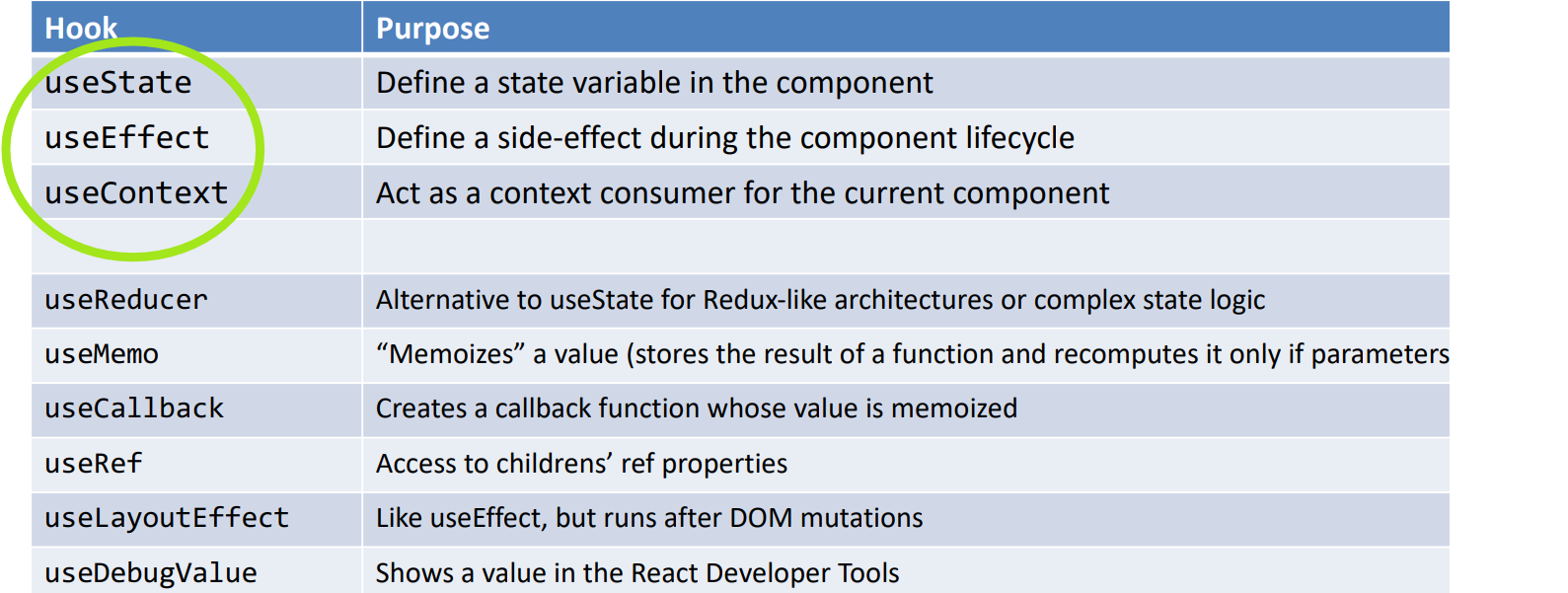
# 04-03 Componenti e stato

Componente funzionale:

* Semplice
* Funzione pura
  + No stato
  + No side effects
* No ciclo di vita
* Può definire handler functions

In React si hanno gli **Hooks** in modo tale da aggirare queste limitazioni in maniera controllata:

* Hook per stato
* Hook per side effect
* Hook per contesto

Bisogna definire un **Hook per ogni funzionalità richiesta**, Hook è una funzione:

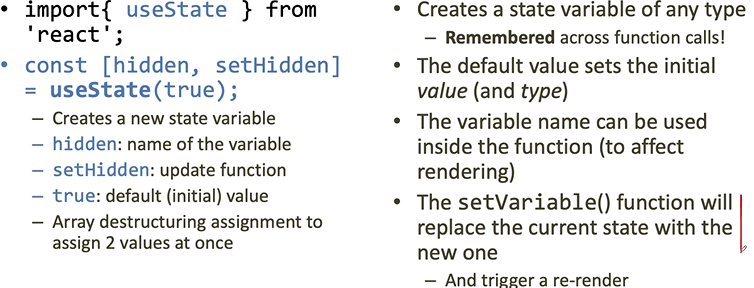
*useState ci serviva per ricordare lo stato per ogni componente.*

### Breve riassunto componenti react:

* Componente react fa parte di un element tree quando viene renderizzato
  + **Props**: porzioni di dati immutabili per passaggio di informazioni da genitori a figli
    - Tramite le props passa determinate informazioni ai componenti figli
    - Riceve delle props, può leggere, visualizzarle, usarle ma non cambiarle
      * ****Read-only
      * ogni attributo in JSX è convertibile in una props
      * Possono essere qualunque oggetto javascript
        + Inclusi elementi react
  + **Contesto**: props globale e implicita, passata automaticamente a tutti i componenti
    - per passare la stessa informazione ± contemporaneamente a tanti componenti
      * Es: cambio di lingua, devo passarlo a tutti i componenti che hanno del testo
      * Es: dark mode
  + **Stato**: dove il componente salva le proprie informazioni localmente
    - se cambia lo stato 🡪 componente si ri-renderizza
    - Può essere letto e modificato solo dal componente stesso
    - Vive dentro il componente
      * Si può mandare il valore attuale dello stato di un componente tramite props agli altri componenti

### Hook useState

Variabile [nomeStato, nomeFunzionePerSetStato]

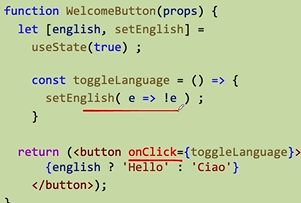
* setStato → rimpiazza l’intero stato
  + conseguente re-rendering
  + Applica modifiche in maniera asincrona



Nota: usare sempre la set, non modificare mai a mano

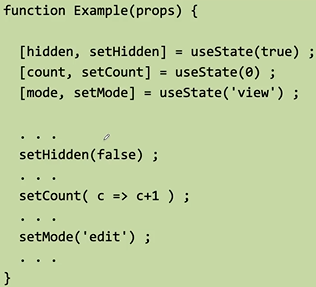
### Aggiornamento dello stato:

* Usando nuovo valore
  + Dipende da props e valori costanti
  + Rimpiazza il corrente
  + Deve avere lo stesso tipo
* Con una funzione (callback) → se la logica dipende dalla precedente
  + Esegue callback
  + Nuovo stato dipendente da vecchio
  + Il valore di ritorno della funzione rimpiazzerà il nuovo stato
    - Deve ritornare un nuovo valore di stato
    - Non deve cambiare, deve tornare uno nuovo

****Cambi di stato sono spesso determinati **da eventi asincroni:**

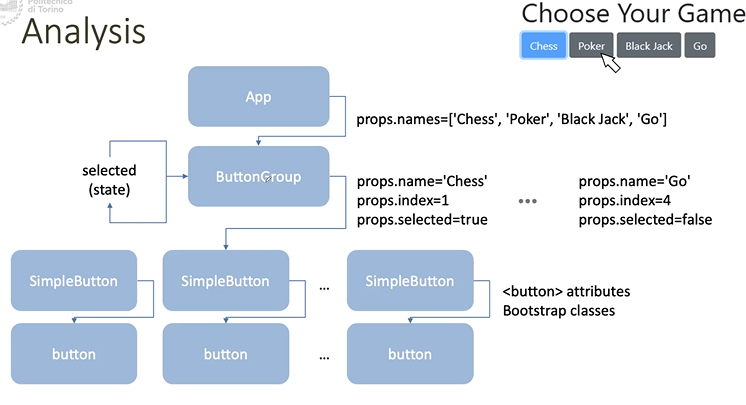
* Gestore eventi DOM (onClick)
* Risposta server (api calls)   
  oppure

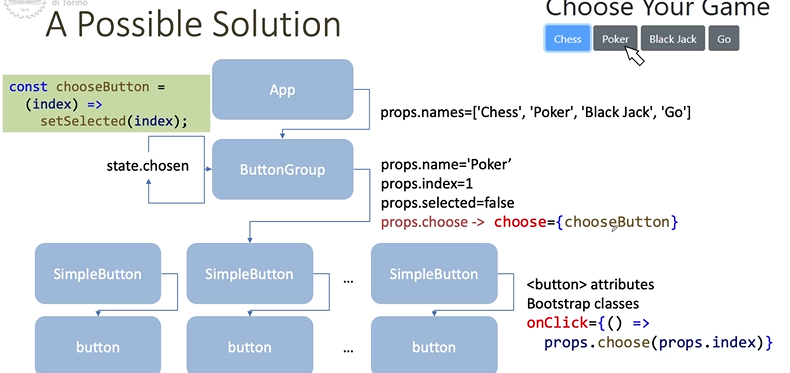
Valore di default (true), viene usato solo al primo rendering del componente (compreso refresh).

Un componente può **avere più stati**, indicando nomi diversi → da fare se ci sono proprietà non collegate tra loro

* + - * Componente si renderizza se uno dei suoi stati cambia

#### Componente figlio vuole cambiare stato padre



Ogni bottone deve avere una reference al metodo per cambiare lo stato del padre.

**Nota generale**: quando possibile è meglio creare componenti senza stato in quanto riutilizzabili e più veloci. Quello che si fa è portare lo stato più in alto (all’antenato più comune) in modo tale da avere lobstato sul padre e i figli possono leggere con le props, possono aggionrare con le callbacks.