# 02- PROGRAMMAZIONE FUNZIONALE

### Immagine che contiene testo, schermata, Carattere Descrizione generata automaticamenteCALLBACK

Funzione che viene passata in un’altra funzione come argomento.   
A quel punto viene poi invocata dalla funzione stessa per fare operazioni.

Può avere comportamento asincrono o sincrono.

**Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamenteCallback sincrone**: sono eseguite interamente quando vengono richiamate (programma aspetta fine esecuzione)

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

**Filter** → metodo funzionale che accetta un criterio e filtra secondo quello

**Callback asincrone**: programma non aspetta la fine dell’esecuzione

### Programmazione funzionale

Immagine che contiene testo, schermata, Carattere, biglietto da visita

Descrizione generata automaticamenteSviluppatori costruiscono e strutturano il codice principalmente usando funzioni.

Stile dichiarativo → maggior visibilità programma

* Funzioni sono cittadini di prima classe in quanto oggetti
  + Posso farci tante cose
* Le funzioni sono di ordine alto
  + Può operare su altre funzioni

NOTA: si evitano modifiche in place

* Se devo modificare un array, non modifco l’array ma una sua copia
  + Prende come parametro delle funzioni
  + Ritorna una funzione
* Funzioni si possono comporre
* Possono essere chiamate in catena
  + Risultato di una è l’input dell’altra

METODI FUNZIONALI:

* **forEach**(f): processa ogni elemento con callback f
  + Immagine che contiene testo, Carattere, schermata

    Descrizione generata automaticamentechiama la callback una volta per ogni elemento
  + currentValue: valore corrente
  + index: indice del valore corrente
  + array: copia dell’array corrente
  + ritorna un undefined, non contatenabile
    - non posso fare .altro
  + non si può bloccare ad un certo punto del for
* every(f), some(f): verofoca se tutti/alcuni elementi nell’array soddisfano la callback booleana f
  + **every**(f)
    - Immagine che contiene testo, Carattere, schermata

      Descrizione generata automaticamentetrue: se tutti gli elementi superano il test
    - false: se c’è almeno uno che non lo supera
  + **Immagine che contiene testo, Carattere, schermata, linea

    Descrizione generata automaticamentesome**(f):
    - true: se c’è almeno uno che supera il test
    - false: se tutti non superano il test
* **Immagine che contiene testo, Carattere, schermata

  Descrizione generata automaticamentemap**(f) → costruire un nuovo array con i valori tornati dalla callback
  + callback deve restituire qualcosa
* Immagine che contiene testo, Carattere, schermata, ricevuta

  Descrizione generata automaticamente**filter**(f) → crea un nuovo array con tutti gli elementi che passano i test   
    
  implementati dalla funzione
* Immagine che contiene testo, schermata, Carattere

  Descrizione generata automaticamente**reduce**: combina gli elementi di un array, usando la callback specificata
  + **restituisce un valore singolo**
    - si calcola progressivamente un risultato
  + 1 parametro obbligatorio→ callback
    - Dentro la callback ha 2 parametri
      * Accumulatore → risultato accumulato fino a quel momento specifico
  + 1 parametro opzionale → da quale ID partire per fare combinazione elementi
    - Se non passato, usa il primo valore come iniziale e iterazione parte da secondo

**Senza metodi funzionali**

* import dayjs from 'dayjs';
* function Answer(text, username, date, score=0) {
* this.text = text;
* this.username=username;
* this.score=score;
* this.date= dayjs(date);
* this.toString = () => {
* return `${this.username} replied '${this.text}' on ${this.date.format('YYYY-MM-DD')} con score ${this.score}`;
* }
* }
* function Question(text, username, date){
* this.text = text;
* this.username=username;
* this.date=dayjs(date);
* this.answers=[];
* this.add = (answer) => {
* this.answers.push(answer);
* }
* this.find = (username) => {
* const foundAnswers = [];
* for(const ans of this.answers){
* if(ans.username===username)
* foundAnswers.push(ans);
* }
* return foundAnswers;
* }
* this.afterDate = () => { //TO DO
* }
* this.listByDate = () => { //TO DO
* }
* this.listByScore = () => { //TO DO
* }
* }
* const question= new Question('Is JS better then python?', 'Peppe Arbo', '2024-02-27');
* const firstAnswer= new Answer('Yes', 'Pluto Mannella', '2024-02-18', -10);
* const secondAnswer= new Answer('Not in a million year', 'Pino Van Rossen', '2024-03-01', 5);
* const thirdAnswer= new Answer('Yes', 'Albert Einstein', '2024-03-01', 1);
* const fourthAnswer= new Answer('Yes', 'Pluto Mannella', '2024-03-10');
* question.add(firstAnswer);
* question.add(secondAnswer);
* question.add(thirdAnswer);
* question.add(fourthAnswer);
* const answerByPluto = question.find('Pluto Mannella');
* console.log(question);
* console.log("Risposte di Pluto" + answerByPluto);

**con metodi funzionali:**

import dayjs from 'dayjs';

function Answer(text, username, date, score=0) {

    this.text = text;

    this.username=username;

    this.score=score;

    this.date= dayjs(date);

    this.toString = () => {

        return `${this.username} replied '${this.text}' on ${this.date.format('YYYY-MM-DD')} con score ${this.score}`;

    }

}

function Question(text, username, date){

    this.text = text;

    this.username=username;

    this.date=dayjs(date);

    this.answers=[];

    this.add = (answer) => {

        this.answers.push(answer);

    }

    this.find = (username) => {

        return this.answers.filter(answer => answer.username === username);

    }

    this.afterDate = (date) => {

        return this.answers.filter(ans => ans.date.isAfter(dayjs(date)));

    }

    this.listByDate = () => {

        return [...this.answers].sort((a,b) => (a.date.isAfter(b.date) ? 1 : -1)); //quadre aggiunte per fare spread ... in modo tale da non avere un ordinamento in place

    }

    this.listByScore = () => {

        return [...this.answers].sort((a,b) => (b.score-a.score)); //ordine decrescente

    }

}

const question= new Question('Is JS better then python?', 'Peppe Arbo', '2024-02-27');

const firstAnswer= new Answer('Yes', 'Pluto Mannella', '2024-02-18', -10);

const secondAnswer= new Answer('Not in a million year','Pino Van Rossen', '2024-03-01',5);

const thirdAnswer= new Answer('Yes', 'Albert Einstein', '2024-03-01', 1);

const fourthAnswer= new Answer('Yes', 'Pluto Mannella', '2024-03-10');

question.add(firstAnswer);

question.add(secondAnswer);

question.add(thirdAnswer);

question.add(fourthAnswer);

const answerByPluto = question.find('Pluto Mannella');

console.log(question);

console.log("\nRisposte di Pluto" + answerByPluto);

console.log(question.listByDate());

console.log(question.listByScore());

console.log(question.afterDate('2024-02-29'));