

FONDAMENTI DI INFORMATICA

Alma Artis
Francesca Pratesi (ISTI, CNR)

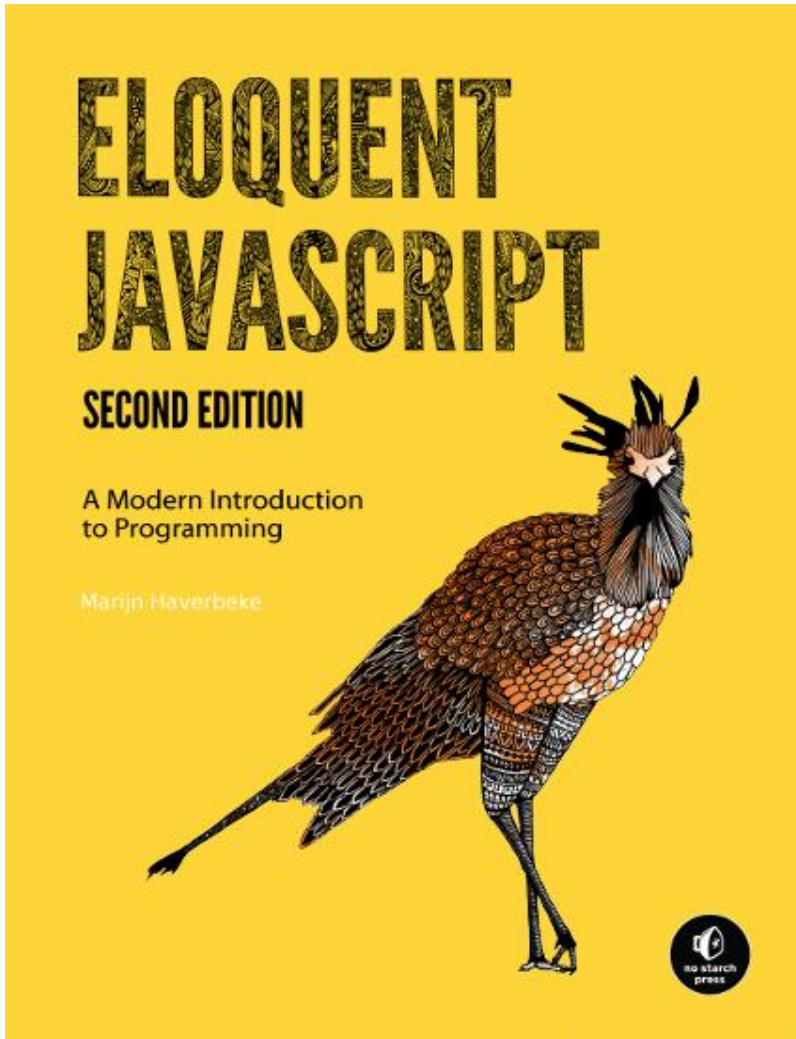
Esercizi + Array



ARRAY

LIBRI E RIFERIMENTI

- Capitolo 4



Eloquent Javascript – Second Edition
Marijn Haverbeke
Licensed under CC license.
Available here: <http://eloquentjavascript.net/>

DATI STRUTTURATI

- I tipi di dati visti finora sono:
 - numeri (interi o razionali)
 - booleani (valori di verità)
 - stringhe (sequenze di caratteri)
- Spesso è necessario manipolare dati più complessi, che presentano una struttura
 - Si parla quindi di Dati Strutturati



ARRAY O VETTORI

- Gli array rappresentano un tipo di dato composto, formato da una sequenza di valori
- Ogni valore è detto **elemento dell'array**
- Ad ogni valore è associato un indice (numero d'ordine)
- Il numero di elementi in un array è detto lunghezza (o dimensione) dell'array



ARRAY

The diagram illustrates the components of the array declaration `vet = [1, 2, 10, -1, 4, 12];`. Annotations include:

- nome dell'array**: An arrow points to the variable name `vet`.
- elementi dell'array**: Four arrows point to the values `1`, `2`, `10`, and `-1` inside the brackets.
- lunghezza = 6**: An arrow points to the closing bracket `]`, indicating the total number of elements.
- Indices and Access**: Below the array, indices and their corresponding access notation are shown:
 - `indice 0` and `vet[0]` point to the first element `1`.
 - `indice 1` and `vet[1]` point to the second element `2`.
 - `indice 2` and `vet[2]` point to the third element `10`.
 - `...` indicates the continuation of the sequence.
 - `indice 5` and `vet[5]` point to the sixth element `12`.

```
vet = [1, 2, 10, -1, 4, 12];
```

indice 0
vet[0]

indice 1
vet[1]

indice 2
vet[2]

...

indice 5
vet[5]

ARRAY IN JS

- In JS gli array sono dinamici, cioè possono avere lunghezza variabile
- Nota: In JS i valori che compongono un array possono essere omogenei (stesso tipo) o disomogenei (tipi diversi).
- Noi ci limitiamo al caso di array omogenei



DICHIARAZIONE E INIZIALIZZAZIONE

La sintassi della dichiarazione di un array prevede alcune alternative

Dichiarazione di un array vuoto:

```
var nome_array = [];
```

```
var nome_array = new Array();
```

```
var nome_array = new Array(lunghezza);
```

Dichiarazione di un array con elementi:

```
var nome_array = [espressione1, espressione2, ..., espressioneK];
```

```
var nome_array = new Array(espressione1, espressione2, ..., espressioneK);
```



ESEMPIO

```
var vettore = [1, -2, 3, 45]; // dichiaro un array di 4  
elementi  
console.log(vettore); // stampo l'array
```

```
[ 1, -2, 3, 45 ]  
Hint: hit control+c anytime to enter REPL.  
> |
```



ACCESSO AGLI ELEMENTI DI UN ARRAY

- Ogni elemento di un array è accessibile usando l'identificatore dell'array e l'indice dell'elemento
`nome_array[indice]`
- L'indice deve essere un valore numerico intero, non negativo
- L'indice può essere anche il risultato di un'espressione
`nome_array[3+n]`
- Importante: Gli array sono indicizzati a partire da 0
- es.: gli indici di un array di 6 elementi variano da 0 (primo elemento) a 5 (ultimo elemento)



ESEMPIO

```
var vettore = [1, -2, 3, 45];  
console.log(vettore[1]); // stampo l'elemento in posizione  
1, cioè -2
```



MODIFICA DI UN ARRAY

- Gli elementi di un array possono essere modificati usando il comando di assegnamento

```
var vettore = new Array(7);
```

```
vettore[0] = 1;
```

nuovo valore

elemento array



RAPPRESENTAZIONE GRAFICA

- Supponiamo di aver dichiarato
`var vettore = new Array(7);`
- Possiamo rappresentare il vettore in forma tabellare

Indice	Elemento	Variabile
0	undefined	vettore[0]
1	undefined	vettore[1]
2	undefined	vettore[2]
3	undefined	vettore[3]
4	undefined	vettore[4]

MODIFICA DI UN ARRAY

```
var vettore = new Array(7);  
vettore[0] = 1;
```

Indice	Elemento	Variabile
0	1	vettore[0]
1	undefined	vettore[1]
2	undefined	vettore[2]
3	undefined	vettore[3]
4	undefined	vettore[4]
5	undefined	vettore[5]

MODIFICA DI UN ARRAY

- L'indice può essere una qualsiasi espressione che abbia un valore intero non negativo

```
var i = Math.round(4.1);  
vettore[i] = Math.log(1);
```

Indice	Elemento	Variabile
0	1	vettore[0]
1	undefined	vettore[1]
2	undefined	vettore[2]
3	undefined	vettore[3]
4	0	vettore[4]

MODIFICA DI UN ARRAY

- Anche il nuovo valore può dipendere dagli elementi dell'array
`vettore[1] = vettore[0] + 2;`
`vettore[2] = vettore[0] + vettore[1];`

Indice	Elemento	Variabile
0	1	vettore[0]
1	3	vettore[1]
2	4	vettore[2]
3	undefined	vettore[3]
4	0	vettore[4]

ESEMPIO

- Esempio: costruire un vettore di lunghezza 10 i cui elementi siano numeri casuali tra 0 e 1



ESEMPIO

- Esempio: costruire un vettore di lunghezza 10 i cui elementi siano numeri casuali tra 0 e 1

```
var vet = new Array(10);  
var i;  
for (i = 0; i < vet.length; i++)  
    vet[i] = Math.random();
```



LO STATO CON GLI ARRAY

- Quando si usano gli array, lo stato viene esteso
- Le variabili che hanno come valore un array assumono nell'ambiente un valore speciale detto reference
 - il reference e' un indirizzo di memoria che individua l'area dello heap in cui e' memorizzato l'array



REFERENCE E HEAP

- Lo heap in cui è memorizzato l'array è costituito da più locazioni di memoria (illustrato come nelle precedenti tabelle)
- Il reference è indicato graficamente come una freccia (che punta allo heap)
 - oppure con *ref* dal momento che l'indirizzo effettivo di memoria è un dettaglio che dipende dal gestore della memoria (non ci interessa)



ESEMPIO DI STATO

```
var vet1 = new Array(6);
```

Stato risultante:

```
{{vet1, }}
```



0	undefined
1	undefined
2	undefined

ESEMPIO DI STATO (2)

```
var vet1 = new Array(6);  
var vet2 = new Array(6);  
var a = 0;  
vet1[0] = 4;  
vet1[1] = a;  
vet2[1] = vet1[1]+2;  
vet2 = vet1;  
vet1[2] = vet2[1] + 1;
```

{{vet1,}}

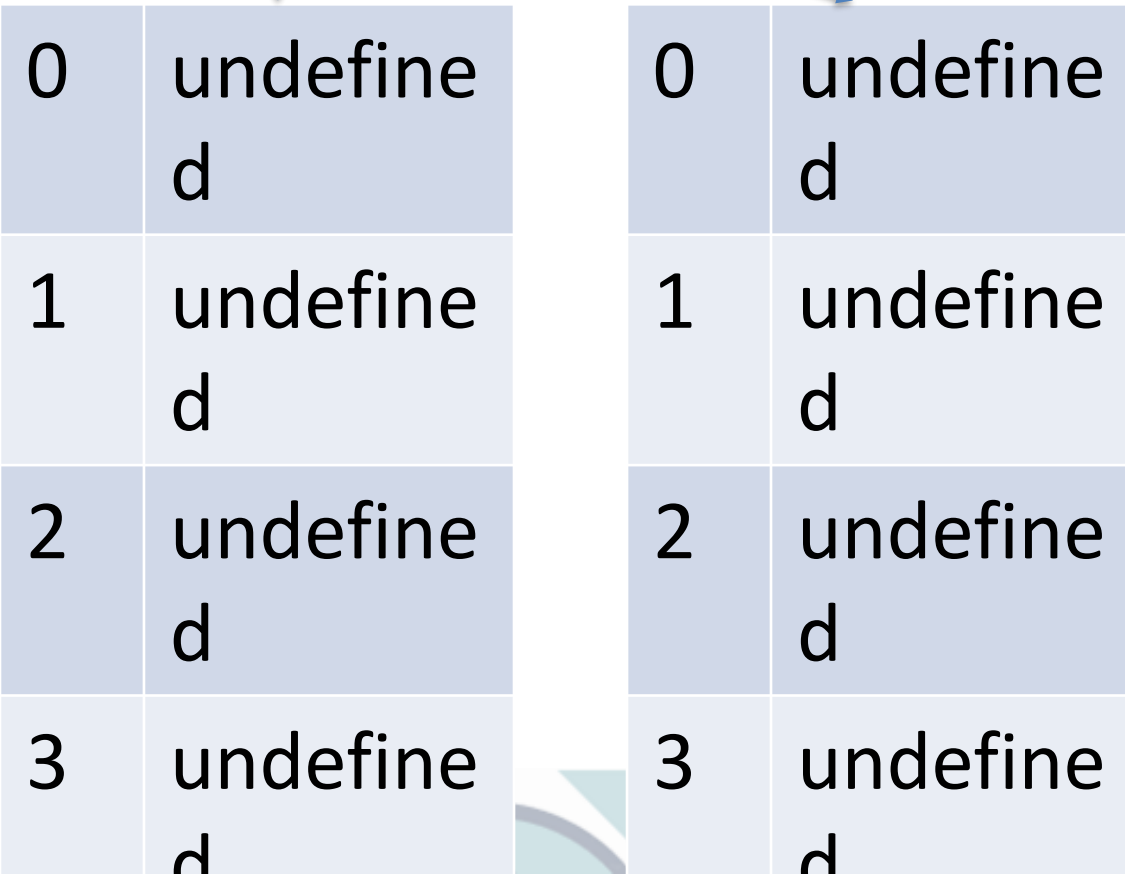


0	undefined
1	undefined
2	undefined
3	undefined

ESEMPIO DI STATO (2)

```
var vet1 = new Array(6);  
var vet2 = new Array(6);  
var a = 0;  
vet1[0] = 4;  
vet1[1] = a;  
vet2[1] = vet1[1]+2;  
vet2 = vet1;  
vet1[2] = vet2[1] + 1;
```

{{vet1, }, (vet2,)}

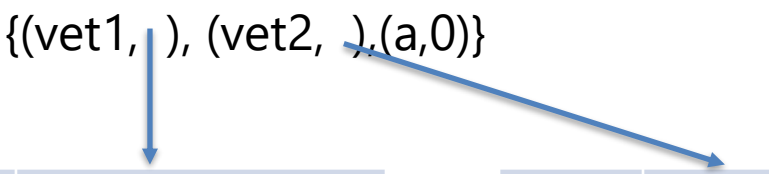


0	undefined	0	undefined
1	undefined	1	undefined
2	undefined	2	undefined
3	undefined	3	undefined

ESEMPIO DI STATO (2)

```
var vet1 = new Array(6);  
var vet2 = new Array(6);  
var a = 0;  
vet1[0] = 4;  
vet1[1] = a;  
vet2[1] = vet1[1]+2;  
vet2 = vet1;  
vet1[2] = vet2[1] + 1;
```

{(vet1,), (vet2,), (a, 0)}




0	undefined
1	undefined
2	undefined
3	undefined

0	undefined
1	undefined
2	undefined
3	undefined

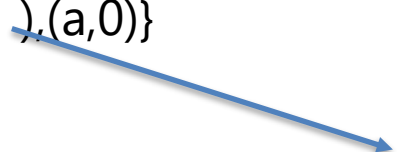
ESEMPIO DI STATO (2)

```
var vet1 = new Array(6);  
var vet2 = new Array(6);  
var a = 0;  
vet1[0] = 4;  
vet1[1] = a;  
vet2[1] = vet1[1]+2;  
vet2 = vet1;  
vet1[2] = vet2[1] + 1;
```

{(vet1,), (vet2,), (a, 0)}



0	4
1	undefined
2	undefined
3	undefined
4	undefined




0	undefined
1	undefined
2	undefined
3	undefined

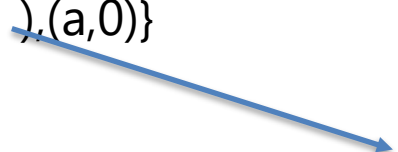
ESEMPIO DI STATO (2)

```
var vet1 = new Array(6);  
var vet2 = new Array(6);  
var a = 0;  
vet1[0] = 4;  
vet1[1] = a;  
vet2[1] = vet1[1]+2;  
vet2 = vet1;  
vet1[2] = vet2[1] + 1;
```

{(vet1,), (vet2,), (a, 0)}



0	4
1	0
2	undefined
3	undefined
4	undefined




0	undefined
1	undefined
2	undefined
3	undefined

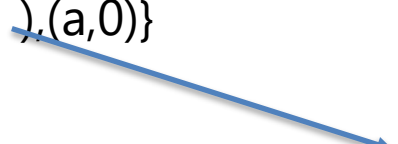
ESEMPIO DI STATO (2)

```
var vet1 = new Array(6);  
var vet2 = new Array(6);  
var a = 0;  
vet1[0] = 4;  
vet1[1] = a;  
vet2[1] = vet1[1]+2;  
vet2 = vet1;  
vet1[2] = vet2[1] + 1;
```

{(vet1,), (vet2,), (a, 0)}



0	4
1	0
2	undefined
3	undefined
4	undefined

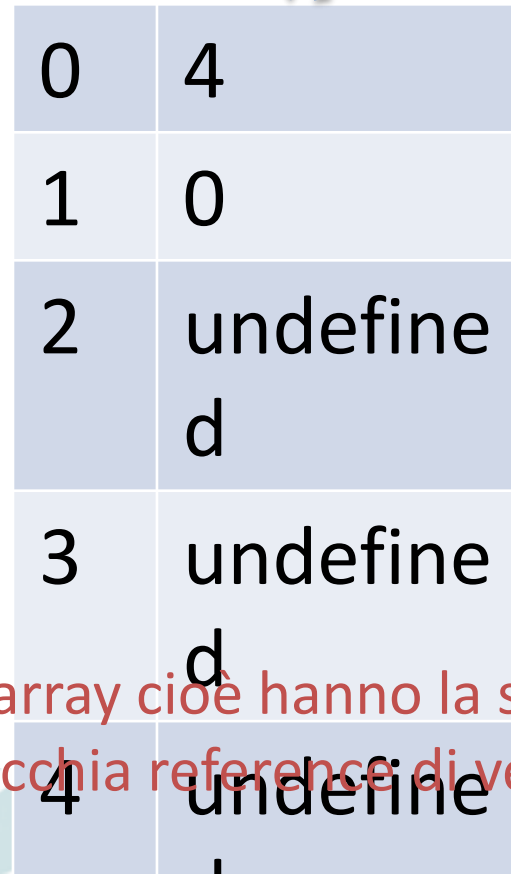


0	undefined
1	2
2	undefined
3	undefined
4	undefined

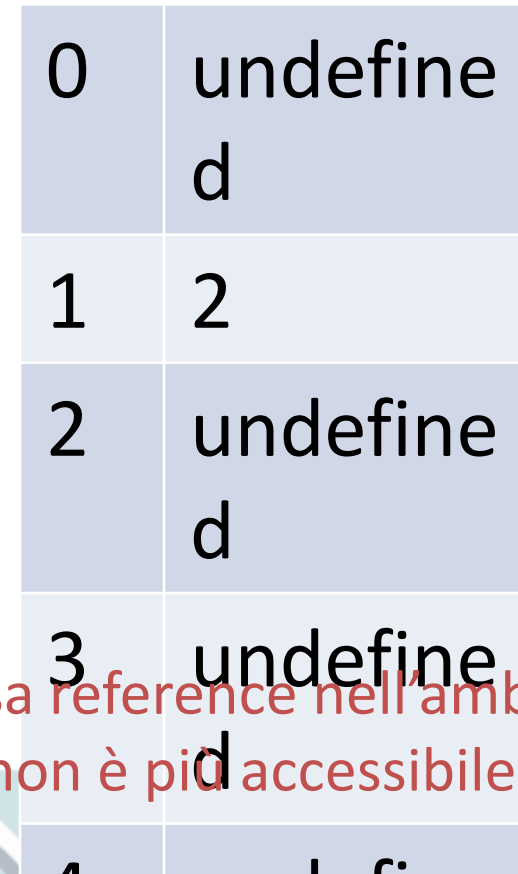
ESEMPIO DI STATO (2)

```
var vet1 = new Array(6);  
var vet2 = new Array(6);  
var a = 0;  
vet1[0] = 4;  
vet1[1] = a;  
vet2[1] = vet1[1]+2;  
vet2 = vet1;  
vet1[2] = vet2[1] + 1;
```

{(vet1,), (vet2,), (a, 0)}



0	4
1	0
2	undefined
3	undefined
4	undefined



0	undefined
1	2
2	undefined
3	undefined
4	undefined

vet1 e vet2 condividono lo stesso array cioè hanno la stessa reference nell'ambiente
l'area di memoria puntata dalla vecchia reference di vet2 non è più accessibile

ESEMPIO DI STATO (2)

```
var vet1 = new Array(6);  
var vet2 = new Array(6);  
var a = 0;  
vet1[0] = 4;  
vet1[1] = a;  
vet2[1] = vet1[1]+2;  
vet2 = vet1;  
vet1[2] = vet2[1] + 1;
```

{(vet1,), (vet2,), (a, 0)}



0	4
1	0
2	1
3	undefined
4	undefined

0	undefined
1	2
2	undefined
3	undefined
4	undefined

la modifica a vet1 modifica anche vet2, dal momento che hanno la stessa reference nell'ambiente

ALCUNE OSSERVAZIONI

- Anche l'operatore di uguaglianza opera sugli array: due variabili array sono uguali se condividono la stessa struttura
 - esercizio: valutare `(vet1 == vet2)` prima e dopo l'istruzione `vet2 = vet1;`
- Gli operatori di confronto (`<` e `>`) non sono definiti su array
 - attenzione: non viene generato errore, ma sempre false!



LUNGHEZZA DI UN ARRAY

- In JS la lunghezza di un array è una proprietà dell'array
- Per conoscere la lunghezza di un array occorre accedere alla sua proprietà `length`
- Sintassi: `nome_array.length`
 - Attenzione: la lunghezza dell'array è uguale al massimo indice + 1



MANIPOLAZIONE DI ARRAY

- La manipolazione di array avviene spesso tramite cicli, e in particolare (di solito) con l'istruzione iterativa for
- L'indice del ciclo varia da 0 a lunghezza-1
 - viene usato per scandire tutto l'array e per accedere a ciascun elemento dell'array

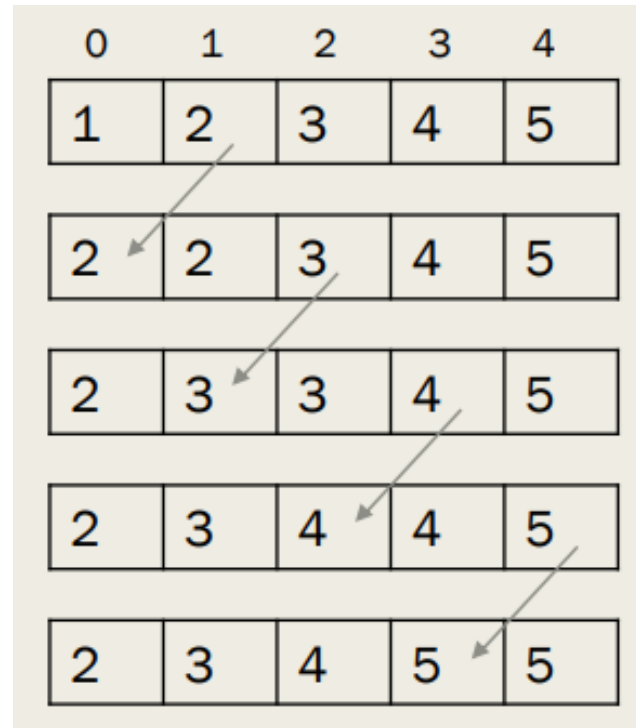


ESEMPIO - SHIFT

Shift (spostamento) a sinistra degli elementi di un vettore

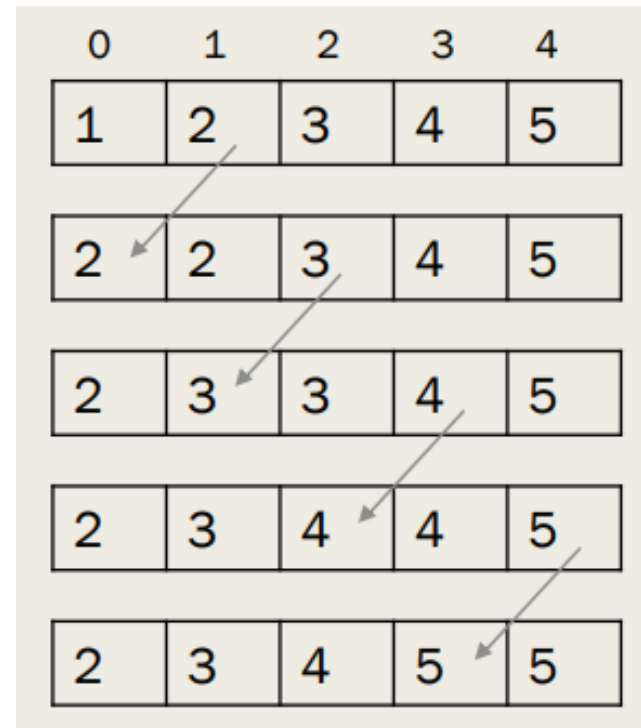
Occorre spostare di una posizione a sinistra tutti gli elementi del vettore.

Es. con array di lunghezza 5



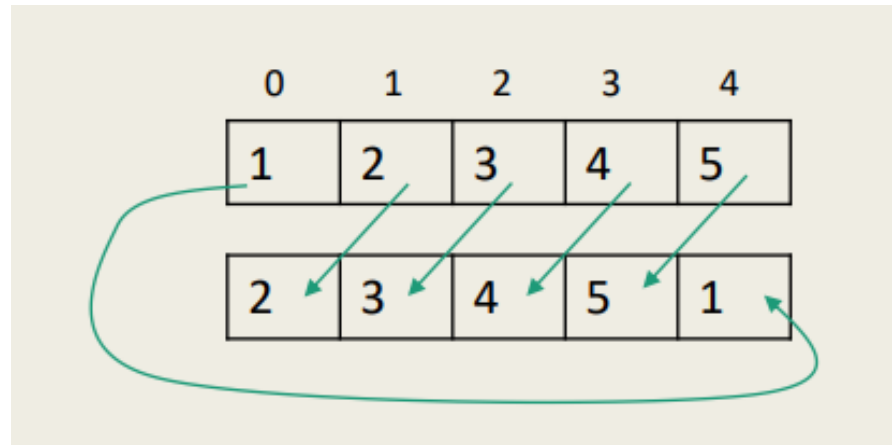
ESEMPIO - SHIFT

```
var vet = new Array(5);  
vet = [1,2,3,4,5];  
for(i = 0; i<vet.length; i++)  
    vet[i]=vet[i+1];
```



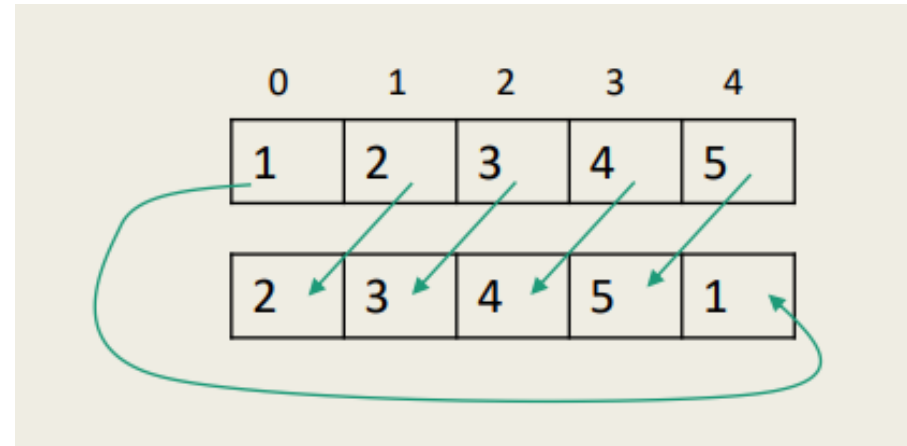
ESEMPIO – SHIFT CIRCOLARE

Lo shift circolare a sinistra è uno shift in cui il primo elemento viene spostato in ultima posizione



ESEMPIO – SHIFT CIRCOLARE

```
var vet = new Array(5);  
vet = [1,2,3,4,5];  
var primo = vet[0];  
for(i = 0; i<vet.length; i++)  
    vet[i]=vet[i+1];  
vet[vet.length-1] = primo;
```



ARRAY COME PARAMETRI DI FUNZIONI

- Quando si passa un vettore come parametro ad una funzione si sta passando la reference del vettore
- Le modifiche sul parametro formale modificano anche il parametro attuale!!!
 - Il passaggio di una reference di un vettore consente alla funzione di modificare gli elementi del vettore passato
- Analogamente, una funzione può restituire un array
 - cioè può restituire il riferimento all'area di memoria in cui sono contenuti i valori degli elementi dell'array
 - questo avviene naturalmente tramite l'istruzione return



ESEMPIO

```
function leggi_array(v){  
    var i;  
    for (i = 0; i<v.length; i++)  
        v[i]=Number(prompt('inserisci un numero'));  
}  
var vettore = new Array(4);  
leggi_array(vettore);  
console.log(vettore);
```



ESEMPIO - ALTERNATIVA

```
function crea_array(k){  
    var i,  
    var v = new Array(k)  
    for (i = 0; i<v.length; i++)  
        v[i]=Number(prompt('inserisci un numero'));  
    return v;  
}  
var lunghezza = 4;  
var vettore;  
Vettore = crea_array(lunghezza);  
console.log(vettore);
```



ESEMPIO DI STATO CON ARRAY E FUNZIONI

```
function leggi_array(v){  
  var i;  
  for (i = 0; i<v.length; i++)  
    v[i]=Number(prompt('inserisci un numero'));  
}
```

```
var vettore = new Array(4);  
leggi_array(vettore);  
console.log(vettore);
```

`{(vettore,),(leggi_array, function(v){...})}`

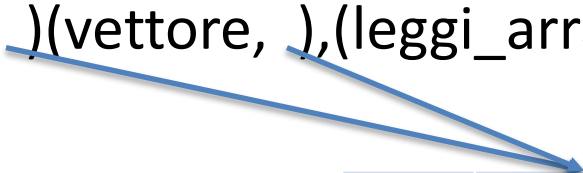


0	undefin ed
1	undefin ed
2	undefin

ESEMPIO DI STATO CON ARRAY E FUNZIONI

```
function leggi_array(v){  
    var i;  
    for (i = 0; i<v.length; i++)  
        v[i]=Number(prompt('inserisci un numero'));  
}  
var vettore = new Array(4);  
leggi_array(vettore);  
console.log(vettore);
```

`{{(i,undefined),(v,)(vettore,),(leggi_array, function(v){...})}}`




0	undefin ed
1	undefin ed
2	undefin

ESEMPIO DI STATO CON ARRAY E FUNZIONI

```
function leggi_array(v){  
  var i;  
  for (i = 0; i<v.length; i++)  
    v[i]=Number(prompt('inserisci un numero'));  
}  
var vettore = new Array(4);  
leggi_array(vettore);  
console.log(vettore);
```

`{(i,0),(v,)(vettore,),(leggi_array, function(v){...})}`




0	10
1	undefined
2	undefined

ESEMPIO DI STATO CON ARRAY E FUNZIONI

```
function leggi_array(v){  
  var i;  
  for (i = 0; i<v.length; i++)  
    v[i]=Number(prompt('inserisci un numero'));  
}  
var vettore = new Array(4);  
leggi_array(vettore);  
console.log(vettore);
```

`{(i,1),(v,)(vettore,),(leggi_array, function(v){...})}`



0	10
1	5
2	undefined
3	undefined

ESEMPIO DI STATO CON ARRAY E FUNZIONI

```
function leggi_array(v){  
    var i;  
    for (i = 0; i<v.length; i++)  
        v[i]=Number(prompt('inserisci un numero'));  
}  
var vettore = new Array(4);  
leggi_array(vettore);  
console.log(vettore);
```

`{(i,2),(v,)(vettore,),(leggi_array, function(v){...})}`



0	10
1	5
2	15
3	undefin

ESEMPIO DI STATO CON ARRAY E FUNZIONI

```
function leggi_array(v){  
    var i;  
    for (i = 0; i<v.length; i++)  
        v[i]=Number(prompt('inserisci un numero'));  
}  
var vettore = new Array(4);  
leggi_array(vettore);  
console.log(vettore);
```

`{(i,3),(v,)(vettore,),(leggi_array, function(v){...})}`




0	10
1	5
2	15
3	20

ESEMPIO DI STATO CON ARRAY E FUNZIONI

```
function leggi_array(v){  
    var i;  
    for (i = 0; i<v.length; i++)  
        v[i]=Number(prompt('inserisci un numero'));  
}  
var vettore = new Array(4);  
leggi_array(vettore);  
console.log(vettore);
```

({vettore, },(leggi_array, function(v){...}))



0	10
1	5
2	15
3	20



ARRAY DINAMICI

ARRAY DINAMICI

- In JS gli array sono strutture dinamiche: il numero degli elementi può variare durante l'esecuzione del programma
- Per aggiungere un elemento ad array è possibile assegnare un valore ad un elemento di indice successivo all'ultimo attualmente esistente



ESEMPIO

```
var L = 5, i = 0;
var vet = new Array(L); //vet ha lunghezza 5
//gli elementi del vettore vengono riempiti con le potenze di 2
for (i = 0; i < L; i++) {
    vet[i] = Math.pow(2,i);
}
vet[L] = Math.pow(2,L); //<--- vet ora ha lunghezza 6
vet[L+2] = -1; //aggiunge due elementi
console.log(vet[L+1]) // che valore viene stampato?
```



ESEMPIO

```
var L = 5, i = 0;  
var vet = new Array(L);  
  
for (i = 0; i < L; i++) {  
    vet[i] = Math.pow(2,i);  
}  
vet[L] = Math.pow(2,L);  
vet[L+2] = -1;  
console.log(vet[L+1])
```

0	undefined
1	undefined
2	undefined
3	undefined
4	undefined

ESEMPIO

```
var L = 5, i = 0;  
var vet = new Array(L);  
  
for (i = 0; i < L; i++) {  
    vet[i] = Math.pow(2,i);  
}  
vet[L] = Math.pow(2,L);  
vet[L+2] = -1;  
console.log(vet[L+1])
```

0	1
1	2
2	4
3	8
4	16

ESEMPIO

```
var L = 5, i = 0;  
var vet = new Array(L);  
  
for (i = 0; i < L; i++) {  
    vet[i] = Math.pow(2,i);  
}  
vet[L] = Math.pow(2,L);  
vet[L+2] = -1;  
console.log(vet[L+1])
```

0	1
1	2
2	4
3	8
4	16
5	32

ESEMPIO

```
var L = 5, i = 0;
var vet = new Array(L);

for (i = 0; i < L; i++) {
    vet[i] = Math.pow(2,i);
}
vet[L] = Math.pow(2,L);
vet[L+2] = -1;
console.log(vet[L+1])
```

0	1
1	2
2	4
3	8
4	16
5	32
6	undefined
7	-1

ESERCIZI

- Scrivere una funzione ricorsiva che, dato un numero N calcola la somma dei primi N numeri pari positivi in maniera ricorsiva.
- Scrivere una funzione ricorsiva potenza che riceva due interi, base e esponente e ritorni il valore della base elevata alla potenza esponente.
- Creare una funzione ricorsiva per calcolare una funzione definita così:
per $m > 0$ allora $f(n, m) = 1 + f(n, m - 1)$
per $m = 0$ allora $f(n, m) = n$
Una volta implementata, provarla e dire cosa calcola la funzione.
- Scrivere una funzione ricorsiva che (senza usare cicli) ricevuti due numeri interi calcoli la somma di tutti i numeri dell'intervallo tra i due numeri, compresi i numeri stessi.
- Scrivere una funzione ricorsiva che ricevuto un intero n stampi n '-' seguiti da n '+'.
Ad esempio se la funzione riceve 3 stampa - - - + + +.

