FUNZIONI E METODI

COSA È UNA FUNZIONE

- Una funzione (o procedura 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 ACTION SCRIPT

- Le ***funzioni*** sono blocchi di codice ***ActionScript*** riutilizzabili in qualsiasi punto di un file SWF
- I ***metodi*** sono semplicemente funzioni che si trovano all'interno di una definizione di **classe** *ActionScript*.

DICHIARAZIONE E DEFINIZIONE

- Una funzione deve essere **dichiarata** e **definita**;
  - cioè vanno specificati i tipi di ingresso e di uscita sui quali la funzione andrà a compiere le proprie operazioni (**DICHIARAZIONE**)
  - e successivamente dovremo scrivere il **corpo** della funzione vera e propria (**DEFINIZIONE**).
  - all'interno del corpo della funzione potrò definire un **valore di ritorno**.

ESEMPIO

```
function somma(n1:Number, n2:Number):Number{
   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:Number;
a = somma(5, 7);
```

  a conterrà 12.

FUNZIONI INCORPORATE

- Nel linguaggio *ActionScript* sono incorporate numerose funzioni che consentono di eseguire determinate attività e di accedere alle informazioni.
- Si può trattare di funzioni globali o di funzioni appartenenti ad una classe incorporata nel linguaggio.
- Si può, ad esempio, ottenere il tempo passato da quando un file SWF è stato lanciato utilizzando getTimer() o il numero di versione di Flash Player in cui è caricato il file utilizzando getVersion().
- Le funzioni appartenenti a un oggetto sono denominate ***metodi***. Quelle che non appartengono a un oggetto sono denominate ***funzioni di primo livello***.

ESEMPIO

- Le funzioni di primo livello sono di facile utilizzo. Per chiamare una funzione, è sufficiente utilizzarne il nome e passare tutti i parametri richiesti. Se, ad esempio, aggiungo il codice *ActionScript* seguente al fotogramma 1 della linea temporale:

```
trace("Hello world!");
```

- Quando si prova il file SWF, verrà visualizzato Hello world! nel pannello Output. La funzione ***trace***, infatti non fa altro che scrivere un messaggio sulla finestra di output e non ritorna alcun valore.

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 *ActionScript* 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, Flash imposta i valori dei parametri mancanti a ***undefined***. Questo può provocare risultati imprevisti.

ESEMPIO

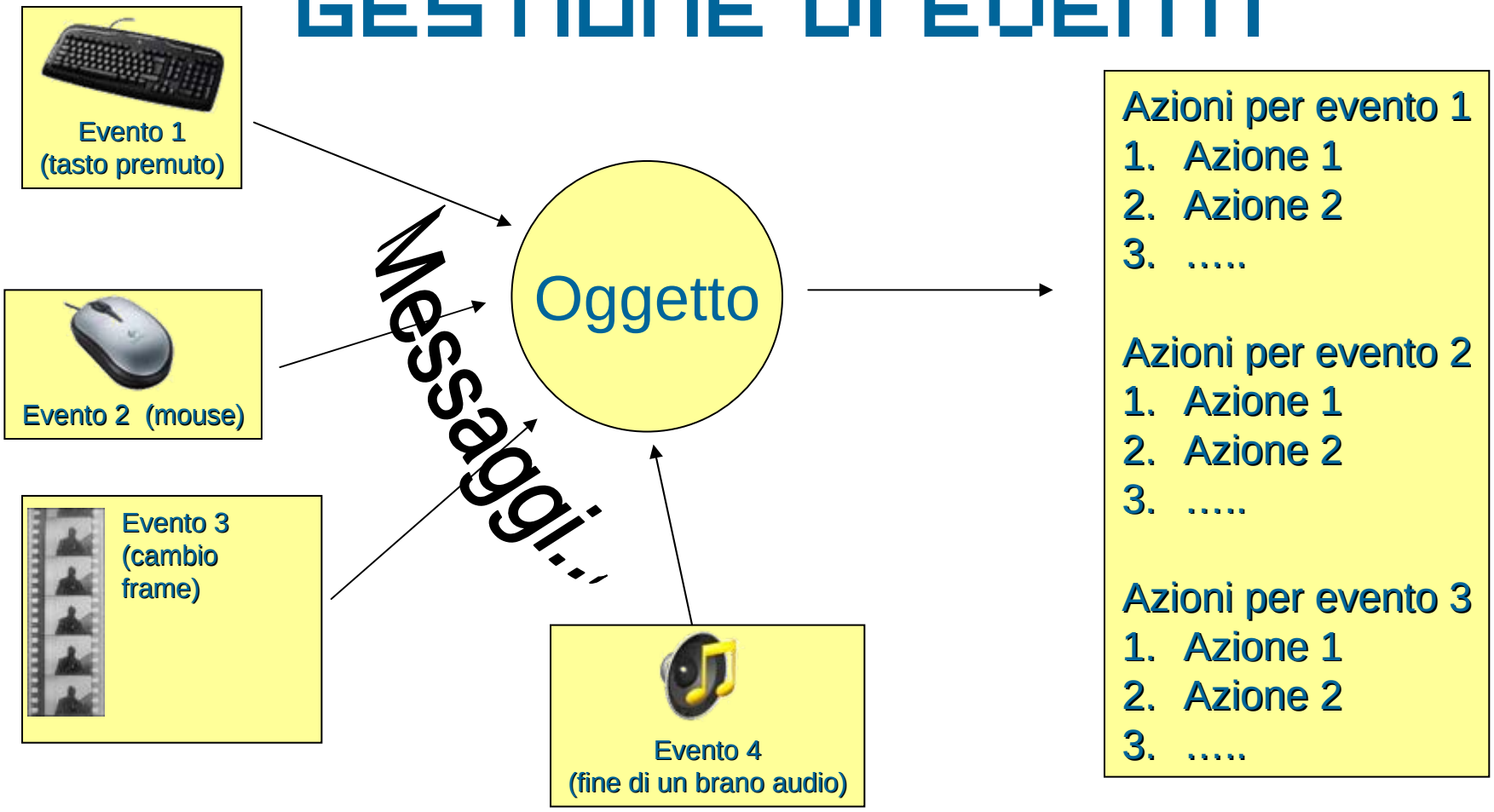
```
function somma(a:Number, b:Number, c:Number):Number {
     return (a + b + c);
}
// sommo tre numeri
trace(somma(1, 4, 6)); // 11
// La somma non è un numero (c è uguale a undefined)
trace(somma(1, 4)); // NaN
// il parametro non dichiarato è ignorato
trace(somma(1, 4, 6, 8)); // 11
```

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.
- Nell'utilizzo dell'istruzione *return* si applicano le regole seguenti:
  - Se per una funzione si specifica un tipo restituito diverso da *Void*, è necessario includere un'istruzione *return* seguita dal valore restituito dalla funzione.
  - Se si specifica un tipo restituito Void, non occorre occorre includere un'istruzione *return*. Se l'istruzione *return* viene specificata, non deve essere seguita da valori.
  - Indipendentemente dal tipo restituito, un'istruzione return può essere utilizzata per uscire da una funzione e restituire il controllo al codice chiamante
  - Se non si specifica un tipo return, l'inclusione di un'istruzione return è opzionale.

FUNZIONI ED EVENTI

PROGRAMMAZIONE È GESTIONE DI EVENTI



GESTIONE DI EVENTI IN ACTIONSCRIPT

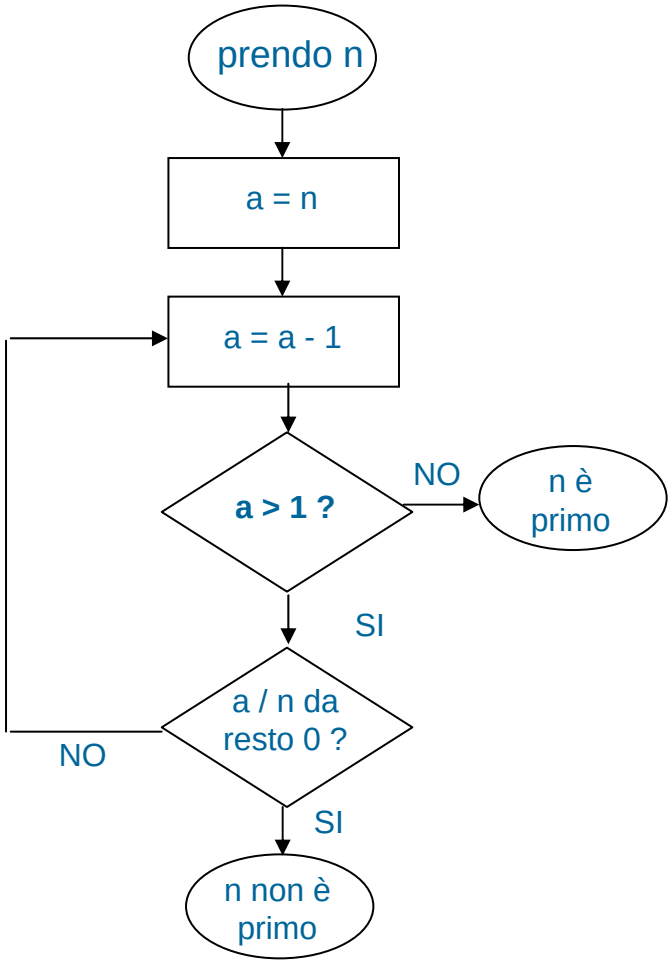
```
//uso di addEventListener
oggetto.addEventListener(nomeEvento:String,
  nomeFunzione:Function);

/* esempio */
pulsante.addEventListener(“click” ,
  calcolaQuadrato);
```

PROGETTAZIONE DI UNA FUNZIONE

NUMERO PRIMO

1: n è il numero da verificare
2: a = n
3: decremento a
4: a è maggiore di 1 ?
5: se no n è primo -> fine
6: a è un divisore di n ?
7: se sì n NON è primo -> fine
8: vado al punto 3



LA FUNZIONE PRIMO IN ACTIONSCRIPT

```
//dichiaro una funzione che accetta come parametro
//un numero e restituisce un valore Boolean (vero o falso)
function primo(n:Number):Boolean {
    /*dichiaro la variabile a di tipo number e le assegno come valore iniziale il valore di n (il numero
    che devo verificare) diminuito di 1*/
    var a:Number = n - 1;
    //inizia un ciclo che continuerà fino a che a è maggiore di 1
    while (a > 1) {
            // se a è un divisore esatto di n (cioè il resto della divisione n/a è 0)...
            if ((n % a) == 0) {
                    // n NON è un numero primo per cui restituisco il valore false
                    // la funzione termina qui e non viene eseguito alcun altro comando
                    return false;
            }
            //altrimenti (se cioè la funzione non si interrompe per effetto
            //dell'istruzione return) diminuisco a di 1 e il ciclo viene ripetuto
            a--;
    }
    // se il ciclo si concluso (a è diventato 1 per diminuzioni successive)
    // significa che non esistono divisori di n e quindi n è primo
    return true;
}
```