LEZIONE 6

JAVASCRIPT

COSA È JAVASCRIPT

- JavaScript è un linguaggio di programmazione.

PROGRAMMA È

- Una serie di istruzioni che il computer è in grado di eseguire
- Che elaborano DATI (**INPUT**)
- Per risolvere un problema implementando un algoritmo
- E ottenere un risultato (**OUTPUT**)

COSA È UN LINGUAGGIO DI PROGRAMMAZIONE

- E' un linguaggio formale dotato di una sintassi ben definita che viene utilizzato per scrivere programmi che realizzano algoritmi.

COSA È UN LINGUAGGIO DI PROGRAMMAZIONE

- Serve a facilitare la programmazione dei calcolatori rendendo possibile descrivere gli algoritmi e le strutture dei dati in una forma più vicina a quella del linguaggio umano scritto.

COSA È UN LINGUAGGIO DI PROGRAMMAZIONE

- A seconda del metodo utilizzato per tradurre il testo delle istruzioni in linguaggio macchina vengono suddivisi in due categorie: compilati e interpretati.

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
- L'interprete utilizzato per eseguirlo è il browser

JAVASCRIPT

1. è in grado di leggere e scrivere gli elementi HTML.
   - Tramite JavaScript è possibile modificare la struttura del documento HTML in tempo reale, senza interagire con il server
2. può essere utilizzato per convalidare i dati inseriti dall'utente prima di inviarli al server.

JAVASCRIPT

3. può essere utilizzato per avere informazioni sul Browser del visitatore.
   - In questo modo possiamo decidere come comportarci a seconda del Browser che sta leggendo la pagina
4. può essere utilizzato per creare i cookie e quindi archiviare e recuperare informazioni sul computer del visitatore

IL TAG SCRIPT

- Il tag **<script>** viene utilizzato per definire uno script lato client, come ad esempio JavaScript.
- Quando l'elemento **<script>** contiene un testo, questo viene interpretato come script da eseguire.
- In alternativa l'attributo **src** consente di collegare alla pagina uno file di script esterno. Quando è presente l'attributo **src** l'elemento **<script>** deve essere vuoto.
- Usi comuni per JavaScript sono la manipolazione delle immagini, la validazione dei form, e cambiamenti dinamici di contenuti.

ATTRIBUTI DI SCRIPT

Attributo	Valore	Descrizione
async	async	Lo scritto è eseguito in modo asincrono (solo per gli script esterni)
charset	*codifica*	Codifica del testo usato nello script esterno
defer	defer	Lo scritto è eseguito quando la pagina è completamente caricata (solo per gli script esterni)
src	*URL*	Url del file di script esterno
type	*media_type*	Esempio: "text/javascript"

SCRIPT INCORPORATO

- Il codice JavaScript va inserito tra l'apertura e la chiusura del tag. Così:

```
<script>
  alert("Ciao da javascript");
</script>
```

- Possiamo inserire il codice JavaScript in qualsiasi parte del documento (nella head oppure nel body) a seconda delle nostre esigenze.

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>
```

GESTIONE DIRETTA EVENTO

- Come abbiamo detto Javascript è fatto principalmente per rispondere a degli eventi, come quello di un utente che clicca un elemento della pagina
- Si può associare direttamente del codice javascript all'evento di un elemento usando appositi attributi come onclick, onload, ecc:

```
<button onclick="alert('Ciao!')">Cliccami !</button>
```

eventi

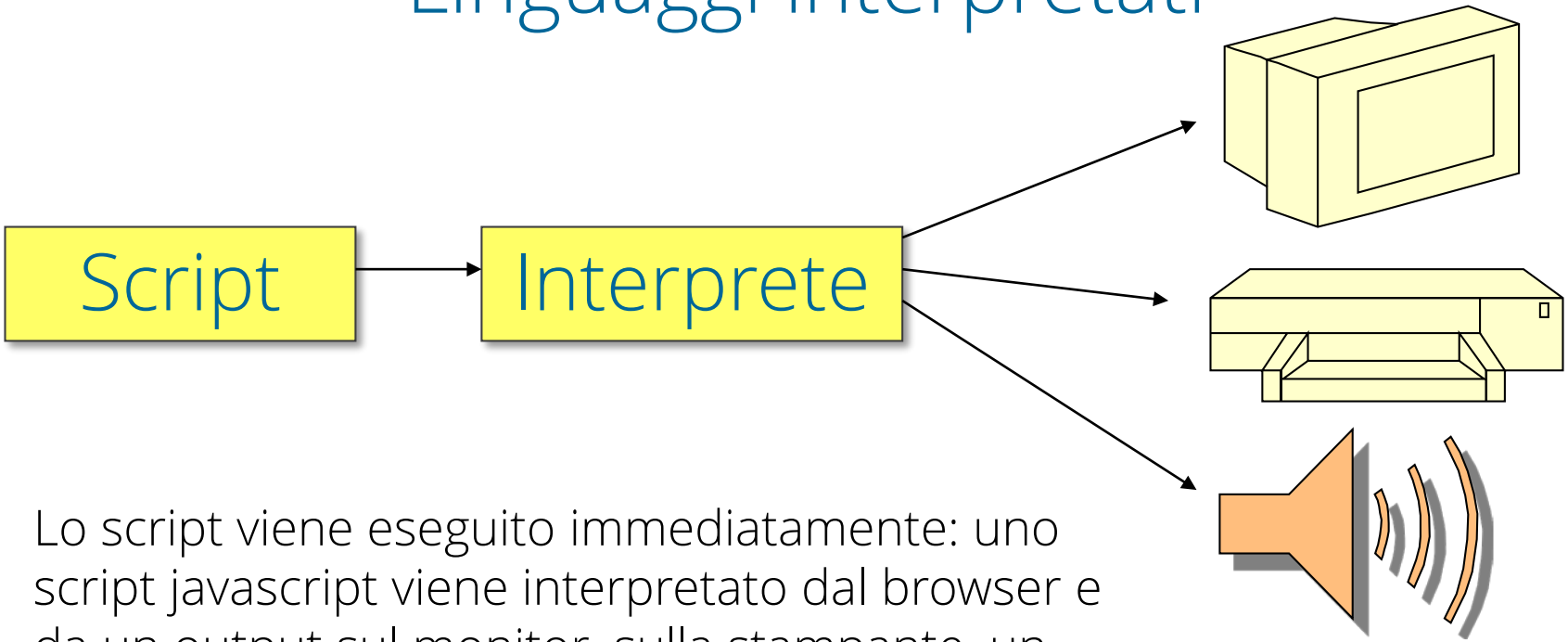
evento	si applica a...	esempio
onload	<body>, <img>	<body onload="alert('ciao');">
onunload	<body>	<body onunload="alert('ciao');“>
onmouseover	<a>, <area>, <input> (submit, button, ecc.)	<a onmouseover="alert('ciao');" href="pagina.html">
onmouseout	<a>, <area>, <input>	<a onmouseout="alert('ciao');" 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');">

No script

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

```
<noscript>
        <div>
                <h3>
                        Per visualzzare correttamente il
                        contenuto della pagina occorre avere
                        JavaScript abilitato.
                </h3>
        </div>
</noscript>
```

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.

COMPILATO <> INTERPRETATO

- compilazione:
  - lo script viene elaborato dal compilatore prima di essere eseguito e la maggior parte degli errori di sintassi vengono individuati
- interpretazione:
  - Lo script viene eseguito così com'è, il controllo della correttezza del codice è affidato direttamente all'esecuzione dello stesso.

COMPILATO <> INTERPRETATO

- compilazione:
  - il programma viene eseguito in uno specifico sistema operativo o in una macchina virtuale in uno scenario tendenzialmente stabile
- iavascript:
  - Viene eseguito nel browser. Browser di diversi produttori possono avere comportamenti leggermente diversi.

GLI ELEMENTI DEL LINGUAGGIO

INTRODUZIONE

- **Istruzione**: parola riservata che il linguaggio usa per i comandi di base del linguaggio
- **Variabile**: nome simbolico a cui è associato un valore che può dipendere dall'input dell'utente e cambiare durante l'esecuzione del programma.
- **Costante**: quantità nota a priori che non dipende dall'input dell'utente e non cambia durante l'esecuzione del programma.
- **Oggetto**: Struttura usata per rappresentare ed elaborare i dati in un linguaggio Object Oriented.
- **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:
  - Istruzioni
  - 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.

COMANDI JAVASCRIPT

Statement	Description
break	Esce da un blocco switch o da un ciclo
continue	Interrompe l'iterazione di un ciclo se una determinate condizione si verifica e contuna con quella successiva
debugger	Interrompe l'esecuzione e lancia il debugger, se disponibile.
do ... while	Esegue un blocco di comandi fino a che una determinata condizione è vera.
for	Esegue un blocco di comandi fino a che una determinata condizione è vera.
for ... in	Esegue un blocco di comandi per ogni elemento presente in un insieme (Array o Object)
function	Dichiara una funzione
if ... else ... else if	Esegue un blocco di comandi quando una condizione è vera
return	Interrompe l'esecuzione di una funzione e ritorna un valore
switch	Organizza una serie di blocchi di istruzioni dioendenti da condizioni alternative.
throw	Genera un errore.
try ... catch ... finally	Gestione degli errori
var	Dichiara una variabile.
while	Esegue un blocco di comandi fino a che una determinata condizione è vera.

Operatori

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

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

OPERATORI ARITMETICI

Operatore	Funzione	Espressione	Valore di y	Valore di x
		Valore iniziale: y=5		
+	Addizione	x = y + 2	y = 5	x = 7
-	Sottrazione	x = y - 2	y = 5	x = 3
*	Moltiplicazione	x = y * 2	y = 5	x = 10
/	Divisione	x = y / 2	y = 5	x = 2.5
%	Resto intero (Modulo)	x = y % 2	y = 5	x = 1
++	Incremento	x = ++y	y = 6	x = 6
		x = y++	y = 6	x = 5
--	Decremento	x = --y	y = 4	x = 4
		x = y--	y = 4	x = 5

OPERATORI DI ASSEGNAZIONE

Operatore	Espressione	Espressione equivalente	Valore di x
	Valori dati: x = 10; y = 5		
=	x = y	x = y	x = 5
+=	x += y	x = x + y	x = 15
-=	x -= y	x = x - y	x = 5
*=	x *= y	x = x * y	x = 50
/=	x /= y	x = x / y	x = 2
%=	x %= y	x = x % y	x = 0

OPERATORI STRINGA

Operatore	Espressione	Espressione equivalente	Risultato
	Valori dati: `text1 = "Buon "; text2 = "giorno"; text3="";`		
=	text1 = text2		text1 'giorno'
=	text3 = text1+text2		text3 'Buon giorno'
+=	text1 += text2	text1 = text1+text2	text1 'Buon giorno'

STRINGHE E NUMERI

Operazione	Espressione	Conversione	Risultato
Somma numero con numero	x = 5 + 5;	Nessuna conversione	10
Somma numero in una stringa con numero	x = '5' + 5;	Converte 5 in '5.	'55'
Somma una stringa con un numero	x = 'Carlo' + 5;	Converte 5 in '5.	'Carlo5'

OPERATORI DI COMPARAZIONE

Operatore	Descrizione	Comparazione	Risultato
		Valore di x: 5 (numero)	
==	Uguale a...	x == 8	false
		x == 5	true
===	Valore e tipo uguali	x === "5"	false
		x === 5	true
!=	Diverso da...	x != 8	true
!==	Valore e tipo diversi	x !== "5"	true
		x !== 5	false
>	Maggiore di...	x > 8	false
<	Minore di...	x < 8	true
>=	Maggiore o uguale a...	x >= 8	false
<=	Minore o uguale a...	x <= 8	true

OPERATORI LOGICI

Operatore	Corrisponde a...	Esempi (x = 3; y = 6;)
&&	And logico	(x < 10 && y > 1) è true
||	Or logico	(x === 5 || y === 5) è false
!	Not logico	!(x === y) è true

OPERATORI SU BIT

Operatore	Descrizione	Esempio	In binario	Risultato	Decimale
&	AND	x = 5 & 1	0101 & 0001	0001	1
|	OR	x = 5 | 1	0101 | 0001	0101	5
~	NOT	x = ~ 5	~0101	1010	10
^	XOR	x = 5 ^ 1	0101 ^ 0001	0100	4
<<	Shift a sin.	x = 5 << 1	0101 << 1	1010	10
>>	Shift a destra	x = 5 >> 1	0101 >> 1	0010	2

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"; 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}

DEFINIRE UNA VARIABILE

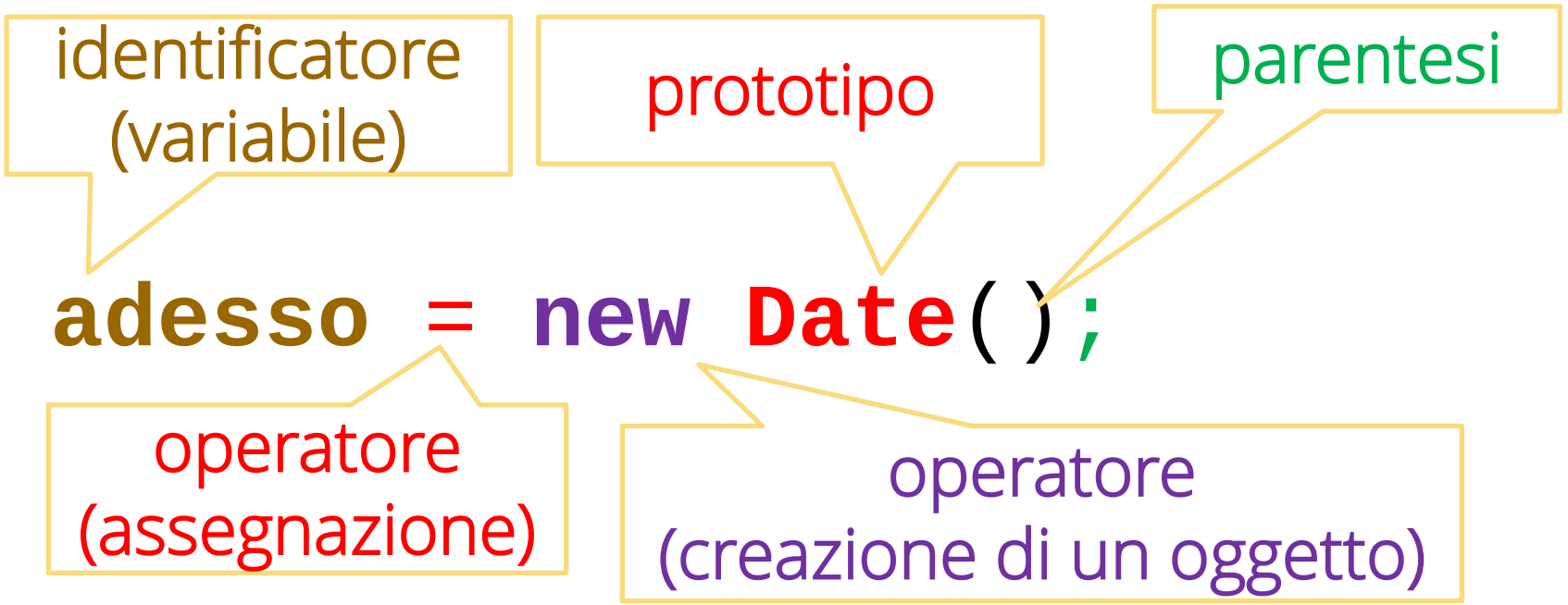
parola chiave (direttiva)

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 GLOBALI

PROPRIETÀ GLOBALI

Propietà	Descrizione
Infinity	Un valore numerico che rappresenta l'infinito positivo e negativo
NaN	Il valore "Not-a-Number"
undefined	Indica che ha una variabile (o a una proprietà) non è stato assegnato alcun valore.

FUNZIONI GLOBALI

Funzione	Descrizione
decodeURI(uri)	Decodifica un URI codificata con encodeURI
decodeURIComponent(uri)	Decodifica un URI codificata con decodeURIComponent
encodeURI(uri)	Codifica un URI (codifica i caratteri speciali eccetto / ? : @ & = + $ #)
encodeURIComponent(uri)	Codifica un URI (codifica i caratteri speciali compresi / ? : @ & = + $ #)
escape(str)	Questa funzione rende una stringa portatile, in modo che possa essere trasmessa attraverso qualsiasi rete a qualsiasi computer che supporti i caratteri ASCII.
eval(str)	Valuta una stringa e la esegue come se fosse il codice di script
isFinite()	Determina se un valore è un numero finito (e legale)
isNaN()	Determina se un valore è non è lo speciale valore NaN
Number()	Converte il valore di un oggetto in un numero
parseFloat()	Analizza una stringa e restituisce un numero in virgola mobile o NaN
parseInt()	Analizza una stringa e restituisce un intero o NaN
String()	Converte il valore di un oggetto in una stringa
unescape()	Decodifica una stringa codificata con escape.

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 1

```
function hello() {
   alert("Ciao gente!");
}
```

- Questo codice dichiara la funzione hello. Non ha parametri e non restituisce valori.
- La funzione viene poi definita dal blocco di codice tra le due parentesi graffe. Il comando usa la funzione **alert** (predefinita) per lanciare un messaggio all'utente. Se scrivo:

  ```
  hello();
  ```

  si aprirà la piccola finestra dei messaggi con scritto ciao gente .

ESEMPIO 2

```
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

- In ogni linguaggio sono incorporate numerose funzioni che consentono di eseguire determinate attività e di accedere alle informazioni.
- *JavaScript* è linguaggio orientato agli oggetti. Tutte le funzioni sono incorporate negli 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)

STRING

CONSTRUCTOR

```
var str = "Ciao!";
var str = new String("Ciao!");
```

PROPRIETÀ

- Gli oggetti della classe String hanno una sola proprietà, la proprietà **length** che restituisce la lunghezza della stringa, cioè il numero di caratteri di cui è composta.

MANIPOLAZIONE

Method	Description
charAt(pos)	Restituisce il carattere alla posizione pos
charCodeAt(pos)	Restituisce il carattere (in formato Unicode) alla posizione pos
concat(s1, s2)	Concatena due stringhe (come s1 + s2)
fromCharCode(code)	Restituisce il carattere corrispondente al valore unicode code
indexOf(searchstring, start)	Restituisce la posizione della prima occorrenza della stringa searchstring in una stringa (-1 se non lo trova). Opzionalmente la ricerca può partire dalla posizione start
lastIndexOf(searchstring, start)	Restituisce la posizione dell'ultima occorrenza della stringa searchstring in una stringa (-1 se non lo trova). Opzionalmente la ricerca può partire dalla posizione start in vece che dall'ultimo carattere.
match(regexp)	Il metodo match cerca le corrispondenza e tra l'espressione regolare regexp e la stringa, e restituisce un array di corrispondenze. Se non viengono trovate corrispondenze viene restituito null.
replace(regexp/substr, newstring)	Replace() cerca una corrispondenza tra una stringa (o un'espressione regolare) e una stringa, e sostituisce la corrispondenze trovate con newstring
search(regexp)	Il metodo search cerca le corrispondenza e tra l'espressione regolare regexp e la stringa, e restituisce la posizione in cui è stata trovata oppure -1 se non viengono trovate corrispondenze.
slice(inizio, fine)	Estrae la parte di una stringa compresa tra inizio e fine e restituisce la parte estratta in una nuova stringa. In caso non sia passato un valore, fine sarà l'ultimo carattere della stringa.
split(char)	Converte la stringa in un array usando char come carattere di separazione.
substr(start, length)	Estrae lentgth caratteri dalla stringa, a partire da start e li restituisce in una nuova stringa. Se length non è specificati vengono restituiti i caratteri da start fino alla fine della stringa.
substring(from, to)	Estrae i caratteri delle stringa tra from e to non compreso. Se to è omesso fino alla fine della stringa.
toLowerCase()	Converte in minuscolo
toUpperCase()	Converte in maiuscolo

ARRAY

CONSTRUCTOR

```
var a = [1,6,78,23];
var a = new Array(1,6,78,23);
```

PROPRIETÀ

- Gli oggetti della classe Array hanno una sola proprietà, la proprietà **length** che restituisce la lunghezza dell'array, cioè il numero di elementi di cui è composto.

METODI

Method	Description
concat(array2,array3, ..., arrayX)	Unisce uno o più array all'array a cui il metodo è applicato, e restituisce una copia degli array così uniti.
indexOf(elemento, start)	Cerca elemento in un array partendo da start (o dall'inizio se start è omesso) e ne restituisce la posizione. Se start è negativo indica la posizione reativa alla fine dell'array.
join(separatore)	Unisce gli elementi di un array in una stringa, e restituisce la stringa. Gli elementi sono separati da separatore. Il separatore di default è la virgola .
lastIndexOf(elemento, start)	Cerca l'ultima ricorrenza di elemento in un array partendo da start (o dall'iniziose start è omesso) e ne restituisce la posizione o -1 se elemento non viene trovato.
pop()	Rimuove l'ultimo elemento di un array, e restituisce l'elemento rimosso.
push(elemento)	Aggiunge elemento alla fine dell'array e restituische la nuova lunghezza.
reverse()	Inverte l'ordine degli elementi dell''array.
shift()	Rimuove il primo elemento di un array, e restituisce l'elemento rimosso.
slice(inizio, fine)	Estrae gli elementi a partire da inizio, fino a fine, non incluso e li restituisce in un nuovo array. L'array originale non viene modificato.
sort(sortfunct)	Ordina gli elementi di un array (alfabetico ascendente) o usa sortfunct per stabilire l'ordine
splice(indice, quanti, item1,....., itemX)	Rimuove quanti elementi dall'array a partire dalla posizine indice e inserisce gli elementi item1, ..., itemX (se forniti) a partire dalla posizine indice. Restituisce gli elementi rimossi.
toString()	Restituisce l'array convertito in stringa.
unshift(elemento)	Aggiunge elemento all'inizio dell'array e restituische la nuova lunghezza

sort

```
var rubrica = [
                {nome:"Mario", cognome:"Rossi" },
                {nome:"Luigi", cognome:"Neri" },
                {nome:"Piero", cognome:"Verdi" },
                {nome:"Mario", cognome:"Bianchi" }
              ];
var sortCognome = function (a,b){
                              if (a.cognome > b.cognome){
                                return 1;
                              } else if (a.cognome == b.cognome){
                                return 0;
                              } else {
                                return 1;
                              }
                          };
rubrica.sort(sortCognome);
```

DATE

CONSTRUCTOR

```
var d = new Date();
var d = new Date(milliseconds);
var d = new Date(dateString);
var d = new Date(year, month, day,
                 hours, minutes,
                 seconds,
                 milliseconds);
```

Metodo statico

Date.parse(str)	Analizza una data in formato stringa e restituisce il numero di millisecondi dalla mezzanotte del 1 Gennaio 1970.

Metodi	Descrizione
getDate()	Restituisce il giorno del mese (1-31)
getDay()	Restituisce il giorno della settimana (0-6, 0 = domenica)
getFullYear()	Restituisce l'anno (quattro cifre)
getHours()	Restituisce l'ora (da 0-23)
getMilliseconds()	Restituisce i millisecondi (0-999)
getMinutes()	Restituisce i minuti (0-59)
getMonth()	Restituisce il mese (0-11)
getSeconds()	Restituisce i secondi (0-59)
getTime()	Restituisce il numero di millisecondi trascorsi dalla mezzanotte del 1 gennaio 1970
getTimezoneOffset()	Restituisce la differenza di tempo tra il GMT e l'ora locale, in pochi minuti
getUTCDate()	Restituisce il giorno del mese, in base all'ora universale (da 1-31)
getUTCDay()	Restituisce il giorno della settimana, in base all'ora universale (da0-6)
getUTCFullYear()	Restituisce l'anno, in base all'ora universale (quattro cifre)
getUTCHours()	Restituisce l'ora, in base all'ora universale (da 0-23)
getUTCMilliseconds()	Restituisce i millisecondi, in base all'ora universale (0-999)
getUTCMinutes()	Restituisce i minuti, in base all'ora universale (da 0-59)
getUTCMonth()	Restituisce il mese, in base all'ora universale (da 0-11)
getUTCSeconds()	Restituisce i secondi, in base all'ora universale (da 0-59)

Metodi	Descrizione
setDate()	Imposta il giorno del mese di un oggetto data
setFullYear()	Imposta l'anno (quattro cifre) di un oggetto data
setHours()	Imposta l'ora di un oggetto data
setMilliseconds()	Imposta i millisecondi di un oggetto data
setMinutes()	Impostare i minuti di un oggetto data
setMonth()	Imposta il mese di un oggetto data
setSeconds()	Imposta i secondi di un oggetto data
setTime()	Consente di impostare una data e un'ora aggiungendo o mezzanotte del primo gennaio 1970
setUTCDate()	Imposta il giorno del mese di un oggetto data, in base
setUTCFullYear()	Imposta l'anno di un oggetto data, in base all'ora
setUTCHours()	Imposta l'ora di un oggetto data, in base all'ora universale
setUTCMillisecond	Imposta i millisecondi di un oggetto data, in base all'ora
setUTCMinutes()	Impostare i minuti di un oggetto data, in base all'ora
setUTCMonth()	Imposta il mese di un oggetto data, in base all'ora
setUTCSeconds()	Impostare i secondi di un oggetto data, in base all'ora
setDate()	Imposta il giorno del mese di un oggetto data
setFullYear()	Imposta l'anno (quattro cifre) di un oggetto data
setHours()	Imposta l'ora di un oggetto data

Metodi	Descrizione
toDateString()	Converte la parte relativa alla data di un oggetto Date in
toISOString()	Restituisce la data come una stringa, utilizzando lo
toJSON()	Restituisce la data come una stringa, formattato come una
toLocaleDateStrin	Restituisce la parte relativa alla data di un oggetto
toLocaleTimeStrin	Restituisce la parte di ora di un oggetto Date come una
toLocaleString()	Converte un oggetto Date in una stringa, utilizzando le
toString()	Converte un oggetto Date in una stringa
toTimeString()	Converte la parte ora di un oggetto Date in una stringa
toUTCString()	Converte un oggetto Date in una stringa, in base all'ora
UTC()	Restituisce il numero di millisecondi in una stringa data a universale

NUMBER

constructor

```
var n = 5;
var n = new Number(5);
var n = 10.6;
var n = new Number(10.6);
```

proprietà statiche

Proprietà	Descrizione
MAX_VALUE	Restituisce il massimo numero consentito in JavaScript
MIN_VALUE	Restituisce il minimo numero consentito in JavaScript
NEGATIVE_INFINITY	Rappresenta l'infinito negativo.
POSITIVE_INFINITY	Rappresenta l'infinito positivo.

metodi

Method	Description
toExponential(x)	Reastituisce una stringa, che rappresenta il numero come notazione esponenziale dove x (opzionale) indica il numero dei decimali da usare
toFixed(x)	Converte il numero in una stringa, con x numero di decimali. Se x non viene specificato nessun decimale.
toPrecision(x)	Converte il numero in una stringa di lunghezza x. Se necessario vengono aggiunti punto decimale e 0.
toString(base)	Converte il numero in una stringa secondo la base specificata da base. La base di default è 10 (numero decimale). Se base vale 2 si ottiene la rappresentazione binaria del numero, se 16 quella esadecimale, ecc.

MATH

Proprietà statiche

Propietà	Descrizione
E	Restituisce il numero di Eulero (circa 2,718)
LN2	Restituisce il logaritmo naturale di 2 (circa 0,693)
LN10	Restituisce il logaritmo naturale di 10 (circa 2,302)
LOG2E	Restituisce il logaritmo in base 2 di E (circa 1,442)
LOG10E	Restituisce il logaritmo in base 10 di E (circa 0,434)
PI	Restituisce PI (circa 3.14)
SQRT1_2	Restituisce la radice quadrata di 1/2 (circa 0,707)
SQRT2	Restituisce la radice quadrata di 2 (circa 1,414)

metodi statici

Method	Description
abs(x)	Restituisce il valore assoluto di x
acos(x)	Restituisce l'arcocoseno di x, in radianti
asin(x)	Restituisce l'arcoseno di x, in radianti
atan(x)	Restituisce l'arcotangente di x come un valore numerico compreso tra-PI / 2 e PI 2 radianti
atan2(y,x)	Restituisce l'arcotangente del quoziente dei suoi argomenti
ceil(x)	Restituisce X arrotondato per eccesso al numero intero più vicino
cos(x)	Restituisce il coseno di x (x è in radianti)
exp(x)	Restituisce il valore di E elevato alla x
floor(x)	Restituisce X arrotondato per difetto al numero intero più vicino
log(x)	Restituisce il logaritmo naturale (base e) di x
max(x,y,z,...,n)	Restituisce il numero con il valore più alto
min(x,y,z,...,n)	Restituisce il numero con il valore più basso
pow(x,y)	Restituisce il valore di x alla potenza y
random()	Restituisce un numero casuale compreso tra 0 e 1
round(x)	Arrotonda x al numero intero più vicino
sin(x)	Restituisce il seno di x (x è in radianti)
sqrt(x)	Restituisce la radice quadrata di x
tan(x)	Restituisce la tangente di un angolo

REGEXP

constructor

```
var re = new RegExp(pattern,
                    mod);
var re = /pattern/modificatori;
var re = /0-9/g;
```

modificatori

Modificato	Descrizione
i	Eseguire case-insensitive di
g	Eseguire una partita globale (trovate prima partita)
m	Effettuare ricerche su righe multiple

parentesi quadre

Espressione	Descrizione
[abc]	Trova qualsiasi carattere tra le parentesi
[^abc]	Trova qualsiasi carattere non tra le parentesi
[0-9]	Trova qualsiasi cifra 0-9
[A-Z]	Trova un carattere tra A maiuscola e Z maiuscola
[a-z]	Trova un carattere tra a minuscola a z minuscola
[A-z]	Trova un carattere da maiuscolo a minuscolo A z
[adgk]	Trova qualsiasi carattere nell'elenco
[^adgk]	Trova un carattere non compreso nell'elenco
(red|blue|green)	Trova una delle alternative indicate

metacaratteri

Metacarattere	Descrizione
..	Trova un singolo carattere, eccetto newline o terminatore di linea
\w	Trova un carattere alfanumerico
\W	Trova un carattere non alfanumerico
\d	Trova una cifra
\D	Trova un carattere non numerico
\s	Trova uno spazio bianco
\S	Trova un carattere non-spazio
\b	Trova un match ad inizio / fine di una parola
\B	Trovare non è una partita ad inizio / fine di una parola
\0	Trova un carattere NUL
\n	Trova un carattere di nuova riga
\f	Trova un carattere di avanzamento modulo
\r	Trova un carattere di ritorno
\t	Trova un carattere di tabulazione
\v	Trova un carattere di tabulazione verticale
\xdd	Trova il carattere specificato da un numero esadecimale dd
\uxxxx	Trova il carattere Unicode specificato da un numero esadecimale xxxx

Quantificatori

Quantificat	Descrizione
n+	Corrisponde a qualsiasi stringa che contiene almeno un n
n*	Corrisponde a qualsiasi stringa che contiene zero o più
n?	Corrisponde a qualsiasi stringa che contiene zero o una
n {X}	Corrisponde a qualsiasi stringa che contiene una sequenza
n {X, Y}	Corrisponde a qualsiasi stringa che contiene una sequenza
n {X,}	Corrisponde a qualsiasi stringa che contiene una sequenza
n$	Corrisponde a qualsiasi stringa con n alla fine.
^n	Corrisponde a qualsiasi stringa con n all'inizio.
?=n	Corrisponde a qualsiasi stringa che viene seguita dalla
?!n	Corrisponde a qualsiasi stringa che non è seguita dalla

prorietà e metodi

Proprietà	Descrizione
global	Specifica se il modificatore "g" è impostato
ignoreCase	Specifica se il modificatore "i" è impostato
lastIndex	L'indice da cui iniziare la prossima ricerca
multiline	Specifica se il modificatore "m" è impostato
source	Il testo del pattern RegExp

Metodo	Descrizione
compile()	Compila un espressione regolare
exec ()	Cerca la prima occorrenza e la restituisce
test ()	Cerca la prima occorrenza . Restituisce vero o falso