

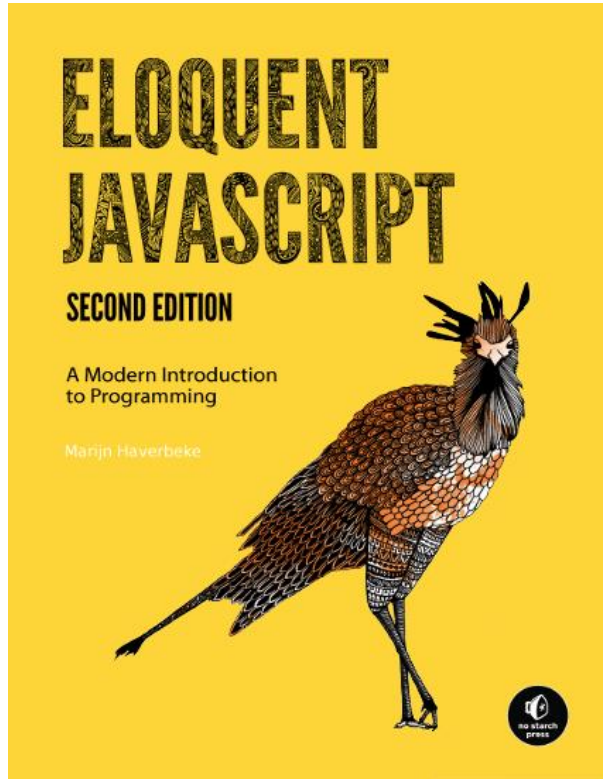
FONDAMENTI DI INFORMATICA

Alma Artis
Francesca Pratesi (ISTI, CNR)

Costrutto Switch

LIBRI E RIFERIMENTI

- Capitolo 2



Eloquent Javascript – Second Edition

Marijn Haverbeke

Licensed under CC license.

Available here: <http://eloquentjavascript.net/>



IL COSTRUTTO SWITCH

POSSIBILE ALTERNATIVA AD UNA SERIE DI IF ANNIDATI

- In alcuni casi è necessario confrontare una variabile con diversi possibile valori

```
if (variabile == valore1) istruzione1();  
else if (variabile == value2) istruzione2();  
else if (variabile == value3) istruzione3();  
else defaultAction();
```

- Per semplificare questo tipo di confronti si può utilizzare il costrutto switch

```
switch (variabile) {  
    case valore1:  
        istruzione1;  
    case valore2:  
        istruzione2;  
    ...  
    case valoreN:  
        istruzioneN;  
    default:  
        istruzioneN+1;  
}
```



ISTRUZIONE SWITCH

- Lo switch testa la variabile tra parentesi e la confronta con i valori indicati nei case
- Quando un confronto è soddisfatto, esegue tutte le istruzioni a partire da quel punto


```
var temperatura = 20;
switch (temperatura) {
  case 31: case 32: case 33: case 34: case 35: case 36: case 37: case 38: case 39: case 40: case
41: case 42: case 43: case 44: case 45:
    console.log('Molto caldo');
  case 21: case 22: case 23: case 24: case 25: case 26: case 27: case 28: case 29: case 30:
    console.log('Caldo');
  case 11: case 12: case 13: case 14: case 15: case 16: case 17: case 18: case 19: case 20:
    console.log('Gradevole');
  case 1: case 2: case 3: case 4: case 5: case 6: case 7: case 8: case 9: case 10:
    console.log('Freddo');
  default:
    console.log('Molto freddo');
}
```



ISTRUZIONE SWITCH

- Lo switch testa la variabile tra parentesi e la confronta con i valori indicati nei case
- Quando un confronto è soddisfatto, esegue tutte le istruzioni a partire da quel punto

```
var temperatura = 20;
switch (temperatura) {
  case 31: case 32: case 33: case 34: case 35: case 36: case 37: case 38: case 39: case 40: case
41: case 42: case 43: case 44: case 45:
    console.log('Molto caldo');
  case 21: case 22: case 23: case 24: case 25: case 26: case 27: case 28: case 29: case 30:
    console.log('Caldo');
  case 11: case 12: case 13: case 14: case 15: case 16: case 17: case 18: case 19: case 20:
    console.log('Gradevole');
  case 1: case 2: case 3: case 4: case 5: case 6: case 7: case 8: case 9: case 10:
    console.log('Freddo');
  default:
    console.log('Molto freddo');
}
```



Questo è il confronto
che dà true in questo
caso → il programma
stamperà
'Gradevole', ma
anche 'Freddo' e
'Molto freddo'

ISTRUZIONE SWITCH (2)

- Per poter eseguire solo un blocco di codice, devo indicare che lo switch termina
- Uso la parola chiave break

```
var temperatura = 20;
switch (temperatura) {
  case 31: case 32: case 33: case 34: case 35: case 36: case 37: case 38: case 39: case 40: case 41: case
42: case 43: case 44: case 45:
    console.log('Molto caldo');
    break;
  case 21: case 22: case 23: case 24: case 25: case 26: case 27: case 28: case 29: case 30:
    console.log('Caldo');
    break;
  case 11: case 12: case 13: case 14: case 15: case 16: case 17: case 18: case 19: case 20:
    console.log('Gradevole');
    break;
  case 1: case 2: case 3: case 4: case 5: case 6: case 7: case 8: case 9: case 10:
    console.log('Freddo');
    break;
  default:
    console.log('Molto freddo');
}
```

ESERCIZIO - 4

- Dato l'esercizio precedente (quello del grado alcolico) risolvere il problema con uno switch.
- Nota: non occorre inserire tutti i valori, ma attenersi a questa nuova tabella

Grado alcolico (g)	Messaggio
$g > 40 \ \&\& \ g < 45$	Superalcolico
$g == 20 \ \ g == 40$	Alcolico
$15 < g \leq 20$	Vino liquoroso
$13 < g \leq 15$	Vino forte
$10 < g \leq 13$	Vino normale
$g \leq 10$	Poco alcolico

ISTRUZIONE SWITCH VS IF

- Una catena di istruzioni if può fornire una soluzione più facile da gestire
 - soprattutto se vogliamo testare intervalli di valori
- D'altro canto, uno switch è più compatto se i valori da testare sono molti



ISTRUZIONE SWITCH CON INTERVALLI

```
var temperatura = -43;
switch (true) {
  case temperatura>30:
    console.log('Molto caldo');
    break;
  case temperatura>20:
    console.log('Caldo');
    break;
  case temperatura>10:
    console.log('Gradevole');
    break;
  case temperatura>0:
    console.log('Freddo');
    break;
  default:
    console.log('Molto freddo');
}
```

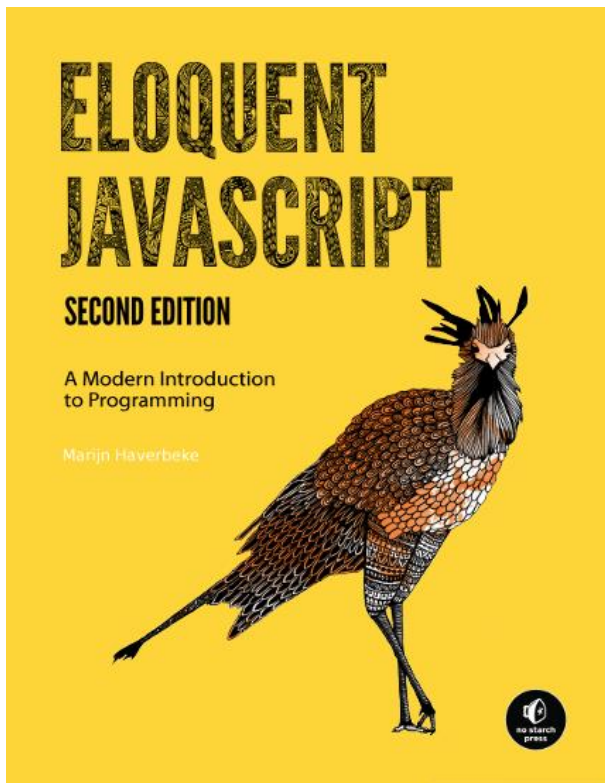




ISTRUZIONI ITERATIVE

LIBRI E RIFERIMENTI

- Capitolo 2



Eloquent Javascript – Second Edition

Marijn Haverbeke

Licensed under CC license.

Available here: <http://eloquentjavascript.net/>

ESEMPIO

- Scrivere un programma in JS che visualizzi tutti i numeri interi da uno a 10.



ESEMPIO

```
1 console.log(1);  
2 console.log(2);  
3 console.log(3);  
4 console.log(4);  
5 console.log(5);  
6 console.log(6);  
7 console.log(7);  
8 console.log(8);  
9 console.log(9);  
10 console.log(10);|
```

ESEMPIO - 2

- Scrivere un programma in JS che visualizzi tutti i numeri interi da uno a 20.



ESEMPIO - 2

```
1 console.log(1);  
2 console.log(2);  
3 console.log(3);  
4 console.log(4);  
5 console.log(5);  
6 console.log(6);  
7 console.log(7);  
8 console.log(8);  
9 console.log(9);  
10 console.log(10);  
11 console.log(11);  
12 console.log(12);  
13 console.log(13);  
14 console.log(14);  
15 console.log(15);  
16 console.log(16);  
17 console.log(17);  
18 console.log(18);  
19 console.log(19);  
20 console.log(20);
```


ESEMPIO - 3

- Scrivere un programma in JS che visualizzi tutti i numeri interi da uno a 50.



```
1 console.log(1);
2 console.log(2);
3 console.log(3);
4 console.log(4);
5 console.log(5);
6 console.log(6);
7 console.log(7);
8 console.log(8);
9 console.log(9);
10 console.log(10);
11 console.log(11);
12 console.log(12);
13 console.log(13);
14 console.log(14);
15 console.log(15);
16 console.log(16);
17 console.log(17);
18 console.log(18);
19 console.log(19);
20 console.log(20);
21 console.log(21);
22 console.log(22);
23 console.log(23);
24 console.log(24);
25 console.log(25);
26 console.log(26);
27 console.log(27);
28 console.log(28);
29 console.log(29);
30 console.log(30);
31 console.log(31);
32 console.log(32);
33 console.log(33);
34 console.log(34);
35 console.log(35);
36 console.log(36);
37 console.log(37);
38 console.log(38);
39 console.log(39);
40 console.log(40);
41 console.log(41);
42 console.log(42);
43 console.log(43);
44 console.log(44);
45 console.log(45);
46 console.log(46);
47 console.log(47);
48 console.log(48);
49 console.log(49);
50 console.log(50);
```



L'idea di usare un programma per risolvere un problema consiste nel lavorare di meno, non di più'



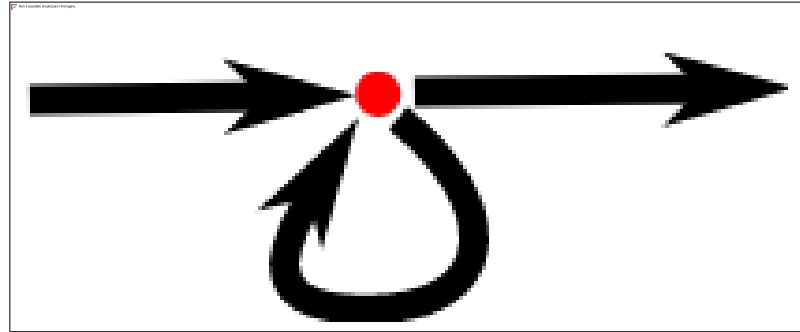
ISTRUZIONI ITERATIVE

- Molti problemi richiedono un **calcolo** che deve essere **ripetuto più volte** per ottenere il risultato finale.
- Le **istruzioni iterative** consentono di indicare la necessità di ripetere un calcolo più volte.



ISTRUZIONI ITERATIVE / CICLI

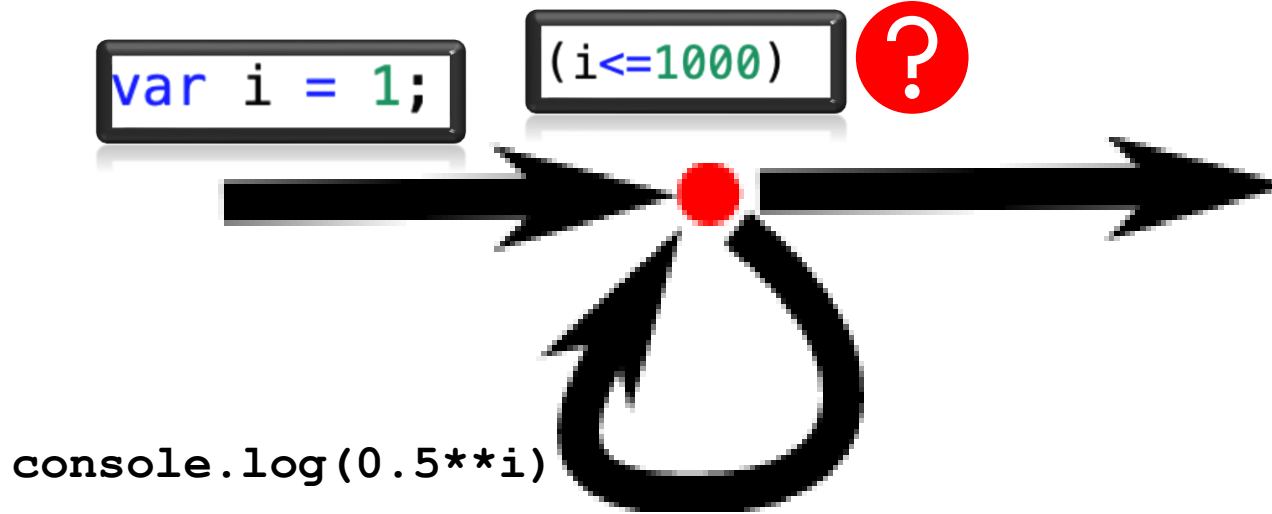
Un modo per eseguire un pezzo di codice (un insieme di istruzioni) ripetutamente



ciclo
(o loop)

IDEA

Scrivere un programma in JS che visualizzi tutte le potenze di 0.5 con esponente da 1 a 1000



TIPI DI ISTRUZIONI ITERATIVE



DETERMINATE

The diagram consists of two identical rectangular boxes side-by-side. Each box has a dark red background with a lighter red rounded rectangle in the center. The word 'DETERMINATE' is written in black capital letters inside the left box, and 'INDETERMINATE' is written in black capital letters inside the right box. The boxes are slightly offset from each other, with the right box appearing to be in front of the left one.

INDETERMINATE

ITERAZIONE DETERMINATA

Il **calcolo** viene ripetuto **un numero fissato di volte**.

Es.:

- fai 10 giri del parco di corsa
- leggi dall'input k numeri
- genera 5 numeri casuali



ITERAZIONE INDETERMINATA

Il **calcolo** viene ripetuto **finchè una condizione è vera**.

Es.:

- finchè non sei sazio mangia
- leggi un numero dall'input finchè non trovi un numero negativo
- ripeti l'esame di Fondamenti di Informatica fino a che il voto ≥ 18





ISTRUZIONI ITERATIVE: FOR

ISTRUZIONE FOR

Obiettivo: gestire le iterazioni determinate in modo diretto

- Utilizziamo una **variabile contatore** per contare quante volte viene eseguito il ciclo



ISTRUZIONE FOR

Sintassi

Il corpo del ciclo può essere un blocco (in questo caso ci sono più istruzioni che vengono ripetute)

```
for (istr1; espr2; istr3)  
    istruzione_ciclo;
```

ripete istruzione_ciclo un certo numero di volte

```
for (istr1; espr2; istr3) {  
    istruzione_blocco1;  
    istruzione_blocco2;  
    // etc...  
    istruzione_bloccoN;  
}
```



ISTRUZIONE FOR

Sintassi

```
for (istr1; espr2; istr3)  
    istruzione;
```

- **istr1**: inizializza un contatore



ISTRUZIONE FOR

Sintassi

```
for (istr1; espr2; istr3)  
    istruzione;
```

- **espr2**: una condizione che verifica se il contatore ha raggiunto il valore di fine ciclo
- se espr2 e' vera continua ad eseguire le istruzioni del ciclo



ISTRUZIONE FOR

Sintassi

```
for (istr1; espr2; istr3)  
    istruzione;
```

- **istr3**: serve a modificare (ad es., incrementa) il valore del contatore




ISTRUZIONE FOR

```
for (istr1; espr2; istr3){  
    istruzione_blocco1;  
    istruzione_blocco2;  
    // etc..  
    istruzione_bloccoN;  
}
```



ISTRUZIONE FOR

inizializzazione

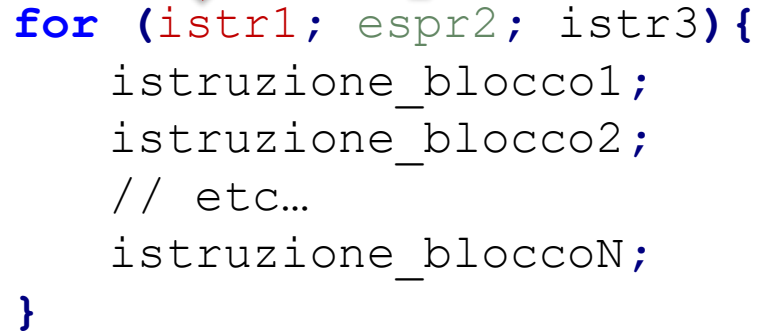


```
for (istr1; espr2; istr3){  
    istruzione_blocco1;  
    istruzione_blocco2;  
    // etc..  
    istruzione_bloccoN;  
}
```

ISTRUZIONE FOR

se è true entra nel corpo del ciclo altrimenti no
(verifica di fine ciclo)

inizializzazione



```
for (istr1; espr2; istr3){  
    istruzione_blocco1;  
    istruzione_blocco2;  
    // etc..  
    istruzione_bloccoN;  
}
```

ISTRUZIONE FOR

se è true entra nel corpo del ciclo altrimenti no
verifica di fine ciclo

inizializzazione

aggiornamento

```
for (istr1; espr2; istr3){  
    istruzione_blocco1;  
    istruzione_blocco2;  
    // etc..  
    istruzione_bloccoN;  
}
```

ISTRUZIONE FOR

se è true entra nel corpo del ciclo altrimenti no
verifica di fine ciclo

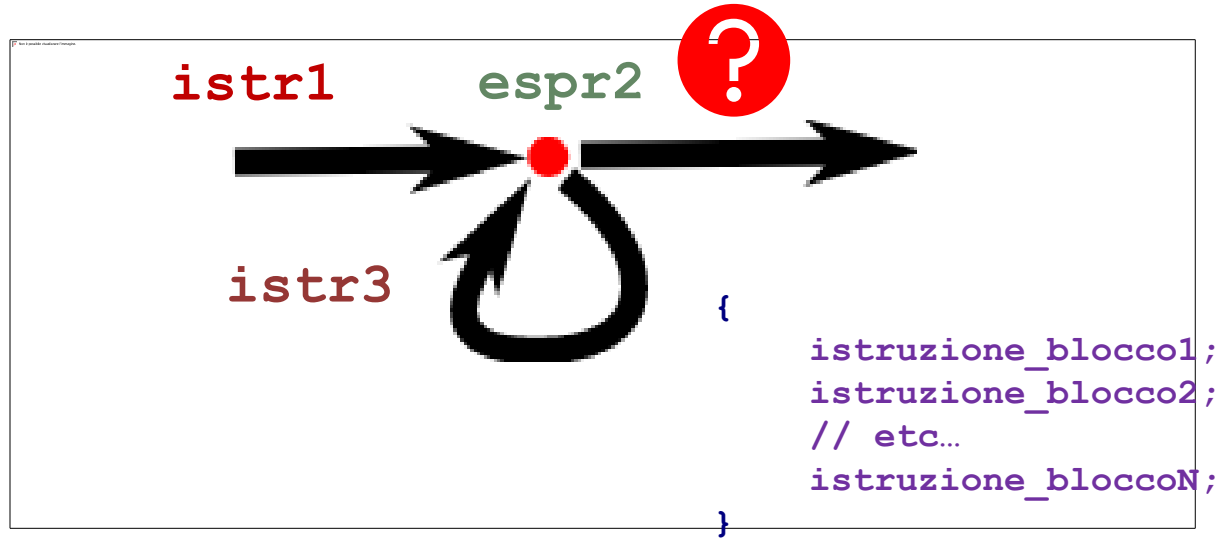
inizializzazione

aggiornamento

```
for (istr1; espr2; istr3) {  
    istruzione_blocco1;  
    istruzione_blocco2;  
    // etc..  
    istruzione_bloccoN;  
}
```

corpo del ciclo
*l'insieme delle
istruzioni che
vengono eseguite
quando il ciclo
viene eseguito*

```
for (istr1; espr2; istr3)
{
    istruzione_blocco1;
    istruzione_blocco2;
    // etc...
    istruzione_bloccoN;
}
```



ISTRUZIONE FOR

```
for (istr1; espr2; istr3){  
    istruzione_blocco1;  
    istruzione_blocco2;  
    // etc..  
    istruzione_bloccoN;  
}
```

Semantica

1. viene eseguita **istr1**
2. viene valutata **espr2**. Se **espr2** e' vera:
 - a) vengono eseguite le istruzioni nel **corpo del ciclo**
 - b) viene eseguita **istr3**
 - c) si ritorna al punto 2



ISTRUZIONE FOR

Esempio

```
var i, k = 10;  
for (i = 1; i <= k; i = i + 1){  
    //istruzioni del ciclo  
}
```



ISTRUZIONE FOR

Esempio

```
var i, k = 10;  
for (i = 1; i <= k; i = i + 1){  
    //istruzioni del ciclo  
}
```

inizializzazione del contatore

ISTRUZIONE FOR

Esempio

```
var i, k = 10;  
for (i = 1; i <= k; i = i + 1){  
    //istruzioni del ciclo  
}
```

verifica se dobbiamo eseguire
ancora altre iterazioni

inizializzazione del contatore

ISTRUZIONE FOR

Esempio

```
var i, k = 10;  
for (i = 1; i <= k; i = i + 1){  
    //istruzioni del ciclo  
}
```

incremento del contatore

verifica se dobbiamo eseguire
ancora altre iterazioni

inizializzazione del contatore

ITERAZIONE DETERMINATA - FOR

Supponiamo di voler eseguire un blocco di istruzioni 10 volte.



ITERAZIONE DETERMINATA - FOR

```
var contatore;  
for (contatore = 1; contatore <= 10; contatore++) {  
    istruzione1;  
    ...  
    istruzioneN;  
}
```



RUOLO DEL CONTATORE

Usando il **for** variamo in modo automatico il valore del contatore, ad esempio

```
for (i = 1; i <= 3; i=i+1)
```

i = 1, 2, 3



RUOLO DEL CONTATORE

Usando il for variamo in modo automatico il valore del contatore, ad esempio

```
for (i = 1; i <= 10; i=i+1)
```

i = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10



RUOLO DEL CONTATORE

Usando il for variamo in modo automatico il valore del contatore, ad esempio

```
for (i = 1; i <= 10; i=i+2)
```

i = 1, 3, 5, 7, 9



RUOLO DEL CONTATORE

Usando il for variamo in modo automatico il valore del contatore, ad esempio

```
for (i = 10; i >= 1; i=i-1)
```

i = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1



RUOLO DEL CONTATORE

Usando il for variamo in modo automatico il valore del contatore, ad esempio

```
for (i = -4; i <= 4; i=i+2)
```

i = -4, -2, 0, 2, 4



ABBREVIAZIONI

- Alcune utili funzioni per abbreviare istruzioni di assegnamento con variabili numeriche
- Data una variabile numerica x :
 - $x = x+1$;
 - si può abbreviare con $x++$;
 - $x = x-1$;
 - si può abbreviare con $x--$;



ABBREVIAZIONI - 2

- Date due variabili numeriche x e y :
 - $x = x + y$;
 - si può abbreviare con $x+=y$;
 - $x = x - y$;
 - si può abbreviare con $x-=y$;
 - $x = x * y$;
 - si può abbreviare con $x*=y$;
 - $x = x / y$;
 - si può abbreviare con $x/=y$;
 - $x = x \% y$;
 - si può abbreviare con $x\%=y$;
 - $x = x ** y$;
 - si può abbreviare con $x**=y$;



ITERAZIONE DETERMINATA - FOR

Alternativa: il contatore parte da 0

```
var contatore;  
for (contatore = 0; contatore < 10; contatore++) {  
    istruzione1;  
    ...  
    istruzioneN;  
}
```



ESEMPIO

*Scrivere una funzione che calcoli e restituisca la **somma** di tutti i **numeri** interi **dispari** compresi **tra 1 ed n** (dove n e' un parametro della funzione)*



ESEMPIO

```
1  ✓ function somma_dispari(n){  
2      var i;  
3      var risultato = 0;  
4  ✓  for(i=1;i<=n;i++){  
5      |     if(i%2!=0) risultato = risultato + i;  
6      |     }  
7      return risultato;  
8  }  
9
```

ESEMPIO 2 (CONTROLLA LE DIFFERENZE)

```
function somma_dispari_ver2(n){  
  var risultato = 0; // questo conterra' la somma finale  
  var i; //contatore  
  for (i = 1; i <= n; i+=2){  
    //il contatore ha sempre valore dispari!  
    risultato = risultato + i;  
  }  
  return risultato;  
}
```



ESEMPIO

Scrivere una funzione che calcoli e restituisca in output la somma delle potenze di due, partendo dall'esponente 0 e arrivando all'esponente n (dove n e' un parametro)



ESEMPIO

```
1  function somma_potenze(n){
2      var risultato = 0; // questo conterra' la somma finale
3
4      var contatore;
5
6      for (contatore = 0; contatore <=n; contatore++)
7          risultato = risultato + Math.pow(2,contatore);
8      return risultato;
9  }
```

ESEMPIO


Scrivere una funzione che generi n numeri casuali e restituisca il loro massimo



ESEMPIO

```
function massimo(n){  
    var max_attuale; //conterra' il massimo corrente  
    //inizializzato al piu' piccolo numero possibile  
    max_attuale = -Infinity;  
    var numero; //conterra' il numero generato ad ogni iterazione  
  
    var contatore; //variabile contatore  
    for (contatore = 1; contatore <= n; contatore++){  
        numero = Math.random(); //genera il numero  
        if (numero > max_attuale)  
            //se il numero letto e' maggiore del massimo attuale  
            max_attuale = numero; //aggiorna il massimo attuale  
    }  
    //alla fine restituisce il massimo  
    return max_attuale;  
}
```

```
function massimo(n){  
    var max_attuale;  
    max_attuale = -Infinity;  
    var numero;  
  
    var contatore;  
    for (contatore = 1; contatore <= n; contatore++){  
        numero = Math.random();  
        if (numero > max_attuale)  
            max_attuale = numero;  
    }  
    return max_attuale;  
}
```



// Scrivere lo stato nei punti indicati dai commenti, indicando con un numero (es "D1", "D2") passaggi diversi nello stesso codice

```
function foo(n){  
  var x=5;  
  // B  
  return x+n;  
}  
  
var i;  
var n=2;  
var somma = 0;  
// A  
somma = foo(5);  
// C  
for(i=1;i<=n;i++){  
  somma = somma + i;  
  // D  
}  
// E  
console.log(somma);
```