REACT

ITS 2024-25

LE BASI

```html
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0">
    <title>Document</title>
  </head>
  <body>
    <div class="content">Content</div>
  </body>
</html>
```

```css
body {
  padding: 0;
  margin: 0;
  background: #fff;
}

.content {
  max-width: 900px;
  margin: 0 auto;
}
```

```
const el = document.querySelector('.content');

el.innerHTML = "Contenuto aggiornato";
```

- `innerHTML` è altamente sconsigliato per ragioni di **sicurezza**

OBIETTIVI

- modificare il contenuto della pagina
- reagire agli eventi che si verificano (es: click)

REACT

- sintassi facile e **sicura** per scrivere HTML usando javascript
- **reactive**: permette di modificare l'HTML in seguito ad eventi

JSX

- sintassi simile ad HTML
- in realtà è javascript “mascherato”

```
<MyComponent textVariable="Contenuto da inserire" />
```

- Componente con l’iniziale maiuscola
- attributi `camelCase`
- gli attributi possono contenere anche numeri, booleani, funzioni, array…
  - *(in HTML gli attributi sono sempre stringhe)*

```
function MyComponent(props) {
  return (
    <div className="content">
      {props.textVariable}
    </div>
  )
}
```

- gli attributi in react si chiamano **props**

```
const Page = () => (
  <Layout>
    <Header>
      <Logo img="./company.png" />
      <Navigation activePage="home" />
      <Search />
    </Header>
    <Container>
      <MyComponent textVariable="Contenuto principale" />
    </Container>
  </Layout>
);
```

- composition

REACTIVE

```
<Search
  onSearch={async (searchQuery) => {
    const searchResults = await
      fetch('http://api.com/search?' + searchQuery);
    setContent(searchResults);
  }}
/>
```

- le funzioni sono first class objects
- posso passare una funzione come parametro di un’altra

```
<Search
  onSearch={async (searchQuery) => {
    const searchResults = await
      fetch('http://api.com/search?' + searchQuery);
    setContent(searchResults);
  }}
/>
```

- le funzioni sono first class objects
- posso passare una funzione come parametro di un’altra

```
const Search = ({ onSearch }) => {
  return (
    <form onSubmit={ () => onSearch(what) }>
      Cerca:
      <input type="text" name="what" />
    </form>
  )
}
```

pseudo-code: il form non funziona così

```
const Search = ({ onSearch }) => {
  return (
    <form onSubmit={ () => onSearch(what) }>
      Cerca:
      <input type="text" name="what" />
    </form>
  )
}
```

pseudo-code: il form non funziona così

```
1 const Search = ({ onSearch }) => {
2   return (
3     <form onSubmit={ () => onSearch(what) }>
4       Cerca:
5       <input type="text" name="what" />
6     </form>
7   )
8 }
```

pseudo-code: il form non funziona così

```
<Search
  onSearch={async (searchQuery) => {
    const searchResults = await
      fetch('http://api.com/search?' + searchQuery);
    setContent(searchResults);
  }}
/>
```

```
<Search
  onSearch={async (searchQuery) => {
    const searchResults = await
      fetch('http://api.com/search?' + searchQuery);
    setContent(searchResults);
  }}
/>
```

```
const Page = () => {
  const [content, setContent] = useState("Contenuto inizial
  return (
    <Layout>
        <Search onSearch={async (searchQuery) => {
          const searchResults = await
            fetch('http://api.com/search?' + searchQuery);
          setContent(searchResults);
        }}/>
      </Header>
      <Container>
        <MyComponent textVariable={content} />
      </Container>
    </Layout>
  );
```

```
const Page = () => {
  const [content, setContent] = useState("Contenuto inizial
  return (
    <Layout>
        <Search onSearch={async (searchQuery) => {
          const searchResults = await
            fetch('http://api.com/search?' + searchQuery);
          setContent(searchResults);
        }}/>
      </Header>
      <Container>
        <MyComponent textVariable={content} />
      </Container>
    </Layout>
  );
```

```
const Page = () => {
  const [content, setContent] = useState("Contenuto inizial
  return (
    <Layout>
        <Search onSearch={async (searchQuery) => {
          const searchResults = await
            fetch('http://api.com/search?' + searchQuery);
          setContent(searchResults);
        }}/>
      </Header>
      <Container>
        <MyComponent textVariable={content} />
      </Container>
    </Layout>
  );
```

```
const Page = () => {
  const [content, setContent] = useState("Contenuto inizial
  return (
    <Layout>
        <Search onSearch={async (searchQuery) => {
          const searchResults = await
            fetch('http://api.com/search?' + searchQuery);
          setContent(searchResults);
        }}/>
      </Header>
      <Container>
        <MyComponent textVariable={content} />
      </Container>
    </Layout>
  );
```

```
2    const [content, setContent] = useState("Contenuto iniziale
3    return (
4      <Layout>
5          <Search onSearch={async (searchQuery) => {
6            const searchResults = await
7              fetch('http://api.com/search?' + searchQuery);
8            setContent(searchResults);
9          }}/>
10       </Header>
11       <Container>
12         <MyComponent textVariable={content} />
13       </Container>
14     </Layout>
15   );
16 }
```

APPROFONDIMENTI

- React.dev

TIPS

```
<div>
  {isLoggedIn ? (
    <AdminPanel />
  ) : (
    <LoginForm />
  )}
</div>
```

- operatori ternari per il conditional rendering

```
<div>
  {isLoggedIn && <AdminPanel />}
</div>
```

- se non ti serve `else`, usa l'operatore logico &&

```
const listItems = products.map(product =>
  <li key={product.id}>
    {product.title}
  </li>
);

return (
  <ul>{listItems}</ul>
);
```

- usa le funzioni degli array

```jsx
function MyButton() {
  const [count, setCount] = useState(0);
  return (
    <button onClick={() => setCount(count + 1)}>
      Clicked {count} times
    </button>
  );
}

export default function MyApp() {
  return (
    <MyButton />
    <MyButton />
  );
}
```

- ogni componente ha il suo stato

```
function MyButton({count, setCount}) {
  return (
    <button onClick={() => setCount(count + 1)}>
      Clicked {count} times
    </button>
  );
}

export default function MyApp() {
  const [count, setCount] = useState(0);
  return (
    <MyButton count={count}, setCount={setCount}/>
    <MyButton count={count}, setCount={setCount}/>
  );
}
```

- stato condiviso tra più componenti

ERRORI DA EVITARE

```
function MyButton() {
  const [count, setCount] = useState(0);
  return (
    <button onClick={() => { count = count + 1 }}> // NO!
      Clicked {count} times
    </button>
  );
}
```

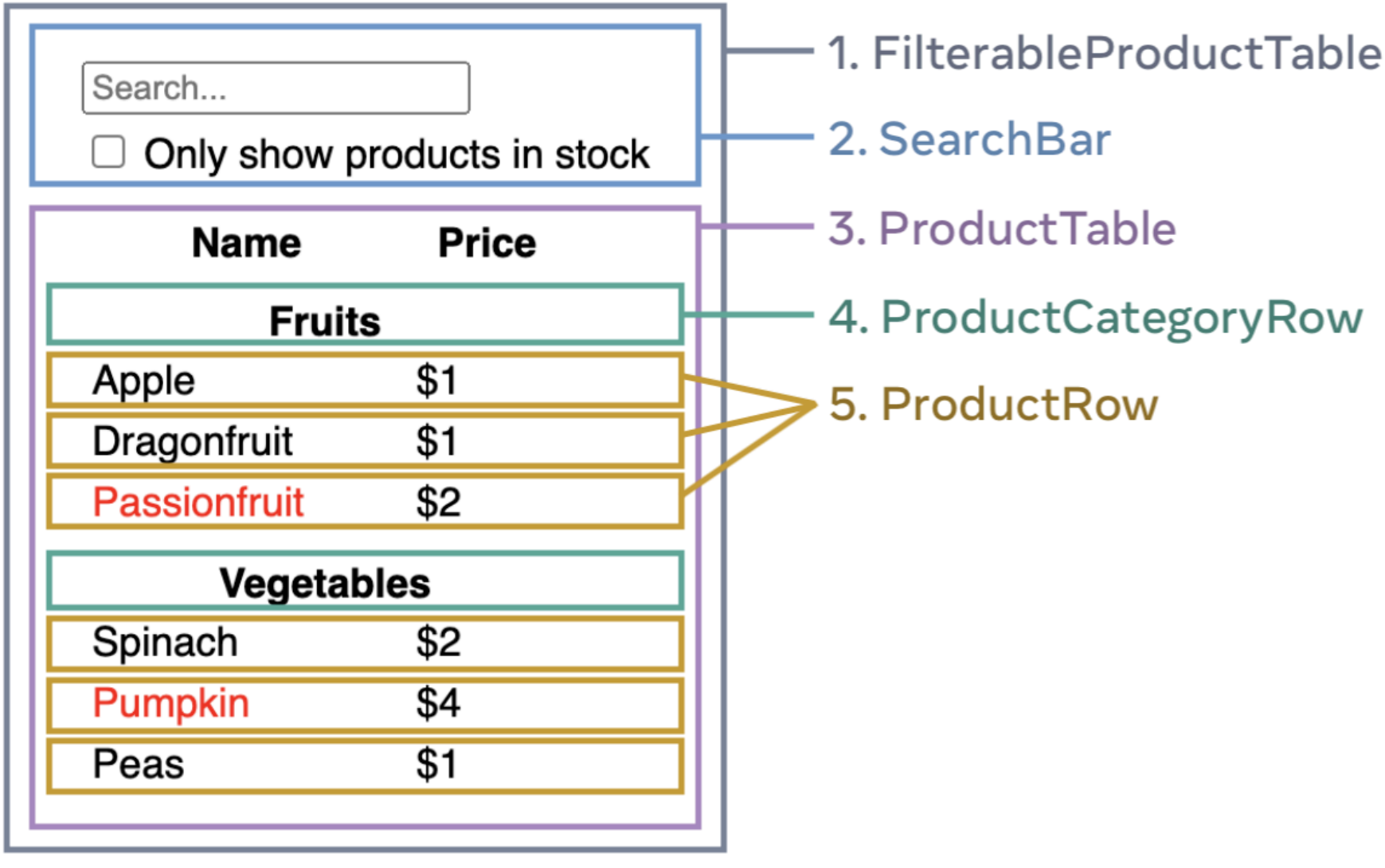
- **mai** modificare lo stato direttamente, utilizza sempre il `setState` (qui chiamato `setCount`)

```
function MyButton({count}) {
  return (
    <button onClick={() => { count = count + 1 }}> // NO!
      Clicked {count} times
    </button>
  );
}
```

- **rispetta** il flusso di dati, sempre dall'alto verso il basso. Non modificare una proprietà di un componente di livello superiore (in questo esempio: una prop)

THINKING IN REACT

React.dev


Search...
Only show products in stock
Name
Price
Fruits
Apple
$1
Dragonfruit
$1
Passionfruit
$2
Vegetables
Spinach
$2
Pumpkin
$4
Peas
$1
1. FilterableProductTable
2. SearchBar
3. ProductTable
4. ProductCategoryRow
5. ProductRow

- disegna una **gerarchia di componenti**

- costruisci una versione **statica** in React

- progetta una rappresentazione minima ma completa dello stato
- identifica dove implementare lo stato
  - quali componenti devono gestire lo stato?
  - qual è il primo genitore comune di due componenti che devono condividere lo stesso stato?
  - **lift state up**

- implementa le prop che contengono le funzioni di callback (**inverse data flow**)

INIZIALIZZARE L'APPLICAZIONE

```html
<html>
  <head></head>
  <body>
    <div id="app"></div>
  </body>
</html>
```

```
import { createRoot } from 'react-dom/client';
import { Page } from './Page.jsx'
const root = createRoot(document.getElementById('app'));
root.render(<Page />);
```

LIBRERIE

Utilizzare una libreria di componenti (UI Kit): MUI

```
export const Header: React.FC = () => {
  return (
    <Stack
      direction="row"
      spacing={2}
      sx={{
        justifyContent: "space-between",
        alignItems: "center",
        paddingTop: "16px",
      }}
    >
      <img src="/logo.svg" height={50} />
      <Stack direction="row" spacing={2}>
        <List role="menubar" orientation="horizontal">
          <ListItem role="none">
```

Utilizzare una libreria per il routing: React Router

```
function App() {
  return (
    <BrowserRouter basename="/">
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Root />} />

          {/* Teacher */}
          <Route path="teacher" element={<TeacherGuard />}>
            <Route index element={<Exams />} />
            <Route path="exam" element={<AddExam />} />
            <Route path="exam/:id" element={<EditExam />} />
            <Route path="subscriptions" element={<Subscription
            <Route path="subscriptions/:date" element={<Sessio
            <Route
```