



Laboratorio I

a.a. 2026/2027

Array

Prima di iniziare: news tutorato!

Potrete comunicare con i tutor del corso (e tra voi) sul gruppo Telegram dedicato:



Prima di iniziare: prove in itinere

Le prove in itinere del primo semestre si terranno in presenza nelle date seguenti:

1. Prima prova in itinere: **4 novembre ore 14:00** (durante le ore di lezione)
2. Seconda prova in itinere: **2 dicembre ore 14:00** (durante le ore di lezione)

Maggiori informazioni sul post Teams corrispondente

Contenuti

- Array
 - Denotazione
 - Esempi, uso come liste
 - Qualche programma con gli Array
- Esercizi su array

Recap: Tipi di dato primitivo e valori letterali in JS/TS

JavaScript consente di denotare **valori letterali** di determinati tipi.

I seguenti sono i corrispondenti tipi in TypeScript:

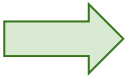
- **boolean**: `true`, `false`
- **undefined**: `undefined` (→ **non è stato definito** nessun valore)
- **null**: `null` (→ so che **non c'è** un valore)
- **number**: `1`, `5`, `-53.38`, `12.3e4` (numeri “normali”, $-(2^{53}-1) \leq n \leq 2^{53}-1$)
- **bigint**: `90071992547434344169871610992n` (interi a precisione infinita)
- **string**: `“ciao ugo”`, `“हिन्दू”`, `‘hello’`, ``x vale ${x}`` (sequenze di caratteri)

Ricordate questo
lucido?

Tipi di dato **non** primitivi

Ci sono poi tipi di dato non primitivi, che in JavaScript ricadono sotto “Object”, e di cui vedremo più avanti la tipizzazione in TypeScript

- **Object:** `{ nome: “Ugo”, età: 19 }` (mappe da chiavi a valori - anche dizionari)



- **Array:** `[ 1, 5, 8, 12, 21, 33 ], [ 1, “ciao!”, [ “banana” ] ]`
(liste con indici numerici)

- **Function:** `x=>2*x, (a,b)=>a+b, x=> { if (x>0) return x; else return -x }` (funzioni)

Gli array

Gli **array** sono una struttura dati che memorizza degli elementi (tradizionalmente tutti dello stesso tipo) in una sequenza ordinata.

Ogni elemento occupa una posizione identificata da un **indice**

indici	0	1	2	3	4	5	6	7	8	9	10
elementi	2	5	18	5	8	9	0	22	7	12	-4

In TypeScript, un array di n elementi ha indici che vanno da 0 ad $n-1$

Gli array

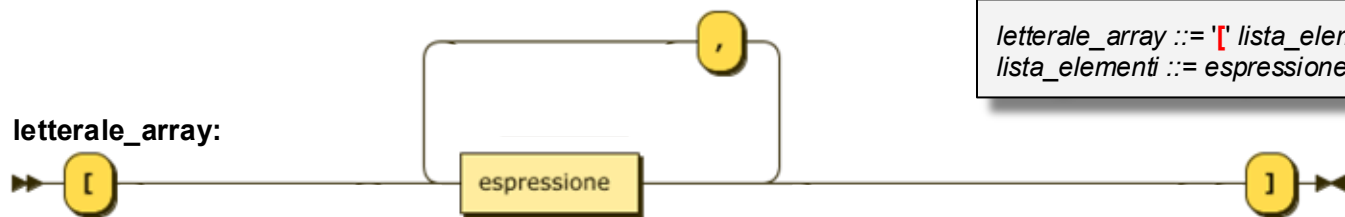
indici	0	1	2	3	4	5	6	7	8	9	10
elementi	2	5	18	5	8	9	0	22	7	12	-4

L'ordine dei numeri naturali (degli indici) definisce un ordine degli elementi: ad esempio, $[1, 2, 2]$ e $[2, 1, 2]$ sono due array diversi.

Gli array sono quindi liste, o sequenze ordinate, anziché insiemi di elementi: la posizione conta, e uno stesso valore può comparire più volte.

NB: In TS gli array sono un tipo speciale di *oggetto*, ma vedremo i dettagli più avanti...

Costanti letterali di tipo array - una prima sintassi

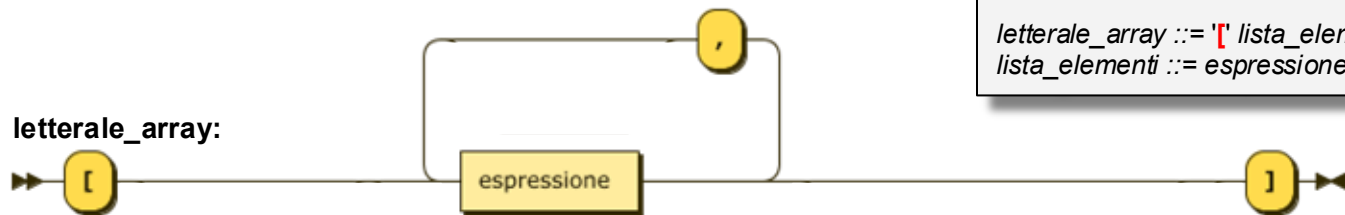


```
letterale_array ::= '[' lista_elementi ']'  
lista_elementi ::= espressione | lista_elementi ',' espressione
```

In pratica, è una serie di espressioni tra parentesi quadre: [*espressione* , *espressione* , ...]

```
let pluto:number[] = [2, 10, 3, 4 , 1]
```

Costanti letterali di tipo array - una prima sintassi

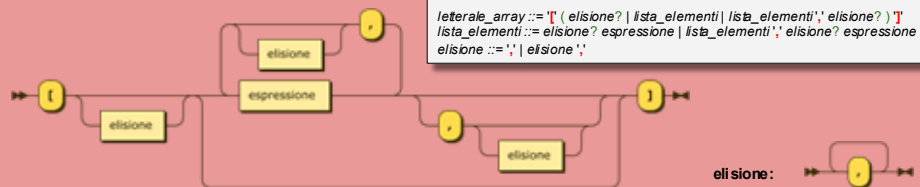


```
letterale_array ::= '[' lista_elementi ']'  
lista_elementi ::= espressione | lista_elementi ',' espressione
```

In pratica, è una serie di espressioni tra parentesi quadre: [*espressione* , *espressione* , ...]

```
let pluto:number[] = [2, 10, 3, 4 , 1]
```

SPOILER: la sintassi completa degli array in TypeScript è un po' più complessa, ma la vedremo più avanti



Il tipo degli array in TypeScript

In TypeScript, per esprimere il tipo di un array **omogeneo** di tipo $\mathfrak{t}$ (ovvero, dove tutti gli elementi hanno lo stesso tipo $\mathfrak{t}$) si utilizza la sintassi $\mathfrak{t}[]$

```
let arr:number[] = [1, 4, 6];  
let arrs:string[] = ["array", "di", "stringhe"];
```

- Si può inizializzare un array vuoto: `let x:number[] = [];`
- Il numero di elementi dell'array può variare

```
let arr:number[];  
arr = [1,3,5];  
arr = [4,-1];
```

Il Tipo Tupla

In TypeScript, si possono anche definire array **non omogenei** formati da elementi di tipo diverso. Per rappresentare il loro tipo, si utilizza un tipo specifico: il **tipo tupla**

- Nel tipo tupla, si indica *per ogni indice dell'array* qual è il tipo dell'elemento corrispondente
- Gli elementi dell'array devono essere forniti obbligatoriamente in quell'ordine con quei tipi (**e il loro numero non deve essere cambiato**).
- Sintassi di una tupla formata dagli N tipi `tipo1, tipo2, ..., tipoN`

`[tipo1, tipo2, ... , tipoN]`

Il Tipo Tupla: esempio

- Tupla formata da una stringa e un numero

```
let x: [string, number]; // Dichiarazione
```

```
x = ["hello", 10]; // Inizializzazione OK
```

Il Tipo Tupla: esempio

- Tupla formata da una stringa e un numero

```
let x: [string, number]; // Dichiarazione
```

```
x = ["hello", 10]; // Inizializzazione OK
```

```
x = [10, "hello"]; // Errore: ordine sbagliato
```

```
Type 'number' is not assignable to type 'string'.
```

```
Type 'string' is not assignable to type 'number'.
```

Il Tipo Tupla: esempio

- Tupla formata da una stringa e un numero

```
let x: [string, number]; // Dichiarazione
```

```
x = ["hello", 10]; // Inizializzazione OK
```

```
x = [10, "hello"]; // Errore: ordine sbagliato
```

```
Type 'number' is not assignable to type 'string'.
```

```
Type 'string' is not assignable to type 'number'.
```

```
x = ["hello", 10, 11]; // Errore: numero elementi sbagliato
```

```
Type '[string, number, number]' is not assignable to type '[string, number]'.
```

```
Source has 3 element(s) but target allows only 2.
```

Operazioni sugli array - accesso e lunghezza

chiave indice	valore elemento
0	"Alina"
1	"Vincenzo"
2	"Giuseppe"
3	"Davide"

Array di lunghezza 4
(indici fra 0 e 3)

Per accedere all'elemento dell'array `arr` che si trova in posizione (indice) `i`, si utilizzano le parentesi quadre: `arr[i]`

La lunghezza dell'array `arr` (numero di elementi che contiene) si può vedere utilizzando `arr.length`

```
let a:string[] = [ "Alina", "Vincenzo", "Giuseppe", "Davide" ]  
  
a[0] // "Alina"  
a[2]="Peppe"  
  
a // [ "Alina", "Vincenzo", "Peppe", "Davide" ]  
a.length → 4
```

Array e comandi iterativi TypeScript

Nella lezione sui comandi iterativi, avevamo accennato che in TypeScript esistono due forme speciali dei cicli **for** ideate appositamente per iterare sugli array

for..in

```
let studenti: string[] = ['Matt', 'Alice', 'Bob'];
for (let i in studenti) {
  console.log(studenti[i]);
}
```

Permette di iterare **sugli indici**

for..of

```
let studenti: string[] = ['Matt', 'Alice', 'Bob'];
for (let s of studenti) {
  console.log(s);
}
```

Permette di iterare **sugli elementi**

Entrambi stampano **'Matt', 'Alice', 'Bob'**

Array di array

Gli elementi di un array possono a loro volta essere array!

Per dichiararli, si utilizzano le doppie parentesi quadre

```
let x: number[][]; // Dichiarazione array bidimensionale  
x = [[3,5],[1,-4],[9]]; // Inizializzazione OK
```

Array di array

Gli elementi di un array possono a loro volta essere array!

Per dichiararli, si utilizzano le doppie parentesi quadre

```
let x: number[][]; // Dichiarazione array bidimensionale  
x = [[3,5],[1,-4],[9]]; // Inizializzazione OK
```

Attenzione: in questo modo restano array omogenei, con tutti gli elementi di tipo array

Array di array

Gli elementi di un array possono a loro volta essere array!

Per dichiararli, si utilizzano le doppie parentesi quadre

```
let x: number[][]; // Dichiarazione array bidimensionale  
x = [[3,5],[1,-4],[9]]; // Inizializzazione OK
```

Attenzione: in questo modo restano array omogenei, con tutti gli elementi di tipo array

```
x = [[3,5],1,[9]]; // Errore: uno degli elementi non è un array!
```

```
Type 'number' is not assignable to type 'number[]'.
```

Array di array

Gli elementi di un array possono a loro volta essere array!

Analogamente, per le tuple

```
let x: [string, number][]; // array di tuple [string, number]
```

```
x = [["a",5],["b",-4]]; // Inizializzazione OK
```

```
x = [["a",5],["b",-4], 1]; // Errore: un elemento non è una tupla!
```

```
Type 'number' is not assignable to type '[string, number]'.
```

Array di array

Gli elementi di un array possono a loro volta essere array!

Analogamente, per le tuple

```
let x: [string, number][]; // array di tuple [string, number]
x = [["a",5],["b",-4]]; // Inizializzazione OK
x = [["a",5],["b",-4], 1]; // Errore: un elemento non è una tupla!
```

Oppure anche

```
let x: [string, number[]]; // tuple di cui un elemento è un array
x = ["a",[5,3,-1,2]]; // Inizializzazione OK
```

Altre operazioni sugli array

Gli array hanno una serie di funzioni che semplificano le operazioni su liste, quali

- `a.push(z)` – aggiunge `z` in coda all'array `a`
- `c=a.concat(b)` – concatena l'array `b` in coda all'array `a`, restituisce il risultato
- `a.pop()` – rimuove l'ultimo elemento da `a`
- `a.shift()` – rimuove il primo elemento di `a`, restituisce l'elemento rimosso; tutti gli altri elementi scorrono di un posto per “chiudere il buco”
- `a.sort()` – ordina un array in base al valore degli elementi

Esercizio 1 - Giorni sempre piu caldi

(Giorni sempre più caldi) Si vogliono ottenere delle statistiche sulle temperature alte registrate durante una settimana. Consideriamo un giorno di **caldo intenso** se la temperatura massima è **uguale o superiore ai 32°C**.

Scrivere un programma che legga da tastiera le temperature massime di sette giorni, e le memorizzi in un array. Successivamente, si conti il numero di giorni in cui c'è stato caldo intenso durante la settimana data, e lo si stampi.

Esempio

Input: [29, 32, 30, 34, 28, 32, 31]

Output: 3

Esercizio 2 - Inventario

Si scriva un programma per gestire un inventario, formato da un insieme di coppie prodotto-quantità. Tali coppie vengono lette da tastiera, fino a quando l'utente inserisce "stop" come nome del prodotto. Stampare la lista dei prodotti inventariati, con le rispettive quantità, su una sola riga.

Esempio

Input: latte, 2, uova, 6, pane, 7, latte, 3, uova, 6, stop

Output: (latte, 5) (uova, 12) (pane, 7)

Esercizio 3 - Media dei voti

Si scriva un programma che, dato un insieme di studenti con i rispettivi voti di 3 esami, calcoli e visualizzi la media aritmetica dei voti di ciascuno. L'utente fornirà il nome di ogni studente, seguito dai suoi 3 voti. L'acquisizione termina quando l'utente inserisce "stop" come nome di uno studente.

Esempio

Input: Luca, 28, 30, 27, Marta, 30, 29, 30, Marco, 18, 22, 25, stop

Output: Luca: 28.33

Marta: 29.67

Marco: 21.67

Esercitazione su EVO!



- Topic: comandi condizionali, iterativi, e array
- Durata: una settimana (da oggi a mercoledì prossimo)
- Scopo: prendere dimestichezza con la piattaforma (dove svolgerete le prove in itinere) ed iniziare a programmare in autonomia

L'esercitazione non viene valutata, è per vostro allenamento!

<https://evo.di.unipi.it/student/courses/43/exams/3wXj17v>

Ricordatevi di utilizzare le credenziali di ateneo (@studenti.unipi.it)

Q & A