



Gestione dello Stato e Comunicazione tra Componenti



# REACT AVANZATO




# Gestione dei Componenti






# Componenti e Stato Interno

- Obiettivo del giorno
    - Esplorare
      - Componenti
      - Gestione stati
      - Comunicazione e gestione degli eventi
- 



# Componenti

- Blocchi di base di un'interfaccia utente
  - Tipi di componenti:
    - Funzionali
    - Classi estensione di componenti del framework
- 

# Componenti Funzionali

```
import React from "react";

const Greeting: React.FC = () => {
  return (
    <div>
      <h1>Ciao, Benvenuto alla lezione!</h1>
    </div>
  );
};

export default Greeting;
```

# Componenti in Classe

```
import React, { Component } from "react";

class Greeting extends Component {
  render() {
    return (
      <div>
        <h1>Ciao, Benvenuto alla lezione!</h1>
      </div>
    );
  }
}

export default Greeting;
```



# Stato Interno

- Ogni componente gestisce delle informazioni che devono essere presentate nella UI o che servono alla presentazione di altri dati
  - Di ogni informazione deve essere possibile comprendere le modifiche al fine di adattare la UI
    - Quindi occorre sapere se e come cambiano al fine di riflettere tali modifiche nella UI

# Stato Interno

- La gestione dello stato interno avviene attraverso l'affidamento di una variabile "di stato" ad uno hook di sistema che rende possibili le "osservazioni" dei cambiamenti
  - `useState()`
    - Restituisce un oggetto complesso, in genere destrutturato in una tupla, che contiene la variabile di stato e il metodo utilizzabile per implementare una sua modifica



# Stato Interno

```
const Counter: React.FC = () => {  
  // Dichiarazione dello stato  
  const [count, setCount] = useState<number>(0);  
  
  // Funzione per incrementare il contatore  
  const handleIncrement = () => {  
    setCount(count + 1);  
  };  
  
  return (  
    <div>  
      <h1>Contatore: {count}</h1>  
      <button onClick={handleIncrement}>Incrementa</button>  
    </div>  
  );  
};
```

# Stato Interno


- `const [count, setCount] = useState<number>(0)`
  - Inizializza lo stato con il valore `0`
    - Restituisce un riferimento alla variabile `count` della quale il componente terrà traccia relativamente ai cambiamenti
    - Restituisce un riferimento al metodo per apportare delle modifiche
      - `setCount(nuovo_valore)`
      - `setCount(vecchio_valore => nuovo_valore)`

# Effetti Collaterali

- `useEffect()`
  - Hook che consente di gestire effetti collaterali nei componenti funzionali
    - Chiamate API, interazione con il DOM, gestione dei eventi
  - Viene eseguito
    - Al montaggio di un componente
    - O alla modifica di una “dipendenza”



# Effetti Collaterali

- Dipendenze
    - Secondo parametro della funzione
      - Array vuoto (`[]`) se si intende eseguire l'effetto solo al montaggio del componente nella vista
      - Array di oggetti dipendenti se si intende eseguire l'effetto al cambiamento di stato di uno di essi
  - Cleanup
    - Restituisce una funzione che viene eseguita quando il componente viene smontato
- 




# Comunicazione tra Componenti

- Strategie
    - Tramite **props**
    - Tramite **Context API** o **Redux**
      - Non trattati in questa sezione del corso
- 



# Context API

- Provider
    - Incapsula i componenti che hanno bisogno dell'accesso al contesto, fornendo lo stato condiviso
  - Hook personalizzato
    - Migliora l'ergonomia dell'uso del contesto, evitando controlli manuali
  - Lettura dati
    - Accede al valore condiviso senza dover passare props
  - Aggiornamento dati
    - Modifica lo stato condiviso usando una funzione del provider
- 

# Event Handling

- Gestione degli eventi (UI o di sistema)
  - Ogni componente prevede degli “handler” il cui nome inizia con il prefisso **on**
- Ogni evento prevede un parametro tipizzato dal framework che porta con sé le informazioni sull'evento

```
<button onClick={handleClick}>Click me!</button>
```

```
const InputExample: React.FC = () => {  
  const [inputValue, setInputValue] = useState<string>("");  
  
  // Gestione del cambiamento dell'input  
  const handleChange = (event: React.ChangeEvent<HTMLInputElement>): void  
    => {  
    setInputValue(event.target.value);  
  };  
  
  return (  
    <div>  
      <label>  
        Inserisci il tuo nome:  
      </label>  
      <input type="text" value={inputValue} onChange={handleChange} />  
    </div>  
  );  
};
```



# Domande e Discussione



Grazie per l'attenzione!