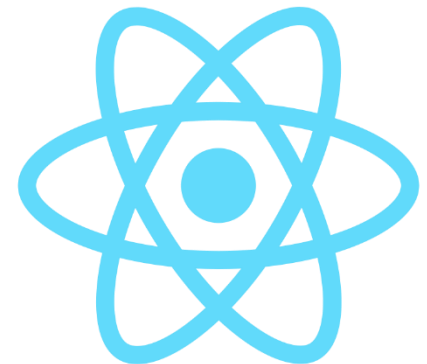


React

Riccardo Cattaneo



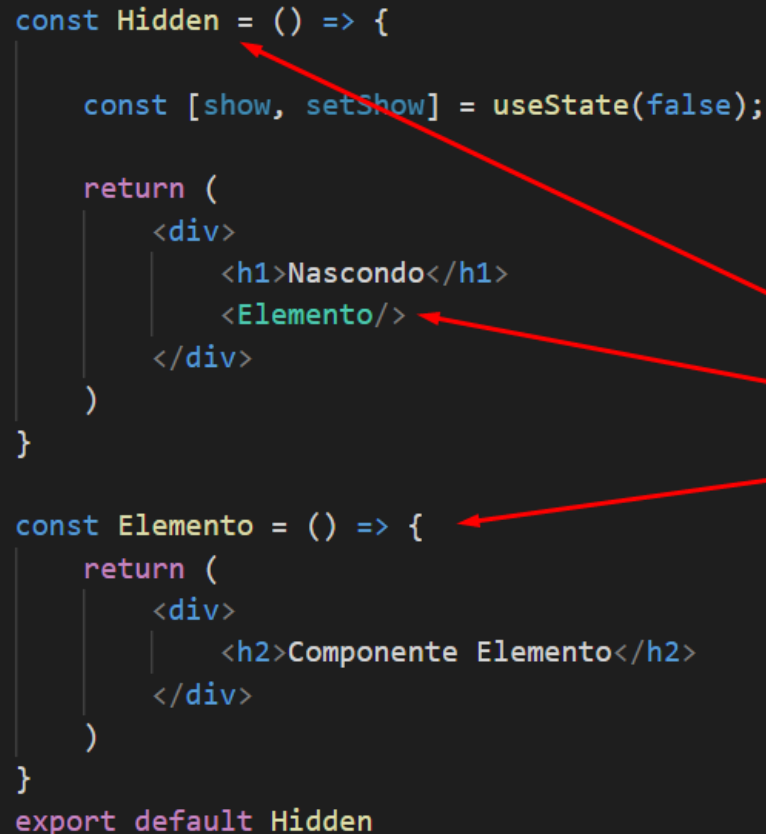
Nascondere Componente

Andiamo a vedere come possiamo nascondere e visualizzare un componente a seconda dello state di una variabile. Ad esempio se abbiamo una situazione del genere :

```
const Hidden = () => {  
  const [show, setShow] = useState(false);  
  
  return (  
    <div>  
      <h1>Nascondo</h1>  
    </div>  
  )  
}
```


Possiamo ora andare a creare un nuovo componente che utilizzeremo all'interno del componente appena creato in questo modo :

```
const Hidden = () => {  
  const [show, setShow] = useState(false);  
  
  return (  
    <div>  
      <h1>Nascondi</h1>  
      <Elemento/>  
    </div>  
  )  
}  
  
const Elemento = () => {  
  return (  
    <div>  
      <h2>Componente Elemento</h2>  
    </div>  
  )  
}  
  
export default Hidden
```



Nel componente principale andiamo ora a creare un bottone che al verificarsi l'evento onClick andrà a settare il valore di show a false se era vero, e vero se era falso, in questo modo :

```
const Hidden = () => {  
  
  const [show, setShow] = useState(false);  
  
  return (  
    <div>  
      <h1>Nascondo</h1>  
      <button onClick={() => setShow(!show)}> Visualizza / Nascondi </button>  
      <Elemento/>  
    </div>  
  )  
}
```



Operatore Ternario

Andiamo ora a vedere come sia possibile «cambiare» la visualizzazione di un componente in base al valore di una variabile.

Ci viene naturale pensare che per risolvere il problema, potremmo utilizzare un classico IF, ma ricordiamoci che all'interno del rendering di un componente stiamo utilizzando **JSX e questo linguaggio non ci consente di utilizzare costrutti condizionali.**

Quindi ci viene in aiuto **l'operatore ternario**, che è una scorciatoia di un costrutto if che a differenza di una if **ritorna sempre un valore**. Questa è la sintassi :


```
nomeDaVerificare === 'valore' ? 'sono vero' : 'sono falso'
```

Se invece la variabile è un booleano basta mettere il nome della variabile :

```
nomeDaVerificare ? 'sono vero' : 'sono falso'
```

Per riprendere il nostro esempio possiamo modificare il bottone che cambia il suo valore in base al valore della variabile show in questo modo :

```
return (  
  <div>  
    <h1>Nascondo</h1>  
    <button onClick={() => setShow(!show)}> {  
      show ? 'Nascondi' : 'Visualizza'  
    } </button>  
    <Elemento/>  
  </div>  
)
```



Short Circuit Evaluation

Una short circuit evaluation è un'espressione che torna un valore in base allo stato di una costante o di una variabile che vogliamo andare a valutare. Esistono 2 tipi di short circuit :

And &&

Or ||

||

Esegue il controllo dell'espressione, se è vuota prende il secondo valore, **altrimenti** il primo :

const esempio = parola || "paperino"

Se la variabile parola ha un valore, allora esempio assume il valore di parola, altrimenti assume il valore "paperino".

&&

Esegue il controllo dell'espressione, se è vera prende il secondo valore :

const esempio = parola && "paperino"

Se la variabile parola ha un valore, allora esempio assume il valore di "paperino".

Quindi ritornando al nostro esempio posso utilizzare lo Short Circuit && per visualizzare o meno il componente Elemento in base al valore di show :

```
const Hidden = () => {  
  const [show, setShow] = useState(false);  
  
  return (  
    <div>  
      <h1>Nascondo</h1>  
      <button onClick={() => setShow(!show)}> {  
        show ? 'Nascondi' : 'Visualizza'  
      } </button>  
  
      {show && <Elemento/>}  
    </div>  
  )  
}
```



Esercizio

Creare un componente con un elenco di News composte da id, titolo e descrizione. Salvare in un file esterno 5 news con testo a piacimento ed all'interno del componente elencarle. Aggiungere un bottone finale che cambia il colore del font e dello sfondo di tutta la pagina. Situazione iniziale sfondo bianco scritte nere. Se clicco sul bottone sfondo nero e scritte bianche. Ad ogni click invertire nuovamente i colori.

Aggiungere bottone Nascondi / Visualizza News