html>
  <head></head>
  <body>
     <p>Testo
     <span>del</span>
     paragrafo.</p>
  </body>
</html>
```



LA STRUTTURA AD ALBERO

- Dopo che un documento viene caricato nel browser, gli oggetti vengono organizzati in memoria nella struttura gerarchica specificato dal DOM.
- Ogni elemento di questa struttura ad albero viene chiamato **nodo**.
- Ogni nodo può essere:
  - un nuovo ramo dell'albero (cioè avere o non avere altri nodi figli)
  - una foglia (non avere nodi figli)
- Nel DOM avremo:
  - elementi
  - nodi di testo

OBJECT REFERENCE

- Javascript agisce sul DOM modificando, eliminando e aggiungendo oggetti.
- Per agire sul DOM lo script deve interagire con qualcuno dei nodi presenti nella struttura ad albero:
  - Per modificarlo
  - Per aggiungere testo
  - Per aggiungere un figlio ecc.
- Avrà bisogno di un riferimento unico al nodo su cui agire
- Ad ogni nodo posso dare un nome unico utilizzando l'attributo id.
  - <p **id**="primoParagrafo" >
  - <img **id**="logo" src="images/logo.jpg" alt="Logo Azienda">
  - <div class="header" **id**="header">

DARE UN NOME AD UN NODO

- Per poter essere utilizzato da uno script l'ID di un oggetto deve seguire alcune regole:
  - non può contenere spazi
  - non devono contenere segni di punteggiatura tranne che per il carattere di sottolineatura (es.: primo_paragrafo)
  - deve essere racchiuso tra virgolette quando viene assegnato all'attributo id
  - non deve iniziare con un carattere numerico
  - Deve essere unico all'interno dello stesso documento

OBJECT

- Object è una grandezza informatica in grado di rappresentare elementi complessi.
- In Javascript per rappresentare le grandezze ho
  - Number: Numeri
  - String: stringhe di caratteri
  - Boolean: vero e falso
  - Null: Nessun valore
  - Object

ESEMPIO

```
var user:Object = new Object();
user.name = "Irving";
user.age = 32;
user.phone = "555-1234";
```

Viene creato un nuovo oggetto denominato user e tre proprietà: name, age e phone che sono tipi di dati String e Numeric.
Lo stesso oggetto può essere creato anche assegnando alla variabile il letterale di tipo *Object* corrispondente.

```
var user:Object;
user = {name:”Irving”,age:32,phone:”555-1234”};
```

Quando si assegna ad una variabile un valore in formato letterale non è necessario richiamare il costruttore della classe con l'operatore *new*. Queato vale sia per *Object* che per *Array.*

PROPRIETÀ E METODI

- Ognuno degli oggetti cha abbiamo visto ha:
- **Proprietà** che ci consentono di leggere o modificare determinate caratteristiche di un elemento
- **Metodi** che ci mettono a disposizione determinate **azioni** che gli oggetti possono compiere

RAPPRESENTAZIONE DEL DOM

- Ogni elemento del DOM è rappresentato come Object
- L'accesso alle proprietà e ai metodi avviene attraverso l'oiperatore di appartenenza (.)
- Se, per esempio, voglio recuperate il riferimento ad un oggetto scrivo:

```
window.document.getElementById('id')
```

EVENTI

- Grazie agli eventi possiamo "impacchettare" il codice scritto attraverso JavaScript e farlo eseguire non appena l'utente esegue una data azione:
  - quando clicca su un bottone di un form possiamo controllare che i dati siano nel formato giusto;
  - quando passa su un determinato link possiamo
  - Quanto è completato il caricamento di una immagine
  - eccetera….

LA LEGGIBILITÀ DEL CODICE

LEGGIBILITÀ

- Scrivere programmi *sensati* e *leggibili* è difficile, ma molto importante
- È essenziale per lavorare in gruppo
- Aiuto il debugging
- Aiuta a riutilizzare il codice e quindi ci risparmia fatica

LEGGIBILITÀ SIGNIFICA:

- Progettare con chiarezza
- Scrivere codice con chiarezza

PROGETTARE CON CHIAREZZA

- Dedicare il tempo necessario alla progettazione della nostra applicazione non è tempo perso.
- Ci aiuterà a chiarire la logica e la sintassi del nostro lavoro.
- Più avremo sviluppato l'algoritmo che sta alla base della nostra applicazione più il nostro programma sarà comprensibile

SCRIVERE CON CHIAREZZA

- La chiarezza della scrittura si ottiene attraverso due *tecniche :*
- L'***indentazione:*** inserire spazi o tabulazioni per mettere subito in evidenza le gerarchie sintattiche del codice.
- I ***commenti***: inserire note e spiegazione nel corpo del codice.

Identazione: un esempio

- Prendiamo in esamo questo brano di codice HTML *:*

```
<table> <tr> <td>a</td> <td>b</td> <td>c</td>
</tr> <tr> <td> <table> <tr> <td>a1</td> </tr>
<tr> <td>a2</td> </tr> </table> </td> <td>b1</td>

<td>c1</td> </tr> </table>
```

Identazione: un esempio

- E confrontiamolo con questo:

```
<table>
      <tr>
        <td>a</td>
        <td>b</td>
        <td>c</td>
      </tr>
      <tr>
        <td>
        <table>
          <tr>
            <td>a1</td>
          </tr>
          <tr>
            <td>a2</td>
          </tr>
        </table>
        </td>
        <td>b1</td>
        <td>c1</td>
      </tr>
</table>
```

IDENTAZIONE

- Si tratta delle stessa tabella, ma nel primo caso ci risulta molto difficile capire come è organizzata. Nel secondo la gerarchia degli elementi risulta molto più chiara.

Identazione

- L'identazione non ha nessun effetto sulla compilazione del programma
- Serve solo a rendere il nostro lavoro più leggibile.

Inserire commenti

- Rende il codice leggibile anche ad altri
- Quando decidiamo di apportare modifiche a cose che abbiamo scritto ci rende la vita più facile.

Delimitatori

- Delimitatori di riga: tutto cio che segue il contrassegno di commento fino alla fine della riga non viene compilato. Esempi:

  //

- Delimitatori di inizio e fine: tutto ciò compreso tra il contrassegno di inizio e il contrassegno di fine non viene compilato.

  /* .... */ **<!-- .... -->**